Version JMHL/2

EGT3
ENGINEERING TRIPOS PART IIB

Monday xx April 20xx 2 to 3.40

Module 4F10

CRIB: DEEP LEARNING AND STRUCTURED DATA

*Answer not more than **three** questions.*

All questions carry the same number of marks.

*The **approximate** percentage of marks allocated to each part of a question is indicated in the right margin.*

*Write your candidate number **not** your name on the cover sheet.*

STATIONERY REQUIREMENTS
Single-sided script paper

SPECIAL REQUIREMENTS TO BE SUPPLIED FOR THIS EXAM
CUED approved calculator allowed
Engineering Data Book

10 minutes reading time is allowed for this paper at the start of the exam.

You may not start to read the questions printed on the subsequent pages of this question paper until instructed to do so.

You may not remove any stationery from the Examination Room.

1 *Generative and discriminative classifiers*

(a) **Generative classifiers** model the joint probability distribution $P(X, Y)$, where X represents the input features and Y the class labels. This involves learning the class-conditional probability distribution $P(X|Y)$ and the prior class probabilities $P(Y)$. Once these distributions are known, predictions are made using Bayes' theorem: $P(Y|X) = P(X|Y)P(Y)/P(X)$. They are useful when we need to generate data or handle missing data, but assumptions about $P(X|Y)$ may not always hold, leading to poor performance. Examples: Naïve Bayes or Gaussian Mixture Models.

Discriminative classifiers directly model the conditional probability $P(Y|X)$ or learn a decision boundary that separates the classes without explicitly modeling $P(X|Y)$ or $P(Y)$. These models focus solely on the relationship between inputs and outputs, aiming to maximize predictive accuracy. They often achieve better classification accuracy as they focus directly on the decision boundary, but they lack the generative capabilities. Examples: logistic regression or Support Vector Machines. [15%]

(b) (i) The probability density function for a multivariate Gaussian distribution is

$$P(x|C_k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right),$$

where $k \in \{1, 2\}$ represents the class index. Using Bayes' theorem, the posterior probability $P(C_k|x)$ is proportional to

$$P(C_k|x) \propto P(x|C_k)P(C_k).$$

Taking the logarithm of the posterior ratio $\ln \frac{P(C_1|x)}{P(C_2|x)}$, the decision boundary is determined by

$$\ln P(C_1|x) - \ln P(C_2|x) = 0.$$

Substituting the Gaussian densities and simplifying, we find

$$(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - (x-\mu_2)^T\Sigma^{-1}(x-\mu_2) + 2\ln\frac{P(C_1)}{P(C_2)} = 0.$$

Expanding and simplifying further yields a linear decision boundary

$$(x^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1) - (x^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2) + \ln\frac{P(C_1)}{P(C_2)} = 0.$$

Reorganizing terms, the decision rule becomes

$$x^T\Sigma^{-1}(\mu_1-\mu_2) + \frac{1}{2}(\mu_2^T\Sigma^{-1}\mu_2 - \mu_1^T\Sigma^{-1}\mu_1) + \ln\frac{P(C_1)}{P(C_2)} = 0.$$

[15%]

(ii) Assuming the true class-conditional distributions are known, the probability of classification error can be expressed as

$$P(\text{Error}) = P(C_1) \int_{R_2} P(x|C_1)\, dx + P(C_2) \int_{R_1} P(x|C_2)\, dx,$$

where R_1 and R_2 are the regions assigned to C_1 and C_2, respectively. [15%]

(c) (i) For the logistic regression model, the conditional probability of class C_1 given the input x is given by

$$P(C_1|x) = \frac{1}{1 + \exp(-w^T x - b)}.$$

The probability of class C_2 is

$$P(C_2|x) = 1 - P(C_1|x).$$

Given a dataset $\{x_n, y_n\}_{n=1}^N$, where $y_n \in \{C_1, C_2\}$ represents the class labels, the likelihood of the dataset is

$$L(w, b) = \prod_{n=1}^{N} P(C_1|x_n)^{\mathbf{I}[y_n=C_1]} P(C_2|x_n)^{\mathbf{I}[y_n=C_2]},$$

where $\mathbf{I}[\cdot]$ is the indicator function which takes value 1 when its input is true and 0 otherwise. Taking the logarithm of the likelihood, the log-likelihood function becomes

$$\ell(w, b) = \sum_{n=1}^{N} \left[\mathbf{I}[y_n = C_1] \ln P(C_1|x_n) + \mathbf{I}[y_n = C_2] \ln P(C_2|x_n)\right].$$

