CS189–Fall 2015 — Solutions to Homework 4

October 25, 2015

Problem 1: Centering and Ridge Regression

(a) Given $J(\mathbf{w}, w_0) = (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0 * 1)^\top (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0 * 1) + \lambda \mathbf{w}^\top \mathbf{w}$, where design matrix X is n x m, w is m x 1, y is n x 1, 1 is a vector of n 1's. The full expansion of $J(\mathbf{w}, w_0)$ is as follows:

$$
\begin{aligned}
J(\mathbf{w}, w_0) &= \mathbf{y}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\mathbf{w} - \mathbf{y}^\top w_0 * 1 && (1)\\
&\quad - \mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} + \mathbf{w}^\top \mathbf{X}^\top w_0 * 1 && (2)\\
&\quad - 1^\top w_0^\top \mathbf{y} + 1^\top w_0^\top \mathbf{X}\mathbf{w} + 1^\top w_0^\top w_0 1 && (3)\\
&\quad + \lambda \mathbf{w}^\top \mathbf{w} && (4)
\end{aligned}
$$

Take derivatives with respect to scalar w_0, set to zero to find optimizer

$$
\begin{aligned}
0 &= -\mathbf{y}^\top 1 + \mathbf{w}^\top \mathbf{X}^\top 1 - 1^\top \mathbf{y} + 1^\top \mathbf{X}\mathbf{w} + 2w_0 * 1^\top 1 && (5)\\
0 &= -2\mathbf{y}^\top 1 + 2\mathbf{w}^\top \mathbf{X}^\top 1 + 2w_0 * 1^\top 1 && (6)\\
0 &= -\mathbf{y}^\top 1 + \mathbf{w}^\top \mathbf{X}^\top 1 + w_0 * 1^\top 1 && (7)
\end{aligned}
$$

Notice that $1^\top 1 = n$, $\mathbf{y}^\top 1 = \sum_{i=1}^{n} y_i$ and that $\mathbf{X}^\top 1 = 0$ since $\bar{x} = 0$. This is since $\bar{x} = \frac{1}{n}\sum x_i$, where i is the i-th row, implies that the sum of all the rows is zero. $X^\top 1$ sums all the rows in X. Thus,

$$
\sum_{i=1}^{n} y_i = 0 + nw_0 \Rightarrow \frac{1}{n}\sum_{i=1}^{n} y_i = w_0 \tag{8}
$$

Using the original $L(\mathbf{w}, w_0)$ function, take derivatives with respect to n x 1 vector $\mathbf{w}$.

$$0 = -\mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\mathbf{w} + \mathbf{X}^\top w_0 * 1 + \mathbf{X}^\top w_0 * 1 + 2\lambda I \mathbf{w} \tag{9}$$

$$0 = -2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\mathbf{w} + 2w_0 \mathbf{X}^\top * 1 + 2\lambda I \mathbf{w} \tag{10}$$

Remember that $\mathbf{X}^\top * 1 = 0$ since $\bar{x} = 0$

$$0 = -\mathbf{X}^\top \mathbf{y} + \mathbf{X}^\top \mathbf{X}\mathbf{w} + \lambda I \mathbf{w} \tag{11}$$

$$\mathbf{X}^\top \mathbf{y} = (\mathbf{X}^\top \mathbf{X} + \lambda I)\mathbf{w} \tag{12}$$

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top \mathbf{y} \tag{13}$$

(b) If you plotted the training vs. validation error for the simple linear regression, it was clear that our model complexity was low enough that overfitting would not be an issue. Thus, applying ridge regression actually did not help to improve validation error. Before applying regularization methods, it is important to check that our model is actually running the risk of overfitting. Otherwise, we are simply weakening the strength of our model without reason. As a result, the RSS should not have improved. At the optimal, the weights should either be close to the original model or slightly smaller.

Problem 2: Derivation of the Bonferroni Correction

Imagine you are playing a dice game with your friends. You roll 2 dice, and to win you must roll two sixes (anything else is a loss). You are a cheater and while your friends dont look, you roll the dice 3 times to increase your chances.

(a) $\frac{1}{36}$

(b) $1 - (1 - \frac{1}{36})^3$

(d) No. Finding one significant result in 5 methods with a certain p-value is not as significant as finding the same confidence out of one method.

(e) The probability of getting a given p-value at least once is $1 - (1 - pval)^m$ $1 - (1 - m * pval) = m * pval$.

