4F10 Deep Learning and Structured Data, 2018

1. *Neural Network Training*

(a) The number of parameters is

$$N = d \times N^{(1)} + K \times N^{(L)} + \sum_{k=1}^{L-1} N^{(k)} \times N^{(k+1)}$$

The system needs to generalise to unseen examples, so the total number of parameters needs to be matched to the quantity of training data (in conjunction with regularisation approaches). [15%]

(b)(i) The cross-entropy training criterion has the form

$$\mathcal{E} = \sum_{p=1}^{n}\sum_{i=1}^{K} t_{pi} \log(y_i(\boldsymbol{x}_p))$$

where

- t_{pi} is the 1-of-K-coding of the value y_p
- $y_i(\boldsymbol{x}_p)$ is the output of element i of the output layer

[15%]

(b)(ii) The form of the soft-max activation is

$$\boldsymbol{\phi}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)}$$

As z_i is real (in these networks) this ensures that the output layer is positive and satisfies the sum-to-one constraint. [10%]

(b)(iii) Need to find

$$\frac{\partial \boldsymbol{x}^{(l+1)}}{\partial \boldsymbol{x}^{(l)}} = \frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{x}^{(l)}} \frac{\partial \boldsymbol{x}^{(l+1)}}{\partial \boldsymbol{z}^{(l)}}$$

These parts can then be treated separately. Only need to consider diagonal elements for the second term

$$\frac{\partial x_i^{(l+1)}}{\partial z_i^{(l)}} = \phi(z_i)(1 - \phi(z_i))$$

from the first examples paper. The other term is simply a linear relationship

$$\frac{\partial \boldsymbol{z}^{(l)}}{\partial \boldsymbol{x}^{(l)}} = \boldsymbol{W}^{(l)}$$

Thus the overall expression is

$$\frac{\partial \boldsymbol{x}^{(l+1)}}{\partial \boldsymbol{x}^{(l)}} = \boldsymbol{W}^{(l)\mathsf{T}} \begin{bmatrix} \phi(z_1)(1-\phi(z_1)) & 0 & \dots & 0 \\ 0 & \phi(z_2)(1-\phi(z_2)) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \phi(z_{N_l})(1-\phi(z_{N_l})) \end{bmatrix}$$

[25%]

(c)(i) For the configuration described

$$\frac{\partial \boldsymbol{x}^{(l+1)}}{\partial \boldsymbol{x}^{(l)}} = \boldsymbol{W}^{\mathsf{T}}$$

Thus the complete derivative form is

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{x}^{(l)}} = \left(\prod_{j=l}^{L} \frac{\partial \boldsymbol{x}^{(l+1)}}{\partial \boldsymbol{x}^{(l)}}\right) \frac{\partial \mathcal{E}}{\partial \boldsymbol{x}^{(L+1)}} = \left(\prod_{j=l}^{L} \boldsymbol{W}^{\mathsf{T}}\right) \frac{\partial \mathcal{E}}{\partial \boldsymbol{x}^{(L+1)}} = \left(\boldsymbol{W}^{\mathsf{T}}\right)^{L-l+1} \frac{\partial \mathcal{E}}{\partial \boldsymbol{x}^{(L+1)}}$$

Then doing eigenvector/eigenvalue decomposition on $\boldsymbol{W}^{\mathsf{T}} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}$

$$\left(\boldsymbol{W}^{\mathsf{T}}\right)^{L-l+1} = \left(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}\right)^{L-l+1} = \mathbf{U}\left(\boldsymbol{\Lambda}\right)^{L-l+1}\mathbf{U}^{-1} \approx \lambda_1^{L-l+1}\mathbf{u}_1\left[\mathbf{U}^{-1}\right]_1$$

if $L >> l$ as the expression is dominated by the largest eigenvalue (λ_1) and associated eigenvector. [25%]

(c)(ii) If the magnitude of the largest eigenvalue of $\boldsymbol{W}$ is greater than 1 then the gradient will "explode". Conversely if the value is smaller than 1 then the gradient will "vanish". This impacts the training of very deep networks and recurrent networks. [10%]

Comment: This question examined Deep Neural Network configurations and training. This was the least popular of the questions. The first two parts of the questions were reasonably well answered with students demonstrating knowledge of the cross-entropy training criterion, and gradients. Answers to part (c) were generally disappointing with students not being able to derive the approximation given in the question.

