



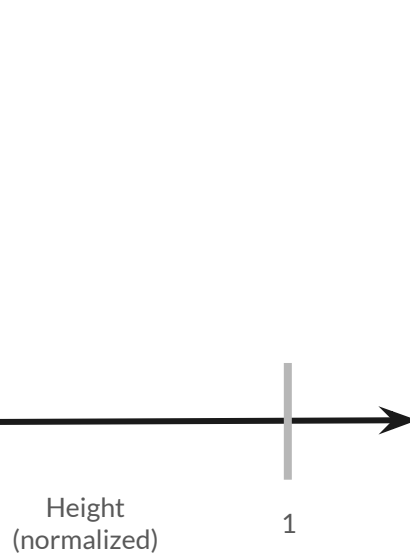
CIS 678 - Machine Learning

- Curse of Dimensionality
- Principal Component Analysis (PCA)



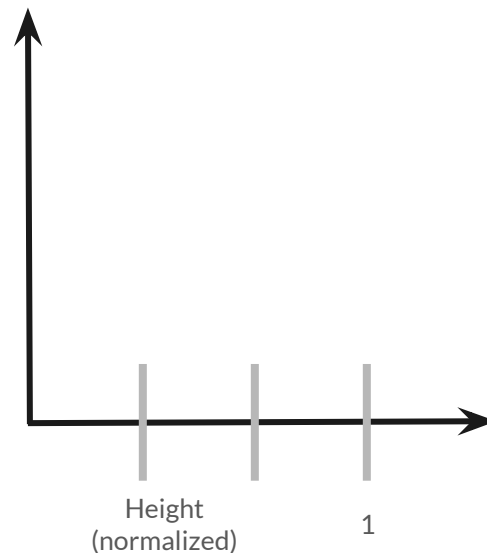
Curse of Dimensionality

- x axis : Heights (min-max normalized: [0-1])



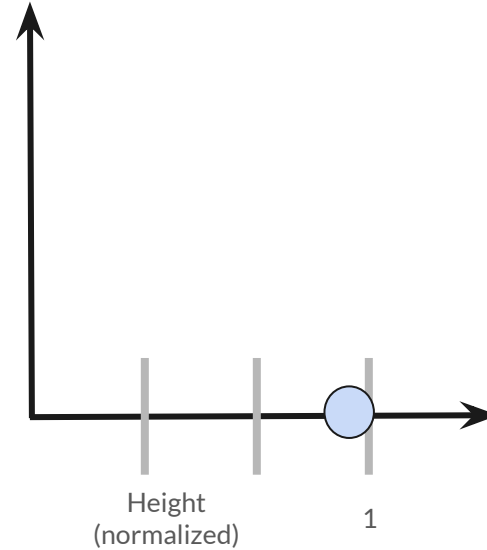
Curse of Dimensionality

- x axis : Heights (min-max normalized: [0-1])
- 3 equal bins (aka numeric to categorical)



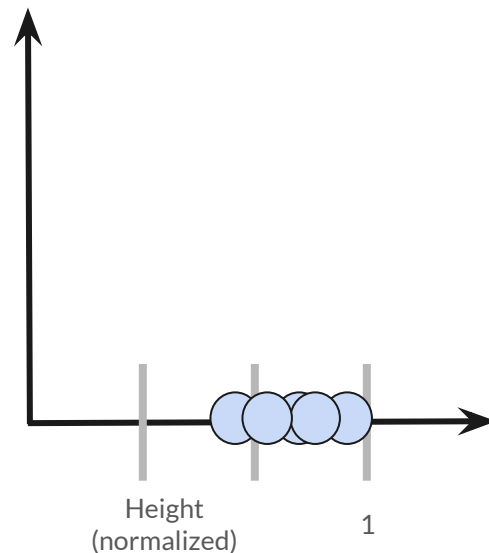
Curse of Dimensionality

- x axis : Heights (min-max normalized: [0-1])
- 3 equal bins (aka numeric to categorical)
- **The tallest person in the class**



