SB2.2 Statistical Machine Learning

Past papers.

A15677W1

2018

SECOND PUBLIC EXAMINATION

Honour School of Mathematics and Statistics Part B: Paper SB2b

Honour School of Mathematics Part B: Paper SB2b

STATISTICAL MACHINE LEARNING

TRINITY TERM 2018

Wednesday 06 June, 14:30–16:15

You may submit answers to as many questions as you wish but only the best two will count for the total mark. All questions are worth 25 marks.

You should ensure that you:

- *start a new answer booklet for each question which you attempt.*
- *indicate on the front page of the answer booklet which question you have attempted in that booklet.*
- *cross out all rough working and any working you do not want to be marked. If you have used separate answer booklets for rough work please cross through the front of each such booklet and attach these answer booklets at the back of your work.*
- *hand in your answers in numerical order.*

If you do not attempt any questions, you should still hand in an answer booklet with the front sheet completed.

For this paper:

- Permitted calculator series: Casio fx-83, Casio fx-85, Sharp EL-531.
- New Cambridge Statistical Tables are provided.

Do not turn this page until you are told that you may do so

Q1

1. (a) [11 marks] Table 1 shows data collected from visitors of a website related to outdoors activities. The website displays advertisements, and the developers are interested in predicting whether a visitor is likely to click on an advertisement based on their estimated profile.

Table 1: Web advertisement data

income	**age**	**likes_sports**	**frequent_visitor**	**clicks_on_ad**
High	$\leqslant 30$	No	No	No
High	$\leqslant 30$	No	Yes	No
Low	$\geqslant 60$	Yes	Yes	No
Medium	$\leqslant 30$	No	No	No
Medium	$\geqslant 60$	No	Yes	No
Low	$\leqslant 30$	Yes	No	Yes
High	> 30 & < 60	No	No	Yes
Medium	$\geqslant 60$	No	No	Yes
Low	$\geqslant 60$	Yes	No	Yes
Low	> 30 & < 60	Yes	Yes	Yes
Medium	$\geqslant 60$	Yes	No	Yes
Medium	$\leqslant 30$	Yes	Yes	Yes
Medium	> 30 & < 60	No	Yes	Yes
High	> 30 & < 60	Yes	No	Yes

(i) You decide to use a Naïve Bayes classifier to predict whether a new user will click on an advertisement. Assuming we are using the $0-1$ loss, write an expression for the decision rule used by the Naïve Bayes classifier. Make sure the expression outlines the key assumption made by Naïve Bayes (simplify the expression as much as you can).

(ii) Use a Naïve Bayes classifier to predict whether a new user
$x = (\texttt{income=Medium}, \texttt{age=`}\leqslant 30\texttt{'}, \texttt{likes\_sports=Yes}, \texttt{frequent\_visitor=No})$
will click on an advertisement. Provide the prediction, as well as details of the calculation used to obtain the result.

(iii) Assume now that we could not estimate the income of the new user, so that its attributes are:
$x = (\texttt{income=?}, \texttt{age=}\leqslant 30, \texttt{likes\_sports=yes}, \texttt{frequent\_visitor=No})$.
How would your calculations change when this attribute is missing?

(b) [9 marks] Assume that you are given a data set of n samples, $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$, and $y_i \in \{0, 1\}$, and decide to use Naïve Bayes to perform classification. Further, assume that all predictors are binary features, that is, each element x_{ij} of the i-th data point can only take values in $\{0, 1\}$. For instance, `age` and `income` in Table 1 are binary features. A binary feature is thus parameterized by a single value, such that $\mathbb{P}(x_{ij} = 1|y_i = k) = \phi_{kj} \in [0, 1]$, and $\mathbb{P}(x_{ij} = 0|y_i = k) = 1 - \phi_{kj}$. Show that under a Naïve Bayes model with only binary predictors, the Bayes classifier $f_{\text{Bayes}}(x)$ minimizing the total risk for the $0-1$ loss has a linear discriminant function of the form

$$f_{\text{Bayes}}(x) = \underset{k=1,2}{\arg\max} \;\; a_k + b_k^\top x,$$

for a choice of a_k and b_k that you should define.

(c) [5 marks] Provide a definition for the *bias* and *variance* of a classifier (you do not need to write formulae). Assume you are designing a classifier, and decide to utilize Naïve Bayes. Describe two possible design choices that would affect the bias and the variance of your classifier.

Q2

2. Consider a training data set $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$, and $y_i \in \mathbb{R}$. Assume $n > p$. You may opt to compactly represent the data set using an $n \times p$ matrix $X \in \mathbb{R}^{n \times p}$, whose rows x_i represent the training points, and a vector y, whose entries y_i represent the labels.

   (a) [11 marks]

      (i) Write the empirical risk minimization problem corresponding to a linear regression with squared loss and L2 regularization with parameter $\lambda \geqslant 0$. Derive a closed-form optimal solution for the empirical risk minimization problem (i.e. find the optimal regression coefficients $\widehat{\beta}$).

      (ii) Consider a pre-processing step such that each training point x_i is transformed using $\phi(x_i)$, for some choice of function $\phi : \mathbb{R}^p \to \mathbb{R}^d$, where $d \leqslant p$. How would the expression for the optimal solution change?

      (iii) Briefly describe how the value of λ affects the *bias* and *variance* of the regression error. Mention one approach you could use to select the optimal value of λ. How do you think the optimal choice of λ would be affected by increasing the size n of the training set?

   (b) [6 marks] Consider augmenting the original data matrix X by adding p rows (data points), so that the i-th added row has value $\sqrt{\gamma}$ at position i, and 0 elsewhere. Assume $\gamma \geqslant 0$. We also augment the vector of observations y by appending a 0 to it, p times, in correspondence of each added point in the X matrix. The data set can now be represented as:

$$\widehat{X} = \begin{bmatrix} X \\ \sqrt{\gamma} I_{p \times p} \end{bmatrix},$$

   and

$$\widehat{y} = \begin{bmatrix} y \\ 0_{p \times 1} \end{bmatrix}.$$

   Show that solving the unregularized linear regression problem (ordinary least square) using this augmented data set is equivalent to performing L2-regularized linear regression, where the regularization parameter is now γ.

   (c) [8 marks] Numerical optimization provides an alternative route for the solution of regularized empirical risk minimization problems such as the one in (a)(i).

      (i) The Newton-Raphson algorithm is a popular approach of this kind. Write an expression for a "Newton update" as defined by this algorithm. Next, using the gradient and Hessian of the empirical risk, show that one Newton update is sufficient to find the optimal solution. Compare the update rule to the closed-form solution from (a)(i), and provide a brief explanation for the relationship between these expressions.

      (ii) The gradient descent algorithm is another popular approach for solving optimization problems. Explain why this approach may sometimes be computationally more efficient than the closed-form optimal solution for regularized linear regression in (a)(i). (No need to provide an exact count of the operations involved, it is sufficient to focus on how these two approaches scale with n and p).

Q3

3. (a) [8 marks] Consider a binary classification task where our goal is to predict $\widehat{y}_i \in \{0, 1\}$ for all data points $x_i \in \mathbb{R}^p, i \in \{1, \ldots, n\}$.

(i) By comparing the predictions $\widehat{y}_i$ to available true labels y_i, we can build a confusion matrix like the one depicted in Table 2. Given a confusion matrix of this kind, define specificity, sensitivity, precision, and recall. Given $TN = 100$, $TP = 300$, $FN = 400$, $FP = 200$, calculate the specificity, sensitivity, precision, and recall.

Table 2: Generic confusion matrix.

	True 0	True 1
Predicted 0	True negatives (TN)	False negatives (FN)
Predicted 1	False positives (FP)	True positives (TP)

(ii) Assume now that an algorithm outputs an estimate of the probability $P(y_i = 1|x_i)$. We can measure performance using the Receiver Operating Characteristic (ROC) curve. Briefly describe the ROC curve, and sketch an example. What are the minimum and maximum values that the area under the ROC curve can take for a binary prediction problem?

(b) [8 marks] Consider the three real-valued data points:

$$(x_1, y_1) = (-1.0, 4.0)$$
$$(x_2, y_2) = (+1.0, 8.0)$$
$$(x_3, y_3) = (+4.0, 6.0).$$

We are interested in fitting a *regression* tree with a single split point, which is called a *stump*, and is only composed of a root and two leaf nodes. Given a possible value v to be used as split threshold for the stump, write an expression for the quality of the split using the squared loss. Using the provided data points, determine whether splitting at $v = 0.0$ is a better choice than splitting at $v = 2.5$, again using the squared loss.

(c) [9 marks] Consider the following training points for a binary classification problem:

$$(x_1, y_1) = (-1.0, -1)$$
$$(x_2, y_2) = (+1.0, +1)$$
$$(x_3, y_3) = (+4.0, -1).$$

We decide to use Adaboost, adopting decision stumps (which are *classification* trees) as weak learners.

(i) What is the weight initially assigned to each point?

(ii) Draw the decision boundary of a first decision stump (indicating positive and negative side).

(iii) What are the weights of each data point after one iteration of the Adaboost algorithm?

(iv) Can Adaboost achieve perfect classification in this example? If it does, how many iterations will it need? Justify your answer.

SB2.2 2018 paper

Q1

under 0-1 loss $h^* = \arg\max_{k \in \{1,\dots,K\}} \mathbb{P}(Y=k \mid X=x)$

(a) (i) Naive Bayes: we have input $x = (x_1, \dots x_5)^T$

$$g_k(x) = \prod_{j=1}^{5} g_{kj}(x_j; \theta_{kj})$$

$$\hat{h}^{(B)}(x) = \arg\max \pi_k \hat{\mathbb{P}}(X=x \mid Y=k)$$

$$= \arg\max \log \pi_k + \log \hat{\mathbb{P}}(X=x \mid Y=k)$$

$$= \arg\max \log \pi_k + \sum_{j=1}^{5} \log \hat{\mathbb{P}}(X_j = x_j \mid Y=k)$$

(ii) $\hat{\pi}_1 = 9/14 \qquad \hat{\pi}_{-1} = 5/14$

$\hat{\mathbb{P}}(\text{Medium} \mid Y=+1) = 4/9 \qquad \hat{\mathbb{P}}(\text{age} \le 30 \mid Y=+1) = 2/9$

$\hat{\mathbb{P}}(\text{like sports} = \text{yes} \mid Y=+1) = 2/3 \qquad \hat{\mathbb{P}}(\text{frequent visitor} \mid Y=+1) = 2/3$

$\hat{\mathbb{P}}(X=x \mid Y=+1) = 2/9 \times 4/9 \times 2/3 \times 2/3 = 0.049$

$\hat{\mathbb{P}}(X=x \mid Y=-1) = 3/5 \times 2/5 \times 1/5 \times 2/5 = 0.019$

$\Rightarrow \hat{\pi}_{+1} \hat{\mathbb{P}}(X=x \mid Y=+1) > \hat{\pi}_{-1} \hat{\mathbb{P}}(X=x \mid Y=-1)$

$\Rightarrow +1$

(iii) $\hat{\pi}_{+1} \hat{\mathbb{P}}(X=x \mid Y=+1) = 9/14 \times 2/9 \times 2/3 \times 2/3 = 4/63$

$\hat{\pi}_{-1} \hat{\mathbb{P}}(X=x \mid Y=-1) = 5/14 \times 3/5 \times 1/5 \times 2/5 = 3/175$

$\Rightarrow +1$

(b) we want: $\mathbb{P}(X=x \mid Y=k) = \prod_{j=1}^{p} \phi_{kj}^{x_j} (1-\phi_{kj})^{1-x_j}$

$$\Rightarrow \log \mathbb{P}(X=x \mid Y=k) = \sum_{j=1}^{p} \left[x_j \log(\phi_{kj}) + (1-x_j) \log(1-\phi_{kj}) \right]$$

$$= \sum_{j=1}^{p} \log(1-\phi_{kj}) + \sum_{j=1}^{p} x_j \log\left(\frac{\phi_{kj}}{1-\phi_{kj}}\right)$$

$\Rightarrow$

$$a_k = \log \pi_k + \sum_{j=1}^{p} \log(1-\phi_{kj})$$

$$b_{kj} = \log\left(\frac{\phi_{kj}}{1-\phi_{kj}}\right)$$

(c) bias

Answer: **(N)** A classifier's bias is the difference between optimal and average classification. Variance is variation in the classifier's output for different data sets of the same size. Given what we discussed in the course, two natural examples the student could make for ways of tuning bias and variance for Naïve Bayes could be (1) to share parameters across classes to decrease variance and increase bias (e.g. if we use Gaussian features, different classes could have the same variance for some of the features); (2) to use regularization while fitting the parameters of the features, e.g. penalizing small values for a multinomial feature, which would increase bias. 5 marks, half credit for bias/variance description, half for examples.

Q2.

(a) (i) Empirical risk minimization, linear regression, squared loss.

$$J(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \frac{\lambda}{n}\|\beta\|^2$$

therefore $\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\|\beta\|^2$

$$= \arg\min_{\beta} \|y - X\beta\|^2 + \lambda\|\beta\|^2$$

$$= \arg\min_{\beta} \underbrace{(\text{const} + \beta^T X^T X\beta - 2(X^T y)^T\beta + \lambda\beta^T\beta)}_{J(\beta)}$$

$$\partial J(\beta)/\partial\beta = 2X^TX\beta - 2X^Ty + 2\lambda\beta$$

$$= 0$$

$$\Rightarrow \hat{\beta} = (X^TX + \lambda I)^{-1}X^Ty$$

(ii) $x_i \to \phi(x_i)$

Then $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \phi(x_i)^T\beta)^2 + \frac{\lambda}{n}\|\beta\|^2$

$$\Rightarrow \hat{\beta} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T y$$

where $\Phi = \begin{pmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_n)^T \end{pmatrix}$ is $n \times d$

(iii) For large λ, we penalize more complex model. So simple model is preferred

Therefore $\lambda \uparrow$, bias $\uparrow$, variance $\downarrow$.

By validation or cross-validation.

b) $X \in \mathbb{R}^{n \times p}$

$$\begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & & \\ x_{n1} & \cdots & x_{np} \end{pmatrix}$$

$$\tilde{X} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & & \\ x_{n1} & \cdots & x_{np} \\ \sqrt{r} & & 0 \\ & \ddots & \\ 0 & & \sqrt{r} \end{bmatrix} \qquad \tilde{X} \in \mathbb{R}^{(n+p) \times p}$$

$$\tilde{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_p \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_{2p} \end{pmatrix}$$

Now: $J(\beta) = \frac{1}{n} \sum_{i=1}^{2p} (y_i - x_i^T \beta)^2$

$$= \frac{1}{n} \sum_{i=1}^{p} (y_i - x_i^T \beta)^2 + \frac{1}{n} \sum_{i=p+1}^{2p} (y_i - x_i^T \beta)^2$$

$$= \frac{1}{n} \sum_{i=1}^{p} (y_i - x_i^T \beta)^2 + \frac{1}{n} \sum_{i=1}^{p} (0 - \sqrt{r}\beta_i)^2$$

$$= \frac{1}{n} \sum_{i=1}^{p} (y_i - x_i^T \beta)^2 + \frac{r}{n} \|\beta\|^2 \quad \square$$

Alternatively:

Since for unregularized linear regression

$$\hat{\beta} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T \tilde{y}$$

$$\Rightarrow \tilde{X}^T \tilde{X} = \begin{bmatrix} X^T & \sqrt{r} I_{p \times p} \end{bmatrix} \begin{bmatrix} X \\ \sqrt{r} I_{p \times p} \end{bmatrix}$$

$$= X^T X + r I_{p \times p}$$

$$\tilde{X}^T \tilde{y} = \begin{bmatrix} X^T & \sqrt{r} I_{p \times p} \end{bmatrix} \begin{bmatrix} y \\ 0_{p \times 1} \end{bmatrix}$$

$$= X^T y$$

$$\Rightarrow \hat{\beta} = (X^T X + r I_{p \times p}) X^T y \quad \square$$

(c) We have $J(\beta) = \frac{1}{n}\|y - X\beta\|^2 + \lambda\|\beta\|^2$

Newton-Raphson:

$$\beta^{(t+1)} = \beta^{(t)} - \left(\nabla^2_\beta J(\beta^{(t)})\right)^{-1} \nabla_\beta J(\beta^{(t)})$$

Gradient: $\nabla_\beta J(\beta_t) = 2(X^TX\beta_t - X^Ty) + 2\lambda\beta_t$

$$\nabla^2_\beta J(\beta_t) = 2(X^TX) + 2\lambda I$$

$$\Rightarrow \quad \beta^{t+1} \leftarrow \beta^t - \left(2(X^TX + \lambda I)\right)^{-1}$$

$$\beta_{t+1} \leftarrow \beta_t - \left(\nabla^2_\beta \hat{R}(\beta_t)\right)^{-1} \nabla_\beta \hat{R}(\beta_t)$$

Gradient: $\nabla_\beta \hat{R}(\beta_t) = 2(X^TX\beta_t - X^Ty) + 2\lambda\beta_t$

$$\nabla^2_\beta \hat{R}(\beta_t) = 2(X^TX + \lambda I)$$

therefore Newton-update gives us: $\beta_{t+1} \leftarrow \beta_t - (X^TX + \lambda I)^{-1}(X^TX\beta_t - X^Ty + \lambda\beta_t)$

$$= \beta_t - (X^TX + \lambda I)^{-1}[(X^TX + \lambda)\beta_t - X^Ty]$$

$$= \beta_t - \beta_t + (X^TX + \lambda I)^{-1}X^Ty$$

$$= (X^TX + \lambda I)^{-1}X^Ty$$

$\Rightarrow$ This minimizes the empirical risk.

(ii) Gradient descent has a lower cost for each single iteration.

Q3.

(a) (i) Sensitivity: $P(h(X)=1 \mid Y=1) = TP/(TP+FN)$

$= 300/(300+400) = 3/7$

Specificity: $P(h(X)=-1 \mid Y=-1) = TN/(TN+FP) = 100/(100+200)$

$= 1/3$

precision: $P(Y=1 \mid h(X)=1) = TP/(FP+TP)$

$= 300/(300+200) = 3/5$

recall: $P(h(X)=1 \mid Y=1) = TP/(TP+FN) = 3/7$

(ii) ROC curve assess the performance of plug in classifiers.

