CSE546 Machine Learning, Autumn 2016: Homework 4

Due: Monday, December 12th, 5pm

Policies

Coding: You must write your own code. You may use any numerical linear algebra package, but you may *not* use machine learning libraries (e.g. sklearn, tensorflow) unless otherwise specified. In addition, each student must write and submit their own code in the programming part of the assignment (we may run your code).

Acknowledgments: We expect you to make an honest effort to solve the problems individually. As we sometimes reuse problem set questions from previous years, covered by papers and webpages, we expect the students not to copy, refer to, or look at the solutions in preparing their answers (referring to unauthorized material is considered a violation of the honor code). Similarly, we expect to not to google directly for answers (though you are free to google for knowledge about the topic). If you do happen to use other material, it must be acknowledged here, with a citation on the submitted solution.

Readings

Read the required material.

Required HW submission format:

The following is the required HW submission format:

- Submit all the answers to the HW as one single *typed* pdf document (not handwritten). This document must contain all plots. It is encouraged you latex all your work, though you may use another comparable typesetting method.
- Additionally, submit your code as a separate archive file (a .tar or .zip file). All code *must* be submitted. The code should be in a runnable file format like .py files or .txt files. Jupyter notebooks are not acceptable.
- List the names of all people you collaborated with and for which question(s). Please mark this as Question 0. Each student must understand, write, and hand in their own answers. Also, each student must understand, write, and hand in their own code.

0 Collaboration and Acknowledgements

List the names of all people you collaborated with and for which question(s). Each student must understand, write, and hand in their own answers. Also, each student must understand, write, and hand in their own code.

1 Manual calculation of one round of EM for a GMM [15 points]

(Extended version of: Murphy Exercise 11.7) In this question we consider clustering 1D data with a mixture of 2 Gaussians using the EM algorithm. You are given the 1-D data points $x = [1 \;\; 10 \;\; 20]$.

1.1 M step [10 points]

Suppose the output of the E step is the following matrix:

$$R = \begin{pmatrix} 1 & 0 \\ 0.4 & 0.6 \\ 0 & 1 \end{pmatrix}$$

where entry $R_{i,c}$ is the probability of observation x_i belonging to cluster c (the responsibility of cluster c for data point i). You just have to compute the M step. You may state the equations for maximum likelihood estimates of these quantities (which you should know) without proof; you just have to apply the equations to this data set. You may leave your answer in fractional form. Show your work.

1. *[2 points]* Write down the likelihood function you are trying to optimize.
2. *[2 points]* After performing the M step for the mixing weights π_1, π_2, what are the new values?
3. *[3 points]* After performing the M step for the means μ_1 and μ_2, what are the new values?
4. *[3 points]* After performing the M step for the standard deviations σ_1 and σ_2, what are the new values?

1.2 E step [5 points]

Suppose the output of the M step is your previous answer. Let us compute the subsequent E step.

1. *[2 points]* Write down the formula for the probability of observation x_i belonging to cluster c.
2. *[3 points]* After performing the E step, what is the new value of R?

2 Neural Nets and Backprop [45 points]

Now we will try our hands on deep learning on the MNIST dataset. Let us use the square loss for the loss function at the top layer.

We will be using 2 layer neural networks throughout, with either tanh hidden units or ReLu hidden units. Note that should you compare your test errors to those of the neural network results in `http://yann.lecun.com/exdb/mnist/` (for comparable network architectures), you should be able to significantly improve upon them.

As before, project each image onto the top 50 PCA directions. This reduces the dimension of each image from 784 to 50. This will speed up the computation.

2.1 With tanh hidden units [15 points]

The input layer should have 50 dimensions (as the image is 50 dimensions, after PCA). Let us use 500 nodes for the hidden layer, with a tanh transfer function. The output layer can simply consist of the predictions of the network. Let us make the output nodes to be linear in the hidden layer.

1. *(2 points)* Specify all your parameter choices: your learning rate (or learning rate scheme), mini-batch size, initialization scheme.
2. *(6 points)* Plot the squared loss after every half epoch (starting with your initial squared error). Please label your axes in a readable way. Plot the loss of both the training and test losses on the same plot. For the 0/1 loss, do the same, except start your plots a little later (e.g. when the 0/1 loss is below 7%) so they are more readable.
3. *(4 points)* What is your final squared loss and 0/1 loss for both the training and test sets?
4. *(3 points)* Choose 10 hidden layer nodes at random and display the learned weights (these are the 50 dimensional weights, visualized back into image space).

2.2 With ReLu hidden units [15 points]

The input layer should have 50 dimensions (as the image is 50 dimensions, after PCA). Let us use 500 nodes for the hidden layer, with a rectified linear (ReLu) transfer function. The output layer can simply consist of the predictions of the network. Let us make the output nodes to be linear in the hidden layer.

