Linear Regression: 하나의 벡터 x를 입력으로 받아서, 스칼라 y를 예측한다. 학습을 진행할 때는 현재의 예측 결과와 실제 결과 값의 MSE 값이 줄어드는 방향으로 가중치 w 값을 업데이트한다.

$$\nabla_{\boldsymbol{w}} \text{MSE}_{\text{train}} = \nabla_{\boldsymbol{w}} \frac{1}{m} ||\hat{\boldsymbol{y}}^{(\text{train})} - \boldsymbol{y}^{(\text{train})}||_2^2 = 0$$

Normal Equation의 경우, 최적의 해(Solution)은 다음과 같다.

$$\boldsymbol{w} = \left(\boldsymbol{X}^{(\text{train})\top} \boldsymbol{X}^{(\text{train})} \right)^{-1} \boldsymbol{X}^{(\text{train})\top} \boldsymbol{y}^{(\text{train})}$$

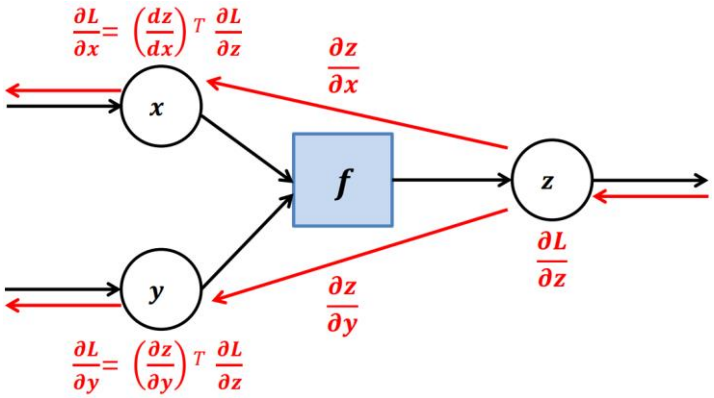
Bias-Variance Trade Off: Train Data에 너무 잘 맞게 학습을 시키면 모델의 복잡도가 높아지며 Train Loss는 0이 된다 (Over-fitting). 이 경우 Variance가 커져서 Test Loss는 오히려 증가할 수 있다. 그렇다고 모델의 복잡도를 너무 낮추면 Bias가 커져서 Test Loss가 오히려 증가할 수 있다 (Under-fitting). 따라서 Test Loss (MSE)가 가장 작아지는 최적의 복잡도(Optimal Capacity)를 찾도록 학습을 진행해야 한다.

$$\text{MSE} = \mathbb{E}[(\hat{\theta}_m - \theta)^2] = \text{Bias}(\hat{\theta}_m)^2 + \text{Var}(\hat{\theta}_m)$$

Back-Propagation: 역전파 알고리즘은 학습 시에 가중치를 업데이트 하기 위해 기울기 (Gradient) 값을 계산해주는 알고리즘이다. 레이어가 중첩되어 있다면 연쇄 법칙 (Chain Rule)을 이용한다. 연쇄 법칙은 다음과 같다.

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

실제 뉴럴 네트워크에서 기울기 값을 계산하는 예시는 다음과 같다.



Regularization: Generalization Error를 줄이기 위한 모든 방법을 이르는 말이다.

Norm Penalty: 모델의 복잡도 (Capacity)를 제한하는 대표적인 방법이다.

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha \Omega(\boldsymbol{\theta})$$

Early Stopping: 네트워크가 너무 작은 Train Loss를 가지지 않도록 학습하는 방법이다. 모델 가중치 복사만 하면 되고, 학습을 다시 할 필요는 없다 (Efficient). 또한 모델이나 알고리즘을 변경할 필요는 없어서 간단하다 (Simple).

Bagging (Bootstrap Aggregating): Sample을 여러 개 뽑아 각 모델을 학습시켜 결과를 집계 (Aggregating)한다.

Boosting: Bagging처럼 복원 랜덤 샘플링을 하지만, 오답에 큰 가중치를 부여한다 (정확도 상승, Outlier에 취약).

Dropout: Training 시기에 랜덤하게 몇 개의 뉴런의 출력 값을 0으로 설정한다 (Testing 시기에는 생략하지 않을 확률 p를 가중치에 곱함). 여러 개의 서로 다른 신경망을 결합한 것과 유사한 효과를 낸다.

Language Models: Sequence of words에 확률을 부여하는 모델을 의미한다. 연쇄 법칙이 적용된다.

