Feature Learning

MInDS @ Mines

Dimensionality reduction is a crucial part of machine learning when dealing with high dimensionality data. Feature selection is a simple approach to working with the available data. Feature extraction can get more complicated where we are transforming the feature space into a new one. We will cover several feature extraction methods including Principal Component Analysis (PCA), kernel PCA, t-distribution stochastic neighbor embedding (t-SNE), and linear discriminant analysis.

Dimensionality reduction is a common solution to deal with the problems that arise due to the curse of dimensionality. Feature extraction is one common approach to dimensionality reduction where we transform the feature space into a new space that achieves the goals we aim to achieve by reducing the data's dimensionality. There are several methods for feature extraction and we will cover some of those next.

Principal Component Analysis (PCA)

Principal component analysis (PCA) is a very popular method for feature extraction that transforms the feature space into a new space using the principal components of the data. Principal components are orthogonal which means that the features are statistically independent. PCA also allows us to move between the two spaces so if we were to determine a new point in the new space, we can transform it back to determine what its features would be in the original space. This reconstruction is critical and so the objective for PCA is to minimize the reconstruction error. The principal components also ensure that each component maximizes the variance of the data along that dimension. The objective for PCA is,

$$\begin{aligned} &\min_{\mathsf{U},\mathsf{V}} ||\mathsf{X} - \mathsf{U}\mathsf{V}^T||_F^2, \\ &s.t.\ \mathsf{U}^T\mathsf{U} = \mathsf{I}. \end{aligned} \tag{1}$$

where X is our $d \times n$ data matrix, U is a $d \times r$ matrix of the principal components, and V is an $n \times r$ matrix of the data in the new principal components' space. If we achieve our goals, the result is that we can approximate,

$$\mathsf{U}\mathsf{V}^T \approx \mathsf{X}. \tag{2}$$

This gives us,

$$\mathsf{U}^T\mathsf{U}\mathsf{V}^T = \mathsf{U}^T\mathsf{X}, \tag{3}$$

$$\mathsf{V}^T = \mathsf{U}^T\mathsf{X}. \tag{4}$$

You may sometimes see the objective of PCA written as $\min_{\mathsf{W}} ||\mathsf{X} - \mathsf{W}\mathsf{W}^T\mathsf{X}||_F^2, s.t.\ \mathsf{W}^T\mathsf{W} = \mathsf{I}.$

V^T gives us the resulting embedding of the data in the new transformed space and U can be referred to as the projection matrix[1] which is the matrix

[1] We often refer to projection matrixes using W and so you may see that used elsewhere.

that projects the original data into the new space. The projection matrix is what allows us to reconstruct the transformed data into the original space as well.

PCA is one method that produces a projection matrix transforming the data from one space into another and back. Some methods exist that are only unidirectional where they can transform data into a new space but can not transform data back into the original space.

Covariance Matrix

We mentioned that part of the goal with PCA is to transform the data into dimensions with the highest variance in each dimension, however this may not be clear since the objective function focuses on reconstruction error. Let's look at how we define the symmetric $d \times d$ covariance matrix, C, as having a value at row j and column k of,

$$c_{j,k} = \frac{1}{n-1}\sum_{i=1}^{n}(x_{i,j} - E[\mathsf{x}_j])(x_{i,k} - E[\mathsf{x}_k]), \tag{5}$$

where, if we were to consider our sample alone, we can determine $E[\mathsf{x}_i]$ as being the mean of column i. The covariance matrix then becomes,

$$\mathsf{C} = \frac{\mathsf{X}^T\mathsf{X}}{n-1}. \tag{6}$$

Useful property 1:

$$||\mathsf{A}||_F^2 = \mathrm{tr}(\mathsf{A}^T\mathsf{A})$$

Useful property 2:

$$\mathrm{tr}(\mathsf{AB}) = \mathrm{tr}(\mathsf{BA})$$

We can approximate the covariance matrix using the Frobenius norm squared of a matrix due to property 1,

$$\mathsf{C} \propto ||\mathsf{X}||_F^2. \tag{7}$$

Back to our original objective function of,

$$\begin{aligned} &\min_{\mathsf{U},\mathsf{V}} ||\mathsf{X} - \mathsf{U}\mathsf{V}^T||_F^2, \\ &s.t.\ \mathsf{U}^T\mathsf{U} = \mathsf{I}. \end{aligned} \tag{8}$$

