Machine Learning

Lecture 4: Linear Classification

王浩

信息科学与技术学院

Email: wanghao1@shanghaitech.edu.cn

本节内容

- Perceptron
- Logistic Regression
- Softmax Regression

Case with $+1, -1$ labels

Example:

•Salary, debt, years in residence, ...	input $\mathbf{x} \in \mathbb{R}^n$
•Approve credit or not	output $y \in \{+1, -1\}$
•True relationship between $\mathbf{x}$ and y	target function f
•Data on customers	data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, b_N)\}$

A simple learning model

- Input vector $\mathbf{x} = [x_1, \ldots, x_d]^T$

- Given importance weights to the different inputs and compute a "Credit Score"

$$\text{"Credit Score"} = \sum_{i=1}^{d} w_i x_i.$$

- Approve credit if the "Credit Score" is acceptable

$$\text{"Approve Score"} = \sum_{i=1}^{d} w_i x_i > \text{threshold. ("credit" is good)}$$

$$\text{"Deny Score"} = \sum_{i=1}^{d} w_i x_i < \text{threshold. ("credit" is bad)}$$

- How to choose the importance weights w_i

input x_i is important	$\rightarrow$	large weight $\lvert w_i \rvert$
input x_i beneficial for credit	$\rightarrow$	positive weight $w_i > 0$
input x_i detrimental for credit	$\rightarrow$	negative weight $w_i < 0$

A simple learning model

"Approve Score" $= \sum_{i=1}^{d} w_i x_i >$ threshold. ("credit" is good)
"Deny Score" $= \sum_{i=1}^{d} w_i x_i <$ threshold. ("credit" is bad)

can be written formally as

$$f(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + w_0\right)$$

The "bias weight" w_0 corresponds to the threshold when the neuron is triggered

A simple learning model

Input vector $\mathbf{x} = [x_1, \ldots, x_n]^T$, want to find

$$f(\mathbf{x}) = \text{sign}\left(\left(\sum_{j=1}^{n} w_j x_j\right) + w_0\right)$$

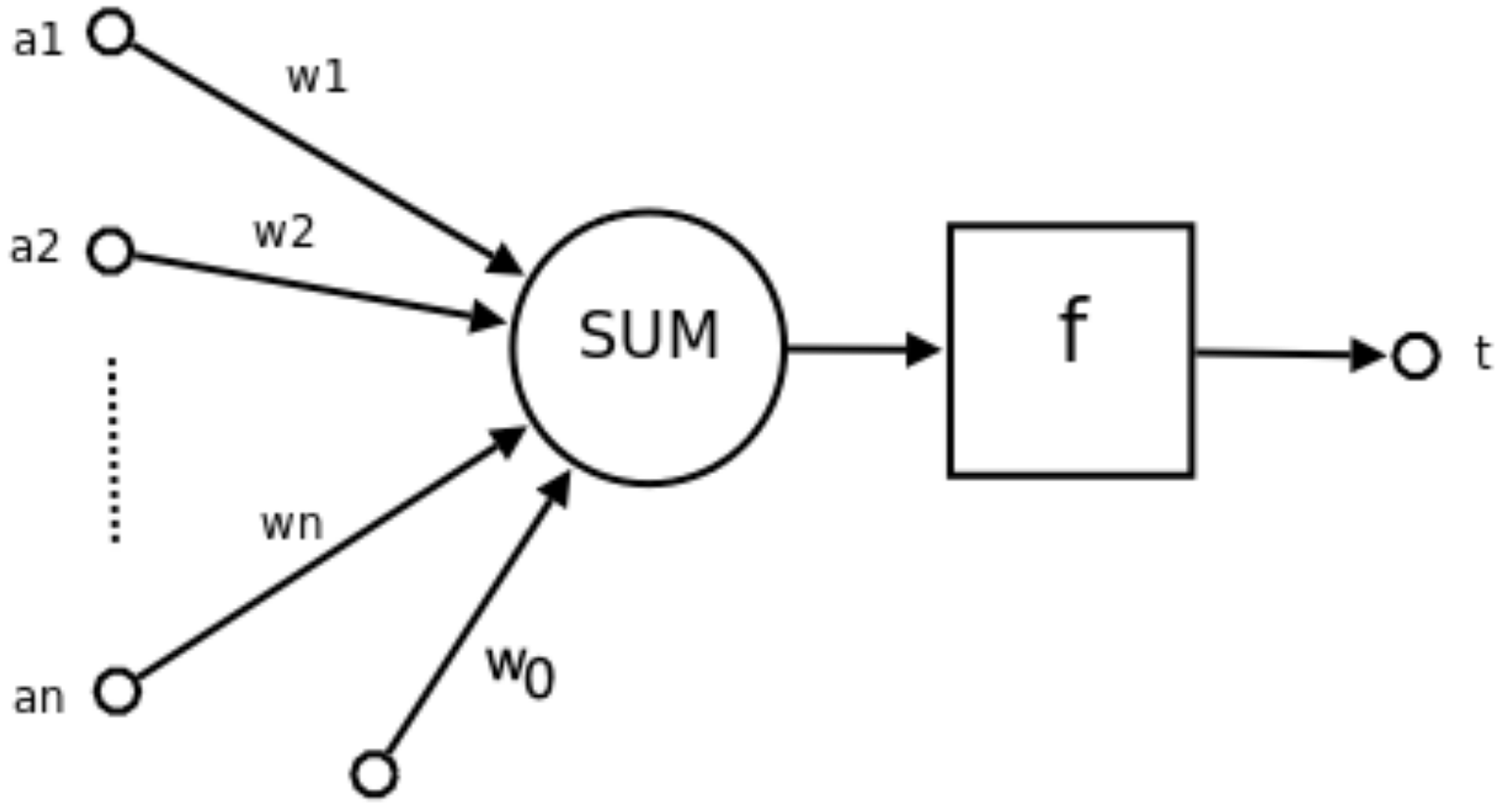
The "bias weight" w_0 corresponds to the threshold when the neuron is triggered.
We have defined a Hypothesis set $\mathcal{H}$ (dummy variable $x_0 \equiv 1$)

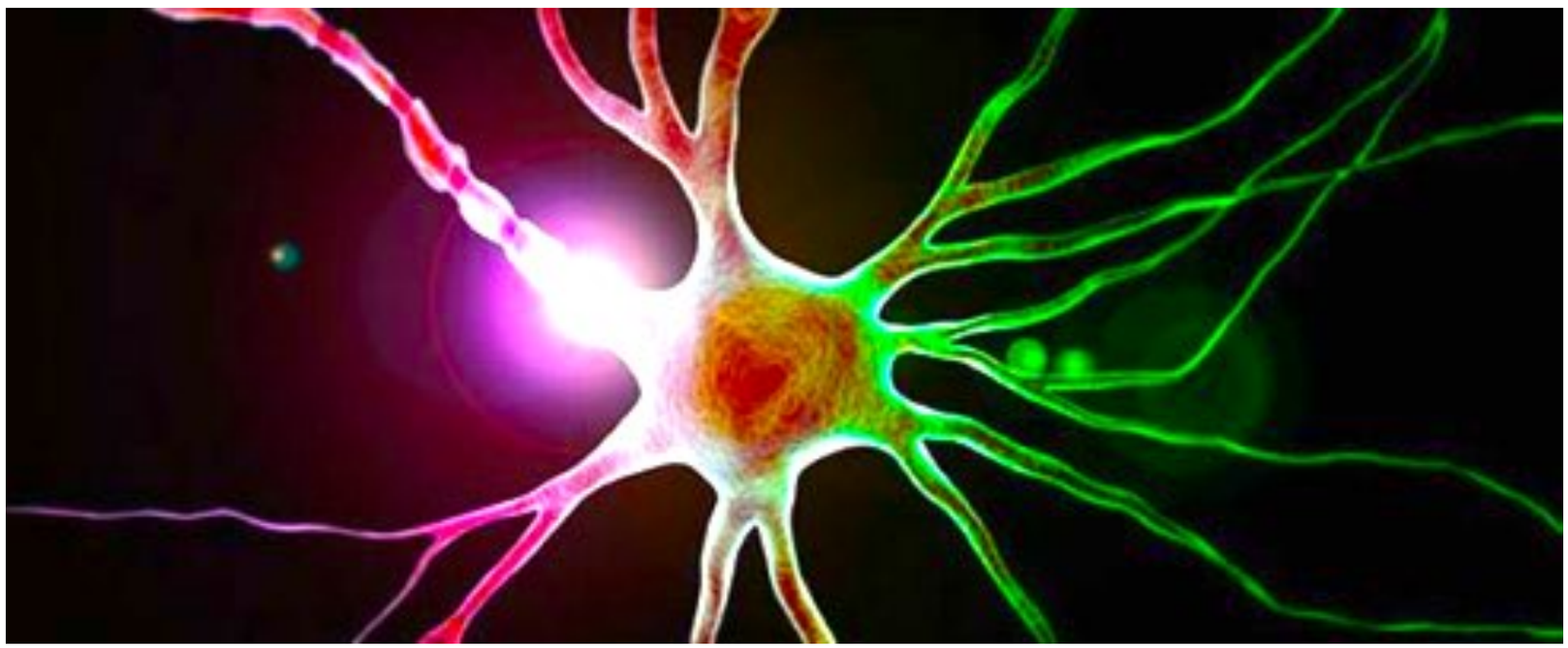
$$\mathcal{H} = \{f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})\}$$

