6.867 Machine Learning

Solutions for Problem Set 4

Wednesday, November 12

Problem 1: Regularized Least-Squares Feature Selection

(1-1) (10pts) First, let's rewrite the least-squares error metric $J(\mathbf{w}; 0)$ as a function of the k-th parameter w_k (viewing remaining parameters $\mathbf{w}_{-k}$ as fixed constants).

$$\begin{aligned} J(w_k; 0) &= \frac{1}{2n}\sum_i (y_i - \mathbf{w}'\phi(\mathbf{x}_i))^2 && (1)\\ &= \frac{1}{2n}\sum_i (y_i - \mathbf{w}'_{-k}\phi_{-k}(\mathbf{x}_i) - w_k\phi_k(\mathbf{x}_i))^2 && (2)\\ &= \frac{1}{2n}\sum_i \left\{\phi_k^2(\mathbf{x}_i)w_k^2 + 2\phi_k(\mathbf{x}_i)(y_i - \mathbf{w}'_{-k}\phi_{-k}(\mathbf{x}_i))w_k + (y_i - \mathbf{w}'_{-k}\phi_{-k}(\mathbf{x}_i))^2\right\} && (3)\\ &= \frac{1}{2}a_k w_k^2 - c_k w_k + d_k && (4) \end{aligned}$$

where

$$\begin{aligned} a_k &= \frac{1}{n}\sum_i \phi_k^2(\mathbf{x}_i) && (5)\\ c_k &= \frac{1}{n}\sum_i \phi_k(\mathbf{x}_i)(y_i - \mathbf{w}'_{-k}\phi_{-k}(\mathbf{x}_i)) && (6)\\ d_k &= \frac{1}{2n}\sum_i (y_i - \mathbf{w}'_{-k}\phi_{-k}(\mathbf{x}_i))^2 && (7) \end{aligned}$$

which is a quadratic function of w_k. Computing the partial derivative w.r.t. w_k we obtain a linear function of w_k.

$$\frac{\partial J(\mathbf{w}; 0)}{\partial w_k} = a_k w_k - c_k \qquad (8)$$

Then, taking the subdifferential of $J(\mathbf{w}; \lambda)$ w.r.t. w_k, we obtain

$$\begin{aligned} \partial_{w_k} J(\mathbf{w}; \lambda) &= \partial_{w_k}\{J(\mathbf{w}; 0) + \lambda\|w\|_1\} && (9)\\ &= \partial_{w_k} J(\mathbf{w}; 0) + \lambda\partial_{w_k}\|\mathbf{w}\|_1 && (10)\\ &= \frac{\partial J(\mathbf{w}; 0)}{\partial w_k} + \lambda\partial_{w_k}|w_k| && (11)\\ &= (a_k w_k - c_k) + \lambda\partial_{w_k}|w_k| && (12) \end{aligned}$$

The subdifferential of the absolute value function is

$$\partial_{w_k}|w_k| = \begin{cases} \{-1\}, & w_k < 0 \\ [-1, +1], & w_k = 0 \\ \{+1\}, & w_k > 0 \end{cases} \tag{13}$$

Hence, scaling each element of the subdifferential by λ and adding $\frac{\partial J(\mathbf{w};0)}{\partial w_k}$ we have

$$\partial_{w_k} J(\mathbf{w};\lambda) = \begin{cases} \{(a_k w_k - c_k) - \lambda\}, & w_k < 0 \\ [-c_k - \lambda, -c_k + \lambda], & w_k = 0 \\ \{(a_k w_k - c_k) + \lambda\}, & w_k > 0 \end{cases} \tag{14}$$

as was to be shown.

Interpretation of c_k: Essentially, c_k measures the sample correlation between the k-th feature $\phi_k(\mathbf{x})$ and the prediction error $e_{-k} = y - w'_{-k}\phi_{-k}(\mathbf{x})$ based upon the other features. If this were zero, then feature k is orthogonal to the prediction error and we couldn't reduce the prediction error by including ϕ_k in our linear predictions. Hence, the *magnitude* of c_k is an indication of how relevant feature ϕ_k is for predicting y (relative to the other features and corresponding parameter settings).

