Deep learning

DR.Fatemi Zadeh

electrical engineering department

Ahmadreza Majlesara 400101861

computer assignment repository

assignment 2

November 13, 2024

Deep learning

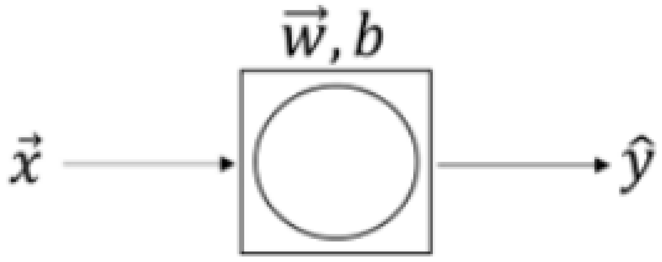
assignment 2

Ahmadreza Majlesara 400101861

computer assignment repository

Question 1 (40 points)

Suppose we have a single-layer model with only one neuron as shown in the figure below, with a sigmoid activation function.



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

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

Show that the cost function has a minimum. Then, write an equation to reach the minimum point.

solution

The cost function $E_{w,b}$ represents the binary cross-entropy loss, which is a convex function with respect to the parameters w and b. to prove the convexity of the function, we need to show that the Hessian matrix of the function is positive semi-definite. we have:

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

where $y_n \in \{0, 1\}$ is the true label, and $\hat{y}(x_n) = \sigma(w^T x_n + b)$ is the prediction, with $\sigma(z) = \frac{1}{1+e^{-z}}$ being the sigmoid function.

$$\hat{y} = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

$$E_n = -[y_n \ln \hat{y} + (1 - y_n) \ln(1 - \hat{y})]$$

now we can calculate first and second derivatives of the cost function with respect to the parameters w and b to find the minimum point.

$$\frac{\partial \hat{y}}{\partial w} = \hat{y}(1-\hat{y})x$$

$$\frac{\partial E_n}{\partial w} = -\left(\frac{y_n}{\hat{y}} - \frac{1-y_n}{1-\hat{y}}\right)\frac{\partial \hat{y}}{\partial w}$$

Substituting $\frac{\partial \hat{y}}{\partial w} = \hat{y}(1-\hat{y})x$:

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

The second derivative (Hessian) of E_n with respect to w is:

$$\frac{\partial^2 E_n}{\partial w \partial w^T} = \frac{\partial}{\partial w}\left((\hat{y}-y_n)x\right)$$

Since $\hat{y} = \sigma(w^T x + b)$, taking the derivative of $\hat{y}-y_n$ with respect to w gives:

$$\frac{\partial^2 E_n}{\partial w \partial w^T} = \hat{y}(1-\hat{y})xx^T$$

The Hessian matrix of the cost function is:

$$H = \sum_n \hat{y}(1-\hat{y})x_n x_n^T$$

since the term $\hat{y}(1-\hat{y})$ is always positive and $x_n x_n^T$ is a positive semi-definite matrix, the Hessian matrix is positive semi-definite and the cost function is convex in respect to the parameters w. now we do the same for the parameter b:

$$\frac{\partial \hat{y}}{\partial b} = \hat{y}(1-\hat{y})$$

$$\Rightarrow \frac{\partial E_{w,b}}{\partial b} = -\sum_n \left[\frac{y_n}{\hat{y}} - \frac{1-y_n}{1-\hat{y}}\right]\frac{\partial \hat{y}}{\partial b}$$

Substituting $\frac{\partial \hat{y}}{\partial b} = \hat{y}(1-\hat{y})$:

$$\frac{\partial E_{w,b}}{\partial b} = \sum_n (\hat{y}-y_n)$$

Now, let's compute the second derivative of $E_{w,b}$ with respect to b.

