2nd Method to solve for the separating hyperplane in the realizable case:

We know that $y_i \langle \underline{w}, \underline{x}_i \rangle \geq 0$ for correct classification.

So, if any example is incorrectly classified, it would give

$$\underbrace{y_i \langle \underline{w}, \underline{x}_i \rangle}_{\text{LHS.}} < 0 \quad \Rightarrow \text{incorrect classification}$$

We design an iterative method

$$\underline{w} \leftarrow \underline{w}^{(0)}$$

We update $\underline{w}$ in steps (iterations)

Scan the examples $(x_i, y_i)_m$

If the i^{th} example is incorrectly classified,

$$y_i \langle \underline{w}^{(t)}, \underline{x}_i \rangle < 0$$

We apply an update step proposed by Rosenblatt $\quad \underline{w}^{(t+1)} \longleftarrow \underline{w}^{(t)} + y_i \underline{x}_i$

This update step will increase the value of $y_i \langle \underline{w}^{(t)}, x_i \rangle$

LHS. after the update

$$y_i \langle \underline{w}^{(t+1)}, \underline{x}_i \rangle = y_i \langle \underline{w}^{(t)} + y_i \underline{x}_i, \underline{x}_i \rangle$$

$$= y_i \langle \underline{w}^{(t)}, \underline{x}_i \rangle + y_i y_i \underline{x}_i^T \underline{x}_i$$

$$= y_i \langle \underline{w}^{(t)}, \underline{x}_i \rangle + \underbrace{y_i^2 \|\underline{x}_i\|^2}_{\text{always +ve}}$$

Perceptron algorithm

Linear hyperplanes.

This is the Perceptron Algorithm.

$\underline{w}^{(0)} \leftarrow (0, \ldots 0)$

for $t = 0, 1, 2 \ldots$

$i{:}49$

if $\exists i$ such that $y_i \langle \underline{w}^{(t)}, x_i \rangle \leq 0$

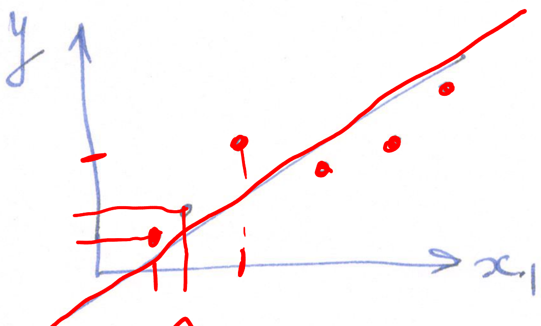
$\underline{w}^{(t+1)} \leftarrow \underline{w}^{(t)} + y_i \underline{x_i}$

else

return $\underline{w}^{(t)}$

The algorithm is guaranteed to converge but only for the Realizable case.

Linear Regression



$(x_i, y_i)\ i=[1..m]$

$\underline{x} = [x_1]$ in the 1D case.

$y = w_0 + w_1 x_1$

y depends linearly on x.

A linear hypothesis can be written as

$$\underbrace{h(x)}_{h_{\underline{w}}(\underline{x})} = \underbrace{\langle \underline{w}, \underline{x} \rangle}_{\text{homogeneous linear map.}}$$

