What is Dimensionality Reduction?

Dimensionality reduction is simply the process of reducing the dimension of your feature set. Your feature set could be a dataset with a hundred columns (i.e features) or it could be an array of points that make up a large sphere in the three-dimensional space. Dimensionality reduction is bringing the number of columns down to say, twenty or converting the sphere to a circle in the two-dimensional space.

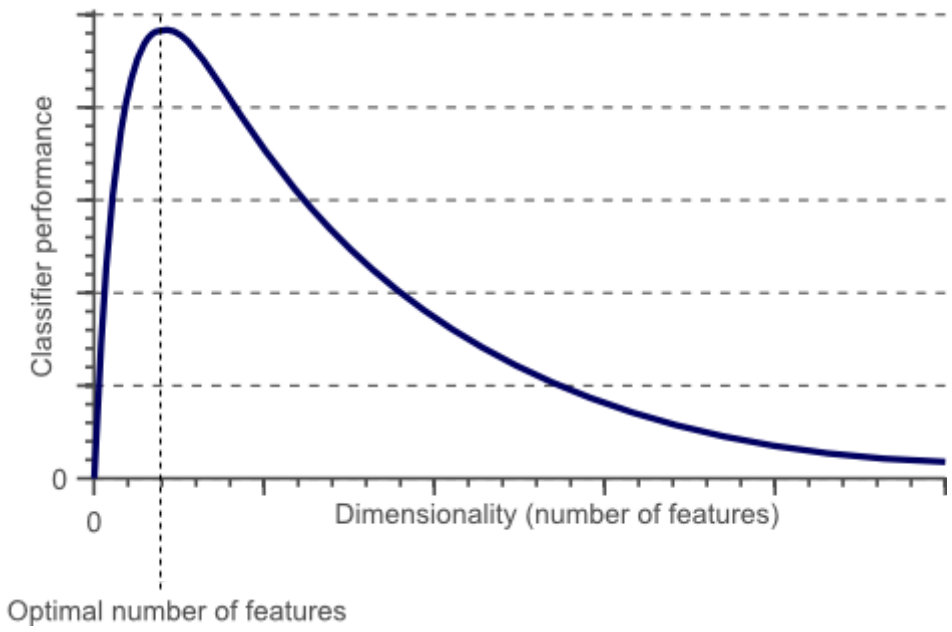
That is all well and good but why should we care? Why would we drop 80 columns off our dataset when we could straight up feed it to our machine learning algorithm and let it do the rest?

The Curse of Dimensionality

We care because the curse of dimensionality demands that we do. The curse of dimensionality refers to all the problems that arise when working with data in the higher dimensions that did not exist in the lower dimensions.

As the number of features increase, the number of samples also increases proportionally. The more features we have, the more samples we will need to have all combinations of feature values well represented in our sample.

As the number of features increases, the model becomes more complex. The more the number of features, the more the chances of overfitting. A machine learning model that is trained on a large number of features, gets increasingly dependent on the data it was trained on and in turn overfitted, resulting in poor performance on real data, beating the purpose.



Avoiding overfitting is a major motivation for performing dimensionality reduction. The fewer features our training data has, the lesser

assumptions our model makes and the simpler it will be. But that is not all and dimensionality reduction has a lot more advantages to offer, like

1. Less misleading data means model accuracy improves.
2. Less dimensions mean less computing. Less data means that algorithms train faster.
3. Less data means less storage space required.
4. Less dimensions allow usage of algorithms unfit for a large number of dimensions
5. Removes redundant features and noise.

Feature Selection and Feature Engineering for dimensionality reduction

Dimensionality reduction could be done by both feature selection methods as well as feature engineering methods.

Feature selection is the process of identifying and selecting relevant features for your sample. Feature engineering is manually generating new features from existing features, by applying some transformation or performing some operation on them.

