0 目录

1 教学目标

1. 了解有效的现代深度学习方法

   首先学习基础知识，然后学习一些NLP领域中重要的方法：RNN，以及attention机制等等

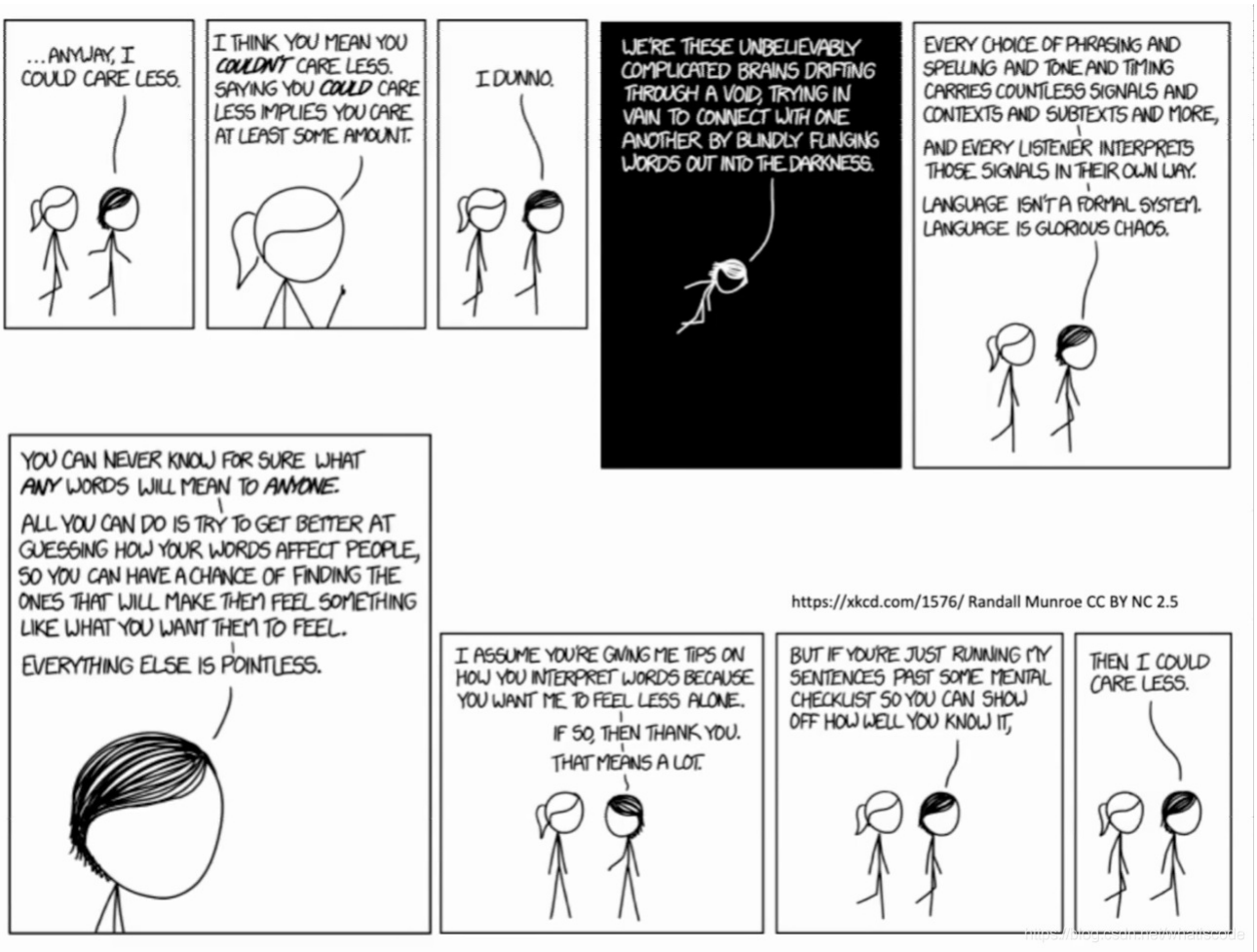
2. 对人类语言的全局把握以及理解和形成人类语言的一些困难

3. 理解并掌握怎样去构造一个系统（PyTorch）来解决NLP领域中的一些主要的问题：

   - Word Meaning
   - Dependency Parsing
   - Machine Translation
   - Question Answering

