CS 542: Machine Learning **Summer I 2019**

Problem Set 2

Lecturer: Prof. Peter Chin *Due: June 6, 2019*

◇ Please submit your solutions via Gradescope by 23:59PM on the due date. Submit the following files:

- Written Solutions, typed or handwritten, pdf or images.
- linear_regression.py
- k_nn.py
- code_report.pdf, covering both programming questions.

◇ Late policy: there will be a penalty of 10% per day, up to three days late. After that no credit will be given.

1. **(60 points) Written Problems**

(a) (15 points) Bishop 3.3

(b) (15 points) Bishop 3.4

(c) (15 points) Bishop 3.11

(d) (15 points) Consider a Bayesian regression problem with N data points, in the dataset $(\mathbf{X}, \mathbf{t})$, with $\mathbf{t} \in \mathbb{R}^N$ the observed values. Given a data point $\mathbf{x}$ and a parameter values $\mathbf{w} \in \mathbb{R}^D$, our model outputs a prediction $y(\mathbf{x}, \mathbf{w}) \in \mathbb{R}$. Take the likelihood to be a Gaussian with mean $y(\mathbf{x}, \mathbf{w})$ and variance σ_1^2. Take as a prior over weights a multivariate Gaussian with mean 0 and covariance matrix $\sigma_2^2\mathbb{I}$. We can then write

$$p(\mathbf{w}|\mathbf{X}, \mathbf{t}) \propto \left(\prod_{n=1}^{N} \mathcal{N}(t_n; y(\mathbf{x}_n, \mathbf{w}), \sigma_1^2)\right) \times \left(\mathcal{N}(\mathbf{w}; 0, \sigma_2^2\mathbb{I})\right)$$

(i) Write down the explicit formulas for these distributions.

(ii) Finding the value of $\mathbf{w}$ that maximizes this posterior (the MAP point estimate) is equivalent to minimizing the negative log likelihood of the posterior. Take the negative natural log and simplify.

(iii) Discard terms that don't depend on $\mathbf{w}$ and thus show that maximizing the posterior is equivalent to minimizing a sum of squared errors with an l_2 penalty on $\mathbf{w}$.

(iv) Usually we formulate this problem by having a single weight λ on the l_2 penalty. If put the above result into this form, what would λ be?

2. **(120 points) Programming**

(a) **(60 points) Linear Regression**

We are given data used in a study of the homicide rate (`HOM`) in Detroit, over the years 1961-1973. The following data were collected by J.C. Fisher, and used in his paper "Homicide in Detroit: The Role of Firearms," Criminology, vol. 14, pp. 387-400, 1976. Each row is for a year, and each column are values of a variable.

FTP	UEMP	MAN	LIC	GR	NMAN	GOV	HE	WE	HOM
260.35	11.0	455.5	178.15	215.98	538.1	133.9	2.98	117.18	8.60
269.80	7.0	480.2	156.41	180.48	547.6	137.6	3.09	134.02	8.90
272.04	5.2	506.1	198.02	209.57	562.8	143.6	3.23	141.68	8.52
272.96	4.3	535.8	222.10	231.67	591.0	150.3	3.33	147.98	8.89
272.51	3.5	576.0	301.92	297.65	626.1	164.3	3.46	159.85	13.07
261.34	3.2	601.7	391.22	367.62	659.8	179.5	3.60	157.19	14.57
268.89	4.1	577.3	665.56	616.54	686.2	187.5	3.73	155.29	21.36
295.99	3.9	596.9	1131.21	1029.75	699.6	195.4	2.91	131.75	28.03
319.87	3.6	613.5	837.60	786.23	729.9	210.3	4.25	178.74	31.49
341.43	7.1	569.3	794.90	713.77	757.8	223.8	4.47	178.30	37.39
356.59	8.4	548.8	817.74	750.43	755.3	227.7	5.04	209.54	46.26
376.69	7.7	563.4	583.17	1027.38	787.0	230.9	5.47	240.05	47.24
390.19	6.3	609.3	709.59	666.50	819.8	230.2	5.76	258.05	52.33

It turns out that three of the variables together are good predictors of the homicide rate: `FTP`, `WE`, and one more variable.

Use methods described in Chapter 3 of the textbook to devise a mathematical formulation to determine the third variable. Implement your formulation and then conduct experiments to determine the third variable. In your report, be sure to provide the step-by-step mathematical formulation (citing Chapter 3 as needed) that corresponds to the implementation you turn in. Also give plots and a rigorous argument to justify the scheme you use and your conclusions.

