Quick Tour of Machine Learning
(機器學習速遊)

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)

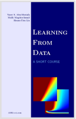
資料科學愛好者年會系列活動, 2015/12/12

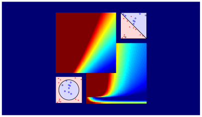
Disclaimer

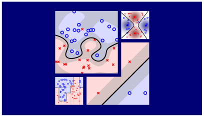
- just **super-condensed** and **shuffled** version of
  - my co-authored textbook "*Learning from Data: A Short Course*"
  - my two NTU-Coursera Mandarin-teaching ML Massive Open Online Courses
    - "*Machine Learning Foundations*": `www.coursera.org/course/ntumlone`
    - "*Machine Learning Techniques*": `www.coursera.org/course/ntumltwo`

—impossible to be complete, with most **math details removed**

- live **interaction** is important







goal: help you **begin** your journey with ML

Roadmap

Learning from Data

- What is Machine Learning
- Components of Machine Learning
- Types of Machine Learning
- Step-by-step Machine Learning

Learning from Data ::
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
$\Leftrightarrow$ improve some performance measure (e.g. prediction accuracy)

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

data $\longrightarrow$ $\boxed{\text{ML}}$ $\longrightarrow$ improved performance measure

An Application in Computational Finance

stock data $\longrightarrow$ $\boxed{\text{ML}}$ $\longrightarrow$ 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. :-)

Machine Learning and Artificial Intelligence

Machine Learning

use data to compute something that improves performance

Artificial Intelligence

compute **something that shows intelligent behavior**

- **improving performance** 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 **and popular** route to realize AI

Learning from Data ::
Components of Machine Learning

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

what to learn? (for improving performance):
'approve credit card good for bank?'

Formalize the Learning Problem

Basic Notations

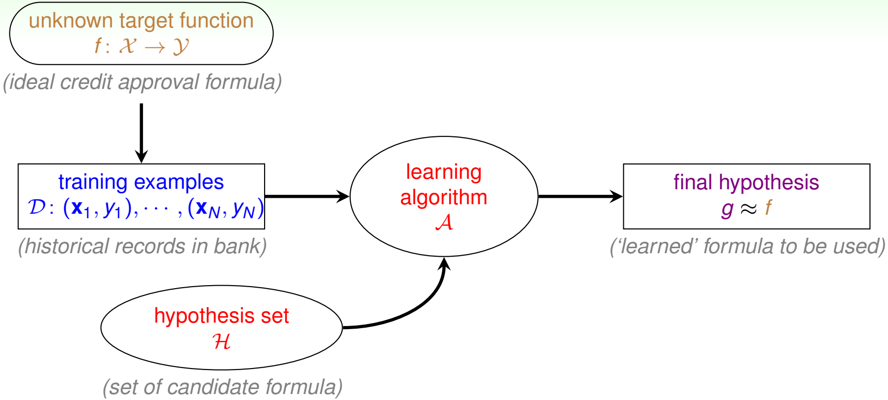
- input: $\mathbf{x} \in \mathcal{X}$ (customer application)
- output: $y \in \mathcal{Y}$ (good/bad after approving credit card)
- **unknown** underlying pattern to be learned $\Leftrightarrow$ target function: $f: \mathcal{X} \to \mathcal{Y}$ (ideal credit approval formula)
- data $\Leftrightarrow$ 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: $g: \mathcal{X} \to \mathcal{Y}$ ('learned' formula to be used), i.e. approve if
  - h_1: annual salary > NTD 800,000
  - h_2: debt > NTD 100,000 (really?)
  - h_3: year in job ≤ 2 (really?)

—all candidate formula being considered: hypothesis set $\mathcal{H}$

—procedure to learn best formula: algorithm $\mathcal{A}$

$$\{(\mathbf{x}_n, y_n)\} \text{ from } f \longrightarrow \boxed{\text{ML } (\mathcal{A}, \mathcal{H})} \to g$$

Practical Definition of Machine Learning



machine learning ($\mathcal{A}$ and $\mathcal{H}$):
use data to compute hypothesis g
that approximates target f

Key Essence of Machine Learning

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

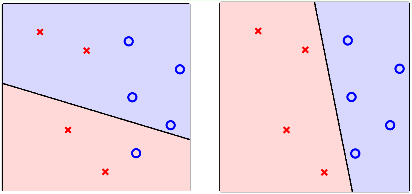
data → ML → improved performance measure

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

Learning from Data :: Types of Machine Learning

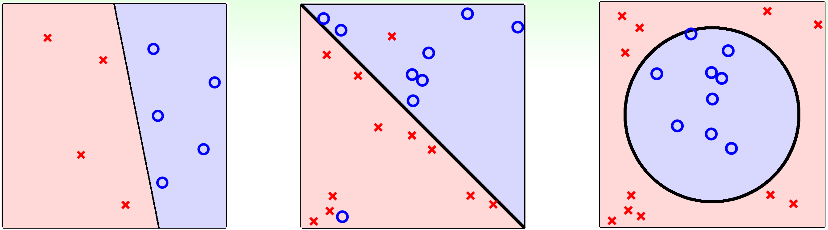
Visualizing Credit Card Problem



- customer features $\mathbf{x}$: points on the plane (or points in $\mathbb{R}^d$)
- labels y: $\circ$ (+1), $\times$ (-1)
  called **binary classification**
- hypothesis h: **lines** here, but possibly other curves
- different curve classifies customers differently

binary classification algorithm:
find **good decision boundary curve** g

More Binary Classification Problems



- credit approve/disapprove
- email spam/non-spam
- patient sick/not sick
- ad profitable/not profitable

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

Binary Classification for Education

data $\longrightarrow$ ML $\longrightarrow$ skill

- data: students' records on quizzes on a Math tutoring system
- skill: predict whether a student can give a correct answer to another quiz question

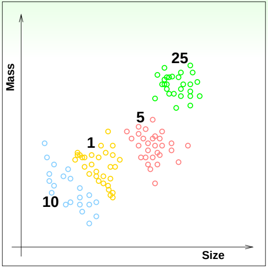
A Possible ML Solution

answer correctly $\approx$ $[\![$recent strength of student $>$ difficulty of question$]\!]$

- give ML 9 million records from 3000 students
- ML determines (**reverse-engineers**) strength and difficulty automatically

key part of the **world-champion** system from National Taiwan Univ. in KDDCup 2010

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 $\Rightarrow$ sick or not
- multiclass classification: patient features $\Rightarrow$ which type of cancer
- regression: patient features $\Rightarrow$ **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 $\Rightarrow$ stock price
- climate data $\Rightarrow$ temperature

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

Regression for Recommender System (1/2)

data $\longrightarrow$ ML $\longrightarrow$ skill

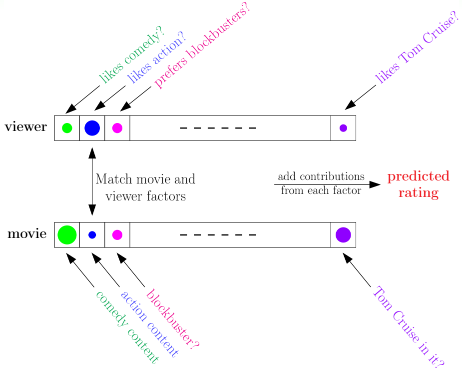
- data: how many users have rated some movies
- skill: predict how a user would rate an unrated movie

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 $\rightarrow$ 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**?

Regression for Recommender System (2/2)



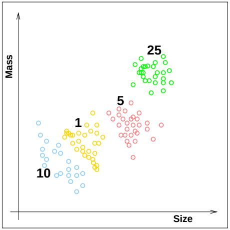
A Possible ML Solution

- 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

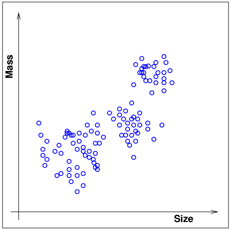
Supervised versus Unsupervised

coin recognition with y_n



supervised multiclass classification

coin recognition without y_n



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

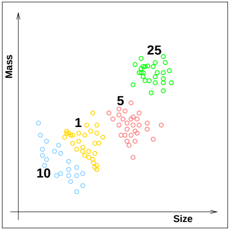
Other Clustering Problems

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

clustering: a challenging but useful problem

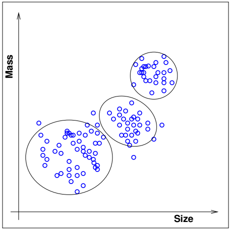
Supervised versus Unsupervised

coin recognition with y_n



supervised multiclass classification

coin recognition without y_n



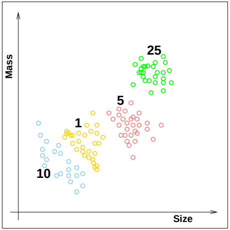
unsupervised multiclass classification
$\Longleftrightarrow$ '**clustering**'

Other Clustering Problems

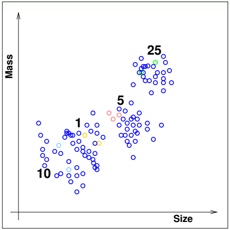
- articles $\Rightarrow$ topics
- consumer profiles $\Rightarrow$ consumer groups

clustering: a challenging but useful problem

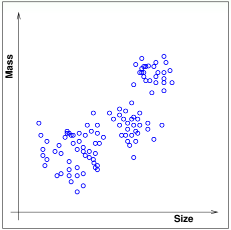
Semi-supervised: Coin Recognition with Some y_n



supervised



semi-supervised



unsupervised (clustering)

Other Semi-supervised Learning Problems

- face images with a few labeled $\Rightarrow$ face identifier (Facebook)
- medicine data with a few labeled $\Rightarrow$ medicine effect predictor

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

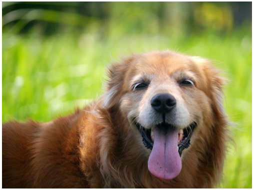
Reinforcement Learning

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)

