Statistical Learning Theory approaches to clustering

Diplomarbeit im Fach Bioinformatik

vorgelegt von

Stefanie Jegelka

im November 2007

Wilhelm-Schickard Institut für Informatik
Universität Tübingen
und
Max Planck Institut für biologische Kybernetik

Betreuer:

Dr. Ulrike von Luxburg
Max Planck Institut für biologische Kybernetik
Tübingen

Prof. Dr. Michael Kaufmann
Wilhelm-Schickard Institut für Informatik
Universität Tübingen

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Erklärung

Die vorliegende Diplomarbeit wurde ausschließlich von mir, Stefanie Jegelka, erstellt. Es wurden keine weiteren Hilfsmittel oder Quellen außer den angegebenen benutzt.

Tübingen, den 28.11.2007, Stefanie Jegelka

Overview

Clustering problems, sometimes also formulated as graph cut problems, are often stated as instances of discrete combinatorial optimization problems. As such they are often NP-hard. Thus, people have developed heuristics and relaxations, though often without any theoretical guarantees on the solution. Here, we consider the clustering problem as a statistical learning problem and try to approximate the best partition of the entire underlying space. Stating the problem in the statistical setting opens ways to remedy the exponential runtime by modifying the optimization problem to simultaneously achieve statistical consistency of the algorithm. Consistency means that, as the number of samples increases, the quality of the returned partition will converge to the quality of the true global optimizer on the underlying space.

To achieve this goal, we present two approaches. First, we add a margin criterion to the objective to induce local robustness of the preferred partition. This leads to a mixed integer linear program that we also state in the context of flow algorithms.

Second, we restrict the set of candidate functions via neighborhood cells around seed nodes, and optimize the original criterion on this limited function space. The resulting algorithm, nearest neighbor clustering (NNC), is statistically consistent and does run in polynomial time by construction. The average runtime is improved via branch and bound and certain heuristics that still guarantee the optimal solution. We show that, despite its simplicity, NNC performs comparably to standard clustering algorithms on the training set and with respect to generalization.

Both approaches are motivated and discussed from a theoretical viewpoint as well as investigated in experiments.

Contents

Chapter 1

Introduction

Deep in concentration, Mr Biss investigates the cryptic data in front of him, revealing relations of various types. Where are communities in that graph? If he could just come up with a nice partition of the customers, into market segments for example. The boss would be impressed by successful product positioning and identified target markets. At the same time and some blocks ahead, Dr. G.N. stares at expression patterns of genes his student measured in a big microarray. Groups of co-expressed genes might give him insight into new functional relations. His colleague nearby racks his brains to find homologous sequences to study gene families, while the librarian downstairs wishes for a method to order the mass of articles on her computer by topic, maybe via key words.

All these people's problems share a common basis: group a set of given data points (that may be considered samples from a larger population) such that points within a group are "similar", and points from different groups have few things in common. In short, they are trying to solve a clustering problem, possibly with a statistical background. Clustering has a wide range of applications in areas such as image processing, layout of electrical circuits, biology, social sciences, psychology and network analysis.

In the following, we consider the given "training data" set $\{X_1, \ldots, X_n\}$ to be a sample from a distribution P on some underlying space $\mathcal{X}$. A similarity or distance function measures relations or "closeness" between two points. Stated like this, clustering is an unsupervised learning problem.

In this setting, it can be interpreted as a graph partitioning problem: The points X_i correspond to nodes in the graph, and edge weights between nodes define similarities. We then seek to cut the graph into parts such that there are few connections between the parts, but the nodes within a part are densely connected.

The properties of a "good" partition or clustering are commonly captured by a quality function Q (or Q_n) that assigns a quality value to each possible partition. The goal is then to find the partition optimizing this function.

A vast amount of research, from areas ranging from combinatorics, statistics, algorithm analysis, optimization, network and graph theory, to social sciences, has been devoted to algorithms, models, theoretical properties of clustering as well as definitions of a good partition and quality measures for clusterings. Hence we cannot enumerate all approaches here. The algorithms may be divided into hierarchical and partitional methods. Hierarchical clustering subsequently splits the data into smaller groups (divisive or top-down), or, bottom up, joins similar groups (agglomerative) until the desired number of clusters is reached. Partitional methods, which we consider in this report, do not create such a tree but split

the data at once. Alternatively, approaches to clustering can be categorized by the data model they assume. We will detail such a distinction in the next section.

1.1 Two approaches to clustering

In view of the data model, the clustering problem can be viewed from two perspectives. On the one hand, we consider the given sample of size n as an independent, separate data set, that means an instance of a certain problem, and hence try to find the best partition of those points by optimizing a quality function Q_n on this data set. This approach traditionally leads to a discrete optimization problem and combinatorial algorithms. On the other hand, we may view the given data as a sample from an underlying data space $\mathcal{X}$ endowed with a probability measure, and thus use the sample to estimate a good partition of the entire space. Ideally, clusters are distinct regions of high density. The quality of the continuous partition is measured by a function Q, and Q_n is its estimator on a finite sample. This is the approach taken by statistical learning theory. In this statistical setting, however, the clustering with the best quality on the discrete data set may not be the discrete correspondent of the optimal partition of the space. In other words, the optimizers of Q and Q_n are not equivalent. In the following, we take a closer look at both approaches.

1.1.1 The "discrete optimization" approach

The "discrete optimization" approach states clustering as an optimization problem on the fixed data set $\{X_1, \ldots, X_n\}$, in general a minimization of a particular quality function Q_n, possibly with additional constraints.

For many objective functions, especially those including a balance criterion on cluster sizes, the optimization becomes NP-hard [see e.g. Kannan et al., 2004, Shi and Malik, 2000, Wagner and Wagner, 1993]. The difficulty of the optimization is grounded in the number of possible partitions. The number of distinct assignments of n points to K clusters is exponential in n [Hastie et al., 2001, p. 461]:

$$\frac{1}{K!}\sum_{i=1}^{K} K(-1)^{K-i}\binom{K}{i} i^n.$$

If the behavior of the objective function for small changes in the assignment is not simple, as for purely additive criteria such as Mincut (see Section 1.2.1), then we must search through an exponential number of partitions.

To avoid exponential runtimes, people usually revert to heuristics or relaxations, trading runtime for an approximate, suboptimal solution. A common compromise is to use iterative greedy descents [Hastie et al., 2001, Ch. 14.3.5] like the k-means algorithm, guaranteed to find a local optimum but not necessarily the global minimizer. The behavior of k-means, for instance, is sensitive to the initialization [see e.g. Milligan, 1980, Peña et al., 1999]. Alternatively, relaxations of certain constraints can make the problem easier to solve, with a loss in the optimality of the solution. Spectral clustering is an example for such a relaxation [see e.g. von Luxburg, 2006]. For some variants of spectral clustering, certain worst-case guarantees can be proved [e.g. Kannan et al., 2004]. Spielman and Teng [1996] show approximation guarantees on bounded-degree planar graphs and finite element meshes. Nevertheless, for the RatioCut objective, there are examples where spectral clustering will not find a good partition [Guattery and Miller, 1995]. For many algorithms, no statements can be made about how far the approximate solution is to the optimal one.

Only simple objectives keep the exact optimization in P. Mincut, for example, can be cast as a Maxflow problem by duality theory. A number of algorithms solve the latter efficiently [see e.g. Papadimitriou and Steiglitz, 1982, Ch. 6, Ahuja et al., 1993, Ch. 6–8].

1.1.2 A glimpse on Statistical Learning Theory

Alternatively, the data may be seen as a sample from a larger distribution, as in Statistical Learning Theory (SLT). In the following, we give an overview of SLT with a focus on classification. For a detailed introduction, refer to Devroye et al. [1996], Vapnik [2001], for instance. The goal in SLT is to learn rules or estimators based on a finite collection of training examples rather than predetermined probability models. Hence, SLT applies to problems whose physics are difficult to model, and there is not sufficient experience for accurate and complete probability models [Nowak, 2007a].

Classification

In classification, we are given a sample $\{(X_1, Y_1), \ldots, (X_n, Y_n)\} \subseteq \mathcal{X} \times \{+1, -1\}$ of points X_i and their true labels Y_i. Those points are drawn i.i.d. from an unknown probability distribution $P(X, Y)$ on $\mathcal{X} \times \{+1, -1\}$. Our goal is to infer a function $f : \mathcal{X} \mapsto \{+1, -1\}$ that correctly predicts the labels of all points from $\mathcal{X}$. A loss function $L(f(X_i), Y_i)$ specifies the cost of a misprediction, for example the 0-1 loss [Vapnik, 2001, p. 19]

$$L(f(X_i), Y_i) = \begin{cases} 0 & \text{if } f(X_i) = Y_i \\ 1 & \text{otherwise.} \end{cases} \tag{1.1.1}$$

The overall expected risk or loss,

$$R(f) = \mathbb{E}_{(X,Y)}[L(f(X), Y] = \int L(f(X), Y) dP(X, Y), \tag{1.1.2}$$

indicates the quality of a partition f that we aim to optimize by choosing the best f from a hypothesis space $\mathcal{F}$. Formally, we seek to find

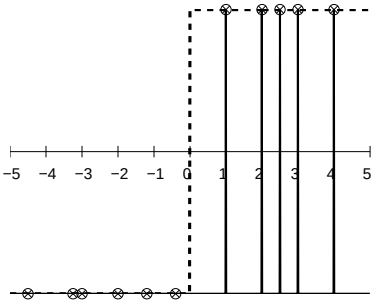
$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} R(f).$$

In more general terms, we define a quality function $Q : \mathcal{F} \to \mathbb{R}$ that measures the goodness of a candidate predictor, and then try to find the $f \in \mathcal{F}$ minimizing this criterion Q. In classification, the quality is measured by the expected rate of mispredictions, so $Q(f) = R(f)$.

Learning needs assumptions

For this inference of f, however, we need take some general principles into account [Bousquet et al., 2004]. Learning is only possible with assumptions. Without assumptions, the training points carry no information about the labels of future observations. If we fix the labels of the training points, we can still label all other points arbitrarily. Which such labeling is the best? This dilemma is summarized in the No Free Lunch Theorem. First, the future observations, for which labels will be predicted, must be related to past ones. Here, this means they stem from the same or distribution $P(X, Y)$ or at least a related one. Another important assumption is continuity: if X is similar to the training point X_i, then Y is expected to be

Figure 1.1.1: Example for overfitting. The X are uniformly distributed on the interval $[-5, 5]$, and the true labeling is -1 for $X < 0$ and $+1$ for $X \geq 0$ (dashed line). The training examples are marked by $\otimes$. A predictor f_n that only labels the $+1$ training examples positively (black line) will achieve zero empirical error but a true error of $R(f_n) = 0.5$, which is the worst possible. See the text for details.



similar to Y_i. Only if the assumptions hold is it possible to construct consistent algorithms, that is with more training data, the predictions approach the optimal ones. Yet with a finite data set, the predictions may be arbitrarily bad, if we can choose f to be any function. Hence and second, the ability to generalize requires specific knowledge, such as assumptions what the optimal classifier f^* looks like. We can encode this model for example in the way we define $\mathcal{F}$, restricting it only to "reasonable" candidates. The resulting algorithm works best on problems where these assumptions are actually met.

Minimize the risk

Once we have fixed the assumptions, how can we find a classifier with minimal expected error? Without knowing the exact joint distribution $P(X, Y)$, we cannot compute the risk functional $R(f)$ directly. Two solutions seem possible: estimate the joint distribution P from the training set, and derive the decision rule from this estimation, or use the training set to directly infer f. Since the former is usually more difficult than the direct design of a rule, we use the latter approach: "When solving a problem, try to avoid solving a more general problem as an intermediate step" [Vapnik, 2001, p. 30]. We hence estimate $R(f)$ by the empirical risk

$$\widehat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(X_i), Y_i) = Q_n(f). \tag{1.1.3}$$

The minimization of this estimate is the basis of the principle of Empirical Risk Minimization (ERM). More generally, we define an estimator Q_n of Q that works on a finite sample of n points, because we cannot compute Q directly.

The natural question to arise here is whether it makes a difference to optimize Q_n or Q. By the law of large numbers, $\widehat{R}(f)$ converges to $R(f)$ for any fixed f as n goes to infinity, like a mean to its expectation. But how much does the quality Q of f_n, an optimizer of Q_n from $\mathcal{F}$, differ from $Q(f^*)$, if f_n was found with a finite sample? The law of large numbers holds only for a fixed f, but we can choose any $f \in \mathcal{F}$ after seeing the data. And what about the speed and variance of convergence?

Let us first take a look why these questions may be important. If $\mathcal{F}$ is large, then we can find an f with zero empirical error for almost any training sample. Figure 1.1.1 shows an extreme example. Consider $\mathcal{X}$ to be the interval $[-5, +5]$, and the true labeling -1 for all points smaller than zero ($X < 0$) and $+1$ for all points greater or equal to zero. With the given training sample and $\mathcal{F}$ including all functions from $[-5, 5]$ to $\{-1, +1\}$, we might minimize $\widehat{R}$ by setting $f_n(X_i) = 1$ for all positive training examples, and $f_n(X) = -1$ for

all other X:

$$f_n(X) = \begin{cases} 1 & \text{for } X \in \{X_i \mid Y_i = 1, 1 \leq i \leq n\} \\ -1 & \text{otherwise.} \end{cases} \tag{1.1.4}$$

Then we have $\widehat{R}(f_n) = 0$, but $R(f_n) = 0.5$, which is as good as random guessing. The strategy of setting $f_n(X) = 1$ only for those X that are in the training sample and labeled positively will yield the correct classifier if all possible X are in the training sample. But with a finite sample and a large set of candidates, $\widehat{R}$ may be not enough to judge about the actual quality of a classifier. We can achieve zero training error, but the expected error is still large. Thus, a small empirical error cannot guarantee a small expected error. This phenomenon is termed overfitting. If $\mathcal{F}$ was restricted to piecewise constant functions, for instance, and still $\widehat{R}(f_n) = 0$, then $R(f_n)$ would decrease with significant probability. Let a be the smallest area around the positive training samples that is compatible with the minimal length $c_{\min}$ of a constant piece. If the points are well separated, then a is $c_{\min}$ times the number of positive training examples. Then $R(f_n) \leq 0.5 - a/10$. Very likely, a grows as n goes to infinity, and faster than the integral over all positive samples.

So what do we actually need for a reliable inference principle?

Consistency

What we wish to have is an inference principle that returns a predictor f_n whose quality converges to the optimal quality. To judge it, we require that its empirical quality approaches the quality of the true minimizer, $Q(f^*)$, in the limit. Then, in the limit, $Q_n(f_n), Q(f_n)$ and $Q(f^*)$ are the same: Q_n estimates the quality $Q(f_n)$ correctly, and f_n is as good as the true optimizer. Such a principle is consistent.

Definition 1 (Consistency)**.** (adapted from [Vapnik, 2001, p. 36]) The ERM principle, returning f_n for a sample of size n, is *consistent* for a set of functions $\mathcal{F}$ and a probability distribution $P(X, Y)$ if the sequences $Q(f_n)$ and $Q_n(f_n)$ converge in probability to the same limit:

$$Q(f_n) \xrightarrow[n\to\infty]{P} Q(f^*) \tag{1.1.5}$$

$$Q_n(f_n) \xrightarrow[n\to\infty]{P} Q(f^*). \tag{1.1.6}$$

Strictly speaking, if the convergence of the sequences is only in probability, then the principle is weakly consistent [Devroye et al., 1996, Def. 6.1]. A refinement of this definition excludes trivial cases. The principle is *nontrivially consistent* if the convergence (1.1.6) also holds for any subset $\mathcal{F}_c$ of $\mathcal{F}$ whose members have an error $Q(f)$ of at least c ($c \in \mathbb{R}$) [Vapnik, 2001, p. 37f].

Vapnik's "Key Theorem of Learning Theory" [Vapnik, 2001, Thm. 2.1] gives a necessary and sufficient condition for consistency: The empirical quality Q_n must converge uniformly (one-sided), that means for all functions in $\mathcal{F}$, to the expected quality Q:

$$\forall \varepsilon > 0 \quad \lim_{n\to\infty} P\{\sup_{f\in\mathcal{F}}(Q(f) - Q_n(f)) > \varepsilon\} = 0$$

If the difference between $Q_n(f)$ and $Q(f)$ is small for *all* $f \in \mathcal{F}$, then the minimizer of Q_n must still be decent with respect to Q.

How can we achieve uniform convergence, and how fast or reliable is this convergence?

Convergence Bounds

The empirical and true risk behave like a mean and its expectation. Concentration-of-measure inequalities bound the deviation between these two quantities. An application of Hoeffding's inequality to $Q = R$ and $Q_n = \widehat{R}$ yields [Bousquet et al., 2004, p. 177]

$$P\{|Q_n(f) - Q(f)| > \varepsilon\} \leq 2\exp\left(-2n\varepsilon^2\right).$$

This bound holds, however, only for one fixed function. For a simultaneous bound for *all* $f \in \mathcal{F}$ we consider the probability that the difference deviates for more than ε for any f. If $|\mathcal{F}|$ is finite, then this probability can loosely be bounded by the sum of the deviation probabilities for each of the $|\mathcal{F}|$ candidates, according to the rule $P(A \cup B) \leq P(A) + P(B)$. Hence, by this union bound, [Bousquet et al., 2004, p. 178]

$$P\{\sup_{f\in\mathcal{F}}(Q(f) - Q_n(f)) > \varepsilon\} \leq 2|\mathcal{F}|\exp\left(-2n\varepsilon^2\right) = 2\exp\left(\ln|\mathcal{F}| - 2n\varepsilon^2\right). \tag{1.1.7}$$

Setting the right hand side to δ, we can rewrite Equation (1.1.7) in terms of confidence intervals: With probability at least $1 - \delta$,

$$Q(f) \leq Q_n(f) + \sqrt{\frac{\ln|\mathcal{F}| + \ln(1/\delta)}{2n}}. \tag{1.1.8}$$

Thus, for convergence, the size of $\mathcal{F}$ must grow sub-exponentially in n. But what happens for an infinitely large function class? Then the complexity of $\mathcal{F}$ can take the place of its size. The complexity measures the richness of $\mathcal{F}$, for instance, how many different assignments on a finite sample can actually be realized by functions from $\mathcal{F}$. We will outline some complexity measures below. For the moment, let us denote the complexity of $\mathcal{F}$ by $\mathsf{C}(\mathcal{F})$. Complexity measures will be described below. If C is the growth function, the VC entropy or the annealed entropy, then, by a trick called symmetrization [Schölkopf and Smola, 2002, Ch. 5.5], one can show that (for classification) [Vapnik, 2001, Ch. 3.1], [Schölkopf and Smola, 2002, p. 138]

$$P\{\sup_{f\in\mathcal{F}}(Q(f) - Q_n(f)) > \varepsilon\} \leq 4\exp\left(\mathsf{C}(\mathcal{F}) - \frac{n\varepsilon^2}{8}\right)$$

or, with probability at least $1 - \delta$,

$$Q(f) \leq Q_n(f) + \sqrt{\frac{8}{n}\left(\mathsf{C}(\mathcal{F}) + \ln\frac{4}{\delta}\right)}. \tag{1.1.9}$$

This equation shows that the complexity of $\mathcal{F}$ is inversely related to the confidence we can have in $Q_n(f)$, and that $\lim_{n\to\infty}\mathsf{C}(\mathcal{F})/n = 0$ is a condition for uniform convergence[1]. So the complexity must grow more slowly than n, or the size of $\mathcal{F}$ must be sub-exponential in n. In the case of overfitting, $Q(f_n)$ and $Q_n(f_n)$ differ a lot because $\mathsf{C}(\mathcal{F})$ is big relative to n. Then a small empirical error does not imply a small expected error. A restriction of the complexity of $\mathcal{F}$ may thus remedy the problem of overfitting.

We can also view these bounds from a different perspective: "minimizing the risk on a smaller set of functions requires fewer observations" [Vapnik, 2001, p. 66]. For a deviation of at most ε with probability $1 - \delta$, we need at least

$$n \geq \frac{8}{\varepsilon^2}\left(\ln\frac{4}{\delta} + \mathsf{C}(\mathcal{F})\right)$$

[1] This is actually the condition for two-sided convergence, one-sided convergence has a slightly weaker condition (see Vapnik [2001, Ch. 2.4] for details).

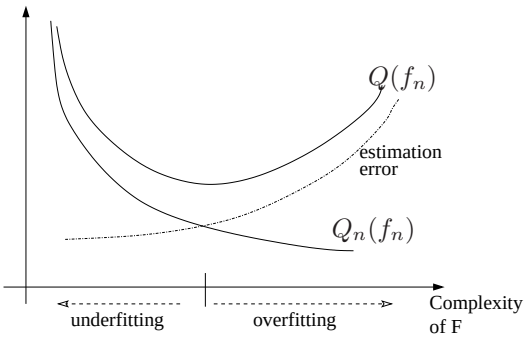


Figure 1.1.2: Over- and underfitting dependent on the complexity of $\mathcal{F}$. If $\mathcal{F}$ is not suited for the problem, then no function in $\mathcal{F}$ can fit the data well, and both $\widehat{R}$ and R are high. If $\mathcal{F}$ is very rich, we can find an explanation for any data, but this may not be a good fit for the entire space, because the estimation error or confidence interval increases. (Figure adapted from [Nowak, 2007b, Lec. 3].)

samples. So the sample complexity n must grow as $O(\mathsf{C}(f))$ for fixed ε and δ. Hence, a function class with restricted complexity yields a more reliable predictor, provided that the class still contains reasonable candidates, meaning that the assumptions encoded in $\mathcal{F}$ match the problem at hand.

Before we explore how the complexity restriction may be implemented in practice, let us take a step back and see where our algorithm can fail. Let Q^* denote the Bayes risk, the minimal achievable risk if all possible functions are taken into account. Note that the corresponding predictor need not be in $\mathcal{F}$. The best we can achieve within $\mathcal{F}$ is $Q(f^*) \geq Q^*$. So the deviation of the risk of our outcome f_n compared to Q^* may be divided into two parts [Nowak, 2007b, Lec. 5]:

$$Q(f_n) - Q^* = \underbrace{(Q(f_n) - Q(f^*))}_{\text{estimation error}} + \underbrace{(Q(f^*) - Q^*)}_{\text{approximation error}} .$$

The estimation error is due to the randomness of the training sample, and measures how good the prediction is with respect to the best prediction in $\mathcal{F}$. The approximation error measures the appropriateness of our assumptions: how much do the restrictions of $\mathcal{F}$ impair the ability to model the problem? The richer $\mathcal{F}$ is, the smaller is the approximation error, but, as the bounds above demonstrate, the larger may be the estimation error. Thus, we need to find a tradeoff between the minimization of those two errors, regarding the complexity of $\mathcal{F}$.

How can this tradeoff be implemented in practice, and how can one measure complexity?

Strategies to avoid overfitting

There are two basic approaches to limiting the complexity of the class of candidate functions [Nowak, 2007b, Lec. 3]:

1. Directly restrict the size or complexity of $\mathcal{F}$. This restricts the estimation error by the bounds above, but also sets a lower bound to the approximation error.

2. Simultaneously minimize the complexity and empirical error, by optimizing a modified term: $f_n = \operatorname{argmin}_{f \in \mathcal{F}} \{Q_n(f) + \lambda \mathsf{C}(f)\}$. Such complexity penalization methods seek to balance the tradeoff between approximation and estimation error. The additional term is sometimes also called a regularizer. The parameter λ usually shrinks with n in practice and may be chosen by hold-out validation.

For clustering, we tried both approaches, a margin-based complexity penalization method (Chapter 2), and a (data-dependent) restriction of the class of admissible functions (Chapter 3).

In the following, we will mention a selection of methods implementing those two strategies. The method of sieves [Grenander, 1981] restricts the function space $\mathcal{F}_n$ according to the number n of samples available, with $|\mathcal{F}_{i-1}| \leq |F_i|$. The predictor f_n is the optimizer of Q_n from $\mathcal{F}_n$: $f_n = \operatorname{argmin}_{f \in \mathcal{F}_n} Q_n(f)$. The class $\mathcal{F}_n$ is, however, chosen independently of the training data. Data-adaptive model spaces are better adapted to the distribution of the data.

The principle of Structural Risk Minimization (SRM) [e.g. Vapnik, 2001, Sec. 4.1] relies on the construction of a hierarchical structure $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots$ of function classes of growing complexity. Within each class, we choose the optimizer of the empirical quality Q_n. Of those optimizers, we choose the one that yields the best value for a generalization bound like (1.1.9). This strategy can also be interpreted as a complexity penalization method: $f_n = \operatorname{argmin}_{f \in \mathcal{F}_i} \min_i Q_n(f) + \operatorname{pen}(\mathcal{F}_i)$, where pen is related to the confidence interval [Bousquet et al., 2004, p. 173].

Further complexity penalization methods include the minimum description length principle (MDL) [e.g. Vapnik, 2001, p. 106], where complexity is measured by the number of bits needed for the description of the predictor, via the compression coefficient. This coefficient depends on the number of candidate functions and the necessary corrections. The margin of Support Vector machines is another measure of complexity [Schölkopf and Smola, 2002, Sec. 7.2], and its maximization is also related to a restriction of the complexity. A connection may also be drawn to Bayesian methods, if the prior is interpreted as the complexity penalization [Nowak, 2007b, Lec. 3].

Note that the choice of the regularizer encodes prior knowledge and assumptions just like the restriction of $\mathcal{F}$.

Measures of Complexity

Over time, a variety of measures of the complexity of a function class have been crafted. The following outline of some representatives is mainly based on the descriptions by Schölkopf and Smola [2002, Ch. 5] and Vapnik [2001, Ch. 2].

For classification, consider the number of ways a sample of n points can be partitioned by functions in $\mathcal{F}$. Of course this number also depends on the particular sample. Denote by $\mathcal{N}(\mathcal{F}, \mathcal{Z}_n)$ the number of ways the sample $\mathcal{Z}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ can be labeled by functions in $\mathcal{F}$. The maximum such number over all samples of size n is denoted by the shattering coefficient $\mathcal{N}(\mathcal{F}, n) = \max_{\mathcal{Z}_n} \mathcal{N}(\mathcal{F}, \mathcal{Z}_n)$. If $\mathcal{F}$ contains all possible functions, then this number is 2^n. If $\mathcal{F}$ is restricted, then there is a limit after which $\mathcal{N}(\mathcal{F}, n)$ does not grow exponentially in n any more. Hence its logarithm measures complexity in a similar way as $\ln |\mathcal{F}|$.

For the 0-1-loss (Eq. (1.1.1)), we can also get the number of possible partitions as follows: Each function $f \in \mathcal{F}$ has a loss vector $\xi_f = (L(f(X_1), Y_1), \ldots, L(f(X_n, Y_n)))$ on the sample $\mathcal{Z}_n$. The number of different loss vectors for the functions in $\mathcal{F}$ equals $\mathcal{N}(\mathcal{F}, \mathcal{Z}_n)$.

The expected logarithm of $\mathcal{N}(\mathcal{F}, \mathcal{Z}_n)$ is the VC entropy $H_{\mathcal{F}}$ of $\mathcal{F}$:

$$H_{\mathcal{F}}(n) := \mathbb{E}_{\mathcal{Z}_n}[\ln \mathcal{N}(\mathcal{F}, \mathcal{Z}_n)].$$

It measures complexity analogous to the logarithm of the size of $\mathcal{F}$, and we may thus replace $\mathsf{C}(\mathcal{F}) = H_{\mathcal{F}}(n)$ in the bound (1.1.9) [Schölkopf and Smola, 2002, p. 138]. Note that the VC entropy depends on the distribution of (X, Y) and is thus difficult to compute. A similar measure is the annealed entropy,

$$H_{\mathcal{F}}^{\text{ann}}(n) = \ln \mathbb{E}_{\mathcal{Z}_n}[\mathcal{N}(\mathcal{F}, \mathcal{Z}_n)].$$

Since it upper bounds the VC entropy, it can also be used as $\mathsf{C}(\mathcal{F})$ above. More finely-grained measures, based on ε-covers of $\mathcal{F}$, are entropy and covering numbers [see Schölkopf and Smola, 2002, Sec. 12.4.2].

A distribution-independent upper bound on the VC and annealed entropy is the growth function, the logarithm of the shattering coefficient:

$$G_{\mathcal{F}}(n) = \max_{\mathcal{Z}_n} \ln \mathcal{N}(\mathcal{F}, \mathcal{Z}_n).$$

We can use it as $\mathsf{C}(\mathcal{F})$ above as well. If $\mathcal{F}$ contains all functions, then the growth function is $n \ln 2$ for all n. It has been proved that for a restricted $\mathcal{F}$ though, there is a maximal number $n = h$ after which the growth function does not grow linearly in n any more, because not all possible partitions may be realized by functions from $\mathcal{F}$ any more. This number h is the VC dimension of $\mathcal{F}$. For $n > h$, the growth function only increases logarithmically in n: $G_{\mathcal{F}}(n) \leq h \left(\ln \frac{n}{h} + 1\right)$ [Schölkopf and Smola, 2002, p. 141].

Shattering coefficients and related measures constitute only one possibility to define complexity. The Rademacher average, for instance, captures the ability of a function class to fit random noise. The richer the function class, the better it can fit or "explain" any given data. A complete definition and bounds may be found in Bousquet et al. [2004, Sec. 5.2].

Another viewpoint is taken by compression and leads to the Minimum description length principle. $\mathcal{F}$ is considered as a set of function tables (codebook) and the labels of a sample as a string. The number of bits required to code such a string via a function in $\mathcal{F}$, relative to the string's length, is the compression coefficient [Vapnik, 2001, Sec. 4.6.1]. The encoding includes the number of the function plus a correction if there is no function in $\mathcal{F}$ that accurately codes the labels. Bounds on the risk or test error can be derived in terms of the compression coefficient [see e.g. von Luxburg, 2001, Sec. IV.1.2].

What is different in Clustering?

The above introduction focused on classification as a learning problem. For clustering, some aspects are different.

Firstly, in clustering, no labels but only the mere data points are given for training. That means Q does not correspond to the risk of misclassification, if there can be such a thing for clustering at all. Instead, it measures the quality of the cut by other characteristics that one might wish a good clustering to have, for example, that the connectedness within clusters should be large, but small across groups. Some quality measures for clustering will be outlined in Section 1.2.1. Q_n then does not sum up some point-wise measure such as errors, but may involve more complex nonlinear terms. Thus, Q_n does not obviously converge to Q like a mean to its expectation, so the concentration inequalities may not be directly applicable as for the 0-1-loss.

The lack of given labels in the definition of Q for classification and clustering may lead to some differences between consistency and stability of labelings for clustering, contrary to classification. Algorithmic stability is often defined as a measure of how much the outcome of an algorithm changes if one sample point is replaced or removed from the training data. The change may be defined either with respect to the expectation of a term involving the quality measure (such as the quantities $\mathbb{E}\left[|L(f_n(X), Y) - L(f_n^{\text{new}}(X), Y)|\right]$ [Bousquet and Elisseeff, 2002] or $\mathbb{E}\left[|Q_n(f_n) - Q_{n-1}(f_n^{\text{new}})|^2\right]$ [Rakhlin et al., 2005] or $|Q(f_n) - Q(f_n^{\text{new}})|$ [Bousquet and Elisseeff, 2002]) or with respect to the actual labelings $f_n(X)$ (for instance, $\|f_n - f_n^{\text{new}}\|_\infty$ [Bousquet and Elisseeff, 2002] or $P\{f_n(X) \neq f_n^{\text{new}}(X)\}$ [Kearns and Ron, 1997]). For classification, there is a close relationship between those two viewpoints. If R_n is significantly smaller than 0.5, then the replacement or removal of one sample point will

only lead to a similar quality if the new predictor f_n^{new} labels points almost exactly like f_n. A complete relabeling will lead to a much larger empirical error, because most of the sample remains the same, and the labels for those points are fixed. For clustering, this is not the case. There might be an almost equally good grouping for which a large number of points is relabeled, so f_n and f_n^{new} differ significantly. Uniform stability, defined with respect to the loss, is closely related to the convergence of $Q_n(f)$ to $Q(f)$ [e.g. Schölkopf and Smola, 2002, Thm. 12.3]. For clustering, however, consistency and fast convergence of $Q_n(f)$ to $Q(f)$ do not say anything about the stability measured on $f_n(X)$ directly. There may be several, very different almost-minimizers of the quality function, and hopping from one to the other will not affect the quality value significantly. This may happen if there is symmetry in the data [Ben-David et al., 2006], as in Figure 1.1.3. Ben-David et al. [2007] show that if there are multiple optimal solutions, then the difference between two clusterings returned by the k-means algorithm for different input samples from the same distribution does not converge to zero as $n \to \infty$. Rakhlin and Caponnetto [2006] study the number of sample points that may be replaced in k-means for the solution to remain stable: if there is a unique optimum for Q_n, then all points can be replaced, and otherwise $\Omega(\sqrt{n})$. For the ERM principle with supervised learning, Caponnetto and Rakhlin [2006] show that under certain assumptions the diameter of the set of ε-minimizers of Q_n with $\varepsilon = o(1/\sqrt{n})$ goes to zero as the sample grows, and hence it becomes less and less likely that the algorithm will jump to a different part of $\mathcal{F}$, changing the predictor significantly. Note however, that they only consider the case that sample points are added, and not that the entire training sample is replaced. Referring to "supervised ERM", they also cite Lee et al. [1996] that the existence of different minimizers of Q seems to be a property of difficult learning problems.

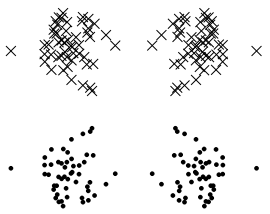


Figure 1.1.3: Symmetric clusters: At least two partitions of this data set may optimize a clustering criterion: Cutting the set horizontally or vertically in the middle. If labels are given (different for x and o-points), then there is only one optimal labeling.

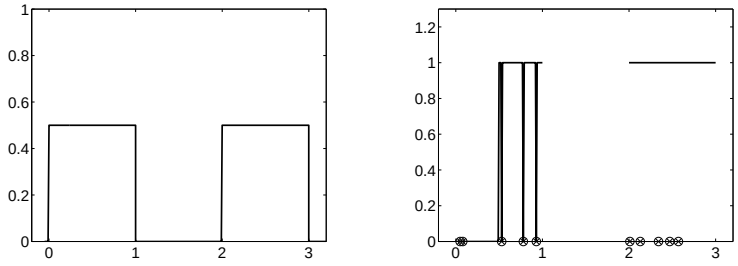
Note however, that consistency as we defined it above for clustering, implies, like in classification, that in the limit, we will get a "globally good" partition, as measured by Q. "Overfitting" can happen in clustering as in classification. A simple example is the following [Bubeck and von Luxburg, 2007]: Let $\mathcal{X} = [0,1] \cup [2,3]$, that means two "obvious" clusters $[0,1]$ and $[2,3]$, and the probability be the normalized Lebesgue measure on $\mathcal{X}$ (Figure 1.1.4(a)). The similarity function $s : \mathcal{X} \times \mathcal{X} \to [0,1]$ indicates the similarity between two points as follows:

$$s(X_i, X_j) = \begin{cases} 1 & \text{if both } X_i, X_j \in [0,1] \text{ or both } X_i, X_j \in [2,3] \\ 0 & \text{otherwise.} \end{cases}$$

A clustering f assigns the points $X_i \in \mathcal{X}$ to clusters $\mathcal{C}_0$ and $\mathcal{C}_1$. We measure cluster quality via the between-cluster-similarity which covers the similarities of points from different clusters:

$$Q(f) = \int_{X \in \mathcal{C}_0} \int_{Y \in \mathcal{C}_1} s(X,Y) dP(Y) dP(X)$$

$$Q_n(f) = \frac{1}{n(n-1)} \sum_{X_i \in \mathcal{C}_0 \cap \mathcal{Z}_n} \sum_{X_j \in \mathcal{C}_1 \cap \mathcal{Z}_n} s(X,Y).$$



(a) Distribution of the data on $\mathcal{X}$

(b) Sample points $\{X_1, \ldots, X_{10}\}$ from $\mathcal{X}$ with overfitted f_n (see text)

Figure 1.1.4: Overfitting example for clustering

In the estimator Q_n, we only sum over sample points from the training sample $\mathcal{Z}_n = \{X_1, \ldots, X_n\}$. As function space $\mathcal{F}$ we allow any measurable partition. We may also add the constraint that the cluster sizes should not be smaller than a fixed $\varepsilon < 0.5$. An optimizer of Q on $\mathcal{X}$ is

$$f^*(X) = \begin{cases} 0 & \text{if } X \in [0,1] \\ 1 & \text{if } X \in [2,3], \end{cases}$$

with $Q(f^*) = 0$. Define another partition f_n, illustrated in Figure 1.1.4(b):

$$f_n(X) = \begin{cases} 0 & \text{if } X \in \mathcal{Z}_n \cap [0,1] \\ 1 & \text{if } X \in \mathcal{Z}_n \cap [2,3] \\ 0 & \text{if } X \in [0,0.5] \setminus \mathcal{Z}_n \\ 1 & \text{if } X \in (0.5,1] \setminus \mathcal{Z}_n \text{ or } X \in [2,3]. \end{cases}$$

It is straightforward to compute that $Q_n(f_n) = 0$, so f_n is a minimizer of Q_n. However, $Q(f_n) = 1/16$ for any finite n, so $Q(f_n) \nrightarrow Q(f^*)$. Therefore, for such a rich function class $\mathcal{F}$, we may get a globally bad clustering even with large training samples.

1.1.3 SLT to remedy NP-hardness

The two previous sections showed two different approaches to clustering: combinatorial optimization and statistical learning theory. The former seeks to optimize Q_n, an often NP-hard problem. From the perspective of SLT, this is usually not the best thing to do, because it corresponds to allowing an exponentially large and rich $\mathcal{F}$. Then, $Q(f_n)$ and $Q(f^*)$ may differ greatly, so we overfit. In addition, the NP-hardness is usually remedied by heuristics and relaxations that often do not provide any theoretical guarantees on the solution. Instead, we may simplify the problem in the flavor of SLT by reducing the complexity of $\mathcal{F}$ or modifying the selection of functions by a complexity criterion. In the following, we attempt two such approaches: (i) modifying Q_n by the addition of a margin (Chapter 2), and (ii) restricting $\mathcal{F}$ to a polynomial size, such that the search through all candidate functions is feasible in polynomial time (Chapter 3).