(1-2) (10pts) Note that the subdifferential $\partial_{w_k} J(\mathbf{w};\lambda)$ is a monotonically increasing, piecewise linear function of w_k with slope $a_k > 0$ and a "jump" of $+2\lambda$ at $w_k = 0$. The value of c_k (relative to λ) controls where the subdifferential "crosses" the w_k-axis (i.e. contains a zero). This zero-intercept $\hat{w}_k$ is precisely the global minimum we seek satisfying the optimality condition $0 \in \partial_{w_k} J(\hat{w}_k;\lambda)$. See Figure 1 for illustrative plots of $\partial_{w_k} J(w_k;\lambda)$ for each of the three cases discussed below:

(a) If $c_k < -\lambda$, then $-c_k - \lambda > 0$ so that the zero-intercept is less than zero (see Fig. 1-a). Hence, we solve $a_k w_k - (c_k + \lambda) = 0$ for the global minimizer:

$$\hat{w}_k = \frac{c_k + \lambda}{a_k} < 0 \tag{15}$$

(b) If $c_k \in [-\lambda, +\lambda]$, then $-\lambda < c_k < +\lambda$, or $-c_k - \lambda < 0 < -c_k + \lambda$, or $0 \in [-c_k - \lambda, -c_k + \lambda] = \partial_{w_k} J(0;\lambda)$ (also see Fig. 1-b). Hence, the global minimum occurs at:

$$\hat{w}_k = 0 \tag{16}$$

(c) If $c_k > +\lambda$, then $-c_k + \lambda < 0$ so that the zero-intercept is greater than zero (see Fig. 1-c). Hence, we solve $a_k w_k - (c_k - \lambda) = 0$ for the global minimizer:

$$\hat{w}_k = \frac{c_k - \lambda}{a_k} > 0 \tag{17}$$

Then, putting these results together, $\hat{w}_k$ is a continuous, monotonically increasing, piecewise linear function of c_k:

$$\hat{w}_k(c_k) = \begin{cases} (c_k + \lambda)/a_k, & c_k < -\lambda \\ 0, & c_k \in [-\lambda, +\lambda] \\ (c_k - \lambda)/a_k, & c_k > +\lambda \end{cases} \tag{18}$$