Feature selection can be done either manually or programmatically. For example, consider you are trying to build a model which predicts people's weights and you have collected a large corpus of data which describes each person quite thoroughly. If you had a column that described the color of each

person's clothing, would that be much help in predicting their weight? I think we can safely agree it won't be. This is something we can drop without further ado. What about a column that described their heights? That's a definite yes. We can make these simple manual feature selections and reduce the dimensionality when the relevance or irrelevance of certain features are obvious or common knowledge. And when it's not glaringly obvious, there are a lot of tools we could employ to aid our feature selection.

1. Heatmaps that show the correlation between features is a good idea.
2. So is just visualising the relationship between the features and the target variable by plotting each feature against the target variable.

Now let us look at a few programmatic methods for feature selection from the popular machine learning library sci-kit learn, namely,

1. Variance Threshold and
2. Univariate selection.

Variance Threshold is a baseline approach to feature selection. As the name suggests, it drops all features where the variance along the column does not exceed a threshold value. The premise is that a feature which doesn't vary much within itself, has very little predictive power.

```
>>> X = [[0, 2, 0, 3], [0, 1, 4, 3], [0, 1, 1, 3]]
>>> selector = VarianceThreshold()
>>> selector.fit_transform(X)
array([[2, 0],
       [1, 4],
      [1, 1]])
```

Univariate Feature Selection uses statistical tests to select features. Univariate describes a type of data which consists of observations on only a single characteristic or attribute. Univariate feature selection examines each feature individually to determine the strength of the relationship of the feature with the response variable. Some examples of statistical tests that can be used to evaluate feature relevance are Pearson Correlation, Maximal information coefficient, Distance correlation, ANOVA and Chi-square. Chi-square is used to find the relationship between categorical variables and Anova is preferred when the variables are continuous.

Scikit-learn exposes feature selection routines likes SelectKBest, SelectPercentile or GenericUnivariateSelect as objects that implement a transform method based on the score of anova or chi2 or mutual information. Sklearn offers f_regression and mutual_info_regression as the scoring functions for regression and f_classif and mutual_info_classif for classification.

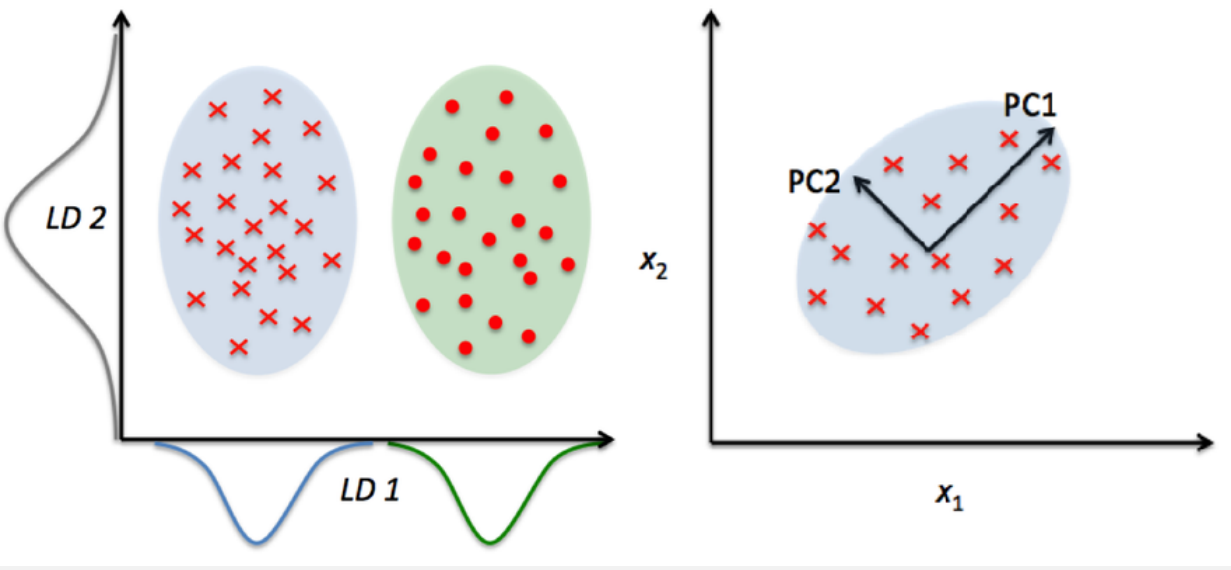
F-Test checks for and only captures linear relationships between features and labels. A highly correlated feature is given higher score and less correlated features are given lower score. Correlation is highly deceptive as it doesn't capture strong non-linear relationships. On the other hand, mutual information methods can capture any kind of statistical dependency, but being nonparametric, they require more samples for accurate estimation.