We can now rewrite our objective,

$$\begin{aligned}
||\mathsf{X} - \mathsf{UV}^T||_F^2 &= \mathrm{tr}((\mathsf{X} - \mathsf{UV}^T)(\mathsf{X} - \mathsf{UV}^T)^T), && \text{Using } ||\mathsf{A}||_F^2 = \mathrm{tr}(\mathsf{A}^T\mathsf{A}). && (9)\\
&= \mathrm{tr}((\mathsf{X} - \mathsf{UV}^T)(\mathsf{X}^T - (\mathsf{UV}^T)^T)) && \text{Using } (\mathsf{A}+\mathsf{B})^T = \mathsf{A}^T + \mathsf{B}^T. && (10)\\
&= \mathrm{tr}((\mathsf{X} - \mathsf{UV}^T)(\mathsf{X}^T - \mathsf{VU}^T)) && \text{Using } (\mathsf{AB})^T = \mathsf{B}^T\mathsf{A}^T. && (11)\\
&= \mathrm{tr}(\mathsf{XX}^T - \mathsf{XVU}^T - \mathsf{UV}^T\mathsf{X}^T + \mathsf{UV}^T\mathsf{VU}^T) && \text{Using } (a-b)(c-d) = ac - ad - bc + bd. && (12)\\
&= \mathrm{tr}(-\mathsf{XVU}^T - \mathsf{UV}^T\mathsf{X}^T + \mathsf{UV}^T\mathsf{VU}^T) && \text{Ignoring constants in objective function.} && (13)\\
&= \mathrm{tr}(-\mathsf{XX}^T\mathsf{UU}^T - \mathsf{UU}^T\mathsf{XX}^T + \mathsf{UU}^T\mathsf{XX}^T\mathsf{UU}^T) && \text{Using } \mathsf{V}^T = \mathsf{U}^T\mathsf{X}, \mathsf{V} = \mathsf{X}^T\mathsf{U}. && (14)\\
&= \mathrm{tr}(-\mathsf{U}^T\mathsf{XX}^T\mathsf{U} - \mathsf{U}^T\mathsf{XX}^T\mathsf{U} + \mathsf{U}^T\mathsf{XX}^T\mathsf{UU}^T\mathsf{U}) && \text{Using } \mathrm{tr}(\mathsf{AB}) = \mathrm{tr}(\mathsf{BA}) && (15)\\
&= \mathrm{tr}(-\mathsf{U}^T\mathsf{XX}^T\mathsf{U} - \mathsf{U}^T\mathsf{XX}^T\mathsf{U} + \mathsf{U}^T\mathsf{XX}^T\mathsf{U}) && \text{Using } \mathsf{U}^T\mathsf{U} = \mathsf{I}. && (16)\\
&= \mathrm{tr}(-\mathsf{U}^T\mathsf{XX}^T\mathsf{U}) && \text{Using summation.} && (17)\\
&= -\mathrm{tr}(\mathsf{U}^T\mathsf{XX}^T\mathsf{U}) && \text{Using } \mathrm{tr}(-\mathsf{A}) = -\mathrm{tr}(\mathsf{A}) && (18)\\
&= -\mathrm{tr}(\mathsf{U}^T\mathsf{X}(\mathsf{U}^T\mathsf{X})^T) && \text{Using } (\mathsf{AB})^T = \mathsf{B}^T\mathsf{A}^T. && (19)\\
&= -||\mathsf{U}^T\mathsf{X}||_F^2 && \text{Using } ||\mathsf{A}||_F^2 = \mathrm{tr}(\mathsf{A}^T\mathsf{A}). && (20)
\end{aligned}$$

This results in,

$$\begin{aligned}
&\min_{\mathsf{U},\mathsf{V}} \; ||\mathsf{X} - \mathsf{U}\mathsf{V}^T||_F^2, && s.t.\ \mathsf{U}^T\mathsf{U} = \mathsf{I}. && (21)\\
=&\min_{\mathsf{U}} \; -||\mathsf{U}^T\mathsf{X}||_F^2, && s.t.\ \mathsf{U}^T\mathsf{U} = \mathsf{I}. && (22)\\
=&\max_{\mathsf{U}} \; ||\mathsf{U}^T\mathsf{X}||_F^2, && s.t.\ \mathsf{U}^T\mathsf{U} = \mathsf{I}. && (23)\\
\approx&\max_{\mathsf{U}} \; cov(\mathsf{U}^T\mathsf{X}), && s.t.\ \mathsf{U}^T\mathsf{U} = \mathsf{I}. && (24)
\end{aligned}$$

This portion covers the intuition behind what PCA aims to do with its result. Next, we'll cover how we actually solve the objective function.

Singular Value Decomposition

With a symmetric matrix we can apply eigenvalue decomposition [2], and with a non-symmetric matrix we can apply singular value decomposition (SVD). SVD allows us to decompose a matrix X into,

[2] We can apply eigenvalue decomposition to a symmetric matrix, A, resulting in

$$\mathsf{A} = \mathsf{Q}\Lambda\mathsf{Q}^T,$$

where Q is a matrix of eigenvector columns and Λ is a diagonal of the eigenvalues of A.

$$\mathsf{X} = \mathsf{U}\mathsf{S}\mathsf{V}^T, \tag{25}$$

where X is a $d \times n$ matrix, U is a unitary $d \times d$ matrix, S is a diagonal matrix of singular values, which are the square root of the respective eigenvalues, and V is a unitary $n \times n$ matrix.

A unitary matrix U is one where $\mathsf{U}^T\mathsf{U} = \mathsf{U}\mathsf{U}^T = \mathsf{I}$.

