Using Neural Networks to Effectively Classify Hand-Written Digits of the MNIST Dataset

Chetan Gandotra
UC San Diego
9500 Gilman Drive
cgandotr@ucsd.edu

Abstract

This report discusses the second programming assignment of our course CSE 253: Neural Networks and Pattern Recognition, its solutions and the inferences we drew. A variable layer neural network was implemented from the scratch and observations were made based on various parameters and before/after adding certain trades of tricks as discussed in Yann LeCun's famous paper "Efficient BackProp". The data-set used was the famous MNIST data-set and a ten-way classification was performed on it. A test data-set accuracy in excess of 97% was achieved using various mechanisms and tricks, which is almost at par with the accuracy reported by LeCun on his website. This is the individual part of the programming assignment.

1 Linear Regression and Sum-of-squared Error

1.1 Introduction

In linear regression problems, we try to fit the data generated from:

$$t = h(x) + \epsilon \tag{1}$$

where *x* is a *K*-dimensional vector, *h(x)* is a deterministic function of *x*, where *x* includes the bias term x_0 and ϵ is random noise that has a Gaussian probability distribution with zero mean and variance σ^2, i.e., ϵ ~ $\mathcal{N}(0, \sigma^2)$. Now, consider *y* to be of form

$$y = \sum_{i=1}^{K} w_i.x_i \tag{2}$$

We need to prove that finding the optimal parameter *w* for the above linear regression problem on the dataset D = (x^1, t^1), (x^2, t^2), ..., (x^N, t^N) is equal to finding the w^* that minimizes the sum of squared error (SSE):

$$w^* = argmin_w \sum_{n=1}^{N} (t^n - y^n)^2 \tag{3}$$

1.2 Methodology

In this section, we will prove that finding the optimal parameter *w* for the above linear regression problem on the dataset D = (x^1, t^1), (x^2, t^2), ..., (x^N, t^N) is equal to finding the w^* that minimizes the sum of squared error (SSE):

$$w^* = argmin_w \sum_{n=1}^{N} (t^n - y^n)^2 \tag{4}$$

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

To begin with, we look at (1). We are saying that the actual label is equal to the predicted label, plus some noise ϵ. Now, we are given that the noise ϵ has a Gaussian probability distribution with zero mean and variance σ^2, i.e., ϵ ~ $\mathcal{N}(0, \sigma^2)$.

Modifying (1), we can write:

$$\epsilon = h(x) - t \tag{5}$$

Also, the Gaussian distribution for ϵ can be written as:

$$p(\epsilon) = \frac{1}{(2\pi\sigma^2)^{1/2}} exp(-\frac{\epsilon^2}{2\sigma^2}) \tag{6}$$

Substituting value of ϵ from (5) and putting *h(x) = y* (which corresponds to the model we are using), we get:

$$p(\epsilon) = \frac{1}{(2\pi\sigma^2)^{1/2}} exp(-\frac{(y-t)^2}{2\sigma^2}) \tag{7}$$

For least error, the probability of predicting a label must be maximized, when provided with the corresponding input features *X*. This is repeated for all training examples from 1 to *N* and hence, we end up taking the multiplication of all probabilities. In a nutshell, we can maximize the probability of correct prediction by finding weights *w* such that the following expression is maximized:

$$max_w \prod_{n=1}^{N} \frac{1}{\sigma(2\pi)^{1/2}} exp(-\frac{(t_n - y_n)^2}{2\sigma^2})$$

The above maximization objective follows from (2), which contains *w*. Removing the constant terms,

$$= max_w \prod_{n=1}^{N} exp(-(t_n - y_n)^2)$$

Taking log and converting maximization objective to minimize by discarding the negative sign, we get:

$$= min_w \sum_{n=1}^{N} (t_n - y_n)^2$$

$$= argmin_w \sum_{n=1}^{N} (t_n - y_n)^2$$

which is essentially the same as equation (4).

1.3 Results

Using the derivation described above, we have proved that finding the optimal parameter *w* for the above linear regression problem on the dataset D = (x^1, t^1), (x^2, t^2), ..., (x^N, t^N) is equal to finding the w^* that minimizes the sum of squared error (SSE):

$$w^* = argmin_w \sum_{n=1}^{N} (t^n - y^n)^2$$

1.4 Discussion

Adjusting the weight vector to get maximum probability of finding the correct label is equivalent to minimizing the sum of squared error. This is true because in both the cases we are trying to reduce the number of mistakes by reducing the distance between actual and predicted values. A good classifier must have less sum of squared error or maximum probability of the label being correct, and for these two statements being equivalent makes sense.

