10-601 Machine Learning (Fall 2010)
Principal Component Analysis

Yang Xu

This note is partly based on Chapter 12.1 in Chris Bishop's book on PRML and the lecture slides on PCA written by Carlos Guestrin in the 10-701 Machine Learning (fall 2009) course.

1 In a nutshell

Principal component analysis or PCA, in essence, is a linear projection operator that maps a variable of interest to a new coordinate frame where the axes represent maximal variability. Expressed mathematically, PCA transforms an input data matrix $\mathbf{X}$ ($N \times D$, N being the number of points, D being the dimension of data) to an output $\mathbf{Y}$ ($N \times D'$, often $D' \leq D$) via the following

$$\mathbf{Y} = \mathbf{X}\mathbf{P} \tag{1}$$

where $\mathbf{P}$ ($D \times D'$) is the projection matrix of which each column is a principal component (PC)—these are unit vectors that bear orthogonal directions. PCA is a handy tool for dimension reduction, latent concept discovery, data visualization and compression, or data preprocessing in general.

2 Why do we care

One could think of many reasons where transforming a data set at hand to a low-dimensional space might be desirable, e.g. it makes the data easier to manipulate with and requires less computational resource. It is, however, important to perform such transformations in a principled way because any kind of dimension reduction might lead to loss of information, and it is crucial that the algorithm preserves the useful part of the data while discarding the noise. Here we motivate PCA from three perspectives and explain why preserving maximal variability makes sense.

2.1 Correlation and redundancy

Suppose you did a survey on height and weight of your classmates and learnt that these were roughly correlated in that taller individuals tend to be heavier

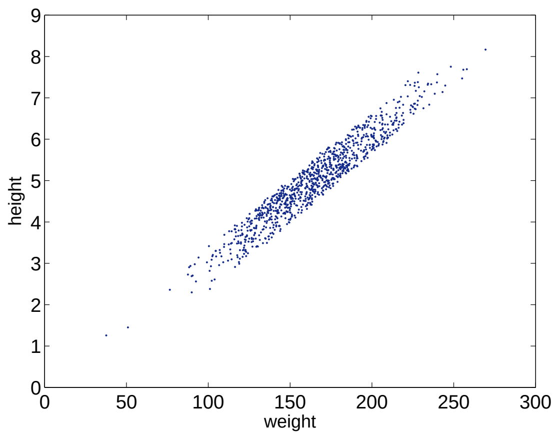

Figure 1: Height vs weight.

and vice versa. You decided to plot these on a graph where each individual is represented in the coordinates (height,weight) as illustrated in Figure 1. Immediately you would find that by tilting these two axes approximately 45 degrees you could capture most of the variability along a single axis. In fact, if the heights and weights were perfectly correlated (e.g. they form a line) you could discard one of the two tilted axes while still capturing the full distribution. In other words, if an algorithm finds a rotation angle of the axes such that maximal variability is preserved, it may help one figure out where correlation lies and which axes to drop, removing redundancies in the data—PCA does exactly this, and more so, it tells you how much variability the rotated axes tend to capture.

2.2 Synergy

Correlations also imply synergies. When synergy exists among a mass of dimensions, dimension reduction becomes extremely efficient—one could represent the mass of dimensions with just a handful. For example, imagine 20 violinists performing in unison in an orchestra. Assuming only a few yawns or does something completely offbeat, the movement of the ensemble 20 violinists could well be characterized as a synergic whole (i.e. a single dimension suffices to represent 20 dimensions). In other words, synergic performing hands (motions) dominate the representation, whereas noise factors such as yawning can be largely ignored. As an another example, when you grasp an object, the joint angles of your fingers tend to curl in synergy. Believe it or not, PCA could capture such synergies too by finding the axis that explains most variance of the ensemble joint motions.

