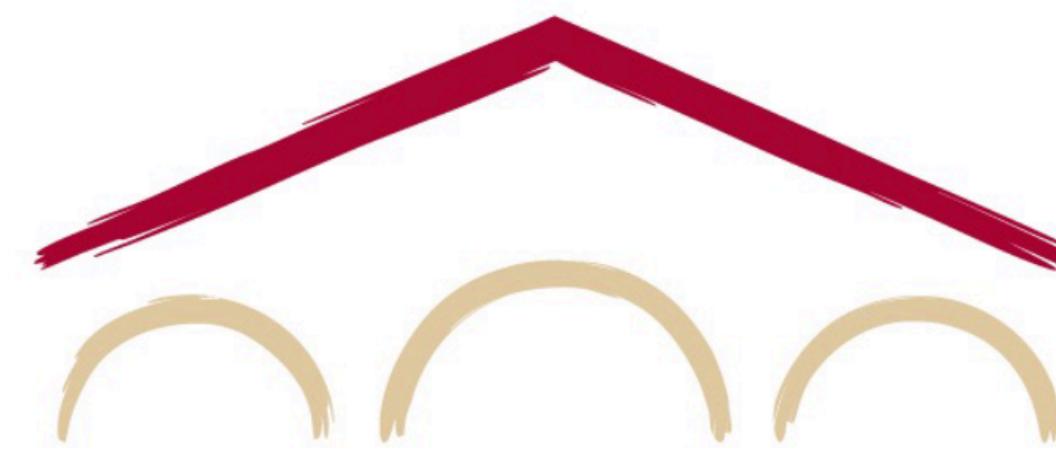


# 深度学习中的自然语言处理 CS224N/Ling284



Christopher Manning 第二讲：词向量、词义和神经网络分类器

# 课程计划

## 第二讲：词向量、词义和神经网络分类器

1. 课程组织 (3 mins)
2. 查看词向量和 word2vec (15 mins)
3. 我们能否通过计数更有效地捕捉词义的本质? (10m)
4. 评估词向量 (10 mins)
5. 词义 (8 分钟)
6. 分类复习和神经网络的区别 (14 分钟)
7. 介绍神经网络 (10 分钟)

关键目标：能够在课程结束时读懂词嵌入论文

## 1. 课程组织

- 来办公室答疑/辅导时段吧!
  - 它们是从昨天开始的
  - 今天, 课后: 18:00–20:50, 在 260-012 教室, 猪特楼 (或 Canvas 上的 Zoom 链接)
  - 来讨论最终项目的想法以及作业
  - 尽量早点来, 经常来, 还有非定期时间!
- 助教办公室时间: 周一至周五 (希望包括周六) 的 3 小时块, 有多名助教
  - 直接来吧! 我们的友好课程工作人员会随时为您提供帮助!
- Chris 的办公时间:
  - 周一 2:45-5 点。仅限亲自到场。下周一会见你!
  - 在 Calendly 上预约 15 分钟的时间段 (即将上线 – 我会发一封 Ed 通知)

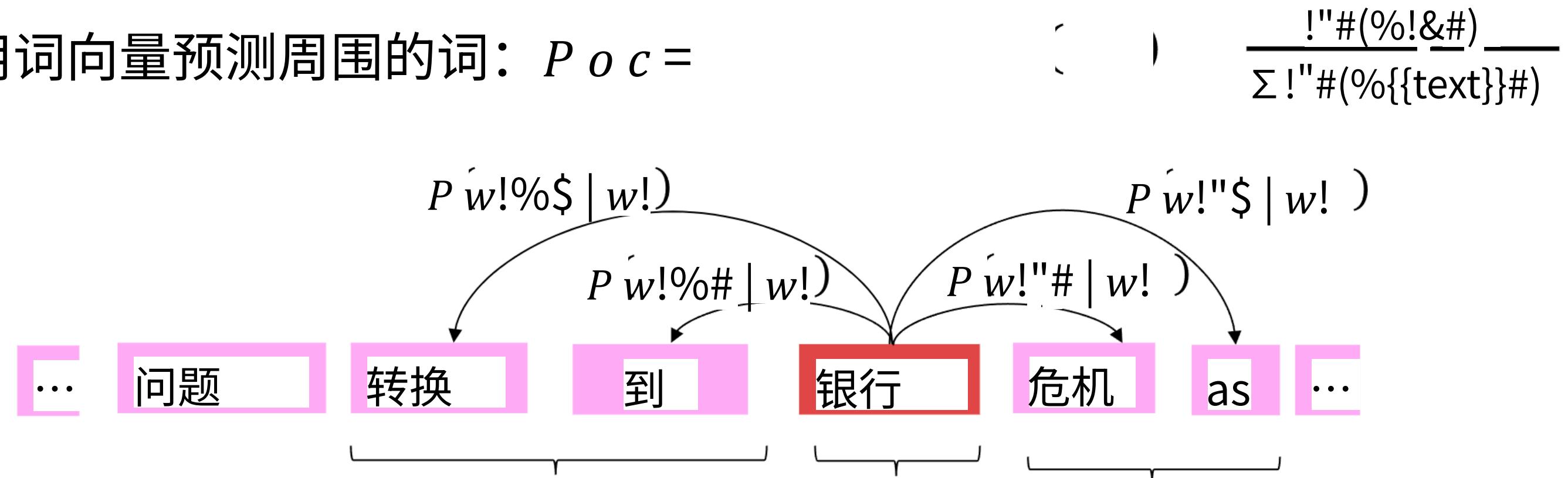
# 用于深度学习项目的 GPU

- 我们希望让更多人能够使用 Azure GPU 来帮助项目等。
- 但获取免费 GPU 变得非常困难；全球 GPU 供应短缺。
- 我们很快会提供更多关于此的信息，要求人们注册 Azure 账户。
- 请尽快阅读并完成！
- 需要很多前置时间 – 比如几周
  - 去年，很多人推迟了使用，因为他们当时不需要，然后他们需要的时候 GPU 就 unavailable 了
- 另外也要考虑是否有其他 GPU 可以用于这门课程



## 2. Review: word2vec 的主要思想

- 使用随机词向量开始
- 遍历整个语料库中的每个词位
- 尝试使用词向量预测周围的词:  $P(o|c) = \frac{e^{w_c^T w_o}}{\sum e^{w_c^T w_i}}$



- 学习: 更新向量以便更好地预测实际周围的词
- 仅此而已, 这种算法学习出的词向量很好地捕捉了词的相似性和词空间中的有意义方向!



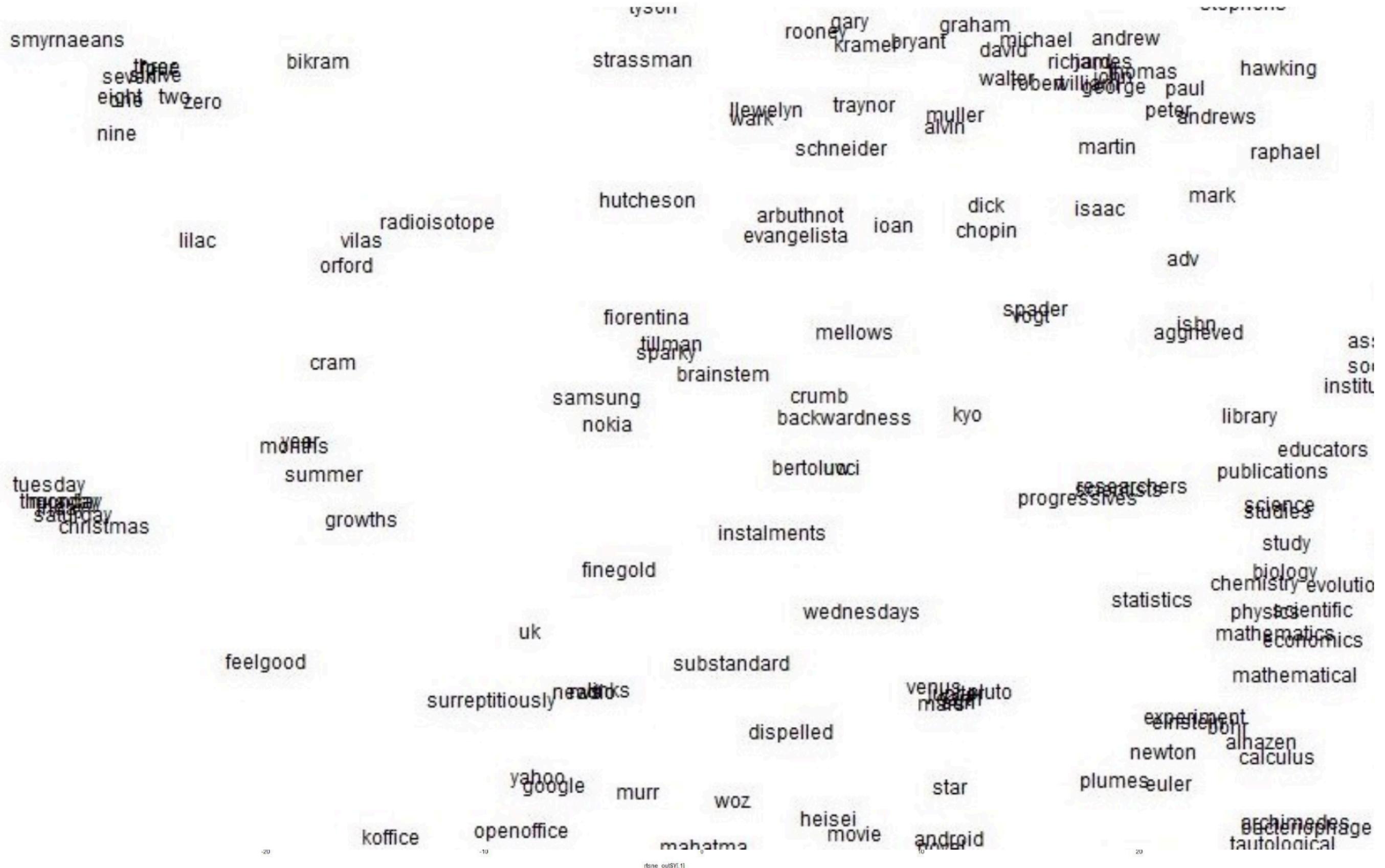
$$\begin{array}{c} \left[ \begin{array}{cccccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{array} \right] \\ U \end{array} \quad \begin{array}{c} \left[ \begin{array}{cccccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{array} \right] \\ V \\ \text{center} \end{array} \quad \begin{array}{c} \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{array} \right] \\ U \cdot v \\ \text{点积} \end{array} \quad \begin{array}{c} \left[ \begin{array}{c} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{array} \right] \\ \text{softmax}(U \cdot v) \\ \text{概率} \end{array}$$

