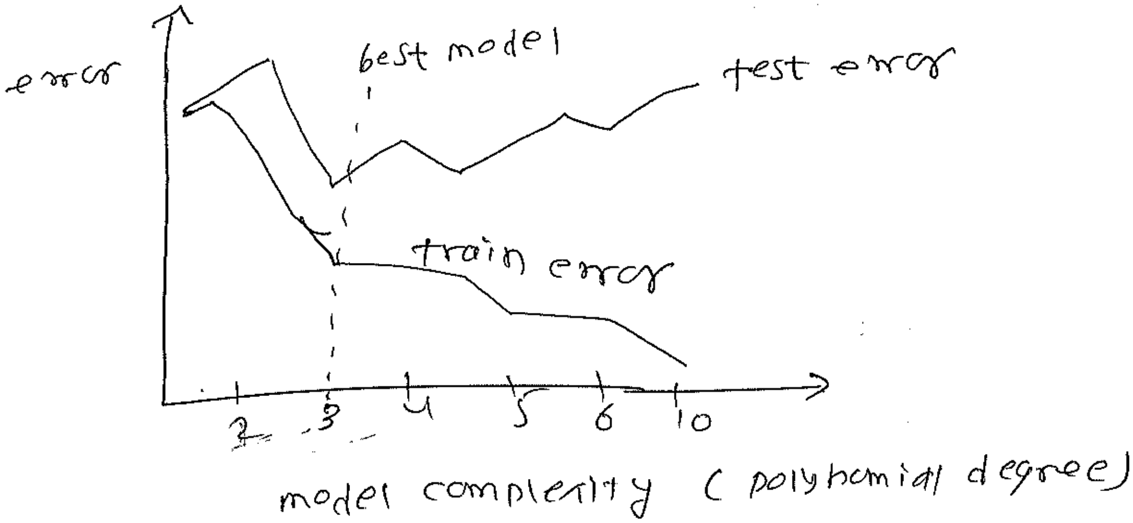
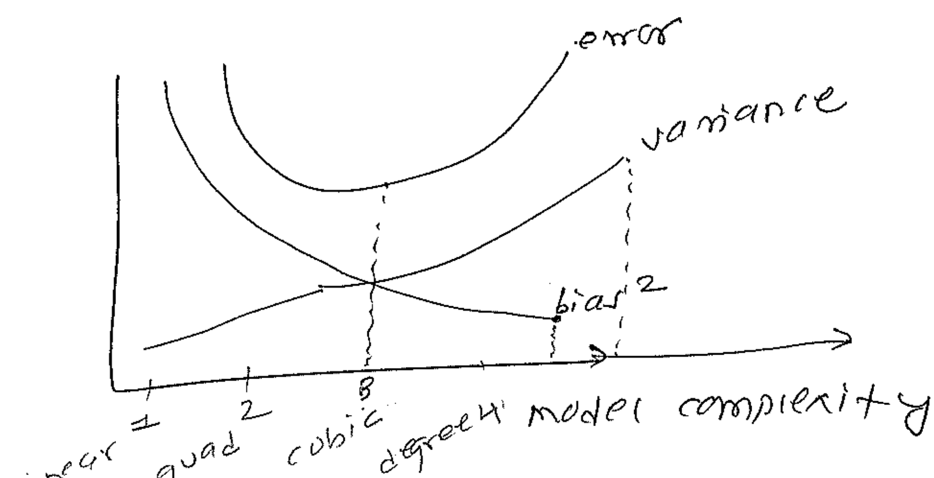
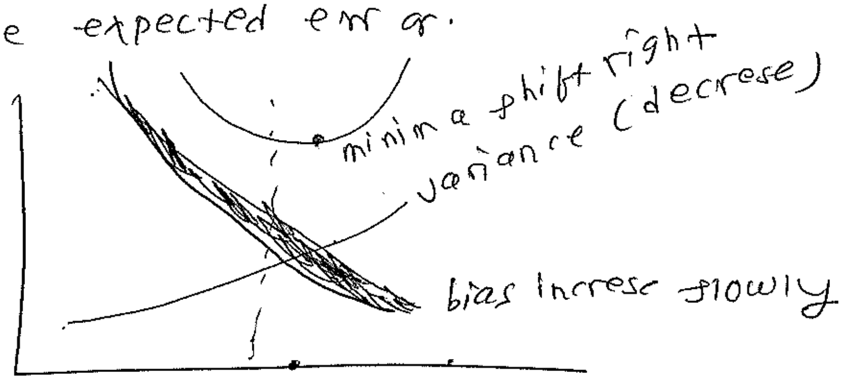
overfitting





when training data increase, variance is reduced and slightly more complicated model minimizes the expected error.



