Solutions to 4F10 Pattern Processing, 2008

1. *Bayes' Decision rule and generative models*
   (a)(i) Bayes' decision rule states

$$\text{Decide } \arg\max_{\omega_j} \{P(\omega_j|\boldsymbol{x})\}$$

which can be expressed for the generative classifiers here as

$$\text{Decide } \arg\max_{\omega_j} \{p(\boldsymbol{x}|\omega_j)P(\omega_j)\}$$

[10%]

(a)(ii) A number of points should be discussed

- Generative models use Bayes' decision rule to express the posterior class probability in term of the likelihood and class priors
- Generative models are minimum error classifiers is if there is
  - infinite training data
  - correct models (likelihood and priors)
  - appropriate training algorithm
- Discriminative models directly model the class posteriors.

[20%]

(b)(i) The expression for the probability of error is

$$\begin{aligned} P(\text{error}) &= P(\boldsymbol{x} \in \Omega_2, \omega_1) + P(\boldsymbol{x} \in \Omega_1, \omega_2) \\ &= P(\boldsymbol{x} \in \Omega_2|\omega_1)P(\omega_1) + P(\boldsymbol{x} \in \Omega_1|\omega_2)P(\omega_2) \\ &= \int_{\Omega_2} p(\boldsymbol{x}|\omega_1)P(\omega_1)d\boldsymbol{x} + \int_{\Omega_1} p(\boldsymbol{x}|\omega_2)P(\omega_2)d\boldsymbol{x} \end{aligned}$$

[15%]

(b)(ii) From the inequality given, $a \leq \sqrt{ab}$, if $a \leq b$

$$\int_{\Omega_2} p(\boldsymbol{x}|\omega_1)P(\omega_1)d\boldsymbol{x} \leq \int_{\Omega_2} \sqrt{p(\boldsymbol{x}|\omega_1)P(\omega_1)p(\boldsymbol{x}|\omega_1)P(\omega_1)}d\boldsymbol{x}$$

as by definition in the region where class 2 is labelled

$$p(\boldsymbol{x}|\omega_1)P(\omega_1) \leq p(\boldsymbol{x}|\omega_2)P(\omega_2)$$

A similar expression can be obtained for region Ω_1. Thus

$$P(\text{error}) \leq \int \sqrt{p(\boldsymbol{x}|\omega_1)P(\omega_1)p(\boldsymbol{x}|\omega_2)P(\omega_2)}d\boldsymbol{x}$$

[25%]

(b)(iii) An expression can be obtained based on the inequality in part (b)(ii). The product of two Gaussians is a, un-normalised, Gaussian. Consider

$$\begin{aligned}&\mathcal{N}(\mathbf{x};\mu_1,\Sigma)\mathcal{N}(\mathbf{x};\mu_2,\Sigma) = \\ &\quad \frac{1}{(2\pi)^d|\Sigma|}\exp\left(-\frac{1}{2}\left(2\mathbf{x}\Sigma^{-1}\mathbf{x}' - 2(\mu_1+\mu_2)\Sigma^{-1}\mathbf{x}' + \mu_1\Sigma^{-1}\mu_1' + \mu_2\Sigma^{-1}\mu_2'\right)\right)\end{aligned}$$

Taking the square-root of this gives

$$\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}\left(\mathbf{x}\Sigma^{-1}\mathbf{x}' - 2\frac{(\mu_1+\mu_2)}{2}\Sigma^{-1}\mathbf{x}' + \frac{1}{2}(\mu_1\Sigma^{-1}\mu_1' + \mu_2\Sigma^{-1}\mu_2')\right)\right)$$

Integrating a Gaussian yields 1, so

$$P(\text{error}) \leq K\int\mathcal{N}(\mathbf{x};\frac{(\mu_1+\mu_2)}{2},\Sigma)d\mathbf{x} = K$$

where the constant K can be expressed as (not forgetting the prior)

$$K = \frac{1}{2}\exp\left(\frac{1}{8}(\mu_1+\mu_2)\Sigma^{-1}(\mu_1+\mu_2)' - \frac{1}{4}\mu_1\Sigma^{-1}\mu_1' - \frac{1}{4}\mu_2\Sigma^{-1}\mu_2'\right)$$

[It was also acceptable to find an expression based on the equality in part (b)(i). This yields an expression in terms of cumulative density functions and requires finding the decision boundary.] [30%]

