What Is Dimension Reduction In Data Science?

Farhad Malik Follow

Dec 21, 2018 · 7 min read ★

We have access to a large amounts of data now. The large amount of data can lead us to situations where by we take every possible data that is available to us and feed it into a forecasting model to predict our target variable. This article aims to explain the common issues associated with introduction of large set of features and provides solutions which we can utilise to resolve those problems.

It is crucial for every data scientist and machine learning expert to understand what dimension reduction techniques are and when to use them.

Photo by Sergi Kabrera on Unsplash

Let's Understand The Issues Better

Occasionally we gather data for our data science project and end up gathering a large set of features. Some of these features (known as variables) are not as important as others. Sometimes the features themselves are correlated with each other. And occasionally we end up over-fitting the problem by introducing too many features. The large number of features make the data set sparse.

Furthermore, it takes a much larger space to store a data set with a large number of features. Moreover, it can get very difficult to analyse and visualize a data set with a large number of dimensions.

> *Dimension reduction can reduce the time that is required to train our machine learning model and it can also benefit in eliminating over-fitting.*

This article outlines the techniques which we can follow to compress our data set onto a new feature subspace of lower dimensionality. I will also be providing details of important dimension reduction techniques.

Please read FinTechExplained disclaimer.

How Do I Define Dimension Reduction?

Imagine you want to e-mail a large set of files to your friend. Uploading and sending the files might take a longer time. You can speed up the process of uploading of the files by zipping the files and e-mail the zipped file instead. Zipping the file compresses large quantity of data into smaller equivalent sets.

> *Dimension reduction is the same principal as zipping the data.*

> *Dimension reduction compresses large set of features onto a new feature subspace of lower dimensional without losing the important information.*

Although the slight difference is that dimension reduction techniques will lose some of the information when the dimensions are reduced.

It is harder to visualise a large set of dimensions. Dimension reduction techniques can be employed to make a 20+ dimension feature space into 2 or 3 dimension subspace.

What Are Different Dimension Reduction Techniques?

Before we take a deep dive into the key techniques, let's quickly understand the two main areas of machine learning:

1. Supervised—when the results of the training set are known
2. Unsupervised—when the final outcome is not known

If you want to get a better understanding of machine learning then have a look at my article:

There are a large number of techniques to reduce the dimensions such as forward/backward feature selection or combining the dimensions together by calculating weighted average of the correlated features. However in this article I will explore two of the main techniques of dimension reduction:

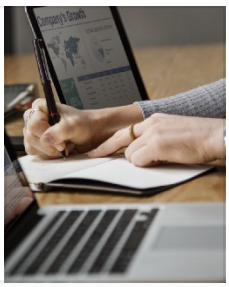
Linear Discriminant Analysis (LDA):

LDA is used for compressing supervised data

When we have a large set of features (classes), and our data is normally distributed and the features are not correlated with each other then we can use LDA to reduce the number of dimensions. LDA is a generalised version of Fisher's linear discriminant.

Calculate z-score to normalise the features that are highly skewed.

If you want to understand how to enrich features and calculate z-score then have a look at this article:



Sci-kit learn offers easy to use LDA tools:

```
from sklearn.lda import LDA
my_lda = LDA(n_components=3)
lda_components = my_lda.fit_transform(X_train, Y_train)
```

This code will result in producing three LDA components for the entire data set.

Photo by NASA on Unsplash

Principal component analysis (PCA):

They are mainly used for compressing unsupervised data.

PCA is a very useful technique that can help de-noise and detect patterns in data. PCA is used in reducing dimensions in images, textual contents and in speech recognition systems.

Sci-kit learn library offers a powerful PCA component classifier. This code snippet illustrates how to create PCA components:

```
from sklearn.decomposition import PCA
pca_classifier = PCA(n_components=3)
my_pca_components = pca_classifier.fit_transform(X_train)
```

It is wise to understand how PCA works.

Understanding PCA

This section of the article provides an overview of the process:

- PCA technique analyses the entire data set and then finds the points with maximum variance.
- It creates new variables such that there is a linear relationship between the new and original variables such that the variance is maximised.
- Covariance matrix is then created for the features to understand their multi-collinearity.
- Once the variance-covariance matrix is computed, PCA then uses the gathered information to reduce the dimensions. It computes orthogonal axes from the original feature axes. These are the axes of directions with maximum variance.

Firstly the eigenvectors of the variance-covariance matrix are calculated. The vector represents the directions of maximum variance which are known as the principal components. The eigenvalues are then created that define magnitude of the principal components.

The eigenvalues are the PCA components.

Therefore, for N dimensions, there will be a NxN variance-covariance matrix and as a result, we will have a eigen vector of N values and N eigen values matrix.

We can use following python modules to create the components:

```
Use linalg.eig to create eigen vectors
Use numpy.cov to compute variance-covariance matrix
```

We need to take the eigen vectors that represent the our data set best. These are the vectors which we have highest eigenvalues.

> *Take eigen vectors that capture about 70% of the variance.*

> *Remember eigenvectors with largest eigenvalues are the ones with highest variance and they are closest to the original data set. Also larger the number of eigenvectors, slower the computation performance.*

I normally take 2–3 top eigen vectors to represent the data set.

If we want to keep sci-kit learn to give us all of the PCA components so that we can assess the variance then initialise PCA with None components:

Photo by Chang Qing on Unsplash

It is important to normalise/standardise the data before performing PCA because PCA is sensitive to the scale of the data in the features.

Kernel principal component analysis (KDA):

They are used for Nonlinear dimensionality reduction

When we have non-linear features then we can project them onto a larger feature set to remove their correlations and to make them linear.

Essentially, non-linear data is mapped and transformed onto a higher-dimensional space. Then PCA is used to reduce the dimensions. However, one downside of this approach is that it is computationally very expensive.

Just like in PCA, we first compute variance-covariance matrix and then eigen vectors and eigen values are prepared with the highest variance to compute principal components.

We then compute kernel matrix. This requires us to construct a similarity matrix. The matrix is then decomposed via creating eigen values and eigen vectors.

Sci-Kit learn offers Kernal PCA modules. To use Kernal PCA, we can use following snippet of code:

```
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components=2,kernel='rbf', gamma=45)
kpca_components = kpca.fit_transform(X)
```

Gamma is a tuning parameter of the RBF kernel.

Photo by Billy Huynh on Unsplash

Benefits Of Dimension Reduction

This section briefly outlines the core benefits of reducing dimensions.

We have access to a large set of data now. When we are building forecasting models that are trained on images, sound and/or textual contents then the input feature sets can end up having a large set of features. It increases space, further adds over-fitting and slows down

the time to train the models. Occasionally features are introduced that end up adding more noise than expected.

One of the key methodologies to improve efficiency in computational intensive tasks is to reduce the dimensions after ensuring most of the key information is maintained. It also eliminates features with strong correlation between them and reduces over-fitting.

Summary

This article provided an overview of the techniques which we can follow to compress our data set onto a new feature subspace of lower dimensionality. It also provided details of important dimension reduction techniques.

Lastly the benefits of dimension reductions were summarised.

Please let me know if there are any questions.