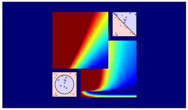
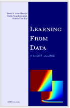
Basics of Machine Learning

from NTU-Coursera Free Mandarin-based Online Course
“Machine Learning Foundations” (機器學習基石) &

My Amazon Best-Seller Book “Learning from Data” (全華代理)





Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science & Information Engineering

National Taiwan University (國立台灣大學資訊工程系)

Roadmap

- What is Machine Learning
- Perceptron Learning Algorithm
- Types of Learning
- Possibility of Learning
- Linear Regression
- Logistic Regression
- Nonlinear Transform
- Overfitting
- Principles of Learning

What is Machine Learning

From Learning to Machine Learning

learning: acquiring skill
with experience accumulated from observations

observations → learning → skill

machine learning: acquiring skill
with experience accumulated/**computed** from data

data → ML → skill

What is skill?

A More Concrete Definition

skill
⇔ improve some performance measure (e.g. prediction accuracy)

machine learning: improving some performance measure
with experience **computed** from data

data → ML → improved performance measure

An Application in Computational Finance

stock data → ML → more investment gain

Why use machine learning?

Yet Another Application: Tree Recognition

- ‘define’ trees and hand-program: **difficult**
- learn from data (observations) and recognize: a **3-year-old can do so**
- ‘ML-based tree recognition system’ can be **easier to build** than hand-programmed system

ML: an **alternative route** to build complicated systems

The Machine Learning Route

ML: an **alternative route** to build complicated systems

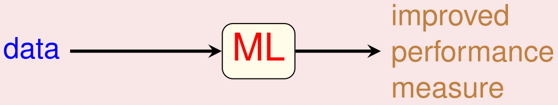
Some Use Scenarios

- when human cannot program the system manually
  —navigating on Mars
- when human cannot 'define the solution' easily
  —speech/visual recognition
- when needing rapid decisions that humans cannot do
  —high-frequency trading
- when needing to be user-oriented in a massive scale
  —consumer-targeted marketing

Give a **computer** a fish, you feed it for a day;
teach it how to fish, you feed it for a lifetime. **:-)**

Key Essence of Machine Learning

machine learning: improving some performance measure
with experience **computed** from data



1. exists some 'underlying pattern' to be learned
—so 'performance measure' can be improved
2. but no programmable (easy) definition
—so 'ML' is needed
3. somehow there is data about the pattern
—so ML has some 'inputs' to learn from

key essence: help decide whether to use ML

Entertainment: Recommender System (1/2)

data → ML → skill

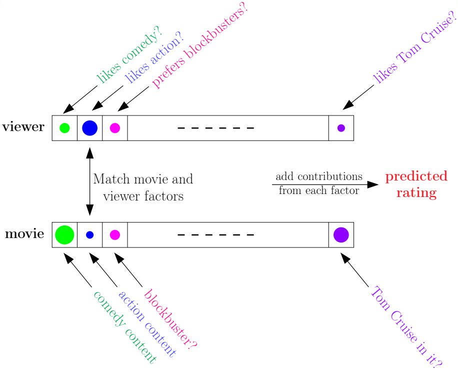
- data: how many users have rated some movies
- skill: predict how a user would rate an unrated movie

A Hot Problem

- competition held by Netflix in 2006
  - 100,480,507 ratings that 480,189 users gave to 17,770 movies
  - 10% improvement = **1 million dollar prize**
- similar competition (movies → songs) held by Yahoo! in KDDCup 2011
  - 252,800,275 ratings that 1,000,990 users gave to 624,961 songs

How can machines **learn our preferences**?

Entertainment: Recommender System (2/2)



A Possible ML Solution

- pattern:
  rating ← viewer/movie factors
- learning:
  known rating
  → learned factors
  → unknown rating prediction

key part of the **world-champion** (again!) system from National Taiwan Univ. in KDDCup 2011

Components of Learning: Metaphor Using Credit Approval

Applicant Information

age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000

unknown pattern to be learned:
'approve credit card good for bank?'

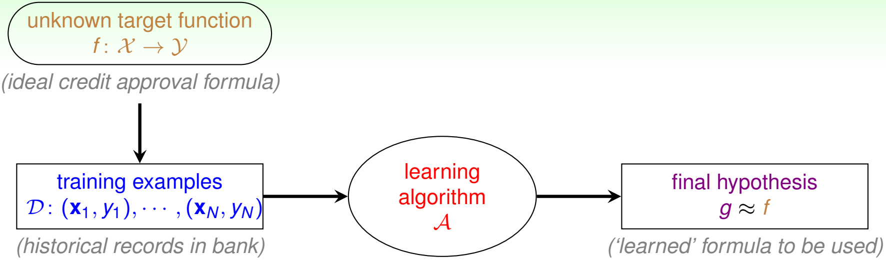
Formalize the Learning Problem

Basic Notations

- input: $\mathbf{x} \in \mathcal{X}$ (customer application)
- output: $y \in \mathcal{Y}$ (good/bad after approving credit card)
- unknown pattern to be learned $\Leftrightarrow$ target function: $f\colon \mathcal{X} \to \mathcal{Y}$ (ideal credit approval formula)
- data $\Leftrightarrow$ training examples: $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$ (historical records in bank)
- hypothesis $\Leftrightarrow$ skill with hopefully good performance: $g\colon \mathcal{X} \to \mathcal{Y}$ ('learned' formula to be used)

$\{(\mathbf{x}_n, y_n)\}$ from f $\longrightarrow$ ML $\longrightarrow$ g

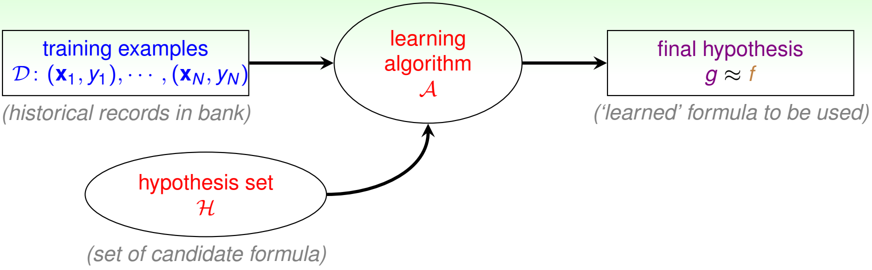
Learning Flow for Credit Approval



- target f **unknown**
  (i.e. no programmable definition)
- hypothesis g hopefully $\approx f$
  but possibly **different** from f
  (perfection 'impossible' when f unknown)

What does g look like?

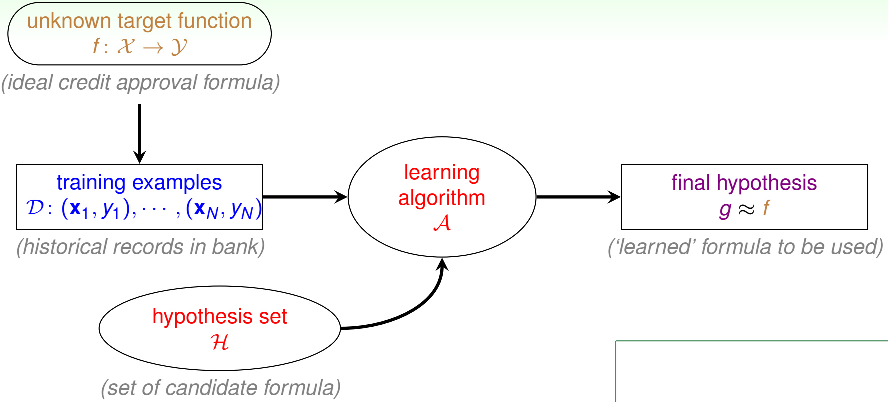
The Learning Model



- assume $g \in \mathcal{H} = \{h_k\}$, i.e. approving if
  - h_1: annual salary > NTD 800,000
  - h_2: debt > NTD 100,000 (really?)
  - h_3: year in job ≤ 2 (really?)
- hypothesis set $\mathcal{H}$:
  - can contain **good or bad hypotheses**
  - up to $\mathcal{A}$ to pick the 'best' one as g

learning model = $\mathcal{A}$ and $\mathcal{H}$

Practical Definition of Machine Learning



machine learning:
use data to compute hypothesis g
that approximates target f

Machine Learning and Data Mining

Machine Learning	Data Mining
use data to compute hypothesis g that approximates target f	use **(huge)** data to **find property** that is interesting

- if 'interesting property' **same as** 'hypothesis that approximate target'
  —**ML = DM** (usually what KDDCup does)
- if 'interesting property' **related to** 'hypothesis that approximate target'
  —**DM can help ML, and vice versa** (often, but not always)
- traditional DM also focuses on **efficient computation in large database**

difficult to distinguish ML and DM in reality

Machine Learning and Artificial Intelligence

Machine Learning

use data to compute hypothesis g that approximates target f

Artificial Intelligence

compute **something that shows intelligent behavior**

- $g \approx f$ is something that shows intelligent behavior —**ML can realize AI**, among other routes
- e.g. chess playing
  - traditional AI: game tree
  - ML for AI: ‘learning from board data’

ML is one possible route to realize AI

Machine Learning and Statistics

Machine Learning

use data to compute hypothesis g that approximates target f

Statistics

use data to **make inference about an unknown process**

- g is an inference outcome; f is something unknown —statistics **can be used to achieve ML**
- traditional statistics also focus on **provable results with math assumptions**, and care less about computation

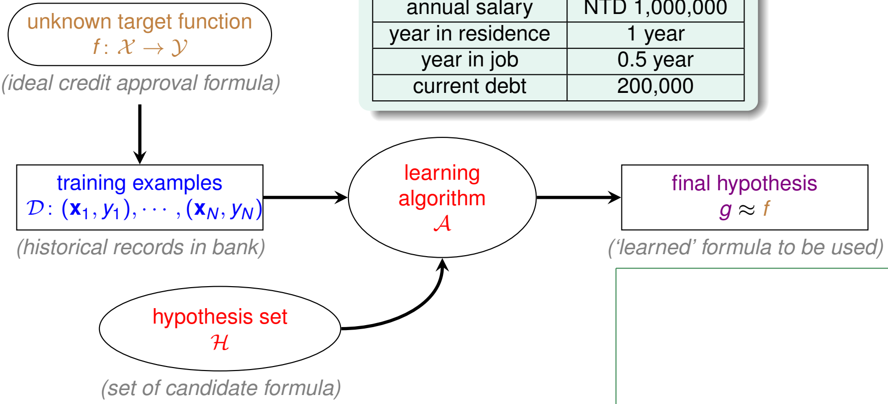
statistics: many useful tools for ML

Perceptron Learning Algorithm

Credit Approval Problem Revisited

Applicant Information

age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000



what hypothesis set can we use?

