Recall from last time …

PCA Algorithm
By Eigen Decomposition

Data Preprocessing

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

Preprocessing (feature scaling/mean normalization):

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

Center the data

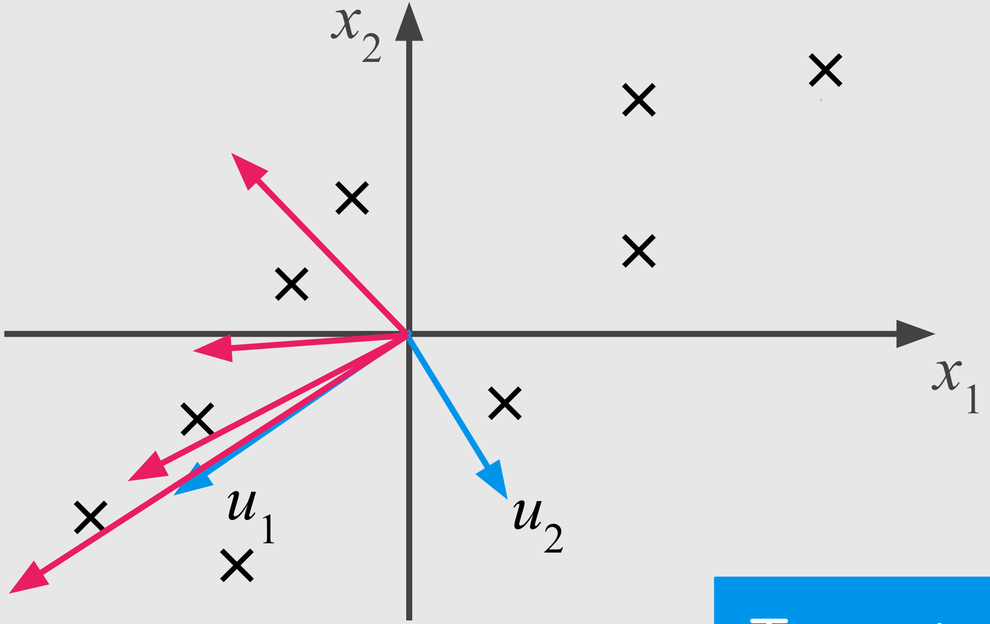
PCA Algorithm

Reduce data from n-dimensions to k-dimensions

Compute "covariance matrix":

$$\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^{\mathrm{T}} \quad \Rightarrow \quad n \times n \text{ matrix}$$

PCA Algorithm



Multiple a vector by $\boldsymbol{\Sigma}$:

$$\begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \times \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.2 \\ -0.2 \end{bmatrix}$$

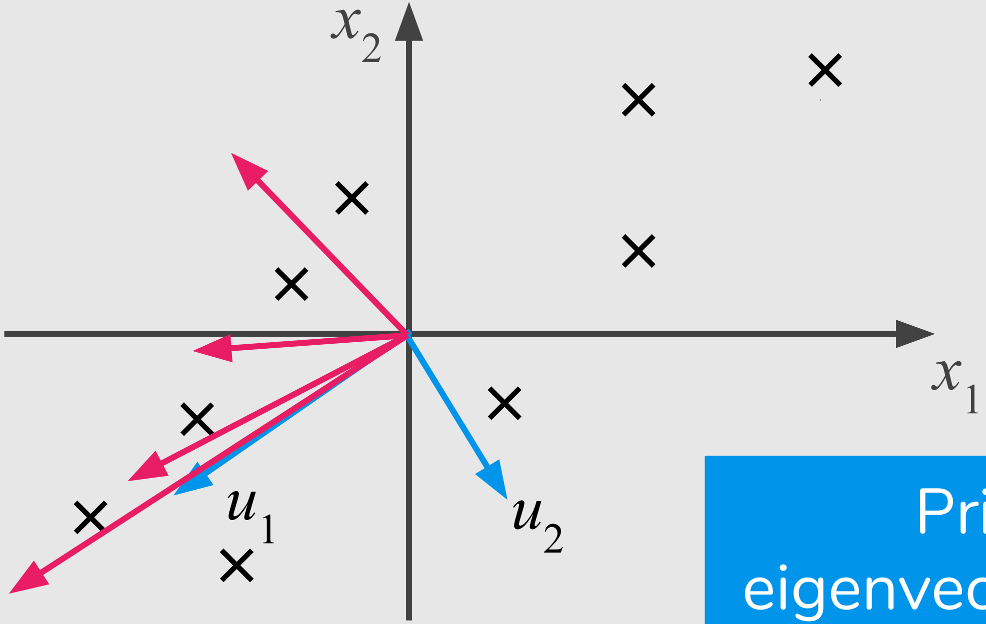
$$\begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \times \begin{bmatrix} -1.2 \\ -0.2 \end{bmatrix} = \begin{bmatrix} -2.5 \\ -1.0 \end{bmatrix}$$

$$\begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \times \begin{bmatrix} -2.5 \\ -1.0 \end{bmatrix} = \begin{bmatrix} -6.0 \\ -2.7 \end{bmatrix}$$

$$\begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \times \begin{bmatrix} -6.0 \\ -2.7 \end{bmatrix} = \begin{bmatrix} -14.1 \\ -6.4 \end{bmatrix}$$

Turns towards direction of variation

PCA Algorithm



Want vectors u which aren't turned: $\mathbf{\Sigma} u = \lambda u$

u = eigenvectors of $\mathbf{\Sigma}$

λ = eigenvalues

Principal components = eigenvectors w. largest eigenvalues

Finding Principal Components

1. Find eigenvalues by solving: $\det(\mathbf{\Sigma} - \lambda \mathrm{I}) = 0$

$$\det\begin{bmatrix} 2.0-\lambda & 0.8 \\ 0.8 & 0.6-\lambda \end{bmatrix} = (2.0-\lambda)(0.6-\lambda)-(0.8)(0.8) = \lambda^2 - 2.6\lambda + 0.56 = 0$$

$$\{\lambda_1, \lambda_2\} = \{2.36, 0.23\}$$

Finding Principal Components

2. Find i^{th} eigenvector by solving: $\boldsymbol{\Sigma} u_i = \lambda_i u_i$

$$\begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} u_{11} \\ u_{12} \end{bmatrix} = 2.36 \begin{bmatrix} u_{11} \\ u_{12} \end{bmatrix} \Rightarrow \begin{array}{l} 2.0u_{11} + 0.8u_{12} = 2.36u_{11} \\ 0.8u_{11} + 0.6u_{12} = 2.36u_{12} \end{array} \Rightarrow u_{11} = 2.2u_{12}$$

$$\Downarrow$$

$$\mathrm{u}_1 \sim \begin{bmatrix} 2.2 \\ 1 \end{bmatrix}$$

$\Downarrow$ Want $\|u_1\|=1$

$$\begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix}$$

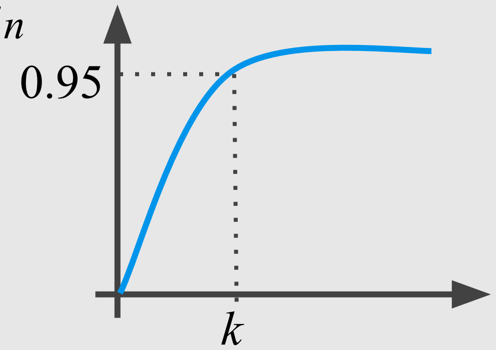
$$\begin{bmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} u_{21} \\ u_{22} \end{bmatrix} = 0.23 \begin{bmatrix} u_{21} \\ u_{22} \end{bmatrix} \Rightarrow u_2 = \begin{bmatrix} -0.41 \\ 0.91 \end{bmatrix}$$

3. 1^{st} PC: $\begin{bmatrix} 0.91 \\ 0.41 \end{bmatrix}$ and 2^{nd} PC: $\begin{bmatrix} -0.41 \\ 0.91 \end{bmatrix}$

How many PCs?

- Have eigenvectors $u_1, u_2, \ldots, u_n$, want $k < n$
- eigenvalue λ_i = variance along u_i
- Pick u_i that explain the most variance:
  - Sort eigenvectors s.t. $\lambda_1 > \lambda_2 > \lambda_3 > \ldots > \lambda_n$
  - Pick first k eigenvectors which explain 95% of total variance
    - Typical threshold: 90%, 95%, 99%

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \leq 1$$



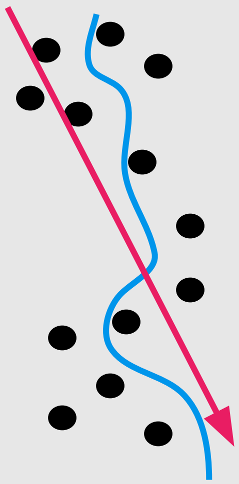
PCA in a Nutshell (Eigen Decomposition)