Let TPR (true positive rate) : FPR (false positive rate)

$= P(\hat{h}_t(X)=1 \mid Y=1) = \beta(t)$ $= P(\hat{h}_t(X)=1 \mid Y=0) = \alpha(t)$

Then ROC curve plots TPR against FPR.

$$AUC = \int_1^0 \beta(t)\, d\alpha(t)$$

$AUC_{max} = 1$; $AUC_{min} = 0$ 0.5

(b)

1. (a) [11 marks] Table 1 shows data collected from visitors of a website related to outdoors activities. The website displays advertisements, and the developers are interested in predicting whether a visitor is likely to click on an advertisement based on their estimated profile.

Table 1: Web advertisement data

income	**age**	**likes_sports**	**frequent_visitor**	**clicks_on_ad**
High	$\leqslant 30$	No	No	No
High	$\leqslant 30$	No	Yes	No
Low	$\geqslant 60$	Yes	Yes	No
Medium	$\leqslant 30$	No	No	No
Medium	$\geqslant 60$	No	Yes	No
Low	$\leqslant 30$	Yes	No	Yes
High	> 30 & < 60	No	No	Yes
Medium	$\geqslant 60$	No	No	Yes
Low	$\geqslant 60$	Yes	No	Yes
Low	> 30 & < 60	Yes	Yes	Yes
Medium	$\geqslant 60$	Yes	No	Yes
Medium	$\leqslant 30$	Yes	Yes	Yes
Medium	> 30 & < 60	No	Yes	Yes
High	> 30 & < 60	Yes	No	Yes

(i) You decide to use a Naïve Bayes classifier to predict whether a new user will click on an advertisement. Assuming we are using the $0-1$ loss, write an expression for the decision rule used by the Naïve Bayes classifier. Make sure the expression outlines the key assumption made by Naïve Bayes (simplify the expression as much as you can).
Answer: B

$$\begin{aligned} f(x) &= \arg\max_k \ \pi_k \mathbb{P}(X = x | Y = k) \\ &= \arg\max_k \ \log \pi_k + \log \mathbb{P}(X = x | Y = k) \\ &= \arg\max_k \ \log \pi_k + \sum_{j=1}^{4} \log P(X_j = x_j | Y = k), \end{aligned}$$

where we used $\mathbb{P}(Y = k) = \pi_k$ (the student may use different notation, or not use logarithms). 3 marks.

(ii) Use a Naïve Bayes classifier to predict whether a new user
$x = (\texttt{income=Medium}, \texttt{age='}\leqslant 30\texttt{'}, \texttt{likes\_sports=Yes}, \texttt{frequent\_visitor=No})$
will click on an advertisement. Provide the prediction, as well as details of the calculation used to obtain the result.
Answer: (S) Denote `clicks_on_ad=yes` with $Y = +1$ and `clicks_on_ad=no` with $Y = -1$. Estimates of class probabilities:

$$\widehat{\pi}_{+1} = 9/14, \qquad \widehat{\pi}_{-1} = 5/14.$$

Estimates of conditional probabilities:

$$\begin{aligned}
\widehat{\mathbb{P}}(\texttt{income=Medium}|Y=+1) &= 4/9, & \widehat{\mathbb{P}}(\texttt{income=Medium}|Y=-1) &= 2/5,\\
\widehat{\mathbb{P}}(\texttt{age='}\leqslant 30\texttt{'}|Y=+1) &= 2/9, & \widehat{\mathbb{P}}(\texttt{age='}\leqslant 30\texttt{'}|Y=-1) &= 3/5,\\
\widehat{\mathbb{P}}(\texttt{likes_sports=Yes}|Y=+1) &= 6/9, & \widehat{\mathbb{P}}(\texttt{likes_sports=Yes}|Y=-1) &= 1/5,\\
\widehat{\mathbb{P}}(\texttt{frequent_visitor=No}|Y=+1) &= 6/9, & \widehat{\mathbb{P}}(\texttt{frequent_visitor=No}|Y=-1) &= 2/5.
\end{aligned}$$

Thus,

$$\widehat{\mathbb{P}}(X=x|Y=+1) = \frac{2}{9}\cdot\frac{4}{9}\cdot\frac{6}{9}\cdot\frac{6}{9} = 0.044 \qquad \widehat{\mathbb{P}}(X=x|Y=-1) = \frac{3}{5}\cdot\frac{2}{5}\cdot\frac{1}{5}\cdot\frac{2}{5} = 0.019.$$

Since $\widehat{\pi}_{+1}\widehat{\mathbb{P}}(X=x|Y=+1) > \widehat{\pi}_{-1}\widehat{\mathbb{P}}(X=x|Y=-1)$, predicted class is $+1$. 5 marks.

(iii) Assume now that we could not estimate the income of the new user, so that its attributes are:
$x = (\texttt{income=?}, \texttt{age=}\leqslant 30, \texttt{likes\_sports=yes}, \texttt{frequent\_visitor=No})$.
How would your calculations change when this attribute is missing?
Answer: (S) In Naïve Bayes, we would simply ignore that attribute. The calculations are now:

$$\widehat{\mathbb{P}}(X=x|Y=+1) = \frac{2}{9}\cdot\frac{6}{9}\cdot\frac{6}{9} = 0.099 \qquad \widehat{\mathbb{P}}(X=x|Y=-1) = \frac{3}{5}\cdot\frac{1}{5}\cdot\frac{2}{5} = 0.048.$$

and the prediction is unchanged. 3 marks.

(b) [9 marks] Assume that you are given a data set of n samples, $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$, and $y_i \in \{0,1\}$, and decide to use Naïve Bayes to perform classification. Further, assume that all predictors are binary features, that is, each element x_{ij} of the i-th data point can only take values in $\{0,1\}$. For instance, `age` and `income` in Table 1 are binary features. A binary feature is thus parameterized by a single value, such that $\mathbb{P}(x_{ij}=1|y_i=k) = \phi_{kj} \in [0,1]$, and $\mathbb{P}(x_{ij}=0|y_i=k) = 1-\phi_{kj}$. Show that under a Naïve Bayes model with only binary predictors, the Bayes classifier $f_{\text{Bayes}}(x)$ minimizing the total risk for the $0-1$ loss has a linear discriminant function of the form

$$f_{\text{Bayes}}(x) = \underset{k=1,2}{\arg\max}\ \ a_k + b_k^\top x,$$

for a choice of a_k and b_k that you should define.
Answer: (S/N)

$$\begin{aligned}
\log \mathbb{P}(X=x|Y=k) &= \sum_{j=1}^p \left[x_j \log \phi_{kj} + (1-x_j)\log(1-\phi_{kj})\right]\\
&= \sum_{j=1}^p \log(1-\phi_{kj}) + \sum_{j=1}^p x_j \log\frac{\phi_{kj}}{1-\phi_{kj}}
\end{aligned}$$

So that the discriminant functions are linear, with:

$$a_k = \log \pi_k + \sum_{j=1}^p \log(1-\phi_{kj})$$

$$b_{kj} = \log\frac{\phi_{kj}}{1-\phi_{kj}}$$

for each k and j. 9 marks. Partial credit for setting up the problem correctly (e.g. first equation) but not isolating the a_k and b_{kj} terms correctly.

(c) [5 marks] Provide a definition for the *bias* and *variance* of a classifier (you do not need to write formulae). Assume you are designing a classifier, and decide to utilize Naïve Bayes. Describe two possible design choices that would affect the bias and the variance of your classifier.

Answer: **(N)** A classifier's bias is the difference between optimal and average classification. Variance is variation in the classifier's output for different data sets of the same size. Given what we discussed in the course, two natural examples the student could make for ways of tuning bias and variance for Naïve Bayes could be (1) to share parameters across classes to decrease variance and increase bias (e.g. if we use Gaussian features, different classes could have the same variance for some of the features); (2) to use regularization while fitting the parameters of the features, e.g. penalizing small values for a multinomial feature, which would increase bias. 5 marks, half credit for bias/variance description, half for examples.

2. Consider a training data set $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$, and $y_i \in \mathbb{R}$. Assume $n > p$. You may opt to compactly represent the data set using an $n \times p$ matrix $X \in \mathbb{R}^{n \times p}$, whose rows x_i represent the training points, and a vector y, whose entries y_i represent the labels.

(a) [11 marks]

(i) Write the empirical risk minimization problem corresponding to a linear regression with squared loss and L2 regularization with parameter $\lambda \geqslant 0$. Derive a closed-form optimal solution for the empirical risk minimization problem (i.e. find the optimal regression coefficients $\widehat{\beta}$).

Answer: **B** Assuming we have incorporated the bias term in β and prepended a column of 1's to the data matrix for notational simplicity,

$$\widehat{R}(\beta) = \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$$

$$\frac{\partial \widehat{R}(\beta)}{\partial \beta} = 2(X^\top X\beta - X^\top y) + 2\lambda\beta$$

$$2(X^\top X\beta - X^\top y) + 2\lambda\beta = 0 \rightarrow \widehat{\beta} = (X^\top X + \lambda I)^{-1} X^\top y.$$

(the student may or may not include a factor of $\frac{1}{n}$ in the above calculations). 5 marks.

(ii) Consider a pre-processing step such that each training point x_i is transformed using $\phi(x_i)$, for some choice of function $\phi : \mathbb{R}^p \rightarrow \mathbb{R}^d$, where $d \leqslant p$. How would the expression for the optimal solution change?

Answer: **B** The problem simply becomes

$$\widehat{R}(\beta) = \|y - \Phi\beta\|_2^2 + \lambda\|\beta\|_2^2,$$

where the matrix $\Phi \in \mathbb{R}^{n \times d}$ contains the transformed data. The solution is also unchanged, besides the new data matrix:

$$2(\Phi^\top \Phi\beta - \Phi^\top y) + 2\lambda\beta = 0 \rightarrow \widehat{\beta} = (\Phi^\top \Phi + \lambda I)^{-1} \Phi^\top y$$

3 marks.

(iii) Briefly describe how the value of λ affects the *bias* and *variance* of the regression error. Mention one approach you could use to select the optimal value of λ. How do you think the optimal choice of λ would be affected by increasing the size n of the training set?

Answer: **B** A larger value of λ would increase bias and reduce variance. The optimal λ could be chosen by validation or cross-validation. Increasing the sample size should lead to a decreased optimal value of λ. 3 marks.

(b) [6 marks] Consider augmenting the original data matrix X by adding p rows (data points), so that the i-th added row has value $\sqrt{\gamma}$ at position i, and 0 elsewhere. Assume $\gamma \geqslant 0$. We also augment the vector of observations y by appending a 0 to it, p times, in correspondence of each added point in the X matrix. The data set can now be represented as:

$$\widehat{X} = \begin{bmatrix} X \\ \sqrt{\gamma} I_{p \times p} \end{bmatrix},$$

and

$$\widehat{y} = \begin{bmatrix} y \\ 0_{p \times 1} \end{bmatrix}.$$

Show that solving the unregularized linear regression problem (ordinary least square) using this augmented data set is equivalent to performing L2-regularized linear regression, where the regularization parameter is now γ.

Answer: **(S/N)** Using the OLS estimator $\widehat{\beta} = (\widehat{X}^\top \widehat{X})^{-1} \widehat{X}^\top \widehat{y}$, where

$$\widehat{X}^\top \widehat{X} = \begin{bmatrix} X & \sqrt{\gamma} I_{p\times p} \end{bmatrix} \begin{bmatrix} X \\ \sqrt{\gamma} I_{p\times p} \end{bmatrix} = X^\top X + \gamma I_{p\times p} \tag{1}$$

and

$$\widehat{X}^\top \widehat{y} = \begin{bmatrix} X & \sqrt{\gamma} I_{p\times p} \end{bmatrix} \begin{bmatrix} y \\ 0_{p\times 1} \end{bmatrix} = X^\top y \tag{2}$$

we recover the ridge estimator

$$\widehat{\beta} = (\widehat{X}^\top \widehat{X} + \gamma I_{p\times p})^{-1} \widehat{X}^\top \widehat{y}$$

6 marks. The student must show intermediate steps.

(c) [8 marks] Numerical optimization provides an alternative route for the solution of regularized empirical risk minimization problems such as the one in point (i).

(i) The Newton-Raphson algorithm is a popular approach of this kind. Write an expression for a "Newton update" as defined by this algorithm. Next, using the gradient and Hessian of the empirical risk, show that one Newton update is sufficient to find the optimal solution. Compare the update rule to the closed-form solution from point (i), and provide a brief explanation for the relationship between these expressions.

Answer: **(N)** The Newton update rule is

$$\beta_{t+1} \leftarrow \beta_t - (\nabla^2_\beta \widehat{R}(\beta_t))^{-1} \nabla_\beta \widehat{R}(\beta_t).$$

Gradient and Hessian are:

$$\nabla_\beta \widehat{R}(\beta_t) = 2(X^\top X \beta_t - X^\top y) + 2\lambda\beta_t$$

$$\nabla^2_\beta \widehat{R}(\beta_t) = 2(X^\top X + \lambda I),$$

and the Newton update rule becomes

$$\begin{aligned} \beta_{t+1} &\leftarrow \beta_t - (\nabla^2_\beta \widehat{R}(\beta_t))^{-1} \nabla_\beta \widehat{R}(\beta_t) \\ &= \beta_t - (X^\top X + \lambda I)^{-1}(X^\top X \beta_t - X^\top y + \lambda\beta_t) \\ &= \beta_t - (X^\top X + \lambda I)^{-1} \left[(X^\top X + \lambda I)\beta_t - X^\top y\right] \\ &= \beta_t - \beta_t + (X^\top X + \lambda I)^{-1} X^\top y \\ &= (X^\top X + \lambda I)^{-1} X^\top y. \end{aligned}$$

Thus the Newton-Raphson update finds the minimum of the empirical risk in one step, and is equivalent to computing the closed-form solution. The reason is that the Newton-Raphson algorithm minimizes a quadratic approximation of the objective function. If the function is quadratic to begin with, as it is the case for the empirical risk of a ridge regression, Newton-Raphson directly minimizes the objective function. 1 marks for Newton update rule, 1 marks for Hessian (gradient previously computed), 2 marks for putting those together, 1 mark for brief discussion.

(ii) The gradient descent algorithm is another popular approach for solving optimization problems. Explain why this approach may sometimes be computationally more efficient than the closed-form optimal solution for regularized linear regression in (i). (No need to provide an exact count of the operations involved, it is sufficient to focus on how these two approaches scale with n and p).

Answer: **(N)** Computing the closed-form solution involves a number of matrix multiplications with cost $O(p^2 \times n)$, and the inversion of a $p \times p$ matrix, with cost $O(p^3)$. Although gradient descent may require several iterations, the cost of a single iteration is lower. In particular, we will not need to perform the $O(p^3)$ cost inversion. (The student may mention stochastic gradient descent, for which the cost of an iteration is even less). 3 marks.

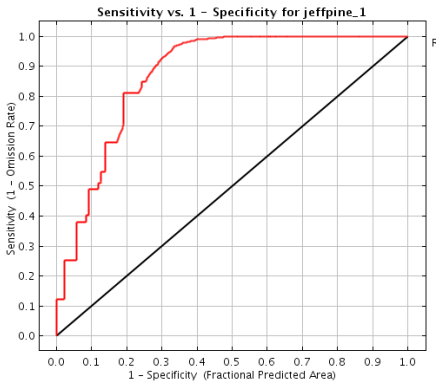


Figure 1: ROC sketch.

3. (a) [8 marks] Consider a binary classification task where our goal is to predict $\widehat{y}_i \in \{0, 1\}$ for all data points $x_i \in \mathbb{R}^p, i \in \{1, \ldots, n\}$.

(i) By comparing the predictions $\widehat{y}_i$ to available true labels y_i, we can build a confusion matrix like the one depicted in Table 2. Given a confusion matrix of this kind, define specificity, sensitivity, precision, and recall. Given $TN = 100$, $TP = 300$, $FN = 400$, $FP = 200$, calculate the specificity, sensitivity, precision, and recall.

Table 2: Generic confusion matrix.

	True 0	True 1
Predicted 0	True negatives (TN)	False negatives (FN)
Predicted 1	False positives (FP)	True positives (TP)

Answer: **B** Sensitivity: TP/(TP + FN). Specificity: TN/(TN + FP). Precision: TP/(TP + FP). Recall: TP/(TP + FN).
Sensitivity: $300/(300 + 400) = 3/7$. Specificity: $100/(100 + 200) = 1/3$. Precision: $300/(300 + 200) = 3/5$. Recall: $300/(300 + 400) = 3/7$. 4 marks.

(ii) Assume now that an algorithm outputs an estimate of the probability $P(y_i = 1|x_i)$. We can measure performance using the Receiver Operating Characteristic (ROC) curve. Briefly describe the ROC curve, and sketch an example. What are the minimum and maximum values that the area under the ROC curve can take for a binary prediction problem?

Answer: **B** Given a value α such that we predict $y_i = 1$ if $P(y_i = 1|x_i) > \alpha$, the ROC curve provides the sensitivity and specificity of a classification algorithm corresponding to different values of α. Sketch in Figure 1. The minimum value is 0.5, the maximum value is 1.
4 marks.

(b) [8 marks] Consider the three real-valued data points:

$$(x_1, y_1) = (-1.0, 4.0)$$
$$(x_2, y_2) = (+1.0, 8.0)$$
$$(x_3, y_3) = (+4.0, 6.0).$$

We are interested in fitting a *regression* tree with a single split point, which is called a *stump*, and is only composed of a root and two leaf nodes. Given a possible value v to be used as split threshold for the stump, write an expression for the quality of the split

using the squared loss. Using the provided data points, determine whether splitting at $v = 0.0$ is a better choice than splitting at $v = 2.5$, again using the squared loss.

