Sharif University of Technology
Electrical Engineering Department

Deep Learning
THW2

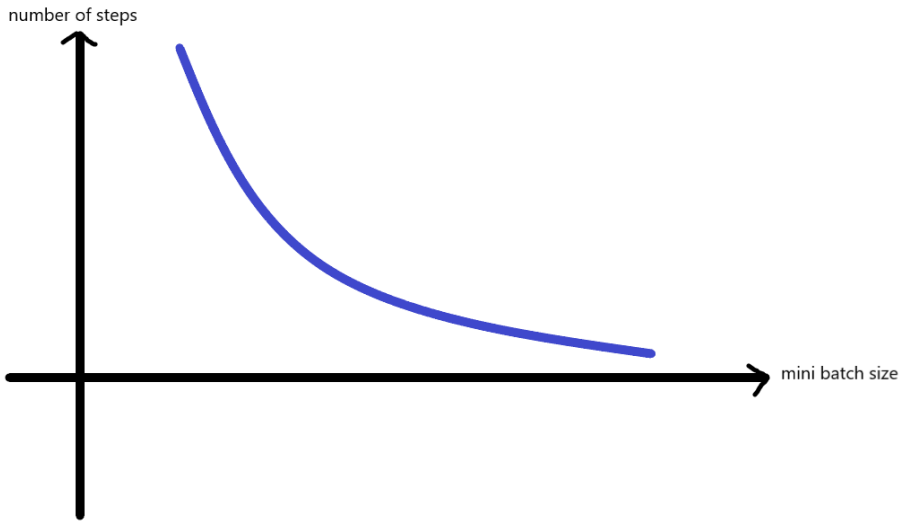
Amir Hossein Yari
99102507

November 19, 2023

Contents

Question 1

a



Small Batch Size:

- Small batch sizes result in noisy gradients because they are computed using a subset of the data.
- This noise can cause the optimization process to oscillate or jump around in the parameter space.
- Despite the noisy updates, small batch sizes allow the model to make frequent updates, adapting quickly to changes in the data, which can be beneficial, especially in the early stages of training.

Large Batch Size:

- Large batch sizes provide more accurate estimates of the gradients since they are computed using more data points, reducing the noise in the updates.
- This reduction in noise helps the optimization process converge more smoothly.
- However, as batch sizes increase, there might be a point of diminishing returns where the computational cost of processing a large batch outweighs the benefits of reduced noise in the updates.

b

Yes, during training in Batch Normalization (BN) layers, a form of noise can be considered to be injected into the activation functions of hidden layers. Batch Normalization introduces a kind of stochastic element during training, which can be thought of as a form of noise.

The main purpose of Batch Normalization is to address issues related to internal covariate shift, which is the change in the distribution of the inputs to a neural network's hidden layers during training. BN normalizes the inputs of each layer by subtracting the batch mean and dividing by the batch standard deviation. Additionally, it introduces learnable parameters (gamma and beta) to scale and shift the normalized values. These learnable parameters provide the network with the flexibility to adapt the normalized values to the specific needs of the layer.

The normalization process helps in reducing the internal covariate shift, making the training more stable and faster. By normalizing the inputs, BN mitigates the vanishing and exploding gradient problems, allowing for more effective training of deep neural networks.

The normalization process in BN also introduces a certain amount of noise because it operates on batches of data, and the statistics (mean and standard deviation) are calculated within each batch. This stochastic element can be considered a form of noise, and it can act as a kind of regularizer during training, similar to dropout or weight regularization. The noise introduced by BN can help the model generalize better to unseen data and prevent overfitting.

In summary, while the primary purpose of Batch Normalization is to normalize and stabilize training, the process of normalizing inputs and introducing learnable parameters can be viewed as injecting a form of noise into the activation functions of hidden layers, contributing to the regularization and improved generalization of the model.

c

No, it might not be suitable for classification tasks.

