Solutions to 4F10 Pattern Processing, 2016

1. *Bayes' Decision rule and generative models*

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

$$\frac{P(\omega_1|x)}{P(\omega_2|x)} \begin{array}{c} \omega_1 \\ > \\ < \\ \omega_2 \end{array} 1$$

   For a generative model this becomes

$$\frac{p(x|\omega_1)P(\omega_1)}{p(x|\omega_2)P(\omega_2)} \begin{array}{c} \omega_1 \\ > \\ < \\ \omega_2 \end{array} 1$$

   [10%]

   (b)(i) A point that lies on the decision boundary satisfies

$$\log(p(\boldsymbol{x};\boldsymbol{\mu}_1,\boldsymbol{\Sigma})) + \log(P(\omega_1)) = \log(p(\boldsymbol{x};\boldsymbol{\mu}_2,\boldsymbol{\Sigma})) + \log(P(\omega_2))$$

   where

$$\boldsymbol{\mu}_1 = \begin{bmatrix} \mu_{11} \\ \vdots \\ \mu_{d1} \end{bmatrix} \quad \boldsymbol{\mu}_2 = \begin{bmatrix} \mu_{12} \\ \vdots \\ \mu_{d2} \end{bmatrix} \quad \boldsymbol{\Sigma} = \text{diag}\left(\begin{bmatrix} \sigma^2 \\ \vdots \\ \sigma^2 \end{bmatrix} \right)$$

   Substituting in the expressions for the covariance and priors yields

$$2(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)'\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}_1'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 = 0$$

   [20%]

   (b)(ii) **Simplest solution is to draw a sketch indicates the distances in standard deviations. This is acceptable as a solution**.

   The decision boundary in (b) defines the two regions. For this form of distribution the posterior will only be a function of the perpendicular distance from the decision boundary measured in standard deviations. First the space is transformed so that all values are normalised by the standard deviation so

$$\mathbf{a} = \boldsymbol{\mu}_1/\sigma, \quad \mathbf{b} = \boldsymbol{\mu}_2/\sigma$$

   Projecting the distribution for class along this line

$$\mathcal{N}\left(x; (\mathbf{b}-\mathbf{a})'\,\mathbf{a}/K, 1\right)$$

   where $K^2 = ||\mathbf{b}-\mathbf{a}||^2$. Perpendicular to this direction will just integrate out to 1 for all positions.

   The decision boundary is the projection of the point half way between the means onto this line. This projection point is

$$a = (\mathbf{a}+\mathbf{b})'(\mathbf{b}-\mathbf{a})/2K$$

   Considering the first element of the probability of error

$$\int_{\mathcal{R}_2} p(\boldsymbol{x}|\omega_1)P(\omega_1)d\boldsymbol{x} = \frac{1}{2}\int_a^\infty \mathcal{N}\left(x; (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)'\,\boldsymbol{\mu}_1/K; 1\right) dx$$

Offsetting the mean of the integral to 0 yields the required form

$$\int_{\mathcal{R}_2} p(\boldsymbol{x}|\omega_1)P(\omega_1)d\boldsymbol{x} = \frac{1}{2}\int_a^\infty \mathcal{N}(x;0;1)\,d\boldsymbol{x}$$

where a is now

$$\begin{aligned} a &= \frac{1}{2K}(\mathbf{b}-\mathbf{a})'(\mathbf{a}+\mathbf{b}) - \frac{1}{K}(\mathbf{b}-\mathbf{a})'\mathbf{a} \\ &= \frac{1}{2K}(\mathbf{a}-\mathbf{b})'(\mathbf{a}-\mathbf{b}) \\ &= \frac{1}{2}\sqrt{(\mathbf{a}-\mathbf{b})'(\mathbf{a}-\mathbf{b})} \end{aligned}$$

Since the probability of error for the second expression will be the same this is the value of c. By symmetry the required answer is $-a$. So the final solution is

$$c = -\frac{1}{\sigma}\sqrt{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)'(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)}$$

[30%]

(c) (i) To get the probability of error using $2d$ sensors requires computing a new value for c, c_2 (from above). The reduction in the probability of error is

$$\Delta P_{\mathsf{e}} = \int_{-\infty}^{c}\mathcal{N}(z;0,1)dz - \int_{-\infty}^{c_2}\mathcal{N}(z;0,1)dz$$

