Homework for Deep Learning

2024 Fall

Overview

Due Tuesday, 15 October 2024 at 23:59:00 China Standard Time

This homework includes written and programming problems related to Lecture 2-4.

- For **written problems**, please write your answers in a `pdf` file. You can use LaTeX, convert from Word, from notebook apps on tablets, scan your handwritten documents, and etc. Your derivation and proof should be completed in the `pdf` file. Besides, your section titles and subsection titles should be the number of question. i.e., # Q1 ## Q1.1.
- For **programming questions**, please submit your source files (basically `py` files). Note that your analysis should written in the `pdf` file. And some requested results should be included in the `pdf` file, too. e.g., your training loss curves.

Please organize your files as follows:

```
report_${student_id}.pdf
codes
   |--README.md
   |--source1.py
   |--source2.py
   |--source3.py
   |--...
```

You should put your codes and a `README.md` file into the `codes` directory. Please describe where to look in your codes for each programming question in `README.md`. For example, **Q5.2**'s code is in `source1.py`. Finally, please zip the `report_${student_id}.pdf` file and `codes` directory (rather than their parent directory) into `${student_id}.zip`. You should submit your `${student_id}.zip` to Web Learner before **Tuesday, 15 October 2024 at 23:59:00 China Standard Time**.

1 A Deep Learning Playground (15 pts)

Visualization of deep networks is beneficial for beginners. However, making neural networks work well in practice is not easy in general since there are too many hyper-parameters to tune, such as the choice of the activation function, the number of hidden layers, the learning rate, and so on. Besides some general guidelines (some standard techniques which are helpful in most cases, such as dropout and data augmentation), experiences are very important.

Though a beginner may often be confused with them, some software is available on the internet to help you better understand neural networks. In this problem, you need to train the neural networks with different choices of hyper-parameters from the following link - A Neural Network Playground - and answer the following questions:

Q1.1 (5 pts) Identify the best configuration you find for different problems and datasets. Here you only need to show your configuration for the classification problem's last dataset (as shown in Figure 1). Please present a screenshot like Figure 1 but includes your configurations and loss curve.

Q1.2 (5 pts) List your findings that how **(1)** the learning rate, **(2)** the activation function, **(3)** the number of hidden layers, and **(4)** the regularization influence the performance and convergence rate.

Q1.3 (5 pts) Do you find that the absolute value of some weights in your model becomes very small after training? Do you think this suggests that you can use a smaller model to achieve similar performance?

2 Linear Models (10 pts)

Q2.1 (5 pts) Consider the geometrical interpretation of the least-squares solution for linear regression. Show that $\hat{\boldsymbol{y}} = \boldsymbol{X}\theta^*$ corresponds to an orthogonal projection of the vector $\boldsymbol{y}$ onto the subspace spanned by the column of $\boldsymbol{X}$ (as depicted in Figure 2).

Q2.2 (5 pts) Given a set of data pts $\{\boldsymbol{x}_n\}$, we can define the *convex hull* to be the set of all pts $\boldsymbol{x}$ given by

