$$= \sum_{i=1}^{N} \log \pi_1 + \log\left[\text{Gaussian dist}\right]$$

$$+ \sum_{i=1}^{N} \log \pi_2 + \log\left[\quad\right]$$

$$\therefore N_1 \log \pi_1 + \sum_{y_i=1}^{N}\left[\quad\right] + \sum_{y_i=2}^{N}\left[\quad\right]$$

$$+ N_2 \log \pi_2 \qquad - \frac{Nd}{2}\log|\Sigma| + \text{constant}$$

- Maximize LL ~~wrt~~ using $\pi_1\ \pi_2\ \mu_1\ \mu_2\ \Sigma$ subject to $\pi_1 + \pi_2 = 1$

Find gradient wrt each parameter. Make it zero.
This is valid only if LL is concave

Cntd on next pg

10/8

* Convex Functions and Optimization

$\vec{\theta} = \theta_1, \theta_2 \ldots \theta_r$. Find $\min F(\vec{\theta})$

$\theta_i \in \mathbb{R}$, $\vec{\theta} \in \mathbb{R}^r$

$F(\theta) \in \{\text{convex, concave, neither}\}$

1 $F(\theta)$ is convex.

* If $F(\theta)$ is convex, then $-F(\theta)$ is concave

PAGE No.
DATE / /

Definition $F(\theta)$ is convex iff $\forall\ \theta_1^*, \theta_2^* \in \mathbb{R}^r$, $\forall\ \lambda \in [0,1]$,

$$\lambda F(\theta_1) + (1-\lambda) F(\theta_2) \geq F(\lambda\theta_1 + (1-\lambda)\theta_2)$$

Chord lies above curve

- For local minimas of differentiable $F(\theta)$.

$$\nabla F(\theta^*) = 0$$

Definition Gradient :-

$$\nabla F(\vec{\theta}) = \begin{bmatrix} \frac{\partial F}{\partial \theta_1} \\ \vdots \\ \frac{\partial F}{\partial \theta_r} \end{bmatrix}$$

Continued from prev page.

$$F(\theta) = N_1 \log \pi_1 + \sum \frac{-1}{2} \frac{(x_{1i} - \mu_{11})^2}{\sigma^2} + \sum \frac{-1}{2} \frac{(x_i - \mu_{21})^2}{\sigma^2}$$

$$+ N_2 \log \pi_2 - N \log \sigma + \cancel{\gamma(1)}$$

Maximize wrt $\pi_1, \pi_2, \mu_{11}, \mu_{21}, \sigma$ such that $\pi_1 + \pi_2 = 1$

Write $\pi_2 = 1 - \pi_1$

$$\therefore \nabla LL(\theta) = \begin{bmatrix} \frac{N_1}{\pi_1} - \frac{N_2}{1-\pi_1} \\ \frac{\sum (x_{1i} - \mu_{11})}{\sigma^2} \\ \frac{\sum (x_{1i} - \mu_{21})}{\sigma^2} \\ \frac{\sum (x_{1i} - \mu_{11})^2}{\sigma^3} + \frac{\sum (x_{1i} - \mu_{21})^2}{\sigma^3} - \frac{N}{\sigma} \end{bmatrix} = \vec{0}$$

$$\pi_1 = \frac{N_1}{N_1 + N_2}, \quad \mu_{11} = \frac{\sum_{y_i=1} x_i}{N_1}, \quad \mu_{22} = \frac{\sum_{y_i=2} x_i}{N_2}$$

$$\sigma = \frac{\sum_{y_i=1} (x_i - \mu_{11})^2 + \sum_{y_i=2} (x_i - \mu_{21})^2}{N_1}$$

III Conditional Probabilistic Classification

Logistic Regression.

Assume $P(y|x) \sim$ Bernoulli(θ), where $y \in \{1, 2\}$

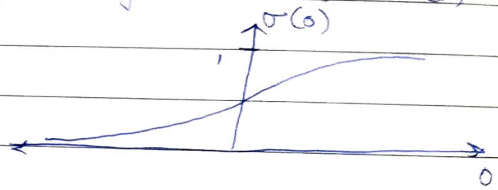
- Parameter of Bernoulli depends on inputs.