• Vanishing Gradient Problem: Sigmoid activation functions squash their input values to a range between 0 and 1. During the training process, as the network learns, gradients are propagated backward through the network to update the weights. The sigmoid function has a tendency to saturate, leading to small gradients. When

initialized with large positive weights, the network is more likely to start in a saturated region, and the gradients during backpropagation may become extremely small. This can result in slow or stalled learning, making it difficult for the network to adapt and improve its performance.
• Output Range Issues: Sigmoid functions output values in the range of (0, 1). In a classification task, where the goal is often to predict probabilities, this might not be ideal. The network might struggle to represent highly confident predictions, especially if the true class probability is close to 0 or 1.

d

The total number of trainable parameters in the given neural network is given by:

$$\begin{aligned}\text{Total Params} = &(\text{Input Dimension} + 1) \times \text{Neurons in the First Hidden Layer} \\ &+ 4 \times (\text{Neurons in Hidden Layers} + 1) \times \text{Neurons in Each Hidden Layer} \\ &+ (\text{Neurons in the Last Hidden Layer} + 1) \times \text{Output Dimension}\end{aligned}$$

Plugging in the values:

$$\begin{aligned}\text{Total Params} &= (20 + 1) \times 10 + 4 \times (10 + 1) \times 10 + (10 + 1) \times 1 \\ &= 21 \times 10 + 4 \times 11 \times 10 + 11 \\ &= 210 + 440 + 11 \\ &= 661\end{aligned}$$

e

The fact that the second method has twice the number of parameters compared to the first method does not necessarily imply that it can learn more complex models. In fact, the example you provided suggests that the two methods are closely related. Softmax regression with two neurons is essentially a generalization of logistic regression.
Softmax Regression:

$$\hat{y} = \text{softmax}(W_s x + b_s) = \begin{bmatrix} \frac{e^{w_{s1} \cdot x + b_{s1}}}{Z} \\ \frac{e^{w_{s2} \cdot x + b_{s2}}}{Z} \end{bmatrix}$$

where $Z = e^{w_{s1} \cdot x + b_{s1}} + e^{w_{s2} \cdot x + b_{s2}}$ is the normalization term.

Logistic Regression:

$$\hat{y} = \sigma(W_l x + b_l) = \frac{1}{1 + e^{-(w_l \cdot x + b_l)}}$$

Now, let's define $w_{s1} = w_l$, $w_{s2} = -w_l$, $b_{s1} = b_l$, and $b_{s2} = -b_l$. With these substitutions:

$$\hat{y} = \begin{bmatrix} \frac{e^{w_{s1} \cdot x + b_{s1}}}{Z} \\ \frac{e^{w_{s2} \cdot x + b_{s2}}}{Z} \end{bmatrix} = \begin{bmatrix} \frac{e^{w_l \cdot x + b_l}}{Z} \\ \frac{e^{-w_l \cdot x - b_l}}{Z} \end{bmatrix}$$

Question 2

a

A committee refers to a group of individual models or experts that are combined to make predictions or decisions. In the context of the document, the committee consists of multiple one-hidden-layer neural networks. The purpose of forming a committee is to improve the performance of the model by leveraging the complementary information provided by each individual expert.

By combining the predictions of multiple experts, the committee can achieve better accuracy and recognition rates compared to a single model. The document describes three different methods for combining the predictions of the committee members: majority voting, average, and median committees. Each method has its own way of aggregating the predictions to make a final decision.

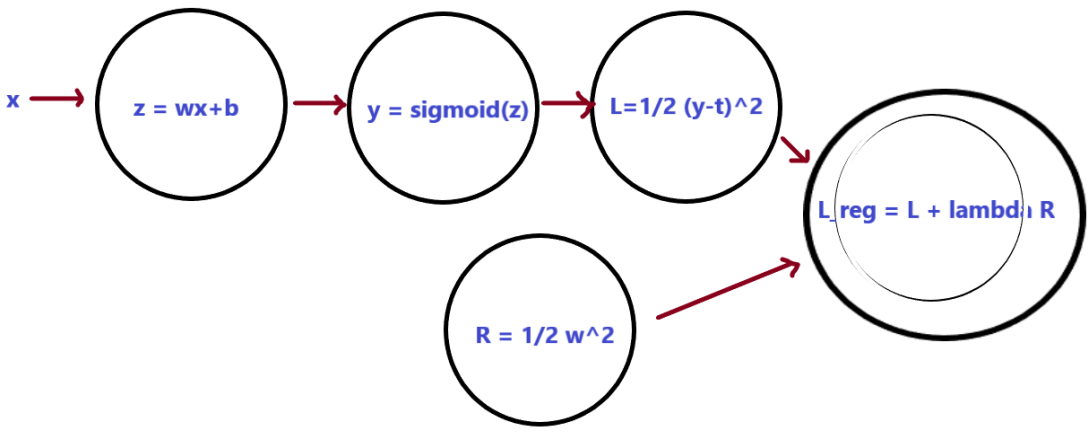
b

The preprocessing involves two main steps. First, the width of the bounding box for each digit is normalized to range from 8 to 20 pixels, with a step-size of 2 pixels, except for the digit "1". This results in 7 different training sets. Second, a deslanted training set is generated by horizontally shearing the digit based on the angle of the first principal component of pixel intensities with respect to the vertical axis.

