Stats 315A – HW2 Solutions

February 17, 2014

If there are any questions regarding the solutions or the grades of HW 2, please contact Austen (`ahead@stanford.edu`) with 'Stats315A-hw2-grading' in the subject line.

Grade Distribution: Total 100 Points

Problem 1: 14 [7 + 7]
Problem 2: 10 [5 + 5]
Problem 3: 15 [3 + 3 + 3 + 3 + 3]
Problem 4: 15 [4 + 5 + 6]
Problem 5: 16 [3 + 3 + 7+ 3]
Problem 6: 30 [4*5(4 points for each method) + 5 + 5]

Problem 1 (ESL Exercise 3.12 & 3.30)

ESL 3.12

We augment the original (centered) data matrix $\mathbf{X}$ with p additional rows $\sqrt{\lambda}\mathbf{I}$, and augment the original response $\mathbf{y}$ with p zeros. So, we have

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}. \tag{1}$$

Now note

$$\|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta\|_2^2 = \left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix} \beta \right\|_2^2 = \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_2^2. \tag{2}$$

So the ordinary least squares objective on the augmented data set is equivalent to the ridge regression objective on the original data set.

ESL 3.30

The solution is similar to 3.12. We augment the original (centered) data matrix $\mathbf{X}$ with p additional rows $\sqrt{\lambda\alpha}\mathbf{I}$, and augment the original response $\mathbf{y}$ with p zeros. So, we have

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda\alpha}\mathbf{I} \end{bmatrix}, \quad \tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}. \tag{3}$$

Also set $\gamma = \lambda(1-\alpha)$. Now we have

$$\begin{aligned} \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta\|_2^2 + \gamma\|\beta\|_1 &= \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\alpha\|\beta\|_2^2 + \gamma\|\beta\|_1 \\ &= \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\left[\alpha\|\beta\|_2^2 + (1-\alpha)\|\beta\|_1\right]. \end{aligned} \tag{4}$$

Thus minimizing the RHS over β is equivalent to minimizing the LHS over β, which is a lasso problem.

Problem 2 (ridge regression duplicate copy)

(a) With out loss of generality, assume $\|x\| = 1$. Consider the SVD of $X = [x; x]$. It is easy to show it is $U_{n\times 2}D_{2\times 2}V_{2\times t}^T$ with $U = [x; z]$,

$$U = [x; z]D = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & 0 \end{pmatrix} V = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{5}$$

where z is any unit vector orthoganal to x, i.e. any solution to $v^T x = 0$.

Now

$$\begin{aligned} \hat{\beta}_\lambda &= V(D^2 + \lambda I)^{-1} D U^T y \\ &= V_1 \frac{d_1}{d_1^2 + \lambda} x^T y \\ &= \frac{1}{2+\lambda} \begin{pmatrix} \hat{\beta}_* \\ \hat{\beta}_* \end{pmatrix} \end{aligned} \tag{6}$$

where $\hat{\beta}_* = x^T y$ is the OLS solution for y on x.

If there are M copies, z becomes a $M-1$ column matrix. $D = \mathbf{diag}(\sqrt{m}, 0, \cdots, 0)$ etc.

$$\hat{\beta}_\lambda = \frac{1}{m+\lambda}\left[\hat{\beta}_*, \cdots, \hat{\beta}_*\right]^T$$

(b) No. Consider the two variable case. The coefficients β, β^* minimize

$$||Y - X\beta - X^*\beta^*||_2^2 + \lambda\left(|\beta| + |\beta^*|\right) = ||Y - X(\beta + \beta^*)||_2^2 + \lambda\left(|\beta| + |\beta^*|\right)$$

Consider any minimizer of this. We can change β and β^* without changing their sum and leave the left side unchanged. The right side will not change either, as long as we do not change their sign.