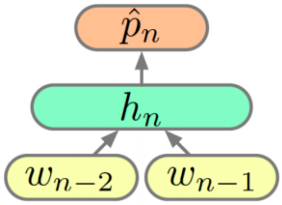
$$p(w_1, w_2, w_3, \dots, w_N) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \times \cdots \times p(w_N|w_1, w_2, \dots\ w_{n-1})$$

Count-based N-Gram Models: 하나의 단어의 확률을 예측할 때 이전 (N – 1)개의 단어의 확률을 고려한다.

$$p(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{count(w_1, w_2)}$$

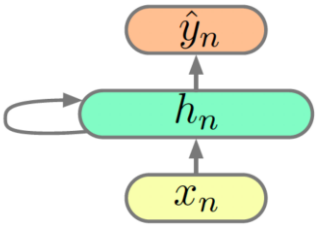
Neural N-Gram: Trigram NN 언어 모델의 예시는 다음과 같다. Count-based N-Gram 모델과 비교했을 때 Unseen 데이터에 더 잘 대응한다. N-Gram 모델과 마찬가지로 N-Gram History가 유한하다는 한계점은 동일하다.

$$h_n = g(V[w_{n-1}; w_{n-2}] + c)$$
$$\hat{p}_n = \text{softmax}(Wh_n + b)$$



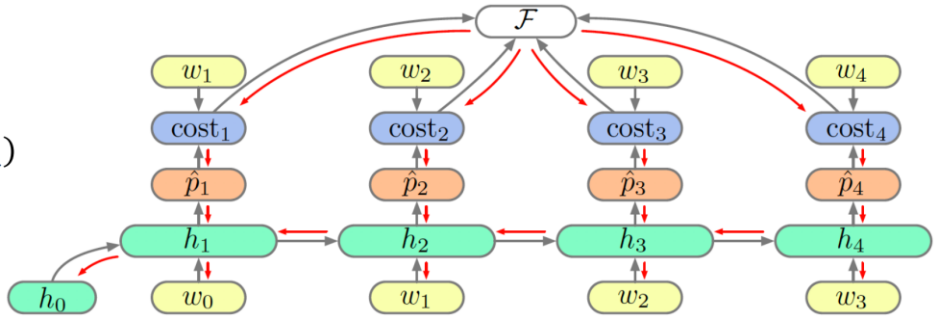
Recurrent Neural Networks (RNN): Unit간의 연결이 순환적인 구조를 가져, Sequential 데이터 처리에 적합하다. N-Gram 방식과 비교했을 때, 전체 History를 고정된 크기의 벡터에 담는다는 점이 특징이다 (More Long Range).

$$h_n = g(V[x_n; h_{n-1}] + c)$$
$$\hat{y}_n = Wh_n + b$$

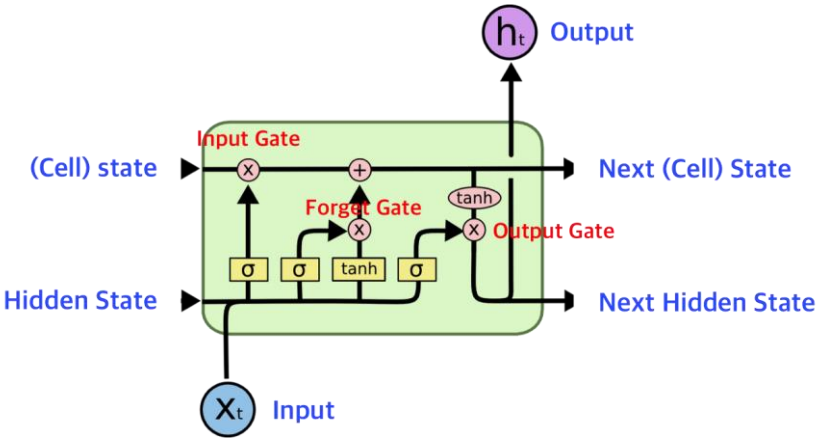


Back Propagation Thorough Time (BPTT): 기본적인 Unrolled Recurrent Network는 한꺼번에 역전파를 진행한다.

$$F = -\frac{1}{N}\sum_n \text{cost}_n(w_n, \hat{p}_n)$$



Long Short Term Memory (LSTM): RNN의 변형으로, Long-term Dependency를 학습하기 위해 사용한다. Cell State와 Hidden State를 모두 가지고 있다. Cell State이 추가되었는데, 오차를 없애지 않고 전체 Chain을 관통한다.

