STATISTICAL LEARNING IN PRACTICE Part III / Lent 2020
Example Sheet 3 (of 4) By courtesy of Dr T. Wang

You have the option to submit your answers to questions 1 and 5 to be marked. If you wish your answers to be marked, please leave them in my pigeon-hole in the central core of the CMS by 11am on 11th March.

1. Consider two-class data $(x_1, y_1), \dots, (x_n, y_n)$, where $x_i \in \mathbb{R}^p$, the class sizes are $N_1, N_2 \geq 1$ (so $N_1 + N_2 = n$), and y_i are coded numerically as $-n/N_1$ and n/N_2 for class 1 and 2, respectively, $i = 1, \dots, n$.

   (i) Show that LDA classifies $x \in \mathbb{R}^p$ to class 2 if
$$x^\top \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) > \frac{1}{2}\hat{\mu}_2^\top \hat{\Sigma}^{-1}\hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^\top \hat{\Sigma}^{-1}\hat{\mu}_1 + \log\left(\frac{N_1}{n}\right) - \log\left(\frac{N_2}{n}\right),$$
   and to class 1 otherwise.

   (ii) Consider the minimisation of the least squares criterion
$$\sum_{i=1}^{n}(y_i - \alpha - x_i^\top \beta)^2.$$
   Show that the solution $\hat{\beta}$ satisfies
$$\left\{(n-2)\hat{\Sigma} + \hat{B}\right\}\hat{\beta} = n(\hat{\mu}_2 - \hat{\mu}_1),$$
   where $\hat{B} := N_1(\hat{\mu}_1 - \hat{\mu})(\hat{\mu}_1 - \hat{\mu})^\top + N_2(\hat{\mu}_2 - \hat{\mu})(\hat{\mu}_2 - \hat{\mu})^\top$ and $\hat{\mu} = n^{-1}\sum_{i=1}^n x_i$.

   (iii) Show that $\hat{B}\hat{\beta}$ is in the direction of $(\hat{\mu}_2 - \hat{\mu}_1)$ and, thus,
$$\hat{\beta} \propto \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1).$$
   Therefore, the least squares regression coefficient is identical to the LDA coefficient, up to perhaps a scalar multiple.

   (iv) Show that this result holds for any (distinct) numerical coding of the two class labels.

   (v) Find the solution $\hat{\alpha}$, and hence the predicted value $\hat{y}_i = \hat{\alpha} + x_i^\top \hat{\beta}$. Consider the following rule: classify the ith observation observation to class 2 if $\hat{y}_i > 0$ and class 1 otherwise. Show that this is not the same as the LDA rule unless the classes have equal numbers of observations.

   (Hastie et al., *Elements of Statistical Learning*, Exercise 4.2)

   Solution

(i) The LDA can be phrases through a (linear) discriminant given by

$$\delta_\ell^{\mathrm{LDA}}(x) = -\frac{1}{2}\|\hat{\Sigma}^{-1/2}(x-\hat{\mu}_\ell)\|_2^2 + \log\left(\frac{N_\ell}{n}\right).$$

Thus, the LDA rule classifies x to class 2 if and only if

$$\delta_2^{\mathrm{LDA}}(x) > \delta_1^{\mathrm{LDA}}(x),$$

which simplifies to the desired inequality after expansion.

(ii) Write $\theta = (\alpha, \beta^\top)^\top$, $Y = (y_1, \dots, y_n)^\top$ and

$$X = \begin{pmatrix} 1 & x_1^\top \\ \vdots & \vdots \\ 1 & x_n^\top \end{pmatrix},$$

we have $\hat{\theta} = (X^\top X)^{-1}X^\top Y \implies (X^\top X)\hat{\theta} = X^\top Y$. Note that

$$X^\top Y = \begin{pmatrix} 0 & x_1y_1 + \cdots + x_ny_n \end{pmatrix} = \begin{pmatrix} 0 & n(\hat{\mu}_2 - \hat{\mu}_1) \end{pmatrix}.$$

Hence, we have

$$\begin{aligned} \hat{\alpha}n + (x_1 + \cdots + x_n)^\top\hat{\beta} &= 0 \\ (x_1 + \cdots + x_n)\hat{\alpha} + (x_1x_1^\top + \cdots + x_nx_n^\top)\hat{\beta} &= n(\hat{\mu}_2 - \hat{\mu}_1). \end{aligned}$$

Eliminate $\hat{\alpha}$ to obtain

$$\sum_{i=1}^n \left(x_ix_i^\top - n\hat{\mu}\hat{\mu}^\top\right)\hat{\beta} = n(\hat{\mu}_2 - \hat{\mu}_1),$$

where $\hat{\mu} = n^{-1}\sum_{i=1}^n x_i = n^{-1}(N_1\hat{\mu}_1 + N_2\hat{\mu}_2)$. The desired result follows by observing

$$\begin{aligned} \sum_{i=1}^n x_ix_i^\top - n\bar{x}\bar{x}^\top &= \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^\top \\ &= \sum_{i:y_i = -n/N_1} (x_i - \hat{\mu}_1)(x_i - \hat{\mu}_1)^\top + N_1(\hat{\mu}_1 - \hat{\mu})(\hat{\mu}_1 - \hat{\mu})^\top + \\ &\quad \sum_{i:y_i = n/N_2} (x_i - \hat{\mu}_2)(x_i - \hat{\mu}_2)^\top + N_2(\hat{\mu}_2 - \hat{\mu})(\hat{\mu}_2 - \hat{\mu})^\top \\ &= (n-2)\hat{\Sigma} + \hat{B}. \end{aligned}$$