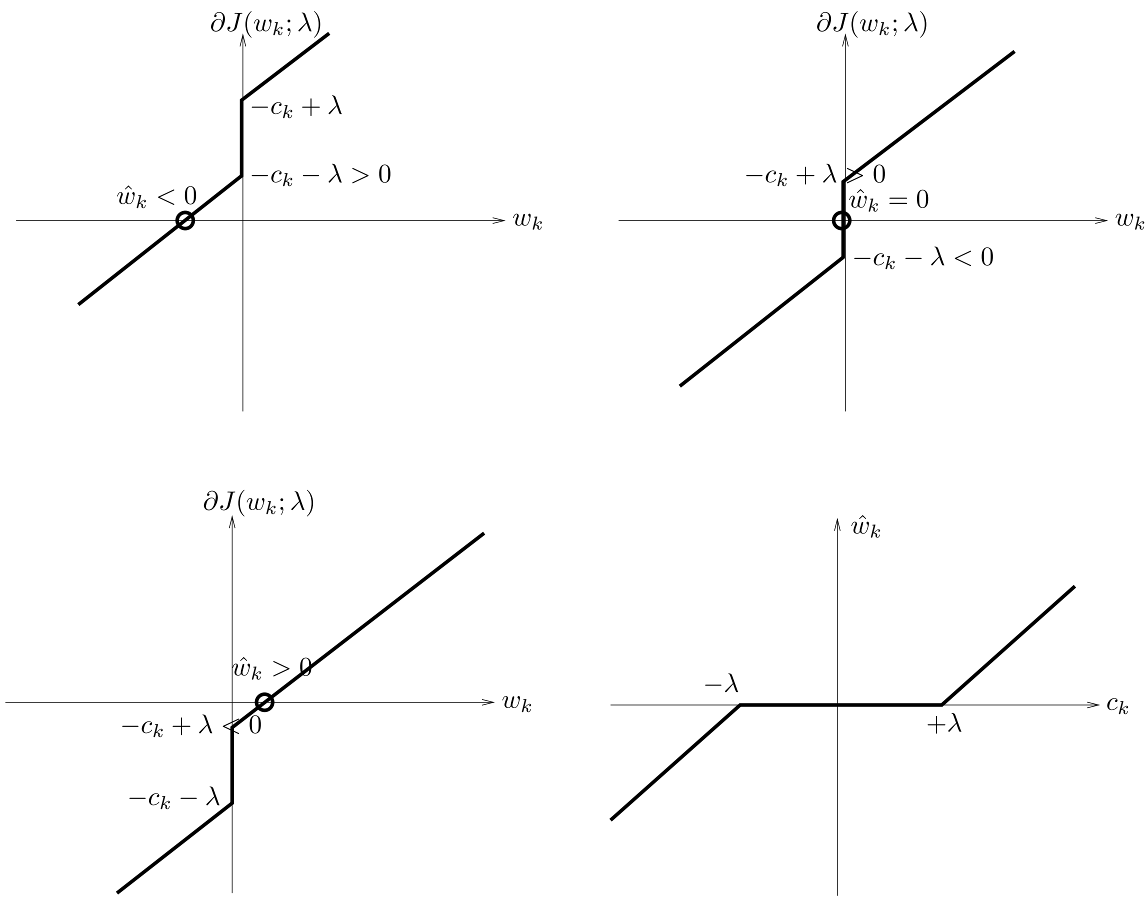

Figure 1: Plots of $\partial J(w_k;\lambda)$ vs. w_k when $c_k<-\lambda$ (top left), $-\lambda<c_k<+\lambda$ (top right), $c_k>+\lambda$ (bottom left). Plot of $\hat{w}_k$ vs. c_k (bottom right).

An illustrative plot of w_k vs. c_k is shown in Figure 1.

Interpretation of λ: The regularization parameter λ acts as a cut-off threshold relative to the coefficient c_k which, as discussed previously, indicates how important feature ϕ_k is for performing linear prediction of y. If $|c_k| < \lambda$, then feature ϕ_k is deemed irrelevant (or nearly so) and is hence omitted from our regularized predictor by setting w_k to zero. (this is the answer we were looking for)

Moreover, when $|c_k| > \lambda$ we do not actually set w_k to the (unregularized) least-squares value c_k/a_k, but rather bias the estimate towards zero by an amount λ/a_k. Hence, λ also controls by how much we "underestimate" the remaining (non-zero) parameters.

(1-3) (10pts) The code you need to add to `reg_least_sq.m` is given below.

To compute c_k:

```
w_not_k = w;
w_not_k(k) = 0.0;
e_not_k = y - X * w_not_k; % prediction error w/o feature k
c_k = X(:,k)'*e_diff_k/n;  % inner product of feature k w/ prediction error
```

To set $\hat{w}_k$:

```
if (c_k < -lambda)
  w_hat_k = (c_k + lambda)/a(k);
elseif (c_k > lambda)
  w_hat_k = (c_k - lambda)/a(k);
else
  w_hat_k = 0.0; % l1-regularization forces wieghts of less informative features to zero
end
```

(1-4) (10pts) The code you needed to modify in `hw4_prob1.m` is shown below (compute training error, l1-norm, penalized objective and training error):

```
% generate additional plots requested in the problem...
l = length(Lambda);
D = zeros(l,5);
for k = 1:l

  w = W(:,k);

  % half avg. squared training error
  D(k,1) = 0.5 * mean((train.y - train.X * w).^2);

  % l1 regularization penalty
  D(k,2) = sum(abs(w));

  % objective
```

```
  D(k,3) = D(k,1) + Lambda(k) * D(k,2);

  % test error
  D(k,4) = 0.5 * mean((test.y - test.X * w).^2);

  % l0 norm
  D(k,5) = length(find(w));

end
```

The requested plots are shown in Figure 2.

Observations;

- As we increase λ, more weight is placed keeping the $\|\mathbf{w}\|_1$ small, less weight is placed on minimizing the mean-squared error. Consequently, the training error is monotonically decreasing while the regularization penalty is monotonically decreasing.

- The minimized objective function, combining the mean-squared error and λ times the penalty, is apparently a smooth, concave, monotonically increasing function of λ (in fact, this follows as this is the "dual function" of a constrained optimization function). Note that this function can't be used to select preferable value of λ. $J(\mathbf{w}; \lambda)$ is only meant for comparing different values of $\mathbf{w}$ for a given λ.

- The test error, on the other hand, tends to exhibit a minimum for a certain critical value of λ. The location of this minima, for this data set atleast, appears to approach zero as we increase the size of the training set. We would expect this, i.e. that the more training data we have, the less regularization we should use.