called the perceptron or linear separator

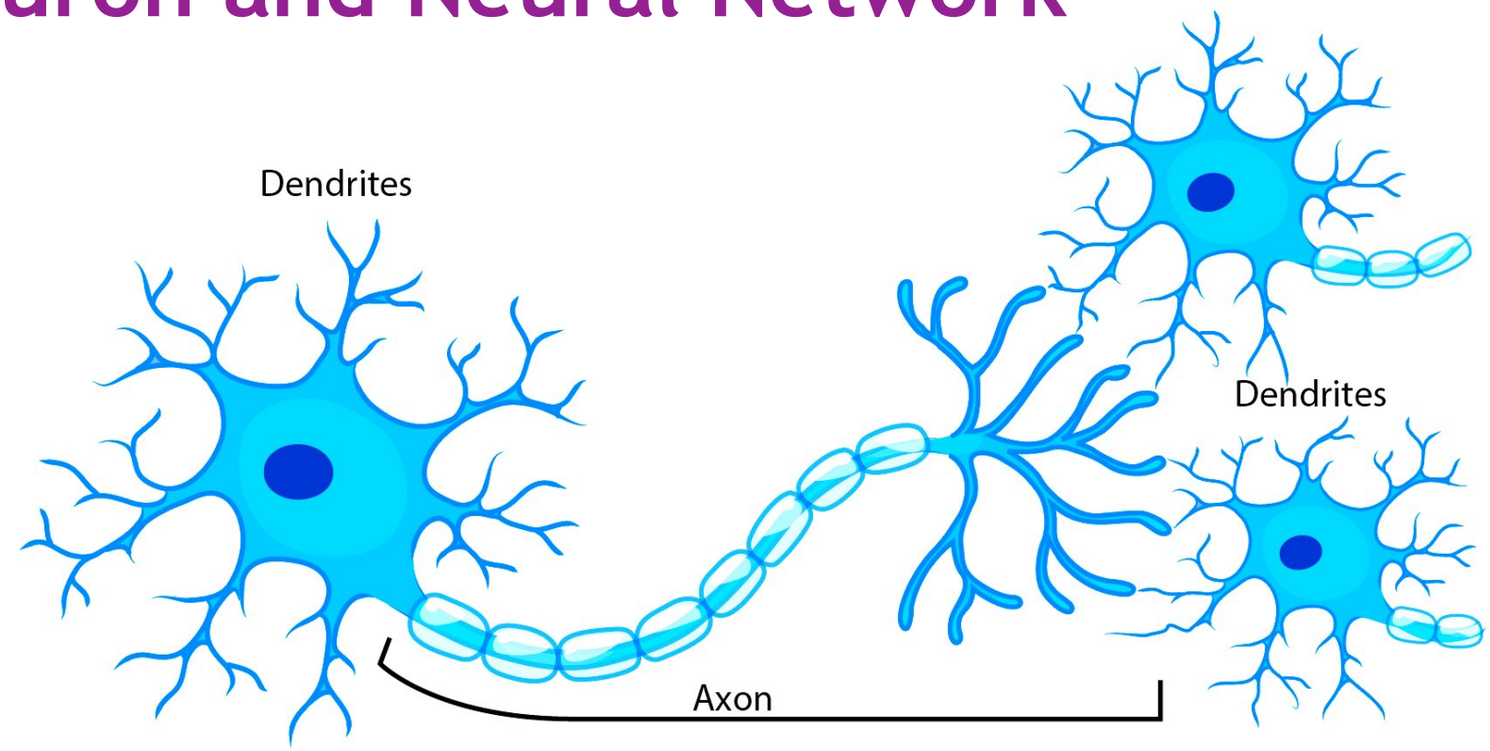
A perceptron fits the data by using a line to separate the $+1$ from -1 data

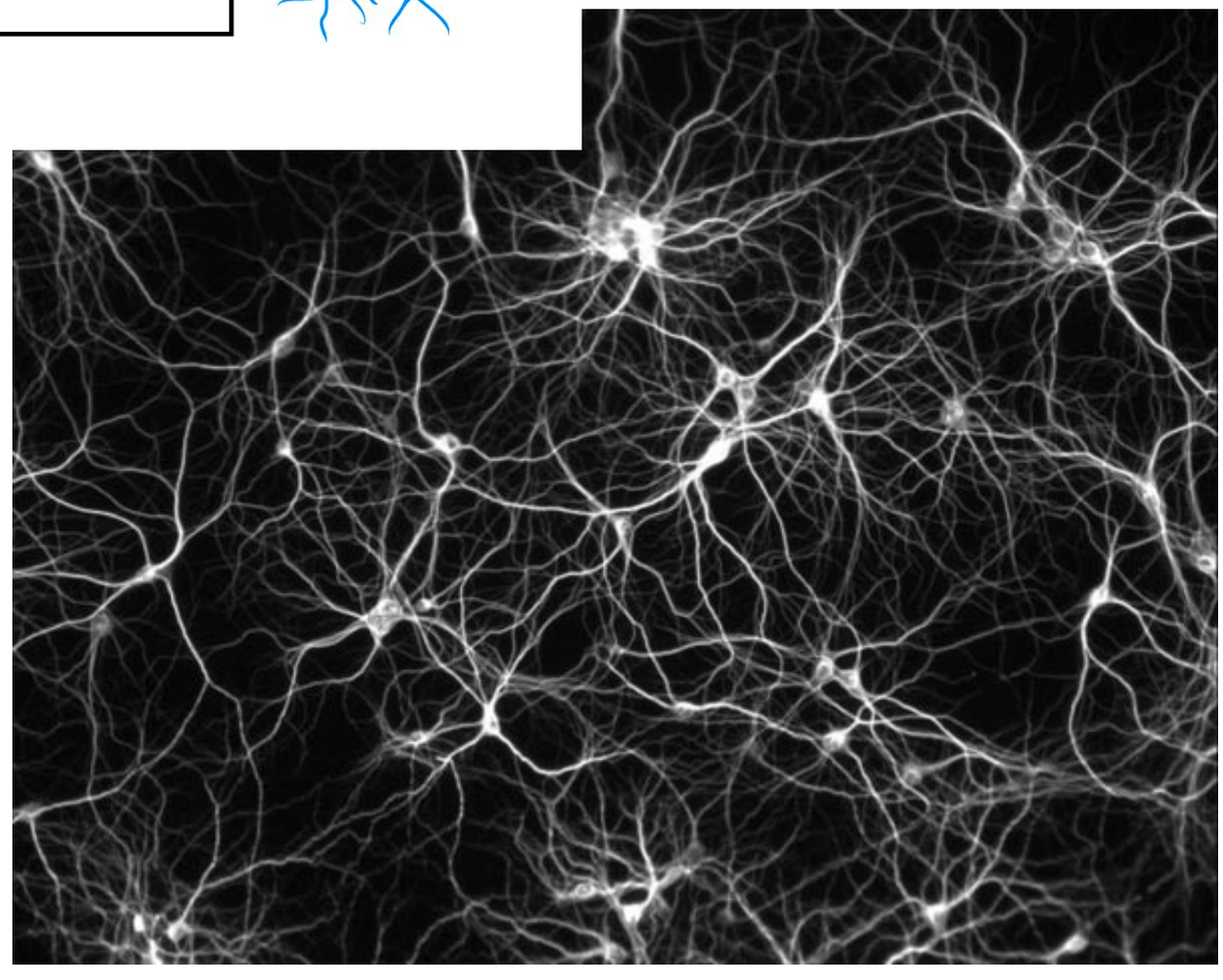
Perceptron and Neuron





Neuron and Neural Network





The perceptron learning algorithm (PLA)

The perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathsf{T}}\mathbf{x})$$

Given the training set:

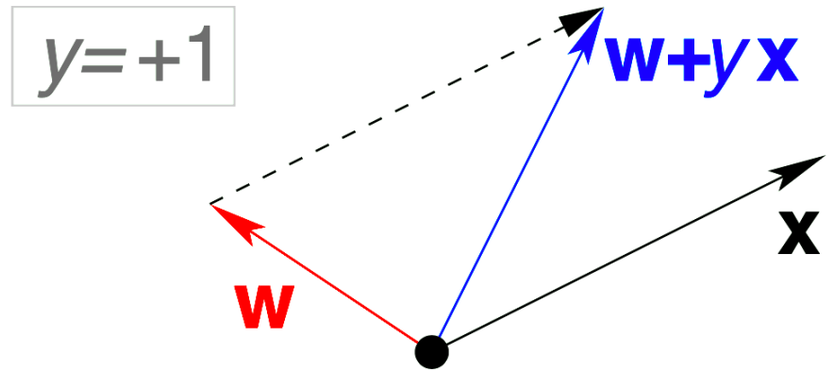
$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

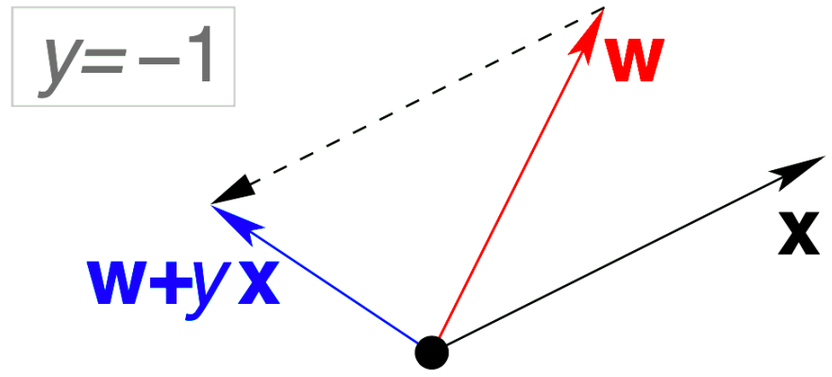
pick a misclassified point:

$$\text{sign}(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n) \neq y_n$$

and update the weight vector:

$$\boxed{\mathbf{w} \leftarrow \mathbf{w} + y_n\mathbf{x}_n}$$





Why adding $\pm\mathbf{x}_i$ to $\mathbf{w}$?