(iii) Using that $\hat{\mu}_1 - \hat{\mu} = n^{-1}N_2(\hat{\mu}_1 - \hat{\mu}_2)$ and $\hat{\mu}_2 - \hat{\mu} = n^{-1}N_1(\hat{\mu}_2 - \hat{\mu}_1)$,

$$\hat{B}\hat{\beta} = \frac{N_1N_2}{n}(\hat{\mu}_2 - \hat{\mu}_1)\{(\hat{\mu}_2 - \hat{\mu}_1)^\top\hat{\beta}\} \propto \hat{\mu}_2 - \hat{\mu}_1.$$

Thus,

$$\hat{\Sigma}\hat{\beta} \propto (\hat{\mu}_2 - \hat{\mu}_1),$$

and the conclusion follows.

(iv) Note that none of the LDA estimated parameters depend on how we label the classes. Given any class labels ℓ_1 and ℓ_2 with $\ell_1 \neq \ell_2$, there is a linear map $g(\tilde{y}) = a\tilde{y} + b$ such that $a \neq 0$ and $\ell_1 = g(-n/N_1)$ and $\ell_2 = g(n/N_2)$. Let $\tilde{y}_i = g^{-1}(y_i)$, $\tilde{\alpha} = (\alpha - b)/a$ and $\tilde{\beta} = \beta/a$, we have

$$\arg\min_{\alpha,\beta} \sum_{i=1}^n (y_i - \alpha - x_i^\top \beta)^2 = \arg\min_{\tilde{\alpha},\beta} \sum_{i=1}^n (\tilde{y}_i - \tilde{\alpha} - x_i^\top \tilde{\beta})^2.$$

Since the optimiser of the right hand side is in the direction of the LDA coefficient by the previous part, the least squares regression coefficient $\hat{\beta}$ for β is also parallel to the LDA coefficient.

(v) From (ii), we see that

$$\hat{\alpha} = -\frac{1}{n}(x_1 + \cdots + x_n)^\top \hat{\beta} = -\hat{\mu}^\top \hat{\beta}.$$

Thus,

$$\hat{y}_i = (x_i - \hat{\mu})^\top \hat{\beta} = \left(x_i - \left(\frac{N_1}{n}\hat{\mu}_1 + \frac{N_2}{n}\hat{\mu}_2 \right) \right)^\top \hat{\beta}.$$

so classifying according to the sign of $\hat{y}_i$ amounts to classifying x_i to class 2 if and only if (recall that $\hat{\beta} = c\hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$)

$$cx^\top \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) > c\left(\frac{N_1}{n}\hat{\mu}_1 + \frac{N_2}{n}\hat{\mu}_2 \right)^\top \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1).$$

If $c > 0$, this is equal to the classification boundary described in (i) if and only if $N_1/n = N_2/n = 1/2$. Thus, we are only missing to show that $c > 0$: from the second display in (ii) and the proof of (iii),

$$(n-2)\hat{\Sigma}\hat{\beta} = (\hat{\mu}_2 - \hat{\mu}_1)\left[n - \frac{N_1 N_2}{n}(\hat{\mu}_2 - \hat{\mu}_1)^\top \hat{\beta} \right];$$

from the proof of (ii) and letting $X_{-1} = (x_1, \ldots, x_n)^\top$,

$$(\hat{\mu}_2 - \hat{\mu}_1)^\top \hat{\beta} = \hat{\beta}^\top \sum_{i=1}^n \left(\frac{1}{n} x_i x_i^\top - \hat{\mu}\hat{\mu}^\top \right) \hat{\beta} = \frac{1}{n} \left\| X_{-1}\hat{\beta} \right\|_2^2 - \hat{\alpha}^2;$$

noting that $\hat{\beta} = (X_{-1}^\top X_{-1})^{-1} X_{-1}^\top (Y - \hat{\alpha}\mathbf{1})$,

$$\left\| X_{-1}\hat{\beta} \right\|_2^2 \leq \|Y - \hat{\alpha}\mathbf{1}\|_2^2 = n\hat{\alpha}^2 + n\frac{n}{N_1}\frac{n}{N_2}$$

and $c \geq 0$ with equality if only if $Y - \hat{\alpha}\mathbf{1} \in \operatorname{span}(X_{-1})$. But if $c = 0$ then $\hat{\Sigma}\hat{\beta} = 0$ which, since $\hat{\Sigma}$ is invertible in general, implies $\hat{\beta} = 0$ and, in particular, $\hat{\mu}_1 = \hat{\mu}_2$, in which case both LDA and the linear regression classifier will classify every point as being in class 1.

2. A researcher wants to build an email spam classifier based on a training set of $n = 500$ emails. They have hand-picked 10 words/symbols that they believe to have the highest discriminating power: `dollar`, `winner`, `password`, `edu`, `credit`, `discount`, `as`, `I`, `fun`, `trial`, and performed a logistic regression in `R`. Each row in the dataset `spamfilter` represent one email. The first column (`spam`) encodes whether an email is spam (code 1) or not (code 0) and the remaining 10 columns the count number of times a particular word/symbol appears in the email. Part of the `R` output is shown below.

```
summary(glm(spam ~ dollar + winner + password + edu + credit + discount + as + I
+ fun + trial, family = binomial)

# Coefficients:
#                    Estimate Std. Error z value Pr(>|z|)
# (Intercept)       -5.391451   1.447857  -3.724   0.0002 ***
# dollar             1.859318   1.333052   1.395   0.1631
# winner             5.680691   1.955451   2.905   0.0037 **
# password           0.923072   1.268399   0.728   0.4668
# edu               -6.890095   1.857351  -3.710   0.0002 ***
# credit             2.269523   1.007462   2.253   0.0243 *
# discount           1.198028   1.785944   0.671   0.5023
# as                -3.176676   1.832839  -1.733   0.0831 .
# I                 -1.866328   0.846315  -2.205   0.0274 *
# fun                4.347929   1.104687   2.066   0.0388 *
# trial              0.864456   1.774681   0.487   0.6262
```

(i) Write down the algebraic form of the fitted logistic regression classifier ψ^{logit} and give the estimated coefficients. How would you interpret the coefficient estimate for covariate `dollar`?