$$\boldsymbol{x} = \sum_n \alpha_n \boldsymbol{x}_n,$$

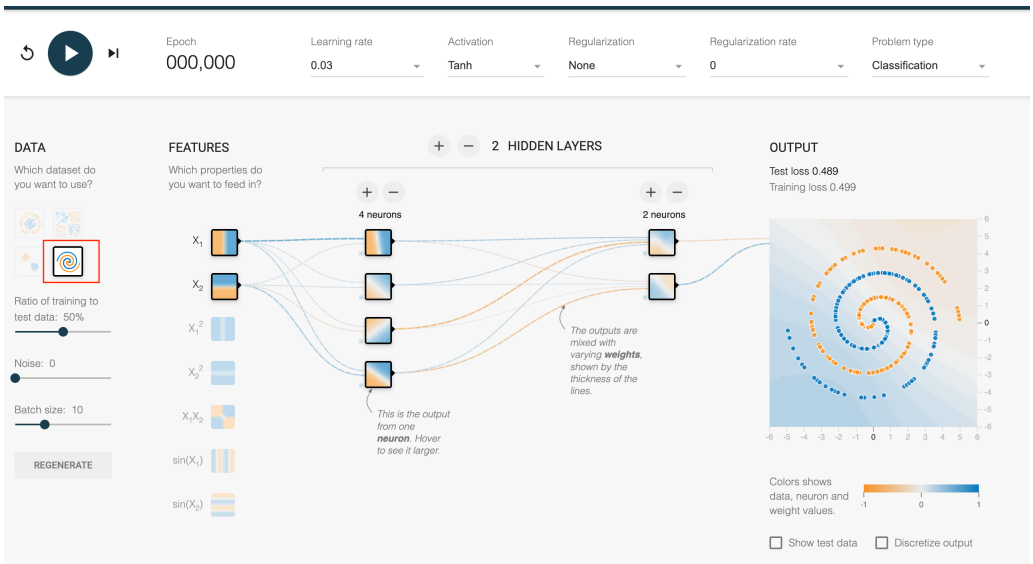

Figure 1

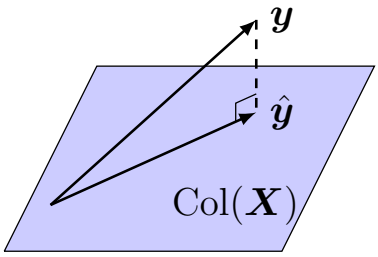

Figure 2: Geometry interpretation of linear regression.

where $\alpha_n \geq 0$ and $\sum_n \alpha_n = 1$. Consider a second set of pts $\{\boldsymbol{y}_n\}$ together with their corresponding convex hull. By definition, the two sets of pts will be linearly separable if there exists a vector $\hat{\boldsymbol{w}}$ and a scalar w_0 such that $\hat{\boldsymbol{w}}^T \boldsymbol{x}_n + w_0 > 0$ for all $\boldsymbol{x}_n$, and $\hat{\boldsymbol{w}}^T \boldsymbol{y}_n + w_0 < 0$ for all $\boldsymbol{y}_n$. Show that if their convex hulls intersect, the two sets of pts cannot be linearly separable, and conversely that if they are linearly separable, their convex hulls do not intersect.

3 Logistic Regression (10 pts)

Q3.1 (5 pts) The *average* negative log-likelihood $J(\boldsymbol{\theta})$ for binary logistic regression can be expressed as

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left[-y^{(i)} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \right) + \log \left(1 + \exp(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}) \right) \right]$$

where $\boldsymbol{x}^{(i)} \in \mathbb{R}^{M+1}$ is the column vector of the feature values of the i-th data point (especially, $\boldsymbol{x}^{(i)}_{M+1} = 1, \forall i$ is the constant feature), $y^{(i)} \in \{0, 1\}$ is the i-th class label, $\boldsymbol{\theta} \in \mathbb{R}^{M+1}$ is the weight vector. When we want to perform logistic ridge regression (i.e., with ℓ_2 regularization), we modify our objective function to be

$$f(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda \frac{1}{2} \sum_{j=0}^{M} \theta_j^2$$

where λ is the regularization weight, θ_j is the jth element in the weight vector $\boldsymbol{\theta}$. Suppose we are updating θ_k with learning rate η, please calculate the gradient $\frac{\partial f(\boldsymbol{\theta})}{\partial \theta_k}$ **(4 pts)**. Then please describe how we update θ_k **(1 pts)**.

