Mehryar Mohri
Foundations of Machine Learning
Courant Institute of Mathematical Sciences
Homework assignment 3
April 5, 2013
Due: April 19, 2013

A. Kernels

1. Let $\mathcal{X}$ be a finite set. Show that the kernel K defined over $2^{\mathcal{X}}$, the set of subsets of $\mathcal{X}$, by

$$\forall A, B \in 2^{\mathcal{X}}, K(A, B) = \exp\Big(-\frac{1}{2}|A \Delta B|\Big),$$

where $A \Delta B$ is the symmetric difference of A and B is PDS (*hint*: you could use the fact that K is the result of the normalization of a kernel function K'). Note that this could define a similarity measure for documents based on the set of their common words, or n-grams, or gappy n-grams, or a similarity measure for images based on some patterns, or a similarity measure for graphs based on their commong sub-graphs.

Solution: $k\colon (A, B) \mapsto |A \cap B|$ is a PDS kernel over $2^{\mathcal{X}}$ since $k(A, B) = \sum_{x \in \mathcal{X}} 1_A(x) 1_B(x) = \Phi(A) \cdot \Phi(B)$, where, for any subset A, $\Phi(A) \in \{0, 1\}^{|\mathcal{X}|}$ is the vector whose coordinate indexed by $x \in \mathcal{X}$ is 1 if $x \in A$, 0 otherwise. Since k is PDS, $K' = \exp(k)$ is also PDS (theorem presented in class: PDS property preserved by composition with power series). Since K is the result of the normalization of K', it is also PDS. □

2. Let $\mathcal{X}$ be a finite set. Let K_0 be a PDS kernel over $\mathcal{X}$, show that K' defined by

$$\forall A, B \in 2^{\mathcal{X}}, K'(A, B) = \sum_{x \in A, x' \in B} K_0(x, x')$$

is a PDS kernel.

Solution: Let Φ_0 be a feature mapping associated to k_0, then $K'(A, B) = \sum_{x \in A, x' \in B} \phi_0(x) \cdot \phi_0(x') = \Big(\sum_{x \in A} \phi_0(x)\Big) \cdot \Big(\sum_{x' \in B} \phi_0(x')\Big) = \Psi(A) \cdot \Psi(B)$, where for any subset A, $\Psi(A) = \sum_{x \in A} \phi_0(x)$. □

3. Show that K defined by $K(x, x') = \frac{1}{\sqrt{1-(\mathbf{x}\cdot\mathbf{x}')}}$ for all $\mathbf{x}, \mathbf{x}' \in X = \{\mathbf{x} \in \mathbb{R}^N : \|\mathbf{x}\|_2 < 1\}$ is a PDS kernel. Bonus point: show that the dimension of the feature space associated to K is infinite (*hint*: one method to show that consists of finding an explicit expression of a feature mapping Φ).

Solution: $f : x \mapsto \frac{1}{\sqrt{1-x}}$ admits the Taylor series expansion

$$f(x) = \sum_{n=0}^{\infty} \binom{1/2}{n} (-1)^n x^n$$

for $|x| < 1$, where $\binom{1/2}{n} = \frac{\frac{1}{2}(\frac{1}{2}-1)\cdots(\frac{1}{2}-n+1)}{n!}$. Observe that $\binom{1/2}{n}(-1)^n > 0$ for all $n \geq 0$, thus, the coefficients in the power series expansion are all positive. Since the radius of the convergence of the series is one and that by the Cauchy-Schwarz inequality $|\mathbf{x}' \cdot \mathbf{x}| \leq \|\mathbf{x}'\|\|\mathbf{x}\| < 1$ for $\mathbf{x}, \mathbf{x}' \in X$, by the closure property theorem presented in class, $(\mathbf{x}, \mathbf{x}') \mapsto f(\mathbf{x}' \cdot \mathbf{x})$ is a PDS kernel.

