CMPE 257 Machine Learning Spring 2019

HW#4 Due May 6th, 11:59 PM, on Canvas

This homework is worth 200 points and is split into two parts. Part 1 consists of problems from the textbook. Part 2 consists of hands-on experience working with the digits dataset and will be released on 4/19. Part 1 is released early to give you a head start.

Part 1

1. (25 points) Problem 6.1 in textbook

Problem 6.1 Consider the following data set with 7 data points.

$$\left(\begin{bmatrix}1\\0\end{bmatrix},-1\right)\left(\begin{bmatrix}0\\1\end{bmatrix},-1\right)\left(\begin{bmatrix}0\\-1\end{bmatrix},-1\right)\left(\begin{bmatrix}-1\\0\end{bmatrix},-1\right)$$
$$\left(\begin{bmatrix}0\\2\end{bmatrix},+1\right)\left(\begin{bmatrix}0\\-2\end{bmatrix},+1\right)\left(\begin{bmatrix}-2\\0\end{bmatrix},+1\right)$$

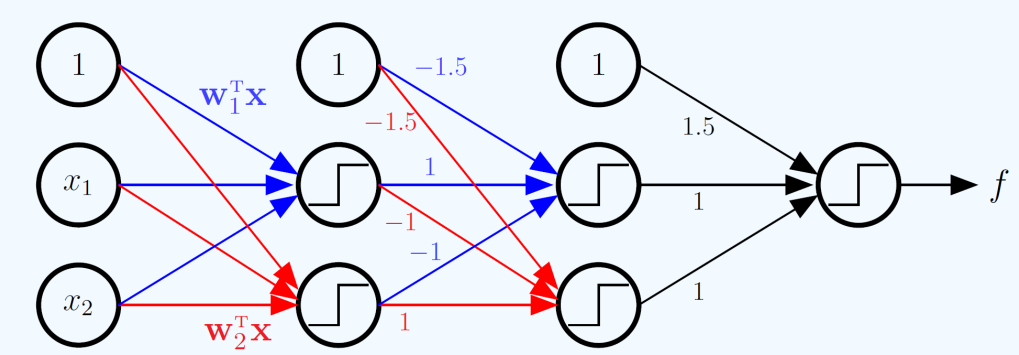
(a) Show the decision regions for the 1-NN and 3-NN rules.

(b) Consider the non-linear transform

$$\begin{bmatrix}x_1\\x_2\end{bmatrix}\mapsto\begin{bmatrix}z_1\\z_2\end{bmatrix}=\begin{bmatrix}\sqrt{x_1^2+x_2^2}\\\arctan(x_2/x_1)\end{bmatrix},$$

which maps $\mathbf{x}$ to $\mathbf{z}$. Show the classification regions in the $\mathbf{x}$-space for the 1-NN and 3-NN rules implemented on the data in the $\mathbf{z}$-space.

2. (25 points) Exercise 7.3 in textbook



Exercise 7.3

Use the graph representation to get an explicit formula for f and show that:

$$f(\mathbf{x})=\text{sign}\left[\text{sign}(h_1(\mathbf{x})-h_2(\mathbf{x})-\tfrac{3}{2})-\text{sign}(h_1(\mathbf{x})-h_2(\mathbf{x})+\tfrac{3}{2})+\tfrac{3}{2}\right],$$

where $h_1(\mathbf{x})=\text{sign}(\mathbf{w}_1^T\mathbf{x})$ and $h_2(\mathbf{x})=\text{sign}(\mathbf{w}_2^T\mathbf{x})$