2. *Training Logistic Regression and the use of the Hessian*

(a) Linear decision boundaries passing through the origin. [10%]

(b) The log-likelihood of the data from class ω_1 can be written as

$$\begin{aligned}\mathcal{L}(\mathbf{b}) &= \sum_{i=1}^{n}(y_i \log(P(\omega_1|\mathbf{x}_i,\mathbf{b})) + (1-y_i)\log(P(\omega_2|\mathbf{x}_i,\mathbf{b})) \\ &= \sum_{i=1}^{n}(y_i \log(P(\omega_1|\mathbf{x}_i,\mathbf{b})) + (1-y_i)\log(1-P(\omega_1|\mathbf{x}_i,\mathbf{b}))\end{aligned}$$

[10%]

(c)(i) Differentiating

$$\begin{aligned}\frac{\partial}{\partial \mathbf{b}}P(\omega_1|\mathbf{x},\mathbf{b}) &= \frac{\exp(-\mathbf{b}'\mathbf{x})}{(1+\exp(-\mathbf{b}'\mathbf{x}))^2}\mathbf{x} \\ &= P(\omega_1|\mathbf{b},\mathbf{x})(1-P(\omega_1|\mathbf{b},\mathbf{x}))\mathbf{x}\end{aligned}$$

Thus

$$\begin{aligned}\frac{\partial}{\partial \mathbf{b}}\mathcal{L}(\mathbf{b}) &= \sum_{i=1}^{n}\mathbf{x}_i\left(y_i(1-P(\omega_1|\mathbf{b},\mathbf{x}_i)) - (1-y_i)P(\omega_1|\mathbf{b},\mathbf{x}_i)\right) \\ &= \sum_{i=1}^{n}\mathbf{x}_i\left(y_i - P(\omega_1|\mathbf{b},\mathbf{x}_i)\right)\end{aligned}$$

This can be used in a gradient style approach where

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

[30%]

(c)(ii) Element j,k of the Hessian is

$$h_{jk} = \frac{\partial^2}{\partial b_j \partial b_k}\mathcal{L}(\mathbf{b})$$

Using the above expression

$$\frac{\partial}{\partial b_j}\left(\sum_{i=1}^{n}\left(y_i - P(\omega_1|\mathbf{b},\mathbf{x}_i)\right)x_{ik}\right) = -\sum_{i=1}^{n}P(\omega_1|\mathbf{b},\mathbf{x}_i)(1-P(\omega_1|\mathbf{b},\mathbf{x}_i))x_{ij}x_{ik}$$

This can be expressed in matrix form as

$$\mathbf{H} = -\left[\mathbf{x}_1,\ldots,\mathbf{x}_n\right]\begin{bmatrix} P(\omega_1|\mathbf{b},\mathbf{x}_1)(1-P(\omega_1|\mathbf{b},\mathbf{x}_1)) & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & P(\omega_1|\mathbf{b},\mathbf{x}_n)(1-P(\omega_1|\mathbf{b},\mathbf{x}_n)) \end{bmatrix}\begin{bmatrix}\mathbf{x}_1' \\ \vdots \\ \mathbf{x}_n'\end{bmatrix}'$$

Thus

$$\begin{aligned}
\mathbf{S} &= [\mathbf{x}_1, \dots, \mathbf{x}_n]' \\
\mathbf{R} &= \left[\begin{array}{ccc} P(\omega_1|\mathbf{b}, \mathbf{x}_1)(1 - P(\omega_1|\mathbf{b}, \mathbf{x}_1)) & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & P(\omega_1|\mathbf{b}, \mathbf{x}_n)(1 - P(\omega_1|\mathbf{b}, \mathbf{x}_n)) \end{array}\right]
\end{aligned}$$

[25%]

(c)(iii) The Hessian may be used for optimisation as

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \mathbf{H}^{-1} \left.\frac{\partial}{\partial \mathbf{b}} \mathcal{L}(\mathbf{b})\right|_{\mathbf{b}^{(k)}}$$

where the Hessian is evaluated at the $\mathbf{b}^{(k)}$. Should discuss

- No need to compute η major issue with gradient descent
- If error surface is quadratic - straight to solution
- Hessian may involve computing a large number of parameters (if feature-space is large).

