Module 1 Project: Bike Sharing

COSC750 - Neural Networks and Deep Learning

Devere Anthony Weaver

Problem A1

A1-1

Answer: (B) `b.shape=(1, 4, 5) c.shape=(4, 5)`

A1-2

Answer: (A) `torch.Size([60])`

A1-3

Answer: (B) `torch.Size([1, 3, 4, 5])`

A1-4

Answer: (C) `torch.Size([1, 1, 3, 4, 5])`

A1-5

Answer: (D) `torch.Size([3, 1, 4, 5])`

Problem A2

The dataset for the binary classification task is:

x_1	x_2	t
0	0	1
1	1	1
1	0	0

We use a linear model with a dummy dimension $x_0 = 1$ to incorporate the bias term w_0. The decision function is:

$$y = w_0 + w_1 x_1 + w_2 x_2. \tag{1}$$

Using a hard threshold activation function, the classification rule is:

$$\hat{t} = \begin{cases} 1, & \text{if } y > 0 \\ 0, & \text{if } y < 0 \end{cases} \tag{2}$$

To ensure the weight vector lies in the strictly feasible region, none of the training examples should be on the decision boundary $y = 0$.

Constraints

First Example: $(x_1, x_2) = (0, 0)$**,** $t = 1$

$$w_0 > 0. \tag{3}$$

Second Example: $(x_1, x_2) = (1, 1)$**,** $t = 1$

$$w_0 + w_1 + w_2 > 0. \tag{4}$$

Third Example: $(x_1, x_2) = (1, 0)$**,** $t = 0$

$$w_0 + w_1 < 0. \tag{5}$$

For the weight vector (w_0, w_1, w_2) to be in the strictly feasible region, it must satisfy:

$$w_0 > 0, \tag{6}$$
$$w_0 + w_1 + w_2 > 0, \tag{7}$$
$$w_0 + w_1 < 0. \tag{8}$$

These constraints ensure that none of the training examples lie on the decision boundary.

Problem A3

Given the function:

$$J(w) = L(w) + \frac{\lambda}{2} w^2, \tag{9}$$

where

$$L(w) = \frac{1}{2}(\sigma(wx + b) - t)^2, \tag{10}$$

we compute the partial derivative .

Step 1: Compute $\frac{\partial L(w)}{\partial w}$

Using the chain rule:

$$\frac{\partial L(w)}{\partial w} = (\sigma(wx + b) - t) \cdot \frac{d}{dw}\sigma(wx + b). \tag{11}$$

Since $\sigma(\cdot)$ is the activation function,

$$\frac{d}{dw}\sigma(wx + b) = \sigma'(wx + b) \cdot x, \tag{12}$$

where $\sigma'(z)$ is the derivative of $\sigma(z)$. Thus,

$$\frac{\partial L(w)}{\partial w} = (\sigma(wx + b) - t) \cdot \sigma'(wx + b) \cdot x. \tag{13}$$

Step 2: Compute $\frac{\partial}{\partial w}\left(\frac{\lambda}{2} w^2\right)$

$$\frac{\partial}{\partial w}\left(\frac{\lambda}{2} w^2\right) = \lambda w. \tag{14}$$

Step 3: Compute $\frac{\partial J(w)}{\partial w}$

$$\frac{\partial J(w)}{\partial w} = \frac{\partial L(w)}{\partial w} + \lambda w. \tag{15}$$

Substituting the expression for $\frac{\partial L(w)}{\partial w}$:

$$\frac{\partial J(w)}{\partial w} = (\sigma(wx + b) - t) \cdot \sigma'(wx + b) \cdot x + \lambda w. \tag{16}$$

$$\frac{\partial J(w)}{\partial w} = (\sigma(wx + b) - t) \cdot \sigma'(wx + b) \cdot x + \lambda w. \tag{17}$$