2 人类语言和词义

2.1 一张XKCD漫画



- 语言带有不确定性
- 每一句话的措辞、拼写、语气以及说话的时间都带有无数的信号和上下文信息，并且每一个听众都以自己的方式去理解这些信号

2.2 人类语言

- 人类和大猩猩最大的区别就是人类拥有**语言系统**
- 人类大概在离开非洲时（约10万年前）拥有了语言
- 语言系统让人类能够更加有效的进行团队合作，这也是人类能够在自然界立于不败之地的原因
- 写作（writing）也是一项具有开创性的发明（5000年前）
- 写作让知识得以传承下去
- 语言系统能够起作用的原因是因为人类具有相似的知识储备。当我们用自然语言对一个场景进行描述时，听者在脑海中可以构建出相似的视觉场景。

2.3 我们怎么样表示一个词的意思（meaning）？

meaning的定义（韦伯斯特词典）：

- 用单词、词组表示概念
- 人们运用单词、符号表示自己的观点
- 通过写作作品、艺术来表达观点

signifier(symbol)⇔signified(idea or thing)

2.4 我们怎么在计算机上表示词义

2.4.1 WordNet

一个包含同义词（synonym）集合和上义词（hypernyms）的词典。

【上位词】：用 “is a”来表示关系的词集列表

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))
```

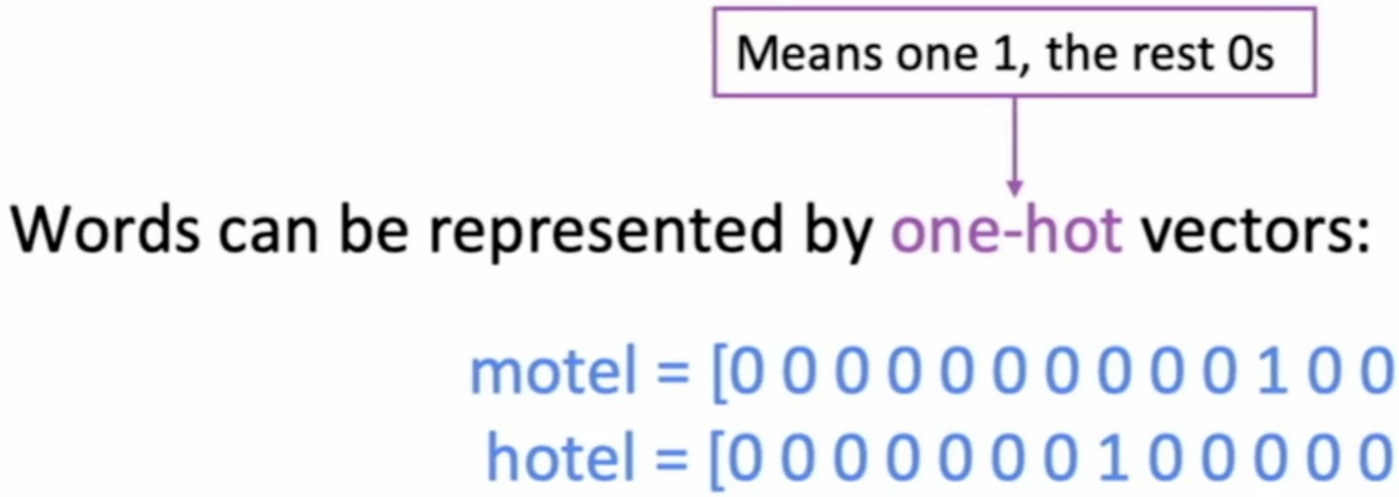
```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),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


