简介：Word2vec是主流词嵌入方法，它不再统计计算词w在邻域中出现s的频率，而是训练一个分类器去预测w邻域中出现s的概率，最终根据分类器的权重构建词向量

Skip-gram模型

任务：给定目标词，预测它的上下文

原理：词相似→上下文相似

不直接算共现，而是做一个分类器区分"是否是真的上下文"

最终分类器权重 = 词向量

算法：1. 定窗口

设窗口大小 L（如±2），取目标词周围的上下文

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 target c3 c4

2、构造正负样本

正样本：选邻域一个词作正样本（真正的上下文）

负样本：在词表中随机选一个词作负样本（噪声）

3、训练二分类器

给定一个词对 (t, c)　t = target　c = context

训练一个分类器，返回 c 是 t 上下文的概率

c是上下文：$P(+1 \mid w, c)$

c不是上下文：$P(-1 \mid w, c) = 1 - P(+1 \mid w, c)$

注：w是t的词向量，c是c的词向量

在训练开始前，这两个向量的值是随便赋的

如何计算 $P(+1 \mid w, c)$？用向量内积度量相似，即 Similarity(w, c)，而 $\text{Similarity}(w, c) \propto w \cdot c$，直接简化为

w·c（实际上Word2vec会对向量作归一化处理，所有向量模长都为1）然后再用sigmoid函数转化为概率，即

$$P(+|w,c) = \sigma(c\cdot w) = \frac{1}{1+\exp(-c\cdot w)}$$

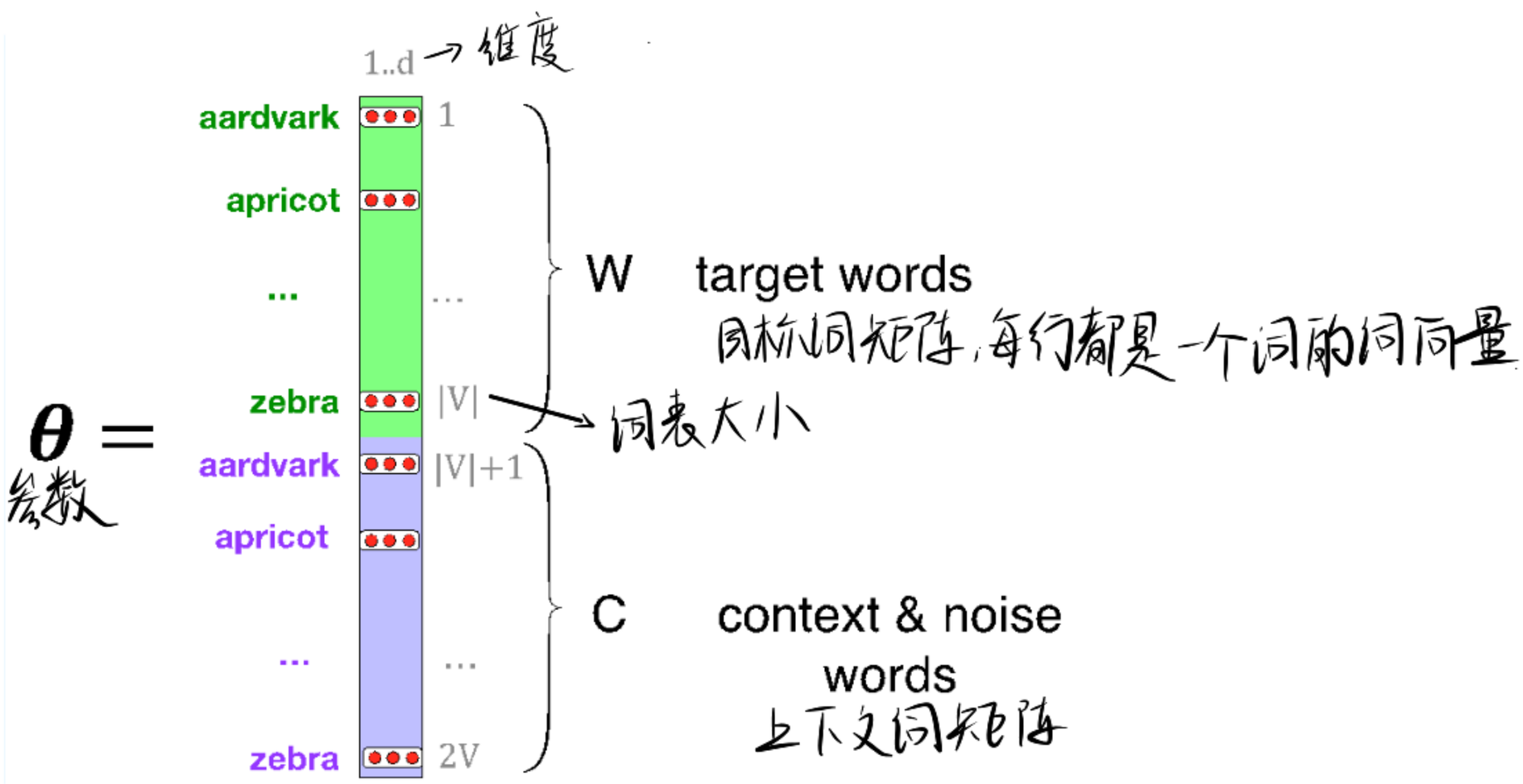
$$P(-|w,c) = 1 - P(+|w,c) = \frac{1}{1+\exp(c\cdot w)}$$

