Distributional Semantics, Pt. II

LING 571 — Deep Processing for NLP
November 6th, 2018
Ryan Georgi

Recap

- We can represent words as vectors
  - Each entry in the vector is a score for its correlation with another word
  - If a word occurs frequently "tall" compared to other words, we might assume height is an important quality of the word
- In these extremely large vectors, most entries are zero

The Curse of Dimensionality

The Problem with High Dimensionality

	tasty	delicious	disgusting	flavorful	tree
pear	0	1	0	0	0
apple	0	0	0	1	1
watermelon	1	0	0	0	0
paw_paw	0	0	1	0	0
family	0	0	0	0	1

The Problem with High Dimensionality

The cosine similarity for these words will be zero!

	tasty	delicious	disgusting	flavorful	tree
pear	0	1	0	0	0
apple	0	0	0	1	1
watermelon	1	0	0	0	0
paw_paw	0	0	1	0	0
family	0	0	0	0	1

The Problem with High Dimensionality

The cosine similarity for these words will be >0 (0.293)

	tasty	delicious	disgusting	flavorful	tree
pear	0	1	0	0	0
apple	0	0	0	1	1
watermelon	1	0	0	0	0
paw_paw	0	0	1	0	0
family	0	0	0	0	1

The Problem with High Dimensionality

But if we could collapse all of these into one “meta-dimension”…

	tasty	delicious	disgusting	flavorful	tree
pear	0	1	0	0	0
apple	0	0	0	1	1
watermelon	1	0	0	0	0
paw_paw	0	0	1	0	0
family	0	0	0	0	1

The Problem with High Dimensionality

Now, these things have "taste" associated with them as a concept

	$<taste>$	tree
pear	1	0
apple	1	1
watermelon	1	0
paw_paw	1	0
family	0	1

Curse of Dimensionality

- Vector representations are sparse, very high dimensional
  - # of words in vocabulary
  - # of relations × # words, etc
- Google 1T 5-gram corpus:
  - In bigram 1M × 1M matrix: < 0.05% non-zero values
- Computationally hard to manage
  - Lots of zeroes
  - Can miss underlying relations

Reducing Dimensionality

- Can we use ***fewer*** features to build our matrices?
- Ideally with
  - **High** ***frequency*** — means fewer zeroes in our matrix
  - **High** ***variance*** — larger spread over values makes items easier to separate

Reducing Dimensionality

- One approach — ***filter*** out features
  - Can exclude terms with too few occurrences
  - Can include only top X most frequently seen features
  - χ^2 selection

Reducing Dimensionality

- Things to watch out for:
  - Feature correlation — if features strongly correlated, give redundant information
  - Joint feature selection complex, computationally expensive

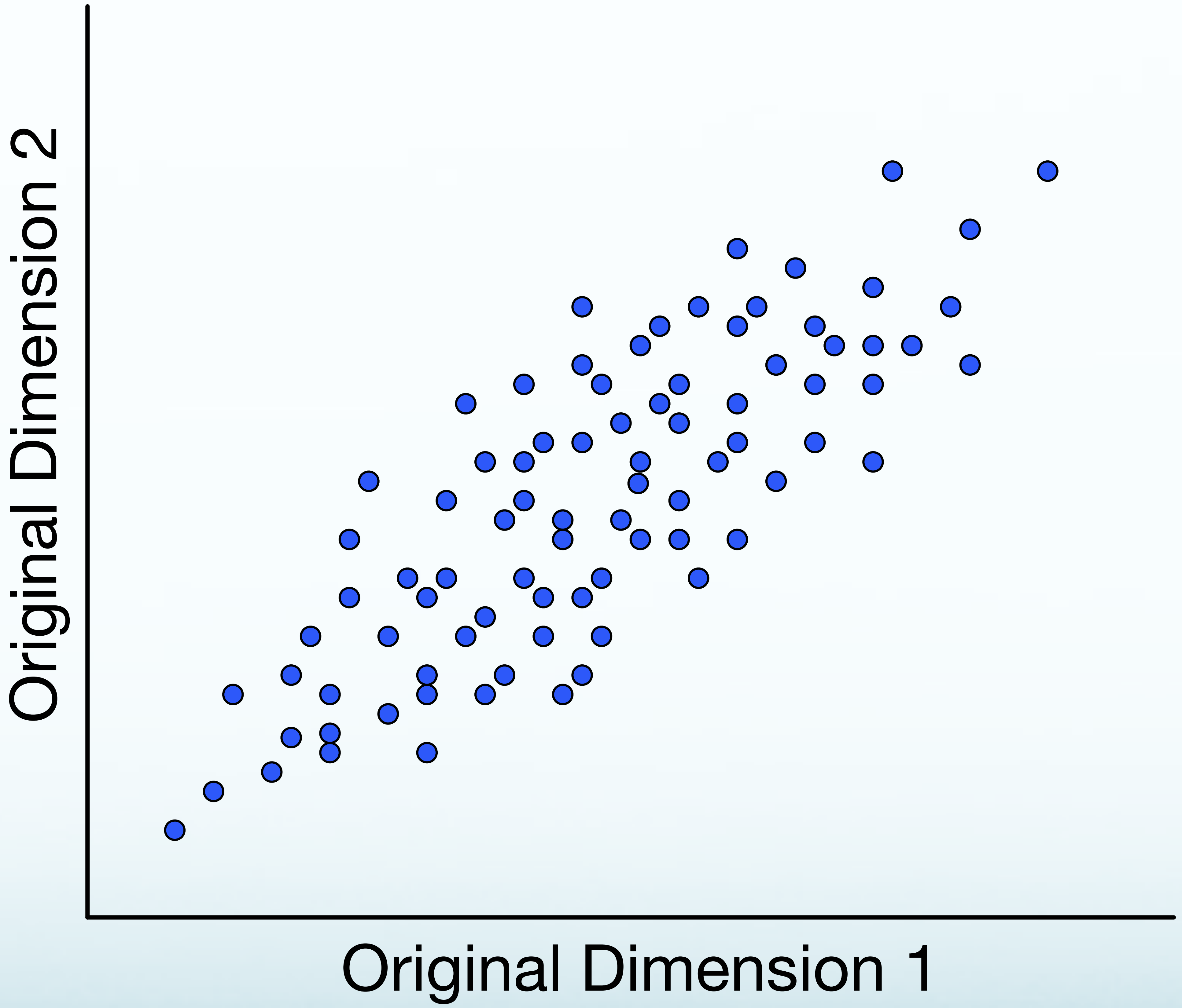
Reducing Dimensionality

- Approaches to project into lower-dimensional spaces
  - Principal Components Analysis (PCA)
  - Locality Preserving Projections (LPP) [link]
  - Singular Value Decomposition (SVD)

