COMS 4771 HW1

Due: Fri Oct 06, 2017

You are allowed to work in groups of (at max) three students. Only one submission per group is required by the due date. Name and UNI of all group members must be clearly specified on the homework. You must cite all the references you used to do this homework. You must show your work to receive full credit.

1 **[Maximum Likelihood Estimation]** Here we shall examine some properties of Maximum Likelihood Estimation (MLE).

(i) Consider the density $p(x|\theta) := \begin{cases} \theta e^{-\theta x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$, for some $\theta > 0$. Suppose that n samples $x_1, \ldots, x_n$ are drawn i.i.d. from $p(x|\theta)$. What is the MLE of θ given the samples?

(ii) Consider the density $p(x|\theta) := \begin{cases} 1/\theta & \text{if } 0 \leq x \leq \theta \\ 0 & \text{otherwise} \end{cases}$, for some $\theta > 0$. Suppose that n samples $x_1, \ldots, x_n$ are drawn i.i.d. from $p(x|\theta)$. What is the MLE of θ given the samples?

(iii) Recall the Gaussian density: $p(x|\mu, \sigma^2) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$, for some mean parameter $\mu \in \mathbb{R}$ and variance parameter $\sigma^2 > 0$. Suppose that n samples $x_1, \ldots, x_n$ are drawn i.i.d. from $p(x|\mu, \sigma^2)$. Show that if μ is unknown, then the MLE σ^2_{ML} is **not** an unbiased estimator of the variance σ^2 for all sample sizes n. What simple modification can we make to the estimate to make it unbiased?

(iv) Show that for the MLE θ_{ML} of a parameter $\theta \in \mathbb{R}^d$ and any known differentiable function $g : \mathbb{R}^d \to \mathbb{R}^k$, the MLE of $g(\theta)$ is $g(\theta_{\text{ML}})$. From this result infer the MLE for the standard deviation (σ) in the same setting as in Part (iii).

2 **[Evaluating Classifiers]** Consider the following decision rule f_t for a two-category problem in $\mathbb{R}$. Given an input $x \in \mathbb{R}$

decide category y_1, if $x > t$; otherwise decide category y_2

(i) What is the error rate for this rule, that is, what is $P[f_t(x) \neq y]$?

(ii) Show that at for the optimally selected threshold value t (i.e., the one which gives minimum error rate), it must be the case that

$$P(X = t|Y = y_1)P(Y = y_1) = P(X = t|Y = y_2)P(Y = y_2).$$

(iii) Assume that the underlying population distribution has equal class priors (i.e., $P[Y = y_1] = P[Y = y_2]$), and the individual class conditionals (i.e., $P[X|Y = y_1]$ and

$P[X|Y = y_1]$) are distributed as Gaussians. Give an example setting of the class conditionals (i.e., give an example parameter settings for the Gaussians) such that for some threshold value t, the rule f_t achieves the Bayes error rate; and similarly, give an example setting of the class conditionals such that for no threshold value t, the rule f_t achieves the Bayes error rate.

3 **[Randomized decision rules]** Consider a "reward-based" prediction framework, where given a continuous state-space X and a discreet action-space A, a learning algorithm $f : X \to A$ decides to perform action $a \in A$ when it is in state $x \in X$. The learner then receives a numerical (possibly negative) reward $R(x, a)$ for performing action $a \in A$ in state $x \in X$. (Note: such reward-based frameworks are common in robotics where the learning agent, i.e., the robot, performs some—possibly randomized—action a in some situation x and receives reward $R(x, a)$. The goal of the learning agent is to maximize its expected reward.)

(i) Assume that f is allowed take a *randomized* action on any state x. That is, in any state $x \in X$, f chooses to perform action a with probability $P[a|x]$. Show that the expected reward received by f, i.e., $\mathbb{E}_{x,a}[f]$ is

$$\int \Big[\sum_a R(x,a)P[a|x]\Big] P[x]dx.$$

(ii) Show that the expected reward is maximized by choosing $P[a|x] = 1$ for the action a associated with the maximal reward $R(x, a)$ (for a given state x). This shows us that there is *no benefit* in randomizing the best decision rule.

(iii) Can one benefit from randomizing a suboptimal rule? Briefly explain.

4 **[Finding (local) minima of generic functions]** Finding extreme values of functions in a closed form is often not possible. Here we will develop a generic algorithm to find the extremal values of a function. Consider a smooth function $f : \mathbb{R} \to \mathbb{R}$.

(i) Recall that Taylor's Remainder Theorem states:
For any $a, b \in \mathbb{R}$, exists $z \in [a, b]$, such that $f(b) = f(a) + f'(a)(b-a) + \frac{1}{2}f''(z)(b-a)^2$.

Assuming that there exists $L \geq 0$ such that for all $a, b \in \mathbb{R}$, $|f'(a) - f'(b)| \leq L|a - b|$, prove the following statement:

For any $x \in \mathbb{R}$, there exists some $\eta > 0$, such that if $\bar{x} := x - \eta f'(x)$, then $f(\bar{x}) \leq f(x)$, with equality if and only if $f'(x) = 0$.

(Hint: first show that the assumption implies that f has bounded second derivative, i.e., $f''(z) \leq L$ (for all z); then apply the remainder theorem and analyze the difference $f(x) - f(\bar{x})$).

(ii) Part (i) gives us a generic recipe to find a new value $\bar{x}$ from an old value x such that $f(\bar{x}) \leq f(x)$. Using this result, develop an iterative algorithm to find a local minimum starting from an initial value x_0.

(iii) Use your algorithm to find the minimum of the function $f(x) := (x - 3)^2 + e^x$. You should code your algorithm in a scientific programming language like Matlab to find the solution.