Here

$$c_2 = -\frac{1}{\sigma}\sqrt{(\boldsymbol{\mu}_3-\boldsymbol{\mu}_4)'(\boldsymbol{\mu}_3-\boldsymbol{\mu}_4)}$$

where

$$\boldsymbol{\mu}_3 = \begin{bmatrix} \mu_{11} \\ \vdots \\ \mu_{1(2d)} \end{bmatrix} \quad \boldsymbol{\mu}_4 = \begin{bmatrix} \mu_{21} \\ \vdots \\ \mu_{2(2d)} \end{bmatrix}$$

The distance can only increase, so the probability of error can only go down, or remain the same. [15%]

(c)(ii) To get the optimal subset, need to find the subset of sensors that has the greatest distance between the means. So simply rank order the sensors according to $\mu_{s1} - \mu_{s2}$ and select the top k. [15%]

(c)(iii) If the number of sensors increases so the number of model parameters to train will increase. This will result in generalisation problems. The performance on the training data will improve (consistent with the analysis). However, the performance on the held-out test data, which is what is of interest for practical systems, will not necessarily continue to improve which means that rather than the performance. There is also an increase in the computational cost as the number of sensors increases. [10%]

Comments This question examined Bayes' decision rule and the associated probability of errors. A number of candidates got confused between scalar and vector values, occasionally swapping between the two with no comments.

2. *Gaussian Mixture Models*

(a) Log-likelihood of the training data is

$$\log(p(x_1,\ldots,x_N|\theta)) = \sum_{i=1}^{N} \log\left(\sum_{m=1}^{M} c_m \mathcal{N}(\mathbf{x}_i;\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)\right)$$

[10%]

(b)(i) Substituting in the expression for the likelihood to the auxiliary function

$$\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = \sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|x_i,\boldsymbol{\theta}) \log\left(\mathcal{N}(\mathbf{x}_i;\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)\right)$$

Differentiate this with respect to $\boldsymbol{\mu}_q$ gives

$$\frac{\partial \mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\mu}}_q} = \sum_{i=1}^{n} P(\omega_q|x_i,\boldsymbol{\theta})\left[\hat{\boldsymbol{\Sigma}}_q^{-1}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_q)\right]$$

Equating to zero (ans setting variable to m) gives

$$\hat{\boldsymbol{\mu}}_m = \frac{\sum_{i=1}^{N} P(\omega_m|\mathbf{x}_i,\boldsymbol{\theta})\mathbf{x}_i}{\sum_{i=1}^{N} P(\omega_m|\mathbf{x}_i,\boldsymbol{\theta})}$$

[25%]

(b)(ii) As the covariance matrix is diagonal, possible to write auxiliary function as

$$\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = \sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|\mathbf{x}_i,\boldsymbol{\theta}) \sum_{j=1}^{d}\left(-\log(\sqrt{2\pi}\hat{\sigma}) - \frac{1}{2}\frac{(x_{ij}-\hat{\mu}_{mj})^2}{\hat{\sigma}_j^2}\right)$$

Differentiating this wrt to $\hat{\sigma}_j$ yields

$$\sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|\mathbf{x}_i,\boldsymbol{\theta})\left(-\frac{1}{\hat{\sigma}_j} + \frac{(x_{ij}-\hat{\mu}_{mj})^2}{\hat{\sigma}_j^3}\right)$$

Equating to zero yields

$$\hat{\sigma}_j^2 = \frac{\sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|x_i,\boldsymbol{\theta})(x_{ij}-\hat{\mu}_{mj})^2}{\sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|x_i,\boldsymbol{\theta})}$$

[20%]

(c)(i) As the covariance matrix is diagonal, possible to write auxiliary function as

$$\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = \sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|\mathbf{x}_i,\boldsymbol{\theta}) \sum_{j=1}^{d}\left(-\log(\sqrt{2\pi\hat{a}_m}\hat{\sigma}_j) - \frac{1}{2}\frac{(x_{ij}-\hat{\mu}_j)^2}{\hat{a}_m\hat{\sigma}_j^2}\right)$$

Differentiating wrt to a_m yields

$$\sum_{i=1}^{n} P(\omega_m|\mathbf{x}_i,\boldsymbol{\theta}) \sum_{j=1}^{d}\left(-\frac{1}{2\hat{a}_m} + \frac{1}{2}\frac{(x_{ij}-\hat{\mu}_j)^2}{\hat{a}_m^2\hat{\sigma}_j^2}\right)$$

