13. tétel

Kruppa Zoltán

2019. június 17.

Kivonat

Dimenzióredukciós módszerek – Magas dimenziós adatok statisztikus tulajdonságai. A főkomponens-analízis és alkalmazásai, t-SNE.

Tartalomjegyzék

1. Introduction

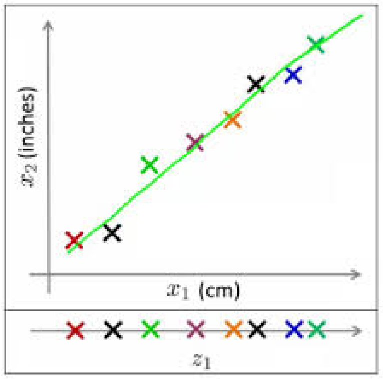
In statistics, machine learning, and information theory, dimensionality reduction or dimension reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.[4]

2. Dimension Reduction[1]

2.1. What are Dimension Reduction techniques?

Dimension Reduction refers to the process of converting a set of data having vast dimensions into data with lesser dimensions ensuring that it conveys similar information concisely. These techniques are typically used while solving **machine learning problems** to obtain better features for a classification or regression task.

Let's look at the image shown below (Fig 1). It shows 2 dimensions x_1 and x_2, which are let us say measurements of several object in cm (x_1) and inches (x_2). Now, if you were to use both these dimensions in machine learning, they will convey similar information and introduce a lot of noise in system, so you are better of just using one dimension. Here we have converted the dimension of data from $2D$ (from x_1 and x_2) to $1D$ (z_1), which has made the data relatively easier to explain.



1. ábra. Measurements of several object in cm

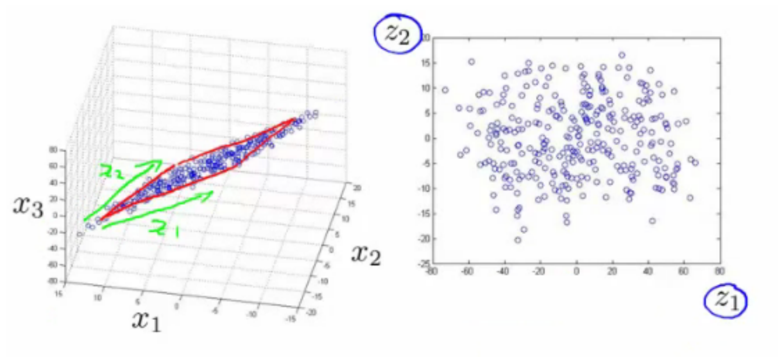
In similar ways, we can reduce n dimensions of data set to k dimensions $(k < n)$. These k dimensions can be directly identified (filtered) or can be a combination of dimensions (weighted averages of dimensions) or new dimension(s) that represent existing multiple dimensions well.

One of the most common application of this technique is **Image processing**. You might have come across this Facebook application – "Which Celebrity Do You Look Like?". But, have you ever thought about the algorithm used behind this?
Here's the answer: To identify the matched celebrity image, we use pixel data and each pixel is equivalent to one dimension. In every image, there are high number of pixels i.e. high number of dimensions. And every dimension is important here. You can't omit dimensions randomly to make better sense of your overall data set. In such cases, dimension reduction techniques help you to find the significant dimension(s) using various method(s). We'll discuss these methods shortly.

2.2. What are the benefits of Dimension Reduction?

Let's look at the benefits of applying Dimension Reduction process:

- It helps in data compressing and reducing the storage space required
- It fastens the time required for performing same computations. Less dimensions leads to less computing, also less dimensions can allow usage of algorithms unfit for a large number of dimensions
- It takes care of multi-collinearity that improves the model performance. It removes redundant features. For example: there is no point in storing a value in two different units (meters and inches).
- Reducing the dimensions of data to $2D$ or $3D$ may allow us to plot and visualize it precisely. You can then observe patterns more clearly. Below you can see that, how a $3D$ data is converted into $2D$. First it has identified the $2D$ plane then represented the points on these two new axis z_1 and z_2.



- It is helpful in noise removal also and as result of that we can improve the performance of models.

