Inspire…Educate…Transform.

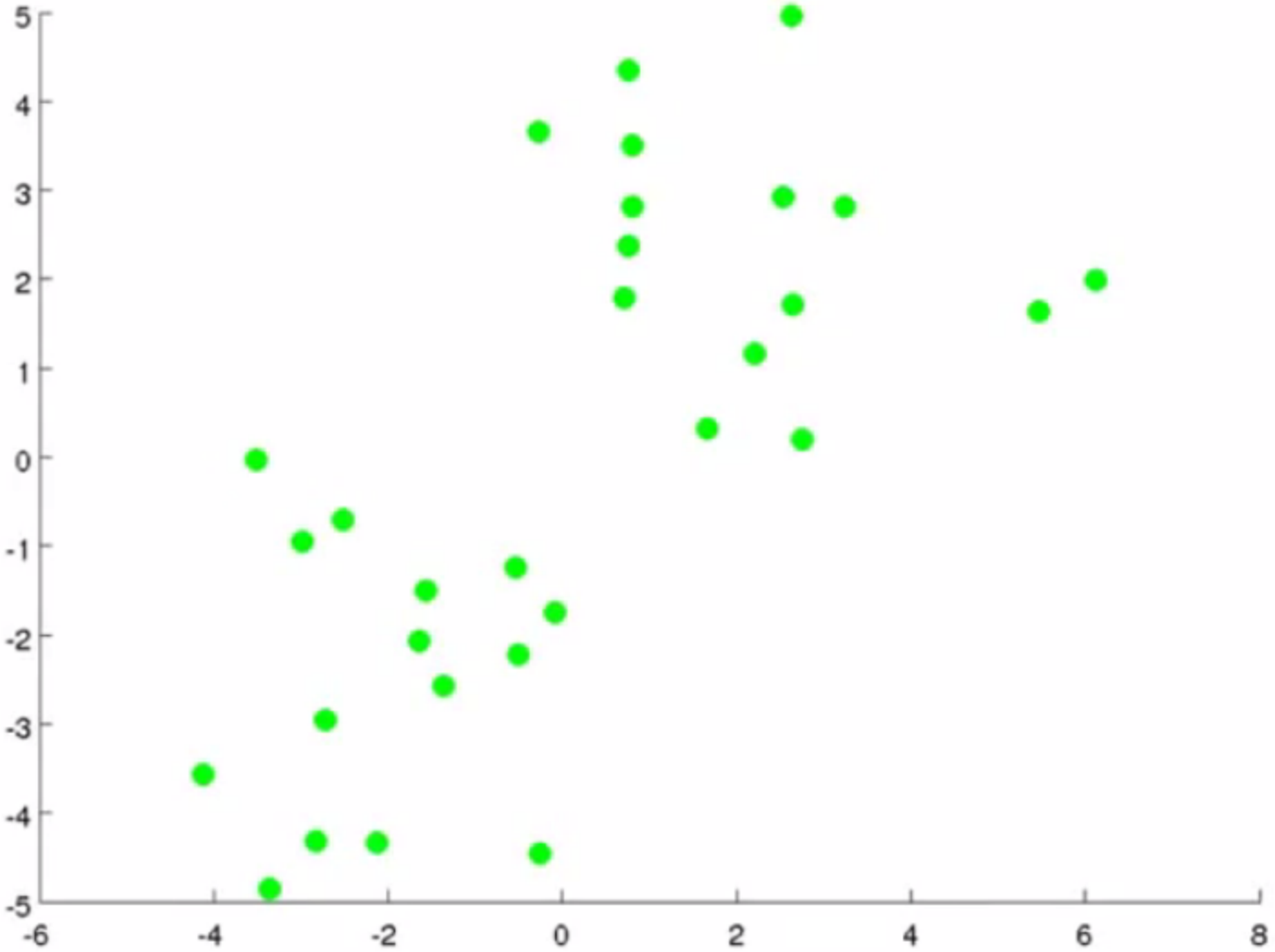
# NLP

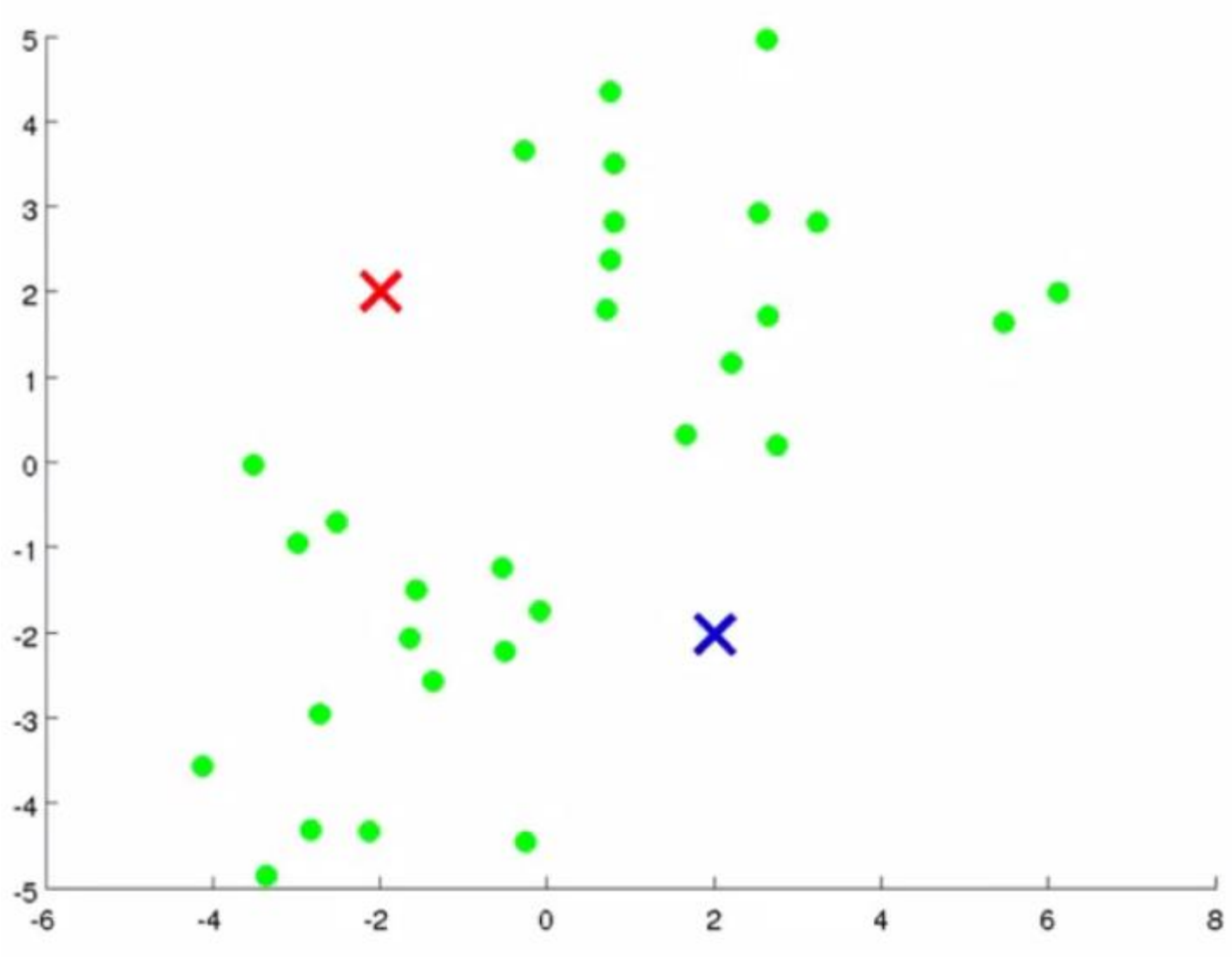
## Text mining and language understanding

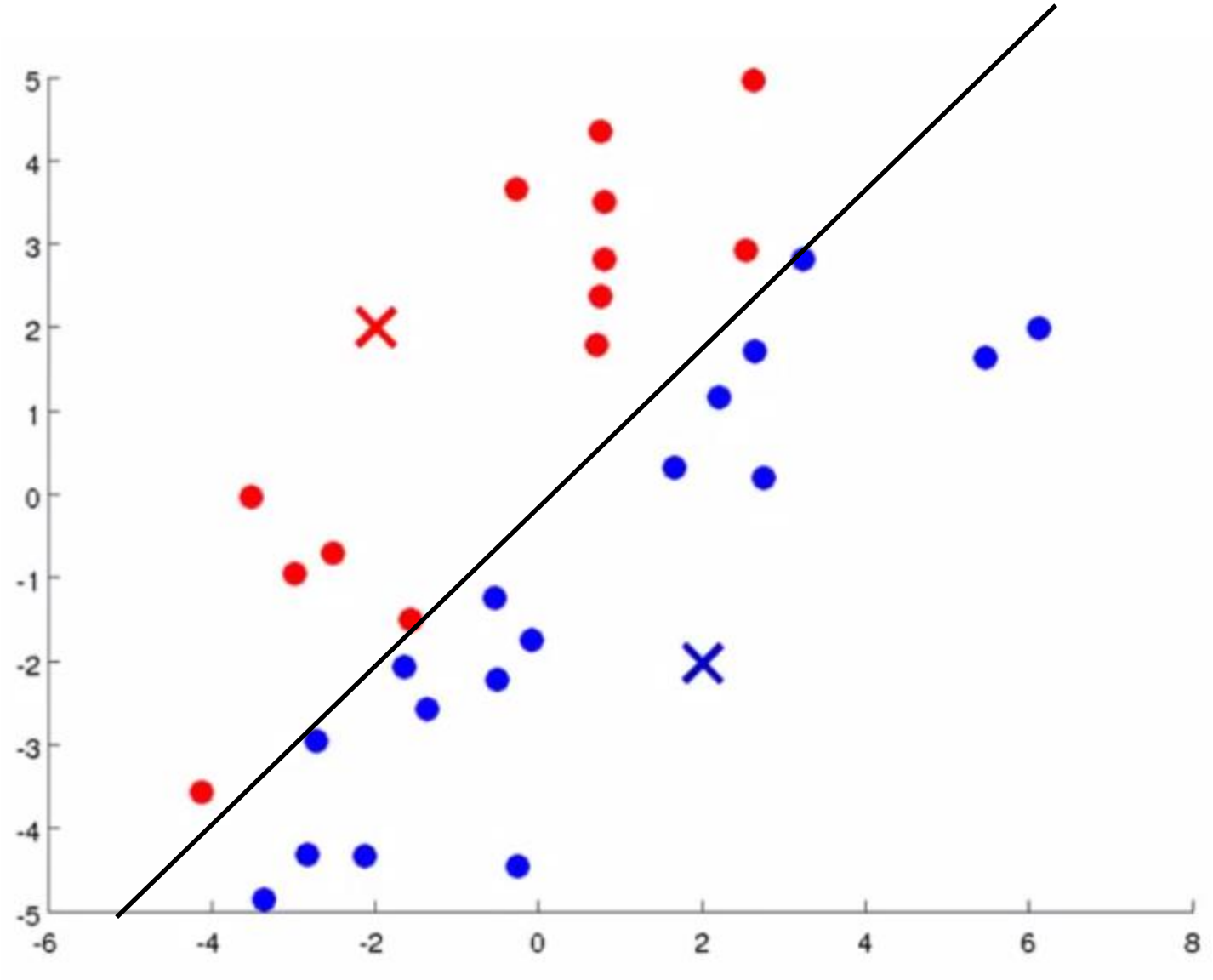
**Dr. K. V Dakshinamurthy**
President, INSOFE

