CSCE 633: Machine Learning

Lecture 33: Dimensionality Reduction-Principal Component Analysis

Texas A&M University

Bobak Mortazavi

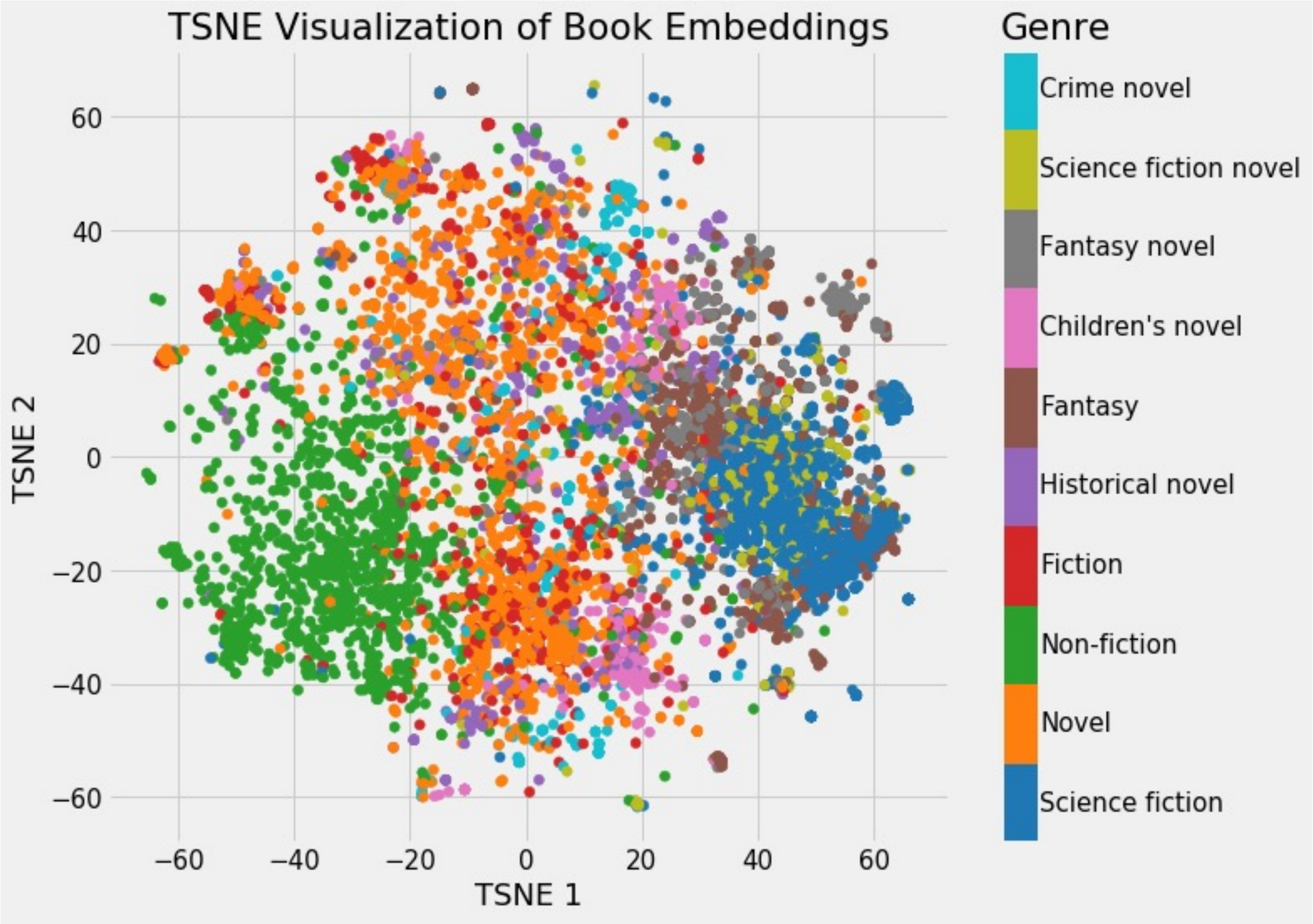
Zhale Nowroozi

Review

- Neural Network Modeling
  - MLP
  - CNN
  - RNN
  - Transformers

Review

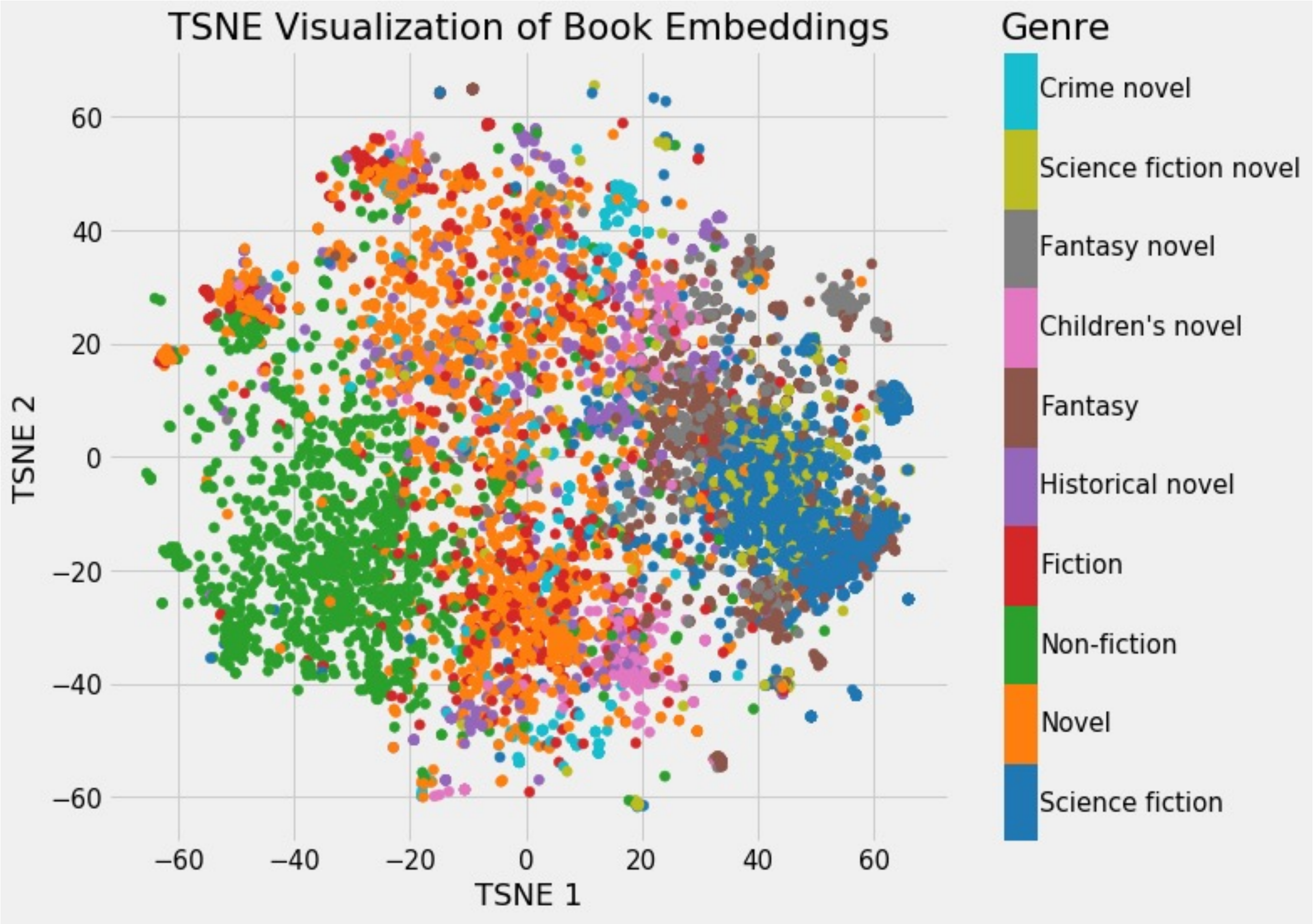
- Neural Network Modeling
  - MLP
  - CNN
  - RNN
  - Transformers
- Transformers in NLP
  - Word Embedding
  - https://projector.tensorflow.org/



