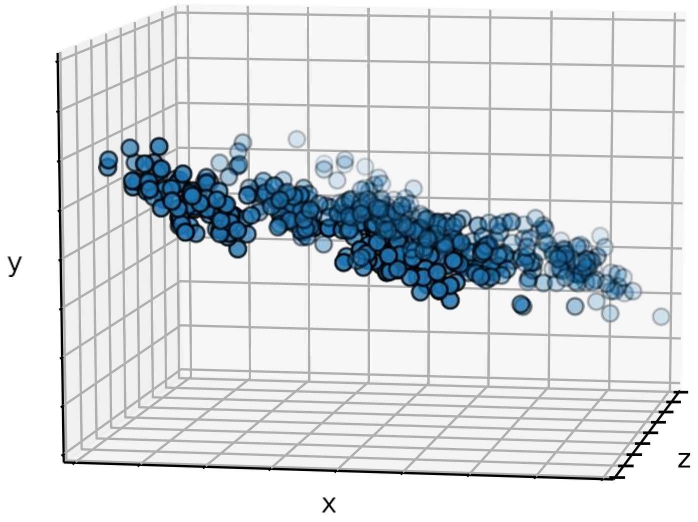
VDPA Course 4

- Prof. Dr. HDR. Darian M. Onchis
- West University of Timisoara
- October 2024
- *darian.onchis@e-uvt.ro*

Principal Component Analysis

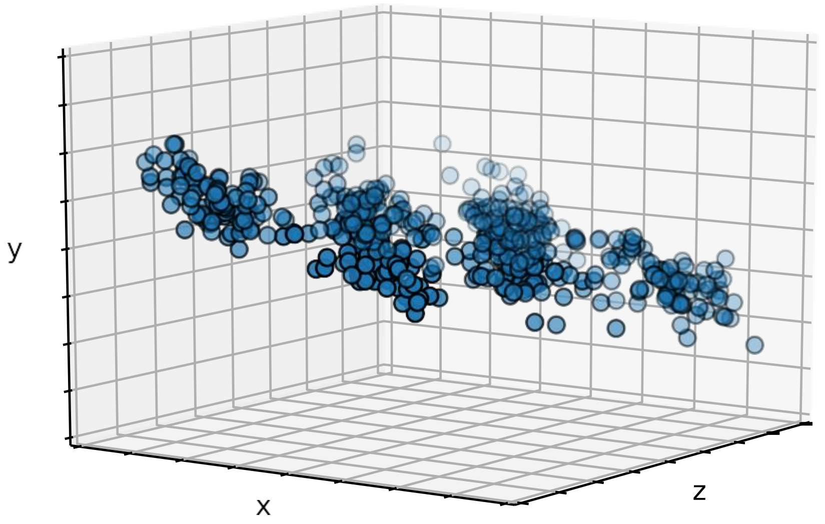
Intuition

- What can we do to get a **better view** of this data set?



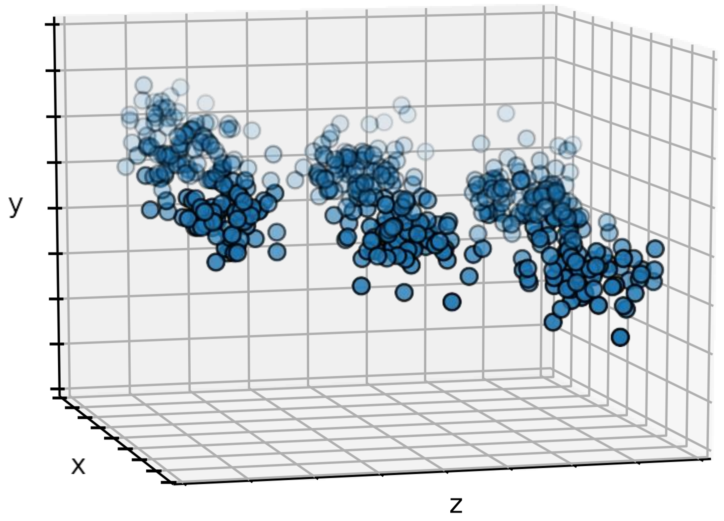
Intuition

- What can we do to get a **better view** of this data set?
- We can try to rotate the axes until we find the **best angle** to view the data



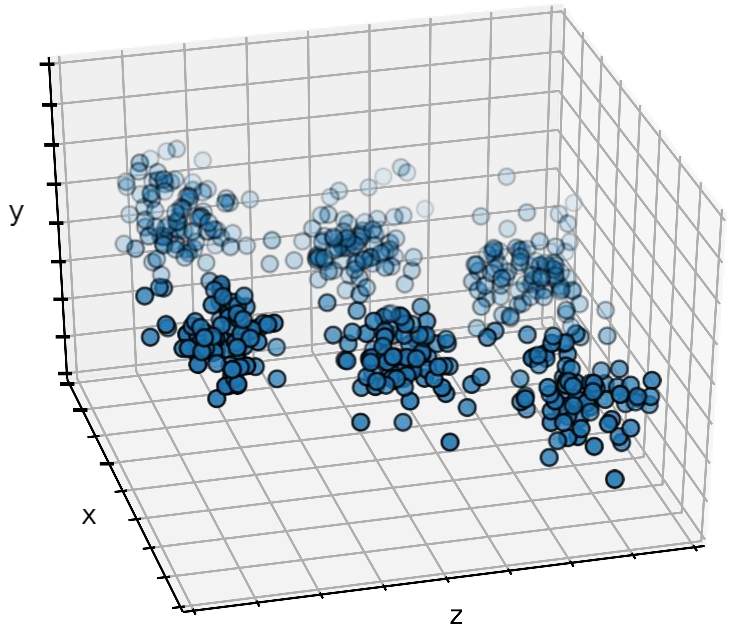
Intuition

- What can we do to get a **better view** of this data set?
- We can try to rotate the axes until we find the **best angle** to view the data



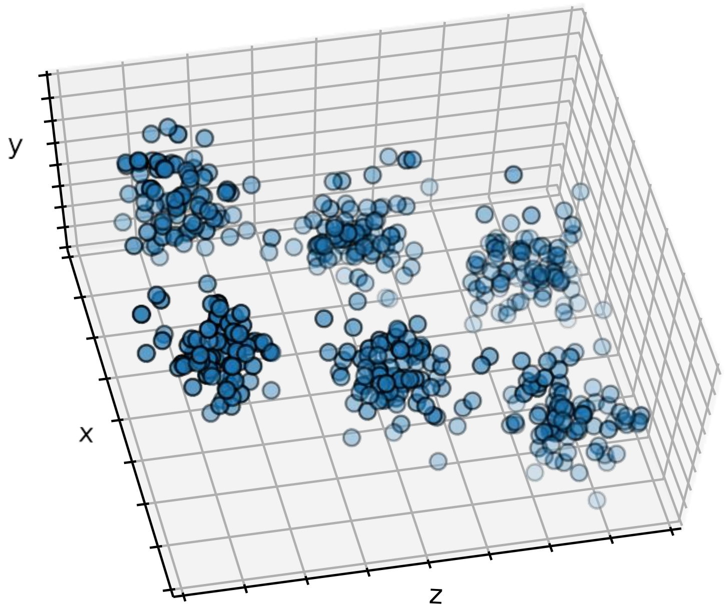
Intuition

- What can we do to get a **better view** of this data set?
- We can try to rotate the axes until we find the **best angle** to view the data



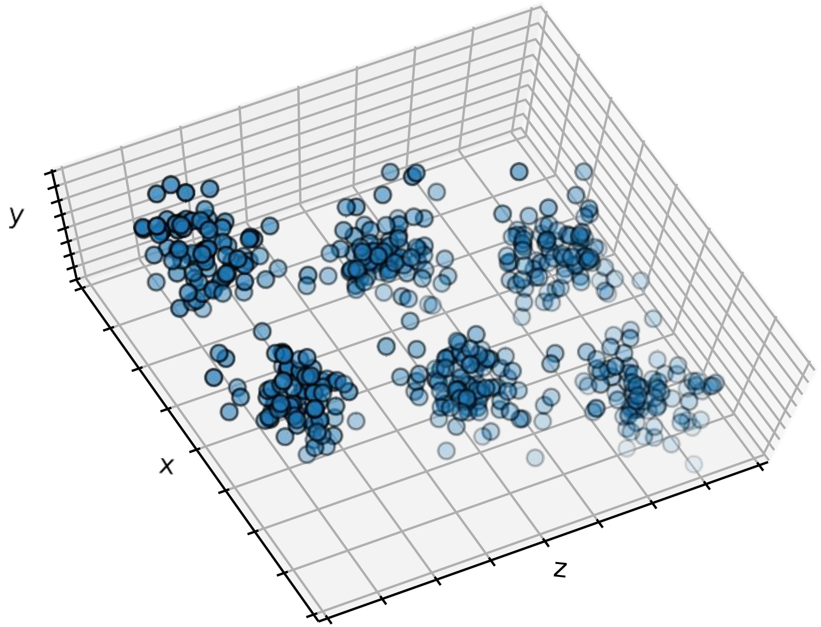
Intuition

- What can we do to get a **better view** of this data set?
- We can try to rotate the axes until we find the **best angle** to view the data



Intuition

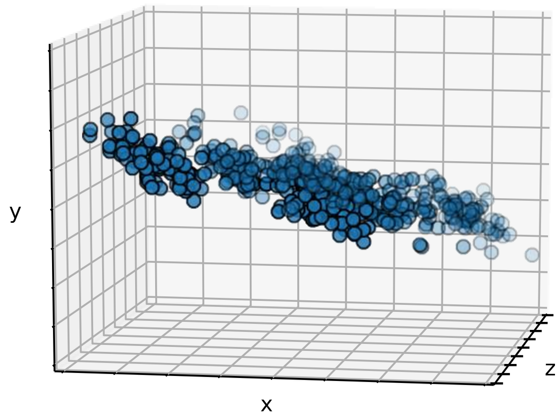
- What can we do to get a **better view** of this data set?
- We can try to rotate the axes until we find the **best angle** to view the data



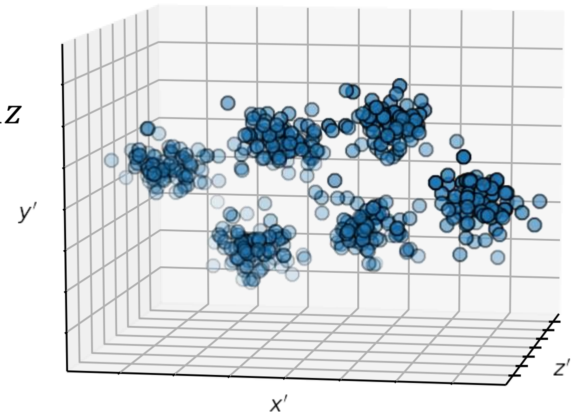
We get a better understanding of the data by looking from this angle than from our original perspective

Intuition

- Rotating the axes is equivalent to changing the coordinate system (or rotating the data)
- The axes that give the best view of the data are called **principal components**
  - These are the directions in which the data varies the most!



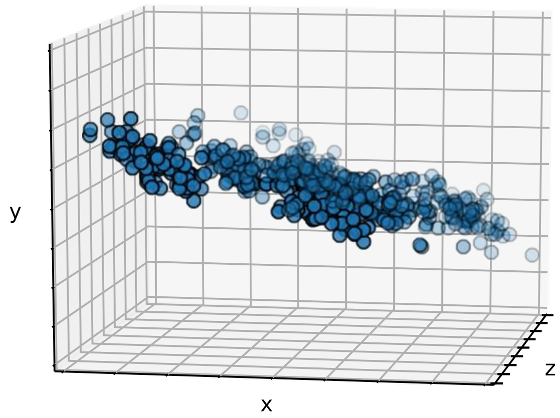
$$x' = 0.56x - 0.25y - 0.79z$$
$$y' = -0.75x + 0.25y - 0.61z$$
$$z' = 0.35x + 0.94y - 0.04z$$



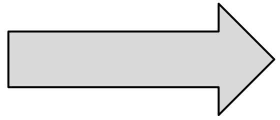
- We also compute how much variance is explained by each component:
  - x' explains ~75%, y' explains ~25%, z' explains <1%

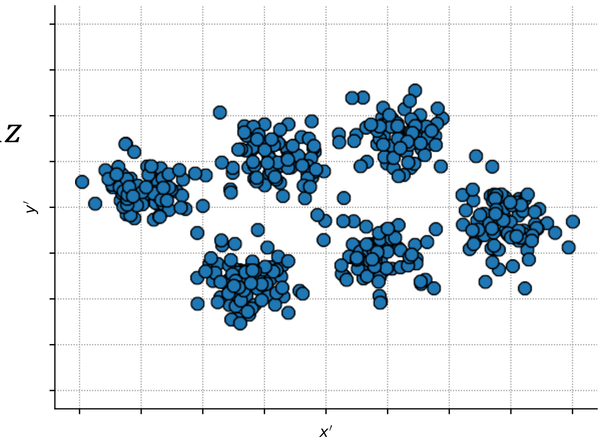
Intuition

- Rotating the axes is equivalent to changing the coordinate system (or rotating the data)
- The axes that give the best view of the data are called **principal components**
  - ➢ These are the directions in which the data varies the most!



$$x' = 0.56x - 0.25y - 0.79z$$
$$y' = -0.75x + 0.25y - 0.61z$$
$$z' = 0.35x + 0.94y - 0.04z$$





- We also compute how much variance is explained by each component:
  - ➢ x' explains ~75%, y' explains ~25%, ~~z' explains <1%~~
  - ➢ We can discard components that explain too little variance!