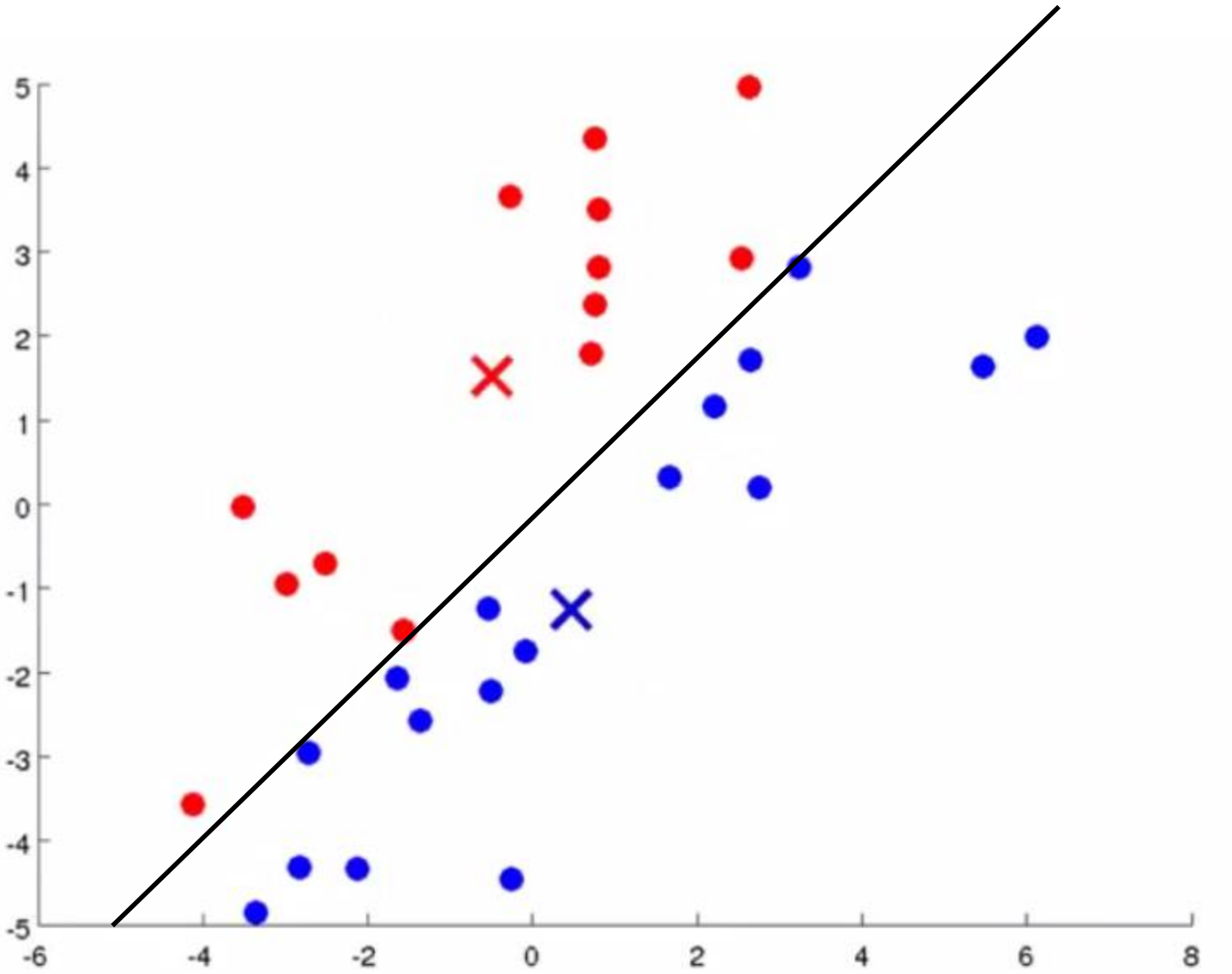
# DOCUMENT CLUSTERING

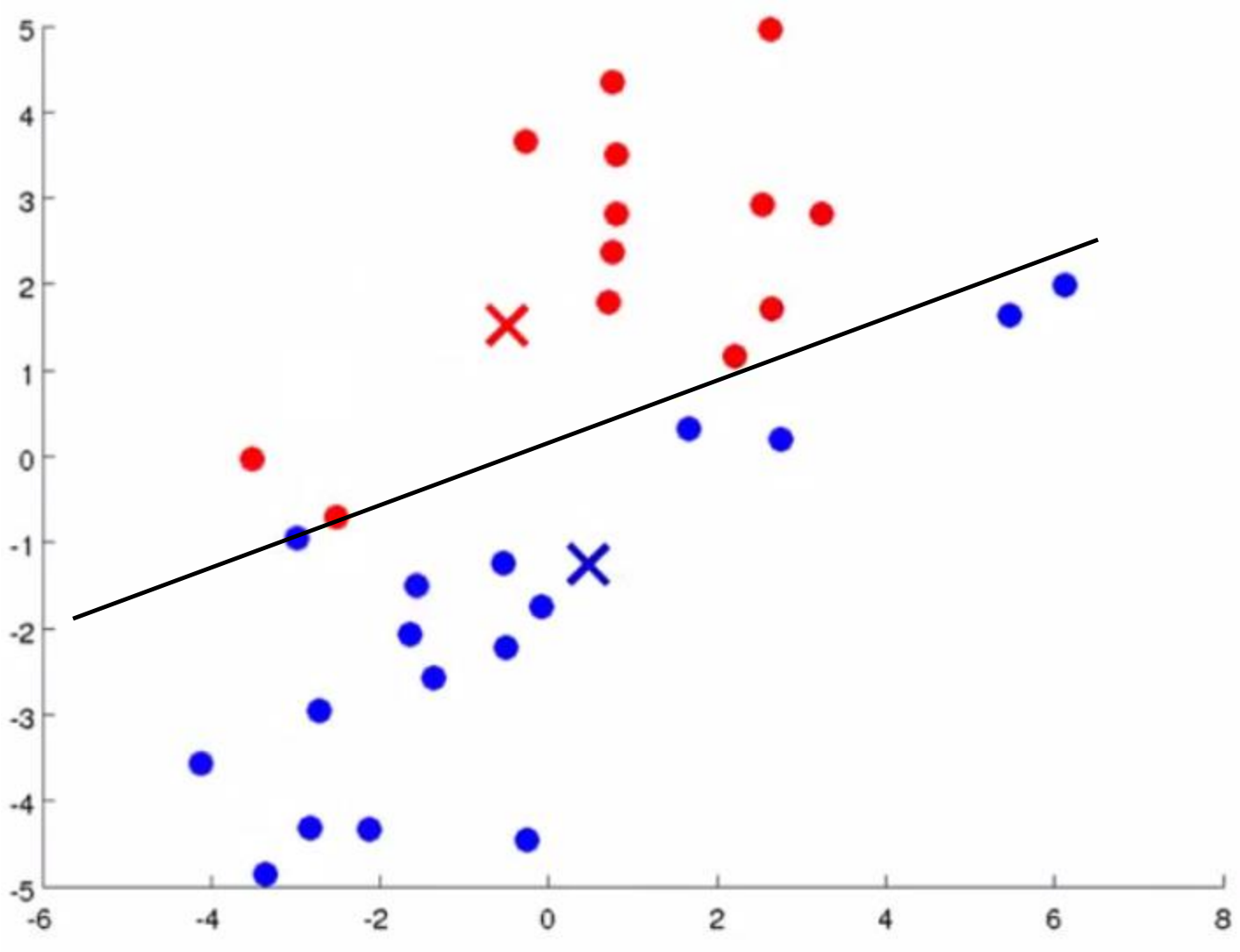
Pictures courtesy Andrew Ngs course on Coursera