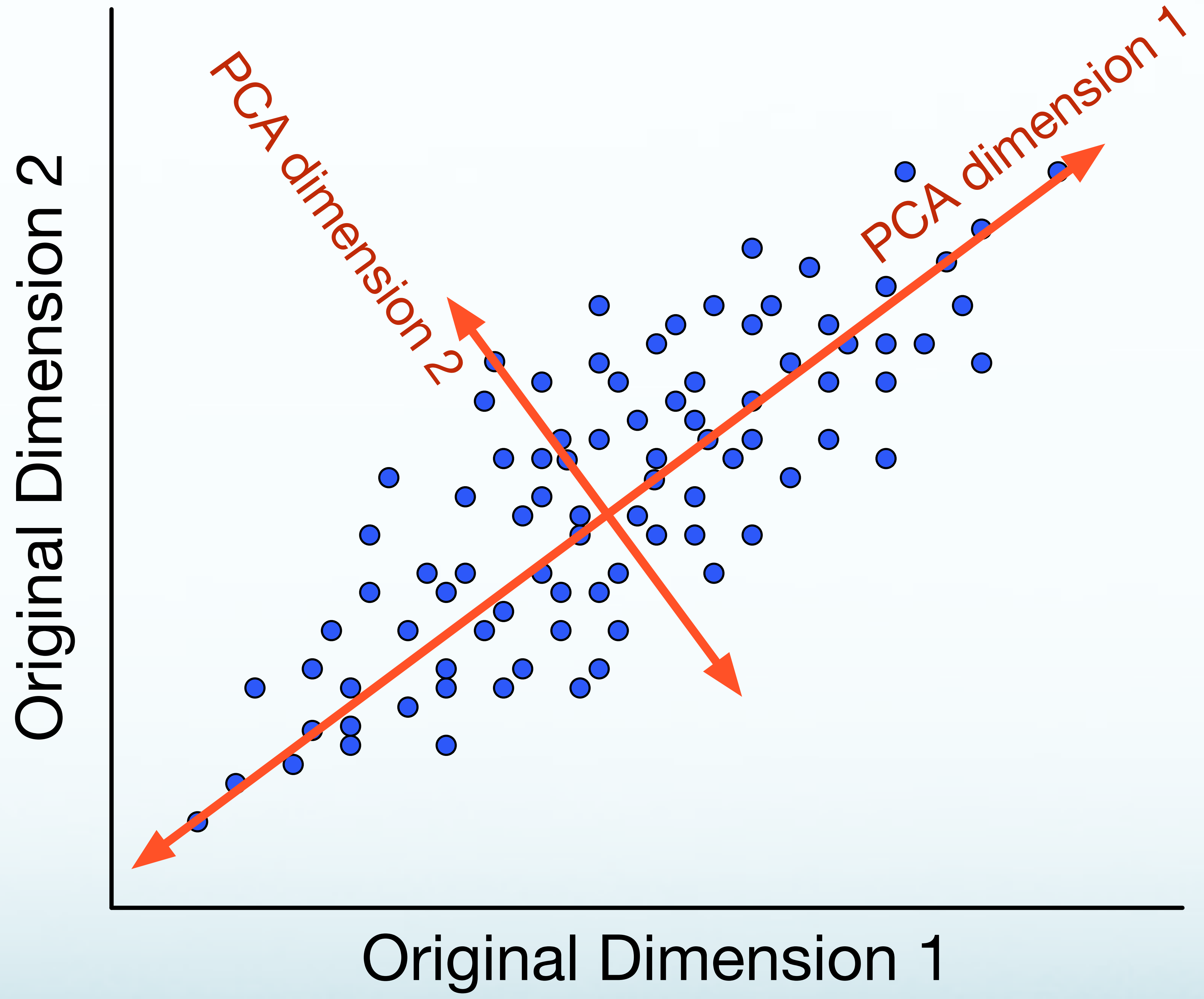
Reducing Dimensionality

- All approaches create new lower dimensional space that
  - Preserves distances between data points
    - (Keep like with like)
- Approaches differ on exactly what is preserved

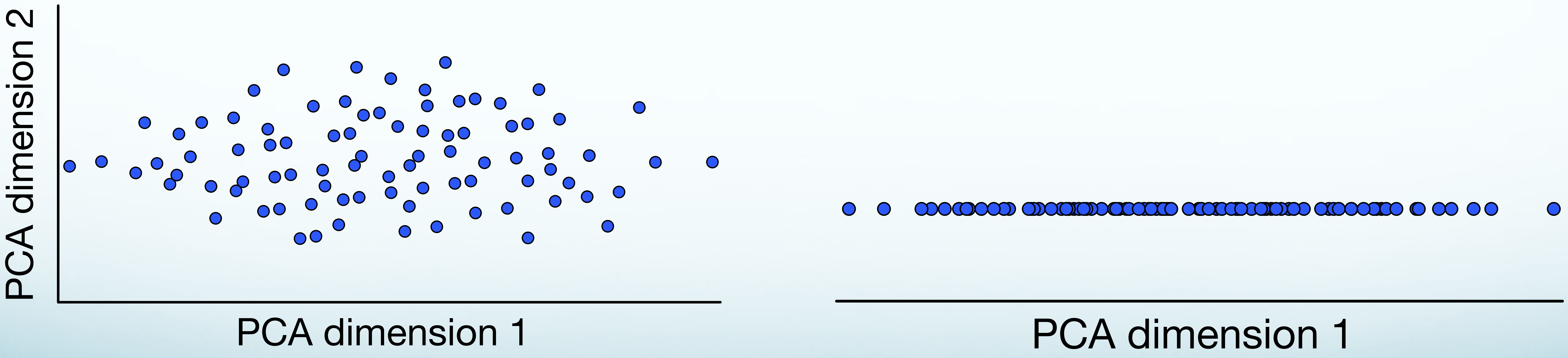
Principal Component Analysis (PCA)