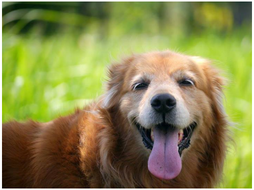
Reinforcement Learning

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



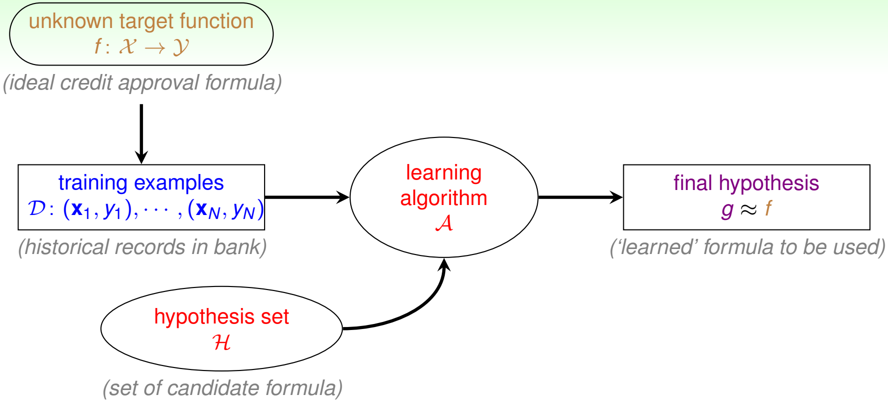
Other Reinforcement Learning Problems Using $(\mathbf{x}, \tilde{y}, \text{goodness})$

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

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

Learning from Data ::
Step-by-step Machine Learning

Step-by-step Machine Learning



1. choose error measure: how $g(\mathbf{x}) \approx f(\mathbf{x})$
2. decide hypothesis set $\mathcal{H}$
3. optimize error and more on $\mathcal{D}$ as $\mathcal{A}$
4. pray for generalization:
   whether $g(\mathbf{x}) \approx f(\mathbf{x})$ for **unseen** $\mathbf{x}$

Choose Error Measure

$g \approx f$ can often evaluate by
averaged $\mathrm{err}\,(g(\mathbf{x}), f(\mathbf{x}))$, called **pointwise error measure**

in-sample (within data)

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

out-of-sample (future data)

$$E_{\mathrm{out}}(g) = \mathop{\mathcal{E}}_{\text{future } \mathbf{x}} \mathrm{err}(g(\mathbf{x}), f(\mathbf{x}))$$

will start from 0/1 error $\mathrm{err}(\tilde{y}, y) = [\![\tilde{y} \neq y]\!]$
for **classification**

Choose Hypothesis Set (for Credit Approval)

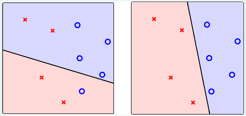
age	23 years
annual salary	NTD 1,000,000
year in job	0.5 year
current debt	200,000

- 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

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

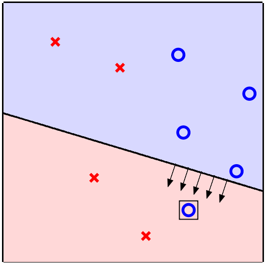


linear (binary) classifier, called 'perceptron' historically

Optimize Error (and More) on Data

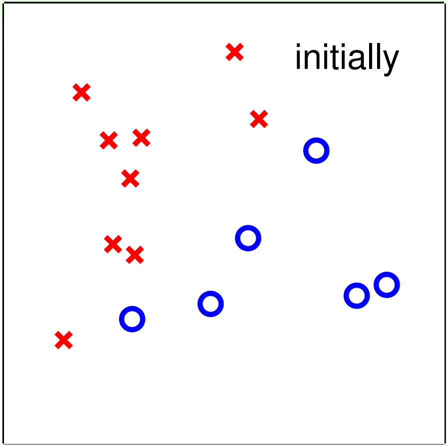
$\mathcal{H}$ = all possible perceptrons, $g = ?$

- 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 $\mathcal{D}$



let's visualize **without math**

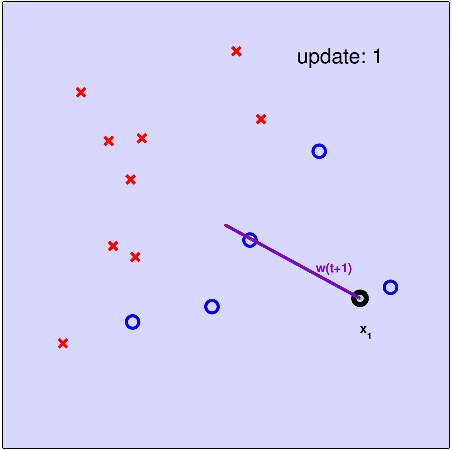
Seeing is Believing



worked like a charm with < 20 lines!!

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

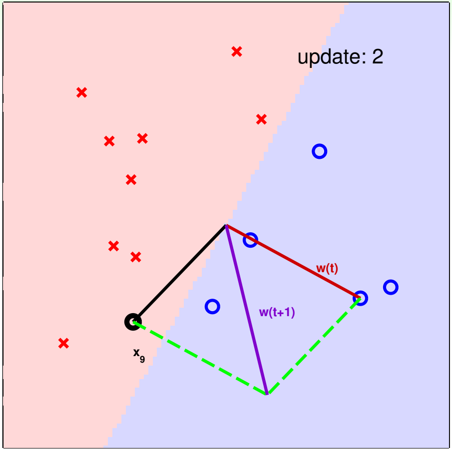
Seeing is Believing



worked like a charm with < 20 lines!!

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

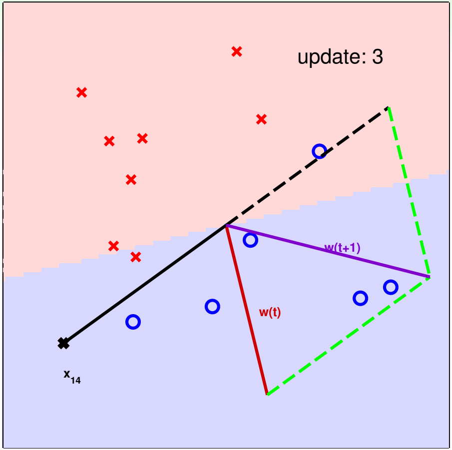
Seeing is Believing



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

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

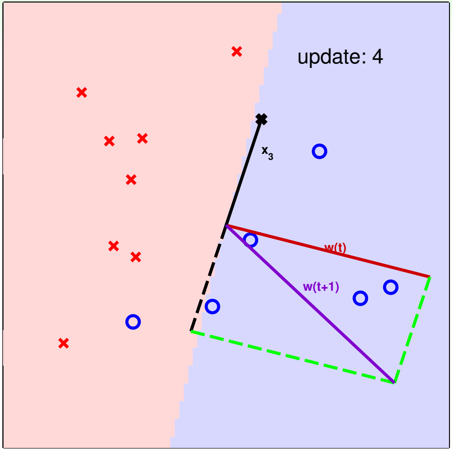
Seeing is Believing



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

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

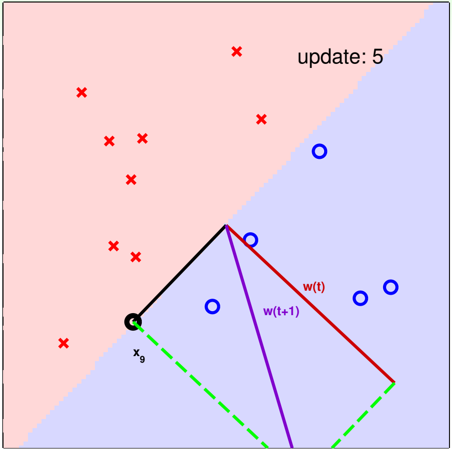
Seeing is Believing



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

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

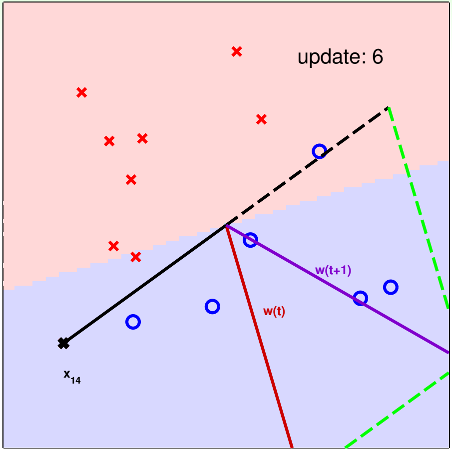
Seeing is Believing



worked like a charm with <20 lines!!

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

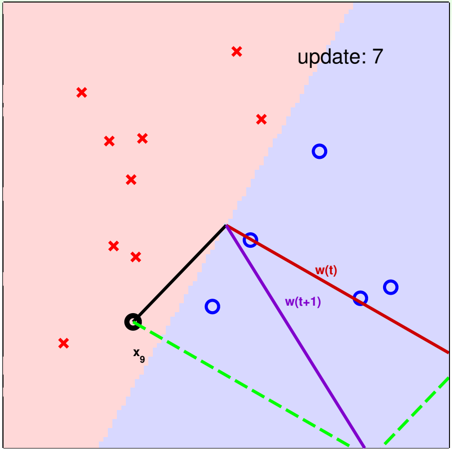
Seeing is Believing



worked like a charm with <20 lines!!

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

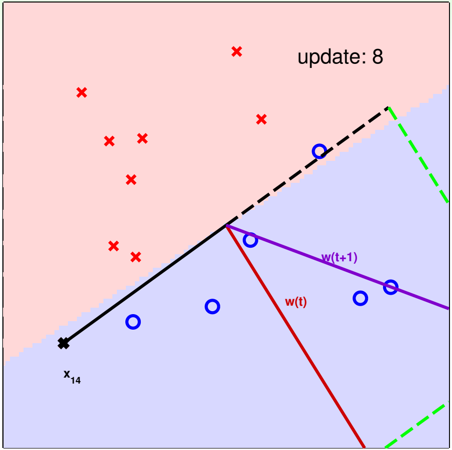
Seeing is Believing



worked like a charm with <20 lines!!

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

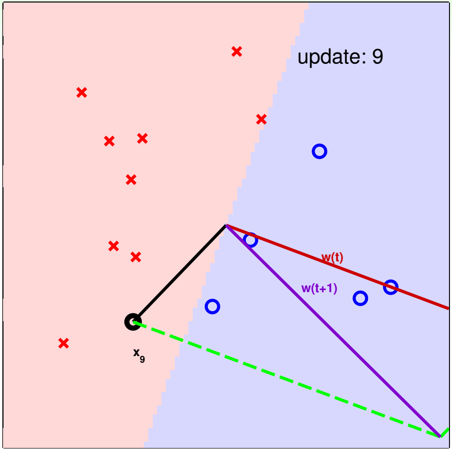
Seeing is Believing



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

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

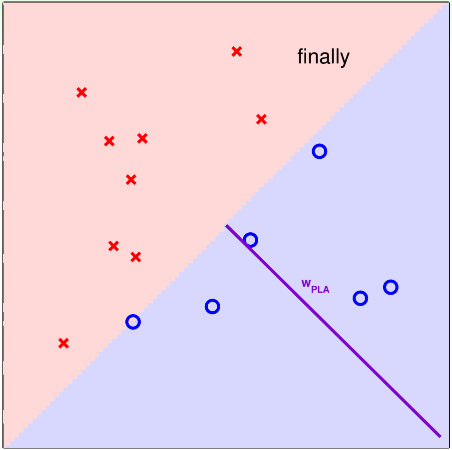
Seeing is Believing



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

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

Seeing is Believing



worked like a charm with <20 lines!!

