Unsupervised learning

Balázs Pintér

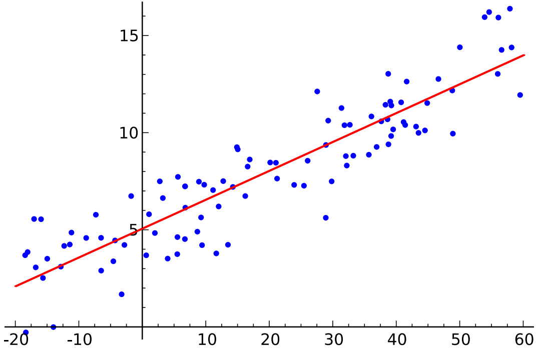
May 10, 2020

Contents

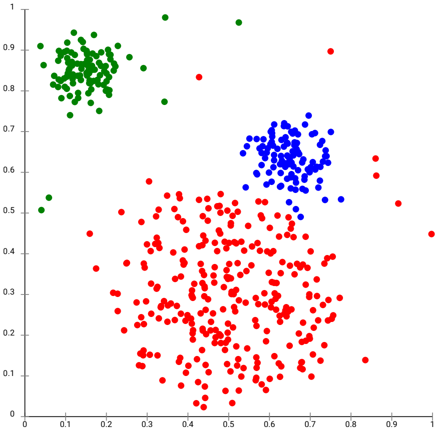
Contents

Supervised learning – classification

Supervised learning – regression



Unsupervised learning – clustering

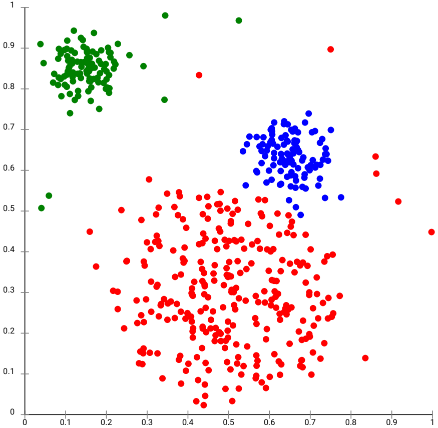


Unsupervised learning

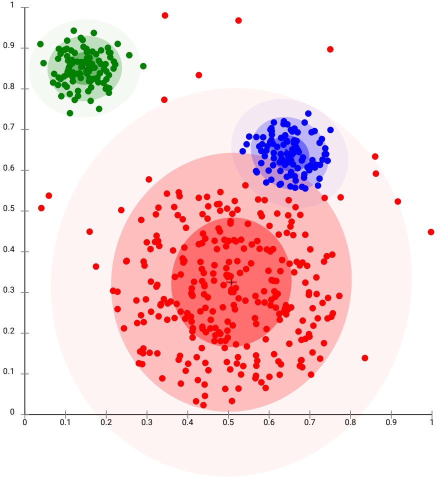
- Supervised learning learns a function from labeled data
- Other approaches
  1. **Unsupervised learning**
  2. Semi-supervised learning
  3. Reinforcement learning
  4. Evolutionary algorithms
  5. Neuroevolution
     `http://www.youtube.com/watch?v=qv6UVOQ0F44`