Problem A4

Computation Graph

We parameterize the weights as follows:

$$w_1 = \cos\theta, \tag{18}$$

$$w_2 = \sin\theta. \tag{19}$$

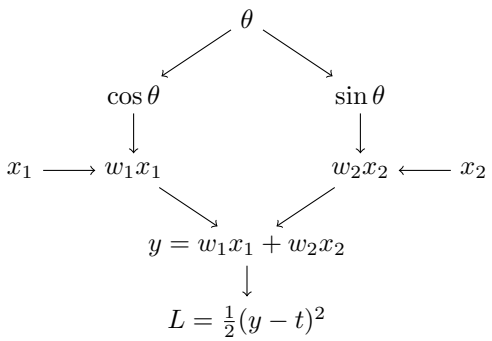
The model output is:

$$y = w_1 x_1 + w_2 x_2 = \cos\theta \cdot x_1 + \sin\theta \cdot x_2. \tag{20}$$

The loss function is:

$$L = \frac{1}{2}(y - t)^2. \tag{21}$$

The computation graph can be represented as:



Backpropagation Update Rules

To compute $\frac{dL}{d\theta}$, we follow these steps:

Step 1: Compute $\frac{dL}{dy}$

$$\frac{dL}{dy} = \frac{d}{dy}\left(\frac{1}{2}(y - t)^2\right) = (y - t). \tag{22}$$

Step 2: Compute $\frac{dy}{dw_1}$ and $\frac{dy}{dw_2}$

$$\frac{dy}{dw_1} = x_1, \quad \frac{dy}{dw_2} = x_2. \tag{23}$$

Step 3: Compute $\frac{dw_1}{d\theta}$ and $\frac{dw_2}{d\theta}$

$$\frac{dw_1}{d\theta} = -\sin\theta, \quad \frac{dw_2}{d\theta} = \cos\theta. \tag{24}$$

Step 4: Chain Rule for $\frac{dL}{d\theta}$

$$\frac{dL}{d\theta} = \frac{dL}{dy} \cdot \left(\frac{dy}{dw_1} \cdot \frac{dw_1}{d\theta} + \frac{dy}{dw_2} \cdot \frac{dw_2}{d\theta} \right). \tag{25}$$

Substituting values:

$$\frac{dL}{d\theta} = (y - t) \cdot (x_1(-\sin\theta) + x_2(\cos\theta)) \tag{26}$$

$$= (y - t) \cdot (-x_1 \sin\theta + x_2 \cos\theta). \tag{27}$$

$$\frac{dL}{d\theta} = (y - t) \cdot (-x_1 \sin\theta + x_2 \cos\theta). \tag{28}$$

This result can be used for updating θ using gradient descent.

Problem A5

Logistic Function and its Derivative

The logistic function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{29}$$

Step 1: Compute $\sigma'(z)$

Differentiating $\sigma(z)$ using the chain rule:

$$\sigma'(z) = \frac{d}{dz} \left(\frac{1}{1 + e^{-z}} \right) \tag{30}$$

$$= -(1 + e^{-z})^{-2} \cdot (-e^{-z}) \tag{31}$$

$$= \frac{e^{-z}}{(1 + e^{-z})^2}. \tag{32}$$

Step 2: Express $\sigma'(z)$ in Terms of $\sigma(z)$

From the definition of $\sigma(z)$, we substitute:

$$e^{-z} = \frac{1 - \sigma(z)}{\sigma(z)}. \tag{33}$$

Thus, rewriting $\sigma'(z)$:

$$\sigma'(z) = \frac{\sigma(z)(1 - \sigma(z))}{1} \tag{34}$$

$$= \sigma(z)(1 - \sigma(z)). \tag{35}$$

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)). \tag{36}$$

Problem C1

Training

The starting date is 2011-01-01 and the ending date is 2012-10-10.

Validation

The starting date is 2012-10-10 and then ending date is 2012-12-10.

Testing

The starting date is 2012-12-10 and then ending date is 2012-12-31.

Problem C2

Yes, my model converges each time I've experimented with it. It appears to go through all the iterations set and not within an arbitrary error measure. So it looks like the number of iterations is the stopping criteria.

Problem C3

Based on the output in the final cell of my notebook, my test MSE was 8087.0249. I'm not sure if this is a "good" MSE value; however, it is the highest one that I was able to achieve along with the lowest training and validation MSE values.

Based on the graph, also did a decent job at prediction, but of course it can always be improved. Changing the number of iterations to 30,000 helped my model achieve the best performance. I attempted 50,000 iterations, but this was rather time consuming so I canceled the run as I wasn't sure how much more of an improvement it would be over the 30,000 iterations.