Khizr Ali Pardhan | 301314376 |March 6th 2020

CMPT 419 – Assignment 2

1 Softmax for Multi-Class Classification

1)

After setting X_1 and X_2 to 0, as that is the point of intersection, we get the following
e^A_1 = 2.71828182846
e^A_2 = 7.38905609893
e^A_2 = 7.38905609893

$$P(C_1|x) = \frac{e^1}{e^1 + e^2 + e^2} = 0.15536$$

$$P(C_2 OR\ C_3|x) = \frac{e^2}{e^1 + e^2 + e^2} = 0.42232$$

There is a 0.155 chance of the point being in region 1
There is a 0.422 chance of the point being in region 2
There is a 0.422 chance of the point being in region 3

2) The probabilities of the two classes that border with the activation will increase or decrease, depending on if they move to/away from the tri-intersection. The probability of the third region will be adjective, such that the sum of the probabilities adds to 1.0.

Suppose we go along the red line between regions 2 and 3, away from the green point, which is where we have the highest uncertainty. Regions 2 and 3 have a probability of 0.422 at the center, this number will increase as we move away from the tri-intersection, while the probably of region 1 will decrease.

3) If we move far away from the intersection point, staying in the middle of region_i, p(C_i|x) will be higher than both the remaining classes. The center will be the maximum point, if the underlines mechanics are gaussian distributions. While Other will approach 0, or be relatively little.

Note that activation Boundaries at tails of the normal distribution.

P(c_i|x) will increase as be get far from the decision boundaries. If we are in the center, we will be relatively high.

2 Error Backpropagation

1)

Note that

y_n *can be denoted by* a_1^3

$\frac{\partial z}{\partial a} = \mathrm{h`(a_j)}$

$\frac{\partial a}{\partial w} = \mathrm{z = h(a_j)}$

$$\frac{\partial E_n(w)}{\partial a_1^{(3)}} = 2 * \frac{1}{2}(y_n - t_n) = (y_n - t_n) = \delta_1^{(3)}$$

Here H(x) is the identity function; used (PRML 5.49, pg 243)

$$\frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = \delta_1^{(3)} * \frac{\partial a_1^{(2)}}{\partial w_{12}^{(2)}} = \delta_1^{(3)} * z_2^{(2)}$$

$$\frac{\partial E_n(w)}{\partial w_{12}^{(3)}} = \delta_1^{(3)} * z_2^{(2)} = \delta_1^{(3)} * h\left(a_2^{(2)}\right) = \delta_1^{(3)} * a_2^{(2)}$$

b)

z=h(x); h(x) here is the logistic function. It's derivative is h(x) * (1-h(x))

Here $\frac{\partial a_1^{(2)}}{\partial w_{11}^{(2)}} = z_1^1$

$$\frac{\partial E_n(w)}{\partial a_1^2} = \delta_1^{(3)} * \frac{\partial a_1^3}{\partial z_1^2} * \frac{\partial z_1^2}{\partial a_1^2} = \delta_1^{(3)} * w_{11}^2 * (z_1^2 * (1 - z_1^2)$$

$$\frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = \frac{\partial E_n(w)}{\partial a_1^2} * \frac{\partial a_1^2}{\partial w_{12}^{(2)}}$$

$$\frac{\partial E_n(w)}{\partial w_{11}^{(2)}} = \left(\delta_1^{(3)} * w_{11}^2 * \left(z_1^2 * (1 - z_1^2)\right)\right) * z_1^1$$

c)

z=h(x); h(x) here is the logistic function. It's derivative is h(x) * (1-h(x))

Here $\frac{\partial a_1^{(2)}}{\partial w_{11}^{(2)}} = z_1^1$

$$\frac{\partial E_n(w)}{\partial a_1^{(1)}} = \delta_1^{(1)} = h`(a_1) * \sum_k \left(w_{k1}^1 \delta_k^2\right)$$

$$\frac{\partial E_n(w)}{\partial a_1^{(1)}} = \delta_1^{(1)} = h(a_1)\left(1 - h(a_1)\right) * \sum_k \left(w_{k1}^1 \delta_k^2\right)$$

$$\frac{\partial E_n(w)}{\partial a_1^{(1)}} = h(a_1)\left(1 - h(a_1)\right)\left[w_{k1}^1\right]^T \delta_k^2; in\ vector\ notation$$