Mathematical Preliminaries

Random variable

- A random variable is a variable whose possible values are outcomes of a random phenomenon, e.g. rolling a die
  - ➢ The source of uncertainty in a random variable can be either:
    - a) objective – the result of a random process
    - b) subjective – the results of incomplete knowledge
  - ➢ A random variable has a probability distribution that specifies the probability of its value falling in any given interval
- A random variate is a particular outcome of a random variable
  - ➢ It is the result of sampling from the probability distribution
- The expected value E (or mean) of a random variable is the long-run average of its random variates
  - ➢ For the discrete case, it is the probability-weighted average of all possible outcomes

$E[$ $] = 3.5$

- The variance σ^2 is a measure of the dispersion of random variates around the expected value
  - ➢ σ is the standard deviation

Random variable

- Let $X = \{x_1, x_2, \dots, x_n\}$ and $Y = \{y_1, y_2, \dots, y_n\}$ be two sets of random variates sampled together from two random variables, i.e. x_i is sampled at the same time as y_i
- X and Y are just subsets of the whole (potentially infinite) population
  - ➢ Then, the sample mean and the sample variance are just estimates of the true mean and variance values:

$$Mean(X) = E[X] = \mu_X = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$Var(X) = \sigma_X^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu_X)^2$$

Random variable

- Let $X = \{x_1, x_2, \dots, x_n\}$ and $Y = \{y_1, y_2, \dots, y_n\}$ be two sets of random variates sampled together from two random variables, i.e. x_i is sampled at the same time as y_i
- X and Y are just subsets of the whole (potentially infinite) population
  - Then, the sample mean and the sample variance are just estimates of the true mean and variance values:

$$Mean(X) = E[X] = \mu_X = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$Var(X) = \sigma_X^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_X)^2$$

$n-1$ instead of n is Bessel's correction that accounts for the biased estimation of the mean

- Covariance measures the joint variability of two random variables, e.g. if larger values of X correspond to larger values of Y and smaller values of X to smaller values of Y:

$$Cov(X,Y) = \sigma_{X,Y}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_X)\cdot(y_i - \mu_Y)$$

Features as random variables

- Consider a data set $X \in \mathbb{R}^{m \times n}$ with m examples and n features:

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix}$$

  - ➢ Each row represents an example (vector of n features)
  - ➢ Each column represents the same feature of all examples
- We can regard each feature (column) as a **random variable**
  - ➢ This means that $x_{i,j}$ (feature j of example i) is a **random variate**
  - ➢ Examples are made up of n random variables that are sampled together
- This makes X a set of m sampling events of n random variables
  - ➢ We can compute the sample mean, variance, covariance

Centering the data set

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix}$$

$$\mu_X = (\mu_1 \quad \mu_2 \quad \cdots \quad \mu_n) = \frac{1}{m}\left(\sum_{i=1}^{m} x_{i,1} \quad \sum_{i=1}^{m} x_{i,2} \quad \cdots \quad \sum_{i=1}^{m} x_{i,n}\right)$$

$$\bar{X} = \begin{pmatrix} x_{1,1}-\mu_1 & x_{1,2}-\mu_2 & \cdots & x_{1,n}-\mu_n \\ x_{2,1}-\mu_1 & x_{2,2}-\mu_2 & \cdots & x_{2,n}-\mu_n \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1}-\mu_1 & x_{m,2}-\mu_2 & \cdots & x_{m,n}-\mu_n \end{pmatrix} = X - I_{m\times 1}\cdot\mu_X$$

$\boxed{\text{Centered version of } X}$

Computing the covariance of the data set

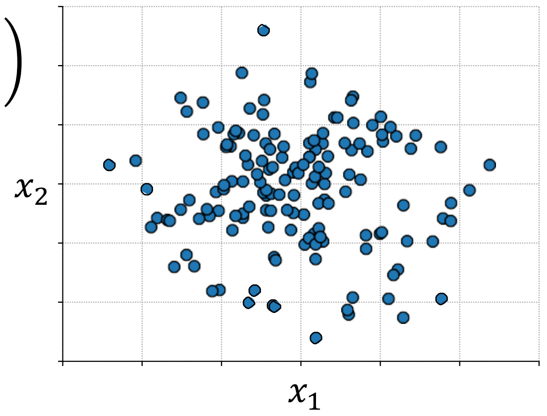
$$\bar{X} = \begin{pmatrix} x_{1,1}-\mu_1 & x_{1,2}-\mu_2 & \cdots & x_{1,n}-\mu_n \\ x_{2,1}-\mu_1 & x_{2,2}-\mu_2 & \cdots & x_{2,n}-\mu_n \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1}-\mu_1 & x_{m,2}-\mu_2 & \cdots & x_{m,n}-\mu_n \end{pmatrix} \quad \bar{X}^T = \begin{pmatrix} x_{1,1}-\mu_1 & x_{2,1}-\mu_1 & \cdots & x_{m,1}-\mu_1 \\ x_{1,2}-\mu_2 & x_{2,2}-\mu_2 & \cdots & x_{m,2}-\mu_2 \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,n}-\mu_n & x_{2,n}-\mu_n & \cdots & x_{m,n}-\mu_n \end{pmatrix}$$

$$\frac{1}{m-1}\bar{X}^T\cdot\bar{X} = \frac{1}{m-1}\begin{pmatrix} \sum_{i=1}^{m}(x_{i,1}-\mu_1)^2 & \sum_{i=1}^{m}(x_{i,1}-\mu_1)\cdot(x_{i,2}-\mu_2) & \cdots & \sum_{i=1}^{m}(x_{i,1}-\mu_1)\cdot(x_{i,n}-\mu_n) \\ \sum_{i=1}^{m}(x_{i,2}-\mu_2)\cdot(x_{i,1}-\mu_1) & \sum_{i=1}^{m}(x_{i,2}-\mu_2)^2 & \cdots & \sum_{i=1}^{m}(x_{i,2}-\mu_2)\cdot(x_{i,n}-\mu_n) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{m}(x_{i,n}-\mu_n)\cdot(x_{i,1}-\mu_1) & \sum_{i=1}^{m}(x_{i,n}-\mu_n)\cdot(x_{i,2}-\mu_2) & \cdots & \sum_{i=1}^{m}(x_{i,n}-\mu_n)^2 \end{pmatrix}$$

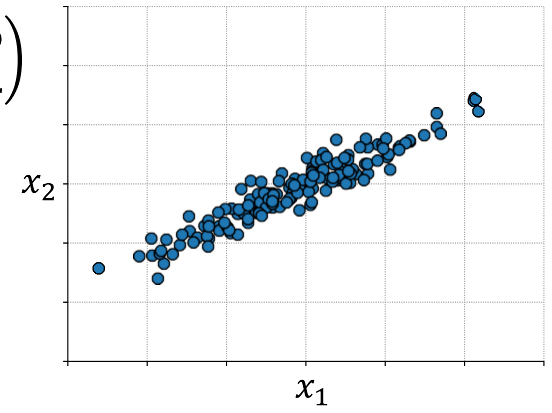
$$= \begin{pmatrix} Var(X_1) & Cov(X_1,X_2) & \cdots & Cov(X_1,X_n) \\ Cov(X_2,X_1) & Var(X_2) & \cdots & Cov(X_2,X_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_n,X_1) & Cov(X_n,X_2) & \cdots & Var(X_n) \end{pmatrix} \overset{\text{def}}{=} Cov(X) = \Sigma_X$$

Covariance of the data set

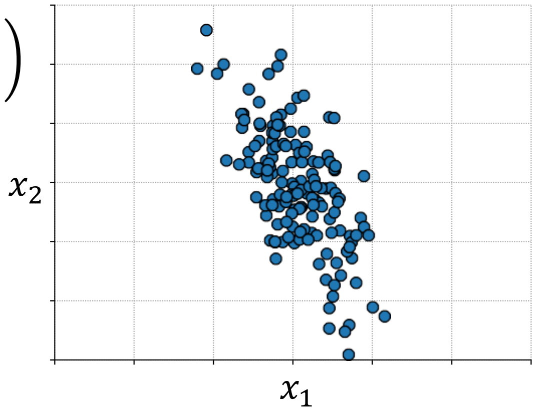
$$\Sigma_X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



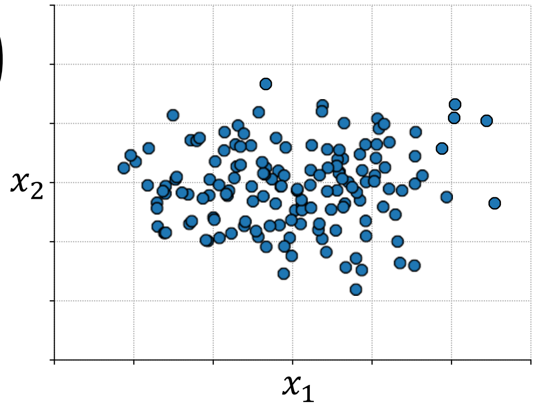
$$\Sigma_X = \begin{pmatrix} 1 & 0.6 \\ 0.6 & 0.4 \end{pmatrix}$$



$$\Sigma_X = \begin{pmatrix} 0.2 & -0.3 \\ -0.3 & 1 \end{pmatrix}$$



$$\Sigma_X = \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix}$$



Eigenvectors and eigenvalues

- An eigenvector $v \in \mathbb{R}^n$ and its corresponding eigenvalue $\lambda \in \mathbb{R}$ of a square matrix $\mathrm{A} \in \mathbb{R}^{n \times n}$ are the solutions to the following equation:

$$A \cdot v = \lambda \cdot v$$

- In other words, when multiplied by A, an eigenvector only gets scaled, i.e. the direction of v does not change
  - ➢ The corresponding eigenvalue λ is the scaling factor
- There are multiple solutions to the equation above, i.e. there are more eigenvectors and corresponding eigenvalues

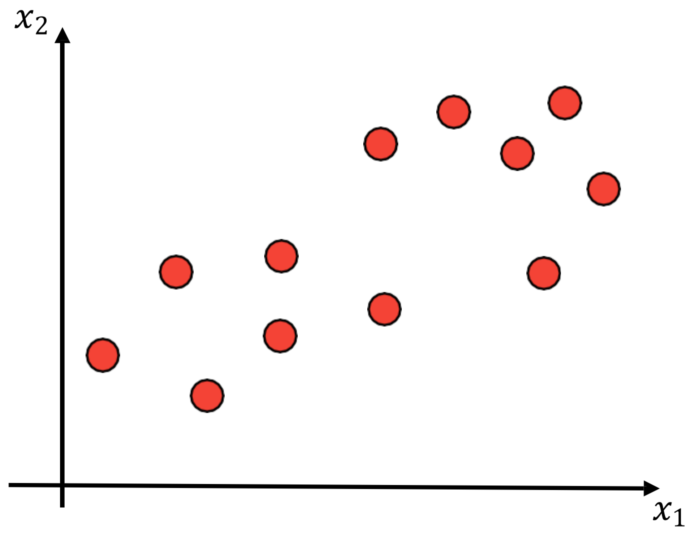
Finding Principal Components

Principal Component Analysis

- Principal Component Analysis (PCA) transforms a data set into a new **orthogonal coordinate system** in which the data is centered and the features are completely uncorrelated
  - ➢ The mean of the new data set is 0
  - ➢ The covariance of any pair of distinct features is 0
- The features of the transformed data set are called principal components
- Principal components are sorted in descending order by variance, i.e. the first component has the largest variance
  - ➢ Components with low variance can be discarded, making PCA a method of dimensionality reduction

