Chapter 3. Hyper Parameters Review

-Y, Bengio et al., Practical recommendations for gradient-based training of deep architectures., 2012.-

2024 DL Session - Team2 4주차 과제물

3.1 Neural Network Hyperparameter

신경망 및 딥러닝 모델의 성능 최적화를 위한 주요 하이퍼파라미터와 계층별 최적화 전략을 다루고 있다. 동시에 신경망 학습에서 하이퍼파라미터의 중요성을 강조하며, 각 하이퍼파라미터가 모델의 학습에 미치는 영향에 대해 소개하고 있다.

근사 최적화의 하이퍼파라미터

1) 초기 학습률

- 학습률은 모델의 학습 속도를 조절하는 가장 중요한 하이퍼파라미터이다.

- 학습률이 너무 높으면 학습이 불안정해지고, 너무 낮으면 학습 속도가 느려진다.

- 일반적으로 0.01의 값을 사용하지만, 모델과 데이터셋에 따라 최적의 학습률은 다를 수 있다. 따라서, 초기 학습률을 여러 값으로 설정해보고 검증 데이터셋에서의 성능을 비교하는 것이 좋다.

2) 학습률 스케줄

- 학습률을 점차 감소시키거나 적응시키는 전략이다.

- 예를 들어, 학습률을 초기에는 일정하게 유지하다가 점차 감소시킬 수 있다.

- 이는 학습 후반부에 안정성을 높이고, 최적의 손실 함수 값을 찾는 데 도움을 준다.

- 시간 상수 τ₩tauτ와 같은 하이퍼파라미터를 조정하여 최적의 스케줄을 찾을 수 있다.

3) 모멘텀

- 모멘텀은 이전 그레디언트의 기여를 현재 그레디언트에 추가하여 학습의 안정성과 수렴 속도를 높이는 방법이다.

- 모멘텀을 사용하면 그레디언트가 급격히 변하는 상황에서도 안정적으로 학습할 수 있다.

- 기본 모멘텀 값은 0.9 또는 0.99로 설정하는 경우가 많으며, 이를 통해 진동을 줄이고 더 빠르게 수렴할 수 있다.

4) 미니배치 크기

- 미니배치 크기는 한 번에 학습에 사용되는 데이터 샘플의 수를 의미한다.

- 일반적으로 1에서 수백까지 설정할 수 있으며, 32 또는 64와 같은 값이 자주 사용된다.

- 작은 미니배치를 사용하면 더 자주 가중치 업데이트가 이루어져 더 빠른 수렴을 유도할 수 있지만, 너무 작으면 그레디언트의 노이즈가 커질 수 있다.

- 큰 값으로 설정하면 그레디언트의 노이즈를 줄일 수 있지만, 계산 비용이 증가하고 메모리 사용량이 많아진다.

- 따라서, 적절한 미니배치 크기를 선택하는 것이 중요하다.

5) 훈련 반복 횟수

- 조기 중지를 통해 최적의 훈련 반복 횟수를 결정할 수 있다.

- 훈련 중 검증 데이터셋의 성능을 모니터링하여 성능이 더 이상 개선되지 않으면 학습을 중지한다.

- 이를 통해 과적합을 방지하고, 최적의 모델을 얻을 수 있다.

- 훈련 반복 횟수는 데이터셋의 크기와 모델의 복잡성에 따라 달라지며, 일반적으로 수천에서 수만 번의 반복이 필요할 수 있다.

계층별 최적화의 하이퍼파라미터

비록 드물게 사용되지만, 다계층 네트워크의 서로 다른 계층에서 최적화 하이퍼파라미터(ex. 학습률) 값을 다르게 설정할 수 있다. 이는 계층별 비지도 사전 학습의 맥락에서 특히 적합하며(이 경우 각 계층이 별도로 학습됨), 하위 계층은 고정된 상태로 유지된다. 계층당 유닛 수가 계층마다 크게 다를 때 특히 유용하다.

일부 연구자들은 모델에서 발견되는 다른 유형의 매개변수(예: 표준 다계층 네트워크의 편향 및 가중치)에 대해 서로 다른 학습률을 사용하는 것을 권장하기도 한다. 그러나 이러한 문제는 정밀도나 분산과 같은 매개변수가 포함될 때 더 중요해진다.