2 Derivation of Delta's, Update Rule and Vectorization

2.1 Introduction

In this question, we have been given a three layer neural network with *J* units in the hidden layer. We use index *k* to represent a node in output layer, index *j* to represent a node in hidden layer and index *i* to represent a node in the input layer. Additionally, the weight from node *i* in the input layer to node *j* in the hidden layer is w_{ij}. Similarly, the weight from node *j* in the hidden layer to node *k* in the output layer is w_{jk}.

Firstly, we are required to derive the expression for δ for both the units of output layer (δ_k) and the hidden layer (δ_j). The definition of δ here is $\delta_i = -\frac{\partial E}{\partial a_i}$, where a_i is the weighted sum of the inputs to unit *i*.

Second, we need to derive the update rule for w_{ij} and w_{jk} using learning rate η, starting with the gradient descent rule:

$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \tag{8}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} \tag{9}$$

The derivative should take into account all of the outputs, so:

$$\delta_j = -\frac{\partial E^n}{\partial a_j^n} = \sum_k \frac{\partial E^n}{\partial y_k} \frac{\partial y_k}{\partial a_j} \tag{10}$$

Finally, we have been asked to write these equations in a vectorized form.

2.2 Methodology

For the first part for this question, we are required to derive expressions for δ_k and δ_j, where $\delta_i = -\frac{\partial E}{\partial a_i}$ We know that:

$$a_k = \sum_j w_{ij} z_j \tag{11}$$

Also,

$$\delta_k^n = -\frac{\partial E^n}{\partial a_k}$$

$$=> \delta_k^n = -\sum_{m=1}^{C} \frac{\partial E^n}{\partial y_m^n} . \frac{\partial y_m^n}{\partial a_k^n} \tag{12}$$

Now, we know that the error for n^{th} example - E^n can be written as:

$$E^n = -\sum_k t_k^n . ln(y_k^n)$$

Taking derivative with respect to y_k,

$$=> \frac{\partial E^n}{\partial y_k} = -t_k^n . \frac{1}{y_k^n} . 1$$

$$=> \frac{\partial E^n}{\partial y_k} = -\frac{t_k^n}{y_k^n} \tag{13}$$

Also, since

$$y_l^n = \frac{e^{a_l^n}}{\sum_{m=1}^{C} e^{a_m^n}}$$

it will have different derivatives when $l = k$ and when $l \neq k$. Thus, the derivatives $\frac{\partial y_k^n}{\partial a_k}$ and $\frac{\partial y_{k'}^n}{\partial a_k}$ ($l \neq k$ case) can be written as:

$$\frac{\partial y_k^n}{\partial a_k} = \frac{e^{a_k^n}}{\sum_{m=1}^{C} e^{a_m^n}} - e^{a_k^n} \left[\frac{1}{(\sum_{m=1}^{C} e^{a_m^n})^2} \right] e^{a_k^n}$$

$$\frac{\partial y_k^n}{\partial a_k} = y_k^n(1 - y_k^n) \tag{14}$$

And,

$$\frac{\partial y_{k'}^n}{\partial a_k} = -e^{a_{k'}^n} e^{a_{k'}^n} \frac{1}{(\sum_{m=1}^{C} e^{a_m^n})^2}$$

$$\frac{\partial y_{k'}^n}{\partial a_k} = -(y_{k'}^n)^2 \tag{15}$$

Putting (13), (14) and (15) in (12),

$$\delta_k^n = -\frac{\partial E^n}{\partial a_k^n} = -\sum_{l=1 \& l \neq k}^{C} (\frac{t_l^n}{y_l^n}(y_l^n)^2) - (\frac{t_k^n}{y_k^n}(y_k^n(1 - y_k^n)))$$

$$\delta_k^n = -\sum_{l=1 \& l \neq k}^{C} (t_l^n y_k^n) - (t_k^n(1 - y_k^n)) \tag{16}$$