Answer: **(S)** Define the sets

$$I_< = \{i : x_i < v\} \qquad\qquad I_\geq = \{i : x_i \geq v\}$$

and the values

$$\beta_< = \frac{\sum_{i \in I_<} y_i}{|I_<|} \qquad\qquad \beta_\geq = \frac{\sum_{i \in I_\geq} y_i}{|I_\geq|}.$$

(Equality can be on either side). To compute the quality $\widehat{Q}_v$ of split at v under the squared error loss we calculate:

$$\widehat{Q}_v = \sum_{i \in I_<} (y_i - \beta_<)^2 + \sum_{i \in I_\geq} (y_i - \beta_\geq)^2,$$

where a smaler value corresponds to higher quality. For split point 1.5 we get

$$\widehat{Q}_{0.0} = \left(8 - \frac{8+6}{2}\right)^2 + \left(6 - \frac{8+6}{2}\right)^2 + (4-4)^2 = 1^2 + 1^2 + 0 = 2$$

For point 3,

$$\widehat{Q}_{2.5} = \left(8 - \frac{8+4}{2}\right)^2 + \left(4 - \frac{8+4}{2}\right)^2 + (6-6)^2 = 2^2 + -2^2 + 0 = 8$$

$v = 0.0$ is the preferred split point. 4 marks for definitions, 4 marks for calculations.

(c) [9 marks] Consider the following training points for a binary classification problem:

$$(x_1, y_1) = (-1.0, -1)$$
$$(x_2, y_2) = (+1.0, +1)$$
$$(x_3, y_3) = (+4.0, -1).$$

We decide to use Adaboost, adopting decision stumps (which are *classification* trees) as weak learners.

(i) What is the weight initially assigned to each point?
Answer: **(S)** 1/3. 1 mark.

(ii) Draw the decision boundary of a first decision stump (indicating positive and negative side).
Answer: **(S)** See Figure 2 (not unique). 2 marks.

(iii) What are the weights of each data point after one iteration of the Adaboost algorithm?
Answer: **(N)** $\epsilon_t = 1/3$, $\beta_t = 1/2 \ln(2) = 0.3465$. Correctly classified points: $D_2(i) = \frac{1/3 \times \exp(-0.3465)}{Z_2} \approx 0.25$, incorrectly classified: $D_2(i) = \frac{1/3 \times \exp(0.3465)}{Z_2} \approx 0.5$, where Z_2 is the normalization factor. 3 marks.

(iv) Can Adaboost achieve perfect classification in this example? If it does, how many iterations will it need? Justify your answer.
Answer: **(N)** Yes, it can. Three iterations are sufficient, for instance with base learners $[-1, -1, -1]$, $[-1, 1, 1]$ and $[1, 1, -1]$ and weights $[0.35, 0.55, 0.80]$ (a sketch is sufficient for full marks). 3 marks.

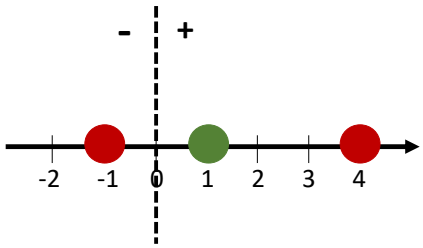


Figure 2: Adaboost decision boundary.

A15677W1

2019

SECOND PUBLIC EXAMINATION

Honour School of Mathematics and Statistics Part B: Paper SB2.2

Honour School of Mathematics Part B: Paper SB2.2

STATISTICAL MACHINE LEARNING

TRINITY TERM 2019

Tuesday 04 June, 14:30–16:15

You may submit answers to as many questions as you wish but only the best two will count for the total mark. All questions are worth 25 marks.

You should ensure that you:

- *start a new answer booklet for each question which you attempt.*
- *indicate on the front page of the answer booklet which question you have attempted in that booklet.*
- *cross out all rough working and any working you do not want to be marked. If you have used separate answer booklets for rough work please cross through the front of each such booklet and attach these answer booklets at the back of your work.*
- *hand in your answers in numerical order.*

If you do not attempt any questions, you should still hand in an answer booklet with the front sheet completed.

For this paper:

- Permitted calculator series: Casio fx-83, Casio fx-85, Sharp EL-531.
- New Cambridge Statistical Tables are provided.

Do not turn this page until you are told that you may do so

2019

1. Consider a data set $\{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$. You may opt to compactly represent the data set using an $n \times p$ matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, whose rows $\mathbf{x}_i$ represent the points, and assume the data is mean-centered: $\sum_i \mathbf{X}_{ij} = 0, \forall j$. Consider the covariance matrix estimated using $\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$.

   (a) [12 marks] You decide to run principal components analysis (PCA) using $\mathbf{S}$.

      (i) Assume we have obtained the first principal component $\mathbf{v}_1$. Formulate the optimization problem that should be solved to derive the *second* principal component $\mathbf{v}_2$, and solve it using the method of Lagrange multipliers.

   Now assume you have obtained the eigendecomposition $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_p]$ is a $p \times p$ orthogonal matrix whose columns are the eigenvectors for $\mathbf{S}$ and $\mathbf{\Lambda}$ is a $p \times p$ diagonal matrix whose diagonal entries are the eigenvalues $\{\lambda_i\}_{i=1}^p$ of $\mathbf{S}$. Assume that $\{\lambda_i\}_{i=1}^p$ are distinct and sorted in decreasing order on the diagonal of $\mathbf{\Lambda}$. Let $Tr(\mathbf{S})$ represent the trace of the matrix $\mathbf{S}$, and assume that $Tr(\mathbf{S}) = 1$. Define $r = \sum_{i=k+1}^p \lambda_i$.

      (ii) What does r represent? Running PCA suggests that k dimensions will be enough to capture 94% of the variance in your data set $\mathbf{X}$. What is the value of r?

      (iii) Let $\mathbf{V}_{1:k} = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$, $k < p$ be the matrix of k eigenvectors corresponding to the largest k eigenvalues. Consider the vector $\mathbf{b}_i = \mathbf{V}_{1:k}\mathbf{V}_{1:k}^\top\mathbf{x}_i$. Write, without proof, an equation representing the relationship between $\{\mathbf{x}_i\}_{i=1}^n$, $\{\mathbf{b}_i\}_{i=1}^n$, and r.

   (b) [13 marks] Now consider the same data set, with the addition of labels: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $y_i \in \mathbb{R}$. Represent the labels by a vector $\mathbf{y}$, whose entries y_i represent the labels. You would like to perform regularized linear regression, and assume the data comes from a linear model $y = \boldsymbol{\beta}^\top\mathbf{x} + \epsilon$, where ϵ is a noise term (we omit the intercept for simplicity, so that it does not need to be considered separately for regularization). Rather than using a direct regularization approach, however, you decide to perform "PCA-regression" performing the following operations:

      1. Compute the matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$ of k-dimensional PCA projections $\mathbf{Z} = \mathbf{X}\mathbf{V}_{1:k}$.
      2. Perform ordinary least squares on the projected data: $\widehat{\boldsymbol{\beta}}_Z = \left(\mathbf{Z}^\top\mathbf{Z}\right)^{-1}\mathbf{Z}^\top y$.
      3. Transform parameters to original coordinates: $\widehat{\boldsymbol{\beta}}_{PCA} = \mathbf{V}_{1:k}^\top\widehat{\boldsymbol{\beta}}_Z$.
      4. Given a new data point $\mathbf{x}_{new}$, predict its label using $\widehat{y}_{new} = \widehat{\boldsymbol{\beta}}_{PCA}^\top\mathbf{x}_{new}$.

      (i) Show that we can write $\widehat{\boldsymbol{\beta}}_{PCA} = \mathbf{V}\mathbf{\Delta}_{PCA}\mathbf{U}^\top\mathbf{y}$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and the matrix $\mathbf{\Delta}_{PCA}$ is 0 everywhere except on the diagonal, and where diagonal elements of $\mathbf{\Delta}_{PCA}$ are a function k and λ_i, the eigenvalues of $\mathbf{S}$.
      [*Hint: you may want to use the singular value decomposition (SVD)* $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$.]

      (ii) Show that performing regression using this approach is closely related to performing $L2$-regularized linear regression, which uses the estimator $\widehat{\boldsymbol{\beta}}_{L2} = \left(\mathbf{X}^\top\mathbf{X} + \alpha\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y}$, where $\alpha \geqslant 0$. Specifically, show that we can write $\widehat{\boldsymbol{\beta}}_{L2} = \mathbf{V}\mathbf{\Delta}_{L2}\mathbf{U}^\top\mathbf{y}$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and the matrix $\mathbf{\Delta}_{L2}$ is 0 everywhere except on the diagonal, where elements are a function of λ_i and α.
      [*Hint: you may again want to use the SVD.*]

      (iii) By comparing $\mathbf{\Delta}_{L2}$ and $\mathbf{\Delta}_{PCA}$, comment on the differences and similarities of these two methods for regularized linear regression.

Q2.

2. (a) [7 marks] Consider performing regression using the squared loss. Formally, you are looking for a function f that minimizes the risk

$$R(f) = \mathbb{E}\left[(Y - f(X))^2\right].$$

Show that the risk may be decomposed as

$$R(f) = (\text{noise}) + (\text{variance}) + (\text{bias})^2,$$

where you should give an explicit expression for each term, and provide a brief description of how they are affected by the choice of learning algorithm and its parameters. You may assume without proof that the optimal choice of f in this setting is given by the conditional mean $f_*(\mathbf{x}) = \mathbb{E}[Y|X = \mathbf{x}]$.

(b) [6 marks] You are given a data set $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{+1, -1\}$, and a *weak learner* $h(\cdot)$ that performs binary classification $h(\mathbf{x}) \in \{+1, -1\}$. Consider a set of weights $w_t(i)$, where $i \in \{1, \ldots, N\}$ is the index of a data point and t indicates the algorithm's iteration. The AdaBoost algorithm is trained using the following procedure:

- Initialize weights $w_0(i) = \frac{1}{N}, \forall i \in \{1, \ldots, N\}$.
- For $t = 1$ to T:
  1. Train a weak classifier $h_t(\cdot)$ by minimizing the weighted classification error ϵ_t (using the current set of weights).
  2. Compute the contribution of $h_t(\cdot)$, denoted by β_t.
  3. Update weights for each point, using the rule

$$w_t(i) = \frac{1}{Z_t} w_{t-1}(i) e^{-\beta_t y_i h_t(\mathbf{x}_i)},$$

  where Z_t is chosen such that $\sum_{i=1}^N w_t(i) = 1$.
- Output $\{\beta_t\}_{t=1}^T$ and $\{h_t(\cdot)\}_{t=1}^T$.

(i) What is a *weak* learner in this algorithm? (You do not need to provide mathematical detail.)

(ii) Provide explicit expressions for ϵ_t and β_t computed in steps 1 and 2 of the algorithm.

(iii) How does AdaBoost classify a new point $\mathbf{x}_{new}$?

(c) [12 marks] Consider the AdaBoost algorithm described in part (b).

(i) Show that

$$w_T(i) = \frac{1}{N} \frac{e^{-y_i \sum_{t=1}^T \beta_t h_t(\mathbf{x}_i)}}{\prod_{t=1}^T Z_t}.$$

(ii) Let $H(\cdot)$ be the final AdaBoost classifier, and define its training error as

$$E = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[H(\mathbf{x}_i) \neq y_i].$$

Show that $E \leqslant \prod_{t=1}^T Z_t$. [*Hint:* $e^{-c} \geqslant 1$ *if* $c \leqslant 0$ *and* $e^{-c} > 0$ *if* $c > 0$.]

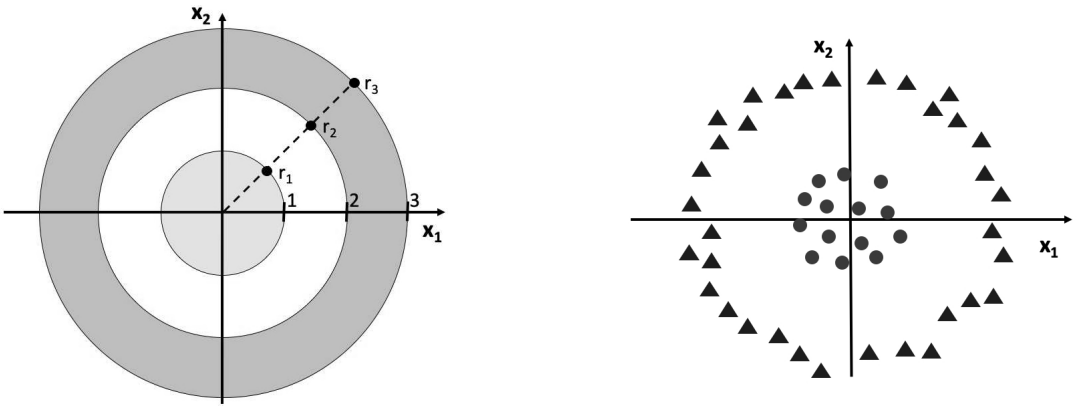
(iii) Use these results to show that $E \leqslant \prod_{t=1}^T 2\sqrt{\epsilon_t(1 - \epsilon_t)}$.

3. (a) [14 marks] Figure 1-a shows three concentric circles of radii $r_1 = 1$, $r_2 = 2$ and $r_3 = 3$. Assume a random set of points with positive label is sampled at distance $d < 1$ from the origin, and a second set of random points with negative label is sampled at distance $2 < d < 3$ from the origin. An example data set is shown in Figure 1-b.

   (i) For each of classification algorithms 1–6 listed below, state whether you would be able to obtain perfect classification accuracy on the training data set, and provide a brief explanation for why this may or may not be possible. Ignore any issues related to convergence (e.g. due to local minima/maxima). In each case, draw a rough sketch of a data set like the one in Figure 1-b and sketch the decision boundaries that you think the algorithm may produce. (State any assumptions you may introduce. There is no need to provide mathematical detail or numerical results.)

      1. Quadratic discriminant analysis.
      2. K-nearest neighbours with $K = 1$.
      3. Decision tree (the same feature may be used multiple times to split the data).
      4. Logistic regression with inputs $\{x_1, x_2\}$.
      5. An artificial neural network with inputs $\{x_1, x_2\}$, one hidden layer containing 2 neurons, and sigmoid activation functions.
      6. An artificial neural network with inputs $\{x_1, x_2\}$, one hidden layer containing 3 neurons, and sigmoid activation functions.

   (ii) Draw a tree that may be obtained running the decision tree algorithm on the data set in Figure 1-b (the same feature may be used multiple times to split the data). Indicate which feature and value is chosen for each split, and the classification rule used in each leaf (e.g. classify as positive/negative). You do not need to prune the tree or limit its depth.



(a) (b)

Figure 1: (a) Points with positive label are sampled at distance $d < 1$ from the origin, points with negative label are sampled at distance $2 < d < 3$ from the origin. (b) A sample data set where circles indicate positive points and triangles indicate negative points.

(b) [11 marks] Let $B = \{\mathbf{x} | \sqrt{\mathbf{x}^\top \mathbf{x}} \leqslant 1, \mathbf{x} \in \mathbb{R}^p\}$ be the p-dimensional unit ball centered at the origin. Consider a training set consisting of N samples, which are drawn uniformly at random from B (i.e. the probability that $\mathbf{x} \in A \subseteq B$ is proportional to the volume of A). Consider a point $\mathbf{x}_0$ located at the origin. The distance to its nearest neighbor is

$$d = \min_{i \in \{1,\ldots,N\}} \sqrt{\mathbf{x}_i^\top \mathbf{x}_i}.$$

(i) In the special case of $p = 1$, derive the cumulative distribution function $P(d \leqslant t)$, $0 \leqslant t \leqslant 1$.

(ii) Derive $P(d \leqslant t)$ for the more general case of any positive integer value of p.
[*Hint: the volume of a p-dimensional sphere of radius r is given by* $\frac{(r\sqrt{\pi})^p}{\Gamma(p/2+1)}$, *where* $\Gamma(\cdot)$ *indicates the Gamma function.*]

(iii) Assume you may obtain a data set of arbitrary size n. How many points do you need, as a function of p, to ensure that with probability 0.9 at least one point is within a distance of 0.1 from the origin?

(iv) How are these calculations related to the *curse of dimensionality*? How will this affect the K-nearest neighbours algorithm?

SB22 2019 Paper

Q1.

(a) (i) $X \in \mathbb{R}^{n \times p}$, $\sum_i x_{ij} = 0$, $S = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^T = \frac{1}{n-1} X^T X$

For the 2nd PC v_2, consider $Z_2 = v_2^T X$

where $v_2 \in \mathbb{R}^p$, $v_2^T v_2 = 1$, v_2 is orthogonal to v_1 (i.e $v_2^T v_1 = 0$)

We wish to find a v_2 s.t. maximizes $Var(Z_2) = v_2^T S v_2$.

Lagrangian of the problem: $L(v_2, \gamma_2, \mu) = v_2^T S v_2 - \gamma_2(v_2^T v_2 - 1) - \mu v_2^T v_1$

$$\frac{\partial L}{\partial v_2} = 2Sv_2 - 2\gamma_2 v_2 - \mu v_1 = 0$$

$$\Rightarrow \quad v_2^T S v_2 - v_2^T \gamma_2 v_2 = 0 \quad \Rightarrow \quad Sv_2 = \gamma_2 v_2$$

Therefore v_2 must be an eigenvector of S. Since we wish to maximize γ_2, we choose γ_2 to be the second largest eigenvalue, v_2 be its corresponding eigenvector.

(ii)

Now: $S = V \Lambda V^T$, $Tr(S) = Tr(\Lambda) = 1$

$r = \sum_{i=k+1}^{p} \lambda_i$ $\sum_{k+1}^{p} \lambda_i / Tr(S)$ is the reconstruction error.

Then r represents the variance explained by PC $k+1$ to PC p.

