Dimensionality Reduction: Principal Component Analysis

Glenn Bruns
CSUMB

Many figures in this deck from Géron, Hands-on Machine Learning with Scikit-Learn and TensorFlow

Learning outcomes

After this lecture you should be able to:

1. List some ML problems with high-dimension feature spaces
2. Explain some problems in applying ML to high-dimension feature spaces
3. Define dimensionality reduction
4. Explain the concept of principal component analysis, and use it in Scikit-Learn

High-dimension data in machine learning

Text mining:
text classification,
sentiment analysis, …

Document 1

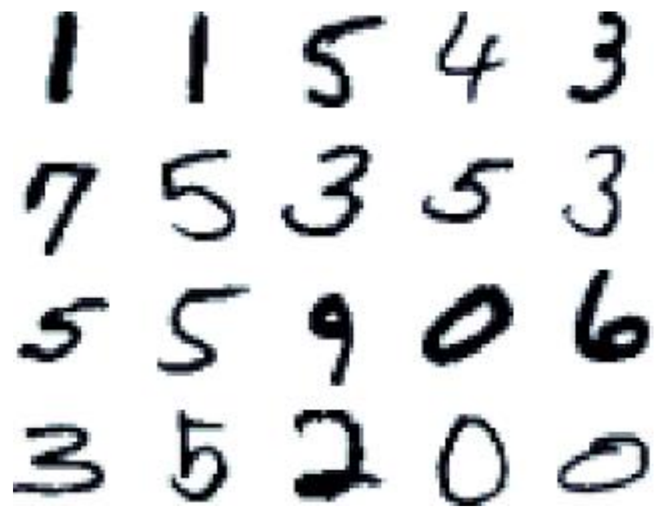
Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Each word a feature

Image processing:
character recognition,
gesture recognition, …



Each pixel a feature

Issues with high-dimension data

Computational:

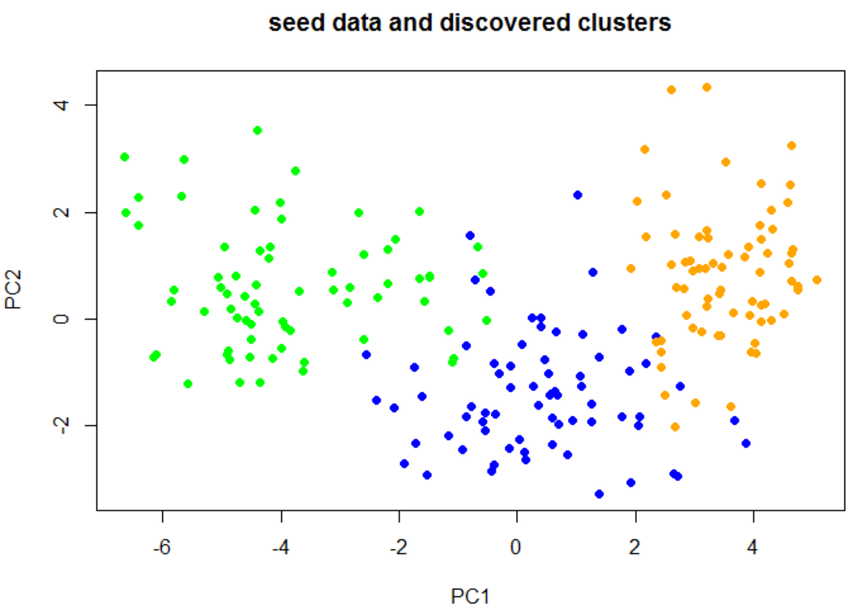
Training becomes very slow when there are thousands or millions of features.

Recall: linear regression using the normal equation has time complexity of about $O(n^3)$

Problems can also arise when the number of features is large relative to the number of instances.

Visual:

How to visualize data of more than 2 or 3 dimensions?



More fundamental issues

High-dimensional space is bizarre.

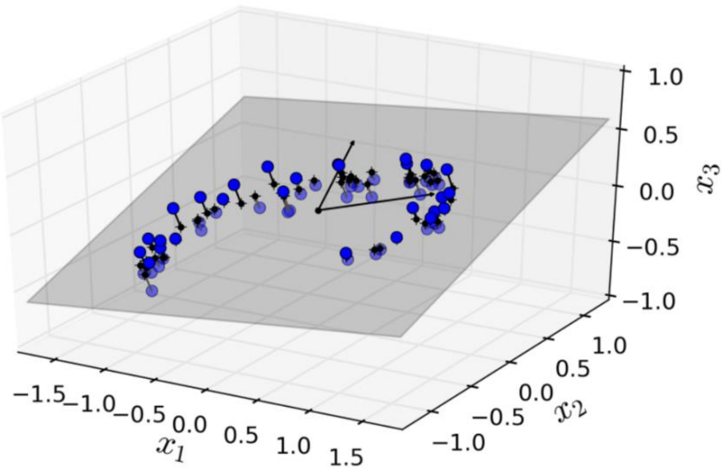
- in a unit square, the chance of a random point being less than 0.001 from a border is 0.4%
- in a 10,000-dimensional unit hypercube, the chance is … > 99.999999%
- if you pick two random points in a unit square, the distance between will be, on average, about 0.52.
- in a 1,000,000-dimensional hypercube, the distance will be about …. 408.