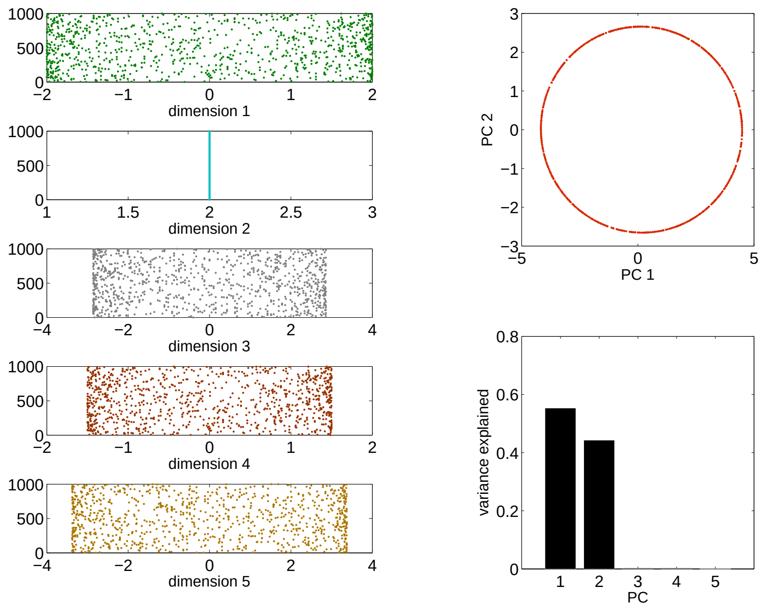

Figure 2: Visualizing low-dimensional data.

2.3 Visualization

It is often sensible to get a feel for the data before hammering any machine learning algorithms on them. Figure 2 demonstrates a 5-dimensional data set where it is difficult to figure out the underlying structure by looking merely at the scatter plot (note that plotting pairwise dimensions would help visualizing the correlations). Applying PCA, however, allows one to discover that the embedded structure is a circle. Note that only the first 2 principal components contain significant variability, suggesting that 3 out of the 5 transformed dimensions could be discarded without much loss of information.

3 A preemptive little exercise

Assuming that all PCA does is finding a projection (or rotation) matrix where along the rotated axes maximal variances of the data are preserved, what would you predict about the columns in matrix **P** in Equation 1 if we were to apply PCA to the data in Figure 1—in other words, can you think of two perpendicular unit vectors that rotate the height-weight axes in such a way where maximal variances are captured (*hint:* 45 might be a good rotating angle)?

Answer: $P_1 \approx [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T, P_2 \approx [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]^T$. This should be quite intuitive—if we assume the desirable tilting angle of the axes is roughly 45 degrees, the corresponding projections are then $[1, 1]$ and $[1, -1]$ (or $[-1, 1]$). Making these unit rotational vectors, we would normalize them by $\sqrt{(|1|^2 + |1|^2} = \sqrt{2}$ resulting in P_1 and P_2. Note that P_1 is what we call the first principal component that captures the most variability, P_2 is the second principal component that retains residual maximal variability in an orthogonal direction to P_1—if height and weight share perfect correlation, the variance along P_2 direction would be zero. If you don't find this example intuitive, no worries and read on.

4 How is it derived

There are a number of ways to derive PCA. Here we focus on the maximal-variance principle and show that the resulting optimization problem boils down to eigendecomposition of the covariance matrix.

Recall the input data matrix of N points is $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$ where each $\mathbf{x}$ is a D-dimensional vector. PCA finds a projection matrix $\mathbf{P} = [\mathbf{p}_1, ..., \mathbf{p}_{D'}]^T$ that maps each point to a low-dimensional space ($D' \leq D$). As described, each $\mathbf{p}$ is a basis vector that maximizes the variance of $\mathbf{X}$ in orthogonal directions with respect to each other, and that the amount of variance preserved decreases from $\mathbf{p}_1$ to $\mathbf{p}_{D'}$. Here we derive the first principal component $\mathbf{p}_1$, although any high-order component can be derived similarly via induction.