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

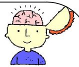
Pray for Generalization

(pictures from Google Image Search)

Parent

(picture, label) pairs

Kid's brain



good hypothesis

alternatives

Target $f(\mathbf{x})$ + noise

examples (picture $\mathbf{x}_n$, label y_n)

learning algorithm

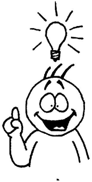
good hypothesis $g(\mathbf{x}) \approx f(\mathbf{x})$

hypothesis set $\mathcal{H}$

challenge:
see only $\{(\mathbf{x}_n, y_n)\}$ without knowing f nor noise
$\overset{?}{\Longrightarrow}$ **generalize** to unseen $(\mathbf{x}, y)$ w.r.t. $f(\mathbf{x})$

Generalization Is Non-trivial

Bob impresses Alice by memorizing every given (movie, rank);
but too nervous about a **new movie** and guesses randomly



(pictures from Google Image Search)

memorize	$\neq$	**generalize**
perfect from Bob's view	$\neq$	**good for Alice**
perfect during training	$\neq$	**good when testing**

take-home message: if $\mathcal{H}$ is **simple** (like lines),
generalization is **usually possible**

Mini-Summary

Learning from Data

- What is Machine Learning

  use data to approximate target
- Components of Machine Learning

  algorithm $\mathcal{A}$ takes data $\mathcal{D}$ and hypotheses $\mathcal{H}$ to get hypothesis g
- Types of Machine Learning

  variety of problems almost everywhere
- Step-by-step Machine Learning

  error, hypotheses, optimize, generalize

Roadmap

Fundamental Machine Learning Models

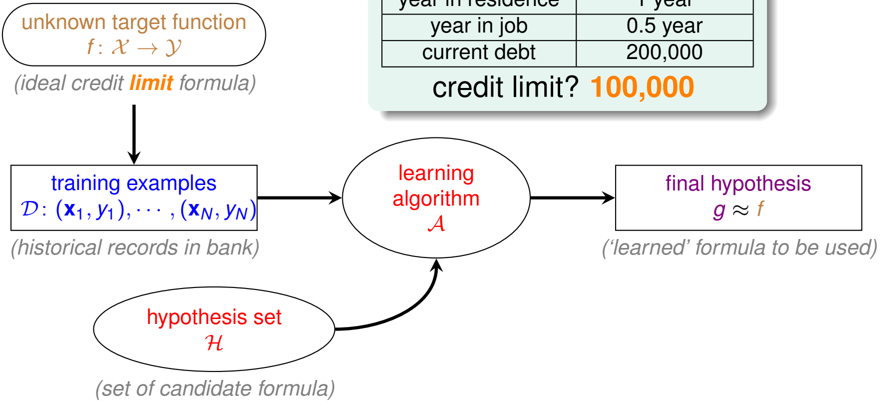
- Linear Regression
- Logistic Regression
- Nonlinear Transform
- Decision Tree

Fundamental Machine Learning Models :: 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**



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

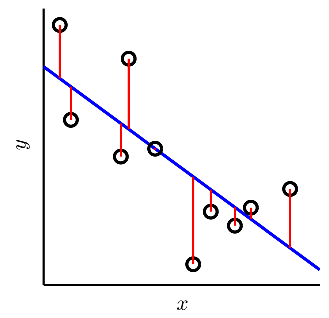
$$y \approx \sum_{i=0}^{d} w_i x_i$$

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

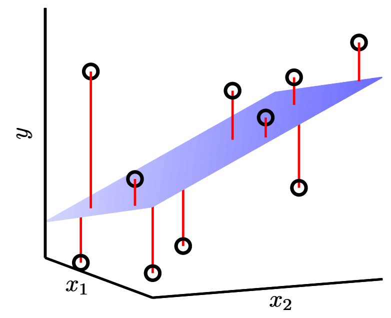
$h(\mathbf{x})$: like **perceptron**, but without the sign

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 $\mathrm{err}(\hat{y}, y) = (\hat{y} - y)^2$

in-sample

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

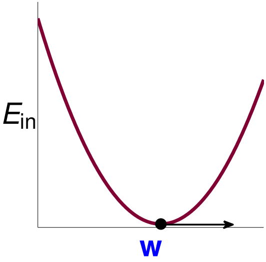
out-of-sample

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

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

Minimize E_{in}

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



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

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

—not possible to 'roll down'

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

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, 'simple' like perceptrons**
- improving iteratively? **somewhat, within an iterative pseudo-inverse routine**

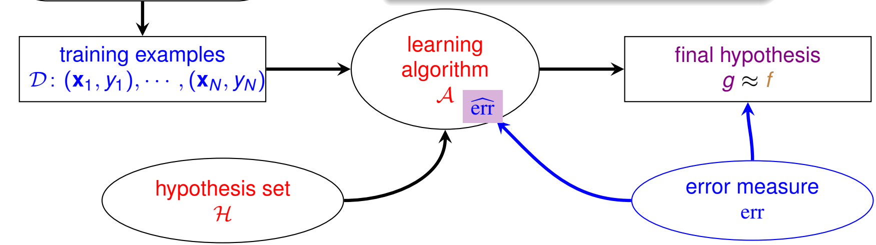
if $E_{\text{out}}(\mathbf{w}_{\text{LIN}})$ is good, **learning 'happened'**!

Fundamental Machine Learning Models :: 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:

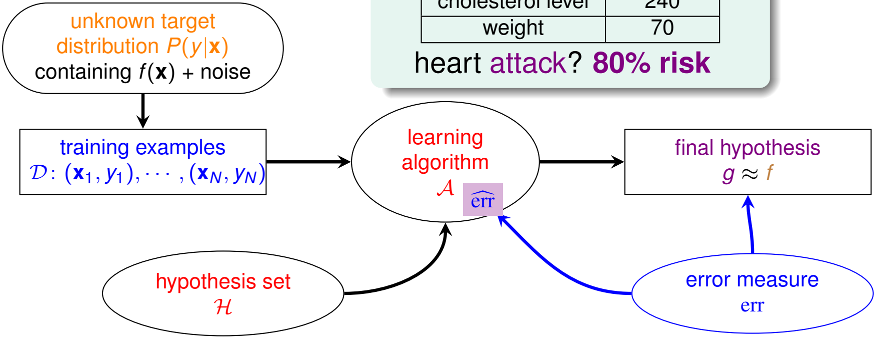
$$\text{ideal } f(\mathbf{x}) = \text{sign}\left(P(+1|\mathbf{x}) - \tfrac{1}{2}\right) \in \{-1, +1\}$$

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

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

actual (noisy) data

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

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

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

actual (noisy) data

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

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

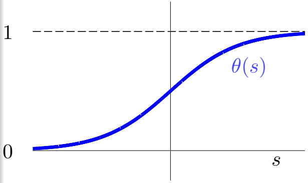
Logistic Hypothesis

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':

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

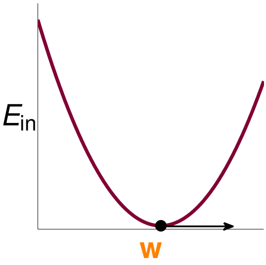
- convert the score to estimated probability by logistic function $\theta(s)$



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

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

a popular error: $E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$ called **cross-entropy** derived from **maximum likelihood**



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

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

most basic algorithm:
gradient descent (roll downhill)

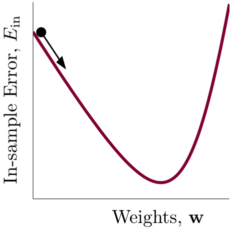
Gradient Descent

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



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

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

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

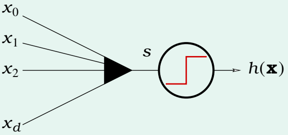
can use more sophisticated tools to speed up

Linear Models Summarized

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

linear classification

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

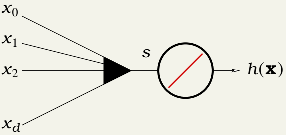


plausible err = 0/1

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

linear regression

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

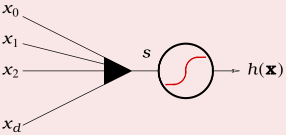


friendly err = squared

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

logistic regression

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



plausible err = cross-entropy

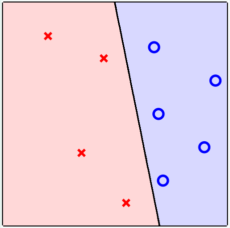
smooth convex $E_{\text{in}}(\mathbf{w})$:
gradient descent

my ‘secret’: **linear first!!**

Fundamental Machine Learning Models :: Nonlinear Transform

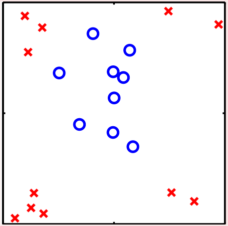
Linear Hypotheses

up to now: linear hypotheses



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

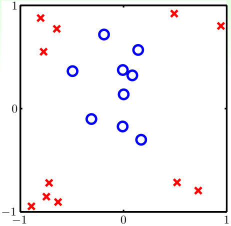
but limited . . .

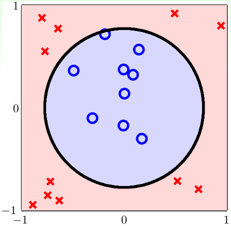


- theoretically: **complexity under control :-)**
- practically: on some $\mathcal{D}$, **large E_{in}** for every line **:-(**

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

Circular Separable





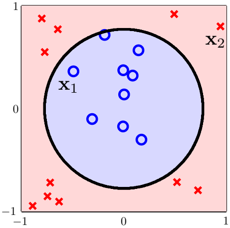
- $\mathcal{D}$ not linear separable
- but **circular separable** by a circle of radius $\sqrt{0.6}$ centered at origin:

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

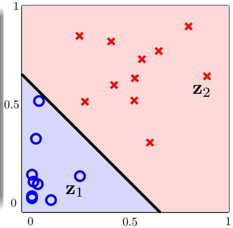
re-derive **Circular**-PLA, **Circular**-Regression, blahblah . . . all over again? **:-)**

Circular Separable and Linear Separable

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



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



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

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, ...