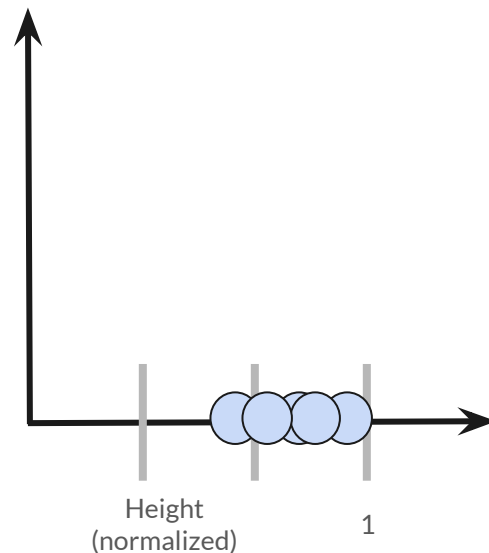
Curse of Dimensionality

- x axis : Heights (min-max normalized: [0-1])
- 3 equal bins (aka numeric to categorical)
- **All of us**



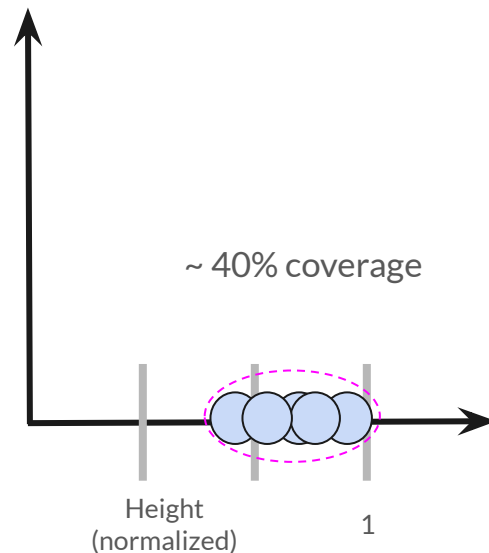
Curse of Dimensionality

- x axis : Heights (min-max normalized: [0-1])
- 3 equal bins (aka numeric to categorical)
- All of us
- I don't think we have someone with 0.5 times height than the tallest



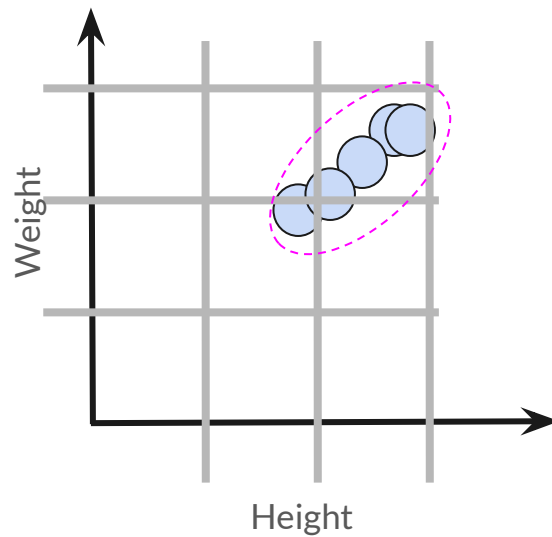
Curse of Dimensionality

- x axis : Heights (min-max normalized: [0-1])
- 3 equal bins (aka numeric to categorical)
- All of us
- I don't think we have someone with 0.5 times height than the tallest
- Is ~40% domain coverage a reasonable assumption?



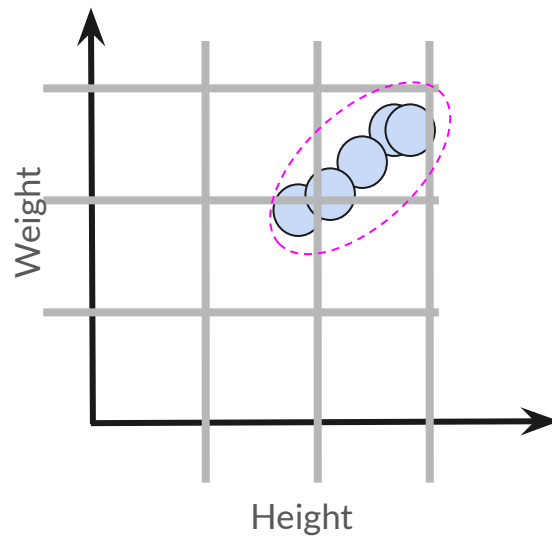
Curse of Dimensionality

- We have added Weight as our second feature dimension
- Do you find the samples represent our class?
- What's the domain coverage now?



