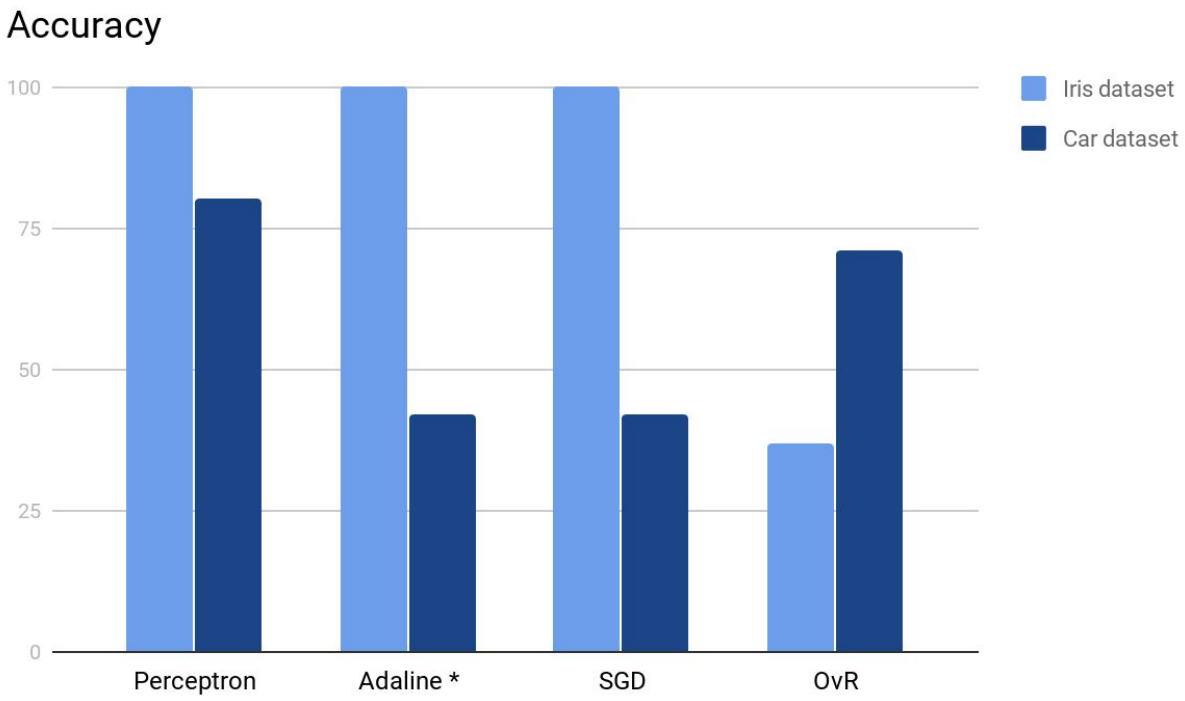
Stepheny Perez
Applied Machine Learning - Project 1

