CS60050: Machine Learning

Autumn 2024

Feature Extraction: PCA and LDA

Somak Aditya Sudeshna Sarkar

CSE Department, IIT Kharagpur

Aug 23 2024

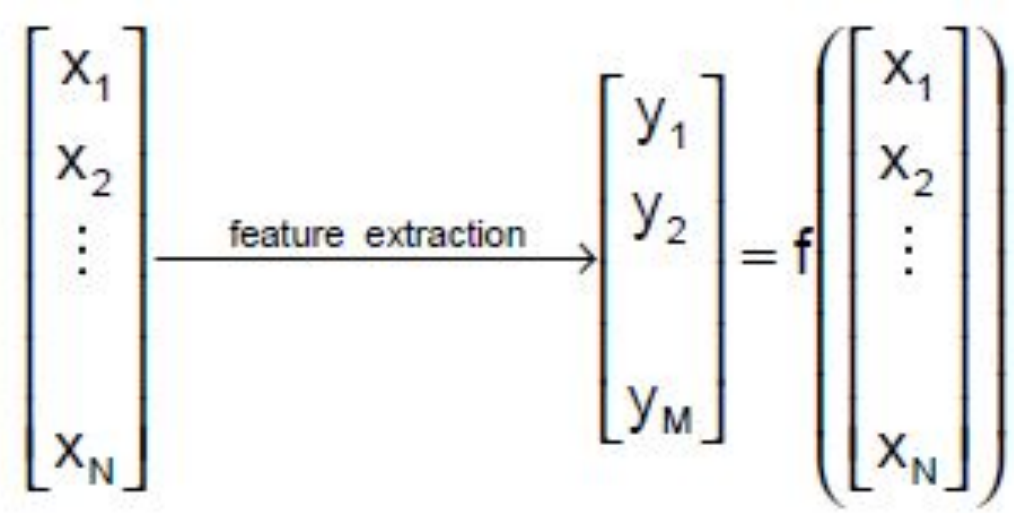
Too many Features

Transforming raw data into features that better represent the underlying problem

- Curse of Dimensionality: Useful to get a lower dimensional representations of the data.

Two ways to "learn" reduced number of features

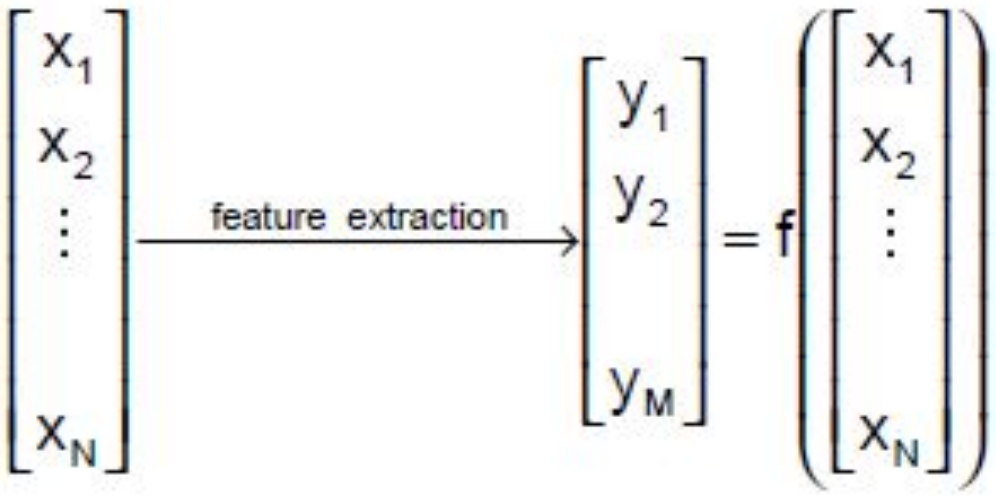
- Feature Selection
- Feature Extraction



Feature Selection and Feature Extraction

Given a set of **n** features

- **Feature selection:** select a subset of **d** features (**d** < **n**) in order to minimize the classification error.
- **Feature Extraction**: Extract hidden (potentially lower dimensional) structure from high dimensional datasets.



Feature Selection vs. Dimensionality Reduction

- Feature Selection
  - When classifying novel patterns, only a small number of features need to be computed (faster classification)
  - Measurement Units (length, weight etc.) of the features are preserved.

- Dimensionality Reduction
  - When classifying novel patterns, all features need to be computed.
  - The measurement units (length, weight etc.) of the features are lost.

Feature Selection*

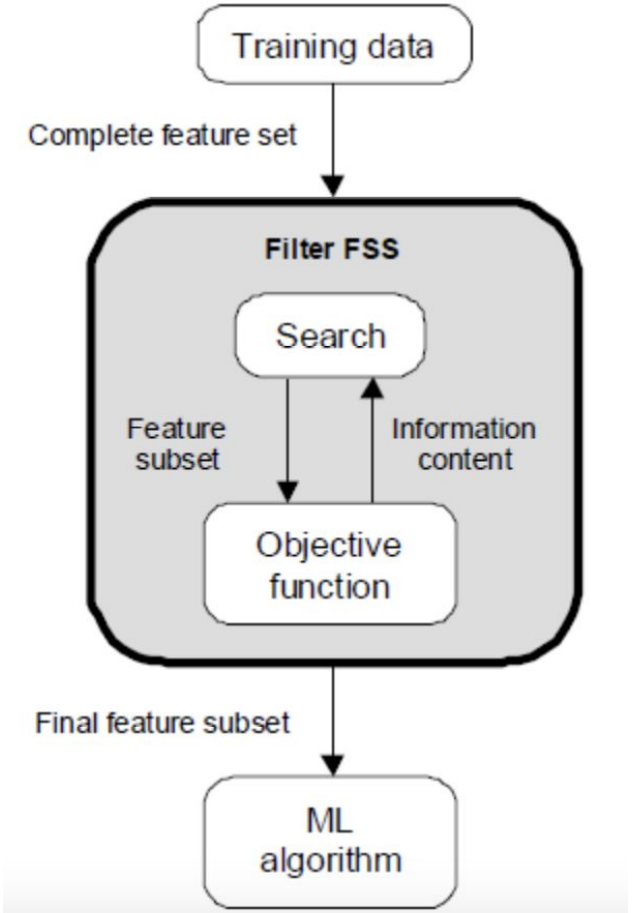
Feature selection is an **optimization** problem.

1. STEP 1: Search the space of possible feature subsets.
2. STEP 2: Pick the subset that is optimal or near-optimal with respect to some objective function.

- Search strategies:
  a) Optimal
  b) Heuristic
- Evaluation strategies

a) Filter methods: Evaluation is independent of the classification algorithm

b) Wrapper methods: Evaluation uses criteria related to the classification algorithm



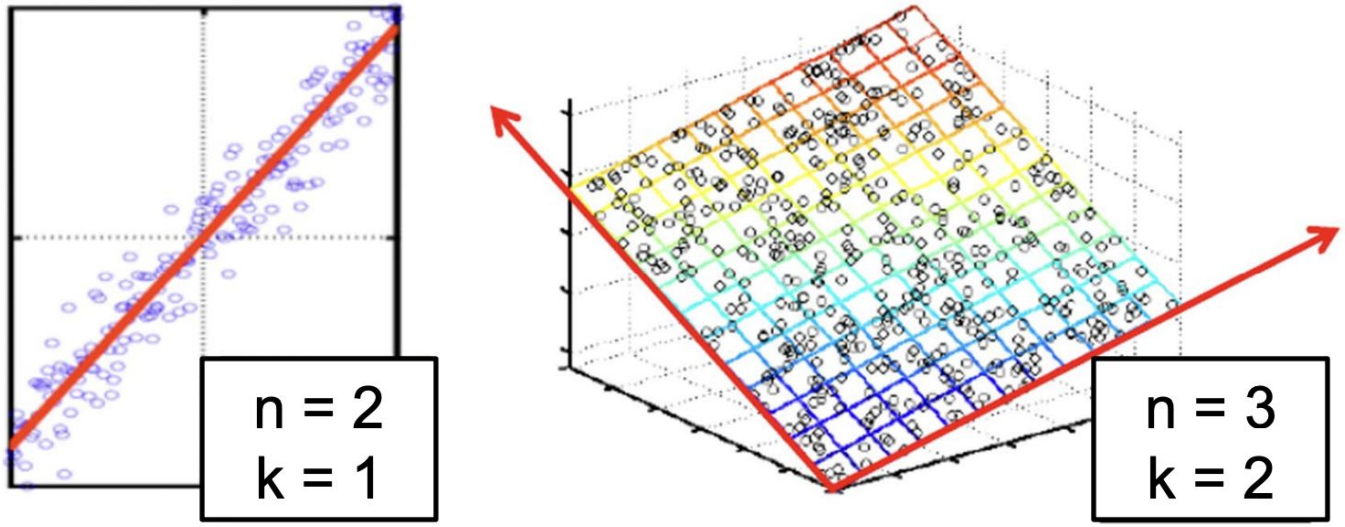
* Details not covered

Feature Extraction: Dimensionality Reduction

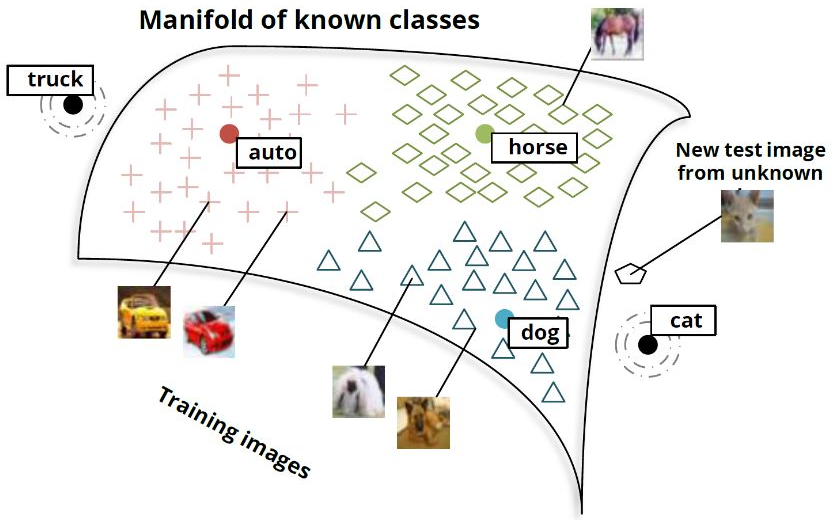
Given data points in n dimensions

- Convert them to data points in $d < n$ dimensions
- With minimal loss of information

Assumption: Data (approximately) lies in a lower-dimensional space.



The Manifold Hypothesis

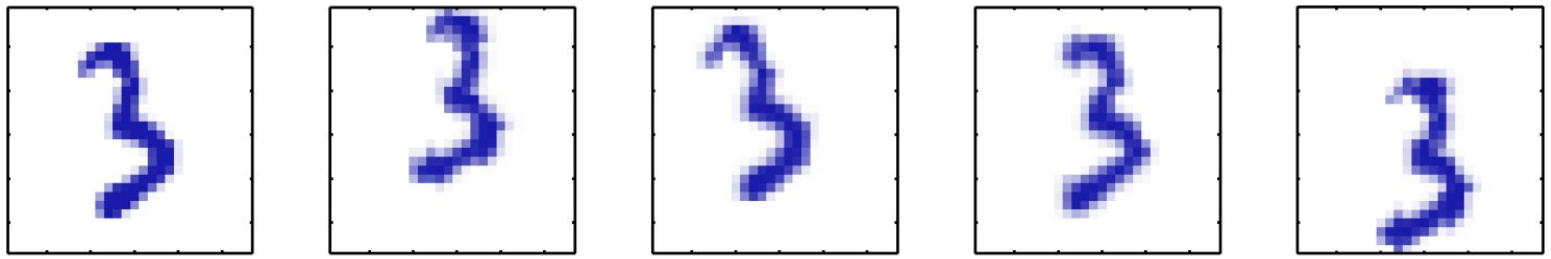


Assumption: The data lie approximately on a manifold of much lower dimension than the input space

- A randomly generated image will almost certainly not look like any real world scene
- Hypothesis: real world images lie on a smooth, low-dimensional manifold

Example (from Bishop)

- Suppose we have a dataset of digits (“3”) perturbed in various ways:



- What operations did I perform? What is the data’s intrinsic dimensionality?
- Here the underlying manifold is nonlinear

Why Dimensionality Reduction?