5 **[A comparative study of classification performance of handwritten digits]** Download the datafile hw1data.mat from the class website. This datafile contains 10,000 images (each of size 28x28 pixels = 784 dimensions) of handwritten digits along with the associated labels. Each handwritten digit belongs to one of the 10 possible categories $\{0, 1, \ldots, 9\}$. There are two variables in this datafile: (i) Variable X is a 10,000x784 data matrix, where each row is a sample image of a handwritten digit. (ii) Variable Y is the 10,000x1 label vector where the i^{th} entry indicates the label of the i^{th} sample image in X.

Special note for those who are not using Matlab: Python users can use `scipy` to read in the mat file, R users can use `R.matlab` package to read in the mat file.

To visualize this data (in Matlab): say you want to see the actual handwritten character image of the 77th datasample. You may run the following code (after the data has been loaded):

```
figure;
imagesc(1-reshape(X(77,:),[28 28])');
colormap gray;
```

To see the associated label value:

```
Y(77)
```

(i) Create a probabilistic classifier (as discussed in class) to solve the handwritten digit classification problem. The class conditional densities of your probabilistic classifier should be modeled by a Multivariate Gaussian distribution. It may help to recall that the MLE for the parameters of a Multivariate Gaussian are:

$$\vec{\mu}_{\text{ML}} := \frac{1}{n}\sum_{i=1}^{n} \vec{x}_i$$

$$\Sigma_{\text{ML}} := \frac{1}{n}\sum_{i=1}^{n} (\vec{x}_i - \vec{\mu}_{\text{ML}})(\vec{x}_i - \vec{\mu}_{\text{ML}})^{\mathsf{T}}$$

You must submit your code via Courseworks to receive full credit.

(ii) Create a k-Nearest Neighbor classifier (with Euclidean distance as the metric) to solve the handwritten digit classification problem.

You must submit your code via Courseworks to receive full credit.

(iii) Which classifier (the one developed in Part (i) or the one developed in Part (ii)) is better? You must justify your answer with appropriate performance graphs demonstrating the superiority of one classifier over the other. Example things to consider: you should evaluate how the classifier behaves on a holdout 'test' sample for various splits of the data; how does the training sample size affects the classification performance.

(iv) As discussed in class, there are several metrics one can use in a Nearest Neighbor classification. Do a similar analysis to justify which of the three metrics: L_1, L_2 or L_∞ is better for handwritten digit classification problem.

6 **[Optimizing over large-scale data]** Here we will use the same data set as in the previous question. Recall that the 784 image pixels are the input features and the digit identifier ($K = 10$) is the supervision label. We shall use the notation $x^i \in \mathbb{R}^{784}$ to represent the i^{th} data point,

and $y^i \in \mathbb{R}^K$ as the corresponding label represented as a K-dimensional "one-hot encoding" vector (i.e., $y^i_k = 1$ if i^{th} data sample has label k, and 0 otherwise). Say, we want to fit a global parameter vector $\theta \in R^K$ to this dataset via an optimization function

$$f(\theta; (X,Y)) := \sum_{(x_i,y_i)} \sum_{k=1}^{k} \sum_{d=1}^{D} -\frac{y^i_k}{2}(x^i_d - \theta_k)^2$$

(i) Compute $\nabla_\theta f$.

(ii) Recall that using the iterative algorithm developed in Question 4(ii), we can find an optimal setting of θ that minimizes f. Here we will study how long does it take to find an optimal setting of θ. Plot the run of your iterative optimization algorithm, with the function value f on the y-axis and the wall clock time on the x-axis.

(iii) Observe that the optimization function f can be written as $f(\theta; (X,Y)) = \sum_i f_i(\theta; (x_i, y_i))$, where $f_i(\theta; (x_i, y_i)) := \sum_{k=1}^{K} \sum_{d=1}^{D} -\frac{y^i_k}{2}(x^i_d - \theta_k)^2$. Compute $\nabla_\theta f_i$. This is the gradient for each sample i.

(iv) An alternate fast way to approximate the gradient (in part (i)) is to simply compute the gradient for a single *random* sample (as done in part (iii)). In doing so, we are (in expectation) approximating the full gradient. Implement a faster approximate version of your iterative algorithm in Question 4(ii) that computes gradient of a random example in each iteration (rather than computing the full gradient). Note that in practice instead of sampling data randomly, one creates an initial random permutation the input data and iterate through the randomly permuted data, taking a gradient step per example each iteration. (Hint: scale the gradient by N, the size of the data to ensure the gradient is unbiased).

(v) For an arbitrary step in the run of your faster approximate iterative algorithm (in part (iv)), plot a histogram of the (scaled) gradient of all the individual data points along with the gradient of all data points at once. What do you notice? Can you explain?

(vi) Plot a run of your fast-approximate iterative optimizing algorithm, with the error on the y-axis and the wall clock time on the x-axis, and compare it with the plot generated in part (ii). (you can have both plots on the same figure to properly compare). What do you notice? Can you explain?

7 **[Understanding model complexity and overfitting]** Here we will empirically study the tradeoff between model complexity and generalizability.

(i) Build a decision tree classifier for the digit dataset that we already introduced in Question 5. In building your decision tree, you may use any reasonable uncertainty measure to determine the feature and threshold to split at in each cell. Make sure the depth of the tree is adjustable with hyperparameter K.

(ii) Ensure that there is a random split between training and test data. Plot the training error and test error as a function of K.

(iii) Do the trends change for different random splits of training and test data?

(iv) How do you explain the difference in the behavior of training and testing error as a function of K?

(v) Based on your analysis, what is a good setting of K if you were deploy your decision tree classifier to classify handwritten digits?