Machine Learning

- formalization

f (target) : $\mathcal{X}$ (input) → y (output) , g (hypothesis) : $\mathcal{X} \to y$ ⇒ 목적은 $g \approx f$

- perceptron $h(x) = \text{sign}(\sum w_i x_i - \text{threshold})$ ⇒ all the hypotheses $h \in \mathcal{H}$

- PLA (perceptron learning algorithm)

$h(x) = \text{sign}(w^T x_n)$

* objective : g에 가까운 최적의 w를 찾는 것.

if $\text{sign}(w^T x_n) \neq y_n$,

* assumption : 선형분리 가능.

$w(t+1) = w(t) + y_n x_n$

I. Feasibility of Learning

⇒ 목적함수 f = unknown. ⇒ (1) 제한적인 데이터에서 f를 어떻게 결정하나?

(1) : In training data D. D는 제공된 데이터 밖의 데이터에서 무엇인가 알려주는가?

↳ A·H와 outside D는 상관이 없음. learning이 불가. ⇒ (2) f를 학습하는 것은 불가능.

(2) : rescue plan , 결정론적이 아닌 확률론적으로 접근.

* Hoeffding Inequality : $\mathbb{P}[|\hat{\mu} - \mu| > \epsilon] \le 2e^{-\epsilon^2 N}$

(probably approximately correct)

⇒ 샘플 수 N이 커지면, 우리가 허용한 ϵ보다 $\mu \cdot \hat{\mu}$의 차이가 작아짐 즉, "$\mu = \hat{\mu}$" (PAC)

⇒ with large N, f와 매우 근접한 h를 추론할 수 있다. ⇒ A는 best h를 찾는 과정.

(3) : μ : true error rate. $\hat{\mu}$: sample error rate. 일반화에서 $\hat{\mu} \approx 0$을 어떻게 아는가?

⇒ $\mathbb{P}[|E_{in}(h) - E_{out}(h)| > \epsilon] < 2e^{-2\epsilon^2 N}$

→ $E_{in}(h)$: in-sample error $\hat{\mu}$. $= \frac{1}{N}\sum [\![h(x_n) \neq f(x_n)]\!]$, $[\![\text{true}]\!] = 1$

$E_{out}(h)$: μ $= \mathbb{P}[h(x) \neq f(x)]$ ⇒ E_{in} : random known, E_{out} : fix unknown.

⇒ Hoeffding inequality는 다중가설에 적용 안됨. 따라서 union bound 이용.

$$\mathbb{P}[|E_{in}(g) - E_{out}(g)| > \epsilon] \le \sum^{M} \mathbb{P}|E_{in}(h_m) - E_{out}(h_m)| > \epsilon \le 2Me^{-2\epsilon^2 N}$$

* 학습의 주된 목표 : 목적함수 f와 유사한 가설 g , $E_{out}(g) \approx 0$ 을 찾는 것.

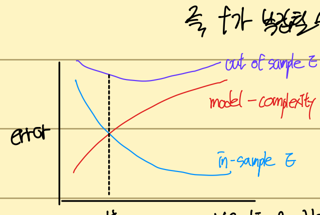
그러나 결정론적인 방법으로는 어려워 확률론적인 방법으로 $E_{out}(g) \approx E_{in}(g)$.

(4) : { $E_{out} \approx E_{in}$ 가 확실한가? Hoeffding inequality로 검증가능.
$E_{in}(g)$를 충분히 작게 만들 수 있는가? 실제 데이터를 학습시키고 확인.

(5) : 가설 H의 복잡성 — M이 커지면 일반화가 저하. E_{in}을 위해서는 H가 복잡해야함 (trade-off)

정답 f의 복잡성 — f가 복잡하면 $E_{in}(g)$가 저하. → H를 복잡하게 만들면 일반화가 어려워짐.

즉, f가 복잡할수록 학습하기 어렵다. 그러나 현실문제는 그리 복잡하지 않음.



