Machine Learning

3장 최적화(optimization)와 딥러닝 모형진단

고려대학교 통계학과
박유성

Contents

01 출력층과 손실함수

- 딥러닝의 설계는 입력특성변수 x의 자료형태에 따라 MLP, CNN, RNN, 또는 이들 신경망의 조합을 결정할 뿐만 아니라 출력의 형태를 결정하는 출력층의 결정도 포함하고 있다
- 연구의 목적이 분류(classification)이면 2개의 그룹으로 분류할 것인지, 3개 이상의 그룹으로 분류할 것인지를 구분해야 하며 연구의 목적이 회귀(regression)이면 출력이 하나의 값인지 두 개 이상의 값인지를 구분해야 한다
- 예를 들어, 특정 사안에 대한 트윗이 찬성인지 반대인지를 구분한다든가 손으로 쓴 우편번호가 0~9의 숫자 중 어느 숫자인지를 구분하는 3개 이상의 클래스를 구분
- 회귀의 경우, 최근 10일 동안 관측된 주식가격을 통해 내일 주식가격을 예측한다든가 또는 내일을 포함하여 향후 3일 동안의 주식가격을 예측

01 출력층과 손실함수

- 목적변수의 실제 참값과 딥러닝에 의해 예측된 값이 최대한 근접해야 딥러닝 모형은 의미를 가지게 된다
- 실제 참값과 딥러닝에 의해 예측된 값의 거리를 측정하는 측도가 필요하며 이를 손실함수(loss function)라고 한다

01 출력층과 손실함수

손실함수(회귀)

$$SSE=\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- 여기에서 n은 표본의 크기 또는 batch size
- y_i는 목적변수, $\hat{y}_i$는 y_i의 예측치로 $E(y_i|x_i)$임
- y_i가 두 개 이상이면, $y_{i1}, y_{i2}, \cdots, y_{ik}$를 k개의 목적변수라고 하고 $\hat{y}_{i1}, \hat{y}_{i2}, \cdots, \hat{y}_{ik}$를 예측치라고 하면

$$SSE=\sum_{i=1}^{n}\sum_{j=1}^{k}(y_{ij} - \hat{y}_{ij})^2$$

- SSE내에 있는 $\hat{y}_i$는 딥러닝 모형에 있는 수많은 모수의 함수이다.
- 최종 출력층의 활성함수는 항등함수(선형)이다.

01 출력층과 손실함수

손실함수(회귀)

- 손실함수 후보: $\sum_{i=1}^{n}|y_i - \hat{y}_i|$ 또는 $\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right|$
- 손실함수를 최소화하는데 사용하는 역전파(backpropagation)가 작동하기 위한 전제조건은 손실함수가 모든 값에서 미분 가능이어야 한다
- $\text{MSE}=\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, $\text{MAE}=\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, $\text{MAPE}=\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right|$

  ⇒모형의 정밀도(accuracy) 측도로 사용하며 과대적합, 과소적합 등을 판단하거나 서로 다른 딥러닝모형을 비교하는데 유용하게 사용된다

01 출력층과 손실함수

손실함수(범주형)

- 목적: 두 개의 클래스분류. 이를 이항분류(binary classification)라고 하며
- 목적변수 y_i= 0 또는 1
- x_i가 입력⇒최종 은닉층의 출력, $h_1, h_2, \cdots, h_m$⇒ 출력층, $z_i = \sum_{j=1}^{m} w_j h_j + b$

$$sigmoid(z_i) = \frac{1}{1+e^{-z_i}} = \hat{y}_i \ (0\sim1) \Rightarrow \ P(y_i = 1|x_i)\text{의 추정치}$$

- $p_i = P(y_i = 1|x_i)$표기하고 n개의 표본이 서로간에 독립이며 y_i가 Bernoulli 확률변수이면, 로그우도함수는

$$\sum_{i=1}^{n}[y_i log p_i + (1-y_i)\log(1-p_i)]$$

이므로 우도함수를 최대로 하는 p_i

01 출력층과 손실함수

손실함수(범주형)

- ***손실함수는***

$$\text{loss} = -\sum_{i=1}^{n}[y_i log p_i + (1 - y_i)\log(1 - p_i)]$$

를 최소화하는 p_i.