Since: $\dfrac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i} = 94\%$, $\sum_{i=1}^{p} \lambda_i = Tr(\Lambda) = 1$

$$\Rightarrow \quad \sum_{i=k+1}^{p} \lambda_i = 6\%$$

(iii) $V_{1:k} = [v_1, \dots v_k]$

$b_i = V_{1:k} V_{1:k}^T x_i$

$$V_{1:k} V_{1:k}^T = [v_1, \dots v_k] \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} = I_{k \times k}$$

$$\begin{pmatrix} — v_1 — \\ \vdots \\ — v_k — \end{pmatrix} \begin{pmatrix} | & & | \\ v_1 & \dots & v_k \\ | & & | \end{pmatrix}$$

$V_{1:k} V_{1:k}^T x_i = \hat{x}_i$

$\sum_{i=1}^{n} \| x_i - b_i \|_2^2 = (n-1) \sum_{j=k+1}^{p} \lambda_j$

$= (n-1) Tr(S) r$

(b) (i) $\{(x_i, y_i)\}_{i=1}^{n}$, $y = \beta^T x + \varepsilon$

$Z = X V_{1:k}$, $\hat{\beta}_Z = (Z^T Z)^{-1} Z^T y$

$\hat{\beta}_{PCA} = V_{1:k} \hat{\beta}_Z$, $\hat{y}_{new} = \hat{\beta}_{PCA}^T x_{new}$,

Since $X \in \mathbb{R}^{n \times p}$, by SVD $X = UDV^T$, ($UU^T = I_{n\times n}$, $V^T V = I_{pp}$, $D \in \mathbb{R}^{n\times p}$).

$X^T = VDU^T$

Therefore:

$Z = X V_{1:k}$

$\hat{\beta}_{PCA} = V_{1:k} [(Z^T Z)^{-1} Z^T y]$

$= V \Delta_{PCA} U^T y$

Δ_{PCA} is matrix with $\lambda_i^{-\frac{1}{2}}$ on the diagonal if $i \le k$, 0 elsewhere

$\Delta_{PCA} = \begin{pmatrix} \lambda_1^{-\frac{1}{2}} & & \\ & \ddots & \\ & & \lambda_k^{-\frac{1}{2}} \end{pmatrix}$

$X = UDV^T$

$X = UDV^T$

$n \times p$ $\quad p \times k$

$V_{1:k} = [v_1 \cdots v_k]$

$= V_{1:k} \left[(UDV^T V_{1:k})^T UDV^T V_{1:k} \right]^{-1} (UDV^T V_{1:k})^T y$

$= V_{1:k} \left(V_{1:k}^T V D^T U^T U D V^T V_{1:k} \right)^{-1} V_{1:k}^T V D^T U^T y$

$= \underbrace{V_{1:k}}_{p\times k} \underbrace{\left((V_{1:k}^T V) D^T D V^T V_{1:k} \right)^{-1}}_{k\times k} \underbrace{V_{1:k}^T V D^T U^T}_{k\times n} \underbrace{y}_{n\times n}$

$= V_{1:k} (V_{1:k}^T V D^T D V^T V_{1:k})^{-1} V_{1:k}^T V D^T U^T y$

$= V_{1:k} V_{1:k}^T (V D^T D V^T)^{-1} V_{1:k} V_{1:k}^T V D^T U^T y$

$= V (D^T D)^{-1} V^T V D^T U^T y$

$= V (D^T D)^{-1} D^T U^T y$

(b) (ii)

$$\hat{\beta}_{L2} = (X^T X + \alpha I)^{-1} X^T y$$

Since $X = UDV^T$

$$\hat{\beta}_{L2} = (VD^T U^T UDV^T + \alpha I)^{-1} VD^T U^T y$$

$$= (VD^T DV^T + \alpha I)^{-1} VD^T U^T y$$

Let $\Gamma = diag(\lambda_i + \alpha)$

$D^T D = (n-1)\Lambda = diag(\lambda_i)$

$$= V\Gamma V^T)^{-1} VD^T U^T y$$

$$= V\Gamma^{-1} V^T VD^T U^T y$$

$$= V \underbrace{\Gamma^{-1} D^T} U^T y$$

Δ_{L2}

Δ_{L2} has diagonal $\frac{\sqrt{\lambda_i}}{\lambda_i + \alpha}$

$S = V\Lambda V^T$, $\Lambda = diag(\lambda_i)$

$(n-1)S = X^T X = VD^T DV^T$, $D^T D = \Lambda$

$$\begin{pmatrix} & \\ & \end{pmatrix}\begin{pmatrix} & \\ & \end{pmatrix} = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

$p \times n$ $\quad$ $n \times p$

Q2. (a) $R(f) = \mathbb{E}[(Y - f(X))^2]$

$$= \mathbb{E}[Y^2 - 2f(X)Y + f^2(X)]$$

$$= \mathbb{E}_X[\mathbb{E}(Y^2 - 2f(X)Y + f^2(X) \mid X)]$$

$$= \mathbb{E}_X[\mathbb{E}(Y^2 \mid X) - 2f(X)\mathbb{E}(Y \mid X) + f^2(X)]$$

$$= \mathbb{E}_X[\mathrm{Var}(Y \mid X) + (\mathbb{E}(Y \mid X) - f(X))^2]$$

$\Rightarrow$ $f_*(X) = \mathbb{E}(Y \mid X)$

Therefore $R(f) - R(f_*) = \mathbb{E}[-2Yf + Y^2 + 2Yf_* - f_*^2]$

$$= \mathbb{E}(-2ff_* + f^2 + f_*^2) = \mathbb{E}[(f - f_*)^2]$$

$$\mathbb{E}(Yf) = \mathbb{E}\,\mathbb{E}(Yf(X)) = \mathbb{E}(ff_*)$$

$\Rightarrow$ $R(f) = R(f_*) + \mathbb{E}[(f - f_*)^2]$

$$= R(f_*) + \mathrm{Var}(f) + (\mathbb{E}(f) - f_*)^2.$$

$R(f_*)$: This is the risk that can't be reduced.

$$\mathbb{E}[(f - f_*)^2] = \mathbb{E}(f^2 - 2ff_* + f_*^2)$$

$$= \mathrm{Var}(f) + (\mathbb{E}(f) - f_*)^2$$

(b)

Part B

Course title: SB2b - Statistical Machine Learning
Lecturer: Pier Francesco Palamara
Checked by: Dino Sejdinovic

Date of this version: 8/03/2019

Do not turn this page until you are told that you may do so

1. (a) [12 marks] Consider a data set $\{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$. You may opt to compactly represent the data set using an $n \times p$ matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, whose rows $\mathbf{x}_i$ represent the points, and assume the data is mean-centered: $\sum_i \mathbf{X}_{ij} = 0, \forall j$. Consider the covariance matrix estimated using $\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^\top = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$. You decide to run principal components analysis (PCA) using $\mathbf{S}$.

(i) Assume we have obtained the first principal component $\mathbf{v}_1$. Formulate the optimization problem that should be solved to derive the *second* principal component $\mathbf{v}_2$, and solve it using the method of Lagrange multipliers.
Answer: (B/S) 6 marks.
We require that $\mathbf{v}_2$ has unit norm (i.e. $\mathbf{v}_2^\top\mathbf{v}_2 = 1$), projections to first and second PCs are uncorrelated (i.e. $\text{cov}(\mathbf{X}v_1, \mathbf{X}v_2) = \mathbf{v}_2^\top\mathbf{S}\mathbf{v}_1 = \lambda_1\mathbf{v}_2^\top\mathbf{v}_1 = 0$), and the variance $\mathbf{v}_2^\top\mathbf{S}\mathbf{v}_2$ of the data projection is maximized. Specifically, the problem is:

$$\begin{aligned} &\text{maximize} && \mathbf{v}_2^\top\mathbf{S}\mathbf{v}_2 \\ &\text{subject to :} && \mathbf{v}_1^\top\mathbf{v}_1 = 1 \;\text{ and }\; \lambda_1\mathbf{v}_2^\top\mathbf{v}_1 = 0. \end{aligned}$$

The Lagrangian form is

$$L(\mathbf{v}_2) = \mathbf{v}_2^\top\mathbf{S}\mathbf{v}_2 - \delta(\mathbf{v}_2^\top\mathbf{v}_2 - 1) - \gamma(\mathbf{v}_2^\top\mathbf{v}_1 - 0).$$

We find the value of $\mathbf{v}_2$ satisfying

$$\Delta_{\mathbf{v}_2} L(\mathbf{v}_2) = 0$$

so we need

$$2\mathbf{S}\mathbf{v}_2 - 2\delta\mathbf{v}_2 - \gamma\mathbf{v}_1 = 0.$$

We pre-multiply by $\mathbf{v}_1^\top$, obtaining

$$2\mathbf{v}_1^\top S\mathbf{v}_2 - 2\delta\mathbf{v}_1^\top\mathbf{v}_2 - \gamma\mathbf{v}_1^\top\mathbf{v}_1 = 0.$$

We know that $\mathbf{v}_1$ has unit norm and we require $\mathbf{v}_2$ orthogonal to $\mathbf{v}_1$, thus

$$2\mathbf{v}_1^\top\mathbf{S}\mathbf{v}_2 - 0 - \gamma = 0 \quad\rightarrow\quad \gamma = 2\mathbf{v}_1^\top\mathbf{S}\mathbf{v}_2 = 2\mathbf{v}_2^\top\mathbf{S}\mathbf{v}_1$$

We also require uncorrelated projections: $\mathbf{v}_2^\top\mathbf{S}\mathbf{v}_1 = 0$, so $\gamma = 0$. Plugging this in:

$$2\mathbf{S}\mathbf{v}_2 - 2\delta\mathbf{v}_2 - \gamma\mathbf{v}_1 = 0 \quad\rightarrow\quad \mathbf{S}\mathbf{v}_2 = \delta\mathbf{v}_2$$

So δ must be an eigenvalue of $\mathbf{S}$ with corresponsing eigenvector $\mathbf{v}_2$. Since we are maximizing and we have excluded the first eigenvector, we choose the second largest eigenvalue $\delta = \lambda_2$.

(ii) Now assume you have obtained the eigendecomposition $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_p]$ is a $p \times p$ orthogonal matrix whose columns are the eigenvectors for $\mathbf{S}$ and $\mathbf{\Lambda}$ is a $p \times p$ diagonal matrix whose diagonal entries are the eigenvalues $\{\lambda_i\}_{i=1}^p$ of $\mathbf{S}$. Assume that $\{\lambda_i\}_{i=1}^p$ are distinct and sorted in decreasing order on the diagonal of $\mathbf{\Lambda}$. Let $Tr(\mathbf{S})$ represent the trace of the matrix $\mathbf{S}$, and assume that $Tr(\mathbf{S}) = 1$. Define $r = \sum_{i=k+1}^p \lambda_i$. What does r represent? Running PCA suggests that k dimensions will be enough to capture 94% of the variance in your data set $\mathbf{X}$. What is the value of r?.
Answer: (S) 3 marks.
The quantity $Tr(\mathbf{S})^{-1}\sum_{i=k+1}^p \lambda_i$ is the "reconstruction error", the squared euclidean distance between original data and the data reconstructed after projecting to

k dimensions and then projecting back to p dimensions. Here $Tr(\mathbf{S})^{-1} = 1$, so r is the reconstruction error. We know that $Tr(\mathbf{S}) = Tr(\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top) = Tr(\mathbf{V}^\top\mathbf{V}\mathbf{\Lambda}) = Tr(\mathbf{\Lambda}) = \sum_{i=1}^{p}\lambda_i = 1$ is the total variance, thus and $Tr(\mathbf{S})^{-1}\sum_{i=1}^{k}\lambda_i = 0.94$. So $r = \sum_{i=k+1}^{p}\lambda_i = (1 - 0.94) = 0.06$.

(iii) Let $\mathbf{V}_{1:k} = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$, $k < p$ be the matrix of k eigenvectors coreponsding to the largest k eigenvalues. Consider the vector $\mathbf{b}_i = \mathbf{V}_{1:k}\mathbf{V}_{1:k}^\top\mathbf{x}_i$. Write an equation representing the relationship between $\{\mathbf{x}_i\}_{i=1}^n$, $\{\mathbf{b}_i\}_{i=1}^n$, and r (you do not need to provide a proof).

Answer: (B) 3 marks.

$\mathbf{b}_i = \mathbf{V}_{1:k}\mathbf{V}_{1:k}^\top\mathbf{x}_i$ is the reconstructed vector. We have

$$\sum_{i=1}^{n}\|\mathbf{x}_i - \mathbf{b}_i\|_2^2 = (n-1)\sum_{j=k+1}^{p}\lambda_j = (n-1)Tr(\mathbf{S})r.$$

(Students may also set $Tr(\mathbf{S}) = 1$.)

(b) [13 marks] Now consider the same data set, with the addition of labels: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$, and $y_i \in \mathbb{R}$. Again, you may opt to compactly represent the data set using an $n \times p$ matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, whose rows $\mathbf{x}_i$ represent the training points, and a vector $\mathbf{y}$, whose entries y_i represent the labels. You would like to perform regularized linear regression, and assume the data comes from a linear model $y = \boldsymbol{\beta}^\top\mathbf{x} + \epsilon$, where ϵ is a noise term (we omit the intercept for simplicity, so that it does not need to be considered separately for regularization). Rather than using a direct regularization approach, however, you decide to perform "PCA-regression" performing the following operations:

1. Compute the matrix $\mathbf{Z} \in \mathbb{R}^{n\times k}$ of k-dimensional PCA projections $\mathbf{Z} = \mathbf{X}\mathbf{V}_{1:k}$.
2. Perform ordinary least squares on the projected data: $\widehat{\boldsymbol{\beta}}_Z = \left(\mathbf{Z}^\top\mathbf{Z}\right)^{-1}\mathbf{Z}^\top y$.
3. Transform parameters to original coordinates: $\widehat{\boldsymbol{\beta}}_{PCA} = \mathbf{V}_{1:k}^\top\widehat{\boldsymbol{\beta}}_Z$.
4. Given a new data point $\mathbf{x}_{new}$, predict its label using $\widehat{y}_{new} = \widehat{\boldsymbol{\beta}}_{PCA}^\top\mathbf{x}_{new}$.

(i) Show that we can write $\widehat{\boldsymbol{\beta}}_{PCA} = \mathbf{V}\mathbf{\Delta}_{PCA}\mathbf{U}^\top\mathbf{y}$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and the matrix $\mathbf{\Delta}_{PCA}$ is 0 everywhere except on the diagonal. Diagonal elements of $\mathbf{\Delta}_{PCA}$ are a function k and λ_i, the eigenvalues of $\mathbf{S}$. [Hint: you may want to use the SVD decomposition $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$.]

Answer: (S) 5 marks.

Using the SVD, we have

$$\mathbf{Z}_k = \mathbf{X}\mathbf{V}_{1:k}$$
$$\widehat{\boldsymbol{\beta}}_{PCA} = \mathbf{V}_{1:k}\left[\left(\mathbf{Z}^\top\mathbf{Z}\right)^{-1}\mathbf{Z}^\top y\right] = \mathbf{V}\mathbf{\Delta}_{PCA}\mathbf{U}^\top y,$$

where $\mathbf{\Delta}_{PCA}$ is a matrix with $\lambda_i^{-1/2}$ on the diagonal if $i \leqslant k$, and 0 elsewhere. (Students may want to use $\mathbf{V}_{1:k} \in \mathbb{R}^{n\times k}$, in which case they should specify $\mathbf{\Delta}_{PCA} \in \mathbb{R}^{k\times k}$ and $\mathbf{U} \in \mathbb{R}^{k\times n}$. Other choices of matrix dimensions are possible but should be specified.)

(ii) Show that performing regression using this approach is closely related to performing $L2$-regularized linear regression, which uses the estimator $\widehat{\boldsymbol{\beta}}_{L2} = \left(\mathbf{X}^\top\mathbf{X} + \alpha\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y}$, where $\alpha \geqslant 0$. Specifically, show that we can write $\widehat{\boldsymbol{\beta}}_{L2} = \mathbf{V}\mathbf{\Delta}_{L2}\mathbf{U}^\top\mathbf{y}$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and the matrix $\mathbf{\Delta}_{L2}$ is 0 everywhere except on the diagonal, where elements are a function of λ_i and α. [Hint: you may again want to

use the SVD decomposition.]

Answer: **(N)** 6 marks.

Using the SVD, and defining $\mathbf{\Gamma} = \text{diag}\,(\lambda_i + \alpha)$

$$\begin{aligned}
\widehat{\boldsymbol{\beta}}_{L2} &= (\mathbf{X}^\top\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \\
&= (\mathbf{V}\mathbf{\Gamma}\mathbf{V}^\top)^{-1}\mathbf{V}\mathbf{D}\mathbf{U}^\top\mathbf{y} \\
&= \mathbf{V}\mathbf{\Gamma}^{-1}\mathbf{V}^\top\mathbf{V}\mathbf{D}\mathbf{U}^\top\mathbf{y} \\
&= \mathbf{V}\mathbf{\Gamma}^{-1}\mathbf{D}\mathbf{U}^\top\mathbf{y} \\
&= \mathbf{V}\mathbf{\Delta}_{L2}\mathbf{U}^\top\mathbf{y}
\end{aligned}$$

where $\mathbf{\Delta}_{L2}$ has diagonal elements given by $\frac{\sqrt{\lambda_i}}{\lambda_i+\alpha}$.

(iii) By comparing $\mathbf{\Delta}_{L2}$ and $\mathbf{\Delta}_{PCA}$, comment on the differences and similarities of these two methods for regularized linear regression.

Answer: **(N)** 2 marks.

"PC-regression" can be seen as a hard-thresholding version of ridge regression, which instead applies a smooth penalty to components with decreasing variance.

2. (a) [7 marks] Consider performing regression using the squared loss. Formally, you are looking for a function f that minimizes the risk

$$R(f) = \mathbb{E}\left[(Y - f(X))^2\right].$$

(i) Show that the risk may be decomposed as

$$R(f) = (\text{noise}) + (\text{variance}) + (\text{bias})^2,$$

Find an explicit expression for each term, and provide a brief description of how they are affected by the choice of learning algorithm and its parameters. You can assume without proving it that the optimal choice of f in this setting is given by the conditional mean $f_*(\mathbf{x}) = \mathbb{E}[Y|X = \mathbf{x}]$.

Answer: (B) 7 marks.

