

Lecture 10

*Lecturer: Eran Halperin**Scribe: Eran Halperin***10.1 Singular Value Decomposition**

We will start this lecture with a description of the singular value decomposition of a matrix X . Let X be an $m \times n$ matrix over the real numbers. We can decompose the X into the product of three matrices, i.e., $X = U\Sigma V^t$, where U is an $m \times m$ orthonormal matrix, V is an $n \times n$ orthonormal matrix, and Σ is a diagonal $m \times n$ matrix with non-negative numbers on the diagonal, such that $\Sigma_{11} \geq \Sigma_{22} \geq \dots \geq \Sigma_{nn}$ in case $n \leq m$. We denote $\lambda_i = \Sigma_{ii}^2$.

$$X = \begin{pmatrix} | & & | \\ u_1 & \dots & u_m \\ | & & | \end{pmatrix} \begin{pmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ 0 & & \sqrt{\lambda_n} \\ & & & \ddots \\ & & & & 0 \end{pmatrix} \begin{pmatrix} - & v_1 & - \\ \vdots & \vdots & \vdots \\ - & v_n & - \end{pmatrix}$$

The derivation of the decomposition was given in the slides for the case where $m = n$. Here we give it in the case where $m > n$. Since the matrix XX^t is symmetric and positive semidefinite, all its eigenvalues are real and nonnegative. Based on the spectral decomposition of XX^t , we know that

$$XX^t = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{pmatrix} U^t,$$

where U is an orthonormal matrix, with the columns corresponding to the eigenvectors of XX^t , and $\lambda_1 \geq \dots \geq \lambda_m$ corresponding to the eigenvalues of XX^t . Note that since the rank of XX^t is at most n , we have $\lambda_{n+1} = \dots = \lambda_m = 0$. Let k be the maximal index of a positive eigenvalue, that is $\lambda_k > \lambda_{k+1} = 0$. Define U_k as the $m \times k$ matrix consisting of the first k columns of U (i.e., the first k eigenvectors of XX^t), and U_{m-k} be the $m \times (m - k)$ matrix consisting of the last $m - k$ columns of U . Also, define Σ_k to be the diagonal matrix with

the square root of the k positive eigenvalues on the diagonal, i.e.,

$$U_k = \begin{pmatrix} | & & | \\ u_1 & \dots & u_k \\ | & & | \end{pmatrix}, \quad \Sigma_k = \begin{pmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_k} \end{pmatrix}$$

We define $V_k = X^t U_k \Sigma_k^{-1}$, and let V_{n-k} be a matrix whose columns consist orthonormal vectors that are orthogonal to the set of columns of V_k . We then define $V = [V_k V_{n-k}]$ (concatenating the two matrices). Clearly, V is an $n \times n$ orthonormal matrix by definition.

Moreover, we define $\Sigma = \begin{pmatrix} \Sigma_k & 0 \\ 0 & 0 \end{pmatrix}$, so that Σ is of size $m \times n$. We note that since $XX^t = U\Sigma^2 U^t$, we have that $\Sigma^2 = U^t X X^t U$, and therefore $0 = U_{m-k}^t X X^t U_{m-k} = \|U_{m-k}^t X\|^2$. Thus, $U_{m-k}^t X = 0$. Noting that $U_k U_k^t + U_{m-k} U_{m-k}^t = I_m$, we get:

$$U\Sigma V^t = U_k \Sigma_k V_k^t = U_k U_k^t X = (I - U_{m-k} U_{m-k}^t) X = X$$

Note that the transposed matrix has the same eigenvectors, since $X^t X v_i = V \Sigma^2 V^t v_i = \lambda_i v_i$.

