SB2.2 Statistical Machine Learning.

Revision.

# May 7 10:30

07 May 2021 10:21

- MSc Exam 2020, Q3 a)ii, iii, b)ii, iii
- MSc Exam 2020, Q4 a)ii, iii
- Miscellaneous questions:
  - 1) In part 1 of the lecture notes (p.7), we are introduced to the linear encoder and corresponding linear decoder. I notice that we assume that if the encoder uses the matrix At, then the decoder would use A, but we haven't considered a different choice of linear decoder. We found in sheet 1 the optimal matrix A under this construction, but could it be that if we used a different matrix for the decoder we could get an even lower empirical risk? I guess my question is more about whether we restrict to the specific decoder because it is optimal, and we can prove it is optimal, or because it is easier than finding an optimal combination of encoder and decoder matrices?
  - 2) In part 2 of the lecture notes (p.41), I'm not quite sure why the approximation when going from (4.6) to (4.7) is appropriate (I am referring to approximating the Hessian of J as a multiple of the identity matrix)? Does this have something to do with normalising the data beforehand at all?
  - 3) In section 4.4 of lecture notes part 2, I see the graphs of Training / Validation / Test empirical risk, but I can't quite find in the paragraphs preceding the graphs where we use the test set, and so unsure of how the values in the graph are calculated? And I assume this is connected, but I'm unsure of why in that same graph the validation error increases so steeply compared to the test error after we pass the optimal iteration?
  - 4) On lecture notes, part II, page 9, we want to maximize u^{T} \dot{B} u/(u^{T}u), why it is achieved by the first eigenvector of \dot{B}?
  - 5) On page 55 of slides04_05 about plug-in estimators. Are these results only valid when we use square loss for regression and 0-1 loss for classification?
  - 6) We talked about some surrogate loss to replace 0-1 loss in lecture 11. I'm wondering if we use weighted loss rather than 0-1 loss, do we still have surrogate loss defined and how they will be like? Could you show an example like deriving the form of logistic surrogate loss if we use weighted loss in the ERM?

Q3. a) ii) $y_i \in \{-1, 1\}$

$$R(\beta) = \sum_{i=1}^{n} -\log s(y_i x_i^T \beta)$$

$$s(z) = \frac{1}{1+e^{-z}} \qquad \frac{ds(z)}{dz} = s(z)(1-s(z)) \qquad s(-z) = 1 - s(z)$$

$$\frac{d\, a^T\beta}{d\beta} = a$$

$$\frac{dR}{d\beta} = -\sum_{i=1}^{n} \frac{y_i x_i\, s(y_i x_i^T\beta)(1-s(y_i x_i^T\beta))}{s(y_i x_i^T\beta)}$$

$$= -\sum_{i=1}^{n} y_i x_i\, s(-y_i x_i^T \beta)$$

$$y_i x_i\, s(-y_i x_i^T\beta) = \begin{cases} x_i\, s(-x_i^T\beta) & \text{if } y_i = 1 \\ -x_i\, s(x_i^T\beta) & \text{if } y_i = -1 \end{cases}$$

$$= \begin{cases} x_i\,(1 - s(x_i^T\beta)) & \text{if } y_i = 1 \\ -x_i\, s(x_i^T\beta) & \text{if } y_i = -1 \end{cases}$$

$\mu_i = s(x_i^T\beta)$

$c_i = \mathbb{1}_{y_i=1}$

$$= x_i(\mathbb{1}_{y_i=1} - \mu_i)$$

$$= x_i(c_i - \mu_i)$$

$$\frac{dR}{d\beta\, d\beta^T} = -\frac{d}{d\beta^T}\left(\sum_{i=1}^{n} y_i x_i\, s(-y_i x_i^T\beta)\right) \qquad \frac{d\, a^T\beta}{d\beta^T} = a^T$$

$$= \sum_{i=1}^{n} y_i x_i \times y_i x_i^T \, s(y_i x_i^T \beta)\,(1 - s(y_i x_i^T \beta))$$

$$= \sum_{i=1}^{n} x_i x_i^T \, s(y_i x_i^T \beta)\,(1 - s(y_i x_i^T \beta))$$

$$= X^T S X$$

b) ii). $R(\beta_1, \beta_0) = \sum_{i=1}^{3} (y_i - \beta_0 - \beta_1 x_i)^2 + \lambda \beta_1^2$

$\lambda$ regularisation on the slope.

L1 regularisation

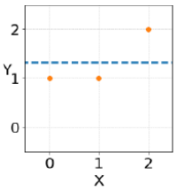



Figure 1

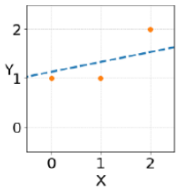



Figure 2

$\lambda = 3$

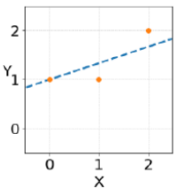



Figure 3

$\lambda = 1$

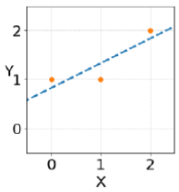



Figure 4

$\lambda = 0$

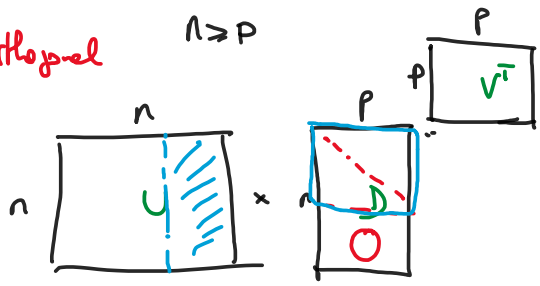
b) iii) $\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$ Ridge estimator.

$\|\hat{\beta}\|^2$

Full SVD: $X = U D V^T$ → p×p orthogonal

$n \times p$ $n \times n$ orthogonal $n \times p$ diagonal

$n \geq p$




Reduced SVD: $X = \tilde{U} \tilde{D} V^T$ ($n \geq p$)

$n \times p$ $p \times p$ diag

**Statistical Machine Learning**

4. (a) [10 marks] Consider performing regression on a training data set $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. You decide to use the 1-nearest neighbour (1NN) algorithm, which predicts the label of a new point to be the same as the label of its nearest training point.

   (i) How is the 1NN algorithm trained?

   (ii) Assume we are using the squared loss, which for a new point $(x_{new}, y_{new})$ and a prediction $\widehat{y}_{new} = f_{1NN}(x_{new})$ is given by

   $$L(y_{new}, \widehat{y}_{new}) = (y_{new} - \widehat{y}_{new})^2 .$$

   You would like to estimate how the 1NN algorithm performs on unseen data. Provide a brief description of the leave-one-out cross-validation (LOO-CV) algorithm and of how that could be used for this task. Describe the role of training, validation, and test sets when performing LOO-CV. If your goal is to obtain the most accurate estimate for the true risk of the 1NN algorithm using the available data, what size do you think the test partition should have?

   (iii) Assume that the data points are sampled according to the following process. The value of $x$ is sampled uniformly in the continuous interval $[0, 1]$. The value of $y \in \mathbb{R}$ is independently sampled from a normal distribution with mean 0 and variance $\sigma^2$. We train with $n$ points $\{x_i, y_i\}_{i=1}^n$ and we want to predict the label of a new point $x_{new}$. We use the squared loss and are interested in comparing two algorithms: 1NN, previously described, and "Always Zero" (A0), which always predicts $\widehat{y}_{new} = 0$. What is the true risk of the 1NN? What is the true risk of A0? Which algorithm would you prefer?

$J$ around a point $\theta \in \mathbb{R}^p$, we have, for $\delta \in \mathbb{R}^p$,

$$J(\theta+\delta) \simeq J(\theta) + \nabla_\theta J(\theta)^\intercal \delta + \frac{1}{2}\delta^\intercal \nabla^2_\theta J(\theta)\delta. \tag{4.6}$$

Replacing the Hessian matrix $\nabla^2_\theta J(\theta)$ with the diagonal $p$-by-$p$ matrix $\frac{1}{\eta}I$ gives the approximation

$$J(\theta+\delta) \simeq J(\theta) + \nabla_\theta J(\theta)^\intercal \delta + \frac{1}{2\eta}\|\delta\|^2. \tag{4.7}$$

At a given point $\theta$, we use the above quadratic function as an approximation to $J(\theta+\delta)$. Taking the derivative with respect to $\delta$, we obtain

$$\nabla_\theta J(\theta) + \frac{1}{\eta}\delta = 0,$$

hence the minimum is achieved for $\delta = -\eta\nabla_\theta J(\theta)$, which corresponds to the gradient descent update. See Figure 4.3 for an illustration. Instead of taking approximating the Hessian, we can use directly the second-order approximation (4.6). Differentiating with respect to $\delta$ and setting to 0, we obtain

$$\nabla_\theta J(\theta) + \nabla^2_\theta J(\theta)\delta = 0$$

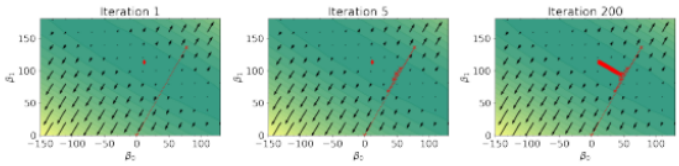



Figure 4.5: Contours of the objective function (unnormalised case), miminum of the objective function (red diamond), and gradient descent updates (crosses). The gradients are represented by arrows.