Source: https://devopedia.org/word-embedding

Review

- Neural Network Modeling
  - MLP
  - CNN
  - RNN
  - Transformers
- Transformers in NLP
  - Word Embedding
  - https://projector.tensorflow.org/
- Supervised vs unsupervised learning



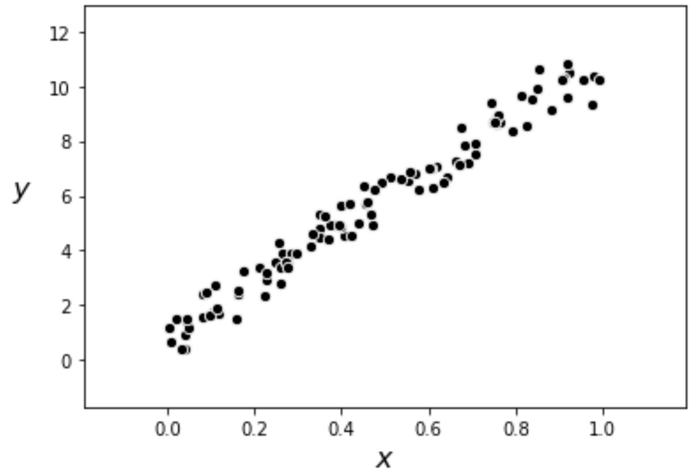
Source: https://devopedia.org/word-embedding

Goals

- Understanding feature scaling and dimension reduction
- Standard normalization
- Principal component analysis

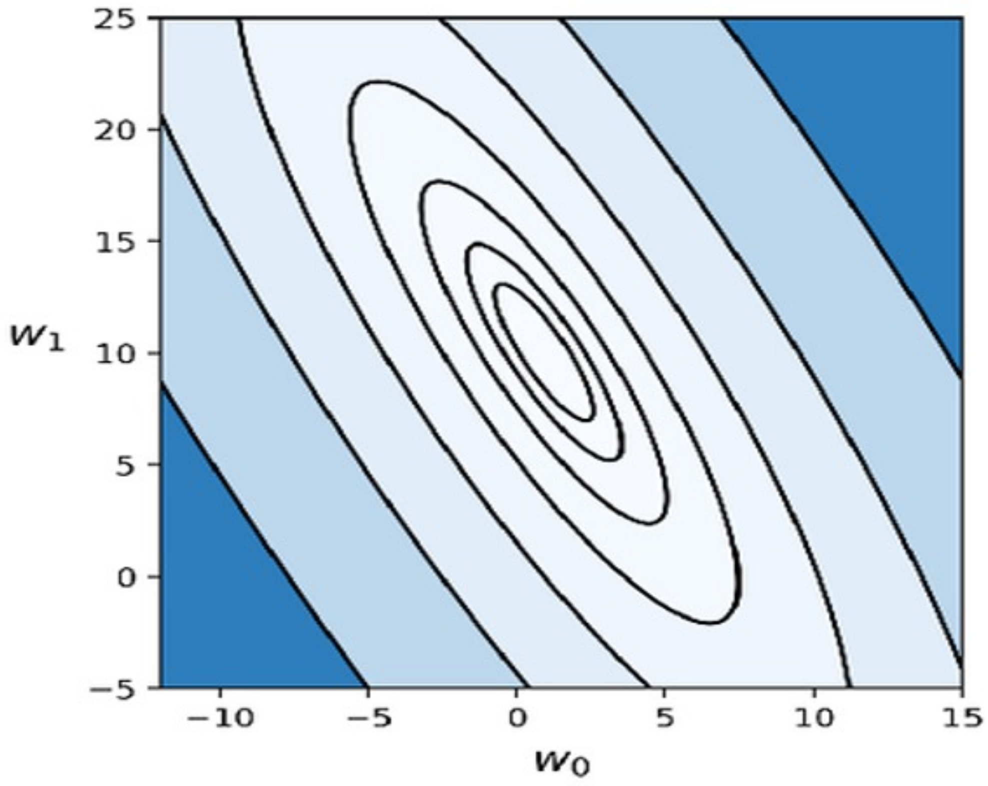
Standard Normalization of Data

- Let's assume we want to perform regression on data as in this figure
- It appears the data is roughly on a line
- Note however that the scales of x and y differ significantly



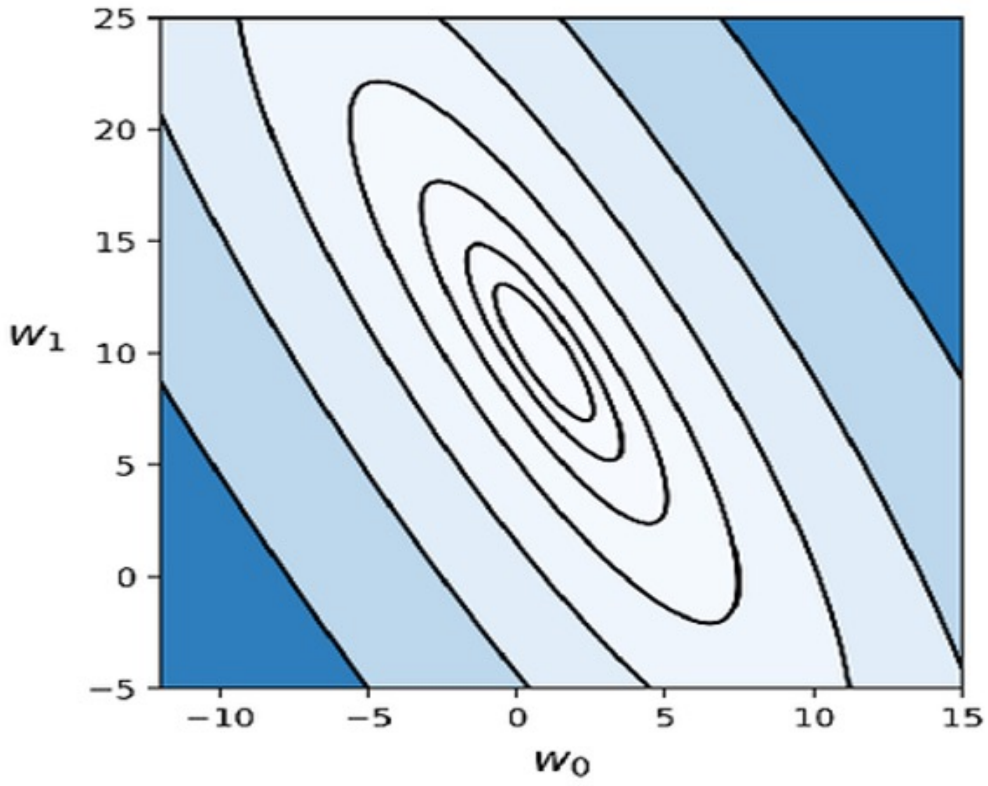
How does this impact modeling?