- Feature extraction: to get a small and effective feature set
  - Compression of the data (features) for computational efficiency
  - Visualization of data
  - Discovering Structure
- Criterion for feature extraction can be different based on different problem settings.
  1. Unsupervised setting: minimize the information loss
  2. Supervised setting: maximize the class discrimination

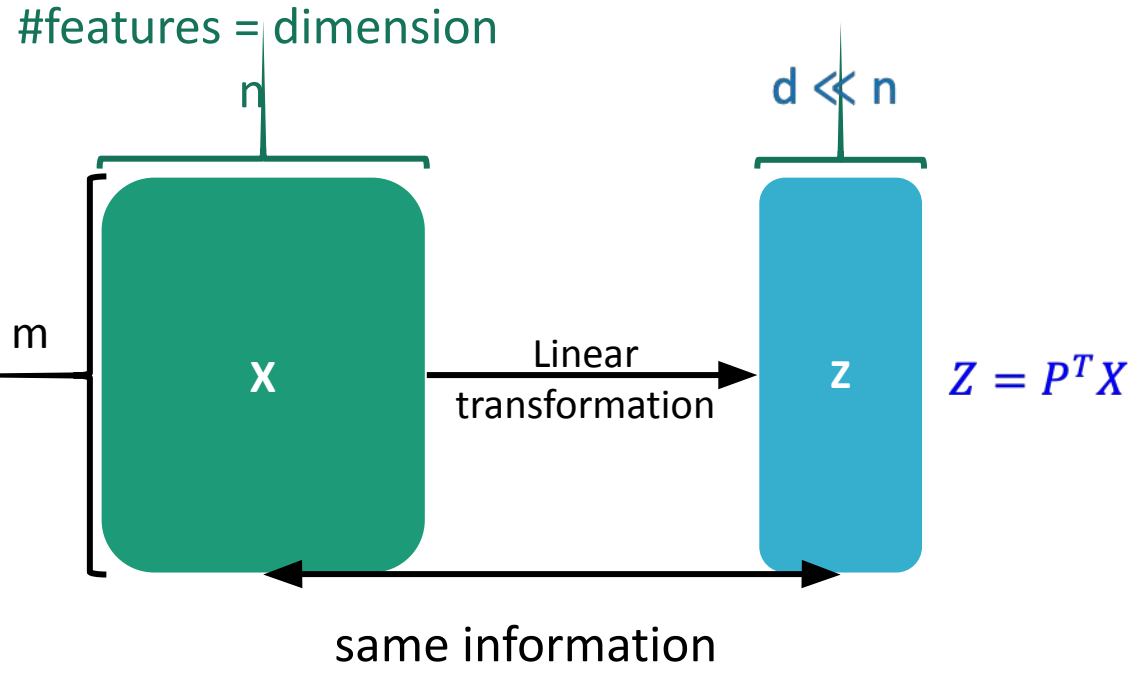
Dimensionality Reduction

- Data set $S = (x_1, x_2, \ldots, x_m)$
- Each x_i has n dimensions
- Data matrix $X \in \mathbb{R}^{m \times n}$
- Reduce dimension to $d < n$:
- A d-dimensional representation of the data in some ways faithful to X

$$Z \in \mathbb{R}^{m \times d}$$

- Find a transformation $P \in \mathbb{R}^{n \times d}$

$$x \in \mathbb{R}^n \rightarrow z = P^T x \in \mathbb{R}^d$$



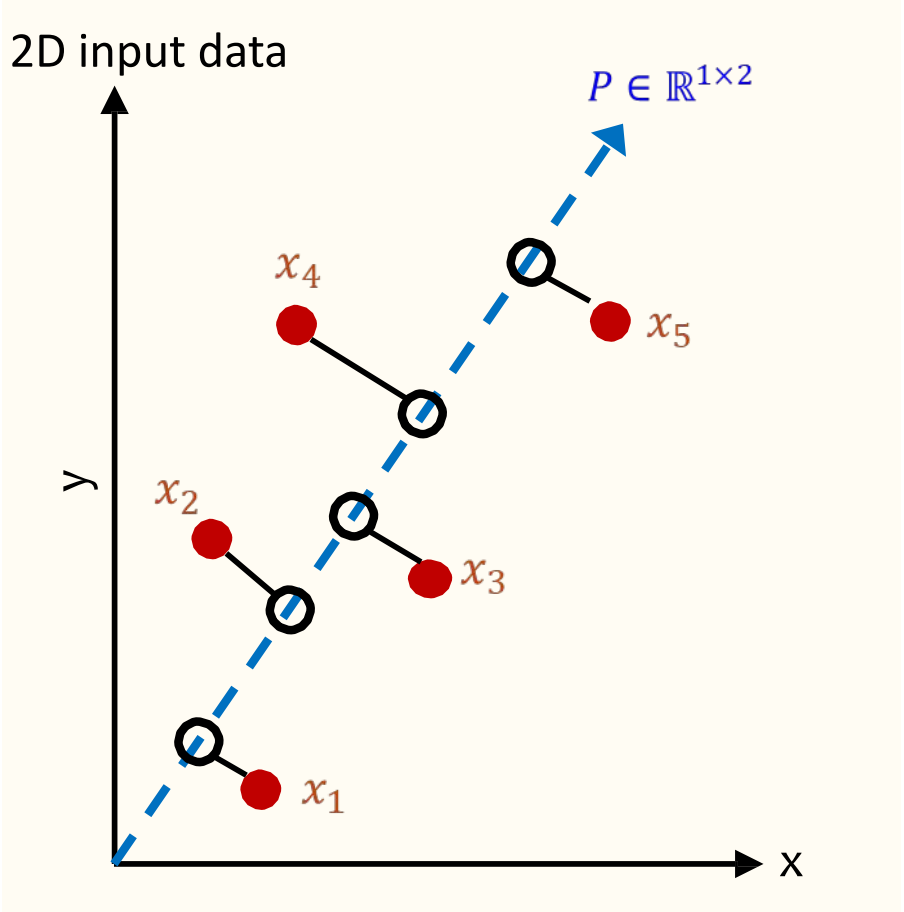
Possibility: Random Projection

- Goal: project from n-dimensions down to d-dimensions
- Data: $D = \{x_i\}_{i=1}^{m}$ where $x_i \in \mathbb{R}^n$
- Algorithm:
  1. Randomly sample matrix $P \in \mathbb{R}^{d \times n}$
  2. Project down: $z_i = Px_i$

     $d\times1 \quad d\times n \quad n\times1$
  3. Project up: $\tilde{x}_i = P^T z_i = P^T(Px_i)$

     $n\times1 \quad n\times d \quad d\times1$

Example: 2D to 1D



https://www.desmos.com/calculator/7x11plypr0

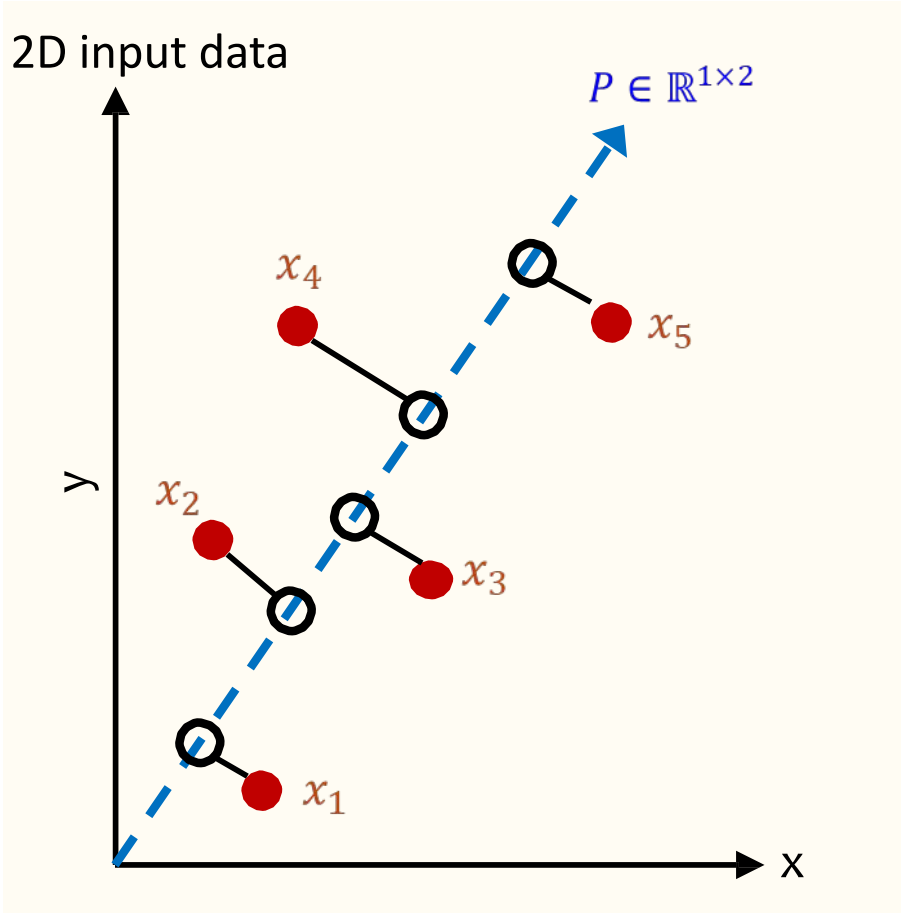
Random Projection

- Goal: project from n-dimensions down to d-dimensions
- Data: $D = \{x_i\}_{i=1}^{m}$ where $x_i \in \mathbb{R}^n$
- Algorithm:
  1. Randomly sample matrix $P \in \mathbb{R}^{d \times n}$
  2. Project down: $z_i = Px_i$

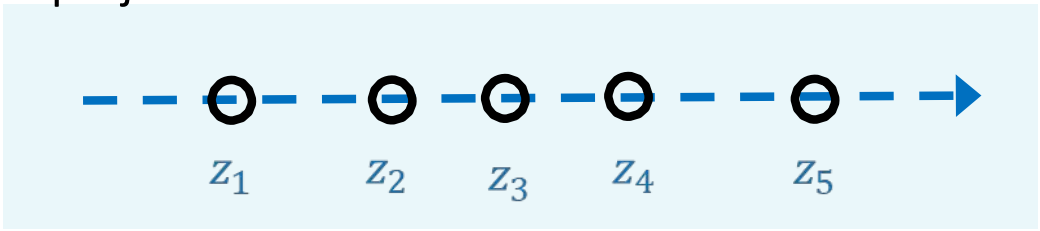
     $d \times 1 \quad d \times n \quad n \times 1$
  3. Project up: $\tilde{x}_i = P^T z_i = P^T(Px_i)$

