NATIONAL UNIVERSITY OF SINGAPORE

AIS5103 Foundations of Deep Learning

(Semester II: AY 2025-26)

Time Allowed: 2 Hours

INSTRUCTIONS TO STUDENTS

1. Please write only your student number. Do not write your name.
2. This assessment paper contains 4 questions and comprises 3 printed pages.
3. Students are required to answer all questions.
4. All questions carry equal marks (25 marks each).
5. Students should write the answers for each question on a new page.
6. This is a CLOSED BOOK examination.
7. Electronic calculators are allowed.

1. Choose the best one among a to d multiple-choice answers:
   I. Consider a multi-layer perceptron with width N (number of neurons in each layer) and depth L (number of layers). The computational complexity of backpropagation is
      a. $O(NL)$
      b. $O(N^2L)$
      c. $O(NL^2)$
      d. $O(N)$

   II. If $p(x)$ is the probability of a single datum, the likelihood function in statistics is
      a. $-p(x)\ln p(x)$
      b. $p(x^{(1)})p(x^{(2)})\cdots p(x^{(N)})$
      c. $\int dx\, p(x)\ln q(x)$
      d. $p(x^{(1)})+p(x^{(2)})+\cdots+p(x^{(N)})$

   III. In a convolutional neural network (CNN), the learnable parameters are such that
      a. it takes care of translational symmetry.
      b. it takes care of rotational symmetry.
      c. it takes into account scale invariance.
      d. it takes care of the group properties.

   IV. The attention mechanism in the transformer means
      a. mixing the features in a nonlinear way.
      b. correlating the tokens to capture long-range dependencies.
      c. focusing on the relevant part of tokens.
      d. improving GPU performance by matrix computation.

   I b, II b, III a, IV b. Note: II the likelihood function is the joint probability of seeing the whole data set $(x^{(1)}, x^{(2)}, \cdots, x^{(n)})$, as a function of the model parameters in the probability distribution p. IV focusing on the relevant part may seem the answer as it is called "attention", but really, it is the correlation of key and query that is important.

2. In this programming problem, we write Python/Pytorch code for the layer normalization and the residual connection. The layer normalization is applied to the design matrix X (with elements x_{ni} where the row index n is for the batch direction, and i index the feature direction).

a. Write a function that takes the design matrix X as input and returns a layer-normalized version of it. Note that the layer-normalized version has learnable parameters for the mean and variance.
b. Let $\mathrm{MLP}(\mathbf{x}^{(n)})$ be a given function for the multilayer perceptron. Use it to define a new function for one block of the residual connection.

a Let the input matrix elements be x_{ni}. Layer normalization means we normalize over the feature direction i for each batch direction n. So, the code will be based on the formulas $\mu_n = \frac{1}{D}\sum_{i=1}^{D} x_{ni}$, $\sigma_n^2 = \frac{1}{D}\sum_{i=1}^{D}(x_{ni} - \mu_n)^2$, $a_{ni} = \frac{x_{ni}-\mu_n}{\sqrt{\sigma_n^2+0.0001}}$, and finally, return $b_{ni} = \alpha_n a_{ni} + \beta_n$. Here we must register α and β as learnable parameters (and can take derivatives just like W). b : return y = MLP(x) + x with input x. Codes omitted.

3. This question is about back-propagation in neural networks. Consider a two-layer feedforward perceptron represented by the equations

$$\mathbf{a} = W^{(1)}\mathbf{x}, \quad \mathbf{z} = h(\mathbf{a}), \quad \mathbf{y} = W^{(2)}\mathbf{z},$$

here, $\mathbf{x}$, $\mathbf{a}$, $\mathbf{z}$, $\mathbf{y}$ are vectors. $h(\cdot)$ is the elementwise nonlinear function with derivative h', $W^{(1)}$ and $W^{(2)}$ are learnable matrices. The loss function is given by mean-squared error $E = \frac{1}{2}\|\mathbf{y} - \mathbf{t}\|_2^2$. $\mathbf{t}$ is the target vector. A labeled data point is the pair $(\mathbf{x}, \mathbf{t})$.

a. Express the differential dE in terms of the differential of the weights, $dW^{(1)}$ and $dW^{(2)}$.
b. Show that the gradients are given by outer products of appropriate vectors, $\frac{\partial E}{\partial W^{(1)}} = \boldsymbol{\delta}\mathbf{x}^T$ and $\frac{\partial E}{\partial W^{(2)}} = \mathbf{e}\mathbf{z}^T$. What are the quantities $\boldsymbol{\delta}$ and $\mathbf{e}$?

