CSE 190 - Neural Network - Homework 1

Justin Liu

October 15, 2015

1 Perceptron

1.1 Decision Boundary

Since y = 1 when:

$$\sum_i w_i x_i >= \theta$$

$$\sum_i w_i x_i - \theta >= 0$$

$$\sum_i w_i x_i + w_0 >= 0$$

In 2D: $w_0 + w_1 x_1 + w_2 x_2 >= 0$

So the decision boundary is:

$$y(x) = 0 \tag{1}$$

The distance from $\sum_i w_i x_i + w_0 = 0$ to the origin is $\frac{|\sum_i w_i * 0 + w_0|}{\|w\|}$.. Therefore, l = $\frac{|w_0|}{\|w\|}$

$$l = \frac{w^T x}{\|w\|} \tag{2}$$

1.2 Table

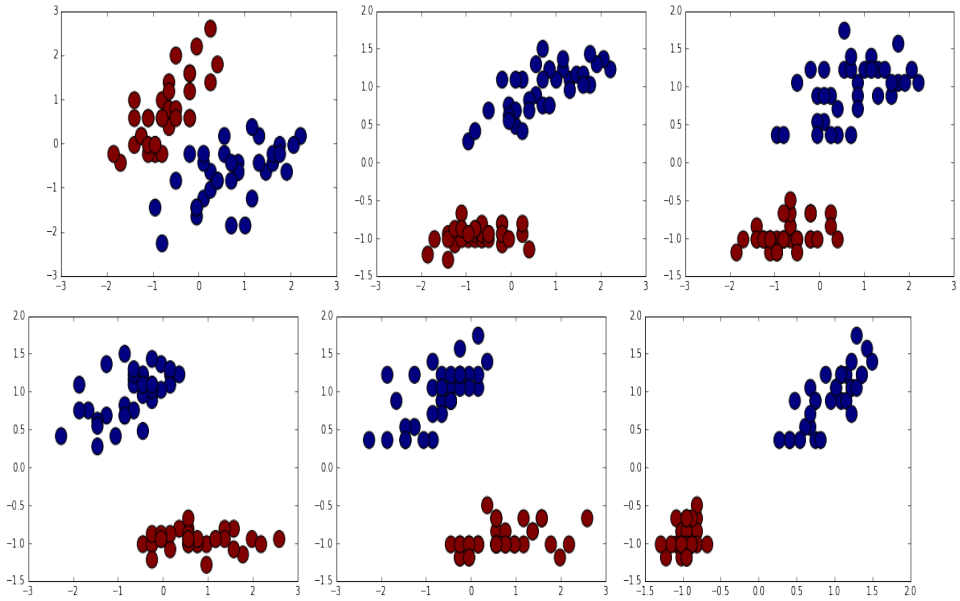
$$w_{i,t+1} = w_{i,t} + (teacher - output)x_i \tag{3}$$

x1	x2	w0	w1	net	out	teacher	θ
1	1	0	0	0	1	0	0
1	1	-1	-1	-2	0	0	1
0	0	-1	-1	0	0	1	1
0	1	-1	-1	-1	0	1	0
1	1	-1	0	-1	1	0	-1
0	1	-2	-1	-1	0	1	0
1	0	-2	0	-1	0	1	-1
1	1	-1	0	-1	1	0	-2
1	0	-2	-1	-2	0	1	-1
1	1	-1	-1	-2	1	0	-2
0	1	-2	-2	-2	0	1	-1
1	1	-2	-1	-3	0	0	-2
0	1	-2	-1	-1	1	1	-2
1	0	-2	-1	-2	1	1	-2
0	0	-2	-1	0	1	1	-2

Of course the solution is not unique, if we plot out all possible points in 2D, then it's easy to see that there are infinite many lines that can separte the two groups.

1.3 Perceptron

Z-score is useful because it standardizes the data such that it is based on how much below or above the standard deviations. Because it transforms the data linearly, it won't break the inner structure of the data and further make them be more accssible for special computation. (In case the data value is huge, ex: 10000000, z-score will reduce it to a reasonable value, say, 1.7)



From left to right, top to bottom: sl vs sw, sl vs pl, sl vs pw, sw vs pl, sw vs pw, pl vs pw. Color = label

Clearly, by taking all permuataion of the input features, each pair is linearly separable.

Rate	K	Error
0.0001	1	0.0
0.0001	10	0.0
0.0001	100	0.0
0.0001	1000	0.0

Rate	K	Error
0.01	1	0.0
0.01	10	0.0
0.01	100	0.0
0.01	1000	0.0

Rate	K	Error
0.1	1	0.0
0.1	10	0.0
0.1	100	0.0
0.1	1000	0.0

Rate	K	Error
0.2	1	0.0
0.2	10	0.0
0.2	100	0.0
0.2	1000	0.0

Rate	K	Error
0.5	1	0.0
0.5	10	0.0
0.5	100	0.0
0.5	1000	0.0

Rate	K	Error
0.9	1	0.33
0.9	10	0.0
0.9	100	0.0
0.9	1000	0.0

Rate	K	Error
0.99	1	0.0
0.99	10	0.0
0.99	100	0.33
0.99	1000	0.0

Rate	K	Error
0.9999	1	0.0
0.9999	10	0.33
0.9999	100	0.0
0.9999	1000	0.0

In general, there seems to be more error given a higher learning rate and more iterations