Now, let $a_n = \binom{1/2}{n}(-1)^n$. Then, for $\mathbf{x}, \mathbf{x}' \in X$,

$$\begin{aligned} f(\mathbf{x}' \cdot \mathbf{x}) &= \sum_{n=0}^{\infty} a_n \Big(\sum_{i=1}^{N} x_i x'_i \Big)^n \\ &= \sum_{n=0}^{\infty} a_n \sum_{s_1+\cdots+s_N=n} \binom{n}{s_1, \ldots, s_N} (x_{i_1} x'_{i_1})^{s_1} \cdots (x_{i_N} x'_{i_N})^{s_N} \\ &= \sum_{s_1,\ldots,s_N \geq 0} a_{s_1+\cdots+s_N} \binom{s_1 + \cdots + s_N}{s_1, \ldots, s_N} (x_{i_1} x'_{i_1})^{s_1} \cdots (x_{i_N} x'_{i_N})^{s_N}, \end{aligned}$$

where the sums can be permuted since the series is absolutely summable. Thus, we can write $f(\mathbf{x}' \cdot \mathbf{x}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$ with

$$\Phi(\mathbf{x}) = \left(\sqrt{a_{s_1+\cdots+s_N} \binom{s_1 + \cdots + s_N}{s_1, \ldots, s_N}} x_{i_1}^{s_1} \cdots x_{i_N}^{s_N} \right)_{s_1,\ldots,s_N \geq 0}.$$

Φ is a mapping to an infinite-dimensional space. □

B. Support Vector Machines

1. Download and install the `libsvm` software library from:

   `http://www.csie.ntu.edu.tw/~cjlin/libsvm/` ,

and briefly consult the documentation to become more familiar with the tools.

2. Consider the `splice` data set

   `http://www.cs.toronto.edu/~delve/data/splice/desc.html`.

   Download the already formatted training and test files of that dataset from
   `http://www.cs.nyu.edu/~mohri/ml13/splice.train.txt`
   `http://www.cs.nyu.edu/~mohri/ml13/splice.test.txt`.

   Use the `libsvm` scaling tool to scale the features of all the data. The scaling parameters should be computed only on the training data and then applied to the test data.

   Solution:

```
# Scale training and test set.
libsvm-3.0/svm-scale -s output/scale.txt \
  output/splice.train.txt > output/splice.train.scaled.txt
libsvm-3.0/svm-scale -r output/scale.txt \
  output/splice.test.txt > output/splice.test.scaled.txt
```

3. Consider the corresponding binary classification which consists of distinguishing two types of splice junctions in DNA sequences using about 60 features. Use SVMs combined with polynomial kernels to tackle this problem.

   To do that, randomly split the training data into ten equal-sized disjoint sets. For each value of the polynomial degree, $d = 1, 2, 3, 4$, plot the average cross-validation error plus or minus one standard deviation as a function of C (let other parameters of polynomial kernels in `libsvm` be equal to their default values), varying C in powers of 10, starting from a small value $C = 10^{-k}$ to $C = 10^{k}$, for some value of k. k should be chosen so that you see a significant variation in training error, starting from a very high training error to a low training error. Expect longer training times with `libsvm` as the value of C increases.

   Solution: Figures in this solution section are courtesy of Jinho Jang.

   Figure 1 displays the cross validation error for polynomial kernels of degree $d \in \{1, 2, 3, 4\}$ as a function of the regularization parameter C. A direct