- 이 손실함수를 binary cross entropy라고 함.
- p_i의 추정치는 $\hat{y}_i$이다.
- 이 의미는 $y_i = 1$일 때 $\hat{y}_i \approx 1$이 되고 $y_i = 0$일 때 $\hat{y}_i \approx 0$ 이 되도록 딥러닝 모수를 추정한다는 것.

01 출력층과 손실함수

손실함수(범주형)

- 목적: 두 개 이상의 c개 클래스분류. 이를 다항분류라고 하며
- 목적변수 y_i는 one-hot encoding

  eg. 세 개의 클래스: y_i=(1,0,0), (0,1,0), 또는 (0,0,1)

- x_i가 입력⇒최종 은닉층의 출력, $h_1, h_2, \cdots, h_m$⇒ 출력층, $z_{ij} = \sum_{k=1}^{m} w_{kj} h_k + b_j$, j $= 1 \cdots, c$

  $sigmoid(z_{ij}) = \frac{1}{1+e^{-z_{ij}}} = \hat{y}_{ij}$ (0~1): $P(y_{ij} = 1|x_i)$의 추정치

- 위 식은 i 번째 표본의 소속이 j인지 아닌지를 구별하는 이항분류임
- 그러므로 이항분류를 c번 반복하여 가장 큰 $\hat{y}_{ij}$를 가진 클래스로 분류(One-versus Rest)

01 출력층과 손실함수

손실함수(범주형)

One-versus-Rest (OvR) 다범주 분류의 손실함수

$$-\sum_{j=1}^{c}\sum_{i=1}^{n}[y_{ij}log\,\hat{y}_{ij}+(1-y_{ij}\log(1-\hat{y}_{ij})]$$

- 그러나 $\sum_{j=1}^{c}\hat{y}_{ij}=1$을 보장하지 않음.
- Softmax fuction:

$$\hat{y}_{ij}=\sigma(z_{ij})=\frac{\exp(z_{ij})}{\sum_{j=1}^{c}\exp(z_{ij})}$$

여기에서 $\hat{y}_{ij}$는 i번째 표본이 j번째 클래스에 속할 확률 p_{ij}의 추정치이며 $\sum_{j=1}^{c}\hat{y}_{ij}=1$를 만족함.

- 표본이 서로간에 독립인 다항분포를 가정하면 로그우도함수는

$$\sum_{i=1}^{n}\sum_{j=1}^{c}y_{ij}logp_{ij}$$

01 출력층과 손실함수

손실함수(범주형)

- 그러므로 softmax 함수를 이용한 손실함수는

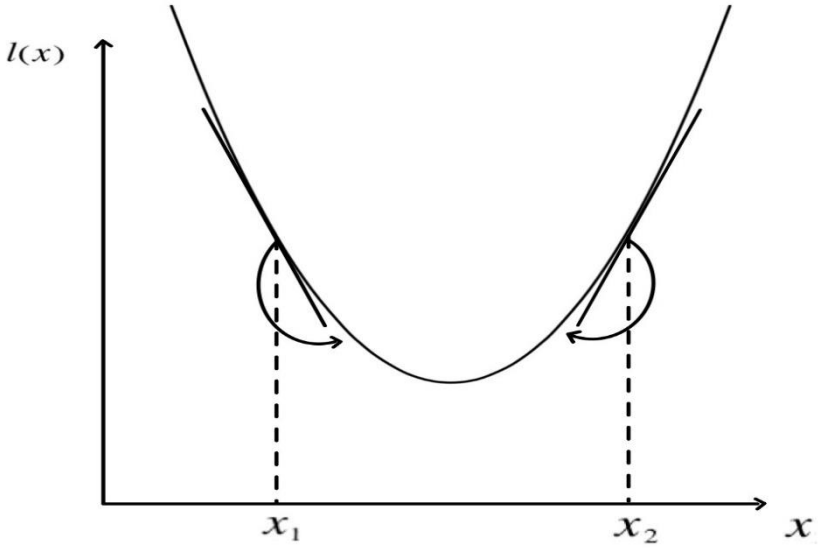
$$-\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij} log\, \hat{y}_{ij}$$

가 된다. 이 손실함수를 categorical cross entropy라고 한다.

