ANU Machine Learning Summer School 2010 Handout for Lab 4: Principal Component Analysis

Daniel Visentin

School of Computer Science
College of Engineering and Computer Science
The Australian National University
27^{th} September 2010

Abstract. In this lab we learn about principal component analysis and apply it to compress a set of images.

1 Introduction

Principal component analysis (PCA) [1–3] is a technique for projecting a data set onto a lower-dimensional subspace (see Figure 1 for an example). Intuitively, it works by projecting the data such that most of the "information" is contained in the first few dimensions of the projection. This makes it suitable for a number of applications:

- Dimensionality reduction: If most of the important information is contained in the first few dimensions of the projected data, the remaining dimensions can often be safely discarded without significant detriment to subsequent analysis. Furthermore, this reduction in dimension will often result in accelerated performance.
- Lossy data compression: By performing dimensionality reduction on a data set, we can represent each data point by a smaller number of components. Applying the inverse projection to the projected data gives an approximation of the original data. Since most of the information is conserved while the dimensionality is reduced, this can be used as a compression scheme.
- Feature extraction: The projection may elucidate some structures/features of the data set which are more useful or natural than it's original representation.
- Data visualisation: It is often difficult to visualise high-dimensional data sets. Since most of the information is contained in the initial dimensions of the projected data, visualising the projected data might still offer some insights into the original data set.

2 The Algorithm

Principal component analysis involves computing the orthogonal projection of a data set onto a space of dimensionality M such that the variance of the projected

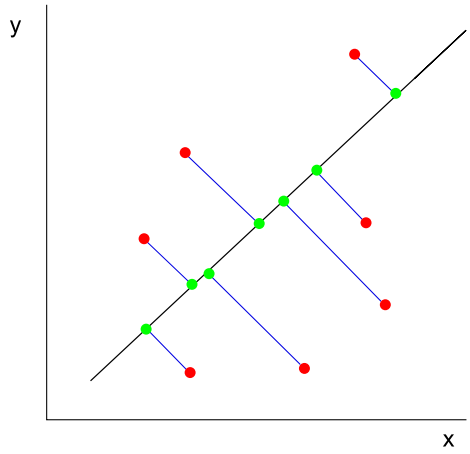

Fig. 1. Example of PCA – here we have projected the red points onto the central line, resulting in the green points. So we have reduced the two dimensional data (red) set to a one dimensional (the line) data set (green). The line was chosen by minimizing the sum of squared projection errors, which is the length of the blue lines.

data is maximised. This is equivalent to computing the orthogonal projection that minimises the average projection cost (squared distance between the data points and their projections). By solving either of these formulations we find that this projection is defined by the M eigenvectors of the data covariance matrix $\boldsymbol{S}$ corresponding to the M largest eigenvalues of $\boldsymbol{S}$. We refer to these eigenvectors as the *principal components* and the space they span as the *principal subspace*.

2.1 Deriving the Principal Components

Let $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ be a data set of D-dimensional column vectors. Let $\boldsymbol{X}$ denote the $D \times N$ data matrix whose i^{th} column is given by $\boldsymbol{x}_i$. Our goal is to project the data onto an M-dimensional space in a way that maximises the projected variance.

We proceed iteratively, beginning with $M = 1$. In this case, the principal subspace is defined by a single D-dimensional unit vector $\boldsymbol{u}_1$. A data point $\boldsymbol{x}_i$ is projected onto the principal subspace by taking the dot product $\boldsymbol{u}_1^T \boldsymbol{x}_i$. We choose $\boldsymbol{u}_1$ which maximises the variance of the projected data

$$\boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{u}_1^T \boldsymbol{x}_i - \boldsymbol{u}_1^T \overline{\boldsymbol{x}})^2 \tag{1}$$

where $\overline{\boldsymbol{x}}$ is the mean vector

$$\overline{\boldsymbol{x}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i,$$

