CS-EJ3211 Machine Learning with Python

Session 6 - Feature Learning

Shamsi Abdurakhmanova

Aalto University
FITech

24.03.22

Feature Learning

- The efficiency of ML methods depends on the choice of features
- Features have to be relevant (useful properties of data points)
- Too many unnecessary features result in:
  - increase in computational resources used
  - overfitting

Feature Learning

Feature Learning - automate the choice of finding good features.

Feature Learning can be:

- supervised - dictionary learning, ANN
- unsupervised - PCA

Learn a hypothesis map that reads in some representation of a datapoint (e.g features) and transforms is to a set of (new) features:

$\mathbf{z} = (z_1, ..., z_D) \rightarrow \mathbf{x} = (x_1, ..., x_n)$,

Dimensionality reduction

Transform high dimensional (many features) dataset Z to a dataset X with lower dimensionality.

Dimensionality reduction

Why?

- reduce the use of computational resources
- prevent overfitting
- data visualization

How?

- How to map datapoint to a space with lower dimensionality?
- How to quantify the information loss after dataset transformation?

Dimensionality reduction - Compression

The goal is to find (learn) a compression map:

$h(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^n$,

that transforms a long feature vector $\mathbf{z} \in \mathbb{R}^D$ to a short feature vector $h(\mathbf{z}) = \mathbf{x} \in \mathbb{R}^n$ $(D \gg m)$.

The new feature vector $\mathbf{x} = h(\mathbf{z})$ is a compressed representation (code) of the original feature vector $\mathbf{z}$.

Dimensionality reduction - Reconstruction

We can reconstruct original vector using a reconstruction map:

$r(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^D$

Reconstructed original feature vector: $\hat{\mathbf{z}} = r(\mathbf{x})$

The difference (reconstruction error): $\hat{\mathbf{z}} - \mathbf{z}$

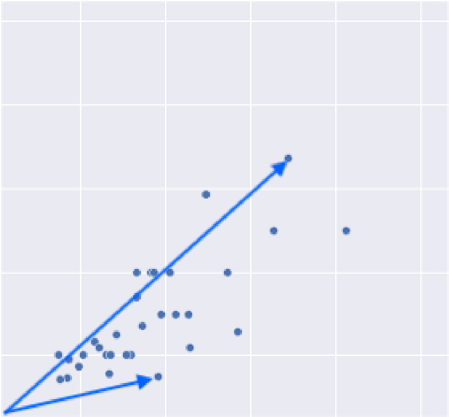
Need to learn a compression map such that: $\hat{\mathbf{z}} \approx \mathbf{z}$, i.e. reduce dimensionality with minimal information loss.

Principal Component Analyses

PCA is a dimensionality reduction method where compression map $h(\cdot)$ is a **linear** map.

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components.

Linear Transformation



3Blue1Brown

Principal Component Analyses

How to map datapoint to a space with lower dimensionality?

PCA, gif

Principal Component Analyses

Linear transformation: $h(\cdot) : \mathbb{R}^2 \to \mathbb{R}^1$:
$\mathbf{z} = (z_1, z_2) \to \mathbf{x} = (x_1)$
$\mathbf{x} = W\mathbf{z}$

Reconstruction: $\hat{\mathbf{z}} = W^T\mathbf{x}$

How to quantify the information loss after dataset transformation?

Information loss is measured with **reconstruction error**:

$$(1/m)\sum_{i=1}^{m} \left\| \mathbf{z}^{(i)} - \hat{\mathbf{z}}^{(i)} \right\|_2^2$$

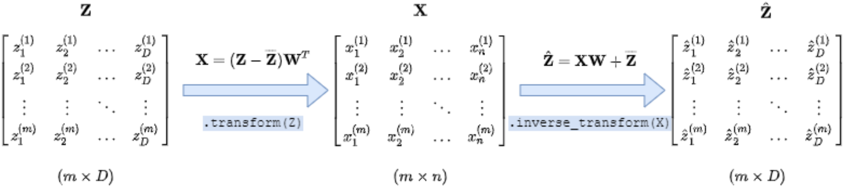
Reconstruction error, gif.
PC1, gif.
Source.

Principal Component Analyses

How to interpret PC components?

Cocktail recipe (PC1 $= x$ amount of feature 1 $+ y$ amount of feature 2), i.e. linear combination of features.

Check out PCA by StatQuest for more explanations.


$\mathbf{Z}$
$\mathbf{X} = (\mathbf{Z} - \bar{\mathbf{Z}})\mathbf{W}^T$
.transform(Z)
$\mathbf{X}$
$\hat{\mathbf{Z}} = \mathbf{X}\mathbf{W} + \bar{\mathbf{Z}}$
.inverse_transform(X)
$\hat{\mathbf{Z}}$
$(m \times D)$
$(m \times n)$
$(m \times D)$