

TTIC 31190: Natural Language Processing

Kevin Gimpel

Winter 2016

Lecture 5: Word Vectors

- Assignment 1 now due Thursday 11:59pm
- Assignment 2 will be assigned on Thursday, due Tuesday, Feb. 2nd

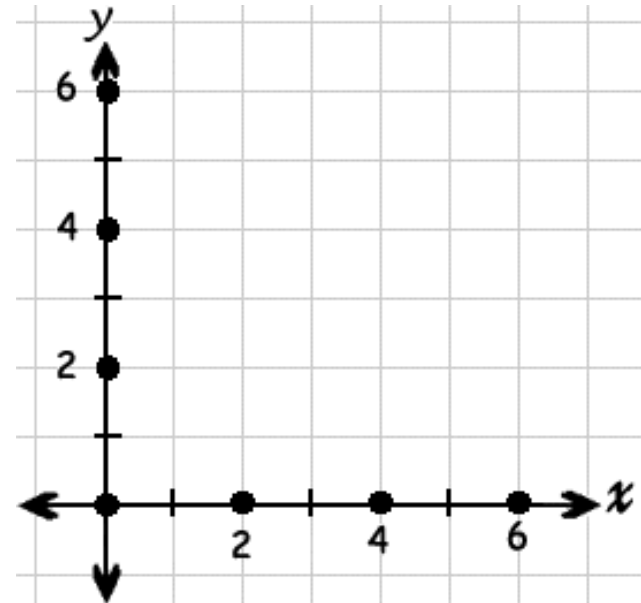
Homonymy or Polysemy?

axes

an edge tool with a heavy bladed head mounted across a handle



a fixed reference line for the measurement of coordinates



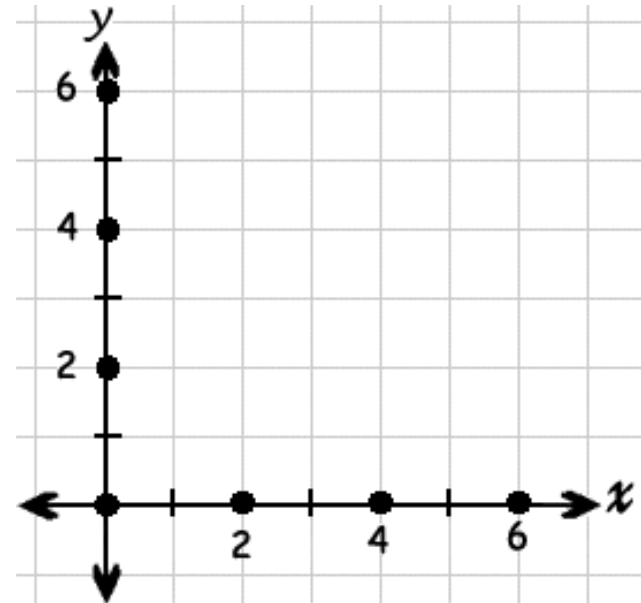
Homonymy or Polysemy?

axes

an edge tool with a heavy bladed head mounted across a handle



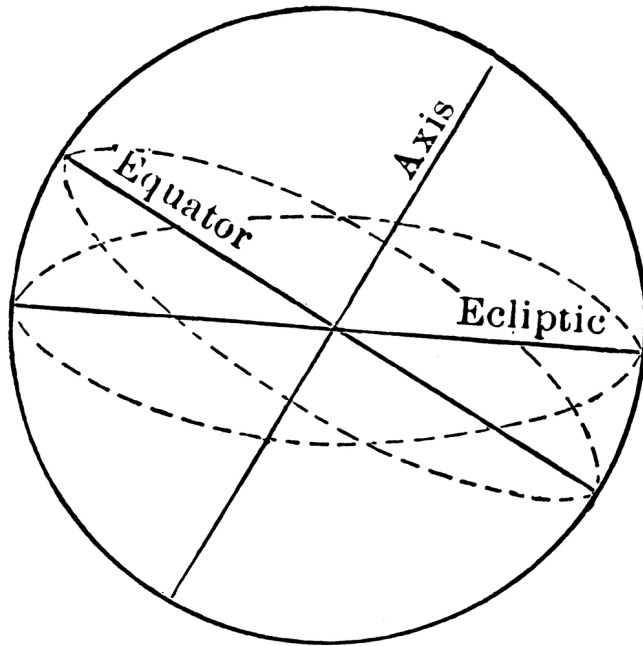
a fixed reference line for the measurement of coordinates



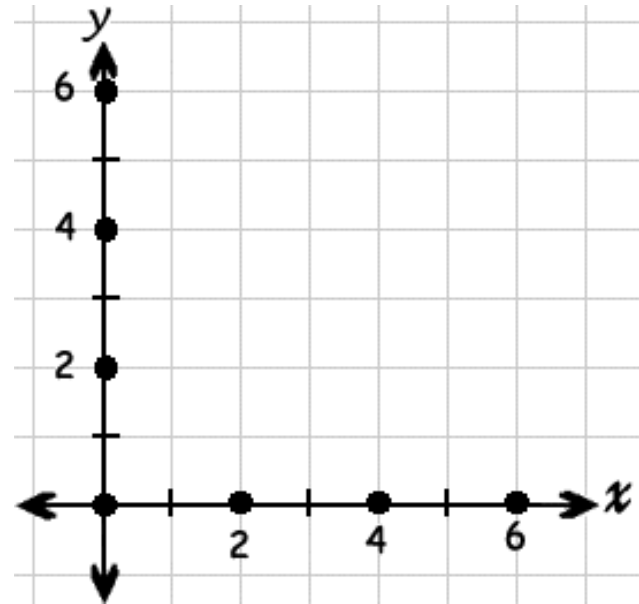
Homonymy or Polysemy?

