LH Neural Computation

Solutions

Main Summer Examinations 2023

Note

Answer ALL questions. Each question will be marked out of 20. The paper will be marked out of 60, which will be rescaled to a mark out of 100.

Question 1

Let $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ be vectors in $\mathbb{R}^d$. Consider the minimization of the following function

$$C(\mathbf{w}) = \frac{1}{2n} \sum_{i=1}^{n} \|\mathbf{w} - \mathbf{x}^{(i)}\|_2^2,$$

where $\|\cdot\|_2$ is the Euclidean norm, i.e., $\|\mathbf{w}\|_2^2 = \sum_{j=1}^{d} w_j^2$ for $\mathbf{w} = (w_1, \ldots, w_d)^\top \in \mathbb{R}^d$.

(a) What is the global minimiser of C? Give your arguments. **[5 marks]**

(b) Suppose we apply stochastic gradient descent to this problem with $\mathbf{w}^{(0)} = (0, 0, \ldots, 0)^\top$, step size $\eta_t = 1/(t+1)$ and $i_t = t+1$ $(t < n)$, i.e., when we derive $\mathbf{w}^{(t+1)}$ from $\mathbf{w}^{(t)}$ we use $\eta_t = 1/(t+1)$ and $\mathbf{x}^{(t+1)}$ to compute a stochastic gradient. Compute $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \mathbf{w}^{(3)}$. Based on these computation, write a general formula for $\mathbf{w}^{(k)}$, $k \leq n$. Give your arguments to explain this general formula. **[11 marks]**

(c) Suppose we apply gradient descent to train a neural network. In which step of gradient descent do we use backpropagation? **[4 marks]**

Model answer / LOs / Creativity:

(a) (Creative, LO1, LO2) The gradient of the objective function is

$$\nabla C(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{w} - \mathbf{x}^{(i)}) = \mathbf{w} - \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}^{(i)}.$$

According to the first-order optimality condition, we have the minimiser is

$$\mathbf{w}^* = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}^{(i)}.$$

(b) (Creative, LO1, LO2) We define $C_i(\mathbf{w}) = \frac{1}{2}\|\mathbf{w} - \mathbf{x}^{(i)}\|_2^2$. The gradient of C_i is

$$\nabla C_i(\mathbf{w}) = \mathbf{w} - \mathbf{x}^{(i)}.$$

For SGD, we have

$$\mathbf{w}^{(1)} = \mathbf{w}^{(0)} - \mathbf{w}^{(0)} + \mathbf{x}^{(1)} = \mathbf{x}^{(1)}$$
$$\mathbf{w}^{(2)} = \mathbf{w}^{(1)} - \frac{1}{2}(\mathbf{w}^{(1)} - \mathbf{x}^{(2)}) = \frac{1}{2}\mathbf{x}^{(1)} + \frac{1}{2}\mathbf{x}^{(2)}$$
$$\mathbf{w}^{(3)} = \mathbf{w}^{(2)} - \frac{1}{3}(\mathbf{w}^{(2)} - \mathbf{x}^{(3)}) = \frac{1}{3}\mathbf{x}^{(1)} + \frac{1}{3}\mathbf{x}^{(2)} + \frac{1}{3}\mathbf{x}^{(3)}.$$

Generalizing from here, we can conclude by induction that

$$\mathbf{w}^{(t)} = \frac{1}{t}\sum_{i=1}^{t}\mathbf{x}^{(i)}, \quad 0 < t \leq n. \tag{1}$$

Indeed,

$$\begin{aligned}\mathbf{w}^{(t)} &= \mathbf{w}^{(t-1)} - \frac{1}{t}\nabla C_t(\mathbf{w}^{(t-1)}) = \mathbf{w}^{(t-1)} - \frac{1}{t}\big(\mathbf{w}^{(t-1)} - \mathbf{x}^{(t)}\big) \\ &= \Big(\frac{t-1}{t}\Big)\mathbf{w}^{(t-1)} + \frac{1}{t}\mathbf{x}^{(t)} = \frac{t-1}{t}\frac{1}{t-1}\sum_{i=1}^{t-1}\mathbf{x}^{(i)} + \frac{1}{t}\mathbf{x}^{(t)} \\ &= \frac{1}{t}\sum_{i=1}^{t}\mathbf{x}^{(i)},\end{aligned}$$

where in the last second identity we use the induction hypothesis (1). Then $\mathbf{w}^{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)}$.