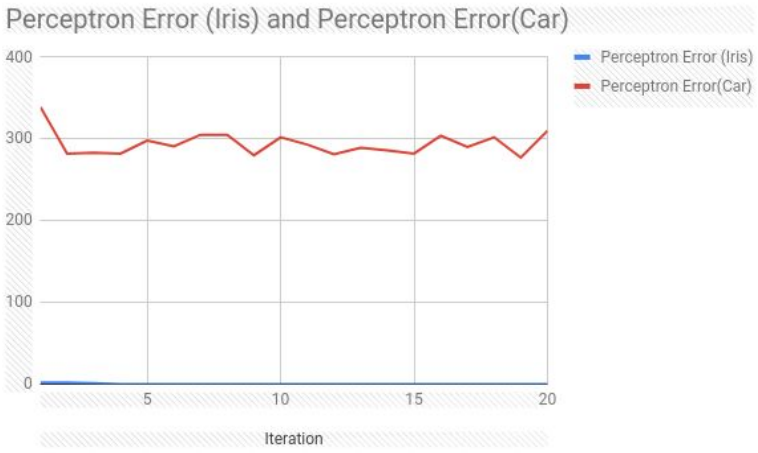
Accuracy

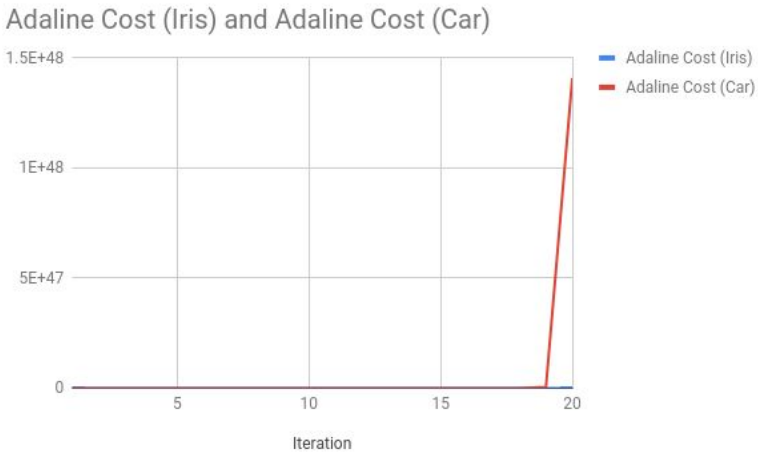
Classifier	Dataset	Accuracy %
Perceptron	iris	100%
Perceptron	car	80.33%
Adaline	iris	100%
Adaline	car	42.24%
SGD	iris	100%
SGD	car	42%
One-Vs-Rest (w/SGD)	iris	37%
One-Vs-Rest (w/SGD)	car	71.03%

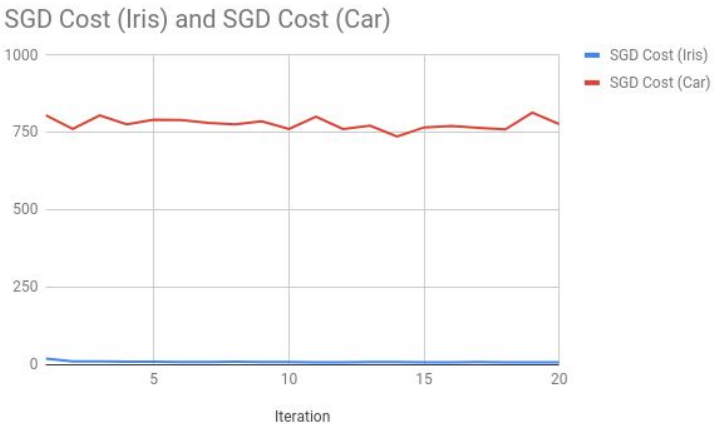
Note: For all classifiers except adaline, this is with all data columns. However, Adaline always behaved strangely with more than 2 attributes (error would increase instead of decrease).



Error and Cost







Note: these values are in the git repository if you want to see the raw data. I manually created these graphs in Excel.

Why is car so much worse?

As made clear above, the car dataset introduces some issues with the classifiers, and they no longer work as well as they do on the iris dataset. There are multiple reasons this would be happening:

1. The dataset is significantly larger.
2. The dataset has more attributes.
3. The dataset has more classes.

I would have to do more testing to find the exact reason. But as you can see, the cost is not decreasing, it's stagnating. This is interesting though, because the accuracy does still increase in most cases with this dataset. It's not clear from the charts, but the error and cost and decreasing in every iteration with the iris dataset, which is to be expected. The exception to this is the adaline classifier, which is increasing with each iteration. This is unexpected behavior, and is probably due my own coding errors.

Feature scaling

Also, I'd like to discuss feature scaling briefly. According to our notes 3.2, we can standardize the data, which allows the data to converge much faster. This is a must-have for Adaline and SGD, as they converged very slowly before adding these lines of code. Due to time constraints I haven't included a graph of this, but it's something I noticed while completing this assignment.

Multi-class classifier

I implemented the OvM classifier with SGD as specified in the assignment. This was surprisingly successful, particularly because the accuracy was higher than SGD, even though it directly uses SGD. OvM and Perceptron were the most effective car classifiers, though obviously OvM provides much more classification information because it considers all possible classes. Also, I did not include a graph of these errors because the classifier is run 3x more, and is no different than SGD. However I did include a file called 'ovr_output.txt' which shows the costs when running this classifier.

Binary classifiers and the 'true' class

I was testing my Perceptron class in the early stages of the assignment, and I noticed that I got significantly different accuracy when I chose 'iris-setosa' vs. 'iris-virginica' as my class that maps to 'true'. With 'iris-setosa' as my 'true' class, my Perceptron accuracy on iris.data is 100%. However, with 'iris-virginica', the accuracy was about 68%, and never converged to 0 in the same way the other did. I also thought it might be because I was originally training on the first 70% of the file, and then testing on the last 30% (meaning there was less training involving 'iris-virginica'). However, even when I changed this, I got similar results. This means that perhaps the data for 'iris-setosa' classes is more meaningful, and stronger conclusions can be drawn from it. With extra time, I would try this with each attribute in each dataset and see how it changes, because I'm sure the results are significant.