(ii) Describe algebraically the decision boundary of this classifier. In practice, one may want to only classify an email as spam if the predicted spam probability is at least a given $q \in (0, 1)$. Let ψ_q^{logit} be the corresponding classifier. Show that $\psi^{\text{logit}} = \psi_{0.5}^{\text{logit}}$. What effect does varying q have on the decision boundary?

Instead of hand-picking significant words, the researcher now wants to simply include 5000 common English words into their logistic regression classifier. The researcher fitted a ridge logistic regression model.

(iii) Explain why unregularised logistic regression is non-identifiable in this case and how the ridge regularisation resolves the non-identifiability issue.

(iv) Write down the optimisation problem solved by the ridge logistic regression algebraically. Describe how gradient descent can be performed to estimate the model coefficients.

Solution

(i) Let $x = (x_j)_{j=1,\dots,10}$ be such that x_j is the number of times the jth word (in the list `dollar`, `winner`, `password`, `edu`, `credit`, `discount`, `as`, `I`, `fun`, `trial`) appears in an email. The logistic regression classifier is of the form

$$\psi^{\text{logit}}(x) = \begin{cases} 1 & \hat{\beta}_0 + x^\top \hat{\beta} \geq 0 \\ 0 & \text{otherwise,} \end{cases}$$

where the coefficients are estimated by maximum likelihood estimation assuming an underlying logistic regression model on the responses. From this data, the estimated coefficients are $\hat{\beta}_0 = -5.39$ and
$\hat{\beta} = (1.86, 5.68, 0.923, -6.89, 2.27, 1.20, -3.18, -1.87, 4.35, 0.864)^\top$. The coefficient estimate for `dollar` is 1.86 and it means that every additional appearance of the word dollar in an email increases the log-odds of the email being spam by 1.86.

(ii) The decision boundary is the hyperplane

$$\{x \in \mathbb{R}^{10} : \hat{\beta}_0 + x^\top \hat{\beta} = 0\}.$$

The predicted spam probability given x is given by $\frac{e^{\hat{\beta}_0 + x^\top \hat{\beta}}}{1 + e^{\hat{\beta}_0 + x^\top \hat{\beta}}}$. Hence the decision boundary for ψ_q^{logit} is given by

$$\{x \in \mathbb{R}^{10} : \hat{\beta}_0 + x^\top \hat{\beta} = \operatorname{logit} q\}.$$

When $q = 0.5$, $\operatorname{logit} q = 0$ and we recover ψ^{logit}. In general, varying q has the effect of parallel shifting the decision boundary.

(iii) The log-likelihood for (β_0, β) is given by

$$\ell(\beta_0, \beta) = \sum_{i=1}^{n} \left\{ y_i \log\left(\frac{e^{\beta^\top x_i + \beta_0}}{e^{\beta^\top x_i + \beta_0} + 1} \right) + (1 - y_i) \log\left(\frac{1}{e^{\beta^\top x_i + \beta_0} + 1} \right) \right\}.$$

For any $\hat{\beta}$ and $\hat{\beta}_0$ maximising the log-likelihood, let $z_i = \hat{\beta}^\top x_i + \hat{\beta}_0$ for all i. Then the simultaneous equations

$$z_i = \beta^\top x_i + \beta_0 \qquad i = 1, \ldots, n,$$

has infinitely many solutions (including $\hat{\beta}, \hat{\beta}_0$) because we have more variables than equations. This results in non-unique optimiser of the log-likelihood and thus the problem is non-identifiable.

The ridge regularisation solves the non-identifiability issue by minimising instead the penalised problem

$$-\ell(\beta_0, \beta) + \lambda \|\beta\|_2^2$$

for some $\lambda > 0$. This is because the Hessian of this quantity is positive definite as soon as $\lambda > 0$ (unlike when $\lambda = 0$, when it is positive semi-definite because $5000 > 500$; you should check both), so if there is an optimiser, it is unique.