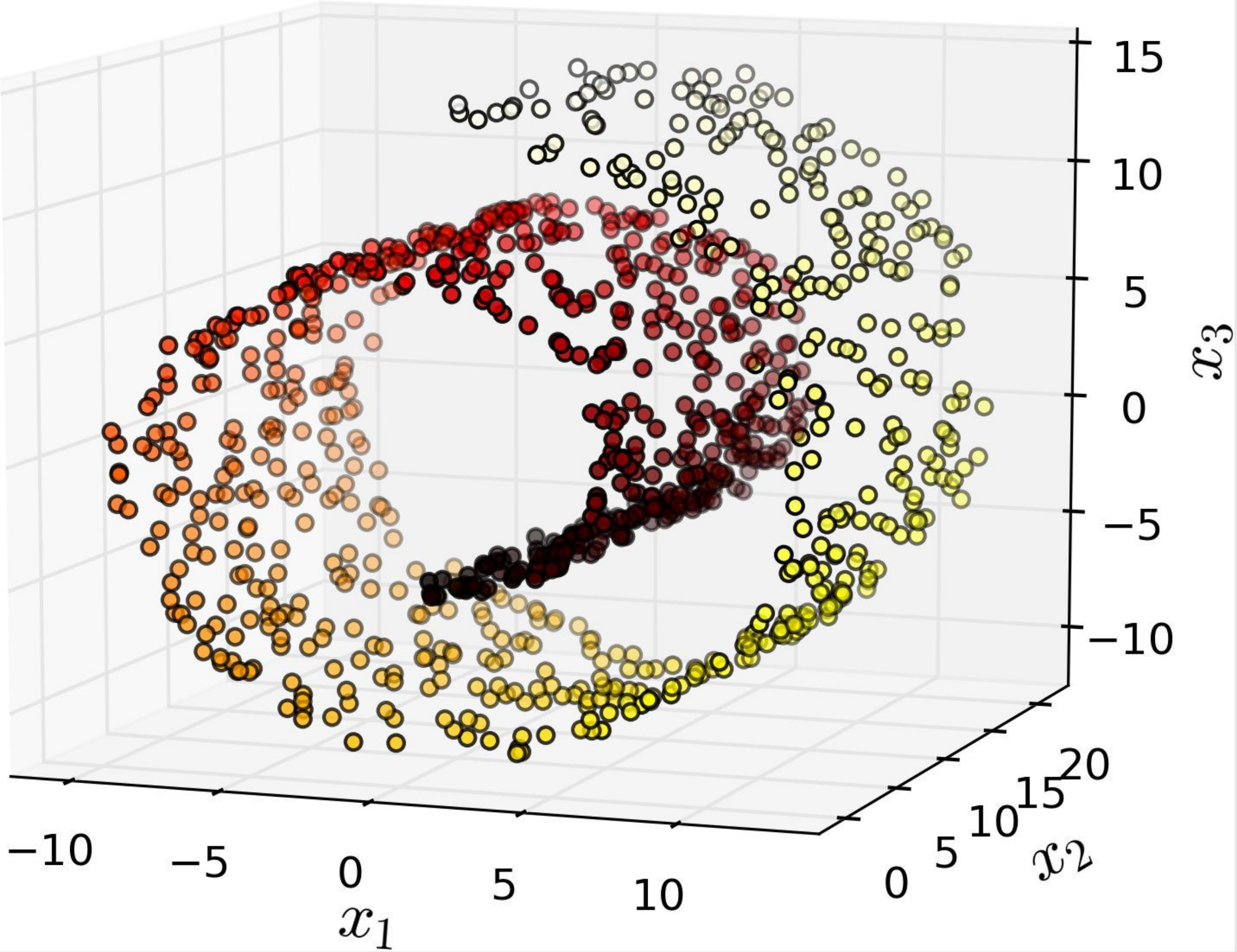
1. Center the data (and normalize)
2. Compute covariance matrix $\mathbf{\Sigma}$
3. Find eigenvectors u and eigenvalues λ
4. Sort eigenvectors and pick first k eigenvectors
5. Project data to k eigenvectors

Practical Issues

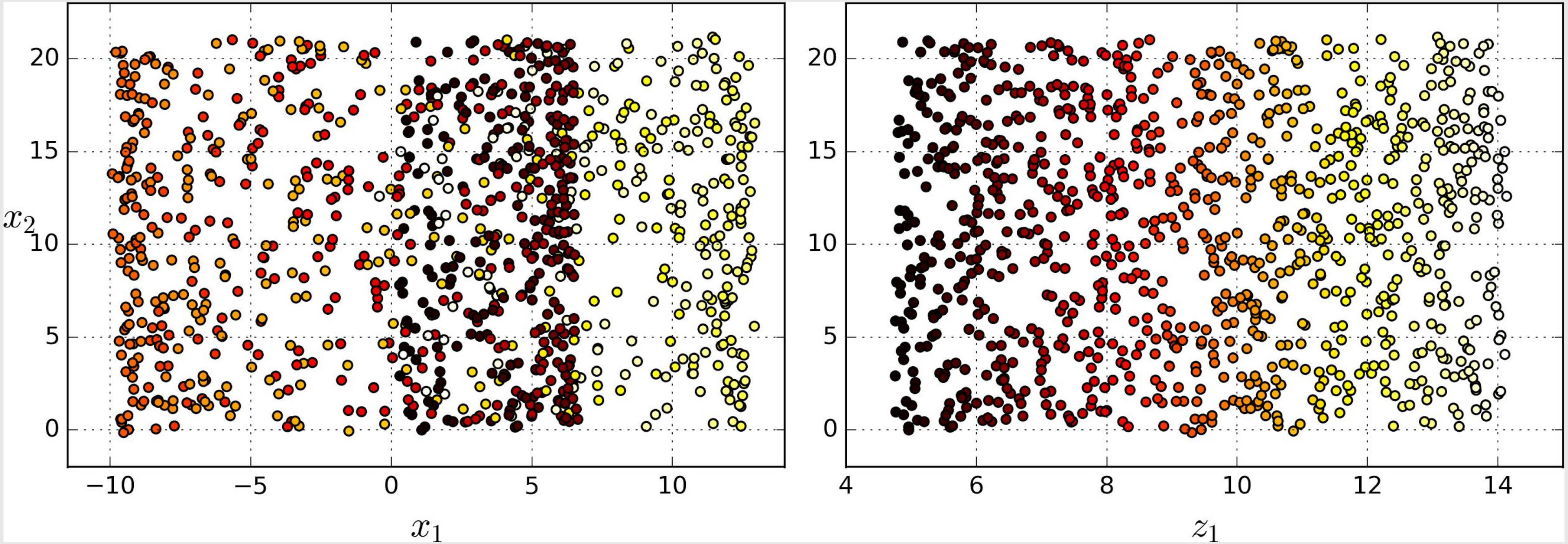
PCA: Practical Issues

- PCA assumes underlying subspace is linear
  - PCA cannot find a curve





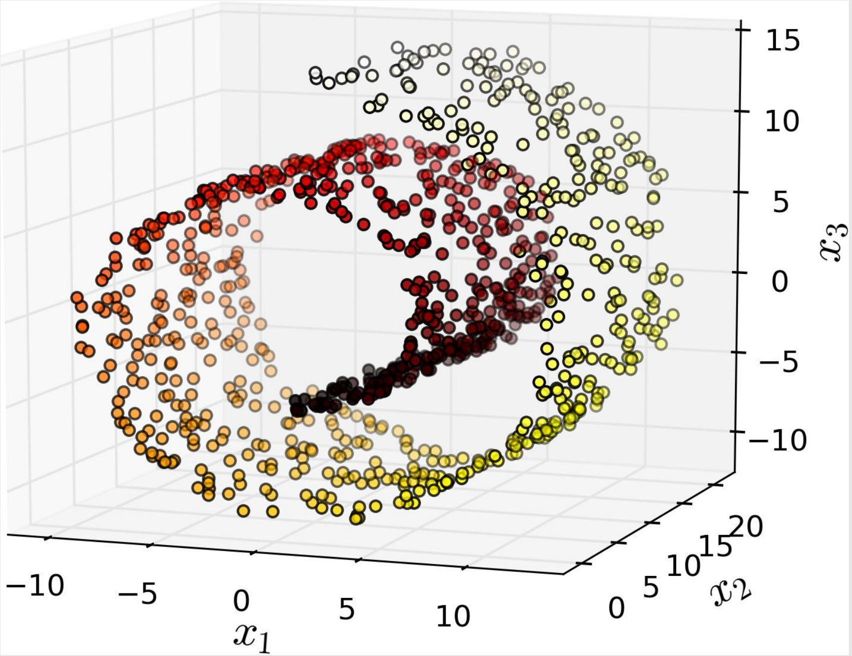
15
10
5
0
−5
−10
x_3
−10
−5
0
5
10
x_1
0
5
10
15
20
x_2



Squashing by projecting onto a plane (left) vs unrolling the swiss roll (right)

Manifold

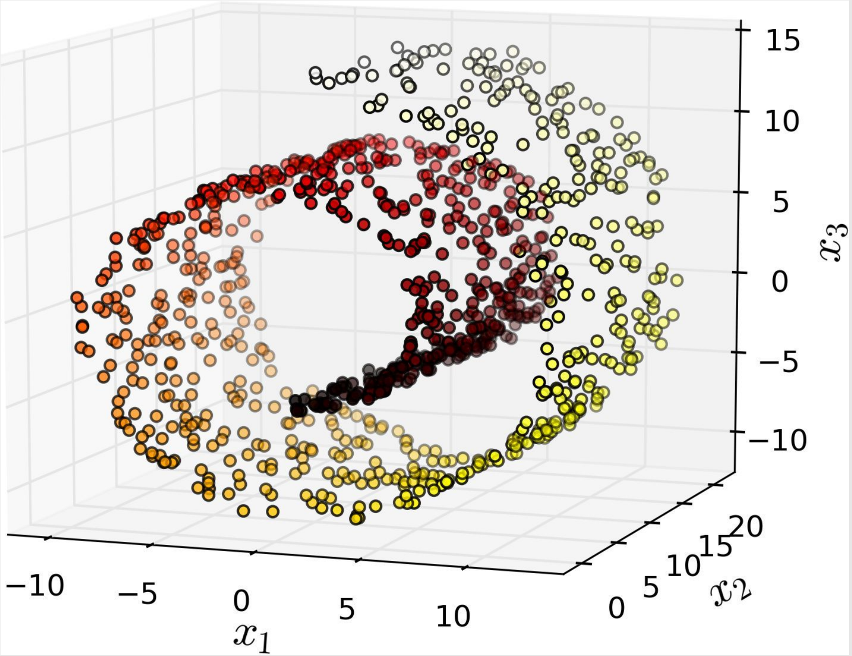
A d-dimensional manifold is a part of an n-dimensional space (where $d < n$) that locally resembles a d-dimensional hyperplane.



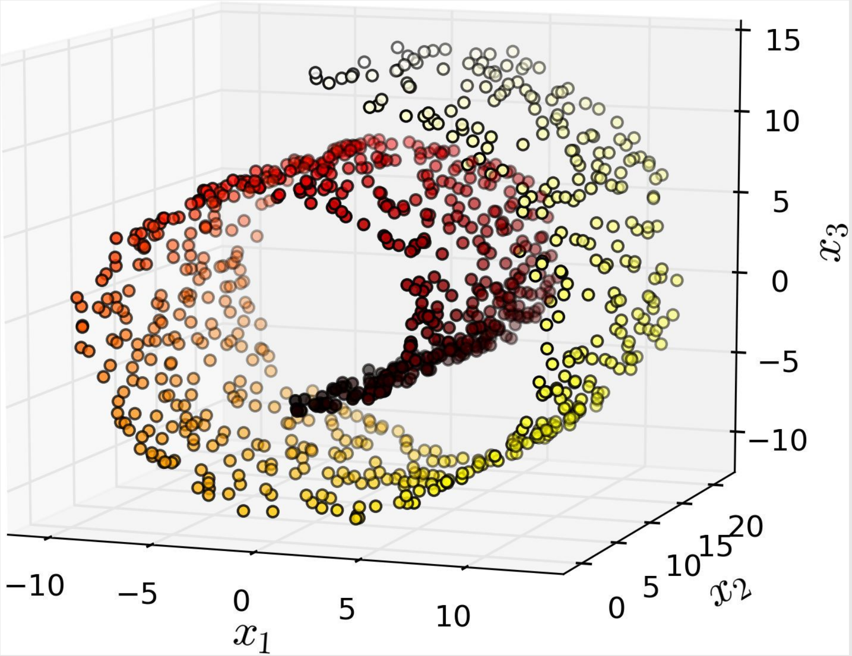
Manifold

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

In the case of the swiss roll, $d = 2$ and $n = 3$: **it locally resembles a 2D plane, but it is rolled in the 3rd dimension.**

