MOCK Final Exam

Exam Duration: 2 Hours
Total Marks Available: 95

Instructions

1. This is a closed-book exam.
2. All answers and explanations must be provided in the answer book. This is also true of Question 2 which requires you to fit LDA and QDA models using the computer.
3. Keeps your answers succinct and to the point. Long rambling answers with irrelevant details will work against you.

Advice: Read the entire exam carefully before starting on your answers.

Question 1. (10 marks)

Suppose we want to solve for the optimal λ in the LASSO regression

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 - \lambda \|\mathbf{x}\|_1$$

via cross-validation when we have a total of pn IID training points. The typical way to do this as follows:

- Choose a set $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$ of possible choices for λ.
- Randomly split the data into p folds: $D_1, \ldots, D_p$ where each fold contains n datapoints.
- For each $\lambda_i \in \Lambda$, and $k = 1, \ldots, p$, train the Lasso classifier on $\{D_1, \ldots, D_{k-1}, D_{k+1}, \ldots, D_p\}$ and test on D_k to obtain the k^{th} sample error, $\mathcal{E}_k(\lambda_i)$
- Use the p values to obtain the mean error $\mathcal{E}(\lambda_i) := \frac{1}{p} \sum_{k=1}^{p} \mathcal{E}_k(\lambda_i)$ for $i = 1, \ldots, m$.
- Let i^* be the value of i that corresponds to the minimum mean error, i.e. $i^* = \text{argmin}_{i \in \{1,\ldots,m\}} \mathcal{E}(\lambda_i)$

Given this procedure answer the following questions:

(a) Assuming the data $D = \cup D_k$ was IID, is $\mathcal{E}(\lambda_i)$ an unbiased estimate of the true error for λ_i when $(p-1)n$ IID training points are used to learn the regression function? Justify your answer. (5 marks)

Solution: Yes, since for each i and k, $\mathcal{E}_k(\lambda_i)$ is an unbiased estimate of $\mathcal{E}(\lambda_i)$. This follows from the IID assumption and in particular since D_k played no role in the training that led to $\mathcal{E}_k(\lambda_i)$.

(b) Is $\mathcal{E}(\lambda_{i^*})$ an unbiased estimate of the true error of λ_{i^*}? Justify your answer. (5 marks)

Solution: No. The act of selecting i^* has introduced a bias and in fact $\mathcal{E}(\lambda_{i^*})$ will be biased downwards. The only way to get an unbiased estimate of $\mathcal{E}(\lambda_{i^*})$ is to test it on a holdout test set that was *never* used at any stage of the training process.

Question 2. (15 marks)

Open the R Notebook *LDA_v_QDA_Mock_Exam.rmd*. The first code chunk reads in two separate data-sets. The first set is the training data-set and consists of 1,000 observations of $(y, \mathbf{x})$ where $y \in \{1, 2\}$ is the target / class and $\mathbf{x} \in \mathbb{R}^8$ is a feature vector. The target y is in the first column and is labelled "class" in the training set data-frame. The test-set contains 9,000 observations of the same random vector $(y, \mathbf{x})$. (The second code chunk provides a summary of the training data while the third chunks prints out the first few rows of the training data.)

(a) Fit an LDA model to the training data. Report the confusion matrix of your fitted model on the test set. (5 marks)

Solution: The confusion matrix of the fitted LDA model on the test set is

		True	
		1	**2**
Predicted	**1**	4321	194
	2	162	4323

with a resulting error rate of 3.96%.

(b) Fit a QDA model to the training data. Report the confusion matrix of your fitted model on the test set. (5 marks)

Solution: The confusion matrix of the fitted QDA model on the test set is

		True	
		1	**2**
Predicted	**1**	4316	190
	2	167	4327

with a resulting error rate of 3.97%.

Question 3. (5 marks)

Indicate which of (a) through (d) is correct and be sure to justify your answer. (Just one or two lines is sufficient.) The lasso, relative to ordinary least squares, is:

(a) More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

(b) More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

(c) Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

(d) Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

Solution: (c) is correct. The penalty term in the Lasso makes it less flexible than least squares. This causes an increase in bias but a decrease in variance.