2. *Bayes Decision Rule and Linear Classifier*

(a) Bayes decision rule states that you should pick the class with the greatest posterior

$$\frac{P(\omega_1|x)}{P(\omega_2|x)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} 1$$

[10%]

(b) As the number of samples is very large it is possible to express the training criterion in terms of expected values over the class distributions. The least squares criterion that must be minimised is

$$E(a) = \frac{1}{2}\int (ax)^2 p(x|\omega_1)dx + \frac{1}{2}\int (ax-1)^2 p(x|\omega_2)dx$$

Differentiate this with respect to the a yields

$$\begin{aligned}\frac{\partial}{\partial a}E(a) &= \int ax^2 p(x|\omega_1)dx + \int (ax-1)xp(x|\omega_2)dx \\ &= a\left(\sigma_1^2+\mu_1^2\right) + a\left(\sigma_2^2+\mu_2^2\right) - \mu_2\end{aligned}$$

The statistics for each of these is given in the question, so we get

$$2a + 8a - 2 = 0$$

Hence the value for a is 0.2 [30%]

(c) The threshold is set at 0.5. The value of x that will correspond to this is $x_T = 2.5$. The probability of error is then given by

$$\begin{aligned}P_{\mathsf{e}} &= \frac{1}{2}\int_{x_T}^{\infty}\mathcal{N}(z;0,2)dz + \frac{1}{2}\int_{-\infty}^{x_T}\mathcal{N}(z;2,4)dz \\ &= \frac{1}{2}\Big((1-F(2.5/\sqrt{2})) + F(0.5/2)\Big)\end{aligned}$$

[25%]

(d)(i) Assuming that majority voting is being used, then the probability of error occurs when two or three of the classifiers make an error. This can be expressed as

$$P_{\mathsf{e}}P_{\mathsf{e}}^{(2)}(1-P_{\mathsf{e}}^{(3)}) + P_{\mathsf{e}}P_{\mathsf{e}}^{(3)}(1-P_{\mathsf{e}}^{(2)}) + P_{\mathsf{e}}^{(2)}P_{\mathsf{e}}^{(3)}(1-P_{\mathsf{e}}) + P_{\mathsf{e}}P_{\mathsf{e}}^{(2)}P_{\mathsf{e}}^{(3)}$$

[15%]

(d)(ii) This probability of error will only occur when the errors made by each of the individual classifiers are independent of each other. As the dimensions are Gaussian distributed and uncorrelated this can be achieved. [10%]

(d)(iii) (d)(ii) is not the lowest probability of error as each of the classifiers is not optimal. To generate the optimal classifier a generative classifier using a multivariate Gaussian distributions for each of the classes. [10%]

Comment: This questions examined the training of a simple (one-dimensional) classifier, and then combining an ensemble of classifiers together. Most students could derive an expression for the value of the classifier parameters, but could not connect this to the distribution of classes given at the start of the question. Part (c) was generally well answered (and expressions based on a accepted). The final part was disappointing as many students did not realise that a maximum voting approach could be used.

