- 第1课：导言与Word2Vec
- 第2课：词向量（第2部分）和词义
- 第3课：词窗口分类、神经网络、矩阵计算
- 第4课：反向传播

第1课：导言与Word2Vec

今年课程的变化

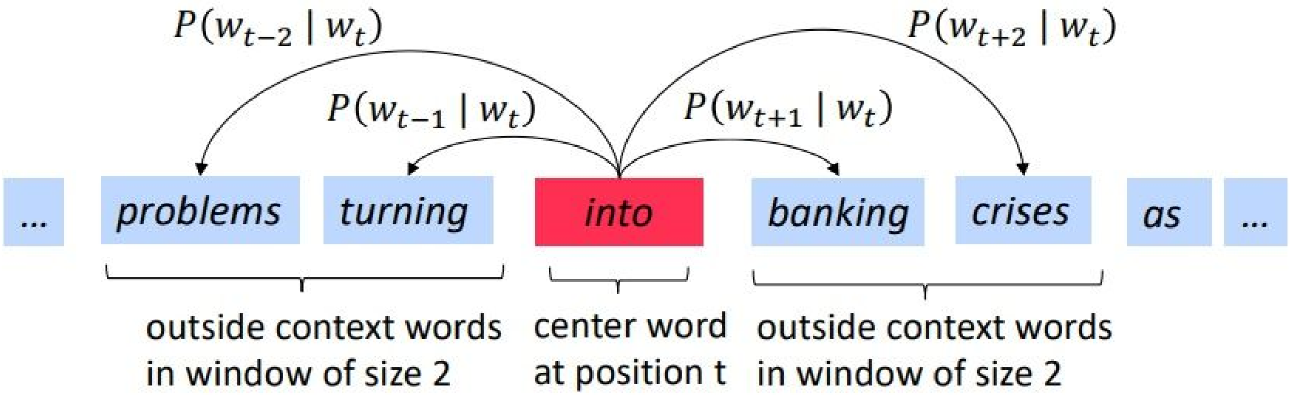
- 包含了更多新内容：字符模型、Transformer、机器学习公平性、多任务学习
- 从TensorFlow改用PyTorch

如何表达词汇的含义？

- 传统方法比如WordNet：记录每个词的同义词集合，以及多层归属关系
- 传统方法当然有一些语义上的问题，那么现代技术希望用稠密向量来表达词汇
- 词汇的含义可能与上下文相关，尤其是多义词

Word2Vec

- [Efficient Estimation of Word Representations in Vector Space (Mikolov et al. 2013)]
- Word2Vec从神经网络语言模型（NNLM）演变简化而来，主要目的为了获得词向量，而并非预测词。