“词袋”模型！

模型在每个位置做出相同的预测

我们希望模型对所有出现在上下文中的词都能给出一个相对较高的概率估计（无论出现频率如何）

# Word2vec 通过在空间中将相似的词放在一起最大化目标函数



## The skip-gram 模型带负采样 (HW2)

- 归一化项计算成本高 (当输出类别很多时):

$$\bullet P(o_c) = \frac{!^{\#}(%\&)}{\sum !^{\#}(%\&)} \quad \xleftarrow{\text{一个大的词的总和}}$$

- 因此，在标准的 word2vec 和 HW2 中，你将实现基于跳词的模型负采样
- 主要思想：训练二元逻辑回归来区分一个真实的词对（中心词和其上下文窗口中的一个词）与几个“噪声”词对（中心词与一个随机词配对）

## Word2vec 算法家族 (Mikolov 等人 2013): 更多细节

为什么用两个向量? à 更容易优化。最后平均一下  
但可以用每个词一个向量来实现算法 … 这样会有一点帮助。两种模型变体:

### 1. 跳词模型 (SG)

根据中心词预测上下文 (“外部”) 词 (位置无关)

### 2. 连续词袋 (CBOW)

从 (袋) 上下文词预测中心词。我们提出了:

Skip-gram 模型

训练的损失函数:

1. 朴素 softmax (简单但当输出类别很多时计算成本高)
2. More optimized variants like hierarchical softmax

Negative sampling 到目前  
为止, 我们已经解释了朴素  
<sup>9</sup> 的 softmax

## The skip-gram 模型带负采样 (HW2)

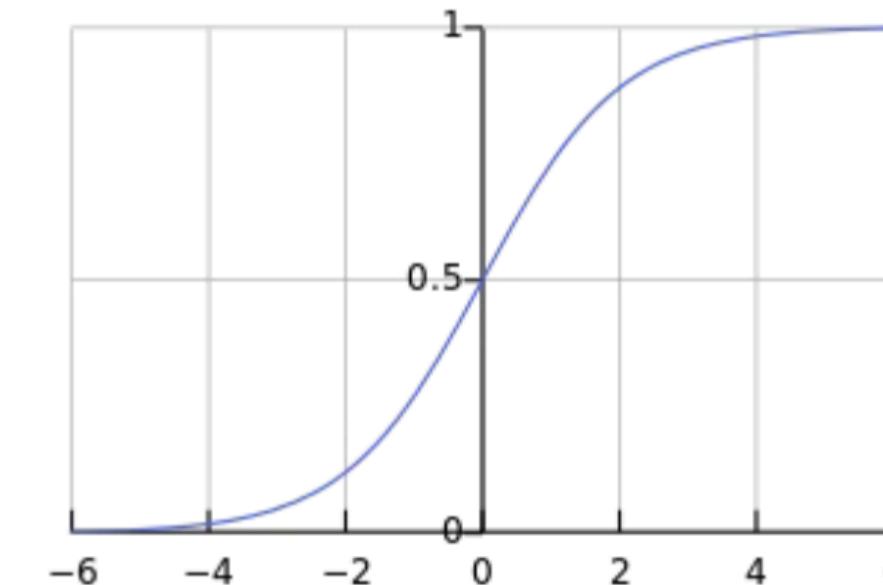
- 引入于：“分布式词与短语的表示及其组合性” (Mikolov 等, 2013)

- 总体目标函数 (他们最大化) :

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J_t(\theta)$$

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{i=1}^k \mathbb{E}_{j \sim P(w)} [\log \sigma(-u_j^T v_c)]$$

- 逻辑/sigmoid 函数: (我们很快  $\sigma(x) = \frac{1}{1+e^{-x}}$  就会成为好朋友)
- 我们最大化两个词共现的概率并在第二部分最小化噪声词的概率



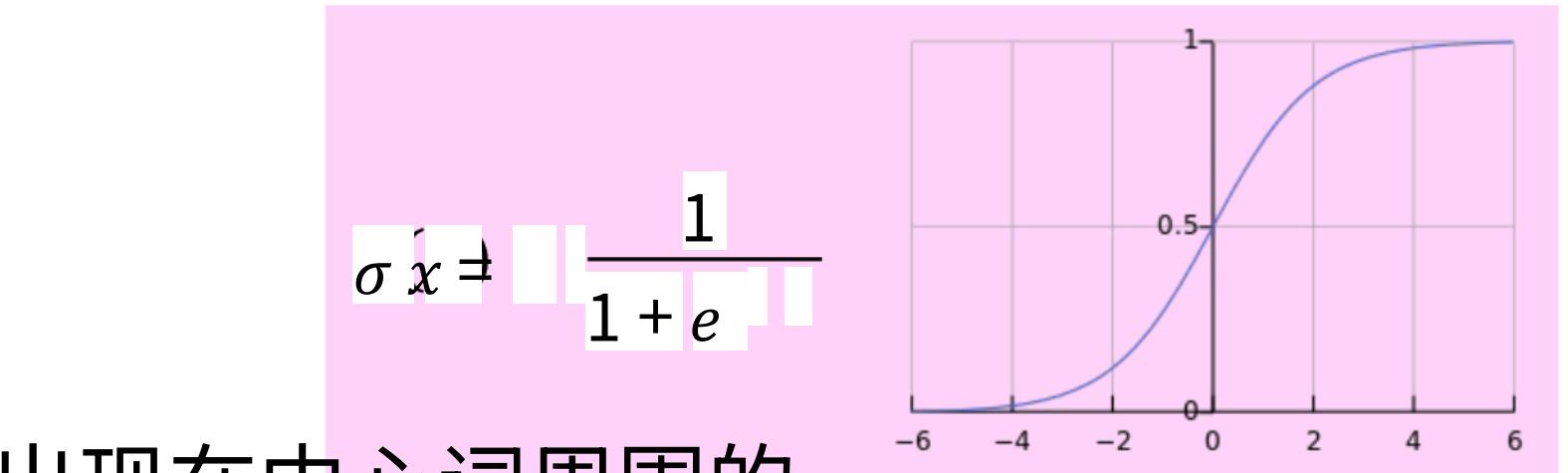
## The skip-gram 模型带负采样 (HW2)

- 使用与这个类和 HW2 一致的记号：

$$J = -\log \sigma(uv^T) + \sum_{w \in \text{negative samples}} \log \sigma(uw^T)$$

- 我们采取  $k$  个负样本（使用词的概率）
- 最大化真实外部词出现的概率；最小化随机词出现在中心词周围的概率
- 使用  $P(w) = U(w)/Z$  进行采样，其中  $U(w)$  是词的未平滑分布，提升到  $3/4$  的幂（我们在起始代码中提供了此函数）
- 幂使得出现频率较低的词被采样得更频繁

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



## 负采样下的随机梯度 [aside]

- 我们在每个窗口迭代计算梯度以进行 SGD
- 在每个窗口中，我们最多只有  $2m + 1$  个单词加上  $2km$  个负样本单词，因此  $\nabla J(\theta)$  非常稀疏！

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

## Stochastic 梯度与负采样 [aside]

- 我们可能只会更新实际出现的词向量！
- 解决方案：要么你需要稀疏矩阵更新操作，只更新嵌入矩阵  $U$  和  $V$  的某些行，要么你需要保留一个哈希表来存储词向量

行而不是列  
在实际 DL 中  
包！

$$[ \quad \quad \quad \quad \quad ]$$

$|V|$

d

⋮ ⋮ ⋮ ⋮

- 如果你有数百万个词向量并且进行分布式计算，那么不发送巨大的更新是非常重要的！

这也是 Adagrad 家族中更高级优化方法的一个特定问题

# 为什么不直接捕获共现计数？

遍历整个语料库（可能多次）有些奇怪；我们为什么不直接累积所有相邻词的统计信息？

## 构建共现矩阵 $X$

- 2 选项：窗口 vs. 全文
- 窗口：类似于 word2vec，使用每个词周围的窗口 à 捕捉一些语法和语义信息（“词空间”）
- 词-文档共现矩阵将给出一般主题（所有体育术语将有相似的条目），导致“潜在语义分析”（“文档空间”）

## 示例：基于窗口的共现矩阵

- 窗口长度 1 (更常见的是 5-10)
- 对称的 (无论是左文脉还是右文脉都无关紧要)
- Example corpus:
  - 我喜欢深度学习
  - 我喜欢 NLP
  - 我喜欢飞行

counts	我	喜欢	欣赏	深度	学习	NLP	飞行.	.	.	.	.
I	0	2	1	0	0	0	0	0	0	0	0
喜欢	2	0	0	1	0	0	1	0	0	0	0
欣赏	1	0	0	0	0	0	0	1	0	0	0
深度	0	1	0	0	0	1	0	0	0	0	0
学习	0	0	0	1	0	0	0	0	0	1	0
NLP	0	1	0	0	0	0	0	0	0	1	0
飞行	0	0	1	0	0	0	0	0	0	1	0
.	0	0	0	0	0	1	1	1	1	0	0

# 共现向量

- 简单的共现向量
  - 向量随词汇表增大而增大
  - 非常高维：需要大量的存储空间（尽管是稀疏的）
  - 后续的分类模型存在稀疏性问题：模型不够稳健
- 低维度向量
  - 想法：在固定的小数量的维度中存储“大部分”的重要信息：密集向量
  - 通常为 25 到 1000 维，类似于 word2vec
  - 如何降低维度？

## 经典方法：X 的维度降低 (HW1)

共现矩阵 X 的奇异值分解 将 X 分解为  $U\Sigma V^T$ ，其中 U 和 V 是正交规范的

$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{X^k} = \underbrace{\begin{bmatrix} * & * \\ * & * \\ * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} * & & \\ & * & \\ & & * \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{V^T}$$

