Deep Learning II

김연지
kimyeonji3@gmail.com

Perceptron의 손실함수

- 퍼셉트론의 목적
  - 주어진 학습 데이터의 입력 정보(x)와 출력 정보(t)의 관계를 잘 찾도록 가중치 w를 조절하기

- 관계가 잘 찾아졌는지는 어떻게 알 수 있을까?
  - 퍼셉트론의 결과물 y가 실제 정답 t에 얼마나 가까운지를 손실 함수(loss function)를 통해 측정 (손실 함수는 객체별로 산출)

- 손실함수 : 판정 결과가 정답인지, 오답인지 판정하고 그 오차를 평가하는 함수 의미
  - 회귀: 주로 squared loss 사용
  - 분류: 주로 cross-entropy loss 사용 (퍼셉트론은 이중 분류만 가능)
  - 전체 데이터 셋에 대해서 현재 퍼셉트론이 얼마나 잘못하고 있는지는 비용 함수(cost function)을 사용하며, 이는 모든 객체의 손실함수에 대한 평균값을 주로 사용함

Perceptron의 손실함수

일반적으로 평균 제곱 오차(MSE)와 교차 엔트로피 오차(CEE)를 사용

평균제곱오차(Mean Squared Error, MSE)

$$E = \frac{1}{2}\sum_{k}(y_k - t_k)^2$$

y : 신경망의 출력(각각의 뉴런의 output)
t : 정답 레이블
k : 데이터의 차원 수

교차 엔트로피 오차(Cross Entropy Error, CEE)

$$E = -\sum_{k} t_k \log y_k$$

log 밑이 e인 자연로그
y : 신경망의 출력
t : 정답 레이블

https://han-py.tistory.com/209

Perceptron의 손실함수

미분 값이 0 이라면 가중치 매개변수를 어느 쪽으로 움직여도 손실 함수의 값은 달라지지 않는다

- 신경망 학습에서는 매개변수의 미분한 값을 단서로 매개변수의 값을 서서히 갱신하는 과정을 반복하며 최적의 매개변수 (가중치 w와 편향 b) 값을 탐색할 때 손실 함수의 값을 가능한 작게 하는 매개변수 값을 찾음.

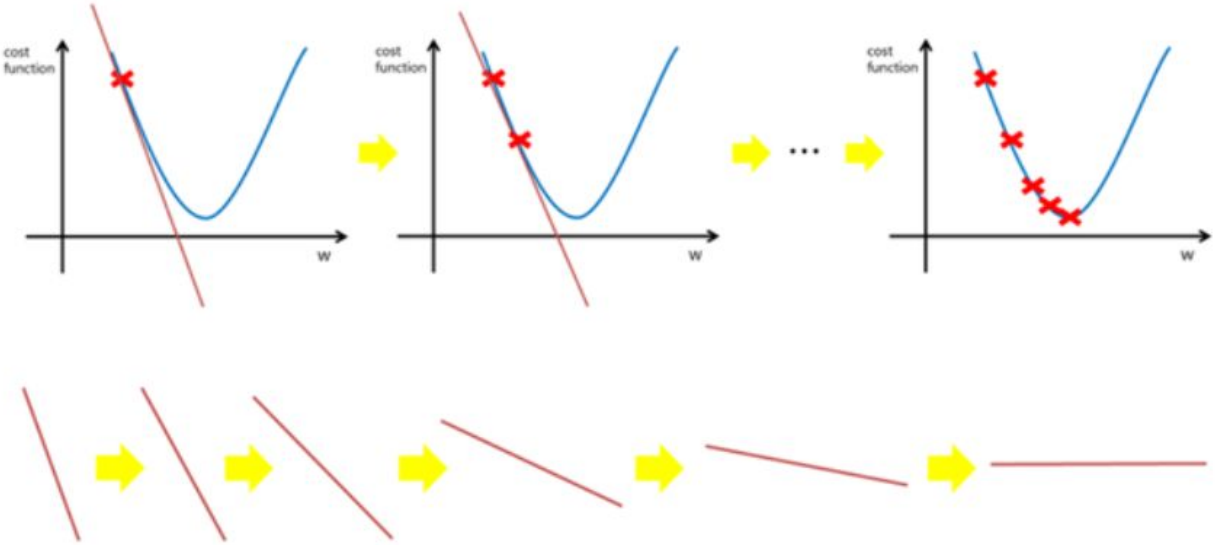
- 손실함수는 매개변수의 변화에 연속적으로 변화하지만, 정확도는 매개변수의 변화에 둔감하고, 불연속적으로 변화하므로 미분이 불가능하다. 미분이 안되면 최적화를 할 수 없어서 정확도가 아닌 손실 함수를 지표를 삼아 학습.