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

Since $\hat{y} = \sigma(w^T x_n + b)$, we have:

$$\frac{\partial^2 E_{w,b}}{\partial b^2} = \sum_n \frac{\partial \hat{y}}{\partial b}$$

Using $\frac{\partial \hat{y}}{\partial b} = \hat{y}(1-\hat{y})$:

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

The Hessian matrix of the cost function is:

$$H = \sum_n \hat{y}(1-\hat{y})$$

Since the term $\hat{y}(1-\hat{y})$ is always positive, the Hessian matrix is positive semi-definite and the cost function is convex in respect to the parameter b. Therefore, the cost function has a minimum point. To reach the minimum point, we can use the gradient descent algorithm to update the parameters w and b iteratively:

$$w \leftarrow w - \eta \sum_n (\hat{y}(x_n) - y_n) x_n$$

$$b \leftarrow b - \eta \sum_n (\hat{y}(x_n) - y_n)$$

Question 2 (50 points)

Answer the following questions regarding Batch Normalization (BN).

part 1

Describe the problem of covariate shift in neural networks and explain how BN addresses it.

solution

In neural networks, covariate shift happens when the parameters of previous layers are changed during training, causing a change in the distribution of inputs to a layer. Each layer must constantly adjust as a result of this change, which slows training and may cause instability.
By normalizing each layer's input to have a constant mean and variance across mini-batches, Batch Normalization (BN) addresses covariate shift. Training becomes quicker and more stable as BN lessens the effect of covariate change by standardizing these inputs.

part 2

Explain how BN helps in the generalization of the network.

solution

Batch Normalization (BN) acts as a regularizer, improving neural network generalization. BN normalizes the inputs within each mini-batch during training, which introduces minor noise as a result of batch statistics variation. This noise works similarly to dropout, adding unpredictability to keep the network from overfitting to the training data. As a result, the network learns more robust characteristics that can generalize to previously unseen data. Furthermore, BN enables the network to use greater learning rates, which might result in faster convergence and better exploration of the solution space, hence promoting generalization.

part 3

Consider a simple BN where we normalize the data without dividing by the standard deviation, meaning we only center the inputs x_i as follows:

$$\mu = \frac{1}{n}\sum_{j=1}^{n} x_j \quad \text{so that} \quad \tilde{x}_i = x_i - \mu$$

where $[x, x_1, \ldots, x_n]$ are the inputs in a mini-batch of size n, and the outputs $[y, y_1, \ldots, y_n]$ are the outputs which $y_i = \gamma \hat{x}_i + \beta$. Assume at the end of a deep network, a cost function L is defined. Compute $\frac{\partial L}{\partial x_i}$ in term of $\frac{\partial L}{\partial y_j}$ for $j = 1, \ldots, n$.

solution

Since $y_j = \gamma \tilde{x}_j + \beta$, we have:

$$\frac{\partial L}{\partial \tilde{x}_i} = \sum_{j=1}^{n} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial \tilde{x}_i}.$$

Now, calculate $\frac{\partial y_j}{\partial \tilde{x}_i}$:

$$\frac{\partial y_j}{\partial \tilde{x}_i} = \begin{cases} \gamma, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

$$\Rightarrow \frac{\partial L}{\partial \tilde{x}_i} = \frac{\partial L}{\partial y_i} \cdot \gamma.$$

$$\Rightarrow \frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial \tilde{x}_i} \frac{\partial \tilde{x}_i}{\partial x_i} + \sum_{k=1}^{n} \frac{\partial L}{\partial \tilde{x}_k} \frac{\partial \tilde{x}_k}{\partial x_i}.$$

since $\tilde{x}_i = x_i - \mu$, we have:

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

and for $k \neq i$:

$$\frac{\partial \tilde{x}_k}{\partial x_i} = -\frac{1}{n}.$$

so we can rewrite the equation as:

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

Now, substitute $\frac{\partial L}{\partial \tilde{x}_k} = \frac{\partial L}{\partial y_k} \cdot \gamma$ for each k:

$$\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial y_i} \cdot \gamma \left(1 - \frac{1}{n}\right) - \sum_{k \neq i} \frac{\partial L}{\partial y_k} \cdot \gamma \cdot \frac{1}{n}.$$

so if we factor out γ from the equation, we get:

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

part 4

In part (3), find $\frac{\partial L}{\partial x_i}$ for the two cases $n \to \infty$ and $n = 1$. What result do you obtain?

solution

From the previous part, we have:

$$\frac{\partial L}{\partial x_i} = \gamma\left(\frac{\partial L}{\partial y_i} - \frac{1}{n}\sum_{j=1}^{n}\frac{\partial L}{\partial y_j}\right).$$

so for $n \to \infty$:

$$\frac{\partial L}{\partial x_i} = \gamma\left(\frac{\partial L}{\partial y_i} - \frac{1}{\infty}\sum_{j=1}^{\infty}\frac{\partial L}{\partial y_j}\right) = \gamma\left(\frac{\partial L}{\partial y_i} - 0\right) = \gamma\frac{\partial L}{\partial y_i}.$$

