HOMEWORK 3 - V4

CSC311 Fall 2019

University of Toronto

Version history:
V0 → V1: add another hint to Q1.a
V1 → V2: deadline extended
V2 → V3: fix typo in Q1.a hint $\pi_{10} \to \pi_9$
V3 → V4: deadline extended

- **Deadline:** Nov. 15, at 23:59.
- **Submission:** You need to submit one pdf file through MarkUs including all your answers, plots, and your code. You can produce the file however you like (e.g. LaTeX, Microsoft Word, etc), as long as it is readable. Points will be deducted if we have a hard time reading your solutions or understanding the structure of your code. For each question, you should append your code right after your answers to that question.

1. Fitting a Naïve Bayes Model, 40 pts. In this question, we'll fit a Naïve Bayes model to the MNIST digits using maximum likelihood. The starter code will download the dataset and parse it for you: Each training sample $(\mathbf{t}^{(i)}, \mathbf{x}^{(i)})$ is composed of a vectorized binary image $\mathbf{x}^{(i)} \in \{0,1\}^{784}$, and 1-of-10 encoded class label $\mathbf{t}^{(i)}$, i.e., $t_c^{(i)} = 1$ means image i belongs to class c.

For $p(c \,|\, \boldsymbol{\pi}) = \pi_c$ and $p(x_j = 1 \,|\, c, \boldsymbol{\theta}, \boldsymbol{\pi}) = \theta_{jc}$, Naïve Bayes defines the joint probability of the each data point $\mathbf{x}$ and its class label c as follows:

$$p(\mathbf{x}, c \,|\, \boldsymbol{\theta}, \boldsymbol{\pi}) = p(c \,|\, \boldsymbol{\theta}, \boldsymbol{\pi}) p(\mathbf{x} \,|\, c, \boldsymbol{\theta}, \boldsymbol{\pi}) = p(c \,|\, \boldsymbol{\pi}) \prod_{j=1}^{784} p(x_j \,|\, c, \theta_{jc}).$$

Here, $\boldsymbol{\theta}$ is a matrix of probabilities for each pixel and each class, so its dimensions are 784×10 (Note that in the lecture, we simplified notation and didn't write the probabilities conditioned on the parameters, i.e. $p(c|\boldsymbol{\pi})$ is written as $p(c)$ in lecture slides).

For binary data, we can write the Bernoulli likelihood as

$$p(x_j \,|\, c, \theta_{jc}) = \theta_{jc}^{x_j} (1 - \theta_{jc})^{(1-x_j)}, \tag{1.1}$$

which is just a way of expressing $p(x_j = 1|c, \theta_{jc}) = \theta_{jc}$ and $p(x_j = 0|c, \theta_{jc}) = 1 - \theta_{jc}$ in a compact form. For the prior $p(\mathbf{t} \,|\, \boldsymbol{\pi})$, we use a categorical distribution (generalization of Bernoulli distribution to multi-class case),

$$p(t_c = 1 \,|\, \boldsymbol{\pi}) = p(c \,|\, \boldsymbol{\pi}) = \pi_c \;\text{ or equivalently }\; p(\mathbf{t} \,|\, \boldsymbol{\pi}) = \Pi_{j=0}^{9} \pi_j^{t_j} \quad \text{where} \quad \sum_{i=0}^{9} \pi_i = 1,$$

where $p(c \,|\, \boldsymbol{\pi})$ and $p(\mathbf{t} \,|\, \boldsymbol{\pi})$ can be used interchangeably. You will fit the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$ using MLE and MAP techniques, and both cases below, your fitting procedure can be written as a few simple matrix multiplication operations.

(a) First, derive the *maximum likelihood estimator* (MLE) for the class-conditional pixel probabilities $\boldsymbol{\theta}$ and the prior $\boldsymbol{\pi}$. Hint-1: We saw in lecture that MLE can be thought of as 'ratio of counts' for the data, so what should $\hat{\theta}_{jc}$ be counting? Derivations should be rigorous. Hint-2: Similar to the binary case, when calculating the MLE for π_j for $j = 0, 1, ..., 8$, write $p(\boldsymbol{t}^{(i)} \mid \boldsymbol{\pi}) = \Pi_{j=0}^{9} \pi_j^{t_j^{(i)}}$ and in the log-likelihood replace $\pi_9 = 1 - \Sigma_{j=0}^{8} \pi_j$, and then take derivatives w.r.t. π_j. This will give you the ratio $\hat{\pi}_j / \hat{\pi}_9$ for $j = 0, 1, .., 8$. You know that $\hat{\pi}_j$'s sum up to 1.

(b) Derive the log-likelihood $\log p(\mathbf{t}|\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\pi})$ for a single training image.