(c) (Bookwork, LO2) We use backpropagation to compute gradient in the gradient descent algorithm.

Question 2

We have the following questions related to convolutional neural networks (CNNs). Please answer them with **justifications**.

(a) Given the convolution kernel $W \in \mathbb{R}^{3\times 3}$ and matrix $A \in \mathbb{R}^{5\times 5}$, we perform the following convolution with padding $= 0$ and stride $= 1$:

$$W \circledast A = B,$$

where $\circledast$ represents a 2D convolution, and A and W are respectively given as follows:

$$A = \{a_{ij}\} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \\ 4 & 5 & 1 & 2 & 3 \\ 3 & 4 & 5 & 1 & 2 \\ 2 & 3 & 4 & 5 & 1 \end{bmatrix} \text{ and } W = \{w_{ij}\} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

Please fill in the missing entries of the matrix $B \in \mathbb{R}^{3\times 3}$ below:

$$B = \{b_{ij}\} = \begin{bmatrix} \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots \\ 5 & \ldots & \ldots \end{bmatrix}.$$

Note that you need to write down B on your answer sheet. **[7 marks]**

(b) Next, following the question (a) above, during convolution each element of A will be multiplied by some element of the convolution kernel $W \in \mathbb{R}^{3\times3}$ a number of times. Fill in the following matrix with the number of times each element is multiplied by an element of the weight matrix W:

$$\begin{bmatrix} 1 & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & 9 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}.$$

Note that you need to write down W on your answer sheet. **[7 marks]**

(c) We then calculate the following loss function f, which involves B computed in the question (a) above and some matrix $C \in \mathbb{R}^{3\times3}$:

$$f = \Sigma_{i=1}^{3}(b_{ii} - c_{ii})^2,$$

where c_{ij} denotes the entries of matrix C which are defined as follows:

$$C = \{c_{ij}\} = \begin{bmatrix} -10 & 0 & 0 \\ 0 & -10 & 0 \\ 0 & 0 & -10 \end{bmatrix}.$$

As is the case with neural networks, the weights of a CNN will be updated using backpropagation which relies on the chain rule to calculate the derivatives of the loss function with respect to the weights. Use the chain rule to calculate and evaluate the derivatives $\frac{\delta f}{\delta w_{11}}$ and $\frac{\delta f}{\delta a_{11}}$, where w_{11} and a_{11} are the first entry of W and A, respectively. Note that W and A are given in the question (a). **[6 marks]**

Model answer / LOs / Creativity:

(a) Bookwork, LO2.
The key point to answer this question is to how to do convolutions with stride 1 and zero padding. In this case, it will result in a reduced matrix size of 3x3. Hence:

$$B = \begin{bmatrix} -15 & -5 & 0 \\ 15 & -15 & -5 \\ 5 & 15 & -15 \end{bmatrix}$$

(b) Creative, LO1, LO2.
This question has not been taught over lectures. However, as long as the student understands how convolutions work, it is straightforward to answer this question. For each entry of A, simply counts the number of times the kernel has occurred

on that pixel. As such, the matrix is given as follows, which is a centrosymmetric matrix.

$$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

(c) Creative, LO1, LO2.
This question is about the chain rules, which we have covered in the first few lectures. To answer this question correctly, the student should understand how convolutions work as well as how to derive a derivative when it comes to convolutions. We have designed the question carefully by using same coefficients so the derivations should have a pattern and hence is easier. Each derivative is worth 3 marks. The student will award half marks if derivations are right but answers wrong.

$$\begin{aligned} \frac{\delta f}{w_{11}} &= \Sigma_{i,j=1}^{3} \frac{\delta f}{B_{ij}} \frac{\delta B_{ij}}{w_{11}} \\ &= \Sigma_{i=1}^{3} \frac{\delta f}{B_{ii}} \frac{\delta B_{ii}}{w_{11}} \\ &= \frac{\delta f}{B_{11}} \frac{\delta B_{11}}{w_{11}} + \frac{\delta f}{B_{22}} \frac{\delta B_{22}}{w_{11}} + \frac{\delta f}{B_{33}} \frac{\delta f}{w_{11}} \\ &= 2 \times (B_{11} + 10) \times 1 + 2 \times (B_{22} + 10) \times 1 + 2 \times (B_{33} + 10) \times 1 \\ &= -30 \end{aligned}$$

$$\begin{aligned} \frac{\delta f}{a_{11}} &= \Sigma_{i,j=1}^{3} \frac{\delta f}{B_{ij}} \frac{\delta B_{ij}}{a_{11}} \\ &= \frac{\delta f}{B_{11}} \frac{\delta B_{11}}{a_{11}} \\ &= 2 \times (B_{11} + 10) \times -1 \\ &= 10 \end{aligned}$$

