PART IIB 2011 4F10: STATISTICAL PATTERN PROCESSING

DR MJF GALES

Solutions to 4F10 Pattern Processing, 2011

1. *Mixture Models and EM*

(a) Gaussian mixture models allow multimodal and asymmetric distributions to be modelled. They may also be used to model correlations in the data if diagonal covariance matrices are being used. Gradient descent (GD) is a general optimisation scheme. It simply requires the specification of the gradient given a set of model parameters. A learning rate, η, must then be specified. GD has no convergence proofs and may diverge, or converge very slowly depending on the value of η used. Expectation-maximisation (EM) requires the specification of a set of hidden variables. These variables allow us to define an auxiliary function that we can hopefully maximise. It has guaranteed converge to a local maximum, but may converge very slowly. In contrast to GD, EM is highly dependent on the choice of hidden variable. For GMMs a standard hidden variable can be defined. [30%]

(b) We need to calculate the gradient. Writting the above expression as

$$l(\mathbf{b}) = \sum_{i=1}^{n} \log\left(f(\mathbf{x}_i, \mathbf{b})\right)$$

We can then write

$$\frac{\partial l(\mathbf{b})}{\partial b_j} = \sum_{i=1}^{n} \frac{1}{f(\mathbf{x}_i, \mathbf{b})} \frac{\partial f(\mathbf{x}_i, \mathbf{b})}{\partial b_j}$$

We also can write

$$\frac{\partial f(\mathbf{x}_i, \mathbf{b})}{\partial b_j} = \sum_{m=1}^{M} \frac{P(\omega_m)}{\sqrt{(2\pi)^d |\mathbf{\Sigma}_m|}} \exp\left(-\frac{1}{2}\sum_{j=1}^{d} \frac{(x_{ij} - \mu_{mj} - b_j)^2}{\sigma_{mj}^2}\right)\left(\frac{x_{ij} - \mu_{mj} - b_j}{\sigma_{mj}^2}\right)$$

So we have

$$\left.\frac{\partial l(\mathbf{b})}{\partial b_j}\right|_{\mathbf{b}^{(k)}} = \sum_{i=1}^{n}\sum_{m=1}^{M}\left(a_{mi}^{(k)}\left(\frac{(x_{ij} - \mu_{mj} - b_j^{(k)})}{\sigma_{mj}^2}\right)\right)$$

Hence

$$a_{mi}^{(k)} = P(\omega_m|\mathbf{x}_i, \mathbf{b}^{(k)})$$

As we are using gradient decent we need to examine $-l(\mathbf{b})$. The gradient descent update rule is then

$$b_j^{(k+1)} = b_j^{(k)} + \eta \left.\frac{\partial l(\mathbf{b})}{\partial b_j}\right|_{\mathbf{b}^{(k)}}$$

[35%]

(c) This was discussed in detail in the lecture notes. Mixture models were also discussed in the third year courses. The standard form of auxiliary function defined in lectures is

$$\mathcal{Q}(\mathbf{b}^{(k)}, \mathbf{b}^{(k+1)}) = \sum_{m=1}^{M} \left[\sum_{i=1}^{n} a_{mi}^{(k)} \log(\mathcal{N}(\mathbf{x}_i - \mathbf{b}^{(k+1)}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)) \right] + \text{const}$$

$a_{mi}^{(k)}$ as defined in the question. Differentiating this with respect to $\mathbf{b}^{(k+1)}$ gives

$$\frac{\partial \mathcal{Q}(\mathbf{b}^{(k)}, \mathbf{b}^{(k+1)})}{\partial \mathbf{b}^{(k+1)}} = \sum_{m=1}^{M} \left[\sum_{i=1}^{n} a_{mi}^{(k)} \left((\mathbf{x}_i - \boldsymbol{\mu}_m - \mathbf{b}^{(k+1)})' \boldsymbol{\Sigma}^{-1} \right) \right]$$

Using the fact that diagonal covariance matrices are being used

$$\frac{\partial \mathcal{Q}(\mathbf{b}^{(k)}, \mathbf{b}^{(k+1)})}{\partial b_j^{(k+1)}} = \sum_{m=1}^{M} \left[\sum_{i=1}^{n} a_{mi}^{(k)} \left(\frac{(x_{ij} - \mu_{mj} - b_j^{(k+1)})}{\sigma_{mj}^2} \right) \right]$$