3.2 Hyper-Parameters of the Model and Training Criterion

Number of Hidden Units n_h

Multilayer Neural Network 내 각 레이어 마다 모델의 capacity를 조절하는 Hidden Unit의 Size를 할당한다.

* Capacity : 머신러닝에서 모델에 있는 학습 파라미터의 수

early stopping과 같은 regularization는 큰 n값을 갖도록 유도된다. (optimal size보다 커지면 계산양이 늘어남) 각 hidden layer마다 다른 size를 가지도록 할 수 있다. 이는 일반적으로 점점 줄어들거나 증가하는 것보다는 모든 층이 같은 size를 가질 경우 더 잘 작동하게 된다. (data-dependent)

일반적으로 overcomplete인 첫 hidden layer가 그렇지 않은 레이어보다 더 잘 작동한다.

*overcomplete : input vector보다 더 큰 size를 가지는 layer

Supervised Neural Network에서 Unsupervised Pre-training을 사용했을 때 더 큰 optimal size를 보인다. pre-training 후에는 많은 hidden unit이 훈련에 사용하는 특정한 task과 무관계한 정보를 담게 되기 때문에 쓸모 있는 정보만 거르기 위해서는 더 큰 hidden layer의 size가 요구된다.

Weight Decay λ

overfitting을 줄이기 위해 training criterion에 regularization term을 도입하여 모델의 capacity를 줄인다.

< regularization term >

1. L1 regularization $\lambda \sum_i \theta_i^2$ *Ridge
2. L2 regularization $\lambda \sum_i |\theta_i|$ *Lasso - 두가지를 동시에 추가 *ElasticNet

- unbiased한 training criterion은 data와 theta의 negative joint likelihood로 설명됨 (= loss function)

Problem : SGD를 계산할 때 total loss와 regularizer을 구해서 최적화를 해야 하나, 우리는 한번에 하나의 mini-batch나 example만을 고려한다는 문제점이 생김

⇒ 따라서 **mini batch update**를 할 때 regularization의 penalty term의 기울기는 λ뿐만 아니라 total size와 배치사이즈를 고려한 **B/T**를 곱해줘야 한다. Training set의 size가 B가 아닐 경우, 즉 마지막 batch에서 size가 B와 어긋날 경우 **B'/B**를 추가하여 **scaling**해준다. pure online setting(training set setting과 data iteration이 없는 경우)에서는 샘플 t에 대해 B/t를 사용하게 된다.

L2 regularization

– early stopping(더 효과적)과 같은 역할을 수행

L1 regularization

– feature selection 효과 : 쓸모 없는 parameter을 0으로 이끌어(⇒ sparsity of parameter values) input filter(*input을 받는 첫번째 레이어의 가중치)을 깨끗하게 만들어주고 해석하기 쉽게 한다.

*라플라스 분포를 따름

$$e^{-\frac{|\theta|}{s}}/s = \frac{1}{\lambda}$$

SGD 적용 시 정확히 0의 값을 가지진 않지만 0과 가까운 값을 가지게 된다.

L1, L2 regularization

– 각각 다른 hyperparameter가 고려되어야 하며 각 레이어마다도 다른 값들이 사용될 수 있다. 특히 input과 output layer의 가중치는 자주 다르게 다뤄진다.

⇒ 모델의 capacity를 제한하기 위해 output 가중치를 정규화하는 것만으로 충분하기 때문

⇒ 또한 input layer의 값이 매우 ***sparse**할 수 있기 때문

*예를 들어 input feature의 몇몇이 0에 가까울 때, 이런 비 활동적인 input feature은 자주 관찰되는 input feature보다 더 많은 정규화를 가해야 한다. 혹은 잘 관찰되지 않는 사건 등을 다루는 등 target variable이 sparse한 상황을 고려해야 한다.

⇒ 두 경우 모두 parameter에 의해 제안되는 효율적인 업데이트 횟수는 실제 (효율적인) 업데이트 횟수보다 적게 된다. 따라서 parameter에 의해 제안되는 업데이트 횟수보다 더 큰 값을 선택하도록 정규화 계수를 조정해야 한다.