General Trick: Compute a function g of $x_1, \dots x_d$, such that

$$g(x) \in [-\infty, \infty]$$

Squash this $g(x)$ to be between $(0, 1]$ so as to look like a Bernoulli parameter. using the sigmoid function $\sigma(a)$

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$



- Define $P(y = 1 | x) = \sigma(g(x)) = \frac{1}{1 + e^{-g(x)}}$

$$P(y = 2 | x) = 1 - P(y = 1 | x) = \frac{1}{1 + e^{g(x)}}$$

Define $y = 2$ to be $y = -1$.

$$P(y | x) = \frac{1}{1 + e^{-yg(x)}} \qquad y \in \{-1, +1\}$$

$g(x)$ is learned from data.

→ Linear Logistic Regression :-

$g(x) = W^T x + W_0$ where.

$$W \in \begin{bmatrix} W_1 \\ \vdots \\ W_d \end{bmatrix} \quad \text{for } W_i \in \mathbb{R}$$

$$g(x) = W_1 x_1 + w_2 x_2 + w_3 x_3 \dots w_0$$

PAGE NO
DATE / /

$$P(y|x) = \frac{1}{1 + e^{-y(\omega_0 x_1 \cdots \omega_d)}}$$ ★ Is $\sum P(y_i|x) = 1$?

for $y = \{-1, 1\}$

ω_0 to ω_d : The $d+1$ parameters of the classifier that will be learned from training data.

- $D \equiv \{(x_1, y_1) \cdots (x_n, y_n)\}$

Find ω_i so as to maximize the conditional likelihood of D.

- As contraire, for generative classifiers, we fit joint likelihood

Conditional	Generative
$\max_\theta \sum_{i=1}^{N} \log P_r(y_i \mid x_i, \theta \equiv W)$	$\max_\theta \sum_{i=1}^{N} \log P_r(x_i, y_i \mid \theta)$

* Conditional is better than generative because it does not make unreasonable assumptions (especially for high-dimensional data)

- Maximize $\sum_{i=1}^{N} \log P(y_i \mid x_i, \theta)$ wrt $\theta = \{\omega_0 \cdots \omega_d\}$

$$= \max_{\vec{W}} \sum_{i=1}^{N} \log \frac{1}{1 + e^{-y_i(W^T x_i + W_0)}}$$

* Sigmoid function is concave.

$$\sigma''(0) = \frac{-e^0}{(1+e^0)^2} < 0$$

* If $F(o)$ is concave in o and $o = W^T x + W_0$, H
F is also concave in $(W_0 \cdots W_d)$

P.T.O.

max $LL(W, W_0, D)$ can be solved by $\nabla LL = 0$

$$LL = \sum_{i=1}^{N} -\log\left(1 + e^{-y_i(W^T x_i + w_0)}\right)$$

$$\nabla LL = -\left[\sum_{i=1}^{N} \frac{e^{-y_i(W^T x_i + w_0)}}{1 + e^{-y_i(W^T x_i + w_0)}} - y_i \right]$$

$$\sum_{i=1}^{N} \square \; (-y_i x_j)$$

eg $d=1$, $N=2$, $D \equiv [(1,1), (-1,-1)]$

$$\nabla LL = -\left[\frac{e^{-(w_1 + w_0)}}{1 + e^{-(w_1 + w_0)}}(-1) + \frac{e^{(-w_1 + w_0)}}{1 + e^{(-w_1 + w_0)}}(1)\right.$$

$$\left.\frac{e^{-(w_1 + w_0)}}{1 + e^{-(w_1 + w_0)}}(-1) + \frac{e^{-(-w_1 + w_0)}}{1 + e^{-w_1 + w_0}}(-1)\right] = 0$$

Find w_1 & w_0

17/2 ∴ Difficult to solve

Use 'Gradient Ascent Algorithm', which is an iterative algorithm
- Start with a guess. Try to converge to the solution.

- Find the decision boundary by solving the following

$$\log P(y=1 \mid x) > \log P(y=-1 \mid x)$$

$$\log\left(\frac{1}{1 + e^{-(W^T x + w_0)}}\right) > \log\left(1 - \frac{1}{1 + e^{-(W^T x + w_0)}}\right)$$

$$= \log\left(\frac{1}{1 + e^{W^T x + w_0}}\right)$$

PAGE NO.
DATE / /

- $\therefore \quad 0 > -(W^T x + W_0)$

$$W^T x + W_0 > 0$$

- Logistic Regression ~~also~~ gives a linear decision boundary.
- Logistic Regression gives better accuracy than LDA.