- 손실 함수는 성능 척도(Performance Measure)와는 다른 개념임
- 성능 지표는 알고리즘의 학습이 끝났을 때 모델의 성능을 평가하기 위한 지표므로 알고리즘 학습 중에는 사용하지 않음.
- 반면, 손실 함수는 알고리즘 학습 중에 학습이 얼마나 잘 되고 있는지 평가하기 위한 지표.

경사하강법(Grandient Descent)

최적화 : 오류 값을 계산하고 해당 값을 최소화하기 위해 가중치를 변경하는 것(손실(cost)을 줄이는 알고리즘)

함수의 기울기(경사)를 구하여 기울기가 낮은 쪽으로 계속 이동시켜 극값(최적값)에 이를 때까지 반복하며 미분 값(기울기)이 최소가 되는 점을 찾아 알맞은 weight(가중치 매개변수)를 찾아냄.

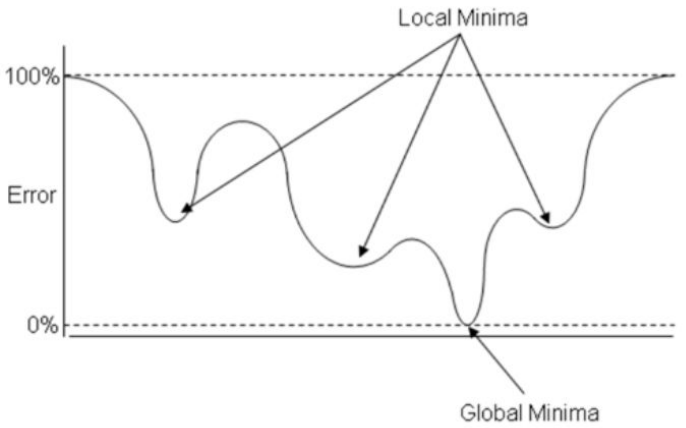


경사하강법(Grandient Descent)

- 미분값(기울기)가 최소값 위치를 찾기 위해 비용함수(Cost Function)의 경사 반대 방향으로 정의한 step size를 가지고 조금씩 움직여 가면서 최적의 파라미터를 찾으려는 방법.
- 경사는 파라미터에 대해 편미분한 벡터를 의미하며 이 파라미터를 반복적으로 조금씩 움직이는 것이 관건.

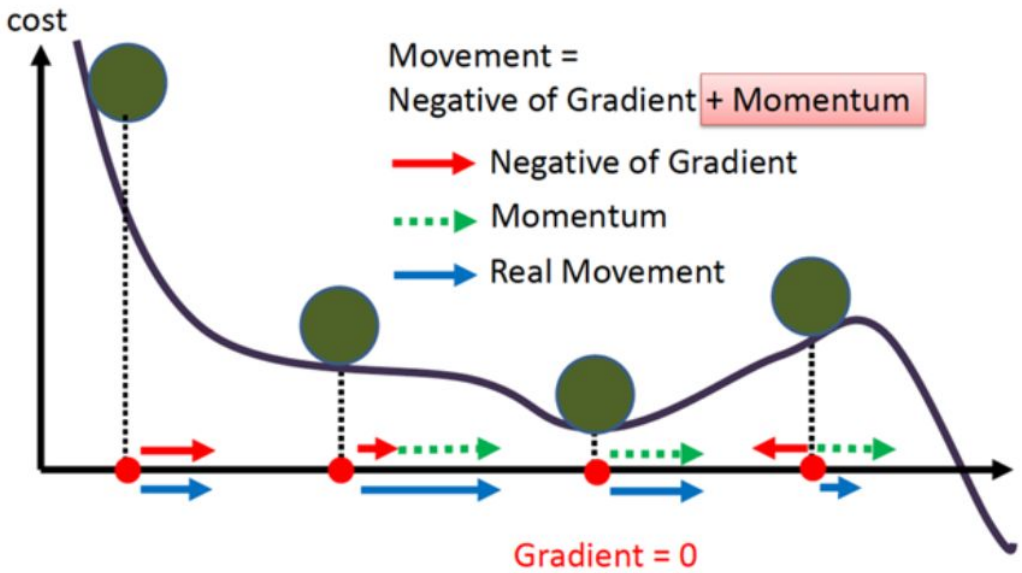
- 학습률(Learning Rate, eta, Step Size)이 너무 작을 경우
  - 알고리즘이 수렴하기 위해 반복해야 하는 값이 많으므로 학습시간이 오래걸림.
  - 지역 최소값(local minimum)에 수렴할 수 있음.

