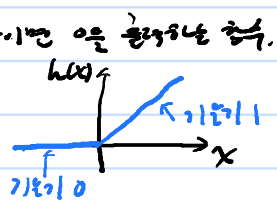
ReLU (Rectified Linear Unit)

- 입력이 0을 넘으면 입력을 그대로 흘리고, 0 이하이면 0을 흘리는 함수.

$$h(x) = \begin{cases} x & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$



원핫 벡터 (One-Hot vector)

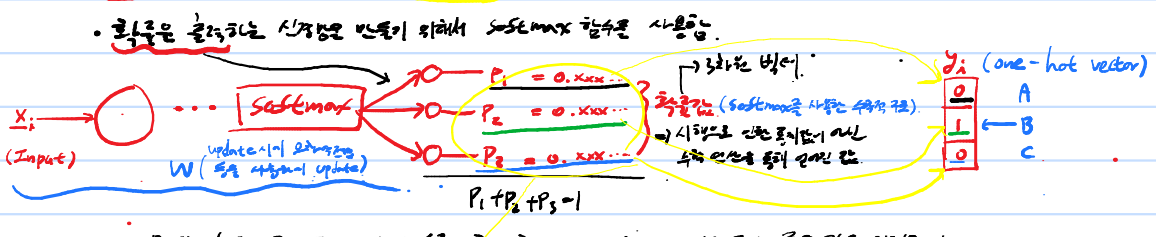
- 여러 class 중 어떤 class에 속하는지를 이진 값으로 표현하는 방법.
- 벡터의 차원은 class의 가짓수와 같다.
- 오직 class의 원소만 1, 나머지는 0값을 갖음.

A → $\underline{t}_A = [1, 0, 0]$
B → $\underline{t}_B = [0, 1, 0]$
C → $\underline{t}_C = [0, 0, 1]$

⇒ 여러 class 문제

교차 엔트로피 오차 (Cross-Entropy Error)

- 다중 class 분류 문제를 효율적인 모델에 사용함. (확률적인 값을 다루는 모델에서 사용할 수 있는 오차)
- 신경망이 출력하는 확률과 정답레이블을 비교하는 방식으로 오차를 계산함.
- 확률을 출력하는 신경망을 만들기 위해서 softmax 함수를 사용함.



⇒ P_2가 높고 P_1, P_3가 낮을수록 훈련한 신경망의 파라미터들이 좋은 값을 가지고 있음.

⇒ 처음에는 랜덤한 값으로 파라미터들이 초기화되어 있음. (오차는 커지겠지)

다중클래스 분류문제의 교차엔트로피 오차

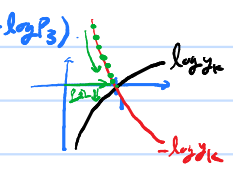
⇒ $L = -\sum_k t_k \log y_k$

y_k : 어떤 한 확률
t_k : 원핫 벡터로 표현된 정답레이블의 k번째 원소

- N개의 Data를 한 번에 고려하는 미니배치를 사용하는 경우.

⇒ $L = -\frac{1}{N}\sum_n \sum_k t_{nk} \log y_{nk}$

ex) $-(t_1 \log y_1 + t_2 \log y_2 + t_3 \log y_3)$
$= -(0 \cdot \log P_1 + 1 \cdot \log P_2 + 0 \cdot \log P_3)$
$= -\log P_2$ 정답 label 값만 남음 (1에 가까워질수록 좋음)



→ P_2가 1에 가까워질수록 교차엔트로피 오차는 점점 0에 가까워지므로 이를 최적화.

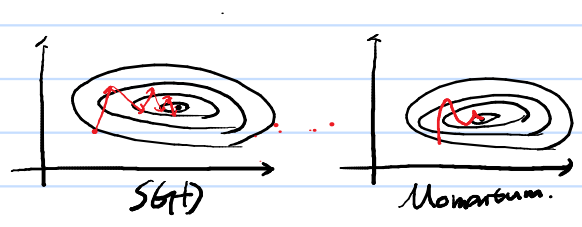
최적화 알고리즘

1. SGD 2. Momentum 3. AdaGrad 4. Adam

- **Momentum**

$v \leftarrow \alpha \cdot v - \eta \frac{\partial L}{\partial w}$ (v를 update)
$w \leftarrow w + v$ (v로 w를 update)

→ 운동량 (속도 벡터) 개념을 추가한 가중치 갱신 방식.
→ parameter tuning ; α, η 두 개.



- **AdaGrad (Adaptive gradient)**

$h \leftarrow h + \frac{\partial L}{\partial w} \odot \frac{\partial L}{\partial w}$ ← ⊙ : 각 원소별 곱셈 (중요)
$w \leftarrow w - \eta \frac{1}{\sqrt{h}} \cdot \frac{\partial L}{\partial w}$

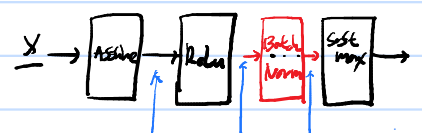
학습률 η를 매번 조절해주는 오차
(기울기가 크면 조금 update.
기울기가 작으면 많이 update.)

- **Adam**

→ Momentum, Adagrad를 융합한 기법.