Feature selection and dim. reduction

What to do when there is a large number of features?

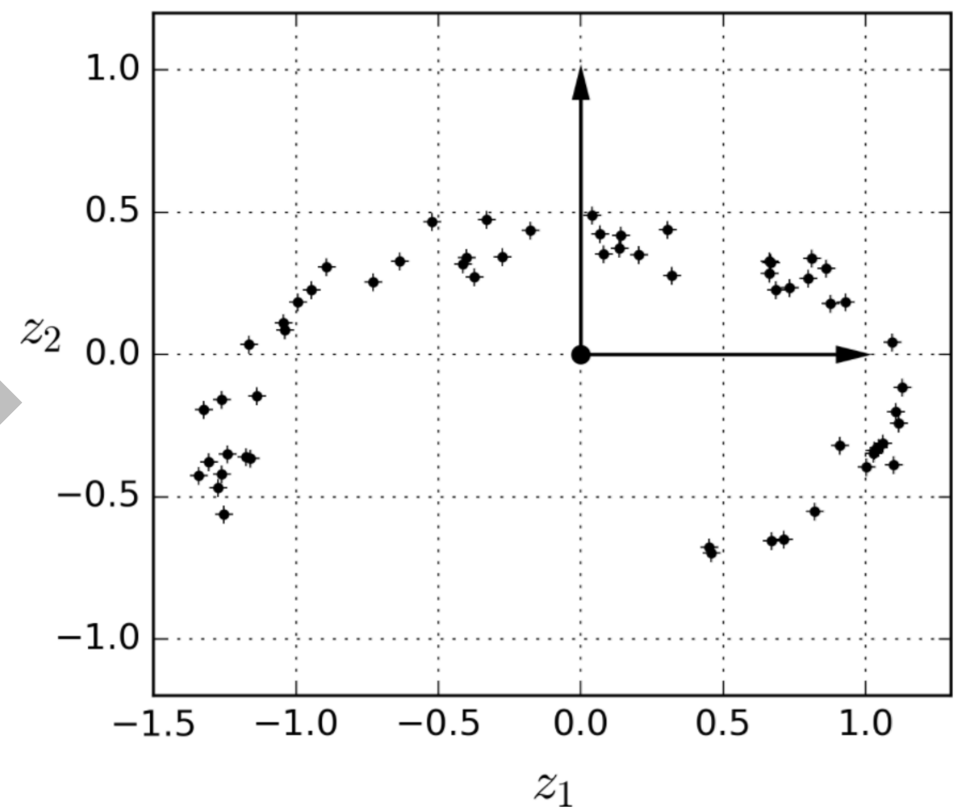
1. Somehow rank the features by “importance”, and keep only the best features
   - statistical feature tests (e.g. relief)
   - feature correlation
   - tests related to predictor performance (e.g. forward selection)
2. Transform the feature space to a lower-dimensional space