- 학습률(Learning Rate, eta, Step Size)이 너무 클 경우
  - 학습 시간은 적게 걸림.
  - 그러나 스텝이 너무 커서 전역 최소값(global minimum)을 가로질러 반대편으로 건너뛰어 최소값에서 멀어질 수 있음.



모멘텀 (Momentum)

- **경사하강법(Grandient Descent)의 해결방안** - 탄력[가속도]
- 기울기에 관성을 부과하여 작은 기울기는 쉽게 넘어갈 수 있도록 만든 것.
- 공을 예로 들면, 언덕에서 공을 굴렸을 때 낮은 언덕은 공의 관성을 이용하여 쉽게 넘어갈 수 있게 하여 지역 최소값을 탈출할 수 있게 한다는 뜻.

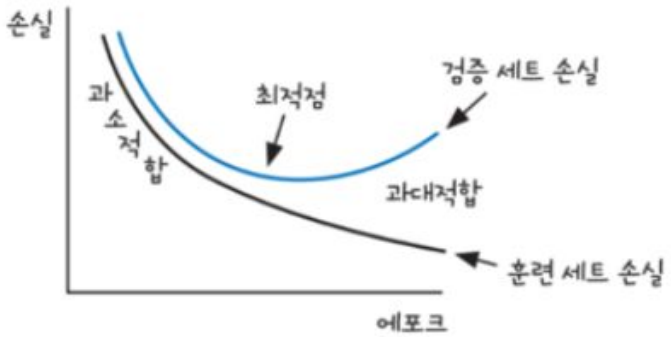


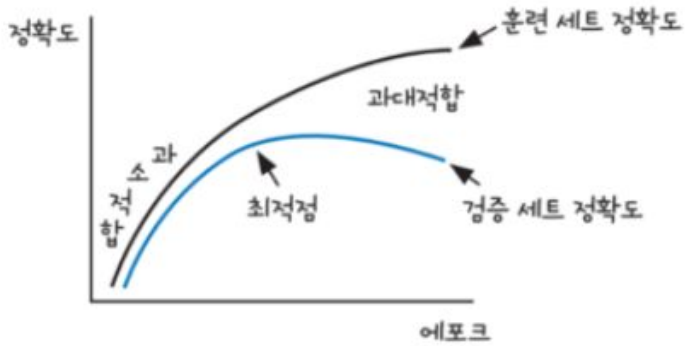
Epoch (에포크)

- 훈련 데이터셋에 포함된 모든 데이터들이 한 번씩 모델을 통과한 횟수로, 모든 학습 데이터셋을 학습하는 횟수
- 1 epoch: 전체 학습 데이터셋이 한 신경망에 적용되어 순전파와 역전파를 통해 신경망을 한번 통과했다는 의미.
- epoch를 높일수록 다양한 무작위 가중치로 학습을 해보는 것이므로 손실값이 내려감
- 그러나 지나치게 epoch를 높이게 되면, 그 학습 데이터셋에 과적합(overfitting)되어 다른 데이터에 대해선 제대로 된 예측을 하지 못할 수 있음.