A Simple Hypothesis Set: the 'Perceptron'

age	23 years
annual salary	NTD 1,000,000
year in job	0.5 year
current debt	200,000

- For $\mathbf{x} = (x_1, x_2, \cdots, x_d)$ '**features of customer**', compute a weighted 'score' and

$$\text{approve credit if} \quad \sum_{i=1}^{d} w_i x_i > \text{threshold}$$

$$\text{deny credit if} \quad \sum_{i=1}^{d} w_i x_i < \text{threshold}$$

- $\mathcal{Y}$: $\{+1(\textbf{good}), -1(\textbf{bad})\}$, 0 ignored—linear formula $h \in \mathcal{H}$ are

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right)$$

called 'perceptron' hypothesis historically

Vector Form of Perceptron Hypothesis

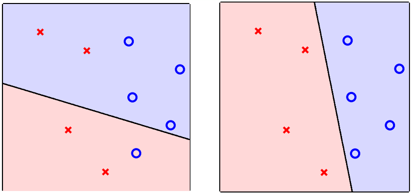
$$
\begin{aligned}
h(\mathbf{x}) &= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right) \\
&= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + \underbrace{(-\text{threshold})}_{w_0} \cdot \underbrace{(+1)}_{x_0}\right) \\
&= \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right) \\
&= \text{sign}\left(\mathbf{w}^T \mathbf{x}\right)
\end{aligned}
$$

- each 'tall' **w** represents a hypothesis *h* & is multiplied with 'tall' **x** —**will use tall versions to simplify notation**

what do perceptrons *h* 'look like'?

Perceptrons in $\mathbb{R}^2$

$$h(\mathbf{x}) = \text{sign}\left(w_0 + w_1 x_1 + w_2 x_2\right)$$



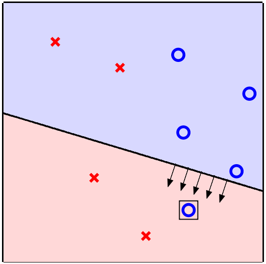
- customer features $\mathbf{x}$: points on the plane (or points in $\mathbb{R}^d$)
- labels y: $\circ$ (+1), $\times$ (-1)
- hypothesis h: **lines** (or hyperplanes in $\mathbb{R}^d$)
  —positive on one side of a line, negative on the other side
- different line classifies customers differently

perceptrons $\Leftrightarrow$ **linear (binary) classifiers**

Select g from $\mathcal{H}$

$\mathcal{H}$ = all possible perceptrons, $g = ?$

- want: $g \approx f$ (hard when f unknown)
- almost necessary: $g \approx f$ on $\mathcal{D}$, ideally $g(\mathbf{x}_n) = f(\mathbf{x}_n) = y_n$
- difficult: $\mathcal{H}$ is of **infinite** size
- idea: start from some g_0, and 'correct' its mistakes on $\mathcal{D}$



will represent g_0 by its weight vector $\mathbf{w}_0$

Perceptron Learning Algorithm

start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

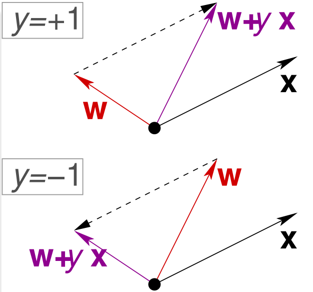
$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

. . . until no more mistakes

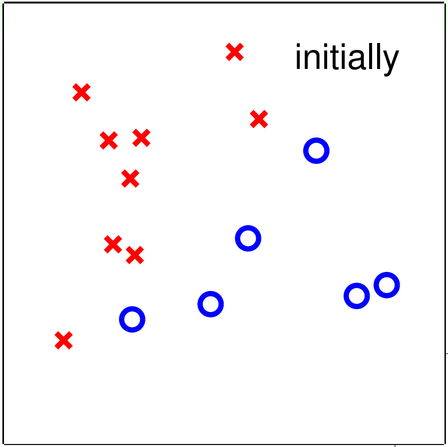
return last $\mathbf{w}$ (called $\mathbf{w}_{\text{PLA}}$) as g



That's it!

—*A fault confessed is half redressed.* :-)

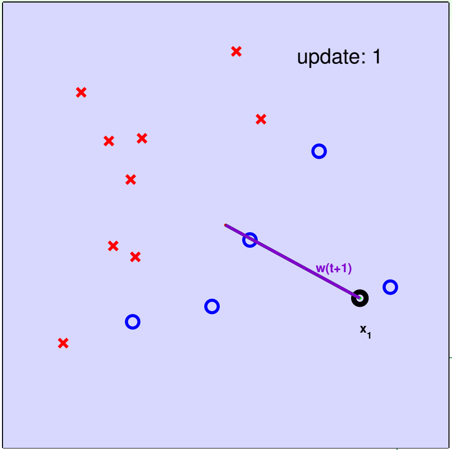
Seeing is Believing



worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

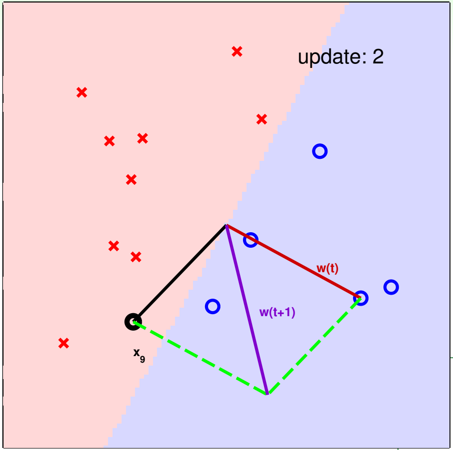
Seeing is Believing



worked like a charm with $<$ 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

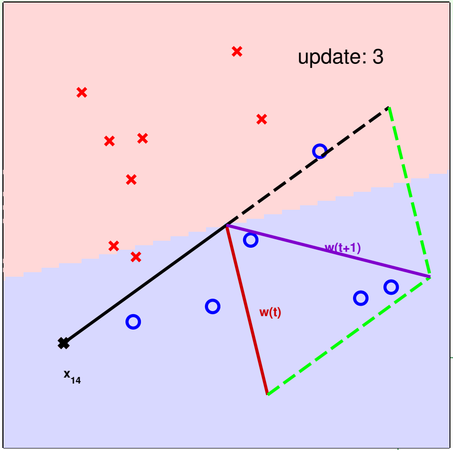
Seeing is Believing



worked like a charm with $<$ 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

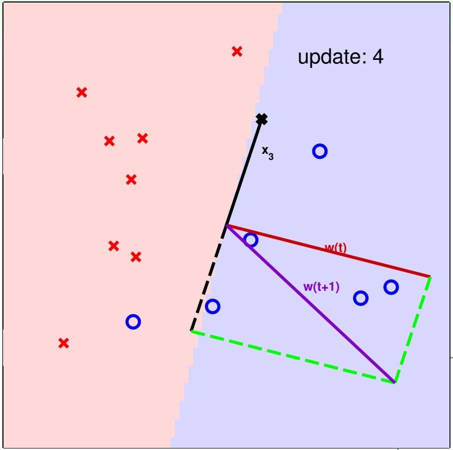
Seeing is Believing



worked like a charm with $<$ 20 lines!!

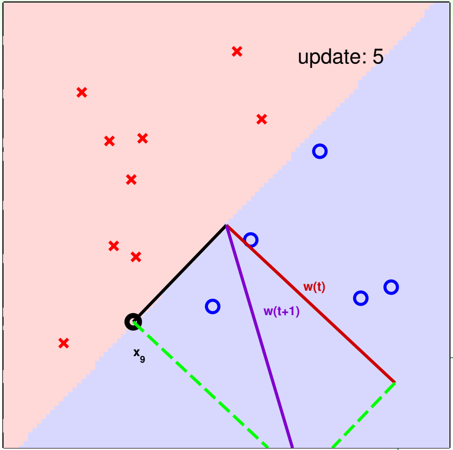
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



worked like a charm with < 20 **lines!!**
(note: made $x_i \gg x_0 = 1$ for visual purpose)

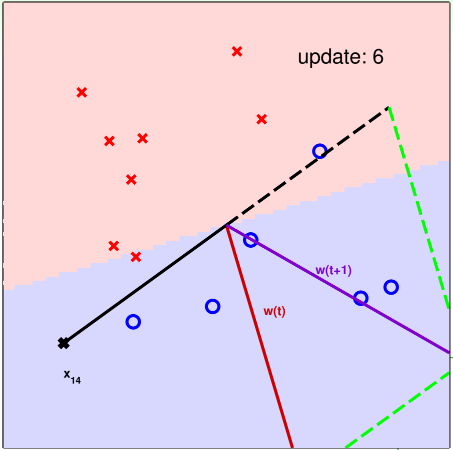
Seeing is Believing



worked like a charm with $<$ 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

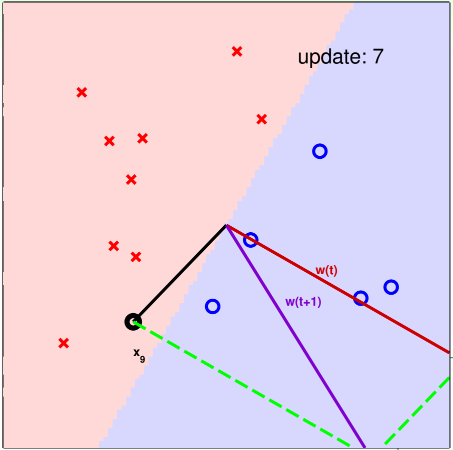
Seeing is Believing



worked like a charm with $<$ 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

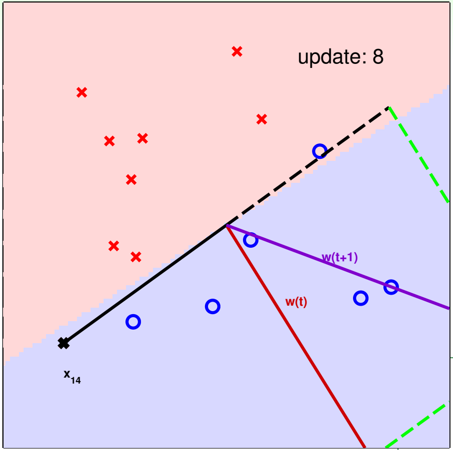
Seeing is Believing



worked like a charm with $<$ 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

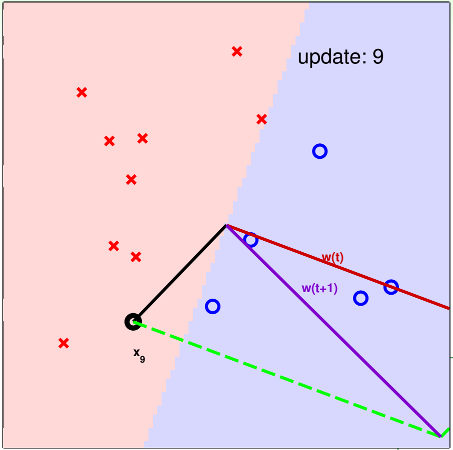
Seeing is Believing



worked like a charm with < 20 lines!!