24/8

- Probabilistic classifiers are easy to extend to more than two classes.

→ For more than two classes,

$$y \in \{1, 2, 3 \dots K\}$$

- $\therefore$ y follows a Multinomial distribution.

$$y \sim \text{Multinomial}(p_1, p_2 \dots p_K)$$

such that $0 < p_j < 1$, $\sum p_j = 1$

- $P(y|x) \sim \text{Multinomial}(p_{1x}, p_{2x} \dots p_{Kx})$

such that $0 < p_{jx} < 1$, $\sum p_{jx} = 1$

- Parameters are dependent on value of X.

$$P(y = j | x) = p_{jx} = \frac{e^{\vec{w_j} X + w_{j0}}}{\sum e^{w_y X + w_{y0}}}$$

Training $D \equiv \{(x_1, y_1) \ldots\ldots (x_N, y_N)\}$

Use likelihood estimator.

No. of parameters $= k(d+1)$ No. of W_{ji}

* One of the optimal solutions is that Ws of $+y$ are negative of Ws of $-y$ (like in the case of binary classification, where we directly used only a single set of Ws)

- Likelihood estimator $= LL(D, W)$

$$= \sum \log[P(y_i | W, \vec{x}_i)]$$

$$= \sum_{i=1}^{N} \left(W_{y_i}^T x_i + W_{y_i 0} - \log\left(\sum e^{W_y^T x_i + W_{y0}}\right)\right)$$

This function is concave in W.

* If $F(z)$ is concave in z, then $F(W^T x)$ is concave in W, with X as a constant and W is multidimensional

- Maximize $LL(D, W)$ wrt W
  - Not trivial
  - Use iterative algorithm

PAGE No.
DATE / /

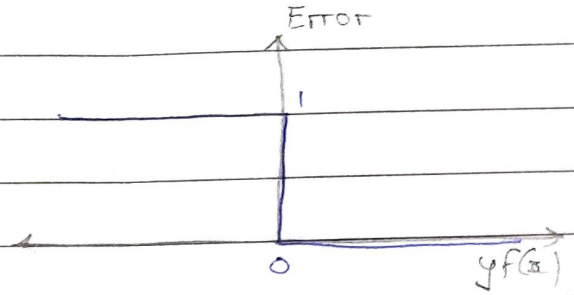
IV Loss Regularization Family Of Classifiers

eg $k = 2$, $y_i \in \{-1, 1\}$

$f(x) = W \cdot X + W_0$

bias ↑

- Classifier :- sign of $f(x)$
- Error = 1 if $y f(x) < 0$
  = 0 otherwise



Training $D \equiv \{(x_i, y_i) \ldots\}$ $1 \le i \le N$

TFT = Ideal value of W

$= \text{argmin} \left(\sum \text{error}\left(y_i, f(x_i | W)\right) \right)$

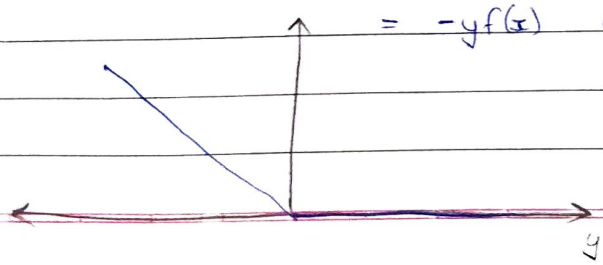
$= \text{arg min} \left(\sum \text{error}\left(y_p (W \cdot x_p + W_0)\right) \right)$

- Error $(y f(x))$ is neither concave nor convex.
  - Line joining any two points does not lie entirely below or above the function.
  - Cannot use concave optimization problem
  - Define new loss functions

→ Perceptron Loss Function

ploss $(y f(x)) = 0$ if $y f(x) \ge 0$

$= -y f(x)$ otherwise



P.T.O.