We have

$$R(f) = \mathbb{E}_X \mathbb{E}\left[(Y - f(X))^2 \middle| X\right],$$

and

$$\begin{aligned}\mathbb{E}\left[(Y - f(X))^2 \middle| X = \mathbf{x}\right] &= \mathbb{E}\left[Y^2 \middle| X = \mathbf{x}\right] - 2f(x)\,\mathbb{E}[Y|\,X = \mathbf{x}] + f(\mathbf{x})^2 \\ &= \text{Var}[Y|X = \mathbf{x}] + (\mathbb{E}[Y|\,X = \mathbf{x}] - f(\mathbf{x}))^2.\end{aligned}$$

We know that the optimal choice of f is given by $f_*(\mathbf{x}) = \mathbb{E}[Y|X = \mathbf{x}]$. The term $\text{Var}[Y|X = \mathbf{x}]$ is the noise, it cannot be removed by the choice of f. We now decompose the residual risk:

$$R(f) - \mathbb{E}_X[\text{Var}[Y|X = \mathbf{x}]] = \mathbb{E}_X\left[(f_*(x) - f(\mathbf{x}))^2\right].$$

This residual risk will depend on data set size, which we now make explicit in the notation. For a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and $f^{(\mathcal{D})}$ trained on $\mathcal{D}$ using a model (hypothesis class) $\mathcal{H}$, denote the average model by $\overline{f}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}} f^{(\mathcal{D})}(\mathbf{x})$. Then

$$\mathbb{E}_{\mathcal{D}}\left[\left(f_*(X) - f^{(\mathcal{D})}(X)\right)^2\right] = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(f^{(\mathcal{D})}(X) - \overline{f}(X)\right)^2\right]}_{\text{Var}_X(\mathcal{H},n)} + \underbrace{\left(\overline{f}(X) - f_*(X)\right)^2}_{\text{Bias}^2_X(\mathcal{H},n)}.$$

The above follows by expanding the square then adding and removing $\overline{f}$. So the risk may be decomposed into noise, variance, and squared bias. A flexible model (e.g. regression with polynomial basis expansions of large degree) will be able to accurately (over-)fit a data set, but this will result in a large error component due to the variance term. On the other hand, insufficiently flexible models (e.g. regression with polynomial basis expansions of low degree) will result in error due to high bias, but will have low variance.

(b) [6 marks] You are given a data set $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{+1, -1\}$, and a *weak learner* $h(\cdot)$ that performs binary classification $h(\mathbf{x}) \in \{+1, -1\}$. Consider a set of weights $w_t(i)$, where $i \in \{1 \dots N\}$ is the index of a data point and t indicates the algorithm's iteration. The AdaBoost algorithm is trained using the following procedure:

- Initialize weights $w_0(i) = \frac{1}{N}, \forall i \in \{1 \dots N\}$.
- For $t = 1$ to T:

1. Train a weak classifier $h_t(\cdot)$ by minimizing the weighted classification error ϵ_t (using the current set of weights).
2. Compute the contribution of $h_t(\cdot)$, denoted by β_t.
3. Update weights for each point, using the rule

$$w_t(i) = \frac{1}{Z_t} w_{t-1}(i) e^{-\beta_t y_i h_t(\mathbf{x}_i)},$$

where Z_t is chosen such that $\sum_{i=1}^N w_t(i) = 1$.

- Output $\{\beta_t\}_{t=1}^T$ and $\{h_t(\cdot)\}_{t=1}^T$.

(i) What is a *weak* learner in this algorithm? (you do not need to provide mathematical detail). **Answer:** **(B)** 1 marks.
A weak classifier (or learner) in this context is a classifier that performs only slightly better than a random classifier.

(ii) Provide an explicit expresion for ϵ_t and β_t computed in steps 1 and 2 of the algorithm. **Answer:** **(B)** 3 marks.

$$\epsilon_t = \sum_n w_t(i) \mathbb{I}[y_i \neq h_t(\mathbf{x}_i)]$$

$$\beta_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$$

(iii) How does AdaBoost classify a new point $\mathbf{x}_{new}$? **Answer:** **(B)** 2 marks.

$$H(\mathbf{x}_{new}) = \text{sign}\left[F(\mathbf{x}_{new})\right] = \text{sign}\left[\sum_{t=1}^T \beta_t h_t(\mathbf{x}_{new})\right]$$

(c) [12 marks] Consider the AdaBoost algorithm described in part (b).

(i) Show that

$$w_T(i) = \frac{1}{N} \frac{e^{-y_i \sum_{t=1}^T \beta_t h_t(\mathbf{x}_i)}}{\prod_{t=1}^T Z_t}$$

Answer: **(S)** 3 marks.

$$\begin{aligned} w_T(i) &= \frac{1}{Z_T} w_{t-1}(i) e^{-\beta_t y_i h_t(\mathbf{x}_i)} \\ &= \frac{1}{N} \frac{1}{\prod_{t=1}^T Z_t} \prod_{t=1}^T e^{-\beta_t y_i h_t(\mathbf{x}_i)} \\ &= \frac{1}{N} \frac{e^{-y_i \sum_{t=1}^T \beta_t h_t(\mathbf{x}_i)}}{\prod_{t=1}^T Z_t} \end{aligned}$$

(ii) Let $H(\cdot)$ be the final AdaBoost classifier, and define its training error as

$$E = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[H(\mathbf{x}_i) \neq y_i].$$

Show that $E \leqslant \prod_{t=1}^{T} Z_t$. [Hint: $e^{-c} \geqslant 1$ if $c \leqslant 0$ and $e^{-c} > 0$ if $c > 0$.]
Answer: **(N)** 4 marks.
Let $F(\mathbf{x}) = \sum_{t=1}^{T} \beta_t h_t(\mathbf{x})$. Using the hint, the training error is

$$\begin{aligned} E &= \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[F(\mathbf{x}_i) y_i < 0] \\ &< \frac{1}{N} \sum_{i=1}^{N} e^{-F(\mathbf{x}_i) y_i < 0} \end{aligned}$$

We know from the previous results that

$$w_T(i) = \frac{1}{N} \frac{e^{-y_i \sum_{t=1}^{T} \beta_t h_t(\mathbf{x}_i)}}{\prod_{t=1}^{T} Z_t}.$$

Rearranging

$$w_T(i) \prod_{t=1}^{T} Z_t = \frac{1}{N} e^{-y_i F(\mathbf{x}_i)}.$$

Substituting in the above expression for E,

$$E \leqslant \sum_{i=1}^{N} w_T(i) \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t \underbrace{\sum_{i=1}^{N} w_T(i)}_{=1} = \prod_{t=1}^{T} Z_t.$$

(iii) Use these results to show that $E \leqslant \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)}$
Answer: **(N)** 5 marks.
$E \leqslant \prod_{t=1}^{T} Z_t$ and Z_t is defined such that $\sum_{i=1}^{N} w_t(i) = 1$, so

$$\begin{aligned} Z_t &= \sum_{i=1}^{N} w_{t-1}(i) e^{-\beta_t y_i h_t(\mathbf{x}_i)} \\ &= \sum_{i:y_i = h_t(\mathbf{x}_i)}^{N} w_{t-1}(i) e^{-\beta_t} + \sum_{i:y_i \neq h_t(\mathbf{x}_i)}^{N} w_{t-1}(i) e^{\beta_t} \\ &= e^{-\beta_t} \underbrace{\sum_{i:y_i = h_t(\mathbf{x}_i)}^{N} w_{t-1}(i)}_{=1-\epsilon_t} + e^{\beta_t} \underbrace{\sum_{i:y_i \neq h_t(\mathbf{x}_i)}^{N} w_{t-1}(i)}_{=\epsilon_t} \\ &= e^{-\beta_t}(1-\epsilon_t) + e^{\beta_t}(\epsilon_t) \\ &= e^{-\frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})}(1-\epsilon_t) + e^{\frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})}(\epsilon_t) \\ &= \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}(1-\epsilon_t) + \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}(\epsilon_t) \\ &= \sqrt{\epsilon_t(1-\epsilon_t)} + \sqrt{(1-\epsilon_t)\epsilon_t} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)}. \end{aligned}$$

3. (a) [14 marks] Figure 2-a shows three concentric circles of radii $r_1 = 1$, $r_2 = 2$ and $r_3 = 3$. Assume a random set of points with positive label is sampled at distance $d < 1$ from the origin, and a second set of random points with negative label is sampled at distance $2 < d < 3$ from the origin. An example data set is shown in Figure 2-b.

(i) For each classification algorithm listed below, state whether you would be able to obtain perfect classification accuracy on the training data set, and provide a brief commentary for why this may or may not be possible. Ignore any issues related to convergence (e.g. due to local minima/maxima). In each case, draw a rough sketch of a data set like the one in Figure 2-b (or simply draw circles as in 2-a), and sketch the decision boundaries that you think the algorithm may produce. (State any assumptions you may introduce. There is no need to provide mathematical detail or numerical results.)

1. Quadratic discriminant analysis.
2. K-nearest neighbors with $K = 1$.
3. Decision tree (the same feature may be used multiple times to split the data).
4. Logistic regression with inputs $\{x_1, x_2\}$.
5. An artificial neural network with inputs $\{x_1, x_2\}$, one hidden layer containing 2 neurons, and sigmoid activation functions.
6. An artificial neural network with inputs $\{x_1, x_2\}$, one hidden layer containing 3 neurons, and sigmoid activation functions.

Answer: **(B/S)** 10 marks.

See Figure 3 for decision boundaries.

1. Yes. Classes have same mean and diagonal covariance matrix that is a scaling of the identity matrix. One class has larger scaling than the other.
2. Yes, by definition training is storing whole data set, and nearest neighbor is the point itself.
3. Yes, the data can be separated using four decision stumps.
4. No, the model is linear. We would need a basis expansion, but the input is $\{x_1, x_2\}$. Any drawn linear decision boundary would be OK.
5. No, we don't have enough flexibility when combining two linear boundaries.
6. Yes, we do have enough flexibility when combining three linear boundaries, getting a triangle that is somewhat smoothed at the corners.

(ii) Draw a tree that may be obtained running the decision tree algorithm on the data set in Figure 2-b (the same feature may be used multiple times to split the data). Indicate which feature and value is chosen for each split, and the classification rule used in each leaf (e.g. classify as positive/negative). You do not need to prune the tree or limit its depth.

Answer: **(S)** 4 marks.

See Figure 4

(b) [11 marks] Let $B = \{\mathbf{x} | \sqrt{\mathbf{x}^\top \mathbf{x}} \leqslant 1, \mathbf{x} \in \mathbb{R}^p\}$ be a $p-$dimensional unit ball centered at the origin. Consider a training set consisting of N samples, which are drawn uniformly at random from B (i.e. the probability that $\mathbf{x} \in A \subseteq B$ is proportional to the volume of A). Consider a point $\mathbf{x}_0$ located at the origin. The distance to its nearest neighbor is

$$d = \min_{i \in \{1 \ldots N\}} \sqrt{\mathbf{x}_i^\top \mathbf{x}_i}$$

(i) Derive the cumulative distribution function (cdf) for d, which is given by $P(d \leqslant t)$ for $0 \leqslant t \leqslant 1$, in the special case of $p = 1$.

Answer: **(S/N)** 3 marks.

$$\begin{aligned} P(d \leqslant t) &= 1 - P(d > t) \\ &= 1 - P(|x_i| > t), \forall i \in \{1 \dots N\} \\ &= 1 - \prod_{i=1}^{n} P(|x_i| > t) \\ &= 1 - (1-t)^N. \end{aligned}$$

(ii) Derive $P(d \leqslant t)$ for the more general case of any positive integer value of p.
[Hint: the volume of a $p-$dimensional sphere of radius r is given by $\frac{(r\sqrt{\pi})^p}{\Gamma(p/2+1)}$, where $\Gamma(\cdot)$ indicates the Gamma function.]
Answer: **(S/N)** 3 marks.

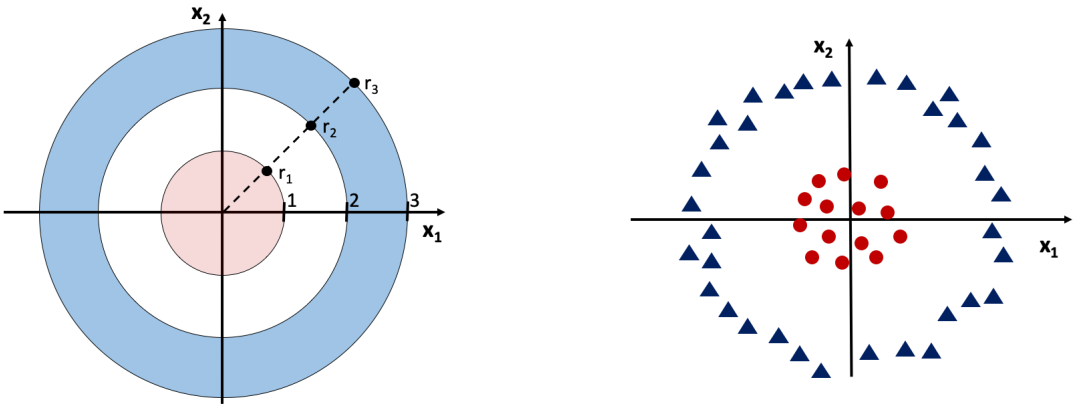
$$\begin{aligned} P(d \leqslant t) &= 1 - P(d > t) \\ &= 1 - P\left(\sqrt{\mathbf{x}_i^\top \mathbf{x}_i} > t, \forall i \in \{1 \dots N\}\right) \\ &= 1 - \prod_{i=1}^{N} P(\sqrt{\mathbf{x}_i^\top \mathbf{x}_i} > t) \\ &= 1 - \left(\frac{(\sqrt{\pi})^p - (t\sqrt{\pi})^p}{(\sqrt{\pi})^p}\right)^N \\ &= 1 - (1-t^p)^N. \end{aligned}$$

(iii) Assume you may obtain a data set of arbitrary size n. How many points do you need, as a function of p, to ensure that with probability 0.9 at least one point is within a distance of 0.1 from the origin?
Answer: **(S)** 3 marks.

$$\begin{aligned} 0.9 &= 1 - (1 - 0.1^p)^N \\ \log(0.1) &= N \log(1 - 0.1^p) \\ N &= \frac{\log(0.1)}{\log(1 - 0.1^p)} \\ N &\approx \frac{\log(0.1)}{-0.1^p} = \log(10)10^p. \end{aligned}$$

(The student should comment on the fact that N grows very quickly wirth p, wither here or in the next part. This is worth one mark, here or below.)

(iv) How are these calculations related to the *curse of dimensionality*? How will this affect the $K-$nearest neighbors algorithm?
Answer: **(S)** 2 marks.
This is a manifestation of the *curse of dimensionality*, whereby the required training data set size grows too quickly as the problem dimensionality is increased (in the above, N grows very quickly with p.). $K-$NN will not be "local" any more, violating the key assumption made to generalize when predicting the label of new points. The student may also argue that for $K-$NN to be effective in high dimensions we will need an exceedingly large training data set.



(a) (b)

Figure 2: Problem 2 data set. (a) Positive points are sampled at distance $d < 1$ from the origin, negative points are sampled at distance $2 < d < 3$ from the origin. (b) A sample data set where red circles indicate positive points, blue triangles indicate negative points.

Answer:

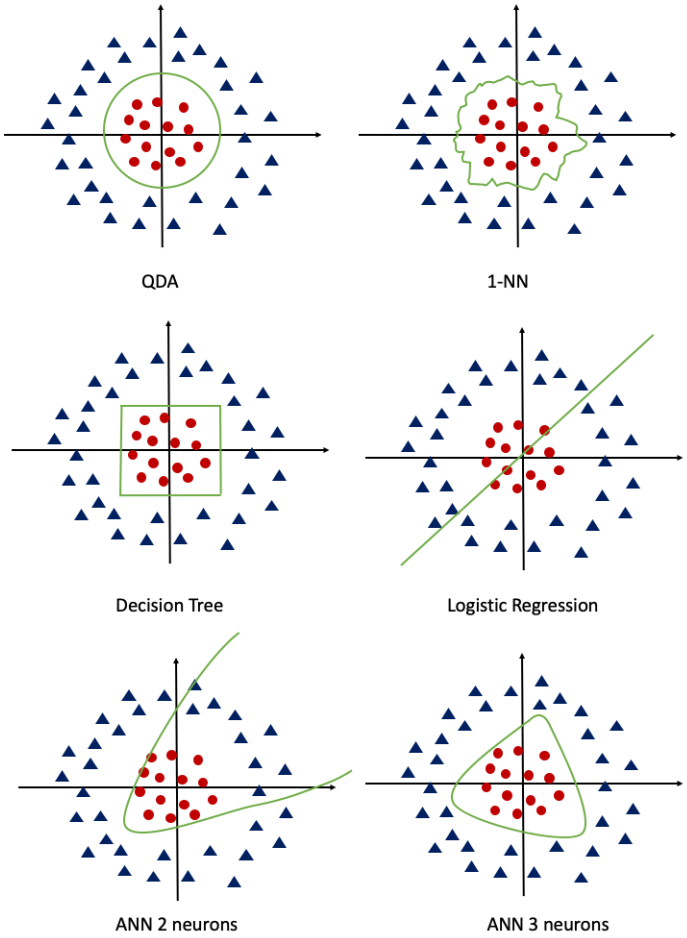


Figure 3: Sketch decision boundaries for each algorithm (data points are a rough reproduction of Figure 2-b).

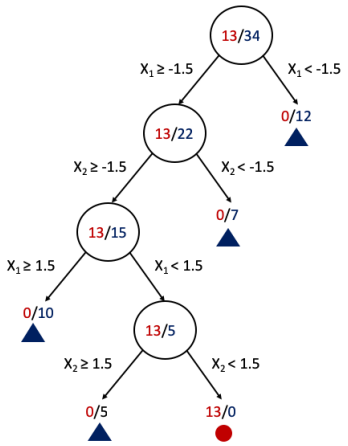


Figure 4: Sketch of a regression tree for the given data set.

A15677W1

SECOND PUBLIC EXAMINATION

Honour School of Mathematics and Statistics Part B: Paper SB2.2

Honour School of Mathematics Part B: Paper SB2.2

STATISTICAL MACHINE LEARNING

TRINITY TERM 2020

Wednesday 10 June

Opening time: 09:30 (BST)

You have 2 hours 45 minutes to complete the paper and upload your answer file

You may submit answers to as many questions as you wish but only the best two will count for the total mark. All questions are worth 25 marks.

