ISTA 421/521 – Homework 5

Due: Wednesday, December 5, 5pm

20 pts total for Undergrads, 24 pts total for Grads

Ken Youens-Clark

Graduate

Instructions

In this assignment, exercises 5 and 6 require you to write small 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.)

The final submission will include (minimally) the two scripts you need to write for problems 5 and 6, and a PDF version of your written part of the assignment. You are required to create either a .zip or tarball (.tar.gz / .tgz) archive of all of the files for your submission and submit your archive to the d2l dropbox by the date/time deadline above.

NOTE: Problem 3 is required for Graduate students only; Undergraduates may complete this problem 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.

Likelihood

$$L = p(\mathbf{X}^c|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

is product of (Gaussian) probabilities

$$= \prod_n^N \mathcal{N}(x_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

$$= \prod_n^N \frac{1}{(2\pi)^{N/2}|\boldsymbol{\Sigma}_c|^{1/2}} \exp\left\{-\frac{1}{2}(x_n \ \ - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(x_n - \boldsymbol{\mu}_c)\right\}$$

$$\propto |\boldsymbol{\Sigma}_c|^{-N_c/2} \exp\left\{\sum_n^{N_c} -\frac{1}{2}(x_n \ \ - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(x_n - \boldsymbol{\mu}_c)\right\}$$

Use log to handle exponent

$$\log(L) = \left(-\frac{N_c}{2}\log(|\boldsymbol{\Sigma}_c|)\right)\left(-\frac{1}{2}\sum_n^{N_c}(x_n \ \ - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(x_n - \boldsymbol{\mu}_c)\right)$$

Take derivative w.r.t. $\boldsymbol{\mu}_c$

$$\frac{\partial \log(L)}{\partial \boldsymbol{\mu}_c} = -\frac{1}{2}\sum_n^{N_c} 2\boldsymbol{\Sigma}_c^{-1}(x_n - \boldsymbol{\mu}_c) = -\sum_n^{N_c} \boldsymbol{\Sigma}_c^{-1}(x_n - \boldsymbol{\mu}_c)$$

Set to zero, solve for $\boldsymbol{\mu}_c$

$$-\sum_n^{N_c} \boldsymbol{\Sigma}_c^{-1}(x_n - \boldsymbol{\mu}_c) = 0$$

$$\sum_n^{N_c} \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c - \sum_n^{N_c} \boldsymbol{\Sigma}_c^{-1}x_n = 0$$

$$\boldsymbol{\Sigma}_c^{-1}N_c\boldsymbol{\mu}_c - \boldsymbol{\Sigma}_c^{-1}\sum_n^{N_c} x_n = 0$$

$$\boldsymbol{\Sigma}_c^{-1}N_c\boldsymbol{\mu}_c = \boldsymbol{\Sigma}_c^{-1}\sum_n^{N_c} x_n$$

$$N_c\boldsymbol{\mu}_c = \sum_n^{N_c} x_n$$

$$\boldsymbol{\mu}_c = \frac{1}{N_c}\sum_n^{N} x_n$$

Take derivative w.r.t. $\boldsymbol{\Sigma}_c$, set to zero, solve

$$\frac{\partial \log(L)}{\partial \boldsymbol{\Sigma}_c} = -\frac{N_c}{2}|\boldsymbol{\Sigma}_c|^{-1} + \frac{1}{2}\sum_n^{N_c}(x_n \ \ - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-2}(x_n - \boldsymbol{\mu}_c)$$

$$\frac{N_c}{2}|\boldsymbol{\Sigma}_c|^{-1} = \frac{1}{2}\sum_n^{N_c}(x_n \ \ - \boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-2}(x_n - \boldsymbol{\mu}_c)$$

The correct answers:

$$\boldsymbol{\mu}_c = \frac{1}{N_c}\sum_{n=1}^{N_c} \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_c = \frac{1}{N_c}\sum_{n=1}^{N_c}(\mathbf{x}_n - \boldsymbol{\mu}_c)(\mathbf{x}_n - \boldsymbol{\mu}_c)^\top$$

2. [4 points] 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: $\mathbf{x}_1, ..., \mathbf{x}_{N_c}$.

   Solution.

$$
\begin{aligned}
L &= P(x_n|\mathbf{q}) \\
&= \left(\frac{s_n!}{\prod_{m=1}^{M} x_{nm}!}\right) \prod_{m=1}^{M} q_m{}^{x_{nm}}
\end{aligned}
$$

   The answer

$$
q_{cm} = \frac{\sum_{n=1}^{N_c} x_{nm}}{\sum_{m'=1}^{M} \sum_{n=1}^{N_c} x_{nm'}}
$$

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 python file, 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 this yourself. 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.

   Solution.

6. [4 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. Remember to randomize your data at

the start of the CV procedure, but use the same CV folds for each K. Make your script search in the range of $1 \leq K \leq 30$. Run your script on both data sources: `knn_binary_data.csv` and `knn_three_class_data.csv`. In your written solution, provide a plot of K (x-axis) to classification error (ratio of points misclassified to total; y-axis) for each data set, and report the *best* k for each. Submit your script as a python file named `knn-cv`.

Solution.