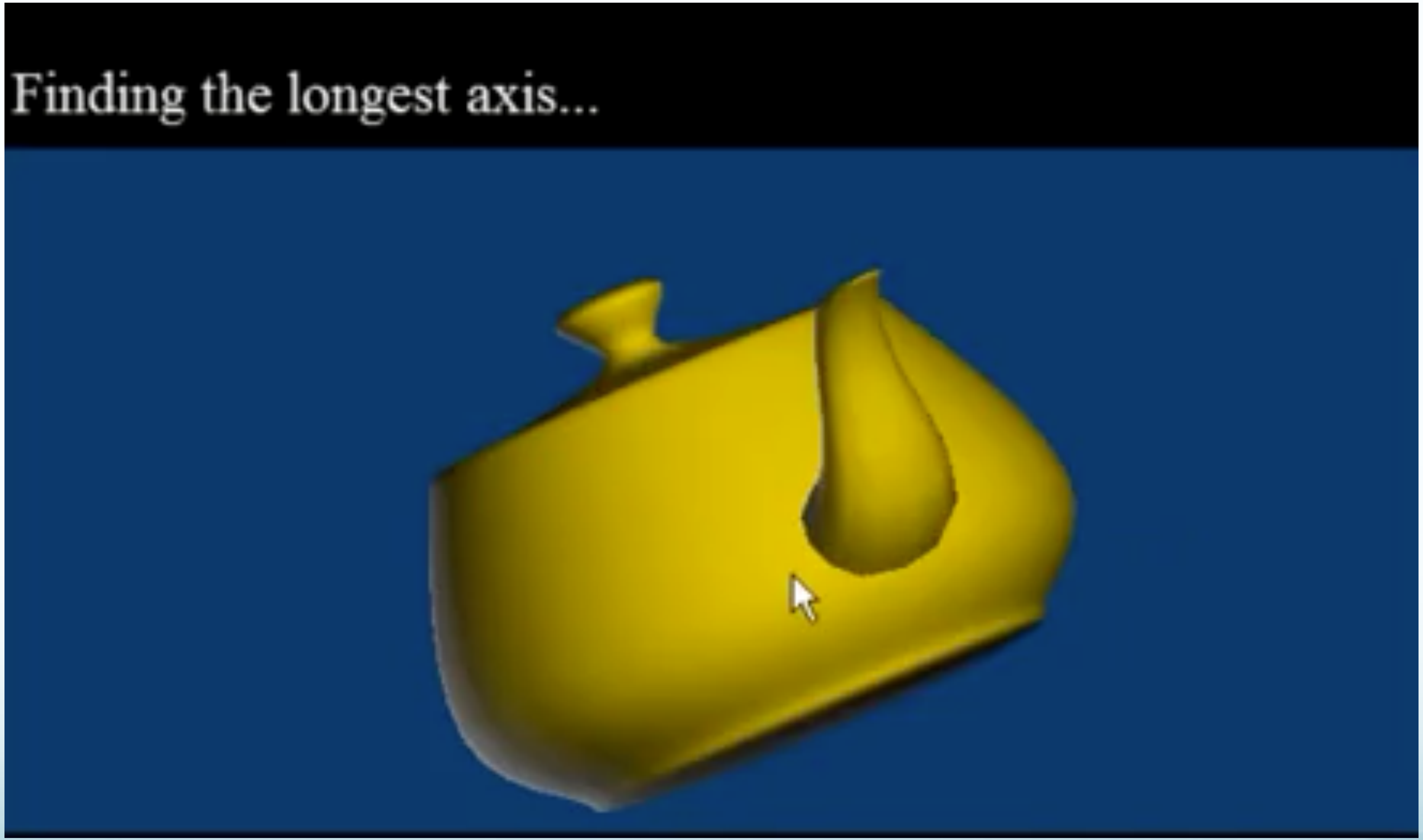
Principal Component Analysis (PCA)



Principal Component Analysis (PCA)



Principal Component Analysis (PCA)



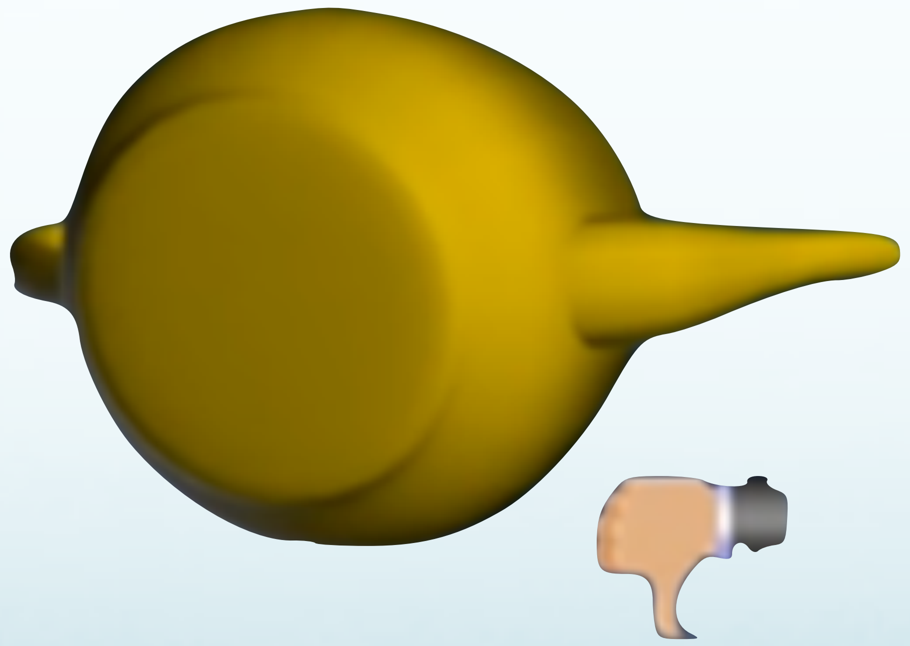
via [A layman's introduction to PCA]

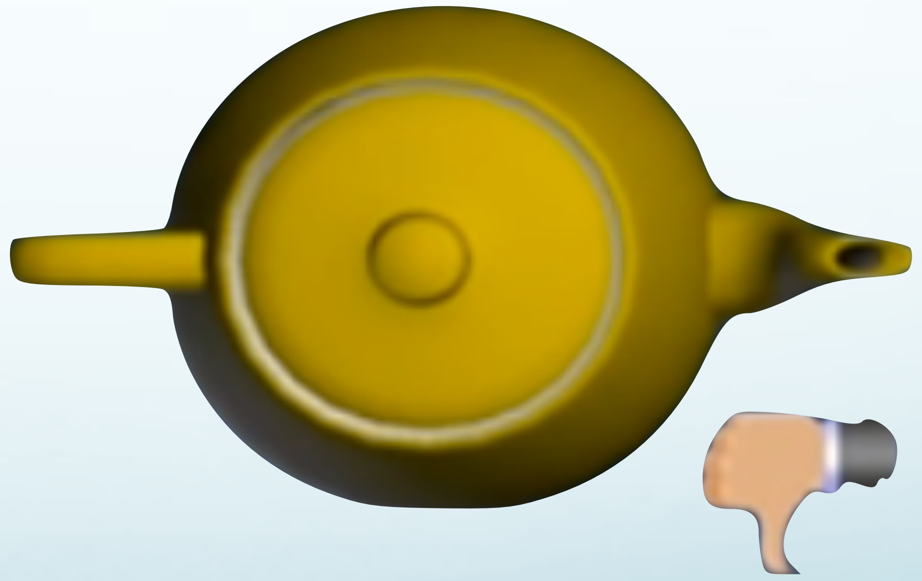
Principal Component Analysis (PCA)

This →

Preserves more information than

These →





via [A layman's introduction to PCA]

Singular Value Decomposition (SVD)

- Enables creation of reduced dimension model
  - Low rank approximation of of original matrix
  - Best-fit at that rank (in least-squares sense)

Singular Value Decomposition (SVD)