Wordnet存在的问题：

- 作为一个较好的资料，但忽略了细微的一些差别：例如词典中‘proficient’与‘good’认为是同义词，但是这只在某些文本上下文中成立。
- 忽略了一些单词的新的含义（无法随时更新）
- 比较主观（缺少客观性）
- 需要人类劳动力来不断地创建和更新
- 不能计算单词之间的相似度

2.4.2 Representing words as discrete symbols

传统NLP中，我们将单词看作是离散的表示，通过one-hot vector来表示：



One-hot表示存在的问题：

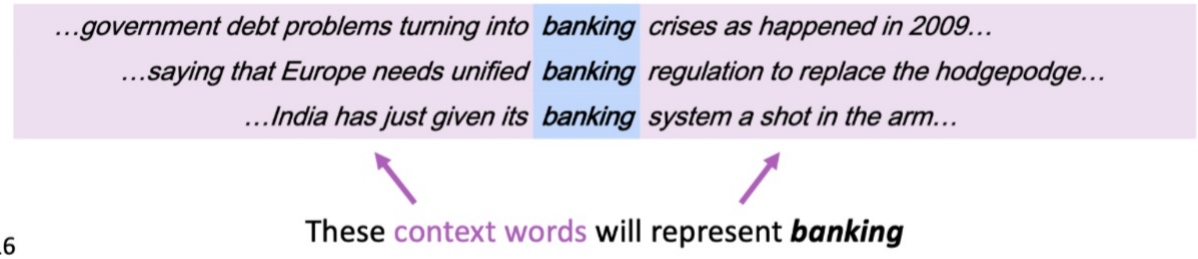
- 如果词汇量太多，向量的维度会变得很大。
- 由于向量是正交的，因此没法表示他们之间啊的相似度。