+ hidden unit에서 bias를 제외한 weight에만 penalty를 주는 방법도 사용

⇒ 강한 정규화에도 predictor가 최적의 상수로 수렴함을 보장할 수 있기 때문이다.

Sparsity of activation α

training criterion에 penalty를 추가하여 hidden unit이 0 혹은 0에 가깝도록 조정한다.

*training distribution을 포함하기 때문에 parameter보다 선행하는 L1과는 수학적으로 다름

*example을 줄이는 regularization의 일종

Question : sparse하게 만드는 이유는?

⇒ 표현의 근본적인 요소들을 분해해주기 때문이다. (일반적으로 큰 sparsity = 많은 수의 hidden unit)

1. **How?**

⇒ L1 norm 혹은 hidden unit의 활성화 함수(student t log prior 등)에 penalty term을 추가

*output이 0에 가깝도록 조정하는 sigmoid 함수에 주로 이용

⇒ hidden unit의 bias가 음의 값을 갖도록 penalty를 부여

layer의 input이 일반적으로 0이 아닌 평균을 갖기에 bias에 penalty를 주는 것은 weight를 곱한 값이 bias처럼 작용할 수 있기 때문에 weight가 bias을 보상할 수 있는(늘릴 수 있는) 위험을 가진다.

⇒ ***sparse auto-encoder** 등의 한정된 모델에서 사용

*regularization을 통해 훈련마다 뉴런의 개수를 제한함으로써 각 뉴런이 더 유용한 특성을 학습할 수 있도록 하는 기법; dropout과 비슷한 효과

2. **How?**

⇒ L1 regularization과 비슷한 **student t regularization**도 존재

⇒ **average output**을 MSE를 통해 규제하거나 p의 확률을 가지는 binomial 분포에 대해 *쿨백-라이블러 발산을 사용하여 정해진 타겟 p에 다가가도록 함

*Kullback-Liebler divergence - 두 확률분포의 차이를 계산하는 데 사용. 어떤 이상적인 분포에 대해 그 분포를 근사하는 다른 분포를 사용해 샘플링을 할 때 발생할 수 있는 정보 엔트로피 차이를 계산

⇒ **activation function**의 선택

특히 rectifying의 성질(*한 방향으로만 값을 가짐)을 가진 ReLU함수, hard tanh(기울기가 0 혹은 1)가 잘 작동한다. activation이 직접적으로 최적화되며 최적화의 결과 0이 나오게 된다.

⇒ 평범한 SGD는 이러한 0을 찾을 수 없기 때문에 objective 함수를 differentiable한 함수와 non-differentiable한 함수로 분리해서 최적해를 찾는 방법인 proximal gradient가 더 적절하다.

Neuron non-linearity

- Sigmoid : Unsupervised pre-training 없이 deep supervised network의 첫 hidden layer에서 사용될 경우 최적화에 문제가 생긴다. (예외적으로 *auto-encoder에서는 잘 작동함)

* auto-encoder : 입력이 들어왔을 때 해당 입력 데이터를 최대한 압축시킨 후, 데이터의 특징을 추출하여 다시 본래의 입력 형태로 복원시키는 신경망

- rectifier(ReLU 계열) : output layer의 unit에서는 기울기의 역전파 계산을 통한 업데이트가 불가한 경우가 존재하기 때문에(saturated한 분포를 가질 때 등) 잘 사용하지 않는다. Hidden layer의 경우 다른 unit이 saturated 해도 기울기의 계산이 다른 hidden unit의 부분집합으로 해결된다.

: output layer의 unit에서는 negative log-likelihood를 고려하고 적절한 조건부확률모델을 고름으로써 output의 nonlinearity와 loss를 보존할 수 있음

squared error, linear outputs - Gaussian output model / cross-entropy, sigmoid - binomial output model / softmax [- log output[taregt class]] - multinomial output variables (예시)

Weights initialization scaling coefficient

bias는 일반적으로 0으로 초기화될 수 있으나 같은 레이어 안 hidden unit 간 대칭성을 깨기 위해 weight의 초기화는 적절히 고려되어야만 한다.