PAGE NO.
DATE / /

☺ ploss is convex and continuous

☹ ploss is not differentiable

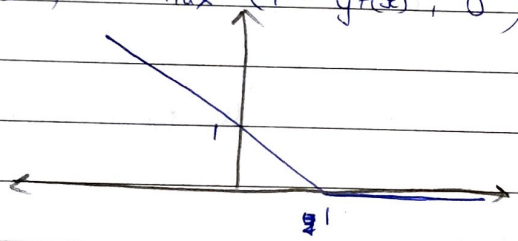
☹ For this classifier, margin of separation is very thin

- $0.00001 \longrightarrow y = 1$
- $-0.00001 \longrightarrow y = -1$

- We want decision boundaries that pass through regions which have low density of points

→ Hinge Loss

- Hinge $(yf(x)) = 1 - yf(x)$ if $yf(x) \leq 1$
  $= 0$ otherwise
- Hinge $(yf(x)) = \max(1 - yf(x), 0)$

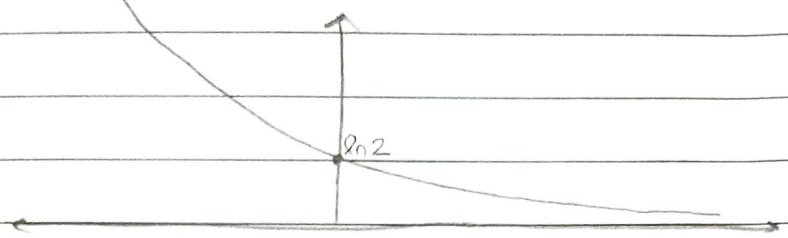


☺ • The shift of 1 unit boosts generalizability.
Error is zero only if $yf(x)$ is significantly more than zero

→ Square Hinge Loss
$= \text{Hinge}^2(yf(x))$

→ Logistic Loss

$$\text{logloss}(yf(x)) = \log\left(1 + e^{-yf(x)}\right)$$



$\sim -z$ for $z \to -\infty$

☺ - Convex in $yf(x)$
- Differentiable
- Gives a good margin. - does not overfit.

9/8

A] Regularizers

- Prevent overfitting

eg 1-dimensional data :- $D = \{(x_1, y_1) \dots (x_N, y_N)\}$ $d = 1$
Use a linear function. It doesn't overfit.

- If above data is synthetically expanded to 9-dimensional data,

$$D' \equiv \{(\vec{x}_1, y_1) \dots (\vec{x}_N, y_N)\}$$

where, say, $\vec{x}_i = (x_i, x_i^2, x_i^3 \dots x_i^9)$

~~Linear~~ Even linear functions will overfit on this 9-dimensional space.

PAGE NO.
DATE / /

- Example of Regularizer :- $R(w) = \left(\sum_{j=1}^{d} |w_j|^{\gamma} \right)^{1/\gamma}$ ditch

Training objective :- Minimize wrt w, w_0 :- $C\left(\sum_{i}^{N} loss(y_i, f(x_i)) \right) + R(w)$

- Note :-
  - Bias term (W_0) is not regularized.
  - C can be chosen depending on how much weight you want to give to $R(w)$
  - Value of γ :-
    - $\gamma = 0$:- Only no. of non-zero W_i's are minimized. L0
    - $\gamma = 1$ L1
    - $\gamma = 2$ L2
      - Used most often.
      $$R(w) = \sum_{i}^{d} w_i^2$$
  - Like the loss function, regularizer is also a convex function in W_i ☺

B] Algorithms for optimizing convex functions.

Assumption :- Solving $\nabla F(w) = 0$ is not easy.

Objective function :- $F(w) = C\left(\sum loss(y f(x)) \right) + R(w)$

- Four types of algorithms :- 0^{th}, 1^{st}, 2^{nd}, semi 2^{nd}

PAGE No.
DATE / /

1. Zeroth order :- ✗ Gradient Computation
   ✓ Applicable to non-differentiable function.
2. First order :- ✓ Gradient Computation
   ✓ Applicable to functions that are not twice differentiable (eg - Square Hinge loss)
3. Second order :- ✓ Hessian Computation
   ← analogous to second differentiation in multi-dimensional data.
4. Semi-second order :- Approximate the Hessian.

