CS6923 Machine Learning, Spring 2018
Prof. Hellerstein, NYU School of Engineering

Homework 2

Submit on NYU Classes by Fri. March 9 at 6:00 p.m. You may work together with one other person on this homework. If you do that, hand in JUST ONE homework for the two of you, with both of your names on it. You may *discuss* this homework with other students but YOU MAY NOT SHARE WRITTEN ANSWERS OR CODE WITH ANYONE BUT YOUR PARTNER.

IMPORTANT SUBMISSION INSTRUCTIONS:

- Submit your solutions in 3 separate files: one file for your written answers to Part I, one file for your written answers/output for the questions in Part II, and one file with your Nearest Neighbor code (a zip file if your code requires more than one file). Do NOT submit everything in one zip file.
- If you are working with a partner, make sure that both of your names/ids are present in ALL your submitted files

Part I: Written Exercises

1. (The following is a "sensor fusion" problem.)

   You are trying to estimate the units of carbon in a chemical compound. Call the percentage of units of carbon θ.

   You look in the literature and find a paper that reports the percentage to be 25 with a standard deviation of 3. In saying this, the paper is actually giving a distribution on θ; specifically, the Gaussian distribution with mean 25 and standard deviation 3. This distribution reflects the "probability" that the authors are assigning to different possible values for θ (in spam filtering, this would be like saying an email is spam with probability 25%, and non-spam with probability 75%).[1]

   Your use your own measuring device, and it reports the percentage to be $y = 28$. Your own measuring device is also imprecise, so you view the reported number as the value of a random variable drawn from a Gaussian distribution with mean equal to the true percentage, θ, and standard deviation 5. You know the standard deviation, but you don't know θ.

[1] This may seem weird, because θ can only be between 0 and 100, and the value of the Gaussian pdf is non-zero even for values of θ outside of that range. But it's common to use the Gaussian in situations such as this one, because it's easier to work with a Gaussian distribution, which has a continuous pdf, rather than some other distribution that is 0 outside of the range $[0, 100]$.

You want to combine the information from the literature with the information you got from your own measuring device, to get an estimate for θ. You will use a Bayesian approach. View the Gaussian from the literature as a prior distribution on θ, the actual fraction of carbon units. By "prior" here, we mean prior to doing your own measurement.

View your estimate of 28 as being the result of drawing once from a Gaussian with mean θ and standard deviation 5.

(a) By Bayes rule, the posterior probability of θ is $p(\theta) = \frac{p(y|\theta)p(\theta)}{p(y)}$. Use Bayes rule, and the assumptions above, to compute θ_{MAP}. You will probably want to take logs. Show your work.

(b) What if you instead used your own measurement 28 ± 5 as the prior, and 25 as the data (with standard deviation 3). Would the value of θ_{MAP} be the same or different in this case?

(c) Generalize the answer your obtained for θ_{MAP}. Suppose the reported percentage in the literature was $\mu_1 \pm \sigma_1$, the standard deviation of your measurement was σ_2, and the measurement you took was y. Give a formula for θ_{MAP} in terms of μ_1, y, σ_1 and σ_2. Your formula should have the form $Ay + B\mu_1$ where A and B are functions of the two standard deviations.

(Hint: This is a special case of a formula presented in text and in lecture for θ_{MAP}, when we are given a sample $\mathcal{X}$ and assume it was derived from a random process that first chooses θ from $\mathbb{N}(\mu_1, \sigma_1)$, and then draws N times from $\mathbb{N}(\theta, \sigma_2)$.}. In this problem, $N = 1$, so plugging $N = 1$ into the equation from class should give the same answer that you are giving here.)

2. When running Naive Bayes, it is common to use add-m smoothing to estimate the probability $P(x_i \mid C)$.

More abstractly, let X be a random variable, distributed according to a discrete probability distribution $\mathcal{D}$ over a finite set V. (X is sometimes called a *categorical* random variable.) Let $p_v = P(X = v)$.

Let $x^1, \ldots, x^N$ be a random sample drawn from the distribution $\mathcal{D}$. Let $v \in V$, and let N_v denote the number of x^t in this sample where $x^t = v$. We can apply the formula for add-m smoothing to this sample to produce an estimate $\hat{p}_v$ of p_v as follows:

$$\hat{p}_v := \frac{N_v + m}{N + |V|m}$$

Fix m to be equal to 0.1, and $|V|$ to be equal to 3. What is the bias of $\hat{p}_v$, as an estimator of p_v?