You should ensure that you observe the following points:

1. Write with a black or blue pen OR with a stylus on a tablet (colour set to black or blue).
2. On the first page, write
   - your candidate number
   - the paper code
   - the paper title
   - and your course title (e.g. Mathematics and Statistics Part B)
   - but ***do not*** enter your name or college.
3. For each question you attempt,
   - start writing on a new sheet of paper,
   - indicate the question number clearly at the top of each sheet of paper,
   - number each page.
4. Before scanning your work,
   - add to the first page, in numerical order, the question numbers attempted,
   - cross out all rough working and any working you do not want to be marked,
   - and orient all scanned pages the same way.
5. Submit a single PDF document with your answers for this paper

If you do not attempt any questions at all on this paper, you should still submit a single page indicating that you have opened the exam but not attempted any questions. Please make sure to write your candidate number on this single page.

1. (a) [8 marks] Consider the three binary classifiers described below, which are trained with a data set $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^p = [x_{i,1}, \ldots, x_{i,p}]^\top$ and $y_i \in \{0, 1\}$. After training a classifier C and learning the optimal parameter values $\widehat{\theta}_C$, we can obtain an estimate of the probability for the class label of a point x, which is given by

$$P(y = k|x, \widehat{\theta}_C), \quad k \in \{0, 1\}.$$

   1. `NBGauss`, a Naive Bayes classifier assuming Gaussian features $x_i|y_i \sim N(\mu_{y_i}, \sigma^2)$. Note that each class may have a different mean, but the variance is the same for both classes. `NBGauss` learns its parameters using maximum likelihood.
   2. `NNLin`, a neural network composed of a single neuron with logistic activation function $s(a) = 1/[1 + \exp(-a)]$. The network takes vectors $x_i = [x_{i,1}, \ldots, x_{i,p}, 1]^\top$ as an input and is trained by minimizing the log-loss

$$L(\widehat{y}_i, y_i) = -\sum_{i=1}^{n} \left[y_i \log(\widehat{y}_i) + (1 - y_i) \log(1 - \widehat{y}_i) \right].$$

      where $\widehat{y}_i = P(y_i = 1|x_i, \widehat{\theta})$.
   3. `NNQuad`, which is equivalent to `NNLin` but takes $x_i = [x_{i,1}, \ldots, x_{i,p}, x_{i,1}^2, \ldots, x_{i,p}^2, 1]^\top$ as an input, i.e. it takes all elements of the data vector, in addition to each element squared, and a bias term.

   (i) After training each classifier $C \in \{\texttt{NBGauss}, \texttt{NNLin}, \texttt{NNQuad}\}$, we may decide to measure its performance on the *training* data set using

$$\ell(C) = -\sum_{i=1}^{n} \left[y_i \log(\widehat{y}_{i,C}) + (1 - y_i) \log(1 - \widehat{y}_{i,C}) \right].$$

   where $\widehat{y}_{i,C} = P(y_i = 1|x_i, \widehat{\theta}_C)$. Let us say that a model C_1 outperforms (in training) model C_2 if $\ell(C_1) \leqslant \ell(C_2)$ for any possible training set.
   For each pair of models in {`NBGauss`, `NNLin`, `NNQuad`}, state whether one model outperforms the other, and briefly explain why. You may presume that all numerical optimization methods will converge to a global optimum.

   (ii) Under which circumstances (e.g. linked to data generating process and training data set size) do you expect `NBGauss` to have a lower *test* error than `NNLin`, and vice versa?

   (b) [17 marks] Consider an L2-regularized regression model with squared loss, data matrix $X \in \mathbb{R}^{n \times p}$, and responses $y \in \mathbb{R}^n$. The empirical risk minimization problem (ignoring constants and the intercept) is given by

$$R(\beta_{L2}) = \|y - X\beta_{L2}\|_2^2 + \lambda \|\beta_{L2}\|_2^2.$$

   (i) Derive an optimal solution $\widehat{\beta}_{L2}$ which uses the matrix $X^\top X \in \mathbb{R}^{p \times p}$ (there is no need to check that this is a globally optimal solution).

   (ii) Show that the optimal solution may be written as $\widehat{\beta}'_{L2} = X^\top (XX^\top + \lambda I)^{-1} y$, where I is the $n \times n$ identity matrix.

   (iii) Computing $\widehat{\beta}_{L2}$ and $\widehat{\beta}'_{L2}$ will require performing several operations. For instance, computing $M^\top v$ where $M \in \mathbb{R}^{n \times p}$ and $v \in \mathbb{R}^n$ requires $n \times p$ operations. Provide an estimate of the cost of computing $\widehat{\beta}_{L2}$ and $\widehat{\beta}'_{L2}$ as a function of n and p. Assume that both n and p are large, but p is substantially larger than n. Would it be computationally cheaper to compute $\widehat{\beta}_{L2}$ or $\widehat{\beta}'_{L2}$? Justify your answer.

(iv) Assume we use a feature expansion $\phi : x_i \in \mathbb{R}^p \to \phi(x_i) \in \mathbb{R}^d$ and let $\Phi \in \mathbb{R}^{n \times d}$ be a matrix that has $\phi(x_i)^\top$ is in its i-th row. Given a new sample x_{new} we can construct the prediction $\widehat{y}_{new}$ as follows:

$$\begin{aligned}\widehat{y}_{new} &= y^\top(\Phi\Phi^\top + \lambda I)^{-1}\Phi\phi(x_{new}) \\ &= y^\top(K + \lambda I)^{-1}\kappa,\end{aligned}$$

where the $\{i, j\}$-th entry of the matrix $K \in \mathbb{R}^{n \times n}$ is given by $k(x_i, x_j) = \phi(x_i)^\top\phi(x_j)$ and the i-th element of the vector $\kappa \in \mathbb{R}^n$ is given by $k(x_i, x_{new}) = \phi(x_i)^\top\phi(x_{new})$. The function $k(\cdot, \cdot)$ is known as a *kernel* and is used to compute the similarity of two data points after applying the feature expansion. Assume that $p = 1$, so that $x_i \in \mathbb{R}$. We decide to use the Gaussian kernel

$$k(x_i, x_j) = \exp\left\{-\frac{1}{2}(x_i - x_j)^2\right\}.$$

Show that in this case we can write $k(x_i, x_j) = \phi(x_i)^\top\phi(x_j)$ where $\phi(x_i)$ and $\phi(x_j)$ are infinite-dimensional vectors. Provide an explicit expression for the entries of $\phi(x_i)$.
[*Hint: You may want to use the Taylor expansion of a real-valued function $f(x)$ about a point a, which is given by* $f(x) = f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \cdots$.]

2. (a) [5 marks] Consider the following image and three different convolutions of it:

(a) Original image

(b) Convolved image 1 (blurrier)

(c) Convolved image 2 (sharper)

(d) Convolved image 3

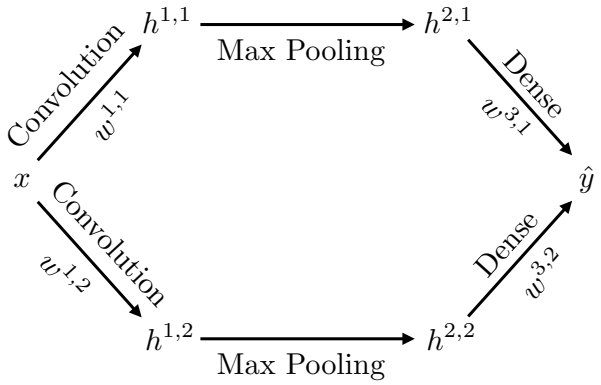
Each convolved image was generated by convolving with one of the following kernels (note one of the kernels is not used):

$$\text{(i)}: \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \text{(ii)}: \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$\text{(iii)}: \begin{bmatrix} -0.5 & -1 & -0.5 \\ -1 & 7 & -1 \\ -0.5 & -1 & -0.5 \end{bmatrix} \qquad \text{(iv)}: \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

For each convolved image, identify which kernel was used to produce it. Explain your reasoning.

(b) [15 marks] Let $(x_1, y_1), \ldots, (x_n, y_n)$, where each $x_i \in \mathbb{R}^{p\times 1}$ and $y_i \in \{0, 1\}$ $\forall i \in \{1, \ldots, n\}$, represent a set of input and output pairs constituting our training data with p-dimensional vector inputs and categorical outputs. Consider the following convolutional neural network (CNN) architecture:



such that the convolution outputs are given by

$$h^{1,1} = s_1(w^{1,1} \circledast x)$$
$$h^{1,2} = s_1(w^{1,2} \circledast x)$$

where $s_1(\cdot)$ is an activation function, $\circledast$ represents a convolution, and $w^{1,m} \in \mathbb{R}^{(2\tau+1)\times 1}$, $m \in \{1, 2\}$, for some integer τ. The max pooling outputs are given by

$$h_j^{2,1} = \max\left(h_{2j-1}^{1,1}, h_{2j}^{1,1}\right), \ \forall j \in \{1, \ldots, d\}$$
$$h_j^{2,2} = \max\left(h_{2j-1}^{1,2}, h_{2j}^{1,2}\right), \ \forall j \in \{1, \ldots, d\}$$

where d is half the number of hidden units in $h^{1,1}$ (and thus also $h^{1,2}$), and you can assume that d is an integer. The network output is given by

$$\widehat{y} = s_3\left(\sum_{j=1}^{d} w_j^{3,1} h_j^{2,1} + w_j^{3,2} h_j^{2,2}\right),$$

where $s_3(\cdot)$ is an activation function. Note that bias and regularization terms are deliberately omitted for simplicity.

(i) Explain why a mix of convolutional and max pooling layers in a CNN can form an effective architecture for dealing with image data.

(ii) Apply the chain rule to write down a set of recursive expressions for backpropagating gradients of the empirical risk

$$R_{\text{emp}} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{y}_i)$$

through the network. Note that in this part of the question you only need to express each gradient as a function of other gradients.

(iii) Presume that we use a log-loss function, a ReLu activation function for s_1, and a logistic activation function for s_3, namely that we take

$$\begin{aligned} L(y_i, \widehat{y}_i) &= -\left[y_i \log(\widehat{y}_i) + (1 - y_i) \log(1 - \widehat{y}_i)\right] \\ s_1(a) &= \max(a, 0) \\ s_3(a) &= \frac{1}{1 + \exp(-a)}. \end{aligned}$$

Derive explicit expressions for all the partial derivatives that are required to complete the backpropagation algorithm.

(c) [5 marks] Consider the risk of this CNN classifier when we train using stochastic gradient descent with random initialization of the weights, namely

$$R = \mathbb{E}\left[L\left(Y, \widehat{Y}(X)\right)\right]$$

where $L(\cdot, \cdot)$ is the log-loss, $\widehat{Y}(X)$ represents the output of the trained network given input X and this expectation is over all randomness in the process (including data generation). Consider further training an ensemble of T CNNs by independently training each with the same dataset but a different initialization—each of which is independently drawn from the same initialization distribution—and then using the empirical average of the predictions for each network as the overall prediction of the ensemble. Prove that the risk of this ensemble CNN,

$$R_e = \mathbb{E}\left[L\left(Y, \widehat{Y}_e(X)\right)\right],$$

is always less than or equal to R.

[*Hint: this result is not specific to the architecture but relies on a specific property of the loss function.*]

3. (a) [9 marks] Let $(x_1, y_1), \ldots, (x_n, y_n)$, where each $x_i \in \mathbb{R}^{p\times 1}$ and $y_i \in \{1, \ldots, K\}$ $\forall i \in \{1, \ldots, n\}$, represent a set of input and output pairs constituting our training data with p-dimensional vector inputs and categorical outputs. Presume that $K < p$ and let $n_k = \sum_{i=1}^{n} \mathbb{I}(y_i = k)$ denote the number of examples of each class, where we assume $n_k > 0$. We define the following terms:

Data mean	$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
Class means	$\mu_k = \frac{1}{n_k}\sum_{i:y_i=k} x_i = \frac{1}{n_k}\sum_{i=1}^{n} x_i \mathbb{I}(y_i = k)$
Total covariance	$S_t = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T$
Within class covariance	$S_w = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu_{y_i})(x_i - \mu_{y_i})^T$
Between class covariance	$S_b = \frac{1}{n-1}\sum_{k=1}^{K} n_k (\mu_k - \overline{x})(\mu_k - \overline{x})^T$.

   (i) Explain the motivation behind principal component analysis (PCA) and provide a formulation for the principal component directions as a solution to a constrained optimization problem in terms of a random input vector X.

   (ii) Given input data $x_1, \ldots, x_n$ such that we approximate $\mathrm{Cov}(X)$ with S_t, explain (without proof) how we can recover the principal components from S_t.

   (iii) Show that $S_t = S_w + S_b$.

   (b) [9 marks] The discriminant coordinates $v_1, \ldots, v_{K-1}$ of Fisher's discriminant analysis (FDA) are defined to solve the following series of optimization problems

$$v_i = \arg\max_{w} \frac{w^T S_1 w}{w^T S_2 w} \quad \text{subject to} \quad w^T S_w v_j = 0,\ \forall j \in \{1, \ldots, i-1\}$$

   where $S_1 = S_b$ and $S_2 = S_w$.

   (i) Explain the principles behind the objective in this formulation, taking care to discuss what each of the numerator and the denominator represent.

   (ii) Show that instead setting $S_1 = S_t$ and $S_2 = S_w$ in the above optimization also produces the same set of discriminant coordinates $v_1, \ldots, v_{K-1}$ and find an expression for the corresponding objective function values in terms of $(v_i^T S_b v_i)/(v_i^T S_w v_i)$.

   (iii) Repeat part (ii) for the case $S_1 = S_b$ and $S_2 = S_t$.

   (c) [7 marks] Let V be the solution to the generalized eigenvector problem given by

$$S_b V = S_w V \Lambda$$

   where Λ is a diagonal matrix of eigenvalues, such that $\Lambda_{j,j} = \lambda_j$ and $\Lambda_{j,k} = 0$ if $j \neq k$. Prove that each discriminant coordinate v_i from part (b) is a column in V. You may assume that S_w is full rank.

   [*Hint: You may wish to consider reparameterising the problem using* $u = S_w^{1/2} w$ *where you should provide an explicit a definition for* $S_w^{1/2}$.]

SB2.2 2020

Q1 (a) $\ell(C) = -\sum_{i=1}^{n} [y_i \log(\hat{y}_i, C) + (1-y_i) \log(1-\hat{y}_i, C)]$

$\hat{y}_{i,C} = \hat{p}(y_i = 1 \mid x_i, \hat{\theta}_C)$

Naive Bayes classifier maximizes:

$\ell = \sum_{i=1}^{n} (\log(\pi_{y_i}) + \log g_{y_i}(x_i))$

$\Rightarrow$ NBGauss minimizes $\ell(C) = -\sum_{i=1}^{n} \log\left(\frac{-x_i - \mu_{y_i}}{2\sigma^2}\right)$

NNQuad outperforms NNLin. because NNLin model is nested within NNQuad.

Note that `NBGauss` is equivalent to Linear Discriminant Analysis with a diagonal covariance matrix, `NNLin` is equivalent to logistic regression, and `NNQuad` is equivalent to logistic regression with linear and quadratic features. The equivalence between these models and a comparison of their performance were discussed in lectures and classes (discussing this equivalence is not required). `NNLin` is trained to maximize the conditional log-likelihood, which is the same as minimizing $\ell(C)$. `BGauss` (equivalent to a particular case of LDA) has a similar objective function, which is not quite the minimization of $\ell(C)$. So `NNLin` is better than `BGauss` on a given training set. `NNLin` is a special case (zero weights to quadratic features) of `NNQuad`, which also optimizes $\ell(C)$ directly. So `NNQuad` outperforms (in training) `NNLin`. It follows that `NNQuad` outperforms `BGauss`.

(ii) If we have x_i is

Both `NBGauss` and `NNLin` result in linear decision boundaries, but `NBGauss` is a generative model with stronger assumptions (inductive bias). There is no clear answer for a data set of size N. If the assumptions match the data generating process, `NBGauss` could converge more quickly to a lower test error than `NNLin`, but that is not guaranteed. `NNLin` has lower bias, and when given enough training data ($n \to \infty$), it should be able to match or outperform `NBGauss`.

(b) (i)

$$R(\beta_{L2}) = \| y - X\beta_{L2} \|_2^2 + \lambda \| \beta_{L2} \|_2^2$$

$$= (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta$$

$$= -\beta^T X^T y + \beta^T X^T X \beta + y^T y - y^T X \beta + \lambda \beta^T \beta$$

$$= \beta^T X^T X \beta - 2 (X^T y)^T \beta + \lambda \beta^T \beta$$

$$\partial R / \partial \beta = 2 X^T X \beta - 2 X^T y + 2\lambda \beta$$

$$\partial R / \partial \beta = 0 \Rightarrow \hat{\beta}_{L2} = (X^T X + \lambda I)^{-1} X^T y$$

(ii) We have:

$$X^T X \beta - X^T y + \lambda \beta = 0$$

$$X X^T X \beta - X X^T y + \lambda X \beta = 0$$

$$(X X^T + \lambda I) X \beta = X X^T y$$

$AB^T = BA^T$

therefore.

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

$(XX^T)^T = XX^T$

$$(X^T X + \lambda I) X^T = X^T X X^T + \lambda X^T$$

$$= X^T (X X^T + \lambda I)$$

$$(X^T X + \lambda I)^{-1} (X^T X + \lambda I) X^T = (X^T X + \lambda I)^{-1} X^T (X X^T + \lambda I)$$

$$X^T = (X^T X + \lambda I)^{-1} X^T (X X^T + \lambda I)$$

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

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

$X^T X$: $p \times p$

$p \begin{pmatrix} \\ \\ \\ \end{pmatrix} \begin{pmatrix} \\ \\ \\ \end{pmatrix}$ $p \times 1$

$\hat{\beta}_{L2} = (X^T X + \lambda I)^{-1} X^T y$ $\quad X : n \times p$

$n \times n$

$\hat{\beta}'_{L2} = X^T (X X^T + \lambda I)^{-1} y$

$n \times n$

we have $p >> n$.

