4F10 Deep Learning and Structured Data, 2024

1. *Optimisation and Regularisation*

(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)

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

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%]

2. *Deep Learning and Sequenec Modelling*

(a) An iterative approach is used to generate sentences. To gennerate the i-th token given the generated word sequence $\hat{\omega}_{1:i-1}$

$$\hat{\omega}_i \sim P(\omega|\hat{\omega}_{1:i-1})$$

Either sampling or selecting the maximum is acceptable. This word is then added to the gendetated sequence. Theprocess is repeated until an end of sentence symbol, `eos`, is generated and the process terminated. [15%]

(b) The simplest form is the standard cross-entropy optimisation

$$\arg\max \left\{ \sum_{i=1}^{N} \sum_{j=1}^{T} \log(\omega_j^{(i)}|\omega_{1:j-1}^{(i)}) \right\}$$

[5%]

(c)(i) The answer should mention:

- the current back-history for the $\omega_{1:i-1}$ is represented as $\mathbf{h}_{i-1}$.
- the distribution is modelled as

$$P(\omega|\hat{\omega}_{1:i-1}) = \texttt{softmax}(\mathbf{W}_{\texttt{y}}\mathbf{h}_{i-1} + \mathbf{b}_{\texttt{y}})$$

- the generated $\hat{\omega}_i$ is mapped to the embedding form $\mathbf{x}_i$ and the history updated using

$$\mathbf{h}_i = \texttt{ReLU}(\mathbf{W}_{\texttt{f}}\mathbf{h}_{i-1} + \mathbf{W}_{\texttt{x}}\mathbf{x}_i + \mathbf{b}_{\texttt{y}})$$

Any form of (non-linear) activation function is acceptable.

[20%]

(c)(ii) Let the size of the history vector be a. The cost is then for the N training samples (ignores activation cost)

$$N\mu_T\left(Va + a^2 + da\right)$$

[15%]

(d)(i) The answer should mention

- the current back-history for the $\omega_{1:i-1}$ is represented as $\mathbf{c}_{i-1}$.

$$\begin{aligned} \mathbf{c}_{i-1} &= \sum_{j=1}^{i-1} \alpha_j \mathbf{W}_{\texttt{c}}\mathbf{x}_j \\ \alpha_j &= \frac{1}{Z}\exp(\mathbf{x}_j\mathbf{W}_{\alpha}\mathbf{x}_j) \end{aligned}$$

[Any form of self-attention is acceptable]

- the distribution is modelled as

$$P(\omega|\hat{\omega}_{1:i-1}) = \texttt{softmax}(\mathbf{W}_{\texttt{y}}\mathbf{c}_{i-1} + \mathbf{b}_{\texttt{y}})$$

[20%]

(d)(ii) Let the size of the context vector $\mathbf{c}_{i-1}$ be a. The cost is then for the N training samples (ignores acticvton cost). For the output ditribution

$$N\mu_T\,(Va)$$

The cost for the back-history is nore complicated as the cost for the i-th word is given by

$$(i-1)\left(d^2 + da\right)$$

If there is no caching (naive implementation) then the cost is approximately quadratic so

$$N\mathcal{O}(\mu_T^2 + \sigma_T^2)\left(d^2 + da\right)$$

Any sensible diascussion is acceptable. [15%]

(e) The RNN-nbased approach has to be run sequentially for each sentence to genetate the history representation. This not necessary for the self-attention based scheme. [10%]

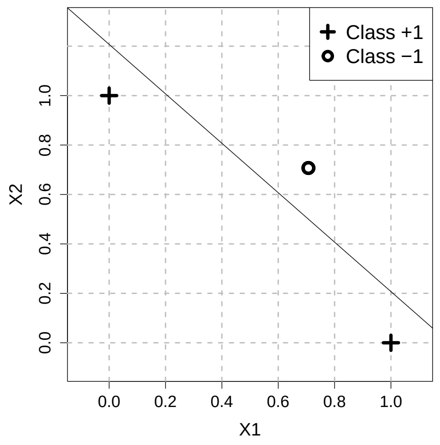
3. *Support vector machines, margin, kernels*

(a) Kernels are functions $k(\mathbf{x}_n, \mathbf{x}_m)$ that compute dot products between feature vectors $\phi(\mathbf{x}_n)$ and $\phi(\mathbf{x}_m)$, that is, $k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m)$, where $\phi(\cdot)$ maps its inputs to a typically higher dimensional space. The main advantage of kernels is that they compute dot products without directly computing $\phi(\cdot)$, 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. [10%]