1.2 Ingredients of clustering algorithms

In the following, we outline some basic ingredients for combinatorial clustering algorithms. One basic problem is the definition of a good clustering. To this end, a vast number of clustering quality functions have been introduced. We describe a small selection of such criteria in Subsection 1.2.1. To apply a graph-based algorithm on data given in coordinates, we must first construct a graph that represents the similarity structure of the data. Subsection 1.2.2 summarizes several possibilities to construct similarity graphs from such data. Subsection 1.2.3 mentions some further questions.

Before proceeding, let us review some notation. We are seeking K clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K \subseteq V$ in a similarity graph $G = (V, E)$ with nodes V and edges E that represents the n sample points. The function w describes the edge weights: $w(X_i, X_j)$ is the weight of edge (X_i, X_j), and $w(\mathcal{C}_i, \mathcal{C}_j)$ is the sum of the weights of the edges connecting $\mathcal{C}_i$ and $\mathcal{C}_j$. As usual in set notation, $\overline{\mathcal{C}_i} = V \setminus \mathcal{C}_i$ is the complement. All the notation is summarized again in the Appendix.

1.2.1 Quality functions for clustering

There is no general objective measure unifying all ideal properties of a clustering. Depending on the data set and the specific application at hand, criteria for a "good" or "reasonable" clustering may vary. In addition, conceptual and algorithmic issues come into play. Some intuitive properties are encountered repeatedly in the definitions of clusters in a data set:

DIS The similarity of points from different clusters should be low. By the inverse relation of distance and similarity, clusters should be far apart. As edge weights represent similarities, this means the sum of the edge weights between clusters should be low.

CLO Points in the same cluster should be similar or close to each other. This criterion can be captured by the sum of the similarities within the clusters.

BAL Clusters should have at least a minimal size or should all have similar sizes (balancing criterion).

In the light of probability distributions underlying the data, the above criteria mean that between clusters, there should be a region of low probability density, whereas the density within the clusters should be significantly bounded away from zero and, integrated over the region of the cluster, about equal for all clusters.

Here, we assume to have K disjoint clusters $\mathcal{C}_i$, $V = \dot{\bigcup}_{i=1}^{K} \mathcal{C}_i$.

Mincut

The Minimum Cut criterion captures the dissimilarity criterion (DIS) of our list above via the sum of between-cluster similarities. It is the sum of the edge weights between clusters, that means the sum of the edges that are cut when the graph is partitioned into the clusters:

$$\text{Mincut}(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \sum_{i=1}^{k} w(\mathcal{C}_i, \overline{\mathcal{C}_i}).$$

Finding the minimum cut is a relatively easy combinatorial problem, and there exist a number of efficient polynomial-time algorithms [see e.g. Stoer and Wagner, 1997, and references

therein]. It can efficiently be solved via its dual problem, the Maximum Flow problem [Papadimitriou and Steiglitz, 1982, Sec. 6.1]. A number of such flow algorithms are presented in Ahuja et al. [1993, Ch. 7], among them one with a complexity of $O(n^2\sqrt{|E|})$.

Limited to between-cluster edges, the Mincut criterion often favors to cut off a single node. This tendency, however, contradicts the balance criterion of cluster sizes (BAL). Hence, a number of others criteria have been designed to remedy this lack.

Bring in cluster sizes: RatioCut and Ncut

Two criteria that combine between-cluster similarities (DIS) as well as the size of the clusters (BAL) are RatioCut and Normalized Cut (Ncut). They sum up the ratio of the two criteria for each cluster.

RatioCut, first introduced by Hagen and Kahng [1992], measures cluster size by the number of nodes in the cluster:

$$\text{RatioCut}(\mathcal{C}_1,\ldots,\mathcal{C}_K) = \sum_{i=1}^{K} \frac{w(\mathcal{C}_i,\overline{\mathcal{C}_i})}{|\mathcal{C}_i|}.$$

Shi and Malik [2000] crafted an analogous criterion, Ncut, where cluster size is determined by the volume, that is the accumulated degrees of the nodes in a cluster:

$$\text{Ncut}(\mathcal{C}_1,\ldots,\mathcal{C}_K) = \sum_{i=1}^{K} \frac{w(\mathcal{C}_i,\overline{\mathcal{C}_i})}{\text{vol}(\mathcal{C}_i)}$$

For both criteria, note that, under the constraint that $\sum_i x_i$ should be constant, the sum $\sum_{i=1}^{K} 1/x_i$ is minimal if all x_i equal. Thus, both criteria aim to balance cluster sizes while simultaneously punishing to cut through densely connected parts of the graph.

The Ncut objective has an additional interpretation with respect to random walks on graphs [Meila and Shi, 2001]. It is the probability of transferring from one cluster to the other: $\text{Ncut}(\mathcal{C},\overline{\mathcal{C}}) = p(\mathcal{C}|\overline{\mathcal{C}}) + p(\overline{\mathcal{C}}|\mathcal{C})$, where $p(\mathcal{C}|\overline{\mathcal{C}})$ is the probability of jumping from a node in cluster $\mathcal{C}$ to one in $\overline{\mathcal{C}}$ when starting in the stationary distribution. This means we favor a partition in which we remain longest in one cluster by expectation.

Despite their practical applicability, there is a drawback to the ratio criteria. The inclusion of the balance criterion renders them NP hard. Wagner and Wagner [1993] show that an additional criterion on cluster sizes ($|\mathcal{C}_i| \geq f(|V|)$ for certain functions f) puts the cut problem in NP. Papadimitriou proved Ncut to be NP-complete [Shi and Malik, 2000].

Nevertheless, the relaxed versions of the ratio criteria are solved efficiently by spectral clustering (see von Luxburg [2006] for details about spectral clustering). RatioCut leads to unnormalized spectral clustering, and Ncut to the normalized version. Even though these algorithms are widely used in practice, there is no general guarantee on the distance of their solution to the optimum. Several other relaxations exist [von Luxburg, 2006], but there is no efficient approximate balanced graph cut with a goodness up to a constant factor. In fact, the approximation problem itself is NP hard [Bui and Jones, 1992].

The ratio criteria are closely related to graph-theoretic properties. Consider $K = 2$ clusters $\mathcal{C}$ and $\overline{\mathcal{C}}$. Then Lovász [1993] defines the conductance for an unweighted graph as the Ncut:

$$\Phi = \min_{\mathcal{C}} \text{vol}(G)\frac{w(\mathcal{C},\overline{\mathcal{C}})}{\text{vol}(\mathcal{C})\,\text{vol}(\overline{\mathcal{C}})} = \min_{\mathcal{C}} \frac{((\text{vol}(\mathcal{C}) + \text{vol}(\overline{\mathcal{C}}))w(\mathcal{C},\overline{\mathcal{C}})}{\text{vol}(\mathcal{C})\,\text{vol}(\overline{\mathcal{C}})}.$$

He summarizes results about relations to the eigengap of the graph Laplacian and an approximation of Φ via multicommodity flows, close to a factor of $O(\log n)$. In the literature,

the above definition is also known as the conductance of the cut $(\mathcal{C}, \overline{\mathcal{C}})$, whereas the conductance of a graph is defined only as the larger of the two summands [Rubinfeld, 2006], as in Bollobás [1998], Chung [1994], for example:

$$h_G = \min_{\mathcal{C}} \frac{|E(\mathcal{C}, \overline{\mathcal{C}})|}{\min(\mathrm{vol}(\mathcal{C}), \mathrm{vol}(\overline{\mathcal{C}}))},$$

where $E(\mathcal{C}, \overline{\mathcal{C}})$ is the set of edges between $\mathcal{C}$ and $\overline{C}$, even for weighted graphs. Hence the numerator differs from the Ncut summand for weighted graphs. This ratio is also referred to as Cheeger constant. It can be bounded by the second smallest eigenvalue λ_2 of the normalized Laplacian, $2h_G \geq \lambda_2 \geq h_G/2$ [Chung, 1994]. The corresponding eigenvector is used for the partition in spectral clustering. The isoperimetric number [Chung, 1994] is the analogue for RatioCut, related to the eigenvalues of the unnormalized Laplacian:

$$h'_G = \min_{\mathcal{C}} \frac{|E(\mathcal{C}, \overline{\mathcal{C}})|}{\min(|\mathcal{C}|, |\overline{\mathcal{C}}|)}.$$

Within-cluster similarity

The minimization of the cut and maximization of cluster volumes favored by Ncut simultaneously maximizes within-cluster-similarities (CLO). This is easy to see by rewriting

$$\sum_{X_i, X_j \in \mathcal{C}} w(X_i, X_j) = \sum_{X_i \in \mathcal{C}, X_j \in V} w(X_i, X_j) - \sum_{X_i \in \mathcal{C} X_j \in \overline{\mathcal{C}}} w(X_i, X_j) = \mathrm{vol}(\mathcal{C}) - w(\mathcal{C}, \overline{C}). \quad (1.2.1)$$

There are, however, also direct criteria to optimize the similarities within clusters by minimizing distances. The Within-sum-of-squares (WSS) sums the (Euclidean) distance of the points in a cluster to the respective cluster center, punishing the within-cluster scatter.

$$\mathrm{WSS}(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \frac{1}{|V|} \sum_{i=1}^{K} \sum_{X_j \in \mathcal{C}_i} \|X_j - c_i\|^2,$$

where $c_i = 1/|\mathcal{C}_j| \sum_{X_j \in \mathcal{C}_i} X_j$ is the center of cluster $\mathcal{C}_i$. The resulting partitions are Voronoi tessellations around the centers. The standard algorithm to optimize WSS is the k-means algorithm [see e.g. Hastie et al., 2001, Sec. 14.3.1]. Analogous criteria replace the centers by medians, such as the objective of the k-medians algorithm. Other distances lead to clusters of different shapes.

Both the between- and within-cluster similarities (BW) are integrated in ratios by the MinMaxCut criterion by Ding et al. [2001], considering criteria DIS and CLO:

$$\mathrm{BW}(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \sum_{i=1}^{K} \frac{w(\mathcal{C}_i, \overline{\mathcal{C}_i})}{\sum_{X_s, X_t \in \mathcal{C}_i} w(X_s, X_t)}.$$

Its solutions resemble those of Ncut, because of the correspondence expressed in Equation (1.2.1).

1.2.2 Similarity graphs: from coordinate data to graphs

Some of the criteria above refer to clusters as results of a graph cut, and thus lead to graph cut algorithms. Similarity graphs make such algorithms applicable to coordinate data.

For their construction from coordinate data, we assume to be given a similarity function $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$. One such function is the Gaussian kernel:

$$s(X_i, X_j) = \exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right). \tag{1.2.2}$$

The data points make the nodes of the graph. We can connect them in a number of ways [see e.g. von Luxburg, 2006]. One possibility is the fully connected graph, where the edge between X_i and X_j has weight $s(X_i, X_j)$. For computational and other reasons, a restriction to a subset of these edges makes sense. For an ε-neighborhood graph, a node only retains the edges to nodes at a distance of less than ε, usually with unit edge weight. For a k-nearest neighbor graph, we set edges from a node to its k nearest neighbors, usually with the edge weight defined by s. In a mutual k-nearest neighbor graph, an edge (X_i, X_j) only exists if X_i is among the k nearest neighbors of X_j and vice versa.

The construction of any of these similarity graphs includes parameters: the similarity function and its parameters (such as the width σ), the neighborhood range ε or the number of neighbors k. The appropriate choice of these parameters may greatly influence the result and is another field of study.

In the experiments in Chapters 2 and 3, we mostly use k-nearest neighbor graphs and the Gaussian kernel as a similarity function.

1.2.3 Other questions

Important questions in clustering are the definition of a "good" clustering, the corresponding quality criterion, its efficient optimization, and theoretical guarantees of the algorithm. Furthermore, the choice of the parameters for similarity graphs raises several questions. In addition, research has been devoted to the question of how to properly evaluate a clustering or clustering algorithm. Another important question is the number K of clusters we assume in the data. The choice of K has been related to the eigengap of the graph Laplacian or stability, to name two criteria [cf. references in von Luxburg, 2006]. In the following, we assume that K has been fixed in advance.

Apart from these choices, algorithmic questions arise: what is a good and efficient method to minimize the objective? Approaches range from constrained optimization problems and their relaxations, greedy approaches, and exact combinatorial methods such as branch and bound, to name a few.

Chapter 2

First approach: Margin

One strategy to avoid overfitting is the introduction of a "penalty term" to make the algorithm prefer specific types of functions. Here, we favor functions that are "robust" to perturbations in the training data: if the edge weights are perturbed by a limited amount, then the chosen partition will still be a good one. This approach is similar to the concept of a margin for Support Vector Machines, seen from a robustness point of view. We will define several variations of margins on graphs and illustrate their behavior on toy examples: the alternatives behave roughly similar, but normalization can make a difference. In addition, the margin defines equivalence classes of partitions.

An inclusion of the margin in the objective results in a mixed integer linear program. Motivated by the Maxflow-Mincut duality, we will solve it via a network flow approach.

For simplicity, we focus on bipartitions ($K = 2$) in this chapter. We start with a short introduction to the concept of margins for linear classifiers in Section 2.1, before we define some margins for graph cuts in Section 2.2. Section 2.3 introduces the resulting optimization problem and its dual. Some experiments are described in Section 2.4, followed by a general discussion of the approach in Section 2.5.

2.1 Margins: what and why

The concept of a margin is one interpretation of a regularizer or complexity penalizer that is included in the objective, as suggested in Section 1.1.2. Margins are most well-known in the context of linear classifiers, leading to Support Vector Machines (SVMs). Hence, for an introduction, we will first focus on margins of linear classifiers, in the case when the data is separable. An extension to the non-separable case are soft margin hyperplanes, described in Schölkopf and Smola [2002, Sec. 7.5].

For a linear classifier, the geometrical margin is the distance of the separating hyperplane to any sample point. Let the hyperplane be $h_{w,b} = \{X \in \mathcal{X} \mid \langle w, b\rangle + b = 0\}$. The geometrical margin is then formally defined as [Schölkopf and Smola, 2002, Def. 7.2]

$$\rho(w, b) = \min_{X_h \in h_{w,b}, 1 \leq i \leq n} \|X_i - X_h\| = \min_i \frac{Y_i(\langle w, X_i\rangle + b)}{\|w\|}.$$

The large margin principle suggests to choose the hyperplane with the largest margin. For canonical hyperplanes ($\min_i |\langle w, X_i\rangle + b| = 1$), the margin is $\rho = 1/\|w\|$. Thus, we seek to minimize the norm of w for regularization. This is the basis of SVMs.

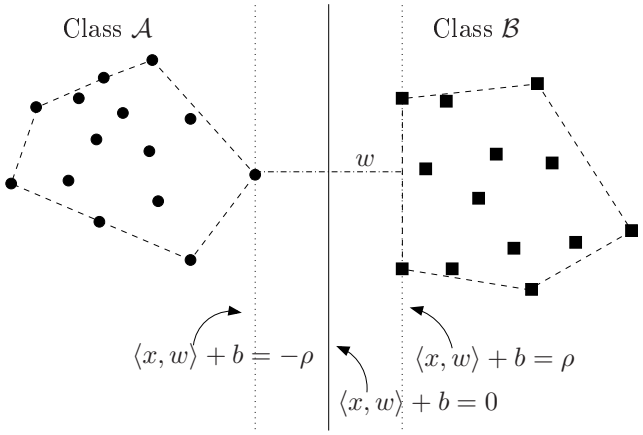


Figure 2.1.1: Finding the maximum margin is equivalent to finding the closest points in the convex hulls of the classes. (Figure adapted from [Bennett and Bredensteiner, 2000])

More general is the definition of a functional margin of a predictor f, where we threshold at zero [Cristianini and Shawe-Taylor, 2000, p. 159]. The margin of one example (X_i, Y_i) is $\rho(X_i) = Y_i f(X_i)$. If $\rho(X_i) > 0$, then the point is classified correctly. The margin of f is the minimal margin of the sample $\mathcal{Z}_n$:

$$\rho(f, \mathcal{Z}_n) = \min_{X_i \in \mathcal{Z}_n} \rho(X_i).$$

For linear classifiers, this is equivalent to the geometrical margin but without normalization by $\|w\|$.

The geometrical margin may also be seen from a dual viewpoint: finding the maximum margin between two classes $\mathcal{A}$, $\mathcal{B}$ is equivalent to finding the two closest points, where one is in each convex hull of one class (derived by Bennett and Bredensteiner [2000] from KKT criteria, Zhou et al. [2002] via a geometric approach). Figure 2.1.1 illustrates the equivalence. The separating hyperplane is orthogonal to the line connecting the two closest points from $\mathcal{A}$ and $\mathcal{B}$, it bisects this line in the middle.

Alternatively, one can view the situation from the perspective of supporting hyperplanes. Bennett and Bredensteiner [2000] show the equivalence for $b = 0$. Consider two parallel hyperplanes, normal to w, that we move apart until they touch the sets $\mathcal{A}$ and $\mathcal{B}$, respectively. That means it is $\langle x, w\rangle \geq \beta$ for points in class $\mathcal{B}$ and $\langle x, w\rangle \leq \alpha$ in class $\mathcal{A}$ with points from each class fulfilling the condition with equality. To find maximally distant such hyperplanes, we maximize the distance $(\beta - \alpha)/\|w\|$ with respect to w, ending up with the SVM optimization problem. The best classifier will be the hyperplane exactly between the two supporting ones. An analogous interpretation exists for soft margin SVMs: try to find the closest points in the reduced convex hulls of the classes. The reduced convex hull only allows a factor smaller than μ for each point in the linear combination and corresponds to the dual concept of enlarging the margin via slack variables [Bennett and Bredensteiner, 2000, Zhou et al., 2002].

But what is so desirable about large margins? Three main arguments are detailed in the following: First, the margin makes the classifier more robust to noise, and also compressible, second, it follows "Occam's razor", and third, it provides theoretical guarantees for generalization and may be interpreted as regularization.

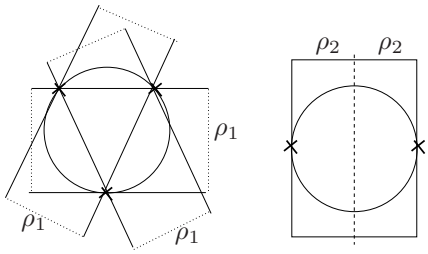


Figure 2.1.2: Illustration of the radius margin bound for SVMs. The data is enclosed in the sphere. A margin of ρ_1 allows three partitions, whereas with a larger margin of ρ_2 only one partition remains. (Figure adapted from [Schölkopf and Smola, 2002, Fig. 5.4].)

Robustness

Intuitively, a large margin classifier is one that is most robust to noise in the data. Imagine, for an SVM, that the sample points move about by a certain amount. The classifier remains correct on the training data as long as no point crosses the hyperplane. Hence, the separator with the largest distance to any point is the one providing most such "freedom" to the data, defined as a movement of radius ρ, while still classifying $\mathcal{Z}_n$ correctly [Schölkopf and Smola, 2002, p. 193].

On the other hand, fix the data points but imagine the hyperplane to rotate by a certain angle, possibly because it was not coded with high accuracy. The larger the margin, the more it may move without a change in the classification of the sample points. Hence, the larger the margin, the lower the accuracy needed to encode the direction w. In this regard, a large margin corresponds to better compression [von Luxburg, 2001, Sec. IV.2.1].

Simplicity

Furthermore, the classifier with the largest margin corresponds to the simplest one, satisfying Occam's razor [von Luxburg, 2001, Sec. III.3]. The Franciscan friar William of Ockham stated the "lex parsimoniae" (law of parsimony/succinctness): if many theories are available, choose the one with the fewest assumptions. The latter is often interpreted as "the simplest". The explanation of any phenomenon should make as few assumptions as possible, thus eliminate those explanations that do not make any difference in the observable prediction of the hypothesis or theory and only keep the simplest [wikipedia].

Generalization bounds

The theoretically most interesting aspect is the derivation of generalization bounds based on the margin. Aiming for a large margin corresponds to favoring a less complex predictor that is less prone to overfitting. For detailed reading, refer to Cristianini and Shawe-Taylor [2000, Sec. 4.3], Vapnik [2001, Sec. 10.3], and Bartlett and Shawe-Taylor [1998].

First, the VC dimension (cf. Section 1.1.2) can be bounded with respect to the margin and the radius $\widetilde{R}$ of a sphere centered at the origin that encloses the data. Let $\mathcal{F}$ be the set of linear classifiers of the form $f(x) = \operatorname{sgn}(\langle w, x\rangle)$ with $\|w\| \leq 1$. The subset $\mathcal{F}_\rho \subseteq \mathcal{F}$ is the subset of such classifiers with a margin of at least ρ on the given training sample $\mathcal{Z}_n$, that means $Y_i f(X_i) \geq \rho$ for all $(X_i, Y_i) \in \mathcal{Z}_n$. Then $\mathcal{F}_\rho$ has a VC dimension of at most $\min\{\widetilde{R}^2/\rho^2, n\} + 1$ [Bartlett and Shawe-Taylor, 1998]. In other terms, the larger the margin with respect to the spread of the data, the less partitions are possible. Figure 2.1.2 illustrates the relation of margin, radius and number of partitions. The figure also illustrates another result about margins: The margin over all dichotomies of $k \leq d + 1$ points in $\mathbb{R}^d$ is maximized when the points form a regular simplex on the sphere [Hush and Shovel,

2001]. The fat-shattering dimension can be bounded analogously to the VC dimension: $\mathrm{fat}_{\mathcal{F}_\rho}(\rho) \leq \widetilde{R}^2/\rho^2$ [Cristianini and Shawe-Taylor, 2000, Thm. 4.16]. Note that the margin is a data-dependent measure of complexity. Hence, it may lead to a data-dependent Structural risk minimization, using a structure of function classes $\mathcal{F}_\rho$ according to the margin [Bartlett and Shawe-Taylor, 1998].

The complexity bounds with respect to the margin imply generalization bounds involving ρ and $\widetilde{R}$. In the separable case with a linear classifier ($\widehat{R}(f) = 0$), for instance, $R(f)$ grows as $O\left(\frac{\widetilde{R}^2}{n\rho^2}\right)$ [Cristianini et al., 1999, citing [Vapnik, 2001]]. If there is noise, the bound may also be stated in terms of the k-th smallest (point-wise) margin, that means we ignore the $k-1$ points with smaller margin as outliers. This relaxation worsens the bound by a square root and an additional summand k/n [Cristianini and Shawe-Taylor, 2000, Thm. 4.19],[Bartlett and Shawe-Taylor, 1998].

The margin also provides an estimation of how benign the data distribution is. If the data is away from the hyperplane, then fewer examples are needed to estimate the separator to a given accuracy. Benign distributions result with high probability in a small margin [Bartlett and Shawe-Taylor, 1998].

Regularization

Related to the bounds, the margin term in the optimization criterion may be seen as a regularizer to penalize complexity. The criterion for soft margin SVMs with kernels is then the sum of an error term and the regularizer $\|w\|^2$. For many kernels, this regularizer measures the smoothness of the function, so smoother ("simpler") functions are preferred [von Luxburg, 2001, Sec. III.3].

Some further views of margins

One interpretation of the margin is that it describes how much the data may be perturbed before another predictor or solution is better than the current one. This view draws connections to the notion of stability. Bilu and Linial [2004] study the NP-hard Maxcut problem and pose the question whether "stable" instances of a problem are easier to solve. They define the stability of an optimal solution by the amount that the edges in the graph may be scaled before another solution is equally good. In that respect, the measure is similar to the margin concept outlined above. Stability here is defined for an instance of the problem (i.e. the graph) and not an algorithm. For instances with a certain stability, the Maxcut problem can be solved in polynomial time. Hence, stability simplifies the optimization. Our hope is to find a margin criterion that has the same potential. However, the authors distinguish between local and global stability. The former considers any other solution to become better, whereas the latter only looks at the possible scaling of currently cut edges at one node, and its potential swap to the other cluster. Local stability does not imply global stability and is thus no basis for their polynomial algorithm. Their concept of distinctness also reminds of a margin: a maximal cut is ρ-distinct if the relative loss of moving any subset of points across the cut is greater than ρ, so there is no solution with a quality within the relative range of ρ. Note, however, that all these concepts are merely based on discrete optimization on given samples without any relations to SLT or underlying distributions.

Similar ideas to the ones we follow below are suggested by Pelckmans et al. [2007], but with regard to transductive graph cuts. They regularize with an average margin, which corresponds to the Mincut. The dual of the relaxed version of the resulting optimization problem may be interpreted as a flow problem, similar to our approach.

The large margin principle for SVMs has also been directly extended to clustering. For clustering, this means to find a hyperplane that separates the points, but without given labels, so the labels are variables as well. Rahimi and Recht [2004] show that the spectral relaxation of Ncut corresponds to a projection of the points into a high dimensional space and the search for a hyperplane in this space with maximum average margin, with an additional balance constraint. The margin is a weighted average of the distances of the points to the plane. The equivalent of the SVM margin, the minimum distance to any point, is suggested but not efficiently solved. Xu et al. [2004] revert to a soft margin to solve the max min margin problem as a semidefinite program. The addition of an offset b for the hyperplane is realized by Valizadegan and Jin [2006].

The concept of a large margin classifier has been extended to metric spaces by von Luxburg [2001, Ch. III] and Hein et al. [2005], using Banach spaces, and Lipschitz functions as decision functions. The Lipschitz constant is used for regularization. Der and Lee [2007] propose a framework for large margin classification in Banach spaces with semi-inner products. In the following, we attempt to apply the margin concept to clustering, based on the viewpoint of robustness or perturbation of the data points.

2.2 In search of a margin for graph cuts

For compatibility with graph cut algorithms, we seek to define a margin for clustering in terms of graphs. Unlike the extensions mentioned above, we cannot rely on an obvious dot product or line separator.

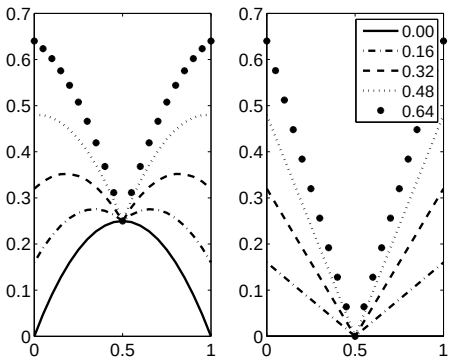
The main idea of a margin as above is that it measures the distance of a point from the "decision boundary". The latter is though not well-defined for clustering. Hence, we revert to the viewpoint of robustness against perturbations and define the margin as the maximum possible perturbation before a point is better assigned to another cluster. The viewpoint of perturbations motivates a variety of definitions of a cut margin. On the one hand, they depend on the cut criterion. On the other hand, the type of perturbation comes into play. No matter if the perturbation is viewed as nodes moving or being resampled, or as a direct modification of the edge weights, what eventually affects the cut value is the change of edge weights within and across cluster limits. So in general, we define the margin of a single node as a measure of how much its edge weights need to be perturbed before it will be better to assign it to another cluster.

Formally, we consider a perturbation as an additive change of ε in the weight of a within-cluster or cross edge. We first define the margin $\rho(v)$ for one node $v \in V$, and then set the global margin to the minimum margin of all nodes in the graph. Denote the two clusters of a given partition f by $\mathcal{A}$ and $\mathcal{B}$, and let $v \in \mathcal{A}$. Let $f_{\mathcal{A}}$ be the partition equal to f that assigns v to $\mathcal{A}$, and $f_{\mathcal{B}}$ the partition where v is moved to $\mathcal{B}$. The remaining assignments remain unchanged. Let w_ε be the edge weights after a change of ε. For the margin, we determine how large ε needs to be such that $f_{\mathcal{B}}$ has a lower cut value than $f_{\mathcal{A}}$.

We first state a normalized margin for Mincut. For completeness, a similar criterion is given for Ncut. Since the latter leads to a quadratic equation, we focus on the Mincut criterion in the rest of the chapter.

A further variation of the margin concept is a soft margin, where some nodes are allowed to have a margin smaller than ρ. This approach could be interpreted as putting the ignored nodes in the "background". The soft margin is not considered further in this report.

Figure 2.2.1: Margin for a clique (each node connected with unit weight to each other node). The partitions are between clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ of size αn and $(1-\alpha)n$. Left: Value of $f(\alpha) - \lambda \min_{v \in V} \rho(v)$, where $f(\alpha)$ is the cut value for a partition with relative cluster sizes α and $1-\alpha$. The different lines correspond to different relative margin weights $\lambda/(n-1)$. The margin moves the optimum from one empty cluster ($\alpha = 0$) to balanced clusters ($\alpha = 0.5$). Right: value of the margin term $-\lambda \min_{v \in V} \rho(v)$. The x axis corresponds to the values of α.



2.2.1 Mincut

The Mincut criterion merely adds the cross-cluster edges, so any perturbation of within-cluster edges cannot affect the cut value. The contribution of v to Mincut$(w, f_{\mathcal{A}})$ are the weights of its adjacent edges to $\mathcal{B}$, $w(v, \mathcal{B})$. $f_{\mathcal{B}}$ is better than $f_{\mathcal{A}}$ if v is more attached to $\mathcal{B}$ than to $\mathcal{A}$, so $w(v, \mathcal{B}) > w(v, \mathcal{A})$. To obtain a number between -1 and 1, we normalize by the degree of v:

$$\rho(v) = \frac{w(v, \mathcal{A}) - w(v, \mathcal{B})}{w(v, V)} = 1 - 2\frac{w(v, \mathcal{B})}{w(v, V)} \in [-1, 1]. \tag{2.2.1}$$

The Mincut margin is a measure how relatively strongly a node is attached to its cluster. Negative values indicate that it is better to move v to $\mathcal{B}$. If v is exactly in between $\mathcal{A}$ and $\mathcal{B}$, the margin will be zero.

Discussion

This margin will favor partitions that only cut a small fraction of the edge weights adjacent to each node. It ignores, however, the absolute contribution of a particular node to the cut value. This ignorance may lead to problems if the node degrees are broadly distributed. On the other hand, absolute values are considered by the Mincut criterion itself, so a combination of both criteria might remedy this problem. In part, the question about contributions leads to the question if all nodes are equally important, or if a node's importance depends on its degree.

For some graphs, the Mincut margin includes a balancing criterion via balancing the edge weights. Consider a clique. Then the partition with the largest margin will be any one where $|\mathcal{A}| = |\mathcal{B}|$, with $\rho(v) = 0$ for each node. Figure 2.2.1 shows how the margin influences the optimal partition for a clique.

Related to the balancing, the margin will disfavor partitions where one cluster consists of one node only, because the margin for this node will reach its minimum of -1.

Matrix notation

In terms of matrices, the above margin is

$$\rho(V) = (D^{-1}W(i_{\mathcal{A}} - i_{\mathcal{B}})) \odot (i_{\mathcal{A}} - i_{\mathcal{B}}) = \mathrm{diag}(i_{\mathcal{A}} - i_{\mathcal{B}})D^{-1}W(i_{\mathcal{A}} - i_{\mathcal{B}}),$$

or, based on the second term in (2.2.1),

$$\rho(V) = 1 - 2\left((D^{-1}W(-i_{\mathcal{B}})) \odot i_{\mathcal{A}} + (D^{-1}W(-i_{\mathcal{A}})) \odot i_{\mathcal{B}}\right),$$

where $i_{\mathcal{A}}$ and $i_{\mathcal{B}}$ are the indicator vectors for $\mathcal{A}$ and $\mathcal{B}$, respectively, and $\odot$ is the entry-wise product. The matrix equations yield a vector whose entries are the margins for all nodes. Here, D is a diagonal matrix where $D(j,j)$ is the degree of node v_j and W is the weight matrix with $W(i,j) = w(v_i, v_j)$.

Relation to other criteria

The idea of looking at relative edge weights within versus between clusters is also inherent in other cut criteria. As an example, the modularity approach by Newman [2006] considers entire clusters and squares the quantities. Let $p(u,v) = w(u,V)w(v,V)/w(V,V)$, then the modularity is

$$\begin{aligned}\mathrm{Mod}(f) &= \frac{1}{w(V,V)} \sum_{u,v \in V} [w(u,v) - p(u,v)]\delta(f(u), f(v)) \\ &= \frac{2(w(\mathcal{A},\mathcal{A})w(\mathcal{B},\mathcal{B}) - w(\mathcal{A},\mathcal{B})^2)}{w(V,V)^2}.\end{aligned}$$

Furthermore, the normalized Mincut margin is similar to the idea of the relative net contribution $\rho_f(\mathcal{T})$ of a group $\mathcal{T}$ of nodes to the cut, and ρ-distinctness, both defined in Bilu and Linial [2004]. Their relative contribution to Maxcut for a single node ($\mathcal{T} = \{v\}$, $v \in \mathcal{A}$) boils down to

$$\rho_f(\mathcal{T}) = \frac{w(v,\mathcal{B}) - w(v,\mathcal{A})}{\min\{w(v,\mathcal{A}), w(v,\mathcal{B})\}}.$$

Note that the authors consider Maxcut, hence the difference is reversed. The normalization is not by degree but by the smaller fraction of edges to a cluster. The great difference in concept is that not only single nodes are considered, but $\mathcal{T}$ can be any subgroup of V. Hence, distinctness is a global measure. A cut is ρ-distinct if the contribution of all possible subgroups is at least ρ. The consideration of all 2^n subgroups, however, makes the distinctness expensive to compute.

Multiplicative perturbation

Alternatively, the perturbation of edge weights could be multiplicative, as is discussed by Bilu and Linial [2004]. For $\varepsilon w(v,\mathcal{B})$ to be larger than $w(v,\mathcal{A})$, it must be $\varepsilon > w(v,\mathcal{A})/w(v,\mathcal{B})$. If $\varepsilon < 1$, then $f_{\mathcal{B}}$ is better than $f_{\mathcal{A}}$ already.

The multiplicative perturbation is equal to the criterion of local stability in Bilu and Linial [2004]. A cut is γ-locally stable in their sense if $\gamma = 1/\rho$ for the multiplicative Mincut margin.

2.2.2 Ncut

Without any perturbation, the Normalized Cut value for $f_{\mathcal{A}}$ and $f_{\mathcal{B}}$ is

$$\begin{aligned}\mathrm{Ncut}(w, f_{\mathcal{A}}) &= \frac{w(\mathcal{A},\mathcal{B})w(V,V)}{w(\mathcal{A},V)w(\mathcal{B},V)} \\ \mathrm{Ncut}(w, f_{\mathcal{B}}) &= \frac{(w(\mathcal{A},\mathcal{B}) - w(v,\mathcal{B}) + w(v,\mathcal{A}))w(V,V)}{(w(\mathcal{A},V) - w(v,V))(w(\mathcal{B},V) + w(v,V))}\end{aligned}$$

As the edge weights within and between clusters are summed up, respectively, it is enough to consider two cases: first, adding ε^w to any edge within $\mathcal{A}$ and, second, ε^b to any edge between $\mathcal{A}$ and $\mathcal{B}$.

Changing the within-cluster edges by ε^w gives

$$\begin{aligned}
\mathrm{Ncut}(w_{\varepsilon^w}, f_{\mathcal{A}}) &= \frac{w(\mathcal{A},\mathcal{B})(w(V,V)+2\varepsilon^w)}{(w(\mathcal{A},V)+2\varepsilon^w)w(\mathcal{B},V)} \\
\mathrm{Ncut}(w_{\varepsilon^w}, f_{\mathcal{B}}) &= \frac{(w(\mathcal{A},\mathcal{B})-w(v,\mathcal{B})+w(v,\mathcal{A})+\varepsilon^w)(w(V,V)+2\varepsilon^w)}{(w(\mathcal{A},V)-w(v,V)+\varepsilon^w)(w(\mathcal{B},V)+w(v,V)+\varepsilon^w)}.
\end{aligned}$$

Setting $\mathrm{Ncut}(w_{\varepsilon^w}, f_{\mathcal{A}}) = \mathrm{Ncut}(w_{\varepsilon^w}, f_{\mathcal{B}})$ yields a polynomial $a(\varepsilon^w)^2 + b\varepsilon^w + c = 0$ with

$$\begin{aligned}
a &= -w(\mathcal{A},\mathcal{B}) + 2w(\mathcal{B},V) \\
b &= -w(\mathcal{A},\mathcal{B})(w(\mathcal{A},V)+w(\mathcal{B},V)) \\
&\quad + w(\mathcal{B},V)(2(w(\mathcal{A},\mathcal{B})-w(v,\mathcal{B})+w(v,\mathcal{A}))+w(\mathcal{A},V)) \\
c &= -w(\mathcal{A},\mathcal{B})(w(\mathcal{A},V)-w(v,V))(w(\mathcal{B},V)+w(v,V)) \\
&\quad + w(\mathcal{B},V)w(\mathcal{A},V)(w(\mathcal{A},B)+w(v,\mathcal{A})-w(v,\mathcal{B})).
\end{aligned}$$

Analogously, the addition of ε^b to a cross edge leads to

$$\begin{aligned}
\mathrm{Ncut}(w_{\varepsilon^b}, f_{\mathcal{A}}) &= \frac{(w(\mathcal{A},\mathcal{B})+\varepsilon^b)(w(V,V)+2\varepsilon^b)}{(w(\mathcal{A},V)+\varepsilon^b)(w(\mathcal{B},V)+\varepsilon^b)} \\
\mathrm{Ncut}(w_{\varepsilon^b}, f_{\mathcal{B}}) &= \frac{(w(\mathcal{A},\mathcal{B})-w(v,\mathcal{B})+w(v,\mathcal{A}))(w(V,V)+2\varepsilon^b)}{(w(\mathcal{A},V)-w(v,V))(w(\mathcal{B},V)+w(v,V)+2\varepsilon^b)}
\end{aligned}$$

and a quadratic polynomial with

$$\begin{aligned}
a &= -2(w(\mathcal{A},V)-w(v,V)) + (w(\mathcal{A},\mathcal{B})+w(v,\mathcal{A})-w(v,\mathcal{B})) \\
b &= -(w(\mathcal{A},V)-w(v,V))(w(\mathcal{B},V)+w(v,V)+2w(\mathcal{A},\mathcal{B})) \\
&\quad + (w(\mathcal{A},\mathcal{B})+w(v,\mathcal{A})-w(v,\mathcal{B}))(w(\mathcal{A},V)+w(\mathcal{B},V)) \\
c &= -(w(\mathcal{A},V)-w(v,V))w(\mathcal{A},\mathcal{B})(w(\mathcal{B},V)+w(v,V)) \\
&\quad + (w(\mathcal{A},\mathcal{B})+w(v,\mathcal{A})-w(v,\mathcal{B}))w(\mathcal{A},V)w(\mathcal{B},V).
\end{aligned}$$

For the margin of v, set $\rho(v) = \min\{-\varepsilon^w, \varepsilon^b\}$ and the margin of f to $\rho(f) = \min_{v\in V}\rho(v)$.