- If we plot the least squares cost function we see that
- The contours are elliptical
- There is a long narrow value along the axis of the ellipses



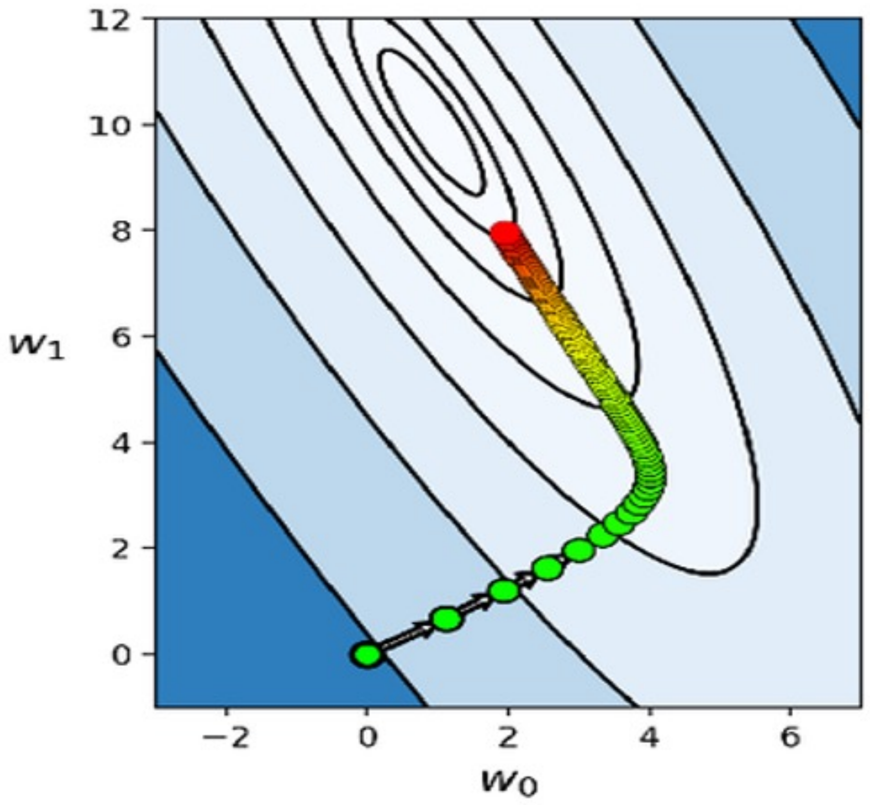
How does this impact modeling?

- If we plot the least squares cost function we see that
- The contours are elliptical
- There is a long narrow value along the axis of the ellipses
- An important question to ask:
  - How does this impact gradient descent optimization?



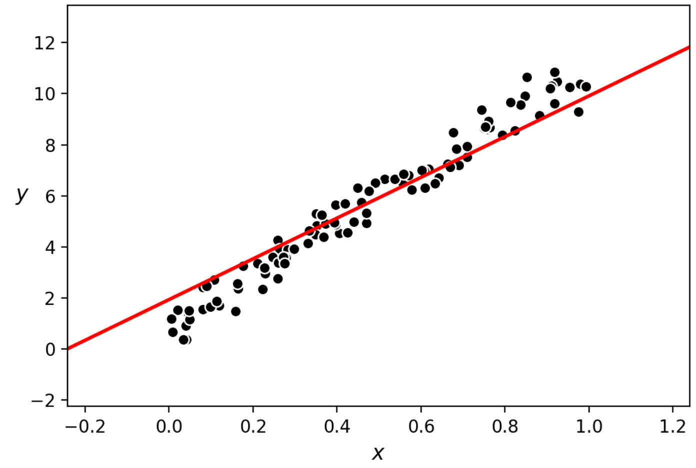
How does this impact modeling?

- If we plot the least squares cost function we see that
- The contours are elliptical
- There is a long narrow value along the axis of the ellipses
- An important question to ask:
  - How does this impact gradient descent optimization?
- They make gradient descent slow
- Unless we get lucky to initialize the algorithm along the short axis it takes gradient descent time to find that axis



Standard Normalization of Data

- Taking that solution we plot the line of best fit and see that we have a poor solution
- So, we will seek to change our shape of our least squares cost function to pave the way for more accurate and quicker gradient descent
- The adjustment we make: standard normalization



TEXAS A&M UNIVERSITY

Centering and Scaling Data

- Assume for every vector x, which has p dimensions, we can scale feature p by:

$$x_p = \frac{x_p - \mu_p}{\sigma_p}$$

- Where

$$\mu_p = \frac{1}{N}\sum_{i=1}^{N} x_{pi}$$

which is the mean of the pth feature across all N subjects in the data

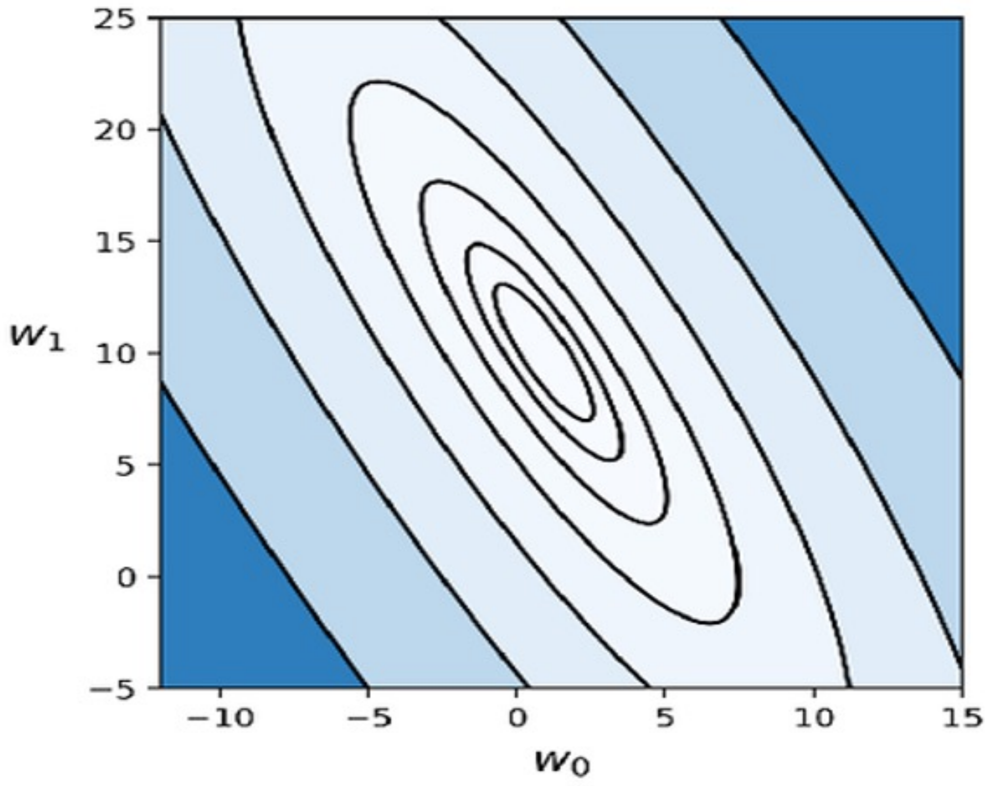
And

$$\sigma_p = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{pi} - \mu_p)^2}$$

