

CS 294 - Practical Machine Learning - Homework 3

October 17, 2006

Part I: clustering

Preliminaries This part of the assignment will be done entirely in R. You will need the following packages: `stats`, `mclust`, `MASS`

Problem 1: K -means In this problem, you are asked to do some K -means experiments on simulated data and deal with some practical issues, such as how to choose K and whether to do standardization or not before running the algorithm.

- (a) Simulate 2-dimensional data from four clusters, each of which is specified by a Gaussian distribution with the same covariance matrix and different means. In function `generateFourClusters`, fill out the code to plot K (ranging from 2 to 6) against within cluster sum of squares (as in slide 31). Choose an appropriate K from the plot and argue why do you choose this particular K . Run function `Kmeans` with chosen K and plot the clustering result. Write down how many points are in wrong clusters. Don't forget to record the sample covariance matrix of the simulated data at this point.
- (b) In slide 11, you have already seen a formula that tells you the average dissimilarity of j th coordinate is twice the sample estimate of the variance of X_j . Here the sample estimate var_j is defined as $var_j = \frac{1}{n} \sum_{i=1}^n (x_{ij} - \bar{x}_j)^2$ where $\bar{x}_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$. Derive that formula.
- (c) Setting $w_j \propto 1/\bar{d}_j$ is equivalent to standardizing the data. Standardization means making the data centered and looks spherical (identity matrix as its sample covariance matrix). In function `standardization`, fill out the code to standardize the data. Now plot K against within cluster sum of squares on standardized data, choose an appropriate K from the plot. Is the K you choose the same as the one you chose in part (a)? Run function `Kmeans` with two chosen K s (if they differ) on **standardized** data. Compare the clustering results with the one you obtained in part (a).
- (d) Explain why standardization in this setting does not work.

Problem 2: GMM

- (a) Run `Mclust` function on the simulated data from problem 1. Notice that in the implementation of `Mclust`, you don't need to specify the number of clusters beforehand. Instead, there are two parameters `minG` and `maxG` specifying the minimum and maximum number of mixture components to be considered. It will then choose the best K that optimizes a criterion named BIC. Try to change the parameters of simulation (esp. common covariance matrix) and see the changes of results (number of clusters and clustering of points) it makes.
- (b) **(optional)** As mentioned in the lecture, there is a strong connection between K -means and GMM. Consider GMM with common spherical covariance matrix $\Sigma = \sigma^2 I$. Show that as $\sigma \rightarrow 0$, the responsibilities r_{ik} for each data point i in E step goes to 1 for one particular k and 0 for all other k s. In this sense, GMM is equivalent to K -means.

Problem 3: Hierarchical Clustering

- (a) Find a simple artificial example that single linkage hierarchical clustering produces extended clusters while complete linkage yields round clusters.
- (b) Download dataset "election.data" from course website. This is the dataset of presidential vote by county in California (2004). We'd like to cluster counties such that counties with similar voting behavior should be more likely to in the same cluster. Run `hclust` on the data using average linkage. Explain why Los Angeles is so different from other counties and why the dendrogram looks like one using single linkage.
- (c) Now we normalize the two columns to avoid population influence. Fill out the code in `electionClustering` (there are only two lines). Read other code in that function and explain the final dendrogram. Try single linkage and complete linkage as well.

Part II: dimensionality reduction

Preliminaries This part of of the assignment will be done entirely in R. You will need the following packages: `fastICA`, `e1071`, `tuneR`, and `pixmap`.

Problem 1: PCA warmup The purpose of this part is to get acquainted with using R for doing linear algebra operations and getting used to thinking about the geometric structure of data. You'll see that PCA can be useful for visualizing high-dimensional data in this toy example.

- (a) The file `warmup.dat` contains 3421 20-dimensional points, one per line. Load them up in R and run PCA on the data using SVD. You should have to write no more than 10 lines of code. Look at the singular values. How many dimensions does this data truly lie in? Plot the data in those dimensions and read off the message. You might have to stretch or flip your plot to see it.