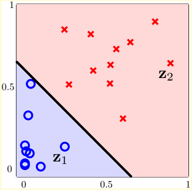
  ellipse $2(x_1 + x_2 - 3)^2 + (x_1 - x_2 - 4)^2 = 1$

  $\Longleftarrow \tilde{\mathbf{w}}^T = [33, -20, -4, 3, 2, 3]$

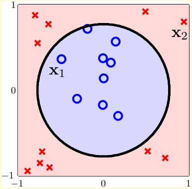
include **lines and constants as degenerate cases**

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**



$\Longleftrightarrow$

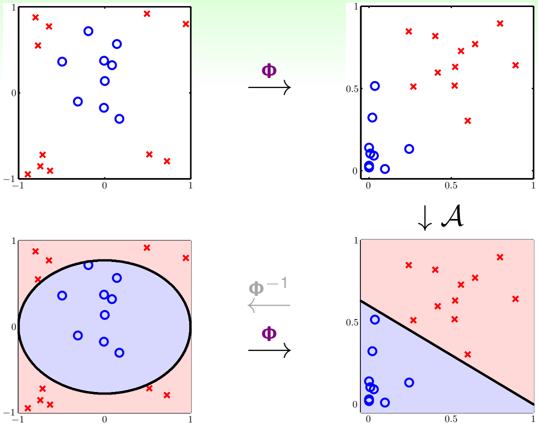


- want: get **good perceptron** in $\mathcal{Z}$-space
- known: get **good perceptron** in $\mathcal{X}$-space with data $\{(\mathbf{x}_n, y_n)\}$

solution: get **good perceptron** in $\mathcal{Z}$-space with data

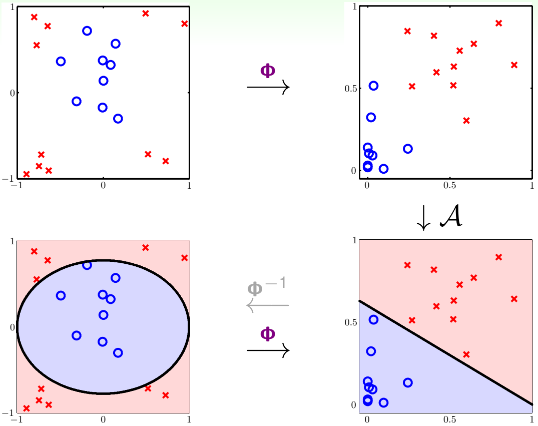
$$\{(\mathbf{z}_n = \mathbf{\Phi}_2(\mathbf{x}_n), y_n)\}$$

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 $\mathbf{\Phi}$ + Linear Models



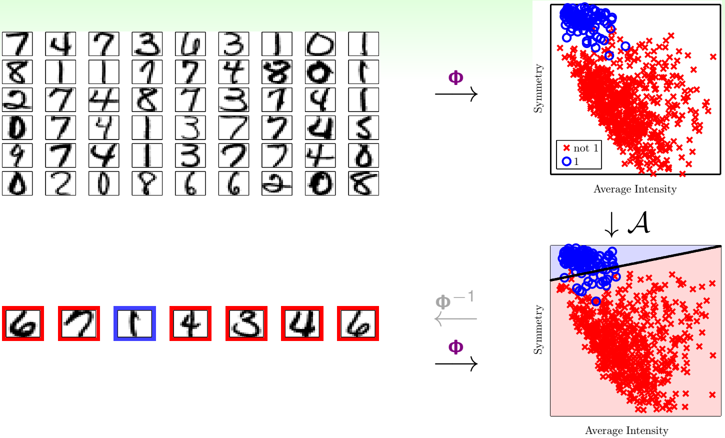
two choices:

- feature transform $\mathbf{\Phi}$
- 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



more generally, 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: $\Phi_Q(\mathbf{x}) = \Big(1, x_1, x_2, \ldots, x_d, x_1^2, x_1x_2, \ldots, x_d^2, \ldots, x_1^Q, x_1^{Q-1}x_2, \ldots, x_d^Q \Big)$

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

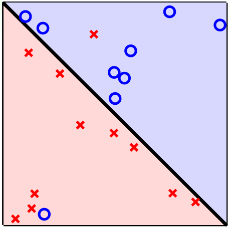
= # ways of $\leq Q$-combination from d kinds with repetitions

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

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

Q large $\Longrightarrow$ **difficult to compute/store**
AND curve too complicated

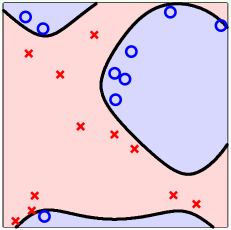
Generalization Issue



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

which one do you prefer? :-)

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



$\mathbf{\Phi}_4$

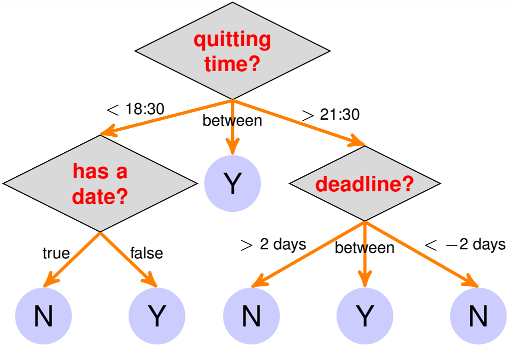
how to pick Q?
model selection (to be discussed) important

Fundamental Machine Learning Models :: Decision Tree

Decision Tree for Watching MOOC Lectures

$$G(\mathbf{x}) = \sum_{t=1}^{T} q_t(\mathbf{x}) \cdot g_t(\mathbf{x})$$

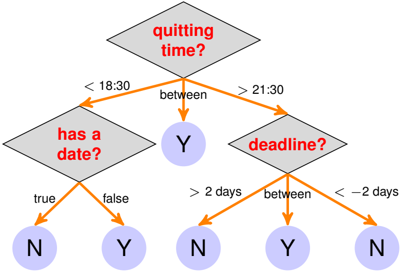
- **base hypothesis** $g_t(\mathbf{x})$: leaf at end of path t, a **constant** here
- **condition** $q_t(\mathbf{x})$: $[\![$is $\mathbf{x}$ on path t?$]\!]$
- usually with **simple internal nodes**



decision tree: arguably one of the most **human-mimicking models**

Recursive View of Decision Tree

Path View: $G(\mathbf{x}) = \sum_{t=1}^{T} [\![\mathbf{x} \text{ on path } t]\!] \cdot \text{leaf}_t(\mathbf{x})$



Recursive View

$$G(\mathbf{x}) = \sum_{c=1}^{C} [\![b(\mathbf{x}) = c]\!] \cdot G_c(\mathbf{x})$$

- $G(\mathbf{x})$: full-tree hypothesis
- $b(\mathbf{x})$: branching criteria
- $G_c(\mathbf{x})$: sub-tree hypothesis at the c-th branch

tree = (root, sub-trees), just like what
your data structure instructor would say :-)

A Basic Decision Tree Algorithm

$$G(\mathbf{x}) = \sum_{c=1}^{C} [\![b(\mathbf{x}) = c]\!] \, G_c(\mathbf{x})$$

function DecisionTree$\left(\text{data } \mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}\right)$
if termination criteria met
    return base hypothesis $g_t(\mathbf{x})$
else
1. learn branching criteria $b(\mathbf{x})$
2. split $\mathcal{D}$ to C parts $\mathcal{D}_c = \{(\mathbf{x}_n, y_n) : b(\mathbf{x}_n) = c\}$
3. build sub-tree $G_c \leftarrow$ DecisionTree$(\mathcal{D}_c)$
4. return $G(\mathbf{x}) = \sum_{c=1}^{C} [\![b(\mathbf{x}) = c]\!] \, G_c(\mathbf{x})$

four choices: number of branches, branching criteria, termination criteria, & base hypothesis

Classification and Regression Tree (C&RT)

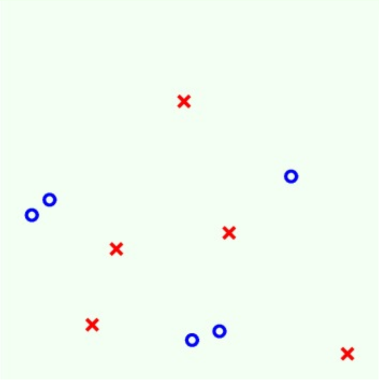
function DecisionTree(data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$)
if termination criteria met
    return base hypothesis $g_t(\mathbf{x})$
else ...
    2 split $\mathcal{D}$ to C parts $\mathcal{D}_c = \{(\mathbf{x}_n, y_n) : b(\mathbf{x}_n) = c\}$

choices

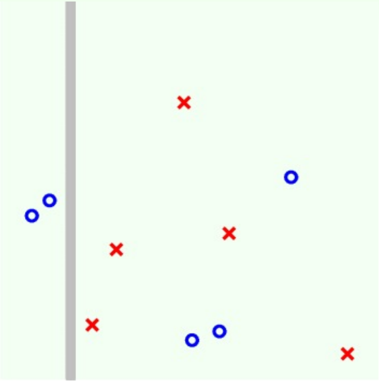
- $C = 2$ (binary tree)
- $g_t(\mathbf{x}) = E_{\text{in}}$-optimal constant
  - binary/multiclass classification (0/1 error): majority of $\{y_n\}$
  - regression (squared error): average of $\{y_n\}$
- branching: threshold some selected dimension
- termination: fully-grown, or better **pruned**

disclaimer:
C&RT here is based on **selected components** of **CART^{TM} of California Statistical Software**