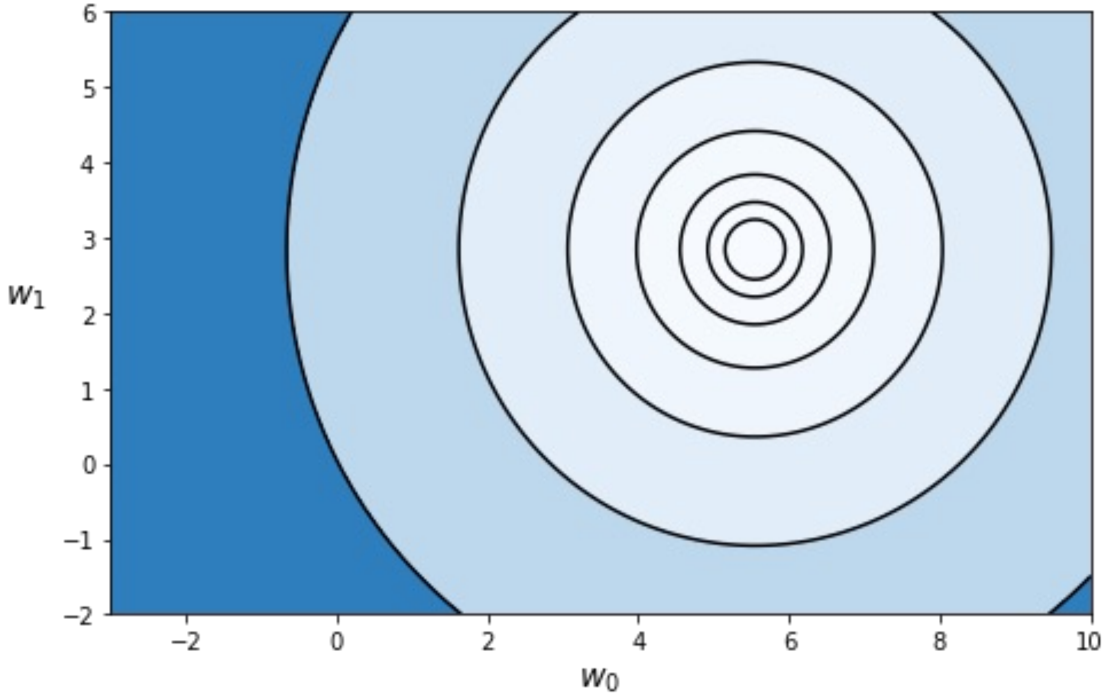
This is often called standard normalization – 0 mean and unit standard deviation for all data elements.

Revisiting our Contours

Without feature normalization

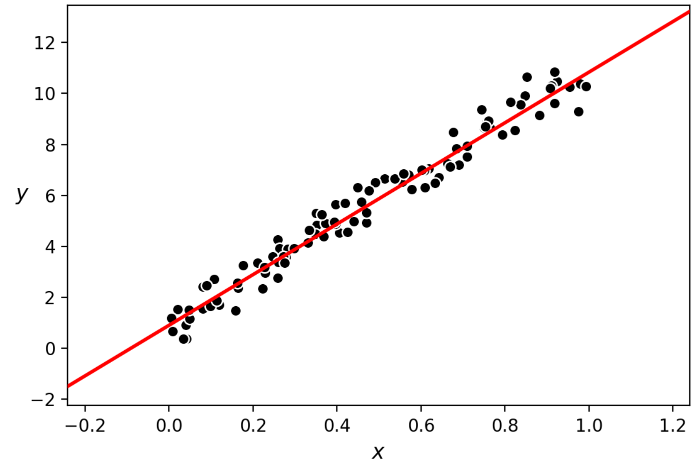


With feature normalization



Standard Normalization of Data

- With only 5 steps of gradient descent we get a much better fit line
- We have to be careful if our denominator is 0 for any specific feature
- Can apply this across all features x



Principal Component Analysis

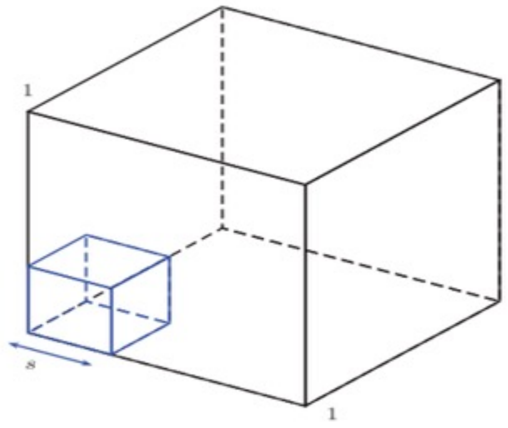
- Changing features and standardizing them has great impact on the loss landscape and optimization
- We can extend this further using a technique called Principal Component Analysis
- In short, PCA will:
  - Rotate mean-centered data
  - Align along the largest orthogonal directions for features before applying standardized scaling
  - Results in more circular contours
  - Simplifies optimization through enabling feature selection – dimension reduction

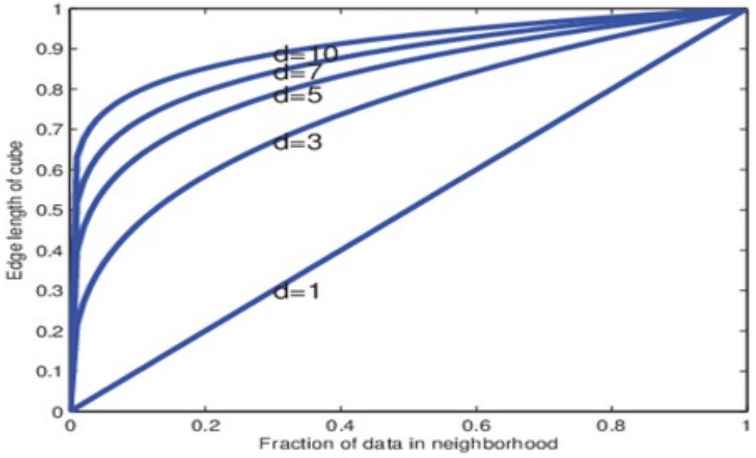
Dimensionality Reduction: Why does it matter?

- Broad Question
  - How can we detect low dimensional structure in high dimensional data
- Motivation
  - Exploratory data analysis: You can easily plot and visualize low dimensional data
  - Compact representation: takes less space
  - Robust statistical modeling: counters the curse of dimensionality

