6.867 Machine Learning

Problem Set 3 Solutions

Due date: Wednesday October 20

Problem 1: Kernels, features and maximum margin

1. **Solution:** For any α,

$$\begin{aligned}
\alpha^T \mathbf{K} \alpha &= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}_j) \\
&= (\sum_{i=1}^{n} \alpha_i \mathbf{\Phi}(\mathbf{x}_i))^T (\sum_{j=1}^{n} \mathbf{\Phi}(\mathbf{x}_j)) \\
&= (\sum_{i=1}^{n} \alpha_i \mathbf{\Phi}(\mathbf{x}_i))^2 \\
&\geq 0
\end{aligned}$$

Hence, $\mathbf{K}$ is a positive semi-definite matrix. ■

2. **Solution:**

(a) For a pair of input points $\mathbf{x}$ and $\mathbf{x}'$, and two possible feature vectors $\Phi^{(1)}(\cdot)$ and $\Phi^{(2)}(\cdot)$ for each point, of lengths n_1 and n_2 respectively, the product kernel value

(which is a scalar) is given by:

$$\begin{aligned}
K(\mathbf{x},\mathbf{x}') &= K_1(\mathbf{x},\mathbf{x}') \times K_2(\mathbf{x},\mathbf{x}') && (1)\\
&= (\Phi^{(1)}(\mathbf{x})^T\Phi^{(1)}(\mathbf{x}')) \times (\Phi^{(2)}(\mathbf{x})^T\Phi^{(2)}(\mathbf{x}')) && (2)\\
&= (\sum_{i=1}^{n_1}\Phi_i^{(1)}(\mathbf{x})\Phi_i^{(1)}(\mathbf{x}')) \times (\sum_{j=1}^{n_2}\Phi_j^{(2)}(\mathbf{x})\Phi_j^{(2)}(\mathbf{x}')) && (3)\\
&= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\Phi_i^{(1)}(\mathbf{x})\Phi_j^{(2)}(\mathbf{x})\Phi_i^{(1)}(\mathbf{x}')\Phi_j^{(2)}(\mathbf{x}') && (4)\\
&= \sum_{k=1}^{n_1\times n_2}\Phi_k(\mathbf{x})\Phi_k(\mathbf{x}') && (5)\\
&= \Phi(\mathbf{x})^T\Phi(\mathbf{x}'), && (6)
\end{aligned}$$

which is a valid inner product. Thus, the product kernel, $K(\mathbf{x},\mathbf{x}')$, is a valid kernel.

(b) The rules we apply at different steps are as follows:
Step 1: Scaling. (Using $f(\mathbf{x}) = (1/||\mathbf{x}||)$, we get a new kernel $K_3(\mathbf{x},\mathbf{x}')$)
Step 2: Sum. ($K_4(\mathbf{x},\mathbf{x}') = 1 + K_3(\mathbf{x},\mathbf{x}')$)
Step 3: Product, twice. (to obtain $K_4(\mathbf{x},\mathbf{x}')^3$)

■

3. **Solution:** Input vectors $x_1 = 0$ and $x_2 = 1$ are mapped to feature vectors $\Phi(x_1) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ and $\Phi(x_2) = \begin{bmatrix} 1 & 2 & 2 \end{bmatrix}^T$.

(a) For a training set consisting of one positive and one negative example, the direction of the weight vector $\hat{\mathbf{w}}_1$ is the same as the direction of a vector from the negative example to the positive example, in the feature space. Thus, direction of $\hat{\mathbf{w}}_1 = [0 \quad 2 \quad 2]^T$

(b) Margin = distance from each support vector to decision boundary (in feature space) = distance from each training point to a point midway between the 2 points (since there are only 2 training examples) = $\sqrt{2}$.

(c) $||\hat{\mathbf{w}}_1|| = 1/(\text{margin})$. We have now specified both the norm and the direction of the weight vector. Therefore, we can calculate the weight vector as $\hat{\mathbf{w}}_1 = [0 \quad 1/2 \quad 1/2]^T$.