Feature selection is the simplest of dimensionality reduction methods. We will look at a few feature engineering methods for dimensionality reduction later.

Linear Dimensionality Reduction Methods

The most common and well known dimensionality reduction methods are the ones that apply linear transformations, like

1. PCA (Principal Component Analysis) : Popularly used for dimensionality reduction in continuous data, PCA rotates and projects data along the direction of increasing variance. The features with the maximum variance are the principal components.
2. Factor Analysis : a technique that is used to reduce a large number of variables into fewer numbers of factors. The values of observed data are expressed as functions of a number of possible causes in order to find which are the most important. The observations are assumed to be caused by a linear transformation of lower dimensional latent factors and added Gaussian noise.
3. LDA (Linear Discriminant Analysis): projects data in a way that the class separability is maximised. Examples from the same class are put closely together by the projection. Examples from different classes are placed far apart by the projection



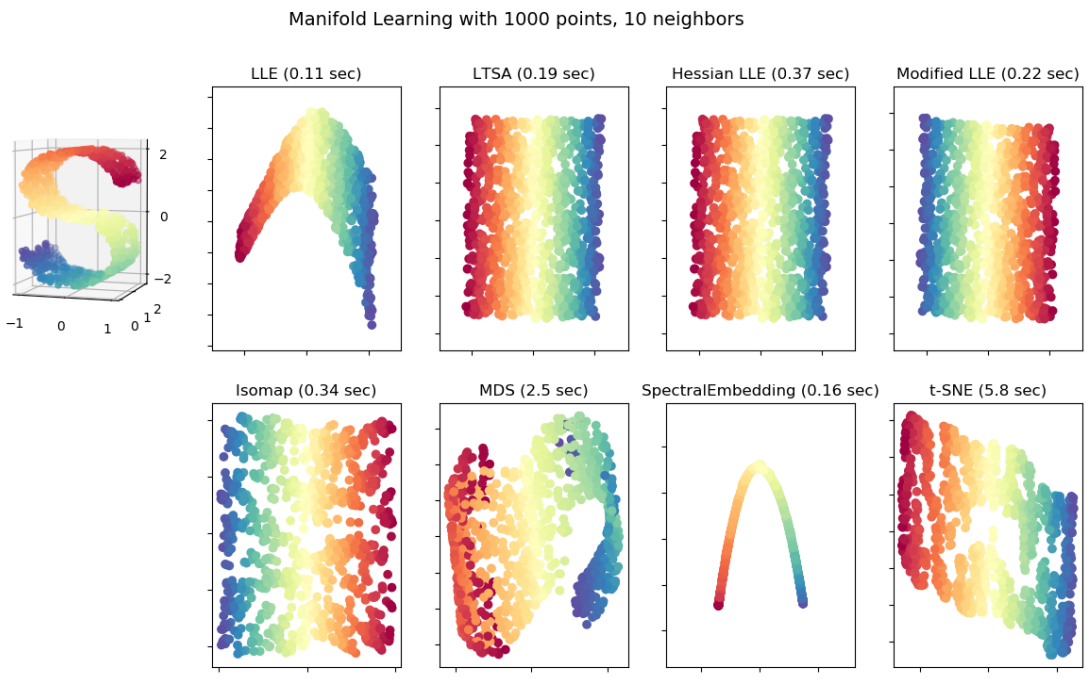
PCA orients data along the direction of the component with maximum variance whereas LDA projects the data to signify the class separability

Non-linear Dimensionality Reduction Methods

Non-linear transformation methods or manifold learning methods are used when the data doesn't lie on a linear subspace. It is based on the manifold hypothesis which says that in a high dimensional structure, most relevant information is concentrated in a small number of low dimensional manifolds. If a linear subspace is a flat sheet of paper, then a rolled up sheet of paper is a simple example of a nonlinear manifold. Informally, this is called a Swiss roll, a canonical problem in the field of non-linear dimensionality reduction. Some popular manifold learning methods are,

1. Multidimensional scaling (MDS) : A technique used for analyzing similarity or dissimilarity of data as distances in geometric spaces. Projects data to a lower dimension such that data points that are close to each other (in terms of Euclidean distance) in the higher dimension are close in the lower dimension as well.