axes

an imaginary line about which a body rotates



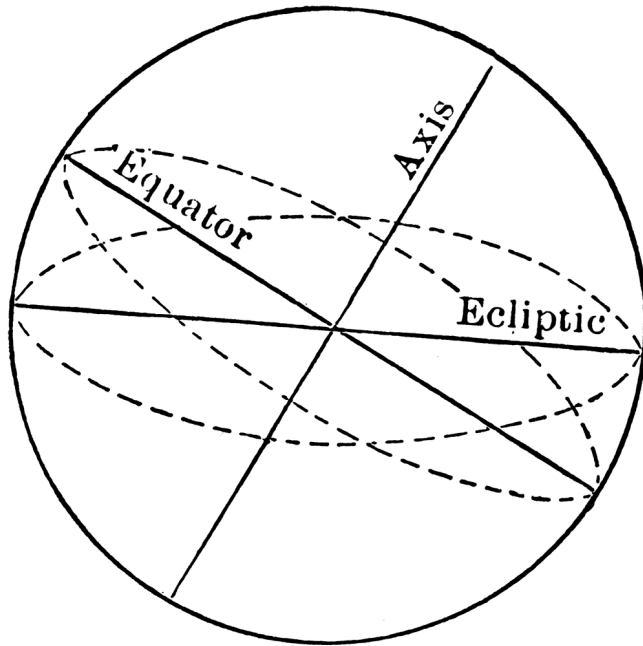
a fixed reference line for the measurement of coordinates



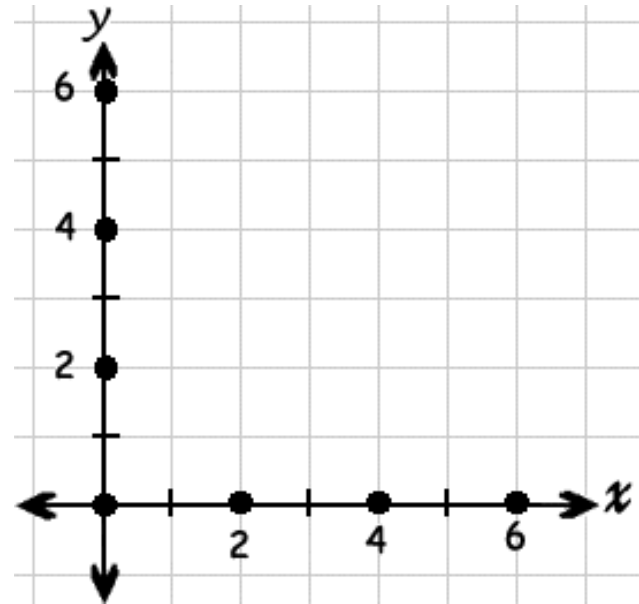
Homonymy or **Polysemy**?

axes

an imaginary line about which a body rotates



a fixed reference line for the measurement of coordinates

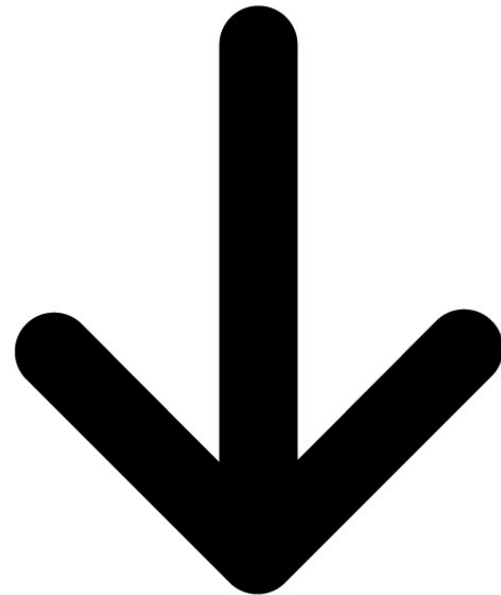
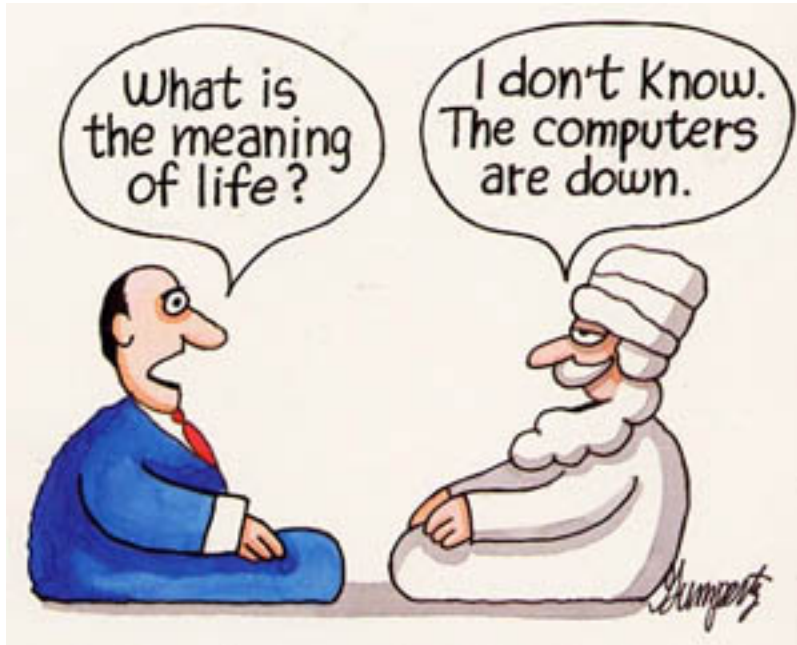


Homonymy or Polysemy?