Constraints are satisfied as equalities at the support vectors (*i.e.*, the two training points). Thus,

$$\begin{aligned}
-1(w_0 + \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}\hat{\mathbf{w}}_1) = 1\\
+1(w_0 + \begin{bmatrix} 1 & 2 & 2 \end{bmatrix}\hat{\mathbf{w}}_1) = 1
\end{aligned}$$

Solving, $w_0 = -1$.

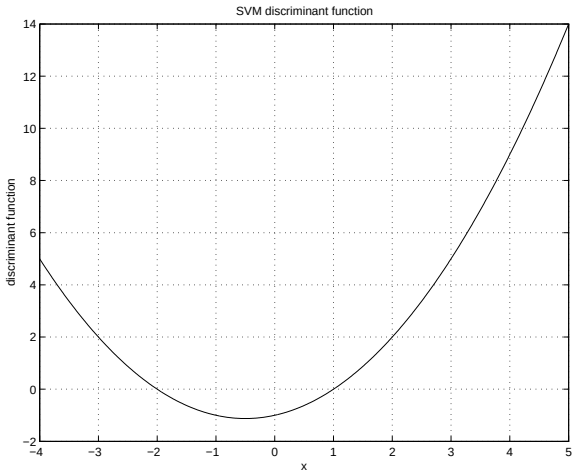


Figure 1: Plot for Problem 1.3 (d)

(d) The discriminant function, as a function of input x, is given by $w_0 + \hat{\mathbf{w}}_1^T \Phi(x) = x^2/2 + x/\sqrt{2} - 1$. See Figure 1.

For large negative inputs x, the SVM classifier output is positive.

The decision boundary is equidistant from the support vectors (*i.e.* the two training points) in feature space, not in input space. There is a non-linear mapping from input space to feature space. Thus, points equidistant from the training points in the input space need not lie on the decision boundary.

■

Problem 2: Support Vector Machines

1. **Solution:**

Set 1 The best kernel is the linear one. Rationale: samples seem to be drawn from two Gaussians of the same covariance, whose decision boundary is linear. It is always best to choose the simplest model to discourage overfitting. You can also choose by the error rate on the test set.

Set 2 The best kernel is the second order polynomial one. Rationale: the decision boundary between the two classes seems to be a parabola, a curve of degree 2. It is certainly not linear, and the Gaussian kernel overfits. You can also choose by the error rate on the test set.

Set 3 The best kernel is the Gaussian kernel. Rationale: the clusters are clearly not separable with curves of degree one or two. Since the Gaussian kernel always

separates points, it is the best choice here. Also, points are similar under the Gaussian kernel if they are close to each other in the original space. It is clear the small distance to the cluster center is the defining property of the classes. You can also choose by the error rate on the test set.

Plots are shown in Figure 2.

■

2. **Solution:**

linear	2^{nd} order	Gaussian
0.1375	0.12	0.085

All the three kernels perform better than logistic regression (which was 0.1425).

■

Problem 3: Non-Separable SVM's

1. **Solution:** Because L is linear in w_0, $\boldsymbol{\xi}$, and ρ, and w_0, $\boldsymbol{\xi}$, and ρ are unconstrained, the coefficients in front of these variables must be 0. Otherwise we could drive L to $-\infty$ by choosing appropriate large values for w_0, $\boldsymbol{\xi}$, or ρ. Thus $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, γ must satisfy the following conditions in order for J to be finite:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{7}$$

$$\alpha_i + \beta_i = \frac{1}{n} \quad \text{for every } i \in \{1, \ldots, n\} \tag{8}$$

$$\sum_{i=1}^{n} \alpha_i = \gamma + \nu \tag{9}$$

When these conditions are satisfied, L becomes:

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

To obtain J under these conditions, we must minimize the above formula for L with respect to $\mathbf{w}_1$ (the only variable left). Either by taking the derivative, or simply by seeing it as a quadratic function, the minimizing $\mathbf{w}_1$ is $\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$. At this $\mathbf{w}_1$, J is:

$$J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

For completeness, $J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma) = -\infty$ if any of the constraints (7), (8), (9) is not satisfied. It follows that the maximum of $J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma)$ with respect to $\boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma$ will not be achieved if any of (7), (8), (9) is not satisfied.