仅保留 k 个奇异值，以便泛化。

$\hat{x}$  是 X 的最佳 k 阶逼近，按最小二乘法衡量。

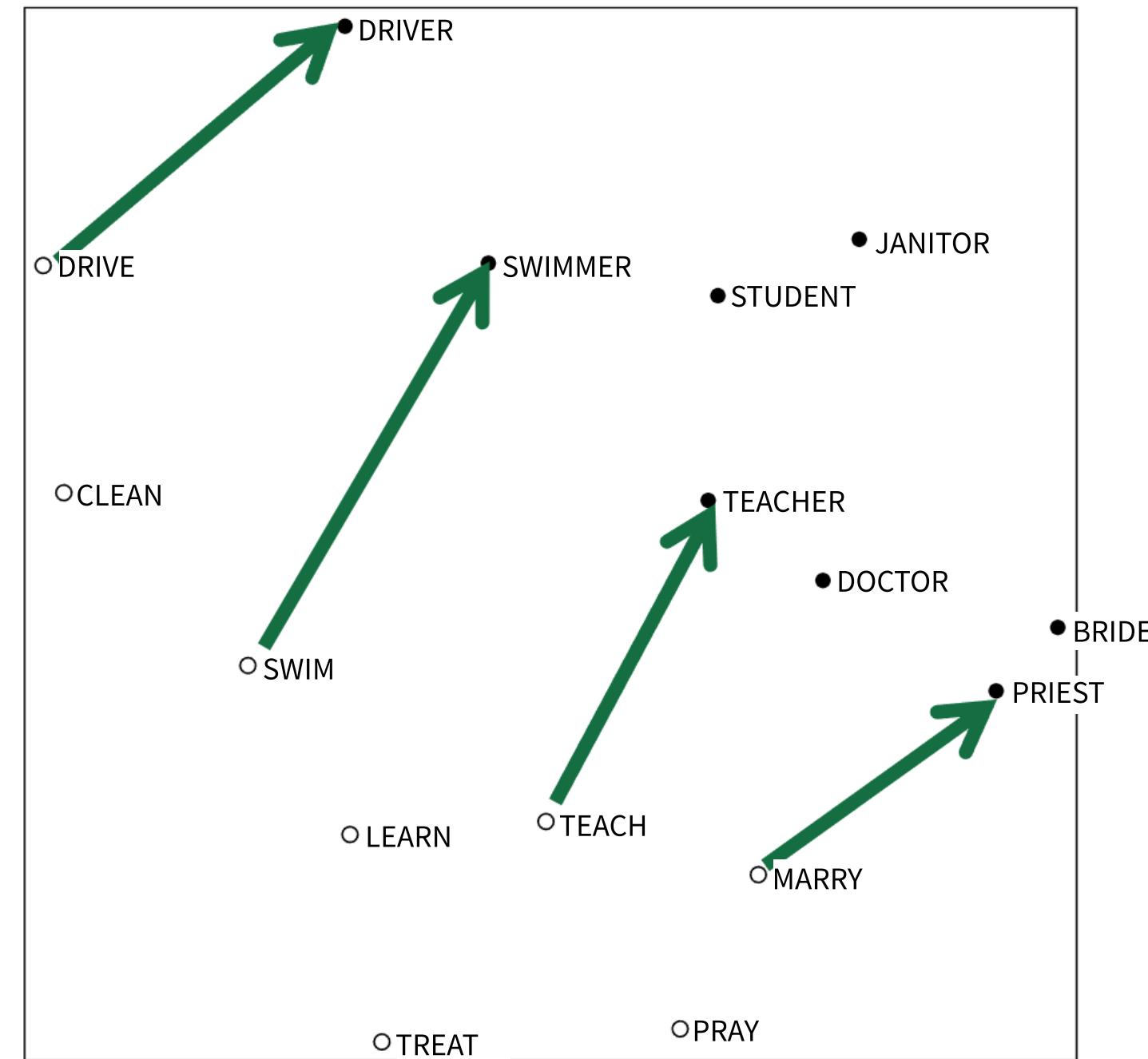
经典的线性代数结果。对于大型矩阵来说，计算起来很昂贵。

## 针对 X 的技巧 (罗德等, 2005 年在 COALS 中使用了几种)

- 直接对原始计数进行 SVD 效果不好!!!
- 对细胞中的计数进行缩放可以有很大帮助
  - 问题：功能词 (the, he, has) 过于频繁à语法影响太大。一些修复措施：
    - 记录频率
    - $\min(X, t)$ , 其中  $t \approx 100$
    - 忽略功能词
- 递增的窗口，靠近的词权重更高，远离的词权重更低
- 使用皮尔逊相关系数代替计数，然后将负值设为 0
- 等等

罗德, 格农曼, 布劳特  
有趣的语义模式在缩放向量中出现

使用词共现建模词义



Rohde 等, 2005 年的 COALS 模型。基于词共现的一种改进的语义相似度模型

表 10

COALS-14K 模型下一组名词的 10 个最近邻及其百分比相关相似度。



Q: 我们如何在词向量空间中捕获共现概率的比例作为线性意义组件?

A: 对数线性模型:

$$w_i \cdot w_j = \log P(i|j)$$

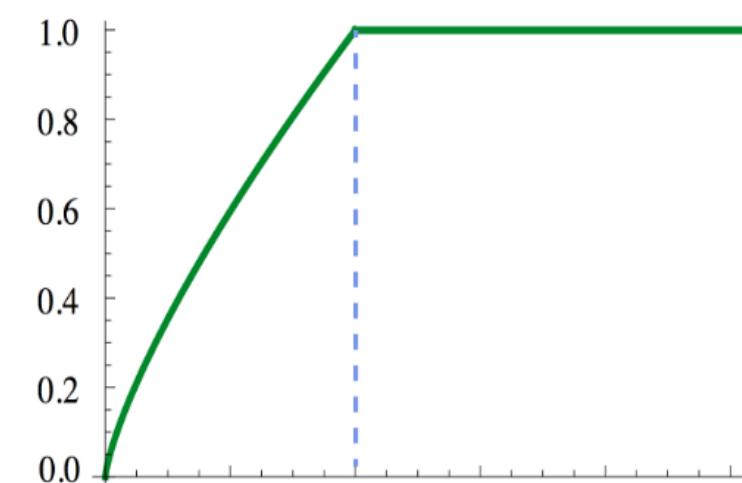
与向量差异

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

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

$f \sim$

- 快速训练
- 可扩展到庞大的语料库



## 4. 如何评估词向量?

- 与 NLP 中的通用评估：内在 vs. 外在相关
- 内在：
  - 针对特定/中间子任务的评估
  - 计算快速
  - 帮助理解系统
  - 除非与实际任务建立联系，否则不清楚是否真的有帮助
- 外部因素：
  - 基于实际任务的评估
  - 计算准确性可能需要很长时间
  - 不清楚是子系统本身的问题还是与其他子系统的交互问题
  - 如果用另一个子系统替换恰好一个子系统后准确性提高，则获胜！