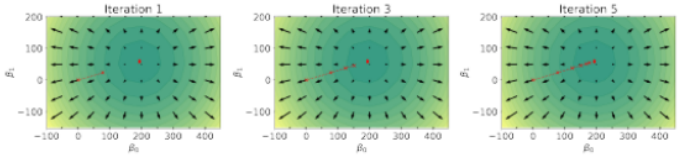



Figure 4.6: Contours of the objective function (normalised case), miminum of the objective function (red diamond), and gradient descent updates (crosses). The gradients are represented by black arrows.

## 4.3 Stochastic Gradient Descent

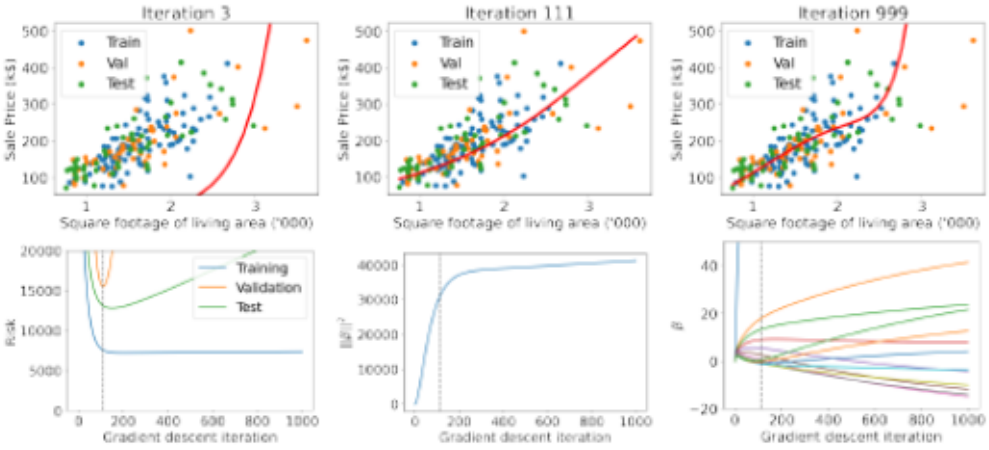


Figure 4.8: Illustration of early stopping and implicit regularisation in gradient descent. Top: Prediction rules at some iterations of the Gradient Descent algorithm. Bottom left: Training, empirical and test risk as a function of the gradient descent iteration. The dotted line corresponds to the iteration where the validation risk is minimised. Bottom middle: Squared norm of beta as a function of the GD iterations. Bottom right: Values of the different dimensions of the parameter $\beta$ as a function of the number of GD iterations.

$$\mathbb{E}[\mathrm{var}(a^\intercal X \mid Y)] \qquad a^\intercal \Sigma a$$

Setting $u = \Sigma^{1/2} a$ yields

$$\frac{u^\intercal B^\bullet u}{u^\intercal u}.$$

Maximisation over $u$ is achieved by the first eigenvector of $B^\bullet$, which corresponds to $u_1^\bullet$ as above. The next eigenvector $u_2^\bullet$ is obtained by finding the vector orthogonal to $u_1^\bullet$ that maximises $\frac{u_2^\intercal B^\bullet u_2}{u_2^\intercal u_2}$, etc.

## Plug-in

- Assume that the conditional probability mass/density function of $Y$ given $X = x$ is $f_\theta(y|x)$ where the parametric form $f_\theta$ is known, but $\theta \in \Theta$ is some unknown parameter.
- In the generative case, $f_\theta(y|x) \propto \pi_\theta(x, y)$, where $\pi_\theta$ is the known parametric form of the joint distribution
- The maximum likelihood estimate of $\theta$ based on the training dataset $d$ is

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log(f_\theta(y_i|x_i)) \quad \text{[Conditional]}$$

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log(\pi_\theta(x_i, y_i)) \quad \text{[Generative]}$$

- The associated estimate of the pmf/pdf is therefore $f_{\widehat{\theta}}$, leading to the plug-in estimators

$$\widehat{h}^{(d)}(x) = \int_{\mathbb{R}} y f_{\widehat{\theta}}(y|x) dy \text{ for regression}$$

$$\widehat{h}^{(d)}(x) = \arg\max_{k=1,\ldots,K} f_{\widehat{\theta}}(k|x) \text{ for classification}$$

55

# May 7 4pm

07 May 2021 15:43

- MSc Exam 2019 Q3 a)ii)
- MSc Exam 2019 Q4
- Left-over Question from this morning:
  We talked about some surrogate loss to replace 0-1 loss in lecture 11. I'm wondering if we use weighted loss rather than 0-1 loss, do we still have surrogate loss defined and how they will be like? Could you show an example like deriving the form of logistic surrogate loss if we use weighted loss in the ERM?
- Miscellaneous Questions:
  1. For lecture3, slides 25, we discussed "Non-convex cluster shapes". Do you mean we want the data to be Non-convex cluster shapes? Because based on what you draw during the lecture, it seems to be difficult for K-means to classify the data when it is convex.
  2. For lecture 3, on slide 30, could you explain how you get log K * n bits?
  3. For lecture 4, slide 16, could you show how you get the between-class variance? What I got is 2\mu^2, instead of \mu^2, so I hope to see which step I went wrong.
  4. For Lecture 8-9, slide 54&55, could you show more details of getting error or order during the Training-Validation-Test procedure?
  5. For Lecture 8-9, on slide 65, we've proved the discriminant function is a/a+b in the problem sheet. However, I still feel a little confused about the thinking process to get it. Could you list out the steps.
  6. For Lecture 11, slide 26, I understand the formula for \beta^(t+1) above, yet I felt confused about how we get the arg min formula, could you explain more?
  7. For Lecture 15, slide 26, we discussed how to split data for decision trees, it says taking O(pnlogn) time. Could you explain how do we get it?
  8. In the last section about boosting. Given that both bagging and boosting are based on ensembles of many individual predictors, why is bagging a variance reduction technique while boosting is a bias reduction technique?
  9. For the part of solution to question 3d in sheet 4, what is R~x(h*) here and how to deduce RHS from LHS?

3. (a) [10 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 a matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$. You can assume the data is mean-centered: $\sum_i x_{ij} = 0, \forall j$.

   (i) You want to perform principal component analysis (PCA) on the data set $\mathbf{X}$. Assume we have fewer observations than features, so that $n < p$. What effects will this have on the rank of the covariance matrix $\mathbf{S} = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$? Will you be able to run the PCA algorithm?

   (ii) Consider the matrix $\mathbf{B} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^\top$. Show that we can perform PCA on either the matrix $\mathbf{B}$ or the covariance matrix, obtaining the same result (state any assumptions you are making and define any matrix decomposition you may use). When would you choose to perform PCA using $\mathbf{B}$ instead of the covariance matrix?

MSc exam 2019

$B = \frac{1}{n-1} XX^T$ B is a n-by-n matrix $S = \frac{1}{n-1} X^T X$ is p-by-p.

$(n-1)B = XX^T$ $(n-1)S = X^T X$ have the same non-zero eigenvalues.

$X = U D V^T$

U: n-by-n orthogonal; D: diagonal n×p; V: p×p orthogonal

$(n-1)B = XX^T = U D V^T V D^T U^T = U D D^T U^T$ (D D^T: diagonal matrix of size n×n)

$(n-1)S = X^T X = V D^T U^T U D V^T = V D^T D V^T$ (D^T D: diagonal matrix of size p×p)

→ B and S have the same non-zero eigenvalues.

$Z = XV = U D V^T V = UD$

if $p > n$ S is p×p (compute it $O(p^2 n)$)

B is n×n (compute it $O(n^2 p)$)

Q4 b)

(ii) Consider the data set in Table 1. Each row of the table represents a possible value of two input features and the corresponding label. Features and labels are all binary. Each possible feature combination may be observed multiple times in the data set. For instance, the point $\mathbf{x} = [1, 1]^\top$ has been observed 4 times with label $y = 1$ and

$x_i \in \{0,1\}^2$

0 times with label $y = 0$. Run a single iteration of a classification tree algorithm (i.e. perform a single split) using this data and entropy as a measure of split quality, obtaining a tree of depth 1 (which is also known as a "stump"). Report as much explicit calculation as is needed to fully justify your results.
[Hint: use base-2 logarithms and consider a split on $X_1$ first. This may simplify calculations.]

