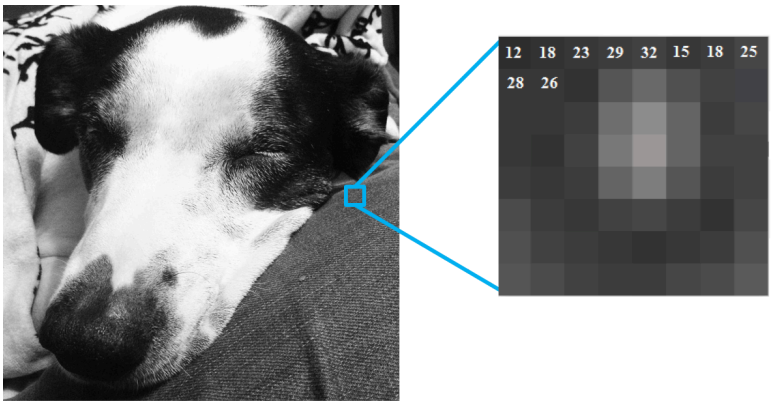
Principal Component Analysis

Contents

Introduction

Many of today's popular data types--like images, documents from the web, genetic data, consumer information--a very "high-dimensional." By high-dimensional we mean that each piece of data consists of many many individual p coordinates. For example, take a grayscale digital image like the one shown in the image below. Such an image is r many small squares called "pixels," each of which has a brightness level between 0 (completely black) and 255 (co white). In other words, a grayscale digital image can be thought of as a matrix whose $(i, j)^{\text{th}}$ value is equal to the b level of the $(i, j)^{\text{th}}$ pixel in the image. (A color image is then just a set of three such matrices, one for each color ch green, and blue.) In a megapixel grayscale image, not too large by today's standards, there are one million pixels/b values--this makes such an image one-million-dimensional datapoint (having one dimension for each of its one mi pixels).



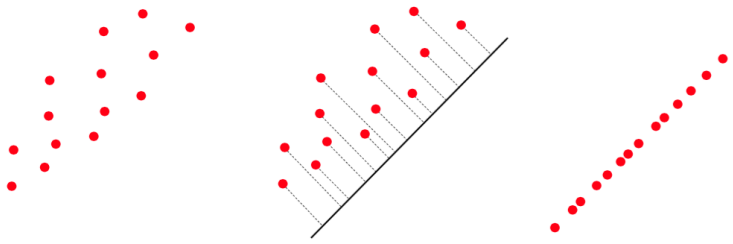
This high dimensionality makes performing standard machine learning tasks like face recognition--or the problem identifying the identity of a person from an image--directly on the image itself computationally intractable, as the calculations involved with such a task typically grows at least quadratically with the dimension of the data (see e.g Newton's method). Because of this property, reducing the dimension of data prior to processing is often a crucial making efficient machine learning algorithms.

Setup

One standard way of reducing the dimension of a data is called **principal component analysis** (or PCA for short). Geometrically speaking, PCA reduces the dimension of a dataset by squashing it onto a proper lower-dimensional

more generally a hyperplane, also often referred to as a subspace) which retains as much of the original data's def characteristics as possible.

The gist of this idea is illustrated with a two-dimensional toy data set shown in the image below, with the two-dim data squashed (or projected) onto a proper one-dimensional line that retains much of the shape of the original dat words, the dimension of the dataset has been reduced from two to one: a 50% reduction. In practice this amount much greater; for example, it is quite common for face recognition tasks to reduce images to 1% or less of their or dimension.



How do we find the PCA subspace? In short, we can set up a problem that, when properly solved, recovers a spani a collection of vectors which spans the proper subspace. To see how this is done, all we need to do is follow our nc begin by writing out our goal with PCA more formally.

