

Introduction to Machine Learning, Spring 2016

Problem Set 3: Kernel methods

Due: Wednesday, February 24, 2016 at 10pm (upload to NYU Classes.)

Important: See problem set policy on the course web site.

Instructions. You may use the programming language of your choice (we strongly recommend using Python). However, you are **not** permitted to use or reference any machine learning code or packages not written by yourself, except for question 2(c)-(f). Your answers to the below, plots, and all code that you write for this assignment should be uploaded to NYU Classes.

1. Kernels

- (a) For any two documents x and z , define $k(x, z)$ to equal the number of unique words that occur in both x and z (i.e., the size of the intersection of the sets of words in the two documents). Is this function a kernel? Justify your answer. (Hint: $k(x, z)$ is a kernel if there exists $\phi(x)$ such that $k(x, z) = \phi(x)^T \phi(z)$).
- (b) Assuming that $\vec{x} = [x_1, x_2]$, $\vec{z} = [z_1, z_2]$ (i.e., both vectors are two-dimensional) and $\beta > 0$, show that the following is a kernel:

$$k_\beta(\vec{x}, \vec{z}) = (1 + \beta \vec{x} \cdot \vec{z})^2 - 1$$

Do so by demonstrating a feature mapping $\Phi(\vec{x})$ such that $k_\beta(\vec{x}, \vec{z}) = \Phi(\vec{x}) \cdot \Phi(\vec{z})$.

- (c) One way to construct kernels is to build them from simpler ones. Assuming $k_1(x, z)$ and $k_2(x, z)$ are kernels, then one can show that so are these:
- i. (scaling) $f(x)f(z)k_1(x, z)$ for any function $f(x) \in \mathcal{R}$,
 - ii. (sum) $k(x, z) = k_1(x, z) + k_2(x, z)$,
 - iii. (product) $k(x, z) = k_1(x, z)k_2(x, z)$.

Using the above rules and the fact that $k(x, z) = x^T z$ is a kernel, show that the following is also a kernel:

$$\left(1 + \left(\frac{x}{\|x\|_2} \right)^T \left(\frac{z}{\|z\|_2} \right) \right)^3.$$

2. The MNIST dataset is a database of handwritten digits. This problem will apply SVMs to automatically classify digits; the US postal service uses a similar optical character recognition (OCR) of zip codes to automatically route letters to their destination. The original dataset can be downloaded at <http://yann.lecun.com/exdb/mnist/>. For this problem, we randomly chose a subset of the original dataset. We have provided you with two data files, `mnist_train.txt`, `mnist_test.txt`. The training set contains 2000 digits, and the test set contains 1000 digits. Each line represents an image of size 28×28 by a vector of length 784, with each feature specifying a grayscale pixel value. The first column contains the labels of the digits, 0–9, the next 28 columns represent the first row of the image, and so on. We also provide two scripts written in MATLAB/Octave and Python, `show_img.m`, `show_img.py` to show a single image; using these will help you have a better understanding of what the data looks like and how it is represented.

Your linear classifier will obtain less than 15% test error, and using a Gaussian kernel you will obtain less than 7% test error! Had you used more training data, SVM with Gaussian kernel can get down to 1.4% test error (degree 4 polynomial obtains 1.1% test error). With further fine-tuning (e.g., augmenting the training set by adding deformed versions of the existing training images), a SVM-based approach can obtain 0.56% test error [2]. The state-of-the-art, which uses a convolutional neural network, obtains 0.23% test error [1].

- (a) Read in `mnist_train.txt`, `mnist_test.txt` and transform them into feature vectors. Normalize the feature vectors so that each feature is in the range $[-1, 1]$. Since in this dataset each feature has minimum value 0 and maximum value 255, you can do this normalization by transforming each column \vec{v} to $2\vec{v}/255 - 1$. The normalization step can be crucial when you incorporate higher-order features. It also helps prevent numerical difficulties while solving the SVM optimization problem.
- (b) Implement multi-class prediction using one-versus-all classification. Train 10 binary classifiers using the Pegasos algorithm from the previous question. For each classifier, you relabel one of the labels to 1, and the other 9 labels to -1. Following learning, you will have 10 distinct weight vectors. To predict the label of an example x , compute the dot product of x with each weight vector, giving you 10 scores, and predict the label with the maximum score.
- (c) Instead of holding out a specific portion of your training data as a validation set, there is another approach to estimate the test error called k -fold cross-validation. Cross-validation is particularly useful when you have a small amount of training data. Cross-validation divides the training data into k parts of equal size. Then, for $i = 1, \dots, k$, we fit a model using all of the data except for the k 'th part, and use the remaining part to compute the validation error. Finally, we report the averaged validation error. Use Scikit-learn's k -fold cross-validation¹ with $k = 5$, and find a model having the smallest cross-validation error from $\lambda = 2^{-5}, 2^{-4}, \dots, 2^1$. Plot the cross-validation error vs. λ . What is the best λ ? For this value of λ , re-train the classifier now using *all* of the training data. What is the test error?
- (d) We now explore the use of non-linear kernels within Support Vector Machines. We will make use of a Scikit-Learn's SVM classifier (SVC)², which provides a nice interface to a powerful library called libSVM. This library implements the SMO algorithm, which is an alternative optimization algorithm for the SVM objective with kernels from what we saw in class [3, 4]. However, SVC by default trains a one-vs-one classifier, so use Scikit-Learn's OneVsRestClassifier³ to train your classifier. Try the default setting of the SVC, which uses the Gaussian kernel with $\gamma = 1/\text{num_features}$, and $C = 1$. Make sure that each feature is scaled to $[-1, 1]$ as in problem (b) which could also be done by using Scikit-Learn's MinMaxScaler. Note that in the library, the Gaussian kernel is of the form $K(\vec{u}, \vec{v}) = e^{-\gamma\|\vec{u}-\vec{v}\|^2}$ (equivalent to what we showed in class

¹http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.cross_val_score.html#sklearn.cross_validation.cross_val_score

²<http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

³<http://scikit-learn.org/stable/modules/multiclass.html#one-vs-the-rest>

when $\gamma = 1/2\sigma^2$) and the optimization problem is of the form

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{j=1}^m \xi_j \\ \text{subject to} \quad & y_j(\mathbf{w} \cdot \mathbf{x}_j + b) \geq 1 - \xi_j, \\ & \xi_j \geq 0. \end{aligned}$$

This can be seen to be equivalent to the SVM optimization problem solved by Pegasos when $C = 1/m\lambda$. Train on the full training set. What is the test error?

- (e) Rather than using the default settings, we can choose the two parameters to be tuned (C and γ) using cross-validation. Report the 10-fold cross-validation error when γ and C are at their default settings.
- (f) Finally, try different γ and C values to find a model with small cross-validation error. What were the best values that you found? What is the cross-validation error? What is the test error for this setting?

References

- [1] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.
- [2] Dennis Decoste and Bernhard Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1-3):161–190, 2002.
- [3] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.
- [4] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in kernel methods*, pages 185–208. MIT Press, 1999.