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

- Update vectors so you can predict well

- 目标函数

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

θ is all variables to be optimized

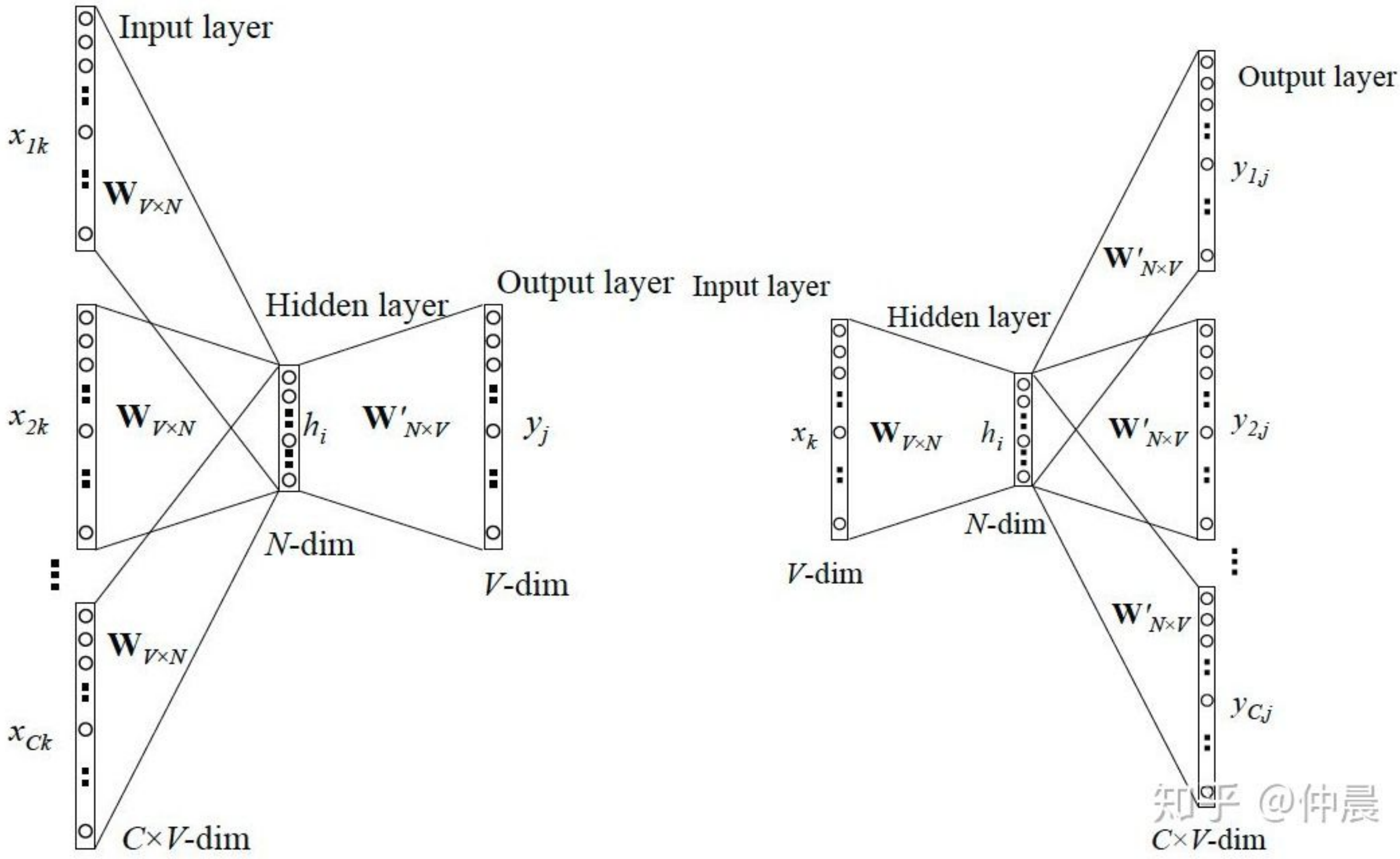
sometimes called *cost* or *loss* function

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

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

Word2Vec包括两种模型

- **CBOW**：用上下文预测这个词
- **Skip-Gram**：预测一个词的上下文



配套的优化方法（梯度下降、SGD）等等。

第2课：词向量（第2部分）和词义

Word2Vec的训练优化，通过两种方式降低计算量：

- hierarchical softmax：本质是把 N 分类问题变成 log(N)次二分类
- negative sampling：本质是预测总体类别的一个子集
  - [Distributed Representations of Words and Phrases and their Compositionality]

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

negative sampling

$P(w)=U(w)^{3/4}/Z$,
the unigram distribution U(w) raised to the 3/4 power
(We provide this function in the starter code).
The power makes less frequent words be sampled more often

negative sampling

词向量两个方向：

- 分解词语共现矩阵，即count based方法：LSA(SVD), HAL；COALS, Hellinger-PCA
- direct prediction方法：Skip-gram/CBOW；NNLM, HLBL, RNN

Global Vectors for Word Representation(GloVe)

- [Encoding meaning in vector differences (Pennington, Socher, and Manning, EMNLP 2014)]
- 基于count based方法，比word2vec效果好一些
- X为共现矩阵，P(x|a) = X(x,a) / X(a)
- P(x|a)/P(x|b)，通过比例来学习a与b哪个和x更加相关或者无关。

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(x\|\text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5×10^{-2}	1.36	0.96