# 固有词向量评估

- 词向量类比

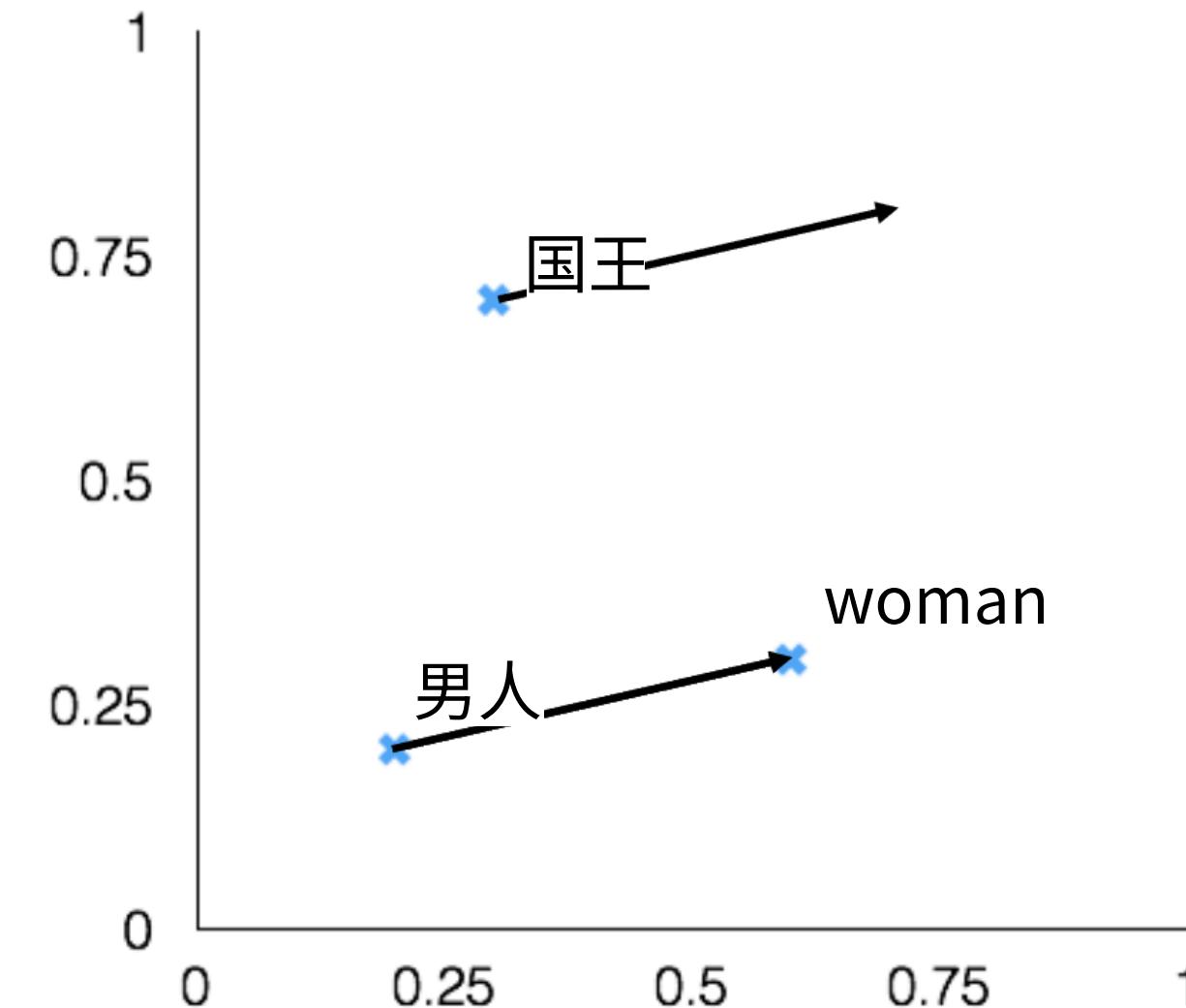
$$a:b :: c:?$$



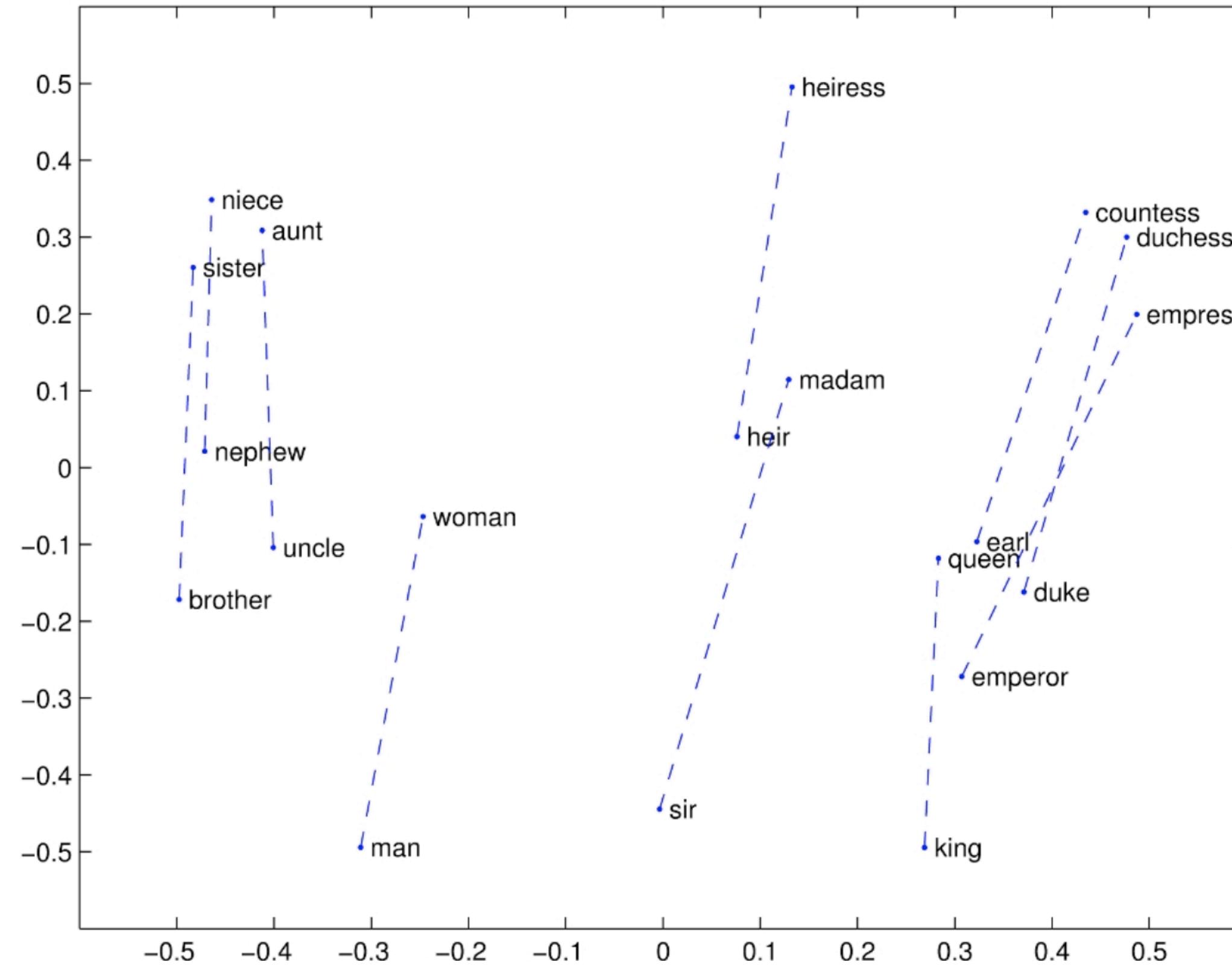
man:woman :: king:?

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

- 通过加法后余弦距离能否很好地捕捉直观的语义和句法类比问题来评估词向量
- 从搜索中丢弃输入词 (!)
- 问题：如果信息存在但不是线性的怎么办？



# GloVe 可视化



## 意义相似度：另一个内在词向量评估

- 词向量距离及其与人类判断的相关性
- 示例数据集：WordSim353 <http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/>

词 1	词 2	人类 (平均值)
老虎	猫	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
CD	stock	1.31
jaguar		0.92

ce, 与  
gy 相关性评估  
ults of a va-  
ll as with  
word2vec  
VDs. With  
SG †) 和  
) 模型在  
4 + Giga-  
0,000 最多  
大小为 10。  
e 显示在  
语料库。  
一个截断  
化了预  
的是典型  
的类型-

词向量距离及其与人类判断的相关性

Table 3: Spearman rank correlation on word similarity tasks. All vectors are 300-dimensional. The CBOWvectors are from the word2vec website and differ in that they

Model Size	WS353	MC	RG	SCWS	RW	SVD	6B
35.3	35.1	42.5	38.3	25.6	SVD-S	6B	56.5
71.0	53.6	34.7	SVD-L	6B	65.7	72.7	75.1
37.0	CBOW6B	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
39.9	GloVe 42B	75.9	83.6	82.9	59.6	47.8	58.3
CBOW100B	68.4	79.6	75.4	59.4	45.5		
C 重试	i 错误原因						

论文的想法也被证明可以改进 skip-gram (SG) 模型 (例如, 同时平均 model on this larger corpus. The fact that this basic SVD model does not scale well to large corpora lends further evidence to the

# 外部词向量评估

- 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.

[C 重试](#) [① 错误原因](#)

Table 4: F1 score on NER task with 50d

vectors. Discrete is the baseline without word vectors. We use publicly available vectors for HPCA, HSMN, and CW. See text for details.

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				

[C 重试](#) [① 错误原因](#)

- Subsequent NLP tasks in this class are other examples. So, more examples soon.

## 4.4 Model Analysis: Vector Length and Context Size

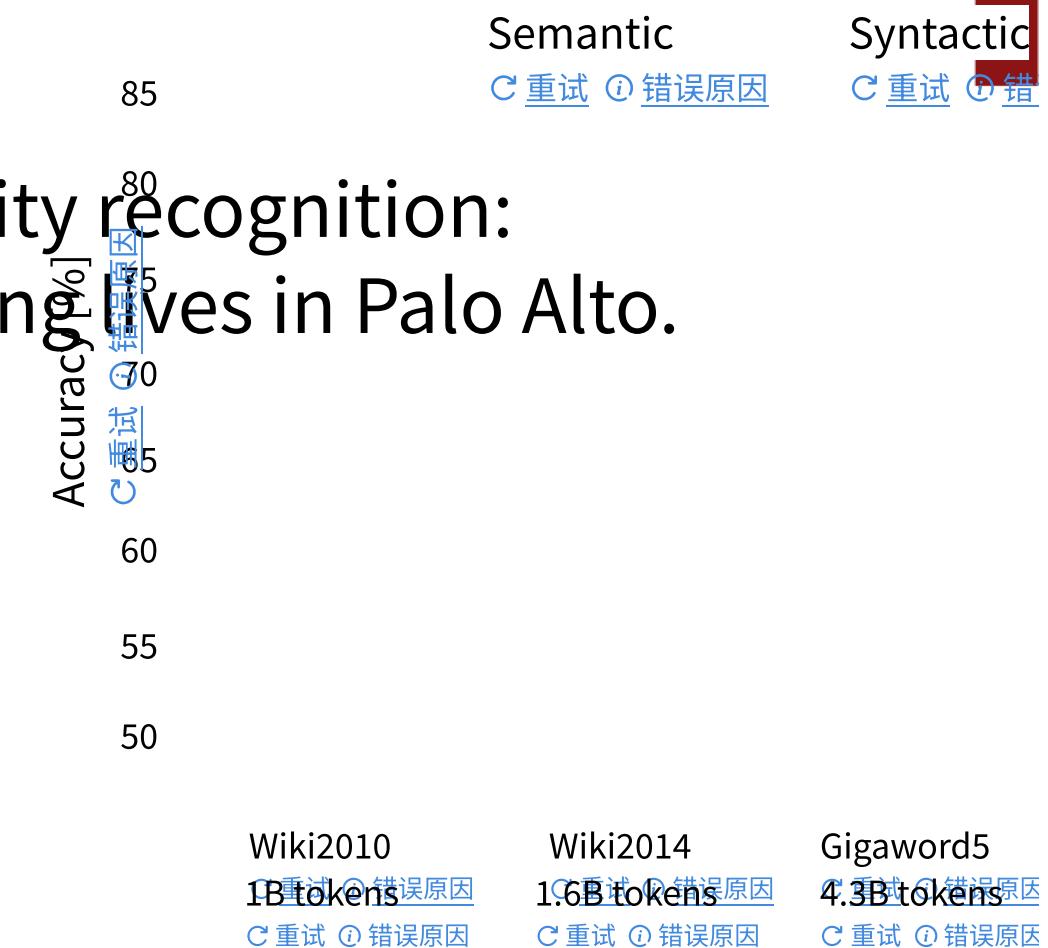


Figure 3: Accuracy on the ana dimensional vectors trained on

[C 重试](#) [① 错误原因](#)

entries are updated to assimila whereas Gigaword  
assimila whereas Gigaword

## 4.6 Model Analysis: Run-time

[C 重试](#) [① 错误原因](#)  
The total run-time is split

## 5. 词义和词义歧义

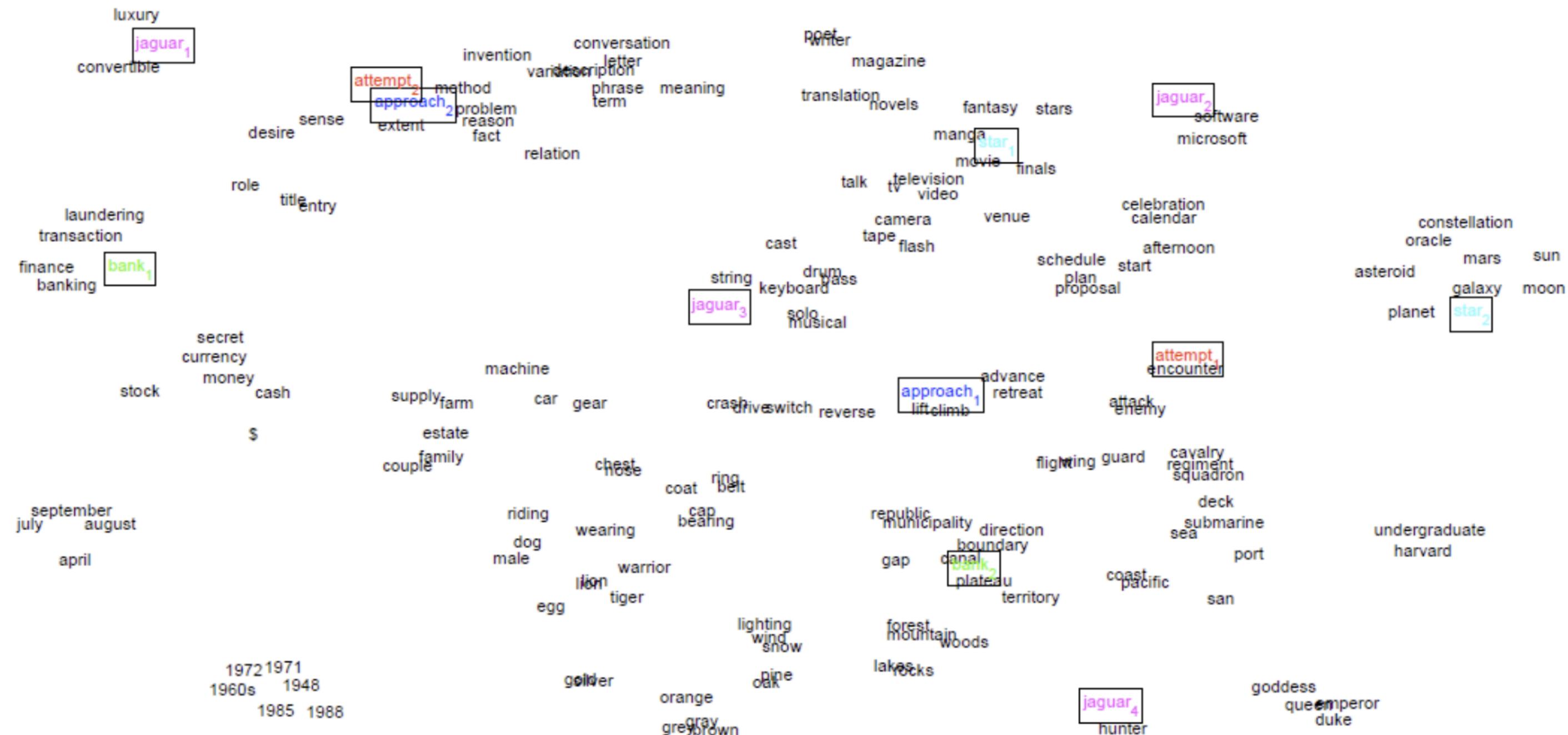
- 大多数单词有很多意思!
  - 尤其是常见的单词
  - 尤其是那些使用了很长时间的单词
- 例如: pike
- 一个向量是否捕捉了所有这些含义, 还是说我们陷入混乱?

