CH 8

강남웅

Learning vs Pure Optimization

- Pure Optimization
  - Finding optimal solution of given function and parameters
- Learning
  - Since we cannot find true underlying distribution of data,

    we try to find or "learn" function that fits to criterion that we make

Empirical Risk Minimization

- True Underlying distribution를 알 수 없기 때문에,

  Empirical distribution을 사용해서 risk를 minimzing하는 방법

- 문제점
  - overfitting 문제에 노출될 수 있다
  - loss function들 중에 직접적으로 derivative를 구할 수 없는 경우가 존재한다

Surrogate Loss Function

- 주어진 Loss function을 optimize하려고 하는데,

  매우 어렵거나 불가능한 경우가 있다

- 이런 경우에는 구할 수 있는 함수들로 원래의 loss function을

  대신해 구하게 되는 함수 : Surrogate Loss Function

Surrogate Loss Function

- 예시)
  - Binary Classification에서 가장 명료하고 직관적인 loss function은 0-1loss이다. (맞으면 1 틀리면 0)
  - 이런 경우 loss function이 미분이 불가능하기 때문에 gradient decent 같은 방법을 사용할 수 없다.
  - 이 때 사용할 수 있는 Surrogate loss function : logistic function

Minibatch Algorithm

- Cost function과 Gradient를 계산할 때,
  모든 dataset을 계산하는 것은 computational cost가 굉장히 높다

- 모든 dataset을 사용하지 않고, 일부만 random sampling한 후 계산하는 방식이 minibatch algorithm

Minibatch Algorithm

- Consideration
  - Standard Error 식에서 알 수 있듯이,

$$\mathrm{SE}(\hat{\mu}_m) = \sqrt{\mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right]} = \frac{\sigma}{\sqrt{m}},$$

    denominator에 존재하는 $\sqrt{m}$ 의 존재
    - dataset이 100배 늘어나면, standard error는 10배만 줄어든다.
    - Computational cost와 standard error의 Trade Off

  - Dataset의 redundancy
    - Sampling하는 과정에서 dataset에 비슷한 데이터들끼리 선별되지 않게 해야한다
    - Dataset에 high correlation을 갖고 있는 data들끼리 모여있다면 shuffle을 반드시해야 한다

Minibatch Algorithm

- Consideration of batch size
  - Batch size가 커질 수록 정확도는 올라가지만, 그 효과는 $1/\sqrt{n}$ 이 된다 (less than linear)
  - Multicore 환경에서 한 번에 training 가능한 size가 존재한다
    - 그것보다 작은 dataset은 효율이 낮아지는 것을 고려해야 한다
  - Parallel하게 진행이 된다면, 한 번의 iteration에 사용되는 (memory size) = (batch size)

Challenges in NN-Optimization

- Ill-Conditioning
  - A small change in the input leading to a large change in the output
  - 가장 대표적인 경우 : ill-conditioning Hessian matrix
    - step size가 ϵ일 때,

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

    - 이 때, $\frac{1}{2}\epsilon^2 g^T H g \gg \epsilon g^T g$ 인 경우에 gradient norm($g^T g$)이 learning과정에서 제대로 작아지지 않는다 => optimal한 solution을 찾기 힘들어진다

Challenges in NN-Optimization

- Local minima
  - Model identifiability
    - 충분히 큰 training set이, model을 하나로 구체화할 수 있다면, identifiable
  - Local minima가 생기는 이유 : Non-identifiable
    - Weight space symmetry
      - latent variable들끼리 exchange해도 같은 model을 도출해낸다면, 하나의 model로 구체화 할 수 없다
    - Absence of weight-dependence term
      - cost function에 output에만 의존하는 term으로만 구성이 되어있는 경우

Challenges in NN-Optimization

- Local Minima caused by non-identifiability
  - non-identifiability에 의해서 생긴 local minima들은 같은 cost function 값을 갖게 된다.
  - 즉, global minima보다 cost function 값이 매우 크지 않으면 문제가 되지 않는다
  - 하지만, 충분히 큰 Neural Network에서 대부분의 local minima는 low cost function value를 갖기 때문에 true global minimum을 찾지 않아도 괜찮다

Challenges in NN-Optimization

- Saddle Point
  - Gradient가 0이 되는 지점
  - local minima/maxima와 다른 점 : Hessian matrix의 eigenvalue가 양수/음수를 모두 갖는다
  - local minima보다 발생할 확률이 매우 낮다 (1 : a^n)
  - gradient descent empirically seems to be able to escape saddle points in many cases

Challenges in NN-Optimization

- Cliffs and Exploding Gradients
  - Extremely steep region => Cliff
  - Multiplication of several large weights 에 의해 발생된다
  - 특히 RNN에서 자주 발생한다
  - (10.11.1) Gradient Clipping으로 피할 수 있다

Challenges in NN-Optimization

- Long-Term Dependencies
  - Computational graph가 deep한 경우 발생
  - 지속적으로 Weight가 곱해져서 발생
    - $W^t = Vdiag(\lambda)^t V^{-1}$ 에서 λ_i가 1보다 클 경우 gradient가 explode되거나 λ_i가 1보다 작을 경우 gradient가 vanish될 수 있다

      => vanishing and exploding gradient problem
  - RNN와 같이 같은 W가 지속적으로 곱해지는 경우가 아닌, feedforward network처럼 서로 다른 W가 곱해지는 경우 발생할 확률이 낮다