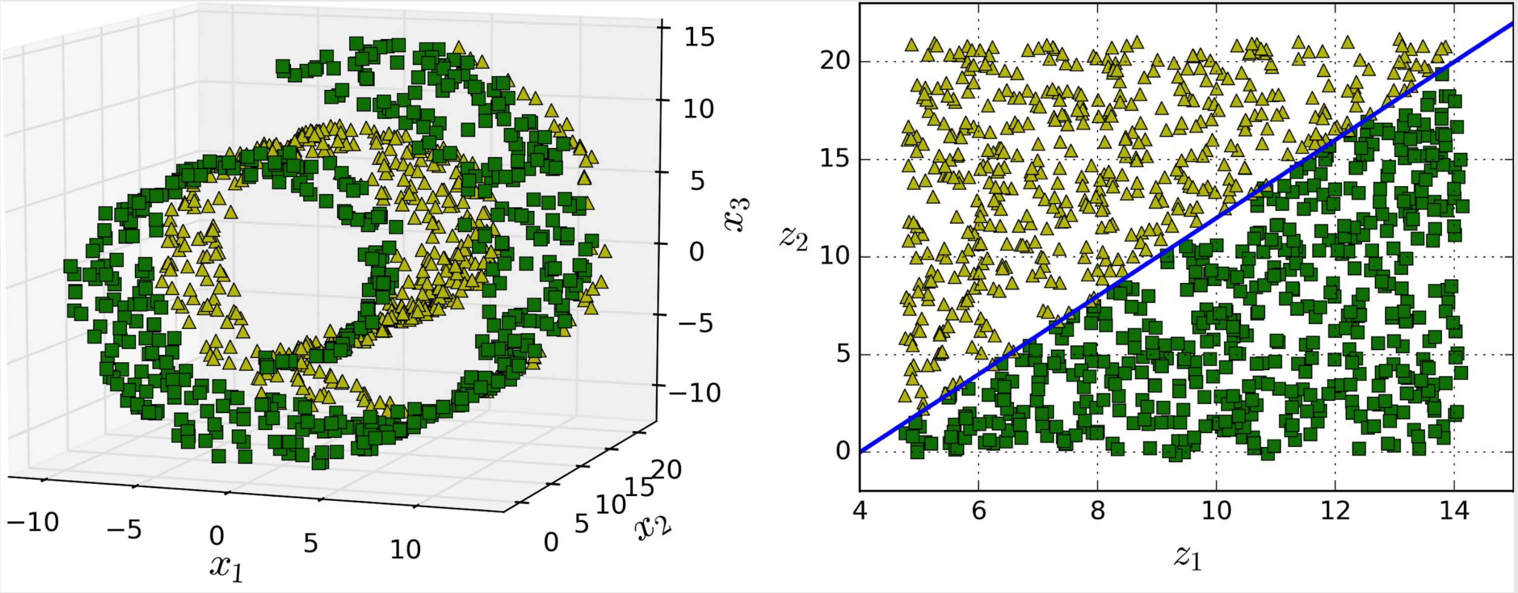


Manifold

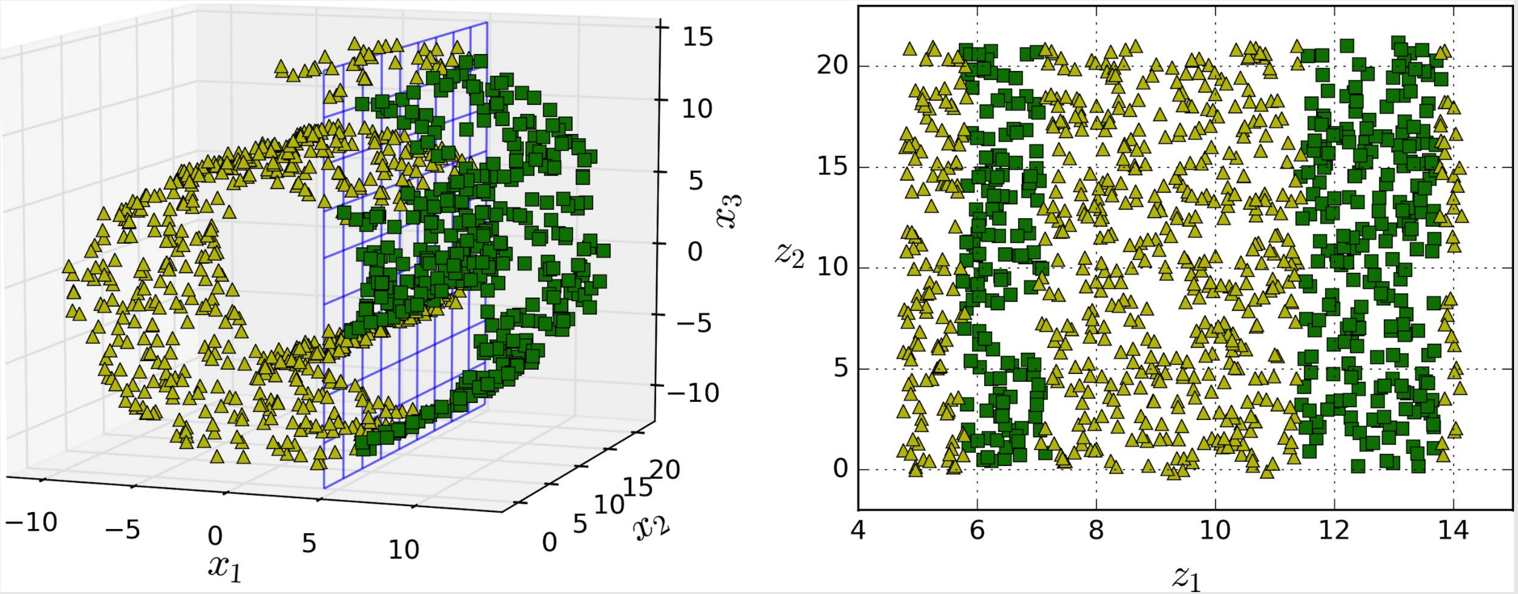


Manifold Assumption:

Most real world high-dimensional datasets lie close to a much lower-dimensional manifold.

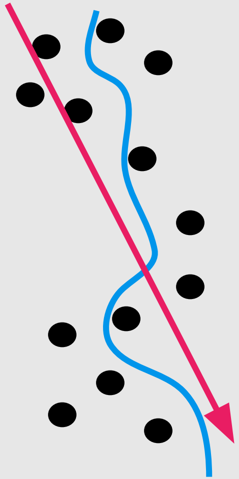


The decision boundary may not always be simpler with lower dimensions



The decision boundary may not always be simpler with lower dimensions.

PCA: Practical Issues



- PCA assumes underlying subspace is linear
  - PCA cannot find a curve
- PCA and Classification
  - PCA is unsupervised
  - PCA can pick direction that makes hard to separate classes

Linear Discriminant Analysis

Machine Learning and Pattern Recognition

Prof. Sandra Avila

Institute of Computing (IC/Unicamp)

MC886/MO444, October 10, 2017

Today's Agenda

- Linear Discriminant Analysis
  - PCA vs LDA
  - LDA: Simple Example
  - LDA Algorithm
  - LDA Step by Step

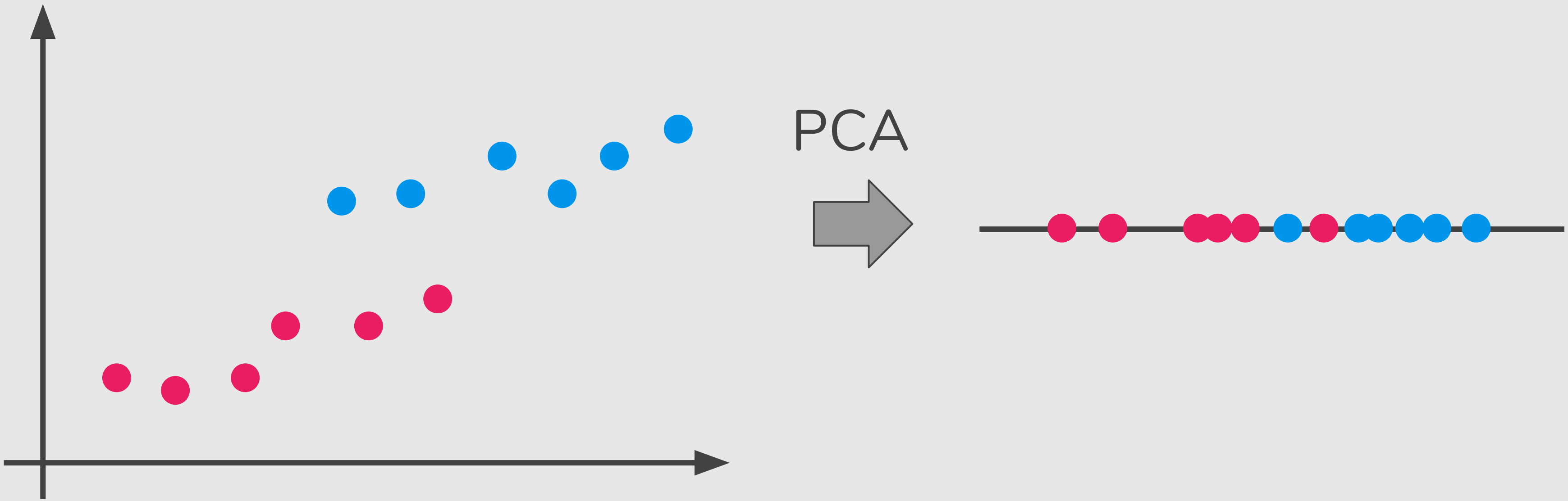
Linear Discriminant Analysis

Linear Discriminant Analysis (LDA)

- LDA pick a new dimension that gives:
  - Maximum separation between means of projected classes
  - Minimum variance within each projected class
- Solution: eigenvectors based on between-class and within-class covariance matrix

