Homework #2

RELEASE DATE: 04/24/2020

DUE DATE: 05/22/2020, BEFORE 13:00

QUESTIONS ABOUT HOMEWORK MATERIALS ARE
WELCOMED ON THE NTU COOL FORUM.

Please upload your solutions (without the source code) to Gradescope as instructed. For problems marked with (*), please follow the guidelines on the course website and upload your source code to NTU COOL. You are encouraged to (but not required to) include a README to help the TAs check your source code. Any programming language/platform is allowed.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English or Chinese with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

This homework set comes with 160 points and 40 bonus points. In general, every homework set would come with a full credit of 160 points, with some possible bonus points.

Descent Methods for Probabilistic SVM

Recall that the probabilistic SVM is based on solving the following optimization problem:

$$\min_{A,B} \quad F(A,B) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1+\exp\left(-y_n\left(A\cdot\left(\mathbf{w}_{\text{SVM}}^T\boldsymbol{\phi}(\mathbf{x}_n)+b_{\text{SVM}}\right)+B\right)\right)\right).$$

1. When using the gradient descent for minimizing $F(A,B)$, we need to compute the gradient first. Let $z_n = \mathbf{w}_{\text{SVM}}^T\boldsymbol{\phi}(\mathbf{x}_n)+b_{\text{SVM}}$, and $p_n = \theta(-y_n(Az_n+B))$, where $\theta(s) = \frac{\exp(s)}{1+\exp(s)}$ is the usual logistic function. What is the gradient $\nabla F(A,B)$ in terms of only y_n, p_n, z_n and N? Prove your answer.

2. When using the Newton method for minimizing $F(A,B)$ (see Homework 3 of Machine Learning Foundations), we need to compute $-(H(F))^{-1}\nabla F$ in each iteration, where $H(F)$ is the Hessian matrix of F at (A,B). Following the notations of the previous question, what is $H(F)$ in terms of only y_n, p_n, z_n and N? Prove your answer.

3. Following the previous queston, prove that the matrix $H(F)$ is positive semi-definite. The result shows that the optimization problem is convex.

Neural Network

4. Consider Neural Network with $\text{sign}(s)$ instead of $\tanh(s)$ as the transformation functions. That is, consider Multi-Layer Perceptrons. In addition, we will take $+1$ to mean logic TRUE, and -1 to mean logic FALSE. Assume that all x_i below are either $+1$ or -1. Write down the weights w_i for the following perceptron

$$g_A(\mathbf{x}) = \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right).$$

to implement

$$\texttt{OR}\left(x_1, x_2, \ldots, x_d\right).$$

Explain your answer.

5. For a Neural Network with at least one hidden layer and $\tanh(s)$ as the transformation functions on all neurons (including the output neuron), when all the initial weights $w_{ij}^{(\ell)}$ are set to 0, what gradient components are also 0? Justify your answer.

6. Consider a Neural Network with $d^{(0)} + 1 = 12$ input units (the constant $x_0^{(0)}$ is counted here as a unit), one output unit, and 48 hidden units (each $x_0^{(\ell)}$ is also counted as a unit). The hidden units can be arranged in any number of layers $\ell = 1, \cdots, L-1$. That is,

$$\sum_{\ell=1,\cdots,L-1} \left(d^{(\ell)} + 1\right) = 48.$$

Each layer is fully connected to the layer above it. What is the maximum possible number of weights that such a network can have? Explain your answer.

Autoencoder

7. Assume an autoencoder with $\tilde{d} = 1$. That is, the $d \times \tilde{d}$ weight matrix W becomes a $d \times 1$ weight vector $\mathbf{w}$, and the linear autoencoder tries to minimize

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{w}\mathbf{w}^T\mathbf{x}_n\|^2.$$

We can solve this problem with stochastic gradient decent by defining

$$\text{err}_n(\mathbf{w}) = \|\mathbf{x}_n - \mathbf{w}\mathbf{w}^T\mathbf{x}_n\|^2$$

and calculate $\nabla_{\mathbf{w}}\text{err}_n(\mathbf{w})$. What is $\nabla_{\mathbf{w}}\text{err}_n(\mathbf{w})$? List your derivation steps.

8. Following Question 7, assume that noise vectors $\boldsymbol{\epsilon}_n$ are generated i.i.d. from a zero-mean, unit variance Gaussian distribution and added to $\mathbf{x}_n$ to make $\tilde{\mathbf{x}}_n = \mathbf{x}_n + \boldsymbol{\epsilon}_n$, a noisy version of $\mathbf{x}_n$. Then, the linear denoising autoencoder tries to minimize

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{w}\mathbf{w}^T(\mathbf{x}_n + \boldsymbol{\epsilon}_n)\|^2.$$

For any fixed $\mathbf{w}$, the expected $E_{\text{in}}(\mathbf{w})$ is $\frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{w}\mathbf{w}^T\mathbf{x}_n\|^2 + \Omega(\mathbf{w})$. What is $\Omega(\mathbf{w})$? List your derivation steps.

