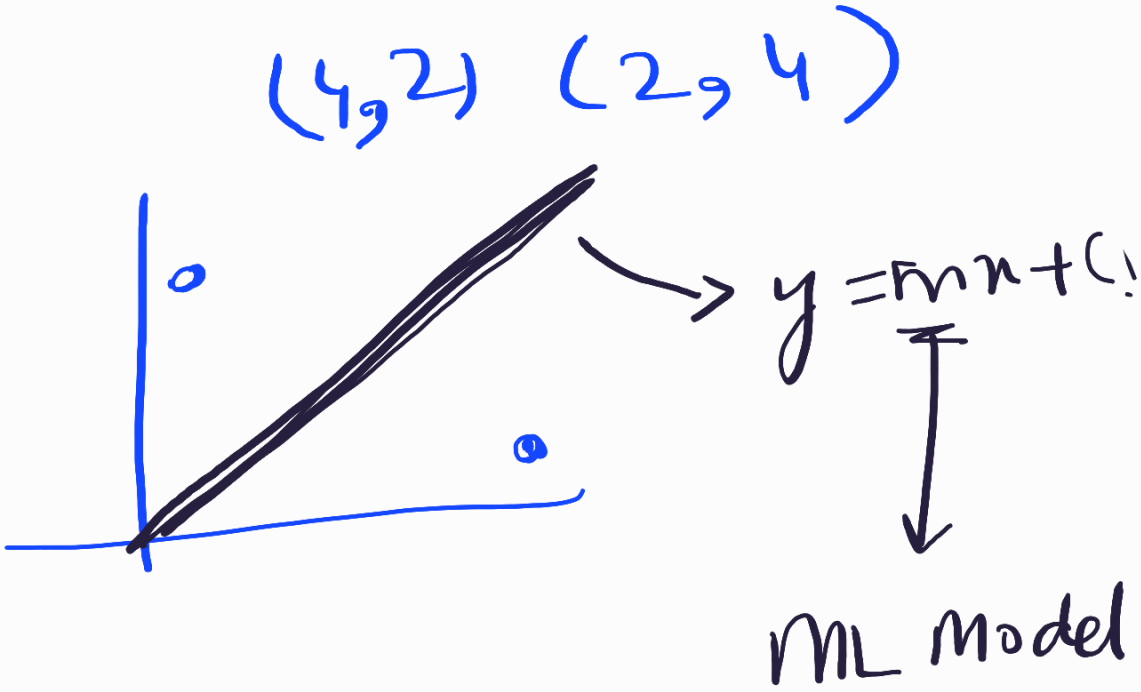
AI Theory

Machine Learning

Recall:



For computer, ML Model is simply a mathematical model.

Goal: find a "predictor"

Application : Spam classification

$D(x, y) \longrightarrow$ **Learner**

x : email

y : spam / N Spam

$\downarrow$

$x \longrightarrow$ **Predictor** $y = mx$

$\downarrow y$

Learner $\rightarrow$ learn trend from data

Predictor $\rightarrow$ From learner, extend to new data

Types of Prediction Tasks

① Binary Classification → Discrete Values

Multiclass

② Regression → Continous Values

③ Ranking → y : permutation

1, 2, 3 → 2 3 1

④ Structured Prediction

Ex : translator

La casa blu → the blue house

Example

House Data :

No Rooms	Bathrooms	Garden	.....

Data D

$$D \rightarrow D_{train}$$

$$D \rightarrow D_{test}$$

Data contains examples.

each row

How to validate model values?

Splitting the data

D_{train}, D_{test}

Validate the model on D_{test}, then give it unseen values.

Feature Extraction

Given input x, output a set of (Feature_name, Feature value) pairs.

abc@gmail.com

Feature extractor

length > 0	1
fracOfAlpha	0.85
contains(@)	1
ends with .com	1
" " .org	0

our feature extractor thinks any email ending with .org is spam.

Feature extractor converts into ML-processable features.