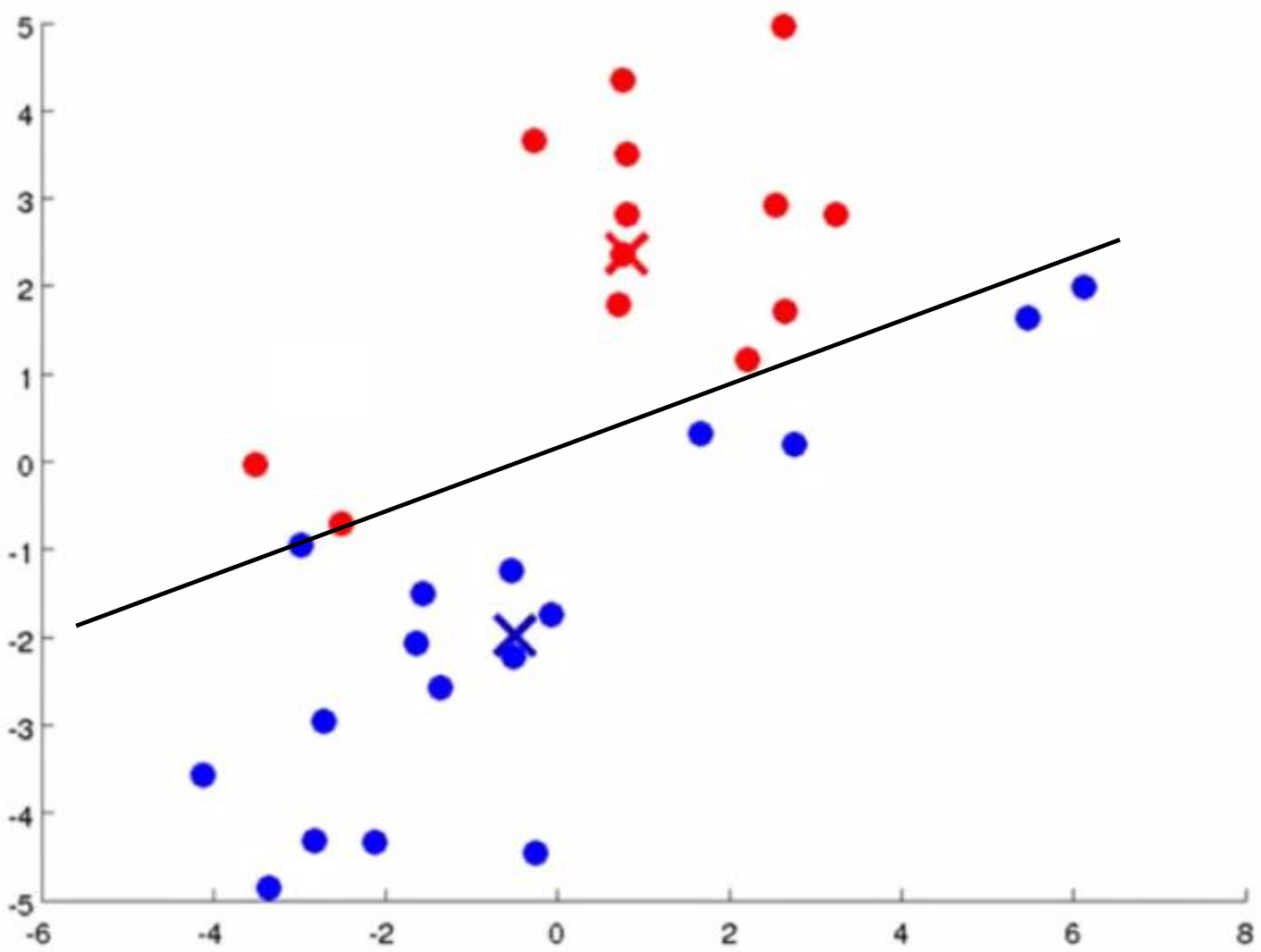


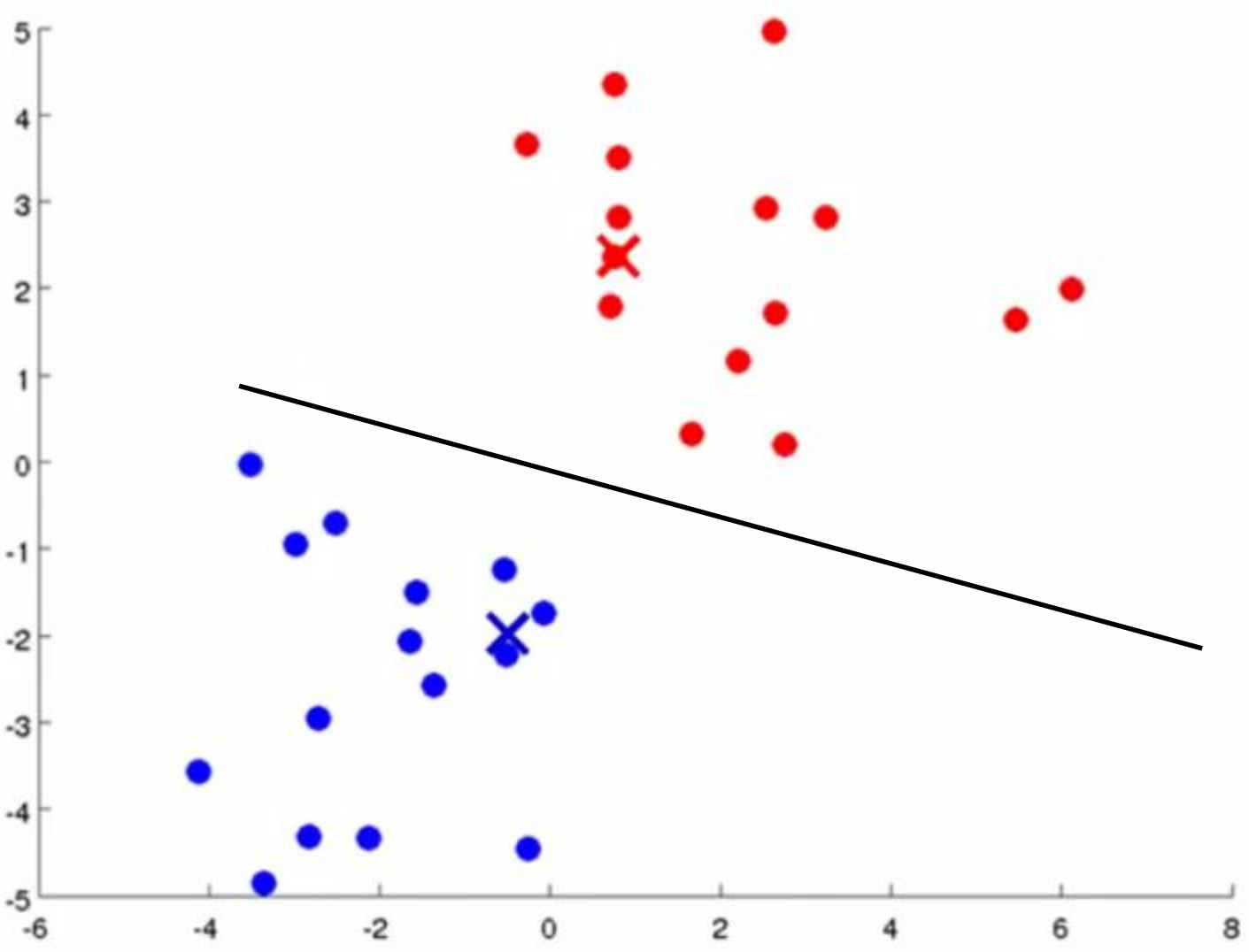

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

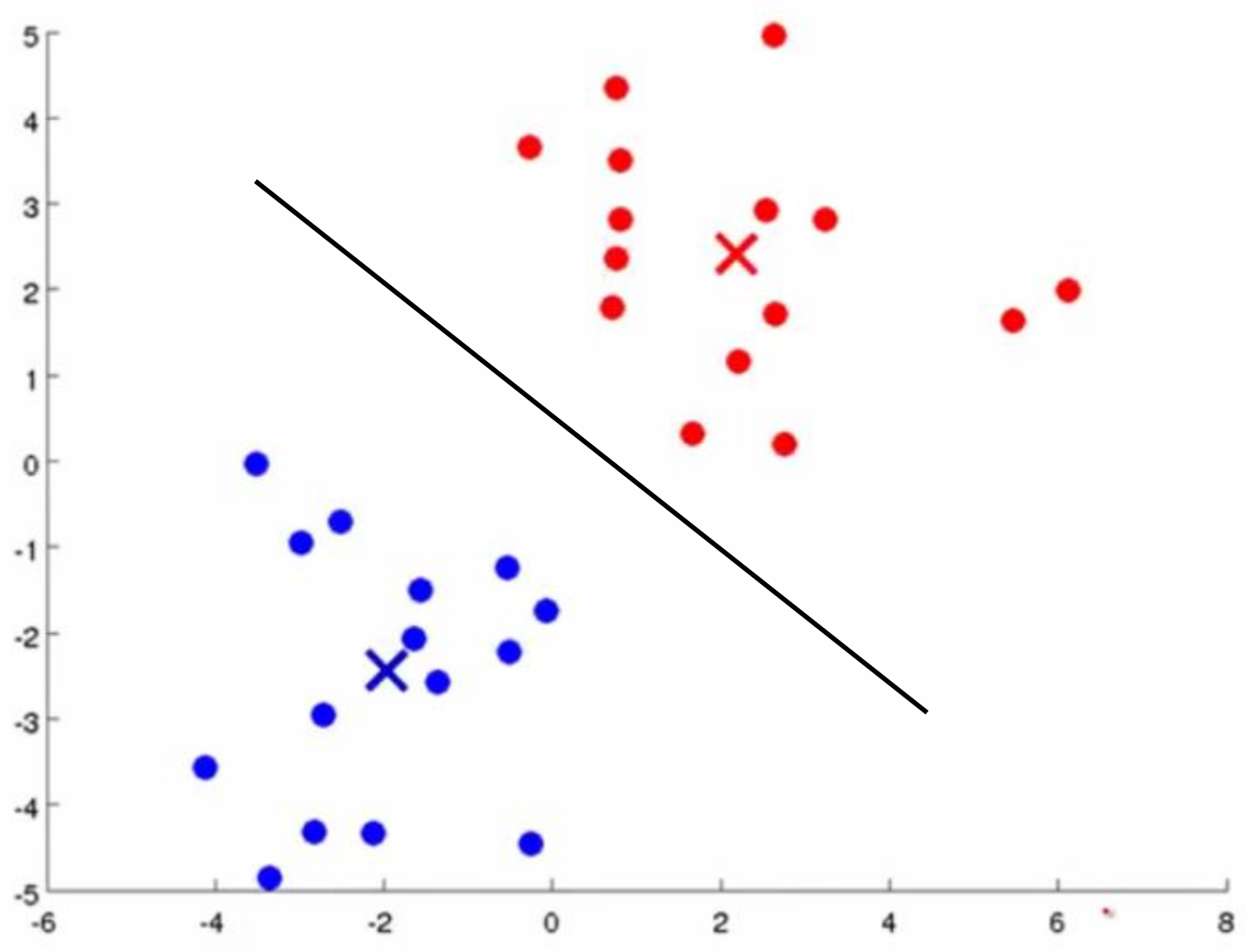

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

# Soft Clustering

- Clustering typically assumes that each instance is given a “hard” assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
- *Soft clustering* gives probabilities that an instance belongs to each of a set of clusters.
- Each instance is assigned a probability distribution across a set of discovered categories (probabilities of all categories must sum to 1).

# Expectation Maximization

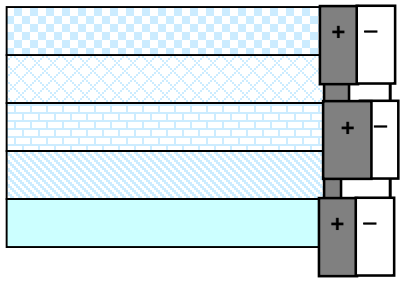
- Probabilistic method for soft clustering. Soft version of *k*-means.
- Direct method that assumes *k* clusters: $\{c_1, c_2, \ldots c_k\}$
- Assumes a probabilistic model of categories that allows computing $P(c_i \mid E)$ for each category, $c_i$, for a given example, $E$.
- For text, typically assume a naïve-Bayes category model.
  - Parameters $\theta = \{P(c_i), P(w_j \mid c_i): i \in \{1,\ldots k\}, j \in \{1,\ldots,|V|\}\}$

# EM

## Initialize:

Assign random probabilistic labels to unlabeled data

*Unlabeled Examples*

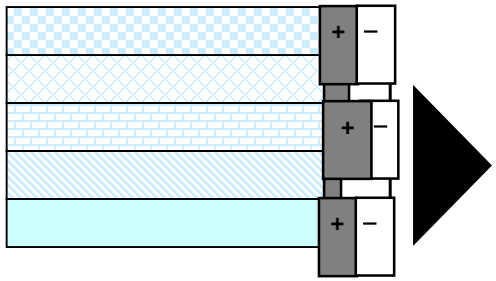


# EM

## Initialize:

Give soft-labeled training data to a probabilistic learner

*Unlabeled Examples*



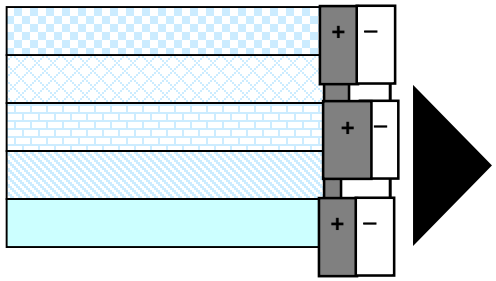

Prob.
Learner

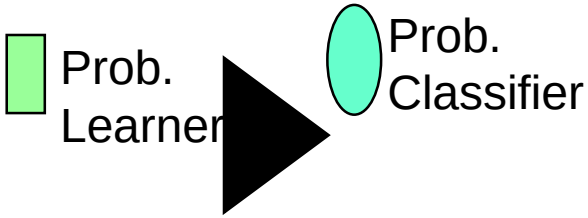
# EM

## Initialize:

Produce a probabilistic classifier

*Unlabeled Examples*

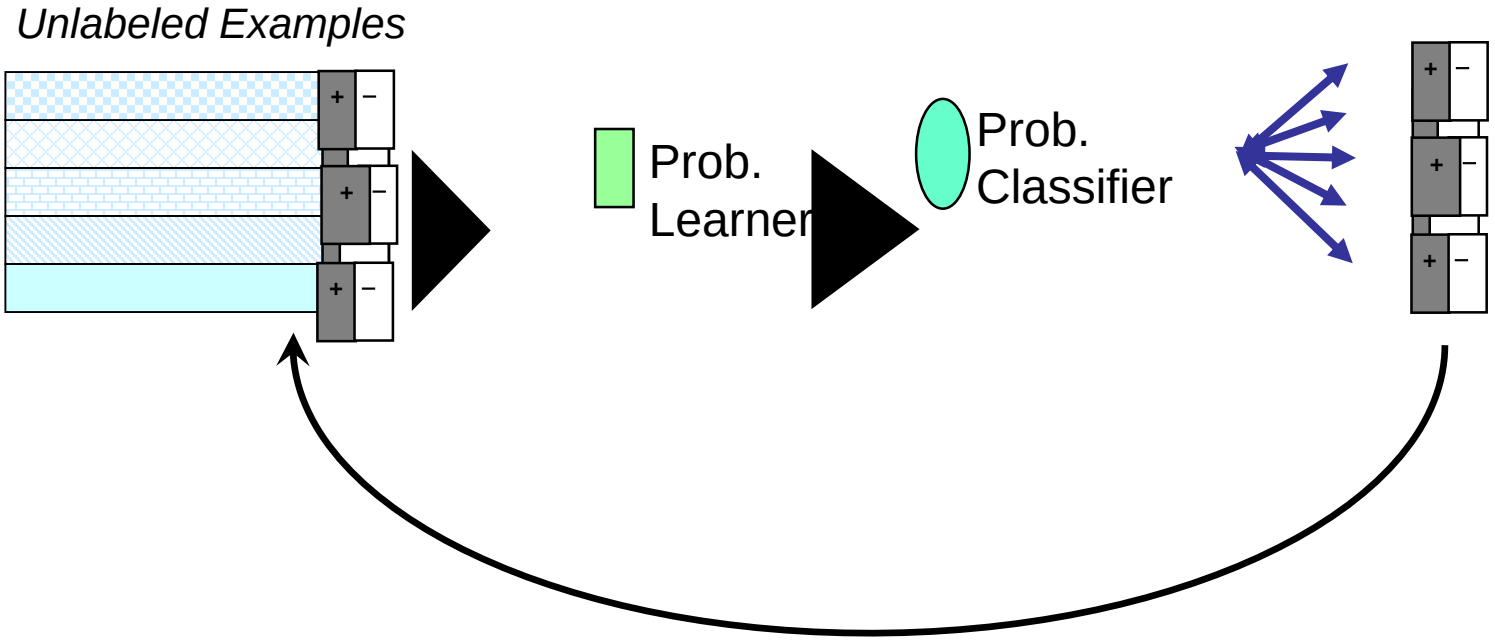


# EM

## E Step:

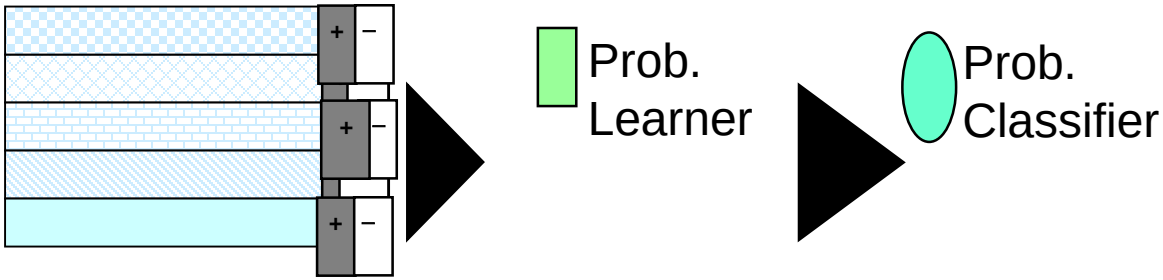
Relabel unlabled data using the trained classifier



# EM

## M step:

Retrain classifier on relabeled data




**Continue EM iterations until probabilistic labels on unlabeled data converge.**

# Naïve Bayes EM

Randomly assign examples probabilistic category labels.

Use standard naïve-Bayes training to learn a probabilistic model with parameters $\theta$ from the labeled data.

Until convergence or until maximum number of iterations reached:

E-Step: Use the naïve Bayes model $\theta$ to compute $P(c_i \mid E)$ for each category and example, and re-label each example using these probability values as soft category labels.

M-Step: Use standard naïve-Bayes training to re-estimate the parameters $\theta$ using these new probabilistic category labels.

# Semi-Supervised EM: Example

- Assume “Catholic” is present in both of the labeled documents for soc.religion.christian, but “Baptist” occurs in *none* of the *labeled* data for this class.
- From labeled data, we learn that “Catholic” is highly indicative of the “Christian” category.
- When labeling unsupervised data, we label several documents with “Catholic” *and* “Baptist” correctly with the “Christian” category.
- When retraining, we learn that “Baptist” is also indicative of a “Christian” document.
- Final learned model is able to correctly assign documents containing *only* “Baptist” to “Christian”.

# Summary of Naïve Bayes

- Bayes' rule can be turned into a classifier
- Maximum A Posteriori (MAP) hypothesis estimation incorporates prior knowledge; Max Likelihood doesn't
- Naive Bayes Classifier is a simple but effective Bayesian classifier for vector data (i.e. data with several attributes) that assumes that attributes are independent given the class.
- Bayesian classification is a generative approach to classification

# Summary: Naive Bayes is Not So Naive

- Naive Bayes won 1st and 2nd place in many text classification applications.
  - E.g.,: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.
- More robust to irrelevant features than many learning methods
  - Irrelevant Features cancel each other without affecting results
  - Decision Trees can suffer heavily from this.
- More robust to concept drift (changing class definition over time)
- Very good in domains with many equally important features
  - Decision Trees suffer from *fragmentation* in such cases – especially if little data
- A good dependable baseline for text classification (but not the best)!
- Optimal if the Independence Assumptions hold: Bayes Optimal Classifier
  - Never true for text, but possible in some domains
- Very Fast Learning and Testing (basically just count the data)
- Low Storage requirements

# LANGUAGE MODELING

# Language modeling problem

- We have a finite set of vocabulary V
  - {the, data, science, study, difficult, is}
  - The stop
  - Is stop
  - The data stop
  - The science stop
  - The data science stop

# Language model

- From a training data set, learn

We need to "learn" a probability distribution $p$
i.e., $p$ is a function that satisfies

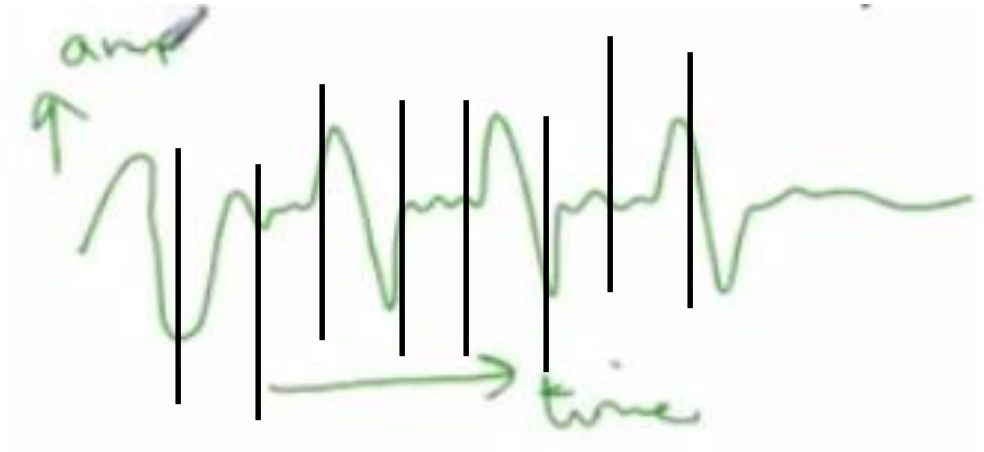
$$\sum_{x \in \mathcal{V}^\dagger} p(x) = 1, \quad p(x) \geq 0 \text{ for all } x \in \mathcal{V}^\dagger$$

# The model should give

- High probability to
  - –The data science study is difficult stop
- Low probability to
  - –The data data science science stop

# An immediate application

- Speech recognition (optical character recognition, handwriting recognition)




Recognize speech

Wreck a nice beach

A good recognition includes not only acoustic matching but also a language model

# A naïve language model

We have $N$ training sentences

For any sentence $x_1 \ldots x_n$, $c(x_1 \ldots x_n)$ is the number of times the sentence is seen in our training data

A naive estimate:

$$p(x_1 \ldots x_n) = \frac{c(x_1 \ldots x_n)}{N}$$

# The issue

- For many sentences, the probability is zero.