假设所有上下文词语相互独立

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L}\sigma(c_i\cdot w)$$ $c_{1:L}$是窗口内L个上下文词

对数化：$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L}\log\sigma(c_i\cdot w)$$

接下来开始训练模型：



对于一个目标词，我们选择一定数量的正样本，对于每个正样本创建k个负样本

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 t c3 c4

k=2

positive examples +		negative examples -			
t	c	t	c	t	c
apricot	tablespoon	apricot	aardvark	apricot	twelve
apricot	of	apricot	puddle	apricot	hello
apricot	preserves	apricot	where	apricot	dear
apricot	or	apricot	coaxial	apricot	forever

给定正负训练样本和初始化的词嵌入向量（即θ中的W矩阵与C矩阵，初始时所有值都是随机的），不断调整

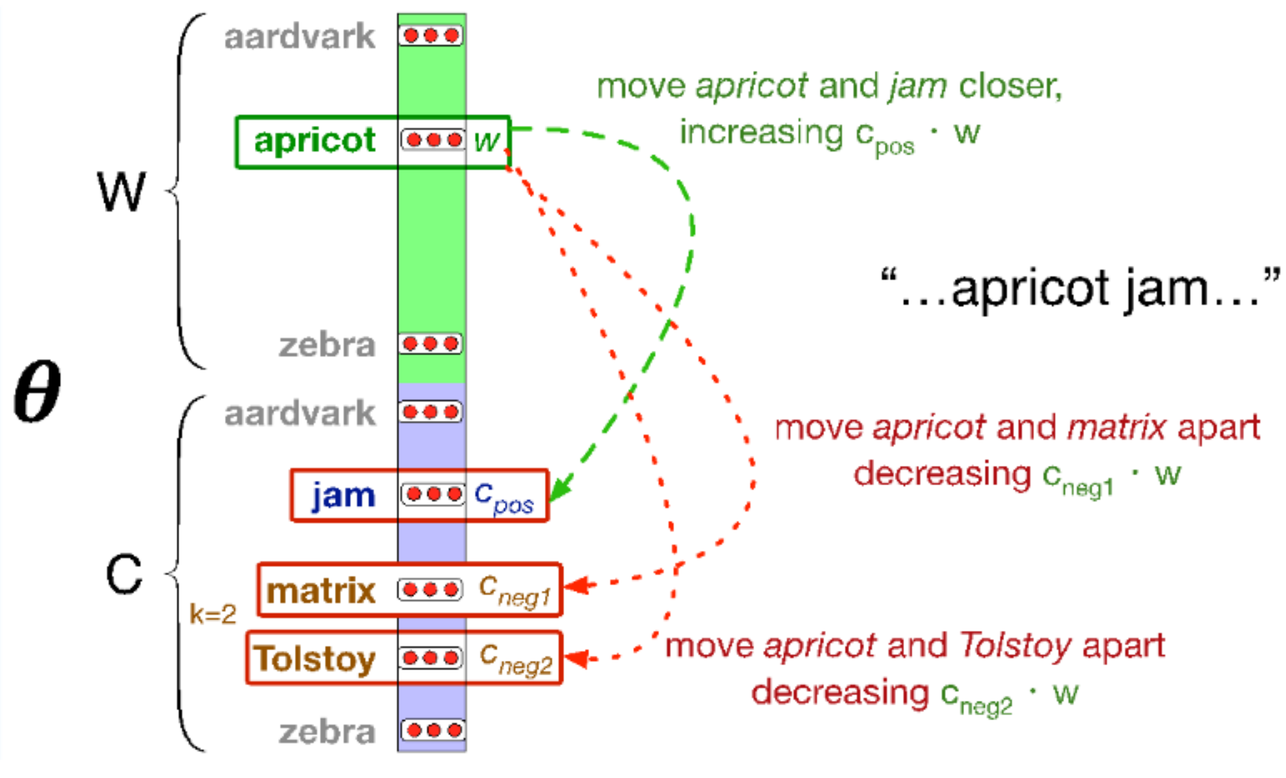
W与C矩阵中的参数使得最大化(w, c_{pos})相似度，最小化(w, c_{neg})相似度，即最大化$P(+1|w, c_{pos})$与$P(-1|w, c_{neg})$