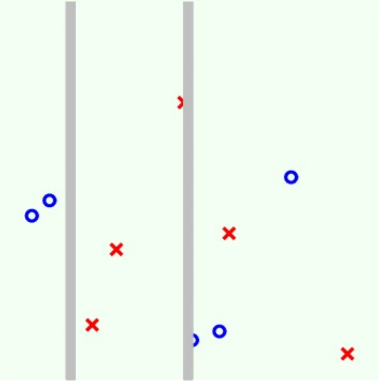
A Simple Data Set



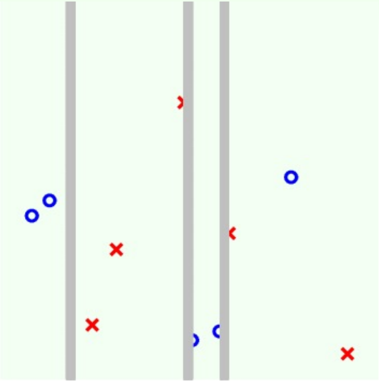
A Simple Data Set



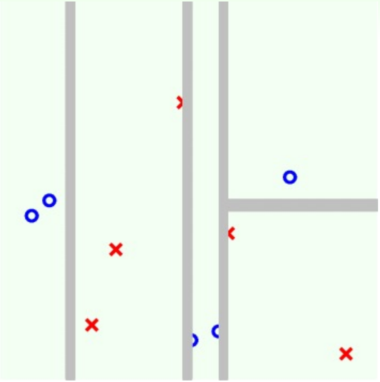
A Simple Data Set



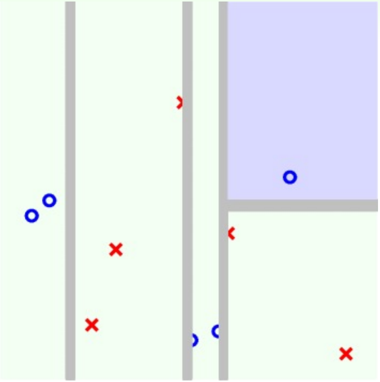
A Simple Data Set



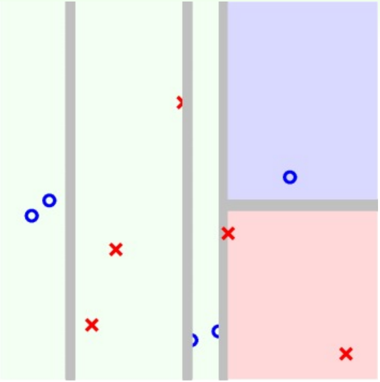
A Simple Data Set



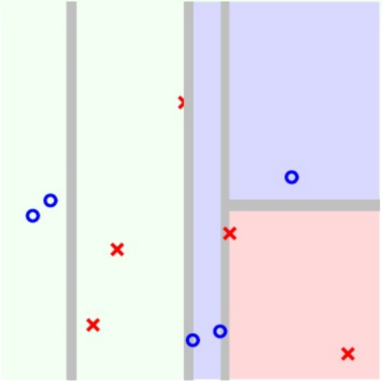
A Simple Data Set



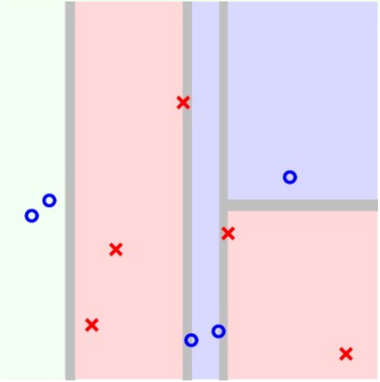
A Simple Data Set



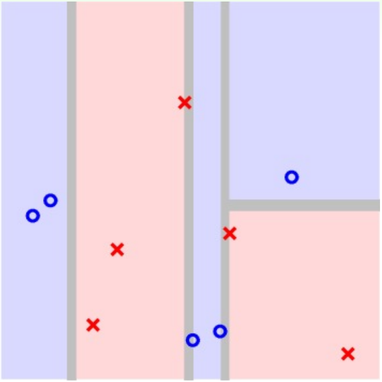
A Simple Data Set



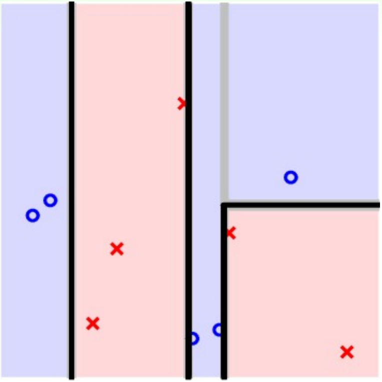
A Simple Data Set



A Simple Data Set



A Simple Data Set



C&RT

C&RT: 'divide-and-conquer'

Practical Specialties of C&RT

- human-explainable
- multiclass easily
- categorical features easily
- missing features easily
- efficient non-linear training (and testing)

—almost no other learning model share all such specialties, except for **other decision trees**

another popular decision tree algorithm: **C4.5**, with different **choices of heuristics**

Mini-Summary

Fundamental Machine Learning Models

- 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
- Decision Tree
  human-like explainable model learned recursively

Roadmap

Hazard of Overfitting

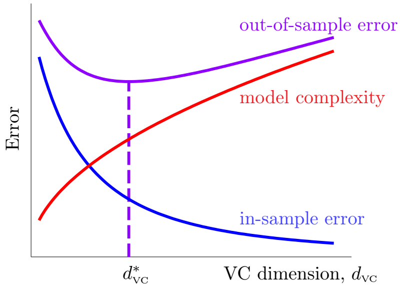
- Overfitting
- Data Manipulation and Regularization
- Validation
- Principles of Learning

Hazard of Overfitting :: Overfitting

Theoretical Foundation of Statistical Learning

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

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

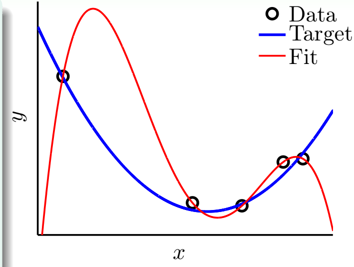


- $d_{\text{VC}}(\mathcal{H})$: complexity of $\mathcal{H}$, $\approx$ # of parameters to describe $\mathcal{H}$
- $d_{\text{VC}} \uparrow$: $E_{\text{in}} \downarrow$ but $\Omega \uparrow$
- $d_{\text{VC}} \downarrow$: $\Omega \downarrow$ but $E_{\text{in}} \uparrow$
- best d^*_{VC} in the middle

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

Bad Generalization

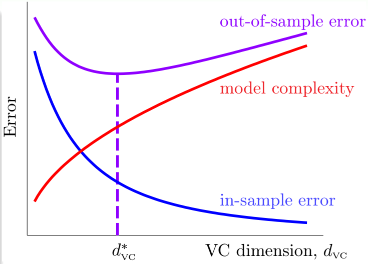
- regression for $x \in \mathbb{R}$ with $N = 5$ examples
- target $f(x)$ = 2nd order polynomial
- label $y_n = f(x_n) + \text{very small noise}$
- linear regression in $\mathcal{Z}$-space + $\mathbf{\Phi}$ = 4th order polynomial
- unique solution passing all examples $\Longrightarrow E_{\text{in}}(g)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



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

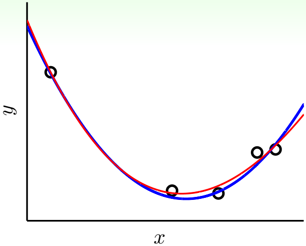
Bad Generalization and Overfitting

- 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** —$E_{\text{in}} \uparrow$, $E_{\text{out}} \uparrow$

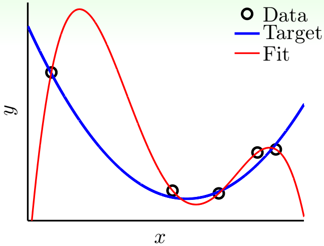


bad generalization: low E_{in}, high E_{out};
overfitting: low**er** E_{in}, high**er** E_{out}

Cause of Overfitting: A Driving Analogy



'good fit' ⟹ **overfit**

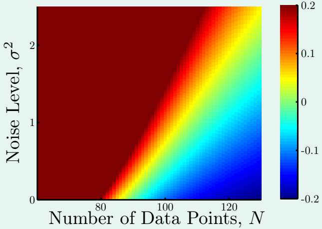


learning	driving
overfit	commit a car accident
use excessive d_{VC}	'drive too fast'
noise	bumpy road
limited data size N	limited observations about road condition

let's 'visualize' overfitting

Impact of Noise and Data Size

impact of σ^2 versus N:
stochastic noise

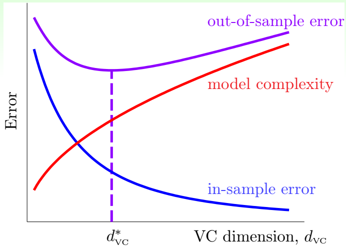


reasons of serious overfitting:

data size $N \downarrow$	overfit $\uparrow$
stochastic noise $\uparrow$	overfit $\uparrow$

overfitting 'easily' happens
(more on ML Foundations, Lecture 13)

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!**

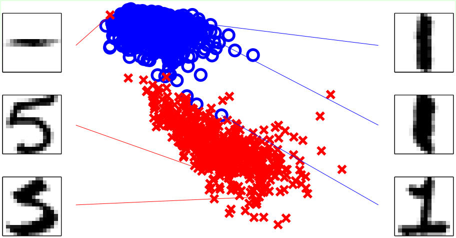
Driving Analogy Revisited

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
data cleaning/pruning	use more accurate road information
data hinting	exploit more road information
regularization	put the brakes
validation	monitor the dashboard

all very **practical** techniques
to combat overfitting

Hazard of Overfitting :: Data Manipulation and Regularization

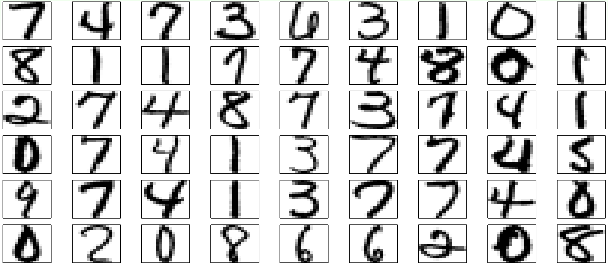
Data Cleaning/Pruning



- if 'detect' the outlier 5 at the top by
  - too close to other ○, or too far from other ×
  - wrong by current classifier
  - . . .
- possible action 1: correct the label (**data cleaning**)
- possible action 2: remove the example (**data pruning**)

possibly helps, but **effect varies**

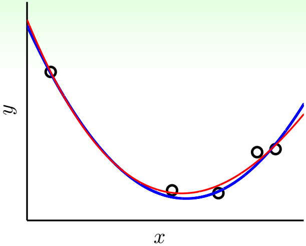
Data Hinting



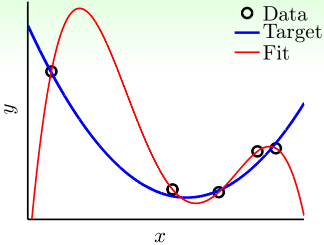
- slightly shifted/rotated digits carry the same meaning
- possible action: add **virtual examples** by shifting/rotating the given digits (**data hinting**)

possibly helps, but **watch out** not to **steal the thunder**

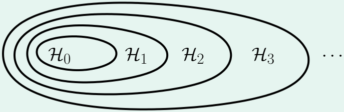
Regularization: The Magic



‘regularized fit’ $\Longleftarrow$ overfit



- idea: ‘step back’ from 10-th order polynomials to 2-nd order ones



- name history: function approximation for **ill-posed problems**

how to step back?

