A9 - Neural Networks

CS 4300
Fall 2016
Leland Stenquist - 2,4,5
Ryan Keepers - 1,3,5

1. Introduction

For the neural networks assignment we implemented the standard Perceptron algorithm to determine our ability to distinguish between images representing a wumpus (the letter ‘w’ in the image), a pit (the letter ‘p’), and gold (the letter ‘g’). In doing so we sought to answer the following questions:

- Can the algorithm correctly classify each letter independent from the other two letters using cross validation on the entire image set?
- Can the algorithm correctly classify each letter independent from the other two letters by training on randomly selected examples from the image set (more accurately representing an online learning environment).
- Can the algorithm correctly classify each letter when the image is condensed to an euler value?
- Can the algorithm correctly classify each letter by using only subsections of each image, rather than the total image?

2. Method

In our implementation we actually created two versions of the Perceptron Algorithms. One was created shortly after the assignment was released, before we were given a method header, and is based on the version of perceptron that we learned in our Machine Learning class. The other is based the method header and our understanding of what was desired for the assignment.

CS4300_perceptron_learning: This function, based on the header given to us, chooses a random example from a data set. It takes the threshold of the dot product of that example and a weight vector. If the value returned predicts correctly is does nothing, else it updates the weight vector. It then checks the accuracy of this weight vector against all the images. If it is not 100% correct it repeats the process. This is done for max_iters or until it gets 100% accuracy.

CS4300_perceptron: This function acts very similar to the function above except it starts with the first example, makes a prediction based on the threshold of the dot product of the example and a weight vector. It updates under the same circumstances described above. The difference

is that this algorithm does this for each example in the training set and then returns a weight vector to be tested.

We have three drivers that are important to note. On of them runs the perceptron with k-fold cross-validation. The other simply runs CS4300_perceptron_learning and the last generates statistics about different trials counts for the algorithms.

CS4300_a9_driver: This is a simple driver that simply runs the CS4300_perceptron_learning algorithm 3 times to generate 3 weights. And prints out a trace of the algorithm.

CS4300_a9_driver_cross_validation: This driver checks the weights given using k-fold cross validation. This is where it trains on n-k of the examples and then tests on the k examples. It does this for every possible version of k in the training set. For example if k where 3 and n where 12, it would do this 4 times. Each time k would be a new 3 values.

CS4300_a9_stats_driver: This driver runs the algorithm for 1, 2, 4, 8, 16, 32, 64 trials 100 times and uses the accuracies gathered to create means, standard deviations, confidence intervals, and a plot. These statistics were then put in the data and analysis section.

All the other functions are helper functions used to help the drivers or the perceptron algorithm accomplish what it needs to accomplish.

3. Verification of Program

Our algorithm was verified largely through the CS4300_a9_driver_cross_validation function. We wrote the algorithm under the assumption that the training set (composed of 27 images: 9 for each letter w, p and g) was linearly separable. Therefore the perceptron algorithm, which guarantees a finite number of errors to learn any linearly separable dataset, should be able to correctly classify the training dataset.