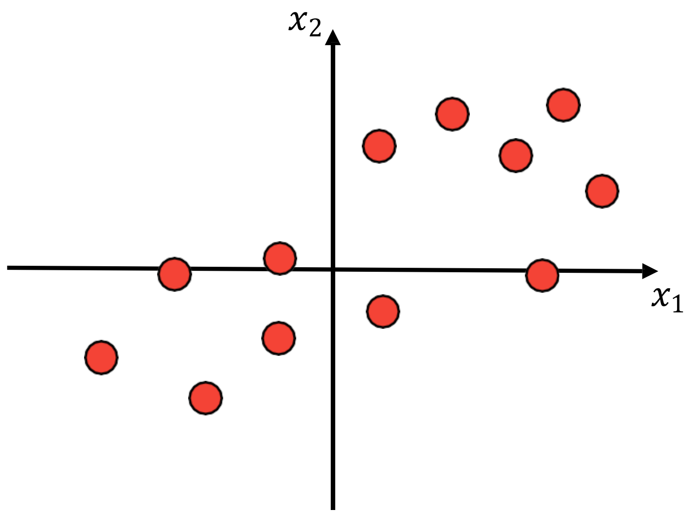
Finding the first principal component

- The first step is centering the data (subtracting the mean from all data points)



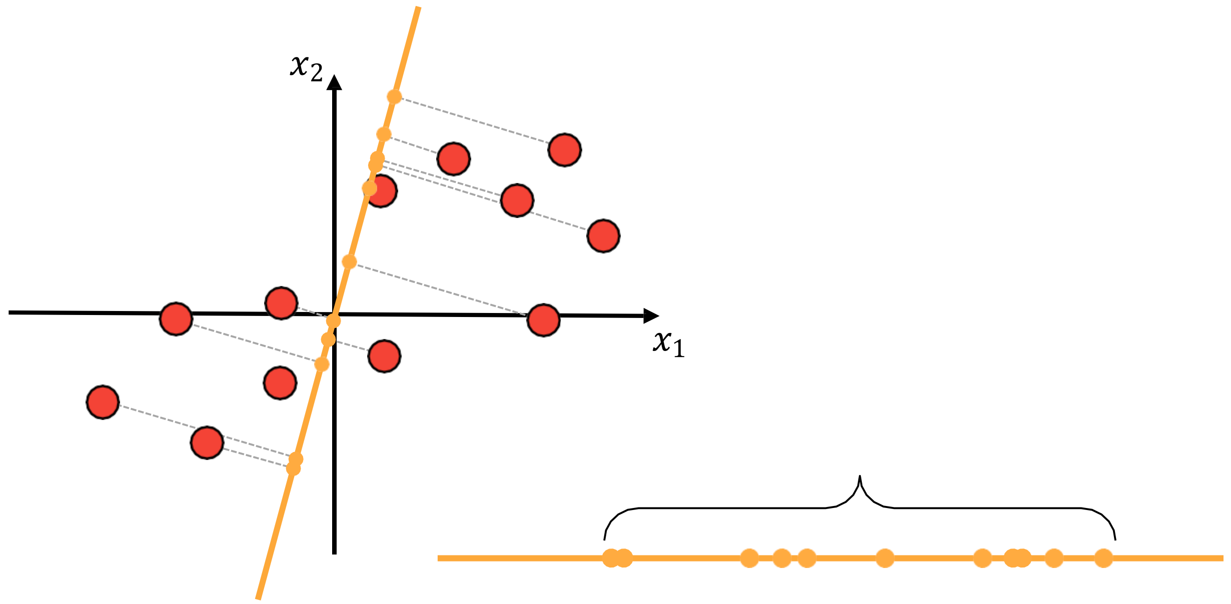
Finding the first principal component

- The first step is centering the data (subtracting the mean from all data points)



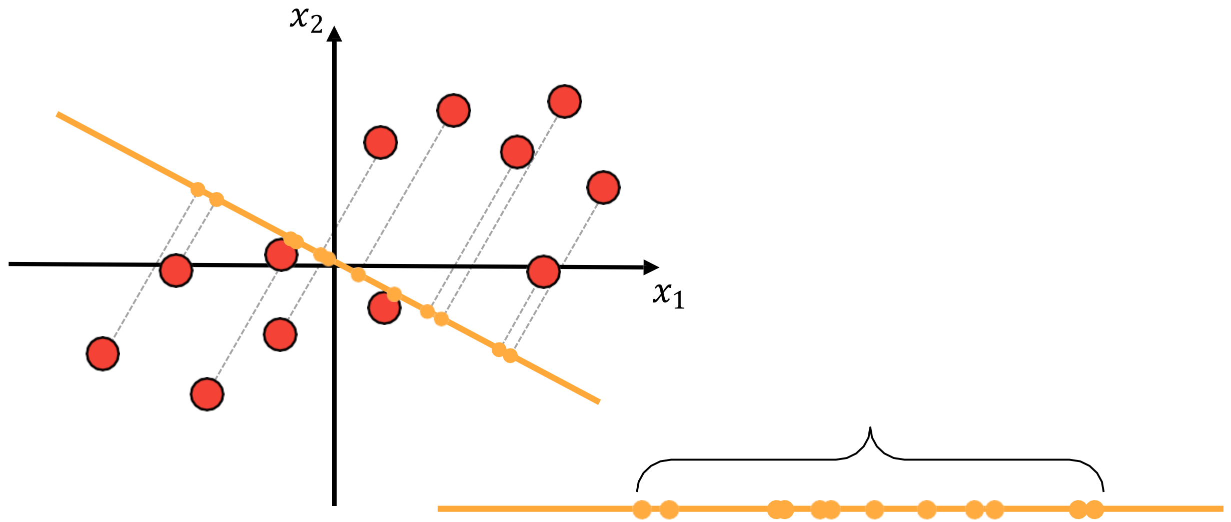
Finding the first principal component

- The first step is centering the data (subtracting the mean from all data points)
- The first principal component is the direction on which the data has the largest variance
  - ➢ We are looking for a line on which the projection of data points is as spread out as possible



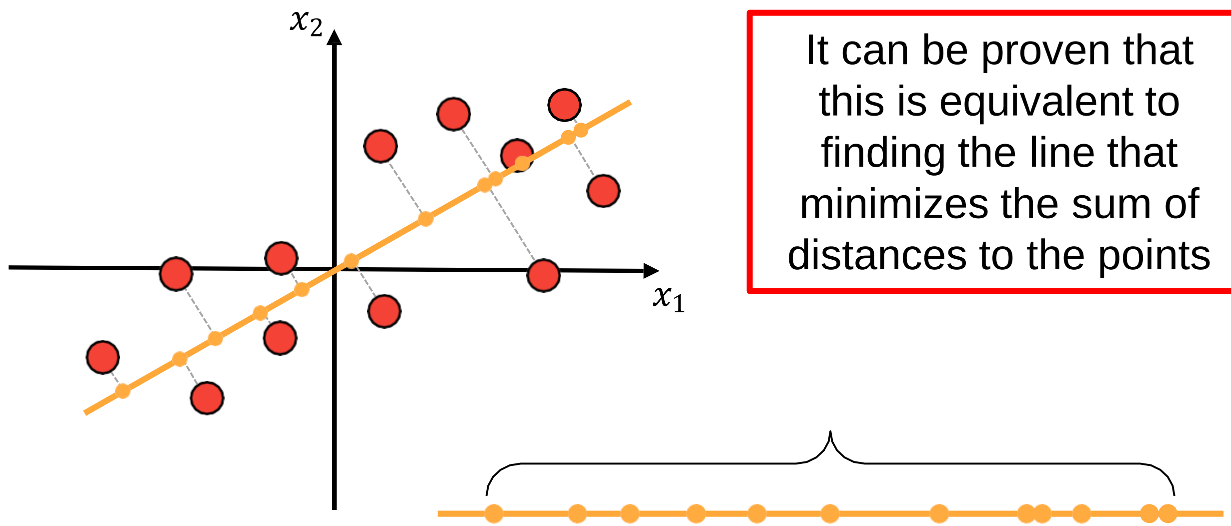
Finding the first principal component

- The first step is centering the data (subtracting the mean from all data points)
- The first principal component is the direction on which the data has the largest variance
  - ➢ We are looking for a line on which the projection of data points is as spread out as possible



Finding the first principal component

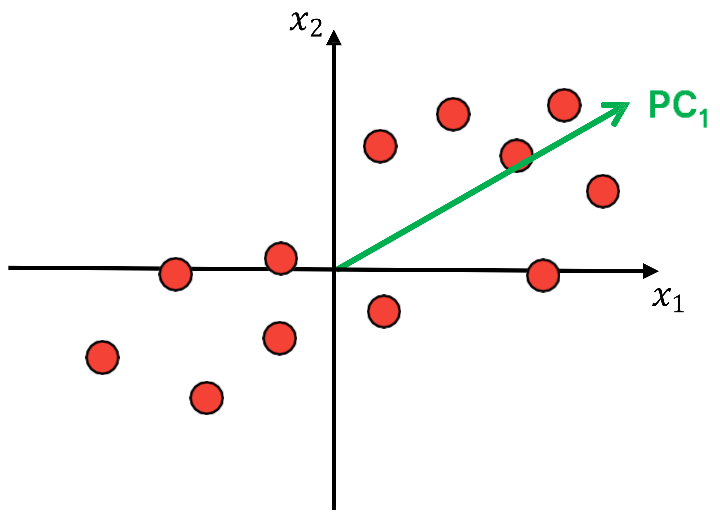
- The first step is centering the data (subtracting the mean from all data points)
- The first principal component is the direction on which the data has the largest variance
  - ➢ We are looking for a line on which the projection of data points is as spread out as possible



It can be proven that this is equivalent to finding the line that minimizes the sum of distances to the points

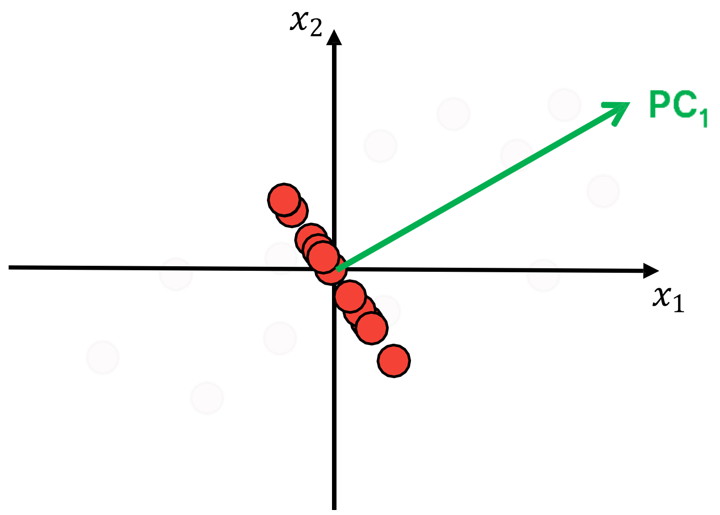
Finding the second principal component

- If we subtract the projection of points on the first PC from all the points, we obtain a data set that has 0 variance on that direction



Finding the second principal component

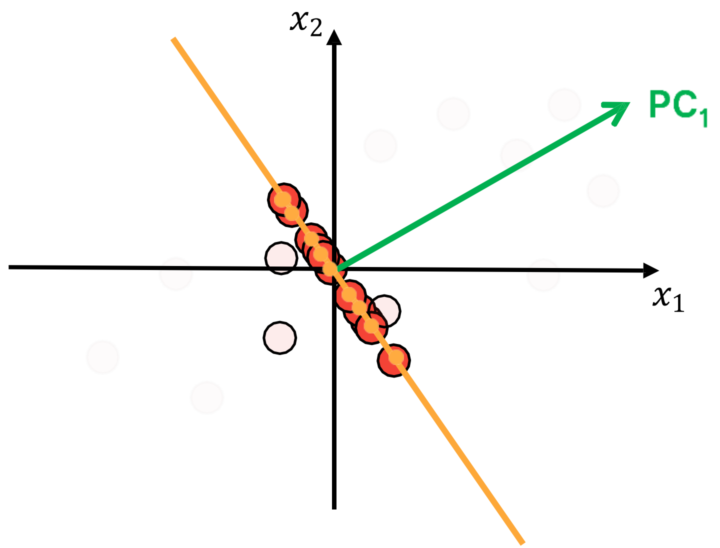
- If we subtract the projection of points on the first PC from all the points, we obtain a data set that has 0 variance on that direction



Finding the second principal component

- If we subtract the projection of points on the first PC from all the points, we obtain a data set that has 0 variance on that direction
- We now look for a line on which the projection of data points from the new data set is as spread out as possible
  - ➢ The result will be the second principal component