By definition, the covariance matrix of $\mathbf{X}$ is

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \mu)(\mathbf{x}_n - \mu)^T \tag{2}$$

where $\mu = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$ is the mean. The resulting variance by projecting the data onto $\mathbf{p}_1$ is simply

$$v' = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{p}_1^T \mathbf{x}_n - \mathbf{p}_1^T \mu)^2 = \mathbf{p}_1^T \mathbf{C} \mathbf{p}_1. \tag{3}$$

Note that v' is a scalar. PCA seeks to find $\mathbf{p}_1$ that maximizes this quantity under the constraint that $\mathbf{p}_1$ is a unit vector (i.e. the projection should be purely rotational without any scaling)

$$\mathbf{p}_1 \leftarrow \max F = \mathbf{p}_1^T \mathbf{C} \mathbf{p}_1 + \lambda_1 (1 - \mathbf{p}_1^T \mathbf{p}_1) \tag{4}$$

where the term associated with λ is the Langrange multiplier that enforces the unit vector constraint. Differentiating F with respect to $\mathbf{p}_1$ and setting the derivative to zero yields the condition for optimal projection

$$\begin{aligned}\frac{dF}{d\mathbf{p}_1} &= 0 \\ \Rightarrow \mathbf{C}\mathbf{p}_1 &= \lambda_1 \mathbf{p}_1. \end{aligned} \tag{5}$$

For those that are familiar with linear algebra, Equation 5 is identical to an eigendecomposition of matrix $\mathbf{C}$ where $\mathbf{p}_1$ is the eigenvector and λ_1 is the eigenvalue (i.e. solving for $det(\mathbf{C} - \lambda_1 \mathbf{I}) = 0$ and substituting λ_1 into $(\mathbf{C} - \lambda_1 \mathbf{I})\mathbf{p}_1 = 0$ for $\mathbf{p}_1$;). (*exercise:* perform an eigendecomposition on a simple matrix [2 3; 0 1].) Thus finding principal components is equivalent in solving an eigendecomposition problem for the covariance matrix $\mathbf{C}$. Note that this connection is intuitive because eigenvalues represent the magnitude of a matrix projected onto the corresponding eigenvectors—here the eigenvectors are the projection of the data covariance and the eigenvalues are the resulting variances from projection. If we repeat this process we would obtain $\mathbf{p}_2, ..., \mathbf{p}_{D'}$ and $\lambda_2, ..., \lambda_{D'}$ (the maximal D' is D assuming the number of samples N is greater than the dimension D). Following eigendecomposition, the covariance matrix $\mathbf{C}$ can be expressed as follows (assuming $D' = D$, i.e. $\mathbf{P}$ is a full projection matrix)

$$\mathbf{C} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T \tag{6}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with elements $\{\lambda_1, \lambda_2, ..., \lambda_D\}$ and $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_D$. Here each column in $\mathbf{P}$ is a principal component and each corresponding λ indicates the variance explained by projecting the data onto that component.

Singular value decomposition

It turns out that PCA falls under a general method for matrix decomposition called singular value decomposition (SVD). The idea behind SVD is to factor an arbitrary matrix ($\mathbf{X}$ of size $N \times D$) into the following

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{7}$$

where $\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_D]$ and $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_D]$ are orthornormal bases for the column and row spaces of $\mathbf{X}$, and $\mathbf{\Sigma}$ is a diagonal matrix with diagonal elements $\{\sigma_1, ..., \sigma_D\}$. Another way of explaining SVD is that we wish to find a mapping between bases in the column space and those in the row space

$$\sigma_i \mathbf{u}_i = \mathbf{X}\mathbf{v}_i \; (i = 1, ..., D) \tag{8}$$

where σ's here can be understood as "stretch factors" that help to match $\mathbf{u}$'s with $\mathbf{v}$'s. Equation 8 can be expressed then expressed in matrix form which yields Equation 7

$$\begin{aligned}\mathbf{U\Sigma} &= \mathbf{XV} \\ \Rightarrow \mathbf{X} &= \mathbf{U\Sigma V}^{-1} \\ \Rightarrow \mathbf{X} &= \mathbf{U\Sigma V}^{T}. \end{aligned} \tag{9}$$