     $n \times 1 \quad n \times d \quad d \times 1$

Example: 2D to 1D



1D projection



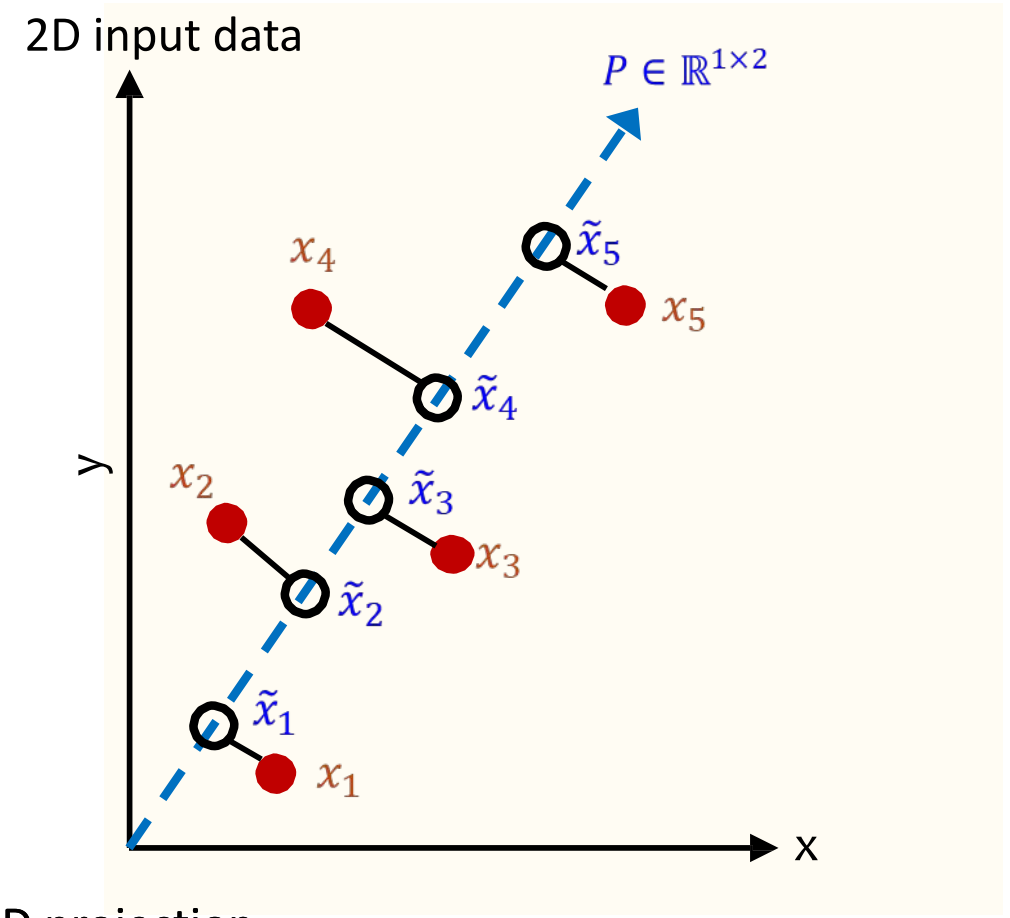
Random Projection

- Goal: project from n-dimensions down to d-dimensions
- Data: $D = \{x_i\}_{i=1}^{m}$ where $x_i \in \mathbb{R}^n$
- Algorithm:
  1. Randomly sample matrix $P \in \mathbb{R}^{d \times n}$
  2. Project down: $z_i = Px_i$

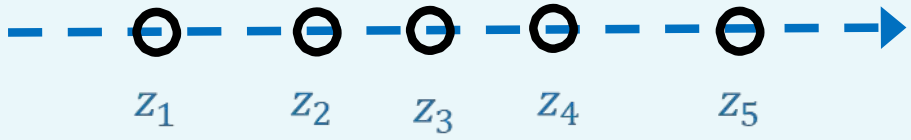
     $d \times 1 \quad d \times n \quad n \times 1$
  3. Project up: $\tilde{x}_i = P^T z_i = P^T(Px_i)$

     $n \times 1 \quad n \times d \quad d \times 1$

Example: 2D to 1D



1D projection



Random Projection

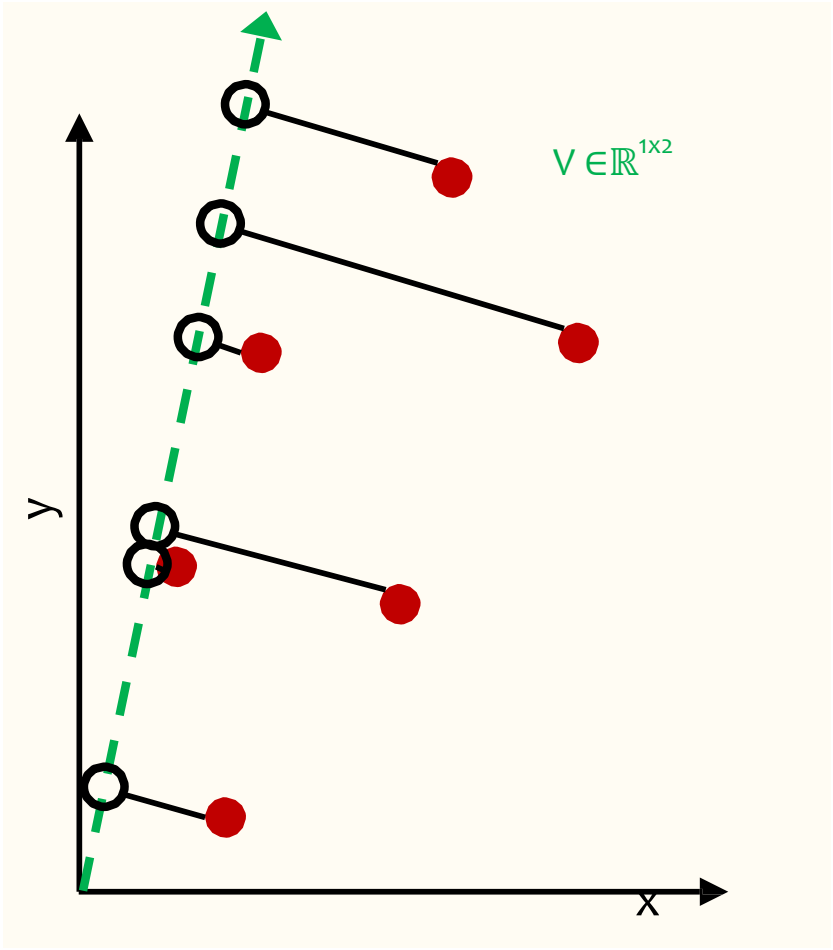
- Goal: project from n-dimensions down to d-dimensions
- Data: $D = \{x_i\}_{i=1}^{m}$ where $x_i \in \mathbb{R}^n$
- Algorithm:
  1. Randomly sample matrix $P \in \mathbb{R}^{n \times d}$
  2. Project down: $z_i = P^T x_i$

     $d \times 1 \quad d \times n \quad n \times 1$
  3. Project up: $\tilde{x}_i = P z_i = P(P^T x_i)$

     $n \times 1 \quad n \times d \quad d \times 1$

Example: 2D to 1D

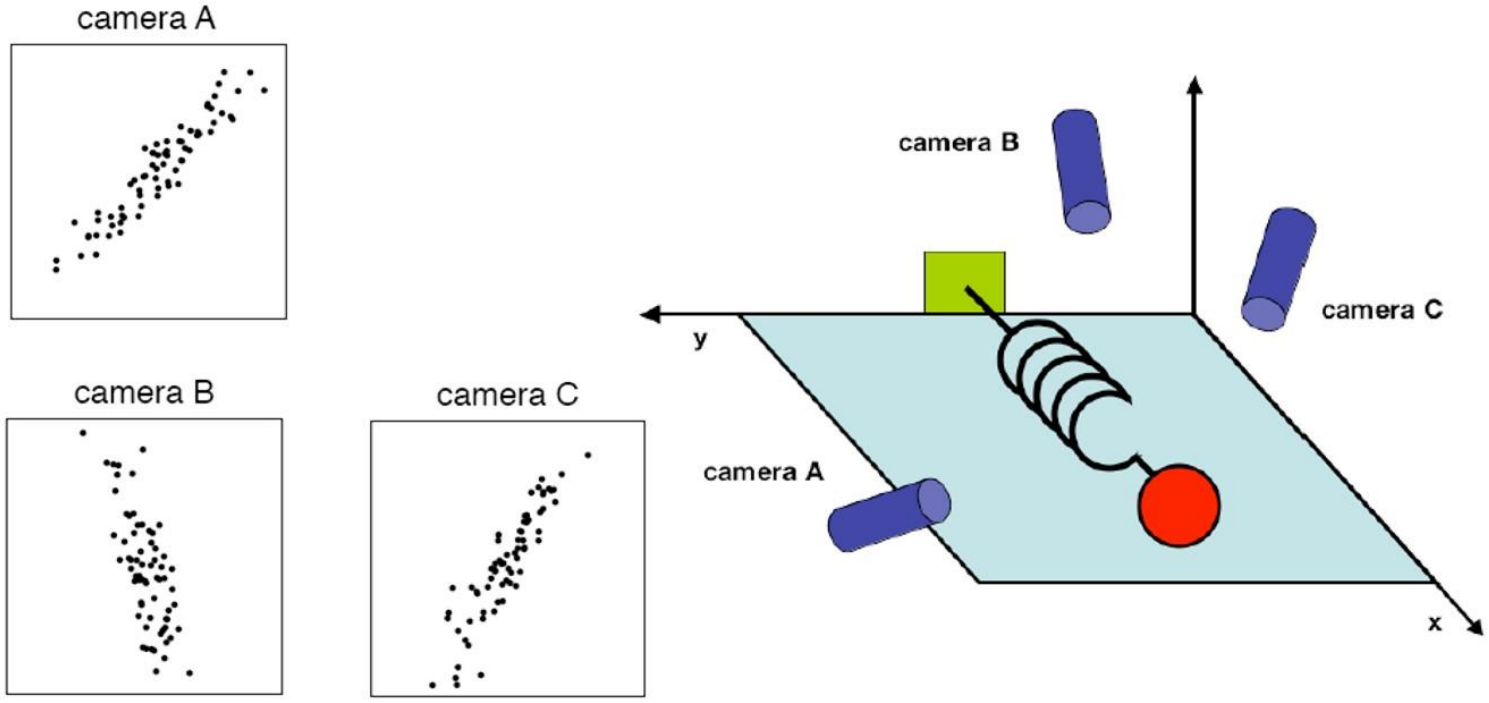
2D input data



Problem: a random projection might give us a poor low dimensional representation of the data

“Optimal” Projection

- Optimal data representation
- 3D data: Find the most informative point of view



- Pull the ball and release.
- Record the coordinates after it finally stops.
  - Cameras take a picture where it stops.
  - 3D becomes 2D representation.

Slide adapted from Javier Hernandez Rivera, MIT MAS 622J, 2010

How Can We Visualize High Dimensional Data?

- E.g., 53 blood and urine tests for 65 patients

	H-WBC	H-RBC	H-Hgb	H-Hct	H-MCV	H-MCH	H-MCHC
A1	8.0000	4.8200	14.1000	41.0000	85.0000	29.0000	34.0000
A2	7.3000	5.0200	14.7000	43.0000	86.0000	29.0000	34.0000
A3	4.3000	4.4800	14.1000	41.0000	91.0000	32.0000	35.0000
A4	7.5000	4.4700	14.9000	45.0000	101.0000	33.0000	33.0000
A5	7.3000	5.5200	15.4000	46.0000	84.0000	28.0000	33.0000
A6	6.9000	4.8600	16.0000	47.0000	97.0000	33.0000	34.0000
A7	7.8000	4.6800	14.7000	43.0000	92.0000	31.0000	34.0000
A8	8.6000	4.8200	15.8000	42.0000	88.0000	33.0000	37.0000
A9	5.1000	4.7100	14.0000	43.0000	92.0000	30.0000	32.0000

Instances

Features

- Difficult to see the correlations between the features...