(iv) The ridge regression minimises the following objective

$$\ell(\beta_0,\beta) = -\sum_{i=1}^n \left\{ y_i \log\left(\frac{e^{\beta^\top x_i+\beta_0}}{e^{\beta^\top x_i+\beta_0}+1}\right) + (1-y_i)\log\left(\frac{1}{e^{\beta^\top x_i+\beta_0}+1}\right)\right\} + \lambda\|\beta\|_2^2,$$

which has gradients

$$\frac{\partial \ell}{\partial \beta_0} = -\sum_{i=1}^n \left(y_i - \frac{e^{\beta^\top x_i+\beta_0}}{e^{\beta^\top x_i+\beta_0}+1}\right) \quad\text{and}\quad \frac{\partial \ell}{\partial \beta} = -\sum_{i=1}^n x_i\left(y_i - \frac{e^{\beta^\top x_i+\beta_0}}{e^{\beta^\top x_i+\beta_0}+1}\right) + 2\lambda\beta.$$

So the gradient descent algorithm has the following form:

- Initialise $\hat\beta^{(0)}$ and $\hat\beta_0^{(0)}$.
- For $t = 1, 2, \ldots$, update

$$\hat\beta^{(t)} \leftarrow \hat\beta^{(t-1)} + \alpha_t\left\{\sum_{i=1}^n x_i\left(y_i - \frac{e^{(\beta^{(t-1)})^\top x_i+\beta_0^{(t-1)}}}{e^{(\beta^{(t-1)})^\top x_i+\beta_0^{(t-1)}}+1}\right) - 2\lambda\beta^{(t-1)}\right\}$$

and

$$\hat\beta_0^{(t)} \leftarrow \hat\beta_0^{(t-1)} + \alpha_t\sum_{i=1}^n \left(y_i - \frac{e^{(\beta^{(t-1)})^\top x_i+\beta_0^{(t-1)}}}{e^{(\beta^{(t-1)})^\top x_i+\beta_0^{(t-1)}}+1}\right)$$

3. Suppose we have observations $x_1, \ldots, x_n \in \mathbb{R}^p$ with associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$. Assume that the two classes are completely separable.

(i) State the quadratic optimisation problem with linear constraints solved by the support vector machine (SVM) classifier.

(ii) Characterise the support vectors for this SVM in terms of the solution to the optimisation problem in (i).

(iii) Let $SVs = \{i : x_i \text{ is a support vector}\}$. Prove that the decision boundary of this SVM will not change if we use only $\{(x_i, y_i) : i \in SVs\}$ as our training data.

(iv) Suppose $SVs = \{1, 2\}$. Describe the decision boundary in terms of x_1, x_2. What is it if $SVs = \{1, 2, 3\}$?

Solution

(i) The optimisation problem solved is

$$\min_{\beta\in\mathbb{R}^p,\beta_0\in\mathbb{R}} \frac{1}{2}\|\beta\|_2^2 \quad \text{subject to } y_i(x_i^\top\beta + \beta_0) \geq 1, \quad i = 1, \ldots, n. \tag{1}$$

(ii) Let β^* and β_0^* be the optimiser of the above optimisation problem. The set of support vectors are $\{x_i : y_i(x_i^\top\beta^* + \beta_0^*) = 1\}$.

(iii) Let $\beta^\dagger$ and $\beta_0^\dagger$ be the optimiser of the SVM problem using $\{(x_i, y_i) : i \in SVs\}$ as training data, we want to show that $\beta^* = \beta^\dagger$ and $\beta_0^* = \beta_0^\dagger$. Since the optimisation problem with support vectors only has less constraints than those in (1),

$$\frac{1}{2}\|\beta^\dagger\|_2^2 \leq \frac{1}{2}\|\beta^*\|_2^2.$$

For any $h \in [0,1]$, let $\beta^{(h)} = h\beta^\dagger + (1-h)\beta^*$ and $\beta_0^{(h)} = h\beta_0^\dagger + (1-h)\beta_0^*$. Then by convexity of the L_2 norm, we have

$$\frac{1}{2}\|\beta^{(h)}\|_2^2 \leq \frac{1}{2}\|\beta^*\|_2^2$$

for all $h \in [0,1]$. Moreover, for any support vector x_i, we have $y_i(x_i^\top\beta^* + \beta_0^*) \geq 1$ and $y_i(x_i^\top\beta^\dagger + \beta_0^\dagger) \geq 1$, and thus $y_i(x_i^\top\beta^{(h)} + \beta_0^{(h)}) \geq 1$. For any non-support vector x_i, we have $y_i(x_i^\top\beta^* + \beta_0^*) > 1$, thus, $y_i(x_i^\top\beta^{(h)} + \beta_0^{(h)}) \geq 1$ for h sufficiently close to 0. Therefore, for sufficiently small $h \in (0,1)$, we have shown that $(\beta^{(h)}, \beta_0^{(h)})$ also satisfies the constraints in (1) and, by uniqueness of the optimiser, it must be that $\beta^{(h)} = \beta^*$ and $\beta_0^{(h)} = \beta_0^*$ for any $h \in (0,1)$ small enough. The conclusion follows.

(iv) Since $(\beta^*, \beta_0^*) = (\beta^\dagger, \beta_0^\dagger)$, both the decision boundary and the set of support vectors remain the same if we remove all non-support vectors from the training set. If $SVs = \{1,2\}$, then y_1 and y_2 are necessarily distinct and the decision boundary, which separates x_1 and x_2 and maximises the distance to the closer of the two points, must be the perpendicular bisector of x_1 and x_2. Similarly, for $SVs = \{1,2,3\}$, we may assume that $y_1 = y_2 \neq y_3$. Then the decision boundary must be parallel to the line connecting x_1 and x_2. Thus, the decision boundary is the perpendicular bisector of the segment connecting x_3 to the projection image of x_3 to the line through x_1 and x_2.

4. (i) Let $\mathcal{X}$ be an arbitrary set. What is a *positive definite kernel* on $\mathcal{X}$?

(ii) Let K_1 and K_2 are two positive definite kernels on $\mathcal{X}$ and $g : \mathcal{X} \to \mathbb{R}$ any real-valued function. Define $K_{\text{sum}}(x,x') = K_1(x,x') + K_2(x,x')$, $K_{\text{prod}}(x,x') = K_1(x,x')K_2(x,x')$ and $K_{\text{conj}}(x,x') = g(x)K_1(x,x')g(x')$. Show that K_{sum}, K_{prod} and K_{conj} are also positive definite kernels.