# pike

- 尖或杆状物
- 一种长形鱼
- 铁路线或系统
- 一种道路
- 未来的（即将来临的）
- 一种身体姿势（如跳水时）
- 用长矛刺死或刺穿
- 艰难行进（沿着长矛前进）
- 在澳大利亚英语中，*pike* 意味着退出做某事：我觉得他本可以爬上那座悬崖，但他退出了！

# 通过全局上下文和多个词原型改进词表示 (黄等, 2012)

- Idea: Cluster word windows around words, retrain with each word assigned to multiple different clusters bank, bank, 等。



# Linear Algebraic Structure of Word Senses, with Applications to Polysemy

(Arora, ..., Ma, ..., TACL 2018)

C 重试 ① 错误原因

- Different senses of a word reside in a linear superposition (weighted sum) in standard word embeddings like word2vec
- $v_{\text{pike}} = \alpha v_1 + \alpha v_2 + \alpha v_3 + \alpha v_4$
- Where  $\alpha = \frac{f}{\# \text{contexts}}$ , etc., for frequency  $f$
- Surprising result:
  - Because of ideas from sparse coding you can actually separate out the senses (provided they are relatively common)!

tie				
trousers	season	scoreline	wires	operatic
blouse	teams	goalless	cables	soprano
waistcoat	winning	equaliser	wiring	mezzo
skirt	league	clinching	electrical	contralto
sleeved	finished	scoreless	wire	baritone
pants	championship	replay	cable	coloratura

## 6. Deep Learning Classification: Named Entity Recognition (NER)

⟳ 重试 ⓘ 错误原因

- The task: find and classify names in text, by labeling word tokens, for example:

⟳ 重试 ⓘ 错误原因

Last night , Paris Hilton wowed in a sequin gown .

⟳ 重试 ⓘ 错误原因 PER PER Samuel Quinn was arrested in the Hilton Hotel in Paris in April 1989 . PER PER LOC LOC LOC DATE DATE

⟳ 重试 ⓘ 错误原因

- Possible uses:
  - ⟳ 重试 ⓘ 错误原因 Tracking mentions of particular entities in documents
    - ⟳ 重试 ⓘ 错误原因 For question answering, answers are usually named entities
    - ⟳ 重试 ⓘ 错误原因 Relating sentiment analysis to the entity under discussion
  - Often ⟳ 重试 ⓘ 错误原因 followed by Entity Linking/Canonicalization into a Knowledge Base such as Wikidata

⟳ 重试 ⓘ 错误原因

# Simple NER: Window classification using binary logistic classifier

C 重试 ① 错误原因

- Idea: classify each word in its context window of neighboring words
- Train logistic classifier on hand-labeled data to classify center word {yes/no} for each class based on a concatenation of word vectors in a window
  - Really, we usually use multi-class softmax, but we're trying to keep it simple J
- Example: Classify “Paris” as +/- location in context of sentence with window length 2:

C 重试 ① 错误原因 the museums in Paris are amazing to see .

C 重试 ① 错误原因 C 重试 ① 错误原因 C 重试 ① 错误原因 C 重试 ① 错误原因

$$x_{\text{window}} = [x_{\text{museums}} \quad x_{\text{in}} \quad x_{\text{Paris}} \quad x_{\text{are}} \quad x_{\text{amazing}}]^T$$

C 重试 ① 错误原因 C 重试 ① 错误原因 C 重试 ① 错误原因 C 重试 ① 错误原因 C 重试 ① 错误原因

- Resulting vector  $x_{\text{window}} = x \in \mathbb{R}$

- To classify all words: run classifier for each class on the vector centered on each word in the sentence

# Classification review and notation

⟳ 重试 ⓘ 错误原因

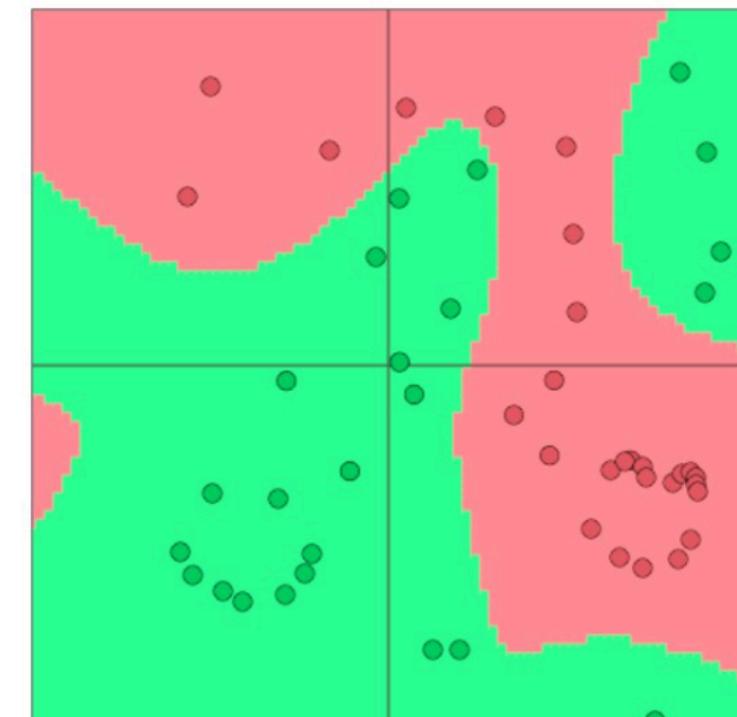
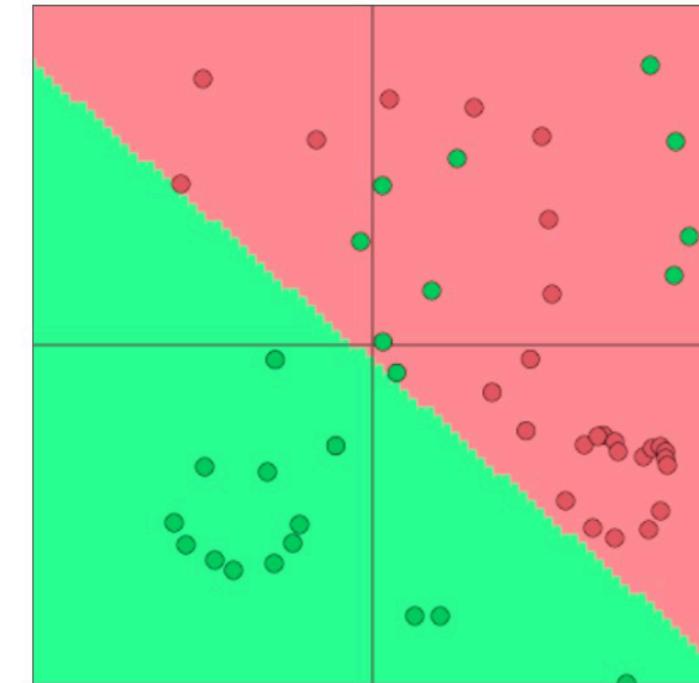
- 监督学习：我们有一个训练数据集，包含样本

$$\{x, y\}$$

- $x_i$  是输入，例如，单词（索引或向量！），句子，文档等。
  - 维度 d
- $y$  是标签（属于 C 个类别中的一个），我们尝试预测，例如：
  - 类别：情感 (+/-)，命名实体，买/卖决策
  - 其他单词
  - 后来：多词序列

# 神经分类

- 典型的 ML/统计 softmax 分类器:  $p(y|x) = \frac{\exp(W_y \cdot x)}{\sum_{c=1}^C \exp(W_c \cdot x)}$
- 学习到的参数  $\theta$  只是  $W$  的元素 (而不是输入表示  $x$ ,  $x$  具有稀疏的符号特征)
- 分类器给出线性决策边界, 这可能会受到限制
- 神经网络分类器的不同之处在于:
  - 我们同时学习词的  $W$  表示和 (分布式!) 表示
  - 词向量  $x$  重新表示了一-hot 向量, 并在中间层向量空间中移动它们, 以便使用 (线性) softmax 分类器进行简单的分类
    - 概念上, 我们有一个嵌入层:  $x = L e$
  - 我们使用深层网络——更多的层——这使我们能够多次重新表示和组合数据, 从而提供一个非线性分类器



