CSDS 391 Introduction to Artificial Intelligence

Total Points: 100 + 20 extra credit

In this assignment, you will train a linear classifier on the iris dataset. Like W4, this will be convenient to do in a language with a good plotting library such as Matlab or python using matplotlib. Submit both your code and your write-up (which should be a pdf file) via Canvas.

Write-up: Your write-up should answer the questions like in the written assignments. When relevant, include the source code in the write-up if it is sufficiently short (say half a page or less), otherwise refer to the file. For example, Exercise 2a asks you to write code to compute the mean-squared error, so you would just need to show the lines for that function. For reference, see §19.6 of the textbook (or §18.6 in the 3rd edition).

Exercise 1. Linear decision boundaries

a) Inspect `irisdata.csv` which is provided with the assignment file on Canvas. Write a program (or use a library) that loads the iris data set and plots the 2nd and 3rd iris classes, similar to the plots shown in lecture on neural networks (i.e. using the petal width and length dimensions). 10 P.

b) Define a function that computes the output of simple one-layer neural network using a logistic non-linearity (linear classification with logistic non-linearity). 5 P.

c) Write a function that plots the decision boundary for the non-linearity above overlaid on the iris data. Choose a boundary (by setting weight parameters by hand) that roughly separates the two classes. Use an output of 0 to indicate the 2nd iris class, and 1 to indicate the 3rd. 5 P.

d) Use a surface plot from a 3D plotting libary (e.g. in matlab or using matplotlib in python) to plot the output of your neural network over the input space. This should be similar to learning curve shown in fig 19.17 (18.17 in 3rd ed.) of the textbook. 5 P.

e) Show the output of your simple classifier using examples from the 2nd and 3rd Iris classes. Choose examples that are unambigious as well as those that are near the decision boundary. 5 P.

Exercise 2. Neural networks

a) Write a program that calculates the mean-squared error of the iris data for the neural network defined above. The function should take three arguments: the data vectors, the parameters defining the neural network, and the pattern classes. 10 P.

b) Compute the mean squared error for two different settings of the weights (i.e. two dfiferent decision boundaries). Like above, select these by hand and choose settings that give large and small errors respectively. Plot both boundaries on the dataset as above. 5 P.

c) Give a mathematical derivation the gradient of the objective function above with respect to the neural network weights. You will have to use chain rule and the derivative of the logistic function as discussed in class. Use w_0 to represent the bias term. You should show and explain each step. 10 P.

d) Show how the gradient can be written in both scalar and vector form. 5 P.

e) Write code that computes the summed gradient for an ensemble of patterns. Illustrate the gradient by showing (i.e. plotting) how the decision boundary changes for a small step. 10 P.

Exercise 3. Learning a decision boundary through optimization

a) Using your code above, write a program that implements gradient descent to optimize the decision boundary for the iris dataset. 5 P.

b) In your program, include code that shows the progress in two plots: the first should show the current decision boundary location overlaid on the data; the second should show the *learning curve*, i.e. a plot of the objective function as a function of the iteration. 5 P.

c) Run your code on the iris data set starting from a random setting of the weights. Note: you might need to restrict the degree of randomness so that the initial decision boundary is visible somewhere in the plot. In your writeup, show the two output plots at the initial, middle, and final locations of the decision boundary. 10 P.

d) Explain how you chose the gradient step size. 5 P.

e) Explain how you chose a stopping criterion. 5 P.

Exercise 4. Extra credit: Using a machine learning toolbox

Use a machine learning toolbox such as keras or scikit-learn to generalize the network to perform classification on the full iris dataset. Your network should have one or more hidden layers and classify all three iris classes using all four data dimensions. Show your results and explain your architecture. +20 P.

In addition to implementing the neural network classifier, also think of an issue to explore. Write your own mini-tutorial to report your results. Here are some examples of topics you can consider:

- illustrate how a multi-layer network makes a decision
- explore different non-linearities
- explore different network architectures and their classication and generalization performance
- contrast two different machine learning algorithms (e.g. a neural network vs k-means clustering) and how they perform on the full iris data set.
- explore algorithms for adjusting the learning rate

There are many tutorials using different toolboxes available on the internet. You can follow any you find useful to get started but do not just copy them for this extra credit portion of the assignment. You can make use of material you find useful, but cite your sources in your writeup. The TAs will check for straight copying, so please don't do this.

If you do use material from a source, simplify it to focus on a single topic and understand it well enough that you can explain it and modify it for our own purposes. Your write up should explain the ideas, code, and results in your own words and make use of tables or figures.

The extra credit is worth 20% of the points, so it should be about 20% of the effort for the main part of the assignment. Trivial examples will receive little or no credit. Your goal should be to teach yourself about something about a neural network-related topic, and the write up should written as if you are explaining it to one of your classmates.