[15%]

(ii) The decision rule classifies x as class C_1 when $P(C_1|x) \geq \theta$. The probability of a false positive is

$$P(\text{False Positive}) = 1 - \theta,$$

and the probability of a false negative is

$$P(\text{False Negative}) = \theta.$$

The optimal threshold satisfies

$$C_{FP} \cdot P(\text{False Positive}) = C_{FN} \cdot P(\text{False Negative}).$$

This leads to $\theta = C_{FP}/(C_{FP} + C_{FN})$.

[20%]

(d) The likelihood for each unlabeled data point x_k^U is given by marginalizing over the class variable:

$$P(x_k^U) = \sum_{k=1}^{2} P(x_k^U|C_k)P(C_k),$$

where $P(x_k^U|C_k)$ is the Gaussian density for class C_k and $P(C_k)$ is the prior probability for class C_k. The overall likelihood for the unlabeled dataset is

$$P(X_U) = \prod_{k=1}^{K} P(x_k^U).$$

The total log-likelihood combines the labeled and unlabeled datasets:

$$\ell(\Theta) = \sum_{n=1}^{N} \ln P(x_n, y_n) + \sum_{k=1}^{K} \ln P(x_k^U),$$

where $P(x_n, y_n) = P(x_n|y_n)P(y_n)$ (labeled data) and $P(x_k^U) = \sum_{k=1}^{2} P(x_k^U|C_k)P(C_k)$ (unlabeled data). This total log-likelihood can be optimised to improve the estimation by using the unlabeled data.

[20%]

2 *Mixture of Gaussians and EM*

(a) The joint distribution of X and Z is given by

$$p(X, Z; \theta) = \prod_{n=1}^{N} \prod_{m=1}^{M} [\pi_m \mathcal{N}(x_n; \mu_m, \Sigma_m)]^{I(z_n=m)} .$$

The marginal likelihood is obtained by summing over all possible latent variables Z

$$p(X; \theta) = \sum_{Z} p(X, Z; \theta).$$

Taking the logarithm, the marginal log-likelihood becomes

$$\begin{aligned} \log p(X; \theta) &= \log \sum_{Z} \prod_{n=1}^{N} \prod_{m=1}^{M} [\pi_m \mathcal{N}(x_n; \mu_m, \Sigma_m)]^{I(z_n=m)} \\ &= \sum_{n=1}^{N} \log \left[\sum_{m=1}^{M} \pi_m \mathcal{N}(x_n; \mu_m, \Sigma_m) \right]. \end{aligned}$$

[20%]

(b) (i) The general auxiliary function is

$$Q(\theta^{(k)}, \theta^{(k+1)}) = \mathbb{E}_{Z \sim P(Z|X;\theta^{(k)})} \left[\log p(X, Z; \theta^{(k+1)}) \right].$$

Substituting $p(X, Z; \theta)$, we have

$$Q(\theta^{(k)}, \theta^{(k+1)}) = \sum_{n=1}^{N} \sum_{m=1}^{M} \gamma_{nm} \left[\log \pi_m + \log \mathcal{N}(x_n; \mu_m, \sigma^2 I) \right],$$

where $\gamma_{nm} = P(z_n = m | x_n, \theta^{(k)})$ are the responsibilities. Expanding the Gaussian term, we obtain

$$\log \mathcal{N}(x_n; \mu_m, \sigma^2 I) = -\frac{d}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|x_n - \mu_m\|^2.$$

Thus,

$$Q(\theta^{(k)}, \theta^{(k+1)}) = \sum_{n=1}^{N} \sum_{m=1}^{M} \gamma_{nm} \left[\log \pi_m - \frac{d}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|x_n - \mu_m\|^2 \right].$$

[20%]