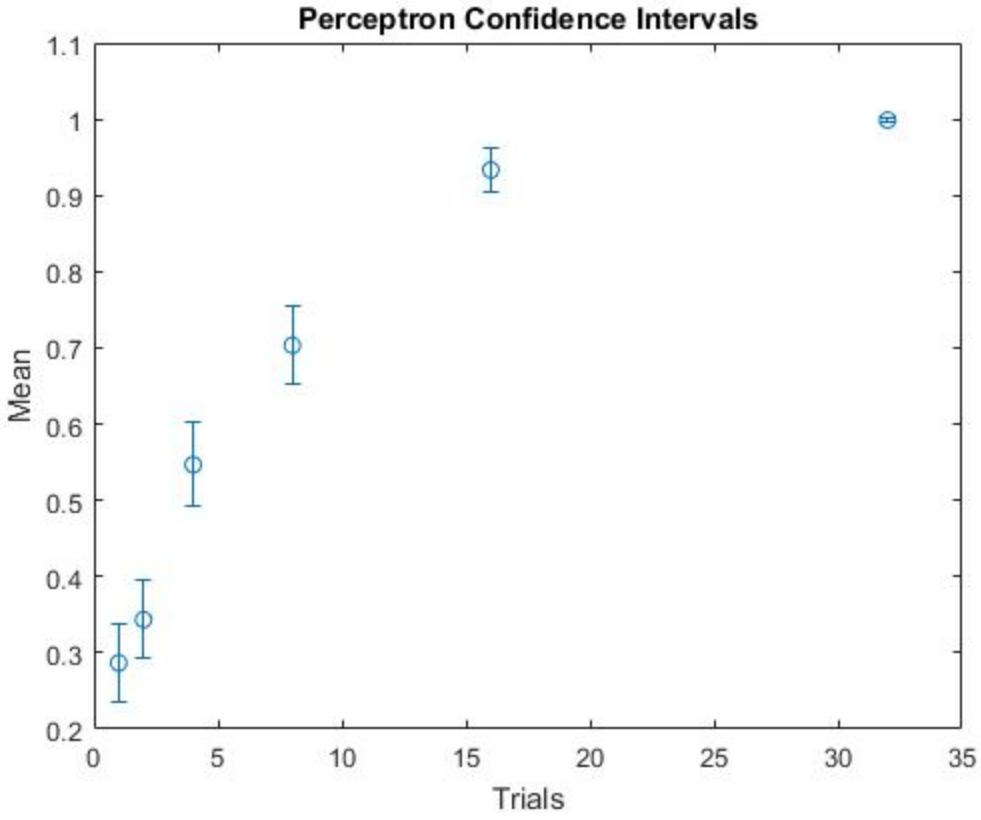
The driver was run on the full set of 27 images with the number of folds set to 3, 9, and 27. For each fold setting, the program was run with the dataset labeled positively for one of the three images. Finally, the resultant weight vectors from training for each image were passed to a classification function: CS4300_classification_success. As part of the labeling process the image structs were provided a field which recorded the true label for each image (pit, wumpus, or gold). Each image was then verified against the three weight vectors, with condition for success being that only one classifier should return positive, and it should be the correct classifier for the image. In all experiments the result was a 100% success rate.

4. Data and Analysis

The data was gathered through a driver by running the experiment for n trials 100 times. The accuracy was gathered for each of those 100 runs. Using that set of 100 accuracies the mean, standard deviation and Confidence interval were gathered.