Following the one hot representation, only one of the labels from $l = 1$ to C in t^n would be 1 and all the others would be 0. Therefore, for input term u where t_k^u is 1, the derivative becomes:

$$\frac{\partial E^u}{\partial a_k^u} = y_k^u - 1$$

$$=> \frac{\partial E^u}{\partial a_k^u} = y_k^u - t_k^u \tag{17}$$

For input term v where some $t_{k'}^n$ is 1 such that $k' \neq k$, this term becomes:

$$\frac{\partial E^v}{\partial a_k^v} = y_k^v$$

$$=> \frac{\partial E^v}{\partial a_k^v} = y_k^v - t_k^v \tag{18}$$

Therefore, combining both the cases together, we can form an expression for δ_k^n as:

$$\delta_k^n = -\frac{\partial E^n}{\partial a_k^n} = t_k^n - y_k^n \tag{19}$$

Now, we wish to derive an expression for δ_j. Note that the derivation below is taken from Professor Gary's class notes.

We have,

$$\delta_j^n = -\frac{\partial E^n}{\partial a_j^n}$$

Using chain rule and summation over all labels,

$$\delta_j^n = -\sum_{k=1}^{C} \frac{\partial E^n}{\partial y_k^n} \frac{\partial y_k^n}{\partial a_j^n}$$

$$=> \delta_j^n = -\sum_{k=1}^{C} \frac{\partial E^n}{\partial y_k^n} \frac{\partial y_k^n}{\partial y_j^n} \frac{\partial y_j^n}{\partial a_j^n}$$

Taking the terms independent of k outside of summation,

$$=> \delta_j^n = -\frac{\partial y_j^n}{\partial a_j^n} \sum_{k=1}^{C} \frac{\partial E^n}{\partial y_k^n} \frac{\partial y_k^n}{\partial y_j^n}$$

Using chain rule again,

$$=> \delta_j^n = -\frac{\partial y_j^n}{\partial a_j^n} \sum_{k=1}^{C} \frac{\partial E^n}{\partial y_k^n} \frac{\partial y_k^n}{\partial a_k^n} \frac{\partial a_k^n}{\partial y_j^n}$$

67 Combining terms using chain rule, we get:

$$=> \delta_j^n = -\frac{\partial y_j^n}{\partial a_j} \sum_{k=1}^{C} \frac{\partial E^n}{\partial a_k^n} \frac{\partial a_k^n}{\partial y_j^n} \tag{20}$$

Since,

$$a_k^n = \sum_j w_{jk} y_j$$

68 we can write the derivative $\frac{\partial a_k^n}{\partial y_j^n}$ as:

$$\frac{\partial a_k^n}{\partial y_j^n} = w_{jk}$$

69 Substituting in (20):

$$\delta_j^n = -\frac{\partial y_j^n}{\partial a_j^n} \sum_{k=1}^{C} \delta_k w_{jk} \tag{21}$$

Since we are using the sigmoid function as activation, we can say that:

$$y_j^n = \sigma(a_j^n)$$

$$=> \frac{\partial y_j^n}{\partial a_j^n} = \sigma(a_j^n)(1 - \sigma(a_j^n))$$

$$=> \frac{\partial y_j^n}{\partial a_j^n} = y_j^n(1 - y_j^n)$$

Substituting back in equation (21):

$$\delta_j^n = -y_j^n(1 - y_j^n) \sum_{k=1}^{C} \delta_k w_{jk}$$

which can also be written as: (without substituting value of $\frac{\partial y_j^n}{\partial a_j^n}$, as done in class):

$$\delta_j^n = -\frac{\partial y_j^n}{\partial a_j^n} \sum_{k} \delta_k w_{jk}$$

70 For the final part of this problem, we need to derive the update rule for w_{ij} and w_{jk}.

Looking at the update rule for w_{jk} first, we have:

$$w_{jk} = w_{jk} - \eta \frac{\partial E}{\partial w_{jk}}$$

Using chain rule,

$$=> w_{jk} = w_{jk} - \eta \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial w_{jk}}$$

Since we need to repeat this for all data points,

$$=> w_{jk} = w_{jk} - \eta \sum_{n=1}^{N} \frac{\partial E^n}{\partial a_k^n} \frac{\partial a_k^n}{\partial w_{jk}}$$

