4F10 Deep Learning and Structured Data, 2021

1. *EM algorithm and mixture models*

   (a) It is necessary to marginalise out over the latent variables

$$\begin{aligned}\log p(\mathbf{X};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}) &= \log\left[\sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi})\right] \\ &= \log\left[\sum_{\mathbf{Z}}\left(\prod_{n=1}^{N}\prod_{m=1}^{M}\pi_m^{\mathbf{1}(z_n=m)}\mathcal{N}(\mathbf{x}_n;\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)^{\mathbf{1}(z_n=m)}\right)\right] \\ &= \sum_{n=1}^{N}\log\left[\sum_{m=1}^{M}\pi_m\mathcal{N}(\mathbf{x}_n;\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)\right] .\end{aligned}$$

[20%]

   (b) i. The auxiliary function is given by

$$\begin{aligned}Q(\boldsymbol{\pi}^{(k)},\boldsymbol{\pi}^{(k+1)}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}^{(k)})\log p(\mathbf{X},\mathbf{Z};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}^{(k+1)}) \\ &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}^{(k)}) \\ &\quad \sum_{n=1}^{N}\sum_{m=1}^{M}\left[\log\left(\pi_m^{(k+1)}\right)^{\mathbf{1}(z_n=m)} + \log\mathcal{N}(\mathbf{x}_n;\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)^{\mathbf{1}(z_n=m)}\right] \\ &= \sum_{n=1}^{N}\sum_{m=1}^{M} p(z_n=m|\mathbf{X};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}^{(k)})\log\pi_m^{(k+1)} + \text{constant}\,.\end{aligned}$$

[20%]

   ii. The Lagrangian is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\pi}^{(k+1)},\lambda) &= Q(\boldsymbol{\theta}^{(k)},\boldsymbol{\theta}^{(k+1)}) + \lambda\left(\sum_{m=1}^{M}\pi_m^{(k+1)} - 1\right) \\ &= \sum_{n=1}^{N}\sum_{m=1}^{M} P(z_n=m|\mathbf{X};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}^{(k)})\log\pi_m^{(k+1)} + \text{constant} + \\ &\quad \lambda\left(\sum_{m=1}^{M}\pi_m^{(k+1)} - 1\right) .\end{aligned}$$

   Differentiating w.r.t. $\boldsymbol{\pi}^{(k+1)}$ and λ yields for $\pi_m^{(k+1)}$

$$\sum_{n=1}^{N}\frac{P(z_n=m|\mathbf{X};\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}^{(k)})}{\pi_m^{(k+1)}} + \lambda = 0$$

$$\sum_{m=1}^{M} \pi_m^{(k+1)} - 1 \;=\; 0$$

Solving these two equations yields the update formula

$$\pi_m^{(k+1)} = \frac{1}{N} \sum_{n=1}^{N} p(z_n = m | \mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}^{(k)}) \,.$$

[20%]

(c) i. We have to sum out $\mathbf{Z}$ and integrate $\mathbf{X}$ (the unobserved variables) to obtain the log-likelihood. As s_n is a binary (observed) variable it can be used to select whether x_n was above or below the threshold. Thus

$$\begin{aligned}
\log p(\mathbf{S}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}, t) &= \log \sum_{\mathbf{Z}} \int \cdots \int p(\mathbf{X}, \mathbf{Z}, \mathbf{S}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}, t)\, dx_1 \cdots dx_N \\
&= \sum_{n=1}^{N} \log \left[\sum_{m=1}^{M} \pi_m \int_{-\infty}^{\infty} \mathcal{N}(x_n; \mu_m, \sigma_m^2) s_n^{\mathbf{1}(x_n \geq t)} (1 - s_n)^{\mathbf{1}(x_n < t)}\, dx_n \right] \\
&= \sum_{n=1}^{N} (1 - s_n) \log \left[\sum_{m=1}^{M} \pi_m \int_{-\infty}^{t} \mathcal{N}(x_n; \mu_m, \sigma_m^2)\, dx_n \right] + \\
&\quad \sum_{n=1}^{N} s_n \log \left[\sum_{m=1}^{M} \pi_m \int_{t}^{\infty} \mathcal{N}(x_n; \mu_m, \sigma_m^2)\, dx_n \right] \\
&= \sum_{n=1}^{N} (1 - s_n) \log \left[\sum_{m=1}^{M} \pi_m \Phi_m(t) \right] + \\
&\quad \sum_{n=1}^{N} s_n \log \left[\sum_{m=1}^{M} \pi_m \overline{\Phi}_m(t) \right] .
\end{aligned}$$