Alternatively, one might consider a setting where ε is added to the within-edges and $-\varepsilon$ to the cross edges. This setting, however, is even more complicated, resulting in a third-order polynomial.

2.2.3 Illustration of the margins

In the following, we investigate the behavior of some margins on toy graphs. First, several margins are compared with respect to their "critical node", and second, the best cuts for two margins illustrated. In the end, we look at the effect of adding edges to a grid graph.

In the first experiment, we computed different margins of all possible cuts of a toy graph. Figure 2.2.2 shows a selection of those cuts, where the cluster assignments are indicated by node color, blue or magenta. The node with the smallest margin is marked in red, one with the largest margin in green. The illustrated margins are:

ρ_1 Additive Ncut margin as defined in Section 2.2.2.

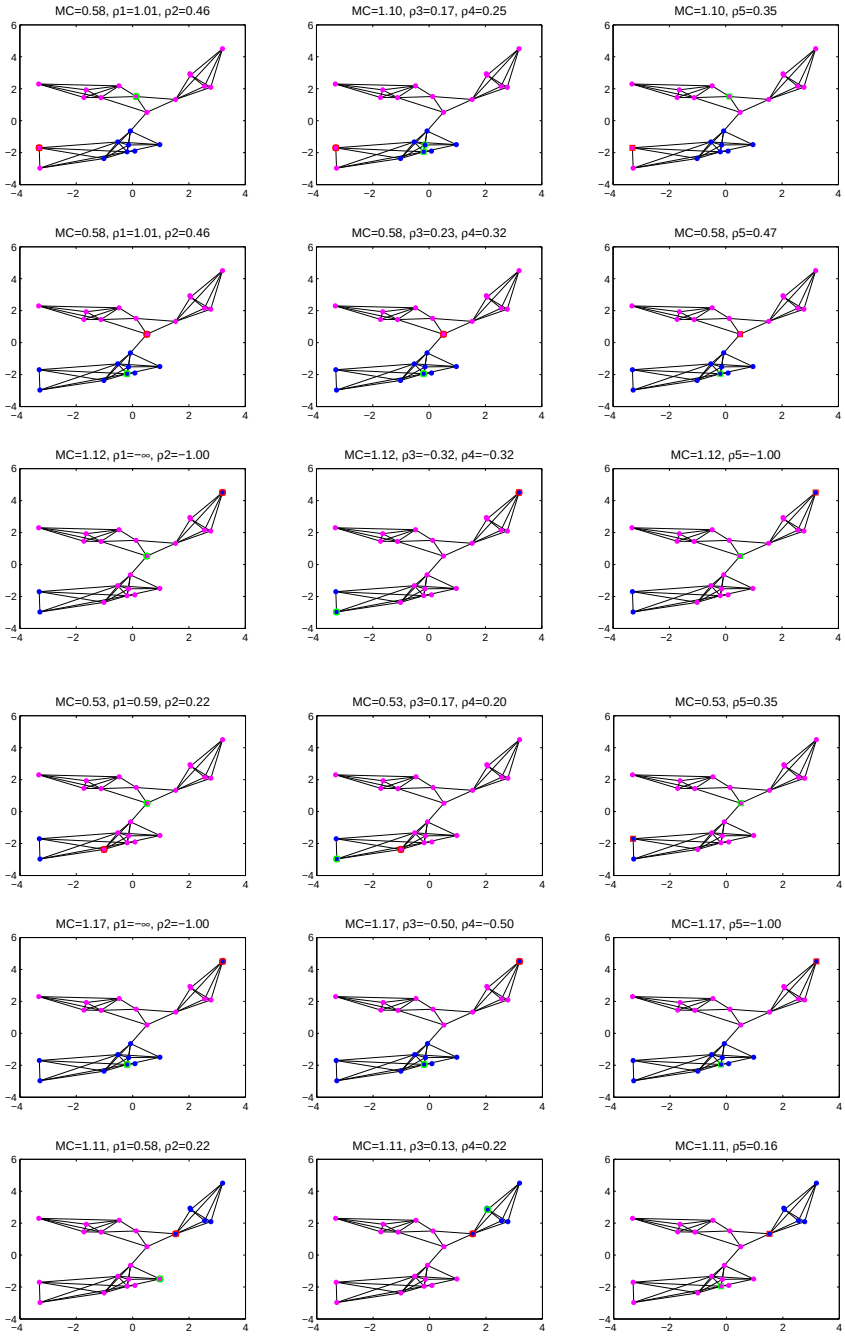


Figure 2.2.2: Illustration of the different margins. One row corresponds to one cut, columns to the different margins. The node with the lowest margin is marked in red, one with the largest margin in green. Margins with odd number have round markers, the others square markers.

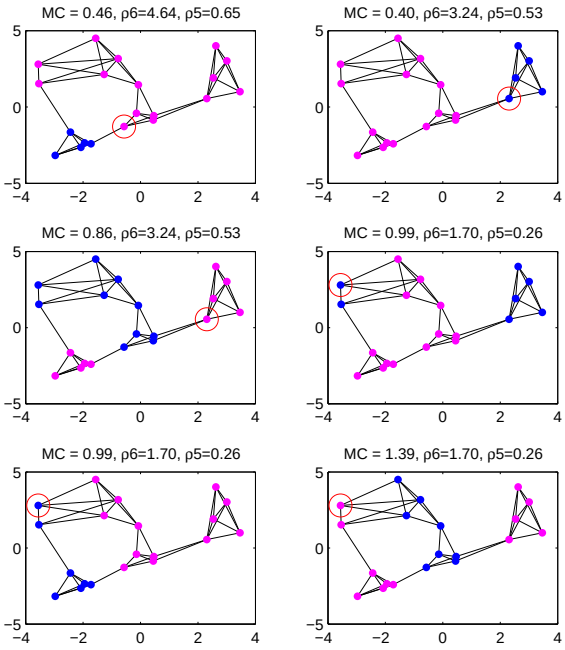


Figure 2.2.3: The six most stable cuts are similar for the additive (ρ_5) and multiplicative (ρ_6) Mincut margins. The node that determines the margin is marked in red.

ρ_2 The same as ρ_1, but normalized by the degree of the node, that means divided by $w(v, V)$.

ρ_3 Ncut margin where ε is added to a within edge and simultaneously subtracted from a cross edge. The computation is then analogous to ρ_1.

ρ_4 The same as ρ_3, but normalized by the degree of the node.

ρ_5 Additive normalized Mincut margin as in Equation (2.2.1).

ρ_6 Multiplicative Mincut margin as described in Section 2.2.1.

In general for this graph, the critical node determining the margin is the same for the different margin variations. It can vary though, as for instance in row four.

If one node is cut off, like in row 5, then its margin is the minimum possible for all variations. For cut 2, all margins reach their peak and, equally, cut 5 achieves the lowest value for each margin. In between those two, there are differences in the order, even between a margin and its normalized form. Hence the normalization does influence the margin value. Note also that the lowest cut value (row 4) does not correspond to the one with the largest margin.

The node with the lowest margin is the same for the normalized and unnormalized versions of the margin in all examples (red markers, square and circle coincide). The point with the largest margin, however, differs in one example for the additive Ncut margin ρ_3 with simultaneous addition and subtraction. In other graphs (not shown here) the node with the lowest margin varied as well, also for ρ_1 and ρ_2. This divergence points to the fact that normalization by node degree can make a difference, not only with regard to the choice of the partition, but also with regard to which node is critical.

nodes in cluster A	(i)			(ii)			(iii)		
	cut	ρ_5	ρ_6	cut	ρ_5	ρ_6	cut	ρ_5	ρ_6
vertical cut									
1, 5, 9, 13	4.0	0.00	1.00	4.0	0.27	1.75	8.0	-0.07	0.88
1, 5, 9, 13, 2, 6, 10, 14	3.0	0.46	2.67	3.0	0.57	3.67	7.0	0.47	2.75
1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15	4.0	0.00	1.00	4.0	0.27	1.75	8.0	-0.07	0.88
diagonal cut									
4	2.0	-1.0	0.00	3.0	-1.00	0.00	4.0	-1.00	0.00
3, 4, 8	3.5	-0.27	0.57	5.5	-0.47	0.36	7.5	-0.47	0.36
2, 3, 4, 7, 8, 12	5.5	-0.46	0.38	8.5	-0.60	0.25	11.5	-0.60	0.25
diagonal cluster – one "row"									
1, 6, 11, 16	11.0	-1.00	0.00	13.0	-1.00	0.00	15.0	-1.00	0.00
5, 10, 15	9.0	-1.00	0.00	10.0	-1.00	0.00	11.0	-1.00	0.00
9, 14	5.5	-1.00	0.00	6.5	-1.00	0.00	7.5	-1.00	0.00
diagonal cluster – two "rows"									
1, 5, 6, 10, 11, 15, 16	9.0	-0.46	0.38	12.0	-0.46	0.38	15.0	-0.50	0.33
5, 9, 10, 14, 15	7.5	-1.00	0.00	9.5	-0.60	0.25	11.5	-0.60	0.25
"around the corner"									
1	2.0	-1.00	0.00	3.0	-1.00	0.00	4.0	-1.00	0.00
1, 2, 5, 6	3.0	0.14	1.33	5.0	0.07	1.14	7.0	0.000	1.00
1, 2, 3, 5, 6, 7, 9, 10, 11	6.0	-0.14	0.75	9.0	-0.33	0.50	12.0	-0.14	0.75

Table 2.1: Cut and margin values for the grid

Figure 2.2.3 displays the six cuts with the largest margin for the normalized additive (ρ_5) and multiplicative Mincut margin ρ_6. As above, the "critical" node determining the margin is marked by a red circle. As we often observed with the toy examples, the best cuts and critical nodes coincide for both margins. The large margin cuts correspond to what one visually considers as clusters.

The figure shows that several cuts can have the same margin. Apart from the assignment of the critical node and its neighbors, the partition can vary as long as no other node gets a lower margin. Hence, the margin values define equivalence classes of partitions with the same margin. The lower the defining limit margin, the larger the class can be (and the higher the risk of potential overfitting). Within one such class, the partition with the lowest cut value is preferable, that is the one minimizing the empirical risk. These ideas remind of the concept of Structural Risk Minimization (cf. Section 1.1.2). Which such equivalence class is chosen in the end depends on the relative weighting of cut value and margin.

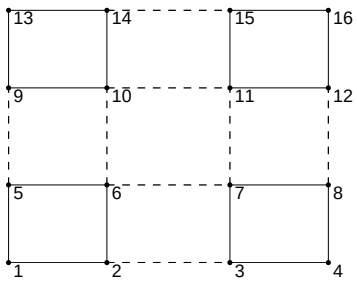


Figure 2.2.4: Grid graph

Note that the equivalence classes differ in the assignment of densely connected sub-clusters whose division results in a margin lower than the one defining the class. Thus, the difference of the partitions within one class may hint to the actual number of clusters, and to densely connected (sub-)communities within larger clusters.

For the behavior on one special graph, consider a grid as in Figure 2.2.4. The dashed edges have a

weight of 0.75, the black edges weigh 1.0. Variations are (ii) the addition of black edges $(1, 13)$, $(2, 14)$, $(3, 15)$, $(4, 16)$, and (iii) the additional addition of edges $(1, 4)$, $(5, 8)$, $(9, 12)$, $(13, 16)$. Table 2.1 displays the cut and margin values for several possible partitions. Intuitively, the squares of four nodes in each corner each form a cluster, so one would partition such that only dashed edges are cut.

Three observations catch the eye: First, two cuts can have the same cut value, but different margins. A vertical cut through the middle or around one square for (i), for instance, have the best cut values, but the vertical partition wins by the margin. Second, in the end, for all grids, the balanced vertical partition is the best by the margin. This corresponds to the intuition, because the added edges strongly connect the squares, creating two groups for (ii). In (iii), all squares are connected equally to each other. For such regularly connected graphs, the margin induces a balancing of cluster sizes to reduce the number of connections of a single node to the other cluster (see also Subsection 2.2.1). Third, the example shows that margin values can increase and decrease with the addition of edges.

2.3 Optimization problem

Let us now modify Q_n by a margin criterion and solve the resulting optimization problem. In the following, we focus on the additive Mincut margin ρ_5. The criterion of the absolute cut value, which is the objective of the original Mincut problem, is extended by the margin, weighted by a factor λ. Thus, the goal is to minimize

$$\min_f \mathrm{Cut}(f) - \lambda\rho(f) \equiv \min_f \mathrm{MCM}(f) \tag{2.3.1}$$

for the given graph, subject to certain conditions given, for example, by the edge weights. We will solve this problem as a Network Flow, based on the Mincut–Maxflow duality [Elias et al., 1956, Ford and Fulkerson, 1956].

The equations are stated in terms of vectors and matrices. Let the given graph $G = (V, E)$ have n nodes and m edges. The partition f is an indicator vector in $\{0, 1\}^n$ for the clusters $\mathcal{C}_0$ and $\mathcal{C}_1$. Let $A \in \{-1, 0, 1\}^{n\times m}$ be the node–edge adjacency matrix and $w \in \mathbb{R}^m_+$ the weight vector. The matrix A has a column for each edge $e_j = (v_i, v_k)$, with entries $A(i, j) = -1$ and $A(k, j) = 1$.

The following outline is based on Papadimitriou and Steiglitz [1982].

2.3.1 The Maxflow Problem and its dual

In the Maximum Flow problem, the graph may be viewed as an electrical circuit, a system of water pipes or roads on which goods are transported. The goal is to assign flow values to the edges such that the total flow from a distinguished node s, the source, to the node t, the sink, is maximized. The flow in an edge may not exceed the capacity given by the edge weight (capacity constraint), and the total inflow to a node must equal the flow out of the node (flow conservation), except for the source and sink. In addition, flow cannot be negative.

In matrix notation, we seek to find a flow assignment $h \in \mathbb{R}^m$ such that the flow v from s to t is maximized:

$$\begin{aligned} &\max_{h,v} v && && (2.3.2)\\ \text{s.t.}\quad & Ah + cv = 0 && \text{(flow conservation)}\\ & h \le w && \text{(capacity constraint)}\\ & h \ge 0. \end{aligned}$$

The variable c indicates source and sink:

$$c(i) = \begin{cases} -1 & \text{for } i = s \\ 1 & \text{for } i = t \\ 0 & \text{otherwise.} \end{cases}$$

It helps to capture the flow from s and into t.

The dual of problem (2.3.2) is known as the Minimum Cut problem:

$$\begin{aligned} &\min_{\gamma,f} \gamma^\top w && && (2.3.3)\\ \text{s.t.}\quad & A^\top f + \gamma \ge 0 && \text{(let } \gamma \text{ mark cross edges)}\\ & c^\top f \ge 1 && \text{(separate } s \text{ and } t\text{)}\\ & \gamma \ge 0. \end{aligned}$$

The second constraint, $-f(s) + f(t) \ge 1$, forces $f(s) = 0$ and $f(t) = 1$, so source and sink are assigned to different groups or clusters. The first constraint demands for each edge (i, j) that $f(i) - f(j) + \gamma(i, j) \ge 0$: if $f(i) = 0$ and $f(j) = 1$, then $\gamma(i, j)$ must be at least one, otherwise it can be zero. Looking at the objective, γ should be as small as possible. Hence, $\gamma(i, j) = 1$ for all edges from nodes in cluster with label zero to nodes in the cluster labeled one, and otherwise $\gamma = 0$[1]. As a result, the objective sums the weight of all such cross edges between the clusters. Direction does play a role here, but for an undirected graph simply replace undirected edges $\{i, j\}$ by edges in both directions, (i, j) and (j, i).

For the Maxflow problem, there exist efficient polynomial-time flow algorithms. Via the dual, these algorithms also serve to efficiently find the minimum cut.

2.3.2 Extension to the margin

Now we add the margin constraint to the Mincut problem, and derive its new dual to end up with a flow problem which is potentially easier to solve, by modifications of existing algorithms.

Recall that the margin for node i in cluster $\mathcal{C}_0$ is

$$\rho(i) = \frac{w(i, \mathcal{C}_0) - w(i, \overline{\mathcal{C}_0})}{d(i)},$$

where $d(i) = w(i, V)$ is the degree. The margin for the partition is $\rho(f) = \min_{i \in V} \rho(i)$.

[1] Practical note: Without integer constraints, Matlab's `linprog` returned non-integer results for f and γ if there were multiple minimal cuts. The objective value may still be correct, though, since the output may be a linear combination of the two optimal solutions.

Extended Mincut

The extended Mincut includes the weighted margin $\rho \in [-1, 1]$, with a factor $\lambda \in \mathbb{R}$:

$$\begin{aligned}
&\min_{\gamma,f,\rho} \gamma^\top w - \lambda\rho \equiv \min_{\gamma,f,\rho} \mathrm{MCM}(f) && && (2.3.4)\\
&\text{s.t.} \quad A^\top f + \gamma \geq 0 && \text{(let } \gamma \text{ mark cross edges)}\\
&\qquad c^\top f \geq 1 && \text{(separate } s \text{ and } t)\\
&\qquad \gamma \geq 0\\
&\mathbf{1}_n - 2\widetilde{W}^{(I)}\gamma \geq \rho\mathbf{1}_n && \text{(margin constraint 1)}\\
&\mathbf{1}_n - 2\widetilde{W}^{(O)}\gamma \geq \rho\mathbf{1}_n && \text{(margin constraint 2)}
\end{aligned}$$

Here, $\mathbf{1}_n$ denotes the $n \times 1$ vector of all ones. The matrices $\widetilde{W} \in \mathbb{R}_+^{n\times m}$ give the normalized edge capacities:

$$\begin{aligned}
\text{outgoing:} \quad \widetilde{W}^{(O)}_{i,(j,k)} &= \begin{cases} \frac{w(i,k)}{d(i)} & \text{if } j = i \\ 0 & \text{otherwise.} \end{cases}\\
\text{incoming:} \quad \widetilde{W}^{(I)}_{i,(j,k)} &= \begin{cases} \frac{w(j,i)}{d(i)} & \text{if } k = i \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

If $\gamma(i,j)$ indicates the edges from nodes labeled 0 to nodes labeled 1, that means cluster $\mathcal{C}_0$ to $\mathcal{C}_1$, then the last pair of constraints encodes the margin criterion. Consider $i \in \mathcal{C}_0$, then

$$\begin{aligned}
1 - 2\sum_{j:(i,j)\in E} \gamma(i,j)\widetilde{W}^{(O)}(i,(i,j)) &= 1 - 2\sum_{j:(i,j)\in E\cap(\mathcal{C}_0\times\mathcal{C}_1)} \gamma((i,j))w(i,j)/d(i)\\
&= \frac{(w(i,\mathcal{C}_0) + w(i,\mathcal{C}_1)) - 2w(i,\mathcal{C}_1)}{d(i)}\\
&= \frac{w(i,\mathcal{C}_0) - w(i,\mathcal{C}_1)}{d(i)}
\end{aligned}$$

describes the margin for node i. For $i \in \mathcal{C}_1$, however, the sum above will be one, because γ is zero for edges out of $\mathcal{C}_1$. The other constraint takes care of nodes in $\mathcal{C}_1$:

$$\begin{aligned}
1 - 2\sum_{j:(i,j)\in E} \gamma(i,j)\widetilde{W}^{(I)}(j,(i,j)) &= 1 - 2\sum_{j:(j,i)\in E\cap(\mathcal{C}_0\times\mathcal{C}_1)} \gamma((j,i))w(j,i)/d(i)\\
&= \frac{(w(i,\mathcal{C}_0) + w(i,\mathcal{C}_1)) - 2w(\mathcal{C}_0,i)}{d(i)}\\
&= \frac{-w(i,\mathcal{C}_0) + w(i,\mathcal{C}_1)}{d(i)}, \qquad (2.3.5)
\end{aligned}$$

if $i \in \mathcal{C}_1$. As above, a vertex $i \in \mathcal{C}_0$ does not have any incoming edges (j,i) with $\gamma(j,i) = 1$, so the term (2.3.5) will be one for that node, the maximum possible margin.

Dual: Flow problem

To derive the dual (as outlined in Boyd and Vandenberghe [2004, Ch. 5]), we start with the Lagrangian of Problem (2.3.4):

$$\begin{aligned} L(f,\gamma,\rho,v,h,g_1,g_2,p) &= w^\top\gamma - \lambda\rho + h^\top(-A^\top f - \gamma) + v(1 - c^\top f) - p^\top\gamma \\ &+ g_1^\top(\rho 1_n + 2(\widetilde{W}^{(O)})^\top\gamma - \mathbf{1}_n) + g_2^\top(\rho\mathbf{1}_n + 2(\widetilde{W}^{(I)})^\top\gamma - \mathbf{1}_n) \\ &= (w - h - p + 2\widetilde{W}^{(O)}g_1 + 2\widetilde{W}^{(I)}g_2)^\top\gamma + (-Ah - vc)^\top f \\ &+ (\lambda + \mathbf{1}_n^\top(g_1 + g_2))\rho + v - \mathbf{1}_n^\top(g_1 + g_2) \\ v, h, g_1, g_2, p &\geq 0. \end{aligned} \tag{2.3.6}$$

The corresponding dual is

$$\begin{aligned} \max_{v,g_1,g_2,h} \quad v - 1_n^\top(g_1 + g_2) \equiv \max_{v,g_1,g_2,h} \quad & v \\ \text{s. t.} \quad h &\leq w + 2((\widetilde{W}^{(O)})^\top g_1 + (\widetilde{W}^{(I)})^\top g_2) \\ Ah + vc &= 0 \\ 1_n^\top(g_1 + g_2) &= \lambda \\ h &\geq 0 \\ v &\geq 0 \\ g_1, g_2 &\geq 0. \end{aligned} \tag{2.3.7}$$

If the third constraint holds, then the objective becomes $v - \lambda$, the same as in the original Maxflow problem (2.3.2), because λ is a constant. Analogous to (2.3.2), the above dual may be interpreted as a flow problem. Again, $h(i,j)$ is the flow in edge (i,j), and v the total flow from s to t. The additional variables g_1 and g_2 are "slack" variables allowing the flow to exceed the capacity on certain edges (first constraint). The total slack is limited by λ via the third constraint.

Hence, the extended flow problem reads as follows. Maximize the flow from source to sink, respecting flow conservation and edge capacities. The capacities may be exceeded by a total of λ. The gain in capacity for the investment g_1 (for outgoing edges) or g_2 (for incoming edges) depends on the relative weight of the edge with regard to the degree of its adjacent nodes. The first constraint states this dependence for each edge (i,j):

$$h(i,j) \leq w(i,j) + 2\left(\frac{w(i,j)}{d(i)}g_1(i) + \frac{w(i,j)}{d(j)}g_2(j)\right). \tag{2.3.8}$$

To maximize the capacity gain, it is best to increase the g on the adjacent node where the edge has higher relative weight, so it makes a larger part of the node's connections. On the other hand, the g values may mark an interaction of edges: Increasing g_1 on a node with many outgoing full edges will increase the flow bound on many such edges and hence be very effective (we get a lot for "paying" one unit), and equally for g_2 and nodes with many incoming full edges. For the original Maxflow problem, the full edges are the ones that determine the maximum flow. In the dual, all cut edges are full edges, defining the minimum cut. Thus, the g variables may be indicators for the cluster borders. Nodes in between the clusters with lower margin and strong connections to the other cluster may have larger g values. Then g is an indicator of problematic or "difficult" nodes.

2.3.3 Discussion

What is the gain of a duality formulation as (2.3.4) and (2.3.6)?

If the dual is easy to solve, then the primal is, too, because the primal variables may be obtainable via complementary slackness and KKT conditions [Boyd and Vandenberghe, 2004, Sec. 5.5].

With binary γ, the extended flow problem may be solvable in polynomial time via a modified flow algorithm. A variety and detailed analysis of such algorithms is given in Ahuja et al. [1993]. Possibly, one could solve the Maxflow problem first, for example with a Preflow-Push algorithm, and then investigate where capacity extensions would be most useful. Maybe even a variation of the Network Simplex algorithm may work. On the other hand, the solution of the Mincut and Mincut with margin can be very different: Mincut often cuts off one node, whereas the margin prevents such behavior. Therefore the Maxflow solution may not be a good starting point for the algorithm. Another problem can arise from g not being integer. Sometimes scaling techniques are applied for polynomial time algorithms. A real value of g may be difficult to find with scaling algorithms, leading to a great number of iterations.

The total unimodularity of adjacency matrices [Cook et al., 1997, Ch. 6.5] can give nice properties to network problems using them as constraint matrices [Ahuja et al., 1993], such as in the original network problems. In the modified problems, the additional constraint matrices $\widetilde{W}^{(O)}$ and $\widetilde{W}^{(I)}$ are not unimodular any more. In fact, we cannot guarantee that the optimal solution of the modified problem is integer. This problem actually shows up in experiments and brings up the problem of how to define the partition that is the solution if the labels f and γ are non-integer. One solution is to explicitly put integer constraints on the labels. Note that these integer constraints make the problem NP-hard.

Other problems are easier to tackle. Note that the optimization problem requires distinguished, pre-separated nodes s, t. The original clustering problem does not include such nodes, hence different assignments of s and t should be tried. However, there is only a polynomial number of such pairs, namely $n(n-1)$, so even trying all pairs can be done in polynomial time, if the optimization problem can be solved in polynomial time.

So far, the approach only works for bipartitions. To extend it to more clusters, one may recursively reapply the algorithm to the identified clusters. The recursion will, however, probably come with a loss in accuracy: there is, for example, no guarantee that the optimal bipartition of three clusters will be one cluster on one side and two on the other instead of one or more clusters being cut.

2.3.4 Implementation: an odyssey of its own

For experimental investigation, the modified optimization problems (2.3.4) and (2.3.6) were solved as linear programs (LP) and mixed integer linear programs (MILP) using the CPLEX solver [cplex].

Due to the lack of a Matlab interface, we created graphs in Matlab and automatically generated the corresponding files in MPS format, for both the Maxflow and Mincut problems, along with corresponding files containing the CPLEX commands. Via a shell script, CPLEX was called with these commands. A pearl script helped to read out the CPLEX Log and filter it for the results, writing them out in a Matlab-readable format.

Not further mentioned will be the fight for licenses ...

2.4 Experimental investigations

The first experiment is an investigation of the influence of integer constraints to the solution, since there is no guarantee for an integer solution any more. In addition, integer constraints make linear programs much harder to solve, so that one possibly only gets a suboptimal solution. The second experiment is a test how the weighting of the margin affects the solution, and the third experiment illustrates the behavior on random graphs. The complications with CPLEX licenses limited the number of repetitions in the experiments.

2.4.1 LP versus MILP

Ideally, we would like to have binary labels to assign the nodes to the two clusters. However, the modified problems (2.3.4) and (2.3.6) do not guarantee an integer solution. To test the influence of integer constraints on the labels, we created random Gaussian graphs and solved the original Maxflow and Mincut problems (Eq. (2.3.2) and (2.3.3), respectively) as well as the modified Mincut and flow problem. The Mincut with margin was solved with and without integer constraints on the labels. The experiment was repeated with different random graphs and CPLEX used for optimization. As expected, the original and its dual as well as the modified Mincut LP and its dual had the same objective values in the end. The labels of the modified problem, however, ended up to be non-integer. The integer constraints resulted in a higher objective, which is less optimal for a minimization problem.

Since the ultimate goal is to obtain a clustering, solving without integer constraints will lead to the problem of how to infer cluster assignments from continuous labels, with the risk of a suboptimal assignment. Hence the LP problem is only a relaxation, similar to the spectral relaxation from integer to continuous variables.

2.4.2 The influence of the margin's weight

The next experiment aims to investigate the influence of the weighting factor λ of the margin. We used seven different graph types from two to four Gaussian clusters. Each cluster had 40 or 60 points, sampled from Gaussian distributions with different means and variances in two dimensions (see Table 2.2). The source and sink nodes were sampled from different Gaussians.

graph	σ	no. of points	means	variances
1	0.5	40, 40	$\mu_1 = (0,0)$, $\mu_2 = (2,2)$	$\Sigma_1 = I$, $\Sigma_2 = (1,0;r,1)$
2	0.45	40, 60	$\mu_1 = (0,0)$, $\mu_2 = (2,2)$	$\Sigma_1 = I$ $\Sigma_2 = (1,0;r,1)$
3	0.5	40, 40	$\mu_1 = (0,0)$, $\mu_2 = (2,3)$	$\Sigma_1 = I$ $\Sigma_2 = (1,0;r,1)$
4	0.5	40, 40	$\mu_1 = (0,0)$, $\mu_2 = (2,2)$	$\Sigma_1 = I$, $\Sigma_2 = I$
5	0.5	40, 40, 40	$\mu_1 = (-1.6,0)$, $\mu_2 = (1.6,0)$, $\mu_3 = (0, \sqrt{3} \cdot 1.6)$	$\Sigma_1 = \Sigma_2 = \Sigma_3 = I$
6	0.4	40, 40, 40, 40	$\mu_1 = (-1,-1)$, $\mu_2 = (-1,1)$, $\mu_3 = (1,-1)$, $\mu_4 = (1,1)$	$\Sigma_i = I$
7	0.4	40, 40, 40, 40	$\mu_1 = (-1.25,-1.25)$, $\mu_2 = (-1.25,1.25)$, $\mu_3 = (1.25,-1.25)$, $\mu_4 = (1.25,1.25)$	$\Sigma_i = I$

Table 2.2: Graphs for the weight experiments. $r \in [0,1]$ is a random number. All graphs are nearest neighbor graphs with 5 neighbors; σ is the parameter for creating the edge weights: $\exp(-\|x-y\|^2/(2\sigma^2))$, where x and y are the coordinates of the adjacent nodes.

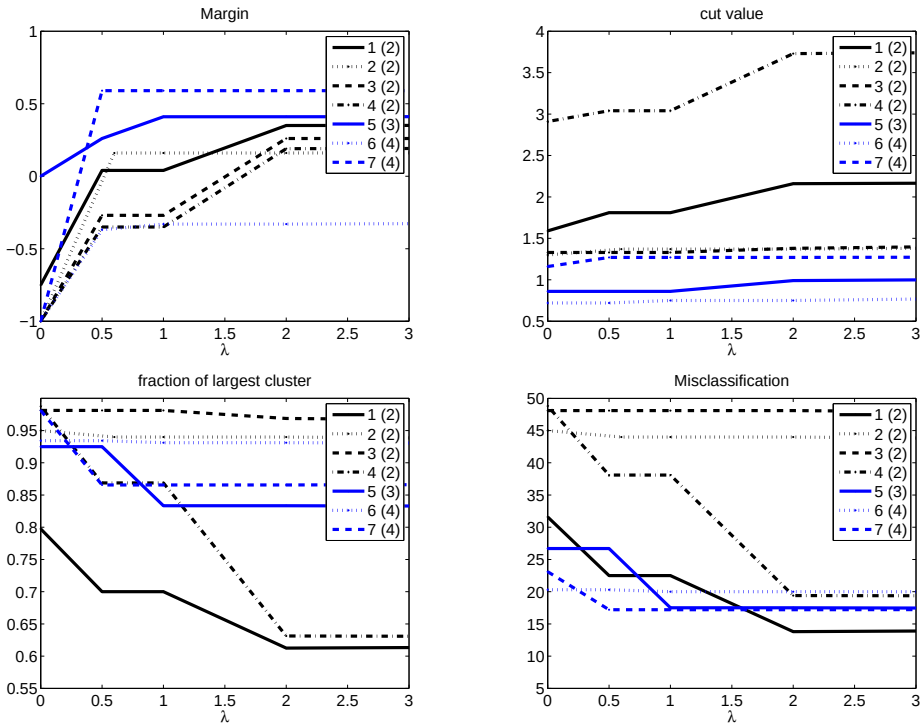
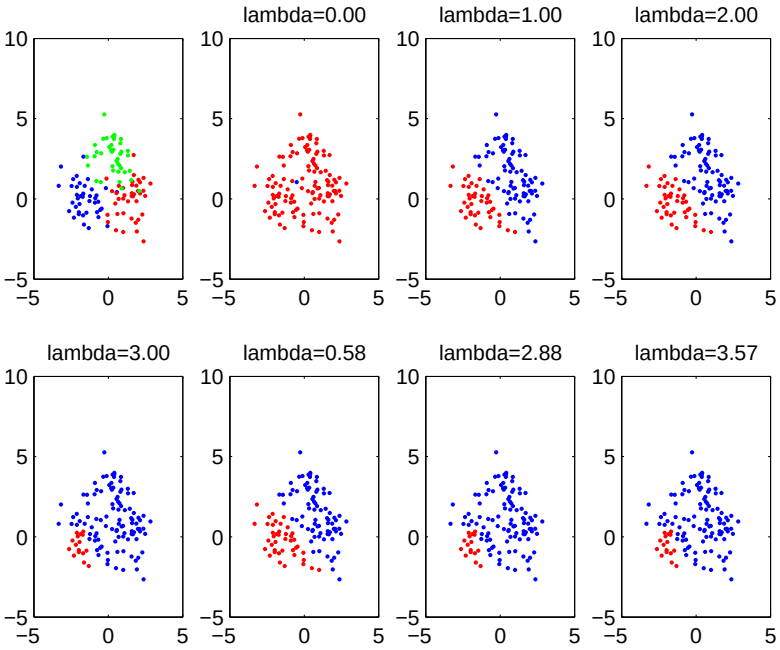


Figure 2.4.1: Influence of λ on the clustering for graph types 1 to 7. The number in parentheses is the number of clusters in the given graph.



λ	0.0	1.0	2.0	3.0	3.6
$\lvert C_0\rvert$	119	79	79	104	104
$\lvert C_1\rvert$	1	41	41	16	16
% error	32.5	5.8	5.8	20.0	20.0
$\mathrm{Cut}(f) - \lambda\rho$	2.64	3.12	2.95	2.71	2.46
$\rho(f)$	-1.00	0.17	0.17	0.44	0.44
$\mathrm{Cut}(f)$	2.64	3.29	3.29	4.03	4.03

Figure 2.4.2: Illustration of the partitions for Graph 5 with varying λ.

The optimization was solved for four instances of graph type 1 and two instances of each other type. The weighting λ was set to $\{0, 1, 2, 3, \overline{w}, 5\overline{w}/n)$, where $\overline{w}$ is the average edge weight. The latter test values take into account that the influence of λ also depends on the average edge weight, as the edge weights determine the magnitude of the cut value. Figure 2.4.1 shows how certain values of the partition develop as λ is increased. The values are averages over the instances. In general, since the solutions are discrete, the values change in steps. After the weighting exceeds a certain threshold, the solution changes and remains stable until the next threshold. The threshold depends on the margin and cut value of the other solutions.

As expected, the higher the weight of the margin, the more it is considered in the optimization, and hence increases with the increase of λ. For the original Mincut ($\lambda = 0$), the margin is often negative, as one point is cut off. If the cut is weighted highly relative to the margin, then the partition with the lowest cut value is chosen. As the importance of the margin rises, different solutions with larger margin and higher cut value are better, because the margin can outweigh the cut value with a sufficient weight.

The figure also shows the relative size of the largest cluster compared to all points. It usually decreases as λ increases, indicating that the balance of cluster sizes improves. But,

when the margin weight increases further, it may also grow again, if a less balanced partition is more locally stable. Without the margin, Mincut often leads to solutions where a small fraction of points or even a single node is cut off the rest, such that only few edges are cut. In such a small group, it is however very likely that one node contributes a large share of the cut edges, leaving it with a very low margin. Hence the margin criterion leads to a minimum size of the clusters where no node contributes more than a limited part of its edges to the cut.

The "misclassification" was computed as classification error, if the generating Gaussian distributions are considered as classes. Via the balancing of cluster sizes, the margin reduces this error in most examples.

Figure 2.4.2 illustrates the partitions for one example (graph type 5) with varying λ. The plot on the upper left shows the clusters by their generating sources. Without margin, one node is cut off the rest. A small weight of the margin, however, changes the optimal solution to a more balanced partition. An increase of more than $\lambda = 2$ leads to yet another partition that separates a smaller, denser group.

In summary, the margin remedies the tendency of Mincut to separate one node off the rest, leading to clusters of a minimal size. In addition, the margin favors groups that are separated and well-connected within the group.

2.4.3 Uniform distribution: influence of the initialization of source and sink

In order to investigate the influence of the initialization, that is the assignment of s and t, we created graphs with $n = 80$, 100 and 150 nodes from a uniform distribution (3 instances for each number of nodes). Usually not generating distinguishable clusters in the graph, the uniform distribution will reveal the effect of the initialization in its extreme. If there are clusters, the effect might be less strong.

The node coordinates are in $[-1, 1]^2$, and the source and sink were chosen in various positions:

1. in opposite corners: $s = (-1, -1)$, $t = (1, 1)$
2. both close together in one corner: $s = (-1, -1)$, $t = (-0.95, -0.95)$
3. s in the center, t in corner: $s = (0, 0)$, $t = (1, 1)$
4. both in the middle: $s = (0, 0)$, $t = (0.05, 0.05)$
5. both within opposite quadrants: $s = (-0.4, -0.4)$, $t = (0.4, 0.4)$
6. s in the center, t chosen randomly from the sample points
7. both chosen randomly from the sample points
8. in opposite corners: $s = (1, -1)$, $t = (-1, 1)$.

Unless they were chosen from the sample points, s and t were added to the n nodes. In addition, λ was varied between 1 and 20, and also set to the average edge weight.