(b) i. The kernel $k(\mathbf{x}, \mathbf{y})$ is a valid kernel when it satisfies Mercer's condition. For this, first, the kernel must be a symmetric function and satisfy $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x})$, which is the case. Second, it must generate always positive definite kernel matrices. This can be easily verified as follows. Let $\mathbf{X}$ be an $N \times D$ design matrix whose rows are N original input features of dimension D. The corresponding kernel matrix is given by

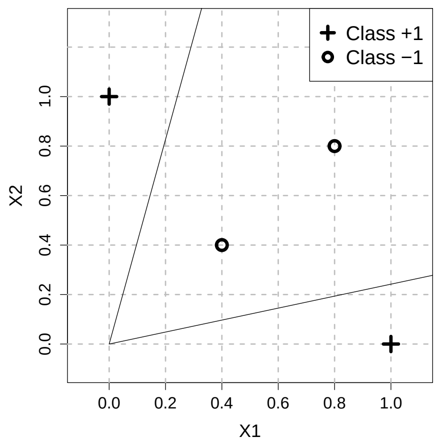
$$\mathbf{K} = \mathbf{AX}(\mathbf{XA})^{\mathrm{T}}\,, \tag{1}$$

where $\mathbf{A}$ is a diagonal matrix with each of its diagonal entries equal to the Euclidean norm of the corresponding row in $\mathbf{X}$. This matrix is clearly positive definite as $\mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u} = a^2$ where $a = \mathbf{u}^{\mathrm{T}}\mathbf{AX}$. An alternative way to show that $k(\mathbf{x}, \mathbf{y})$ is a valid kernel is to note that $k(\mathbf{x}, \mathbf{y})$ is computing a dot product with a given feature mapping $\boldsymbol{\phi}(\cdot)$. In particular, $k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{y})$, where $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}/||\mathbf{x}||$. [10%]



ii. The max margin classifier is a line that is parallel to the two pints with class +1. The two points with class -1 overlap. The max margin classifier must be perpendicular to the shortest line connecting the points with class -1 and the line connecting the points with class +1. The decision boundary will intersect this line at half its length. [20%]

iii. The decision border in input space is given by those points whose coordinate vector, after standardization to have unit length, fall on the decision border in the transformed space. Since all standardized coordinate vectors have the same length, the only variable determining when a point falls on the decision boundary is the angle of the coordinate vector for that point. Two angle values will correspond to the corresponding unit-length vector falling on the decision boundary in the transformed space. These are approximately 13 degrees and 76 degree angles. The points whose coordinate vectors have these angles are shown in the plot below:



[25%]

(c) i. Let $\mathbf{x} = [x_1, x_2]^{\mathrm{T}}$ be the input dimensions of a first data point and $[x_1', x_2']^{\mathrm{T}}$ be the input dimensions of a second one. The product of two one-dimensional kernels evaluated at x_1 and x_1' and x_2 and x_2' is given by

$$\begin{aligned} k'(x_1, x_1') \times k'(x_2, x_2') &= \exp\left\{-\frac{(x_1 - x_1')^2}{2}\right\} \times \exp\left\{-\frac{(x_2 - x_2')^2}{2}\right\} \\ &= \exp\left\{-\frac{(x_1 - x_1')^2 + (x_2 - x_2')^2}{2}\right\} \\ &= \exp\left\{-\frac{(\mathbf{x} - \mathbf{x}')^{\mathrm{T}}(\mathbf{x} - \mathbf{x}')}{2}\right\} \\ &= \exp\left\{-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2}\right\} \end{aligned}$$

[15%]

ii. We have that

$$\begin{aligned} \phi(x)^{\mathrm{T}}\phi(y) &= f(x)f(y)\left[1, \frac{x}{\sqrt{1!}}, \frac{x^2}{\sqrt{2!}}, \frac{x^3}{\sqrt{3!}}, \ldots\right]^{\mathrm{T}}\left[1, \frac{y}{\sqrt{1!}}, \frac{y^2}{\sqrt{2!}}, \frac{y^3}{\sqrt{3!}}, \ldots\right] \\ &= f(x)f(y)\left(1 + \frac{xy}{1!} + \frac{(xy)^2}{2!} + \frac{(xy)^3}{3!} + \cdots\right) \\ &= f(x)f(y)\exp\{xy\}\,. \end{aligned}$$