Epoch (에포크)

- 훈련 세트 정확도는 에포크가 진행될수록 꾸준히 증가하는 반면,
  테스트 세트 점수는 어느 순간 감소하기 시작함.
  - 이 시점이 모델이 **과대적합되기 시작**하는 곳.
  - **조기 종료(early stopping):** 과대적합이 시작하기 전에 훈련을 멈추는 것.





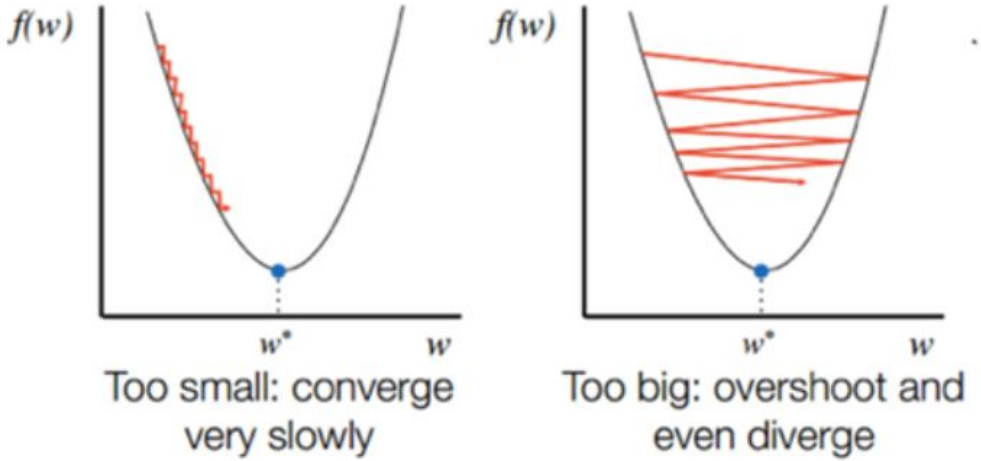
Batch, Epoch


batch size
훈련셋
1 epoch
batch
2 epoch
batch
N epoch
batch
bskyvision.com
확대
샘플

학습률(Learning Rate, eta)

한 번 학습할 때 얼마만큼 학습해야 하는지를 의미 - 학습한 이후에 가중치 매개변수가 매번 갱신됨

- 학습률을 너무 크게 잡으면 한 점으로 수렴하지 않고 발산



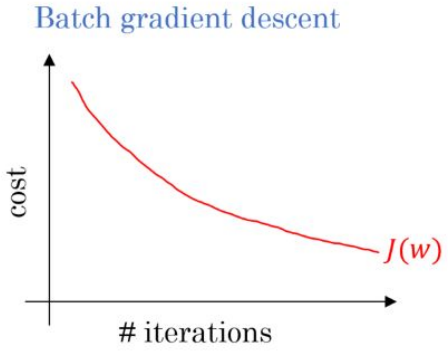
https://angeloyeo.github.io/2020/08/16/gradient_descent.html

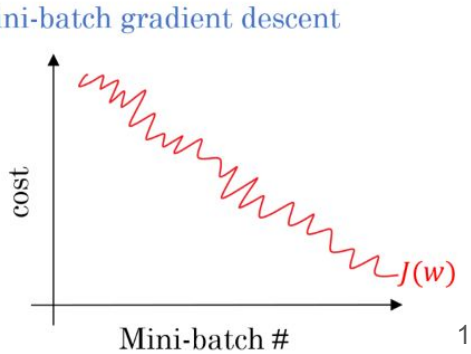
미니 배치 경사 하강법 (Mini-Batch Gradient Descent)

- 미니 배치 경사 하강법은 배치 크기를 줄이고, 확률적 경사 하강법을 사용.

  ex) 학습 데이터가 1000개이고, batch size를 100으로 잡았다고 할 때 총 10개의 mini batch가 나옴.
  이 mini batch 하나당 한번씩 SGD 진행하므로 1 epoch당 총 10번의 SGD 진행

- 배치 크기는 총 학습 데이터셋의 크기를 배치 크기로 나눴을 때, 딱 떨어지는 크기로 하는 것이 좋음.