[20%]

ii. Looking at the form of the answer in part (c)(i) it can be seen that this is still in the form of a mixture model where there are two components, one for $s_n = 0$ and the other $s_n = 1$, where the component priors between the two mixture models. Thus the form of EM will again use the same form of latent variable, $P(z_n | \mathbf{x}_n, \boldsymbol{\theta}^{(k)})$. This is in fact a simpler model as, given the mean and variances, component priors are computed that as closely as possible reflect the fraction of samples above, or below, the threshold.

[20%]

Comments This question examined Gaussian Mixture Models and the use of the EM algorithm. This was the least popular question and a number of candidates

submitted poor answers. It was disappointing that the standard form of the log-likelihood of the GMM could not be derived by some candidates using the expression provided. The final part of the question was poorly done, with few candidates being able to derive the appropriate form for the log-likelihood.

2. *Bayes' decision rule, generative and discriminative models and expected loss*

(a) For both decision rules it is assumed that the posterior that comes from the model is "correct".

i. Assigning test sample $\mathbf{x}^\star$ to class ω_j, the loss can be expressed as

$$\begin{aligned}\mathcal{L}(\mathbf{x}^\star, \omega_j) &= \sum_{k \neq j} P(\omega_k|\mathbf{x}^\star) \\ &= 1 - P(\omega_j|\mathbf{x}^\star)\end{aligned}$$

The decision rule then becomes for the generative models

$$\begin{aligned}i^\star &= \arg\max_j \{P(\omega_j|\mathbf{x})\} \\ &= \arg\max_j \left\{ \frac{p(\mathbf{x}^\star|\omega_j)P(\omega_j)}{\sum_{k=1}^K p(\mathbf{x}^\star|\omega_k)P(\omega_k)} \right\} \\ &= \arg\max_j \{p(\mathbf{x}^\star|\omega_j)P(\omega_j)\}\end{aligned}$$

[15%]

ii. The loss now depends on the correct class. For the decision rule assume that the posterior from the generative model is the true posterior. Assigning test sample $\mathbf{x}^\star$ to class ω_j, the loss can be expressed as

$$\mathcal{L}(\mathbf{x}^\star, \omega_j) = \sum_{k \neq j} l_k P(\omega_k|\mathbf{x}^\star)$$

The decision rule now becomes

$$\begin{aligned}i^\star &= \arg\min_j \left\{ \sum_{k \neq j} l_k P(\omega_k|\mathbf{x}) \right\} \\ &= \arg\min_j \left\{ \sum_{k \neq j} l_k \frac{p(\mathbf{x}^\star|\omega_k)P(\omega_k)}{\sum_{m=1}^K p(\mathbf{x}^\star|\omega_m)P(\omega_m)} \right\} \\ &= \arg\min_j \left\{ \sum_{k \neq j} l_k P(\mathbf{x}^\star|\omega_k)P(\omega_k) \right\}\end{aligned}$$

This can also be expressed

$$\begin{aligned}i^\star &= \arg\min_j \left\{ \sum_{k} l_k P(\mathbf{x}^\star|\omega_k)P(\omega_k) - l_j P(\mathbf{x}^\star|\omega_j)P(\omega_j) \right\} \\ &= \arg\max_j \{l_j P(\mathbf{x}^\star|\omega_j)P(\omega_j)\}\end{aligned}$$

as the summation over all classess does not depend on the class being selected. [15%]

(b) The expected loss is the probability that $\mathbf{x}$ falls in Ω_k and the label from $\omega \neq \omega_k$ summed over all classes.