3. *Support Vector Machines and Kernels*

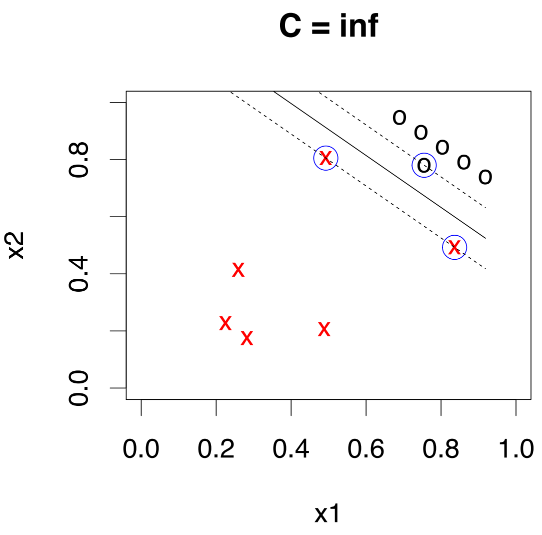
a) Kernels are functions $k(\mathbf{x}_n, \mathbf{x}_m)$ that compute dot products between feature vectors $\boldsymbol{\phi}(\mathbf{x}_n)$ and $\boldsymbol{\phi}(\mathbf{x}_m)$, that is, $k(\mathbf{x}_n, \mathbf{x}_m) = \boldsymbol{\phi}(\mathbf{x}_n)^\mathrm{T}\boldsymbol{\phi}(\mathbf{x}_m)$, where $\boldsymbol{\phi}(\cdot)$ maps its inputs to a typically higher dimensional space. The main advantage of kernels is that they compute dot products without directly computing $\boldsymbol{\phi}(\cdot)$, allowing to work in very high-dimensional spaces. They also allow us to obtain a non-linear version of any linear machine learning algorithm that can be written in terms of dot products on data points. Slack variables are used to allow for constraint violation in the SVM classifier, enabling the classifier to make mistakes on the training data. [15%]

b)i) The constraints are $t_n(\mathbf{w}^\mathrm{T}\mathbf{x}_n + b) \geq 1 - \xi_n$, $\xi_n \geq 0$ for $n = 1, \ldots, N$. [10%]

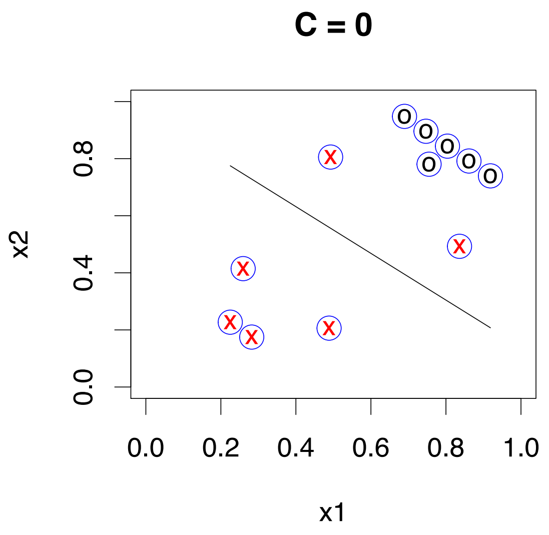
b)ii) The value of ξ_n is zero in this case since the value of ξ_n is minimized subject to the constraint $t_n(\mathbf{w}_\star^\mathrm{T}\mathbf{x}_n + b_\star) + \xi_n \geq 1$. In this case $t_n(\mathbf{w}_\star^\mathrm{T}\mathbf{x}_n + b_\star) \geq 1$ so ξ_n is mimized to take value zero. [10%]

b)iii)

(a) The value of ξ_n is in this case $\xi_n = 1 - t_n(\mathbf{w}_\star^\mathrm{T}\mathbf{x}_n + b_\star)$ since ξ_n is being minimized but it must also guarantee that the constraint $t_n(\mathbf{w}_\star^\mathrm{T}\mathbf{x}_n + b_\star) + \xi_n \geq 1$. Therefore, the constraint is tight now and $\xi_n = 1 - t_n(\mathbf{w}_\star^\mathrm{T}\mathbf{x}_n + b_\star)$. [10%]



c)i) [10%]



c)ii) [10%]

c)iii) We expect $C \approx 0$ to be useful when the top two points with class X are outliers and $C = \infty$ to be better when there are no outliers. [10%]

d)i) The new objective is

$$\frac{1}{2}||\mathbf{w}||^2 + C \sum_{n=1}^{N} \ell([\mathbf{w}^{\mathrm{T}}\mathbf{x} + b]t_n)\,.$$

[15%]

d)ii) The function $l(x)$ is not differentiable at 0 and its gradient is flat for $x > 0$ so we will not be able to use gradient descent. [10%]

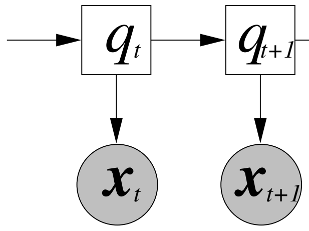
Comment: This questions examined the students' knowledge of Support Vector Machines (SVMs), and the training criterion that can be used. This was the most popular question. The students showed a good understanding of SVMs, though many answers had minor mistakes.