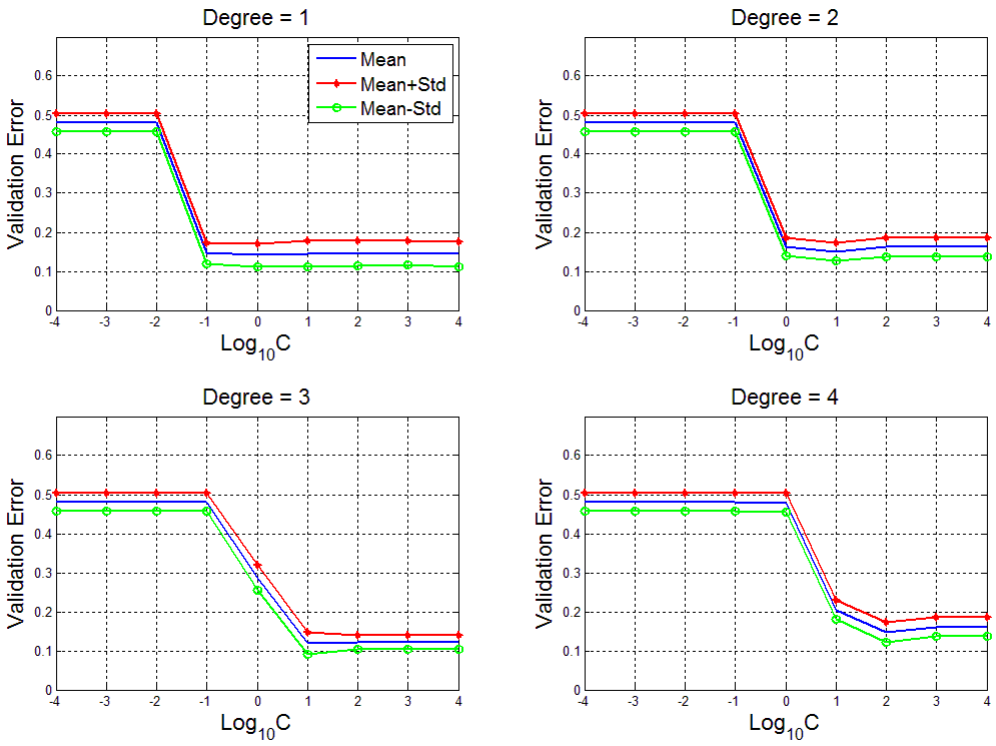

Figure 1: Validation error for SVMs on the DNA data with polynomial kernels of varying degrees $d = \{1, 2, 3, 4\}$ and regularization parameter $C = 10^k$; $k \in \{-4, -3, \ldots, 4\}$.

comparison of the mean CV error appears in Figure 2. Generally, we do not see dramatic differences in performance between values of $C > 10^2$ and $d = \{1, 2, 4\}$. As a curiosity, Figure 3 demonstrates that for $C = 10$, we eventaully see an overfitting phenomenon as we increase the degree beyond the scope of the assignment.

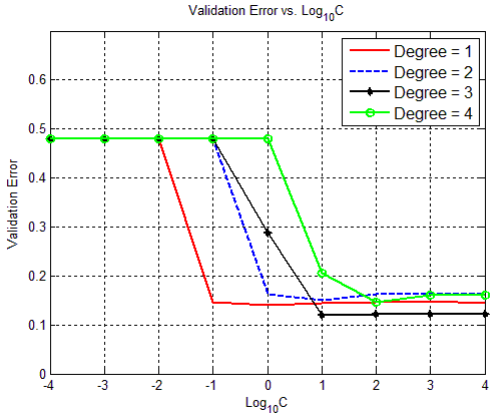

Figure 2: A direct comparison of the polynomial degrees and the validation error for SVMs on the DNA data with polynomial kernels of varying degrees $d = \{1, 2, 3, 4\}$ and regularization parameter $C = 10^k$; $k \in \{-4, -3, \ldots, 4\}$. We see that once $C \geq 10^2$, there are not dramatic differences between the settings.

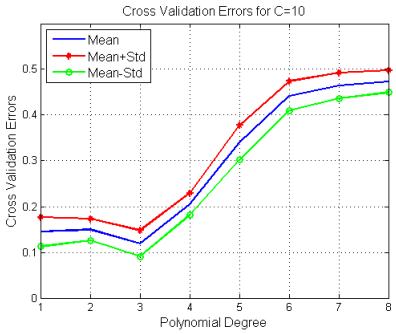

Figure 3: Validation errors for further polynomail degrees beyond those suggested in the problem statement. The algorithm begins to overfit as the polynomial degree increases.

Figure 4: The test and validation error as a function of degree (left panel) as well as the number of total and marginal support vectors (right panel).

4. Let (C^*, d^*) be the best pair found previously. Fix C to be C^*. Plot the ten-fold cross-validation error and the test errors for the hypotheses obtained as a function of d. Plot the average number of support vectors obtained as a function of d. How many of the support vectors lie on the margin hyperplanes?

   Solution: The first plot in Figure 4 shows that the test error decreases (slightly) with an increase in degree and also that the cross-validation error is (slightly) optimistic when compared to the test error on the held-out dataset.