10.2 SVD and linear regression

Before we turn to the main theme of the lecture, let us consider an important application of SVD. Recall that in linear regression we were interested in solving the following problem:

$$\hat{a} = \min_a \|y - Xa\|_2^2$$

The solution to this problem is obtained by the Normal Equations as $\hat{a} = (X^t X)^{-1} X^t y$. If $X^t X$ is singular, there are many possible solutions. One sensible criterion that chooses one among all possible solutions, is a solution of the Normal Equations with a minimal L_2 norm. That is, we search for a that optimizes the following (note that unless otherwise stated, the norm is the L_2 norm):

$$\begin{aligned} \min \quad & \|a\| \\ \text{s.t.} \quad & X^t X a = X^t y \end{aligned}$$

If we use the SVD decomposition $X = U\Sigma V^t$, we get that the constraint is $V\Sigma^2 V^t a = V\Sigma U^t y$. We can multiply by V^t both sides, and we get $\Sigma^2 V^t a = \Sigma U^t y$.

Now, let us express a as a linear combination of the eigenvectors. This can always be done since the eigenvectors form an orthonormal basis in R^n . Thus, we write $a = \sum_{i=1}^n \alpha_i v_i$.

Since the eigenvectors are orthonormal, we have $\|a\|^2 = \sum_{i=1}^n \alpha_i^2$, so we are interested in minimizing the sum $\sum_i \alpha_i^2$. Now, $a^t V = (\alpha_1, \dots, \alpha_n)$, and therefore

$$\Sigma^2 V^t a = \begin{pmatrix} \alpha_1 \lambda_1 \\ \alpha_2 \lambda_2 \\ \dots \\ \dots \\ \alpha_n \lambda_n \end{pmatrix}.$$

Now, we plug in the constraint $\Sigma^2 V^t a = \Sigma U^t y$ and we get that $\alpha_i \lambda_i = \sqrt{\lambda_i} u_i^t y$. Now, if $\lambda_i > 0$ we get that $\alpha_i = \frac{u_i^t y}{\sqrt{\lambda_i}}$, and otherwise α_i is a free variable (its value does not affect the constraint). Therefore, it is best to choose $\alpha_i = 0$ if we want to minimize $\sum_{i=1}^n \alpha_i^2$. Therefore, we get:

$$a = \sum_{i; \lambda_i > 0} \frac{1}{\sqrt{\lambda_i}} u_i^t y v_i = V \Sigma^{-1} U^t y.$$

Note that in the above formulation, we denote by Σ^{-1} the $n \times m$ diagonal matrix with $\Sigma_{ii}^{-1} = \frac{1}{\Sigma_{ii}}$ if $\Sigma_{ii} > 0$, and $\Sigma_{ii}^{-1} = 0$ otherwise. The matrix $V \Sigma^{-1} U^t$ is called the pseudo-inverse of X^t .

Note that the solution for Ridge regression with regularization parameter ϵ is $\hat{a}_{Ridge, \epsilon} = (X^t X + \epsilon I)^{-1} X^t y$. By plugging in the SVD, we get that $\hat{a}_{Ridge, \epsilon} = V(\Sigma^t \Sigma + \epsilon I)^{-1} \Sigma^t U^t y$. Now,

$$(\Sigma^t \Sigma + \epsilon I)^{-1} \Sigma^t = \begin{pmatrix} \frac{\sqrt{\lambda_1}}{\epsilon + \lambda_1} & & & \\ & \ddots & & \\ & & \frac{\sqrt{\lambda_n}}{\epsilon + \lambda_n} & \\ 0 & & & 0 \end{pmatrix} \xrightarrow{\epsilon \rightarrow 0} \Sigma^{-1}$$

10.3 Unsupervised learning in high dimension

During most of the course, we discussed the supervised learning scenario, in which we get a training data in the form of pairs (x_i, y_i) , where x_i is a vector of features and y_i is a label (and we saw in the last lecture that by label we may mean a real number). We then get test samples, in which we only obtain the features x_i and we need to estimate the value of y_i .

