



Machine Learning

LINEAR TRANSFORMATIONS & DIMENSIONALITY REDUCTION

Cigdem Beyan
A. Y. 2024/2025



TODAY

- Problems of high dimensional data
 - Curse of dimensionality
 - Overfitting
- Linear Dimensionality Reduction Methods
 - Principal Component Analysis
 - Fisher Linear Discriminant Analysis
- Non-Linear Dimensionality Reduction Methods



- In practical problems you may find training data in various forms:
 - Few samples
 - Few samples and many features
 - Fixed features that cannot be selected
 - Not-independent features
- There are two important things to consider for designing a classifier:
 - **Computational complexity** of the system
 - The **influence of the dimensionality** (the number of features) of the training set on the performance of the classifier.

- If the features are statistically **independent**, then it is more likely to achieve the optimal performance.
- In general, if the performance is not in the desired level, you can **add features**, but this brings in the cost of complicating the classifier and the feature extractor.
- If **adding features** leads to **worse** performance, then this could be because:
 - Choosing an **incorrect model** (e.g., Gaussian hypothesis or conditional probability hypothesis)
 - **Insufficient number of samples.**

COMPLEXITY

- **Complexity** (in terms of *running time*) **increases** when number of features aka **dimensionality d** **increases**.
- A lot of methods already have at least **$O(nd^2)$** complexity, when **n** is the number of samples.
 - For example, covariance matrix calculation
- This, as **d** increases, **$O(nd^2)$** complexity may be really costly.

CURSE OF DIMENSIONALITY

- If \mathbf{d} is large, and the number of samples \mathbf{n} **is too small** (or in some cases even smaller), then the problem of **curse of dimensionality** appear.
 - We cannot perform accurate parameter estimation, classification,...
- *For example, in the case of covariance matrix calculation, for accurate estimation, \mathbf{n} should be much bigger than \mathbf{d}^2 .*
 - Otherwise, the model would be too complicated for the data and **overfitting** occurs.



CURSE OF DIMENSIONALITY: OVERFITTING

- Overfitting occurs when a model learns to **perform well on the training data** but **fails to generalize to unseen data**.
- When **d** is high, the data become increasingly sparse and spread out in the high-dimensional space. This can lead to overfitting because
 - The model may learn patterns that are specific to the training data.
 - The model has more freedom to fit the noise (e.g., irrelevant features) in the training data rather than the underlying patterns.

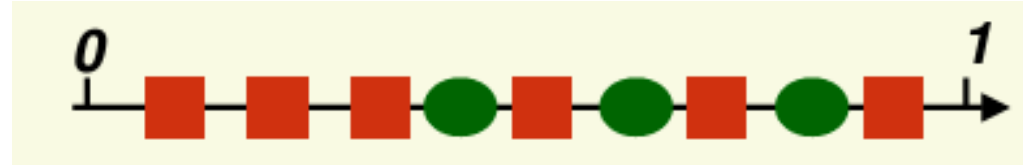


CURSE OF DIMENSIONALITY: OVERFITTING

- To mitigate the effects of overfitting and the curse of dimensionality,
 - **Reduce the dimensionality** of features (TODAY)
 - Combine features in some way: **feature extraction**
 - Apply **feature selection**
 - Apply **regularization**
- At this point it is important to **reduce the dependence of the features**.

CURSE OF DIMENSIONALITY: EXAMPLE

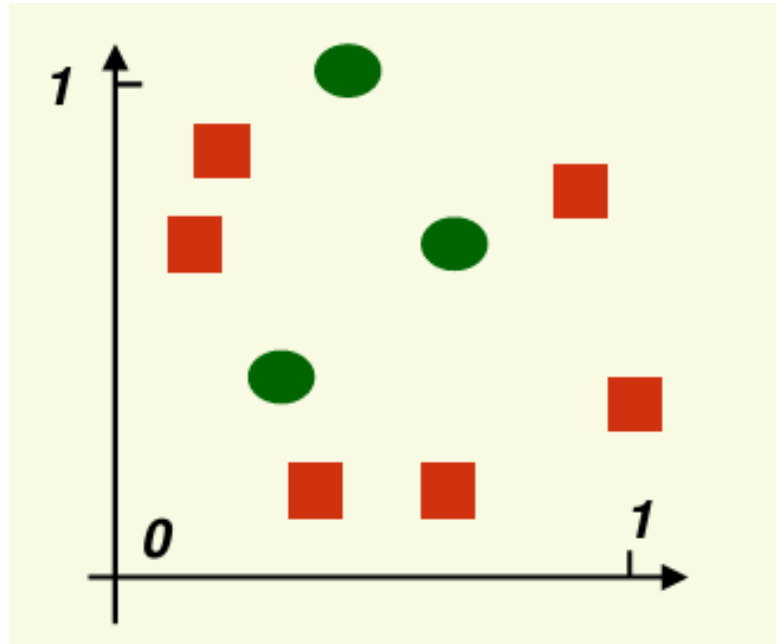
- Suppose we use k-NN classifier when $k=1$
- If we have only 1 feature and 9 samples



- This feature is not discriminative, i.e., it does not separate the classes well.
- What about using 2 features?
- But, keep in mind that to perform well k-NN requires dense samples (not sparse data) and a lot of samples.
- To maintain the same density as in 1D case, we need to add 9^2 .

CURSE OF DIMENSIONALITY: EXAMPLE

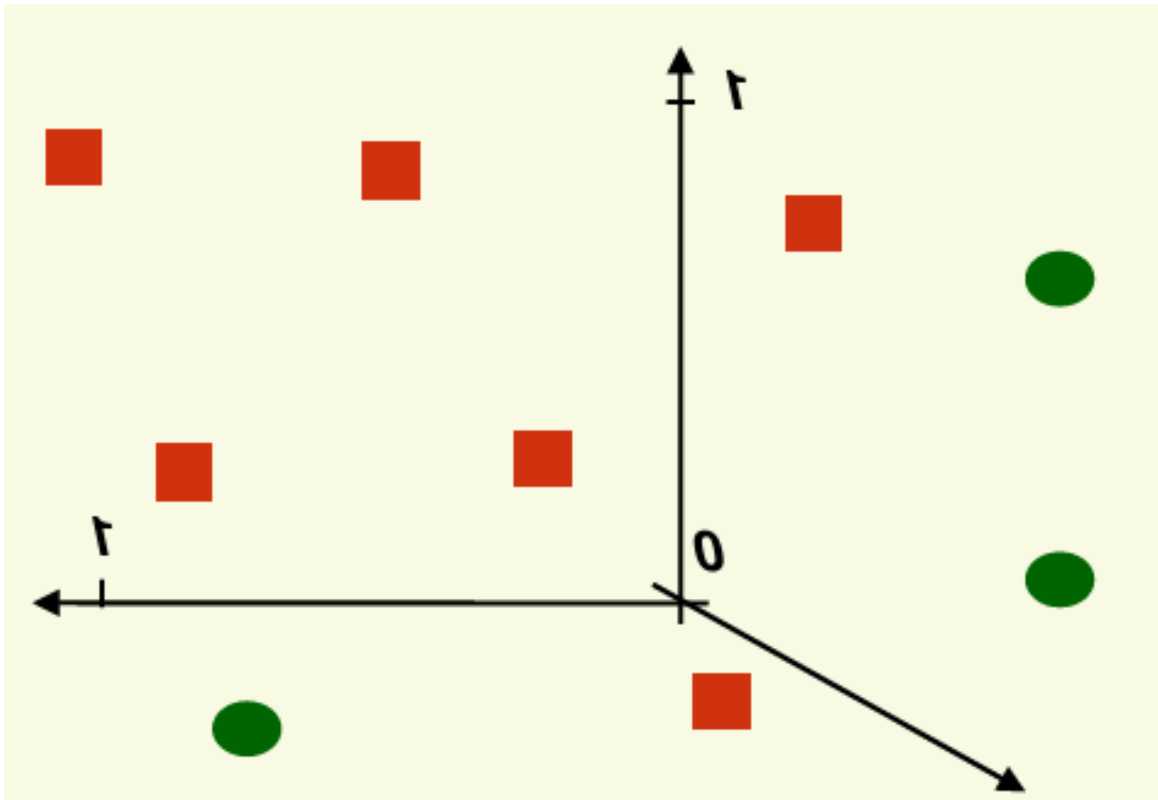
- We still have 9 samples, but we have now 2 features
- This feature space is too sparse to make 1-NN work well.