- In practice, we could select the value of λ by computing the leave-one-out cross validation metric for all λ and seeking the value of λ which minimizes this estimate of generalization error. This should exhibit similar behavior as we see here for the test error.

- Overall, increasing λ drives parameters w_k to zero. The larger we set λ, the more parameters are forced to zero so that the l_0 norm tends to decrease with λ. This is in contrast to using l_2 regularization which also forces parameters towards zero, but only causes parameters to approach zero asympototically rather then suddenly as we observe here.

- It is also interesting to note that, while the overall trend is of decreasing parameter values as we increase λ, in some cases a few of the parameters can actually increase temporarily as we increase λ but eventually begin to decrease. This, apparently, has to do with the interaction of the features (the importance of each feature is relative to the other feature present and the weight placed on those other features).

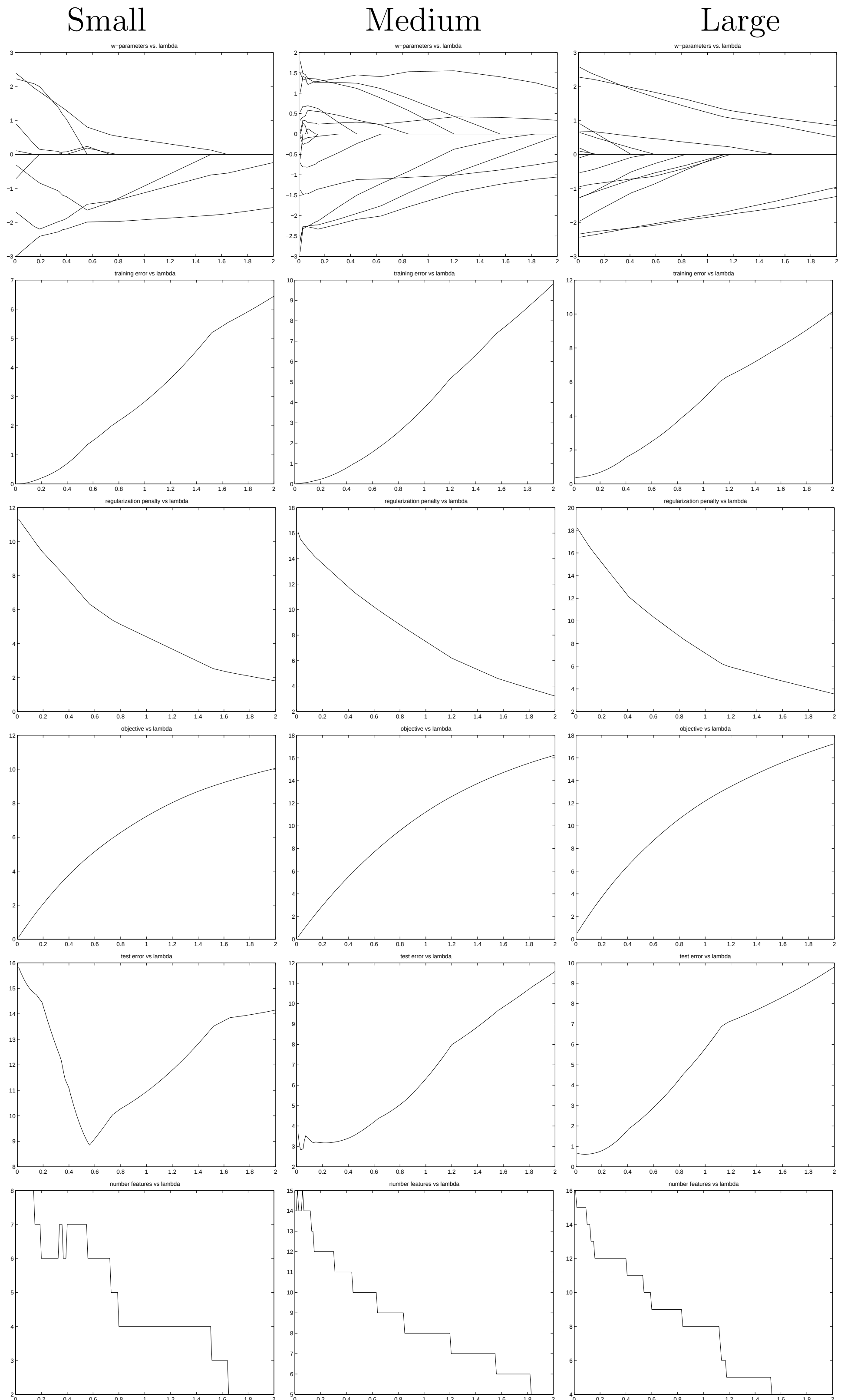

Figure 2: Results for 3 training sets: small (1st column), medium (2nd column) and large (3rd column). Plots of $w_k(\lambda)$ vs. λ for all k (1st row), training error vs. λ (2nd row), l_1 norm vs. λ (3rd row), minimized objective vs. λ (4th row), test error vs. λ (5th row), l_0 norm vs. λ (6th row).

Problem 2: Boosting

(2-1) (10pts) First, we show that the minimization in Step 2 of the general algorithm (LHS below), with $\text{Loss}(z) = e^{-z}$, is the same as the minimization performed by AdaBoost (RHS below), e.g. that

$$\arg\min_{\alpha} \sum_i \text{Loss}\left(y_i h_{k-1}(\mathbf{x}_i) + \alpha y_i h(\mathbf{x}_i; \hat{\theta}_k)\right) = \arg\min_{\alpha} \sum_i \tilde{W}_i^{(k-1)} \exp\left\{-\alpha y_i h(\mathbf{x}_i; \hat{\theta}_k\right\} \tag{19}$$

with (from AdaBoost)

$$\tilde{W}_i^{(k-1)} = c \cdot \exp\{-y_i h_{k-1}(\mathbf{x}_i)\} \tag{20}$$

