6.867 Machine Learning

Problem Set 3

Due date: Wednesday October 20, 2004

Please address all questions and comments about this problem set to `6867-staff@csail.mit.edu`. You will need to use MATLAB for some of the problems but most of the code is provided. If you are not familiar with MATLAB, please consult

`http://www.ai.mit.edu/courses/6.867/matlab.html`

and the links therein.

Problem 1: Kernels, features, and maximum margin

1. *(5 points)* We have discussed two different definitions of kernels in class:

   - Definition 1: $K(\mathbf{x}, \mathbf{x}')$ is a kernel if it can be written as an inner product $\phi(\mathbf{x})^T \phi(\mathbf{x}')$ for some feature mapping $\mathbf{x} \rightarrow \phi(\mathbf{x})$.
   - Definition 2: $K(\mathbf{x}, \mathbf{x}')$ is a kernel if for any finite set of training examples, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the $n \times n$ matrix $\mathbf{K}$ such that $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is positive semi-definite.

   Show that Definition 1 implies Definition 2. *Hint:* you could show this by proving that for any real numbers $\alpha_1, \ldots, \alpha_n$ and points $\mathbf{x}_1, \ldots, \mathbf{x}_n$,

   $$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

   if the kernel can be written as $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$.

   The converse holds as well (the definitions are equivalent) but the proof is a bit more involved. Moreover, any symmetric positive semi-definite $n \times n$ matrix $\mathbf{K}$ can be interpreted as coming from some kernel $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ evaluated on some n points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. There are many possible kernels (and sets of points) that would give rise to the same $n \times n$ matrix, however. The knowledge that at least one such kernel exists is sometimes useful when trying to optimize a kernel for a particular set of points (we can cast the optimization directly in terms of symmetric positive semi-definite matrices without worrying about feature maps).

2. One way to construct kernels is to build them from simpler ones. We have already seen three possible "construction rules": assuming $K_1(\mathbf{x}, \mathbf{x}')$ and $K_2(\mathbf{x}, \mathbf{x}')$ are kernels then so are

   - (scaling) $f(\mathbf{x})K_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$, $f(\mathbf{x}) \in \mathcal{R}$
   - (sum) $K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$
   - (product) $K_1(\mathbf{x}, \mathbf{x}')K_2(\mathbf{x}, \mathbf{x}')$

   (a) *(5 points)* Let $\phi^{(1)}(\mathbf{x})$ and $\phi^{(2)}(\mathbf{x})$ be the feature vectors corresponding to kernels $K_1(\mathbf{x}, \mathbf{x}')$ and $K_2(\mathbf{x}, \mathbf{x}')$, respectively. These feature vectors may be of different lengths. Show that the product kernel $K_1(\mathbf{x}, \mathbf{x}')K_2(\mathbf{x}, \mathbf{x}')$ is a kernel by showing that its feature vectors are given explicitly by $\phi(\mathbf{x})$ whose (i, j) component (doubly indexed vector) is $\phi_i^{(1)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x})$.

   (b) *(10 points)* Use the construction rules to build a normalized cubic polynomial kernel

   $$K(\mathbf{x}, \mathbf{x}') = \left(1 + \left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right)^T \left(\frac{\mathbf{x}'}{\|\mathbf{x}'\|}\right)\right)^3 \tag{1}$$

   You can assume that you already have a constant kernel $K_0(\mathbf{x}, \mathbf{x}') = 1$ and a linear kernel $K_1(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$. Identify which rules you are employing at each step.

3. Let's now explore the effect of feature vectors on the maximum margin solution. Consider a simple one dimensional case where we have only two training examples $(x_1 = 0, y_1 = -1), (x_2 = \sqrt{2}, y_2 = 1)$, and we map each input to a feature vector $\phi(x) = [1 \ \sqrt{2}x \ x^2]^T$. In other words, we are effectively using a second order polynomial kernel. We'd like to find and understand the maximum margin solution $\hat{\mathbf{w}}_1 = [\hat{w}_1, \hat{w}_2, \hat{w}_3]^T$ and $\hat{w}_0$ to

   $$\begin{aligned} &\min \|\mathbf{w}_1\|^2 \text{ subject to} \\ &y_1[w_0 + \phi(x_1)^T\mathbf{w}_1] - 1 \geq 0 \\ &y_2[w_0 + \phi(x_2)^T\mathbf{w}_1] - 1 \geq 0 \end{aligned}$$

   Please return your derivations along with the specific answers.

   (a) *(5 points)* Using your knowledge of the maximum margin boundary write down a vector that points in the same direction as $\hat{\mathbf{w}}_1$

   (b) *(5 points)* What is the value of the margin that the we can achieve in this case?

   (c) *(5 points)* By relating the margin and $\hat{\mathbf{w}}_1$, provide the actual solution $\hat{\mathbf{w}}_1$. What is $\hat{w}_0$?

   (d) *(5 points)* Plot the resulting discriminant function in MATLAB. How are large negative x classified? Briefly explain why the discriminant function does not equal zero at the midpoint $x = \sqrt{2}/2$ between x_1 and x_2.