$p = 2$

$y \in \{0, 1\}$

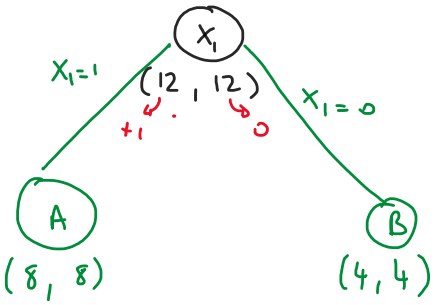
Table 1: Classification tree data set.

| $X_1$ | $X_2$ | $y$ | count |
|---|---|---|---|
| 1 | 1 | 1 | 4 |
| 1 | 0 | 1 | 4 |
| 0 | 1 | 1 | 3 |
| 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 8 |
| 0 | 1 | 0 | 2 |
| 0 | 0 | 0 | 2 |

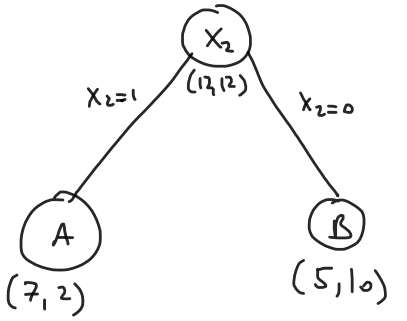
(iii) Assume you trained a full tree (reaching maximum depth) using this data set, and assume that you classify using a majority vote. Would you be able to achieve perfect classification accuracy on the training data?

(iv) Describe one approach you may use to prevent overfitting when using decision tree learning.

Split (according to first dimension)




Split (according to $x_2$)




change in impurity of split: Change of impurity of $1 - 1 = 0$

$q_{left} \times i(\hat{p}_{left}) + q_{right}\, i(\hat{p}_{right}) = 1$

Change in impurity: $1 - 0.86$ or

Split 1: $q_{left} = \frac{16}{24}$ $q_{right} = \frac{8}{24}$

$\hat{p}_{left} = \hat{p}_{right} = \frac{1}{2}$

$i(\hat{p}_{left}) = i(\hat{p}_{right}) = \frac{1}{2}$

Favour second split as it leads to reduced impurity.

b) What is meant by bias and variance of an algorithm?

For an input $x$

$b(x) = \mathbb{E}_D[\hat{h}^{(D)}(x)] - h^*(x)$ ← Bayes prediction rule

$v(x) = \mathrm{Var}_D(\hat{h}^{(D)}(x))$

$\mathbb{E}_D[(\hat{h}^{(D)}(x) - h^*(x))^2] = b(x)^2 + v(x)$

$\mathbb{E}_x\,\mathbb{E}_D[(\hat{h}^{(D)}(x) - h^*(x))^2] = \mathbb{E}_x[b(x)^2] + \mathbb{E}_x[v(x)]$.

bias part — variance part

$D_1, \cdots, D_B$ iid with the same distribution as $D$

$$\hat{h}_{agg}(x) = \frac{1}{B}\sum_{j=1}^{B} \hat{h}^{(D_j)}(x)$$

$$\mathbb{E}\left[\hat{h}^{(D_i)}(x)\right] = \bar{h}(x)$$

$$\mathbb{E}_{D_1,\cdots,D_B}\left[\hat{h}_{agg}(x)\right] = \frac{1}{B}\sum_{j} \mathbb{E}_{D_j}\left[\hat{h}^{(D_j)}(x)\right] = \bar{h}(x)$$

We talked about some surrogate loss to replace 0-1 loss in lecture 11. I'm wondering if we use weighted loss rather than 0-1 loss, do we still have surrogate loss defined and how they will be like? Could you show an example like deriving the form of logistic surrogate loss if we use weighted loss in the ERM?

PS4: $L(y, h(x)) = \mathbb{1}_{y \neq h(x)}$

$\psi$-risk: $R_\psi(f) = \mathbb{E}\left[\psi(Y f(X))\right]$

$\psi_1(z) = e^{-z}$ [boosting]

$\psi_2(z) = (1-z)^2$ [least-square]

$\psi_3(z) =$

## K-means Additional Comments

- **Good practice initialisation.** Randomly pick $K$ training examples (without replacement) and set $\mu_1, \mu_2, \ldots, \mu_K$ equal to those examples
- **Sensitivity to distance measure.** Euclidean distance can be greatly affected by measurement unit and by strong correlations. Can use Mahalanobis distance instead:

$$\|x - y\|_M = \sqrt{(x-y)^\top M^{-1}(x-y)}$$

where $M$ is positive semi-definite matrix, e.g. sample covariance.
- **Sensitivity to outliers.**
- **Non-convex cluster shapes.**

## Vector Quantisation

- A related algorithm developed in the signal processing literature for **lossy data compression**.
- $\mathbf{X}$ represented by $n \times p$ real numbers
- Store instead:
  - the **codebook** of $K$ **codewords** $\mu_1, \ldots, \mu_K$ ($K \times p$ real numbers)
  - for each vector $x_i$ its cluster assignment $z_i$ ($\lceil \log K \rceil \times n$ bits).
- As with K-means, $K$ must be specified. Increasing $K$ improves the quality of the compressed image but worsens the data compression rate, so there is a clear tradeoff.
- Some audio and video codecs use this method.
- Stochastic optimization algorithm for K-means was originally developed for VQ.

## Decomposition of the variance and class separability

- If $X \in \mathbb{R}$, the ratio

$$\frac{\mathrm{cov}(\mathbb{E}[X \mid Y])}{\mathbb{E}[\mathrm{cov}(X \mid Y)]} \geq 0$$

between the between-class and within-class variances is a measure of the separability between the classes. Larger values indicate more separated classes.
- For instance, if $\Pr(Y = 1) = \Pr(Y = -1) = 1/2$ and $X \mid Y = y \sim \mathcal{N}(y\mu, \sigma^2)$, for some $\mu \geq 0$ and $\sigma > 0$, we have

$$\mathbb{E}[\mathrm{cov}(X \mid Y)] = \sigma^2, \;\; \mathrm{cov}(\mathbb{E}[X \mid Y]) = \mu^2.$$

$\sigma^2$ controls the within-class variance, and $\mu^2$ the between-class variance.

## Training-Validation-Test procedure

- In Step 1, as we use the training set $d_{\text{train}}$ to estimate the $\widehat{h}_j^{(d_{\text{train}})}$, we cannot use $\widehat{R}^{(d_{\text{train}})}(\widehat{h}_j^{(d_{\text{train}})})$ as an estimate for $R(\widehat{h}_j^{(d_{\text{train}})})$
- We therefore use in step 2a) another dataset (the validation set), that has not been used for training
- For any $j$, and any training set $d_{\text{train}}$,

$$\mathbb{E}[\widehat{R}^{(D_{\text{val}})}(\widehat{h}_j^{(D_{\text{train}})}) \mid D_{\text{train}} = d_{\text{train}}] = R(\widehat{h}_j^{(d_{\text{train}})})$$

  where the expectation is taken over the validation random sample.
- Unbiased estimate of the true risk of the $j$th learned prediction rule.
- Error of order $O(1/n_{\text{val}})$

## Iterative reweighted least squares

- Let $\beta^{(t)}$ be the parameter value after $t$ iterations of the Newton-Raphson algorithm. Let $\mu^{(t)}$ and $S^{(t)}$ be the corresponding vectors and matrices.
- Newton-Raphson update

$$\begin{aligned}
\beta^{(t+1)} &= \beta^{(t)} - (\nabla^2_\beta J(\beta^{(t)}))^{-1} \nabla_\beta J(\beta^{(t)}) \\
&= \beta^{(t)} + (\mathbf{\Phi}^\intercal S^{(t)} \mathbf{\Phi})^{-1} \mathbf{\Phi}^\intercal (c - \mu^{(t)}) \\
&= (\mathbf{\Phi}^\intercal S^{(t)} \mathbf{\Phi})^{-1} \mathbf{\Phi}^\intercal S^{(t)} (\mathbf{\Phi}\beta^{(t)} + (S^{(t)})^{-1}(c - \mu^{(t)})) \\
&= (\mathbf{\Phi}^\intercal S^{(t)} \mathbf{\Phi})^{-1} \mathbf{\Phi}^\intercal S^{(t)} z^{(t)}
\end{aligned}$$

  where $z^{(t)} = \mathbf{\Phi}\beta^{(t)} + (S^{(t)})^{-1}(c - \mu^{(t)})$. Then $\beta^{(t+1)}$ is a solution of the weighted least squares problem

$$\begin{aligned}
\beta^{(t+1)} &= \arg\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} S_{ii}^{(t)} (z_i^{(t)} - \phi(x_i)^\intercal \beta)^2 \\
&= \arg\min_{\beta \in \mathbb{R}^p} (z^{(t)} - \mathbf{\Phi}\beta)^\intercal S^{(t)} (z^{(t)} - \mathbf{\Phi}\beta).
\end{aligned}$$

## Computational Considerations

## Computational Considerations

- Numerical input features $x_{ij}$
  - We could split on any feature, with any threshold
  - However, for a given feature, the only split points we need to consider are the $n$ values in the training data for this feature.
  - If we sort each feature by these $n$ values, we can quickly compute our metric of interest (squared loss or impurity)
    - This takes $O(pn \log n)$ time
- Categorical input features $x_{ij}$
  - Assuming $q$ distinct categories, there are $2^{q-1} - 1$ possible partitions we can consider.

