ELEC 425/825 - Fall 2019
Machine Learning and Deep Learning
Assignment I

Due: 9:00PM, October 15th, 2019

Important Notes

Please kindly understand this assignment is an **"individual"** assignment. You are allowed to discuss the general idea of the assignment but your discussion may not include the specific answers to any of the problems. Please kindly understand that any form of plagiarism may lead to a mark of 0 for your assignment.

Submission Guide

- Submit a **zip** file to "Assignment I" Dropbox folder on onQ, which contains: (1) A **pdf** file that includes your answers to the assignment questions; your answer should be typed but not hand-written; **source code you write to solve a problem should be included in this PDF file**, right after your answers to that question. (2) You should also put your source code into separate source code files; for example, if you use Matlab, you should also include your .m files in your zip file. (3) Include also a README file telling us what each program/code file is for and how to run your code.
- **Late submission:** If you are late, for every 2 hours, you will lose **15**% of your assignment marks.

1 Training Conditional Gaussian Classifiers (4 marks)

1.1 Handwritten Digit Data

You will build two classifiers to label images of handwritten digits collected by the United States Post Office. The images $\mathbf{x}$ are 8 by 8 in size, which we will represent as a vector of dimension 64 by listing all the pixel values in raster scan order. The labels y are $1, 2, ..., 9, 10$ corresponding to which character was written in the image. Label 10 is used for the digit '0'. There are 700 training cases and 400 test cases for each digit, they can be found in the files `a1digits.mat` and `a1digits.zip`. Before we start, here are some Matlab tips:

- The `imagesc` function can be used to display vectors as images. In particular, try the line:
  `imagesc(reshape(xxx,8,8)'); axis equal; axis off; colormap gray;`
  to display some vector **xxx** you want to display. The subplot command is useful for displaying many small images beside each other.
- The `repmat` command in conjunction with sum and the operators .* and ./ are helpful in re-normalizing arrays so that the rows or columns sum to one.
- The expression `(M > a)` for a matrix M and a scalar a performs the comparison at every element and evaluates to a binary matrix the same size as `M`.

1.2 Training (4 marks)

In this question, you'll train class conditional Gaussians with independent features (spherical covariance matrices). Start with the following generative model for a discrete class label $k \in \{1, 2, ..., K\}$ and a real valued vector of D features $\mathbf{x} = (x_1, x_2, ..., x_D)$:

$$p(C_k) = \alpha_k \tag{1}$$

$$p(\mathbf{x}|C_k) = (2\pi\sigma^2)^{-D/2} exp\{-\frac{1}{2\sigma^2}\sum_{i=1}^{D}(x_i - \mu_{ki})^2\} \tag{2}$$

where α_k is the prior on class k, σ^2 is the shared variance for all features in all classes, and μ_{ki} is the mean of the feature i conditioned on class k.

- Using maximum likelihood estimation, one can derive the maximum-likelihood solution for conditional Gaussian for the class k as follows:

$$\hat{u}_{ki} = \frac{\sum_{j=1}^{m_k} x_{kji}}{m_k} \tag{3}$$

  where m_k is the number of training data points in class k. Note that σ^2 is shared and can be estimated with:

$$\hat{\sigma}^2 = \frac{\sum_{k=1}^{K}\sum_{j=1}^{m_k}\sum_{i=1}^{D}(x_{kji} - \mu_{ki})^2}{DM} \tag{4}$$

  where M is the number of all training data points over all K classes. (As mentioned in tutorial, in Matlab you can often use matrix/vector operations to avoid explicit *for/while* loop, which often makes your code look concise.)

- Using the above maximum-likelihood solution and the provided training data, you can learn/fit class conditional Gaussians. You should get parameters μ_{ki} for $k \in (0, ..., 9), i \in (1, ..., 64)$.
  (You can assume $\alpha_k = 1/10$ since all classes have the same number of observations.)

- Hand in plot showing an 8 by 8 image of each mean μ_k, all ten means side by side (try using `subplot`). Also write somewhere on the plot the value of σ (the pixel noise standard deviation).

2 Training Naive Bayes Classifiers (4 marks)

Using the same dataset provided above, please train a Naive Bayes Classifier.

- Convert the real-valued features $\mathbf{x}$ into binary features $\mathbf{b}$ by thresholding: $b_i = 1$ if $x_i > 0.5$ otherwise $b_i = 0$.

- Using these new binary features $\mathbf{b}$ and the class labels, train a Naive Bayes classifier on the training set:

$$p(C_k) = \alpha_k \tag{5}$$
$$p(b_i = 1|C_k) = \eta_{ki} \tag{6}$$
$$p(\mathbf{b}|C_k, \eta) = \prod_i \eta_{ki}^{[b_i=1]}(1 - \eta_{ki})^{[b_i=0]} \tag{7}$$

- You should get parameters $\eta_{ki} \equiv p(b_i = 1|C_k)$ for $k \in (0, ..., 9), i \in (1, ..., 64)$. (You can assume all class priors are equal since all classes have the same number of observations.)

- Hand in plot showing an 8 by 8 image of each vector η_k, all ten side by side (try using `subplot`).

3 Test Performance (2 marks)

- Use the parameters you fit on the training set to compute $p(C_k|\mathbf{x})$ for each of the test cases under both Naive Bayes and Gaussian-conditionals.

- Select the most likely class for each test case under each classifier. If this matches the label, the classifier is correct. If not, the classifier has made an error. Hand in a 2 by 11 table showing how many errors (out of 400) each classifier makes on each of the 10 test sets and what the overall error rate (in %) is.