Projection



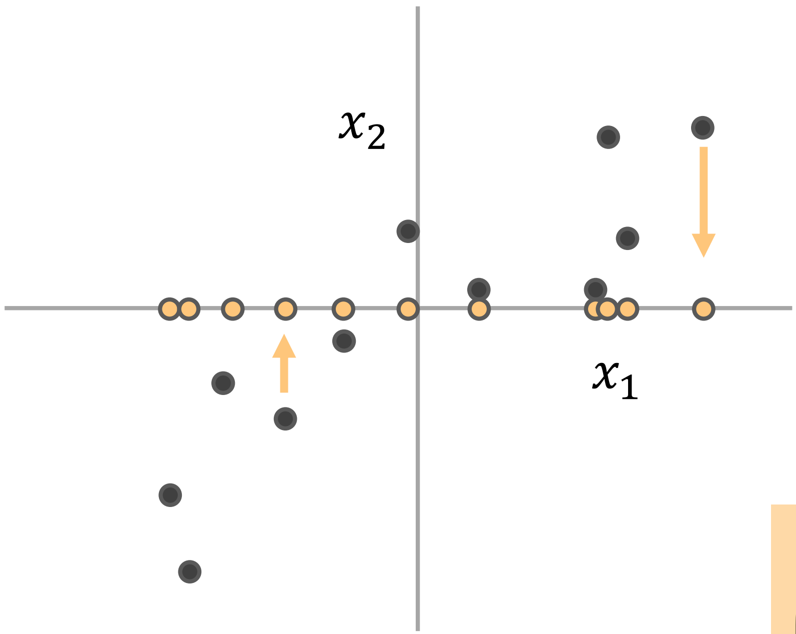
Original feature space

Transformed feature space



- fewer dimensions
- none of original features are present

Dim. reduction by projection

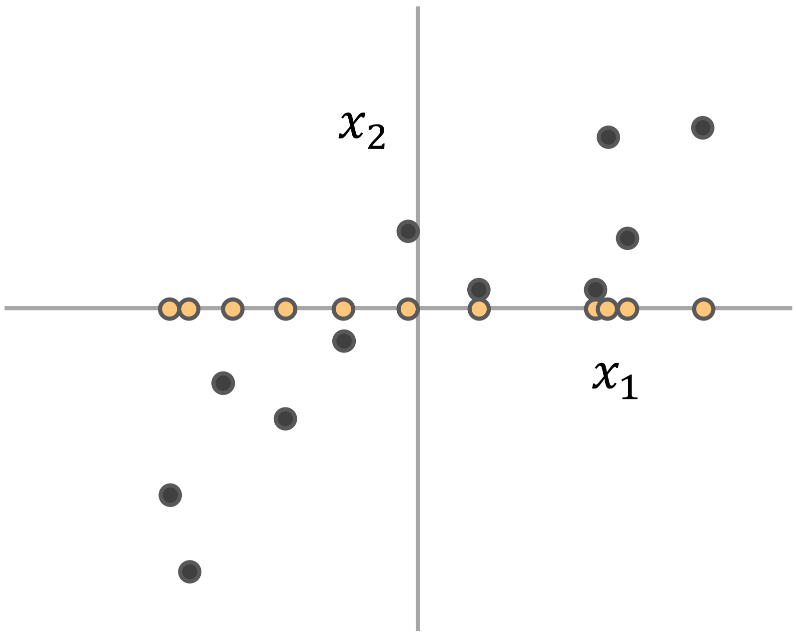


Here is a 2D features space, with features x_1 and x_2

To make the data 1-dimensional, we could project the values onto the first dimension

$$\begin{pmatrix} -4 & -3 \\ -3.5 & -4 \\ -3 & -1 \\ -2.5 & -1.5 \\ -2 & -.5 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} -4 & 0 \\ -3.5 & 0 \\ -3 & 0 \\ -2.5 & 0 \\ -2 & 0 \end{pmatrix}$$

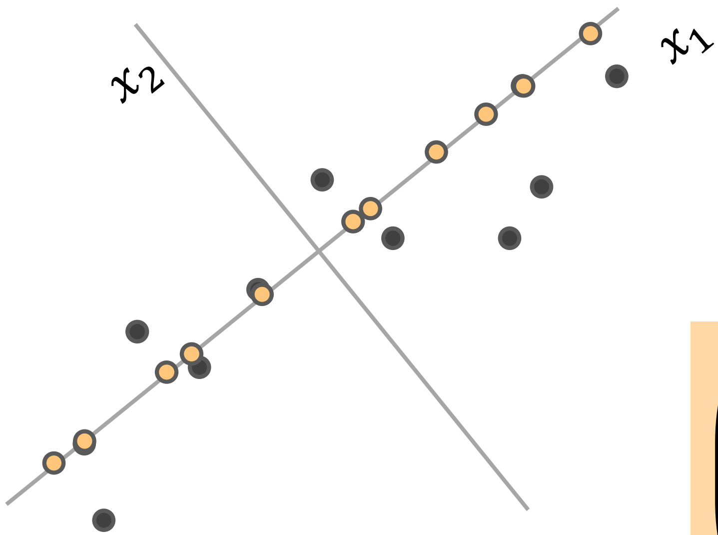
Dim. reduction by projection



Now we have 1D data, but is it the best 1D data we could create?

