CAP 6610 Machine Learning, Spring 2022

Homework 1 Solution

1. (10 points) *Numerical check of the least squares solution.* Use your favorite language to generate a random 40×10 matrix $\boldsymbol{\Phi}$ and a random 40-vector $\boldsymbol{\psi}$. Compute the least squares solution $\boldsymbol{\theta}^\star = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\psi}$ and the associated loss $\|\boldsymbol{\Phi}\boldsymbol{\theta}^\star - \boldsymbol{\psi}\|^2$. (There may be several ways to do this, depending on the software package you use. In MATLAB or Julia, the command is simply `Phi\psi`.) Generate a random 10-vectors $\boldsymbol{\delta}$ and verify that $\|\boldsymbol{\Phi}(\boldsymbol{\theta}^\star + \boldsymbol{\delta}) - \boldsymbol{\psi}\|^2 > \|\boldsymbol{\Phi}\boldsymbol{\theta}^\star - \boldsymbol{\psi}\|^2$ holds. Repeat several times with different values of $\boldsymbol{\delta}$.

   Submit your code, including the code that checks whether the expected inequality that involves $\boldsymbol{\delta}$ holds.

2. (10 points) *Smokers.* According to the Center for Disease Control (CDC), "Compared to nonsmokers, men who smoke are about 23 times more likely to develop lung cancer and women who smoke are about 13 times more likely." The CDC also states that roughly 15% of all women are smokers, 18% of all adults are smokers. Assume that half the adults are women.

   (a) If you learn that a woman has been diagnosed with lung cancer, and you know nothing else about her, what is the probability she is a smoker?

   (b) What fraction of adult smokers in the USA are women?

   Hint. Let A be the event that "a woman smokes," and B be the event that "a woman gets lung cancer."

   Solution.

   (a) Let c be the fraction of nonsmoking women who get lung cancer (over some time period). Then $13c$ is the fraction of smoking women who get lung cancer over the same time period.

   The total probability a woman getting lung cancer is the sum of the probability the woman is smoker and gets lung cancer plus the probability she is a nonsmoker and gets lung cancer. Let A be the event corresponding to a woman smoking and let B be the event corresponding to a woman getting cancer. Thus, we have

   $$\Pr[B] = \Pr[B|A]\Pr[A] + \Pr[B|A^c]\Pr[A^c] = 13c(0.15) + c(0.85).$$

   Therefore, the conditional probability a woman is a smoker given she got cancer is

   $$\begin{aligned}\Pr[A|B] &= \frac{\Pr[B|A]\Pr[A]}{\Pr[B|A]\Pr[A] + \Pr[B|A^c]\Pr[A^c]} \\ &= \frac{13c(0.15)}{13c(0.15) + c(0.85)} = \frac{1.95}{2.80} \approx 70\%.\end{aligned}$$

   (b) The fraction of adults that smoke, namely 18%, is the average of the fraction of women that smoke, 15%, and the fraction of men that smoke. It follows that 21% of men smoke because $(15 + 21)/2 = 18$. Thus, the ratio of the number of women that smoke to the total number of adults that smoke is

   $$15/(15 + 21) = 5/12 \approx 0.4167.$$

Another way to arrive at the same answer is to define the events W, S, which encode whether one is a woman (W) or not (W^c) and whether one smokes or not (S, S^c, respectively). Then, we are interested in

$$\Pr[W|S] = \frac{\Pr[W,S]}{\Pr[S]} = \frac{\Pr[S|W]\Pr[W]}{\Pr[S]} = \frac{0.15 \times 0.5}{0.18} \approx 0.4167.$$

3. (10 points) Recall that the PMF of a Poisson random variable is

$$p(x;\lambda) = \frac{\lambda^x e^{-\lambda}}{x!},$$

with one parameter λ. Given i.i.d. samples $x_1, ..., x_n \sim \text{Pois}(\lambda)$, derive the maximum likelihood estimate (MLE) for λ.