(f) No. $1 - (1 - pval)^m = 1 - (1 - 0.0001)^5 0,000 = 0.9933$ (thats a really bad pvalue, small pvalues are good!) Bonferroni gives $d * pval = 50,000 * 0.0001 = 5 > 1$ (this is not possible for a probability!) The hypothesis $pval << 1$ is violated.

Problem 3: Independence vs. Correlation

(a) Essentially, there are 4 possible points (X, Y) can be, all with equal probability ($\frac{1}{4}$): $\{(0,1),(0,-1),(1,0),(-1,0)\}$, If graphed onto the Cartesian Plane, these point form "crosshairs".

To show that X and Y are uncorrelated, we need to prove:

$$E[(X - \mu_X)(Y - \mu_Y)] = E[X - \mu_X]E[Y - \mu_Y]$$

$$E[XY] = E[X]E[Y] = 0$$

Since for μ_X and μ_Y, we see that

$$E[X] = E[Y] = \frac{1}{2} * 0 + \frac{1}{2} * (\frac{1}{2} + \frac{-1}{2}) = 0$$

Notice for that whenever X is nonzero, Y is zero (vice versa). Thus, E[XY] = 0 since one of the terms is always zero, and we have shown that X and Y are uncorrelated. However, to show that X and Y are independent, we must show that:

$$P(X|Y) = P(X)$$

Unfortunately, this is not the case. $P(X = 0) = \frac{1}{2}$, but $P(X = 0|Y = 1) = 0$. Thus, X and Y are not independent.

(b) To prove independence for X and Y, we need to show that $P(X, Y) = P(X)P(Y)$. We know that $P(X|Y) = P(X)$ since B_1 is random 0-1 from the perspective of Y. Similar arguments can be made for all pairs of random variables. Thus, X, Y, and Z are pairwise independent.

However, $P(X|Y, Z) \neq P(X)$ since given the information in Y and Z, we can predict the value in X by the following relation. Thus, it is not mutually independent.

$$Y \oplus Z = (B_2 \oplus B_3) \oplus (B_1 \oplus B_3) = B_1 \oplus B_2 \oplus 0 = B_1 \oplus B_2 = X$$

(c) There are two places independence assumptions may arise: assuming independence in the sampling of our data, or assuming independence between the features of our data. We've talked more about the first case, which underlies more machine learning algorithms than feature independence. When we've derived Maximum Likelihood Estimates, we've always assumed that the data points are independent–which is why we can multiply their probabilities together. However, there are also machine learning algorithms that assume feature independence–e.g. Naive Bayes, one of the simplest generative classifiers.

As presented, the algorithms we've discussed in class (aside from Gaussian processes) have very poor performance on time series data.

Problem 4: Isocontours of Normal Distributions

(a) See appendix for graphs

Problem 5: Visualizing Eigenvectors of Gaussian Covariance Matrix

See appendix for graphs.
This is the full code for Q0, courtesy of Pat Virtue

```
% Sample the data

mu1 = 3;
mu2 = 4;
sigma1 = 3;
sigma2 = 2;

N = 100;

x1 = sigma1*randn(1,N) + mu1;
x2 = 1/2*x1 + sigma2*randn(1,N) + mu2;

X = [x1;x2];

% a) Compute the mean
mu = mean(X,2)

% b) Compute the covariance matrix
X_center = X-repmat(mu,[1 N]);
sigma = (X_center)*(X_center)'/N

% c) Compute the eigenvectors and eigenvalues
[V, D] = eig(sigma)
V = -[V(:,2) V(:,1)]
D = [D(2,2) 0; 0 D(1,1)];

% d) Plot the data and eigenvectors (and the mean)

figure
plot(x1, x2, 'k.', 'MarkerSize', 20)
axis image % To make it square
axis([-15 15 -15 15])

hold on
quiver(mu(1), mu(2), D(1,1)*V(1,1), D(1,1)*V(2,1), 'LineWidth', 4)
```

```
quiver(mu(1), mu(2), D(2,2)*V(1,2), D(2,2)*V(2,2), 'LineWidth', 4)

plot(mu(1), mu(2), 'gx', 'LineWidth', 4, 'MarkerSize', 20)

% e) Plot the data points rotated to the eigenvector axes

X_rot = V'*X_center;

figure
plot(X_rot(1,:), X_rot(2,:), 'k.', 'MarkerSize', 20)
axis image % To make it square
axis([-15 15 -15 15])
```

Problem 6: Covariance Matrixes and Decompositions

(a) We know that without loss of generality that the covariance matrix $\Sigma_X \in \mathbb{R}^{N,N}$ corresponding to random variable $X \in \mathbb{R}^N$ is positive semidefinite, which means that $\forall x \in \mathbb{R}^N$, $x^\top \Sigma x \geq 0$. Unfortunately, we require that Σ be positive definite ($\forall x \in \mathbb{R}^N$, $x^\top \Sigma x > 0$) in order for it to be invertible, since invertible matrices cannot have any eigenvalues be 0.