From the definitions of δ_k and z_j, we can write:

$$=> w_{jk} = w_{jk} + \eta \sum_{n=1}^{N} \delta_k^n z_j^n \tag{22}$$

From equations derived above for previous parts of question 2, we have:

$$=> w_{jk} = w_{jk} + \eta \sum_{n=1}^{N} (t_k^n - y_k^n) z_j^n \tag{23}$$

Similarly, we need to work for the update rule for w_{ij}. Using the generic rule, we have:

$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

Using chain rule,

$$=> w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}}$$

Changing to summation over all input points form, we get:

$$=> w_{ij} = w_{ij} - \eta \sum_{n=1}^{N} \frac{\partial E^n}{\partial a_j^n} \frac{\partial a_j^n}{\partial w_{ij}}$$

Substituting the value of δ_j, as computed previously,

$$=> w_{ij} = w_{ij} + \eta \sum_{n=1}^{N} \delta_j^n z_i^n \tag{24}$$

$$=> w_{ij} = w_{ij} + \eta \sum_{n=1}^{N} y_j^n (y_j^n - 1) \sum_{k=1}^{C} \delta_k^n w_{jk} z_i^n$$

$$=> w_{ij} = w_{ij} + \eta \sum_{n=1}^{N} y_j^n (y_j^n - 1) \sum_{k=1}^{C} (t_k^n - y_k^n) w_{jk} z_i^n \tag{25}$$

Finally, we have been asked to write the vector representation of the equations we just derived.

Let X be the input matrix of dimension N x I, where I is the number of input features and N is the number of examples in our data-set. Note that if we add the bias term as well, the dimension of X changes to N x $(I+1)$. We define W_{ij} as the weights matrix from input layer to hidden layer, having dimensions $(I+1)$ x J. Similarly, we have another weights matrix W_{jk} which includes weights from the hidden layer to the output layer. Given the matrix form, a row in W matrix holds weight parameters from that starting unit to all units in the next layer.

Continuing our general representation, we define Z^l as the output of layer l. Thus, we can write:

$$Z^1 = X$$

$$A^{l+1} = W^l Z^l$$

$$Z^{l+1} = \sigma(A^{l+1})$$

where W^l is W_{ij} when $l = 0$ and W_{jk} when $l = 1$. Note that we can have used σ or the sigmoid activation function above while it can be generalized for any other activation function as well.

Now, we re-write our update rules from equations (22) and (24) generically using a vectorized form as follows:

$$W^l = W^l + \eta(\delta^{l+1}(Z^l)^T) \tag{26}$$

[As stated above, W^l is W_{ij} when $l = 0$ and W_{jk} when $l = 1$] where

$$\delta^l = (W^l)^T \delta^{l+1} .* \sigma'(Z^l) \tag{27}$$

and δ for last layer (output) is δ^k, written as:

$$\delta^k = T - Y \tag{28}$$

In (27), the operator $.*$ means element wise product. In (28), T is the actual output whereas Y is the output predicted by our model, and is obtained using the Z value of last layer.

2.3 Results

We derived the expression for δ_k^n as:

$$\delta_k^n = t_k^n - y_k^n$$

and the expression for δ_j^n as:

$$\delta_j^n = -\frac{\partial y_j^n}{\partial a_j^n} \sum_k \delta_k w_{jk}$$

The update rule for w_{ij} is given as:

$$w_{ij} = w_{ij} + \eta \sum_{n=1}^{N} y_j^n (y_j^n - 1) \sum_{k=1}^{C} (t_k^n - y_k^n) w_{jk} z_i^n$$

and that for w_{jk} is written as:

$$w_{jk} = w_{jk} + \eta \sum_{n=1}^{N} (t_k^n - y_k^n) z_j^n$$

The vectorized form of update rule can be written as:

$$W^l = W^l + \eta(\delta^{l+1}(Z^l)^T)$$

where

$$\delta^l = (W^l)^T \delta^{l+1} .* \sigma'(Z^l)$$

and

$$\delta^k = T - Y$$

2.4 Discussion

All the results above seem to be self explanatory and are derived mathematically using known facts. The expressions above are hence genuine and will be used by us for the remaining of this report. The computation is much faster when we update all w_{ij}s and w_{jk}s at the same time using matrix multiplications rather than **for** loops.