Question 4. (20 marks)

Consider the situation where we have data from two classes – class 0 and class 1. We make the following mixture model for data from class 0 which places a normal density on each data-point:

$$\begin{aligned} p(\mathbf{x} \mid c=0) &= \frac{1}{N_0} \sum_{n \in \text{class } 0} \mathrm{N}\left(\mathbf{x} \mid \mathbf{x}_n, \sigma^2 \mathbf{I}\right) \\ &= \frac{1}{N_0} \frac{1}{(2\pi\sigma^2)^{d/2}} \sum_{n \in \text{class } 0} e^{-(\mathbf{x}-\mathbf{x}_n)^\top (\mathbf{x}-\mathbf{x}_n)/(2\sigma^2)} \end{aligned}$$

where d is the dimension of a data-point $\mathbf{x}$ and N_0 is the number of training points in class 0. This is a so-called *Parzen estimator* and it models the data as a uniform weighted sum of normal distributions centred on the training points.

Similarly, for data from class 1 we assume

$$p(\mathbf{x} \mid c=1) = \frac{1}{N_1} \frac{1}{(2\pi\sigma^2)^{d/2}} \sum_{n \in \text{class } 1} e^{-(\mathbf{x}-\mathbf{x}_n)^\top (\mathbf{x}-\mathbf{x}_n)/(2\sigma^2)}.$$

(a) Explain how you would classify a new point $\mathbf{x}^*$? (10 marks)

Solution: To classify a new point $\mathbf{x}^*$, we would use Bayes rule

$$p(c=0 \mid \mathbf{x}^*) = \frac{p(\mathbf{x}^* \mid c=0)p(c=0)}{p(\mathbf{x}^* \mid c=0)p(c=0) + p(\mathbf{x}^* \mid c=1)p(c=1)}.$$

We obviously can get a similar expression for $p(c=1 \mid \mathbf{x}^*)$. We can estimate $p(c=0)$ and $p(c=1)$ by $N_0/(N_0+N_1)$ and $N_1/(N_0+N_1)$, respectively. To see which class is most likely we could use the ratio

$$\frac{p(c=0 \mid \mathbf{x}^*)}{p(c=1 \mid \mathbf{x}^*)} = \frac{p(\mathbf{x}^* \mid c=0)p(c=0)}{p(\mathbf{x}^* \mid c=1)p(c=1)}$$

and classify to class 0 if this ratio is greater than 1. Otherwise we classify to class 1.

(b) Explain clearly how would you choose an appropriate value of σ^2? (5 marks)

Solution: We could select σ^2 using cross-validation.

(c) What sort of classifier would you obtain in the limit $\sigma^2 \to 0$? (It's ok to guess!) (5 marks)

Solution: The classifier becomes the 1-nearest neighbor classifier in the limit as $\sigma^2 \to 0$. The intuition is that as $\sigma \to 0$ the normal densities become more and more peaked at their means, i.e. at the training points, and the influence of the closest point in each class begins to dominate the influence of all the other training points. This question was really testing your intuition as to what this classifier is doing.

One can prove the result more carefully as follows (but you weren't required to do this). Let n_0 denote the index of the point in class 0 that is closest to $\mathbf{x}^*$. Similarly let n_1 denote the index of the point in class 1 that is closest to $\mathbf{x}^*$. Then for very small σ^2 it can be seen (by considering limits) that

$$\frac{p(c=0 \mid \mathbf{x}^*)}{p(c=1 \mid \mathbf{x}^*)} \approx \frac{e^{-(\mathbf{x}^*-\mathbf{x}^{n_0})/(2\sigma^2)} p(c=0)/N_0}{e^{-(\mathbf{x}^*-\mathbf{x}^{n_1})/(2\sigma^2)} p(c=1)/N_1} = \frac{e^{-(\mathbf{x}^*-\mathbf{x}^{n_0})/(2\sigma^2)}}{e^{-(\mathbf{x}^*-\mathbf{x}^{n_1})/(2\sigma^2)}}.$$

In the limit as $\sigma^2 \to 0$ this ratio converges to 0 or 1 and we classify (with certainty) to class 0 if $\mathbf{x}^*$ is closer to $\mathbf{x}_{n_0}$ than to $\mathbf{x}_{n_1}$. Otherwise we classify to class 1.

Question 5. (10 marks)

Consider the following process for training a binary classifier $(\mathbf{x}, k)$ where $\mathbf{x} \in \mathbb{R}^2$ and $k \in \{1, 2\}$ is the class. We first plot the data in $\mathbb{R}^2$ and note that it is not even approximately linearly separable. We then perform a simple transformation $\mathbf{x} \to \mathbf{z} \in \mathbb{R}^2$ and note that the transformed data is now close to being linearly separable. We therefore decide to use the transformed data to construct our classifier. Towards this end we split the data, $\mathbf{z}_1, \ldots, \mathbf{z}_n$, into a training set and a test set. We train a linear classifier on the training set and then use the test set to estimate its generalization error.