The purpose of this preprocessing is to decorrelate the errors of the individual experts in the committee. By training each individual model on a different preprocessed version of the data, the models are exposed to different variations of the digits. This helps prevent the models from becoming overly dependent on specific features or variations in the data.

Question 3

a



$$\frac{\partial L_{reg}}{\partial L} = 1$$
$$\frac{\partial L_{reg}}{\partial R} = \lambda$$
$$\frac{\partial L_{reg}}{\partial y} = \frac{\partial L_{reg}}{\partial L}\frac{\partial L}{\partial y} = y - t$$
$$\frac{\partial L_{reg}}{\partial z} = \frac{\partial L_{reg}}{\partial L}\frac{\partial L}{\partial y}\frac{\partial y}{\partial z} = (y-t) \times \sigma(z) \times (1-\sigma(z))$$
$$\frac{\partial L_{reg}}{\partial w} = \frac{\partial L_{reg}}{\partial L}\frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w} + \frac{\partial L_{reg}}{\partial R}\frac{\partial R}{\partial w} = (y-t) \times \sigma(z) \times (1-\sigma(z)) \times x + \lambda \times w$$
$$\frac{\partial L_{reg}}{\partial x} = \frac{\partial L_{reg}}{\partial L}\frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial x} = (y-t) \times \sigma(z) \times (1-\sigma(z)) \times w$$
$$\frac{\partial L_{reg}}{\partial b} = \frac{\partial L_{reg}}{\partial L}\frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial b} = (y-t) \times \sigma(z) \times (1-\sigma(z)) \times 1 = (y-t) \times \sigma(z) \times (1-\sigma(z))$$

b

1. **Symmetry Issues:**
   - If all the weights are initialized to the same value, the neurons in a layer will learn the same features during training, and the network will not be

able to learn diverse representations. This symmetry issue hampers the ability of the neural network to capture complex patterns in the data.

2. **Vanishing or Exploding Gradients:**
   - Poorly chosen initial values may lead to the vanishing or exploding gradient problem. If the weights are initialized too small, gradients during backpropagation can become extremely small, causing the model to learn very slowly or not at all (vanishing gradients). On the other hand, if the weights are initialized too large, gradients can become extremely large, causing the model to diverge (exploding gradients).
3. **Convergence Issues:**
   - Inappropriate initialization can affect the convergence of the optimization algorithm. If the weights are not set to appropriate values, the network may converge to a suboptimal solution or fail to converge at all. Proper initialization helps the network start with reasonable parameter values and facilitates faster convergence.
4. **Training Time:**
   - Inefficient initialization may lead to longer training times. If the network takes a long time to converge or struggles with the learning process due to poor parameter initialization, it can increase the overall training time and resource requirements.

c

Update rule for w:

$$w_1 = w_0 - 0.1 \times ((\sigma(w_0 x_0 + b_0) - t) \times \sigma(w_0 x_0 + b_0) \times (1 - \sigma(w_0 x_0 + b_0)) \times x + \lambda \times w_0)$$

Update rule for b:

$$b_1 = b_0 - 0.1 \times (\sigma(w_0 x_0 + b_0) - t) \times \sigma(w_0 x_0 + b_0) \times (1 - \sigma(w_0 x_0 + b_0))$$

t is the output of network for input x_0.

Question 4

a

1. **Compute Gradient g_t:**

$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$$

Compute the gradient of the objective function f_t with respect to the parameters θ at iteration t. This gradient represents the steepest increase of the function.

2. **Update First Moment Estimate m_t:**

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

Update the first moment estimate m_t, an exponentially decaying average of past gradients. β_1 controls the decay rate of this estimate.

3. **Update Second Moment Estimate v_t:**

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Update the second moment estimate v_t, an exponentially decaying average of past squared gradients. β_2 controls the decay rate.

4. **Bias-Corrected First Moment Estimate $\hat{m}_t$:**

$$\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$$

Correct the bias in the first moment estimate by normalizing it. This prevents bias towards zero, especially during early iterations.