Contents

Example – distribution-based clustering



Example – distribution-based clustering



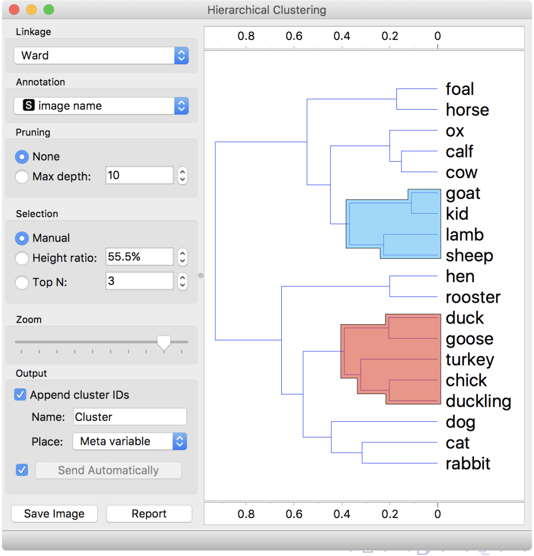
Problem

- Assign items to groups where similar items should belong to the same group
  - They should be as similar as possible within a group
  - They should be less similar between group
- The items are usually points in $\mathbb{R}^n$ or nodes in a graph, like
  - Client data for market segmentation
  - Documents modeled as bag of words to determine topics or group search results
  - Contexts of words to induce word senses
  - Data about which servers are active together to optimize traffic
- A group is called a *cluster*

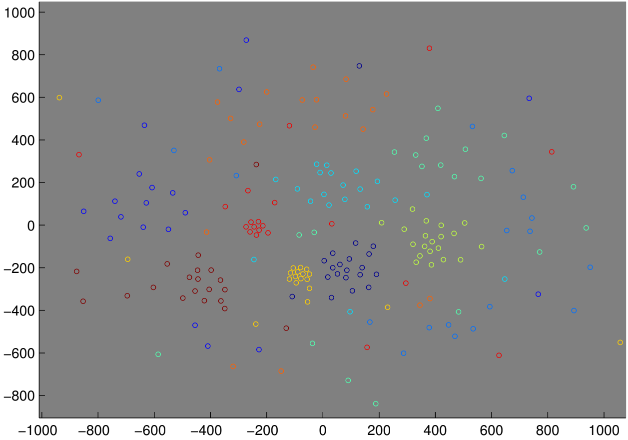
Hard and soft clustering

- Hard and soft clustering
  - Hard clustering: an item can belong only to a single cluster
  - Soft clustering: weights describe the degree to which an item belongs to the clusters
- Finer distinctions
  - Overlapping clustering: an item can belong to multiple clusters, but it either belongs to a cluster or not
  - Hierarchical clustering: the clusters are organized into a hierarchy. The items that belong to a child cluster also belong to the parent cluster.

Hierarchical clustering

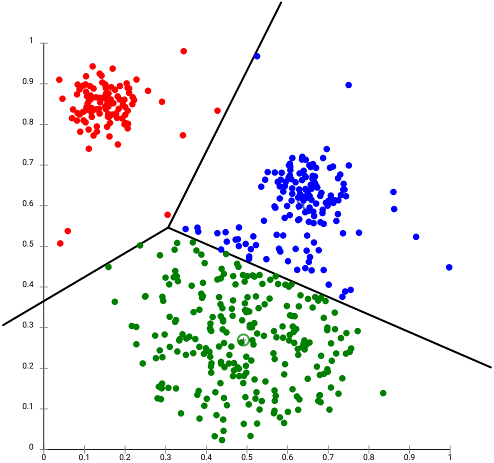


Naturally occurring clustering of word representations – words with the same meaning are the same color (t-SNE)



Contents

Example



Problem

- Given: k, the number of clusters
- Each cluster is represented by its centroid
  - the mean of the points in the cluster
- Find the k centroids and assign the points to these in a way that minimizes the squared distances of the points from the centroid of their cluster
  - Equivalent to minimizing pairwise distances within clusters
  - NP-hard, so we approximate
  - We can only find a local optimum
  - We can run it multiple times with different random initializations

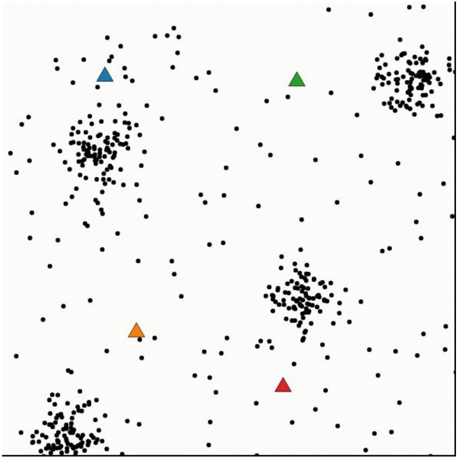
k-means problem

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \left\| \mathbf{x} - \mathbf{y} \right\|^2$$