What, if anything, is "wrong" with this learning process? If you think nothing is "wrong" explain why you think the learning process is sound.

Solution: Generalization error corresponds to the expected error over "unobserved" or new data observations. One approximates the generalization error by the average error over the

test data. For this approximation to be unbiased, it is essential one not use *any* information from the test data. In the process outlined above, we transform the data *after* looking at the entire data set, including the portion that we later keep aside as test data.

This will bias the test error low since some of the test points will likely have influenced the choice of the linear transformation. Consider the case where the training data was linearly separable in the original set of co-ordinates; however, the training plus test data was not. If we choose the transformation only as a function of the training data then we might have chosen the identity transformation, i.e. we might not have transformed the data at all. But if we looked at the training plus test data to choose a transformation then we would most likely have chosen a non-linear one.

Question 6. (15 marks)

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ denote (centered) sample observations of $\mathbf{x} \in \mathbb{R}^d$ and suppose we wish to solve the following problem:

$$\begin{aligned} & \min_{\mathbf{B},\mathbf{Z}} \sum_{i=1}^{n} \sum_{j=1}^{d} \left[x_{j,i} - \sum_{k=1}^{K} b_{j,k} z_{k,i} \right]^2 \\ \equiv \quad & \min_{\mathbf{B},\mathbf{Z}} \|\mathbf{X} - \mathbf{B}\mathbf{Z}\|_F^2 \end{aligned} \tag{1}$$

where $\mathbf{X} = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n]$ is a $(d \times n)$ matrix whose columns are the n sample observations.

Fact: It is well known that an optimal solution to (1) is

$$\begin{aligned} \mathbf{B}^* &= \mathbf{\Gamma}_{1:K} \\ \mathbf{Z}^* &= \mathbf{P}_{1:K} = [\mathbf{p}_{1:K}^1 \ \cdots \ \mathbf{p}_{1:K}^n] \end{aligned}$$

where:

- $\mathbf{\Gamma}_{1:K}$ contains the K eigen vectors of $\mathbf{\Sigma}$, the sample variance-covariance matrix of the $\mathbf{x}_i$'s, corresponding to the K largest eigen values
- $\mathbf{p}_{1:K}^i$ are the first K principal components for the i^{th} data-point $\mathbf{x}_i$.

This is simply stating that the familiar PCA approximation $\mathbf{X} \approx \mathbf{\Gamma}_{1:K}\ \mathbf{P}_{1:K}$ is the optimal solution to (1).

Consider now the matrix completion problem where we only have partial information on the entries of the $d \times n$ matrix $\mathbf{X}$. Specifically, let $\Omega \subseteq \{1, \ldots, d\} \times \{1, \ldots, n\}$ denote the observed entries of $\mathbf{X}$. We want to find matrices $\mathbf{B}, \mathbf{Z}$ to solve

$$\equiv \quad \min_{\mathbf{B},\mathbf{Z}} \sum_{(u,i) \in \Omega} \left(x_{ui} - \sum_{k=1}^{K} b_{uk} z_{ki} \right)^2 . \tag{2}$$

This is a non-convex optimisation problem so we can only hope in general to find good local optimal solutions to it.

(a) Explain how you could use the above fact to design an iterative algorithm to find a solution to (2) and therefore impute / estimate the missing values in $\mathbf{X}$. (To be clear, no equations are required here or in part (b) below and just a few lines will suffice.) (8 marks).

Solution: Start off with an initial guess of the unknown values x_{ui} for $(u,i) \notin \Omega$. Let $\mathbf{X}_0$ be the complete matrix containing these guesses as well as the known observed values. Now run a PCA (justified by the fact above) so that $\mathbf{X}_0 \approx \mathbf{\Gamma}^0_{1:K}\,\mathbf{P}^0_{1:K}$. Use $\mathbf{\Gamma}^0_{1:K}\,\mathbf{P}^0_{1:K}$ to update your guesses of x_{ui} for $(u,i) \notin \Omega$. Now repeat the procedure and continue for a fixed number of iterations or until your guesses x_{ui} for $(u,i) \notin \Omega$ converge.

(b) Explain how you would use a least-squares algorithm to find a solution to (2) and therefore impute / estimate the missing values in $\mathbf{X}$. (7 marks).

Solution: An (alternating) least squares approach to solving (2) is to select an initial $\hat{\mathbf{B}}$ and then iterate the following two steps until convergence:

(i) Optimize over $\mathbf{Z}$ for given $\hat{\mathbf{B}}$. Let $\hat{\mathbf{Z}}$ be the optimal solution.