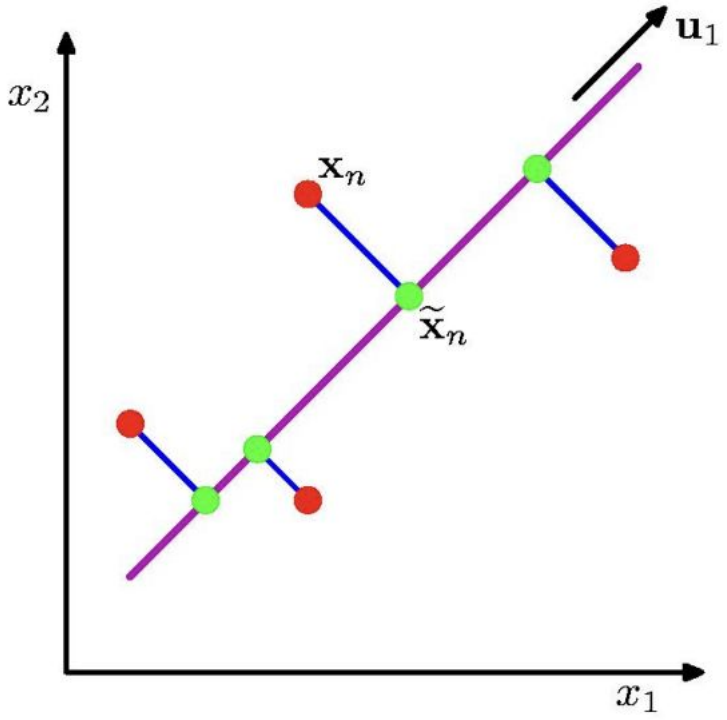
Data Visualization

- Is there a representation better than the raw features?
  - Is it really necessary to show all the 53 dimensions?
  - … what if there are strong correlations between the features?

Could we find *the smallest subspace of the 53-D space* that keeps the *most information* about the original data?

One Solution: Principal Component Analysis

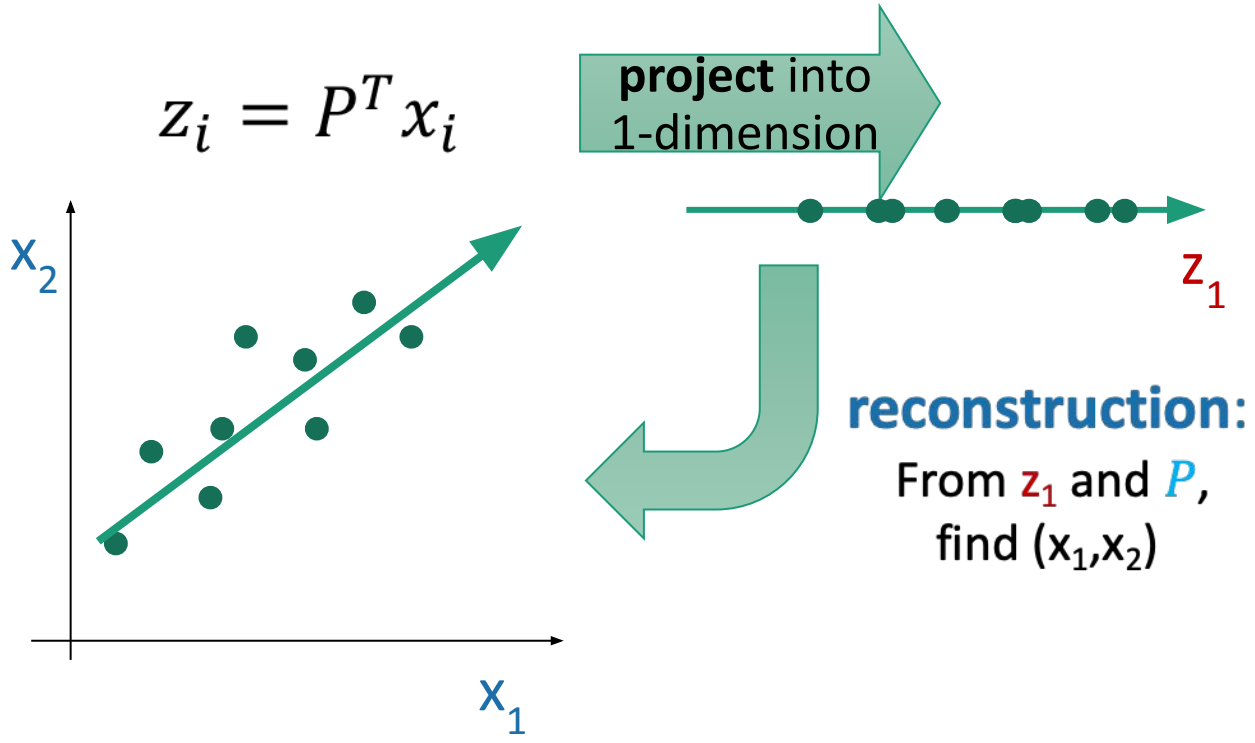
Principle Component Analysis



- Orthogonal projection of data onto lower-dimension linear space that...
  - maximizes variance of projected data (purple line)
  - minimizes mean squared distance between data point and projections (sum of blue lines)

Linear projection and Reconstruction

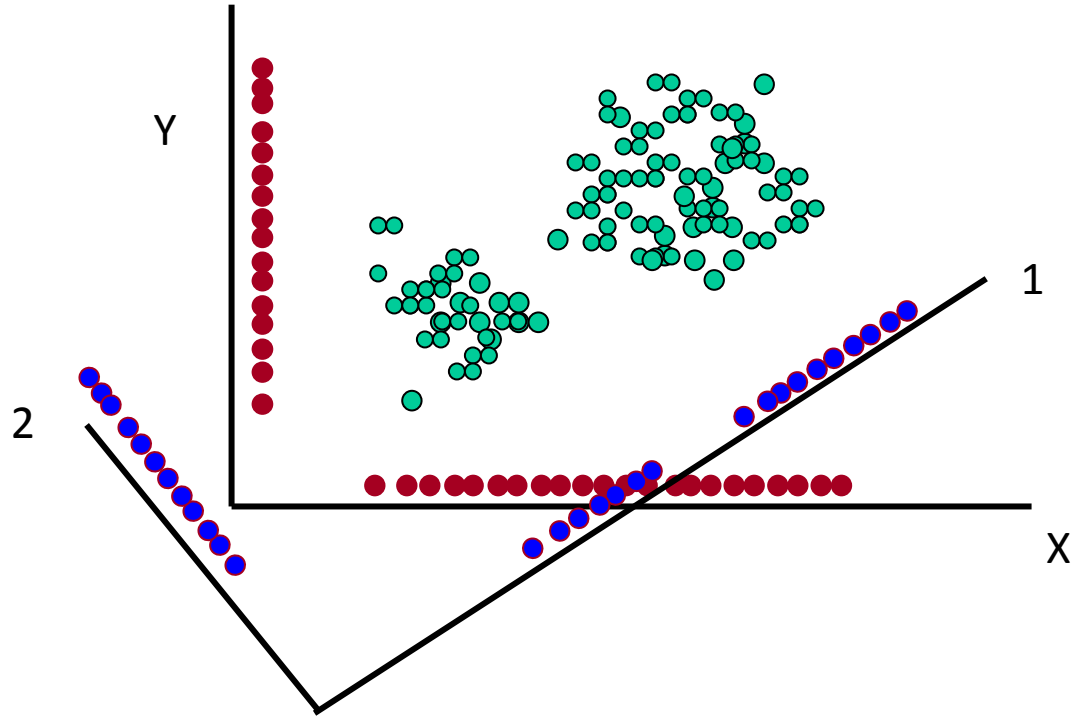
1. Minimizes the projection error.



Slide adapted from CSE546, Univ of Washington

Projected variance

2. Maximize projected variance



Projection onto axis *1* has maximum variance (and incidentally shows the clustered character of the data)

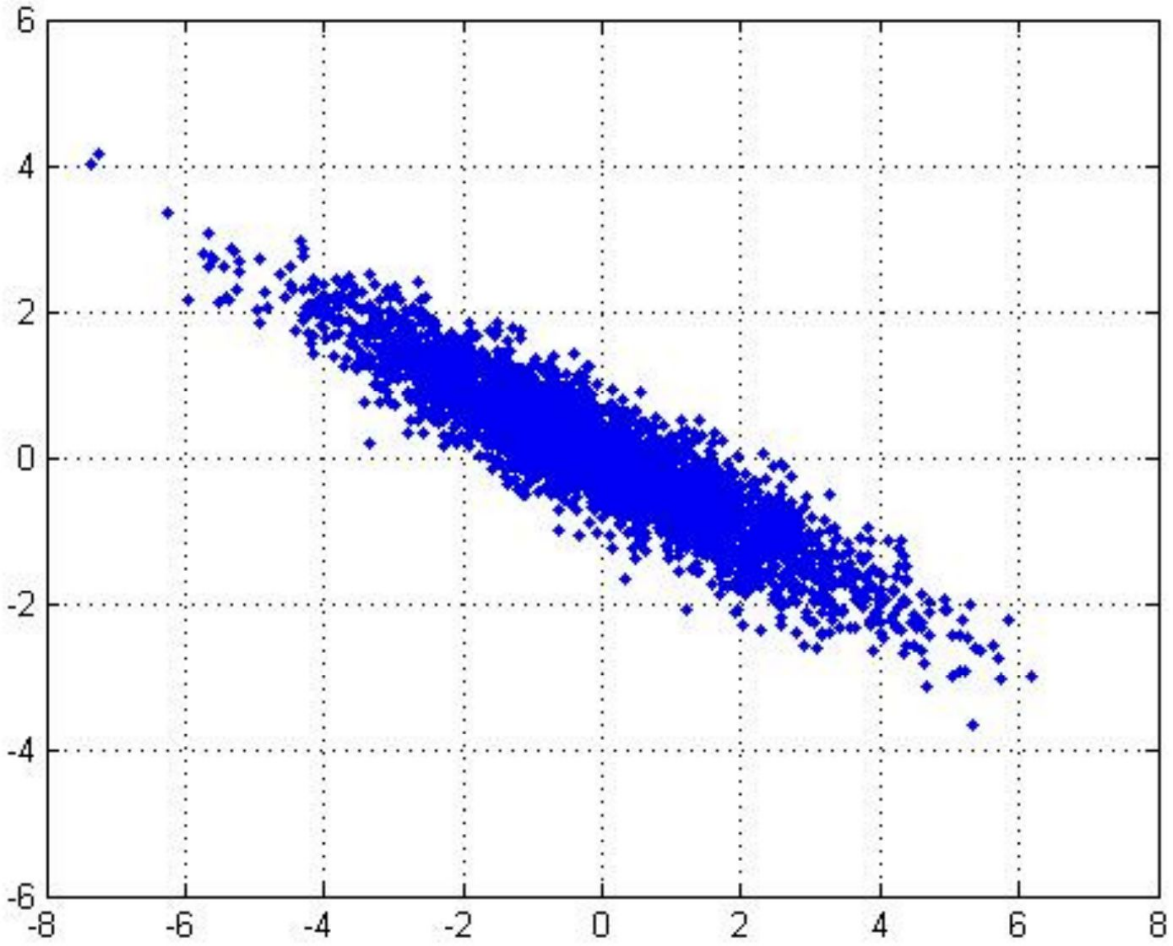
Axis *2* may be able to be eliminated since it accounts for a small amount of the variance

[via Dan Roth]