   The second plot shows that the total number of marginal support vectors increases with d (as does the dimension of the feature space) while the total number of overall support vectors decreases. This also implies that the number of support vectors due to mistakes is decreasing, which agrees with the first plot. □

5. Suppose we replace in the primal optimization problem of SVMs the penalty term $\sum_{i=1}^m \xi_i = \|\boldsymbol{\xi}\|_1$ with $\|\boldsymbol{\xi}\|_2^2$, that is we use the quadratic hinge loss instead. Give the associated dual optimization problem and compare it with the dual optimization problems of SVMs.

   Solution: The optimization problem for this version of SVMs can be written as follows:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^m \xi_i^2$$
$$\text{subject to } \ \forall i \in [1,m],\ y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \ \wedge \ \xi_i \geq 0.$$

   The objective function is convex as a sum of the two convex function $\mathbf{w} \mapsto \|\mathbf{w}\|^2$ and $\boldsymbol{\xi} \mapsto \|\boldsymbol{\xi}\|^2$ and is infinitely differentiable. The constraints are affine functions. Slater's condition holds, the KKT theorem applies, and strong duality holds. The corresponding Lagrange function can be written as

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^m \xi_i^2 - \sum_{i=1}^m \alpha_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^m \beta_i\xi_i,$$

   with dual variables $\alpha_i, \beta_i \geq 0$. Differentiating with respect to the primal

variables gives:

$$\nabla_{\mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0 \implies \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$
$$\nabla_b L = -\sum_{i=1}^{m} \alpha_i y_i = 0 \implies \sum_{i=1}^{m} \alpha_i y_i = 0$$
$$\nabla_{\xi_i} L = 2C\xi_i - \alpha_i - \beta_i = 0 \implies \alpha_i + \beta_i = 2C\xi_i.$$

Plugging in these equalities in L yields:

$$\begin{aligned} L &= -\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m} \alpha_i - C \sum_{i=1}^{m} \xi_i^2 \\ &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \frac{1}{2C} \sum_{i=1}^{m} (\alpha_i + \beta_i)^2. \end{aligned}$$

Thus, this gives the following dual optimization problem:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \frac{1}{2C} \sum_{i=1}^{m} (\alpha_i + \beta_i)^2$$
$$\text{subject to } (\boldsymbol{\alpha} \geq 0) \wedge (\boldsymbol{\beta} \geq 0) \wedge (\sum_{i=1}^{m} \alpha_i y_i = 0).$$

The maximization over $\boldsymbol{\beta}$ clearly gives $\boldsymbol{\beta} = 0$. This corresponds to the fact that the conditions $\xi_i \geq 0$ were in fact not necessary. In view of that, the dual optimization problem becomes

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \frac{1}{2C} \sum_{i=1}^{m} \alpha_i^2$$
$$\text{subject to } (\boldsymbol{\alpha} \geq 0) \wedge (\sum_{i=1}^{m} \alpha_i y_i = 0).$$

Let $\mathbf{K} \in \mathbb{R}^{m \times m}$ denote the kernel matrice associated to the sample, $\mathbf{K}_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$, and $\widetilde{\mathbf{K}}$ the kernel matrix defined by $\widetilde{\mathbf{K}} = \mathbf{K} + \frac{1}{2C}\mathbf{I}$. Then, the problem can be equivalently written as

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \widetilde{\mathbf{K}}_{ij}$$
$$\text{subject to } (\boldsymbol{\alpha} \geq 0) \wedge (\sum_{i=1}^{m} \alpha_i y_i = 0).$$

The dual problem therefore coincides with that of SVMs in the separable case modulo the change of the kernel matrix. □

6. In class, we presented margin-based generalization bounds in support of the SVM algorithm based on the standard hinge loss. Can you derive similar margin-based generalization bounds when the quadratic hinge loss is used? To do that, you could use instead of the margin loss function Φ_ρ defined in class the function Ψ_ρ defined by

$$\Psi_\rho(u) = \begin{cases} 1 & \text{if } u \leq 0 \\ \left(\frac{u}{\rho} - 1\right)^2 & \text{if } u \in [0, \rho] \\ 0 & \text{otherwise,} \end{cases}$$

and show that it is a Lipschitz function. Compare the empirical and complexity term of your generalization bound to those given in class using Φ_ρ.