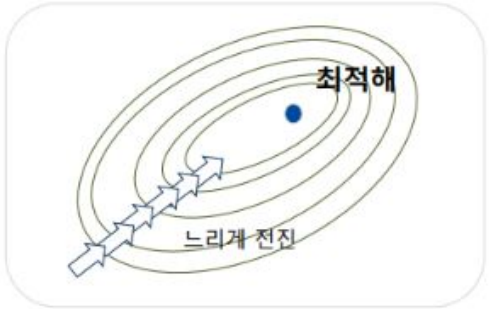
딥러닝시 고려 사항

속도와 정확도 문제를 해결하는 고급 경사 하강법

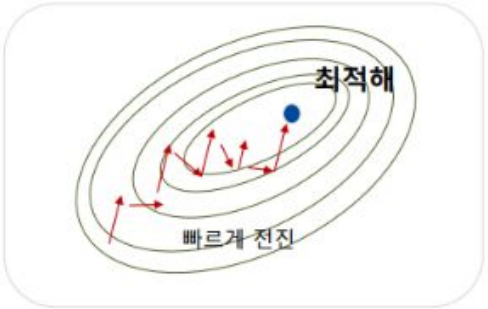
- 경사 하강법은 정확하게 가중치를 찾아가지만, 한 번 업데이트할 때마다

전체 데이터를 미분해야 하므로 계산량이 매우 많다는 단점

- 확률적 경사 하강법(SGD)
  - 전체 데이터를 사용하는 것이 아니라, 랜덤하게 추출한 일부 데이터를 사용
  - 일부 데이터를 사용하므로 더 빨리 그리고 자주 업데이트를 하는 것이 가능해짐



경사 하강법

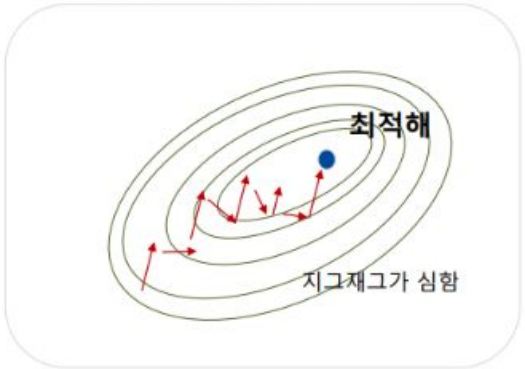


확률적 경사 하강법

딥러닝시 고려 사항

속도와 정확도 문제를 해결하는 고급 경사 하강법

- 모멘텀
  - 경사 하강법과 마찬가지로 매번 기울기를 구하지만, 이를 통해 오차를 수정하기 전 바로 앞 수정 값과 방향(+, -)을 참고하여 같은 방향으로 일정한 비율만 수정되게 하는 방법

