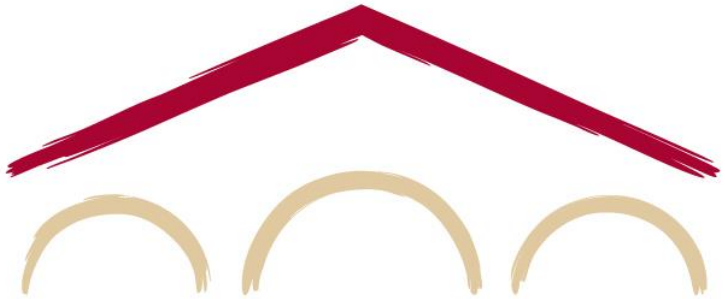
Natural Language P
与 Deep Learning
CS224N/Ling284



Diyi Yang

Lecture 2：Word Vectors

课程计划

Ｌｅｃｔｕｒｅ　２：Ｗｏｒｄ　Ｖｅｃｔｏｒｓ

1. 课程安排（３　分钟）
2. Ｗｏｒｄ２ｖｅｃ　简介（１５　分钟）
3. Ｗｏｒｄ２ｖｅｃ　目标函数梯度（２５　分钟）
4. 优化基础（５　分钟）
5. 我们能否通过计数更有效地捕捉词义的本质？（１０　分钟）
6. 评估　ｗｏｒｄ　ｖｅｃｔｏｒｓ（１０　分钟）

Key Goal: **understand word meaning can be represented by a high-dimensional vector of** 实数，并能在课程结束时阅读　ｗｏｒｄ　ｅｍｂｅｄｄｉｎｇｓ　论文

1． 课程安排

- 旁听 / 候补名单
  - 其他问题请发邮件至 cs224n-win2526-staff@lists.star
- 来参加 office hours / 答疑！
  - 他们今天开始了
  - Come to discuss **final project ideas** 以及作业
  - 尽量早来、常来、错峰来！
- TA office hours：周一至周六的 3 小时时段，多位 TA 在线
  - 直接来就好！我们友好的课程团队将随时为你提供帮助！
  - https://web.stanford.edu/class/cs224n/office_hours.html
- **Instructors' office hours** (in person by default):
  - Diyi: Tuesdays 3:30-4:30pm
  - Yejin: Fridays 4:30-5:30pm

2． 我们如何表示一个词的含义？

Definition: **meaning** (Webster dictionary)

- 由一个词、短语等所代表的概念
- 一个人想通过词语、符号等表达的想法
- 在写作、艺术等作品中表达的思想

最常见的语言学意义思考方式：

signifier (symbol) ⇔ signified (idea or thing)

= 指称语义学

tree ⇔ {  ,  ,  , …}

我们如何在计算机中获得可用的含义？

以往最常见的 NLP 解决方案：同义词集

Use, e.g., WordNet, a thesaurus containing lists of synonym sets and **hypernyms** (“is a” relationships)

e.g., synonym sets containing “good”:

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g., hypernyms of “panda”:

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

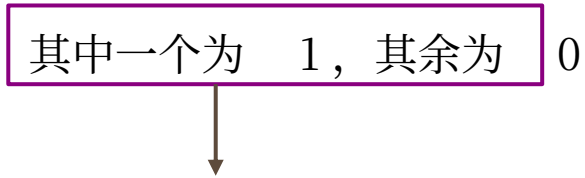
WordNet 等资源的问题

- 一个有用的资源但缺少细微差别：
  - e.g., “proficient” is listed as a synonym for “good”
    这只在某些上下文中正确
  - Also, WordNet list offensive synonyms in some synonym sets without any
    词的内涵或适当性的覆盖范围
- 缺少词的新含义：
  - e.g., wicked, badass, nifty, wizard, genius, ninja, bombest
  - 无法保持更新！
- 主观的
- 需要人工创建和调整
- Can’t be used to accurately compute word similarity (see following slides)

将词表示为离散符号

In traditional NLP, we regard words as discrete symbols:

hotel, conference, motel – a localist representation



Such symbols for words can be represented by one-hot vectors: (独热)

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

向量维度 = 词表中的词数（如 500,000+）

将词作为离散符号的问题

Example: in web search, if a user searches for "Seattle motel", we would like to match documents containing "Seattle hotel"

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

But these two vectors are orthogonal

There is no natural notion of **similarity** 对于 one-hot 向量！

解决方案：

- Could try to rely on WordNet's list of synonyms to get similarity?
  - 但众所周知它会严重失败：不完整性等
- 换一种方式：学习在向量本身中编码相似性

通过上下文表示词

- Distributional semantics: **A word's meaning is given**
  通过经常出现在附近的词
  - *"You shall know a word by the company it keeps"* (J. R. Firth 1957: 11)
  - 现代统计　ＮＬＰ　最成功的思想之一！
- When a word *w* appears in a text, its **context** is the set of words that appear nearby（在固定大小的窗口内）。
- We use the many contexts of *w* to build up a representation of *w*

……政府债务问题演变为	***banking***	*crises as happened in 2009…*
……说欧洲需要统一的	***banking***	*regulation to replace the hodgepodge…*
……印度刚刚给出了其	***banking***	给系统注入一针强心剂……

These context words will represent ***banking***

Word vectors

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts, measuring similarity as the vector dot (scalar) product

$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix} \qquad \textit{monetary} = \begin{pmatrix} 0.413 \\ 0.582 \\ -0.007 \\ 0.247 \\ 0.216 \\ -0.718 \\ 0.147 \\ 0.051 \end{pmatrix}$$