$\boldsymbol{S}$ is the $D \times D$ data covariance matrix

$$\boldsymbol{S} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T = \frac{1}{N}\left(\boldsymbol{X} - \overline{\boldsymbol{X}}\right)\left(\boldsymbol{X} - \overline{\boldsymbol{X}}\right)^T,$$

and $\overline{\boldsymbol{X}}$ is the $D \times N$ matrix whose columns are $\overline{\boldsymbol{x}}$. Since $\boldsymbol{u}_1^T\boldsymbol{S}\boldsymbol{u}_1 \to \infty$ as $\|\boldsymbol{u}_1\| \to \infty$, we constrain $\boldsymbol{u}_1$ to be a unit vector $\|\boldsymbol{u}_1\| = 1$. Introducing the Largrange multiplier λ_1 and maximising gives the result [3]

$$\boldsymbol{S}\boldsymbol{u}_1 = \lambda_1\boldsymbol{u}_1.$$

So $\boldsymbol{u}_1$ must be an eigenvector of the data covariance $\boldsymbol{S}$ and the projected covariance $\boldsymbol{u}_1^T\boldsymbol{S}\boldsymbol{u}_1$ is maximised when $\boldsymbol{u}_1$ is the eigenvector with the largest eigenvalue λ_1.

For $M = 2$, we can proceed by trying to find a $\boldsymbol{u}_2$ that minimizes the projected variance in the direction orthogonal to $\boldsymbol{u}_1$. Generalizing this for any $M \leq D$, we find that the required $\boldsymbol{u}_i$'s are the M eigenvectors of $\boldsymbol{S}$ corresponding to the M largest eigenvalues λ_i (sorted in decreasing order). These eigenvectors are called the *principle components*. For future convenience, we let $\boldsymbol{U}$ be the $D \times M$ matrix whose i^{th} column is given by the i^{th} principal component $\boldsymbol{u}_i$.

2.2 Projection into the principal subspace

The projection of the data point $\boldsymbol{x}_i$ into the principal subspace is the M-dimensional vector $\boldsymbol{y}_i$ defined by

$$\boldsymbol{y}_i := (\boldsymbol{u}_1^T\boldsymbol{x}_i, \boldsymbol{u}_2^T\boldsymbol{x}_i, \ldots, \boldsymbol{u}_M^T\boldsymbol{x}_i)^T = \boldsymbol{U}^T\boldsymbol{x}_i.$$

In words, the k^{th} component of the projection is given by the dot product of the k^{th} principal component $\boldsymbol{u}_k$ with the data point $\boldsymbol{x}_i$. We can extend this to calculate the projection $\boldsymbol{Y}$ of the entire data set in one matrix operation

$$\boldsymbol{Y} = \boldsymbol{U}^T\boldsymbol{X}$$

where the i^{th} column of $\boldsymbol{Y}$ corresponds to $\boldsymbol{y}_i$.

2.3 PCA approximation

We can use PCA as a compression technique by projecting a data set into the principal subspace and storing the mean, the projections, and the principal components. We can then "decompress" these stored values by performing the inverse transformation back into the original space. A suitable choice of the principal subspace dimension M will result in a satisfactory reproduction of the original data and potentially save on space. In particular, we will need to store

$$\underbrace{D}_{\text{mean}} + \underbrace{NM}_{\text{projections}} + \underbrace{MD}_{\text{principal components}} = M(N+D) + D$$

numbers as opposed to ND numbers for the original data set. For M much less than N and large D this can result in a substantial saving.

The PCA approximation $\tilde{\boldsymbol{x}}_i$ of the data point $\boldsymbol{x}_i$ is given by

$$\begin{aligned}\tilde{\boldsymbol{x}}_i &:= \sum_{j=1}^{M} \boldsymbol{u}_j(\boldsymbol{u}_j^T\boldsymbol{x}) + \sum_{j=M+1}^{D} \boldsymbol{u}_j(\boldsymbol{u}_j^T\overline{\boldsymbol{x}}) \\ &= \overline{\boldsymbol{x}} + \sum_{j=1}^{M} \boldsymbol{u}_j(\boldsymbol{u}_j^T(\boldsymbol{x}_i - \overline{\boldsymbol{x}})) \\ &= \overline{\boldsymbol{x}} + \left(\sum_{j=1}^{M} \boldsymbol{u}_j\boldsymbol{u}_j^T\right)(\boldsymbol{x}_i - \overline{\boldsymbol{x}}) \\ &= \overline{\boldsymbol{x}} + \boldsymbol{U}\boldsymbol{U}^T(\boldsymbol{x}_i - \overline{\boldsymbol{x}})\end{aligned}$$