To save typing, vectors are in normal font rather than roman bold. a: Let $y - t = e$, then $E = \frac{1}{2}e^T e$, so $dE = e^T de = e^T d(y - t) = e^T dy = e^T d(W^{(2)}z)$. Using the Leibniz rule, $d\left(W^{(2)}z\right) = \left(dW^{(2)}\right)z + W^{(2)}dz$. Substituting this into dE, we get $dE = e^T dW^{(2)}z + e^T W^{(2)}dz$. Note that dE is a scalar, so we can alternatively write in terms of trace, in order to use the cyclic property of trace, thus we also have $dE = Tr\left(e^T dW^{(2)}z\right) + Tr\left(e^T W^{(2)}dz\right) = Tr\left(ze^T dW^{(2)}\right) + Tr\left(e^T W^{(2)}h'(a) \otimes da\right)$. Here we used $z = h(a)$, and $dz = h'(a)da$, but the vector da and $h'(a)$ are elementwise multiplication. To make this clear, we use the symol $\otimes$ so that $h'(a) \otimes da$ is still a column vector. Now $da = d\left(W^{(1)}x\right) = \left(dW^{(1)}\right)x$ $(as\ x\ is\ a\ constant)$. Since $Tr(A) = Tr(A^T)$, we transpose the term inside the traces and put $\left(dW^{(1\ or\ 2)}\right)^T$ as the first matrix to get a norm convenient form as $dE = Tr\left(\left(dW^{(2)}\right)^T ez^T\right) + Tr(\left(dW^{(1)}\right)^T h'(a) \otimes W^{(2)^T} ex^T)$. This is the final form. b: We can read off the partial derivatives as the coefficients of dW in the above trace formula. Note the coefficients of the differential is the gradient, $dE = \nabla E \cdot dW$. Thus, $e = y - t$, $\delta = h' \otimes (\left(W^{(2)}\right)^T e)$. Note: see my slides for week 5, page 3 and

4. Since the quantities such as e, y or a, or z are (column) vectors or W matrices, the order of these quantities is very important, e.g., eW does not make sense, as e is a column vector (N by 1), and W is a matrix (N by M), so we should write $e^T W$. Note that since the quantities we are dealing with are vectors and matrices, the order is important, so if Wx makes sense, then xW does not. So you cannot change the positions freely as if they are scalar. If you find the vector/matrix notation confusing, you can just work with indices with components as in my slides.

4. The method of normalizing flow is to generate an arbitrary distribution $P(\mathbf{x})$ driven from a standard normal distribution $N(0,1)$ by latent variables $\mathbf{z}$. In the normalizing flow with real non-volume preserving (NVP), the transformation from the latent variables $\mathbf{z}$ to $\mathbf{x}$ is according to

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_A \\ \mathbf{x}_B \end{pmatrix} = \begin{pmatrix} \mathbf{z}_A \\ \exp(\mathbf{s}(\mathbf{z}_A, w)) \odot \mathbf{z}_B + \mathbf{b}(\mathbf{z}_A, w) \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} \mathbf{z}_A \\ \mathbf{z}_B \end{pmatrix}$$

where $\boldsymbol{z}_A, \mathbf{z}_B, \mathbf{x}_A, \mathbf{x}_B, \mathbf{s}, \mathbf{b}$ are all vectors, and $\odot$ stands for elementwise multiplication. The functions $\mathbf{s}$ and $\mathbf{b}$ are learnable functions implemented, e.g., by multi-layer perceptions.

   a. Give the explicit formula for the inverse map, i.e., express $\mathbf{z}$ in terms of $\mathbf{x}$.
   b. Compute the Jacobian matrix $J = \frac{\partial \mathbf{z}}{\partial \mathbf{x}}$.
   c. In the normalizing flow, what is the loss or error function? Give the loss function explicitly in terms of the Jacobian.

a the inverse transform is

$$\mathbf{z} = \begin{pmatrix} \mathbf{z}_A \\ \mathbf{z}_B \end{pmatrix} = \begin{pmatrix} \mathbf{x}_A \\ \exp(-\mathbf{s}(\mathbf{x}_A, w)) \odot (\mathbf{x}_B - \mathbf{b}(\mathbf{x}_A, w)) \end{pmatrix}$$

b $J = \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial \mathbf{z}_A}{\partial x_A} & \frac{\partial \mathbf{z}_A}{\partial x_B} \\ \frac{\partial \mathbf{z}_B}{\partial \mathbf{z}_A} & \frac{\partial \mathbf{z}_B}{\partial \mathbf{x}_B} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \ldots & \mathrm{diag}(e^{-s_j}) \end{pmatrix}$

note since $\mathbf{z}$ *or* $\mathbf{x}$*, or* $\mathbf{z}_A$ *etc, in boldface are vectors, the partial derivative means the partial derivatives of all the components of the vector, i.e.,* $J_{ij} = \frac{\partial z_i}{\partial x_j}$*. So, AA entry is an identity matrix I (not the number 1), and BB entry is a diagonal matrix with diagonal elements given by* e^{-s_j} *at the j-th entry. The sizes of these matrices depends on the dimensions of the subvectors.*

c use the negative log-likelihood, this is (for each data point x) Loss = $-\ln p_x(x) = -\ln(p_z(z) \det J) = \frac{z^2}{2} + \sum_j s_j$ *(we need use the inverse transform to express z in terms of x. Note that det J =* $\prod_j e^{-s_j}$*.*

Note: see my course slides on week 11 (ML-slides-10.pptx), page 9-12. The inverse transform, J and det(J) are on page 12; the loss function is the negative log-likelihood on page 10.

--- the end of paper --- WJS