When might our covariance matrix Σ have an eigenvalue of 0? The most general case would be when one or more of the RV's are dependent on the others. We require our basis of eigenvectors to be independent in order for the covariance matrix to have no eigenvalues of 0. Another case would be when one or more of the RV's are deterministic.

For example, consider a random variable $X \in \mathbb{R}^N$ where the first $n-1$ elements are standard normal RV's, but the n-th element is deterministic (for example $X_n = 15$ always). Thus, all elements for the final row and column of Σ will be 0 since $cov(X_i, X_n) = 0 \ \forall i$, impying that Σ has an eigenvalue of 0

We can easily convert random variable X into a new random variable $X' \in \mathbb{R}^{N-1}$ by deleting the Nth dependent term, resulting in a positive definite covariance matrix $\Sigma_{X'}$ and invertible. This transformation does not result in any loss of information.

(b) Use the Spectral Decomposition Theorem to convert Σ into the following, where U is a unitary matrix of orthonormal eigenvectors $\vec{e_i} \ \forall i \in [0...N]$ and D is a diagonal matrix

with eigenvalues $\lambda_i \ \forall i \in [0...N]$ located at indices corresponding to eigenvectors in U. Note: all eigenvalues > 0 since Σ is positive definite

$$\Sigma = UDU^\top \Rightarrow \Sigma^{-1} = (UDU^\top)^{-1} = (U^\top)^{-1}D^{-1}U^{-1} = UD^{-1}U^\top \tag{14}$$

This is since a unitary matrix U is such that $U^{-1} = U^\top$. Note that if diagonal matrix D has values $d_{i,i} \ \forall i$, then D^{-1} has value $\frac{1}{d_{i,i}} \ \forall i$. Once again, since Σ was positive definite, the value $\frac{1}{d_{i,i}}$ exists.

Now, we decompose D^{-1} into its square-root by defining Q as a diagonal matrix with diagonal values $\frac{1}{\sqrt{d_{i,i}}}$. Verify that $QQ = D^{-1}$ and that $Q^\top = Q$. Thus, we have:

$$\Sigma^{-1} = UD^{-1}U^\top = UQQU^\top = UQQ^\top U^\top \tag{15}$$

$$\Sigma^{-1} = A^\top A \tag{16}$$

Where we defined $(UQ)^\top = A$. Therefore

$$x^\top \Sigma^{-1} x = x^\top A^\top A x = (Ax)^\top (Ax) = \|Ax\|_2^2 \tag{17}$$

Note: This process is closely related to Cholesky Decomposition, which will require one to use QR Decomposition to show that $\Sigma^{-1} = LL^\top$ for all invertible covariance matrices Σ where L is a lower triangular matrix.

(c) $x^\top \Sigma^{-1} x$ is a scalar written in vector quadratic form. It looks like an incomprehensible value, but when we convert it to $\|Ax\|_2^2$, we see that in reality its just the squared L2 norm of Ax, which measures the squared distance from the data vector x from the mean (in this case 0). Note that we can change the mean to be any arbitrary value without loss of generality.

(d) Recall from Part B our decomposition for Σ^{-1}, which was as follows where U is a unitary matrix, D is a diagonal matrix.

$$\Sigma^{-1} = UD^{-1}U^\top = A^\top A \tag{18}$$

Note that $\|x\|_2 = 1$ and $\|Ux\|_2 = 1$ since unitary matrices are orthonormal and preserve magnitude. Define $q = Ux$, we have

$$\|Ax\|_2^2 = x^\top A^\top A x = x^\top U D^{-1} U^\top x = q^\top D^{-1} q \tag{19}$$

We can choose our x such that q will be any Euclidean Basis Vector $\vec{e_i}$ such that the ith element is 1 and all other elements are 0. Therefore, the maximum value that $\|Ax\|_2^2$ is $\frac{1}{\lambda_i}$, where λ_i is the minimum eigenvalue of Σ. The minimum value that

$\|Ax\|_2^2$ is $\frac{1}{\lambda_j}$, where λ_j is the maximum eigenvalue of Σ.