① Show that $X^T A Y$ is a valid kernel if A is sym. p.s.d.

$\Rightarrow$ $A = Q^T \Lambda Q$ where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$ and $\lambda_1, \lambda_2, \cdots, \lambda_n \geq 0$

define $F = \text{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \cdots, \sqrt{\lambda_n})$ then $\Lambda = F^T F$

$\Rightarrow A = Q^T F^T F Q$

$\Rightarrow x^T A y = x^T Q^T F^T F Q y$

$= (FQx)^T (FQy)$

$= \phi(x)^T \phi(y)$

where $\phi(x) = FQx$

Aside:
$(AB)^T = B^T A^T$
$(ABC)^T = C^T B^T A^T$

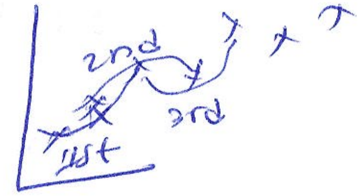
$((AB)C)^T = C^T (AB)^T$
$= C^T B^T A^T$

⑥ cubic spline smoothing

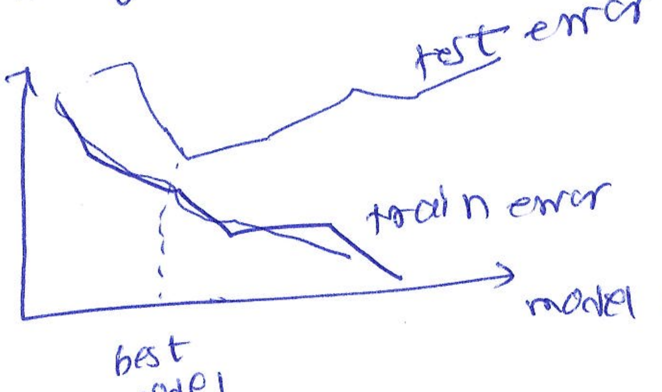
Take 3 points x $x-x_i$ $x+x_i$

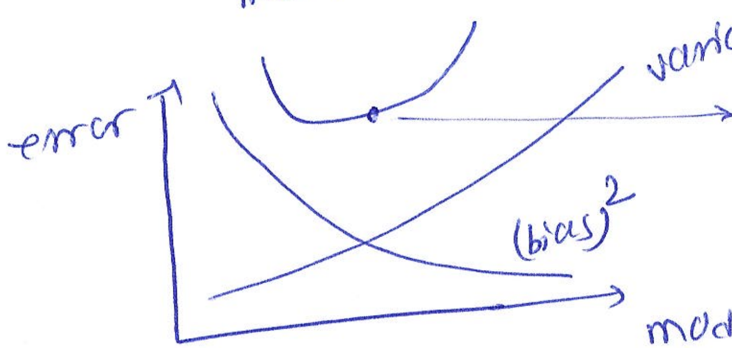
$$s_i(x) = a_i(x-x_i)^3 + b_i(x-x_i)^2 + c_i(x-x_i) + d_i$$

$$\forall x \in [x_i, x_{i+1}]$$



⑦ overfitting





→ when training data increses variance decrese bias increse and model becomes slightly complex to minimize error.

① Likelihood function for Logistic Regression

$p(t_n|w) = h_n^{t_n} \cdot (1-h_n)^{1-t_n}$

$$p(t|w) = \prod_{n=1}^{N} h_n^{t_n} (1-h_n)^{1-t_n}$$

$$\text{cost } E = \frac{1}{N} \cdot (-\ln p) = -\frac{1}{N} \sum_{n=1}^{N} \left[t_n \ln h_n + (1-t_n) \ln(1-h_n) \right]$$

$$\text{grad} = \nabla_w J = \frac{1}{N} \cdot \sum_{n=1}^{N} (h_n - t_n) \cdot x_n$$