(ii) Maximizing Q with respect to μ_m leads to

$$\mu_m^{(k+1)} = \frac{\sum_{n=1}^{N} \gamma_{nm} x_n}{\sum_{n=1}^{N} \gamma_{nm}},$$

where $\gamma_{nm} = \frac{\pi_m^{(k)} \mathcal{N}(x_n; \mu_m^{(k)}, \sigma^2 I)}{\sum_{j=1}^{M} \pi_j^{(k)} \mathcal{N}(x_n; \mu_j^{(k)}, \sigma^2 I)}$. [20%]

(c) (i) The joint distribution with binary observations Y is

$$p(X, Z, Y; \theta) = p(Y|X) p(X, Z; \theta),$$

where $p(Y|X) = \prod_{n=1}^{N} p(y_n|x_n)$ and $p(y_n|x_n)$ is defined as

$$p(y_n|x_n) = \begin{cases} 1, & x_n \in R, \\ 0, & x_n \notin R. \end{cases}$$

[10%]

(ii) The joint distribution $p(Y; \boldsymbol{\theta})$ is obtained by summing out the missing X variables and their corresponding latent variables Z:

$$p(Y; \boldsymbol{\theta}) = \prod_{n=1}^{N} \sum_{m=1}^{M} \pi_m \left[\int_{x_n \in R} \mathcal{N}(x_n; \mu_m, \Sigma_m)\, dx_n \right]^{y_n} \cdot \left[1 - \int_{x_n \in R} \mathcal{N}(x_n; \mu_m, \Sigma_m)\, dx_n \right]^{1-y_n}, \quad (1)$$

where $y_n = 1$ if $x_n \in R$, and $y_n = 0$ otherwise. Here, $\int_{x_n \in R} \mathcal{N}(x_n; \mu_m, \Sigma_m)\, dx_n$ represents the probability mass that the m-th Gaussian component assigns to the region R. [10%]

(iii) The responsibilities γ_{nm}, representing $P(z_n = m \mid y_n, \boldsymbol{\theta}^{(k)})$, are computed as

$$\gamma_{nm} = \frac{\pi_m^{(k)} P(y_n \mid z_n = m, \boldsymbol{\theta}^{(k)})}{\sum_{j=1}^{M} \pi_j^{(k)} P(y_n \mid z_n = j, \boldsymbol{\theta}^{(k)})}, \quad (2)$$

where $P(y_n \mid z_n = m, \boldsymbol{\theta}^{(k)})$ is

$$P(y_n \mid z_n = m, \boldsymbol{\theta}^{(k)}) = \begin{cases} \int_{x_n \in R} \mathcal{N}(x_n; \mu_m^{(k)}, \Sigma_m^{(k)})\, dx_n, & y_n = 1, \\ 1 - \int_{x_n \in R} \mathcal{N}(x_n; \mu_m^{(k)}, \Sigma_m^{(k)})\, dx_n, & y_n = 0. \end{cases} \quad (3)$$

This explicitly incorporates the observed binary labels y_n and the region R into the computation of responsibilities. [20%]

3 *Support vector machines*

(a) A Support Vector Machine (SVM) is a supervised learning model used for binary classification tasks. It seeks to find the hyperplane that best separates the data into two classes. The margin is defined as the distance between the hyperplane and the closest data points, which are known as **support vectors**. These support vectors are critical as they determine the position and orientation of the hyperplane. By maximizing the margin, SVM favours better generalization to unseen data. [15%]

(b) (i) The primal optimization problem is

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n,$$

subject to

$$y_n(w^\top x_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0, \quad n = 1, \ldots, N.$$

The Lagrangian is

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} \alpha_n \left[y_n(w^\top x_n + b) - 1 + \xi_n\right] - \sum_{n=1}^{N} \mu_n \xi_n,$$

where $\alpha_n \geq 0$ and $\mu_n \geq 0$ are Lagrange multipliers. [15%]

(ii) The Karush-Kuhn-Tucker (KKT) conditions are

- 1. Primal feasibility: $y_n(w^\top x_n + b) \geq 1 - \xi_n$ and $\xi_n \geq 0$.
- 2. Dual feasibility: $\alpha_n \geq 0$, $\mu_n \geq 0$.
- 3. Complementary slackness: $\alpha_n \left[y_n(w^\top x_n + b) - 1 + \xi_n\right] = 0$ and $\mu_n \xi_n = 0$.
- 4. Stationarity:

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n, \quad \sum_{n=1}^{N} \alpha_n y_n = 0, \quad \mu_n = C - \alpha_n.$$

Support vectors correspond to data points for which $\alpha_n > 0$. [15%]

(c) (i) The classifier's output is determined as

$$f(x) = \sum_{n=1}^{N} \alpha_n y_n k(x, x_n) + b,$$

where b is derived from the support vectors. The decision boundary is given by $f(x) = 0$.