[15%]

(c)(iv) The Hessian is negative-definite for this problem. This implies that the error function is a concave function so has a unique maximum. [10%]

3. *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

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

The output value is again considered to be

$$y = f(\boldsymbol{x}) + \epsilon$$

The values for $\boldsymbol{w}$ needs to estimated. Following the standard linear interpolation example, for the training data

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

So for the training data

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

If the inverse $\boldsymbol{\Phi}^{-1}$ exists then the ML estimate is

$$\hat{\boldsymbol{w}} = \boldsymbol{\Phi}^{-1}\boldsymbol{y}$$

It has been shown that for a large class of functions $\phi()$ if the set of points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ is distinct then $\boldsymbol{\Phi}^{-1}$ exists. [30%]

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

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

[10%]

(b)(i) the form of the squared exponential function is

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

This is a stationary covariance function. [15%]

(b)(ii) Interested in the joint distribution

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

Using the equality given in the question

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

Again to get the distribution of y the noise variance is simply added. So

$$\begin{aligned} c &= k(\boldsymbol{x},\boldsymbol{x}) \\ \mathbf{d} &= \mathbf{k}(\boldsymbol{x},\boldsymbol{X}) \\ \mathbf{E} &= \mathbf{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_\epsilon^2\mathbf{I} \end{aligned}$$

[30%]

(c) Points to mention are

- Prediction variance using Gaussian process is always larger
- Gaussian process variance increases as distance from training points increases

[15%]

4. *M*ixture Models and the Exponential Family

(a) Log-likelihood of the training data is

$$\log(p(x_1,\ldots,x_n|\lambda_1,\ldots,\lambda_M)) = \sum_{i=1}^{n}\log\left(\sum_{m=1}^{M} c_m\lambda_m^{x_i}(1-\lambda_m)^{(1-x_i)}\right)$$

[15%]

(b)(i) EM is an iterative approach to estimating the model parameters. Given the current estimates of the model parameters, $\boldsymbol{\lambda}$, the new estimates, $\hat{\boldsymbol{\lambda}}$, are found using

- Compute component posteriors, $P(\omega_m|x_i,\boldsymbol{\lambda})$, using current parameters.
- Using the Auxiliary function, $\mathcal{Q}(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}})$, compute the new parameters.

[15%]

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

$$\mathcal{Q}(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}}) = \sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|x_i,\boldsymbol{\lambda})\left(x_i\log(\hat{\lambda}_m) + (1-x_i)\log(1-\hat{\lambda}_m)\right)$$

Differentiate this with respect to $\hat{\lambda}_q$ give

$$\frac{\partial\mathcal{Q}(\boldsymbol{\lambda},\hat{\boldsymbol{\lambda}})}{\partial\lambda_q} = \sum_{i=1}^{n} P(\omega_q|x_i,\boldsymbol{\lambda})\left[\frac{x_i}{\hat{\lambda}_q} - \frac{(1-x_i)}{(1-\hat{\lambda}_q)}\right]$$

Equating to zero gives

$$(1-\hat{\lambda}_q)\sum_{i=1}^{n} P(\omega_q|x_i,\boldsymbol{\lambda})x_i = \hat{\lambda}_q\sum_{i=1}^{n} P(\omega_q|x_i,\boldsymbol{\lambda})(1-x_i)$$

Rearranging yields

$$\hat{\lambda}_q = \frac{\sum_{i=1}^{n} P(\omega_q|x_k,\boldsymbol{\lambda})x_i}{\sum_{k=1}^{n} P(\omega_j|x_i,\boldsymbol{\lambda})}$$

[30%]

(c)(i) Re-expressing the Bernoulli distribution

$$\begin{aligned}
p(x|\omega_m,\lambda_m) &= \lambda_m^{x}(1-\lambda_m)^{(1-x)} \\
&= \exp\left(x\log(\lambda_m) + (1-x)\log(1-\lambda_m)\right) \\
&= (1-\lambda_m)\exp\left(x\log\left(\frac{\lambda_m}{1-\lambda_m}\right)\right)
\end{aligned}$$

so

$$\begin{aligned}
\alpha_m &= \log\left(\frac{\lambda_m}{1-\lambda_m}\right) \\
Z_m &= \frac{1}{(1-\lambda_m)}
\end{aligned}$$

[20%]

(c)(ii) Substituting in the expression for the exponential family

$$\mathcal{Q}(\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}) = \sum_{i=1}^{n} \sum_{m=1}^{M} P(\omega_m | x_i, \boldsymbol{\alpha}) \left[-\log(\hat{Z}_m) + \hat{\boldsymbol{\alpha}}_m \mathbf{f}(x_i) \right]$$

Points to mention are:

- Sufficient statistics for auxiliary function are simply

$$\sum_{i=1}^{n} P(\omega_m | x_i, \boldsymbol{\alpha}); \quad \sum_{i=1}^{n} P(\omega_m | x_i, \boldsymbol{\alpha}) \mathbf{f}(x_i)$$

- Solution is not normally linear as Z_m is a function of $\boldsymbol{\alpha}_m$.