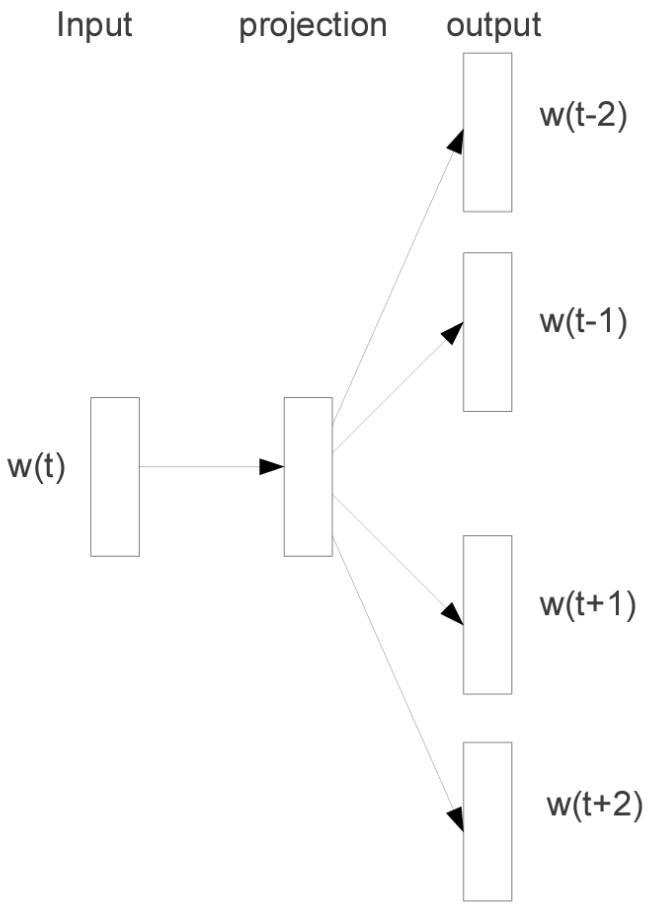
Note: word vectors are also called (word) embeddings or （神经）词表示
They are a 分布式 representation

3． Word2vec：概述

Word2vec 是一个学习 word vectors 的框架
(Mikolov et al. 2013)

想法：

- We have a large corpus (“body”) of text: a long list of words
- Every word in a fixed vocabulary is represented by a vector
- Go through each position *t* in the text, which has a center word *c* and context (“outside”) words *o*
- Use the similarity of the word vectors for *c* and *o* to calculate 概率 of *o* given *c* （或反过来）
- Keep adjusting the word vectors 最大化这个概率

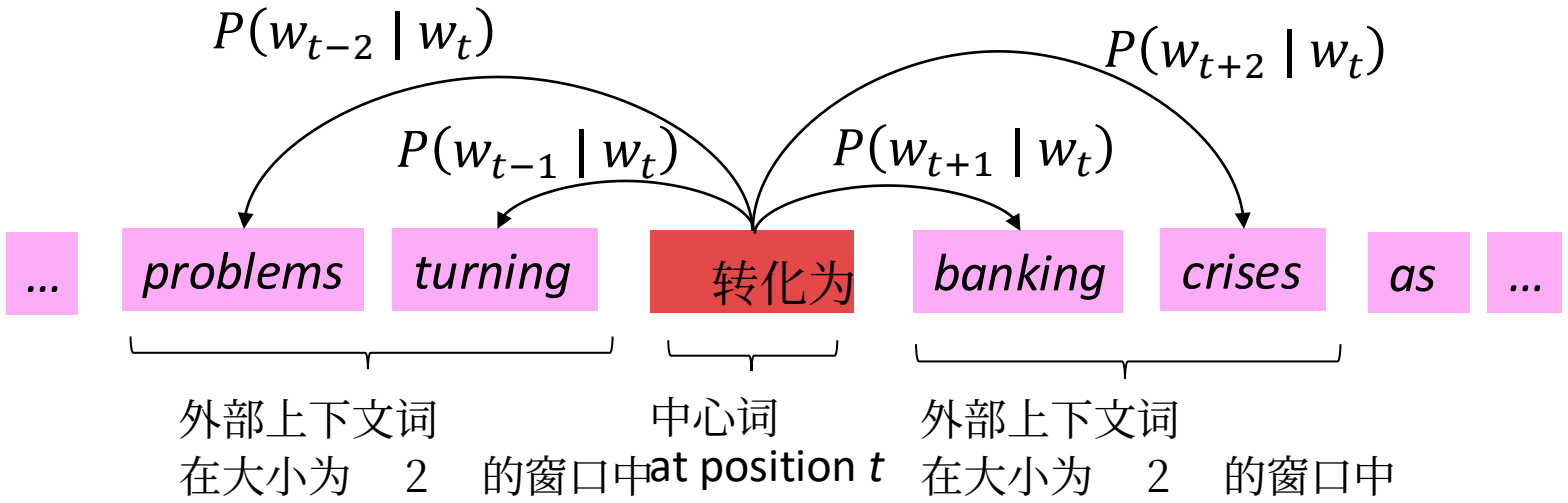


Skip-gram model
(Mikolov et al. 2013)

Word2Vec 概述

用于计算的示例窗口和过程

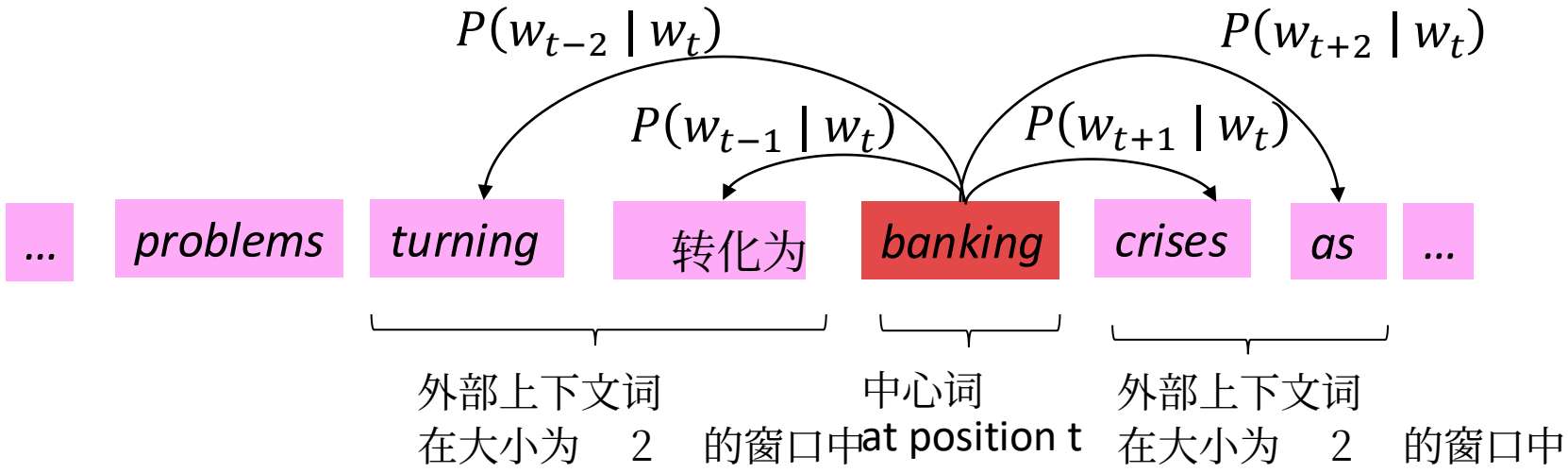
$P(w_{t+j} \mid w_t)$



Word2Vec 概述

用于计算的示例窗口和过程

$P(w_{t+j} \mid w_t)$



Word2Vec：目标函数

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_t。数据似然：

似然 $=$ $$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

θ is all variables
待优化

sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function ⟺ Maximizing predictive accuracy