- Shall we add more features? What if we go to 3D?

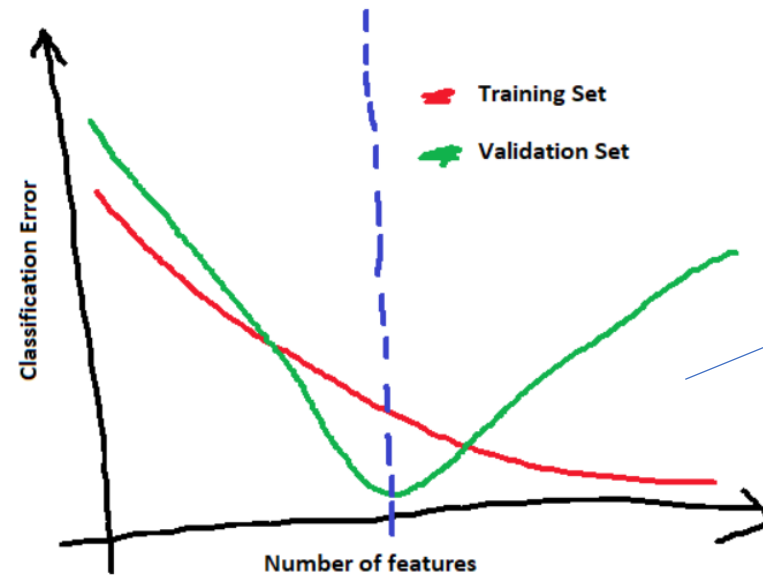
CURSE OF DIMENSIONALITY: EXAMPLE

- With 3 features and still 9 samples, the things became worse for 1-NN algorithm:



- To obtain the same dense data as in 1D case with 9 samples, for 3D we need 9^3 samples!
- This shows the importance of having more data samples, especially when the dimensionality increases.

- Overall, if n samples is dense enough in **1D**
- Then, in d dimensions, we need n^d samples
 - Pay attention that n^d grows really too fast as a function of d
- For a fixed number of samples, as we add features, the classification error becomes:





DIMENSIONALITY REDUCTION

- Aim: represent samples with fewer features
 - While preserving the **structure** of the data as much as possible
 - **Discriminative**: keep only structure that affects class separability
- We will construct a new set of dimensions, e.g., by the linear combination of original dimensions.
 - Combine in somehow old features to create new features.
 - Ideally the new feature vector should retain all important information from the old feature vector.

DIMENSIONALITY REDUCTION

Formal definition

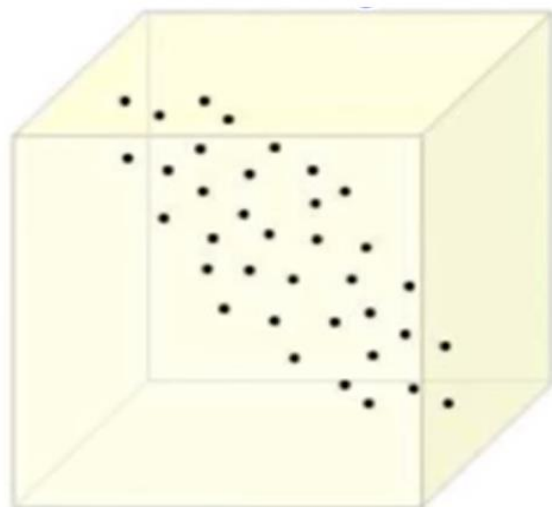
- We look for a transformation **W** that leads from poorly structured **y** samples into a new simpler, but more discriminant feature set **x** for classification. $\rightarrow \mathbf{x} = \mathbf{W} \mathbf{y}$
 - **X** represents the new features with k data points ($k \times m$)
 - **Y** represents the original features with k data points ($k \times n$)
 - $n \gg m$



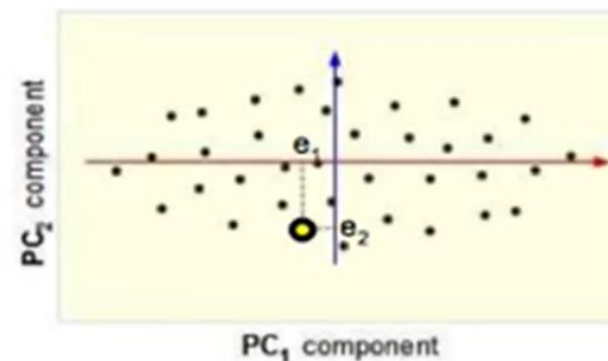
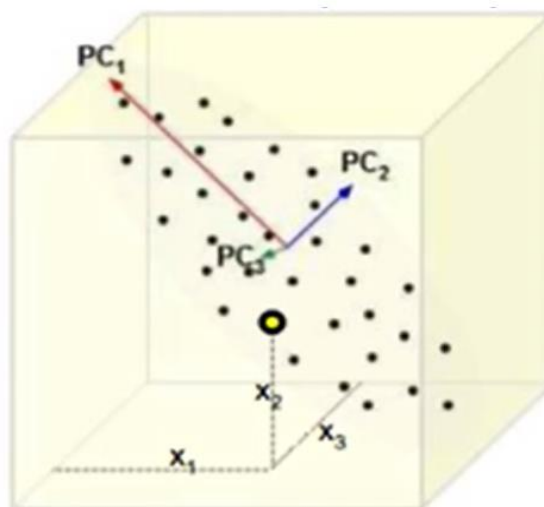
PRINCIPAL COMPONENT ANALYSIS (PCA)

- Perhaps the most frequently used method to reduce the dimensionality.
- PCA defines a set of principal components
 - 1st component: direction of the **greatest variability** (largest variance) in the data
 - 2nd component: perpendicular to 1st, greatest variability of what is left
 - And so on until *original dimensionality*
- Once you obtain the new feature representations (new dimensions)
 - Change the coordinates of every data point to these new dimensions, then apply the ML algorithm
 - Note that the new feature representations has the same variance as old features. *PCA preserves the largest variances in the data.*

PRINCIPAL COMPONENT ANALYSIS (PCA)



3D



2D



WHEN TO APPLY PCA?