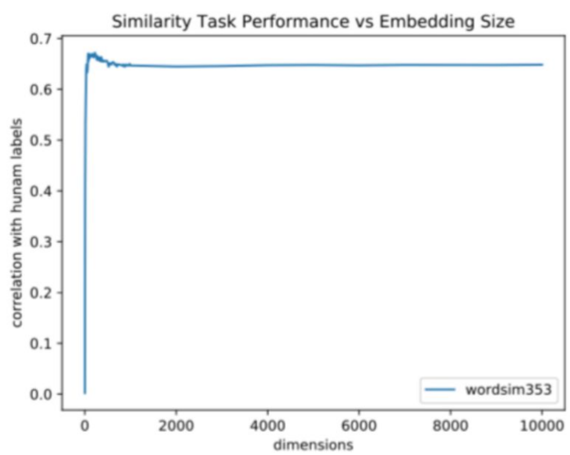
- 为了数学好算，w(i)·w(j) = logP(i|j)

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

GloVe优化目标

对Word Embedding维度的研究

- [On the Dimensionality of Word Embedding (Zi Yin and Yuanyuan Shen, NeurIPS 2018)]



应对多义词问题

- [Improving Word Representations Via Global Context And Multiple Word Prototypes (Huang et al. 2012)]
  - 根据窗口词，将多义词分拆
- [Linear Algebraic Structure of Word Senses, with Applications to Polysemy (Arora, ..., Ma, ..., TACL 2018)]

Python学习（略）

第3课：词窗口分类、神经网络、矩阵计算

分类问题：

- softmax
- 交叉熵
- 线性分类器与非线性分类器

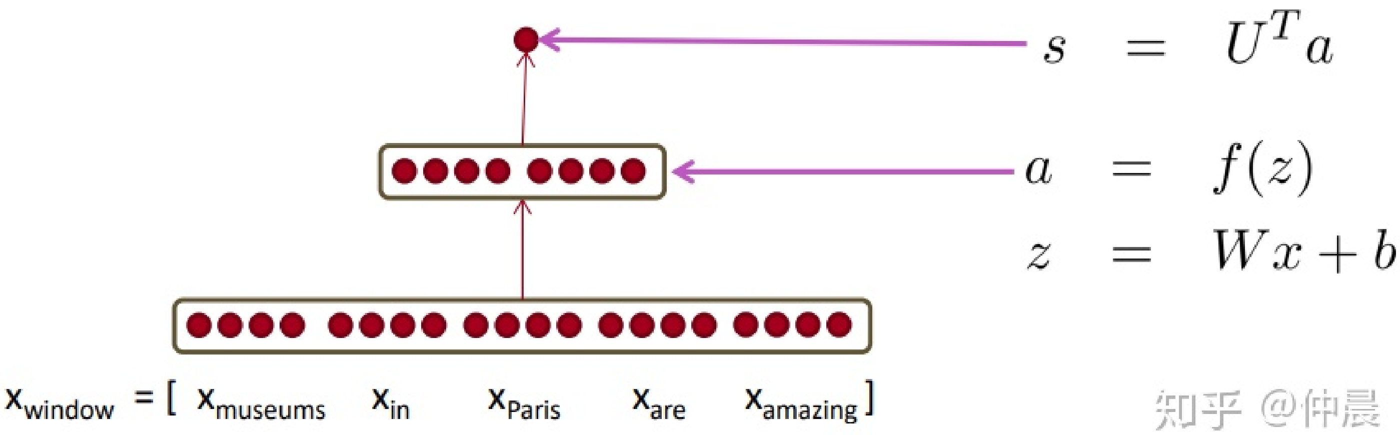
神经网络，非线性分类器的强大表达能力

命名实体识别（NER）问题

- 找出句子中实体词，并标注出是哪一类的实体词。

NER尝试之一：**Window Classification 窗口分类**

- 取一个word window拼接成向量，进行分类。
- max-margin loss：J = max(0, 1-s+s_c），主要为了正例计算分数尽量大于负例。



导数Jacobian

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \qquad \boxed{\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{ij} = \frac{\partial f_i}{\partial x_j}}$$

对于 = f( )情况

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} = \begin{pmatrix} f'(z_1) & & 0 \\ & \ddots & \\ 0 & & f'(z_n) \end{pmatrix} = \mathrm{diag}(\boldsymbol{f}'(\boldsymbol{z}))$$

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{W}$$

$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I} \quad \text{(Identity matrix)}$$

$$\frac{\partial}{\partial \boldsymbol{u}}(\boldsymbol{u}^T \boldsymbol{h}) = \boldsymbol{h}^{\boldsymbol{T}}$$

Fine print: This is the correct Jacobian. Later we discuss the "shape convention"; using it the answer would be h.

