ISTA 421/521 – Homework 5

Due: Monday, December 2, 5pm

18 pts total for Undergrads, 28 pts total for Grads

Shraddha Satish Thumsi

Graduate

Instructions

In this assignment, exercises 5 and 6 require you to write python scripts; the details for those scripts, along with their .py name are described in the exercises. All of the exercises in this homework require written derivations, short answers, and/or plots, so you will also submit a .pdf of your written answers. (You can use LaTeX or any other system (including handwritten; plots, of course, must be program-generated) as long as the final version is in PDF. Handwritten pdf submissions MUST BE WRITTEN CLEARLY or we will not grade them!)

NOTE: Problems 2 and 3 are required for Graduate students only; Undergraduates may complete these exercises for extra credit equal to the point value.

(FCMA refers to the course text: Rogers and Girolami (2016), *A First Course in Machine Learning*, second edition. For general notes on using LaTeX to typeset math, see:
`http://en.wikibooks.org/wiki/LaTeX/Mathematics`)

1. [5 points] Adapted from **Exercise 5.3** of FCMA p.202:

   Compute the maximum likelihood estimates of $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ for class c of a Bayesian classifier with Gaussian class-conditionals and a set of N_c objects belonging to class c, $\mathbf{x}_1$, ..., $\mathbf{x}_{N_c}$.

   Solution.

2. [6 points] **Required only for Graduates** Adapted from **Exercise 5.4** of FCMA p.204:

   Compute the maximum likelihood estimates of q_{mc} for class c of a Bayesian classifier with multinomial class-conditionals and a set of N_c, M-dimensional objects belonging to class c: $\mathbf{x}_1, ..., \mathbf{x}_{N_c}$.

   HINT: You will need to enforce the constraint that the sum of the M q_{mc} parameters sum to 1!

   Solution.

3. [4 points] **Required only for Graduates** Adapted from **Exercise 5.5** of FCMA p.204:

   For a Bayesian classifier with multinomial class-conditionals with M-dimensional parameters $\mathbf{q}_c$, compute the posterior Dirichlet for class c when the prior over $\mathbf{q}_c$ is a Dirichlet with constant parameter α and the observations belonging to class c are the N_c observations $\mathbf{x}_1, ..., \mathbf{x}_{N_c}$.

   Solution.

4. [3 points] For a support vector machine, if we remove one of the support vectors from the training set, does the size of the maximum margin decrease, stay the same, or increase for that dataset? Why? Also justify your answer by providing a simple dataset (no more than 2-dimensions) in which you identify the support vectors, draw the location of the maximum margin hyperplane, remove one of the support vectors, and draw the location of the resulting maximum margin hyperplane. Drawing this by hand is sufficient.

   Solution.

5. [4 points] In this exercise you will use the python script, `knn.py`, which contains code to plot the decision boundary for a k-nearest neighbors classifier. Two data sources are also provided in the `data` directory: `knn_binary_data.csv` and `knn_three_class_data.csv`.

   In `knn.py`, the function `knn` is to compute the k-nearest neighbors classification for a point `p`, but the functionality is not currently implemented. You *can* run the code in its unimplemented state, but the decision boundary will not display (everything will be considered the same class). You must implement the core of the classifier. Use a Euclidean distance measure for determining the neighbors (note: you don't need to take the square root! The sum of squares is sufficient). You do not need to implement any fancy indexing – you can simply search for the k nearest points by searching over the pairwise distance of the input (`p`) to all of the "training" inputs (`x`). Use the maximum class frequency as the class decision rule. You do not need to worry about doing anything special for breaking ties. The numpy function `argsort` and the python `collections.Counter` may be of use, but are not required. Submit the `knn.py` file with your implementation. In your written solution, run the code on both of the provided data sources (`knn_binary_data.csv` and `knn_three_class_data.csv`.), and for each, plot the decision boundary for $k = \{1, 5, 10, 59\}$. Include informative captions and describe any patterns you see in the decision boundaries as k changes.

   A set of unit tests are provided to test your output classification for the mesh of inputs used to generate the decision-boundary plots.

   Solution.

6. [6 points] Using your implementation of your KNN classifier in exercise 5, write a script to perform 10-fold cross-validation in a search for the best choice of K. You will implement your script in the stub python file `knn-cv.py`.

Remember to randomly partition your data at the start of the CV procedure. For this relatively small data set (especially for the 3-class case), the randomization of the 10-fold partitioning will lead to a fair amount of variance, so repeat this overall procedure 10 times, averaging the accuracy results (i.e,: have an outer loop that runs 10-fold CV 10 times, averaging accuracy per k across the 10-fold CV results).

Run your script on both data sources: `knn_binary_data.csv` and `knn_three_class_data.csv`. For `knn_binary_data.csv`, search for K in the range $1 \leq K \leq 60$, and for `knn_three_class_data.csv`, search for K in the range $1 \leq K \leq 80$.

In your written solution, provide a plot of K (x-axis) against classification error (i.e., the ratio of points misclassified to total, on the y-axis) for each data set, and report the *best* k for each.

There are no unit tests for this exercise.

Solution.