Week 4

태그	
열	

Lecture 4 – Backpropagation

Deriving gradient

- 역전파 위해서 미분을 통해 gradient 구해야 함
- gradient를 윈도우 안의 각 word vector에 나눠서 update한다
- Word vector를 이용해 retraining 할 때 빠질 수 있는 함정
  - 영화 감상문을 분류하기 위해 회귀 분류 모델을 학습시킨다고 가정
  - training data에는 “TV”와 “Telly”가 있다
  - testing data에는 “television”이 있다
  - pre-trained word vector에서는 세 개의 단어가 모두 가까운 거리에 있다
  - 이 상태에서 word vector를 gradient를 이용해 update하면, training data에 있던 단어들은 벡터 공간에서 위치가 달라지지만 그렇지 않은 testing data에 있던 단어는 제자리에 있게 된다

Backpropagation

- weight를 곱하고 bias를 더한 후 softmax 등의 함수를 거쳐 나온 s의 순서를 거꾸로 가면서 구한 각 parameter에 대한 gradient를 넘겨준다.
- node는 upstream gradient를 전달받는다
- 목표는 정확한 “downstream gradient”를 다음 node에 넘겨주는 것
- 각 node는 local gradient를 갖고 있으므로 이 local gradient와 upstream gradient 에 chain rule을 적용해서 계산하면 downstream gradient를 구할 수 있다
  - [downstream gradient] = [upstream gradient] * [local gradient]
- 여러 개의 input이 있는 node는 upstream gradient를 전달 받아서 각 input parameter에 대한 local gradient를 곱한 값을 계산한다.
- + 노드는 upstream gradient는 각 input에 분배한다

- max 노드는 upstream gradient를 routing 한다
- * 노드는 upstream gradient를 switch 한다

Back-prop in general computation graph

1. Fprop: visit nodes in topological sort order
    - compute value of node given predecessors
2. Bprop:
    - initialize output gradient = 1
    - visit nodes in reverse order

Done correctly, big O() complexity of fprop and bprop is **the same**

Regularization

- Needed to prevent overfitting

Non-linearities: The starting points

- logistic(”sigmoid”): 확률 반환
- tanh
- hard tanh

Non-linearities: The new world order

- ReLU: 음수면 0, 양수면 값 그대로 반환
    - feed-forward deep network 만들 때 가장 먼저 사용해볼 것은 ReLU이다 - 빠르고 성능이 좋다

Parameter initialization

- 보통 처음엔 weight를 정해줘야 한다

Optimizers

- Plain SGD도 보통은 잘 작동함
- 더 다양한 종류가 있음 (ex: adam, RMSprop ...)

Learning rates

- 상수값을 이용해도 된다

- 보통은 학습시킴에 따라 learning rate이 줄어들도록 하면 더 좋은 결과가 나온다