쿠글 6주차 - 유재성

1. 순전파(Forward Propagation)

- NN 모델의 입력층부터 출력층까지 순서대로 변수들을 계산하고 저장(bias는 생략)

$$z = W^{(1)}x$$

- $W^{(1)}$은 은닉층(hidden layer)의 가중치 파라미터, 중간 변수 z를 활성화 함수(activation function)ϕ에 입력해서 벡터 길이가 h인 은닉층(hidden layer)변수를 얻음

$$h = \phi(z)$$

- 은닉 변수 h도 중간 변수
- 출력층의 가중치 $W^{(2)}$만을 사용한다고 가정하면, 출력층의 변수 다음과 같음

$$o = W^{(2)}h$$

- 손실 함수(loss function)를 l이라고 하고, 샘플 레이블을 y라고 가정하면, 하나의 데이터 샘플에 대한 손실(loss)값은 다음과 같음

$$L = l(o, y)$$

- l_2놈 정규화(norm regularization)의 정의에 따라, 하이퍼파라미터(hyper-parameter) λ가 주어졌을 때, 정규화 항목은 다음과 같음

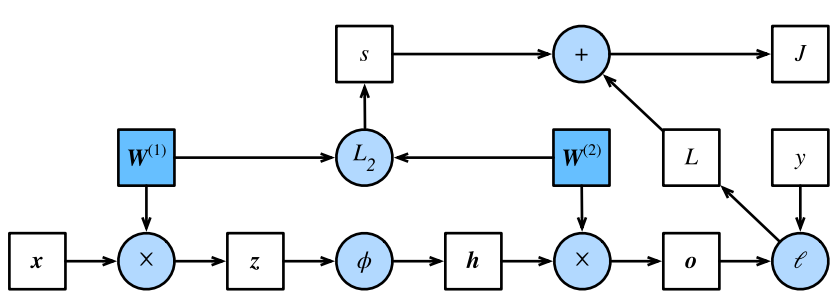
$$s = \frac{\lambda}{2}(\|W^{(1)}\|_F^2 + \|W^{(2)}\|_F^2)$$

- 여기서 행렬의 Frobenius norm은 행렬을 벡터로 바꾼 후 계산하는 L_2놈과 같음.
- 마지막으로, 한 개의 데이터 샘플에 대한 모델의 정규화된 손실(regularized loss)값 계산

$$J = L + s$$

- J는 주어진 데이터 샘플에 대한 목표 함수(objective function)

2. 순전파의 연산 그래프



- 왼쪽 아래는 입력, 오른쪽 위는 출력

3. 역전파(back propagation)

- 역전파는 NN 파라미터에 대한 그래디언트(gradient)를 계산하는 방법
- 일반적으로 NN layers와 관련된 목적 함수의 중간 변수들과 파라미터들의 그래디언트를 **출력층에서 입력층 순으로 계산**하고 저장 (∵ chain rule)
- 하나의 은닉층을 갖는 NN인 경우
- $J = L + s$의 그래디언트 계산

$$\frac{\partial J}{\partial L} = 1 \quad and \quad \frac{\partial J}{\partial s} = 1$$

- 출력층 o의 변수들에 대한 목적 함수의 그래디언트에 체인룰 적용

- $prod$ 연산은 전치(transpose)같은 필요 연산을 수행후 곱을 수행하는 것을 의미

$$\frac{\partial J}{\partial o} = prod(\frac{\partial J}{\partial L}, \frac{\partial L}{\partial o}) = \frac{\partial L}{\partial o}$$

- 두 파라미터에 대해 정규화 항목의 그래디언트 계산

$$\frac{\partial s}{\partial W^{(1)}} = \lambda W^{(1)} \, and \, \frac{\partial s}{\partial W^{(2)}} = \lambda W^{(2)}$$

- 출력층과 가장 가까운 모델 파라미터들에 대해 목적함수의 그래디언트$\frac{\partial J}{\partial W^{(2)}}$를 다음과 같이 체인룰을 적용하여 계산

$$\frac{\partial J}{\partial W^{(2)}} = prod(\frac{\partial J}{\partial o}, \frac{\partial o}{\partial W^{(2)}}) + prod(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial W^{(2)}}) = \frac{\partial J}{\partial o} h^T + \lambda W^{(2)}$$