목적	출력층의 활성함수	손실함수	정밀도
회귀	항등	SSE	MSE, MAE, MAPE
이항분류	sigmoid	binary cross entropy	accuracy, precision, recall
다항분류	softmax	binary cross entropy categorical cross entropy	accuray, precision, recall

- accuracy는 정확하게 분류한 비율을 말하고, precision은 예측한 클래스가 맞은 비율, recall은 관심있는 클래스가 제대로 예측된 비율

02 역전파(backpropagation)



- 손실함수 $l(x)$는 2차함수
- $x = x_1$에서 $l(x)$의 미분값은 양수, $x = x_2$에서 $l(x)$의 미분값은 음수

$$\Rightarrow x \leftarrow x - \eta \frac{\partial l(x)}{\partial x},$$

여기에서 $\eta > 0$ 이며 이를 학습률(learning rate)이라고 한다.

02 역전파(backpropagation)

- $x - \eta \frac{\partial l(x)}{\partial x}$에서 $l(x)$의 미분값을 계산한 후, $x \leftarrow x - \eta \frac{\partial l(x)}{\partial x}$으로 x를 update하여 이러한 update를 $\frac{\partial l(x)}{\partial x} = 0$이 될 때까지 x를 update하면 $l(x)$를 최소화하는 x를 구하게 됨.
- 학습률 η는 매우 작은 값임
- 아주 작은 학습률을 곱해서 조금씩 최신화하는 이유는
- 딥러닝에서 다루는 손실함수가 간단한 2차함수가 아닌 아주 복잡한 함수이며
- 특히, 국소최소값(local minimum)과 안장점(saddle point)에 빠지는 것을 방지하기 위함이 첫 번째 목적이고,
- 두 번째 목적은 곧 논의할 과대적합(overfitting)을 방지하기 위함이다.

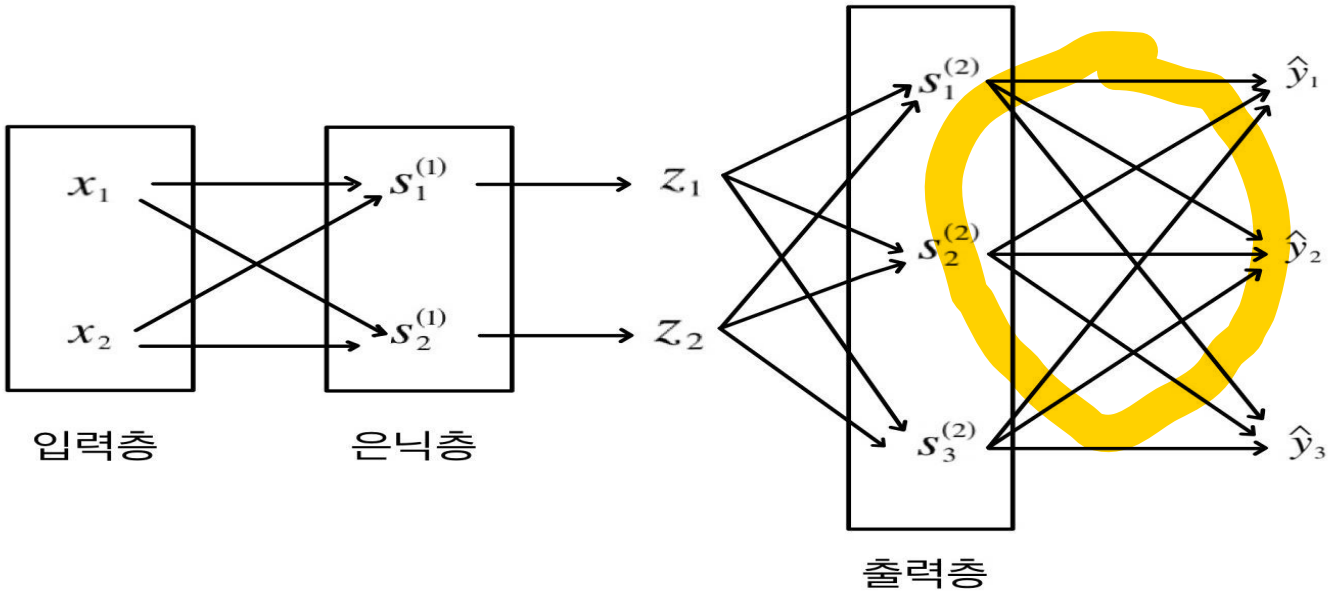
02 역전파(backpropagation)