The magic begins when we derive the covariance of $\mathbf{X}$, assuming it is the input data matrix as in the case of PCA

$$\begin{aligned}\mathbf{X}^T\mathbf{X} &= (\mathbf{U\Sigma V}^T)^T\mathbf{U\Sigma V}^T \\ &= \mathbf{V\Sigma}^T\mathbf{U}^T\mathbf{U\Sigma V}^T \\ &= \mathbf{V\Sigma}^2\mathbf{V}^T. \end{aligned} \tag{10}$$

Now compare Equation 10 with Equation 6—they are identical only that $\sigma_i^2 = \lambda_i\,(i = 1, ..., D)$! In other words, performing SVD is equivalent to PCA where the eigenvectors or principal components are found in $\mathbf{V}$. So what is the extra gain of SVD? Note that SVD simultaneously find the eigenvectors for $\mathbf{X}^T\mathbf{X}$ (in $\mathbf{V}$) and $\mathbf{XX}^T$ (in $\mathbf{U}$; try work this out yourself). In cases where we have more dimensions than data points, i.e. $D \geq N$, it is often convenient to decompose $\mathbf{XX}^T$ ($N \times N$) instead of the covariance matrix ($D \times D$) to save computation.

5 How is it used

The primary use of PCA is to reduce the dimension of data—given D-dimensional data $\mathbf{x}$ ($D \times 1$), PCA maps it to $\mathbf{y} = \mathbf{P}^T\mathbf{x}$ with a lower dimension D'. This reduction is often achieved by truncating the columns in the projection matrix $\mathbf{P}$ based on the amount of variance that one desires to retain. To be concrete, remember that $\mathbf{\Lambda}$ contains in its diagonal the eigenvalues in decreasing order, and the fractional variance accounted for say with the first M λ's $(\lambda_1, ..., \lambda_M)$ is simply their sum relative to the sum of all λ's. Suppose we wish to retain as much as 95% variance, then we would keep the number of columns in $\mathbf{P}$ such that the fractional sum of their corresponding λ values is at least 0.95 and discard any remaining columns (note that the fractional sum of the remaining λ's is only 0.05, implying they might just be noise—at least that is the hope). Effectively what this means is that we have set λ's with extremely small values to zero in $\mathbf{\Lambda}$—let's call the resulting diagonal matrix $\mathbf{\Lambda}'$. The truncated covariance matrix $\mathbf{C}' = \mathbf{P\Lambda}'\mathbf{P}^T$ would then have a lower rank than the original $\mathbf{C}$, hence $\mathbf{C}'$ is a low-rank approximate of $\mathbf{C}$.

Truncation is quite an art itself although there are cases where this is obvious. If the data has an intrinsic low-dimensional representation, e.g. the circle example in Figure 2, then it is likely that only a few λ's would take significantly large values. In the extreme case, however, imagine that the λ values decrease very slowly (e.g. a near-uniform distribution), then it becomes difficult to determine

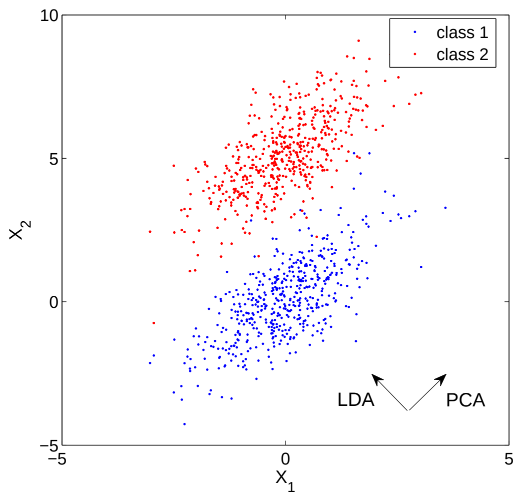

Figure 3: PCA vs LDA.

where to truncate if one should truncate at all.