Solution. Maximum likelihood tries to solve the following problem

$$\underset{\lambda}{\text{minimize}} \quad \sum_{i=1}^{n} \left(x_i \log \lambda - \lambda - \log(x_i!)\right).$$

Taking derivative with respect to λ and set it equal to zero gives

$$\sum_{i=1}^{n} \left(\frac{x_i}{\lambda} - 1\right) = 0.$$

The result is

$$\lambda = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

again the sample average of the data set.

4. (10 points) The function `randn(d,1)` generates a multivariate normal variable $\boldsymbol{x} \in \mathbb{R}^d$ with zero mean and covariance $\boldsymbol{I}$. Describe how to generate a random variable from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. *Hint.* If $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\top)$.

Solution. Find a "square-root decomposition" of $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$. This can be done by, for example, the Cholesky decompositon or the eigen-decomposition $\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$ by setting $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}^{-1/2}$. Let $\boldsymbol{z}$ be generated from $\mathcal{N}(0, \boldsymbol{I})$, then $\boldsymbol{L}^\top\boldsymbol{z} + \boldsymbol{\mu}$ follows the distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

5. (10 points) Consider a data set in which each data sample i is associated with a weighting factor $r_i > 0$, and we instead try to minimize the weighted MSE function

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} r_i(y_i - \boldsymbol{\phi}_i^\top\boldsymbol{\theta})^2.$$

Find an expression for the solution $\boldsymbol{\theta}^\star$ that minimizes this loss function.

Solution. The gradient with respect to one component function $r_i(y_i - \boldsymbol{\phi}_i^\top\boldsymbol{\theta})^2$ is

$$\boldsymbol{\phi}_i r_i(\boldsymbol{\phi}_i^\top\boldsymbol{\theta} - y_i).$$

Therefore, the gradient for the whole function is

$$\sum_{i=1}^{n} \boldsymbol{\phi}_i r_i(\boldsymbol{\phi}_i^\top\boldsymbol{\theta} - y_i) = \boldsymbol{\Phi}^\top\boldsymbol{R}(\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y}),$$

where $\boldsymbol{R}$ is a diagonal matrix with the (i,i)th entry equal to r_i. Setting it equal to zero gives

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^\top \boldsymbol{R} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{R} \boldsymbol{y}.$$

6. (50 points) The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The data set can be downloaded here: `<http://qwone.com/~jason/20Newsgroups/>`. For simplicity, we will just focus on the "bag-of-words" representation of the documents given in the Matlab/Octave section. In this case, the input $\boldsymbol{x}_i$ is a vector of word histogram of doc i, and the output y_i is the news group that it belongs to.

   The data has been divided into a training set and test set. You will build a model from the training set, and evaluate its performance on the test set. The vocabulary size is more than 60,000, which is bigger than the number of documents. (Also, you will find that the `test_data` matrix has more rows than the `train_data` matrix, which means some words in the test data never appear in the training data.) To simplify computation, you will prune the data first by keeping only the words that have appeared more than 1,000 times in the training data (which is approximately 300) and keep only those rows in both the `train_data` matrix and the `test_data` matrix.

   (a) One effective classifer by Tom Mitchell is an instance of the naive Bayes model. First, the actual word count is ignored in the input data; we only consider whether a word j appears in doc i or not. Then each feature in $\boldsymbol{x}$ can be viewed as a Bernoulli random variable. Furthermore, the naive Bayes assumption states that each of these Bernoulli random variables are conditionally independent given the label y, i.e., $p(\boldsymbol{x}|y) = \prod p(x_j|y)$. Each of the $p(x_j|y)$ can be easily estimated using the training data.

   (b) To incorporate the word count, we can impose a rather different probabilistic model. Assume each doc is a huge multinomial random variable, with cardinality equal to the vocabulary size and the total number of draws is the length of that doc, given the label. In other words, $p(\boldsymbol{x}|y)$ is multinomial.

   (c) We can also assume each $p(\boldsymbol{x}|y)$ follows a multivariate normal distribution. Here we assume their covariance matrices are the same, which means the classifer is equivalent to the linear discriminant analysis. Of course, most people don't believe that bag-of-words actually follows a normal distribution, so some pre-processing is used. Do a Google search of TF-IDF and apply that to the data set before training your LDA model.