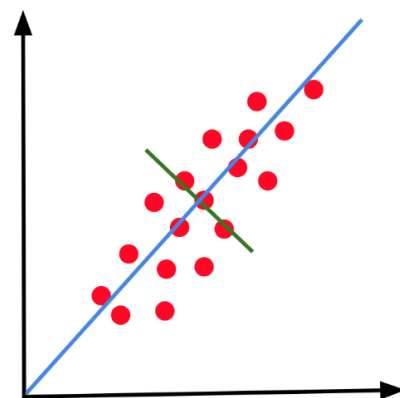
- 1) Do you want to reduce the number of variables, but aren't able to identify variables to completely remove from consideration?
- 2) Do you want to ensure your variables are independent of one another?
- 3) Are you comfortable making your independent variables less interpretable?

If you answered “yes” to all three questions, then PCA is a good method to use.

If you answered “no” to question 3, you **should not** use PCA.

PRINCIPAL COMPONENT ANALYSIS (PCA)

- What is the direction of largest variance in data?
 - If \mathbf{x} has multivariate distribution $\mathbf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, direction of largest variance is given by **eigenvector** corresponding to **the largest eigenvalue** of $\boldsymbol{\Sigma}$.



- Therefore, we need to be looking at the covariance matrix of the data.
- PCA can be applied to distributions also other than Gaussian.

PCA ALGORITHM

- **Step 1:** Standardize the given data
 - Subtract the **mean** of each feature from the data. (The new data has zero mean.)
- **Step 2:** Calculate the **covariance matrix** of the standardized data.
- **Step 3:** Compute the **eigenvectors** and **eigenvalues** of the covariance matrix.
- **Step 4:** Sort the **eigenvalues** in descending order and choose the **top k eigenvectors** as **principal components**.
- **Step 5:** Multiply the standardized data matrix obtained in Step 1 by the selected principal components matrix. In this way, you obtained the new features with k number of features.

PCA WITH MATH

- Given the training data $\mathbf{x} = \{\mathbf{x}_1^t, \mathbf{x}_2^t, \dots, \mathbf{x}_M^t\}$ with \mathbf{M} number of samples with n dimensions
- The average is defined as $\mathbf{m}_x = E\{\mathbf{x}\}$ where $E\{.\}$ is the Expected Value operator

$$\mathbf{m}_x = \frac{1}{M} \sum_{k=1}^M \mathbf{x}_k$$

- The covariance matrix of this training data is defined as

$$\mathbf{C}_x = E\{(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^t\} \quad \mathbf{C}_x = \frac{1}{M} \sum_{k=1}^M (\mathbf{x}_k - \mathbf{m}_x)(\mathbf{x}_k - \mathbf{m}_x)^t = \frac{1}{M} \sum_{k=1}^M \mathbf{x}_k \mathbf{x}_k^t - \mathbf{m}_x \mathbf{m}_x^t$$

- \mathbf{C}_x has the size of $n \times n$;
- Elements c_{ii} are the **variance** of the i -th component of the set of \mathbf{x} ;
- c_{ij} are the related **covariances** between components i and j ;

PCA WITH MATH

- $\mathbf{C}_{\mathbf{x}}$ is real and symmetrical;
 - if \mathbf{x}_i and \mathbf{x}_j are unrelated, then $c_{ij} = c_{ji} = 0$.
- Since $\mathbf{C}_{\mathbf{x}}$ is real and symmetrical, one can always find a set of **n orthonormal eigenvectors**
 - Let \mathbf{e}_i and λ_i , $i=1,2,\dots,n$, be the **eigenvectors** and the related **eigenvalues** of $\mathbf{C}_{\mathbf{x}}$, respectively, ordered in descending order, ie.,
 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$



PCA WITH MATH

- Let **A** be a matrix whose rows are formed by the eigenvectors of **C_x** ordered as above, and use it to transform vectors **x** as follows

$$\mathbf{y} = \mathbf{A} (\mathbf{x} - \mathbf{m}_x)$$

- The average of the vectors **y** is zero and the covariance matrix:

$$\mathbf{m}_y = \mathbf{0}$$

$$\mathbf{C}_y = \mathbf{A} \mathbf{C}_x \mathbf{A}^T$$

PCA WITH MATH

- In addition, \mathbf{C}_y is a diagonal matrix with the eigenvalues of \mathbf{C}_x in the diagonal.

$$\mathbf{C}_y = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$$

- The elements of \mathbf{y} are therefore decorrelated
- Since the rows of \mathbf{A} are orthonormal vectors, then $\mathbf{A}^{-1} = \mathbf{A}^T$ and each vector \mathbf{x} can be calculated by \mathbf{y} :

$$\mathbf{y} = \mathbf{A} (\mathbf{x} - \mathbf{m}_x) \rightarrow \mathbf{x} - \mathbf{m}_x = \mathbf{A}^T \mathbf{y} \rightarrow \mathbf{x} = \mathbf{A}^T \mathbf{y} + \mathbf{m}_x$$

- Suppose you build \mathbf{A} with the first k (larger) eigenvectors, $k < n$.

PCA WITH MATH

- Then \mathbf{A}_k is a matrix of dimension $k \times n$ and vectors \mathbf{y} have dimension k .
- Reconstructed vectors \mathbf{x} will no longer be accurate and are given by:

$$\hat{\mathbf{x}} = \mathbf{A}_k^T \mathbf{y} + \mathbf{m}_x$$

PCA EXAMPLE

- Let's assume that we drive all the steps in the PCA algorithm and obtained the following eigenvalues and eigenvectors.

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

- Herein the smaller eigenvalues has **less significance**, therefore, if we ignore it and obtain the new data only with the other eigenvalue's(1.2840....) eigenvectors, then we loose some information, but not much.



HOW MANY PRINCIPAL COMPONENTS?