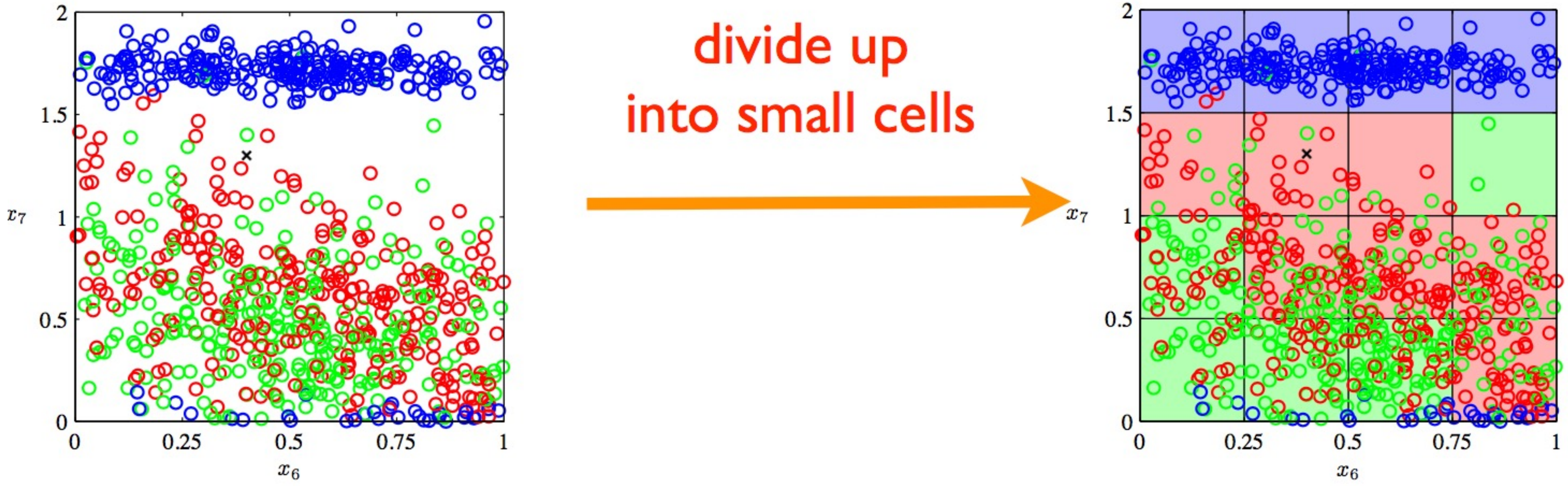
Curse of Dimensionality

- In high dimensional data
  - Intuition tends to fail in higher dimensions about similarity
  - Problems become harder to generalize
  - Harder to systemically search
- On the positive side
  - Blessing of non-uniformity – examples tend not to be uniformly distributed in high dimensions

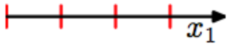




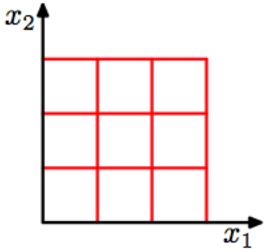
What is the curse of dimensionality: Example with trees



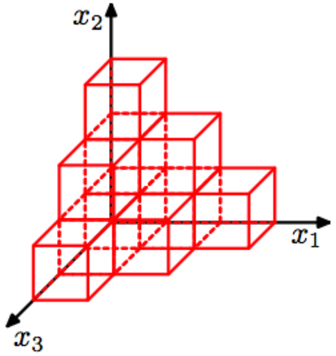
Curse of Dimensionality: Number of Cells to divide



$D = 1$



$D = 2$



$D = 3$

# of cells $\quad r^D$

r: number of divisions in each dimension

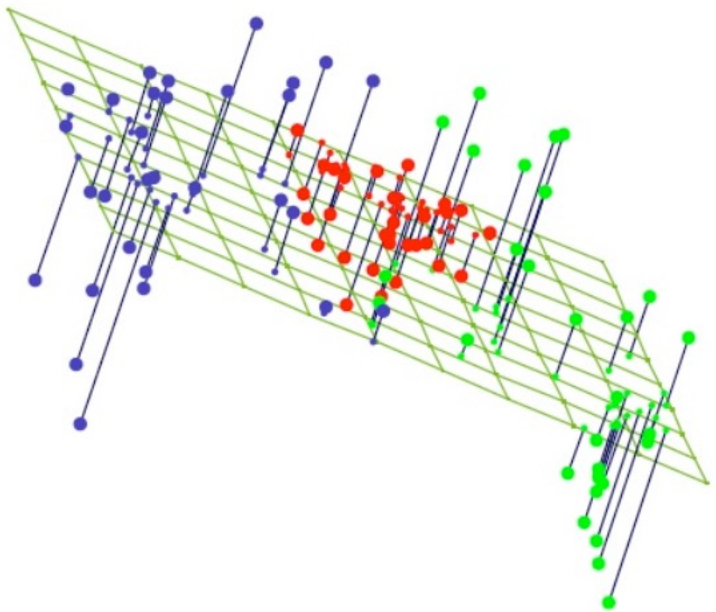
Large number of cells, even if D is only moderately large

So to cover the whole space reasonably well you need exponentially larger number of training data points

Linear Dimensionality Reduction

To go from X in P dimensional space to Z in M dimensional space where P >> M
Create a linear transformation of

$$z = U^T x$$



Transforming Predictors

- Methods thus far have selected from all predictors x for all dimensions i = 1, … , P
- What if we transform them to z for i = 1, …, M that represent M < P linear combinations of our original P predictors
- That is

$$Z_m = \sum_{j=1}^{P} \phi_{jm} X_j$$

For constants ϕ_{jm}, m = 1, …, M

Fitting Regression

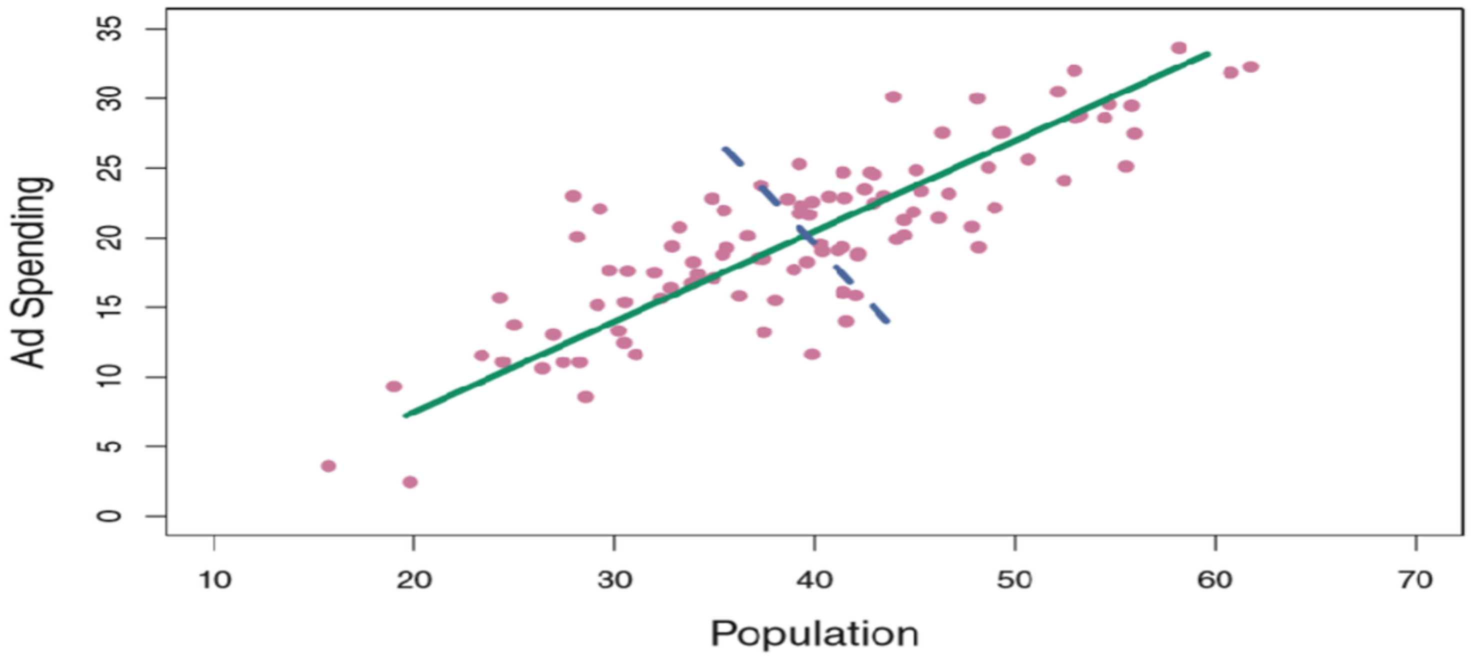
- We can use this to fit

$$y_i = w_0 + \sum_{m=1}^{M} w_m Z_{im} + \epsilon_i$$

For all subjects i from 1 to N.