where c is a normalization constant (weights sum to 1). Evaluating the objective in LHS gives;

$$\begin{aligned} \sum_i \text{Loss}\left(y_i h_{k-1}(\mathbf{x}_i) + \alpha y_i h(\mathbf{x}_i; \hat{\theta}_k)\right) &= \sum_i \exp\{-y_i h_{k-1}(\mathbf{x}_i)\} \exp\{-\alpha y_i h(\mathbf{x}_i; \hat{\theta}_k)\} &(21)\\ &= \frac{1}{c} \sum_i \tilde{W}_i^{(k-1)} \exp\left\{-\alpha y_i h(\mathbf{x}_i; \hat{\theta}_k\right\} &(22)\end{aligned}$$

which is proportional to the objective minimized by AdaBoost so that minimizing value of α is the same for both algorithms.

Second, we show that the weight assignments in Step 3 of the general algorithm (for stage k) are the same as in given by AdaBoost (written for stage $k-1$ above).

$$\begin{aligned} \tilde{W}_i^{(k)} &= -c \cdot dL(y_i h_k(\mathbf{x}_i)) &(23)\\ &= c \cdot \exp\{-y_i h_k(\mathbf{x}_i)\} &(24)\end{aligned}$$

which is the same as in AdaBoost.

(2-2) (10pts) With the logistic loss $\text{Loss}(z) = \log(1 + e^{-z})$ we have

$$dL(z) = -\frac{e^{-z}}{1 + e^{-z}} \tag{25}$$

The weights are then given by

$$\tilde{W}_k^{(k)} = c \cdot \frac{\exp(-y_i h_k(\mathbf{x}_i)}{1 + \exp(-y_i h_k(\mathbf{x}_i))} \tag{26}$$

with normalization constant

$$c = \left(\sum_i \frac{\exp(-y_i h_k(\mathbf{x}_i)}{1 + \exp(-y_i h_k(\mathbf{x}_i))}\right)^{-1} \tag{27}$$

(2-3) (10pts) At stage k, $\hat{\alpha}$ is chosen to minimize $J(\alpha, \hat{\theta}_k)$, e.g. to solve $\frac{\partial J(\alpha; \hat{\theta}_k)}{\partial \alpha} = 0$. In general,

$$\frac{\partial J(\alpha; \hat{\theta}_k)}{\partial \alpha} = \frac{1}{n} \sum_i \frac{\partial}{\partial \alpha} L(y_i h_{k-1}(\mathbf{x}_i) + \alpha y_i h(\mathbf{x}_i; \hat{\theta}_k)) \tag{28}$$

$$
\begin{aligned}
&= \frac{1}{n}\sum_i dL(y_i h_{k-1}(\mathbf{x}_i) + \alpha y_i h(\mathbf{x}_i;\hat{\theta}_k)) y_i h(\mathbf{x}_i;\hat{\theta}_k) && (29)\\
&\propto \sum_i \tilde{W}_i^{(k)} y_i h(\mathbf{x}_i;\hat{\theta}_k) && (30)
\end{aligned}
$$