So, first let us say that we have P data points $\mathbf{x}_1 \ldots \mathbf{x}_P$, each a vector of dimension N. Expressing our goal with PC formally, we want to determine a good $K (< N)$-dimensional subspace that represents the data well (the value of typically chosen according to computational limitations or via a skree plot{[}red link{]}). Remember from {[}red li algebra{]} that in N-dimensional space any K-dimensional subspace is spanned by K linearly independent vector we will denote as $\mathbf{c}_1 \ldots \mathbf{c}_K$ (note these are also length N vectors as well). In other words, any point $\mathbf{x}$ in this subspa defined by a linear combination of the spanning vectors as

$$\sum_{k=1}^{K} \mathbf{c}_k w_k = \mathbf{x},$$

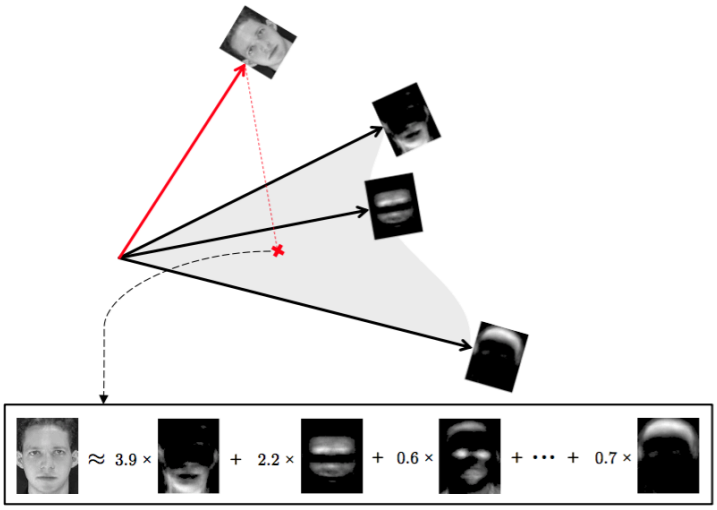
where the exact value of the weights $w_1, \ldots, w_K$ is dependent on the particular $\mathbf{x}$ chosen. With this in mind, how c formally express the goal of PCA?

Well, suppose our dataset is in fact represented very well by the subspace spanned by a particular set of vectors $\mathbf{c}$ Formally, then, if our p^{th} data point lies approximately in the span of $\mathbf{c}_1, \ldots, \mathbf{c}_K$, we have that

$$\sum_{k=1}^{K} \mathbf{c}_k w_{k,p} \approx \mathbf{x}_p,$$

where the weights $w_{1,p}, \ldots, w_{K,p}$ are tuned specifically for the point $\mathbf{x}_p$ and $\approx$ is interpreted somewhat loosely (as approximately in this span). Saying that all P points lie approximately in this span then means that the above hold

As a quick example, in the image below, we show a prototype of PCA dimension reduction applied to the task of fa recognition. Here, given a database of known individuals, where each input data $\mathbf{x}_p$ is a facial image, we apply PCA the dimension of the database. As shown in the figure, since the data itself consists of facial images, the spanning will tend to look like (rather ghostly) "basis faces."



Pictures of a single individual in the database will tend to cluster together in the lower dimensional PCA subspace, represented primarily by a few of the same basis faces. This means that the entire database itself--consisting of ir various people--will tend to look like a collection of fairly distinct clusters of points on our PCA subspace. To recog individual in a new picture, we then project the new image down onto the learned PCA subspace and identify the i by simply finding the closest cluster of projected facial images from our database and assigning the associated ide the person in the new image.

The General PCA Subspace

How do we find a set of K spanning vectors for our desired subspace? There are several ways. In this section, we numerical optimization problem that, when solved, provides some set of spanning vectors for the desired subspac section that follows, we derive a classic set of spanning vectors--known as principal components--which can be s "in closed form."

The Setup

Note that by stacking the spanning vectors column-wise into an $N \times K$ matrix $\mathbf{C}$ as $\mathbf{C} = [\mathbf{c}_1|\mathbf{c}_2| \cdots |\mathbf{c}_K]$ and by $\mathbf{w}_p = \begin{bmatrix} w_{1,p} & w_{2,p} & \cdots & w_{K,p} \end{bmatrix}^T$ we can write $\sum_{k=1}^{K} \mathbf{c}_k w_{k,p} = \mathbf{C}\mathbf{w}_p$. With this notation the previous equatio be written more compactly: $\mathbf{C}\mathbf{w}_p \approx \mathbf{x}_p$.