- Training size is 20 million in 1990s
- Now it is in billions.

# The cure is a Markov model

Consider a sequence of random variables $X_1, X_2, \ldots X_n$.
Each random variable can take any value in a finite set $\mathcal{V}$.
For now we assume the length $n$ is fixed (e.g., $n = 100$).

Our goal: model

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

# First order Markov process

$$
\begin{aligned}
& P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n) \\
= \quad & P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \\
= \quad & P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_{i-1} = x_{i-1})
\end{aligned}
$$

The first-order Markov assumption: For any $i \in \{2 \ldots n\}$, for any $x_1 \ldots x_i$,

$$P(X_i = x_i | X_1 = x_1 \ldots X_{i-1} = x_{i-1}) = P(X_i = x_i | X_{i-1} = x_{i-1})$$

# Second order Markov process

$$
\begin{aligned}
& P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n) \\
= \; & P(X_1 = x_1) \times P(X_2 = x_2 | X_1 = x_1) \\
& \times \prod_{i=3}^{n} P(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1}) \\
= \; & \prod_{i=1}^{n} P(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})
\end{aligned}
$$

(For convenience we assume $x_0 = x_{-1} =$ *, where * is a special "start" symbol.)

# When the length is varying

We would like the length of the sequence, $n$, to also be a random variable

A simple solution: always define $X_n = \text{STOP}$ where STOP is a special symbol

Then use a Markov process as before:

$$P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$$

$$= \prod_{i=1}^{n} P(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

(For convenience we assume $x_0 = x_{-1} = *$, where * is a special "start" symbol.)

# Trigram language models

A trigram language model consists of:

1. A finite set $\mathcal{V}$
2. A parameter $q(w|u,v)$ for each trigram $u,v,w$ such that $w \in \mathcal{V} \cup \{\text{STOP}\}$, and $u,v \in \mathcal{V} \cup \{*\}$.

For any sentence $x_1 \ldots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \ldots (n-1)$, and $x_n = \text{STOP}$, the probability of the sentence under the trigram language model is

$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-2}, x_{i-1})$$

where we define $x_0 = x_{-1} = *$.

# An example

For the sentence

the dog barks STOP

we would have

$$p(\text{the dog barks STOP}) = q(\text{the}|*, *) \times q(\text{dog}|*, \text{the}) \times q(\text{barks}|\text{the, dog}) \times q(\text{STOP}|\text{dog, barks})$$

# How do you estimate the probabilities?

Remaining estimation problem:

$$q(w_i \mid w_{i-2}, w_{i-1})$$

For example:

$$q(\text{laughs} \mid \text{the, dog})$$

A natural estimate (the "maximum likelihood estimate"):

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

$$q(\text{laughs} \mid \text{the, dog}) = \frac{\text{Count(the, dog, laughs)}}{\text{Count(the, dog)}}$$

# Sparse data problems

A natural estimate (the "maximum likelihood estimate"):

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

$$q(\text{laughs} \mid \text{the, dog}) = \frac{\text{Count(the, dog, laughs)}}{\text{Count(the, dog)}}$$

Say our vocabulary size is $N = |\mathcal{V}|$, then there are $N^3$ parameters in the model.

e.g., $N = 20,000 \quad \Rightarrow \quad 20,000^3 = 8 \times 10^{12}$ parameters

# Linear interpolation

Trigram maximum-likelihood estimate

$$q_{\mathsf{ML}}(w_i \mid w_{i-2}, w_{i-1}) = \frac{\mathsf{Count}(w_{i-2}, w_{i-1}, w_i)}{\mathsf{Count}(w_{i-2}, w_{i-1})}$$

Bigram maximum-likelihood estimate

$$q_{\mathsf{ML}}(w_i \mid w_{i-1}) = \frac{\mathsf{Count}(w_{i-1}, w_i)}{\mathsf{Count}(w_{i-1})}$$

Unigram maximum-likelihood estimate

$$q_{\mathsf{ML}}(w_i) = \frac{\mathsf{Count}(w_i)}{\mathsf{Count}()}$$

# Linear interpolation

Take our estimate $q(w_i \mid w_{i-2}, w_{i-1})$ to be

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \quad & \lambda_1 \times q_{\mathsf{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2 \times q_{\mathsf{ML}}(w_i \mid w_{i-1}) \\ & + \lambda_3 \times q_{\mathsf{ML}}(w_i) \end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all $i$.

# Discounting methods

Say we've seen the following counts:

| $x$ | Count$(x)$ | $q_{\text{ML}}(w_i \mid w_{i-1})$ |
|---|---|---|
| the | 48 | |
| the, dog | 15 | 15/48 |
| the, woman | 11 | 11/48 |
| the, man | 10 | 10/48 |
| the, park | 5 | 5/48 |
| the, job | 2 | 2/48 |
| the, telescope | 1 | 1/48 |
| the, manual | 1 | 1/48 |
| the, afternoon | 1 | 1/48 |
| the, country | 1 | 1/48 |
| the, street | 1 | 1/48 |

The maximum-likelihood estimates are high (particularly for low count items)

# Discounting methods

Now define "discounted" counts,

$\text{Count}^*(x) = \text{Count}(x) - 0.5$

New estimates:

| $x$ | $\text{Count}(x)$ | $\text{Count}^*(x)$ | $\frac{\text{Count}^*(x)}{\text{Count(the)}}$ |
|---|---|---|---|
| the | 48 | | |
| the, dog | 15 | 14.5 | 14.5/48 |
| the, woman | 11 | 10.5 | 10.5/48 |
| the, man | 10 | 9.5 | 9.5/48 |
| the, park | 5 | 4.5 | 4.5/48 |
| the, job | 2 | 1.5 | 1.5/48 |
| the, telescope | 1 | 0.5 | 0.5/48 |
| the, manual | 1 | 0.5 | 0.5/48 |
| the, afternoon | 1 | 0.5 | 0.5/48 |
| the, country | 1 | 0.5 | 0.5/48 |
| the, street | 1 | 0.5 | 0.5/48 |

# Missing probability mass

We now have some "missing probability mass":

$$\alpha(w_{i-1}) = 1 - \sum_{w} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

e.g., in our example, $\alpha(the) = 10 \times 0.5/48 = 5/48$

# Katz Back-Off Models (Trigrams)

- For a trigram model, first define two sets

$$\mathcal{A}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0\}$$
$$\mathcal{B}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) = 0\}$$

- A trigram model is defined in terms of the bigram model:

$$q_{BO}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \dfrac{\text{Count}^*(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ \dfrac{\alpha(w_{i-2}, w_{i-1}) q_{BO}(w_i \mid w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} q_{BO}(w \mid w_{i-1})} & \text{If } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-2}, w_{i-1})} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w)}{\text{Count}(w_{i-2}, w_{i-1})}$$

# Summary

- Three steps in deriving the language model probabilities:
  1. Expand $p(w_1, w_2 \ldots w_n)$ using Chain rule.
  2. Make Markov Independence Assumptions
     $p(w_i \mid w_1, w_2 \ldots w_{i-2}, w_{i-1}) = p(w_i \mid w_{i-2}, w_{i-1})$
  3. Smooth the estimates using low order counts.

- Other methods used to improve language models:
  - "Topic" or "long-range" features.
  - Syntactic models.

It's generally hard to improve on trigram models though!!

# Goodness of a language model

- We have some test data
  - Test data is out of box and has not been used in model development
  - m sentences

# Goodness of the model

We could look at the probability under our model $\prod_{i=1}^{m} p(s_i)$. Or more conveniently, the *log probability*

$$\log \prod_{i=1}^{m} p(s_i) = \sum_{i=1}^{m} \log p(s_i)$$

# Perplexity

In fact the usual evaluation measure is *perplexity*

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \frac{1}{M}\sum_{i=1}^{m} \log p(s_i)$$

and $M$ is the total number of words in the test data.

Lower perplexity is better

# Intuition about perplexity

Say we have a vocabulary $\mathcal{V}$, and $N = |\mathcal{V}| + 1$ and model that predicts

$$q(w|u,v) = \frac{1}{N}$$

$$l = \frac{1}{M}\sum_{1}^{m} \log(p_{s_i})$$

However, $p_{s_i} = \frac{1}{N}^{k_i}$ where $k_i$ is the length of the ith sentence.

Log(psi)= ki log(1/N)

Also, $\sum_1^m k_i = M$

# The perplexity is

$$l = log\frac{1}{N}\ and\ Perplexity = 2^{-\log(\frac{1}{N})} = N$$

Perplexity is a measure of how effectively we reduced the need of vocabulary

# Perplexity: More intuition

Results from Goodman ("A bit of progress in language modeling"), where $|\mathcal{V}| = 50,000$

A unigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i)$.
Perplexity $= 955$

A bigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-1})$.
Perplexity $= 137$

A trigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-2}, x_{i-1})$.
Perplexity $= 74$

# Perplexity: More intuition

Four gram and five grams are observed to give slightly less perplexities with large vocabularies.

But, trigrams are the optimum

# Tagging problems

- What they are?

- Generative models
- Markov, hidden Markov models and dynamic programming (Viterbi algorithm)

# Example of Tagging: Part of Speech

- Input
  - Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarters results
  - Studied as early as in 1980s

# Example of Tagging: POS

Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.

N = Noun
V = Verb
P = Preposition
Adv = Adverb
Adj = Adjective
...

# Named entity extraction: What we want

OUTPUT: Profits soared at [Company Boeing Co.], easily topping forecasts on [Location Wall Street], as their CEO [Person Alan Mulally] announced first quarter results.

# Named entity recognition

OUTPUT:

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

NA = No entity
SC = Start Company
CC = Continue Company
SL = Start Location
CL = Continue Location

...

# Goal

**Training set:**

1 Pierre/NNP Vinken/NNP ,/, 61/CD years/NNS old/JJ ,/, will/MD join/VB the/DT board/NN as/IN a/DT nonexecutive/JJ director/NN Nov./NNP 29/CD ./.
2 Mr./NNP Vinken/NNP is/VBZ chairman/NN of/IN Elsevier/NNP N.V./NNP ,/, the/DT Dutch/NNP publishing/VBG group/NN ./.
3 Rudolph/NNP Agnew/NNP ,/, 55/CD years/NNS old/JJ and/CC chairman/NN of/IN Consolidated/NNP Gold/NNP Fields/NNP PLC/NNP ,/, was/VBD named/VBN a/DT nonexecutive/JJ director/NN of/IN this/DT British/JJ industrial/JJ conglomerate/NN ./.
...
38,219 It/PRP is/VBZ also/RB pulling/VBG 20/CD people/NNS out/IN of/IN Puerto/NNP Rico/NNP ,/, who/WP were/VBD helping/VBG Huricane/NNP Hugo/NNP victims/NNS ,/, and/CC sending/VBG them/PRP to/TO San/NNP Francisco/NNP instead/RB ./.

- From the training set, induce a function/algorithm that maps new sentences to their tag sequences.

# Constraints

- Local
  - "can" has a bias to be a verb. But, it can be a noun
- Contextual
  - By the constraints posed by the surrounding words

Influential/JJ members/NNS of/IN the/DT House/NNP Ways/NNP and/CC Means/NNP Committee/NNP introduced/VBD legislation/NN that/WDT would/MD restrict/VB how/WRB the/DT new/JJ savings-and-loan/NN bailout/NN agency/NN can/MD raise/VB capital/NN ./.

- "Local": e.g., *can* is more likely to be a modal verb MD rather than a noun NN
- "Contextual": e.g., a noun is much more likely than a verb to follow a determiner
- Sometimes these preferences are in conflict:

  *The trash can is in the garage*

# Hidden Markov models

- I am not allowed to see the actual sequence. But, only the outcomes
  - Not the actual POS sequence. But, the words.
  - So, the Markov chain is hidden

# Hidden Markov models

We have an input sentence $x = x_1, x_2, \ldots, x_n$
($x_i$ is the $i$'th word in the sentence)

We have a tag sequence $y = y_1, y_2, \ldots, y_n$
($y_i$ is the $i$'th tag in the sentence)

We'll use an HMM to define

$$p(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$$

for any sentence $x_1 \ldots x_n$ and tag sequence $y_1 \ldots y_n$ of the same length.