VC dim ≈ H의 복잡성.

model complexity ⇑ E_{in} ⇓ , $E_{out} - E_{in}$ ⇑ (trade-off)

- Error & Noise

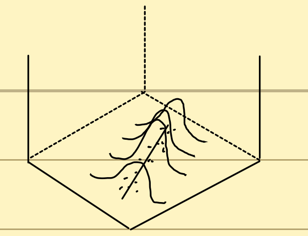
* Error ; $h \approx f$, outcome of learning * Noise ; not uniquely, f의 가변적인 산출.

⇒ Error = $E(h, f)$ $= e(h(x), f(x))$, $E_{in}(h) = \frac{1}{N}\sum e(h(x_n), f(x_n))$, $E_{out}(h) = \mathbb{E}_x[e(h(x), f(x))]$

		f: P	f: N
h	P	TP	FP
	N	FN	TN

FP : 제1종 오류, Yes라고 했지만 No인 경우
FN : 제2종 오류, No라고 했지만 Yes인 경우

} error measure은 사용자에 맞게 cost를 줘야함.



- Target 'distribution'

$y = f(x)$ 를 대신해 $y \sim P(y|x)$, input x에 따른 랜덤변수 output y로 설정.
(결정론적인)

II. Linear Classification · Regression

i) classification

PLA : $h(x) = \text{sign}(w^T x)$ → ① tolerate noise ② nonlinear transformation
(NP-hard (매우 어려움))

PPLA (pocket algorithm) ; 기존의 w보다 E_{in}이 낮게 나오면 업데이트

ex) 손글씨 분류 ⇒ multi class 분류를 여러개의 binary 분류로 나누기 / feature extraction 사용

ii) Regression $y \sim P(y|x)$

파라미터 추정 방법 :
{ least square — ordinary least squares (OLS) (MSE 최소화.) · generalized least squares.
maximum likelihood — ridge · lasso · least absolute deviation regression
others — Bayesian linear regression, principal component regression

$E_{out}(h) = \mathbb{E}[(h(x) - y)^2]$, $E_{in}(h) = \frac{1}{N}\sum (h(x_n) - y_n)^2 = \frac{1}{N}\|Xw - y\|^2 = \frac{1}{N}(w^T X^T X w - 2w^T X^T y + y^T y)$

↳ 2차 함수 형태.

$W_{lin} = \underset{w \in \mathbb{R}^{d+1}}{\text{argmin}}\, E_{in}(w)$ ⇒ gradient 이용.

* gradient $\nabla_w (w^T A w) = (A + A^T)w$, $\nabla_w (w^T b) = b$. { scalar w $E_{in}(w) = aw^2 - 2bw + c$
vector w $E_{in}(w) = w^T A w - 2w^T b + c$

$\nabla E_{in}(w) = \frac{2}{N}(X^T X w - X^T y) = 0$ ⇒ $w = (X^T X)^{-1} X^T y$

↳ $X^T X$ 역행렬 가능 시 : $X^\dagger = (X^T X)^{-1} X^T$ ⇒ 유사역행렬 (pseudo-inverse) 사용가능.

" 불가능 시 : 유사역행렬이 가능하나, unique하지 않음.

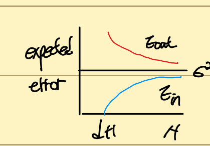
⇒ 대부분 $X^T X$는 역행렬이 가능하고, 데이터가 많으면 연산 코스트가 비싸짐.

역행렬 없이 $X^\dagger$ 계산 : $X = U\Sigma V^T$ (SVD, 특이값분해) 가능. $X^T X w_{lin} = X^T y$

- Generalization of linear regression $E_{out}(g) - E_{in}(g) = O(\frac{d}{N})$, N : 데이터 개수, d : dimension

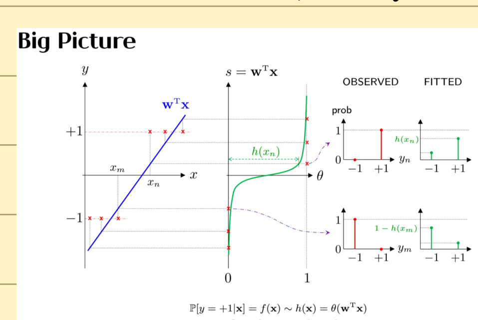
$E_{in} = \sigma^2(1 - \frac{d+1}{N}) < \sigma^2$