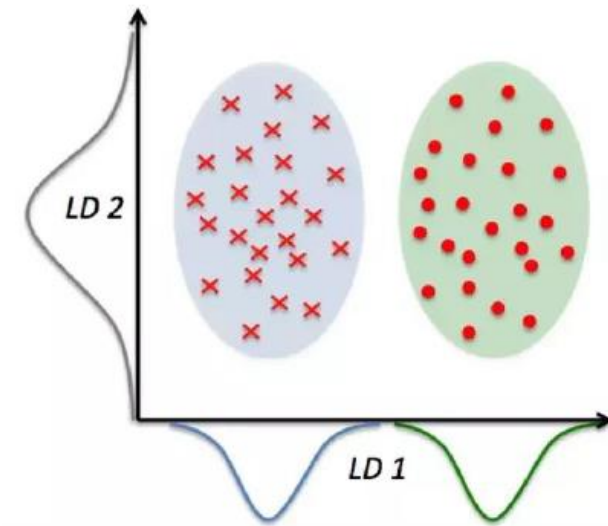
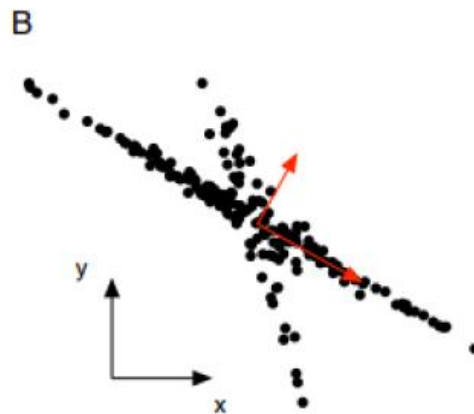
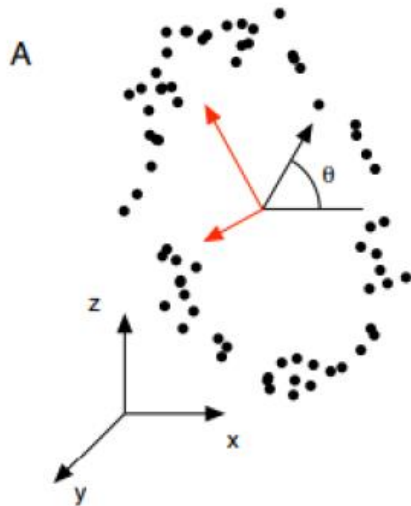
- In the previous example, we had only 2 features (2 dimensional data), then if we want to drop some of the dimensions, it is trivial to represent the data in the form of one dimension.
- However, if we have high number of dimensions (even anything more than 2) then
 - how to decide the number of principal components (aka the new dimension of the data)?

HOW MANY PRINCIPAL COMPONENTS?

- Depends on the goal and the application
- No means of validating it, unless we use it in the context of a **supervised method** (e.g., reduce the input dimensionality for a supervised algorithm)
- Nonetheless we can compute the cumulative proportion of eigenvalues for the first k *principal components* allowing us to estimate the amount of information loss.
 - As a threshold one can use 90%, 95%, etc.

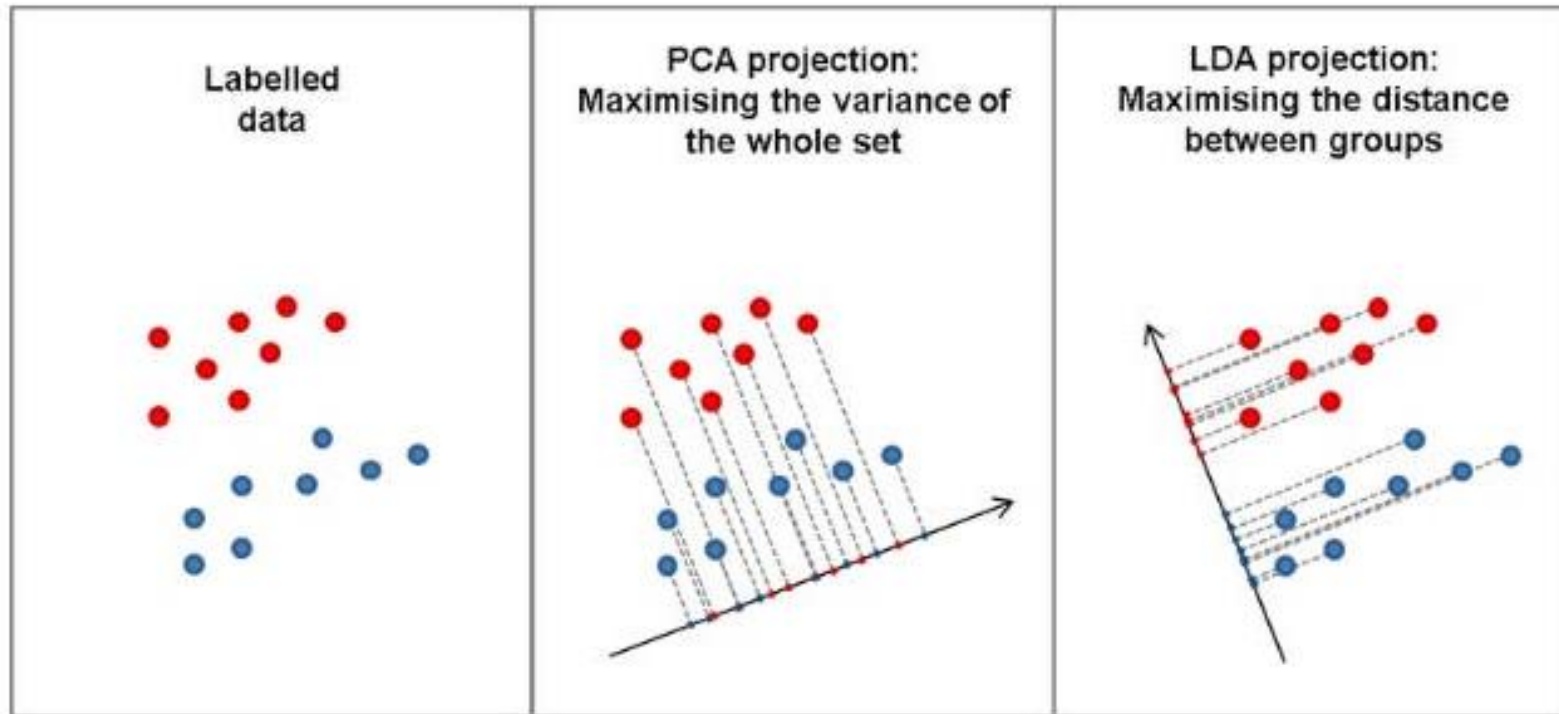
PCA DRAWBACKS

- It assumes that the variables are **linearly correlated**, in many cases, we already know that they are not.
- Only the axes with the **greatest variance** are considered, there are some cases in which this involves huge mistakes.