This results in constants w that are the same linear regression but constrains them in M dimensions instead of P.

Visualizing PCA



PCA: Populations and Ads

- Assume you had a fit regression of the form

$$Z_1 = 0.839 * (pop - \overline{pop}) + 0.544 * (ad - \overline{ad})$$

- We can call 0.839 and 0.544 the principal component loadings – they provide the size and direction of the transformation on each particular axis
- What we want to do is maximize the

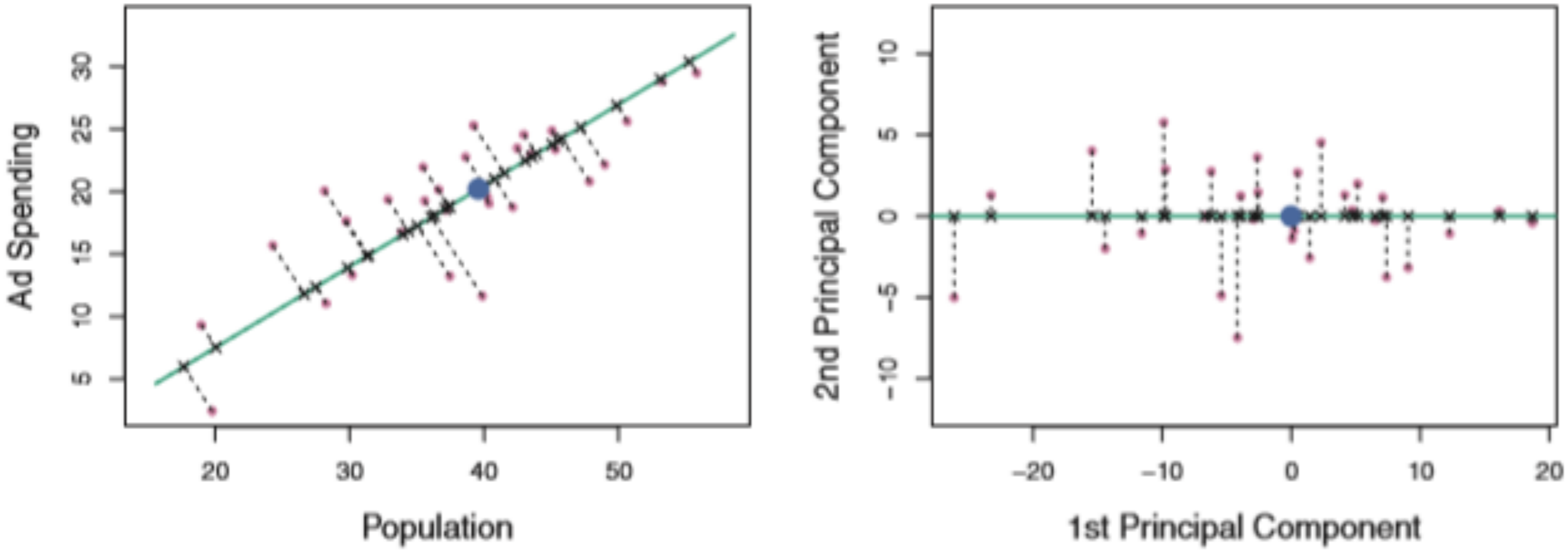
Var $(\phi_1 * (pop - \overline{pop}) + \phi_2 * (ad - \overline{ad})$

So each element has

$$z_{1i} = 0.839 * (pop_i - \overline{pop}) + 0.544 * (ad_i - \overline{ad})$$

And so the z_{1i} and on up to M are the principal component scores

PCA Transformation



PCA Challenges in Unsupervised Learning

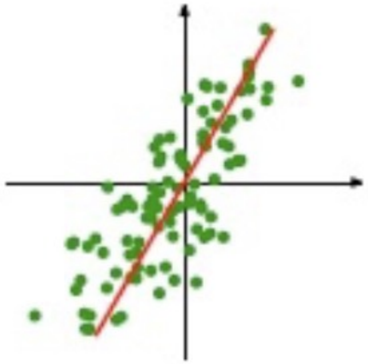
- How do you learn these loadings without supervised labels?
- No simple goal to compare against
- Exploratory data analysis means there is no way to check if your answer is correct
- Infeasible to do a bunch of 2D scatter plots across all dimensions p

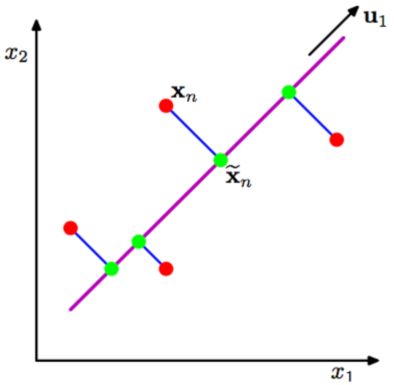
PCA: Representation

Input: D = $\{x_i\}_{i=1}^{N}, where\ x_i \in\ \mathbb{R}^p$
Output: Projected data $\{z_i\}_{i=1}^{N}\ where\ z_i \in\ \mathbb{R}^M$ and M ≪ p
Projection into Subspace: $U \in\ \mathbb{R}^{pXM}$

$$z_i =\ U^T x_i$$
$$U^T U = I$$

Evaluation Metric: Many possible metrics can yield the same solution, Maximize the captured variance, minimize the projection error, minimize reconstruction error, etc.





How do we do this: Matrix Diagonalization

- Converting a square matrix into a special type of matrix (i.e. diagonal), which shares the same fundamental properties of an underlying matrix
- Find a square matrix A of dimension D, that can be decomposed into

$$A = P\,\Lambda\,P^{-1}$$

- Where

$$\Lambda = diag(\lambda_1, \ldots, \lambda_D)$$

The eigenvalues of A

And

$$P = [e_1, \ldots, e_D]$$

Where each e is an eigenvector of A

PCA Optimization of an X

- First, create the covariance matrix (which is a square matrix) from X

$$S = \frac{1}{N} X^T X = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T$$