down

in an inactive or inoperative state

being or moving lower in position or less in some value

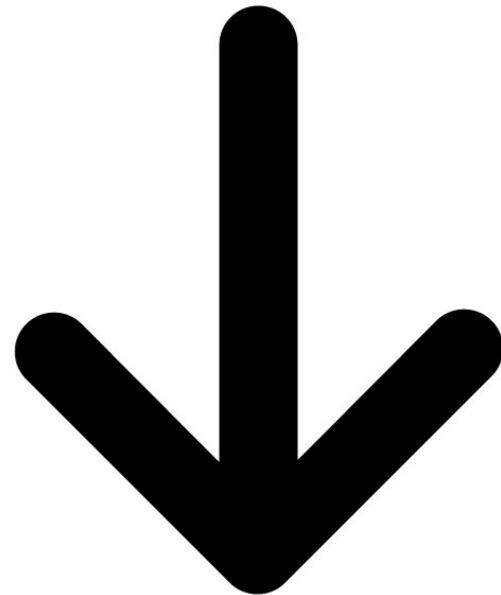
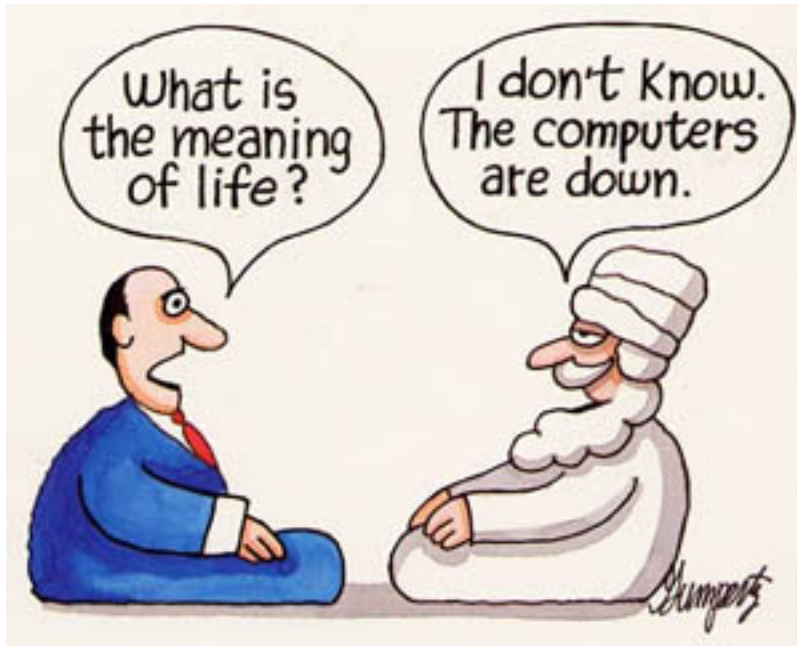


Homonymy or **Polysemy**?

down

in an inactive or inoperative state

being or moving lower in position or less in some value



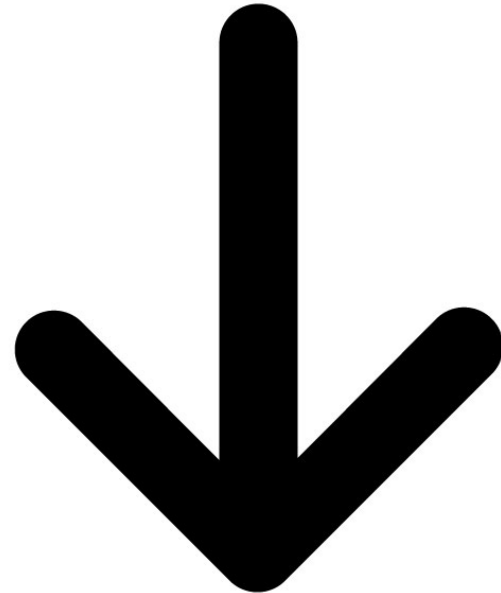
Homonymy or Polysemy?

down

soft fine feathers



being or moving lower in position or less in some value



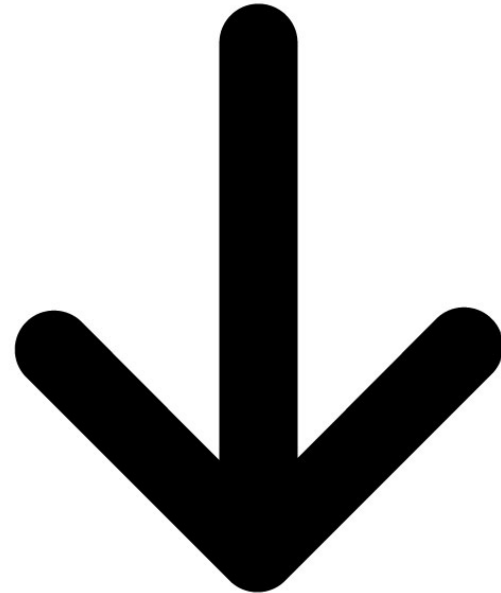
Homonymy or Polysemy?

down

soft fine feathers



being or moving lower in position or less in some value



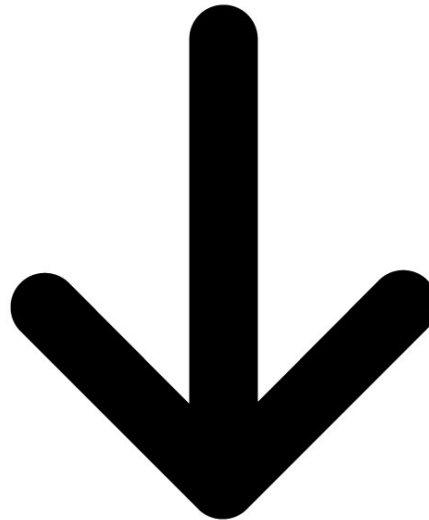
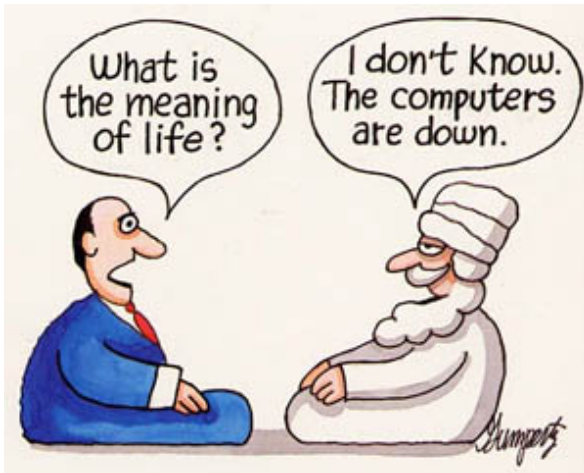
Homonymy or Polysemy?

down

in an inactive or inoperative state

being or moving lower in position or less in some value

unhappy





is a (hyponym/hypernym/meronym/holonym) of





is a (hyponym/hypernym/meronym/**holonym**) of



piano₁



is a

(hyponym/hypernym)

of

instrument₁



piano₁



is a

(**hyponym**/hypernym)

of

instrument₁



- why am I showing you pictures instead of words?
- hypernymy, meronymy, etc. are relationships between **synsets**, not words