From (5.67) on pg 246 of PRML we get

$$\frac{\partial E_n(w)}{\partial w_{11}^{(1)}} = \delta_1^{(1)} z_1^0 = \delta_1^{(1)} x_1$$

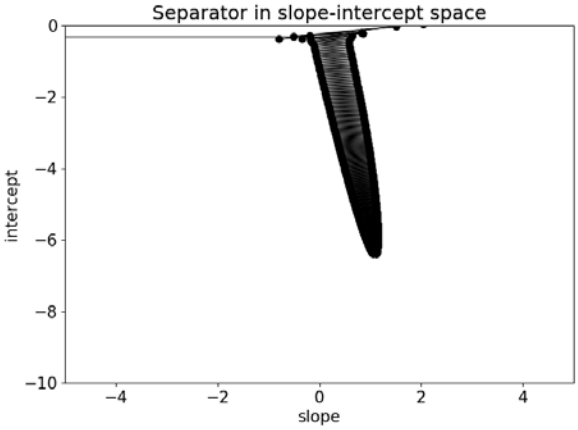
$$\frac{\partial E_n(w)}{\partial w_{11}^{(1)}} = \left(h(a_1)\left(1 - h(a_1)\right) * \sum_k \left(w_{k1}^1 \delta_k^2\right) \right) * z_1^0 = \delta_1^{(1)} x_1$$

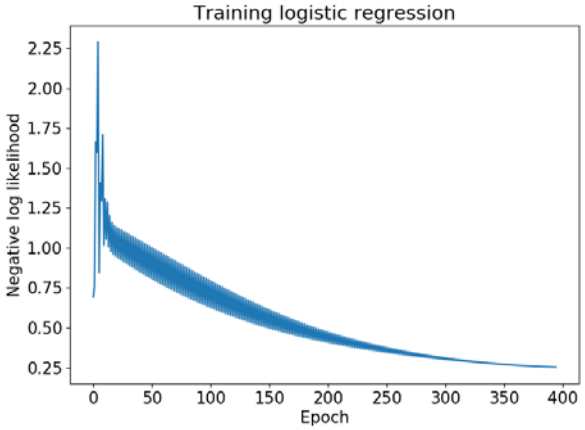
3 Logistic Regression

1.

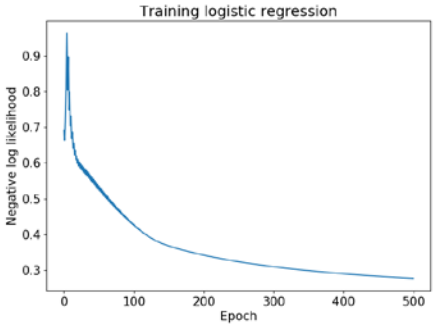
These plots are oscillating because, due to the choice of learning rate. If we move from eta = 0.5 to 0.3, the oscillating will diminish significantly. Some oscillating is expected, however lots of oscillation is detrimental.

A too high of a learning rate many not find the/a minima, while too low of a learning rate many take more than otherwise necessary.