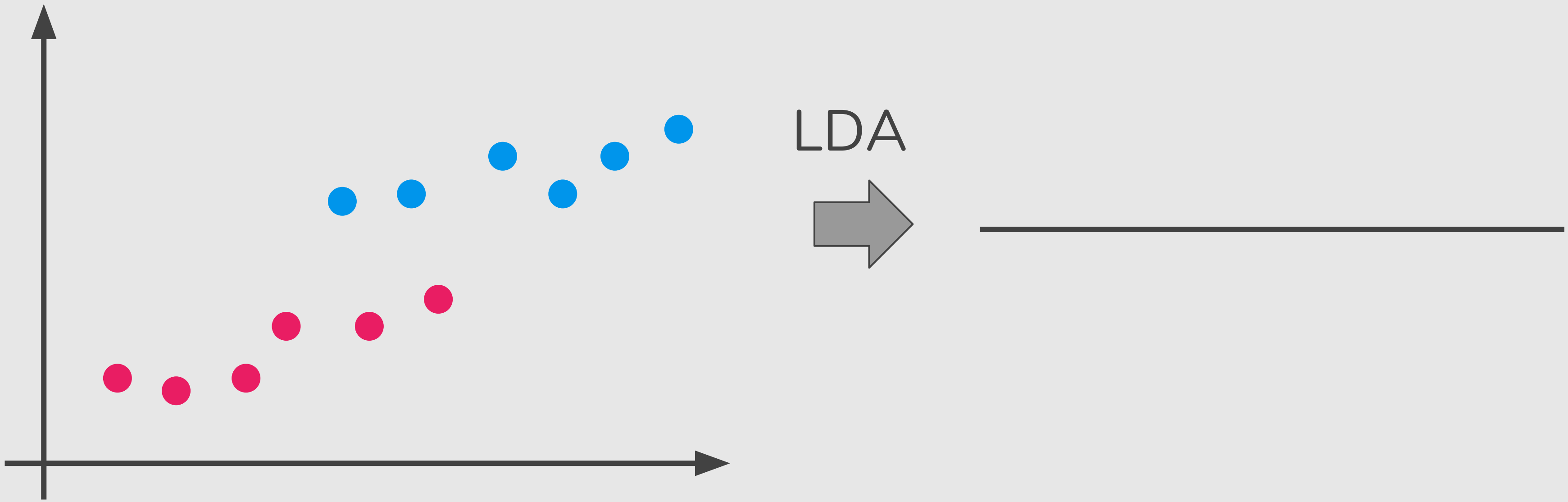
LDA: Simple Example

Reducing 2D to 1D



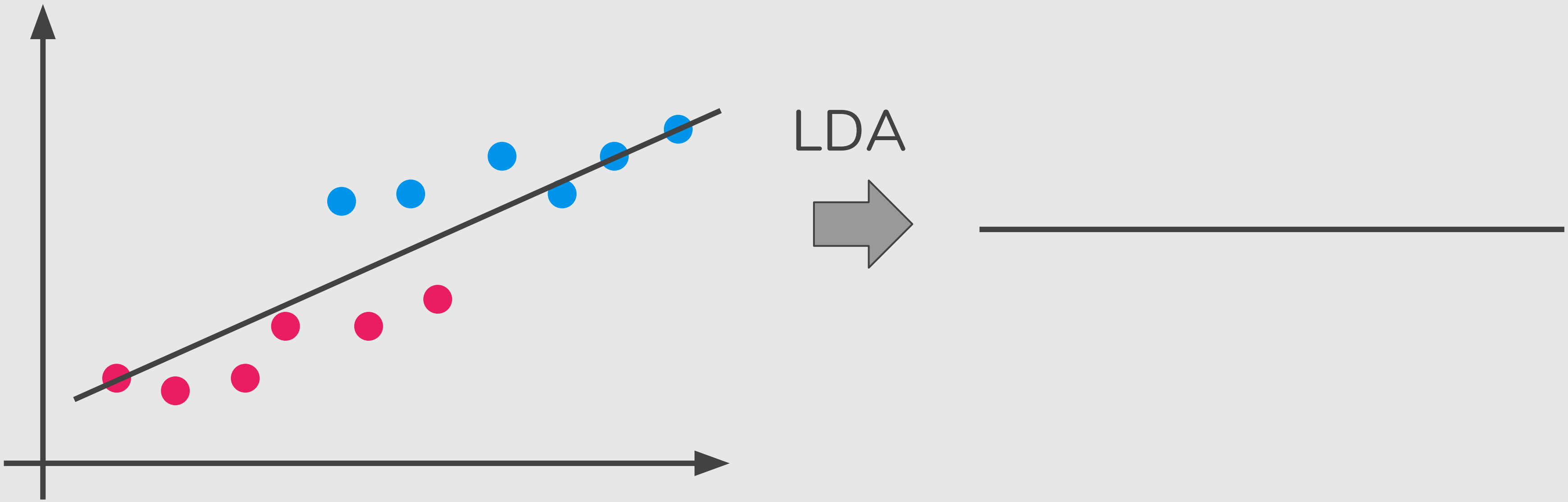
LDA: Simple Example

Reducing 2D to 1D



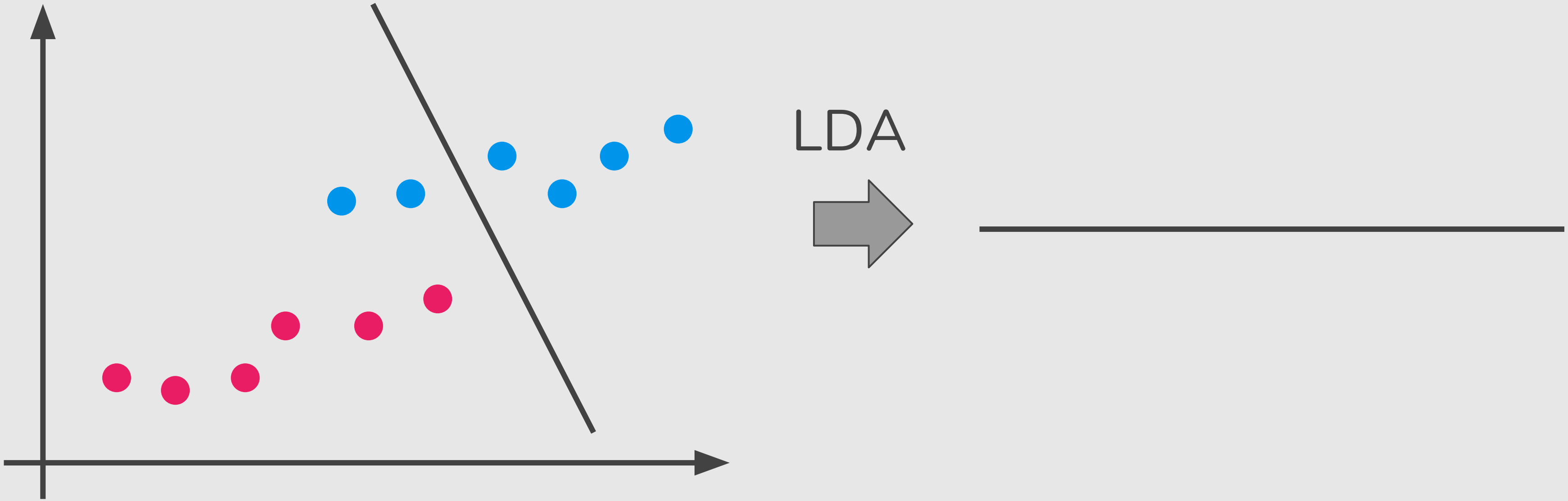
LDA: Simple Example

Reducing 2D to 1D



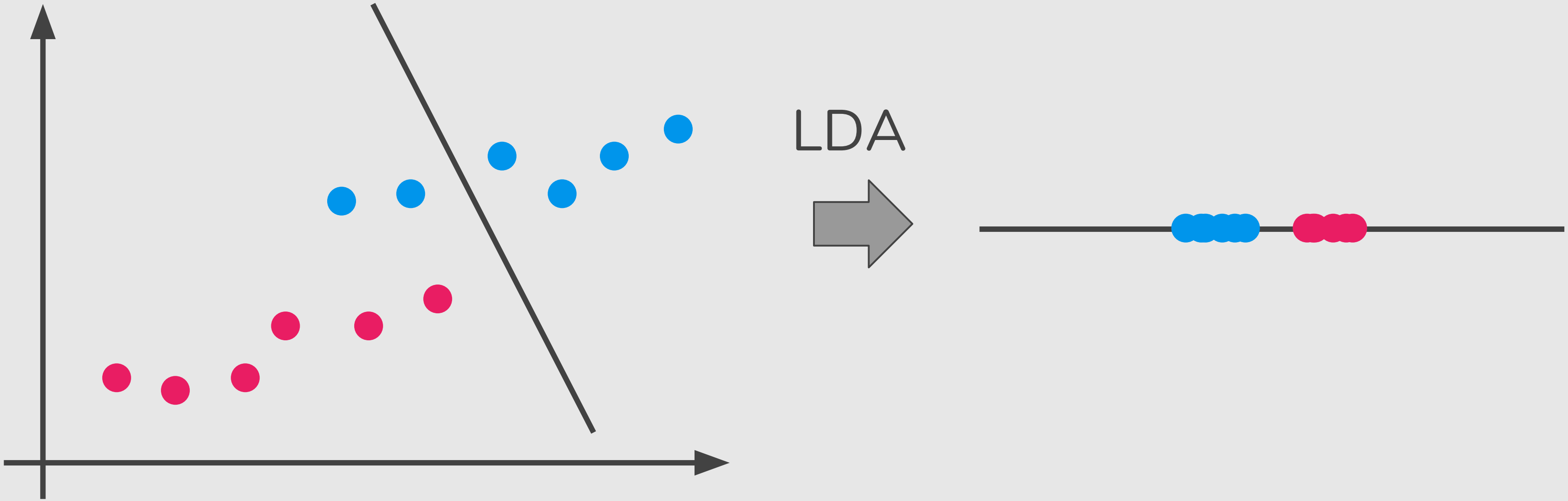
LDA: Simple Example

Reducing 2D to 1D



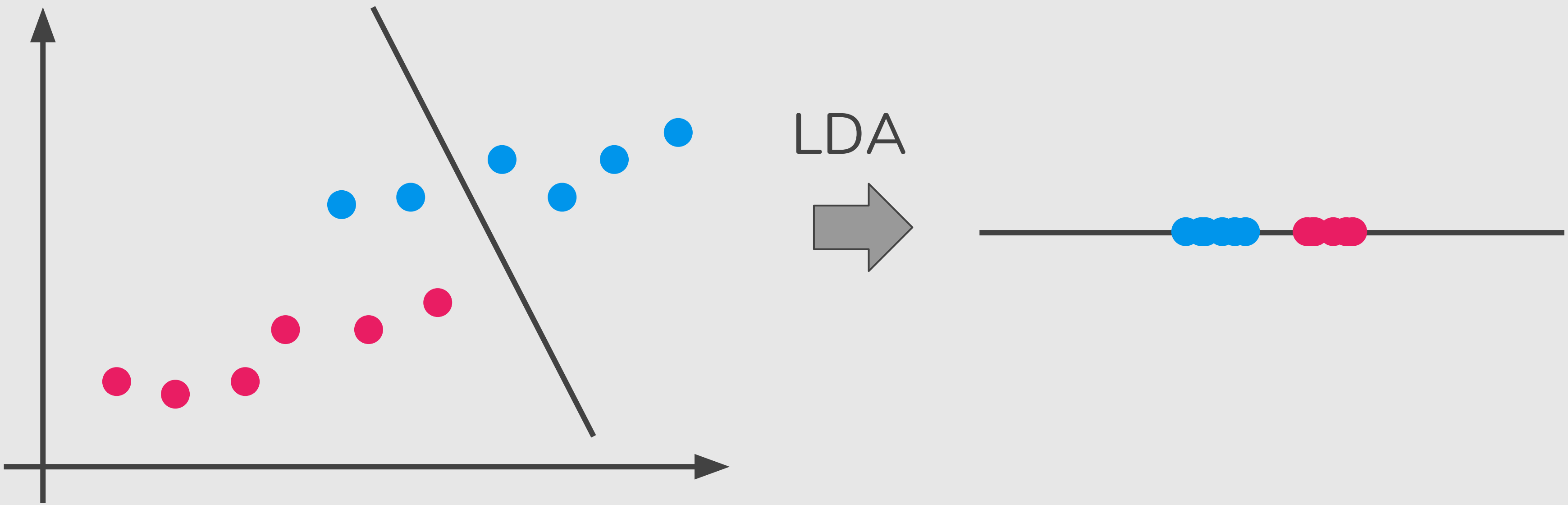
LDA: Simple Example

Reducing 2D to 1D



LDA: Simple Example

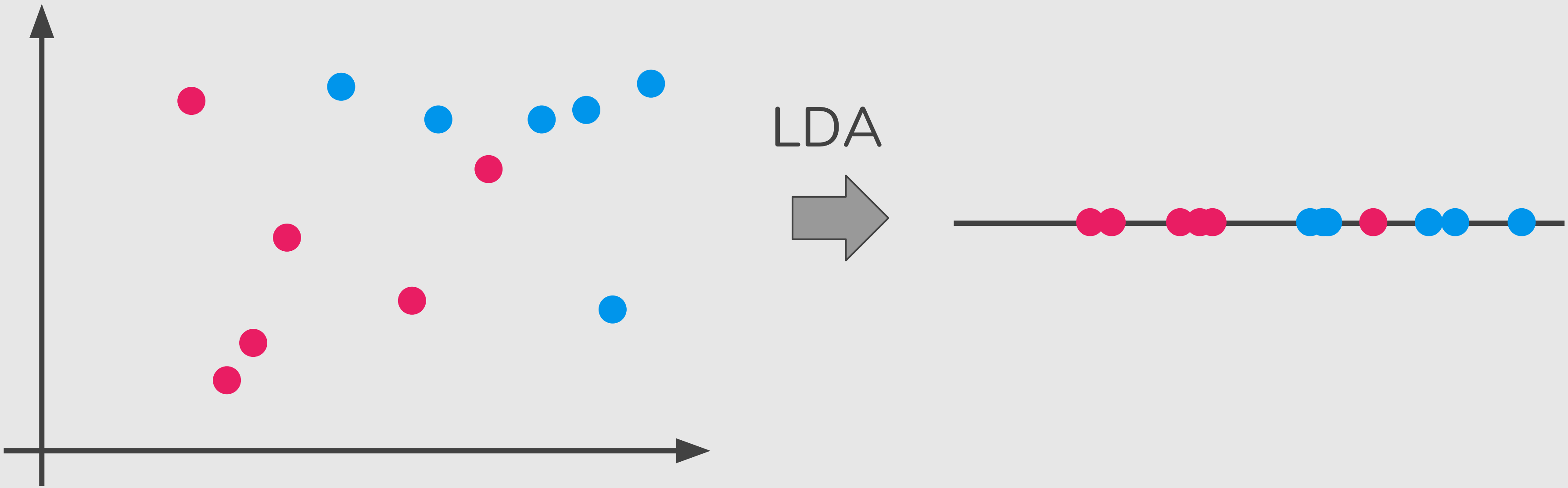
Reducing 2D to 1D



How LDA create a new axis?

How LDA create a new axis?

Reducing 2D to 1D



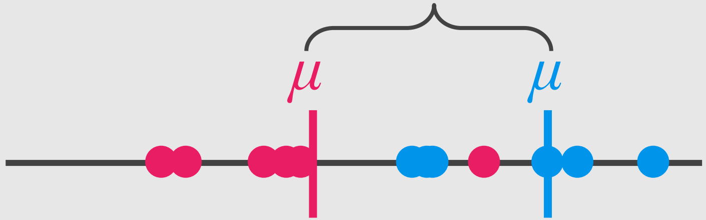
How LDA create a new axis?

The new axis is created according two criteria:

How LDA create a new axis?

The new axis is created according two criteria:

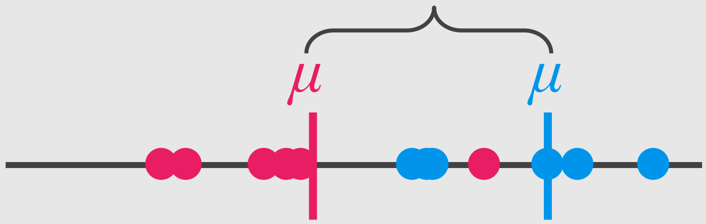
1. Maximize the distance between the means:



How LDA create a new axis?

The new axis is created according two criteria:

1. Maximize the distance between the means:

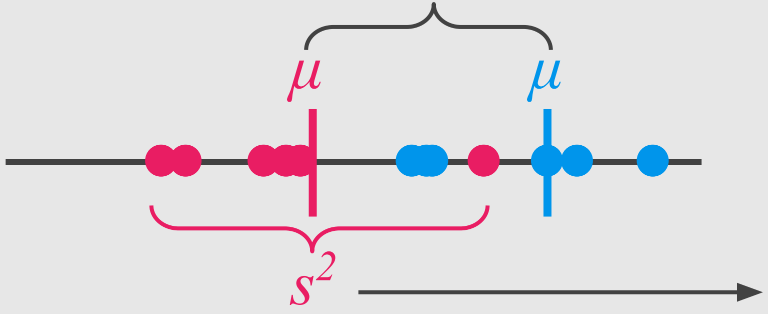


2. Minimize the variation (which LDA calls scatter) within each class.

How LDA create a new axis?

The new axis is created according two criteria:

1. Maximize the distance between the means:



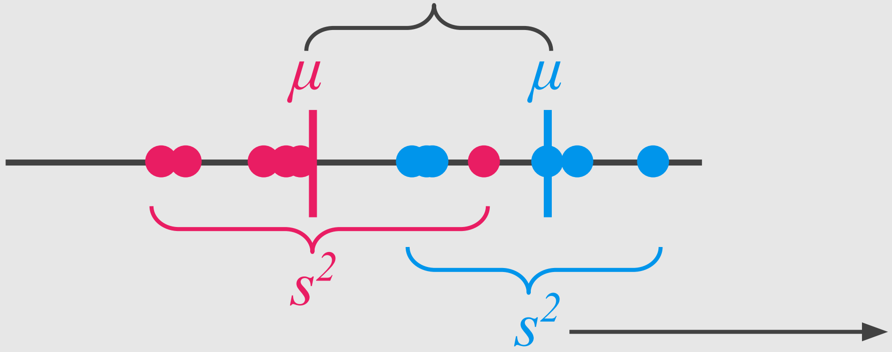
This is the scatter around the pink dots.

2. Minimize the variation (which LDA calls scatter) within each class.

How LDA create a new axis?

The new axis is created according two criteria:

1. Maximize the distance between the means:



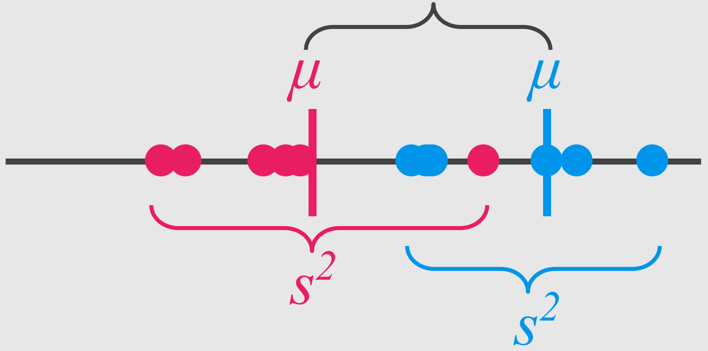
This is the scatter around the blue dots.

2. Minimize the variation (which LDA calls scatter) within each class.

How LDA create a new axis?

The new axis is created according two criteria:

1. Maximize the distance between the means:



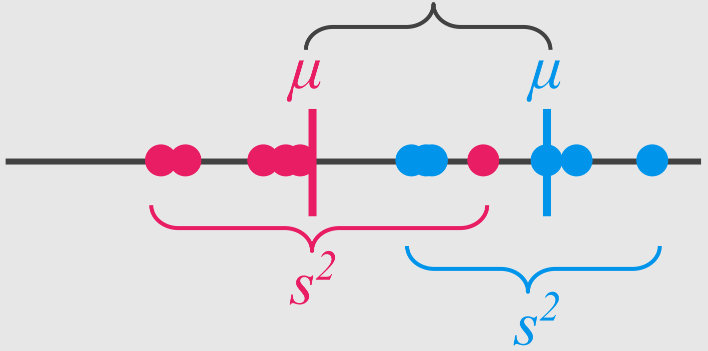
$$\frac{(\mu - \mu)^2}{s^2 + s^2}$$

2. Minimize the variation (which LDA calls scatter) within each class.

How LDA create a new axis?

The new axis is created according two criteria:

1. Maximize the distance between the means:



$$\frac{(\mu - \mu)^2}{s^2 + s^2}$$

$(\mu - \mu)^2 \longrightarrow$ Ideally large

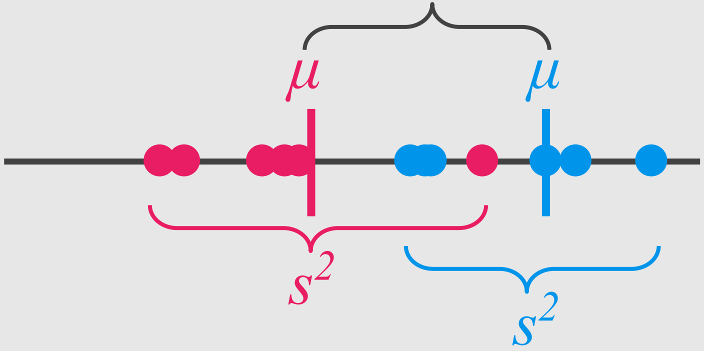
$s^2 + s^2 \longrightarrow$ Ideally small

2. Minimize the variation (which LDA calls scatter) within each class.

How LDA create a new axis?