- (b) Did you center your data first? Try both centering it and not centering it. Explain the difference between the principal components obtained in each case. Which set of principal components is more representative of the underlying structure of the data?
- (c) Now center your data. Try changing the scale of the dataset by arbitrarily picking one of the 20 dimensions and multiplying it by say, 1000. How does this operation change the top principal components, qualitatively and quantitatively? If you did not notice a large qualitative change, describe briefly what kind of transformation would cause such a change.

Problem 2: PCA and generalization on a face detection task Besides visualization, a very important use of PCA is as a preprocessing step of obtaining features and using them in a following phase. In this problem, we will use PCA to generate a set of features to use for face detection. We'll see why performing PCA is useful in terms of generalization to the test set.

First, call the function `loadAllFaces()` to read in both the training and test images. The images will be stored in `trainPos`, `trainNeg`, `testPos`, and `testNeg`, which correspond to the positive and negative examples for both training and test sets. You can draw a face by calling `plotImg(trainPos[,4])`, for example.

- (a) Before getting into classification, run PCA on the positive training examples (`trainPos`). Take one of these images vectors x . Project x onto the first K principal components, and then try to reconstruct the image given the projection. You can call `plotImg()` to see what the reconstruction x' looks like and also compute the reconstruction error $\|x - x'\|^2$ (squared distance between the vectors). Start with $K = 1$ and incrementally increase K . Note both how quickly (or slowly) the reconstructed image converges back to the original image, both qualitatively and quantitatively. Does the error eventually reach 0?
- (b) Now, pick a positive *test* example and do the same as in (a). Does the reconstruction require more or fewer principal components to reach the same fixed error threshold? Does the error eventually reach 0?
- (c) Now let's turn to classification. Suppose we are trying to build a classifier to detect face versus non-face. Without PCA, we can train a simple SVM classifier just using the pixel intensities as the features. The current code right now does that: `augmentFeatures()` just returns the same pixel feature representation of the images. Modify this code along with `computePCA()` so that the SVM classifier will be able to use the K principal component features. `evaluateFeatures()` takes your `augmentFeatures()` as an argument and returns the training and test error. Starting with $K = 1$, increase K and note how the training and test error change. Optional: repeat the above experiment, but this time including the PCA features *in addition* to the original pixel features.
- (d) PCA only makes use of the positive training examples to create features. How might LDA be more suitable for extracting features on this task?

- (e) Comment on the bag-of-pixels representation that we have been using. Suppose that the faces were *not* centered, but rather were positioned at different offsets in different images. How might PCA be adversely affected by this?

Problem 3: CCA Recall that CCA requires two views of every data point.

- (a) What are the pair of subspaces recovered by CCA if the two views are entirely independent of each other? Note that independence implies uncorrelatedness.
- (b) What are the pair of subspaces recovered by CCA if the two views are entirely dependent (say, identically equal $x_i = y_i$)?
- (c) Are these solutions unique?

Fortunately, most of the time, data is somewhere in between the two extremes.

Problem 4: ICA Recall that ICA finds a linear transformation of the data which makes the resulting components most independent.

- (a) In `data/sounds` are 4 wav files, which contain 4 signals that have been mixed using different coefficients. Complete the code in `p4()`, which loads these 4 files, performs ICA to separate out the original sources. You should be able to mostly separate the four sounds. Listen to them to check your output. Note that using PCA to separate the sources does not work.
- (b) Explain why ICA would fail if two of the original sources were exactly the same. When else would ICA fail?

Problem 5 (optional): This problem is for those who want to think more about the math behind these dimensionality reduction techniques.

- (a) Show that the eigenvalues of a symmetric matrix XX^T are the squares of the singular values of X .
- (b) Show that the data projected onto the principal components are uncorrelated.