Q3.2 (5 pts) Data is separable in one dimension if there exists a threshold t such that all values less than t have one class label and all values greater than or equal to t have the other class label. Suppose you train an **unregularized** logistic regression model for infinite iterations on training data that is separable in at least one dimension. In that case, the corresponding weight(s) can be infinity in magnitude. **(1)** What is an explanation for this phenomenon? **(3 pts) (2)** How does ℓ_1 and ℓ_2 regularization help correct the problem in the previous question? Please analyze ℓ_1 and ℓ_2 separately. **(2pts)**
Hint: Suppose you find a weight vector yielding a decision boundary that fully separates a dataset. Think about what the probability for each point is and whether you could adjust the weights to make the overall likelihood more favorable.

4 Neural Network and Backpropagation (20 pts)

Q4.1 Convolutional Neural Networks. In this problem, consider only the convolutional layer of a standard implementation of a CNN, as described in the lecture. We are given image X and filter F below. Note that we use no bias. **(10 pts)**

$$X = \begin{bmatrix} -15 & -10 & -13 & 12 & 6 & 2 \\ 8 & 12 & 10 & -1 & 3 & -15 \\ 8 & 10 & 11 & -14 & -14 & 5 \\ -4 & 4 & -11 & -6 & 13 & 14 \\ -13 & 3 & 8 & -3 & -2 & 15 \\ 14 & 12 & -2 & -9 & -11 & 0 \end{bmatrix}, \quad F = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad Y = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

(a) Input (b) Output candidate 1 (c) Output candidate 2

Figure 3

1. Let X be convolved with F using no padding and a stride of 1 to produce an output Y. What is value of j in the output Y?**(2 pts)** According to Lecture 4, if we convolve Figure 3a by the filter F, which of Figure 3b and 3c is more likely to be the output? **(2 pts)**
2. Suppose you had an input feature map of size $(6, 4)$ and filter size $(2, 2)$, using no padding and a stride of $(2, 2)$. What would be the resulting output size?**(2 pts)** How should we select the padding size to make the output the same size as the input $(6, 4)$? **(2 pts)**

   The first value of each provided tuple is used for the height dimension, and the second value is used for the width dimension.

   The padding on the left is the same as the right, and the padding on the top is the same as the bottom.

   - output size: height= width=
   - padding: left=right= top=bottom=
3. Suppose for the convolutional layer, we are given black and white images of size 22×22. Using one single 4×4 filter with a stride of 2 and no padding, what is the number of parameters you are learning in this layer? **(2 pts)**

Q4.2 (10 pts) Backpropagation. Consider this directed Acyclic Graph (DAG). Notice that it looks different from the fully-connected Neural Networks we have seen before. Recall from the lecture that you can perform backpropagation on any DAG. We will work through backpropagation on this graph. Let $(\boldsymbol{x}, \boldsymbol{y})$ be the training example that is considered, where

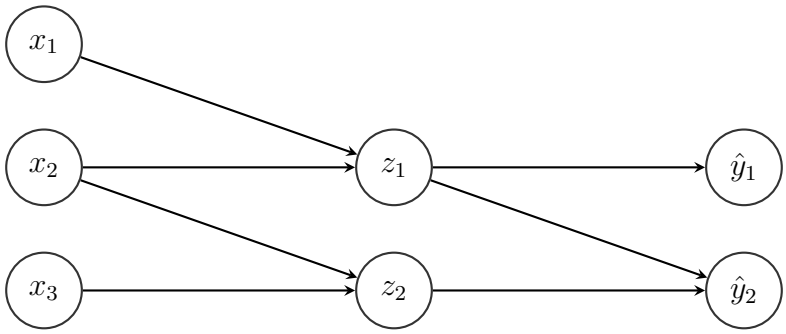

Figure 4: A Directed Acyclic Graph (DAG)