**Solution:** The first inequality follows by definition of the Bayes classifier. As $\eta$ is continuous, $X_{n,(1)} \to \widetilde{x}$ implies $\eta(X_{n,(1)}) \to \eta(\widetilde{x})$ almost surely. Therefore, for any fixed $\widetilde{x}$,

$$\begin{aligned} g(\widetilde{x}, X_{n,(1)}) &:= \Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}, X_{n,(1)}) \\ &= \eta(\widetilde{x})(1 - \eta(X_{n,(1)})) + (1 - \eta(\widetilde{x}))\eta(X_{n,(1)}) \\ &\to 2\eta(\widetilde{x})(1 - \eta(\widetilde{x})) \end{aligned}$$

almost surely as $n \to \infty$. Using the dominated convergence theorem, for any $\widetilde{x}$

$$\begin{aligned} \Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X} = \widetilde{x}) &= \mathbb{E}[g(\widetilde{x}, X_{n,(1)})] \\ &\to 2\eta(\widetilde{x})(1 - \eta(\widetilde{x})) \end{aligned}$$

as $n \to \infty$. Note that

$$2\eta(\widetilde{x})(1 - \eta(\widetilde{x})) = 2R_{\widetilde{x}}(h^\star)(1 - R_{\widetilde{x}}(h^\star))$$

and

$$\mathbb{E}[R(\widehat{h}^{(D_n)})] = \Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}))$$

Another application of the dominated convergence theorem yields

$$\begin{aligned} \Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X})) &= \mathbb{E}_{\widetilde{X}}[\Pr(\widetilde{Y} \neq \widehat{h}^{(D_n)}(\widetilde{X}) \mid \widetilde{X})] \\ &\to 2\,\mathbb{E}[R_{\widetilde{X}}(h^\star)(1 - R_{\widetilde{X}}(h^\star))] \end{aligned}$$

as $n \to \infty$. Finally, (using Jensen's inequality, or just developing the expression)

$$\begin{aligned} \mathbb{E}[R_{\widetilde{X}}(h^\star)(1 - R_{\widetilde{X}}(h^\star))] &\leq \mathbb{E}[R_{\widetilde{X}}(h^\star)](1 - \mathbb{E}[R_{\widetilde{X}}(h^\star)]) \\ &= R(h^\star)(1 - R(h^\star)). \end{aligned}$$

# May 14 10:30am

14 May 2021 10:20

MSc Exam 2018 Q3c) (neural networks)
Part B Exam 2018

- Q1b,c (Naive Bayes, Bias/variance)
- Q2 c (Newton-Raphson)
- Q3 b) c)iii-iv) (trees, adaboost)

Other Questions:

- Could you please explain the graph on slides 38 of lecture slides04_05 in a bit more detail? For example, what is the expression for the projection direction of the data and how the black dashed line is drawn. Does the black dashed line always pass through the intersection point of two ellipse?
- For slides 11 of lecture slides12, for the regression case, I can understand that the nonparametric estimator for conditional cdf take that form. But how can we derive the fact that the estimates are average of y_i, i \belongs to knn(x) from this conditional cdf?
- For Q1 (a) in problem sheet 3, you mentioned in class that logistic regression would have higher variance. However, in this case, I think logistic regression only have (p+1) parameters while naïve bayes has (1+2p+p).

(c) [6 marks] Consider a binary classification problem with training data $\{(x_i, y_i)\}_{i=1}^n$, with $x_i \in \mathbb{R}^2$ and $y_i \in \{0, 1\}$. Connect each decision boundary (a, b, c, d) shown on the left in figure 1 to any neural network on the right (1, 2, 3, 4) that could have produced it as the classification output. You can assume that each network output has been binarized using a suitable threshold rule. (Note that more than one neural network may generate the same decision boundary.)

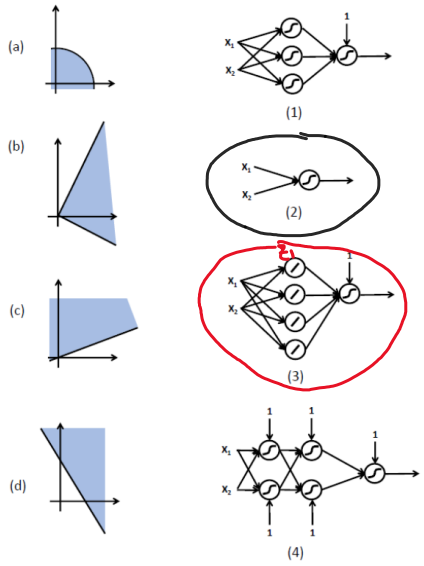



Figure 1: Decision boundaries and neural networks for Q1(c).

No hidden unit
$M(x_i) = \tau(w_1 x_{i1} + w_2 x_{i2})$
(2) → (c)

Linear classifier: with boundary: $a x_{i1} + b x_{i2} = 0$

$M(x_i) = \tau(w^{(0)T} \times z_i + b^{(0)})$
$z_i = W^{(1)T} \times x_i$

$M(x_i) = \tau(\tilde{w}^T x_i + b^{(0)})$
$\tilde{w} = W^{(1)} \times w^{(0)}$

⇒ Linear classifier with boundaries: $a x_{i1} + b x_{i2} + c = 0$
(3) → (c)(d).

(1), (4) → (a, b, c, d)

Part B 2018

(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

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

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

$(X, Y)$ $X = (X_1, \ldots, X_p)$ $X_j \in \{0, 1\}$ $\pi_k = P(Y = k)$
$Y \in \{0, 1\}$.

Naive Bayes assumption: $P(X = x | Y = k) = \prod_{j=1}^{p} P(X_j = x_j | Y = k)$

$P(X_j = 1 | Y = k) = \phi_{kj}$

From Q1.i) :

classifier: $h(x) = \arg\max_k \log \pi_k + \sum_{j=1} \log \mathbb{P}(X_j = x_j \mid Y = k)$

$$\sum_{j=1}^{p} \log P(X_j = x_j \mid Y = k) = \sum_{j=1}^{p} \left( x_j \log \phi_{kj} + (1 - x_j) \log (1 - \phi_{kj}) \right)$$

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

$$f_{Bayes}(x) = \arg\max_{k=0,1} a_k + b_k^T x$$

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

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

(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.

(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 $p$).

(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.

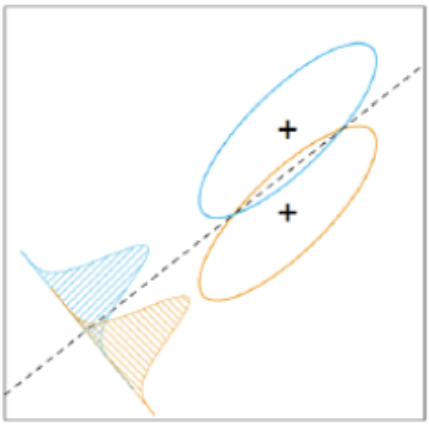
(c) [9 marks] Consider the following training points for a binary classification problem:

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

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.