$$\begin{aligned}\mathcal{L} &= \sum_{k=1}^{K} \int_{\Omega_k} \sum_{i \neq k} l_i P(\omega_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ &= \sum_{k=1}^{K} \int_{\Omega_k} \sum_{i \neq k} l_i p(\mathbf{x}, \omega_i) d\mathbf{x} \\ &= \sum_{k=1}^{K} \int_{\Omega_k} \sum_{i \neq k} l_i p(\mathbf{x}|\omega_i) P(\omega_i) d\mathbf{x}\end{aligned}$$

[15%]

(c) i. The i-th output o_i of the softmax function for K inputs $a_1, \ldots, a_K$ is

$$o_i = \frac{\exp(a_i)}{\sum_{k=1}^{K} \exp(a_k)} \,. \tag{1}$$

The class conditional probabilities are obtained when $a_k = \log(P(\omega_k)p(\mathbf{x}|\omega_k))$. Since

$$p(\mathbf{x}|\omega_k) = \prod_{d=1}^{D} \mathcal{N}(x_d|\mu_d^k \, s_d^k) \,,$$

we, therefore, have that

$$\begin{aligned}a_k &= \log(P(\omega_k) + \sum_{d=1}^{D} \left[-\frac{1}{2s_d^k}(x_d - \mu_d^k)^2 - \frac{1}{2}\log(2\pi s_d^k) \right] \\ &= \log(P(\omega_k) - \sum_{d=1}^{D} \left[\frac{1}{2}\log(2\pi s_d^k) + \frac{(\mu_d^k)^2}{2s_d^k} + \frac{1}{2s_d^k}x_d^2 - \frac{\mu_d^k}{s_d^k}x_d \right] \\ &= c_k + \mathbf{x}^{\mathrm{T}}\mathbf{A}_k\mathbf{x} + \mathbf{x}^T\mathbf{b}_k \,,\end{aligned}$$

where $c_k = \log(P(\omega_k) - \sum_{d=1}^{D} \left[\frac{1}{2}\log(2\pi s_d^k) + \frac{(\mu_d^k)^2}{2s_d^k} \right]$, $\mathbf{A}_k = -\mathrm{diag}(0.5/s_1^k, \ldots, 0.5/s_D^k)$ and $\mathbf{b}_k = (\mu_1^k/s_1^k, \ldots, \mu_D^k/s_D^k)^{\mathrm{T}}$. The last line in the equation above is a quadratic function in $\mathbf{x}$. [30%]

ii. If maximum likelihood estimation is used then there are closed form solutions for the class-conditional probability distributions (simple multivariate Gaussian estimation) and the priors. This is simple, but assumes that the Gaussian distributions are an accurate representation of the conditional distributions, and the priors are well estimated.

From the previous part the posterior of the class can be expressed in terms of a softmax and parameters $c_k, \mathbf{A}_k, \mathbf{b}_k$. This can be done using gradient descent, but does not assume that the model parameters are correct. There are the standard problems with gradient descent: learning rates, local minima.
A good answer will also give the modified version of part (b) for the training data fovided

$$\begin{aligned}\mathcal{L} &= \sum_{k=1}^{K} \int_{\Omega_k} \sum_{i \neq k} l_i P(\omega_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ &\approx \frac{1}{N} \sum_{k=1}^{K} \sum_{n=1}^{N} \sum_{i \neq y_n} l_i P(\omega_i|\mathbf{x}_n^{(k)})\end{aligned}$$

Various answers acceptable provided they have been thought through [25%]

Comments This questions examined decision boundaries and Bayes' decision rule. The question was generally well answered with candidates showing a good understanding of how different losses impact the form of decision rule. The answers to the last part of this question were disappointing with many candidates not discussing the simplicity of simply estimating parameters of a multivariate Gaussian versus directly minimising the expected loss.