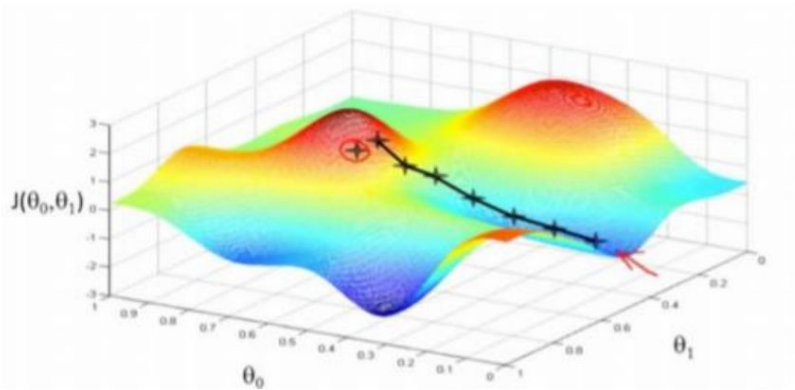


확률적 경사 하강법

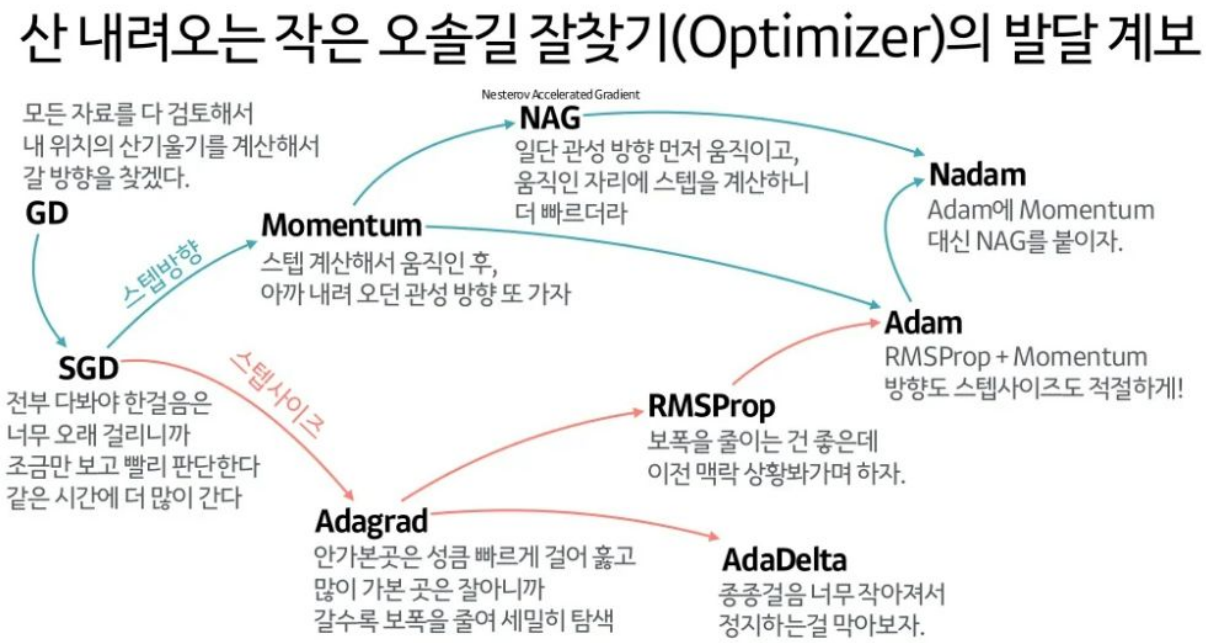
모멘텀을 적용한 확률적 경사 하강법

딥러닝시 고려 사항

- 지역최적화 방지 기법
- 가중치 갱신 방향
  - 가중치 갱신 방법 : 옵티마이저 : https://imgur.com/NKsFHJb
- 옵티마이저
  - 기울기 X 학습률 연산
  - 기울기 : 가중치 갱신 방향 정보
  - 학습률 : 데이터를 학습하면서 긍정적으로 변경될 수 있게 설계



딥러닝시 고려 사항



자습해도 모르겠던 딥러닝, 머리속에 인스톨 시켜드립니다.

딥러닝시 고려 사항

- 고급 경사 하강법과 케라스 내부에서의 활용법

고급 경사 하강법	개요	효과	케라스 사용법
1. 확률적 경사 하강법 (SGD)	랜덤하게 추출한 일부 데이터를 사용해 더 빨리, 자주 업데이트를 하게 하는 것	속도 개선	keras.optimizers.SGD(lr = 0.1) 케라스 최적화 함수를 이용합니다.
2. 모멘텀 (Momentum)	관성의 방향을 고려해 진동과 폭을 줄이는 효과	정확도 개선	keras.optimizers.SGD(lr = 0.1, momentum = 0.9) 모멘텀 계수를 추가합니다.
3. 네스테로프 모멘텀 (NAG)	모멘텀이 이동시킬 방향으로 미리 이동해서 그레이디언트를 계산. 불필요한 이동을 줄이는 효과	정확도 개선	keras.optimizers.SGD(lr = 0.1, momentum = 0.9, nesterov = True) 네스테로프 옵션을 추가합니다.