9. On page 11 of Lecture 213, we mentioned that it is sometimes useful to tie the encoding weights and the decoding weights of the autoencoder to be the same. More formally, consider an autoencoder without any $x_0^{(\ell)}$. That is, the encoding weights are just $w_{ij}^{(1)}$ and the decoding weights are $w_{ji}^{(2)}$ for $i \in \{1, 2, \ldots, d\}$ and $j \in \{1, 2, \ldots, \tilde{d}\}$. Assume that $u_{ij} = w_{ij}^{(1)} = w_{ji}^{(2)}$. Write down the error function E of a basic autoencoder (page 11 of Lecture 213) as a function of u_{ij}.

10. Following Question 9, consider the same error function E as a function of $\mathbf{w}$ instead of $\mathbf{u}$ as if we do not tie the weights $\mathbf{w}$ by $\mathbf{u}$. That is, we do not have the constraints $w_{ij}^{(1)} = w_{ji}^{(2)}$ and it is possible that the two are not equal. Let's call the error function E_{10} to distinguish it from the previous error function, now called E_9. Prove that

$$\frac{\partial E_9(\mathbf{u})}{\partial u_{ij}} = \frac{\partial E_{10}(\mathbf{w})}{\partial w_{ij}^{(1)}} + \frac{\partial E_{10}(\mathbf{w})}{\partial w_{ji}^{(2)}}.$$

Experiments with Autoencoder

Implement the basic autoencoder on page 11 of Lecture 213 and train it with gradient descent (i.e. backpropogation with all data) using a learning rate of 0.1 and $T = 5000$. In addition, please initialize each weight $w_{ij}^{(\ell)}$ (including the bias) at the ℓ-th layer with a uniform random number within $[-U, U]$, where $U = \sqrt{\frac{6}{1+d^{(\ell-1)}+d^{(\ell)}}}$. Please use the normalized error function

$$\text{err}_{\mathbf{x}}(g) = \frac{1}{d}\sum_{i=1}^{d}(g_i(\mathbf{x}) - x_i)^2$$

per example $\mathbf{x}$ instead of

$$\sum_{i=1}^{d}(g_i(\mathbf{x}) - x_i)^2.$$

Please also remember to include $x_0^{(0)}$ and $x_0^{(1)}$ as shown on page 9 of Lecture 213.

11. (*) Run the algorithm **without the constraint** $w_{ij}^{(1)} = w_{ji}^{(2)}$ on the last 256 columns of the following file for training:

`http://amlbook.com/data/zip/zip.train`

Plot $E_{\text{in}}(g)$, which is the average $\text{err}_{\mathbf{x}}$ of the training examples, as a function of $\log_2 \tilde{d}$ for $\log_2 \tilde{d} \in \{1, 2, 3, 4, 5, 6, 7\}$. Describe your findings.

12. (*) Following Question 11, use the last 256 columns of the following file for testing:

http://amlbook.com/data/zip/zip.test

Plot $E_{\text{out}}(g)$, which is the aveage $\text{err}_{\mathbf{x}}$ of the test examples, as a function of $\log_2 \tilde{d}$ for $\log_2 \tilde{d} \in \{1, 2, 3, 4, 5, 6, 7\}$. Describe your findings.

13. (*) Following Question 11, but run the algorithm **with the constraint** $w_{ij}^{(1)} = w_{ji}^{(2)}$ and plot $E_{\text{in}}(g)$ as a function of $\log_2 \tilde{d}$ for $\log_2 \tilde{d} \in \{1, 2, 3, 4, 5, 6, 7\}$ and compare the plot with your result in Question 11. Describe your findings.

14. (*) Following Question 13, and plot $E_{\text{out}}(g)$ (on the test data defined in Question 12) as a function of $\log_2 \tilde{d}$ for $\log_2 \tilde{d} \in \{1, 2, 3, 4, 5, 6, 7\}$ and compare the plot with your result in Question 12. Describe your findings.

15. (*) Following Question 11, but the PCA algorithm on page 22 of Lecture 213 instead (please remember to do the mean shifting in step 1 of the PCA algorithm). Define

$$g(\mathbf{x}) = \text{WW}^T(\mathbf{x} - \bar{\mathbf{x}}) + \bar{\mathbf{x}}$$

as the 'linear autoencoder' that PCA learns. Plot $E_{\text{in}}(g)$ as a function of $\log_2 \tilde{d}$ for $\log_2 \tilde{d} \in \{1, 2, 3, 4, 5, 6, 7\}$ and compare the plot with your result in Question 13. Describe your findings.

16. (*) Following Question 15, and plot $E_{\text{out}}(g)$ (on the test data defined in Question 12) as a function of $\log_2 \tilde{d}$ for $\log_2 \tilde{d} \in \{1, 2, 3, 4, 5, 6, 7\}$ and compare the plot with your result in Question 14. Describe your findings.

Bonus: VC Dimension of Neural Networks

17. (Bonus 20%) Prove that for $\Delta \geq 2$, if $N \geq 3\Delta \log_2 \Delta$, $N^{\Delta} + 1 < 2^N$.

18. (Bonus 20%) Consider a hypothesis set $\mathcal{H}_{3A}$ that consists of all d-3-1 neural networks with $\text{sign}(\cdot)$ as all the transformation functions, and with $(w_0, w_1, w_2, w_3) = (-2.5, +1, +1, +1)$ for the output neuron ***only***. Use the facts above (or not) to prove that the VC dimension of $\mathcal{H}_{3A}$ is less than

$$3 \cdot \big(3(d+1)+1\big) \log_2\big(3(d+1)+1\big).$$