Step Back by Minimizing the Augmented Error

Augmented Error

$$E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$$

VC Bound

$$E_{\text{out}}(\mathbf{w}) \le E_{\text{in}}(\mathbf{w}) + \Omega(\mathcal{H})$$

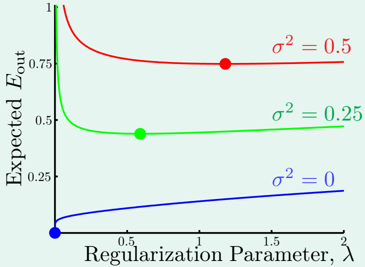
- regularizer $\mathbf{w}^T\mathbf{w}$: complexity of a single hypothesis
- generalization price $\Omega(\mathcal{H})$: complexity of a hypothesis set
- if $\frac{\lambda}{N}\Omega(\mathbf{w})$ 'represents' $\Omega(\mathcal{H})$ well,
  E_{aug} is a better proxy of E_{out} than E_{in}

minimizing E_{aug}:

(heuristically) operating with the better proxy;
(technically) enjoying flexibility of whole $\mathcal{H}$

The Optimal λ

stochastic noise

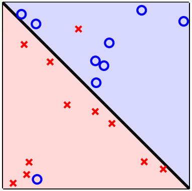


- more noise $\Longleftrightarrow$ more regularization needed
  —more bumpy road $\Longleftrightarrow$ putting brakes more
- noise **unknown**—important to **make proper choices**

how to choose?
validation!

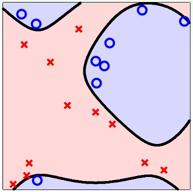
Hazard of Overfitting :: Validation

Model Selection Problem



$\mathcal{H}_1$

which one do you prefer? :-)



$\mathcal{H}_2$

- given: M models $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$, each with corresponding algorithm $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_M$
- goal: select $\mathcal{H}_{m^*}$ such that $g_{m^*} = \mathcal{A}_{m^*}(\mathcal{D})$ is of low $E_{\text{out}}(g_{m^*})$
- unknown E_{out}, as always **:-)**
- arguably the **most important** practical problem of ML

how to select? **visually?**
—no, can you really visualize $\mathbb{R}^{1126}$? :-)

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}:
  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

$$E_{\text{out}}(g_m^-) \leq E_{\text{val}}(g_m^-) + \text{'small'}$$

Model Selection by Best E_{val}

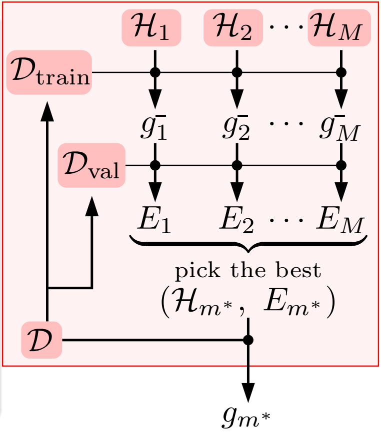
$$m^* = \operatorname*{argmin}_{1 \le m \le M} \left(E_m = E_{\text{val}}(\mathcal{A}_m(\mathcal{D}_{\text{train}}))\right)$$

- generalization guarantee for all m:

$$E_{\text{out}}(g_m^-) \le E_{\text{val}}(g_m^-) + \text{`small'}$$

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

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



$$E_{\text{out}}(g_{m^*}) \le E_{\text{out}}(g_{m^*}^-) \le E_{\text{val}}(g_{m^*}^-) + \text{`small'}$$

V-fold Cross Validation

making validation more stable

- V-fold cross-validation: random-partition of $\mathcal{D}$ **to V equal parts**,

$$\overbrace{\mathcal{D}_1 \mid \mathcal{D}_2 \mid \mathcal{D}_3 \mid \mathcal{D}_4 \mid \mathcal{D}_5 \mid \mathcal{D}_6 \mid \mathcal{D}_7 \mid \mathcal{D}_8 \mid \mathcal{D}_9 \mid \mathcal{D}_{10}}^{\mathcal{D}}$$

train — validate — train

take $V-1$ for training and 1 for validation orderly

$$E_{\text{cv}}(\mathcal{H}, \mathcal{A}) = \frac{1}{V} \sum_{v=1}^{V} E_{\text{val}}^{(v)}(g_v^-)$$

- selection by E_{cv}: $m^* = \underset{1 \le m \le M}{\text{argmin}}(E_m = E_{\text{cv}}(\mathcal{H}_m, \mathcal{A}_m))$

practical rule of thumb: $V = 10$

Final Words on Validation

'Selecting' Validation Tool

- ***V*-Fold** generally preferred over single validation if computation allows
- **5-Fold or 10-Fold** generally works well

Nature of Validation

- all training models: select among hypotheses
- all validation schemes: select among finalists
- all testing methods: just evaluate

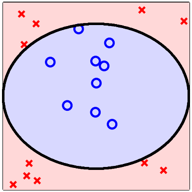
validation still more optimistic than testing

do not fool yourself and others **:-)**,
report test result, not best validation result

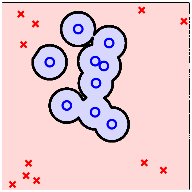
Hazard of Overfitting :: Principles of Learning

Occam's Razor for Learning

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



which one do you prefer? :-)



My KISS Principle:

Keep It Simple, ~~Stupid~~ ***Safe***

Sampling Bias

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

philosophical explanation:
study Math hard but test English: **no strong test guarantee**

A True Personal Story

- Netflix competition for movie recommender system:
  10% **improvement = 1M US dollars**
- on my own validation data, first shot, showed 13% improvement
- **why am I still teaching in NTU? :-)**
  validation: random examples within data;
  test: “last” user records “after” data

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

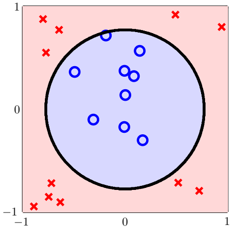
Visual Data Snooping

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 $\mathbf{z} = \left(\text{sign}(0.6 - x_1^2 - x_2^2)\right)$?

—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)

Mini-Summary

Hazard of Overfitting

- Overfitting

  the 'accident' that is everywhere in learning
- Data Manipulation and Regularization

  clean data, synthetic data, or augmented error
- Validation

  honestly simulate testing procedure for proper model selection
- Principles of Learning

  simple model, matching test scenario, and no snooping

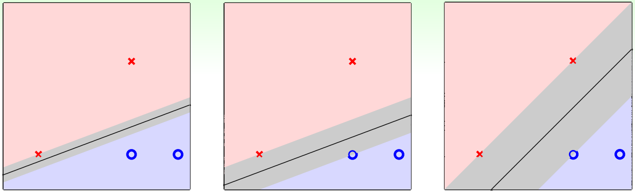
Roadmap

Modern Machine Learning Models

- Support Vector Machine
- Random Forest
- Adaptive (or Gradient) Boosting
- Deep Learning

Modern Machine Learning Models :: Support Vector Machine

Motivation: Large-Margin Separating Hyperplane

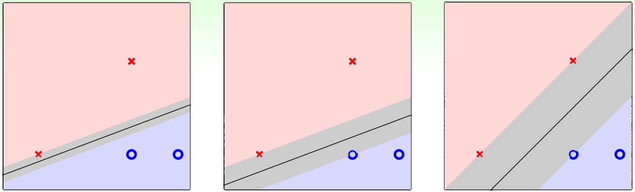


$$\max_{\mathbf{w}} \quad \text{fatness}(\mathbf{w})$$
$$\text{subject to} \quad \mathbf{w} \text{ classifies every } (\mathbf{x}_n, y_n) \text{ correctly}$$
$$\text{fatness}(\mathbf{w}) = \min_{n=1,\ldots,N} \text{distance}(\mathbf{x}_n, \mathbf{w})$$

- fatness: formally called **margin**
- correctness: $y_n = \text{sign}(\mathbf{w}^T \mathbf{x}_n)$

initial goal: find **largest-margin separating** hyperplane

Motivation: Large-Margin Separating Hyperplane



$$\max_{\mathbf{w}} \quad \text{margin}(\mathbf{w})$$

$$\text{subject to} \quad \text{every } y_n \mathbf{w}^T \mathbf{x}_n > 0$$

$$\text{margin}(\mathbf{w}) = \min_{n=1,\ldots,N} \text{distance}(\mathbf{x}_n, \mathbf{w})$$

- fatness: formally called **margin**
- correctness: $y_n = \text{sign}(\mathbf{w}^T \mathbf{x}_n)$

initial goal: find **largest-margin** **separating** hyperplane

Soft-Margin Support Vector Machine

initial goal: find **largest-margin** **separating** hyperplane

- soft-margin (**practical**) SVM: not insisting on **separating**:
  - minimize large-margin regularizer + $C\cdot$ separation error,
  - just like regularization with augmented error

$$\min E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$$

- two forms:
  - finding hyperplane in original space (**linear first!!**)
    LIBLINEAR `www.csie.ntu.edu.tw/~cjlin/liblinear`
  - or in **mysterious transformed space** hidden in '**kernels**'
    LIBSVM `www.csie.ntu.edu.tw/~cjlin/libsvm`

linear: 'best' linear classification model;
non-linear: 'leading' non-linear classification model **for mid-sized data**

Hypothesis of Gaussian SVM

Gaussian kernel $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2\right)$

$$
\begin{aligned}
g_{\text{SVM}}(\mathbf{x}) &= \text{sign}\left(\sum_{\text{SV}} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b\right) \\
&= \text{sign}\left(\sum_{\text{SV}} \alpha_n y_n \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}_n\|^2\right) + b\right)
\end{aligned}
$$

- linear combination of Gaussians centered at SVs $\mathbf{x}_n$
- also called Radial Basis Function (RBF) kernel

Gaussian SVM:
find α_n to combine Gaussians centered at $\mathbf{x}_n$
& achieve large margin in infinite-dim. space

