Assignment 2

Ehsan Merrikhi 400101967
github repository

Deep Learning

Dr. Fatemizadeh

November 15, 2024

Deep Learning

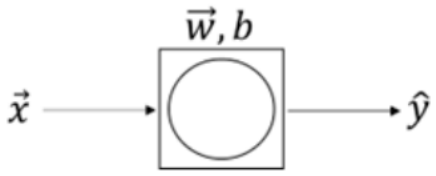
Assignment 2

Ehsan Merrikhi 400101967
github repository

Contents

Question 1 (40 Points)

Assume a single-layer model with only one neuron, as shown below, which has a sigmoid activation function.



For a binary classification problem, we consider the following cost function:

$$E_{w,b} = -\sum_n y_n \ln \hat{y}(x_n) + (1 - y_n) \ln(1 - \hat{y}(x_n))$$

Show that the cost function has a minimum. Then, write a backpropagation relation to reach the minimum point.

Soloution

We need to minimize the cost function $E_{w,b}$ with respect to w, b. $E_{w,b}$ is a convex function.
proof of $E_{w,b}$ convexity:

$$\frac{\partial E_n}{\partial w} = (\frac{1 - y_n}{1 - \hat{y}} - \frac{y_n}{\hat{y}})\frac{\partial \hat{y}}{\partial w} \quad \& \quad \hat{y} = \frac{1}{1 + e^{-(w^T x + b)}} \rightarrow \frac{\partial \hat{y}}{\partial w} = \hat{y}(1 - \hat{y})x$$

$$\implies \frac{\partial E_n}{\partial w} = (\hat{y} - y_n)x$$

$$\frac{\partial^2 E_n}{\partial w \partial w^T} = \frac{\partial}{\partial w}((\hat{y} - y_n)x) = \hat{y}(1 - \hat{y})xx^T \implies \frac{\partial^2 E_{w,b}}{\partial w \partial w^T} = \sum_n \hat{y}(1 - \hat{y})x_n x_n^T$$

The term $\hat{y}(1 - \hat{y})$ is always positive and $x_n x_n^T$ is PSD which means Hessian matrix is PSD and therefore $E_{w,b}$ is a convex.

$$\frac{\partial E_n}{\partial b} = (\frac{1 - y_n}{1 - \hat{y}} - \frac{y_n}{\hat{y}})\frac{\partial \hat{y}}{\partial b} \quad \& \quad \hat{y} = \frac{1}{1 + e^{-(w^T x + b)}} \rightarrow \frac{\partial \hat{y}}{\partial b} = \hat{y}(1 - \hat{y})$$

$$\implies \frac{\partial E_n}{\partial b} = (\hat{y} - y_n)$$

$$\frac{\partial^2 E_n}{\partial^2 b} = \frac{\partial}{\partial b}(\hat{y} - y_n) = \hat{y}(1 - \hat{y}) \implies \frac{\partial^2 E_{w,b}}{\partial^2 b} = \sum_n \hat{y}(1 - \hat{y})$$

The term $\hat{y}(1 - \hat{y})$ is always positive which means Hessian matrix is PSD and therefore $E_{w,b}$ is a convex.
Now we know $E_{w,b}$ is convex and it has a global minimum. to find the minimum of this function we use the algorithm below.

$$w_{n+1} = w_n - \eta \sum_n (\hat{y}(x_n) - y_n)x_n \quad \& \quad b_{n+1} = b_n - \eta \sum_n (\hat{y}(x_n) - y_n)$$

Question 2 (50 Points)

Answer the following questions about Batch Normalization (BN).

a. Describe the covariate shift problem in neural networks and explain how BN solves it.

