Exercise I

AMTH/CPSC 663b - Spring semester 2018

Published: Thursday, February 1, 2018
Due: Thursday, February 15, 2018 - 4:00 PM

Please put all relevant files & solutions into a single folder titled `<lastname and initials>_assignment1` and then zip that folder into a single zip file titled `<lastname and initials>_assignment1.zip`, e.g. for a student named Tom Marvolo Riddle, `riddletm_assignment1.zip`. Include a single PDF titled `assignment1.pdf` and any Python scripts specified. Any requested plots should be sufficiently labeled for full points. Your homework should be submitted to Canvas before Thursday, February 15, 2017 at 4:00 PM.

Programming assignments should use built-in functions in Python and TensorFlow; In general, you may use the `scipy` stack [1]; however, exercises are designed to emphasize the nuances of machine learning and deep learning algorithms - if a function exists that trivially solves an entire problem, please consult with the TA before using it.

Problem 1

Provide an example application where each of the following architectures or techniques would be useful. Do not reuse examples from class.

1. Multilayer perceptron
2. Convolutional Neural Network
3. Recurrent Neural Network
4. Autoencoder
5. Ultra deep learning
6. Deep reinforcement learning

Include your answers in a PDF titled `assignment1.pdf`.

Problem 2

1. Show that the linear regression solution that minimizes MSE is

$$\mathbf{w}^* = (X^T X)^{-1} X^T \mathbf{y}.$$

Hint: You can find the equations for linear regression MSE in the lecture 2 slides.

2. Write code in Python that randomly generates N points sampled uniformly in the interval $x \in [-1, 3]$. Then output the function $y = x^2 - 3x + 1$ for each of the points generated. Then write code that adds zero-mean Gaussian noise with standard deviation σ to y. Make plots of x and y with $N \in \{15, 100\}$ and $\sigma \in \{0, .05, .2\}$ (there should be six plots in total). Save the point sets for following questions.
Hint: You may want to check the NumPy library for generating noise.

3. Find the optimal weights (in terms of MSE) for fitting a polynomial function to the data in all 6 cases generated above using a polynomial of degree 1, 2, and 9. Use the equation given above. Do not use built-in methods for regression. Plot the fitted curves on the same plot as the data points (you can plot all 3 polynomial curves on the same plot). Report the fitted weights and the MSE in tables. Do any of the models overfit or underfit the data?

4. Apply L2 norm regularization to the cases with $\sigma = 0.05$ and $N \in \{15, 100\}$. Vary the parameter λ, and choose three values of λ that result in the following scenarios: underfitting, overfitting, and an appropriate fit. Report the fitted weights and the MSE in each of these scenarios.
Hint: Check slides of lecture 2 for details on L2 norm regularization.

Include your answers and plots in a PDF titled `assignment1.pdf`. Include you code in a file titled `prob2`

Problem 3

1. Load the dataset from file assignment1.zip

2. Write a program that applies a k-nn classifier to the data with $k \in \{1, 5, 10, 15\}$. Calculate the test error using both leave-one-out validation and 5-fold cross validation. Plot the test error as a function of k. You may use the existing methods in scikit-learn or other libraries for finding the k-nearest neighbors, but do not use any built-in k-nn classifiers. Also, do not use any existing libraries or methods for cross validation. Do any values of k result in underfitting or overfitting?

3. Apply two other classifiers of your choice to the same data. Possible algorithms include (but are not limited to) logistic regression, QDA, naive Bayes, SVM, and decision trees. You may use any existing libraries. Use 5-fold cross validation to calculate the test error. Report the training and test errors. If any tuning parameters need to be selected, use cross-validation and report the training and test error for several values of the tuning parameters. Which of the classifiers performed best? Did any of them underfit or overfit the data? How do they compare to the k-nn classifiers in terms of performance?
Hint: You may want to check out the `scikit-learn` library.

Include your answers and plots in a PDF titled `assignment1.pdf`. Include your code for parts 2 and 3 in a file titled `prob3`.

Problem 4

1. Suppose we take all the weights and biases in a network of perceptrons, and multiply them by a positive constant, c > 0. Show that the behavior of the network doesn't change. (Exercise in Ch1 Nielsen book)

2. Given the same setup of `problem 4.1` - a network of perceptrons - suppose that the overall input to the network of perceptrons has been chosen and fixed. Suppose the weights and biases are such that $wx+b \neq 0$ for the input x to any particular perceptron in the network. Now replace all the perceptrons in the network by sigmoid neurons, and multiply the weights and biases by a positive constant $c > 0$. Show that in the limit as $c \to \infty$ the behavior of this network of sigmoid neurons is exactly the same as the network of perceptrons. How can this fail when $wx + b = 0$ for one of the perceptrons? (Exercise in Ch1 Nielsen book)

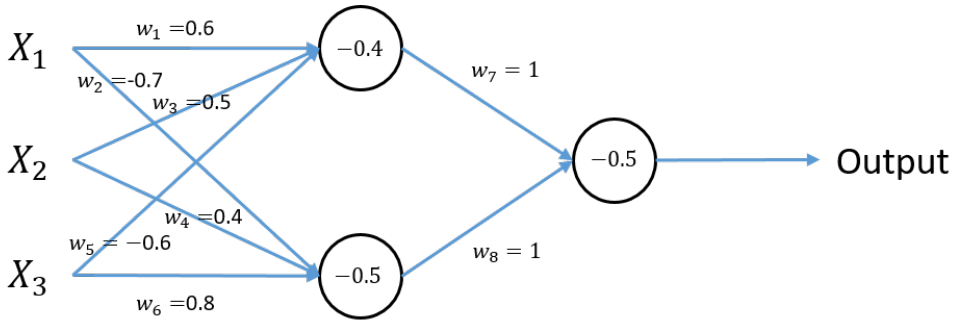

Figure 1: Multilayer perceptron with three inputs and one hidden layer.

3. For each possible input of the MLP in Figure 1, calculate the output. I.e., what is the output if $X = [0, 0, 0]$, $X = [0, 0, 1]$, etc. You should have 8 cases total.

4. If we change the perceptrons in Figure 1 to sigmoid neurons what are the outputs for the same inputs (e.g., inputs of [0,0,0], [0,0,1], ...)?

5. Using perceptrons with appropriate weights and biases, design an adder that does two-bit binary addition. Don't forget to include the carry bit.

Include your answers and a picture of your adder in a PDF titled `assignment1.pdf`.

Optional Problem

This problem will not be graded and thus should not be turned in. However, it will give you practice in training a neural network.

Run the Python code given in Chapter 1 of the Nielsen book in the section titled "Implementing our network to classify digits". You can find a link to the code at the beginning of the section. Verify that you understand each line of the code and that you obtain the same results as given in the book.

References

[1] "The scipy stack specification." [Online]. Available: https://www.scipy.org/stackspec.html