We will now turn to the unsupervised scenario. In the unsupervised scenario we get a set of test features x_1, \dots, x_m , and we need to assign labels to the samples. Clearly this

is a harder problem, and often there is not enough information in order to say something meaningful. But in many cases it is possible to learn something based on the relations between the points x_i in the feature space. For instance, we introduced earlier on the k-means algorithm, and the EM for mixture of Gaussians. These clustering algorithms are unsupervised learning algorithms that use the Euclidean distances between the points in order to cluster them together. There are other possible conclusions we could potentially draw; for instance, we could figure out that some of the samples are outliers, i.e., they are obtained by a different process.

Dealing with the unsupervised case becomes intrinsically difficult when the feature space is high-dimensional. There are many such cases. Consider for example the task of document classification; each document can be described by its 'bag of words', i.e., the histogram of words used in the document. Thus, a document is described as a vector in a high dimension (the number of dimensions is the number of words in dictionary). Many of the words are not informative with respect to the task of classification; ideally, we would like to work in a lower dimension that is represented only by the words that are relevant to the classification. Of course, there is a circular argument here, since if we knew which words are relevant we would know the classification.

There are many other examples for high dimensional data in unsupervised (as well as supervised) scenarios. When we try to cluster photos, each photo consists of thousands or sometimes millions of pixels, so we can view them as vectors over millions of dimensions. In genetics we are usually interested in the clustering individuals based on their genetic content. Their genetic information consists of more than 20,000 genes, and the genome of an individual can be described by a long string of length 3,000,000,000. Thus, in this case the dimension of each sample is 20,000 or 3,000,000,000, depending on the study. Again, if we think of a typical biological process, it will not involve all 20,000 genes, and ideally we would like to study only the relevant genes, thus work in a lower dimension.

10.4 Principal Component Analysis

Due to the above limitation of high-dimensional data, we will now introduce a method for dimensionality reduction. Dimensionality reduction is useful since the number of features reduces to the number of dimensions, thus the risk of over fitting is lower. We can also view the task of dimensionality reduction as a compression task - we preserve the structure of the data (as much as possible) but at a lower dimension. Finally, by reducing the original dimension to two or three dimensions we can visualize the data - the visualization often allows us to cluster the data manually, and to exclude outliers from the data.

More formally, assume we have a set of input vectors x_1, \dots, x_n , where $x_i \in R^m$. Let $X = (x_1, \dots, x_n)$ be the $m \times n$ matrix composed of these vectors as columns. Our goal is to reduce the dimension m . Particularly, we are interested to find a set of vectors z_1, \dots, z_n ,

such that z_i is a projection of x_i on a lower dimensional hyperplane. Note that in this scribe all the vectors are column vectors. More formally, we are searching for an orthonormal matrix U of size $m \times d$, where $d \ll m$, such that $z_i = U^t x_i \in R^d$. Denote the columns of U by u_1, \dots, u_d ; the orthonormality constraints imply that $u_i^t u_j = 0$ if $i \neq j$, and $\|u_i\| = 1$.

The main question remaining is how do we choose U wisely. We can think of the above procedure as compression. In the encoding step, we encode x_i by assigning $z_i = U^t x_i \in R^d$. In the decoding step we decode z_i by extracting $x'_i = U z_i = U U^t x_i \in R^m$. Note that the projection $x_i \rightarrow U U^t x_i \in R^m$ projects the point x_i into the space spanned by u_1, \dots, u_d . To see this, note that $U U^t u_i = u_i$, and that if v is orthogonal to u_1, \dots, u_d we have $u_i^t v = 0$, and therefore $U U^t v = U(U^t v) = 0$.

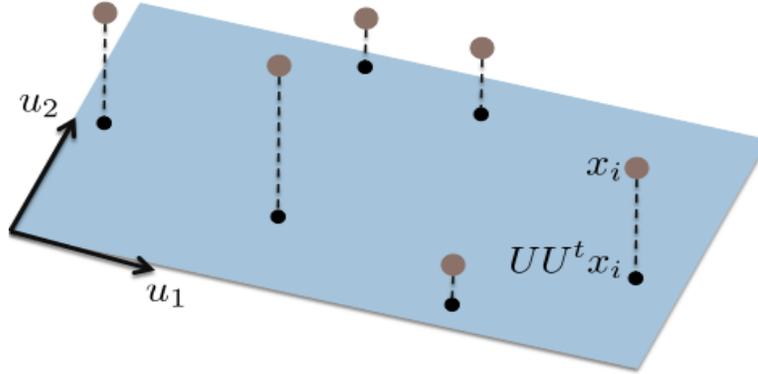


Figure 10.1: Geometric interpretation of PCA

This provides a very natural criterion for choosing U . We will choose a matrix U that minimizes the reconstruction error, i.e.,

