SHUBHENDU TRIVEDI

DISCRIMINATIVE LEARNING OF SIMILARITY AND GROUP EQUIVARIANT REPRESENTATIONS

DISCRIMINATIVE LEARNING OF SIMILARITY AND GROUP EQUIVARIANT REPRESENTATIONS

SHUBHENDU TRIVEDI

PhD Thesis

August 2018

— Dissertation Committee —

DISCRIMINATIVE LEARNING OF SIMILARITY AND GROUP EQUIVARIANT REPRESENTATIONS

A thesis presented
by
SHUBHENDU TRIVEDI

in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science.
Toyota Technological Institute at Chicago
Chicago, Illinois
August, 2018

— Thesis Committee —

Dr. Kevin Gimpel		
Committee member	Signature	Date
Dr. Risi Kondor		
Committee member	Signature	Date
Dr. Brian D. Nord		
Committee member	Signature	Date
Dr. Gregory Shakhnarovich		
Thesis/Research Advisor	Signature	Date
Dr. Avrim Blum		
Chief Academic Officer	Signature	Date

DISCRIMINATIVE LEARNING OF SIMILARITY AND GROUP EQUIVARIANT REPRESENTATIONS

A thesis presented
by
SHUBHENDU TRIVEDI

in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science.
Toyota Technological Institute at Chicago
Chicago, Illinois
August, 2018

— Thesis Committee —

Dr. Kevin Gimpel	[illegible]	8/16/18
Committee member	Signature	Date
Dr. Risi Kondor	[illegible]	8/16/18
Committee member	Signature	Date
Dr. Brian D. Nord	[illegible]	8/16/18
Committee member	Signature	Date
Dr. Gregory Shakhnarovich	[illegible]	8/16/18
Thesis/Research Advisor	Signature	Date
Dr. Avrim Blum	[illegible]	8/21/18
Chief Academic Officer	Signature	Date

... for (and in veneration of) my loving parents: Smt. Jyotsna Trivedi and Shri M. L. Trivedi.

... to the memory of my grandfather: Shri R. C. Trivedi.

... for one of my dearest friends: Babar Majeed Saggu.

... and finally to: M.

FRANZ KAFKA: BEFORE THE LAW

Before the law sits a gatekeeper. To this gatekeeper comes a man from the country who asks to gain entry into the law. But the gatekeeper says that he cannot grant him entry at the moment. The man thinks about it and then asks if he will be allowed to come in later on. "It is possible," says the gatekeeper, "but not now." At the moment the gate to the law stands open, as always, and the gatekeeper walks to the side, so the man bends over in order to see through the gate into the inside. When the gatekeeper notices that, he laughs and says: "If it tempts you so much, try it in spite of my prohibition. But take note: I am powerful. And I am only the most lowly gatekeeper. But from room to room stand gatekeepers, each more powerful than the other. I can't endure even one glimpse of the third." The man from the country has not expected such difficulties: the law should always be accessible for everyone, he thinks, but as he now looks more closely at the gatekeeper in his fur coat, at his large pointed nose and his long, thin, black Tartar's beard, he decides that it would be better to wait until he gets permission to go inside. The gatekeeper gives him a stool and allows him to sit down at the side in front of the gate. There he sits for days and years. He makes many attempts to be let in, and he wears the gatekeeper out with his requests. The gatekeeper often interrogates him briefly, questioning him about his homeland and many other things, but they are indifferent questions, the kind great men put, and at the end he always tells him once more that he cannot let him inside yet. The man, who has equipped himself with many things for his journey, spends everything, no matter how valuable, to win over the gatekeeper. The latter takes it all but, as he does so, says, "I am taking this only so that you do not think you have failed to do anything." During the many years the man observes the gatekeeper almost continuously. He forgets the other gatekeepers, and this one seems to him the only obstacle for entry into the law. He curses the unlucky circumstance, in the first years thoughtlessly and out loud, later, as he grows old, he still mumbles to himself. He becomes childish and, since in the long years studying the gatekeeper he has come to know the fleas in his fur collar, he even asks the fleas to help him persuade the gatekeeper. Finally his eyesight grows weak, and he does not know whether things are really darker around him or whether his eyes are merely deceiving him. But he recognizes now in the darkness an illumination which breaks inextinguishably out of the gateway to the law. Now he no longer has much time to live. Before his death he gathers in his head all his experiences of the entire time up into one question which he has not yet put to the gatekeeper. He waves to him, since he can no longer lift up his stiffening body. The gatekeeper has to bend way down to him, for the great difference has changed things to the disadvantage of the man. "What do you still want to know, then?" asks the gatekeeper. "You are insatiable." "Everyone strives after the law," says the man, "so how is that in these many years no one except me has requested entry?" The gatekeeper sees that the man is already dying and, in order to reach his diminishing sense of hearing, he shouts at him, "Here no one else can gain entry, since this entrance was assigned only to you. I'm going now to close it."

[Trans. by Ian Johnston. Here, law might originate from the Hebrew word Torah, thus also having the meaning truth]

ABSTRACT

One of the most fundamental problems in machine learning is to compare examples: Given a pair of objects we want to return a value which indicates degree of (dis)similarity. Similarity is often task specific, and pre-defined distances can perform poorly, leading to work in metric learning. However, being able to learn a similarity-sensitive distance function also presupposes access to a rich, discriminative representation for the objects at hand. In this dissertation we present contributions towards both ends. In the first part of the thesis, assuming good representations for the data, we present a formulation for metric learning that makes a more direct attempt to optimize for the k-NN accuracy as compared to prior work. Our approach considers the choice of k neighbors as a discrete valued latent variable, and casts the metric learning problem as a large margin structured prediction problem. We present experiments comparing to a suite of popular metric learning methods. We also present extensions of this formulation to metric learning for kNN regression, and discriminative learning of Hamming distance. In the second part, we consider a situation where we are on a limited computational budget i.e. optimizing over a space of possible metrics would be infeasible, but access to a label aware distance metric is still desirable. We present a simple, and computationally inexpensive approach for estimating a well motivated metric that relies only on gradient estimates, we also discuss theoretical as well as experimental results of using this approach in regression and multiclass settings. In the final part, we address representational issues, considering group equivariant neural networks (GCNNs). Equivariance to symmetry transformations is explicitly encoded in GCNNs; a classical CNN being the simplest example. Following recent work by Kondor et. al., we present a SO(3)-equivariant neural network architecture for spherical data, that operates entirely in Fourier space, while using tensor products and the Clebsch-Gordan decomposition as the only source of non-linearity. We report strong experimental results, and emphasize the wider applicability of our approach, in that it also provides a formalism for the design of fully Fourier neural networks that are equivariant to the action of any continuous compact group.

Thesis Advisor: Gregory Shakhnarovich

Title: Associate Professor

PUBLICATIONS

The ideas in this thesis have appeared (or are about to appear) in the following publications, pre-prints and technical reports

[1] Shubhendu Trivedi, David McAllester, and Gregory Shakhnarovich. "Discriminative Metric Learning by Neighborhood Gerrymandering." In: *Advances in Neural Processing Systems*. 2014, pp. 3392–3400

[2] Shubhendu Trivedi, Jialei Wang, Samory Kpotufe, and Gregory Shakhnarovich. "A Consistent Estimator of the Expected Gradient Outerproduct." In: *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence*. AUAI. 2014, pp. 819–828.

[3] Shubhendu Trivedi and Jialei Wang. "The Expected Jacobian Outerproduct" Preprint. 2018.

[4] Risi Kondor, Shubhendu Trivedi, and Zhen Lin. "A Fully Fourier Space Spherical Convolutional Neural Network based on Clebsch-Gordan Transforms." *Provisional US patent application*, 2018.

[5] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. "Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network." arXiv:1806.09231, Pre-print, 2018.

The following publications, pre-prints and technical reports that the dissertation author was also a contributor in (as a result of work initiated after January 2013), but are not part of this dissertation.

[6] Fei Song, Shubhendu Trivedi, Yutao Wang, Gábor N. Sárközy, and Neil T. Heffernan. "Applying Clustering to the Problem of Predicting Retention within an ITS: Comparing Regularity Clustering with Traditional Methods." In: *Proceedings of the 26th AAAI FLAIRS Conference*. 2013, pp. 527–532

[7] Risi Kondor, Truong Hy Song, Horace Pan, Brandon M. Anderson, and Shubhendu Trivedi. "Covariant compositional networks for learning graphs." arXiv:1801.02144, Pre-print, 2018.

[8] Truong Son Hy, Shubhendu Trivedi, Horace Pan, Brandon M. Anderson, and Risi Kondor. "Predicting molecular properties with covariant compositional networks." In: *The Journal of Chemical Physics* 148.24 (2018), p. 241745.

[9] Risi Kondor and Shubhendu Trivedi. "On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups." In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 2747–2755.

[10] Rohit Nagpal and Shubhendu Trivedi. "A Module-Theoretic Perspective on Equivariant Steerable Convolutional Neural Networks", Pre-print, 2018.

[11] Joao Caldeira, W. L. Kimmy Wu, Brian D. Nord, Camille Avestruz, Shubhendu Trivedi, and Kyle T. Story. "DeepCMB: Lensing Reconstruction of the Cosmic Microwave Background with Deep Neural Networks", Pre-print, 2018.

[12] Zhen Lin, Nick D. Huang, W. L. Kimmy Wu, Brian D. Nord, and Shubhendu Trivedi. "DeepCMB: Classification of Sunyaev-Zel'dovich Clusters in Millimeter Wave Maps using Deep Learning", Pre-print, 2018.

CREDIT ASSIGNMENT

1. Work presented in chapter 3 was joint work with Gregory Shakhnarovich and David McAllester. G. Shakhnarovich was the primary contributor in an earlier iteration of the work presented. The latent structural SVM formulation was originally due to D. McAllester and G. Shakhnarovich. The dissertation author was the primary contributor in later iterations, and contributed ideas, proposed inference procedures, refinements, carried out experiments and contributed to the write-up. Some of the sections and figures in chapter 3 are excerpted directly from the following report: Shubhendu Trivedi, David McAllester, and Gregory Shakhnarovich. "Discriminative Metric Learning by Neighborhood Gerrymandering." In: *Advances in Neural Processing Systems*. 2014, pp. 3392–3400

2. Research presented in sections 4.1 and 4.2 was joint work with Behnam Neyshabur and Gregory Shakhnarovich. The idea of using asymmetry is due to B. Neyshabur. The dissertation author was the primary contributor and contributed ideas, did the experimental evaluation as well as the complete write-up.

3. Work presented in section 4.3 was joint work with Gregory Shakhnarovich. The dissertation author was the primary contributor in all aspects of the presented work.

4. Work presented in chapter 6 was joint work with Jialei Wang, Samory Kpotufe and Gregory Shakhnarovich. The dissertation author initiated the project with S. Kpotufe and G. Shakhnarovich. The idea of using the expected gradient outer product is due to G. Shakhnarovich. J. Wang and S. Kpotufe were the primary contributors in the theoretical analysis. The dissertation author was the primary contributor in the experimental evaluation, as well as contributed ideas for the theoretical analysis and did part of the write-up. Some of the text and figures in chapter 6 are excerpted directly from the following report: Shubhendu Trivedi, Jialei Wang, Samory Kpotufe, and Gregory Shakhnarovich. “A Consistent Estimator of the Expected Gradient Outerproduct.” In: *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence*. AUAI. 2014, pp. 819–828.

5. Work on the expected Jacobian outer product presented in chapter 7 was joint with Jialei Wang. The dissertation author was the primary contributor (jointly with J. Wang) in all aspects of the presented work and contributed to the theoretical analysis, did the experimental evaluation and did the complete write-up. The work also involved inputs by S. Kpotufe.

6. Work presented in chapter 8 was joint work with Risi Kondor and Zhen Lin. The presented work is a direct consequence of a theoretical result (not part of the dissertation) that appeared in the following publication: Risi Kondor and Shubhendu Trivedi. “On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups.” In:*Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 2747–2755. The idea of using

the Clebsch-Gordan transform is due to R. Kondor. The dissertation author was one of the primary contributors (jointly with R. Kondor and Z. Lin) and contributed ideas, the experimental evaluation and contributed to part of the write up. The text appearing in section 8.8 is wholly excerpted from the following report: Risi Kondor, Zhen Lin, and Shubhendu Trivedi. “Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network.” arXiv:1806.09231, Pre-print, 2018.

ACKNOWLEDGMENTS

It feels mildly disappointing to write this section *ex post facto*; particularly in the anticlimactic aftertaste following the very brief but intense period of frenzy that went into putting this dissertation together. Nevertheless it is making me reflect on this journey and my time in Chicago. I came to Chicago and to TTI after a fulfilling and productive random walk, but soon enough, within a year, a combination of a lack of preparedness as well as a couple of extremely unusual personal events soon threatened to turn it into a nightmare. Wherefore, it gives me satisfaction that it turned to be a remarkable, intellectually stimulating and uplifting *personal* experience. Surely, graduate school is not supposed to be easy for anyone, by definition and by design, and it might seem like an exercise in *cheap* vanity to say that personal circumstance made it much harder than it ought to have been. What I intend to convey is that though I put a lot of sweat into this thesis, yet by itself, it does not mean *anything* to me. Indeed, a few months here and there, and given the frenetic activity and pace, it just might have appeared completely different in character and in form, or even in its topic of focus. What is important to me is what the journey has taught me in its wake, and like most good journeys, the best parts of it:

Teach us to care and not to care
Teach us to sit still[1]

Therefore, I will use this section to express my gratitude to everyone who has played a major part in it. I did wonder for a while if I were not being indulgent, immodest, or giving a supposedly common experience too much weight, thus flying in the face of my alleged avowal to stoicism. I apologize for breaking tradition and not keeping it the right measure of impersonal and stolid. I also apologize for its length, however, my closest friends, if they were to read it would understand why.

I will begin with my advisor: Gregory Shakhnarovich. I think it would be preposterous to attempt to thank Greg for all that he has done for me and taught me, but I will try. I came to Chicago after an interview with Greg; impulsively changing my mind after having decided to enroll for graduate school in NYC. I was struck with his attention to detail: never allowing any minor detail to be swept under the proverbial rug, in fact often refusing to move forward till it was clarified, thus forcing me to think clearly as a result. Almost all my interactions with Greg seemed to have an inherent didactic value, perhaps by design, since it is something that also reflects in his excellent course. I learned a great deal from my early meetings with him, his wisdom, his flair for fairness, good humour, straight-shooting ways and aversion to bullshit. Other than my parents, Greg is the only person responsible for seeing me through graduate school. Often I meandered through various UChicago departments and thus technically he never had to care or bother, but I always knew that he had my back. I sometimes worry that I might have frequently disappointed Greg, other than testing his patience to the limit. Because of all that Greg has taught me and done for me, I hope I can make him proud someday. Greg was my primary advisor for Parts I & II of this dissertation.

I am truly grateful for my interactions with Risi Kondor, who in many ways has been my second advisor. I was drawn to Risi because of his proclivity to gravitate towards deep problems, my own modest undergraduate training in signal processing, and his organizing a study group on the regularity lemma–which was a major component of my master's thesis, which I was curious about. After that I became a regular in all his classes and group meetings, and felt lucky to be associated with his group after summer 2013. Risi is a very deep thinker, with a wide range of

1 *Ash Wednesday, T. S. Eliot*

knowledge, who has always tried to rub it on to his students, for which I am grateful. Risi was my primary advisor for part III of this dissertation.

Next I would like to thank Samory Kpotufe and Brian D. Nord. Samory was my co-advisor for work presented in chapter 6 of this dissertation. Through him I came to appreciate classical statistical theory and learning theory, as well as the art of thinking about machine learning problems theoretically. It is again difficult to express how grateful I feel for my interactions with Brian, which were both thoroughly enjoyable and uplifting. I thank him for his steady friendship and welcoming me to his astrophysics group at UChicago of which I have been a part of since the summer of 2017. Brian taught me a lot about problems in physics and the applicability of machine learning to them: through his weekly group meetings, through numerous one to one meetings as well our collaboration on a number of projects. In many ways Brian also was like my advisor to who I usually turned towards for counsel in case of professional issues during the last year of school as well the more prosaic side of being a grad student. I am also particularly grateful to Brian for serving on my committee, carefully reading through multiple iterations of this document and giving challenging comments on nearly every page.

I am also thankful to Kevin Gimpel for agreeing to serve on my committee despite the expedited time-line, for his comments to improve the quality of his dissertation, as well as putting up with my ever shifting deadlines with patience. I would also like to thank Rohit Nagpal for being a great teacher, good friend and collaborator. In the stressful period of job applications when I found myself stuck, he was generous enough to take up my problem and not only helped me solve it, but also invested the time to teach me every week and shed my fear of representation theory with no expectation of return.

I am grateful to my co-authors and collaborators with who I have worked on several interesting projects during my time in Chicago (listed in chronological order): Fei Song, Yutao Wang, Gábör N. Sárközy, Neil T. Heffernan, Gregory Shakhnavorich, David McAllester, Samory Kpotufe, Jialei Wang, Behnam Neyshabur, Ryohei Fujimaki, Risi Kondor, Horace Pan, Truong Son Hy, Brandon M. Anderson, Kirk Swanson, Joshua Lequieu, Zhen Lin, Rohit Nagpal, Brian D. Nord, Camille Avestruz, João Caldeira, W. L. Kimmy Wu, Nick Huang and Kyle Story.

Amongst faculty members at TTI, I would particularly like to thank David McAllester, Karen Livescu, Madhur Tulsiani and Yury Makarychev. David and Karen were amongst my favorite people at the TTI. I have enjoyed almost all my interactions with David and learned a lot from them. In the ocean of the tough crowd that is TTI in scientific matters, I found David's encyclopedic knowledge and his easy warmth refreshing and inspiring. Despite his stature, he always listened to my very frequent and ill-posed ramblings, always patiently error-correcting and reframing them. I also found his constant presence in the deep learning reading group, that I organized for 5 years, gratifying and learned a lot from his comments, as I frequently found myself to be the presenter. I regret not picking up speed faster and not collaborating with him more. I am grateful to Karen for help on many occasions (especially when I was required to take a course mid-quarter). I regret not writing up my speech course project for publication, despite her suggestion; for it would have been a fitting chapter in this dissertation. I am also thankful to Yury for going out of his way and spending a considerable amount of time, outside the purview of official coursework, to help improve my algorithmic thinking. I am thankful to Madhur for his help on various occasions as the director of graduate studies, as well as helping me with my random theory questions often.

Other than David, Greg, Karen, Kevin, Madhur and Yury, I would also like to thank other permanent faculty members at the TTI: Sadaoki Furui, Avrim Blum, Julia Chuzhoy, Nathan Srebro, Matthew Walter and Jinbo Xu, for their efforts in making TTI a truly lively and vibrant unit, while maintaining the highest research standards. When I started, while it looked like a very interesting place from the outside, I have no hesitation in saying that it was tough for students. However, just in a few years I have seen the change as it matures, and now I see it as an ideal for how an academic unit ought to be organized.

I have also learned a lot from my frequent interactions with the following research assistant professors at TTI: Mohit Bansal, Srinadh Bhojanapalli, Suriya Gunasekar, Mehrdad Mahdavi, Michael Maire, Subhransu Maji, Mesrob Ohannessian, George Papandreou, Karl Stratos and Ryota Tomioka. In particular I would like to thank Michael and Mesrob for their help on many occasions and Mesrob for being generous with his time, and being game for working through problems and books together.

While I have never been a "course person"; lacking the discipline to do well, I nevertheless made an attempt to sit through all sorts of courses that seemed interesting. In particular I learned a lot from the fantastic courses taught by László Babai, Alexander Razborov, Gregory Shaknavorich, David McAllester and Mary Silber. Laci is by far the best teacher I have encountered, and I made it a point to sit through whatever he taught. Greg's well prepared course (that I also had the privilege to TA) and his from-first-principles approach to teaching was the inspiration for my own graduate course at UChicago when I got the opportunity to teach. David's type theory course was more of a philosophy course, that I found both painful and thoroughly enjoyed. Although I was always behind by one week throughout, Mary's dynamical systems course remains the only course in my entire student life, where I have worked through the entire textbook.

Amongst the students at TTI, I would begin with Haris Angelidakis (Geia sou Hari! Xairomai poly pou eisai Chicago!). I am grateful to Haris for the inspiring company and for being one of my closest friends in Chicago. Haris was one of the few people who I could ask for help without my pride getting in the way, and who I could always ask to come for a cigar or a walk at even 4 in the morning. My equation with Haris was such that if we did not interact even for a day, it felt unusual and weird, and without him, my stay in Chicago would have been that much more drab and uninteresting than what it became. I also learned a lot from our study groups on proofs, measure theory and graph theory. I would also like to thank the early "deserters": Kaustav Kundu, Abhishek Sen and Vikas "Monty Parbat" Garg, who made the first couple of years pass in a jiffy. I would also like to thank Mrinalkanti Ghosh and Omar Montasser for their frequent help. I really respected Mrinal for his command over Kolmogorov complexity and ergodic theory and the frequent conversations about CS theory that I engaged with him. I thank him for putting up with my frequent mood swings, and the deluge of really bad jokes as one of my officemates. I would like to thank Behnam Neyshabur for our work together and Rachit Nimawat for his frequent help and sharing my appreciation of the lake. I would also like to thank Somaye Hashemifar, Avleen Bijral, Falcon Dai, Sudarshan Babu, Igor Vasiljević, Routian Luo, Kevin Stangl, Mohammadreza Mostajabi, Shane Settle, Lifu Tu, Andrea Daniele, Pedro Savarese, Payman Yadollahpour and Davis Yoshida for making day to day life during grad school fun and enjoyable. Out of the various interns that have passed through TTI, I would like to thank Akash Kumar, Abhishek Sharma and Dimitri Hanukaev. Dima has since become a good friend and it is rare even now for a fortnight to pass without a conversation on Israeli or Indian politics.

I used to joke in my first three years, that if my graduate training were cast as a structured prediction problem, then TTI with its tough crowd, which I often found challenging, would constitute the loss augmented inference part; while UChicago, where I felt smart, would constitute the inference part. I have already referred to the role that Risi, Brian and Rohit have played in my graduate career. But amongst Stats/CS/Booth students and postdocs, I would like to thank Sabyasachi Chatterjee, who I interacted with almost daily since we worked at the same coffee shop; Naiqing Gu for his kindness, confidence in me and introducing me to many interesting problems in networks; Gustav Larsson for always being helpful and inspiring. I am also thankful to my frequent conversations about work, life and research with Goutham Rajendran, Liwen Zhang and Hin Yin Tsang. Out of the students in Risi's group, I am thankful for my interactions with Yi Ding, Brandon Anderson, Jonathan Eskreis-Winkler, Hanna Torrence, Horace Pan and in particular Pramod Kaushik Mudrakarta.

In other reaches of UChicago and my meanderings through it, I would like to thank Ayelet Fine, Ana Ilievska, Katie Shapiro, Alexander Belikov, Pierre Gratia and Julia Thomas. I counted Ana as amongst my good friends in Chicago, and I always appreciated her veering every conversation about machine learning towards humanistic implications. I thank Pierre Gratia for sharing my obsessive love for books and Alexander Belikov for introducing me to many interesting problems in transport and graph curvature. I consider it an honor to count Julia Thomas as amongst my great friends. Despite being a professor (at Notre Dame) and a Japan expert, she always made me feel like the expert, and despite being twice my age she taught me a thing or two about youthfulness. I will miss Julia and our frequent walks circling the lake talking about literature and science. I would also like to thank all the friends I made due to my association with Doc Films, and Kagan Arik for being my Aikido sensei for many years, till I shattered my tarsals.

It might seem out of place for a graduate student to say so, but I also had the privilege to interact with various students initially as a "teacher" (in capacities as TA, full instructor and my tendency to find people to teach privately). Finding it fulfilling, I put a lot of energy into teaching and eventually ended up learning a lot from the experience. In many cases some of the students ended up being great friends as time went by, or my teachers and even collaborators and co-authors. In particular I would like to thank Zhen Lin, who is perhaps one of the smartest and most hardworking people I have known; Kirk Swanson, for our collaboration and introducing me to interesting problems in glassy dynamics; Milica Popović, for her kindness and her penchant to surprise. I consider it a great honor to be able to call her one of my great friends; Xinguo Fan; Zimo "silent plum" Li; Nasr Maswood, for his friendship and our frequent conversations and meetings despite his moving out of UChicago and Philip Sparks, who I find inspiring and who makes me feel proud.

I have also had three long research visits during my PhD. I would like to thank Ryohei Fujimaki and Yusuke Muruoka for their mentorship and Maxine Clochard and Ákos Kovacs for the hospitality.

Next I would like to thank all the past and present denizens of 22E; my various house-mates, who have had a major role in making the whole PhD life enjoyable. In particular, I would like to thank Ankan Saha, Pooya Hatami, Sarah Perou, Yuan Li, Emily Schofield, Adil Tobaa, Yael Levy, Thomas Gao and Shubham Toshniwal. I used to really appreciate my frequent conversations with Ankan and Pooya, usually on CS theory, mathematics and politics, extending late in the night till early morning. I am also grateful to Yuan for his generosity and willingness to help unpack my frequent theory questions. I am still psyched by the fact that Yael, who was a Buber scholar, didn't believe during the entirety of her stay that I did not study religion. I am also grateful to Thomas and (SLT specialist) Shubham for putting up with my extremely erratic schedules as might be expected in final year of graduate school. I also enjoyed my interactions with Shubham, who I saw little of at TTI before we became house-mates.

I am very grateful to many of my friends who were around me most of the time I was in Chicago. In particular, I would like to thank Srikant Veeraraghavan for his constant and steady companionship, and our frequent and somehow unplanned adventures. I always looked forward to my weekly meetings with Vaibhav Pandit who somehow ended up in Chicago after a long random walk of his own, thus somehow recreating the time from our high school days. I was always thankful for my interactions with Predrag and Milica Popović, who often, unknowingly, provided a lot of emotional support. I would also like to thank Pramod Mudrakarta, Elyse, Anamika Acharya, Nora Pfeiffer, Adinath Narasgond, Aniket Joglekar, Gasthi, Brenda Oord and Gabriela Jäger.

This acknowledgment section would be incomplete if it did not include a reference to the time I have spent in McGriffet House, where I spent as much time during graduate school as I spent at TTI, and did a sizable chunk of the work that is in this dissertation. Being a regular, I gradually came to know everyone who showed up there, and made many great friends. It helped that most people there, including some Math/Stat professors, assumed I was a professor at UChicago. But I would like to thank three in particular: Muriel Bernardi, Matt Jones and Ben Tianen. Muriel is one of the loveliest people I have met (not just in UChicago), and I am grateful to know her. I always looked forward to her inspiring company in midst of the madness and stress of my final year, and I think the graduate school experience would have been severely impoverished without knowing her. She was a breath of fresh air in the UChicago crowd and I really appreciated her kindness, absurdly funny sense of humor, and how she always inspired me to be a better person. Like in the case of Haris, my almost daily interactions, frequent walks and long conversations with Matt had a major role in keeping life in Chicago interesting.

I am also grateful to my advisors from my earlier pit-stops: Gábor Sárközy, Neil Heffernan and Kalyani Joshi. Not a month went by when I did not hear from atleast one of them, checking on me, encouraging me and constantly making me feel supported.

Outside of TTI-C and UChicago, I would also like to thank Taco S. Cohen, Caglar Gulcehre, Song Liu and Faruk Ahmed. Taco was kind enough to not only share pre-prints of two of his papers, relevant to some of my work, long before they appeared online, but also helped me with detailed instructions to place a chapter in this dissertation (which unfortunately had to be cropped out due to want of time towards the end). Caglar threw a few problems in dynamical systems at me, that were interesting enough for me to take relevant courses. Although I did not end up working on those specific problems, interactions with him have had a lasting impact on me and I foresee using the knowledge acquired in my research in the near feature. I am also thankful to Song for sharing several of his problems on the Stein estimator during his UChicago visit and discussing them, while we attempted a cross-Atlantic collaboration. I am thankful to Faruk for saving me when I somehow landed in Montréal to present the work in chapter 3 at NIPS, without any money or a working phone.

I am immensely grateful to all my friends from my time in Pune (a time, which, despite a complete lack of research resources and mentorship, I refer to as my *halcyon seasons, solstice of my days*[2]); who I consider to be my best, closest and most dependable friends. Despite the thousands of miles in distance, our friendships only keep getting better with time. I am grateful for their constant support and love, and for being a steady source of strength. It would be impossible to name them without this section becoming as long as the dissertation itself, I could only name Pandit since he was in Chicago. I will however express my gratitude to Ritika, for the role in making me whoever I am today, as well as the early encouragement to pursue research, without which this dissertation would have never happened.

Finally, I think it would be presumptuous to even attempt to thank my parents and my siblings Divya and Dewanshu for everything that they have done for me; not the least for their sacrifices and encouragement–always supporting me no matter what I did and seeing me through the proverbial yellow brick road. I find it remarkable that despite my parents' backgrounds, the importance of all-round scholarly pursuits and the sacrifice it naturally entails is something they tried to instill in all their children from the very beginning. It is hard to find words to appreciate their dedication and disarming authenticity. I dedicate this thesis to them, as well as to my late grandfather: Prof. Ramesh Chandra Trivedi, whose immense serenity and wisdom I found awe-inspiring, and who might *just* have been proud.

Shubhendu Trivedi
Cambridge, MA
September 1, 2018.

2 Memoirs of Hadrian, Marguerite Yourcenar

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

1 INTRODUCTION AND OVERVIEW

One of the most fundamental questions in machine learning is to compare examples: Given a pair of objects (ξ_1, ξ_2), we want to automatically predict a value $\Psi(\xi_1, \xi_2) \in \mathbb{R}$, the magnitude of which indicates the degree of similarity or dissimilarity between ξ_1 and ξ_2. To underline the central nature of this problem, it is useful to consider the wide range of machine learning algorithms that explicitly or implicitly rely on a notion of pairwise similarity. Some such methods include: example based approaches such as k nearest neighbors [28]; clustering algorithms such as k-means [101], mean-shift and centroid based methods, spectral clustering [152]; the various flavours of kernel methods such as support vector machines [12], kernel regression, Gaussian processes [126] etc.

The similarity between a pair of objects is customarily obtained as a function of some pre-defined pairwise distance, which in turns depends on the nature of the objects ξ_1 and ξ_2. If the objects live in an explicit feature space, the Euclidean distance is a common choice; similarly, the χ^2-squared distance is frequently used if the objects reside in a simplex; likewise, the Levenshtein distance may be employed if the objects are strings (see [34] for an exhaustive catalog of distance measures). As might be expected, such distances often fail to account for the quirks of a particular dataset and task at hand, and indeed, one might expect improved performance if the distance function is instead tailored to the task. Designing such distance functions automatically is the motivation behind the area of *metric learning* [168].

In its most general form, distance metric learning leverages examples provided for the task at hand, in order to wriggle out a better suited, task-specific distance function. For example: If the task is clustering, and we are provided with sets of items and complete clusterings over these sets, we would like to exploit this side information to estimate the distance function that can help cluster future sets better. Yet another example, which is by far most commonly addressed in the metric learning literature is when the task is classification or regression using a nearest neighbor method. The side information furnished to us comprises of labels of points, which are then used to learn a distance function that can improve k-NN performance. We explicate further on the latter example in what follows, in a relatively simple setting, to better motivate and build ground to summarize the main contributions of this thesis.

Suppose we are working with a classification problem, which is specified by a suitable instance space $(\mathcal{X}, d)$, assumed to be a metric space, and a label space $\mathcal{Y}$. In particular, we assume that $\mathcal{X} \subset \mathbb{R}^d$, therefore $\xi_1, \xi_2 \in \mathbb{R}^d$. We also assign $\Psi(\xi_1, \xi_2) = +1$ if ξ_1 and ξ_2 are of the same class, and $\Psi(\xi_1, \xi_2) = -1$ otherwise. Furthermore, given a map $\xi \mapsto \Phi(\xi)$ parameterized by $\mathcal{W}$, let us suppose the distance between ξ_1 and ξ_2 is given as:

$$D_{\mathcal{W}}(\xi_1, \xi_2) = \|\Phi(\xi_1; \mathcal{W}) - \Phi(\xi_2; \mathcal{W})\|_2^2$$

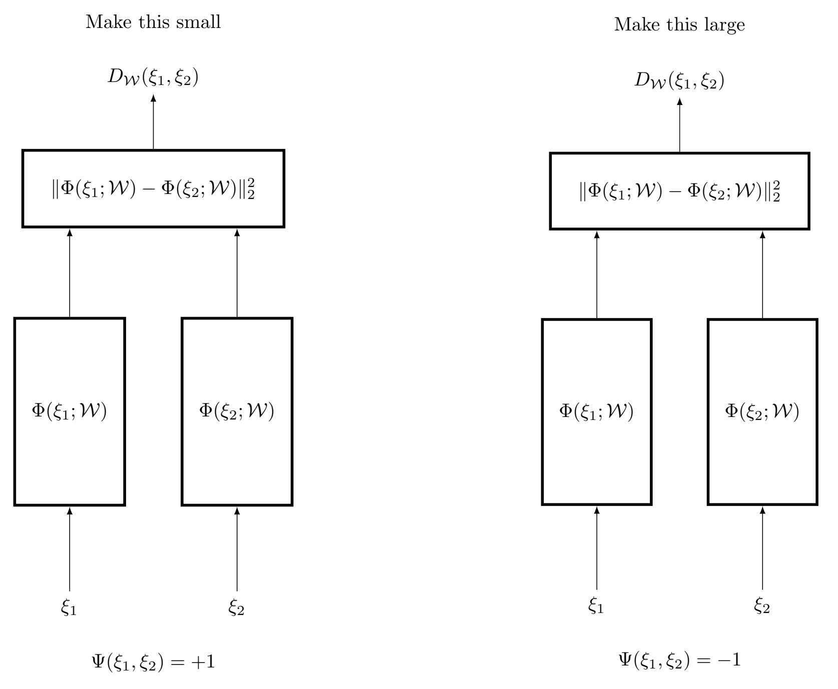

Make this small
$D_{\mathcal{W}}(\xi_1, \xi_2)$
$\|\Phi(\xi_1; \mathcal{W}) - \Phi(\xi_2; \mathcal{W})\|_2^2$
$\Phi(\xi_1; \mathcal{W})$
$\Phi(\xi_2; \mathcal{W})$
ξ_1
ξ_2
$\Psi(\xi_1, \xi_2) = +1$
Make this large
$D_{\mathcal{W}}(\xi_1, \xi_2)$
$\|\Phi(\xi_1; \mathcal{W}) - \Phi(\xi_2; \mathcal{W})\|_2^2$
$\Phi(\xi_1; \mathcal{W})$
$\Phi(\xi_2; \mathcal{W})$
ξ_1
ξ_2
$\Psi(\xi_1, \xi_2) = -1$

Figure 1.1: A typical contrastive loss setup that learns mappings that are similarity sensitive

We can set the optimization so as to update parameters $\mathcal{W}$ such that $D_{\mathcal{W}}(\xi_1, \xi_2)$ is increased if $\Psi(\xi_1, \xi_2) = -1$, and $D_{\mathcal{W}}(\xi_1, \xi_2)$ is decreased if $\Psi(\xi_1, \xi_2) = +1$. This is illustrated in figure 1.1.

1.1 THE INTERPLAY BETWEEN SIMILARITY AND REPRESENTATION LEARNING

While the above example illustrates a simple method to learn a similarity-sensitive distance function, there are a few crucial issues that were swept under the proverbial rug, which we unpack below.

First of all, notice that in the example we did not assume anything about the structure of ξ_1 and ξ_2, except that they were points in $\mathbb{R}^d$. In such a setting $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^p$ (with $d = p$ not necessarily true) corresponds to a mapping, such that in the transformed space distances are more reflective of similarity. In short, we assume that we already have a good feature representation for our data, on top of which a distance function could be learned. Indeed, if the feature representation is poor i.e. has poor class discriminative ability, then learning similarity sensitive distances would be hard if not impossible.

However, the objects ξ_1 and ξ_2 might come endowed with richer structure, as is often the case in various applications of machine learning. For example, (ξ_1, ξ_2) might be a pair of images, or a pair of sets, or a pair of spherical images, or a pair of point-clouds and so on. In such cases $\Phi : \xi \mapsto \mathbb{R}^d$ could instead be a module that learns a representation for the object that is inherently discriminative and models natural invariances in the data.

To drive home this point, consider the example illustrated in figure 1.1 again, but with the modification that the objects ξ_1 and ξ_2 are large $d \times d$ images, and $\mathcal{W}$ represents the parameters of a fully-connected feed-forward network. The system is then trained to be similarity sensitive as discussed. Such a system is likely to perform poorly, because the fully connected network is unlikely to generate good representations for the images. On the other hand, if $\mathcal{W}$ instead represents the parameters of a Convolutional Neural Network (CNN) [91], the system is far more likely to succeed. That this should be the case is not hard to see, indeed, since CNNs are known to generate extremely good representations for images.

As illustrated by the above example, in the context of similarity learning, there are two notions that are crucial to good performance:

1 Having a rich; discriminative representation for the type of input ξ, which models natural invariances and symmetries in the data.

2 If the underlying task is nearest neighbor classification or regression, as is usually the case in similarity learning, we would want to devise a loss that is a more direct proxy to nearest neighbor performance.

Both these notions: having an appropriate representation for the data type at hand, as well as working with the right notion of loss for the learning of similarity reinforce each other, and can also be learned jointly end-to-end. Nevertheless, as already noted, getting both of these aspects in order is pivotal to good performance. In this dissertation, we make contributions towards both aspects, which we describe below, while also outlining the organization of this document.

1.2 DISCRIMINATIVE METRIC LEARNING

In **Part i** of this dissertation, we wholly focus on the loss formulation for the discriminative learning of similarity and distance, while ignoring representational issues. That is, we assume that the inputs $\xi_i \in \mathbb{R}^d$, and that the representation is good enough for the task at hand.

As already discussed, often, k-NN prediction performance is the real motivation for metric and similarity learning, on which there is a large literature. Typically such methods set the problem as an optimization problem, with the metric updated in such a way that good neighbors (say from the correct class for a query point) are pulled together, while bad neighbors are pushed away. We utilize **Chapter 2** to review some popular methods for discriminative metric learning.

In **Chapter 3**, we propose a formulation for metric learning that makes a more direct attempt to optimize for the k-NN accuracy as compared to prior work. Our approach considers the choice of k neighbors as a discrete valued latent variable, and casts the metric learning problem as a large margin structured prediction problem. This formulation allows us to use the arsenal of techniques for structural latent support vector machines for the problem of metric learning. We also devise procedures for exact inference and loss augmented inference in this model, and also report experimental results for our method, comparing to a suite of popular metric learning methods.

In **Chapter 4**, we consider the direct loss minimization approach to metric learning from Chapter 3 and apply it in three different settings: Asymmetric similarity learning, discriminative learning of Hamming distance, and metric learning for improving k-NN regression.

1.3 METRIC ESTIMATION WITHOUT LEARNING

In **Part ii** of the dissertation we consider a somewhat different tack: Suppose $\xi_i \in \mathbb{R}^d$ and that this is a good representation of the data. However, we now consider a situation where we are on a limited computational budget i.e. optimizing over a space of possible metrics would be infeasible. Nevertheless we still want access to a good metric that could improve k-NN classification and regression performance as compared to the plain Euclidean distance.

In **Chapter 6**, we consider the case of regression and binary classification i.e. when we have an unknown regression function $f : \mathbb{R}^d \to \mathbb{R}$, and consider the metric given by the Expected Gradient Outer Product (EGOP)

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x})^\top \right).$$

We give a cheap estimator for the EGOP and prove that it remains statistically consistent under mild assumptions, while also showing empirically, that using the EGOP as a metric improves k-NN regression performance.

In **Chapter 7**, we consider the multi-class case i.e. when we have an unknown function $f : \mathbb{R}^d \to \mathbb{S}^c$ with $\mathbb{S}^c = \{\mathbf{y} \in \mathbb{R}^c | \forall i \; y_i \geq 0, \mathbf{y}^T \mathbf{1} = 1\}$, and consider the metric given by the Expected Jacobian Outer Product (EJOP)

$$\mathbb{E}_X G(X) \triangleq \mathbb{E}_{\mathbf{x}} \left(\mathbf{J}_f(\mathbf{x}) \mathbf{J}_f(\mathbf{x})^T \right)$$

where $\mathbf{J}_f$ is the Jacobian of f. Like in the case of EGOP, we give a rough estimator, that not only remains statistically consistent under reasonable assumptions, but also gives improvements in k-NN classification performance.

1.4 GROUP EQUIVARIANT REPRESENTATION LEARNING

In **Part iii** of this thesis we address the representational issues discussed earlier in this chapter. Chapter **??** introduces group equivariant neural networks and makes the case on how such neural networks exploit natural invariances in the data. We argue that group equivariance is an useful inductive bias in many domains. We start with the simple case of planar CNNs, and then review more recent efforts on generalizing classical CNNs in different settings. In chapter 8 we give an example of a group equivariant representation learning module: a SO(3) equivariant spherical convolution neural network that operates entirely in Fourier space.

Part I

DISCRIMINATIVE METRIC LEARNING

2 INTRODUCTION TO DISCRIMINATIVE METRIC LEARNING

Amongst the oldest [28] and most widely used tools in machine learning are nearest neighbor methods (see [134] for a survey). Despite their simplicity, they are often successful and come with attractive properties. For instance, the k-NN classifier is universally consistent [140], being the first learning rule for which this was demonstrated to be the case. Additionally, nearest neighbor methods use local information and are inherently non-linear; while also being relatively resilient to label noise, since prediction requires averaging across k labels. Moreover, it is trivial to add new classes to the data without requiring any fresh model training.

While nearest neighor rules can often be efficacious, their performance tends to be limited by two factors: the computational cost of searching for nearest neighbors and the choice of the metric (distance measure) defining "nearest". The cost of searching for neighbors can be reduced with efficient indexing (see for example [3, 9, 29]) or learning compact representations e.g. [51, 89, 118, 154]. We will defer addressing this issue till Chapter 4. In this part of the dissertation we instead focus on the choice of the metric. The metric is often taken to be Euclidean, Manhattan or χ^2 distance. However, it is well known that in many cases these choices are suboptimal in that they do not exploit statistical regularities that can be leveraged from labeled data. Here, we focus on supervised metric learning. In particular, we present a method of learning a metric so as to optimize the accuracy of the resulting nearest neighbor estimator.

Existing works on metric learning (the overwhelming majority of which is for classification) formulate learning as an optimization task with various constraints driven by considerations of computational feasibility and reasonable, but often vaguely justified principles [49, 50, 71, 107, 141, 156, 157, 168]. A fundamental intuition is shared by most of the work in this area: an ideal distance for prediction is distance in the target space. Of course, that can not be measured, since prediction of a test example's target is what we want to use the similarities to begin with. Instead, one could learn a similarity measure with the goal for it to be a good proxy for the target similarity. Since the performance of kNN prediction often is the real motivation for similarity learning, the constraints typically involve "pulling" good neighbors (from the correct class for a given point in the case of classification) closer while "pushing" the bad neighbors farther away. The exact formulation of "good" and "bad" varies but is defined as a combination of proximity and agreement between targets.

To give a flavor of some of these constraints and principles in order to improve nearest neighbor performance downstream, we review some well known metric learning algorithms in what follows. Yet another purpose for doing so will also be to set the ground for placing our approach in context. To begin to do so, we first suppose the classification problem is specified by a suitable instance space $(\mathcal{X}, d)$, which is assumed to be a metric space, and a label space $\mathcal{Y} \in \mathbb{Z}_+$. The distance between any two points $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ is denoted as $D_{\mathcal{W}}(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathcal{W}$ are the parameters that specify the distance measure.

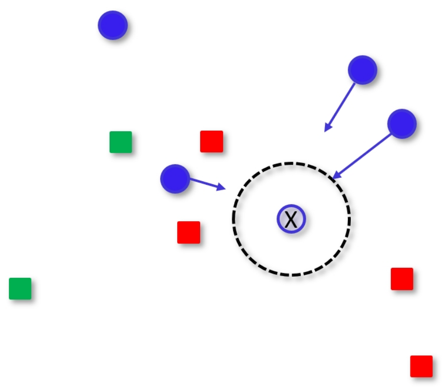


Figure 2.1: An illustration of the approach to metric learning taken by [168]. Look at the text for more details

There might be considerable freedom in deciding what $\mathcal{W}$ should be, it could just represent the identity matrix, a low-rank projection matrix, or the parameters of a neural network. By far, the most popular family of metric learning algorithms involve learning a Mahalanobis distance.

Mahalanobis Distances

Suppose $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X} \subset \mathbb{R}^d$, and $\mathbf{W} \in \mathbb{R}^{d \times d}$ as well as $\mathbf{W} \succeq 0$. In the context of metric learning, the Mahalanobis distance has come to refer to all distances of the form $D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j)}$. However, its eponymous distance measure, proposed in 1936 in the context of anthropometry [102], was defined in terms of a covariance matrix Σ as $D_\Sigma(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i, \mathbf{x}_j)^T \Sigma^{-1} (\mathbf{x}_i, \mathbf{x}_j)}$. It might also be worthwhile to note, that despite the prevalence of the term “metric learning", it is somewhat of a misnomer. This is because $D_{\mathbf{W}}$ infact defines a pseudo-metric i.e. $\forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \mathcal{X}$, it satisfies:

a. $D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) \geq 0$

b. $D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_i) = 0$

c. $D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = D_{\mathbf{W}}(\mathbf{x}_j, \mathbf{x}_i)$

d. $D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_k) \leq D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) + D_{\mathbf{W}}(\mathbf{x}_j, \mathbf{x}_k)$

There is a large body of work on similarity learning done with the stated goal of improving kNN performance, which would be impossible to review justly. Therefore, we stick to reviewing some salient approaches that also help place our own approach in context. Some of the earliest work in what could be considered proto-metric learning goes back to Short and Fukunaga (1981) [46], with a string of follow up works in the 90s, for example see Hastie and Tibshirani [56]. However, in much of the recent work in the past decade and a half, the objective can be written as a combination of some sort of regularizer on the parameters of similarity, with loss reflecting the desired “purity” of the neighbors under learned similarity. Optimization then balances violation of these constraints with regularization.

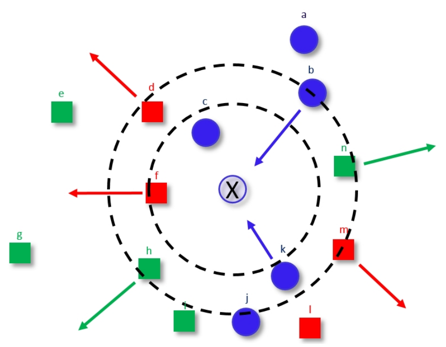


Figure 2.2: An illustration of the approach to metric learning taken by [157]. We refer the reader to the text for more details

In this sense the area of metric learning could be considered to have started with the influential work of Xing *et al.* [168]. In this method, the "good" neighbors are defined as all similarly labeled points, while "bad" neighbors are defined as all points that have a different label. During optimization, the metric is deformed such that each class is mapped into a ball of a fixed radius, but no separation is enforced between the classes (see figure 2.1). Letting $\mathcal{S}$ and $\mathcal{D}$ denote the sets of pairs of similar and dissmilar points respectively, we may write this approach as the following optimization problem:

$$\min_{\mathbf{W}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} \|\mathbf{x}_i - \mathbf{x}_j\|^2_{\mathbf{W}} \tag{2.1}$$

$$\textbf{s. t.} \min_{\mathbf{W}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} \|\mathbf{x}_i - \mathbf{x}_j\|^2_{\mathbf{W}} \geq 1 \tag{2.2}$$

$$\mathbf{W} \succeq 0 \tag{2.3}$$

Where, $\|\cdot\|^2_{\mathbf{W}}$ is the squared Mahalanobis distance parameterized by $\mathbf{W}$. Evidently, the immediate problem with this approach is that it has little relation to the actual kNN objective. Indeed, the k-NN objective does not require that similar points should be clustered together, and as a consequence methods of a similar flavour optimize an objective that is much harder than what is required for good k-NN performance.