2.3. What are the common methods to perform Dimension Reduction?

There are many methods to perform Dimension reduction. I have listed the most common methods below:

1. **Missing Values:** While exploring data, if we encounter missing values, what we do? Our first step should be to identify the reason then impute missing values/ drop variables using appropriate methods. But, what if we have too many missing values? Should we impute missing values or drop the variables?

2. **Low Variance:** Let's think of a scenario where we have a constant variable (all observations have same value, 5) in our data set. Do you think, it can improve the power of model? Of course NOT, because it has zero variance. In case of high number of dimensions, we should drop variables having low variance compared to others because these variables will not explain the variation in target variables.

3. **Decision Trees:** It can be used as a ultimate solution to tackle multiple challenges like missing values, outliers and identifying significant variables. It worked well in our Data Hackathon also. Several data scientists used decision tree and it worked well for them.

4. **Random Forest:** Similar to decision tree is Random Forest. It would also be recommend using the in-built feature importance provided by random forests to select a smaller subset of input features. Just be careful that random forests have a tendency to bias towards variables that have more no. of distinct values i.e. favor numeric variables over binary/categorical values.

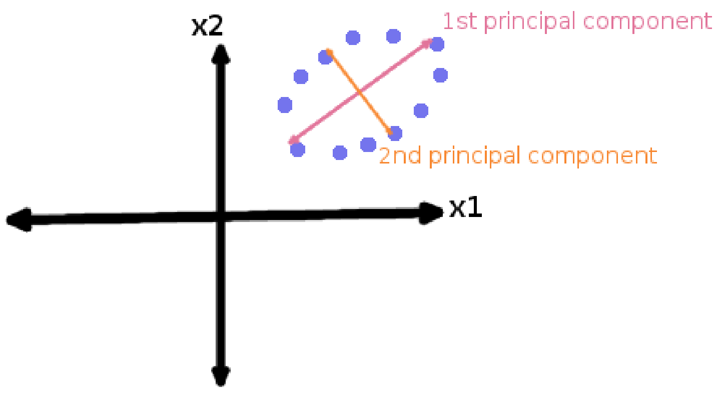
5. **Factor Analysis:** Let's say some variables are highly correlated. These variables can be grouped by their correlations i.e. all variables in a particular group can be highly correlated among themselves but have low correlation with variables of other group(s). Here each group represents a single underlying construct or factor. These factors are small in number as compared to large number of dimensions. However, these factors are difficult to observe. There are basically two methods of performing factor analysis:

   - EFA (Exploratory Factor Analysis)

- CFA (Confirmatory Factor Analysis)

6. **Principal Component Analysis (PCA):** In this technique, variables are transformed into a new set of variables, which are linear combination of original variables. These new set of variables are known as **principle components**. They are obtained in such a way that first principle component accounts for most of the possible variation of original data after which each succeeding component has the highest possible variance.

   The second principal component must be orthogonal to the first principal component. In other words, it does its best to capture the variance in the data that is not captured by the first principal component. For two-dimensional dataset, there can be only two principal components. Below is a snapshot of the data and its first and second principal components. You can notice that second principle component is orthogonal to first principle component.



   The principal components are sensitive to the scale of measurement, now to fix this issue we should always standardize variables before applying PCA. Applying PCA to your data set loses its meaning. If interpretability of the results is important for your analysis, PCA is not the right technique for your project.

3. Principal component analysis

I wrote this chapter based on the Mark Richardson's note[3].
Assume that we start with a data set that is represented in terms of an $m \times n$ matrix, $\mathbf{X}$ where the n columns are the samples (e.g. observations) and the m rows are the variables. We wish to linearly transform this matrix, $\mathbf{X}$ into another matrix, $\mathbf{Y}$, also of dimension $m \times n$, so that for some $m \times m$ matrix, $\mathbf{P}$,

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

This equation represents a change of basis. If we consider the *rows* of $\mathbf{P}$ to be the row vectors $\mathbf{p}_1\mathbf{p}_2, \cdots \mathbf{p}_m$, and the *columns* of $\mathbf{X}$ to be the column vectors $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, then can be interpreted in the following way.