(iii) Let $\mathcal{X} = \mathbb{R}^p, p \geq 1$. Using Part (ii) or otherwise, show that $K(x,x') = (c + x^\top x')^d$ and $K(x,x') = \exp\{-\gamma\|x - x'\|_2^2\}$ are both positive definite kernels for any $c > 0$, $d \in \mathbb{N}$ and $\gamma > 0$.

Solution

(i) A positive definite kernel on $\mathcal{X}$ is a real-valued symmetric function on $\mathcal{X}$, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, satisfying that for any $n \in \mathbb{N}$, $x_1, \ldots, x_n \in \mathcal{X}$, the matrix $(k(x_i, x_j))_{i,j=1,\ldots,n}$ is positive semidefinite.

(ii) Take any $n \in \mathbb{N}$ and $x_1, \ldots, x_n \in \mathcal{X}$. Let $v \in \mathbb{R}^n$ be an arbitrary vector. We have

$$v^\top(K_{\text{sum}}(x_i, x_j))_{i,j}v = v^\top(K_1(x_i, x_j))_{i,j}v + v^\top(K_2(x_i, x_j))_{i,j}v \geq 0$$

Let $u = (u_1, \dots, u_n)^\top$ be such that $u_j = v_j g(x_j)$. Then

$$v^\top (K_{\text{conj}}(x_i, x_j))_{i,j} v = u^\top (K_1(x_i, x_j))_{i,j} u \geq 0.$$

Thus, K_{sum} and K_{conj} are both positive definite kernels. To show K_{prod} is positive definite, it suffices to show that if $A, B \in \mathbb{R}^{n\times n}$ are symmetric and positive semidefinite, then their pointwise product $A \circ B$ (Hadamard product) is also positive semidefinite. By eigendecomposition, we may write $A = A_1 + \cdots + A_n$ and $B = B_1 + \cdots + B_n$ where each A_r and B_s are positive semidefinite matrices with rank at most 1. Since pointwise product satisfies distributive law, it suffices to prove $A_r \circ B_s$ is positive semidefinite for any r, s. If $A_r = aa^\top$ and $B_s = bb^\top$, then $A_r \circ B_s = cc^\top$, where $c = (c_1, \dots, c_n)^\top$ satisfies $c_i = a_i b_i$ for $i = 1, \dots, n$. This proves that K_{prod} is a positive definite kernel.

(iii) We first check that the linear kernel $(x, x') \mapsto x^\top x'$ is a positive definite kernel. This is true since $v^\top (x_i^\top x_j)_{i,j} v = \|Xv\|_2^2$, where X is a matrix with columns $x_1, \dots, x_n$. By (ii), positive definite kernels are closed under addition and multiplication, so $(x, x') \mapsto (c + x^\top x')^d$ is positive definite. If $K(x, x') = \exp\{-\gamma\|x - x'\|_2^2\}$, then for $g(x) = e^{-\gamma\|x\|^2}$, we have

$$K(x, x') = g(x) \exp\left\{2\gamma x^\top x'\right\} g(x').$$

Hence it suffices to show that $(x, x') \mapsto e^{2\gamma x^\top x'}$ is positive definite. This is a (convergent) infinite sum of multiples of powers of $x^\top x'$, and by part (ii) it is positive semidefinite.

5. Question 3 of the 2017–2018 past paper. You may find it in the following link: https://www.maths.cam.ac.uk/postgrad/part-iii/files/pastpapers/2018/paper_218.pdf.

Solution

(a) Let $Y_i = 1$ if the ith client has defaulted and 0 otherwise. Let $x_{i,1}$, $x_{i,2}$ and $x_{i,3}$ be the income (in 1000s), age (in years) and loan amount (in 1000s) of the ith client. The model fitted in `model1` has the algebraic form

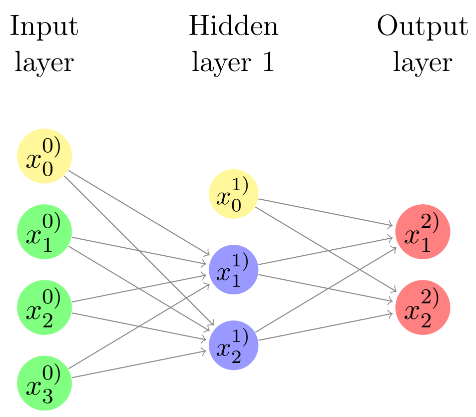
$$Y_i \overset{\text{ind}}{\sim} \text{Bernoulli}(p_i), \qquad \log\left(\frac{p_i}{1 - p_i}\right) = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,3}.$$

The estimated coefficients are $\hat\beta_0 = -1.32$, $\hat\beta_1 = -0.00592$, $\hat\beta_2 = -0.00858$ and $\hat\beta_3 = 0.0130$. By large sample asymptotics or asymptotic normality of MLE, $\hat\beta_2$ has asymptotic normal distribution. Thus, an approximate 95% confidence interval is $[-0.00858 - 0.00383 \times 2, -0.00858 + 0.00383 \times 2] = [-0.0162, -0.00092]$.