First, the change of results for varying λ and initialization was tested with $n = 80$ and $n = 100$ nodes (3 examples each). One criterion to choose the best solution from different initializations is the objective function value MCM(f). For each value of λ, we chose the solution with the best MCM. Figure 2.4.3 shows these results. The first plot illustrates the

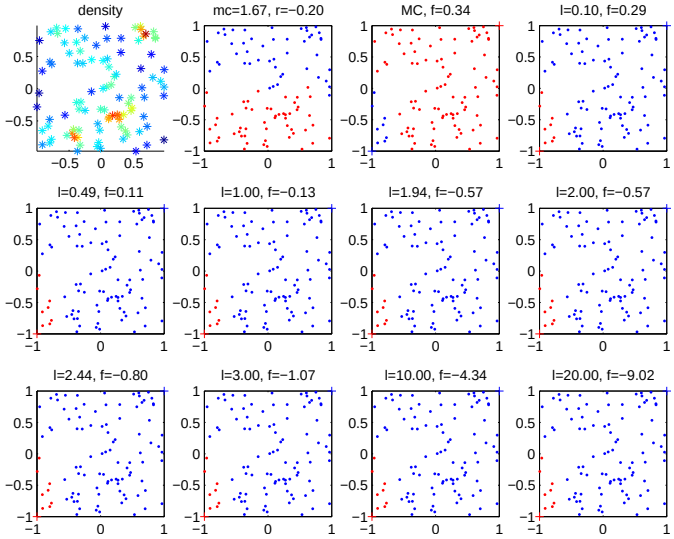


Figure 2.4.3: Optimization results for a uniform distribution, $n = 100$. The first plot illustrates the density in the graph, the second the solution for spectral clustering (Ncut), and the third the optimum for the Mincut problem. The remaining plots are the partitions for the cut with margin, l denotes λ, and f is the value of the objective, MCM.

density in the graph measured by the degree of the nodes. Dark warm colors denote a higher density, dark cold colors a lower density. In this respect, cluster limits for solutions with good objective values are not always in regions of low density. The cuts account for the actual existing edges, and so cutting a large amount of low-weight edges does not necessarily yield a lower cut value than cutting few heavy edges, as long as those heavy edges are not too large a fraction of their adjacent nodes' degrees. The second plot displays the result of normalized spectral clustering. The Ncut objective favors a balanced partition with a cut that is not too large. Nevertheless, the Mincut margin is negative for this solution. The remaining plots show the best initializations by MCM(f). For this graph, initialization 1 yields the best solutions for all values of λ, and the best partition is always the same. In other examples, the best initialization varied with λ. Here, the best Mincut solution has both a relatively large margin and a low cut. Other Mincut solutions resulted in better balancing, but due to the uniform distribution larger clusters also mean that more edges are cut, and hence only a great gain by the margin can counteract the high cut value.

The difference of the results for this example, depending on the position of source and sink, is demonstrated in Figure 2.4.4. Usually, a dense subregion around one special node is cut off the rest. As λ increases, single-node clusters are disfavored in addition. From left to right, top to bottom, λ takes values 0, 1.94, 10, and 20. Particularly difficult is initialization 4, where both s and t are closely adjacent in the center. The partition must cut this heavy edge. Moreover, this graph hardly has any nodes close to the source, apart from t. Mincut separates the source, but even with higher λ only one or two nodes are added to the source cluster. Similarly, initialization 2 separates the source. To remedy the margin of -1, higher weightings of ρ lead to solutions where two other nodes join the sink, but not the closest node besides the sink. The other initializations lead to visually more satisfying partitions.

The observations for all graphs are similar to those for the illustration. In summary, of-

ten, cutting out a small and dense (relatively to the rest) cluster yields the best MCM value. As λ increases gradually, the balanced clusters become more balanced. Often, though, the balancing deteriorates for large λ. Partly, the same clusters are discovered that Ncut finds, but they are usually not the optimal ones by the MCM objective. Furthermore, as above, the initialization has a great influence on the outcome, at least for uniform distributions. In part, however, the solutions are the same, if there is some (visual) structure.

Apart from the influence of λ and s,t, the number k of neighbors in the k-nearest neighbor graph, was varied between 4 and 7 for $n = 80$, 100, 150 (2 examples each). Some tendencies for this small number of examples seem to exist: for small weights of the margin, the clusters appear to be more balanced (by number) if k is small, for large λ this is the case if k is large. This observation might be an effect of the increased cut value and degree as k grows. Furthermore, Mincut's preference of very small clusters seems higher for larger k, possibly again because of the increased cut value for larger groups, especially if the group is not well interconnected.

In general, the experiments with a uniform distribution show a significant influence of the initialization. The differences may be less strong, however, if the graph has more structure.

2.4.4 Summary

In summary, the experiments show that (i) the flow problem is only a relaxation of the problem with integer labels, bringing up the problem of the discretization of labels; (ii) the margin mainly remedies Mincut's preference for single-node clusters, but only on a local scale, favoring the cutoff of densely connected subregions; (iii) the initialization can play a great role, particularly for graphs with weak structure. In addition, the illustrations with toy graphs in Section 2.2.3 reveal that the margin defines equivalence classes of partitions.

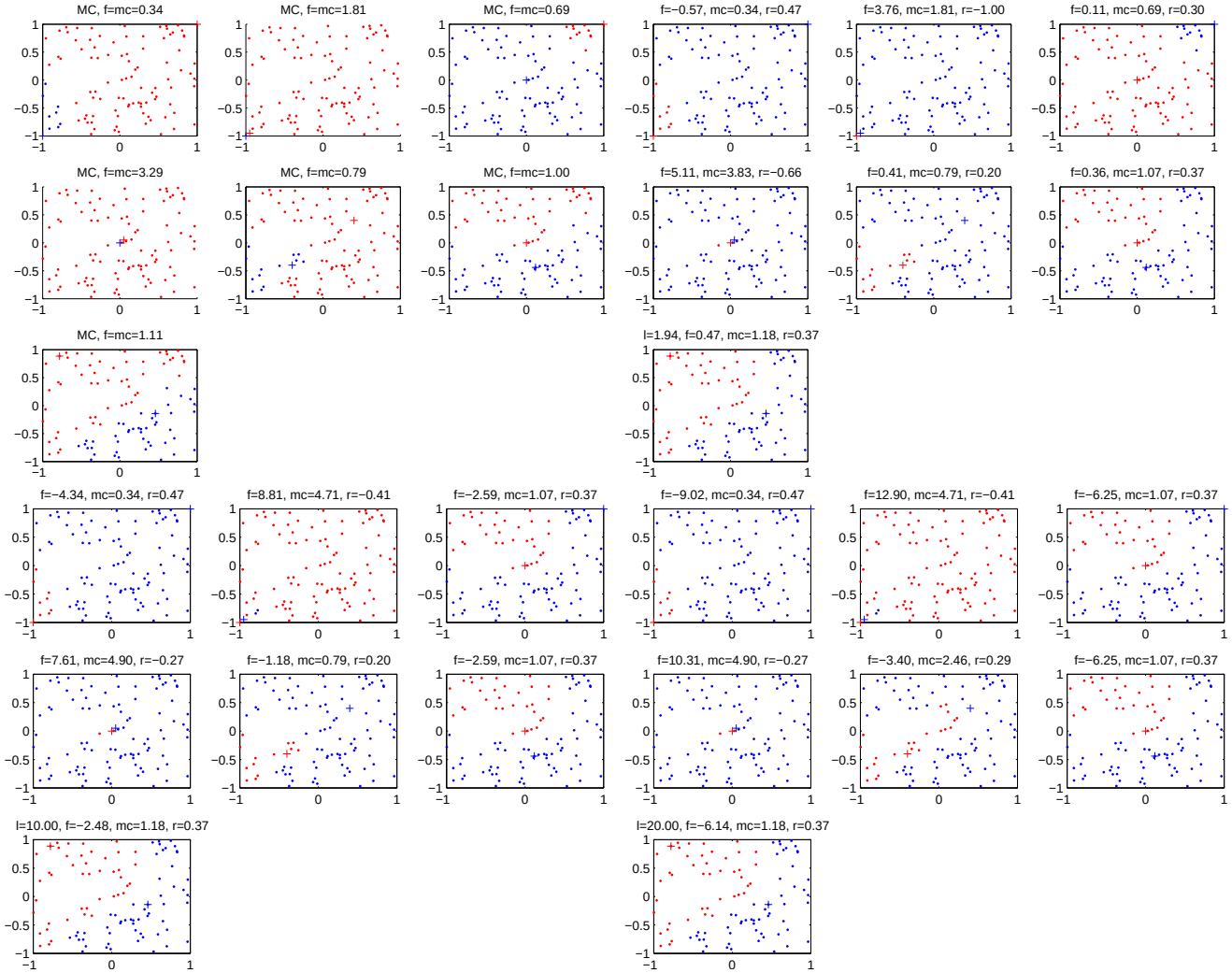


Figure 2.4.4: Optimization results for a uniform distribution for varying λ and initializations. Different source and sink nodes lead to very different partitions. The margin weight λ of 0 (upper left panel), 1.94 (top right), 10 (bottom left) and 20 (bottom right panel) is denoted l in the last plot of each panel. Each panel chows the 7 different initializations. f is the objective function value, mc the cut value and r the margin.

2.5 Discussion and Critique

Let us take a step back and put the Mincut margin into a more global picture. Are the results intuitively reasonable? Does the modification meet the initial goals? The interpretation of margins in Section 2.1 stated three advantages: robustness, simplicity and generalization ability. Before we turn to these criteria, let us make a note on the first question.

Judging by the experiments, the Mincut with margin criterion is an improvement upon the mere Mincut objective because it avoids tiny clusters, particularly of single nodes, in the outcome. Instead, it favors the cutoff of densely connected, separated subgroups in the graph. On a local scale, this tendency seems reasonable, but what about the global picture? The underlying goal was to minimize the expected risk for the entire graph. The global structure of the graph is only indirectly considered via the local interconnections and densities. Eigenvalue criteria such as used in spectral clustering, might directly include a more global picture.

Global and local stability

By its definition, the margin is large if the points may be perturbed a lot without better being assigned to another cluster. This type of robustness is closely related to the question of the stability of the partition optimizing Q_n for different n and different samples. For classification, stability of f_n usually comes with consistency (cf. Section 1.1.2). For clustering, closeness of $Q(f)$ and $Q(g)$ may not imply closeness of f and g. Our graph margins compare one partition to the ones differing by the assignment of one point, under the aspect of perturbation. But what about the swap of an entire densely connected group? Whilst the move of the group may improve the cut, the reassignment of a single member probably won't. Hence, it is possible that a modification of the edge weights below the limit of the margin leads to another partition being optimal, one that differs by more than the assignment of the node in question. Such behavior is covered by none of the above margin criteria. In that sense they are all *local*. A global criterion would consider *any* other partition to become better if the graph is perturbed. Such a measure, but for an instance rather than a partition, is the distinctness defined in Bilu and Linial [2004] (see Section 2.2.1).

Indeed, Bilu and Linial [2004] show that global stability (the optimality of the partition with regard to all other possible partitions) implies local stability, but not vice versa. (Maxcut is NP-hard even for local stability.) The demonstration by an example for our multiplicative margin can be transferred to Mincut: Consider a graph G with an optimal partition with margin ρ. From this graph, we construct another graph $G^\times$ with an analogous partition of the same global stability, but arbitrary local stability. $G^\times$ has nodes $V \times \{0, 1\}$. Connect nodes $\{v_i, 0\}$ and $\{v_i, 1\}$ by an edge of weight $\tau \max_{v \in V} d(v) = \tau \max_{v \in V} w(v, V)$ (in G), $\tau \geq 1$. Then each node has at least a margin of τ, so the margin of the partition is at least τ. By increasing τ, the local stability or margin can be modified arbitrarily, while the global stability remains the same.

By their definition of global stability, however, only the optimal Mincut can be stable, because for all other partitions, there already exists a better partition without any perturbation. Thus, such a global criterion in our "margin against perturbation" approach would only reinforce the cut minimizing the empirical error. It is thus a property of the graph along with its optimal cut, and not of any partition. So the global stability of a partition requires another definition and provides future work. In addition, the global stability is expensive to compute, due to the exponential number of cuts to compare to. The question remains open whether cheaper locality is enough. Unlike for clustering, for linear classifiers

in a "reasonable" problem ($Q(f^*) \ll 0.5$), local and global stability may be more related, because the labels already provide some structure.

Global stability is closely related to the set of almost-minimizers of Q_n and Q. If the diameter of this set does not converge to zero as $n \to \infty$, there will always be several distinct minima, and only a small perturbation will suffice to make one superior to the other. In that respect, global stability is related to the uniqueness of the optimum (within a certain close range of quality), but maybe not an appropriate criterion for picking f_n.

Generalization?

Another big issue are generalization bounds, stating that a large margin prevents overfitting, so that $Q(f_n)$ converges to $Q(f^*)$. Intuitively, the cutoff of a single node is nothing with good generalization abilities. The Mincut margin avoids this behavior.

As also shown experimentally in Section 2.4, the Mincut margin defines classes of partitions with the same ρ. The functions within one class differ by the assignment of dense subgroups. The actual complexity of one class, however, remains unclear. In some sense, the unnormalized margin is a measure for the local sharpness of a minimum. Sharp local minima, for continuous functions, are important in optimization theory for convergence properties of algorithms [see e.g. Burke and Ferris, 1993].

In that sense, the restriction via local margins may provide a type of "cover" of $\mathcal{F}$. If a partition has a positive margin, then none of its immediate neighbors (differing by the assignment of one point, that is Hamming distance one) can have a positive margin, because otherwise it would be better to swap the differing point without a change of weights. The exact type of restriction of $\mathcal{F}_\rho$ and its complexity, however, still remain to be determined. This would be necessary to study generalization bounds.

What do the remaining candidates, the almost-minimizers of $Q_n + \rho$, look like? If the clusters are separated by a low density region, then, with high probability, a partition close to f^* will have a decent margin or low cut value (or both), because it is not likely to sample a point from the low density region. If the sampling is sparse, however, there may also be other empirically good candidates that cut through a cluster and thus are "far apart" in $\mathcal{F}$ from f^*. The only remedy will be more samples, because it is more likely to sample a point from a high density region than from the separating low density region. If there is, however, symmetry in the data, so that f^* is not unique in the set of almost-minimizers of Q, then the margin may not help to identify f^*. It is then still possible, however, that $Q(f_n)$ is close to $Q(f^*)$ as n grows. A question is if the margin provides any guarantees in that regard, pointing back to the question of the complexity of $\mathcal{F}_\rho$ with respect to the magnitude of ρ. Indeed, it is easy to "mess up" the margin by an outlier point in the low density region that is connected roughly equally to both clusters.

Algorithmic restrictions

Apart from open theoretical questions, the implementation of the margin approach in this chapter brings some algorithmic restrictions. First, the optimization problem is so far restricted to $K = 2$ clusters. For more clusters, the initial two clusters may be partitioned recursively. This recursion may, however, come with a loss in the quality of the results. An extension of the MILP to more than two clusters will raise its complexity. Another drawback is the relaxation. Our aim was to facilitate the clustering problem via SLT strategies. But only the LP has polynomial complexity [Khachiyan, 1979], integer linear programs are in NP

[Papadimitriou and Steiglitz, 1982, p. 343], and thus also MILPs. The relaxation, though, brings up questions like the transformation of continuous to discrete labels.

As a last point, the result depends on the initialization of source and sink, so the algorithm should be repeated a polynomial number of times if there is no prior knowledge.

The open questions provide lots of future work. Some of them are answered for the clustering algorithm we turn to in the next chapter. It is not a relaxation and proved to be consistent.

Chapter 3

Second approach: Nearest neighbor clustering

A general strategy to achieve statistical consistency is the reduction of the class $\mathcal{F}$ of candidate functions (see Section 1.1.2). Here, we will restrict $\mathcal{F}$ in a data-dependent way: we will divide the graph into neighborhoods around $O(\log n)$ seed nodes and require the partition function to be constant within each neighborhood. For a fixed neighborhood structure, only a polynomial number of candidate partitions remains. Hence, they can be enumerated in polynomial time to find the best amongst them, and the NP hard optimization problem is transformed into one in P. This transformation is based on "reasonable" strategies motivated by statistical learning theory, rather than on (often uncontrollable) heuristics or relaxations. A branch and bound approach will further limit the average-case complexity. Nearest neighbor clustering is compatible with a variety of cut criteria such as Ncut, Rcut and WSS. Above all, this algorithm can be proved to be statistically consistent.

In experiments, we will compare this approach to spectral clustering and the k-means algorithm, with respect to the clustering of a graph and its generalization. In addition, we will examine different distance functions for constructing the neighborhoods, and attempt an analysis of the impact of the selection of seeds.

Parts of this chapter have been published in von Luxburg et al. [2007a].

3.1 Nearest neighbor clustering

Nearest neighbor clustering realizes the reduction of the function class by a restriction to functions that are constant within cells of the data space. We will now outline the basic idea, its realization and the complexity of the resulting algorithm.

In the following, we assume to be given a set of data points $\mathcal{Z}_n = \{X_1, \ldots, X_n\}$ and pairwise distances $d_{ij} = d(X_i, X_j)$ or pairwise similarities $s_{ij} = s(X_i, X_j)$. Let Q_n be the finite sample quality function to optimize on the sample. To follow the statistical learning theory approach outlined above, we have to optimize Q_n over a "small" set $\mathcal{F}_n$ of partitions of $\mathcal{Z}_n$. Essentially, we put three requirements on $\mathcal{F}_n$: First, the number of functions in $\mathcal{F}_n$ should be at most polynomial in n. Second, in the limit of $n \to \infty$, the class $\mathcal{F}_n$ should be rich enough to approximate any measurable partition of the underlying space. Third, in order to perform the optimization, we need to be able to enumerate all members of this class. So the function class $\mathcal{F}_n$ should be "constructive" in some sense. A convenient choice

satisfying all those properties is the class of "nearest neighbor partitions". To define this class, we partition the graph into $m \ll n$ neighborhood regions. The class then comprises all functions $f : \mathcal{X} \to \{1, \ldots, K\}$ that are constant within the neighborhoods. Of those functions, we choose the one minimizing Q_n.

Algorithm

We construct the neighborhoods as a Voronoi tessellation of the graph. To this end, we randomly sample m seed points $X_{s_1}, \ldots X_{s_m}$ among the n data points to be the "centers" of the cells. Each point is then assigned to its closest seed. The points assigned to seed X_{sj} form, together with the seed, the neighborhood set Z_j. Note that this definition can be stated both for similarity and dissimilarity data. In the case of dissimilarity data we build the Voronoi cells based on the nearest neighbor relation, while in case of similarity values we build the cells based on the "most similar neighbor" relation.

Obviously, the function class $\mathcal{F}_n$ contains $O(K^m)$ functions, which is polynomial in n if the number m of seeds satisfies $m = m(n) = O(\log n)$. Given $\mathcal{F}_n$, the simplest polynomial-time optimization algorithm is the brute-force approach to evaluate $Q_n(f)$ for all $f \in \mathcal{F}_n$ and choose the solution $f_n = \operatorname{argmin}_{f \in \mathcal{F}_n} Q_n(f)$. We call the resulting clustering the *nearest neighbor clustering* and denote it by $\mathrm{NNC}(Q_n)$.

The complete algorithm then includes the following steps:

1. Randomly choose m seed points among the n data points $X_1, \ldots, X_n$.
2. Assign each non-seed point to its closest seed point to construct the neighborhoods Z_j, $1 \leq j \leq m$. The neighborhoods form "super-nodes" Z_j.
3. Choose f_n minimizing Q_n from $\mathcal{F}_n$; f_n assigns a label to each Z_j. We will optimize the partition on a contracted graph where the neighborhoods are represented by super-nodes.
4. The points X_i are labeled according to their assigned neighborhood.

Complexity

Let us take a closer look at the complexity of this algorithm. For a fixed set of seeds, it runs in time $O(nm + (q(n,m) + r(n,m))K^m)$. The first term covers the construction of the Voronoi cells in a naive implementation. The second part comes from computing and evaluating all possible partitions. Let $q(n,m)$ denote the complexity of evaluating $Q_n(f)$ for a fixed function f [1]. For Ncut, $q(n,m) = O(Kn^2)$, which may be reduced to $O(Km^2)$ after a contraction that is done once and costs $O(n^2)$ (cf. Subsection 3.4.1). The term $r(n,m)$ represents the time to compute one partition. The enumeration of $\mathcal{F}$ then costs $O(K^m r(n,m))$. We can naively enumerate all partitions in $\mathcal{F}_n$ by a count from 0 to K^m in a K-ary system, assigning labels $0, \ldots, K-1$. Such a count takes

$$\sum_{\ell=1}^{m} K^\ell = \frac{K^{m+1} - 1}{K - 1} = O\left(\frac{K^{m+1}}{K-1}\right)$$

steps, because the first digit is changed K times, the second K^2 times and so on. This results in an amortized cost of $r(n,m) = O\left(\frac{K}{K-1}\right) = O\left(1 + 1/(K-1)\right)$ per candidate partition.

[1] In fact, q may also depend on the number of edges, which is at most n^2.

Note that in such a count, some partitions are equivalent under a renaming of labels, so in practice we can actually restrict the count to those numbers where all labels occur and their first appearance is in ascending order from left to right.

For example, if we choose $m = c\log(n)$ for a small constant c, we obtain polynomial runtime in $O(n\log(n) + (r(n, \log n) + q(n, \log n))n^{c \log K})$. The $n \times n$ distance matrix is considered a precomputed input here. Distances will be discussed in Section 3.3.

Stated in a general way like above, the NNC strategy is compatible with a variety of quality functions. One popular criterion is the Normalized Cut objective. Apart from Ncut, we use RatioCut and the k-means objective (WSS) in experiments. The proof of consistency comprises all these criteria.

3.2 NNC is consistent

Nearest neighbor clustering is statistically consistent for many clustering quality functions, that means $Q(f_n)$ converges to $Q(f^*)$ in probability. A complete proof may be found in Bubeck and von Luxburg [2007], von Luxburg et al. [2007a][2]. Here we only provide a short summary of the main results.

The theorems require the introduction of some further notation. First, we define a predicate $A(f)$ for each clustering function $f : \mathbb{R}^d \to \{1, \ldots, K\}$ that implements some assumption. It may, for instance, be true if all clusters have a certain minimal size. The predicate $A_n(f)$ is an "estimator" of $A(f)$ based on a finite sample only. The predicates will define membership in the function classes. Let $m := m(n) \leq n$ be the number of seeds used in nearest neighbor clustering. To simplify notation we assume in this section that the seeds are the first m data points; all results remain valid for any other (even random) choice of seeds. As data space we use $\mathcal{X} = \mathbb{R}^d$. We define:

$$\mathrm{NN}_m(x) := \mathrm{NN}_{m(n)}(x) := \mathrm{argmin}_{y \in \{X_1, \ldots, X_m\}} \|x - y\| \quad (\text{ for } x \in \mathbb{R}^d)$$

$$\mathcal{F} := \{f : \mathbb{R}^d \to \{1, \ldots, K\} \mid f \text{ continuous P-a.e. and } A(f) \text{ true}\}$$

$$\mathcal{F}_n := \mathcal{F}_{X_1, \ldots, X_n} := \{f : \mathbb{R}^d \to \{1, \ldots, K\} \mid f \text{ satisfies } f(x) = f(\mathrm{NN}_m(x)), \text{ and } A_n(f) \text{ is true}\}$$

$$\widetilde{\mathcal{F}}_n := \bigcup_{X_1, \ldots, X_n \in \mathbb{R}^d} \mathcal{F}_{X_1, \ldots, X_n}.$$

So $\mathcal{F}_n$ is the space of nearest neighbor partitions based on a specific sample, and $\widetilde{\mathcal{F}}_n$ the union of such spaces over all possible samples.

Furthermore, let $Q : \mathcal{F} \to \mathbb{R}$ be the quality function we aim to minimize, and $Q_n : \mathcal{F}_n \to \mathbb{R}$ an estimator of this quality function on a finite sample. With this notation, the true clustering f^* on the underlying space and the nearest neighbor clustering f_n introduced in the last section are given by

$$f^* \in \mathrm{argmin}_{f \in \mathcal{F}} Q(f) \qquad \text{and} \qquad f_n \in \mathrm{argmin}_{f \in \mathcal{F}_n} Q_n(f).$$

In addition, the proof involves the optimum function achievable in the subspace $\mathcal{F}_n$ and the restriction of the optimal f^* to satisfy the nearest neighbor rule:

$$f_n^* \in \mathrm{argmin}_{f \in \mathcal{F}_n} Q(f) \qquad \text{and} \qquad \widetilde{f^*}(x) := f^*(\mathrm{NN}_m(x)).$$

[2]The proof in Bubeck and von Luxburg [2007] is more general with a variant of the predicate, but the techniques are the same. The proof in von Luxburg et al. [2007b] is the same as the one sketched here. Note that the proof is by Bubeck and von Luxburg, and thus the only section in this chapter that is not my work.

As distance function between different clusterings f, g we will use

$$\mathrm{L_n}(f,g) := P\{f(X) \neq g(X) \mid X_1, \ldots, X_n\}$$

(we need the conditioning in case f or g depend on the data, it has no effect otherwise).

Endowed with the notation, we can turn to the theorems.

Theorem 2 (Consistency of nearest neighbor clustering)**.** *Let $(X_i)_{i\in\mathbb{N}}$ be a sequence of points drawn i.i.d. according to some probability measure P on $\mathbb{R}^d$, and $m := m(n)$ the number of seed points used in nearest neighbor clustering. Let $Q : \mathcal{F} \to \mathbb{R}$ be a clustering quality function, $Q_n : \widetilde{\mathcal{F}}_n \to \mathbb{R}$ its estimator, and $A(f)$ and $A_n(f)$ some predicates. Assume that:*

1. *$Q_n(f)$ is a consistent estimator of $Q(f)$ which converges sufficiently fast:*

$$\forall \varepsilon > 0, \;\; K^m (2n)^{(d+1)m^2} \sup_{f\in\widetilde{\mathcal{F}}_n} P\{|Q_n(f) - Q(f)| > \varepsilon\} \to 0.$$

2. *$A_n(f)$ is an estimator of $A(f)$ which is "consistent" in the following way:*

$$P\{A_n(\widetilde{f}^*) \;\; true\} \to 1 \qquad \text{and} \qquad P\{A(f_n) \;\; true\} \to 1.$$

3. *Q is uniformly continuous with respect to the distance L_n between $\mathcal{F}$ and $\mathcal{F}_n$:*

$$\forall \varepsilon > 0 \; \exists \delta(\varepsilon) > 0 \; \forall f \in \mathcal{F} \; \forall g \in \mathcal{F}_n : \quad \mathrm{L_n}(f,g) \leq \delta(\epsilon) \implies |Q(f) - Q(g)| \leq \varepsilon.$$

4. *$lim_{n\to\infty} m(n) = +\infty$.*

Then nearest neighbor clustering as introduced in Section 3.1 is weakly consistent, that is $Q(f_n) \to Q(f^*)$ in probability.

Proof. (Sketch) The proof bounds the probability of divergence, $P(|Q(f_n) - Q(f^*)| \geq \varepsilon)$ by repeatedly splitting it into terms that are bounded separately. We first remove the absolute value, then we split one part into estimation and approximation error that can be handled independently.

First, split the absolute value into its two sides:

$$P\{|Q(f_n) - Q(f^*)| \geq \varepsilon\} \leq P\{Q(f_n) - Q(f^*) \leq -\varepsilon\} + P\{Q(f_n) - Q(f^*) \geq \varepsilon\}.$$

As a consequence of Assumption (2), $f_n \in \mathcal{F}$ with high probability, so the first term on the right hand side converges to 0.

The main work consists in bounding the second term. By the triangle inequality with f_n^*, we get the estimation and approximation errors:

$$P\big\{Q(f_n) - Q(f^*) \geq \varepsilon\big\} \;\leq\; P\big\{Q(f_n) - Q(f_n^*) \geq \varepsilon/2\big\} \;+\; P\big\{Q(f_n^*) - Q(f^*) \geq \varepsilon/2\big\}.$$

Estimation Error. One can show that

$$P\{Q(f_n) - Q(f_n^*) \geq \varepsilon/2\} \;\leq\; P\{\sup_{f\in\mathcal{F}_n} |Q_n(f) - Q(f)| \geq \varepsilon/4\}.$$

Even though the right hand side resembles the standard quantities often considered in statistical learning theory, it is not straightforward to bound as we do not assume that $Q(f) = \mathbb{E}Q_n(f)$. To circumvent the further complication of the data-dependency of $\mathcal{F}_n$, we replace it by the larger class $\widetilde{\mathcal{F}}_n$, which is not data dependent. Using symmetrization by a

ghost sample (cf. [Devroye et al., 1996, Sec. 12.3]), we then move the supremum out of the probability:

$$P\Big\{ \sup_{f\in\mathcal{F}_n} |Q_n(f) - Q(f)| \geq \varepsilon/4 \Big\} \;\leq\; 2S_K(\widetilde{\mathcal{F}}_n, 2n) \frac{\sup_{f\in\widetilde{\mathcal{F}}_n} P\big\{|Q_n(f) - Q(f)| \geq \varepsilon/16\big\}}{\inf_{f\in\widetilde{\mathcal{F}}_n} P\big\{|Q_n(f) - Q(f)| \leq \varepsilon/8\big\}} \tag{3.2.1}$$

The unusual denominator in Equation (3.2.1) emerges in the symmetrization step as we do not assume $Q(f) = \mathbb{E}Q_n(f)$. The bound also involves the shattering coefficient $S_K(\widetilde{\mathcal{F}}_n, 2n)$ (cf. Section 1.1.2). It is well known [e.g. Devroye et al., 1996, Sec. 21.5] that the number of Voronoi partitions of n points using m cells in $\mathbb{R}^d$ is bounded by $n^{(d+1)m^2}$, hence the number of nearest neighbor clusterings into K classes is bounded by $S_K(\widetilde{\mathcal{F}}_n, n) \leq K^m n^{(d+1)m^2}$.

Hence, Assumption (1) implies that for fixed ε and $n \to \infty$ the right hand side of Equation (3.2.1), and thus also the estimation error, converges to 0.

Approximation Error. For the approximation error, we replace f_n^* by $\tilde{f}^*$. If $A_n(\widetilde{f}^*)$ is true, then $\widetilde{f}^* \in \mathcal{F}_n$, and by the definition of f_n^* we have

$$\begin{aligned} &Q(f_n^*) - Q(f^*) \leq Q(\widetilde{f}^*) - Q(f^*) \quad \text{and thus} \\ &P\big\{Q(f_n^*) - Q(f^*) \geq \varepsilon\big\} \;\leq\; P\{A_n(\widetilde{f}^*) \text{ false}\} \;+\; P\big\{\widetilde{f}^* \in \mathcal{F}_n \text{ and } Q(\widetilde{f}^*) - Q(f^*) \geq \varepsilon\big\}. \end{aligned}$$

The first term on the right hand side converges to 0 by Assumption (2). The second expression can be bounded via the distance L_n, using Assumption (3):

$$P\big\{\widetilde{f}^* \in \mathcal{F}_n,\; Q(\widetilde{f}^*) - Q(f^*) \geq \varepsilon\big\} \;\leq\; P\big\{Q(\widetilde{f}^*) - Q(f^*) \geq \varepsilon\big\} \;\leq\; P\big\{\,\mathrm{L_n}(f^*, \widetilde{f}^*) \geq \delta(\varepsilon)\big\}.$$

Techniques from Fritz [1975] help to show that if n is large enough, then the distance between a function $f \in \mathcal{F}$ evaluated at x and the same function evaluated at $\mathrm{NN}_m(x)$ is small. Namely, for any $f \in \mathcal{F}$ and any $\varepsilon > 0$ there exists some $b(\delta(\varepsilon)) > 0$ which does not depend on n and f such that

$$P\{\mathrm{L_n}(f, f(\mathrm{NN}_m(\cdot))) > \delta(\varepsilon)\} \;\leq\; (2/\delta(\varepsilon)\, \exp(-mb(\delta(\varepsilon))).$$

The quantity $\delta(\varepsilon)$ has been introduced in Assumption (3). Assumption (4) ensures that for every fixed ε, the right hand side converges to 0, making the approximation error vanish. □

Let us apply the general theorem to particular objective functions. We sketch the proof for Ncut and only mention the result for other objectives in Theorem 4. Let the similarity function $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ be upper bounded by a constant C. For a clustering $f : \mathbb{R}^d \to \{1, \ldots, K\}$ denote by $f_k(x) := \mathbb{1}_{f(x)=k}$ the indicator function of the k-th cluster. Define the empirical and true cut, volume, and Normalized cut as follows:

$$\begin{aligned} &\mathrm{cut}_n(f_k) := \tfrac{1}{n(n-1)} \textstyle\sum_{i,j=1}^n f_k(X_i)(1 - f_k(X_j))s(X_i, X_j) \\ &\mathrm{cut}(f_k) := \mathbb{E}_{X,Y}\big(f_k(X)(1 - f_k(Y))s(X, Y)\big) \\ &\mathrm{vol}_n(f_k) := \tfrac{1}{n(n-1)} \textstyle\sum_{i,j=1}^n f_k(X_i)s(X_i, X_j) \qquad\qquad \mathrm{vol}(f_k) := \mathbb{E}_{X,Y}\big(f_k(X)s(X, Y)\big) \\ &\mathrm{Ncut}_n(f) := \textstyle\sum_{k=1}^K \frac{\mathrm{cut}_n(f_k)}{\mathrm{vol}_n(f_k)} \qquad\qquad \mathrm{Ncut}(f) := \textstyle\sum_{k=1}^K \frac{\mathrm{cut}(f_k)}{\mathrm{vol}(f_k)} \end{aligned}$$

Unlike the empirical risk in classification, the quality estimator $Q_n = \mathrm{Ncut}_n$ for Ncut, is not unbiased: $\mathbb{E}\,\mathrm{Ncut}_n(f) \neq \mathrm{Ncut}(f)$. Hence, in the proof we will resort to its unbiased parts instead: $\mathbb{E}\,\mathrm{cut}_n(f) = \mathrm{cut}(f)$ and $\mathbb{E}\,\mathrm{vol}_n(f) = \mathrm{vol}(f)$. The predicate will embody a constraint

on cluster sizes: fix a constant $a > 0$, a sequence $(a_n)_{n\in\mathbb{N}}$ with $a_n \geq a_{n+1}$ and $a_n \to a$ and define

$$\begin{aligned} A(f) \text{ is true } &:\Longleftrightarrow \text{ vol}(f_k) > a \ \ \forall k = 1, \ldots, K \\ A_n(f) \text{ is true } &:\Longleftrightarrow \text{ vol}_n(f_k) > a_n \ \ \forall k = 1, \ldots, K \end{aligned} \tag{3.2.2}$$

Theorem 3 (Consistency of **NNC(Ncut$_n$)**). *Let $(X_i)_{i\in\mathbb{N}}$ be a sequence of points drawn i.i.d. according to some probability measure P on $\mathbb{R}^d$ and $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ be a similarity function which is upper bounded by a constant C. Let $m := m(n)$ be the number of seed points used in nearest neighbor clustering, $a > 0$ an arbitrary constant, and $(a_n)_{n\in\mathbb{N}}$ a monotonically decreasing sequence with $a_n \to a$. Then nearest neighbor clustering using $Q := \text{Ncut}$, $Q_n := \text{Ncut}_n$, and A and A_n as defined in (3.2.2) is weakly consistent if $m(n) \to \infty$ and $m^2 \log n/(n(a - a_n)^2) \to 0$.*

Proof. To check that all assumptions of Theorem 2 are satisfied, we first establish that

$$\{|\operatorname{cut}_n(f_k) - \operatorname{cut}(f_k)| \leq a\varepsilon\} \cap \{|\operatorname{vol}_n(f_k) - \operatorname{vol}(f_k)| \leq a\varepsilon\} \subset \left\{ \left| \frac{\operatorname{cut}_n(f_k)}{\operatorname{vol}_n(f_k)} - \frac{\operatorname{cut}(f_k)}{\operatorname{vol}(f_k)} \right| \leq 2\varepsilon \right\}.$$

Applying the McDiarmid inequality to cut_n and vol_n, respectively, yields that for all $f \in \widetilde{\mathcal{F}}_n$

$$P\{|\operatorname{Ncut}(f) - \operatorname{Ncut}_n(f)| > \varepsilon\} \leq 4K \exp\left(-\frac{na^2\varepsilon^2}{8C^2K^2} \right).$$

Together with $m^2 \log n/(n(a - a_n)^2) \to 0$ this shows Assumption (1) of Theorem 2. The proof of Assumption (2) is a bit technical, but in the end also follows by applying the McDiarmid inequality to $\text{vol}_n(f)$. Assumption (3) follows by establishing that for $f \in \mathcal{F}$ and $g \in \mathcal{F}_n$ we have

$$|\operatorname{Ncut}(f) - \operatorname{Ncut}(g)| \leq \frac{4CK}{a} L_n(f, g).$$

□

As examples for other quality functions compatible with consistent NNC, consider RatioCut, Within-sum-of-squares, and the ratio of between- and within-cluster similarity:

$$\begin{aligned} \text{RatioCut}_n(f) &:= \textstyle\sum_{k=1}^K \frac{\text{cut}_n(f_k)}{n_k} & \text{RatioCut}(f) &:= \textstyle\sum_{k=1}^K \frac{\text{cut}(f_k)}{\mathbb{E} f_k(X)} \\ \text{WSS}_n(f) &:= \textstyle\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K f_k(X_i) \|X_i - c_{k,n}\|^2 & \text{WSS}(f) &:= \mathbb{E} \textstyle\sum_{k=1}^K f_k(X) \|X - c_k\|^2 \\ \text{BW}_n &:= \textstyle\sum_{k=1}^K \frac{\text{cut}_n(f_k)}{\text{vol}_n(f_k) - \text{cut}_n(f_k)} & \text{BW} &:= \textstyle\sum_{k=1}^K \frac{\text{cut}(f_k)}{\text{vol}(f_k) - \text{cut}(f_k)} \end{aligned}$$

Here $n_k := \sum_i f_k(X_i)/n$ is the fraction of points in the k-th cluster, and $c_{k,n} := \sum_i f_k(X_i) X_i/(nn_k)$ and $c_k := \mathbb{E} f_k(X) X/\mathbb{E} f_k(X)$ are the empirical and true cluster centers.