(note: made $x_i \gg x_0 = 1$ for visual purpose)

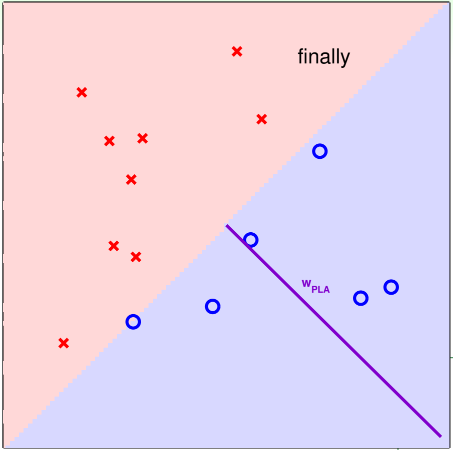
Seeing is Believing



worked like a charm with < 20 **lines!!**

(note: made $x_i \gg x_0 = 1$ for visual purpose)

Seeing is Believing



worked like a charm with < 20 lines!!

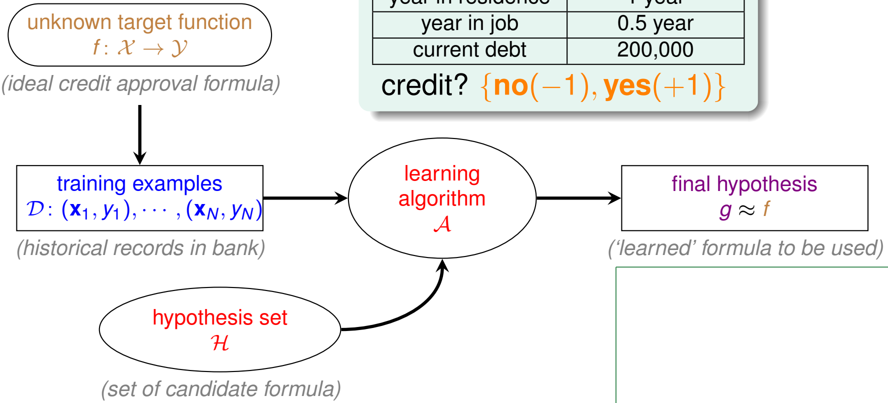
(note: made $x_i \gg x_0 = 1$ for visual purpose)

Types of Learning

Credit Approval Problem Revisited

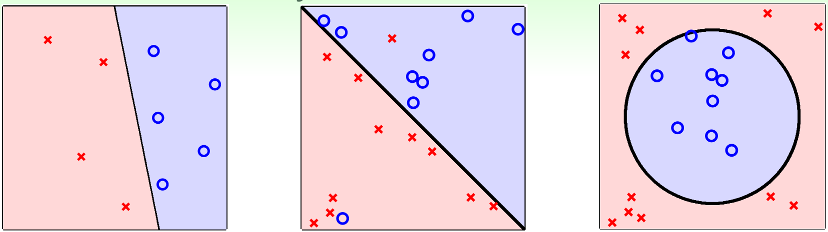
age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000

credit? $\{$**no**(-1), **yes**$(+1)\}$



$\mathcal{Y} = \{-1, +1\}$: **binary classification**

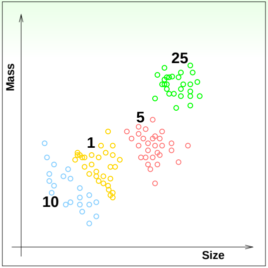
More Binary Classification Problems



- credit approve/disapprove
- email spam/non-spam
- patient sick/not sick
- ad profitable/not profitable
- answer correct/incorrect (KDDCup 2010)

core and important problem with
many tools as **building block of other tools**

Multiclass Classification: Coin Recognition Problem



- classify US coins (1c, 5c, 10c, 25c) by (size, mass)
- $\mathcal{Y} = \{1c, 5c, 10c, 25c\}$, or
  $\mathcal{Y} = \{1, 2, \cdots, K\}$ **(abstractly)**
- binary classification: special case with $K = 2$

Other Multiclass Classification Problems

- written digits $\Rightarrow 0, 1, \cdots, 9$
- pictures $\Rightarrow$ apple, orange, strawberry
- emails $\Rightarrow$ spam, primary, social, promotion, update (Google)

many applications in practice,
especially for ‘recognition’

Regression: Patient Recovery Prediction Problem

- binary classification: patient features ⇒ sick or not
- multiclass classification: patient features ⇒ which type of cancer
- regression: patient features ⇒ **how many days before recovery**
- $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = [\text{lower}, \text{upper}] \subset \mathbb{R}$ (bounded regression)
  —**deeply studied in statistics**

Other Regression Problems

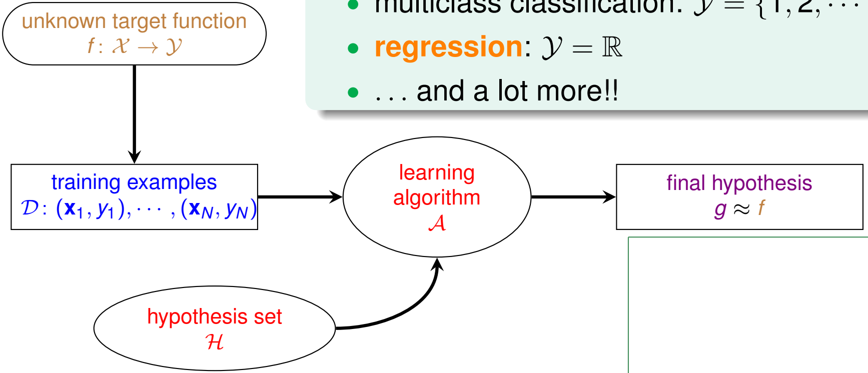
- company data ⇒ stock price
- climate data ⇒ temperature

also core and important with many 'statistical' tools as **building block of other tools**

Mini Summary

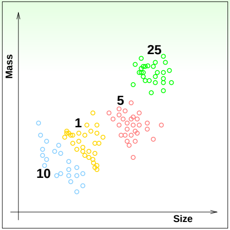
Learning with Different Output Space $\mathcal{Y}$

- **binary classification**: $\mathcal{Y} = \{-1, +1\}$
- multiclass classification: $\mathcal{Y} = \{1, 2, \cdots, K\}$
- **regression**: $\mathcal{Y} = \mathbb{R}$
- ... and a lot more!!



core tools: binary classification and regression

Supervised: Coin Recognition Revisited



unknown target function
$f: \mathcal{X} \rightarrow \mathcal{Y}$

training examples
$\mathcal{D}: (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$

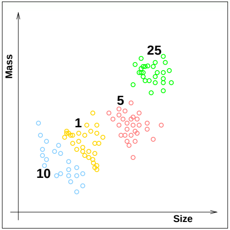
learning algorithm
$\mathcal{A}$

final hypothesis
$g \approx f$

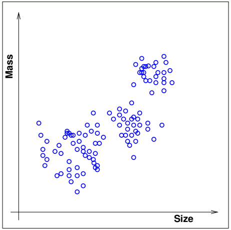
hypothesis set
$\mathcal{H}$

supervised learning:
every $\mathbf{x}_n$ **comes with corresponding** y_n

Unsupervised: Coin Recognition without y_n



supervised multiclass classification



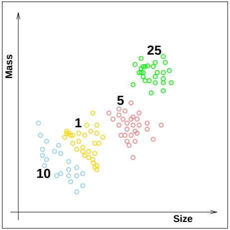
unsupervised multiclass classification
$\Longleftrightarrow$ '**clustering**'

Other Clustering Problems

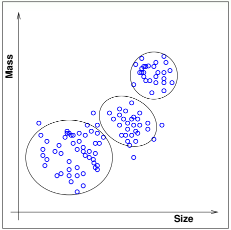
- articles $\Rightarrow$ topics
- consumer profiles $\Rightarrow$ consumer groups

clustering: a challenging but useful problem

Unsupervised: Coin Recognition without y_n



supervised multiclass classification



unsupervised multiclass classification
$\Longleftrightarrow$ '**clustering**'

Other Clustering Problems

- articles $\Rightarrow$ topics
- consumer profiles $\Rightarrow$ consumer groups

clustering: a challenging but useful problem

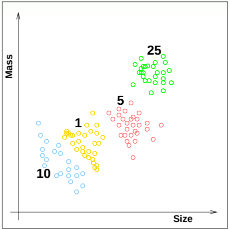
Unsupervised: Learning without y_n

Other Unsupervised Learning Problems

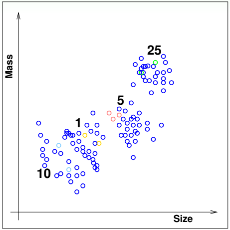
- clustering: $\{\mathbf{x}_n\} \Rightarrow \text{cluster}(\mathbf{x})$
  ($\approx$ 'unsupervised multiclass classification')
  —i.e. articles $\Rightarrow$ topics
- **density estimation**: $\{\mathbf{x}_n\} \Rightarrow \text{density}(\mathbf{x})$
  ($\approx$ 'unsupervised bounded regression')
  —i.e. traffic reports with location $\Rightarrow$ dangerous areas
- **outlier detection**: $\{\mathbf{x}_n\} \Rightarrow \text{unusual}(\mathbf{x})$
  ($\approx$ extreme 'unsupervised binary classification')
  —i.e. Internet logs $\Rightarrow$ intrusion alert
- . . . and a lot more!!

unsupervised learning: diverse, with possibly very different performance goals

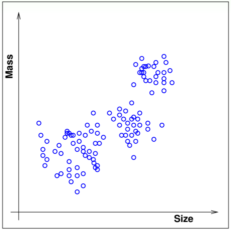
Semi-supervised: Coin Recognition with Some y_n



supervised



semi-supervised



unsupervised (clustering)

Other Semi-supervised Learning Problems

- face images with a few labeled ⇒ face identifier (Facebook)
- medicine data with a few labeled ⇒ medicine effect predictor

semi-supervised learning: leverage unlabeled data to avoid 'expensive' labeling

Reinforcement Learning

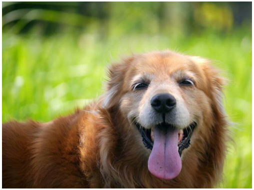
a 'very different' but natural way of learning

Teach Your Dog: Say 'Sit Down'

The dog pees on the ground.

BAD DOG. THAT'S A VERY WRONG ACTION.