$$\mathbf{PX} = (\mathbf{Px}_1, \mathbf{Px}_2 \cdots \mathbf{Px}_n) = \begin{pmatrix} \mathbf{p}_1\mathbf{x}_1 & \mathbf{p}_1\mathbf{x}_2 & \cdots & \mathbf{p}_1\mathbf{x}_n \\ \mathbf{p}_2\mathbf{x}_1 & \mathbf{p}_2\mathbf{x}_2 & \cdots & \mathbf{p}_2\mathbf{x}_n \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_m\mathbf{x}_1 & \mathbf{p}_m\mathbf{x}_2 & \cdots & \mathbf{p}_m\mathbf{x}_n \end{pmatrix} = \mathbf{Y} \tag{2}$$

Note that $p_i, x_j \in \mathbb{R}^m$, and so $\mathbf{p}_i\mathbf{x}_j$ is just the standard Euclidean inner (dot) product. This tells us that the original data, $\mathbf{X}$ is being projected on to the *columns* of $\mathbf{P}$. Thus, the rows of $\mathbf{P}$, $\{\mathbf{p}_1\mathbf{p}_2, \cdots \mathbf{p}_m\}$ are a new basis for representing the columns of X. The rows of $\mathbf{P}$ will later become our principal component directions

We now need to address the issue of what this new basis should be, indeed what is the 'best' way to re-express the data in $\mathbf{X}$ - in other words, how should we define *independence* between principal components in the new basis?

Principal component analysis defines independence by considering the variance of the data in the original basis. It seeks to de-correlate the original data by finding the directions in which variance is maximised and then use these directions to define the new basis. Recall the definition for the variance of a random variable, Z with mean, μ.

$$\sigma_Z^2 = E((Z-\mu)^2) \tag{3}$$

Suppose we have a vector of n discrete measurements, $\tilde{\mathbf{r}} = (\tilde{r}_1, \tilde{r}_2, \cdots, \tilde{r}_n)$, with mean μ_r. If we subtract the mean from each of the measurements, then we obtain a translated set of measurements $\mathbf{r} = (r_1, r_2, \cdots, r_n)$, that has zero mean. Thus, the variance of these measurements is given by the relation

$$\sigma_r^2 = \frac{1}{n}\mathbf{r}\mathbf{r}^T \tag{4}$$

If we have a second vector of n measurements, $\mathbf{s} = (s_1, s_2, \cdots, s_n)$, again with zero mean, then we can generalise this idea to obtain the *covariance* of $\mathbf{r}$ and $\mathbf{s}$. Covariance can be thought of as a measure of how much two variables change together. Variance is thus a special case of covariance, when the the two variables are identical. It is in fact correct to divide through by a factor of $n-1$ rather than n, a fact which we shall not justify here.

$$\sigma_{rs}^2 = \frac{1}{n-1}\mathbf{r}\mathbf{s}^T \tag{5}$$

We can now generalise this idea to considering our $m \times n$ data matrix, $\mathbf{X}$. Recall that m was the number of variables, and n the number of samples. We can therefore think of this matrix, $\mathbf{X}$ in terms of m row vectors, each of length n.

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & a_{m,2} & \cdots & x_{m,n} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} \in \mathbb{R}^{m\times n}, \mathbf{x}_1^T \in \mathbb{R}^n \tag{6}$$

Since we have a row vector for each variable, each of these vectors contains all the samples for one particular variable. So for example, $\mathbf{x}_i$ is a vector of the n samples for the i^th variable. It therefore makes sense to consider the following matrix product.

$$\mathbf{C_X} = \frac{1}{1-n}\mathbf{X}\mathbf{X}^T = \frac{1}{1-n}\begin{pmatrix} \mathbf{x}_1\mathbf{x}_1^T & \mathbf{x}_1\mathbf{x}_2^T & \cdots & \mathbf{x}_1\mathbf{x}_n^T \\ \mathbf{x}_2\mathbf{x}_1^T & \mathbf{x}_2\mathbf{x}_2^T & \cdots & \mathbf{x}_2\mathbf{x}_n^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_m\mathbf{x}_1^T & \mathbf{x}_m\mathbf{x}_2^T & \cdots & \mathbf{x}_m\mathbf{x}_n^T \end{pmatrix} \in \mathbb{R}^{m\times n} \tag{7}$$

If we look closely at the entries of this matrix, we see that we have computed all the possible covariance pairs between the m variables. Indeed, on the diagonal entries, we have the variances and on the off-diagonal entries, we have the *covariances.* This matrix is therefore known as the *Covariance Matrix.*

Now let us return to the original problem, that of linearly transforming the original data matrix using the relation $\mathbf{Y}$=$\mathbf{PX}$, for some matrix, $\mathbf{P}$. We need to decide upon some features that we would like the transformed matrix, $\mathbf{Y}$ to exhibit and somehow relate this to the features of the corresponding covariance matrix $\mathbf{C_Y}$.

