Homework 4 - Nov. 3

Deadline: Monday, Nov. 24, at 11:59pm.

Submission: You need to submit through Crowdmark (you can find the link on Quercus) with your answers to Questions 1, 2, 3, and 4. You will upload your answers to each subquestion separately. You can produce the submissions however you like (e.g. LaTeX, Microsoft Word, scanner), as long as they are readable.

Marking: Your mark will be out of 7.5. This mark will be your mark on *either* question 1, 2, 3, *or* 4. We will decide which question to mark after the deadline, and we will mark the same question for everyone in the class. To aid your learning, we will be releasing the solutions to both questions, and covering the solutions in office hours.

Neatness: We reserve the right to deduct a point for neatness, if we have a hard time reading your solutions or understanding the structure of your code.

Late Submission: 10% of the total possible marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.

Computing: To install Python and required libraries, see the instructions on the course web page.

Other Policies: See the syllabus[1] for detailed collaboration, generative AI, and academic integrity policies.

1. **[7.5 pts] Principal Component Analysis.** In this problem, you will gain intuition on how PCA works by implementing the algorithm on the same digits dataset. You will complete the provided code in `pca.py` and experiment with the completed code. Carefully read the provided code in `pca.py`. You should understand the code instead of using it as a black box. You will apply the PCA algorithm to the 600×256 digit images (computing all 256 eigenvalues and eigenvectors).

   (a) **[3 pts]** Implement the function `pca` located at `pca.py`. While implementing the function, remember to vectorize the operations; you should not write any `for-loops`. Include your code in the report. You may optionally visualize the eigenvectors with the provided function `show_eigenvectors`.

   (b) **[4 pts]** 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 in function `pca_classify`. You will do the classification under different K values to see the effect of K. Choose $K = \{2, 5, 10, 20, 30\}$ and, under each K, classify the validation digits using 1-NN. Plot and report results, where the plot should show the curve of validation set classification accuracy versus number of eigenvectors you keep, i.e., K. Include the code in your report as well.

   (c) **[0.5 pt]** What is the best choice of K for 1-NN according to the validation accuracy? Report the classification accuracy of this classifier over the test data.

[1] https://www.cs.toronto.edu/~cmaddis/courses/sta314_f25/sta314_f25_syllabus.pdf

2. **[7.5pts] Categorial Distribution.** In this problem you will consider a Bayesian approach to modelling categorical outcomes. Let's consider fitting the categorical distribution, which is a discrete distribution over K outcomes, which we'll number 1 through K. The probability of each category is explicitly represented with parameter θ_k. For it to be a valid probability distribution, we clearly need $\theta_k \geq 0$ and $\sum_k \theta_k = 1$. We'll represent each observation $\mathbf{x}$ as a 1-of-K encoding, i.e, a vector where one of the entries is 1 and the rest are 0. Under this model, the probability of an observation can be written in the following form:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{k=1}^{K} \theta_k^{x_k}.$$

Suppose you observe a dataset,

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}.$$

Denote the count for outcome k as $N_k = \sum_{i=1}^{N} x_k^{(i)}$ and $N = \sum_{k=1}^{K} N_k$. Recall that each data point is in the 1-of-K encoding, i.e., $x_k^{(i)} = 1$ if the ith datapoint represents an outcome k and $x_k^{(i)} = 0$ otherwise.

(a) **[2.5pts]** First, derive $\hat{\theta}_k$, which is the *maximum likelihood estimator (MLE)*, for the class probabilities θ_k. You may assume that $N_k > 0$ for this question. Derivations should be rigorous.

Hint 1: We saw in lecture that MLE can be thought of as 'ratio of counts' for the data, so what should $\hat{\theta}_k$ be counting?

Hint 2: Similar to the binary case, write $p(\mathbf{x}^{(i)} \,|\, \boldsymbol{\theta}) = \Pi_{k=1}^{K} \theta_k^{x_k^{(i)}}$. The challenge in maximizing the log-likelihood is that this problem is a constrained maximization under the constraint that $\sum_{k=1}^{K} \theta_k = 1$. To overcome this, we will use a generalization of the idea from the coin flip example in lecture. We can turn the MLE maximization problem into an easier constrained problem by setting $\theta_K = 1 - \sum_{k=1}^{K-1} \theta_k$. Note that this is a maximization problem over the set $\{(\theta_k)_{k\neq K} | \theta_k > 0, \sum_{k\neq K} \theta_k < 1\}$. Although this is still constrained, the log likelihood is a concave function that goes to $-\infty$ as we approach the boundary of the constraint set, so the function attains its maximum in the interior of the region when $\partial \log p(\mathcal{D}|\boldsymbol{\theta})/\partial\theta_k = 0$.

(b) **[2.5pts]** Now we will take a Bayesian approach. For the prior, we'll use the Dirichlet distribution, which is defined over the set of probability vectors (i.e. vectors that are nonnegative and whose entries sum to 1). Its PDF is as follows:

$$p(\boldsymbol{\theta}) \propto \theta_1^{a_1-1} \cdots \theta_K^{a_k-1}.$$

What is the probability distribution of the posterior distribution $p(\boldsymbol{\theta} \,|\, \mathcal{D})$? Don't just give the density of the posterior, say which family of distributions it belongs to.

(c) **[1.5pts]** Still assuming the Dirichlet prior distribution, determine the MAP estimate of the parameter vector $\boldsymbol{\theta}$. For this question, you may assume each $a_k > 1$.