FISHER LINEAR DISCRIMINANT

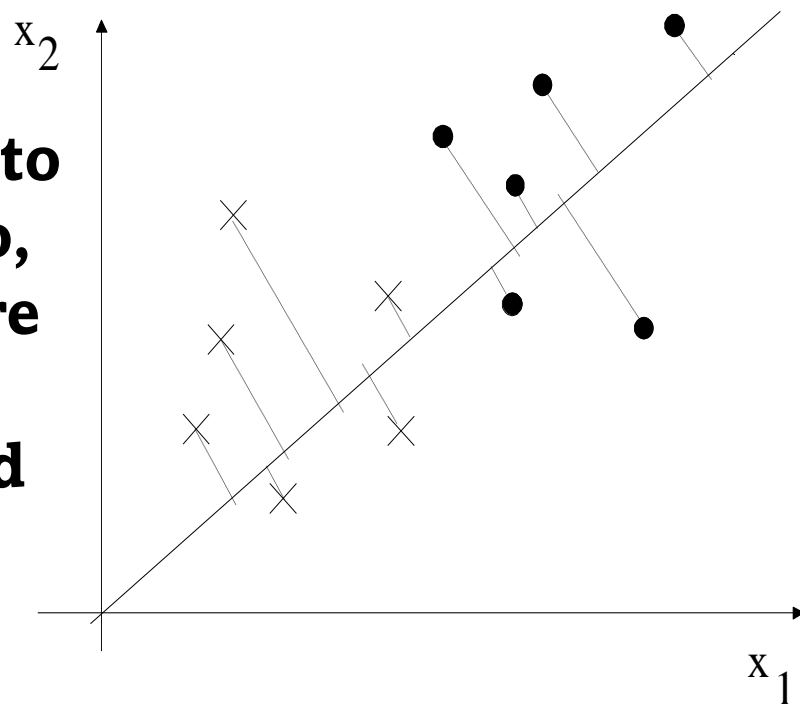
- Also called **Linear Discriminant Analysis**
- **Main idea:** Finding a projection to a line such that data samples from different classes are well separated.



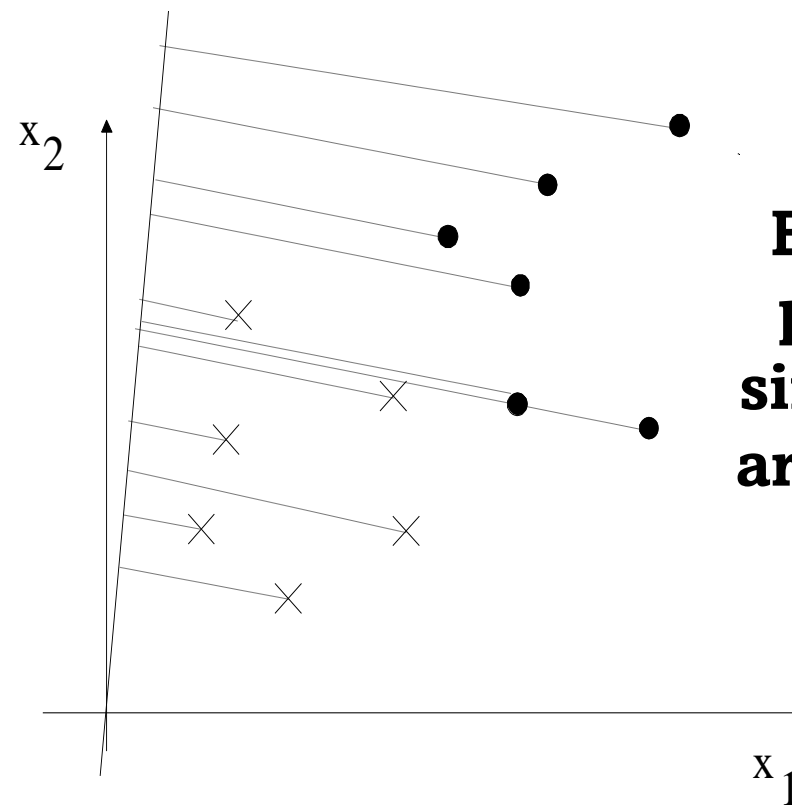
FISHER LINEAR DISCRIMINANT

- Also called **Linear Discriminant Analysis**
- **Main idea:** Finding a projection to a line such that data samples from different classes are well separated.

Good line to project to, classes are well separated

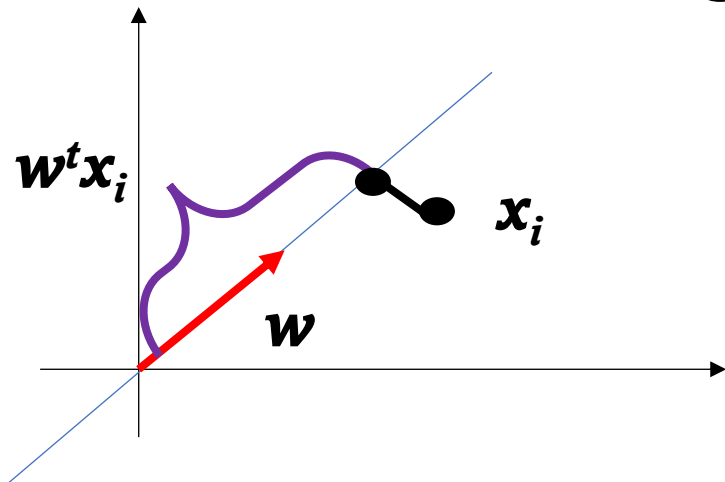


Bad line to project to, since classes are mixed up



FISHER LINEAR DISCRIMINANT

- Suppose we have 2 classes and **d**-dimensional samples: $\mathbf{x}_1, \dots, \mathbf{x}_n$ where
 - n_1 samples come from the first class
 - n_2 samples come from the second class
- Consider projection on a line
- Let the line direction is given by unit vector **w**



- Scalar $\mathbf{w}^t \mathbf{x}_i$ is the distance of projection of \mathbf{x}_i from the origin.
- This $\mathbf{w}^t \mathbf{x}_i$ is the projection of \mathbf{x}_i into a one dimensional subspace.

FISHER LINEAR DISCRIMINANT

- Therefore, the projection of sample \mathbf{x}_i onto a line in direction \mathbf{w} is given by $\mathbf{w}^t \mathbf{x}_i$
- We also need to measure the separation between projections of different classes, since we aim to discriminate them.
 - The **difference** in the **means** of the samples is considered as a measure of separation.

$$\tilde{m}_1 = \mathbf{w}^t \cdot \mathbf{m}_1$$

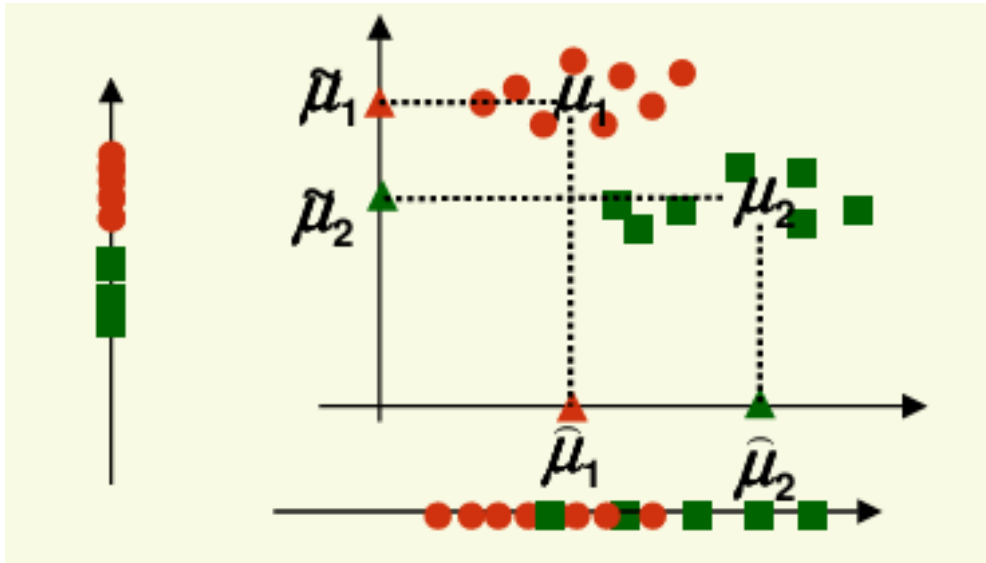
$$\tilde{m}_2 = \mathbf{w}^t \cdot \mathbf{m}_2$$