(ii) Optimize over $\mathbf{B}$ for given $\hat{\mathbf{Z}}$. Let $\hat{\mathbf{B}}$ be the optimal solution.

Note that steps (i) and (ii) both require the solution of a least squares problem. This algorithm will converge and we can impute / estimate the missing values of $\mathbf{X}$ using the corresponding values of $\mathbf{B}^*\mathbf{Z}^*$ where $\mathbf{B}^*$ and $\mathbf{Z}^*$ are the matrices you obtain after the algorithm converges.

Question 7. (20 marks)

Show that the K-means clustering algorithm can be kernelized, i.e. is amenable to the kernel trick. To get you started, let $\phi(\mathbf{x}) \in \mathbb{R}^M$ be the feature vector where $\mathbf{x} \in \mathbb{R}^m$. (Typically $M >> m$ with possible $M = \infty$.) Recall that the k^{th} cluster center at the end of iteration t satisfies

$$\mathbf{m}_k^{(t)} := \underset{\mathbf{m}\in\mathbb{R}^M}{\operatorname{argmin}} \sum_{j\in\mathcal{C}_k^{(t)}} \|\phi(\mathbf{x}_j) - \mathbf{m}\|_2^2 = \frac{1}{|\mathcal{C}_k^{(t)}|} \sum_{j\in\mathcal{C}_k^{(t)}} \phi(\mathbf{x}_j)$$

for $k = 1,\ldots,K$ where K is the number of clusters and $|\mathcal{C}_k^{(t)}|$ is the number of points (in iteration t) assigned to cluster k. The cluster assignment of data-point i in iteration $t+1$ of the algorithm is

$$\mathcal{C}^{(t+1)}(i) = \underset{k}{\operatorname{argmin}} \left\| \phi(\mathbf{x}_i) - \mathbf{m}_k^{(t)} \right\|_2^2.$$

Solution: The cluster assignment of data-point i in iteration $t+1$ of the algorithm is

$$\begin{aligned}
\mathcal{C}^{(t+1)}(i) &= \operatorname*{argmin}_k \left\| \phi(\mathbf{x}_i) - \mathbf{m}_k^{(t)} \right\|_2^2 \\
&= \operatorname*{argmin}_k \left\| \phi(\mathbf{x}_i) - \frac{1}{|\mathcal{C}_k^{(t)}|} \sum_{j \in \mathcal{C}_k^{(t)}} \phi(\mathbf{x}_j) \right\|_2^2 \\
&= \operatorname*{argmin}_k \left(\phi(\mathbf{x}_i) - \frac{1}{|\mathcal{C}_k^{(t)}|} \sum_{j \in \mathcal{C}_k^{(t)}} \phi(\mathbf{x}_j) \right)^\top \left(\phi(\mathbf{x}_i) - \frac{1}{|\mathcal{C}_k^{(t)}|} \sum_{j \in \mathcal{C}_k^{(t)}} \phi(\mathbf{x}_j) \right) \\
&= \operatorname*{argmin}_k \left\{ \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_i) - \frac{2}{|\mathcal{C}_k^{(t)}|} \sum_{j \in \mathcal{C}_k^{(t)}} \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) + \frac{1}{|\mathcal{C}_k^{(t)}|^2} \left(\sum_{j \in \mathcal{C}_k^{(t)}} \phi(\mathbf{x}_j) \right)^\top \left(\sum_{j \in \mathcal{C}_k^{(t)}} \phi(\mathbf{x}_j) \right) \right\} \\
&= \operatorname*{argmin}_k \left\{ K(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{|\mathcal{C}_k^{(t)}|} \sum_{j \in \mathcal{C}_k^{(t)}} K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{|\mathcal{C}_k^{(t)}|^2} \sum_{j,\ell \in \mathcal{C}_k^{(t)}} K(\mathbf{x}_j, \mathbf{x}_\ell) \right\} \qquad (3)
\end{aligned}$$

where $K(\mathbf{x}_i, \mathbf{x}_\ell) := \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_\ell)$. The importance of (3) is that the cluster assignment can be computed without reference to ϕ. In particular, only the kernel function $K(\cdot, \cdot)$ is required! The kernel K-means clustering algorithm proceeds by computing (3) for all training points $i = 1, \ldots, N$ in iteration $t+1$, and then computing the $|\mathcal{C}_k^{(t+1)}|$'s before proceeding to iteration $t+2$. We continue iterating until convergence (to a local minimum) occurs.

Note that while ϕ is indeed required to compute the $\mathbf{m}_k^{(t)}$'s you don't actually need the $\mathbf{m}_k^{(t)}$'s to do K-means clustering!