   (d) Build a least squares classifier using the TF-IDF representation of the data plus a constant 1 as the features. For more than 2 classes the least squares classifier will not be exactly the same as the multi-class LDA, but highly related.

   For each case, derive the mathematical expressions for the corresponding classifiers. Be specific about how to calculate each and every model parameter from data. Pick a language that you like and program these four classifiers using the training set, and report their prediction accuracy on the test set.

   Remark. When calculating the likelihoods for Naive Bayes-Bernoulli and multinomials, the expression $\prod_i p_i^{x_i}$ may give extremely small numbers that could cause underflow. We can overcome this issue by instead calculating $\sum_i x_i \log(p_i)$, which does not change which one is the maximum. However, beware of not taking $\log(0)$.

   Submit your code and make sure they are executable. We may or may not check your code.

   Solution. To make the prediction, you will need $p(y)$ and $p(\boldsymbol{x}|y)$. For all models, $p(y)$ is simply obtained from counting the number of documents of a specific class divided by the total number of docs, i.e.,

$$p(c) = \pi_c = \frac{n_c}{\sum_{j=1}^{k} n_j}.$$

Now we explain how to estimate $p(\boldsymbol{x}|y)$ for each class.

(a) Define the feature representation $\boldsymbol{\phi}_i$ for the ith document as

$$\boldsymbol{\phi}_i(j) = \begin{cases} 1 & \boldsymbol{x}_i(j) \neq 0 \\ 0 & \boldsymbol{x}_i(j) = 0. \end{cases}$$

For this model, we assume that each $\boldsymbol{\phi}_i(j)$ follows a Bernoulli distribution, and the conditional joint completely factors. The probability that term j appears in a doc from class c is $p(\phi_j|c) = p_{cj}^{\phi_j}(1-p_{cj})^{1-\phi_j}$ with the parameter p_{cj}, which can be estimated by

$$p_{cj} = \frac{1}{n_c} \sum_{i \in \text{class } c} \boldsymbol{\phi}_c(j).$$

For a new document represented as a 0-1 vector $\boldsymbol{\phi}$, the probability that it belongs to class c is proportional to

$$\pi_c \prod_{j=1}^{m} p_{cj}^{\phi_j}(1-p_{cj})^{1-\phi_j}.$$

To prevent underflow, we can take the log of that quantity. The resulting classifier takes the following form:

$$\begin{aligned} \widehat{y} &= \arg\max_c \left(\log \pi_c + \sum_{j=1}^{m} \phi_j \log p_{cj} + \sum_{j=1}^{m} (1-\phi_j) \log(1-p_{cj}) \right) \\ &= \arg\max_c (\boldsymbol{w}_c^\top \boldsymbol{\phi} + \beta_c), \end{aligned}$$

where

$$\boldsymbol{w}_c(j) = \log \frac{p_{cj}}{1-p_{cj}}, \qquad \beta_c = \log \pi_c + \sum_{j=1}^{m} \log(1-p_{cj}).$$

The prediction accuracy is 28.42% on the test set.

(b) if we directly model a document as a multinomial distribution for each class, we need to estimate a vector $\boldsymbol{p}_c$ for each class. This can be done as

$$\boldsymbol{p}_c = \frac{1}{\sum_{i \in \text{class } c} \boldsymbol{1}^\top \boldsymbol{x}_i} \sum_{i \in \text{class } c} \boldsymbol{x}_i.$$

The probability that a new document $\boldsymbol{x}$ belongs to class c is proportional to

$$\pi_c \prod_{j=1}^{m} p_{cj}^{x_j},$$

where the term that involve the factorials are the same for all classes, therefore inconsequential in making the predictions. Again, to prevent underflow, we take the log, resulting in the following classifier:

$$\begin{aligned} \widehat{y} &= \arg\max_c \left(\log \pi_c + \sum_{j=1}^{m} x_j \log p_{cj} \right) \\ &= \arg\max_c (\boldsymbol{w}_c^\top \boldsymbol{x} + \beta_c), \end{aligned}$$

where

$$\boldsymbol{w}_c(j) = \log p_{cj}, \qquad \beta_c = \log \pi_c.$$

The prediction accuracy is 39.23% on the test set.