Roadmap

- classification
- words
- lexical semantics
 - word sense
 - word vectors
- language modeling
- sequence labeling
- syntax and syntactic parsing
- neural network methods in NLP
- semantic compositionality
- semantic parsing
- unsupervised learning
- machine translation and other applications

Noun

- **S: (n)** [fool](#), [sap](#), [saphead](#), [muggins](#), [tomfool](#) (a person who lacks good judgment)
- **S: (n)** [chump](#), [fool](#), [gull](#), [mark](#), [patsy](#), [fall guy](#), [sucker](#), [soft touch](#), [mug](#) (a person who is gullible and easy to take advantage of)
- **S: (n)** [jester](#), [fool](#), [motley fool](#) (a professional clown employed to entertain a king or nobleman in the Middle Ages)

ambiguity

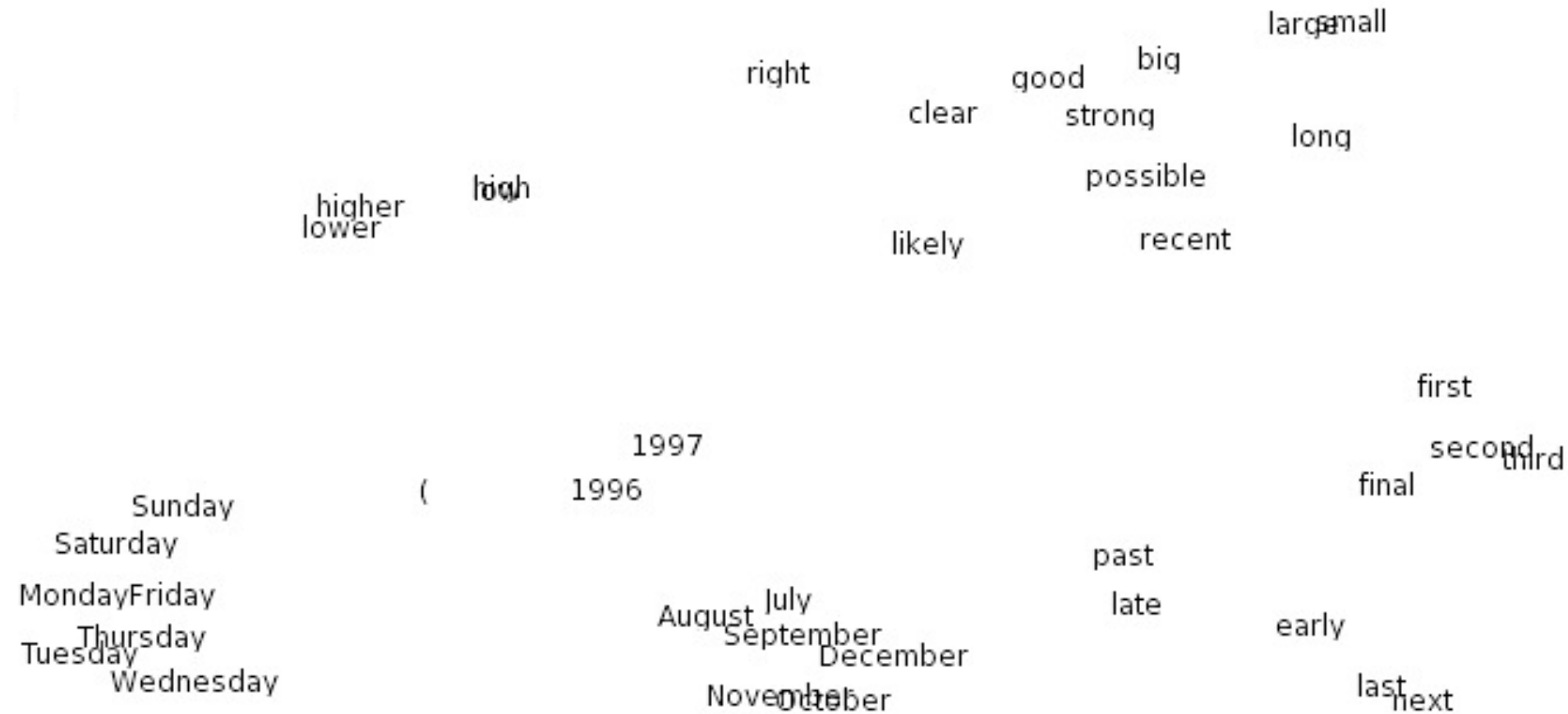
- one form, multiple meanings → split form
 - the three senses of *fool* belong to different synsets

variability

- multiple forms, one meaning → merge forms
 - each synset contains senses of several different words

- are we finished? have we solved the problem of representing word meaning?
- issues:
 - WordNet has limited coverage and only exists for a small set of languages
 - WSD requires training data, whether supervised or seeds for semi-supervised
 - WordNet only tells us *whether* two forms are similar or different, not the amount of similarity/dissimilarity
- better approach: jointly learn representations for all words from raw, unlabeled text

Vector Representations of Words



t-SNE visualization from Turian et al. (2010)

Why vector models of word meaning? computing the similarity between words

tall is similar to *height*

question answering:

*Q: How **tall** is Mt. Everest?*

*A: “The official **height** of Mount Everest is 29029 feet”*

distributional models of meaning
= vector space models of meaning
= vector semantics

Zellig Harris (1954):

- “oculist and eye-doctor ... occur in almost the same environments”
- “If A and B have almost identical environments we say that they are synonyms.”

J.R. Firth (1957):

- “You shall know a word by the company it keeps!”

Warren Weaver (1955):

“But if one lengthens the slit in the opaque mask, until one can see not only the central word in question but also say N words on either side, then **if N is large enough one can unambiguously decide the meaning of the central word...**”



Intuitions of Distributional Models

- suppose I gave you the following corpus:
 - A bottle of **tesgüino** is on the table
 - Everybody likes **tesgüino**
 - Tesgüino** makes you drunk
 - We make **tesgüino** out of corn.
- what is **tesgüino**?
- from context, we can guess **tesgüino** is an alcoholic beverage like beer
- intuition: two words are similar if they have similar word contexts

Many ways to get word vectors

some based on counting, some based on prediction/learning

some sparse, some dense

some have interpretable dimensions, some don't

shared ideas:

model meaning of a word by “embedding” it in a vector space

these word vectors are also called “**embeddings**”

contrast: in traditional NLP, word meaning is represented by a vocabulary index (“word #545”), including in assignment 1!