- $W^{(1)}$에 대한 그래디언트는 출력층으로부터 은닉층까지 역전파를 진행해야함. 먼저 은닉층 변수에 대한 그래디언트는 다음과 같음

$$\frac{\partial J}{\partial h} = prod(\frac{\partial J}{\partial o}, \frac{\partial o}{\partial h}) = {W^{(2)}}^T \frac{\partial J}{\partial o}$$

- 활성화 함수ϕ는 각 요소별로 적용되기 때문에, 중간 변수 z에 대한 그래디언트는 요소별 곱하기 연산자를 사용해야함 이를 $\odot$이라 한다면

$$\frac{\partial J}{\partial z} = prod(\frac{\partial J}{\partial h}, \frac{\partial h}{\partial z}) = \frac{\partial J}{\partial h} \odot \phi'(z)$$

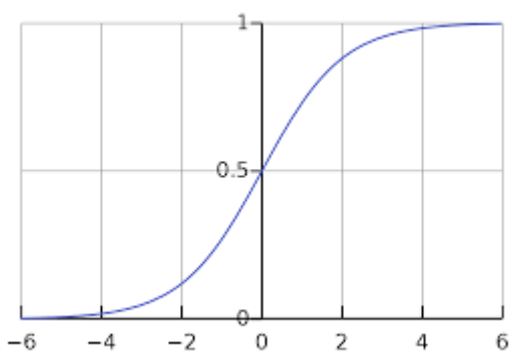
- 마지막으로 입력층과 가장 가까운 모델 파라미터에 대한 그래디언트 $\frac{\partial J}{\partial W^{(1)}}$

$$\frac{\partial J}{\partial W^{(1)}} = prod(\frac{\partial J}{\partial z}, \frac{\partial z}{\partial W^{(1)}}) + prod(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial W^{(1)}}) = \frac{\partial J}{\partial z} x^T + \lambda W^{(1)}$$

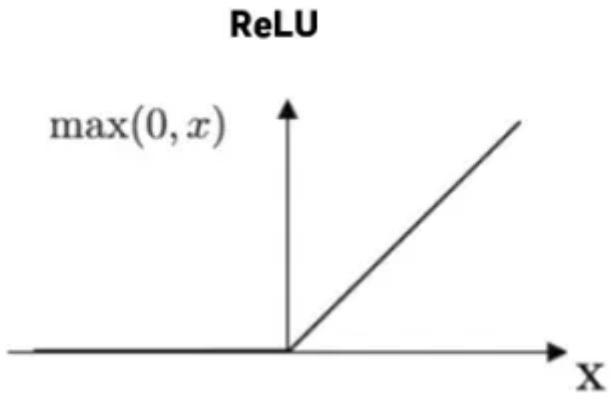
4. 초기 NN이 가지고 있던 문제점 & 해결

1) Gradient Vanishing(기울기 소멸 문제)

- 은닉층이 많은 다층 퍼셉트론에서 은닉층을 많이 거칠수록 전달되는 오차가 크게 줄어들어 학습이 되지 않는 현상



- 이를 Relu와 같은 함수를 활성화함수로 선택하면 해결할 수 있음



2) Weight Initialization(가중치 초기화)

- 초기값을 모두 0으로 설정한 경우 모든 뉴런이 같은 값을 나타냄 → 역전파 과정에서 각 가중치의 업데이트가 동일하게 이뤄짐 → 학습 효과 감소

- Xavier Initialization
  - 이전 노드와 다음 노드의 개수에 의존하는 방법
  - 비선형함수(tanh)에서 효과적
  - ReLU함수에서 사용 시 출력 값 0으로 수렴하는 문제점

$$W \sim N(0, Var(W))$$
$$Var(W) = \sqrt{\frac{2}{n_{in} + n_{out}}}$$

- He Initialization
  - ReLU를 활성화 함수로 사용 시 주로 사용하는 초기화 방법

$$W \sim N(0, Var(W))$$
$$Var(W) = \sqrt{\frac{2}{n_{in}}}$$

3) Batch Normalization(배치 정규화)