Problem 3 (microarray data)

a)

$$\hat{\beta}_\lambda = (X^T X + \lambda I_p)^{-1} X^T y, \tag{7}$$

where $X^T X + \lambda I_p$ is the $10,000 \times 10,000$ matrix. The computations is of the order of $p^3 = 10^{12}$ operations.

b) The derivative condition is

$$(X^T X + \lambda I_p)\hat{\beta}_\lambda = X^T y. \tag{8}$$

which means

$$\lambda \hat{\beta}_\lambda = X^T (y - X\hat{\beta}_\lambda) \tag{9}$$

So $\alpha = \frac{1}{\lambda}\left(y - X\hat{\beta}_\lambda\right)$.

We then plug in $X^T\alpha$ for β in equation 8 to get

$$(X^T X + \lambda I_p) X^T \alpha = X^T y$$

Now premultiply by X on both sides to get

$$X(X^T X + \lambda I_p) X^T \alpha = X X^T y$$

ie

$$X X^T (X X^T + \lambda I_p)\alpha = X X^T y$$

or

$$\alpha = (X X^T + \lambda I_p)^{-1} y$$

Hence

$$\hat{\beta}_\lambda = X^T \alpha = X^T (X X^T + \lambda I_p)^{-1} y$$

c) Then use SVD of X to get

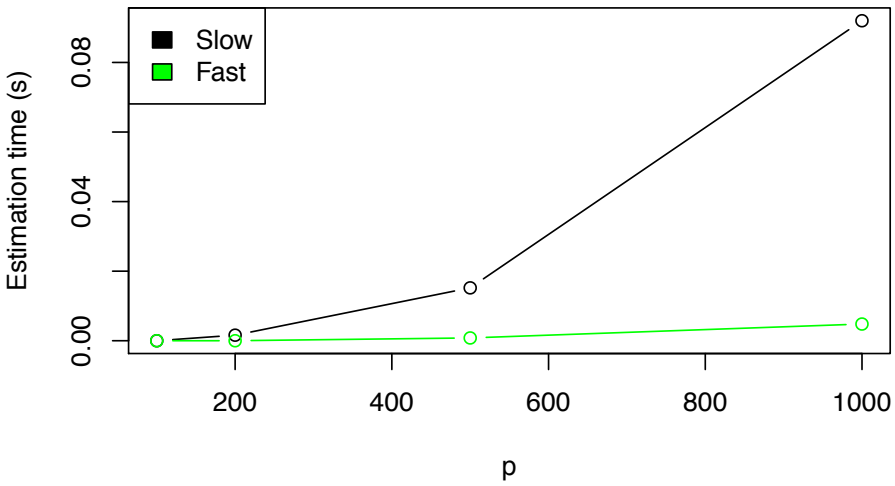
$$\hat{\beta}_\lambda = V D U^T (U D^2 U^T + \lambda I_p)^{-1} y = V D (D^2 + \lambda I)^{-1} U^T y$$

$$\hat{\theta}_\lambda = D(D^2 + \lambda I)^{-1} U^T y$$

Since D is a diagonal matrix, the inversion of $D^2 + \lambda I_n$ is trivial for all λ. Hence computing $\hat{\theta}_\lambda$ almost costs nothing once the SVD of X is done; and we can use it for all λ without any extra cost.

d) With a new observation x_0 we predict $x_0\hat{\beta}_\lambda$. We can optimize the regression fit by selecting a good λ using cross-validation.

e) The results are shown in the following figure, with the naive method in black and the efficient method in green. We see that the efficient method is much faster on large problems.