Word2Vec：目标函数

- 我们想最小化目标函数：

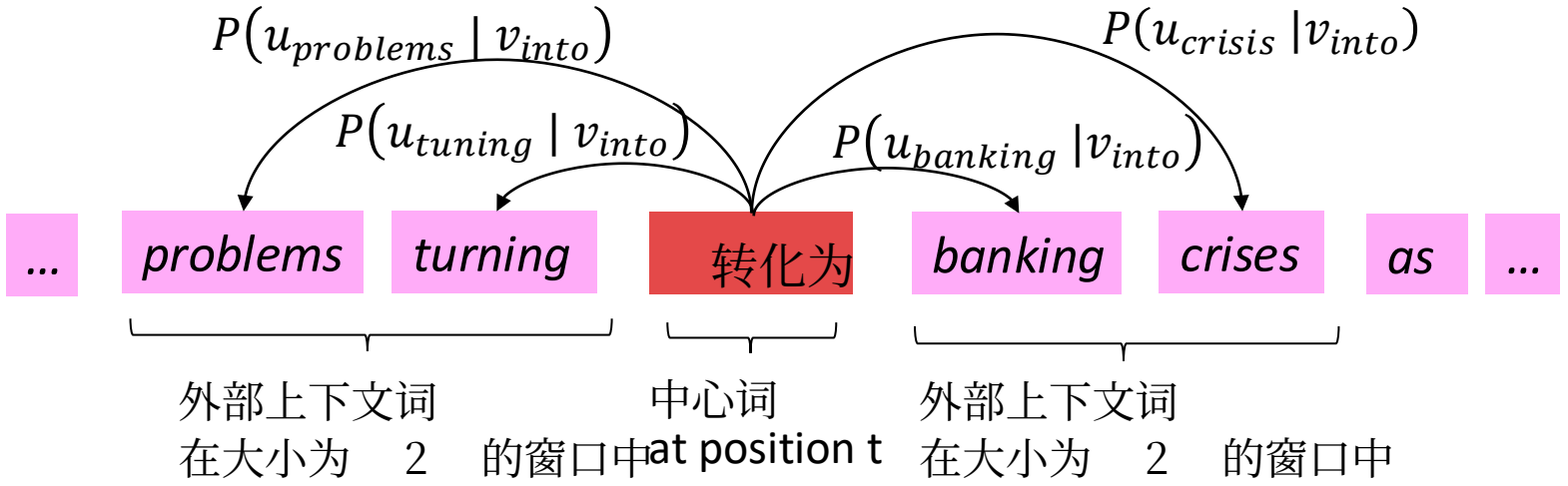
$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

- 问题： How to calculate $P\left(w_{t+j} \mid w_t; \theta\right)$?
- 回答： We will *use two* vectors per word *w*:
  - v_w when *w* is a center word
  - u_w when *w* is a context word
- Then for a center word *c* and a context word *o*:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

带向量的 Word2Vec

- 用于计算的示例窗口和过程 $P(w_{t+j} \mid w_t)$
- $P(u_{problems} \mid v_{into})$ short for $P(problems \mid into\ ; u_{problems}, v_{into}, \theta)$



Word2Vec：预测函数

② Exponentiation makes anything positive

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

① Dot product compares similarity of *o* and *c*.

$u^T v = u \cdot v = \sum_{i=1}^{n} u_i v_i$

更大的点积 = 更大的概率

③ Normalize over entire vocabulary

给出概率分布

- This is an example of the **softmax function** $\mathbb{R}^n \to (0,1)^n$ ← Open region

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - 在 Deep Learning 中常用

But sort of a weird name

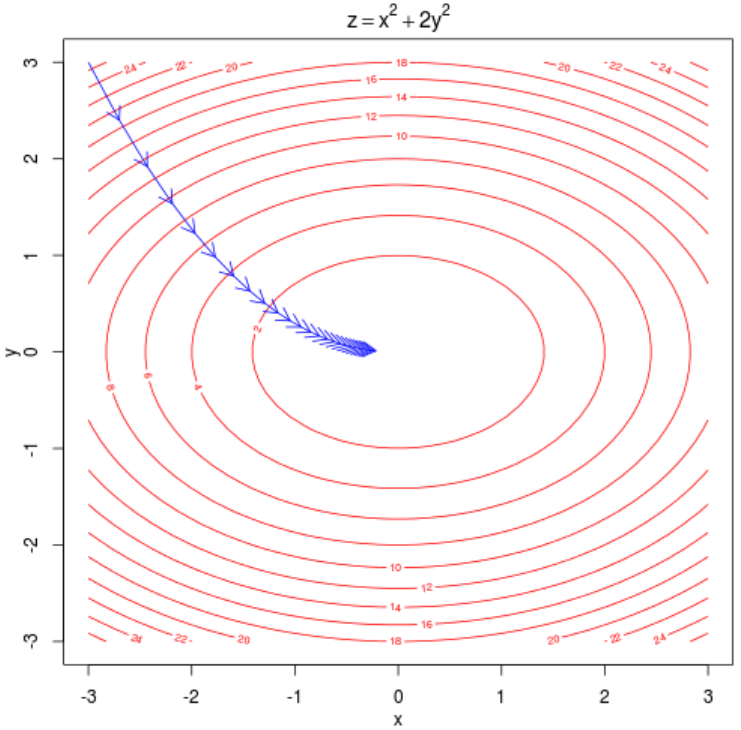
因为它返回一个分布！

训练模型：优化参数值以最小化损失

为了训练模型，我们逐步调整参数以最小化损失

- Recall: θ represents **all** the 模型参数，在一个长向量
- In our case, with *d*-dimensional vectors and *V*-many words, we have →
- Remember: every word has 两个向量