② perceptron criterion

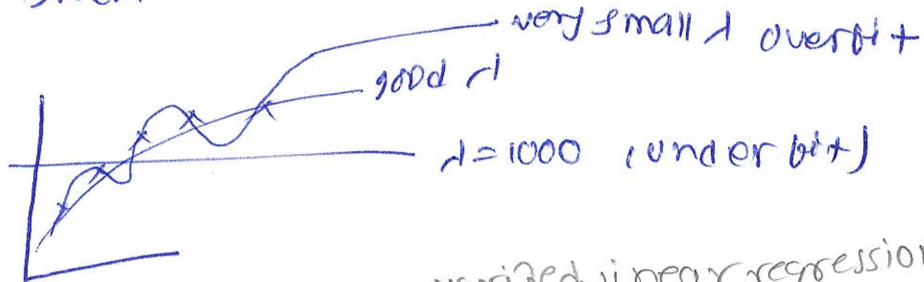
minimize $E_p(w) = -\sum_{n \in \text{Misclassified}} t_n w^T x_n$

initialize $w=0$, $\bar{w}=0$, $\tau=1$
for $i = 1 : N$
$h_n = \text{sgn}(w^T x_n)$
if $h_n \neq t_n$ then
$w = w + t_n x_n$
$\tau = \tau + 1$
$\bar{w} = \bar{w} + w$
$\tau = \tau + 1$
return $\bar{w}/\tau$

Test: $\text{sgn}(w^T x)$

① L2 → more penalty larger weights
does not drive weights to zero
weight vector is NOT sparse.

② $\lambda \uparrow$ ⇒ high bias, low variance, underfit, simpler model
drives weights closer to zero.
small change in training data ⇒ big change in estimate



cost function for L2-regularized linear regression

N-examples
M-features

$$X = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & x_1^M \\ x_2^1 & x_2^2 & x_2^3 & x_2^M \\ x_N^1 & x_N^2 & x_N^3 & x_N^M \end{bmatrix}$$

③ $$J = \frac{1}{2N}\sum_{n=1}^{N} (h_n - t_n)^2 + \frac{\lambda}{2}\|w\|^2$$

$$J(w) = \frac{1}{2N}\sum_{n=1}^{N}\left(w_0 + \sum_{j=1}^{M} w_j x_n^j - t_n\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{M} w_j^2$$

(we omit w_0 in regularization)

regularizing w_0 = shifting origin of target
⇒ same change in all target values ⇏ similar change in estimates

④ Batch GD

for num_iters:

$w^{\tau+1} = w^{\tau} - \eta \nabla J$

learning rate

Stochastic GD

for num_iters:

for sample in data:

$w^{\tau+1} = w^{\tau} - \eta \nabla J$

momentum GD $+\gamma v^{\tau}$

$v^{\tau+1} = \eta \nabla J(w^{\tau})$

$w^{\tau+1} = w^{\tau} - v^{\tau+1}$

$\gamma \approx 0.9$

Nesterov Accelerated gradient

$$w^{\tau+1} = w^{\tau} - \eta \nabla J(w^{\tau} - \gamma w^{\tau}) - \gamma w^{\tau}$$

(5) Gradient Descent → $v^{\tau+1} = \begin{cases} \eta \nabla J(w^\tau) & \text{vanilla} \\ \eta \nabla J(w^\tau) + \gamma v^\tau & \text{with momentum} \\ \eta \nabla J(w^\tau - \gamma v^\tau) + \gamma v^\tau & \text{Nesterov Accelerated gradient NAG} \end{cases}$

η = learning rate
γ = momentum hyperparameter
$\tilde{w}$ = weight vector at time stamp $\tilde{\tau}$

$$w^{\tau+1} = w^\tau - v^{\tau+1}$$

$\gamma \approx 0.9$

(2b) Types of GD based on data used:

Batch GD → use all data
Stochastic GD → use only one example and update w after each iteration
minibatch GD → use constant number of examples and update w after each

$$J = \frac{1}{2N} \sum_{n=1}^{N} (h_n - t_n)^2$$

$$\nabla J = \frac{1}{N} \sum_{n=1}^{N} (h_n - t_n) \cdot x_n = \frac{1}{N} \cdot \text{np.sum}((h-t) \cdot x)$$