We also have that

$$\begin{aligned}
k'(x,y) &= \exp\left\{-(x-y)^2/2\right\} \\
&= \exp\left\{-x^2/2 - y^2/2 + xy\right\} \\
&= \exp\left\{-x^2/2\right\}\exp\left\{-y^2/2\right\}\exp\left\{xy\right\} .
\end{aligned}$$

Therefore, $\phi(x)^{\mathrm{T}}\phi(y) = \phi(x)^{\mathrm{T}}\phi(y)$ when $f(x) = \exp\{-x^2/2\}$. [20%]

4. *Expectation Maximisation and Mixture Models*

(a) Z_m must satisfy

$$Z_m = \int \exp\left(\boldsymbol{\alpha}_m^{\mathsf{T}}\mathbf{f}(\mathbf{x})\right) dx$$

this ensures that the PDF integrates to 1. [10%]

(b) The expression for the log-likelihood is

$$l(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \log\left(\sum_{m=1}^{M} c_m \frac{1}{Z_m} \exp\left(\boldsymbol{\alpha}_m^{\mathsf{T}}\mathbf{f}(\mathbf{x}_i)\right)\right)$$

[15%]

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

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}) &= \sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|\mathbf{x}_i, \boldsymbol{\alpha})\left[-\log(\hat{Z}_m) + \hat{\boldsymbol{\alpha}}_m^{\mathsf{T}}\mathbf{f}(\mathbf{x}_i)\right] \\ &= \sum_{m=1}^{M}\left(-\log(\hat{Z}_m)\left(\sum_{i=1}^{n} P(\omega_m|\mathbf{x}_i, \boldsymbol{\alpha})\right) + \hat{\boldsymbol{\alpha}}_m^{\mathsf{T}}\left(\sum_{i=1}^{n} P(\omega_m|\mathbf{x}_i, \boldsymbol{\alpha})\mathbf{f}(\mathbf{x}_i)\right)\right)\end{aligned}$$

Sufficient statistics for auxiliary function is simply

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

for each of the components of the model. [20%]

(c)(ii) Differentiating the auxiliary function yields

$$\begin{aligned}\frac{\partial}{\partial \hat{\boldsymbol{\alpha}}_m}\mathcal{Q}(\boldsymbol{\alpha}, \hat{\boldsymbol{\alpha}}) &= \sum_{i=1}^{N} P(\omega_m|\mathbf{x}_i, \boldsymbol{\alpha})\frac{\partial}{\partial \hat{\boldsymbol{\alpha}}_m}\left(\log(c_m) - \log(Z_m) + \hat{\boldsymbol{\alpha}}_m^{\mathsf{T}}\mathbf{f}(\mathbf{x}_i)\right) \\ &= \sum_{i=1}^{N} P(\omega_m|\mathbf{x}_i, \boldsymbol{\alpha})\left[-\frac{1}{Z_m}\frac{\partial}{\partial \hat{\boldsymbol{\alpha}}_m}Z_m + \mathbf{f}(\mathbf{x}_i)\right]\end{aligned}$$

In the general case the normalisation term will not be linear, so general optimisation approaches are required, for example gradient descent. [20%]

(d)(i) Examining the form of the exponential for

$$-\sum_{i=1}^{d}\frac{(x_i - \mu_{mi})^2}{2\sigma_{mi}^2} = -\sum_{i=1}^{d}\left(\frac{x_i^2}{2\sigma_{mi}^2} - \frac{\mu_{mi}x_i}{\sigma_{mi}^2} + \frac{\mu_{mi}^2}{2\sigma_{mi}^2}\right)$$

Thus the forms in terms of the exponential family can be expresed as

$$
\begin{aligned}
\mathbf{f}(\mathbf{x}) &= \left[x_1, \ldots, x_d, x_1^2, \ldots, x_d^2\right]^{\mathsf{T}} \\
\boldsymbol{\alpha}_m &= \left[\frac{\mu_{m1}}{\sigma_{m1}^2}, \ldots, \frac{\mu_{md}}{\sigma_{md}^2}, \frac{-1}{2\sigma_{m1}^2}, \ldots, \frac{-1}{2\sigma_{md}^2}\right]^{\mathsf{T}} \\
Z_m &= \frac{\exp(-\sum_{i=1}^d \mu_{mi}^2/2\sigma_{mi}^2)}{\sqrt{(2\pi)^d|\Sigma_m|}}
\end{aligned}
$$

[Multiple equivalent forms are acceptable] [20%]

(d)(ii) The standard optimisation approach for $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ has a closed form solution for each EM iteration as the mean can be optimised independently of the variance. Conversely using the general exponential form is iterative for each EM iteration. [15%]