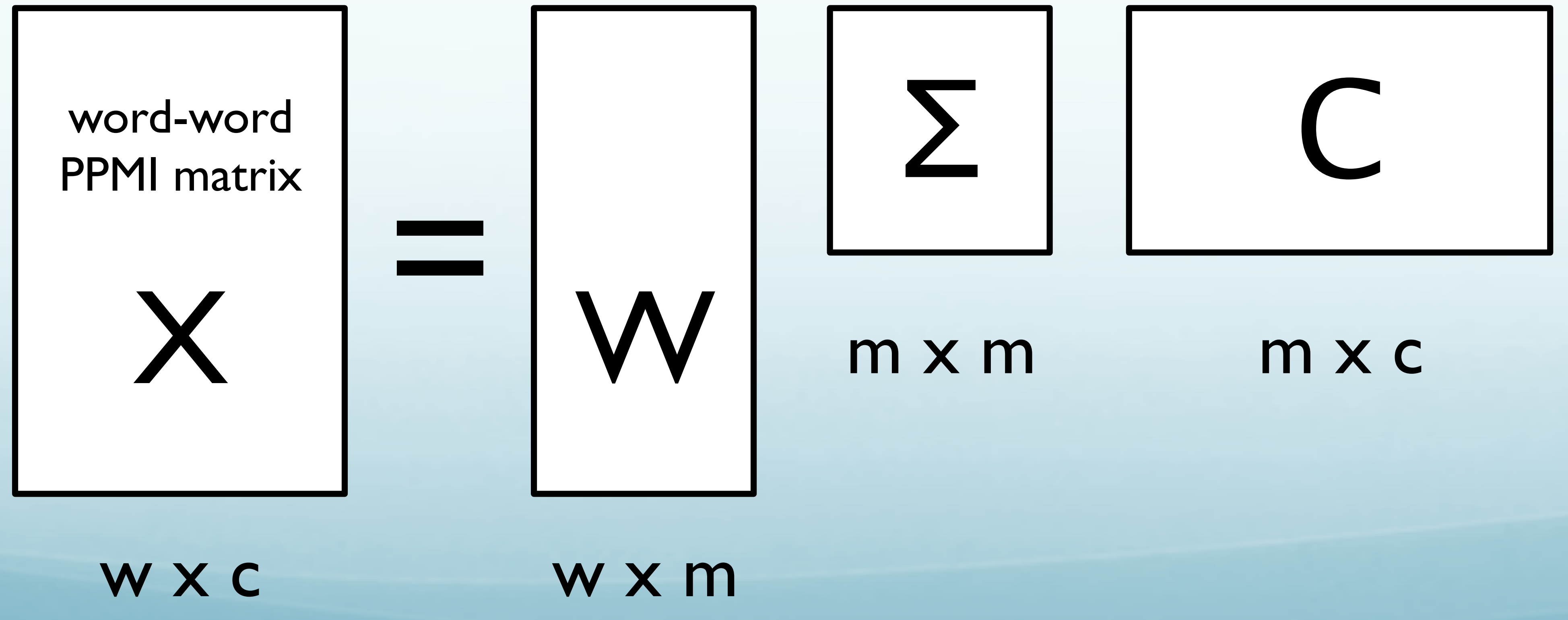
- Original matrix: high dimensional, sparse
  - Similarities missed due to word choice, etc
- Create new, projected space
  - More compact, better captures important variation
- Landauer et al (1998) argue identifies underlying "concepts"
  - Across words with related meanings

Latent Semantic Analysis (LSA)

- Apply SVD to $|V| \times c$ term-document matrix X
  - $V \rightarrow$ Vocabulary
  - $c \rightarrow$ documents
  - X
    - *row* → word
    - *column* → document
    - *cell* → count of word/document

Latent Semantic Analysis (LSA)

- Factor X into three new matrices:
  - $W \rightarrow$ one row per word, but columns are now arbitrary m dimensions
  - $\Sigma \rightarrow$ Diagonal matrix, where every (1,1) (2,2) etc… is the *rank* for m
  - $C^T \rightarrow$ arbitrary m dimensions, as spread across c documents



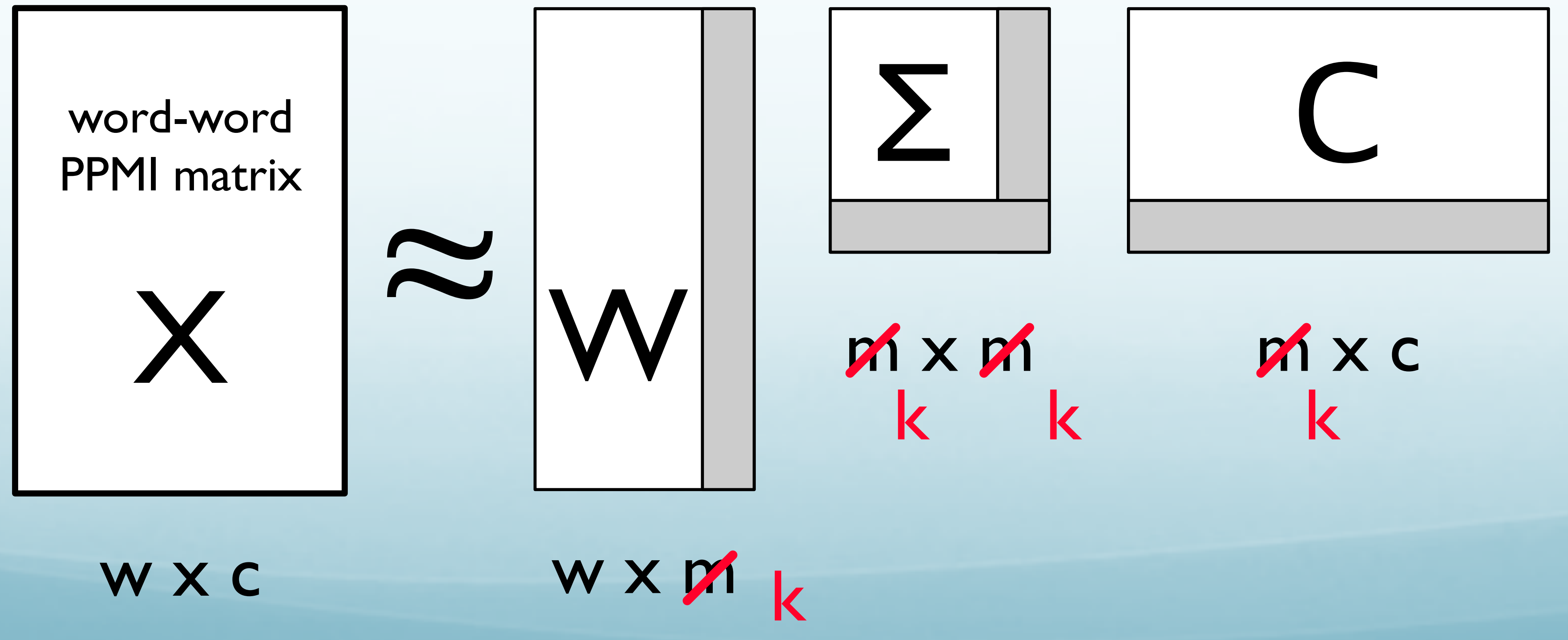
SVD Animation

youtu.be/R9UoFyqJca8

Enjoy some 3D Graphics from 1976!