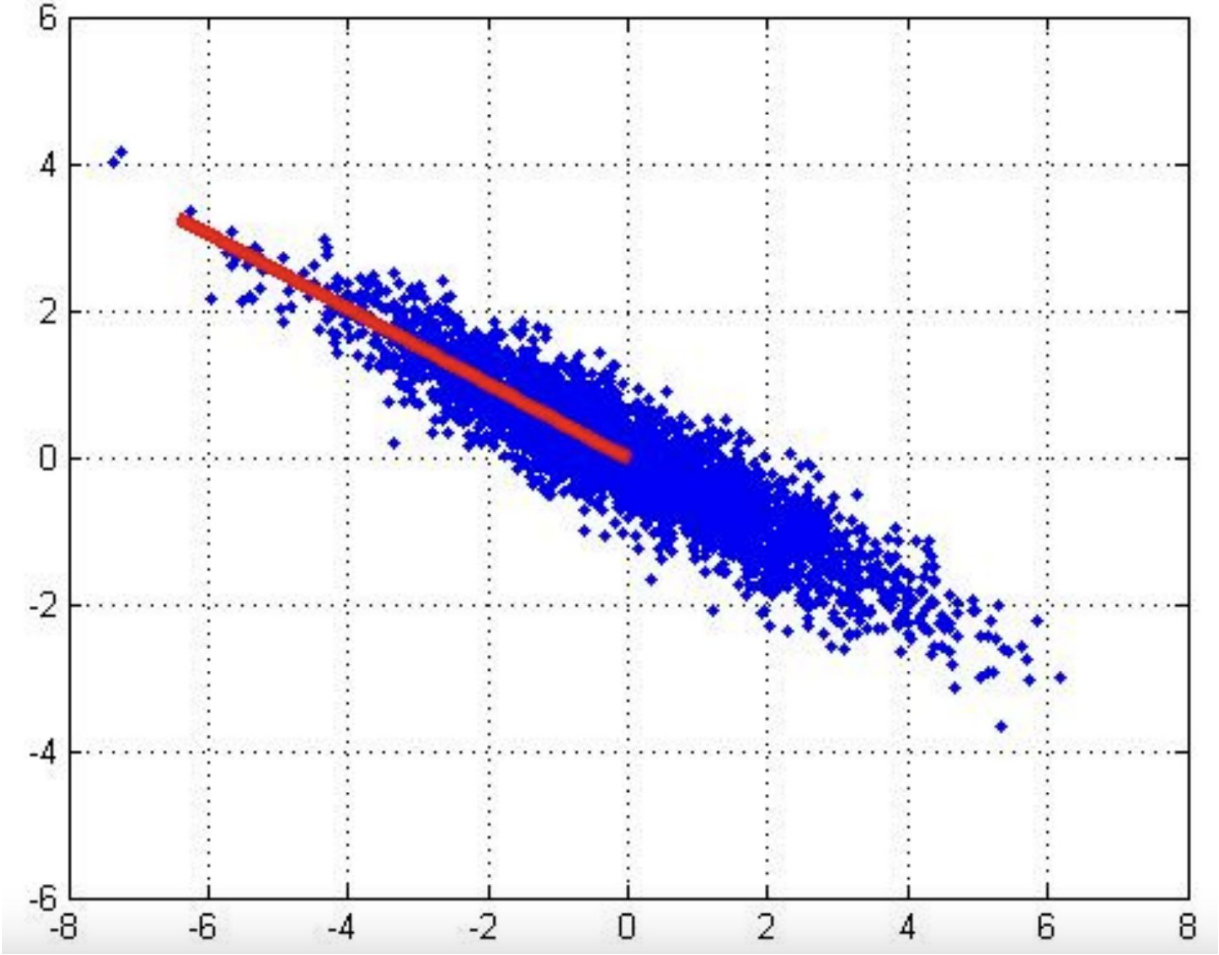
Principal Components (Background)

- **Vectors** originating from the center of mass
- Principal component #1 points in the direction of the **largest variance**
- Each subsequent principal component…
  - is **orthogonal** to the previous ones, and
  - points in the directions of the **largest variance of the residual subspace**

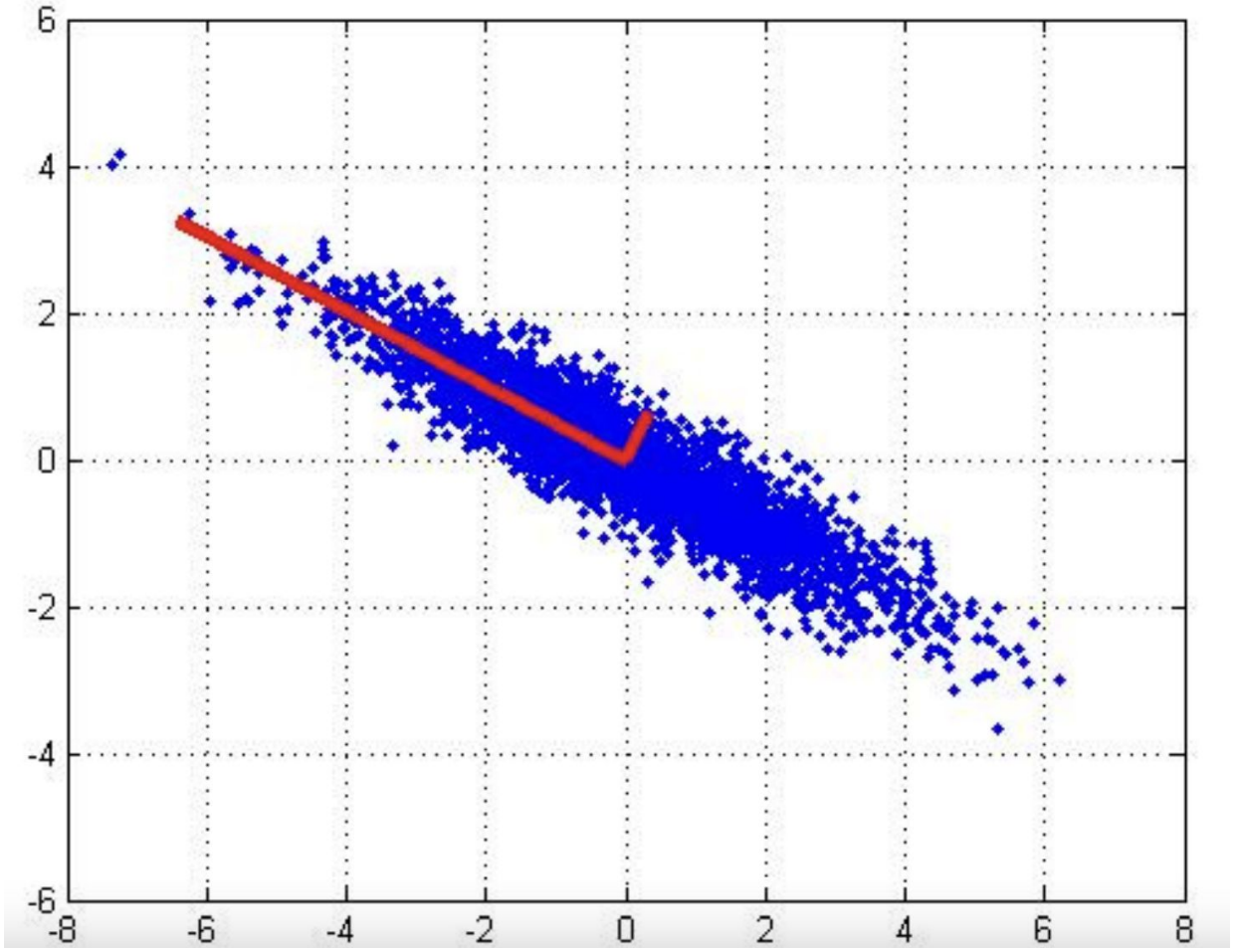
PCA Example: 2D Gaussian Dataset



PCA Example: 1st PCA Axis

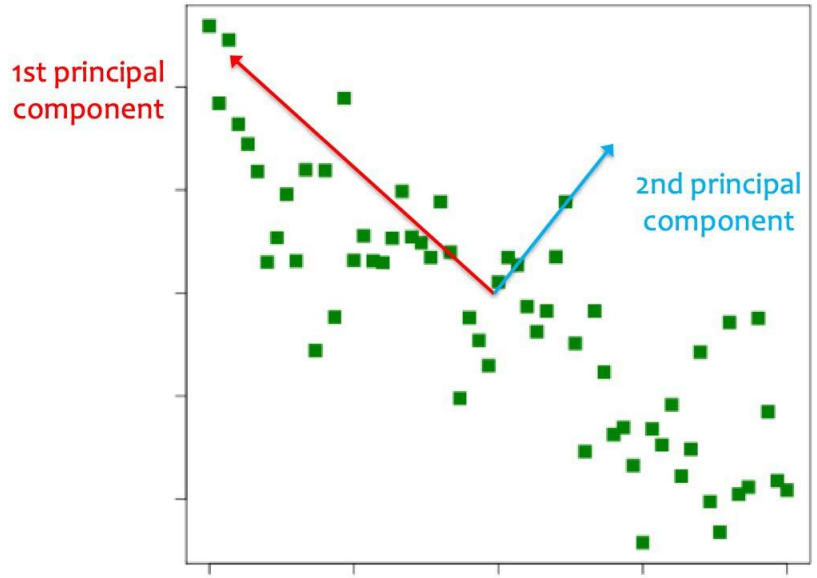


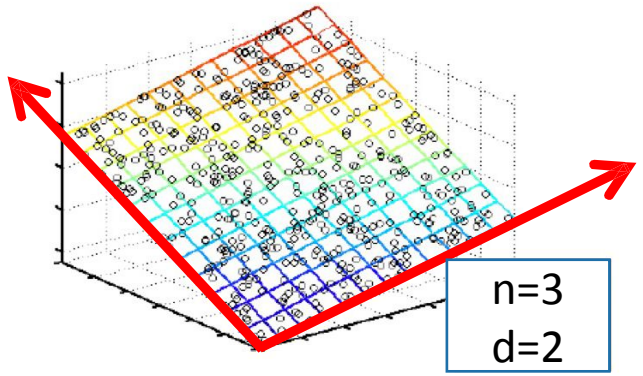
PCA Example: 2nd PCA Axis



Principal Component Analysis (PCA)

- Assume that the data lies on a low d-dimensional linear subspace
  - Goal: identify the axes of that subspace, and project each point onto hyperplane
- Algorithm: find the d eigenvectors with largest eigenvalue