5. **Bias-Corrected Second Moment Estimate $\hat{v}_t$:**

$$\hat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$$

Correct the bias in the second moment estimate by normalizing it. This ensures unbiased estimates.

6. **Update Parameters θ_t:**

$$\theta_t \leftarrow \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Update the parameters θ using the Adam update rule. α is the learning rate, and ϵ is a small constant for numerical stability. The update incorporates both first and second moment estimates.

b

The values of m_t tend to zero with bias due to the nature of the exponential decay in the update rule:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

Starting with $m_0 = 0$, we can express m_t recursively:

$$\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\
&= \beta_1(\beta_1 m_{t-2} + (1 - \beta_1) g_{t-1}) + (1 - \beta_1) g_t \\
&= \beta_1^2 m_{t-2} + \beta_1(1 - \beta_1) g_{t-1} + (1 - \beta_1) g_t \\
&= \beta_1^3 m_{t-3} + \beta_1^2(1 - \beta_1) g_{t-2} + \beta_1(1 - \beta_1) g_{t-1} + (1 - \beta_1) g_t \\
&\vdots \\
&= \beta_1^t m_0 + \sum_{i=0}^{t-1} \beta_1^i (1 - \beta_1) g_{t-i-1}
\end{aligned}$$

As t becomes larger, the term $\beta_1^t m_0$ tends to zero, and the sum becomes dominated by the terms with g_{t-i-1}. This is why m_t tends to zero with bias – the exponentially decaying factor β_1^t reduces the influence of past gradients on m_t, causing it to diminish. Now, let's look at the bias-corrected term $\hat{m}_t$:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} = \frac{\beta_1^t m_0 + \sum_{i=0}^{t-1} \beta_1^i (1 - \beta_1) g_{t-i-1}}{1 - \beta_1^t}$$

As t becomes larger, β_1^t dominates the denominator, making it approach zero. This effectively counteracts the diminishing effect of the exponentially decaying factor on m_t. The bias correction term $1 - \beta_1^t$ prevents $\hat{m}_t$ from becoming too small, maintaining a more accurate estimate of the true first moment.

Question 5

a

Let $J(w) = w^T Hw$. The gradient of J with respect to w is given by:

$$\nabla J(w) = 2Hw$$

Now, substitute the eigenvalue decomposition of H into the expression for the gradient:

$$\nabla J(w) = 2(Q\Lambda Q^T)w$$

Using the fact that $Q^T Q = I$ (where I is the identity matrix), we can simplify this expression:

$$\nabla J(w) = 2Q\Lambda(Q^T w)$$

Now, the update rule for the gradient descent method is given by:

$$w_{\text{new}} = w_{\text{old}} - \epsilon \nabla J(w_{\text{old}})$$

Substitute the expression for the gradient:

$$w_{\text{new}} = w_{\text{old}} - 2\epsilon Q\Lambda(Q^T w_{\text{old}})$$

b

We know that:

$$w_t = w_{t-1} - 2\epsilon Q\Lambda(Q^T w_{t-1})$$

So:

$$w_1 = w_0 - 2\epsilon Q\Lambda Q^T w_0 = (I - 2\epsilon Q\Lambda Q^T)w_0 = Q(I - 2\epsilon\Lambda)Q^T w_0$$
$$w_2 = Q(I - 2\epsilon\Lambda)Q^T w_1 = Q(I - 2\epsilon\Lambda)Q^T Q(I - 2\epsilon\Lambda)Q^T w_0 = Q(I - 2\epsilon\Lambda)^2 Q^T w_0$$
$$\vdots$$
$$w_t = Q(I - 2\epsilon\Lambda)^t Q^T w_0$$

c

- **Positive Definiteness of H:**

The matrix H must be positive definite. This condition ensures that the objective function has a unique minimum.

• **Learning Rate (ϵ) Selection:**

$$-1 < (I - 2\epsilon\Lambda) < 1 \Rightarrow -1 < 1 - 2\epsilon\lambda_i < 1 \Rightarrow \epsilon < \frac{1}{\lambda_{\max}}$$

So The step size (ϵ) must be chosen such that it satisfies the condition $0 < \epsilon < \frac{1}{\lambda_{\max}}$, where $\lambda_{\max}$ is the maximum eigenvalue of H.