3. *Deep Learning and Sequence models*

(a) Overall network structure. There is a sequence of d-dimensional vectors. This will be quite long (100's) since we are told that the input is a spoken sentence and there is a new vector every 10 ms. There needs to be a single output obtained from the model to obtain the overall classification i.e. the variable length sequence needs to be mapped to a fixed length vector. A standard network configuration would include:

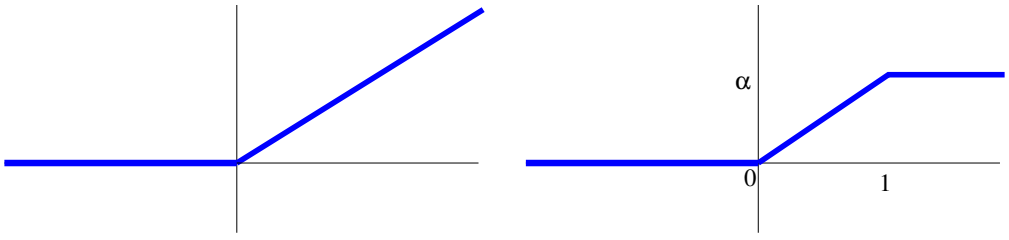
- final output needs to 5-dimensional where each output is associated with one of the emotion classes;
- a soft-max activation function should be used for the last layer of the network so it has the form of posterior class distribution
- Input would be fed in one frame at a time into the recurrent model
- The recurrent units are defined by

$$\mathbf{h}_t = \mathbf{f}^{\mathtt{h}}\left(\mathbf{W}^{\mathtt{f}}_{\mathtt{h}}\mathbf{x}_t + \mathbf{W}^{\mathtt{r}}_{\mathtt{h}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathtt{h}}\right)$$

  where $\mathbf{h}_t$ is the history vector at time t and the two history weight matrices are $\mathbf{W}^{\mathtt{f}}_{\mathtt{h}}$ forward, $\mathbf{W}^{\mathtt{r}}_{\mathtt{h}}$ recursion and $\mathbf{x}_t$ is the input vector . Here $\mathbf{f}^{\mathtt{h}}(\cdot)$ is the recurrent unit activation function.
- There could be a fully connected layer following the recurrent layer.

The overall output must be formed from a fixed dimension vector. This could be done by forming an average, or self-attention, of the history vector outputs (or a further layer on top of this) and then passing this to the softmax or from taking the final history vector output. In this case the average or self-attention is to be preferred since the final value will depend too much on the end of the spoken sentence (& the sentence has 100's of input vectors). [30%]

(b) i. Sketches of the two activation functions are:



Need to compute the first and second moments. First moment given by

$$\int_{-\infty}^{\infty} \phi(x)p(x)dx = \alpha \int_{0}^{\infty} xp(x)dx$$

It is possible to show that when $p(x) = \mathcal{N}(x; 0, \sigma^2)$

$$\int_{0}^{\infty} xp(x)dx = \frac{\sigma}{2}\sqrt{\frac{2}{\pi}}$$

Hence

$$\int_{-\infty}^{\infty} \phi(x)p(x)dx = \alpha\frac{\sigma}{2}\sqrt{\frac{2}{\pi}} = \alpha\sigma\sqrt{\frac{1}{2\pi}}$$

and the second moment

$$\int_{-\infty}^{\infty} (\phi(x))^2 p(x)dx = \int_{0}^{\infty} \alpha^2 x^2 p(x)dx = \alpha^2\sigma^2/2$$

So the total variance on the output is

$$\hat{\sigma}^2 = \alpha^2\sigma^2/2 - \alpha^2\frac{\sigma^2}{2\pi} = \frac{\alpha^2\sigma^2}{2\pi}(\pi - 1)$$

[20%]

ii. The simplest approach is to ensure that the output variance matches the input variances for the initialisation (as discussed in lectures). This function has an added complexity as the mean is non-zero. If the network is deep this could result in a large offset for some layers so a also needs to be dealt with. Any reasonable, well motivated approach will be accepted.
There are assumed to be N nodes for the hidden layer. Each of the output is assumed to be (approximately independent). Assuming that the input to the previous time instance is approximately Gaussian distributed zero mean variance σ^2, then interested in the $z = \mathbf{w}^\mathsf{T}\mathbf{x} + b$. Assuming all independent then