- cannot easily show the dog that y_n = sit when $\mathbf{x}_n =$ 'sit down'
- but can 'punish' to say $\tilde{y}_n$ = pee is wrong



Other Reinforcement Learning Problems Using $(\mathbf{x}, \tilde{y}, \text{goodness})$

- (customer, ad choice, ad click earning) $\Rightarrow$ ad system
- (cards, strategy, winning amount) $\Rightarrow$ black jack agent

reinforcement: learn with **'partial/implicit information'** (often sequentially)

Reinforcement Learning

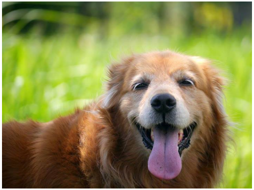
a 'very different' but natural way of learning

Teach Your Dog: Say 'Sit Down'

The dog sits down.

Good Dog. Let me give you some cookies.

- still cannot show y_n = sit when $\mathbf{x}_n$ = 'sit down'
- but can 'reward' to say $\tilde{y}_n$ = sit is good



Other Reinforcement Learning Problems Using $(\mathbf{x}, \tilde{y}, \text{goodness})$

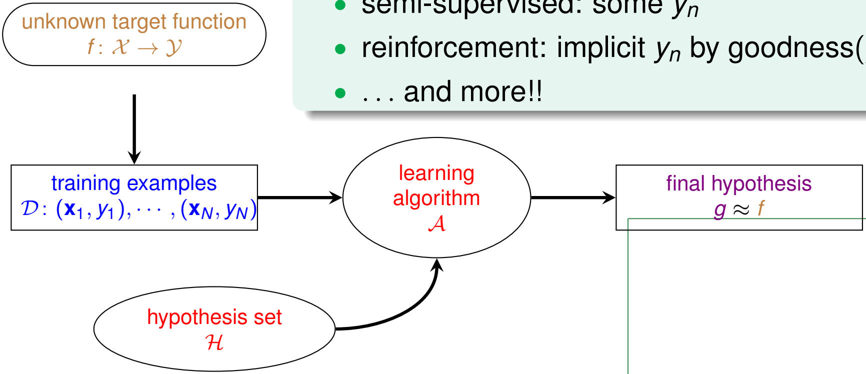
- (customer, ad choice, ad click earning) ⇒ ad system
- (cards, strategy, winning amount) ⇒ black jack agent

reinforcement: learn with **'partial/implicit information'** (often sequentially)

Mini Summary

Learning with Different Data Label y_n

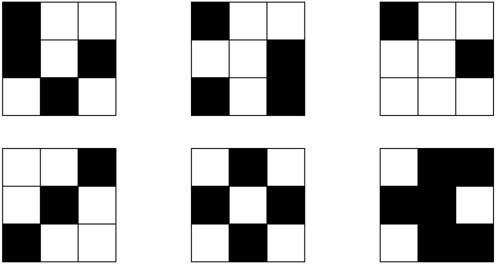
- **supervised**: all y_n
- unsupervised: no y_n
- semi-supervised: some y_n
- reinforcement: implicit y_n by goodness($\tilde{y}_n$)
- ... and more!!



core tool: supervised learning

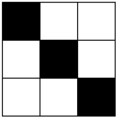
Possibility of Learning

A Learning Puzzle



$y_n = -1$

$y_n = +1$

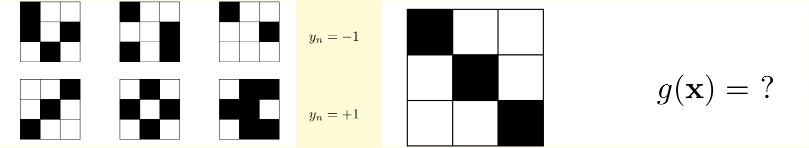


$g(\mathbf{x}) = ?$

let's test your 'human learning' with 6 examples :-)

Two Controversial Answers

whatever you say about $g(\mathbf{x})$,



truth $f(\mathbf{x}) = +1$ because . . .

- symmetry ⇔ +1
- (black or white count = 3) or (black count = 4 and middle-top black) ⇔ +1

truth $f(\mathbf{x}) = -1$ because . . .

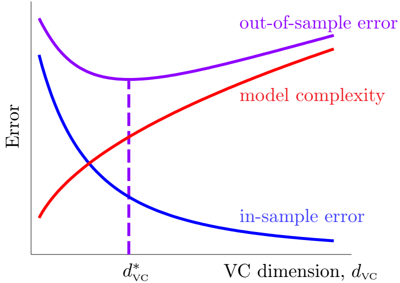
- left-top black ⇔ -1
- middle column contains at most 1 black and right-top white ⇔ -1

all valid reasons, your **adversarial teacher** can always call you '**didn't learn**'. **:-(**

Theoretical Foundation of Statistical Learning

if training and testing from same distribution, with a high probability,

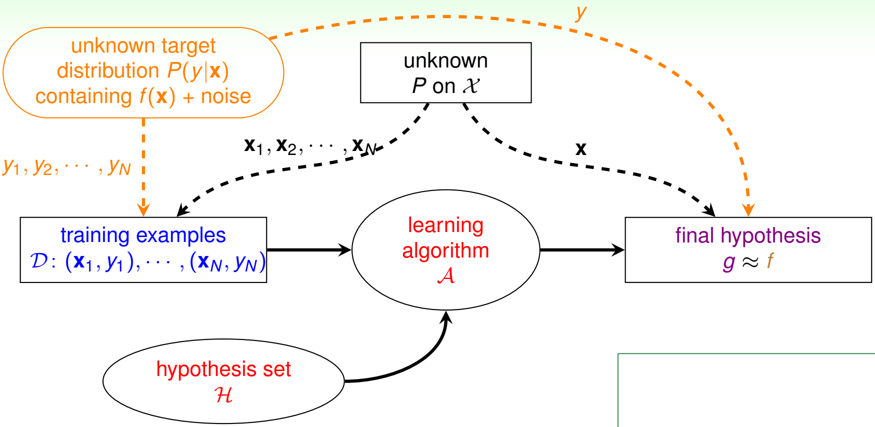
$$\underbrace{E_{\text{out}}(g)}_{\text{test error}} \leq \underbrace{E_{\text{in}}(g)}_{\text{training error}} + \underbrace{\sqrt{\frac{8}{N}\ln\left(\frac{4(2N)^{d_{\text{VC}}(\mathcal{H})}}{\delta}\right)}}_{\Omega\text{:price of using } \mathcal{H}}$$



- $d_{\text{VC}}(\mathcal{H})$: VC dimension of $\mathcal{H}$ $\approx$ # of parameters to describe $\mathcal{H}$
- $d_{\text{VC}} \uparrow$: $E_{\text{in}} \downarrow$ but $\Omega \uparrow$
- $d_{\text{VC}} \downarrow$: $\Omega \downarrow$ but $E_{\text{in}} \uparrow$
- best d^*_{VC} in the middle

powerful $\mathcal{H}$ not always good!

The New Learning Flow



if control complexity of $\mathcal{H}$ properly and minimize E_{in}, **learning possible** :-)

Linear Regression

Credit **Limit** Problem

age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000

credit limit? **100,000**

unknown target function
$f: \mathcal{X} \to \mathcal{Y}$

*(ideal credit **limit** formula)*

training examples
$\mathcal{D}: (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$

(historical records in bank)

learning algorithm
$\mathcal{A}$

final hypothesis
$g \approx f$

('learned' formula to be used)

hypothesis set
$\mathcal{H}$

(set of candidate formula)

$\mathcal{Y} = \mathbb{R}$: **regression**

Linear Regression Hypothesis

age	23 years
annual salary	NTD 1,000,000
year in job	0.5 year
current debt	200,000

- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of customer', approximate the desired credit limit with a weighted sum:

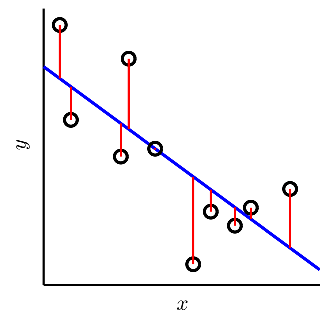
$$y \approx \sum_{i=0}^{d} w_i x_i$$

- linear regression hypothesis: $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

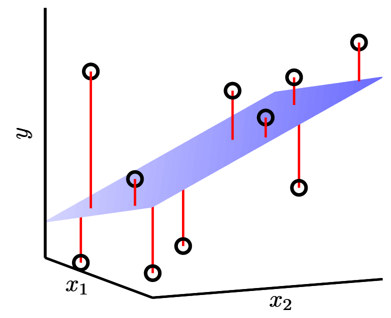
$h(\mathbf{x})$: like **perceptron**, but without the sign

Illustration of Linear Regression

$\mathbf{x} = (x) \in \mathbb{R}$



$\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$



linear regression:
find lines/hyperplanes with small residuals

The Error Measure

popular/historical error measure:

squared error $\text{err}(\hat{y}, y) = (\hat{y} - y)^2$

in-sample

$$E_{\text{in}}(h\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\underbrace{h(\mathbf{x}_n)}_{\mathbf{w}^T\mathbf{x}_n} - y_n)^2$$

out-of-sample

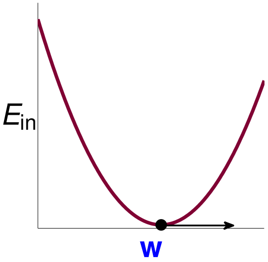
$$E_{\text{out}}(\mathbf{w}) = \underset{(\mathbf{x},y)\sim P}{\mathcal{E}}(\mathbf{w}^T\mathbf{x} - y)^2$$

next: how to minimize $E_{\text{in}}(\mathbf{w})$?