$$\left. \begin{aligned} \mathbf{m}_1 &= \frac{1}{N_1} \cdot \sum_{i=1}^{N_1} \mathbf{x}_i^{(1)} \\ \mathbf{m}_2 &= \frac{1}{N_2} \cdot \sum_{i=1}^{N_2} \mathbf{x}_i^{(2)} \end{aligned} \right\} \text{means before the transformation}$$

$$\left. \begin{aligned} \tilde{m}_1 &= \frac{1}{N_1} \cdot \sum_{i=1}^{N_1} y_i^{(1)} \\ \tilde{m}_2 &= \frac{1}{N_2} \cdot \sum_{i=1}^{N_2} y_i^{(2)} \end{aligned} \right\} \text{means after the transformation}$$

FISHER LINEAR DISCRIMINANT

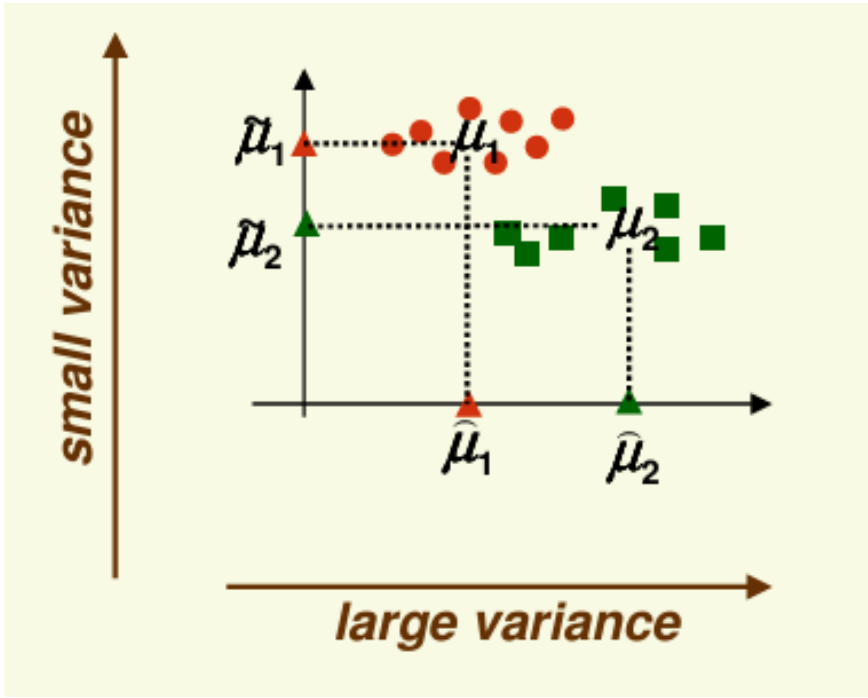
- $|\tilde{m}_1 - \tilde{m}_2|$ seems like a good measure to calculate the separation between projections of different classes.
- How good is $|\tilde{m}_1 - \tilde{m}_2|$ as a measure of separation?
 - The larger $|\tilde{m}_1 - \tilde{m}_2|$ the better is the expected separation



- The vertical axes is a better line than the horizontal axes to project to for class separability
- However
 - $|\tilde{m}_1 - \tilde{m}_2| > |m_1 - m_2|$

FISHER LINEAR DISCRIMINANT

- The problem with $|\tilde{m}_1 - \tilde{m}_2|$ is that it does not consider the **variance** of the classes!!!
 - This implies that we need to improve our objective function.
 - We can **normalize** $|\tilde{m}_1 - \tilde{m}_2|$ by a factor which is **proportional to variance**.



FISHER LINEAR DISCRIMINANT

- We want to obtain that the difference between the **means** of the two (transformed) classes is **large** compared to **the standard deviation** of each class.
- Thus, we define the Fisher linear discriminant as the linear function $\mathbf{w}^t \mathbf{X}$ for which the function J is maximum:

$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

- where \tilde{s}_1 & \tilde{s}_2 are the **scatters** of the classified samples ω_1 and ω_2 , respectively, defined as:

$$\tilde{s}_i^2 = \sum_{j=1}^{N_i} \left(y_j^{(i)} - \tilde{m}_i \right)^2$$

FISHER LINEAR DISCRIMINANT

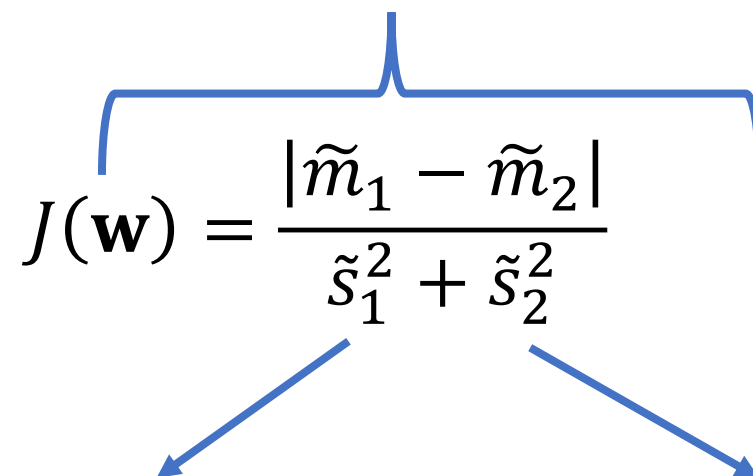
- We want that the dispersions are small enough, that is, the samples of a class are quite **concentrated around the mean value**.
- To get J as an explicit function of \mathbf{w} , *scatter matrices* S_i and S_w are defined as follows:

$$S_i = \sum_{j=1}^{N_i} \left(\mathbf{x}_j^{(i)} - \mathbf{m}_i \right) \left(\mathbf{x}_j^{(i)} - \mathbf{m}_i \right)^t \quad S_w = S_1 + S_2 \quad \begin{array}{l} \text{within class} \\ \text{scatter} \\ \text{matrix} \end{array}$$

- Notice that the scatter is just sample variance multiplied by n .
 - Scatter measures the same thing as variance, the spread of data around the mean.