Covariance can be considered to be a measure of how well correlated two variables are. The PCA method makes the fundamental assumption that the variables in the transformed matrix should be as uncorrelated as possible. This is equivalent to saying that the covariances of different variables in the matrix $\mathbf{C_Y}$, should be as close to zero as possible (covariance matrices are always positive definite or positive semi-definite). Conversely, large variance values interest us, since they correspond to interesting dynamics in the system (small variances may well be noise). We therefore have the following requirements for constructing the covariance matrix, $\mathbf{C_Y}$:

1. Maximise the signal, measured by variance (maximise the diagonal entries)
2. Minimise the covariance between variables (minimise the off-diagonal entries)

We thus come to the conclusion that since the minimum possible covariance is zero, we are seeking a diagonal matrix, $\mathbf{C_Y}$. If we can choose the transformation matrix, $\mathbf{P}$ in such a way that $\mathbf{C_Y}$ is diagonal, then we will have achieved our objective.

We now make the assumption that the vectors in the new basis, $\mathbf{p}_1\mathbf{p}_2,\cdots\mathbf{p}_m$ are orthogonal (in fact, we additionally assume that they are orthonormal). Far from being restrictive, this assumption enables us to proceed by using the tools of linear algebra to find a solution to the problem. Consider the formula for the covariance matrix, $\mathbf{C_Y}$ and our interpretation of $\mathbf{Y}$ in terms of $\mathbf{X}$ and $\mathbf{P}$.

$$\begin{aligned} \mathbf{C_Y} = \frac{1}{1-n}\mathbf{X}\mathbf{X}^T = \frac{1}{1-n}(\mathbf{PX})(\mathbf{PX})^T = \frac{1}{1-n}(\mathbf{PX})(\mathbf{P}^T\mathbf{X}^T) = \frac{1}{1-n}\mathbf{P}(\mathbf{X}\mathbf{X}^T)\mathbf{P}^T \\ i.e. \quad \mathbf{C_Y} = \frac{1}{1-n}\mathbf{PSP}^T \quad where \quad S = \mathbf{X}\mathbf{X}^T \end{aligned} \tag{8}$$

Note that $\mathbf{S}$ is an $m \times m$ symmetric matrix, since $(\mathbf{X}\mathbf{X}^T)^T = (\mathbf{X}^T)^T(\mathbf{X}\mathbf{X}^T) = \mathbf{X}\mathbf{X}^T$. We now invoke the well known theorem from linear algebra that every square symmetric matrix is orthogonally (orthonormally) diagonalisable. That is, we can write:

$$\mathbf{S} = \mathbf{EDE}^T \tag{9}$$

Where**E** is an $m \times m$ orthonormal matrix whose columns are the orthonormal eigenvectors of **S**, and **D** is a diagonal matrix which has the eigenvalues of **S** as its (diagonal) entries. The rank, r, of **S** is the number of orthonormal eigenvectors that it has. If B turns out to be rank-deficient so that r is less than the size, m, of the matrix, then we simply need to generate $m - r$ orthonormal vectors to fill the remaining columns of **S**.

It is at this point that we make a choice for the transformation matrix, **P**. By choosing the *rows* of **P** to be the eigenvectors of **S**, we ensure that $\mathbf{P} = \mathbf{E}^T$ and vice-versa. Thus, substituting this into our derived expression for the covariance matrix, $\mathbf{C_Y}$ gives:

$$\mathbf{C_Y} = \frac{1}{1-n}\mathbf{PSP}^T = \frac{1}{1-n}\mathbf{E}^T(\mathbf{EDE}^T)\mathbf{E} \qquad (10)$$