Now that we know what we ideally want---namely, a) a set of vectors $\mathbf{c}_1, \ldots, \mathbf{c}_K$ that spans a K-dimensional subsp which our dataset approximately lies and b) proper weights $w_{k,p}$ for each spanning vector-datapoint pair--how do actually find them? Let's follow our nose a bit further.

From Approximate Equality to Minimizing Function

Notice that if we have a good spanning set and weights, then the relationship for PCA holding for the p^{th} point me the squared distance between $\mathbf{C}\mathbf{w}_p$ and $\mathbf{x}_p$ should be quite small. In other words, the value

$$\|\mathbf{C}\mathbf{w}_p - \mathbf{x}_p\|_2^2$$

should be minimal (given indeed that the spanning vectors are linearly independent). In fact, if it is true that all of points lie close to the subspace, then the sum of these values over the dataset should be small as well; that is,

$$g = \sum_{p=1}^{P} \|\mathbf{C}\mathbf{w}_p - \mathbf{x}_p\|_2^2$$

should be minimal as well. In other words, a great set of values for our spanning set and corresponding weights mi
the above quantity, which we can also refer to as a multivariable function g. Although it may seem that we have si
phrased in a slightly different way, we have actually made a crucial step here. This is because we no longer need to
that we have the proper spanning set and weights: if we can minimize the quantity in the equation above, assumin
spanning vectors are linearly independent, we have indeed found them. In fact, many problems in machine learnin
form of a minimization problem--including linear regression, logistic regression, and neural networks--and some p
like k-means clustering even aim to minimize an extremely similar function.

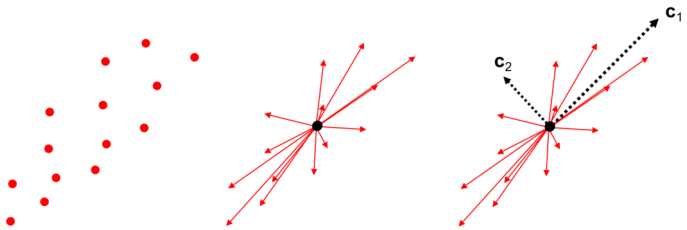
And we can solve this problem--an entire field of study known as mathematical or nonlinear optimization focuses
on methods for minimizing such multivariable functions. Any set of spanning vectors returned by minimizing the a
technically defines a PCA subspace.

Deriving Principal Component Spanning Vectors

Here, we derive a classic set of "principal component" spanning vectors, which can be computed in closed form ar
addition to being linearly independent, are actually orthonormal [link] as well (that is, they are all perpendicular to
and have unit length). This is done by deriving one principal component vector at a time--from the most to least
representative of the dataset. However, note that while these spanning vectors can be derived in "closed form," th
require considerable computation to employ in practice.

Data Points, Data Vectors

The derivation of each principal component spanning vectors results from thinking of our datapoints as vectors. F
we can always switch back and forth in our minds between thinking of a "point" equivalently as a vector, or an arr
stemming from the origin whose head lies precisely where the point does. This is illustrated with a toy dataset in th
below, where for simplicity the data is assumed to be centered at the origin. We will assume our data is origin-cent
-this will just simplify the calculations that follow and will not affect the final results. Moreover, this is usually done
practice prior to performing PCA anyway.