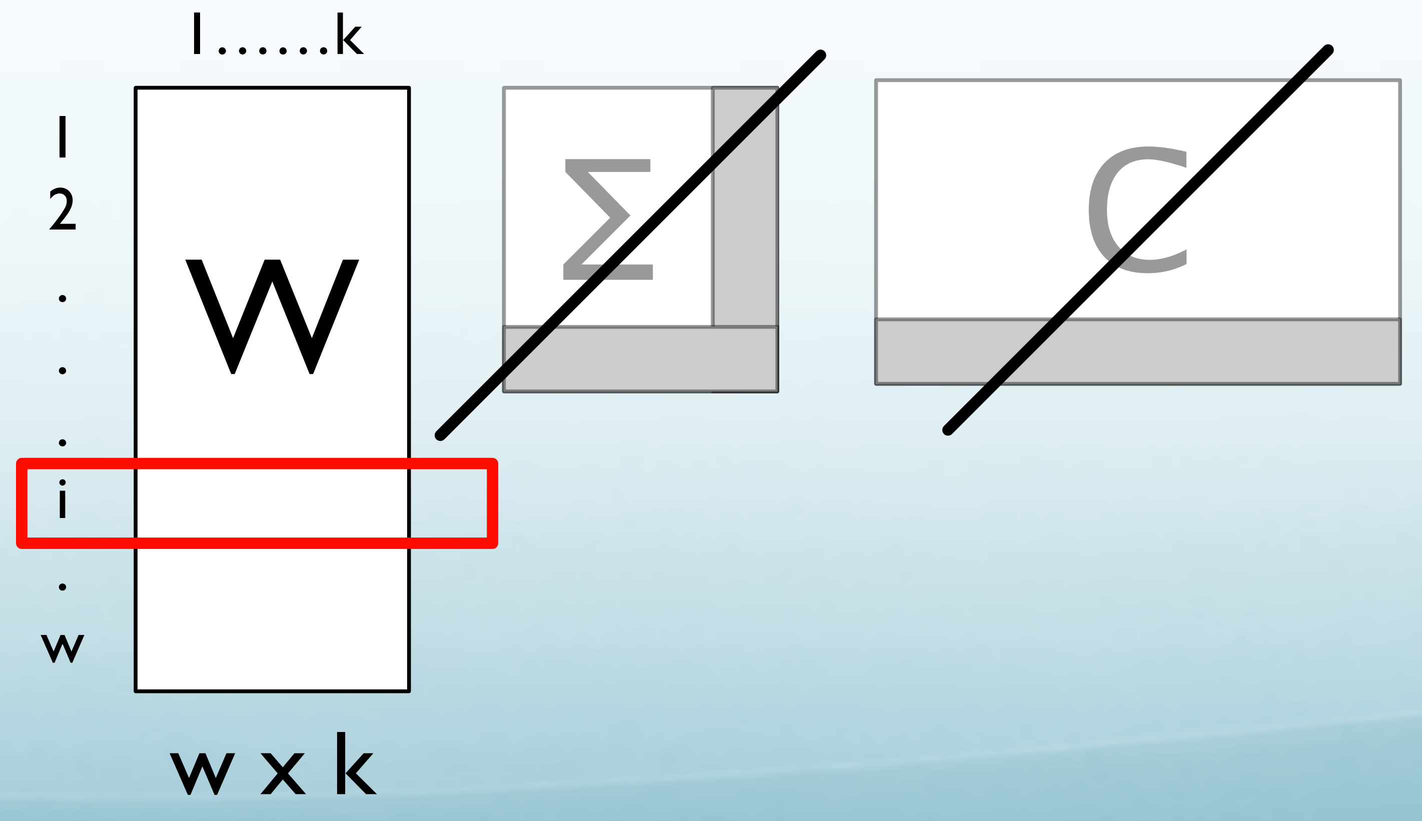
Latent Semantic Analysis (LSA)

- LSA implementations typically:
  - ***truncate*** initial m dimensions to top k



Latent Semantic Analysis (LSA)

- LSA implementations typically:
  - ***truncate*** initial m dimensions to top k
  - then ***discard*** Σ and C matrices
    - Leaving matrix W
    - Each row is now an "embedded" representation of each w across k dimensions



Singular Value Decomposition (SVD)

Original Matrix X (zeroes blank)

	Avengers	Star Wars	Iron Man	Titanic	The Notebook
User1	1	1	1		
User2	3	3	3		
User3	4	4	4		
User4	5	5	5		
User5		2		4	4
User6				5	5
User7		1		2	2

Singular Value Decomposition (SVD)

$W \ (w \times m)$

	m1	m2	m3
User1	0.13	0.02	-0.01
User2	0.41	0.07	-0.03
User3	0.55	0.09	-0.04
User4	0.68	0.11	-0.05
User5	0.15	-0.59	0.65
User6	0.07	-0.73	-0.67
User7	0.07	-0.29	-0.32

$\Sigma \ (m \times m)$

	m1	m2	m3
m1	12.4		
m2		9.5	
m3			1.3

$C \ (m \times c)$

	Avengers	Star Wars	Iron Man	Titanic	The Notebook
m1	0.56	0.59	0.56	0.09	0.09
m2	0.12	-0.02	0.12	-0.69	-0.69
m3	0.40	-0.80	0.40	0.09	0.09

Singular Value Decomposition (SVD)

$\boldsymbol{W}\ (w \times m)$

	m1	m2	m3
User1	0.13	0.02	-0.01
User2	0.41	0.07	-0.03
User3	0.55	0.09	-0.04
User4	0.68	0.11	-0.05
User5	0.15	-0.59	0.65
User6	0.07	-0.73	-0.67
User7	0.07	-0.29	-0.32

$\boldsymbol{\Sigma}\ (m \times m)$

	m1	m2	m3
m1	12.4		
m2		9.5	
m3			1.3

"Sci-fi-ness"

$\boldsymbol{C}\ (m \times c)$

	Avengers	Star Wars	Iron Man	Titanic	The Notebook
m1	0.56	0.59	0.56	0.09	0.09
m2	0.12	-0.02	0.12	-0.69	-0.69
m3	0.40	-0.80	0.40	0.09	0.09

Singular Value Decomposition (SVD)

W $(w \times m)$

	m1	m2	m3
User1	0.13	0.02	-0.01
User2	0.41	0.07	-0.03
User3	0.55	0.09	-0.04
User4	0.68	0.11	-0.05
User5	0.15	-0.59	0.65
User6	0.07	-0.73	-0.67
User7	0.07	-0.29	-0.32

Σ $(m \times m)$

	m1	m2	m3
m1	12.4		
m2		9.5	
m3			1.3

"Romance-ness"