Then the most likely tag sequence for $x$ is

$$\arg \max_{y_1 \ldots y_n} p(x_1 \ldots x_n, y_1, y_2, \ldots, y_n)$$

# HMM computation

- $p(x, y) = p(y).p(x|y)$

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i|y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i)$$

Parameters of the model:

- $q(s|u, v)$ for any $s \in \mathcal{S} \cup \{\text{STOP}\}$, $u, v \in \mathcal{S} \cup \{*\}$
- $e(x|s)$ for any $s \in \mathcal{S}$, $x \in \mathcal{V}$

# Example

If we have $n = 3$, $x_1 \ldots x_3$ equal to the sentence *the dog laughs*, and $y_1 \ldots y_4$ equal to the tag sequence D N V STOP, then

$$
\begin{aligned}
& p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) \\
= \; & q(\mathtt{D}|*,*) \times q(\mathtt{N}|*,\mathtt{D}) \times q(\mathtt{V}|\mathtt{D},\mathtt{N}) \times q(\mathtt{STOP}|\mathtt{N},\mathtt{V}) \\
& \times e(\textit{the}|\mathtt{D}) \times e(\textit{dog}|\mathtt{N}) \times e(\textit{laughs}|\mathtt{V})
\end{aligned}
$$

- STOP is a special tag that terminates the sequence
- We take $y_0 = y_{-1} = *$, where * is a special "padding" symbol

# HMMs

$$p(x_1 \dots x_n, y_1 \dots y_n) = \underbrace{q(\text{STOP}|y_{n-1}, y_n) \prod_{j=1}^{n} q(y_j \mid y_{j-2}, y_{j-1})}_{\text{Markov Chain}} \times \underbrace{\prod_{j=1}^{n} e(x_j \mid y_j)}_{x_j\text{'s are observed}}$$

# Estimation of parameters

$$q(\text{Vt} \mid \text{DT}, \text{JJ}) = \lambda_1 \times \frac{\text{Count}(\text{Dt}, \text{JJ}, \text{Vt})}{\text{Count}(\text{Dt}, \text{JJ})} + \lambda_2 \times \frac{\text{Count}(\text{JJ}, \text{Vt})}{\text{Count}(\text{JJ})} + \lambda_3 \times \frac{\text{Count}(\text{Vt})}{\text{Count}()}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1, \quad \text{and for all } i, \lambda_i \geq 0$$

$$e(\text{base} \mid \text{Vt}) = \frac{\text{Count}(\text{Vt}, \text{base})}{\text{Count}(\text{Vt})}$$

If we have never seen x, then e will be zero

# Why is this important?

- Proper nouns etc. would be mostly unseen

- Even with million word training, this is a problem

# Fix

**Step 1**: Split vocabulary into two sets

Frequent words = words occurring $\geq 5$ times in training
Low frequency words = all other words

**Step 2**: Map low frequency words into a small, finite set, depending on prefixes, suffixes etc.

# Fix

[Bikel et. al 1999] **(named-entity recognition)**

| Word class | Example | Intuition |
|---|---|---|
| twoDigitNum | 90 | Two digit year |
| fourDigitNum | 1990 | Four digit year |
| containsDigitAndAlpha | A8956-67 | Product code |
| containsDigitAndDash | 09-96 | Date |
| containsDigitAndSlash | 11/9/89 | Date |
| containsDigitAndComma | 23,000.00 | Monetary amount |
| containsDigitAndPeriod | 1.00 | Monetary amount, percentage |
| othernum | 456789 | Other number |
| allCaps | BBN | Organization |
| capPeriod | M. | Person name initial |
| firstWord | first word of sentence | no useful capitalization information |
| initCap | Sally | Capitalized word |
| lowercase | can | Uncapitalized word |
| other | , | Punctuation marks, all other words |

Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA quarter/NA results/NA ./NA

`firstword`/NA soared/NA at/NA `initCap`/SC Co./CC ,/NA easily/NA `lowercase`/NA forecasts/NA on/NA `initCap`/SL Street/CL ,/NA as/NA their/NA CEO/NA Alan/SP `initCap`/CP announced/NA first/NA quarter/NA results/NA ./NA

NA = No entity
SC = Start Company
CC = Continue Company
SL = Start Location
CL = Continue Location
...

# THE FIX IS SIMPLE BUT HEURISTIC DRIVEN

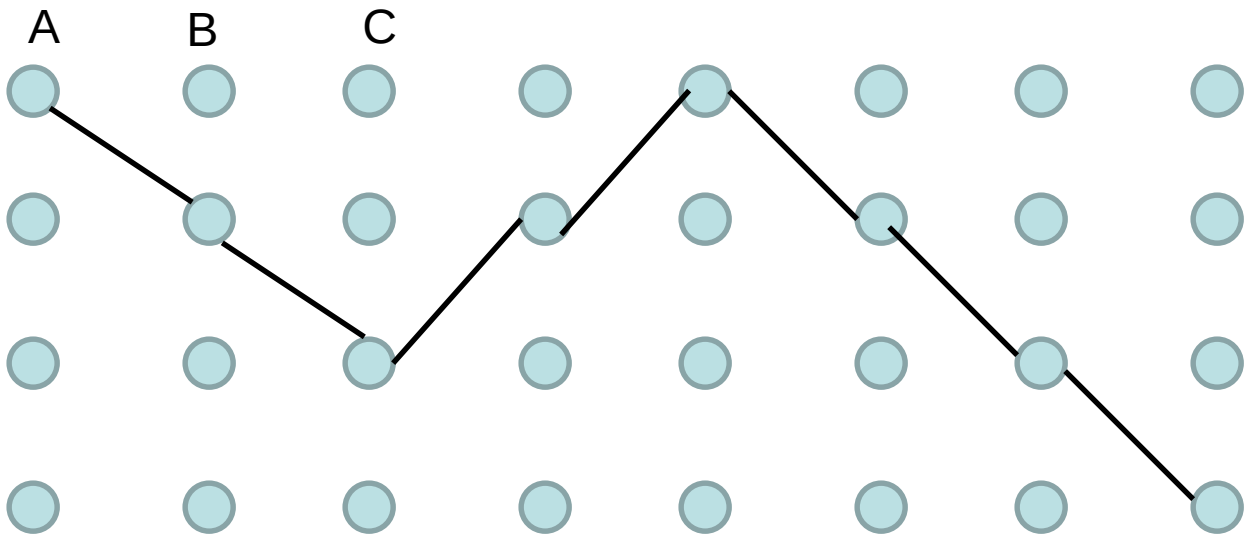
# Brute force

Problem: for an input $x_1 \dots x_n$, find

$$\arg \max_{y_1 \cdots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

where the $\arg\max$ is taken over all sequences $y_1 \dots y_{n+1}$ such that $y_i \in \mathcal{S}$ for $i = 1 \dots n$, and $y_{n+1} = \text{STOP}$.

# Dynamic programming



# The concept

- We compute exhaustively from A to B
  - 4*4=16
- Now to go to C from B
  - We consider only the best path xB1[Ci], xB2[Ci], xB3[Ci] and xB4[C4]
  - From 64, the count comes down to 16
- Now to go to D from C
  - xyC1[Di], xyC2[Di], xyC3[Di] and xyC4[Di]
  - From 256, the count comes down to 16

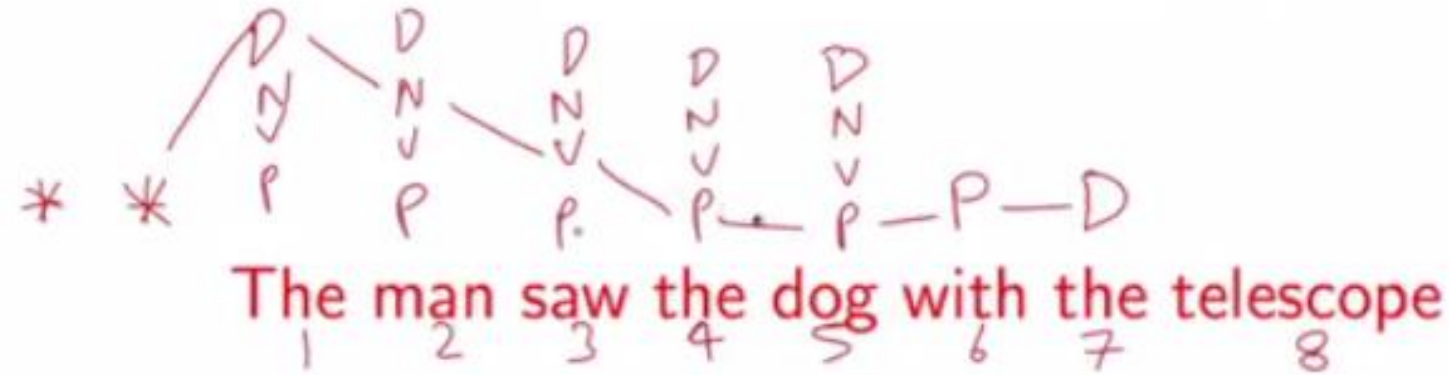
# Dynamic programming

- We are not choosing just the local optima

- However, only the optimal path in a stage is considered

# Example

$\pi(k, u, v)$ = maximum probability of a tag sequence ending in tags $u, v$ at position $k$



# Viterbi Algorithm

- A way to apply the model to test sentences

Problem: for an input $x_1 \dots x_n$, find

$$\arg \max_{y_1 \dots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

where the $\arg\max$ is taken over all sequences $y_1 \dots y_{n+1}$ such that $y_i \in \mathcal{S}$ for $i = 1 \dots n$, and $y_{n+1} = \text{STOP}$.

We assume that $p$ again takes the form

$$p(x_1 \dots x_n, y_1 \dots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i | y_i)$$

Recall that we have assumed in this definition that $y_0 = y_{-1} = *$, and $y_{n+1} = \text{STOP}$.

Hidden markov model taggers are very simple to train (just need to compile counts from the training corpus)

Perform relatively well (over 90% performance on named entity recognition)

Main difficulty is modeling

$$e(word \mid tag)$$

can be very difficult if "words" are complex

# Basic sentiment analysis

- How to do?
  - Count the number of positive and negative sentiments

# Emotional changes in 20th century literature?

- How do we even solve it?
  - Find all the words in all the books published in each year between 1900 and 2000
  - Find the number of words representing each emotion and take a sum

# So, how much time we need to solve?

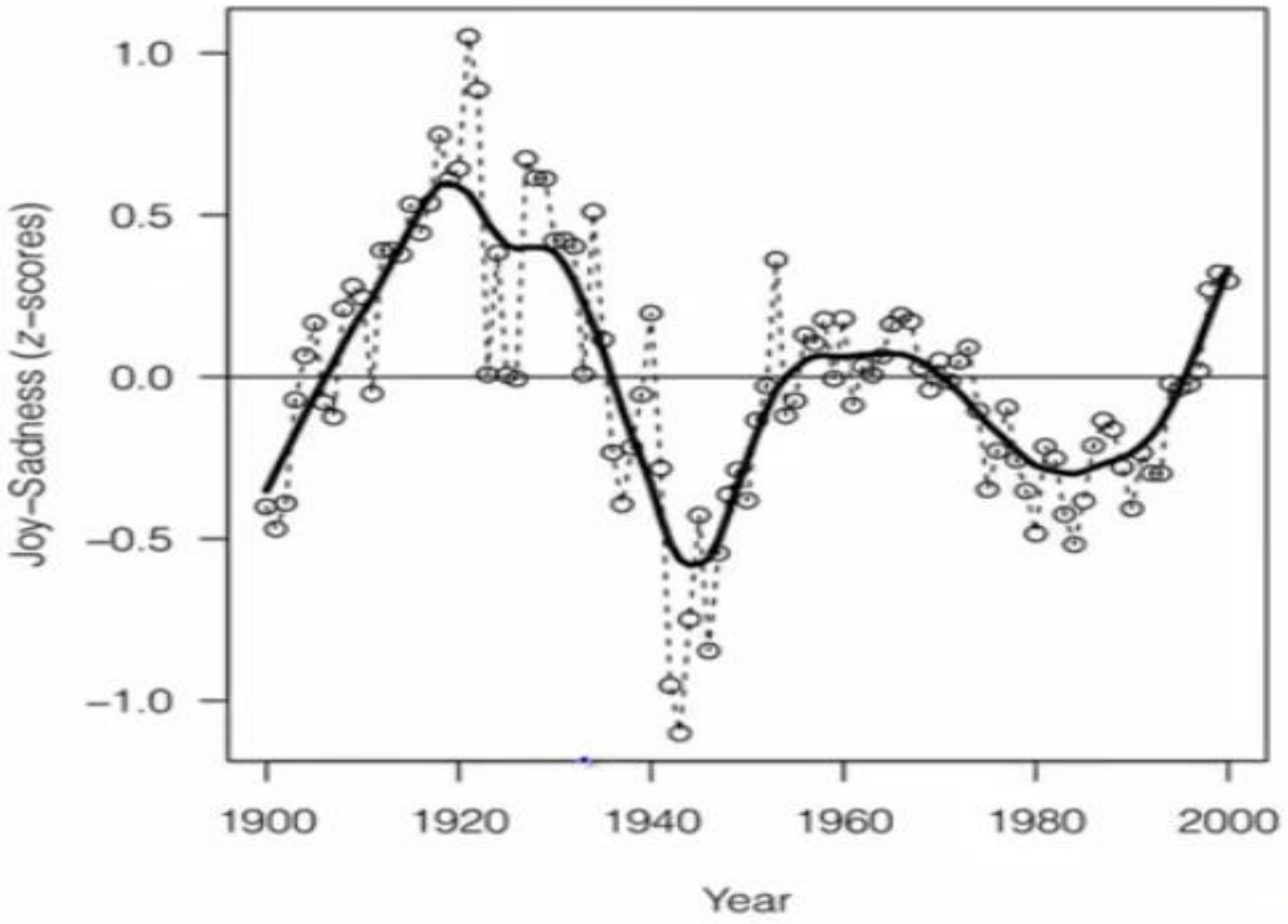
- Google N gram
- Sentiment scores
- Word net