so that we must have

$$
\sum_i \tilde{W}_i^{(k)} y_i h(\mathbf{x}_i;\hat{\theta}_k) = 0 \tag{31}
$$

Then, the weighted training error for $h(\mathbf{x};\hat{\theta}_k)$ (relative to the updated weights $\tilde{W}_i^{(k)}$ determined by $\hat{\alpha}$) is

$$
\begin{aligned}
e_k &= \frac{1}{2}\left\{1 - \sum_i \tilde{W}_i^{(k)} y_i h(\mathbf{x}_i;\hat{\theta}_k)\right\} && (32)\\
&= \frac{1}{2}(1-0) && (33)\\
&= \frac{1}{2} && (34)
\end{aligned}
$$

(2-4) (5pts) The following excerpt from `boost_logistic.m` computes the weights.

```
% insert weight update here
W = exp(-H.*y)./(1+exp(-H.*y));
W = W/sum(W);
```

(2-5) (10pts) See the following script `hw4_prob2.m`:

```
clear all;
close all;
data = loaddata;

err = zeros(50,1);
for k = 1:50

  % train -- run boosting algorithm for k iterations
  model = boost_logistic(data.xtrain,data.ytrain,k);

  % test
  y_est = sign(eval_boost(model,data.xtest));
  err(k) = sum(y_est ~= data.ytest);

end

plot(err,'o-');
xlabel('Number test examples misclassified');
ylabel('Number of Boosting Iterations');
refresh;
print -deps boost_plot.eps;
```

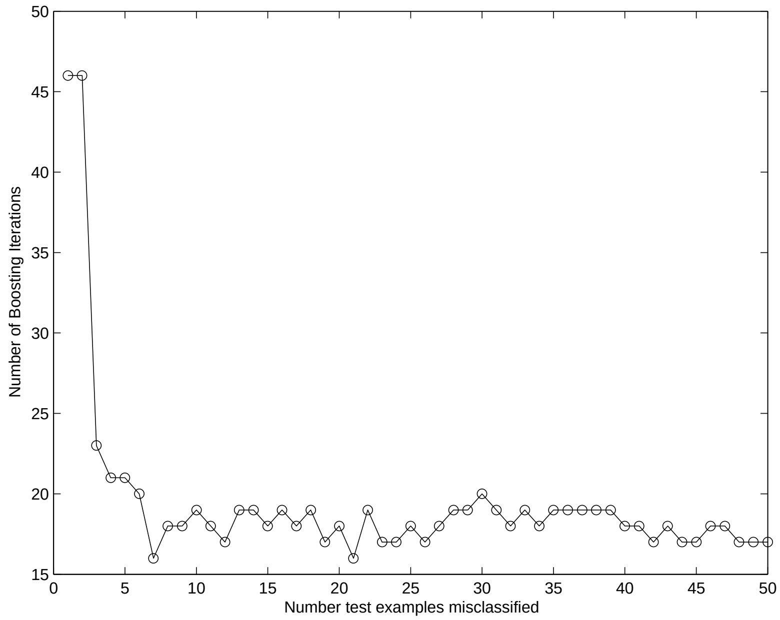

Figure 3: Plot of number of misclassified test cases (out of 483 cases) vs. number of boosting iterations.

Problem 3: VC-Dimension

Part I: Linear Classifiers

In this part we consider the set of linear classifiers $\mathcal{H}_d = \{h_{\mathbf{w}} : \mathcal{R}^d \rightarrow \{-1,+1\} | \mathbf{w} \in \mathcal{R}^d\}$ comprised of classifiers of the form:

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} +1, & \mathbf{w}'\mathbf{x} > 0 \\ -1, & \mathbf{w}'\mathbf{x} \leq 0 \end{cases} \tag{35}$$

(3-1) (10pts) We wish to show the existence of a set of d points $\mathbf{x}^1, \ldots, \mathbf{x}^d \in \mathcal{R}^d$ such that for any arbitrary choice of labels $y^1, \ldots, y^d \in \{-1,+1\}$ there exists a $\mathbf{w} \in \mathcal{R}^d$ s.t. $h_{\mathbf{w}}(\mathbf{x}^k) = y^k$ for $k = 1, \ldots, d$. Then, by definition, $\mathcal{H}_d$ shatters the set $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^d\}$.