- 가중치 소멸 또는 폭발 문제를 해결하기 위한 접근 방법 중 하나
- 활성화함수의 활성화값 또는 출력값을 정규화하는 작업
- 학습 속도 개선, 가중치 초깃값 선택의 의존성 감소, 과적합 위험 감소

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

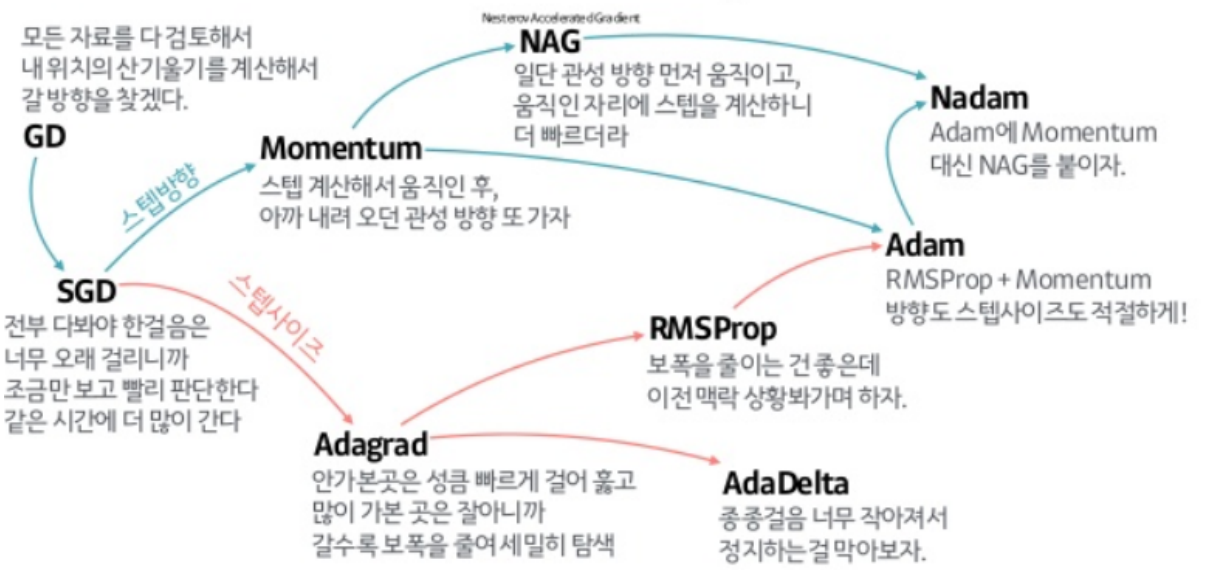
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

4) Optimizer

- Gradient Descent는 W와 loss를 고려해 업데이트 시켜주는 것
- nn이 매우 클 때 속도가 느려지는 문제가 있음
- 과정에서 속도와 방향이 중요함



- Adam Optimization

> 잘 모르겠으면 Adam을 써라 - 누군가

  - Adagrad + RMSProp의 장점을 섞어 놓은 것

5) Overfitting

- Bias & Variance
  - High bias(underfitting): layer 수 늘리기, epoch늘리기 등
  - High variance(overfitting): 정규화 수행, data늘리기 등
- Regularization
  - 일반적으로 파라미터값이 커질수록 과적합이 발생하기 때문에 L1, L2 항을 추가하여 파라미터 크기를 제약 → 과적합 방지

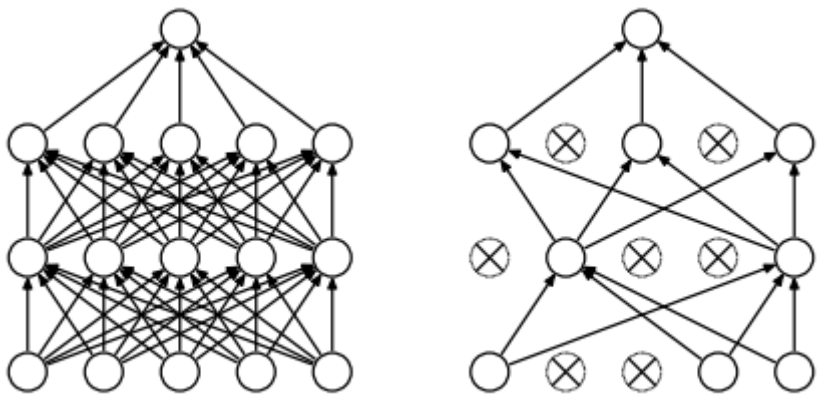
L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