Equating this to zero the estimate at the gives

$$\sum_{m=1}^{M} \left[\sum_{i=1}^{n} a_{mi}^{(k)} \frac{\hat{b}_j}{\sigma_{mj}^2} \right] = \sum_{m=1}^{M} \left[\sum_{i=1}^{n} a_{mi}^{(k)} \left(\frac{(x_{ij} - \mu_{mj})}{\sigma_{mj}^2} \right) \right]$$

So

$$\hat{b}_j = \frac{1}{\sum_{m=1}^{M} \left[\sum_{i=1}^{n} \frac{a_{mi}^{(k)}}{\sigma_{mj}^2} \right]} \sum_{m=1}^{M} \left[\sum_{i=1}^{n} a_{mi}^{(k)} \left(\frac{(x_{ij} - \mu_{mj})}{\sigma_{mj}^2} \right) \right]$$

[35%]

This question looked at Gaussian mixture models and EM training. This was the most popular question, but disappointingly answered. Most candidates understood the attributes of mixture models. However a number moved the log within the summation of the mixture model log-likelihood when performing the differentiation. Most could quote the standard form for the auxiliary function.

2. *Support Vector Machines and Kernels*

(a) 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 Gaussian kernel is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)$$

The kernel can be tuned to a particular task by modifying σ. This parameter determines the smoothness of the resulting decision boundaries. [20%]

(b)(i) The kernel function is

$$\begin{aligned} k(x_i, x_j) &= \frac{1}{2} + \sum_{r=1}^{N} \cos(rx_i)\cos(rx_j) + \sin(rx_i)\sin(rx_j) \\ &= -\frac{1}{2} + \sum_{r=0}^{N} \cos(r(x_i - x_j)) \\ &= -\frac{1}{2} + \frac{1}{2}\sum_{r=0}^{N} (\exp(ir(x_i - x_j)) + \exp(-ir(x_i - x_j))) \end{aligned}$$

This is two geomentric progressions yielding

$$\begin{aligned} k(x_i, x_j) &= -\frac{1}{2} + \frac{1}{2}\left(\frac{1 - \exp(i(N+1)(x_i - x_j))}{1 - \exp(i(x_i - x_j))} + \frac{1 - \exp(-i(N+1)(x_i - x_j))}{1 - \exp(-i(x_i - x_j))}\right) \\ &= -\frac{1}{2} + \frac{1}{2}\left(\frac{\exp(-i(x_i - x_j)/2) - \exp(i(N+1/2)(x_i - x_j))}{\exp(-i(x_i - x_j)/2) - \exp(i(x_i - x_j)/2)}\right. \\ &\qquad \left. + \frac{\exp(i(x_i - x_j)/2) - \exp(-i(N+1/2)(x_i - x_j))}{\exp(i(x_i - x_j)/2) - \exp(-i(x_i - x_j)/2)}\right) \end{aligned}$$

It is then straight forward to show that

$$k(x_i, x_j) = \frac{\sin((N+1/2)(x_i - x_j))}{2\sin((x_i - x_j)/2)}$$

[30%]

(b)(ii) 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. [15%]

(b)(iii) Answer should include:

- the feature-space for this kernel has a fixed dimensionality $2N+1$. In theory the Gaussian kernel can have an infinite dimensional feature space as $\sigma \to 0$.
- both kernels are stationary as they are both functions of differences between the features vector.
- the feature-space described here will map multiple points in the input-space to the same point in feature space $\Phi(x) = \Phi(x + 2\pi N)$. This mapping is not the case for the Gaussian kernel.

[15%]

(c) From the lecture notes Kesler's construct was mentioned. The multi-class problem can be converted to a 2-class problem. Consider an extended observation $\tilde{\mathbf{x}}$ which belongs to class ω_1. Then to be correctly classified

$$\tilde{\mathbf{w}}_1'\tilde{\mathbf{x}} - \tilde{\mathbf{w}}_j'\tilde{\mathbf{x}} > 0, \quad j = 2, \ldots, K$$