If we have $X_i \perp\!\!\!\perp X_j \ \forall i, j$, then $cov(X_i, X_j) = 0 \ \forall i, j$ meaning that off diagonal terms for Σ are 0. Thus, we can find Σ^{-1} directly, where

$$\Sigma_{i,j}^{-1} = \begin{cases} \frac{1}{\sigma_i^2} \text{ if } i == j \\ 0 \text{ else} \end{cases}$$

Therefore, if we have $X_i \perp\!\!\!\perp X_j \ \forall i, j$, the maximum value that $\|Ax\|_2^2$ is $\frac{1}{\sigma_i^2}$, where σ_i^2 is the minimum variance. The minimum value of $\|Ax\|_2^2$ is $\frac{1}{\sigma_j^2}$, where σ_j^2 is the maximum variance.

To maximize $f(X)$, we want the superscript above the exponent to be minimal since there is a negative sign. Thus, for $\|Ax\|_2^2$ to be minimal, we want to choose x to be the vector corresponding to the eigenvector corresponding to the maximal eigenvector λ_j or maximum variance σ_j^2 if independent.

Problem 7: Gaussian Classifiers For Digits

(a) Say we have i.i.d observations $X_1 ... X_n$, then the MLEs for a multivariate Gaussian distribution are as follows

MLE of $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X$, MLE of $\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})'$

Note that the MLEs are just the sample mean and covariances. The MLE of $\hat{\mu}$ is unbiased, but the MLE of $\hat{\Sigma}$ is biased, with $E[\hat{\Sigma}] = \frac{n-1}{n}\Sigma$. In order to get an unbiased sample, use $\frac{1}{n-1}$ instead of $\frac{1}{n}$.

(b) For each training set, count all the labels belonging to set θ_i. Note that there are 10 sets in total (numbers 0...9). Divide the count for a particular set θ_i by the total across all sets. One can thus view the distribution as the sample prior (empirically, how many of a class have we seen in the total data set).

$$P(\theta_i) = \frac{\theta_i}{\sum_{j=1}^{10} \theta_j}$$

(c) I visualized the covariance matrix for the "zero" class, for the 10000 image set from train-small.mat (See the appendix for details). Upon visualization through imagesc, the 784 x 784 matrix appears to show an X of positive values in the center, with negative values along the edges of the X. The X is enclosed by a 150 zeros on every

side. Note, the values themselves are extremely close to zero, about $+/-10^{-4}$. Also, the non-zero values in the visualization appear to be pixelated, as in the X seems to be formed by dots of non-zero values on a zero canvas.

The interpretation for the band of zeros surrounding the X (and for zeros dispersed the non-zero area) is that pixel values at those locations from 1 to 784 never vary, in which case it is likely that one of the pixels is always zero. The reason the image looks pixelated is because we had to convert a 28 x 28 image to 1 x 784: the resulting row vector of pixels will have slices of non-zero values representing the original digits non-zero pixels (the rest of the vectors will be zeros). Thus, taking a covariance of these row vectors will result in pixelation patches within the overall covariance matrix. Finally, the non-zero indices represent the density/most likely areas for the image to have non-zero pixels.

(d) (i) Since the multivariate Gaussians share the same covariance matrix Σ_{avg}, the decision boundary is linear since the covariance matrices will cancel out.

With the 100 data point training set, the algorithm achieved a 51.1% error rate. With the 10000 data point training set, the algorithm achieved a 11.72% error rate. See appendix for full graph.

(ii) In this case, the multivariate Gaussians do not share the same covariance matrix Σ_i, thus the decision boundary is quadratic since the covariance matrices will not cancel out. Essentially, the cross terms will remain (with maximum degree of two), and this will curve the decision boundary.

With the 100 data point training set, the algorithm achieved a 21.2% error rate. With the 10000 data point training set, the algorithm achieved a 7.67% error rate. See appendix for full graph.

(iii) Part b generates a significantly superior error rate. This is because it does not make the assumption that all classes share the same covariance matrix, and allows the decision boundary to properly mimic the optimal class choice (instead of approximating using a linear boundary). This comes at the cost of calculation efficiency, as now the computer has to calculate cross terms.