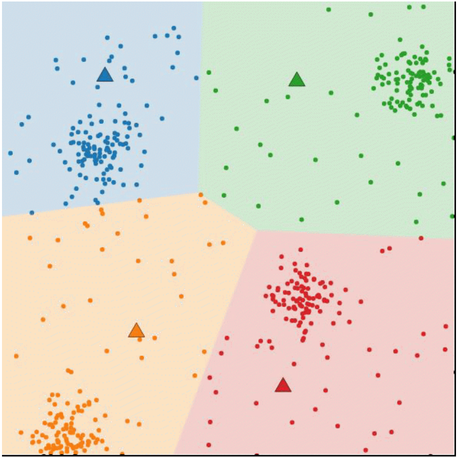
Algorithm

- Given: k and the items in $\mathbb{R}^n$
- Initialize centroids $m_1^{(1)}, m_2^{(1)}, \ldots, m_k^{(1)}$ by either
  - choosing k points randomly to be the centroids, or
  - assigning each point randomly to a cluster and computing the centroids of these clusters
- Do the following two steps until convergence
  1. Assign each item to the nearest centroid:
  $$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \le \left\| x_p - m_j^{(t)} \right\|^2 \; \forall j, 1 \le j \le k \right\}$$
  2. Compute the new centroids:
  $$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