d

In Newton's method, the update rule for the weight vector w is given by:

$$w_{\text{new}} = w_{\text{old}} - \eta(H^{-1}\nabla J(w_{\text{old}}))$$

Here, $\nabla J(w)$ is the gradient of the objective function and H is the Hessian matrix of the objective function. In our case, the objective function is $w^T H w$, so the Hessian matrix is simply $2H$.

$$\begin{gathered} w_{\text{new}} = w_{\text{old}} - \eta((2H)^{-1}\nabla J(w_{\text{old}})) \\ \Rightarrow w_{\text{new}} = w_{\text{old}} - \eta((2H)^{-1}2Hw_{\text{old}}) \\ \Rightarrow w_{\text{new}} = w_{\text{old}} - \eta I w_{\text{old}} \\ w_{\text{new}} = (I - \eta I)w_{\text{old}} \\ w_{\text{new}} = (1 - \eta)w_{\text{old}} \end{gathered}$$

This means that, in each iteration, the weight vector is scaled by a factor of $1 - \eta$. The convergence behavior depends on the choice of the step size η. If η is too large, the method might overshoot and diverge; if η is too small, the method might converge slowly.

e

1. **Computational Complexity:** Computing and inverting the Hessian matrix is computationally expensive, especially for large neural networks with millions of parameters.
2. **Stochasticity and Mini-Batch Training:** Newton's method assumes access to the full dataset for each iteration, making it less compatible with the stochasticity introduced by mini-batch training, which is commonly used in deep learning.

Question 6

a

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

Taking the partial derivative with respect to w_k:

$$\frac{\partial J_1}{\partial w_k} = -\left(y_d - \sum_{k=1}^{n}(w_k + \delta_k)x_k\right) \cdot x_k$$

Now, we need to find the expected value of this expression, given that $\delta_k \sim N(0, \alpha w_k^2)$. The expected value $\mathbb{E}[\delta_k]$ is 0, and the expected value $\mathbb{E}[\delta_k^2]$ is αw_k^2. So, taking the expected value:

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial J_1}{\partial w_k}\right] &= -\mathbb{E}\left[\left(y_d - \sum_{k=1}^{n}(w_k + \delta_k)x_k\right) \cdot x_k\right] \\
&= -\mathbb{E}\left[\left(y_d - \sum_{k=1}^{n}(w_k + \delta_k)x_k\right) \cdot x_k\right] \\
&= -\mathbb{E}\left[\left(y_d - \sum_{k=1}^{n} w_k x_k - \delta_k x_k\right) \cdot x_k\right] \\
&= -\mathbb{E}\left[\left(y_d - \sum_{k=1}^{n} w_k x_k\right) \cdot x_k - \delta_k x_k^2\right] \\
&= -\mathbb{E}\left[\left(y_d - \sum_{k=1}^{n} w_k x_k\right) \cdot x_k\right] + \mathbb{E}[\delta_k x_k^2]
\end{aligned}$$

Now, let's simplify the first term:

$$\begin{aligned}
\mathbb{E}\left[\left(y_d - \sum_{k=1}^{n} w_k x_k\right) \cdot x_k\right] &= \mathbb{E}[y_d x_k] - \mathbb{E}\left[\sum_{k=1}^{n} w_k x_k^2\right] \\
&= y_d \mathbb{E}[x_k] - \sum_{k=1}^{n} w_k \mathbb{E}[x_k^2]
\end{aligned}$$

Since $\mathbb{E}[x_k^2]$ is the variance of x_k, denoted as σ_k^2, we can simplify further:

$$= y_d \mathbb{E}[x_k] - \sum_{k=1}^{n} w_k \sigma_k^2$$

Therefore, the final expression for the expected value of the gradient is:

$$\mathbb{E}\left[\frac{\partial J_1}{\partial w_k}\right] = -\left(y_d \mathbb{E}[x_k] - \sum_{k=1}^{n} w_k \sigma_k^2\right) + \alpha w_k^2 x_k^2$$

b

The term $-y_d \mathbb{E}[x_k]$ represents the contribution of the data-dependent part of the loss. The term $\sum_{k=1}^{n} w_k \sigma_k^2$ is the regularization term that penalizes large weights. This term discourages the model from relying too much on any particular feature by introducing a penalty proportional to the square of the weight multiplied by the corresponding dropout variance (σ_k^2). The last term $\alpha w_k^2 x_k^2$ is an additional regularization term introduced by the Gaussian dropout. It penalizes large weights by scaling the weight's square with the square of the corresponding input value (x_k^2) and a hyperparameter α, which controls the strength of the dropout.