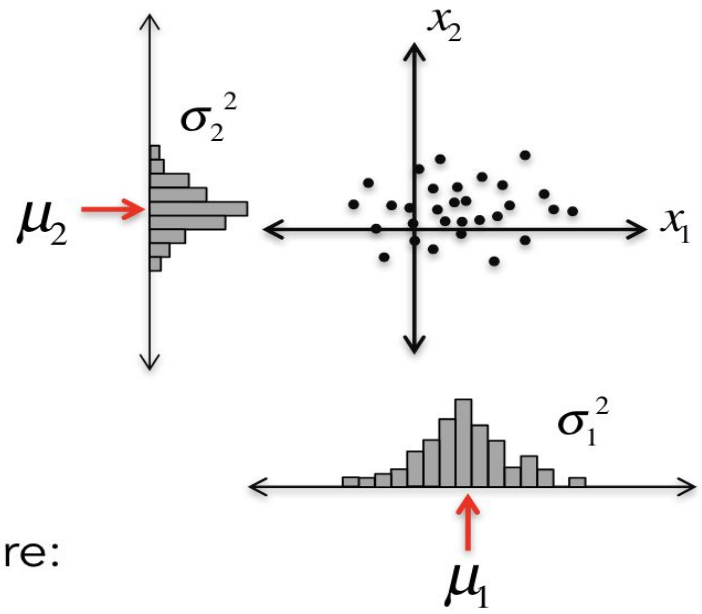




Variance and Covariance

- Now let's say we have m simultaneous observations of variables and . x_1 x_2

$$\begin{pmatrix} x_1^{(j)} \\ x_2^{(j)} \end{pmatrix}, \quad j = 1,2,3....,m$$

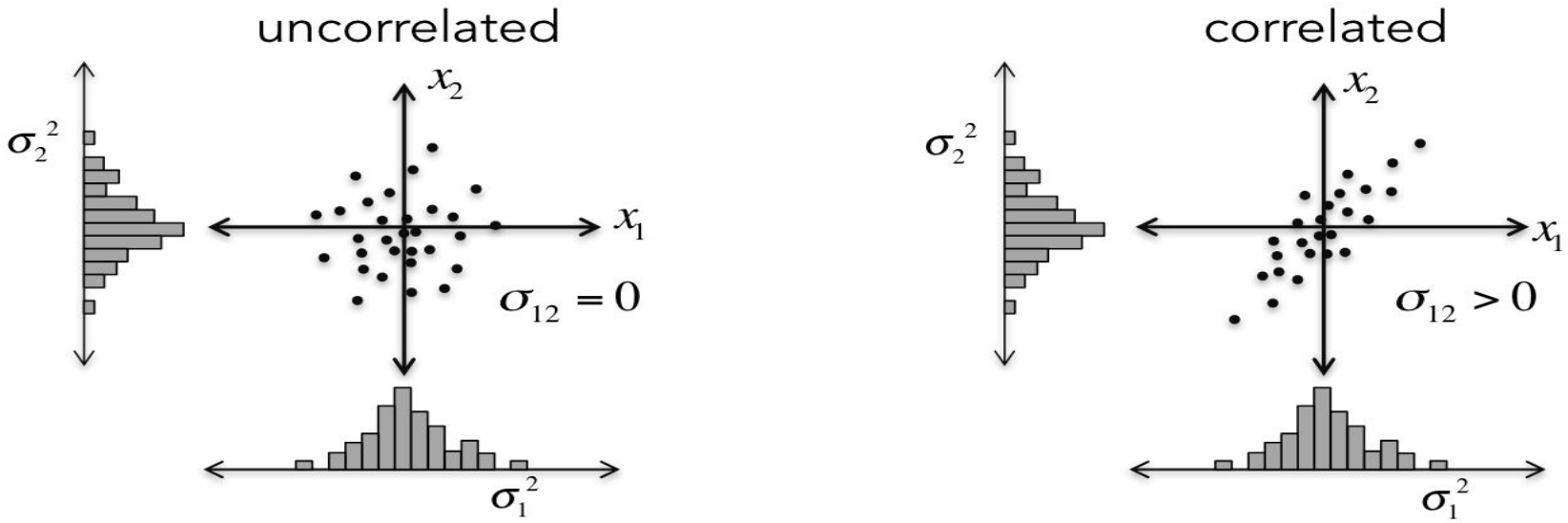


- The mean and variance of x_1 and x_2 are:

$$\mu_1 = \frac{1}{m}\sum_{j=1}^{m} x_1^{(j)} \qquad \mu_2 = \frac{1}{m}\sum_{j=1}^{m} x_2^{(j)}$$

$$\sigma_1^2 = \frac{1}{m}\sum_{j=1}^{m}\left(x_1^{(j)} - \mu_1\right)^2 \qquad \sigma_2^2 = \frac{1}{m}\sum_{j=1}^{m}\left(x_2^{(j)} - \mu_2\right)^2$$

Variance and Covariance



- x_1 has the same variance in both of these cases… also x_2
- So we need another measure to describe the relation between x_1 and x_2 .

covariance

$$\sigma_{12} = \frac{1}{m}\sum_{j=1}^{m}\left(x_1^{(j)} - \mu_1\right)\left(x_2^{(j)} - \mu_2\right)$$

correlation

$$\rho = \frac{\sigma_{12}}{\sqrt{\sigma_1^2 \sigma_2^2}}$$

Example Covariance Matrices

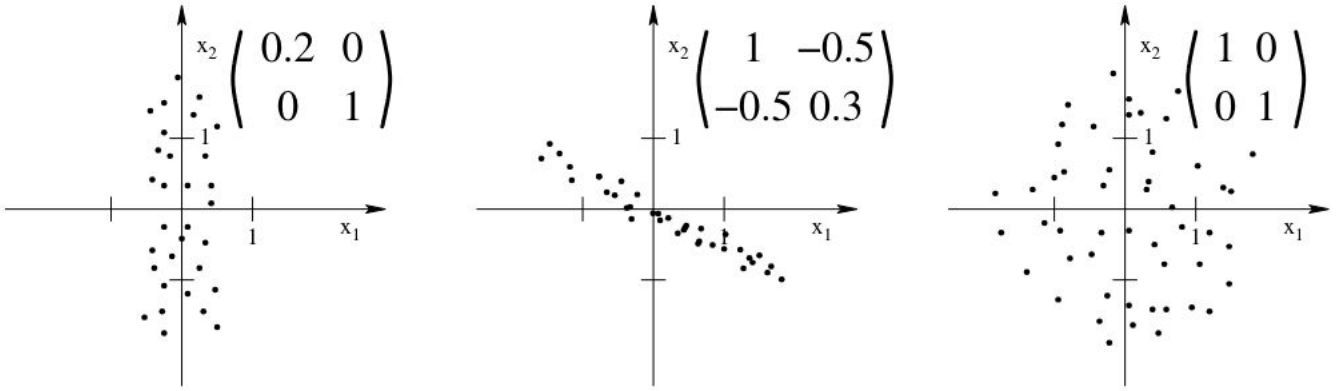


Figure 3: Several data distributions and their covariance matrices.

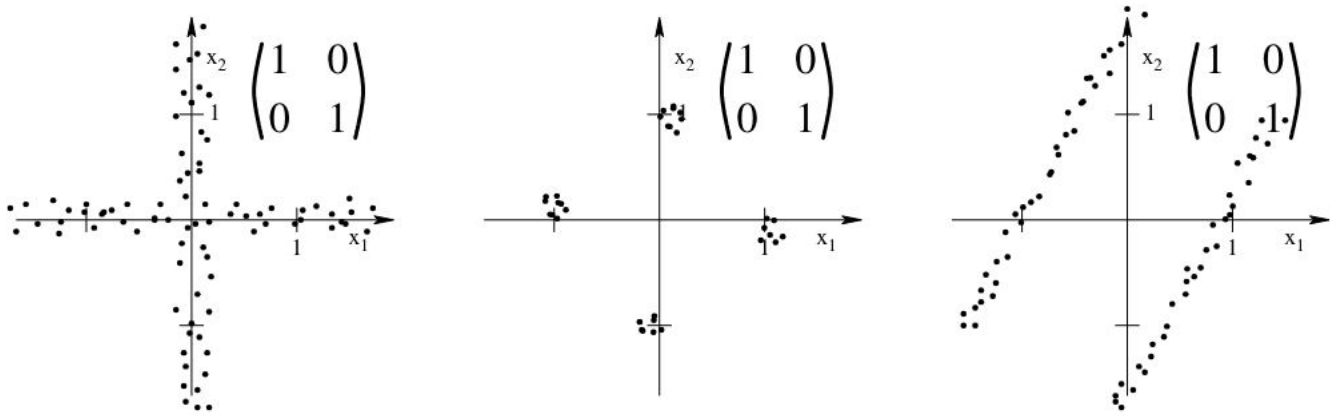


Figure 4: Different data distributions with identical covariance matrices.

Basic PCA algorithm

- Start from $n \times m$ data matrix $\boldsymbol{X}$
- **Recenter**: subtract mean from each row of $\boldsymbol{X}$

$$\boldsymbol{X_c} \leftarrow \boldsymbol{X} - \bar{\boldsymbol{X}}$$

- **Compute covariance** matrix:

$$\Sigma \leftarrow \frac{1}{m} \boldsymbol{X_c} \boldsymbol{X_c^T}$$

- Find **eigen vectors and values** of $\Sigma = V \Lambda V^T$ (V =eigenvectors, Λ =diagonal matrix of eigenvalues, $n \times n$ matrices)
- **Principal components:** Find d eigen vectors with highest eigen values (d<n)
  - Assemble them into $n \times d$ matrix V
- **Project:** $\mathbf{Z} = \boldsymbol{V^T X}$

Covariance Matrix

- $\Sigma \leftarrow \frac{1}{m} \boldsymbol{X_c} \boldsymbol{X_c^T}$

- $= \frac{1}{m} \begin{bmatrix} x_{11} & x_{12} & .. & x_{1m} \\ x_{21} & x_{22} & .. & x_{2m} \end{bmatrix} \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \\ \dots & \dots \\ x_{1m} & x_{2m} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$

- $\sigma_{12} = \sigma_{21}$, Therefore symmetric

PCA algorithm computations

- Start from $n \times m$ data matrix $\boldsymbol{X}$
- **Recenter**: subtract mean from each row of $\boldsymbol{X}$

$$\boldsymbol{X_c} \leftarrow \boldsymbol{X} - \bar{\boldsymbol{X}}$$

- **Compute covariance** matrix:

$$\Sigma \leftarrow \frac{1}{m} \boldsymbol{X_c} \boldsymbol{X_c^T}$$

- Find **eigen vectors and values** of Σ
- **Principal components:** Find d eigen vectors with highest eigen values
- Assemble them into $n \times d$ matrix V
- **Project:** $\mathbf{Z} = \boldsymbol{V^T X}$

- Zero mean: $x_{ij} = x_{ij} - \mu_i$

$$\mu_i = \frac{1}{m} \sum_{j=1}^{m} x_{ij}$$

- Unit variance: $x_{ij} = \frac{x_{ij}}{\sigma_j}$

$$\sigma_i = \frac{1}{m} \sum_{j=1}^{m} \left(x_{ij} - \mu_i \right)^2$$

- Covariance Matrix: $\Sigma = XX^T$
- Compute eigenvectors of Σ : V^T
- Project X onto the d principal components

$$\underset{d \times m}{\boldsymbol{Y}} = \underset{d \times n}{\boldsymbol{V^T}} \times \underset{n \times m}{\boldsymbol{X}}$$

PCA Example

Data

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Center Data

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

Covariance Matrix

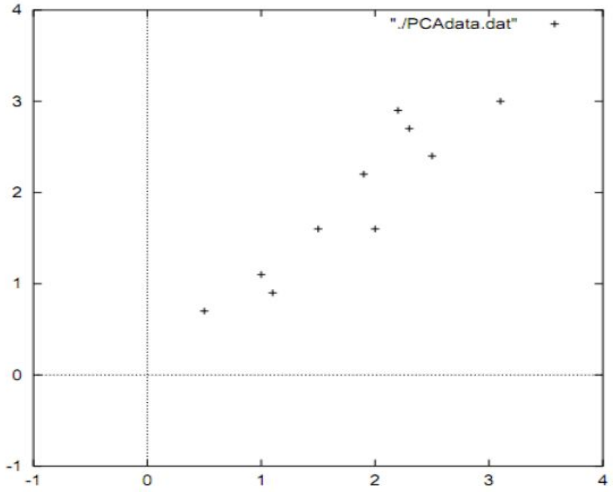
0.61655	0.61544
0.61544	0.71655

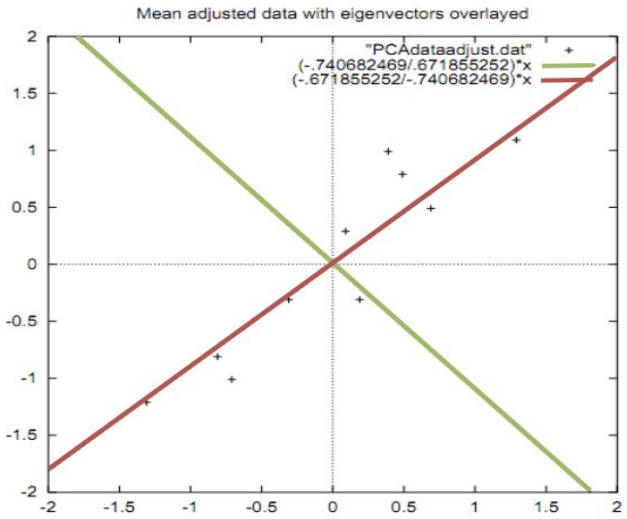
Eigenvectors

-0.73518	-0.67787
0.67787	-0.73518

Eigenvalues

0.04908	0
0	1.28403





Why are the Eigenvectors of Covariance Matrix the Principal Components?

From either

1) the maximum-variance or
2) minimum-square-residual objective,

it can be shown that the principal components are eigenvectors of the data's covariance matrix.

PCA Objective Functions

- What is the first principal component p1 chosen by PCA?