$$U = \arg \min_{U \in R^{m \times d}} \sum_i \|x_i - U U^t x_i\|^2. \quad (10.1)$$

We now consider another possible criterion. Assume without loss of generality that the data points are centered at zero, that is

$$\sum_{i=1}^n x_i = 0.$$

Under a generative model in which the points are sampled from a generative distribution, an unbiased estimate of the variance can be shown to be

$$\text{Var}(X) = \frac{1}{n-1} \sum_{i=1}^n x_i^t x_i.$$

The reason for the $\frac{1}{n-1}$ coefficient instead of $\frac{1}{n}$ is so that the estimate will be unbiased, but this is not critical, if you are not familiar with statistics you can simply ignore the coefficient or think of it as $\frac{1}{n}$. The estimate of the variance of the projected distance is:

$$\text{Var}(X') = \frac{1}{n-1} \sum_{i=1}^n x_i^t U U^t U U^t x_i = \frac{1}{n-1} \sum_{i=1}^n x_i^t U U^t x_i = \frac{1}{n-1} \sum_{i=1}^n \|U^t x_i\|^2. \quad (10.2)$$

A sensible criterion for dimensionality reduction would be to choose U so that the variance $\text{Var}(X')$ is maximized, i.e., intuitively the structure of the data is preserved as much as possible. We note, however, the following equality, based on Pythagoras (see Figure 10.2):

$$\|x_i\|^2 = \|U^t x_i\|^2 + \|x_i - U U^t x_i\|^2.$$

Since $\|x_i\|$ does not depend on U , we see that **minimizing the reconstruction error is equivalent to maximizing the variance**. The goal in principal component analysis (PCA) is therefore to minimize the reconstruction error (see Equation 10.1), and to maximize the projected variance (Equation 10.2).

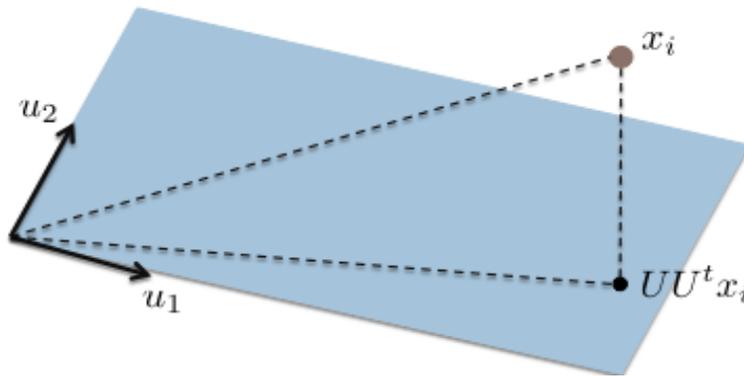


Figure 10.2: Pythagoras: $\|x_i\|^2 = \|U^t x_i\|^2 + \|x_i - U U^t x_i\|^2$.

10.5 Computing PCA

10.5.1 Computing PCA in one dimension

We first show how PCA is computed when we consider projections to one dimensions, i.e., $u_1 = U \in R^m$, where $\|u_1\| = 1$. In this case, we can rewrite the objective function as

$$u_1 = \arg \max_{u \in R^m, \|u\|=1} \sum_{i=1}^n x_i^t u u^t x_i$$

It turns out we can reformulate the problem as an eigenvalue problem using the following observation:

$$\sum_{i=1}^n x_i^t u u^t x_i = \sum_{i=1}^n (u^t x_i)^2 = \|u^t X\|_2^2 = u^t X X^t u.$$

Now, the largest eigenvector v_1 of a matrix A can be found by maximizing the following:

$$v_1 = \arg \max_{\|v\|=1} v^t A v$$

Therefore, we get that u_1 is simply the eigenvector corresponding to the largest eigenvalue of $X X^t$.

