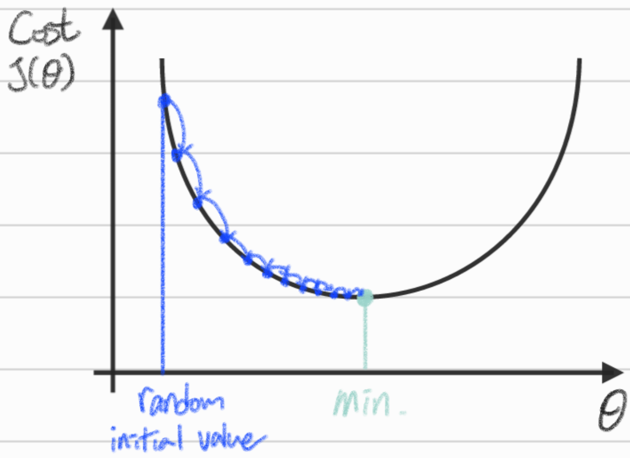
Neural Classifiers

"Bag of words" model : the model makes the same predictions
at each position
고차원 벡터공간에서 유사한 word 끼리 분포

Optimization : Gradient Descent 경사하강법
- Good word vectors 학습위해 $J(\theta)$ minimize
- GD는 θ 바꾸어 $J(\theta)$ minimize 하는 algorithm



현재 θ에서 $J(\theta)$의 gradient 구하고
take small step in the direction of negative gradient

Update equation : $\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$
learning rate, stepsize

(for a single parameter) : $\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$

문제는 $J(\theta)$가 모든 windows in corpus의 function 이라는 점
즉, $\nabla_\theta J(\theta)$ computation은 very expensive

∴ SGD (Stochastic Gradient Descent) 사용

word 하나에 two vectors → Easier optimization, Average both at the end

Two model variants:

1. Skip-grams (SG)
   - predict context words given center word
2. Continuous Bag of Words (CBOW)
   - predict center word from context words

Co-occurrence matrix gives a representation of words as co-occurrence vectors

: huge sparse matrix

Singular Value Decomposition (SVD) ⇒ 차원축소

- factorizes X into $U\Sigma V^T$, U & V are orthogonal

Encoding meaning components in vector differences

- Ratios of co-occurrence probabilities can encode meaning components

Log-bilinear model : $w_i \cdot w_j = \log P(i|j)$

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

GloVe - fast training, scalable to huge corpora, good performance even with small corpus & vectors

Word vector evaluation

Intrinsic - specific subtask에 평가, fast to compute

Extrinsic - real task에 평가, take a long time to compute

Analogy evaluation   Correlation evaluation

Word senses and word sense ambiguity

Most words have lots of meaning
- Especially common words
- Especially words that have existed for a long time

Linear algebraic structure of word senses, with applications to Polysemy
- different senses of a word reside in a linear superposition in standard word embeddings like word2vec

weighted sum
가중합

$$V_{pike} = a_1 pike_1 + a_2 pike_2 + a_3 pike_3$$

$$a_1 = \frac{f_1}{f_1+f_2+f_3}$$

의외로 self-disambiguate 해서 좋은 결과