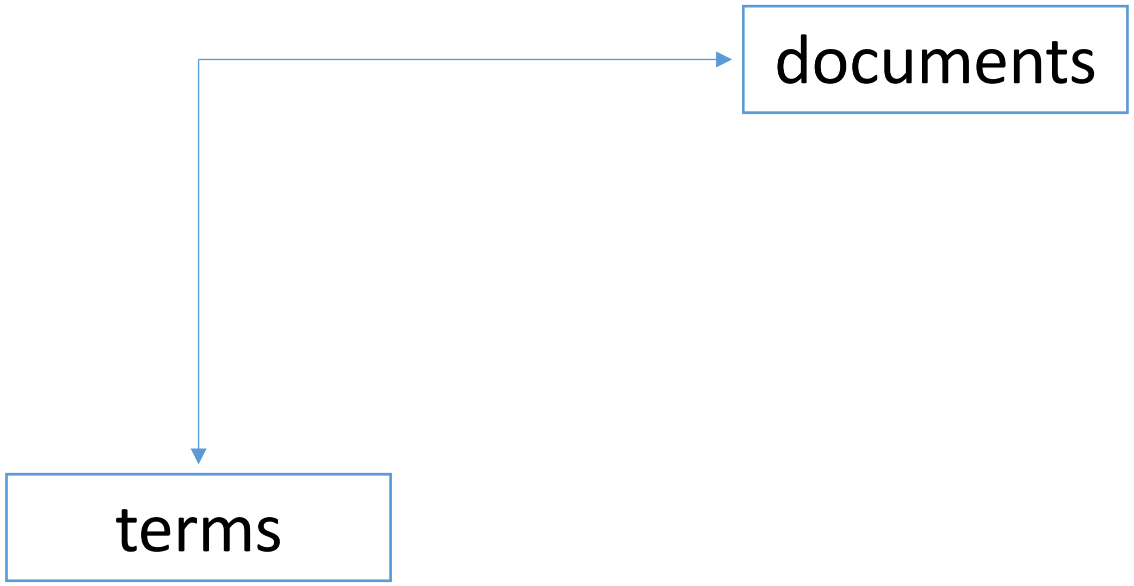
Modelling Text

Embeddings via Vector Spaces

Co-occurrence Matrices

- We represent how often a word occurs in a document
  - **Term-document matrix**



Term-document matrix

- Each cell: count of word w in a document d
  - Each document is a $\boxed{\text{count vector in } \mathbb{N}^{|V|}}$: a column below
  - where $|V|$ is the size of the vocabulary (number of unique words)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Similarity in term-document matrices

Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

For now we have any similarity metric, such as the cosine or the euclidean

The words in a term-document matrix

- Each word has a $\boxed{\text{count vector in } \mathbb{N}^{|D|}}$: a row below
- where $|D|$ is the number of documents

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

The words in a term-document matrix

- Two **words** are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

The word-word or word-context matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of $\pm$ 4 words
- A word is now defined by a vector over counts of context words
- Instead of each vector being of length $|D|$
- Each vector is now of length $|V|$
- The word-word matrix is $|V|\times|V|$

Word-Word matrix
Sample contexts $\pm$ 7 words

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	
…			…				

Word-word matrix

- We showed only 4x6, but the real matrix is 50,000 x 50,000
  - So it's very **sparse**
    - Most values are 0
  - That's OK, since there are lots of efficient algorithms for sparse matrices

- The size of windows depends on your goals
  - The shorter the windows , the more **syntactic** the representation

    $\pm$ 1-3 very syntacticy
  - The longer the windows, the more **semantic** the representation

    $\pm$ 4-10 more semanticy

2 kinds of co-occurrences between 2 words

(Schütze and Pedersen, 1993)

- First-order co-occurrence (**syntagmatic association**):
  - They are typically nearby each other
  - *wrote* is a first-order associate of *book* or *poem*.

- Second-order co-occurrence (**paradigmatic association**):
  - They have similar neighbours
  - *wrote* is a second- order associate of words like *said* or *remarked*

Vector Semantics

Positive Pointwise Mutual Information

(PPMI)

Problem with raw counts

- Raw word frequency is not a great measure of association between words
  - It's very skewed
    - "the" and "of" are very frequent, but maybe not the most discriminative
- We'd rather have a measure that asks whether a context word is **particularly informative** about the target word
  - Positive Pointwise Mutual Information (PPMI)

Pointwise Mutual Information

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

$$\mathrm{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

PMI between two words: (Church & Hanks 1989)

Do words x and y co-occur more than if they were independent?