(d) **[1pt]** Now, suppose that your friend said that they had a hidden $N+1$st outcome, $\mathbf{x}^{(N+1)}$, drawn from the same distribution as the previous N outcomes. Your friend does not want to reveal the value of $\mathbf{x}^{(N+1)}$ to you. So, you want to use your Bayesian model

to predict what *you* think $\mathbf{x}^{(N+1)}$ is likely to be. The "proper" Bayesian predictor is the so-called *posterior predictive distribution*:

$$p(\mathbf{x}^{(N+1)}|\mathcal{D}) = \int p(\mathbf{x}^{(N+1)}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta}$$

What is the probability that the $N+1$ outcome was k, i.e., the probability that $x_k^{(N+1)} = 1$, under your posterior predictive distribution? *Hint: A useful fact is that if* $\boldsymbol{\theta} \sim \text{Dirichlet}(a_1, \ldots, a_K)$*, then*

$$\mathbb{E}[\theta_k] = \frac{a_k}{\sum_{k'} a_{k'}}.$$

Report your answers to the above questions.

3. **[7.5pts] Gaussian Naïve Bayes.** In this question, you will derive the maximum likelihood estimates for Gaussian Naïve Bayes, which is just like the naïve Bayes model from lecture, except that the features are continuous, and the conditional distribution of each feature given the class is (univariate) Gaussian rather than Bernoulli. Start with the following generative model for a random discrete class label $t \in \{1, 2, ..., K\}$ and a random real valued vector of D features $\mathbf{x} \in \mathbb{R}^D$:

$$p(t = k) = \alpha_k \tag{0.1}$$

$$p(\mathbf{x}|t = k) = \left(\prod_{d=1}^{D} 2\pi\sigma_d^2\right)^{-1/2} \exp\left\{-\sum_{d=1}^{D} \frac{1}{2\sigma_d^2}(x_d - \mu_{kd})^2\right\} \tag{0.2}$$

where $\alpha_k \geq 0$ is the prior on class k, $\sigma_d^2 > 0$ are the variances for each feature, which are shared between all classes, and $\mu_{kd} \in \mathbb{R}$ is the mean of the feature d conditioned on class k. We write $\boldsymbol{\alpha}$ to represent the vector with elements α_k and similarly $\boldsymbol{\sigma}$ is the vector of variances. The matrix of class means is written $\boldsymbol{\mu}$ where the kth row of $\boldsymbol{\mu}$ is the mean for class k.

   (a) **[1.5pt]** Use Bayes' rule to derive an expression for $p(t = k|\mathbf{x})$. *Hint: Use the law of total probability to derive an expression for* $p(\mathbf{x})$.

   (b) **[1.5pt]** Write down an expression for the likelihood function (LL)

$$\ell(\boldsymbol{\theta}) = \log p(t^{(1)}, \mathbf{x}^{(1)}, t^{(2)}, \mathbf{x}^{(2)}, \cdots, t^{(N)}, \mathbf{x}^{(N)}) \tag{0.3}$$

   of a particular dataset $D = \{(t^{(1)}, \mathbf{x}^{(1)}), (t^{(2)}, \mathbf{x}^{(2)}), \cdots, (t^{(N)}, \mathbf{x}^{(N)})\}$ with parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ and this model. (Assume the data are i.i.d.)

   (c) **[4.5pts]** Take partial derivatives of the likelihood with respect to each of the parameters μ_{kd} and with respect to the shared variances σ_d^2. Report these partial derivatives and find the maximum likelihood estimates for μ_{kd} and σ_d^2. You may assume that each class appears at least once in the dataset, i.e. the number of times N_k that class k appears in the dataset is $N_k > 0$.

Report your answers to the above questions.

4. **[7.5pts] Gaussian Discriminant Analysis.** For this question you will build classifiers to label images of handwritten digits. Each image is 16 by 16 pixels and is represented as a vector of dimension 256 by listing all the pixel values in raster scan order. The images are grayscale and the pixel values are between 0 and 1. The labels y are $0, 1$ corresponding to

which character was written in the image, either "2" or "3". These are the same digits we used in HW3 with logistic regression.

A skeleton (`gda.py`) is is provided for each question that you should use to structure your code.

Using maximum likelihood, fit a set of 2 class-conditional Gaussians with a separate, full covariance matrix for each class. Remember that the conditional multivariate Gaussian probability density is given by,

$$p(\mathbf{x} \mid t = k) = (2\pi)^{-D/2}|\Sigma_k|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right\} \tag{0.4}$$

where $\boldsymbol{\mu}_k \in \mathbb{R}^D, \Sigma_k \in \mathbb{R}^{D\times D}$ and positive-definite. You should take $p(t = k) = \frac{1}{2}$. You will compute parameters μ_{kj} and Σ_k for $k \in \{0, 1\}, j \in \{0, \ldots, 255\}$. You should implement the covariance computation yourself (i.e. without the aid of 'np.cov'). *Hint: To ensure numerical stability you may have to add a small multiple of the identity to each covariance matrix. For this assignment you should add* $0.1\mathbf{I}$ *to each covariance matrix.*

(a) **[7pts]** Complete the 5 functions that are not complete in the starter code (**[1.4pts]** each). Include the function body for each completed function in your report.

(b) **[0.5pts]** Report the average conditional log-likelihood, i.e. $\frac{1}{N}\sum_{i=1}^{N} \log p(t^{(i)} \mid \mathbf{x}^{(i)})$, of the trained model on both the train and test set. Report the accuracy, of the classifier that selects most likely posterior class for each data point using the trained model, on the train and test set.