The new axis is created according two criteria:

1. Maximize the distance between the means:



2. Minimize the variation (which LDA calls scatter) within each class.

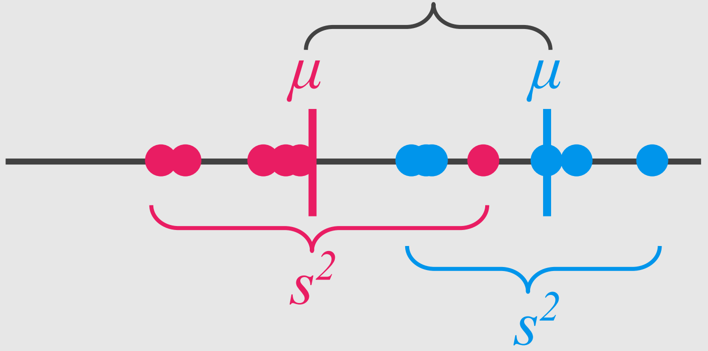
Let’s call $(\mu - \mu)$ d for distance.

$$\frac{(\mu - \mu)^2}{s^2 + s^2} \quad \begin{matrix} \longrightarrow \text{Ideally large} \\ \longrightarrow \text{Ideally small} \end{matrix}$$

How LDA create a new axis?

The new axis is created according two criteria:

1. Maximize the distance between the means:



Let's call $(\mu - \mu)$ d for distance.

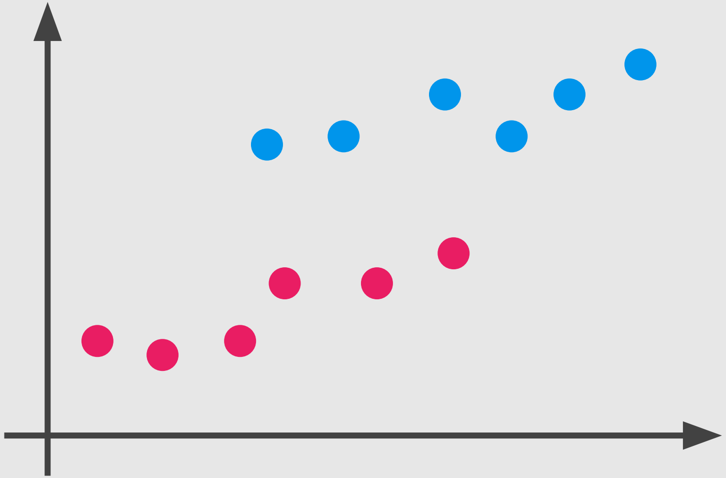
$$\frac{d^2}{s^2 + s^2}$$

d^2 → Ideally large

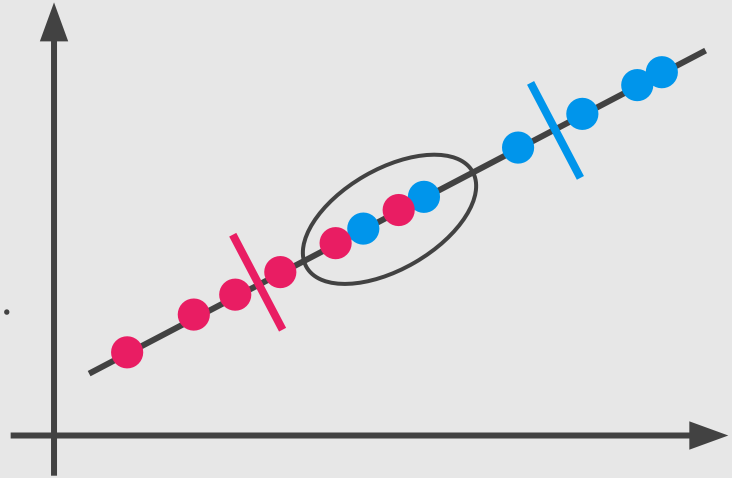
$s^2 + s^2$ → Ideally small

2. Minimize the variation (which LDA calls scatter) within each class.

Why both distance and scatter are important?

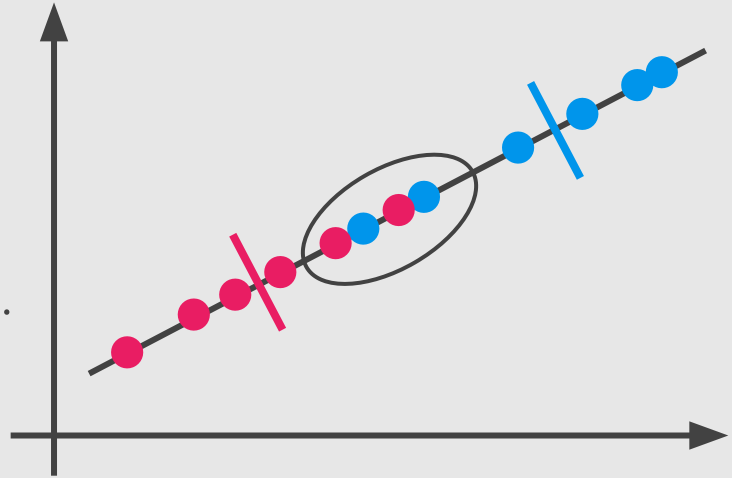


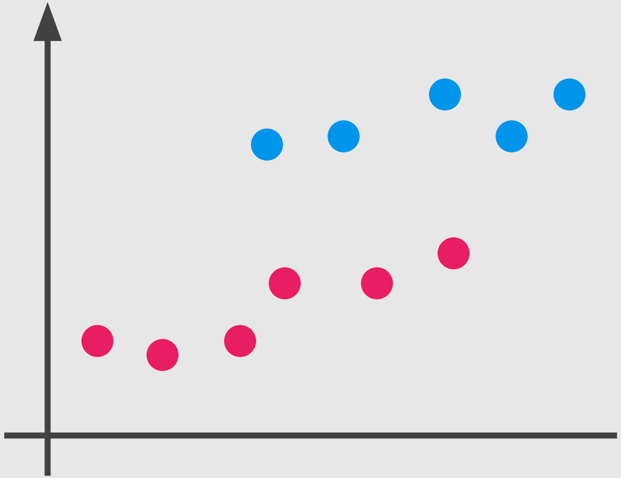
If we only maximize the distance between means ...



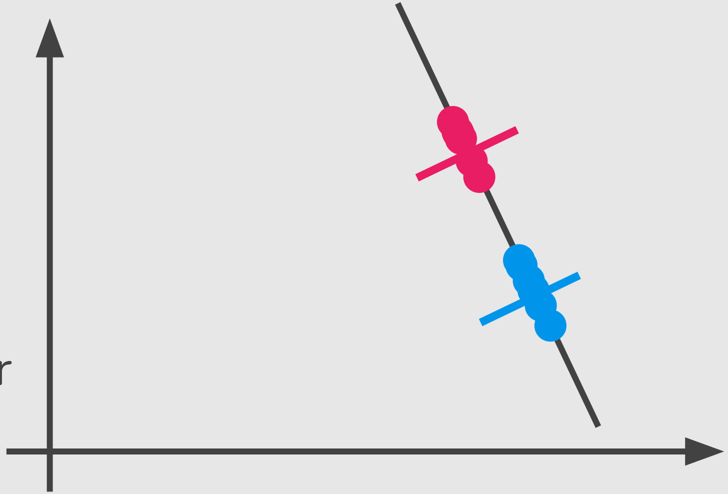
Why both distance and scatter are important?

If we only maximize the distance between means ...

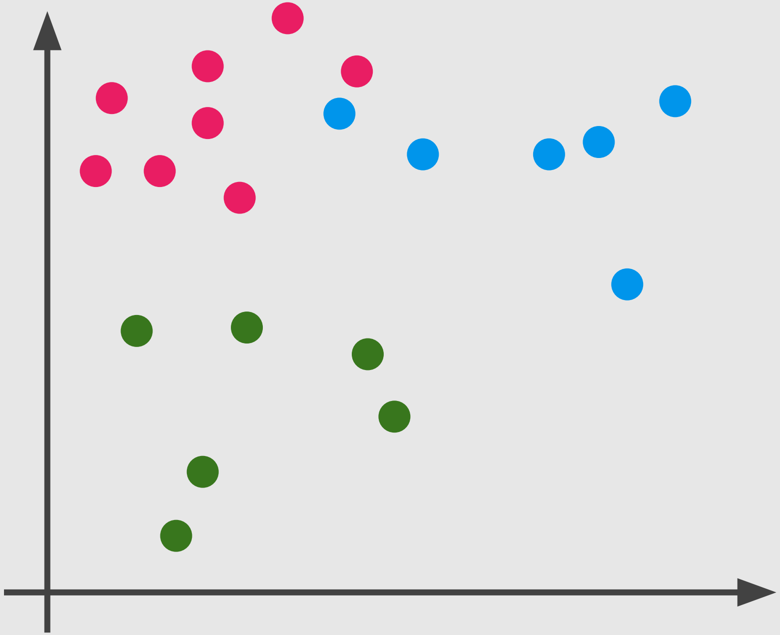




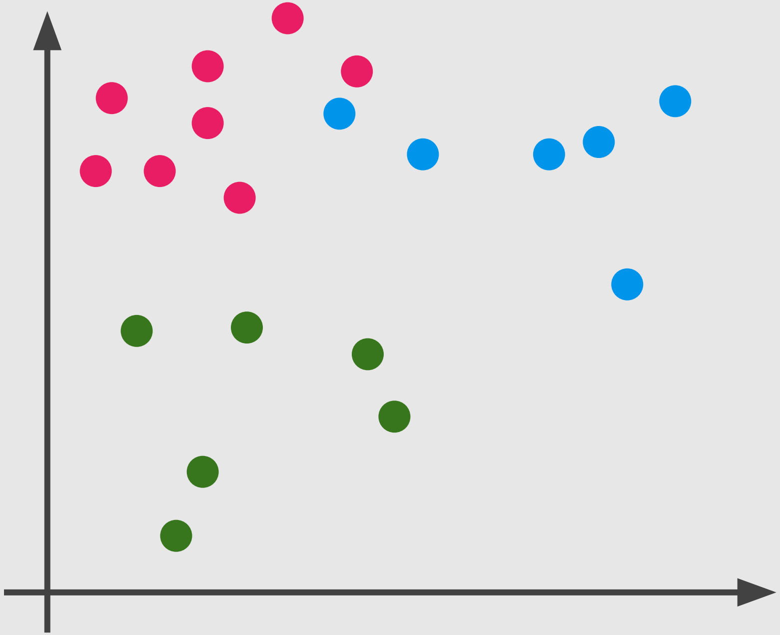
If we optimize the distance between means and scatter



What if we have 3 classes?



