

Perceptron

def: 인공 뉴런을 위한 수학적 모델.

원리: 입력을 선형적으로 결합. 임계값 척도에 Y/N 결정.

입력에 대한 선형 결정 경계 생성.

$$h = \mathbf{w}^T \mathbf{x} + b \text{ 형태 계산} \Rightarrow \text{활성화 함수 적용}, \hat{y} = \sigma(h)$$

최종 출력: 입력의 가중합이 활성화 함수 적용한 결과.

활성화 함수: 가중합에 비선형 변환 추가해 모델의 비선형 관계 학습 가능.

... 활성화 함수가 없거나 선형이라면 전체 모델이 선형 모델과 일치

$$\text{e.g., sigmoid } (= \frac{1}{1+e^{-x}}), \text{ ReLU } (= \max(x, 0))$$

Backpropagation

손실함수에 대한 네트워크 매개변수들의 기울기 효율적 계산.

기울기 (gradient)를 통해 손실함수 최소화하는 업데이트 진행 방향 결정

Chain rule.

Forward pass : 각 계층의 매개변수와 함께 사용해 중간 변수, 최종 예측값 계산

(출력층 \rightarrow 입력층) 방향 $\Delta_w L, \Delta_b L$ 계산.

$$\text{update: } w = w - \eta \cdot \nabla_L(w)$$

{ GD: 예전 전체 데이터셋 사용.

mini-batch SGD : 매 iteration마다 훈련 데이터 셋에서 미니배치 무작위 추출 minibatch 대로 평균 기울기 계산, 전체 셋의 기울기 추정.

계산 효율↑. Stochastic \Rightarrow noise 추가 \Rightarrow local minima 탈출 가능.

학습률: 너무 작으면 최적화 느림, 너무 크면 발산/진동 발생 가능.

SGD 개선한 최적화 알고리즘.

Momentum: 속도 (velocity) 드립, 과거 기울기 가중이동평균 활용해 업데이트.

Nesterov momentum: 모멘텀 드립, 예측된 위치에서 기울기 계산.

AdaGrad: 매개변수별 기울기 동적 조절. 과거 모든 기울기의 누적합 사용.

기울기 큰 매개변수의 학습률 크게 감소. 기울기 작은 매개변수는 작게 감소.

RMSProp: AdaGrad 학습률 감소 해소 \Rightarrow 기울기 제곱평균 대신 기울기동가중평균 채택. 과거 정보 decay. 최신정보 비중↑.

Adam: 1st momentum: 기울기 자주동가중평균 계산. 모멘텀 합(방향) 추정.

2nd momentum: 기울기 제곱 자주동가중평균. 스케일링 합(크기) 추정.

Loss function 최소화하는 최적의 매개변수 찾기!

\hookrightarrow 모델 예측과 정답 간 불일치 정도 정량화. 초기: MSE. 분류: cross-entropy.

매개변수: 가중치 + 편향. 입력과 출력 사이 관계 결정.

기울기 (gradient): 각 매개변수가 아주 작은 양만큼 변화 때 손실함수가 얼마나 빠르게 변화하는 손실함수 표면에서 가장 가파르게 상승하는 방향. \Rightarrow 흡수 방향으로 업데이트!

Overfitting 모델이 훈련 데이터셋에 너무 맞춰져 훈련 데이터 특이점까지 학습. 정작 새로운 데이터에 대해 성능 저하 발생. \Rightarrow training error ↓ generalization error ↑.

why?: 모델이 너무 복잡(표현력 높다) or 훈련 데이터 부족

Regularization: 일반화 오류 감소 목적의 수정 사항. 과적합 방지. 일반화 성능 개선

고전: 손실함수 + 평활화 항. \Rightarrow 모델 복잡성 제한. (smoother)

$$J(w) = \text{MSE} + \lambda \Omega(w) \Rightarrow J_{\text{L2}}(\text{ridge}) \quad \Omega(w) = \mathbf{w}^T \mathbf{w} : \text{weight decay}$$

$$J_{\text{L1}}(\text{Lasso}) \quad \Omega(w) = \|w\|_1 : \text{가중치 초기 제한.}$$

Early Stopping, Data Augmentation, Ensemble, ... 일부 불필요 가중치 0으로.

Dropout: 훈련 중 각 계층에 noise 주입 \Rightarrow 각 계층 활성화 값 일부를 0으로! (즉, 확률 P)

신경망이 특정 hidden unit이 아니라 다른 유연하게 학습하는 것 (co-adaptation) 방지.