Soloution

Covariate Shift and Internal Covariate Shift:
Covariate shift occurs when the input data distribution to a neural network changes during training, making learning more challenging. In neural networks, each layer's output serves as the input to the next layer, so as the parameters (weights and biases) in each layer are updated, the input distribution to each subsequent layer changes. This phenomenon, known as **internal covariate shift**, slows down training and disrupts learning stability by requiring the network to adapt to constantly changing distributions.
How Batch Normalization (BN) Addresses Internal Covariate Shift:

(a) **Normalization:** For each mini-batch, BN calculates the mean and variance of the inputs to a layer, normalizing these inputs to have zero mean and unit variance. This stabilization helps maintain a consistent distribution of inputs for each layer.

(b) **Learnable Scaling and Shifting:** After normalization, BN introduces learnable parameters for scaling and shifting. These parameters allow the model to adjust the normalized values if zero mean and unit variance are not optimal for a specific layer or task.

b. Explain how BN helps with the generalization of the network.

Soloution

Batch Normalization (BN) and Generalization:
Batch Normalization (BN) enhances neural network generalization by introducing several beneficial effects during training:

(a) **Reducing Internal Covariate Shift:** By normalizing inputs to each layer, BN stabilizes their distributions.

(b) **Acting as a Regularizer:** The slight noise from mini-batch statistics has a regularizing effect, reducing overfitting.

(c) **Allowing Higher Learning Rates:** BN enables the use of higher learning rates, accelerating training.

(d) **Reducing the Need for Strong Regularization:** BN's regularizing effect means that dropout or strong L2 regularization can be reduced or skipped, simplifying the model while retaining important features.

(e) **Smoothing the Loss Landscape:** BN smooths the optimization path, leading to flatter minima, which are associated with better generalization on unseen data.

c. Consider a simplified BN where we only consider the data without dividing by the standard deviation, meaning we simply use the mean for input values $\mu = \frac{1}{n}\sum_{j=1}^{n} x_j$, and define $\hat{x}_i = x_i - \mu$, where $[x_1, x_2, \ldots, x_n]$ are the input data (with n as the number of data in the mini-batch). The outputs are also $[y_1, y_2, \ldots, y_n]$, which is $y_i = \gamma\hat{x}_i + \beta$ assume the cost function at the end of neural network defined as L $\frac{\partial L}{\partial x_i}$ based on $\frac{\partial L}{\partial y_j}$ for $j = 1, \ldots, n$.

Soloution

To solve this problem, let's calculate the gradient $\frac{\partial L}{\partial x_i}$ in terms of $\frac{\partial L}{\partial y_j}$.
Problem Recap
We have:
- A simplified Batch Normalization (BN) where we only use the mean, not the standard deviation, to normalize.
- The mean of inputs $x_1, x_2, \ldots, x_n$ is:

$$\mu = \frac{1}{n}\sum_{j=1}^{n} x_j$$

- We define the normalized input as:

$$\hat{x}_i = x_i - \mu$$

- The outputs $y_1, y_2, \ldots, y_n$ are linear functions of $\hat{x}_i$, meaning each y_i depends on $\hat{x}_i$.
1. **Express $\hat{x}_i$ in terms of x_i:**

$$\hat{x}_i = x_i - \mu = x_i - \frac{1}{n}\sum_{j=1}^{n} x_j$$

2. **Compute $\frac{\partial \hat{x}_i}{\partial x_k}$:**
- For $i = k$:

$$\frac{\partial \hat{x}_i}{\partial x_i} = 1 - \frac{1}{n}$$

- For $i \neq k$:

$$\frac{\partial \hat{x}_i}{\partial x_k} = -\frac{1}{n}$$

3. **Express $\frac{\partial L}{\partial x_i}$ using the chain rule:**
By the chain rule:

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial \tilde{x}_i}.1 + \sum_{k=1}^{n} \frac{\partial L}{\partial \tilde{x}_k} - \frac{1}{n} \quad \& \quad \frac{\partial L}{\partial \tilde{x}_k} = \frac{\partial L}{\partial y_i}.\gamma$$

$$\implies \frac{\partial L}{\partial x_i} = \gamma(\frac{\partial L}{\partial y_i} - \frac{1}{n}\sum_{k=1}^{n} \frac{\partial L}{\partial y_k})$$

Since each y_j is a linear function of $\hat{x}_j$, we can express the result based on this relationship.

d. In part (c), calculate the value of $\frac{\partial L}{\partial x_i}$ for the two cases $n \to \infty$ and $n = 1$. What result do you obtain?