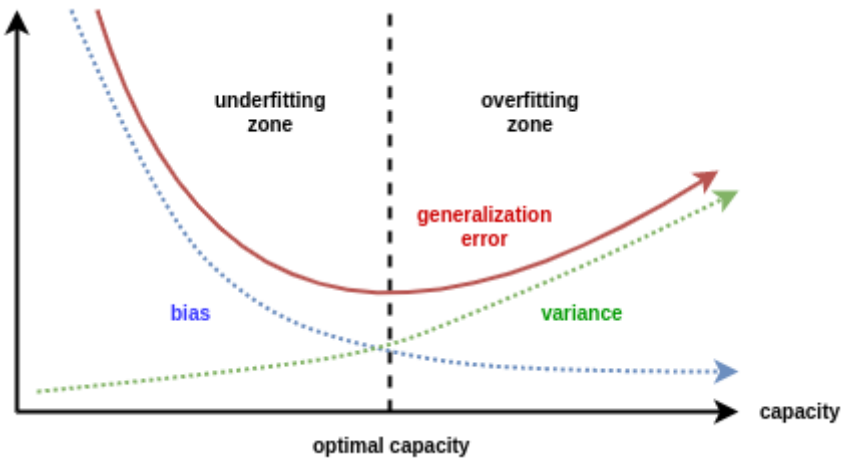
- Dropout
  - "node를 꺼버리는" 기법
  - 활성화 함수를 통과한 값 중 일부를 랜덤하게 0으로 만듦
  - 랜덤하게 만들어주는 비율은 하이퍼 파라미터