$X^T X$: $n p^2$

$\hat{\beta}_{L2}$: $+ \lambda I$: p^2

$(\,)^{-1}$: $O(p^3)$

$X^T y$: $p n$

$p \times p \quad \times \quad p \times 1$: p^2

$n p^2 + p^2 + O(p^3) + p n + p^2$

$\hat{\beta}'_{L2}$ = $p n^2 + n^2 + O(n^3) + n^2 + p n$.

$\hat{\beta}_{L2}$ easier

(N)

$$\hat{y}_{new} = y^T (\Phi\Phi^T + \lambda I)^{-1} \Phi \phi(x_{new})$$

$$= y^T (K + \lambda I)^{-1} k$$

K : $K_{ij} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

$$k(x_i, x_j) = \exp\left[-\frac{1}{2}(x_i - x_j)^2 \right] dx$$

We have $k(x_i, x_i) = 1 = \phi(x_i)^T \phi(x_i)$ $\quad \phi(x) \in \mathbb{R}^p$

expand about x_j :

$$k(x_i, x_j) = e^{-\frac{1}{2}(x_i - x_j)^2}$$

$$= \exp(-\tfrac{1}{2}x_i^2) \exp(-\tfrac{1}{2}x_j^2) \exp(x_i x_j) dx$$

$$= \exp(-\tfrac{1}{2}x_i^2) \exp(-\tfrac{1}{2}x_j^2) \sum_{n=0}^{\infty} \frac{x_i^n x_j^n}{n!}$$

$$\Rightarrow \phi(x_i)^T = \exp(-\tfrac{1}{2}x_i^2) \left[1, x_i, \frac{x_i^2}{\sqrt{2}}, \dots \right]$$

Q2.

(a)

(c) is sharpen. => (iii) ✓ emphasizes difference.

(b) is blur => (iv) ✓ local average.

(d) is edge detection. => (ii) ✓ emphasize vertical line, suppresses horizontal line.

(b)

Q3.

(a) PCA motivation: find orthogonal basis $v_1 \dots v_p$ s.t.

i) v_1 is the direction of greatest variance.

let $z_1 = a_1^T X$, $a_1 = (a_{11} \dots a_{1p})^T$

$v_1^* = \underset{a_1}{\operatorname{argmax}} \operatorname{Var}(z_1)$

$= \underset{a_1}{\operatorname{argmax}}\ a_1^T \Sigma a_1$ subject to $a_1^T a_1 = 1$

$\Rightarrow \Sigma a_1 = \lambda_1 a_1$, $\operatorname{Var}(z_1) = a_1^T \Sigma a_1$

$= \lambda_1$

$\Rightarrow$ v_1 is the eigenvector of greatest eigenvalue λ_1.

(ii) $\hat{\Sigma} = S = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T$

find eigenvectors of S.

(iii) To show: $S_t = S_W + S_B$

$$S_t = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(x_i-\bar{x})^T$$

$$= \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu_{y_i}+\mu_{y_i}-\bar{x})(x_i-\mu_{y_i}+\mu_{y_i}-\bar{x})^T$$

$$= \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu_{y_i})(x_i-\mu_{y_i})^T + \frac{1}{n-1}\sum_{i=1}^{n}(\mu_{y_i}-\bar{x})(\mu_{y_i}-\bar{x})^T$$

$$+ \underbrace{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu_{y_i})(\mu_{y_i}-\bar{x})^T + (\mu_{y_i}-\bar{x})(x_i-\mu_{y_i})^T}_{0}$$

$$= \sum_{i=1}^{n}(x_i-\mu_{y_i})(x_i-\mu_{y_i})^T + \sum_{i=1}^{n}(\mu_{y_i}-\bar{x})(\mu_{y_i}-\bar{x})^T$$

(b)

(i) $v_i = \arg\max_w \frac{w^T S_1 w}{w^T S_2 w}$ $\quad S_1 = S_b$, $S_2 = S_w$

subject to $w^T S_w v_j = 0$

Since. $w^T S_1 w$: Between class variance of $w^T x$

$w^T S_2 w$: within class variance of $w^T x$

$\Rightarrow \frac{w^T S_1 w}{w^T S_2 w}$ = separability of $w^T x$

we wish to find w that maximizes separability.

(ii) $S_1 = S_t$, $S_2 = S_w$

$$\frac{w^T S_t w}{w^T S_2 w} = \frac{w^T (S_b + S_w) w}{w^T S_w w}$$

$$= \frac{w^T S_b w + w^T S_w w}{w^T S_w w}$$

$$= 1 + \frac{w^T S_b w}{w^T S_w w}$$

(iii) $S_1 = S_b$, $S_2 = S_t$.

$$\frac{w^T S_b w}{w^T S_t w} = \frac{w^T S_b w}{w^T S_b w + w^T S_w w}$$

$$= \left(1 + \frac{w^T S_w w}{w^T S_b w}\right)^{-1}$$

maximize. (*) $\Rightarrow$ minimize. $\frac{w^T S_w w}{w^T S_b w}$.

$\Rightarrow$ Maximize. $\frac{w^T S_b w}{w^T S_w w}$.

(iii) Repeat part (ii) for the case $S_1 = S_b$ and $S_2 = S_t$.

Answer: **(N)** 4 marks.

In a similar manner to the last part, we start by noting that

$$\begin{aligned}\frac{w^T S_b w}{w^T S_t w} &= \frac{w^T S_b w}{w^T S_b w + w^T S_w w} \\ &= \frac{(w^T S_w w)^{-1} w^T S_b w}{(w^T S_w w)^{-1} w^T S_b w + 1}.\end{aligned}$$

Now if $f(w) > -1, \forall w$, then

$$\arg\max_w \frac{f(w)}{1 + f(w)} = \arg\max_w f(w)$$

such we see that the optimum is again unchanged. We further have that

$$(v_i^T S_1 v_i)/(v_i^T S_2 v_i) = \frac{\lambda_i}{1 + \lambda_i}$$

c) $S_b V = S_w V \Lambda$

$\Lambda_{jj} = \lambda_j$

$$u = S_w^{1/2} w.$$

$$u^T u = w^T S_w w.$$

$$v_i = \arg\max \frac{w^T S_1 w}{w^T S_2 w}.$$

$$S_b V = S_w V \Lambda$$

Part B

Course title: SB2b - Statistical Machine Learning
Lecturer: Pier Francesco Palamara
Checked by: Yee-Whye Teh

Date of this version: 20/03/2020

Do not turn this page until you are told that you may do so

1. (a) [8 marks] Consider the three binary classifiers described below, which are trained with a data set $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^p = [x_{i,1}, \ldots, x_{i,p}]^\top$ and $y_i \in \{0, 1\}$. After training a classifier C and learning the optimal parameter values $\widehat{\theta}_C$, we can obtain an estimate of the probability for the class label of a point x, which is given by

$$P(y = k|x, \widehat{\theta}_C), \quad k \in \{0, 1\}.$$

1. `NBGauss`, a Naive Bayes classifier assuming Gaussian features $x_i|y_i \sim N(\mu_{y_i}, \sigma^2)$. Note that each class may have a different mean, but the variance is the same for both classes. `NBGauss` learns its parameters using maximum likelihood.
2. `NNLin`, a neural network composed of a single neuron with logistic activation function $s(a) = 1/[1 + \exp(-a)]$. The network takes vectors $x_i = [x_{i,1}, \ldots, x_{i,p}, 1]^\top$ as an input and is trained by minimizing the log-loss

$$L(\widehat{y}_i, y_i) = -\sum_{i=1}^{n} \left[y_i \log(\widehat{y}_i) + (1 - y_i) \log(1 - \widehat{y}_i) \right].$$

   where $\widehat{y}_i = P(y_i = 1|x_i, \widehat{\theta})$.
3. `NNQuad`, which is equivalent to `NNLin` but takes $x_i = [x_{i,1}, \ldots, x_{i,p}, x_{i,1}^2, \ldots, x_{i,p}^2, 1]^\top$ as an input, i.e. it takes all elements of the data vector, in addition to each element squared, and a bias term.

(i) After training each classifier $C \in$ (`NBGauss`, `NNLin`, `NNQuad`), we may decide to measure its performance on the *training* data set using

$$\ell(C) = -\sum_{i=1}^{n} \left[y_i \log(\widehat{y}_{i,C}) + (1 - y_i) \log(1 - \widehat{y}_{i,C}) \right].$$

where $\widehat{y}_{i,C} = P(y_i = 1|x_i, \widehat{\theta}_C)$. Let us assume that a model C_1 outperforms (in training) model C_2 if it is true that $\ell(C_1) \leq \ell(C_2)$, for any possible training set. For each pair of models in (`NBGauss`, `NNLin`, `NNQuad`), state whether one model outperforms the other, and briefly explain why. You may presume that all numerical optimization methods will converge to a global optimum.

Answer: **(S)** 4 marks.

Note that `NBGauss` is equivalent to Linear Discriminant Analysis with a diagonal covariance matrix, `NNLin` is equivalent to logistic regression, and `NNQuad` is equivalent to logistic regression with linear and quadratic features. The equivalence between these models and a comparison of their performance were discussed in lectures and classes (discussing this equivalence is not required). `NNLin` is trained to maximize the conditional log-likelihood, which is the same as minimizing $\ell(C)$. `BGauss` (equivalent to a particular case of LDA) has a similar objective function, which is not quite the minimization of $\ell(C)$. So `NNLin` is better than `BGauss` on a given training set. `NNLin` is a special case (zero weights to quadratic features) of `NNQuad`, which also optimizes $\ell(C)$ directly. So `NNQuad` outperforms (in training) `NNLin`. It follows that `NNQuad` outperforms `BGauss`.

(ii) Under which circumstances (e.g. linked to data generating process and training data set size) do you expect `NBGauss` to have a lower *test* error than `NNLin`, and vice versa?

Answer: **(S)** 4 marks.

Both `NBGauss` and `NNLin` result in linear decision boundaries, but `NBGauss` is a generative model with stronger assumptions (inductive bias). There is no clear

answer for a data set of size N. If the assumptions match the data generating process, `NBGauss` could converge more quickly to a lower test error than `NNLin`, but that is not guaranteed. `NNLin` has lower bias, and when given enough training data ($n \to \infty$), it should be able to match or outperform `NBGauss`.

(b) [17 marks] Consider an L2-regularized regression model with squared loss, data matrix $X \in \mathbb{R}^{n \times p}$, and responses $y \in \mathbb{R}^n$. The empirical risk minimization problem (ignoring constants and the intercept) is given by:

$$R(\beta_{L2}) = \|y - X\beta_{L2}\|_2^2 + \lambda\|\beta_{L2}\|_2^2$$

(i) Derive an optimal solution $\widehat{\beta}_{L2}$ which uses the matrix $X^\top X \in \mathbb{R}^{p \times p}$ (there is no need to check that this is a globally optimal solution).
Answer: **(B)** 3 marks.

$$\frac{\partial \widehat{R}(\beta_{L2})}{\partial \beta_{L2}} = 2(X^\top X\beta_{L2} - X^\top y) + 2\lambda\beta_{L2}$$
$$2(X^\top X\beta_{L2} - X^\top y) + 2\lambda\beta_{L2} = 0 \to \widehat{\beta}_{L2} = (X^\top X + \lambda I_{p\times p})^{-1}X^\top y$$

(ii) Show that the optimal solution may be written as $\widehat{\beta}'_{L2} = X^\top(XX^\top + \lambda I)^{-1}y$, where I is the $n \times n$ identity matrix.
Answer: **(N)** 5 marks.

$$\begin{aligned}
(X^\top X + \lambda I)X^\top &= X^\top XX^\top + \lambda X^\top \\
&= X^\top(XX^\top + \lambda I) \\
(X^\top X + \lambda I)^{-1}(X^\top X + \lambda I)X^\top &= (X^\top X + \lambda I)^{-1}X^\top(XX^\top + \lambda I) \\
X^\top &= (X^\top X + \lambda I)^{-1}X^\top(XX^\top + \lambda I) \\
X^\top(XX^\top + \lambda I)^{-1}y &= (X^\top X + \lambda I)^{-1}X^\top(XX^\top + \lambda I)(XX^\top + \lambda I)^{-1}y \\
X^\top(XX^\top + \lambda I)^{-1}y &= (X^\top X + \lambda I)^{-1}X^\top y.
\end{aligned}$$

(iii) Computing $\widehat{\beta}_{L2}$ and $\widehat{\beta}'_{L2}$ will require performing several operations. For instance, computing $M^\top v$ where $M \in \mathbb{R}^{n\times p}$ and $v \in \mathbb{R}^n$ requires $n \times p$ operations. Provide an estimate of the cost of computing $\widehat{\beta}_{L2}$ and $\widehat{\beta}'_{L2}$ as a function of n and p. Assume that both n and p are large, but p is substantially larger than n. Would it be computationally cheaper to compute $\widehat{\beta}_{L2}$ or $\widehat{\beta}'_{L2}$? Justify your answer.
Answer: **(S)** 4 marks.
For $\widehat{\beta}_{L2} = (X^\top X + \lambda I)^{-1}X^\top y$
- $X^\top X$ will cost $p^2 n$
- $+\lambda I$ will cost p^2
- matrix inversion will cost $O(p^3)$ (or $O(p^{2.37})$ as shown in lectures)
- multiplying by $X^\top y$ will cost pn
- final multiplication for $p \times p$ matrix and $p \times 1$ will cost p^2

Therefore $p^2 n + p^2 + O(p^3) + pn + p^2$.
For $\widehat{\beta}'_{L2} = X^\top(XX^\top + \lambda I)^{-1}y$.
- $XX^\top$ will cost pn^2
- $+\lambda I$ will cost n^2
- matrix inversion will cost $O(n^3)$ (or $O(n^{2.37})$)
- multiplying by y will cost n^2
- final multiplication for $p \times n$ matrix and $n \times 1$ will cost pn

Therefore $pn^2 + n^2 + O(n^3) + n^2 + pn$.
It is more convenient to compute $\widehat{\beta}'_{L2}$ if $p \gg n$.

(iv) Assume we use a feature expansion $\phi : x_i \in \mathbb{R}^p \to \phi(x_i) \in \mathbb{R}^d$ and let $\Phi \in \mathbb{R}^{n \times d}$ be a matrix that has $\phi(x_i)^\top$ is in its $i-$th row. Given a new sample x_{new} we can construct the prediction $\widehat{y}_{new}$ as follows:

$$\begin{aligned}\widehat{y}_{new} &= y^\top (\Phi\Phi^\top + \lambda I)^{-1} \Phi \phi(x_{new}) \\ &= y^\top (K + \lambda I)^{-1} \kappa,\end{aligned}$$

where the $\{i, j\}$-th entry of the matrix $K \in \mathbb{R}^{n \times n}$ is given by $k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$ and the i-th element of the vector $\kappa \in \mathbb{R}^n$ is given by $k(x_i, x_{new}) = \phi(x_i)^\top \phi(x_{new})$. The function $k(\cdot, \cdot)$ is known as a *kernel* and is used to compute the similarity of two data points after applying the feature expansion. Assume that $p = 1$, so that $x_i \in \mathbb{R}$. We decide to use the Gaussian kernel

$$k(x_i, x_j) = \phi(x_i)^\top \phi(x_j) = \exp\left\{-\frac{1}{2}(x_i - x_j)^2\right\}.$$

Show that in this case we can write the vectors $\phi(x_i)$ and $\phi(x_j)$ as infinite-dimensional. Provide an explicit expression for the entries of $\phi(x_i)$. *[Hint: You may want to use the Taylor expansion of a real-valued function $f(x)$ about a point a, which is given by $f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$]*

Answer: **(N)** 5 marks.
We have

$$\begin{aligned}\phi(x_i)^\top \phi(x_j) &= \exp\left\{-\frac{1}{2}(x_i - x_j)^2\right\} \\ &= \exp\left(-\frac{1}{2}x_i^2\right) \exp\left(-\frac{1}{2}x_j^2\right) \exp\left(x_i x_j\right) \\ &= \exp\left(-\frac{1}{2}x_i^2\right) \exp\left(-\frac{1}{2}x_j^2\right) \sum_{n=0}^{\infty} \frac{x_i^n x_j^n}{n!},\end{aligned}$$

from which we can deduce

$$\phi(x_i) = \exp\left(-\frac{1}{2}x_i^2\right) \left[1, x_i, \frac{x_i^2}{\sqrt{2!}}, \frac{x_i^3}{\sqrt{3!}}, \frac{x_i^4}{\sqrt{4!}}, \dots\right]^\top.$$

2. (a) [5 marks] Consider the following image and three different convolutions of it:

(a) Original image

(b) Convolved image 1 (blurrier)

(c) Convolved image 2 (sharper)

(d) Convolved image 3

Each convolved image was generated by convolving with one of the following kernels (note one of the kernels is not used)

$$\text{(i)}: \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \text{(ii)}: \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

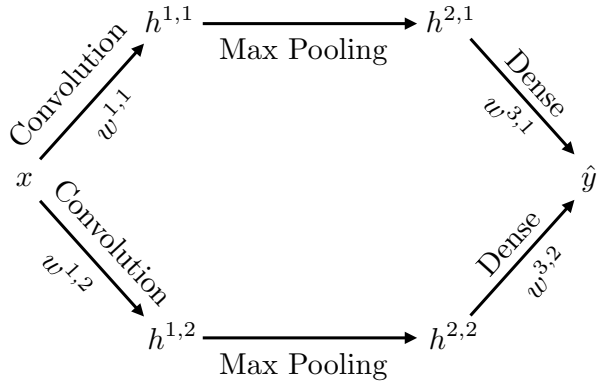
$$\text{(iii)}: \begin{bmatrix} -0.5 & -1 & -0.5 \\ -1 & 7 & -1 \\ -0.5 & -1 & -0.5 \end{bmatrix} \qquad \text{(iv)}: \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

For each convolved image, identify which kernel was used to produce it. Explain your reasoning.

Answer: (S) 5 marks.