Appendix

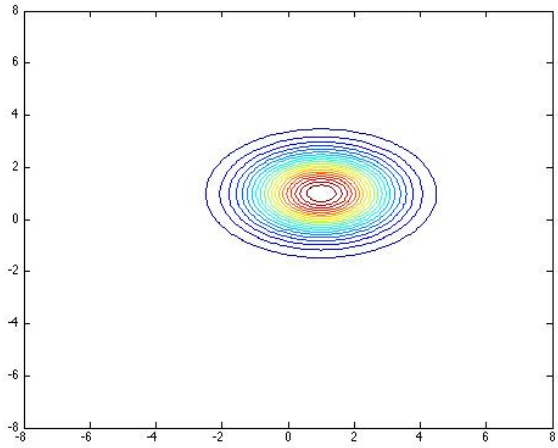

Figure 1: Problem 4a

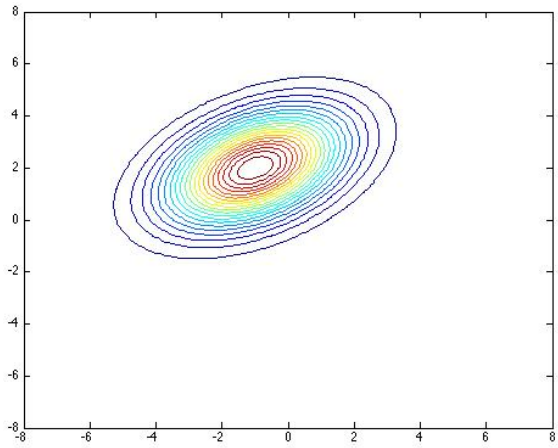

Figure 2: Problem 4b

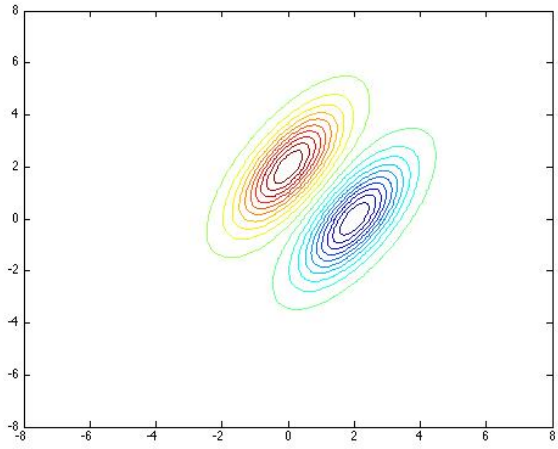

Figure 3: Problem 4c

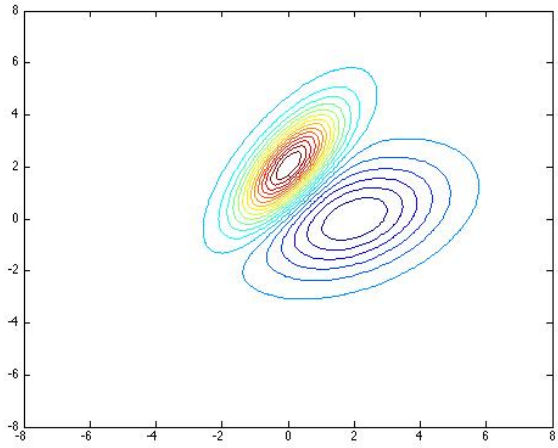

Figure 4: Problem 4d

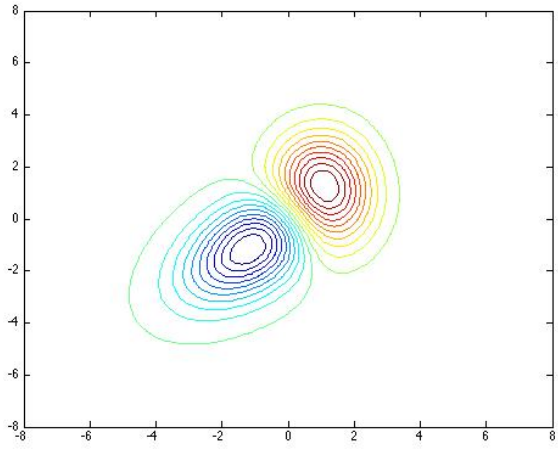

Figure 5: Problem 4e

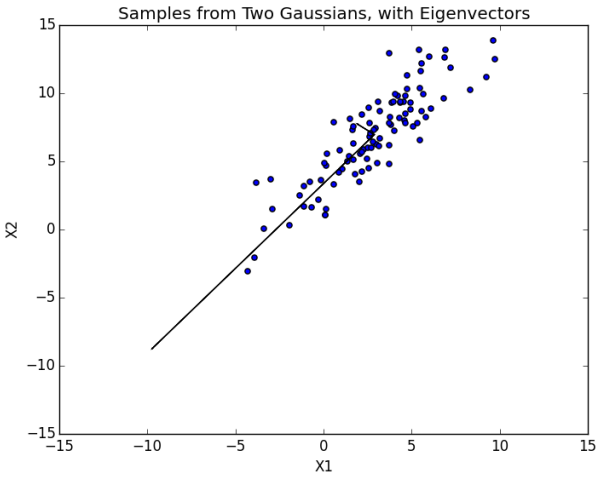