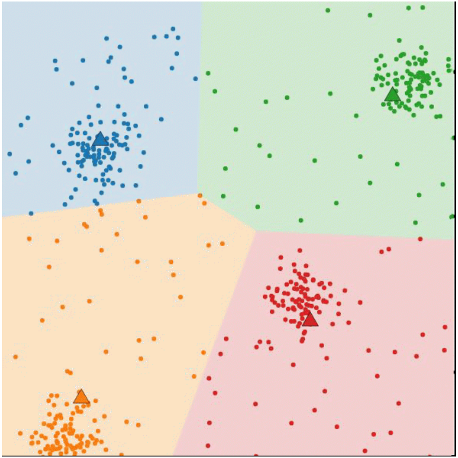
Algorithm



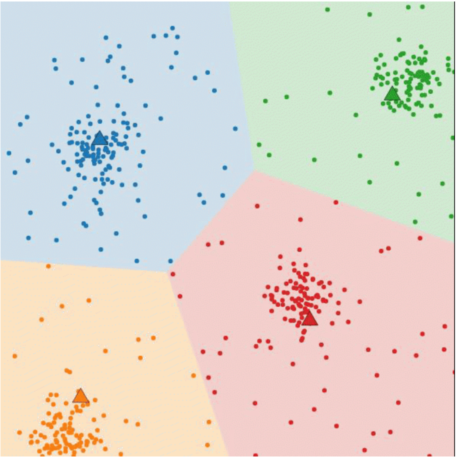
Algorithm



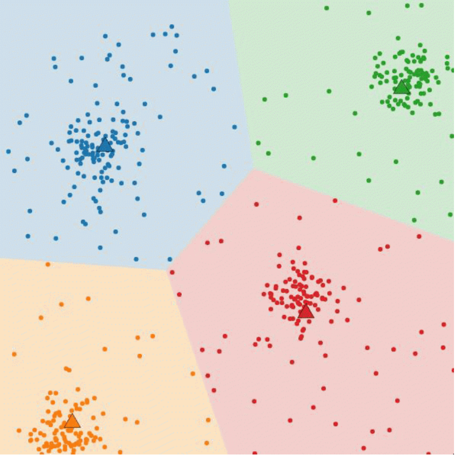
Algorithm



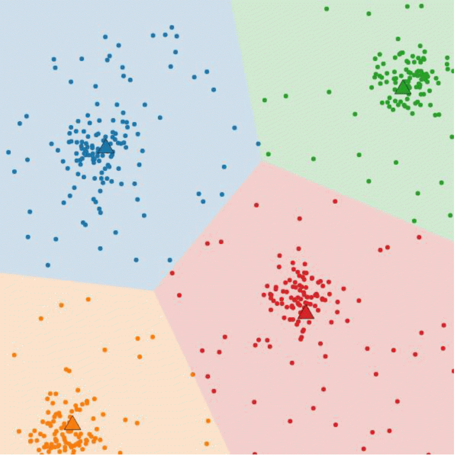
Algorithm



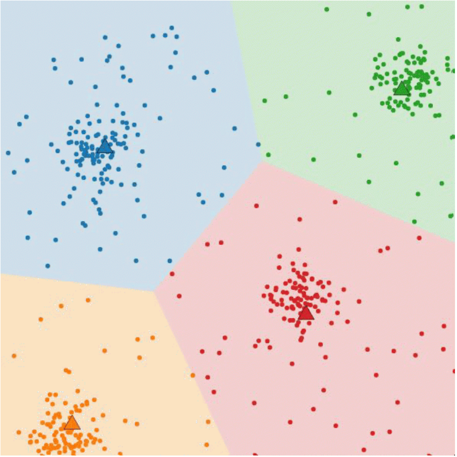
Algorithm



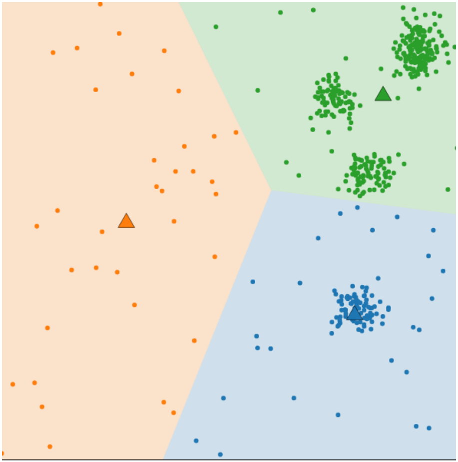
Algorithm



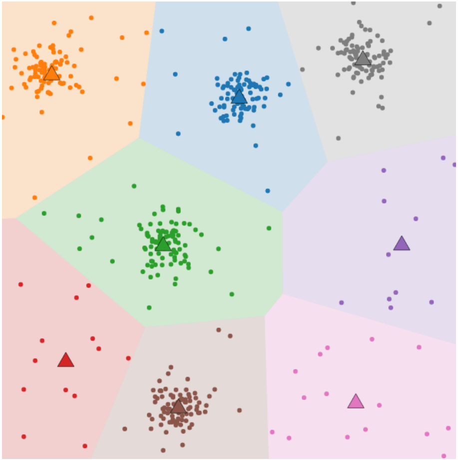
Algorithm



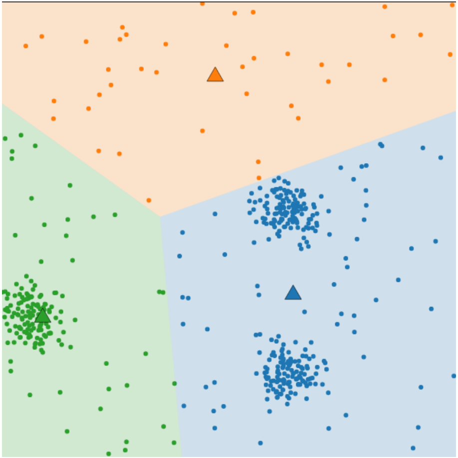
Issue – k is too small



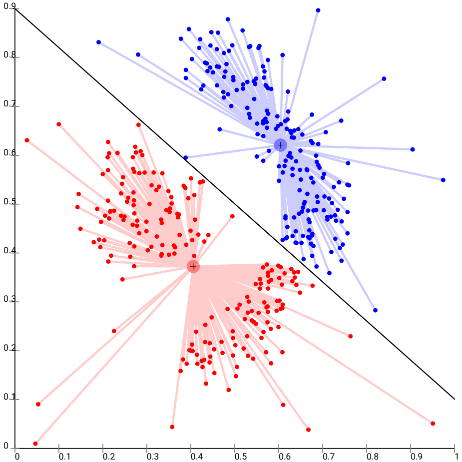
Issue – k is too large



Issue – bad initialization



Issue – the real clusters are not centroid based



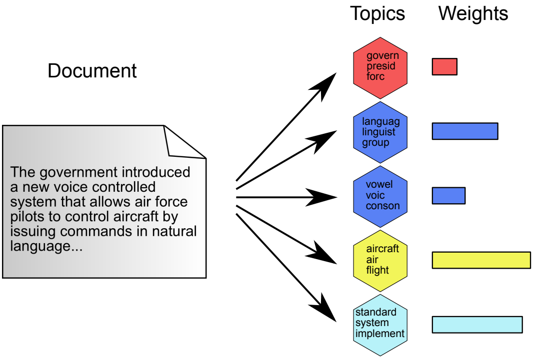
Python examples

- Color quantization: http://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html
- Document clustering: https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html

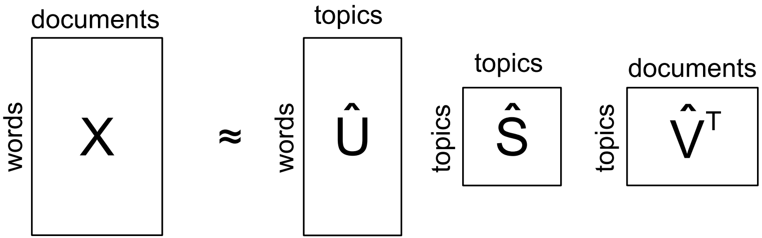