Theorem 4 (Consistency of **NNC(RatioCut$_n$)**, **NNC(WSS$_n$)**, and **NNC(BW$_n$)**). *Let f_n and f^* be the empirical and true minimizers of nearest neighbor clustering using* RatioCut_n, WSS_n, *or* BW_n, *respectively. Then, under assumptions similar to the ones in Theorem 3, we have* $\text{RatioCut}(f_n) \to \text{RatioCut}(f^*)$, $\text{WSS}(f_n) \to \text{WSS}(f^*)$, *and* $\text{BW}(f_n) \to \text{BW}(f^*)$ *in probability. For details, see von Luxburg et al. [2007b].*

3.3 Distance functions

An important part from a practical viewpoint is the construction of the neighborhoods. Each node in the graph is assigned to its closest seed. To measure closeness, we need a distance measure between nodes in the graph. Its importance lies in the determination of the structure of the neighborhood cells and hence its impact on the solution by defining $\mathcal{F}_n$.

The proof above is based on the Euclidean distance. The result might be extended to other distances, particularly those that behave "reasonably" in the limit, that means as $n \to \infty$, each point has a nonempty ε-neighborhood, should be applicable as well[3].

In the following, we will introduce a variety of distances that we used, the Euclidean distance as well as graph-specific distances motivated by Markov chains, random walks and electrical networks.

To remain in P, the matrix of distances between seed nodes and all other points must be constructible in polynomial time. By the formulas given below, all distances can be computed in polynomial time, and hence also all the $O(n \log n)$ distances required for the construction of the neighborhoods. Some of the distances below require eigenvalues and -vectors. Note that the arithmetic complexity of solving the eigenproblem for an $n \times n$ matrix to a relative precision of 2^{-b} is in $O(n^3 + n(\log^2 n) \log b)$ [Pan and Chen, 1999]. (For an overview of eigenvector methods for symmetric matrices, see e.g. Dhillon [1997, Ch. 2].)

Euclidean distance

The Euclidean distance between two points $X, Y \in \mathbb{R}^d$ is defined as

$$\sqrt{\sum_{i=1}^{d} (X^{(i)} - Y^{(i)})^2},$$

where $X^{(i)}$ and $Y^{(i)}$ denote the i-th entry in the vectors X and Y, respectively. It is a commonly used distance in real spaces and generates circle-shaped neighborhoods. If the input data is given as a graph with edge weights, however, it needs to be embedded in some Euclidean space $\mathbb{R}^d$ for applicability of this distance. The embedding can then have a great influence on the result, and there are various possibilities to choose it and its dimension d. Hence, other distances that account for the structure of the graph may be better suited.

Commute or Resistance distance and its interpretations

The commute distance is commonly used for graphs and has been investigated in a number of different respects. Two interpretations complement each other: the expected traveling time in a random walk, and the resistance in an electrical network. The former concept is related to Markov chains.

First consider distance in light of a random walk on the given graph. Given a start node, we randomly choose an adjacent node to which we transfer. The random walk consists of a sequence of such transitions. In a finite, undirected graph, such a walk is actually a finite, time-reversible Markov chain [Lovász, 1993]. Properties of and connections between random walks, Markov chains and electrical networks are discussed in Lovász [1993], Bollobás [1998, Ch. 9] and Aldous and Fill [2001, Ch. 3]. Starting at node X_i, the probability to directly

[3] If, in the limit $n \to \infty$, some points become separated, the distance is probably not appropriate for all objectives in extreme cases (see the counterexample in Figure 3.3.1, described below).

get to node X_j is

$$p(X_i, X_j) = \frac{w(X_i, X_j)}{d(X_i)},$$

where $d(X_i)$ is the degree of X_i. In such a setting, the commute distance $C(X_i, X_j)$ between X_i and X_j is the expected number of steps it takes to travel from X_i to X_j and back.

Let us look at some ways to compute the distance. Direct formulas have been proved in terms of the normalized and unnormalized Laplacian. We outline both alternatives.

1. Let $L_{rw} = I - D^{-1}W$ be the normalized Laplacian of the graph, where D is a diagonal matrix with $D(i,i) = w(X_i, V)$. Lovász [1993] shows that

$$C(X_i, X_j) = 2\,\mathrm{vol}(G) \sum_{k=2}^{n} \frac{1}{\lambda_k} \left(\frac{v_{kj}}{\sqrt{d(X_j)}} - \frac{v_{ki}}{\sqrt{d(X_i)}} \right)^2 \tag{3.3.1}$$

   where λ_k is the k-th eigenvalue of L_{rw} and v_{ki} the i-th entry in the k-th eigenvector of L_{rw}. Here and in the following, we number eigenvalues in nondescending order, so $\lambda_1 = 0$. The volume is $\mathrm{vol}(G) = w(V, V)$. This formula is closely related to Expression (3.3.3) for the hitting time, because $C(X_i, X_j) = H(X_i, X_j) + H(X_j, X_i)$.

   Formula (3.3.1) shows that the commute time depends on the difference of the eigenvector entries, weighted by the inverse of the corresponding eigenvalue. The Fiedler vector will hence have the greatest influence. Spectral clustering usually partitions the points into clusters according to their entries in the Fiedler vector, but only considers this second eigenvector.

2. According to Klein and Randić [1993] (see also [Gutman and Xiao, 2004, Xiao and Gutman, 2003]), the commute distance can also be computed via the pseudoinverse of the (unnormalized) graph Laplacian $L = D - W$:

$$C(X_i, X_j) = L^\dagger(i,i) + L^\dagger(j,j) - L^\dagger(i,j) - L^\dagger(j,i) = \sum_{k=2}^{n} \frac{1}{\lambda_k} (v_{ki} - v_{kj})^2. \tag{3.3.2}$$

   Here, λ_k is the k-th eigenvalue of L and v_{ki} the i-th entry of the k-th eigenvector of L.

Let us turn to some further relations and interpretations of the commute distance. For an unweighted graph, the commute distance is related to the expected sojourn time $S_k(X_i \to X_j)$, the expected number of times X_k is visited before we reach node X_j, starting from X_i: $C(X_i, X_j)/(2\,\mathrm{vol}(G)) = S_i(X_i \to X_j)/d(X_i)$ [Bollobás, 1998, p. 315]. It seems reasonable that there is a connection between the commute time and the expected number of revisits of X_i before X_j is reached. This normalized commute time also equals the expected number of times any edge is traversed on a walk from X_i to X_j and back [Bollobás, 1998, p. 315].

Apart from the random walk view, the distance may be interpreted with respect to electrical networks and springs. Consider the graph as an electrical network, where the inverse edge weights define resistances between the nodes. Then the commute time between nodes X_i and X_j corresponds to the resistance $R(X_i, X_j)$ between X_i and X_j in the circuit: $C(X_i, X_j) = 2\,\mathrm{vol}(G) R(X_i, X_j)$ [Lovász, 1993]. Resistance distance is also studied in Klein and Randić [1993].

Furthermore, an unweighted graph can be viewed as a system of springs with unit Hooke constant. Nail nodes X_i and X_j down at positions 1 and 0 on the real line. The graph will find its equilibrium. The force pulling the nails is then the conductance $1/R(X_i, X_j) = 2\,\mathrm{vol}(G)/C(X_i, X_j)$. The energy of the system is $1/(2R(X_i, X_j))$ [Lovász, 1993].

Returning to NNC and its neighborhood structure, note that the commute distance may implicitly balance neighborhood sizes. This tendency could be favorable for criteria that include balancing, such as Ncut or RatioCut. Let us detail the intuition behind the balance conjecture. Neighborhoods are constructed around seed nodes. A node X_i is assigned to the seed that it reaches fastest by expectation, including the return time to X_i. Or, from the other perspective, a seed "gets" the nodes that it reaches first, if the expected return time is also considered. If a seed node X_s has many neighbor nodes (in a high-density region), the connection to each will only form a small part of its degree, and hence the transition probability to each neighbor is relatively low. This decreased probability leads to a larger expected time to reach any adjacent node and other nodes "behind" it, unless those nodes are well-connected to most direct neighbors of X_s. If the target node is well-connected to many other nodes, then again the probability of "distraction" is higher for the return. Hence, a neighborhood of such a well-connected, "heavy" seed may spread in many directions, but not too far, whereas a seed with few neighbors (in a low-density region) reaches them "faster" and thus may cover a wider area. This is, however, just a conjecture without any mathematical proof.

Hitting time or Access time

The hitting or access time $H(X_i, X_j)$ in a Markov chain or random walk is the expected number of steps before node X_j is visited, starting at node X_i [Lovász, 1993]. Hence, the commute time is the hitting time from X_i to X_j and back: $C(X_i, X_j) = H(X_i, X_j) + H(X_j, X_i)$.

Like the commute distance, it can be computed via the graph Laplacian, as described in Lovász [1993]. The matrix H of hitting times is the solution of the equation $L_{rw}H = \mathbf{1}_{n\times n} - 2\,\mathrm{vol}(G)D$. Even though L_{rw} is singular, we know that $H(i,i) = 0$ for all nodes X_i. Hence, we compute H by subtracting from each column of $\widehat{H} = L_{rw}^{\dagger}(\mathbf{1}_{n\times n} - 2\,\mathrm{vol}(G)D)$ its diagonal entry.

For an unweighted graph (weights in $\{0,1\}$), the hitting time can be expressed as

$$H(X_s, X_t) = 2\,\mathrm{vol}(G)\sum_{k=2}^{n}\frac{1}{\lambda_k}\left(\frac{v_{kt}^2}{d(X_t)} - \frac{v_{ks}v_{kt}}{\sqrt{d(X_s)d(X_t)}}\right), \tag{3.3.3}$$

where λ_k is the k-th eigenvalue of L_{rw} and v_{ks} the s-th entry in the k-th eigenvector of L_{rw}.

Note that, in contrast to the commute distance, the hitting time is not symmetric: in general $H(X_i, X_j)$ and $H(X_j, X_i)$ differ.

Similar to the commute distance, the hitting time can be expressed in terms of electrical networks [Chandra et al., 1996]. In the circuit described by an unweighted graph, inject $d(X_j)$ units of current into each node $X_j \in V$, and remove $2|E|$ units from X_i. Then $H(X_i, X_j)$ denotes the voltage $\widehat{V}(X_j) - \widehat{V}(X_i)$ at X_j for all $X_j \in V$. By a change of signs, $H(X_j, X_k)$ is the voltage at X_j relative to X_k (i.e. $\widehat{V}(X_j) - \widehat{V}(X_k)$) if $2|E|$ units of current are injected at X_j and $d(X_k)$ extracted at each $X_k \in V$. A superposition of the first and second current leads to a net current of $2|E|$ from X_j to X_i. With this current, the

voltage difference of X_j and X_i is $H(X_i, X_j) + H(X_j, X_i) = C(X_j, X_i) = 2|E|R(X_j, X_i)$, the commute distance or resistance, in conformity with Ohm's law.

Note that the hitting time offers different ways to construct the neighborhoods, because it is not symmetric. One could either take the distance from a seed to a point or vice versa. In the experiments below we chose the latter, which conceptually corresponds to a generalization from the point's perspective: starting a random walk at the node, assign it to the seed that it reaches first, by expectation.

Variations based on the Commute distance

In some cases, the eigenvectors of the unnormalized Laplacian may approach Dirac functions. This can happen for the eigenvectors whose corresponding eigenvalues are not significantly below the minimum degree in the graph [von Luxburg et al.], and hence more often towards the end of the spectrum. Figure 3.3.1 illustrates this behavior for a toy example[4]. The illustrated results become stronger as the graph approaches a clique with uniform weights.

In a Dirac vector, the entry for one point is very different from all other entries. If the weighting $1/\lambda_k$ for the corresponding eigenvector is not small enough, this difference will lead to a large commute distance of this point with all others, because the distance is the sum of the differences in the eigenvector entries (Equation (3.3.2)). As an effect, this point, "far away" from all others, is likely to end up in a cluster by itself.

Even though the eigenvectors of the normalized Laplacians do not converge to Dirac functions, the eigenvectors can become noisy towards the end of the spectrum (see Figure 3.3.1). They are inherently less smooth than the earlier ones, as they represent higher oscillations. This noisiness, along with the property of the eigenvectors to become more similar towards the end of the spectrum, could lead to numerical inaccuracies. Thus, the impact of those vectors could distort the result when computing a distance as in Equation (3.3.2).

If the differences of the eigenvector entries for each point are weighted by $1/\lambda_k - 1/\lambda_n$, then the end of the spectrum is less influential. The corresponding eigenvectors are considered less, but the relative impact of the first vectors remains the same, as in a subtractive normalization. In comparison, spectral clustering only takes the first K eigenvectors into account, but with equal weights.

We introduce two variants of the commute distance, with the new weighting. The normalized commute distance (ND) is based on the non-symmetric normalized Laplacian, $L_{rw} = I - D^{-1}W$. The symmetric normalized Laplacian, $L_{sym} = I - D^{-1/2}WD^{-1/2}$, on the other hand, forms the basis of the symmetric normalized commute distance (SND):

$$\begin{aligned}
\mathrm{ND}(X_i, X_j) &= 2\,\mathrm{vol}(G) \sum_{k=2}^{n} \left(\frac{1}{\lambda_k(L_{rw})} - \frac{1}{\lambda_n(L_{rw})} \right) \left(v_{ki}(L_{rw}) - v_{kj}(L_{rw}) \right)^2 \\
\mathrm{SND}(X_i, X_j) &= 2\,\mathrm{vol}(G) \sum_{k=2}^{n} \left(\frac{1}{\lambda_k(L_{sym})} - \frac{1}{\lambda_n(L_{sym})} \right) \left(\frac{v_{ki}(L_{sym})}{\sqrt{d(X_i)}} - \frac{v_{kj}(L_{sym})}{\sqrt{d(X_j)}} \right)^2 \\
&= C(X_i, X_j) - \frac{2\,\mathrm{vol}(G)}{\lambda_n(L_{sym})} \sum_{k=2}^{n} \left(\frac{v_{ki}(L_{sym})}{\sqrt{d(X_i)}} - \frac{v_{kj}(L_{sym})}{\sqrt{d(X_j)}} \right)^2 .
\end{aligned}$$

[4]Note that the Dirac behavior becomes stronger as the kernel width grows and the graph approaches a clique with uniform weights. For RatioCut in such a clique, however, any cut is equal ($\mathrm{RatioCut}(f) = 2n$), so the clustering achieved with L is still okay from the perspective of the Rcut objective. This is, however, not the case for all objectives. Therefore the example shows that in practice, the objective and distance should "match". We will not go into further detail here, though.

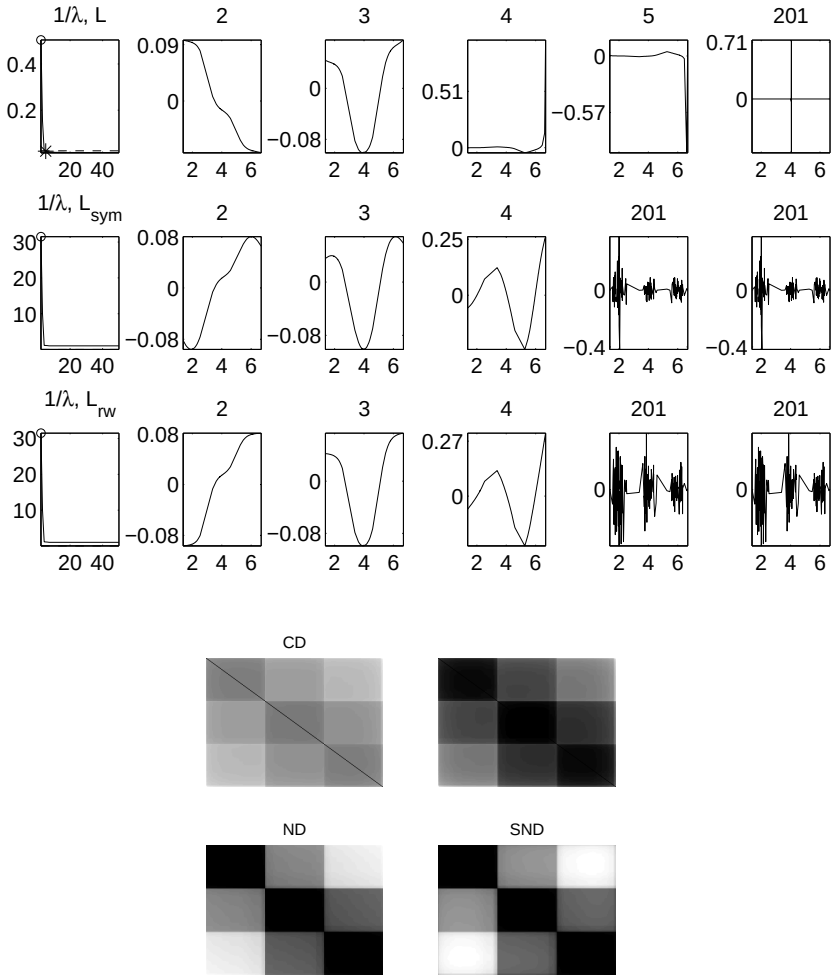


Figure 3.3.1: Laplacian eigenvalues and -vectors for a toy example: 201 points in 1D drawn from 3 Gaussians with means 2, 4, 6 and $\sigma = 0.25$. The similarity graph is based on a Gaussian kernel of width 1. The first panel shows the inverse eigenvalues (first column) and some eigenvectors (other columns, number indicated in title) for the unnormalized Laplacian L (row 1), L_{rw} (row 2) and L_{sym} (row 3). The asterisk indicates when $\lambda_i > \max_j d(X_j)$, it only happens for L (here, for $i = 5$). The eigenvectors of L approach Dirac functions early, when their weight is still high relative to that of the more useful eigenvectors. The spectra of the normalized Laplacians decay faster, and only towards the end of the spectrum the vectors become noisy. The second panel shows the distance matrices (darker is smaller). With the CD, points are very dissimilar, particularly the edge points to others. With ND and SND, this is not the case. The untitled plot is the analogue of ND and SND for L. (Note that the color scale is not the same in all plots, they are scaled separately.)

The last line shows that the commute distance is basically modified by the sum of all differences, weighted by the inverse of the largest eigenvalue. Theoretically, these equations should all be the same [von Luxburg, 2006]. In practice, however, toy experiments with Matlab lead to slightly different solutions. This divergence could be due to numerical problems.

Despite the motivation of better balancing, in the experiments (see Section 3.5.5), ND and SND did not lead to less variance in cluster sizes on real data.

3.4 Implementation

The basic description of nearest neighbor clustering in Section 3.1 raises the question of implementational details that improve computational efficiency, together with their impact on the solution.

Here, we will outline two aspects. First, an optimization of partitions of a contracted graph is computationally more efficient than an equivalent optimization on the original graph. Second, the brute-force search through $\mathcal{F}_n$ can be improved by a branch and bound approach that we describe for $K=2$ clusters and Ncut. For background reading on branch and bound methods see Brusco and Stahl [2005]. Both branch and bound and the contraction guarantee to find the optimal solution and thus do not influence the result, contrary to the choice of the distance function.

We will conclude this section with some heuristics to improve the average runtime. Note that these heuristics still guarantee an optimal solution.

3.4.1 Optimization over super-points

In NNC, the candidate functions are constant on the neighborhoods. So the neighborhood structure opens ways for a compression of possible solutions. Let us detail and prove one such approach.

The general representation of a partition f of n data points requires $O(n)$ space, giving the label for each point. Similarly, to evaluate $Q_n(f)$ on the graph, we need to look at n points and all $O(n^2)$ edges or similarities. This constitutes a substantial effort in the optimization, because we must compute each candidate partition and evaluate Q_n for it. The class $\mathcal{F}_n$ of nearest neighbor partitions, however, allows for a more efficient representation of the partitions and evaluation of Q_n, by considering a contracted graph of super-points.

Let us state this idea more formally, before we prove that the optimization on the contracted graph is equal to that on the original graph. For an original data set $V = \{X_1, \ldots, X_n\}$ with a similarity function $s : V \mapsto \mathbb{R}^+$, define m super-nodes Z_i representing the neighborhood cells. Super-node Z_i contains all points assigned to the i-th seed. These super-points may be interpreted as nodes in a contracted graph, endowed with a super-similarity function $\bar{s}(Z_s, Z_t) := \sum_{X_i \in Z_s, X_j \in Z_t} s(X_i, X_j)$. If the data is given as a graph, the edge weights define similarities, and an analogous summation over neighborhoods yields edge weights $\bar{w}$ in the contracted graph from the edge weights w. Note that the contracted graph has self loops, that is $\bar{w}(Z_t, Z_t) > 0$, if there are edges within neighborhoods. Each partition $f \in \mathcal{F}_n$ can then be represented as an extension of a function $\bar{f} : \{Z_1, \ldots, Z_m\} \mapsto \mathbb{R}^+$ on the contracted set, with $f(X_i) = \bar{f}(Z_s)$ if X_i belongs to the s-th cell. So the extension to V consists of labeling each X_i by the label $\bar{f}(Z_s)$ of the super-point Z_s corresponding to its neighborhood. Similarly to the similarities, Q_n can be defined on the partitions $\bar{f}$ of the

contraction, such as

$$\overline{\mathrm{Ncut}}(\bar{f}) = \sum_{\ell=1}^{K} \frac{\bar{w}(\mathcal{C}_\ell, V\backslash\mathcal{C}_\ell)}{\overline{\mathrm{vol}}(\mathcal{C}_\ell)}, \tag{3.4.1}$$

$$\text{with} \qquad \overline{\mathrm{vol}}(\mathcal{C}_\ell) = \sum_{Z_i,\in\mathcal{C}_\ell, 1\leq j\leq m} \bar{w}(Z_i, Z_j).$$

To compute $\overline{\mathrm{Ncut}}$, we can replace the $n \times n$ weight or similarity matrix of the original graph by an $m \times m$ matrix of $\bar{w}$ for the super-points. Apart from memory savings, the cost of the summation then drops from $O(n)$ to $O(m)$ for each partition.

After a reformulation, the WSS objective can as well be computed in terms of super-points and a precomputable constant (first term).

$$\overline{\mathrm{WSS}}(\bar{f}) = \frac{1}{n} \sum_{X_i \in V} X_i^\top X_i - \frac{1}{n} \sum_{\ell=1}^{K} \frac{1}{\sum_{Z_u \in \mathcal{C}_\ell} |Z_u|} \sum_{Z_s, Z_t \in \mathcal{C}_\ell} \widetilde{Z}_s^\top \widetilde{Z}_t, \tag{3.4.2}$$

$$\text{with} \qquad \widetilde{Z}_s = \sum_{X_i \in Z_s} X_i,$$

$$|Z_s| = \sum_{X_i \in Z_s} 1$$

The following proposition justifies the replacement of $\mathrm{Ncut}(f)$ and $\mathrm{WSS}(f)$ by $\overline{\mathrm{Ncut}}(\bar{f})$ and $\overline{\mathrm{WSS}}(\bar{f})$ in the optimization.

Proposition 5 (Equivalence of optimization on nodes and super-points)**.** *Let the super-nodes Z_i and the weights $\bar{w}$ of the contracted graph be defined as above. Assume that $f : V \mapsto \mathbb{R}^+$ is a function on the original vertices which is constant within each neighborhood Z_i. Denote by $\bar{f} : \{Z_1, \ldots, Z_m\} \mapsto \mathbb{R}^+$ the corresponding partition of the super-nodes. Furthermore, let $\mathrm{Ncut}(f)$ and $\mathrm{WSS}(f)$ be the quality function on the original graph, and $\overline{\mathrm{Ncut}}(\bar{f})$ and $\overline{\mathrm{WSS}}(\bar{f})$ their correspondents on the contraction, as defined in Equations (3.4.1) and (3.4.2).*

Then $\mathrm{Ncut}(f) = \overline{\mathrm{Ncut}}(\bar{f})$ and $\mathrm{WSS}(f) = \overline{\mathrm{WSS}}(\bar{f})$.

Proof. We first prove equivalence for Ncut, then for WSS.

It is easy to see that the sums of weights between neighborhoods suffice to exactly compute the Ncut on the full graph for clusters $\mathcal{C}_1$ to $\mathcal{C}_K$. The original criterion is

$$\mathrm{Ncut}(f) = \sum_{\ell=1}^{k} \frac{w(\mathcal{C}_\ell, V\backslash\mathcal{C}_\ell)}{\mathrm{vol}(\mathcal{C}_\ell)}.$$

The numerators are

$$\begin{aligned} w(\mathcal{C}_\ell, V\backslash\mathcal{C}_\ell) &= \sum_{X_i \in \mathcal{C}_\ell} \sum_{X_j \notin \mathcal{C}_\ell} w(X_i, X_j) \\ &= \sum_{Z_s \subseteq \mathcal{C}_\ell} \sum_{X_i \in Z_s} \sum_{Z_t \nsubseteq \mathcal{C}_\ell} \sum_{X_j \in Z_t} w(X_i, X_j) \\ &= \sum_{Z_s \subseteq \mathcal{C}_\ell} \sum_{Z_t \nsubseteq \mathcal{C}_\ell} \bar{w}(Z_s, Z_t) \\ &= \sum_{Z_s \subseteq \mathcal{C}_\ell} \bar{w}(Z_s, V\backslash\mathcal{C}_\ell) = \bar{w}(\mathcal{C}_\ell, V\backslash\mathcal{C}_\ell). \end{aligned} \tag{3.4.3}$$

Analogously, the summation for the denominator can be rewritten in terms of the diagonal elements of the $m \times m$ matrix of $\bar{w}$:

$$\begin{aligned}\mathrm{vol}(\mathcal{C}_\ell) &= \sum_{X_i \in \mathcal{C}_\ell, X_j \in V} w(X_i, X_j) \\ &= \sum_{Z_s \in \mathcal{C}_\ell, 1 \le t \le m} \; \sum_{X_i \in Z_s, X_j \in Z_t} w(X_i, X_j) \\ &= \sum_{Z_s \in \mathcal{C}_\ell, 1 \le t \le m} \bar{w}(Z_s, Z_t). \end{aligned} \tag{3.4.4}$$

Reformulations (3.4.3) and (3.4.4) imply that $\frac{w(\mathcal{C}_\ell, V \setminus \mathcal{C}_\ell)}{\mathrm{vol}(\mathcal{C}_\ell)} = \frac{\bar{w}(\mathcal{C}_\ell, V \setminus \mathcal{C}_\ell)}{\overline{\mathrm{vol}}(\mathcal{C}_\ell)}$ for all ℓ and thus $\mathrm{Ncut}(f) = \overline{\mathrm{Ncut}}(\bar{f})$.

In a similar manner, we can restate WSS for the contracted points. Let $c_\ell = \left(\sum_{X_i \in \mathcal{C}_\ell} X_i\right) / |\mathcal{C}_\ell|$ be the mean of cluster $\mathcal{C}_\ell$, and $n = |V|$. Then

$$\begin{aligned}\mathrm{WSS}(f) &= \frac{1}{n} \sum_{\ell=1}^{K} \sum_{X_i \in \mathcal{C}_\ell} \|X_i - c_\ell\|^2 \\ &= \frac{1}{n} \sum_{\ell=1}^{K} \sum_{X_i \in \mathcal{C}_\ell} \left(\frac{1}{|\mathcal{C}_\ell|} \sum_{X_j \in \mathcal{C}_\ell} (X_i - X_j) \right)^\top \left(\frac{1}{|\mathcal{C}_\ell|} \sum_{X_k \in \mathcal{C}_\ell} (X_i - X_k) \right) \\ &= \frac{1}{n} \left(\sum_{X_i \in V} X_i^\top X_i - \sum_{\ell=1}^{K} \frac{1}{|\mathcal{C}_\ell|} \sum_{X_i, X_j \in \mathcal{C}_\ell} X_i^\top X_j \right) \\ &= \frac{1}{n} \sum_{X_i \in V} X_i^\top X_i - \frac{1}{n} \sum_{\ell=1}^{K} \frac{1}{|\mathcal{C}_\ell|} \sum_{Z_s, Z_t \in \mathcal{C}_\ell} \sum_{X_i \in Z_s} \sum_{X_j \in Z_t} X_i^\top X_j \\ &= \frac{1}{n} \sum_{X_i \in V} X_i^\top X_i - \frac{1}{n} \sum_{\ell=1}^{K} \frac{1}{|\mathcal{C}_\ell|} \sum_{Z_s, Z_t \in \mathcal{C}_\ell} \left(\sum_{X_i \in Z_s} X_i \right)^\top \left(\sum_{X_j \in Z_t} X_j \right) \\ &= \frac{1}{n} \sum_{X_i \in V} X_i^\top X_i - \frac{1}{n} \sum_{\ell=1}^{K} \frac{1}{\sum_{Z_u \in \mathcal{C}_\ell} |Z_u|} \sum_{Z_s, Z_t \in \mathcal{C}_\ell} \widetilde{Z}_s^\top \widetilde{Z}_t \\ &= \overline{\mathrm{WSS}}(\bar{f}). \end{aligned}$$

Thus, it suffices to precompute the sum of the squared norms of the points as well as the dot products of the sums $\widetilde{Z}_s = \sum_{X_i \in Z_s} X_i$. □

3.4.2 A branch and bound algorithm

Not only the computation of Q_n, but also the search through $\mathcal{F}_n$ can be sped up on average. Even though, for $m = O(\log n)$, all possible $O(2^{\log n})$ partitions can be enumerated in polynomial time, more efficient strategies exist. For better average-case runtime, we revert to a branch and bound approach, which is still guaranteed to return the optimal solution. We will prove this claim for our algorithm after a description of it. In the following, we operate on the contracted graph and assume to search for $K = 2$ clusters.

The general idea of branch and bound is as follows. We represent the solution space as a tree and assume that we know an upper bound θ_u on the objective function value of the optimal solution (for example, θ_u could be the value of a particular initial candidate clustering $\bar{f}_0$). Then we descend in the tree, and at each vertex decide whether the current branch of the tree might contain a better solution than the one given by the upper bound θ_u. To this end, we need to compute a lower bound θ_l on the objective function values of all solutions represented in the current branch of the tree. If $\theta_l > \theta_u$ we know that the current branch only contains solutions which are worse than the one we already have, and we can prune this branch, saving the time to inspect it any further.

Let us look at the example of solving NNC(Ncut) for two clusters $\mathcal{C}_+$ and $\mathcal{C}_-$, with respective labels $+1$ and -1. Figure 3.4.2 outlines the recursive algorithm in pseudocode. Initially, Z_1, in a cluster by itself, is labeled by $+1$, and all other nodes by -1. In each step, the two branches consist of fixing the label of the next super-point to be positive or remain negative. For K clusters, we will then have K branches to choose from. Before we prove correctness of our algorithm, we go into more detail about the branching strategies.

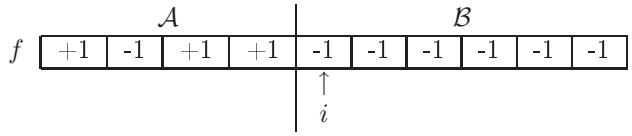


Figure 3.4.1: Illustration of the sets $\mathcal{A}$, $\mathcal{B}$ and f in recursion i, in which the assignment of Z_i is determined. The fixed subsets are $\mathcal{A}^- = \{Z_2\}$ and $\mathcal{A}^+ = \{Z_1, Z_3, Z_4\}$.

Assume we have already determined the labels $l_1, \ldots, l_{i-1} \in \{\pm 1\}$ of the first $i-1$ super-points. For those points we introduce the set $\mathcal{A} = \{Z_1, \ldots, Z_{i-1}\}$ with subsets $\mathcal{A}^- := \{Z_j \mid j < i, l_j = -1\}$ and $\mathcal{A}^+ := \{Z_j \mid j < i, l_j = +1\}$. The remaining points form set $\mathcal{B} = \{Z_i, \ldots, Z_m\}$ so that $V = \mathcal{A} \cup \mathcal{B}$. All points in $\mathcal{B}$ get the label -1 by default. In recursion level i, we decide about moving Z_i to cluster $\mathcal{C}_+$. Figure 3.4.1 schematically illustrates the sets.

The branch and bound strategy instructs to investigate whether the movement of one or more points from $\mathcal{B}$ to the $+1$ cluster has the potential to improve the Ncut. More formally, the question is whether the "branch" of clusterings that agree on the current fixed labels $l_1, \ldots, l_{i-1}$ on $\mathcal{A}$ could contain a solution which is better than any previously considered partition.

We try to answer this question by exclusions. If we cannot exclude a better solution in the current branch, we explore it further. We determine exclusions in two steps (Part 3. in Figure 3.4.2), a direct and an indirect one. First, we determine if an improvement of the current Ncut is possible at all by relabeling any node in $\mathcal{B}$, and keeping labels 1 to $i-1$ fixed (Step I). If our conditions are not satisfied, the current branch cannot lead to any improvement. Their satisfaction, however, does not imply that the branch does indeed contain a better solution. Thus, in Step II, we compute a lower bound θ_l on the solutions in the branch and compare it to the current upper bound θ_u.

Both steps consider the two factors of the quality function separately, namely the "cut term" and the "volume term" in the product $\mathrm{Ncut}(f) = \mathrm{cut}(\mathcal{C}_+, \mathcal{C}_-) \cdot (1/\mathrm{vol}(\mathcal{C}_+) + 1/\mathrm{vol}(\mathcal{C}_-))$. Step I consists of one condition for the cut and one for the volume term, and in Step II we bound the cut and volume terms separately. Hence, we structure either step into a cut (I.c, II.c) and a volume (I.v, II.v) part.

Branch and bound algorithm for Ncut: $f^* = \texttt{bbncut}(\bar{S}, i, f, \theta_u)\{$

1. Set $g := f$; set $\mathcal{A}^-$, $\mathcal{A}^+$, and $\mathcal{B}$ as described in the text
2. // Deal with special cases:
   - If $i = m$ and $\mathcal{A}^- = \emptyset$ then return f.
   - If $i = m$ and $\mathcal{A}^- \neq \emptyset$:
     - Set $g_i = +1$.
     - If $\mathrm{Ncut}(g) < \mathrm{Ncut}(f)$ return g, else return f.
3. // Pruning:
   - If $\max_{j \geq i}\{\bar{s}(j, \mathcal{A}^+) - \bar{s}(j, \mathcal{A}^-)\} \leq 0$ (I.c), and $\mathrm{vol}(\mathcal{A}^+) > \mathrm{vol}(\mathcal{A} \cup \mathcal{B})/2$ (I.v), then return f.
   - Compute lower bound θ_l as described in the text.
   - If $\theta_l \geq \theta_u$ then return f.
4. // If no pruning possible, recursively call bbncut:
   - Set $g_i = +1$, $\theta'_u := \min\{\mathrm{Ncut}(g), \theta_u\}$, call $g' := \texttt{bbncut}(\bar{S}, g, i+1, \theta'_u)$
   - Set $g_i = -1$, $\theta''_u := \min\{\mathrm{Ncut}(g), \theta'_u\}$, call $g'' := \texttt{bbncut}(\bar{S}, g, i+1, \theta''_u)$
   - If $\mathrm{Ncut}(g') \leq \mathrm{Ncut}(g'')$ then return g', else return g''. }

Figure 3.4.2: *Branch and bound algorithm for solving* NNC(Ncut) *for* $K = 2$. *The algorithm is initially called with the super-similarity matrix* $\bar{S}$, $i = 2$, $f = (+1, -1, -1, ..., -1)$, *and* θ_u *the Ncut value of* f.

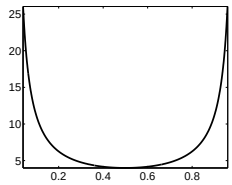


Figure 3.4.3: The volume term $1/x + 1/(1-x)$ for $x \in (0, 1)$.

Let us take a closer look at the conditions and bounds, starting with I.c and I.v. The criteria of Step I are very simple: assigning at least one point in $\mathcal{B}$ to C_+ can only lead to an improvement if this either decreases the cut term or the volume term of Ncut (or both).

I.c Necessary for an improvement of the cut term is that at least one point in $\mathcal{B}$ is more attached to the nodes in $\mathcal{A}^+$ than to the nodes in $\mathcal{A}^- \cup \mathcal{B}$ labeled by -1. More formally, it must be that $\max_{j \geq i}\{\bar{s}(Z_j, \mathcal{A}^+) - \bar{s}(Z_j, \mathcal{A}^-)\} \geq 0$

I.v The volume term is minimized if both clusters are equal in volume, $\mathrm{vol}(\mathcal{C}_+) = \mathrm{vol}(\mathcal{C}_-) = \mathrm{vol}(V)/2$, and increases as the sizes become more unbalanced (Figure 3.4.3). Thus, if $\mathcal{C}_+$ already takes up more than half the volume, $\mathrm{vol}(\mathcal{C}_+) = \mathrm{vol}(\mathcal{A}^+) > \mathrm{vol}(V)/2$, an additional node will further impair the volume balance and increase the term. Therefore, the volume criterion is $\mathrm{vol}(\mathcal{A}^+) \leq \mathrm{vol}(V)/2$.

If neither condition is satisfied, we retract. Otherwise, we proceed to Step II.

In Step II, we compute a lower bound θ_l and compare it to an upper bound θ_u on the optimal Ncut value, namely to the Ncut value of the best function we have seen so far. If $\theta_l \geq \theta_u$, then no improvement is possible by any clustering in the current branch of the tree, and we retract. To compute θ_l, assume we assign a non-empty set $\mathcal{B}^+ \subset \mathcal{B}$ to cluster

$\mathcal{C}_+$, and keep the labels -1 in remaining set $\mathcal{B}^- = \mathcal{B} \setminus \mathcal{B}^+$. Let $\alpha(\mathcal{C}_+, \mathcal{C}_-) = \mathrm{cut}(\mathcal{C}_+, \mathcal{C}_-)$ and $\beta(\mathcal{C}_+, \mathcal{C}_-) = 1/\operatorname{vol}(\mathcal{C}_+) + 1/\operatorname{vol}(\mathcal{C}_-)$ denote the cut and volume terms, respectively. We bound them separately by α' and β' to set $\theta_l = \alpha'\beta'$.