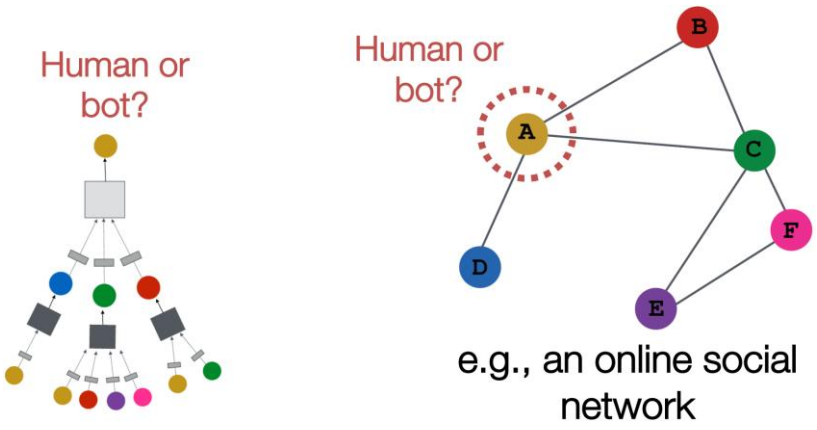


Stochastic Gradient Descent (SGD): Update weights for each sample (Fast, Online, but sensitive to noise).

Mini-batch SGD: Update weights for a small set of samples (Fast, Online, and robust to noise).

$$E = \frac{1}{2}\sum_{n\in B}(y^n - \hat{y}^n)^2 \qquad \boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) - \epsilon\frac{\partial E^B}{\partial \boldsymbol{w}_i}$$

Graph Convolution: 각 노드들은 인접한 노드 (Local Neighborhoods)들의 정보를 집계 (Aggregate)하여 노드 임베딩 (Node Embeddings)을 생성한다. 각 노드는 고유한 Computation Graph를 가지게 되며, Convolution을 수행할 때는 간단한 방법으로로 주변 노드의 Features의 평균을 이용할 수 있다. 일반적인 CNN과 마찬가지로, Convolution 부분의 파라미터는 모든 노드에게 같은 파라미터로 공유된다는 특징이 있다.



GGNN (Gated Graph Sequence Neural Networks): Neighborhood aggregation with RNN state update. 기본적으로 Graph Convolution의 레이어가 깊어지면 Overfitting, Vanishing/Exploding Gradients 문제가 발생할 수 있다. 이를 해결하기 위한 방법으로 GGNN을 사용할 수 있다.

Equivariance: 불변성 (Invariance)이란 변하지 않는 성질을 의미한다 (Equivariance의 한 종류). 이미지의 경우 좌우 반전이나 약간의 회전을 시켜도 Feature Representation이 동일해야 한다 (Equivariant). 실제로 CNN은 Max-pooling과 같은 기법을 이용하여 공간에 대해 Translation Equivariance를 가진다.

Group-Equivariant Representation: Representation with the structure of a linear G-space, for some group G.

Group-Equivariant Convolution Networks: 이동, 회전 등을 포함하는 Group Convolution Layers를 이용하여 Representation Learning 측면에서 높은 Equivariance를 가지도록 학습하는 네트워크이다. G-CNN은 일반적인 CNN에 비해서 Weight Sharing이 더 많이 이루어지는 방식으로 데이터를 모델링한다.

Empirical Risk Minimization: 기계 학습을 진행할 때 True Risk에는 접근할 수 없으므로, Empirical Risk를 이용하여 모델을 최적화한다.

Batch Normalization: 네트워크의 각 레이어에서의 데이터 (Batch)의 분포를 정규화하는 기법이다. 네트워크의 Overfitting을 방지하고, 가중치 초기화 문제, 그리고 학습 속도가 개선되는 특징이 있다.

Momentum: Gradient Descent 과정에서 속도 (Velocity) 파라미터를 추가하여, 가중치가 변화하는 방향으로 더 많이 변화할 수 있도록 한다.

AdaGrad: 학습 과정에서 Learning Rate를 조절하기 위해 Learning Rate Decay를 이용하는 방식이다. 다만 AdaGrad는 Convex Structure에 도달하기 전에 기울기가 너무 작아질 수 있는데, 이후에 제안되는 RMSProp은 Exponentially Weighted Moving Average를 이용하여 이 문제를 개선한다.