Contents

Topic models

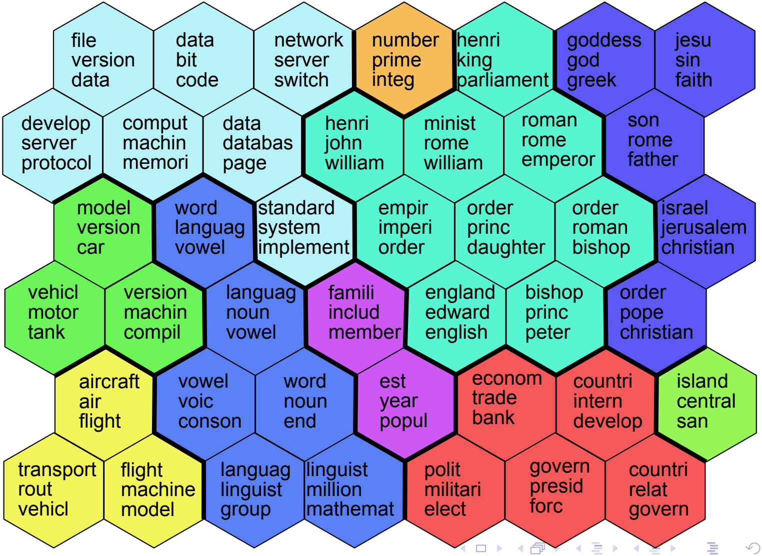
- Soft clustering example: topic models
  - What topics is a document about?
  - What words belong to a topic?
  - Example: Latent Semantic Analysis (LSA), which is a SVD



Latent Semantic Analysis



Group-sparse regularization



Example: generating odd one out puzzles

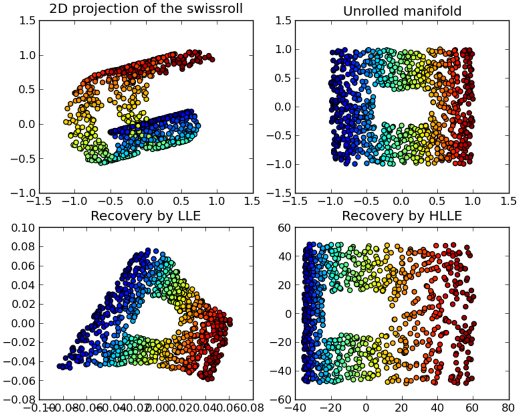
Words that belong together				Odd one out
cao	wei	liu	emperor	**king**
superman	clark	luthor	kryptonite	**batman**
devil	demon	hell	soul	**body**
egypt	egyptian	alexandria	pharaoh	**bishop**
singh	guru	sikh	saini	**delhi**
language	dialect	linguistic	spoken	**sound**
mass	force	motion	velocity	**orbit**
voice	speech	hearing	sound	**view**
athens	athenian	pericles	corinth	**ancient**
data	file	format	compression	**image**
function	problems	polynomial	equation	**physical**