C $(m \times c)$

	Avengers	Star Wars	Iron Man	Titanic	The Notebook
m1	0.56	0.59	0.56	0.09	0.09
m2	0.12	-0.02	0.12	-0.69	-0.69
m3	0.40	-0.80	0.40	0.09	0.09

Singular Value Decomposition (SVD)

W $(w \times m)$

	m1	m2	m3
User1	0.13	0.02	-0.01
User2	0.41	0.07	-0.03
User3	0.55	0.09	-0.04
User4	0.68	0.11	-0.05
User5	0.15	-0.59	0.65
User6	0.07	-0.73	-0.67
User7	0.07	-0.29	-0.32

Σ $(m \times m)$

	m1	m2	m3
m1	12.4		
m2		9.5	
m3			1.3

Catchall (noise)

C $(m \times c)$

	Avengers	Star Wars	Iron Man	Titanic	The Notebook
m1	0.56	0.59	0.56	0.09	0.09
m2	0.12	-0.02	0.12	-0.69	-0.69
m3	0.40	-0.80	0.40	0.09	0.09

LSA Document Contexts

- Deerwester et al, 1990: "*Indexing by Latent Semantic Analysis*"
  - Titles of scientific articles

c1	***Human*** machine ***interface*** for ABC ***computer*** applications
c2	A ***survey*** of ***user*** opinion of ***computer system response time***
c3	The ***EPS user interface*** management *system*
c4	***System*** and ***human system*** engineering testing of ***EPS***
c5	Relation of ***user*** perceived ***response time*** to error measurement
m1	The generation of random, binary, ordered ***trees***
m2	The intersection ***graph*** of paths in ***trees***
m3	***Graph minors*** IV: Widths of ***trees*** and well-quasi-ordering
m4	***Graph minors***: A ***survey***

Document Context Representation

- Term x document:
  - $corr$(human, user) $=$ -0.38; $corr$(human, minors)$=$-0.29

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Improved Representation

- Reduced dimension projection:
  - $corr(\text{human, user}) = 0.98; \quad corr(\text{human, minors}) = -0.83$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.05	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.33	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

Python Tutorial for LSA

- For those interested in seeing how LSA works in practice:
  - technowiki.wordpress.com/2011/08/27/latent-semantic-analysis-lsa-tutorial/

Prediction-Based Models

Prediction-based Embeddings

- LSA models: good, but expensive to compute
- *Skip-gram* and *Continuous Bag of Words* (CBOW) models
- Intuition:
  - Words with similar meanings share similar contexts
  - Train language models learn to predict context words
  - Models train embeddings that make current word more like nearby words and less like distance words
  - Provably related to PPMI models under SVD

Embeddings: Skip-Gram vs. Continuous Bag of Words

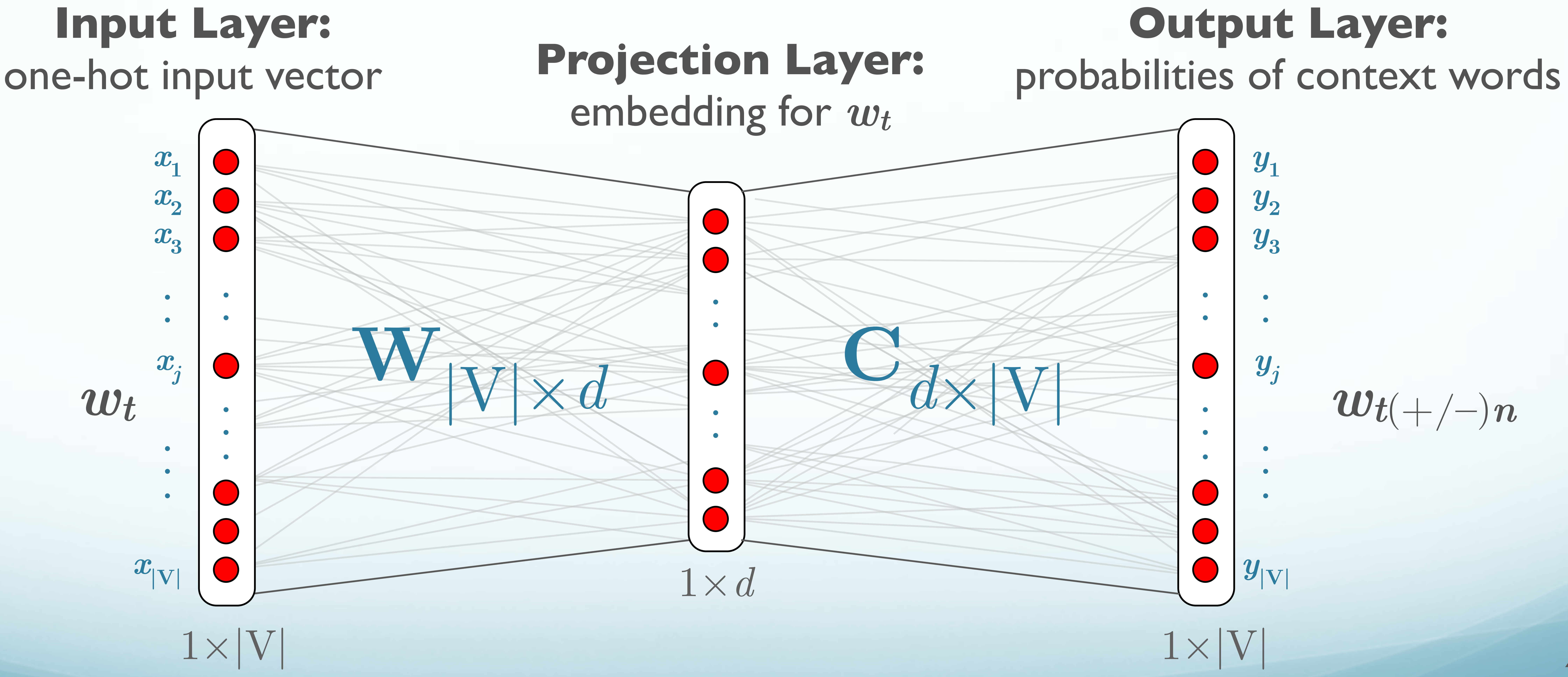
- Continuous Bag of Words (CBOW):
  - $P(\boldsymbol{word} \mid \boldsymbol{context})$
  - Input: $(w_{t\text{-}1},\ w_{t\text{-}2},\ w_{t+1},\ wt_{+2}\ \ldots)$
  - Output: $p(\boldsymbol{w_t})$
- Skip-gram:
  - $P(\boldsymbol{context} \mid \boldsymbol{word})$
  - Input: $\boldsymbol{w_t}$
  - Output: $p(w_{t\text{-}1},\ w_{t\text{-}2},\ w_{t+1},\ wt_{+2}\ \ldots)$