II.c The cut term consists of a fixed part, the cut edges between $\mathcal{A}^+$ and $\mathcal{A}^-$, and a variable part, the cut edges adjacent to nodes in $\mathcal{B}$. To bound α from below, consider two subcases. If $\mathcal{A}^- = \emptyset$, at least one node from $\mathcal{B}$ must remain in $\mathcal{C}_-$, and the fixed part is zero. Hence, the cut is at least as big as the minimum attachment of any node in $\mathcal{B}$ to $\mathcal{A}^+$. Otherwise, any nonempty subset of $\mathcal{B}$ may be moved, and the cut consists at least of the fixed part plus the minimum weight between a node in $\mathcal{B}$ and $\mathcal{A}^-$. More formally, using the convention $\bar{s}(\mathcal{A}, \emptyset) = 0$, set

$$\alpha' = \begin{cases} \bar{s}(\mathcal{A}^+, \mathcal{A}^-) + \min_{j \geq i} \bar{s}(Z_j, \mathcal{A}^-) & \text{if } \mathcal{A}^- \neq \emptyset \\ \min_{j \geq i} \bar{s}(Z_j, \mathcal{A}^+) & \text{otherwise.} \end{cases}$$

II.v As stated above, $\beta(\mathcal{C}_+, \mathcal{C}_-)$ is minimal if $\operatorname{vol}(\mathcal{C}_+) = \operatorname{vol}(\mathcal{C}_-) = \operatorname{vol}(V)/2$. The volume term can only decrease to this value of $4/\operatorname{vol}(V)$ if $\operatorname{vol}(\mathcal{A}^+) \leq \operatorname{vol}(V)/2$, because the addition $\mathcal{B}^+$ to $\mathcal{C}_+ = \mathcal{A}^+ \cup \mathcal{B}^+$ is nonempty. If $\mathcal{A}^+$ already covers half the entire volume, then an increase is unavoidable. This rise is minimal if the node in $\mathcal{B}$ with the smallest degree is moved to $\mathcal{C}_+$, because it leads to the least further deterioration of the volume balance. In formal terms, set

$$\beta' = \begin{cases} 4/\operatorname{vol}(V) & \text{if } \operatorname{vol}(\mathcal{A}^+) \leq \operatorname{vol}(V)/2 \\ \min_{j \geq i} \{1/\operatorname{vol}(\mathcal{A}^+ \cup Z_j) + 1/\operatorname{vol}(\mathcal{A}^- \cup \mathcal{B} \setminus Z_j)\} & \text{otherwise.} \end{cases}$$

If $\theta_l \geq \theta_u$, we retract, otherwise we recursively investigate the sub-branches of setting $l_i = +1$ and $l_i = -1$, and keep θ_u updated. This recursion is Step 4. in Figure 3.4.2. Some heuristics can improve the average runtime, as outlined in Subsection 3.4.3.

The complete algorithm returns a global minimizer of Ncut, as the next section shows.

Correctness

Lemma 6 (Correctness of `bbncut`)**.** *Let the labels of $\mathcal{A}(i) = \mathcal{A}^+(i) \cup \mathcal{A}^-(i) = \{Z_1, \ldots, Z_{i-1}\}$ be fixed, and assume the input θ_u is a strict upper bound on the* Ncut *values of the partitions in conformity with the labels on $\mathcal{A}$. Then the algorithm* `bbncut` *returns an assignment of the vertices $\mathcal{B}(i) = \{Z_i, Z_{i+1}, \ldots, Z_m\}$ that optimizes the* Ncut *criterion, with fixed labels on $\mathcal{A}(i)$.*

Proof. We prove Lemma 6 by induction on the number $n_{\mathcal{B}} = |\mathcal{B}(i)|$ of nodes to assign. A final assignment is represented by $\mathcal{C}_+ = \mathcal{A}^+ \cup \mathcal{B}^+$ and $\mathcal{C}_- = \mathcal{A}^- \cup \mathcal{B}^-$. For notational issues, let `bbncut`$(\mathcal{A}^-(i), \mathcal{A}^+(i), \mathcal{B}(i))$ denote the solution returned by `bbncut` for inputs $\mathcal{A}^+(i)$, $\mathcal{A}^-(i)$ and $\mathcal{B}(i)$, where vertices starting from Z_i were assigned to a cluster.

Base Case. Assume $i = m$, so $\mathcal{B}(m) = \{Z_m\}$ and $n_{\mathcal{B}} = 1$. If $\mathcal{A}^-(i) = \emptyset$, then the best assignment of Z_m is to cluster $\mathcal{C}_-$, that is $\mathcal{B}^-(i) = \{Z_m\}$. Otherwise $\mathcal{C}_- = \mathcal{A}^-(i) \cup \mathcal{B}^-(i) = \emptyset$ will make the Ncut value infinity. Hence the algorithm correctly sets $\mathcal{B}^-(i) = \{Z_m\}$.

If $\mathcal{A}^-(i) \neq \emptyset$, then the best assignment is the one of $B^-(i) = \{Z_m\}$ and $\mathcal{B}^+(i) = \{Z_m\}$ minimizing the Ncut. The former has already been computed and corresponds to the current cut value. So the algorithm returns the correct assignment and is thus correct for $n_{\mathcal{B}} = 1$.

Inductive Step. Now, assuming that `bbncut` is correct for $n_{\mathcal{B}}$, let us see that `bbncut` is correct for $n_{\mathcal{B}}+1$ as well (provided $m > n_{\mathcal{B}}$, otherwise we are finished anyway). Let $i = m - n_{\mathcal{B}}$, so we know that `bbncut` gives a correct solution for all i' with $m \geq i' > i$.

We will consider the possibilities of failure one by one. First, we show that if we reach the recursion (Step 4), then the correct solution is returned. Then we look at the pruning criteria in Step 3 and show that they are only met if no improvement is possible in the current branch. Hence we always reach the recursion if the current branch contains a better partition. As to pruning, we first investigate the direct criteria I.c and I.v, and then the components α' (II.c) and β' (II.v) of the lower bound.

Assume that no pruning criterion is met and we are in Step 4. The assignment of nodes up to Z_{i-1} is fixed. The best solution is then the best of those assignments in conformity with either $(\mathcal{A}^-(i) \cup \{Z_i\}, \mathcal{A}^+(i))$ or $(\mathcal{A}^-(i), \mathcal{A}^+(i) \cup \{Z_i\})$. Thus, take the better of the best solution for each such sub-assignment, that means of $\texttt{bbncut}(\mathcal{A}^-(i) \cup \{Z_i\}, \mathcal{A}^+(i), \mathcal{B}(i+1))$ and $\texttt{bbncut}(\mathcal{A}^-(i), \mathcal{A}^+(i) \cup \{Z_i\}, \mathcal{B}(i+1))$. The recursive calls return the correct solutions by the correctness for $i' > i$. Hence, the correct solution is returned if the branching criterion is fulfilled in Step 4.

If the current assignment $\mathcal{C}_- = \mathcal{A}^-(i) \cup \mathcal{B}(i)$, $\mathcal{C}_+ = \mathcal{A}^+(i)$ is the optimal solution (i.e. $\mathcal{B}^+(i) = \emptyset$), it will either be returned in the recursion ($B^-(i) = \mathcal{B}(i)$, by the correctness for $n_{\mathcal{B}}$), or by skipping the recursion.

So only one possibility of failure remains: Assume the best solution f_n is $\mathcal{C}_- = \mathcal{A}^-(i) \cup B^-_*$, $\mathcal{C}_+ = \mathcal{A}^+(i) \cup B^+_*$ with $B^+_* \neq \emptyset$, but the recursion is skipped because some pruning criterion is met in Step 3. That means either both I.c and I.v are not satisfied, or $\theta_l \geq \theta_u$, so α' or β' is no lower bound. We demonstrate that this assumption leads to a contradiction for both I and II. Let θ_u be the Ncut value of the best solution encountered so far, so $\mathrm{Ncut}(f_n) = \mathrm{Ncut}(\mathcal{A}^-(i) \cup \mathcal{B}^-_*, \mathcal{A}^+(i) \cup \mathcal{B}^+_*) < \theta_u$. Denote by α^* and β^* the cut and volume terms for f_n, respectively. The current Ncut is $\mathrm{Ncut}(\mathcal{A}^+(i), \mathcal{A}^-(i) \cup \mathcal{B}(i)) > \mathrm{Ncut}(f_n)$. In the following treatments, we always consider the sets for i, so we leave out this index for notational simplicity.

Direct criteria (I) Assume that both I.c and I.v are not fulfilled.

I.c Condition I.c implies that $\bar{s}(\mathcal{A}^-, Z_j) > \bar{s}(\mathcal{A}^+, Z_j)$ for all $Z_j \in \mathcal{B}$, so

$$\begin{aligned}\alpha^* &= \bar{s}(\mathcal{A}^+, \mathcal{A}^-) + \bar{s}(\mathcal{A}^+, \mathcal{B}^-_*) + \bar{s}(\mathcal{A}^-, \mathcal{B}^+_*) + \bar{s}(\mathcal{B}^-_*, \mathcal{B}^+_*) \\ &> \bar{s}(\mathcal{A}^+, \mathcal{A}^-) + \bar{s}(\mathcal{A}^+, \mathcal{B}^-_*) + \bar{s}(\mathcal{A}^+, \mathcal{B}^+_*) \\ &= \alpha(\mathcal{A}^+, \mathcal{A}^- \cup \mathcal{B}). \end{aligned} \tag{3.4.5}$$

The last term is the current cut term.

I.v From I.v, we know that $\mathrm{vol}(\mathcal{A}^+) > \mathrm{vol}(V)/2$, so the addition of any node to $\mathcal{A}^+$ can only increase the current volume term. Hence, β^* is greater than the current β:

$$\beta^* > \beta(\mathcal{A}^+, \mathcal{A}^- \cup \mathcal{B}). \tag{3.4.6}$$

Equations (3.4.5) and (3.4.6) imply that

$$\mathrm{Ncut}(f_n) = \alpha^*\beta^* > \alpha(\mathcal{A}^+, \mathcal{A}^- \cup \mathcal{B}) \cdot \beta(\mathcal{A}^+, \mathcal{A}^- \cup \mathcal{B}),$$

a contradiction to f_n's optimality.

Lower bound (II) The second possibility is that $\theta_l \geq \theta_u > \mathrm{Ncut}(f_n)$, so the lower bound is incorrect. By looking first at α' and then at β', we show that this contradicts the optimality of f_n.

II.c For the cut term, we distinguish two subcases, (i) $\mathcal{C}_-$ does not have a fixed member yet ($\mathcal{A}^- = \emptyset$), or (ii) some points in $\mathcal{A}$ are labeled -1. If $\mathcal{A}^- = \emptyset$, at least one node in $\mathcal{B}$ must remain in $\mathcal{C}_-$. So $\bar{s}(\mathcal{A}^+, B_*^-) \geq \min_{j \geq i} \bar{s}(\mathcal{A}^+, Z_j)$ and it is

$$\begin{aligned}\alpha' &= \bar{s}(\mathcal{A}^+, \mathcal{A}^-) + \min_{Z_j \in B} \bar{s}(\mathcal{A}^+, Z_j) \\ &\leq \bar{s}(\mathcal{A}^+, \mathcal{A}^-) + \bar{s}(\mathcal{A}^+, \mathcal{B}_*^-) + \bar{s}(\mathcal{B}_*^+, \mathcal{A}^-) + \bar{s}(\mathcal{B}_*^+, \mathcal{B}_*^-) = \alpha^*.\end{aligned}$$

If $\mathcal{A}^- \neq \emptyset$, then any nonempty subset of $\mathcal{B}$ may be moved to $\mathcal{C}_-$, and hence the cut value includes at least the edges from this subset to the fixed $\mathcal{A}^+$, that is $\bar{s}(B_*^+, \mathcal{A}^-) \geq \min_{j \geq i} \bar{s}(Z_j, \mathcal{A}^-)$. This implies

$$\begin{aligned}\alpha' &= \bar{s}(\mathcal{A}^+, \mathcal{A}^-) + \min_{Z_j \in B} \bar{s}(Z_j, \mathcal{A}^-) \\ &\leq \bar{s}(\mathcal{A}^+, \mathcal{A}^-) + \bar{s}(\mathcal{A}^+, \mathcal{B}_*^-) + \bar{s}(\mathcal{B}_*^+, \mathcal{A}^-) + \bar{s}(\mathcal{B}_*^+, \mathcal{B}_*^-) = \alpha^*.\end{aligned}$$

It follows that in any case, we have $\alpha' \leq \alpha^*$, so α' is indeed a lower bound.

II.v So only the volume term remains. If $\mathrm{vol}(\mathcal{A}) \leq \mathrm{vol}(V)/2$, then β' is set to the minimum possible $4/\mathrm{vol}(V)$, and thus $\beta' \leq \beta^*$. Otherwise

$$\begin{aligned}\beta' &= \min_{j \geq i} \left\{ (\mathrm{vol}(\mathcal{A}^+) + d(Z_j))^{-1} + (\mathrm{vol}(V) - \mathrm{vol}(\mathcal{A}^+) - d(Z_j))^{-1} \right\} \\ &\leq (\mathrm{vol}(\mathcal{A}^+) + \mathrm{vol}(B_*^+))^{-1} + (\mathrm{vol}(V) - (\mathrm{vol}(\mathcal{A}^+) + \mathrm{vol}(B_*^+)))^{-1} = \beta^*\end{aligned}$$

because $h(x) = 1/x + 1/(1-x)$ is strictly monotonically increasing on $[0.5, 1]$ and $B_*^+ \neq \emptyset$. So in any case $\beta' \leq \beta^*$.

The assumption $\theta_u \leq \theta_l$ together with the conclusions $\alpha' \leq \alpha^*$ and $\beta' \leq \beta^*$ implies that

$$\theta_u \leq \theta_l = \alpha'\beta' \leq \alpha^*\beta^* = \mathrm{Ncut}(f_n),$$

a contradiction to f_n being better than the best solution encountered so far.

In summary, both the violation of both direct criteria and the incorrectness of θ_l lead to a contradiction. So, if f_n is optimal, we will reach the recursion in Step 4.

This means that if the optimal solution has sub-assignments $(\mathcal{A}^-(i), \mathcal{A}^+(i))$ and $B_*^+ \neq \emptyset$, the recursion is taken and the correct assignment returned. This argument completes the proof of correctness for $n_\mathcal{B} + 1$, implying correctness for all $n_\mathcal{B} < m$ by induction. □

Corollary 7. *Calling* `bbncut` *for* $i = 2$ *with* $\mathcal{A}^+ = \{Z_1\}$ *and* θ_u *being the* Ncut *value for the partition* $\mathcal{C}_+ = \{Z_1\}$, $\mathcal{C}_- = \mathcal{B}^{(2)}$ *solves* Ncut *for the entire graph,* $\{Z_1, \ldots Z_m\}$.

Proof. The corollary follows from Lemma 6 and the fact that fixing the label of Z_1 does not fix the partition in any way: There are two instantiations of the optimal partition by a mere swap of labels. □

3.4.3 Heuristics

The branch and bound strategy from above may be further improved by a number of heuristics. Note that both branch and bound as well as the following heuristics still guarantee a correct solution. In turn, they cannot influence the worst-case runtime. Nevertheless,

the average runtime is sped up, as we will demonstrate below. Let us first motivate some heuristics and show their influence.

Reflections about heuristics raise the question for factors determining the runtime. The worst case, that we cannot exclude completely, is the enumeration of all functions in $\mathcal{F}_n$. However, in average, "nicer" cases, branch and bound helps to exclude subsets of $\mathcal{F}_n$ via pruning. So the earlier the pruning happens, the larger the subset that is excluded and the smaller the set of candidate functions remaining. Thus, we would like to identify a "useless" branch as soon as possible. Pruning is determined by two criteria, the direct conditions I.c and I.v, and the bounds θ_l and θ_u, that we hence try to satisfy soon:

I.c The cut criterion is satisfied if the nodes in $\mathcal{B}$ are more attached to $\mathcal{A}^-$ than to $\mathcal{A}^+$. This is only possible if $\mathcal{A}^-$ has a sufficient size, or if all nodes in $\mathcal{B}$ that are closely attached to $\mathcal{A}^-$ are moved to $\mathcal{A}$ as soon as possible.

I.v The volume criterion requires for pruning that $\text{vol}(\mathcal{A}) \geq \text{vol}(V)/2$, hence $\mathcal{A}$ should grow in volume early.

θ_u As the bound criterion is $\theta_l \geq \theta_u$, a tight upper bound excludes more branches. Hence, try to achieve good quality values early.

II.c The cut bound α' involves both $\mathcal{A}^+$ and $\mathcal{A}^-$, so it will be more discriminative if both $\mathcal{A}^+$ and $\mathcal{A}^-$ have significant size or, better, volume (as volume is related to edges).

II.v The volume bound is only discriminative if $\text{vol}(\mathcal{A}) \geq \text{vol}(V)/2$, so $\mathcal{A}$ should grow in volume early.

The question is how to integrate as many of those demands as possible. Many require that $\mathcal{A}$ or its subsets should grow fast, in volume or size. The number of nodes in $\mathcal{A}$ is the level of the recursion, so it is not due to change. The volume, however, grows fastest if we fix the label of the nodes with highest degree first. Both $\mathcal{A}^+$ and $\mathcal{A}^-$ simultaneously grow early if the labels in $\mathcal{A}$ are approximately uniformly distributed over the volume. If the nodes are ordered non-increasingly by degree, then this condition is satisfied if the nodes in $\mathcal{A}$ are labeled roughly alternately. In consequence, the sets $\mathcal{A}^+$ and $\mathcal{A}^-$ are approximately balanced in volume in most subsets $\{Z_1, \dots Z_{i-1}\}$ for varying recursion levels i. As a side effect, this strategy may improve the lower bound θ_u with Ncut, as the volume balance is favored in the subset $\mathcal{A}$. If $\mathcal{B}$ is large in volume, this will of course not be the case. Diminishing the size of $\mathcal{A}^-$ for better balancing (by number), however, contradicts the requirement of I.c. But if high-degree nodes are labeled early, then $\mathcal{B}$ will be as small as possible in volume, thus influencing the balance as little as possible, while most other demands are met.

So how can we realize this balanced labeling by degree? The algorithm itself leaves room for changes in various places:

1. in the implementation: order of nodes, initial θ_u
2. in the order of the pruning criteria
3. in the order of the recursion in Step 4: branch first into $g_i = -1$ or $g_i = +1$; this determines the labels in $\mathcal{A}$ in the earliest recursions
4. reorder nodes in $\mathcal{B}$ at lower levels of the recursion (as it is quite complicated to keep track of the reorderings throughout recursions, we did not implement this possibility).

The quick growth of $\mathcal{A}$ may be influenced in the implementation already. As we label the nodes in ascending order by their number, we order them by degree, starting with the heaviest node. Thus, the higher the degree, the earlier a node will be moved to $\mathcal{A}$.

The order of the recursions influences the first labelings in $\mathcal{A}$. In consecutive recursion levels, we alternately branch into the -1 and $+1$ branch first, that means once set $g_i = +1$ first and in the next level $g_{i+1} = -1$ first. This means that in the first path we take we alternately assign nodes to cluster $\mathcal{C}_-$ and $\mathcal{C}_+$ for a rough balancing. The better the first partitions found, the tighter will be the the upper bound in early recursions. The alternation goes hand in hand with the reordering of nodes by degree. Hopefully, this leads to early tight upper bounds and discriminative pruning conditions.

The direct pruning conditions are simpler to compute than the lower bound, and they are definitive. Hence it makes sense to test the improvement conditions I.c and I.v before computing θ_l.

The initial upper bound should be as tight as possible, but cheap to compute. One such possibility is the minimum possible Ncut if one cluster consists of only one node. Without self-loops on any node ($w(X_i, X_i) = 0$ for all i), this value is

$$\min_i \frac{d(X_i)}{(\mathrm{vol}(G) - d(X_i))} + 1.$$

The fraction is minimized by the node X_i with the lowest degree, as Figure 3.4.4 illustrates. Thus, it only needs to be evaluated for the first node and the node with minimum degree. In view of this degree aspect, the heuristic start node with highest degree will definitely not be in a cluster by itself.

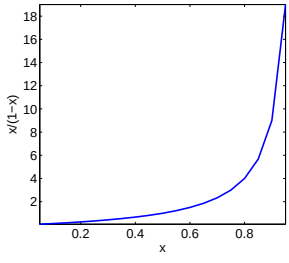


Figure 3.4.4: Function $x/(1-x)$ is minimized by small x.

If, by chance, θ_u already corresponds to the best cluster assignment f_n, and this is not the separation of X_1 we start with, then the algorithm might prune the best search path, because it only follows a path if an *improvement* to θ_u is possible. The algorithm will however still terminate soon, because a good θ_u from the beginning prunes many search paths immediately. We can then simply compare the Ncut value of the returned solution to θ_u. If it is larger, then we know that the assignment corresponding to the initial θ_u was the optimal one, that means one cluster only consists of a single node.

So how do the heuristics behave in practice, as the number n of nodes or the number k of neighbors in the k-nearest neighbor graph changes? Figure 3.4.5 illustrates the runtime without branch and bound ('all'), compared to branch and bound with the alternating heuristic (BB) and with both alternation and sorted nodes (BBsort). The branch and bound algorithm always tested condition I.c and I.v before considering θ_l, and the initial upper bound was always set to cutting off the node with the lowest degree. All the figures refer to the mere optimization without any reduction of the function class, so 2^n partitions are to consider.

The runtime (Figure 3.4.5) as well as the number of evaluations of Ncut and the number of recursions (Figure 3.4.6) is drastically reduced by branch and bound, and further by the sorting heuristic. The runtime increases with n but not much with k, as Figure 3.4.5 shows. The number k of neighbors in the graph affects the summation in evaluating Ncut. The increase in recursions in Figure 3.4.7 may be due to a later satisfaction of the pruning criteria with higher k. If each node has more connections to neighbors, then the movement of one node may be more influential to the cut quality. The addition of neighbors makes

more difference for smaller k, when the additional neighbors are close enough to make edges of significant weight.

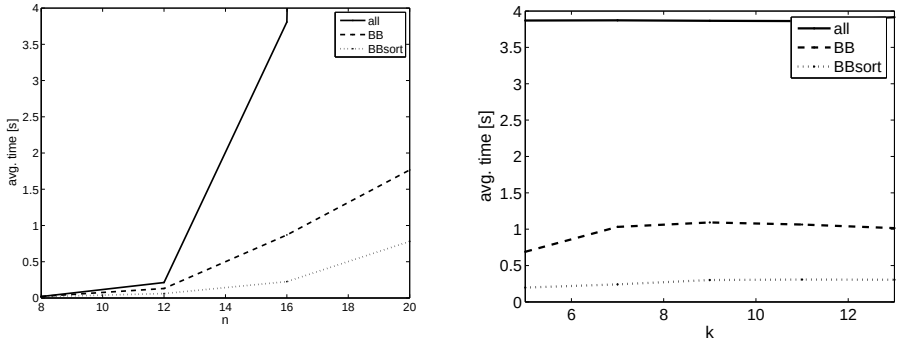


Figure 3.4.5: Runtime changes, for increasing n with $k = 5$ neighbors (left), or increasing k with $n = 16$ (right).

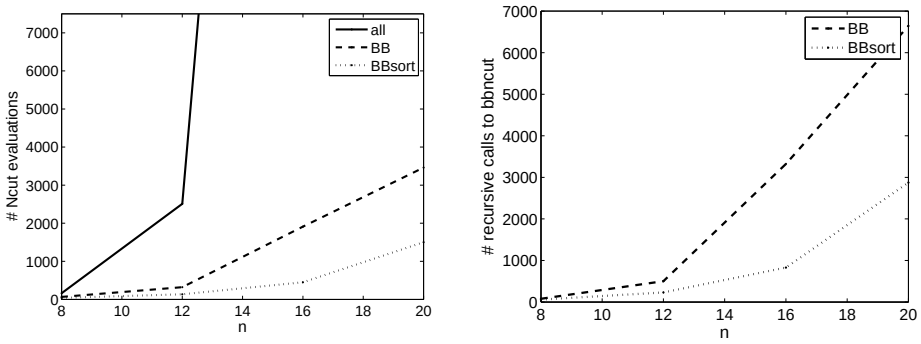


Figure 3.4.6: Number of calls to Ncut (left) and `bbncut` (right) for increasing n with $k = 5$.

In summary, the heuristics improve the average runtime, particularly with respect to the number of nodes.

3.5 Experiments

An important feature of nearest neighbor clustering is its statistical consistency: for large n, it reveals an approximately correct clustering. Its behavior on smaller samples is the subject of the first set of experiments. In Section 3.5.3 we will compare the results of NNC to results from heuristics designed to directly optimize the given objective function Q_n. Since generalization is a key aspect in Learning Theory, we do not only compare the value of Q_n of the solutions but also their generalizability in Section 3.5.4.

Even though the differences may be small in the limit for large samples, the choice of the distance can influence the performance of nearest neighbor clustering for smaller n. Hence in Section 3.5.5 we compare results for different distance functions.

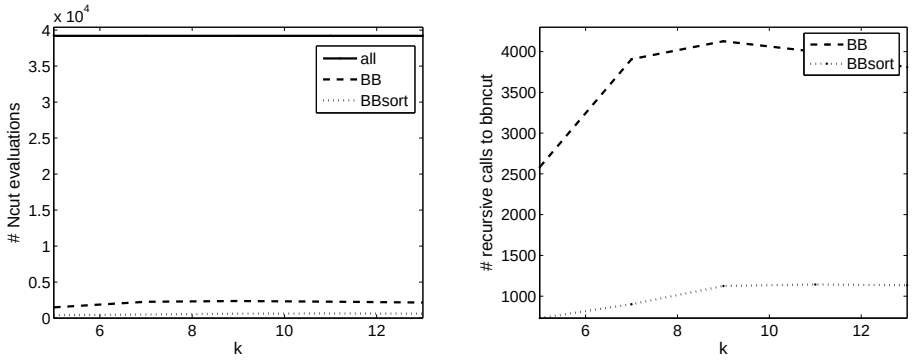


Figure 3.4.7: Number of calls to Ncut (left) and `bbncut` (right) for increasing k with $n = 16$.

A further impact on the results may be the number and selection of seed points. We take a closer look at seed-related issues in Section 3.5.6.

3.5.1 Construction of graphs

In the experiments, we use artificial and real data. The artificial data usually consists of points sampled from a mixture of Gaussians in $\mathbb{R}^d$.

The real data sets will be described in the following. If the data is not already given as a weighted or unweighted graph, a k-nearest neighbor graph is constructed by connecting each node to its k closest neighbors (cf. Section 1.2.2). The weight of edge (X_i, X_j) is given by the Gaussian kernel (see Eq. (1.2.2)). In general, we use undirected graphs, that means if edge (X_i, X_j) exists, then the graph also contains edge (X_j, X_i) with equal weight. If the original graph was directed, we added a reciprocal edge of equal weight for each edge in the graph.

3.5.2 Real data sets

We collected graphs from several sources about a variety of data. Here we describe the source of the data in the experiments. If the resulting graphs were not connected, we used the largest connected component. Table 3.1 shows the number of nodes and edges for each graph.

COSIN biological data

The COSIN website [cosin] provides weighted and unweighted networks from several sources. We use four such biological networks.

The *helico* data set is a protein-protein interaction network from Helicobacter pylori and thus unweighted. The underlying data from 2000 and 2001 stems from the Database of Interacting Proteins [Database of Interacting Proteins].

The other protein-protein interaction network, *ecoli.interaction*, refers to E. coli. The nodes denote proteins, and the edges confirmed interactions. The data was created in 2005 in the Emili Lab [Butland et al., 2005].

Metabolic pathways in E. coli are described by *ecoli.metabolic*. Here, the nodes are metabolites. Unweighted edges between nodes denote involvement in the same catalytic

reaction. The database was developed by Ma and Zheng [2003] based on the Kyoto Encyclopedia of Genes and Genomes (KEGG).

Another protein-related network refers to protein folding. It describes the conformation space of a 20 residue antiparallel beta-sheet peptide. The conformations were sampled from simulations of molecular dynamics. Snapshots along the trajectory are grouped into nodes by secondary structure. The edges refer to transitions between structures. The network only contains conformations that occur at least 20 times in the simulation. The graph *beta3s* is a reduced version of this conformation network by Rao and Caflisch [2004].

Other protein interactions

Protein interactions in Saccaromyces cerevisiae are contained in the protein network *yeastProtInt* from Barabási. The data is further described in Jeong et al. [2001].

Other interaction networks from Saccaromyces cerevisiae were used in [Tsuda et al., 2005]. We downloaded four of their networks and used the largest component of each. In contrast to the description in the paper, all graphs are weighted. The first, *protNW1*, is based on the Pfam domain structure. A protein is represented by a 4950-dimensional binary vector, in which each bit indicates the presence or absence of one Pfam domain. An edge is created if the inner product of the two adjacent node vectors exceeds 0.06. The edge weight is the inner product. The co-participation of the node proteins in complexes, determined by tandem affiliation purification, is contained in *protNW2*. Two nodes are connected if there is a bait-prey relationship between them. Physical protein-protein interactions are covered by *protNW3*. The last network, *protNW4*, describes genetic interaction of the proteins. A detailed description of the original data can be found in Lanckriet et al. [2004].

Microarray Data

The *cellcycle* network is based on a microarray data set from Spellman et al. [1998]. The study investigates genes whose expression levels vary periodically with the cell cycle. We use their selection of 800 genes that meet the authors' criterion for cell cycle regulation. Of those genes, we deleted the three with the most missing values (where more than half the column entries were missing). One gene in the list was not in the data set. The data was preprocessed as described in the paper, referring to Eisen et al. [1998]. Ignoring the missing values, the columns (features) were standardized and, as a measure of similarity, the inner product (correlation) for all genes computed, again ignoring the missing values. From the resulting correlation matrix C, the Euclidean distances were taken as $\sqrt{2-2C}$ (entry-wise square root). The rest of the process was the same as for the other data sets. A modified version of Matlab's k-means algorithm used the Euclidean distances derived from C and took a random selection of the data points as initial centers.

Psychophysics: leaf confusion matrices

The leaf confusion matrix C, kindly provided by Frank Jäkel (unpublished data), is rather small and the result of a psychophysics experiment. The subject had to decide whether the presented leaf was leaf i or not. Entry $C(i, j)$ represents the number of times the subject identified leaf j as leaf i divided by the number of times leaf j was presented. We symmetrized this matrix by setting $W = C + C'$, and either set the diagonal to zero (*confus*) or allow self loops (*confusN*).

COSIN AS Internet graphs

The *AS-* graphs from COSIN [cosin] represent the "Autonomous Systems topology of the Internet". The nodes are autonomous systems, and an edge indicates a physical connection between two systems. The underlying BGP data has been collected by the University of Oregon Route Views Project, and is available at the "Global ISP interconnectivity by AS number" webpage [Measurement and Team] of the National Laboratory of Applied Network Research. Self loops and parallel edges were removed in the graphs at COSIN, which are in LEDA format. We use the smallest graphs in the collection, from 1997/11/08, 1998/04/02, 1998/07/03, 1998/10/02, 1999/01/14 and 1999/04/02.

Data from Newman's collection

A power network, coauthorships and political blogs are described by three graphs from Mark Newman's collection [Newman]. The topology of the Western States Power Grid of the United States forms the unweighted and undirected *power* graph [Watts and Strogatz, 1998].

The *netscience* graph represents coauthorship of scientists working on network theory and experiments. It was compiled from bibliographies of two review articles, with some references added by hand [Newman, 2006].

Political blogs from the 2004 presidential election in the United States form the basis for the *polblogs* network by Adamic and Glance [2005]. Links between the blogs were extracted from a crawl of the front page of the blogs and the posts. The study aimed to investigate interactions between liberal and conservative blogs as well as the structures of the two communities.

Emails

The *email* network from Arenas represents the email interchanges between members of the University Rovira i Virgili in Tarragona [Guimera et al., 2003].

UCI data sets

In contrast to the graphs described so far (except *cellcycle*), the UCI data sets are vectorial, giving features or coordinates for each node. For a graph representation, we constructed k-nearest neighbor graphs from this data, using Euclidean distances. The *breastcancer*, *diabetis*, *german*, *heart*, *image*, *splice* and *thyroid* data sets are provided at [Rätsch]. The data was used like that in [Mika et al., 1999] and [Rätsch et al., 2001].

In addition, we downloaded data directly from the UCI repository [uci]. For the breast-cancer-wisconsin data (*bcw*), points with missing values were removed. We also use the BUPA liver-disorders (*bupa*), *ionosphere* and pima-indians-diabetes (*pima*) sets. In each data set, we standardized the features.

Note that these data are benchmark data for classification. They do not necessarily have a clear structure for clustering which might otherwise be the intuitive "best" solution.

USPS handwritten digits

The United States Postal Service provides a database of handwritten digits from zero to nine. For experiments with two clusters, we construct graphs from pairs of two consecutive digits: zero and one, two and three, four and five, six and seven and eight and nine. Again, the features were standardized.

data	n	|E|	
ecoli.interaction	230	695	u
ecoli.metabolic	563	709	u
helico	710	1,450	u
beta3s	1,287	23,948	w
yeastProtInt	1,458	1,948	u
protNW1	641	9,791	w
protNW2	970	1,819	w
protNW3	944	1,536	w
protNW4	499	757	w
cellcycle	797	8,990 ($k = 7$)	w
confus	26	588	w
confusN	26	614	w
AS-19971108	3,015	5,156	u
AS-19980402	3,522	6,324	u
AS-19980703	3,797	6,936	u
AS-19981002	4,180	7,768	u
AS-19990114	4,517	8,376	u
AS-19990402	4,885	9,276	u
netscience	379	914	w
polblogs	1,222	16,714	u
power	4,941	6,594	u
email	1,133	5,451	u
breastcancer	257	2,012 ($k = 6$)	w
diabetis	768	7,626 ($k = 7$)	w
german	1,000	10,360 ($k = 7$)	w
heart	270	2,342 ($k = 7$)	w
image	2,086	22,122 ($k = 8$)	w
splice	2,990	47,390 ($k = 9$)	w
thyroid	215	1,752 ($k = 6$)	w
bcw	683	7,220 ($k = 7$)	w
bupa	345	2,968 ($k = 6$)	w
ionosphere	351	3,492 ($k = 6$)	w
pima	768	7,626 ($k = 7$)	w
USPS 0 vs. 1	2,822	33,588 ($k = 8$)	w
USPS 2 vs. 3	1,753	20,558 ($k = 8$)	w
USPS 4 vs. 5	1,568	18,870 ($k = 8$)	w
USPS 6 vs. 7	1,626	19,046 ($k = 8$)	w
USPS 8 vs. 9	1,529	18,660 ($k = 8$)	w

Table 3.1: Number of nodes and edges in the graphs we used in the experiments. The letter 'u' indicates that the graph is unweighted, 'w' means 'weighted'.

3.5.3 Performance on the training set

We first compare the performance of Nearest Neighbor clustering, on given graphs or k-nearest neighbor graphs from subsets of real data, with that of an algorithm designed to directly optimize the quality function Q_n. As to objective functions, we concentrate on Ncut and the within-sum-of-squares, WSS. The former is directly minimized by normalized spectral clustering (SC, for an overview of spectral clustering see e.g. [von Luxburg, 2006]). The latter criterion is the objective of the k-means algorithm.

Note that the restriction of the k-means algorithm to coordinate data makes it inappropriate for direct network data. Hence we only use WSS on the numeric vectorial data sets.

Here, we focus on finding two clusters from neighborhoods around $m = \ln n$ randomly (uniformly) chosen seed points. For the neighborhoods, a node is assigned to its closest seed by commute distance on the graph (commute distance was computed with the unnormalized Laplacian according to the matrix formula, Equation 3.3.2). Each algorithm was run with $r = 50$ initializations. For NNC, "initialization" means the choice of a set of seeds, whereas for k-means it is the choice of the initial cluster centers. In spectral clustering (SC), the means in the post-processing step involving k-means may be chosen anew. Of those r runs, the solution with the lowest objective is chosen.

Network data

network	NNC				SC
	CD	HT	ND	SND	
helico	0.159	0.183	0.167	0.167	0.159
ecoli.interaction	0.060	0.112	0.060	0.060	0.060
ecoli.metabolic	0.029	0.029	0.029	0.029	0.036
beta3s	0.003	0.003	0.003	0.003	0.003
yeastProtInt	0.035	0.035	0.035	0.040	0.056
protNW1	0.000	0.000	0.000	0.000	0.000
protNW2	0.017	0.023	0.017	0.017	1.009
protNW3	0.006	0.007	0.006	0.007	0.008
protNW4	0.011	0.011	0.011	0.011	0.013
confus	0.360	0.373	0.360	0.360	0.360
confusN	0.220	0.220	0.220	0.220	0.220
AS-19971108	0.016	0.017	0.016	0.016	0.016
AS-19980402	0.013	0.014	0.013	0.013	1.006
AS-19980703	0.021	0.098	0.021	0.021	0.021
AS-19981002	0.040	0.050	0.040	0.088	0.039
AS-19990114	0.081	0.055	0.057	0.051	0.051
AS-19990402	0.111	0.133	0.059	0.055	0.097
netscience	0.009	0.009	0.009	0.009	0.009
polblogs	0.111	0.111	0.111	0.111	0.111
power	0.003	0.003	0.003	0.004	0.005
email	0.265	0.257	0.273	0.279	0.266

Table 3.2: Ncut values for the solutions achieved with NNC and spectral clustering on the network data. The distance functions are: commute distance (CD), hitting time (HT), normalized commute distance (ND), and symmetric normalized commute distance (SND).

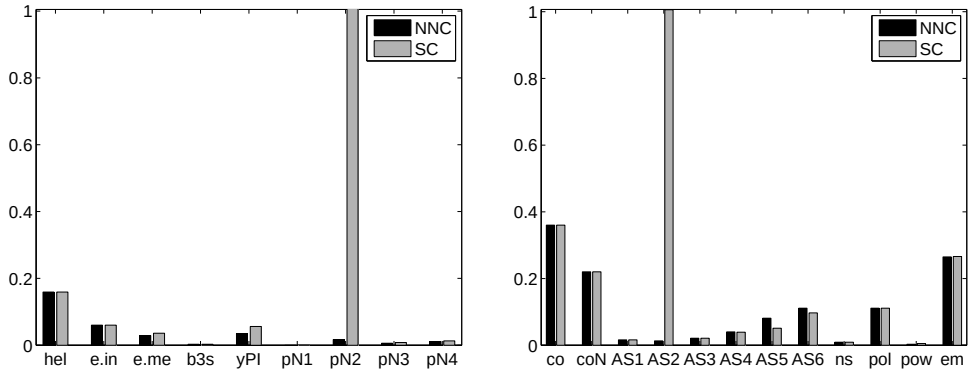


Figure 3.5.1: Ncut results for NNC with commute distance and spectral clustering, for the biological (left) and other networks (right).