PLA implements our idea: start at some weights and try to improve it

"Incremental learning" on a single example at a time

Does PLA work?

Definition

(Linearly Separable) Suppose we have the training data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ with $y_i = 1$ for $i \in \mathcal{P}$ and y_i for $i \in \mathcal{N}$. Set $\mathcal{D}$ is linear separable if there exists $\mathbf{w}_* \in \mathbb{R}^n$ such that

$$\mathbf{w}_*^T \mathbf{x}_i > 0, \quad i \in \mathcal{P}, \quad \text{and} \quad \mathbf{w}_*^T \mathbf{x}_i < 0, \quad i \in \mathcal{N}.$$

Theorem

Assume $\mathcal{D}$ is linear separable. Then after finite steps, the perceptron algorithm with initial point $\mathbf{w}_0 = 0$ finds a hyperplane that separates the data.

Proof

Since the training data set $\mathcal{D}$ is linear separable, there exists a $\mathbf{w}_*$ such that

$$y_i = \text{sign}(\mathbf{w}_*^T \mathbf{x}_i),$$

which implies that $y_i \mathbf{w}_*^T \mathbf{x}_i > 0, i = 1, ..., m.$ It follows that

$$y_i \mathbf{w}_*^T \mathbf{x}_i \geq \min_i y_i \mathbf{w}_*^T \mathbf{x}_i > 0.$$

Therefore,

$$\begin{aligned}
\mathbf{w}_*^T \mathbf{w}_{(t+1)} &= \mathbf{w}_*^T (\mathbf{w}_{(t)} + y_{i(t)} \mathbf{x}_{i(t)}) \\
&= \mathbf{w}_*^T \mathbf{w}_{(t)} + y_{i(t)} \mathbf{w}_*^T \mathbf{x}_{i(t)} \\
&> \mathbf{w}_*^T \mathbf{w}_{(t)} + \min_i y_i \mathbf{w}_*^T \mathbf{x}_i.
\end{aligned}$$

It follows that at $t = T$

$$\mathbf{w}_*^T \mathbf{w}_{(T)} > \ldots > \mathbf{w}_*^T \mathbf{w}_{(0)} + T \min_i y_i \mathbf{w}_*^T \mathbf{x}_i = T \min_i y_i \mathbf{w}_*^T \mathbf{x}_i.$$

According to the algorithm iteration, we know that $y_{i(t)} \mathbf{w}_{(t)}^T \mathbf{x}_{i(t)} \leq 0.$ It follows that

$$\begin{aligned}
\|\mathbf{w}_{(t+1)}\|_2^2 &= \|\mathbf{w}_{(t)} + y_{i(t)} \mathbf{x}_{i(t)}\|_2^2 \\
&= \|\mathbf{w}_{(t)}\|_2^2 + 2 y_{i(t)} \mathbf{w}_{(t)}^T \mathbf{x}_{i(t)} + \|y_{i(t)} \mathbf{x}_{i(t)}\|_2^2 \\
&\leq \|\mathbf{w}_{(t)}\|_2^2 + \|y_{i(t)} \mathbf{x}_{i(t)}\|_2^2 \\
&\leq \|\mathbf{w}_{(t)}\|_2^2 + \max_i \|\mathbf{x}_i\|_2^2.
\end{aligned}$$

Proof

Therefore, at $t = T$,

$$\begin{aligned}\|\mathbf{w}_{(T)}\|_2^2 &\leq \|\mathbf{w}_{(T-1)}\|_2^2 + \max_i \|\mathbf{x}_i\|_2^2 \\ &\leq \ldots \leq \|\mathbf{w}_{(0)}\|_2^2 + T\max_i \|\mathbf{x}_i\|_2^2 \\ &\leq T\max_i \|\mathbf{x}_i\|_2^2.\end{aligned}$$

It follows that

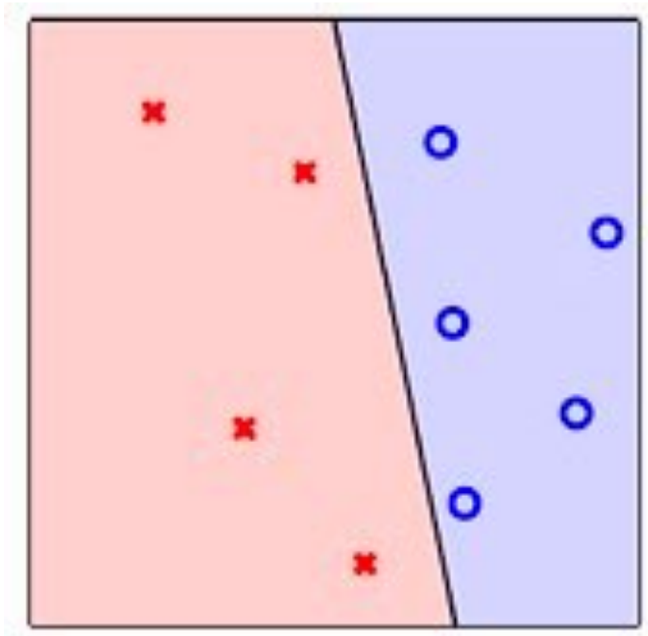
$$\frac{\mathbf{w}_*^T\mathbf{w}_{(T)}}{\|\mathbf{w}_*\|\|\mathbf{w}_{(T)}\|} \geq \frac{T\min_i(y_i\mathbf{w}_*^T\mathbf{x}_i)}{\|\mathbf{w}_*\|\|\mathbf{w}_{(T)}\|} \geq \frac{T\min_i(y_i\mathbf{w}_*^T\mathbf{x}_i)}{\|\mathbf{w}_*\|\sqrt{T}\max_i\|\mathbf{x}_i\|} \geq \sqrt{T}\frac{\min_i(y_i\mathbf{w}_*^T\mathbf{x}_*)}{\|\mathbf{w}_*\|\max_i\|\mathbf{x}_i\|}.$$

The left hand side of this is the cos of the angle between $\mathbf{w}_*$ and $\mathbf{w}_{(T)}$, which is bounded above by 1. Hence we know after at most

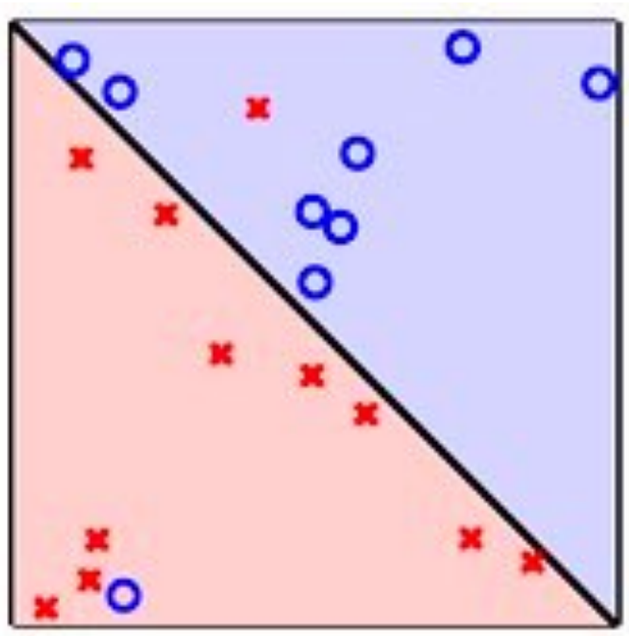
$$T_{\max} = \left\lceil \left(\frac{\|\mathbf{w}_*\|\max_i\|\mathbf{x}_i\|}{\min_i(y_i\mathbf{w}_*^T\mathbf{x}_*)} \right)^2 \right\rceil$$

iterations, the algorithm will terminate.