4. *Sequence Models and EM*

(a) The two conditional independence assumptions for HMMs are:

- states are conditionally independent given the current state;
- observations are conditionally independent given the state that generated it.

The graphical model for this task is



[15%]

(b) The log-likelihood can be written as

$$\log(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T) = \log\left(\sum_{\boldsymbol{q}\in\boldsymbol{Q}} P(\boldsymbol{q})p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T|\boldsymbol{q})\right) = \log\left(\sum_{\boldsymbol{q}\in\boldsymbol{Q}} a_{q_T\mathtt{s}_N}\prod_{t=1}^{T} a_{q_{t-1}q_t}p(\boldsymbol{x}_t|q_t)\right)$$

[15%]

(c)(i) The terms in the auxiliary function are

- $P(q_t = \mathtt{s}_j|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T;\boldsymbol{\theta})$ is the state posterior given the current model parameters are observation sequence
- $p(\boldsymbol{x}_t|\mathtt{s}_j,\hat{\boldsymbol{\theta}})$ is the probability of the observation given the "new" model parameters and state $\mathtt{s}_j$.

This expression can be used to iteratively estimate the model parameters.

0 Initialise model parameters

1 Obtain state posteriors given current parameters

2 Maximise auxiliary function

3 Goto (1) until converged

[15%]

(c)(ii) From the question

$$\begin{aligned}\alpha_j(t) &= \log\left(p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_t,q_t=\mathtt{s}_j)\right)\\ \beta_j(t) &= \log\left(p(\boldsymbol{x}_{t+1},\ldots,\boldsymbol{x}_T|q_t=\mathtt{s}_j)\right)\end{aligned}$$

Adding this expressions together

$$\alpha_j(t)+\beta_j(t) = \log\left(p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T,q_t=\mathtt{s}_j)\right)$$

This can be used to obtain the "normalisation" term

$$Z = \log\left(\sum_{j=2}^{N-1} \exp(\alpha_j(t) + \beta_j(t))\right) = \log\left(p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)\right)$$

The expression required can then be written as

$$P(q_t = \mathtt{s}_j | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T; \boldsymbol{\theta}) = \exp\left(\alpha_j(t) + \beta_j(t) - Z\right) = \gamma_j(t)$$

[15%]

(c)(iii) The auxiliary function can be written as

$$\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K - \sum_{t=1}^{T} \sum_{j=2}^{N-2} \gamma_j(t) \sum_{k=1}^{d} \left[\frac{(x_{tk} - \hat{\mu}_{jk})^2}{2\hat{\sigma}_{jk}^2}\right]$$

Differentiating with respect to $\hat{\mu}_{ik}$ and equating to zero yields

$$\hat{\mu}_{ik} = \frac{\sum_{t=1}^{T} \gamma_i(t) x_{tk}}{\sum_{t=1}^{T} \gamma_i(t)}$$

[25%]

(d) The simplest configuration for the network is

- 1-of-K coding of the $N-2$ emitting states as the input
- output is a set of $N-2$ d-dimensional mean and log-variances (ensures variances positive)
- linear activation functions for the means and log-variances

The network topology for the hidden units will depend on the amount of training data. For this configuration the network will not be expected to perform any better than the standard Gaussian distribution. [15%]

Comment: This question examined hidden Markov models, and how EM can be used to train the model. It was disappointing that few students could correctly give the expression for the log-likelihood of the observation sequence for the HMM, but most students were able to derive the update formula for the mean.