Since $\boldsymbol{\beta}$ does not appear in the criterion, the constraints $\alpha_i + \beta_i = \frac{1}{n},\ \beta_i \geq 0$ can be rewritten as $\frac{1}{n} - \alpha_i \geq 0$. Similarly $\sum_{i=1}^n \alpha_i = \gamma + \nu,\ \gamma \geq 0$ becomes $\sum_{i=1}^n \alpha_i - \nu \geq 0$.

■

2. **Solution:** For this part we assume $\boldsymbol{\alpha}$ is known as a result of solving the dual optimization, and we must express w_0 and ρ in terms of $V\alpha$ and the training data. $\mathbf{w}_1$ is already known from the derivation of the dual optimization: $\mathbf{w_1} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$

   Let i be a positive support vector ($y_i = 1$) and j be a negative one ($y_j = -1$). Then:

$$\begin{aligned} w_0 + \mathbf{x}_i^T \mathbf{w}_1 &= \rho \\ -w_0 - \mathbf{x}_j^T \mathbf{w}_1 &= \rho \end{aligned}$$

   Adding the two equations together we get

$$\rho = \frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{w}_1$$

   To get a robust estimate $\hat{\rho}$, we can consider all positive-negative pairs of SV's, and average the values we obtain from the above equation. An alternative is to take the median.

   Now given $\hat{\rho}$, we can solve for w_0 from and SV constraint:

$$w_0 = y_i \hat{\rho} - \mathbf{x}_i^T \mathbf{w}_1$$

   To get a robust estimate, we can average all the above equations for all SV's, or take the median.

   An alternative solution is to view the problem as solving an over-constrained system of linear equations with 2 unknowns. A robust solutions to such a system that minimizes the error between predicted and real values in the least square sense is given by the *pseudo*-inverse of a (non necessarily square) matrix.

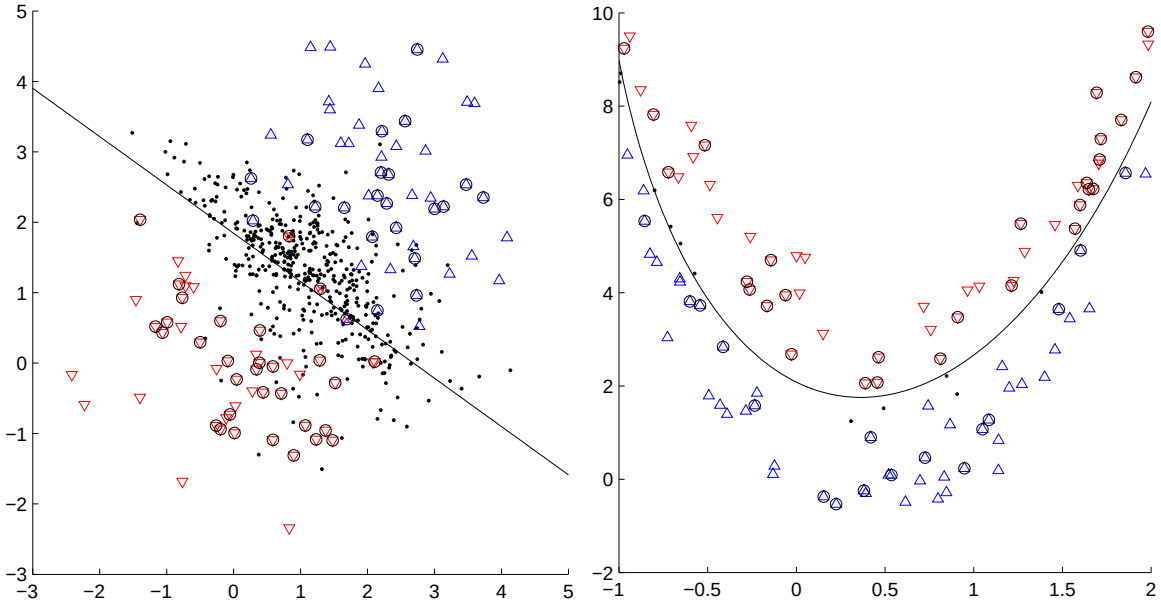
■

3. **Solution:** The code for solving the ν-SVM is shown in Figure 3.