[15%]

(ii) The dual problem is convex because

- 1. The objective function involves quadratic terms and is concave with respect to α (negative definite Hessian).
- 2. The constraints form a convex set.

[20%]

(iii) Given the dataset $(x_1 = -1, y_1 = +1)$, $(x_2 = 0, y_2 = -1)$, $(x_3 = 1, y_3 = +1)$ with linear kernel $k(x, y) = xy$ and $C = 1$, the dual optimization problem is

$$\max_{\alpha_1,\alpha_2,\alpha_3} \quad W(\alpha) = \sum_{n=1}^{3} \alpha_n - \frac{1}{2} \sum_{n=1}^{3} \sum_{m=1}^{3} \alpha_n \alpha_m y_n y_m k(x_n, x_m),$$

subject to

$$\sum_{n=1}^{3} \alpha_n y_n = 0, \quad 0 \le \alpha_n \le C, \quad \text{for } n = 1, 2, 3.$$

The linear kernel is $k(x, y) = xy$. Using the given data points, the kernel matrix K is

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$$

The dual objective function becomes

$$W(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} \begin{bmatrix} \alpha_1 y_1 & \alpha_2 y_2 & \alpha_3 y_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 y_1 \\ \alpha_2 y_2 \\ \alpha_3 y_3 \end{bmatrix}.$$

Simplify the quadratic term,

$$\frac{1}{2} \sum_{n=1}^{3} \sum_{m=1}^{3} \alpha_n \alpha_m y_n y_m k(x_n, x_m) = \frac{1}{2} \left(\alpha_1^2 - 2\alpha_1 \alpha_3 + \alpha_3^2 \right).$$

Thus, the objective is

$$W(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} \left(\alpha_1^2 - 2\alpha_1 \alpha_3 + \alpha_3^2 \right).$$

The constraints are

- 1. Equality constraint,

$$\sum_{n=1}^{3} \alpha_n y_n = 0 \quad \Longrightarrow \quad \alpha_1 - \alpha_2 + \alpha_3 = 0.$$

(cont.

- 2. Box constraints,

$$0 \leq \alpha_1, \alpha_2, \alpha_3 \leq 1.$$

From the equality constraint,

$$\alpha_2 = \alpha_1 + \alpha_3.$$

Substitute this into the objective function,

$$W(\alpha_1, \alpha_3) = \alpha_1 + (\alpha_1 + \alpha_3) + \alpha_3 - \frac{1}{2}\left(\alpha_1^2 - 2\alpha_1\alpha_3 + \alpha_3^2\right).$$

Simplify,

$$W(\alpha_1, \alpha_3) = 2\alpha_1 + 2\alpha_3 - \frac{1}{2}\left(\alpha_1^2 - 2\alpha_1\alpha_3 + \alpha_3^2\right).$$

We expand and combine terms,

$$W(\alpha_1, \alpha_3) = 2\alpha_1 + 2\alpha_3 - \frac{1}{2}\alpha_1^2 + \alpha_1\alpha_3 - \frac{1}{2}\alpha_3^2.$$

We then simplify to obtain

$$W(\alpha_1, \alpha_3) = 2(\alpha_1 + \alpha_3) - \frac{1}{2}(\alpha_1 - \alpha_3)^2.$$

To maximize this function we note that to minimize the penalty term $(\alpha_1 - \alpha_3)^2$ we need to set $\alpha_1 = \alpha_3$ and to maximize $2(\alpha_1 + \alpha_3)$ we need to make $\alpha_1 + \alpha_3$ as large as possible. Since the equality constraint says $\alpha_2 = \alpha_1 + \alpha_3$ and the box constraint says $\alpha_2 \leq 1$, the maximum value of the first term $2(\alpha_1 + \alpha_3)$ in the objective above is obtained when $\alpha_1 + \alpha_3 = 1$. Solving $\alpha_1 = \alpha_3$ and $\alpha_1 + \alpha_3 = 1$ gives

$$\alpha_1 = 0.5, \quad \alpha_3 = 0.5.$$

Substituting these into the constraint for α_2 leads to

$$\alpha_2 = 0.5 + 0.5 = 1.0.$$

The optimal values of the dual variables are then

$$\alpha_1 = 0.5, \quad \alpha_2 = 1.0, \quad \alpha_3 = 0.5.$$

[20%]

