Ex 1: $x_t \in \mathbb{R}^d$

Each x_t represents a 128×128 pixel image ($d = 128^2$).

want to use $\{x_t : t = 1, 2, \dots, n\}$ (training images) to learn a classifier

classifier $\xrightarrow{\text{def}}$ $f : \mathbb{R}^d \longrightarrow \{+1, -1\}$

dog cat

if x contains a cat, we hope $f(x) = -1$

x contains a dog, hope $f(x) = +1$.

use $\{x_t : t = 1, 2, \dots, n\}$ to learn f.

$$\begin{cases} (x_1, y_1) \\ (x_2, y_2) \\ \vdots \\ (x_n, y_n) \end{cases}$$

$\Rightarrow$ Actually not because we still need label $\{(x_t, y_t) : t = 1, \dots, n\}$

To each training image, we have a label $y_t \in \{+1, -1\}$.

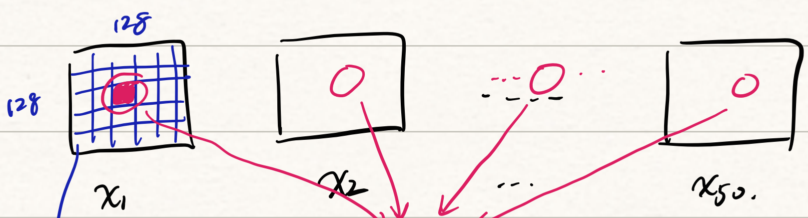
↳ $\mathcal{D} = \{(x_t, y_t) : t = 1, 2, \dots, n\}$.

↳ use $\mathcal{D}$ to learn a classifier f. $\Rightarrow$ $f : \mathbb{R}^d \longrightarrow \{+1, -1\}$.

Ex of a rule for classifier f. $d = 128^2 = 2^{14}$. pixels (dim of data)

$n = 50$ images

Each $\boxed{\text{pixel value}}$ is an integer in $\{0, 1, \dots, 255\}$.



$\in \{0, 1, \dots, 255\}$.

there may exist one pixel that each value is different.

$x_{ti} \rightarrow i^{th}$ pixel in t^{th} image x_t $\quad x_t \rightarrow \begin{pmatrix} x_{t1} \\ \vdots \\ x_{td} \end{pmatrix}$.

One classification Rule: (may not reasonable)

for x', $f(x') = \{ y_t$, if $x_{ti} = x_i'$ for some $t = 1, \dots, n$.

-1, else.

Rmk: f classifiers the training set perfectly! → memorized the training sets.

$\Rightarrow$ Training loss $= 0$.

Qi: Is this classifier "GOOD"?

↳ What is good? → Answer is not good.

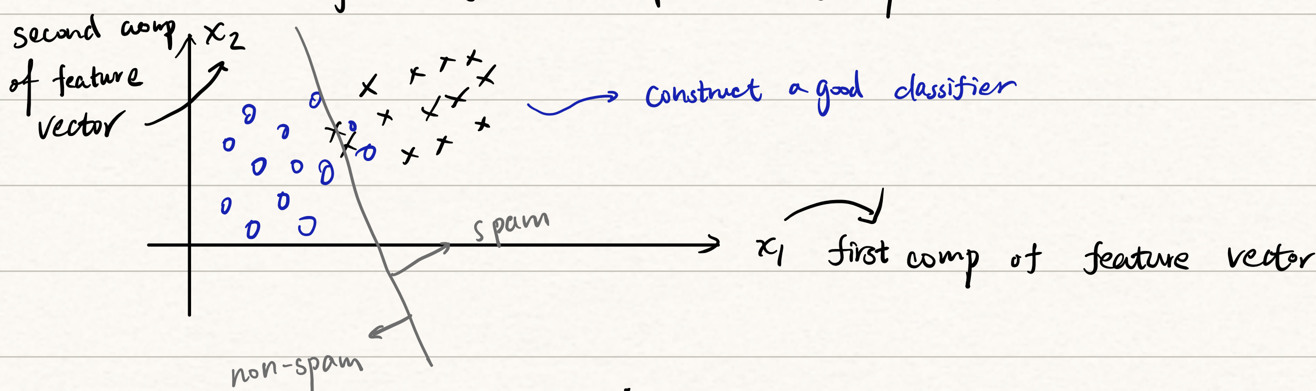
main idea in machine learning is to generalize well.

But this classifier f does not generalize well to new data.

Ex. (x_t, y_t) $\quad x_t \in \mathbb{R}^2 \quad y_t \in \{\text{spam}, \text{not spam}\}$

↳ email example

$\begin{cases} x_{t1}: \text{ \# of words in Email } t \text{ are "loan"} \\ x_{t2}: \text{ \# of words in Email } t \text{ are "prize"}. \end{cases}$



→ construct a good classifier

Supervised Learning $\Rightarrow$ x_t has label.

