CP8318/CPS803 Machine Learning

Assignment 3

Rohaan Ahmed
PhD Student - Computer Science
rohaan.ahmed@ryerson.ca

November 12, 2020

Instructor: Dr. Nariman Farsad
November 12, 2020

1. Neural Networks: MNIST image classification

1.1 Unregularized Model

Baseline

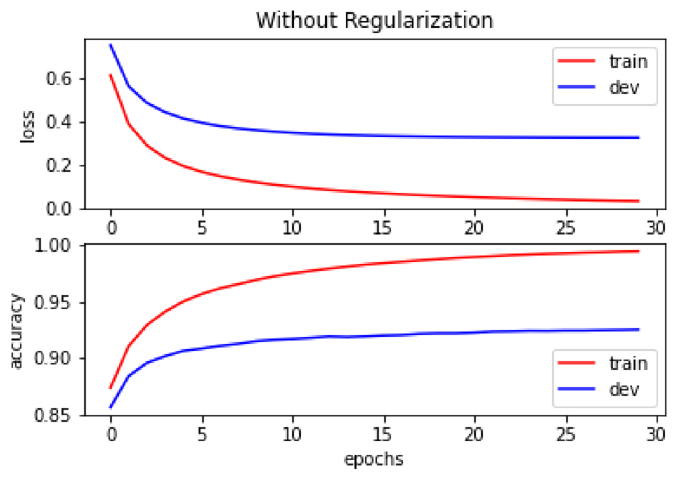

Figure 1: Unregularized Model - Train and Dev Accuracy and Loss vs Epoch

1.2 Regularized Model

Regularized

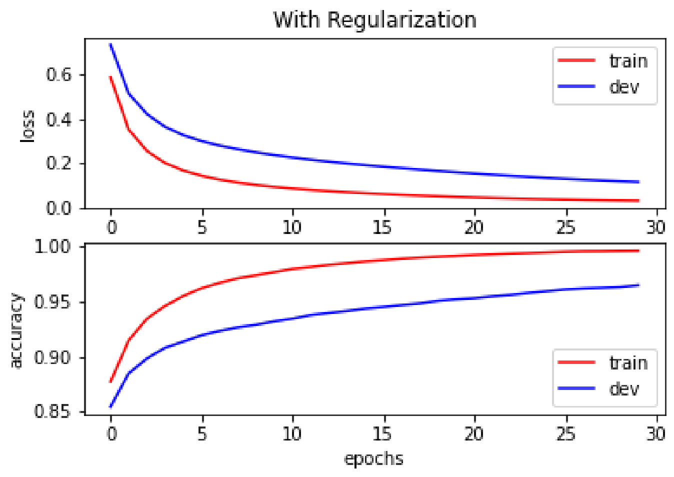

Figure 2: Regularized Model - Train and Dev Accuracy and Loss vs Epoch

1.3 Final Test

Unregularized Model

At the end of Training Epoch 30 of 30:

- Training Accuracy = 0.99086
- Dev Accuracy = 0.9295
- Test Set Accuracy = 0.927700
- Difference between Test and Training Accuracy = 6.3%

Regularized Model
At the end of Training Epoch 30 of 30:

- Training accuracy = 0.99616
- Dev accuracy = 0.966
- Test Set Accuracy = 0.965400
- Difference between Test and Training Accuracy = 3.1%

Observations

The difference between the loss and accuracy plot-lines is greater in the unregularized variant, indicating that there is a larger discrepency between classification of dev and training datasets. This points to "overfitting", i.e., when a model learns the specifics of a training dataset rather than the generalized idea.

This hypothesis of overfitting without regularization is strengthened when the models are evaluated on the test set. The regularized variant of the model performs far better on the test dataset than the unregularized version, despite having similar performance on the training dataset.