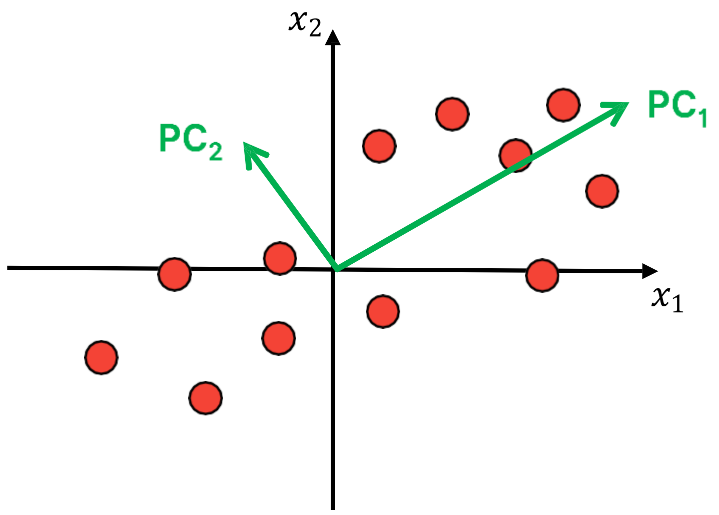
Trivial in two dimensions



The second PC is the direction in which the data varies the most, after eliminating the variance on the first PC

Finding further principal components

- If we subtract the projection of points on the first PC from all the points, we obtain a data set that has 0 variance on that direction
- We now look for a line on which the projection of data points from the new data set is as spread out as possible
  - ➢ The result will be the second principal component

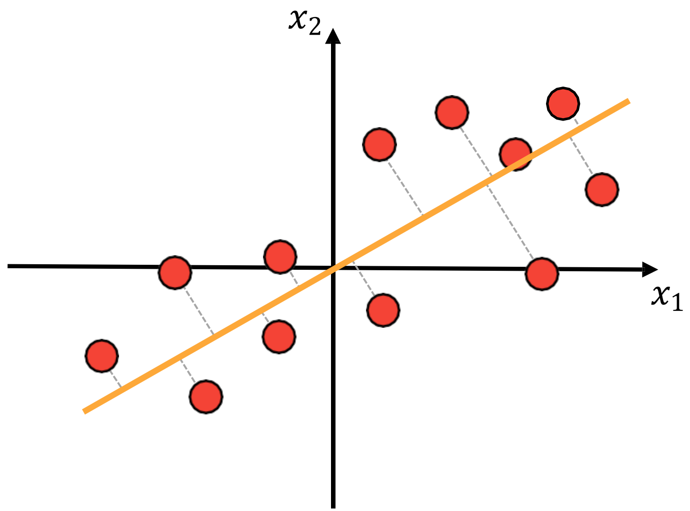


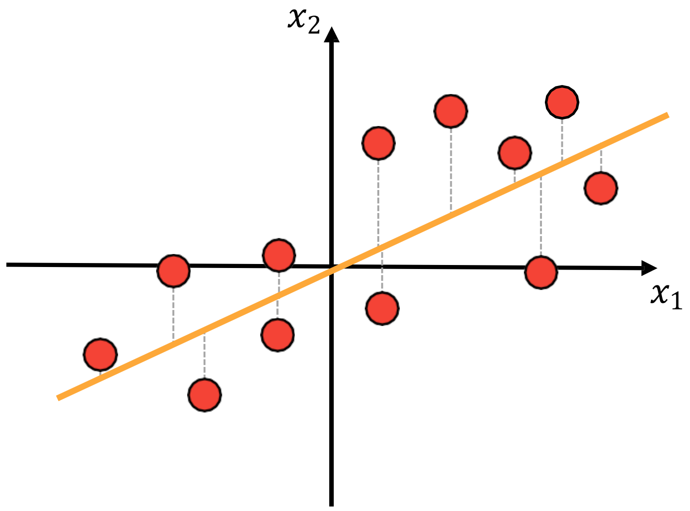
The process would go on for multiple dimensions

PCA versus Linear Regression

PCA finds the line that minimizes the **sum of distances** to the data points

Linear Regression finds the line that minimizes the **sum of squared distances** to the labels





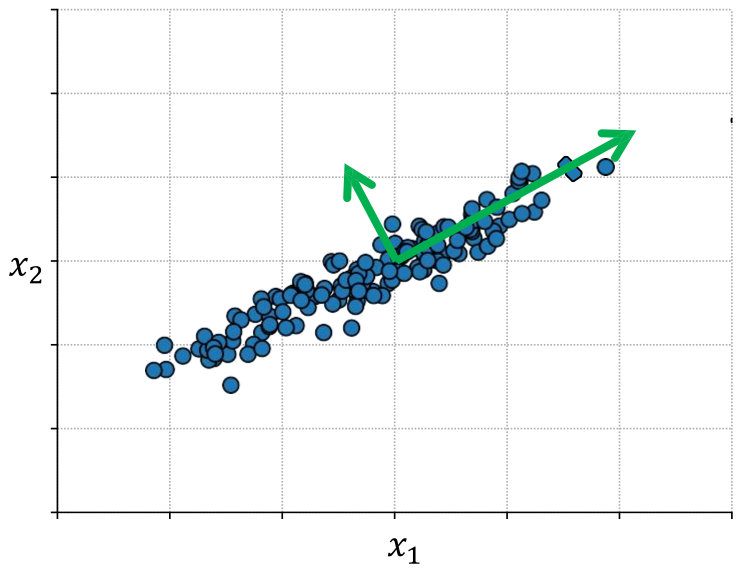
x_1 and x_2 are both features, i.e. independent variables

x_2 is the label, i.e. dependent variable

PCA as Eigen Decomposition

PCA as Eigen Decomposition

- Let $X \in \mathbb{R}^{m \times n}$ be a data set with covariance Σ_X
- Q: How do the eigenvectors of Σ_X look like?
  - ➢ Intuitively, consider Σ_X responsible for the shape of X
  - ➢ Recall that the eigenvectors are the vectors that do not change direction when multiplied by Σ_X
- A: The eigenvectors of the covariance matrix are in fact the principal components of matrix X
  - ➢ The corresponding eigenvalues are equal to the amount of variance explained by each component



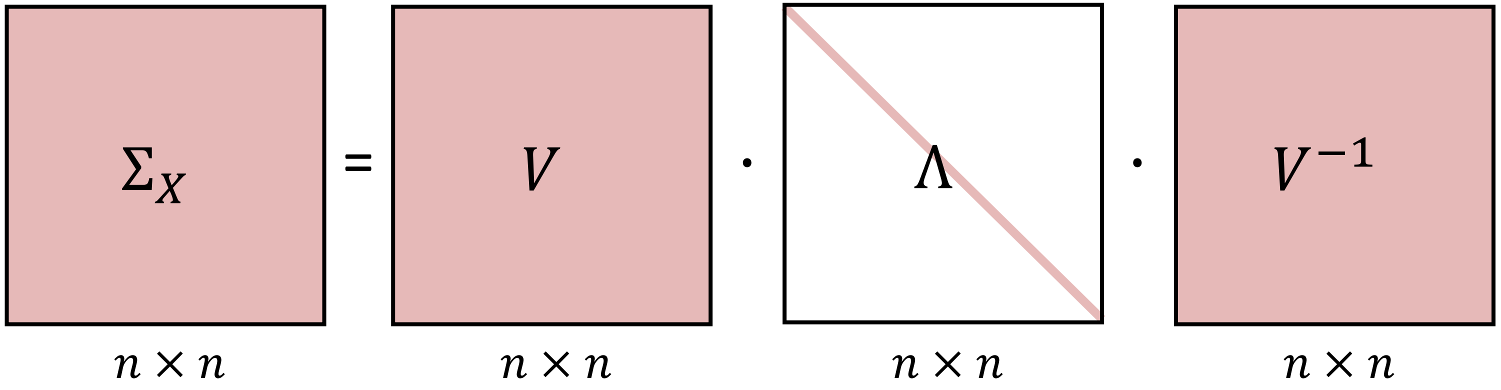
Eigen Decomposition

$\Sigma_X = V \cdot \Lambda \cdot V^{-1}$, where: $V \in \mathbb{R}^{n\times n} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix}$

$v_i \in \mathbb{R}^{n\times 1}$ are column eigenvectors

$$\Lambda \in \mathbb{R}^{n\times n} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

Diagonal eigenvalues



PCA as Eigen Decomposition (Python)

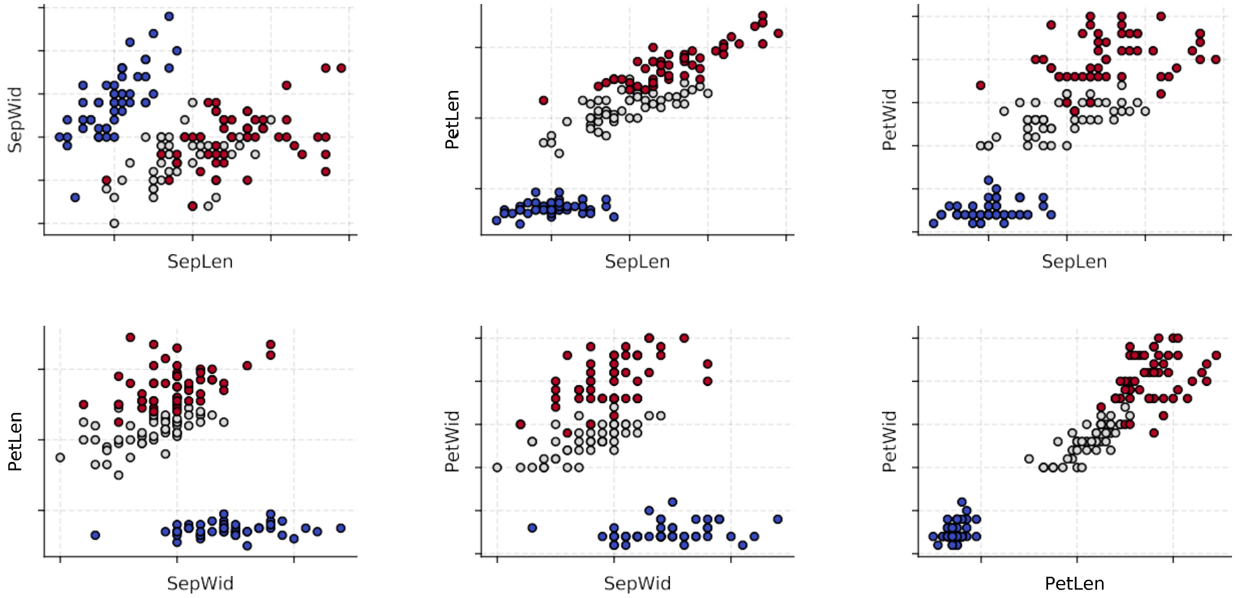
```
def PCA(X,k):
    [m,n] = X.shape
    miu = np.mean(X, axis=0) # compute mean of each feature
    X_bar = X - miu # center X
    cov_X = 1 / (m-1) * np.matmul(X_bar.T, X_bar) # compute covariance
    V, lambdas = np.linalg.eig(cov_X) # apply eigen decomposition
    idx = np.argsort(np.diag(-lambdas)) # sort lambdas descdending
    V_star = V[:,idx[:k]] # permute and keep only first k PCs
    X_star = np.matmul(X,V_star) # apply transformation to X
    return X_star, V_star
```

PCA (Python)

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
# number of components to keep, if ”None” it keeps all components