In order to show that this $v^t A v$ is maximized at the dominant eigenvector, let v_1, \dots, v_n be the n eigenvectors of A , with corresponding eigenvalues $\lambda_1 \geq \dots \geq \lambda_n$. Assume $v = \sum_i \alpha_i v_i$, that is, $\alpha_i = v_i^t v$. Then

$$v^t A v = v^t \sum_{i=1}^n \alpha_i A v_i = \left(\sum_{i=1}^n \alpha_i v_i^t \right) \left(\sum_{i=1}^n \alpha_i \lambda_i v_i \right) = \sum_{i=1}^n \alpha_i^2 \lambda_i. \quad (10.3)$$

The second equality holds since $A v_i = \lambda_i v_i$ as v_i is an eigenvector of A , and the third equality holds since $v_i^t v_j = 0$ if $i \neq j$, and $v_i^t v_i = 1$. Now,

$$\sum_{i=1}^n \alpha_i^2 \lambda_i \leq \lambda_1 \sum_{i=1}^n \alpha_i^2 = \lambda_1,$$

where the last equality holds since the vector v is a unit vector. The statement holds since when setting $v = v_1$ we get $v_1^t A v_1 = \lambda_1$.

10.5.2 Computing PCA in d dimensions

We now turn to the case of PCA in d dimensions. In this case, we want to maximize $\sum_{i=1}^n x_i^t U U^t x_i$. Note that $U U^t = \sum_{j=1}^d u_j u_j^t$, and we can therefore write

$$\sum_{i=1}^n x_i^t U U^t x_i = \sum_{i=1}^n x_i^t \left(\sum_{j=1}^d u_j u_j^t \right) x_i = \sum_{j=1}^d u_j^t X X^t u_j. \quad (10.4)$$

This is very similar to the one dimensional case. Again, let v_1, \dots, v_m be the eigenvectors of $X X^t$, with eigenvalues $\lambda_1 \geq \dots \geq \lambda_m$. We will use the following fact (shown above in Equation 10.3):

$$u_j^t X X^t u_j = \sum_{k=1}^m (u_j^t v_k)^2 \lambda_k. \quad (10.5)$$

Using Equations 10.4 and 10.5, we get:

$$\sum_{i=1}^n x_i^t U U^t x_i = \sum_{j=1}^d u_j^t X X^t u_j = \sum_{j=1}^d \sum_{k=1}^m (u_j^t v_k)^2 \lambda_k = \sum_{k=1}^m \lambda_k \sum_{j=1}^d (u_j^t v_k)^2 \quad (10.6)$$

Now, note that v_k is a unit vector, and U is orthonormal, and therefore $\sum_{j=1}^d (u_j^t v_k)^2 \leq 1$, and $\sum_{k=1}^m (u_j^t v_k)^2 = 1$, thus $\sum_{k=1}^m \sum_{j=1}^d (u_j^t v_k)^2 = d$. Denote $a_k = \sum_{j=1}^d (u_j^t v_k)^2$. Consider the following linear program:

$$\begin{aligned} \max \quad & \sum_{k=1}^m \lambda_k a_k \\ \text{s.t.} \quad & \sum_{k=1}^m a_k = d \\ & 0 \leq a_k \leq 1 \end{aligned}$$