Soloution

To calculate $\frac{\partial L}{\partial x_i}$ for the two cases $n \to \infty$ and $n = 1$, let's analyze each case separately.
Case 1: $n = 1$ When $n = 1$, there is only a single data point in the mini-batch. Therefore, the mean μ of the input data is simply x_1 itself, so we have:

$$\mu = x_1.$$

This implies that $\hat{x}_i = x_i - \mu = x_1 - x_1 = 0$. As a result, y_1 will be independent of x_1, and thus:

$$\frac{\partial L}{\partial x_1} = 0.$$

Case 2: $n \to \infty$ When $n \to \infty$, the mean μ is computed over a very large number of inputs, and each individual x_i has a negligible effect on μ. Therefore, in this limit, $\frac{\partial \mu}{\partial x_i} \approx 0$, and $\hat{x}_i \approx x_i - \mu$ acts as if μ is constant.
In this case, we calculate $\frac{\partial L}{\partial x_i}$ as:

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial y_i} \cdot \frac{\partial y_i}{\partial \hat{x}_i} \cdot \frac{\partial \hat{x}_i}{\partial x_i} = \frac{\partial L}{\partial y_i} \cdot \gamma \cdot 1. =$$

Thus, we get:

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial y_i} \cdot \frac{\partial y_i}{\partial \hat{x}_i}.$$

Result - For $n = 1$, $\frac{\partial L}{\partial x_i} = 0$. - For $n \to \infty$, $\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial y_i} \cdot \gamma$, which indicates that the effect of normalization vanishes as n becomes very large.

Question 3 (50 Points)

Consider a two-layer neural network for classifying K classes with the following relationships:

Let the input x be in the form of a vector with dimension $d_x \times 1$, and the labels $y \in \{0,1\}^K$ are one-hot encoded. Also, the hidden layer has d_a neurons.

$$z^{(1)} = W^{(1)}x + b^{(1)}$$
$$\hat{a}^{(1)} = LeakyReLU(z^{(1)}), \quad \alpha = 0.01$$
$$a^{(1)} = Dropout(\hat{a}^{(1)}, p = 0.2)$$
$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}$$
$$\hat{y} = softmax(z^{(2)})$$
$$L = \sum_{i=1}^{K} -y_i \log(\hat{y}_i)$$

1. Calculate $\frac{\partial \hat{y}_k}{\partial z_i^{(2)}}$ and simplify your answer based on $\hat{y}$.

Soloution

$$\hat{y} = softmax(z^{(2)}) = \frac{e^{z_k^{(2)}}}{\sum_{i=1}^{K} e^{z_i^{(2)}}}$$

for $i = k$:

$$\implies \frac{\partial \hat{y}_k}{\partial z_k^{(2)}} = \frac{e^{z_k^{(2)}} \sum_{i=1}^{K} e^{z_i^{(2)}}}{(\sum_{i=1}^{K} e^{z_i^{(2)}})^2} - \frac{e^{z_i^{(2)}}}{(\sum_{i=1}^{K} e^{z_i^{(2)}})^2} = \frac{e^{z_k^{(2)}}}{\sum_{i=1}^{K} e^{z_i^{(2)}}} \frac{\sum_{i=1}^{K} e^{z_i^{(2)}} - e^{z_k^{(2)}}}{\sum_{i=1}^{K} e^{z_i^{(2)}}} = \hat{y}_k(1-\hat{y}_k)$$

for $i \neq k$:

$$\implies \frac{\partial \hat{y}_k}{\partial z_j^{(2)}} = \frac{-e^{z_k^{(2)}} e^{z_j^{(2)}}}{(\sum_{i=1}^{K} e^{z_i^{(2)}})^2} = -\frac{e^{z_j^{(2)}}}{\sum_{i=1}^{K} e^{z_i^{(2)}}} \frac{e^{z_k^{(2)}}}{\sum_{i=1}^{K} e^{z_i^{(2)}}} = -\hat{y}_k \hat{y}_j$$