1. *(2 points)* Specify all your parameter choices: your learning rate (or learning rate scheme), mini-batch size, initialization scheme.
2. *(6 points)* Plot the squared loss after every half epoch (starting with your initial squared error). Please label your axes in a readable way. Plot the loss of both the training and test losses on the same plot. For the 0/1 loss, do the same, except start your plots a little later (e.g. when the 0/1 loss is below 7%) so they are more readable.
3. *(4 points)* What is your final squared loss and 0/1 loss for both the training and test sets?
4. *(3 points)* Choose 10 hidden layer nodes at random and display the learned weights (these are the 50 dimensional weights, visualized back into image space).

2.3 With ReLu hidden units + ReLu output units [15 points]

The input layer should have 50 dimensions, and let us use 500 nodes for the hidden layer, with a rectified linear (ReLu) transfer function. Now the output layer — the ten predictions made by the network — should be a ReLu unit where each output is a ReLu taking as input a linear combination of the hidden layer outputs.

1. *(2 points)* Specify all your parameter choices: your learning rate (or learning rate scheme), mini-batch size, initialization scheme.
2. *(6 points)* Plot the squared loss after every half epoch (starting with your initial squared error). Please label your axes in a readable way. Plot the loss of both the training and test losses on the same plot. For the 0/1 loss, do the same, except start your plots a little later (e.g. when the 0/1 loss is below 7%) so they are more readable.
3. *(4 points)* What is your final squared loss and 0/1 loss for both the training and test sets?
4. *(3 points)* Choose 10 hidden layer nodes at random and display the learned weights (these are the 50 dimensional weights, visualized back into image space).

3 EM v.s. Gradient Descent [15 points]

Let us gain some intuition as to how EM differs from gradient descent. The setting is where we have a family of distributions $\Pr(\mathbb{X}, \mathbb{Z}|\theta)$ where θ is a vector in $\mathbb{R}^d$. We think of $\mathbb{X}$ as the observable variables (our data), and $\mathbb{Z}$ as the latent or hidden variables.

Given the observed data $\mathbb{X}$, the MLE is the solution to the following optimization problem:

$$\theta_{\text{MLE}} = \arg\max_{\theta} L(\theta) \text{ where } L(\theta) = \log \Pr(\mathbb{X}|\theta)$$

The EM algorithm seeks to maximize the log likelihood function, $L(\theta)$. Now let us consider gradient descent.

1. *(8 points)* Show that:
$$\nabla L(\theta) = \mathbb{E}_{\mathbb{Z}\sim\Pr(\mathbb{Z}|\mathbb{X},\theta)} \nabla \log \Pr(\mathbb{X}, \mathbb{Z}|\theta)$$

2. *(3 points)* Suppose starting from θ, Alice does one gradient update. Now suppose Bob, also starting with θ, performs an E step, and then, instead of doing an exact M-step, Bob does a gradient update on the objective function in the M-step. Both Alice and Bob use the same learning rates. Are Alice and Bob doing the same thing? Give a brief justification of your answer.

3. *(4 points)* Suppose now that you run the EM algorithm to convergence (assume it converged). Do you reach a critical point of the likelihood function? (i.e. do you reach a point where the gradient of the log likelihood function is 0). Give a brief justification of your answer.

4 Markov Decision Processes and Dynamic Programming [15 points]

Consider an MDP with a finite set of states and a finite set of action. Denote the reward function by $R(x, a)$, which is the instantaneous reward at state x upon taking action a. Let $\Pr(x'|x, a)$ be the transition probability of moving from state x to x' upon taking action a. Let $0 \le \gamma < 1$ be the discount factor.

Let V denote a value function, which associates a value $V(x)$ for each state x. Let $\text{Bell}(V)$ be the Bellman update operator when applied to V. Specifically, it is defined as follow: $\widetilde{V} = \text{Bell}(V)$ where

$$\widetilde{V}(x) = \max_a \left(R(x, a) + \gamma \sum_{x'} \Pr(x'|x, a) V(x') \right)$$

Let us now prove that this update rule converges geometrically to the optimal values.

1. *(8 points)* Show that the Bellman operator is a contraction mapping. Specifically, show that for any two value functions
$$\|\text{Bell}(V_1) - \text{Bell}(V_2)\|_\infty \le \gamma \|V_1 - V_2\|_\infty$$
where the $\|Z\|_\infty$ denotes the infinity norm of a vector Z, i.e. $\|Z\|_\infty = \max_s |Z(s)|$.

2. *(3 points)* Suppose that upon repeated updating, we reach a fixed point, i.e. we find a V such that $\text{Bell}(V) = V$. Show that this V is unique. This V is the value of the optimal policy.

3. *(4 points)* Suppose we find a V such that $\text{Bell}(V) = V$. Specify the optimal policy $\Pi(x)$, which specifies the action to be taken in state x, in terms of V and other relevant quantities.