Problem2: Support Vector Machines

We have provided the following Matlab routines for constructing support vector machine classifiers:

`Klinear.m, Kpoly.m, Kgaussian.m` A linear, polynomial, and Gaussian kernel. Besides the two vectors, the kernel takes an extra parameter. In the case of the polynomial kernel, the extra parameter is the degree. In the case of the Gaussian, the extra parameter is the standard deviation. The linear kernel ignores this parameter. The name of the kernel must be prefixed by '`@`' when passed to `svm_train.m`.

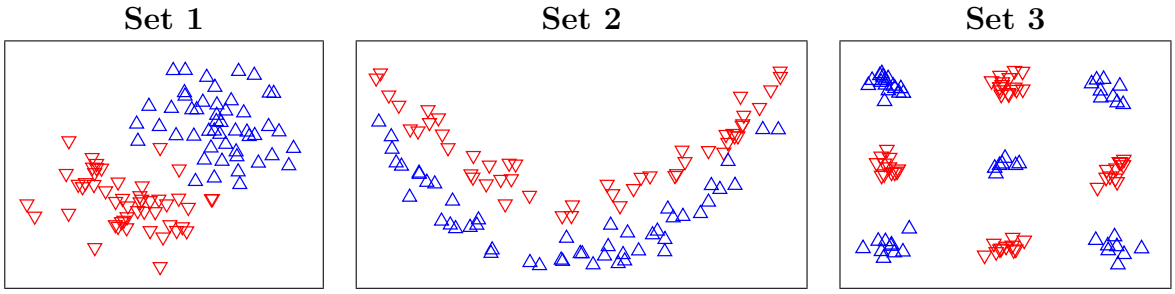
`svm_train.m` Trains a SVM given training data, a kernel, kernel parameter, and the C penalty on training errors.

`svm_discrim_func.m` Evaluates the SVM hyperplane on a set of test points. Data is classified according to the sign of this evaluation.

`svm_plot.m` Plots the SVM decision boundary and the supplied labeled datapoints.

`svm_test.m` Runs an SVM experiment by training the SVM on the supplied training data, and testing it on the supplied test data. Plots the SVM decision boundary, and the test errors.

We have also provided four train/test sets of data. The first three are artificially generated 2-dimensional data, while the last one is the 64-dimensional digit dataset from Problem Set 2. You can load all data with `load data_ps3_2.mat`. The 2-dimensional datasets are pictured below:



1. *(10 points)* For the first three datasets, consider the linear, second order polynomial, and Gaussian of standard deviation 1 kernels. For each dataset provide a rationale for which kernel should be the best for training a SVM classifier on that dataset. Choose the best kernel for each dataset and plot the decision boundary and test errors with `svm_test.m`. Hand in the three plots. (for consistency, everybody should use $C = 1000$).

2. *(10 points)* For the digit dataset (`set4_train` and `set4_test`), train and test SVM's with a linear, polynomial of degree 2, and Gaussian of standard deviation 1.5 kernels. Report the test errors. How does the performance of the SVM compare to the logistic regression classifier of Problem Set 2? (Note: you will have to use `svm_train.m` and `svm_discrim_func.m` directly, because `svm_test.m` is for 2D data only).

Problem 3: Non-Separable SVM's

Given a set of training points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ and associated class labels $y_1, y_2, \ldots, y_n \in \{-1, 1\}$, the linear support vector machine finds a linear separation $\mathbf{x}^T\mathbf{w}_1 + w_0 = 0$ between the training points in each class that maximizes the margin (the distance to the two training sets). New points are then classified according to the side of the separation on which they fall.

If the training points are not linearly separable, in the search for the optimal decision boundary we need to allow for some training points to be on the wrong side of the linear separation. One way to achieve this that you have seen during lecture is through the following optimization problem:

$$\begin{array}{ll} \min_{w_0, \mathbf{w}_1, \boldsymbol{\xi}} & \frac{1}{2}\|\mathbf{w}_1\|^2 + C\sum_{i=1}^n \xi_i \\ \text{s.t.} & y_i[w_0 + \mathbf{x}_i^T\mathbf{w}_1] \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{array}$$

The slack variables $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n)$ express the amount by which each point is violating the margin constraint ($\xi_i = 0$ when the constraint is satisfied), while the constant C represents the penalty for violated constraints.

The above formulation has the shortcoming that the actual value of C has no obvious interpretation. It is therefore difficult to select C a priori. Here we will derive an alternative formulation that addresses this issue. Specifically, we consider the following optimization problem (called $\nu-$SVM):

$$\begin{array}{ll} \min_{w_0, \mathbf{w}_1, \boldsymbol{\xi}, \rho} & \frac{1}{2}\|\mathbf{w}_1\|^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^n \xi_i \\ \text{s.t.} & y_i[w_0 + \mathbf{x}_i^T\mathbf{w}_1] \geq \rho - \xi_i \\ & \xi_i \geq 0 \\ & \rho \geq 0 \end{array}$$

The role of C here is replaced by a scalar variable ρ and a positive constant ν which we can interpret more clearly. The number of support vectors turns out to be roughly $n\nu$, thus ν selects the expected number of training errors. ρ, on the other hand, is set by the optimization problem.