$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ and $\boldsymbol{y} = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T$. All the other nodes in the graph are defined as:

$$
\begin{aligned}
z_1 &= \text{ReLU}(w_{1,1}x_1 + w_{2,1}x_2) \\
z_2 &= w_{2,2}x_2^2 + w_{3,2}x_3 + b_2 \\
\hat{y}_1 &= \sigma(m_{1,1}z_1^3 + c_1) \\
\hat{y}_2 &= m_{1,2}\sin(z_1) + m_{2,2}\cos(z_2)
\end{aligned}
$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, and $\text{ReLU}(x) = \max(0, x)$. Let $\boldsymbol{\theta}$ be the set of all parameters to be learned in this graph. We have that

$$\boldsymbol{\theta} = \{w_{1,1}, w_{2,1}, w_{2,2}, w_{3,2}, m_{1,1}, m_{1,2}, m_{2,2}, b_2, c_1\}$$

For every set of input $\boldsymbol{x} = (x_1, x_2, x_3)$, we will define objective of the problem as minimizing the loss function

$$J(\boldsymbol{\theta}) = \log\left((y_1 - \hat{y}_1)^2\right) + \log\left((y_2 - \hat{y}_2)^2\right)$$

Assume that you have already gone through the forward pass with inputs $\boldsymbol{x} = (x_1, x_2, x_3)$ and stored all the relevant values. The following questions will derive the backpropagation algorithm applied to the above DAG.

1. First, we will derive the gradients with respect to the outputs. What are the expressions for $\frac{\partial J}{\partial \hat{y}_1}$? Write your solution in terms of $\hat{y}_1$. **(2 pts)**

2. Now, we will derive the gradients associated with the last layer, i.e., nodes $\hat{y}_1, \hat{y}_2$. Note that for the full backpropagation algorithm, you would need to calculate the gradients of the loss function with respect to every parameter $(m_{1,1}, m_{1,2}, m_{2,2}, c_1)$ as well as every input of the layer (z_1, z_2), but we are not asking for all of them in this part. For all of the questions in this part, you should use the Chain Rule, and write your solution in terms of values from the forward pass, or **gradients with respect to the outputs of this layer,** $\frac{\partial \mathbf{J}}{\partial \hat{\mathbf{y}}_1}$, $\frac{\partial \mathbf{J}}{\partial \hat{\mathbf{y}}_2}$, because you have already calculated these values. In addition, use the sigmoid function $\sigma(x)$ in your answer instead of its explicit form. *Hint*: Write the derivative of $\sigma(x)$ as a function of $\sigma(x)$, i.e., $\sigma'(x) = f\left(\sigma(x)\right)$

   What is the expression for $\frac{\partial J}{\partial z_1}$, $\frac{\partial J}{\partial z_2}$, $\frac{\partial J}{\partial m_{1,1}}$, $\frac{\partial J}{\partial m_{1,2}}$, and $\frac{\partial J}{\partial c_1}$? **(5 pts)**

3. Lastly, we will derive the gradients associated with the second layer, ie nodes z_1, z_2. Note that for the full backpropagation algorithm, you need to calculate the gradients of the loss function with respect to every parameter (every $w_{i,j}, b_2$). However, we do not need to calculate the gradients with respect to the inputs of this layer (x_1, x_2, x_3) because they are fixed inputs of the model. For all of the questions in this part, you should use the Chain Rule, and write your solution in terms of values from the forward pass or **gradients with respect to the outputs of this layer,** $\frac{\partial \mathbf{J}}{\partial \mathbf{z_1}}, \frac{\partial \mathbf{J}}{\partial \mathbf{z_2}}$, because you have already calculated these values.

   What is the expression for $\frac{\partial J}{\partial w_{2,2}}, \frac{\partial J}{\partial b_2}$? **(2 pts)**

   Recall that $\text{ReLU}(x) = \max(x, 0)$. The ReLU function is not differentiable at $x = 0$, but for backpropagation, we define its derivative as

$$\text{ReLU}'(x) = \begin{cases} 0, x < 0 \\ 1, \text{ otherwise} \end{cases}$$

   Now, what is the expression for $\frac{\partial J}{\partial w_{1,1}}$? Explicitly write out the cases. **(2 pts)**

5 Do We Need Zero Training Loss After Achieving Zero Training Error? (45 pts)

Overfitting is one of the most significant interests and concerns in the machine learning community. One way of identifying overfitting is to see whether the generalization gap, the test minus the training loss, is increasing or not. We can further decompose the generalization gap increasing into two stages: The first stage is when both the training and test losses are decreasing, but the training loss decreases faster than the test loss. The next stage is when the training loss drops, but the test loss increases.