As $n \to \infty$, the term $\frac{1}{n}\sum_{j=1}^{n}\frac{\partial L}{\partial y_j}$ becomes the average of $\frac{\partial L}{\partial y_j}$ over all of the data.so the Normalization term is removed, and the gradient is scaled by γ.(which is the same as the case without normalization)
for $n = 1$:

$$\frac{\partial L}{\partial x_i} = \gamma\left(\frac{\partial L}{\partial y_i} - \frac{1}{1}\sum_{j=1}^{1}\frac{\partial L}{\partial y_j}\right) = \gamma\left(\frac{\partial L}{\partial y_i} - \frac{\partial L}{\partial y_i}\right) = 0.$$

When $n = 1$, there is only one element in the batch. Thus, we have $\mu = x_1$, so $\tilde{x}_1 = x_1 - \mu = 0$. In this case, the output $y_1 = \gamma \cdot 0 + \beta = \beta$, which is constant and does not depend on x_1.
Since y_1 does not depend on x_1, $\frac{\partial L}{\partial x_1} = 0$. This means that the gradient is zero, and the network does not learn anything from the data.

Question 3 (50 points)

Consider a two-layer neural network for K-class classification with the following relationships. The input x has a dimension of d_x, and the output $y \in \{0,1\}^K$ is in one-hot encoded format. The hidden layer has d_a neurons.

$$z^{(1)} = W^{(1)}x + b^{(1)}$$
$$\hat{a}^{(1)} = \text{LeakyReLU}(z^{(1)}, \alpha = 0.1)$$
$$a^{(1)} = \text{Dropout}(\hat{a}^{(1)}, p = 0.2)$$
$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}$$
$$\hat{y} = \text{softmax}(z^{(2)})$$
$$L = \sum_{i=1}^{K} -y_i \log(\hat{y}_i)$$

part 1

Compute $\frac{\partial \hat{y}_k}{\partial z_i^{(2)}}$and simplify your answer in term of $\hat{y}$.

solution

we know that the softmax function is defined as:

$$\hat{y}_k = \frac{e^{z_k^{(2)}}}{\sum_{j=1}^{K} e^{z_j^{(2)}}}$$

we want to compute the derivative of $\hat{y}_k$ with respect to $z_i^{(2)}$. we have to find derivative in two cases: in case $i = k$, we have:

$$\frac{\partial \hat{y}_k}{\partial z_k^{(2)}} = \frac{\partial}{\partial z_k^{(2)}} \left(\frac{e^{z_k^{(2)}}}{\sum_{j=1}^{K} e^{z_j^{(2)}}} \right)$$

so we can write:

$$\frac{\partial \hat{y}_k}{\partial z_k^{(2)}} = \frac{e^{z_k^{(2)}} \sum_{j=1}^{K} e^{z_j^{(2)}} - e^{z_k^{(2)}} 2}{\left(\sum_{j=1}^{K} e^{z_j^{(2)}} \right)^2}$$

we can simplify the above equation as:

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

in case $i \neq k$, we have:

$$\frac{\partial \hat{y}_k}{\partial z_i^{(2)}} = \frac{\partial}{\partial z_i^{(2)}} \left(\frac{e^{z_k^{(2)}}}{\sum_{j=1}^{K} e^{z_j^{(2)}}} \right)$$

we can write:

$$\frac{\partial \hat{y}_k}{\partial z_i^{(2)}} = \frac{-e^{z_k^{(2)}} e^{z_i^{(2)}}}{\left(\sum_{j=1}^{K} e^{z_j^{(2)}} \right)^2}$$

we can simplify the above equation as:

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

so the final answer is:

$$\frac{\partial \hat{y}_k}{\partial z_i^{(2)}} = \begin{cases} \hat{y}_k (1 - \hat{y}_k) & \text{if } i = k \\ -\hat{y}_k \hat{y}_i & \text{if } i \neq k \end{cases}$$

part 2

Assume that the vector y consists of all zeros except one component at k, which is equal to 1,(i.e., $y_k = 1$ and $y_i = 0$ for $i \neq k$). Compute $\frac{\partial L}{\partial z^{(2)}}$ and simplify your answer in terms of $\hat{y}$.