To derive the dual optimization problem in this case we associate Lagrange multipliers $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$, and γ with each of the constraints (these will

be the new optimization variables in the dual problem), and construct the unconstrained objective:

$$L(w_0,\mathbf{w}_1,\boldsymbol{\xi},\rho;\boldsymbol{\alpha},\boldsymbol{\beta},\gamma) = \frac{1}{2}\|\mathbf{w}_1\|^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^{n}\xi_i + \\ + \sum_{i=1}^{n}\alpha_i\left[\rho - \xi_i - y_i[w_0 + \mathbf{x}_i^T\mathbf{w}_1]\right] - \sum_{i=1}^{n}\beta_i\xi_i - \gamma\rho$$

The problem we wish to solve is

$$\min_{w_0,\mathbf{w}_1,\boldsymbol{\xi},\rho}\ \max_{\substack{\boldsymbol{\alpha}\geq\mathbf{0}\\ \boldsymbol{\beta}\geq\mathbf{0}\\ \gamma\geq 0}} L(w_0,\mathbf{w}_1,\boldsymbol{\xi},\rho;\boldsymbol{\alpha},\boldsymbol{\beta},\gamma)$$

By switching the order of min and max (which can be shown to be equivalent in this case) we will instead solve:

$$\max_{\substack{\boldsymbol{\alpha}\geq\mathbf{0}\\ \boldsymbol{\beta}\geq\mathbf{0}\\ \gamma\geq 0}}\ \min_{w_0,\mathbf{w}_1,\boldsymbol{\xi},\rho} L(w_0,\mathbf{w}_1,\boldsymbol{\xi},\rho;\boldsymbol{\alpha},\boldsymbol{\beta},\gamma)$$

1. *(10 points)* One can rewrite L in the following form:

$$L(w_0,\mathbf{w}_1,\boldsymbol{\xi},\rho;\boldsymbol{\alpha},\boldsymbol{\beta},\gamma) = -\left(\sum_{i=1}^{n}\alpha_i y_i\right)w_0 + \left[\frac{1}{2}\|\mathbf{w}_1\|^2 - \left(\sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i^T\right)\mathbf{w}_1\right] + \\ + \sum_{i=1}^{n}\left(\frac{1}{n} - \alpha_i - \beta_i\right)\xi_i + \left(\sum_{i=1}^{n}\alpha_i - \gamma - \nu\right)\rho$$

   What constraints $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and γ must satisfy so that the minimum

$$J(\boldsymbol{\alpha},\boldsymbol{\beta},\gamma) = \min_{w_0,\mathbf{w}_1,\boldsymbol{\xi},\rho} L(w_0,\mathbf{w}_1,\boldsymbol{\xi},\rho;\boldsymbol{\alpha},\boldsymbol{\beta},\gamma)$$

   is finite? Provide an expression for $J(\boldsymbol{\alpha},\boldsymbol{\beta},\gamma)$ within those constraints.

   When actually solving the dual problem, maximizing $J(\boldsymbol{\alpha},\boldsymbol{\beta},\gamma)$, we can further remove $\boldsymbol{\beta}$ and γ from the optimization problem since they merely impose the following constraints on $\boldsymbol{\alpha}$:

$$\alpha_i \leq \frac{1}{n} \text{ and } \sum_{i=1}^{n}\alpha_i \geq \nu \tag{2}$$

   in addition to the typical constraints $\alpha_i \geq 0$ and $\sum_{i=1}^{n}\alpha_i y_i = 0$. Try to understand where the new constraints are coming from.

2. *(10 points)* If as a result of optimizing α_i (and β_i) they both turn out to be > 0 for some training point $\mathbf{x}_i$, then their associated constraints must hold with equality, that is $\xi_i = 0$ and $y_i[w_0 + \mathbf{x}_i^T \mathbf{w}_1] = \rho$. Let $(\mathbf{x}'_1, y'_1), (\mathbf{x}'_2, y'_2), \ldots, (\mathbf{x}'_m, y'_m)$ be the subset of the training points for which this is true. Use this property to derive a formula for ρ and w_0. Try to make the estimate of ρ and w_0 as robust as possible by using the entire subset of training points in the formulas.

3. *(10 points)* In this part of the problem you are asked to fill in the Matlab code for the provided function `nusvm_train.m` that should optimize ν-SVM. Like in the function `svm_train.m` that optimizes the simpler formulation involving C, the generic Matlab quadratic programming routine `quadprog` performs the actual optimization. The provided function needs only the parameters to `quadprog`, that is the definition of the optimization problem. To understand the type of optimization solved by `quadprog` read its documentation (`doc quadprog` at the Matlab prompt). The calculation of w_0 is already provided in the function. Turn in a printout of the modified `nusvm_train.m`.

   One you filled in the parameters to `quadprog`, you can use `nusvm_test.m` to run experiments with the ν-SVM. Train the ν-SVM on `set2_train` with a Gaussian kernel of standard deviation 1 and the following values for ν: 0.01, 0.1, 0.3, 0.5, and 0.7. Report the number of training errors, as well as the test error on `set2_test`. Does the parameter ν have the expected effect?