EE 5531 Probability and Stochastic Processes
Fall Semester 2015

Monday October 12, 2015

Miniproject 1 — Due Monday November 2, 2015 at 12:05pm (after class)

Part 1: Low-dimensional Structure in High-dimensional Data

Background

This week we will discuss the multivariate Gaussian distribution, and we will specify an expression for the density of a Gaussian random vector that is valid when the covariance matrix is full-rank, and hence invertible. (These notions are ubiquitous in classical statistical signal processing problems and analyses!)

In practice, however, we often wish to expand our interpretation of the multivariate Gaussian distribution to cases where the covariance matrix is *rank-deficient*, or is well-approximated by a covariance matrix that is rank-deficient (for good reason, as you'll see in part 2). This part of the miniproject will help you visualize and understand this important case.

Tasks

For this part of the miniproject, you need to download the `MATLAB` script `'gauss_scatter.m'` from the course website. This script generates several thousand realizations of a 3-dimensional zero-mean Gaussian random vector drawn from a distribution having covariance matrix of rank r (where $r = 1, 2$, or 3), and allows you to visualize the resulting "point clouds" in 3D space. Experiment with the script using different values of the rank parameter r. To help aid your visualization, use the *Rotate 3D* feature of the MATLAB plotter to view the point clouds from different orientations.

Now, do the following:

1. **Submit** one representative scatter plot for each of the three cases: $r = 1$, $r = 2$, and $r = 3$, for the default case in the `MATLAB` script, where each of the (nonzero) eigenvalues of the covariance matrices are equal to one.

2. In the above experiments, what structure do you notice in the cases $r = 1$ and $r = 2$? More specifically, in each case the data vectors are all points in $\mathbb{R}^3$ (by construction), but what can you say about the *intrinsic* structure or *effective* dimension of the data? You may need to rotate the point clouds to see this structure. **Submit** a written explanation of your observations.

3. Now, experiment with the case where the (nonzero) eigenvalues of the covariance matrix are not all equal to one, but rather are generated randomly (as realizations of a uniform$[0, 1]$ continuous random variable). The comments in the `MATLAB` script provided explain how to make this change. How do these unequal eigenvalues affect the shape of the point cloud (especially when $r = 2$ and $r = 3$)? **Submit** a written explanation of your observations.

4. Finally, modify the `MATLAB` code to simulate a case where the covariance matrix has rank 2, where the largest nonzero eigenvalue is equal to 1, and the other nonzero eigenvalue is equal to 0.01. Even though the covariance matrix has rank 2, what can you say about the *approximate* or *effective* dimension of the data in this case? **Submit** a representative plot of the data in this case, along with a written explanation of your observations.

Part 2: Principal Component Analysis (PCA)

Background

Suppose that we wish to approximate a vector $x \in \mathbb{R}^n$ by another vector $\widehat{x} \in \mathbb{R}^n$ that is constrained to reside in some linear subspace of estimators of dimension r. Let U be an $n \times r$ matrix with orthonormal columns u_i, for $i = 1, 2, \ldots, r$, and let us denote by $\mathcal{X}$ the class of estimators formed as linear combinations of the columns of U. That is

$$\mathcal{X} = \left\{ v \in \mathbb{R}^n \ : \ v = \sum_{i=1}^{r} a_i u_i, \ a_i \in \mathbb{R}, \ i = 1, 2, \ldots, r \right\}.$$

Following our discussion in class, we may seek to choose our approximation to be the vector in our class of estimators having minimum *Euclidean distance* to x. That is,

$$\begin{aligned} \widehat{x} &= \arg\min_{v \in \mathcal{X}} \|x - v\|^2 \\ &= \arg\min_{a \in \mathbb{R}^r} \|x - Ua\|^2, \end{aligned}$$

where in the last line the vector $a \in \mathbb{R}^r$ is the vector of coefficients of the columns of U in the approximation. In this case, it is easy to show (e.g., using vector calculus, or the *orthogonality principle!*) that the best estimator is obtained by the orthogonal projection of x into the space of estimators $\mathcal{X}$, denoted by $\widehat{x} = \mathcal{P}_{\mathcal{X}}(x)$, where

$$\mathcal{P}_{\mathcal{X}}(x) \triangleq U(U^T U)^{-1} U^T x = U U^T x.$$

The last equality follows from the assumption that the columns of U are orthonormal.