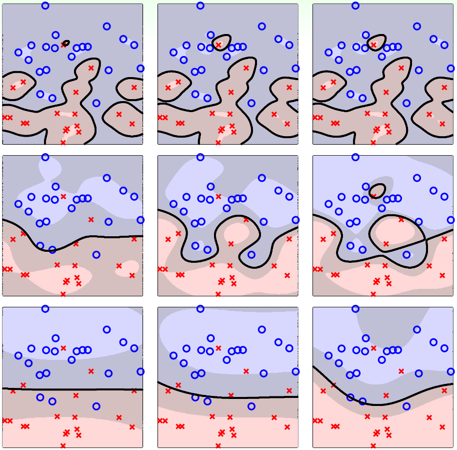
Support Vector Mechanism

	large-margin **hyperplanes** **+ higher-order transforms** with **kernel trick** **+ noise tolerance** of **soft-margin**
#	**not many**
boundary	**sophisticated**

- transformed vector $\mathbf{z} = \mathbf{\Phi}(\mathbf{x}) \Longrightarrow$ efficient kernel $K(\mathbf{x}, \mathbf{x}')$
- store optimal $\mathbf{w} \Longrightarrow$ store a few SVs and α_n

new possibility by Gaussian SVM:
infinite-dimensional linear classification, with generalization 'guarded by' large-margin **:-)**

Practical Need: Model Selection



- large $\gamma \Longrightarrow$ sharp Gaussians $\Longrightarrow$ 'overfit'?
- complicated even for (C, γ) **of Gaussian SVM**
- more combinations if including other kernels or parameters

how to select? **validation :-)**

Step-by-step Use of SVM

strongly recommended: 'A Practical Guide to Support Vector Classification'

`http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`

1. scale each feature of your data to a suitable range (say, $[-1, 1]$)
2. use a Gaussian RBF kernel
3. use cross validation and grid search to choose good (γ, C)
4. use the best (γ, C) on your data
5. do testing with the learned SVM classifier

all included in `easy.py` of LIBSVM

Modern Machine Learning Models :: Random Forest

Random Forest (RF)

random forest (RF) = bagging (random sampling) + fully-grown C&RT decision tree

function RandomForest($\mathcal{D}$)
For $t = 1, 2, \ldots, T$

1. request size-N' data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain tree g_t by DTree($\tilde{\mathcal{D}}_t$)

return $G = \text{Uniform}(\{g_t\})$

function DTree($\mathcal{D}$)
if termination return base g_t
else

1. learn $b(\mathbf{x})$ and split $\mathcal{D}$ to $\mathcal{D}_c$ by $b(\mathbf{x})$
2. build $G_c \leftarrow$ DTree($\mathcal{D}_c$)
3. return $G(\mathbf{x}) = \sum_{c=1}^{C} [\![b(\mathbf{x}) = c]\!]\, G_c(\mathbf{x})$

- highly **parallel**/**efficient** to learn
- **inherit pros** of C&RT
- **eliminate cons** of fully-grown tree

Feature Selection

for $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, want to remove

- **redundant** features: like keeping one of 'age' and 'full birthday'
- **irrelevant** features: like insurance type for cancer prediction

and only 'learn' **subset-transform** $\mathbf{\Phi}(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_{d'}})$ with $d' < d$ for $g(\mathbf{\Phi}(\mathbf{x}))$

advantages:

- **efficiency**: simpler hypothesis and shorter prediction time
- **generalization**: 'feature noise' removed
- **interpretability**

disadvantages:

- **computation**: 'combinatorial' optimization in training
- **overfit**: 'combinatorial' selection
- **mis-interpretability**

decision tree: a rare model
with **built-in feature selection**

Feature Selection by Importance

idea: if possible to calculate

$$\text{importance}(i) \text{ for } i = 1, 2, \ldots, d$$

then can select $i_1, i_2, \ldots, i_{d'}$ of top-d' importance

importance by linear model

$$\text{score} = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^{d} w_i x_i$$

- intuitive estimate: $\text{importance}(i) = |w_i|$ with some 'good' $\mathbf{w}$
- getting 'good' $\mathbf{w}$: learned from data
- non-linear models? often **much harder**

but **'easy'** feature selection in RF

Feature Importance by Permutation Test

idea: random test
—if feature i needed, 'random' values of $x_{n,i}$ degrades performance

permutation test:

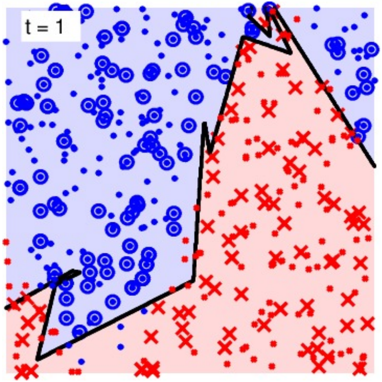
$$\text{importance}(i) = \text{performance}(\mathcal{D}) - \text{performance}(\mathcal{D}^{(p)})$$

with $\mathcal{D}^{(p)}$ is $\mathcal{D}$ with $\{x_{n,i}\}$ replaced by permuted $\{x_{n,i}\}_{n=1}^{N}$

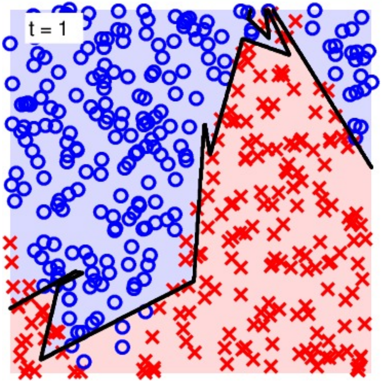
permutation test: a general statistical tool that can be easily coupled with RF

A Complicated Data Set

g_t $(N' = N/2)$

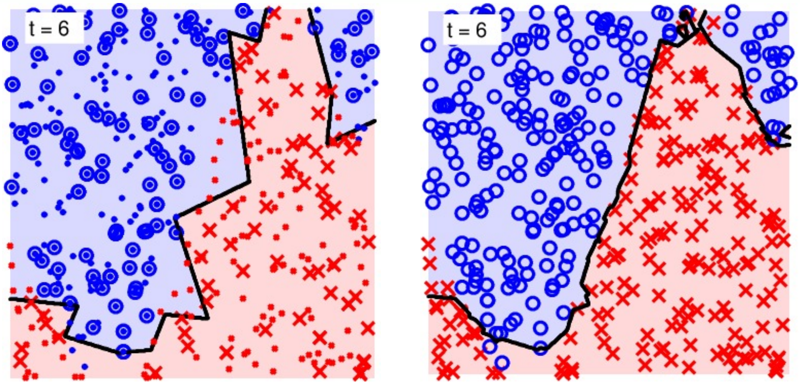


G with first t trees



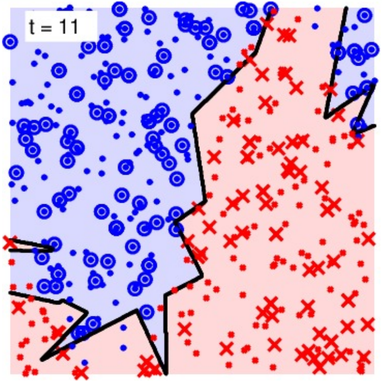
A Complicated Data Set

g_t $(N' = N/2)$ G with first t trees

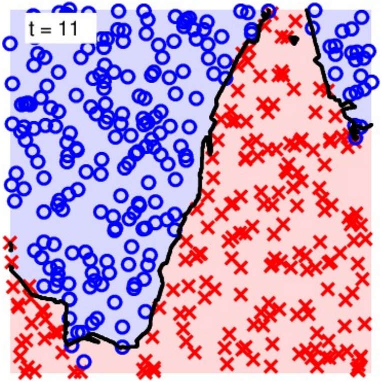


A Complicated Data Set

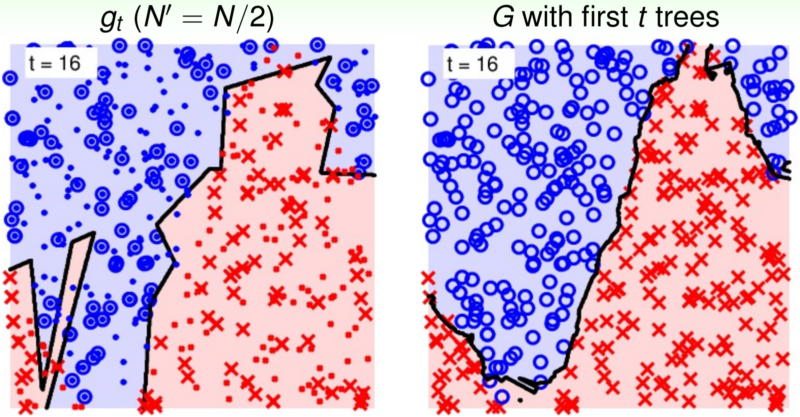
g_t $(N' = N/2)$



G with first t trees

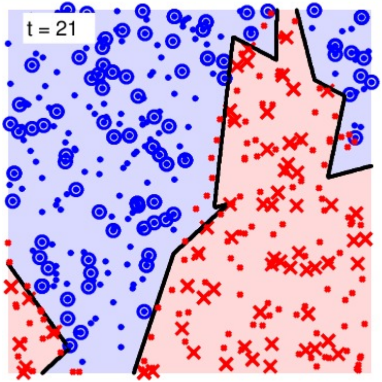


A Complicated Data Set

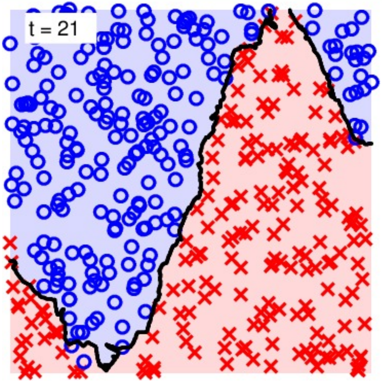


A Complicated Data Set

g_t ($N' = N/2$)



G with first t trees



'easy yet robust' nonlinear model

Modern Machine Learning Models :: Adaptive (or Gradient) Boosting

Apple Recognition Problem

- is this a picture of an apple?
- say, want to teach a class of **6 year olds**
- gather photos under CC-BY-2.0 license on Flicker (**thanks to the authors below!**)

(APAL stands for Apple and Pear Australia Ltd)

Dan Foy `https://flic.kr/p/jNQ55`

APAL `https://flic.kr/p/jzP1VB`

adrianbartel `https://flic.kr/p/bdy2hZ`

ANdrzej cH. `https://flic.kr/p/51DKA8`

Stuart Webster `https://flic.kr/p/9C3Ybd`

nachans `https://flic.kr/p/9XD7Ag`

APAL `https://flic.kr/p/jzRe4u`

Jo Jakeman `https://flic.kr/p/7jwtGp`

APAL `https://flic.kr/p/jzPYNr`

APAL `https://flic.kr/p/jzScif`

Apple Recognition Problem

- is this a picture of an apple?
- say, want to teach a class of **6 year olds**
- gather photos under CC-BY-2.0 license on Flicker (**thanks to the authors below!**)