(b) The decision boundary is

$$\left\{x \in \mathbb{R}^3 : -1.32 - 0.00592x_1 - 0.00858x_2 + 0.0130x_3 = \log\left(\frac{p}{1-p}\right)\right\}.$$

(c) Sketch of the network architecture:



From now on in this exercise we use alternative notation to the one used in the course that, for working with sufficiently small networks, may be more convenient to use: we number the non-bias nodes and their associated linear combinations globally and without reference to their layer, so $x_j := x_j^{0)}, j = 1, 2, 3, x_j := x_{j-3}^{1)}, j = 4, 5, x_j := x_{j-5}^{2)}, j = 6, 7$, and $s_1, \dots, s_7$ are defined analogously; the bias nodes are all called x_0 and identified with the value 1; and, $\beta_{j,k}$ is the parameter corresponding to edge $j \to k$. Then, algebraically, the model fitted in `model2` is

$$Y_i \overset{\text{ind}}{\sim} \text{Bern}(p_i), \quad p_i = p(x_i) = \frac{e^{s_7}(x_i)}{e^{s_6}(x_i) + e^{s_7}(x_i)},$$

where $s_6(x_i)$ and $s_7(x_i)$ are recursively defined via (we view x_4, x_5, s_6, s_7 as functions of $x_{i,1}$, $x_{i,2}$, $x_{i,3}$ and $x_{i,4}$ and drop the dependence on i for simplicity)

$$\begin{aligned}
s_6 &= \beta_{4,6}x_4 + \beta_{5,6}x_5 + \beta_{0,6}x_0 \\
s_7 &= \beta_{4,7}x_4 + \beta_{5,7}x_5 + \beta_{0,7}x_0 \\
x_4 &= g(\beta_{1,4}x_{i,1} + \beta_{2,4}x_{i,2} + \beta_{3,4}x_{i,3} + \beta_{0,4}x_0) \\
x_5 &= g(\beta_{1,5}x_{i,1} + \beta_{2,5}x_{i,2} + \beta_{3,5}x_{i,3} + \beta_{0,5}x_0),
\end{aligned}$$

where $g(s) = \frac{e^s}{e^s+1}$ is the sigmoid (expit) function. Let $S = \{0, 1, 2, 3\} \times \{4, 5\} \cup \{0, 4, 5\} \times \{6, 7\}$. Then $\beta = (\beta_{u,v} : (u, v) \in S)$ are unknown parameters.

(d) The log-likelihood function is $\ell(\beta) = \sum_{i=1}^n \big\{Y_i \log p_i + (1 - Y_i)\log(1 - p_i)\big\}$, where $p_i := p(x_i)$ is computed as a function of β and $x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}$ as in (c).

Stochastic gradient ascent can be carried out as follows:

- initialise weights $\hat\beta^{(0,n)} = (\hat\beta_{u,v}^{(0,n)} : (u, v) \in S)$;
- for each training epoch $e = 1, 2, \dots, E$, set $\hat\beta^{(e,0)} := \hat\beta^{(e-1,n)}$, randomly shuffle the indices $\{1, \dots, n\}$ and, for each of the new indices $i = 1, \dots, n$, (dropping the superscript (e, i) from the appropriate xs, ss, βs and p)

– compute $p = p^{(e,i)}$ by a forward pass through the network as

$$\begin{aligned}
&\text{input layer: } x_v \leftarrow x_{i,v} \quad \text{for } v \in \{1,2,3\}\\
&\text{hidden layer: } s_v \leftarrow \sum_{u\in\{1,2,3,0\}} \beta_{u,v} x_u, \quad x_v \leftarrow f(s_v) \quad \text{for } v \in \{4,5\};\\
&\text{output layer: } s_v \leftarrow \sum_{u\in\{4,5,0\}} \beta_{u,v} x_u, \quad x_v \leftarrow f(s_v) \quad \text{for } v \in \{6,7\};\\
&\text{modelled probability: } p = \frac{e^{s_7}}{e^{s_6}+e^{s_7}};
\end{aligned}$$

– compute the partial derivative of $\ell_i = Y_i \log p_i + (1-Y_i)\log(1-p_i)$ with respect to parameters by backpropagation

$$\begin{aligned}
&\text{modelled probability: } \frac{\partial \ell_i}{\partial p_i} = \frac{Y_i}{p_i}\\
&\text{output layer: } \frac{\partial \ell_i}{\partial s_6} = \frac{Y_i}{p_i}\frac{\partial p_i}{\partial s_6} = -\frac{Y_i}{p_i}\frac{e^{s_6+s_7}}{(e^{s_6}+e^{s_7})^2} \text{ and } \frac{\partial \ell_i}{\partial s_7} = \frac{Y_i}{p_i}\frac{e^{s_6+s_7}}{(e^{s_6}+e^{s_7})^2}\\
&\qquad \frac{\partial \ell_i}{\partial \beta_{uv}} = \frac{\partial \ell_i}{\partial s_v} x_u \quad \text{for } (u,v) \in \{4,5,0\} \times \{6,7\}\\
&\text{hidden layer: } \frac{\partial \ell_i}{\partial x_u} = \sum_{v\in\{6,7\}} \frac{\partial \ell_i}{\partial s_v}\beta_{u,v} \text{ and } \frac{\partial \ell_i}{\partial s_u} = \frac{\partial \ell_i}{\partial x_u} g'(s_u) \quad \text{for } u \in \{4,5\}\\
&\qquad \frac{\partial \ell_i}{\partial \beta_{uv}} = \frac{\partial \ell_i}{\partial s_v} x_u \quad \text{for } (u,v) \in \{1,2,3,0\} \times \{4,5\}
\end{aligned}$$