## kNN as a plug-in method

- Can interpret kNN as a plug-in approach
- For classification, $\Pr(Y = c \mid X = x)$ is approximated by

$$\frac{1}{k} \sum_{i \in \text{knn}(x)} \mathbb{1}_{y_i = c}$$

- For regression, the conditional cdf $\Pr(Y \leq y \mid X = x)$ is approximated by

$$\frac{1}{k} \sum_{i \in \mathbf{knn}(x)} \mathbb{1}_{y_i \leq y}$$

- Nonparametric estimators of the conditional distribution

# May 14 4pm

14 May 2021 11:58

I probably won't have time to cover all questions - leftover questions will be covered the following week

Part B Exam 2019
- Q1ii, b/i,ii (PCA)
- Q2 (bias/variance, adaboost)
- Q3b (curse of dimensionality in knn)

Part B Exam 2020
- Q1b iii, iv (ridge regression and kernels)

Part C 2015 paper:
- Q1b (knn theory: multiclass extension of PS4 Q3)

Part B 2017 paper:
- Q1a (bias/variance)
- Q2d (logistic regression and bias/variance tradeoff)

Part B 2017

1. (a) [6 marks] Consider a supervised learning problem.
   (i) Briefly describe what is meant by the *bias/variance tradeoff* in machine learning.
   (ii) Name and describe a method that might be used with the goal of decreasing variance while increasing bias.
   (iii) On a particular problem, would you expect Linear Discriminant Analysis (LDA) or Quadratic Discriminant Analysis (QDA) to exhibit more bias? Why?

Q2 d)

(d) [10 marks] Consider fitting logistic regression to a dataset $x_1, \ldots, x_n \in \mathbb{R}^d$, where each $x_i = (x_i^1, \ldots, x_i^d)^\top$, with labels $y_1, \ldots, y_n \in \{+1, -1\}$, and obtaining estimates $\widehat{a}$ and $\widehat{b}$ for the decision boundary $a + b^\top x$. Now consider refitting the model with an extra predictor variable included, so each $x_i = (x_i^1, \ldots, x_i^d, x_i^{d+1})$.
   (i) What effect do you expect to see on the training error? Will it increase, decrease, or stay the same? Briefly explain.
   (ii) What effect do you expect to see on the test error? Will it increase, decrease, or stay the same? Briefly explain.
   (iii) If you observe that the estimates of $a$ and $b$ diverge to infinity, what might have happened? What strategy would you use to fix this problem?

Part B - 2019

Q1.

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 $r$.

See PS I Q3. $L(x_i, V_{1:k} V_{1:k}^T x_i)$

$$\frac{1}{n-1} \sum_{i=1}^{n} \| x_i - V_{1:k} V_{1:k}^T x_i \|^2 = \sum_{i=k+1}^{n} \lambda_i = r$$

PCA autoencoder, using k first PCs.

$$= \text{trace}(S) - \sum_{i=1}^{k} \lambda_i$$

$$\text{trace}(S) = \sum_{i=1}^{p} \lambda_i \quad : \text{ total variance of } X$$

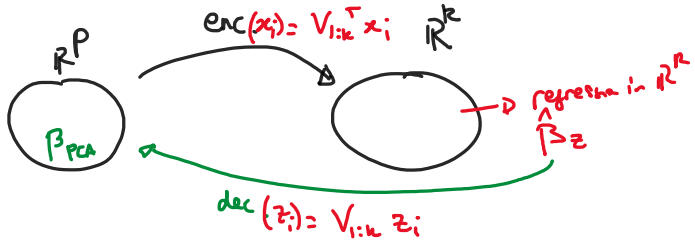
$$\sum_{i=1}^{k} \lambda_i \quad : \text{ total variance of } Z \text{ (Projection on first k PCs)}$$

(b) [13 marks] Now consider the same data set, with the addition of labels $\{(x_i, y_i)\}_{i=1}^n$ ...

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 $y = \beta^\top \mathbf{x} + \epsilon$, where $\epsilon$ 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: $\widehat{\beta}_Z = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top y$.
3. Transform parameters to original coordinates: $\widehat{\beta}_{PCA} = \mathbf{V}_{1:k} \widehat{\beta}_Z$.
4. Given a new data point $\mathbf{x}_{new}$, predict its label using $\widehat{y}_{new} = \widehat{\beta}_{PCA}^\top \mathbf{x}_{new}$.

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




$$\hat{\beta}_Z = (Z^T Z)^{-1} Z^T y$$

Projections on the PC components are centered, uncorrelated, with sample variance $= \lambda_i$

i) $Z^T Z = k \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_k \end{pmatrix}$

$X = U D V^T$ $\quad$ $D = n \begin{pmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_p} \\ & 0 & \end{pmatrix}$ $\quad$ $V_{1:k} = p \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$ (k columns)

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

$Z = X V_{1:k} = U \boxed{D V^T V_{1:k}}$ diagonal

$V^T V_{1:k} = p \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \\ & 0 & \end{pmatrix}$ $k$

$D V^T V_{1:k} = n \begin{pmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_k} \\ & 0 & \end{pmatrix}$ $\quad$ ($k$ columns)

$\beta_{PCA} = V_{1:k} \underbrace{(Z^T Z)^{-1} (D V^T V_{1:k})^T}_{n} \times U^T y$

$k \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sqrt{\lambda_k}} \\ & 0 & \end{pmatrix}$

$= V \Delta_{PCA} U^T y$ $\quad$ $\Delta_{PCA} = p \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sqrt{\lambda_k}} \\ & 0 & \end{pmatrix}$ $n$ (columns)

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

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

in i): $\hat{\beta}_{L2} = V \Delta_{L2} U^T y$ $\quad$ $\Delta_{L2}$ is diagonal

with value $\frac{\sqrt{\lambda_i}}{\lambda_i + \alpha}$

## Q2. (AdaBoost)

(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 $E \leqslant \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

i) $w_T(i) = \frac{1}{Z_T} \times w_{T-1}(i) \times e^{-\beta_t y_i h_t(x_i)}$

$= \frac{1}{N} \times \frac{1}{\prod_{t=1}^{T} Z_t} \prod_{t=1}^{T} e^{-\beta_t y_i h_t(x_i)}$

$= \frac{1}{N \prod_{t=1}^{T} Z_t} e^{-\sum_{t=1}^{T} \beta_t y_i h_t(x_i)}$

ii) $H(\cdot)$ the final classifier $H(x) = \text{sign}\left(\sum_{t=1}^{T} \beta_t h_t(x)\right)$

discriminant function: $F(x) = \sum_{t=1}^{T} \beta_t h_t(x)$

$E = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{H(x_i) \neq y_i}$

Show that $E \leq \prod_{t=1}^{T} Z_t$

$e^{-c} \geq 1$ if $c \leq 0$

$e^{-c} > 0$ if $c > 0$

$\Rightarrow e^{-c} \geq \mathbb{1}_{c<0}$ for each $i$

$\mathbb{1}_{F(x_i)y_i<0} \leq e^{-F(x_i)y_i}$

$E = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{F(x_i)y_i<0} \leq \frac{1}{N} \sum_{i=1}^{N} e^{-F(x_i)y_i}$

$w_T(i) = \frac{1}{N} \frac{e^{-y_i F(x_i)}}{\prod_t Z_t} \Rightarrow w_T(i) \prod_t Z_t = \frac{1}{N} e^{-y_i F(x_i)}$

$E \leq \sum_{i=1}^{N} w_T(i) \left(\prod_{t=1}^{T} Z_t\right) = \prod_{t=1}^{T} Z_t$

(iii) Prove $E \leq \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)}$

$$Z_t = \sum_{i=1}^{N} w_t(i) = \sum_{i=1}^{N} w_{t-1}(i)\, e^{-\beta_t y_i h_t(x_i)}$$

$$= \sum_{i: y_i = h_t(x_i)} w_{t-1}(i)\, e^{-\beta_t} + \sum_{i: y_i \neq h_t(x_i)} w_{t-1}(i)\, e^{+\beta_t}$$

$$= e^{-\beta_t} \underbrace{\sum_{i: y_i = h_t(x_i)} w_{t-1}(i)}_{1-\varepsilon_t} + e^{\beta_t} \underbrace{\sum_{i: y_i \neq h_t(x_i)} w_{t-1}(i)}_{\varepsilon_t}$$

$$= e^{-\beta_t}(1-\varepsilon_t) + e^{\beta_t}\varepsilon_t \qquad \beta_t = \frac{1}{2}\log\frac{1-\varepsilon_t}{\varepsilon_t}$$

$$= \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}(1-\varepsilon_t) + \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}\varepsilon_t$$

$$= 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

⚠ Notations in this Q + for lectures notes here

$\varepsilon_t = \sum w_{t-1}(i)\, \mathbb{1}_{y_i \neq h_t(x_i)}$

(⚠ typo in solution to Q26.ii)

Q

(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?

# May 21

- MSc 2020 Q4bii (neural nets)
- MSc 2018 Q4 a iii (feature expansion)
- MSc 2018 Q4 b iii (quadratic discriminant analysis)
- Part B 2020
  - Q1b iii, iv (ridge regression and kernels)
- Part C 2015
  Q1b (knn theory: multiclass extension of PS4 Q3)
- Questions:
  - What is the log-loss function for multi-class logistic regression?
  - In the context of plug in classifiers, can we use a loss function - as in does it make sense to use the squared loss for logistic regression?
  - Also in the context of plug ins, is the log loss the negative of the log likelihood?