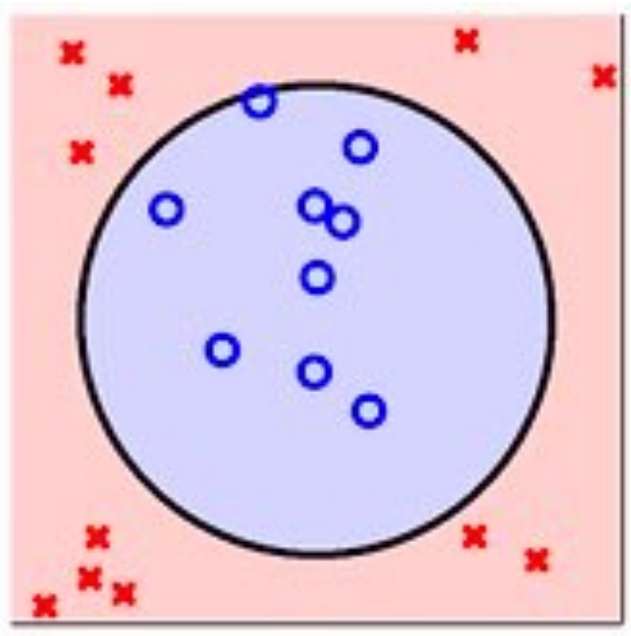
Linear inseparable case



(linear separable)



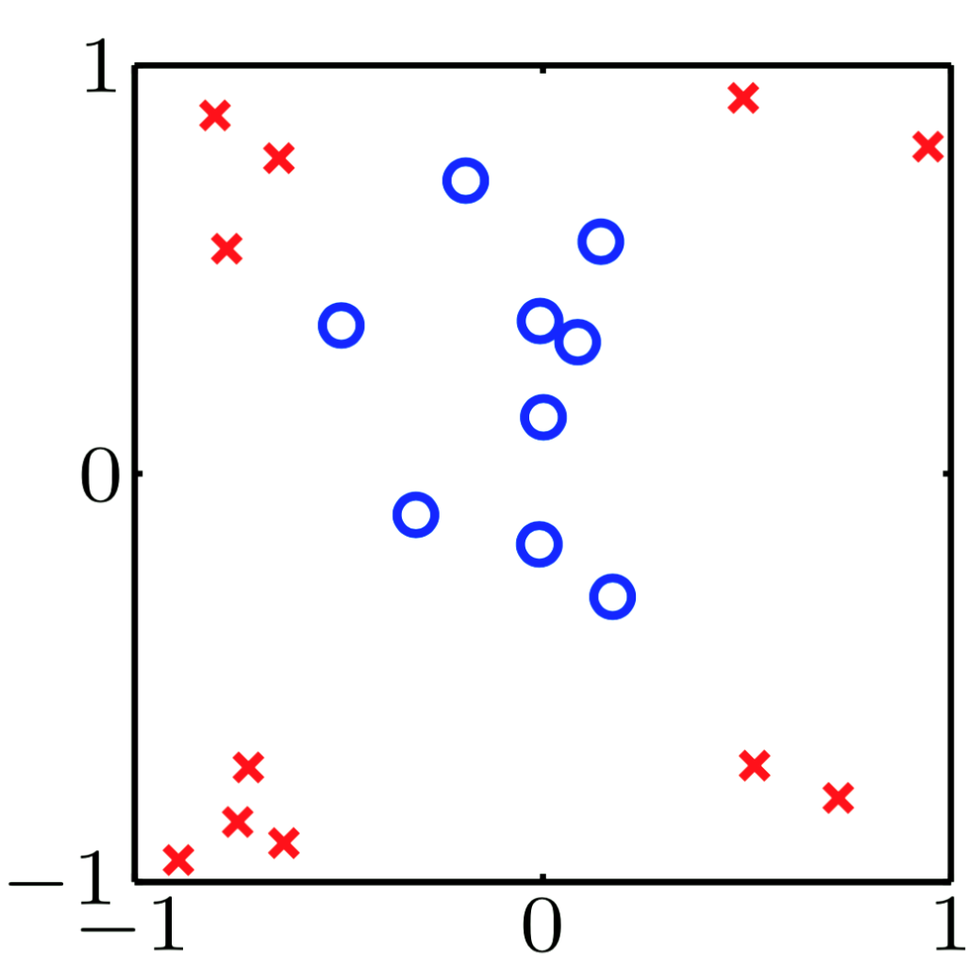
(not linear separable)

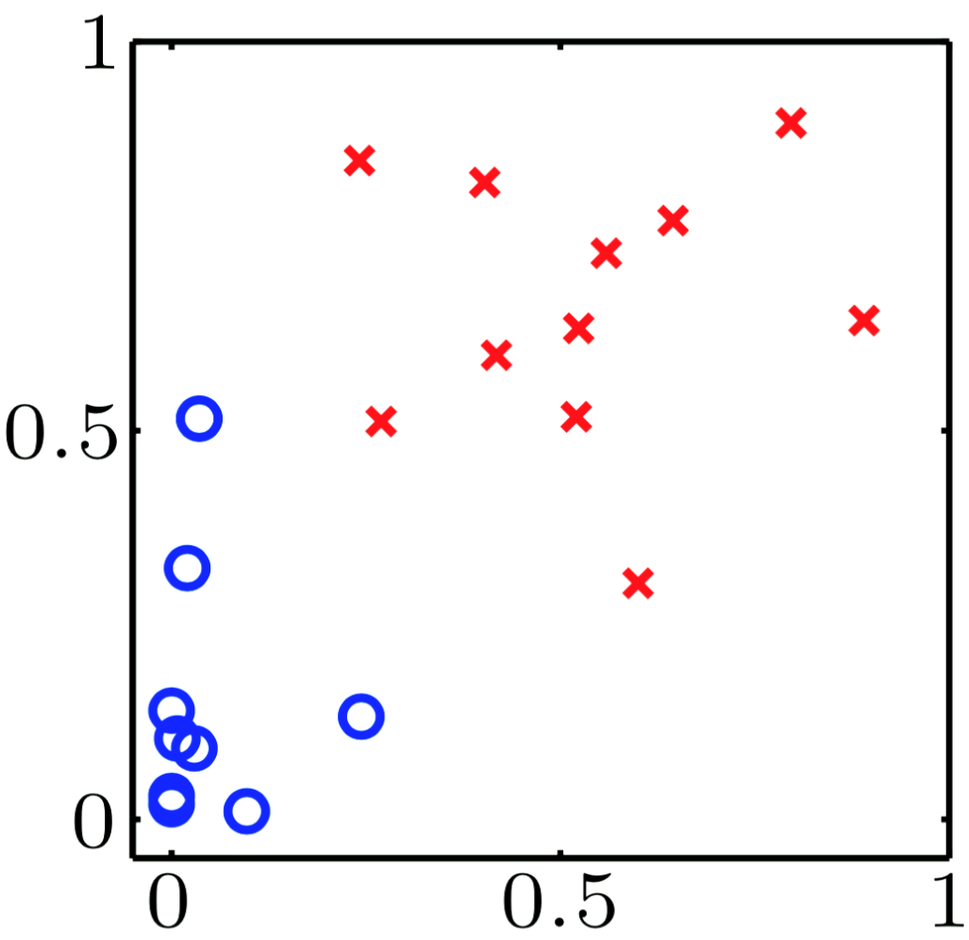


(not linear separable)

Transform the data nonlinearly

$$(x_1, x_2) \xrightarrow{\Phi} (x_1^2, x_2^2)$$





Linear in what?

Linear regression implements

$$\sum_{i=0}^{\mathsf{n}} w_i \; x_i$$

Linear classification implements

$$\text{sign}\left(\sum_{i=0}^{\mathsf{n}} w_i \; x_i\right)$$

Algorithms work because of **linearity in the weights**

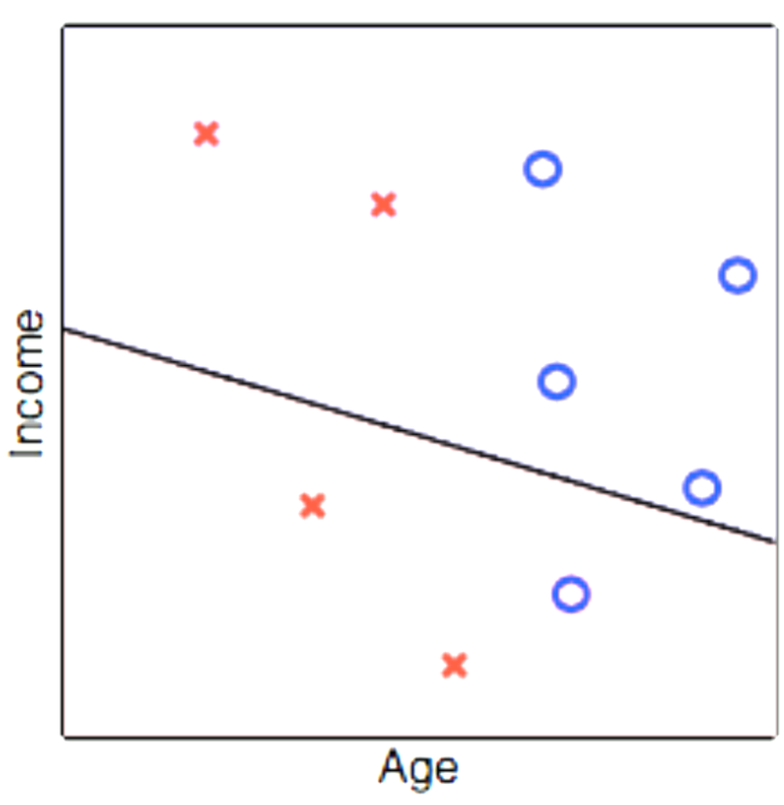
Why do we just use linear regression?