- Then, diagonalize S (ie compute the eigenvalues and eigenvectors)

$$S = P \, \Lambda \, P^{-1}$$

- Finally, choose the eigenvectors corresponding to some M largest eigenvalues

$$U = [e_1, \dots, e_M] \in \mathbb{R}^{pXM}$$

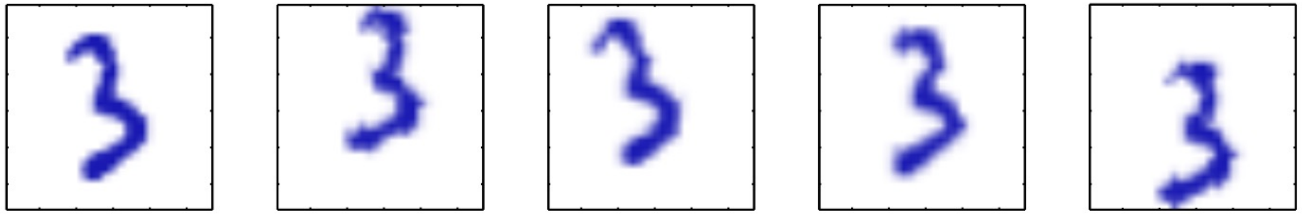
PCA Algorithm

- Step 0: Mean normalize the input features
- Step 1: Compute the covariance matrix S from input matrix X
- Step 2: Diagonalize S and find the eigenvector matrix P
- Step 3: Chose the first M << P eigenvectors or principal components (corresponding to the M largest eigenvalues) and form reduced matrix U
- Step 4: Project data onto the reduced space

$$z_i = U^T x_i$$

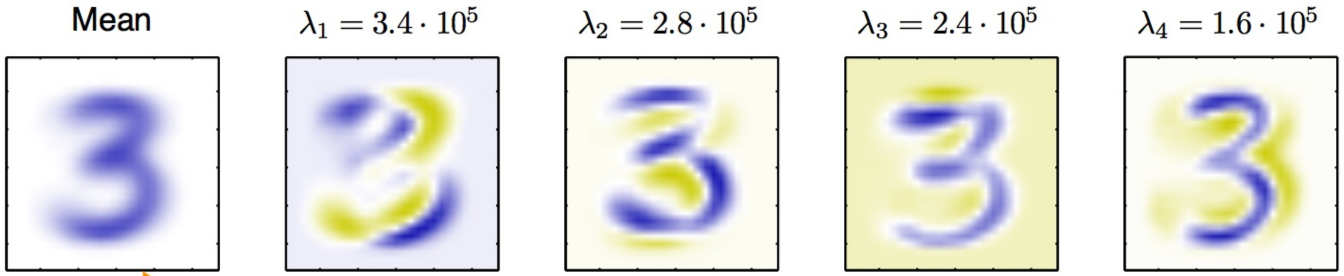
PCA Visualization

Original Images



Eigenvectors

they look like blurred original images



Used to centralize inputs

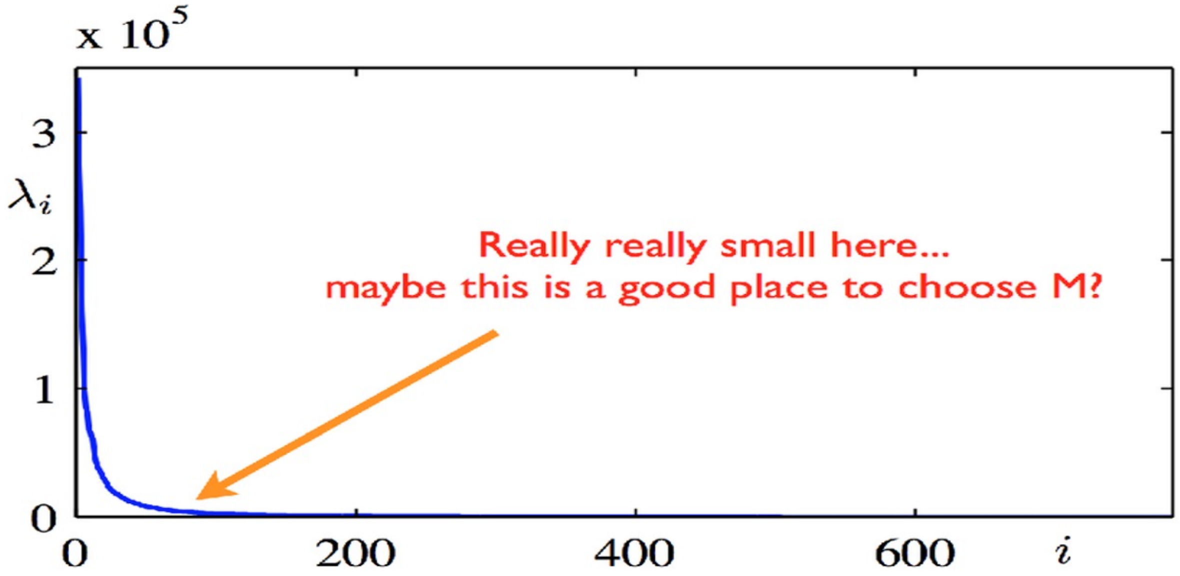
Choosing the right number of components

Plot the eigenspectrum (magnitude of eigenvalues over number of eigenvectors)

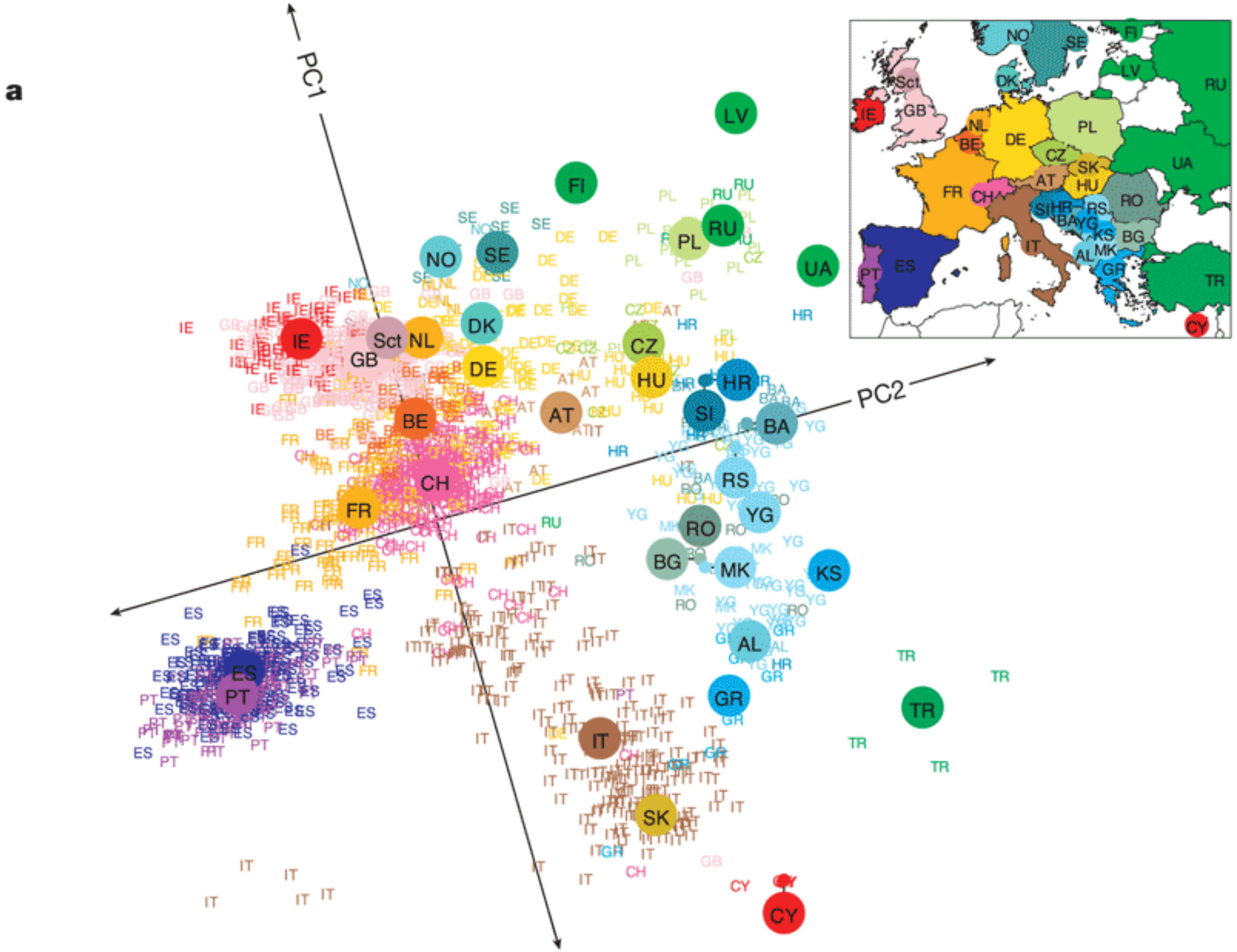
Choose a threshold such as

$$\frac{\sum_{j=1}^{M} \lambda_j}{\sum_{j=1}^{P} \lambda_j} \geq \tau$$

Where it is common to chose 95%, 99% etc.