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$



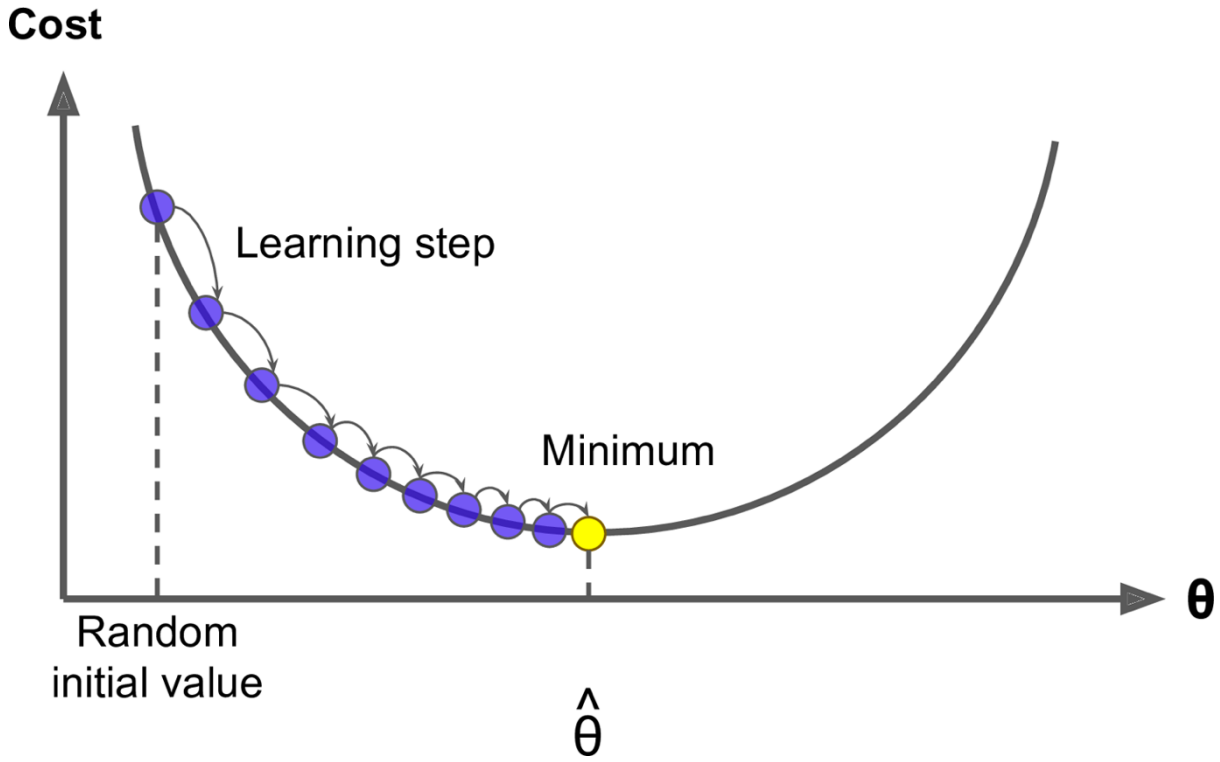
- 我们通过沿梯度下降来优化这些参数（见右图）
- We compute all vector gradients!

互动环节！

- $L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P\left(w_{t+j} \mid w_t; \theta\right)$
- For a center word *c* and a context word *o*: $P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$

4. 优化：Gradient Descent

- We have a cost function $J(\theta)$ we want to minimize
- **Gradient Descent** is an algorithm to minimize $J(\theta)$
- **Idea:** for current value of θ, calculate gradient of $J(\theta)$, then take small step in direction 负梯度的 。重复。



Note: Our objectives may not be convex like this ☹

But life turns out to be okay ☺

Gradient Descent（梯度下降）

- 更新公式（矩阵形式）：

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

α = *step size* or learning rate（学习率）

- 更新公式（单参数）：

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

- 算法：

```
while True:
    theta_grad = evaluate_gradient(J,corpus,theta)
    theta = theta - alpha * theta_grad
```

Stochastic Gradient Descent（随机梯度

- 问题　　：$J(\theta)$ is a function of **all** 语料库中的窗口（可能数十亿！）
  - So $\nabla_\theta J(\theta)$ is 计算非常昂贵
- 你将等待很长时间才能进行一次更新！

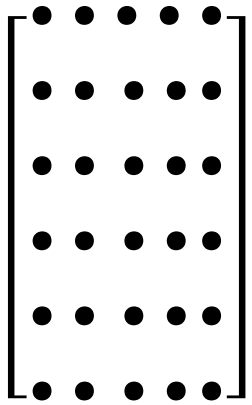
- **Very** bad idea for pretty much all neural nets!
- **Solution:** 随机梯度下降（SGD）
  - 反复采样窗口，每次采样后更新
- 算法：

```
while True:
    window = sample_window(corpus)
    theta_grad = evaluate_gradient(J,window,theta)
    theta = theta - alpha * theta_grad
```

Mini Batch

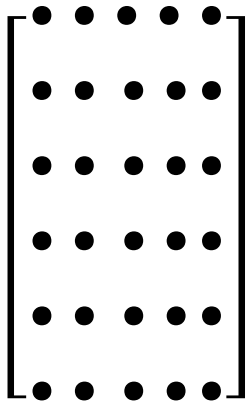
Gradient

Word2vec 参数……及计算



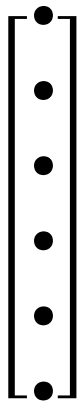
U

outside



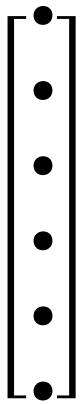
V

center



$U \cdot v_4{}^T$

dot product



$\text{softmax}(U \cdot v_4{}^T)$

probabilities

"Bag of words" model!