Roadmap

- count-based word vectors
- dimensionality reduction
- prediction-based word vectors

Distributional Word Vectors

- we'll start with the simplest way to create word vectors:
- count occurrences of context words
 - so, vector for *pineapple* has counts of words in the context of *pineapple* in a dataset
 - one entry in vector for each unique context word
 - stack these vectors for all words in a vocabulary V to produce a count matrix C
 - C is called the **word-context matrix** (or **word-word co-occurrence matrix**)

Counting Context Words

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

- we showed 4x6, but actual matrix is $|V| \times |V|$
 - very large, but very **sparse** (mostly zeroes)
 - lots of efficient algorithms for sparse matrices
 - in your next homework assignment, you will use a smaller vocabulary V_c for the context, so your word-context matrix will be $|V| \times |V_c|$

Context Window Size

- size of context window affects vectors
- one table below uses window size 1 and the other uses window size 10. which is which?
- (think of each row as a cluster):

window size A

Mr. Mrs. Dr. Ms. Prof.

truly wildly politically financially

his your her its

window size B

takeoff altitude airport carry-on

clinic physician doctor medical

financing equity investors firms

Context Window Size

- size of context window affects vectors
- one table below uses window size 1 and the

more syntactic/functional,
same part-of-speech tag

more semantic/topical,
mix of part-of-speech tags

window size **1**

Mr. Mrs. Dr. Ms. Prof.

truly wildly politically financially

his your her its

window size **10**

takeoff altitude airport carry-on

clinic physician doctor medical

financing equity investors firms

Measuring similarity

- given 2 word vectors, how should we measure their similarity?
- most measure of vectors similarity are based on **dot product** (or **inner product**):

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^d u_i v_i$$

– high when vectors have large values in same dimensions

Problem with dot product

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^d u_i v_i$$

- dot product is larger if vector is longer
- vector length:

$$\|\mathbf{u}\| = \sqrt{\sum_{i=1}^d u_i^2}$$

- frequent words \rightarrow larger counts \rightarrow larger dot products
- this is bad: we don't want a similarity metric to be sensitive to word frequency

Solution: cosine similarity

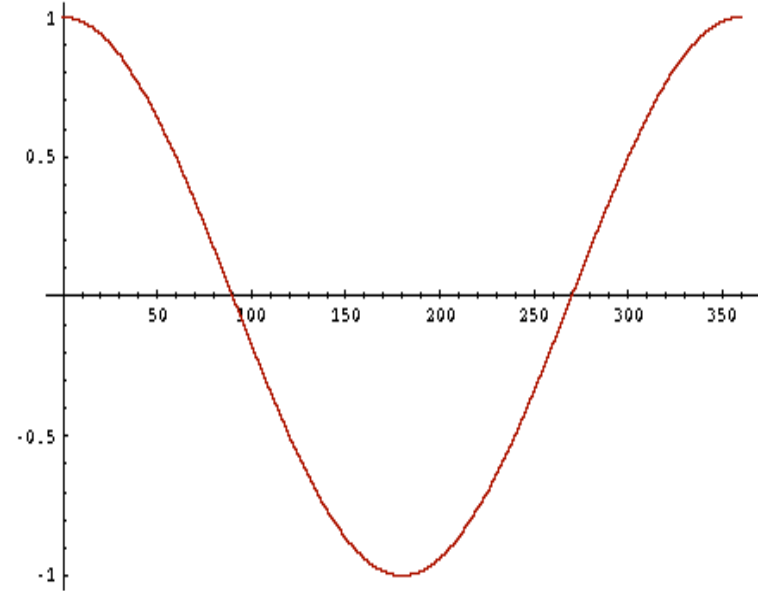
- divide dot product by lengths of the vectors

$$\frac{u \cdot v}{\|u\| \|v\|}$$

- turns out to be the cosine of the angle between them!

Cosine as a similarity metric

- -1: vectors point in opposite directions
 - +1: vectors point in same directions
 - 0: vectors are orthogonal
-
- word counts are non-negative, so cosine ranges from 0 to 1



Problems with raw counts

- raw word counts are not a great measure of association between words
 - why not?
 - very skewed: *the* and *of* are frequent, but not the most discriminative
- rather have a measure that asks whether a context word is **informative** about the center word
 - **positive pointwise mutual information (PPMI)**

Pointwise Mutual Information (PMI)

- do two events x and y co-occur more often than if they were independent?

$$\text{pmi}(x; y) = \log \frac{p(x, y)}{p(x)p(y)}$$

- here, x is the center word and y is the word in the context window
- each of these probabilities can be estimated from the counts collected from the corpus
- replace raw counts with pmi scores

Positive Pointwise Mutual Information (PPMI)

- PMI ranges from $-\infty$ to $+\infty$
- but negative values are problematic:
 - things are co-occurring **less than** we expect by chance
 - unreliable without enormous corpora
- so we just replace negative PMI values by 0, calling it positive PMI (PPMI)

Alternative to PPMI

- **tf-idf**: (that's a hyphen not a minus sign)
- product of two factors:
 - **term frequency** (TF; Luhn, 1957): count of word (or possibly log of count)
 - **inverse document frequency** (IDF; Sparck Jones, 1972)
 - N : total number of documents
 - df_i : # of documents with word i

$$\text{idf}_i = \log \left(\frac{N}{df_i} \right)$$

Roadmap

- count-based word vectors
- dimensionality reduction
- prediction-based word vectors

Sparse versus dense vectors

- so far, our vectors are
 - **long** (length $|V| = 20,000$ to $50,000$)
 - **sparse** (mostly zero)
- why might we want to reduce vector dimensionality?