A popular family of approaches to metric learning that has a somewhat better motivated objective than the above, are based on the Large Margin Nearest Neighbor (LMNN) algorithm [157]. In LMNN, the constraints for each training point involve a set of predefined "target neighbors" from the correct class, and "impostors" from other classes. The optimization is such that a margin is imposed between the "target neighbors" and the "impostors"(see figure 2.2). Such an objective may be written as:

$$\min_{\mathbf{L}} \sum_{i,j:j\rightsquigarrow i} D_{\mathbf{L}}(\mathbf{x}_i,\mathbf{x}_j)^2 + \mu \sum_{k:y_i\neq y_k} [1 + D_{\mathbf{L}}(\mathbf{x}_i,\mathbf{x}_j)^2 - D_{\mathbf{L}}(\mathbf{x}_i,\mathbf{x}_k)^2]_+ \tag{2.4}$$

where $\mathbf{W} = \mathbf{L}^T\mathbf{L}$; $\mu > 0$; y_i denotes the label for $\mathbf{x}_i$; the notation $j \rightsquigarrow i$ indicates that $\mathbf{x}_j$ is a "target neighbor"of $\mathbf{x}_i$ and $[z]_+ = \max(z,0)$ denotes the hinge loss.

Despite being somewhat more suited to the underlying k-NN objective, the LMNN objective still has some issues that could affect its performance. To begin, the set of target neighbors are chosen at the onset based on the euclidean distance (in absence of a priori knowledge). Moreover as the metric is optimized, the set of "target neighbors" is not dynamically updated. There is no reason to believe that the original choice of neighbors based on the euclidean distance is optimal while the metric is updated. Yet another issue is that in LMNN the target neighbors are forced to be of the same class. In doing so it does not fully leverage the power of the kNN objective, which only needs a majority of points to have the correct label. Extensions of LMNN [71, 156] allow for non-linear metrics, but retain the same general flavor of constraints.

In Neighborhood Component Analysis (NCA) [50] a different kind of proxy for classification error is used: the piecewise-constant error of the kNN rule is replaced by a soft version. This leads to a non-convex objective that is optimized via gradient descent. To write the objective, we denote the probability that a point $\mathbf{x}_i$ selects $\mathbf{x}_j$ as its neighbor by p_{ij}. In this set up, the point $\mathbf{x}_i$ will be assigned the class of point $\mathbf{x}_j$. Given $\mathbf{W} = \mathbf{L}^T\mathbf{L}$, we can define p_{ij} as:

$$p_{ij} = \frac{\exp\Big(- \|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j\|_2^2\Big)}{\sum_{k\neq i}\exp\Big(- \|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_k\|_2^2\Big)} \text{ and } p_{ii} = 0 \tag{2.5}$$

The probability that a point will be correctly classified is then given by:

$$p_i = \sum_{\mathbf{x}_j \in C_i} p_{ij} \tag{2.6}$$

where C_i denotes the set of all points that have the same class as $\mathbf{x}_i$. Finally, we can write the NCA objective (to be maximized) as follows:

$$Obj(\mathbf{L}) = \sum_i \sum_{\mathbf{x}_j \in C_i} p_{ij} \tag{2.7}$$

One of the features of NCA is that it trades off convexity for attempting to directly optimize for the choice of nearest neighbor. This issue of non-convexity was partly remedied in [49], by optimization of a similar stochastic rule while attempting to collapse each class to one point. While this makes the optimization convex, collapsing classes to distinct points is unrealistic in practice. Another recent extension of NCA [141] generalizes the stochastic classification idea to kNN classification with $k > 1$. Out of all the methods reviewed so far, k-NCA is the only method that comes closest to optimize directly for the k-NN task loss.

There is also a wide plethora of metric learning methods that optimize for some kind of ranking loss. We discuss two examples here. In Metric Learning to Rank (MLR)[107], the constraints involve all the points: the goal is to push all the correct matches in front of all the incorrect ones. The idea essentially is: given a query point and a metric parameterized by $\mathbf{W}$ finding the distance with the database points should sort them in such a way that good neighbors end up in the front. While important for retrieval, this is again not the same as requiring correct classification. In addition to global optimization constraints on the rankings (such as mean average precision for target class), the authors allow localized evaluation criteria such as Precision at k, which can be used as a surrogate for classification accuracy for binary classification, but is a poor surrogate for multi-way classification. Direct use of *k*NN accuracy in optimization objective is briefly mentioned in [107], but not pursued due to the difficulty in loss-augmented inference. This is because the interleaving technique of [67] that is used to perform inference with other losses based inherently on contingency tables, fails for the multiclass case (since the number of data interleavings could possibly be exponential). A similar approach is taking in [119], where the constraints are derived from triplets of points formed by a sample, correct and incorrect neighbors. Again, these are assumed to be set statically as an input to the algorithm, and the optimization focuses on the distance ordering (ranking) rather than accuracy of classification.

Before concluding, we must note that in this chapter we have not reviewed any of the deep metric learning techniques. This is because all such techniques that we are aware of are based on a flavour of loss as one of the above, with the only difference that the mapping of each point onto a metric space is done by a neural network. The focus and main novelty of the work presented in the next chapter lies in its loss as compared to the techniques discussed. Indeed, the function that is used to map the points to a suitable metric space is an orthogonal consideration.

Having considered a general background on the metric learning problem, along with various loss formulations that have been proposed to attack it, we now proceed to give a formulation that attempts to give a more direct proxy for *k*-NN classification.

3 METRIC LEARNING BY NEIGHBORHOOD GERRYMANDERING

OUTLINE

In this chapter, we give a formulation for metric learning that facilitates a more direct attempt to optimize for the kNN accuracy as compared to previous work. We also show that our formulation makes it natural to apply standard learning methods for structural latent support vector machines (SVMs) to the problem of supervised metric learning. While we test this approach for the case of Mahalanobis metric learning, as emphasized in the previous chapter, the focus here is on obtaining a better proxy for the k-NN loss rather than the nature of mapping.

To achieve our stated goal of formulating the metric learning problem such that it is more direct in optimizing for the underlying task: k-NN accuracy, we consider looking at the nearest neighbor problem a bit differently.

In the kNN prediction problem, given a query point and fixing the underlying metric, there is an implicit hidden variable: the choice of k "neighbors". The inference of the predicted label from these k examples is trivial: by simple majority vote among the associated labels for classification, and by taking a weighted average in the case of regression. In the case of classification, given a query point, there can possibly exist a very large number of choices of k points that might correspond to zero loss: any set of k points with the majority of correct class will do. Whereas, in the case of regression, there can exist a very large number of choices of k points that might correspond to a loss less than a tolerance parameter (since zero loss would be impossible in most scenarios). We would like a metric to "prefer" one of these "good" example sets over any set of k neighbors which would vote for a wrong class (or in the case of regression correspond to a high loss). Note that to win, it is not necessary for the right class to account for all the k neighbors – it just needs to get more votes than any other class. As the number of classes and the value of k grow, so does the space of available good (and bad) example sets.

These considerations motivate our approach to metric learning. It is akin to the common, albeit negatively viewed, practice of *gerrymandering* in drawing up borders of election districts so as to provide advantages to desired political parties, e.g., by concentrating voters from that party or by spreading voters of opposing parties. In our case, the "districts" are the cells in the Voronoi diagram defined by the Mahalanobis metric, the "parties" are the class labels voted for by the neighbors falling in each cell, and the "desired winner" is the true label of the training points associated with the cell. This intuition is why we refer to our method as *neighborhood gerrymandering* in the title.

A bit more technically, we write kNN prediction as an inference problem with a structured latent variable being the choice of k neighbors. Thus learning involves minimizing a sum of a structural latent hinge loss and a regularizer [12]. Computing structural latent hinge loss involves loss-adjusted inference — one must compute loss-adjusted values of both the output value (the label) and the latent items (the set of nearest neighbors). The loss augmented inference corresponds to a choice of worst k neighbors in the sense that while having a high average similarity they also correspond to a high loss ("worst offending set of k neighbors"). Given the inherent combinatorial considerations, the key to such a model is efficient inference and loss augmented inference. We give an efficient algorithm for *exact* inference. We also design an optimization algorithm based on stochastic gradient descent on the surrogate loss. Our approach achieves kNN accuracy higher than state of the art for most of the data sets we tested on, including some methods specialized for the relevant input domains.

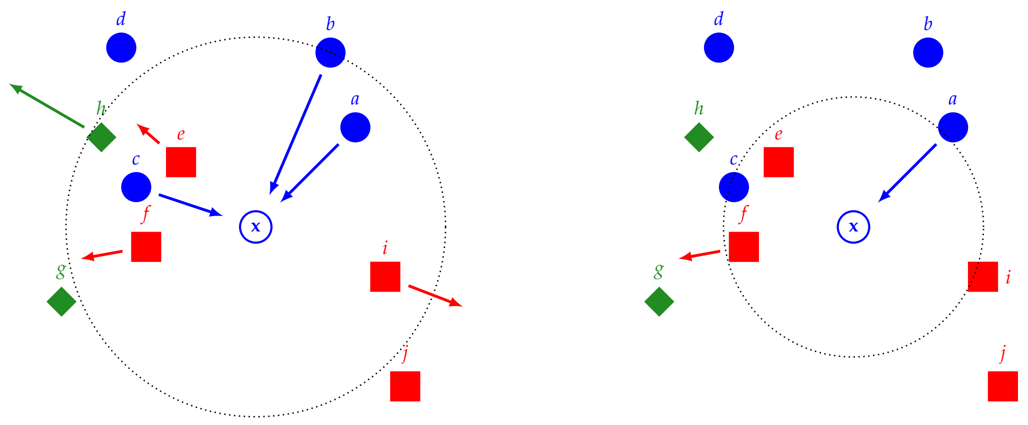


Figure 3.1: Illustration of objectives of LMNN (left) and our structured approach to "neighborhood gerrymandering" (right) for $k = 3$. The point $\mathbf{x}$ of class blue is the query point. In LMNN, the target points are the nearest neighbors of the same class, which are points a, b and c (the circle centered at $\mathbf{x}$ has radius equal to th e farthest of the target points i.e. point b). The LMNN objective will push all the points of the wrong class that lie inside this circle out (points $e, f, h, i,$ and$j$), while pushing in the target points to enforce the margin. On the other hand, for our structured approach (right), the circle around $\mathbf{x}$ has radius equal to the distance of the farthest of the three nearest neighbors irrespective of class. Our objective only needs to ensure zero loss. This would be achieved by pushing in point a of the correct class (blue) while pushing out the point having the incorrect class (point f). Note that two points of the incorrect class lie inside the circle ($e,$ and$f$), both being of class red. However f is pushed out and not e since it is farther from $\mathbf{x}$.

As stated toward the end of the previous chapter, although we initially restrict ourselves to learning a Mahalanobis distance in an explicit feature space, the formulation is easily extensible to nonlinear similarity measures such as those defined by nonlinear kernels, provided computing the gradients of similarities with respect to metric parameters is feasible. In such extensions, the inference and loss augmented inference steps remain unchanged. Our formulation can also naturally handle a user-defined loss matrix on labels rather than just a zero-one loss. We propose a series of extensions to the case of kNN regression, Asymmetric metric learning and the discriminative learning of Hamming distance in Chapter 4. The extension to regression seems particularly foreboding given that in this case the number of "classes" is uncountable. This is attacked by both modifying the objective suitably and presenting algorithms for inference and loss augmented inference to give a suitable approximation that is shown to perform well on standard benchmarks.

3.1 GERRYMANDERING IN CONTEXT

In chapter 2, we introduced the metric learning problem and discussed some canonical approaches to the problem. In this short section, we hark back to some of the approaches discussed there to put our framework in context.

1 **Clustering type objectives:** Such methods for learning the metric are exemplified by the work of Xing *et al.* [168]. In such approaches, "good" neighbors for a given query point are all points that have a similar label. The metric is learned so as to map all such "good" neighbors into a ball of fixed radius. However, the k-NN objective does not require such clustering behaviour for good performance. In that sense our approach is more direct in leveraging the k-NN objective.

2 **LMNN type objectives:** As discussed earlier, in LMNN type algorithms [157], for a query point, the "good" neighbors are a set of "target neighbors" which are a) predefined and b) are all of the same class. In a way, the role of "target neighbors" in LMNN is not quite unlike the "best correct set of k neighbors" (h^* in Section 3.3) in our method. Moreover, in LMNN type methods, the"target neighbors" are predefined based on the Eucidean metric and then fixed throughout learning. In our method, the set of "good" neighbors h^* are dynamically updated as the metric is learned. Yet another departure in our approach to that of LMNN is that for good k-NN performance, we don't need all the "good" neighbors to be of the same class. In our method we provide inference procedures that ensure leveraging the k-NN objective more directly by only focusing on having a majority of points to be of the correct class. This is also illustrated in an example in figure 3.1.

3 **NCA:** Our method is similar to NCA [50] type methods in that it also trades off convexity (details in Section 3.3) in order to directly optimize for the choice of nearest neighbor. However, traditional NCA type algorithms focus only on 1-NN. The work of [141] that generalizes [50] to instead focus on the right selection of k nearest neighbors for classification is closest in spirit to our work, and as far as we are aware the only work attempts to optimize directly for the k-NN objective. Experimentally, we found our method gave superior performance.

4 **Ranking objectives:** The original inspiration for this work was *metric learning to rank*[107], which optimizes for a ranking objective. As discussed this is not the same as requiring correct classification. Moreover, as discussed the approach of [107] fails for the multiclass case given the inference used. We take a very different approach to loss augmented inference, using targeted inference and the classification loss matrix, and can easily extend it to arbitrary number of classes

3.2 DISCRIMINATIVE LOSS MINIMIZATION FOR CLASSIFICATION

In this section we formally set up the problem. Note that we first state the distance and similarity formulation in its full generality, to illustrate that it is more widely applicable, and not just to the case of learning global linear projections. We then modify it to work with the Mahalanobis distance that is eventually dealt with in the rest of this chapter, and for which detailed experimentation is carried out.

3.2.1 *Problem setup*

We are given N training examples $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, represented by a "native" feature map, $\mathbf{x}_i \in \mathbb{R}^d$, and their class labels $\mathbf{y} = [y_1, \ldots, y_N]^T$, with $y_i \in [R]$, where $[R]$ stands for the set $\{1, \ldots, R\}$. We are also given the loss matrix $\boldsymbol{\Lambda}$ with $\Lambda(r, r')$ being the loss incurred by predicting r' when the correct class is r. We assume $\Lambda(r, r) = 0$, and $\forall (r, r')$, $\Lambda(r, r') \geq 0$. Most generally, we are interested in squared distances defined as:

$$D_{\mathcal{W}}(\mathbf{x}, \mathbf{x}_i) = \|\Phi(\mathbf{x}; \mathcal{W}) - \Phi(\mathbf{x}_i; \mathcal{W})\|_2^2 \tag{3.1}$$

Where $\Phi(\mathbf{x}; \mathcal{W})$ is a map (possibly non-linear), $\mathbf{x} \rightarrow \Phi(\mathbf{x})$, parameterized by $\mathcal{W}$. Let $h \subset X$ be a set of examples in X. For a given $\mathcal{W}$ we define the distance score of h w.r.t. a point $\mathbf{x}$ as

$$S_{\mathcal{W}}(\mathbf{x}, h) \;=\; -\frac{\alpha}{K} \sum_{\mathbf{x}_j \in h} \|\Phi(\mathbf{x}; \mathcal{W}) - \Phi(\mathbf{x}_j; \mathcal{W})\|_2^2 + \beta \tag{3.2}$$

where α, β and K are constants. This formulation of the distance score permits use of the dot product to measure similarity as well, as long as it is normalized to unit length.

For the rest of this chapter, we are interested in the *Mahalanobis metrics*

$$D_{\mathbf{W}}\left(\mathbf{x}, \mathbf{x}_i\right) \;=\; \left(\mathbf{x} - \mathbf{x}_i\right)^T \mathbf{W} \left(\mathbf{x} - \mathbf{x}_i\right), \tag{3.3}$$

which are parameterized by positive semidefinite $d \times d$ matrices $\mathbf{W}$, which can be seen as learning a linear map $\mathbf{x} \rightarrow \mathbf{L}\mathbf{x}$ where $\mathbf{W} = \mathbf{L}^T\mathbf{L}$, while satisfying fixed constraints (usually to optimize for kNN performance). For a given $\mathbf{W}$ we define the distance score of h w.r.t. a point $\mathbf{x}$ as

$$S_{\mathbf{W}}(\mathbf{x}, h) \;=\; -\sum_{\mathbf{x}_j \in h} D_{\mathbf{W}}\left(\mathbf{x}, \mathbf{x}_j\right) \tag{3.4}$$

Hence, the set of k nearest neighbors of $\mathbf{x}$ in X is

$$h_{\mathbf{W}}(\mathbf{x}) \;=\; \underset{|h|=k}{\operatorname{argmax}}\, S_{\mathbf{W}}(\mathbf{x}, h). \tag{3.5}$$

For the remainder of this discussion, we will assume that k is known and fixed. Note that, from any set h of k examples from X, we can predict the label of $\mathbf{x}$ by (simple) majority vote:

$$\widehat{y}\,(h) \;=\; \text{majority}\{y_j : \mathbf{x}_j \in h\},$$

with ties resolved by a heuristic, e.g., according to 1NN vote. In particular, the kNN classifier predicts $\widehat{y}(h_{\mathbf{W}}(\mathbf{x}))$. Due to this deterministic dependence between $\widehat{y}$ and h, we can define the classification loss incurred by a voting classifier when using the set h as

$$\Delta(y, h) \;=\; \Lambda\left(y, \widehat{y}(h)\right). \tag{3.6}$$

3.3 LEARNING AND INFERENCE

One might want to learn $\mathbf{W}$ to minimize training loss

$$\sum_i \Delta\left(y_i, h_{\mathbf{W}}(\mathbf{x}_i)\right)$$

However, this fails due to the intractable nature of classification loss Δ. We will follow the usual remedy: define a tractable surrogate loss.

While already discussed earlier, here we must note again that in our formulation, the output of the prediction is a structured object $h_{\mathbf{W}}$, for which we eventually report the deterministically computed $\widehat{y}$. Structured prediction problems usually involve loss which is a generalization of the hinge loss; intuitively, it penalizes the gap between score of the correct structured output and the score of the "worst offending" incorrect output (the one with the highest score *and* highest Δ).

However, in our case, we have an additional complication in that there is no single correct output h, since in general many choices of h would lead to correct $\widehat{y}$ and zero classification loss: any h in which the majority votes for the right class. Ideally, we want $S_{\mathbf{W}}$ to prefer *at least one* of these correct hs over all incorrect hs.

This intuition leads to the following surrogate loss definition:

$$L(\mathbf{x}, y, \mathbf{W}) = \max_h \left[S_{\mathbf{W}}(\mathbf{x}, h) + \Delta(y, h)\right] \tag{3.7}$$

$$- \max_{h:\Delta(y,h)=0} S_{\mathbf{W}}(\mathbf{x}, h). \tag{3.8}$$

This is quite different in spirit from the notion of margin sometimes encountered in ranking problems where we want all the correct answers to be placed ahead of all the wrong ones. Here, we only care to put *one* correct answer on top; it does not matter which one, hence the max in (3.8).

3.4 STRUCTURED FORMULATION

Our choice of the loss L was motivated by intuitive arguments for what might correspond to a better proxy for the underlying task of k-NN prediction. However, it turns out that our problem is an instance of a familiar type of problems: latent structured prediction [169], and thus our choice of loss can be shown to form an upper bound on the empirical task loss Δ.

First, we note that the score $S_{\mathbf{W}}$ can be written as

$$S_{\mathbf{W}}(\mathbf{x}, h) = \left\langle \mathbf{W}, -\sum_{\mathbf{x}_j \in h} (\mathbf{x} - \mathbf{x}_j)(\mathbf{x} - \mathbf{x}_j)^T \right\rangle, \tag{3.9}$$

where $\langle \cdot, \cdot \rangle$ stands for the Frobenius inner product. Defining the *feature map*

$$\mathbf{\Psi}(\mathbf{x}, h) \triangleq -\sum_{\mathbf{x}_j \in h} (\mathbf{x} - \mathbf{x}_j)(\mathbf{x} - \mathbf{x}_j)^T, \tag{3.10}$$

we get a more compact expression $\langle \mathbf{W}, \mathbf{\Psi}(\mathbf{x}, h) \rangle$ for (3.9).

Going a step further, we can encode the deterministic dependence between y and h by a so-called "compatibility" function

$$A(y,h) = \begin{cases} 0 & \text{if } y = \widehat{y}(h) \\ -\infty & \text{otherwise} \end{cases}$$

This notion of compatibility allows us to write the joint inference of y and (hidden) h performed by kNN classifier as

$$\widehat{y}_{\mathbf{W}}(\mathbf{x}), \widehat{h}_{\mathbf{W}}(\mathbf{x}) = \operatorname*{argmax}_{h,y} \left[A(y,h) + \langle \mathbf{W}, \mathbf{\Psi}(\mathbf{x},h) \rangle \right]. \tag{3.11}$$

This is the familiar form of inference in a latent structured model [44, 169] with latent variable h. So, notwithstanding the somewhat unusual property of our model where the latent h completely determines the inferred y, we can show the equivalence to "normal" latent structured prediction.

3.4.1 *Learning by gradient descent*

We define the objective in learning $\mathbf{W}$ as

$$\min_{\mathbf{W}} \|\mathbf{W}\|_F^2 + C \sum_i L\left(\mathbf{x}_i, y_i, \mathbf{W}\right), \tag{3.12}$$

where $\|\cdot\|_F^2$ stands for Frobenius norm of a matrix.[1] The regularizer is convex, but as in other latent structured models, the loss L is non-convex due to the subtraction of the max in (3.8). To optimize (3.12), one can use the convex-concave procedure (CCCP) [170] which has been proposed specifically for latent SVM learning [169]. However, CCCP tends to be slow on large problems. Furthermore, its use is complicated here due to the requirement that $\mathbf{W}$ be positive semidefinite (PSD). This means that the inner loop of CCCP includes solving a semidefinite program, making the algorithm slower still. Instead, we opt for a much faster, and perhaps simpler, choice: stochastic gradient descent (SGD), described in Algorithm 1.

Algorithmus 1 : Stochastic gradient descent

```
**Input :** labeled data set $(X, Y)$, regularization parameter $C$, learning rate $\eta(\cdot)$
initialize $\mathbf{W}^{(0)} = \mathbf{0}$
**for** $t = 0, \ldots,$ *while not converged* **do**
    sample $i \sim [N]$
    $\widehat{h}_i = \text{argmax}_h \left[ S_{\mathbf{W}^{(t)}}(\mathbf{x}_i, h) + \Delta(y_i, h) \right]$
    $h_i^* = \text{argmax}_{h:\Delta(y_i,h)=0} S_{\mathbf{W}^{(t)}}(\mathbf{x}_i, h)$
    $\delta\mathbf{W} = \left[ \frac{\partial S_{\mathbf{W}}(\mathbf{x}_i, \widehat{h}_i)}{\partial \mathbf{W}} - \frac{\partial S_{\mathbf{W}}(\mathbf{x}_i, h_i^*)}{\partial \mathbf{W}} \right] \Bigg|_{\mathbf{W}^{(t)}}$
    $\mathbf{W}^{(t+1)} = (1 - \eta(t))\mathbf{W}^{(t)} - C\delta\mathbf{W}$
    project $\mathbf{W}^{(t+1)}$ to PSD cone
```

1 We discuss other choices of regularizer in Section 3.6.

The SGD algorithm requires solving two inference problems ($\widehat{h}$ and h^*), and computing the gradient of $S_{\mathbf{W}}$ which we address below.[2]

Algorithmus 2 : Targeted inference

```
Input : $\mathbf{x}$, $\mathbf{W}$, target class $y$, $\tau \triangleq [\![\text{ties forbidden}]\!]$
Output : $\text{argmax}_{h:\widehat{y}(h)=y} S_{\mathbf{W}}(\mathbf{x})$
Let $n^* = \lceil \frac{k+\tau(R-1)}{R} \rceil$        // min. required number of neighbors from $y$
$h := \varnothing$
for $j = 1, \ldots, n^*$ do
    $h := h \cap \underset{\mathbf{x}_i:\, y_i=y, i\notin h}{\text{argmin}} \; D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_i)$
for $l = n^*+1, \ldots, k$ do
    define $\#(r) \triangleq |\{i : \mathbf{x}_i \in h, y_i = r\}|$ // count selected neighbors from class $r$
    $h := h \cap \underset{\mathbf{x}_i:\, y_i=y, \text{ or } \#(y_i)<\#(y)-\tau, i\notin h}{\text{argmin}} \; D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_i)$
return $h$
```

3.4.1.1 *Targeted inference of h_i^**

Here we are concerned with finding the highest-scoring h constrained to be compatible with a given target class y. We give an $O(N \log N)$ algorithm in Algorithm 2. Proof of its correctness and complexity analysis is in section 3.4.1.4.

The intuition behind Algorithm 2 is as follows. For a given combination of R (number of classes) and k (number of neighbors), the minimum number of neighbors from the target class y required to allow (although not guarantee) zero loss, is n^* (see Proposition 1 in section 3.4.1.4). The algorithm first includes n^* highest scoring neighbors from the target class. The remaining $k - n^*$ neighbors are picked by a greedy procedure that selects the highest scoring neighbors (which might or might not be from the target class) while making sure that no non-target class ends up in a majority.

When using Alg. 2 to find an element in H^*, we forbid ties, i.e. set $\tau = 1$.

3.4.1.2 *Loss augmented inference $\widehat{h}_i$*

Calculating the max term in (3.7) is known as loss augmented inference. We note that

$$\max_{h'} \langle \mathbf{W}, \mathbf{\Psi}(\mathbf{x}, \mathbf{h}') \rangle + \Delta(y, h') = \max_{y'} \Big\{ \underbrace{\max_{h' \in H^*(y')} \langle \mathbf{W}, \mathbf{\Psi}(\mathbf{x}, \mathbf{h}') \rangle}_{=\langle \mathbf{W}, \mathbf{\Psi}(\mathbf{x}, \mathbf{h}^*(\mathbf{x}, \mathbf{y}')) \rangle} + \Lambda(y, y') \Big\} \quad (3.13)$$

which immediately leads to Algorithm 3, relying on Algorithm 2. The intuition: perform targeted inference for each class (as if that were the target class), and the choose the set of neighbors for the class for which the loss-augmented score is the highest. In this case, in each call to Alg. 2 we set $\tau = 0$, i.e., we allow ties, to make sure the argmax is over all possible h's.

2 We note that both inference problems over h are done in leave one out settings, i.e., we impose an additional constraint $i \notin h$ under the argmax, not listed in the algorithm explicitly.

```
Algorithmus 3 : Loss augmented inference
  Input : $\mathbf{x}$, $\mathbf{W}$,target class $y$
  Output : $\text{argmax}_h \left[S_{\mathbf{W}}(\mathbf{x}, h) + \Delta(y, h)\right]$
  for $r \in \{1, \dots, R\}$ do
      $h^{(r)} := h^*(\mathbf{x}, \mathbf{W}, r, 1)$                    // using Alg. 2
      Let Value $(r) := S_{\mathbf{W}}(\mathbf{x}, h^{(r)}), + \Lambda(y, r)$
  Let $r^* = \text{argmax}_r$Value $(r)$
  return $h^{(r^*)}$
```

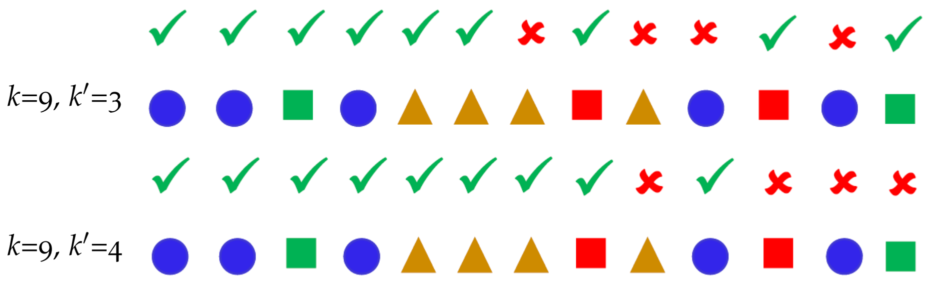


Figure 3.2: Inference as packing k neighbors with smallest distance while ensuring correct vote. For more details see sections 3.4.1 and 3.4.1.3

3.4.1.3 *Some more intuition behind inference procedures*

While in the previous section we have provided inference procedures and section 3.4.1.4 contains the analysis and proof of correctness, in this section we briefly provide more intuition. We can view of inference in this model as wanting to pack k neighbors with the smallest distance, while ensuring the correct vote. For instance, suppose we want $k \leq k'$ neighbors from a target class. In that case, we start with k' nearest neighbors from the target class, and then proceed in the order of decreasing distance. While doing so, we also pick neighbors from the wrong class, but such that we also ensure that do not let any class have $\geq k'$ points. This is then done for all feasible values of k', and from this we select the best set. This is illustrated further in figure 3.2

3.4.1.4 *Analysis and Proof of correctness of Algorithm 2*

First of all it is easy to see that Algorithm 2 terminates. There are $k - n^*$ iterations after initialization (of the first n^* points) and this amounts to at most a linear scan of X. We need $O(N \log N)$ time to sort the data and then finding $\mathbf{h}^*$ involves $O(N)$, thus the algorithm runs in time $O(N \log N)$.

We need to prove that the algorithm returns h^* as defined earlier. First, we establish the correctness of setting n^*:

Proposition 1. *Let R be the number of classes, and let $\#(h, y)$ be the count of neighbors from target class y included in the assignment h. Then, $\Delta(y^*, h) = 0$ only if $\#(h, y^*) \geq n^*$, where*

$$n^* = \begin{cases} \left\lceil \frac{k+R-1}{R} \right\rceil & \textit{if ties not allowed,} \\ \left\lceil \frac{k}{R} \right\rceil & \textit{if ties allowed.} \end{cases}$$

We prove it below for the case with no ties; the proof when ties are allowed is very similar.

Proof. Suppose by contradiction that $\Delta(y^*, h) = 0$ and $\#(h, y^*) \leq \lceil \frac{k+R-1}{R} \rceil - 1$. Then, since no ties are allowed, for all $y \neq y^*$, we have $\#(h, y^*) \leq \lceil \frac{k+R-1}{R} \rceil - 2$, and

$$\sum_y \#(h, y) \leq (R-1) \left(\left\lceil \frac{k+R-1}{R} \right\rceil - 2 \right) \tag{3.14}$$

$$+ \left\lceil \frac{k+R-1}{R} \right\rceil - 1 \tag{3.15}$$

$$< k, \tag{3.16}$$

a contradiction to $|h| = k$. □

Next, we prove that the algorithm terminates and produces a correct result. For the purposes of complexity analysis, we consider R (but not k) to be constant, and number of examples from each class to be $O(N)$.

Claim 1. *Algorithm 2 terminates after at most $O((N+k)\log N)$ operations and produces an h such that $|h| = k$.*

Proof. The elements of X can be held in R priority queues, keyed by $D_{\mathbf{W}}$ values, one queue per class. Construction of this data structure is an $O(N \log N)$ operation, carried out before the algorithm starts. To initialize h with n^* values, the algorithm retrieves n^* top elements from the priority queue for class y^*. An $O(n^* \log N)$ operation. Then, for each of the iterations over l, the algorithm needs to examine at most one top element from R queues, which costs $O(\log N)$; each such iteration increases $|h|$ by one. Thus after $k - n^*$ iterations $|h| = k$; the total cost is thus $O(k \log N)$. Combined with the complexity of data structure construction mentioned above, this concludes the proof. □

Note that for typical scenarios in which $N \gg k$, the cost will be dominated by the $N \log N$ data structure setup.

Claim 2. *Let h^* be returned by Algorithm 2. Then,*

$$h^* = \operatorname*{argmax}_{h:|h|=k,\, \Delta(y^*,h)=0)} S_{\mathbf{W}}(\mathbf{x}, h), \tag{3.17}$$

i.e., the algorithm finds the highest scoring h with total of k neighbors among those h that attain zero loss.

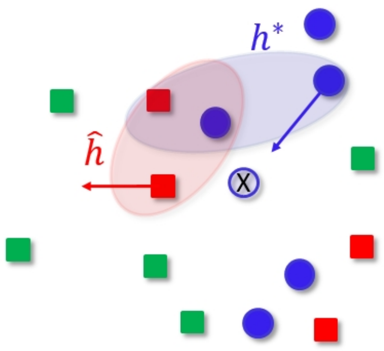


Figure 3.3: Illustration of interpretation of the gradient update, more details in the text

Proof. From Proposition 1 we know that if $\#(h, y) < n^*$, then h does not satisfy the $\Delta(\mathbf{x}, h) = 0$ condition. $|h| \geq n^*$ to (3.17) without altering the definition.

We will call h "optimal for l" if

$$h = \operatorname*{argmax}_{h:|h|=n^*+l, \#(h,y)\geq n^*, \Delta(y^*,h))=0} S_{\mathbf{W}}(\mathbf{x}, h).$$

We now prove by induction over l that this property is maintained through the loop over l in the algorithm.

Let $h^{(j)}$ denote choice of h after j iterations of the loop, i.e., $|h| = n^* + j$. Suppose that $h^{(l-1)}$ is optimal for $l - 1$. Now the algorithm selects $\mathbf{x}_a \in X$, such that

$$\mathbf{x}_a = \operatorname*{argmin}_{\mathbf{x}_i: y_i=y, \text{ or } \#(y_i)<\#(y)-\tau, \mathbf{x}_i \notin h} D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_i). \tag{3.18}$$

Suppose that $h^{(l)}$ is not optimal for l. Then there exists an $\mathbf{x}_b \in X$ for which $D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_b) < D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_a)$ such that picking $\mathbf{x}_b$ instead of $\mathbf{x}_a$ would produce h optimal for l. But $\mathbf{x}_b$ is not picked by the algorithm; this can only happen if conditions on the argmin in (3.18) are violated, namely, if $\#(y_b) = \#(y) - \tau$; therefore picking $\mathbf{x}_b$ would violate conditions of optimality of $h^{(l)}$, and we get a contradiction.

It is also clear that after initialization with k highest scoring neighbors in y^*, h is optimal for $l = 0$, which forms the base of induction. We conclude that $h^{(k-n^*)}$, i.e. the result of the algorithm, is optimal for $k - n^*$, which is equivalent to definition in (3.17).

□

3.4.1.5 *Gradient update*

Finally, we need to compute the gradient of the distance score. Since it is linear in $\mathbf{W}$ as shown in (3.9), we have

$$\frac{\partial S_{\mathbf{W}}(\mathbf{x}, h)}{\partial \mathbf{W}} = \mathbf{\Psi}(\mathbf{x}, h) = -\sum_{\mathbf{x}_j \in h} (\mathbf{x} - \mathbf{x}_j)(\mathbf{x} - \mathbf{x}_j)^T. \tag{3.19}$$

Thus, the update in Alg 1 has a simple interpretation, illustrated in Fig 3.1. For every $\mathbf{x}_i \in h^* \setminus \widehat{h}$, it "pulls" $\mathbf{x}_i$ closer to $\mathbf{x}$. For every $\mathbf{x}_i \in \widehat{h} \setminus h^*$, it "pushes" it farther from $\mathbf{x}$; these push and pull refer to increase/decrease of Mahalanobis distance under the updated $\mathbf{W}$. Any other $\mathbf{x}_i$, including any $\mathbf{x}_i \in h^* \cap \widehat{h}$, has no influence on the update. This is a difference of our approach from LMNN, MLR etc. This is illustrated in Figure 3.1. In particular h^* corresponds to points a, c and e, whereas $\widehat{h}$ corresponds to points c, e and f. Thus point a is pulled while point f is pushed.

Since the update does not necessarily preserve $\mathbf{W}$ as a PSD matrix, we enforce it by projecting $\mathbf{W}$ onto the PSD cone, by zeroing negative eigenvalues. Note that since we update (or "downdate") $\mathbf{W}$ each time by matrix of rank at most $2k$, the eigendecomposition can be accomplished more efficiently than the naïve $O(d^3)$ approach, e.g., as in [139].

Using first order methods, and in particular gradient methods for optimization of non-convex functions, has been common across machine learning, for instance in training deep neural networks. Despite lack (to our knowledge) of satisfactory guarantees of convergence, these methods are often successful in practice; we will show in the next section that this is true here as well. However, care should be taken to ensure validity of the method, and we discuss this briefly before reporting on experiments.

A given $\mathbf{x}$ imposes a Voronoi-type partition of the space of $\mathbf{W}$ into a finite number of cells; each cell is associated with a particular combination of $\widehat{h}(\mathbf{x})$ and $h^*(\mathbf{x})$ under the values of $\mathbf{W}$ in that cell. The score $S_{\mathbf{W}}$ is differentiable (actually linear) on the interior of the cell, but may be non-differentiable (though continuous) on the boundaries. Since the boundaries between a finite number of cells form a set of measure zero, we see that the score is differentiable almost everywhere.

3.5 EXPERIMENTS

We compare the error of kNN classifiers using metrics learned with our approach to that with other learned metrics. For this evaluation we replicate the protocol in [71], using the seven data sets in Table 3.1. For all data sets, we report error of kNN classifier for a range of values of k; for each k, we test the metric learned for that k. Competition to our method includes Euclidean Distance, LMNN [157], NCA, [50], ITML [30], MLR [107] and GB-LMNN [71]. The latter learns non-linear metrics rather than Mahalanobis.

For each of the competing methods, we used the code provided by the authors. In each case we tuned the parameters of each method, including ours, in the same cross-validation protocol. We omit a few other methods that were consistently shown in literature to be dominated by the ones we compare to, such as χ^2 distance, MLCC, M-LMNN. We also could not include χ^2-LMNN since code for it is not available; however published results for $k = 3$ [71] indicate that our method would win against χ^2-LMNN as well.

Isolet and USPS have a standard training/test partition, for the other five data sets, we report the mean and standard errors of 5-fold cross validation (results for all methods are on the same folds). We experimented with different methods for initializing our method (given the non-convex objective), including the euclidean distance, all zeros etc. and found the euclidean initialization to be always worse. We initialize each fold with either the diagonal matrix learned by ReliefF [75] (which gives a scaled euclidean distance) or all zeros depending on whether the scaled euclidean distance obtained using ReliefF was better than unscaled euclidean distance. In each experiment, $\mathbf{x}$ are

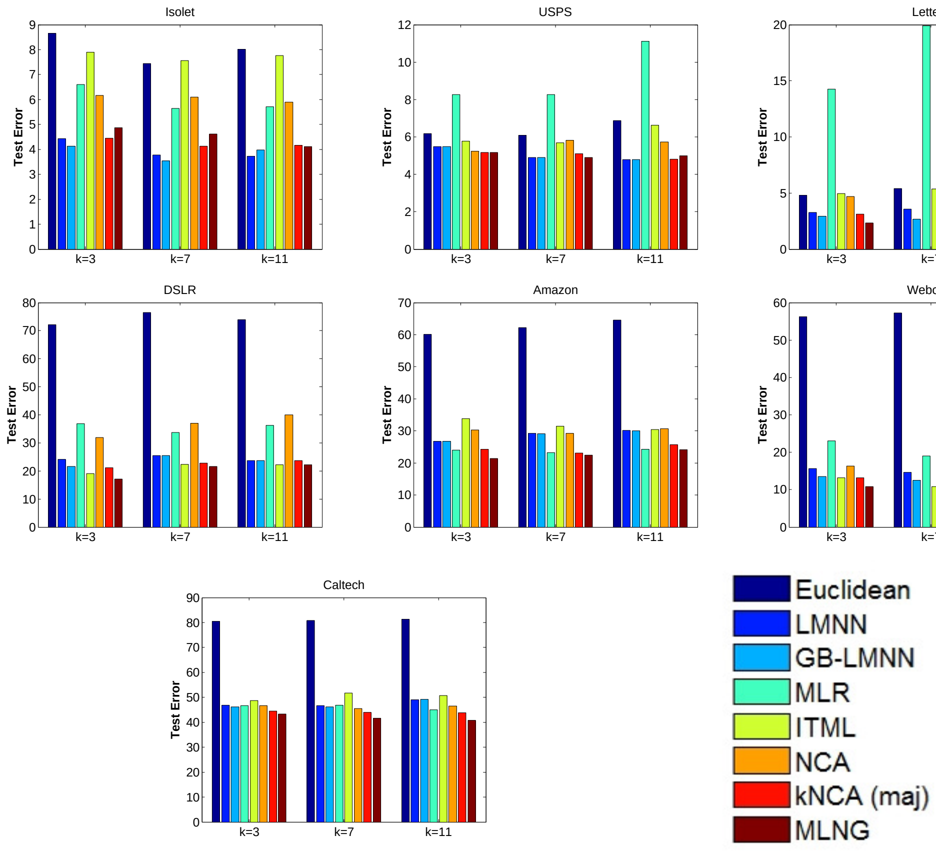


Figure 3.4: *k*NN errors for *k*=3, 7 and 11 on various datasets when the features are scaled by z-scoring

scaled by mean and standard deviation of the training portion.[3] The value of C is tuned on on a 75%/25% split of the training portion. Results using different scaling methods are also reported.

Our SGD algorithm stops when the running average of the surrogate loss over most recent epoch no longer decreases substantially, or after max. number of iterations. We use learning rate $\eta(t) = 1/t$.

The results show that our method dominates other competitors, including non-linear metric learning methods, and in some cases achieves results significantly better than those of the competition. The results for the initialization and set-up mention above are illustrated in table 3.1, as well as figure 3.4

3.5.1 *Runtimes using different methods*

Here we include the training times in seconds for one fold of each dataset. These timings are for a single partition, for optimal parameters for $k = 7$. These experiments were run on a 12-core Intel Xeon E5-2630 v2 @ 2.60GHz and are reported in 3.2. We notice that the approach, while competitive is quite slow due to exact inference and loss augmented

3 For Isolet we also reduce dimensionality to 172 by PCA computed on the training portion.

k = 3

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
d	170	256	16	800	800	800	800
N	7797	9298	20000	157	958	295	1123
C	26	10	26	10	10	10	10
Euclidean	8.66	6.18	4.79 ±0.2	75.20 ±3.0	60.13 ±1.9	56.27 ±2.5	80.5 ±4.6
LMNN	4.43	5.48	3.26 ±0.1	24.17 ±4.5	26.72 ±2.1	15.59 ±2.2	46.93 ±3.9
GB-LMNN	**4.13**	5.48	2.92 ±0.1	21.65 ±4.8	26.72 ±2.1	13.56 ±1.9	46.11 ±3.9
MLR	6.61	8.27	14.25 ±5.8	36.93 ±2.6	24.01 ±1.8	23.05 ±2.8	46.76 ±3.4
ITML	7.89	5.78	4.97 ±0.2	19.07 ±4.9	33.83 ±3.3	13.22 ±4.6	48.78 ±4.5
1-NCA	6.16	5.23	4.71 ±2.2	31.90 ±4.9	30.27 ±1.3	16.27 ±1.5	46.66 ±1.8
k-NCA	4.45	**5.18**	3.13 ±0.4	21.13 ±4.3	24.31 ±2.3	13.19 ±1.3	44.56 ±1.7
ours	4.87	**5.18**	**2.32 ±0.1**	**17.18** ±4.7	**21.34** ±2.5	**10.85** ±3.1	**43.37** ±2.4

k = 7

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
Euclidean	7.44	6.08	5.40 ±0.3	76.45 ±6.2	62.21 ±2.2	57.29 ±6.3	80.76 ±3.7
LMNN	3.78	**4.9**	3.58 ±0.2	25.44 ±4.3	29.23 ±2.0	14.58 ±2.2	46.75 ±2.9
GB-LMNN	**3.54**	**4.9**	2.66 ±0.1	25.44 ±4.3	29.12 ±2.1	12.45 ±4.6	46.17 ±2.8
MLR	5.64	8.27	19.92 ±6.4	33.73 ±5.5	23.17 ±2.1	18.98 ±2.9	46.85 ±4.1
ITML	7.57	5.68	5.37 ±0.5	22.32 ±2.5	31.42 ±1.9	**10.85 ±3.1**	51.74 ±2.8
1-NCA	6.09	5.83	5.28 ±2.5	36.94 ±2.6	29.22 ±2.7	22.03 ±6.5	45.50 ±3.0
k-NCA	4.13	5.1	3.15 ±0.2	22.78 ±3.1	23.11 ±1.9	13.04 ±2.7	43.92 ±3.1
ours	4.61	**4.9**	**2.54 ±0.1**	**21.61 ±5.9**	**22.44** ±1.3	11.19 ±3.3	**41.61 ±2.6**

k = 11

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
Euclidean	8.02	6.88	5.89 ±0.4	73.87 ±2.8	64.61 ±4.2	59.66 ±5.5	81.39 ±4.2
LMNN	**3.72**	**4.78**	4.09 ±0.1	23.64 ±3.4	30.12 ±2.9	13.90 ±2.2	49.06 ±2.3
GB-LMNN	3.98	**4.78**	**2.86 ±0.2**	23.64 ±3.4	30.07 ±3.0	13.90 ±1.0	49.15 ±2.8
MLR	5.71	11.11	15.54 ±6.8	36.25 ±13.1	24.32 ±3.8	17.97 ±4.1	44.97 ±2.6
ITML	7.77	6.63	6.52 ±0.8	**22.28 ±3.1**	30.48 ±1.4	11.86 ±5.6	50.76 ±1.9
1-NCA	5.90	5.73	6.04 ±2.8	40.06 ±6.0	30.69 ±2.9	26.44 ±6.3	46.48 ±4.0
k-NCA	4.17	4.81	3.87 ±0.6	23.65 ±4.1	25.67 ±2.1	11.42 ±4.0	43.8 ±3.1
ours	4.11	4.98	3.05 ±0.1	**22.28** ±4.9	**24.11 ±3.2**	**11.19** ±4.4	**40.76 ±1.8**

Table 3.1: *k*NN errors for *k*=3, 7 and 11. Features were scaled by z-scoring.

inference, each of which are expensive steps. The run time increases linearly both in the number of classes as well as *k*.

Dataset	DSLR	Caltech	Amazon	Webcam	Letters	USPS	Isolet
LMNN	358.11	1812.1	1545.1	518.7	179.77	782.66	1762.1
GB-LMNN	410.13	1976.4	1680.9	591.29	272.87	3672.9	2882.6
MLR	4.93	124.42	88.96	85.02	838.13	1281	33.20
MLNG	413.36	1027.6	2157.2	578.74	6657.3	3891.7	3668.9

Table 3.2: Comparison of runtimes

3.5.2 *Experimental results using different feature normalizations*

For the sake of completeness, we also experimented with different feature normalization other than z-scoring to see if it impacted the results significantly. Somewhat surprisingly, we did notice a deterioration in performance when no feature normalization was done. We report the results for the case of no feature normalization, for the same competition in table 3.4. We also ran the same set of experiments for the case when we did histogram normalization. In this case obviously, we could only run experiments in the case of 4 datasets: DSLR, Amazon, Webcam and Caltech, which allowed for such normalization. Results for experiments with such a feature normalization are reported in table 3.3

3.6 CONCLUSION AND SUMMARY OF WORK

In this part of the dissertation we proposed a formulation of the metric learning for kNN classifier as a structured prediction problem, with discrete latent variables representing the selection of k neighbors. We also provided efficient algorithms for exact inference in this model, including for loss augmented inference. While proposed in the context of metric learning, these procedures might be of wider interest. We also devised a stochastic gradient descent based procedure for learning in this model. The proposed approach allows us to learn a Mahalanobis metric with an objective which is a more direct proxy for the stated goal (improvement of classification by kNN rule) than previously proposed similarity learning methods. Our learning algorithm is simple yet efficient, converging on all the data sets we have experimented with in run-times significantly lesser or comparable than other methods, such as LMNN and MLR.