Dim. reduction by projection

What if we rotate the coordinate system and then project?



$$\begin{pmatrix} -4 & -3 \\ -3.5 & -4 \\ -3 & -1 \\ -2.5 & -1.5 \\ -2 & -.5 \end{pmatrix} \begin{pmatrix} a & 0 \\ c & 0 \end{pmatrix} = \begin{pmatrix} -5 & 0 \\ -4.7 & 0 \\ -3.2 & 0 \\ -2.7 & 0 \\ -2.2 & 0 \end{pmatrix}$$

Principal component analysis

Main idea:

- The single best axis is the one with most variance
- The next best axis is the one, of those orthogonal to the best axis, with the most variance

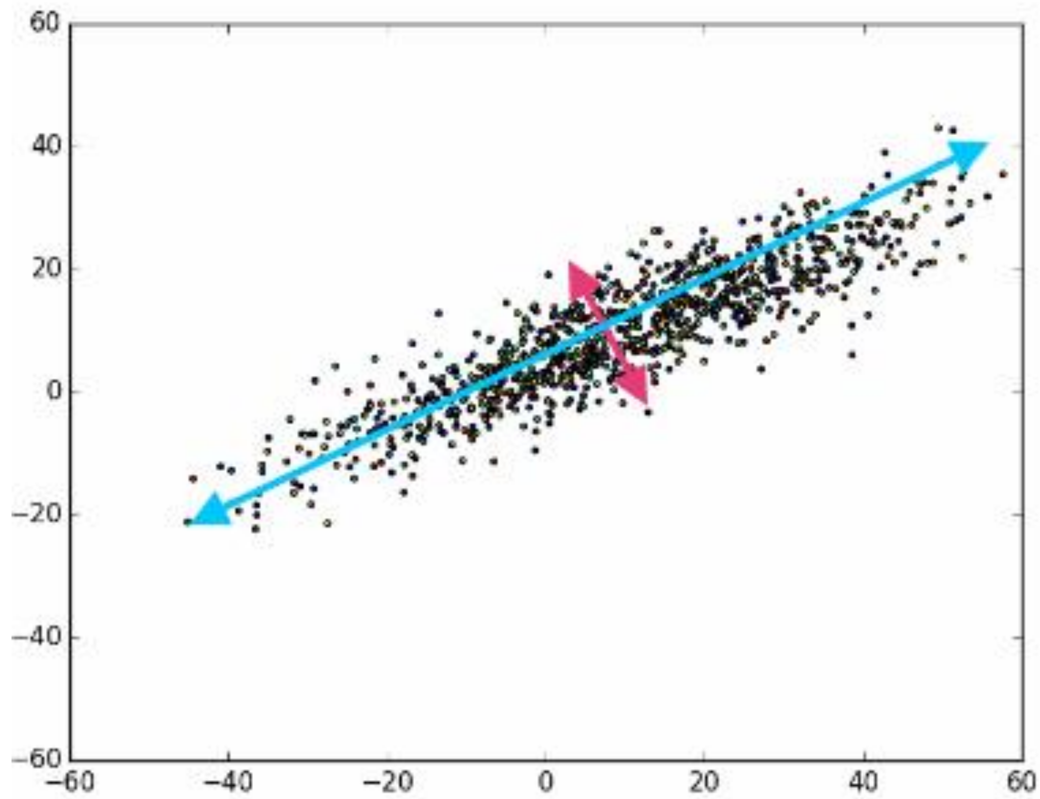
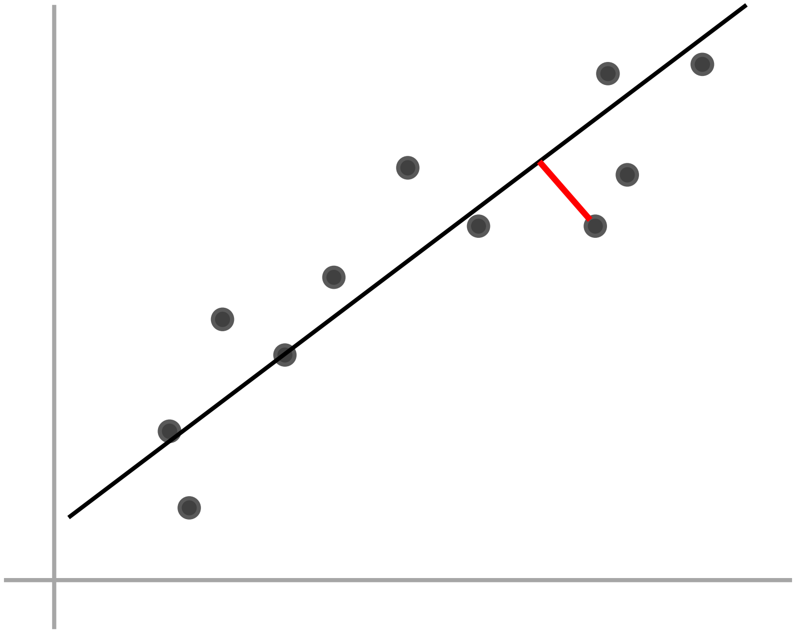


figure: austingwalters.com/pca-principal-component-analysis/

Why axis with largest variance?

