CENG501 Deep Learning Homework 1

İsmail Karabaş
Student ID: 2375186

October 18, 2023

Q1.1

$L1(w,b) = \frac{1}{N}\sum_{i=1}^{N} |y_i - f(x_i; w, b)|$

Let $d_i = y_i - f(x_i; w, b)$.

Case 1 when $d_i > 0$

So $|d_i| = d_i$
Now, the loss function becomes:
$L1(w,b) = \frac{1}{N}\sum_{i=1}^{N} d_i$
Partial Derivative with Respect to w:
$\frac{\partial L1(w,b)}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial d_i}{\partial w}$
Now, $d_i = y_i - f(x_i; w, b)$, so we need to find $\frac{\partial d_i}{\partial w}$. Using the chain rule:
$\frac{\partial d_i}{\partial w} = \frac{\partial}{\partial w}(y_i - f(x_i; w, b))$

In the context of the linear function $f(x_i; w, b) = \sigma(w^T x_i + b)$, where $\sigma(t)$ is a step function, the expression $w^T x_i$ is a dot product between the weight vector w and the input vector x_i.
Now, when finding the partial derivative of $f(x_i; w, b)$ with respect to w, we consider the term $w^T x_i$. The derivative of a dot product of vectors is computed with respect to each component of the vectors, and the result is the vector itself. Therefore, $\frac{\partial}{\partial w}(w^T x_i) = x_i$.
In simpler terms, when we vary a single component w_j in w, the change in $w^T x_i$ is precisely x_{ij}, the corresponding component of the input vector x_i. This leads to the derivative being x_i.
Substituting this back into the expression for the partial derivative:

$$\frac{\partial L1(w,b)}{\partial w} = -\frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\frac{\partial L1(w,b)}{\partial b} = -\frac{1}{N}\sum_{i=1}^{N} 1$$

Case 2 When $|d_i| = 0$:

it implies $d_i = 0$. In this case, the absolute value does not contribute to the loss. The partial derivatives become:

$$\frac{\partial L1(w,b)}{\partial w} = 0$$

$$\frac{\partial L1(w,b)}{\partial b} = 0$$

Case 3: When $d_i < 0$:

In this case, $|d_i| = -d_i$. The loss function becomes:
$L1(w,b) = \frac{1}{N}\sum_{i=1}^{N} -d_i$
Now, let's find the partial derivatives for this case:
$\frac{\partial L1(w,b)}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial d_i}{\partial w}$
$\frac{\partial L1(w,b)}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} 1$
Substituting the derivative of d_i with respect to w:
$\frac{\partial d_i}{\partial w} = -x_i$
We get:

$$\frac{\partial L1(w,b)}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\frac{\partial L1(w,b)}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} 1$$

Overall Solution:

The partial derivatives can be expressed in a concise manner using the sign function:

$$\frac{\partial L1(w,b)}{\partial w} = -\frac{1}{N}\sum_{i=1}^{N} \text{sign}(d_i) \cdot x_i$$

$$\frac{\partial L1(w,b)}{\partial b} = -\frac{1}{N}\sum_{i=1}^{N} \text{sign}(d_i)$$

Q1.3 Plots of the Model

For the following two plots, the used command is: `python main.py data/linear_data_train.pickle.gz data/linear_data_test.pickle.gz 35 .1`

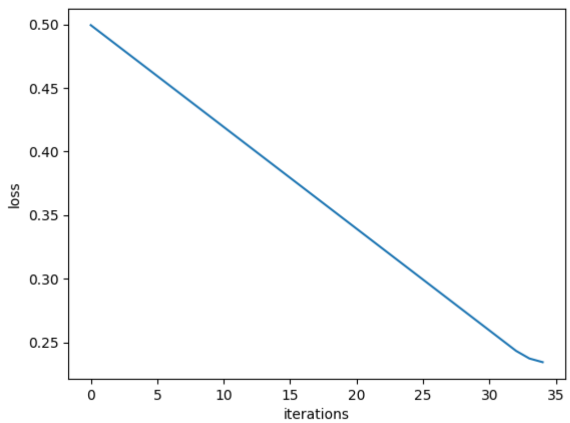

Figure 1: Loss vs Iterations

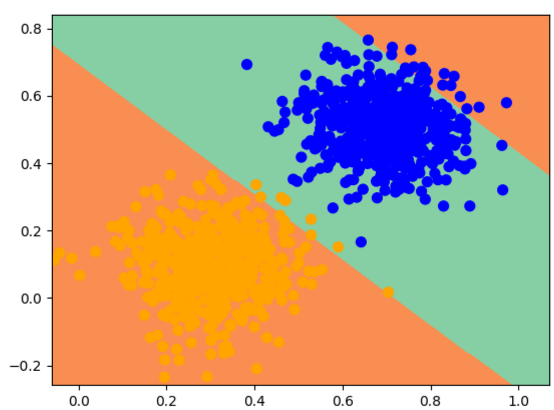

Figure 2: Weights Distribution

For the following two plots, the used command is: `python main.py data/mnist_01_train.pickle.gz data`
`50 .001`
I changed parameters till finding a good Learning behaviour just like described in the homework text.