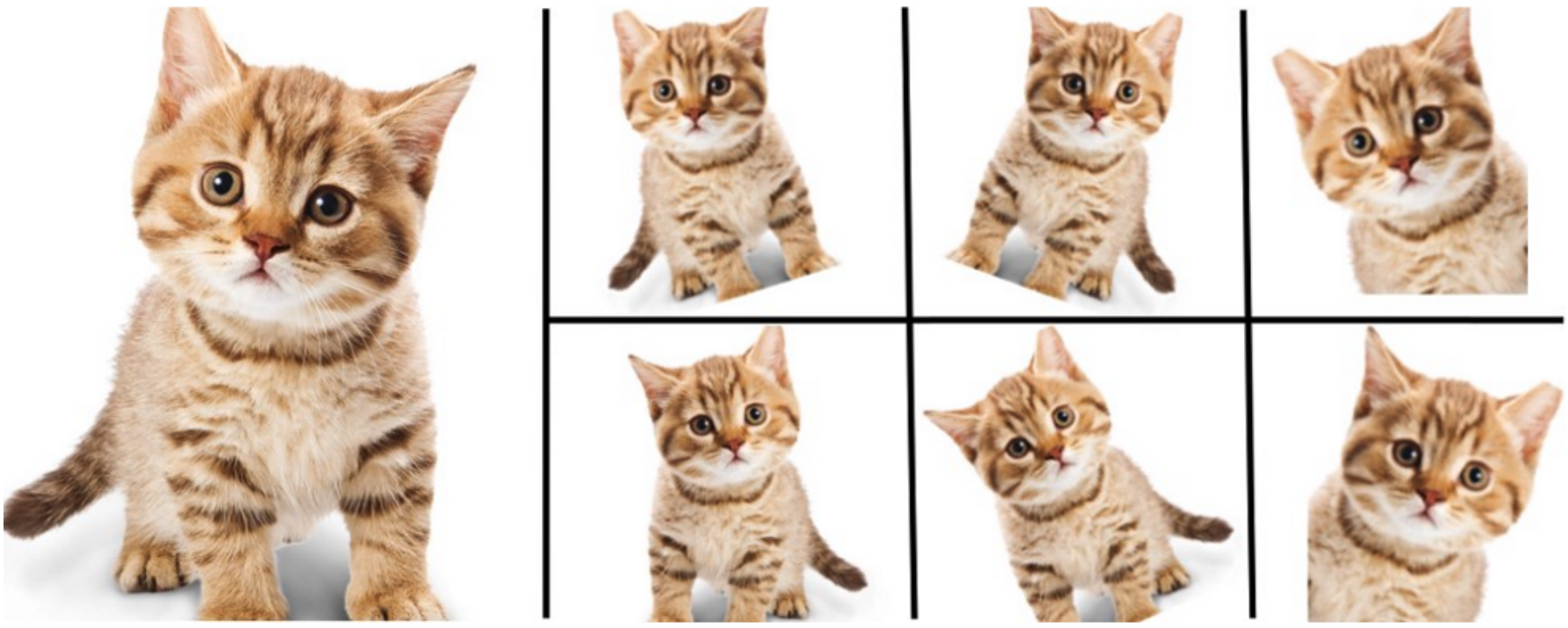
(a) Standard Neural Net

(b) After applying dropout.

- Early Stopping
  - Validation loss가 증가하는 시점에서 학습을 멈춤
  - 과적합 방지



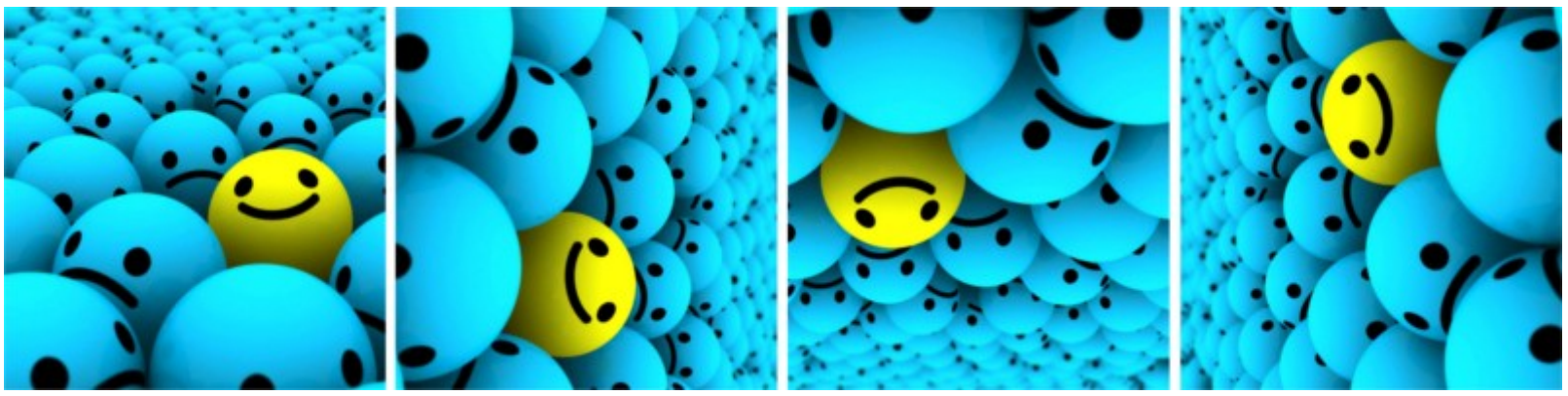
- Data Augmentation



- 임의의 잡음이나 translation을 가해서 test set과의 괴리를 줄이는 목적

1. 영상 좌우 또는 위 아래 반전(flip): 반대 방향의 객체가 있을 가능성이 있을 때

2. 영상 회전(rotation): test set에 회전된 객체가 있을 가능성이 있을 때



3. 영상 줌인, 줌 아웃 조절(scale)

4. 영상 일부 자르기(crop): 일부만 찍혀있는 경우를 보상 가능

5. 영상 위치 변환(translation): 4와 마찬가지

6. 가우시안 잡음: 영상의 high frequency 성분 억제에 좋음

- data set의 특성을 고려하여 특정 data augmentation 적용을 고려해야 함
  - 예를 들어 글자를 인식하는 문제에서 글자 이미지를 좌우또는 상하로 뒤집는 기법은 적당하지 않음 → 글자의 의미가 왜곡될 가능성 큼
  - 심장을 촬영한 영상도 마찬가지로 뒤집기 기법이 적절하지 않음