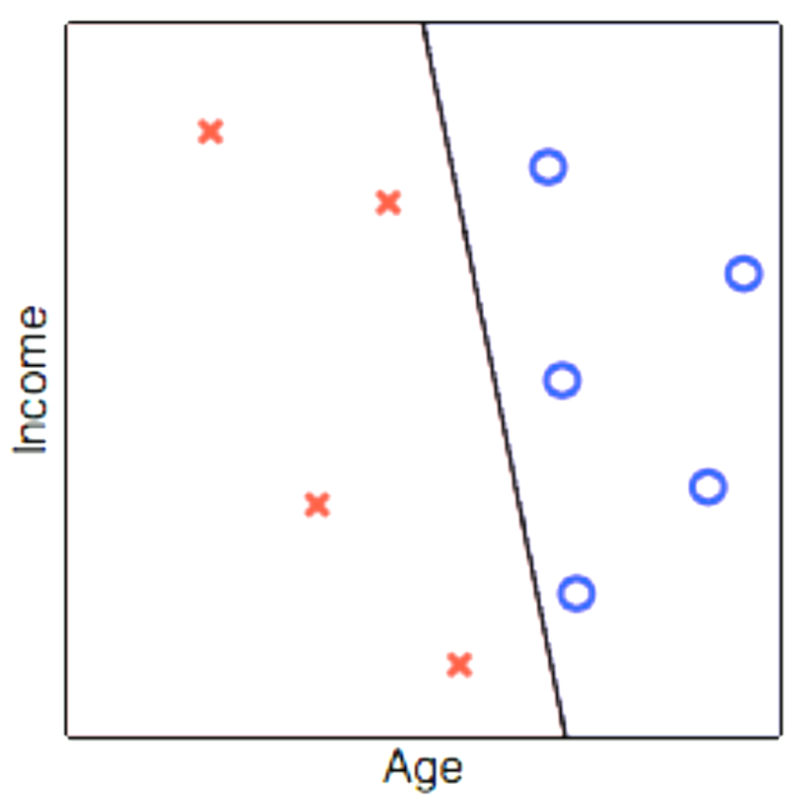
We have defined a Hypothesis set $\mathcal{H}$

$$\mathcal{H} = \{f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})\}$$

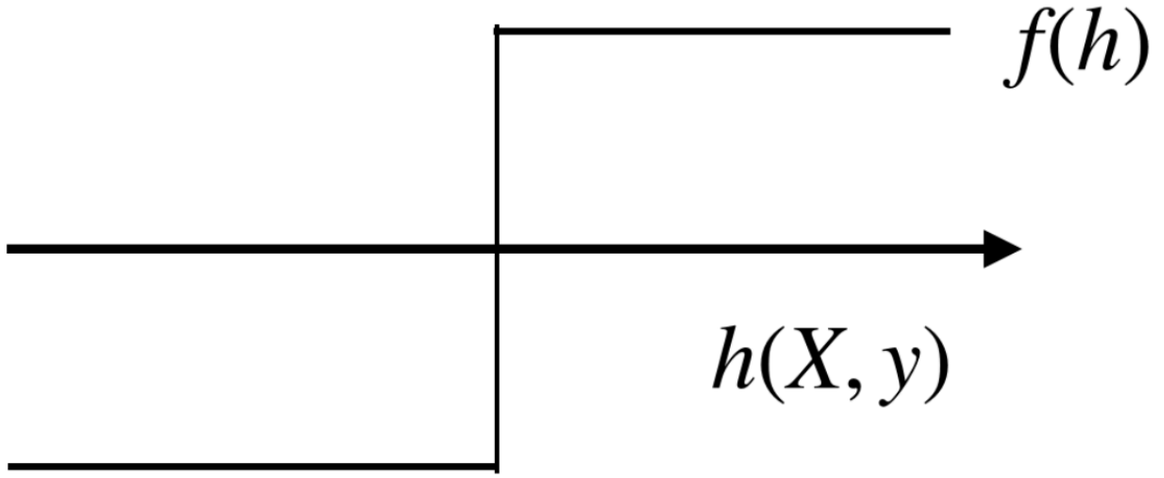
called the perceptron or linear separator

A perceptron fits the data by using a line to separate the $+1$ from -1 data

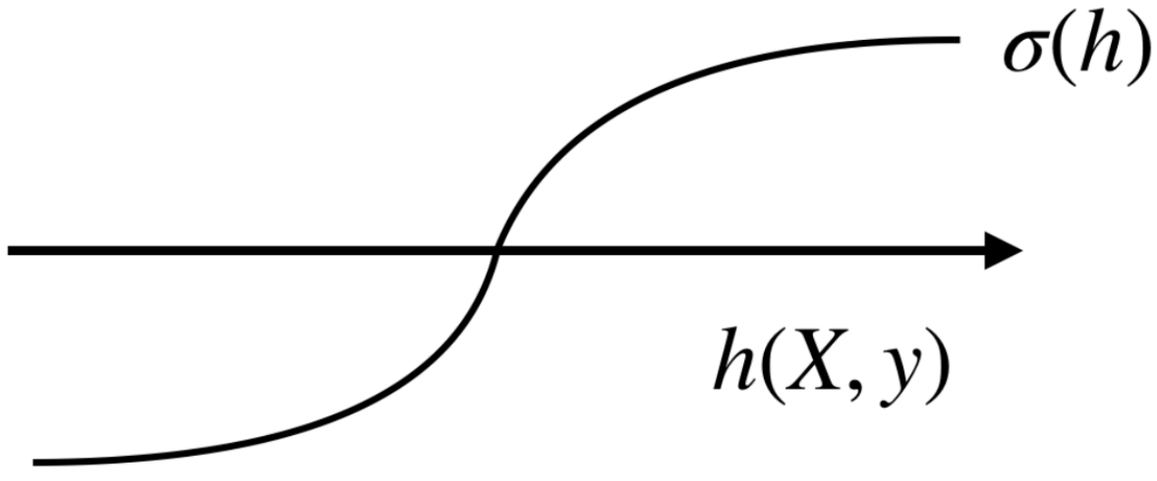




Finding loss functions



- Can we use the threshold function? Why?
- Using the smooth "sigmoid" function, looks like an "S"