– for some learning rate α_e, update weights

$$\hat{\beta}^{(e,i)}_{u,v} \leftarrow \hat{\beta}^{(e,i-1)}_{u,v} + \alpha_e \cdot \frac{\partial \ell_i}{\partial \beta_{u,v}} \Big|_{\beta_{u,v}=\hat{\beta}^{(e,i-1)}_{u,v}} .$$

6. An advertiser wanted to understand how different web design decisions influence the effectiveness of an online advertisement. They showed the advertisement to all users visiting the website while varying the font typeface (cagegorical variable `font`, with two levels `sanserif` and `serif`), display style (variable `display`, with two levels `banner` and `popup`) and seriousness of writing (`writing`, numerical variable taking value between 1 and 10). For each user, the advertiser recorded whether the advertisement was clicked.

```
head(ad)
#   click     font display writing
# 1   yes    serif  banner       3
# 2    no sanserif  banner       8
# 3    no sanserif  banner       1
# 4    no    serif   popup       7
# 5    no    serif   popup       2
# 6    no    serif   popup       9

x <- model.matrix(~font*display, data=ad)[, -1]
y <- model.matrix(~click-1, data=ad)
layer_relu <- layer_dense(units = 2, activation = 'relu', input_shape = dim(x)[2])
layer_softmax <- layer_dense(units = 2, activation = 'softmax')
ad.nnet <- keras_model_sequential(list(layer_relu, layer_softmax))
compile(ad.nnet, optimizer='sgd', loss='categorical_crossentropy', metrics='acc')
fit(ad.nnet, x, y, batch_size=1, epochs=5)
```

(i) Sketch a diagram of the neural network. Write down algebraically the neural network model fitted in `ad.nnet`, associating `sanserif` and `serif` to 0 and 1, `banner` and `popup` to 0 and 1, and `no` and `yes` to 0 and 1. Let β be the vector of all coefficients in model `ad.nnet`. Show that the maximum likelihood estimator for β is not unique if not restrictions are imposed on it.

(ii) Assume that all weights in the neural network are initialised to be equal to 1 and that we train the network by maximising the log-likelihood $\ell(\beta)$ via (vanilla) stochastic gradient descent with constant learning rate $\alpha = 0.1$, where α is assumed to have absorbed the sample size. What are the values of the weights after one training step in the first epoch if the data are not randomly shuffled?

(iii) If we maximise instead the regularised log-likelihood

$$\ell(\beta) - \frac{\lambda}{2}\|\beta\|_2^2$$

for some $\lambda > 0$, how would your stochastic gradient step in part (ii) change?

Solution

(i) The sketch is as in the figure above.
Let (x_i, y_i) be the ith observation, where entries of $x_i = (x_{i1}, x_{i2}, x_{i3})$ are coded by $x_{i1} = 0$ if font for ith visitor is sanserif and 1 otherwise, $x_{i2} = 0$ if display is banner and 1 otherwise, $x_{i3} \in \{1, \dots, 10\}$ quantifies the seriousness of writing for the ith visitor, and $y_i = 1$ if the ith visitor clicked on the advert and $y_i = 0$ if not. Then algebraically, the neural network model fitted is

$$y_i \overset{\text{ind}}{\sim} \text{Bern}(p_i), \quad p_i = p(x_i) = x_1^{2)}(x_i) = \frac{e^{s_1^{2)}}(x_i)}{e^{s_1^{2)}(x_i)} + e^{s_2^{2)}}(x_i)},$$

(note that we can choose to model p_i by $x_1^{2)}$ or $x_2^{2)}$ without this changing the trained classifier) where $s_1^{2)}(x_i)$ and $s_2^{2)}(x_i)$ (we drop the dependence on i for simplicity) are recursively defined via

$$\begin{aligned}
&x_1^{0)} = x_{i1}, \quad x_2^{0)} = x_{i2}, \quad x_3^{0)} = x_{i1}x_{i2}\\
&s_1^{1)} = \beta_{1,1}^{1)}x_1^{0)} + \beta_{1,2}^{1)}x_2^{0)} + \beta_{1,3}^{1)}x_3^{0)} + \beta_{1,0}^{1)}, \quad s_2^{1)} = \beta_{2,1}^{1)}x_1^{0)} + \beta_{2,2}^{1)}x_2^{0)} + \beta_{2,3}^{1)}x_3^{0)} + \beta_{2,0}^{1)}\\
&x_1^{1)} = \max\{s_1^{1)}, 0\}, \quad x_2^{1)} = \max\{s_2^{1)}, 0\}\\
&s_1^{2)} = \beta_{1,1}^{2)}x_1^{1)} + \beta_{1,2}^{2)}x_2^{1)} + \beta_{1,0}^{2)}, \quad s_2^{2)} = \beta_{2,1}^{2)}x_1^{1)} + \beta_{2,2}^{2)}x_2^{1)} + \beta_{2,0}^{2)}\\
&x_1^{2)} = \frac{e^{s_1^{2)}}}{e^{s_1^{2)}} + e^{s_2^{2)}}}, \quad x_2^{2)} = \frac{e^{s_2^{2)}}}{e^{s_1^{2)}} + e^{s_2^{2)}}}.
\end{aligned}$$

To show that the MLE for β is not unique unless further restrictions are imposed, note that if we replace $\beta_{1,0}^{2)}$ and $\beta_{2,0}^{2)}$ by $\beta_{1,0}^{2)} + c$ and $\beta_{2,0}^{2)} + c$ respectively for any $c \in \mathbb{R}$, the probability output p_i is unchanged and hence the log-likelihood is unchanged (in particular, the model is not identifiable).