1. Intuitively it seems right

2. It minimizes "reconstruction error"



PCA is an unsupervised learning method. It does not optimize fitness of the data for a supervised learning task.

Computing the principal components

One method is to use Singular Value Decomposition:

$$X = U\,\Sigma\,V^T$$

Recall that if X is an $m \times n$ matrix, then V^T is an $n \times n$ orthogonal matrix.

The columns $c_1, \ldots, c_n$ of V are the principal components:

$$\mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{c_1} & \mathbf{c_2} & \cdots & \mathbf{c_n} \\ | & | & & | \end{pmatrix}$$

this is a corrected version of Fig. 8-1 in the Géron text

Principal components in NumPy

```
X_centered = X - X.mean(axis=0)
U, s, V = np.linalg.svd(X_centered)
c1 = V.T[:,0]    # first principal component
c2 = V.T[:,1]    # second principal component
W2 = V.T[:, :2]  # matrix with first 2 components
```

V contains n principal components; one for each of the original features in X.

Projecting down to d dimensions

$$X_d = X\, W_d$$

Here W_d is the first d columns of V. Note the sizes:

$$m \times n \qquad n \times d \qquad \rightarrow \qquad m \times d$$

Example: suppose our data consists of 3 instances, and we want to reduce from 4 to 2 features.

$$\begin{array}{ccccc} X & & W_d & & X_\mathrm{d} \\ \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ & & & \\ & & & \end{pmatrix} & & \begin{pmatrix} a & \\ b & \\ c & \\ d & \end{pmatrix} & = & \begin{pmatrix} {\color{red}?} & \\ & \\ & \end{pmatrix} \end{array}$$

an original feature vector

first principal component

Projecting to d dimensions in sklearn

$$X_d = X\, W_d$$

Here W_d is the first d columns of V^T.

In Scikit-Learn:

```
W2 = V.T[:, :2]
X2D = X_centered.dot(W2)
```

PCA directly in Scikit-Learn

```
from sklearn.decomposition import PCA


pca = PCA(n_components=2)
X2D = pca.fit_transform(X)
```

The attribute .components_ of pca will give you the principal components. (Each row of this attribute is a component.)

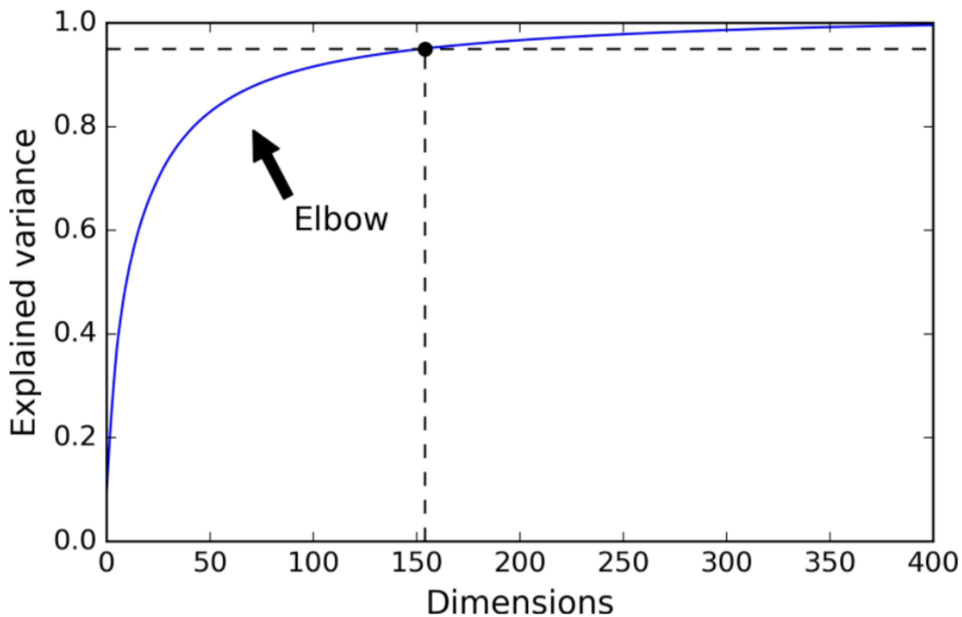
How many dimensions to project to?