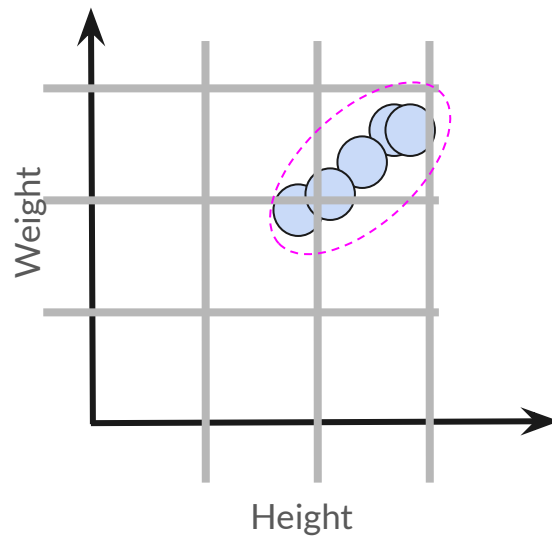
Curse of Dimensionality

- We have added Weight as our second feature dimension
- Do you find the samples represent our class?
- What's the domain coverage now?
- Is ~10% a reasonable assumption?



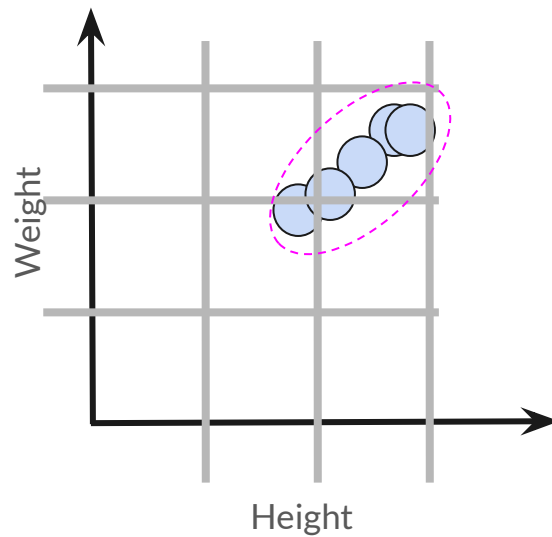
Curse of Dimensionality

- We have added Weight as our second feature dimension
- Do you find the samples represent our class?
- What's the domain coverage now?
- Is ~10% a reasonable assumption?
- **See how numbers dropped from ~40% to ~10%**



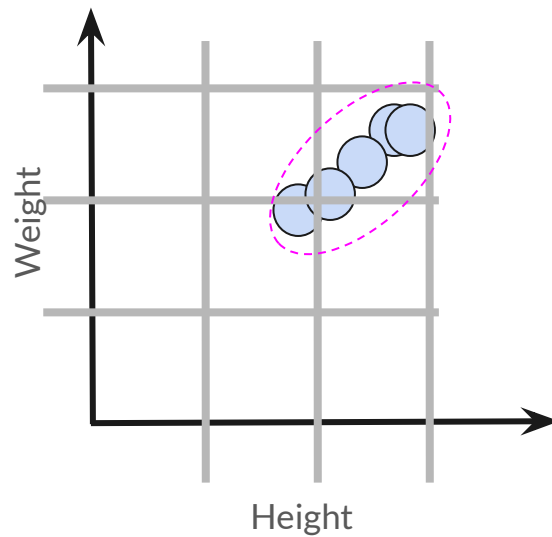
Curse of Dimensionality

- We have added Weight as our second feature dimension
- Do you find the samples represent our class?
- What's the domain coverage now?
- Is ~10% a reasonable assumption?
- **See how numbers dropped from ~40% to ~10%**
- **The other way, the empty space grew up from ~60% to ~90%**



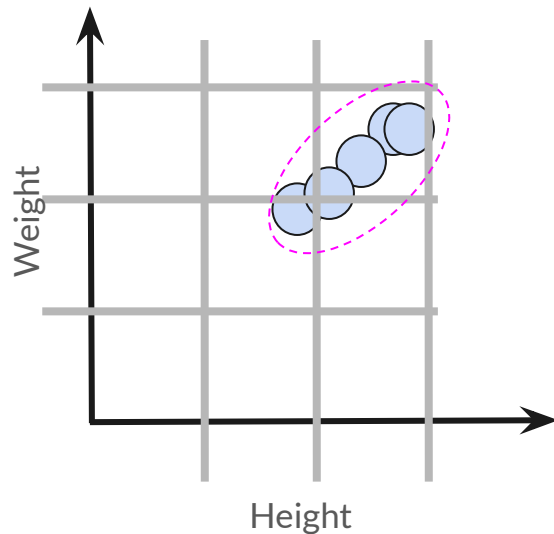
Curse of Dimensionality

- We have added Weight as our second feature dimension
- Do you find the samples represent our class?
- What's the domain coverage now?
- Is ~10% a reasonable assumption?
- See how numbers dropped from ~40% to ~10%
- The other way, the empty space grew up from ~60% to ~90%
- What's the number would look like if we add **2** more dimensions?