但通常, 它相对于预最终层表示是线性的

## Softmax 分类器

$$p(y|x) = \frac{\exp(W_y \cdot x)}{\sum_{c=1}^C \exp(W_c \cdot x)}$$

再次，我们可以将预测函数拆分为三个步骤：

1. 对于  $W$  的每一行  $y$ ，计算与  $x$  的点积：

$$W_y \cdot x = \sum_{i=1}^d W_{yi} x_i = f_y$$

2. 应用 softmax 函数以获得归一化概率：

$$p(y|x) = \frac{\exp(f_y)}{\sum_{c=1}^C \exp(f_c)} = \text{softmax}(f)$$

3. 选择最大概率的  $y$

• 对于每个训练样本  $(x,y)$ ，我们的目标是最大化正确类别的概率  $y$ ，或者我们可以最小化该类别的负对数概率：

$$-\log p(y|x) = -\log \left( \frac{\exp(f_y)}{\sum_{c=1}^C \exp(f_c)} \right)$$

# NER: 中心词为地名的二分类

- 我们进行监督训练，如果是一个地点就希望得到高分

$$J\theta = b \quad s = \frac{1}{1 + e}$$

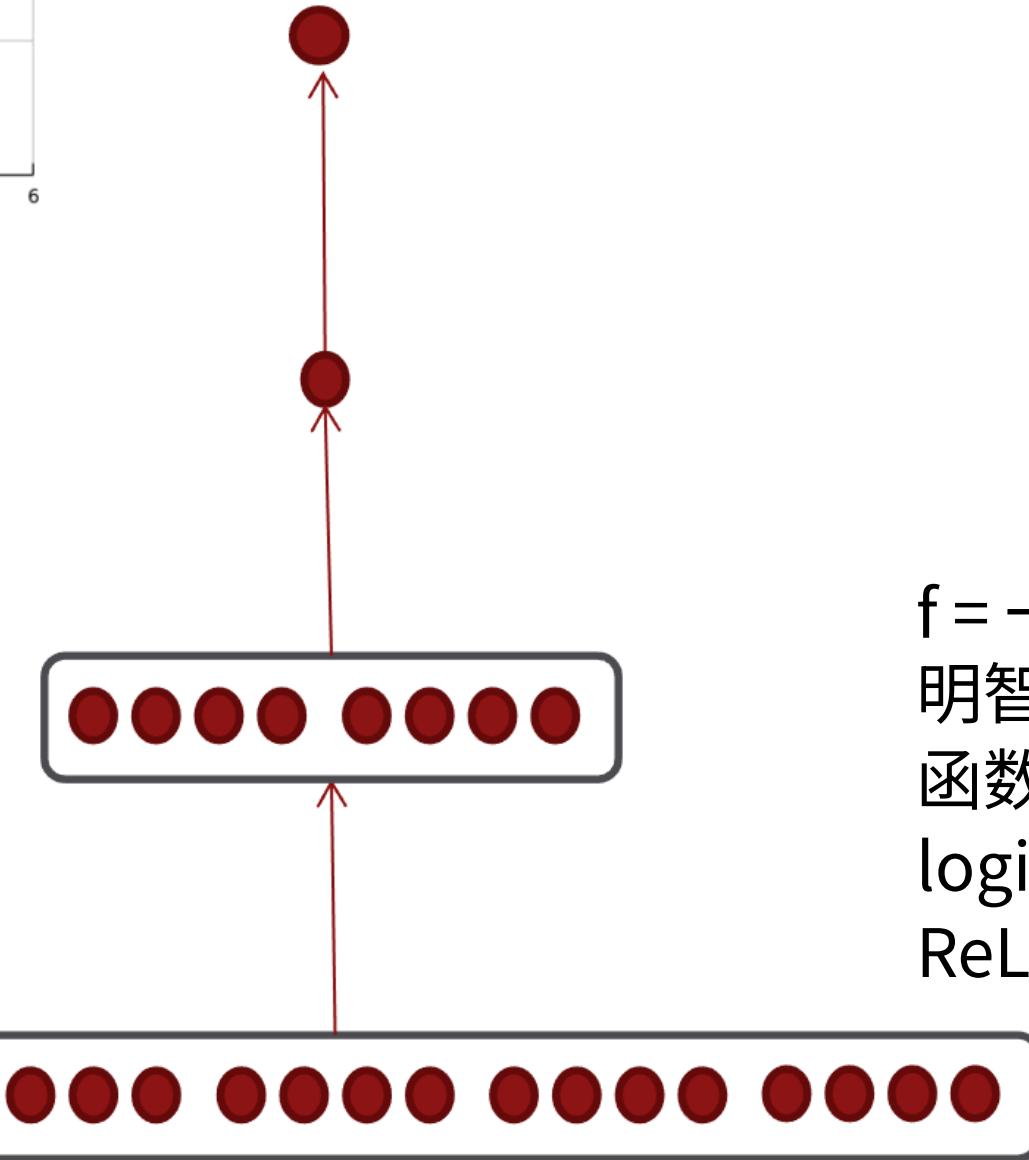
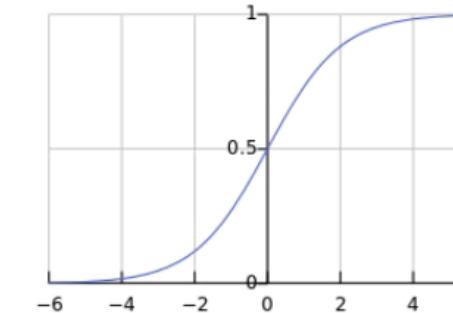
类的预测模型概率

$$s = u^T h$$

$$h = f(Wx + b)$$

$x$  (input)

$$x = [x_{\text{博物馆}} \quad \text{新} \quad x_{\text{巴黎}} \quad x_{\text{存在}} \quad x_{\text{amazing}}]$$



$f = \text{一些元素-明智的非线性函数, 例如, logistic、tanh、ReLU}$

# 训练使用“交叉熵损失”——你可以在 PyTorch 中使用这个！

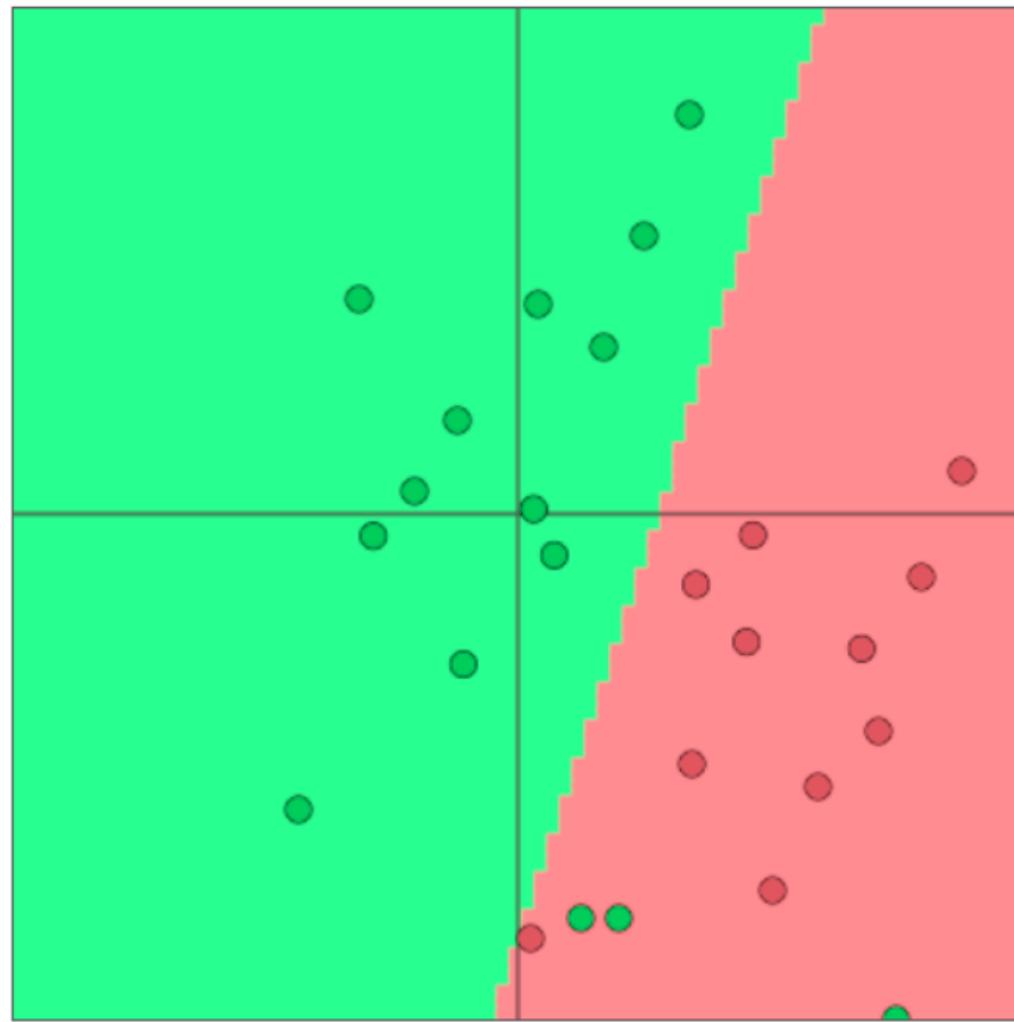
- 到目前为止，我们的目标是最大化正确类  $y$  的概率，或者等价地，我们可以最小化该类的负对数概率
- 现在用信息论中的交叉熵来重新表述
- 令真实概率分布为  $p$ ；令我们计算的模型概率为  $q$
- 交叉熵是：
$$H(p, q) = - \sum_{c=1}^C p(c) \log q(c)$$
- 假设一个真实（或正确或黄金或目标）概率分布，在正确类为 1，其他地方为 0， $p = [0, \dots, 0, 1, 0, \dots, 0]$ ，则：
- 由于  $p$  为 one-hot，唯一剩下的项是真实类  $y$  的负对数概率： $-\log p(y|x)$

交叉熵可以用其他方式与更有趣的  $p$  一起使用，但现在只需知道你将在 PyTorch 中使用它作为损失

# 对完整数据集进行分类

- 全数据集  $\{x, y\}$  的交叉熵损失函数

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N -\log \left( \frac{e^{f_{y_i}}}{\sum_{c=1}^C e^{f_c}} \right)$$