The virtue of this kind of approximation is that only r *degrees of freedom* are necessary to describe the (hopefully accurate!) approximation of x, once an "appropriate subspace" is identified. But how should we identify an appropriate subspace?

This is where Principal Component Analysis (PCA) comes into play. PCA is one of the most widely used statistical data analysis techniques to date [1, 2]. The PCA idea, at a high level, is to represent (ostensibly) high-dimensional data in a succinct way by projecting it into a lower dimensional subspace of estimators. Further, the PCA approach effectively *learns* an appropriate subspace for representing the data *from the data itself.*

Suppose that we have access to a set of *training data*, where each data point in the training data set is a vector $z_i \in \mathbb{R}^n$, for $i = 1, 2, \ldots, p$, implying a total of p training data points. One way to think of the PCA idea is to interpret the data points as iid *random vectors* drawn from the same distribution, which is characterized as having (initially, unknown) mean vector m_Z and covariance matrix C_Z. Here, the covariance matrix C_Z may be rank deficient, or approximately so...in this case, we can approximate new data vectors $x \in \mathbb{R}^n$ (that are NOT in the training set) by their projections into a low-dimensional subspace described in terms of the span of the subset of eigenvectors of C_Z corresponding to a subset of the largest eigenvalues.

Of course, in practice we won't know m_Z or C_Z exactly, so instead we can attempt to estimate them from the data. In particular, let $\widetilde{m}_Z \in \mathbb{R}^n$ denote the empirical mean, or average, of the vectors in the training set, and let us form the empirical estimate of the covariance matrix:

$$\widetilde{C}_Z = \frac{1}{p-1} \sum_{i=1}^{p} (z_i - \widetilde{m}_Z)(z_i - \widetilde{m}_Z)^T.$$

We can perform an eigen-decomposition of the empirical covariance matrix $\widetilde{C}_Z$, so that

$$\widetilde{C}_Z = V \Sigma V^T,$$

where V is an $n \times n$ unitary matrix, and Σ is a diagonal matrix of eigenvalues with non-negative entries along the diagonal (the diagonal entries of Σ are *eigenvalues* associated with the eigenvectors that are the corresponding columns of V).

Now, we let U be an $n \times r$ matrix (with $r < p$) whose columns are the columns of V corresponding to the largest eigenvalues along the diagonal of Σ. These r column vectors are called *principal component vectors*, and are interpreted as vectors that span the linear subspace in which most of the variation (or variance) of the data is exhibited. Note that this definition implies that, for $r = 1$, the first principal component direction is the direction of maximum variation of the *mean-centered* data (or, the direction of maximum variance), the second principal component direction is orthogonal to the first, and captures the direction of the next most variation, and so on. This interpretation gives us a principled way to identify a (low-dimensional) subspace of estimators, and we can identify the best estimator from that subspace using the orthogonal projection approach described above.

The only remaining caveat in this approach is the following. Recall that we subtracted the empirical mean from each of the vectors in the training set in order to compute the empirical covariance matrix. This was done so that the principal components correspond to directions of maximum variance. Note that we could apply eigendecomposition to the empirical *correlation* matrix instead (i.e., don't mean-center the training data vectors), but in this case the first principal component can be biased by a large non-zero mean in the training data. At any rate, when forming our final estimate using the approach described above, we need to undo the mean-centering by *adding the empirical mean* to our estimate obtained by projecting into the low-dimensional subspace spanned by the principal components.

How good is this approach of *subspace identification using PCA*, followed by *subspace approximation*, on real data? This is what you'll investigate in this part of the mini project.

Tasks

For this part of the miniproject, you need to download the database of faces available (for free) online at `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`. Specifically, following the link `http://www.cl.cam.ac.uk/research/dtg/attarchive/pub/data/` will take you directly to the download site where you can find the file '`att_faces.zip`', a zip file of 400 face images. The data corresponds to 10 different faces for each of 40 people.

Download the data, and unpack it into its own folder. Now, download the `MATLAB` script '`reader.m`' from the course website, and place it in the root folder of the unpacked data from above. This script will effectively parse and reshape the data into two matrices, which are then saved to the workspace. The first matrix is the training data set – 9 of the 10 images from each person are obtained, each is reshaped into a vector (by stacking columns of the corresponding image), and these columns are assembled into a matrix Z having 360 columns. The second matrix is the *test data* – a matrix T which corresponds to (similarly vectorized) versions of the single remaining image from each person (for a total of 40 columns).