定义损失函数：$L_{CE} = -\log [P(+1|w, c_{pos}) \prod_{i=1}^{k} P(-1|w, c_{negi})]$

$$= -[\log P(+1|w, c_{pos}) + \sum_{i=1}^{k} \log P(-1|w, c_{negi})]$$

$$= -[\log P(+1|w, c_{pos}) + \sum_{i=1}^{k} \log (1 - P(+1|w, c_{negi}))]$$

$$= -[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{negi} \cdot w)]$$



通过随机梯度下降法调整词的权重(即词向量)最小化L_{CE}

$$w^{t+1} = w^t - h \frac{d}{dw} L(f(x; w), y)$$

而对于这里的L_{CE}，梯度为：

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(c_{pos} \cdot w) - 1] \cdot w$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(c_{neg} \cdot w)] \cdot w$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(c_{pos} \cdot w) - 1] c_{pos} + \sum_{i=1}^{k} [\sigma(c_{negi} \cdot w)] c_{negi}$$

SGD更新为：

$$c_{pos}^{t+1} = c_{pos}^{t} - \eta [\sigma(c_{pos}^{t} \cdot w^t) - 1] \cdot w^t$$

$$c_{neg}^{t+1} = c_{neg}^{t} - \eta [\sigma(c_{neg}^{t} \cdot w^t)] w^t$$

$$w^{t+1} = w^t - \eta \{ [\sigma(c_{pos}^{t} \cdot w^t) - 1] c_{pos}^{t} + \sum_{i=1}^{k} [\sigma(c_{negi}^{k} \cdot w^k)] c_{negi}^{t} \}$$

最后W和C都收敛，得到每个目标词的权重，作为该词的嵌入向量。

总结：

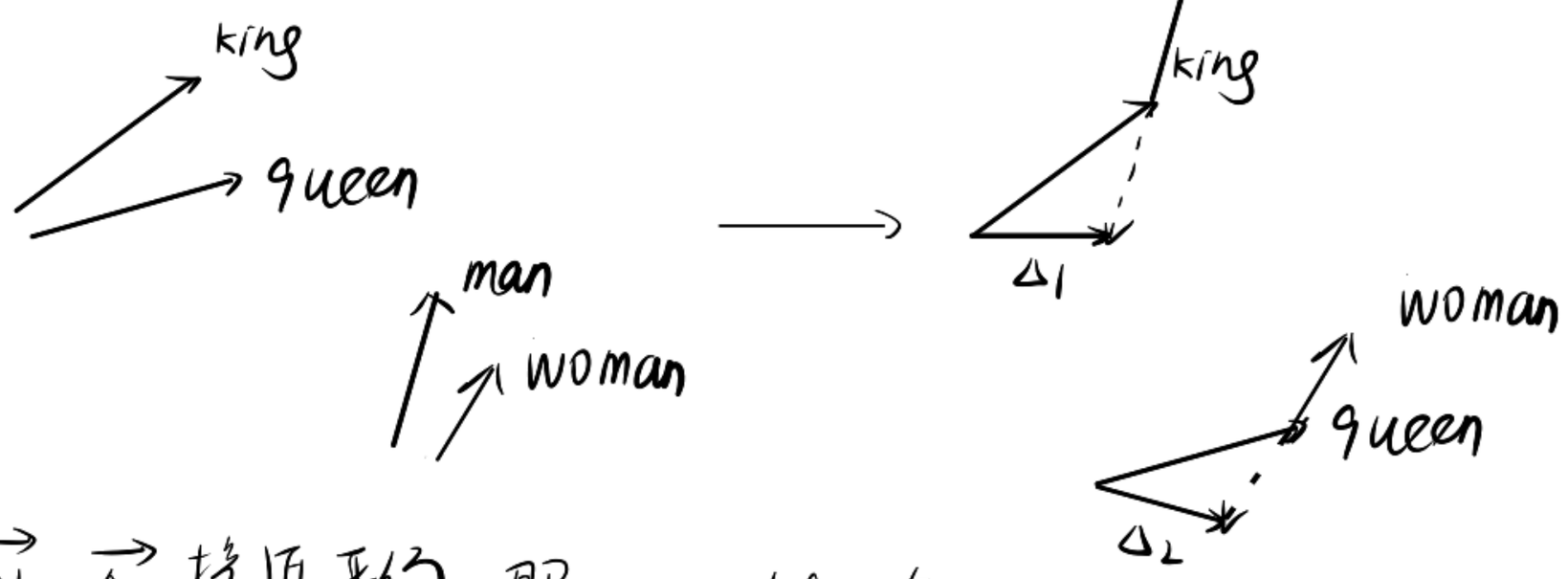
使用V个随机的d-维向量初始化嵌入向量
基于嵌入向量相似度训练一个分类器
从训练语料中选择上下文相关词对为正向样本
选择训练语料中未出现的上下文相关词对为负向样本
训练分类器同时调节所有词的嵌入向量直到损失函数达到最小
分类器的权重向量为最终各个单词的嵌入向量

Word2vec的推理能力

由于Word2vec词向量是密集的、连续的，它可以做一些运算，而最有意思的是它总结规律、作出推理的能力

如：king - man + woman ≈ queen.

数学语言：



$\vec{\Delta_1}$、$\vec{\Delta_2}$接近平行，即 man - king 与 woman - queen 偏移量几乎相同

这种能力使Word2vec能学习复数、时态、比较级甚至风格、语气等规律。

而TF-IDF不具备这种能力，因为它的向量维度太高且稀疏，不成结构，无法运算