1/8

A Zeroth Order :- Only $f(w)$ is known

→ Calculate $f(w)$ for many many many points, $\Delta w = \epsilon$ apart, and return the minimum $f(w)$ found.
Toferance of ϵ is applicable

☺ Can be applied for any function, even non-convex functions
☹ Duh.

→ Dichotomous Binary Search
a and b are on opposite sides of minima.
while ($|b-a| \geq \epsilon$):
Pick two points w_1 and w_2, ϵ apart, in the middle of a and b
If ($F(w_1) > F(w_2)$): $a = w_1$
Else : $b = w_2$

No. of iterations = No. of function calls $= T/2$

If T = no. of evaluations, $(b-a)(0.5)^{T/2} = \epsilon$

$\Rightarrow T = \frac{2 \ln \epsilon}{\ln (0.5)}$

- This algorithm surely terminates, and gives correct minimum
- Cannot be used in higher dimensions

B First order methods

$f(w)$ and grad $f(w)$ are known

→ 1D ~~function~~ data :- Modified Bisection Algorithm

while $|b-a| \geq \epsilon$:-

Choose $w \in (a,b)$

If $f'(w) = 0$:- return w

> 0 :- $b = w$

< 0 :- $a = w$

→ First order methods for higher dimensional data

- Taylor Series expansion

$$F(w) = F(w^0) + \nabla F(w^0)^T (w - w^0) + |w - w^0| \left(\alpha(w^0, w)\right)$$

such that $\alpha(w^0, w) \to 0$ as $w^0 \to w$

Hence, every differentiable function can be linearly approximated in the neighbourhood of any point

$$\tilde{F}(w) = F(w^0) + \nabla(F(w^0))^T (w - w^0)$$

Write $w = \lambda \vec{e} + w^0$, where $\vec{e}$ is a unit vector in d dimensions and $\lambda \in \mathbb{R}$, $\lambda \geq 0$, λ is small.

$$\tilde{F}(w^0 + \lambda\vec{e}) = F(w^0) + \lambda \underbrace{\nabla F(w^0)^T \cdot \vec{e}}_{\leq 0}$$

Move in direction opposite to gradient:- Choose $\vec{e}$ such that $\nabla F(w^0)^T \vec{e} \leq 0$

Then you will get $F(w) \leq F(w^0)$

Algorithm w^0 = arbitrary initial value

while $(|\nabla F(w^t)| \geq \epsilon)$ {

i) Find $\vec{e}$ s.t. $\nabla F(w^t) \cdot \vec{e} \leq 0$

ii) $\lambda^* = \arg\min [F(w^t + \lambda\vec{e})]$ ..... 1D convex optimization

iii) $w^{t+1} = \lambda\vec{e} + w^t$.

→ Gradient Descent Algorithm :-

- Choose $\vec{e} = -\nabla F(w^0)$
- Pick λ as any small positive value. (Avoids calculation of λ^*)

PAGE NO.
DATE / /

- Applying for logistic loss minimization,

$$F(w) = \sum_{i=1}^{N} \log\left(1 + e^{-wx_i y_i}\right)$$

$$\nabla F(w) = \sum_{i=1}^{N} \frac{e^{-wx_i y_i}}{1 + e^{-wx_i y_i}} (-x_i y_i) \quad \text{---- vector}$$

→ Off-the-shelf optimization algorithm :- 'LBFGS'
- The loop is implemented by the algorithm.
You only need to evaluate $F(w)$ and $\nabla F(w)$ at any given w.

→ Stochastic Gradient Descent Algorithm.
Instead of calculating precise gradient, calculate gradient using only a subset of data points.
This subset is called a 'batch'
- Makes execution time lower.

C Second Order Methods

- For a twice differential function, Hessian is a $d \times d$ matrix where d is the dimensionality of data.

$$H_{ij} = \frac{\partial^2 F}{\partial w_i \partial w_j}$$

PAGE No.
DATE / /

- For any twice differentiable function, second order Taylor expansion

$$F(w) = F(w^0) + \nabla(F(w^0))^T (w - w^0) + \frac{1}{2}(w - w^0)^T H_F(w^0)(w - w^0) + \|w - w^0\|^2 [\alpha(w^0, w - w^0)]$$

where $\alpha \to 0$ as $w^0 \to w$