The axis of the first component is the most informative, the axis of the second component the next most informative…

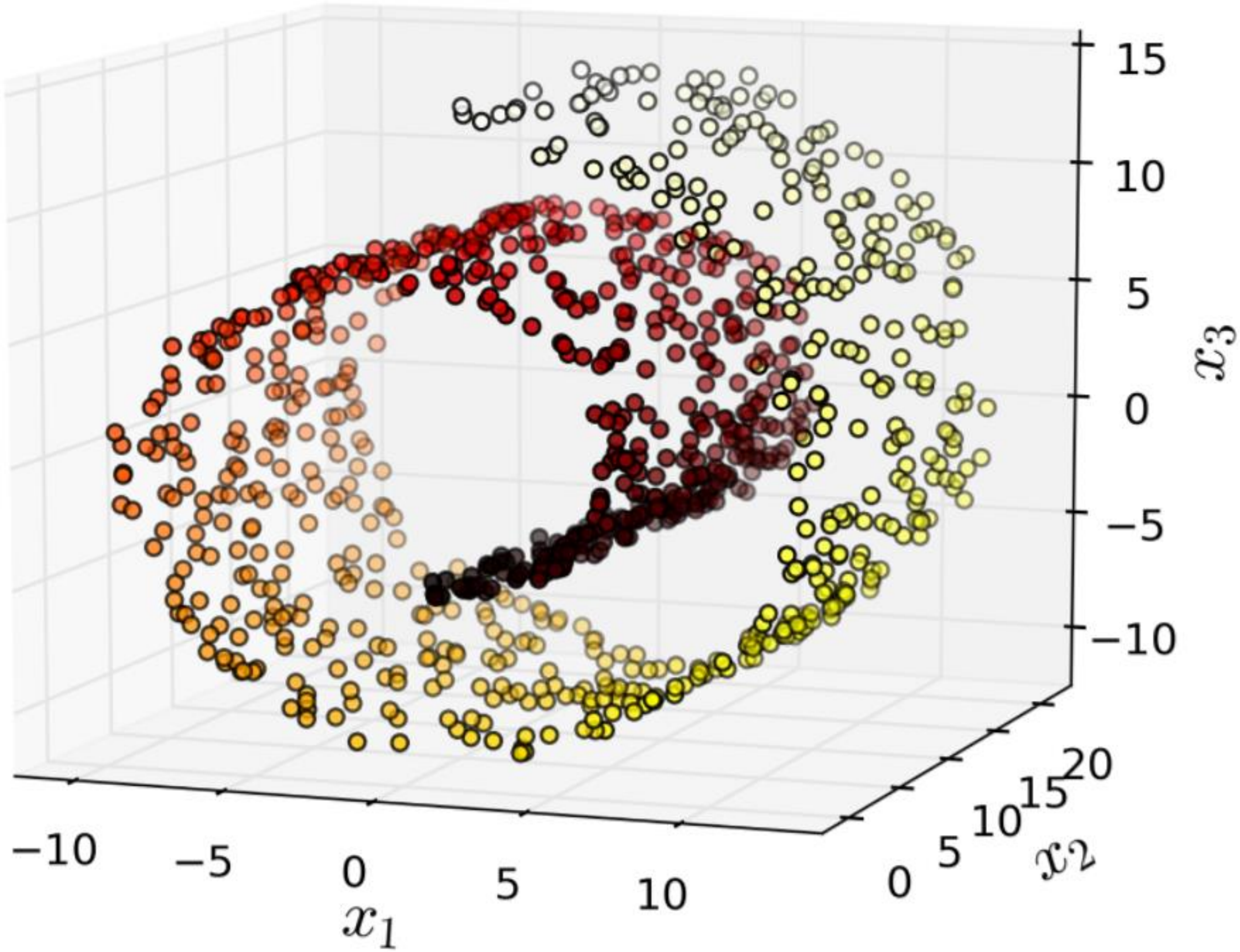
The explained variance ratio shows how much of your original dataset's variance lies along each component.

How many dimensions to project to?

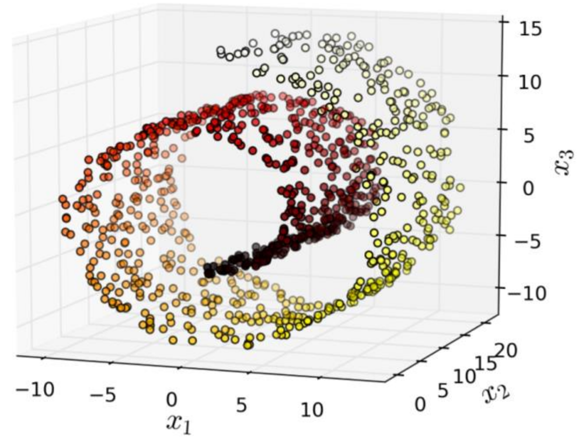
```
pca = PCA()
pca.fit(X)
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum > 0.95) + 1

# the easier way
pca = PCA(n_components = 0.95)
X_reduced = pca.fit_transform(X)
```