(c) Fit the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$ using the training set with MLE, and try to report the average log-likelihood per data point $\frac{1}{N} \Sigma_{i=1}^{N} \log p(\mathbf{t}^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$, using Equation (1.1). What goes wrong? (it's okay if you can't compute the average log-likelihood here).

(d) Plot the MLE estimator $\hat{\boldsymbol{\theta}}$ as 10 separate greyscale images, one for each class.

(e) Derive the *Maximum A posteriori Probability* (MAP) estimator for the class-conditional pixel probabilities $\boldsymbol{\theta}$, using a Beta(3, 3) prior on each θ_{jc}. Hint: it has a simple final form, and you can ignore the Beta normalizing constant.

(f) Fit the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$ using the training set with MAP estimators from previous part, and report both the average log-likelihood per data point, $\frac{1}{N} \Sigma_{i=1}^{N} \log p(\mathbf{t}^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$, and the accuracy on both the training and test set. The accuracy is defined as the fraction of examples where the true class is correctly predicted using $\hat{c} = \text{argmax}_c \log p(t_c = 1|\mathbf{x}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$.

(g) Plot the MAP estimator $\hat{\boldsymbol{\theta}}$ as 10 separate greyscale images, one for each class.

2. Generating from a Naïve Bayes Model, 30 pts. Defining a joint probability distribution over the data lets us generate new data, and also lets us answer all sorts of queries about the data. This is why these models are called *Generative Models*. We will use the Naïve Bayes model trained in previous question to generate data.

(a) True or false: Given this model's assumptions, any two pixels x_i and x_j where $i \neq j$ are independent given c.

(b) True or false: Given this model's assumptions, any two pixels x_i and x_j where $i \neq j$ are independent after marginalizing over c.

(c) Using the parameters fit using MAP in Question 1, produce random image samples from the model. That is, randomly sample and plot 10 binary images from the marginal distribution $p(\mathbf{x}|\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$. Hint: To sample from $p(\mathbf{x} \mid \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$, first sample random variable c from $p(c \mid \hat{\boldsymbol{\pi}})$ using `np.random.choice`, then depending on the value of c, sample x_j from $p(x_j \mid c, \hat{\theta}_{jc})$ for $j = 1, ..., 784$ using `np.random.binomial(1,..)`. These functions can take matrix probabilities as input, so your solution to this part should be a few lines of code.

(d) (Optional - 0 pts) One of the advantages of generative models is that they can handle missing data, or be used to answer different sorts of questions about the model. Derive $p(\mathbf{x}_{bottom}|\mathbf{x}_{top}, \boldsymbol{\theta}, \boldsymbol{\pi})$, the marginal distribution of a single pixel in the bottom half of an image given the top half, conditioned on your fit parameters. Hint: you'll have to marginalize over c.

(e) (Optional - 0 pts) For 20 images from the training set, plot the top half the image concatenated with the marginal distribution over each pixel in the bottom half. i.e. the bottom half of the image should use grayscale to represent the marginal probability of each pixel being 1 (darker for values close to 1).

3. Principal Component Analysis, 30 pts. Using the numpy datafile `digits.npy` and the `utils.py` dataloading helper code, you will find 6 sets of 16×16 greyscale images in vector format (the pixel intensities are between 0 and 1 and were read into the vectors in a raster-scan manner). The images contain handwritten 2's and 3's, scanned from postal envelopes. `train2` and `train3` contain examples of 2's and 3's respectively to be used for training. There are 300 examples of each digit, stored as 300×256 matrices. Note that each data vector is a row of data matrices returned by `load_data` function. `valid2` and `valid3` contain data to be used for validation (100 examples of each digit) and `test2` and `test3` contain test data to be used for final evaluation **only** (200 examples of each digit).

Apply the PCA algorithm to the 600 x 256 digit images (computing all 256 of the eigenvalues and eigenvectors, don't forget to center the data). Then you should plot the (sorted) eigenvalues as a descending curve. This plot shows the spectrum of the data, and roughly tells you how much variance is contained along each eigenvector direction. Then view the first 3 eigen-images (reshape each of the first 3 eigenvectors and use `imagesc` to see these as images) as well as the mean of data. This part is for you to gain some intuition about how PCA works. You do not need to write this part up!

(a) For each image in the validation set, subtract the mean of training data and project it into the low-dimensional space spanned by the first K principal components of training data. After projection, use a 1-NN classifier on K dimensional features (the code vectors) to classify the digit in the low-dimensional space. You need to implement the classifier yourself. You will do the classification under different K values to see the effect of K. Here, let K = 2, 5, 10, 20, 30, and under each K, classify the validation digits using 1-NN. Plot the results, where the plot should show the curve of validation set classification error rates versus number of eigenvectors you keep, i.e., K.

(b) If you wanted to choose a particular model from your experiments as the best, which model (number of eigenvectors) would you select? Why?

(c) Report the performance of your final classifier over the test data.