

CAP 6610 Machine Learning, Spring 2025

Homework 2

Due Feb. 20, 11:59pm

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 \mathbf{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.

1. 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, $p(\mathbf{x}|y)$ is multinomial; for each class c , the parameter for the multinomial distribution is a nonnegative vector \mathbf{p}_c that sums to one. Describe how to estimate \mathbf{p}_c . Write the resulting classifier in the form $\hat{y} = \max_c (\mathbf{w}_c^\top \mathbf{x} + \beta_c)$. Explain how to obtain \mathbf{w}_c and β_c from $p(y)$ and $p(\mathbf{x}|y)$.

Remark. When calculating the likelihoods for 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 $\log(0)$.

2. We can also assume each $p(\mathbf{x}|y)$ follows a multivariate normal distribution. Here we assume their covariance matrices are the same, which means the classifier 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.

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 two classifiers using the training set, and report their prediction accuracy on the test set.

Now let us apply some of the unsupervised methods by ignoring their labels. We will only consider the training data. You are required to code the algorithms by yourselves in the language of your choice.

3. *LSI/PCA via orthogonal iteration.* Implement the orthogonal iteration algorithm that finds the PCA projection matrix Θ of a data matrix Φ . You are allowed to use a pre-existing function of QR. Apply tf-idf to the term-document matrix to obtain Φ and feed it into your orthogonal iteration algorithm. Remember to use sparse matrix operations to avoid unnecessary memory/computational complexities. Set $k = 2$ and let the algorithm run until Θ doesn't change much. Then get $\mathbf{Y} = \Theta^\top \Phi$. Each column of \mathbf{Y} is a two-dimensional vector that you can plot on a plain. Plot all the documents on a two-dimensional plain, and use a different color for each point that belong to different news groups.

4. *GMM via EM*. Implement the EM algorithm for the Gaussian mixture model (with different means and covariances for each Gaussian component). The data matrix Φ is obtained from LSI with $k_{\text{LSI}} = 100$ using the previous orthogonal iteration algorithm. Run the EM algorithm for GMM with $k_{\text{GMM}} = 20$ until convergence. For each Gaussian component $\mathcal{N}(\mu_c, \Sigma_c)$, calculate $\Theta \mu_c$ where Θ is the PCA projection; the vector $\Theta \mu_c$ should be element-wise nonnegative. For each cluster c , show the 10 terms that have the highest value in $\Theta \mu_c$. The index-term mapping can be found here <http://qwone.com/~jason/20Newsgroups/vocabulary.txt>. Does the result make sense?