Let's define the k-th point $\mathbf{x}^k$ to have all zero entries except for a one in entry k.

$$\mathbf{x}_i^k = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases} \tag{36}$$

Given an arbitrary set of labels $Y = (y^1, \ldots, y^d)$ let's define the corresponding $\mathbf{w}$ by $\mathbf{w}_k(Y) = y^k$. Then,

$$h_{\mathbf{w}(Y)}(\mathbf{x}^k) = \text{sign}(\mathbf{w}'(Y)\mathbf{x}^k) = \text{sign}(\mathbf{x}_k(Y)) = y^k \tag{37}$$

which is precisely what we wished to show. Hence, we have exhibited a set X with d points which is shattered by $\mathcal{H}_d$.

The VC-dimension of $\mathcal{H}_d$ is defined as

$$VC(\mathcal{H}_d) = \max\{|X| \ : \ \mathcal{H}_d \text{ shatters } X\} \tag{38}$$

Since our X is shattered by $\mathcal{H}_d$, we must have that $VC(\mathcal{H}_d) \geq |X| = d$.

(3-2) (10pts) We now wish to show that no set of $d+1$ points can be shattered by $\mathcal{H}_d$.

Proof by contradiction. Suppose there exists such a set $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^{d+1}\}$ which can be shattered by $\mathcal{H}_d$. We wish to show that this leads to a logical contradiction.

Without any loss of generality, assume that X does not contain the origin (if it did, it couldn't possible be shattered by $\mathcal{H}_d$ because any labeling which assigns a $y = +1$ to the $x = 0$ element can't be produced by the $h_{\mathbf{w}}$ decision rule which always decides $h_{\mathbf{w}}(\mathbf{0}) = -1$ for any choice of $\mathbf{w}$).

Consider the set $\hat{X} = \{\mathbf{0}\} \cup X$. By Radon's theorem, we can partition $\hat{X}$ into subsets S_1 and S_2 s.t. the convex hulls of S_1 and S_2 intersect (contain a common point). Let S_1 be the set containing $\mathbf{x}^0 = \mathbf{0}$. Then, let $\hat{Y} = (y^0, y^1, \ldots, y^{d+1})$ be the labeling of these points where $y^k = -1$ for each point $\mathbf{x}^k \in S_1$ and $y^k = +1$ for each point $x^k \in S_2$. By assumption, there exists a $\mathbf{w}$ s.t. $h_{\mathbf{w}}(x^k) = y^k$ for $k = 0, 1, \ldots, d+1$ (since we can shatter X and this holds by construction for $\mathbf{x}^0$). This means that all the points S_1 are contained in the open positive half-space

$$H_{\mathbf{w}}^{+} = \{\mathbf{x} | \mathbf{w}'\mathbf{x} > 0\} \tag{39}$$

while all the points S_2 are contained in the closed negative half-space $H_{\mathbf{w}}^{-} = \mathcal{R} \setminus H_{\mathbf{w}}^{+}$. Likewise the convex hulls of these two sets are contained by their respective half-spaces (due to the convexity of half-spaces). However, this contradicts the claim that the convex hulls of S_1 and S_2 intersect since any point $\mathbf{x}$ in both convex hulls must then lie in both $H_{\mathbf{w}}^{+}$ and $H_{\mathbf{w}}^{-} = \mathcal{R} \setminus H_{\mathbf{w}}^{+}$ which is nonsense (these sets are by definition disjoint).

Hence, there does not exist a set of $d+1$ points shattered by $\mathcal{H}_d$. Of coarse, this implies that there does not exist any set of $n \geq d+1$ points which can be shattered by $\mathcal{H}_d$ (otherwise, any $d+1$ of these points could also be shattered). Consequently, $VC(\mathcal{H}_d) < d+1$.

Combining this result with the earlier result from (3-1) gives $d \leq VC(\mathcal{H}_d) < d+1 \Rightarrow VC(\mathcal{H}_d) = d$.