Matrix Form of $E_{\text{in}}(\mathbf{w})$

$$
\begin{aligned}
E_{\text{in}}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^T\mathbf{w} - y_n)^2 \\
&= \frac{1}{N}\left\|\begin{matrix} \mathbf{x}_1^T\mathbf{w} - y_1 \\ \mathbf{x}_2^T\mathbf{w} - y_2 \\ \dots \\ \mathbf{x}_N^T\mathbf{w} - y_N \end{matrix}\right\|^2 \\
&= \frac{1}{N}\left\|\begin{bmatrix} --\mathbf{x}_1^T-- \\ --\mathbf{x}_2^T-- \\ \dots \\ --\mathbf{x}_N^T-- \end{bmatrix}\mathbf{w} - \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix}\right\|^2 \\
&= \frac{1}{N}\|\underbrace{\mathrm{X}}_{N\times d+1}\underbrace{\mathbf{w}}_{d+1\times 1} - \underbrace{\mathbf{y}}_{N\times 1}\|^2
\end{aligned}
$$

$$\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$



- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, **convex**
- necessary condition of 'best' $\mathbf{w}$

$$\nabla E_{\text{in}}(\mathbf{w}) \equiv \begin{bmatrix} \frac{\partial E_{\text{in}}}{\partial w_0}(\mathbf{w}) \\ \frac{\partial E_{\text{in}}}{\partial w_1}(\mathbf{w}) \\ \cdots \\ \frac{\partial E_{\text{in}}}{\partial w_d}(\mathbf{w}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$

—not possible to 'roll down'

task: find $\mathbf{w}_{\text{LIN}}$ such that $\nabla E_{\text{in}}(\mathbf{w}_{\text{LIN}}) = \mathbf{0}$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2 = \frac{1}{N}\left(\mathbf{w}^T \underbrace{\mathrm{X}^T\mathrm{X}}_{\mathrm{A}}\mathbf{w} - 2\mathbf{w}^T \underbrace{\mathrm{X}^T\mathbf{y}}_{\mathbf{b}} + \underbrace{\mathbf{y}^T\mathbf{y}}_{c}\right)$$

one w only

$$E_{\text{in}}(w) = \frac{1}{N}\left(aw^2 - 2bw + c\right)$$

$$\nabla E_{\text{in}}(w) = \frac{1}{N}\left(2aw - 2b\right)$$

simple! :-)

vector $\mathbf{w}$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\left(\mathbf{w}^T\mathrm{A}\mathbf{w} - 2\mathbf{w}^T\mathbf{b} + c\right)$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\left(2\mathrm{A}\mathbf{w} - 2\mathbf{b}\right)$$

similar (**derived by definition**)

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}\left(\mathrm{X}^T\mathrm{X}\mathbf{w} - \mathrm{X}^T\mathbf{y}\right)$$

Optimal Linear Regression Weights

task: find $\mathbf{w}_{\text{LIN}}$ such that $\frac{2}{N}\left(\mathrm{X}^T\mathrm{X}\mathbf{w} - \mathrm{X}^T\mathbf{y}\right) = \nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$

invertible $\mathrm{X}^T\mathrm{X}$

- **easy!** unique solution

$$\mathbf{w}_{\text{LIN}} = \underbrace{\left(\mathrm{X}^T\mathrm{X}\right)^{-1}\mathrm{X}^T}_{\text{pseudo-inverse } \mathrm{X}^\dagger}\mathbf{y}$$

- often the case because $N \gg d+1$

singular $\mathrm{X}^T\mathrm{X}$

- **many** optimal solutions
- one of the solutions

$$\mathbf{w}_{\text{LIN}} = \mathrm{X}^\dagger\mathbf{y}$$

by defining $\mathrm{X}^\dagger$ in other ways

practical suggestion:

use **well-implemented $\dagger$ routine**

instead of $\left(\mathrm{X}^T\mathrm{X}\right)^{-1}\mathrm{X}^T$

for numerical stability when **almost-singular**

Linear Regression Algorithm

1. from $\mathcal{D}$, construct **input matrix** X and **output vector** $\mathbf{y}$ by

$$\mathrm{X} = \underbrace{\begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ & \cdots & \\ -- & \mathbf{x}_N^T & -- \end{bmatrix}}_{N\times(d+1)} \quad \mathbf{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}}_{N\times 1}$$

2. calculate pseudo-inverse $\underbrace{\mathrm{X}^\dagger}_{(d+1)\times N}$

3. return $\underbrace{\mathbf{w}_{\text{LIN}}}_{(d+1)\times 1} = \mathrm{X}^\dagger\mathbf{y}$

simple and efficient
with **good † routine**

Is Linear Regression a 'Learning Algorithm'?

$$\mathbf{w}_{\text{LIN}} = \mathrm{X}^{\dagger}\mathbf{y}$$

No!

- analytic (**closed-form**) solution, 'instantaneous'
- not improving E_{in} nor E_{out} iteratively

Yes!

- good E_{in}? **yes, optimal!**
- good E_{out}? **yes, finite d_{VC} like perceptrons**
- improving iteratively? **somewhat, within an iterative pseudo-inverse routine**

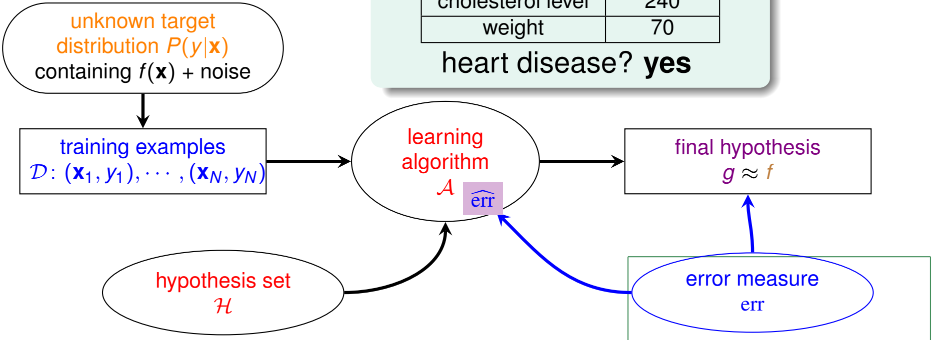
if $E_{\text{out}}(\mathbf{w}_{\text{LIN}})$ is good, **learning 'happened'**!

Logistic Regression

Heart Attack Prediction Problem (1/2)

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240
weight	70

heart disease? **yes**



binary classification:

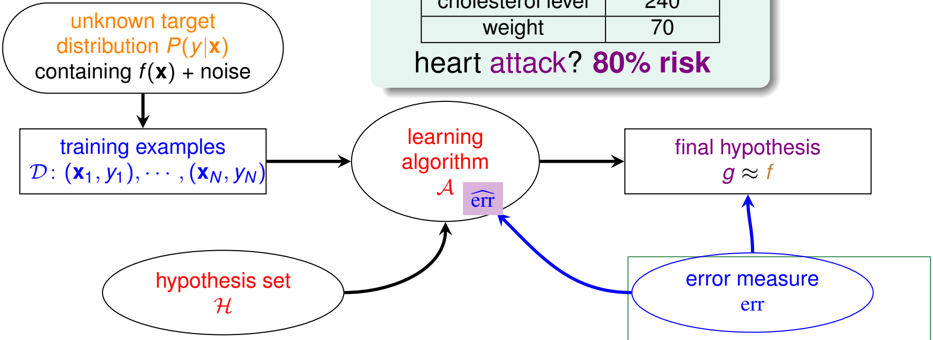
$$\text{ideal } f(\mathbf{x}) = \text{sign}\left(P(+1|\mathbf{x}) - \frac{1}{2}\right) \in \{-1, +1\}$$

because of classification err

Heart Attack Prediction Problem (2/2)

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240
weight	70

heart attack? **80% risk**



'soft' binary classification:

$$f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$$

Soft Binary Classification

target function $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$

ideal (noiseless) data

$$\begin{array}{lll} (\mathbf{x}_1, y'_1 & = 0.9 & = P(+1|\mathbf{x}_1)) \\ (\mathbf{x}_2, y'_2 & = 0.2 & = P(+1|\mathbf{x}_2)) \\ \vdots & & \\ (\mathbf{x}_N, y'_N & = 0.6 & = P(+1|\mathbf{x}_N)) \end{array}$$

actual (noisy) data

$$\begin{array}{lll} (\mathbf{x}_1, y_1 & = \circ & \sim P(y|\mathbf{x}_1)) \\ (\mathbf{x}_2, y_2 & = \times & \sim P(y|\mathbf{x}_2)) \\ \vdots & & \\ (\mathbf{x}_N, y_N & = \times & \sim P(y|\mathbf{x}_N)) \end{array}$$

same data as hard binary classification,
different **target function**

Soft Binary Classification

target function $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$

ideal (noiseless) data

$$\begin{aligned}
&(\mathbf{x}_1, y'_1 &&= 0.9 &&= P(+1|\mathbf{x}_1))\\
&(\mathbf{x}_2, y'_2 &&= 0.2 &&= P(+1|\mathbf{x}_2))\\
&\vdots\\
&(\mathbf{x}_N, y'_N &&= 0.6 &&= P(+1|\mathbf{x}_N))
\end{aligned}$$

actual (noisy) data

$$\begin{aligned}
&(\mathbf{x}_1, y'_1 &&= 1 &&= [\![\circ \overset{?}{\sim} P(y|\mathbf{x}_1)]\!])\\
&(\mathbf{x}_2, y'_2 &&= 0 &&= [\![\circ \overset{?}{\sim} P(y|\mathbf{x}_2)]\!])\\
&\vdots\\
&(\mathbf{x}_N, y'_N &&= 0 &&= [\![\circ \overset{?}{\sim} P(y|\mathbf{x}_N)]\!])
\end{aligned}$$

same data as hard binary classification,
different **target function**

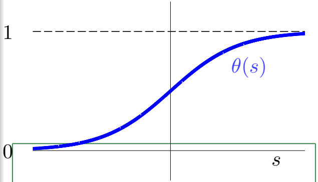
Logistic Hypothesis

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240

- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of patient', calculate a weighted 'risk score':

$$s = \sum_{i=0}^{d} w_i x_i$$

- convert the score to estimated probability by logistic function $\theta(s)$



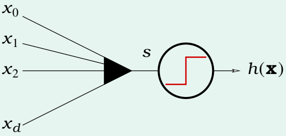
logistic hypothesis:
$$h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$$

Three Linear Models

linear scoring function: $s = \mathbf{w}^T\mathbf{x}$

linear classification

$$h(\mathbf{x}) = \text{sign}(s)$$

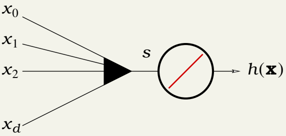


plausible err = 0/1

(small flipping noise)

linear regression

$$h(\mathbf{x}) = s$$

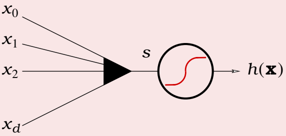


friendly err = squared

(easy to minimize)

logistic regression

$$h(\mathbf{x}) = \theta(s)$$



err = ?

how to define

$E_{\text{in}}(\mathbf{w})$ **for logistic regression**?