1. Option 1: The vector that minimizes the reconstruction error

$$v_1 = \underset{v:\|v\|^2=1}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \|x_i - \tilde{x}_i\|^2$$

$$= \underset{v:\|v\|^2=1}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \|x_i - V(V^T x_i)\|^2$$

2. Option 2: The vector that maximizes the variance

$$v_1 = \underset{v:\|v\|^2=1}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} (V^T x_i)^2$$

PCA Objective Functions

1. Option 1: Reconstruction error

$$v_1 = \underset{v:\|v\|^2=1}{\text{argmin}} \frac{1}{m}\sum_{i=1}^{m} \|x_i - V(V^T x_i)\|^2$$

$$\equiv \underset{v:\|v\|^2=1}{\text{argmin}} \frac{1}{m}\sum_{i=1}^{m} \|x_i - V(V^T x_i)\|^2$$

$$\equiv \underset{v:\|v\|^2=1}{\text{argmin}} \frac{1}{m}\sum_{i=1}^{m} (\|x_i\|^2 - (V^T x_i)^2)$$

$$\equiv \underset{v:\|v\|^2=1}{\text{argmin}} \frac{1}{m}\sum_{i=1}^{m} (\text{constant} - (V^T x_i)^2)$$

$(V^T x_i)^2 = x_i^T (VV^T) x_i$ = variance of the projected data

Option 1 = Variance of the original –variance of the projected data

2. Option 2: The vector that maximizes the variance

$$u_1 = \underset{u:\|u\|^2=1}{\text{argmax}} \frac{1}{m}\sum_{i=1}^{m} (U^T x_i)^2$$

$$= \underset{u:\|u\|^2=1}{\text{argmax}} \frac{1}{m}\sum_{i=1}^{m} (U^T x_i)^2$$

$$u_1 = \underset{u:\|u\|^2=1}{\text{argmax}} \frac{1}{m}\sum_{i=1}^{m} (U^T x_i)^2$$

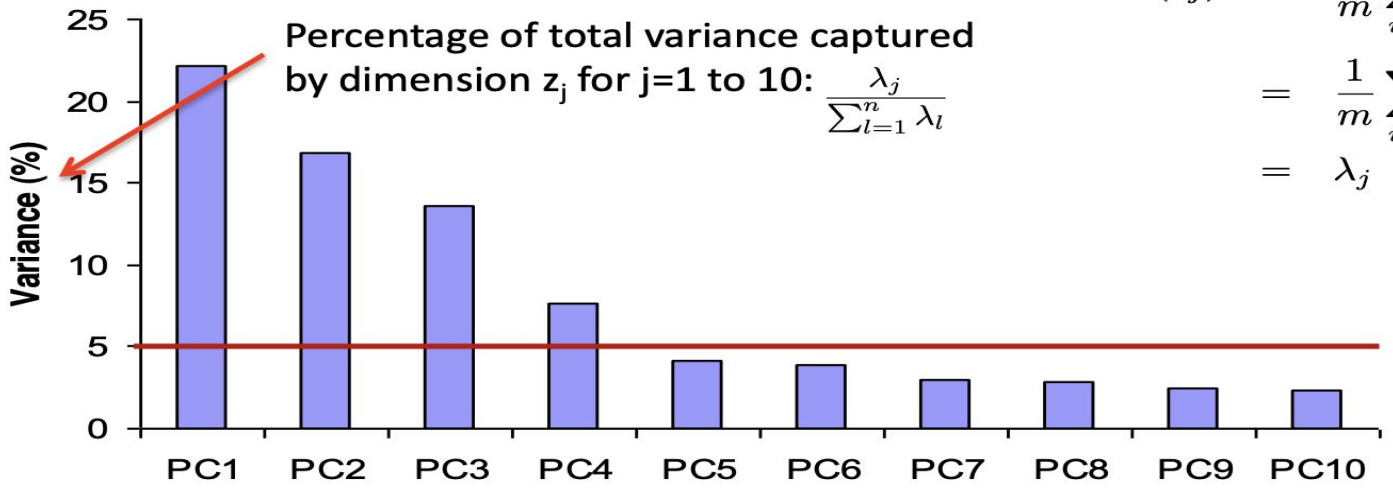
Minimized when $(VV^T) = \mathrm{I}$

Exactly what eigenvectors do

Dimensionality Reduction with PCA

- In high-dimensional problem, data usually lies near a linear subspace, as noise introduces small variability
- Only keep data projections onto principal components with large eigenvalues

Can *ignore* the components of lesser significance.



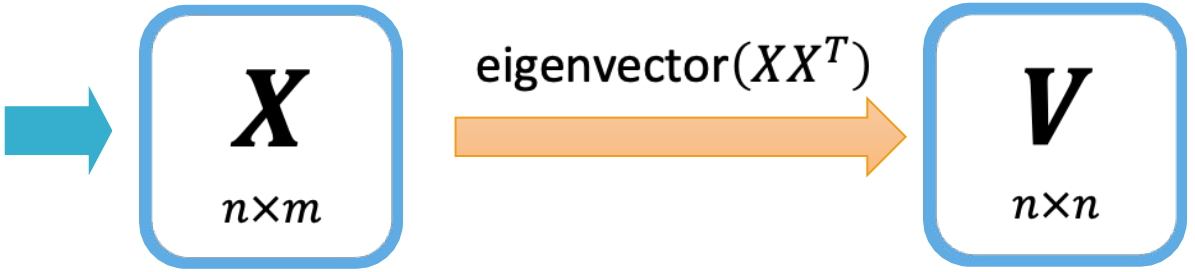
$$\begin{aligned}\mathrm{var}(z_j) &= \frac{1}{m}\sum_{i=1}^{m}(z_j^i)^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}(x^i \cdot u_j)^2 \\ &= \lambda_j\end{aligned}$$

You might lose some information, but if the eigenvalues are small, you don't lose much

PCA Applications: EigenFaces for Facial Recognition

Want to identify specific person, based on facial image

- Robust to glasses, lighting,…
- Can't use all 256x256 pixels.



n = # pixels/image

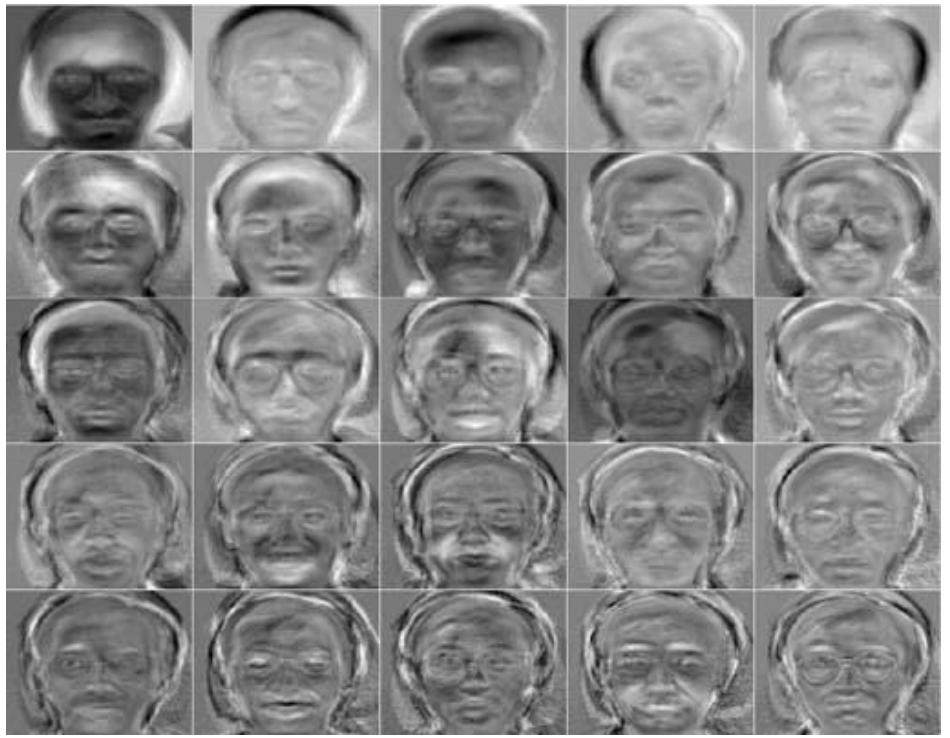
m = # images

Yale Faces database is 320x243 grayscale
15 people, 11 pictures per person

PCA Applications: EigenFaces

Faces

EigenFaces

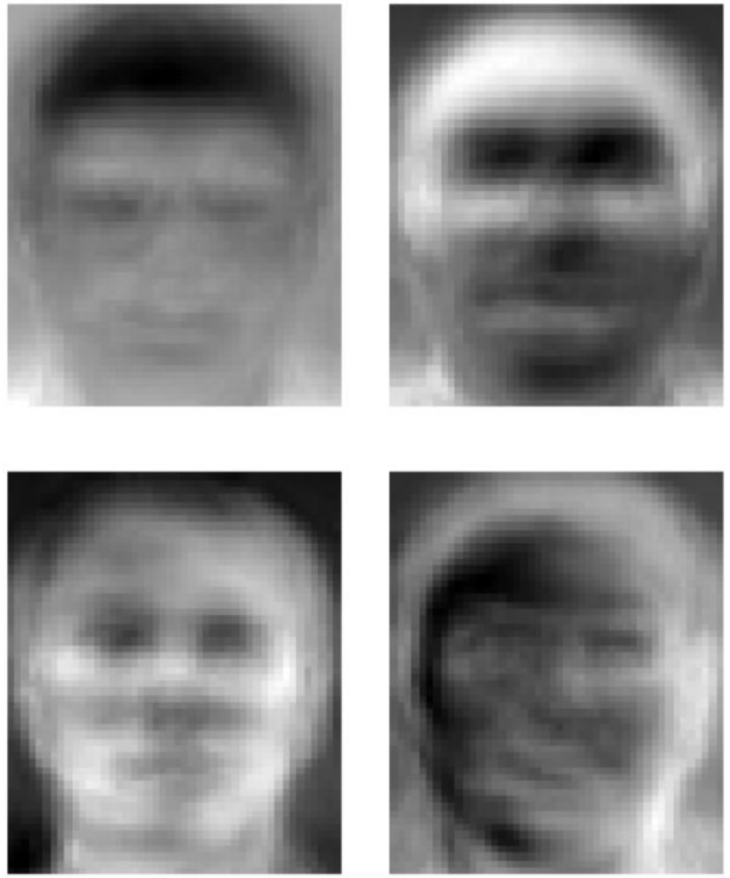


Eigenfaces are the eigenvectors of the covariance matrix of the probability distribution of the vector space of human faces

- Eigenfaces are standardized face ingredients
- A human face may be considered to be a combination of these standard faces