Question 3

(a) An auto-encoder (AE) consists of an encoding unit f_φ, a latent representation z, and a decoding unit g_θ. The goal of an auto-encoder is to learn to produce output $\hat{x} = g_\theta(z) = g_\theta(f_\varphi(x))$ for a given input x, such that $\hat{x} = x$ and where $z = f_\varphi(x)$.

(i) During the model training process, the auto-encoder model can naively learn the identity function making it a useless model. Describe briefly why we consider such a model as a useless model and how we make the auto-encoder learn a useful latent representation z instead of an identity function. **[5 marks]**

(ii) Given the trained auto-encoder consisting of f_φ, g_θ and z, consider that it has the ability for good self-reconstruction. Is this auto-encoder suitable for generating synthetic (or new) data? Justify your answer with brief reasoning. **[4 marks]**

(b) A variational auto-encoder (VAE) relies on the following loss function which consists of two terms:

$$\mathcal{L}_{VAE} = \mathcal{L}_{rec} + \mathcal{L}_{reg},$$

where $\mathcal{L}_{rec}$ represents the reconstruction loss and $\mathcal{L}_{reg}$ represents the regularisation loss. Briefly describe about what happens if we exclude the $\mathcal{L}_{reg}$ term from the VAE loss function such that $\mathcal{L}_{VAE} = \mathcal{L}_{rec}$ to train the VAE. **[5 marks]**

(c) A generative adversarial network (GAN) consists of two units: a generator G_θ to generate fake data samples and a discriminator D_φ to recognise whether a sample is real or fake. We need to design a good loss function to train the GAN discriminator unit for learning its parameters φ. We have designed the below loss function for discriminator learning:

$$\min_{\varphi} \mathbb{E}_{x\sim p_{data}(x)}\Big[-\log\big(1 - D_\varphi(x)\big)\Big] + \mathbb{E}_{z\sim p(z)}\Big[-\log\big(D_\varphi(G_\theta(z))\big)\Big],$$

where $\mathbb{E}$ denotes the expectation operator, $x \sim p_{data}(x)$ denotes input sample x drawn from real data distribution $p_{data}(x)$, and $z \sim p(z)$ denotes GAN generated data z drawn from fake data distribution $p(z)$.

Briefly describe what this loss function is doing and how it can be changed to help in GAN discriminator training. **[6 marks]**

Model answer / LOs / Creativity:

(a) Creative, LO1, LO2.

(i) The model is considered useless as it is learning $z = f_\varphi(x)$ and $\hat{x} = g_\theta(z)$. This does not enable the model to learn any latent representation which can be useful for dimensionality reduction, compression, clustering, or semi-supervised learning.

The solution to this problem is the introduction of a layer between the encoder f_φ and the decoder g_θ as a 'bottleneck' layer, such that it forces the latent representation z by having its dimensionality much smaller than the dimensionality of the input x.

(ii) The AE is not trained for generation since it is trained primarily for self-reconstruction. In the latent representation z space, there are gaps as the model tries to keep similar data mapped to 'clusters' while keeping gap between dissimilar data to achieve good self-reconstruction ability. This gap, however, does not allow it to learn to generate new data.

(b) Creative, LO1, LO2, LO4
If we minimize only the reconstruction loss, the encoder in VAE can learn to predict means that are arbitrarily far. They can take values outside the area covered by the 'prior'. In such case, the 'ideal' standard deviations for reconstruction loss are zero for every x.

(c) Bookwork, LO1, LO2, LO4

(i) The current loss function assigns a high loss value $-\log(1 - D_\varphi(x))$ when the sample x is real and a low loss value $-\log(D_\varphi(G_\theta(z)))$ when the sample z is fake. This is opposite of what it should be doing.

(ii) To correct it, we need to change this loss function to the following form to make it work for discriminator learning:

$$\min_{\varphi} \mathbb{E}_{x\sim p_{data}(x)}\Big[-\log(D_\varphi(x))\Big] + \mathbb{E}_{z\sim p(z)}\Big[-\log\big(1 - D_\varphi(G_\theta(z))\big)\Big]$$