We used the Frobenius norm as the choice of the regularizer in our experiments. This was motivated by the intuition of wanting to do capacity control but without biasing our model towards any particular form. In our experiments, we have also experiments with other schemes for regularization such as using the trace norm of $\mathbf{W}$ and the shrinkage towards Euclidean distance, $\|\mathbf{W} - \mathbf{I}\|_F^2$, but found both to be inferior to $\|\mathbf{W}\|_F^2$. Our suspicion for such behavior is that often the optimal matrix that parameterizes the distance function i.e. $\mathbf{W}$ corresponds to a highly anisotropic scaling of data dimensions, and thus an initial bias towards $\mathbf{I}$ may be unhealthy.

The results in this section of the dissertation are restricted to learning the Mahalanobis metric, which is an appealing choice for a number of reasons. In particular, learning such metrics is equivalent to learning linear embedding of the data, allowing very efficient

k = 3

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
d	170	256	16	800	800	800	800
N	7797	9298	20000	157	958	295	1123
C	26	10	26	10	10	10	10
Euclidean	-	-	-	26.71 ±11	37.26 ±2.3	23.39 ±5.3	58.42 ±3.7
LMNN	-	-	-	23.53 ±7.6	**26.30 ±1.6**	**11.53 ±6.7**	43.72 ±3.5
GB-LMNN	-	-	-	23.53 ±7.6	**26.30 ±1.6**	**11.53 ±6.7**	**43.54 ±3.5**
MLR	-	-	-	24.78 ±14.2	32.35 ±4.5	14.58 ±3.5	52.18 ±2.0
ITML	-	-	-	22.22 ±9.9	32.67 ±3.2	12.88 ±6.1	51.74 ±4.2
NCA	-	-	-	29.84 ±8.1	33.72 ±2.1	21.36 ±4.9	54.50 ±2.0
ours	-	-	-	**21.63 ±6.1**	28.08 ±2.4	14.58 ±5.4	45.33 ±2.8

k = 7

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
Euclidean	-	-	-	32.46 ±8.3	38.2 ±1.6	27.46 ±5.9	56.9 ±2.9
LMNN	-	-	-	26.11 ±8.6	25.47 ±1.6	**10.51 ±4.9**	41.77 ±4.0
GB-LMNN	-	-	-	**25.48 ±10.9**	**25.36 ±1.7**	**10.51 ±4.9**	**41.59 ±3.6**
MLR	-	-	-	27.94 ±9.0	30.16 ±3.0	16.95 ±3.4	49.51 ±3.6
ITML	-	-	-	22.28 ±8.8	32.88 ±3.3	13.90 ±6.3	50.59 ±4.7
NCA	-	-	-	37.48 ±8.2	33.09 ±1.9	23.39 ±5.3	51.74 ±2.6
ours	-	-	-	**25.65 ±7.1**	27.24 ±2.7	17.29 ±5.0	44.62 ±2.6

k = 11

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
Euclidean	-	-	-	35.02 ±8.9	37.57 ±2.3	30.51 ±4.8	56.55 ±2.4
LMNN	-	-	-	49.64 ±5.7	**24.84 ±2.1**	**10.17 ±3.8**	43.19 ±2.7
GB-LMNN	-	-	-	43.89 ±5.6	25.16 ±2.0	**10.17 ±3.8**	**43.10 ±3.1**
MLR	-	-	-	28.63 ±7.7	30.48 ±2.4	17.63 ±5.3	48.18 ±3.8
ITML	-	-	-	**24.82 ±5.1**	31.10 ±2.6	15.25 ±6.3	50.32 ±3.9
NCA	-	-	-	41.37 ±4.7	32.88 ±1.5	24.07 ±8.4	51.20 ±3.9
ours	-	-	-	31.79 ±7.2	28.49 ±2.8	17.65 ±3.5	45.95 ±4.8

Table 3.3: kNN error,for k=3, 7 and 11. Mean and standard deviation are shown for data sets on which 5-fold partition was used. These experiments were done after histogram normalization. Best performing methods are shown in bold. Note that the only non-linear metric learning method in the above is GB-LMNN

methods for metric search. As mentioned in section 3.2, we can consider more general notions of distance:

$$D_{\mathcal{W}}(\mathbf{x}, \mathbf{x}_i) = \|\Phi(\mathbf{x}; \mathcal{W}) - \Phi(\mathbf{x}_i; \mathcal{W})\|_2^2$$

k = 3

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
d	170	256	16	800	800	800	800
N	7797	9298	20000	157	958	295	1123
C	26	10	26	10	10	10	10
Euclidean	8.98	5.03	4.31 ±0.2	58.01 ±5.0	56.89 ±2.4	40.34 ±4.2	74.89 ±3.2
LMNN	4.17	5.38	3.26 ±0.1	23.53 ±5.6	28.08 ±2.2	**11.19 ±5.6**	44.97 ±2.6
GB-LMNN	**3.72**	5.03	2.50 ±0.2	23.53 ±5.6	28.08 ±2.2	11.53 ±5.5	44.70 ±2.4
MLR	17.32	8.42	45.70 ±18.7	35.69 ±7.6	**23.40 ±1.7**	20 ±4.6	47.11 ±1.7
ITML	6.86	**4.78**	4.35 ±0.2	24.82 ±10.9	34.77 ±4.7	12.20 ±4.1	53.97 ±3.2
NCA	5.07	5.18	4.39 ±1.1	24.19 ±5.8	29.54 ±1.4	12.88 ±4.9	46.84 ±2.0
ours	4.11	5.13	**2.24 ±0.1**	**21.01 ±4.1**	26.20 ±2.6	13.56 ±4.6	**44.54 ±2.9**

k = 7

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
Euclidean	6.93	5.08	4.69 ±0.2	60.46 ±5.2	59.07 ±4.5	43.05 ±3.7	72.3 ±3.3
LMNN	4.04	5.28	3.53 ±0.2	24.15 ±9.0	28.19 ±2.8	13.56 ±4.5	43.90 ±2.4
GB-LMNN	**3.72**	5.03	**2.32 ±0.2**	24.80 ±8.1	28.29 ±3.1	13.14 ±5.8	43.54 ±2.2
MLR	23.28	8.12	33.61 ±16.8	38.17 ±10.9	**23.79 ±3.9**	20.34 ±2.9	45.60 ±4.8
ITML	5.90	5.23	4.93 ±0.5	**23.57 ±9.6**	32.46 ±3.2	**11.19 ±5.7**	52.63 ±3.3
NCA	5.52	4.98	5.06 ±1.1	37.58 ±5.7	31.01 ±2.0	16.81 ±5.9	43.90 ±2.4
ours	4.07	**4.93**	**2.49 ±0.1**	29.94 ±7.6	26.10 ±2.1	13.24 ±3.1	**42.83 ±3.1**

k = 11

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
Euclidean	7.95	5.68	5.26 ±0.2	61.71 ±6.4	61.48 ±3.7	49.15 ±3.9	73.1 ±3.6
LMNN	**3.85**	5.73	4.09 ±0.2	49.6 ±5.5	27.04 ±1.8	14.58 ±4.6	44.61 ±1.3
GB-LMNN	3.98	6.33	2.96 ±0.1	45.18 ±10.5	27.25 ±2.2	14.58 ±4.6	45.55 ±6.9
MLR	33.61	10.26	35.50 ±16.5	34.40 ±8.2	**24.21 ±3.4**	18.31 ±5.3	46.04 ±1.9
ITML	7.18	5.88	5.35 ±0.3	**28.04 ±7.7**	33.09 ±2.1	**12.54 ±5.4**	51.91 ±3.3
NCA	5.52	5.03	5.8 ±1.3	45.18 ±6.5	32.47 ±1.7	19.32 ±7.5	**44.17 ±2.6**
ours	3.87	**4.98**	**2.8 ±0.2**	33.00 ±5.7	26.10 ±2.7	14.24 ±6.5	45.76 ±2.9

Table 3.4: kNN error,for k=3, 7 and 11. No feature scaling was applied in these experiments. Mean and standard deviation are shown for data sets on which 5-fold partition was used. Best performing methods are shown in bold. Note that the only non-linear metric learning method in the above is GB-LMNN.

Where $\Phi(\mathbf{x}; \mathcal{W})$ is a map (possibly non-linear), $\mathbf{x} \to \Phi(\mathbf{x})$, parameterized by $\mathcal{W}$.

In such cases, learning S when the map is non-linear can be seen as optimizing for a kernel with discriminative objective of improving kNN performance. Such a model would be more expressive, and using methods for automatic differentiation should be straightforward to optimize.

In the next chapter of this disseration, we explore some extensions of this approach. First, we consider the case when the query and database point are mapped to different subspaces. This was inspired by the work of Neyshabur *et al.* on asymmetric hashing [114] [115]. Next, we leverage the objective for the discriminative learning of Hamming distance, which gives us compact binary representations for fast retrieval. Lastly, we modify the inference procedures to make the approach amenable to learn suitable metrics for the (harder) case of *k*-NN regression.

4 EXTENSIONS

OUTLINE
The main contribution of the *Neighborhood Gerrymandering* method was primarily in its *loss* and inference procedures; its use for Mahalanobis metric learning being just one use case. In this chapter this wider applicability is demonstrated in three different settings: Asymmetric similarity learning, metric learning when the labels are continuous and finally learning compact binary codes that are similarity sensitive in Hamming space.

In the previous chapter, the problem of metric learning for k-NN classification was formulated as a large margin structured prediction problem, with the choice of neighbors represented by discrete latent variables. Efficient algorithms for exact inference and loss-augmented inference in this model (dubbed as *neighborhood gerrymandering*) were provided, and it was argued; with supporting experiments, that this formulation gave a more direct proxy for nearest neighbor classification as compared to prior art in metric learning. It was also noted that while the method was only tested in the case of learning Mahalanobis distances for points living in an explicit feature space, the methodology was more generally applicable.

To impress upon this point, we again consider the distance computation that might be used:

$$D_{\mathcal{W}}(\mathbf{x}, \mathbf{x}_i) = \|\Phi(\mathbf{x}; \mathcal{W}) - \Phi(\mathbf{x}_i; \mathcal{W})\|_2^2 \tag{4.1}$$

Note that there is considerable freedom in choosing the map $\Phi(\mathbf{x}; \mathcal{W})$, and while the experiments reported in Chapter 3 were specifically for the Mahalanobis distance i.e. $\mathbf{x} \rightarrow \mathbf{L}\mathbf{x}$ such that $\mathbf{L}^T\mathbf{L} = \mathbf{W} \succeq 0$, the map $\mathbf{x} \rightarrow \Phi(\mathbf{x}; \mathcal{W})$ could be non-linear, with $\mathcal{W}$ representing the parameters of a deep neural network. With the same structured formulation, the procedures for inference and loss-augmented inference would remain the same, and the optimization would involve training a Siamese-like network while optimizing the *gerrymandering* objective. This direction was explored by dissertation author, however, in this chapter we present work on three somewhat different directions.

1 **Asymmetric Similarity Computation:** Note that in equation 4.1, the *query* point $\mathbf{x}$ and the *database* point $\mathbf{x}_i$ involve the same transformation parameterized by $\mathcal{W}$. But it is clear that this need not be the case, indeed, we might modify 4.1 to the following form:

$$D_{\mathcal{W}}(\mathbf{x}, \mathbf{x}_i) = \|\Phi(\mathbf{x}; \mathcal{W}) - \Phi'(\mathbf{x}_i; \mathcal{W}')\|_2^2 \tag{4.2}$$

while ensuring that Φ and Φ' maps to the same metric space i.e. $\Phi, \Phi' : \mathbb{R}^d \rightarrow \mathbb{R}^p$. In particular, we work with linear transformations: we transform the query point as $\mathbf{x} \rightarrow \mathbf{U}\mathbf{x}$, and the database point as $\mathbf{x} \rightarrow \mathbf{V}\mathbf{x}$, with $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d \times d}$. This approach, its motivation and empirical validation is discussed in section 4.1.

2 **Similarity Computation in Hamming Space:** As discussed in chapter 2, nearest neighbor methods are limited by two factors: first, the choice of the metric defining "nearest", and second, efficient indexing facilitating fast retrieval from large datasets. In chapter 3, as well as in sections 4.1 and 4.3, we only focus on the former. The latter, however, is important for the success of nearest neighbor methods as well. This is often done by generating compact binary codes for the data in either a supervised or unsupervised fashion. In section 4.2 we discuss an approach that generates binary codes while optimizing for the nearest neighbor performance in Hamming space using the gerrymandering objective. In particular, $\Phi : \mathbb{R}^d \rightarrow \mathcal{H}$, where $\mathcal{H}$ is the Hamming space, containing 2^c binary codes of length c. The Hamming space is endowed with a metric, the Hamming distance, which implies that with a good choice of Φ, we can use the same inference procedures as in chapter 3 to optimize for binary codes such that the nearest neighbor classification performance in Hamming space is improved.

3 **Metric Learning with Continuous Labels:** In preceding discussions we have worked with an instance space $(\mathcal{X}, d)$, which is a metric space, and a discrete label space $\mathcal{Y} \in \mathbb{Z}_+$. We assume an unknown, smooth function $f : \mathcal{X} \rightarrow \mathcal{Y}$, and try to optimize for the metric such that accuracy of the k-nearest neighbor classifier is improved. We now change tack and work with the case when the labels are not discrete i.e. $\mathcal{Y} \in \mathbb{R}$. Therefore, the problem becomes that of optimizing for a metric such that k-nearest neighbor regression performance improves. This problem is also somewhat different than the previous two in that the inference and loss-augmented inference procedures, which worked for discrete labels, are no longer directly applicable. Thus we need to suitably modify them in order to make inference tractable. This approach is discussed further in section 4.3.

In the following three sections, we develop the ideas outlined above in detail.

4.1 ASYMMETRIC METRIC LEARNING

Recall that for some $\mathbf{W} \succeq 0$, we could factorize it as $\mathbf{W} = \mathbf{L}^T\mathbf{L}$. Thus, the squared distance may be written as:

$$D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} - \mathbf{x}_i)^T\mathbf{W}(\mathbf{x} - \mathbf{x}_i) = \|\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}_i\|_2^2 = (\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}_i)^T(\mathbf{L}\mathbf{x} - \mathbf{L}x_i) \tag{4.3}$$

We can thus think of the metric learning problem as learning the same projection matrix $\mathbf{L}$ for both the query (denoted $\mathbf{x}$) and the database points (denoted $\mathbf{x}_i$). In this section, we instead consider the following alternative formulation instead:

$$D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{U}\mathbf{x} - \mathbf{V}\mathbf{x}_i\|_2^2 = (\mathbf{U}\mathbf{x} - \mathbf{V}\mathbf{x}_i)^T(\mathbf{U}\mathbf{x} - \mathbf{V}x_i) \tag{4.4}$$

In this formulation, all the points are not subject to the same global transformation: the query and the database points are (linearly) projected separately, thus making the distance computation asymmetric.

The above (Eq. 4.4) may be rewritten as follows:

$$D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_i) = \begin{bmatrix} \mathbf{x} & \mathbf{x}_i \end{bmatrix} \begin{bmatrix} \mathbf{U}^T\mathbf{U} & -\mathbf{U}^T\mathbf{V} \\ -\mathbf{V}^T\mathbf{U} & \mathbf{V}^T\mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_i \end{bmatrix} \tag{4.5}$$

Therefore, the problem of learning $\mathbf{U}$ and $\mathbf{V}$ as specified in the similarity computation of 4.4 is equivalent to learning a matrix $\mathbf{W} \in \mathbb{R}^{2d \times 2d}$ such that $\mathbf{W} \succeq 0$, with

$$\mathbf{W} = \begin{bmatrix} \mathbf{U}^T\mathbf{U} & -\mathbf{U}^T\mathbf{V} \\ -\mathbf{V}^T\mathbf{U} & \mathbf{V}^T\mathbf{V} \end{bmatrix}$$

We will return to the formulation of the metric learning problem in this setting in the next section. But before doing so, it perhaps might be pertinent to point out the motivation for this approach.

The idea of subjecting the query and database points to different projections was inspired by work on hashing [114] [115], and was explored by the dissertation author [146] with the first author of [114][115], immediately after the publication of [147]. The main (and somewhat counterintuitive) message of [115] was the following: Usually, the similarity $S(\mathbf{x}, \mathbf{x}_i)$ for query point $\mathbf{x}$ and database point $\mathbf{x}_i$, is approximated by the Hamming distance between the outputs of the same hash function $f(\mathbf{x})$ and $f(\mathbf{x}_i)$, for some $f \in \{\pm 1\}^k$. Now, instead of using the same hash function for both the query and database points, suppose two distinct functions $f(\mathbf{x})$ and $g(\mathbf{x}_i)$ were used instead, then, even in cases where the target similarity happens to be symmetric, this asymmetry in the similarity computation affords representational advantages and reduces code length. Thus a natural question to consider was to see if asymmetry offered any advantages in the case of discriminative metric learning as well. With this brief background on the motivation, we now proceed to formulate the problem using the *gerrymandering* formalism described in the previous chapter.

4.1.1 *Formulation*

Coming back to the distance computation in equations 4.4 and 4.5: given $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{d \times d}$, for any $h \subset \mathbf{X}$ with $|h| = k$, we can define the similarity between $\mathbf{x}$ and h in direct analogy with that in the previous chapter:

$$S_{\mathbf{U},\mathbf{V}}(\mathbf{x}, h) = -\sum_{\mathbf{x}_i \in h} (\mathbf{U}\mathbf{x} - \mathbf{V}\mathbf{x}_i)^T (\mathbf{U}\mathbf{x} - \mathbf{V}\mathbf{x}_i) \tag{4.6}$$

Likewise, we can use the above measure of similarity to define the following surrogate loss for k-NN classification:

$$L(\mathbf{x}, y, \{\mathbf{U}, \mathbf{V}\}) = \max_h \left[S_{\mathbf{U},\mathbf{V}}(\mathbf{x}, h) + \Delta(y, h)\right] - \max_{h:\Delta(y,h)=0} S_{\mathbf{U},\mathbf{V}}(\mathbf{x}, h) \tag{4.7}$$

Given the loss formulation, we are now left with an appropriate penalty for capacity control. While there are many options to consider, a natural choice is to penalize for the Frobenius norm of the full matrix $\mathbf{W}$. The objective then becomes:

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{W}\|_F + C \sum_i L(\mathbf{x}_i, y_i, \{\mathbf{U}, \mathbf{V}\}) \tag{4.8}$$

The derivatives of the loss and the regularizer with respect to $\mathbf{U}$ (query gradient) and $\mathbf{V}$ (database gradient) are worked out to be:

$$\frac{\partial S_{\mathbf{U},\mathbf{V}}(\mathbf{x},h)}{\partial \mathbf{U}} = -2\left(\sum_{\mathbf{x}_i \in h} \mathbf{U}(\mathbf{x}\mathbf{x}^T) - (\mathbf{V}\mathbf{x}_i)\mathbf{x}^T\right) \tag{4.9}$$

$$\frac{\partial S_{\mathbf{U},\mathbf{V}}(\mathbf{x},h)}{\partial \mathbf{V}} = -2\left(\sum_{\mathbf{x}_i \in h} \mathbf{V}(\mathbf{x}_i\mathbf{x}_i^T) - (\mathbf{U}\mathbf{x})\mathbf{x}_i^T\right) \tag{4.10}$$

$$\frac{\partial \|W\|_F}{\partial \mathbf{U}} = 2(\mathbf{U}\mathbf{U}^T)\mathbf{U} + 4(\mathbf{V}\mathbf{V}^T)\mathbf{U} \tag{4.11}$$

$$\frac{\partial \|W\|_F}{\partial \mathbf{V}} = 2(\mathbf{V}\mathbf{V}^T)\mathbf{V} + 4(\mathbf{U}\mathbf{U}^T)\mathbf{V} \tag{4.12}$$

Other possibly well motivated regularizers (that were also tried during experimentation) are $\|\mathbf{U}^T\mathbf{U}\|_F$ or $\|\mathbf{U}\|_F$ in the update equation for $\mathbf{U}$ and $\|\mathbf{V}^T\mathbf{V}\|_F$ or $\|\mathbf{V}\|_F$ in the update equation for $\mathbf{V}$.

4.1.2 *Optimization*

With all the machinery stated and out of the way, we are finally in a position to write how an iteration of the learning algorithm proceeds. Each iteration t of the algorithm consists of three steps:

a. Targeted inference of h_i^* for each sample $\mathbf{x}_i$:

$$h_i^* = \underset{h:\Delta(y_i,h)=0}{\operatorname{argmax}}\ S_{\mathbf{U}^{(t)},\mathbf{V}^{(t)}}(\mathbf{x}_i,h) \tag{4.13}$$

This can be done by algorithm 2 in time $O(N \log N)$, but with the slight modification of using equation 4.6 for the similarity computation instead.

b. Loss augmented inference of $\hat{h}_i$ for each sample $\mathbf{x}_i$:

$$\hat{h}_i = \operatorname{argmax} h \left[S_{\mathbf{U}^{(t)},\mathbf{V}^{(t)}}(\mathbf{x}_i,h) + \Delta(y_i,h)\right] \tag{4.14}$$

This can be done by algorithm 3, again by using 4.6 for the similarity instead.

c. Gradient updates for $\mathbf{U}$ and $\mathbf{V}$.

$$\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} - \eta^{(t)}\left[\frac{\partial S_{\mathbf{U},\mathbf{V}}(\mathbf{x},\hat{h}_i)}{\partial \mathbf{U}} - \frac{\partial S_{\mathbf{U},\mathbf{V}}(\mathbf{x},h_i^*)}{\partial \mathbf{U}} + 2(\mathbf{U}\mathbf{U}^T)\mathbf{U} + 4(\mathbf{V}\mathbf{V}^T)\mathbf{U}\right]$$

$$\mathbf{V}^{(t+1)} = \mathbf{V}^{(t)} - \eta^{(t)}\left[\frac{\partial S_{\mathbf{U},\mathbf{V}}(\mathbf{x},\hat{h}_i)}{\partial \mathbf{V}} - \frac{\partial S_{\mathbf{U},\mathbf{V}}(\mathbf{x},h_i^*)}{\partial \mathbf{V}} + 2(\mathbf{V}\mathbf{V}^T)\mathbf{V} + 4(\mathbf{U}\mathbf{U}^T)\mathbf{V}\right]$$

4.1.3 *Experiments and Conclusion*

The experimental setup is the same as in 3.5: We consider the same datasets, replicate the protocol in [71], consider the same test train splits, cross validation procedure and report k-NN errors for $k = 3, 7, 11$ for different methods on exactly the same folds. However, we do not repeat the experiments for different feature normalizations other than z-scoring, as they were consistently found to not help. This was also reflected in the results reported in 3.5. Therefore the numbers reported in 4.1 are identical to that in 3.1, except for the last column.

The only difference as compared to 3.5 is that the initialization using ReliefF [75] was done such that $\mathbf{U}$ and $\mathbf{V}$ were initialized to the same diagonal matrix; with the diagonal elements being the square-root of the weights obtained by using ReliefF. The experiments show no clear trend, although one thing is clear: there is marginal improvement over [147], and at least the results obtained by using the asymmetric distance as consistently better than the other methods in the competition. These results demonstrate that using this asymmetric similarity metric in conjunction with the *gerrymandering* formulation did slightly better than the case where the distance computation was symmetric.

4.2 HAMMING DISTANCE METRIC LEARNING

As discussed earlier, performance of nearest neighbor classification methods are often limited by two factors:

1 Computational cost of searching for nearest neighbors in a large database.

2 The choice of the underlying metric that defines "nearest".

In preceding discussions in this dissertation, we have exclusively focused on addressing (2): the choice of metric. In this chapter we turn our attention to (1), while still maintaining the flavor of solutions that were used to address (2).

The cost searching for nearest neighbors is usually addressed by efficient indexing and searching for approximate nearest neighbors instead (see [3, 9, 29, 64, 90] and references therein). The motivation for some such methods is simple: Usually for some task, approximate nearest neighbors rather than exact nearest neighbors should be good enough, assuming the data is well behaved. Relaxing the requirement for exact nearest neighbors can allow for sub-linear time search, which can be substantial for extremely large dataset sizes.

Yet another (but related to the above) approach is instead searching in Hamming space. That is, the data is projected onto a (potentially) lower dimensional Hamming space, where fast exact or approximate nearest neighbor search might be carried out. Needless to say, the embeddings generated must reflect some properties of the data. To this end, the original work on Locality Sensitive Hashing [29, 64], the codes were found using random projections, such that two points in the Hamming space were likely to be close if they were close in the original feature space. It has been observed that while random projections generate codes that are faithful to the pairwise distance, the code lengths can become prohibitively large.

k = 3

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
d	170	256	16	800	800	800	800
N	7797	9298	20000	157	958	295	1123
C	26	10	26	10	10	10	10
Euclidean	8.66	6.18	4.79 ±0.2	75.20 ±3.0	60.13 ±1.9	56.27 ±2.5	80.5 ±4.6
LMNN [157]	4.43	5.48	3.26 ±0.1	24.17 ±4.5	26.72 ±2.1	15.59 ±2.2	46.93 ±3.9
GB-LMNN [71]	**4.13**	5.48	2.92 ±0.1	21.65 ±4.8	26.72 ±2.1	13.56 ±1.9	46.11 ±3.9
MLR [107]	6.61	8.27	14.25 ±5.8	36.93 ±2.6	24.01 ±1.8	23.05 ±2.8	46.76 ±3.4
ITML [30]	7.89	5.78	4.97 ±0.2	19.07 ±4.9	33.83 ±3.3	13.22 ±4.6	48.78 ±4.5
1-NCA [50]	6.16	5.23	4.71 ±2.2	31.90 ±4.9	30.27 ±1.3	16.27 ±1.5	46.66 ±1.8
k-NCA	4.45	**5.18**	3.13 ±0.4	21.13 ±4.3	24.31 ±2.3	13.19 ±1.3	44.56 ±1.7
MLNG [147]	4.87	**5.18**	**2.32 ±0.1**	**17.18 ±4.7**	**21.34 ±2.5**	**10.85 ±3.1**	**43.37 ±2.4**
Asym-MLNG	4.65	**5.17**	**2.39 ±0.1**	19.01 ±3.6	23.45 ±1.9	**10.53 ±4.7**	**43.6 ±2.1**

k = 7

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
Euclidean	7.44	6.08	5.40 ±0.3	76.45 ±6.2	62.21 ±2.2	57.29 ±6.3	80.76 ±3.7
LMNN [157]	3.78	**4.9**	3.58 ±0.2	25.44 ±4.3	29.23 ±2.0	14.58 ±2.2	46.75 ±2.9
GB-LMNN [71]	**3.54**	**4.9**	2.66 ±0.1	25.44 ±4.3	29.12 ±2.1	12.45 ±4.6	46.17 ±2.8
MLR [107]	5.64	8.27	19.92 ±6.4	33.73 ±5.5	23.17 ±2.1	18.98 ±2.9	46.85 ±4.1
ITML [30]	7.57	5.68	5.37 ±0.5	22.32 ±2.5	31.42 ±1.9	**10.85 ±3.1**	51.74 ±2.8
1-NCA [50]	6.09	5.83	5.28 ±2.5	36.94 ±2.6	29.22 ±2.7	22.03 ±6.5	45.50 ±3.0
k-NCA	4.13	5.1	3.15 ±0.2	22.78 ±3.1	23.11 ±1.9	13.04 ±2.7	43.92 ±3.1
MLNG [147]	4.61	**4.9**	2.54 ±0.1	**21.61 ±5.9**	**22.44 ±1.3**	11.19 ±3.3	**41.61** ±2.6
Asym-MLNG	4.63	**4.9**	**2.34 ±0.1**	23.65 ±3.9	23.84 ±2.8	11.4 ±2.3	41.35 ±2.2

k = 11

Dataset	Isolet	USPS	letters	DSLR	Amazon	Webcam	Caltech
Euclidean	8.02	6.88	5.89 ±0.4	73.87 ±2.8	64.61 ±4.2	59.66 ±5.5	81.39 ±4.2
LMNN [157]	**3.72**	**4.78**	4.09 ±0.1	23.64 ±3.4	30.12 ±2.9	13.90 ±2.2	49.06 ±2.3
GB-LMNN [71]	3.98	**4.78**	2.86 ±0.2	23.64 ±3.4	30.07 ±3.0	13.90 ±1.0	49.15 ±2.8
MLR [107]	5.71	11.11	15.54 ±6.8	36.25 ±13.1	24.32 ±3.8	17.97 ±4.1	44.97 ±2.6
ITML [30]	7.77	6.63	6.52 ±0.8	**22.28 ±3.1**	30.48 ±1.4	11.86 ±5.6	50.76 ±1.9
1-NCA [50]	5.90	5.73	6.04 ±2.8	40.06 ±6.0	30.69 ±2.9	26.44 ±6.3	46.48 ±4.0
k-NCA	4.17	4.81	3.87 ±0.6	23.65 ±4.1	25.67 ±2.1	11.42 ±4.0	43.8 ±3.1
MLNG [147]	4.11	4.98	3.05 ±0.1	**22.28 ±4.9**	**24.11 ±3.2**	**11.19 ±4.4**	**40.76 ±1.8**
Asym-MLNG	4.0	**4.81**	**2.4 ±0.1**	23.78 ±4.3	**24.11 ±3.9**	**11.1 ±3.7**	43.7 ±2.7

Table 4.1: *k*NN errors for *k*=3, 7 and 11 (asymmetric metric learning versus other methods). Features were scaled by z-scoring.

Moreover, such approaches are completely label oblivious: the codes generated do not reflect any semantic structure, and often the purpose of nearest neighbor search is some downstream task-specific application (like classification). The approach to instead machine learn the binary codes such that they are *similarity sensitive* to the underlying semantic structure was taken in the pioneering works of Shakhnarovich [133], the *Semantic Hashing* or *supermarket search* of Salakhutdinov and Hinton [130], and Weiss *et al.* [159]. To motivate machine learning of binary codes, consider we have to search an image from a large database which is similar to a given query image. We can then ask a number of binary questions to make search easier: Is the query a color image or a grayscale image? Is there is a dog in the query or not? Does the query represent an indoor scene or an outdoor scene and so on. We can think of the codes as encoding a set of binary choices that are generated with the explicit goal to reflect label structure. If the underlying task is k-NN classification, we would want the code to be able to reflect the class of the image.

Following [130, 133, 159], in the last decade there has been an explosion in research in this area of learning compact, similarity-sensitive binary codes, which is nearly impossible to review. In any case, for the purpose of this chapter, it is not necessary either. For some prominent works we direct the reader to [51, 89, 118, 154] and the references therein, or more recent works that cite these.

To motivate our approach to the problem of learning binary codes that are similarity sensitive, we first hark back to work discussed in this dissertation thus far: We considered methods for learning the underlying metric or notion of similarity, this was achieved by projecting the data into a real space such that the underlying nearest neighbor classification accuracy in this space was improved. Indeed, the loss function employed was such that it was a more direct proxy to the k-NN classification error.

In this section, we suitably modify the framework proposed earlier for learning binary codes such that nearness in the Hamming space is a better proxy for classification accuracy. In other words, we learn a mapping from points in $\mathbb{R}^d$ to $\mathcal{H}$ in such a way that the accuracy of the k-NN classifier in the Hamming space is improved, by using the gerrymandering loss. Our main competitor in this regard is the method of Nourouzi *et al.* [119], also inspired by Latent Structural SVMs, where a triplet loss was used to learn a Hamming distance between two points that is reflective of similarity. As already noted, the triplets are assumed to be set statically as an input to the algorithm, and the optimization focuses on the distance ordering rather than accuracy of classification. We use a loss function akin to that proposed in chapter 3, but now adapted for the case of optimizing for the similarity directly in Hamming space. The gerrymandering loss being a more direct proxy for k-NN classification performance, can perhaps aid in the learning of compact binary codes that directly reflect nearest neighbor accuracy, and thus is perhaps better motivated than the approach of [119]. In the next section, we formulate the problem and our approach to attack it.

4.2.1 *Formulation*

We are interested in the problem of discriminative learning of Hamming distance between points. To this end, we first consider the Hamming distance between asymmetric linear binary hashes with code length c: Once we have used this to introduce the framework, we will consider other variants, such as using non-linear binary hashes as well as symmetric hashes.

$$D_{\mathbf{U},\mathbf{V}}(\mathbf{x},\mathbf{x}_i) = \sum_{j=1}^{c} \mathbf{1}_{\mathrm{sgn}(\mathbf{U}_j\mathbf{x}) \neq \mathrm{sgn}(\mathbf{V}_j\mathbf{x}_i)} \tag{4.15}$$

where $\mathbf{U},\mathbf{V} \in \mathbb{R}^{c*d}$ are the parameters of the model. For any $h \in \mathbf{X}$, we define the distance score of h w.r.t. a point $\mathbf{x}$ as:

$$S_{\mathbf{U},\mathbf{V}}(\mathbf{x},h) = hc - \sum_{\mathbf{x}_i \in h} D_{\mathbf{U},\mathbf{V}}(\mathbf{x},\mathbf{x}_i) = \left\langle \mathrm{sgn}(\mathbf{U}\mathbf{x}), \sum_{\mathbf{x}_i \in h} \mathrm{sgn}(\mathbf{V}\mathbf{x}_i) \right\rangle \tag{4.16}$$

Therefore, the set of k nearest neighbors of $\mathbf{x}$ in $\mathbf{X}$ is:

$$h_{\mathbf{U},\mathbf{V}}(\mathbf{x}) = \underset{|h|=k}{\mathrm{argmax}}\, S_{\mathbf{U},\mathbf{V}}(\mathbf{x},h) \tag{4.17}$$

We will assume that k is known. Given any set h, we can predict the labels of $\mathbf{x}$ by the majority vote among members of h. We use a surrogate loss similar to

$$L(\mathbf{x},y,\{\mathbf{U},\mathbf{V}\}) = \max_h \left[S_{\mathbf{U},\mathbf{V}}(x,h) + \Delta(y,h)\right] - \max_{h:\Delta(y,h)=0} S_{\mathbf{U},\mathbf{V}}(\mathbf{x},h) \tag{4.18}$$

4.2.2 *Optimization*

Each iteration t of the algorithm consists of four steps:

a. Targeted inference of h_i^* for each sample $\mathbf{x}_i$:

$$h_i^* = \underset{h:\Delta(y_i,h)=0}{\mathrm{argmax}}\; S_{\mathbf{U}^{(t)},\mathbf{V}^{(t)}}(\mathbf{x}_i,h) \tag{4.19}$$

b. Loss augmented inference of $\hat{h}_i$ for each sample $\mathbf{x}_i$:

$$\hat{h}_i = \underset{h}{\mathrm{argmax}} \left[S_{\mathbf{U}^{(t)},\mathbf{V}^{(t)}}(\mathbf{x}_i,h) + \Delta(y_i,h)\right] \tag{4.20}$$

c. Gradient updates for $\mathbf{U}$ and $\mathbf{V}$. Let b be the mini batch at iteration t. Since the sign function is not differentiable, we can approximate it by another function f. In this case, the updates will be

$$\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} - \eta \sum_{\mathbf{x}_i \in b} \left[f'(\mathbf{U}\mathbf{x}_i) \circ \left(\sum_{\mathbf{x}_j \in \hat{h}_i} \mathrm{sgn}(\mathbf{V}\mathbf{x}_j) - \sum_{\mathbf{x}_j \in h_i^*} \mathrm{sgn}(\mathbf{V}\mathbf{x}_j) \right) \right] \mathbf{x}_i^\top$$

$$\mathbf{V}^{(t+1)} = \mathbf{V}^{(t)} - \eta \sum_{\mathbf{x}_i \in b} \left[\sum_{\mathbf{x}_j \in \hat{h}_i} \left(\mathrm{sgn}(\mathbf{U}\mathbf{x}_i) \circ f'(\mathbf{V}\mathbf{x}_j) \right) \mathbf{x}_j^\top - \sum_{\mathbf{x}_j \in h_i^*} \left(\mathrm{sgn}(\mathbf{U}\mathbf{x}_i) \circ f'(\mathbf{V}\mathbf{x}_j) \right) \mathbf{x}_j^\top \right]$$

where possible options for the function f are:

a) $f(x) = x$ and therefore $f'(x) = 1$.

b) $f(x) = \tanh(x)$ and therefore $f'(x) = 1 - \tanh^2(x)$

d. Normalization:

$$\mathbf{U}^{(t+1)} = \frac{\mathbf{U}^{(t+1)}}{\left\|\mathbf{U}^{(t+1)}\right\|_F} \tag{4.21}$$

$$\mathbf{V}^{(t+1)} = \frac{\mathbf{V}^{(t+1)}}{\left\|\mathbf{V}^{(t+1)}\right\|_F} \tag{4.22}$$

4.2.3 *Symmetric Variant*

We have (with $\mathbf{W} \in \mathbb{R}^{c*d}$):

$$S_{\mathbf{W}}(\mathbf{x}, h) = hc - \sum_{\mathbf{x}_i \in h} D_{\mathbf{W}}(\mathbf{x}, \mathbf{x}_i) = \left\langle \operatorname{sgn}(\mathbf{W}\mathbf{x}), \sum_{\mathbf{x}_i \in h} \operatorname{sgn}(\mathbf{W}\mathbf{x}_i) \right\rangle \tag{4.23}$$

Gradient:

$$\frac{\partial S_{\mathbf{W}}(\mathbf{x}, h)}{\partial \mathbf{W}} = \left[\left(\sum_{\mathbf{x}_i \in h} \operatorname{sgn}(\mathbf{W}\mathbf{x}_i)\right) \circ f'(\mathbf{W}\mathbf{x})\right] \mathbf{x}^\top + \left[\sum_{\mathbf{x}_i \in h} \left(\operatorname{sgn}(\mathbf{W}\mathbf{x}) \circ f'(\mathbf{W}\mathbf{x}_i)\right) \mathbf{x}_i^\top\right] \tag{4.24}$$

We also add the following penalty to the objective function as suggested in [159]. This encourages each bit, averaged over the training data, to be zero mean before quantization

$$\frac{1}{2}\|mean_{\mathbf{x}} \operatorname{sgn}(\mathbf{W}\mathbf{x})\|_2^2 \tag{4.25}$$

Which adds the following term to the update of $\mathbf{W}$

$$mean_{\mathbf{x}} \left(\left[\operatorname{sgn}(\mathbf{W}\mathbf{x}) \circ f'(\mathbf{W}\mathbf{x})\right] \mathbf{x}^\top \right) \tag{4.26}$$

Note that we also add a term akin to 4.25 to the asymmetric variant of our algorithm in the experiments with two additional terms suitably added in the gradient updates of $\mathbf{U}$ and $\mathbf{V}$.

Before we begin to describe our experiments, we first mention how we carried out nearest neighbor search in Hamming space.

4.2.4 *Distance Computation in Hamming Space*

When we consider a query point $\mathbf{x} \in \mathcal{H}$ and look for nearest neighbors in the database, there is a high probability of a tie, especially when code lengths are shorter. We experimented with several options for tie breaking, but eventually settled on using what is referred to in the literature as *asymmetric hamming distance*. Note that this nomenclature is rather unfortunate given our formulation of Hamming metric learning that uses different hash functions for the query and the database points. This notion of the asymmetric hamming distance used in the literature (see [38, 53, 119]) for retrieving points in Hamming space is quite different. We use the approach used by [119] as we found it to consistently give superior results. For a given query $\mathbf{x} \in \mathbb{R}^d$, we do not binarize it while searching for neighbors in the database. The database points however live in $\mathcal{H}$. The distance between the query and the database point is given by:

$$AsymH(\mathbf{x}, \mathbf{x}_i; \mathbf{s}) = \frac{1}{4}\|\mathbf{x}_i - \tanh(Diag(\mathbf{s})\mathbf{x})\|_2^2$$

As also observed by [119], the distance computation is relatively insensitive to the choice of scaling parameters $\mathbf{s} \in \mathbb{R}^d$. However, after some experimentation, we set the scale parameters to be such that the real valued projection of the query vector has an average absolute value of 0.4.

4.2.5 *Experiments and Conclusion*

To test the efficacy of our method, we compare it directly with the experiments of [119] on MNIST. For training, we initialize **U** and **V** (for the asymmetric case) and **W** (for the symmetric case) as random gaussian matrices with mean zero and standard deviation of 1. We stop training when the running average of the surrogate loss over the last epoch does not decrease substantially, or when the maximum number of epochs is reached. We use a decaying learning rate schedule as $\eta(t) = \frac{1}{t}$, and set the momentum parameter to 0.9. For training, to compare with the approach [119], we fix k to be 3 and 30 while training our system. Finally, we set aside 10,000 points from the MNIST training set for the purpose of tuning the regularization constant.

We compare results that report the kNN error, and compare directly with the results of [119], as well as results reported using baseline methods that do not use binary codes to find nearest neighbors.

We observe the following clear trends: The linear hash function used in the gerrymandering framework, both in the symmetric and asymmetric case performs worse than the linear setting in [119]. We also observe that the performance improves in our case when the code length increases, almost equaling the performance of the linear setting in [119] at higher code lengths. It is also noticeable that the asymmetric variant consistently gives better performance than the symmetric variant, suggesting that the asymmetry indeed helps in learning better codes. Next, in the two-layer setting: The asymmetric variant of the Gerrymandering loss consistently performs better than all the settings explored in [119], while the symmetric variant is comparable to [119], but usually slightly worse. Another interesting observation is that the gerrymandering loss in the two layer setting gives performance roughly similar as the code length increases, as contrasted to [119] where it steadily improves. This shows that the loss is able to learn more compact and discriminative binary codes.

In conclusion, in this section, we have presented a novel method for the discriminative learning of Hamming distance, and on experiments demonstrated that it works better than the competition in this space. One drawback of our method as compared to [119] is that the inference procedures are quite expensive to solve exactly, making it quite slow. We leave the development of approximate inference procedures for future work, which will enable to scale our method to much larger dataset sizes.

Hash Function/Loss	k	32 bits	64 bits	128 bits
Linear/pairwise hinge	3	4.3	2.78	2.46
Linear/triplet	3	3.88	2.90	2.51
Two-Layer (tanh)/pairwise	30	1.50	1.36	1.35
Two-Layer (tanh)/triplet	30	1.45	1.29	1.20
Linear/MLNG (SH)	3	5.7	3.6	2.7
tanh/MLNG (SH)	30	2.3	2.17	1.91
Two-Layer (tanh)/MLNG (SH)	30	1.39	1.31	1.28
Linear/MLNG (ASH)	3	4.9	3.2	2.49
tanh/MLNG (ASH)	30	2.15	2.11	1.76
Two-Layer (tanh)/MLNG (ASH)	30	1.2	1.18	1.25

Table 4.2: k NN classification errors on MNIST using Hamming distance metric learning by gerrmandering, compared to the approach of [119]. All results reported use the distance computation explicated in Section 4.2.4. MLNG refers to the Gerrymandering loss, SH refers to the use of symmetric hashes, while ASH to asymmetric hashes.

4.3 METRIC LEARNING FOR K-NN REGRESSION

To motivate this part of this chapter, we first review the basic setting in kernel regression, and use it to motivate the work of Weinberger and Tesauro [158]. We then use this to place our contribution in contrast.

Recall the standard regression setting: We have an unknown, smooth $f : \mathbb{R}^d \to \mathbb{R}$, that we have to estimate based on labeled data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. The labels are a noisy version of the actual function outputs i.e. $y_i = f(\mathbf{x}_i) + \epsilon$, where ϵ is unknown. The task then is to use this sample to get an estimate $\hat{f}$ of f that minimizes some loss function. In the non-parametric setting, a standard regression technique is kernel regression that predicts an output $\hat{y}_i$ for an input $\mathbf{x}_i$ based on a weighted average of some of the inputs that are selected as its neighbors, the weighing is done as a function of the distance. This is given as below:

$$\hat{y}_i = \sum_{j \neq i}^{N} w(\mathbf{x}_i, \mathbf{x}_j) y_j \tag{4.27}$$

To define the choice of weights $w(\mathbf{x}_i, \mathbf{x}_j)$, we first define a kernel function $K : \mathbb{R} \to \mathbb{R}$ which satisfies the following conditions:

$$\int K(x)dx = 1 \quad \text{and} \quad K(x) = K(-x)$$

While there is a vast literature on the choice of kernels, we content ourselves by defining only the Gaussian kernel, which suffices for the purpose of our discussion:

$$K(x) = \exp\left(\frac{-x^2}{2\sigma^2}\right)$$

Then, for any pair of points $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathbb{R}^d$, we can define the kernel as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

We can then define the weights $w(\mathbf{x}_i, \mathbf{x}_j)$ in equation 4.27 simply as:

$$w(\mathbf{x}_i, \mathbf{x}_j) = \frac{K(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j \neq i}^{N} K(\mathbf{x}_i, \mathbf{x}_j)}$$

More generally, we can write the kernel for some distance function $d(\mathbf{x}_i, \mathbf{x}_j)$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-d(\mathbf{x}_i - \mathbf{x}_j)^2}{2\sigma^2}\right)$$

For our familiar parameterization of d as the Mahalanobis distance i.e. $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)}$ with $\mathbf{W} \succeq 0$ and $\mathbf{W} = \mathbf{L}^T\mathbf{L}$, the kernel might be written as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j\|}{2\sigma^2}\right) \tag{4.28}$$

Working with 4.28 and 4.27, optimizing for $\mathbf{L}$ to minimize the loss $L = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ is the main contribution of *metric learning for kernel regression* [158]. This work remains to be the only major work in the literature that optimizes for the metric under a regression objective. The authors in [158] consider a gaussian kernel, which facilitates computing gradients with ease, and thus making optimization straightforward. In many scenarios, we are interested in using k nearest neighbors, however, the problem of metric learning in such a setting is not straightforward considering the combinatorial nature of the problem, as well as taking into account the fact that defining sets of pairs of similar and dissimilar points is no longer straightforward as was in the classification case (where we could deem a pair of points to be similar if they belonged to the same class). The only work that we are aware of that optimizes for a metric under a nearest neighbor based regression objective is that of [72]. However, it works with a NCA [50] type objective, minimizing the expected squared loss, and thus only working with a 1-NN regressor. In the next section, we outline our method that attempts to bridge this gap and learns a metric when the downstream task is k-NN regression.

4.3.1 *Problem formulation and Optimization*

In this section, we are interested in the problem of learning a Mahalanobis metric for the case of k-NN regression rather than kernel regression. Note that in k-NN regression 4.27 only has a particular choice of weights, we consider the following choice: if $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$, then $w(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{k}$ and 0 otherwise. Here, we show that with a suitable modification of the approach proposed in 3, we can obtain an efficient metric learning algorithm for the case of k-NN regression as well.