2. Assume the true label vector y is one-hot encoded with $y_k = 1$ and the remaining values are zero ($y_k = 1, y_i = 0, \forall i \neq k$). Calculate $\frac{\partial L}{\partial z_i^{(2)}}$ and simplify your answer based on $\hat{y}$.

Soloution

$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \overset{\text{one-hot}}{=} -\log(\hat{y}_k) \quad \& \quad \hat{y}_i = \frac{e^{z_i^{(2)}}}{\sum_{j=1}^{K} e^{z_j^{(2)}}}$$

$$\frac{\partial L}{\partial z_i^{(2)}} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_i^{(2)}}$$

Since $L = -\log(\hat{y}_k)$, we have:

$$\frac{\partial L}{\partial \hat{y}_i} = \begin{cases} -\frac{1}{\hat{y}_k} & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

Differentiate $\hat{y}_i$ with Respect to $z_j^{(2)}$
1. **If $i = j$:**

$$\frac{\partial \hat{y}_i}{\partial z_i^{(2)}} = \hat{y}_i(1 - \hat{y}_i)$$

2. **If $i \neq j$:**

$$\frac{\partial \hat{y}_i}{\partial z_j^{(2)}} = -\hat{y}_i \hat{y}_j$$

Combine Results
1. for $i = k$:

$$\frac{\partial L}{\partial z_k^{(2)}} = -\frac{1}{\hat{y}_k} \cdot \hat{y}_k(1 - \hat{y}_k) = \hat{y}_k - 1$$

2. for $i \neq k$:

$$\frac{\partial L}{\partial z_i^{(2)}} = -\frac{1}{\hat{y}_k} \cdot (-\hat{y}_i \hat{y}_k) = \hat{y}_i$$

3. Calculate $\frac{\partial L}{\partial W^{(1)}}$.

Soloution

$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \quad \& \quad \frac{\partial L}{\partial z^{(2)}} = \hat{y} - y$$

$$z^{(2)} = W^{(2)} a^{(1)} + b^{(2)} \implies \frac{\partial z^{(2)}}{\partial a^{(1)}} = \left(W^{(2)}\right)^T$$

$$a^{(1)} = Dropout(\hat{a}^{(1)}, p = 0.2) \implies \frac{\partial a^(1)}{\partial \hat{a}^{(1)}} = \cdots \odot M \begin{cases} 1, & \text{with probability} 1 - p \\ 0, & \text{with probability} p \end{cases}$$

$$\hat{a}^{(1)} = LeakyReLU(z^{(1)}), \quad \alpha = 0.01 \implies \frac{\partial \hat{a}^{(1)}}{\partial z_i^{(1)}} = A = \begin{cases} 1, & z_i^{(1)} > 0 \\ 0.01 & z_i^{(1)} \leq 0 \end{cases}$$

$$z^{(1)} = W^{(1)} x + b^{(1)} \implies \frac{\partial z^{(1)}}{\partial W^{(1)}} = x^T$$

Conclusion:

$$\implies \frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial \hat{a}^{(1)}} \frac{\partial \hat{a}^{(1)}}{\partial z_i^{(1)}} \frac{\partial z^{(1)}}{\partial W^{(1)}} = ((\hat{y} - y)(W^{(2)})^T \odot M \odot A) x^T$$

Question 4 (40 Points)

Show that the Hessian matrix of a transformation like $\psi(u, v, z) = y(u, v, z)$ can be written in terms of the Jacobian matrix of the gradient of this transformation. Note that the variables u, v, and z are one dimensional, and y is a function of those variables.