*output unit은 각기 다른 gradient signal을 받기 때문에, 대칭을 깨트리는 것은 output weight을 고려하지 않아 weight을 0으로 설정하는 것이 가능

- **initialization range에 대해 scaling coefficient를 hyperparameter로 사용**하는 것을 제안(Bergstra and Bengio, 2012)

⇒ fan-in(하나의 뉴런으로 들어오는 입력의 수), fan-out(하나의 뉴런에서 나오는 출력의 수) 중 fan-in의 square-root의 역수를 사용하는 방법 혹은 fan-in과 fan-out을 동시에 사용하는 scaling이 제안됨.

⇒ 이러한 방법을 통해 초기화와 관련된 hyperparameter를 따로 조정하지 않아도 분포가 적절히 유지됨.

Unsupervised pre-training의 사용 여부와 그 알고리즘 종류는 파라미터의 초기화에 중요하다.

(*Unsupervised pre-training은 대부분 좋은 결과를 이끌어내지만 긴 학습시간과 추가적인 파라미터의 조정이 필요함)

Random Seeds

랜덤 초기화, 데이터 샘플링, *RBM과 같은 확률적 모델에서 사용되는 hidden unit의 샘플링 등에서는 랜덤한 값을 취하기 때문에 어떤 값을 취하냐에 따라 결과가 달라진다.

*Restricted Boltzmann Machine : 관측 가능한 변수들로 이루어진 층 하나와 잠재변수들로 이루어진 층 하나를 담은 무향 확률 그래프 모형

- 그러나 Erhan et al. (2010b)가 만들어낸 random seed에 따른 test error의 히스토그램을 보면 random seed의 선택은 hyperparameter를 찾는 과정에서 무시될 수 있을 만큼 작은 변화를 보임.
- 컴퓨팅 파워가 좋다면 각기 다른 seed의 가장 좋은 파라미터 값의 집합으로 성능을 향상시킬 수 있음

(*혹은 Bagging과 베이지안 정리를 사용하여 여러 모델, 알고리즘의 평균을 내는 방법도 고려)

Preprocessing

rawdata를 적절한 input으로 변환한다.

- element-wise standardization : 평균을 빼고 표준편차로 나눠 계산

- PCA : Principal Component의 개수와 얼마나 분산을 설명할 건지를 정해야 하기 때문에 parameter 수가 늘어남, 차원축소 효과

- 각 요소의 Uniformization : 누적확률분포를 추정하고 각 요소의 x를 누적 확률분포 값으로 변환

- 로그, 제곱근 등의 비선형성 변환 : 입력 요소 분포의 꼬리를 줄일 수 있음. 가우시안과 가까워 짐.

3.3 Manual Search and Grid Search

possible to make the optimization of hyperparameters a more reproducible and automated process, using techniques such as grid search or better, random search, or even hyper-parameter optimization, discussed below.

→ 대규모 컴퓨터 클러스터 기반의 현대 컴퓨팅 시설을 사용하면 그리드 탐색, 랜덤 탐색, 또는 하이퍼파라미터 최적화와 같은 기술을 통해 하이퍼파라미터 최적화를 더 재현 가능하고 자동화된 과정으로 만들 수 있다.

Hyper-parameters 탐색에 대한 일반 지침

We call a numerical hyper-parameter one that involves choosing a real number or an integer (where order matters), as opposed to making a discrete symbolic choice from an unordered set.

Best value on the border

- pay attention as to whether or not the best value found is near the border of the investigated interval

- 최적값이 조사된 구간의 경계 근처에 있는지 여부를 주의해야 함. 경계 근처에 있으면 그 경계를 넘어 더 나은 값을 찾을 수 있으므로 그 경계를 넘어 탐색하는 것이 좋다. 하이퍼 파라미터와 검증 오류 간의 관계가 잡음이 있을 수 있으므로 너무 적은 값을 시도하는 것은 일반적으로 충분하지 않다. 예를 들어, 수치적 하이퍼파라미터에 대해 단 3개의 값만 시도하는 것은 충분하지 않다.

Scale of values considered

- By choosing the interval large enough to start with, but based on previous experience with this hyper-parameter, we ensure that we do not get completely wrong results. Now instead of choosing the intermediate values linearly in the chosen interval, it often makes much more sense to consider a linear or uniform sampling in the log-domain (in the space of the logarithm of the hyper-parameter).