Likelihood

target function
$f(\mathbf{x}) = P(+1|\mathbf{x})$
$\Leftrightarrow$
$$P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases}$$

consider $\mathcal{D} = \{(\mathbf{x}_1, \circ), (\mathbf{x}_2, \times), \ldots, (\mathbf{x}_N, \times)\}$

probability that f generates $\mathcal{D}$

$$P(\mathbf{x}_1)P(\circ|\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)P(\times|\mathbf{x}_2) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)P(\times|\mathbf{x}_N)$$

likelihood that h generates $\mathcal{D}$

$$P(\mathbf{x}_1)h(\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

- if $h \approx f$,
  then likelihood(h) $\approx$ probability using f
- probability using f usually **large**

Likelihood

target function
$f(\mathbf{x}) = P(+1|\mathbf{x})$ $\Leftrightarrow$ $P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases}$

consider $\mathcal{D} = \{(\mathbf{x}_1, \circ), (\mathbf{x}_2, \times), \ldots, (\mathbf{x}_N, \times)\}$

probability that f generates $\mathcal{D}$

$$P(\mathbf{x}_1)f(\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)(1 - f(\mathbf{x}_2)) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)(1 - f(\mathbf{x}_N))$$

likelihood that h generates $\mathcal{D}$

$$P(\mathbf{x}_1)h(\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

- if $h \approx f$,
  then likelihood(h) $\approx$ probability using f
- probability using f usually **large**

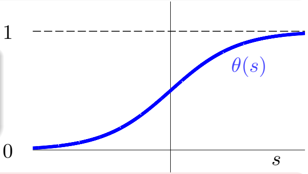
Likelihood of Logistic Hypothesis

likelihood(h) $\approx$ (probability using f) $\approx$ **large**

$$g = \underset{h}{\text{argmax}} \ \text{likelihood}(h)$$

when logistic: $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$



$$\text{likelihood}(h) = P(\mathbf{x}_1)h(\mathbf{x}_1) \times P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times \ldots P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

$$\text{likelihood}(\text{logistic } h) \propto \prod_{n=1}^{N} h(y_n \mathbf{x}_n)$$

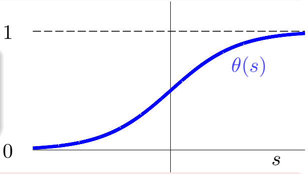
Likelihood of Logistic Hypothesis

likelihood(h) $\approx$ (probability using f) $\approx$ **large**

$$g = \underset{h}{\text{argmax}} \ \text{likelihood}(h)$$

when logistic: $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$



$$\text{likelihood}(h) = P(\mathbf{x}_1)h(+\mathbf{x}_1) \times P(\mathbf{x}_2)h(-\mathbf{x}_2) \times \ldots P(\mathbf{x}_N)h(-\mathbf{x}_N)$$

$$\text{likelihood}(\text{logistic } h) \propto \prod_{n=1}^{N} h(y_n\mathbf{x}_n)$$

Cross-Entropy Error

$$\max_{h} \quad \text{likelihood(logistic } h) \propto \prod_{n=1}^{N} h(y_n \mathbf{x}_n)$$

Cross-Entropy Error

$$\max_{\mathbf{w}} \quad \text{likelihood}(\mathbf{w}) \propto \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

Cross-Entropy Error

$$\max_{\mathbf{w}} \quad \ln \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

Cross-Entropy Error

$$\min_{\mathbf{w}} \frac{1}{N}\sum_{n=1}^{N} - \ln \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

$$\theta(s) = \frac{1}{1+\exp(-s)} \quad : \quad \min_{\mathbf{w}} \frac{1}{N}\sum_{n=1}^{N} \ln\left(1+\exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$

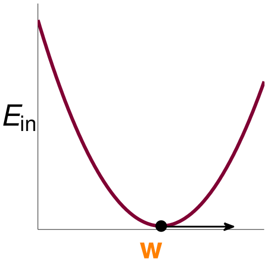
$$\Longrightarrow \min_{\mathbf{w}} \underbrace{\frac{1}{N}\sum_{n=1}^{N} \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n)}_{E_{\text{in}}(\mathbf{w})}$$

$\mathrm{err}(\mathbf{w}, \mathbf{x}, y) = \ln\left(1+\exp(-y\mathbf{w}\mathbf{x})\right)$:

cross-entropy error

Minimizing $E_{in}(\mathbf{w})$

$$\min_{\mathbf{w}} \quad E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$



- $E_{in}(\mathbf{w})$: continuous, differentiable, twice-differentiable, **convex**
- how to minimize? locate **valley**

$$\text{want } \nabla E_{in}(\mathbf{w}) = \mathbf{0}$$

first: derive $\nabla E_{in}(\mathbf{w})$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(\underbrace{1+\exp(\overbrace{-y_n\mathbf{w}^T\mathbf{x}_n}^{\bigcirc})}_{\square}\right)$$

$$\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial \ln(\square)}{\partial \square}\right)\left(\frac{\partial(1+\exp(\bigcirc))}{\partial \bigcirc}\right)\left(\frac{\partial -y_n\mathbf{w}^T\mathbf{x}_n}{\partial w_i}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\quad\right)\left(\quad\right)\left(\quad\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\exp(\bigcirc)}{1+\exp(\bigcirc)}\right)\left(-y_n x_{n,i}\right) = \frac{1}{N}\sum_{n=1}^{N}\theta(\bigcirc)\left(-y_n x_{n,i}\right)$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\theta\left(-y_n\mathbf{w}^T\mathbf{x}_n\right)\left(-y_n\mathbf{x}_n\right)$$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(\underbrace{1+\exp(\overbrace{-y_n\mathbf{w}^T\mathbf{x}_n}^{\bigcirc})}_{\square}\right)$$

$$\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial \ln(\square)}{\partial \square}\right)\left(\frac{\partial(1+\exp(\bigcirc))}{\partial \bigcirc}\right)\left(\frac{\partial - y_n\mathbf{w}^T\mathbf{x}_n}{\partial w_i}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{\square}\right)\left(\exp(\bigcirc)\right)\left(-y_n x_{n,i}\right)$$

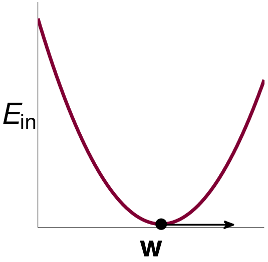
$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\exp(\bigcirc)}{1+\exp(\bigcirc)}\right)\left(-y_n x_{n,i}\right) = \frac{1}{N}\sum_{n=1}^{N}\theta(\bigcirc)\left(-y_n x_{n,i}\right)$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\theta\left(-y_n\mathbf{w}^T\mathbf{x}_n\right)\left(-y_n\mathbf{x}_n\right)$$

Minimizing $E_{\text{in}}(\mathbf{w})$

$$\min_{\mathbf{w}} \quad E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$

$$\text{want } \nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right) = \mathbf{0}$$



scaled θ-weighted sum of $-y_n\mathbf{x}_n$

- all $\theta(\cdot) = 0$: only if $y_n \mathbf{w}^T \mathbf{x}_n \gg 0$ —linear separable $\mathcal{D}$
- weighted sum = **0**: non-linear equation of **w**

closed-form solution? no :-(

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

when stop, return last $\mathbf{w}$ as g

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

1. (equivalently) pick some n, and update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \left[\!\left[\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n\right]\!\right] y_n \mathbf{x}_n$$

when stop, return last $\mathbf{w}$ as g

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. (equivalently) pick some n, and update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \underbrace{1}_{\eta} \cdot \underbrace{\left(\left[\!\left[\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n \right]\!\right] \cdot y_n \mathbf{x}_n \right)}_{\mathbf{v}}$$

when stop, return last $\mathbf{w}$ as g

choice of $(\eta, \mathbf{v})$ and stopping condition defines **iterative optimization approach**

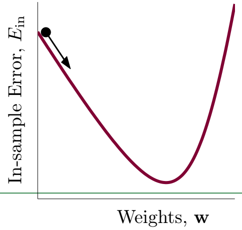
Iterative Optimization

For $t = 0, 1, \ldots$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last $\mathbf{w}$ as g

- PLA: $\mathbf{v}$ comes from mistake correction
- smooth $E_{\text{in}}(\mathbf{w})$ for logistic regression: choose $\mathbf{v}$ to get the ball roll '**downhill**'?
  - direction $\mathbf{v}$: (assumed) of unit length
  - step size η: (assumed) positive



a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\underbrace{\mathbf{w}_t + \eta \mathbf{v}}_{\mathbf{w}_{t+1}})$$

Linear Approximation

a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\mathbf{w}_t + \eta \mathbf{v})$$

- still non-linear optimization, now **with constraints**
  —not any easier than $\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w})$
- local approximation by linear formula makes problem easier

$$E_{\text{in}}(\mathbf{w}_t + \eta \mathbf{v}) \approx E_{\text{in}}(\mathbf{w}_t) + \eta \mathbf{v}^T \nabla E_{\text{in}}(\mathbf{w}_t)$$

if η really small (Taylor expansion)

an **approximate** greedy approach for some given **small** η:

$$\min_{\|\mathbf{v}\|=1} \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

Gradient Descent

an **approximate** greedy approach for some given **small** η:

$$\min_{\|\mathbf{v}\|=1} \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

- optimal $\mathbf{v}$: opposite direction of $\nabla E_{\text{in}}(\mathbf{w}_t)$

$$\mathbf{v} \propto -\nabla E_{\text{in}}(\mathbf{w}_t)$$

- fixed learning-rate gradient descent: for **small** η,
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$

gradient descent:
a simple & popular optimization tool

Putting Everything Together

Logistic Regression Algorithm

initialize $\mathbf{w}_0$

For $t = 0, 1, \cdots$

1. compute

$$\nabla E_{\text{in}}(\mathbf{w}_t) = \frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right)$$

2. update by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

...until $\nabla E_{\text{in}}(\mathbf{w}_{t+1}) = 0$ or enough iterations

return last $\mathbf{w}_{t+1}$ as g

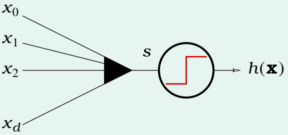
can use more sophisticated tools to speed up

Linear Models Revisited

linear scoring function: $s = \mathbf{w}^T \mathbf{x}$

linear classification

$$h(\mathbf{x}) = \text{sign}(s)$$

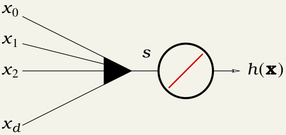


plausible err = 0/1

discrete $E_{\text{in}}(\mathbf{w})$:
solvable in special case

linear regression

$$h(\mathbf{x}) = s$$

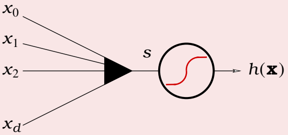


friendly err = squared

quadratic convex $E_{\text{in}}(\mathbf{w})$:
closed-form solution

logistic regression

$$h(\mathbf{x}) = \theta(s)$$



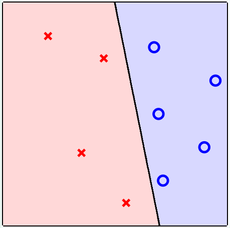
plausible err = cross-entropy

smooth convex $E_{\text{in}}(\mathbf{w})$:
gradient descent

Nonlinear Transform

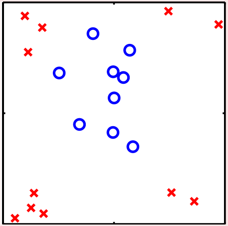
Linear Hypotheses

up to now: linear hypotheses



- visually: **'line'-like** boundary
- mathematically: linear scores $s = \mathbf{w}^T\mathbf{x}$

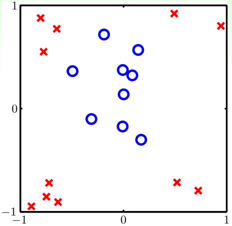
but limited ...