기대값 변경 X \Rightarrow 0 설정하면 나머지 매개변수 (1-P)로 나누기. rescaling

$$h' = \begin{cases} 0 & (\text{P 확률}) \\ \frac{h}{1-P} & (\text{그 외}) \end{cases} \Rightarrow \mathbb{E}[h'] = h \text{ 유지.}$$

테스트 시, dropout 비활성화

$$\text{Sigmoid} = \frac{1}{1+e^{-x}} \in (0, 1), \quad \text{Sigmoid}' = \text{Sigmoid}(1 - \text{Sigmoid})$$

Vanishing gradient: Sigmoid 도함수가 x=0에서 최대값 0.25 도달. x가 0에서 멀어질수록 도함수값이 0에 수렴. 신경망 구조설계를 기울기 항 여러번 더함 \Rightarrow 전체 sum 수렴. 매개변수 업데이트 마이너스 \Rightarrow 학습 불가.

$$\begin{aligned} j \text{ 계층 활성화 값 } z_j &= 1 \Rightarrow \text{입력이 매우 큰 양수. } \frac{dz_j}{dh_j} \approx 0. \\ &\Rightarrow \frac{dl}{dh_j} 역시 0으로 수렴. 학습 정체. \end{aligned}$$

Sigmoid 단점: Vanishing gradients. \Rightarrow ReLU 단점. Simple, no vanishing

optimizing hardness. $\quad \text{ReLU}' = 1 \text{ when input} > 0$

$$[=0 \text{ when input} \leq 0]$$

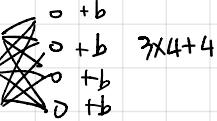
\hookrightarrow dying ReLU.

trainable parameters

$$\text{fully connected (MLP)} \Rightarrow (D_i \times D) + D \quad \left(\begin{array}{l} D_i = \text{입력개수} \\ D = \text{출력개수} \end{array} \right)$$

$$+ (D \times D_o) + D_o \quad \left(\begin{array}{l} D_o = \text{출력개수} \end{array} \right)$$

50개 입력 30개 출력개수 (가 총 3개)
 $\Rightarrow (50 \times 30) + 30 + (30 \times 3) + 3 = 2213.$



$$\text{CNN} \Rightarrow C_i = \text{입력채널}, C_o = \text{출력채널}, k_h = \text{커널 높이}, k_w = \text{커널 너비}$$

$$\text{커널 개수} = C_i \times C_o \times k_h \times k_w$$

$$\text{전체 개수} = C_o$$

$$\text{총 개수} = C_i \times C_o \times k_h \times k_w + C_o$$

마지막 단계 수 폭증 방지

locality: 각 출력이 국소지역에 국한.

마지막 단계: 동일 커널을 모든 장소에서 걸쳐 사용.

$$* \text{Convolution output size} = \left(\frac{H+2P-F}{S} + 1 \right) \times \left(\frac{W+2P-F}{S} + 1 \right) \times \# \text{Filter}$$

H: input height P: padding size F: filter size S: stride

Convolution 입력 텐서와 커널 텐서 결합해 출력 텐서 생성. (sliding window)
 $\xrightarrow{\text{(multi)}}$
 translation invariance, equivariance

이동이 각각 위치에 끊임없이 신호 X. 입력 이동시 출력도 이동.

locality

Pooling: 해상도 축소 (down sampling) \Rightarrow 계산 복잡도 감소.

전역적 표본 학습 가능 \Rightarrow 풀링 layer가 global representation을 인식하기.

pooling window sliding \Rightarrow max/avg pooling + stride + padding

출력 채널 수 = 입력 채널 수 (\because 입력 채널 간 합친다)

VGGNet: 개별 레이어 대신 반복되는 패턴 사용해 구성. 깊고 짧은 network \rightarrow !

block \Rightarrow convolution layer + padding \Rightarrow 해상도 유지. padding = 1, 3x3 kernel
 ReLU \Rightarrow 비선형성.

pooling 계층 \Rightarrow stride = 2, 2x2 max pooling 사용해 해상도 $\frac{1}{2}$

FL \Rightarrow 블록 통과 시 해상도 절반. 차를 수 (깊이) 증가.

마지막 conv 통과 시 출력 flatten. fully connected layer 통해 최종 예측.

ResNet

네트워크 깊이 증가 \Rightarrow 훈련 오류 증가 (degradation problem)

Sol) residual block (connection)

일반적: $f(x)$ 라는 미명 적용하는

res block: $g(x) = f(x) - x$ 잡여 미명 학습. \Rightarrow output: $y = f(x) + x$

residual connection

FL: VGG 3x3 conv layer.

res block \Rightarrow 3x3 conv layer 2x2 + Batch Norm + ReLU.

활성화 값 전파 증가하게 보장. 순전파 안정화

\hookrightarrow 출력이 입력 X와 동일 모양!

1x1 conv layer로 입력 X 모양 바꾸는.