From <https://canvas.ox.ac.uk/courses/65441/pages/revision-sessions-information>

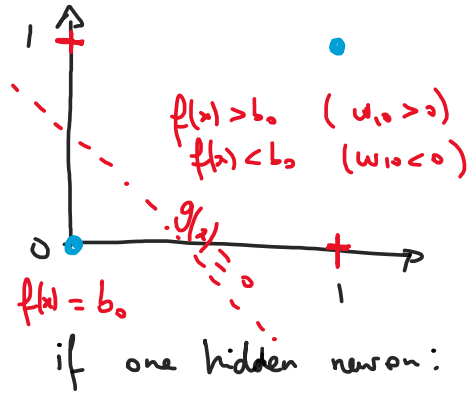
MSc 2020 Q4 b ii)

(ii) Consider a classification task with input data $x \in \mathbb{R}^2$ and output $y \in \{0,1\}$. The training data is given by:

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |

XOR

Deep learning look




if one hidden neuron:

$$f(x) = b_0 + w_{10} \max\left(0, b_1 + \sum_{i=1}^{2} w_{i1} x_i\right)$$

$g(x)$

You decide to use a neural network where the two inputs are connected to each of $m$ neurons in the hidden layer. Each neuron in the hidden layer builds a linear combination of its input and passes that through a rectified linear activation function (ReLu). The output of a neuron in the hidden layer is thus given by

$$h_\ell = \max\left\{0,\ b_\ell + \sum_{i=1}^{2} w_{i\ell}\, x_i\right\}, \qquad \ell \in \{1, \ldots, m\},$$

(note the presence of a bias term $b_\ell$). A linear combination of these values is then provided as the final prediction using

$$\widehat{y} = b_o + \sum_{\ell=1}^{m} w_{\ell o}\, h_\ell.$$

Determine the minimum number of neurons needed in the hidden layer $m$ so that the network will produce the correct label for all training data points. Given your choice of $m$, provide values for all network parameters (weights and biases) so that it will produce the correct label for all training data points.
[*Hint: you may want to first write an explicit expression for the function encoded by the neural network.*]

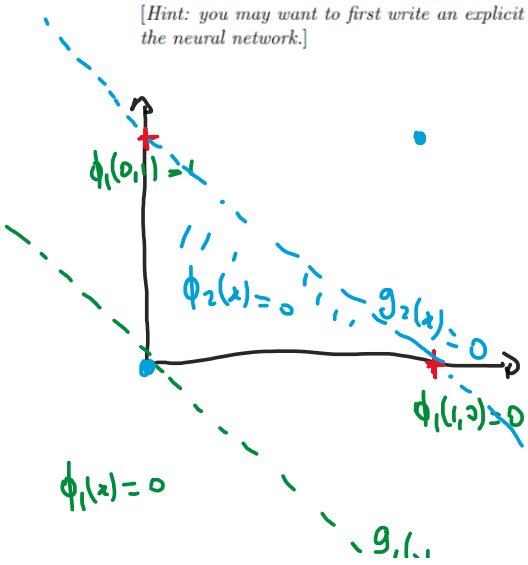
$b_0 = 0 \quad w_{10} = 1$

$m = 2$

Take $f(x) = \phi_1(x) - \alpha\, \phi_2(x)$

$$\phi_1(x) = \max\left(0, \underbrace{b_1 + \textstyle\sum w_{i1} x_i}_{g_1(x)}\right)$$

$$\phi_2(x) = \max\left(0, \underbrace{b_2 + \textstyle\sum w_{i2} x_i}_{g_2(x)}\right)$$




$g_1(x) = x_1 + x_2$

$\phi_1((0,0)) = 0 \qquad \phi_1(0,1) = \phi_1(1,0) = 1$

$w_1 = 0$ $\quad \psi_1(1,1) = \alpha$

$g_2(x) = x_1 + x_2 - 1$

$\phi_2(0,0) = \phi_2(1,0) = \phi_2(0,1) = 0$

$\phi_2(1,1) = 1$

Setting $\alpha = 2$ :

$f(1,1) = \phi_1(1,1) - 2\phi_2(1,1) = 0$

$\Rightarrow$ Perfect classification.

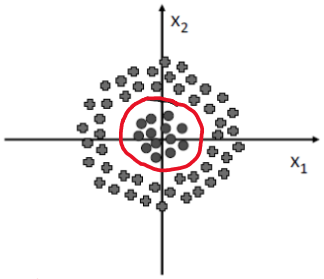
**Answer:** (N) 5 marks.
One neuron is not sufficient to represent the XOR function (bookwork). The network may be designed as in the figure, using $m = 2$. The function the network computes is given by

$$\hat{y} = \begin{bmatrix} w_3 \\ w_4 \end{bmatrix}^\top \max\left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \right\} + b$$

and $\{w_{11} = 1, w_{12} = 1, w_{21} = 1, w_{22} = 1, c_1 = 0, c_2 = -1, w_3 = 1, w_4 = -2, b = 0\}$ produce labels $[0, 0, 1, 1]^\top$ as desired. Note that there are multiple solutions, partial marks for writing the function correctly.

MSC 2018 Q4. a.(iii)

4. (a) [8 marks] You are working on a classification problem with training data $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{0, 1\}$. The training data are represented in Figure 2.
   (i) Briefly describe the notion of "linear separability" for a data set with two classes.
   (ii) You decide to use logistic regression for the data set in Figure 2. Will you be able to obtain perfect classification accuracy on the training data without applying any transformations? Justify your answer.
   (iii) Some data sets may be made linearly separable by applying a suitable transformation $\phi : \mathbb{R}^p \to \mathbb{R}^d$. Provide a choice of $\phi(\cdot)$ that would make the data set of Figure 2 linearly separable. State any assumptions you are making.




Classification boundary of the form $ax_1^2 + bx_2^2 = c$

$f(x) = \alpha + \beta^T \phi(x)$ $\quad \phi(x) = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}$

or

$f(x) = \beta^T \phi(x)$ $\quad \phi(x) = \begin{pmatrix} 1 \\ x_1^2 \\ x_2^2 \end{pmatrix}$ $\quad a(x_1 - \mu_1)^2 + b(x_2 - \mu_2)^2 = c$

Note: In solns: $\phi(x) = \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix}$

Q4 b ii

(iii) You are given a data set with data points $\{(x_i, y_i)\}_{i=1}^n$, with $x_i \in \mathbb{R}$ and $y_i \in \{0, 1\}$. Half of the points come from class $A$, and half come from class $B$. You assume that data from either class follow a Gaussian distribution. After computing the empirical mean and standard deviation for each class, you determine that the means are $\hat{\mu}_A = 0$ and $\hat{\mu}_B = 0$, while the standard deviations are $\hat{\sigma}_A = 3$, $\hat{\sigma}_B = 1$. For what values of $x$ will you classify a new data point as coming from class $B$?

Q4 b ii

**Answer:** (S). Assuming $g_k(x)$ is the density of a Gaussian distribution, and $\pi_k$ is the class probability,

$$\ln \widehat{\pi}_k g_k(x) = c + \left(\frac{x-\widehat{\mu}_k}{\widehat{\sigma}_k}\right)^2 - 2\ln(\widehat{\pi}_k) + 2\ln(\widehat{\sigma}_k)$$

Classify as A when $\ln \widehat{\pi}_A g_A(x) \geqslant \ln \widehat{\pi}_B g_B(x)$, i.e. since priors as the same, when

$$\left(\frac{x-\widehat{\mu}_A}{\widehat{\sigma}_A}\right)^2 + 2\ln(\widehat{\sigma}_A) \leqslant \left(\frac{x-\widehat{\mu}_B}{\widehat{\sigma}_B}\right)^2 + 2\ln(\widehat{\sigma}_B)$$

$$\frac{1}{\widehat{\sigma}_A^2}\left(x^2 - 2x\widehat{\mu}_A + \widehat{\mu}_A^2\right) - \frac{1}{\widehat{\sigma}_B^2}\left(x^2 - 2x\widehat{\mu}_B + \widehat{\mu}_B^2\right) + 2\ln\widehat{\sigma}_A - 2\ln\widehat{\sigma}_B \leqslant 0$$

$$x^2\left(\frac{1}{\widehat{\sigma}_A^2} - \frac{1}{\widehat{\sigma}_B^2}\right) + 2x\left(\frac{\widehat{\mu}_B}{\widehat{\sigma}_B^2} - \frac{\widehat{\mu}_A}{\widehat{\sigma}_A^2}\right) + \left(\frac{\widehat{\mu}_A^2}{\widehat{\sigma}_A^2} - \frac{\widehat{\mu}_B^2}{\widehat{\sigma}_B^2} + 2\ln\widehat{\sigma}_A - 2\ln\widehat{\sigma}_B\right) \leqslant 0$$

Given $\widehat{\mu}_A = 0$, $\widehat{\mu}_B = 0$, $\widehat{\sigma}_A = 3$, $\widehat{\sigma}_B = 1$, Classify B if

$$x \in \left(\pm\frac{3}{2}\sqrt{\ln 3}\right)$$

$\pi_k = \Pr(Y = k)$

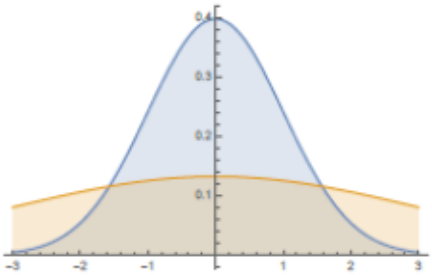



Figure 2: QDA solution sketch.