- MLP, CNN, RNN 모형은 수많은 모수를 가지고 있다.
- 이 모수들은 앞에 정의된 손실함수를 최소화하는 과정에서 동시에 최신화하게 된다.
- 이는 모든 모수에 대한 손실함수의 미분값이 필요하다는 것을 의미한다.
- 그러나 상위 은닉층에 있는 모수는 하위 은닉층의 함수로 구성되어 있으므로 소위 역전파(backpropagation)를 이용하여 딥러닝 모형에 있는 모든 모수의 미분값을 구하게 된다.

02 역전파(backpropagation)

- 손실함수를 최소화하는 모수를 구하는 방법
- 딥러닝내 모수의 초기치를 임의로 부여한 후, 입력층→은닉층 → 출력층 순으로 모수값을 대입하여 손실함수를 계산
- 손실함수의 미분값을 출력층 → 은닉층 → 입력층으로, 즉 역순으로(이를 역전파라고 한다) 계산하여 $x - \eta \frac{\partial l(x)}{\partial x}$에 의해 모수를 최신화한다.
- 이 과정을 과대적합 문제가 발생하지 않을 때까지 반복하여 최적의 모수를 산출하게 된다

02 역전파(backpropagation)



- 입력변수가 2개, 2개의 노드를 가진 은닉층, 그리고 3개의 클래스를 분류하기 위한 3개의 노드를 가진 출력층으로 구성된 MLP 모형
- 은닉층의 활성함수는 sigmoid를, 출력층의 활성함수는 softmax를 사용
- 그러므로 손실함수는 categorical cross entropy
- 첨자의 복잡성을 피하기 위해 표본의 개수는 1개라고 가정한다.

02 역전파(backpropagation)

- $s_1^{(1)} = w_1 x_1 + w_2 x_2 + b_1,\ s_2^{(1)} = w_3 x_1 + w_4 x_2 + b_2$

- $z_1 = \frac{1}{1+e^{-s_1^{(1)}}},\ z_2 = \frac{1}{1+e^{-s_2^{(1)}}}$

- $s_1^{(2)} = \eta_1 z_1 + \eta_2 z_2 + b_3,\ s_2^{(2)} = \eta_3 z_1 + \eta_4 z_2 + b_4,\ s_3^{(2)} = \eta_5 z_1 + \eta_6 z_2 + b_5$

- $P(y_1 = 1)$의 추정치를 $\hat{y}_1$, $P(y_2 = 1)$의 추정치를 $\hat{y}_2$, $P(y_3 = 1)$의 추정치를 $\hat{y}_3$ 이라고 할 때

- $\hat{y}_1 = \frac{\exp(s_1^{(2)})}{\sum_{j=1}^{3} \exp(s_j^{(2)})},\ \hat{y}_2 = \frac{\exp(s_2^{(2)})}{\sum_{j=1}^{3} \exp(s_j^{(2)})},\ \hat{y}_3 = \frac{\exp(s_3^{(2)})}{\sum_{j=1}^{3} \exp(s_j^{(2)})}$

- 손실함수 $\ell = -\sum_{i=1}^{3} y_i \, log \hat{y}_i$

02 역전파(backpropagation)

- $\frac{\partial \ell}{\partial \eta_4}$를 구하기 위해, $\eta_4 \rightarrow s_2^{(2)} \rightarrow \hat{y}_1, \hat{y}_2, \hat{y}_3$ 즉, η_4는 $s_2^{(2)}$에 영향을 주고 $s_2^{(2)}$는 $\hat{y}_1, \hat{y}_2, \hat{y}_3$에 영향을 주므로 chain rule에 의해

$$\frac{\partial \ell}{\partial \eta_4} = \frac{\partial \ell}{\partial \hat{y}_1}\frac{\partial \hat{y}_1}{\partial s_2^{(2)}}\frac{\partial s_2^{(2)}}{\partial \eta_4} + \frac{\partial \ell}{\partial \hat{y}_2}\frac{\partial \hat{y}_2}{\partial s_2^{(2)}}\frac{\partial s_2^{(2)}}{\partial \eta_4} + \frac{\partial \ell}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_2^{(2)}}\frac{\partial s_2^{(2)}}{\partial \eta_4}$$

$$= y_1\hat{y}_2 z_2 + y_2(1-\hat{y}_2)z_2 + y_3\hat{y}_2 z_2$$

$$= \hat{y}_2(y_1 + y_2 + y_3)z_2 - y_2 z_2 = (\hat{y}_2 - y_2)\, z_2 \quad (y_1 + y_2 + y_3 = 1)$$