Solution: Ψ_ρ is differentiable over $]0, \rho[$ and its differential at u is $\Psi'_\rho(u) = \frac{2}{\rho}\left(\frac{u}{\rho} - 1\right)$, thus $|\Psi'_\rho(u)| \leq \frac{2}{\rho}$. This can be used to show straightforwardly that Ψ_ρ is $\frac{2}{\rho}$-Lipschitz. In view of that, using the same proof techniques as those used to prove the general margin bound of slide 34, lecture 4, we can conclude that for any $\delta > 0$, with probability at least $1 - \delta$,

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m} \Psi_\rho(y_i h(x_i)) + \frac{4}{\rho}\mathfrak{R}_m(H) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Note that $\Psi_\rho = \Phi_\rho^2$ and that $\Psi_\rho \leq \Phi_\rho$, thus the empirical term in this bound is always smaller than the one in the bound presented in class. But, the complexity term is multiplied by a factor of 2: $\frac{4}{\rho}\mathfrak{R}_m(H)$ instead of $\frac{2}{\rho}\mathfrak{R}_m(H)$. □

C. Boosting

1. Let $\Psi : \mathbb{R} \to \mathbb{R}$ denote the function defined by $\Psi(u) = (1-u)^2 1_{u \leq 1}$. Show that Ψ is an upper bound on the binary loss function and that it is convex and differentiable. Use Ψ to derive a boosting-style algorithm as in the case of the exponential function used in AdaBoost using coordinate descent. Describe the algorithm in detail.

Solution: Since $\Psi \geq 0$, for $u > 0$, the inequality $\Psi(u) \geq 1_{u \leq 0}(u)$ holds. For $u \leq 0$, $(1-u)^2 \geq (1-0)^2 = 1_{u \leq 0}(u)$. Thus, Ψ is an upper bound on $1_{u \leq 0}$.

The functions $u \mapsto (1-u)^2$ and $u \mapsto 0$ are both continuously differentiable and admit the same differential value, 0, at $u = 1$. Ψ is thus continously differentiable. For $u \leq 1$, $\Psi'(u) = -2(1-u)$ and for $u \geq u$, $\Psi'(u) = 0$, thus Ψ' is non-decreasing and Ψ is convex.

We adopt the notation already used in class. The objective function becomes

$$F(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \Psi(y_i f(x_i)) = \sum_{i=1}^{m} \Psi\Big(y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\Big).$$

We denote by f_t the function defined after t rounds of boosting: $f_t = \sum_{s=1}^{t} \alpha_s h_s$. For convenience, we also define $f_0 = 0$. Note that since $\Psi' \leq 0$, $\sum_{i=1}^{m} \Psi'(y_i f_t(x_i)) = 0$ implies that $\Psi'(y_i f_t(x_i)) = 0$ for all $i \in [1, m]$, that is $y_i f_t(x_i) \geq 1$. f_t is then correctly classifying all points with margin 1. If that occurs, we can stop boosting. In the following, we will therefore assume that $\sum_{i=1}^{m} \Psi'(y_i f_t(x_i)) \neq 0$ for all $t \in [1, T]$ where T is the minimum of a predefined number of rounds and the last round at which this condition holds.

The original distribution D_1 is the uniform distribution over the sample: $D_1(i) = \frac{1}{m}$ for all $i \in [1, m]$. The distribution D_t at the end of the $(t-1)$th round, $t \in [2, T]$ is defined by

$$D_t(i) = \frac{\Psi'(y_i f_{t-1}(x_i))}{\sum_{i=1}^{m} \Psi'(y_i f_{t-1}(x_i))}.$$

Let ϵ_t denote the coordinate descent direction at time t, which is determined by $\mathbf{e}_t = \operatorname{argmin}_t \frac{dF(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_t)}{d\eta}\big|_{\eta=0}$. By definition,

$$F(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_t) = \sum_{i=1}^{m} \Psi\Big(- y_i \sum_{s=1}^{t-1} \alpha_s h_s(x_i) - \eta y_i h_t(x_i)\Big).$$