(ii) From the table, we have $x_1 = (1, 0, 3)$ and $y_1 = 1$. Using that all entries of β equal 1, in the forward pass, we have

$$\begin{aligned}
&x_1^{0)} = 1, \quad x_2^{0)} = 0, \quad x_3^{0)} = 0\\
&s_1^{1)} = 2, \quad s_2^{1)} = 2, \quad x_1^{1)} = 2, \quad x_2^{1)} = 2\\
&s_1^{2)} = 5, \quad s_2^{2)} = 5, \quad x_1^{2)} = 1/2, \quad x_2^{2)} = 1/2.
\end{aligned}$$

The log-likelihood satisfies

$$\ell(\beta) = \sum_{i=1}^{n} y_i \log p_i + (1 - y_i)\log(1 - p_i) =: \frac{1}{n}\sum_{i=1}^{n} \ell_i(\beta).$$

Recalling $p_1 = x_1^{2)}$, the contribution of the first observation to the log-likelihood is

$$\ell_1 = y_1 \log p_1 + (1 - y_1)\log(1 - p_1).$$

Thus, partial derivatives of ℓ_1 with respect to all weights can be computed via backpropagation as follows.

– output layer:

$$\frac{\partial \ell_1}{\partial x_1^{2)}} = \frac{1}{x_1^{2)}} = 2$$

$$\frac{\partial \ell_1}{\partial s_1^{2)}} = \frac{\partial \ell_1}{\partial x_1^{2)}} \frac{e^{s_1^{2)}+s_2^{2)}}}{(e^{s_1^{2)}} + e^{s_2^{2)}})^2} = \frac{1}{2}, \quad \frac{\partial \ell_1}{\partial s_2^{2)}} = \frac{\partial \ell_1}{\partial x_1^{2)}} \frac{-e^{s_1^{2)}+s_2^{2)}}}{(e^{s_1^{2)}} + e^{s_2^{2)}})^2} = -\frac{1}{2}.$$

$$\frac{\partial \ell_1}{\partial \beta_{1,1}^{2)}} = -\frac{\partial \ell_1}{\partial \beta_{2,1}^{2)}} = \frac{\partial \ell_1}{\partial \beta_{1,2}^{2)}} = -\frac{\partial \ell_1}{\partial \beta_{2,2}^{2)}} = \frac{1}{2} \cdot 2 = 1, \quad \frac{\partial \ell_1}{\partial \beta_{1,0}^{2)}} = -\frac{\partial \ell_1}{\partial \beta_{2,0}^{2)}} = \frac{1}{2} \cdot 1 = \frac{1}{2}.$$

– hidden layer:

$$\frac{\partial \ell_1}{\partial x_1^{1)}} = \frac{\partial \ell_1}{\partial s_1^{2)}} \beta_{1,1}^{2)} + \frac{\partial \ell_1}{\partial s_2^{2)}} \beta_{2,1}^{2)} = 0, \quad \frac{\partial \ell_1}{\partial x_2^{1)}} = \frac{\partial \ell_1}{\partial s_1^{2)}} \beta_{1,2}^{2)} + \frac{\partial \ell_1}{\partial s_2^{2)}} \beta_{2,2}^{2)} = 0.$$

$$\frac{\partial \ell_1}{\partial s_1^{1)}} = \frac{\partial \ell_1}{\partial x_1^{1)}} \mathbf{1}\{s_1^{1)} > 0\} = 0, \quad \frac{\partial \ell_1}{\partial s_2^{1)}} = \frac{\partial \ell_1}{\partial x_2^{1)}} \mathbf{1}\{s_2^{1)} > 0\} = 0.$$

$$\frac{\partial \ell_1}{\partial \beta_{j,k}^{1)}} = 0 \text{ for all } j, k.$$

Thus, after one training step and due to α having absorbed n, the weights are updated as

$$\beta_{j,k}^{h)} = \begin{cases} 1 + 0.1 \times 0 = 1 & \text{if } h = 1 \\ 1 + 0.1 \times 1 = 1.1 & \text{if } k > 0 \text{ and } h = 2 \\ 1 + 0.1 \times 0.5 = 1.05 & \text{if } k = 0 \text{ and } h = 2. \end{cases}$$

(Remark: this example shows how symmetry-breaking is needed for more than just all zero initialising weights.)

(iii) We can write

$$\ell(\beta) - \frac{\lambda}{2}\|\beta\|_2^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\ell_i(\beta) - \frac{\lambda}{2}\|\beta\|_2^2\right).$$

Differentiating we see that all we need to do is to subtract $\frac{\lambda}{n}\beta_{j,k}^{h)}$ to $\frac{\partial \ell_1}{\partial \beta_{j,k}^{h)}}$ computed in part (ii). (Remark: this is known as weight decay in the neural networks literature.)

7. *(Exercise with* `R`*)* Download the `letter` dataset from the course website:

`http://www.statslab.cam.ac.uk/~tw389/teaching/SLP18/data`

Each row of the data contains 16 different attributes of a pixel image of one of the 26 capital letters in the English alphabet, together with the letter label itself. More information about this dataset can be found at `https://archive.ics.uci.edu/ml/datasets/Letter+Recognition`. Construct a classifier using one of the methods covered in this course. Then test your classifier on the test dataset `letter_test` from the course website. What is your test error? (Note that all design decision of your classifier must be based on the training data.)