FISHER LINEAR DISCRIMINANT

**Aim: projected means
are far from each other**

$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2}$$


**Aim: want scatter in class 1
is as small as possible, i.e.,
samples of class 1 cluster
around the projected mean
 \tilde{m}_1 .**

**Aim: want scatter in class 2
is as small as possible, i.e.,
samples of class 2 cluster
around the projected mean
 \tilde{m}_2 .**

FISHER LINEAR DISCRIMINANT

- All we need to do now is to write \mathbf{J} as a function of \mathbf{w} and maximize it.

$$\begin{aligned} \tilde{s}_i^2 &= \sum_{j=1}^{N_i} \left(y_j^{(i)} - \tilde{m}_i \right)^2 \\ S_i &= \sum_{j=1}^{N_i} \left(\mathbf{x}_j^{(i)} - \mathbf{m}_i \right) \left(\mathbf{x}_j^{(i)} - \mathbf{m}_i \right)^t \\ S_w &= S_1 + S_2 \\ J(\mathbf{w}) &= \frac{|\tilde{m}_1 - \tilde{m}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2} \end{aligned} \quad \left. \begin{aligned} &\tilde{s}_i^2 = \sum_{j=1}^{N_i} (y_j - \tilde{m}_i)^2 = \sum_{j=1}^{N_i} \left(\mathbf{w}^t \mathbf{x}_j^{(i)} - \mathbf{w}^t \mathbf{m}_i \right)^2 \\ &= \sum_{j=1}^{N_i} \left(\mathbf{w}^t \mathbf{x}_j^{(i)} - \mathbf{w}^t \mathbf{m}_i \right)^t \left(\mathbf{w}^t \mathbf{x}_j^{(i)} - \mathbf{w}^t \mathbf{m}_i \right) \\ &= \sum_{j=1}^{N_i} \left(\mathbf{w}^t (\mathbf{x}_j^{(i)} - \mathbf{m}_i) \right)^t \left(\mathbf{w}^t (\mathbf{x}_j^{(i)} - \mathbf{m}_i) \right) \\ &= \sum_{j=1}^{N_i} \left((\mathbf{x}_j^{(i)} - \mathbf{m}_i)^t \mathbf{w} \right)^t \left((\mathbf{x}_j^{(i)} - \mathbf{m}_i)^t \mathbf{w} \right) \\ &= \sum_{j=1}^{N_i} (\mathbf{x}_j^{(i)} - \mathbf{m}_i) (\mathbf{x}_j^{(i)} - \mathbf{m}_i)^t \mathbf{w} = \mathbf{w}^t S_i \mathbf{w} \end{aligned} \right\}$$

FISHER LINEAR DISCRIMINANT

- All we need to do now is to write \mathbf{J} as a function of \mathbf{w} and maximize it.

$$\tilde{s}_i^2 = \sum_{j=1}^{N_i} (y_j - \tilde{m}_i)^2 = \sum_{j=1}^{N_i} (\mathbf{w}^t \mathbf{x}_j^{(i)} - \mathbf{w}^t \mathbf{m}_i)^2 = \mathbf{w}^t S_i \mathbf{w}$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = \mathbf{w}^t S_1 \mathbf{w} + \mathbf{w}^t S_2 \mathbf{w} = \mathbf{w}^t S_w \mathbf{w}$$

- Let's rewrite the separations of the projected means:

$$\left. \begin{aligned} \tilde{m}_1 &= \mathbf{w}^t \cdot \mathbf{m}_1 \\ \tilde{m}_2 &= \mathbf{w}^t \cdot \mathbf{m}_2 \end{aligned} \right\} \begin{aligned} (\tilde{m}_1 - \tilde{m}_2)^2 &= (\mathbf{w}^t m_1 - \mathbf{w}^t m_2)^2 \\ &= (\mathbf{w}^t m_1 - \mathbf{w}^t m_2)(\mathbf{w}^t m_1 - \mathbf{w}^t m_2)^t \\ &= \mathbf{w}^t (m_1 - m_2)(m_1 - m_2)^t \mathbf{w} \\ &= \mathbf{w}^t S_B \mathbf{w} \end{aligned}$$

FISHER LINEAR DISCRIMINANT

$$\begin{aligned}(\tilde{m}_1 - \tilde{m}_2)^2 &= (\mathbf{w}^t m_1 - \mathbf{w}^t m_2)^2 \\&= (\mathbf{w}^t m_1 - \mathbf{w}^t m_2)(\mathbf{w}^t m_1 - \mathbf{w}^t m_2)^t \\&= \mathbf{w}^t (m_1 - m_2)(m_1 - m_2)^t \mathbf{w} \\&= \mathbf{w}^t S_B \mathbf{w}\end{aligned}$$

- \mathbf{S}_B measures separation between the means of two classes **before projection!**
- So we get $\mathbf{J}(\mathbf{w})$. Since we are able to express \mathbf{J} as a direct function of \mathbf{w} , we can take its derivative with respect to \mathbf{w} and put equal to $\mathbf{0}$.

$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2} = \frac{\mathbf{w}^t S_B \mathbf{w}}{\mathbf{w}^t S_w \mathbf{w}}$$

FISHER LINEAR DISCRIMINANT

- By taking the derivative we get:

$$\frac{\partial J}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = S_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

which is the Fisher transform.

EXAMPLE

- Class 1 has 5 samples $\mathbf{x}_1 = [(1,2),(2,3),(3,3),(4,5),(5,5)]$
- Class 2 had 6 samples $\mathbf{x}_2 = [(1,0),(2,1),(3,1),(3,2),(5,3),(6,5)]$

- First compute the mean for each class

$$m_1 = [3, 3.6] \quad m_2 = [3.3, 2]$$

- Compute scatter matrices S_1 and S_2 for each class

$$S_1 = \begin{bmatrix} 10 & 8 \\ 8 & 7.2 \end{bmatrix} \quad S_2 = \begin{bmatrix} 17.3 & 16 \\ 16 & 16 \end{bmatrix}$$

Within the class scatter: $S_w = S_1 + S_2 = \begin{bmatrix} 27.3 & 24 \\ 24 & 23.2 \end{bmatrix}$

EXAMPLE

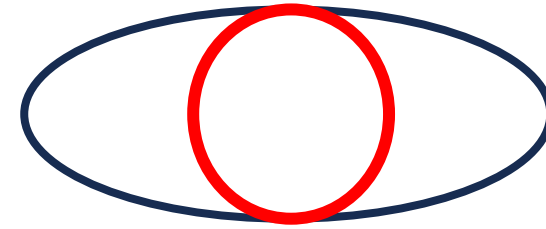
- Calculate the inverse of $S_w = [0.39 \ -0.41; -0.41 \ 0.47]$
- Finally the optimal line direction \mathbf{w}

$$\mathbf{w} = S_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

- $\mathbf{w} = [-0.79 \ 0.89]$
- The last step is to compute the actual ***1D*** vector:
 - $y_1 = \mathbf{w}^t \mathbf{x}_1^t = [-0.79 \ 0.89] [1 \ 2 \ 3 \ 4 \ 5; 2 \ 3 \ 3 \ 5 \ 5] = [0.81 \ \dots \ 0.4]$
 - $y_1 = \mathbf{w}^t \mathbf{x}_1^t = [-0.79 \ 0.89] [1 \ 2 \ 3 \ 3 \ 5 \ 6; 0 \ 1 \ 1 \ 2 \ 3 \ 5] = [-0.65 \ \dots \ -0.25]$

FDA DRAWBACKS

- Reduces dimensionality only to **$d-1$**
- For complex data, projection to even the best line may result in unseparable projected samples
- $J(\mathbf{w})$ is always 0 if $m_1 = m_2$
- When $J(\mathbf{w})$ is always large: classes have large overlap when projected to any line. But such a case is also problematic for PCA, making it fail.