# find the principal components:
pca.fit(X)

# rotate X into the new coordinate system:
X_pca = pca.transform(X)

# coeficients of the original components to produce the new components:
pca.components_

# variance of projections per component:
pca.explained_variance_

# ratio of explained variance per component:
pca.explained_variance_ratio_
```

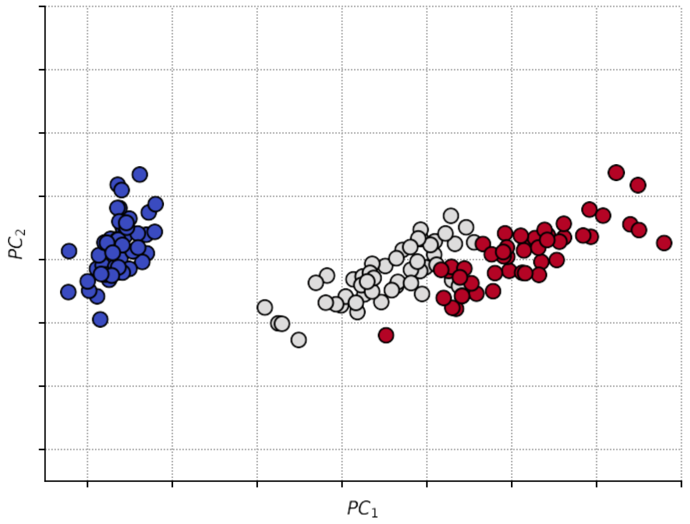
Example

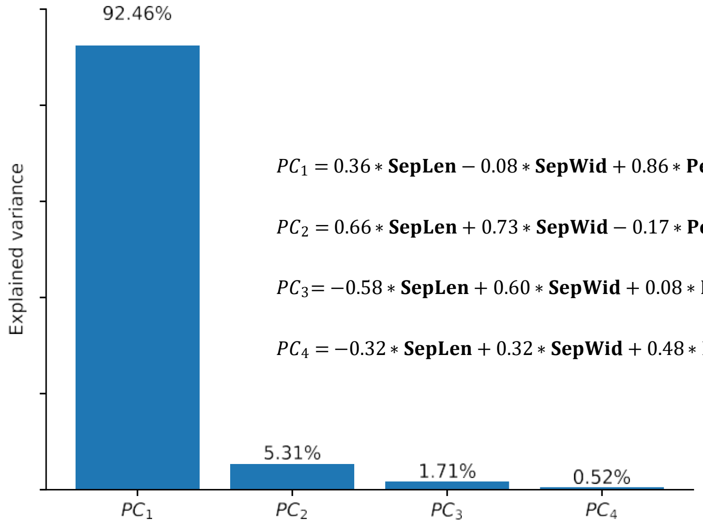
- Iris data set: 150 data samples of flowers with 4 features and 3 classes
  - ➢ $X \in \mathbb{R}^{150\times 4}$
  - ➢ Features: sepal length (cm), sepal width (cm), petal length (cm), petal width (cm)



Example

- Iris data set: 150 data samples of flowers with 4 features and 3 classes
  - $X \in \mathbb{R}^{150\times 4}$
  - Features: sepal length (cm), sepal width (cm), petal length (cm), petal width (cm)





$$PC_1 = 0.36 * \mathbf{SepLen} - 0.08 * \mathbf{SepWid} + 0.86 * \mathbf{PetLen} + 0.36 * \mathbf{PetWid}$$

$$PC_2 = 0.66 * \mathbf{SepLen} + 0.73 * \mathbf{SepWid} - 0.17 * \mathbf{PetLen} - 0.08 * \mathbf{PetWid}$$

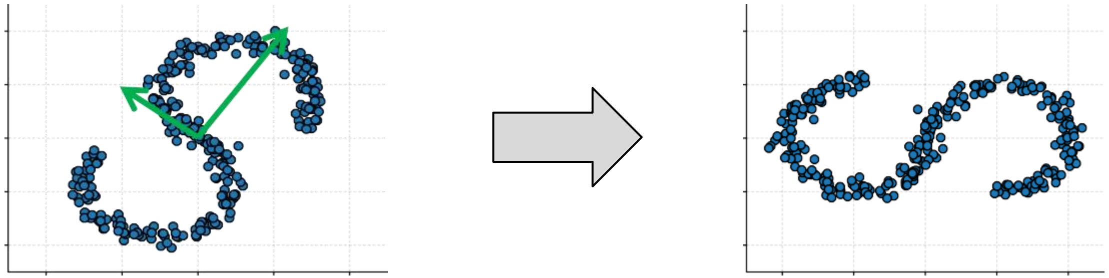
$$PC_3 = -0.58 * \mathbf{SepLen} + 0.60 * \mathbf{SepWid} + 0.08 * \mathbf{PetLen} + 0.55 * \mathbf{PetWid}$$

$$PC_4 = -0.32 * \mathbf{SepLen} + 0.32 * \mathbf{SepWid} + 0.48 * \mathbf{PetLen} - 0.75 * \mathbf{PetWid}$$

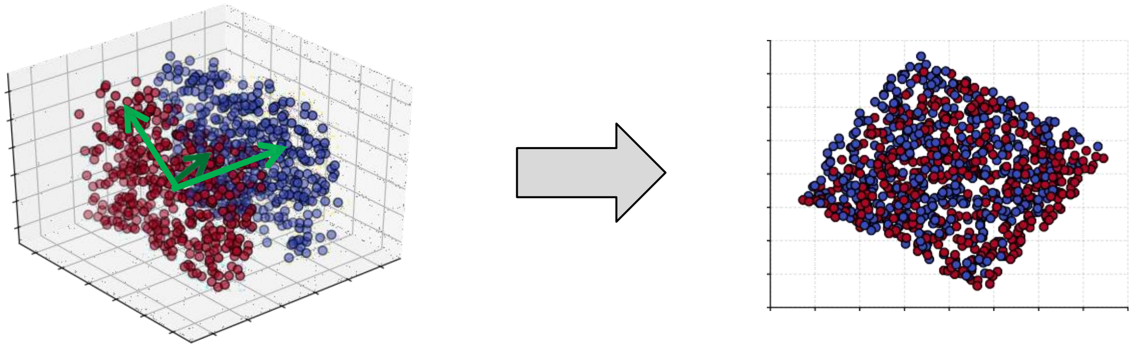
First two components explain almost 98% of variance

Limitations

- PCA does not consider non-linear correlations:



- PCA does not take labels into account, only variance of features:



Summary

- Principal Component Analysis transforms a data set into a new orthogonal coordinate system in which the data is centered and the features are completely uncorrelated
- The directions of the new coordinate system are called principal components
  - PCs are sorted by variance, such that the first PC has the largest variance (it explains most of the data variance)
- Components with low variance can be removed, making PCA a method of dimensionality reduction
- PCA does not treat non-linear correlations and it does not take labels into account

Data Visualization

Problem Formulation

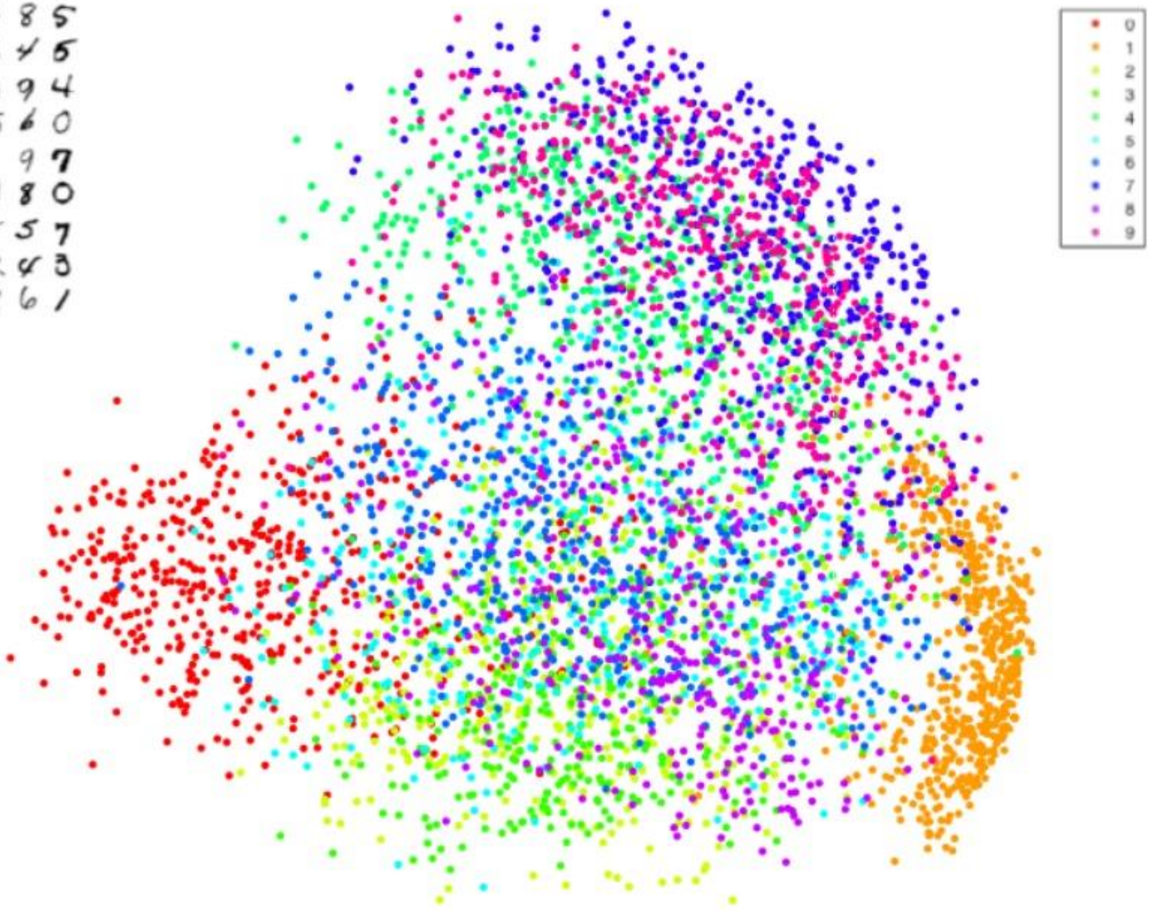
- How can we visualize high dimensional data?
    - ➢ Learn an embedding that preserves as much of the significant structure of the high-dimensional data as possible in the low-dimensional map

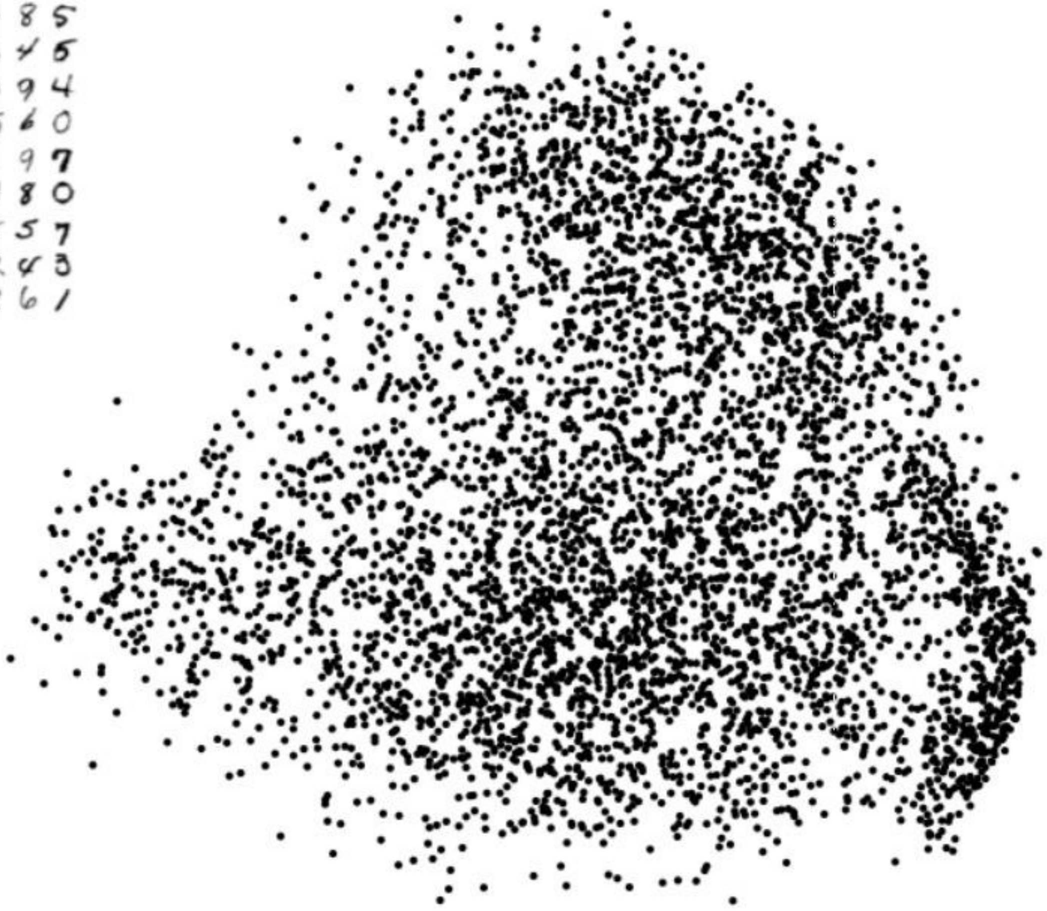
$$\boxed{X} = \{x_1, x_2, \dots, x_n\} \longrightarrow \boxed{Y} = \{y_1, y_2, \dots, y_n\}$$

High-dimensional data set

Two or three-dimensional data

- Why not just PCA?


0
1
2
3
4
5
6
7
8
9