$$w = w - \eta \, \text{grad}$$

(5c) GD vs Normal eqn

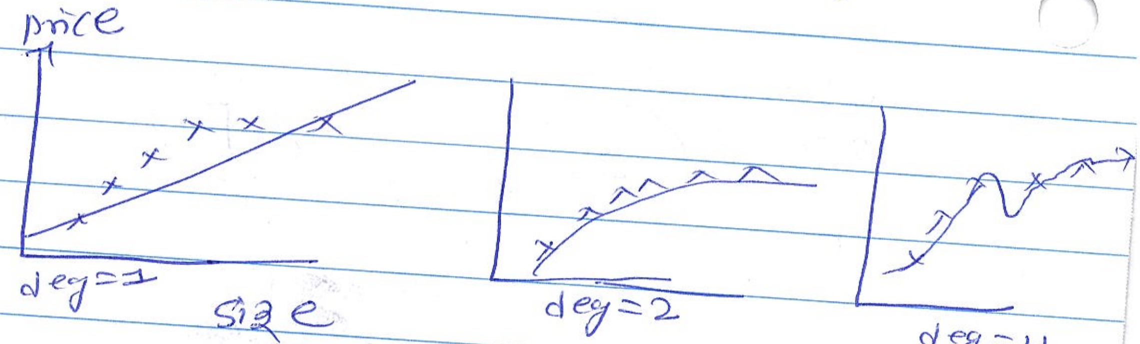
$w = w - \eta \, \text{grad}$

$$w = \underbrace{(X^T X)^{-1} X^T}_{\text{pseudoinverse (pinv)}} t$$

$$w = \underbrace{(X^T X + \lambda N I)^{-1}}_{} X^T t$$

for L2 regularized I is identity matrix of shape $X^T X$
$I[0][0] = 0$ for not to regularize bias term

Bias-variance Trade off



$w_0 + w_1 x$ | $w_0 + w_1 x + w_2 x^2$ | $w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

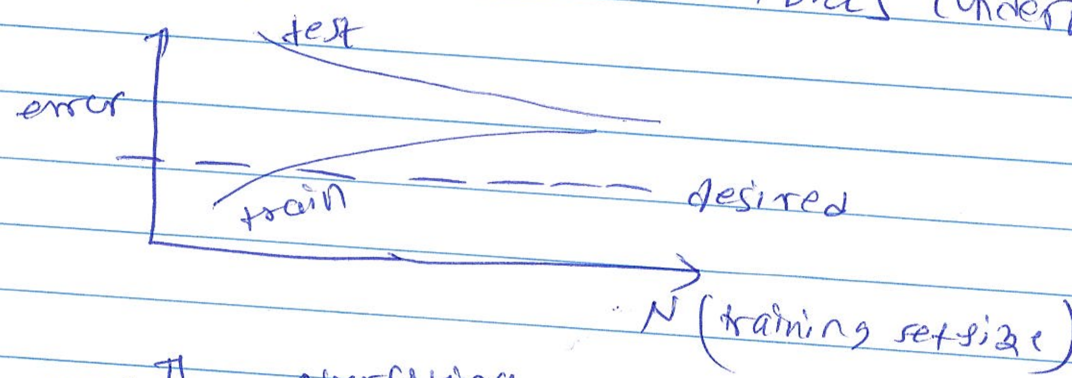
high bias (underfit) (simple model) | just right | high var. (overfit)

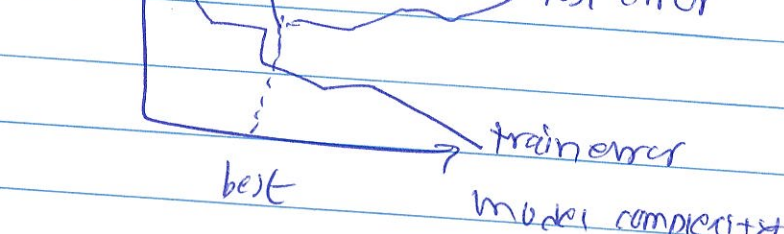
more training eg → fix high σ^2 (overfitting)

smaller feature set → fix high σ^2 (")

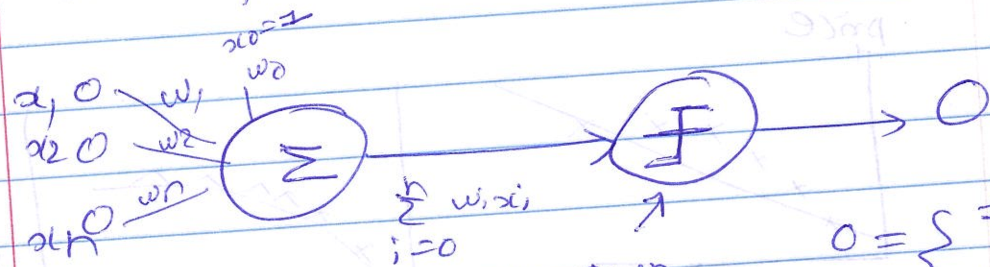
larger feature set → fix high bias (underfitting)