(c) Let's first consider the TF-IDF preprocessing. There is no unique definition, but here's one reasonable choice. Term frequency (TF) is simply taken as $\log(x_{ij}+1)$, so zero remains zero, but bigger values becomes less significant as their raw counts. Inverse document frequency (IDF) is defined for each term over the training set: the total number of documents divided by the number of documents that contains term j, and then we take the log of it. Using the $\boldsymbol{\phi}_i$ vectors that we defined before, then it is $\log(n/\sum_{i=1}^n \phi_i(j))$. As a result, the training data, term j in document i is modified to be

$$\log(x_{ij}+1)\log\frac{n}{\sum_{i=1}^n \phi_i(j)}.$$

We will apply the same normalization to the test sets. The term frequency is again log of the word count plus one. We will use the IDF obtained from the training data. In practice, new docs may come one at a time, and we should use a fixed normalization scheme to keep the model consistent.

Now let's talk about LDA. The probability that a doc belongs to one class is proportional to

$$\pi_c \det(2\pi\boldsymbol{\Sigma})^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_c)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_c).\right)$$

Again we take the log and ignore terms that are common for all classes, and the resulting classifier is

$$\widehat{y} = \arg\max_c \left(\boldsymbol{\mu}_c^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_c^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c + \log\pi_c\right),$$

where

$$\boldsymbol{\mu}_c = \frac{1}{n_c}\sum_{i\in\text{class } c}\boldsymbol{x}_i, \qquad \boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i\boldsymbol{x}_i^\top - \frac{1}{n}\sum_{c=1}^k n_c\boldsymbol{\mu}_c\boldsymbol{\mu}_c^\top.$$

The prediction accuracy is 39.95% on the test set.

(d) This one is straight-forward. The classifier takes the form $\widehat{y} = \arg\max_c(\boldsymbol{w}_c^\top\boldsymbol{x} + \beta_c)$ where the model parameters $\boldsymbol{w}_1,\dots,\boldsymbol{w}_{20}$ and $\beta_1,\dots,\beta_{20}$ are obtained from solving the least squares problem

$$\underset{\boldsymbol{W},\boldsymbol{\beta}}{\text{minimize}} \quad \left\|[\,\boldsymbol{X}\ \ \boldsymbol{1}\,]\begin{bmatrix}\boldsymbol{W}\\ \boldsymbol{\beta}^\top\end{bmatrix} - \boldsymbol{\Psi}\right\|^2,$$

where

$$\boldsymbol{W} = [\,\boldsymbol{w}_1\ \cdots\ \boldsymbol{w}_{20}\,], \quad \boldsymbol{\beta} = [\,\beta_1\ \cdots\ \beta_{20}\,]^\top, \quad \boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1^\top\\ \vdots\\ \boldsymbol{x}_n^\top\end{bmatrix},$$

and $\boldsymbol{\Psi}\in\mathbb{R}^{n\times 20}$ is defined as

$$\Psi_{ic} = \begin{cases}1, & c = y_i,\\ 0, & c\neq y_i.\end{cases}$$

The prediction accuracy is 41.73% on the test set.

Notice that all three classifiers have a linear form $\arg\max_c(\boldsymbol{w}_c^\top\boldsymbol{x}+\beta_c)$.

The accuracies are not particularly impressive. However, considering there are 20 categories to predict, they are still significantly higher than a random guess of about 5% accuracy.

We eliminated words that don't appear very often in the data set, mainly for more efficient computation. Intuitively, those less-commonly used words may be more indicative in the classification task. Indeed, if we keep all the words that have appeared in the training set, the models in part (a) and (b) could give about 75% accuracy. There will be computational issues in part (c) and (d), on the other hand. We will discuss this in future lectures.