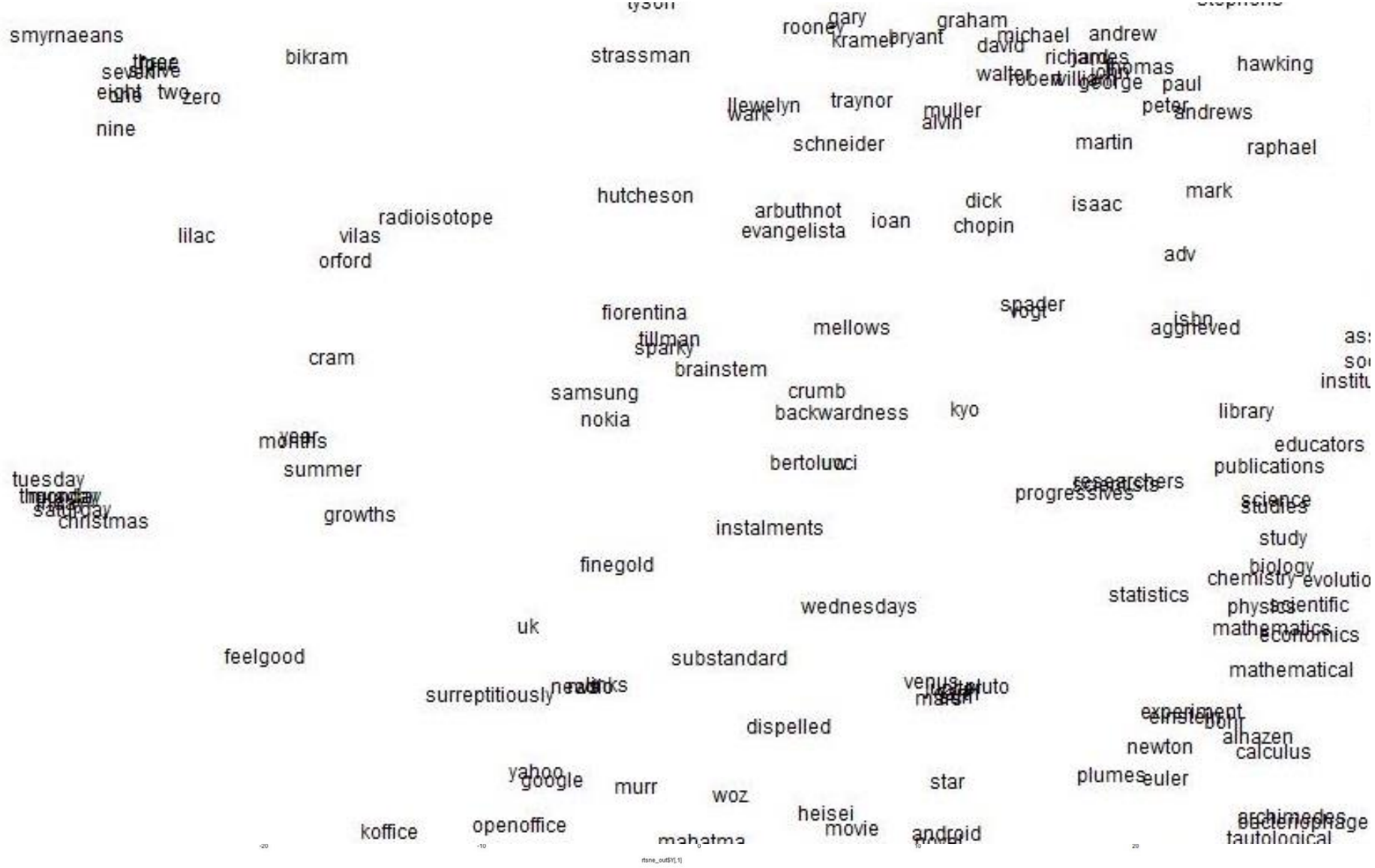
The model makes the same predictions at each position

We want a model that gives a reasonably high probability estimate to *all* words that occur in the

上下文（根本不经常）

Word2vec maximizes objective function by

将相似的词放在空间中的相近位置



Word2vec 算法族（Mikolov 等人 2013）：更多

Why two vectors? → Easier optimization. Average both at the end

- 但可以仅用每个词一个向量实现该算法……而且有一点帮助

两种模型变体：

1. Skip-grams（SG）
   Predict context ("outside") words (position independent) given center word
2. 连续词袋（CBOW）
   Predict center word from (bag of) context words

We presented: Skip-gram 模型

训练的损失函数：

1. Naïve softmax（简单但昂贵的损失函数，当输出类别很多时）
2. 更优化的变体如 hierarchical softmax
3. Negative sampling（负采样）

So far, we explained **naïve softmax**

The skip-gram model with negative sampling

- 归一化项计算成本高（当输出类别很多时）：

- $$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

对所有词求和

- Hence, standard word2vec implements the skip-gram model with 负采样

- Main idea: train binary logistic regressions to differentiate a true pair (center word and a word in its context window) versus several “noise” pairs (the center word paired with 一个随机词）

带 negative sampling 的 skip-gram (Mikolov et al. 2013)

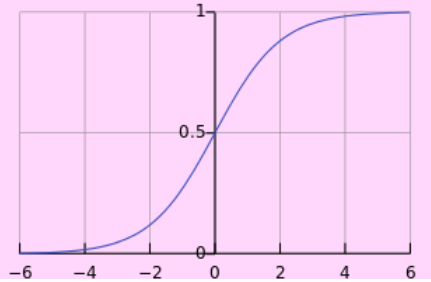
- We take *k* negative samples (using word probabilities)
- Maximize probability that real outside word appears;
  最小化随机词出现在中心词周围的概率
- 使用与本课一致的符号，我们最小化：

$$J_{neg-sample}(\boldsymbol{u}_o, \boldsymbol{v}_c, U) = -\log\sigma(\boldsymbol{u}_o^T\boldsymbol{v}_c) - \sum_{k\in\{K\ sampled\ indices\}} \log\sigma(-\boldsymbol{u}_k^T\boldsymbol{v}_c)$$

sigmoid rather than softmax

- The logistic/sigmoid function:
  (we'll become good friends soon ☺)

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- Sample with P(*w*)=U(*w*)$^{3/4}$ / Z，unigram 分布 U（w） 的 3/4 次方
  - 幂次使低频词被更频繁地采样

带 negative sampling 的随机梯度 [补充]

- 我们在每个窗口迭代地计算 SGD 的梯度
- 在每个窗口中，我们最多只有 2 个 $m + 1$ words plus $2km$ negative words with negative sampling, so $\nabla_\theta J_t(\theta)$ is very sparse!