Equating to zero yields

$$\hat{a}_m = \frac{\sum_{i=1}^{n} P(\omega_m|\mathbf{x}_i,\boldsymbol{\theta})\sum_{j=1}^{d}(x_{ij}-\hat{\mu}_j)^2/\hat{\sigma}_j}{d\sum_{i=1}^{n} P(\omega_m|\mathbf{x}_i,\boldsymbol{\theta})}$$

Unfortunately $\hat{a}_m$ is a function of $\hat{\sigma}$. It is therefore necessary to interleave the updates - this is a GEM update (not discussed in lectures). [30%]

(c)(ii) Discussion should include:

- model in (b) allows multi-modal data to be modelled, (c) is unimodal;
- model in (b) allows non-symmetric data to be modelled, (c) is symmetric;
- (c) is a method to model symmetric data that is non-Gaussian efficiently;
- (c) has significantly fewer parameters (depending on d);
- computational contrast (possible to efficiently compute tied variance systems.

[15%]

Comments A question examining the attributes of Gaussian mixture models and parameter estimation. This was the most popular question on the exam paper. The candidates showed a good knowledge of estimating the mean parameters using Expectation-Maximisation. A number of candidates were unable to correctly estimate the variance parameters, often not taking advantage of the fact that the covariance matrix was diagonal.

3. *Parameter Optimisation*

(a)(i) The values are

$$\begin{aligned} \mathbf{b} &= \boldsymbol{\nabla} E(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}^{(\tau)}} \\ \mathbf{A} &= \boldsymbol{\nabla}^2 E(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}^{(\tau)}} \end{aligned}$$

the gradient and the Hessian respectively at the current model parameters. [15%]

(a)(ii) Letting

$$\boldsymbol{\Delta\theta}^{(\tau)} = (\boldsymbol{\theta} - \boldsymbol{\theta}^{(\tau)})$$

gives the following differential expression

$$\frac{\partial}{\partial \boldsymbol{\Delta\theta}^{(\tau)}} E(\boldsymbol{\theta}) = \mathbf{b} + \mathbf{A}\boldsymbol{\Delta\theta}^{(\tau)})$$

Equating this to zero gives

$$\boldsymbol{\Delta\theta}^{(\tau)} = -\mathbf{A}^{-1}\mathbf{b}$$

Thus the value is given by

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(\tau)} - \mathbf{A}^{-1}\mathbf{b}$$

[25%]

(b) Writing out the quadratic form for a single dimension yields (ignoring the subscript i)

$$E(\theta) \approx E(\theta^{(\tau)}) + \Delta\theta b + \Delta\theta^2 a/2$$

At the new update value require that

$$\begin{aligned} g^{(\tau+1)} &= 0 \\ g^{(\tau)} &= b \\ g^{(\tau-1)} &= b - \Delta\theta^{(\tau-1)} a \end{aligned}$$

From part $a(i)$ the update is given by $-b/a$. Solving these three equations yields the update rule (details of derivation not given). [35%]

(c) In (a) it is necessary to calculate of the Hessian. For large numbers of parameters this may not be practical (it's a square matrix). It may also not be of full rank hence there can be issues with the inversion. It can also head off towards a maximum as well. In contrast for (b) no need to invert compute Hessian. However each dimension is assumed independent in the Hessian (diagonal approximation). This may be poor.

There is also the need to consider the computational cost of accumulating the Hessian statistics and those of the gradients. Possible to relate optimisation to finite difference second-order approximation for the diagonal elements of the Hessian. [25%]

Comments This questions examined the students' knowledge of second order optimisation approaches. A well answered question with the candidates showing a good understanding of second-order approaches and approximation approaches.

4. *Generalisation for Neural Networks*

(a)(i) The output of the softmax can be interpreted as a probability. The training criterion described is the negative cross-entropy criterion where the network (with the 1-of-K coding) maximises the probability of the correct label. [15%]

(a)(ii) Find derivative (only final answer given here)

$$\frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \mathbf{w}_j} = \sum_{i=1}^{n} t_{ij} \left(\mathbf{x}_i - \frac{\exp(\mathbf{w}_j' \mathbf{x}_i)}{\sum_{k=1}^{K} \exp(\mathbf{w}_k' \mathbf{x}_i)} \mathbf{x}_i \right) - \sum_{k \neq j} \sum_{i=1}^{n} t_{ik} \left(\frac{\exp(\mathbf{w}_k' \mathbf{x}_i)}{\sum_{l=1}^{K} \exp(\mathbf{w}_l' \mathbf{x}_i)} \mathbf{x}_i \right)$$

