WASP: AIML Module 2

June 2022

1 Logistic Regression

Logistic regression uses an s-shaped function that maps a set up inputs to a value between 0 and 1, defined by:

$$p(y = 1|t) = \frac{e^z}{1 + e^z}$$

where z is a function, of some weights, inputs and biases (w, x, b, respectively), such that

$$z(x) = w \cdot x + b$$

and

$$y(x) = 0 \text{ if } z(x) < .5, \text{ else } 1$$

The loss can be defined as:

$$L(x) = \frac{1}{n} \sum_{i=1}^{n} log\Bigg(1 + e^{-y_i(x)}\Bigg)$$

where n is the number of samples. Our objective is to minimize the total loss across all of the samples by changing the parameters w, b. Since this function is 'nice' in the Calculus sense, we can minimize it analytically, using the gradient.

1.1 Automatic Differentiation

In addition to the three methods implemented below, I verified my implementation against an implementation taken from the 'autograd' documentation [1]. Weights from my implementation were (.303, -1.31) while weights from the autograd implementation are (.304, -1.313). The difference is due to differences in the floating point operations between the two methods. Both implementations can be seen in the .ipynb and .py files in this repository.

[1]https://github.com/HIPS/autograd/blob/master/examples/logistic$_regression.py$

2 Experiments

I looked at both the feature mapping experiments as well as the learning rate experiments.

2.1 Learning Rate

In this experiment, we used the original feature map to derive two more based on power transformations of the original data using powers 2 and 3 respectively.

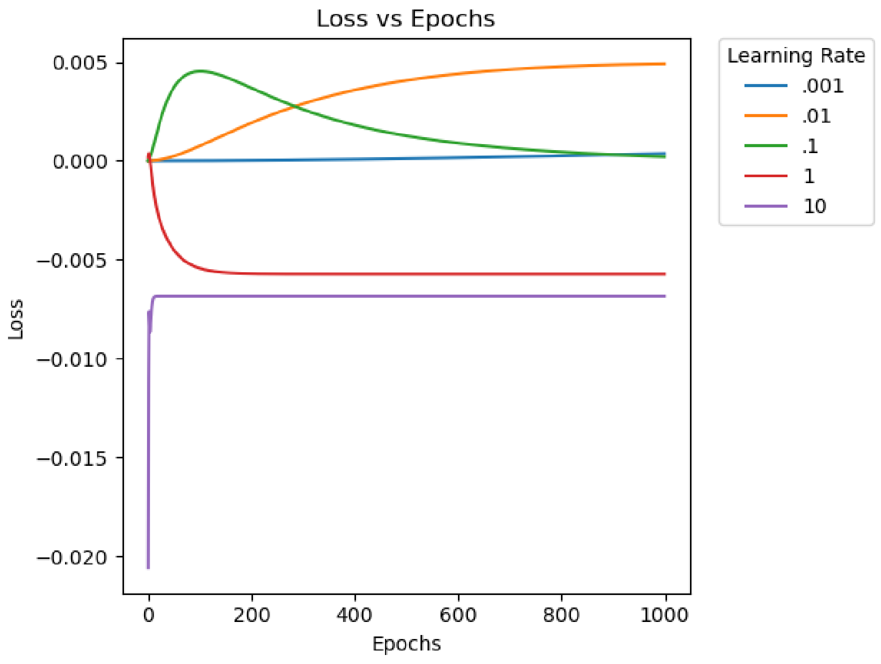


Figure 1: Pictured are three logistic regression models that are trained on various power transformations of the data. As we can plainly see, increasing the learning rate tends to increase the final loss. However, if the learning rate is less than or equal to the Lipschitz constant then a larger step size could offer improved loss for a fixed number of epochs.

2.2 Feature Mapping

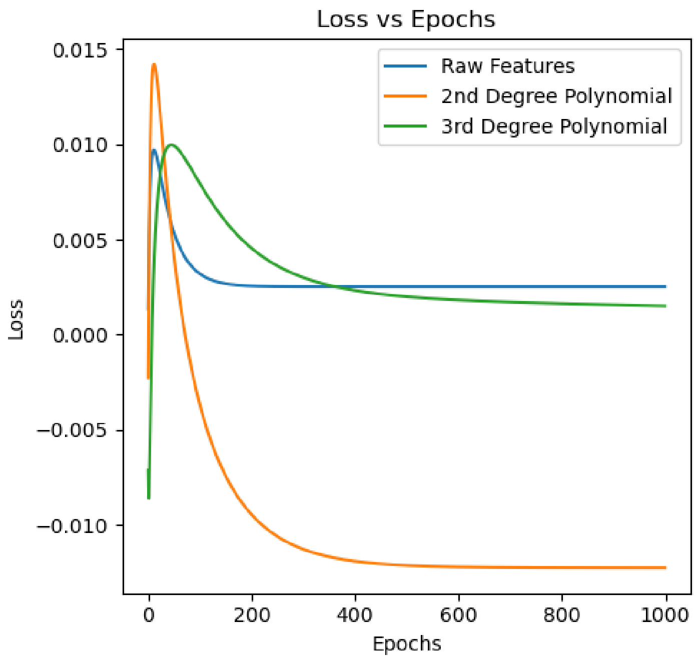


Figure 2: Pictured are three logistic regression models that are trained using various learning rate strategies. As we can see, the raw features and the 3rd degree features converge faster, but they all converge on roughly the same loss, which a slight disadvantage to the 2nd degree transformation.

