CSC411 Machine Learning
Project 2: Deep Neural Network

Ariel Kelman
Student No: 1000561368

Gideon Blinick
Student No:

13 February 2018

1 Introduction

The following figure shows 10 random images from the training set of each of the digits.

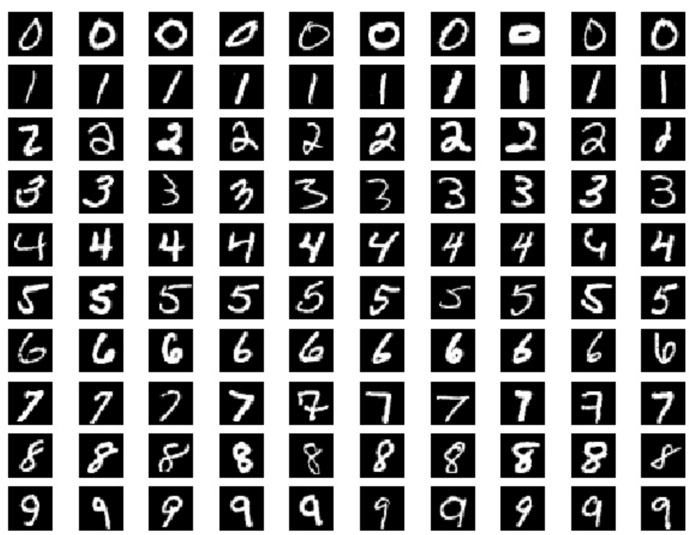

Figure 1: 10 random samples from the training set for each digit. This image was generated using `plot_samples()`.

The data was downloaded from the assignment webpage, and imported into Python using the provided code. The data was already divided into training and testing sets...

All results and plots can be generated using the python file... To reproduce the report, simply run it through latex.

2 The Network

The following function implements a neural network with no hidden layers, with the output passed through a softmax layer to estimated probabilities.

```
def func():
```

The network is described by the weights and biases from the 784 = 28*28 inputs (representing pixel intenisites of the input image) to ten "output" nodes (with the identity as the activation function). The output from this layer is what is passed through the sofmax function $p_k = \frac{e^{o_k}}{\sum_q e^{o_q}}$.

The weights are represented as a 784 by 10 matrix W, where w_{ij} (i^{th} row, j^{th} column) represents the weight from the i^{th} input to the j^{th} output. When computing the network on a given sample, the transpose of W is multiplied by the column vector (or matrix when computing on multiple samples) representing the input. The biases are represented by a 10 by 1 vector, one entry for each output.

3 Gradient

The cost function is taken to be $\sum_k y_k ln(p_k)$ for one sample, where y_k is 1 for the correct class and 0 otherwise, and p_k is the prediction probability for class k. Writing this in vector notation (replacing the sum with a vector dot product) and summing over all the samples gives:

$$C = -\sum_s y^{(s)} ln(p^{(s)})$$

where ln is applied pointwise, and s is an index over the samples. $y^{(s)}$ is a vector of 0's with a 1 in the position representing the correct digit.

The following results will be used in the derivations throughout this section:

$$\frac{\partial p_k}{\partial o_q} = \begin{cases} -p_k p_q & \text{if } k \neq p \\ p_q(1-p_q) & \text{if } k = p \end{cases}$$

These results follow directly from the definition of the softmax function.

3.1 Gradient wrt w_{ij}

Differentiating the cost with respect to a general weight w_{ij}, and applying the chain rule:

$$\begin{aligned} \frac{\partial C}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \sum_s y^{(s)} ln(p^{(s)}) \\ &= \sum_s \sum_q \frac{\partial C}{\partial o_q} \frac{\partial o_q}{\partial w_{ij}} \\ &= -\sum_s \sum_q (p_q - y_q) x_i \end{aligned}$$

where $\frac{\partial C}{\partial o_q}$ was computed by

$$\begin{aligned} \frac{\partial C}{\partial o_q} &= \sum_s \left[\frac{\partial C}{\partial p_q} \frac{\partial p_q}{\partial o_q} + \sum_{k \neq q} \frac{\partial C}{\partial p_k} \frac{\partial p_k}{\partial o_q} \right] \\ &= -\sum_s \left[y_q \frac{1}{p_q} p_q (1-p_q) + \sum_{k \neq q} y_k \frac{1}{p_k} p_k p_q \right] \\ &= -\sum_s p_q - y_q \end{aligned}$$

The last line follows because y is 1 only for the correct result. As the outputs are linear functions of both the inputs and the weights, the derivative with respect to a weight is given by the value of the input to that weight, namely x_i. The (s) superscript indicating the sample index was omitted for clarity of presentation.

3.2 Vectorized Gradient Code

Vectorizing the above result, by representing all input training images in a matrix X (784 by the number of samples) gives

$$\nabla_W C = (p - y)$$