딥러닝시 고려 사항

- 고급 경사 하강법과 케라스 내부에서의 활용법

4. 아다그라드 (Adagrad)	변수의 업데이트가 잦으면 학습률을 적게 하여 이동 보폭을 조절하는 방법	보폭 크기 개선	keras.optimizers.Adagrad(lr = 0.01, epsilon = 1e − 6) 아다그라드 함수를 사용합니다. ※ 참고: 여기서 epsilon, rho, decay 같은 파라미터는 바꾸지 않고 그대로 사용하기를 권장하고 있습니다. 따라서 lr, 즉 learning rate(학습률) 값만 적절히 조절하면 됩니다.
5. 알엠에스프롭 (RMSProp)	아다그라드의 보폭 민감도를 보완한 방법	보폭 크기 개선	keras.optimizers.RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e − 08, decay = 0.0) 알엠에스프롭 함수를 사용합니다.
6. 아담(Adam)	모멘텀과 알엠에스프롭 방법을 합친 방법	정확도와 보폭 크기 개선	keras.optimizers.Adam(lr = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e − 08, decay = 0.0) 아담 함수를 사용합니다.

딥러닝시 고려 사항

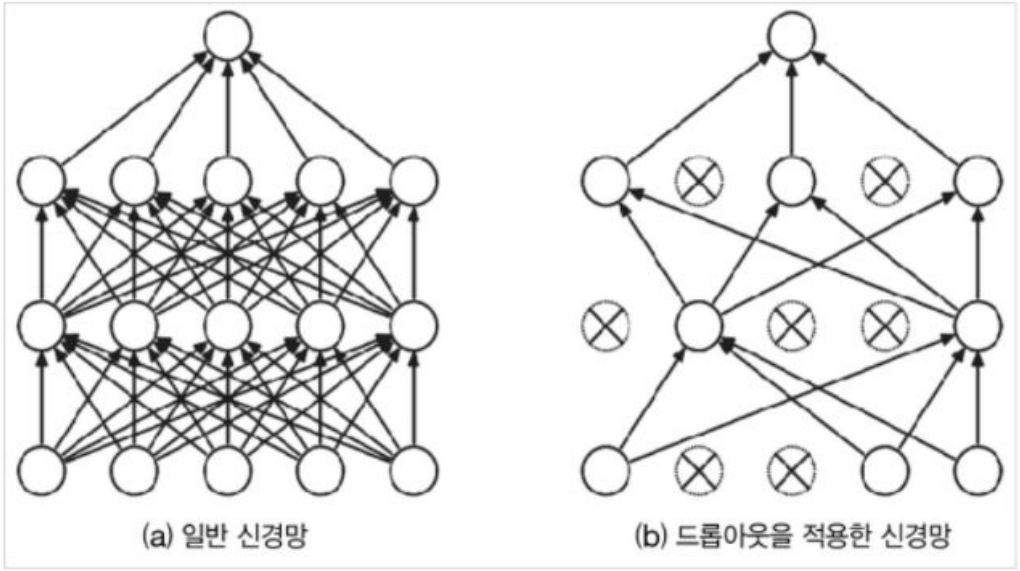
과적합(overfitting)

- 모델이 학습 데이터셋 안에서는 일정 수준 이상의 예측 정확도를 보이지만, 새로운 데이터에 적용하면 잘 맞지 않는 것을 의미
- 학습한 데이터에만 종속적으로 맞춰져 있어 그 외 데이터는 잘 맞지 않는 상황을 의미

- Overfitting에 대한 해결책 : Dropout(망 부분 생략) 적용한 분류 DNN
- 드롭아웃(Dropout)
  - 과적합을 해결하는 가장 효과가 좋은 방법
  - 단순한 기법을 통해 신경망의 일반화 성능을 향상시킬 수 있는데, 학습시킬 때 무작위로 뉴런을 '드롭아웃 (제외)'하기임
  - 드롭 아웃을 적용하면 실질적으로 많은 모델을 생성, 학습하고 그 안에서 예측을 실행하기 때문에 성능이 향상됨

딥러닝시 고려 사항

- 드롭아웃(Dropout)
  - 망에 있는 입력 layer나 hidden layer의 일부 뉴런을 생략(drop out)하고 줄어든 신경망을 통해 학습 수행 일정
  - mini-batch 구간 동안 생략 된 망에 대한 학습을 끝내면 다 시 무작위로 다른 뉴런들을 생략하면서 반복적으로 학습 수행



흙으실분