PCA applications -Eigenfaces

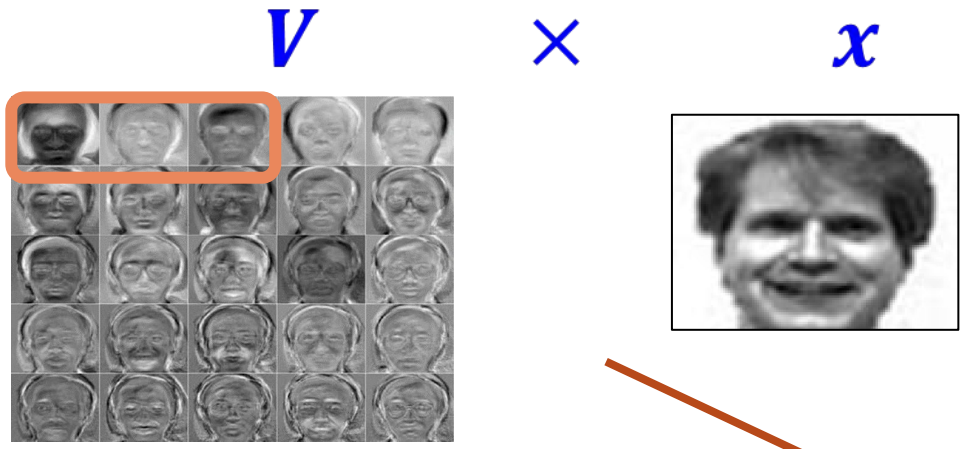
- the principal eigenface looks like a bland androgynous average human face



http://en.wikipedia.org/wiki/Image:Eigenfaces.png

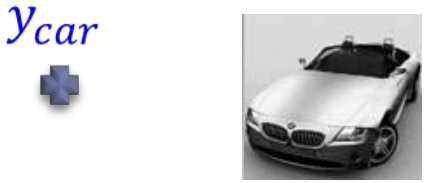
PCA Applications: EigenFaces

Dimensionality Reduction



Outlier Detection

$$\underset{c}{\text{argmin}} \, ||y_{car} - y^c|| < \theta_2$$



Classification

$$\underset{c}{\text{argmin}} \, ||y - y^c|| < \theta_1$$

Other dimensionality reduction methods

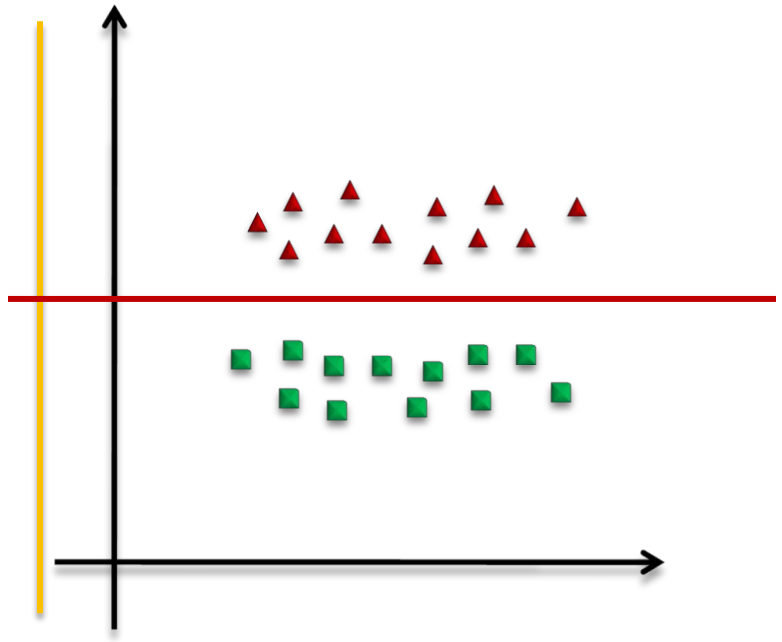
- Kernel PCA
- Independent component analysis
- t-SNE (t-distributed stochastic neighbor embedding)
- Autoencoder
- …

Linear Discriminant Analysis

Supervised Dimensionality Reduction:
Linear Discriminant Analysis

- PCA: Perform dimensionality reduction while preserving as much of the variance in the high dimensional space as possible.
- LDA: Perform dimensionality reduction while preserving as much of the class discriminatory information as possible.

Supervised Dimensionality Reduction



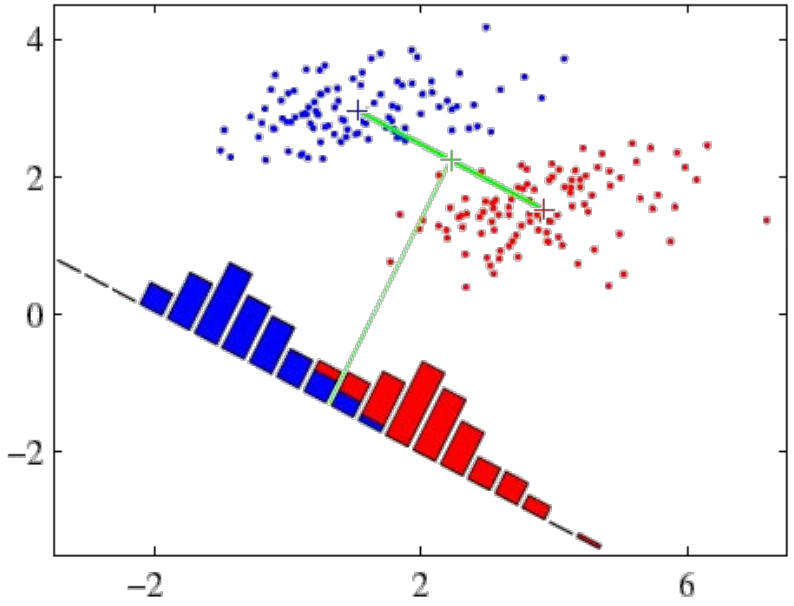
Principal Component Analysis

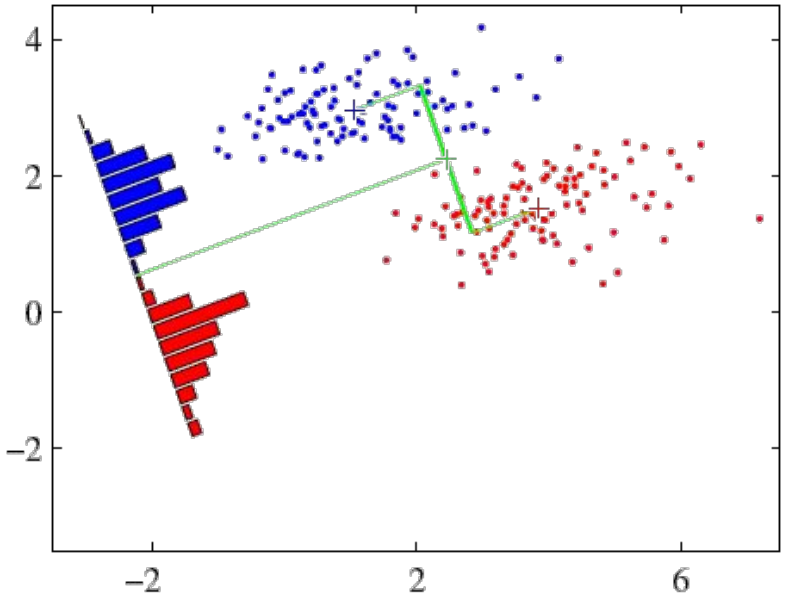
- High variance
- Does not consider discriminability

Fisher Linear Discriminant
Linear Discriminant Analysis

- Does not consider variance
- Good discriminability

Linear Discriminant Analysis





$$y = W^T x$$

- Assumptions for new basis:
  1. Maximize distance between projected class means
  2. Minimize projected class variance

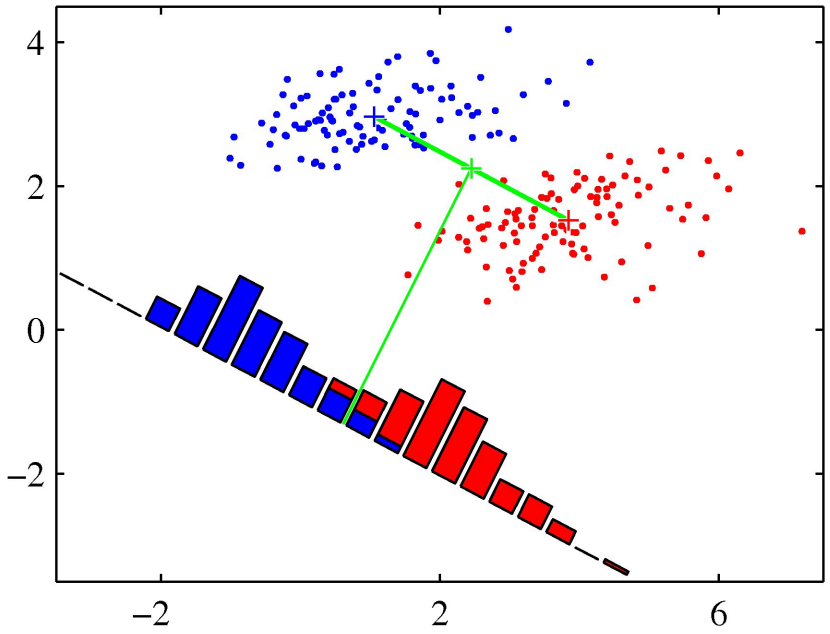
Fisher Linear Discriminant

- Consider the mean vectors of a binary classification problem

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

- Constrain $\|w\|^2 = 1$
- Choosing the distance between the means as the objective function:

$$\mathbf{w} = \arg \max_{\mathbf{w} \in \mathcal{W}} \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$$



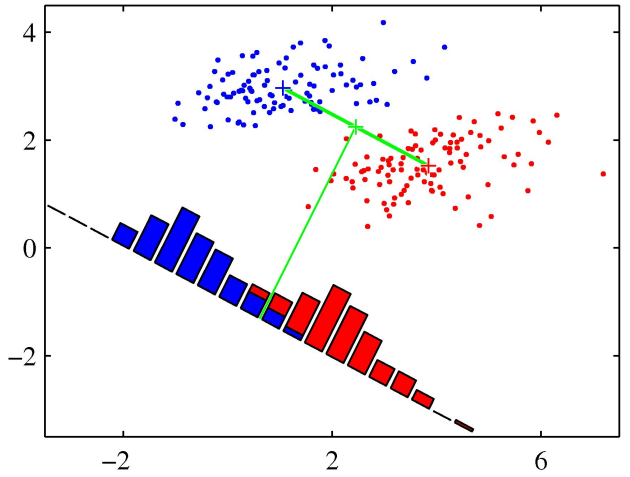
Fisher Linear Discriminant

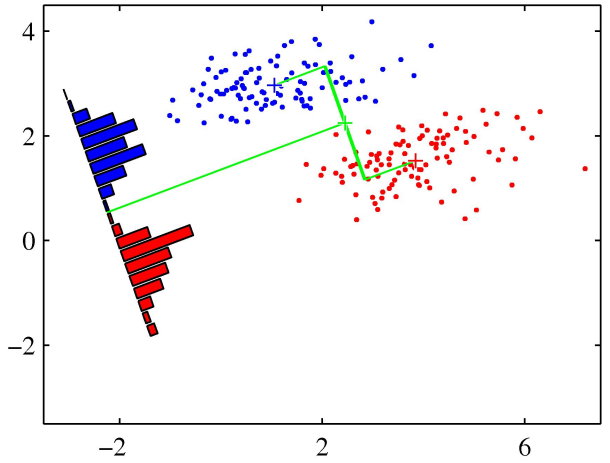
- Fisher: maximize a function that represents the difference between the means, normalized by a measure of within-class variability (scatter).
- Define the scatter for each class (variance)

$$s_k^2 = \sum_{n \in \mathcal{C}_k} \left[\mathbf{w}^T (\mathbf{x}_n - \mathbf{m}_k) \right]^2$$

- Fisher Criterion (2 classes)

$$J(\mathbf{w}) = \frac{\left[\mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1) \right]^2}{s_1^2 + s_2^2}$$





Fisher Linear Discriminant

Is the ratio of between-class variance to the within-class variance where

$$J(\mathbf{w}) = \frac{\left[\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)\right]^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}$$

Within-class variance ("scatter")

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

Between-class covariance ("scatter")

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

Solution: $\boxed{\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)}$

Fisher Linear Discriminant

- Differentiating $J(\mathbf{w})$ w.r.t. $\mathbf{w}$, maximized when

$$(\mathbf{w}^T\mathbf{S}_W\mathbf{w})\mathbf{S}_W\mathbf{w} = (\mathbf{w}^T\mathbf{S}_W\mathbf{w})\mathbf{S}_B\mathbf{w}$$

1. Don't care about $(\mathbf{w}^T\mathbf{S}_B\mathbf{w})$ or $(\mathbf{w}^T\mathbf{S}_W\mathbf{w})$ as there are just scaling factors
2. Also, $\mathbf{S}_B\mathbf{w}$ is always in direction of $(\mathbf{m}_2-\mathbf{m}_1)$

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

A Generalized Eigenvalue Problem