$E_{out} = \sigma^2(1 + \frac{d+1}{N}) > \sigma^2$



iii) Logistic regression

$h(x) = \theta(w^T x)$, θ : logistic function , $\theta(s) = \frac{e^s}{1 + e^s} = \frac{1}{1 + e^{-s}}$



* Cross-entropy error measure

$E_{in}(w) = \frac{1}{N}\sum \ln(1 + e^{-y_n w^T x_n})$

$P(y|x) = \begin{cases} h(x), & y = +1 \\ 1 - h(x), & y = -1 \end{cases}$

→ MLE : 독립사건, $\prod^{N} P(y_n|x_n)$

$= -\frac{1}{N}\ln(\prod P(y_n|x_n)) = \frac{1}{N}\sum \ln \frac{1}{\theta(y_n w^T x_n)}$

$= \frac{1}{N}\sum \{ [\![y_n = 1]\!] \ln \frac{1}{h(x_n)} + [\![y_n = -1]\!] \ln \frac{1}{1 - h(x_n)} \}$

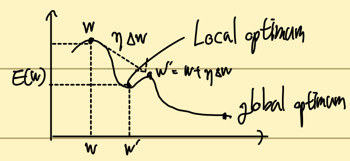
⇒ minimize E_{in} by setting $\nabla E_{in}(w) = 0$

Cross Entropy 'observed' 'fitted'

⇒ $p \log \frac{1}{q} + (1-p) \log \frac{1}{1-q}$, $\{p, 1-p\} \cdot \{q, 1-q\}$

- Iterative optimization ; iteratively set the gradient to zero

  * gradient descent (general algorithm)



  → direction? $\Delta w = -\nabla E_{in}(w(t))$

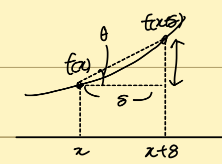
  how much? η = learning rate

  → caution : local optimum → start point에 의존.

  but 로지스틱에서는 cross-entropy를 사용하므로 convex로 증명.

  $w(t+1) = w(t) + \eta \cdot \hat{v}$ ⇒ pick $\hat{v}$. $E_{in}(w(t+1))$이 감소, 즉, { $E_{in}(w(t+1)) - E_{in}(w(t))$ 감소 ← 작을수록 좋음. / $E_{in}(w(t)) - E_{in}(w(t+1))$ 증가 }

  ⇒ Taylor 전개. $f(x+\delta) - f(x) \approx \nabla f(x)^T \delta$



  $\|x\|_2$ $x^T y \geq -\|x\|$, $y = -\frac{x}{\|x\|}$ 일 때 $x \cdot y$는 180°

  $\Delta E_{in} = E_{in}(w(t+1)) - E_{in}(w(t))$ → x·y가 180° 일 때

  $\approx \eta \nabla E_{in}(w(t))^T \hat{v} \geq -\eta \|E_{in}(w(t))\|$ 즉, best $\hat{v} = -\frac{\nabla E_{in}(w(t))}{\|\nabla E_{in}(w(t))\|}$ 이면 ΔE_{in} = 최소.

  ⇒ η : too small → far from local optimum / too large → bouncing around ⇒ large to small (ideal)

  $\eta_t = \eta \|\nabla E_{in}(w(t))\|$, norm of gradient를 이용.

  ⇒ $w(t+1) = w(t) - \eta \nabla E_{in}(w(t))$ approximate O(1) by stochastic gradient descent

- optimization

  iter upper bound 설정. · $\|\nabla E_{in}(w(t))\|$ threshold 설정. · E_{in} lower bound 설정

  +) batch · stochastic · mini-batch gradient descent 등이 있음.