Soloution

Gradient of y:

$$\nabla y = \begin{bmatrix} \frac{\partial y}{\partial z} \\ \frac{\partial y}{\partial v} \\ \frac{\partial y}{\partial u} \end{bmatrix}$$

where u, v, and z are scalar variables.

Jacobian of ∇y:

$$J(\nabla y) = \begin{bmatrix} \frac{\partial^2 y}{\partial y^2} & \frac{\partial^2 y}{\partial y \partial v} & \frac{\partial^2 y}{\partial y \partial z} \\ \frac{\partial^2 y}{\partial v \partial u} & \frac{\partial^2 y}{\partial y^2} & \frac{\partial^2 y}{\partial v \partial z} \\ \frac{\partial^2 y}{\partial z \partial u} & \frac{\partial^2 y}{\partial z \partial v} & \frac{\partial^2 y}{\partial z^2} \end{bmatrix}$$

Calculate the Hessian matrix H of scalar function $\psi(u, v, z)$

$$H = \begin{bmatrix} \frac{\partial^2 \psi}{\partial y^2} & \frac{\partial^2 \psi}{\partial y \partial v} & \frac{\partial^2 \psi}{\partial y \partial z} \\ \frac{\partial^2 \psi}{\partial v \partial u} & \frac{\partial^2 \psi}{\partial y^2} & \frac{\partial^2 \psi}{\partial v \partial z} \\ \frac{\partial^2 \psi}{\partial z \partial u} & \frac{\partial^2 \psi}{\partial z \partial v} & \frac{\partial^2 \psi}{\partial z^2} \end{bmatrix}$$

Comparing these two matrices we can see the hessian of the transformation can be expressed as the Jacobian of the gradient vector of y.

Question 5. (40 points)

The error function in a network with applying Gaussian Dropout is described as follows.

$$J_1 = 0.5 \left(y_d - \sum_{k=1}^{n} \delta_k W_k x_k \right)^2$$

where $\delta_k \sim \text{Normal}(1, \sigma^2)$. Calculate the mathematical expectation of the function's gradient with respect to the variable W_i and simplify as much as possible.

Soloution

$$\frac{\partial J_1}{\partial W_i} = \frac{\partial}{\partial W_i} \left[0.5 \left(y_d - \sum_{k=1}^{n} \delta_k W_k x_k \right)^2 \right]$$

Using the chain rule:

$$\frac{\partial J_1}{\partial W_i} = \left(y_d - \sum_{k=1}^{n} \delta_k W_k x_k \right) \cdot (-\delta_i x_i)$$

2. Expected Gradient
Now, we take the expectation of the gradient with respect to δ_i.

$$\delta_i \sim \text{Normal}(1, \sigma^2) \implies \mathbb{E}[\delta_i] = 1$$

$$\mathbb{E}\left[\frac{\partial J_1}{\partial W_i} \right] = -\mathbb{E}\left[\delta_i x_i y_d - \sum_{k=1}^{n} \delta_i x_i \delta_k W_k x_k \right] = -\mathbb{E}\left[\delta_i\right] x_i y_d + x_i \sum_{k=1}^{n} \mathbb{E}\left[\delta_i \delta_k\right] W_k x_k$$

$$\mathbb{E}\left[\delta_i = 1\right] \quad \& \quad \mathbb{E}\left[\delta_i \delta_k\right] = \begin{cases} \mathbb{E}\left[\delta_i\right]^2 + Var(\delta_i) = 1 + \sigma^2, & i = k \\ \mathbb{E}\left[\delta_i\right] \mathbb{E}\left[\delta_k\right] = 1, & \text{otherwise} \end{cases}$$

$$\implies \mathbb{E}\left[\frac{\partial J_1}{\partial W_i} \right] = -x_i y_d + x_i((1+\sigma^2) W_i x_i + \sum_{k=1, k\neq i}^{n} W_k x_k)$$

$$= -x_i(y_d - (1+\sigma^2) W_i x_i - \sum_{k=1, k\neq i}^{n} W_k x_k)$$

Can you offer and interpretation of Regularization with this type of Dropout? If so, introduce the Non-Regularized target function.