(iii) Computing $\widehat{\beta}$ involves the inversion of a matrix of size $p \times p$, which you would like to avoid, as it is computationally expensive (e.g. we have $p \gg n$). Assume the matrix has rank $r$, with $r \leqslant \min(n,p)$. As a speed-up, you decide to use the SVD decomposition $X = UDV^\top$, where $D = diag(s_i)$ is an $r \times r$ diagonal matrix with positive singular values $s_i$ at the $i$-th position of the $i$-th row, $U$ is an $n \times r$ matrix, and $V$ is an $r \times p$ matrix. Show that you can efficiently compute the optimal coefficients $\widehat{\beta}$ using

$$\widehat{\beta} = V diag\left(\frac{s_i}{s_i^2 + \lambda}\right) U^\top y.$$

**Answer: (N)**

$$\begin{aligned}
\widehat{\beta} &= (X^\top X + \lambda I)^{-1} X^\top y \\
&= (V diag\left(s_i^2 + \lambda\right) V^\top)^{-1} V D U^\top y \\
&= V diag\left(\frac{1}{s_i^2 + \lambda}\right) V^\top V D U^\top y \qquad (1) \\
&= V diag\left(\frac{s_i}{s_i^2 + \lambda}\right) U^\top y
\end{aligned}$$

(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* $f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$.]

(c) [*14 marks*] Consider a classification problem with a training dataset $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^p$ and $y_i \in \{1, 2, \ldots, K\}$. We denote by $g_k(x)$ the conditional density of $X$ given $Y = k$ and assume that $g_k(x) > 0$ for all $x \in \mathbb{R}^p$, and the class probabilities as $\pi_k = \mathbb{P}(Y = k)$. We further denote $q_k(x) = \mathbb{P}(Y = k|X = x)$. Consider the Bayes classifier (minimizing risk w.r.t. 0/1 loss $\mathbf{1}\{f(X) \neq Y\}$):

$$f_{\text{Bayes}}(x) = \underset{k \in \{1,\ldots,K\}}{\arg\max}\ \pi_k g_k(x).$$

(i) Write the conditional expected loss $R_{\text{Bayes}}(x) = \mathbb{P}[f_{\text{Bayes}}(X) \neq Y|X = x]$ at a given test point $X = x$ in terms of $\{q_k(x)\}_{k=1}^K$.

(ii) For $x \in \mathbb{R}^p$, denote by $x'$ its nearest neighbour in the training dataset. Define the 1-nearest neighbour classifier $f_{1\text{NN}}(x)$ for the problem above. Assuming that as $n \to \infty$ the training data fills the space such that $q_k(x') \to q_k(x)$, $\forall x, k$, give the limit $R_{1\text{NN}}(x)$ of $\mathbb{P}[f_{1\text{NN}}(X) \neq Y|X = x, X' = x']$ as $n \to \infty$, in terms of $\{q_k(x)\}_{k=1}^K$.

(iii) Denote $R_{\text{Bayes}} = \mathbb{E}_X[R_{\text{Bayes}}(X)]$ and $R_{1\text{NN}} = \mathbb{E}_X[R_{1\text{NN}}(X)]$. Show that

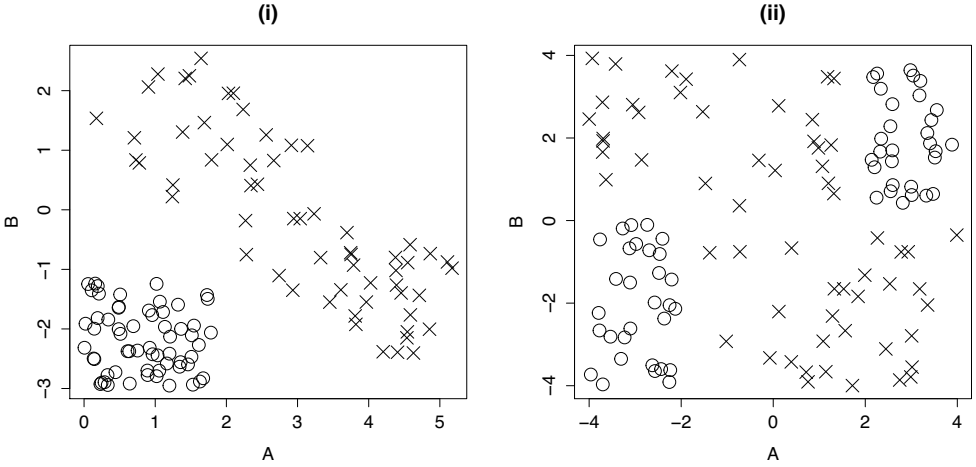
$$R_{1\text{NN}} \leqslant R_{\text{Bayes}}\left(2 - \frac{K}{K-1} R_{\text{Bayes}}\right).$$

When does the equality hold?

**Statistical Data Mining and Machine Learning (full question)**

7. (a) Which of the following classifiers are generative: linear discriminant analysis, logistic regression, support vector machine, naïve Bayes? Name one advantage of generative classifiers over discriminative ones.

   (b) You are given points from two classes: "○" and "×", which are described in terms of variables `A` and `B`. For each of the sets of points in the figures below, draw a decision tree of depth 2 that can separate the given data completely (by using thresholding of *a single variable* in each node).




   (c) Consider a binary classification problem with classes denoted $-1$ and $+1$. For a *soft classification rule* $f : \mathbb{R}^p \to \mathbb{R}$, the exponential loss is defined as $L(y, f(x)) = e^{-yf(x)}$.

      (i) Briefly comment on the difference between the exponential and the logistic loss in terms of how they treat the misclassified examples.

      (ii) Show that the optimal classification rule, i.e. $f^*$ which minimises the risk $R(f) = \mathbb{E}_{XY} e^{-Yf(X)}$, is given by

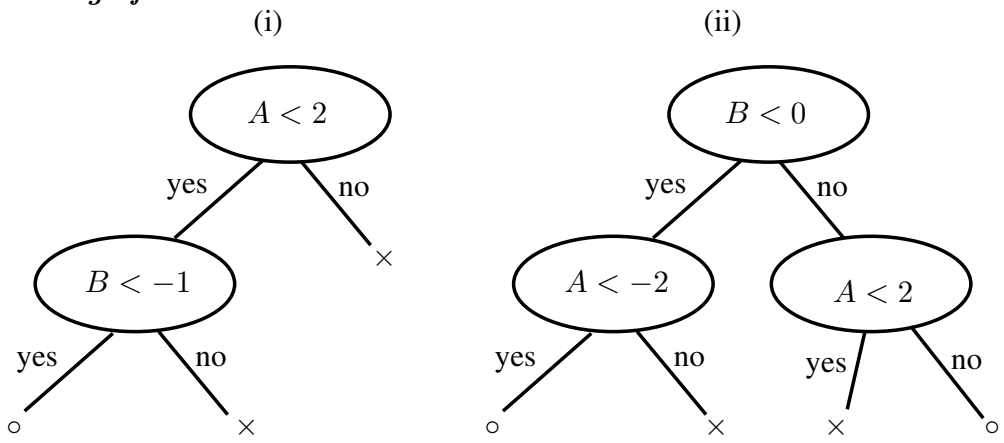
$$f^*(x) = \frac{1}{2} \log \frac{\mathbb{P}(Y = +1 | X = x)}{\mathbb{P}(Y = -1 | X = x)}.$$

      (iii) Consider the linear decision boundary $f(x) = w^\top x$. Write down the objective function $J(w)$ given by the $L_2$-regularised empirical risk with respect to the exponential loss. Derive the gradient descent update rule for $w$.