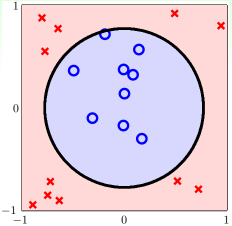


- theoretically: d_{VC} **under control :-)**
- practically: on some $\mathcal{D}$, **large** E_{in} for every line **:-(**

how to **break the limit** of linear hypotheses

Circular Separable





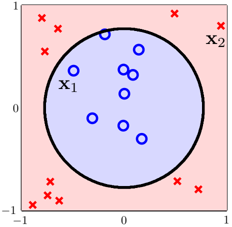
- $\mathcal{D}$ not linear separable
- but **circular separable** by a circle of radius $\sqrt{0.6}$ centered at origin:

$$h_{\text{SEP}}(\mathbf{x}) = \text{sign}\left(-x_1^2 - x_2^2 + 0.6\right)$$

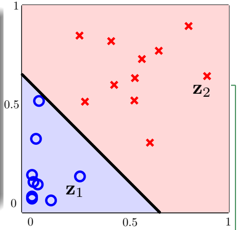
re-derive **Circular**-PLA, **Circular**-Regression, blahblah . . . all over again? **:-)**

Circular Separable and Linear Separable

$$h(\mathbf{x}) = \text{sign}\left(\underbrace{0.6}_{\tilde{w}_0} \cdot \underbrace{1}_{z_0} + \underbrace{(-1)}_{\tilde{w}_1} \cdot \underbrace{x_1^2}_{z_1} + \underbrace{(-1)}_{\tilde{w}_2} \cdot \underbrace{x_2^2}_{z_2} \right)$$
$$= \text{sign}\left(\tilde{\mathbf{w}}^T \mathbf{z}\right)$$



- $\{(\mathbf{x}_n, y_n)\}$ circular separable $\Longrightarrow \{(\mathbf{z}_n, y_n)\}$ linear separable
- $\mathbf{x} \in \mathcal{X} \overset{\mathbf{\Phi}}{\longmapsto} \mathbf{z} \in \mathcal{Z}$: **(nonlinear) feature transform $\mathbf{\Phi}$**



circular separable in $\mathcal{X} \Longrightarrow$ linear separable in $\mathcal{Z}$

vice versa?

Linear Hypotheses in $\mathcal{Z}$-Space

$$(z_0, z_1, z_2) = \mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = (1, x_1^2, x_2^2)$$

$$h(\mathbf{x}) = \tilde{h}(\mathbf{z}) = \text{sign}\left(\tilde{\mathbf{w}}^T \mathbf{\Phi}(\mathbf{x})\right) = \text{sign}\left(\tilde{w}_0 + \tilde{w}_1 x_1^2 + \tilde{w}_2 x_2^2\right)$$

$\tilde{\mathbf{w}} = (\tilde{w}_0, \tilde{w}_1, \tilde{w}_2)$

- $(0.6, -1, -1)$: circle (∘ inside)
- $(-0.6, +1, +1)$: circle (∘ outside)
- $(0.6, -1, -2)$: ellipse
- $(0.6, -1, +2)$: hyperbola
- $(0.6, +1, +2)$: **constant ∘ :-)**

lines in $\mathcal{Z}$-space
$\Longleftrightarrow$ **special** quadratic curves in $\mathcal{X}$-space

General Quadratic Hypothesis Set

a 'bigger' $\mathcal{Z}$-space with $\mathbf{\Phi}_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$

perceptrons in $\mathcal{Z}$-space $\Longleftrightarrow$ quadratic hypotheses in $\mathcal{X}$-space

$$\mathcal{H}_{\mathbf{\Phi}_2} = \left\{ h(\mathbf{x}) \colon h(\mathbf{x}) = \tilde{h}(\mathbf{\Phi}_2(\mathbf{x})) \text{ for some linear } \tilde{h} \text{ on } \mathcal{Z} \right\}$$

- can implement **all possible quadratic curve boundaries**: circle, ellipse, **rotated** ellipse, hyperbola, parabola, . . .

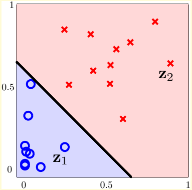
  ellipse $2(x_1 + x_2 - 3)^2 + (x_1 - x_2 - 4)^2 = 1$

  $\Longleftarrow \tilde{\mathbf{w}}^T = [33, -20, -4, 3, 2, 3]$
- include **lines and constants as degenerate cases**

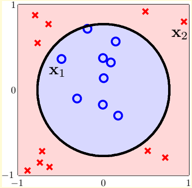
next: **learn** a good quadratic hypothesis g

Good Quadratic Hypothesis

$\mathcal{Z}$-space		$\mathcal{X}$-space
perceptrons	$\Longleftrightarrow$	quadratic hypotheses
good perceptron	$\Longleftrightarrow$	**good quadratic hypothesis**
separating perceptron	$\Longleftrightarrow$	**separating quadratic hypothesis**



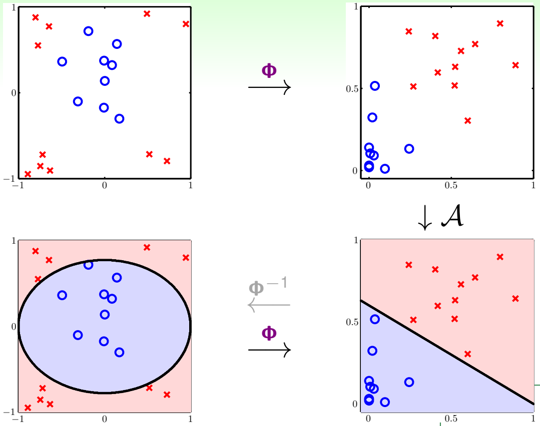
$\Longleftrightarrow$



- want: get **good perceptron** in $\mathcal{Z}$-space
- known: get **good perceptron** in $\mathcal{X}$-space with data $\{(\mathbf{x}_n, y_n)\}$

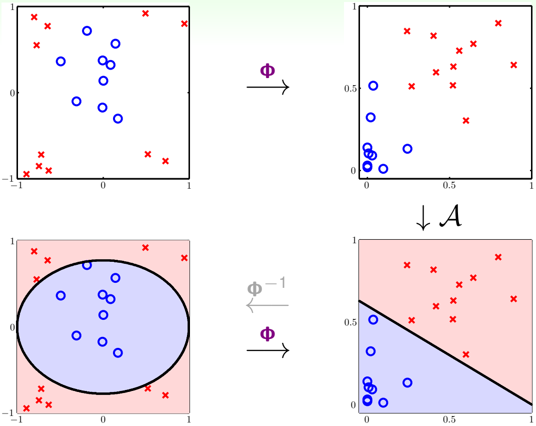
todo: get **good perceptron** in $\mathcal{Z}$-space with data $\{(\mathbf{z}_n = \mathbf{\Phi}_2(\mathbf{x}_n), y_n)\}$

The Nonlinear Transform Steps



1. transform original data $\{(\mathbf{x}_n, y_n)\}$ to $\{(\mathbf{z}_n = \mathbf{\Phi}(\mathbf{x}_n), y_n)\}$ by $\mathbf{\Phi}$
2. get a good perceptron $\tilde{\mathbf{w}}$ using $\{(\mathbf{z}_n, y_n)\}$ and your favorite linear algorithm $\mathcal{A}$
3. return $g(\mathbf{x}) = \text{sign}\left(\tilde{\mathbf{w}}^T \mathbf{\Phi}(\mathbf{x})\right)$

Nonlinear Model via Nonlinear Φ + Linear Models



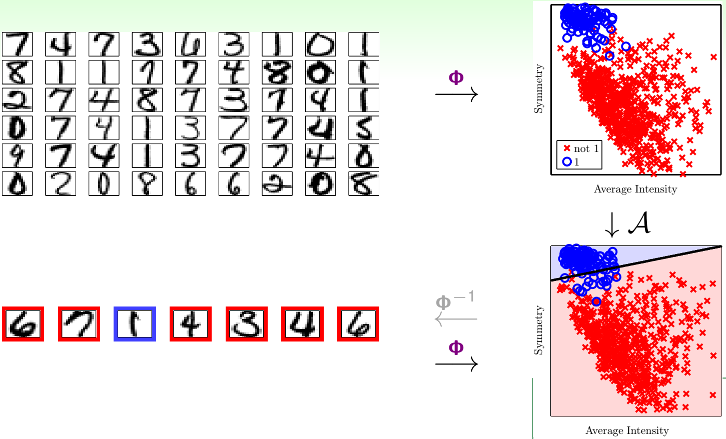
two choices:

- feature transform Φ
- linear model $\mathcal{A}$, **not just binary classification**

Pandora's box :-):

can now freely do **quadratic PLA, quadratic regression, cubic regression, . . ., polynomial regression**

Feature Transform Φ



not new, not just polynomial:

raw (pixels) $\xrightarrow{\text{domain knowledge}}$ **concrete (intensity, symmetry)**

the force, too good to be true? :-)

Computation/Storage Price

Q-th order polynomial transform: $\boldsymbol{\Phi}_Q(\mathbf{x}) = \Big($ $1,$
$x_1, x_2, \ldots, x_d,$
$x_1^2, x_1 x_2, \ldots, x_d^2,$
$\ldots,$
$x_1^Q, x_1^{Q-1} x_2, \ldots, x_d^Q \Big)$

$\underbrace{1}_{\tilde{w}_0} + \underbrace{\tilde{d}}_{\text{others}}$ dimensions

= # ways of $\leq Q$-combination from d kinds with repetitions

= $\binom{Q+d}{Q}$ = $\binom{Q+d}{d}$ = $O\left(Q^d\right)$

= efforts needed for computing/storing $\mathbf{z} = \boldsymbol{\Phi}_Q(\mathbf{x})$ and $\tilde{\mathbf{w}}$

Q large $\Longrightarrow$ **difficult to compute/store**

Model Complexity Price

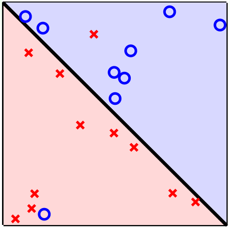
Q-th order polynomial transform: $\mathbf{\Phi}_Q(\mathbf{x}) = \Big(1, x_1, x_2, \ldots, x_d, x_1^2, x_1 x_2, \ldots, x_d^2, \ldots, x_1^Q, x_1^{Q-1} x_2, \ldots, x_d^Q \Big)$

$\underbrace{1}_{\tilde{w}_0} + \underbrace{\tilde{d}}_{\text{others}}$ dimensions = $O(Q^d)$

- number of free parameters $\tilde{w}_i = \tilde{d} + 1 \approx d_{\text{VC}}(\mathcal{H}_{\mathbf{\Phi}_Q})$

Q large $\Longrightarrow$ **large** d_{VC}

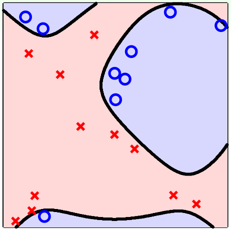
Generalization Issue



$\mathbf{\Phi}_1$ (original $\mathbf{x}$)

which one do you prefer? :-)

- $\mathbf{\Phi}_1$ 'visually' preferred
- $\mathbf{\Phi}_4$: $E_{\text{in}}(g) = 0$ but overkill



$\mathbf{\Phi}_4$

how to pick Q? **visually**, maybe?