Notation

- x_t: training samples / feature vectors
- d: # of features / dimension in x_t.
- n: # of training samples.
- y_t: labels ⇝ multi classification. $y_t \in \{1, \dots, K\}$. / binary classification

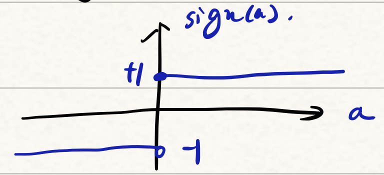
---

Linear Classifiers & training error.

Def: A linear classifier $f: \mathbb{R}^d \longrightarrow \{-1, +1\}$.
(linear classifier)

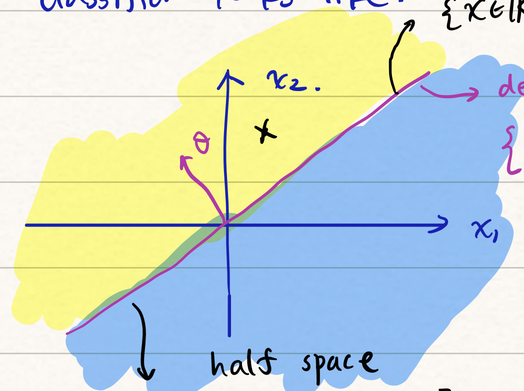
$$f_\theta(x) = \text{sign}\{\langle x, \theta \rangle\}.$$

① $\text{sign}(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0. \end{cases}$



② $\langle x, \theta \rangle = \theta^T x = \sum_{i=1}^{d} \theta_i x_i$

linear classifier looks like: half space. $\{x \in \mathbb{R}^2 : \langle x, \theta \rangle \geq 0\}. \Rightarrow f_\theta(x) = +1$



→ decision boundary. $\{x \in \mathbb{R}^2 : \langle \theta, x \rangle = 0\}$.

half space
$\{x \in \mathbb{R}^2 : \langle x, \theta \rangle \leq 0\} \Rightarrow f_\theta(x) = -1$.

Def
(training Error)

$$\hat{E}(\theta) = \frac{1}{n} \sum_{t=1}^{n} \text{Loss}(y_i, \boxed{f_\theta(x_i)}) \in [0,1].$$

label. ↳ $f(x_i; \theta)$

classifier

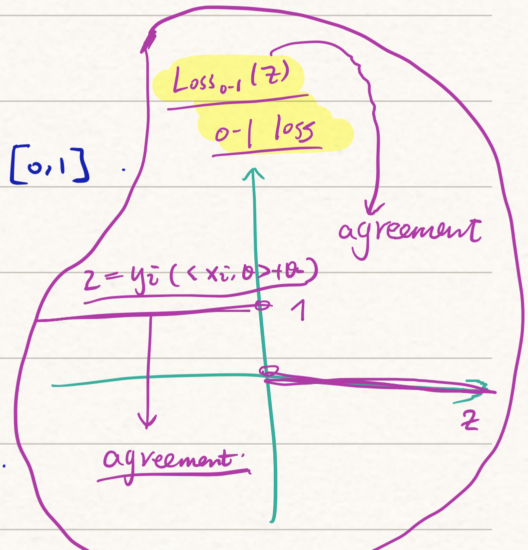
$$\text{Loss}(y, \hat{y}) = \mathbb{1}\{y \neq \hat{y}\}. = \begin{cases} 1, & y \neq \hat{y} \\ 0, & y = \hat{y}. \end{cases}$$

↳ 0-1 loss function.

Indicative function

NOT Expectation (Actually just a function).

Many loss function. $\Rightarrow$ { Entropy loss func, Hinge loss, Least square loss



☆ Def: linearly separable-$\mathcal{D}$

A dataset $\mathcal{D} = \{(x_t, y_t) \mid t = 1, 2, \ldots n\}$ is linearly separable

$\Leftrightarrow \exists\, \theta \in \mathbb{R}^d$ s.t $y_t \langle \theta, x_t \rangle \geq 0$ for $\forall\, t = 1, 2, \ldots, n$

↳ labeled correctly. (exists a hyperplane).

↳ perceptron

Non-linearly separable $\Rightarrow$ SVM.

Rmk: $\mathcal{D}$ is linearly separable $\Leftrightarrow$ $\exists\, \theta \in \mathbb{R}^d$. s.t $\hat{E}(\theta) = 0$. (training ~~error~~)

if $f_\theta(\cdot)$ is a linear classifier.

Rmk: for linearly classifier, i.e., $f_\theta(x) = \text{sign}(\langle \theta, x \rangle)$.

the decision boundary passes through origin

(we don't put the OFFSET!)

$\{x \in \mathbb{R}^d : \langle \theta, x \rangle = 0\} \to$ pass through the origin.

Perceptron Algorithm (Rosenblatt. 1960s). $\boxed{\text{linearly separable}}$?

1). Arbitrarily initialize θ to be $\theta^{(0)}$

2). cycle through all n training samples

If $y_t \langle x_t, \theta^{(k)} \rangle \le 0$ (i.e. there is a disagreement

between label y_t & prediction $\text{sgn}\{\langle x_t, \theta^{(k)} \rangle\}$.