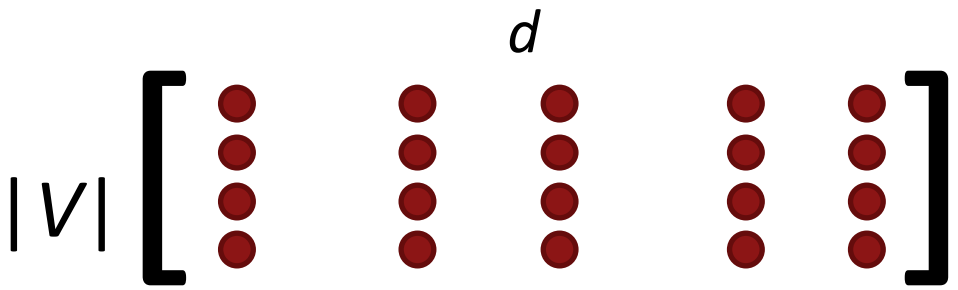
$$\nabla_\theta J_t(\theta) = \begin{bmatrix} 0 \\ \vdots \\ \nabla_{v_{like}} \\ \vdots \\ 0 \\ \nabla_{u_I} \\ \vdots \\ \nabla_{u_{learning}} \\ \vdots \end{bmatrix} \in \mathbb{R}^{2dV}$$

带 negative sampling 的随机梯度 [补充]

- 我们可能只更新实际出现的 word vectors！

- Solution: either you need sparse matrix update operations to only update certain rows of full embedding matrices *U* and *V*,
  或者你需要为 word vectors 保留一个哈希表

Rows not columns in actual DL packages!



- If you have millions of word vectors and do distributed computing, it is important to not have to send gigantic
  传递更新！

5． 为什么不直接捕获共现计数？

There’s something weird about iterating through the whole corpus (perhaps many times); why don’t we just accumulate all the statistics of what words appear near each other?!?

Building a co-occurrence matrix *X*

- 2 options: windows vs. full document
- Window: Similar to word2vec, use window around each word → captures some syntactic and semantic information (“word space”)
- Word-document co-occurrence matrix will give general topics (all sports terms will have similar entries) leading to “Latent Semantic Analysis” (“document space”)

示例：基于窗口的共现矩阵

- 窗口长度　1（更常见：5 - 10）
- 对称的（左右上下文无关）

- Example corpus:
  - 我喜欢　Deep　Learning
  - 我喜欢　NLP
  - 我喜欢飞行

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

共现向量

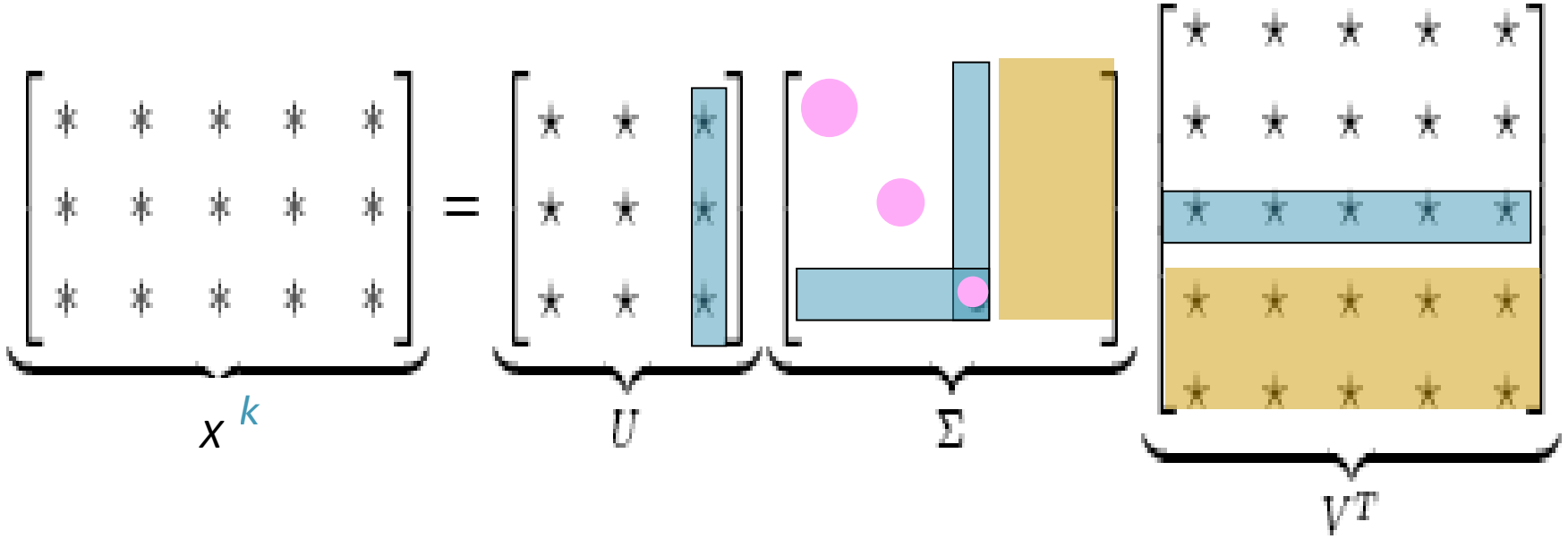
- 简单的计数共现向量
  - 向量大小随词表增大而增加
  - 非常高维：需要大量存储（虽然是稀疏的）
  - Subsequent classification models have sparsity issues → Models are less robust

- 低维向量
  - Idea: store “most” of the important information in a fixed, small number of 维度：一个稠密向量
  - 通常 ２５－１０００ 维，与 ｗｏｒｄ２ｖｅｃ 相似
  - 如何降低维度？

经典方法：对　X　进行降维（HW 1）

Singular Value Decomposition of co-occurrence matrix *X*

Factorizes *X* into $U\Sigma V^T$, where *U* and *V* are orthonormal (unit vectors and orthogonal)



Retain only *k* 奇异值，以便泛化。

$\hat{X}$ is the best rank *k* approximation to *X* , in terms of least squares.