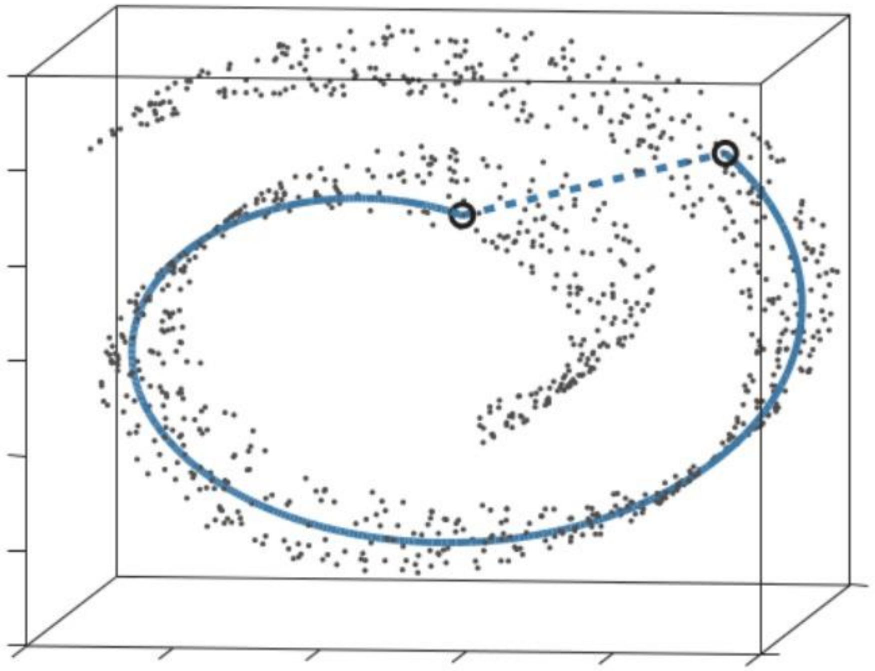


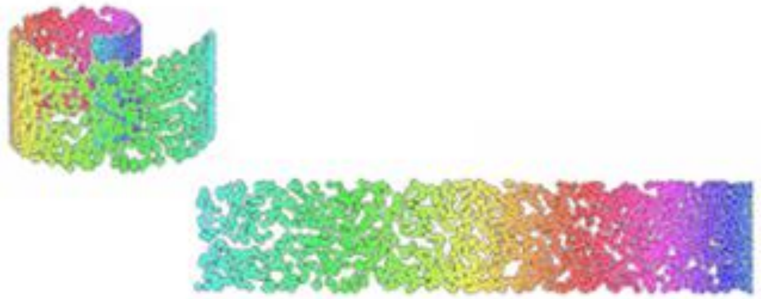
Why not PCA?

- Recall the PCA objective: project data onto a lower dimensional subspace, such that the variance is maximized
- *Why the bad results?*
    1. Linear projection
    2. Mostly preserves distances between dissimilar points
        - ➢ Is this really what we want for the purpose of visualization?

Why not PCA?

- In complex datasets, large distances are usually less indicative

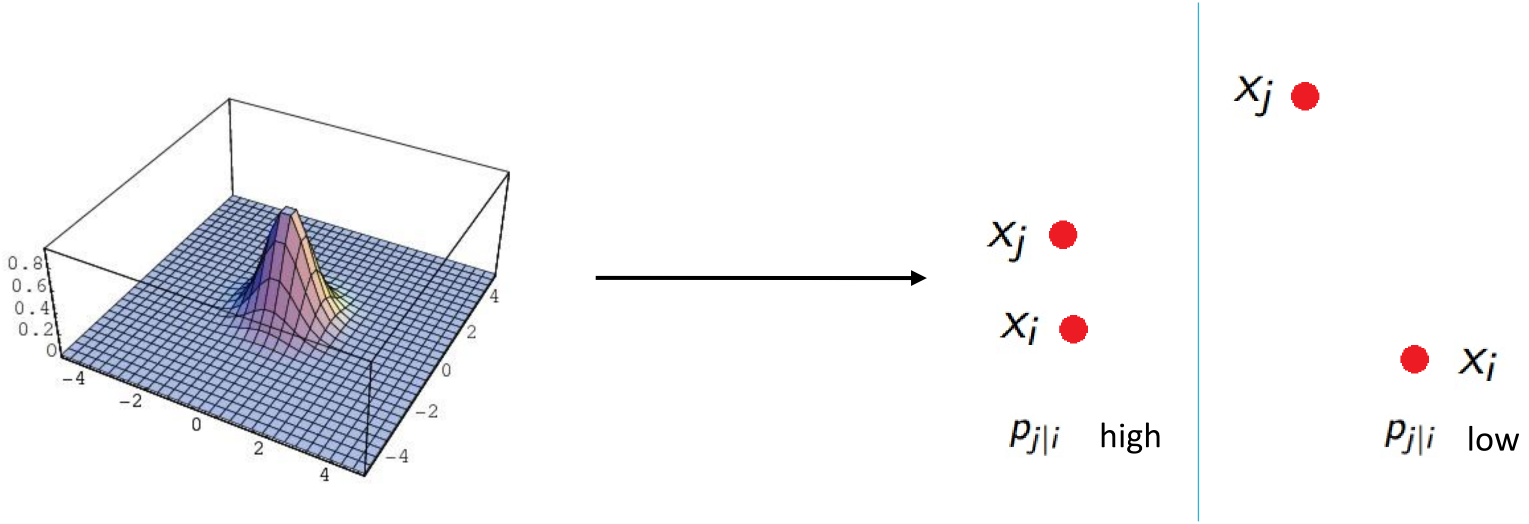




Stochastic Neighbor Embedding

SNE (Stochastic Neighbor Embedding)

- In contrast to PCA, SNE focuses on maintaining the **nearest neighbors** in the lower dimensional map
- Introduced in [Hinton & Roweis, NIPS 2002]
- SNE converts the pairwise Euclidean distances between points into a probability density $p_{j|i}$



SNE (Stochastic Neighbor Embedding)

- Similarity of data points in high-dimension:

$$p_{j|i} = \frac{exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\sigma_i^2}\right)}{\sum_{k\neq i} exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)}$$

- Similarity of data points in low-dimension:

$$q_{j|i} = \frac{exp\left(-\frac{\left\|y_i - y_j\right\|^2}{2\sigma_i^2}\right)}{\sum_{k\neq i} exp\left(-\frac{\|y_i - y_k\|^2}{2\sigma_i^2}\right)}$$

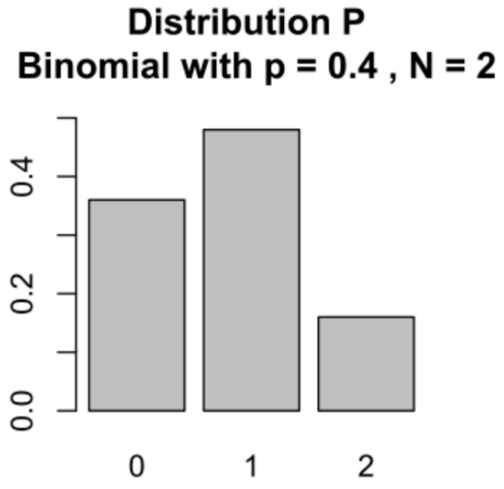
- If we were able to perfectly preserve all similarities in the data, $p_{j|i}$ and $q_{j|i}$ will be equivalent

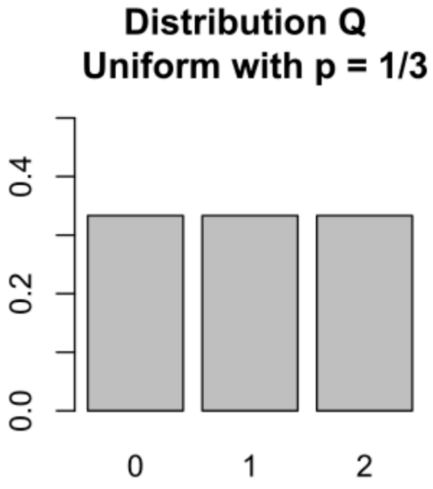
SNE (Stochastic Neighbor Embedding)

- SNE minimizes an objective function that measures the discrepancy between similarities in the data and similarities in the map:

$$C = \sum_i D_{KL}(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \cdot log \frac{p_{j|i}}{q_{j|i}}$$

Kullback-Leibler Divergence





x	0	1	2
Distribution *P*(x)	0.36	0.48	0.16
Distribution *Q*(x)	0.333	0.333	0.333

Kullback-Leibler Divergence

$$\begin{aligned} D_{\mathrm{KL}}(P \parallel Q) &= \sum_{x\in\mathcal{X}} P(x)\ln\left(\frac{P(x)}{Q(x)}\right) \\ &= 0.36\ln\left(\frac{0.36}{0.333}\right) + 0.48\ln\left(\frac{0.48}{0.333}\right) + 0.16\ln\left(\frac{0.16}{0.333}\right) \\ &= 0.0852996 \end{aligned}$$

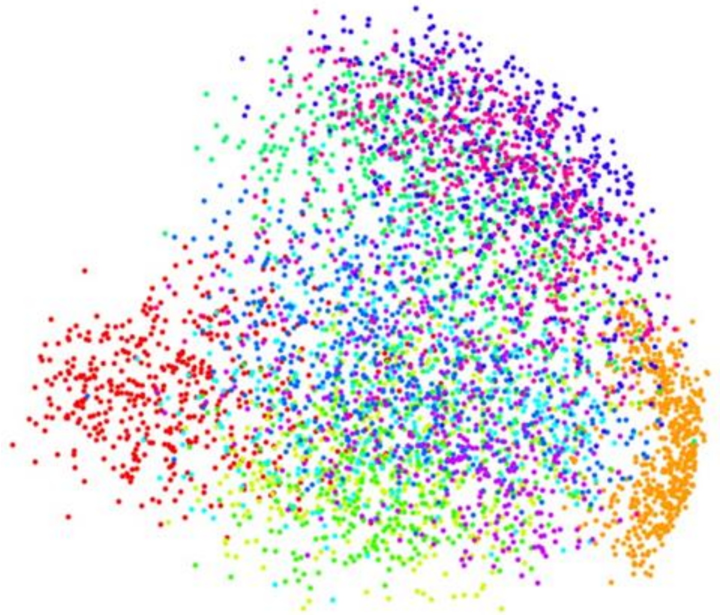
$$\begin{aligned} D_{\mathrm{KL}}(Q \parallel P) &= \sum_{x\in\mathcal{X}} Q(x)\ln\left(\frac{Q(x)}{P(x)}\right) \\ &= 0.333\ln\left(\frac{0.333}{0.36}\right) + 0.333\ln\left(\frac{0.333}{0.48}\right) + 0.333\ln\left(\frac{0.333}{0.16}\right) \\ &= 0.097455 \end{aligned}$$

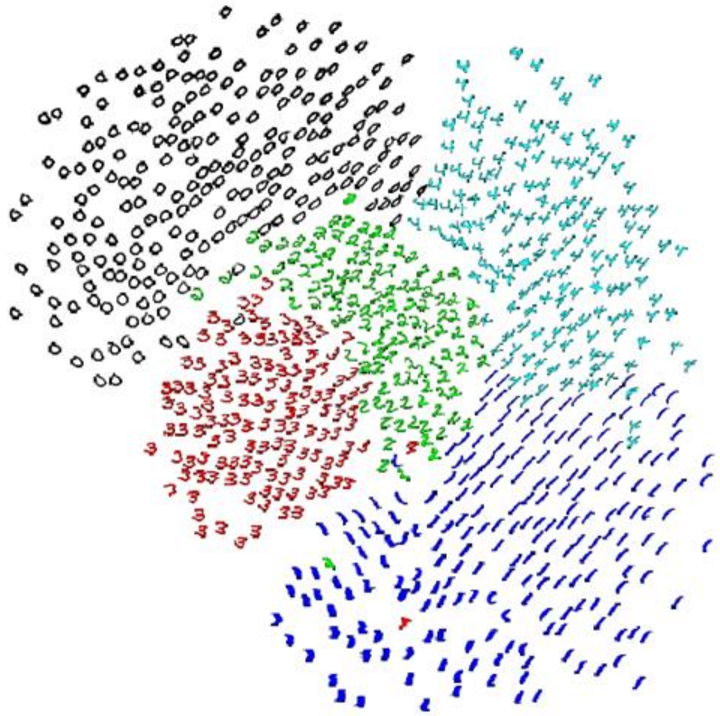
SNE (Stochastic Neighbor Embedding)

$$C = \sum_i D_{KL}(P_i \| Q_i) = \sum_i \sum_j \underbrace{\boxed{p_{j|i}}}_{\text{weight}} \cdot \underbrace{\boxed{log \frac{p_{j|i}}{q_{j|i}}}}_{\text{error term}}$$

