Introduction to Deep Learning for Computer Vision Assignment 3: Simple Linear Classifier II

Paul Molina-Plant

January 30, 2019

Abstract

In this assignment, I finished a implementing a linear classifier. This included implementing the loss function for logistic regression and implementing cross validation.

In section 1, I derive the gradient for cross entropy use in logistic regression.
The weights gradient is,

$$\frac{\partial L_i}{\partial W} = (P_{ij} - \delta_{ij})x_i$$

And the bias gradient is,

$$\frac{\partial L_i}{\partial b_i} = P_{ij} - \delta_{ij}$$

In section 2, I plotted the results of the classifier being run on cifar10 in linear svm and logistic regression mode. For this dataset, linear svm produced a marginally more accurate model.

In section 3, I plotted the results of the classifier being run in logistic regression mode with cross validation. I ran the classifer twice, once with a learning rate of $lr_1 = 10^{-4}$ and then with $lr_2 = 10^{-2}$. I observed that the accuracy and loss plots of lr_1 stabalize quickly, which means lr_1 is a good learning rate. In contrast, the plots of lr_2 never stabilizes because the learning rate is too high, causing the descent algorithm to consistently overshoot.

1 Derivation of the gradient for cross entropy

1.1 The cross entropy function

$$L_i = -log\frac{\sum_j t_{ij}e^{s_{ij}}}{\sum_k e^{s_{ik}}} \tag{1}$$

I assume *log* base e for (1).

$$t_{ij} = \delta_{ij} = \begin{cases} 1 & j = y_i \\ 0 & j \neq y_i \end{cases} \tag{2}$$

Softmax P_{ij}

$$P_{ij} = \frac{e^{s_{ij}}}{\sum_k e^{s_{ik}}} \tag{3}$$

Among the k classes, $j = y_i$ for exactly one of them. Therefore, we can ignore the other $k-1$ terms in the sum of (1).

$$L_i = -ln\frac{e^{s_{ij}}}{\sum_k e^{s_{ik}}} = -lnP_{ij} \tag{4}$$

1.2 Gradient of W

Computing the gradient $\frac{\partial L_i}{\partial W}$ via chain rule.

$$\frac{\partial L_i}{\partial W} = \frac{\partial L_i}{\partial P_{ij}}\frac{\partial P_{ij}}{\partial s_{ij}}\frac{\partial s_{ij}}{\partial W}$$

Linear classifier $f(x_i, W) = s_{ij}$.

$$s_{ij} = Wx_i + b_i \tag{5}$$

The partial derivative of s_{ij} wrt W is nonzero when $j = y_i$ and zero everywhere else. This condition is expressed with δ_{ij} (2).

$$\frac{\partial s_{ij}}{\partial W} = \delta_{ij}x_i$$

The partial derivative of P_{ij} (3) wrt W by applying the chain rule.

$$\begin{aligned}
\frac{\partial P_{ij}}{\partial W} &= \frac{\partial}{\partial W}\frac{e^{s_{ij}}}{\sum_k e^{s_{ik}}} = \frac{\delta_{ij}x_i e^{s_{ij}} - e^{s_{ij}}x_i e^{s_{ij}}}{(\sum_k e^{s_{ik}})^2} \\
&= \left(\frac{e^{s_{ij}}}{\sum_k e^{s_{ik}}}\right)\left(\frac{\delta_{ij}x_i\sum_k e^{s_{ik}} - x_i e^{s_{ij}}}{\sum_k e^{s_{ik}}}\right) \\
&= P_{ij}(\delta_{ij}x_i - P_{ij}x_i) \\
&= P_{ij}x_i(\delta_{ij} - P_{ij})
\end{aligned}$$

Finally the partial derivative of L_i (4) wrt W.

$$\begin{aligned}
\frac{\partial L_i}{\partial W} &= \frac{\partial}{\partial W}(-lnP_{ij}) \\
&= -\frac{1}{P_{ij}}P_{ij}x_i(\delta_{ij} - P_{ij}) \\
&= (P_{ij} - \delta_{ij})x_i
\end{aligned} \tag{6}$$

1.3 Gradient of b_i

Computing the gradient $\frac{\partial L_i}{\partial b_i}$ via chain rule.

$$\frac{\partial L_i}{\partial b_i} = \frac{\partial L_i}{\partial P_{ij}} \frac{\partial P_{ij}}{\partial s_{ij}} \frac{\partial s_{ij}}{\partial b_i}$$

The partial derivative of s_{ij} (5) wrt b_i is nonzero when $j = y_i$ and zero everywhere else. This condition is expressed with δ_{ij} (2).

$$\frac{\partial s_{ij}}{\partial W} = \delta_{ij}$$

The partial derivative of P_{ij} (3) wrt b_i by applying the chain rule.

$$\frac{\partial P_{ij}}{\partial b_i} = \frac{\partial}{\partial W} \frac{e^{s_{ij}}}{\sum_k e^{s_{ik}}} = \frac{\delta_{ij} e^{s_{ij}} \sum_k e^{s_i k} - e^{s_{ij}} e^{s_{ij}}}{(\sum_k e^{s_{ik}})^2} = \delta_{ij} P_{ij} - P_{ij}^2$$