The optimal solution of this linear program is $\lambda_1 + \dots + \lambda_d$. In order to see this, note that if for some $k < d$ we have $a_k < 1$, we must have $k' > d$ for which $a_{k'} > 0$. Consider an alternative solution with $a_k = a_k + \epsilon$, $a_{k'} = a_{k'} - \epsilon$. This solution is at least as large since we increase the total solution by $\epsilon(\lambda_k - \lambda_{k'}) \geq 0$. Therefore, without loss of generality $a_k = 1$ for $k \leq d$, and the value of the solution is $\lambda_1 + \dots + \lambda_d$. Based on this, and based on Equation 10.6 we see that

$$\sum_{i=1}^n x_i^t U U^t x_i = \sum_{k=1}^m \lambda_k \sum_{j=1}^d (u_j^t v_k)^2 = \sum_{k=1}^m \lambda_k a_k \leq \sum_{k=1}^d \lambda_k.$$

We note that when $u_k = v_k$ then $a_k = 1$ for $k = 1, \dots, d$, and therefore this is an optimal solution. There are other optimal solutions, and in fact any orthonormal basis spanning the space defined by v_1, \dots, v_k is an optimal solution to the PCA. However, we are interested in the specific solution in which $u_k = v_k$, since then we can view the PCA as an iterative process, in which we first obtain u_1 as the maximal eigenvector of $X X^t$, we then compute u_2 , as the second largest eigenvector of $X X^t$, etc.

Thus, in order to compute the solution to PCA we need to compute the matrix $X X^t$ and find its d dominant eigenvectors. This task is usually more expensive and less numerically stable than the calculation of the singular value decomposition. Therefore, we will usually compute the SVD of X , i.e., $X = U \Sigma V^t$, and use the columns of U as the principal components, i.e., the solution to the PCA.

10.5.3 PCA - General Comments and Practical Issues

The vectors u_1, \dots, u_d are called the principal components, and the entries of the vectors are called the loadings (i.e., the loadings of the first principal component are u_{11}, \dots, u_{1m}). Based on the previous section, we see that the contribution of the i -th principal component to the overall variance of the projected data is $u_i^t X X^t u_i = \lambda_i$. We can therefore determine the number of PCs based on the eigenvalue distribution, namely, we can calculate exactly the percentage of variance explained by the first d eigenvectors.

The PC scores are the projection of x_i onto each of the principal components. The scores of the i -th principal component is given by $u_i^t X$. Now, using the singular value decomposition $X = U \Sigma V^t$, we observe that

$$u_i^t X = u_i^t U \Sigma V^t = \sqrt{\lambda_i} v_i.$$

Thus, we note that the scores themselves are eigenvectors of $X^t X$, i.e.,

$$v_i = \frac{u_i^t X}{\|u_i^t X\|} = \arg \max_{\|v\|=1, \forall j < i, v^t v_j = 0} v^t X^t X v$$

In practice, in order to get a sensible interpretation of the data, it is recommended to standardize the features, so that each one of them has a variance 1. Without standardization different features will contribute differently to the total variance. Take for example a case in which we measure the weight of people in grams and the height in meters. Clearly, without normalization most of the variance of the data will appear to arise from the weight, but this is only due to the fact that we used a highly variant measurement. Notably, in some cases it is still important to keep the original values if one would like to preserve the original variance of each feature - for example, arguably when considering PCA on genetic data, it may be better not to normalize each of the genetic variants if we want to be able to interpret the biological reason for the PCs.

PCA is a great tool for the discovery of outliers. When considering the first PCs we can cluster the samples in the projected space, and check whether some of the samples fall outside the clusters. Often points fall outside the clusters since the process of selecting these samples or measuring their features was in some way different than the rest of the group (for instance, maybe these samples incurred many more errors in the measurements).

Typically, the first few PCs capture variation in the data that is an artifact of the process in which the data was generated. For example, in the case of genetic data the first PCs would normally represent the situation in which the data was generated, for example in which lab or which equipment was used to generate the genomic measurements. In the case of PCA of pictures (e.g., in eigenfaces), the first PCs may reflect the direction of the illumination in the photo, or general properties of the camera used. Clearly, these type of variation are not of interest when studying genetics or when studying properties of the photos. One solution

to this issue is to simply ignore these PCs when treating PCA as a compression tool, or alternatively to regress the data on the first PCs when treating PCA as a normalization tool.