Curse of Dimensionality

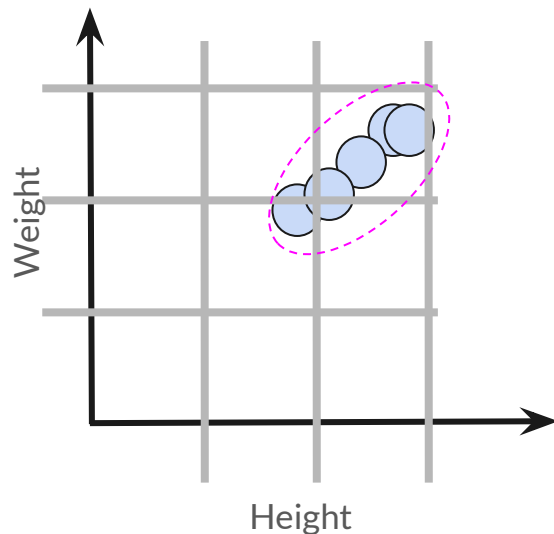
- We have added Weight as our second feature dimension
- Do you find the samples represent our class?
- What's the domain coverage now?
- Is ~10% a reasonable assumption?
- See how numbers dropped from ~40% to ~10%
- The other way, the empty space grew up from ~60% to ~90%
- What's the number would look like if we add **2** more dimensions?



- Empty space grows exponentially with the increase in adding new features.

Curse of Dimensionality

- We have added Weight as our second feature dimension
- Do you find the samples represent our class?
- What's the domain coverage now?
- Is ~10% a reasonable assumption?
- See how numbers dropped from ~40% to ~10%
- The other way, the empty space grew up from ~60% to ~90%
- **What's the number would look like if we add 2 more dimensions?**



- Empty space grows exponentially with the increase in adding new features.
- **Data distribution becomes sparse, and difficult to learn a good model.**



Dimensionality Reduction

General idea

- You have some data of feature dimension size, $|D|$



Dimensionality Reduction

General idea

- You have some data of feature dimension size, $|D|$
- You want to compress them to of size, $|d|$
- $|d| < |D|$



Dimensionality Reduction

- Linear
 - Principal Component Analysis (PCA)
 - SVD
- Neural Networks
 - Auto Encoders
 - RBMs



Covariance

- Variance relationship between a pair of variables

$$cov_{x,y} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{N - 1}$$



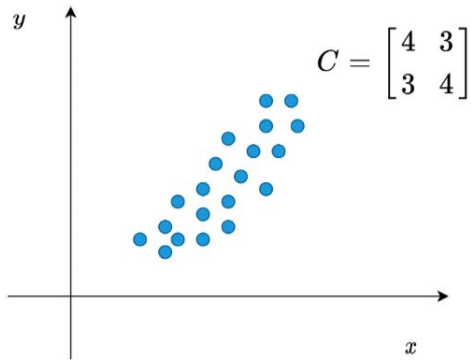
Covariance

- Variance relationship between a pair of variables
- It's a symmetric matrix; right?

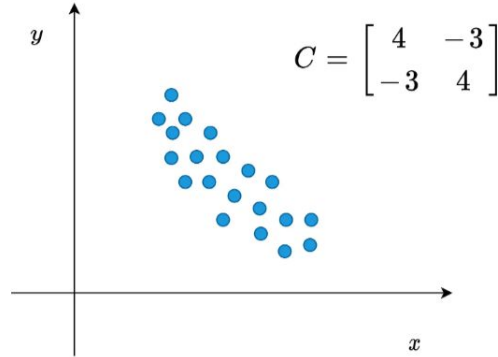
$$cov_{x,y} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{N - 1}$$