ν	0.01	0.1	0.3	0.5	0.7
training error	0	0.01	0.05	0.07	0.18
test error	0.013	0.014	0.073	0.105	0.195

   ν has the expected effect in the sense the number of training errors roughly increases with ν. The fraction of training errors is not exactly ν because of the limited sample on which the SVM is trained (in the limit they should become equal).

■



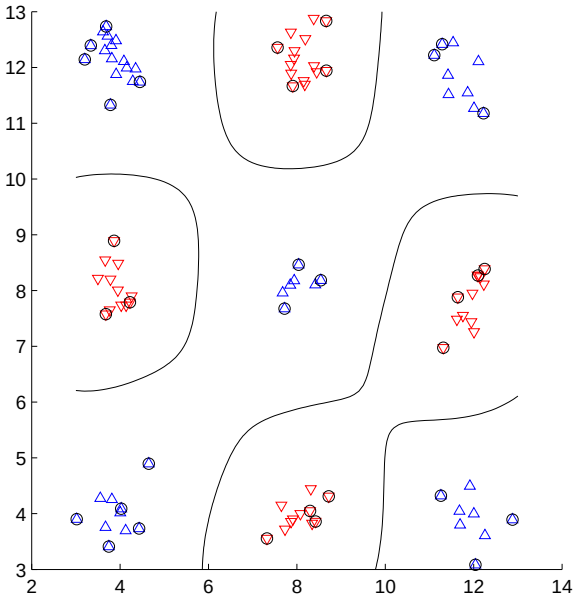


Figure 2: Plots for problem 3 part 1

```
function svm = nusvm_train(data, kernel, param, nu)

y = data.y;
X = data.X;
n = length(y);

% evaluate the kernel matrix
K = feval(kernel,X,X,param); % n x n positive semi-definite matrix
K = (K+K')/2; % should be symmetric. if not, may replace by equiv symm kernel.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For Part 4 of the problem, you must fill in the following section.
% Make sure you undestand the parameters to 'quadprog' (doc quadprog)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D = diag(y); % diagonal matrix with D(i,i) = y(i)
H = D*K*D; % H(i,j) = y(i)*K(i,j)*y(j)
f = zeros(n,1);
A = -ones(1,n);
b = -nu;
Aeq = y';
beq = 0.0;
LB = zeros(n,1);
UB = 1/n * ones(n,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X0 = zeros(n,1);

warning off; % suppress 'Warning: Large-scale method ...'
alpha = quadprog(H+1e-10*eye(n),f,A,b,Aeq,beq,LB,UB,X0)
warning on;

% essentially, we have added a (weak) regularization term to
% the dual problem favoring minimum-norm alpha when solution
% is underdetermined. this is also important numerically
% as any round-off error in computation of H could potentially
% cause dual problem to become ill-posed (minimizer at infinity).
% regularization term forces Hessian to be positive definite.
% select support vectors.
S = find(alpha > eps);
NS = length(S);
beta = alpha(S).*y(S);
XS = X(S,:);
% estimate w0 robustly (bias parameter)
dpos = find((y > 0) & (alpha > 0) & (alpha < 1/n));
dneg = find((y < 0) & (alpha > 0) & (alpha < 1/n));
    margvecs = [dpos ; dneg];
npos = length(dpos);
nneg = length(dneg);
Mpos = reshape(repmat(reshape(K(S,dpos), [NS npos 1]), [1 1 nneg]), [NS npos * nneg]);
Mneg = reshape(repmat(reshape(K(S,dneg), [NS 1 nneg]), [1 npos 1]), [NS npos * nneg]);
rho = mean(0.5*beta'*(Mpos - Mneg));
w0 = median(rho*y(margvecs) - sum(diag(beta)*K(S,margvecs))');
% store the results
svm.kernel = kernel;
svm.NS = NS;
svm.w0 = w0;
svm.beta = beta;
svm.XS = XS;
svm.rho = rho;
svm.param = param;
```

Figure 3: Solution to problem 3 part 3