Now, since **E** is an orthonormal matrix, we have $\mathbf{E}^T\mathbf{E} = \mathbf{I}$, where **I** is the $m \times m$ identity matrix. Hence, for this special choice of **P**, we have:

$$\mathbf{C_Y} = \frac{1}{1-n}\mathbf{D} \qquad (11)$$

A last point to note is that with this method, we automatically gain information about the relative importance of each principal component from the variances. The largest variance corresponds to the first principal component, the second largest to the second principal component, and so on. This therefore gives us a method for organising the data in the diagonalisation stage. Once we have obtained the eigenvalues and eigenvectors of $\mathbf{S} = \mathbf{XX}^T$, we sort the eigenvalues in descending order and place them in this order on the diagonal of **D**. We then construct the orthonormal matrix,**E** by placing the associated eigenvectors in the same order to form the columns of **E** (i.e. place the eigenvector that corresponds to the largest eigenvalue in the first column, the eigenvector corresponding to the second largest eigenvalue in the second column etc.).

We have therefore achieved our objective of diagonalising the covariance matrix of the transformed data. The principal components (the rows of **P**) are the eigenvectors of the covariance matrix, $\mathbf{XX}^T$, and the rows are in order of 'importance', telling us how 'principal' each principal component is.

3.1. The Singular Value Decomposition

In this section, we will examine how the well known singular value decomposition (SVD) from linear algebra can be used in principal component analysis. Indeed, we will show that the derivation of PCA in the previous section and the SVD are closely related. We will not derive the SVD, as it is a well established result, and can be found in any good book on numerical linear algebra.

Given $\mathbf{A} \in \mathbb{R}^{n\times m}$, not necessarily of full rank, a singular value decomposition of **A** is:

$$\mathbf{A} = \mathbf{U\Sigma V}^T \qquad (12)$$

Where

$$\mathbf{U} \in \mathbb{R}^{n\times n} \quad \textit{is orthonormal}$$

$$\mathbf{\Sigma} \in \mathbb{R}^{n\times m} \quad \textit{is diagonal}$$

$$\mathbf{V} \in \mathbb{R}^{m\times m} \quad \textit{is orthonormal}$$

In addition, the diagonal entries, σ_i, of Σ are non-negative and are called the *singular* values of $\mathbf{A}$. They are ordered such that the largest singular value, σ_1 is placed in the (1, 1) entry of Σ, and the other singular values are placed in order down the diagonal, and satisfy $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_p \geq 0$, where $p = min(n, m)$. Note that we have reversed the row and column indexes in defining the SVD from the way they were defined in the derivation of PCA in the previous section. The reason for doing this will become apparent shortly.

Since $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{V} \in \mathbb{R}^{m\times m}$ are orthonormal matrices, their columns form bases for, respectively, the vector spaces $\mathbb{R}^n$ and $\mathbb{R}^m$. Therefore, any vector $\mathbf{b} \in \mathbb{R}^n$ can be expanded in the basis formed by the columns of $\mathbf{U}$ (also known as the *left singular vectors* of $\mathbf{A}$) and any vector $\mathbf{x} \in \mathbb{R}^m$ can be expanded in the basis formed by the columns of $\mathbf{V}$ (also known as the *right singular vectors* of $\mathbf{A}$). The vectors for these expansions $\hat{\mathbf{b}}$ and $\hat{\mathbf{x}}$, are given by:

$$\hat{\mathbf{b}} = \mathbf{U}^T\mathbf{b} \quad \textit{and} \quad \hat{\mathbf{x}} = \mathbf{V}^T\mathbf{x} \tag{13}$$

Now, if the relation $\mathbf{b} = \mathbf{A}\mathbf{x}$ holds, then we can infer the following:

$$\mathbf{U}^T\mathbf{b} = \mathbf{U}^T\mathbf{A}\mathbf{x} \rightarrow \hat{\mathbf{b}} = \mathbf{U}^T(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T)\mathbf{x} \rightarrow \hat{\mathbf{b}} = \boldsymbol{\Sigma}\hat{\mathbf{x}} \tag{14}$$

Thus, the SVD allows us to assert that every matrix is diagonal, so long as we choose the appropriate bases for the domain and range spaces.