# Approach

- The sentiment score of a book is the n-grams * its sentiment score

- Normalize for various lengths

# Results


1.0
0.5
0.0
−0.5
−1.0
Joy−Sadness (z−scores)
1900
1920
1940
1960
1980
2000
Year

# A more refined approach

- Create a feature matrix
  - 1 gram
  - 2 gram
  - POS

# Supervised learning

- Manual tagging
- Heuristics (distant supervision)
- Build a classifier

# Sentiment Analysis

- Demo:

  http://www.sentiment140.com

- Paper:

  http://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf

# Sentiment modeling

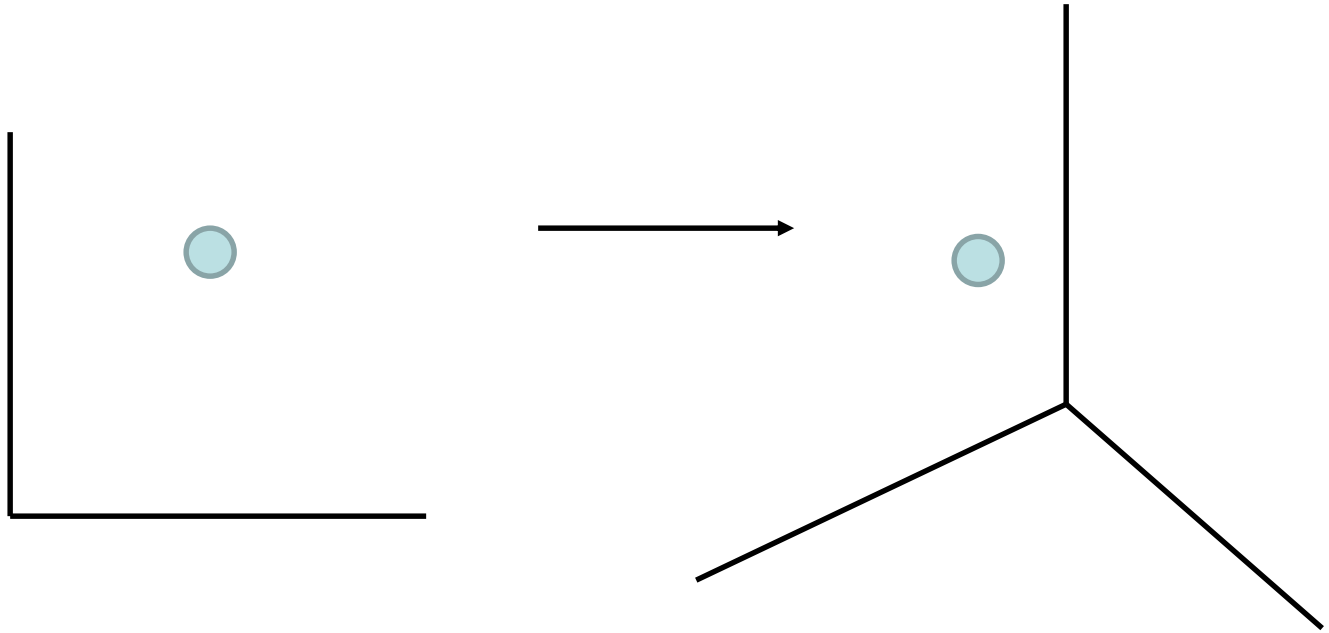
- What can be done with Word2Vec?
- Thoughts?

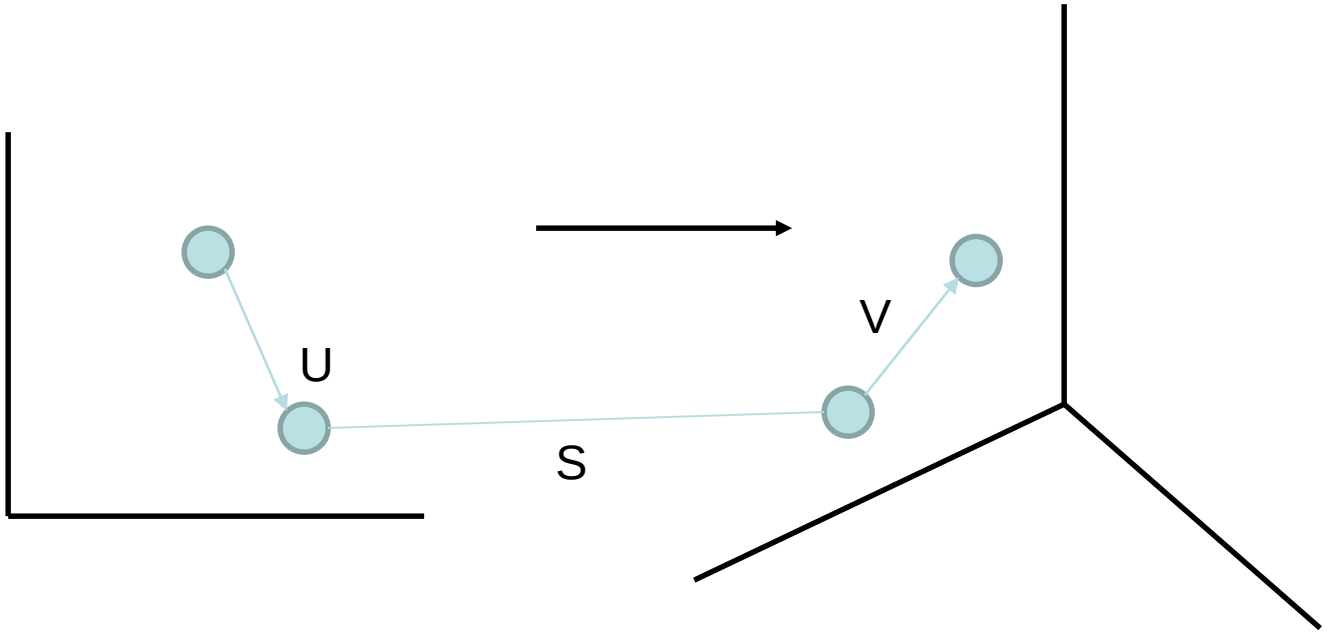
# MATRIX DECOMPOSITION

# WHAT DO WE DO FOR NON-SQUARE TRANSFORMATION MATRICES?

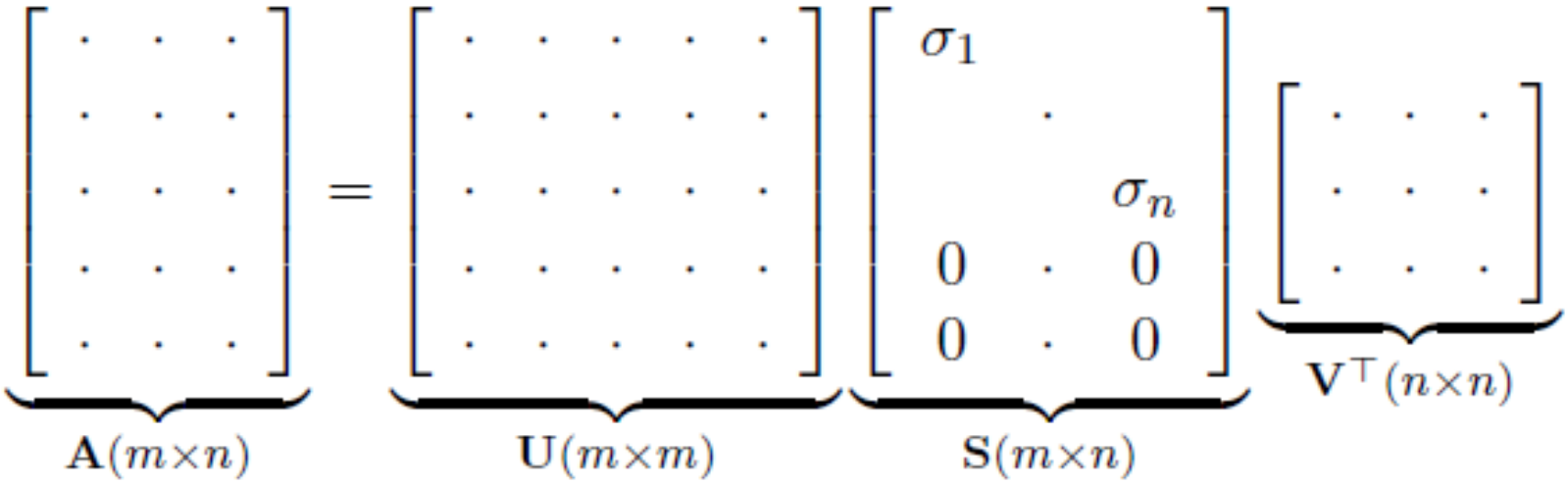
# Non square matrices transform between spaces

$$A_{mXn}Xb_{nX1} = C_{mX1}$$




U
V
S

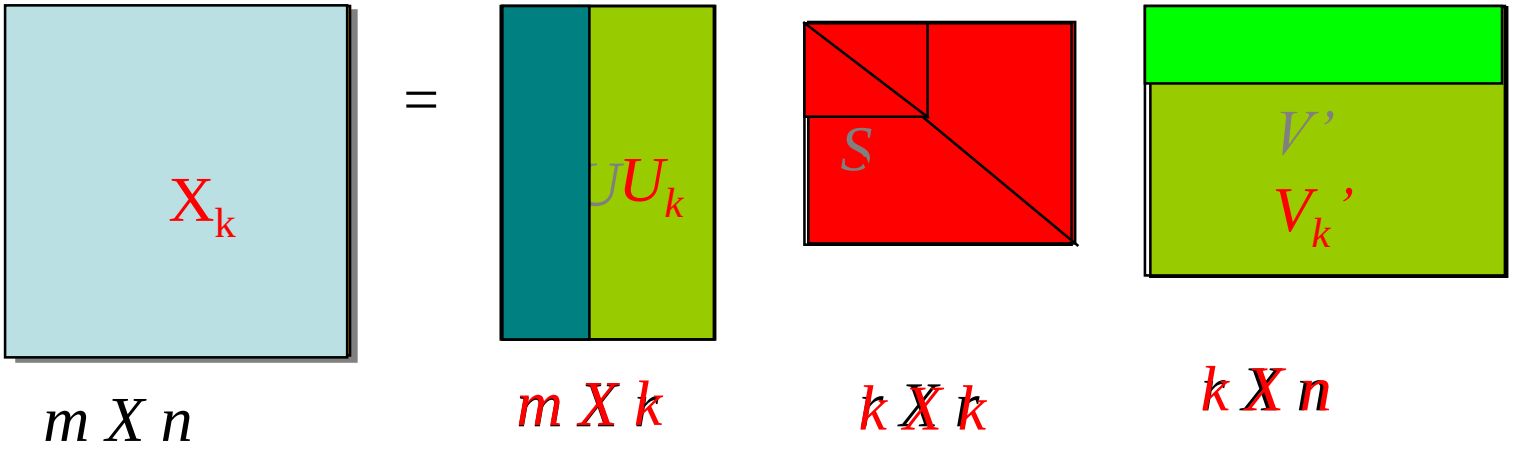
# What is SVD




U and V rotate

S stretches

# SVD: Mathematical Background




The reconstructed matrix $X_k = U_k.S_k.V_k'$ is the closest *rank-k* matrix to the original matrix *R*.

# SVD intuition

- The data may have fewer latent factors that explain it better
  - Students & marks may have some fundamental qualities
  - Players and stadium have some latent factors
  - Terms and documents, Viewers and movies etc.

- SVD enables us to split such data into their own latent factors and remove noise

# SVD and PCA intuition

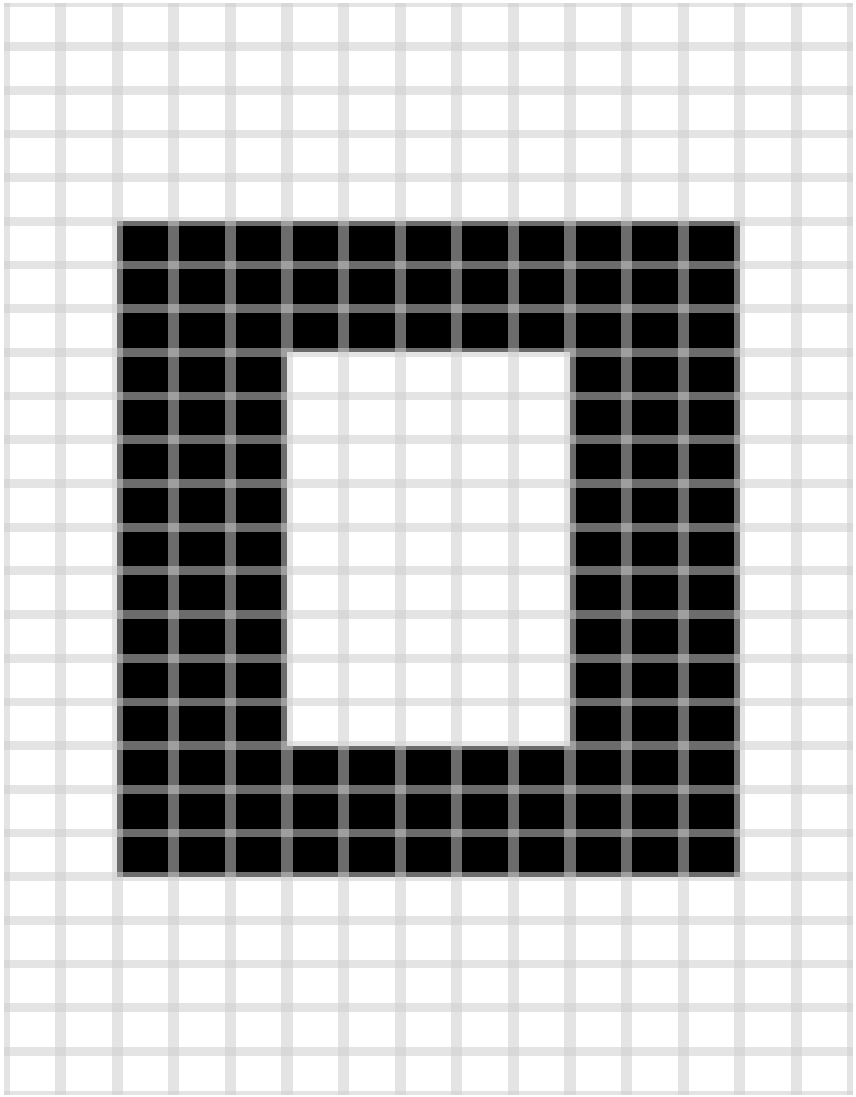
- The original space has a cloud of points
- They may form some rough shape
- They can be reduced into more fundamental factors represented by smaller set of features