Then, we can write

$$\begin{aligned}
\frac{dF(\alpha_{t-1} + \eta \mathbf{e}_t)}{d\eta}\bigg|_{\eta=0} &= \sum_{i=1}^{m} y_i h_t(x_i) \Psi'\Big(y_i \sum_{s=1}^{t-1} \alpha_s h_s(x_i)\Big) \\
&= \sum_{i=1}^{m} y_i h_t(x_i) D_t(i) \Big[\sum_{i=1}^{m} \Psi'\big(y_i f_{t-1}(x_i)\big)\Big] \\
&= [(1-\epsilon_t) - \epsilon_t] \Big[\sum_{i=1}^{m} \Psi'\big(y_i f_{t-1}(x_i)\big)\Big] \\
&= [2\epsilon_t - 1] \Big[- \sum_{i=1}^{m} \Psi'\big(y_i f_{t-1}(x_i)\big)\Big],
\end{aligned}$$

SQUARELOSSBOOST($S=((x_1,y_1),\ldots,(x_m,y_m))$)

for $i \leftarrow 1$ **to** m **do**
  $D_1(i) \leftarrow \frac{1}{m}$
for $t \leftarrow 1$ **to** T **do**
  $h_t \leftarrow$ base classifier in H with small error $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$
  $\alpha_t \leftarrow \eta$ ▷ obtained using search using expression of η.
  $f_t \leftarrow f_{t-1} + \alpha_t h_t$
  for $i \leftarrow 1$ **to** m **do**
    $D_{t+1}(i) \leftarrow \frac{\Psi'(y_i f_t(x_i))}{\sum_{i=1}^m \Psi'(y_i f_t(x_i))}$
return $h = \text{sign}(f_T)$

Figure 5: Pseudocode of the boosting algorithm derived.

where $\epsilon_t = \Pr_{i\sim D_t}[h_t(x_i) \neq y_i]$. Since $\Psi' \leq 0$, the base classifier h_t chosen at round t is thus the one with the smallest weighted error ϵ_t according to the distribution D_t.

To determine the step, we search η such that:

$$\frac{dF(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_t)}{d\eta} = 0 \Leftrightarrow \sum_{i=1}^m y_i h_t(x_i) \Psi'\Big(y_i f_{t-1}(x_i) + \eta y_i h_t(x_i)\Big) = 0.$$

Let $J(\eta)$ be defined by $J(\eta) = \{i\colon y_i f_{t-1}(x_i) + \eta y_i h_t(x_i) < 1\}$. Using $(y_i h_t(x_i))^2 = 1$ The condition can then be rewritten as

$$\begin{aligned}
\frac{dF(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_t)}{d\eta} = 0 &\Leftrightarrow \sum_{i=1}^m y_i h_t(x_i)\Big(1 - y_i f_{t-1}(x_i) - \eta y_i h_t(x_i)\Big) = 0 \\
&\Leftrightarrow \eta = \frac{1}{|J(\eta)|} \sum_{i \in J(\eta)} y_i h_t(x_i) - h_t(x_i) f_{t-1}(x_i) \\
&\Leftrightarrow \eta = \frac{\sum_{i \in J(\eta)} h_t(x_i)(y_i - f_{t-1}(x_i))}{|J(\eta)|}.
\end{aligned}$$

This does not provide a closed form solution, but the best step η_t at time t can be determined using this formula by varying η. The pseudocode of the algorithm is given in Figure 1.

2. Implement that algorithm with boosting stumps and apply the algorithm to the same data set as question B with the same training and test sets. Plot