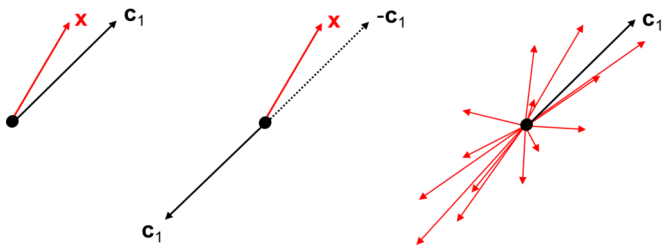


Thinking of a set of origin-centered points (left panel) as a set of vectors (middle and right panels--note in both
origin is shown as a large black dot from which the vectors stem for visualization purposes only) motivates the s
principal component spanning vectors (as those which highly correlate with the data). In the right panel, the two
black arrows are the first and second principal components of the data, $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively. Here the first co
is drawn longer for visualization purposes only.

Now that we are thinking in terms of vectors, let's reason out the value of the first principal component, or the spa
vector of our ideal subspace. This should be the unit length (spanning) that determines a line about which our dat
most spread out or, in other words, the one that generally aims in the same direction as our data vectors.

The First Principal Component: the Setup

Take the trivial example of a single data vector $\mathbf{x}$, as illustrated in the picture below; we can tell if a spanning vecto generates such a line for $\mathbf{x}$ if $\mathbf{c}_1$ or $-\mathbf{c}_1$ lies in the same direction as $\mathbf{x}$. Mathematically speaking, we can tell if this the case if the inner product between the two vectors $\mathbf{x}^T\mathbf{c}_1$ or if $\mathbf{x}^T\left(-\mathbf{c}_1\right)$ is as large and positive (the former indi $\mathbf{c}_1$ points in the direction of $\mathbf{x}$, while the latter indicating -$\mathbf{c}_1$ does). Combining these two possibilities, we can then the unit length $\mathbf{c}_1$ generates a representative line for the vector $\mathbf{x}$ if $\left(\mathbf{x}^T\mathbf{c}_1\right)^2$ is large (it is always positive).



(left panel) A vector $\mathbf{c}_1$ points in the direction of a data vector $\mathbf{x}$, and the inner product $\mathbf{x}^T\mathbf{c}_1$ is large and positiv (middle panel) Likewise, this is a case where $-\mathbf{c}_1$ points in the direction of $\mathbf{x}$ and $\mathbf{x}^T\left(-\mathbf{c}_1\right)$ is large and positive. (right panel) The case with P data vectors holds analogously (see text for further details).

Analogously, $\mathbf{c}_1$ generates a representative line for P data vectors $\mathbf{x}_1, \ldots, \mathbf{x}_P$ when the total squared inner produc the data

$$\sum_{p=1}^{P}\left(\mathbf{x}_p^T\mathbf{c}_1\right)^2$$

is large, and the larger the better. In fact, we want this to be as large as possible, since the larger its value the bett generated by $\mathbf{c}_1$ captures the spread of the dataset. So, how do we determine the $\mathbf{c}_1$ that maximizes this quantity

The First Principal Component: the Derivation

Here comes the math. Let's rewrite the equation above--this will help tease out the answer. Denote by $\mathbf{X}$ the $N \times$ formed by stacking the data vectors columnwise as $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_P \end{bmatrix}$. Then the above can be written equivalently as

$$\sum_{p=1}^{P}\left(\mathbf{x}_p^T\mathbf{c}_1\right)^2 = \|\mathbf{X}^T\mathbf{c}_1\|_2^2 = \mathbf{c}_1^T\mathbf{X}\mathbf{X}^T\mathbf{c}_1.$$

Here's where we need a fundamental fact from linear algebra known as the eigenvalue decomposition of a matrix the spectral theorem of symmetric matrices. Using this fact, we can decompose the matrix $\mathbf{X}\mathbf{X}^T$ as $\mathbf{X}\mathbf{X}^T = \sum_{p=1}^{P}$ where each d_p is a real eigenvalue, $d_1 \geq d_2 \geq \cdots \geq d_P$, and the set of $N \times 1$ eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_P$ are orthonc is, orthogonal and of unit length). Replacing $\mathbf{X}\mathbf{X}^T$ with this eigendecomposition in the equation above, we then h equivalently that

$$\mathbf{c}_1^T\mathbf{X}\mathbf{X}^T\mathbf{c}_1 = \mathbf{c}_1^T\left(\sum_{p=1}^{P} d_p\mathbf{e}_p\mathbf{e}_p^T\right)\mathbf{c}_1 = \sum_{p=1}^{P} d_p\mathbf{c}_1^T\mathbf{e}_p\mathbf{e}_p^T\mathbf{c}_1 = \sum_{p=1}^{P} d_p\left(\mathbf{e}_p^T\mathbf{c}_1\right)^2.$$