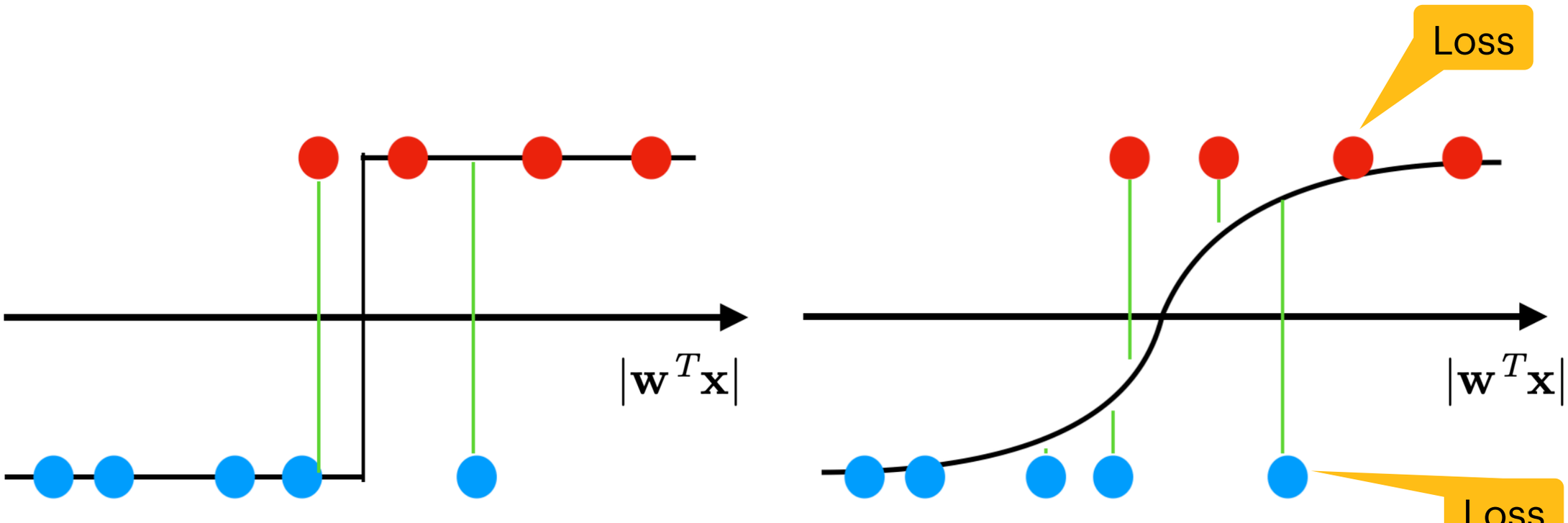


$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

$$\frac{\partial \sigma}{\partial t} = \sigma(t)(1 - \sigma(t))$$

$$1 - \sigma(t) = \sigma(-t)$$

Finding loss functions

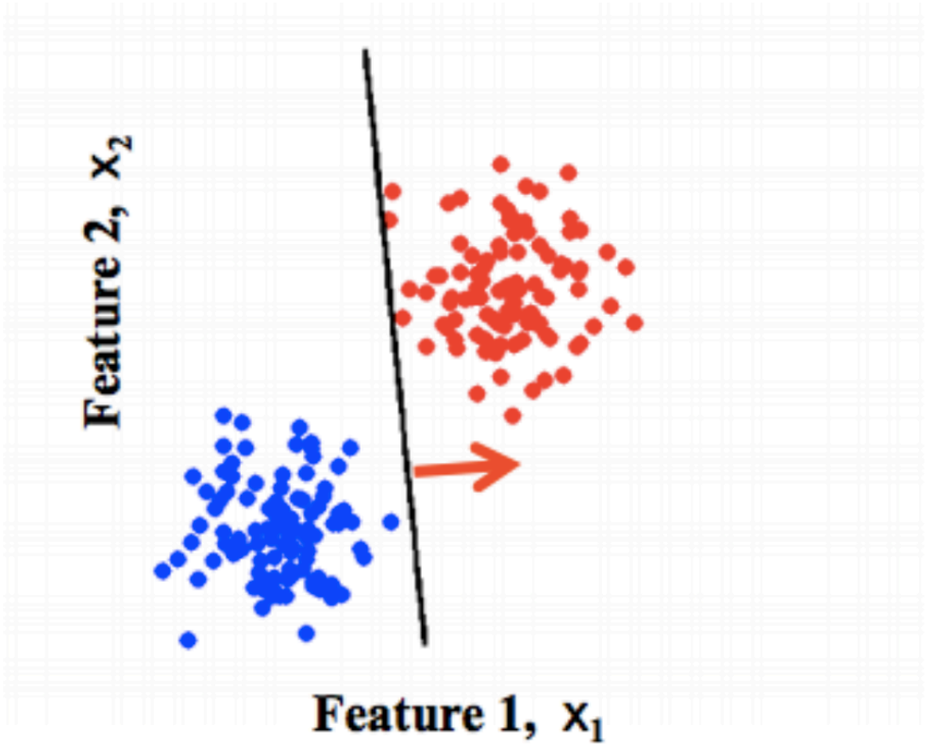


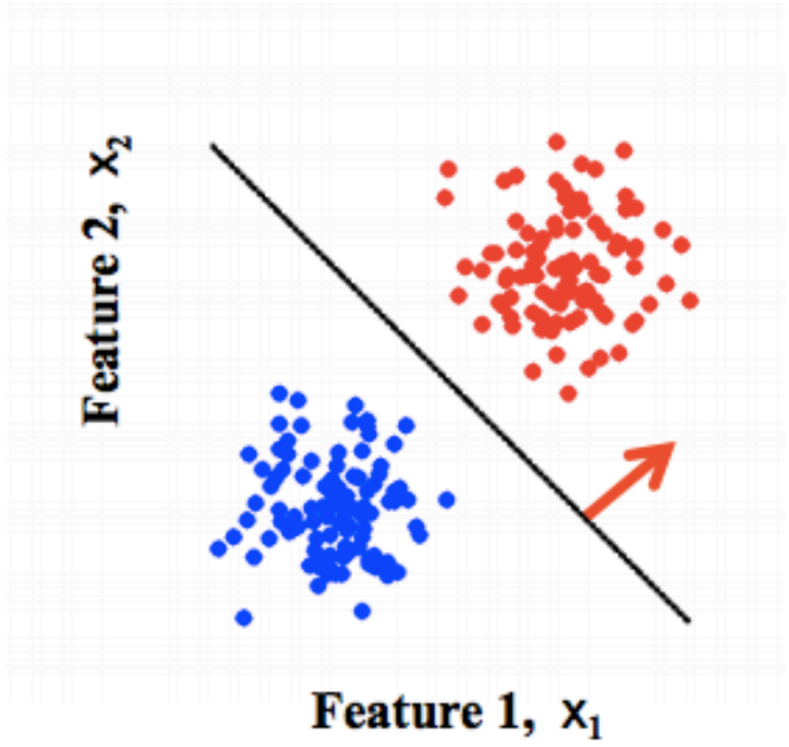
Classification Error = MSE = 2/9 **MSE =** $(0^2 + 1^2 + 0.2^2 + 0.25^2 + 0.5^2 + \dots)/9$

- Far from the decision boundary: $|\mathbf{w}^T\mathbf{x}|$ large, fit curve well, small error
- Nearby the boundary: $|\mathbf{w}^T\mathbf{x}|$ near 1/2, larger error!

Finding loss functions

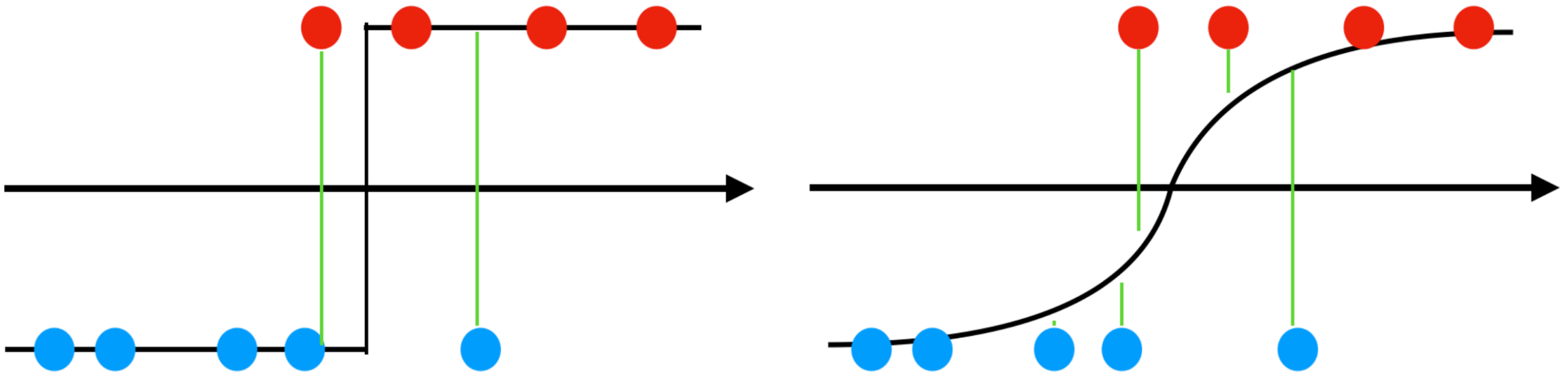
- Which decision boundary is "better"?
  - Both have zero training error
  - But, one of them seems intuitively better...





- Side benefit of "smoothed" error function
  - Encourages data to be far from the decision boundary

Finding loss functions



Classification Error = MSE = 2/9

MSE = $(0^2 + 1^2 + 0.2^2 + 0.25^2 + 0.5^2 + \ldots)/9$

- The MSE of fitting the "S"-shape may not be good. Why?

More complicated error measure

For $y \in \{+1, -1\}$

$$E_{in}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \ln(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}).$$

It looks complicated and ugly ....

But, actually not..., it could be beautiful

- it is based on intuitive probabilistic interpretation of h
- it is very convenient and mathematically friendly ('easy' to minimize)

Verify this error/loss function...

The probabilistic interpretation

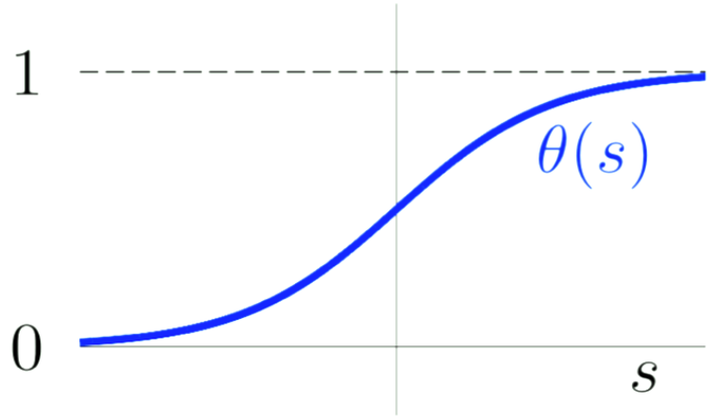
$$p(\boldsymbol{x}) = \mathbb{P}[y = +1 \mid \boldsymbol{x}]$$

$$p(\boldsymbol{x}) = \mathbb{P}[y = +1 \mid \boldsymbol{x}] = \begin{cases} \sigma(\boldsymbol{w}^T\boldsymbol{x}), & y = +1 \\ 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}), & y = -1 \end{cases}$$