Remember: 随机梯度下降

更新公式：

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

$\alpha$  = 步长或学习率

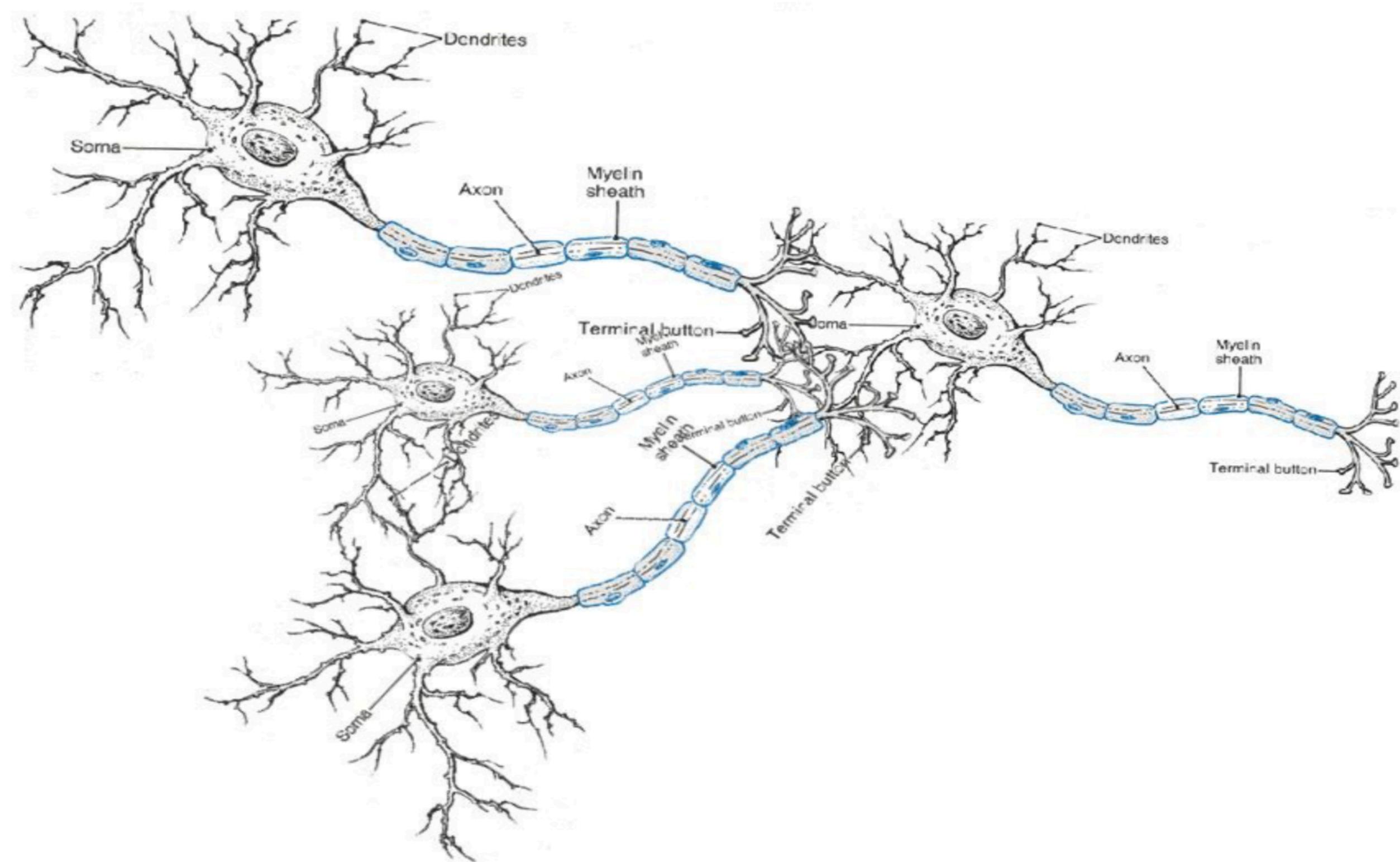
即，对于每个参数： $\theta = \theta - \frac{\text{HC}^{\circ}}{\text{HC}}$

在深度学习中， $\theta$  也包括数据表示（例如，词向量）！

我们如何计算  $\nabla J(\theta)$ ？

1. 手工计算
2. 算法上：反向传播算法（下一讲！）

## 7. 神经计算



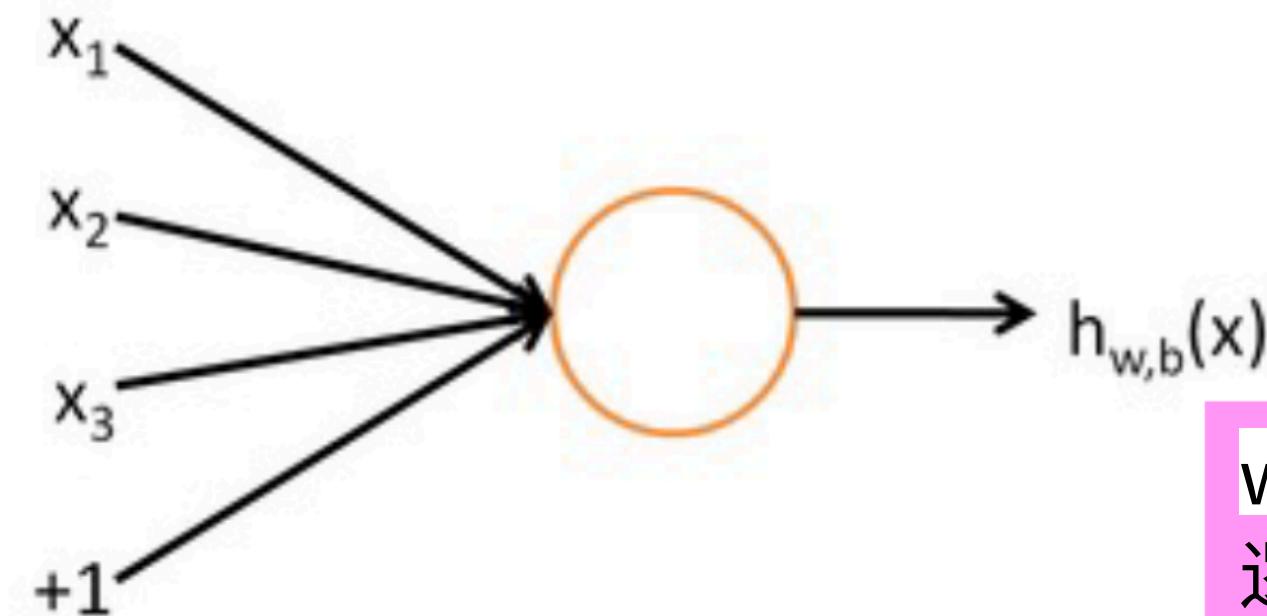
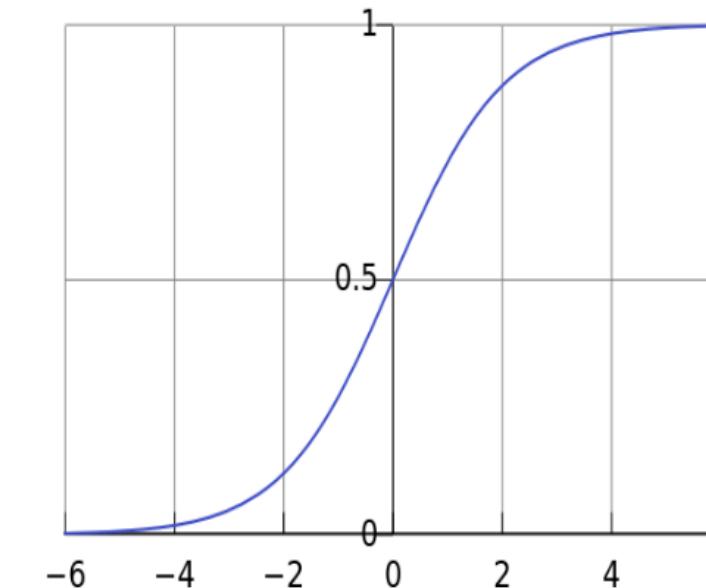
一个二元逻辑回归单元有点类似于一个神经元