2.4.3 Representing words by their context

Distributional semantics：一个单词的含义通常由在它附近经常出现的单词给出的。

对于文本中的一个单词w，它的上下文就是出现在它附近的一组单词（在一个划定好size的窗口下）

通过许多包含w的文本中的上下文来构建w的含义表示：



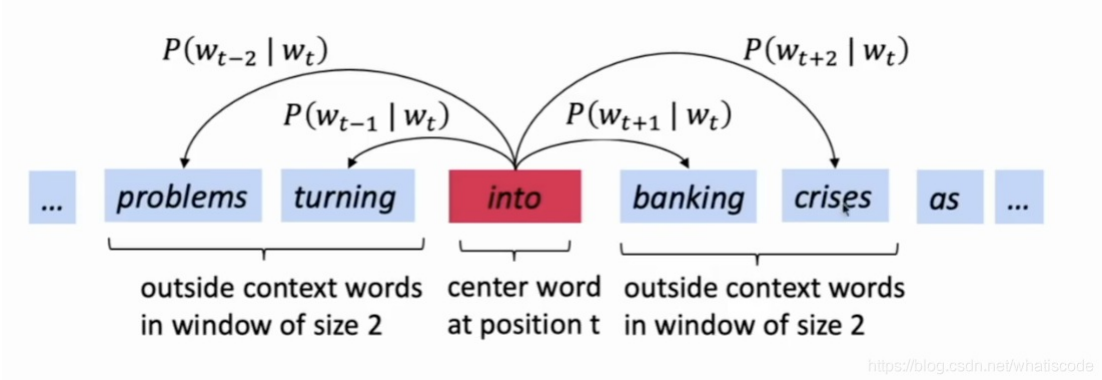
3 Word2vec介绍

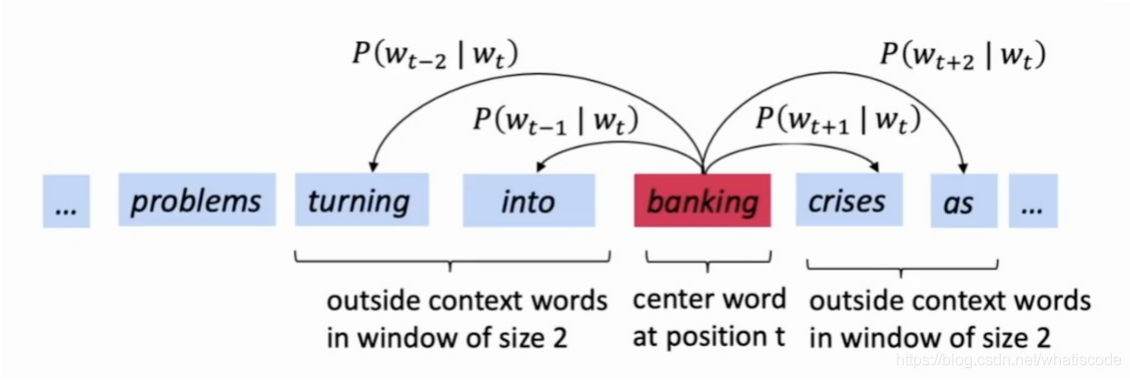
Word2vec是一种学习词向量的框架。

3.1主要思想

- 包含大量的文本语料
- 固定词表中的每一个单词由一个词向量表示
- 文本中的每个单词位置 t，有一个中心词c，和它的上下文 o（除了 c 的外部单词）。
- 通过 c 和 o 的词向量相似性来计算 P（o|c）
- 不断的调整词向量，最大化概率

3.2 计算 P (w t + j ｜ w t) P(w_{t+j}|w_t) P (wt+j ｜ wt)的示例





3.3 Word2vec的目标函数

对于每个位置 t = 1,…,T，固定窗口大小m，给定中心词 w j w_j wj:

l i k e l i h o o d = L (θ) = ∏ t = 1 T ∏ − m ≤ j ≤ m　　j ≠ 0 p (w t + j ｜ w t ; θ) likelihood=L(\theta)=\prod_{t=1}^{T}\prod_{-m\le j \le m\\\\\j\ne0}^{}p(w_{t+j}|w_t;\theta)
likelihood = L(θ) = t=1∏T −m≤j≤m　j□=0∏p(wt+j ｜ wt;θ)

- θ \theta θ 是需要优化的参数

J （ θ ） = − 1 T l o g L (θ) = − 1 T ∑ t = 1 T ∑ − m ≤ j ≤ m　　j ≠ 0 l o g P (w t + j ｜ w t ; θ) J（\theta）=-\frac 1TlogL(\theta)=-\frac1T\sum_{t=1}^^T\sum_{-m\le j\le m\\\\\j\ne 0}logP(w_{t+j}|w_t;\theta)
J（θ） = −T1logL(θ) = −T1t=1∑T −m≤j≤m　j□=0∑logP(wt+j ｜ wt;θ)

- J (θ) J(\theta) J(θ)为损失函数（这里是平均负对数似然）；
- 负号将最大化损失函数转化为最小化损失函数；
- log函数方便将乘法转化为求和（优化处理）

3.4 如何计算 P (w t + j ｜ w t ; θ) P(w_{t+j}|w_t;\theta) P (wt+j ｜ wt;θ)

对于每个单词 w 我们使用两个向量 v w v_w vw 和 u w u_w uw

- v w v_w vw 表示当 w 是中心词时
- u w u_w uw 表示当 w 是上下文单词时
  对于中心词 c 和上下文单词 o，有： P (o ｜ c) = e x p (u o T v c) ∑ w ϵ V e x p (u w T v c) P(o|c)=\frac {exp(u_o^Tv_c)}{\sum_{w\epsilon V}exp(u_w^Tv_c)} P(o ｜ c) = ∑wϵV exp(uwTvc)exp(uoTvc)