In order to describe the approach, we briefly state the setting again (which is similar to 3). That is, suppose we are given N training examples $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and their outputs $\mathbf{Y} = [y_1, \ldots, y_n]^T$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. For a subset $h \subset X$ with $|h| = k$, we can define its measure of similarity with a query point $\mathbf{x}$ as:

$$S_{\mathbf{W}}(\mathbf{x},h) = -\sum_{\mathbf{x}_j \in h} D_{\mathbf{W}}\left(\mathbf{x},\mathbf{x}_j\right) \tag{4.29}$$

where

$$D_{\mathbf{W}}\left(\mathbf{x},\mathbf{x}_i\right) = \left(\mathbf{x}-\mathbf{x}_i\right)^T \mathbf{W}\left(\mathbf{x}-\mathbf{x}_i\right), \tag{4.30}$$

Thus, we work with the same measure of similarity as in 3, but a departure from the formulation occurs here as we move forward to state a reasonable objective. This is because, in the case of regression, the outputs y_i are no longer discrete, but real valued. Therefore, unlike in 3, the loss $\Delta(y,h)$ may not necessarily be zero for any set $h \subset X$. Note that, given h, the squared loss for a query point $(\mathbf{x},y)$ is:

$$\Delta(y,h) = \left(y - \frac{1}{k}\sum_{\mathbf{x}_i \in h} y_i\right)^2 \tag{4.31}$$

Armed with the notion of similarity as well as the loss, we can make a first attempt to define a loss for the task of metric learning for k-NN regression as follows. For some $\gamma > 0$:

$$L(\mathbf{x},y,\mathbf{W}) = \max_h \left[S_{\mathbf{W}}(\mathbf{x},h) + \gamma\Delta(y,h)\right] - \max_h \left[S_{\mathbf{W}}(\mathbf{x},h) - \gamma\Delta(y,h)\right] \tag{4.32}$$

Adding the regularizer, this supplies us with the following objective

$$\min_{\mathbf{W}} \|\mathbf{W}\|_F + C\sum_i L(\mathbf{x}_i,y_i,\mathbf{W}) \tag{4.33}$$

At first blush, the optimization for this objective is similar to the variants of the *gerrymandering* approach discussed earlier. That is,

Each iteration t of the algorithm consists of the following steps:

a. Targeted inference of h_i^* for each sample $\mathbf{x}_i$:

$$h_i^* = \operatorname*{argmax}_h \left[S_{\mathbf{W}}(\mathbf{x}_i,h) - \gamma\Delta(y_i,h)\right] \tag{4.34}$$

b. Loss augmented inference of $\hat{h}_i$ for each sample $\mathbf{x}_i$:

$$\hat{h}_i = \operatorname*{argmax}_h \left[S_{\mathbf{W}}(\mathbf{x}_i,h) + \gamma\Delta(y_i,h)\right] \tag{4.35}$$

c. Gradient update for $\mathbf{W}$.

However, for the loss defined in 4.31, this optimization is hard to solve exactly. Indeed, we can make the following claim:

Claim 3. *Given data $\mathbf{X}$ and corresponding labels $\mathbf{Y}$, matrix $\mathbf{W}$ and query point $\mathbf{x}_i$, $h \subset X$ with $|h| > 1$ and $\gamma > 0$ the problem of finding h_i^* and $\hat{h}_i$ is NP Hard.*

While the proof of this claim is omitted, this can be shown by an appropriate reduction to a modified version of the subset sum problem. We could resort to relaxations of the above problem, and thus work with suitable approximation algorithms to optimize for 4.34 and 4.35. Instead, we work with the following simple modification to our loss:

$$\hat{\Delta}(y, h) = \frac{1}{k} \sum_{\mathbf{x}_i \in h} (y - y_i)^2 \tag{4.36}$$

Then, we have the following modified inference problems:

$$h_i^* = \operatorname*{argmax}_h \left[S_{\mathbf{W}}(\mathbf{x}_i, h) - \gamma \hat{\Delta}(y_i, h) \right] \tag{4.37}$$

$$\hat{h}_i = \operatorname*{argmax}_h \left[S_{\mathbf{W}}(\mathbf{x}_i, h) + \gamma \hat{\Delta}(y_i, h) \right] \tag{4.38}$$

Note that both of these problems are easy to solve: Fixing $\mathbf{W}$ and $\gamma > 0$, for each query point $(\mathbf{x}, y)$, we simply need to sort the data in ascending order by the sum of their loss and similarity to the query and then pick the top and bottom k points to be h_i^* and $\hat{h}_i$ respectively.

Before proceeding to report experiments on this approach and compare it with [158], it is instructive to see what the relation between the losses 4.31 and 4.36 is. This is made explicit in the following claim.

Claim 4. $\hat{\Delta}(y, h)$ *upper bounds the loss* $\Delta(y, h)$

Proof. We have: $\hat{\Delta}(y, h) = \frac{1}{k} \sum_{i \in h} (y - y_i)^2$

Expanding: $\hat{\Delta}(y, h) = \frac{1}{k} \sum_{i \in h} (y^2 + y_i^2 - 2yy_i) = \left(y^2 + \frac{1}{k^2} \sum_{i \in h} y_i^2 \sum_{i \in h} 1^2 - \frac{2y}{k} \sum_{i \in h} y_i \right)$

Now, using Cauchy Schwarz on the second term and rearranging, we have:

$\hat{\Delta}(y, h) \geq \left(y^2 + \left(\frac{1}{k} \sum_{i \in h} y_i \cdot 1 \right)^2 - \frac{2y}{k} \sum_{i \in h} y_i \right) = \left(y - \frac{1}{k} \sum_{i \in h} y_i \right)^2$

or, $\hat{\Delta}(y, h) \geq \Delta(y, h)$

$\square$

4.3.2 *Alternate notions of h^**

The set h^* may also be defined in other ways. We consider two such definitions here. First uses the notion of an ϵ-insensitive loss (similar to the notion used in Support Vector Regression [138]): We can define a set $\mathcal{H}$ where for a query $(\mathbf{x}_i, y_i)$

$$\mathcal{H}_i = \{h | \Delta(y_i, h) \leq \epsilon\}$$

Then we may define h^* as:

$$h_{\epsilon i}^* = h_i^* = \arg\max_{h \in \mathcal{H}} \left[S_{\mathbf{W}}(\mathbf{x}_i, h) \right] \tag{4.39}$$

ϵ then becomes a hyper-parameter to be found by cross-validation.

Yet another notion of h^* may be defined as follows:

$$h^*_{Mi} = h^*_i = \arg\min_h \Delta(y_i, h) \tag{4.40}$$

Note that using this definition makes the overall objective convex. The inference and loss augmented procedures are straightforward for all these definitions.

4.3.3 *Experiments*

For the purpose of experimental evaluation of our approach, we make a direct comparison to the experiments reported in [158] by working with the DELVE (Data for Evaluating Learning in Valid Experiments) datasets [1]. In particular we work with 16 datasets, 8 from the Kin family and 8 from Pumadyn family of datasets. The Kin datasets were obtained from realistic simulations of a 8-link all robot arm, and consist of four datasets having 8 dimensions and four 32 dimensional datasets. Likewise, the Pumadyn datasets were obtained from the simulations of dynamics of a Puma 560 robot arm, and also consist of four 8 dimensional and four 32 dimensional datasets. The various features represent the torque, angular momentum, angular positions of the robotic arm, and the output is a real number.

Each of these datasets have 8092 points. Following the protocol in [158], each dataset is split into four disjoint training sets, each of size 1024 and one unique test set, also of size 1024. For each training set we do a 5 fold cross validation for parameter tuning. In the experiments reported here we use the following naming protocol: <NAME>$-$axy. Where <NAME> is the name of the dataset. 'a' denotes an integer that signifies the dimensionality of the dataset. 'x' takes on the character values of either'f' or 'n', where 'f' denotes that the dataset is largely linear, while 'n' denotes non-linearity. 'y' takes on character values of 'm' or 'h', where 'm' denotes medium noise, while 'h' denotes high noise. Thus, for an example, the nomenclature Puma-32nh would imply we are referring to a dataset from the Pumadyn family with 32 features, in which the regression target is a non-linear function of the input and that the dataset has high noise.

For testing, we consider the following baselines: linear regression, a nearest neighbor model in which the k is chosen over a range of values by cross-validation, a nearest neighbor model in which the coordinates are weighted by ReliefF [75] and the appropriate value of k is chosen by cross validation, Gaussian process regression [160], the MLKR approach of [158]. All the numbers reported are relative to a baseline method that uses the mean of the training regression targets in case the squared error is minimized (for more details on the protocol and the DELVE suite, we point the reader to the manual [125]. For the approach in [158], we use code provided by the authors). In all cases we minimize for the squared error.

For our approach, we work with two models: One in which the metric is learned by optimizing for $\mathbf{L}$ (such that $\mathbf{L}^T\mathbf{L} = \mathbf{W}$) using the formulation implicit through equations 4.37 and 4.38, while initializing $\mathbf{L} = \mathbf{I}$. The second model is similar except that we use the formulation described in 4.1, and initialize the matrices $\mathbf{U}$ and $\mathbf{V}$ as diagonal matrices, where the diagonal carries the square root of the ReliefF coefficients after they

1 Available online at https://www.cs.toronto.edu/ delve/data/datasets.html

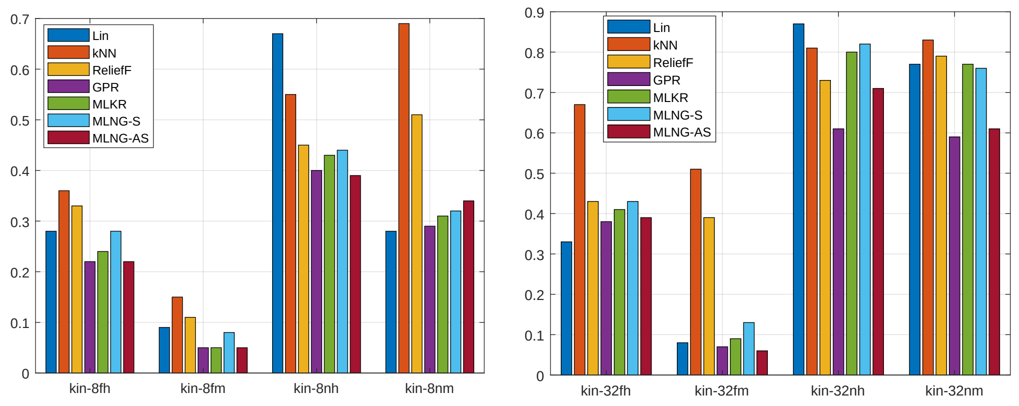


Figure 4.1: Regression errors on the Kin datasets. (Here Lin stands for linear regression, ReliefF for *k*NN regression after feature weighing by ReliefF, GPR for Gaussian Process Regression, MLKR for Metric Learning for Kernel Regression, MLNG-S for metric learning by neighborhood gerrymandering for regression using symmetric similarity computation, while MLNG-AS refers to the same method but with an asymmetric notion of similarity)

are normalized to be between 0 and 1. For both these models we cross validate for the batch size (from values 1, 3, 5, 11), the parameter γ (for values increasing logarithmically from 0.00001 to 100), and the number of neighbors k (for values from 1, 3, 5, 7, 11, 21 and 31).

Figures 4.1 and 4.2 illustrate the results. On the 'linear' datasets (marked with 'f'), a recurring trend is that plain linear regression does quite well as compared to nearest neighbors. Thus it is interesting to see that all three metric learning methods evaluated here perform better than it. In the non-linear datasets on the other hand, all other methods other than the metric learning methods and Gaussian process regression perform poorly. The main trend that we see in this set of experiments is that Metric Learning for Kernel Regression [158] usually performs better than the symmetric variant of our algorithm, while being consistently worse than the asymmetric variant with ReliefF initialization. In summary, Gaussian Process Regression and the asymmetric variant of our algorithm consistently perform better, with our algorithm often performing slightly better.

4.3.4 *Conclusion of the Regression Experiments*

In summary, in this section we proposed an approach for metric learning for k-NN Regression. As noted earlier, literature on the topic is quite sparse, despite the fact that in many settings having a good metric while the downstream task is *k*NN regression might be desirable. In our experiments, we observed that the symmetric approach performs better than most baselines while underperforming the approach of [158] and Gaussian process regression, while the asymmetric approach performs at par with Gaussian Process Regression, often out-performing it, while consistently outperforming the approach of [158]. Our algorithm has very simple and efficient inference procedures; with the optimization for each dataset completing in a matter of a few minutes. Therefore

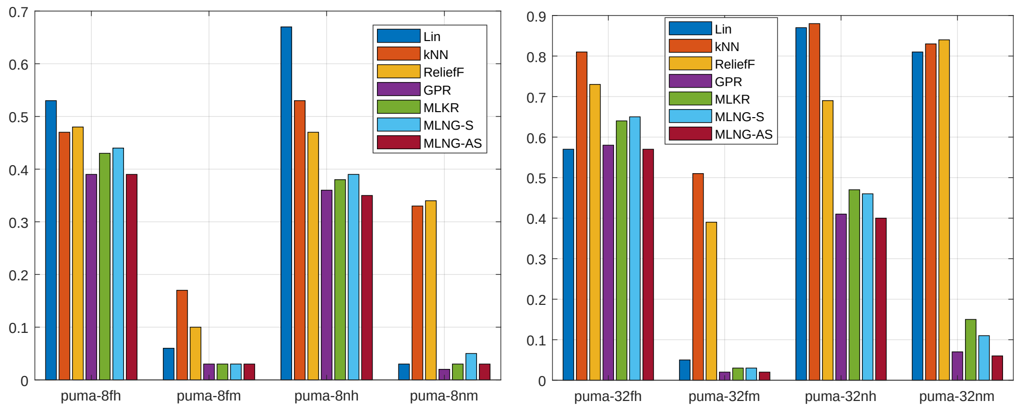


Figure 4.2: Regression errors on the Puma datasets (Here Lin stands for linear regression, ReliefF for *k*NN regression after feature weighing by ReliefF, GPR for Gaussian Process Regression, MLKR for Metric Learning for Kernel Regression, MLNG-S for metric learning by neighborhood gerrymandering for regression using symmetric similarity computation, while MLNG-AS refers to the same method but with an asymmetric notion of similarity)

it is perhaps worthwhile to consider the problem of metric learning in the continuous label setting in more detail, while also considering more applications where it could be useful.

4.4 CONCLUSION OF PART I AND SUMMARY

The conclusion of the regression experiments also brings us to the conclusion of this part of the dissertation. Below we summarize the main contributions made:

SUMMARY OF PART I

1 In Chapter 3, we presented an approach to metric learning that is more direct in trying to optimize for the *k*-NN accuracy than methods previously proposed. The approach formulates the problem of metric learning for *k*-NN classification as a large margin structured prediction problem, with the choice of neighbors represented by discrete latent variables, making it natural to use machinery from latent structural support vector machines to the task of metric learning. We also provided exact procedures for inference and loss-augmented inference in this model, and validated the approach by comparing to a range of Mahalanobis distance metric learning methods.

2 In this chapter we explored the formalism explicated on in 3 in different settings. In section 4.1, we explored an approach to similarity learning in which the similarity computation is asymmetric: the query and database points are subjected to different transformations. In section 4.2.4 we combined the approach of Chapter 3 and Section 4.1 to learn the Hamming distance such that it is a better proxy for

k-NN classification performance. Lastly, in section 4.3, we presented an approach to metric learning for *k*-NN regression that is consistently shown to perform better than its main competitors.

Often, while doing *k*-NN classification and regression, we might be on a limited computational budget– we might not be able to optimize for a metric over a space of possible metrics. However, we would still like to have access to a metric that is not completely label agnostic, but can be estimated cheaply, as performance using simply the Euclidean metric might be quite poor. Estimation of such a metric for improving *k*-NN classification and regression performance is the focus of the next part of this dissertation.

Part II

SINGLE PASS METRIC ESTIMATION

5 METRIC ESTIMATION VIA GRADIENTS

In the preceding part of this dissertation we proposed and worked with a more direct approach to metric learning; direct in that it tries to optimize for the metric using a differentiable loss that aims to be a more reasonable proxy for the underlying task: k-NN classification or regression. In this part, we take another, if somewhat indirect approach to the problem of identifying a good metric based only on gradient estimates of the unknown classification or regression function f. While somewhat roundabout from the perspective of the underlying task, this approach has the advantage that the metric can be estimated by a single pass over the dataset, while affording significant improvements in non-parametric regression and classification tasks. The rest of this section is used to motivate the problem, as well as to stage ground for the following two chapters which propose two variants of a gradient based metric estimator, which despite their simplicity remain statistically consistent under fairly mild assumptions.

To begin, recall that in high dimensional classification and regression problems, the task is to infer the unknown function f. We are given a set of observations $(\mathbf{x}, \mathbf{y})_i, i = 1, 2, \ldots n.$, with $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ and the labels $\mathbf{y}_i$ are noisy versions of the function values $f(\mathbf{x}_i)$. We are interested in distance based (non-parametric) regression, which provides our function estimate:

$$f_n(\mathbf{x}) = \sum_{i=1}^{n} w(\mathbf{x}, \mathbf{x}_i)\mathbf{y}_i$$

$w(\mathbf{x}, \mathbf{x}_i)$ depends on the distance ρ. Where $\rho(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')\mathbf{W}(\mathbf{x} - \mathbf{x}')}$ and $\mathbf{W} \succeq 0$. Note that, when $\mathbf{W} = \mathbf{I}$, ρ is the Euclidean distance.

The problem of estimating unknown f becomes significantly harder as d increases due to the *curse of dimensionality*. To remedy this situation, several pre-processing techniques are used, each of which rely on a suitable assumption about the data and/or about f. For instance, a conceptually simple, yet often reasonable assumption that can be made is that f might not vary equally along all coordinates of $\mathbf{x}$. Letting $f_i' = \nabla f^T e_i$ denote the derivative along coordinate i, and $\|f_i'\|_{1,\mu} \equiv \mathbb{E}_{\mathbf{x}\sim\mu} f_i'(\mathbf{x})$, we can use the above distance based estimator by setting ρ such that

$$\mathbf{W}_{i,j} = \begin{cases} \|f_i'\|_{1,\mu} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

This *gradient weighting* rescales the space such the ball $\mathcal{B}_\rho$ contains more points relative to the Euclidean ball $\mathcal{B}$ (figure 5.1 for an example in $\mathbb{R}^2$). This is the intuition pursued in works such as [85], [86] (which are also the inspiration for what follows in this part of the dissertations), with an emphasis on deriving an efficient, yet consistent estimator for the gradient. Using gradient weights for coordinate scaling in this manner has strong theoretical grounding as is shown in these works, in that it has the effect of reducing the regression variance, while keeping the bias in control.

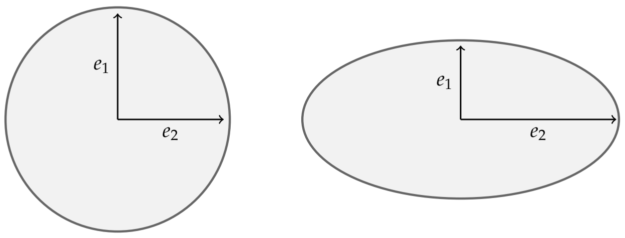


Figure 5.1: Left: Euclidean ball $\mathcal{B}$ which assigns equal importance to both directions e_1 and e_2. Right: ball $\mathcal{B}_\rho$ such that $\|f_1'\|_{1,\mu} \gg \|f_2'\|_{1,\mu}$, giving its ellipsoidal shape. Relative to the $\mathcal{B}$; $\mathcal{B}_\rho$ will have more mass in direction e_2

While appealing in its simplicity, gradient weighing has an obvious drawback: the metric only involves diagonal $\mathbf{W}$. In general $\mathbf{W} \succeq 0$, need not be diagonal and may be decomposed as $\mathbf{W} = \mathbf{V}\Sigma\mathbf{U}^T$ where $\mathbf{U}, \mathbf{V}$ are orthogonal matrices. In such a case the data would not only be rescaled but also rotated. This is illustrated in figure 5.2.

Using the above motivation to construct a covariance type matrix but only using gradients via an iterative algorithm (which involved taking the outer product of the gradients $\nabla f(X) \cdot \nabla f(X)^\top$ and summing over all the points) was used by us to derive an operator, with the following property: If the function f does not vary along some direction $\mathbf{v} \in \mathbb{R}^d$, then $\mathbf{v}$, must lie in the nullspace of the operator. But we later discovered that this operator was already known in the literature in a different context. To define things clearly, put it in proper context and also outline our contributions, we first take a step back and consider the motivation that we mentioned earlier. The unknown classification or regression function f might not vary equally in all coordinates. We used this fact to review the approach of [85], [86] above.

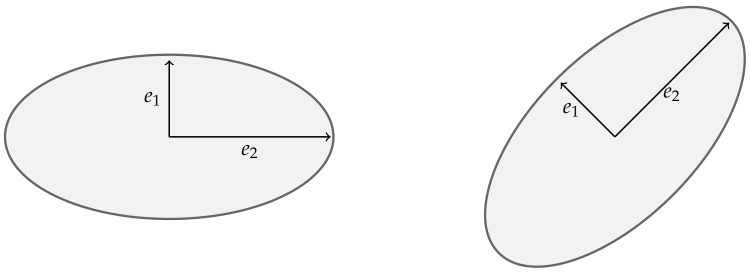


Figure 5.2: Left: Rescaled ball $\mathcal{B}_\rho$ which is still axis aligned but rescales the coordinates. Right: Ball that not only rescales the data but also rotates it

This simple observation is also motivation for a plethora of variable selection methods. In variable selection, the assumption that is used is that for f, we have $f(\mathbf{x}) = g(P\mathbf{x})$, where $P \in \{0,1\}^{k \times d}$ projects X down to $k < d$ coordinates that are most relevant to predicting the output y. This assumption is generalized further in *multi-index regression* e.g.[55, 96, 123, 166]). This is done by letting $P \in \mathbb{R}^{k \times d}$ project $\mathbf{x}$ down to a k-dimensional subspace of $\mathbb{R}^d$. Put differently, this is a generalization because here it is assumed that while f might vary along all coordinates, it actually only depends on an unknown k-dimensional subspace. Such a subspace is called a *relevant* subspace. The task then becomes finding the said relevant subspace rather than chopping coordinates since they all might be relevant in predicting the output y.

Work to recover this relevant subspace (which is sometimes also referred to in the literature as *effective dimension reduction* [96]) gives rise to the expected gradient outerproduct (EGOP):

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x})^{\top} \right).$$

This operator (which superficially seems similar to the Fisher information matrix) is useful beyond the multi-index motivation mentioned above. That is, even when there is no clearly relevant dimension-reduction P, as is usually likely in practice, one might still expect that f does not vary equally in all directions. Therefore, beyond the use of EGOP for dimension-reduction, we might use it instead to weight any direction $v \in \mathbb{R}^d$ according to its relevance as captured by the average variation of f along v (encoded in the EGOP). The weighting approach will be the main use of EGOP considered in this work. That is, we use the EGOP in the following way: let $VDV^{\top}$ be a spectral decomposition of the estimated EGOP, we use it to transform the input $\mathbf{x}$ as $D^{1/2}V^{\top}\mathbf{x}$. Also, for constructing the EGOP we need to compute gradient estimates. Just as in the case of gradient estimation, optimal estimators of the EGOP can be expensive in practice. In this part of the thesis we also show that a simple, efficient difference based estimator suffices in that it remains statistically consistent under mild assumptions.

It is important to note that estimating the EGOP and using it to transform the inputs as $\mathbf{x} \mapsto D^{1/2}V^{\top}\mathbf{x}$ and then using it for k-classification and regression does not involve any *learning*. Thus this approach is related to but distinct from metric learning in that a metric is not optimized for over a space of possible metrics parametrized by positive semi-definite matrices. This approach is also online and cheap: we only require $2d$ estimates of the function f at $\mathbf{x}$, and can also be used for preprocessing for standard metric learning methods. Work on the EGOP is explicated upon in Chapter 6.

As will be described later, the EGOP can be used for metric weighing in the setting where $f : \mathbb{R}^d \to \mathbb{R}$ and thus only in the case of regression and binary classification. For the multi-class case, we could treat it as a multinomial regression problem, where the unknown function $f : \mathbb{R}^d \to \mathbb{S}^c$ where $\mathbb{S}^c = \{\mathbf{y} \in \mathbb{R}^c | \mathbf{y} \geq 0, \mathbf{y}^T\mathbf{1} = 1\}$. This leads to a similar operator based on computing the Jacobian of this vector valued function, which we call the Expected Jacobian Outer Product (EJOP).

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\mathbf{J}_f(\mathbf{x}) \mathbf{J}_f(\mathbf{x})^T \right)$$

We describe metric weighing experiments for non-parametric classification and also show that a simple estimator for the EJOP remains statistically consistent under mild assumptions in Chapter 7.

To summarize, in this part of the thesis, we make the following contributions:

1 We describe a simple estimator for the Expected Gradient Outerproduct (EGOP)

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x})^{\top} \right).$$

and show that it remains statistically consistent under mild assumptions.

2 We use the EGOP (with a spectral decomposition $VDV^\top$) in non-parameteric regression by using it to transform the inputs as $\mathbf{x}$ as $D^{1/2}V^\top\mathbf{x}$ and show it improves performance in several real world datasets.

3 We extend the EGOP to the multiclass case, proposing a variant called the Expected Jacobian Outer Product (EJOP)

$$\mathbb{E}_{\mathbf{x}}G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}}\left(\mathbf{J}_f(\mathbf{x})\mathbf{J}_f(\mathbf{x})^T\right)$$

4 For the EJOP, we propose a simple estimator and also prove that it remains statistically consistent under reasonable assumptions.

5 Similarly to the case of the EGOP, we use the EJOP for transforming the input space and also demonstrate that it improves performance in various non-parametric classification tasks.

6 THE EXPECTED GRADIENT OUTER PRODUCT

In high dimensional classification and regression problems, the task is to infer the unknown, smooth function $f : \mathcal{X} \rightarrow \mathcal{Y}$. To this end, we are provided n of functional estimates that comprises our data. In other words, we have $\{(\mathbf{x}, \mathbf{y})_1, (\mathbf{x}, \mathbf{y})_2, \ldots, (\mathbf{x}, \mathbf{y})_n\}$, with $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ and the labels $\mathbf{y}_i \approx f(\mathbf{x}_i)$ i.e. they are noisy versions of the function values. We are interested in distance based regression, which provides our function estimate:

$$f_n(\mathbf{x}) = \sum_{i=1}^{n} w(\mathbf{x}, \mathbf{x}_i)\mathbf{y}_i$$

The weights $w(\mathbf{x}, \mathbf{x}_i)$, depend on the underlying metric. In Chapter 5 we made the case for metric estimation from gradients in case of a restricted computational budget, where optimizing over a space of metrics might not be feasible. In particular, in the case of regression and binary classification, we consider the metric given by the Expected Gradient Outerproduct, which is written as.

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x})^\top \right).$$

Originally proposed in the context of multi-index regression, the EGOP recovers the average variation of f in all directions. To see this: For some $\mathbf{v} \in \mathbb{R}^d$, the directional derivative at $\mathbf{x}$ along $\mathbf{v}$ is given by $f'_{\mathbf{v}}(\mathbf{x}) = \nabla f(\mathbf{x})^\top \mathbf{v}$, in other words

$$\mathbb{E}_X \left| f'_{\mathbf{v}}(\mathbf{x}) \right|^2 = \mathbb{E}_{\mathbf{x}} \left(\mathbf{v}^\top G(\mathbf{x}) \mathbf{v} \right) = \mathbf{v}^\top \left(\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \right) \mathbf{v}$$

From the above, it follows that, if f does not vary along $\mathbf{v}$, $\mathbf{v}$ must be in the null-space of the EGOP matrix $\mathbb{E}_X G(X)$, since $\mathbb{E}_X \left| f'_v(X) \right|^2 = 0$. [165] infact show that considering f is continuously differentiable on a compact space $\mathcal{X}$, the column space of $\mathbb{E}_X G(X)$ is exactly the *relevant* subspace defined by P (recall that P is the relevant subspace defined in Chapter 5 i.e. $f(\mathbf{x}) = g(P\mathbf{x})$ with $P \in \mathbb{R}^{k \times d}$, with P being the subspace most relevant to predicting the output y)

As already discussed in Chapter 5, the EGOP is useful beyond the multi-index setting, where its utility is to recover the relevant subspace P. That is, we might expect that in most practical, real world settings, a clear relevant subspace might not exist. Nevertheless, we can still expect that f does not vary uniformly in all directions. The usefulness of the EGOP in such settings can be to weight any direction $\mathbf{v} \in \mathbb{R}^d$ according to its relevance as captured by the average variation of f along $\mathbf{v}$ (encoded in the EGOP). It is this weighting use of the EGOP that we will consider in this chapter.

The estimation of the EGOP can be done in various sophisticated ways, which can however be prohibitively expensive. For instance an optimal way of estimating $\nabla f(x)$, and hence the EGOP, is to estimate the slope of a linear approximation to f locally at each $x = X_i$ in an n-sample $\{(X_i, Y_i)\}_{i=1}^n$. Local linear fits can however be prohibitively expensive since it involves multiplying and inverting large-dimensional matrices at all X_i. This can render the approach impractical although it is otherwise well motivated.

One of the main messages of the work discussed in this chapter is that the EGOP does need to be estimated optimally for the utility of it that we discussed above. That is, it just needs to be estimated well enough to use towards improving classification and regression. To this end, we consider the following cheap, albeit very rough estimator. Let f_n denote an initial estimate of f (we use a kernel estimate); for the i-th coordinate of $\nabla f(x)$, we use the rough estimate

$$\Delta_{t,i} f_n(x) = \frac{(f_n(x + te_i) - f_n(x - te_i))}{2t}, \; t > 0.$$

Now, let $G_n(x)$ be the outer-product of the resulting gradient estimate $\hat{\nabla} f_n(x)$, the EGOP is estimated as $\mathbb{E}_n G_n(X)$, the empirical average of G_n. The exact procedure is given in Section 6.2.1.

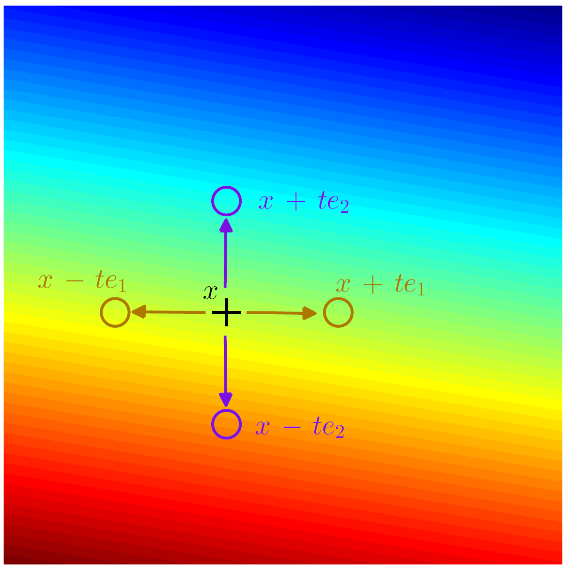


Figure 6.1: A simple illustration of the difference based gradient estimator when $\mathbf{x} \in \mathbb{R}^2$. We perturb the input along each coordinate, record the value of $f_{n,h}(\mathbf{x})$, and get a finite difference estimate. The background color represents the functional values

We must first demonstrate that this rough estimator is, in fact, a sound estimator. To this end, we show it remains a statistically consistent estimate of the EGOP under very general distributional conditions. These assumptions being milder than the usual conditions on proper gradient estimation (for detailed assumptions see Section 6.2.2). The main consistency result and key difficulties (having to do with interdependencies in the estimate) are discussed in Section 6.3.

We also show, through extensive experiments that preprocessing the data with this cheap EGOP estimate can still significantly improve the performance of non-parametric classification and regression procedures in many real-world datasets. The experimentation is described in Section 6.4. In the next Section 6.1, we first give a quick overview of some of the relevant work and place ours in context.

6.1 RELATED WORK

The work of Kpotufe *et al.* [85], [86] already briefly discussed in Chapter 5 was the direct inspiration for the work described in this chapter. Kpotufe *et al.* consider estimating the coordinates f_i' of ∇f in a similar fashion as described here. However, there is a notable difference, in that [85], [86] are only concerned with a variable selection setting. That is, each coordinate i of X is to be weighted by an estimate of $\mathbb{E}_X |f_i'(X)|$, which is their quantity of interest. In this chapter we consider the more general approach of estimating the Expected Gradient Outerproduct. We also study its consistency and applicability in the context of non-parametric classification and regression.

Another body of literature, which is quite closely related to the work described in this chapter, is in the context of methods for multi-index regression. Many methods developed for doing multi-index regression use the so-called *inverse regression* approach (e.g. [96]), and many of them operate by incorporating estimates of derivate functionals of the unknown f. These approaches can be found in works as early as [123], and typically estimate ∇f as the slope of local linear approximations of f.

Comparatively recent works of [112, 165] build a much clearer bridge between various approaches to multi-index regression. In particular, they also related the EGOP to the *covariance*-type matrices typically estimated in inverse regression. Besides, [112, 165] also propose an alternative estimator for the gradient, rather than using local linear slopes. Their approach estimates ∇f via a regularized least-squares objective over an Reproduced Kernel Hibert Space. This approach is still expensive, since the least-square solution involves inverting an $n \times n$ feature matrix. In contrast our less sophisticated approach will take time in the order of n times the time to estimate f_n (in practice, we could employ fast range search methods when f_n is a fast kernel regressor).

As already described, the primary utility of the EGOP in multi-index regression is to recover the *relevant* subspace given by P in the model $f(\mathbf{x}) = g(P\mathbf{x})$. The data can first be projected to this subspace before doing predicting on the projected data.

In this chapter, we do not make a case for any particular methodology that leverages the EGOP for preprocessing the data. Instead, our experiments focus on the use of EGOP as a metric for distance based non-parametric regression and classification. That is, suppose $VDV^\top$ is the spectral decomposition of the estimated EGOP, we then use this to transform the input as follows $\mathbf{x} \mapsto D^{1/2}V^\top\mathbf{x}$. Our use of the EGOP does not rely on the multi-index model holding, but rather on a more general model where P might be a full-dimensional rotation (i.e. all directions are relevant), but g varies more in some coordinate than in others. The diagonal element $D_{i,i}$ recovers $\mathbb{E}_\mathbf{x}(g_i'(\mathbf{x}))^2$ where g_i' denotes coordinate i of ∇g, while $V^\top$ recovers P.

6.2 SETUP AND DEFINITIONS

We consider a regression or classification setting where the input X belongs to a space $\mathcal{X} \subset \mathbb{R}^d$, of bounded diameter 1. The output Y is real. We are interested in the unknown *regression function* $f(x) \triangleq \mathbb{E}[Y|X = x]$ (in the case of classification with $Y \in \{0, 1\}$, this is just the probability of 1 given x).

For a vector $x \in \mathbb{R}^d$, let $\|x\|$ denote the Euclidean norm, while for a matrix A, let $\|A\|_2$ denote the spectral norm, i.e. the largest singular value $\sigma_{\max}(A)$.

We use $\mathrm{im}(A)$ to denote the column space of matrix $A \in R^{n\times m}$: $\mathrm{im}(A) = \{\mathbf{Y} \in \mathbb{R}^n : \mathbf{Y} = A\mathbf{x} \text{ for some } \mathbf{x} \in \mathbb{R}^m\}$, and the $\ker(A)$ to denote the null space of matrix $A \in R^{n\times m}$: $\ker(A) = \{\mathbf{x} \in \mathbb{R}^m : A\mathbf{x} = 0\}$. We use $A \circ B$ to denote the entry-wise product of matrices A and B.

As a little aside, we use both $\mathbf{x}$ and X to refer to d dimensional vectors ($\mathbf{X}$ to a matrix) in this chapter henceforth as well as the next chapter. The purpose of latching onto this notational freedom will be clear from the proofs, where working with $\mathbf{x}$ can cause confusion.

6.2.1 *Estimation procedure for the Expected Gradient Outerproduct*

We let μ denote the marginal of $P_{X,Y}$ on $\mathcal{X}$ and we let μ_n denote its empirical counterpart on a random sample $\mathbf{X} = \{X_i\}_{i=1}^n$. Given a labeled sample $(\mathbf{X}, \mathbf{Y}) = \{(X_i, Y_i)\}_1^n$ from $P^n_{X,Y}$, we estimate the EGOP as follows.

We consider a simple kernel estimator defined below, using a Kernel K satisfying the following admissibility conditions:

Definition 1 (Admissible Kernel). *$K : \mathbb{R}_+ \mapsto \mathbb{R}_+$ is nonincreasing, $K > 0$ on $[0, 1)$, and $K(1) = 0$.*

Using such an admissible kernel K, and a bandwidth $h > 0$, we consider the regression estimate $f_{n,h}(\mathbf{x}) = \sum_i \omega_i(\mathbf{x}) Y_i$ where

$$\omega_i(x) = \frac{K(\|x - X_i\|/h)}{\sum_j K(\|x - X_j\|/h)} \text{ if } B(x,h) \cap \mathbf{X} \neq \varnothing,$$
$$\omega_i(x) = \frac{1}{n} \text{ otherwise.}$$

For any dimension $i \in [d]$, and $t > 0$, we first define

$$\Delta_{t,i} f_{n,h}(\mathbf{x}) \triangleq \frac{f_{n,h}(\mathbf{x} + te_i) - f_{n,h}(\mathbf{x} - te_i)}{2t}.$$

This is a rough estimate of the line-derivative along coordinate i. However, for a robust estimate we also need to ensure that enough sample points contribute to the estimate. To this end, given a confidence parameter $0 < \delta < 1$ (this definiton for δ is assumed in the rest of this work), define $A_{n,i}(X)$ as the event that

$$\min_{s\in\{t,-t\}} \mu_n(B(X + se_i, h/2)) \geq \frac{2d \ln 2n + \ln(4/\delta)}{n}.$$

The gradient estimate is then given by the vector

$$\hat{\nabla} f_{n,h}(\mathbf{x}) = \left(\Delta_{t,i} f_{n,h}(\mathbf{x}) \cdot \mathbf{1}_{A_{n,i}(\mathbf{x})}\right)_{i\in[d]}.$$

Note that, in practice we can just replace $A_{n,i}(X)$ with the event that the balls $B(X + se_i, h), s \in \{-t, t\}$, contain samples.

Finally, define $G_n(x)$ as the outer-product of $\hat{\nabla} f_{n,h}(\mathbf{x})$, we estimate $\mathbb{E}_X G(X)$ as

$$\mathbb{E}_n G_n(X) \triangleq \frac{1}{n} \sum_{i=1}^{n} \hat{\nabla} f_{n,h}(X_i) \cdot \hat{\nabla} f_{n,h}(X_i)^\top.$$

6.2.2 *Distributional Quantities and Assumptions*

For the analysis, our assumptions are quite general. In fact we could simply assume, as is common, that μ has lower-bounded density on a compact support $\mathcal{X}$, and that f is continuously differentiable; all the assumptions below will then hold. We list these more general detailed assumptions to better understand the minimal distributional requirements for consistency of our EGOP estimator.

A1 (Noise)**.** Let $\eta(X) \triangleq Y - f(X)$. We assume the following general noise model: $\forall \delta > 0$ there exists $c > 0$ such that $\sup_{x \in \mathbf{X}} \mathbb{P}_{Y|X=x} (|\eta(x)| > c) \leq \delta$. We denote by $C_Y(\delta)$ the infimum over all such c. For instance, suppose $\eta(X)$ has exponentially decreasing tail, then $\forall \delta > 0$, $C_Y(\delta) \leq O(\ln 1/\delta)$.

Last the variance of $(Y|X = x)$ is upper-bounded by a constant σ_Y^2 uniformly over $x \in \mathbf{X}$. The next assumption is standard for nonparametric regression/classification.

A2 (Bounded Gradient)**.** Define the *τ-envelope* of $\mathcal{X}$ as $\mathcal{X} + B(0, \tau) \triangleq \{z \in B(x, \tau), x \in \mathcal{X}\}$.

We assume there exists τ such that f is continuously differentiable on the τ-envelope $\mathcal{X} + B(0, \tau)$. Furthermore, for all $x \in \mathcal{X} + B(0, \tau)$, we have $\|\nabla f(x)\| \leq R$ for some $R > 0$, and ∇f is uniformly continuous on $\mathcal{X} + B(0, \tau)$ (this is automatically the case if the support $\mathcal{X}$ is compact).

The next assumption generalizes common smoothness assumptions: it is typically required for gradient estimation that the gradient itself be Hölder continuous (or that f be second-order smooth). These usual assumptions imply the more general assumptions below.

A3 (Modulus of continuity of ∇f)**.** Let $\epsilon_{t,i} = \sup_{\mathbf{x} \in \mathcal{X}, s \in [-t,t]} |f_i'(\mathbf{x}) - f_i'(\mathbf{x} + s e_i)|$. We assume $\epsilon_{t,i} \xrightarrow{t \to 0} 0$ which is for instance the case when ∇f is uniformly continuous on an envelope $\mathcal{X} + B(0, \tau)$.
The next two assumptions capture some needed regularity conditions on the marginal μ. To enable local approximations of $\nabla f(x)$ over $\mathcal{X}$, the marginal μ should not concentrate on the boundary of $\mathcal{X}$. This is captured in the following assumption.

A4 (Boundary of $\mathcal{X}$)**.** Define the (t, i)**-boundary** of $\mathcal{X}$ as $\partial_{t,i}(\mathcal{X}) = \{\mathbf{x} : \{\mathbf{x} + t e_i, x - t e_i\} \not\subseteq \mathcal{X}\}$. Define the vector $\mu_{\partial_t} = (\mu(\delta_{t,i}(\mathcal{X})))_{i \in [d]}$. We assume that $\mu_{\partial_t} \xrightarrow{t \to 0} \mathbf{0}$. This is for instance the case if μ has a continuous density on $\mathcal{X}$.
Finally we assume that μ has mass everywhere, so that for samples X in dense regions, $X \pm t e_i$ is also likely to be in a dense region.

A5 (Full-dimensionality of μ)**.** For all $x \in \mathcal{X}$ and $h > 0$, we have $\mu(B(x, h)) \geq C_\mu h^d$. This is for instance the case if μ has a lower-bounded density on $\mathcal{X}$.

6.3 CONSISTENCY OF THE ESTIMATOR $\mathbb{E}_n G_n(X)$ OF $\mathbb{E}_X G(X)$

We establish consistency by bounding $\|\mathbb{E}_n G_n(X) - \mathbb{E}_X G(X)\|_2$ for finite sample size n. The main technical difficulties in establishing the main result below have to do with the fact that each gradient approximation $\Delta_{t,h} f_{n,h}(X)$ at a sample point X depends on all other samples in $\mathbf{X}$. These inter-dependencies are circumvented by proceeding in steps which consider related quantities that are less sample-dependent.

Theorem 1 (Main). *Assume A1, A2 and A5. Let $t < \tau$ and suppose $h \geq (\log^2(n/\delta)/n)^{1/d}$. There exist $C = C(\mu, K(\cdot))$ and $N = N(\mu)$ such that the following holds with probability at least $1 - 2\delta$. Define $A(n) = \sqrt{Cd \cdot \log(n/\delta)} \cdot C_Y^2(\delta/2n) \cdot \sigma_Y^2 / \log^2(n/\delta)$. Suppose $n \geq N$, we have:*

$$\|\mathbb{E}_n G_n(X)] - \mathbb{E}_X G(X)\|_2 \leq \frac{6R^2}{\sqrt{n}} \left(\sqrt{\ln d} + \sqrt{\ln \frac{1}{\delta}} \right) +$$
$$\left(3R + \|\epsilon_t\| + \sqrt{d} \left(\frac{hR + C_Y(\delta/n)}{t} \right) \right) \cdot \Bigg[\|\epsilon_t\| +$$
$$\frac{\sqrt{d}}{t} \sqrt{\frac{A(n)}{nh^d} + 2h^2 R^2} + R \left(\sqrt{\frac{d \ln \frac{d}{\delta}}{2n}} + \|\mu_{\partial_t}\| \right) \Bigg]$$

Proof. Start with the decomposition

$$\begin{aligned} \|\mathbb{E}_n G_n(X) - \mathbb{E}_X G(X)\|_2 \leq & \|\mathbb{E}_n G(X) - \mathbb{E}_X G(X)\|_2 \\ & + \|\mathbb{E}_n G_n(X) - \mathbb{E}_n G(X)\|_2. \end{aligned} \tag{6.1}$$

The two terms of the r.h.s. are bounded separately in Lemma 2 and 12. □

Remark. Under the additional assumptions A3 and A4, the theorem implies consistency for $t \xrightarrow{n\to\infty} 0$, $h \xrightarrow{n\to\infty} 0$, $h/t^2 \xrightarrow{n\to\infty} 0$, and $(n/\log n) h^d t^4 \xrightarrow{n\to\infty} \infty$, this is satisfied for many settings, for example $t \propto h^{1/4}$, $h \propto (1/n)^{1/(2(d+1))}$.

The bound on the first term of (6.1) is a direct result of the below concentration bound for random matrices:

Lemma 1. *[68, 149]. Consider a random matrix $A \in \mathbb{R}^{d \times d}$ with bounded spectral norm $\|A\|_2 \leq M$. Let $A_1, A_2, ..., A_n$ be i.i.d. copies of A. With probability at least $1 - \delta$, we have*

$$\left\| \frac{1}{n} \sum_{i=1}^{n} A_i - \mathbb{E} A \right\|_2 \leq \frac{6M}{\sqrt{n}} \left(\sqrt{\ln d} + \sqrt{\ln \frac{1}{\delta}} \right).$$

We apply the above concentration to the i.i.d. matrices $G(X), X \in \mathbf{X}$, using the fact that $\|G(X)\|_2 = \|\nabla f(X)\|^2 \leq R^2$.

Lemma 2. *Assume A2. With probability at least $1-\delta$ over the i.i.d sample $\mathbf{X} \triangleq \{X_i\}_{i=1}^n$, we have*

$$\|\mathbb{E}_n G(X) - \mathbb{E}_X G(X)\|_2 \le \frac{6R^2}{\sqrt{n}} \left(\sqrt{\ln d} + \sqrt{\ln \frac{1}{\delta}} \right).$$

The next Lemma provides an initial bound on the second term of (6.1).

Lemma 3. *Fix the sample $(\mathbf{X}, \mathbf{Y})$. We have:*

$$\begin{aligned} \|\mathbb{E}_n G_n(X) - \mathbb{E}_n G(X)\|_2 \le & \mathbb{E}_n \|\nabla f(X) - \hat{\nabla} f_{n,h}(X)\| \\ & \cdot \max_{x \in \mathbf{X}} \|\nabla f(x) + \hat{\nabla} f_{n,h}(x)\|. \end{aligned} \tag{6.2}$$

Proof. We have by a triangle inequality $\|\mathbb{E}_n G_n(X) - \mathbb{E}_n G(X)\|_2$ is bounded by:

$$\mathbb{E}_n \left\| \left(\hat{\nabla} f_{n,h}(X) \cdot \hat{\nabla} f_{n,h}(X)^\top - \nabla f(X) \cdot \nabla f(X)^\top \right) \right\|_2.$$

To bound the r.h.s above, we use the fact that, for vectors a, b, we have

$$aa^\top - bb^\top = \frac{1}{2}(a-b)(b+a)^\top + \frac{1}{2}(b+a)(a-b)^\top,$$

implying that

$$\begin{aligned} \left\| aa^\top - bb^\top \right\|_2 \le & \frac{1}{2} \left\| (a-b)(b+a)^\top \right\|_2 \\ & + \frac{1}{2} \left\| (b+a)(a-b)^\top \right\|_2 \\ = & \left\| (b+a)(a-b)^\top \right\|_2 \end{aligned}$$

since the spectral norm is invariant under matrix transposition.
We therefore have that $\|\mathbb{E}_n G_n(X) - \mathbb{E}_n G(X)\|_2$ is at most

$$\begin{aligned} & \mathbb{E}_n \|(\nabla f(X) - \hat{\nabla} f_{n,h}(X)) \cdot (\nabla f(X) + \hat{\nabla} f_{n,h}(X))^\top\|_2 \\ & = \mathbb{E}_n \|\nabla f(X) - \hat{\nabla} f_{n,h}(X)\| \cdot \|\nabla f(X) + \hat{\nabla} f_{n,h}(X)\| \\ & \le \mathbb{E}_n \|\nabla f(X) - \hat{\nabla} f_{n,h}(X)\| \cdot \max_{x \in \mathbf{X}} \|\nabla f(x) + \hat{\nabla} f_{n,h}(x)\|. \end{aligned}$$

□

Thus the matrix estimation problem is reduced to that of an average gradient estimation. The two terms of (6.2) are bounded in the following two subsections. These sections thus contain the bulk of the analysis. All omitted proofs are found in the supplementary.