# R implementation of SVD

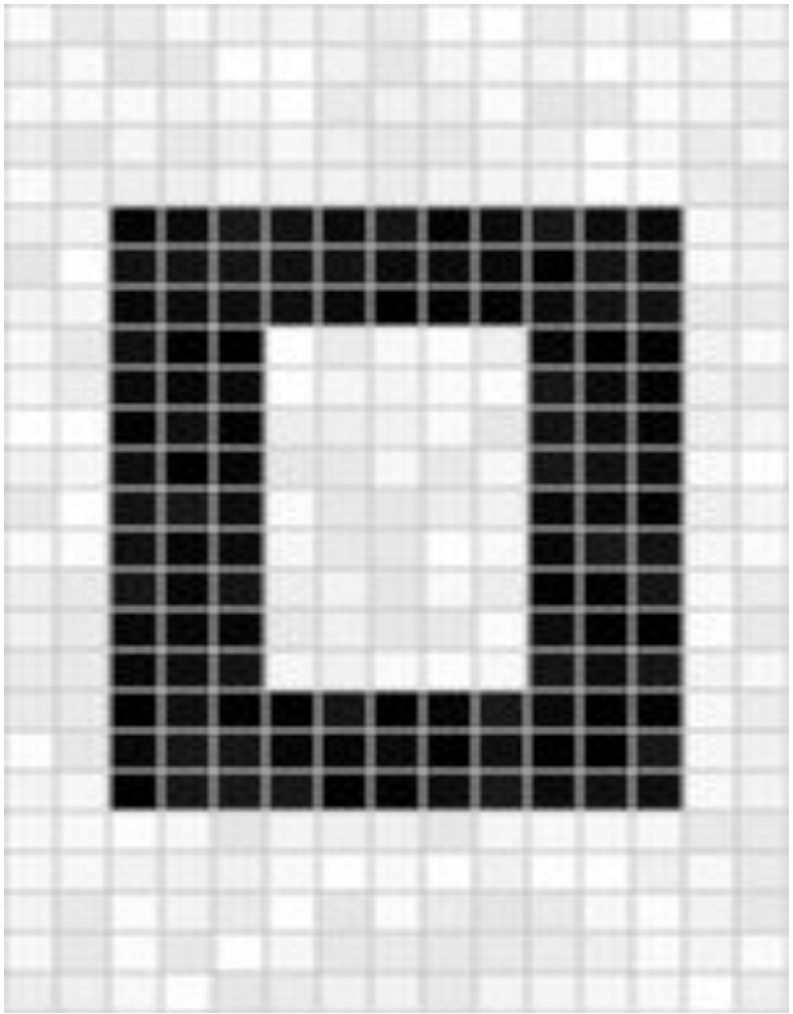
# Applications

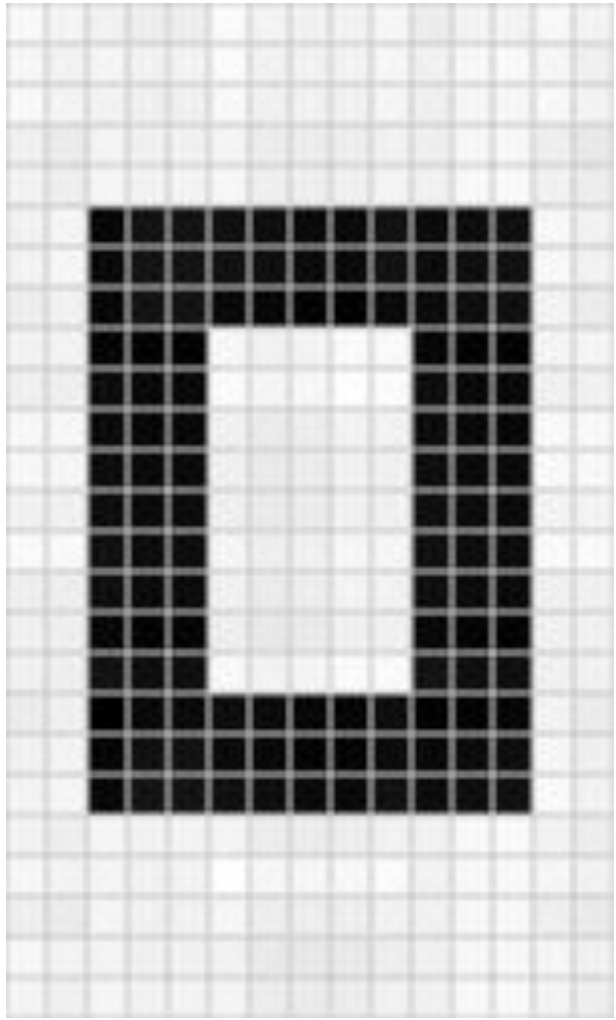
- Compression
- Noise removal
- Recommender system
- Text mining (LSI)
- Spectral clustering
- Feature engineering

# Image compression



# Noise reduction





# Recommender system

- A bunch of users are watching and rating movies
- Users as rows and movies as columns
- Each row will have many zeroes

# Recommender systems

- Have an expert identify qualities of each movie or that are worthy or users that tell how similar their tastes are

- Identify similar movies (or users) using distance metrics

# Recommenders through SVD

- https://www.igvita.com/2007/01/15/svd-recommendation-system-in-ruby/
  - Create a user space through SVD
  - Remove unwanted columns (low eigen values)
  - For every user compute the cosine similarity based distance and identify closest users
  - Look up the ratings they gave to the movie

# Matrix factorization and recommenders

- Quicker than SVD (computationally simpler)
  - http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/
  - http://www2.research.att.com/~volinsky/papers/ieeecomputer.pdf

# Matrix factorization

- $A_{mXn} = P_{mXr} * Q_{rXn}{}^{T}$

- There are r latent features that describe the quality

- Minimize the error by identifying suitable values of P and Q

# Stochastic gradient descent

$$e_{ui} \overset{def}{=} r_{ui} - q_i^T p_u.$$

Then it modifies the parameters by a magnitude proportional to $\gamma$ in the opposite direction of the gradient, yielding:

- $q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$
- $p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$

# Alternating least squares

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj}$$
$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{ik}$$

Having obtained the gradient, we can now formulate the update rules for both $p_{ik}$ and $q_{kj}$:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj}$$
$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}$$

Original

| | D1 | D2 | D3 | D4 |
|---|---|---|---|---|
| U1 | 5 | 3 | - | 1 |
| U2 | 4 | - | - | 1 |
| U3 | 1 | 1 | - | 5 |
| U4 | 1 | - | - | 4 |
| U5 | - | 1 | 5 | 4 |

After alternating least squares

| | D1 | D2 | D3 | D4 |
|---|---|---|---|---|
| U1 | 4.97 | 2.98 | 2.18 | 0.98 |
| U2 | 3.97 | 2.40 | 1.97 | 0.99 |
| U3 | 1.02 | 0.93 | 5.32 | 4.93 |
| U4 | 1.00 | 0.85 | 4.59 | 3.93 |
| U5 | 1.36 | 1.07 | 4.89 | 4.12 |

# SVD in text mining

- Search
- Summarization

# Summarization

- Not writing abstract. But, picking the most important topical sentences
  - Term sentence matrix
  - Latent factors for sentence matrix and pick the highest values sentence for each topic

| | Sentence 1 | Sentence 2 | Sentence 3 | Sentence 4 |
|---|---|---|---|---|
| Topic 1 | 0.22 | 1.51 | 2.11 | 7.67 |
| Topic 2 | 5.33 | 3.22 | 1.10 | 2.43 |
| Topic 3 | 3.43 | 5.34 | 0.74 | 0.71 |
| Topic 4 | 2.11 | 1.31 | 9.54 | 2.33 |

# Limitations

- How many sentences should we choose?
- What about second best sentences in a variety of topics?

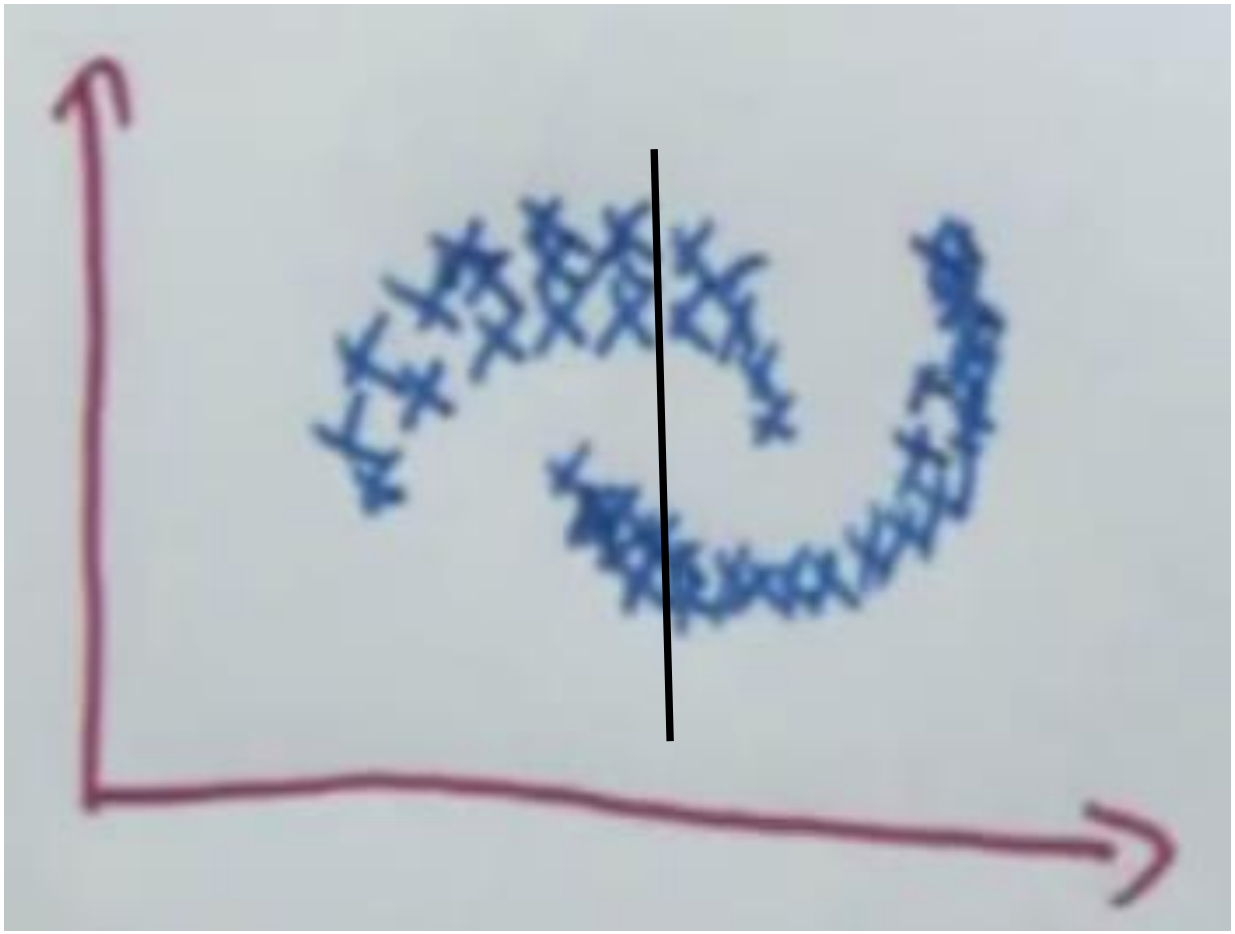
$$s_k = \sqrt{\sum_{i=1}^{n} v_{k,i}^2 \cdot \sigma_i^2} \quad ,$$

https://charlesmartin14.wordpress.com/2012/10/09/spectral-clustering/

http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf

# SPECTRAL CLUSTERING

# Will K-Means work?



# WHEN AFFINITY IS MORE IMPORTANT

# Spectral methods: Clustering

1. project your data into $R^n$
2. define an *Affinity* matrix $A$, using a Gaussian Kernel $K$ or say just an Adjacency matrix (i.e. $A_{i,j} = \delta_{i,j}$)
3. construct the Graph Laplacian from $A$ (i.e. decide on a normalization)
4. solve an Eigenvalue problem, such as $Lv = \lambda v$ (or a Generalized Eigenvalue problem $Lv = \lambda Dv$)
5. select k eigenvectors $\{v_i, i = 1, k\}$ corresponding to the k lowest (or highest) eigenvalues $\{\lambda_i, i = 1, k\}$, to define a k-dimensional subspace $P^t LP$
6. form clusters in this subspace using, say, k-means

# Affinity

Given 2 data points $x_i, x_j$ (projected in $R^n$), we define an Affinity $A_{i,j}$ that is positive, symmetric, and depends on the Euclidian distance $\|x_i - x_j\|$ between the data points

$$A_{i,j} \simeq exp(-\alpha \|x_i - x_j\|^2)$$

We might provide a hard cut off $R$, so that

$$A_{i,j} = 0 \text{ if } \|x_i - x_j\|^2 \geq R$$

# Graph laplacian

$$K_{i,j} \simeq exp(-\alpha \|x_i - x_j\|^2) = I - \alpha \|x_i - x_j\|^2 + \cdots$$