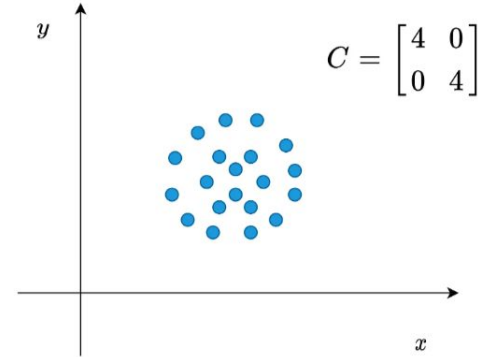
Positive
Covariance



Negative
Covariance



Zero
Covariance

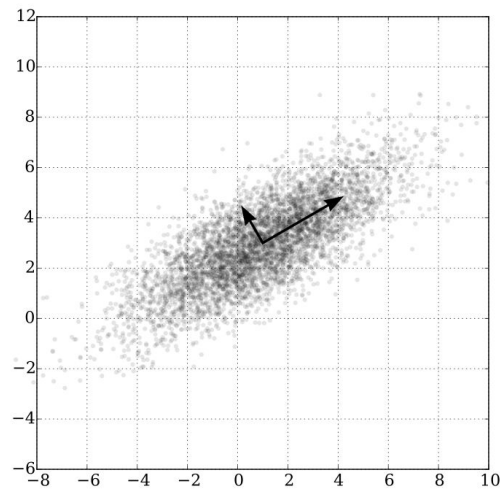




Principal Component Analysis (PCA)

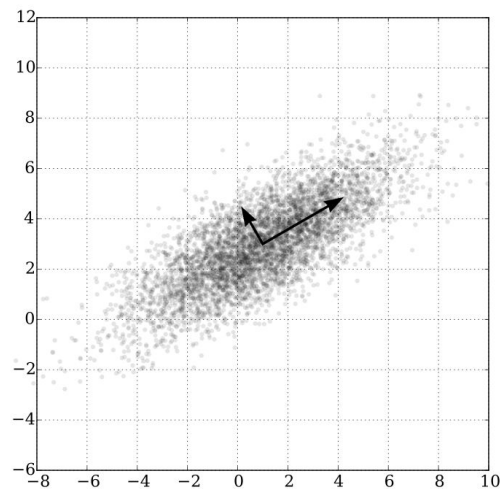
General idea: 2D Gaussian example

- Features x and y shows some relationships



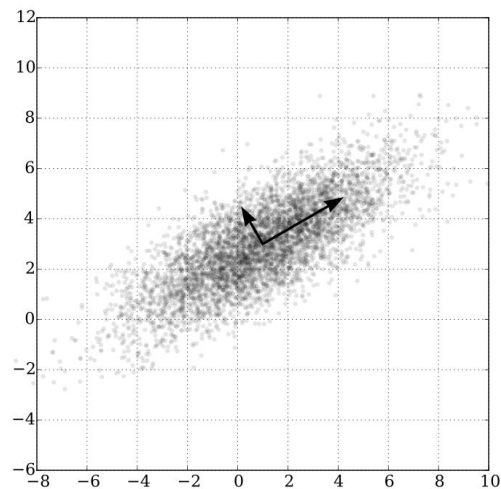
General idea: 2D Gaussian example

- Features x and y shows some relationships
- This 2D Gaussian has its own coordinates (off the reference cartesian coordinates x' and y' ; right?)



General idea: 2D Gaussian example

- Features x and y shows some relationships
- This 2D Gaussian has its own coordinates (off the reference cartesian coordinates x' and y' ; right?)
- **The principal components**





How to ..



PCA steps

1. Let assume $D_{m \times n}$ is a given data matrix



PCA steps

1. Let assume $D_{m \times n}$ is a given data matrix
2. Apply standard scalar (normalization)



PCA steps

1. Let assume $D_{m \times n}$ is a given data matrix
2. Apply standard scalar (normalization)
3. Estimate Covariance (matrix), $A_{n \times n}$



PCA steps

1. Let assume $D_{m \times n}$ is a given data matrix
2. Apply standard scalar (normalization)
3. Estimate Covariance (matrix), $A_{n \times n}$



PCA steps

1. Let assume $D_{m \times n}$ is a given data matrix
2. Apply standard scalar (normalization)
3. Estimate Covariance (matrix), $A_{n \times n}$
4. Compute Eigenvalues and Eigenvectors of the Covariance Matrix by solving the linear dynamical system at the right