$$\mathbb{E}\left[\mathbf{w}^\mathsf{T}\mathbf{x} + b\right] = \mathbb{E}\left[\mathbf{w}\right]^\mathsf{T}\mathbb{E}\left[\mathbf{x}\right] + \mathbb{E}\left[b\right]$$

setting both the means of $\mathbf{w}$ and b to zero assures a zero mean. set $b = 0$. For the variance, noting all elements independent (non-bold are elements of the vector)

$$\mathbb{E}\left[(\mathbf{w}^\mathsf{T}\mathbf{x})^2\right] = N\mathbb{E}\left[w^2\right]\mathbb{E}\left[x^2\right] = N\sigma_{\mathtt{w}}^2\frac{\alpha^2\sigma^2}{2}$$

To set this to be σ^2

$$w_i \sim \mathcal{N}\left(0, \frac{2}{N\alpha^2}\right)$$

[20%]

iii. The form given in (2) is expected to be more sensitive to initialisation, as there is an upper limit floor on the activation function. There are thus two saturation regions, compared to a single one in the ReLU. [15%]

(c) The simplest approach is to append the noise vector to each of the elements of the sequence. So the system is now trained and evaluated using

$$\tilde{\mathbf{x}}_t = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{n} \end{bmatrix}$$

This allows the network to implicitly remove the noise from each of the speech and noise samples $\mathbf{x}_t$. Note any sensible form is acceptable. [15%]

Comments This questions examined the students' knowledge of deep learning for sequence data. Many candidates showed a good knowledge of the deep-learning approaches discussed in the course, Only a few students missed the importance of generating a fixed length representation for the final classification layer from the sequence of observations.

4. *Support vector machines, margin, kernels*

(a) The margin of a classifier is the distance from the classifier's decision boundary to the nearest data point. Classifiers with large margin work very well empirically, have theoretical results supporting them and are expected to be more robust when the amount of training data is limited. [10%]

(b) i. The squared Euclidean distance (SED) between $\mathbf{x}$ and $\mathbf{x}'$ can be expressed in terms of dot products as follows:

$$\text{SED} = (\mathbf{x} - \mathbf{x}')^{\mathrm{T}}(\mathbf{x} - \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2(\mathbf{x}')^{\mathrm{T}}\mathbf{x}\,.$$

[15%]

ii. We can use the kernel k to write the squared Euclidean distance in the non-linear feature space given by k as follows:

$$\text{SED}_k = k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}', \mathbf{x})\,.$$

[15%]

iii. The nearest-neighbour classifier based on k will outperform the original one when the feature space induced by k has the property that data points from different classes are farther away, but data points from the same class are closer to each other. [15%]

(c) i. The magnitude of the margin is

$$\frac{\mathbf{w}^{\mathrm{T}}(\mathbf{x}_+ - \mathbf{x}_-)}{2\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}} = \frac{(\mathbf{w}^{\mathrm{T}}\mathbf{x}_+ + b - \mathbf{w}^{\mathrm{T}}\mathbf{x}_- - b)}{2\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}} = \frac{1}{\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}}\,.$$

[15%]

ii. Maximizing $1/\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}$ and minimizing $\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}$ is equivalent since $1/x$ is a monotonically decreasing function. Minimizing $\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}$ and $\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$ is again equivalent because $\sqrt{x}$ and $1/2x$ are monotonically increasing functions. [10%]

iii. Since at the solution, $\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$, we can re-write the dual objective as

$$\tilde{\mathcal{L}} = \sum_{n=1}^{N} a_n - \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}\,.$$

At the solution, this should be equal to the original objective, the constraints are all satisfied

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

In particular, since $\mathcal{L} = \tilde{\mathcal{L}}$, after rearranging, we obtain

$$\frac{1}{\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}} = 1/\sqrt{\sum_{n=1}^{N} a_n}$$

which proves the result since $M = \frac{1}{\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}}$. [20%]

Comments This question examined the use of kernels and Support Vector Machine (SVM) training. This was the most popular question and was well answered A number of candidates assumed that the decision boundaries for nearest neighbour classifiers were linear, but the SVM answers were generally good.