链式法则

For multiple variables at once: **multiply Jacobians**

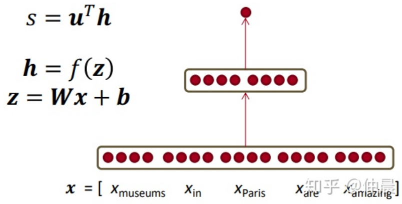
$$\boldsymbol{h} = f(\boldsymbol{z})$$

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \ldots$$

举例：完整的导数推导，计算s对W和b的梯度

$$\frac{\partial s}{\partial \boldsymbol{W}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}}$$



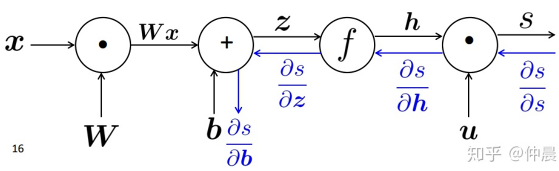
采用链式法则直接矩阵相乘，会产生一些奇怪情况，还是应该以微分项的单个元素作为考虑对象

$$\frac{\partial s}{\partial W_{ij}} = \underbrace{\delta_i}_{\text{Error signal from above}} \underbrace{x_j}_{\text{Local gradient signal}}$$

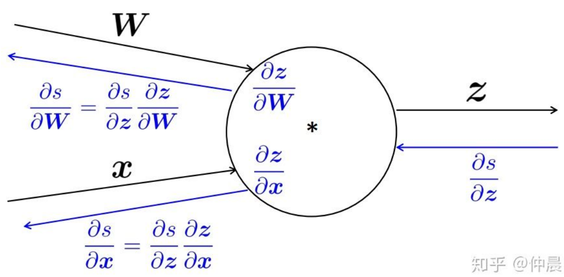
$$\begin{array}{ccccc} \dfrac{\partial s}{\partial \boldsymbol{W}} & = & \boldsymbol{\delta}^T & & \boldsymbol{x}^T \\ [n \times m] & & [n \times 1] & & [1 \times m] \end{array}$$

第4课：反向传播

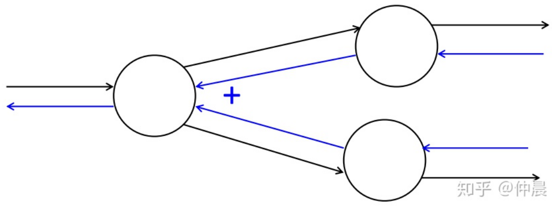
基于链式法则，向后反向传播



多并一



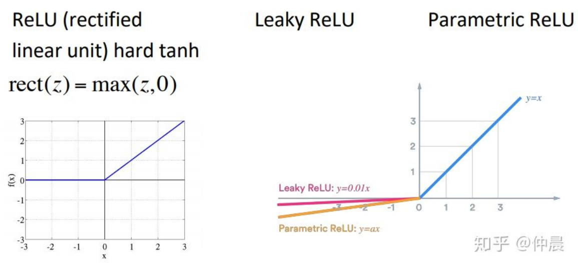
一拆多



正则化

$$J(\theta)=\frac{1}{N}\sum_{i=1}^{N}-\log\left(\frac{e^{f_{y_i}}}{\sum_{c=1}^{C}e^{f_c}}\right)\boxed{+\lambda\sum_{k}\theta_k^2}$$

激活函数



参数初始化

Xavier initialization has variance inversely proportional to fan-in n_{in} (previous layer size) and fan-out n_{out} (next layer size):

$$\mathrm{Var}(W_i)=\frac{2}{n_{\text{in}}+n_{\text{out}}}$$

优化器与学习率

These models give per-parameter learning rates

- Adagrad
- RMSprop
- Adam ← A fairly good, safe place to begin in many cases
- SparseAdam
- ...