To answer this question, it helps to define a Bernoulli variable Y, associated with X, whose value is 1 if $X = v$, and 0 otherwise. Then, associated with the sample $x^1, \ldots, x^N$, you have Bernoulli random variables $y^1, \ldots, y^N$ and $N_v = y^1 + \ldots + y^N$.

3. Suppose you work for a credit card company, and you are designing a procedure that will be used to decide whether to approve a credit card purchase at a store.

   - If the purchase is approved, but it is being made with a stolen card, then the credit card company will have to reimburse the store for the amount of the purchase.
   - If the purchase is not approved, and the card is not a stolen card, then the customer and store may get angry, cancel the card, and write a bad review of the card on social media. The company thinks that this will ultimately cost them $\$5x$, where x is the amount of the purchase.
   - If the purchase is denied and was being made with a stolen card, then the credit card company doesn't lose or earn money.
   - If the purchase is approved, and the card is not stolen, then the store makes a processing fee worth 2% of the purchase.

   If we are using a Bayesian approach, where we estimate posterior probabilities $P(C|x)$, we can try to minimize *risk* by choosing the action that will result in smaller expected cost.

   (You should consider the processing fee to be a negative cost to the company. That is, if e.g. the processing fee is \$1.25, then that is a cost of -\$1.25 to the company.)

   Suppose a person tries to make a \$350 purchase using their credit card. Based on available data (time and place of transaction, previous purchase history of the customer), the company's existing software estimates that $P(Stolen) = 0.26$.

   (a) What will be the cost to the company if the purchase is approved, and the card is a stolen card?

   (b) What will be the cost to the company if the purchase is approved, and the card is not a stolen card?

   (c) Using the assumption that $P(Stolen) = 0.26$, what is the expected cost to the company if the purchase is approved ?

   (d) Using the assumption that $P(Stolen) = 0.26$, what is the expected cost to the company if the purchase is denied?

   (e) Which decision minimizes the company's risk (expected cost), to approve or to deny the purchase?

4. Consider the following dataset for *multiple* linear regression, that is, the problem of fitting a linear function when there is more than one independent variable (input variable).

x_1	x_2	r
1	3	3
4	2	7
5	4	2
8	7	1

   (a) For multiple linear regression, the line $g(x) = w_k x_k + w_{k-1} x_{k-1} + \ldots + w_1 x_1 + w_0$ minimizing squared error can be computed using the following (where superscripts are NOT exponents – they are the indices t of the examples x^t):

$$D = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \ldots & x_k^1 \\ 1 & x_1^2 & x_2^2 & \ldots & x_k^2 \\ \ldots & \ldots & \ldots & \ldots & \\ 1 & x_1^N & x_2^N & \ldots & x_k^N \end{bmatrix}$$

$$r = \begin{bmatrix} r^1 \\ r^2 \\ \ldots \\ r^N \end{bmatrix}$$

$$w = (D^T D)^{-1} D^T r$$

   Give the values for D and $(D^T D)^{-1}$ for the above dataset. Then compute w and give the resulting linear equation for $g(x)$.

   (Note the similarity between the D matrix here and the D matrix in the solution given in class for fitting a polynomial in one variable. This is not a coincidence.)

   (b) Suppose we tried to repeat the previous question using only the first 2 examples in the dataset, instead of all of them. Compute $(D^T D)^{-1}$ in this case. What happens? Why?

   (c) Consider a regularized error function,

$$E_{2,r} := \frac{1}{2}\left(\sum_t \frac{1}{2}(g(x^t) - r^t)^2 + \lambda \sum_{i=1}^{2} w_i^2\right)$$

   Note that the penalty term $\lambda \sum_{i=1}^{2} w_i^2$ does NOT include the square of w_0.

   What is the error of the linear function $g(x) = w_2 x_2 + w_1 x_1 + w_0$ on the dataset above, with respect to the regularized error function that we are calling $E_{2,r}$? Express your answer as a function of the w_i's and λ.

(d) Set $\lambda = 2$. By taking partial derivatives, write a system of 3 equations in the 3 unknowns w_0, w_1, w_2 such that the solution to the system of equations are the coefficients of $g(x)$ minimizing the regularized error function $E_{2,r}$ on the above dataset.

(e) Give the equation of the linear function that minimizes the regularized error function $E_{2,r}$ error on this dataset. (You may want to use a linear equation solver such as numpy np.linalg.solve, or another tool.)