solution

Given that y is a one-hot encoded vector, we have $y_k = 1$ for a specific class k and $y_i = 0$ for all $i \neq k$. The loss function L for this classification problem, using cross-entropy, is:

$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i).$$

so we can write:

$$L = -\log(\hat{y}_k)$$

we want to compute the derivative of L with respect to $z^{(2)}$. we have:

$$\frac{\partial L}{\partial z^{(2)}} = \frac{\partial}{\partial z^{(2)}} \left(-\log(\hat{y}_k) \right) = -\frac{1}{\hat{y}_k} \frac{\partial \hat{y}_k}{\partial z^{(2)}}$$

we computed $\frac{\partial \hat{y}_k}{\partial z^{(2)}}$ in the previous part, so for the case $i = k$, we have:

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

and for the case $i \neq k$, we have:

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

so we can write the final answer as:

$$\frac{\partial L}{\partial z^{(2)}} = \begin{cases} \hat{y}_k - 1 & \text{if } i = k \\ \hat{y}_i & \text{if } i \neq k \end{cases}$$

part 3

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

solution

To compute $\frac{\partial L}{\partial W^{(1)}}$, we need to expand each part of the gradient step by step. first we compute the gradient with respect to $z^{(2)}$:

$$\frac{\partial L}{\partial z^{(2)}} = \hat{y} - y$$

second we compute the gradient with respect to $a^{(1)}$:

$$\frac{\partial L}{\partial a^{(1)}} = \frac{\partial L}{\partial z^{(2)}} \cdot \left(W^{(2)}\right)^T = (\hat{y} - y) \cdot \left(W^{(2)}\right)^T$$

as we know the Dropout layer is a binary mask, we can write:

$$M = \begin{cases} 1, & \text{with probability } 1-p, \\ 0, & \text{with probability } p. \end{cases}$$

so we can write:

$$\frac{\partial L}{\partial \hat{a}^{(1)}} = (\hat{y} - y) \cdot \left(W^{(2)}\right)^T \circ M$$

where $\circ$ is the element-wise product. now we compute the gradient with respect to $z^{(1)}$:

$$\frac{\partial \hat{a}_i^{(1)}}{\partial z_i^{(1)}} = \begin{cases} 1, & \text{if } z_i^{(1)} > 0, \\ 0.1, & \text{if } z_i^{(1)} \leq 0. \end{cases}$$

if we use the definition of $\mathbb{I}$ we can write LeakyReLU as

$$\mathbb{I}(z^{(1)} > 0)z^{(1)} + 0.1\mathbb{I}(z^{(1)} \leq 0)z^{(1)}$$

so we can write:

$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial \hat{a}^{(1)}} \circ \text{LeakyReLU}'(z^{(1)})$$

Therefore:

$$\frac{\partial L}{\partial z^{(1)}} = \left(\left((\hat{y}-y)\cdot\left(W^{(2)}\right)^T\right)\circ M\right)\circ \text{LeakyReLU}'(z^{(1)})$$

where $\text{LeakyReLU}'(z^{(1)})$ is a vector with elements 1 for positive $z_i^{(1)}$ and 0.1 for non-positive $z_i^{(1)}$. Finally, we compute the gradient with respect to $W^{(1)}$:

$$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial z^{(1)}}\cdot x^T$$

Therefore:

$$\frac{\partial L}{\partial W^{(1)}} = \left(\left((\hat{y}-y)\cdot\left(W^{(2)}\right)^T\right)\circ M\circ \text{LeakyReLU}'(z^{(1)})\right)x^T$$

Question 4 (40 points)

Show that the Hessian matrix of the transformation $y(u,v,z) = \psi(u,v,z)$ can be expressed as the Jacobian matrix of the gradient of this transformation. Note that the variables z, v, and u are 1-dimensional, and y is also a function in terms of them.

solution

The gradient of $y(u,v,z)$ with respect to (u,v,z) is given by:

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

The Hessian matrix $H(y)$ is defined as:

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

The Jacobian matrix of the gradient ∇y with respect to (u,v,z) is obtained by differentiating each component of the gradient vector with respect to u, v, and z. Differentiating $\frac{\partial y}{\partial u}$ with respect to u, v, and z gives the first row of the Hessian:

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

Differentiating $\frac{\partial y}{\partial v}$ with respect to u, v, and z gives the second row of the Hessian:

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

Differentiating $\frac{\partial y}{\partial z}$ with respect to u, v, and z gives the third row of the Hessian:

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

Thus, we can clearly see that:

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

By comparing the two matrices, we see that the Hessian matrix of the transformation $y(u,v,z) = \psi(u,v,z)$ can be expressed as the Jacobian matrix of the gradient of this transformation.