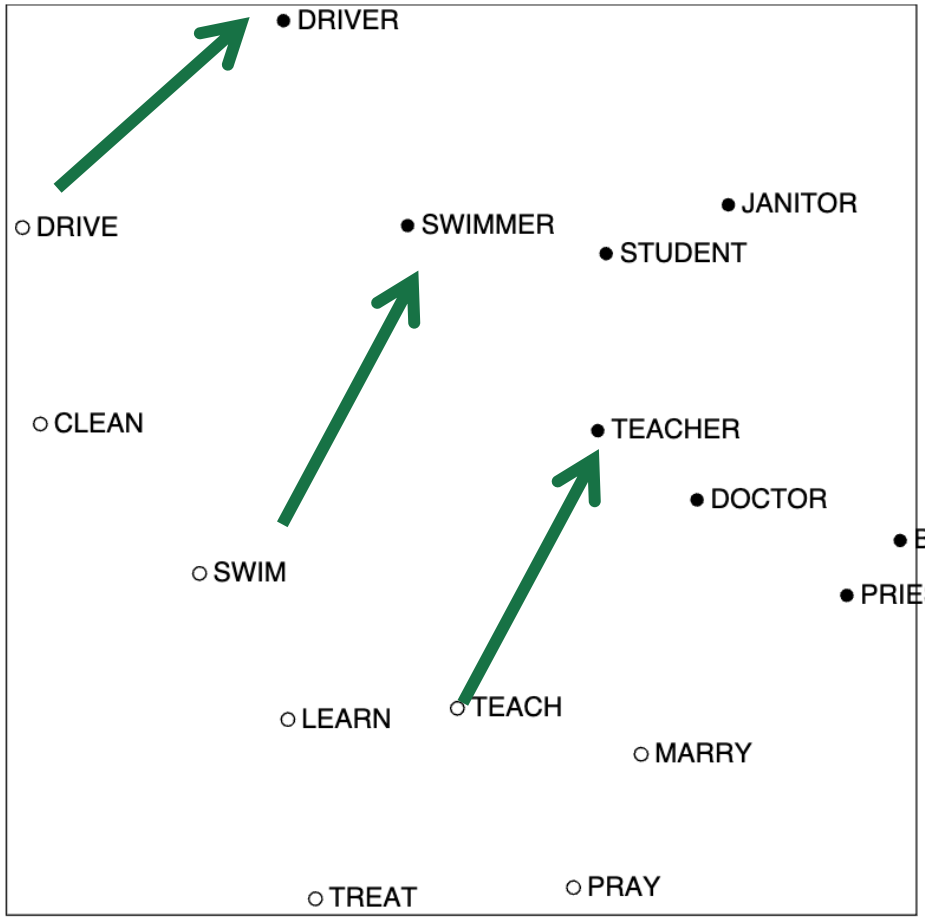
经典线性代数结果。对大矩阵计算成本高。

Hacks to X

- Running an SVD on raw counts doesn't work well!!!

- Scaling the counts in the cells can help 很多
  - 问题：功能词（ *the, he, has*) are too frequent → syntax has too much impact. Some fixes:
    - 取频率的对数
    - min(*X*, *t*), with *t* ≈ 100
    - 忽略功能词
- 渐变窗口，对距离较近的词赋予更高的计数
- 使用相关性代替计数，然后将负值设为 0
- Etc.

缩放后的向量中出现了有趣的语义模式



COALS 模型来自
Rohde et al. ms., 2005. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence

GloVe [Pennington, Socher, and Manning, EMNLP 2014]:

在向量差中编码语义成分

Q: How can we capture ratios of co-occurrence probabilities as 词向量空间中的线性语义成分？

GloVe [Pennington, Socher, and Manning, EMNLP 2014]:

在向量差中编码语义成分

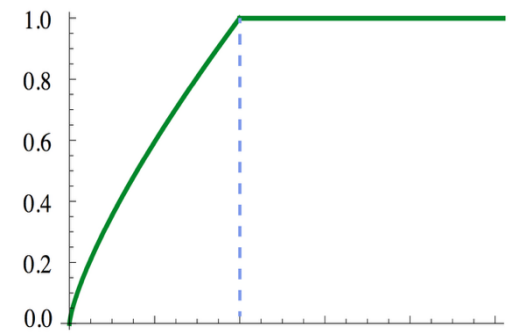
Q: How can we capture ratios of co-occurrence probabilities as 词向量空间中的线性语义成分？

A：Log-bilinear 模型：$w_i \cdot w_j = \log P(i|j)$

用向量差

$$w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$$

损失：$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

$f \sim$ 

- Fast training
- Scalable to huge corpora

6. 如何评估 word vectors？

- 与 NLP 中的一般评估相关：内在 vs. 外在
- 内在：
  - 在特定 / 中间子任务上评估
  - 计算速度快
  - 有助于理解该系统
  - Not clear if it’s helpful unless correlation to real task is established
- 外在：
  - 在真实任务上评估
  - 计算精度可能需要很长时间
  - 不清楚是子系统本身有问题还是其交互或其他子系统有问题
  - If replacing exactly one subsystem with another improves accuracy → Winning!

内在 word vector 评估

- Word Vector 类比

a:b :: c:?

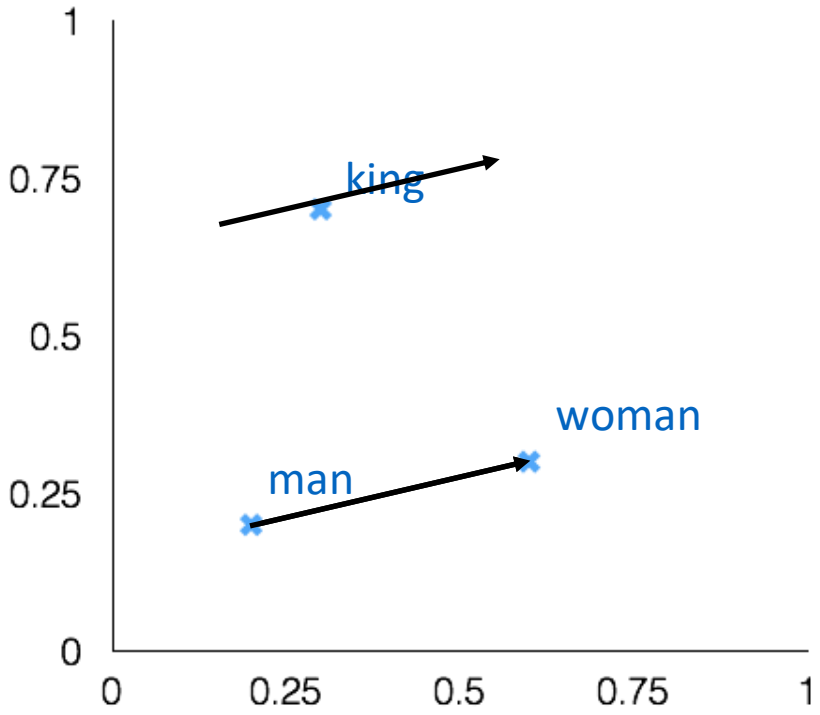
man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