R-code:

```
lambda = 1
N=100

slow_fit = function(X,y,lambda){
  k = ncol(X)
  beta = solve(t(X)%*%X+lambda*diag(k))%*%t(X)%*%y
  return(beta)
}
fast_fit = function(X,y,lambda){
  k = nrow(X)
  Xsvd = svd(X)
  beta = Xsvd$v%*%solve(diag(Xsvd$d^2) +
      lambda*diag(k))%*%diag(Xsvd$d)%*%t(Xsvd$u)%*%y
  return(beta)
}

r = 10
p_vec = c(100,200,500,1000)
slow_time = numeric(4)
fast_time = numeric(4)
slow_time_sd = numeric(4)
fast_time_sd = numeric(4)
for (i in 1:4){
```

```
  p = p_vec[i]
  X = matrix(rnorm(N*p),nrow=N)
  y = matrix(rnorm(N),ncol=1)
  slow_time[i] = mean(replicate(r,system.time(slow_fit(X,y,lambda))[2]))
  fast_time[i] = mean(replicate(r,system.time(fast_fit(X,y,lambda))[2]))
}

pdf('p6-plot.pdf',width=6,height=4)
plot(p_vec, slow_time, xlab="p", ylab="Estimation time (s)", col='black', type="b")
lines(p_vec, fast_time, col='green', type="b")


legend("topleft",fill=c("black","green"),legend=c("Slow","Fast"))
dev.off()
```

'R-code'

Problem 4 (K-class logistic regression)

a) Add a constant c to β_{jk} for $k = 1, \cdots, K$, the probabilities will stay the same, hence the model is unidentifiable in its current form.

b) No we do not need to refit the model.

We simply rebase the model by subtracting all β's by β_1. i.e. $\beta_j^{new} = \beta_j - \beta_1$.

This concludes $\hat{\beta}_K^{new} = 0 - \hat{\beta}_1$.

The standard errors do change. Let Σ_j be the $K \times K$ covariance matrix for

$$\hat{\beta}_j = \begin{pmatrix} \hat{\beta}_{j,1} \\ \cdots \\ \hat{\beta}_{j,K} \end{pmatrix} \tag{10}$$

with last row and column 0.

Now

$$\hat{\beta}_{jl}^{new} = \hat{\beta}_{jl} - \hat{\beta}_{j1} = \left[-1, 0, \cdots, 0, \underset{l}{1}, 0, \cdots\right] = c_l^T \hat{\beta}_j$$

Hence $\hat{\beta}_j^{new} = C\hat{\beta}_j$, $\mathbf{Cov}(\hat{\beta}_j^{new}) = C\Sigma_j C^T$

The likelihood stays the same.

c) Adding a constant c to β_{jk} for $k = 1, \cdots, K$ does not affect the first term, but changes the second term. Hence a unique solution can be found for the penalized likelihood.

Assume $\hat{\beta}_j, j = 1, \cdots, P$ are the solution vectors of the penalized likelihood.

$$S(\mathbf{c}) = \sum_{j=1}^{P} \|\hat{\beta}_j + c_j \mathbf{1}_K\|_2^2$$

will achieve its minimum at $\mathbf{c} = \mathbf{0}$.

In other words, $\frac{\partial S}{\partial c_j} = 2(\hat{\beta}_j + c_j \mathbf{1}_K)^T \mathbf{1}_K = 0$ has solution $c_j = 0$.

We conclude $\hat{\beta}_j$ has mean 0, i.e.

$$\hat{\beta}_j^T \mathbf{1}_K = 0$$

Problem 5 (Kernel LDA)

a) Define $P_\phi = \left\{p : p = \sum_{n=1}^{N} a_i \phi(\mathbf{x}_n) \text{ for some } \mathbf{a}\right\}$ to be the space spanned by the training points. $P_{\phi^\perp} = \left\{p : p^T q = 0 \quad \forall q \in P_\phi\right\}$ to be the orthogonal complement of P_ϕ. If $p > N$, $P_{\phi^\perp}$ is nonempty. Since $\mathbf{m}_i^\phi \in P_\phi$, take any $w_0 \in P_{\phi^\perp}$, then $w_0^T \mathbf{S}_B^\phi = 0$ and $w_0^T \mathbf{S}_W^\phi = 0$.
Hence, for any w, we have

$$J(\mathbf{w} + \mathbf{w_0}) = \frac{(\mathbf{w} + \mathbf{w_0})^T \mathbf{S}_B^\phi (\mathbf{w} + \mathbf{w_0})}{(\mathbf{w} + \mathbf{w_0})^T \mathbf{S}_W^\phi (\mathbf{w} + \mathbf{w_0})} = \frac{\mathbf{w}^T \mathbf{S}_B^\phi \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^\phi \mathbf{w}} = J(\mathbf{w})$$