Skip-Gram Model

- Learns two embeddings
  - W : word
  - C : context of some fixed dimension
- Prediction task:
  - Given a word, predict each neighbor word in window
  - Compute $p(w_k|w_j)$ represented as $c_k \cdot v_j$
    - For each context position
  - Convert to probability via softmax

$$p(w_k|w_j) = \frac{\exp(c_k \cdot v_j)}{\sum_{i \in |V|} \exp(c_i \cdot v_j)}$$

Skip-Gram Network Visualization



Training The Model

- Issue:
  - Denominator computation is very expensive
- Strategy:
  - Approximate by negative sampling:
    - + example: true context
    - – example: k other words

Training The Model

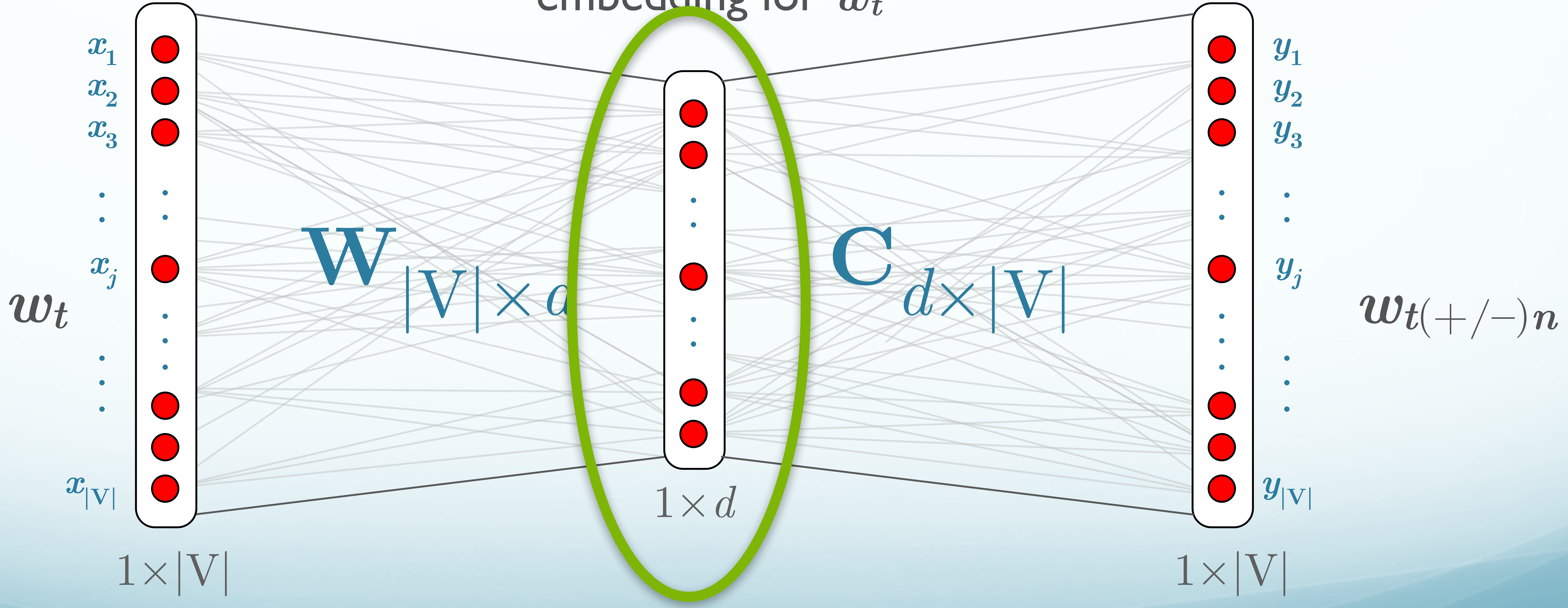
- Approach:
  - Randomly initialize W, C
  - Iterate over corpus, update w/stochastic gradient descent
  - Update embeddings to improve loss function
- Use trained embeddings directly as word representations

Skip-Gram Network Visualization

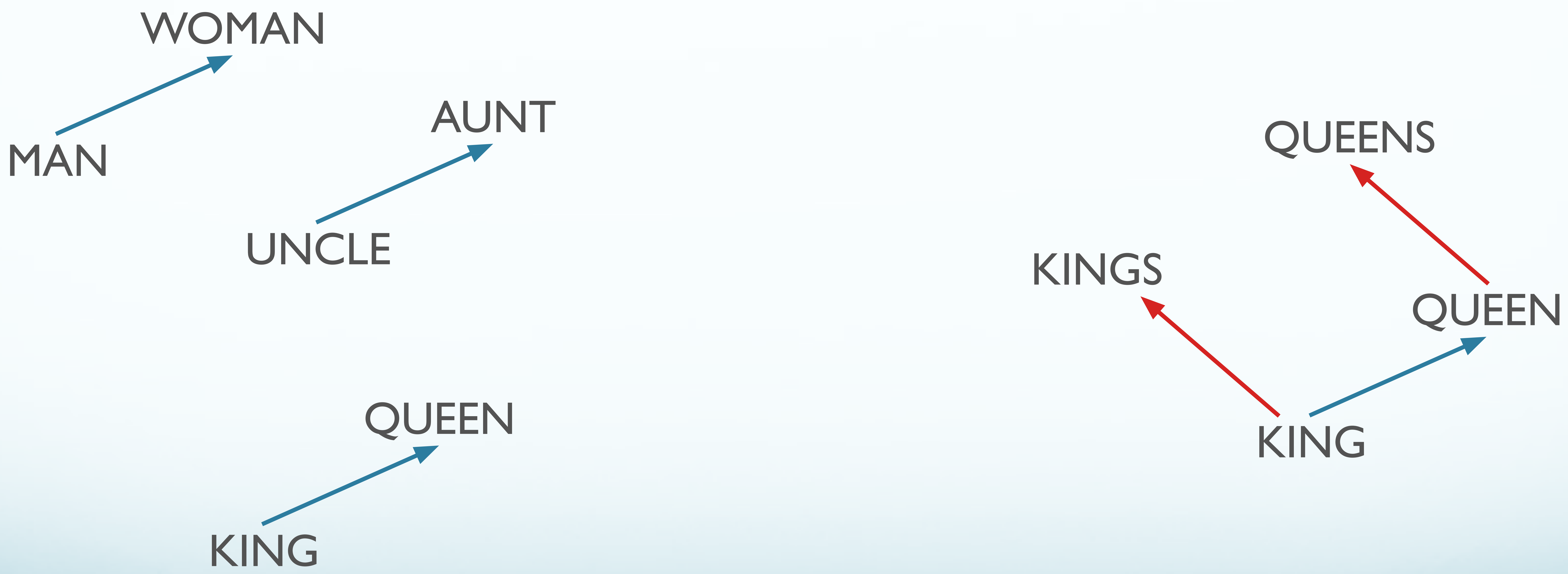
Input Layer:
one-hot input vector

Projection Layer:
embedding for w_t

Output Layer:
probabilities of context words



Relationships via Offsets



Diverse Applications

- Unsupervised POS tagging
- Word Sense Disambiguation
- Essay Scoring
- Document Retrieval
- Unsupervised Thesaurus Induction
- Ontology/Taxonomy Expansion
- Analogy Tests, Word Tests
- Topic Segmentation

Distributional Similarity for Word Sense Disambiguation

Distributional Similarity for WSD

- So, how do we attempt to compute homonymy?