Finally the partial derivative of L_i (4) wrt b_i.

$$\begin{aligned} \frac{\partial L_i}{\partial b_i} &= \frac{\partial}{\partial b_i} (-ln P_{ij}) \\ &= -\frac{1}{P_{ij}} (\delta_{ij} P_{ij} - P_{ij}^2) \\ &= P_{ij} - \delta_{ij} \end{aligned} \tag{7}$$

1.4 Summary of derivations

The weights gradient was,

$$\frac{\partial L_i}{\partial W} = (P_{ij} - \delta_{ij}) x_i$$

The bias gradient was,

$$\frac{\partial L_i}{\partial b_i} = P_{ij} - \delta_{ij}$$

2 Training curves

The following curves were obtained by running the HOG model on cifar-10 dataset with default parameters. Linear SVM was marginally more accurate, but both performed similarly.

2.1 Linear SVM

Best validation accuracy: 46.58%
Test accuracy: 46.47%

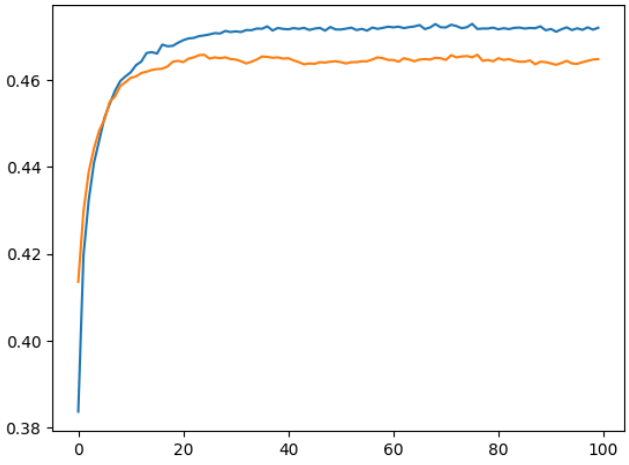

Figure 1: Validation Accuracy.

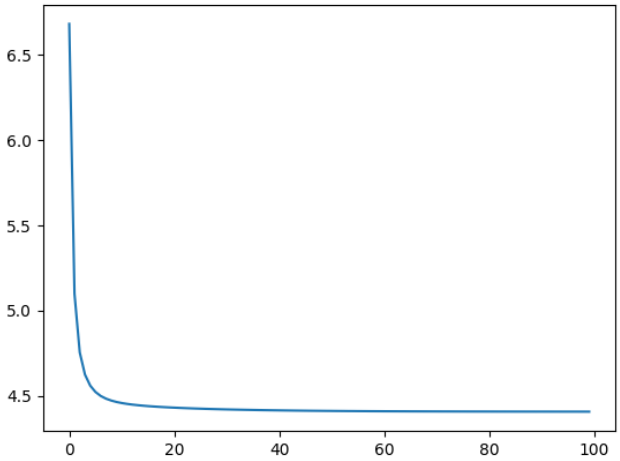

Figure 2: Validation Loss.

2.2 Logistic Regression

Best validation accuracy: 42.60%
Test accuracy: 42.449%

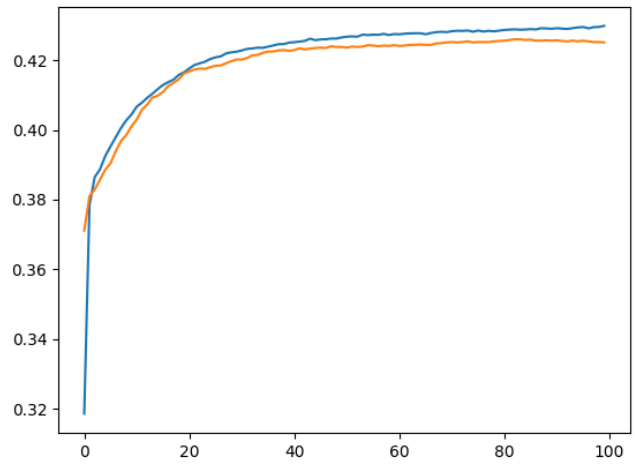

Figure 3: Validation Accuracy.

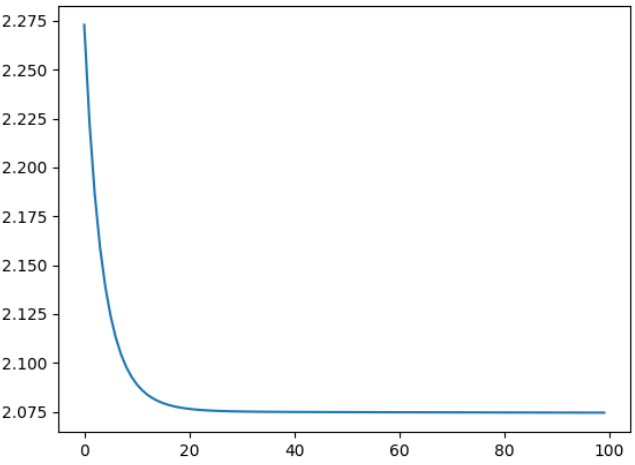

Figure 4: Validation Loss.