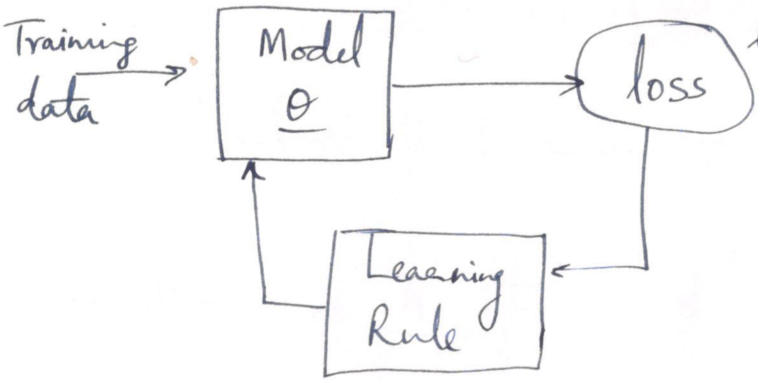
Augmented vectors

Loss function

$\ell(h, (\underline{x}, y)) = \;?$

5 x4 loss 1 $\rightarrow$ $(h(\underline{x}) - y)^2$ squared error loss

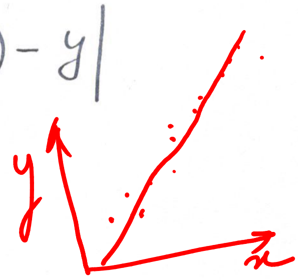
x6 loss -1 reward $|h(x) - y|$ absolute error loss.



$h(x) = \langle w, x \rangle$

$$\ell(h,(x,y)) = (h(x)-y)^2$$

$$|h(x)-y|$$



Empirical Risk Minimization (ERM Rule)

Emp. Risk is expected loss over the m training examples.

Non realizable

Risk. $$L_S(h) = \underbrace{\left(\frac{1}{m}\right)\sum_{i=1}^{m} \left(h(x_i) - y_i\right)^2}_{\text{E Risk.}}$$

L_S

S : Sample (Training set)

L_D : True Risk

D Data generating distribution

Objective: To solve the ERM problem

$$w^* = \arg\min_{w} \underbrace{L_S(h)}_{\text{E Risk.}} = \arg\min_{w} \frac{1}{m}\sum_{i=1}^{m} \left(\langle w, x_i\rangle - y_i\right)^2$$

Differentiating w.r.t w and equating to zero.

$$\frac{2}{m}\sum_{i=1}^{m} \left(\langle w, x_i\rangle - y_i\right) x_i = 0$$

$$\sum_{i=1}^{m} \left(x_i^T w - y_i\right) x_i = 0$$

$x_i x_i^T$

$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ 3×1 $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$ 1×3 $= 3\times3$ outer product matrix

$$\underbrace{\sum_{i=1}^{m} x_i x_i^T}_{A} w = \underbrace{\sum_{i=1}^{m} y_i x_i}_{b}$$

Writing in the form $\underline{A}\;\underline{w} = \underline{b}$

$A = \sum_{i=1}^{m} \underline{x_i}\,\underline{x_i}^T$ outer product of vectors.

$\underline{w} = \underline{A}^{-1}\,\underline{b}$ if $\underline{A}$ is invertible

$A = VDV^T$

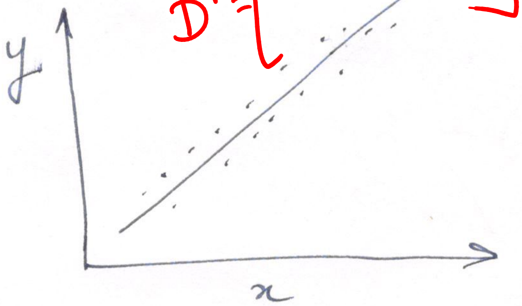
$\begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}$ $\quad D^{+} = \begin{bmatrix} 1/d_1 & & & \\ & 1/d_2 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}$