Note: the file `detroit.npy` containing the data is given on the resources section of our course Piazza. To load the data into Python, use `X=numpy.load('detroit.npy')` command. Least-squares linear regression in Python can be done with the help of `numpy.linalg.lstsq()`.

(b) **(60 points) k Nearest Neighbors**

For this problem, you will be implementing the k-Nearest Neighbor (k-NN) classifier and evaluating on the `Credit Approval` (CA) dataset. It describes credit worthiness data (in this case, binary classification).[1] We have split the available data into a training set `crx.data.training` and a testing set `crx.data.testing`. These are both comma-separated text files (CSVs).

[1]`http://archive.ics.uci.edu/ml/datasets/Credit+Approval`

The first step to working with the CA dataset is to process the data. In looking at the data description `crx.names`, note that there are some missing values, there exist both numerical and categorical features, and that it is a relatively balanced dataset (meaning a roughly equal number of positive and negative examples – not that you should particularly care in this case, but something you should look for in general). A great Python library for handling data like this is Pandas[2]. You can read in the data with `X = pandas.read_csv('crx.data.training', header=None, na_values='?')`. The last option tells Pandas to treat the character "?" as a missing value.

Pandas holds data in a "dataframe." We'll deal with individual rows and columns, which Pandas calls "series." Pandas contains many convenient tools, but the most basic you'll use is `X.iloc[i,j]`, accessing the element in the i-th row and j-th column. You can use this for both getting and setting values. You can also slice like in normal Python, grabbing the i-th row with `[i,:]`.

正负 You can view the first 20 rows with `X.head(20)`. The last column, number 15, contains the labels. You'll see some elements are missing, marked with "NaN". While there are more sophisticated (and better) methods for imputing missing values, for this assignment, we will just use mean/mode imputation. This means that for feature 0, you should replace all of the question marks with a b as this is the mode, the most common value (regardless if you condition on the label or not). For real-valued features, just replace missing values with the label-conditioned mean (e.g. $\mu(x_1|+)$ for instances labeled as positive).

The second aspect one should consider is normalizing features. Nominal features can be left in their given form where we define the distance to be a constant value (e.g. 1) if they are different values, and 0 if they are the same. However, it is often wise to normalize real-valued features. For the purpose of this assignment, we will use z-scaling, where

$$z_i^{(m)} \leftarrow \frac{x_i^{(m)} - \mu_i}{\sigma_i} \tag{2.1}$$

such that $z_i^{(m)}$ indicates feature i for instance m (similarly $x_i^{(m)}$ is the raw input), μ_i is the average value of feature i over all instances, and σ_i is the corresponding standard deviation over all instances.

In the python file `k_nn.py`, include the following functions:

i. A function `impute_missing_data()` that accepts two Pandas dataframes, one training and one testing, and returns two dataframes with missing values filled in. In your report include your exact methods for each type of feature. Note that you are free to impute the values using statistics over the entire dataset (training and testing combined) or just training, but please state your method.

ii. A function `normalize_features()` that accepts a training and testing dataframe and returns two dataframes with real-valued features normalized.

[2] https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

iii. A function `distance()` that accepts two rows of a dataframe and returns a float, the L2 distance: $\mathcal{D}_{L2}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_i (a_i - b_i)^2}$. Note that we define $\mathcal{D}_{L2}$ to have a component-wise value of 1 for categorical attribute-values that disagree and 0 if they do agree (as previously implied). Remember not to use the label column in your distance calculation!

iv. A funtion `predict()` that accepts three arguments: a training dataframe, a testing dataframe, and an integer k - the number of nearest neighbors to use in predicting. This function should return a column of +/- labels, one for every row in the testing data.

v. A function `accuracy()` that accepts two columns, one true labels and one predicted by your algorithm, and returns a float between 0 and 1, the fraction of labels you guessed correctly.

In your report, include accuracy results on `crx.data.testing` for at least three different values of k.

vi. Try your algorithm on some other data! We've included the "lenses" dataset.[3] It has no missing values and only categorical attributes, so no need for imputation or normalization. Include accuracy results from `lenses.testing` in your report as well.

The code you submit must be your own. If you find/use information about specific algorithms from the Web, etc., be sure to cite the source(s) clearly in your sourcecode. You are not allowed to submit code downloaded from the internet (obviously).

[3]`https://archive.ics.uci.edu/ml/datasets/Lenses`