There are more kinds of **plants** and animals in the rainforests than anywhere else on Earth. Over half of the millions of known species of **plants** and animals live in the rainforest. Many are found nowhere else. There are even **plants** and animals in the rainforest that we have not yet discovered.

Biological Example

The Paulus company was founded in 1938. Since those days the product range has been the subject of constant expansions and is brought up continuously to correspond with the state of the art. We're engineering, manufacturing and commissioning world-wide ready-to-run **plants** packed with our comprehensive know-how. Our Product Range includes pneumatic conveying systems for carbon, carbide, sand, lime and many others. We use reagent injection in molten metal for the…

Industrial Example

Label the First Use of "Plant"

Word Representation

- 2^{nd} Order Representation:
  - Identify words in context of w
  - For each x in context of w:
    - Compute x vector representation
  - Compute centroid of these $\vec{x}$ vector representations

Computing Word Senses

- Compute context vector for each occurrence of word in corpus
- Cluster these context vectors
  - # of clusters = # of senses
- Cluster centroid represents word sense
- Link to specific sense?
  - Pure unsupervsed: no sense tag, just i^{th} sense
  - Some supervision: hand label clusters, or tag training

Disambiguating Instances

- To disambiguate an instance *t* of *w*:
  - Compute context vector for instance
  - Retrieve all senses of *w*
  - Assign *w* sense with closest centroid to *t*

Example Sense Selection for Plant Data

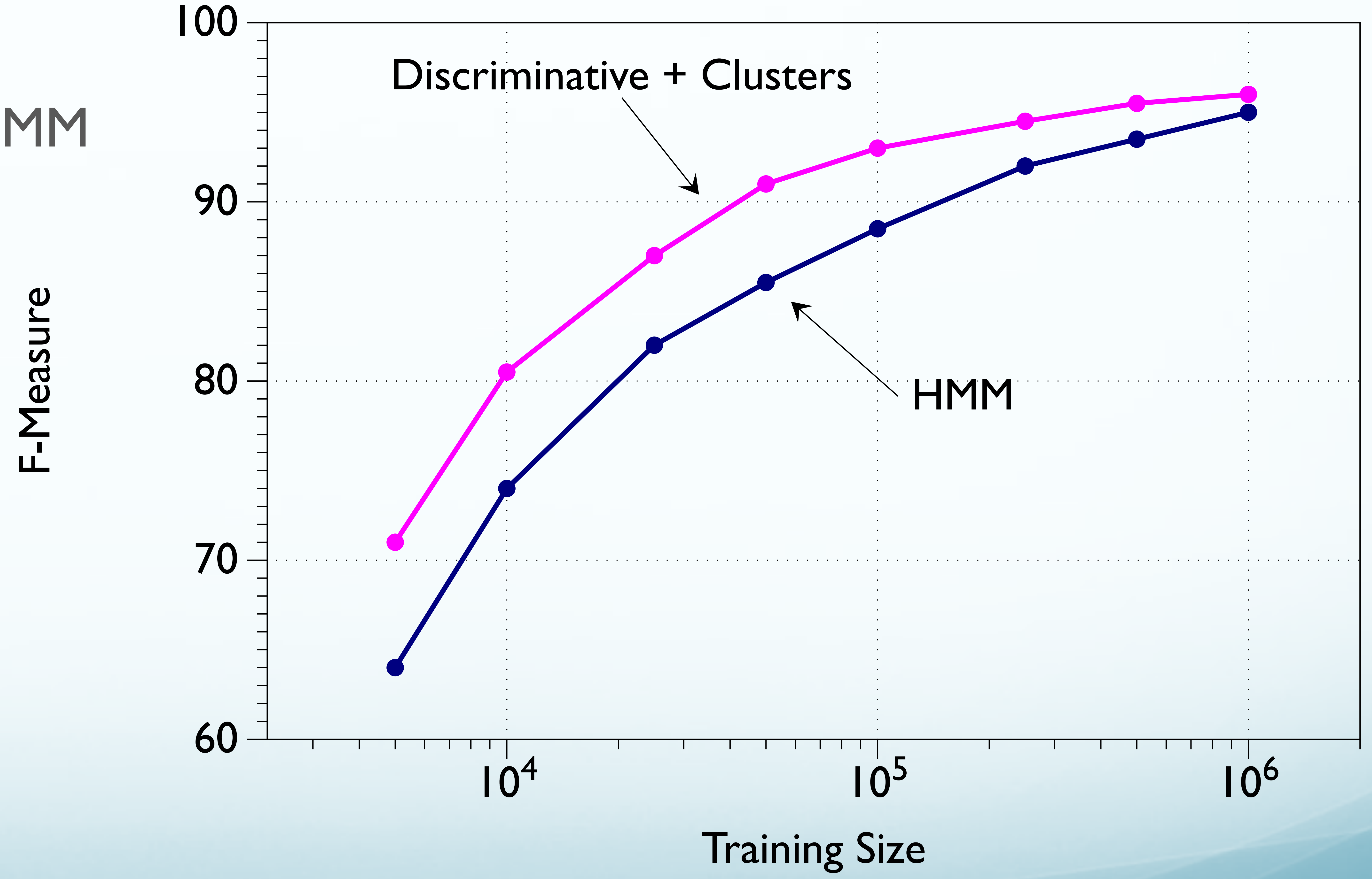
- Build a Context Vector
  - 1,001 character window - Whole Article
- Compare Vector Distances to Sense Clusters
  - Only 3 content words in common
  - Distance context vectors
  - Clusters - build automatically, label manually
- Result: 2 different, correct senses
  - 92% on pairwise tasks

Local Context Clustering

- "Brown" (aka IBM) clustering (1992)
  - Generative model over adjacent words
  - Each w_i has class c_i
    - $\log P(W) = \sum_i \log P(w_i|c_i) + \log P(c_i|c_{i-1})$
  - Greedy clustering
    - Start with each word in own cluster
    - Merge clusters based on log prob of text under model
      - Merge those which maximize $P(W)$

Clustering Impact

- Improves downstream tasks
  - Named Entity Recognition vs. HMM
    - Miller et al '04



Distributional Models: Summary

- Upsurge in distributional compositional
  - Embeddings:
    - Discriminatively trained, low dimensional representations
    - e.g. word2vec
      - skipgrams, etc. over large corpora
  - Composition:
    - Methods for combining word vector models
    - Capture phrasal, sentential meanings

Exercise!

Let's Make Some Data!

Human Word Similarity Judgments

To complete the survey, go to **PollEv.com/rgeorgi**

0 surveys done

1 survey underway

Start the presentation to see live content. Still no live content? Install the app or get help at **PollEv.com/app**