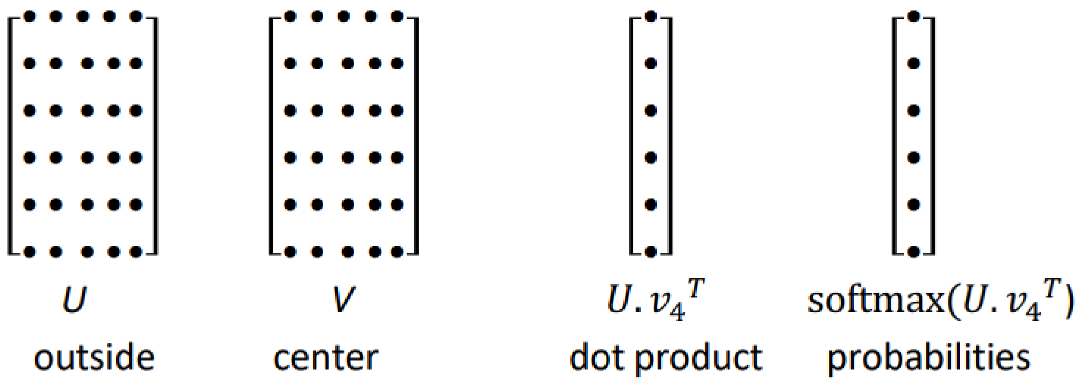
Lecture2 Word Vectors and Word Senses

Word2vec parameters and computations

上一节最后我们提到我们要对每一个单词进行embedding，我们需要根据损失函数不断对其进行求梯度运算，优化才可以得到最终的词向量。因此对于整个文本我们对于每一个单词，可以利用矩阵运算对其进行加速。

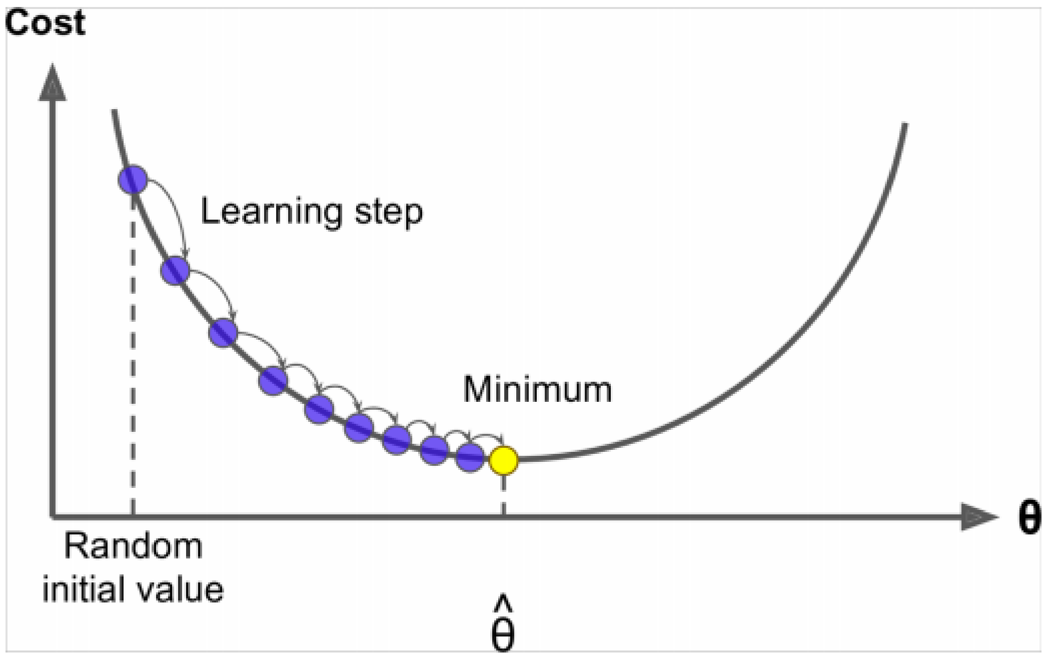


其中每一行代表一个词向量，V是center也就是当前的词，U就是其余的词。因此我们的目的也就是：

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

Optimization: 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 the direction of negative gradient. Repeat.



根据梯度下降算法，我们可以对参数进行更新，更新的equation有如下两种，分别是对所有参数更新，还有是对其中一个参数进行更新

- Update equation (in matrix notation):

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

$\boxed{\alpha = \textit{step size} \text{ or } \textit{learning rate}}$

- Update equation (for a single parameter):

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

Stochastic Gradient Descent

这里其实有个地方一直没学明白，就是随机梯度下降说的是对单个样本进行优化，在看过许多网上资料后，都说随机梯度下降前期快，后期效果并没有梯度下降好。既然都是优化，为何后期效果会不好?

- Problem: $J(\theta)$ is a function of all windows in the corpus
    - So it is very expensive to compute
- We would wait a very long time before making a single updata!
- Solution: SGD
    - Repeatedly sample windows, and update after each one

Stochastic gradients with word vectors

由于在训练的时候采用mini-batch，但是整体的词库有几百万个，因此我们利用SGD进行更新时，参数的权重矩阵是非常稀疏的。

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

因此我们想只更新出现的词。

Word2vec: More details

Two model variants:

- Skip-grams (SG)
  Predict context ("outside") words (position independent) given center word
- Continuous Bag of Words (CBOW)
  Predict center word from (bag of) context words

Additional effiiency in training

Negative sampling - Focus on naive softmax

The skip-gram model with negative sampling

- Main idea: train binary logistic regressions for a true pair (center word and word in its context window) versus several noise pairs (**the center word** paired with **a random word**)

负采样的新目标函数：

$$J_{neg-sample}(\boldsymbol{o}, \boldsymbol{v}_c, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c))$$

- We take **k** negative samples (using word probabilities)
- Maximize probability that real outside word appears, minimize prob. that random words appear around center word

如何选择negative words

- Use unigram distribution来选择negative words

要注意的一点是，一个单词被选座negative sample的概率跟它出现的频次有关，出现的频次越高越容易被选作negative words

公式如下：

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n}(f(w_j)^{3/4})}$$

每个单词被赋予一个权重，即 $f(w_{i})$ 它代表单词出现的频次，公式中开3/4的完全是基于经验。