PCA performs
reasonably well,
but FDA fails



SUMMARY

- Dimensionality reduction with **linear methods** like PCA and FDA allow us to
 - Visualize the training set and then work on it.
 - Reduce the time and space complexity
 - Remove multi-collinearity (can be helpful to handle overfitting)
 - Remove irrelevant features from the data
 - Handle curse-of dimensionality
- Once dimensionality reduction is applied, we typically apply classification or other tasks using the new feature space. Especially FDA is very suitable for classification tasks since it focuses on discrimination of classes.



NON-LINEAR DIMENSIONALITY REDUCTION

- **t-Distributed Stochastic Neighbor Embedding (t-SNE):** is effective for visualizing high-dimensional data in lower dimensions, often used for **clustering**.
- **Uniform Manifold Approximation and Projection (UMAP):** preserves both global and local structures of the data.
- **Autoencoders:** Neural network-based models that learn a compressed representation of the data.
- **Factor Analysis:** A probabilistic model that assumes observed variables are linear combinations of underlying factors plus noise.
- **Isomap (Isometric Mapping):** Non-linear method that focuses on preserving the geodesic distances between all pairs of data points.



NON-LINEAR DIMENSIONALITY REDUCTION

- **Locally Linear Embedding (LLE):** Non-linear method that seeks a low-dimensional representation while preserving the local relationships within neighborhoods.
- **Kernel PCA:** Extends PCA using the kernel trick, allowing it to capture non-linear relationships.
- **Random Projection:** A simple, computationally efficient method that projects data onto a lower-dimensional subspace with a random projection matrix.



T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING (T-SNE)

BY LAURENS VAN DER MAATEN AND GEOFFREY HINTON IN 2008

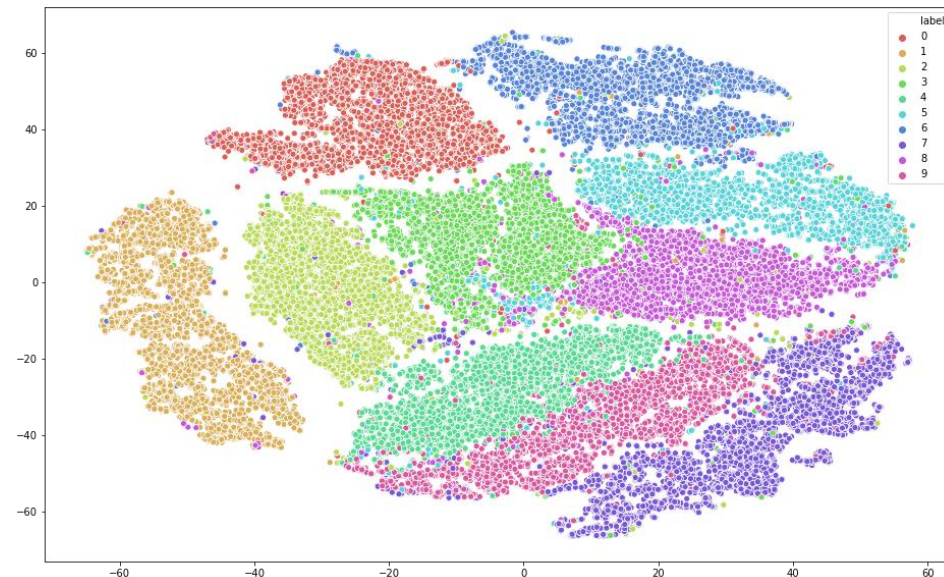
- t-SNE is a nonlinear technique that is well suited for embedding high dimension data into lower dimensional data (2D or 3D) for **data visualization**.
- Non-linear dimensionality reduction means that the algorithm allows us to **separate data that cannot be separated by a straight line**.
- **Stochastic** → not definite but random probability (standard deviation is unknown)
Neighbor Embedding → find the similarity among the data samples which are in a neighborhood



T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING (T-SNE)

BY LAURENS VAN DER MAATEN AND GEOFFREY HINTON IN 2008

- t-SNE is an algorithm that generates slightly different results each time on the same data set, focusing on retaining the structure of neighbor points.
 - It is an indeterministic approach.



T-SNE ALGORITHM

1) Given a dataset **X** with **n data points** in a high-dimensional space (D-dimensional), compute pairwise similarities between data points using a conditional Gaussian probability distribution **P_{ij}** :

$$P_{ij} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

where $\|x_i - x_j\|$ is the Euclidean distance between points x_i and x_j , and σ_i is the **variance** that is given as an input. We do not set the variance directly. Instead, we specify the expected number of neighbors, called **perplexity**.

T-SNE ALGORITHM

A perplexity is more or less a target number of neighbors for our central point. Basically, the higher the perplexity is the higher value variance has.

2) To make the optimization easier, the probabilities are symmetrized and obtain \hat{P}_{ij} that represents our high dimensional data:

$$\hat{P}_{ij} = \frac{P_{ij} + P_{ji}}{2n}$$

3) Define a lower-dimensional representation \mathbf{Y} for the data points (d-dimensional, where d is usually 2 or 3).

T-SNE ALGORITHM

4) Compute pairwise similarities Q_{ij} in the lower-dimensional space using a **t-distribution**:

$$Q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}$$

where \mathbf{y}_i and \mathbf{y}_j are the coordinates of the corresponding points in the lower dimensional space.

5) Symmetrize the pairwise similarities to obtain \hat{Q}_{ij}

$$\hat{Q}_{ij} = \frac{Q_{ij} + Q_{ji}}{2n}$$

T-SNE ALGORITHM

6) Our goal is to find new dimensions through optimization such that all \hat{P}_{ij} and \hat{Q}_{ij} are as close as possible. This closeness is found by **Kullback-Leibler divergence**:

$$C = \sum_i \text{KL}(\hat{P}_i || \hat{Q}_i) = \sum_i \sum_j \hat{P}_{ij} \log \frac{\hat{P}_{ij}}{\hat{Q}_{ij}}$$

A higher Kullback-Leibler (KL) divergence between two probability distributions indicates a greater dissimilarity or mismatch between them.

7) Use gradient descent to minimize the KL divergence with respect to the coordinates \mathbf{Y} in the lower-dimensional space.

PCA vs. t-SNE



blog.DailyDoseofDS.com

UNIVERSITÀ
di **VERONA**

PCA

Purpose	Dimensionality reduction
Type of algorithm	Deterministic
Unique solution?	Yes
Projection type	Linear
Type of approach	Global
Interpretation	PCA is just a rotation of axes

t-SNE

Data visualisation in low dimensions (typically 2D)
Stochastic
No
Non-linear
Local/Global*
Subjective interpretation

**We do not explicitly specify global structure preservation in tSNE.*



That's all

Cigdem Beyan
cigdem.beyan@univr.it
<https://cbeyan.github.io/>