There cannot be a unique solution to $\arg\max J(\mathbf{w})$.

b)

$$\mathbf{w}^T \mathbf{m}_i^\phi = \sum_{j=1}^{N} \alpha_j \phi^T(\mathbf{x}_j) \times \frac{1}{N_i} \sum_{n=1}^{N_i} \phi(\mathbf{x}_n^i) = \sum_{j=1}^{N} \alpha_j \times (\frac{1}{N_i} \sum_{n=1}^{N_i} \phi^T(\mathbf{x}_j)\phi(\mathbf{x}_n^i)) = \alpha^T \mathbf{m}_i$$

c)

$$\begin{aligned} \mathbf{w}^T \mathbf{S}_B^\phi \mathbf{w} &= \mathbf{w}^T (\mathbf{m}_2^\phi - \mathbf{m}_1^\phi)(\mathbf{m}_2^\phi - \mathbf{m}_1^\phi)^T \mathbf{m} \\ &= \mathbf{w}^T (\mathbf{m}_2^\phi \mathbf{m}_2^{\phi T} - \mathbf{m}_2^\phi \mathbf{m}_1^{\phi T} - \mathbf{m}_1^\phi \mathbf{m}_2^{\phi T} + \mathbf{m}_1^\phi \mathbf{m}_1^{\phi T}) \mathbf{w} \\ &= \alpha^T (\mathbf{m}_2 \mathbf{m}_2^T - \mathbf{m}_2 \mathbf{m}_1^T - \mathbf{m}_1 \mathbf{m}_2^T + \mathbf{m}_1 \mathbf{m}_1^T) \alpha \\ &= \alpha^T (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \alpha \end{aligned} \tag{11}$$

Hence

$$\mathbf{M} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

From the definition of $\mathbf{m}_i, \mathbf{K}_i, \mathbf{1}_{N_i}$, it can be easily verified that

$$\mathbf{m}_i = \mathbf{K}_i \mathbf{1}_{N_i}$$

$$\mathbf{w}^T\phi(\mathbf{x}_l^i) = \sum_{n=1}^{N} \alpha_n \phi^T(\mathbf{x}_n)\phi(\mathbf{x}_l^i) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}_n, \mathbf{x}_l^i) = \alpha^T \mathbf{K}_i(\cdot, l)$$