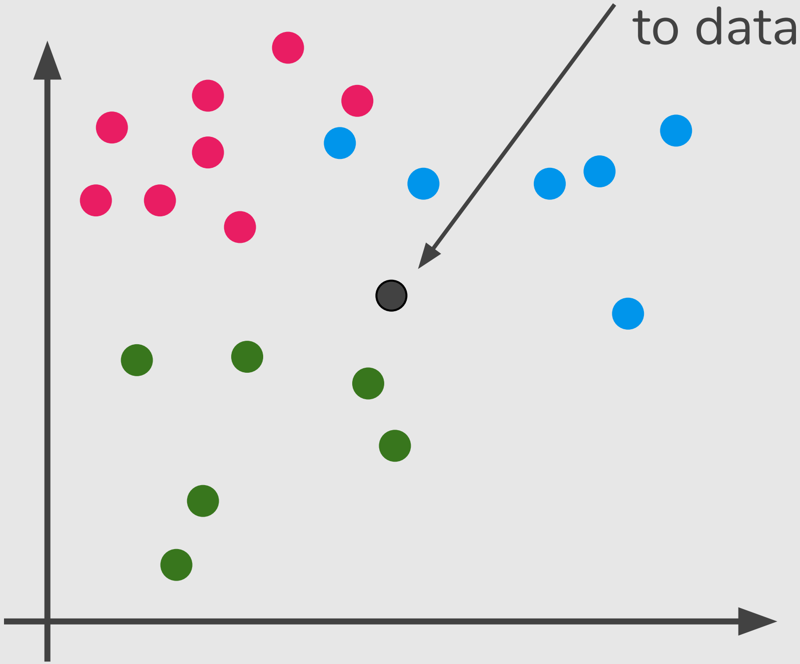
What if we have 3 classes?



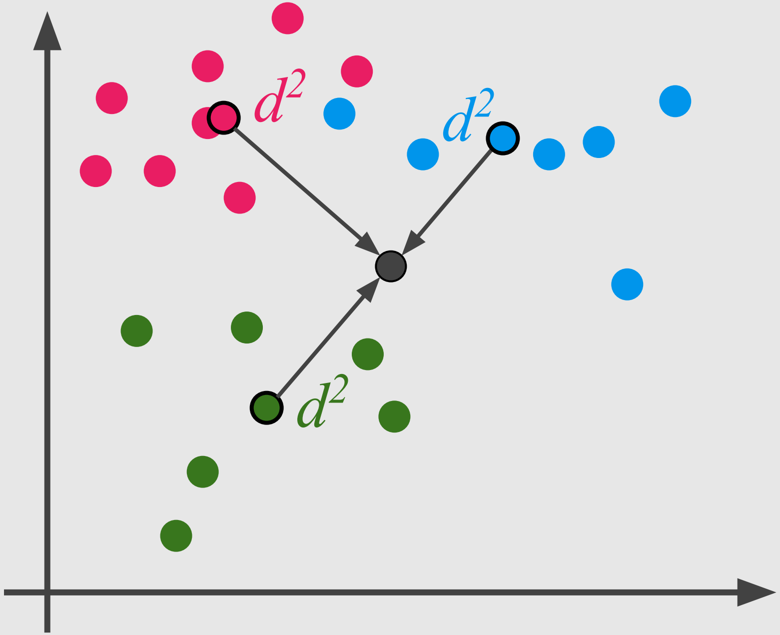
How we measure the distance among the means?

What if we have 3 classes?

Find the point that is central to data.

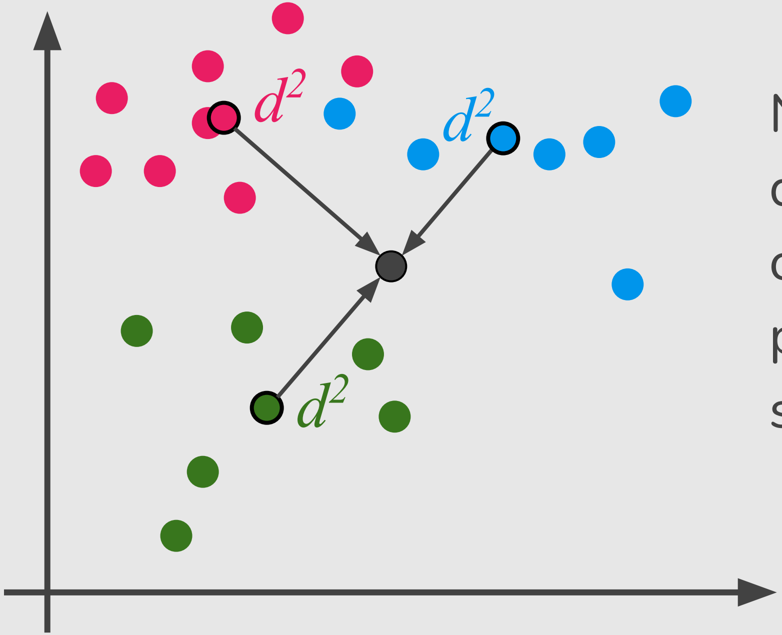


What if we have 3 classes?



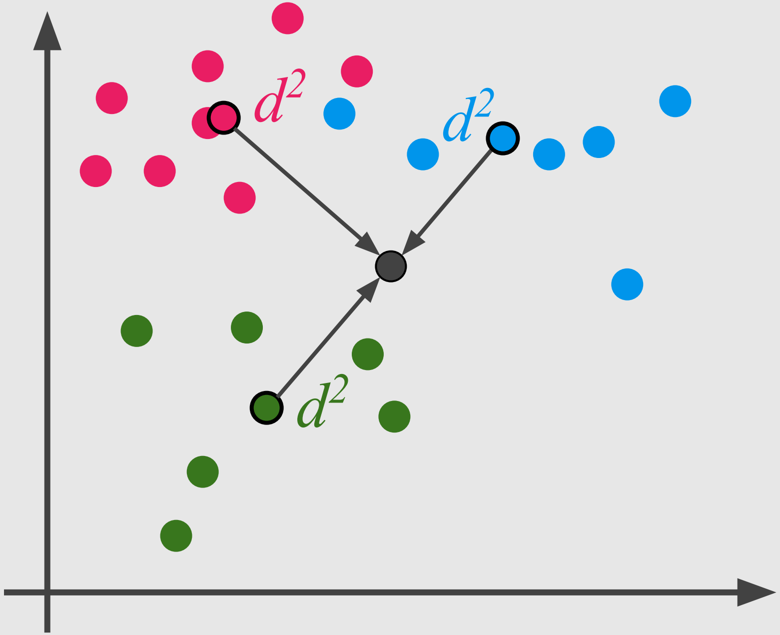
Then measure the distance between a point that is central in each class and the main central point.

What if we have 3 classes?



Now maximize the distance between each class and the central point while minimize the scatter for each class.

What if we have 3 classes?

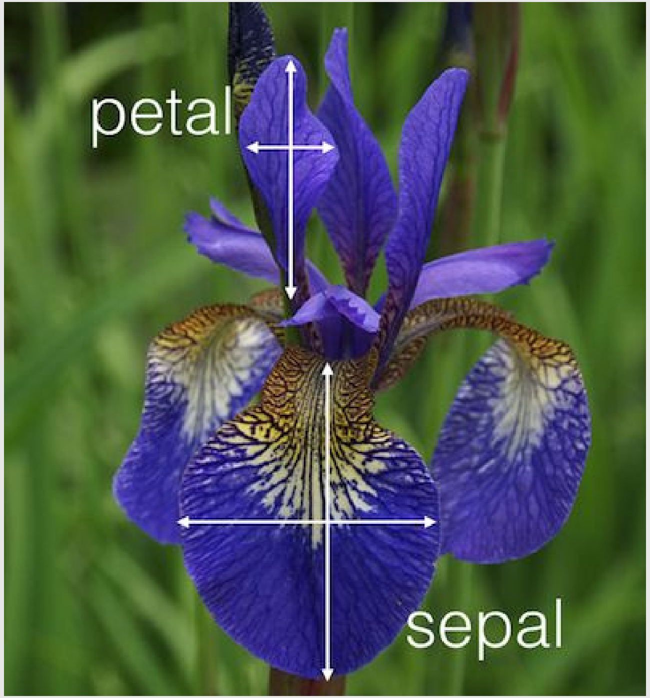


$$\frac{d^2 + d^2 + d^2}{s^2 + s^2 + s^2}$$

LDA in a Nutshell (Eigen Decomposition)

1. Compute the d-dimensional mean vectors for the different classes.
2. Compute the scatter matrices (between-class S_B and within-class S_W).
3. Compute the eigenvectors ($u_1, u_2, \ldots, u_d$) and eigenvalues ($\lambda_1, \lambda_2, \ldots, \lambda_d$) for the scatter matrices $S_W^{-1} S_B$.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors.
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace.

LDA Step by Step



http://sebastianraschka.com/Articles/2014_python_lda.html

150 iris flowers from three different species.