Learning curve for high bias (underfit)





perceptron



$$O = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

output

$$= \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise} \end{cases}$$

① cost for multivariate linear regression

$$J = \frac{1}{2N} \sum_{n=1}^{N} (h_n - t_n)^2 \qquad h = w^T x \qquad h = X1 @ w.T$$

$$J = \frac{1}{2N} \sum_{n=1}^{N} \Big(\underbrace{\sum_{j=0}^{M} w_j x_n^j}_{} - t_n \Big)^2 \qquad h_n = \sum_{j=0}^{M} w_j x_n^j$$

t

$h_1 = w_0 + w_1 x_1^{(1)} + w_2 x_1^{(2)} + w_M x_1^{(M)}$ first row of sample

$$w = [w_0 \; w_1 \; w_2 \; w_3]_{1\times 4}$$

bias

X1

$$\begin{bmatrix} 1 & 10 & 100 \\ 1 & 20 & 200 \\ 1 & 30 & 300 \\ 1 & 50 & 500 \end{bmatrix}_{50\times 4} \qquad \begin{bmatrix} 1K \\ 2K \\ 3K \\ 50K \end{bmatrix}_{50\times 1}$$

$h = (50,4) \cdot (4,1) = (50,1)$

2 features age, floorsize

$$e = h - t = \begin{bmatrix} h_1 - t_1 \\ h_2 - t_2 \\ \vdots \\ h_{50} - t_{50} \end{bmatrix}_{50\times 1}$$

SSE = (h-t) ** 2

$$MSE = \frac{(h-t) ** 2}{N}$$

MSE = np.mean (SSE)

RMSE = np.sqrt (MSE)

J = np.sum((h-t)**2) /2/N

J = 0.5 * np.mean (np.sum (h-t)**2)

$J = \frac{1}{2} \times MSE$ (single float)

$$\boxed{J = \frac{0.5}{len(t)} (h-t)^T (h-t)}$$

$$(h-t)^T (h-t) = [h_1-t_1 \; h_2-t_2 \cdots]_{1\times 50} \begin{bmatrix} h_1-t_1 \\ h_2-t_2 \\ h_n-t_n \end{bmatrix}_{50\times 1}$$

$$= \Big[\sum_n (h_n-t_n)^2\Big]_{1\times 1}$$

② gradient of J

$$\nabla_w J = \nabla_w \frac{1}{2N} \sum_n (h-t)^2 = \Big[\frac{\partial J}{\partial w_0} \; \frac{\partial J}{\partial w_1} \; \frac{\partial J}{\partial w_2} \; \frac{\partial J}{\partial w_M} \Big]_{4\times 1}$$

= (h-t).T @ X1 / N

= (1,50) . (50,4)

grad = $\nabla_w J$ = (1,4)

$$\boxed{grad\text{-}ols = (h-t).T @ X1}$$

$$\frac{\partial J}{\partial w_j} \; \frac{1}{2N} \sum_{n=1}^{N} (x w_n - t)^2 = \frac{1}{2N} \sum (x_n w_0 + x_n w_1 + \cdots + x_n w_M)$$

$$= \frac{1}{2N} \sum_{n=1}^{N} 2 (x w - t) \cdot X$$

[summation vanishes] we use matrix

$$\nabla_w J = \frac{1}{N} (h-t) \cdot X$$

(4) $X =$ design matrix $[\;]_{N \times M}$ $N =$ samples, $m =$ features

$X1 =$ biased design matrix $\left[\begin{smallmatrix}1 \\ 1\end{smallmatrix}\;\right]_{N, M+1}$