- Different errors have a different cost:
  - Close points mapped to far points ($p_{j|i}$ high, $q_{j|i}$ low) $\Longrightarrow$ high penalty
  - Far points mapped to close points ($p_{j|i}$ low, $q_{j|i}$ high) $\Longrightarrow$ low penalty
- SNE focuses on preserving the local structure of the data!

PCA versus SNE


PCA on MNIST *(0-9)*


SNE on MNIST *(0-5)*

Optimization

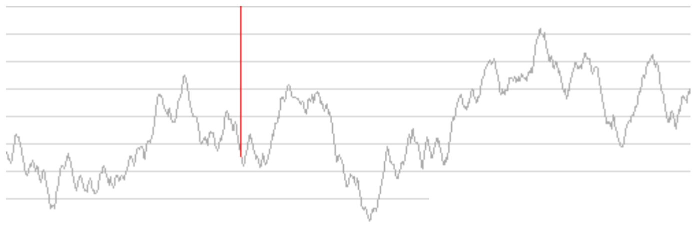
- Gradient:

$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} \left(p_{j|i} - q_{j|i} + p_{j|i} - q_{j|i}\right) \cdot \left(y_i - y_j\right)$$

- Update rule:

$$Y^{(t)} = Y^{(t-1)} + \rho \frac{\partial C}{\partial y_i} + \mu\left(Y^{(t-1)} - Y^{(t-2)}\right)$$

- Optimization is performed using Gradient Descent
- KL-divergence is **not** convex – no guaranties!
- Techniques to avoid "bad" local minimum:
  - ➢ Large momentum ("keep going")
  - ➢ Simulated Annealing (random noise)
- Good visualization requires good choice of parameters

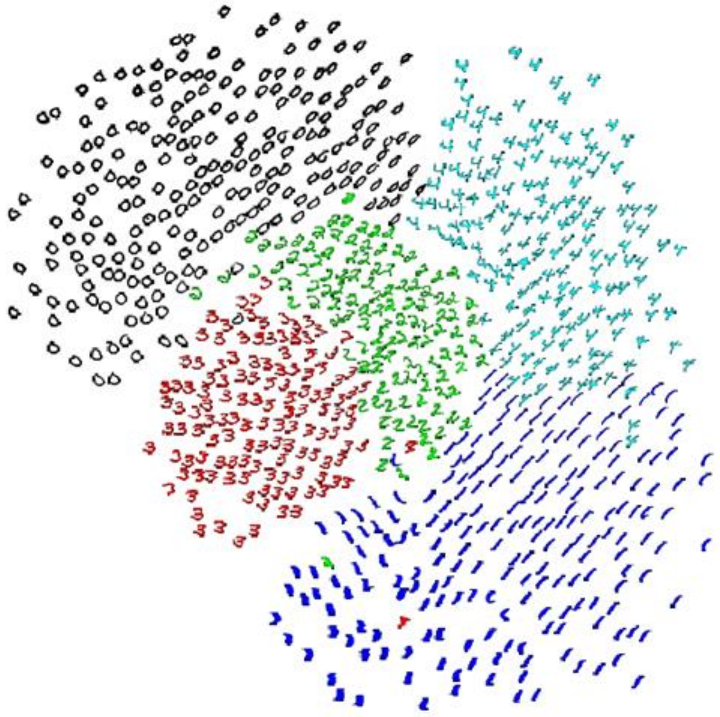


Are we overfitting?

Machine Learning	**versus** **Visualization**
Goal: Generalization	Goal: Visualization
Given a training set, do well on a test set.	We just want to "do well" on our data ("training set")
Overfitting is undesirable	**"Overfitting" is desirable**

Crowding Problem

- The main problem with SNE:
  - ➢ Points tend to *“crowd”* together in the center of the map

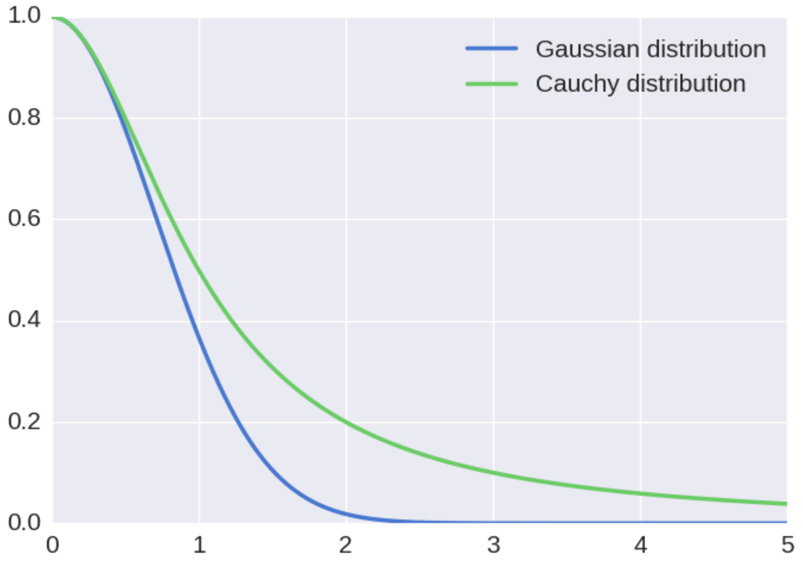


t-SNE

t-SNE

- t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction and visualization technique
- Just as SNE, it maps the data points in a lower-dimensional space, such that points which were close in the original space remain close in the embedding space
  - ➢ It focuses on preserving local structure and does not preserve global structure
  - ➢ The results are non-deterministic
  - ➢ It is computationally expensive
- t-SNE does not produce a transformation of the space, it generates an embedding into a completely new space
- The features in the new space are difficult to interpret

Mismatched tails can compensate for mismatched dimensionalities



$$C = \sum_i D_{KL}(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \cdot log \frac{p_{j|i}}{q_{j|i}}$$

- Cauchy distribution = t-Student with one degree of freedom distribution

https://github.com/oreillymedia/t-SNE-tutorial

t-SNE

- Introduced in [van der Maaten & Hinton, JMLR 2008]
- Main difference:

$$q_{j|i} = \frac{\left(1 + -\left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k \neq l}(1 + -\|y_k - y_l\|^2)^{-1}}$$

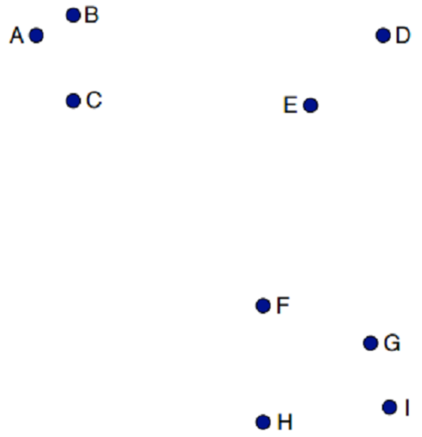