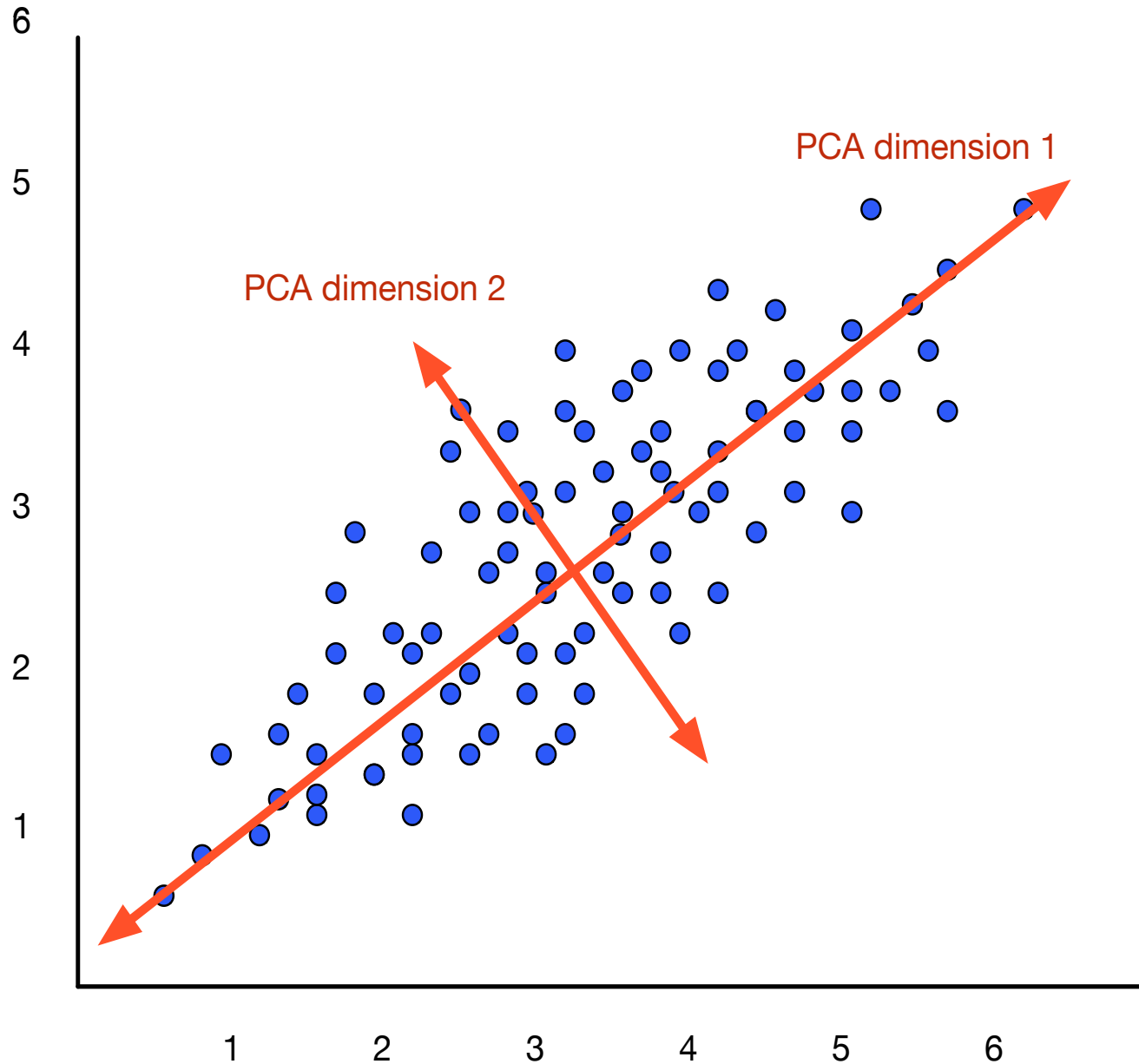
Why reduce dimensionality?

- short vectors may be easier to use as features (fewer weights to tune)
- reducing dimensionality may better handle variability in natural language due to synonymy:
 - *car* and *automobile* are synonyms, but represented as distinct dimensions
 - this fails to capture similarity between a word with *car* as a neighbor and one with *automobile* as a neighbor

Dimensionality Reduction: Intuition

- approximate an N -dimensional dataset using fewer dimensions:
 - rotate axes into a new space
 - in which first dimension captures most variance in original dataset
 - next dimension captures next most variance, etc.
- many such (related) methods:
 - principal component analysis (PCA)
 - factor analysis
 - singular value decomposition (SVD)

Dimensionality reduction



Singular Value Decomposition

SVD is a way to factorize a matrix

any rectangular $w \times c$ matrix X equals the product of 3 matrices:

W: rows match original but each of 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 $m \times m$ matrix of **singular values** expressing the importance of each dimension.

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

SVD applied to word-context matrix

$$\begin{bmatrix} X \\ |V| \times |V| \end{bmatrix} = \begin{bmatrix} W \\ |V| \times |V| \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix} \begin{bmatrix} C \\ |V| \times |V| \end{bmatrix}$$

(simplifying here by assuming the matrix has rank $|V|$)

Truncated SVD on word-context matrix

$$\begin{bmatrix} X \\ |V| \times |V| \end{bmatrix} = \begin{bmatrix} W \\ |V| \times k \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix} \begin{bmatrix} C \\ k \times |V| \end{bmatrix}$$

matrix containing
new k -dimensional
word vectors;
 k might be 50 to 1000

SVD embeddings versus sparse vectors

- dense SVD embeddings sometimes work better than sparse PPMI matrices at tasks like word similarity
 - denoising: low-order dimensions may represent unimportant information
 - truncation may help the models generalize better to unseen data
 - having a smaller number of dimensions may make it easier for classifiers to properly weight the dimensions for the task
 - dense models may do better at capturing higher order co-occurrence

Roadmap

- count-based word vectors
- dimensionality reduction
- prediction-based word vectors

Other ways to learn word vectors

- let's use our classification framework
- we want to use unlabeled text to train the vectors
- we can convert our unlabeled text into a classification problem!
- how? (there are many possibilities)