2. Isometric Feature Mapping (Isomap) : Projects data to a lower dimension while preserving the geodesic distance (rather than Euclidean distance as in MDS). Geodesic distance is the shortest distance between two points on a curve.
3. Locally Linear Embedding (LLE): Recovers global non-linear structure from linear fits. Each local patch of the manifold can be written as a linear, weighted sum of its neighbours given enough data.
4. Hessian Eigen Mapping (HLLE): Projects data to a lower dimension while preserving the local neighbourhood like LLE but uses the Hessian operator to better achieve this result and hence the name.
5. Spectral Embedding (Laplacian Eigenmaps): Uses spectral techniques to perform dimensionality reduction by mapping nearby inputs to nearby outputs. It preserves locality rather than local linearity
6. t-distributed Stochastic Neighbor Embedding (t-SNE): Computes the probability that pairs of data points in the high-dimensional space are related and then chooses a low-dimensional embedding which produces a similar distribution.

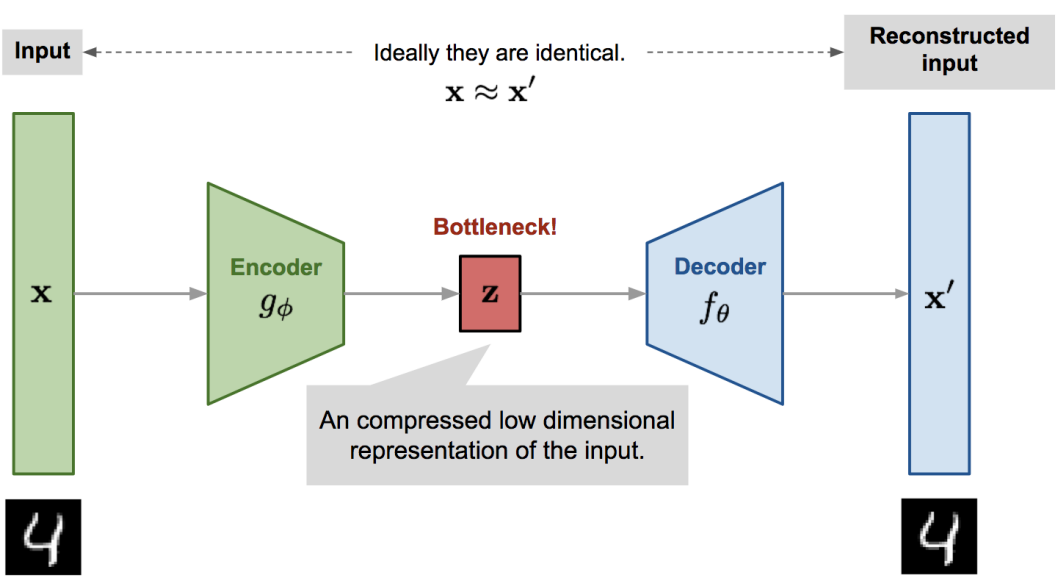


Shows the resulting projection from applying different manifold learning methods on a 3D S-Curve

Auto-encoders

Another popular dimensionality reduction method that gives spectacular results are auto-encoders, a type of artificial neural network that aims to copy their inputs to their outputs. They compress the input into a **latent-space representation**, and then reconstruct the output from this representation. An autoencoder is composed of two parts :

1. **Encoder:** compresses the input into a latent-space representation.
2. **Decoder:** reconstruct the input from the latent space representation.



In subsequent posts, let us look more deeply into linear and non-linear dimensionality reduction methods.