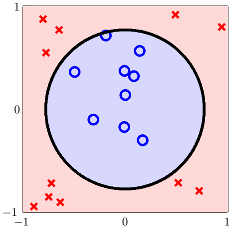
Danger of Visual Choices

first of all, can you really 'visualize' when $\mathcal{X} = \mathbb{R}^{10}$? **(well, I can't :-))**

Visualize $\mathcal{X} = \mathbb{R}^2$

- full $\mathbf{\Phi}_2$: $\mathbf{z} = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$, $d_{\text{VC}} = 6$
- or $\mathbf{z} = (1, x_1^2, x_2^2)$, $d_{\text{VC}} = 3$, **after visualizing**?
- or better $\mathbf{z} = (1, x_1^2 + x_2^2)$, $d_{\text{VC}} = 2$?
- or even better $\mathbf{z} = \left(\text{sign}(0.6 - x_1^2 - x_2^2)\right)$?

—careful about **your brain's 'model complexity'**

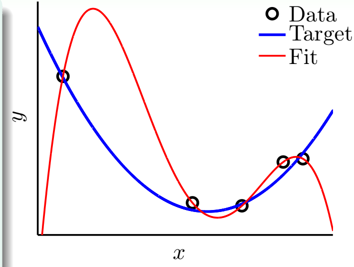


for VC-safety, $\mathbf{\Phi}$ shall be decided **without 'peeking'** data

Overfitting

Bad Generalization

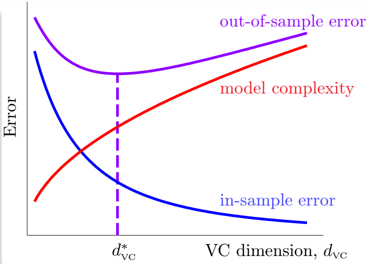
- regression for $x \in \mathbb{R}$ with $N = 5$ examples
- target $f(x)$ = 2nd order polynomial
- label $y_n = f(x_n) +$ very small noise
- linear regression in $\mathcal{Z}$-space + $\mathbf{\Phi}$ = 4th order polynomial
- unique solution passing all examples $\Longrightarrow E_{\text{in}}(g) = 0$
- $E_{\text{out}}(g)$ **huge**



bad generalization: low E_{in}, high E_{out}

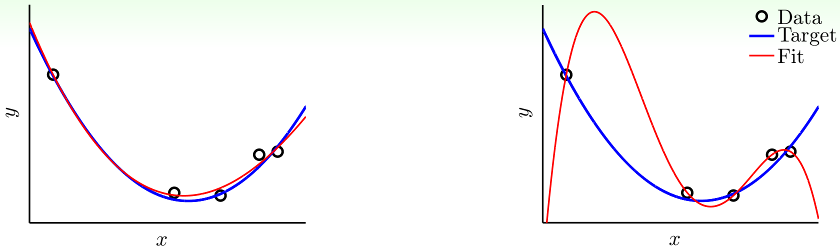
Bad Generalization and Overfitting

- take $d_{\text{VC}} = 1126$ for learning: bad generalization —(E_{out} - E_{in}) large
- switch from $d_{\text{VC}} = d^*_{\text{VC}}$ to $d_{\text{VC}} = 1126$: **overfitting** —$E_{\text{in}} \downarrow$, $E_{\text{out}} \uparrow$
- switch from $d_{\text{VC}} = d^*_{\text{VC}}$ to $d_{\text{VC}} = 1$: **underfitting** —$E_{\text{in}} \uparrow$, $E_{\text{out}} \uparrow$



bad generalization: low E_{in}, high E_{out};
overfitting: low**er** E_{in}, high**er** E_{out}

Cause of Overfitting: A Driving Analogy

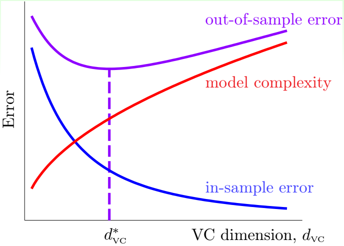


'good fit' $\Longrightarrow$ **overfit**

learning	driving
overfit	commit a car accident
use excessive d_{VC}	'drive too fast'
noise	bumpy road
limited data size *N*	limited observations about road condition

what shall we do?

Linear Model First



- tempting sin: use $\mathcal{H}_{1126}$, low $E_{\text{in}}(g_{1126})$ to fool your boss —**really? :-(a dangerous path of no return**
- safe route: $\mathcal{H}_1$ first
  - if $E_{\text{in}}(g_1)$ good enough, **live happily thereafter :-)**
  - otherwise, move right of the curve **with nothing lost except 'wasted' computation**

linear model first:
simple, efficient, **safe**, **and workable!**

Driving Analogy Revisited

learning	driving
overfit	commit a car accident
use excessive d_{VC}	'drive too fast'
noise	bumpy road
limited data size N	limited observations about road condition
start from simple model	drive slowly
regularization	put the brakes
data cleaning/pruning	use more accurate road information
data hinting	exploit more road information
validation	monitor the dashboard

all very **practical** techniques
to combat overfitting

Validation Set $\mathcal{D}_{\text{val}}$

$$\begin{array}{ccccc} E_{\text{in}}(h) & & & & E_{\text{val}}(h) \\ \uparrow & & & & \uparrow \\ \underbrace{\mathcal{D}}_{\text{size } N} & \rightarrow & \underbrace{\mathcal{D}_{\text{train}}}_{\text{size } N-K} & \cup & \underbrace{\mathcal{D}_{\text{val}}}_{\text{size } K} \\ \downarrow & & \downarrow & & \\ g_m = \mathcal{A}_m(\mathcal{D}) & & g_m^- = \mathcal{A}_m(\mathcal{D}_{\text{train}}) & & \end{array}$$

- $\mathcal{D}_{\text{val}} \subset \mathcal{D}$: called **validation set**—'on-hand' simulation of test set
- to connect E_{val} with E_{out}:
  $\mathcal{D}_{\text{val}} \overset{iid}{\sim} P(\mathbf{x}, y) \Longleftarrow$ select K examples from $\mathcal{D}$ at random
- to make sure $\mathcal{D}_{\text{val}}$ 'clean':
  feed only $\mathcal{D}_{\text{train}}$ to $\mathcal{A}_m$ for model selection

$$E_{\text{out}}(g_m^-) \le E_{\text{val}}(g_m^-) + O\left(\sqrt{\frac{\log M}{K}}\right)$$

Model Selection by Best E_{val}

$$m^* = \underset{1 \le m \le M}{\text{argmin}} (E_m = E_{\text{val}}(\mathcal{A}_m(\mathcal{D}_{\text{train}})))$$

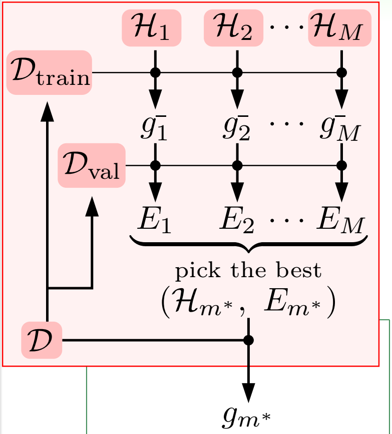
- generalization guarantee for all m:

$$E_{\text{out}}(g_m^-) \le E_{\text{val}}(g_m^-) + O\left(\sqrt{\frac{\log M}{K}}\right)$$

- heuristic gain from $N - K$ to N:

$$E_{\text{out}}\left(\underbrace{g_{m^*}}_{\mathcal{A}_{m^*}(\mathcal{D})}\right) \le E_{\text{out}}\left(\underbrace{g_{m^*}^-}_{\mathcal{A}_{m^*}(\mathcal{D}_{\text{train}})}\right)$$

—**learning curve, remember? :-)**

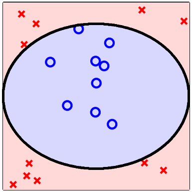


$$E_{\text{out}}(g_{m^*}) \le E_{\text{out}}(g_{m^*}^-) \le E_{\text{val}}(g_{m^*}^-) + O\left(\sqrt{\frac{\log M}{K}}\right)$$

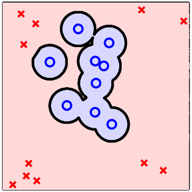
Principles of Learning

Occam's Razor for Learning

The simplest model that fits the data is also the most plausible.



which one do you prefer? :-)



Sampling Bias

If the data is sampled in a biased way, learning will produce a similarly biased outcome.

- technical explanation:
  data from $P_1(\mathbf{x}, y)$ but test under $P_2 \neq P_1$: **VC fails**
- philosophical explanation:
  study Math hard but test English: **no strong test guarantee**

practical rule of thumb:
match test scenario as much as possible

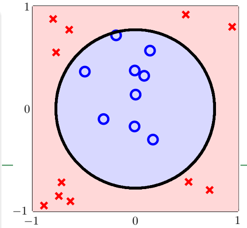
Visual Data Snooping

If a data set has affected any step in the learning process, its ability to assess the outcome has been compromised.

Visualize $\mathcal{X} = \mathbb{R}^2$

- full $\mathbf{\Phi}_2$: $\mathbf{z} = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$, $d_{\text{VC}} = 6$
- or $\mathbf{z} = (1, x_1^2, x_2^2)$, $d_{\text{VC}} = 3$, **after visualizing**?
- or better $\mathbf{z} = (1, x_1^2 + x_2^2)$, $d_{\text{VC}} = 2$?
- or even better $\mathbf{z} = (\text{sign}(0.6 - x_1^2 - x_2^2))$?

—careful about **your brain's 'model complexity'**



if you torture the data long enough, it will confess :-)

Dealing with Data Snooping

- truth—**very hard to avoid**, unless being extremely honest
- extremely honest: **lock your test data in safe**
- less honest: **reserve validation and use cautiously**

- be blind: avoid **making modeling decision by data**
- be suspicious: interpret research results (including your own) by proper **feeling of contamination**

one secret to winning KDDCups:

careful balance between
data-driven modeling (snooping) and
validation (no-snooping)

Summary

- What is Machine Learning
  use data to approximate unknown target
- Perceptron Learning Algorithm
  correct by mistake
- Types of Learning
  classification/regression; [un-]supervised/reinforcement
- Possibility of Learning
  impossible in general, possible statistically
- Linear Regression
  analytic solution by pseudo inverse
- Logistic Regression
  minimize cross-entropy error with gradient descent
- Nonlinear Transform
  the secrete 'force' to enrich your model
- Overfitting
  the 'accident' in learning
- Principles of Learning
  simple model, matching test scenario, & no snooping