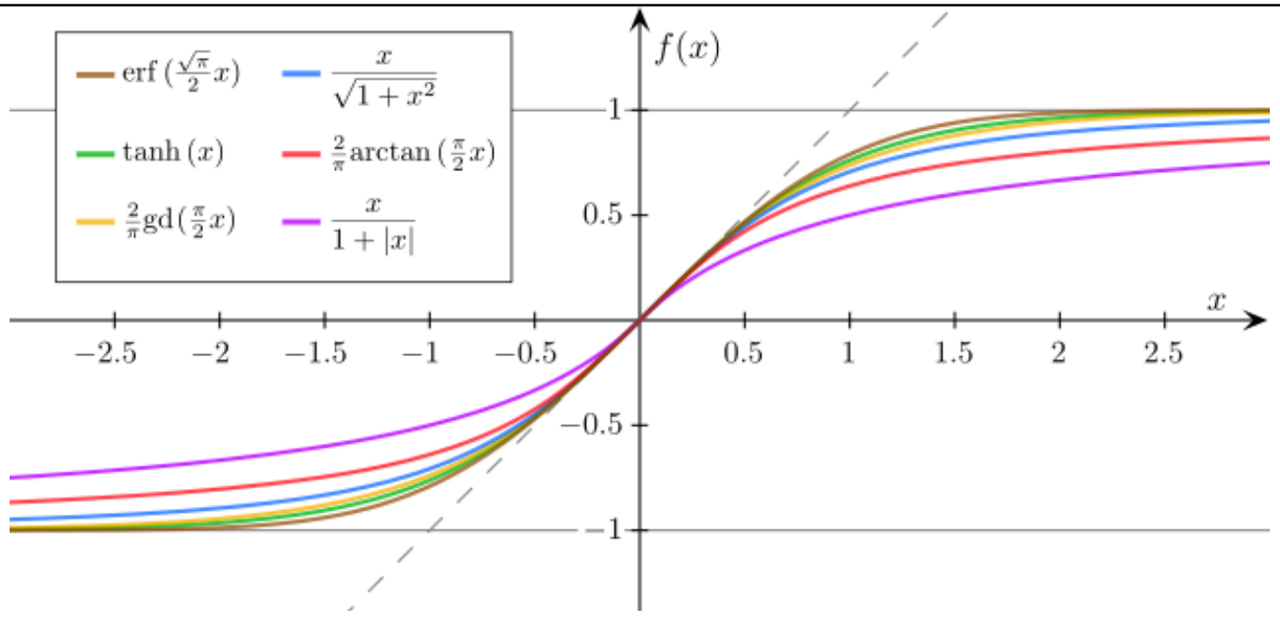
Originally, just use

$$p(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x}) = (\text{sign}(\boldsymbol{w}^T\boldsymbol{x}) + 1)/2$$

The logistic function θ



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



- Generalised logistic function

$$f(x) = (1+e^{-x})^{-\alpha}, \quad \alpha > 0$$

- Smoothstep function

$$f(x) = \begin{cases} \left(\int_0^1 (1-u^2)^N \, du\right)^{-1} \int_0^x (1-u^2)^N \, du, & |x| \le 1 \\ \mathrm{sgn}(x) & |x| \ge 1 \end{cases}$$

$N \ge 1$

- Some algebraic functions, for example

$$f(x) = \frac{x}{\sqrt{1+x^2}}$$

- Logistic function

$$f(x) = \frac{1}{1+e^{-x}}$$

- Hyperbolic tangent (shifted and scaled version of the logistic function, above)

$$f(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- Arctangent function

$$f(x) = \arctan x$$

- Gudermannian function

$$f(x) = \mathrm{gd}(x) = \int_0^x \frac{1}{\cosh t} \, dt = 2\arctan\left(\tanh\left(\frac{x}{2}\right)\right)$$

- Error function

$$f(x) = \mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, dt$$

There are many other "S"-shape options

Error Measure: likelihood

$$P(y \mid \mathbf{x}) = \begin{cases} \sigma(\boldsymbol{w}^T\boldsymbol{x}), & y = +1 \\ 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}), & y = -1 \end{cases}$$

$$P(y \mid \mathbf{x}) = \sigma(y\ \boldsymbol{w}^T\boldsymbol{x})$$

Properties about σ :

$$\sigma(-s) = 1 - \sigma(s), \quad \sigma'(s) = \frac{e^s}{(1+e^s)^2} = \sigma(s)(1-\sigma(s))$$

Likelihood of $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ is

$$\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n) = \prod_{n=1}^{N} \boldsymbol{\sigma}(y_n \mathbf{w}^{\mathsf{T}} \mathbf{x}_n)$$

MLE

Maximize the likelihood, is to minimize:

$$-\frac{1}{N}\ln\left(\prod_{n=1}^{N}\boldsymbol{\sigma}(y_n\ \mathbf{w}^{\mathsf{T}}\ \mathbf{x}_n)\right)$$

$$=\frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{\boldsymbol{\sigma}(y_n\ \mathbf{w}^{\mathsf{T}}\ \mathbf{x}_n)}\right) \qquad \left[\boldsymbol{\sigma}(s)=\frac{1}{1\ +e^{-s}}\right]$$

$$E_{\text{in}}(\mathbf{w})\ =\frac{1}{N}\sum_{n=1}^{N}\underbrace{\ln\left(1+e^{-y_n\mathbf{w}^{\mathsf{T}}\mathbf{x}_n}\right)}_{\mathsf{e}\left(h(\mathbf{x}_n),y_n\right)}$$

"cross-entropy" error

Prediction and Error measures

- What does "$g \approx f$" mean?
- Error Measure: $E(g, f)$
- Almost always point wise definition: $e(h(\boldsymbol{x}), f(\boldsymbol{x}))$

  Examples:

Squared error: $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right) = \left(h(\mathbf{x}) - f(\mathbf{x})\right)^2$

Binary error: $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right) = [\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$

- Overall error $E(h, f)$ = average of pointwise errors $e(h(\boldsymbol{x}), f(\boldsymbol{x}))$
- In-sample error:

$$E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} e(h(\boldsymbol{x}_n), f(\boldsymbol{x}_n))$$

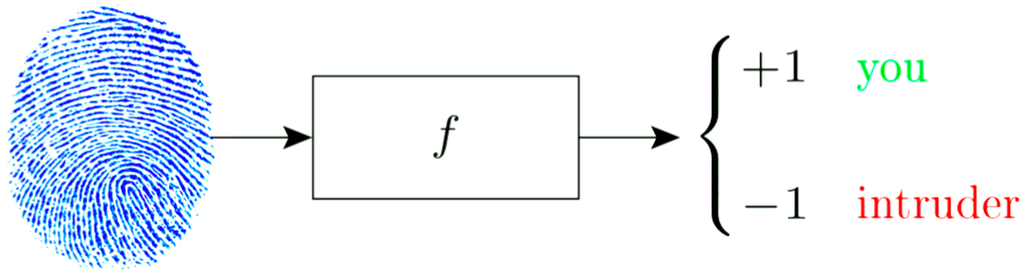
- Out-of-sample error:

$$E_{out}(h) = E_X[\sum_{n=1}^{N} e(h(\boldsymbol{x}_n), f(\boldsymbol{x}_n))]$$

How to choose the error measure

- Fingerprint verification:
- Two types of errors:

  false accept & false reject
- How do we penalize each type?



How do we penalize each type?

		f: $+1$	f: -1
h	$+1$	no error	*false accept*
	-1	*false reject*	no error

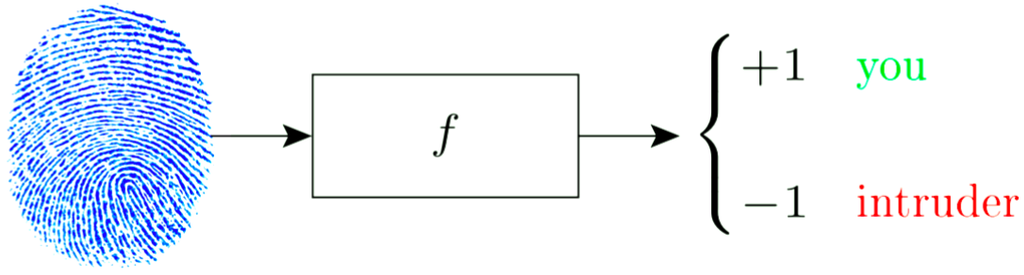
The error measure – for supermarkets

Supermarket verifies fingerprint for discounts

False reject is costly; customer gets annoyed!

False accept is minor; gave away a discount
and intruder left their fingerprint ☺

		f	
		$+1$	-1
h	$+1$	0	1
	-1	10	0



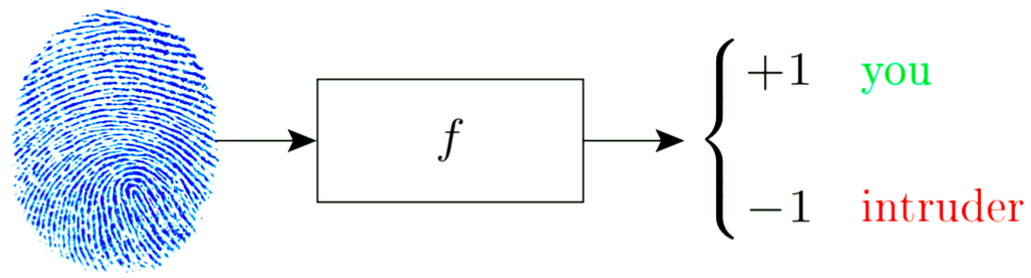
The error measure – for CIA

CIA verifies fingerprint for security

False accept is a disaster!

False reject can be tolerated
Try again; you are an employee ☺

		f	
		$+1$	-1
h	$+1$	0	1000
	-1	1	0



The error should be specified by the user.