$t = [\;]_{N \times 1}$ column vector

$w = [w_0 \; w_1 \; w_2 \; w_M]_{1, M+1}$ row vector (2d array)

w = np.array(w).reshape(1, X1.shape[1]) $= (1, M+1)$

$h = X1 \,@\, w.T = (N, M+1)(M+1, 1) = (N, 1)$ same as t

linear regression

$e = h - t = N, 1 = 50, 1$

$$SSE = \sum_{n=1}^{N} (h-t)^2$$

$$MSE = \frac{1}{N} \sum_{n=1}^{N} (h-t)^2$$

$$J = \frac{1}{2N} \sum_{n=1}^{N} (h-t)^2 = \text{np.sum}((h-t)**2) / 2 / \text{float}(N)$$

$$RMSE = E_{RMS} = \sqrt{MSE}$$

Normal eqn: $w = \underbrace{(X^T X)^{-1} @ X^T}_{\text{moore penrose pseudo inverse of } X} @ \, t$

np.linalg.pinv(X)

multivariate Linear

bias	X floor-size	bed	t
			2K
	100	2	3K
1	200	3	5K
1	500	5	

t ⇓

m features and one bias

$$X = \begin{bmatrix} x_0^{(1)}=1 & x_1^{(1)} & \cdots & x_m^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \cdots & x_m^{(2)} \\ \vdots & \vdots & & \\ x_0^{N} & x_1^{(N)} & & x_m^{(N)} \end{bmatrix}_{N \times m+1} \quad \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_n \end{bmatrix}_{N \times 1}$$

There are m features and one bias total m+1

There are N samples for each feature

$$J = \frac{1}{2N}|XW - t|^2 = \frac{1}{2N}(XW-t)^T(XW-t)$$

$$\nabla_W J = \frac{1}{\cancel{2}}\frac{d}{dx}(XW-t) \cdot \cancel{2}(XW-t)$$

$$\frac{d}{dx}(x^T x) = 2x$$

x is a matrix

$$0 = X^T(XW-t)$$

$$\boxed{\frac{d}{dx}(Ax+b)^T(Ax+b) = 2A^T(Ax+b)}$$

$$X^T X W = X^T t$$

derivatives of matrix products

weights $\boxed{W = (X^T X)^{-1}(X^T t)}$

L1 vs L2 norm
more used

L2 → more penalty on large weights, but doesn't drive small weights to zero.

L1 → Less penalty for large wt, but leads many weights ~~towards~~ to (or very very close) to zero. leading to weight vector to be sparse.

Bias-variance Tradeoff

(a) Ridge: $\sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = RSS + \lambda \sum_{j=1}^{p} \beta_j^2$

shrinkage parameter which has to be tuned via validation set → regularization strength

Ridge
- $\lambda \uparrow$ ⇒ var↓ bias↑
- meaning: small change in training data big change in parameter estimates
- effect will increase with no of parameters

(b) Lasso β_j^2 is $|\beta_j|$

Ridge shrinks w_j but do not make $\emptyset$ but lasso makes them zero, makes less no of wts.

Logistic Regression → classification (binary)

linear regression $h = w^T x$

logistic regression $h = \sigma(x) = \frac{1}{1+e^{-w^T x}} = h(x) = y(x)$

likelihood function $p(t|w) = \prod_n h_n^{t_n} (1-h_n)^{1-t_n}$

-ve log likelihood, E or $J = -\ln p(t|w)$

$$= -\sum_{n=1}^{N} [t_n \ln h_n + (1-t_n) \ln(1-h_n)]$$

∴ cost or Error or Loss function

$$\boxed{E_D = -\sum_n [t_n \ln h_n + (1-t_n) \ln(1-h_n)]}$$

where $t_n \in \{0,1\}$ NOT $\{-1,1\}$

add regularizer,

$$E_w = \frac{\lambda}{2} w^T w$$

then L2-regularized logistic regression cost function

cost = $\boxed{E = -\frac{1}{N}\sum_n [t_n \ln h_n + (1-t_n) \ln(1-h_n)] + \frac{\lambda}{2} w^T w}$

λ is regularizer

$$h(x) = \sigma(x) = \frac{1}{1+e^{-w^T x}}$$

$$h_n = h(x_n) = \sigma(x_n) = \frac{1}{1+e^{-w^T x_n}}$$ is sigmoid f^n

classification (multiclass)