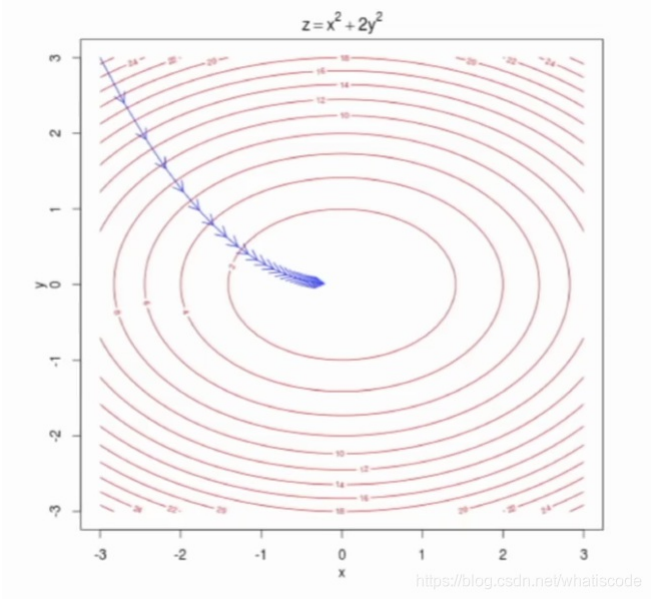
在上式中：

- 分子取幂函数使得始终可以为正
- 向量 u o u_o uo 和向量 v c v_c vc 点乘，点乘结果越大，向量之间越相似
- u T v = u · v = ∑ i = 1 n u i v i u^Tv=u·v=\sum_{i=1}^nu_iv_iuTv = u· v = ∑i=1nuivi
- 对整个词表标准化，给出概率分布
- softmax函数进行归一化（深度学习中常用）： R n → R n \Bbb{R^n}\to \Bbb{R^n} Rn→Rn s o f t m a x （ x ） = e x p (x i) ∑ j = 1 n e x p (x j) = p i softmax（x）=\frac {exp(x_i)}{\sum_{j=1}^nexp(x_j)}=p_i softmax（x） = ∑j=1nexp(xj)exp(xi) = pi 注：用于将任意值 x i x_i xi 映射到概率分布 p i p_i pi

4 Word2vec目标函数梯度

4.1 训练模型

通过调整参数的方式来最小化损失函数



4.1.1 训练模型的方法：计算所有的向量梯度

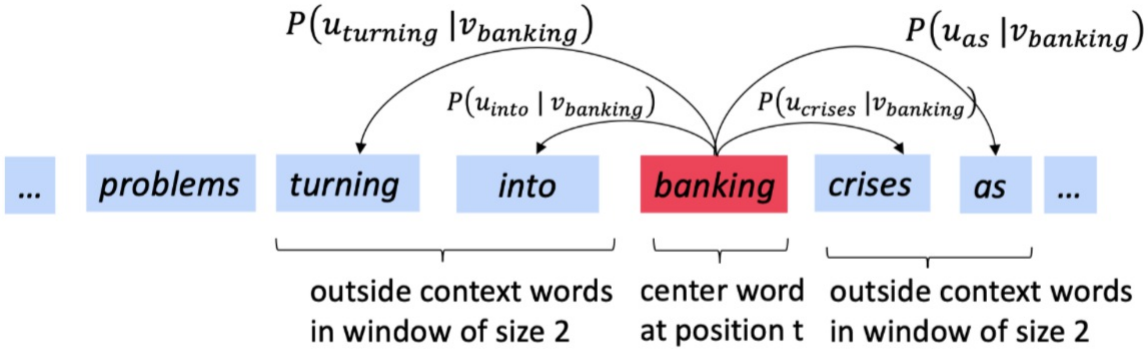
由于整个模型里只有一个参数 θ \theta θ ,所以我们只要优化这一个参数就行。如一个 d 维，词典大小为 V 的模型所包含的参数（每个单词有两个向量）：