Soloution

The non regularizied function looks like:

$$J_{1\text{non-reg}} = 0.5 \left(y_d - \sum_{k=1}^{n} W_k x_k \right)^2$$

Using the chain rule:

$$\frac{\partial J_{1\text{non-reg}}}{\partial W_i} = (-x_i) \left(y_d - \sum_{k=1}^{n} W_k x_k \right) = (-x_i) \left(y_d - W_i x_i - \sum_{k=1, k\neq i}^{n} W_k x_k \right)$$

Comparing the gradient of $J_{1\text{non-reg}}$ with J_1 we see the term $-\sigma^2 W_i x_i$.
This means we are adding some noise to the W_i in the training stage.
We conclude Dropout is added noise to weights matrix in the training stage.

Question 6. (40 points)

Prove the convergence of the Newton method for the function $f(x) = g'(x)$. Assume that f is twice differentiable and continuous at the optimal point x^*, where $f(x^*) = 0$ and $f'(x^*) \neq 0$. Then, show that the Second Order Newton method, which finds x^* as the optimal point of f, also converges for the optimal point of the function g.

Soloution

define the error $e_k = x_k - x^* \rightarrow x^* = x_k - e_k$

$$f(x) = g'(x) \rightarrow x_{k+1} = x_k - \frac{g'(x)}{g''(x)} = x_k - \frac{f(x)}{f'(x)}$$

$$f(x^*) = f(x_k - e_k) \simeq f(x_k) - e_k f'(x_k) + \frac{e_k^2}{2} f''(s_k) \text{for some } x_k < s_k < x^*$$

$$f(x^*) = 0 \rightarrow f(x_k) - e_k f'(x_k) + \frac{e_k^2}{2} f''(s_k) = 0 \rightarrow \frac{f(x_k)}{f'(x_k)} - e_k + \frac{e_k^2}{2f'(x_k)} f''(s_k)$$

$$\rightarrow x_k - x_{k-1} - x_k + x^* + \frac{e_k^2}{2f'(x_k)} f''(s_k) = 0$$

$$\rightarrow x_{k-1} - x^* = \frac{(x_k - x^*)^2}{2f'(x_k)} f''(s_k) \implies |x_{k-1} - x^*| \leq \frac{(x_k - x^*)^2}{2|f'(x_k)|} |f''(s_k)|$$

This means for $k \rightarrow \infty$ we have $x_k \rightarrow x^*$ and $f(x_k) \rightarrow f(x^*)$.

Question 7. (40 points)

Consider a neural network for multiclass classification with K classes, where we use the **softmax** function in the output layer. The input to the softmax layer is a vector $\mathbf{z} = [z_1, z_2, \ldots, z_K]^\top \in \mathbb{R}^K$, and the softmax output is defined as:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i = 1, 2, \ldots, K.$$

The true labels of the classes are represented as a one-hot vector $\mathbf{y} = [y_1, y_2, \ldots, y_K]^\top$, where $y_k = 1$ if the k-th class is the correct class, and $y_k = 0$ otherwise. The cross-entropy loss function is defined as:

$$L(\mathbf{z}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k.$$

1. **(a)** Prove that the gradient of the loss function with respect to $\mathbf{z}$ is given by:

$$\nabla_{\mathbf{z}} L = \hat{\mathbf{y}} - \mathbf{y}.$$

Soloution

$$\nabla_{\mathbf{z}} L(\mathbf{z}, \mathbf{y}) = \nabla_{\mathbf{z}} \left(-\sum_{k=1}^{K} y_k \log \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \right) = \nabla_{\mathbf{z}} \left(-\sum_{k=1}^{K} y_k z_i + \sum_{k=1}^{K} y_k \left(\log \sum_{j=1}^{K} e^{z_j} \right) \right)$$