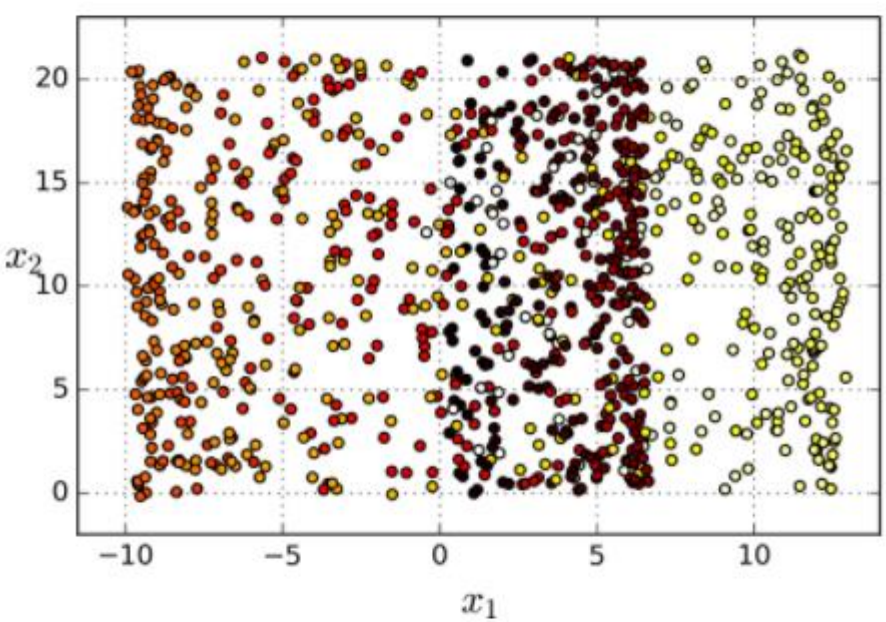


The swiss roll



Projection isn't always helpful





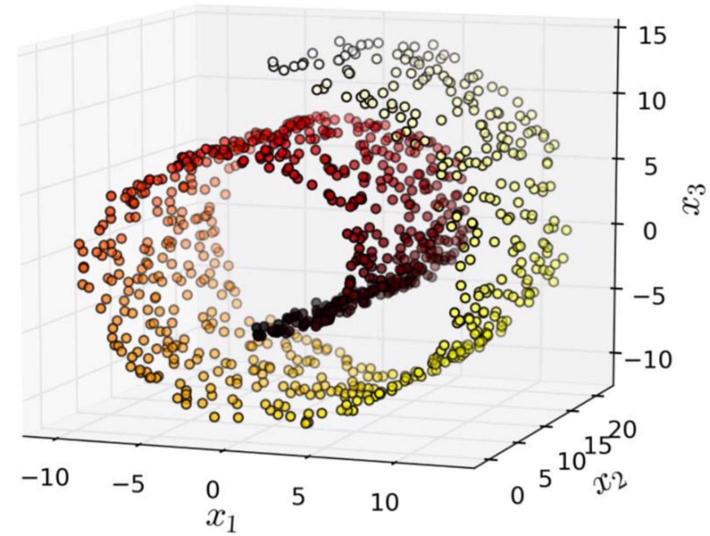
Dim. reduction by Manifold learning

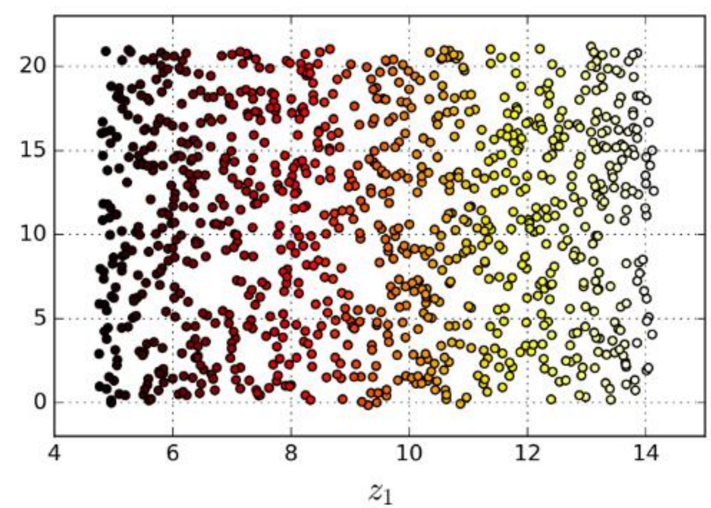
Look around you: the earth is flat.