$$\mathrm{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
    - Unreliable without enormous corpora
      - Imagine w_1 and w_2 whose probability is each 10^{-6}
      - Hard to be sure $p(w_1,w_2)$ is significantly different than 10^{-12}
    - Plus it's not clear people are good at "unrelatedness"
- So we just replace negative PMI values with 0
- Positive PMI (PPMI) between $word_1$ and $word_2$:

$$\mathrm{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

Computing PPMI on a term-context matrix

- Matrix F with W rows (words) and C columns (contexts)
- f_{ij} is # of times w_i occurs in context c_j

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \begin{cases} pmi_{ij} \text{ if } pmi_{ij} > 0 \\ 0 \text{ otherwise} \end{cases}$$

Computing PPMI on a term-context matrix

Entry of cell

Sum of a row

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

Sum of a column

Sum of the table

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \begin{cases} pmi_{ij} \text{ if } \ pmi_{ij} > 0 \\ 0 \text{ otherwise} \end{cases}$$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

	Count(w,context)				
	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

$$p(w = information, c = data) = \frac{6}{19} \approx 0.316$$

$$p(w = information) = \frac{11}{19} \approx 0.579$$

$$p(c = data) = \frac{7}{19} \approx 0.368$$

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{N} \qquad p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{N}$$

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.000	0.000	0.053	0.000	0.053	0.105
pineapple	0.000	0.000	0.053	0.000	0.053	0.105
digital	0.105	0.053	0.000	0.053	0.000	0.211
information	0.053	0.316	0.000	0.211	0.000	0.579
p(context)	0.158	0.368	0.105	0.263	0.105	

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*} p_{*j}}$$

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0,000	0,000	0,053	0,000	0,053	0,105
pineapple	0,000	0,000	0,053	0,000	0,053	0,105
digital	0,105	0,053	0,000	0,053	0,000	0,211
information	0,053	0,316	0,000	0,211	0,000	0,579
p(context)	0,158	0,368	0,105	0,263	0,105	

$$pmi(information, data) = \log_2 \left(\frac{0.316}{0.579 * 0.368} \right) \approx 0{,}57$$

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2,25	-	2,25
pineapple	-	-	2,25	-	2,25
digital	1,66	0,00	-	0,00	-
information	0,00	0,57	-	0,47	-

Weighting PMI

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values

- Two solutions:
  - Give rare words slightly higher probabilities
  - Use add-one smoothing (which has a similar effect)

Weighting PMI: Giving rare context words slightly higher probability

- Raise the context probabilities to $\alpha = 0.75$:

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

- This helps because $P_\alpha(c) > P(c)$ for rare c
  - Consider two events, $P(a) = 0.99$ and $P(b) = 0.01$
  - $P_\alpha(a) = \frac{0.99^{0.75}}{0.99^{0.75}+0.01^{0.75}} = 0.97,\ P_\alpha(b) = \frac{0.01^{0.75}}{0.99^{0.75}+0.01^{0.75}} = 0.03$

Laplace (add-1) smoothing

Laplace Smoothed Count(w,context)

	computer	data	pinch	result	sugar
apricot	1	1	2	1	2
pineapple	1	1	2	1	2
digital	3	2	1	2	1
information	2	7	1	5	1

p(w,context) [add-1] **p(w)**

	computer	data	pinch	result	sugar	p(w)
apricot	0.026	0.026	0.051	0.026	0.051	0.179
pineapple	0.026	0.026	0.051	0.026	0.051	0.179
digital	0.077	0.051	0.026	0.051	0.026	0.231
information	0.051	0.179	0.026	0.128	0.026	0.410
p(context)	0.179	0.282	0.154	0.231	0.154	

Use Add-2 smoothing

add-2 Smoothed Count(w,context)

	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

p(w,context) [add-2] **p(w)**

	computer	data	pinch	result	sugar	p(w)
apricot	0,034	0,034	0,051	0,034	0,051	0,203
pineapple	0,034	0,034	0,051	0,034	0,051	0,203
digital	0,068	0,051	0,034	0,051	0,034	0,237
information	0,051	0,136	0,034	0,102	0,034	0,356
p(context)	0,186	0,254	0,169	0,220	0,169	

PPMI versus add-2 smoothed PPMI

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

		PPMI(w,context)[add-2]			
	computer	data	pinch	result	sugar
apricot	0,00	0,00	0,56	0,00	0,56
pineapple	0,00	0,00	0,56	0,00	0,56
digital	0,62	0,00	0,00	0,00	0,00
information	0,00	0,58	0,00	0,37	0,00

Vector Semantics

Measuring similarity:

the cosine

Cosine for computing similarity

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v} \bullet \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|} \bullet \frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal

Cosine similarity

- Geben Sie hier eine Formel ein.Which pair of words is more similar?