Applications: Clustering
Finding patterns and structure in unstructured data



Unsupervised learning

- Find patterns/structure/sub-populations in data “knowledge discovery”
- Training data does not contain outputs
- Less well-defined problem with no obvious error metrics
- Examples: topic modeling, market segmentation, handwritten digits, news stories, etc.

K-Means Clustering

- Input D = $\{x_i\}_{i=1}^{N}, where\ x_i \in \ \mathbb{R}^p$
- Output: Clusters $\mu_1, \dots, \mu_K$
- Decision: Define cluster membership, provide a cluster id assigned to each sample x

$$A(x_i) \ \in \ \{1, \dots, K\}$$

- Evaluation Metric: Distortion Measure

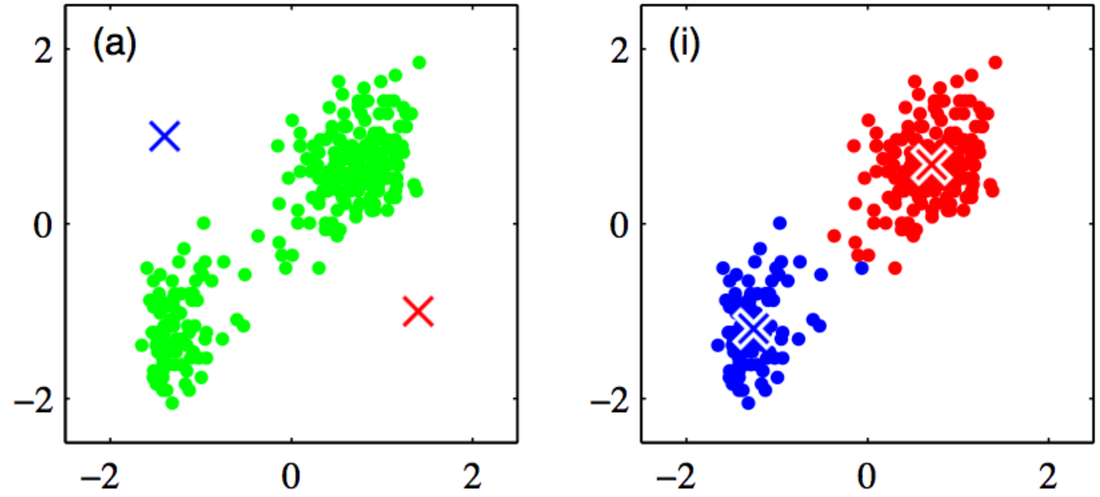
$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_i \ - \ \mu_k\|_2^2$$

Where $r_{nk} = 1$ if $A(x_i) = k$

- Intuition: Data points get assigned to cluster k if they are close to centroid μ_k

K-Means Clustering

- Input D = $\{x_i\}_{i=1}^{N}, where\ x_i \in \ \mathbb{R}^p$
- Output: Clusters $\mu_1, \ldots, \mu_K$
- Decision: Define cluster membership, provide a cluster id assigned to each sample x



- Evaluation Metric: Distor

Where $r_{nk} = 1$ if $A(x_i) =$

- Intuition: Data points get assigned to cluster k if they are close to centroid μ_k

K – means Algorithm

- Optimization Function

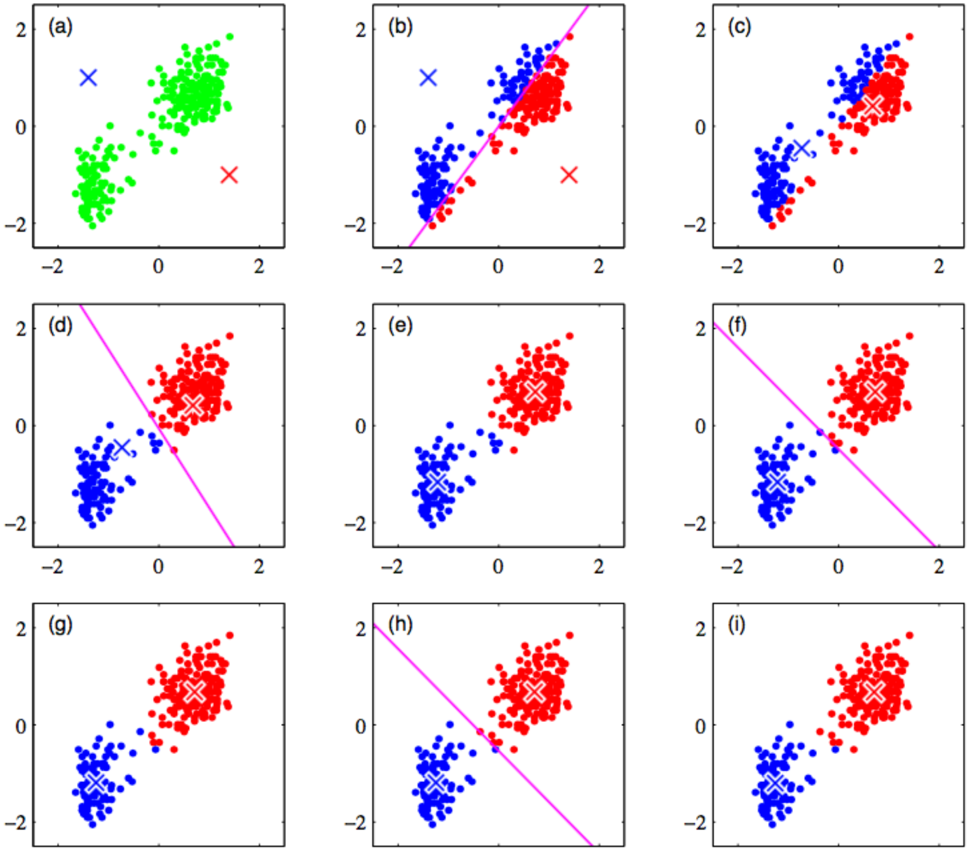
$$\min_{r_{nk}} J = \min_{r_{nk}} \sum_{i=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_i - \mu_k\|_2^2$$

- Step 0: Initialize μ_k to some random values
- Step 1: Assume the current μ_k is fixed, minimize J over r_{nk} which leads to cluster assignment
- Step 2: Assume the fixed cluster assignment, update the cluster centroids as in

$$\mu_k = \frac{\sum r_{nk} \, x_n}{\sum r_{nk}}$$

- Step 3: decide to stop or return to step 1

Visualization of K-Means



Best Performance: PCA and Then Cluster

- PCA! Know how to use libraries to reduce dimensions
- K-Means – know how to determine if there is underlying structure in your data
- Other methods: TSNE, UMAP
- **Next time: Unsupervised Learning and clustering**