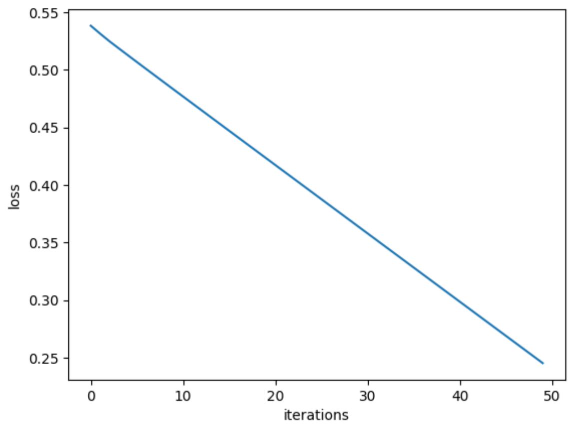

Figure 3: Loss vs Iterations

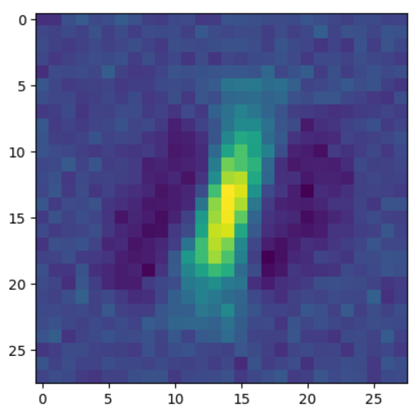

Figure 4: Weights Distribution

Q1.4

Given the definition of the L1 loss function in:

$L1(w,b) = \frac{1}{N}\sum_{i=1}^{N} |y_i - f(x_i; w, b)|$

where $f(x_i; w, b) = \sigma(w^T x_i + b)$ and $\sigma(t) = \begin{cases} 0 & \text{if } t \leq 0 \text{ or } t > 1 \\ t & \text{otherwise} \end{cases}$

Let's use the properties of convex functions and operations according to the lecture slides.

1. **Convexity of Absolute Value:** - The absolute value function $|z|$ is convex, but not differentiable at $z = 0$.

2. **Convexity of Piecewise Function:** - For $t \leq 0$ and $t > 1$, $\sigma(t)$ is constant (0 and t respectively), and constant functions are convex. - For $0 < t \leq 1$, $\sigma(t) = t$, which is also a linear function which is convex in piecewise manner. However, Overall function is not continuos. Therefore it is not a convex function

3. **Convexity of L1 Loss:** - The L1 loss is a sum of not convex functions which remains as not convex.

In conclusion, based on the provided definition, the L1 loss function is $\boxed{\textbf{not convex}}$ in terms of the parameters of the classifier (w and b). The non-convexity of the L1 loss is attributed to the non-convexity of the piecewise function $\sigma(t)$.

If $\sigma(t)$ was convex , we could say that the loss function is convex because functions obtained from convex functions and convexity preserving operations are convex.

Q1.5

Algorithm 1 Using K-Fold Cross-Validation for Best Parameter Selection

Set `best_accuracy` to 0
Divide the dataset D into k folds
for each possible hyperparameter C **do**
    Set `total_accuracy` to 0
    for each fold in the dataset **do**
        Use the current fold as the validation set
        By combining the remaining $k-1$ folds create the training set
        Train the classifier with hyperparameter C using the training set
        Test the classifier on the validation set and calculate accuracy `current_accuracy`
        Update `total_accuracy` by adding `current_accuracy`
    end for
    Calculate average accuracy: $\texttt{average\_accuracy} = \frac{\texttt{total\_accuracy}}{k}$
    if `average_accuracy` is better than `best_accuracy` **then**
        Update `best_accuracy` and `best_hyperparameter`
    end if
end for
`best_hyperparameter` is the best hyperparameter for best performance

Q1.6

Algorithm 2 Using Cross-Validation for Future Performance Estimation

Divide the dataset D into k folds
Set `total_performance` to 0
for each fold in the dataset **do**
    Use the current fold as the validation set
    By combining the remaining $k-1$ folds create the training set
    Train the model M using the training set
    Test performance of the model on the validation set
    Calculate the accuracy of the model `current_performance`
    Accumulate `total_performance` by adding the `current_performance`
end for
Calculate the average performance: $\texttt{average\_performance} = \frac{\texttt{total\_performance}}{k}$
Represent the estimated future performance of the model M with `average_performance`