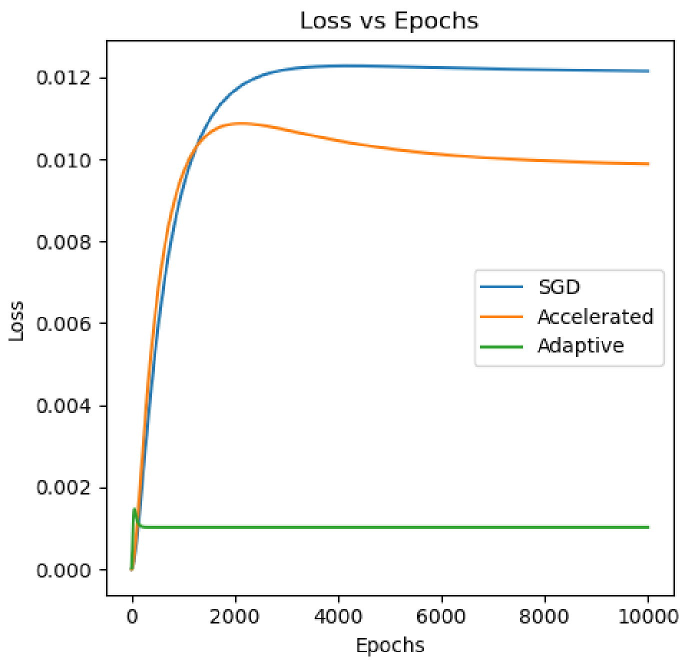
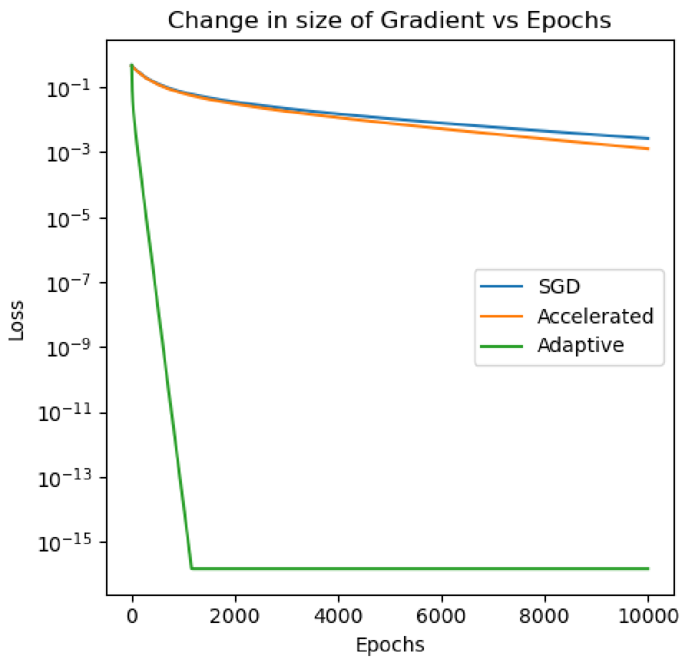


Figure 3: Pictured are three logistic regression models that are trained using various learning rate strategies. As we can see, the adaptive method converges incredibly quickly and nearly to zero while the adaptive method lags behind. The stochastic gradient technique performs the worst, due to the fixed step size preventing a smaller loss.



(Previously I had a bug in the implementation that calculated the magnitude of the partial derivative with respect to a single variable rather than the entire gradient, which explains why the loss looked as expected but the gradient size did not.

Figure 4: Pictured are three logistic regression models that are trained using various learning rate step sizes. The accelerated method takes marginally less time than the stochastic gradient method with a step size = 10 (as optimized above). The adaptive method, however, is much faster. Furthermore, the accelerated and adaptive methods are able to find a smaller absolute loss than the SGD method because the step size isn't fixed, allowing the method to converge in regions that it would 'step-over' with a larger learning rate. The adaptive converges after only a few epochs, but has a larger absolute loss than the accelerated method.
(Previously I had a bug in the implementation that calculated the magnitude of the partial derivative with respect to a single variable rather than the entire gradient, which explains why the loss looked as expected but the gradient size did not.