6.3.1 *Bound on* $\mathbb{E}_n \|\nabla f(X) - \hat{\nabla} f_{n,h}(X)\|$

The analysis of this section relies on a series of approximations. In particular we relate the vector $\hat{\nabla} f_{n,h}(\mathbf{x})$ to the vector

$$\tilde{\nabla} f(\mathbf{x}) \triangleq \left(\Delta_{t,i} f(\mathbf{x}) \cdot \mathbf{1}_{A_{n,i}(\mathbf{x})}\right)_{i\in[d]}.$$

In other words we start with the decomposition:

$$\begin{aligned}\mathbb{E}_n \|\nabla f(X) - \hat{\nabla} f_{n,h}(X)\| \leq & \mathbb{E}_n \|\nabla f(X) - \tilde{\nabla} f(X)\| \\ & + \mathbb{E}_n \|\tilde{\nabla} f(X) - \hat{\nabla} f_{n,h}(X)\|. \end{aligned} \tag{6.3}$$

We bound each term separately in the following subsections.

6.3.1.1 *Bounding* $\mathbb{E}_n \|\nabla f(X) - \tilde{\nabla} f(X)\|$

We need to introduce vectors $\mathbf{I}_n(x) \triangleq \left(\mathbf{1}_{A_{n,i}(\mathbf{x})}\right)_{i\in[d]}$, and $\overline{\mathbf{I}_n(x)} \triangleq \left(\mathbf{1}_{\bar{A}_{n,d}(\mathbf{x})}\right)_{i\in[d]}$. We then have:

$$\begin{aligned}\mathbb{E}_n \|\nabla f(X) - \tilde{\nabla} f(X)\| \leq & \ \mathbb{E}_n \|\nabla f(X) \circ \overline{\mathbf{I}_n(X)}\| \\ & + \mathbb{E}_n \|\nabla f(X) \circ \mathbf{I}_n(X) - \tilde{\nabla} f(X)\|. \end{aligned} \tag{6.4}$$

The following lemma bounds the first term of (6.4).

Lemma 4. *Assume A2 and A5. Suppose* $h \geq (\log^2(n/\delta)/n)^{1/d}$. *With probability at least* $1-\delta$ *over the sample of* $\mathbf{X}$*:*

$$\mathbb{E}_n \left\|\nabla f(X) \circ \overline{\mathbf{I}_n(X)}\right\| \leq R \cdot \left(\sqrt{\frac{d \ln \frac{d}{\delta}}{2n}} + \|\mu_{\partial_t}\|\right).$$

Proof. By assumption, $\|\nabla f(\mathbf{x})\| \leq R$, so we have

$$\mathbb{E}_n \left\|\nabla f(X) \circ \overline{\mathbf{I}_n(X)}\right\| \leq R \cdot \mathbb{E}_n \left\|\overline{\mathbf{I}_n(X)}\right\|. \tag{6.5}$$

We bound $\left\|\overline{\mathbf{I}_n(X)}\right\|$ as follows. For any $i \in [d]$, define the events $A_i(X) \equiv \min_{\{t,-t\}} \mu(B(X + se_i, h/2)) \geq 3 \cdot \frac{2d \ln 2n + \ln(4/\delta)}{n}$, and define the vector $\overline{\mathbf{I}(X)} \triangleq \left(\mathbf{1}_{\bar{A}_i(X)}\right)_{i\in[d]}$.
By relative VC bounds [151], let $\alpha_n = \frac{2d \ln 2n + \ln(4/\delta)}{n}$, then with probability at least $1-\delta$ over the choice of $\mathbf{X}$, for all balls $B \in R^d$ we have $\mu(B) \leq \mu_n(B) + \sqrt{\mu_n(B)\alpha_n} + \alpha_n$. Therefore, with probability at least $1-\delta$, $\forall i \in [d]$ and x in the sample $\mathbf{X}$, $\bar{A}_{n,i}(x) \Rightarrow \bar{A}_i(x)$.
Moreover, since $\|\overline{\mathbf{I}(X)}\| \leq \sqrt{d}$, by Hoeffding's inequality,

$$\mathbb{P}(\mathbb{E}_n \|\overline{\mathbf{I}(X)}\| - \mathbb{E}_X \|\overline{\mathbf{I}(X)}\| \geq \epsilon) \leq e^{-\frac{2n\epsilon^2}{d}}.$$

It follows that, with probability at least $1-\delta$,

$$\begin{aligned}\mathbb{E}_n\|\overline{\mathbf{I}_n(X)}\| \leq& \mathbb{E}_n\|\overline{\mathbf{I}(X)}\| \\ \leq& \mathbb{E}_X\|\overline{\mathbf{I}(X)}\| + \sqrt{\frac{d\ln\frac{1}{\delta}}{2n}} \\ \leq& \sqrt{\mathbb{E}_X\|\overline{\mathbf{I}(X)}\|^2} + \sqrt{\frac{d\ln\frac{1}{\delta}}{2n}}, \end{aligned} \tag{6.6}$$

by Jensen's inequality. We bound each of the d terms of $\mathbb{E}_X\|\overline{\mathbf{I}(X)}\|^2 = \sum_{i\in[d]}\mathbb{E}_X\mathbf{1}_{\bar{A}_i(X)}$ as follows.

Fix any $i\in[d]$. We have $\mathbb{E}_X\mathbf{1}_{\bar{A}_i(X)} \leq \mathbb{E}_X[\mathbf{1}_{\bar{A}_i(X)}|X\in\mathcal{X}\backslash\partial_{t,i}(\mathcal{X})] + \mu(\partial_{t,i}(\mathcal{X}))$. Notice that $\mathbb{E}_X[\mathbf{1}_{\bar{A}_i(X)}|X\in\mathcal{X}\backslash\partial_{t,i}(\mathcal{X})] = 0$ since, by assumption, $\mu(B(x+se_i, h/2)) \geq C_\mu(h/2)^d \geq 3\alpha$ whenever $h\geq(\log^2(n/\delta)/n)^{1/d}$. Hence, we have

$$\sqrt{\mathbb{E}_X\|\overline{\mathbf{I}(X)}\|^2} \leq \sqrt{\sum_{i\in[d]}\mu^2(\partial_{t,i}(\mathcal{X}))}.$$

Combine this last inequality with (6.5) and (6.6) and conclude. □

The second term of (6.4) is bounded in the next lemma.

Lemma 5. *Fix the sample* $\mathbf{X}$*. We have* $\max_{X\in\mathbf{X}}\|\nabla f(X)\circ\mathbf{I}_n(X) - \hat{\nabla}f(X)\| \leq \|\epsilon_t\|$.

Proof. For a given coordinate $i\in[d]$, let f_i' denote the directional derivative $e_i^\top\nabla f$ along i. Pick any $x\in\mathcal{X}$. Since $f(\mathbf{x}+te_i) - f(\mathbf{x}-te_i) = \int_{-t}^{t} f_i'(\mathbf{x}+se_i)ds$, we have

$$\begin{aligned} 2t(f_i'(\mathbf{x}) - \epsilon_{t,i}) \leq& f(\mathbf{x}+te_i) - f(\mathbf{x}-te_i) \\ \leq& 2t(f_i'(\mathbf{x}) + \epsilon_{t,i}) \end{aligned}$$

Thus $|\frac{1}{2t}(f(\mathbf{x}+te_i) - f(\mathbf{x}-te_i)) - f_i'(\mathbf{x})| \leq \epsilon_{t,i}$. We therefore have that $\|\nabla f(\mathbf{x})\circ\mathbf{I}_n(x) - \hat{\nabla}f(\mathbf{x})\|$ equals

$$\begin{aligned} &\sqrt{\sum_{i=1}^{d}\left(f_i'(\mathbf{x})\cdot\mathbf{1}_{A_{n,i}(\mathbf{x})} - \Delta_{t,i}f(\mathbf{x})\cdot\mathbf{1}_{A_{n,i}(\mathbf{x})}\right)^2} \\ &= \sqrt{\sum_{i=1}^{d}\left(\frac{1}{2t}(f(\mathbf{x}+te_i) - f(\mathbf{x}-te_i)) - f_i'(\mathbf{x})\right)^2} \\ &\leq \|\epsilon_t\|. \end{aligned}$$

□

The last two lemmas can then be combined using equation (6.4) into the final bound of this subsection.

Lemma 6. *Assume A2 and A5. Suppose* $h\geq(\log^2(n/\delta)/n)^{1/d}$*. With probability at least* $1-\delta$ *over the sample* $\mathbf{X}$*:*

$$\begin{aligned}\mathbb{E}_n\|\nabla f(X) - \hat{\nabla}f(X)\| \leq& R\cdot\left(\sqrt{\frac{d\ln\frac{d}{\delta}}{2n}} + \|\mu_{\partial_t}\|\right) \\ &+ \|\epsilon_t\|.\end{aligned}$$

6.3.1.2 *Bounding* $\mathbb{E}_n\|\hat{\nabla} f(X) - \hat{\nabla} f_{n,h}(X)\|$

We need to consider bias and variance functionals of estimates $f_{n,h}(x)$. To this end we introduce the expected estimate $\tilde{f}_{n,h}(x) = \mathbb{E}_{\mathbf{Y}|\mathbf{X}} f_{n,h}(x) = \sum_{i=1}^{n} w_i(x) f(X_i)$. The following lemma bounds the bias of estimates $f_{n,h}$. The proof relies on standard ideas.

Lemma 7 (Bias of $f_{n,h}$)**.** *Assume A2. Let $t < \tau$. We have for all $X \in \mathbf{X}$, all $i \in [d]$, and $s \in \{-t, t\}$:*

$$|\tilde{f}_{n,h}(X + se_i) - f(X + se_i)| \cdot \mathbf{1}_{A_{n,i}(\mathbf{x})} \leq hR.$$

Proof. Let $x = X + se_i$. Using a Taylor approximation on f to bound $|f(X_i) - f(x)|$, we have

$$\begin{aligned} |\tilde{f}_{n,h}(\mathbf{x}) - f(\mathbf{x})| &\leq \sum_{i\in[d]} w_i(x)|f(X_i) - f(x)| \\ &\leq \sum_{i\in[d]} w_i(x)\|X_i - x\| \cdot \sup_{\mathcal{X}+B(0,\tau)} \|\nabla f\| \\ &\leq hR. \end{aligned}$$

□

The following lemma bounds the variance of estimates $f_{n,h}$ averaged over the sample $\mathbf{X}$. To obtain a high probability bound, we relie on results of Lemma 7 in [85]. However in [85], the variance of the estimator if evaluated at a point, therefore requiring local density assumptions. The present lemma has no such local density requirements given that we are interested in an average quantity over a collection of points.

Lemma 8 (Average Variance)**.** *Assume A1. There exist $C = C(\mu, K(\cdot))$, such that the following holds with probability at least $1 - 2\delta$ over the choice of the sample $(\mathbf{X}, \mathbf{Y})$. Define $A(n) = \sqrt{Cd \cdot \ln(n/\delta)} \cdot C_Y^2(\delta/2n) \cdot \sigma_Y^2$, for all $i \in [d]$, and all $s \in \{-t, t\}$:*

$$\mathbb{E}_n|\tilde{f}_{n,h}(X + se_i) - f_{n,h}(X + se_i)|^2 \cdot \mathbf{1}_{A_{n,i}(X)} \leq \frac{A(n)}{nh^d}$$

Proof of Lemma 8. Fix the sample $\mathbf{X}$ and consider only the randomness in $\mathbf{Y}$. The following result is implicit to the proof of Lemma 7 of [85]: with probability at least $1 - 2\delta$, for all $X \in \mathbf{X}$, $i \in [d]$, and $s \in \{-t, t\}$, we have (where, for simplicity, we write $x = X + se_i$) $|\tilde{f}_{n,h}(x) - f_{n,h}(x)|^2 \cdot \mathbf{1}_{A_{n,i}(X)}$ is at most

$$\frac{Cd \cdot \log(n/\delta) C_Y^2(\delta/2n) \cdot \sigma_Y^2}{n\mu_n((B(x, h/2))}.$$

Fix $i \in [d]$ and $s \in \{-t, t\}$. Taking empirical expectation, we get $\mathbb{E}_n|\tilde{f}_{n,h}(x) - f_{n,h}(x)|^2$ is at most

$$\frac{\sqrt{Cd \cdot \ln(n/\delta)} \cdot C_Y^2(\delta/2n) \cdot \sigma_Y^2}{n} \sum_{j\in[n]} \frac{1}{n(x_j, h/2)}$$

where $x_j = X_j + se_i$, and $n(x_i, h/2) = n\mu_n(B(x_i, h/2))$ is the number of samples in $B(x_i, h/2)$. Let $\mathcal{Z} \subset \mathbb{R}^d$ denote a minimal $h/4$-cover of $\{X_1, ..., X_n\}$. Since $\mathcal{X}$ has bounded

diameter, such a cover has size at most $C_{\mathcal{X}}(h/4)^d$ for some $C_{\mathcal{X}}$ depending on the support $\mathcal{X}$ of μ.
Assume every x_j is assigned to the closest $z \in \mathcal{Z}$, where ties can be broken any way, and write $x_j \to z$ to denote such an assignment. By definition of Z, x_j is contained in the ball $B(z, h/4)$, and we therefore have $B(z, h/4) \subset B(x_j, h/2)$.

Thus

$$\begin{aligned}\sum_{j\in[n]} \frac{1}{n(x_j, h/2)} &= \sum_{z\in\mathcal{Z}} \sum_{x_j\to z} \frac{1}{n(x_j, h/2)} \\ &\leq \sum_{z\in\mathcal{Z}} \sum_{x_j\to z} \frac{1}{n(z, h/4)} \\ &\leq \sum_{z\in\mathcal{Z}} \frac{n(z, h/4)}{n(z, h/4)} = |\mathcal{Z}| \leq C_{\mathcal{X}}(h/4)^{-d}.\end{aligned}$$

Combining with the above analysis finishes the proof. □

The main bound of this subsection is given in the next lemma which combines the above bias and variance results.

Lemma 9. *Assume A1 and A2. There exist $C = C(\mu, K(\cdot))$, such that the following holds with probability at least $1 - 2\delta$ over the choice of $(\mathbf{X}, \mathbf{Y})$. Define $A(n) = \sqrt{Cd \cdot \ln(n/\delta)} \cdot C_Y^2(\delta/2n) \cdot \sigma_Y^2$:*

$$\mathbb{E}_n \|\hat{\nabla} f(X) - \hat{\nabla} f_{n,h}(X)\| \leq \frac{\sqrt{d}}{t} \sqrt{\frac{A(n)}{nh^d} + 2R^2h^2}.$$

Proof. In what follows, we first apply Jensen's inequality, and the fact that $(a+b)^2 \leq 2a^2 + 2b^2$. We have:

$$\begin{aligned}&\mathbb{E}_n \|\hat{\nabla} f(X) - \hat{\nabla} f_{n,h}(X)\| \\ &= \mathbb{E}_n \left(\sum_{i\in[d]} |\Delta_{t,i} f_{n,h}(X) - \Delta_{t,i} f(X)|^2 \cdot \mathbf{1}_{A_{n,i}(X)} \right)^{1/2} \\ &\leq \left(\sum_{i\in[d]} \mathbb{E}_n |\Delta_{t,i} f_{n,h}(X) - \Delta_{t,i} f(X)|^2 \cdot \mathbf{1}_{A_{n,i}(X)} \right)^{1/2} \\ &\leq \frac{\sqrt{d}}{2t} \left(\max_{i\in[d], s\in\{-t,t\}} 4\mathbb{E}_n |f_{n,h}(\tilde{X}) - f(\tilde{X})|^2 \cdot \mathbf{1}_{A_{n,i}(X)} \right)^{1/2} \qquad (6.7)\end{aligned}$$

where $\tilde{X} = X + se_i$. Next, use the fact that for any $s \in \{-t, t\}$, we have the following decomposition into variance and bias terms

$$\begin{aligned}&|f_{n,h}(X + se_i) - f(X + se_i)|^2 \\ &\leq 2|f_{n,h}(X + se_i) - \tilde{f}_{n,h}(X + se_i)|^2 \\ &+ 2|\tilde{f}_{n,h}(X + se_i) - f(X + se_i)|^2.\end{aligned}$$

Combine this into (6.7) to get a bound in terms of the average bias and variance of estimates $f_{n,h}(X + se_i)$. Apply Lemma 7 and 8 and conclude. □

6.3.1.3 *Main Result of this Section*

The following theorem provides the final bound of this section on $\mathbb{E}_n \|\nabla f(X) - \hat{\nabla} f_{n,h}(X)\|$. It follows directly from the decomposition of equation 6.3 and Lemmas 6 and 21.

Lemma 10. *Assume A1, A2 and A5. Let $t < \tau$ and suppose $h \geq (\log^2(n/\delta)/n)^{1/d}$. With probability at least $1-2\delta$ over the choice of the sample $(\mathbf{X}, \mathbf{Y})$, we have*

$$\begin{aligned}\mathbb{E}_n \|\nabla f(X) - \hat{\nabla} f_{n,h}(X)\| \leq & \frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d} + 2R^2h^2} \\ & + R\left(\sqrt{\frac{d \ln \frac{d}{\delta}}{2n}} + \|\mu_{\partial_t}\|\right) + \|\epsilon_t\|.\end{aligned}$$

6.3.2 *Bounding* $\max_{X \in \mathbf{X}} \|\nabla f(X) + \hat{\nabla} f_{n,h}(X)\|$

Lemma 11. *Assume A1 and A2. With probability at least $1-\delta$, we have*

$$\begin{aligned}\|\nabla f(X) + \hat{\nabla} f_{n,h}(X)\| \leq & 3R + \|\epsilon_t\| \\ & + \sqrt{d}\left(\frac{hR + C_Y(\delta/n)}{t}\right).\end{aligned}$$

Proof. Fix $X \in \mathbf{X}$.We have

$$\begin{aligned}\|\nabla f(X) + \hat{\nabla} f_{n,h}(X)\| \leq & 2\|\nabla f(X)\| \\ & + \|\nabla f(\mathbf{x}) - \hat{\nabla} f_{n,h}(X)\| \\ \leq & 2R + \|\nabla f(X) - \hat{\nabla} f(\mathbf{x})\| \\ & + \|\hat{\nabla} f(X) - \hat{\nabla} f_{n,h}(X)\|. \end{aligned} \tag{6.8}$$

We can bound the second term of (6.8) above as follows.

$$\begin{aligned}\|\nabla f(X) - \hat{\nabla} f(X)\| \leq & \|\nabla f(X) \circ \mathbf{I}_n(X) - \hat{\nabla} f(X)\| \\ & + \|\nabla f(X) \circ \overline{\mathbf{I}_n(X)}\| \\ \leq & \|\epsilon_t\| + R,\end{aligned}$$

where we just applied Lemma 5.
For the third term of (6.8), $\|\hat{\nabla} f(\mathbf{x}) - \hat{\nabla} f_{n,h}(\mathbf{x})\|$ equals

$$\sqrt{\sum_{i \in [d]} (|\Delta_{t,i} f_{n,h}(\mathbf{x}) - \Delta_{t,i} f(\mathbf{x})| \cdot \mathbf{1}_{A_{n,i}(\mathbf{x})})^2}.$$

As in the proof of Lemma 21, we decompose the above summand into bias and variance terms, that is:

$$\begin{aligned}& |\Delta_{t,i} f_{n,h}(\mathbf{x}) - \Delta_{t,i} f(\mathbf{x})| \\ & \leq \frac{1}{t} \max_{s \in \{-t,t\}} |\tilde{f}_{n,h}(\mathbf{x} + se_i) - f(\mathbf{x} + se_i)| \\ & + \frac{1}{t} \max_{s \in \{-t,t\}} |\tilde{f}_{n,h}(\mathbf{x} + se_i) - f_{n,h}(\mathbf{x} + se_i)|.\end{aligned}$$

By Lemma 7, $|\tilde{f}_{n,h}(\mathbf{x}+se_i) - f(\mathbf{x}+se_i)| \leq Rh$ for any $s \in \{-t, t\}$.
Next, by definition of $C_Y(\delta/n)$, with probability at least $1-\delta$, for each $j \in [n]$, Y_j has value within $C_Y(\delta)$ of $f(X_j)$. It follows that $|\tilde{f}_{n,h}(X+se_i) - f_{n,h}(X+se_i)| \leq C_Y(\delta/n)$ for $s \in \{-t, t\}$.
Thus, with probability at least $1-\delta$, we have

$$\|\hat{\nabla} f(X) - \hat{\nabla} f_{n,h}(X)\| \leq \sqrt{d}\left(\frac{hR + C_Y(\delta/n)}{t}\right).$$

Combine these bounds in (6.8) and conclude. □

6.3.3 *Final Bound* $\|\mathbb{E}_n G_n(X) - \mathbb{E}_n G(X)\|_2$

We can now combine the results of the last two subsections, namely Lemma 10 and 11, into the next lemma, using the bound of Lemma 3.

Lemma 12. *Assume A1, A2 and A5. Let $t < \tau$ and suppose $h \geq (\log^2(n/\delta)/n)^{1/d}$. With probability at least $1-2\delta$ over the choice of the sample $(\mathbf{X}, \mathbf{Y})$, we have that $\|\mathbb{E}_n G_n(X) - \mathbb{E}_n G(X)\|_2$ is at most*

$$\left(3R + \|\epsilon_t\| + \sqrt{d}\left(\frac{hR + C_Y(\delta/n)}{t}\right)\right) \cdot \left[\frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d} + 2h^2R^2} + R\left(\sqrt{\frac{d\ln\frac{d}{\delta}}{2n}} + \|\mu_{\partial_t}\|\right) + \|\epsilon_t\|\right].$$

6.4 EXPERIMENTS

In this section we describe experiments aimed at evaluating the utility of EGOP as a metric estimation technique for regression or classification. We consider a family of non-parametric methods that rely on the notion of distance under a given Mahalanobis metric $\mathbf{M}$, computed as $(\mathbf{x}-\mathbf{x}')^T\mathbf{M}(\mathbf{x}-\mathbf{x}')$.

In this setup, we consider three choices of $\mathbf{M}$: (i) identity, i.e., Euclidean distance in the original space; (ii) the estimated gradient weights (GW) matrix as in [85], i.e., Euclidean distance weighted by the estimated $\Delta_{t,i} f_n$, and (iii) the estimated EGOP matrix $\mathbb{E}_n G_n(X)$. The latter corresponds to Euclidean distance in the original space under linear transform given by $[\mathbb{E}_n G_n(X)]^{1/2}$. Note that a major distinction between the metrics based on GW and EGOP is that the former only scales the Euclidean distance, whereas the latter introduces a rotation.

Each choice of $\mathbf{M}$ can define the set of neighbors of an input point x in two ways: (a) k nearest neighbors (kNN) of x for a fixed k, or (b) neighbors with distance $\leq h$ for a fixed h; we will refer to this as hNN. When the task is regression, the output values of the neighbors are simply averaged; for classification, the class label for x is decided by majority vote among neighbors. Note that hNN corresponds to kernel regression with the boxcar kernel.

Thus, we will consider six methods, based on combinations of the choice of metric M and the definition of neighbhors: kNN, kNN-GW, kNN-EGOP, hNN, hNN-GW, and hNN-EGOP.

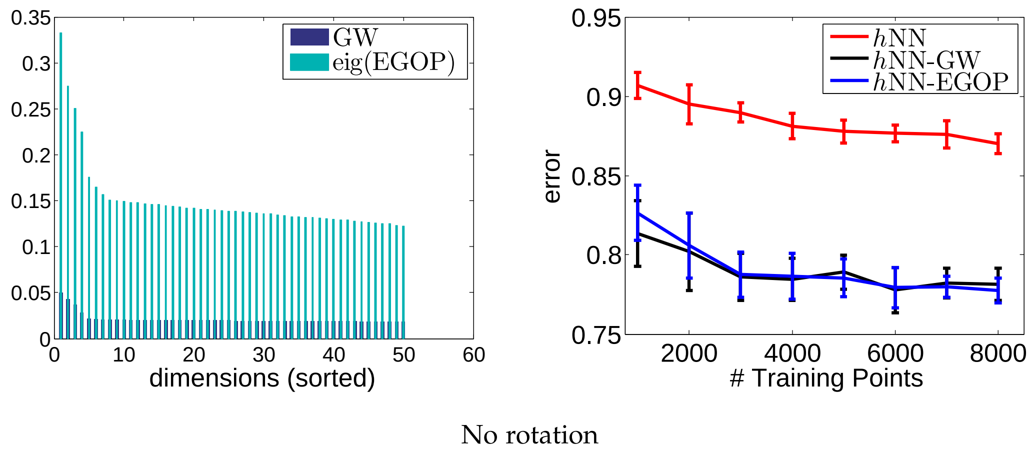


Figure 6.2: Synthetic data, d=50, without rotation applied after generating y from $\mathbf{x}$. The figure shows error of hNN with different metrics and the profile of derivatives recovered by GW and EGOP. In the case when there is no rotation, the performance of GW is similar to that returned by the EGOP

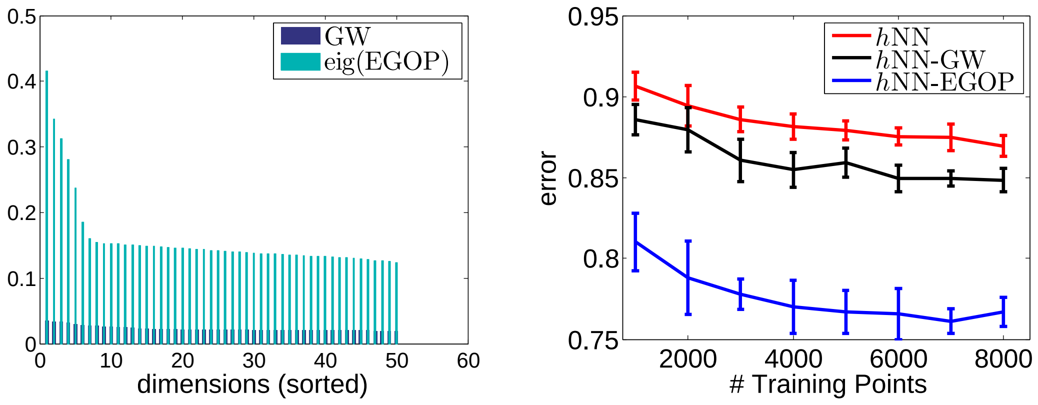
6.4.1 *Synthetic Data*

In order to understand the effect of varying the dependence of f on the input coordinates, on the quality of the metric estimated by the EGOP as well as other approaches, we first consider experiments on synthetic data. For the purpose of these experiments, the output is generated as follows: We set $y = \sum_i sin(c_i x_i)$, with the sum over all the dimensions of $\mathbf{x}\mathbb{R}^d$. The profile of the $\mathbf{c}$ vector is responsible for the degree upto which the value of $\mathbf{x}_i$ affects the output y.

We set $c[1] = 50$ and then $c[i] = 0.6 * c[i-1]$ for $i = 2 : 50$, and sampled $d = 50$-dimensional input over a bounded domain. In this data, we consider two cases: The first denoted (R), in which the input features are transformed by a random rotation in $\mathbb{R}^d$, *after* y has been generated; and the second, denoted (I) in which the input features are preserved. Under these conditions we evaluate the out of sample regression accuracy with original metric, GW and EGOP-based metrics, for different value of n; in each experiment, the values of h and t are tuned by cross-validation on the training set.

From the results that we can see in figures 6.2 and ?? is that reweighting examples by either gradient weights or by using the expected gradient outerproduct helps in performance in all cases. However, in the case when the synthetic data is rotated, as might be expected, the performance of the case when the EGOP is used for the reweighing, is not significantly affected as compared to the no rotation case. This is in sharp contrast to the case of gradient weights: which is able to recover a good metric (as can be seen by the accuracy) in the no-rotation case, however, its performance falls steeply when the data is rotated.

In order to get some insight into the nature of the metrics that were estimated from this synthetic data, we also plot profiles of the estimated feature relevance. For the gradients weights approach, these are just the weights obtained. For the EGOP, we use



Rotation

Figure 6.3: Synthetic data, d=50, with rotation applied after generating y from $\mathbf{x}$. As in the companion figure 6.2 we show error of hNN with different metrics, along with the profile of derivatives recovered by both GW and EGOP. The deterioration of the error performance of the Gradient Weights approach after the feature space is subject to a random rotation is noteworthy.

the eigenvalues of the matrix as a measure of feature importance. In other words, for gradient weights this corresponds to values on the diagonal of $\mathbf{M}$, and for EGOP of the (square roots) of the eigenvalues of $\mathbf{M}$. Plots in figures 6.2 and ?? also show these profiles (sorted in descending order). By inspecting at these profiles, it is clear that the EGOP is largely invariant to rotation of the feature space, and is much better at recovering the relevance of the features according to what was prescribed by the $\mathbf{c}$ vector described above.

6.4.2 *Regression Experiments*

After the experiments on synthetic data, we now present some results on real world datasets. The name of the datasets, along with information such as their dimensionality, number of training and test points etc., is mentioned in Table 6.1. For each data set, we report the results averaged over ten random training/test splits.

As a measure of performance we compute for each experiment the *normalized mean squared error* (nMSE): mean squared error over test set, divided by target variance over that set. This can be interpreted as fraction of variance in the target unexplained by the regressor.

In each experiment the input was normalized by the mean and standard deviation of the training set. For each method, the values of h or k as wel as t (the bandwidth used to estimate finite differences for GW and EGOP) were set by two fold cross-validation on the training set.

6.4.3 *Classification Experiments*

The setup for classification data sets is very similar for regression, except that the task is binary classification, and the labels of the neighbors selected by each prediction method are aggregated by simple majority vote, rather than averaging as in regression. The performance measure of interest here is classification error. As in regression experiments, we normalized the data, tuned all relevant parameters by cross validation on training data, and repeated the entire experimental procedure ten times with random training/test splits.

In addition to the baselines listed above, in classification experiments we considered another competitor: the popular feature relevance determination method called ReliefF [74, 84]. A highly engineered method that includes heuristics honed over considerable time by practitioners, it has the same general form of assigning weights to features as do GW and EGOP.

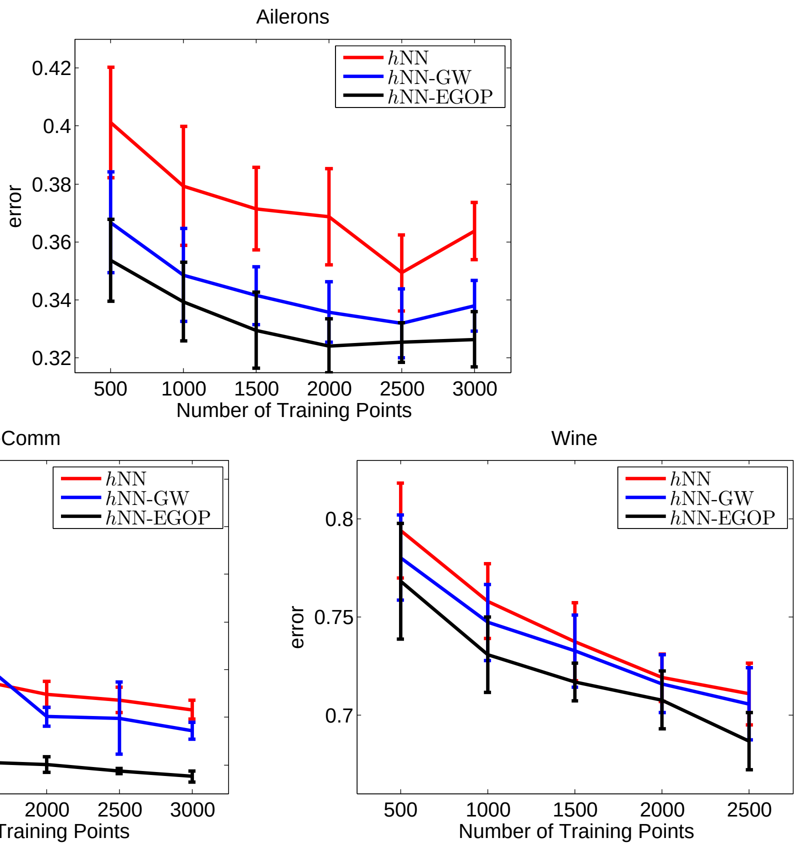


Figure 6.4: Regression error (nMSE) as a function of training set size for Ailerons, TeleComm, Wine data sets.

6.4.4 *Results*

The detailed results are reported in Tables 6.1 and 6.2. These correspond to a single value of training set size. Plots in Figures 6.4 and 6.5 show a few representative cases for regression and classification, respectively, of performance of different methods as a

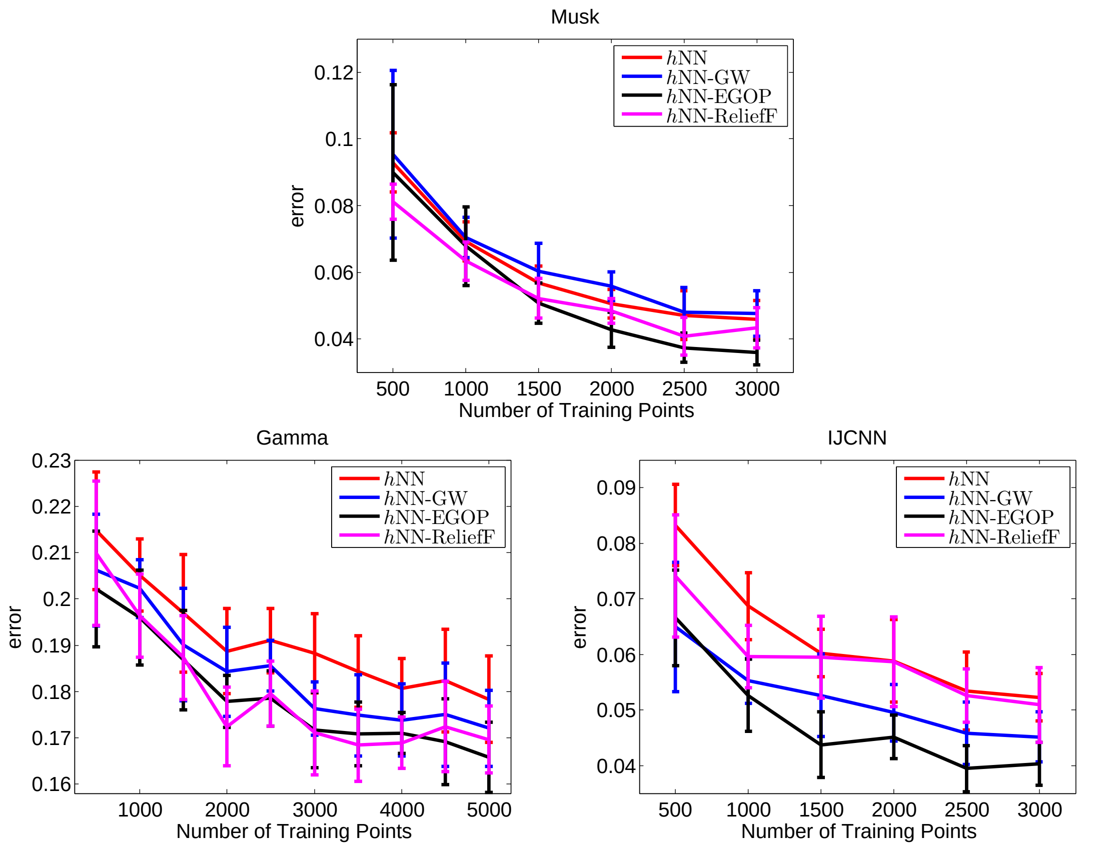


Figure 6.5: Classification error as a function of training set size for Musk, Gamma, IJCNN data sets.

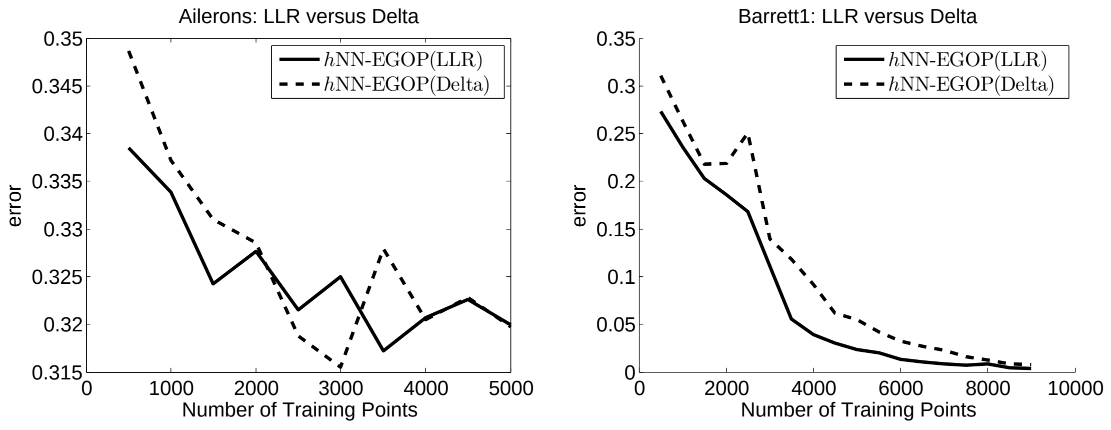


Figure 6.6: Comparison of EGOP estimated by our proposed method vs. locally linear regression, for Ailerons and Barrett1 datasets. See the text for more details including runtime

function of training set size; it is evident from these that while the performance of all methods tends to improve if additional training data are available, the gaps methods persist across the range of training set sizes.

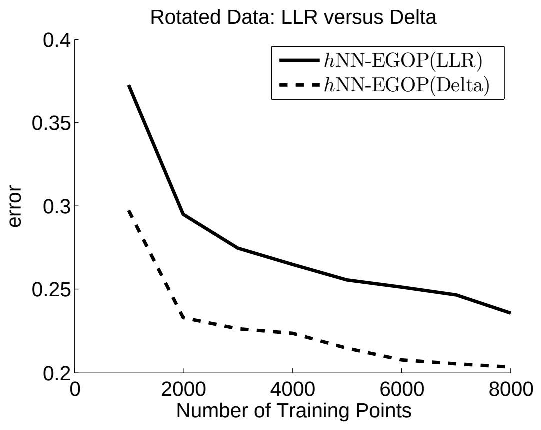


Figure 6.7: Comparison of EGOP estimated by our proposed method vs. locally linear regression for a synthetic dataset (with rotation). This synthetic data is similar to the one used in section 6.4.1 but with $d = 12$ and c = [5, 3, 1, .5, .2, .1, .08, .06, .05, .04, .03, .02].

From the results in Tables 6.1 and 6.2, we can see that the -EGOP variants dominate the -GW ones, and that both produce gains relative to using the original metric. This is true both for kNN and for kernel regression (hNN) methods, suggesting general utility of EGOP-based metric, not tied to a particular non-parametric mechanism. We also see that the metrics based on estimated EGOP are competitive with ReliefF.

6.4.5 *Experiments with Local Linear Regression*

As mentioned earlier in the paper, our estimator for EGOP is an alternative to an estimator based on computing the slope of locally linear regression (LLR) [21] over the training data. We have compared these two estimation methods on a number of data sets, and the results are plotted in Figure 6.6. In these experiments, the bandwidth of LLR was tuned by a 2-fold cross-validation on the training data.

We observe that despite its simplicity, the accuracy of predictors using EGOP-based metric estimated by our approach is competitive with or even better than the accuracy with EGOP estimated using LLR. As the sample size increases, accuracy of LLR improves. However, the computational expense of LLR-based estimator also grows with the size of data, and in our experiments it became dramatically slower than our estimator of EGOP for the larger data sizes. This confirms the intuition that our estimator is an appealing alternative to LLR-based estimator, offering a good tradeoff of speed and accuracy.

To impress upon the reader the computational advantage of our simple estimator over LLR, we also report the following running times (averaged over the ten random runs) for the same using our method and LLR respectively for the highest sample size used in the above real world datasets: Ailerons (128.13s for delta and 347.48s for LLR), Barrett (377.03s for delta and 1650.55s for LLR). Showing that our rough estimator is significantly faster than Local Linear Regression while giving competitive performance. These timings were recorded on an Intel i7 processor with CPU @ 2.40 GHz and 12 GB of RAM.

Table 6.1: Regression results, with ten random runs per data set.

Dataset	d	train/test	*h*NN	*h*NN-GW	*h*NN-EGOP
Ailerons	5	3000/2000	0.3637 ± 0.0099	0.3381 ± 0.0087	**0.3264** ± 0.0095
Concrete	8	730/300	0.3625 ± 0.0564	0.2525 ± 0.0417	**0.2518** ± 0.0418
Housing	13	306/200	0.3033 ± 0.0681	**0.2628** ± 0.0652	0.2776 ± 0.0550
Wine	11	2500/2000	0.7107 ± 0.0157	0.7056 ± 0.0184	**0.6867** ± 0.0145
Barrett1	21	3000/2000	0.0914 ± 0.0106	**0.0740** ± 0.0209	0.0927 ± 0.0322
Barrett5	21	3000/2000	0.0906 ± 0.0044	**0.0823** ± 0.0171	0.0996 ± 0.0403
Sarcos1	21	3000/2000	0.1433 ± 0.0087	**0.0913** ± 0.0054	0.1064 ± 0.0101
Sarcos5	21	3000/2000	0.1101 ± 0.0033	0.0972 ± 0.0044	**0.0970** ± 0.0064
ParkinsonM	19	3000/2000	0.4234 ± 0.0386	0.3606 ± 0.0524	**0.3546** ± 0.0406
ParkinsonT	19	3000/2000	0.4965 ± 0.0606	**0.3980** ± **0.0738**	0.4168 ± 0.0941
TeleComm	48	3000/2000	0.1079 ± 0.0099	0.0858 ± 0.0089	**0.0380** ± 0.0059

Dataset			*k*NN	*k*NN-GW	*k*NN-EGOP
Ailerons			0.3364 ± 0.0087	0.3161 ± 0.0058	**0.3154** ± 0.0100
Concrete			0.2884 ± 0.0311	**0.2040** ± 0.0234	0.2204 ± 0.0292
Housing			0.2897 ± 0.0632	**0.2389** ± 0.0604	0.2546 ± 0.0550
Wine			0.6633 ± 0.0119	0.6615 ± 0.0134	**0.6574** ± 0.0171
Barrett1			0.1051 ± 0.0150	**0.0843** ± 0.0229	0.1136 ± 0.0510
Barrett5			0.1095 ± 0.0096	**0.0984** ± 0.0244	0.1120 ± 0.0315
Sarcos1			0.1222 ± 0.0074	**0.0769** ± 0.0037	0.0890 ± 0.0072
Sarcos5			0.0870 ± 0.0051	0.0779 ± 0.0026	**0.0752** ± 0.0051
ParkinsonM			0.3638 ± 0.0443	**0.3181** ± 0.0477	0.3211 ± 0.0479
ParkinsonT			0.4055 ± 0.0413	0.3587 ± 0.0657	**0.3528** ± 0.0742
TeleComm			0.0864 ± 0.0094	0.0688 ± 0.0074	**0.0289** ± 0.0031

Table 6.2: Classification results with 3000 training/2000 testing.

Dataset	d	*h*NN	*h*NN-GW	*h*NN-EGOP	*h*NN-ReliefF
Cover Type	10	0.2301 ± 0.0104	0.2176 ± 0.0105	0.2197 ± 0.0077	**0.1806** ± 0.0165
Gamma	10	0.1784 ± 0.0093	0.1721 ± 0.0082	**0.1658** ± 0.0076	0.1696 ± 0.0072
Page Blocks	10	0.0410 ± 0.0042	0.0387 ± 0.0085	**0.0383** ± 0.0047	0.0395 ± 0.0053
Shuttle	9	0.0821 ± 0.0095	0.0297 ± 0.0327	**0.0123** ± 0.0041	0.1435 ± 0.0458
Musk	166	0.0458 ± 0.0057	0.0477 ± 0.0069	**0.0360** ± 0.0037	0.0434 ± 0.0061
IJCNN	22	0.0523 ± 0.0043	0.0452 ± 0.0045	**0.0401** ± 0.0039	0.0510 ± 0.0067
RNA	8	0.1128 ± 0.0038	0.0710 ± 0.0048	**0.0664** ± 0.0064	0.1343 ± 0.0406

Dataset		*k*NN	*k*NN-GW	*k*NN-EGOP	*k*NN-ReliefF
Cover Type		0.2279 ± 0.0091	0.2135 ± 0.0064	0.2161 ± 0.0061	**0.1839** ± 0.0087
Gamma		0.1775 ± 0.0070	0.1680 ± 0.0075	0.1644 ± 0.0099	**0.1623** ± 0.0063
Page Blocks		0.0349 ± 0.0042	0.0361 ± 0.0048	**0.0329** ± 0.0033	0.0347 ± 0.0038
Shuttle		0.0037 ± 0.0025	0.0024 ± 0.0016	**0.0021** ± 0.0011	0.0028 ± 0.0021
Musk		0.2279 ± 0.0091	0.2135 ± 0.0064	0.2161 ± 0.0061	**0.1839** ± 0.0087
IJCNN		0.0540 ± 0.0061	0.0459 ± 0.0058	**0.0413** ± 0.0051	0.0535 ± 0.0080
RNA		0.1042 ± 0.0063	0.0673 ± 0.0062	**0.0627** ± 0.0057	0.0828 ± 0.0056

7 THE EXPECTED JACOBIAN OUTER PRODUCT

Outline
The Expected Gradient Outer Product, which was the focal point of the previous chapter, is an interesting operator that emerges naturally from the theory of multi-index regression and *effective dimension reduction*. While it is straightforward to estimate while working with an unknown regression function $f : \mathbb{R}^d \to \mathbb{R}$, it is unclear how such an operator could be estimated when the unknown regression function is vector valued $f : \mathbb{R}^d \to \mathbb{R}^c$, while retaining the original multi-index motivation. In this chapter we give a generalization of the traditional EGOP for this case. We also show that a rough estimator for it remains statistically consistent under natural assumptions, while also providing gains in real world non-parametric classification tasks when used as a distance metric.

In the previous chapter, we worked with the following object, namely, the Expected Gradient Outer Product (EGOP):

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x})^\top \right)$$

Where f was an unknown regression function $f : \mathbb{R}^d \to \mathbb{R}$. The EGOP has the attractive property that it captures the average variation of f in all directions. As has been discussed earlier, in practice the function f might not vary equally along all coordinates: some features might be more important than the others, this being the motivation for variable selection methods as well as feature weighing methods such as [85], [86]. More generally, even if all the features have a bearing toward predicting the output $y \in \mathbb{R}$, there might exist an unknown k dimensional subspace on which y *effectively* depends upon. Such a *relevant* subspace can be recovered by doing a singular value decomposition of the EGOP. Even more generally, as might be the case frequently in practice, even a *relevant* subspace P might not exist. However, the EGOP is still useful as f is unlikely to vary equally in all directions: it can be employed to weight different directions according to their relevance. This was the motivation for using the EGOP as a metric in the previous chapter for non-parametric regression. Indeed, given the spectral decomposition $\mathbf{V}D\mathbf{V}^\top$ of the EGOP, we can transform the input $\mathbf{x}$ as $\mathbf{x} \mapsto \mathbf{D}^{1/2}\mathbf{V}^\top\mathbf{x}$. $\mathbf{V}$ rotates the data, while $\mathbf{D}^{1/2}$ weighs the coordinates. Using this transformation of the input was shown to improve regression performance on almost all datasets.