Figure 6: Problem 5d

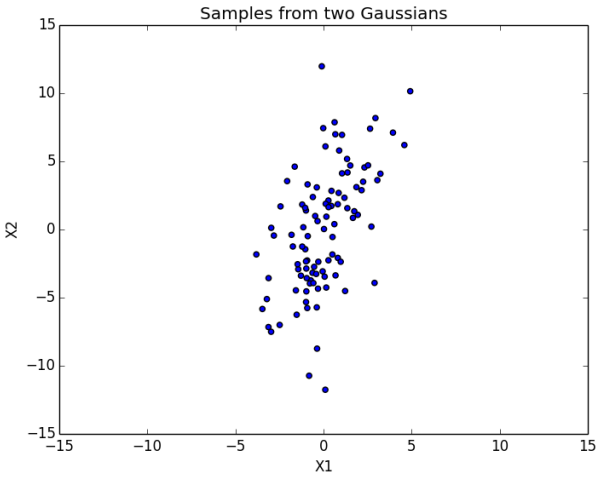

Figure 7: Problem 5e

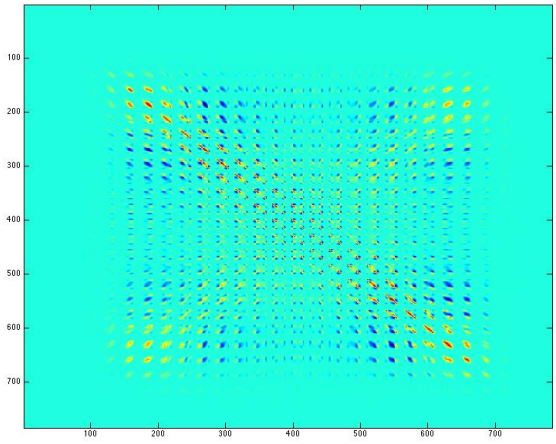

Figure 8: Problem 7iii

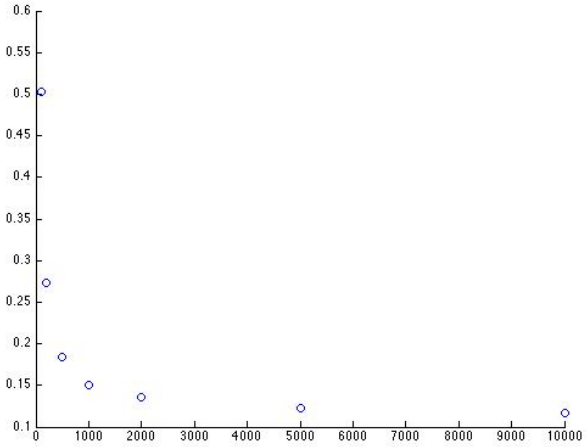

Figure 9: Problem 7iv-a: Using an average covariance matrix across classes per data set

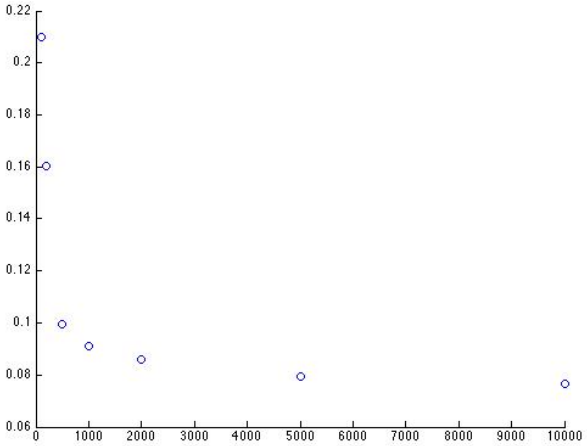

Figure 10: Problem 7iv-b: Using specific covariance matrices