Contents

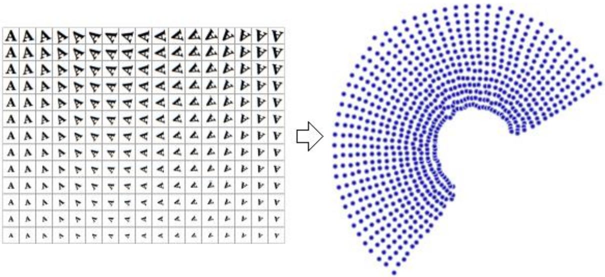
Why dimensionality reduction?

- The data are low dimensional in a higher dimensional space
- Data visualization
- Noise reduction
- Decrease the complexity of the learning problem (better results, smaller runtimes, ...)
- We can conjecture new relationships on the visualized lower dimensional data
- We need a lower dimensional and/or dense representation to solve a problem

Example: Swiss roll

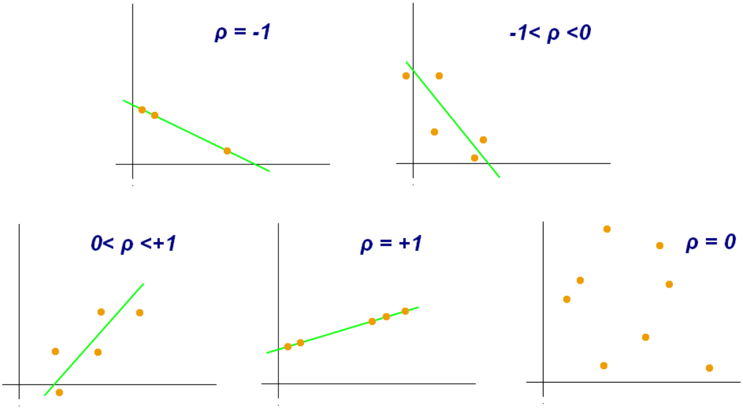


Example: Rotating a letter

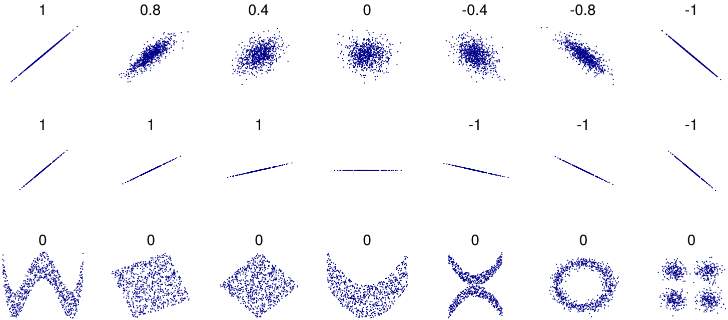


Contents

Covariance, correlation



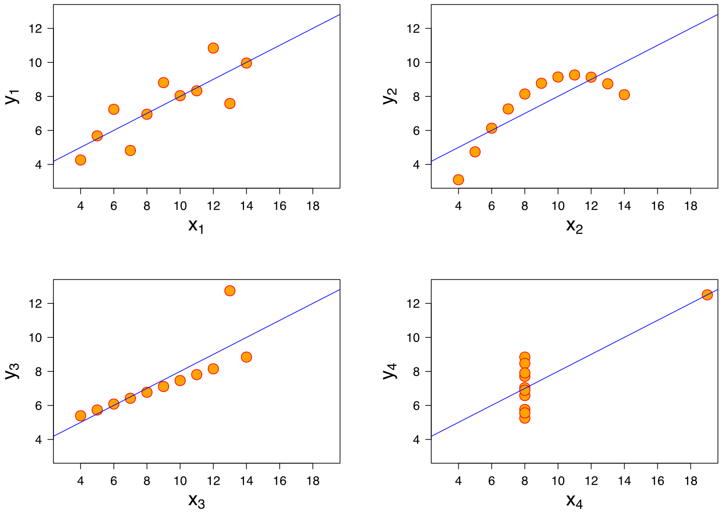
Covariance, correlation



Covariance, (Pearson) correlation

- They measure the degree to which the X, Y random variables move together
- They only show linear relationships
- $\mathrm{Cov}(X, Y) = \mathrm{E}\left[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])\right]$
- For example, if it's positive: If $X > E(X)$, then $Y > E(Y)$, if $X < E(X)$, then $Y < E(Y)$
- $\mathrm{Cov}(X, Y) = \mathrm{E}\left[XY\right] - \mathrm{E}[X]\,\mathrm{E}[Y]$
- Correlation: "normalized" covariance, between -1 and 1
- $\rho_{X,Y} = \mathrm{corr}(X, Y) = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$
- $r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$

The correlation is 0.816 for all four datasets

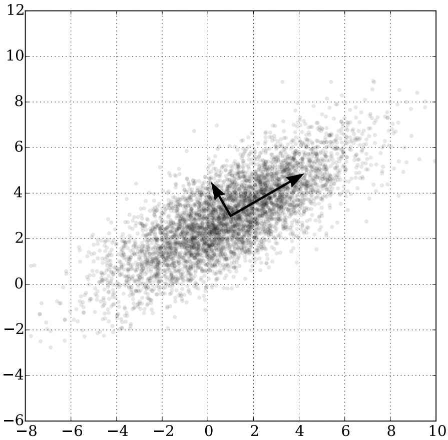


Covariance matrix

- $\mathbf{X}$ is a vector whose elements are random variables
- The entries of the covariance matrix are covariances between X_i, X_j
- $\Sigma_{ij} = \mathrm{cov}(X_i, X_j) = \mathrm{E}\left[(X_i - \mu_i)(X_j - \mu_j)\right]$
- $\mu_i = \mathrm{E}(X_i)$
- $$\begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\ \mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)] & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix}$$