and

$$\mathbf{w}^T\mathbf{w} = \alpha^T \mathbf{K} \alpha$$

where $\mathbf{K}(i,j) = k(\mathbf{x}_i, \mathbf{x}_j)$

Plugging into the denominator,

$$\begin{aligned}
\mathbf{w}^T\mathbf{S}_W^\phi\mathbf{w} &= \mathbf{w}^T \left(\sum_{i=1}^{2} \sum_{l=1}^{N_i} \left(\phi(\mathbf{x}_l^i) - \mathbf{m}_i^\phi\right) \left(\phi(\mathbf{x}_l^i) - \mathbf{m}_i^\phi\right)^T + \gamma\mathbf{I} \right) \mathbf{w} \\
&= \sum_{i=1}^{2} \sum_{l=1}^{N_i} \mathbf{w}^T \left(\phi(\mathbf{x}_l^i) - \mathbf{m}_i^\phi\right) \left(\phi(\mathbf{x}_l^i) - \mathbf{m}_i^\phi\right)^T \mathbf{w} + \gamma\mathbf{w}^T\mathbf{w} \\
&= \sum_{i=1}^{2} \sum_{l=1}^{N_i} \alpha^T \left(\mathbf{K}_i(\cdot, l) - \mathbf{m}_i\right) \left(\mathbf{K}_i(\cdot, l) - \mathbf{m}_i\right)^T \alpha + \gamma\alpha^T\mathbf{K}\alpha \\
&= \sum_{i=1}^{2} \alpha^T \left(\sum_{l=1}^{N_i} \mathbf{K_i}(\cdot, l)\mathbf{K_i}(\cdot, l)^T - \sum_{l=1}^{N_i} \mathbf{K}_i(\cdot, l)\mathbf{m}_i^T - \sum_{l=1}^{N_i} \mathbf{m}_i\mathbf{K}_i(\cdot, l)^T + N_i\mathbf{m}_i\mathbf{m}_i^T \right) \alpha + \gamma\alpha^T\mathbf{K}\alpha \\
&= \sum_{i=1}^{2} \alpha^T \left(\mathbf{K}_i\mathbf{K}_i^T - \mathbf{K}_i\mathbf{1}_{N_i}\mathbf{K}_i^T - \mathbf{K}_i\mathbf{1}_{N_i}\mathbf{K}_i^T + \mathbf{K}_i\mathbf{1}_{N_i}\mathbf{K}_i^T\right) \alpha + \gamma\alpha^T\mathbf{K}\alpha \\
&= \alpha^T \left(\sum_{i=1}^{2} \mathbf{K}_i(\mathbf{I} - \mathbf{1}_{N_i})\mathbf{K}_i^T + \gamma\mathbf{K} \right) \alpha
\end{aligned} \tag{12}$$

Hence

$$\mathbf{N} = \sum_{i=1}^{2} \mathbf{K}_i(\mathbf{I} - \mathbf{1}_{N_i})\mathbf{K}_i^T + \gamma\mathbf{K}$$

d) **Approach 1** Introduce the Lagrange multiplier λ and define the unconstrained optimization problem to be maximizing $L(\mathbf{w})$, where

$$L(\mathbf{w}) = \mathbf{w}^T\mathbf{S}_B^\phi\mathbf{w} + \lambda(\mathbf{w}^T(\mathbf{S}_W^\phi + \gamma\mathbf{I})\mathbf{w} - 1)$$

Taking derivatives w.r.t $\mathbf{w}$ and λ and setting to 0 we get

$$2\mathbf{S}_B^\phi\mathbf{w} + 2\lambda(\mathbf{S}_W^\phi + \gamma\mathbf{I})\mathbf{w} = 0 \tag{13}$$

Define Φ to be the $N \times p$ matrix of the training data points $\phi(x_i)$. Note that $\mathbf{m}_2^\phi - \mathbf{m}_1^\phi = C_B\Phi$ for some matrix C_B, Hence $\mathbf{S}_B^\phi = (\mathbf{m}_2^\phi - \mathbf{m}_1^\phi)^T(\mathbf{m}_2^\phi - \mathbf{m}_1^\phi) = \Phi^T C_B^T C_B \Phi = \Phi^T O_B \Phi$ where O_B

is an $N \times N$ matrix. Similarly, $\mathbf{S}_W^\phi = \Phi^T O_S \Phi$ for some $N \times N$ matrix O_S.

$$\begin{aligned} \mathbf{w} &= -\frac{1}{\lambda\gamma}(\mathbf{S}_B^\phi \mathbf{w} + \lambda \mathbf{S}_W^\phi \mathbf{w}) \\ &= -\frac{1}{\lambda\gamma}\Phi^T(O_B^\Phi \Phi\mathbf{w} + O_S^\Phi \Phi\mathbf{w}) \end{aligned} \tag{14}$$

which means $\mathbf{w} = \Phi^T \mathbf{w}_0$ for some $\mathbf{w}_0$, i.e. $\mathbf{w}$ must be in the space spanned by Φ.