- 최적화 알고리즘마다 학습률 등 hyperparameter를 어떻게 설정하느냐에 따라 결과가 달라짐.
- 모든 종류의 문제에서 항상 뛰어난 하나의 기법이 존재하는 것은 아니고, 각각 장단점이 있음.
  - SGD, Adam ...

배치 정규화 (Batch Normalization)



* 오차가 덜 발생하는 것은 Network이고 배치 정규화를 사용해서 학습의 품질을 높일 수 있음.

(정규화의 수치적 지표의 편차로 인해 오차가 발생할 수 있음)
↳ 0 근처의 값이 0이 되어버리면 아주 큰 값에 ∞가 폭발적이

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$
$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$ ⇒ 새로운 측정데이터 정규화.

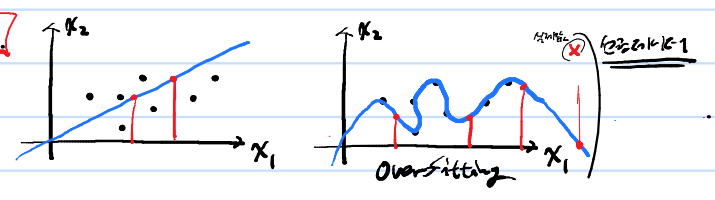
⇒ 각 layer의 x까지 분포가 큰 것을 정규화를 시켜 평균이 0이되고 분산이 1이 되도록
- 미니배치 $B = \{x_1, x_2, \cdots, x_m\}$에 대해, 데이터 분포의 평균이 0, 분산이 1이 되도록 만들어줌. ⇒ 배치 정규화
- ε은 임의의 매우 작은 값을 사용해서 0으로 나누는 에러가 발생하지 않게 함. (σ_B^2가 0이 될 수 있음)

⇒ 데이터 분포가 조정, 특정한 목적에 X

- 훈련과정의 중간에 들어가 배치 정규화 과정을 집어넣는다.

오버피팅 (overfitting)

- 신경망이 훈련데이터에만 과도하게 적응해, 새로운 데이터에는 제대로 대응하지 못하는 상태.



→ 매개변수가 많은 경우. (다항식의 차수가 엄청 높음)
→ 훈련 데이터가 적은 경우.

가중치 감소 (Weight Decay)

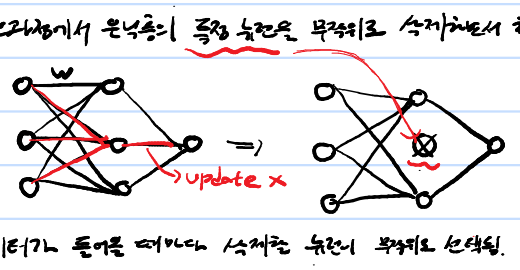
- 오버피팅은 가중치 매개변수 값이 매우 큰 값을 사용하는 경우가 많음.
- 학습 과정에서 큰 가중치를 갖는 경우 패널티를 부과하는 방법. ← 규제, Regulation.
- 가중치의 L2 norm을 손실함수에 추가하는 방법

$$L = n + \frac{1}{2}\lambda w^2$$

n : 손실함수
λ : 정규화/규제의 강도를 지정하는 하이퍼파라미터.
($\lambda \uparrow$ → 페널티 ↑
$\lambda \downarrow$ → 페널티 ⇓)
↳ 매개변수값 ↑ ⇒ 손실값 ↑

드롭아웃 (DropOut)

- 신경망 모델이 복잡해질 때 주로 사용.
- 훈련과정에서 은닉층의 특정 뉴런을 무작위로 삭제하면서 학습.



⇒ 훈련할 때마다 특정한 뉴런을 삭제하는 형태로 계속 훈련하는 게 아니라 새로운 데이터가 들어올 때마다 어떤 뉴런이 삭제되는지 무작위로 선택.

- 데이터가 들어올 때마다 삭제할 뉴런이 무작위로 선택됨.
- 훈련이 끝난 이후 예측 단계에는 모든 뉴런을 사용함.
- 일종의 앙상블 학습, ensemble learning으로 볼 수 있음. (오버피팅 방지)

신경망이 크기는 훈련하는 데 복잡 삭제/하는지에 따라서 서로 다른 신경망, 서로 다른 신경망이 같은 목표를 가지고 학습. 마지막에 합쳐져서 모든 뉴런들이 학습 발휘

⇒ 훈련을 통해 좋은 성능을 내는 신경망을 만들고자 함.

검증 데이터 (Validation Data)

- 훈련데이터 → 매개변수 (w) 학습에 사용.
- 검증 데이터 → 하이퍼파라미터 튜닝에 사용.
- 시험 데이터 → 범용적 성능 평가.

(하이퍼파라미터 튜닝을 시험 데이터로 하면 시험 데이터에 대한 오버피팅 될 수 있음)

하이퍼파라미터 최적화.

1. 하이퍼파라미터 값 범위를 설정한다.
2. 범위 내에서 무작위로 추출한다.
3. 추출한 하이퍼파라미터로 학습하고, 검증 데이터로 정확도를 평가한다.
4. (2~3)단계를 여러번 실시, 가장 좋은 결과를 내는 결과를 바탕으로 하이퍼파라미터 값 범위를 재설정한다. (1단계)

⇒ 적절 범위를 좁혀가며 최종적으로 특정 하이퍼파라미터를 결정하고 test dataset으로 성능평가.