Word/Word	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1

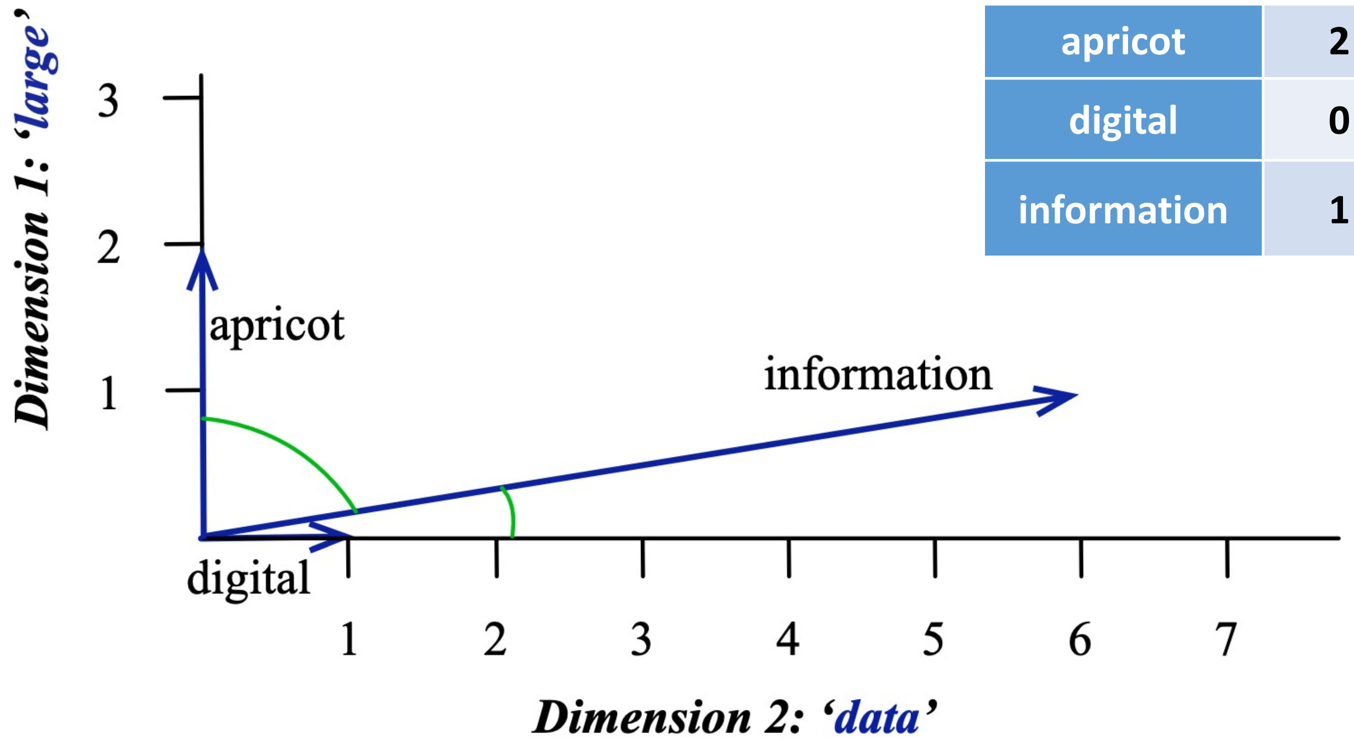
$$cosine(apricot, information) = \frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = \frac{2}{\sqrt{4\sqrt{38}}} = 0.16$$

$$cosine(digital, information) = \frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = \frac{8}{\sqrt{5\sqrt{38}}} = 0.58$$

$$cosine(apricot, digital) = \frac{0+0+0}{\sqrt{4+0+0}\sqrt{0+1+4}} = 0$$

Visualizing vectors and angles

Word/Word	large	data
apricot	2	0
digital	0	1
information	1	6



Other possible similarity measures

$$\text{sim}_{\text{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} (v_i+w_i)}$$

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2})$$

Vector Semantics

Dense Vectors

Sparse versus dense vectors

- PPMI vectors are
  - **long** (length |V|= 20,000 to 50,000)
  - **sparse** (most elements are zero)

- Alternative: learn vectors which are
  - **short** (length 200-1000)
  - **dense** (most elements are non-zero)