<u>Mathematically</u>:

$$\phi(x) = [\phi_1(x) \ldots\ldots \phi_d(x)]$$

where $\phi(x) \in \mathbb{R}$ is a point in a high-dimensional space.

Weight Vector

For each feature j, have a real number w_j representing contribution of FEATURE TO PREDICTION.

$$y = \underset{\substack{\uparrow \\ \text{weights}}}{w} x$$

But we also studied

$$y = mx$$

How do these not contradict

one another?

$$y = \vec{w} \cdot \vec{x}$$

ex:

$$\begin{array}{c} w_1 \\ w_2 \\ w_3 \end{array} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \quad \begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \\ \begin{bmatrix} 5 & 2 & 9 \end{bmatrix} \end{array}$$

<u>dot product</u>

results in <u>scalar value</u>.

Linear Predictors

Weight vector

$w \in R^d$

Feature vector

$\phi(x) \in R^d$

$$w \cdot \phi(x) = \sum_{j=1}^{d} w_j \cdot \phi_j(x)$$

dot product

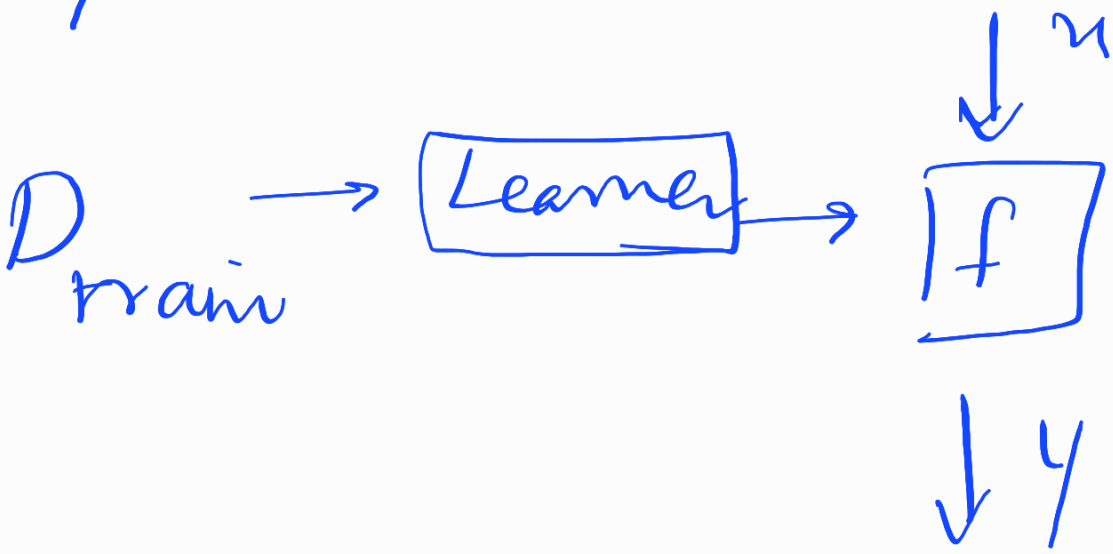
(Binary) Linear Classifier

$$f_w(x) = \text{sign}(w \phi(x)) = \begin{cases} +1, & > 0 \\ -1, & < 0 \\ ? & = 0 \end{cases}$$