$$\frac{\partial}{\partial z_i} \left(-\sum_{k=1}^{K} y_k z_i + \sum_{k=1}^{K} y_k \left(\log \sum_{j=1}^{K} e^{z_j} \right) \right) = -\sum_{k=1}^{K} y_k + \sum_{k=1}^{K} y_k \left(\frac{z_i}{\sum_{j=1}^{K} e^{z_j}} \right) = \hat{y}_i - 1$$

$$\implies \nabla_{\mathbf{z}} L(\mathbf{z}, \mathbf{y}) = \hat{y} - y$$

2. **(b) Matrix H is symmetric and positive semi-definite:**

   (i) Calculate the Hessian matrix $H \in \mathbb{R}^{K \times K}$ of the loss function with respect to $\mathbf{z}$.

Soloution

we know $\frac{\partial}{\partial z_i} L(\mathbf{z}, \mathbf{y}) = \hat{y}_i - 1$.

$$\rightarrow \frac{\partial^2}{\partial z_i^2} L(\mathbf{z}, \mathbf{y}) = \frac{\partial}{\partial z_i} (\hat{y}_i - 1) = \frac{\partial}{\partial z_i} \left(\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} - 1 \right) = \hat{y}_i (1 - \hat{y}_i)$$

$$\rightarrow \frac{\partial^2}{\partial z_i \partial z_j} L(\mathbf{z}, \mathbf{y}) = \frac{\partial}{\partial z_j} (\hat{y}_i - 1) = \frac{\partial}{\partial z_j} \left(\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} - 1 \right) = -\hat{y}_i \hat{y}_j$$

$$\implies H = \begin{bmatrix} \hat{y}_1(1 - \hat{y}_1) & -\hat{y}_1 \hat{y}_2 & \ldots & -\hat{y}_1 \hat{y}_K \\ -\hat{y}_2 \hat{y}_1 & \hat{y}_2(1 - \hat{y}_2) & \ldots & -\hat{y}_2 \hat{y}_K \\ \vdots & \vdots & \ddots & \vdots \\ -\hat{y}_K \hat{y}_1 & -\hat{y}_K \hat{y}_2 & \ldots & \hat{y}_K(1 - \hat{y}_K) \end{bmatrix}.$$

(ii) Prove that H is positive semi-definite.

Soloution

We know $H = diag(\hat{y}) - \hat{y}\hat{y}^T$ so we can calculate $x^T H x$.

$$x^T diag(\hat{y})x - x^T \hat{y}\hat{y}^T x = \sum_{i=1}^{K} \hat{y}_i x_i^2 - (\hat{y}^T x)^2$$

If we can show $(\hat{y}^T x)^2 < \sum_{i=1}^{K} \hat{y}_i x_i^2$ then we know Hessian matrix is PSD.
From Cauchy-Schwarz inequality we know $(a^T b)^2 \leq (a^T a)(b^T b)$

$$(\hat{y}^T x)^2 = (\sum_{i=1}^{K} \hat{y}_i x_i)^2 \quad \& \quad \sum_{i=1}^{K} \hat{y}_i x_i^2 = \left(\sum_{i=1}^{K} \hat{y}_i\right)\left(\sum_{i=1}^{K} \hat{y}_i x_i^2\right)$$

$$\implies (\sum_{i=1}^{K} \hat{y}_i x_i)^2 \leq \left(\sum_{i=1}^{K} \hat{y}_i\right)\left(\sum_{i=1}^{K} \hat{y}_i x_i^2\right)$$

This means $\sum_{i=1}^{K} \hat{y}_i x_i^2 - (\hat{y}^T x)^2$ is always positive and Hessian is PSD.

(iii) Using the result from part (ii), conclude whether the cross-entropy loss function $L(\mathbf{z}, \mathbf{y})$ is convex with respect to $\mathbf{z}$ or not.

Soloution

From part (*ii*) we know the Hessian matrix is PSD.
In the context of convexity we know the function is convex if and only if its Hessian matrix is PSD.