The unnormalized Graph Laplacian is defined as the difference of 2 matrices

$$L_{i,j} = D_{i,j} - W_{i,j}$$

## Other laplacians

1. Simple Laplacian $L = D - A$
2. Normalized Laplacian $L_N = D^{-1/2} L D^{-1/2}$
3. Generalized Laplacian $L_G = D^{-1} L$
4. Relaxed Laplacian $L_\rho = L - \rho D$
5. Ng, Jordan, & Weiss Laplacian $L_{NJW} = D^{-1/2} A D^{-1/2}$, and where $A_{i,i} = 0$
6. and we note the related, smoothed Kernel for Kmeans Kernel Clustering

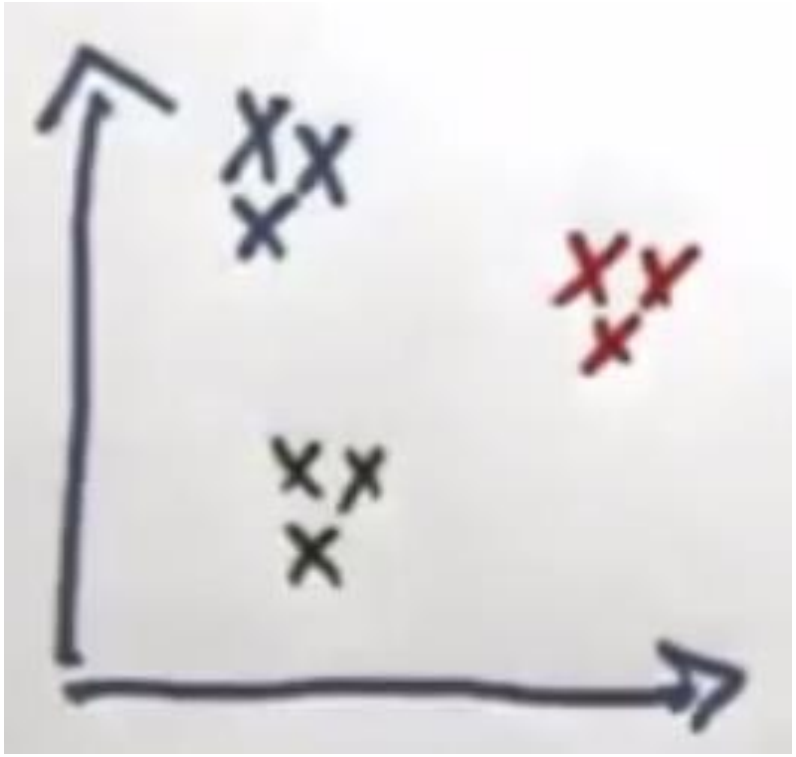
$$K = \sigma D^{-1} + D^{-1} A D^{-1}$$

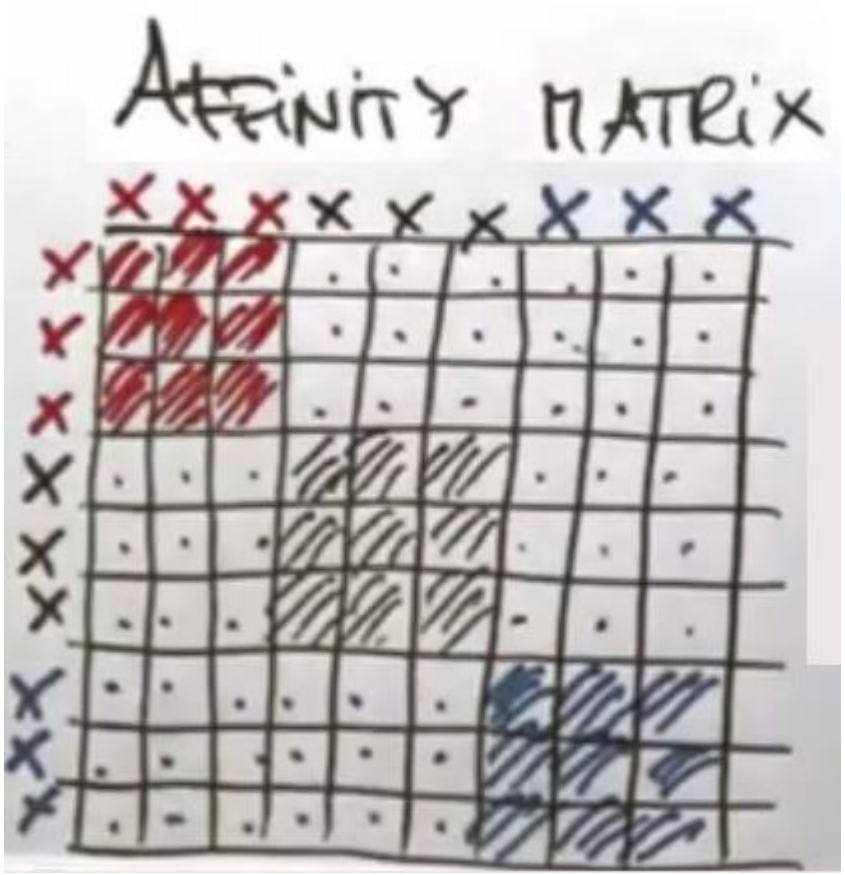
# Eigen space

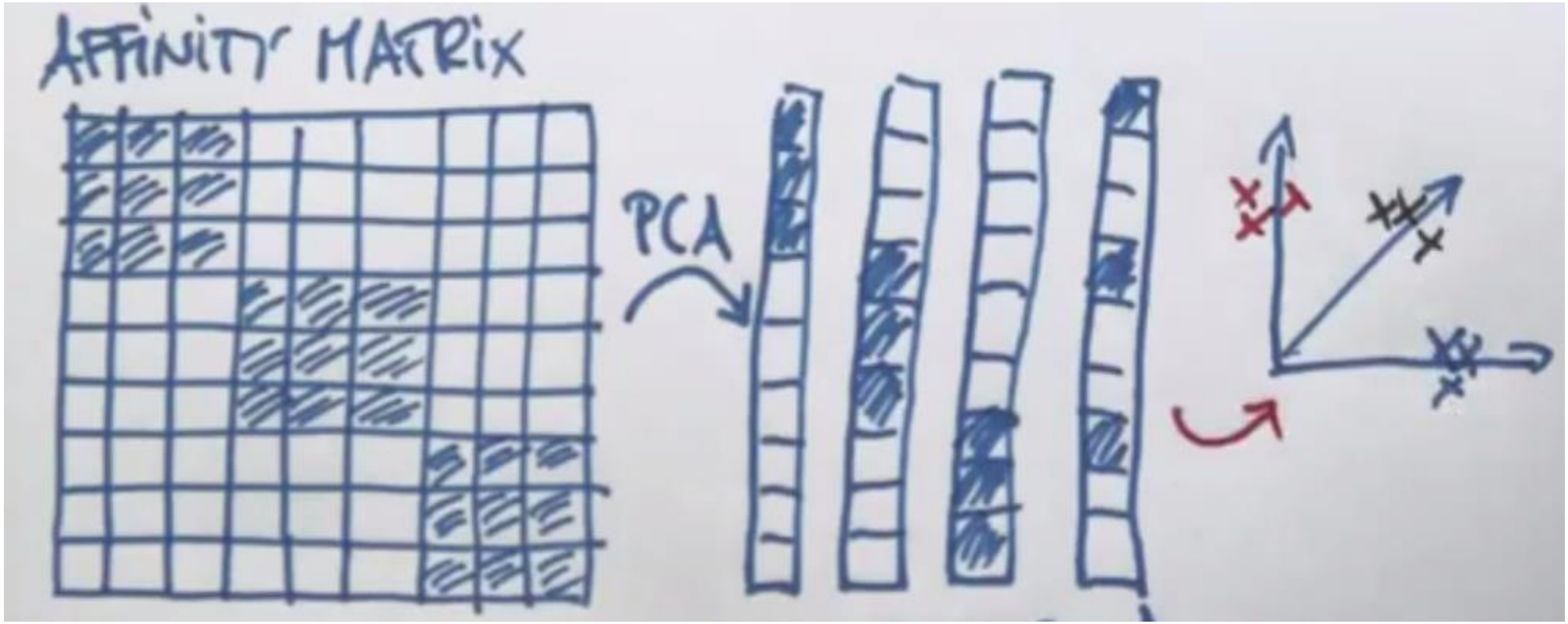
If good clusters can be identified, then the Laplacian $L$ is approximately block-diagonal, with each block defining a cluster. So, if we have 3 major clusters $(C1, C2, C3)$, we would expect

$$\begin{matrix} L_{1,1} & L_{1,2} & L_{1,3} \\ L_{2,1} & L_{2,2} & L_{2,3} \\ L_{3,1} & L_{2,3} & L_{3,3} \end{matrix} \sim \begin{matrix} L_{1,1} & 0 & 0 \\ 0 & L_{2,2} & 0 \\ 0 & 0 & L_{3,3} \end{matrix}$$

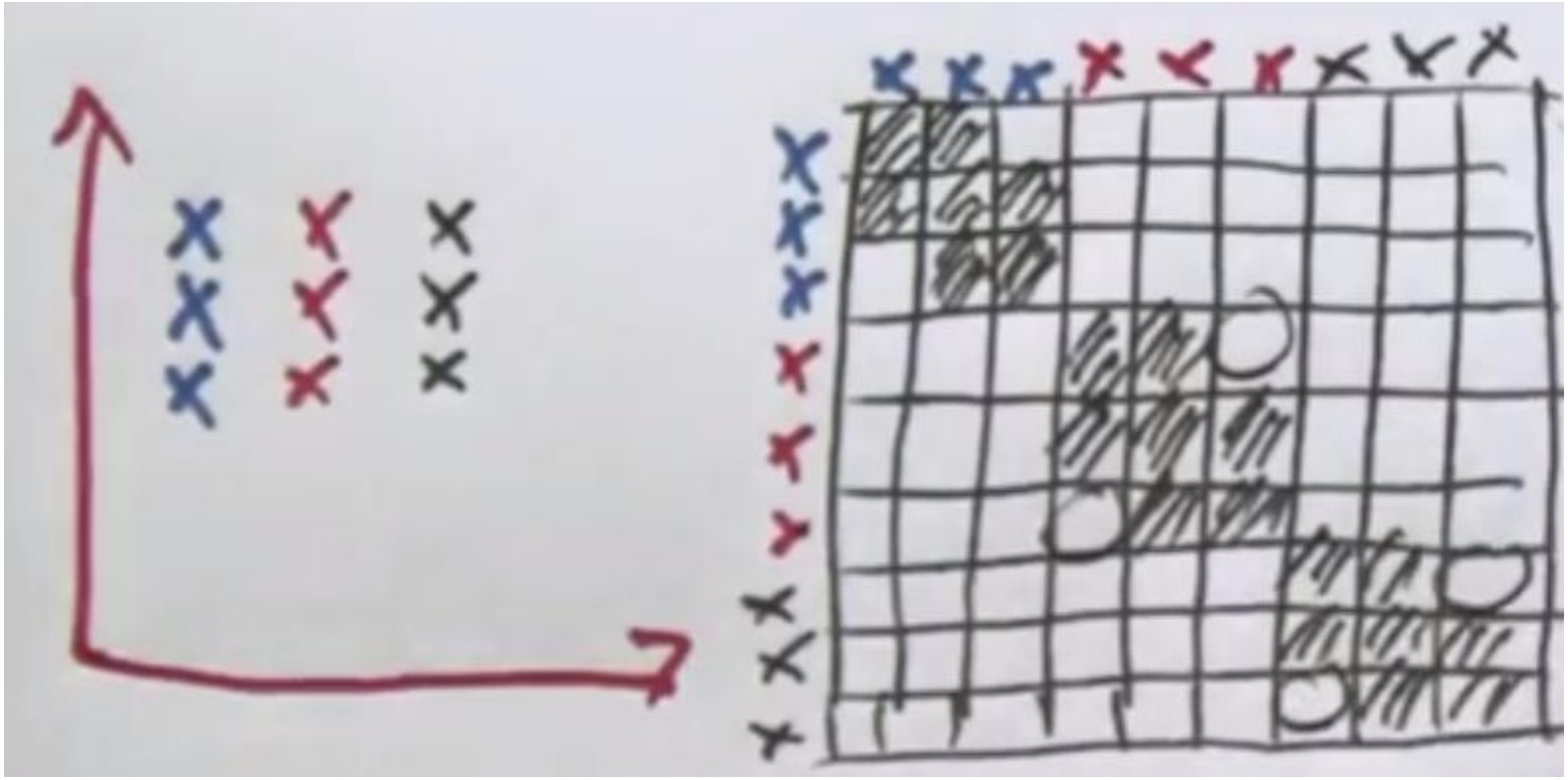
# Spectral methods






AFFINITY MATRIX
PCA

# Spectral clustering: Magic



Inspire...Educate...Transform.

# International School of Engineering


Plot 63/A, 1st Floor, Road # 13, Film Nagar, Jubilee Hills, Hyderabad - 500 033

For Individuals: +91-9502334561/63 or 040-65743991

For Corporates: +91-9618483483

Web: http://www.insofe.edu.in

Facebook: https://www.facebook.com/insofe

Twitter: https://twitter.com/Insofeedu

YouTube: http://www.youtube.com/InsofeVideos

SlideShare: http://www.slideshare.net/INSOFE

LinkedIn: http://www.linkedin.com/company/international-school-of-engineering