The images below show some test samples taken randomly from the Test Dataset, with the associated true Labels and Predictions:

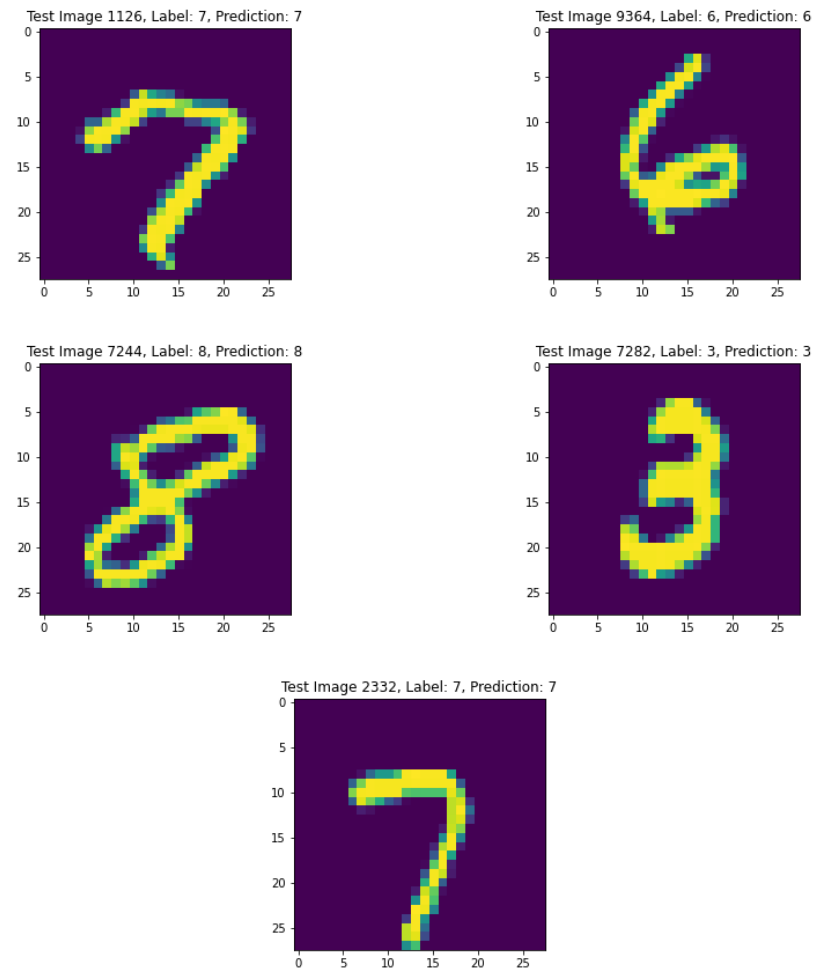

Figure 3: Random Test Samples with Regularized Model Predictions

Another interesting observation is that, even with a simple 1-layer-deep architecture, our model converges very quickly. The regularized model crosses the 90% accuracy mark on the dev set in approximately 3 epochs. For some applications in industry, this may be sufficient.

2. CP8318 Only Question: A Simple Neural Network

2.1

As shown in the course notes, the Chain Rule gives us:

$$\frac{\partial loss}{\partial w_{1,2}^{[1]}} = \frac{\partial loss}{\partial o}\frac{\partial o}{\partial h_2}\frac{\partial h_2}{\partial w_{1,2}^{[1]}}$$

If $g(x)$ is the sigmoid function, then its derivative $g'(x)$ is given by:

$$g'(x) = g(x)(1 - g(x))$$

giving (after substitution and algebra):

$$\implies \frac{\partial loss}{\partial w_{1,2}^{[1]}} = \frac{2}{m}\sum_{i=1}^{m}(o^{(i)} - y^{(i)})o^{(i)}(1 - o^{(i)})w_2^{[2]}h_2^{(i)}(1 - h_2^{(i)})x_1^{(i)}$$

where:

$$h_2^{(i)} = g(x_1^{(i)}w_{1,2}^{[1]} + x_2^{(i)}w_{2,2}^{[1]} + w_{0,2}^{[1]})$$

The update rule is:

$$w_{1,2}^{[1]} := w_{1,2}^{[1]} - \alpha\frac{\partial loss}{\partial w_{1,2}^{[1]}} \tag{0.1}$$

2.2

Letting the activation functions for h_1, h_2, h_3 be the linear function $f(x) = x$ will map the input features $x \in \mathbb{R}^2$ to hidden layer features $a^{[1]} \in \mathbb{R}^3$. The output of the hidden layer will thus be a "linear scaled mapping" of 2d input features into 3 dimensions.

In theory, then, we should be able to achieve 100% accuracy if the data were linearly separable in $\mathbb{R}^3$, i.e., Class 0 and Class 1 were lineraly separable in 3 dimensions.

However, looking at the plot, we can see that Class 0 is enclosed within the convex boundary of Class 1. In order to separate the two classes, we would certainly require non-linear mapping. Therefore, we conclude that 100% accuracy is not possible with a linear function for hidden layer activation.