# Report

**Anonymous Author(s)**
Affiliation
Address
email

## 1 Softmax Regression via Gradient Descent

### 1.1 Problem definition

In this problem, we need to classify MNIST datasets using softmax regression. In the experiments, we only use the first 20,000 training images and the last 2,000 test images.

### 1.2 Methods

We use softmax regresion for this problem.
**Derive the gradient for Softmax Regression:**
The cross-entropy cost function can be expressed as,

$$E = -\sum_{n}\sum_{k=1}^{c} t_k^n \ln y_k^n \tag{1}$$

Where,

$$y_k^n = \frac{\exp\left(a_k^n\right)}{\sum_{k'} \exp\left(a_{k'}^n\right)} \tag{2}$$

And,

$$a_k^n = w_k^T x^n \tag{3}$$

We can calculate the gradient for softmax regression as follows,

$$\begin{aligned} -\frac{\partial E^n(w)}{\partial w_{jk}} &= -\frac{\partial E^n(w)}{\partial a_k^n}\frac{\partial a_k^n}{\partial w_{jk}} \\ &= -\sum_{k'}\frac{\partial E^n(w)}{\partial y_{k'}^n}\frac{\partial y_{k'}^n}{\partial a_k^n}\frac{\partial a_k^n}{\partial w_{jk}} \end{aligned} \tag{4}$$

And

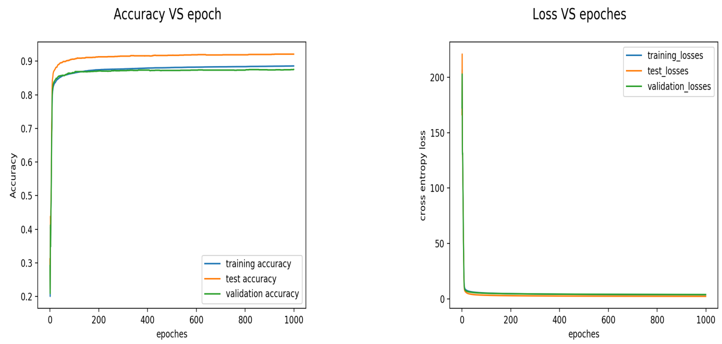
$$\frac{\partial y_{k'}^n}{\partial a_k^n} = y_{k'}^n \delta_{kk'} - y_{k'} y_k \tag{5}$$

Where $\delta_{kk} = 1$ if $k = k'$, otherwise $\delta_{kk} = 0$. And

$$\frac{\partial E^n(w)}{\partial y_{k'}^n} = -\frac{t_{k'}}{y_{k'}} \tag{6}$$

Substitute Equation (5) and Equation (6) into Equation (4) we get,

$$-\frac{\partial E^n(w)}{\partial w_{jk}} = (t_k - y_k) x_j^n \tag{7}$$




(a) label 1 (b) label 2

Figure 1: 2 Figures side by side

**Prepossing**: First, we extract the first 20,000 training images and the last 2,000 test images. Then normailize the images to make sure the pixel values are in the range of [0,1]. Convert the labels to one-hot vectors. Divide the training images into two parts, the first 10% are used for as a hold-out set and the rest 90% are used for training.

**Experiments settings:** To determine the best type of regurization and the best $\lambda$, we try $L_2$ regularization and $L_1$ regularization seperately.

For the $L_2$ regularizartion, we search the best $\lambda$ in the set $\{0.01, 0.001, 0.0001\}$. If the accuracy on the hold-out set decreases for 3 epochs, we stop the algorithm and use the weights with the minimum error (highest accuracy) on the hold-out set as the final answer. For the $L_1$ regularization, we follow the same steps. Then we compare the results get from these two regularization methods and use it as the best final result.

### 1.3 Results

(a) In the experiments, we find that using $L_2$ regularization with $\lambda = 0.01$ obtain the best result on the validation set. With an accuracy of $0.9045\%$ on the validation set. With such settings, the accuracy on the test set is $0.927\%$.

(b) In this experiement, we use $L_2$ regularization with $\lambda = 0.01$. The figure is shown in Fig **??**.

(c) In this experiement, we use $L_2$ regularization with $\lambda = 0.01$. The figure is shown in Fig **??**.

(d) We plot the results in Fig 2. We can see that the image of the weight and the corresponding image of the average digit is almost the same. The reason is that we classify the images based on the inner product of the pixesls with the weights. And the inner product is maximized when the angle between the weight and the image is zero. So we see that the image of the weight and the corresponding image of the average digit is similar.

### 1.4 Discussion

### Acknowledgments

.

### References

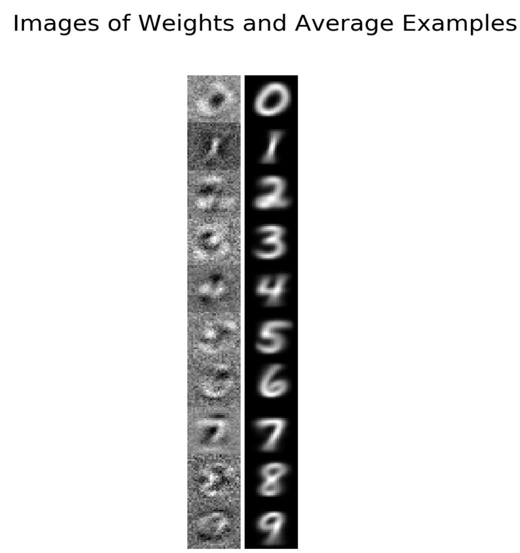



Figure 2