Approach 2 Introduce the Lagrange multiplier λ and define the unconstrained optimization problem to be maximizing $L(\mathbf{w})$, where

$$L(\mathbf{w}) = \mathbf{w}^T \mathbf{S}_B^\phi \mathbf{w} + \lambda(\mathbf{w}^T(\mathbf{S}_W^\phi + \gamma\mathbf{I})\mathbf{w} - 1)$$

Taking derivatives w.r.t $\mathbf{w}$ and λ and setting to 0 we get

$$2\mathbf{S}_B^\phi \mathbf{w} + 2\lambda(\mathbf{S}_W^\phi + \gamma\mathbf{I})\mathbf{w} = 0 \tag{15}$$

Decompose $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$, where $\mathbf{w}_1 \in P_\phi$ and $\mathbf{w}_2 \in P_{\phi^\perp}$, we wish to prove that $\mathbf{w}_2 = 0$.

Plugging into 15 yields(dropping the constant 2)

$$\mathbf{S}_B^\phi(\mathbf{w}_1 + \mathbf{w}_2) + \lambda(\mathbf{S}_W^\phi + \gamma\mathbf{I})(\mathbf{w}_1 + \mathbf{w}_2) = 0$$

Notice $\mathbf{S}_B^\phi \mathbf{w}_2 = 0$ and $\mathbf{S}_W^\phi \mathbf{w}_2 = 0$,

$$\mathbf{S}_B^\phi \mathbf{w}_1 + \lambda(\mathbf{S}_W^\phi + \gamma\mathbf{I})\mathbf{w}_1 + \lambda\gamma\mathbf{w}_2 = 0$$

$\lambda\gamma \neq 0$, hence $\mathbf{w}_2 = 0$. i.e. $\mathbf{w} \in P_\phi$.

Problem 6 (zipcode classification)

Classification Method	Training Error	Test Error
(a) LDA on original 256 dimensional data	0.0159	0.0874
(b) LDA on the leading 64 principal components	0.0410	0.0935
(c) LDA on the leading 32 principal components	0.0575	0.0933
(d) LDA on filtered data	0.0336	0.0752
(e) Multiple linear logistic regression on filtered data	0.0000	0.0955

4 points each for fitting each classfication method and getting the right training and test error
For part (c), there are several ways that data centering could have been handled, which will give different training/test errors. Be lenient about this when grading (do not deduct points). Also if they solve for the old version of the problem set(doing pca on three classes separately), do not deduct points either.
For part (b), the centering of the test data should be done with respect to the centers from the training data, not the test data. -1 point if the centering is done using the centers from the test data.

Conclusion & Comments:

I. Among the 4 classifiers, method (d) has the best performance on the test set.

II. Compared to method (a), (b) increases both training error and test error. This suggests using principal components for feature selection in this context can be the wrong thing to do since the direction in the feature space that best separates the classes may have low variance (in which case we would be throwing out the very information we need).

III. The training error in (d) is higher than the training error in (a), but the test error in (d) is lower than the test error in (a). This suggests that lda in the original space (part a) is overfitting to the training set. By filtering, we reduce the model complexity and thereby decrease overfitting.

We don't actually ask for any of these conclusions, so do not reward/deduct points for them.

Filtering & Constrained GLMs

We compare fitting a constrained linear logistic model with the full set of pixels to fitting an unconstrained linear logistic model with the filtered features of procedure (c). i.e.

Consider the logistic linear model

$$f(x) = \frac{1}{1 + \exp(-x^T \beta)}$$

where we constrain the coefficient β to be piecewise constant in 2×2 blocks.

We wish to prove it is equivalent to

$$f(x) = \frac{1}{1 + \exp(-x_{filter}^T \theta)}$$

.

Let $h_k(i,j), k = 1, \cdots, 64$ be the Haar basis function over the image domain, where k is the index of the 64 blocks.

$$h_k(i,j) = \begin{cases} 1 & (i,j) \in \text{ block } k \\ 0 & (i,j) \notin \text{ block } k \end{cases} \tag{16}$$

Then the constrained coefficients $\hat{\beta}(i,j) = \sum_{k=1}^{64} h_k(i,j)\theta_k$. Rewrite each Haar basis matrix as vectors and stack the 64 basis vectors as a matrix $\underset{256\times 64}{H}$, we get

$$\hat{\beta} = \underset{256\times 64}{H}\theta$$

When predicting with the constrained coefficients,

$$x^T \hat{\beta} = xH\theta = x_{filter}\theta^*$$

The last equation holds if we reparametrize $\theta^* = 4\theta$.