However, as was apparent, all the experimental results reported in Chapter 6 were for regression and binary classification. Around the time of the publication of [148], it was unclear if a similar metric could be estimated for the multiclass case. We noticed this to uniformly be the case in the use of the EGOP throughout the multi-index regression literature (for instance see the the experiments reported by [165], which also involve regression and binary classification only).

In this part of the dissertation, we generalize the EGOP such that it can also be estimated efficiently in the multi-class setting, and similarly be used to reweigh features in nonparametric multi-class classification tasks. Like in the case of the EGOP, we propose a rough estimator, which is cheap to estimate. We also prove that under similarly mild assumptions as for the EGOP, that it remains statistically consistent. We also provide experimental evidence that this generalization, which we call the expected Jacobian outer product (EJOP), can give significant improvements on classification error in real-world datasets, when used as the underlying metric in nonparametric classifiers.

Before we develop further on the EJOP, it might be instructive to first consider the EGOP for the case of binary classification. It might not be immediately obvious to the reader that the EGOP, which is well grounded for the case of nonparametric regression, carries through seamlessly for binary classification. We provide reasoning below that shows why this is the case, which also serves to motivate our approach to proposing an estimator for the EJOP.

7.1 EGOP AND BINARY CLASSIFICATION

To demonstrate that arguments used to motivate the EGOP in the case of nonparametric regression also work for binary classification, we first show that k-NN and ϵ-NN are plug-in classifiers. The same reasoning also works for other nonparametric regression methods, but we keep ourselves to nearest neighbors. For the sake of completeness, we begin by a standard definition.

Definition 2 (Bayes Classifier). *Suppose $\eta(x) = \mathbb{P}(Y = 1|X = \mathbf{x})$ and*

$$f(x) = \begin{cases} 1 & \text{if } \eta(x) > \frac{1}{2} \\ 0 & \text{if } \eta(x) \leq \frac{1}{2} \end{cases}$$

Then $f(x)$ is called the Bayes classifier.

Definition 3 (Plug-in Classifier). *Suppose $\hat{\eta}(x)$ is an estimate of $\eta(x)$ obtained from $\{(x_i, y_i)\}_{i=1}^N$, and*

$$\hat{f}(x) = \begin{cases} 1 & \text{if } \hat{\eta}(x) > \frac{1}{2} \\ 0 & \text{if } \hat{\eta}(x) \leq \frac{1}{2} \end{cases}$$

Then $\hat{f}(x)$ is called a plug-in classifier.

Consider $\hat{\eta}(x) = \sum_{i=1}^N w_i \mathbb{1}[y_i = 1]$ with $\sum_{i=1}^N w_i = 1$. If C is the set of selected neighbors. Then, in the case of k-NN:

$$w_i(x) = \begin{cases} \frac{1}{k} & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases}$$

Likewise, in the case of ϵ-NN

$$w_i(x) = \begin{cases} \frac{1}{|\mathcal{B}(x,\epsilon)|} & \text{if } i \in \mathcal{B}(x,\epsilon) \\ 0 & \text{otherwise} \end{cases}$$

Proposition 2. *k-NN and ϵ-NN are plug-in classifiers*

Proof. The plug-in classifier might be rewritten as:

$$\begin{aligned}\hat{f}(x) &= \mathbb{1}\left(\hat{\eta}(x) > \frac{1}{2}\right)u \\ &= \mathbb{1}\left(\sum_{i=1}^{N} w_i \mathbb{1}[y_1 = 1] > \frac{1}{2}\right) \\ &= \mathbb{1}\left(\sum_{i=1}^{N} w_i (2\mathbb{1}[y_1 = 1] - 1) > 0\right) \\ &= \mathbb{1}\left(\sum_{i=1}^{N} w_i (\mathbb{1}[y_1 = 1] - \mathbb{1}[y_1 = 0]) > 0\right) \\ &= \mathbb{1}\left(\sum_{i=1}^{N} w_i \mathbb{1}[y_1 = 1] > \sum_{i=1}^{N} w_i \mathbb{1}[y_1 = 0]\right)\end{aligned}$$

□

While laying out this trivial reasoning might seem unnecessarily excessive, the main message that we want to impress upon the reader is that k-NN and ϵ-NN are plug-in classifiers. Note that, $\hat{\eta}(x)$, which is the plug-in classifier is a regression estimate of the Bayes classifier $\eta(x)$, and it is well known that the 0-1 classification error of the plug-in methods is related to the regression estimate (see Devroye *et al.* [33]). Thus the reasoning used for non-parametric regression in the case of gradient weights[85], [86], EGOP [148], [165] etc., also carries to the case of binary classification; one can just use $\hat{\eta}$ to find the derivatives.

7.2 THE MULTICLASS CASE

For the multi-class case, we can consider

$$\hat{\eta}_k(x) = \sum_{i=1}^{N} w_i \mathbb{1}[y_i = k]; \text{ where } \sum_{i=1}^{N} w_i = 1 \text{ and } k \in \{1, \dots, r\}$$

and define $\hat{f}(x) = \max_k \hat{\eta}_k(x)$.

We can consider using $\hat{f}(x)$ for finding derivatives, but that is prevented by the appearance of the max. Alternatively, we could consider a vector valued function (c being the number of classes)

$$\tilde{f}(x) = [\hat{\eta}_1, \dots, \hat{\eta}_c]$$

and then use the differences to find the derivatives. The latter approach seems in direct analogy to the case of binary classification, thus we use it to define the Jacobian Outer Product. Note that the properties of this plug-in and whether it is similar to the standard plug-in defined above is beyond the scope of this chapter (see Devroye *et al.* [33]). We simply content ourselves with using it to define the EJOP, and this intuition is borne out by being able to use it to prove a consistency result akin to the EGOP. We now conclude these meanderings to better motivate the EJOP, and proceed to define it more formally in the following section.

7.3 THE EXPECTED JACOBIAN OUTERPRODUCT

Recall that in high dimensional classification problems over $\mathbb{R}^d$, the unknown (multinomial regression) function f could be considered to be a vector-valued function mapping to a probability simplex $\mathbb{S}^c = \{\mathbf{y} \in \mathbb{R}^c | \forall i \; y_i \geq 0, \mathbf{y}^T\mathbf{1} = 1\}$, where c is the number of classes. Or, more concisely, $f : \mathbb{R}^d \to \mathbb{S}^c$. The prediction for some point $\mathbf{x}$ is then given by: $y = \text{argmax}_{i=1,\ldots,c} f_i(\mathbf{x})$.

For f, at point $\mathbf{x}$, we can define the Jacobian as:

$$\mathbf{J}_f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_c(\mathbf{x})}{\partial x_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1(\mathbf{x})}{\partial x_d} & \frac{\partial f_2(\mathbf{x})}{\partial x_d} & \cdots & \frac{\partial f_c(\mathbf{x})}{\partial x_d} \end{bmatrix}$$

We are interested in the quantity

$$\mathbb{E}_X G(X) = \mathbf{J}_f(\mathbf{x})\mathbf{J}_f(\mathbf{x})^T$$

Let f_n be an initial estimate of f, for which we use a kernel estimate, then for the $(i,j)^{th}$ element of $\mathbf{J}_f(\mathbf{x})$, we can use the following rough estimate:

$$\Delta_{t,i,j} f_n(\mathbf{x}) = \frac{f_{n,i}(\mathbf{x} + t\mathbf{e}_j) - f_{n,i}(\mathbf{x} - t\mathbf{e}_j)}{2t}, t > 0$$

Let $\mathbf{J}_n(\mathbf{x})$ be the Jacobian estimate at $\mathbf{x}$. The Jacobian outer product is then estimated as $\mathbb{E}_n \mathbf{J}_n(\mathbf{x})\mathbf{J}_n(\mathbf{x})^T$, which is the empirical average of $\mathbf{J}_n(\mathbf{x})\mathbf{J}_n(\mathbf{x})^T$.

Note: Many of the assumptions and notation used overlap with that employed in chapter 6. We introduce new notation as needed, and if occasionally dictated for ease of exposition, redefine some term already defined in chapter 6.

7.3.1 *Function Estimate*

First, we need to specify the function estimate, that is used both for the theoretical analysis, and the experiments reported.

Again, considering c to be the number of classes, let the c dimensional vector valued function estimate be denoted by $\bar{f}_{n,h}(\mathbf{x})$, such that $\bar{f}_{n,h}(\mathbf{x}) \in \mathbb{S}^c$

$$\bar{f}_{n,h}(\mathbf{x}) = [\bar{f}_{n,h,1}(\mathbf{x}), \ldots, \bar{f}_{n,h,c}(\mathbf{x})] \text{ s.t. } \bar{f}_{n,h,i}(\mathbf{x}) > 0 \text{ and } \sum_i \bar{f}_{n,h,i}(\mathbf{x}) = 1$$

The prediction in that case is given by:

$$\hat{y} = \arg\max_i \bar{f}_{n,h,i}(\mathbf{x})$$

.

We use the following kernel estimate: $\bar{f}_{n,h,c}(\mathbf{x}) = \sum_i w_i \mathbb{1}\{Y_i = c\}$, where:

$$w_i(\mathbf{x}) = \frac{K(\|\mathbf{x} - \mathbf{x}_i\|/h)}{\sum_j K(\|\mathbf{x} - \mathbf{x}_j\|/h)} \text{ if } B(\mathbf{x}, h) \cap \mathbf{x} \neq \phi,$$

$$w_i(\mathbf{x}) = \frac{1}{n} \text{ otherwise}$$

Note that for k-NN, $w_i(\mathbf{x}) = \frac{1}{k}$ and for ϵ-NN, $w_i(\mathbf{x}) = \frac{1}{|B(\mathbf{x},h)|}$

While estimating gradients, we actually work with the softmaxed output

$$f_{n,h,i}(\mathbf{x}) = \frac{\exp(\bar{f}_{n,h,i}(\mathbf{x}))}{\sum_j \exp(\bar{f}_{n,h,j}(\mathbf{x}))}$$

Additionally, in the experiments we use a temperature term in the softmax for affording ease in gradient computation. But we omit this aspect from the discussion to keep the theoretical analysis simple, in any case, appearance of the temperature term does not affect any of the discussion to follow.

7.3.2 *Note on the nomenclature*

A natural question to ask is regarding the use of the name: Expected Jacobian Outer Product (EJOP), as compared to simply the Expected Gradient Outer Product (EGOP), after all we still find gradients, even in the multiclass case. We simply use different terminology to distinguish the two cases, especially given the lack of work on *effective dimension reduction* and multi-index regression for multinomial regression.

7.4 NOTATION AND SETUP

For a vector $x \in \mathbb{R}^d$, we denote the euclidean norm as $\|x\|$. For a matrix, we denote the spectral norm, which is the largest singular value of the matrix $\sigma_{\max}(A)$ as $\|A\|_2$. The column space of a matrix $A \in R^{n\times m}$ is denoted as $\text{im}(A)$ where $\text{im}(A) = \{\mathbf{Y} \in \mathbb{R}^n | \mathbf{Y} = A\mathbf{x} \text{ for some } \mathbf{x} \in \mathbb{R}^m\}$, and $\ker(A)$ is used to denote the null space of matrix $A \in R^{n\times m}$: $\ker(A) = \{\mathbf{x} \in \mathbb{R}^m | A\mathbf{x} = 0\}$. We use $A \circ B$ to denote the Hadamard product of matrices A and B.

Let the estimated nonparametric function be $f_{n,h,c}(\mathbf{x}) = \sum_i \omega_i(\mathbf{x})\mathbb{1}\{y_i = c\}$, and $\tilde{f}_{n,h,c}(\mathbf{x}) = \sum_i \omega_i(\mathbf{x})\mathbb{P}(y_i = c|x_i)$. Our estimated gradient at dimension i is given as

$$\Delta_{t,i} f_{n,h,c}(\mathbf{x}) = \frac{f_{n,h,c}(\mathbf{x} + te_i) - f_{n,h,c}(\mathbf{x} - te_i)}{2t},$$

and the estimated and true gradients for class c are given as:

$$\hat{\nabla} f_{n,h,c}(\mathbf{x}) = \begin{bmatrix} \Delta_{t,1} f_{n,h,c}(\mathbf{x}) \cdot \mathbb{1}_{A_{n,1}(\mathbf{x})} \\ \Delta_{t,2} f_{n,h,c}(\mathbf{x}) \cdot \mathbb{1}_{A_{n,2}(\mathbf{x})} \\ \dots \\ \Delta_{t,d} f_{n,h,c}(\mathbf{x}) \cdot \mathbb{1}_{A_{n,d}(\mathbf{x})} \end{bmatrix}, \hat{\nabla} f_c(\mathbf{x}) = \begin{bmatrix} \Delta_{t,1} f_c(\mathbf{x}) \cdot \mathbb{1}_{A_{n,1}(\mathbf{x})} \\ \Delta_{t,2} f_c(\mathbf{x}) \cdot \mathbb{1}_{A_{n,2}(\mathbf{x})} \\ \dots \\ \Delta_{t,d} f_c(\mathbf{x}) \cdot \mathbb{1}_{A_{n,d}(\mathbf{x})} \end{bmatrix}$$

Where $A_{n,i}(X)$ is the event that enough samples contribute to the estimate $\Delta_{t,i} f_{n,h}(X)$:

$$A_{n,i}(X) \equiv \min_{\{t,-t\}} \mu_n(B(X + se_i, h/2)) \geq \frac{2d \ln 2n + ln(4/\delta)}{n}$$

and likewise

$$A_i(X) \equiv \min_{\{t,-t\}} \mu(B(X + se_i, h/2)) \geq 3 \cdot \frac{2d \ln 2n + ln(4/\delta)}{n}$$

note that μ_n, μ are empirical mass and mass of a ball, respectively.

We denote indicators of the events $A_{n,i}(X)$ and $A_i(X)$ in the following form

$$\mathbb{I}_n(x) = \begin{bmatrix} \mathbb{1}_{A_{n,1}(\mathbf{x})} \\ \mathbb{1}_{A_{n,2}(\mathbf{x})} \\ \dots \\ \mathbb{1}_{A_{n,d}(\mathbf{x})} \end{bmatrix}, \overline{\mathbb{I}_n(x)} = \begin{bmatrix} \mathbb{1}_{\bar{A}_{n,1}(\mathbf{x})} \\ \mathbb{1}_{\bar{A}_{n,2}(\mathbf{x})} \\ \dots \\ \mathbb{1}_{\bar{A}_{n,d}(\mathbf{x})} \end{bmatrix}, \mathbb{I}(x) = \begin{bmatrix} \mathbb{1}_{A_1(\mathbf{x})} \\ \mathbb{1}_{A_2(\mathbf{x})} \\ \dots \\ \mathbb{1}_{A_d(\mathbf{x})} \end{bmatrix}, \overline{\mathbb{I}(x)} = \begin{bmatrix} \mathbb{1}_{\bar{A}_1(\mathbf{x})} \\ \mathbb{1}_{\bar{A}_2(\mathbf{x})} \\ \dots \\ \mathbb{1}_{\bar{A}_d(\mathbf{x})} \end{bmatrix}.$$

Let the Jacobian matrix $\mathbf{J}_f(\mathbf{x}) \in \mathbb{R}^{d\times c}$ to be

$$\mathbf{J}_f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_c(\mathbf{x})}{\partial x_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1(\mathbf{x})}{\partial x_d} & \frac{\partial f_2(\mathbf{x})}{\partial x_d} & \cdots & \frac{\partial f_c(\mathbf{x})}{\partial x_d} \end{bmatrix}$$

where c is the number of classes. And $f_k(x) = \mathbb{P}(y = k|x), \forall k \in [c]$ represents the conditional distribution of the class labels.

Then the Jacobian outer product matrix $G(\mathbf{x})$ is $G(\mathbf{x}) = \mathbf{J}_f(\mathbf{x})\mathbf{J}_f(\mathbf{x})^T$.The estimated Jacobian matrix is

$$\hat{\mathbf{J}}_f(\mathbf{x}) = \begin{bmatrix} \hat{\nabla} f_{n,h,1}(\mathbf{x}) & \hat{\nabla} f_{n,h,2}(\mathbf{x}) & \dots & \hat{\nabla} f_{n,h,k}(\mathbf{x}) \end{bmatrix}$$

the estimated Jacobian product matrix is denoted $\hat{G}(\mathbf{x}) = \hat{\mathbf{J}}_f(\mathbf{x})\hat{\mathbf{J}}_f(\mathbf{x})^T$.

7.4.1 *Assumptions*

The assumptions are the same as in 6.2.2, with the following modifications: first to the bounded gradient assumption such that it extends to each class.

- Bounded Gradient: $\|\nabla f_k(\mathbf{x})\|_2 \leq R, \forall \mathbf{x} \in \mathcal{X}, k \in [c]$.

Secondly, we modify the assumption on the modulus of continuity of ∇f_k similarly Let $\epsilon_{t,k,i} = \sup_{\mathbf{x}\in\mathcal{X}, s\in[-t,t]} \left| \frac{\partial f_k(\mathbf{x})}{\partial x_i} - \frac{\partial f_k(\mathbf{x}+se_i)}{\partial x_i} \right|$ and $\epsilon_{t,i} = \max_k \epsilon_{t,c,i}$, define the (t,i)-boundary of $\mathcal{X}$ as $\partial_{t,i}(\mathcal{X}) = \{\mathbf{x} : \{\mathbf{x} + te_i, x - te_i\} \nsubseteq \mathcal{X}\}$. When μ has continues density on $\mathcal{X}$ and ∇f_k is uniformly continuous on $\mathcal{X} + B(0,\tau)$, we have $\mu(\partial_{t,i}(\mathcal{X})) \xrightarrow{t\to 0} 0$ and $\epsilon_{t,k,i} \xrightarrow{t\to 0} 0$.

7.5 CONSISTENCY OF ESTIMATOR $\mathbb{E}_n\hat{G}(X)$ OF THE JACOBIAN OUTERPRODUCT $\mathbb{E}_X G(X)$

To show that the estimator $\mathbb{E}_n\hat{G}(X)$ is consistent, we proceed to bound $\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_X G(X)\|$ for finite n, which is encapsulated in the theorem that follows. There are two main difficulties in the proof, which are addressed by a sequence of lemmas. One has to do with the fact that the gradient estimate at any point depends on all other points, and second, having gradient estimates for c classes.

MAIN RESULT

Theorem 2. *Let $t+h \leq \tau$, and let $0 \leq \delta \leq 1$. There exist $C = C(\mu, K(\cdot))$ and $N = N(\mu)$ such that the following holds with probability at least $1-2\delta$. Define $A(n) = \sqrt{Cd \cdot \log(kn/\delta)} \cdot 0.25/\log^2(n/\delta)$. Let $n \geq N$, we have:*

$$\|\mathbb{E}_n\hat{G}(X)] - \mathbb{E}_X G(X)\|_2 \leq \frac{6R^2}{\sqrt{n}}\left(\sqrt{\ln d} + \sqrt{\ln\frac{1}{\delta}}\right) + k\left(3R + \sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2} + \sqrt{d}\left(\frac{hR+1}{t}\right)\right)$$
$$\left[\frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d} + h^2R^2} + R\left(\sqrt{\frac{d\ln\frac{d}{\delta}}{2n}} + \sqrt{\sum_{i\in[d]} \mu^2(\partial_{t,i}(\mathcal{X}))}\right) + \sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2}\right]$$

Proof. We begin with the following decomposition:

$$\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_X G(X)\|_2 \leq \|\mathbb{E}_n G(X) - \mathbb{E}_X G(X)\|_2 + \|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_n G(X)\|_2$$

The first term on the right hand side i.e. $\|\mathbb{E}_n G(X) - \mathbb{E}_X G(X)\|_2$ is bounded using Lemma 14; by using Lemma 15 we bound the second term $\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_n G(X)\|_2$, this is done with respect to $\sum_{k\in[c]} \mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2$; therefore we need to bound $\sum_{k\in[c]} \mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2$, which is done by employing Theorem 3 which concludes the proof. □

Remark. The theorem implies consistency for $t \xrightarrow{n\to\infty} 0$, $h \xrightarrow{n\to\infty} 0$, $h/t^2 \xrightarrow{n\to\infty} 0$, and $(n/\log n)h^d t^4 \xrightarrow{n\to\infty} \infty$, this is satisfied for many settings, for example $t \propto h^{1/4}$, $h \propto \frac{1}{\ln n}$.

7.5.1 *Bounding* $\|\mathbb{E}_n G(X) - \mathbb{E}_X G(X)\|_2$

To bound this term, like in the case of the EGOP, we use the following random matrix concentration result.

Lemma 13. *[68, 149]. For the random matrix $\mathbf{X} \in \mathbb{R}^{d_1\times d_2}$ with bounded spectral norm $\|\mathbf{X}\|_2 \leq M$, let $d = \min\{d_1, d_2\}$, and $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n$ are i.i.d. samples, with probability at least $1-\delta$, we have*

$$\left\|\frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i - \mathbb{E}\mathbf{X}\right\|_2 \leq \frac{6M}{\sqrt{n}}\left(\sqrt{\ln d} + \sqrt{\ln\frac{1}{\delta}}\right)$$

Recall the bounded gradient assumption $\|G(X)\|_2 = \|\nabla f(X)\|_2^2 \leq R^2$. Using this assumption we can apply the above lemma to i.i.d matrices $G(X), X \in \mathbf{X}$, yielding the following lemma.

Lemma 14. *With probability at least $1-\delta$*

$$\|\mathbb{E}_n G(X) - \mathbb{E}_X G(X)\|_2 \leq \frac{6R^2}{\sqrt{n}}\left(\sqrt{\ln d} + \sqrt{\ln\frac{1}{\delta}}\right)$$

Next we proceed to bound the second term in the decomposition mentioned in the proof of theorem 2.

7.5.2 *Bounding* $\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_n G(X)\|_2$

A first bound is provided by the following lemma:

Lemma 15. *Exist constant c, with probability at least* $1-\delta$*:*

$$\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_n G(X)\|_2 \leq \sum_{k\in[c]} \mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \cdot \max_{x\in\mathbf{X}} \|\nabla f_k(X) + \hat{\nabla} f_{n,h,k}(X)\|_2$$

Proof. First we can write the term on the l.h.s in terms of the gradients for each class:

$$\begin{aligned}\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_n G(X)\|_2 =& \|\mathbb{E}_n[\hat{G}(X) - G(X)]\|_2 \\ =& \left\|\sum_{k\in[c]} \mathbb{E}_n[\nabla f_k(X)\cdot\nabla f_k(X)^T - \hat{\nabla} f_{n,h,k}(X)\cdot\hat{\nabla} f_{n,h,k}(X)^T]\right\|_2 \\ \leq& \sum_{k\in[c]} \left\|\mathbb{E}_n[\nabla f_k(X)\cdot\nabla f_k(X)^T - \hat{\nabla} f_{n,h,k}(X)\cdot\hat{\nabla} f_{n,h,k}(X)^T]\right\|_2\end{aligned}$$

next, we notice that $\nabla f_k(\mathbf{x})\cdot\nabla f_k(\mathbf{x})^T - \hat{\nabla} f_{n,h,k}(\mathbf{x})\cdot\hat{\nabla} f_{n,h,k}(\mathbf{x})^T$ can be rewritten as:

$$\begin{aligned}\nabla f_k(\mathbf{x})\cdot\nabla f_k(\mathbf{x})^T - \hat{\nabla} f_{n,h,k}(\mathbf{x})\cdot\hat{\nabla} f_{n,h,k}(\mathbf{x})^T =& \frac{1}{2}\cdot(\nabla f_k(\mathbf{x}) + \hat{\nabla} f_{n,h,k}(\mathbf{x}))\cdot(\nabla f_k(\mathbf{x}) - \hat{\nabla} f_{n,h,k}(\mathbf{x}))^T \\ &+ \frac{1}{2}\cdot(\nabla f_k(\mathbf{x}) - \hat{\nabla} f_{n,h,k}(\mathbf{x}))\cdot(\nabla f_k(\mathbf{x}) + \hat{\nabla} f_{n,h,k}(\mathbf{x}))^T\end{aligned}$$

Using this yields:

$$\begin{aligned}\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_n G(X)\|_2 \leq& \frac{1}{2}\sum_{k\in[c]} \|\mathbb{E}_n[(\nabla f_k(X) + \hat{\nabla} f_{n,h,k}(X))\cdot(\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X))^T]\|_2 \\ &+ \frac{1}{2}\sum_{k\in[c]} \|\mathbb{E}_n[(\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X))\cdot(\nabla f_k(X) + \hat{\nabla} f_{n,h,k}(X))^T]\|_2 \\ =& \sum_{k\in[c]} \|\mathbb{E}_n[(\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X))\cdot(\nabla f_k(X) + \hat{\nabla} f_{n,h,k}(X))^T]\|_2\end{aligned}$$

By using Jensen's inequality, we have:

$$\begin{aligned}&\mathbb{E}_n[(\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X))\cdot(\nabla f_c k(X) + \hat{\nabla} f_{n,h,k}(X))^T]\|_2 \leq \\ &\qquad \mathbb{E}_n\|(\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X))\cdot(\nabla f_k(X) + \hat{\nabla} f_{n,h,k}(X))^T\|_2\end{aligned}$$

combining the above, gives us the following bound on $\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_n G(X)\|_2$

$$\begin{aligned}\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_n G(X)\|_2 \leq& \sum_{k\in[c]} \mathbb{E}_n\|(\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X))\cdot(\nabla f_k(X) + \hat{\nabla} f_{n,h,k}(X))^T\|_2 \\ =& \sum_{k\in[c]} \mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \cdot \|\nabla f_k(X) + \hat{\nabla} f_{n,h,k}(X)\|_2. \\ \leq& \sum_{k\in[c]} \mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \cdot \max_{X\in\mathbf{X}} \|\nabla f_k(X) + \hat{\nabla} f_{n,h,k}(X)\|_2.\end{aligned}$$

□

The above bound has a dependence on $\|\nabla f_k(X)+\hat{\nabla} f_{n,h,k}(X)\|_2$, which we now proceed to bound below:

7.5.3 *Bounding* $\|\nabla f_k(X)+\hat{\nabla} f_{n,h,k}(X)\|_2$

We first bound the max term, by the following lemma:

Lemma 16. *$\forall c \in [k]$, we have*

$$\max_{X\in\mathbf{X}} \|\nabla f_k(X)+\hat{\nabla} f_{n,h,k}(X)\|_2 \le 3R + \sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2} + \sqrt{d}(\frac{hR+1}{t})$$

Proof. $\forall x \in \mathbf{X}$, we have

$$\begin{aligned}\|\nabla f_k(\mathbf{x})+\hat{\nabla} f_{n,h,k}(\mathbf{x})\|_2 &\le \|\nabla f_k(\mathbf{x})\|_2 + \|\hat{\nabla} f_{n,h,k}(\mathbf{x})\|_2\\ &\le 2\|\nabla f_k(\mathbf{x})\|_2 + \|\nabla f_k(\mathbf{x})-\hat{\nabla} f_{n,h,k}(\mathbf{x})\|_2\\ &\le 2R + \|\nabla f_k(\mathbf{x})-\hat{\nabla} f_k(\mathbf{x})\|_2 + \|\hat{\nabla} f_k(\mathbf{x})-\hat{\nabla} f_{n,h,k}(\mathbf{x})\|_2\end{aligned}$$

Next, we adopt the steps as in the proof for Lemma 21, and get the following bound:

$$\|\hat{\nabla} f_k(\mathbf{x})-\hat{\nabla} f_{n,h,k}(\mathbf{x})\|_2 \le \sqrt{\sum_{i\in[d]} (|\Delta_{t,i} f_{n,h,k}(\mathbf{x})-\Delta_{t,i} f_k(\mathbf{x})|\cdot \mathbb{1}_{A_{n,i}(\mathbf{x})})^2},$$

this is because

$$\begin{aligned}|\Delta_{t,i} f_{n,h,k}(\mathbf{x})-\Delta_{t,i} f_k(\mathbf{x})|\cdot \mathbb{1}_{A_{n,i}(\mathbf{x})} \le& \frac{1}{t}\max_{s\in\{-t,t\}} |\tilde{f}_{n,h,k}(\mathbf{x}+se_i)-f_k(\mathbf{x}+se_i)|\cdot \mathbb{1}_{A_{n,i}(\mathbf{x})}\\ &+\frac{1}{t}\max_{s\in\{-t,t\}} |\tilde{f}_{n,h,k}(\mathbf{x}+se_i)-f_{n,h,k}(\mathbf{x}+se_i)|\cdot \mathbb{1}_{A_{n,i}(\mathbf{x})},\end{aligned}$$

we also know that

$$\max_{s\in\{-t,t\}} |\tilde{f}_{n,h,k}(X+se_i)-f_{n,h,k}(X+se_i)| \le 1.$$

Thus we obtain the following bound:

$$\|\hat{\nabla} f_k(X)-\hat{\nabla} f_{n,h,k}(X)\|_2 \le \sqrt{d}(\frac{hR+1}{t})$$

While, we also have that

$$\begin{aligned}\|\nabla f_k(X)-\hat{\nabla} f_k(X)\|_2 \le& \|\nabla f_k(X)\circ \mathbf{I}_n(X)-\hat{\nabla} f_k(X)\|_2 + \|\nabla f_k(X)\circ \overline{\mathbf{I}_n(X)}\|_2\\ \le& R+\sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2}\end{aligned}$$

Combining the above completes the proof □

Next we need to bound $\mathbb{E}_n\|\nabla f_k(X)-\hat{\nabla} f_{n,h,k}(X)\|_2$, which we do so in the next subsection:

7.5.4 *Bound on* $\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2$

We first decompose $\mathbb{E}_n\|\nabla f_c(X) - \hat{\nabla} f_{n,h,k}(X)\|_2$ as:

$$\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \leq \\ \mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_k(X)\|_2 + \mathbb{E}_n\|\hat{\nabla} f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2$$

the first term in the r.h.s of the above i.e. $\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_k(X)\|_2$ can in turn be decomposed as:

$$\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_k(X)\|_2 \leq \\ \mathbb{E}_n\|\nabla f_k(X) \circ \mathbb{I}_n(X) - \hat{\nabla} f_k(X)\|_2 + \mathbb{E}_n\|\nabla f_k(X) \circ \overline{\mathbb{I}_n(X)}\|_2$$

We need to bound both terms that appear on the r.h.s of the above, which we do so in the next two subsections, starting with the second term.

7.5.4.1 *Bounding* $\mathbb{E}_n\|\nabla f_k(X) \circ \overline{\mathbb{I}_n(X)}\|_2$

The bound is encapsulated in the following lemma:

Lemma 17. *With probability at least $1-\delta$ over the choice of X:*

$$\mathbb{E}_n\|\nabla f_k(X) \circ \overline{\mathbb{I}_n(X)}\|_2 \leq R\left(\sqrt{\frac{d\ln\frac{d}{\delta}}{2n}} + \sqrt{\sum_{i\in[d]} \mu^2(\partial_{t,i}(\mathcal{X}))}\right)$$

Proof. We begin by recalling the bounded gradient assumption: $\|\nabla f(\mathbf{x})\|_2 \leq R$, using which we get

$$\mathbb{E}_n\|\nabla f(X) \circ \overline{\mathbb{I}_n(X)}\|_2 \leq R\mathbb{E}_n\|\overline{\mathbb{I}_n(X)}\|_2$$

By relative VC bounds [151], if we set $\alpha_n = \frac{2d\ln 2n + \ln(4/\delta)}{n}$, then with probability at least $1-\delta$ over the choice of X, for all balls $B \in R^d$ we have $\mu(B) \leq \mu_n(B) + \sqrt{\mu_n(B)\alpha_n} + \alpha_n$. Thus, with probability at least $1-\delta$, $\forall i \in [d]$, $\bar{A}_{n,i}(X) \Rightarrow \bar{A}_i(X)$. Moreover, since $\|\overline{\mathbb{I}(X)}\|_2 \leq \sqrt{d}$, then by Hoeffding's inequality,

$$\mathbb{P}(\mathbb{E}_n\|\overline{\mathbb{I}(X)}\|_2 - \mathbb{E}_X\|\overline{\mathbb{I}(X)}\|_2 \geq \epsilon) \leq e^{-\frac{2n\epsilon^2}{d}}$$

applying the union bound, we have the following with probability at least $1-\delta$

$$\mathbb{E}_n\|\overline{\mathbb{I}_n(X)}\|_2 \leq \mathbb{E}_n\|\overline{\mathbb{I}(X)}\|_2 \leq \mathbb{E}_X\|\overline{\mathbf{I}_n(X)}\|_2 + \sqrt{\frac{d\ln\frac{d}{\delta}}{2n}}$$

But note that we have:

$$\mathbb{E}_X\mathbb{1}_{\bar{A}_i(X)} \leq \mathbb{E}_X[\mathbb{1}_{\bar{A}_i(X)}|X \in \mathcal{X}\backslash\partial_{t,i}(\mathcal{X})] + \mu(\partial_{t,i}(\mathcal{X}))$$

to see why this is true observe that $\mathbb{E}_X[\mathbf{1}_{\bar{A}_i(X)}|X \in \mathcal{X}\backslash\partial_{t,i}(\mathcal{X})] = 0$ because $\mu(B(x+se_i, h/2)) \geq C_\mu(h/2)^d \geq 3\alpha$ when we set $h \geq (\log^2(n/\delta)/n)^{1/d}$.

So, we have:

$$\mathbb{E}_X\|\overline{\mathbb{I}_n(X)}\|_2 \leq \sqrt{\sum_{i\in[d]} \mu^2(\partial_{t,i}(\mathcal{X}))}$$

Thus with probability at least $1-\delta$, we obtain the following:

$$\mathbb{E}_n\|\nabla f_k(X) \circ \overline{\mathbf{I}_n(X)}\|_2 \leq R\left(\sqrt{\frac{d\ln\frac{d}{\delta}}{2n}} + \sqrt{\sum_{i\in[d]} \mu^2(\partial_{t,i}(\mathcal{X}))}\right)$$

□

Next we need to bound the first term that appeared on the r.h.s. of the decomposition of $\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_k(X)\|_2$, reproduced below for ease of exposition:

$$\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_k(X)\|_2 \leq \\ \mathbb{E}_n\|\nabla f_k(X) \circ \mathbb{I}_n(X) - \hat{\nabla} f_k(X)\|_2 + \mathbb{E}_n\|\nabla f_k(X) \circ \overline{\mathbb{I}_n(X)}\|_2$$

7.5.4.2 *Bounding* $\mathbb{E}_n\|\nabla f_k(X) \circ \mathbb{I}_n(X) - \hat{\nabla} f_k(X)\|_2$

This bound is encapsulated in the following lemma

Lemma 18. *We have*

$$\mathbb{E}_n\|\nabla f_k(X) \circ \mathbb{I}_n(X) - \hat{\nabla} f_k(X)\|_2 \leq \sqrt{\sum_{i\in[d]} \epsilon_{t,c,i}^2}$$

Proof. We start with the simple observation regarding the envelope:

$$f_k(\mathbf{x}+te_i) - f_k(\mathbf{x}-te_i) = \int_{-t}^{t} \frac{\partial f_k(\mathbf{x}+se_i)}{\partial x_i} ds$$

using this we have

$$2t\left(\frac{\partial f_k'(\mathbf{x})}{\partial x_i} - \epsilon_{t,k,i}\right) \leq f_k(\mathbf{x}+te_i) - f_k(\mathbf{x}-te_i) \leq 2t\left(\frac{\partial f_k'(\mathbf{x})}{\partial x_i} + \epsilon_{t,k,i}\right)$$

Thus we have

$$\left|\frac{1}{2t}(f_c(\mathbf{x}+te_i) - f_c(\mathbf{x}-te_i)) - \frac{\partial f_c'(\mathbf{x})}{\partial x_i}\right| \leq \epsilon_{t,c,i}$$

using which we have the following

$$\begin{aligned}
\|\nabla f_k(\mathbf{x}) \circ \mathbb{I}_n(x) - \hat{\nabla} f_k(\mathbf{x})\|_2 &= \sqrt{\sum_{i=1}^{d}\left|\frac{\partial f_k'(\mathbf{x})}{\partial x_i} \cdot \mathbb{1}_{A_{n,i}(\mathbf{x})} - \Delta_{t,i} f_k(\mathbf{x}) \cdot \mathbb{1}_{A_{n,i}(\mathbf{x})}\right|^2} \\
&\leq \sqrt{\sum_{i=1}^{d}\left|\frac{1}{2t}(f_k(\mathbf{x}+te_i) - f_k(\mathbf{x}-te_i)) - \frac{\partial f_k'(\mathbf{x})}{\partial x_i}\right|^2} \\
&\leq \sqrt{\sum_{i\in[d]} \epsilon_{t,k,i}^2}
\end{aligned}$$

Taking empirical expectation on both sides finishes the proof. □

Taking a step back, recall again the decomposition of $\mathbb{E}_n\|\nabla f_c(X) - \hat{\nabla} f_{n,h,k}(X)\|_2$:

$$\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \leq \\ \mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_k(X)\|_2 + \mathbb{E}_n\|\hat{\nabla} f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2$$

the first term in the r.h.s of the above i.e. $\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_k(X)\|_2$ was in turn decomposed as:

$$\mathbb{E}_n\|\nabla f_k(X) - \hat{\nabla} f_k(X)\|_2 \leq \\ \mathbb{E}_n\|\nabla f_k(X) \circ \mathbb{I}_n(X) - \hat{\nabla} f_k(X)\|_2 + \mathbb{E}_n\|\nabla f_k(X) \circ \overline{\mathbb{I}_n(X)}\|_2$$

The analysis in the previous subsection was bounding these two terms individually. Now we turn our attention towards bounding $\mathbb{E}_n\|\hat{\nabla} f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2$

7.5.4.3 *Bounding* $\mathbb{E}_n\|\hat{\nabla} f(X) - \hat{\nabla} f_{n,h}(X)\|_2$

First we introduce a lemma which is a modification of Lemma 6 appearing in [85]

Lemma 19. *Let $t + h \leq \tau$. We have for all $i \in [d]$, and all $s \in \{-t, t\}$:*

$$|\tilde{f}_{n,h,c}(\mathbf{x} + se_i) - f_c(\mathbf{x} + se_i)| \cdot \mathbb{1}_{A_{n,i}(\mathbf{x})} \leq hR$$

Proof. The proof follows the same logic as in [85], with the last step modified appropriately. To be more specific, let $x = X + se_i$, let $v_i = \frac{X_i - x}{\|X_i - x\|_2}$, then we have

$$\begin{aligned} |\tilde{f}_{n,h,c}(\mathbf{x} + se_i) - f_c(\mathbf{x} + se_i)| &\leq \sum_{i\in[d]} w_i(x)|f(X_i) - f(x)| \\ &= \sum_{i\in[d]} w_i(x)\left|\int_0^{\|X_i - x\|_2} v_i^T \nabla f(x + tv_i)dt\right| \\ &\leq \sum_{i\in[d]} w_i(x)\|X_i - x\|_2 \cdot \max_{x' \in \mathcal{X} + B(0,\tau)} \|v_i^T \nabla f(x)\|_2 \\ &\leq \sum_{i\in[d]} w_i(x)\|X_i - x\|_2 R \\ &\leq hR \end{aligned}$$

□

Lemma 20. *There exist a constant $C = C(\mu, K(\cdot))$, such that the following holds with probability at least $1 - 2\delta$ over the choice of X. Define $A(n) = 0.25 \cdot \sqrt{Cd \cdot \ln(kn/\delta)}$, for all $i \in [d], k \in [c]$, and all $s \in \{-t, t\}$:*

$$\mathbb{E}_n|\tilde{f}_{n,h,k}(X + se_i) - f_{n,h,k}(X + se_i)|^2 \cdot \mathbb{1}_{A_{n,i}(X)} \leq \frac{A(n)}{nh^d}$$

Proof. The proof follows a similar line of argument as made for the proof of Lemma 7 in [85]. First fix any $k \in [c]$, Assume $A_{n,i}(X)$ is true, and fix $x = X + se_i$. Taking conditional expectation on $\mathbf{Y}^n = Y_1, ..., Y_n$ given $\mathbf{X}^n = X_1, ..., X_n$, we have

$$\mathbb{E}_{\mathbf{Y}^n|\mathbf{X}^n}|f_{n,h,k}(x) - \tilde{f}_{n,h,k}(x)|^2 \leq 0.25 \cdot \sum_{i\in[n]} (w_i(x))^2 \leq 0.25 \cdot \max_{i\in[n]} w_i(x)$$

Use $\mathbf{Y}_x^n$ to denote corresponding Y_i of samples $X_i \in B(x,h)$.

Next, we consider the random variable

$$\psi(\mathbf{Y}_x^n) = |f_{n,h,k}(x) - \tilde{f}_{n,h,k}(x)|^2$$

Let $\mathcal{Y}_\delta$ denote the event that for all $Y_i \in \mathbf{Y}^n$, $|Y_i - f(X_i)|^2 \leq 0.25$. We know $\mathcal{Y}_\delta$ happens with probability at least $1/2$. Thus

$$\begin{aligned}\mathbb{P}_{\mathbf{Y}^n|\mathbf{X}^n}(\psi(\mathbf{Y^n}_x) > 2\mathbb{E}_{\mathbf{Y}^n|\mathbf{X}^n}\psi(\mathbf{Y^n}_x) + \epsilon) &\leq \mathbb{P}_{\mathbf{Y}^n|\mathbf{X}^n}(\psi(\mathbf{Y^n}_x) > \mathbb{E}_{\mathbf{Y}^n|\mathbf{X}^n,\mathcal{Y}_\delta}\psi(\mathbf{Y^n}_x) + \epsilon) \\ &\leq \mathbb{P}_{\mathbf{Y}^n|\mathbf{X}^n,\mathcal{Y}_\delta}(\psi(\mathbf{Y^n}_x) > \mathbb{E}_{\mathbf{Y}^n|\mathbf{X}^n,\mathcal{Y}_\delta}\psi(\mathbf{Y^n}_x) + \epsilon) + \delta/2\end{aligned}$$

By McDiarmid's inequality, we have

$$\mathbb{P}_{\mathbf{Y}^n|\mathbf{X}^n,\mathcal{Y}_\delta}(\psi(\mathbf{Y^n}_x) > \mathbb{E}_{\mathbf{Y}^n|\mathbf{X}^n,\mathcal{Y}_\delta}\psi(\mathbf{Y^n}_x) + \epsilon) \leq \exp\left\{-2\epsilon^2 \cdot \delta_Y^4 \sum_{i\in[n]} w_i^4(x)\right\}$$

The number of possible sets $\mathbf{Y}_x^n$ (over $x \in \mathcal{X}$) is at most the n-shattering number of balls in $\mathbb{R}^d$, using Sauer's lemma we get the number is bounded by $(2n)^{d+2}$. By union bound, with probability at least $1-\delta$, for all $x \in \mathcal{X}$ satisfying $B(x,h/2)\bigcap \mathbf{X}^n \neq \varnothing$,

$$\begin{aligned}\psi(\mathbf{Y}_x^n) &\leq 2\mathbb{E}_{\mathbf{Y}^n|\mathbf{X}^n}\psi(\mathbf{Y^n}_x) + \sqrt{0.25\cdot(d+2)\cdot\log(n/\delta)\cdot\sum_{i\in[n]} w_i^4(x)} \\ &\leq 2\sqrt{\mathbb{E}_{\mathbf{Y}^n|\mathbf{X}^n}\psi^2(\mathbf{Y^n}_x)} + \sqrt{0.25\cdot(d+2)\cdot\log(n/\delta)\cdot\delta_Y^4 \max_{i\in[n]} w_i^2(x)} \\ &\leq \sqrt{Cd\cdot\log(n/\delta)\cdot 0.25/n^2\mu_n^2(B(x,h/2))}\end{aligned}$$

Take a union bound over $k \in [c]$, and take empirical expectation, we get $\forall k \in [c]$

$$\mathbb{E}_n|\tilde{f}_{n,h,k}(X+se_i) - f_{n,h,k}(X+se_i)|^2 \leq \frac{0.25\cdot\sqrt{Cd\cdot\ln(cn/\delta)}}{n}\sum_{i\in[n]}\frac{1}{n(x_i,h/2)}$$

where $n(x_i,h/2) = n\mu_n(B(x_i,h/2))$ is the number of points in Ball $B(x_i,h/2)$.

Let $\mathcal{Z}$ denote the minimum $h/4$ cover of $\{x_1,...,x_n\}$, which means for any x_i, there is a $z \in \mathcal{Z}$, such that x_i is contained in the ball $B(z,h/4)$. Since $x_i \in B(z,h/4)$, we have $B(z,h/4) \in B(x_i,h/2)$. We also assume every x_i is assigned to the closest $z \in \mathcal{Z}$, and write $x_i \to z$ to denote such x_i. Then we have:

$$\begin{aligned}\sum_{i\in[n]}\frac{1}{n(x_i,h/2)} &= \sum_{z\in\mathcal{Z}}\sum_{x_i\to z}\frac{1}{n(x_i,h/2)} \\ &\leq \sum_{z\in\mathcal{Z}}\sum_{x_i\to z}\frac{1}{n(z,h/4)} \\ &\leq \sum_{z\in\mathcal{Z}}\frac{n(z,h/4)}{n(z,h/4)} \\ &= |\mathcal{Z}| \leq C_\mu(h/4)^{-d}\end{aligned}$$

Combining above analysis finishes the proof. □

Lemma 21. *There exists a constant $C = C(\mu, K(\cdot))$, such that the following holds with probability at least $1-2\delta$. Define $A(n) = 0.25 \cdot \sqrt{Cd \cdot \ln(kn/\delta)}$, $\forall k \in [c]$:*

$$\mathbb{E}_n \|\hat{\nabla} f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \leq \frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d} + h^2R^2}$$

Proof. First we can write the following bound for the l.h.s:

$$\begin{aligned}
\mathbb{E}_n \|\hat{\nabla} f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 &\leq \mathbb{E}_n \sqrt{\sum_{i\in[d]} |\Delta_{t,i} f_{n,h,k}(X) - \Delta_{t,i} f_k(X)|^2 \cdot \mathbb{1}_{A_{n,i}(X)}} \\
&\leq \sqrt{\sum_{i\in[d]} \mathbb{E}_n |\Delta_{t,i} f_{n,h,k}(X) - \Delta_{t,i} f_k(X)|^2 \cdot \mathbb{1}_{A_{n,i}(X)}} \\
&\leq \sqrt{\sum_{i\in[d]} \frac{1}{t^2} \max_{s\in\{-t,t\}} \mathbb{E}_n |f_{n,h,k}(X+se_i) - f_k(X+se_i)|^2 \cdot \mathbb{1}_{A_{n,i}(X)}}
\end{aligned}$$

First observe that:

$$\begin{aligned}
\mathbb{E}_n |f_{n,h,k}(X+se_i) - f_k(X+se_i)|^2 \cdot \mathbb{1}_{A_{n,i}(X)} &\leq \mathbb{E}_n |\tilde{f}_{n,h,k}(X+se_i) - f_k(X+se_i)|^2 \cdot \mathbb{1}_{A_{n,i}(X)} \\
&+ \mathbb{E}_n |\tilde{f}_{n,h,k}(X+se_i) - f_{n,h,k}(X+se_i)|^2 \cdot \mathbb{1}_{A_{n,i}(X)}
\end{aligned}$$

Also notice that: $\mathbb{E}_n |\tilde{f}_{n,h,k}(X+se_i) - f_k(X+se_i)|^2 \cdot \mathbb{1}_{A_{n,i}(X)}$ and $\mathbb{E}_n |\tilde{f}_{n,h,k}(X+se_i) - f_{n,h,k}(X+se_i)|^2 \cdot \mathbb{1}_{A_{n,i}(X)}$ can be respectively bounded by two lemmas from above, thus we get with probability at least $1-2\delta$

$$\mathbb{E}_n |f_{n,h,k}(X+se_i) - f_k(X+se_i)|^2 \leq h^2R^2 + \sqrt{\frac{A(n)}{nh^d}}$$

Combining above we get with probability at least $1-2\delta$, $\forall k \in [c]$

$$\|\hat{\nabla} f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \leq \frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d} + h^2R^2}$$

□

The following theorem provides a bound on $\mathbb{E}_n \|\nabla f(X) - \hat{\nabla} f_{n,h}(X)\|_2$:

Theorem 3. *With probability at least $1-2\delta$ over the choice of X, we have $\forall k \in [c]$:*

$$\begin{aligned}
\mathbb{E}_n \|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \leq \frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d} + h^2R^2} &+ R\left(\sqrt{\frac{d \ln \frac{d}{\delta}}{2n}} + \sqrt{\sum_{i\in[d]} \mu^2(\partial_{t,i}(\mathcal{X}))}\right) \\
&+ \sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2}
\end{aligned}$$

Proof. We start with the now familiar decomposition:

$$\begin{aligned}
\mathbb{E}_n \|\nabla f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \leq &\mathbb{E}_n \|\hat{\nabla} f_k(X) - \hat{\nabla} f_{n,h,k}(X)\|_2 \\
&+ \mathbb{E}_n \|\nabla f_k(X) \circ \mathbb{I}_n(X) - \hat{\nabla} f_k(X)\|_2 + \mathbb{E}_n \|\nabla f_k(X) \circ \overline{\mathbb{I}_n(X)}\|_2
\end{aligned}$$

By Lemma 17 we bound $\mathbb{E}_n \|\nabla f(X) \circ \overline{\mathbb{I}_n(X)}\|_2$; by Lemma 18 we bound $\mathbb{E}_n \|\nabla f(X) \circ \mathbb{I}_n(X) - \hat{\nabla} f(X)\|_2$; by Lemma 21 we bound $\mathbb{E}_n \|\hat{\nabla} f(X) - \hat{\nabla} f_{n,h}(X)\|_2$. Combining these results concludes the proof. □

7.6 BOUNDS ON EIGENVALUES AND EIGENSPACE VARIATIONS

In the above section, we established that $\mathbb{E}_n\hat{G}(X)$ is a consistent estimator of $\mathbb{E}_X G(X)$. In this section, we also establish consistency of its eigenvalues and eigenspaces,. The analysis here is based upon results from matrix perturbation theory [97, 98].