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

我们可以通过梯度下降的方式优化参数，梯度下降会用到链式法则。

- 迭代计算每个中心词向量和上下文词向量随着滑动窗口移动的梯度
- 依次迭代更新窗口中所有的参数

示例：



31

5 优化方法

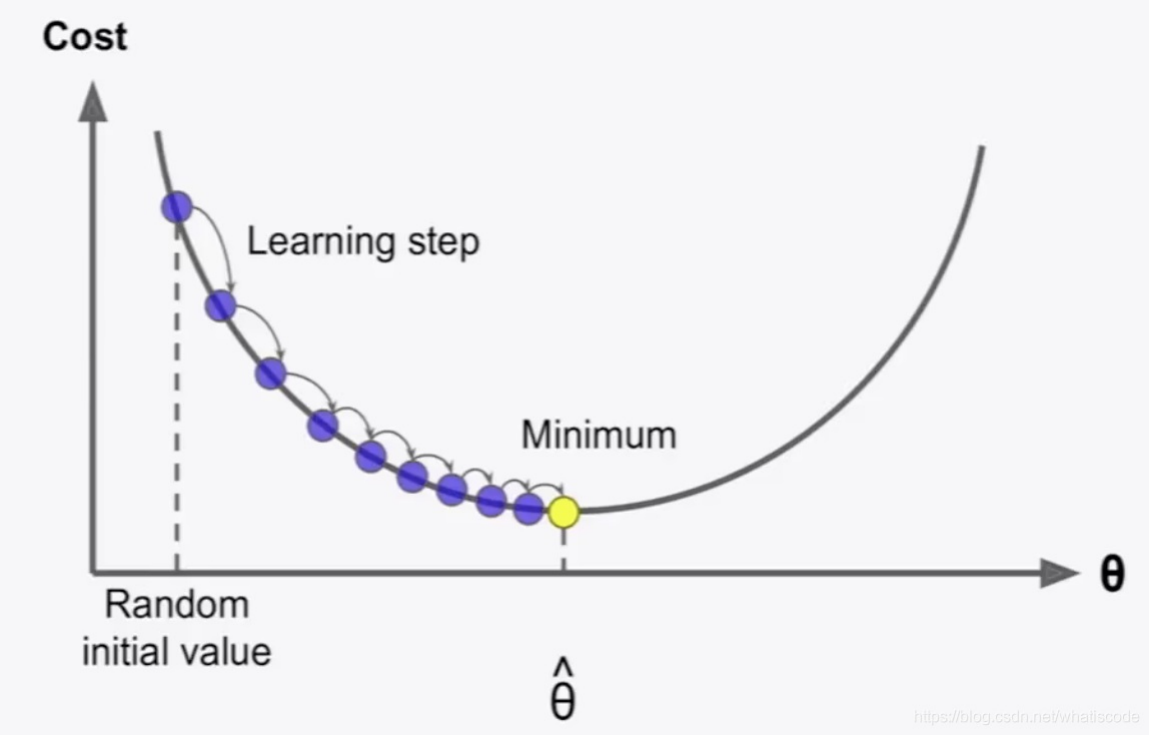
5.1 梯度下降法

最小化损失函数 J (θ) J(\theta) J(θ)

对于当前 θ \theta θ ,计算 J (θ) J(\theta) J(θ) 的梯度

然后小步重复朝着负梯度方向更新方程里的参数 α = (s t e p   s i z e)   o r   (l e a r n i n g   r a t e) \alpha=(step\ size)\ or\ (learning\ rate) α = (step size) or (learning rate)
θ n e w = θ o l d − α ∇ θ J (θ) \theta^{new}=\theta^{old}-\alpha \nabla_\theta J(\theta) θnew = θold − α∇θJ(θ)

更新唯一的参数 θ \theta θ: θ j n e w = θ j o l d − α α α   θ j o l d J (θ) \theta_j^{new}=\theta_j^{old}-\alpha \frac \alpha{\alpha\ \theta_j^{old}}J(\theta) θjnew = θjold − αα θjoldαJ(θ)




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

5.2 随机梯度下降SGD

由于 J (θ) J(\theta) J(θ) 是在语料文本中所有窗口的方程

当语料很大的时候，计算梯度会消耗巨大

解决办法：SGD

不断sample窗口，不断更新

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