- $\frac{\partial \ell}{\partial w_1}$을 위해 $w_1 \rightarrow s_1^{(1)} \rightarrow z_1 \rightarrow s_1^{(2)}, s_2^{(2)}, s_3^{(2)} \rightarrow \hat{y}_1, \hat{y}_2, \hat{y}_3$이므로

$$\frac{\partial \ell}{\partial w_1} = \sum_{j=1}^{3}\sum_{k=1}^{3}\frac{\partial \ell}{\partial \hat{y}_k}\frac{\partial \hat{y}_k}{\partial s_j^{(2)}}\frac{\partial s_j^{(2)}}{\partial z_1}\frac{\partial z_1}{\partial s_1^{(1)}}\frac{\partial s_1^{(1)}}{\partial w_1} = \sum_{j=1}^{3}(\hat{y}_j - y_j)\eta_{2j-1} z_1(1-z_1)x_1$$

03 최적화 알고리즘

- l_i를 i번째 표본의 손실이라고 정의할 때 총 표본 n에 대한 손실은

$$l = \sum_{i=1}^{n} l_i$$

- 기울기 하강법(gradient descent): 손실함수가 아래로 볼록, no local minimum일 때

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{\partial l}{\partial \theta}\Big|_{\theta=\theta^{(t)}}$$

- $n = 1$이면 확률적 기울기 하강법(stochastic gradient descent): local minimum에 강점, too noisy
- Batch: n을 k개로 나눈 단위 → batch stochastic gradient descent

03 최적화 알고리즘

- 학습률 η의 결정에 따라 최적화 알고리즘이 개발됨
- 전체 데이터에 적용되는 2개의 최적화 알고리즘

1. Adagradoptimizer

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \eta \frac{1}{\sqrt{\sum_{r=1}^{t}(\theta_i^{(r)})^2 + \varepsilon}} \frac{\partial l^{(t)}}{\partial \theta_i}$$

여기에서 $\frac{\partial l^{(t)}}{\partial \theta_i} = \frac{\partial l}{\partial \theta_i}\Big|_{\theta_i=\theta_i^{(t)}}$이고 **아주 작은 값** $\varepsilon > 0$ 보통 10^{-8}

⇒ 학습률이 반복 누적제곱합에 반비례 θ_i가 클수록 최신화를 작게 하고 작을수록 최신화를 많이 함.

03 최적화 알고리즘

2. Rprop 최적화(Resilient backpropagation optimization)

전체 데이터를 이용한 최적화 알고리즘 중 가장 우수함.

Δ_{min}, $\Delta_{max,}$ 그리고 $\delta_i^t = sign(\frac{\partial l^{(t)}}{\partial \theta_i} \times \frac{\partial l^{(t-1)}}{\partial \theta_i})$를 정의한다.

$$\Delta_i^{(t+1)} = \begin{cases} \min\left(1.2\Delta_i^{(t)}, \Delta_{max}\right) & \text{만약 } \delta_i^t > 0 \\ \max\left(0.5\Delta_i^t, \Delta_{min}\right) & \text{만약 } \delta_i^t < 0 \\ \Delta_i^{(t)} & \text{만약 } \delta_i^t = 0 \end{cases}$$

$$\theta_i^{(t+1)} = \theta_i^{(t)} - sign(\frac{\partial l^{(t)}}{\partial \theta_i})\Delta_i^{(t+1)}$$

03 최적화 알고리즘

2. Rprop 최적화(Resilient backpropagation optimization)

◆ $\delta_i^t > 0$ 이면 손실의 미분이 같은 방향이므로 수렴속도를 높이고

◆ $\delta_i^t < 0$이면 손실함수의 미분 값의 부호가 바뀌어서 최소값을 지났다는 의미이므로 최소값 주변에서의 진동을 방지하고 최소값으로 접근하도록 수렴속도를 줄이게 된다.