Sparse versus dense vectors

- Why dense vectors?
  - Short vectors may be easier to use as features in machine learning (less weights to tune)
  - Dense vectors may generalize better than storing explicit counts
  - They may do better at capturing synonymy:
    - *car* and *automobile* are synonyms; but are represented as distinct dimensions; this fails to capture similarity between a word with *car* as a neighbour and a word with *automobile* as a neighbour

Three methods for getting short dense vectors

- Singular Value Decomposition (SVD) (short)
  - A special case of this is called LSA – Latent Semantic Analysis
- “Neural Language Model”-inspired predictive models
  - skip-grams and CBOW
- Brown clustering/GloVe

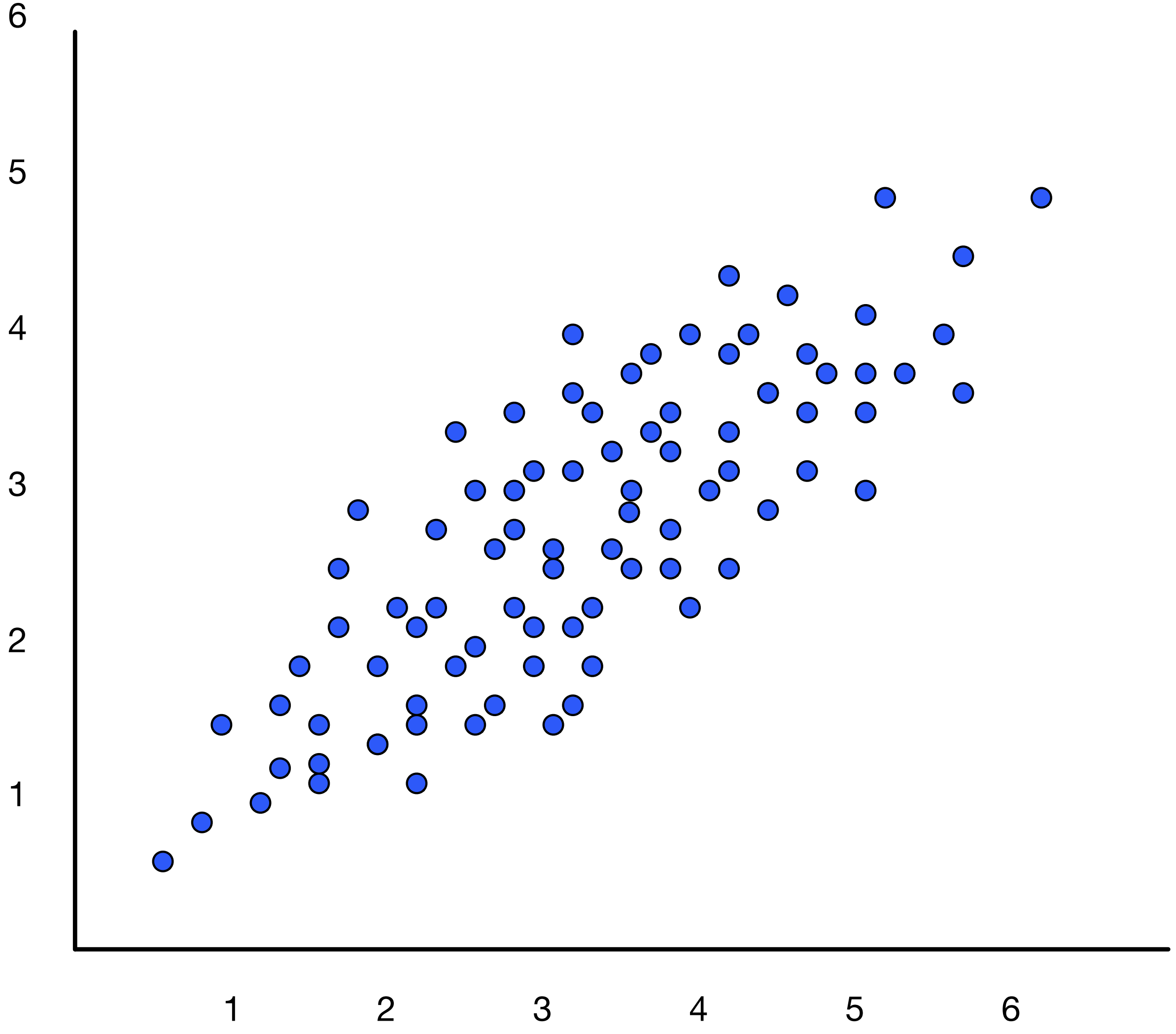
Vector Semantics

Dense Vectors via SVD

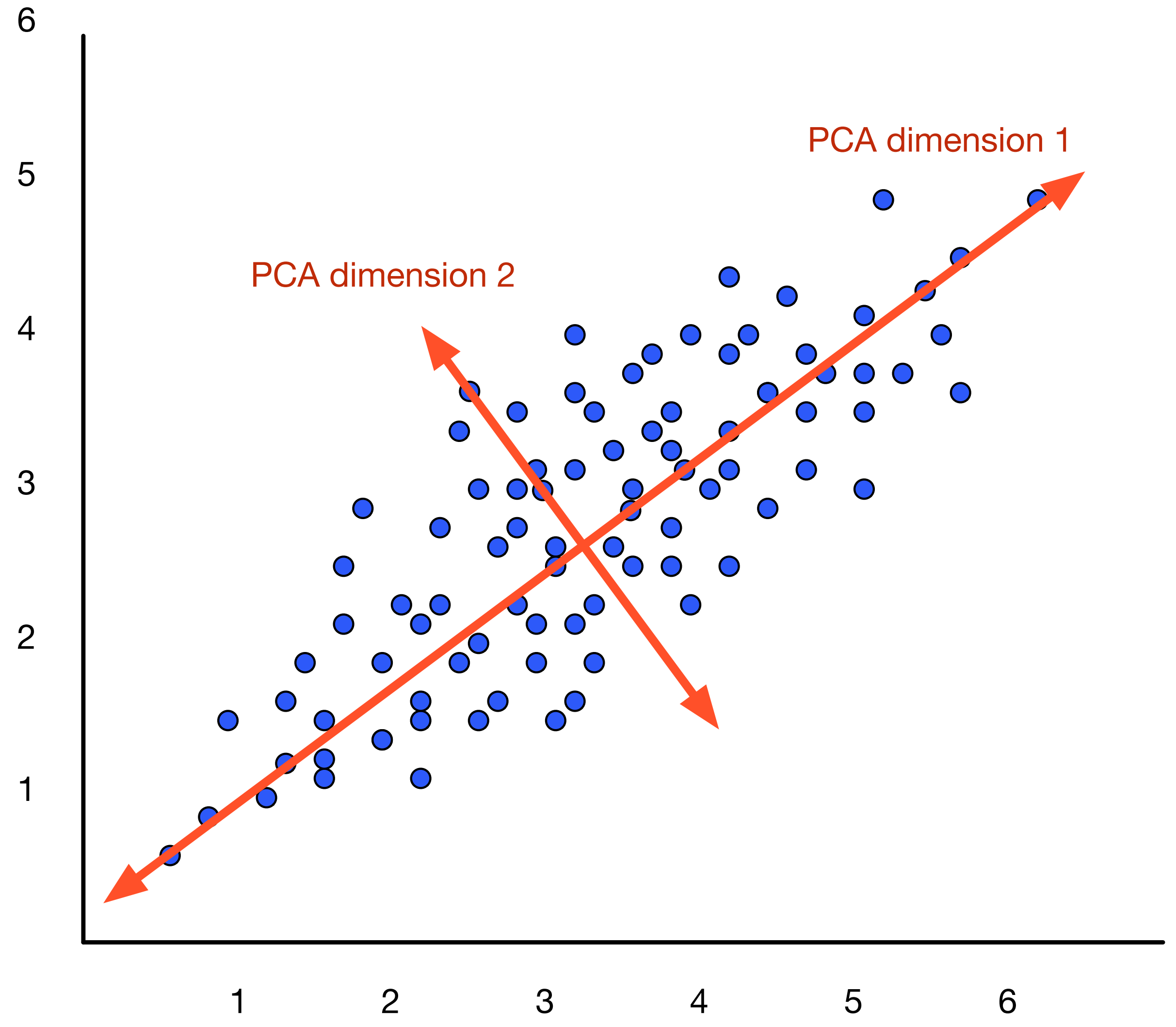
Intuition

- Approximate an N-dimensional dataset using fewer dimensions
- By first rotating the axes into a new space
- In which the highest order dimension captures the most variance in the original dataset
- And the next dimension captures the next most variance, etc.
- Many such (related) methods:
  - PCA – principle components analysis
  - Factor Analysis

Dimensionality reduction



Dimensionality reduction



Singular Value Decomposition

Done in lecture "Information Retrieval"

Any rectangular w x c matrix X equals the product of 3 matrices:

W: Rows corresponding to original but m columns represents a dimension in a new latent space, such that

- m column vectors are orthogonal to each other
- Columns are ordered by the amount of variance in the dataset each new dimension accounts for

S: Diagonal *m* x *m* matrix of **singular values** expressing the importance of each dimension

C: Columns corresponding to original but m rows corresponding to singular values