3 Cross validation results

The following validation results were obtained by running the HOG model on cifar-10 dataset with logistic regression and cross validation. Learning rate was adjusted between tests, other parameters were left default.

3.1 Learning Rate $10^{-}4$

Average best validation accuracy: 42.95%
Good learning rate because accuracy curve stabilizes at 43% and loss curve rapidly falls off and stabilizes.

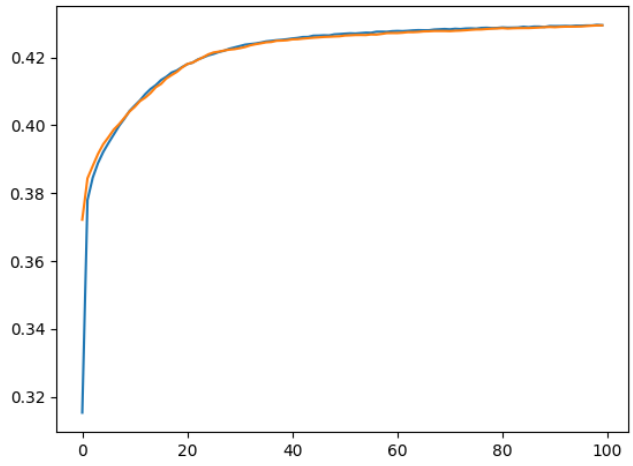

Figure 5: Validation Accuracy.

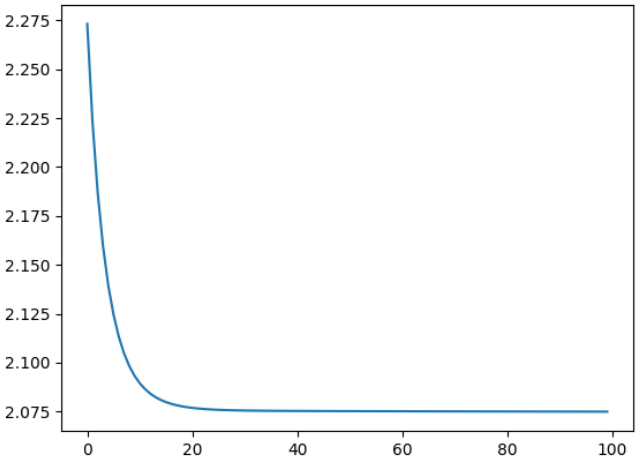

Figure 6: Validation Loss.

3.2 Learning Rate $10^{-}2$

Average best validation accuracy: 43.77%
A bad learning rate because accuracy and loss curves never stabilize. This is caused by the learning rate being too high, which causes the descent algorithm to overshoot.

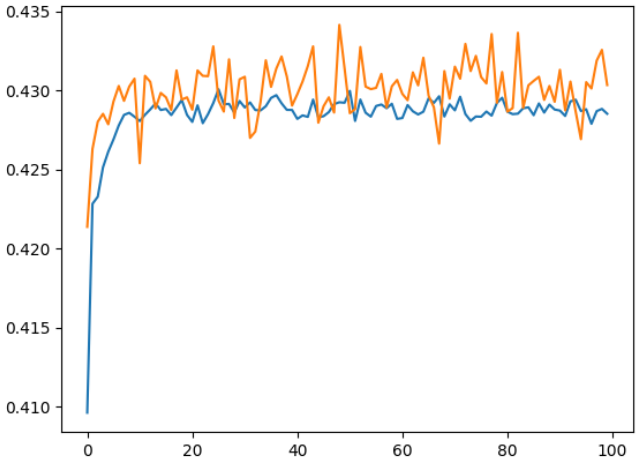

Figure 7: Validation Accuracy.

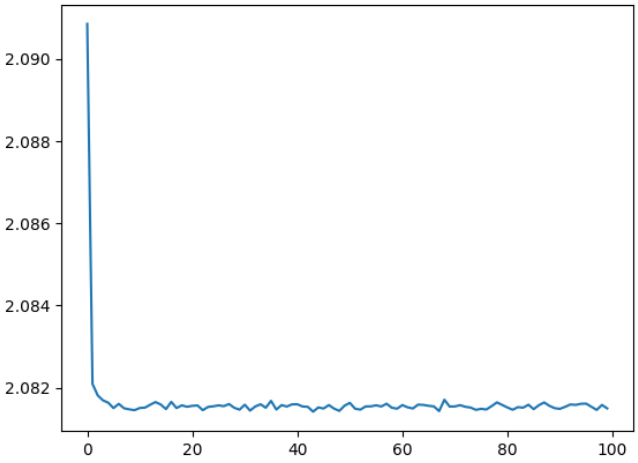

Figure 8: Validation Loss.