03 최적화 알고리즘(배치 확률화 기울기 하강법)

- RMSprop (Root mean squared propagation)

$$g_i^{(t)} = \alpha g_i^{(t-1)} + (1-\alpha)\left(\frac{\partial l^{(t)}}{\partial \theta_i}\right)^2$$

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \frac{\eta}{\sqrt{g_i^{(t)} + \varepsilon}} \frac{\partial l^{(t)}}{\partial \theta_i}$$

여기에서 일반적으로 $\alpha = 0.999, \varepsilon = 10^{-8}$

$$\Rightarrow g_i^{(t)} = \sum_{r=0}^{t} \alpha^r (1-\alpha)\left(\frac{\partial l^{(t-r)}}{\partial \theta_i}\right)^2 + \alpha^t g_i^{(0)}$$

◆ 가장 최근의 손실함수변동에 가장 큰 가중치를 주고 지수적 감소형태의 가중치

◆ 직전 최신화에서 손실함수의 변동에 많이 기여한 모수의 최신화는 줄이고 적게 기여한 모수의 최신화는 증가시킴.

03 최적화 알고리즘(배치 확률화 기울기 하강법)

- Adaptive delta (AdaDelta)

AdaDelta는 RMSprop에 모수의 변화량 $\Delta\theta_i^{(t)} = \theta_i^{(t)} - \theta_i^{(t-1)}$를 추가함

$$g_i^{(t)} = \alpha g_i^{(t-1)} + (1-\alpha)\left(\frac{\partial l^{(t)}}{\partial \theta_i}\right)^2$$

$$h_i^{(t)} = \beta h_i^{(t-1)} + (1-\beta)(\Delta\theta_i^{(t)})^2$$

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \frac{\sqrt{h_i^{(t)}+\varepsilon}}{\sqrt{g_i^{(t)}+\varepsilon}}\frac{\partial l^{(t)}}{\partial \theta_i}$$

$\Rightarrow \theta_i$의 변동이 크게 변했다는 것은 θ_i가 손실함수 감소에 기여가 크다는 의미이므로 θ_i의 변동에 비례하여 최신화 강도 증가.

03 최적화 알고리즘(배치 확률화 기울기 하강법)

- Adaptive moment (Adam)

$$m_i^{(t)} = \beta_1 m_i^{(t-1)} + (1-\beta_1)\frac{\partial l^{(t)}}{\partial \theta_i}$$

$$g_i^{(t)} = \beta_2 g_i^{(t-1)} + (1-\beta_2)\left(\frac{\partial l^{(t)}}{\partial \theta_i}\right)^2$$

지수평균: $\bar{m}_i^{(t)} = \frac{m_i^{(t)}}{1-\beta_1^t}$, $\bar{g}_i^{(t)} = \frac{g_i^{(t)}}{1-\beta_2^t}$

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \frac{\eta}{\sqrt{\bar{g}_i^{(t)}+\varepsilon}}\bar{m}_i^{(t)}$$

03 최적화 알고리즘(배치 확률화 기울기 하강법)

- Momentum

$$v_i^{(t+1)} = \alpha v_i^{(t)} - \eta \frac{\partial l^{(t)}}{\partial \theta_i}\Big|_{\theta_i=\theta_i^{(t)}}$$

$$\theta_i^{(t+1)} = \theta_i^{(t)} + v_i^{(t+1)}$$

$\Rightarrow \frac{\partial l^{(t)}}{\partial \theta_i}$이 계속해서 동일한 부호를 가지면 $v_i^{(t+1)}$의 절대값은 증가.

$\Rightarrow \theta_i^{(t+1)}$의 변화가 커짐. 반대로 $\frac{\partial l^{(t)}}{\partial \theta_i}$의 부호가 교차하면 $\frac{\partial l^{(t)}}{\partial \theta_i}$의 절대값이 작아짐.

$\Rightarrow \theta_i^{(t+1)}$의 변화가 작아짐.