**Long Question 2**

(a) ***Answer: Bookwork. 4 marks***
Linear discriminant analysis and naïve Bayes are generative. Generative classifiers can more easily handle unlabeled training data and missing features. Generative model does not have to be retrained from scratch when we add more classes.

(b) ***Answer: Straightforward. 6 marks***




(c) ***Answer: (i) Straightforward, (ii) Straightforward. 10 marks.***

(i) The exponential loss is much more strict about very badly classified examples: where $y_i f(x_i) < 0$ and large in magnitude.

(i) It suffices to consider the expected conditional loss $\mathbb{E}\left[e^{-Yf(x)}|X=x\right]$. Let us denote $\pi(x) = \mathbb{P}\left(Y=+1|X=x\right)$. By differentiating and setting to zero

$$\begin{aligned}\frac{\partial}{\partial f(x)}\mathbb{E}\left[e^{-Yf(x)}|X=x\right] &= \frac{\partial}{\partial f(x)}\left[\pi(x)e^{-f(x)} + (1-\pi(x))e^{f(x)}\right] \\ &= -\pi(x)e^{-f(x)} + (1-\pi(x))e^{f(x)} \\ &= 0 \implies \frac{\pi(x)}{1-\pi(x)} = e^{2f(x)},\end{aligned}$$

which, since the second derivative is clearly positive, proves the claim.

(ii) By differentiating $J(w) = \frac{1}{n}\sum_{i=1}^{n} e^{-y_i x_i^\top w} + \lambda\|w\|_2^2$, we obtain

$$w^{\text{new}} = (1-\lambda)w + \frac{\epsilon}{n}\sum_{i=1}^{n} e^{-y_i x_i^\top w} y_i x_i.$$

The model is very similar to regularised logistic regression but uses a different loss, $L(y, f(x)) = e^{-yf(x)}$ rather than the log-loss $L(y, f(x)) = \log(1 + e^{-yf(x)})$. Logistic regression has an analogous update rule with logistic function $s(-y_i x_i^\top w) = e^{-y_i x_i^\top w}/(1 + e^{-y_i x_i^\top w})$ weighting $y_i x_i$ in the sum.

**Statistical Data Mining and Machine Learning (full question)**

7. (a) Explain the terms *training set*, *validation set*, *test set*, and *cross-validation*.

   (b) What is the difference between generative and discriminative modelling?

   (c) Consider a binary classification problem with dataset $(x_i, y_i)_{i=1}^n$ with $y_i \in \{+1, -1\}$. We use logistic regression to model the conditional distribution of the labels $(y_i)$ given the data vectors $(x_i)$. The objective function is

$$J(a, b) = \frac{C}{2}\|b\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i(a + b^\top x_i))).$$

   (i) Explain what each term in the objective function is and what it is for.

   (ii) Show that the objective function is convex. Why is convexity a useful property of the objective function?

   (iii) Suppose that the loss associated with incorrectly predicting $+1$ when the true label is $-1$ is 10 times larger than incorrectly predicting $-1$ when the true label is $+1$. How might you formulate an objective function for your logistic regression model that better reflects this unbalanced loss?

   (d) Consider a $K$-class classification problem with dataset $(x_i, y_i)_{i=1}^n$ with $y_i \in \{1, \ldots, K\}$. Suppose we model the conditional distribution of the labels as follows:

$$p(y_i = k|x_i) = \frac{\exp(a_k + b_k^\top x_i)}{\sum_{j=1}^K \exp(a_j + b_j^\top x_i)}.$$

   Write down the $L_1$-regularized empirical risk associated with the log loss for this model, and derive a gradient descent learning algorithm.

(Q7) -Wording of question has been adjusted for final / draft paper

**Long Question 7**

(a) Explain the terms training set, test set, validation set, and cross-validation.

***Answer: Bookwork. 5 marks.***
*Training set is the part of the data set used to optimize or learn the model parameters.*
*Test set is the part of the data set used to evaluate a learnt model.*
*Validation set is the part of the data set used to evaluate a learnt model with a particular tuning parameter setting, and which is used to choose the tuning parameter setting.*
*Cross-validation is a technique to reduce variance in the estimate of generalization performance used to determine tuning parameters. The training set is split into a number of folds, and on each round one fold is held as validation and rest as training set.*

(b) What is the difference between generative and discriminative modelling?

***Answer: Bookwork. 2 marks.***
*In generative modelling the whole data distribution is modelled, while in discriminative modelling only the conditional distribution of the labels or responses of interest are modelled.*

(c) Consider a binary classification problem with dataset $(x_i, y_i)_{i=1}^n$ with $y_i \in \{+1, -1\}$. We use logistic regression to model the conditional distribution of the labels given the data vectors. The objective function is

$$J(a, b) = \frac{C}{2}\|b\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i(a + b^\top x_i)))$$

(i) Explain what each term in the objective function is and what is it for.

***Answer: Bookwork. 3 marks.***
*The first term on RHS is the regularization term. The regularization used is $L_2$ regularization. $C$ is a tuning parameter governing how strong the regularization is. The second term is the empirical risk associated with the log loss of the logistic regression. The parameters to be learnt are $a$ and $b$.*

(ii) Show that the objective function is convex. Why is convexity a useful property of the objective function?

***Answer: Bookwork. 4 marks.***
*We check that second derivative is positive definite. The second derivative of regularization term is $CI \succ 0$. For the second term, for simplicity, we can concatenate $a$ and $b$ to form a vector of length $p + 1$, while we include another 1 as another entry in $x_i$. Effectively we drop the $a$ term. The first derivative is*

$$\sum_{i=1}^{n} s(y_i(b^\top x_i))y_i x_i^\top$$

*where $s$ is the logistic function. The second derivative is*

$$\sum_{i=1}^{n} s(y_i(b^\top x_i))(1 - s(y_i(b^\top x_i)))y_i^2 x_i x_i^\top$$

*$x_i x_i^\top$ is positive semidefinite, while $s(y_i(b^\top x_i))(1 - s(y_i(b^\top x_i)))y_i^2$ is positive, so we have a positive sum of positive semidefinite terms, so is positive semidefinite.*
*Convexity is important because it means there is only one optimum, and it can be found easily using a variety of methods.*

(iii) Suppose that the loss associated with incorrectly predicting $+1$ when the true label is $-1$ is 10 times larger than incorrectly predicting $-1$ when the true label is $+1$. How might you formulate an objective function for training your logistic regression model that better reflects this unbalanced loss?

***Answer: Harder. 2 marks.***
*We can use the following objective:*

$$J(a,b) = \frac{C}{2}\|b\|^2 + \sum_{i:y_i=+1} \log(1+\exp(-(a+b^\top x_i))) + 10 \sum_{i:y_i=-1} \log(1+\exp(+(a+b^\top x_i)))$$

*So we weight the negatively labelled data points 10 times more, to force method to get these data points right more often.*

(d) Consider a $K$-class classification problem with dataset $(x_i, y_i)_{i=1}^n$ with $y_i \in \{1, \ldots, K\}$.
Suppose we model the conditional distribution of the labels as follows:

$$p(y_i = k|x_i) = \frac{\exp(a_k + b_k^\top x_i)}{\sum_{j=1}^K \exp(a_j + b_j^\top x_i)}$$

Write down the $L_1$-regularized empirical risk associated with the log loss for this model, and derive a gradient descent learning algorithm.

***Answer: Harder. 4 marks.***
*The objective is*

$$J = \left( \sum_{k=1}^K C\|b_k\|_1 + \sum_{i:y_i=k} a_k + b_k^\top x_i \right) - \sum_{i=1}^n \log \sum_{j=1}^K \exp(a_j + b_j^\top x_i)$$

*The gradient is*

$$\begin{aligned}
\frac{dJ}{db_{k\ell}} &= C\,sign(b_{k\ell}) + \sum_{i:y_i=k} x_{i\ell} - \sum_{i=1}^n p(y_i = k|x_i) x_{i\ell} \\
&= C\,sign(b_{k\ell}) + \sum_{i=1}^n (\mathbf{1}(y_i = k) - p(y_i = k|x_i)) x_{i\ell} \\
\frac{dJ}{da_k} &= \sum_{i=1}^n (\mathbf{1}(y_i = k) - p(y_i = k|x_i))
\end{aligned}$$

*A gradient descent algorithm proceeds by taking steps downhill, so*

$$\begin{aligned}
a_k^{new} &= a_k - \epsilon \frac{dJ}{da_k} \\
b_{k\ell}^{new} &= b_{k\ell} - \epsilon \frac{dJ}{db_{k\ell}}
\end{aligned}$$



**End of Last Page**