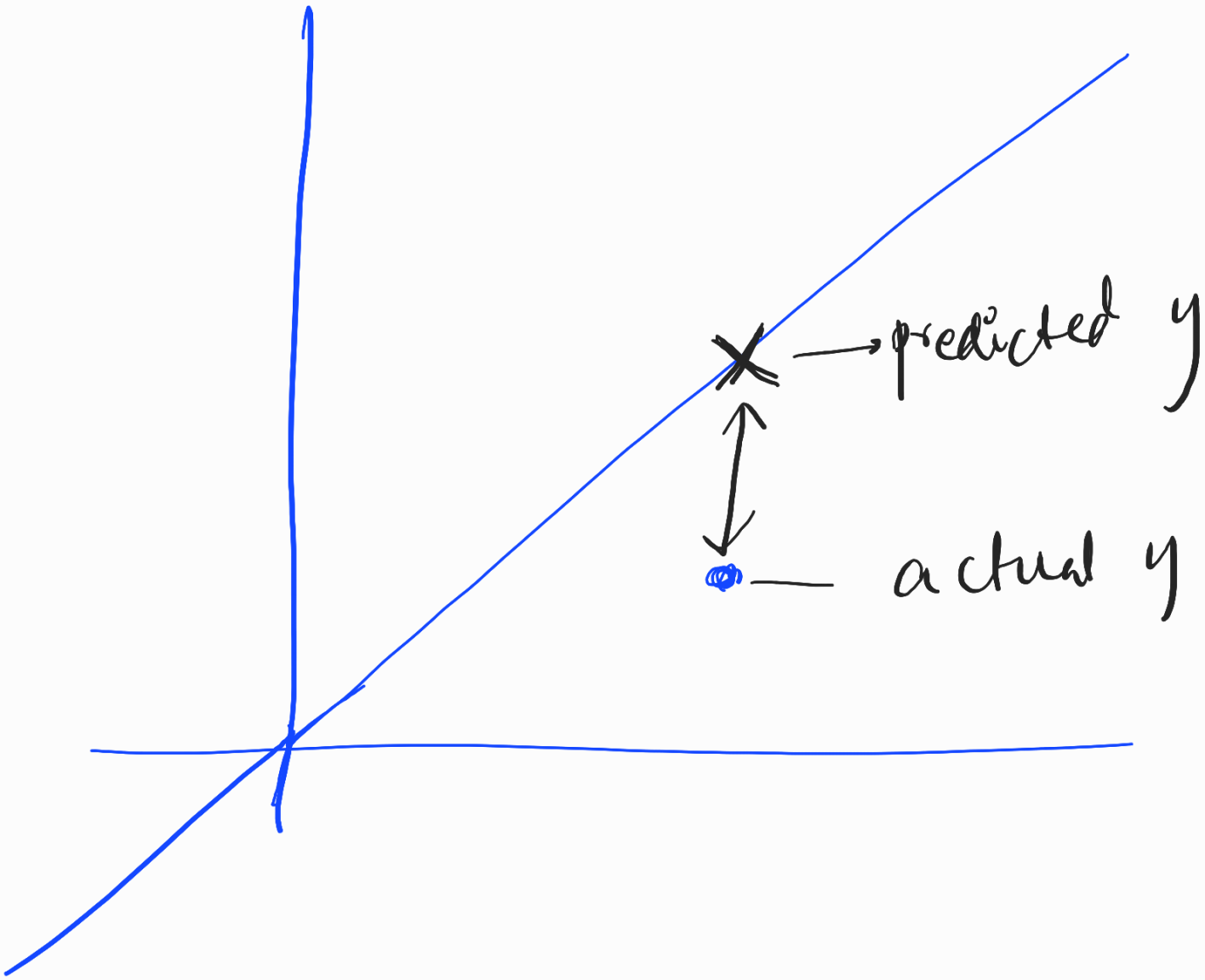
Force outputs in +1, −1

Learner

Framework



Optimization Problem → optimization Algo



This graph leads to another concept:

Loss function:

Diff b/w predicted & actual.

Score :→ (in the context of Binary classification)

Score on an example (x, y) is $w \cdot \phi(x)$, how confident we are for $+1$.

Margin

How correct we are.