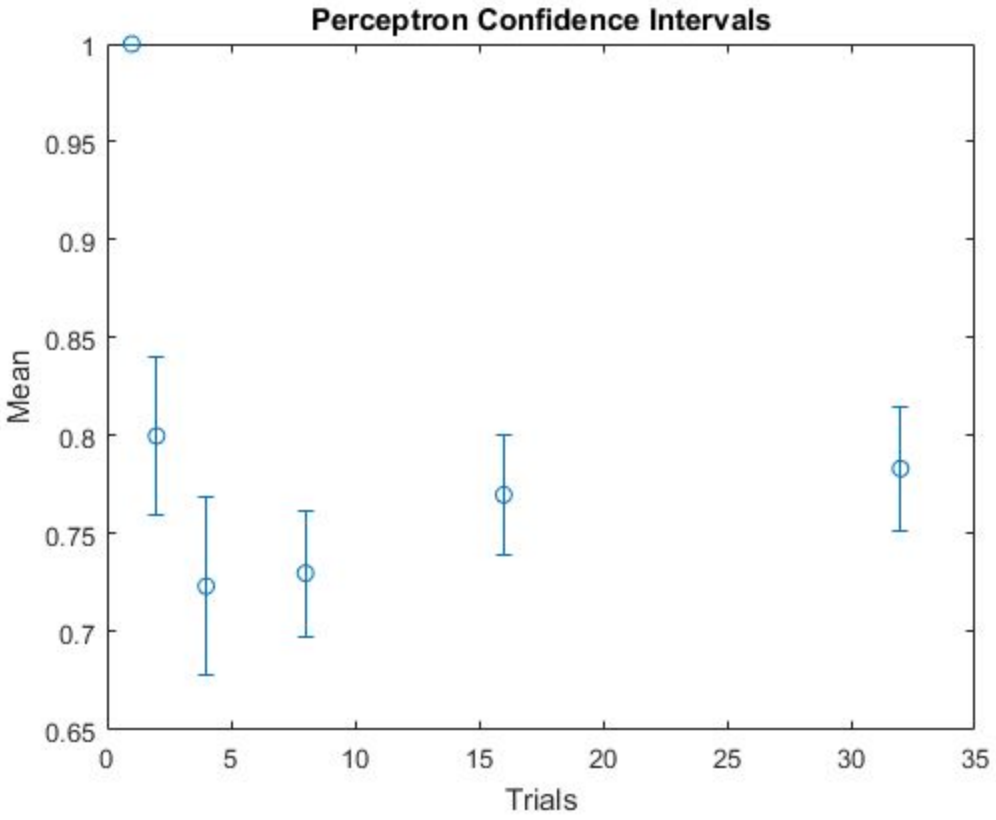
The following figure shows the confidence intervals based on using the image pixels as features.

trials	1	2	4	8	16	32
mean	0.2667	0.31	0.5567	0.75	0.9511	0.9989
Standard Deviation	0.2638	0.2649	0.2640	0.2479	0.1175	0.0111
Confidence Interval	0.2143 - 0.3190	0.2574 - 0.3626	0.5043 - 0.6091	0.7008 - 0.7992	0.9278 - 0.9744	0.9967 - 1.0011



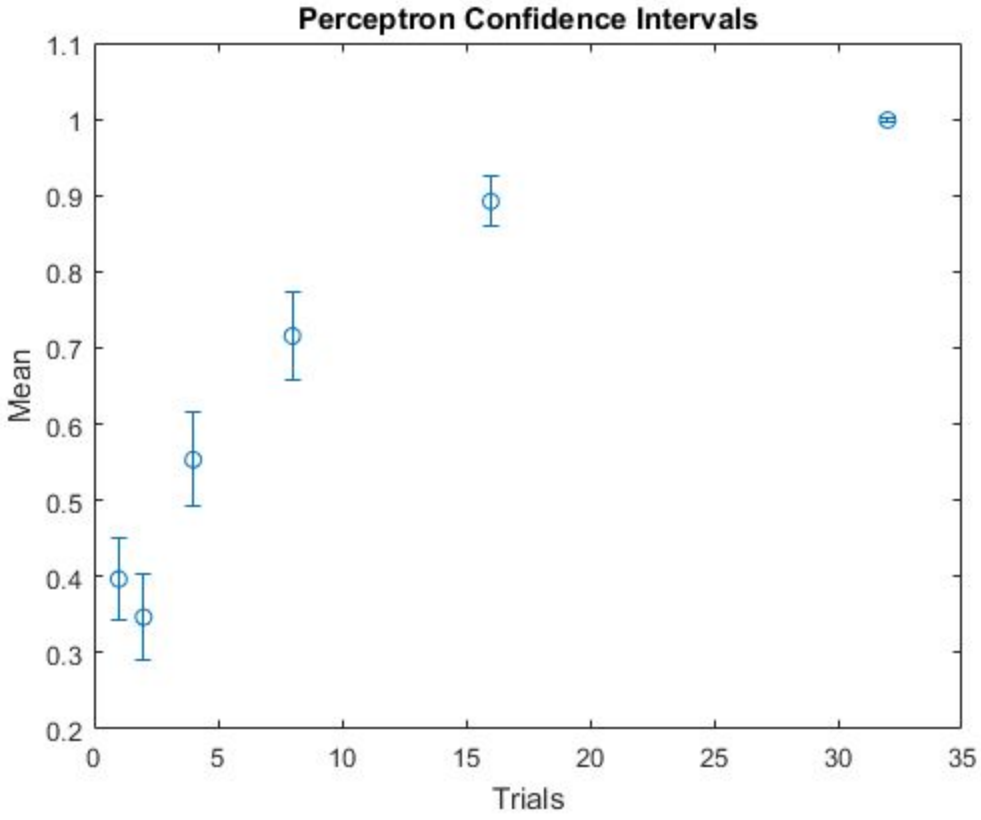
This shows the statistics based on using the Euler value as a feature:

trials	1	2	4	8	16	32
mean	1	0.8	0.7233	0.73	0.77	0.7833
Standard Deviation	0	0.201	0.2275	0.162	0.1549	0.1598
Confidence Interval	1.0000 - 1.0000	0.7601 - 0.8399	0.6782-0.7685	0.6979-0.7621	0.7393-0.8007	0.7516-0.815



This is the confidence interval of breaking the breaking the image into three sub-sections and using those as the features.

trials	1	2	4	8	16	32
mean	0.3967	0.3467	0.5533	0.7156	0.8922	0.9989
Standard Deviation	0.2667	0.2879	0.3115	0.2939	0.1629	0.0111
Confidence Interval	0.3437-0.4496	0.2895-0.4038	0.4915-0.6151	0.6572-0.7739	0.8599-0.9246	0.9967-1.0011



5. Interpretation

We sought originally to verify whether the perceptron algorithm could correctly classify the images under four conditions: First, whether standard perceptron could correctly classify the images using cross validation. Next, if it could classify them correctly by representing an online

learning environment. Finally we wanted to test the online learning environment with two variations on the dataset: the eulerian representation of each image, and a subset of data from each image.

The first case was shown to be successful in the Verification of Program section with CS4300_a9_driver_cross_validation. The second case and the data subset case were both shown successful within the Data and Analysis section. The only case which was shown to not be totally successful was the eulerian representation.

The eulerian representation reduced each image to a single integer representation: -1 for G, 0 for P, and 1 for W. Although geometrically differentiated from each other, these three values cannot be fully learned by perceptron. This is due to the VC dimension limitation of a single linear separator, which happens to be 2, and this particular eulerian representation requires a separator that can handle a VC dimension of 3.

To explain the above argument: the VC dimension is the maximal number of points that can be correctly classified by a separator for every arrangement of labels on the points. In our particular case, G and W are separable because they create a linear sequence to the effect of (1, 0, 0) or (0, 0, 1), respectively, where 1 is a positive label and 0 is a negative label. To correctly separate them, a line only need to be drawn between the 1 and nearest 0. However the pit, P, creates a linear sequence of (0, 1, 0). This cannot be separated by a single line. The best that Perceptron can do in this case is a 66% success rate.

6. Critique

The perceptron is a very powerful algorithm for online learning. If extended to be used in a neural network this algorithm can become very useful. We noticed that the perceptron converges fairly quickly on the data set given if the dataset is linearly separable. This is actually a guarantee of the perceptron algorithm. Given enough examples it will find a weight vector that separates a linear data set. That makes this a very interesting algorithm.

Given that not all data sets are linearly separable we can see examples where the CS4300_perceptron_learning algorithm never finds a classier. This happened when we ran the algorithm against examples with one feature(euler value). Because we were working in 2 dimension and the labels where 1, 0, and -1, it could not find a way to fit a line to split those three labels.

In most cases in this experiment it seemed that perceptron was able to effectively classify the pictures given enough trials.

7. Log

Ryan : 6 hours
Sections 1, 3, 5
Leland : 8 hours
Sections 2, 4, 6