This can be used within a standard gradient ascent process. Note the negative cross-entropy term needs to be maximised. Now

$$\mathbf{w}_j^{(n)} = \mathbf{w}_j^{(n-1)} + \eta \left. \frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \mathbf{w}_j} \right|_{\boldsymbol{\lambda}^{(n-1)}}$$

[30%]

(b)(i) The negative cross-entropy is being maximised. The regularisation term penalises large weight values, provided that a is positive, and attempts to ensure that the weights are small. This prevents extreme weight values, improving regularisation. As a increase so the system is encouraged to have smaller values for the weights. [15%]

(b)(ii) This expression is identical to (a)(ii), other than the additional term. Thus

$$\frac{\partial \mathcal{F}(\boldsymbol{\lambda})}{\partial \mathbf{w}_j} = \frac{\partial \mathcal{L}(\boldsymbol{\lambda})}{\partial \mathbf{w}_j} - 2a\mathbf{w}_j$$

[15%]

(c)(i) The number of parameters in the original system is $d \times K$. Introducing a intermediate linear layers results in $d \times b + b \times K$ parameters. The total number of model parameters will decrease if

$$b < \frac{d \times K}{d + K}$$

By decreasing the number of model parameters, generalisation may improve, though the power of the network may be decreased. [15%]

(c)(ii) Though this is a multi-layer perceptron, the linear activation function means that EBP is not required. As it is possible to write

$$\mathbf{W} = \mathbf{W}_1 \times \mathbf{W}_2$$

[10%]

Comments This question was based on neural network training and generalisation. This was the least popular question. The performance on this question was disappointing. Few candidates understood how an additional layer could reduce the number of model parameters.

5. *Support Vector Machines and Kernels*

(a) The training criterion for the perceptron algorithm is the sum of the perpendicular distance from the decision boundary of all misclassified points. There is no unique solution to this cost function. For the SVM classifier the cost function is to maximum the margin, i.e.

$$\max_{\mathbf{w},b} \min \{||\mathbf{x} - \mathbf{x}_i||; \langle \mathbf{w}, \mathbf{x} \rangle + b = 0, i = 1, \ldots, m\}$$

This has the dual form which required minimising

$$\min_{\mathbf{w},b} \tau(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$$

subject to

$$y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) \geq 1$$

for all $i = 1, \ldots, m$.

When the data is not linearly separable *slack variables* must be introduced. The soft-margin classifier is obtained by minimising

$$\tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{m} \xi_i$$

subject to

$$\begin{aligned} y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}$$

[25%]

(b) Kernels map from the input-space to a higher dimensional feature-space. A linear classifier is then built in this feature space. This results in a non-linear decision boundary in the original feature space. The general form for the polynomial kernel is

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i . \mathbf{x}_j + 1)^d$$

[15%]

(c)(i) For the in-homogeneous polynomial kernel the feature-space has the form (others are possible, but this is the most informative)

$$\boldsymbol{\Phi}(x) = \begin{bmatrix} a_0 & a_1 x & a_2 x^2 & \ldots & a_N x^N \end{bmatrix}'$$

where the values of $a_0, \ldots, a_N$ are determined by the order of the polynomial. There are thus two main differences

(a) the kernel operates on the exponential of the input-space

(b) there are no scaling factors - depending on the metric used this may or not affect the decision boundary generated.

[20%]

(c)(ii) The kernel function is found from noting that the dot product is a geometric progression

$$\begin{aligned} k(x_i, x_j) &= 1 + \sum_{r=1}^{N} \exp(r(x_i + x_j)) \\ &= \frac{1 - e^{(N+1)(x_i + x_j)}}{1 - e^{(x_i + x_j)}} \end{aligned}$$

[25%]

(d) The kernelised version of the rule is

$$g_i(x) = \sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b$$

Once the number of support vectors are known then the computational cost is independent of the number of training samples and N. The cost scales linearly with the number of support vectors. It is also necessary to consider the cost of the kernel calculation. This scales linearly with the dimension of the observations. [15%]

Comments This question examined the candidates knowledge of kernels and the resulting feature space. A well answered question. The main mistakes made were comparing inhomogeneous polynomial kernels and the form given in the question.