- Information $h(x) := -\log P(x)$, 드물게 일어나면 높은 정보를 가짐.
- Entropy $H[x] := -\sum p_i \log p_i$, 정보의 평균, 불확실성의 척도. { $-(0.8\log 0.8 + 0.2\log 0.2) = 0.5$ / $-(\log 1 + \log 0) = 1$ }
- KL-divergence $KL(p|q) \triangleq (-\sum p_i \log q_i) - (-\sum p_i \log p_i) = -\sum p_i \log(\frac{q_i}{p_i})$, 정보량의 손실.
- Cross-Entropy $CE(p,q) \triangleq -\sum p_i \log q_i$, 예측된 확률의 차이 $f(x) \to h(x)$; p, q

  $p\log\frac{1}{q} - (1-p)\log\frac{1}{1-q}$ $\{p, 1-p\}$ observed $\{q, 1-q\}$ fitted

IV. NN

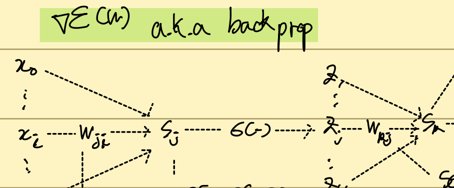
- function signal (forward propagation) · error signal (backward propagation)
- soft max function : $\sigma : \mathbb{R}^K \mapsto \mathbb{R}^K$, $\sigma(h)_j = \frac{e^{h_j}}{\sum^K e^{h_k}}$, multinomial prob. distribution
- hidden unit $z_j(x) = \sigma(\sum^d w_{ji} x_i)$

  output unit $h_k(x) = \sigma(\sum^H w_{kj} z_j) = \sigma(\sum_{j=0}^H w_{kj} \sigma(\sum_{i=0}^d w_{ji} x_i))$

  $e_k = h_k - y_k$, $\varepsilon_n = \frac{1}{2}\sum e_{kn}^2$, $\varepsilon_p = \frac{1}{N}\sum^N \varepsilon_n$ → $\varepsilon_n = \varepsilon_n(w) = \frac{1}{2}\sum(h_k - y_k)^2$

  find $w^* = \arg\min_w \frac{1}{N}\sum \varepsilon_n(w)$, $\varepsilon(w) = \varepsilon(w) - \eta\nabla\varepsilon(w)$

  $\nabla\varepsilon(w)$ a.k.a backprop



  local gradient (delta error) $\delta_k = \frac{\partial\varepsilon}{\partial s_k} = \frac{\partial \varepsilon}{\partial e_k} \cdot \frac{\partial e_k}{\partial h_k} \cdot \frac{\partial h_k}{\partial s_k} = e_k \cdot \sigma'(s_k)$

  sensitivity factor : $\frac{\partial\varepsilon}{\partial w_{kj}} = \frac{\partial\varepsilon}{\partial s_k} \cdot \frac{\partial s_k}{\partial w_{kj}} = \delta_k \cdot z_j$

  $\delta_j = \frac{\partial\varepsilon}{\partial z_j} = \frac{\partial \varepsilon}{\partial z_j} \cdot \frac{\partial z_j}{\partial s_j}$

  $\frac{\partial\varepsilon}{\partial w_{ji}} = \frac{\partial\varepsilon}{\partial s_j} \cdot \frac{\partial s_j}{\partial w_{ji}} = \delta_j \cdot x_i$

  $w_{kj} \leftarrow w_{kj} + \Delta w_{kj}$

  $\delta_j = \sigma'(s_j)\sum^K \delta_k w_{kj}$

  $\Delta w_{kj} = -\eta \frac{\partial \varepsilon}{\partial w_{kj}} = -\eta \cdot \delta_k \cdot z_j$

  $\Delta w_{ji} = -\eta \cdot \frac{\partial\varepsilon}{\partial w_{ji}} = -\eta \cdot \delta_j \cdot x_i$

  * Widrow-Hoff learning rule aka LMS rule

  $\sigma(s) = s \Rightarrow \delta_k = e_k \sigma'(s_k) = e_k$

  $\Delta w_{kj} = -\eta \cdot e_k \cdot z_j$

- single gate $in \to f \to out$ $\frac{\partial\varepsilon}{\partial in} = \frac{\partial\varepsilon}{\partial out} \cdot \frac{\partial out}{\partial in} \Rightarrow \frac{\partial\varepsilon}{\partial out} \cdot f'(in)$

  Multiplication $in_2, in_1 \to f \to out$ (sum) $\frac{\partial\varepsilon}{\partial in_1} = \frac{\partial\varepsilon}{\partial out} \cdot in_2$, $\frac{\partial\varepsilon}{\partial in_2} = \frac{\partial\varepsilon}{\partial out} \cdot in_1$

V. CNN

VI. RNN