- 수치적 하이퍼파라미터 값을 탐색하는 것은 시작 간격을 선택하는 것을 포함하며, 이는 일종의 하이퍼-하이퍼파라미터이다. 시작 간격을 충분히 크게 선택하고 이전 경험을 기반으로 하면 완전히 잘못된 결과를 얻지 않도록 할 수 있다. 이제 선택한 간격에서 값을 선형적으로 선택하는 대신, 로그 도메인에서 선형 또는 균일 샘플링을 고려하는 것이 더 합리적이다. 예를 들어, 학습률이 0.01인 결과는

0.011인 결과와 유사할 가능성이 크지만, 0.001인 결과는 0.002인 결과와는 상당히 다를 수 있다. 다양한 값 간의 비율이 변화의 영향을 더 잘 예측할 수 있을 것이다. 따라서 양의 값을 갖는 수치적 하이퍼파라미터의 경우 로그 도메인에서 균일하게 또는 정규적으로 간격을 두어 값을 탐색하는 것이 일반적으로 선호된다.

Computational considerations

- Often, one has to consider computational cost, either of training or prediction. Computing resources for training and prediction are limited and generally condition the choice of intervals of considered values.

- 검증 오류뿐만 아니라 종종 훈련이나 예측의 계산 비용도 고려해야 한다. 훈련 및 예측을 위한 계산 자원이 제한되어 있으며, 이는 고려되는 값의 간격을 조건화 해야 한다. 예를 들어, 은닉 유닛의 수나 훈련 반복 횟수를 늘리면 계산양도 증가함. 검증 오류의 계산적으로 저렴한 추정치를 사용하여 일부 하이퍼파라미터를 선택하는 것도 흥미로운 아이디어이다. 예를 들어, Saxe et al. (2011)은 무작위 가중치를 사용하여 일부 하이퍼파라미터를 선택할 수 있음을 보여주었다. 이 값이 잡음이 많고 편향된 추정치임에도 불구하고 이는 충분히 상관 관계가 있다. 따라서 이는 하이퍼파라미터 선택에 충분함을 확인하였다.

Coordinate Descent and Multi-Resolution Search

manual search and with access to only a single computer → 합리적인 전략은 좌표 하강법

- 하이퍼파라미터 중 하나만 변경하고, 지금까지 발견된 최적의 하이퍼파라미터 구성에서 변경한다. 표준 좌표 하강법 대신, 가장 민감한 변수(예: 학습률)를 정기적으로 미세하게 조정 가능하다.

Automated and Semi-automated Grid Search

- Once some interval or set of values has been selected for each hyper-parameter (thus defining a search space), a simple strategy that exploits parallel computing is the grid search.

→ 각 하이퍼파라미터에 대해 일정한 간격이나 값 집합을 선택한 후(따라서 탐색 공간을 정의한 후), 병렬 컴퓨팅을 활용할 수 있는 간단한 전략은 그리드 탐색!

- 그리드 탐색의 장점은 다른 최적화 전략(예: 좌표 하강법)과 비교할 때 완전히 병렬화할 수 있다는 점.

- 대규모 컴퓨터 클러스터를 사용할 수 있는 경우 병렬화를 활용할 수 있는 모델 선택 전략을 선택

하는 것이 좋다.

- 그리드 탐색의 실질적인 단점은 병렬화된 작업 세트 중 하나라도 실패하면 그리드를 완료하기 위해 다른 작업을 시작해야 하며, 이는 전체 컴퓨팅 시간을 증가시킨다.

- 일반적으로 single grid search은 충분하지 않으며, 실무자는 각 검색 결과에 따라 고려된 값의 범위를 조정하여 a sequence of grid search 수행한다. 이는 수동으로 할 수 있지만, 이 절차는 considering the idea of multi-resolution search로 자동화 할 수 있다.

- 각 그리드 탐색은 이전에 발견된 최상의 솔루션 주변에서 더 국부적으로 수행.

- 또한, 각 그리드 탐색이 몇몇 하이퍼 파라미터에만 집중하도록 coordinate descent 아이디어를 추가할 수도 있다.