03 최적화 알고리즘(배치 확률화 기울기 하강법)

- Nesterov

$$\tilde{\theta}_i^{(t)} = \theta_i^{(t)} + \alpha v_i^{(t)}$$

$$v_i^{(t+1)} = \alpha v_i^{(t)} - \eta \frac{\partial l^{(t)}}{\partial \theta_i}\Big|_{\theta_i=\tilde{\theta}_i^{(t)}}$$

$$\theta_i^{(t+1)} = \theta_i^{(t)} + v_i^{(t+1)}$$

$\Rightarrow \frac{\partial l^{(t)}}{\partial \theta_i}$의 평가값을 $\tilde{\theta}_i^{(t)}$으로 변경

04 딥러닝모형의 진단과 일반화(generalization)

- 딥러닝을 포함하여 머신러닝의 궁극적 목표는 추정에 전혀 사용한 적이 없는 (never-seen-before) 데이터에도 성능이 좋은, 모형의 일반화 (generalization)에 있다
- 이를 위해 딥러닝에서는 모수를 추정하기 이전에 미리 전체 데이터에서 따로 떼어 놓고 이 데이터를 시험데이터(testing data)라고 한다. 나머지 데이터로 딥러닝 모형을 설정하고 모수를 추정한 후, 시험데이터에 이 학습된 모형을 적용하여 모형의 일반화를 점검하게 된다.
- 딥러닝 모형을 설정하고 모수를 추정하는 데이터는 학습데이터(training data)와 검증데이터(validation data)로 구성되어 있다.

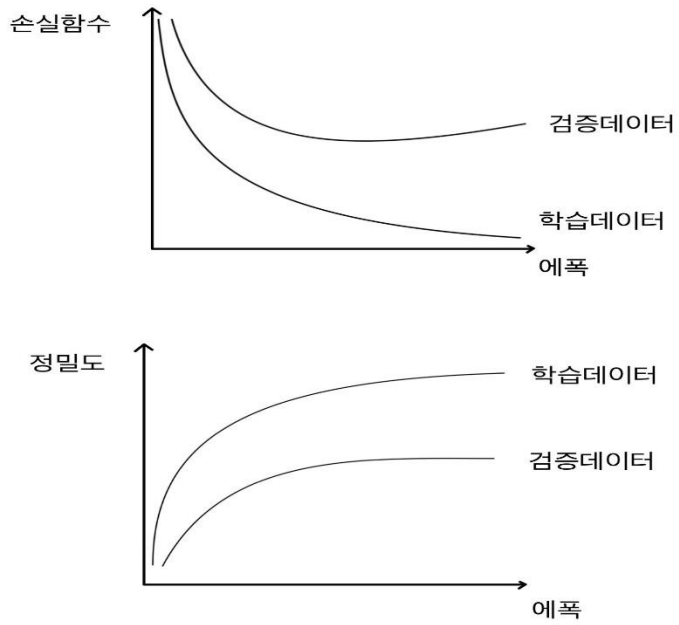
04 딥러닝모형의 진단과 일반화(generalization)

- 학습데이터는 MLP, CNN, RNN 등 딥러닝 모형에 포함되어 있는 모수를 추정하고 손실함수값과 모형의 정밀도를 계산하는데 사용한다.
- 검증데이터는 딥러닝 모형의 설계를 위한 은닉층의 수, 딥러닝 아키텍쳐, 최적화 알고리즘의 선택, 과대적합문제를 해결하기 위한 규제화(regularization), dropout, 배치표준화(batch normalization) 사용여부 등 수 많은 모형 튜닝(tuning)에 사용되는 데이터이다.
- 그러므로 검증데이터는 초모수(hyperparameters)를 조절하여 학습데이터에서 학습된 모형의 성능을 향상시키는데 사용된다.
- 검증데이터는 딥러닝 모형을 설정하는데 정보를 제공하므로 모형 설정에 전혀 정보를 제공하지 않는 시험데이터와 구별된다.

04 딥러닝모형의 진단과 일반화(generalization)

- 검증데이터가 딥러닝 모형의 튜닝 임무를 마치면, 딥러닝 모형의 최종적인 모수 추정은 학습데이터+검증데이터에 적용하여 구해야한다
- 데이터가 많을수록 좀 더 정확한 모수추정치를 구한다는 통계적인 관점 때문이다.

04 딥러닝모형의 진단과 일반화(generalization)



- 위 그림은 과대적합의 예를 보여주고 있다.
- 일반적으로 평균모형보다 높은 목표정밀도를 미리 설정해 놓고 학습데이터의 정밀도가 이에 도달하지 못하면 모형이 과소적합(under-fitting)되었다고 말한다.
- 과소적합되면 딥러닝의 은닉층을 늘리거나 은닉층의 노드수를 증가시킨다.
- 그래도 문제가 해결되지 않으면 딥러닝 모형의 설계를 변경하여야 한다.