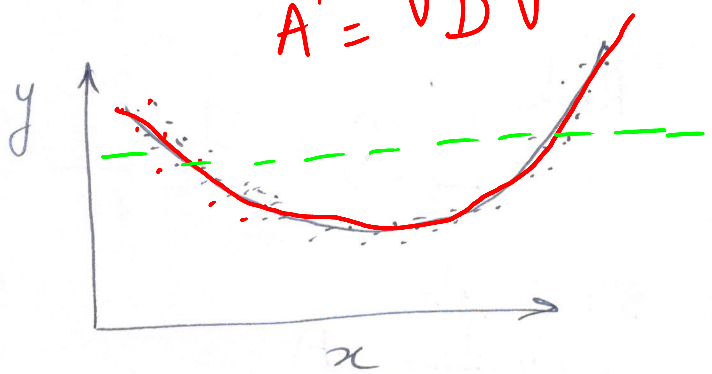
else

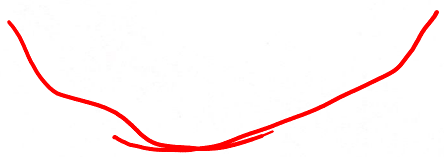
$\underline{w} = \underline{A}^{+}\,\underline{b}$

where A^{+} is the pseudo inverse.

$A^{+} = VD^{+}V^T$







A linear function is not a good fit.

Polynomial Regression

Assuming that x is a scalar

$\mathcal{H} =$ linear fn., Quadratic, Cubic, quartic, n??

$y = \underline{p(x)} = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots + a_n x^n$

y depends on x through a polynomial function.

$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$ Unknown.

We use a trick $\quad x \to \underline{X} = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^n \end{bmatrix}$ $\quad \underline{w} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$

$p(x) = \langle \underline{X}, \underline{w} \rangle$ $\quad \mathbb{R} \to \mathbb{R}^{n+1}$

After transformation of the input feature, we can estimate the polynomial mapping of the input $\underline{x}$ to y by using $\underline{\text{linear regression}}$:
$x \in \mathbb{R}$.

$P(y \in +1) = p_1$

$P(y \in -1) = 1 - p_1$

Linear regression maps $\underline{x}$ to a scalar y.

$$\underline{x} \xrightarrow[\text{linear}]{\langle \underline{w}, \underline{x} \rangle} y \xrightarrow{\text{squash}} \underbrace{[0,1]}_{\text{probability}}$$

$\underline{y \in \mathbb{R}}$.

If we can ~~restrict~~ squash the output ~~in~~ the range $[0,1]$, it can be interpreted as probability.

If we consider the task of binary classification (target $y \in \{0,1\}$), we can interpret the squashed output of linear regression as the probability of input $\underline{x}$ taking the class label $y=1$.

$$\underline{x} \xrightarrow[\text{linear}]{\langle \underline{w}, \underline{x} \rangle} z \xrightarrow[\text{squashing function}]{\phi = \frac{1}{1+\exp(-z)}} \phi(z)$$

$z \in \mathbb{R}$

squashing / sigmoid

$$\phi(z) = \frac{1}{1+\exp(-z)}$$

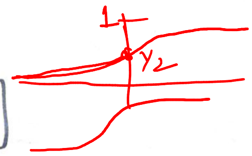
$z = 0$: $\frac{1}{2}$

$z \to \infty$: 1

$z \to -\infty$: 0

$z \in \mathbb{R}$.

$\phi(z) \in [0,1]$



$\phi(z)$ is interpreted as the probability that $\underline{x}$ takes label $y=1$

$1-\phi(z)$ " " $\underline{x}$ takes label $y=0$

By squashing the linear regression output, we have a $\underline{\text{classifier}}$ which gives probability of the $\underline{\text{class label}}$ for binary classification task.

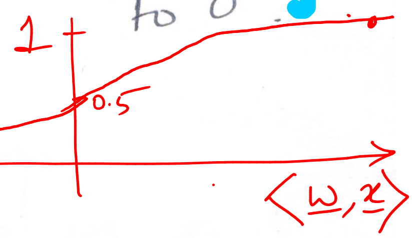
$\to \{1, 0\}$

$\{1, -1\}$

The Hypothesis class for a logistic regression model is given by

$$H_{sig} = \{ \underline{x} \longmapsto \phi_{sig}(\langle \underline{w}, \underline{x} \rangle) : \underline{w} \in \mathbb{R}^d \}$$

If $\langle \underline{w}, \underline{x} \rangle$ is large, small $\phi_{sig}(\langle \underline{w}, \underline{x} \rangle)$ is close to 1. to 0



$h_w(x) \in [0, 1]$ $\boxed{h_w(x) = \phi(\langle \underline{w}, \underline{x} \rangle).}$

Loss formulation

for $y=1$ $h_w(x)$ large — What we expect for correct prediction

GT

$1 - h_w(x)$ should be small for correct for $y=1$

loss

for $y=0$ $h_w(x)$ small. ✓ OK.