- The main diagonal contains the variances
- Equivalent: $\Sigma = \mathrm{E}(\mathbf{X}^\top \mathbf{X}) - \mu^\top \mu$

Contents

Example – normal distribution in 2d



Principal component analysis

- Principal component analysis (PCA)
- Demo: `http://setosa.io/ev/principal-component-analysis/`
- The dataset is transformed to a new coordinate system whose axes are orthogonal
- The projection of the dataset with the greatest variance is on the first axis (principal component)
- The projection with the second greatest variance is on the second principal component, ...
- New variables/data: we project the original variables to the principal components. These are uncorrelated.
- Dimensionality reduction: We discard the axes (and coordinates) with small variance

Principal component analysis

- $\mathbf{X} \in \mathbb{R}^{n \times p}$: data set, one row is an item
- $\mathbf{t}_{(i)} = (t_1, \ldots, t_l)_{(i)}$: the items transformed to the new coordinate system using $\mathbf{w}_{(k)} = (w_1, \ldots, w_p)_{(k)}$

$$t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)} \qquad \text{for} \qquad i = 1, \ldots, n \qquad k = 1, \ldots, l$$

- Maximizing variance on the first principal component

$$\mathbf{w}_{(1)} = \underset{\|\mathbf{w}\|=1}{\arg\max} \left\{ \sum_i (t_1)^2_{(i)} \right\} = \underset{\|\mathbf{w}\|=1}{\arg\max} \left\{ \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w} \right)^2 \right\}$$

Principal component analysis

- The same with a matrix:

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \{\|\mathbf{X}\mathbf{w}\|^2\} = \arg\max_{\|\mathbf{w}\|=1} \left\{\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\right\}$$

- As $\mathbf{w}$ is a unit vector:

$$\mathbf{w}_{(1)} = \arg\max \left\{\frac{\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}}{\mathbf{w}^T\mathbf{w}}\right\}$$