There are therefore $K-1$ inequalities requiring that the $K(d+1)$-dimensional vector

$$\boldsymbol{\alpha} = \begin{bmatrix} \tilde{\mathbf{w}}_1 \\ \vdots \\ \tilde{\mathbf{w}}_K \end{bmatrix}$$

correctly classifies all $K-1$ set of $K(d+1)$-dimensional samples

$$\boldsymbol{\gamma}_{12} = \begin{bmatrix} \tilde{\mathbf{x}} \\ -\tilde{\mathbf{x}} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\gamma}_{13} = \begin{bmatrix} \tilde{\mathbf{x}} \\ \mathbf{0} \\ -\tilde{\mathbf{x}} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad \ldots, \quad \boldsymbol{\gamma}_{1K} = \begin{bmatrix} \tilde{\mathbf{x}} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ -\tilde{\mathbf{x}} \end{bmatrix}$$

[20%]

This questions examined the students' knowledge of support vector machines and the use of kernels. The derivation of the form of kernel for the score-space was disappointing, as it had a standard form from the Part 1B Engineering Maths course. Most candidates understood the use, and computational costs, of using kernel representations.

3. *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) 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

$$a + 3a - 1 = 0$$

Hence the value for a is 0.25 [30%]

(c) The threshold is set at 0.5. The value of x that will correspond to this is $x_T = 2$. The probability of error is then given by

$$\begin{aligned}P(\text{error}) &= \frac{1}{2}\int_{x_T}^{\infty}\mathcal{N}(z;0,1)dz + \frac{1}{2}\int_{-\infty}^{x_T}\mathcal{N}(z;1,2)dz \\ &= \frac{1}{2}\left((1-F(2)) + F(1/(\sqrt{2}))\right)\end{aligned}$$

[30%]

(d) A point lying on the decision boundary will satisfy

$$-\log(\sqrt{2\pi}) - \frac{x^2}{2} = -\log(\sqrt{4\pi}) - \frac{(x-1)^2}{4}$$

This may be written as

$$x^2 + 2x - (1 + 4\log(\sqrt{2})) = 0$$

Solving this expression gives

$$x = -1 \pm \sqrt{2 + 4\log(\sqrt{2})}$$

Solving this yields a value of $x = 0.8402$ (a simple sketch illustrates that this is the appropriate root to take). To get the new value of a we need $0.8402a = 0.5$ hence $a = 0.5951$. [30%]

A question based on Bayes' decision rule and least squares estimation. Most candidates understood the use of Bayes' rule to determine decision boundaries. However a number could not use the least squares criterion given to find the estimate of the model parameters, some failing to spot the mean and the variances as limit values with large amounts of data and known distributions.

4. *Neural Network Training*

(a)(i) The values are

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

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

(a)(ii) Letting

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

gives the following differential expression

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

Equating this to zero gives

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

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 b the update is given by $-b/a$. Solving these three equations yields the update rule. [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. [25%]

An unpopular question on second-order optimisation approaches. Those candidates that attempted the question generally performed very well showing a good understanding of second-order approaches and the impact of various assumptions.

5. *ML prediction and Gaussian Processes*

(a)(i) [From lecture notes] Consider a basis function of the form $\phi(||\boldsymbol{x} - \boldsymbol{x}_i||)$, where $\phi()$ is some non-linear function and $||\boldsymbol{x} - \boldsymbol{x}_i||$ is a distance of the vector $\boldsymbol{x}$ from the prototype vector $\boldsymbol{x}_i$. For the case of n training examples each being used as a prototype, the mapping can be defined as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \phi(||\boldsymbol{x} - \boldsymbol{x}_i||) = \boldsymbol{\phi}(\boldsymbol{x})' \boldsymbol{w}$$

where

$$\boldsymbol{\phi}(\boldsymbol{x}) = \left[\begin{array}{ccc} \phi(||\boldsymbol{x} - \boldsymbol{x}_1||) & \ldots & \phi(||\boldsymbol{x} - \boldsymbol{x}_n||) \end{array} \right]'$$

Expressing the training data in matrix format

$$\boldsymbol{\Phi} = \left[\begin{array}{ccc} \phi(||\boldsymbol{x}_1 - \boldsymbol{x}_1||) & \ldots & \phi(||\boldsymbol{x}_1 - \boldsymbol{x}_n||) \\ \vdots & \ddots & \vdots \\ \phi(||\boldsymbol{x}_n - \boldsymbol{x}_1||) & \ldots & \phi(||\boldsymbol{x}_n - \boldsymbol{x}_n||) \end{array} \right] = \left[\begin{array}{c} \boldsymbol{\phi}(\boldsymbol{x}_1)' \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_n)' \end{array} \right]$$