How does this link in to the previous analysis of PCA? Consider the $n \times m$ matrix, $\mathbf{A}$, for which we have a singular value decomposition,$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. There is a theorem from linear algebra which says that the non-zero singular values of $\mathbf{A}$ are the square roots of the nonzero eigenvalues of $\mathbf{A}\mathbf{A}^T$ or $\mathbf{A}^T\mathbf{A}$. The former assertion for the case $\mathbf{A}^T\mathbf{A}$ is proven in the following way:

$$\mathbf{A}^T\mathbf{A} = (\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T)^T(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T) = (\mathbf{U}\boldsymbol{\Sigma}^T\mathbf{V}^T)(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T) = \mathbf{V}(\boldsymbol{\Sigma}^T\boldsymbol{\Sigma})\mathbf{V}^T \tag{15}$$

We observe that $\mathbf{A}^T\mathbf{A}$ is similar to $\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$, and thus it has the same eigenvalues. Since $\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$ is a square $(m \times m)$, diagonal matrix, the eigenvalues are in fact the diagonal entries, which are the squares of the singular values. Note that the nonzero eigenvalues of each of the covariance matrices, $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ are actually identical.

It should also be noted that we have effectively performed an eigenvalue decomposition for the matrix, $\mathbf{A}^T\mathbf{A}$. Indeed, since $\mathbf{A}^T\mathbf{A}$ is symmetric, this is an orthogonal diagonalisation and thus the eigenvectors of $\mathbf{A}^T\mathbf{A}$ are the columns of $\mathbf{V}$. This will be important in making the practical connection between the SVD and and the PCA of matrix $\mathbf{X}$, which is what we will do next.

Returning to the original $m \times n$ data matrix, $\mathbf{X}$, let us define a new $n \times m$ matrix, $\mathbf{Z}$:

$$\mathbf{Z} = \frac{1}{\sqrt{1-n}}\mathbf{X}^T \tag{16}$$

Recall that since the m rows of $\mathbf{X}$ contained the n data samples, we subtracted the row average from each entry to ensure zero mean across the rows. Thus, the new matrix, $\mathbf{Z}$ has *columns* with zero mean. Consider forming the $m \times m$ matrix, $\mathbf{Z}^T\mathbf{Z}$:

$$\mathbf{Z}^T\mathbf{Z} = (\frac{1}{\sqrt{1-n}}\mathbf{X}^T)^T(\frac{1}{\sqrt{1-n}}\mathbf{X}^T) = \frac{1}{1-n}\mathbf{X}\mathbf{X}^T$$
$$i.e. \qquad \mathbf{Z}^T\mathbf{Z} = \mathbf{C}_\mathbf{X} \tag{17}$$

We find that defining $\mathbf{Z}$ in this way ensures that $\mathbf{Z}^T\mathbf{Z}$ is equal to the covariance matrix of $\mathbf{X}$, $\mathbf{C}_\mathbf{X}$. From the discussion in the previous section, the principal components of $\mathbf{X}$ (which is what we are trying to identify) are the eigenvectors of $\mathbf{C}_\mathbf{X}$. Therefore, if we perform a singular value decomposition of the matrix $\mathbf{Z}^T\mathbf{Z}$, the principal components will be the columns of the orthogonal matrix, $\mathbf{V}$.

The last step is to relate the SVD of $\mathbf{Z}^T\mathbf{Z}$ back to the change of basis represented by equation:

$$\mathbf{Y} = \mathbf{P}\mathbf{X} \tag{18}$$

We wish to *project* the original data onto the directions described by the principal components. Since we have the relation $\mathbf{V} = \mathbf{P}^T$, this is simply:

$$\mathbf{Y} = \mathbf{V}^T\mathbf{X} \tag{19}$$

If we wish to recover the original data, we simply compute (using orthogonality of $\mathbf{V}$):

$$\mathbf{X} = \mathbf{V}^T\mathbf{Y} \tag{20}$$

3.2. Applications

3.2.1. Quantitative finance

In quantitative finance, principal component analysis can be directly applied to the risk management of interest rate derivative portfolios. Trading multiple swap instruments which are usually a function of 30-500 other market quotable swap instruments is sought to be reduced to usually 3 or 4 principal components, representing the path of interest rates on a macro basis. Converting risks to be represented as those to factor loadings (or multipliers) provides assessments and understanding beyond that available to simply collectively viewing risks to individual 30-500 buckets.