7.6.1 *Eigenvalues variation*

We begin by considering the following lemma for eigenvalues variation from matrix perturbation theory:

Lemma 22. *[97] Suppose both G and $\hat{G}$ are Hermitian matrices of size $d \times d$, and admit the following eigen-decompositions:*

$$G = X\Lambda X^{-1} \quad \textit{and} \quad \hat{G} = \hat{X}\hat{\Lambda}\hat{X}^{-1}$$

where X and $\hat{X}$ are nonsingular and

$$\Lambda = diag(\lambda_1, \lambda_2, ...\lambda_d) \quad \textit{and} \quad \hat{\Lambda} = diag(\hat{\lambda}_1, \hat{\lambda}_2, ...\hat{\lambda}_d)$$

and $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d$, $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq ... \geq \hat{\lambda}_d$. Thus for any unitary invariant norm $\|\cdot\|$, we have

$$\|diag(\lambda_1 - \hat{\lambda}_1, \lambda_2 - \hat{\lambda}_2, ..., \lambda_d - \hat{\lambda}_d)\| \leq \|G - \hat{G}\|$$

More specifically, when considering the spectral norm, we have

$$\max_{i\in[d]} |\lambda_i - \hat{\lambda}_i| \leq \|G - \hat{G}\|_2$$

and when considering the Frobenius norm, we have

$$\sqrt{\sum_{i\in[d]} |\lambda_i - \hat{\lambda}_i|^2} \leq \|G - \hat{G}\|_F$$

Using the above lemma, we obtain the following theorem that bounds the eigenvalue variation:

EIGENVALUE VARIATION BOUND

Theorem 4. *Let $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d$ be the eigen-values of $\mathbb{E}_X G(X)$, let $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq ... \geq \hat{\lambda}_d$ be the eigen-values of $\mathbb{E}_n\hat{G}(X)$. There exist $C = C(\mu, K(\cdot))$ and $N = N(\mu)$ such that the following holds with probability at least $1 - 2\delta$. Define $A(n) = \sqrt{Cd \cdot \log(n/\delta)} \cdot C_Y^2(\delta/2n) \cdot \sigma_Y^2/\log^2(n/\delta)$. Let $n \geq N$, we have:*

$$\max_{i\in[d]} |\lambda_i - \hat{\lambda}_i| \leq \frac{6R^2}{\sqrt{n}}(\sqrt{\ln d} + \sqrt{\ln\frac{1}{\delta}}) + \left(3R + \sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2} + \sqrt{d}(\frac{hR + C_Y(\delta)}{t})\right)$$
$$\left[\frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d} + h^2R^2} + R\left(\sqrt{\frac{d\ln\frac{d}{\delta}}{2n}} + \sqrt{\sum_{i\in[d]} \mu^2(\partial_{t,i}(\mathcal{X}))}\right) + \sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2}\right]$$

Proof. By Lemma 22, we bound $\max_{i\in[d]} |\lambda_i - \hat{\lambda}_i|$ with respect to $\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_X G(X)\|_2$; by Theorem 2 we bound $\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_X G(X)\|_2$. □

7.6.2 *Eigenspace variation*

First we introduce the following definition:

Definition 4. ***(Angles between two subspaces)*** *Let $X, \hat{X} \in \mathbb{R}^{d\times k}$ have full column rank k. The angle matrix $\Theta(X,\hat{X})$ between X and $\hat{X}$ is defined as:*

$$\Theta(X,\hat{X}) = \arccos((X^TX)^{-\frac{1}{2}}X^T\hat{X}(\hat{X}^T\hat{X})^{-1}\hat{X}^TX(X^TX)^{-\frac{1}{2}})^{\frac{1}{2}}$$

More specifically, when $k=1$, it reduces to the angle between two vectors:

$$\Theta(\mathbf{x},\hat{\mathbf{x}}) = \arccos\frac{|\mathbf{x}^T\hat{\mathbf{x}}|}{\|\mathbf{x}\|_2\|\hat{\mathbf{x}}\|_2}$$

Armed with this definition, we consider the following lemma on eigenspace variation:

Lemma 23. *[98] Suppose both G and $\hat{G}$ are Hermitian matrices of size $d\times d$, and admit the following eigen-decompositions:*

$$G = \begin{bmatrix} X_1 & X_2 \end{bmatrix}\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}\begin{bmatrix} X_1^{-1} \\ X_2^{-1} \end{bmatrix} \quad \text{and} \quad \hat{G} = \begin{bmatrix} \hat{X}_1 & \hat{X}_2 \end{bmatrix}\begin{bmatrix} \hat{\Lambda}_1 & 0 \\ 0 & \hat{\Lambda}_2 \end{bmatrix}\begin{bmatrix} \hat{X}_1^{-1} \\ \hat{X}_2^{-1} \end{bmatrix}$$

where $X = \begin{bmatrix} X_1 & X_2 \end{bmatrix}$ and $\hat{X} = \begin{bmatrix} \hat{X}_1 & \hat{X}_2 \end{bmatrix}$ are unitary. We have

$$\|\sin\Theta(X_1,\hat{X}_1)\|_2 \le \frac{\|(\hat{G}-G)X_1\|_2}{\min_{\lambda\in\lambda(\Lambda_1),\hat{\lambda}\in\lambda(\Lambda_2)}|\lambda-\hat{\lambda}|}$$

Using the above lemma, we get the following theorem for eigenspaces variant:

EIGENSPACE VARIATION

Theorem 5. *Write the eigen-decompositions of $\mathbb{E}_X G(X)$ and $\mathbb{E}_n\hat{G}(X)$ as*

$$\mathbb{E}_X G(X) = \begin{bmatrix} X_1 & X_2 \end{bmatrix}\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}\begin{bmatrix} X_1^{-1} \\ X_2^{-1} \end{bmatrix}, \mathbb{E}_n\hat{G}(X) = \begin{bmatrix} \hat{X}_1 & \hat{X}_2 \end{bmatrix}\begin{bmatrix} \hat{\Lambda}_1 & 0 \\ 0 & \hat{\Lambda}_2 \end{bmatrix}\begin{bmatrix} \hat{X}_1^{-1} \\ \hat{X}_2^{-1} \end{bmatrix}$$

There exist constants $C = C(\mu, K(\cdot))$ *and* $N = N(\mu)$ *such that the following holds with probability at least* $1 - 2\delta$. *Define* $A(n) = \sqrt{Cd \cdot \log(n/\delta)} \cdot C_Y^2(\delta/2n) \cdot \sigma_Y^2 / \log^2(n/\delta)$. *Let* $n \geq N$*:*

$$\|\sin\Theta(X_1, \hat{X}_1)\|_2 \leq \frac{\|X_1\|_2}{\min_{\lambda\in\lambda(\Lambda_1),\hat{\lambda}\in\lambda(\Lambda_2)} |\lambda - \hat{\lambda}|} \left(\frac{6R^2}{\sqrt{n}} (\sqrt{\ln d} + \sqrt{\ln \frac{1}{\delta}}) + \left(3R + \sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2} + \sqrt{d}(\frac{hR + C_Y(\delta)}{t}) \right) \left[\frac{\sqrt{d}}{t} \sqrt{\frac{A(n)}{nh^d}} + h^2R^2 + R \left(\sqrt{\frac{d \ln \frac{d}{\delta}}{2n}} + \sqrt{\sum_{i\in[d]} \mu^2(\partial_{t,i}(\mathcal{X}))} \right) + \sqrt{\sum_{i\in[d]} \epsilon_{t,i}^2} \right] \right)$$

Proof. By Lemma 23, we bound $\|\sin\Theta(X_1, \hat{X}_1)\|_2$ with respect to $\|X_1(\mathbb{E}_n\hat{G}(X) - \mathbb{E}_X G(X))\|_2$, since $\|X_1(\mathbb{E}_n\hat{G}(X) - \mathbb{E}_X G(X))\|_2 \leq \|X_1\|_2 \cdot \|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_X G(X)\|_2$, and by Theorem 2 we bound $\|\mathbb{E}_n\hat{G}(X) - \mathbb{E}_X G(X)\|_2$. Combining these concludes the proof. □

7.7 RECOVERY OF PROJECTED SEMIPARAMETRIC REGRESSION MODEL

In this section, the last on the theoretical analysis, we return to the multi-index motivation of the EGOP and EJOP discussed in the introduction to this chapter. For ease of exposition, we restrict our discussion to the EGOP, but the same argument also works for the EJOP.

Consider the following projected semiparametric regression model:

$$f(\mathbf{x}) = g(V^T\mathbf{x})$$

where $V \in \mathbb{R}^{d\times r}, r \ll d$ is a dimension-reduction projection matrix, and g is a non-parametric function. Without loss of generality, we assume $V = [v_1, v_2, ..., v_r]$, where $v_i \in R^d, i \in [r]$ is a set of orthonormal vectors, and the gradient outer product (GOP) matrix of g : $\mathbb{E}_X[\nabla g(V^TX) \cdot \nabla g(V^TX)^T]$ is nonsingular. The following proposition gives the eigen-decomposition of gradient outer product (GOP) matrix of f: $\mathbb{E}_X G(\mathbf{x})$

Proposition 3. *Suppose the eigen-decomposition of* $\mathbb{E}_X[\nabla g(V^TX) \cdot \nabla g(V^TX)^T]$ *is given by:*

$$\mathbb{E}_X[\nabla g(V^TX) \cdot \nabla g(V^TX)^T] = Z\Lambda Z^{-1}$$

then we have the following eigen-decomposition of $\mathbb{E}_X G(X)$*:*

$$\mathbb{E}_X G(X) = \begin{bmatrix} VZ & U \end{bmatrix} \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Z^{-1}V^T \\ U^T \end{bmatrix}$$

where $U = [u_1, u_2, ..., u_{d-r}]$, $u_i \in [d - r]$ *is a set of orthonormal vectors in ker*(V^T).

Proof. Since $f(\mathbf{x}) = g(V^T\mathbf{x})$, we have $\nabla f(\mathbf{x}) = V\nabla g(V^T\mathbf{x})$. Thus we get:

$$\mathbb{E}_X G(X) = V\mathbb{E}_X[\nabla g(V^TX) \cdot \nabla g(V^TX)^T]V^T = VZ\Lambda Z^{-1}V^T$$

When we check the eigen-decomposition given in the proposition, the above equation is satisfied. Moreover, since

$$\begin{bmatrix} VZ & U \end{bmatrix} \begin{bmatrix} Z^{-1}V^T \\ U^T \end{bmatrix} = \begin{bmatrix} Z^{-1}V^T \\ U^T \end{bmatrix} \begin{bmatrix} VZ & U \end{bmatrix} = I$$

concludes the proof. □

Since Z in the above proposition is nonsingular, we get that $\mathrm{im}(V) = \mathrm{im}(VZ)$, which means that the column space of projection matrix V is exactly the subspace spanned by the top-r eigenvectors of the GOP matrix $\mathbb{E}_X G(X)$. This point has also been noticed by [95, 164, 167].

Lastly, we need to show that the projection matrix V can be recovered using the estimated GOP matrix. This is captured in the following two theorems:

Recovery of Semi-parametric model

Theorem 6. *Suppose the function f we want to estimate has the form $f(\mathbf{x}) = g(V^T\mathbf{x})$, and $\tilde{V} \in \mathbb{R}^{d\times r}$ is the matrix composed by the top-r eigenvectors of $\mathbb{E}_n\hat{G}(X)$, then with probability at least $1-2\delta$:*

$$\|\sin\Theta(V,\tilde{V})\|_2 \leq \frac{1}{\lambda_{\min}}\left(\frac{6R^2}{\sqrt{n}}(\sqrt{\ln d}+\sqrt{\ln\frac{1}{\delta}}) + \left(3R + \sqrt{\sum_{i\in[d]}\epsilon_{t,i}^2} + \sqrt{d}(\frac{hR+C_Y(\delta)}{t})\right)\right.$$
$$\left.\left[\frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d}+h^2R^2} + R\left(\sqrt{\frac{d\ln\frac{d}{\delta}}{2n}} + \sqrt{\sum_{i\in[d]}\mu^2(\partial_{t,i}(\mathcal{X}))}\right) + \sqrt{\sum_{i\in[d]}\epsilon_{t,i}^2}\right]\right)$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\mathbb{E}_X[\nabla g(V^TX)\cdot\nabla g(V^TX)^T]$. Suppose $\lambda_1,\lambda_2,\ldots,\lambda_{d-r}$ are the lowest $d-r$ eigenvalues of $\mathbb{E}_n\hat{G}(X)$, and with probability at least $1-2\delta$:

$$max_{i\in[d-r]}|\lambda_i| \leq (\frac{6R^2}{\sqrt{n}}(\sqrt{\ln d}+\sqrt{\ln\frac{1}{\delta}}) + \left(3R + \sqrt{\sum_{i\in[d]}\epsilon_{t,i}^2} + \sqrt{d}(\frac{hR+C_Y(\delta)}{t})\right)$$
$$\left[\frac{\sqrt{d}}{t}\sqrt{\frac{A(n)}{nh^d}+h^2R^2} + R\left(\sqrt{\frac{d\ln\frac{d}{\delta}}{2n}} + \sqrt{\sum_{i\in[d]}\mu^2(\partial_{t,i}(\mathcal{X}))}\right) + \sqrt{\sum_{i\in[d]}\epsilon_{t,i}^2}\right])$$

Proof. We only sketch the proof. First of all, notice that V is a semi-orthogonal matrix, therefore $\|V\|_2 = 1$. When this observation is combined with above proposition and Theorem 5, we get a proof of the first part of the theorem. For proving the second part of the theorem, first observe that by proposition 3, the lowest $d-r$ eigenvalues of $\mathbb{E}_X G(X)$ are all zeros. This observation when combined with lemma 22 finishes the proof. □

7.8 CLASSIFICATION EXPERIMENTS

In this section, we give a brief experimental evaluation that examines the utility of the EJOP as a technique for metric estimation, when used in the setting of non-parametric classification. As in chapter 6, we consider non-parametric classifiers that rely on the notion of distance, parameterized by a matrix $\mathbf{M} \succeq 0$, with the squared distance computed as $(\mathbf{x} - \mathbf{x}')^T \mathbf{M}(\mathbf{x} - \mathbf{x}')$.

In the experiments reported in this section, we consider three different choices for $\mathbf{M}$:

1 $\mathbf{M} = \mathbf{I}$, which corresponds to the Euclidean distance

2 $\mathbf{M} = \mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix, the notion of distance in this case corresponds to a scaled Euclidean distance. In particular, in the absence of a gradients weights [85], [86] like approach for the multiclass case, we instead obtain weights by using the ReliefF procedure [75], which estimates weights for the multiclass case by a series of one versus all binary classifications.

3 $\mathbf{M} = \mathbb{E}_n G_n(X)$, where $\mathbb{E}_n G_n(X)$ is the estimated EJOP matrix.

In particular, letting $VDV^\top$ denote the spectral decomposition of $\mathbf{M}$, we use it to transform the input $\mathbf{x}$ as $D^{1/2}V^\top \mathbf{x}$ for the distance computation. Next, for a fixed choice of $\mathbf{M}$, we can define nearest neighbors of a query point $\mathbf{x}$ in various ways. We consider the following two ways:

1 k nearest neighbors (denoted henceforth as kNN) for fixed k

2 Neighbors that have distance $\leq h$ for fixed h from the query. We denote this as hNN. This corresponds to nonparametric classification using a boxcar kernel.

7.8.1 *A First Experiment on MNIST*

We first consider the MNIST dataset to test the quality of the EJOP metric, and if it improves upon plain Euclidean distance. In this case, we only test it for the kNN case, fixing $k = 7$. We set aside 10,000 points as a validation set, which is used to obtain the ReliefF weights, as well as for tuning the parameter t_i for $i = 1, \dots, 784$ in the EJOP estimation. While the t_i can be tuned separately for each class, we ignore that option in this set of experiments. Note that no preprocessing is applied on the images, and the metric estimation, as well as classification is done using the raw images. The results on the test set are illustrated in the following table:
While MNIST is a considerably easy task, the improvement given by the use of the EJOP as the distance metric over the plain Euclidean distance is substantial. This could perhaps be improved further by tuning t_i separately for each class. We will take this approach in the experiments described in the next section.

7.8.2 *Experiments on Datasets in [147] and [71]*

Next, we consider the datasets considered in [147] and [71], on which experiments are described in Chapters 3 and 4 as well. First we report experiments using plain Euclidean

Method	Error %
Euclidean	4.93
ReliefF	4.11
EJOP	2.37

Table 7.1: Error rates on MNIST using EJOP as the underlying metric, and comparison to Euclidean distance and scaled Euclidean distance

distance, h-NN and k-NN when the EJOP is used as the metric. The train/test splits are reported in the table. We split 20 % of the training portion to tune for h, k and t_i, the results reported are over 10 random runs as in Chapter 6.

Dataset	d	N	train/test	Euclidean	h-NN	k-NN
Isolet	172	7797	4000/2000	14.17 ± 0.7	10.14 ± 0.9	8.67 ± 0.6
USPS	256	9298	4000/2000	7.87 ± 0.2	7.14 ± 0.3	6.67 ± 0.4
Letters	16	20000	4000/2000	7.65 ± 0.3	5.12 ± 0.7	4.37 ± 0.4
DSLR	800	157	100/50	84.85 ± 4.8	41.13 ± 2.1	35.01 ± 1.4
Amazon	800	958	450/450	66.17 ± 2.8	41.07 ± 2.3	39.85 ± 1.5
Webcam	800	295	145/145	61.43 ± 1.7	24.86 ± 1.2	23.71 ± 2.1
Caltech	800	1123	550/500	85.41 ± 3.5	54.65 ± 2.6	52.86 ± 3.1

Table 7.2: Results comparing classification error rates on the datasets used in [71] using plain Euclidean distance, hNN and kNN while using the EJOP as the metric

Next, we consider the same datasets, and report results obtained on the same folds using three popular metric learning methods. In particular, we consider Large Margin Nearest Neighbors (LMNN) [157], Information Theoretic Metric Learning (ITML) [30] and Metric Learning to Rank (MLR) [107]. Since these methods explicitly optimize for the metric over a space of possible metrics, the comparison is manifestly unfair, since in the case of the EJOP, there is only one metric, which is estimated from the training samples. The setup is the same as discussed above, with the following addition for the metric learning methods: We learn the metric for $k = 5$, and test is using whatever k that was returned while tuning for the EJOP. We observe that despite its simplicity, EJOP does a decent job as compared to the metric learning methods, in some cases returning error rates comparable to those returned by MLR and ITML.

7.9 SUMMARY OF PART ON METRIC ESTIMATION

We conclude this part of the dissertation with a summary of the work undertaken, and some potential avenues for future work. Chapters 6 and 7 made the following contributions:

Dataset	h-NN	k-NN	ITML	LMNN	MLR
Isolet	10.14 ± 0.9	8.67 ± 0.6	8.43 ± 0.3	5.3 ± 0.4	6.59 ± 0.3
USPS	7.14 ± 0.3	6.67 ± 0.4	6.57 ± 0.2	6.23 ± 0.5	6.76 ± 0.3
Letters	5.12 ± 0.7	4.37 ± 0.4	5 ± 0.7	4.1 ± 0.4	17.81 ± 5.1
DSLR	41.13 ± 2.1	35.01 ± 1.4	21.65 ± 3.1	29.65 ± 3.7	41.54 ± 2.3
Amazon	41.07 ± 2.3	39.85 ± 1.5	39.83 ± 3.5	33.08 ± 4.2	29.65 ± 2.6
Webcam	24.86 ± 1.2	23.71 ± 2.1	15.31 ± 4.3	19.78 ± 1.5	27.54 ± 3.9
Caltech	54.65 ± 2.6	52.86 ± 3.1	52.37 ± 4.2	52.15 ± 3.2	51.34 ± 4.5

Table 7.3: Results comparing classification error rates given by the EJOP, and three popular metric learning methods

Summary of Part II

1 We described a simple estimator for the Expected Gradient Outerproduct (EGOP)

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x})^{\top} \right).$$

and demonstrated that it remains statistically consistent under mild assumptions. The estimated EGOP was then showed to be useful in nonparametric regression tasks when used as the underlying metric.

2 We extended the EGOP to the multiclass case, proposing a generalization that we refer to as the Expected Jacobian Outer Product (EJOP)

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\mathbf{J}_f(\mathbf{x}) \mathbf{J}_f(\mathbf{x})^T \right)$$

As in the case of the EGOP, we proposed a rough estimator for the EJOP, and also showed that it remained statistically consistent under similar assumptions. The EJOP was then used and shown to be experimentally useful as a metric in non-parametric classification tasks.

7.10 POTENTIAL AVENUES FOR FUTURE WORK

7.10.1 *Label Aware Dimensionality Reduction*

As discussed in Chapter 5, an attractive quality of the EGOP is that it recovers the average variation of f in *all* directions. It is this property that makes it useful for *effective dimension reduction*, that is, finding a $k << d$ dimensional subspace that is most relevant to predicting the output y. As discussed in Section 7.7, this multi-index motivation also carries through for the multiclass case by the EJOP.

Although explored somewhat cursorily by the dissertation author, it would be interesting to leverage the multi-index motivation of both the EGOP and the EJOP for the task of dimensionality reduction of data that takes into account the labels as well. This is contrasted to methods such as PCA, where the covariance matrix construction is completely label oblivious. Some experiments for dimensionality for the case of regression are reported by [112, 165] and by using metric learning are reported by [158], however not many applications were explored. The EGOP and EJOP can possibly be used to give a handy method for class aware dimensionality reduction.

7.10.2 *Operators that take into account local geometry*

We have the following, somewhat hand-wavy analogy between the EGOP and EJOP when put side by side with PCA. PCA helps recover directions according to how much variance in the data is explained by them, whereas the EGOP and EJOP help us recover directions according to the average variation of f. Both methods involve construction of a covariance matrix, and lose local information. We illustrate this with the EGOP

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x})^{\top} \right).$$

While gradients are local objects, since in the estimation of the EGOP, we take expectation over $\mathbf{x}$, all information about the local geometry is averaged out. We would like to construct operators that don't lose local information, and maybe give a non-linear map to a subspace that is most relevant to predict the output.

We can perhaps take inspiration from the literature in non-linear dimensionality reduction to search for an alternative. An attractive method, that unlike PCA does retain local information is exemplified by Laplacian Eigenmaps of Belkin and Niyogi [4]. In such methods, dimensionality reduction is achieved by the spectral decomposition of an operator that encodes the local geometry of the data. Usually, such an operator is a diffusion based object, such as the Graph Laplacian, defined as:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$$

where $\mathbf{D}$ and $\mathbf{W}$ are the degree and adjacency matrices respectively, of an appropriate nearest neighbor graph constructed on the data points. Taking a cue from this, we could define a *diffusion map using gradients* $\mathbf{W}$, for the regression and binary classification case as follows:

$$\mathbf{W}_{i,j} = \mathbf{W}_f(\mathbf{x}_i, \mathbf{x}_j) = exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_1} - \frac{\|\frac{1}{2}(\nabla f(\mathbf{x}_i) + \nabla f(\mathbf{x}_j)\dot{)}(\mathbf{x}_i - \mathbf{x}_j)\|^2}{\sigma_2} \right)$$

Such an operator has infact been discussed by [112, 165], but not explored in detail. For the multiclass case, we could consider the following:

$$\mathbf{W}_{i,j} = \mathbf{W}_f(\mathbf{x}_i, \mathbf{x}_j) = exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_1} - \frac{\|\frac{1}{2}(|\nabla f_c(\mathbf{x}_i)| + |\nabla f_c(\mathbf{x}_j|)\dot{(}\mathbf{x}_i - \mathbf{x}_j)\|^2}{\sigma_2} \right)$$

Where the operation $|\cdot|$ takes a matrix and sums over rows. In the above case $|\nabla f_c(\mathbf{x})|$ would be a d dimensional object, rather than $d \times c$.

Preliminary experiments on using the above operators for non-linear class-aware dimensionality reduction, as well as metric reweighing has yielded encouraging results. However, a detailed study is left for future work.

Finally, a somewhat more challenging avenue for future work would be to obtain consistent estimators for such objects, which are also cheap to estimate. Recall that in Eignemaps type methods, proving consistency involves showing that the eigenvectors of the graph Laplacian approach the eigenfunctions of the corresponding Laplace-Beltrami operator in the limit (see for example [5, 153]). It is not clear if such results (akin to those in sections 7.6 and 7.7) could be shown for the gradient based operators defined above. However, it could be a fruitful line of work to try and extend the EGOP and EJOP in such a way that the local geometry of the data could be taken into account.

Part III

GROUP EQUIVARIANT REPRESENTATION LEARNING

8 DISCRIMINATIVE REPRESENTATION LEARNING FOR SPHERICAL DATA

In the previous chapter we motivated group equivariant representation learning, in particular discriminative learning of such representations. In this chapter, we give a particular example: We describe a $SO(3)$ equivariant spherical CNN, which while learning $SO(3)$ equivariant representations discriminatively, also has the unusual feature that it can operate completely in Fourier space. Work presented in this chapter has appeared in the following publication [81].

Our starting point is the following theorem:

Theorem 7 (Kondor and Trivedi [82])**.** *A neural network connecting layers of the form $L^2(X_i, \mathbb{C}^{n_i})$ for a sequence of G-spaces X_i is G-equivariant if and only if it is a composition of G-convolutions on the X_i spaces and nonlinearities applied to $\mathbb{C}^{n_i}$*

A more general result for *steerable convolution* appears in the recent works of Cohen *et al.* [25], [26]. However, for our discussion it suffices to only consider the discussion in [82]. One of the main contributions of [82] is to give a spectral account for group equivariant networks, making the above theorem actionable to design neural networks that are equivariant to the action of general compact groups. In particular, [82] demonstrates that if a compact group G acts on the inputs of the neural network, then there is a natural Fourier transformation with respect to the group G, which gives a sequence of Fourier matrices at each layer. In particular, the linear operation at a given layer will be equivariant to the action of G if and only if it involves multiplying the Fourier matrices with learnable weight matrices from the right. It is this insight that we will use to present a neural network architecture that operates on spherical data, while being equivariant to rotations of the sphere.

We follow recent work on Spherical CNNs by Cohen *et al.* [24] (also see [43]), which presents a $SO(3)$ equivariant spherical neural network architecture using a generalized $SO(3)$ Fourier transform. One of the drawbacks of their approach is that the non-linearity still needs to be applied in real space, which leads to a non-conventional architecture which involves forward and backward Fourier transforms, which while being expensive can also cause numerical errors. In what follows we propose a spherical CNN architecture that is strictly more general, but at the same time operates entirely in Fourier space. It must be noted that our methodology is more general in its import–it can be used to design neural networks that are equivariant to the action of any continuous compact group.

In the next section, we describe the general set-up and notation to explicate on our approach.

8.1 NOTATION AND BASIC DEFINITIONS

8.1.1 *The Unit Sphere*

The sphere $\mathcal{S}^2$ with unit radius can be defined as the set of points $\mathbf{x} \in \mathbb{R}^3$ such that $\|\mathbf{x} - \mathbf{x}_0\| = 1$, where $\mathbf{x}_0$ is the origin. We can represent a sphere conveniently in spherical coordinates: for some $\mathbf{x} = [x_1, x_2, x_3]$ we can write $x_1 = r\cos\theta\sin\phi$, $x_2 = r\sin\theta\sin\phi$ and $x_3 = r\cos\phi$, where $\theta \in [0, 2\pi]$ is the azimuthal coordinate i.e. the longitude and $\phi \in [0, \pi]$ is the polar coordinate i.e. the co-latitude.

8.1.2 *Signals*

We work with spherical images represented by $f(\theta,\phi)$ and corresponding filters $h(\theta,\phi)$, which are taken to be continuous, complex valued functions. That is:

$$f, h : \mathcal{S}^2 \to \mathbb{C}^k$$

For most of the discussion in this chapter we simply work with $f, h : \mathcal{S}^2 \to \mathbb{C}$ for ease of exposition.

8.1.3 *Rotations*

We denote a rotation $R \in SO(3)$, and parametrize it by the familiar ZYZ Euler angles α, β, γ and denote it as $R(\alpha,\beta,\gamma)$. Any rotation $R(\alpha,\beta,\gamma)$ can thus be written as the following sequence of rotations along the z and y axes:

$$R(\alpha,\beta,\gamma) = R_z(\gamma)R_y(\beta)R_z(\alpha) \qquad \alpha,\gamma \in [0,2\pi), \beta \in [0,\pi]$$

Thus any spherical image $h(\theta,\phi)$ when subject to rotation R could be denoted as:

$$h_R(\theta,\phi) = R_z(\gamma)R_y(\beta)R_z(\alpha)(h)(\theta,\phi) \tag{8.1}$$

Alternatively, if x denotes the point at position (θ,ϕ), we denote it as

$$h_R(x) = h(R^{-1}x) \qquad R \in SO(3) \tag{8.2}$$

8.2 CORRELATION ON THE SPHERE

In classical convolutional neural networks, given an input feature map $f : \mathbb{Z}^2 \to \mathbb{R}$ and a filter $g : \mathbb{Z}^2 \to \mathbb{R}$, the value of the output feature map at some point $(-x, -y)$ is simply the inner product between the input and the filter translated by (x, y). Thus the process of correlation here can just be seen as pattern matching: the output map would have a stronger activation if it has a high correlation with the filter.

In order to define a spherical CNN, we would want to first state an appropriate notion for correlation between $f, g \in L^2(\mathcal{S}^2)$, when g is rotated and matched with f in analogy with the planar CNN case. The difference in this case however is that, unlike in the planar case, where the translation group and the input (the plane) that it acts on are isomorphic to each other, in the spherical case, they are no longer the same. This can lead to some consternation regarding the correct notion of spherical correlation.

However, as beautifully pointed out by Chirikjian and Kyatkin [19], a definition of correlation that does not veer off from the notion of pattern matching discussed above is rather simple:

$$(h \star f)(R) = \frac{1}{4\pi} \int_0^{2\pi} \int_{-\pi}^{\pi} [h_R(\theta,\phi)]^* f(\theta,\phi) \cos\theta d\theta d\phi \qquad R \in SO(3) \tag{8.3}$$

* denotes complex conjugation. Thus the spherical correlation is function on the rotation group $SO(3)$ rather than on $\mathcal{S}^2$.

At first blush, the rather foreboding double integral in equation 8.3 is what we would want to implement in our neural network. But this is problematic, one reason for which is that no perfectly symmetrical discretizations for spheres exist [144].

8.3 FILTERS AND FEATURE MAPS IN FOURIER SPACE

Instead of working with $f(\theta,\phi)$ and $h(\theta,\phi)$ in real space, we instead move to the Fourier domain. It is well known that for functions on the sphere $f \in L^2(\mathcal{S}^2)$, in direct analogy for periodic functions on the circle, the eigenfunctions of the spherical Laplacian give a basis. These basis functions are the so-called spherical harmonics. We can thus represent $f(\theta,\phi)$ and $h(\theta,\phi)$ in terms of their spherical harmonics expansions.

$$f(\theta,\phi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \hat{f}_m^\ell(\theta,\phi) Y_m^\ell(\theta,\phi) \tag{8.4}$$

$$h(\theta,\phi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \hat{h}_m^\ell(\theta,\phi) Y_m^\ell(\theta,\phi) \tag{8.5}$$

As might be already clear, $Y_m^\ell(\theta,\phi)$ are the spherical harmonics with $\ell \geq 0$ and $m \in \{-\ell,\dots,\ell\}$, and are written as:

$$Y_m^\ell(\theta,\phi) = (-1)^m \sqrt{\frac{(2\ell+1)(\ell-m)!}{4\pi(\ell+m)!}} P_m^\ell(\cos\theta) e^{im\phi}, \qquad m = -\ell,\dots,\ell \tag{8.6}$$

here P_m^ℓ denote the associated Legendre functions.

The coefficients of this spherical Fourier transform are found as follows:

$$\hat{f}_m^\ell = \frac{1}{4\pi} \int_{(\theta,\phi)\in\mathcal{S}^2} f(\theta,\phi) Y_m^\ell(\theta,\phi) \cos\theta d\theta d\phi \tag{8.7}$$

$$\hat{h}_m^\ell = \frac{1}{4\pi} \int_{(\theta,\phi)\in\mathcal{S}^2} h(\theta,\phi) Y_m^\ell(\theta,\phi) \cos\theta d\theta d\phi \tag{8.8}$$

Above we have described how to write $f(\theta,\phi)$ and $h(\theta,\phi)$ in Fourier space. However, recall that correlation defined in equation 8.3 was a function on $SO(3)$. We thus need to work with a Fourier transform on the rotation group. Thankfully, non-commutative harmonic analysis [19] provides us with such a notion. For functions $f \in L^2(SO(3))$.

The Fourier transform can be seen as a change of basis for the L_2 space of complex valued functions on $SO(3)$ to the irreducible representations. More specifically, for some function $g : SO(3) \to \mathbb{C}$, the SO(3)-Fourier transform is the collection of the following matrices:

$$G_\ell = \int_{SO(3)} g(R)\rho_\ell(R)d\mu(R) \qquad \ell = 0,1,2,\dots \tag{8.9}$$

Where $\rho_\ell(R) \in \mathbb{C}^{2\ell+1\times 2\ell+1}$ are the Wigner D-matrices, which are the irreducible representations for the group $SO(3)$. As a corollary of Schur's first lemma, we also know that the spherical harmonics also provide us with a basis for the irreducible representations of $SO(3)$. That is $Y^\ell_R = \rho_\ell(R)Y^\ell(\theta,\phi)$, and the elements of $\rho_\ell(R)$ are given as:

$$\rho^{mn}_\ell(R) = e^{-im\gamma}d^\ell_{mn}(\cos\beta)e^{-in\alpha} \qquad m,n = -\ell\dots,\ell \tag{8.10}$$

Where d^ℓ_{mn} correspond to the Wigner little-d matrices. This is a good point to revisit the choice to keep activations and filters to be complex valued $f,h : \mathcal{S}^2 \to \mathbb{C}$. Note that ρ_ℓ matrices are complex valued, and thus allowing activations and filters to be also complex valued simplifies implementation, as well will see when we describe our network.

Coming back, having defined the Fourier transform, we would also need the inverse Fourier transform, which is defined as below:

$$g(R) = \sum_{\ell=0}^{\infty} Tr[G_\ell\rho_\ell(R^{-1})] \tag{8.11}$$

Having described Fourier transforms for $f \in L^2(\mathcal{S}^2)$ and $f' \in L^2(SO(3))$, we now consider our spherical correlation formulation again:

$$(h \star f)(R) = \frac{1}{4\pi}\int_0^{2\pi}\int_{-\pi}^{\pi}[h_R(\theta,\phi)]^* f(\theta,\phi)\cos\theta d\theta d\phi \qquad R \in SO(3) \tag{8.12}$$

It can be shown (see [19] and Appendix of [24]) that the $SO(3)$ correlation satisfies a Fourier theorem, reducing finding $SO(3)$ Fourier coefficients to simply pointwise multiplications of the spherical Fourier coefficients. That is, in the above equation, each component is simply given as (here † denotes the hermitian conjugate):

$$[\widehat{h\star f}]_\ell = \hat{f}_\ell\hat{h}^\dagger_\ell \qquad \ell = 0,1,\dots,L \tag{8.13}$$

In layers $s = 2,\dots,S$ of a spherical CNN, the filters and the activations are no longer a function on the sphere, but rather on $SO(3)$. In that case, rather unsurprisingly (see equation 8.10 and preceding discussion), we have a similar convolution theorem

$$[\widehat{h\star f}]_\ell = F_\ell H^\dagger_\ell \qquad \ell = 0,1,\dots,L \tag{8.14}$$

and since we are working with functions on $SO(3)$, F_ℓ and H_ℓ are of course matrices.

The approach of Cohen *et al.* is essentially based on equations 8.13 and 8.14, where instead of working with the continuous function f, which as we have already seen might be complicated to work with, we work with the coefficients $\hat{f}_\ell$ with $\ell = 0,1,\dots,L$ and regard them as the activations of the neural network. Likewise $\hat{h}_\ell$ with $\ell = 0,1,\dots,L$ are regarded as the learn-able filters.

8.4 A GENERALIZED SO(3)-COVARIANT SPHERICAL CNN

In the previous section we discussed two Fourier theorems, which form the bedrock on which the work of [24] was based. We now consider $SO(3)$ correlation i.e. equation 8.12 again, but view it from an algebraic point of view. Specifically, we would like to first nail down, how it behaves under rotations. To begin, we consider the fact that when a spherical function $f(\theta, \phi)$ is subject to a rotation as discussed in 8.1.3, then the Fourier components are modulated by the Wigner D-matrix corresponding to the rotation R i.e.

$$f \mapsto f_R \iff \hat{f}_\ell \mapsto \rho_\ell(R)\hat{f}_\ell \tag{8.15}$$

Likewise, for a function $h : SO(3) \to \mathbb{C}$, which is subject to a rotation R, we have an analogous effect on the Fourier matrices i.e. they are modulated by the corresponding Wigner D-matrix[1].

$$h(R') \mapsto h(R^{-1}R') \iff G_\ell \mapsto \rho_\ell(R)G_\ell \tag{8.16}$$

Where G_ℓ are the Fourier matrices of h. The following proposition states that matrices output in equation 8.13 exhibits similar behavior.

Proposition 4. *Suppose $f : \mathcal{S}^2 \to \mathbb{C}$ is an activation function that under a rotation R transforms as $f \mapsto f(R^{-1}x) \quad R \in SO(3)$, and also suppose $h : \mathcal{S}^2 \to \mathbb{C}$ is a filter. Then, each component in the cross-correlation formula 8.13 transforms as:*

$$\widehat{[h \star f]}_\ell \mapsto \rho_\ell(R)\widehat{[h \star f]}_\ell \tag{8.17}$$

An identical claim can be made in the context of equation 8.14, which we state separately for the sake of completeness.

Proposition 5. *Suppose $f : SO(3) \to \mathbb{C}$ is an activation function that under a rotation R transforms as $f \mapsto f_R(R') \quad R \in SO(3)$, and also suppose $h : SO(3) \to \mathbb{C}$ is a filter. Then, each component in the cross-correlation formula 8.14 transforms as:*

$$\widehat{[h \star f]}_\ell \mapsto \rho_\ell(R)\widehat{[h \star f]}_\ell \tag{8.18}$$

Notice that equation 8.17 describes how spherical harmonic vectors transform under a rotation, while equation 8.18 describes the behaviour of Fourier matrices under a rotation. This similarity is not superficial. Indeed, we could understand the latter to mean that each column of the Fourier matrices will instead transform according to 8.17. It is this observation that leads us to a general definition of a SO(3) covariant Spherical CNN.

1 The usage of modulation is in analogy with classical Fourier analysis on the real line. Where a shift in the time domain causes the frequency to be multiplied by a complex exponential $x(t - t_0) \iff e^{-i\omega t_0}X(\omega)$. In the case of Fourier analysis on compact groups, a *shift* in the time domain, in this case a rotation, corresponds to a modulation by the irreducible representation in the frequency domain (in this case the Wigner D-matrix corresponding to R). Note that while $\mathbb{R}$ is not compact $e^{-i\omega t_0}$ is infact an irreducible representation for t_0.

Generalized $SO(3)$-covariant Spherical CNN

Definition 5. *Let $\mathcal{N}$ be a $S+1$ layer feed-forward network which takes as input $f^0 : \mathcal{S}^2 \to \mathbb{C}$. We say that $\mathcal{N}$ is a generalized $SO(3)$-covariant Spherical CNN if the output of each layer can be expressed as a collection of vectors:*

$$\hat{f}^s = \left(\hat{f}^s{}_{0,1}, \hat{f}^s{}_{0,2}, \ldots, \hat{f}^s{}_{0,\tau_0^s}, \hat{f}^s{}_{1,1}, \hat{f}^s{}_{1,2}, \ldots, \hat{f}^s{}_{1,\tau_1^s}, \ldots\ldots\ldots \hat{f}^s{}_{L,\tau_L^s}\right) \tag{8.19}$$

where each $\hat{f}^s{}_{\ell,j} \in \mathbb{C}^{2\ell+1}$ is a ρ_ℓ-covariant vector in the sense of 8.17. We call each individual $\hat{f}^s{}_{\ell,j}$ vector an irreducible fragment of $\hat{f}^s$. The integer vector $\tau^s = (\tau_0^s, \tau_1^s, \ldots, \tau_L^s)$ that counts the number of fragments for each ℓ, we call as the type of $\hat{f}^s$

The above gives a concrete definition of a $SO(3)$-covariant spherical CNN, however, to fully specify the neural network, we have to explicate on three things:

1 A linear transformation in each layer that involves learnable weights. Given that the output of each layer has the form in equation 8.19, we need to specify how they can be mixed. Moreover, the linear transformation must be covariant.

2 A covariant non-linearity on top of the linear transformation.

3 Final output that is rotation-invariant.

We consider these points one by one.

8.5 COVARIANT LINEAR TRANSFORMATIONS

For a neural network to be covariant, the linear transformation applied at each layer must also be covariant. In the case of the network defined above, the prescription for this is encapsulated in the following proposition. Note that this proposition is a special case of the theorem introduced in the introduction of this chapter.

Proposition 6. *Suppose $\hat{f}^s$ is a $SO(3)$-covariant activation function that has the form $\hat{f}^s = \left(\hat{f}^s{}_{0,1}, \hat{f}^s{}_{0,2}, \ldots, \hat{f}^s{}_{0,\tau_0^s}, \hat{f}^s{}_{1,1}, \hat{f}^s{}_{1,2}, \ldots, \hat{f}^s{}_{1,\tau_1^s}, \ldots\ldots\ldots \hat{f}^s{}_{L,\tau_L^s}\right)$, and yet another function $\hat{g}^s = \mathcal{L}(\hat{f}^s)$, which is a linear function of $\hat{f}^s$ expressed similarly. Then $\hat{g}^s$ is $SO(3)$-covariant iff each $\hat{g}^s{}_{\ell,j}$ fragment is a linear combination of fragments from $\hat{f}^s$ with the same ℓ*

Recall that each fragment $\hat{f}^s{}_{\ell,j}$ is $2\ell+1$ dimensional. If we concatenate all the fragments corresponding to a fixed ℓ into a matrix denoted F_ℓ^s, and likewise do the same for $\hat{g}$. Then the proposition basically says that $G_\ell^s = F_\ell^s W_\ell^s$ for all ℓ. It is these parameters that are learned in our network. We must also note the generality of this formulation by considering that both equations 8.17and 8.18 are particular cases, although the W_ℓ does not yield to a good interpretation in terms of cross-correlation.

8.6 COVARIANT NON-LINEARITIES

Next we turn our attention to the design of a non-linearity that is both differentiable as well as covariant. The choice of non-linearity is absolutely crucial to the success of neural networks. Besides, in the case of networks that are equivariant, usually we work with non-linearities in real space. The reason for this to easy to understand. Being pointwise operations, these are automatically equivariant. Designing a non-linearity that is both covariant and differentiable in Fourier space is far more challenging. It is for this reason that other work in group equivariant networks always apply the non-linearity in real space. However, these backward-forward transformations can be expensive, and can be a cause for a number of complications, including partially losing equivariance due to quadrature.

Here we take a rather unusual route to solve this problem: We take tensor products between fragments, but note that since each of the fragments was irreducible, after tensor products they no longer might be so. To maintain covariance, we would want the fragments to be irreducible. This problem can be solved exactly by the so called Clebsch-Gordan decomposition.

In representation theory, the Clebsch-Gordan decomposition arises in the context of decomposing the tensor product of irreducible representations in a direct sum of irreducibles. In particular, for the group $SO(3)$, it takes the form:

$$\rho_{\ell_1}(R) \otimes \rho_{\ell_2}(R) = C_{\ell_1,\ell_2} \left[\bigoplus_{\ell=|\ell_1-\ell_2|}^{\ell_1+\ell_2} \rho_\ell(R) \right] C^T_{\ell_1,\ell_2}$$

Equivalently, we can write:

$$\rho_\ell(R) = C^T_{\ell_1,\ell_2,\ell} \Big[\rho_{\ell_1}(R) \otimes \rho_{\ell_2}(R) \Big] C_{\ell_1,\ell_2,\ell}$$

Where $C_{\ell_1,\ell_2,\ell}$ are appropriate blocks of C_{ℓ_1,ℓ_2}. The utility of the CG-transform for our purpose is encapsulated in the following lemma:

Lemma 24. *Let $\hat{f}_{\ell_1}$ and $\hat{f}_{\ell_2}$ denote ρ_{ℓ_1} and ρ_{ℓ_2} covariant vectors, and let ℓ denote any integer between $|\ell_1 - \ell_2|$ and $\ell_1 + \ell + 2$. Then*

$$\hat{g}_\ell = C^T_{\ell_1,\ell_2,\ell} \Big[\hat{f}_{\ell_1} \otimes \hat{f}_{\ell_2} \Big] \tag{8.20}$$

is a ρ_ℓ covariant vector.