which is equivalent to

$$\tilde{\boldsymbol{x}}_i = \overline{\boldsymbol{x}} + \boldsymbol{U}\boldsymbol{y}_i$$

where we define $\boldsymbol{y}_i := \boldsymbol{U}^T(\boldsymbol{x}_i - \overline{\boldsymbol{x}})$ to be the projection into the principal subspace of the mean-centered data point $\boldsymbol{x}_i - \overline{\boldsymbol{x}}$. The approximation of the entire data set can be computed in one step as

$$\widetilde{\boldsymbol{X}} := \overline{\boldsymbol{X}} + \boldsymbol{U}\boldsymbol{U}^T\left(\boldsymbol{X} - \overline{\boldsymbol{X}}\right)$$

where the i^{th} column of $\widetilde{\boldsymbol{X}}$ corresponds to $\tilde{\boldsymbol{x}}_i$.

2.4 PCA for high-dimensional data

In order to use PCA we need to compute the eigenvectors of the $D \times D$ data covariance matrix $\boldsymbol{S}$. For high-dimensional data sets this is often unfeasibly slow. As an example, consider a data set of 100×100 images. Then we need to find the eigenvectors of a $10,000 \times 10,000$ matrix! However it is often the case in such circumstances that we have a much smaller number of data points N than the data dimension D. We can exploit this by letting $\widehat{\boldsymbol{X}} := \boldsymbol{X} - \overline{\boldsymbol{X}}$ and noting that since the eigenvectors $\boldsymbol{u}_i$ satisfy

$$\frac{1}{N}\boldsymbol{S}\boldsymbol{u}_i = \frac{1}{N}\widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}^T\boldsymbol{u}_i = \lambda_i\boldsymbol{u}_i$$

they must also satisfy

$$\frac{1}{N}\widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{X}}\widehat{\boldsymbol{X}}^T\boldsymbol{u}_i = \lambda_i\widehat{\boldsymbol{X}}^T\boldsymbol{u}_i.$$

So, by finding the eigenvectors $\boldsymbol{v}_i = \widehat{\boldsymbol{X}}^T\boldsymbol{u}_i$ of the $N \times N$ matrix

$$\frac{1}{N}\widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{X}}$$

we can retrieve the principal components using

$$\boldsymbol{u}_i = \frac{1}{(N\lambda_i)^{1/2}}\widehat{\boldsymbol{X}}\boldsymbol{v}_i.$$

3 Examples

3.1 Eigenfaces

In this example we have access to a data set composed of 400 images of faces. Each image is normalised to 70×100 pixels, then flattened and stored as as a 7000-dimensional vector. By applying PCA to this data set we can derive a set of "eigenfaces" by interpreting the principal components as images (figure 2). We can also use PCA approximation to compress and reconstruct the faces as seen in figure 3. Because of the high-dimensionality of the data, it is necessary to use the formulation in section 2.4.

Fig. 2. First 10 eigenfaces.

Fig. 3. Compression of a face for $M = 25, 50, 100, 200, 300, 400$.

4 Navigating the Code

The load_all_faces function loads each image from the faces directory into a 2-dimensional numpy array where each column corresponds to an image. The code then calls the PCA function to compute the principal components of the data set. In order to compute the eigenvectors and eigenvalues we can use either of

the numpy functions numpy.linalg.eig or numpy.linalg.svd. The documentation for these functions can be found at:

http://docs.scipy.org/doc/numpy/reference/routines.linalg.html

It is recommended you use the singular value decomposition (SVD) function for this purpose since it returns as its first value a matrix of eigenvectors sorted by decreasing eigenvalue (Note: the first return value corresponds to the eigenvectors because $\boldsymbol{S}$ and $\widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{X}}$ are real symmetric matrices–this is not true in general). Whereas if you do use the numpy.linalg.eig function you will have to sort the eigenvectors yourself. The code then saves the eigenvectors as images into the eigenfaces directory. Finally, the code computes the PCA approximations for varying subspace dimensions and stores the results in the compressed directory.

References

1. Smith, L.: A tutorial on principal components analysis. Cornell University, USA **51** (2002) 52
2. Jolliffe, I.: Principal component analysis. Springer verlag (2002)
3. Bishop, C., et al.: Pattern recognition and machine learning. Springer New York (2006)