PCA has also been applied to share portfolios in a similar fashion, both to portfolio risk and to risk return. One application is to reduce portfolio risk, where allocation strategies are applied to the "principal portfolios" instead of the underlying stocks. A second is to enhance portfolio return, using the principal components to select stocks with upside potential

3.2.2. Neuroscience

A variant of principal components analysis is used in neuroscience to identify the specific properties of a stimulus that increase a neuron's probability of generating an action potential. This technique is known as spike-triggered covariance analysis. In a typical application an experimenter presents a white noise process as a stimulus (usually either as a sensory input to a test subject, or as a current injected directly into the neuron) and records a train of action potentials, or spikes, produced by the neuron as a result. Presumably, certain features of the stimulus make the neuron more likely to spike. In order to extract these features, the experimenter calculates the covariance matrix of the spike-triggered ensemble, the set of all stimuli (defined and discretized over a finite time window, typically on the order of 100 ms) that immediately preceded a spike. The eigenvectors of the difference between the spike-triggered covariance matrix and the covariance matrix of the prior stimulus ensemble (the set of all stimuli, defined over the same length time window) then indicate the directions in the space of stimuli along which the variance of the spike-triggered ensemble differed the most from that of the prior stimulus ensemble. Specifically, the eigenvectors with the largest positive eigenvalues correspond to the directions along which the variance of the spike-triggered ensemble showed the largest positive change compared to the variance of the prior. Since these were the directions in which varying the stimulus led to a spike, they are often good approximations of the sought after relevant stimulus features.[5]

In neuroscience, PCA is also used to discern the identity of a neuron from the shape of its action potential. Spike sorting is an important procedure because extracellular recording techniques often pick up signals from more than one neuron. In spike sorting, one first uses PCA to reduce the dimensionality of the space of action potential waveforms, and then performs clustering analysis to associate specific action potentials with individual neurons.

PCA as a dimension reduction technique is particularly suited to detect coordinated activities of large neuronal ensembles. It has been used in determining collective variables, i.e. order parameters, during phase transitions in the brain.[5]

4. t-distributed stochastic neighbor embedding

This chapter summarizes the work of L. van der Maaten, and G. Hinton[2].

4.1. Stochastic Neighbor Embedding

Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities. The similarity of datapoint x_j to datapoint x_j is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i. For nearby datapoints, $p_{j|i}$ is relatively high, whereas for widely separated datapoints, $p_{j|i}$ will be almost infinitesimal. Mathematically, the conditional probability $p_{j|i}$ is given by

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-||x_i - x_j||^2/2\sigma_i^2)} \tag{21}$$

where σ_i is the variance of the Gaussian that is centered on datapoint x_i. We set the value of $p_{i|i}$ to zero. For the low-dimensional counterparts y_i and y_j of the high-dimensional datapoints x_i and x_j , it is possible to compute a similar conditional probability, which we denote by $q_{j|i}$. We set the variance of the Gaussian that is employed in the computation of the conditional probabilities $q_{j|i}$. to $\frac{1}{\sqrt{2}}$. Hence, we model the similarity of map point y_j to map point y_i by

$$q_{j|i} = \frac{exp(-||x_i - x_j||^2)}{\sum_{k\neq i} exp(-||x_i - x_j||^2)} \tag{22}$$

We set $q_{j|i} = 0$

If the map points y_i and y_j correctly model the similarity between the high-dimensional datapoints x_i and x_j , the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal. SNE finds a low-dimensional data representation by minimizing the inconsistency between $p_{j|i}$ and $q_{j|i}$. Furthermore, the Kullback–Leibler divergence method is used to measure the degree of accuracy in the way $q_{j|i}$ models $p_{j|i}$. SNE uses a gradient descent method to minimize the sum of Kullback–Leibler divergences over all data points that incurs some cost. The cost function C is given by

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}} \tag{23}$$

in which P_i represents the conditional probability distribution over all other datapoints given datapoint x_i, and Q_i represents the conditional probability distribution over all other map points given map point y_i