Update $\theta^{(k+1)} = \theta^{(k)} + y_t x_t$

Some proof of Perceptron Algorithm

1. Considering $\mathcal{D}$ contains only 1 sample, i.e., $\mathcal{D} = \{(x_1, y_1)\}$.

the simplest case $|\mathcal{D}| = 1$

initialize $\theta^{(0)}$. ① if $y_1 \langle x_1, \theta^{(0)} \rangle > 0 \to$ over!

② if $y_1 \langle x_1, \theta^{(0)} \rangle \le 0$

then we have $\theta^{(1)} = \theta^{(0)} + y_1 x_1$ (Update)

Check: $y_1 \langle x_1, \theta^{(1)} \rangle$

$= y_1 \langle x_1, \theta^{(0)} + y_1 x_1 \rangle$

$= y_1 \langle x_1, \theta^{(0)} \rangle + \boxed{\overset{\to 1}{y_1^2} \langle x_1, x_1 \rangle}$ $\|x_1\|^2$.

$> y_1 \langle x_1, \theta^{(0)} \rangle$.

To conclude, after updating, we have:

$$y_1 \langle \underline{x}_1, \underline{\theta}^{(1)} \rangle \geq y_1 \langle \underline{x}_1, \underline{\theta}^{(0)} \rangle$$

(increased by $\|x_1\|^2$).

More generally, $y_1 \langle \underline{\theta}^{(k+1)}, \underline{x}_1 \rangle = y_1 \langle \underline{\theta}^{(k)}, \underline{x}_1 \rangle + \|\underline{x}_1\|^2$

→ After updating, $y_1 \langle \underline{\theta}^{(k+1)}, \underline{x}_1 \rangle$ increases by $\|x_1\|^2 > 0$.

→ After $\left\lceil \frac{|\langle \theta^{(0)}, x_1 \rangle|}{\|x_1\|^2} \right\rceil$ iterations, it will hold $y_1 \langle \theta^{(k)}, x_1 \rangle > 0$.

☆ In summary, if $|\mathcal{D}| = 1$, then perceptron always converges after $\left\lceil \frac{|\langle \theta^{(0)}, x_1 \rangle|}{\|x_1\|^2} \right\rceil$ # of steps.

⇓

this cannot be simply generalized to $|\mathcal{D}| = k$

because we cannot guarantee other training samples (x_k, y_k)'s $y_k \langle \theta^{(k)}, x_k \rangle$ value → it will be affected!

Perceptron Convergence Theorem (need this to prove)

Notation: (γ-linearly separable).

Def 1: Dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ is γ-linearly separable

if there exists $\underline{\theta}^* \in \mathbb{R}^d$, s.t. $y_t \langle \underline{\theta}^*, \underline{x}_t \rangle \geq \gamma > 0 \quad \forall t = 1, 2, \ldots, n$

→ $\gamma = \min_{1 \le t \le n} y_t \langle \underline{\theta}^*, \underline{x}_t \rangle$.

Def 2: $\mathcal{D}$ is R-bounded if $\|\underline{x}_t\| \le R$ for every $t = 1, 2, \ldots, n$.

Rmk: Any dataset $\mathcal{D}$ that is finite is R-bounded. (for some R).

→ take $R = \max_{1 \le t \le n} \|\underline{x}_t\|$ → while countable is not

⇓

the concept of limit ⇔ sup.

⇓

not bounded.

Thm: [Perceptron Convergence Thm]. (Novikoff 1962).

Initialize $\boxed{\underline{\theta}^{(0)} = \underline{0} \in \mathbb{R}^d}$. Assuming that $\mathcal{D}$ is γ-linearly separable & R-bounded. Then, the total # of updates of the Perceptron alg K satisfies

not easy to satisfy

easy to satisfy

$$K \leq R^2 \frac{\|\underline{\theta}^*\|^2}{\gamma^2}$$

$\Rightarrow$ give the upper bound of the # of convergence.

to get the upper bound of iteration

where $\underline{\theta}^*$ is the vector in the def. of γ-linearly separable.

Rmk: if γ is big $\Leftrightarrow$ good saparability

$\Rightarrow$ K tends to be small

$\Rightarrow$ a few iterations.

☆

Outline:

Classifier $\xrightarrow{f_\theta(\underline{x}) = \langle \underline{x}, \underline{\theta} \rangle}$ Linear classifier (with no offset) $\longrightarrow$ Algorithm $\Rightarrow$ Update Parameter.

$\Downarrow$

Training Error

$$\hat{E}(\theta) = \frac{1}{n} \sum_{t=1}^{n} \text{Loss}(y_t, f_\theta(\underline{x}))$$

- $\boxed{\text{0-1 Loss}}$
- hinge Loss
- entropy Loss (log loss).

$$L_{0\text{-}1}(z) = \begin{cases} z > 0, & 0 \\ z < 0, & 1 \end{cases}$$

$z = y_t \cdot \langle \theta, x_t \rangle$

$\Downarrow$

agreement.

$L_{0\text{-}1}(z) = (1-z)^+$.