4 *Transformer neural networks*

(a) (i) The output layer should be a **softmax layer** that maps the output of the final transformer block to a probability distribution over the French vocabulary. The output layer involves: 1) a **weight matrix** of shape (d, V_{French}) and 2) a **bias vector** of shape (V_{French}). The total number of parameters in the output layer is $d \times V_{\text{French}} + V_{\text{French}}$. [10%]

(ii) A. **Poistional encodings**: they enable the model to consider the order of the sequence elements during attention computation. Without positional encoding, the model would treat all input embeddings as a "bag of words" without any temporal context. Sinusoidal functions are used to encode positions. These functions allow the model to generalize to unseen sequence lengths because of their continuous nature.

B. **Self-attention**: Allows to compute relationships between all words in the sequence in parallel, including long-range ones, which are crucial for tasks like translation where word order and context are vital. It helps the model focus on relevant parts of the input sequence for generating each output word. Computes attention scores between words by taking dot products between transformed versions of the input embeddings. These scores are then normalized using a softmax function to produce attention weights, which determine how much influence each word has on another. [20%]

(b) (i) The self-attention mechanism performs the following multiplications for a sequence of length $(T + T')$ with feature dimension d:

- **Value, Key an Query Matrices**. Calculating the value, query and key matrices V, Q, and K involves in each case multiplying an embedding matrix $(T+T') \times d$ with a weight matrix $d \times d$, costing $3(T + T')d^2$.
- **Query-Key Multiplication**. Calculating the product of query and key matrices $QK^\top$ involves multiplying a $(T + T') \times d$ matrix with a $d \times (T + T')$ matrix, costing $(T + T')^2 d$.
- **Final Multiplication**. Multiplying the attention matrix $((T + T') \times (T + T'))$ with the value matrix V $((T + T') \times d)$ costs $(T + T')^2 d$.

The total cost is $2(T + T')^2 d + 3(T + T')d^2$. For long sequences $(T + T' \gg 1)$, the quadratic term in $T + T'$ dominates, posing challenges for scalability.

[20%]

(ii) Multi-head attention extends single-head attention by using multiple attention

heads. Each head learns a separate representation by applying attention with independently trained query, key and value (Q, K, and V) projections of the embeddings. The outputs of all heads are linearly combined to produce the final output. Multiple attention heads allow the model to attend to different parts of the input sequence simultaneously. This enables the model to capture different types of relationships simultaneously and provides richer and more versatile representations of the sequence elements. [15%]

(iii) Masking is applied to the self-attention mechanism to restrict the flow of information. A **causal mask** matrix is used to ensure that the attention mechanism at position t only considers embeddings from positions $< t$. Mathematically, the key-query product for future embeddings are set to $-\infty$ before evaluating the softmax function. For query and key matrices Q and K, we have

$$\text{Attention-Weights}(Q, K) = \text{softmax}\left(QK^T + M\right),$$

where M is the mask matrix, defined as

$$M[i, j] = \begin{cases} 0 & \text{if } j < i, \\ -\infty & \text{if } j \geq i. \end{cases}$$

The addition of $-\infty$ ensures that the softmax outputs for future embeddings become zero, effectively masking them.

[15%]

(c) Comparison and contrast between transformers and RNNs:

- **Parallelism**: Transformers process all tokens simultaneously, while RNNs require sequential processing, making transformers faster for long sequences by using GPU acceleration.
- **Sequence Length**: Transformers handle long-range dependencies efficiently due to self-attention, whereas RNNs struggle with vanishing gradients.
- **Scalability**: Transformers scale better with hardware accelerators, while RNNs are limited by sequential operations. However, transformers scale worse with sequence length as their cost is quadratic, while RNNs have a linear cost.

[20%]

END OF PAPER

THIS PAGE IS BLANK