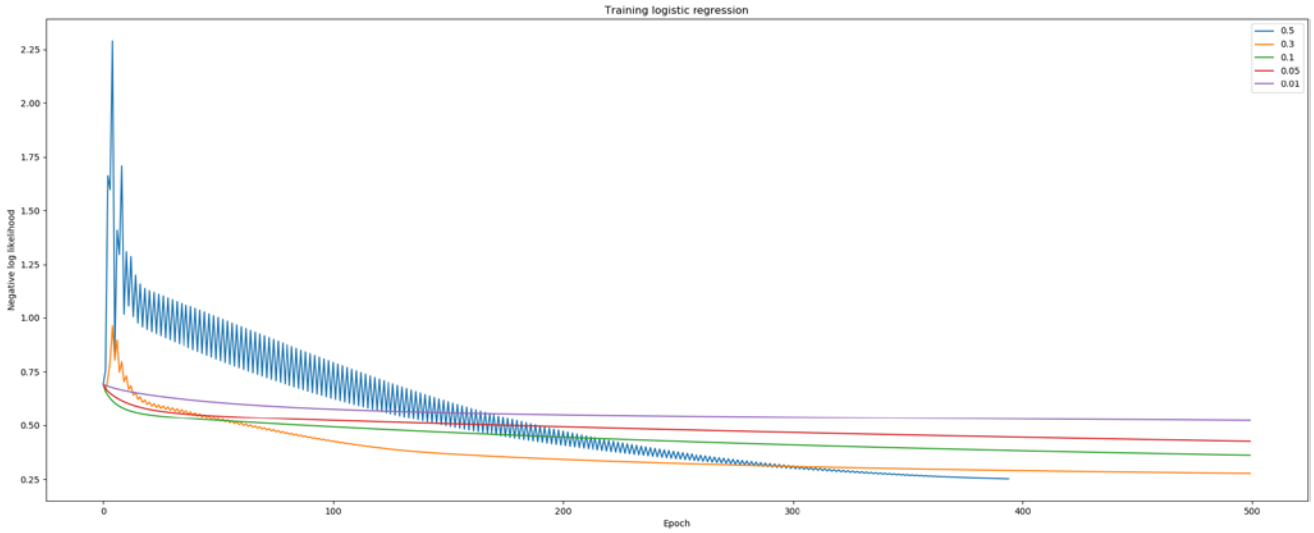


The below is a plot when we set eta to 0.3, some is present, oscillating as expected, but there is relatively little

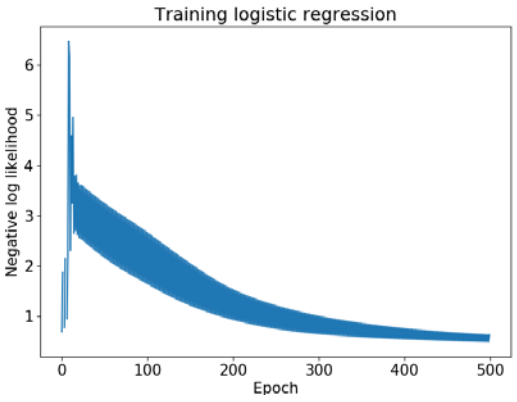


2.

Above is a plot of different learning rates, negative log likelihood to Epoch. Lower learning rates appear to take longer to converge, conversely, higher learning rates appear to take less time to converge. This is not a universal truth, here is a counter example. Eta = 0.5 was the best by over 100 epochs, is very import!



Eta = 1.5 takes longer than eta = 0.5, as seen below.



We want an eta that is not too high and not too low. A higher eta appears to increase oscillation, with intuitively makes sense, as it 'overshoots'. Therefore, some more advance methods use a variable step size.