[20%]

5. *Support Vector Machines and Speaker Verification*

(a)(i) The following steps are used in Speaker Verification with SVMs

- Train the UBM GMM on all the enrolment data.
- MAP adapt the UBM GMM to the enrolment data of each of the speakers.
- For each enrolled speaker compute the Fisher score-space. To obtain "negative" examples use other speaker's data with the same adapted GMM.
- Train the SVM
- During verification, extract the SVM for the claimed identity and recognise.

[20%]

(a)(ii) The log-likelihood may be expressed as

$$\log(p(\mathbf{O}^{(m)}|\theta)) = \sum_{i=1}^{T^{(m)}} \log\left(\sum_{m=1}^{M} c_m \mathcal{N}(\mathbf{o}_i; \boldsymbol{\mu}_m, \Sigma_m)\right)$$

Standard problem to compute the score-space (described in lectures) Considering just the means of a GMM

$$\boldsymbol{\phi}(\boldsymbol{O}^{(m)}) = \left[\begin{array}{c} \sum_{t=1}^{T^{(m)}} P(\mathtt{1}|\boldsymbol{o}_t, \hat{\boldsymbol{\theta}})\hat{\boldsymbol{\Sigma}}_1^{-1}(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_1) \\ \vdots \\ \sum_{t=1}^{T^{(m)}} P(\mathtt{M}|\boldsymbol{o}_t, \hat{\boldsymbol{\theta}})\hat{\boldsymbol{\Sigma}}_M^{-1}(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_M) \end{array}\right]$$

This is a $M \times d$ features vector. [30%]

(b)(i) For the linear kernel, the sequence kernel looks like

$$\begin{aligned} k(\mathbf{O}^{(m)}, \mathbf{O}^{(n)}) &= \sum_{i=1}^{T^{(m)}} \sum_{j=1}^{T^{(n)}} \mathbf{o}_i^{(m)\prime} \mathbf{o}_j^{(n)} \\ &= \left(\sum_{j=1}^{T^{(n)}} \mathbf{o}_i^{(m)\prime}\right)\left(\sum_{i=1}^{T^{(m)}} \mathbf{o}_j^{(n)}\right) \\ &= T^{(m)}T^{(n)}\boldsymbol{\mu}^{(m)\prime}\boldsymbol{\mu}^{(n)\prime} \end{aligned}$$

Compare this to the Fisher kernel with a single component (assuming $\boldsymbol{\mu} = \mathbf{0}$)

$$\left(\sum_{i=1}^{T^{(m)}} \boldsymbol{\Sigma}^{-1}(\mathbf{o}_i^{(m)} - \boldsymbol{\mu})\right)' \left(\sum_{i=1}^{T^{(n)}} \boldsymbol{\Sigma}^{-1}(\mathbf{o}_j^{(n)} - \boldsymbol{\mu})\right) = T^{(m)}T^{(n)}\boldsymbol{\mu}^{(m)\prime}\boldsymbol{\Sigma}^{-2}\boldsymbol{\mu}^{(n)\prime}$$

The covariance matrix for the component should be an identity matrix for the two kernels to yield the same values. Also the global mean, $\boldsymbol{\mu}$, needs to be zero. This equates to sphering the data prior to constructing the classifiers. [25%]

(b)(ii) Gaussian kernel has the form

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

The following points should be mentioned

- Fisher kernel requires explicit mapping into the feature-space, this is not necessary (or possible) for the Gaussian kernel.
- The computation cost for the Fisher kernel is a function of the number of components M.
- Both schemes use non-linear transformations to derive the feature-space.
- Computational costs are
  - Fisher kernel, $\mathcal{O}(T^{(m)}) + T^{(n)})$ for to derive posteriors - Md dot-product.
  - Sequence kernel, $\mathcal{O}(T^{(m)})T^{(n)})$ as all combinations of observations examined.

[25%]