Hence the two procedures are equivalent.

5 points for showing that the constrained model and the filtered model are equivalent.

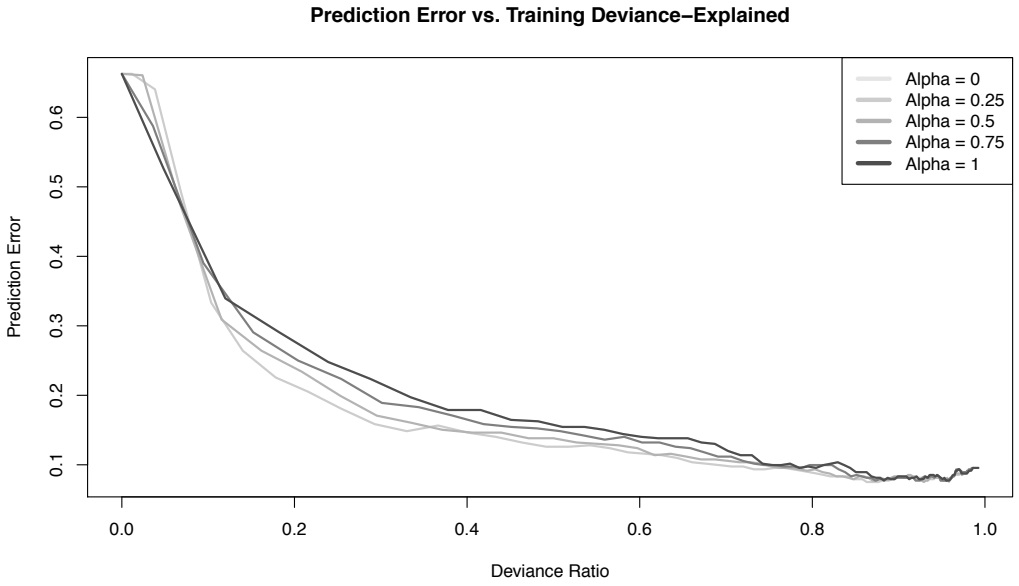


Figure 1: Test error vs % Deviance explained on the training data

Test Error vs Deviance explained on training data

The figure shows the plot of test error against the % deviance explained on the training data for $\alpha = 0,0.25,0.5,0.75$ and 1 respectively. Test-error is generally decreasing in % deviance explained. The rate of decrease slackens down around 40 % but isn't drastic enough to stop training at that level. Also, as the value of α decreases we have lower test error curves. This implies that all the filtered features are informative for prediction.