↳ $(w \cdot \phi(x))\, y$

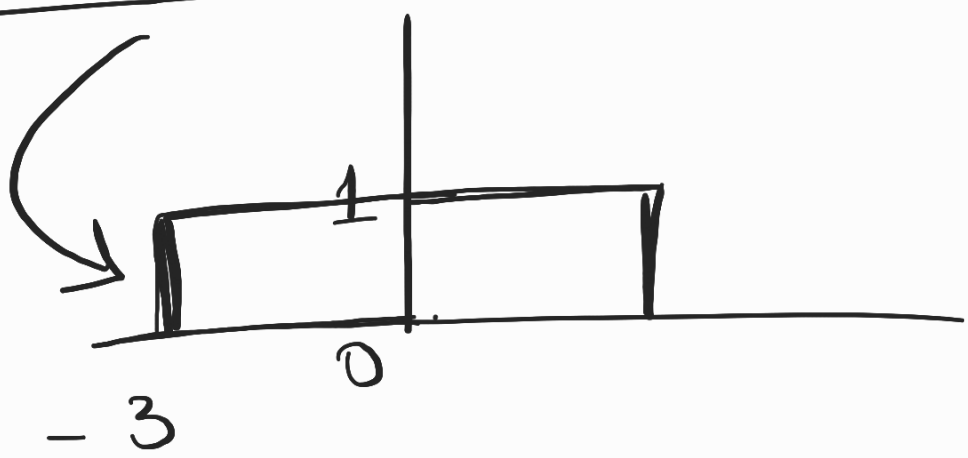
Zero-one loss

$$\text{Loss}_{0-1}(x, y, w) = 1\left[f_w(x) \neq y\right]$$

boolean 0/1

$$= 1\left[\text{margin} \leq 0\right]$$

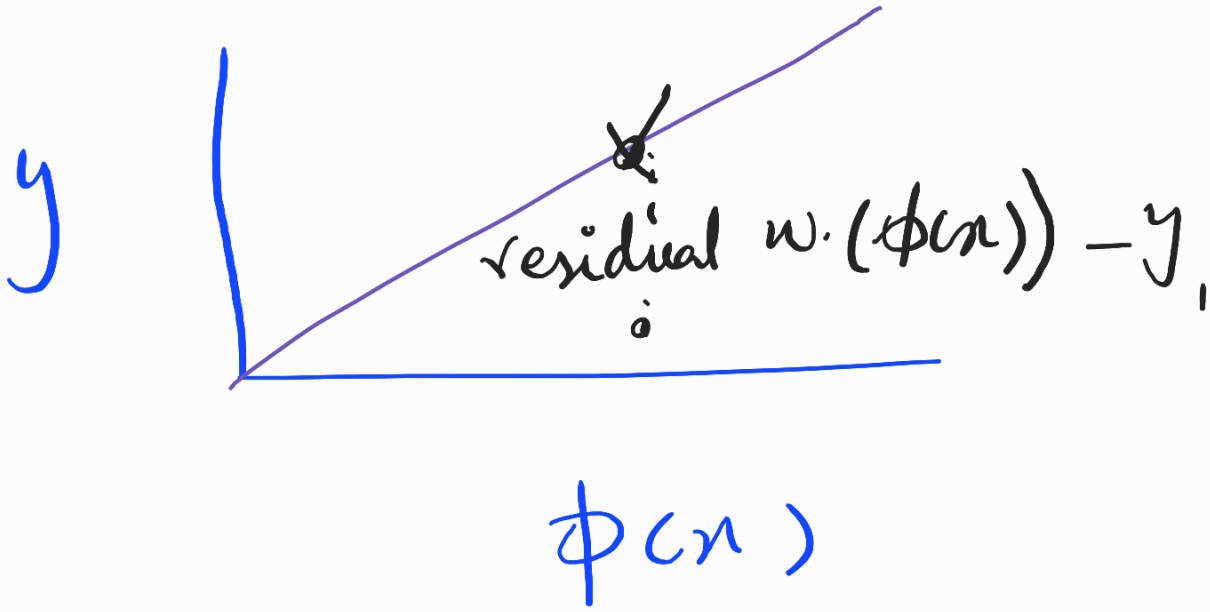
Discrete



Used for binary class.