PCA is particularly useful if one has a limited amount of data (e.g. images). Once the data is transformed to a low-dimensional space, one could do whatever is applicable to the original data set only now that the data is easier to handle with, e.g. regression, classification, etc. The limitation of a PCA-based regression is that it is usually hard to interpret the transformed dimensions (or covariates). For example, suppose we map the height-weight coordinates in Figure 1 to a one-dimensional space by tilting the axes 45 degrees, what would you interpret this new dimension as—it is difficult to directly associate it with an explicit physical meaning such as height and weight. In the next section, we discuss further the pitfall of PCA in classification.

6 When does it go wrong

Throughout we have doctrinated the idea that dimension reduction by maximizing variability is a sensible thing to do. Here we critique this notion and claim that it does not always work. Figure 3 shows a data set generated from two distinct classes colored in blue and red. As you now know PCA, you figure projecting these points along roughly 45 tilting angle (indicated by the arrow for PCA on the bottom right of the figure) would give you the first principal component that represents maximal variability. Note, however, that if our goal is to classify these points, this projection does only more harm than good—the majority of blue and red points would land overlapped on the first principal component, hence PCA would confuse the classifier! What went wrong?

The answer is simple—maximal variability does not imply maximal discriminability. Although the hope of retaining large variances is to preserve useful information and discard noise, it is not always compatible with the class configuration of the data. The ideal projection in this particular example is at an angle perpendicular to the first principal component (i.e. arrow for LDA in Figure 3), which clearly captures less variability yet yields almost perfect classification.

Another way to explain why PCA does not work well here is that it does not make use of the class labels—recall that PCA is a completely unsupervised algorithm that takes only the input $\mathbf{X}$ but not the class label vector (one could attempt to concatenate the labels to $\mathbf{X}$ but it is unlikely to help—think about why). Here we briefly introduce the concept of a supervised algorithm called linear discriminant analysis (LDA) that serves both as a dimension reduction method and a classifier. So in case you get frustrated that your classifier doesn't work after PCA preprocessing, at least you have something to resort to.

Like PCA, LDA also yields a linear projection of the data. Differed from PCA, LDA exploits class labels for the data in determining its projection. Specifically, LDA finds a projection that maximizes the following ratio

$$R = \frac{\mathbf{p^T C_b p}}{\mathbf{p^T C_i p}} \tag{11}$$

where $\mathbf{p}$ is the projection vector (assuming a line projection for simplicity), $\mathbf{C_b} = \sum_c (\mu_c - \mu)(\mu_c - \mu)^T$ is the between-class variability and $\mathbf{C_i} = \sum_c \sum_{n_c} (\mathbf{x}_{n_c} - \mu_c)(\mathbf{x}_{n_c} - \mu_c)^T$ is the within-class variability (here c is the class label, μ_c is the mean of class c, μ is the mean of the entire data set, and $\mathbf{x}_{n_c}$ refers to data point n in class c). Intuitively what it means is that the projection seeks to maximize the difference across classes meanwhile minimizing the variability within each class. Note that this criterion seems directly compatible with how well one could classify say two classes of data—if you project the data such that the two classes are further apart but are "close-knitted" within, it seems easy to distinguish them. Relating to Figure 3, the discriminant projection is the direction along the arrow for LDA. Imaging collapsing all the data on that projected line, you would obtain clear and non-overlapped blue and red points, yielding excellent classification. Note, however, that the maximal number of dimensions via LDA projection is $C - 1$ where C is the total number of classes.

7 Further reading

A pretty good reference is a tutorial written by Jonathon Shlens entitled "A Tutorial on Principal Component Analysis" that links PCA to singular value decomposition. Chapter 4.1.4–4.1.6 in Chris Bishop's book on PRML covers LDA in much more detail.