R-code:

```
library(MASS);
# load data if necessary:
if(sum(ls()=="train.2") + sum(ls()=="train.3") < 2){
path = "http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/zip.digits/"
train.3 = read.csv(paste(path,"train.3",sep=""),header=F)
train.5 = read.csv(paste(path,"train.5",sep=""),header=F)
train.8 = read.csv(paste(path,"train.8",sep=""),header=F)
}
xtrain = rbind(as.matrix(train.3),as.matrix(train.5),as.matrix(train.8))
ytrain = rep(c(3,5,8),c(nrow(train.3),nrow(train.5),nrow(train.8)))
test = as.matrix(read.table("zip.test"))
ytest = test[,1]
xtest = test[ytest==3 | ytest==5 | ytest==8,-1]
ytest = ytest[ytest==3 | ytest==5 | ytest==8]
# view digit as an image
showDigit <- function(x,...){
p=length(x)
```

```
d = sqrt(p)
if(d != round(d)) stop("Must be perfect square")
x=matrix(as.numeric(x),d,d)
image(x[,d:1],axes=F,...)
}
#output
#store training and test errors
pred_errors = matrix(0,nrow=5,ncol=2)
#part a ---------------------------------------------------------------------
l=lda(xtrain,ytrain)
pred_errors[1,1] = sum(predict(l)$class!=ytrain)/length(ytrain)
pred_errors[1,2] = sum(predict(l,xtest)$class!=ytest)/length(ytest)
#part b ---------------------------------------------------------------------
#centering based on the training data
xtrain.center<-apply(xtrain,2,mean);
xxtrain = scale(xtrain,center=xtrain.center,scale=F)
xxtest = scale(xtest,center=xtrain.center,scale=F)
V = svd(xxtrain)$v[,1:64]
pcstrain = xxtrain%*% V
pcstest = xxtest%*% V
l=lda(pcstrain,ytrain)
pred_errors[2,1] = sum(predict(l)$class!=ytrain)/length(ytrain)
pred_errors[2,2] = sum(predict(l,pcstest)$class!=ytest)/length(ytest)

#part c ---------------------------------------------------------------------
V = svd(xxtrain)$v[,1:32]
pcstrain = xxtrain%*% V
pcstest = xxtest%*% V
l=lda(pcstrain,ytrain)
pred_errors[3,1] = sum(predict(l)$class!=ytrain)/length(ytrain)
pred_errors[3,2] = sum(predict(l,pcstest)$class!=ytest)/length(ytest)

#part d ---------------------------------------------------------------------
filterdigit <- function(x){
  # averages each non-overlapping 2x2 block
  x = matrix(x,16,16)
  twos = rep(1:2,8)
  x = x[twos==1,]+x[twos==2,]
  x = x[,twos==1]+x[,twos==2]
  as.vector(x)/4
}
xtrain = t(apply(xtrain,1,filterdigit))
xtest = t(apply(xtest,1,filterdigit))
l = lda(xtrain,ytrain)
pred_errors[4,1] = sum(predict(l)$class!=ytrain)/length(ytrain)
pred_errors[4,2] = sum(predict(l,xtest)$class!=ytest)/length(ytest)

#part e ---------------------------------------------------------------------
library(glmnet)
l = glmnet(xtrain,factor(ytrain),family="multinomial")
pred_errors[5,1] = sum(as.numeric(predict(l,xtrain,s=l$lambda[99],type="class"))!=
    ytrain)/length(ytrain)
pred_errors[5,2] = sum(as.numeric(predict(l,xtest,s=l$lambda[99],type="class"))!=
    ytest)/length(ytest)
print(pred_errors)
#plot of deviance explained vs test err
```

```
alpha.values= c(0,.25,.5,.75,1);
#pred.err = matrix(0,length(alpha.values),100)
#dev.explained = matrix(0,length(alpha.values),100)
pred.err = list()
dev.explained = list()
for (i in 1:length(alpha.values)){
l= glmnet(xtrain,factor(ytrain),family="multinomial",alpha=alpha.values[i]);
dev.explained[[i]] = l$dev.ratio
pred.err[[i]] = as.numeric(apply(predict(l,xtest,type="class")!=ytest,2,mean));
}
my.colors = c("gray90","gray80","gray70","gray50","gray30","black")
pdf("p6-plot.pdf", width=10, height=6)
plot(dev.explained[[i]],pred.err[[i]],type="l",col=my.colors[1],lwd=2,
     xlab="Deviance Ratio",ylab="Prediction Error");
title("Prediction Error vs. Training Deviance-Explained")
for(i in 2:length(alpha.values)){
lines(dev.explained[[i]],pred.err[[i]],col=my.colors[i],lwd=2)
}
alpha.legend = character(length(alpha.values))
for(i in 1:length(alpha.values)){alpha.legend[i] = paste("Alpha =",alpha.values[i])}
legend(x="topright",legend = alpha.legend,lwd=4,col=my.colors)
dev.off()
#output prediction scores
print(round(pred_errors,4))
```

'R-code'