$f$  = 非线性激活函数（例如，sigmoid）， $w$  = 权重， $b$  = 偏置， $h$  = 隐藏层， $x$  = 输入

$$h_{w,b}(x) = f(wx + b)$$

b: 可以有一个“始终开启”的偏置特征，提供一个先验类别，或者将其分离出来，作为偏置项

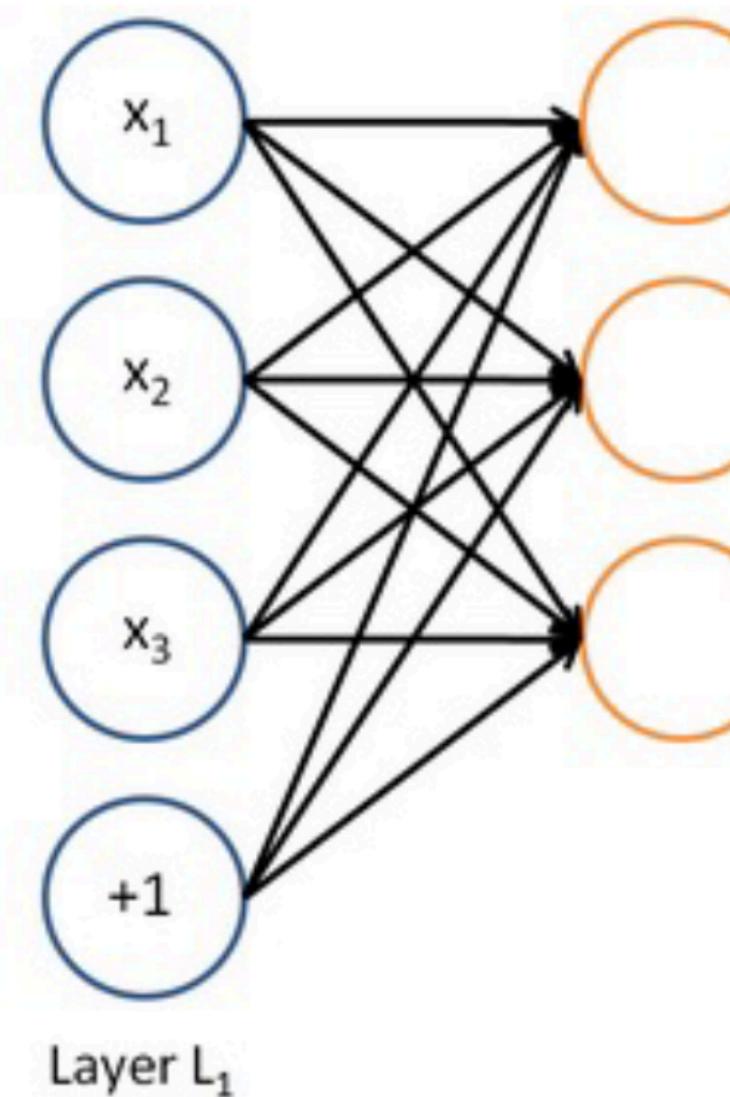
$$f(z) = \frac{1}{1 + e^{-z}}$$



$w, b$  是这个神经元的参数，即，这个逻辑回归模型的参数

# 一个神经网络 = 同时运行多个逻辑回归

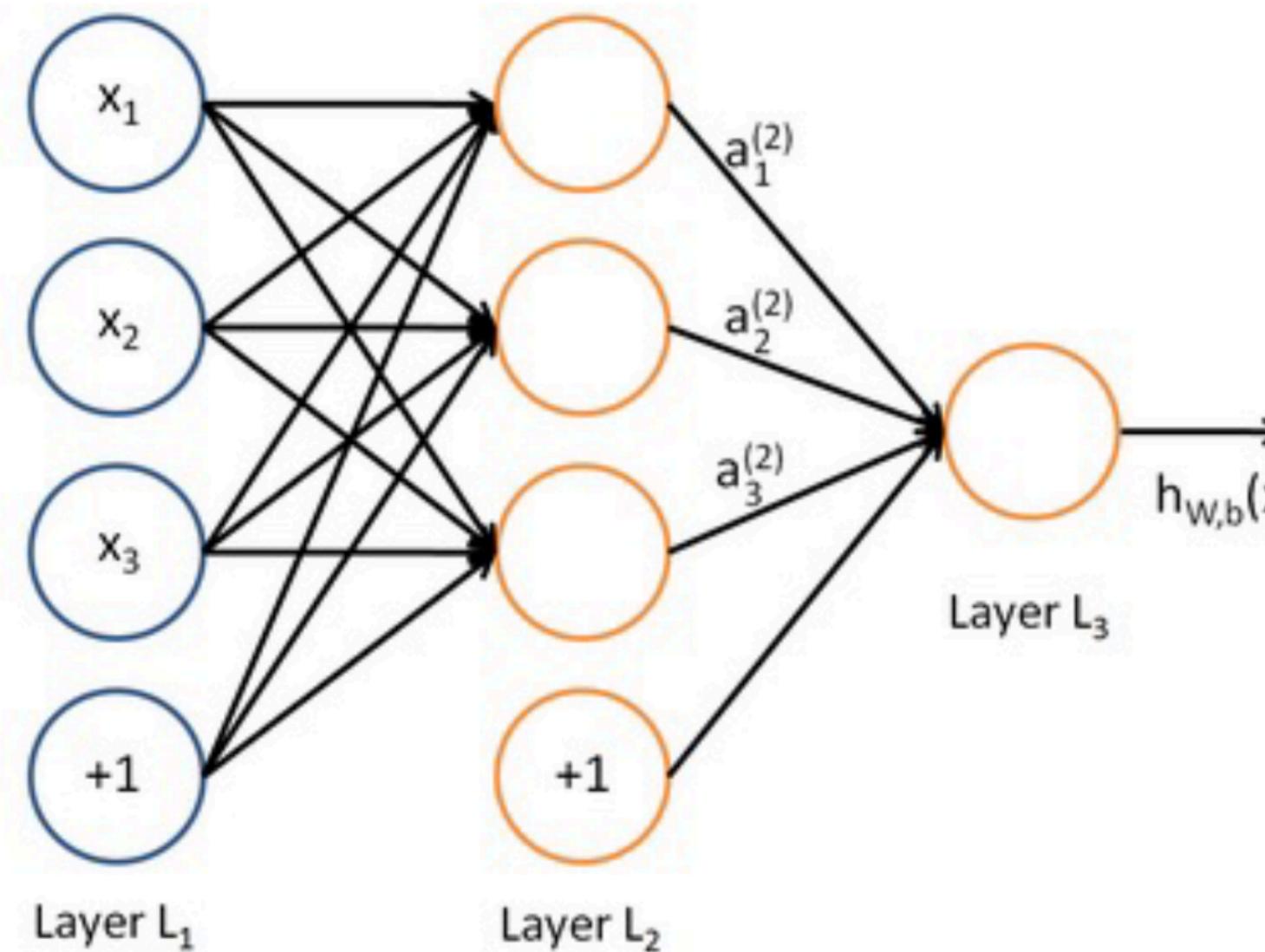
如果我们通过一堆逻辑回归函数输入一个向量，那么我们会得到一个输出向量……



但我们不需要提前决定这些逻辑回归试图预测哪些变量！

# 一个神经网络 = 同时运行多个逻辑回归

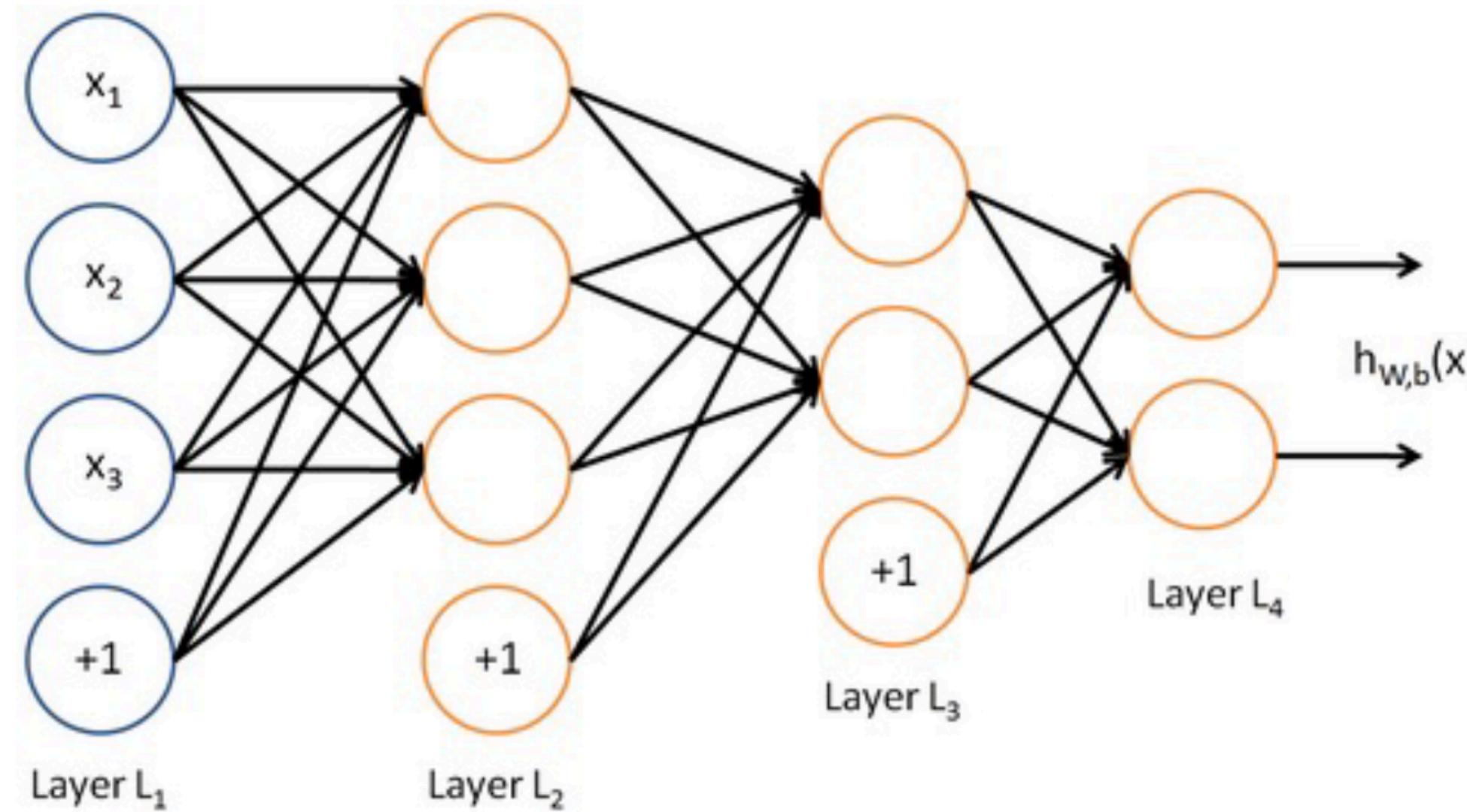
……我们可以将其输入到另一个逻辑回归函数中，生成复合函数



是损失函数将指导中间  
隐藏变量应该是什么，  
以便在预测下一层的目  
标等方面做得很好

# 一个神经网络 = 同时运行多个逻辑回归

不知不觉中，我们有了多层神经网络……



这使我们能够  
重新表示和  
组成我们的数据多次并  
在原始输入方面学习一  
个高度非线性的分类器

(但通常在预最终层表示方面是线性  
的)

# 层的矩阵表示

我们有

$$\begin{aligned} a_1 &= f(Wx + Wx + Wx + b) \\ &= f(Wx + Wx + Wx + b) \text{ 等} \\ a_2 &= \dots \end{aligned}$$

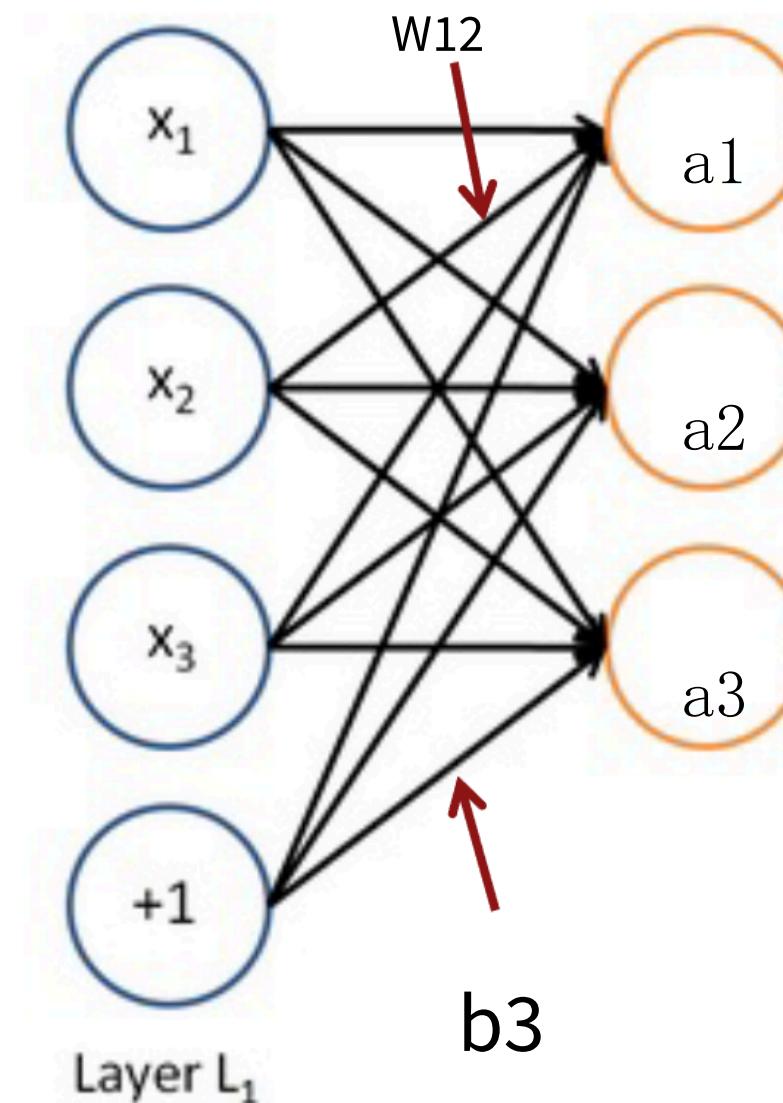
在矩阵表示中

$$z = Wx + b$$

$$a = f(z)$$

激活函数  $f$  逐元素应用：

$$f([z, z, z]) = [f(z), f(z), f(z)]$$



# 非线性函数（如 f 或 sigmoid）：为什么需要它们

- 神经网络进行函数近似，例如，回归或分类
  - 如果没有非线性操作，深度神经网络就只能进行线性变换
  - 额外的层可能只是被编译成一个单一的线性变换： $WWx = Wx$
  - 但是，如果有更多的包含非线性操作的层，它们可以近似更复杂的函数！

