Homework 4 - Math Foundation for Machine Learning Engineers

kipngeno koech - bkoech

November 3, 2024

1 Warm Up Derivatives (10 Points)

Compute the first and second derivatives of the following functions:

1. $f(x) = \mathbf{M}^\top \mathbf{M}\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{M} \in \mathbb{R}^{m \times m}$. What happens if $\mathbf{M} = \mathbf{M}^\top$?
2. $f(\mathbf{X}) = \text{tr}(\mathbf{M}\mathbf{X})$, where $\mathbf{M} \in \mathbb{R}^{m \times n}$ and $\mathbf{X} \in \mathbb{R}^{n \times m}$, and tr is the trace of a square matrix.
3. $f(\mathbf{X}) = \text{tr}(\mathbf{X}\mathbf{X}^\top)$, where $\mathbf{X} \in \mathbb{R}^{n \times n}$, and tr is the trace of a square matrix.
4. $f(\mathbf{X}) = \mathbf{a}^\top \mathbf{X}\mathbf{b}$, where $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$.
5. $f(\mathbf{X}) = \mathbf{a}^\top \mathbf{X}^\top \mathbf{X}\mathbf{b}$, where $\mathbf{X} \in \mathbb{R}^{n \times m}$, $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^m$.

2 Ridge Regression (20 Points)

In ordinary least squares (OLS) regression, we seek to minimize the sum of squared residuals:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \tag{1}$$

where $\mathbf{y}$ is the vector of observed target values and $\mathbf{X}$ is the matrix of input features.

We have shown in class and during recitation that the optimal parameters for OLS regression can be given by:

$$\boldsymbol{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{2}$$

However, OLS can have several issues:

- The matrix $\mathbf{X}^\top \mathbf{X}$ may be non-invertible or ill-conditioned when features are correlated (multi-collinearity).
- OLS can lead to overfitting, especially when the number of features is large relative to the number of observations.
- OLS is sensitive to outliers since the cost function is based on squared residuals.

2.1 Ridge Regression Solution (10 Points)

Ridge regression modifies the OLS objective function by adding a penalty term:

$$\min_{\boldsymbol{\beta}} \left(\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|^2 \right) \tag{3}$$

where λ is a regularization parameter that controls the strength of the penalty. Show that the optimal parameters $\boldsymbol{\beta}$ is:

$$\boldsymbol{\beta} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \tag{4}$$

where $\mathbf{I}$ is the identity matrix. This ensures that the solution is always unique (since $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ is always invertible) and less sensitive to collinear features or outliers.

2.2 Coding Ridge Regression (10 Points)

Complete the Ridge Regression code in the attached notebook and report the effect of changing the bias parameter.

3 Artificial Neural Network (60 Points)

In our recent recitation, we discussed artificial neural networks (ANNs) as complex functions that represent the intricate architecture of the human brain's neural network. These functions are subject to optimization to improve performance, a task often achieved through backpropagation and gradient descent, wherein the network adjusts its weights to minimize error and enhance accuracy in tasks such as classification and prediction.

We can summarize optimizing the model parameters (weights and biases) as follows:

Learning in deep neural networks using Gradient Descent can be summarized as follows:

1. **Initialization:** Initialize the network's weights and biases, often randomly or by some heuristic.
2. **Forward Pass:** Input data is passed through the network. Each neuron computes a weighted sum of the inputs and a bias, followed by an activation function.
3. **Loss Calculation:** The network's prediction is compared to the true outcome using a loss function to calculate the error.
4. **Backward Pass (Backpropagation):** Calculate the gradient of the loss function with respect to each parameter using the chain rule.
5. **Gradient Descent:** Adjust the parameters in the opposite direction of the gradient to minimize the error.
6. **Update Parameters:** Update the weights by the equation:

$$w_{\text{new}} = w_{\text{old}} - \alpha \cdot \nabla_w L,$$

where α is the learning rate, $\nabla_w L$ is the gradient of the loss with respect to the weights, and w represents the weights.

7. **Iteration:** Repeat the forward pass, loss calculation, backpropagation, and parameter update until the loss converges or a stopping criterion is met.

3.1 Chain Rule and Backpropagation (20 Points)

Consider a network structured with two input nodes, two hidden nodes, and a single output node. Nodes 3 and 4 use the sigmoid and tanh functions, respectively. The remaining nodes are designed with linear activation, meaning that their outputs are directly proportional to their inputs.

We will follow the procedures described in the previous section.

1. **Initialization:** The weights are assigned as follows:

   The inputs of node 1 and node 2 are 1.0 and -1.0 respectively.

$$W_{13} = 1, \quad W_{14} = -1, \quad W_{23} = 2, \quad W_{24} = -3, \quad W_{35} = 2, \quad W_{45} = -1.$$

   The hyperbolic tangent function is given by:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

   The sigmoid function is:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

   There is no bias term in this network.