- Image 1 is kernel (iv). This is because this kernel will have a smoothing effect (it takes a local average) and thus blurs the image.
- Image 2 is kernel (iii). This kernel has the effect of sharpening effect as it emphasizes differences between a pixel and its neighbors.
- Image 3 is kernel (ii). This kernel has an embossing effect that emphasizes vertical lines and suppresses horizontal lines. You can tell that it is not (i) (which emphasizes horizontal lines) because of which aspects of the image are more noticeable.

(b) [15 marks] Let $(x_1, y_1), \ldots, (x_n, y_n)$, where each $x_i \in \mathbb{R}^{p \times 1}$ and $y_i \in \{0, 1\}\ \forall i \in \{1, \ldots, n\}$, represent a set of input and output pairs constituting our training data with p-dimensional vector inputs and categorical outputs. Consider the following convolutional neural network (CNN) architecture:



such that the convolution outputs are given by

$$h^{1,1} = s_1(w^{1,1} \circledast x)$$
$$h^{1,2} = s_1(w^{1,2} \circledast x)$$

where $s_1(\cdot)$ is an activation function, $\circledast$ represents a convolution, and $w^{1,m} \in \mathbb{R}^{(2\tau+1)\times 1}$ for some integer τ. The max pooling outputs are given by

$$h_j^{2,1} = \max\left(h_{2j-1}^{1,1}, h_{2j}^{1,1}\right), \ \forall j \in \{1, \ldots, d\}$$
$$h_j^{2,2} = \max\left(h_{2j-1}^{1,2}, h_{2j}^{1,2}\right), \ \forall j \in \{1, \ldots, d\}$$

where d is half the number of hidden units in $h^{1,1}$ (and thus also $h^{1,2}$), and you can assume that d is an integer. The network output is given by

$$\widehat{y} = s_3\left(\sum_{j=1}^{d} w_j^{3,1} h_j^{2,1} + w_j^{3,2} h_j^{2,2}\right),$$

where $s_1(\cdot)$ is an activation function. Note that bias and regularization terms are deliberately omitted for simplicity.

(i) Explain why a mix of convolutional and max pooling layers in a CNN can form an effective architecture for dealing with image data.

Answer: **(B)** 3 marks.

Credit will generally be given for sensible answers: the key is to explain the computational benefits and why they help us to pick out features in images. Important points the students might make include:

- Convolutions massively reduce the number of parameters stored in memory and the number of computations required, allowing us to have deeper networks with more hidden units; deeper layers are typically able to detect more complex features.
- Max pooling can reduce the number of parameters and help protect against overfitting
- A combination of convolutions and max pooling can naturally introduce spatial (and potentially rotational) invariances such that is does not matter where features are in an image
- Extensive empirical justification for such architectures exists in the literature.

(ii) Apply the chain rule to write down a set of recursive expressions for backpropagating gradients of the empirical risk

$$R_{\text{emp}} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \widehat{y}_i)$$

through the network. Note that in this part of the question you only need to express each gradient as a function of other gradients.

Answer: (B/S) 5 marks.

There are no parameters for the pooling layer and so the gradients we need to calculate are $\partial R_{\text{emp}}/\partial w_j^{3,m}$ and $\partial R_{\text{emp}}/\partial w_\ell^{1,m}$, for which we have

$$\frac{\partial R_{\text{emp}}}{\partial w_j^{3,m}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial R_{\text{emp}}}{\partial \widehat{y}_i}\frac{\partial \widehat{y}_i}{\partial w_j^{3,m}} \quad \forall m \in \{1,2\}, j \in \{1,\ldots,d\}$$

$$\frac{\partial R_{\text{emp}}}{\partial w_\ell^{1,m}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{s=1}^{2d} \frac{\partial R_{\text{emp}}}{\partial h_{i,s}^{1,m}}\frac{\partial h_{i,s}^{1,m}}{\partial w_\ell^{1,m}} \quad \forall m \in \{1,2\}, \ell \in \{1,\ldots,2\tau+1\}$$

To complete the set of backpropagation rules we also need the gradients of the hidden units, for which we have

$$\frac{\partial R_{\text{emp}}}{\partial h_{i,j}^{2,m}} = \frac{\partial R_{\text{emp}}}{\partial \widehat{y}_i}\frac{\partial \widehat{y}_i}{\partial h_{i,j}^{2,m}} \quad \forall m \in \{1,2\}, j \in \{1,\ldots,d\}, i \in \{1,\ldots,n\}$$

$$\frac{\partial R_{\text{emp}}}{\partial h_{i,s}^{1,m}} = \sum_{j=1}^{d} \frac{\partial R_{\text{emp}}}{\partial h_{i,j}^{2,m}}\frac{\partial h_{i,j}^{2,m}}{\partial h_{i,s}^{1,m}} \quad \forall m \in \{1,2\}, s \in \{1,\ldots,2d\}, i \in \{1,\ldots,n\}$$

Note that using a valid indexing approach is very important here: marks will be deducted for errors or not indexing properly.

(iii) Presume that we use a log-loss function, a ReLu activation function for s_1, and a logistic activation function for s_3, namely that we take

$$\begin{aligned} L(y_i, \widehat{y}_i) &= -\left[y_i \log(\widehat{y}_i) + (1-y_i)\log(1-\widehat{y}_i)\right] \\ s_1(a) &= \max(a, 0) \\ s_3(a) &= \frac{1}{1+\exp(-a)}. \end{aligned}$$

Derive explicit expressions for all the partial derivatives that are required to complete the backpropagation algorithm.

Answer: (S/N) 7 marks.

For this part of the question we effectively just need to calculate each of the component terms from the previous part. Note that each of these can be calculated in isolation and we can make use of the following identities (each of which can be derived straightforwardly from scratch)

$$\frac{\partial \max(a,b)}{\partial a} = \mathbb{I}(a > b) \quad \Rightarrow \quad \frac{\partial s_1(a)}{\partial a} = \mathbb{I}(a > 0)$$

$$\frac{\partial s_3(a)}{\partial a} = s(a)(1-s(a)).$$

Using the above we can now derive expressions for all the gradients as follows, where we have used a to denote inputs to their corresponding hidden unit activation

function

$$\frac{\partial R_{\text{emp}}}{\partial \widehat{y}_i} = -\frac{y_i}{\widehat{y}_i} + \frac{1-y_i}{1-\widehat{y}_i}$$

$$\frac{\partial \widehat{y}_i}{\partial w_j^{3,m}} = \frac{\partial \widehat{y}_i}{\partial a_i^3}\frac{\partial a_i^3}{\partial w_j^{3,m}} = \widehat{y}_i\left(1-\widehat{y}_i\right)h_{i,j}^{2,m}$$

$$\frac{\partial \widehat{y}_i}{\partial h_{i,j}^{2,m}} = \frac{\partial \widehat{y}_i}{\partial a_i^3}\frac{\partial a_i^3}{\partial h_{i,j}^{2,m}} = \widehat{y}_i\left(1-\widehat{y}_i\right)w_j^{3,m}$$

$$\frac{\partial h_{i,j}^{2,m}}{\partial h_{i,s}^{1,m}} = \frac{\partial \max(h_{i,2j-1}^{1,m}, h_{i,2j}^{1,m})}{\partial h_{i,s}^{1,m}}$$

$$= \mathbb{I}(s=2j-1)\mathbb{I}\left(h_{i,2j-1}^{2,m} > h_{i,2j}^{2,m}\right) + \mathbb{I}(s=2j)\mathbb{I}\left(h_{i,2j}^{2,m} > h_{i,2j-1}^{2,m}\right)$$

$$\frac{\partial h_{i,s}^{1,m}}{\partial w_\ell^{1,m}} = \frac{\partial h_{i,s}^{1,m}}{\partial a_{i,s}^{1,m}}\frac{\partial a_{i,s}^{1,m}}{\partial w_\ell^{1,m}} = \mathbb{I}\left(\sum_{\ell=1}^{2\tau+1} w_\ell^{1,m} x_{i,(s-\tau+\ell-1)} > 0\right) x_{i,(s-\tau+\ell-1)}.$$

Note that it is not necessary to substitute these equations back into to our answer from (i) to give complete representations for the gradients of the parameters: by construction, backpropagation is done recursively so we actively avoid this calculation.

(c) [5 marks] Consider the risk of this CNN classifier when we train using stochastic gradient descent with random initialization of the weights, namely

$$R = \mathbb{E}\left[L\left(Y, \widehat{Y}(X)\right)\right]$$

where $L(\cdot,\cdot)$ is the log-loss, $\widehat{Y}(X)$ represents the output of the trained network given input X and this expectation is over all randomness in the process (including data generation). Consider further training an ensemble of T CNNs by independently training each with the same dataset but a different initialization—each of which is independently drawn from the same initialization distribution—and then using the empirical average of the predictions for each network as the overall prediction of the ensemble. Prove that the risk of this ensemble CNN,

$$R_e = \mathbb{E}\left[L\left(Y, \widehat{Y}_e(X)\right)\right],$$

is always less than or equal to R. *[Hint: this result is not specific to the architecture but relies on a specific property of the loss function.]*

Answer: (N) 5 marks.

The result can be proven by carefully defining the ensemble prediction, namely,

$$\widehat{y}_e = \frac{1}{T}\sum_{t=1}^{T}\widehat{y}_t$$

and then applying Jensen's inequality.

To prove the desired result, let us start by inserting the definition of $\widehat{y}_e$ into the risk, along with the loss function itself

$$R_e = \mathbb{E}\left[L\left(Y, \frac{1}{T}\sum_{t=1}^{T}\widehat{Y}_t(X)\right)\right]$$

$$= \mathbb{E}\left[-\sum_{k=1}^{K}\mathbb{I}(Y=k)\log\left(\frac{1}{T}\sum_{t=1}^{T}\widehat{Y}_t(X)\right)\right]$$

which, leveraging convexity and Jensen's inequality, gives

$$\begin{aligned}
&\leq \mathbb{E}\left[-\sum_{k=1}^{K}\mathbb{I}(Y=k)\frac{1}{T}\sum_{t=1}^{T}\log\left(\widehat{Y}_t(X)\right)\right] \\
&= \mathbb{E}\left[-\sum_{k=1}^{K}\mathbb{I}(Y=k)\log\left(\widehat{Y}_1(X)\right)\right] \\
&= R.
\end{aligned}$$

Note that the above is general so that $\widehat{Y}_t(X)$ denotes a K-long vector representing the estimated probability for the class labels. Entries of these vector are non-negative and sum to one. Here we have a binary classifier, so $K = 2$.

3. (a) [9 marks] Let $(x_1, y_1), \ldots, (x_n, y_n)$, where each $x_i \in \mathbb{R}^{p\times 1}$ and $y_i \in \{1, \ldots, K\}$ $\forall i \in \{1, \ldots, n\}$, represent a set of input and output pairs constituting our training data with p-dimensional vector inputs and categorical outputs. Presume that $K < p$ and let $n_k = \sum_{i=1}^{n} \mathbb{I}(y_i = k)$ denote the number of examples of each class and define the following terms

Data mean	$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
Class means	$\mu_k = \frac{1}{n_k}\sum_{i:y_i=k} x_i = \frac{1}{n_k}\sum_{i=1}^{n} x_i \mathbb{I}(y_i = k)$
Total covariance	$S_t = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T$
Within class covariance	$S_w = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu_{y_i})(x_i - \mu_{y_i})^T$
Between class covariance	$S_b = \frac{1}{n-1}\sum_{k=1}^{K} n_k (\mu_k - \overline{x})(\mu_k - \overline{x})^T$

(i) Explain the motivation behind principal component analysis (PCA) and provide a formulation for the principal component directions as a solution to a constrained optimization problem in terms of a random input vector X.
Answer: (B) 3 marks.
PCA aims to find an orthogonal basis of the data space which maximizes the variances. It is often used as a linear dimensionality reduction technique.
The PCA directions are defined to solve the optimization

$$v_i = \arg\max_{w} \text{Var}(w^T X) \quad \text{subject to } v_i^T v_i = 1 \text{ and } v_i^T v_j = 0 \ \forall j \in \{1, \ldots, i-1\}$$

(ii) Given input data $x_1, \ldots, x_n$ such that we approximate $\text{Cov}(X)$ with S_t, explain (without proof) how we can recover the principal components from S_t.
Answer: (B) 1 mark.
It is sufficient to state that they are given by the eigenvectors of S_t in decreasing order of their eigenvalue.

(iii) Show that $S_t = S_w + S_b$.
Answer: (B) 5 marks.

$$\begin{aligned}
S_t =& \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T \\
=& \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \overline{x} + \mu_{y_i} - \mu_{y_i})(x_i - \overline{x} + \mu_{y_i} - \mu_{y_i})^T \\
=& \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu_{y_i})(x_i - \mu_{y_i})^T + \frac{1}{n-1}\sum_{i=1}^{n} (\mu_{y_i} - \overline{x})(\mu_{y_i} - \overline{x})^T \\
&+ \underbrace{\frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu_{y_i})(\mu_{y_i} - \overline{x})^T + (\mu_{y_i} - \overline{x})(x_i - \mu_{y_i})^T}_{0}
\end{aligned}$$

$$
\begin{aligned}
&= S_w + \frac{1}{n-1}\sum_{i=1}^{n} \mathbb{I}(k = y_i)\,(\mu_k - \overline{x})\,(\mu_k - \overline{x})^T \\
&= S_w + \frac{1}{n-1}\sum_{k=1}^{K} n_k\,(\mu_k - \overline{x})\,(\mu_k - \overline{x})^T \\
&= S_w + S_b
\end{aligned}
$$

(b) [9 marks] The discriminant coordinates $v_1, \ldots, v_{K-1}$ of Fisher's discriminant analysis (FDA) are defined to solve the following series of optimization problems

$$
v_i = \arg\max_w \frac{w^T S_1 w}{w^T S_2 w} \quad \text{subject to} \quad w^T S_w v_j = 0,\ \forall j \in \{1, \ldots, i-1\}
$$

where $S_1 = S_b$ and $S_2 = S_w$.

(i) Explain the principles behind the objective in this formulation, taking care to discuss what each of the numerator and the denominator represent.
Answer: **(B)** 2 marks.
The motivation is that this represents a signal-to-noise ratio for the class labeling: we want to maximize the separation between the class centroids while minimizing the within class variances, after projecting on w.

(ii) Show that instead setting $S_1 = S_t$ and $S_2 = S_w$ in the above optimization also produces the same set of discriminant coordinates $v_1, \ldots, v_{K-1}$ and find an expression for the corresponding objective function values in terms of $(v_i^T S_b v_i)/(v_i^T S_w v_i)$.
Answer: **(N)** 3 marks.
Noting from earlier that $S_t = S_w + S_b$, we can prove this by simply noting that

$$
\frac{w^T S_t w}{w^T S_w w} = 1 + \frac{w^T S_b w}{w^T S_w w}
$$

such that the optimum is the same. We can further use this to see that

$$
(v_i^T S_t v_i)/(v_i^T S_w v_i) = (v_i^T S_b v_i)/(v_i^T S_w v_i) + 1.
$$

(iii) Repeat part (ii) for the case $S_1 = S_b$ and $S_2 = S_t$.
Answer: **(N)** 4 marks.
In a similar manner to the last part, we start by noting that

$$
\begin{aligned}
\frac{w^T S_b w}{w^T S_t w} &= \frac{w^T S_b w}{w^T S_b w + w^T S_w w} \\
&= \frac{(w^T S_w w)^{-1} w^T S_b w}{(w^T S_w w)^{-1} w^T S_b w + 1}.
\end{aligned}
$$

Now if $f(w) > -1, \forall w$, then

$$
\arg\max_w \frac{f(w)}{1 + f(w)} = \arg\max_w f(w)
$$

such we see that the optimum is again unchanged. We further have that

$$
(v_i^T S_1 v_i)/(v_i^T S_2 v_i) = \frac{\lambda_i}{1 + \lambda_i} \tag{1}
$$

(c) [7 marks] Let V be the solution to the generalized eigenvector problem given by

$$S_b V = S_w V \Lambda$$

where Λ is a diagonal matrix of eigenvalues, such that $\Lambda_{j,j} = \lambda_j$ and $\Lambda_{j,k} = 0$ if $j \neq k$. Prove that each discriminant coordinate v_i from part (b) is a column in V. You may assume that S_w is full rank. *[Hint: You may wish to consider reparameterising the problem using $u = S_w^{1/2} w$ where you should provide an explicit a definition for $S_w^{1/2}$.]*

Answer: **(S/N)** 7 marks. Starting with the eigendecomposition of S_w, we have

$$S_w = U_w \Lambda_w U_w^T$$

and can thus define

$$S_w^{1/2} \triangleq U_w \Lambda_w^{1/2} U_w^T$$

where $\Lambda_w^{1/2}$ is the element-wise square root of Λ_w.

We can thus define $u = S_w^{1/2} w$ and impose, without loss of generality, the restriction that $u^T u = 1$ through simple scaling of w. Noting that $S_w^{1/2}$ is invertible and $\left(S_w^{-1/2}\right)^T = S_w^{-1/2}$, we can thus express our optimization as

$$\begin{aligned} z_i = \arg\max_u \left(u^T S_w^{-1/2} S_b S_w^{-1/2} u \right) \\ \text{subject to} \quad u^T u = 1, \\ u^T z_j = 0, \forall j \in \{1, \ldots, i-1\} \end{aligned}$$

where $v_j = S_w^{-1/2} z_j$

Exploiting equivalence to the earlier PCA result, the solutions to this are the eigenvectors of $S_w^{-1/2} S_b S_w^{-1/2}$ ordered in decreasing size of their eigenvalue. Note further that these eigenvectors are orthogonal because the matrix is symmetric, which in turn ensures the constraints are satisfied.

We thus now have

$$\begin{aligned} S_w^{-1/2} S_b S_w^{-1/2} z_i &= \lambda_i z_i \\ S_w^{-1/2} S_b v_i &= \lambda_i S_w^{1/2} v_i \\ S_b v_i &= \lambda_i S_w v_i \\ S_b V &= S_w V \Lambda \end{aligned}$$

such that the v_i form the columns of V.