Table 3.2 compares the results of NNC with various distances and spectral clustering (SC) on the networks. Figure 3.5.1 illustrates the differences for the commute distance. The influence of the distances is discussed in greater detail in Subsection 3.5.5. In general, the results of NNC and SC are comparable despite the simplicity of the NNC algorithm.

Given that NNC has been proved to be consistent, but not (yet?) SC, one would expect the solutions returned by the algorithms to differ at least to some degree. In general, they do differ, but sometimes not very much. This may be due to the relation of the commute distance and spectral clustering: both use the eigenvector(s) of the Laplacian to recognize "closeness".

In general, it is difficult to actually evaluate a clustering with respect to how reasonable it is for the given data. Hence, let us look at a small example, where the solutions hardly differ: the leaf confusion matrix. Figure 3.5.2 shows the leafs and lists the groupings. Without self loops in the graph, both algorithms yield the same solutions. If self loops are allowed, the results do not change for spectral clustering and the SND distance, and differ by the assignment of node one (leaf 2) for the commute and ND distances. The difference between the cut values is, however, minimal. Moreover, the clustering is also visually reasonable: the leaves are grouped into leaves with big dents, such as maple leaves, and more globally round-shaped or long leaves that are at most serrated.

Sometimes, on the other hand, the Q_n values for the solutions by NNC and SC differ substantially, as for the protein network *protNW2*. They still differ for networks 3 and 4, but not for network 1, where the Ncut is $40 \cdot 10^{-5}$ for all variants. Why do these networks behave so differently? One reason may be that the networks *protNW2* to *protNW4* are much sparser than *protNW1* (see Table 3.1). Another reason may be grounded in the structure of the spectrum of the graph Laplacians. The neighborhood structure in NNC is based on the distance function, which is based on the spectrum and eigenvectors of the Laplacian. This structure determines $\mathcal{F}_n$ and thus the solution of NNC. The commute and hitting times sum up differences of eigenvector entries of L, weighted by the inverse eigenvalue (cf. Subsection 3.3). ND and SND use all eigenvectors of a normalized Laplacian with a weight[5]. Spectral clustering, on the other hand, is limited to the lowest subset of the spectrum of L_{rw}, without weight on the eigenvectors. Hence, if the spectrum of L_{rw} is steep in the first part, one would not expect the distances used by SC and NNC to differ much. With a flatter

[5]The eigenvalues of L_{rw} and L_{sym} are the same [von Luxburg, 2006].

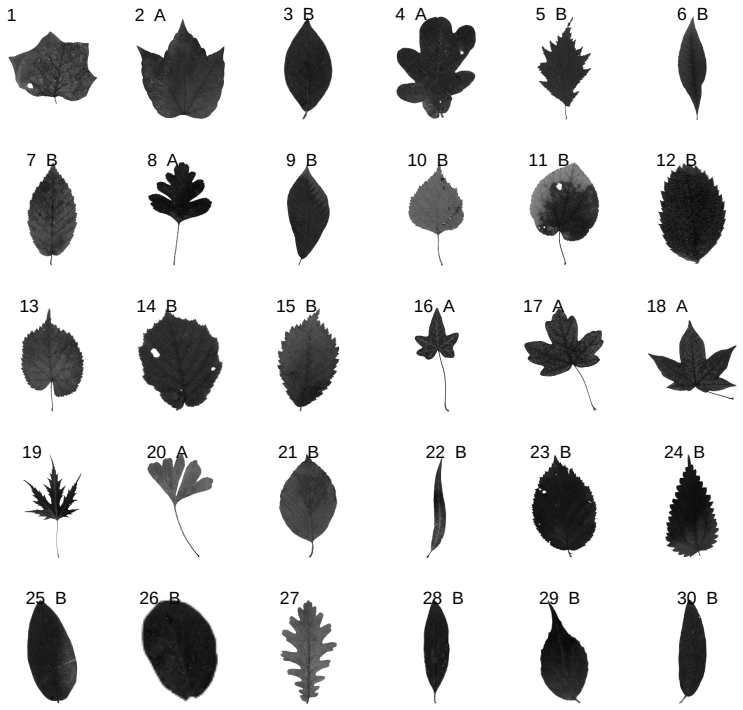


Figure 3.5.2: Leaves for the leaf confusion matrix. Leaves 1, 13, 19 and 27 are not included in the matrix. The NNC and SC clusterings then group $(2, 4, 8, 16, 17, 18, 20)$ and $(3, 5, 6, 7, 9, 10, 11, 12, 14, 15, 21, 22, 23, 24, 25, 26, 28, 29, 30)$. (image kindly provided by Frank Jäkel)

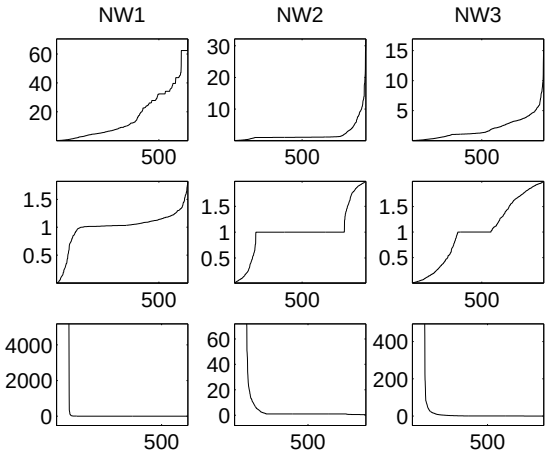


Figure 3.5.3: Spectra of the unnormalized (top) and normalized Laplacians (middle) for protein networks 1, 2 and 3. The bottom row is the weighting $1/\lambda$ for the spectrum of L_{rw}. Note that the axes have been shifted for the inverse spectra to make the graphs more visible.

spectrum, however, a larger part of the eigenvectors will receive significant weights for the CD, HT, ND and SND. This part, however, is not entirely considered in SC. As a result, the solutions of the algorithms may differ as well. Indeed, the spectrum of L_{rw} increases fastest for Network 1, as Figure 3.5.3 shows. The steepness of the spectrum does not provide a full explanation though, because the spectrum for Network 2 seems to rise faster than that for Network 3 in the beginning. Another factor may be the relative drop of weights: it is largest for Network 1, and smallest for Network 2. The striking plateau in the spectra for the normalized and unnormalized Laplacians of Network 2 could also play a role, possibly also with respect to numerical stability.

Coordinate data: k-NN and full graphs

From the coordinate data, we constructed k-nearest neighbor graphs with $k = \lceil \ln n \rceil$ and the Gaussian kernel to transfer distance to similarities (see Subsection 3.5.1). The width σ of the kernel was the overall mean distance of a point to its k-th nearest neighbor (σ_1, if marked by '*') or the mean distance to the k nearest neighbors (σ_2).

For each data set, we generated $z = 40$ training sets by subsampling $n/2$ points. Each algorithm was repeated $r = 50$ times on each training set, with different random initializations. On each set, the best of the r partitions (by the quality function) was taken as the solution. Denote the quality value of run r by $q(r)$. We report results in the form $\text{mean}_z(\min_r q(r)) \pm \text{standdarddev}_z(\min_r q(r))$.

For Ncut, we compare SC and NNC with the commute distance on the nearest neighbor graph. The WSS is optimized by k-means on the one hand and NNC with Euclidean distances on the other, the latter to better resemble k-means and the objective, both using the Euclidean distance.

Only the *cellcycle* data required a different preprocessing because of missing values. See Section 3.5.2 for further details. In consequence, we obtained a matrix of dot products that was used to construct the adjacency matrix of the graphs. A modified version of Matlab's k-means algorithm minimized the WSS for comparison.

The first row for each data set in Table 3.3 shows the Ncut and WSS values for the solutions by NNC and SC or k-means (extensions expressed in the other rows are discussed in the next subsection). The results of NNC and the comparison algorithms are, as for the networks, in the same range. For Ncut, NNC is better on some graphs, whereas for WSS,

	Ncut				WSS	
	NNC	δ	SC	δ	NNC	k-means
cellcycle*	0.10 ± 0.01		0.12 ± 0.02		0.78 ± 0.03	0.78 ± 0.03
	0.15 ± 0.03		0.16 ± 0.02		0.78 ± 0.02	0.78 ± 0.03
	0.19 ± 0.06		0.19 ± 0.04		0.80 ± 0.03	0.79 ± 0.02
breastcancer*	0.09 ± 0.02		0.11 ± 0.02		7.04 ± 0.21	6.95 ± 0.19
	0.21 ± 0.07	2.35	0.22 ± 0.07	2.06	7.12 ± 0.22	7.12 ± 0.20
	0.21 ± 0.10	2.37	0.21 ± 0.06	1.94	7.26 ± 0.23	7.18 ± 0.22
diabetis*	0.03 ± 0.02		0.03 ± 0.02		6.71 ± 0.22	6.62 ± 0.22
	0.05 ± 0.05	4.70	0.04 ± 0.03	3.07	6.72 ± 0.22	6.72 ± 0.22
	0.06 ± 0.11	6.37	0.04 ± 0.03	2.73	6.91 ± 0.23	6.83 ± 0.22
german*	0.02 ± 0.02		0.02 ± 0.02		18.56 ± 0.28	18.26 ± 0.27
	0.03 ± 0.03	3.18	0.04 ± 0.08	NaN	18.45 ± 0.32	18.35 ± 0.30
	∞	∞	∞	∞	18.90 ± 0.30	18.62 ± 0.29
heart*	0.17 ± 0.02		0.18 ± 0.03		10.77 ± 0.47	10.65 ± 0.46
	0.30 ± 0.07	1.80	0.28 ± 0.03	1.55	10.74 ± 0.46	10.75 ± 0.46
	0.29 ± 0.10	1.74	0.26 ± 0.04	1.44	11.02 ± 0.50	10.98 ± 0.46
image*	0.00 ± 0.00		0.05 ± 0.22		12.23 ± 0.72	12.17 ± 0.71
	0.10 ± 0.30	∞	∞	∞	12.27 ± 0.73	12.24 ± 0.73
	0.05 ± 0.22	∞	0.14 ± 0.34	∞	12.39 ± 0.72	12.33 ± 0.73
splice*	0.44 ± 0.16		0.36 ± 0.10		69.89 ± 0.24	68.99 ± 0.24
	0.66 ± 0.18	1.87	0.58 ± 0.09	1.80	69.18 ± 0.25	69.03 ± 0.24
	∞	∞	∞	∞	70.48 ± 0.32	69.93 ± 0.27
bcw*	0.02 ± 0.01		0.02 ± 0.01		3.98 ± 0.26	3.97 ± 0.26
	0.08 ± 0.07		0.04 ± 0.01		3.98 ± 0.26	3.98 ± 0.26
bupa*	0.13 ± 0.08		0.15 ± 0.09		4.29 ± 0.30	4.26 ± 0.31
	∞		∞			
ionosphere*	0.04 ± 0.01		0.06 ± 0.03		25.77 ± 1.63	25.72 ± 1.63
	0.14 ± 0.12		0.12 ± 0.11		25.77 ± 1.63	25.76 ± 1.63
pima*	0.03 ± 0.03		0.03 ± 0.03		6.73 ± 0.23	6.62 ± 0.22
	0.09 ± 0.13		0.05 ± 0.04		6.73 ± 0.23	6.73 ± 0.23
usps0v1	0.00 ± 0.00		0.00 ± 0.00		160.76 ± 5.03	160.47 ± 5.04
	0.01 ± 0.02	198.17	0.00 ± 0.00	1.24	160.50 ± 5.04	160.51 ± 5.04
usps2v3	0.04 ± 0.01		0.05 ± 0.01		224.14 ± 2.97	222.01 ± 2.94
	0.05 ± 0.02	1.30	0.06 ± 0.01	1.18	222.29 ± 2.95	222.14 ± 2.95
usps4v5	0.02 ± 0.00		0.02 ± 0.00		215.72 ± 3.45	214.64 ± 3.42
	0.04 ± 0.04	2.19	0.02 ± 0.00	1.28	214.74 ± 3.42	214.74 ± 3.42
usps6v7	0.00 ± 0.00		0.00 ± 0.00		186.28 ± 5.06	186.16 ± 5.05
	0.01 ± 0.02	19.62	0.00 ± 0.00	1.19	186.17 ± 5.05	186.17 ± 5.05
usps8v9	0.04 ± 0.01		0.04 ± 0.01		224.13 ± 8.45	220.59 ± 8.41
	0.10 ± 0.21	2.61	0.09 ± 0.18	7.12	224.23 ± 7.73	222.50 ± 7.86

Table 3.3: Results for coordinate data with k-NN graphs. The first row shows the mean training performance, the second and third show performance on the test set with point-wise and 'nln' extensions, respectively (Extensions are discussed in Subsection 3.5.4). δ is the mean of the quotient between test and training performance on the test set. The width of the Gaussian kernel was σ_1 for those graphs marked by '*', and σ_2 for the others. For the nln extension, it was σ_2 for all. Note: The NNC code for Ncut had a small bug here. The corrected values are in Table B.1 in the Appendix. They do not differ much, though, only that NNC is actually better than here.

Table 3.4: Results for full graphs of coordinate data. The first line for each graph is the objective value on the training set, the second line is the quality on the test set (by point-wise extension).

	Ncut	
	NNC	SC
cellcycle	0.69 ± 0.23	0.55 ± 0.18
	0.65 ± 0.15	0.56 ± 0.18
breast-cancer	0.43 ± 0.21	0.34 ± 0.18
	0.60 ± 0.05	0.58 ± 0.05
diabetis	0.07 ± 0.07	0.04 ± 0.04
	∞	∞
german	0.39 ± 0.19	0.23 ± 0.09
	0.39 ± 0.18	0.31 ± 0.04
heart	0.72 ± 0.04	0.69 ± 0.02
	0.80 ± 0.10	0.70 ± 0.02
image	0.00 ± 0.00	0.00 ± 0.00
	∞	∞
splice	0.99 ± 0.00	0.97 ± 0.00
	0.99 ± 0.00	0.97 ± 0.00

the k-means algorithm is usually a little better or equivalent. Some of the graphs, however, admittedly had an unfavorable structure. The k-nearest neighbor graph for *german* has three components, and the *image* graph has very low edge weights, down to a minimum nonzero value of $4.35 \cdot 10^{-138}$. The Laplacian of *usps0v1* is badly conditioned, so the results are questionable for this graph, too. The remaining data should be well-behaved.

One might argue that the proof of consistency assumes that the full spatial information, represented by exact distances, is available, whereas the pruning of edges to nodes other than the k nearest neighbors may introduce inaccuracies. Hence we also constructed full graphs from some coordinate data sets. Table 3.4 summarizes the results (first row for each data set). Here, NNC performs a little worse than spectral clustering. The problem with these full graphs is that the spectrum often seems to be very biased to one point: for distances based on the spectrum, such as the commute distance, one single seed point is the closest to almost all other points. In that case, the resulting solution will be very unbalanced. The function space has been reduced in an unfavorable way. This does not happen as strongly for the k-NN graphs. One reason may be that the additional many small edges average out the high edge weights (when, for probability in the Markov chain, edge weights are divided by the degree), and thus the graph is more similar to a clique than the k-NN graph. In a clique, the cuts are more equal and the structure is weaker. Then the eigenvectors corresponding to the end of the spectrum of the unnormalized Laplacian are also more prone to converge to Dirac functions (cf. Section 3.3). In fact, the ND and SND distances, motivated by the Dirac problems, lead to better results here, which sometimes reach the performance of spectral clustering.

3.5.4 Generalization ability

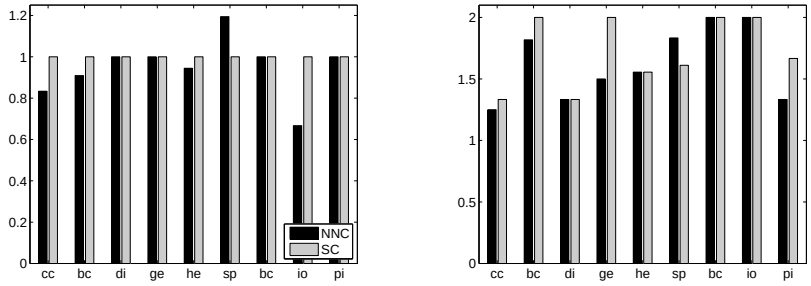
After observing that NNC performs comparably to "direct optimization" algorithms on a training set, we would like to measure the amount of overfitting induced by the algorithms. For each of the coordinate data sets we clustered $n/2$ points and extended the clustering to the other $n/2$ points. Then we compared the objective function values on the test set labeled by the extension. This experiment was also repeated $z = 40$ times, and in each repetition, the partition out of $r = 50$ initializations was chosen as f_n.

There are several possibilities of extension, for example

point-wise (pw) Add the test points one by one (separately) to the training set and assign the label that leads to the lower objective value on the $n/2 + 1$ points, leaving the labels of the $n/2$ training points fixed.

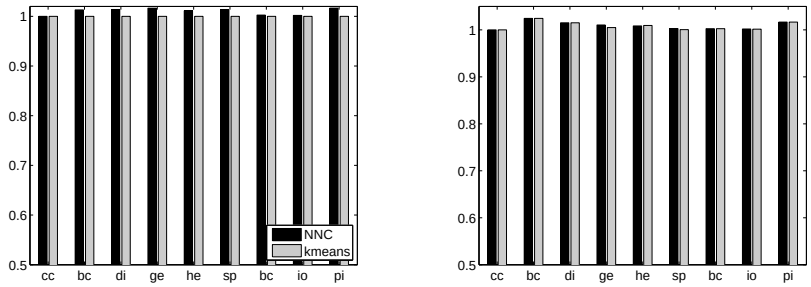
nearest labeled neighbor (nln) Label the entire test set at once by assigning to each test point the label of its nearest labeled neighbor (nln). Closeness is measured by the distance used to create the neighborhoods.

closest cluster center With WSS and the k-means algorithm, one might also want to assign the test points to their closest cluster center. This yields, for our graphs, the same results as the point-wise extension.



(a) Average Ncut value on the training set, NNC versus spectral clustering (SC)

(b) Average Ncut value on the test set with point-wise extension, NNC versus SC



(c) Average WSS value on the training set, NNC versus k-means

(d) Average WSS value on the test set with point-wise extension, NNC versus k-means

Figure 3.5.4: Value of Q_n with direct optimization and extension (from Tables 3.3 and B.1). All values are normalized by the objective value for the respective comparison algorithm (SC, k-means) on the training set. For WSS, the quality of the partition of the test set is similar to or even better than that of the training set, whereas for Ncut, the extension comes with a greater relative loss.

Table 3.3 shows that for the extensions, NNC is roughly comparable the other algorithms. This result questions our expectation of NNC being less prone to overfitting. The most likely explanation is that both k-means and spectral clustering already have reasonably good extension properties. This may be due to the fact that, like NNC, both algorithms

consider only a subclass of all partitions: Voronoi partitions for k-means and partitions induced by eigenvectors for spectral clustering.

The value ∞ occurs in the table if the extension generated an empty cluster in one of the z runs and Ncut is the criterion. The table also shows the mean ratio of the objective values for labeling via an extension versus running the algorithm directly on the set as training set. This ratio is often worse for NNC than for SC. Note, however, that the NNC training quality values are in general lower than those for SC.

Figure 3.5.4 illustrates the difference of training and extension qualities relative to the training performance of SC and k-means for some of the data in Table 3.3 for WSS and Table B.1 for Ncut. That means we divided the test quality by the training quality. With Ncut, the extension appears to come with a greater loss than with WSS. For the latter, NNC has a better test than training performance on some data sets. For k-means, this only occurs on the *splice* data set. This slight decrease or stagnation in the objective from direct labeling to extension for NNC, together with a simultaneous slight increase in the objective value for k-means, equalizes the test performance of both algorithms.

In summary, the compared algorithms perform similarly with respect to extensions of their solutions. Similar results were achieved by an analogous comparison of NNC and SC with the RatioCut quality function.

More than 2 clusters

In a setting like above, we tested the training and extension performance for three and four clusters. Tables 3.5, 3.6 and 3.7 show the results. For Ncut, we used the commute distance and point-wise extensions, and for WSS the Euclidean distance. For the latter, a point in the test set was assigned to its closest cluster center.

The more clusters we search for, the higher is the probability that one of these clusters will not be assigned a point in the extension on the test set, particularly in small data sets. The risk is probably even higher if the clusters of the training set are unbalanced. An empty cluster in the test set leads to an infinite Ncut value. Therefore, the test performance is an average only over those repetitions in which neither algorithm generated an empty cluster on the test set. For comparison, the tables also indicate the corresponding training error, averaged only over these non-infinity runs.

Apparently, the solutions of spectral clustering are more likely to end up with empty clusters on the extension. Often, the extensions for SC seem to be worse than for NNC, possibly indicating an overfitting by SC. This tendency was less striking for only two clusters. In fact, judging from the averages in Figure 3.5.5, NNC performs better than SC on the training set but worse on the extension. The NNC extensions for more clusters, however, are equal to or better than those of SC, despite relatively worse training performance. In some cases with $K = 3$, the training performance for NNC is worse than for SC, but the objective on the extension is better. Note that with more clusters, the number of points per cluster is smaller. Maybe the advantages of NNC against overfitting come more into play in such a setting.

With the WSS objective, the training and test performance of NNC is mostly slightly worse than that of k-means. Nevertheless, the same tendency as for two clusters appears: Whereas the value of Q_n is often lower on the extension than on the training set for NNC, the opposite is the case for k-means, closing the gap between the performance of NNC and k-means. Looking at Figure 3.5.5, this relative difference between the algorithms on training and test set seems to become stronger on average.

In summary, these experiments demonstrate that NNC's solutions are comparable to

	NNC			SC		
distance	Q_n	δ	%∅	Q_n	δ	%∅
ED	0.74 ± 0.07			0.47 ± 0.05		
	1.06 ± 0.41	1.48	7.5	1.40 ± 0.48	3.03	27.5
CD	0.57 ± 0.09			0.47 ± 0.05		
	1.12 ± 0.45	1.98	7.5	1.43 ± 0.49	3.09	27.5
HT	0.83 ± 0.22			0.47 ± 0.05		
	1.33 ± 0.42	1.78	22.5	1.37 ± 0.52	3.01	27.5
ND	0.50 ± 0.06			0.47 ± 0.05		
	1.02 ± 0.44	2.05	10.0	1.43 ± 0.48	3.05	27.5
SND	0.48 ± 0.05			0.47 ± 0.05		
	1.07 ± 0.48	2.23	12.5	1.38 ± 0.47	2.97	27.5

Table 3.5: Ncut results for NNC and spectral clustering for 4 clusters on the *cellcycle* data. The first line is Q_n on the training set, the second Q_n on the test set labeled by point-wise extension. δ is the average ratio between test and training performance and %∅ the percentage of the z runs in which the extension created an empty cluster. The average Q_n value on the extension is only over runs with nonempty clusters. The kernel width was σ_1. Obviously, the results for NNC depend on the distance used. The extension of SC's solution is more likely to create empty clusters.

	NNC		SC		NNC		k-means	
data	Q_n	%∅	Q_n	%∅	Q_n	%∅	Q_n	%∅
bcw	0.08 ± 0.02		0.09 ± 0.02		3.28 ± 0.17		3.24 ± 0.17	
	0.08 ± 0.02		0.08 ± 0.02		3.28 ± 0.17		3.24 ± 0.17	
	0.43 ± 0.31	22.5	0.43 ± 0.38	37.5	3.30 ± 0.19	0.0	3.32 ± 0.21	0.0
ionosphere	0.12 ± 0.04		0.15 ± 0.05		23.39 ± 1.68		23.18 ± 1.67	
	0.12 ± 0.04		0.14 ± 0.05		23.39 ± 1.68		23.18 ± 1.67	
	0.44 ± 0.33	12.5	0.63 ± 0.38	30.0	23.73 ± 1.65	0.0	23.71 ± 1.67	0.0
pima	0.13 ± 0.05		0.11 ± 0.03		5.86 ± 0.22		5.65 ± 0.21	
	0.12 ± 0.03		0.11 ± 0.02		5.86 ± 0.22		5.65 ± 0.21	
	0.20 ± 0.19	10.0	0.35 ± 0.41	45.0	5.73 ± 0.20	0.0	5.68 ± 0.21	0.0
breastcancer	0.22 ± 0.05		0.25 ± 0.05		6.08 ± 0.17		5.89 ± 0.15	
	0.22 ± 0.05		0.25 ± 0.05		6.08 ± 0.17		5.89 ± 0.15	
	0.60 ± 0.31	7.5	0.79 ± 0.32	5.0	6.03 ± 0.18	0.0	5.98 ± 0.17	0.0
diabetis	0.12 ± 0.05		0.11 ± 0.04		5.84 ± 0.24		5.63 ± 0.23	
	0.12 ± 0.05		0.11 ± 0.03		5.84 ± 0.24		5.63 ± 0.23	
	0.18 ± 0.09	15.0	0.32 ± 0.39	52.5	5.72 ± 0.24	0.0	5.68 ± 0.24	0.0
heart	0.44 ± 0.06		0.42 ± 0.06		9.97 ± 0.34		9.71 ± 0.33	
	0.44 ± 0.06		0.42 ± 0.06		9.97 ± 0.34		9.71 ± 0.33	
	0.94 ± 0.31	2.5	0.85 ± 0.19	0.0	9.98 ± 0.35	0.0	9.97 ± 0.35	0.0

Table 3.6: Training and extension results for 3 clusters and Ncut (left) and WSS (right). The first line is the average training error on all z repetitions, the second line the training error on only those repetitions where the extension did not end up with an empty cluster, and the third line is Q_n on the test set labeled by point-wise extension (Ncut) or assignment to the closest center (WSS).

data	NNC Q_n	%∅	SC Q_n	%∅	NNC Q_n	%∅	k-means Q_n	%∅
bcw	0.15 ± 0.03		0.17 ± 0.04		2.93 ± 0.16		2.81 ± 0.14	
	0.16 ± 0.04		0.17 ± 0.04		2.93 ± 0.16		2.81 ± 0.14	
	0.61 ± 0.43	47.5	0.49 ± 0.24	30.0	2.86 ± 0.13	0.0	2.84 ± 0.14	0.0
ionosphere	0.23 ± 0.06		0.26 ± 0.07		21.39 ± 1.77		21.10 ± 1.71	
	0.24 ± 0.06		0.27 ± 0.09		21.39 ± 1.77		21.10 ± 1.71	
	1.02 ± 0.60	35.0	0.88 ± 0.53	35.0	21.46 ± 1.74		21.40 ± 1.72	
pima	0.33 ± 0.07		0.33 ± 0.05		5.33 ± 0.20		5.06 ± 0.18	
	0.33 ± 0.05		0.33 ± 0.04		5.33 ± 0.20		5.06 ± 0.18	
	0.72 ± 0.41	15.0	1.12 ± 0.69	47.5	5.20 ± 0.21	0.0	5.14 ± 0.21	0.0
breastcancer	0.38 ± 0.07		0.40 ± 0.06		5.54 ± 0.17		5.31 ± 0.15	
	0.37 ± 0.08		0.39 ± 0.06		5.54 ± 0.17		5.31 ± 0.15	
	1.14 ± 0.56	15.0	1.27 ± 0.47	17.5	5.59 ± 0.17	0.0	5.52 ± 0.16	0.0
diabetis	0.34 ± 0.07		0.33 ± 0.05		5.30 ± 0.21		5.04 ± 0.19	
	0.34 ± 0.07		0.32 ± 0.05		5.30 ± 0.21		5.04 ± 0.19	
	0.82 ± 0.35	20.0	0.90 ± 0.53	50.0	5.19 ± 0.25	0.0	5.13 ± 0.21	0.0
heart	0.82 ± 0.10		0.74 ± 0.09		9.36 ± 0.33		8.99 ± 0.32	
	0.82 ± 0.10		0.74 ± 0.09		9.38 ± 0.30		9.01 ± 0.29	
	1.74 ± 0.42	15.0	1.70 ± 0.39	25.0	9.42 ± 0.37	2.5	9.35 ± 0.34	0.0

Table 3.7: Clustering and extension results like in Table 3.6, but for 4 clusters.

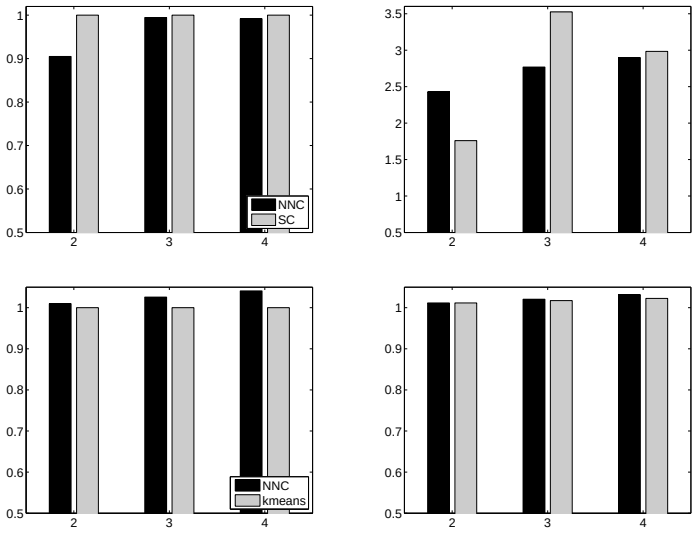


Figure 3.5.5: Quality values (normalized by the performance of the comparison algorithm on the training data) on the training (left) and test sets (right) for 2, 3 and 4 clusters for the data in Tables 3.6 and 3.7. The objective was Ncut (top) or WSS (bottom).

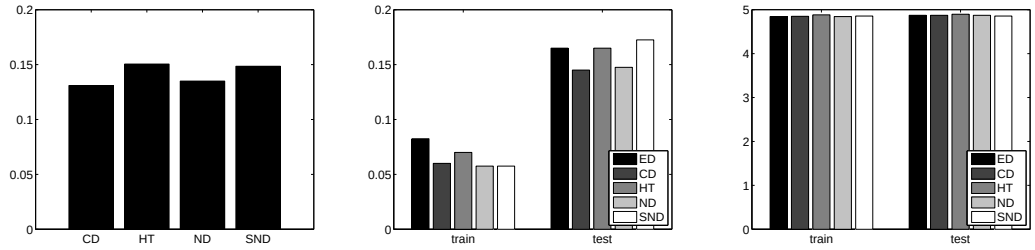


Figure 3.5.6: The distances' impact on the average objective values on the training and test sets. Left: Networks (Table 3.2), middle: Ncut for coordinate data (breastcancer, diabetis, german, heart, image, splice, cellcycle, the USPS sets, bcw, ionosphere, pima with σ_2), right: WSS for coordinate data (breastcancer, diabetis, german, heart, image, splice, thyroid, cellcycle with σ_2).

those of direct optimization algorithms even for more than two clusters, and they also show good extension properties even on finite graphs with a relatively small average number of nodes per cluster. The extensions of NNC's solutions even seem to be less likely to generate empty clusters.

3.5.5 Distances

Section 3.3 introduces a choice of distance functions to use in Nearest neighbor clustering. In the following, we will analyze the impact of the different distances on the outcome of the algorithm.

By intuition, a distance must fit the objective function in capturing the concept of "density", "closeness", "connectedness" inherent in Q_n and the definition of a "good clustering" Q_n represents. In addition, the structure of the particular graph might play a role, too. On well-separated data sets, the distances may all be very similar. The difference between the Euclidean and commute distances, for instance, depends on the embedding: the distances will differ a lot in a graph that is shaped like a 'C'. Hence, there is no universal solution to the problem of which distance suits best, and the exact correspondence between distance and quality function still remains a question to be answered.

Distances and Q_n

Tables 3.2 and 3.5 of the objective values on networks, training sets and extensions demonstrate that the quality of the clustering can depend on the distance.

For Ncut, the Euclidean distance is usually the worst, because it only covers distances in the embedding space $\mathbb{R}^d$, and not directly the graph structure with the existing edges. It is related to the edge weights by the Gaussian kernel, but many such edges were pruned in the graph. The other distances are related by the graph Laplacians, which also form the connection to the relaxed Ncut problem that is solved by spectral clustering. They consider the graph structure via the Laplacian.

These tendencies show up in Figure 3.5.6, which compares the average objective values for different distances. The solutions were generated with the same seed sets. For Ncut, the commute distance works best on average, for direct optimization and extensions of the clustering. On k-nearest neighbor graphs, the Euclidean distance is least appropriate. The

other distances are very similar on the networks (see also Table 3.2). The SND distance differs most between training and test: whilst Q_n is rather low on the training set, it even exceeds the objective for the Euclidean distance on the extension. This looks like overfitting, but may also be grounded in bad extension properties of SND, as it is also used to find the nearest neighbor for labeling the test points. Otherwise, CD, ND and SND are comparable on the training set and given networks, as is also the tendency in Table 3.5 for four clusters. CD, ND and the hitting time give, on average, the best solutions on the extension.

The distance most in line with WSS is the Euclidean distance, as becomes obvious in Figure 3.5.6. For WSS, performance is best with the Euclidean distance, even though the average differences between the distances are not great for the tested graphs.

Properties of the neighborhoods

The distance function defines $\mathcal{F}_n$ via the neighborhoods it helps to generate. It determines which seed a node is assigned to. Let us thus take a look at the properties of neighborhoods.

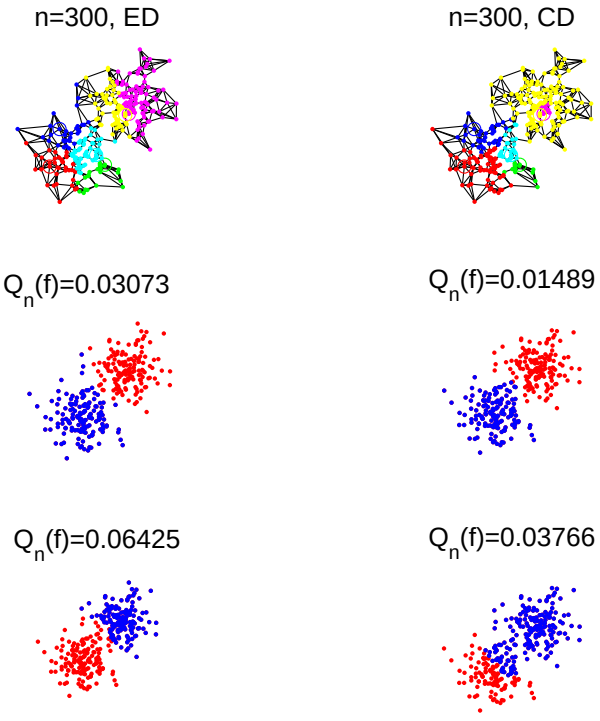


Figure 3.5.7: Neighborhoods for Euclidean (left) and Commute distance (right) with the corresponding best Ncut clustering (second row). The third row is the best clustering with a different set of seeds. There, the neighborhood structure prevented a better cut.

Figure 3.5.7 shows an example of neighborhoods in a toy graph with $n = 300$ points sampled from two Gaussians. Whilst the neighborhoods with the Euclidean distance have a circular shape, the neighborhood cells by commute distance are more uneven, and can

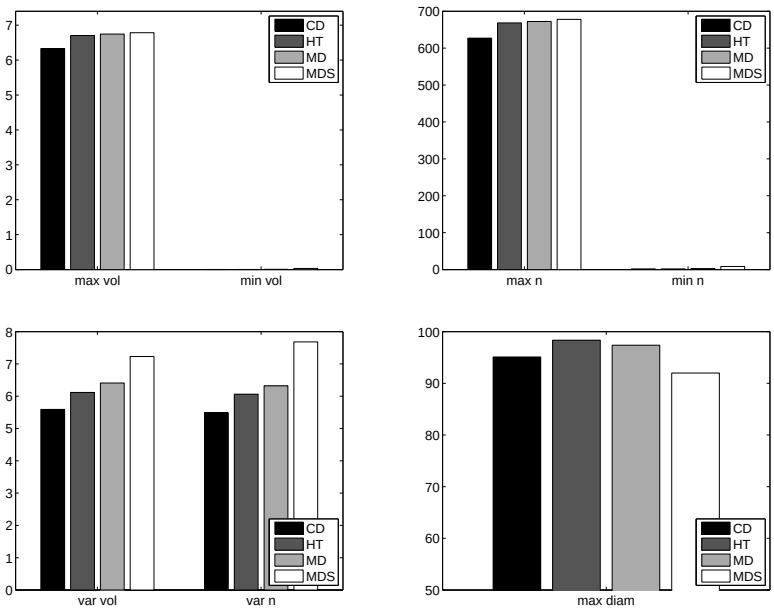


Figure 3.5.8: Neighborhood properties for a selection (see text) of the network graphs. From left to right, top to bottom: Maximum and minimum normalized volume, maximum and minimum size (nodes), variance in volume and number of nodes, maximum diameter in percent of the diameter of the entire graph. The values are averages over 50 repetitions and the different graphs.

here even be ring-shaped. A closer look at such plots reveals one problem: a neighborhood crossing the region of low density rules out any cut through this area. Exactly this region with few edges, however, is the intuitive place for a cut. The growing number of Voronoi cells, resulting from an increase of n, will remedy this problem for large n.

Apart from the toy graph, a further analysis of the neighborhood structure seems interesting. How much do the sizes of the cells vary? A very unbalanced distribution of neighborhood sizes may only leave unbalanced partitions in $\mathcal{F}_n$ that might result in empty clusters when extended on a test set. Therefore we computed some properties of the neighborhoods on a selection of graphs for $r = 50$ seed sets with different distances. Of the networks, we used *AS-19981002*, *AS-19980402*, *AS-19971108*, *helico*, *email*, *beta3s*, *protNW1-4*, *polblogs*, *ecoli.interaction*, *ecoli.metabolic* and *netscience*, and of the coordinate data sets *breastcancer*, *cellcycle*, *diabetis* and *ionosphere*.