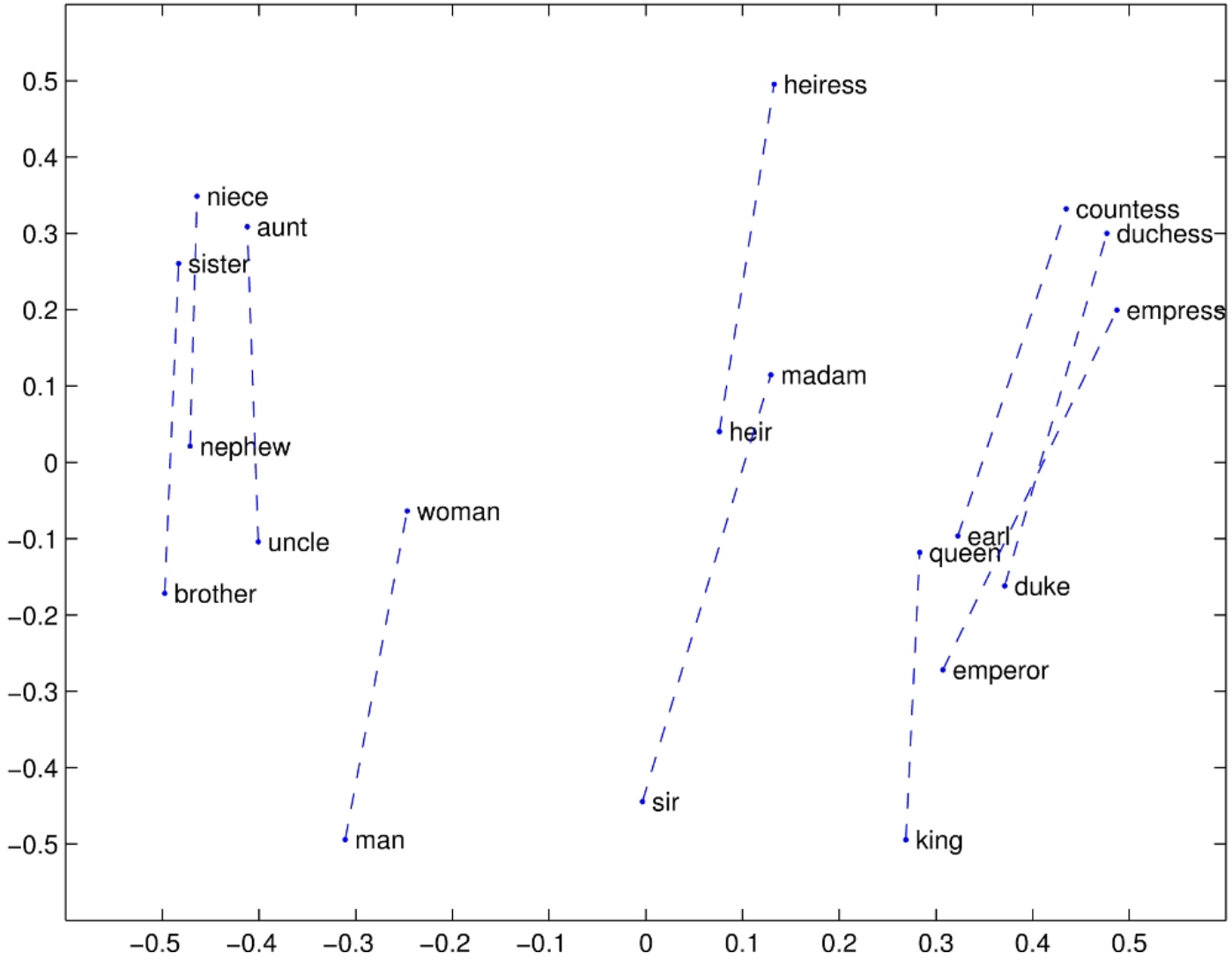
- Evaluate word vectors by how well their cosine distance after addition captures intuitive

  语义和句法类比问题
- 从搜索中丢弃输入词（！）
- Problem: What if the information is there but

  不是线性的？



GloVe 可视化


0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
niece
aunt
sister
nephew
uncle
brother
woman
man
heiress
madam
heir
sir
countess
duchess
empress
earl
queen
duke
emperor
king

语义相似度：另一种内在 word vector 评估

- Word vector 距离及其与人类判断的相关性
- Example dataset: WordSim353
  http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Word 1	Word 2	人类（均值）
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

相关性评估

- Word vector 距离及其与人类判断的相关性

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

外在 word vector 评估

- One example where good word vectors should help directly: **named entity recognition**: identifying references to a person, organization or location: Chris Manning lives in Palo Alto.

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

课程计划

Ｌｅｃｔｕｒｅ　２：Ｗｏｒｄ　Ｖｅｃｔｏｒｓ

1. 课程安排（３　分钟）
2. Ｗｏｒｄ２ｖｅｃ　简介（１５　分钟）
3. Ｗｏｒｄ２ｖｅｃ　目标函数梯度（２５　分钟）
4. 优化基础（５　分钟）
5. 我们能否通过计数更有效地捕捉词义的本质？（１０　分钟）
6. 评估　ｗｏｒｄ　ｖｅｃｔｏｒｓ（１０　分钟）

Key Goal: understand word meaning can be represented by a high-dimensional vector of 实数，并能在课程结束时阅读　ｗｏｒｄ　ｅｍｂｅｄｄｉｎｇｓ　论文