Now, since d_1 is the largest eigenvalue, we can see that it must be the smallest upper bound possible on this quant

$$\sum_{p=1}^{P} d_p\left(\mathbf{e}_p^T\mathbf{c}_1\right)^2 \leq \sum_{p=1}^{P} d_1\left(\mathbf{e}_p^T\mathbf{c}_1\right)^2 = d_1\sum_{p=1}^{P}\left(\mathbf{e}_p^T\mathbf{c}_1\right)^2 = d_1.$$

Here the last equality follows from the fact that the eigenvectors form a basis over which we may decompose $\mathbf{c}_1$ a $\sum_{p=1}^{P} \alpha_p \mathbf{e}_p$ and hence $\alpha_p = \mathbf{e}_p^T \mathbf{c}_1$, and since $\mathbf{c}_1$ has unit length by assumption we have that $\|\mathbf{c}_1\|_2^2 = \sum_{p=1}^{P} \left(\mathbf{e}_p^T \mathbf{c}_1\right)^2$

Now, since $\mathbf{c}_1$ as well as the eigenvectors are all unit length, if we set $\mathbf{c}_1 = \mathbf{e}_1$, we can actually achieve this upper b since the eigenvectors are orthonormal, i.e. $\mathbf{e}_p^T \mathbf{e}_1 = 0$ when $p \neq 1$, and $\mathbf{e}_p^T \mathbf{e}_1 = 1$ when $p = 1$) since we have

$$\sum_{p=1}^{P} d_p \left(\mathbf{e}_p^T \mathbf{c}_1\right)^2 = \sum_{p=1}^{P} d_p \left(\mathbf{e}_p^T \mathbf{e}_1\right)^2 = d_1.$$

Therefore, the first principal component is $\mathbf{c}_1 = \mathbf{e}_1$, i.e. the eigenvector associated with the largest eigenvalue of t $\mathbf{X}\mathbf{X}^T$.

Deriving the Second, Third, ... Principal Components

Deriving the additional principal components follows by induction in a manner very much analogous with the deriv the first. That is, suppose we have derived the first $k-1$ principal components as the $k-1$ eigenvectors associa the $k-1$ largest eigenvalues of the matrix $\mathbf{X}\mathbf{X}^T$. We then want to show that the k^{th} principal component $\mathbf{c}_k$ is th eigenvector of $\mathbf{X}\mathbf{X}^T$ associated with the k^{th} largest eigenvalue.

In order to do this, we again use the eigen-decomposition of $\mathbf{X}\mathbf{X}^T$ in order to write the above as

$$\mathbf{c}_k^T \mathbf{X}\mathbf{X}^T \mathbf{c}_k = \sum_{p=1}^{P} d_p \left(\mathbf{e}_p^T \mathbf{c}_k\right)^2 = \sum_{p=k}^{P} d_p \left(\mathbf{e}_p^T \mathbf{c}_k\right)^2,$$

where the last equality holds because $\mathbf{c}_k$ is assumed perpendicular to the first $k-1$ principal components. Follow argument for the first principal component, here again we can derive an upper bound on the above as

$$\sum_{p=k}^{P} d_p \left(\mathbf{e}_p^T \mathbf{c}_k\right)^2 \leq d_k \sum_{p=k}^{P} \left(\mathbf{e}_p^T \mathbf{c}_k\right) \leq d_k$$

and complete the proof by again noting that we can meet the upper bound by setting $\mathbf{c}_k = \mathbf{e}_k$.