Other ways to learn word vectors

- aside: any labeled dataset can be used to learn word vectors (depending on model/features)
- how could you use your assignment 1 classifiers to produce word vectors?
- learned feature weights for my 5-way sentiment classifier (binary unigram features), for two words:

feel-good

label	weight
strongly positive	0.025
positive	0.035
neutral	-0.045
negative	0
strongly negative	-0.015

dull

label	weight
strongly positive	0
positive	0
neutral	-0.04
negative	0.015
strongly negative	0.025

Modeling, Inference, Learning for Word Vectors

inference: solve argmax

modeling: define score function

$$\operatorname{classify}(x, \theta) = \operatorname{argmax}_y \operatorname{score}(x, y, \theta)$$

learning: choose θ

- before modeling/inference/learning, we must define (x, y) pairs!
- this isn't text classification, where we had gold standard labels for y
- we have to think of ways to create (x, y) pairs and define the spaces of inputs and outputs

Modeling, Inference, Learning for Word Vectors

inference: solve argmax

modeling: define score function

$$\operatorname{classify}(x, \theta) = \operatorname{argmax}_y \operatorname{score}(x, y, \theta)$$

learning: choose θ

- **skip-gram** (Mikolov et al., 2013):
 - x = a word
 - y = a word in an N -word window of x in a corpus
- **continuous bag-of-words** (CBOW; Mikolov et al., 2013):
 - x = a sequence of N words with the center word omitted
 - y = the center word

Modeling, Inference, Learning for Word Vectors

inference: solve argmax

modeling: define score function

this becomes much more expensive!
(loops over all word types)

learning: choose θ

- **skip-gram** (Mikolov et al., 2013):
 - x = a word
 - y = a word in an N -word window of x in a corpus
- **continuous bag-of-words** (CBOW; Mikolov et al., 2013):
 - x = a sequence of N words with the center word omitted
 - y = the center word

skip-gram training data (window size = 5)

corpus (English Wikipedia):

agriculture is the traditional mainstay of the cambodian economy .

but benares has been destroyed by an earthquake .

...

inputs (x)	outputs (y)
agriculture	<s>
agriculture	is
agriculture	the
is	<s>
is	agriculture
is	the
is	traditional
the	is
...	...

CBOW training data (window size = 5)

corpus (English Wikipedia):

*agriculture is the traditional mainstay of the cambodian economy .
but benares has been destroyed by an earthquake .*

...

inputs (x)	outputs (y)
{<s>, is, the, traditional}	agriculture
{<s>, agriculture, the, traditional}	is
{agriculture, is, traditional, mainstay}	the
{is, the, mainstay, of}	traditional
{the, traditional, of, the}	mainstay
{traditional, mainstay, the, cambodian}	of
{mainstay, of, cambodian, economy}	the
...	...

skip-gram model

$$\text{classify}(x, \theta) = \underset{y}{\text{argmax}} \text{ score}(x, y, \theta)$$

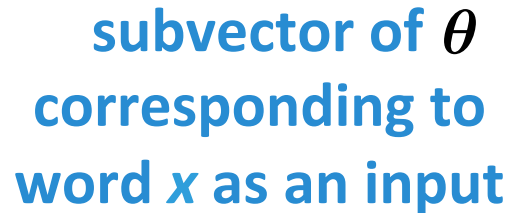
- here's our data:

inputs (x)	outputs (y)
agriculture	<s>
agriculture	is
agriculture	the
is	<s>
...	...

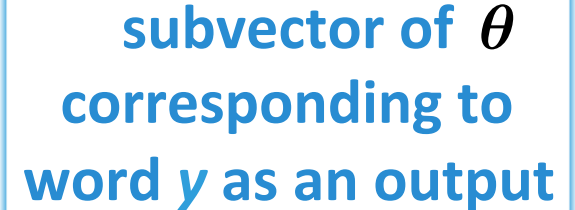
- how should we define the score function?

skip-gram score function

subvector of θ
corresponding to
word x as an input

A blue-bordered box containing the text "subvector of theta corresponding to word x as an input". A blue arrow points from the bottom center of this box down towards the equation below.

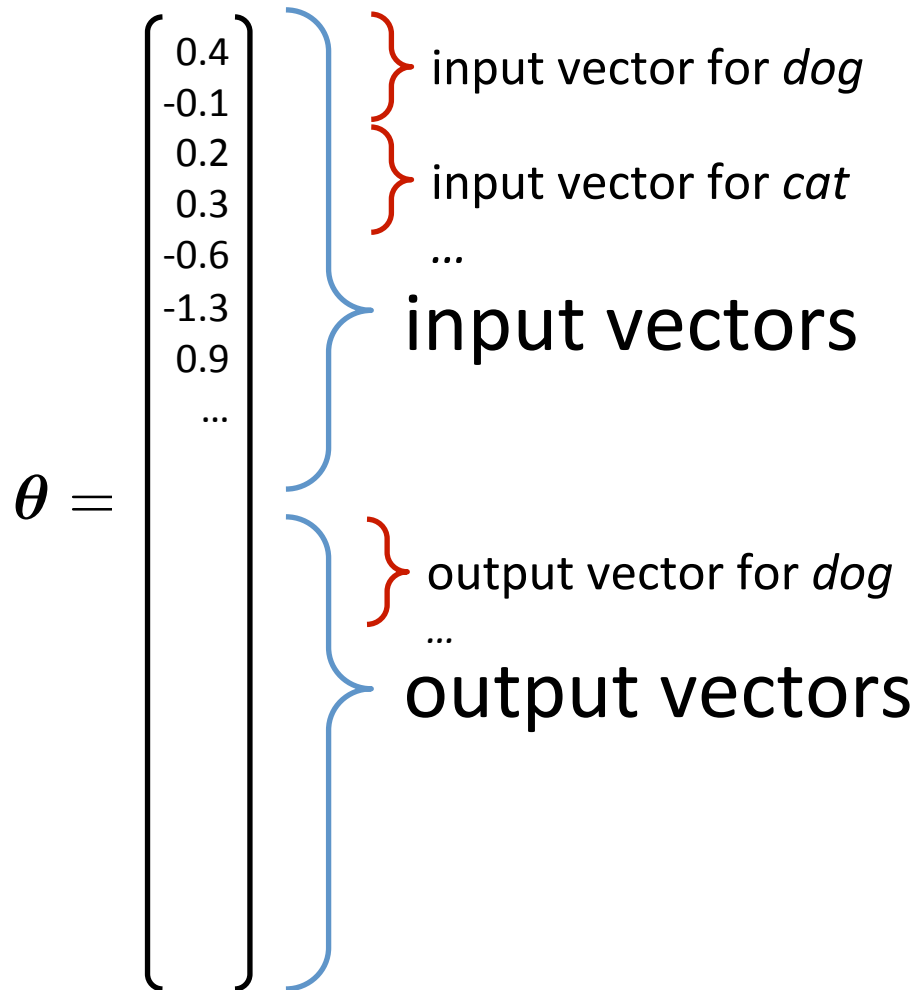
subvector of θ
corresponding to
word y as an output