We can represent the covariance matrix in terms of this decomposition as,

$$\begin{aligned}
\mathsf{C} &= \frac{\mathsf{X}^T\mathsf{X}}{n-1}, && (26)\\
&= \frac{\mathsf{V}\mathsf{S}\mathsf{U}^T\mathsf{U}\mathsf{S}\mathsf{V}^T}{n-1}, && (27)\\
&= \mathsf{V}\frac{\mathsf{S}^2}{n-1}\mathsf{V}^T. && (28)
\end{aligned}$$

We can also decompose the covariance matrix into,

$$\mathsf{C} = \mathsf{V}\Lambda\mathsf{V}^T, \tag{29}$$

where V contains the eigenvector columns and Λ is the diagonal of eigenvalues of C.

The eigenvalues of the covariance matrix can be calculated as the square of the singular values of the original matrix,

$$\frac{\mathsf{S}^2}{n-1} = \Lambda. \tag{30}$$

We can calculate the principal components of a matrix by calculating the eigenvectors of its covariance matrix with the largest r eigenvalues,

$$\mathsf{U} = eig(\mathsf{C}, r). \tag{31}$$

The eigenvectors of the covariance matrix with the largest eigenvalues are the directions at which the data has the largest variance. This shows us that the

Victor Powell and Lewis Lehe created a great demo of PCA in action that explains it visually.

principal components solve two related objectives of maximizing the variance per component and minimizing the reconstruction error. With PCA, we can transform our features into a new space with independent dimensions that can be projected back into the original space with minimal error.

Kernel PCA

Based on what we've already covered, PCA can achieve great results in various cases of data. However, PCA uses linear transformations in order to determine the new space. This aspect of PCA can be limiting. Similar to other linear methods, we can apply a kernel to our data and allow our model to function in non-linear space based on the applied kernel. This leads us to kernel PCA.

Kernel PCA applies feature transformations to the data before projecting it into a new space. Kernel PCA maximizes the variance described by each principal component of the unknown space projected into by the selected kernel. This approach allows for non-linear dimensionality reduction due to the unknown space possessing non-linear properties in the original space.

To better understand this problem, we can redefine the covariance matrix based on the feature transformations,

$$\mathsf{C} = \phi(\mathsf{X})^T \phi(\mathsf{X}). \tag{32}$$

The covariance matrix here is not a $d \times d$ matrix since the kernel dimensions are not in d dimensions. It is in fact an $m \times m$ matrix, where m is the number of dimensions in the unknown kernel space, where $m \gg d$. We can solve for U here again by calculating the eigenvectors of the largest r eigenvalues of the covariance matrix.

With kernel PCA, we can use a variety of kernels. Some popular ones are linear, polynomial and radial basis function,

$$\underset{Linear}{K}(\mathsf{x}_i, \mathsf{x}_j) = \mathsf{x}_i^T \mathsf{x}_j + 1, \tag{33}$$

which is a special case of polynomial kernel with degree, $d = 1$,

$$\underset{Polynomial}{K}(\mathsf{x}_i, \mathsf{x}_j) = (\mathsf{x}_i^T \mathsf{x}_j + 1)^d, \tag{34}$$

with degree d,

$$\underset{RBF}{K}(\mathsf{x}_i, \mathsf{x}_j) = \exp(\frac{-||\mathsf{x}_i - \mathsf{x}_j||^2}{2\sigma^2}), \tag{35}$$

with width σ.

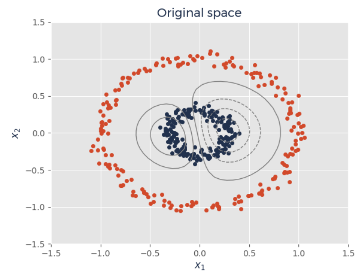

Figure 1: Example data to be used with kernel PCA.

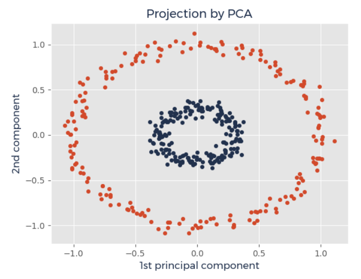

Figure 2: Example data projected using PCA. We can see that PCA isn't able to perform well in separating the data given the need for a non-linear transformation.

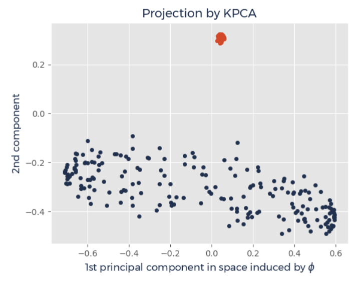

Figure 3: Example data projected using kernel PCA.

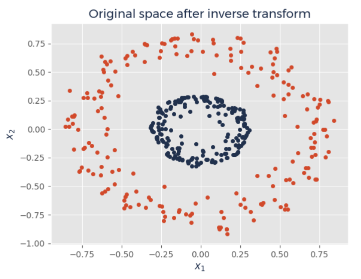

Figure 4: Example data in original space after transformation and inverse transformation with kernel PCA.