GT

$1-h_w(x)$ is large OK.

$h_w(x)$ should be small. for correct for $y=0$

Simplifying the loss for $y=1$

$$1 - h_w(x) = 1 - \frac{1}{1+\exp(-\langle \underline{w}, \underline{x} \rangle)}$$

$$= \left(\frac{\exp(-\langle \underline{w}, \underline{x} \rangle)}{1+\exp(-\langle \underline{w}, \underline{x} \rangle)} \right) \times \frac{\exp(\langle \underline{w}, \underline{x} \rangle)}{\exp(\langle \underline{w}, \underline{x} \rangle)}$$

$$= \frac{1}{1+\exp(\langle \underline{w}, \underline{x} \rangle)}$$

Loss for $y=0$ $\quad h_w(x) = \dfrac{1}{1+\exp(-\langle \underline{w}, \underline{x} \rangle)}$

$y \in \{0,1\}$ 15

<u>Loss for $y=0$ / $y=-1$</u>

$$\left(\frac{1}{1+\exp(-\langle \underline{w}, \underline{x}\rangle)}\right)$$

<u>Loss for $y=1$</u>

$$\left(\frac{1}{1+\exp(\langle \underline{w}, \underline{x}\rangle)}\right)$$

If we <u>recode</u> the target $\underline{y \in \{-1, 1\}}$ instead of $y \in \{0,1\}$, we can unify the two losses for $y=-1$ and $y=1$

<u>Unified loss</u> $$\frac{1}{1+\exp(y\langle \underline{w}, \underline{x}\rangle)}$$

Captures both the loss terms.

<u>ERM</u>

The objective function to be minimized can be written as

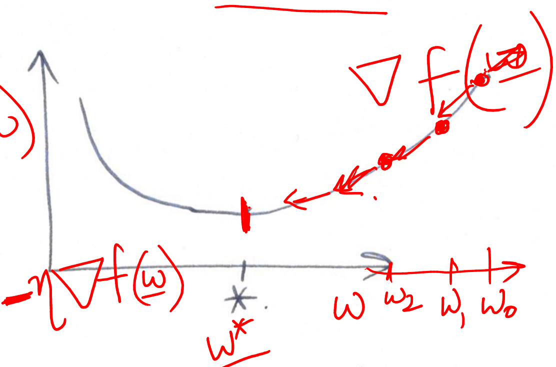
$$\min_{\underline{w}} \frac{1}{1+\exp(y\langle \underline{w}, \underline{x}\rangle)} \quad \text{or} \quad \min_{\underline{w}} 1+\exp(-y\langle \underline{w}, \underline{x}\rangle)$$

<u>Log of the loss</u>

$$\boxed{\min_{\underline{w}} \log\left(1+\exp(-y\langle \underline{w}, \underline{x}\rangle)\right)}$$

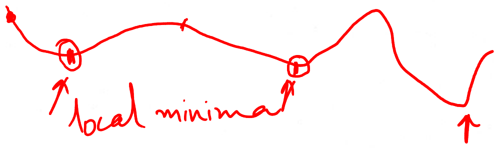
The ~~loss function~~ is <u>convex</u>.

Convex have only the global minimum



$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla f(w)$

Use <u>gradient descent</u>



Half space classifier	Realizable Case	0-1 loss	Linear Prog.
Linear Regression	Non Realizable Case	Squared Error loss	Pseudo Inverse
Logistic Regression	Non Realizable Case	Logistic Regression loss	Gradient Descent