So for the training data

$$\boldsymbol{y} = \boldsymbol{\Phi} \boldsymbol{w} + \boldsymbol{\epsilon}$$

FInally considering the prior

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{0}, \sigma_{\mathrm{w}}^2 \mathbf{I})$$

We need the posterior weight MAP estimate. Now

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{\Phi} \boldsymbol{w}, \sigma_{\epsilon}^2 \mathbf{I})$$

Need the MAP estimate so

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} \left\{ \log(\mathcal{N}(\boldsymbol{y}; \boldsymbol{\Phi} \boldsymbol{w}, \sigma_{\epsilon}^2 \mathbf{I})) + \log(\mathcal{N}(\boldsymbol{w}; \boldsymbol{0}, \sigma_{\mathrm{w}}^2 \mathbf{I})) \right\}$$

Solving (standard completing the square)

$$\hat{\boldsymbol{w}} = \frac{1}{\sigma_{\epsilon}^2} \left(\frac{1}{\sigma_{\epsilon}^2} \boldsymbol{\Phi}' \boldsymbol{\Phi} + \frac{1}{\sigma_{\mathrm{w}}^2} \mathbf{I} \right)^{-1} \boldsymbol{\Phi}' \boldsymbol{y}$$

[40%]

(a)(ii) As the noise is independent of $f(\mathbf{x})$, the prediction is

$$p(y|\hat{\boldsymbol{w}}, \boldsymbol{x}) = \mathcal{N}(y; \hat{\boldsymbol{w}}' \boldsymbol{\phi}(\boldsymbol{x}), \sigma_{\epsilon}^2)$$

[10%]

(b)(i) Interested in the joint distribution

$$\begin{bmatrix} f(\boldsymbol{x}) \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(\boldsymbol{x},\boldsymbol{x}) & \mathbf{k}(\boldsymbol{x},\boldsymbol{X})' \\ \mathbf{k}(\boldsymbol{x},\boldsymbol{X}) & \mathbf{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_\epsilon^2\mathbf{I} \end{bmatrix}\right)$$

Using the equality given in the question

$$p(f(\boldsymbol{x})|\boldsymbol{y},\boldsymbol{X}) = \mathcal{N}(f(\boldsymbol{x}); \mathbf{k}(\boldsymbol{x},\boldsymbol{X})'(\mathbf{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{y}, k(\boldsymbol{x},\boldsymbol{x}) - \mathbf{k}(\boldsymbol{x},\boldsymbol{X})'(\mathbf{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{k}(\boldsymbol{x},\boldsymbol{X}))$$

Thus (noting this is a scalar)

$$\begin{aligned} \mu_{\mathtt{f}} &= \mathbf{k}(\boldsymbol{x},\boldsymbol{X})'(\mathbf{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{y} \\ &= \left((\mathbf{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{y}\right)'\mathbf{k}(\boldsymbol{x},\boldsymbol{X}) \end{aligned}$$

Thus

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = (\mathbf{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_\epsilon^2\mathbf{I})^{-1}\mathbf{y}$$

The form of the squared exponential function is

$$k(\boldsymbol{x},\boldsymbol{x}_j) = \alpha \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}_j||^2}{2\sigma_1^2}\right)$$

[25%]

(b)(ii) The prediction with this has the form

$$p(y|\boldsymbol{x},\boldsymbol{X},\boldsymbol{y}) = \mathcal{N}(y; \mu_{\mathtt{f}}, \sigma_{\mathtt{f}}^2 + \sigma_\epsilon^2)$$

(c) The two forms of prediction for the mean are closely related to one another. If a Gaussian (or RBF) basis is used then $k(\boldsymbol{x},\boldsymbol{x}_j)$ and $\phi(||\boldsymbol{x} - \boldsymbol{x}_i||)$ have identical forms. However when using the MAP estimate the variance on the prediction is always constant. For Gaussian Processes the variance will depend on proximity to training examples. Thus the confidence in the prediction is more realistic for GPs. [15%]

This question examined the candidates knowledge of Gaussian Processes and their relationship to MAP-based approaches. It was disappointing that few candidates could derive the MAP-estimation of the weights from first principles. Most candidates showed a good understanding of the form of the mean of the Gaussian process.