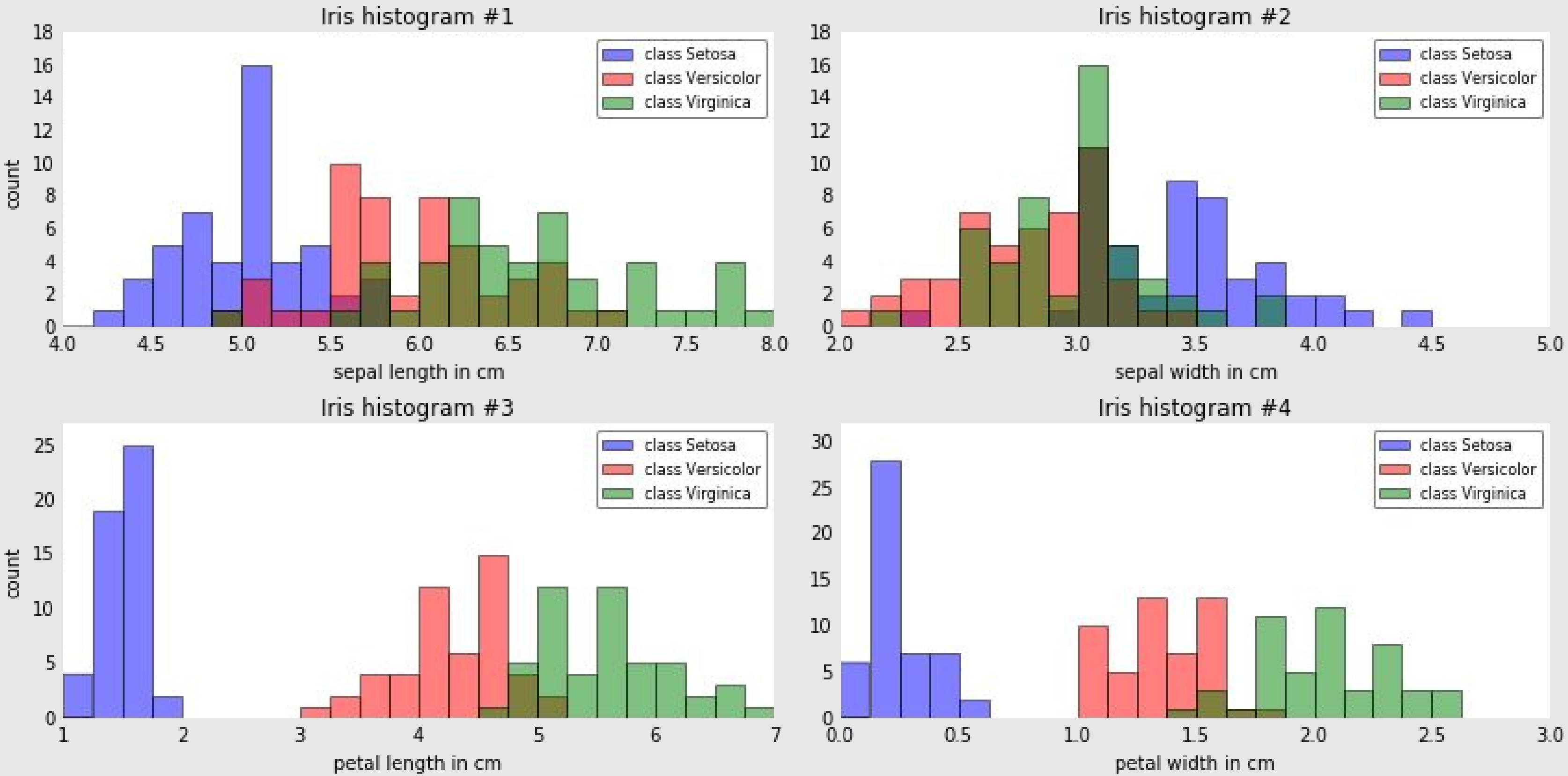
The three classes in the Iris dataset:

1. Iris-setosa (n=50)
2. Iris-versicolor (n=50)
3. Iris-virginica (n=50)

The four features of the Iris dataset:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

LDA Step by Step



LDA Step by Step

1. Compute the d-dimensional mean vectors for the different classes.

LDA Step by Step

1. Compute the d-dimensional mean vectors for the different classes.

$$\mu_1 : [\ 5.01\ \ 3.42\ \ 1.46\ \ 0.24\ ]$$
$$\mu_2 : [\ 5.94\ \ 2.77\ \ 4.26\ \ 1.33\ ]$$
$$\mu_3 : [\ 6.59\ \ 2.97\ \ 5.55\ \ 2.03\ ]$$

LDA Step by Step

2. Compute the **scatter matrices** (between-class S_B and within-class S_W)

   Within-class scatter matrix S_W:

$$S_W = \sum_{i=1}^{c} S_i \text{ , where } S_i = \sum_{x \in D_i}^{n} (x - \mu_i)(x - \mu_\mathrm{i})^\mathrm{T}$$

LDA Step by Step

2. Compute the **scatter matrices** (between-class S_B and within-class S_W)

   Within-class scatter matrix S_W:

$$\begin{bmatrix} 38.96 & 13.68 & 24.61 & 5.66 \\ 13.68 & 7.04 & 8.12 & 4.91 \\ 24.61 & 8.12 & 27.22 & 6.25 \\ 5.66 & 4.91 & 6.25 & 6.18 \end{bmatrix}$$

LDA Step by Step

2. Compute the **scatter matrices** (between-class S_B and within-class S_W)

   Between-class scatter matrix S_B:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^{\mathrm{T}}$$

   where μ is the overall mean, and μ_i and N_i are the sample mean and sizes of the respective classes.

LDA Step by Step

2. Compute the **scatter matrices** (between-class S_B and within-class S_W)

   Between-class scatter matrix S_B:

$$\begin{bmatrix} 63.21 & -19.53 & 165.16 & 71.36 \\ -19.53 & 10.98 & -56.05 & -22.49 \\ 65.16 & -56.05 & 436.64 & 186.91 \\ 71.36 & -22.49 & 186.91 & 80.60 \end{bmatrix}$$

LDA Step by Step

3. Compute the eigenvectors $(u_1, u_2, \ldots, u_d)$ and eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_d)$ for the scatter matrices $S_W^{-1} S_B$.

u_1: $\begin{pmatrix} -0.205 \\ -0.387 \\ 0.546 \\ 0.714 \end{pmatrix}$ u_2: $\begin{pmatrix} -0.009 \\ -0.589 \\ 0.254 \\ -0.767 \end{pmatrix}$ u_3: $\begin{pmatrix} 0.179 \\ -0.318 \\ -0.366 \\ 0.601 \end{pmatrix}$ u_4: $\begin{pmatrix} 0.179 \\ -0.318 \\ -0.366 \\ 0.601 \end{pmatrix}$

λ_1: 3.23e+01 λ_2: 2.78e-01 λ_3: -4.02e-17 λ_4: -4.02e-17

LDA Step by Step

4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors.

   Eigenvalues in decreasing order:

   32.27
   0.27
   5.71e-15
   5.71e-15

LDA Step by Step

4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors.

Eigenvalues in decreasing order:

- 32.27
- 0.27
- 5.71e-15
- 5.71e-15

Variance explained:

- λ_1: 99.15%
- λ_2: 0.85%
- λ_3: 0.00%
- λ_4: 0.00%

LDA Step by Step

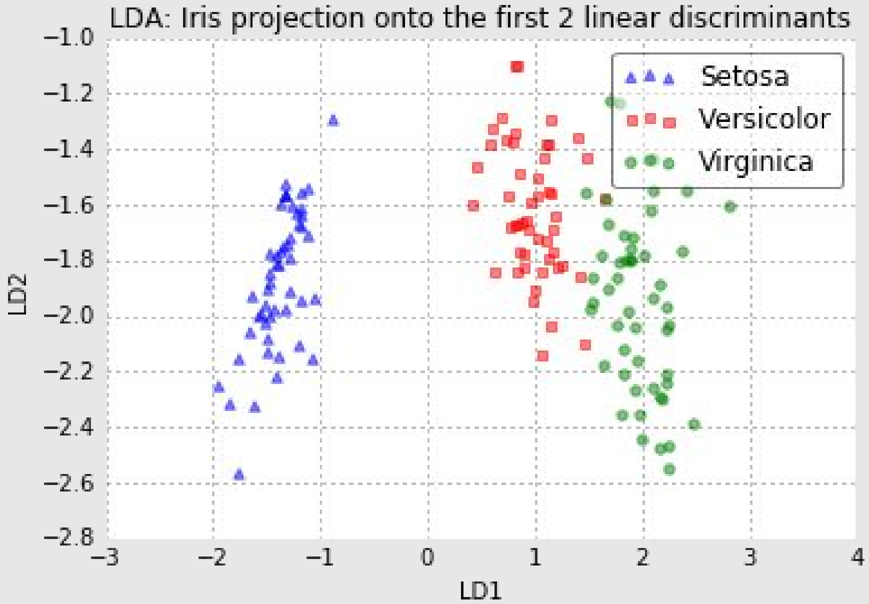
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors.

$$u_1: \begin{pmatrix} -0.205 \\ -0.387 \\ 0.546 \\ 0.714 \end{pmatrix} \quad u_2: \begin{pmatrix} -0.009 \\ -0.589 \\ 0.254 \\ -0.767 \end{pmatrix} \Rightarrow \begin{pmatrix} -0.205 & -0.009 \\ -0.387 & -0.589 \\ 0.546 & 0.254 \\ 0.714 & -0.767 \end{pmatrix}$$

λ_1: 3.23e+01 $\quad$ λ_2: 2.78e-01

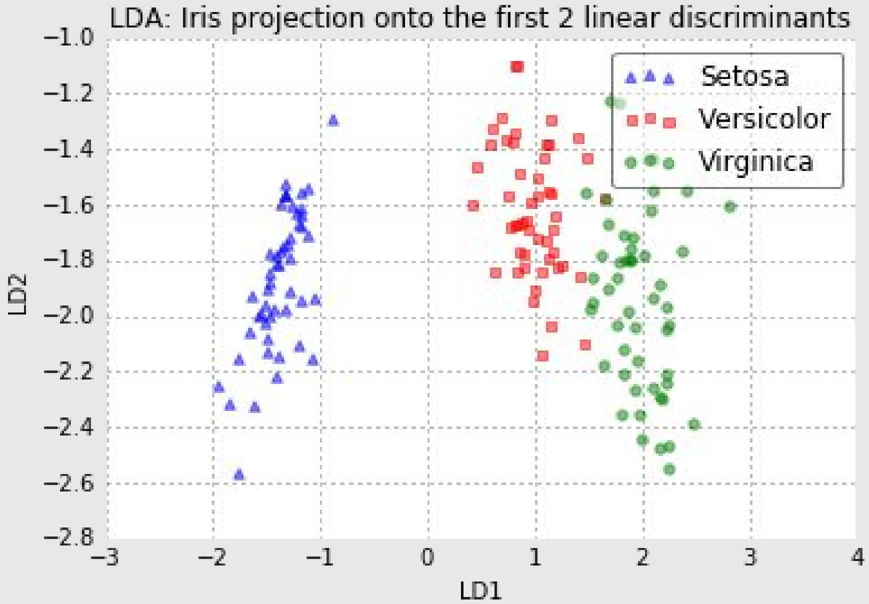
LDA Step by Step

5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace.



LDA Step by Step ➡ http://sebastianraschka.com/Articles/2014_python_lda.html

5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace.



References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 8 “Dimensionality Reduction”
- Pattern Recognition and Machine Learning, Chap. 12 “Continuous Latent Variables”
- Pattern Classification, Chap. 10 “Unsupervised Learning and Clustering”