A blue-bordered box containing the text "subvector of theta corresponding to word y as an output". A blue arrow points from the bottom center of this box down towards the equation below.

$$\text{score}(x, y, \theta) = \theta^{(\text{in}, x)} \cdot \theta^{(\text{out}, y)}$$

- dot product of two vectors, one for each word
- subtlety: different vector spaces for input and output
- no interpretation to vector dimensions (*a priori*)

skip-gram parameterization



What will the skip-gram model learn?

$$\text{score}(x, y, \theta) = \theta^{(\text{in}, x)} \cdot \theta^{(\text{out}, y)}$$

- corpus:

an earthquake destroyed the city

the town was destroyed by a tornado

- sample of training pairs:

inputs (x)	outputs (y)
destroyed	earthquake
earthquake	destroyed
destroyed	tornado
tornado	destroyed
...	...

- output vector for *destroyed* encouraged to be similar to input vectors of *earthquake* and *tornado*

Learning

$$\text{classify}(x, \theta) = \underset{y}{\operatorname{argmax}} \text{ score}(x, y, \theta)$$



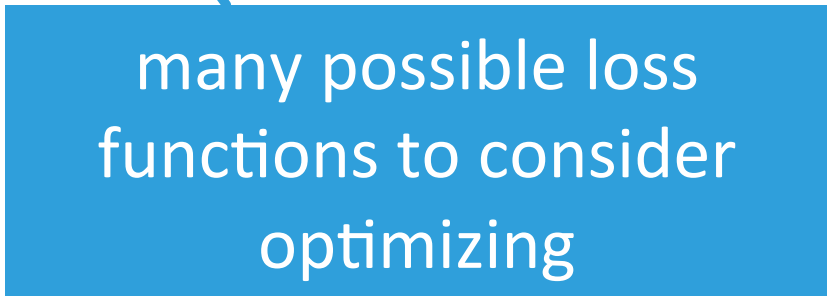
learning: choose θ

- you could use any loss function we have talked about
- Mikolov et al. (2013) use **log loss**, which is a new loss function for us

Empirical Risk Minimization with Surrogate Loss Functions

- given training data: $\mathcal{T} = \{\langle \mathbf{x}^{(i)}, y^{(i)} \rangle\}_{i=1}^{|\mathcal{T}|}$
where each $y^{(i)} \in \mathcal{L}$ is a label
- we want to solve the following:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{T}|} \operatorname{loss}(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$



many possible loss
functions to consider
optimizing

$$\operatorname{loss}_{\text{perc}}(x, y, \boldsymbol{\theta}) = -\operatorname{score}(x, y, \boldsymbol{\theta}) + \max_{y' \in \mathcal{L}} \operatorname{score}(x, y', \boldsymbol{\theta})$$

Log Loss

$$\text{loss}_{\log}(x, y, \theta) = -\log p_{\theta}(y | x)$$

- minimize negative log of conditional probability of output given input
 - sometimes called “maximizing conditional likelihood”
- but wait, we don’t have a probabilistic model, we just have a $\text{score}(x, y, \theta)$ function

Score \rightarrow Probability

- we can turn our score into a probability by exponentiating (to make it positive) and normalizing:

$$p_{\theta}(y | x) = \frac{\exp\{\text{score}(x, y, \theta)\}}{\sum_{y' \in \mathcal{L}} \exp\{\text{score}(x, y', \theta)\}}$$

- this is often called a “softmax” function

Log Loss

$$\begin{aligned}\text{loss}_{\log}(x, y, \theta) &= -\log p_{\theta}(y | x) \\ &= -\log \frac{\exp\{\text{score}(x, y, \theta)\}}{\sum_{y' \in \mathcal{L}} \exp\{\text{score}(x, y', \theta)\}} \\ &= -\text{score}(x, y, \theta) + \log \sum_{y' \in \mathcal{L}} \exp\{\text{score}(x, y', \theta)\}\end{aligned}$$

- si
 - ju
- log loss is used in:
logistic regression classifiers,
conditional random fields,
maximum entropy (“maxent”) models θ)

loss_p

Log Loss

$$\begin{aligned}\text{loss}_{\log}(x, y, \theta) &= -\log p_{\theta}(y | x) \\ &= -\log \frac{\exp\{\text{score}(x, y, \theta)\}}{\sum_{y' \in \mathcal{L}} \exp\{\text{score}(x, y', \theta)\}} \\ &= -\text{score}(x, y, \theta) + \log \sum_{y' \in \mathcal{L}} \exp\{\text{score}(x, y', \theta)\}\end{aligned}$$

- pro
all p

approximations are commonly
used in practice:
hierarchical softmax,
negative sampling

g over

loss_{perc}

y', θ)

word2vec

- `word2vec` toolkit implements training for skip-gram and CBOW models
- very fast to train, even on large corpora
- pretrained embeddings available

A simple way to investigate the learned representations is to find the closest words for a user-specified word. The `distance` tool serves that purpose. For example, if you enter 'france', `distance` will display the most similar words and their distances to 'france', which should look like:

Word	Cosine distance
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.563291
catalonia	0.534176

Embeddings capture relational meaning!

$\text{vector}(\textit{king}) - \text{vector}(\textit{man}) + \text{vector}(\textit{woman}) \approx \text{vector}(\textit{queen})$

$\text{vector}(\textit{Paris}) - \text{vector}(\textit{France}) + \text{vector}(\textit{Italy}) \approx \text{vector}(\textit{Rome})$