Table of error types

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

$$\text{精确率：Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

$$\text{召回率：Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

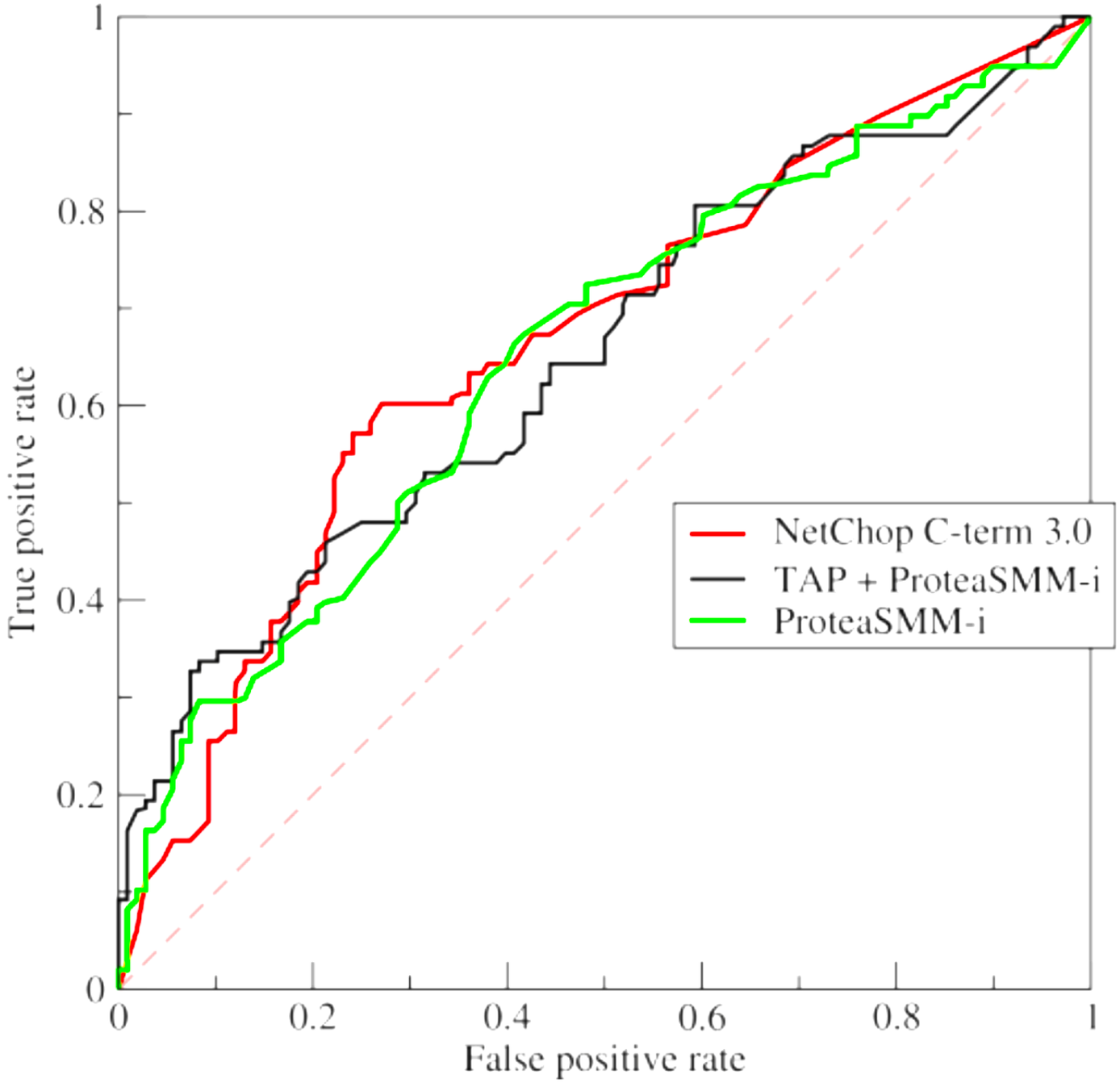
$$\text{准确率：accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Prediction and Test

After obtaining w, what is your prediction function for the future data x?

Remember you only have $\sigma(w^T \cdot)$

What is the threshold? —your choice.



Cases with 0-1 labels

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

$$P(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = 1 \\ 1 - h(\mathbf{x}) & \text{for } y = 0 \end{cases}$$

$$h(\boldsymbol{x}) = \mathbb{P}[y = +1 \mid \boldsymbol{x}] = \begin{cases} \sigma(\boldsymbol{w}^T \boldsymbol{x}), & y = +1 \\ 1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}), & y = 0 \end{cases}$$

$$h(\boldsymbol{x}) = \mathbb{P}[y = +1 \mid \boldsymbol{x}] = \sigma(\boldsymbol{w}^T \boldsymbol{x})^y \Big(1 - \sigma(\boldsymbol{w}^T \boldsymbol{x})\Big)^{1-y}$$

Cases with 0-1 labels

$$E_{in}(\mathbf{w}) = -\sum_{i=1}^{m} \left[y_i \ln \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \ln \left(1 - \sigma(\mathbf{w}^T \mathbf{x}_i)\right) \right]$$

$$\mathbf{g} = \frac{d}{d\mathbf{w}} E_{in}(\mathbf{w}) = \sum_{i=1}^{m} (\sigma_i - y_i)\mathbf{x}_i = \mathbf{X}^T(\boldsymbol{\sigma} - \mathbf{y})$$

$$\mathbf{H} = \frac{d}{d\mathbf{w}} g(\mathbf{w})^T = \sum_{i=1}^{m} (\nabla_{\mathbf{w}} \sigma_i)(\mathbf{x}_i)^T$$

$$= \sum_{i=1}^{m} \sigma_i(1 - \sigma_i)\mathbf{x}_i(\mathbf{x}_i)^T = \mathbf{X}^T\mathbf{D}\mathbf{X}$$

where $\mathbf{D} = \text{diag}(\sigma_i(1 - \sigma_i))$.

Logit model (optional)

$$logit(p(\mathbf{x})) = \log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right)$$

Assume this is a linear model:

$$logit(p(\mathbf{x})) = \mathbf{w}^T\mathbf{x}$$

we have

$$\frac{p(\mathbf{x})}{1-p(\mathbf{x})} = e^{\mathbf{w}^T\mathbf{x}}$$

implying

$$p(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) \quad 1-p(\mathbf{x}) = 1-\sigma(\mathbf{w}^T\mathbf{x})$$

Multiclass cases (softmax regression)

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(M)}, y^{(M)}) \qquad y \in \{1, \ldots, K\}$$

$$\mathbb{P}(y = k | \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}_k{}^T \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^T \mathbf{x})}$$

$$p(y|\mathbf{x}) = \begin{bmatrix} P(y=1|\mathbf{x}) \\ P(y=2|\mathbf{x}) \\ \vdots \\ P(y=K|\mathbf{x}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\mathbf{w}_i^T \mathbf{x})} \begin{bmatrix} \exp(\mathbf{w}_1^T \mathbf{x}) \\ \exp(\mathbf{w}_2^T \mathbf{x}) \\ \vdots \\ \exp(\mathbf{w}_K^T \mathbf{x}) \end{bmatrix}$$

Multiclass cases (softmax regression)

$$p(y|\mathbf{x}) = \begin{bmatrix} P(y=1|\mathbf{x}) \\ P(y=2|\mathbf{x}) \\ \vdots \\ P(y=K|\mathbf{x}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\mathbf{w}_i^T \mathbf{x})} \begin{bmatrix} \exp(\mathbf{w}_1^T \mathbf{x}) \\ \exp(\mathbf{w}_2^T \mathbf{x}) \\ \vdots \\ \exp(\mathbf{w}_K^T \mathbf{x}) \end{bmatrix}$$

$$T_{i,k} = \begin{cases} 1 & \text{if } y^{(i)} = k \\ 0 & \text{otherwise.} \end{cases}$$

$$\boldsymbol{J(W)} = -\left[\sum_{i=1}^{M} \sum_{k=1}^{K} T_{i,k} \log \frac{\exp(\mathbf{w}_k^T \mathbf{x}^{(i)})}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^T \mathbf{x}^{(i)})} \right]$$

Multiclass cases (softmax regression)

$$p(y|\mathbf{x}) = \begin{bmatrix} P(y=1|\mathbf{x}) \\ P(y=2|\mathbf{x}) \\ \vdots \\ P(y=K|\mathbf{x}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\mathbf{w}_i^T \mathbf{x})} \begin{bmatrix} \exp(\mathbf{w}_1^T \mathbf{x}) \\ \exp(\mathbf{w}_2^T \mathbf{x}) \\ \vdots \\ \exp(\mathbf{w}_K^T \mathbf{x}) \end{bmatrix}$$

$$\mathbb{P}(y=k|\mathbf{x},\mathbf{w}) = \frac{\exp({\mathbf{w}_k}^T \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^T \mathbf{x})}$$

$$(\mathbf{w}_1 - \overline{\mathbf{w}}, \ldots, \mathbf{w}_K - \overline{\mathbf{w}})$$

$$J(\boldsymbol{W}) = -\left[\sum_{i=1}^{M} T_{i,K} \log \frac{1}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^T \mathbf{x}^{(i)})} + \sum_{i=1}^{M} \sum_{k=1}^{K-1} T_{i,k} \log \frac{\exp(\mathbf{w}_k^T \mathbf{x}^{(i)})}{\sum_{k=1}^{K} \exp(\mathbf{w}_k^T \mathbf{x}^{(i)})}\right]$$

Advantages of Logistic Regression

Algorithms for solving logistic algorithms