If A is a $n \times n$ matrix, solving this linear dynamical system will give n **eigenvalues**, and n associated n **eigenvectors**

$$AX = \lambda X$$

$$AX - \lambda X = 0$$

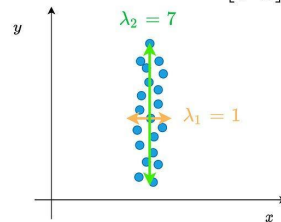
or

$$(A - \lambda I) X = 0$$

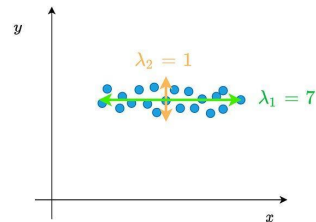
PCA

Notebook presentation

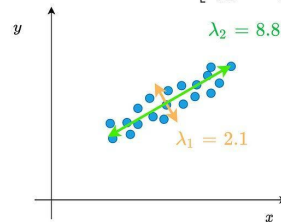
1 $C = \begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix}$ $\lambda_{1,2} = [1 \ 7]$
 $V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



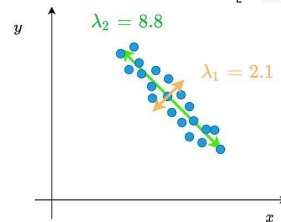
2 $C = \begin{bmatrix} 7 & 0 \\ 0 & 1 \end{bmatrix}$ $\lambda_{1,2} = [7 \ 1]$
 $V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



3 $C = \begin{bmatrix} 4 & 3 \\ 3 & 7 \end{bmatrix}$ $\lambda_{1,2} = [2.1 \ 8.8]$
 $V = \begin{bmatrix} -0.8 & -0.5 \\ 0.5 & -0.8 \end{bmatrix}$



4 $C = \begin{bmatrix} 4 & -3 \\ -3 & 7 \end{bmatrix}$ $\lambda_{1,2} = [2.1 \ 8.8]$
 $V = \begin{bmatrix} -0.8 & 0.5 \\ -0.5 & -0.8 \end{bmatrix}$



λ = eigenvalues

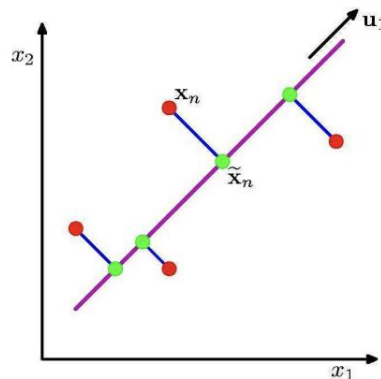
V = eigenvectors

What are we optimizing?

- 2D example (red points)
- Green points are 1D projections/transformations
- We are reducing data definitions from 2D to 1D; this can be generalized from D to M dimensions

Two techniques (in general):

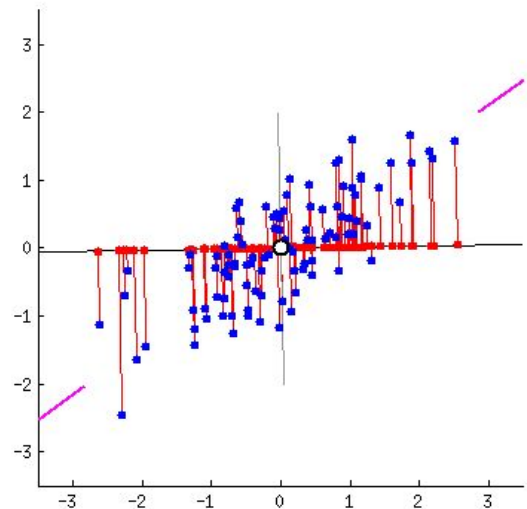
- Maximize variance
- Minimize errors (distance between each green-red pair)



What are we optimizing?

Two techniques (in general):

- Maximize variance
- Minimize errors (distance between each green-red paris)



ref

What are we optimizing?

Standard PCA: Variance maximization

- One dimensional example
- Objective: maximize projected variance w.r.t. \mathbf{u}_1

$$\frac{1}{N} \sum_{n=1}^N \{\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}}\}^2 = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$$

- where sample mean and data covariance are:

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$

- Must constrain $\|\mathbf{u}_1\|$: via Lagrange multiplier, maximize w.r.t \mathbf{u}_1

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda(1 - \mathbf{u}_1^T \mathbf{u}_1)$$

- Optimal \mathbf{u}_1 is principal component (eigenvector with maximal eigenvalue)