The first plot in Figures 3.5.8 and 3.5.9 shows the average volume of the smallest and largest cell (by volume), normalized by $100 \log(n)/\operatorname{vol}(G)$. The average maximum and minimum size by number of nodes behaves similarly. For the networks, the difference between minimum and maximum size and volume seem to be even greater than for the coordinate data sets. In fact, on eight of the fifteen graphs (*AS-19981002*, *AS-19980402*, *AS-19971108*, *helico*, *email*, *beta3s*, *protNW2*, *polblogs*), the smallest neighborhood by commute distance and hitting time has an average volume of zero, so it consists of one node only. The ND distance leads to similarly small neighborhoods. On the coordinate sets, however, this was not the case. It may be that the degree is more balanced in the k-NN graphs. One would expect it to be better if the sizes vary not too much but provide a basis for a balanced cut.

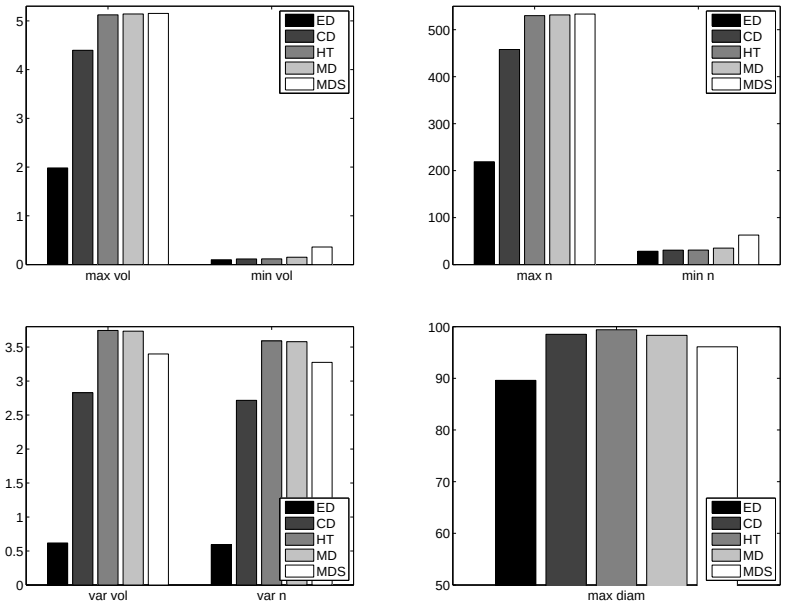


Figure 3.5.9: Neighborhood properties like in Figure 3.5.8, but for coordinate data sets *breastcancer*, *cellcycle*, *diabetis* and *ionosphere*.

Particularly, a cell of one node only does not intuitively seem like a helpful choice.

The third plot then illustrates the average variance in neighborhood volume and size. On average, the Euclidean distance seems to generate the most equal clusters in size, whereas SND (for the networks) or the hitting time and ND (for coordinate data) create the most unequal ones. The larger variance of the HT in comparison to the CD is intuitively reasonable by the interpretations: for the HT, we assign a node to the seed it reaches fastest, independent of the return time. The CD considers both directions and thus also takes of the density around the seed (cf. Subsection 3.3). The variance in volume, however, does not directly covary with the quality of the clusterings (Figure 3.5.6). For the selection of network graphs, the best results were on average achieved with the commute distance, whereas the hitting time seems less appropriate. This result is in part in conformity with the variance properties. SND leads to less favorable results on some networks, but on average it seems not bad, contrary to the average of the variance in neighborhood size.

The last plot shows the average diameter of the cell with the largest diameter, in relation to the diameter of the entire graph (in percent). Apparently, the hitting time leads to the most spread-out neighborhoods. Strikingly, for almost all distances, the diameter of the longest neighborhood cell is very close or identical to the diameter of the graph, so the cell stretches over the entire graph. In view of the large variance in cell sizes, however, this observation becomes less surprising.

3.5.6 Selection of seeds

The dependence of neighborhoods and cluster quality on the distances raises the question of the relation between the set of seeds and the clustering. Therefore we attempt to study the correlation of certain properties of the set of seeds with the quality of the partition.

What makes a "good" set of seeds?

The first experiment investigates how different measures on the seed set correlate with the Ncut and Rcut value. The results for both quality functions were similar, so we concentrate on Ncut here. The neighborhoods were computed with three distances: the commute distance, the hitting time from a seed to a node and the hitting time from a node to a seed. Apart from the graphs for *thyroid* ($k = 6$, $\sigma = 10$), *heart* ($k = 8$, $\sigma = 100$), *USPS0vs1* ($k = 6$, $\sigma = 100$), *USPS2vs3* ($k = 6$, $\sigma = 100$), *USPS8vs9* ($k = 6$, $\sigma = 100$), we used ten instances of each of four toy graphs. The toy graphs were constructed as nearest neighbor graphs with $k = 5$ neighbors. For the nodes, we sampled n_1 and n_2 points from two Gaussian distributions with means μ_1, μ_2, and variances Σ_1, Σ_2:

1. $\mu_1 = (0, 0)$, $\mu_2 = (3, 3)$; $\Sigma = I$, $n_1 = n_2 = 800$
2. $\mu_1 = (0, 0)$, $\Sigma_1 = (0.5, 0; 0, 1)$; $\mu_2 = (3, 0)$, $\Sigma_2 = (0.5, 0; 0, 2)$; $n_1 = 534$, $n_2 = 1066$
3. $\mu_1 = (0, 0)$, $\Sigma_1 = (0.5, 0; 0, 1)$; $\mu_2 = (3, 0)$, $\Sigma_2 = (0.5, 0; 0, 2)$; $n_1 = n_2 = 800$
4. $\mu_1 = \texttt{zeros}^6(10, 1)$, $\Sigma_1 = \texttt{diag}([0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 0.4, 0.4, 0.8, 0.8])$, $\mu_2 = 1.5 \cdot \texttt{ones}(10, 1)$, $\Sigma_2 = \texttt{diag}([0.8, 0.4, 0.5, 1.0, 0.5, 0.6, 0.8, 1.0, 1.0, 0.4])$; $n_1 = n_2 = 800$.

We computed the following measures and took averages over $r = 100$ randomly (uniformly) chosen seed sets of size $\lceil \ln n \rceil$ for each graph:

1. Average degree of the seeds: If degree is seen as an estimate for density, this measure encodes if a node is "central" within a cluster or at the boundary where the density decreases. Is it better if the seeds are located within clusters or at the boundary? The relative location of the sample points to each other, though, is ignored.
2. Variance of the degrees of the seed points, divided by the variance of the degrees of the entire graph: maybe it is better to have samples from regions of different densities (encoded by different degrees)?
3. Maximum commute distance from a seed to a non-seed node: Intuitively, this is a worst-case measure how well the sample covers the data space, that means how well it is distributed.
4. Maximum hitting time from a seed to a non-seed point: $\max_{x \in S, y \in V \setminus S} H(x, y)$, where S denotes the seed set. A motivation for this was Matthew's theorem [Aldous and Fill, 2001, Thm. 26, Chapter 2] which states that the maximum and minimum expected cover time (over all vertices) can be bounded by the maximum and minimum HT between any two vertices, respectively.
5. Maximum hitting time from a non-seed point to a seed point: $\max_{x \in S, y \in V \setminus S} H(y, x)$
6. Average commute distance between the seeds: This measure shows how distributed the seed set is on average.
7. Minimum commute distance between the seeds
8. Maximum commute distance between the seeds

[6] The vectors and matrices are noted like calls to Matlab functions to save space.

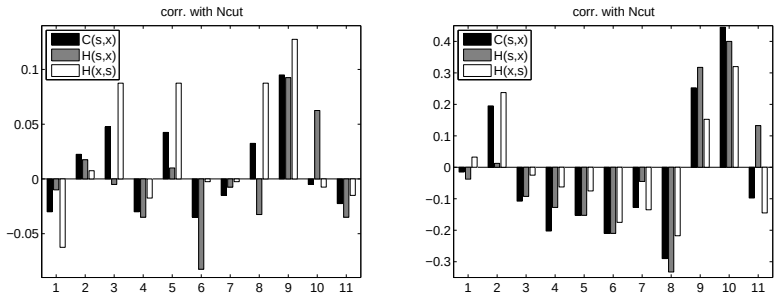


Figure 3.5.10: Average correlations of the measures with Ncut. The x axis refers to the index of the criterion.

Toy data Real data

9. Maximum commute distance from a point to its closest seed point: $\max_{y \in V \setminus S} \min_{x \in S} C(x, y)$. This measure of maximum spread of a neighborhood is related to how well distributed the seeds are.

10. Maximum hitting time from a seed point to a node that is assigned to it: $\max_{y \in V \setminus S} \min_{x \in S} H(x, y)$

11. Maximum hitting time from a node to the seed it is assigned to: $\max_{y \in V \setminus S} \min_{x \in S} H(y, x)$.

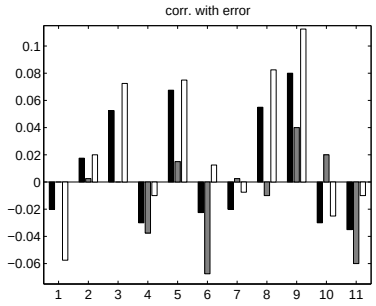


Figure 3.5.11: Correlation with "error" for the toy graphs.

We computed the Pearson correlation of each measure with the Ncut value and classification error. For the error, cluster labels were taken as class labels and true labels corresponded to the distribution a node was drawn from. The correlation varied a lot from one graph to the other, so that a definitive conclusion was impossible from the obtained data. In addition, the data was mostly widespread, so that a linear relationship was at least not directly obvious by visual judgment.

In general, correlations were stronger for the real data sets than for the toy graphs. The latter were possibly more regular. The average correlations for the artificial and real data are partly contradictory, as Figure 3.5.10 demonstrates. Measures 3, 5 and 8 correlate positively for the toy graphs, but negatively for the real data. Intuitively, we expected a positive correlation for 3 and 5, because larger distances between seeds and the other points indicate that the seeds might not be well-distributed. On the other hand, these measures may be easily distorted by outlier points. For Measure 8, one would expect a negative correlation, if better distributed seeds create more even neighborhoods. On the other hand, the largest maximum distance is realized if all seeds but one are concentrated in one place, and one outlier is far apart. The latter could be the reason for the positive correlation for the real data. In retrospect, Measure 8 cannot capture well the covering of the entire seed set.

A rather strong correlation on both sets is with the maximum commute distance between a node and the seed it is assigned to (Measure 9). The larger the distance, the more spread is at least one neighborhood cell, and this seems to be negative for the cluster quality. Surprisingly, the same measure with the hitting time from node to seed behaves differently. This observation indicates the difference between the commute distance and its summands based on hitting time.

Measure 6, the average commute distance between the seeds, correlates negatively on both sets. This is somewhat surprising, if the measure covers the spread of the seed set, because one expects well-distributed seeds to create more even neighborhoods. Similarly to Measure 8, an outlier could though distort the average. Again, the exact location of the seeds is ignored but probably important.

The correlations with "classification error", shown in Figure 3.5.11, are very similar to those with Ncut in Figure 3.5.10.

In sum, the correlations were rather weak and diffuse. Possibly the direct correlation was not the best measure. Moreover, we neither checked for combinations of measures nor nonlinear relations. This further analysis may have revealed more information, as it is probably more than one factor that counts.

Picking the seeds (non-uniformly) by degree

The previous subsection demonstrates that it is hard to make out what exact properties a "good" set of seeds shows. The hope was that the quality of the partition by NNC could be enhanced by a biased draw of the seeds.

One simple criterion for seeds is their degree. Figures 3.5.10 indicates a slight negative correlation of the degree of the seeds with the Ncut value. Therefore we studied the effect of choosing the seed nodes uniformly from the $p\%$ of V with the highest degree. The percentage was set to $p = 100$, 95, 80, 50, and 20. We constructed nearest neighbor graphs ($k = 5$, $\sigma = 1.0$) from $n = 800$ and $n = 1600$ points drawn from a mixture of two Gaussians. The following models were used:

0 $\mu_1 = (0, 0)$, $\mu_2 = (3, 3)$, $\Sigma_i = I$.

1 $\mu_1 = (0, 0)$, $\mu_2 = (3, 0)$, $\Sigma_i = (0.5, 0; 0, 2)$.

2 $\mu_1 = (0, 0)$, $\mu_2 = (3, 0)$, $\Sigma_1 = (0.5, 0; 0, 1)$, $\Sigma_1 = (0.5, 0; 0, 2)$.

3 $\mu_1 = (0, \ldots, 0)$, $\mu_2 = (1.5, \ldots, 1.5)$, $\Sigma_1 = \texttt{diag}(0.5, 0.5, 0.5, 1.0, 1.0, 1.0, 0.4, 0.4, 0.8, 0.8)$, $\Sigma_2 = \texttt{diag}(0.8, 0.4, 0.5, 1.0, 0.5, 0.6, 0.8, 1.0, 1.0, 0.4)$.

For each model, we generated 10 instances and repeated NNC with $r = 100$ seed sets on each. The results in Figure 3.5.12 are averages over those runs, for each model. The tendencies were similar for $n = 800$ and $n = 1600$ nodes, thus we only show the figures for the latter.

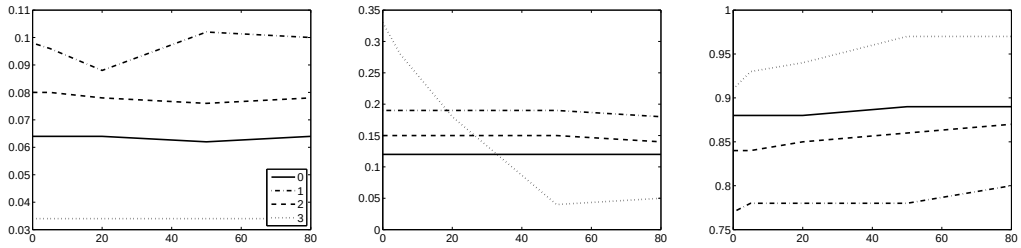


Figure 3.5.12: Best Ncut value (left), average Ncut value (middle) and average volume ratio of the clusters (right) over $r = 100$ seed sets for 40 artificially generated graphs from 4 models with $n = 1600$ nodes. The x axis denotes p, number 0 to 3 are the graph numbers.

The first plot illustrates the average best Ncut value for the $r = 100$ repetitions. It remains roughly stable as p varies. The average Ncut value (middle plot), however, decreases

with the restriction of the seed candidates. Here, the graphs behave differently though: the change is minor for graphs 0 to 2, but large for graph 3, which is higher-dimensional and maybe well-separable. The rightmost plot displays the average ratio of the two cluster volumes, $\mathrm{vol}(\mathcal{C}_0)/\mathrm{vol}(C_1)$, where $\mathrm{vol}(\mathcal{C}_0) < \mathrm{vol}(\mathcal{C}_1)$. This ratio increases with the degree limitation, indicating that the cluster volumes become more equal. Hence, there may be a change in the structure of the neighborhoods.

In conclusion, even if the tendencies are small and depend on the graph, a bias of the seeds towards larger degrees seems to improve the partitions and make the volumes of the clusters more balanced.

Seed sets of varying size

Recall that the complexity of the NNC algorithm is polynomial for a seed set of size $c \log n$, where c is constant. This constant should be chosen wisely, though, as the size of $\mathcal{F}_n$ is exponential in c. The next experiment demonstrates how the partitions change as c grows from 1 to 3. As expected, the average Ncut values decrease.

Here we used the same graph models as in the previous subsection, generating again 10 instances for each model and sampling $r = 100$ seed sets for each graph. We repeated the same procedure for $n = 800$ and $n = 1600$ nodes. We always picked $\lceil c \ln n \rceil$ seeds. The size of the seed set S determines the size of the function class $\mathcal{F}_n$. As it grows, we can choose f_n from a larger variety of candidates and hence expect to achieve better objective values. Note, however, that we did not investigate the extension properties of these partitions.

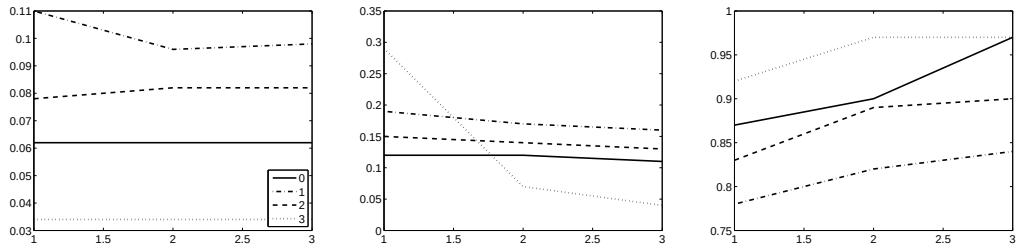


Figure 3.5.13: Best Ncut value (left), average Ncut value (middle) and average volume ratio of the clusters (right) over $r = 100$ seed sets for 40 artificially generated graphs from 4 models with $n = 1600$ nodes. The x axis denotes c.

As shown in the first plot in Figure 3.5.13, the Ncut value of best of the r partitions does not change much as the seed set grows. The average Ncut value, on the contrary, decreases for all graphs (middle plot). So the probability of achieving a good cut by Q_n is higher. As in the previous experiment with seed degrees, the tendencies are strongest on the higher-dimensional graph (Model 3). The rightmost plot demonstrates that, as the number of neighborhood cells increases, the volumes of the clusters become more balanced. The balance of cluster volumes is one criterion considered by the Ncut objective. The dependency of the volume ratio on c is reasonable, as a finer cell structure allows for a better fine-tuning of the cut and better balancing.

On the whole, the observed tendencies are analogous to those in the previous experiment: both a bias in the degree of the seed nodes and a larger seed set lead to a higher probability of achieving a good partition (by Ncut as Q_n), and to more balanced cluster volumes.

3.6 Summary and Discussion

In this chapter, we presented an approach to clustering that aims to achieve statistical consistency by a reduction of the space of candidate functions. We only allowed functions that are constant on Voronoi tessellations of the space. The result is an algorithm that is statistically consistent and runs in polynomial time, moving the graph cut problem for objectives like Ncut and RatioCut from NP to P. In that regard, we achieved the goal of simplifying an NP-hard optimization problem based on reflections from Statistical Learning Theory, replacing heuristics by controllable simplifications.

In addition, the proof of consistency is stronger than results for other common clustering algorithms. Pollard [1981] proved that the minimizer of WSS on a finite sample converges to the true global minimizer. But the k-means algorithm is not guaranteed to find this minimizer. The solution of spectral clustering converges to a limit clustering [von Luxburg et al.], which is, however, conjectured to potentially deviate from the true minimizer of Ncut. Recall that spectral clustering only solves a relaxed version of the Ncut problem, whereas NNC directly optimizes Q_n over $\mathcal{F}_n$.

Nevertheless, there is more than a theoretical side to each algorithm. We showed how to improve the average running time of Nearest neighbor clustering by branch and bound and certain heuristics. Experiments reveal that NNC performs roughly the same as direct optimization algorithms, on both training and test sets. Against our expectations, particularly for $K = 2$, it could not outperform the comparison algorithms as to generalization. A reason may be that spectral clustering and k-means inherently reduce the space of candidate functions, similar to the directly motivated restriction of NNC: k-means is limited to Voronoi tessellations of the data space, and the solutions of spectral clustering are induced by some eigenvectors of the Laplacian. The minimizer of WSS that we aim for with k-means converges to the true global minimizer. K-means is restricted to local minima, but with the $r = 50$ restarts we provided in the experiments, the chances of getting a good partition are rather high, especially for two clusters, if there are not too many local minima. On the other hand, given its simplicity, NNC performs surprisingly well. Its results apparently improve, in comparison to SC or k-means, as the number K of clusters grows. As K grows, there might be more local minima that at least k-means can get stuck in.

The approach followed by NNC leaves several directions for further development: on the one hand, the algorithm can be optimized, for instance by the selection of seed nodes to increase the probability of a "good" neighborhood structure. On the other hand, the restriction of $\mathcal{F}_n$ could be done in a related, but more sophisticated way, such as by the continuity of candidate functions, for example measured by some Lipschitz constant. Another big issue for future work, also from a theoretical perspective, is the matching of the distance function with the objective.

Chapter 4

Conclusion

The initial goal of the studies presented in this report was to modify an NP-hard optimization problem, namely graph cuts for clustering, to gain in two respects: First, we aimed to simplify the problem to ideally make it solvable in polynomial time. Instead of using a heuristic without any guarantees on the solution, we took the viewpoint of Statistical Learning Theory to simultaneously achieve a second advantage: statistical consistency. That means we consider the data as a sample from a larger distribution, and try to optimize the criterion for the entire space. Consistency ensures that the quality of the partition returned by the algorithm will converge to the quality of the global optimizer on the entire space.

To achieve these aims, we took two approaches: First, we added a penalty term, a margin, to the discrete objective. Our margin considers robustness of the solution, though only locally. We stated the resulting problem as a flow problem. However, only the relaxed version is in P. In addition, some theoretical questions remain to be studied: To what kind of restriction of the function space does the margin correspond? Does it correspond to the complexity of the function space, and, if so, in what way? These are crucial questions with regard to proving consistency.

Second, we restricted the space of candidate functions to those constant on neighborhood cells. The resulting algorithm is statistically consistent and runs in polynomial time. Its performance on finite samples is comparable to that of standard algorithms such as spectral clustering or k-means. This approach shows that the initial aim is achievable even with a simple algorithm. Now it can be extended in several directions. The neighborhood criterion induces a certain continuity constraint. Analogously, more sophisticated restrictions of $\mathcal{F}$ are conceivable, for instance by continuity criteria such as Lipschitz constants. Furthermore, the NNC algorithm can be improved by a more elaborate selection of the seeds, which may give guarantees on the neighborhoods and thus $\mathcal{F}_n$. The construction of the neighborhood cells itself bears room for development, too. The matching of the quality Q and the distance function, for example, is also of theoretical interest.

Note, however, that neither approach guarantees the stability of the partition itself, because consistency is defined with respect to the quality function. The study of stability with respect to the partition functions f_n is another wide research topic.

Nevertheless, the approaches presented here are a first example of how to combine combinatorial optimization and SLT. We are keen to see more involved methods and approaches to follow.

Acknowledgments

I would like to thank my advisors, Ulrike von Luxburg and Michael Kaufmann, for their constant advice, encouragement and patience, and particularly for their support even during a long period of illness. I am also thankful to my advisors as well as the WSI and MPI for making this cooperation possible.

Furthermore, I am grateful to Frank Jäkel for providing the leaf confusion matrix and the corresponding image, and also for interesting discussions. Finally, thanks to Suvrit Sra for help with grammatical and language issues, as well as for support and encouragement.

Bibliography

Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows*. Prentice Hall, 1993.

D. J. Aldous and J. A. Fill. *Reversible Markov Chains and Random Walks on Graphs*. 2001. in preparation, available online: `http://www.stat.berkeley.edu/~aldous/RWG/book.html`.

Alexandre Arenas. Network data sets. `http://www.etse.urv.es/~aarenas/data/welcome.htm`.

Albert-László Barabási. Resources: Network databases. `http://www.nd.edu/~networks/resources.htm`.

Peter Bartlett and John Shawe-Taylor. *Advances in Kernel Methods – Support Vector Learning*, chapter Generalization Performance of Support Vector Machines and Other Pattern Classifiers. MIT Press, Cambridge, USA, 1998.

Shai Ben-David, Ulrike von Luxburg, and Dávid Pál. A sober look at clustering stability. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006)*, pages 5–19, Berlin, 2006. Springer.

Shai Ben-David, Dávid Pál, and Hans U. Simon. Stability of k-means clustering. In Nader H. Bshouty and Claudio Gentile, editors, *20th Annual Conference on Learning Theory (COLT 2007)*, number 4539 in Lecture Notes in Computer Science, pages 20–34, San Diego, CA, USA, 2007. Springer.

Kristin P. Bennett and Erin J. Bredensteiner. Duality and geometry in SVM classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 57–64. Morgan Kaufmann, 2000.

Yonathan Bilu and Nathan Linial. Are stable instances easy? Manuscript, November 2004.

Béla Bollobás. *Modern Graph Theory*. Number 184 in Graduate Texts in Mathematics. Springer, 1998.

Olivier Bousquet and André Elisseeff. Stability and generalization. *JMLR*, 2(3):499–526, 2002.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. *Advanced Lectures on Machine Learning – MLSS Summer Schools 2003*, chapter Introduction to Statistical Learning Theory. Number 3176 in LNAI. Springer, 2004.

Steven Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Michael J. Brusco and Stephanie Stahl. *Branch-and-Bound Applications in Combinatorial Data Analysis*. Statistics and Computing. Springer, 2005.

Sébastien Bubeck and Ulrike von Luxburg. Overfitting of clustering and how to avoid it. Preprint, 2007.

T. N. Bui and C. Jones. Finding good approximate vertex and edge partitions is NP-hard. *Inf. Process. Lett.*, 42(3):153–159, 1992.

J. V. Burke and M. C. Ferris. Weak sharp minima in mathematical programming. *SIAM Journal on Control and Optimization*, 31(5):1340–1359, 1993.

Gareth Butland, José Manuel Peregrín-Alvarez, Joyce Li, Wehong Yang, Xiaochun Yang, Veronica Canadien, Andrei Starostine, Dawn Richards, Bryan Beattie, Nevan Krogan, Michael Davey, John Parkinson, Jack Greenblatt, and Andrew Emili. Interaction network containing conserved and essential protein complexes in escherichia coli. *Nature*, 433:531–537, February 2005.

Andrea Caponnetto and Alexander Rakhlin. Stability properties of empirical risk minimization over Donsker classes. *The Journal of Machine Learning Research*, 7:2565 – 2583, Dec 2006.

Ashok K. Chandra, Prabhakar Raghavan, Walter L. Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover time. *Computational Complexity*, 6(4):312–340, December 1996.

Fan R. K. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1994.

William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. Series in Discrete Mathematics and Optimization. Wiley-Interscience, 1997.

cosin. COevolution and Self-organisation In dynamical Networks (COSIN). `http://151.100.123.37/data.html`.

cplex. Cplex solver. `http://www.ilog.com/products/cplex/`.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

Nello Cristianini, Colin Campbell, and John Shawe-Taylor. Dynamically adapting kernels in support vector machines. In *Proceedings of the 1998 conference on Advances in neural information processing systems II (NIPS)*, pages 204–210, Cambridge, MA, USA, 1999. MIT Press.

Database of Interacting Proteins. `http://dip.doe-mbi.ucla.edu/`.

Ricky Der and Daniel Lee. Large-margin classification in banach spaces. In Marina Meila and Xiaotong Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, San Juan, Puerto Rico, 2007.

Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition.* Number 31 in Applications of Mathematics – Stochastic Modelling and Applied Probability. Springer, 1996.

Inderjit S. Dhillon. *A New $O(N^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem.* PhD thesis, University of California, Berkeley, 1997.

C. Ding, X. He, H. Zha, M. Gu, and H. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the first IEEE International Conference on Data Mining (ICDM)*, pages 107–114, Washington, D.C., 2001.

M.B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Custer analysis and display of genome-wide expression patters. *Proceedings of the National Academy of Sciences, Genetics*, 95:14863–14868, Dec 1998.

P. Elias, A. Feinstein, and C. E. Shannon. Note on maximum flow through a network. *IRE Transactions on Information Theory IT-2*, pages 117–119, 1956.

L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian J. Math.*, 8: 399–404, 1956.

J. Fritz. Distribution-free exponential error bound for nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 21(5):552–557, 1975.

U. Grenander. *Abstract Inference.* Wiley, New York, 1981.

Stephen Guattery and Gary L. Miller. On the performance of the spectral graph partitioning methods. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms (SODA95)*, pages 233–242. ACM-SIAM, 1995.

R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 68(065103(R)), 2003.

I. Gutman and W. Xiao. Generalized inverse of the Laplacian matrix and some applications. *Bulletin, Classe des Sciences Mathématiques et Naturelles, Sciences mathématiques, Académie serbe des sciences et des arts, Beograd*, CXXIX(29):15–23, 2004.

L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 11(9):1074–1085, 1992.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, 2001.

Matthias Hein, Olivier Bousquet, and Bernhard Schölkopf. Maximal margin classification for metric spaces. *Journal of Computer and System Sciences*, 71(3):333–359, 10 2005.

Don Hush and Clint Shovel. On the VC dimension of bounded margin classifiers. *Machine Learning*, 45:33–45, 2001.

Hawoon Jeong, Sean P. Mason, Albert-László Barabási, and Zoltán N. Oltvai. Lethality and centrality in protein networks. *Nature*, (411):41–42, 2001.

Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

Michael Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Computational Learning Theory*, pages 152–162, 1997.

L. G. Khachiyan. A polynomial time algorithm in linear programming. *Doklady Akademiya Nauk SSSR*, (244):1093–1069, 1979.

D. J. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12: 81–95, 1993.

G. R. G. Lanckriet, M. Deng, N. Christianini, M. I. Jordan, and W. S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, Big Island, HI, 2004.

Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. The importance of convexity in learning with squared loss. In *Proceedings of the ninth annual conference on Computational learning theory (COLT)*, pages 140–146, Desenzano del Garda, Italy, 1996. ACM.

László Lovász. Random walks on graphs: A survey. *Combinatorics: Paul Erdös is Eighty*, 2:1–46, 1993.

Hongwu Ma and An-Ping Zheng. The connectivity structure, giant strong component and centrality of metabolic networks. *Bioinformatics*, 19(11):1423–1430, July 2003.

NLANR Measurement and Operations Analysis Team. Global ISP interconnectivity by AS number. `http://moat.nlanr.net/AS/`.

M. Meila and J. Shi. A random walks view of spectral segmentation. In *8th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2001.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Proc. NNSP*, pages 41–48, 1999.

Glenn W. Milligan. An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, 45(3):325–342, 1980.

Mark E. J. Newman. Network data. `http://www-personal.umich.edu/~mejn/netdata/`.

Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E 74*, (036104), 2006.

Robert Nowak. Statistical decision and learning theory. `www.ece.wisc.edu/~nowak/ece901/lecture1.pdf`, 2007a. Lecture Notes, Lecture 0.

Robert Nowak. Statistical decision and learning theory. `www.ece.wisc.edu/~nowak/SLT07.html`, 2007b. Lecture Notes.

Victor Y. Pan and Zhao Q. Chen. The complexity of the matrix eigenproblem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing (STOC)*, pages 507–516, Atlanta, Georgia, United States, 1999. ACM.

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

K. Pelckmans, J. Shawe-Taylor, J.A.K Suykens, and B. de Moor. Margin based transductive graph cuts using linear programming. In *AISTATS*, 2007.

J. M. Peña, J. A. Lozano, and P. Larrañaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.

D. Pollard. Strong consistency of k-means clustering. *Annals of Statistics*, 9(1):135–140, 1981.

Ali Rahimi and Ben Recht. Clustering with normalized cuts is clustering with a hyperplane. *Statistical Learning in Computer Vision*, 2004.

Alexander Rakhlin and Andrea Caponnetto. Stability of k-means clustering. In B. Schölkopf, J. Platt, and T. Hoffmann, editors, *Neural Information Processing Systems (NIPS)*, volume 19, pages 1121–1128. MIT Press, 2006.

Alexander Rakhlin, Sayan Mukherjee, and Tomaso Poggio. Stability results in learning theory. *Analysis and Applications*, 3(4):397–417, 2005.

F. Rao and A. Caflisch. The protein folding network. *Journal of Molecular Biology*, 342: 299–306, 2004.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42 (3):287–320, 2001.

Gunnar Rätsch. Benchmark repository used in Rätsch et al. [2001] and Mika et al. [1999]. `http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm`.

Ronitt Rubinfeld. Lecture notes for "Randomness and Computation" – lecture 13. people.csail.mit.edu/ronitt/COURSE/S06/l13.pdf, 2006. Scribe: Punyashloka Biswal.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2002.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell-cycle regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, Dec 1998.

Daniel A. Spielman and Shang-Hua Teng. Spectral partitioning works: planar graphs and finite element meshes. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS '96)*, pages 96–105, 1996.

Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44 (4):585–591, July 1997.

K. Tsuda, H. Shin, and B. Schölkopf. Fast protein classification with multiple networks. *Bioinformatics*, 21 (Suppl. 2):ii59–ii65, Sept. 2005. Data at `http://www.kyb.tuebingen.mpg.de/bs/people/tsuda/eccb05.html`.

uci. UCI machine learning repository. `www.ics.uci.edu/~mlearn/MLRepository.html`.

Hamed Valizadegan and Rong Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *Neural Information Processing Systems (NIPS)*, Vancouver, Canada, 2006.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer, 2001.

Ulrike von Luxburg. *Statistical Learning with Similarity and Dissimilarity Functions*. PhD thesis, Max Planck Institute for Biological Cybernetics, 2001.

Ulrike von Luxburg. A tutorial on spectral clustering. Technical Report 149, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, Aug 2006.

Ulrike von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *Annals of Statistics*. to appear.

Ulrike von Luxburg, Sébastien Bubeck, Stefanie Jegelka, and Michael Kaufmann. Consistent minimization of clustering objective functions. In *Advances in Neural Information Processing (NIPS)*, 2007a. accepted.

Ulrike von Luxburg, Sébastien Bubeck, Stefanie Jegelka, and Michael Kaufmann. Supplementary material to "Consistent minimization of clustering objective functions". `http://www.tuebingen.mpg.de/~ule`, 2007b.

Dorothea Wagner and Frank Wagner. Between min cut and graph bisection. In *Proceedings of the 18th International Symposium on the Mathematical Foundations of Computer Science (MFCS)*, pages 744–750. Springer, 1993.

D. J. Watts and S. H. Strogatz. Collective dynamics of 'small world' networks. *Nature*, 393: 440–442, 1998.

wikipedia. Wikipedia – the free encyclopedia. `www.wikipedia.org`.

W. Xiao and I. Gutman. Resistance distance and Laplacian spectrum. *Theoretical Chemistry Accounts*, 110:284–289, 2003.

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *Neural Information Processing Systems (NIPS)*, pages 1537–1544, 2004.

Dengyong Zhou, Baihua Xiao, Huibin Zhou, and Ruwei Dai. Global geometry of SVM classifiers. Technical Report 30-5-02, AI Lab, Institute of Automation, Chinese Academy of Sciences, 2002.

Appendix A

Notation and Abbreviations

$V \subseteq \mathcal{X}$	set of nodes
E	set of edges
$\mathcal{C}_i \subseteq V$	i-th cluster
$\overline{\mathcal{C}_i} = V \setminus \mathcal{C}_i$	set of nodes that are not in the i-th cluster
K	number of clusters
$w : E \to \mathbb{R}$, $w \in \mathcal{W}$	edge weights
$s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$	similarity function
$f : V \to \{0, 1\}$, $f \in \mathcal{F}$	partition function
P	complexity class of problems that can be solved in polynomial time (on a deterministic Turing Machine)
NP	complexity class (non-deterministic polynomial time) of decision problems that can be solved in polynomial time on a non-deterministic Turing Machine
SC	Spectral clustering
NNC	Nearest Neighbor Clustering

We use a generalized notation for sums of edge weights: let $A \subseteq V$ and $v \in V$, then

$$w(v, A) = \sum_{u \in A} w(v, u),$$

and analogously for $w(A, v)$ and $w(A, B)$ with $B \subseteq V$. We also use the notation $\text{cut}(A, B) = w(A, B)$.

Graph properties

$\text{vol}(G) = \text{vol}(V) = w(V, V)$	volume of the graph
$D \in \mathbb{R}^{n \times n}$	diagonal matrix of node degrees with $D(i, i) = d(X_i)$
$W \in \mathbb{R}^{n \times n}$	symmetric matrix of edge weights
$L = D - W$	graph Laplacian (see von Luxburg [2006] for properties of Laplacians)
$L_{rw} = I - D^{-1}W \in \mathbb{R}^{n \times n}$	normalized graph Laplacian
$L_{sym} = I - D^{-1/2}WD^{-1/2}$	symmetric normalized Laplacian

Learning Theory

SLT	Statistical Learning Theory
$\mathcal{F}$	class of candidate functions/predictors from which we choose f
$\mathcal{F}_n$	restriction of $\mathcal{F}$
$R : \mathcal{F} \to \mathbb{R}$	true risk
$\widehat{R} : \mathcal{F} \times \mathcal{X}^n \to \mathbb{R}$	empirical risk
$Q : \mathcal{F} \to \mathbb{R}$	quality functional
$Q_n : \mathcal{F} \times \mathcal{X}^n \to \mathbb{R}$	empirical estimate of Q
$f^* = \operatorname{argmin}_{f \in \mathcal{F}} Q(f)$	true global optimizer in $\mathcal{F}$
$f_n = \operatorname{argmin}_{f \in \mathcal{F}_n} Q_n(f)$	optimizer of Q_n from $\mathcal{F}_n$
ρ	Margin

Distance and Quality functions

Ncut	Normalized Cut
Rcut	Ratio Cut
Mincut	Minimum Cut
WSS	Within-sum-of-squares
BW	Between-Within cluster similarity
ED	Euclidean distance
CD	Commute distance
HT	Hitting time
ND	Normalized commute distance
SND	Symmetric normalized commute distance

Appendix B

Correction

	SC	NNC
breast-cancer	0.11 ± 0.02 0.22 ± 0.07	0.10 ± 0.02 0.20 ± 0.07
diabetis	0.03 ± 0.02 0.04 ± 0.03	0.03 ± 0.02 0.04 ± 0.04
german	0.02 ± 0.02 0.04 ± 0.08	0.02 ± 0.02 0.03 ± 0.03
heart	0.18 ± 0.03 0.28 ± 0.03	0.17 ± 0.02 0.28 ± 0.04
splice	0.36 ± 0.10 0.58 ± 0.09	0.43 ± 0.16 0.66 ± 0.17
bcw	0.02 ± 0.01 0.04 ± 0.01	0.02 ± 0.01 0.04 ± 0.03
ionosphere	0.06 ± 0.02 0.12 ± 0.11	0.04 ± 0.01 0.12 ± 0.11
pima	0.03 ± 0.03 0.05 ± 0.04	0.03 ± 0.03 0.04 ± 0.03
cellcycle	0.12 ± 0.02 0.16 ± 0.02	0.10 ± 0.01 0.15 ± 0.02

Table B.1: Correction for Table 3.3. The values are very similar, only that the generalization performance of NNC is actually better than in the old table.