It is a 2-D “manifold”, a 3-dimensional space that locally resembles a 2-D space.

A d-dimensional manifold is a part of an n-dimensional space (d < n) that locally resembles a d-dimensional hyperplane.

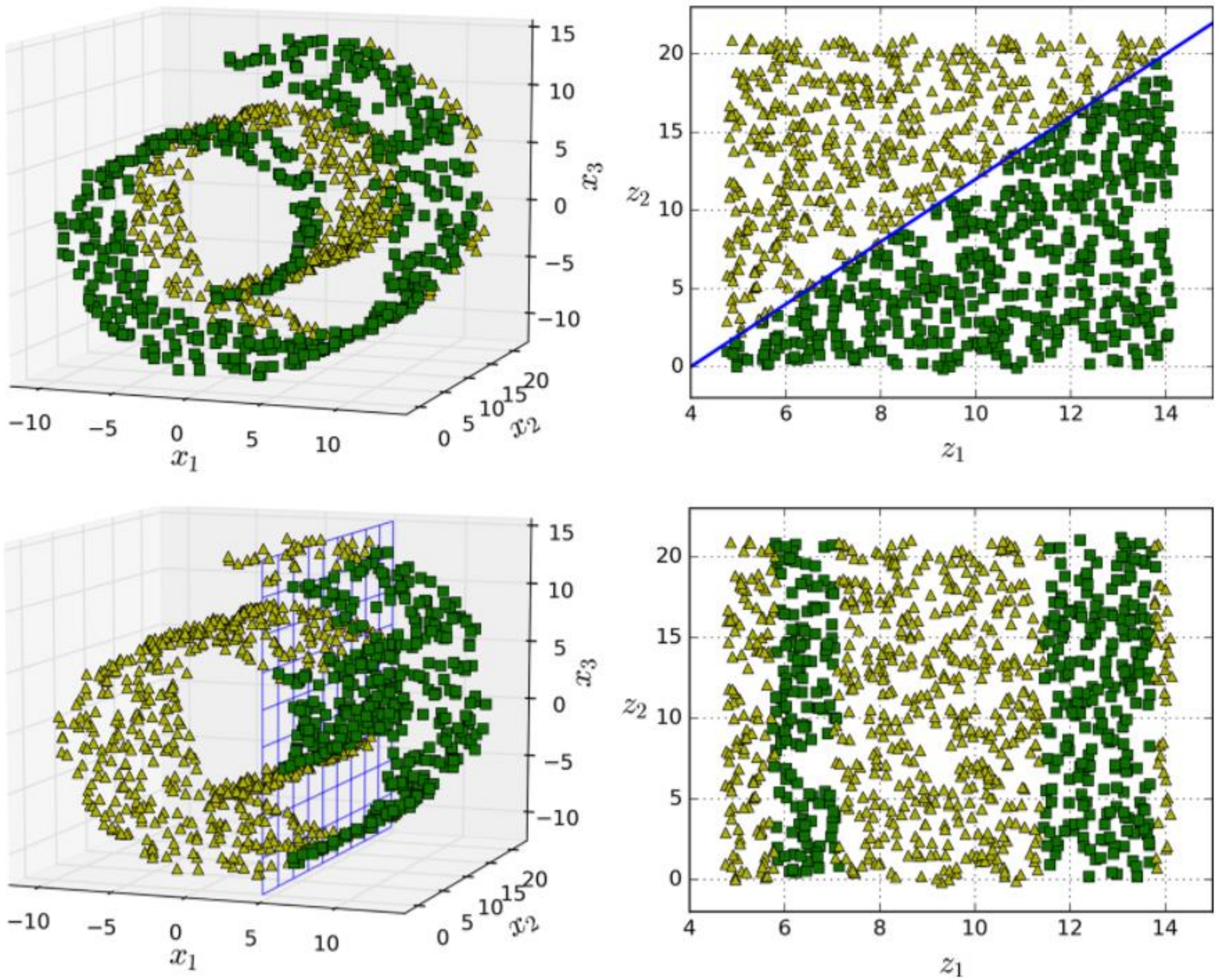
The swiss roll also locally resembles a 2-D plane.





some material on this page from Geron

Manifold learning



Summary

- Computational and visualization issues with high-dimensional data
- Not only that, high-dimensional spaces are bizarre and distances are strange
- Project and manifold learning are two main methods for dimensionality reduction
- Principal component analysis is a projection method
- The explained variance ratio can help in deciding on the reduced # of dimensions