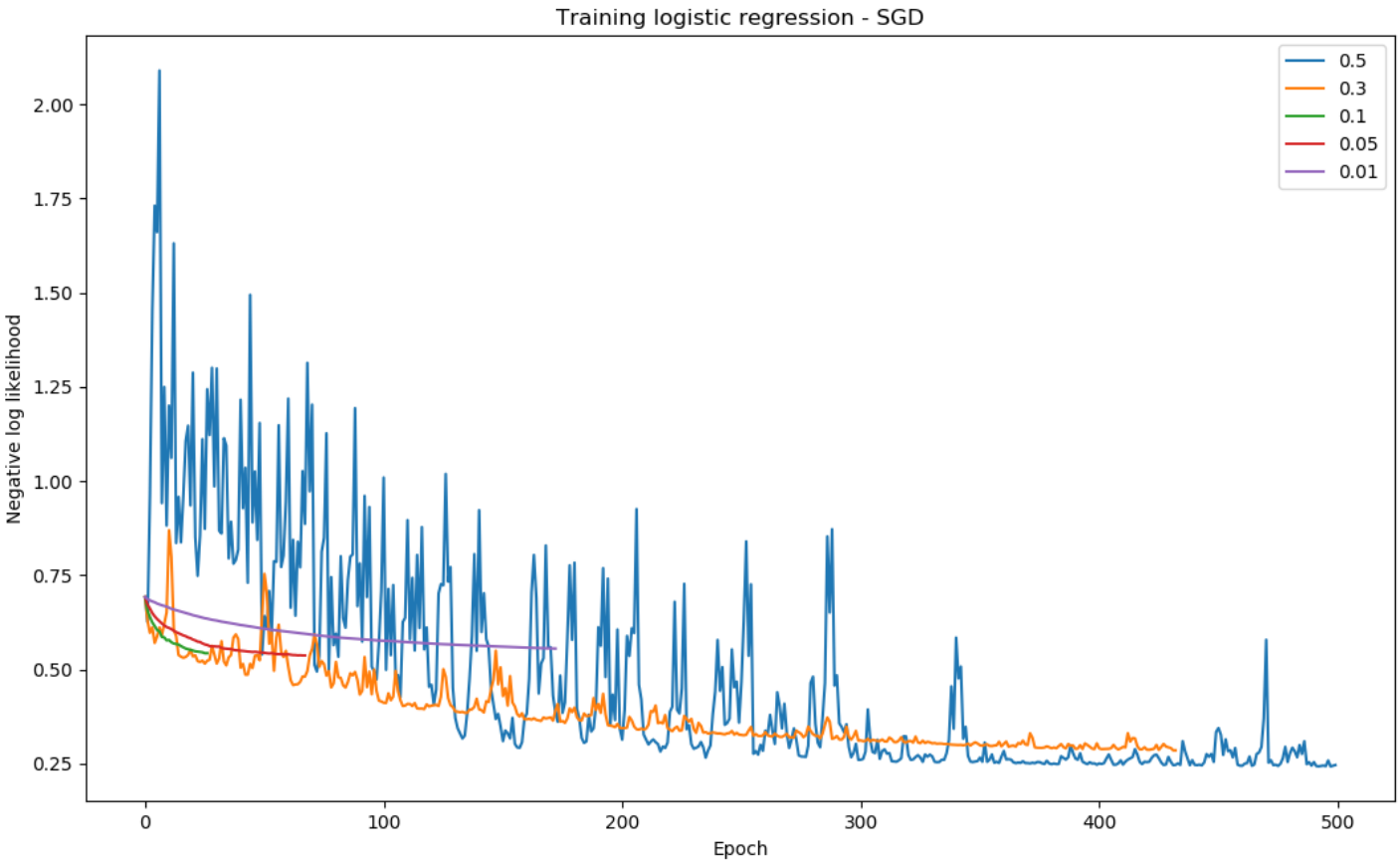
3.

Below is my plot of Stochastic gradient descent, with a single batch size. As per our textbook which states, "*stochastic gradient descent*, makes an update to the weight vector based on one data point at a time" (pg 240; PRML), I used a batch size of one.

From my informal testing, All five etas either converges reach max epochs faster on Gradient decent (runs about three times faster), than on stochastic gradient decent. Stochastic gradient decent takes less epochs on average. With the termination conditions being epoch = 500, or tol = 0.00001, the defaults. Convergence in stochastics gradient decent happens in most cases, in significantly less epochs. An epoch is my no means a (proper) measure of runtime.

In terms of runtime, for my code, Gradient decent is faster, however, in both theory and number of epochs, stochastics gradient decent is faster. We are plotting epochs on the x axis, so I will use that as my measurement.

I conclude stochastic gradient decent is faster.



It is interesting to note that our middle-zone learning rates, eta = 0.1, 0.3, achieved a low log-loss in relatively less epochs.

If we where the remove the exit condition, `tol`, we can gain more insights. All lower learning rates converge at a sub-par local minimum.

4 Fine-Tuning a Pre-Trained Network

I choose task #2, Run validation of the model every few training epochs on validation or test set of the dataset and save the model with the best validation error.

Please refer to the included readme for the rest for the information