- This is the Rayleigh-quotient, whose largest possible value is the largest eigenvalue of $\mathbf{X}^T\mathbf{X}$, where $\mathbf{w}$ is the corresponding eigenvector
- This is also true for the other components $\rightarrow$ the principal components are the eigenvectors of $\mathbf{X}^T\mathbf{X}$

Principal component analysis – algorithm

- Our dataset is in the $\mathbf{X}$ matrix
- Make the dataset zero mean (subtract the mean)
- Compute covariance matrix $\mathbf{Q} = \mathbf{X}^T\mathbf{X}$
- Determine the eigenvalues and eigenvectors of this matrix
- The eigenvectors are the principal components, the basis that consists of the eigenvectors is the new coordinate system
- The principal component that corresponds to the largest eigenvalue has the largest variance, and so on
- Dimensionality reduction: we only keep the k principal components with the largest eigenvalues

PCA and SVD

SVD

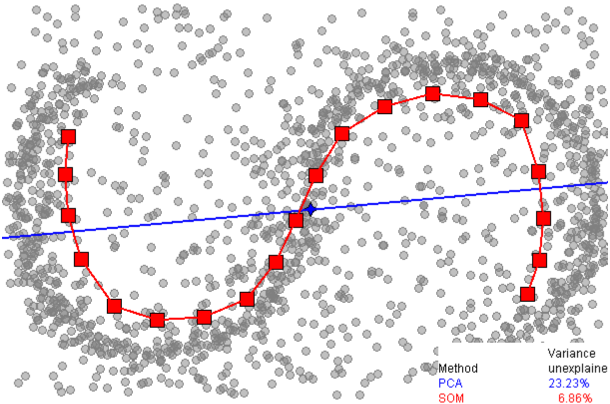
$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{W}^T$$

Computing PCA with SVD

$$\begin{aligned}\mathbf{X}^T\mathbf{X} &= \mathbf{W}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{W}^T \\ &= \mathbf{W}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{W}^T \\ &= \mathbf{W}\hat{\mathbf{\Sigma}}^2\mathbf{W}^T\end{aligned}$$

- $\mathbf{W}$ contains the eigenvectors of $\mathbf{X}^T\mathbf{X}$. The singular values are the square roots of the eigenvalues.

PCA is linear too

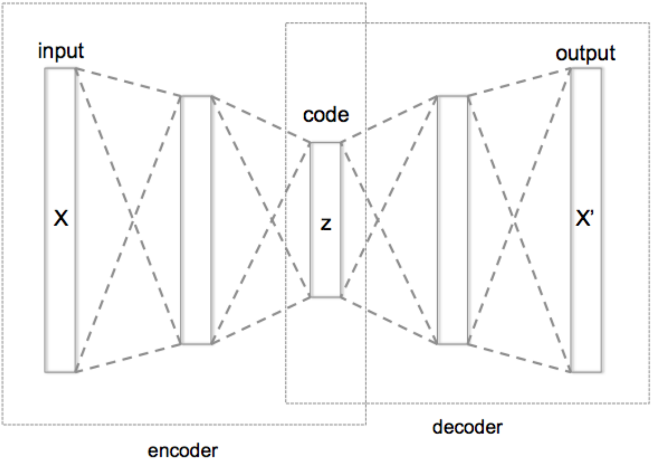


Python examples

- Importance of feature scaling: http://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html

Contents

Autoencoders



Autoencoders

Simple autoencoders

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2 = \|\mathbf{x} - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})) + \mathbf{b}')\|^2$$

- This simple autoencoder projects to the subspace of PCA
- Flexible, there are many variations
  - Denoising autoencoder: produces noiseless output from noisy input
  - Sparse autoencoder: the hidden representation is sparse
  - VAE: A probabilistic framework, approximates the posterior distribution
- Autoencoders can be important when pretraining a deep neural network
- `https://transcranial.github.io/keras-js/#/mnist-vae`

Thank you for your attention!

Thank you for your attention!