Linear Regression

$$y = f_w(x) = w \cdot \phi(x)$$



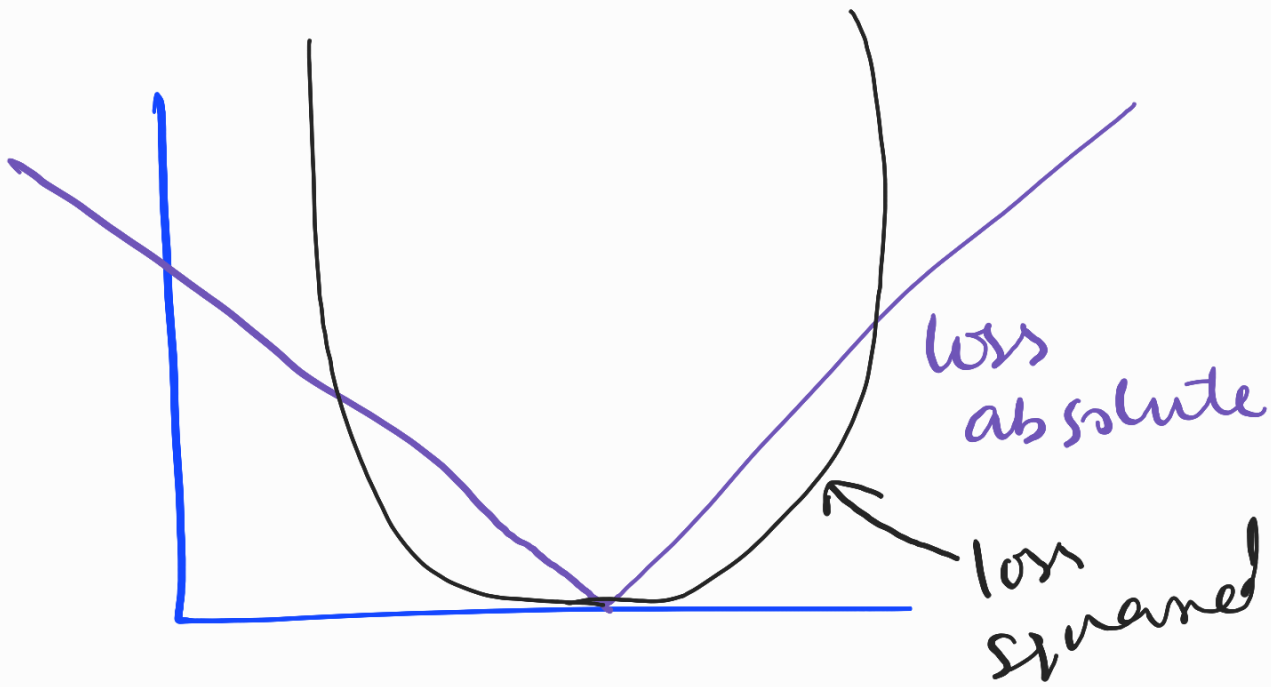
Residual:

The amount $\underline{w(\phi(x)) - y}$

by which Prediction overshoots the actual value.

$$\text{Loss}_{\text{squared}} = (\hat{y} - y)^2$$

$$\text{Loss}_{\text{absolute}} = |\hat{y} - y|$$



Recall steps of ML :

1- Initialize $\vec{w}$.

2- For i in epochs :

2.1 - Calculate $y = \vec{w} \cdot \vec{x}$

2.2 - Calculate Loss

2.3 - $\partial loss / \partial w$

2.4 - update $\vec{w}$ acc. to loss using formula:

$$w = w - \alpha \frac{\partial loss}{\partial w}$$

regularization parameter

Loss minimization framework

$$\text{TrainLoss}(w) = \frac{1}{|D_{train}|} \sum_{(x,y) \in D_{train}} \text{Loss}$$

choose $\min_{w \in R^d} \text{TrainLoss}(w)$

Actually:

How to check how to update weights? (decrease / increase)

Tangent represents the direction.

first order derivative

Take derivative of loss function and put values. It gives the trend.

$$\frac{d}{dw} \text{Loss} = (wx - y)^2$$

$$= 2(wx - y) \quad x$$

$$\frac{d}{dw} \text{loss} = 2x(wx - y)$$

This loss is for a linear model using loss MSE.

$$\text{num_epochs} \propto \frac{1}{\text{alpha}}$$