The remaining parameter to be selected is the variance σ_i of the Gaussian that is centered over each high-dimensional datapoint, x_i. It is not likely that there is a single value of σ_i that is optimal for all datapoints in the data set because the density of the data is likely to vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Any particular value of σ_i induces a probability distribution, P_i, over all of the other datapoints. This distribution has an entropy which increases as σ_i increases. SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by the user. The perplexity is defined as

$$Prep(P_i) = 2^{H(P_i)}, \tag{24}$$

where $H(P_i)$ is the Shannon entropy of Pi measured in bits

$$H(P_i) = -\sum_i p_{j|i} log_2 p_{j|i} \tag{25}$$

4.2. t-SNE

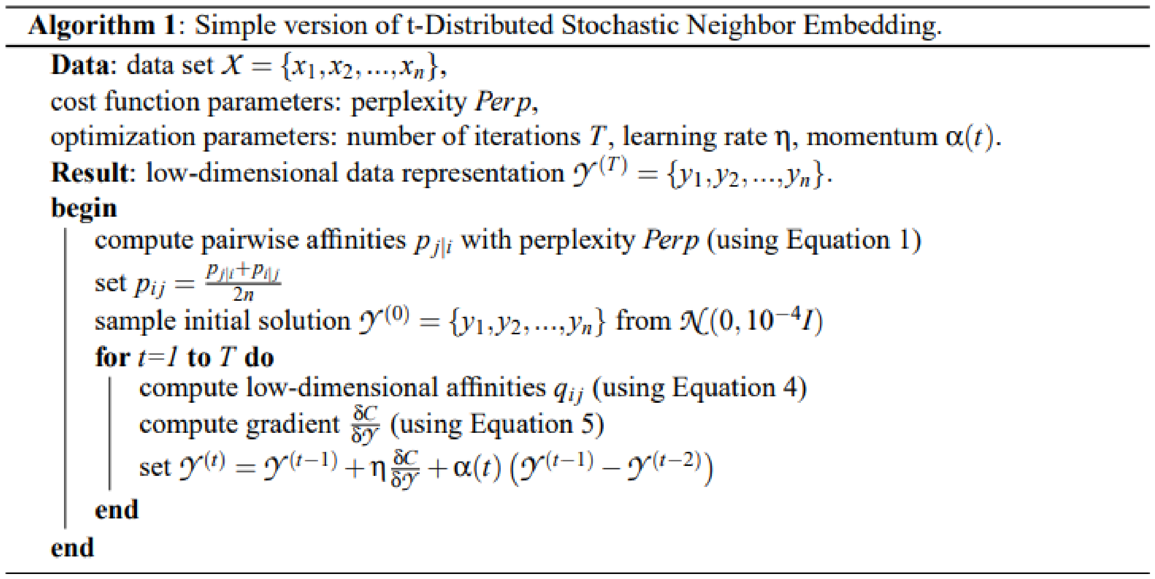
In t-SNE, we employ a Student t-distribution with one degree of freedom (which is the same as a Cauchy distribution) as the heavy-tailed distribution in the low-dimensional map. Using this distribution, the joint probabilities q_{ij} are defined as

$$q_{ij} = \frac{(1 + ||x_i - x_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||x_i - x_j||^2)^{-1}} \tag{26}$$

We use a Student t-distribution with a single degree of freedom, because it has the particularly nice property that $(1 + ||x_i - x_j||^2)^{-1}$ approaches an inverse square law for large pairwise distances. $||x_i - x_j||$ in the low-dimensional map.

The gradient of the Kullback-Leibler divergence between P and the Student-t based joint probability distribution Q(Equation 26) is given by

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||x_i - x_j||^2)^{-1} \tag{27}$$



Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

```
Data: data set $X = \{x_1, x_2, \dots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \dots, y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \dots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for $t$=1 to $T$ do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
    end
end
```

Hivatkozások

[1] Analytics Vidhya, Beginners Guide To Learn Dimension Reduction Techniques. https://www.analyticsvidhya.com/blog/2015/07/dimension-reduction-methods/.

[2] Maaten and Hinton, Visualizing Data using t-SNE. http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[3] Mark Richardson: Principal Component Analysis. http://www.dsc.ufcg.edu.br/~hmg/disciplinas/posgraduacao/rn-copin-2014.3/material/SignalProcPCA.pdf.

[4] WikiPedia, Dimensionality reduction. https://en.wikipedia.org/wiki/Dimensionality_reduction.

[5] WikiPedia, Principal component analysis. https://en.wikipedia.org/wiki/Dimensionality_reduction.