Question 5 (40 points)

The error function in a network with Gaussian Dropout applied is described as follows:

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

where $\delta_k \sim \mathcal{N}(1, \sigma^2)$. Simplify the expected value of the gradient of the objective function with respect to the variable W_i as much as possible.

solution

The gradient of J_1 with respect to W_i is:

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

Now, we compute $\mathbb{E}\left[\frac{\partial J_1}{\partial W_i}\right]$:

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

using the linearity of expectation, we can expand this to:

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

first we compute $\mathbb{E}[\delta_i]$ and $\mathbb{E}[\delta_i \delta_k]$. as it is mentioned in the question, $\delta_i \sim \mathcal{N}(1, \sigma^2)$.so we have:

$$\mathbb{E}[\delta_i] = 1$$

now for computing $\mathbb{E}[\delta_i \delta_k]$, we have two cases. if $i = k$, then:

$$\mathbb{E}[\delta_i \delta_k] = \mathbb{E}\left[\delta_i^2\right] = \mathrm{Var}(\delta_i) + \mathbb{E}[\delta_i]^2 = \sigma^2 + 1.$$

if $i \neq k$, then:

$$\mathbb{E}[\delta_i \delta_k] = \mathbb{E}[\delta_i]\,\mathbb{E}[\delta_k] = 1 \times 1 = 1.$$

the last term comes from the fact that the two random variables are independent. so we have:

$$\mathbb{E}[\delta_i \delta_k] = \begin{cases} \sigma^2 + 1 & \text{if } i = k, \\ 1 & \text{if } i \neq k. \end{cases}$$

Substitute these values back into the expression:

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

Or equivalently:

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

Can you define a form of regularization using this type of Dropout? If so, introduce both Regularized and Non-Regularized targets accordingly.

solution

Without regularization, the objective function is simply the squared error function J_1, which we are trying to minimize. It is given by:

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

When Gaussian Dropout is applied, the objective function becomes as follows which $\delta_k \sim \mathcal{N}(1,\sigma^2)$:

$$J_{1,\text{reg}} = 0.5\mathbb{E}_\delta\left[\left(y_d - \sum_{k=1}^{n} \delta_k W_k x_k\right)^2\right],$$

The expectation in $J_{1,\text{reg}}$ can be expanded as follows:

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

The first term, $0.5\left(y_d - \sum_{k=1}^{n} W_k x_k\right)^2$, is the original (non-regularized) objective function.
The second term, $0.5\sigma^2 \sum_{k=1}^{n} W_k^2 x_k^2$, represents the regularization term added by Gaussian Dropout. Non-Regularized Target can be shown as:

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

and Regularized Target (with Gaussian Dropout) as:

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

Question 6 (40 points)

Prove the convergence of Newton's method for the function $f(x) = g'(x)$. Assume f is twice differentiable and has an optimal point x^* with $f(x^*) = 0$ and $f'(x^*) \neq 0$. Then show that Newton's method can find x^*, which is also the optimal point of the function g, proving the convergence.

solution

For a function $f(x)$, Newton's update rule is given by:

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

we write taylor expansion of $f(x)$ around x^* up to the second order:

$$f(x) \approx f(x^*) + f'(x^*)(x - x^*) + \frac{f''(x^*)}{2}(x - x^*)^2$$

Since $f(x^*) = 0$, this simplifies to:

$$f(x) \approx f'(x^*)(x - x^*) + \frac{f''(x^*)}{2}(x - x^*)^2.$$

now we define $e_k = x_k - x^*$ as the error at iteration k. Substituting $x_k = x^* + e_k$ into the taylor expansion of $f(x)$ gives:

$$f(x_k) \approx f'(x^*)e_k + \frac{f''(x^*)}{2}e_k^2.$$

using this in the Newton's update rule gives:

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

Expanding this gives:

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

Simplifying, we obtain:

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

The above equation shows that:

$$|e_{k+1}| \approx C|e_k|^2,$$

where $C = \left|\frac{f''(x^*)}{2f'(x^*)}\right|$ is a constant. This shows that the error e_k decreases quadratically. Therefore, Newton's method converges to x^*.