Part II: Hands-On Exercises

In this part of the homework we will perform experiments using a version of the automobile dataset from the UC Irvine Repository. The problem will be to predict the miles per gallon (mpg) of a car, given its displacement and horsepower. Each example in the dataset corresponds to a single car.

The training data is in auto_train.csv, and the testing data is in auto_test.csv. The first row of each of these files has the names of the input attributes and the output attribute. The file auto_info.txt has the dataset information.

We will be using linear and polynomial regression on this problem. These are parametric approaches. We will also use non-parametric approaches, namely Nearest Neighbor, and k Nearest Neighbor. The intuition behind Nearest Neighbor is that similar examples x should have similar outputs r.

The basic Nearest Neighbor algorithm is simple and can be used for classification or regression. Given a training set, just store it. Then, when you need to predict the output for a new example x, find the "most similar" example x^t in the training set (this is the "Nearest Neighbor" of x). Predict that the output on x is r^t.

The basic Nearest Neighbor algorithm does not handle outliers well. It has high variance as an algorithm, meaning that its predictions can vary a lot depending on which examples happen to appear in the training set. The k Nearest Neighbor algorithm addresses these problems by using the k examples in the training set that are "most similar" to x. For regression, it predicts using the *average* of their output values. For classification, it predicts using the *most frequent* label of their labels. (So Nearest Neighbor is the same as k Nearest Neighbor, when $k = 1$.)

One of the most important decisions in implementing Nearest Neighbor algorithms is the definition of the "distance" between two examples. We will use a common distance function, Euclidean distance. So, e.g., the distance between example $(2, 5)$ and example $(7, 9)$ is $\sqrt{5^2 + 4^2} = \sqrt{41}$. The nearest neighbor of x is the example in the training set whose distance from x is as *small* as possible.

For the problems below, if it says "package allowed", you do not have to implement the algorithm from scratch. You can use a package or tool and you do not need to show your code. If you use a package, make sure that any options are set correctly and that the package is implementing the specified algorithm.

If you are looking for a python package for regression, one option is to use scikit-learn:

- http://scikit-learn.org/stable/modules/linear_model.html#polynomial-regression-extending-linear-models-with-basis-functions
- http://scikit-learn.org/stable/modules/linear_model.html#polynomial-regression-extending-linear-models-with-basis-functions

1. For now, ignore the horsepower attribute. Plot the data in the training set (with displacement on the horizontal axis, and mpg on the vertical axis).

   If you don't already have a favorite way to plot the data, the following is a https://matplotlib.org/gallery/pyplots/pyplot_formatstr.html#sphx-glr-gallery-pyplots-pyplot-formatstr-py

2. Using simple linear regression, train a linear function to predict mpg *based on displacement only*, using the training data.

   Plot the learned line, and report the training and testing error. Use the squared error function $Err = \frac{1}{2}(\sum_t (g(x^t) - r^t)^2)$.

   Package allowed.

3. Do polynomial regression: train polynomials of degree 2 and 4 and 6 to predict mpg *based on displacement only*, using the training data. Plot the 3 resulting curves and report the training and test errors in both cases (squared error Err). Do you see evidence of overfitting? Package allowed.

4. Do multiple linear regression: Train a linear function on the training set, using both input attributes, horsepower and displacement. Report the squared error Err on the test set only. Package allowed.

5. Implement k Nearest Neighbor from scratch (do not use a package). Using the data in the training set, predict the output for each example in the test, for $k = 1$, $k = 3$, and $k = 20$ (so you will run the algorithm 3 times, once for each value of k). Report the squared error Err on the test set.

6. Did k Nearest Neighbor perform better with $k = 3$ or with $k = 20$? Why do you think that happened?

7. There are many variations of k Nearest Neighbor, which can sometimes significantly improve its performance.

   - You can use a different distance function. It could be a custom distance function designed specially for your data, or a standard distance function such as the Manhattan distance or the Chebyshev distance.
   - Instead of computing an average of the k neighbors, you can compute a *weighted* average of the neighbors. A common way to do this is to weight each of the neighbors by a factor of $1/d$, where d is its distance from the test example. The weighted average of neighbors $x^1, \ldots, x^k$ is then $(\sum_{t=1}^{k} (1/d^t) r^t)/(\sum_{t=1}^{k} (1/d^t))$, where d^t is the distance of the tth neighbor.

Implement one variation of k Nearest Neighbor and choose your k. Report the squared error Err on the test set.