Softmax Regression

Training set: $(x_1, t_1), (x_2, t_2), \ldots (x_n, t_n)$

$$X = [1, x_1, x_2, \ldots, x_n]$$

$$t_1, t_2, \ldots, t_n \in \{1, 2, \ldots, K\}$$

$$w_k = [w_{k0}, w_{k1}, w_{k2}, \ldots]^T$$

one weight vector per class,

$$p(c_k|x) = \frac{e^{w_k^T x}}{\sum_j e^{w_j^T x}}$$

prob that nth example x_n has class t_n

$$p(t_n|x_n) = \frac{e^{w_{t_n}^T x_n}}{\sum_j e^{w_{t_j}^T x_n}}$$

likelihood is the joint probability of all classes,

$$L(w) = \prod_{n=1}^{N} p(t_n|x_n)$$

MLE

sample: $X_1, X_2, \ldots X_N$

prob of x is $f(x_n, w)$

joint prob of $X_1, X_2, \ldots X_N$ is called likelihood $L(w)$

$$L(w) = p(X_1 = x_1, \ldots X_N = x_N)$$

$$= f(x_1, w) \ldots$$

pdf of x_1

$$L(w) = \prod_{n=1}^{N} f(x_n, w)$$

cost is the -ve log likelihood,

$$E_D(w) = -\frac{1}{N} \ln L(w)$$

$$= -\frac{1}{N} \ln \prod_{n=1}^{N} p(t_n|x_n)$$

$$= -\frac{1}{N} \sum_n \ln p(t_n|x_n)$$

$$= -\frac{1}{N} \sum_n \ln \frac{e^{w_{t_n}^T x_n}}{Z(x_n)}$$

w_{t_n} is weight vector for nth example

w_k is weight vector for all the examples belonging to class k.

$$\boxed{E_D(w) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_k(t_n) \ln\left(\frac{e^{w_k^T x_n}}{Z(x_n)}\right)}$$

MAP solution for LR

$$p(t|w) = \prod_{n=1}^{N} h_n^{t_n} (1-h_n)^{1-t_n}$$

$$p(w) = \left(\frac{\alpha}{2\pi}\right)^{\frac{m+1}{2}} e^{-\frac{\alpha}{2} w^T w} \propto e^{-\frac{\alpha}{2} w^T w}$$

$$p(w|t) = \frac{p(t|w)\; p(w)}{p(t)} \propto p(t|w)\; p(w)$$

now, $\vec{w}_{MAP} = \underset{w}{\operatorname{argmax}}\; p(w|t)$

$$= \underset{w}{\operatorname{argmax}}\; p(t|w)\; p(w)$$

$$= \underset{w}{\operatorname{argmin}}\; -\ln p(t|w) - \ln p(w)$$

$$= \underset{w}{\operatorname{argmin}}\; -\ln \prod_n h_n^{t_n} (1-h_n)^{1-t_n} - \ln e^{-\frac{\alpha}{2} w^T w}$$

$$= \underset{w}{\operatorname{argmin}}\; -\sum_n \ln \left(h_n^{t_n} (1-h_n)^{1-t_n} \right) + \frac{\alpha}{2} w^T w$$

$$= \underset{w}{\operatorname{argmin}}\; \underbrace{-\sum_n \left[t_n \ln h_n + (1-t_n) \ln (1-h_n) \right]} + \underbrace{\frac{\alpha}{2} w^T w}$$

$$\boxed{\vec{w}_{MAP} = \underset{w}{\operatorname{argmin}}\; E_D(w) + E_W(w)}$$

Ans

softmax Regression

$$p(c_k|x) = \frac{e^{w_k^T x}}{\sum_j e^{w_j^T x}}$$

note:

$$h(w) = \frac{1}{\sum_{k=1}^{K} e^{w_k^T x}} \begin{bmatrix} e^{w_1^T x} \\ e^{w_2^T x} \\ \vdots \\ e^{w_K^T x} \end{bmatrix}$$

$$p(t_n|x_n) = \frac{e^{w_{t_n}^T x_n}}{\sum_{j=1}^{N} e^{w_{t_n}^T x_n}}$$

$$E_D(w) = \frac{1}{N} \cdot -\ln \prod_n p(t_n|x_n)$$

$$= -\frac{1}{N} \ln \prod_n \frac{e^{w_{t_n}^T x_n}}{\sum_j e^{w_{t_n}^T x_n}}$$

$$= -\frac{1}{N} \cdot \sum_n \ln \frac{e^{w_{t_n}^T x_n}}{\sum_j e^{w_{t_n}^T x_n}}$$

$$\boxed{E_D(w) = -\frac{1}{N} \sum_n \sum_k \delta_k(t_n) \ln \left(\frac{e^{w_k^T x_n}}{\sum_k e^{w_k^T x_n}} \right)}$$

Regularizer

$$+ \frac{\alpha}{2} \sum_{k=1}^{K} w_k^T w_k$$

$$\frac{\partial E_w}{\partial w_k} = \alpha w_k$$

(no summation)