- 사람은 하이퍼파라미터 탐색을 매우 잘 수행할 수 있으며, 사람을 탐색 과정에 포함하면 학습 알고리즘의 버그나 원하지 않거나 예상치 못한 동작을 감지하는 데 도움이 된다.

- However, for the sake of reproducibility(재현성), machine learning researchers should strive to use procedures that do not involve human decisions in the middle, only at the outset .

하이퍼 파라미터의 레이어별 최적화

- 비지도 사전 학습을 사용하는 경우, 하이퍼파라미터 선택과 관련된 좌표 하강법과 저비용의 상대적인 검증 성능 평가를 결합할 수 있는 기회 있다.

- 하위 계층(입력에 가까운 계층)에 대한 하이퍼파라미터를 먼저 탐색한 후 상위 계층을 학습하는 것.

- 먼저 첫 번째 계층을 비지도 학습으로 여러 하이퍼파라미터 값으로 학습하고, 최종 네트워크가 내부 표현으로 이 단일 계층만 갖는 경우 얻을 수 있는 상대적인 검증 오류를 추정한다.

- 이 greedy evaluation에 따라 좋은 하이퍼파라미터 값 집합을 찾음 (또는 찾은 값 중 가장 좋은 것만 사용) → 이러한 값들을 사용하여 두 번째 계층을 같은 방식으로 학습하고 하이퍼파라미터 최적화를 진행한다.

- 완전히 탐욕적인 접근법은 현재까지 최상의 구성만 유지하는 것이지만, 전체적으로 K개의 최적 구성을 유지하면 첫 번째 계층 이후의 계층에 대한 하이퍼파라미터 선택의 계산 비용이 K배 증가

 - 이는 첫 번째 계층과 두 번째 계층의 모든 하이퍼 파라미터에서 최상의 K구성을 탐색하여 세 번째 계층 하이퍼파라미터를 탐색할 시작점으로 사용하기 때문. greedy layer-wise pretraining은 상위 계층을 사전 학습할 때 하위 계층을 수정하지 않으므로 계산 효율성이 매우 높다. 이 절차를 통해 비지도 사전 학습 단계와 관련된 하이퍼파라미터를 설정할 수 있으며, 그 후 지도 세부 조정 단계에서 선택할 하이퍼 파라미터가 남는다.

3.4 Random Sampling of Hyper Parameters

Grid Search 문제점

Hyper Parameter의 수가 늘어남에 따라 고려해야 할 Hyper Parameter의 수가 기하급수적으로 증가한다. 따라서 시간과 자원이 무한하게 많이 필요하게 된다. 이를 보완하는 다른 방식으로...

Random sampling

고르게 분포된 확률분포에서 독립적으로 샘플링을 하여 Hyper Parameter를 탐색한다. 이는 특정 값들을 미리 알지 못하는 상황에서 매우 유용하게 작용한다.

- Hyper Parameter의 수가 2~3개를 넘어가면 Grid보다 좋은 효율성을 가지고 수렴한다.
- 이는 더 다양하고 넓은 범위의 값을 탐색할 수 있기 때문.

Q. 만약 Hyper parameter에 대한 사전 지식이 존재한다면?

- Binary Hyper parameter에 대하여 다항분포를 기반으로 샘플링을 하는 것이 효과적이다
- 이와 같이, 사전에 파라메터에 대한 정보가 존재하면 이에 맞는 분포에서 샘플링을 진행
- 더욱 정교한(sophisticated)분포를 만들 것임.
- 만약 사전정보가 없으면, 균등분포를 사용

Grid보다 상대적으로 우위인 점 – 병렬처리

- Grid는 병렬처리에서 도중 오류가 발생시 다시 처음부터 탐색해야 하지만, Random은 그럴 필요가 없다.
- Grid는 적절히 좋은 결과를 만들었을 때, 이어서 탐색하기 어렵지만, Random은 다시 확률적으로 파라메터를 뽑아 진행하면 되기 때문에 독립적인 실험을 동시에 여러 개 진행할 수 있다.
- 결론적으로 최적의 값을 더 빠르게 찾을 수 있다.