2. **Forward Pass (5 Points):** Compute the output of node 5.
3. **Loss Calculation (2 Points):** The loss function is $L(y, \hat{y}) = (\hat{y} - y)^2$, where y is the output of node 5 (answer from previous part) and $\hat{y}$ is the desired output. In this part, we let $\hat{y} = 1.0$. Compute the loss of the network.
4. **Backpropagation (8 Points):** Calculate the gradient of the loss function with respect to each parameter using the chain rule. Remember the parameters are normally the weights and biases, but here we don't use a bias so the parameters are the weights. So for each weight, write down the derivative with respect to that weight
5. **Gradient Descent and Parameter Update (5 Points):**Use the gradient descent optimization technique to update the parameters. Since we are minimizing the loss, we will go in the opposite direction of the gradient.

   $$w_{\text{new}} = w_{\text{old}} - \alpha \cdot \nabla_w L,$$

   use learning rate $\alpha = 0.1$.

   After one iteration, report the weight values in the table below (keep only 2 decimal digits): example in table below

W_{13}	W_{14}	W_{23}	W_{24}	W_{35}	W_{45}
0.50	0.60	0.55	0.65	0.75	0.85

3.2 Checkpoint 1

1. Are you comfortable calculating the derivatives ?
2. Are you comfortable applying the gradient descent ?
3. Are you comfortable completing the previous section ?

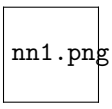
If your answer is NO, please post on piazza, attend office hour for a TA to help you understand. Do not move to next section. You might see this in your exam.

If your answer is Yes, congratulations, you learned some of the basics of neural network and it will help you in the Introduction to deep learning and Introduction to machine learning course.

3.3 Bonus (5 Points)

Run another iteration (5 points) Use the updated weights of previous section. However the inputs of node 1 and node 2 are 2.0 and -2.0 respectively, the desired output is 2. What is your predicted output, and what is your updated weights ?

3.4 One Layer MLP (40 Points)



In a one-layer Multi-Layer Perceptron (MLP) with a single hidden layer, we represent the weights as matrices to efficiently handle the numerous connections between layers. For a network with m input neurons and n hidden neurons, the weights and biases are defined as follows:

- Weights of the Hidden Layer ($\mathbf{W}_h$): An $m \times n$ matrix.
- Biases of the Hidden Layer ($\mathbf{b}_h$): An n-dimensional vector.
- Weights of the Output Layer ($\mathbf{W}_o$): An $n \times k$ matrix (assuming k output neurons).
- Biases of the Output Layer ($\mathbf{b}_o$): A k-dimensional vector.

The sigmoid activation function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Its derivative, which is used in the computation of the gradient, is:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

The Mean Squared Error (MSE) loss function is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

where N is the number of samples, Y is the true output, and $\hat{Y}$ is the predicted output.

Write down the gradients of the MSE with respect to the following parameters (15 points):

1. Gradient with respect to $\mathbf{W}_o$:
$$\frac{\partial MSE}{\partial \mathbf{W}_o}$$
2. Gradient with respect to $\mathbf{b}_o$:
$$\frac{\partial MSE}{\partial \mathbf{b}_o}$$
3. Gradient with respect to $\mathbf{W}_h$:
$$\frac{\partial MSE}{\partial \mathbf{W}_h}$$
4. Gradient with respect to $\mathbf{b}_h$:
$$\frac{\partial MSE}{\partial \mathbf{b}_h}$$

Write down the gradient descent update rules for the parameters using a learning rate of 0.1 (5 points).

Complete the MLP codes in the starter notebook (20 points).

Bonus (10 points): Add Another layer hidden layer to MLP. and complete the codes in the 2 Layer MLP. You should still use sigmoid activation on the new layer. What is the effect of adding another layer ?

4 Evaluations (10 Points)

In class, we discussed solving a linear regression using the pseudo-inverse, which gives the same result as minimizing the residual sum of squares. In the starter notebook, I have provided a solution to the linear regression problem using SVD and the pseudo-inverse. Compare the root mean square error (RMSE) of this method, the ridge regression method, and the one-layer MLP. Why do you think the errors are different ? Which one has the lowest error and why ?

I have provided a keras implementation of a deeper neural network with 4 layers and it ouputs the lowers RMSE error. However, there are some techniques added here like dropout, weight initialization, different activation function, and Adam optimizer (we used SGD optimizer in previous question). If you are interested in knowing more about deep neural networks, then you should consider taking the course introduction to deep learning offered here at CMU by Prof Bhiksha.