(3-3) (5pts) Suppose that there existed $n > d$ points $X = (\mathbf{x}^1, \ldots, \mathbf{x}^n)$ such that we could shatter the set $\Phi = (\phi(\mathbf{x}^1), \ldots, \phi(\mathbf{x}^n)) \subset \mathcal{R}^d$ with a decision rule of the form

$$h_\alpha(\phi) = \begin{cases} +1, & \alpha'\phi > 0 \\ -1, & \alpha'\phi \leq 0 \end{cases} \tag{40}$$

for some $\alpha \in \mathcal{R}^d$. This would contradict the result just shown in (3-2). Hence, there does not exist such a set and the VC-dimension of the set of these classifiers, based upon features $\phi : \mathcal{X} \to \mathcal{R}^d$, is at most d.

Part II: Decision Stumps

In this part, we consider the set of classifiers $\mathcal{H}$ on $\mathcal{R}^d$ comprised of *decision stumps*, e.g. decision rules of the form

$$h_{i,a,b}(\mathbf{x}) = \begin{cases} +1, & ax_i - b \geq 0 \\ -1, & ax_i - b < 0 \end{cases} \tag{41}$$

where $i \in \{1, ..., d\}$, $a \in \{-1, +1\}$ and $b \in \mathcal{R}$.

(3-4) (5pts) Let $X = (x^1, \dots, \mathbf{x}^n) \subset \mathcal{R}^d$. Show that these n points can be labeled in at most $2dn$ different ways using decision stumps. For each choice of $i \in \{1, \dots, d\}$ and $a \in \{-1, +1\}$, we can sweep b from $-\infty$ to $+\infty$ to generate (at most) n different labelings of X. Since there are $2d$ possible choices of i and a, this generates at most $2dn$ possible labelings of X.

(3-5) (5pts) Suppose that the VC-dimension of $\mathcal{H}$ is n. This means there exists a set X comprised of n points which $\mathcal{H}$ can shatter. For this set X we can generate any of the 2^n possible +1/-1 labelings of the n points in X. But, by the result of (3-4), we must have that $2^n \leq 2dn \Rightarrow n \leq \log_2 2dn$ (this holds for all X having n points including the X that we can shatter). To use the hint, write this as $n - \log_2 n \leq 1 + \log_2 d$. For $n \geq 3$, we have that $n/2 \leq n - log_2 n \leq 1 + \log_2 d \Rightarrow n \leq 2(1 + \log_2 d)$ which bounds the growth of the VC-dimension $n(d)$ as the dimension of the input space d becomes large.

(3-6) (optional) Solution omitted.

The last part of the problem is concerned with the set of classifiers

$$\mathcal{H}_m = \left\{ h_\alpha(x) = \text{sign}\left(\sum_{k=1}^{m} \alpha_k h(\mathbf{x}; \theta_k) \right) \middle| \, \alpha \in \mathcal{R}^m \right\} \tag{42}$$

comprised of decision rules formed by weighted combinations of m decision stumps where θ_k are the parameters of the k-th decision stump.

(3-7) (10pts) We compute an upper bound on the number of labelings of n points $X = \{\mathbf{x}^1, \dots, \mathbf{x}^n\} \subset \mathcal{R}^d$ we can generate with the set $\mathcal{H}_m$.

First, observe that choosing the parameters $\theta_1, \dots, \theta_m$ of the m decisions stumps essentially just allows us to generate some number K of distinct feature specifications $\Phi = (\phi(\mathbf{x}^1), \dots, \phi(\mathbf{x}^n))$ where each column $\phi(\mathbf{x}^k) = (h(\mathbf{x}^k; \theta_1), \dots, h(\mathbf{x}^k; \theta_m))'$. As was shown in (3-4), by varying θ_k we can generate at most $2dn$ possible labelings corresponding to row k of Φ. Since we can choose θ_k independently for each row, we can generate at most $(2dn)^m$ different +1/-1 matrices Φ. Hence, we can decompose

$$\mathcal{H}_m = \cup_{k=1}^{K} \mathcal{H}_{\Phi^k} \tag{43}$$

with $K \leq (2dn)^m$ and where $\mathcal{H}_{\Phi^k}$ is a linear decision rule of the form described in (3-3). For each $\mathcal{H}_{\Phi^k}$ we have that the VC-dimension is at most m. By the lemma, the number of possible labelings we can generate with $\mathcal{H}_{\Phi^k}$ (by varying α) for n points is bounded above by $(2n/m)^m$. Hence, we can generate at most $K(2n/m)^m \leq (2dn)^m(2n/m)^m$ labelings with the set of classifiers $\mathcal{H}_m$.