For this part of the miniproject, you are to do the following:

1. Read section 8.5 in the textbook, which describes estimation of covariance matrices and provides some `MATLAB` code for implementation.

2. Write your own code to extract the r principal components from a given training data set. Your code should take as an input an $n \times p$ training data matrix Z and return a matrix, say U, having n rows and $r < p$ orthonormal columns.

   Hint: Rather than compute the empirical covariance matrix and do eigen-decomposition on it directly (which can be computationally intensive on MATLAB), you can instead find U using the singular value decomposition*, or SVD, as follows. Let Zc be the $n \times p$ "centered" matrix obtained by subtracting the mean of the columns* ***(i.e., the sum of the columns, divided by the number of columns)*** *of Z from each column of Z. Then, the MATLAB command* `'[A,B,C] = svd(Zc/sqrt(p-1));'` *returns matrices A, B, and C, such that $Zc/\sqrt{p-1} = ABC^T$, with A and C each square and unitary, and B diagonal with non-negative entries on the diagonal arranged from largest to smallest. The columns of A are exactly the eigenvectors of $ZcZc^T/(p-1)$, which is just the empirical covariance matrix, while the diagonal entries of B are called singular values and are just the positive square roots of the eigenvalues associated with the corresponding columns of A. Thus, the first r principal components are just the first r columns of A obtained using the SVD.*

   Feel free to experiment with other built-in `MATLAB` *functions to compute the eigenvectors more efficiently...such as* `eigs`*. You can always use the* `MATLAB` *command* `help`*, as in* `help eigs` *to learn the syntax and functionality of a* `MATLAB` *function.*

3. Consider the first two images in the test set (corresponding to the first two columns of T). Compute approximations of these test images obtained by first subtracting from each the empirical mean of the training set, projecting each onto the subspaces spanned by the first $r = 0, 1, 25, 50$, and 100 principal components of the empirical covariance matrix **of the training data (from part 2 above)**, then adding back to each the empirical mean **of the training data** (as discussed above). (Note – the projection onto the subspace spanned by zero principal components is null. The approximation in this case just corresponds to approximating the data using the empirical mean!) **Submit** figures showing the original images along with their 5 approximations, **as well as a written discussion** of your observations on how the quality of the approximation varies as r increases. You may find the MATLAB commands `'reshape'`, `'subplot'`, and `'imagesc'` useful when creating the figures. You may also want to force the images to be displayed in grayscale using the command `'colormap gray'`. (Feel free to experiment with other images from the test set! You don't need to submit the approximation results for any others besides the first two, though.)

4. We can be a bit more principled about *how* we identify the appropriate subspace dimension. Recall the SVD described above in step 2. Use the MATLAB command `'b = diag(B);'` to extract the main diagonal of the matrix B of singular values. Now, we can plot the (sorted) eigenvalues, arranged from largest to smallest, by plotting the squares of the entries of b. Note that this should be a vector of length 360. Qualitatively speaking, do you notice that the number of "relevant" eigenvalues is small relative to the total? **Submit** this plot, along with a written discussion describing what you believe is the approximate effective dimension of the data (use what you learned in Part 1 of the project!).

Part 3: Experimentation!

Background

Parts 1 and 2 above!

Tasks

For this part, you will apply the PCA approach to a *different* data set of your choosing. See, for example, the repository `http://www.cs.cmu.edu/~cil/v-images.html`, or use any other data set you wish (e.g., any data sets from the US Government's open data resource `'data.gov'` or the 'Data Science' challenges website `'kaggle.com'`). Indeed, the PCA approach is applicable to many diverse forms of data (e.g., speech or audio data segments, video frames, network traffic data, etc.).

Submit the plot of eigenvalues of the training data for your data set, and explain what you believe is an appropriate effective dimension (analogously to task 4 in Part 2 above). Also, **submit** a few plots showing approximations of some test data (points that must be omitted from the training set!) using the principal components you learned from your training data (analogously to task 3 in Part 2 above).

References

[1] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[2] I. Jolliffe, *Principal component analysis*, Wiley Online Library, 2005.