The algorithm then consists of finding 8.20 between all pairs of fragments and then stacking them horizontally, resulting in possibly very wide matrices: in our parlance the activations, or number of channels increase substantially. This can be controlled by fixing, for each ℓ, the maximum number of fragments to be $\bar{\tau}_\ell$. Thankfully, this can be done by using the learnable weight matrices (discussed in section 8.5).

8.7 FINAL INVARIANT LAYER

Since we need the network to be rotation invariant, we implement this by considering only the $\hat{f}^S_{0,j}$ fragments in the last layer. This is because the $\ell = 0$ representation is constant, and thus rotation invariant. We can then connect fully connected layers on top of this last Fourier layer.

With all the ingredients in place, we now describe our experiments.

8.8 EXPERIMENTS

In this section we describe experiments that give a direct comparison with those reported by Cohen *et al.* [24]. We choose these experiments as the Spherical CNN proposed in [24] is the only direct competition to our method. Besides, the comparison is also instructive for two different reasons: Firstly, while the procedure used in [24] is exactly equivariant in the discrete case, for the continuous case they use a discretization which causes their network to partially lose equivariance with changing bandwidth and depth, whereas our method is always equivariant in the exact sense. Secondly, owing to the nature of their architecture and discretization, [24] use a more traditional non-linearity i.e. the ReLU, which is also quite powerful. In our case, to maintain full covariance and to avoid the quadrature, we use an unconventional quadratic non-linearity in Fourier space. Because of these two differences, the experiments will hopefully demonstrate the advantages of avoiding the quadrature and maintaining full equivariance despite using a purportedly weaker nonlinearity.

Cohen *et al.* present two sets of experiments: In the first sequence, they study the numerical stability of their algorithm and quantify the equivariance error due to the quadrature. In the second, they present results on three datasets comparing with other methods. Since our method is fully equivariant, we focus on the second set of experiments.

8.8.1 *Rotated MNIST on the Sphere*

We use a version of MNIST in which the images are painted onto a sphere and use two instances as in [24]: One in which the digits are projected onto the Northern hemisphere and another in which the digits are projected on the sphere and are also randomly rotated.

The baseline model is a classical CNN with 5×5 filters and 32, 64, 10 channels with a stride of 3 in each layer (roughly 68K parameters). This CNN is trained by mapping the digits from the sphere back onto the plane, resulting in nonlinear distortions. The second model that we compare to is the Spherical CNN proposed in [24]. For this method, we use the same architecture as reported by the authors i.e. having layers S^2 convolution – ReLU – $SO(3)$ convolution – ReLU – Fully connected layer with bandwidths 30, 10 and 6, and the number of channels being 20, 40 and 10 (resulting in a total of 58K parameters).

For our method we use the following architecture: We set the bandlimit $L_{max} = 8$, and keep $\tau_l = \lceil \frac{12}{\sqrt{L+1}} \rceil$, using a total of 5 layers as described in section **??**, followed by a fully connected layer of size 256 by 10. We use batch normalization [65] on the fully connected layer, and a variant of batch normalization that preserves covariance in the Fourier layers. This method takes a moving average of the standard deviation for a particular fragment for all examples seen during training till then and divides by it, the parameter corresponding to the mean in usual batch normalization is kept to be zero as anything else will break covariance. Finally, we concatenate the output of each F_0^s in each internal layer, which are $SO(3)$ invariant scalars, along with that of the last layer to construct the fully connected layer. We observed that having these skip connections was crucial to facilitate smooth training. The total number of parameters was 342086, the network was trained by using the ADAM optimization procedure [73] with a batch size of 50 and a learning rate of 5×10^{-4}. We also used L2 weight decay of 0.00001 on the trainable parameters.

We report three sets of experiments: For the first set both the training and test sets were not rotated (denoted NR/NR), for the second, the training set was not rotated while the test was randomly rotated (NR/R) and finally when both the training and test sets were rotated (denoted R/R).

Method	NR/NR	NR/R	R/R
Baseline CNN	97.67	22.18	12
Cohen *et al.*	95.59	94.62	93.4
Ours (FFS2CNN)	96	95.86	95.8

We observe that the baseline model's performance deteriorates in the three cases, effectively reducing to random chance in the R/R case. While our results are better than those reported in [24], they also have another characteristic: they remain roughly the same in the three regimes, while those of [24] slightly worsen. We think this might be a result of the loss of equivariance in their method.

8.8.2 *Atomization Energy Prediction*

Next, we apply our framework to the QM7 dataset [10, 129], where the goal is to regress over atomization energies of molecules given atomic positions (p_i) and charges (z_i). Each molecule contains up to 23 atoms of 5 types (C, N, O, S, H). We use the Coulomb Matrix (CM) representation proposed by [129], which is rotation and translation invariant but not permutation invariant. The Coulomb matrix $C \in \mathbb{R}^{N \times N}$ is defined such that for a pair of atoms $i \neq j$, $C_{ij} = (z_i z_j)/(|p_i - p_j|)$, which represents the Coulomb repulsion, and for atoms $i = j$, $C_{ii} = 0.5 z_i^{2.4}$, which denotes the atomic energy due to charge. To test our algorithm we use the same set up as in [24]: We define a sphere S_i around p_i for each atom i. Ensuring uniform radius across atoms and molecules and ensuring no intersections amongst spheres during training, we define potential functions $U_z(x) = \sum_{j \neq i, z_j = z} \frac{ziz}{|x - p_i|}$ for every z and for every x on S_i. This yields a T channel spherical signal for each atom in a molecule. This signal is then discretized using Driscol-Healy [39] grid using a bandwidth of $b = 10$. This gives a sparse tensor representation of dimension $N \times T \times 2b \times 2b$ for every molecule.

Our spherical CNN architecture has the same parameters and hyperparameters as in the previous subsection except that $\tau_l = 15$ for all layers, increasing the number of parameters to 1.1 M. Following [24], we share weights amongst atoms and each molecule is represented as a $N \times F$ tensor where F represents F_0^s scalars concatenated together. Finally, we use the approach proposed in [171] to ensure permutation invariance. The feature vector for each atom is projected onto 150 dimensions using a MLP. These embeddings are summed over atoms, and then the regression target is trained using another MLP having 50 hidden units. Both of these MLPs are jointly trained. The final results are presented below, which show that our method outperforms the Spherical CNN of Cohen *et al.*. The only method that delivers better performance is a MLP trained on randomly permuted Coulomb matrices [110], and as [24] point out, this method is unlikely to scale to large molecules as it needs a large sample of random permutations, which grows rapidly with N.

Method	RMSE
MLP/Random CM [110]	**5.96**
LGIKA (RF) [124]	10.82
RBF Kernels/Random CM [110]	11.42
RBF Kernels/Sorted CM [110]	12.59
MLP/Sorted CM [110]	16.06
Spherical CNN [24]	8.47
Ours (FFS2CNN)	**7.91**

8.8.3 *3D Shape Recognition*

Finally, we report results for shape classification using the SHREC17 dataset [131], which is a subset of the larger ShapeNet dataset [18] having roughly 51300 3D models spread over 55 categories. It is divided into a 70/10/20 split for train/validation/test. Two versions of this dataset are available: A regular version in which the objects are consistently aligned and another where the 3D models are perturbed by random rotations. Following [24] we focus on the latter version, as well as represent each 3D mesh as a spherical signal by using a ray casting scheme. For each point on the sphere, a ray towards the origin is sent which collects the ray length, cosine and sine of the surface angle. In addition to this, ray casting for the convex hull of the mesh gives additional information, resulting in 6 channels. The spherical signal is discretized using the Discroll-Healy grid [39] with a bandwidth of 128. We use the code provided by [24] for generating this representation.

We use a ResNet style architecture, but with the difference that the full input is not fed back but rather different frequency parts of it. We consider $L_{max} = 14$, and first train a block only till $L = 8$ using $\tau_l = 10$ using 3 layers. The next block consists of concatenating the fragments obtained from the previous block and training for two layers till $L = 10$, repeating this process till L_{max} is reached. These later blocks use $\tau_l = 8$. As earlier, we concatenate the F_0^s scalars from each block to form the final output layer, which is connected to 55 nodes forming a fully connected layer. We use Batch Normalization in the final layer, and the normalization discussed in 8.8.1 in the Fourier layers. The model was trained with ADAM using a batch size of 100 and a learning rate of 5×10^{-4}, using L2 weight decay of 0.0005 for regularization. The total number of parameters was roughly 2.3M. We compare our results using the SHREC competition evaluation script to some of the top performing models on SHREC (which use architectures specialized to the task) as well as the model of Cohen *et al.*. Our method, like the model of Cohen *et al.* is task agnostic and uses the same representation. Despite this, it is able to consistently come second or third in the competition (while being neck to neck with Cohen *et al.*), showing that it affords an efficient method to learn from spherical signals.

Method	P@N	R@N	F1@N	mAP	NDCG
Tatsuma_ReVGG	0.705	0.769	0.719	0.696	0.783
Furuya_DLAN	0.814	0.683	0.706	0.656	0.754
SHREC16-Bai_GIFT	0.678	0.667	0.661	0.607	0.735
Deng_CM-VGG5-6DB	0.412	0.706	0.472	0.524	0.624
Spherical CNNs [24]	0.701	0.711	0.699	0.676	0.756
FFS2CNNs (ours)	0.707	0.722	0.701	0.683	0.756

8.9 CONCLUSION

In conclusion, in this chapter, we presented a SO(3)-equivariant neural network architecture for spherical data, that operates entirely in Fourier space, while using tensor products and the Clebsch-Gordan decomposition as the only source of non-linearity. We report strong (and perhaps surprising) experimental results. While we specifically presented a spherical CNN, our approach is more widely applicable in that it also provides a formalism for the design of fully Fourier neural networks that are equivariant to the action of any continuous compact group.

9 CONCLUSIONS AND FUTURE DIRECTIONS

9.1 PART I

9.1.1 *Conclusions*

In chapter 2 we reviewed some relevant literature on metric learning; following which, in chapter 3, we proposed a metric learning method that makes a more direct attempt to optimize for k-NN accuracy than existing methods. While the approach is more general in its formulation (in that it can handle non-linear metrics as well), in chapter 3 we demonstrated its efficacy for learning Mahalanobis metrics while comparing to a number of popular competing methods. In chapter 4, we proposed a number of extensions of this approach, applying it to asymmetric similarity learning, discriminative learning of Hamming distance, and metric learning for improving k-NN regression performance. In each case we reported competitive results. Below we underline some straightforward avenues for future work:

9.1.2 *Future Directions*

1 A drawback of the approaches presented in Part i, with the exception of section 4.3, is poor scalability. For every gradient update, the procedures require exact inference and loss-augmented inference. For small dataset sizes this is desirable, however being expensive operations (see section 3.4.1.4) they restrict scaling these methods to very large datasets. One future avenue of work is to make these methods more scalable while retaining some of their positive characteristics as constrasted to methods such as Large Margin Nearest Neighbors (LMNN). Some strategies to achieve this could take the route of doing exact inference for h^* and $\hat{h}$ for a fixed number of gradient updates $N' << N$ (where N is the number of training examples) in the beginning of the optimization. Once the initial Euclidean metric is improved to a somewhat better performing metric, the sets h^* and $\hat{h}$ can be fixed for the next p gradient updates, after which they are updated again by doing exact inference. This process could be repeated to convergence. Another route could be to pick large batches and do exact inference in only a given batch, and not the entire dataset.

2 Yet another avenue for future work is to propose approximate inference procedures for h^* and $\hat{h}$, and combining them with the approaches outlined above.

3 Extending the approach using deep neural networks to do the mapping is also an obvious extension. This was explored by the dissertation author, but not extensively. It is arguable that the metric thus learned could be a better proxy for similarity than approaches based on triplet based losses.

4 For section 4.3, unlike other approaches presented in Part i, the inference procedures were intractable. We thus resorted to a modification of the loss function to make inference tractable. It would be interesting to explore approximation algorithms for the original, intractable formulations for h^* and $\hat{h}$. Yet another approach to this problem, explored by the dissertation author to some degree, is to keep the original intractable objective, and devising a Metropolis-Hastings type procedure for sampling sets and then updating the metric. Lastly, it would also be interesting to cast the framework in section 4.3 as a structured prediction energy network in the spirit of Tu and Gimpel [150].

9.2 PART II

9.2.1 *Conclusions*

In part ii of the dissertation, we proposed a simple estimator for the Expected Gradient Outerproduct (EGOP)

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x})^{\top} \right),$$

moreover, we also showed that it remains statistically consistent under mild assumptions. The primary use of the estimated EGOP was as the underlying metric in non-parametric regression, and we showed that it improved performance as compared to the Euclidean distance in several real world datasets. We also generalized the EGOP to the multiclass case, proposing a variant called the Expected Jacobian Outer Product (EJOP)

$$\mathbb{E}_{\mathbf{x}} G(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{x}} \left(\mathbf{J}_f(\mathbf{x}) \mathbf{J}_f(\mathbf{x})^{T} \right),$$

for which we also proposed a simple estimator and showed that it remained statistically consistent under similarly mild assumptions. We also showed that the estimated EJOP improved non-parameteric classificaiton when used as a metric.

9.2.2 *Future Directions*

1 One immediate use case for the approaches presented in Part ii, namely, the estimated Expected Gradient Outer Product (EGOP) and Expected Jacobian Outer Product (EJOP), is for dimensionality reduction, that unlike PCA type methods recover a subpsace most relevant to predicting the output. This has not been explored in detail and could potentially be a useful addition to the standard toolbox for dimensionality reduction.

2 The EGOP and the EJOP use gradients, which are local objects, but due to the expectation taken over $\mathbf{x}$, they lose all local information, only giving the average variation of the unknown regression or classification function f in direction $\mathbf{v}$. It would be interesting to explore the utility of diffusion based objects that take into account local geometry as well, first analogous to the EGOP

$$\mathbf{W}_{i,j} = \mathbf{W}_f(\mathbf{x}_i, \mathbf{x}_j) = exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_1} - \frac{\|\frac{1}{2}(\nabla f(\mathbf{x}_i) + \nabla f(\mathbf{x}_j))\dot{(}\mathbf{x}_i - \mathbf{x}_j)\|^2}{\sigma_2} \right)$$

and then analogous to the EJOP

$$\mathbf{W}_{i,j} = \mathbf{W}_f(\mathbf{x}_i, \mathbf{x}_j) = exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_1} - \frac{\|\frac{1}{2}(|\nabla f_c(\mathbf{x}_i)| + |\nabla f_c(\mathbf{x}_j|)\dot{(}\mathbf{x}_i - \mathbf{x}_j)\|^2}{\sigma_2} \right)$$

$|\cdot|$ takes a matrix and sums over rows, and explore them for both recovering a metric, as well as for non-linear label-aware dimensionality reduction.

3 For the operators defined above it would also be interesting to obtain consistency results, similar in spirit to those obtained for Laplacian Eigenmaps type methods [5, 153]. For such methods, demonstrating consistency amounts to showing that the eigenvectors of the graph Laplacian converge to the eigenfucntions of the corresponding Laplace-Beltrami operator in the limit. However, this promises to be a rather challenging project, which might also require considerable refinement in definitions of these objects.

9.3 PART III

9.3.1 *Conclusions*

In chapter **??** we briefly reviewed work on discriminative group equivariant representation learning, arguing that equivariance to symmetry transformations affords a strong inductive bias in various tasks. In chapter 8, following recent work by Kondor and Trivedi [82], we proposed a SO(3)-equivariant neural network architecture for spherical data, that operates entirely in Fourier space, while using tensor products and the Clebsch-Gordan decomposition as the only source of non-linearity. We reported strong experimental results, and emphasized the wider applicability of our approach, in that it also provides a formalism for the design of fully Fourier neural networks that are equivariant to the action of any continuous compact group.

9.3.2 *Future Directions*

1 We first outline a future avenue for work that relates directly to the contributions presented in Chapter 8. Although the network architecture presented is the most general possible for the problem and mathematically elegant in its conception, while also giving excellent performance, it does lead to networks that are considerably bulkier than networks trained by [24]. In view of the dissertation author this inefficiency might be a consequence of the non-locality of filters. In vision tasks, it is perhaps much better motivated to use filters that operate on a small, spatially contiguous domain of the input. Thus, the use of more global filters could be the reason that the networks slid toward having more parameters to also pick up more local features on their own. Enforcing locality of filters in order to improve the efficiency of our network further is the most immediate line of future work. This should have relevance to not just $SO(3)-$equivariant networks that operate on data that lives on $\mathcal{S}^2$, but to networks for vision tasks that are required to be equivariant to the action of general compact continuous groups.

2 More directly related to the more general theme covered in Part iii of this dissertation, is to design convolutional neural networks that encode more structure from the data and task at hand, by considering different groups and their homogeneous spaces. A simple example would be to design a $SIM(3)-$equivariant architecture, to follow works that present $SE(3)-$equivariant networks (for example see [155]). This could perhaps be considered low-hanging fruit, notwithstanding the fact that fast implementations of such architectures may require considerable engineering effort.

3 There are many applications where exact invariances to symmetry transformations are important, such as in Robotics and motion planning, tomography, camera calibration, molecular dynamics, protein kinematics etc. In these areas there already exists a large literature on using non-commutative harmonic analysis for functions defined on some homogeneous space of a group of interest. An extensive review of such approaches and half a dozen applications is given in [19]. Naturally, designing equivariant architectures that extend older approaches to also avoid feature engineering is an obvious line of work to pursue.

4 Most of the work on group equivariant neural networks reviewed in chapter **??** relies on the assumption that the functions are defined on a suitable homogeneous space of the symmetry group of interest. Indeed, the architecture presented in chapter 8 is rooted in the fact that the manifold $\mathcal{S}^2$ is a homogeneous space of the rotation group $SO(3)$. There is recent interest in extending the convolutional neural network formalism to more general manifolds [106] that might not come equipped with a clear group action. Work that prescribes construction of theoretically well motivated convolutional networks on such spaces promises to be a very fruitful line for future work.

5 In most of the work discussed thus far, we have considered neural networks that are equivariant to explicit (and known) symmetry transforms. A considerably difficult project would be to instead to *learn the symmetry group*, without prior knowledge of symmetries in the data. The only work that we are aware of in this direction is that of Anselmi *et al.* [2].

6 The theory and design of covariant neural architectures in the context of recurrent neural networks and general dynamical systems also promises to be an interesting project. A simple situation that illustrates this occurs in Koopman mode analysis [16], where we want the Koopman invariant subspace to respect some underlying symmetry (for example if the physical system is subject to a rotation). To our knowledge there is no work toward this very reasonable end goal.

7 It would be interesting to explore connections and usages of equivariant networks in the context of *Pattern Theory* [113], which is a mathematical formalism to study patterns from the bottom up–with a focus on building compositional vocabularies. While pattern theory has influenced directly, or indirectly many modern machine learning algorithms, it by itself has largely been forgotten in the machine learning mainstream. However, the general philosophy and approach of Pattern Theory remains relevant and can be seen as echoed in many recent works in group equivariant architectures, and could also provide inspiration for future work. While broad

in its coverage, it would be useful to consider salient aspects in some notable works within pattern theory. First is the construction of shapes, manifolds and surfaces. Second is the comparison of such objects, and lastly is a methodology to define variability (deformations) and using appropriate probability measures for inference. Usually the space of variability of such objects is an orbit under symmetry transformations. As already hinted, yet another important aspect about Pattern Theory is its focus on compositionality. The case for general covariant compositional architectures has been made in recent work [63], [83], and it would also be interesting to extend these works to also be able to define probabilistic models in the spirit of [113].There is rich mathematical literature on probabilities on algebraic structures (see for example [54]) that Pattern Theory draws upon and could also provide fertile ground for the growth of more general, topologically sane work in neural networks.

BIBLIOGRAPHY

[1] F. Anselmi, J. Z. Leibo, J. Mutch, A. Tacchetti, and T. Poggio. "Unsupervised learning of invariant representations with low sample complexity." In: *Technical Report: MIT Center for Brains, Minds and Machines* (2014).

[2] Fabio Anselmi, Georgios Evangelopoulos, Lorenzo Rosasco, and Tomaso Poggio. "Symmetry Regularization." In: *Technical Report: MIT Center for Brains, Minds and Machines* (2017).

[3] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. "An optimal algorithm for approximate nearest neighbor searching fixed dimensions." In: *Journal of the ACM (JACM)* 45.6 (1998), pp. 891–923.

[4] Mikhail Belkin and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation." In: *Neural computation* 15 (2003), pp. 1373–1396.

[5] Mikhail Belkin and Partha Niyogi. "Convergence of Laplacian Eigenmaps." In: *Advances in Neural Information Processing Systems*. 2007, pp. 129–136.

[6] Yoshua Bengio. "Learning deep architectures for AI." In: *Foundations and trends in Machine Learning* (2009), pp. 1–127.

[7] Yoshua Bengio and Yann LeCun. "Scaling learning algorithms towards AI." In: *Large-scale kernel machines* (2007), pp. 1–41.

[8] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. "Greedy layer-wise training of deep networks." In: *Advances in neural information processing systems* (2007), pp. 153–160.

[9] Alina Beygelzimer, Sham Kakade, and John Langford. "Cover trees for nearest neighbor." In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 97–104.

[10] L. C. Blum and J.-L. Reymond. "970 million druglike small molecules for virtual screening in the chemical universe database GDB-13." In: *Journal of the American Chemical Society* (2009).

[11] W. Boomsma and J. Frellsen. "Spherical convolutions and their application in molecular modelling." In: NIPS (2017), pp. 3436–3446.

[12] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers." In: *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT'92)*. ACM Press, 1992, pp. 144–152.

[13] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean data." In: *IEEE Signal Process. Mag.* 34.4 (2017), pp. 18–42. DOI: 10.1109/MSP.2017.2693418. URL: https://doi.org/10.1109/MSP.2017.2693418.

[14] Joan Bruna and Stephane Mallat. "Invariant scattering convolutional networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2013), pp. 1872–1886.

[15] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral Networks and Locally connected networks on graphs." In: *iclr*. 2014.

[16] Marko Budišić, Ryan Mohr, and Igor Mezić. "Applied Koopmanism." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.4 (2012), p. 047510. DOI: 10.1063/1.4772195. eprint: https://doi.org/10.1063/1.4772195. URL: https://doi.org/10.1063/1.4772195.

[17] Michael A. Casey. "Auditory group theory with applications to statistical basis methods for structured audio." In: *PhD Thesis, MIT* (1998).

[18] A. X. Chang et al. "ShapeNet: An Information-Rich 3D Model Repository." In: *arXiv:1512.03012* (2015). arXiv: arXiv:1512.03012.

[19] Gregory S. Chirikjian and Alexander B. Kyatkin. *Harmonic Analysis for Engineers and Applied Scientists: Updated and Expanded Edition*. Courier Dover Publications, 2016.

[20] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification." In: *arXiv:1202.2745* (2012). arXiv: arXiv:1202.2745.

[21] William S Cleveland and Susan J Devlin. "Locally weighted regression: an approach to regression analysis by local fitting." In: *Journal of the American Statistical Association* 83.403 (1988), pp. 596–610.

[22] T. S. Cohen and M. Welling. "Group equivariant convolutional networks." In: *Proceedings of The 33rd International Conference on Machine Learning* 48 (2016), pp. 2990–2999. arXiv: 1602.07576.

[23] T. S. Cohen and M. Welling. "Steerable CNNs." In: *ICLR*. 2017.

[24] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. "Spherical CNNs." In: *International Conference on Learning Representations* (2018).

[25] Taco S. Cohen, Mario Geiger, and Maurice Weiler. "Intertwiners between Induced Representations (with Applications to the Theory of Equivariant Neural Networks)." In: *arXiv* (2018). arXiv: 1803.10743. URL: https://arxiv.org/abs/1803.10743.

[26] Taco S. Cohen, Mario Geiger, and Maurice Weiler. "The Quite General Theory of Equivariant Convolutional Networks." In: *to appear* (2018).

[27] Taco S. Cohen and Max Welling. "Transformation properties of learned visual representations." In: *iclr*. 2015.

[28] Thomas Cover and Peter Hart. "Nearest Neighbor Pattern Classification." In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27.

[29] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. "Locality-sensitive hashing scheme based on p-stable distributions." In: *Proceedings of the twentieth annual symposium on Computational geometry*. ACM. 2004, pp. 253–262.

[30] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. "Information-theoretic metric learning." In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 209–216.

[31] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering." In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 3844–3852. URL: http://papers.nips.cc/paper/6081-convolutional-neural-networks-on-graphs-with-fast-localized-spectral-filtering.pdf.

[32] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering." In: *nips*. 2016.

[33] Luc Devroye, Lászlo Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*. Cambridge, MA, USA: Springer, 1997. ISBN: 0387946187.

[34] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer, Berlin, Heidelberg, 2009. ISBN: 3642002331.

[35] S. Dieleman, J. De Fauw, and K. Kavukcouglu. "Exploiting cyclic symmetry in convolutional neural networks." In: *icml*. 2016.

[36] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. "Exploiting Cyclic Symmetry in Convolutional Neural Networks." In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 1889–1898. URL: http://dl.acm.org/citation.cfm?id=3045390.3045590.

[37] M. N. Do and M. Vetterli. "Rotation invariant texture characterization and retrieval using steerable wavelet-domain hidden markov models." In: *IEEE Transactions on Multimedia* 4 (2002), pp. 146–158.

[38] Wei Dong, Moses Charikar, and Kai Li. "Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces." In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (2008), pp. 123–130.

[39] J. R. Driscoll and D. M. Healy. "Computing Fourier transforms and convolutions on the 2-sphere." In: *Advances in Applied Mathematics* (1994).

[40] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. "Convolutional Networks on Graphs for Learning Molecular Fingerprints." In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 2224–2232. URL: http://dl.acm.org/citation.cfm?id=2969442.2969488.

[41] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gomez-Bombarelli, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. "Convolutional Networks on Graphs for Learning Molecular Fingerprints." In: *nips*. 2015.

[42] C. Esteves, C. Allen-Blanchette, X. Zhou, and K. Daniilidis. "Polar Transformer Networks." In: *arXiv:1709.01889* (2017).

[43] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. " Learning SO(3) Equivariant Representations with Spherical CNNs." In: *arXiv* (2017). arXiv: 1711.06721. URL: https://arxiv.org/abs/1711.06721.

[44] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. "Object detection with discriminatively trained part-based models." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.9 (2010), pp. 1627–1645.

[45] William T. Freeman and Edward H. Adelson. "The Design and Use of Steerable Filters." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13 (1991), pp. 891–906.

[46] Keinosuke Fukunaga and R. Short. "The optimal distance measure for nearest neighbor classification." In: *IEEE transactions on Information Theory* 27.5 (1981), pp. 622–627.

[47] R. Gens and P. Domingos. "Deep Symmetry Networks." In: *NIPS 2014* (2014), pp. 1–9.

[48] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. "Neural Message Passing for Quantum Chemistry." In: (). arXiv: arXiv:1704.01212.

[49] Amir Globerson and Sam Roweis. "Metric Learning by Collapsing Classes." In: *Advances in Neural Information Processing Systems 18*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Cambridge, MA: MIT Press, 2006, pp. 451–458.

[50] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. "Neighbourhood components analysis." In: *NIPS*. 2004.

[51] Yunchao Gong and Svetlana Lazebnik. "Iterative quantization: A procrustean approach to learning binary codes." In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 817–824.

[52] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Shrejil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems* (2014), pp. 2672–2680.

[53] Albert Gordo, Florent Perronnin, Yunchao Gong, and Svetlana Lazebnik. "Asymmetric distances for binary embeddings." In: *IEEE transactions on pattern analysis and machine intelligence* 36.1 (2014), pp. 33–47.

[54] Ulf Grenander. *Probabilities on algebraic structures*. Courier Corporation, 2008.

[55] Wolfgang Hardle, Peter Hall, Hidehiko Ichimura, et al. "Optimal smoothing in single-index models." In: *The annals of Statistics* 21.1 (1993), pp. 157–178.

[56] Trevor Hastie and Robert. Tibshirani. "Discriminant Adaptive Nearest Neighbor Classification." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18.6 (1996), pp. 607–616.

[57] Mikael Henaff, Joan Bruna, and Yann LeCun. *Deep convolutional networks on graph structured data*. Tech. rep. 2015. arXiv: arXiv:1506.05163.

[58] Geoffrey E. Hinton. "A parallel computation that assigns canonical object-based frames of reference." In: *ijcai*. 1981.

[59] Geoffrey E. Hinton, Alex Krizhevksy, and Sida D. Wang. "Transforming Auto-Encoders." In: *icann*. 2011.

[60] Geoffrey E. Hinton, James L. McClelland, and David E. Rumelhart. "Distributed representations." In: *Carnegie-Mellon University* (1984), pp. 1–127.

[61] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." In: *Neural Computation* 18.7 (2006), pp. 1527–1554.

[62] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." In: *Science* 313.5786 (2006), pp. 504–507.

[63] Truong Son Hy, Shubhendu Trivedi, Horace Pan, Brandon M. Anderson, and Risi Kondor. "Predicting molecular properties with covariant compositional networks." In: *The Journal of Chemical Physics* 148.24 (2018), p. 241745. DOI: 10.1063/1.5024797. eprint: https://doi.org/10.1063/1.5024797. URL: https://doi.org/10.1063/1.5024797.

[64] Piotr Indyk and Rajeev Motwani. "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality." In: *Proceedings of the 13th annual ACM Symposium on Theory of Computing*. 1998, pp. 604–613.

[65] S. Ioffe and C. Szefedy. "Batch Normalization: Accelerating deep network training by reducing internal covariate shift." In: *International Conference on Machine Learning* (2015).

[66] Joern-Henrik Jacobsen, Bert de Brabandere, and Arnold W.M. Smeulders. *Dynamic Steerable Blocks in Deep Residual Networks*. Tech. rep. 2017. arXiv: arXiv:1706.00598.

[67] Thorsten Joachims. "A support vector method for multivariate performance measures." In: *Proceedings of the 22nd International Conference on Machine Learning*. ACM Press, 2005, pp. 377–384.

[68] Sham Kakade. "Lecture Notes on Multivariate Analysis, Dimensionality Reduction, and Spectral Methods." In: *STAT 991, Spring* (2010).

[69] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. "Locally scale-invariant convolutional neural networks." In: *nips*. 2014.

[70] Steven Kearns, Kevin McCloskey, Marc Brendl, Vijay Pande, and Patrick Riley. "Molecular graph convolutions: Moving beyond fingerprints." In: *Journal of Computer Aided Molecular Design* 30 (2016), pp. 595–608.

[71] Dor Kedem, Stephen Tyree, Kilian Weinberger, Fei Sha, and Gert Lanckriet. "Nonlinear Metric Learning." In: *Advances in Neural Information Processing Systems 25*. 2012, pp. 2582–2590.

[72] Philipp W. Keller, Shie Mannor, and Doina Precup. "Automatic basis function construction for approximate dynamic programming and reinforcement learning." In: *Proceedings of the 23rd International Conference on Machine Learning*. 2006.

[73] D. P. Kingma and J. Ba. "ADAM: A method for stochastic optimization." In: *International Conference on Learning Representations* (2015).

[74] Kenji Kira and Larry A. Rendell. "The Feature Selection Problem: Traditional Methods and a New Algorithm." In: *AAAI*. Ed. by William R. Swartout. AAAI Press / The MIT Press, 1992, pp. 129–134. ISBN: 0-262-51063-4. URL: http://dblp.uni-trier.de/db/conf/aaai/aaai92.html#KiraR92.

[75] Kenji Kira and Larry M. Rendell. "The feature selection problem: Traditional methods and a new algorithm." In: *Proceedings of AAAI*. AAAI. 1992, pp. 129–134.

[76] Jyri J. Kivinen and Christopher K. I. Williams. "Transformation equivariant restricted Boltzmann machines." In: *icann*. 2011.

[77] J. J. Koenderink and A. J. van Doorn. "Receptive Field Families." In: *Biological Cybernetics* 63 (1990), pp. 291–297.

[78] R. Kondor. "N-body Networks: a Covariant Hierarchical Neural Network Architecture for Learning Atomic Potentials." In: *ArXiv e-prints* (2018). arXiv: 1803.01588 [cs.LG].

[79] Risi Kondor. *A novel set of rotationally and translationally invariant features for images based on the non-commutative bispectrum*. Tech. rep. 2007. arXiv: arXiv:0701127v3.

[80] Risi Kondor. "Group theoretical models in machine learning." In: *PhD Thesis, Columbia University* (2008).

[81] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. "Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network." In: *arXiv:1806.09231* (2018). arXiv: arXiv:1806.09231.

[82] Risi Kondor and Shubhendu Trivedi. "On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 2747–2755. URL: http://proceedings.mlr.press/v80/kondor18a.html.

[83] Risi Kondor, Truong Hy Song, Horace Pan, Brandon M. Anderson, and Shubhendu Trivedi. "Covariant compositional networks for learning graphs." In: *arXiv:1801.02144* (2018). arXiv: arXiv:1801.02144.

[84] Igor Kononenko, Edvard Simec, and Marko Robnik-Sikonja. "Overcoming the myopia of inductive learning algorithms with RELIEFF." In: *Applied Intelligence* 7 (1997), pp. 39–55.

[85] Samory Kpotufe and Abdeslam Boularias. "Gradient Weights help Nonparametric Regressors." In: *NIPS*. 2012, pp. 2870–2878.

[86] Samory Kpotufe, Abdeslam Boularias, Thomas Schultz, and Kyoungok Kim. "Gradient Weights improve Regression and Classification." In: *Journal of Machine Learning Research* 17 (2016), pp. 1–34.

[87] Kai Krajsek and Rudolf Mester. "A unified theory for steerable and quadrature filters." In: *Communications in Computer and Information Science* 4 (2007), pp. 5–13.

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances In Neural Information Processing Systems* (2012), pp. 1–9. ISSN: 10495258. DOI: http://dx.doi.org/10.1016/j.protcy.2014.09.007. arXiv: 1102.0183.

[89] Brian Kulis and Trevor Darrell. "Learning to hash with binary reconstructive embeddings." In: *Advances in neural information processing systems* 22 (2009), pp. 1042–1050.

[90] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. "Efficient search for approximate nearest neighbor in high dimensional spaces." In: *SIAM Journal on Computing* 30.2 (2000), pp. 457–474.

[91] Y LeCun, B Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Backpropagation applied to handwritten zip code recognition.” In: *Neural Computation* 1 (1989), pp. 541–551.

[92] Y LeCun, L Bottou, Y Bengio, D Henderson, and P Haffner. “Gradient-based learning applied to document recognition.” In: *Proceedings of the IEEE* 86(11) (1998), pp. 2278–2324.

[93] Karel Lenc and Andrea Vedaldi. “Understanding image representations by measuring their equivariance and equivalence.” In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2015), pp. 991–999.

[94] Reiner Lenz. “Group Theoretical Model of Feature Extraction.” In: *Journal of the Optical Society of America* 6 (1989), pp. 827–834.

[95] Ker-Chau Li. “Sliced Inverse Regression for Dimension Reduction.” In: *Journal of the American Statistical Association* 86.414 (1991), pp. 316–327.

[96] Ker-Chau Li. “Sliced inverse regression for dimension reduction.” In: *Journal of the American Statistical Association* 86.414 (1991), pp. 316–327.

[97] Ren-Cang Li. “Relative perturbation theory I: eigenvalue and singular value variations.” In: *SIAM Journal on Matrix Analysis and Applications* 19 (1998), pp. 956–982.

[98] Ren-Cang Li. “Relative perturbation theory II: eigenspace and singular space variations.” In: *SIAM Journal on Matrix Analysis and Applications* 20 (1999), pp. 471–492.

[99] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. “Gated Graph Sequence Neural Networks.” In: *iclr*. 2016.

[100] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints.” In: *International Journal of Computer Vision* 60 (2004), pp. 91–110.

[101] James B. MacQueen. “Some Methods for classification and Analysis of Multivariate Observations.” In: *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* 1 (1967), pp. 281–297.

[102] Prasanta Chandra Mahalanobis. “On the generalized distance in statistics.” In: *Proceedings National Institute of Science of India* 49.2 (1977), pp. 234–256.

[103] S Mallat. *Group Invariant Scattering*. Tech. rep. 2012. arXiv: arXiv:1101.2286v3.

[104] Roberto Manduchi, Pietro Perona, and Doug Shy. “Efficient Deformable Filter Banks.” In: *IEEE Transactions on Signal Processing* 46 (1998), pp. 1168–1173.

[105] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. *Rotation equivariant vector field networks*. Tech. rep. 2017. arXiv: arXiv:1612.09346.

[106] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. “Geodesic Convolutional Neural Networks on Riemannian Manifolds.” In: *Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. ICCVW ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 832–840. ISBN: 978-1-4673-9711-7. DOI: 10.1109/ICCVW.2015.112. URL: http://dx.doi.org/10.1109/ICCVW.2015.112.

[107] B. McFee and G. R. G. Lanckriet. "Metric Learning to Rank." In: *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 2010.

[108] Markus Michaelis and Gerald Sommer. "A lie group approach to steerable filters." In: *Pattern Recognition Letters* 16 (1995), pp. 1165–1174.

[109] Shakir Mohamed and Balaji Lakshminarayanan. "Learning in implicit generative models." In: *arXiv:1610.03483* (2016). arXiv: arXiv:1610.03483.

[110] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, O.A. von Lilienfeld, and K. Müller. "Learning invariant representations of molecules for atomization energy prediction." In: *NIPS*. 2012.

[111] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 5425–5434. DOI: 10.1109/CVPR.2017.576. URL: https://doi.org/10.1109/CVPR.2017.576.

[112] Sayan Mukherjee, Qiang Wu, Ding-Xuan Zhou, et al. "Learning gradients on manifolds." In: *Bernoulli* 16.1 (2010), pp. 181–207.

[113] David Mumford and Agnès Desolneux. *Pattern theory: the stochastic analysis of real-world signals*. AK Peters/CRC Press, 2010.

[114] Behnam Neyshabur and Nathan Srebro. "On Symmetric and Asymmetric LSHs for Inner Product Search." In: *International Conference on Machine Learning*. 2015, pp. 1926–1934.

[115] Behnam Neyshabur, Nathan Srebro, Ruslan Salakhutdinov, Yury Makarychev, and Payman Yadollahpour. "The power of asymmetry in binary hashing." In: *Advances in Neural Information Processing Systems*. 2013, pp. 2823–2831.

[116] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs." In: *icml*. 2016.

[117] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 2014–2023. URL: http://proceedings.mlr.press/v48/niepert16.html.

[118] Mohammad Norouzi and David J Fleet. "Minimal loss hashing for compact binary codes." In: *mij* 1 (2011), p. 2.

[119] Mohammad Norouzi, David Fleet, and Ruslan Salakhutdinov. "Hamming Distance Metric Learning." In: *Advances in Neural Information Processing Systems 25*. 2012, pp. 1070–1078.

[120] E. Oyallon and S. Mallat. "Deep roto-translation scattering for object classification." In: *cvpr*. 2015.

[121] Pietro Perona. "Deformable Kernels for Early Vision." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17 (1995), pp. 488–499.

[122] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli. "Image Denoising Using Scale Mixtures of Gaussians in the Wavelet Domain." In: *IEEE Transactions on Image Processing* 12 (2003), pp. 1338–1351.

[123] James L Powell, James H Stock, and Thomas M Stoker. "Semiparametric estimation of index coefficients." In: *Econometrica: Journal of the Econometric Society* (1989), pp. 1403–1430.

[124] A. Raj, A. Kumar, Y. Mroueh, and P.T. Fletcher et al. "Local group invariant representations via orbit embeddings." In: *arXiv:1612.01988* (2016). arXiv: arXiv:1612.01988.

[125] Carl E. Rasmussen, Radford M. Neal, Geoffrey E. Hinton, Drew van Camp, Michael Revow, Zoubin Ghahramani, R. Kustra, and Robert Tibshirani. "The DELVE Manual." In: (1996).

[126] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA, USA: MIT Press, 2005. ISBN: 026218253X.

[127] S. Ravanbakhsh, J. Schneider, and B. Poczos. "Equivariance Through Parameter-Sharing." In: *Proceedings of International Conference on Machine Learning*. 2017.

[128] Marco Reisert. "Group integration techniques in pattern analysis: A kernel view." In: *PhD Thesis, Albert-Ludwigs University* (2008).

[129] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld. "Fast and accurate modeling of molecular atomization energies with machine learning." In: *Physical Review Letters* (2012).

[130] Ruslan Salakhutdinov and Geoffrey E. Hinton. "Semantic Hashing." In: *International Journal of Approximate Reasoning* 50.7 (2009), pp. 969–978.

[131] M. Savva et al. "Large-Scale 3D Shape Retrieval from ShapeNet Core55." In: *Eurographics Workshop on 3D Object Retrieval* (2017).

[132] Jürgen Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural networks* (2015), pp. 85–117.

[133] Gregory Shakhnarovich. *Learning task-specific similarity*. Ph.D. thesis, Massachutsetts Institute of Technology, 2005.

[134] Gregory Shakhnarovich, Trevor Darell, and Piotr Indyk. *Nearest-neighbor methods in learning and vision: theory and practice (neural information processing)*. Cambridge, MA, USA: MIT Press, 2006. ISBN: 026219547X.

[135] Laurent Sifre and Stephane Mallat. "Rotation, scaling and deformation invariant scattering for texture discrimination." In: *cvpr*. 2013.

[136] Eero P. Simoncelli, William T. Freeman, Edward H. Adelson, and David J. Heeger. "Shiftable Multiscale Transforms." In: *IEEE Transactions on Information Theory* 38 (1992), pp. 587–607.

[137] H. Skibbe. "Spherical tensor algebra for biomedical image analysis." In: *PhD Thesis, Albert-Ludwigs University* (2013).

[138] Alex J. Smola and Bernhard Schölkopf. "A tutorial on support vector regression." In: *Statistics and Computing* 14.3 (2004), pp. 199–222.

[139] Peter Stange. “On the efficient update of the singular value decomposition.” In: *PAMM* 8.1 (2008), pp. 10827–10828.

[140] Charles J. Stone. “Consistent nonparametric regression.” In: *The Annals of Statistics* (1977), pp. 595–620.

[141] David Tarlow, Kevin Swersky, Ilya Sutskever, and Richard S Zemel. “Stochastic k-Neighborhood Selection for Supervised and Unsupervised Learning.” In: *Proceedings of the 30th International Conference on Machine Learning* 28 (2013), pp. 199–207.

[142] P. C. Teo and Y. Hel-Or. “Design of multiparameter steerable functions using cascade basis reduction.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21 (1999), pp. 552–556.

[143] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. “Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds.” In: *arXiv:1802.08219 [cs]* (2018). arXiv: 1802.08219. URL: http://arxiv.org/abs/1802.08219.

[144] William P. Thurston. *Three-Dimensional Geometry and Topology, Volume 1*. Princeton University Press, 1997.

[145] Tijman Tieleman. “Optimizing Neural Networks that Generate Images.” In: *PhD Thesis, University of Toronto* (2014).

[146] Shubhendu Trivedi. “Notes on Asymmetric Metric Learning for k-NN Classification.” In: *Unpublished Notes*. 2015.

[147] Shubhendu Trivedi, David Mcallester, and Gregory Shakhnarovich. “Discriminative Metric Learning by Neighborhood Gerrymandering.” In: *Advances in Neural Information Processing Systems*. 2014, pp. 3392–3400.

[148] Shubhendu Trivedi, Jialei Wang, Samory Kpotufe, and Gregory Shakhnarovich. “A Consistent Estimator of the Expected Gradient Outerproduct.” In: *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence*. AUAI. 2014, pp. 819–828.

[149] Joel. A. Tropp. “User-Friendly Tools for Random Matrices: An Introduction.” In: *Tutorial at NIPS* (2012).

[150] Lifu Tu and Kevin Gimpel. “Learning approximate inference networks for structured prediction.” In: *arXiv preprint* (2018). arXiv: 1803.03376. URL: https://arxiv.org/abs/1807.02547.

[151] V. Vapnik and A. Chervonenkis. “On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities.” In: *Theory of Probability and Its Applications* 16.2 (1971), pp. 264–280.

[152] Ulrike Von Luxburg. “A tutorial on spectral clustering.” In: *Statistics and computing* 17.4 (2007), pp. 395–416.

[153] Ulrike Von Luxburg, Mikhail Belkin, and Olivier Bousquet. “Consistency of spectral clustering.” In: *Annals of Statistics* (2008), pp. 555–586.

[154] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. “Sequential projection learning for hashing with compact codes.” In: *Proceedings of International Conference on Machine Learning*. 2010.

[155] Maurice Weiler, Marco Geiger, Max Welling, Wouter Boomsma, and Taco S. Cohen. "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data." In: *arXiv preprint* (2018). arXiv: 1807.02547. URL: https://arxiv.org/abs/1807.02547.

[156] Kilian Q Weinberger and Lawrence K Saul. "Fast solvers and efficient implementations for distance metric learning." In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1160–1167.

[157] Kilian Q Weinberger and Lawrence K Saul. "Distance metric learning for large margin nearest neighbor classification." In: *The Journal of Machine Learning Research* 10 (2009), pp. 207–244.

[158] Kilian Q. Weinberger and Gerald Tesauro. "Metric Learning for Kernel Regression." In: *Artificial Intelligence and Statistics*. 2007, pp. 612–619.

[159] Yair Weiss, Antonio Torralba, and Rob Fergus. "Spectral Hashing." In: *Advances in Neural Information Processing Systems*. 2008.

[160] Christopher K. I. Williams and Carl E. Rasmussen. "Gaussian Processes for Regression." In: *Advances in Neural Processing Sysmtems*. 1996, pp. 514–520.

[161] Jeffrey Wood. "Invariant Pattern Recognition: A Review." In: *Pattern Recognition* 29 (1996), pp. 1–17.

[162] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. "Harmonic Networks: Deep Translation and Rotation Equivariance." In: *arXiv:1612.04642* (2016). arXiv: 1612.04642.

[163] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. *Harmonic Networks: Deep Translation and Rotation Equivariance*. Tech. rep. 2017. arXiv: arXiv:1612.04642.

[164] Qiang Wu, Justin Guinney, Mauro Maggioni, and Sayan Mukherjee. "Learning Gradients: Predictive Models that Infer Geometry and Statistical Dependence." In: *Journal of Machine Learning Research* 11 (2010), pp. 2175–2198.

[165] Qiang Wu, Justin Guinney, Mauro Maggioni, and Sayan Mukherjee. "Learning gradients: predictive models that infer geometry and statistical dependence." In: *The Journal of Machine Learning Research* 11 (2010), pp. 2175–2198.

[166] Yingcun Xia, Howell Tong, WK Li, and Li-Xing Zhu. "An adaptive estimation of dimension reduction space." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 64.3 (2002), pp. 363–410.

[167] Yingcun Xia, Howell Tong, W. K. Li, and Li-Xing Zhu. "An adaptive estimation of dimension reduction space." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 64.3 (2002), pp. 363–410. ISSN: 1467-9868. DOI: 10.1111/1467-9868.03411. URL: http://dx.doi.org/10.1111/1467-9868.03411.

[168] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. "Distance Metric Learning, with Application to Clustering with Side-information." In: *Advances in Neural Information Processing Systems 15*. MIT Press, 2002, pp. 505–512.

[169] Chun-Nam John Yu and Thorsten Joachims. "Learning structural SVMs with latent variables." In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, pp. 1169–1176.

[170] Alan L Yuille, Anand Rangarajan, and AL Yuille. "The concave-convex procedure (CCCP)." In: *Advances in neural information processing systems* 2 (2002), pp. 1033–1040.

[171] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. "Deep Sets." In: *arXiv:1703.06114* (2017). arXiv: `arXiv:1703.06114`.