instead of:

$$q_{j|i} = \frac{exp\left(-\frac{\left\|y_i - y_j\right\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} exp\left(-\frac{\|y_i - y_k\|^2}{2\sigma_i^2}\right)}$$

https://github.com/oreillymedia/t-SNE-tutorial

t-SNE gradient interpretation

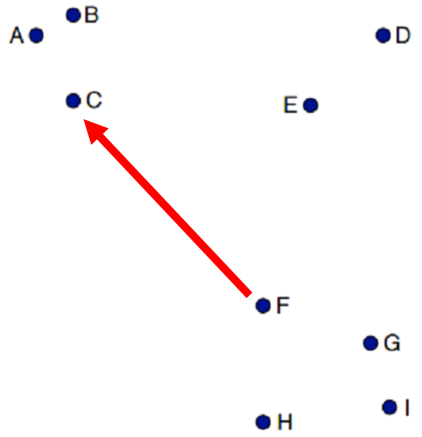
$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{j|i} - q_{j|i}) \cdot \left(1 + \|y_i - y_j\|^2\right)^{-1} \cdot (y_i - y_j)$$



t-SNE gradient interpretation

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} \left(p_{j|i} - q_{j|i}\right) \cdot \left(1 + \left\|y_i - y_j\right\|^2\right)^{-1} \cdot \boxed{\left(y_i - y_j\right)}$$

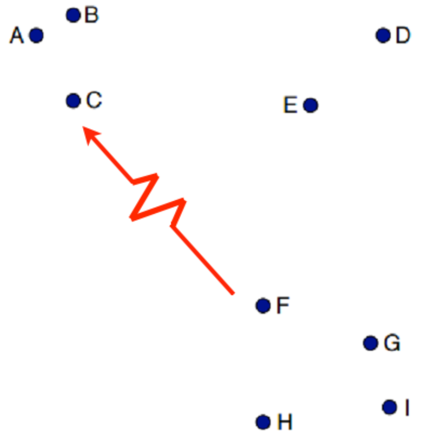
spring



t-SNE gradient interpretation

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} \boxed{(p_{j|i} - q_{j|i}) \cdot \left(1 + \|y_i - y_j\|^2\right)^{-1}} \cdot (y_i - y_j)$$

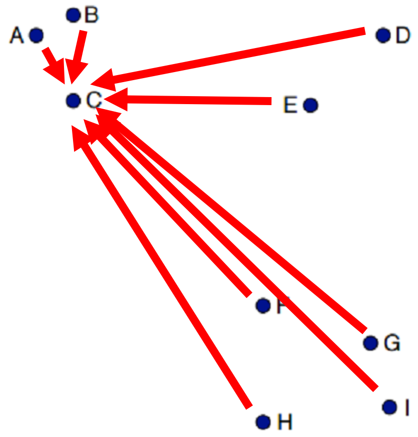
exertion / compression



t-SNE gradient interpretation

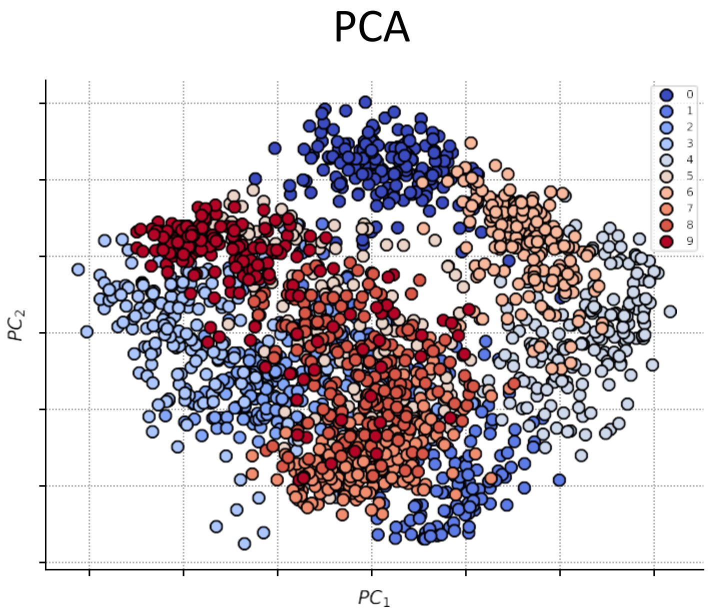
$$\frac{\partial C}{\partial y_i} = 4 \boxed{\sum_{j \neq i}} (p_{j|i} - q_{j|i}) \cdot \left(1 + \|y_i - y_j\|^2\right)^{-1} \cdot (y_i - y_j)$$

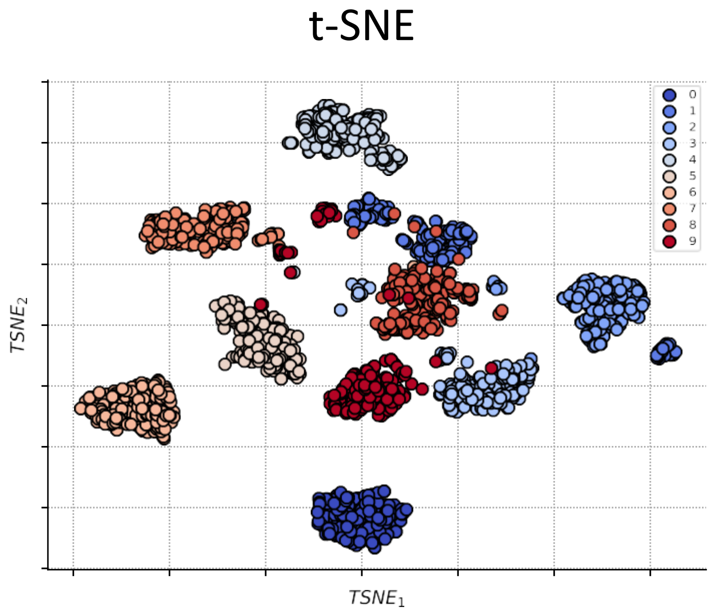
- What is the time complexity?
- $O(n^2)$
- Limited to datasets with only few thousands of points!



PCA versus t-SNE

- Applied on UCI Optical Recognition of Handwritten Digits Data Set
  - ➢ 5620 samples of 8×8-pixel images with 10 classes of handwritten digits



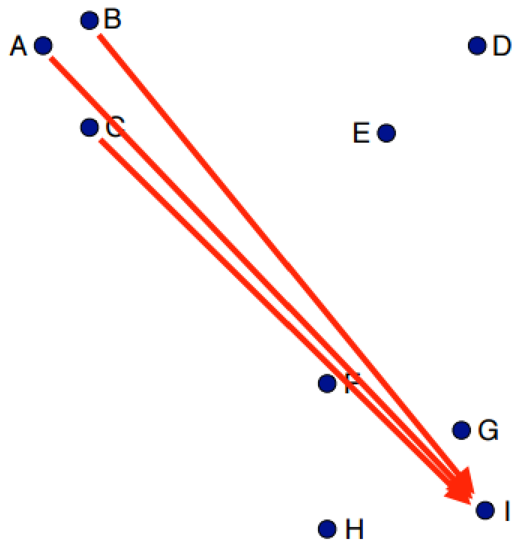


Barnes-Hut-SNE

- Introduced in [van der Maaten, ICLR 2013]
- Implementation of t-SNE with a better time complexity: $O(n \cdot \log n)$
  - ➢ Scales to larger data sets
- It reduce the number of pairwise forces that needs to be computed
  - ➢Idea – forces exerted by a group of points on a point that is relatively far away are all very similar

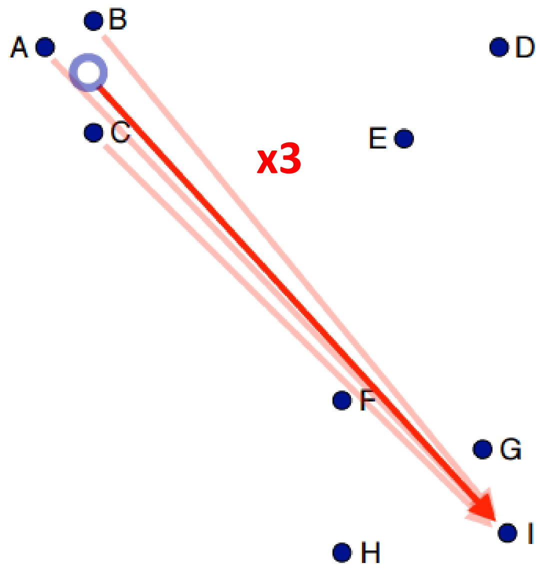
Barnes-Hut Approximation

- Many of the pairwise interactions between points are very similar:

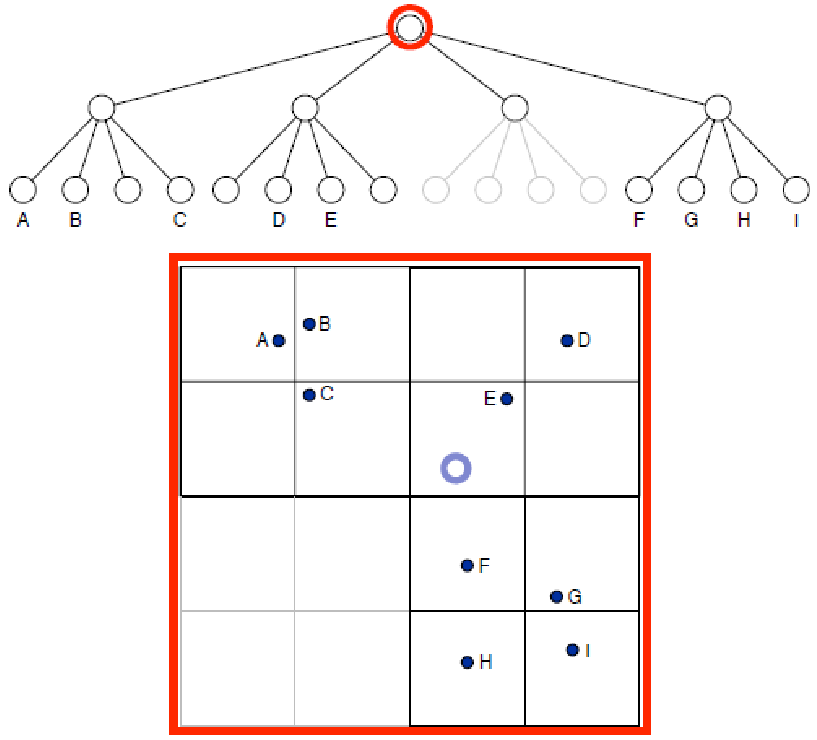


Barnes-Hut Approximation

- Approximate similar interactions by a single interaction

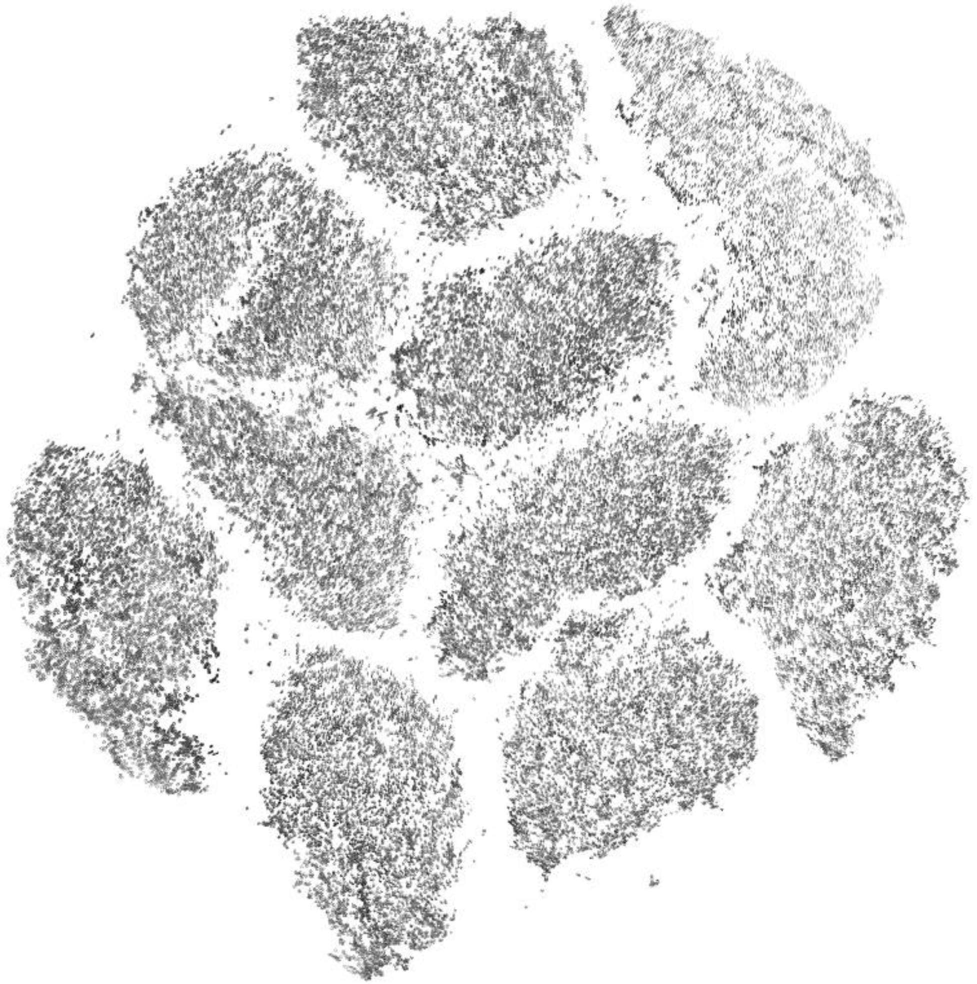


Solution based on Quadtree



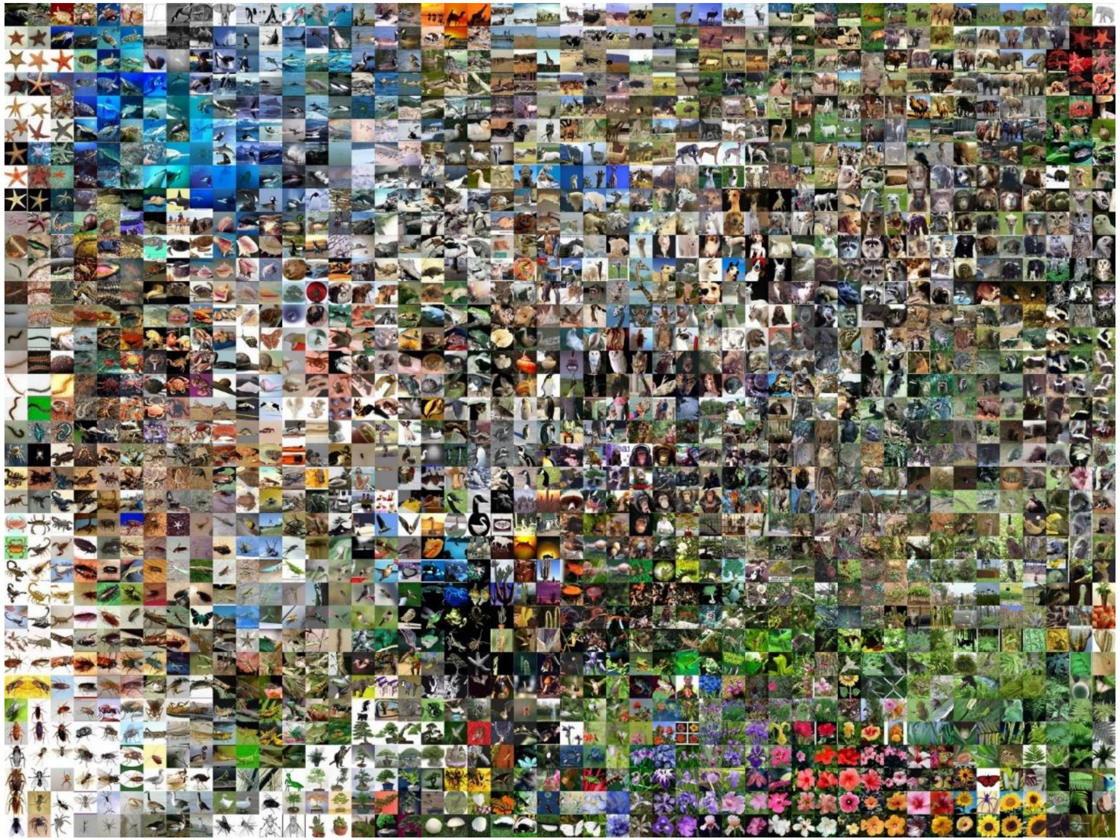
- Barnes-Hut-SNE on all 70,000 MNIST images

- Improvement in run time from days to ~10 minutes



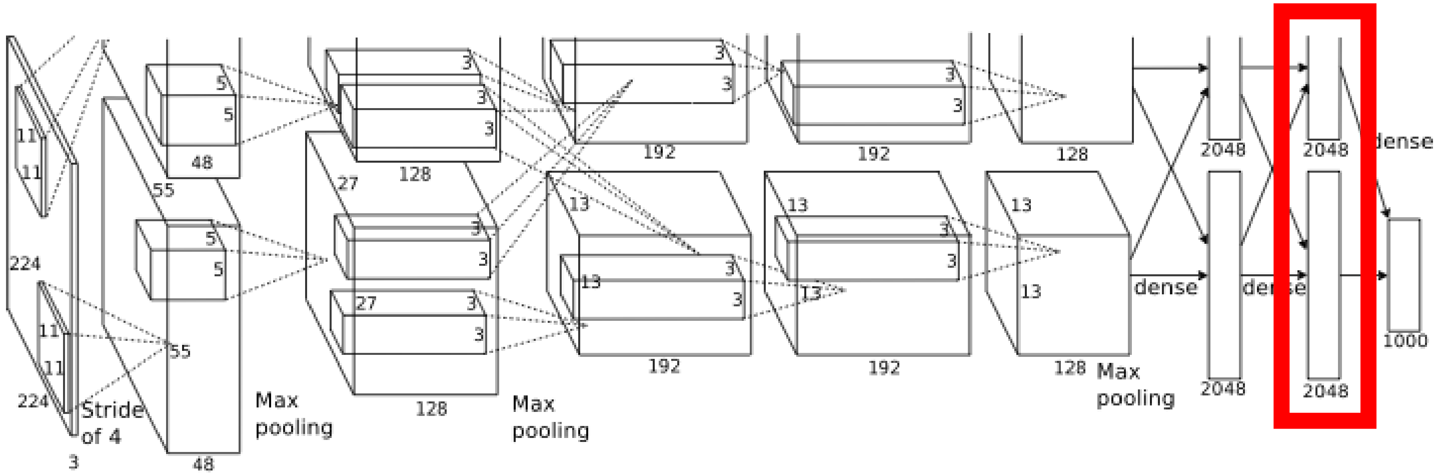
t-SNE on AlexNet embeddings

ImageNet



t-SNE on AlexNet embeddings

AlexNet



224
224
11
11
11
11
Stride of 4
3
5
5
5
5
55
55
48
48
Max pooling
27
27
3
3
3
3
3
3
128
128
Max pooling
13
13
192
192
192
192
13
13
128
128
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000