2 Logistic and Softmax Regression

2.1 Gradient for Logistic

Start derive:

$$\frac{\partial E(\theta)}{\partial \theta_j} = - \sum_{i=1}^{N} (y^i \frac{1}{\sigma(\theta^T x^i)} - (1-y^i) \frac{1}{1-\theta^T x^i}) * \frac{\partial}{\partial \theta_j} \sigma(\theta^T x^i)$$
$$\frac{\partial E(\theta)}{\partial \theta_j} = - \sum_{i=1}^{N} (y^i \frac{1}{\sigma(\theta^T x^i)} - (1-y^i) \frac{1}{1-\theta^T x^i}) * \sigma(\theta^T x^i)(1-\sigma(\theta^T x^i)) * \frac{\partial}{\partial \theta_j} (\theta^T x^i)$$
$$\frac{\partial E(\theta)}{\partial \theta_j} = - \sum_{i=1}^{N} (y^i \frac{1}{\sigma(\theta^T x^i)} - (1-y^i) \frac{1}{1-\theta^T x^i}) * \sigma(\theta^T x^i)(1-\sigma(\theta^T x^i)) * x_j^i$$
$$\frac{\partial E(\theta)}{\partial \theta_j} = - \sum_{i=1}^{N} (y^i (1-\sigma(\theta^T x^i)) - (1-y^i))(\sigma(\theta^T x^i)) * x_j^i$$
$$\frac{\partial E(\theta)}{\partial \theta_j} = - \sum_{i=1}^{N} x_j^i (y^i - \sigma(\theta^T x^i))$$
$$\frac{\partial E(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} x_j^i (\sigma(\theta^T x^i) - y^i)$$

2.2 Gradient for Softmax

Take one example:

$$\text{A} = \sum_{j=0}^{K} 1 * (y=j)(log(e^{\theta^{j,T} x}) - log(\sum_{l=0}^{K} e^{\theta^{l,T} x}))$$
$$\text{A} = \sum_{j=0}^{K} 1 * (y=j)(\theta^{j,T} x - log(\sum_{l=0}^{K} e^{\theta^{l,T} x}))$$
$$\text{A} = \sum_{j=0}^{K} 1 * (y=j)(\theta^{j,T} x - log(e^{\theta^{0} x} + e^{\theta^{1} x} + ... + e^{\theta^{K} x}))$$
$$\frac{\partial A}{\partial \theta^k} = 1 * (y=k)[x - P(y=k|x,\theta)x]$$
$$\frac{\partial A}{\partial \theta^k} = x * [1 * (y=k) - 1 * (y=k) P(y=k|x,\theta)]$$
$$\frac{\partial A}{\partial \theta^k} = x * [1 * (y=k) - P(y=k|x,\theta)]$$
$$\bigtriangledown \theta^k E(\theta) = - \sum_{i=1}^{N} \frac{\partial A}{\partial \theta^k} = - \sum_{i=1}^{N} x * [1 * (y=k) - P(y=k|x,\theta)]$$

2.3 Logistic Regression

Set learning rate = 0.0000001 and epoches = 500, obtained the results (10 way classifcation)
Used SGD for faster convergence
Here is a sample result:

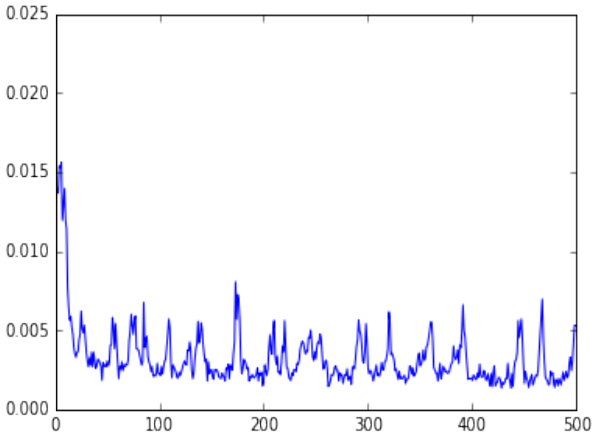
Digit	Rate	Test Error
0	10^-7	0.011
1	10^-7	0.0075
2	10^-7	0.027
3	10^-7	0.032
4	10^-7	0.0285
5	10^-7	0.032
6	10^-7	0.0225
7	10^-7	0.0305
8	10^-7	0.2995
9	10^-7	0.041

Overall Test Error: 0.16565

Interestly, the algorithm classfies 0 and 1 pretty well while having a hard time classifying 8, 9 and 3. This observation is reasonable !

2.4 Softmax Regression

Using same learning rate of 0.0000001, epoch = 500, SGD, obtained the training plot as follow:



And the overall test error rate: 0.13

The test accuracy is slightly better than logistic regression, but it depends on your data as well as your training methodology. In general, I would say they are pretty close giving the same paramters.

3 Conclusion

In conclusion, we are introduced to perceptron and logsitic regression (both binary and multi-class classification) algorithms. Overall it was a good exercise. Some of the things worth noticing:

3.1 SGD vs Batch

Batch mode is slow and may cause exp overflow, so I used a randomly chosen subset of the data to train the model. It is much faster.

3.2 Accuracy

In the Mnist experiment, muti-class classfication seems to perform slightly better than pre-processed binary classfication. But in general, accuracy is depend on many factors such as training iterations, adaptive learning rate...etc.

3.3 Suggestion

For future class homework on Mnist, we should specify sensitivity and specificity because the number of labels of 0 and 1 are unbalanced.

Appendices