Question 7 (40 points)

Consider a neural network for multi-class classification with K classes, where the output layer uses the softmax activation function. The input to the softmax layer is $z = [z_1, z_2, \dots, z_K]^T \in \mathbb{R}^K$, and the output of the softmax layer is defined as follows:

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

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

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

part 1

Prove that the gradient of the loss function with respect to z is given by:

$$\nabla_z L = \hat{y} - y.$$

solution

differentiating $L(z, y)$ with respect to z_i. Substituting $\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$, we get:

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

we can simplify the above equation as:

$$L(z, y) = -\sum_{k=1}^{K} y_k \left(z_k - \log \sum_{j=1}^{K} e^{z_j} \right).$$

now we can differentiate $L(z, y)$ with respect to z_i to get the gradient the derivative of first term is easily calculated. for the second term we have:

$$\frac{\partial}{\partial z_i} \left(\sum_{k=1}^{K} y_k \log \left(\sum_{j=1}^{K} e^{z_j} \right) \right) = \sum_{k=1}^{K} y_k \cdot \frac{1}{\sum_{j=1}^{K} e^{z_j}} \cdot e^{z_i} = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \sum_{k=1}^{K} y_k.$$

Since $\sum_{k=1}^{K} y_k = 1$ (as y is a one-hot vector), this simplifies to:

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

so the gradient of the loss function with respect to z is given by:

$$\frac{\partial L}{\partial z_i} = -y_i + \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} = -y_i + \hat{y}_i.$$

Therefore, the gradient of the loss function with respect to z is $\nabla_z L = \hat{y} - y$.

part 2

The Hessian Matrix and Its Semi-Definiteness:

part 2.1

Compute the Hessian matrix of the loss function $L(z,y)$ with respect to z as $H \in \mathbb{R}^{K\times K}$.

solution

From the previous result, we know that the gradient of L with respect to z is:

$$\nabla_z L = \hat{y} - y.$$

Thus, $\frac{\partial L}{\partial z_i} = \hat{y}_i - y_i$. To find the Hessian H, we need to compute the second derivative of L with respect to z_i and z_j:

$$H_{ij} = \frac{\partial^2 L}{\partial z_i \partial z_j}.$$

We have:

$$\frac{\partial^2 L}{\partial z_i \partial z_j} = \frac{\partial}{\partial z_j}\left(\frac{\partial L}{\partial z_i}\right) = \frac{\partial}{\partial z_j}(\hat{y}_i - y_i) = \frac{\partial \hat{y}_i}{\partial z_j} = \frac{\partial}{\partial z_j}\left(\frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}\right).$$

so we have:

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{e^{z_i}\cdot \delta_{ij}\sum_{k=1}^{K} e^{z_k} - e^{z_i}\cdot e^{z_j}}{\left(\sum_{k=1}^{K} e^{z_k}\right)^2},$$

where δ_{ij} is 1 if $i = j$ and 0 otherwise. so $H_{ij} = \hat{y}_i(\delta_{ij} - \hat{y}_j)$. Therefore, the Hessian matrix H is:

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

in another form we can write H as:

$$H = \operatorname{diag}(\hat{y}) - \hat{y}\hat{y}^T.$$

part 2.2

Prove that the Hessian matrix H is semi-definite.

solution

we calculated the Hessian matrix H as:

$$H = \text{diag}(\hat{y}) - \hat{y}\hat{y}^T$$

this matrix is symmetric, so we only need to show that it is positive semi-definite. Let $x \in \mathbb{R}^K$ be an arbitrary vector. Then:

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

we can rewrite the second term as $(x^T \hat{y})^2$. the first term is always non-negative, and the second term is the square of a real number, so it is also non-negative. to prove that H is positive semi-definite, we need to show that this expression is always non-negative. to do so we use the Cauchy-Schwarz inequality:

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

Therefore, $x^T H x \geq 0$ for all $x \in \mathbb{R}^K$, which implies that H is positive semi-definite.

part 3

Using the result from part (2), determine if the loss function $L(z, y)$ is a convex function with respect to z or not.

solution

From part (2), we know that the Hessian matrix H is positive semi-definite. A function is convex if its Hessian is positive semi-definite. Therefore the loss function $L(z, y)$ is a convex function with respect to z.