3. (25 points) Exercise 7.7 in textbook

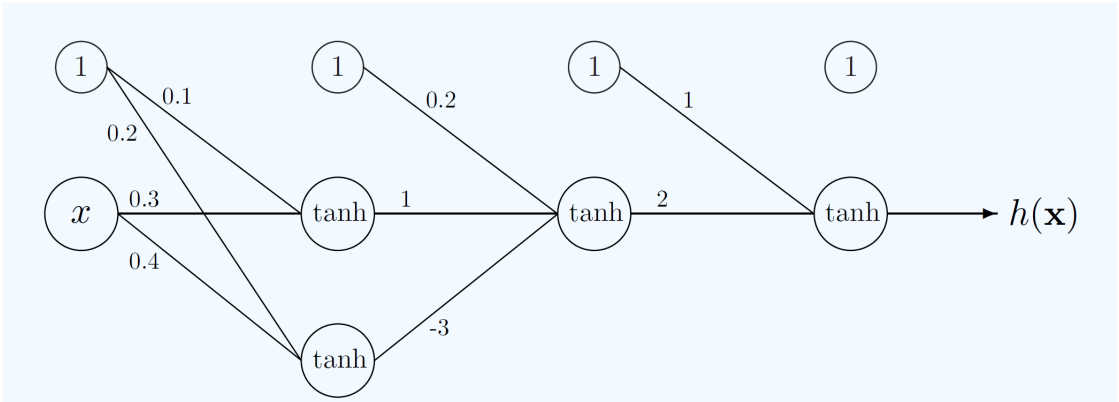
Exercise 7.7

For the sigmoidal perceptron, $h(\mathbf{x}) = \tanh(\mathbf{w}^{\mathrm{T}}\mathbf{x})$, let the in-sample error be $E_{\mathrm{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\tanh(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) - y_n)^2$. Show that

$$\nabla E_{\mathrm{in}}(\mathbf{w}) = \frac{2}{N}\sum_{n=1}^{N}(\tanh(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) - y_n)(1 - \tanh^2(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n))\mathbf{x}_n.$$

If $\mathbf{w} \to \infty$, what happens to the gradient; how this is related to why it is hard to optimize the perceptron.

4. (25 points) Exercise 7.8 in textbook



Exercise 7.8

Repeat the computations in Example 7.1 for the case when the output transformation is the identity. You should compute $\mathbf{s}^{(\ell)}, \mathbf{x}^{(\ell)}, \boldsymbol{\delta}^{(\ell)}$ and $\partial \mathrm{e}/\partial \mathrm{W}^{(\ell)}$

P.S. The above picture only shows the picture of the neural network in example 7.1. You can access these e-chapters of the textbook for free on amlbook.com. You are strongly suggested to go through example 1 to cement your understanding of how neural networks work before attempting this problem.

Part 2

1. (45 points) Write your own neural network implementation:

1.1 Write a function to perform forward propagation and back propagation for gradient descent on a 2 input ($d^{(0)} = 2$), m-hidden unit ($d^{(1)} = m$), 1 output neural network with 2 layers. Use tanh as the activation function on the neurons. For gradient descent, use squared error as the error measure.

If you would like to make the code generic, think about how you need to specify the neural network with the layers and the number of neurons per layer. E.g., the functions for forward and back propagation may take an array of $W^{(l)}$ matrices and the input vector $x^{(0)}$ as the parameters.

Note: There is a lot of code available online and implementations that you can easily copy from. I highly discourage you to do so for two main reasons: (i) We will use software similarity checkers and you risk getting caught for plagiarism. Please use your creativity to solve the homework and not to get around the system.; (ii) The course is meant for you to learn about machine learning and get a peek behind the curtain on what exactly happens in the libraries you might end up using in practice. Although you may never need to write anything from scratch, these exercises are meant to solidify your fundamental understanding of the topic.

2. (45 points) Use your neural network implementation from above to classify digits data (provided in Digits.zip on Canvas) separating digits labeled as '1' from 'not 1'. You will need to preprocess the data to get these labels. Take some time to familiarize yourself with the dataset.

Randomly initialize the weights and use the sign function to report the output in the output layer for classification. Plot $E_{in}(w)$ verses iteration number, stopping after $2*10^6$ iterations.

Submit your ipynb notebooks.

3. (10 points) Read Section 7.5 (Beefing up Gradient Descent) from the textbook and comment on what needs to change in your code to perform (i) variable rate gradient descent; (ii) steepest descent; (iii) conjugate gradients. You can optionally try out these methods on the dataset.

Submission instructions:

- Please read the submission instructions on Canvas for naming conventions. Please use meaningful names for variables and file names.
- Discussions are encouraged, but do not copy your answers from external sources or each other.
- Please cite all the sources you used in your submission.
- For questions on the textbook problems, you can check out the book forum: http://book.caltech.edu/bookforum/

Fun optional exercises using the digits data – not graded

Run SVM, nearest neighbor, RBFs, and clustering on the same dataset by implementing the algorithms from scratch (if you like) or using existing libraries (and learn what the libraries do).

Try regularization techniques with these algorithms.