Let, $E_n = \sum_k \delta_k(t_n) w_k^T x_n - \sum_k \delta_k(t_n) \ln \left(\sum_k e^{w_k^T x_n} \right)$

$$\frac{\partial E_n}{\partial w_j} = \delta_j(t_n) x_n - \delta_j(t_n) \cdot \frac{1}{\sum_j e^{w_j^T x_n}} \cdot e^{w_j^T x_n} \cdot \delta_j(t_n) x_n$$

prevent overflow

$$p(c_k|x_n) = \frac{e^{w_k^T x_n}}{\sum_k e^{w_k^T x_n}}$$

$$c = \max_{1 \le k \le K} w_k^T x_n$$

$$p(c_k|x_n) = \frac{e^{(w_k^T x_n - c)}}{\sum_k \exp(w_k^T x_n - c)}$$

$$\frac{\partial E_n}{\partial w_k} = \delta_k(t_n) x_n - \frac{e^{w_k^T x_n}}{\sum_k e^{w_k^T x_n}} \cdot x_n$$

$$\frac{\partial E_n}{\partial w_k} = [\delta_k(t_n) - p(c_k|x_n)] x_n$$

$$\Rightarrow \boxed{\frac{\partial E_D(w)}{\partial w_k} = -\frac{1}{N} \sum_n [\delta_k(t_n) - p(c_k|x_n)] x}$$

there are K-such equations for each classes.

① MLE VS MAP

MLE = maximize p(data|params) by searching over parameters

MAP = maximize p(param|data) by searching over params and accounting for prior over params

MLE → finds w by maximizing likelihood fn p(x|w)

MAP → maximizes the posterior prob p(w|x)

② classification maps inputs to discrete outputs
Regression " continuous " .

③ PCA & feature selection

similarity : reduce the dimension of data
difference : feature selection finds a subset of features,
PCA produces a smaller new set

perceptron

perceptron criterion: $w^T x_n > 0$ for $t_n = +1$

$w^T x_n < 0$ for $t_n = -1$

want: $t_n w^T x_n > 0$ for all patterns

minimize: $-w^T x_n t_n$ for all misclassified patterns M

$$\boxed{\underbrace{E_p}_{\text{perceptron}}(w) = -\sum_{n \in M} w^T x_n t_n}$$

M: mistakes (misclassified)

Binary perceptron

initialize $\vec{w} = 0$

for $n = 1, \cdots N$

$h_n = \text{sgn}(w^T x_n)$

if $h_n \neq t_n$ then

$w = w + t_n x_n$

repeat until convergence or given number of epochs

① why kernels are symmetric?

Inner products are symmetric by definitions, so, therefore if the kernel function represent an inner product in some Hilbert space, then the kernel function must be symmetric as well.

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

$$= \langle \phi(y), \phi(x) \rangle \quad (\because \text{property of inner product})$$

$$= K(y, x)$$

② Show $K(x, z) = x^T A^T A z$ is a valid kernel.

let $\phi(x) = Ax$,

then,

$$\langle \phi(x), \phi(z) \rangle = \phi(x)^T \phi(z)$$

$$= (Ax)^T (Az)$$

$$= x^T A^T A z$$

$$= K(x, z)$$

$\therefore$ $K(x, z)$ is an inner product in some Hilbert space.

(3) AA^T is PSD matrix (inderit)

<u>proof</u> Let, $A \in \mathbb{R}^{m \times n}$

$\lambda =$ eigen value of AA^T

$q =$ eigen vector of λ

$(AA^T) q = \lambda q$ (premultiply by q^T)

$\Rightarrow q^T A A^T q = q^T \lambda q$ $\because AA^T = A^T A$

$$\Rightarrow \lambda = \frac{q^T A A^T q}{q^T q} = \frac{q^T A^T A q}{q^T q}$$

$$= (Aq)^T Aq / q^T q$$

$$= \frac{z^T z}{q^T q} \quad \text{where } z = A^T q$$

here $z^T z \geq 0$

$q^T q \geq 0$

$\therefore \lambda \geq 0$ so AA^T is PSD.

...d estimation maximum a priori

MLE vs MAP

- MLE maximizes the $\ln p(D|w)$ likelihood of model (w) w.r.t. data D

- MAP maximizes the $\ln p(w|D)$ likelihood of data D w.r.t. model w

moreover, MAP additionally uses priors.

Low variance High variance

Low Bias

High Bias