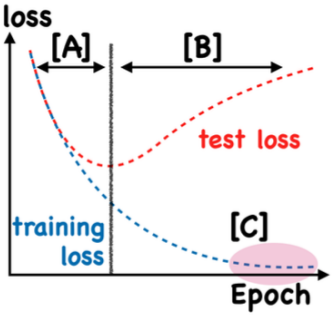

Figure 5

Q5.1 (5 pts) Give a possible explanation of why the test loss increases when training loss decreases in the second stage.

Recent works on over parameterization and double descent curves have shown that learning until zero training error is meaningful to achieve a lower generalization error. However, whether zero training loss is necessary after achieving zero training error remains an open issue. This question, we will investigate this interesting question. To prepare for this, let's first get familiar with what will happen using the classic training scheme.

Q5.2 (10 pts) Train a one-hidden-layer feed-forward neural network with 500 units and the ReLU activation function on the MNIST dataset for 500 epochs. Split the original training dataset into training and validation data with a proportion of 8:2. Use stochastic gradient descent with a learning rate of 0.1 and a momentum of 0.9 to optimize your network. Do not use any data augmentation or manual learning rate decay. Minimizing the empirical loss for training:

$$\hat{R}(g) = \frac{1}{N}\sum_{i=1}^{N} l(g(x_i), y_i), \tag{1}$$

where y_i is the one-hot groundtruth label, $g(x_i)$ is the output of your network, and l is a loss function, such as cross entropy. After training, show the curves of how training error and validation error change. By error, we mean the percentage of samples that are wrongly classified.

Q5.3 (10 pts) In this question, try to make the training loss *float* around a small constant value, in order to prevent the training loss from approaching zero. Specifically, minimize the following loss for training:

$$\tilde{R}(g) = |\hat{R}(g) - b| + b \tag{2}$$

where $\hat{R}(g)$ is defined in Eq. 1. Search the optimal b in $\{0.00, 0.01, 0.02, \ldots, 0.20\}$. Report the curves showing how training error and validation error change under the optimal b. Compare the final performance to the results in **Q5.2**.

Q5.4 (5 pts) Based on the above experiments, do you think we need zero training loss after achieving zero training error?

Q5.5 (15 pts) Train a convolutional neural network on the same training and validation data in **Q5.2**. A suggested network architecture could be:

1. Conv2d(in_channels=1,out_channels=16,kernel=5x5,stride=1x1,padding=2x2);
2. ReLU();
3. MaxPool2d(kernel=2,stride=2,padding=0,dilation=1,ceil_mode=False);
4. Conv2d(in_channels=16,out_channels=32,kernel=5x5,stride=1x1,padding=2x2);
5. ReLU();
6. MaxPool2d(kernel=2,stride=2,padding=0,dilation=1,ceil_mode=False);
7. Linear().

You are also welcome to design your own network architecture. Compare the performance with the one-hidden-layer feed-forward neural network implemented in **Q5.2**. Discuss the differences between two networks.

Deep Learning Hands-On

In this homework, you can use any deep learning library to build your neural networks. We recommend PyTorch, and you can find a tutorial with examples here.

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. You can find an introduction to the python-mnist interface here.

Grading

This assignment counts for a final grade of 10 points. Following is the conversion of your homework score (implied by points after each exercise) and final grade:

$$\text{Final Grade} = \left\lceil \frac{\text{This homework score}}{100} \times 10 \right\rceil$$

We will actively be checking for plagiarism. If plagiarism is found, your final grade will be **F**.

Acknowledgment

This homework references UC Berkely and CMU courses.