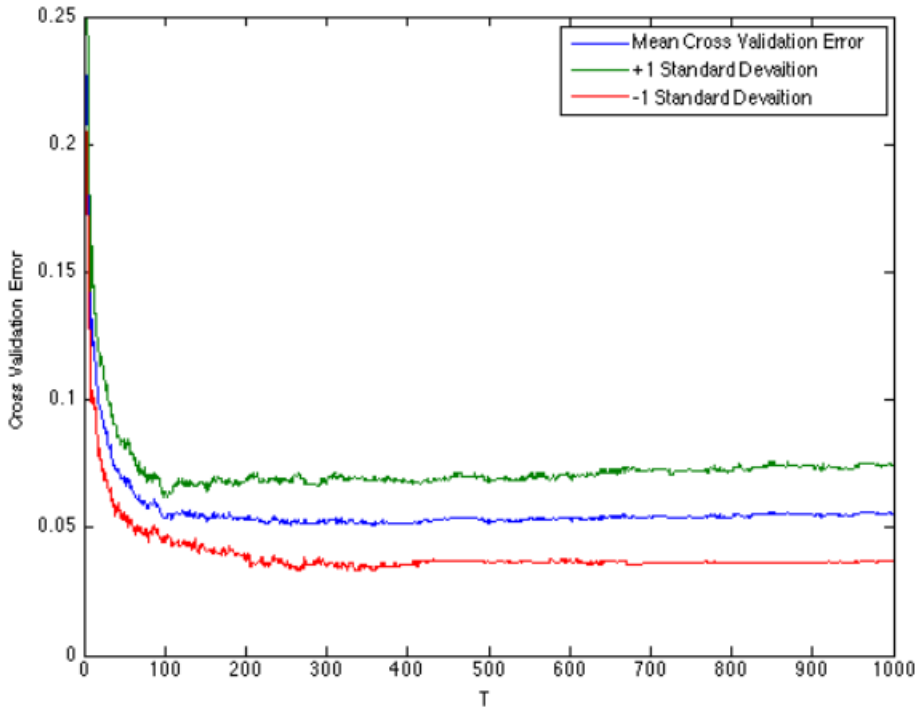

Figure 6: Cross validation error shown with ± 1 standard deviation.

the average cross-validation error plus or minus one standard deviation as a function of the number of rounds of boosting T by selecting the value of this parameter out of $\{10, 10^2, \ldots, 10^k\}$ for a suitable value of k, as in question B. Let T^* be the best value found for the parameter. Plot the error on the training and test set as a function of the number of rounds of boosting for $t \in [1, T^*]$. Compare your results with those obtained using SVMs in question B.

Solution:

In figure 6 we see a typical plot of the average cross-validation performance shown with ± 1 standard deviation. Note that the error descreases exponentially and eventually levels out after roughly 150 iterations. In figure **??** we see the training and test error up to and after $T^* = 150$ rounds. Note that the test error eventually levels off, while the training error continues to decrease

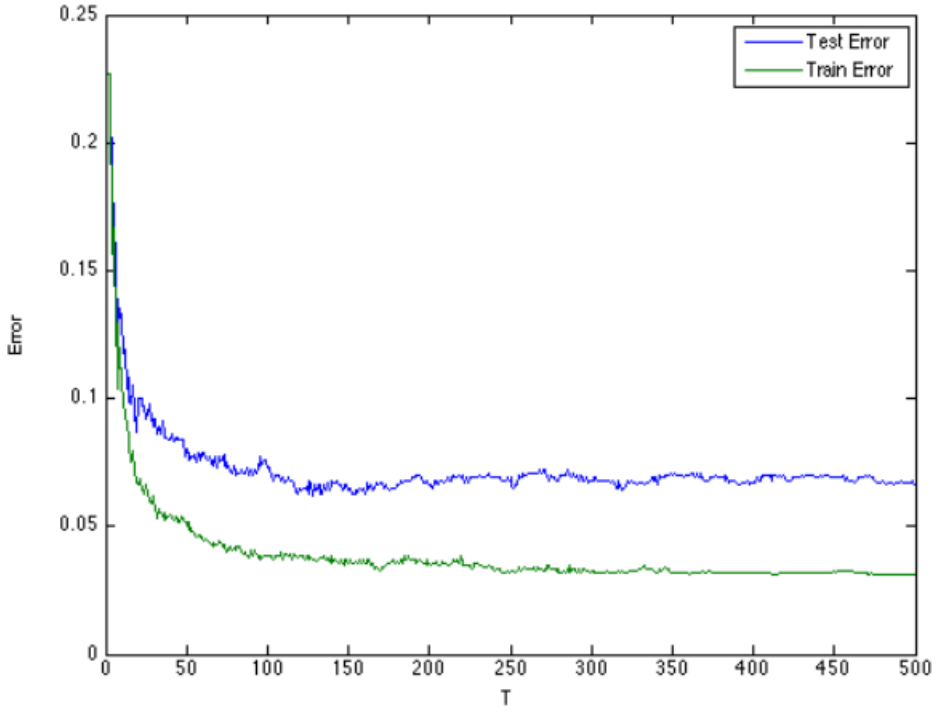

Figure 7: Test and training error for boosting vs number of rounds.

towards zero. Overall, the AdaBoost algorithm performs only slightly better than SVM algorithm increasing accuracy by about 3%. □