04 딥러닝모형의 진단과 일반화(generalization)

- 학습데이터의 목표 정밀도는 충족되었으나 검증데이터의 정밀도가 그림과 같이 학습데이터의 정밀도와 크게 차이가 나면,
- 딥러닝 모형에 일반적으로 일어나는 현상이면서 가장 큰 문제인 과대적합(over-fitting)문제가 발생했다고 진단한다.
- 검증데이터에 과대적합문제가 발생하면 시험데이터(testing data)에도 동일한 과대적합문제가 발생한다.
- 그러므로 과대적합이 발생한 딥러닝 모형은 일반화(generalization)에 실패했다고 말한다.

04 딥러닝모형의 진단과 일반화(generalization)

과대적합문제를 해결하는 방법

- 가장 좋은 방법은 데이터를 더 수집하여 학습데이터의 크기를 늘리는 것

좀 더 많은 데이터는 좀 더 많은 불확실성을 모형에 부과하게 되어 모형의 성능을 통계적으로 향상시키기 때문이다.

- 그러나 대부분의 경우, 데이터의 크기를 늘리는 것은 쉽지 않다. 대체 방법으로 딥러닝의 은닉층을 줄이거나 은닉층의 노드수를 줄이는 것

- 노드수를 줄이는 방법 중에 하나는 모수에 대한 규제화(regularization)이다.

L_1 regularization : 손실함수=기존의 손실함수+$\lambda_1 \sum |\theta_i|$

L_2 regularization : 손실함수=기존의 손실함수+$\lambda_2 \sum \theta_i^2$

04 딥러닝모형의 진단과 일반화(generalization)

과대적합문제를 해결하는 방법

- 출력층 직전 은닉층의 노드수를 줄이는 것

  통계적 관점에서 출력층 직전 은닉층 노드수가 설명변수의 수가 됨

- Dropout

- dropout는 은닉층의 출력값 중 일부를 임의로 0으로 놓는 것을 말한다.
- 학습(learning)과정에서 은닉층의 출력값 중 일부를 0으로 한다는 것은 다음 층(은닉층 또는 출력층)의 입력의 일부가 0으로 입력된다는 것이다.
- 역전파를 이용한 모수의 최신화가 안되는 효과를 가지게 된다. 이는 모형의 불확실성을 증가시켜 과대적합문제를 해결하는 데에 유용하다.

04 딥러닝모형의 진단과 일반화(generalization)

과대적합문제를 해결하는 방법

- 활성함수에 아주 작은 값 또는 아주 큰 값이 입력되면 활성함수의 미분값은 거의 0에 가까운 값을 가지게 된다.
- 예를 들어, sigmoid함수를 고려하면 일정수준 이하 또는 일정수준 이상의 값이 sigmoid 함수에 입력되면 각각 거의 0 또는 1의 값을 출력하게 되므로 이러한 작은 값 또는 큰 값에서의 sigmoid 함수의 미분 값은 0에 가깝게 된다.
- 그러므로 선형결합의 값이 아주 작은 값 또는 아주 큰 값이 연달아 입력되면 선형결합 가중치(모수)가 역전파로 최신화를 하지 못해, 수렴속도가 아주 느리게 되어 결과적으로 최적화의 실패요인이 된다. 배치정규화는 이러한 문제를 해결하는 우수한 도구이다.

04 딥러닝모형의 진단과 일반화(generalization)

과대적합문제를 해결하는 방법

배치정규화

- 배치의 크기 m, 특성변수 x를 표준화

$$\tilde{x}_i = \frac{x_i - batch\ mean\ of\ x_i}{batch\ standard\ deviation\ of\ x_i},\ i = 1, \cdots, m$$

- $x_i^* = \alpha\tilde{x}_i + \beta$
- 여기에서 α와 β는 딥러닝 모수임
- 배치정규화는 각 은닉층내에서 활성화 함수를 적용하기 직전에 실행되어야 한다.
- 이외에도 instance normalization, layer normalization, group normalization이 있음.

Q & A