Mr. Roboto.

https://flic.kr/p/i5BN85

Richard North

https://flic.kr/p/bHhPkB

Richard North

https://flic.kr/p/d8tGou

Emilian Robert Vicol

https://flic.kr/p/bpmGXW

Nathaniel McQueen

https://flic.kr/p/pZv1Mf

Crystal

https://flic.kr/p/kaPYp

jfh686

https://flic.kr/p/6vjRFH

skyseeker

https://flic.kr/p/2MynV

Janet Hudson

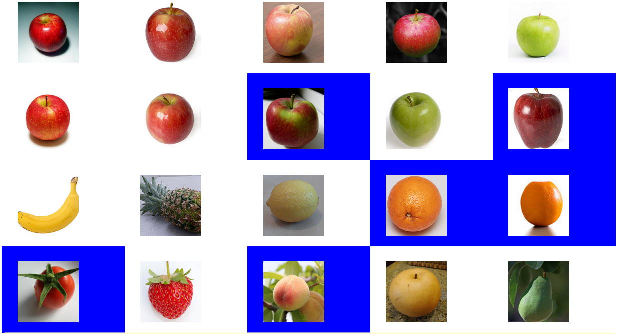
https://flic.kr/p/7QDBbm

Rennett Stowe

https://flic.kr/p/agmnrk

Our Fruit Class Begins

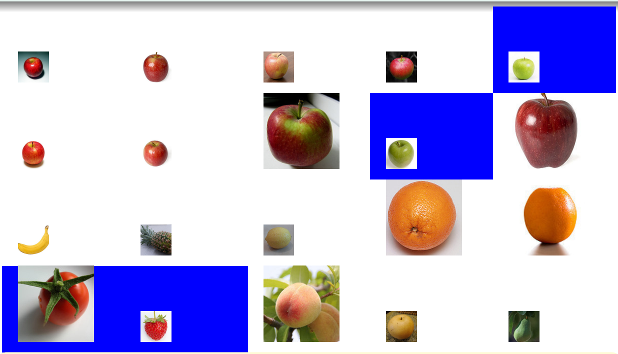
- Teacher: Please look at the pictures of apples and non-apples below. Based on those pictures, how would you describe an apple? Michael?
- Michael: I think apples are **circular**.



(Class): Apples are **circular**.

Our Fruit Class Continues

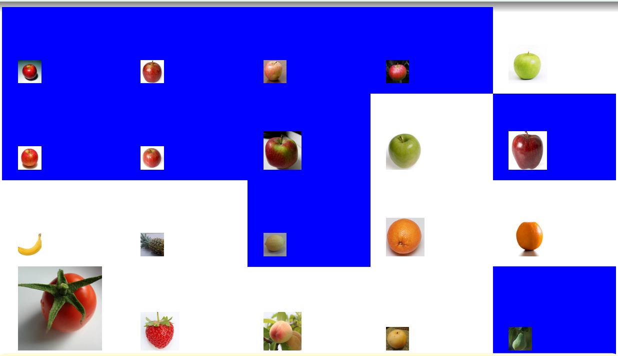
- Teacher: Being circular is a good feature for the apples. However, if you only say circular, you could make several mistakes. What else can we say for an apple? Tina?
- Tina: It looks like apples are **red**.



(Class): Apples are somewhat **circular** and somewhat **red**.

Our Fruit Class Continues More

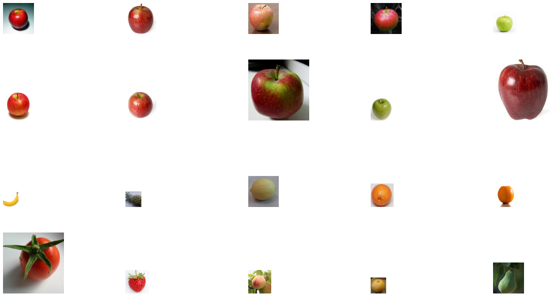
- Teacher: Yes. Many apples are red. However, you could still make mistakes based on circular and red. Do you have any other suggestions, Joey?
- Joey: Apples could also be **green**.



(Class): Apples are somewhat **circular** and somewhat **red** and possibly **green**.

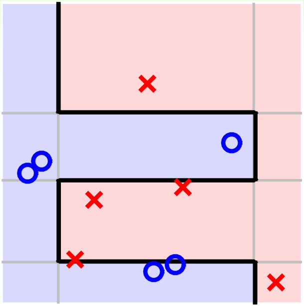
Our Fruit Class Ends

- Teacher: Yes. It seems that apples might be circular, red, green. But you may confuse them with tomatoes or peaches, right? Any more suggestions, Jessica?
- Jessica: Apples have **stems** at the top.



(Class): Apples are somewhat **circular**, somewhat **red**, possibly **green**, and may have **stems** at the top.

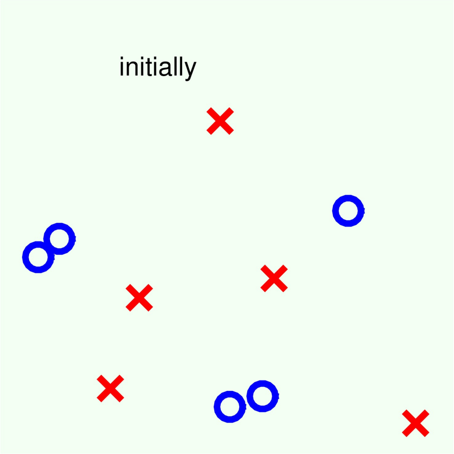
Motivation



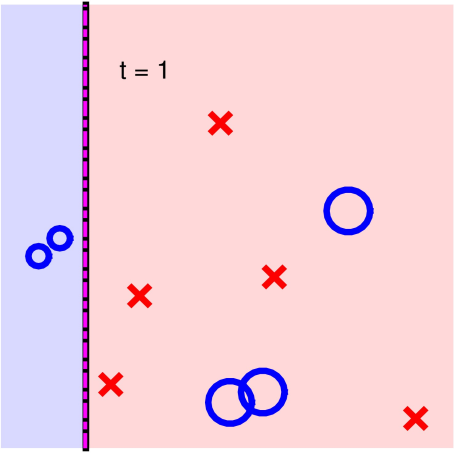
- students: simple hypotheses g_t (like vertical/horizontal lines)
- (Class): sophisticated hypothesis G (like black curve)
- Teacher: a tactic learning algorithm that **directs the students to focus on key examples**

next: demo of such an algorithm

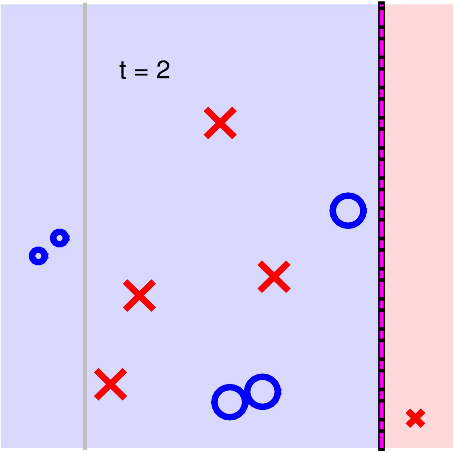
A Simple Data Set



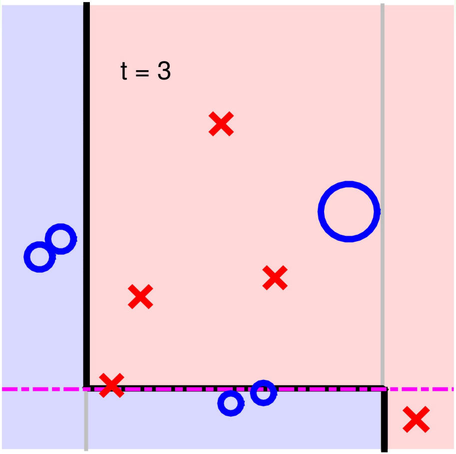
A Simple Data Set



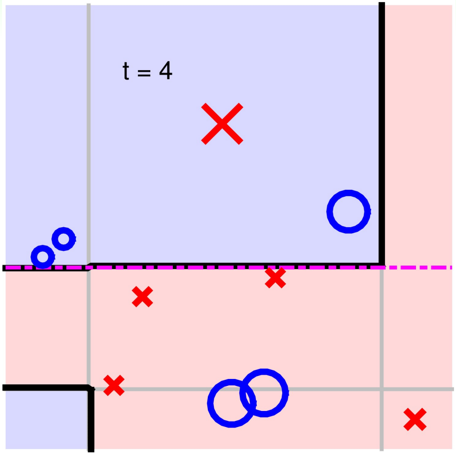
A Simple Data Set



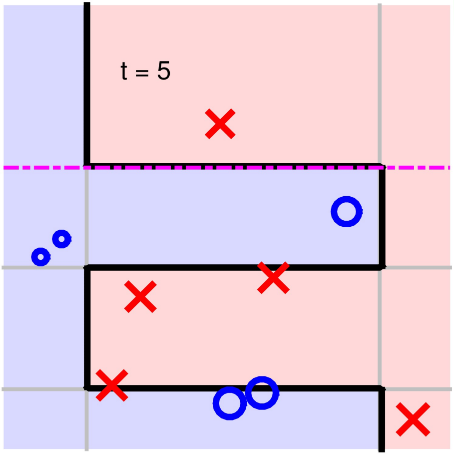
A Simple Data Set



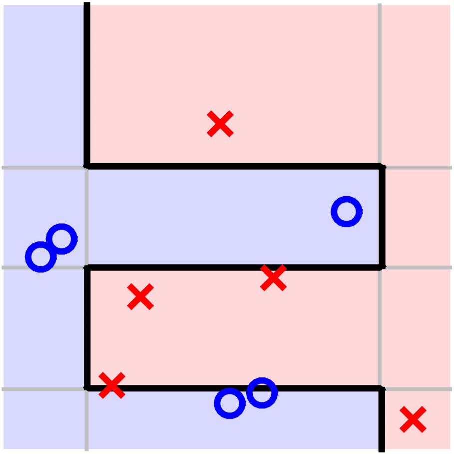
A Simple Data Set



A Simple Data Set



A Simple Data Set



'Teacher'-like algorithm works!

Putting Everything Together

Gradient Boosted Decision Tree (GBDT)

$s_1 = s_2 = \ldots = s_N = 0$

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

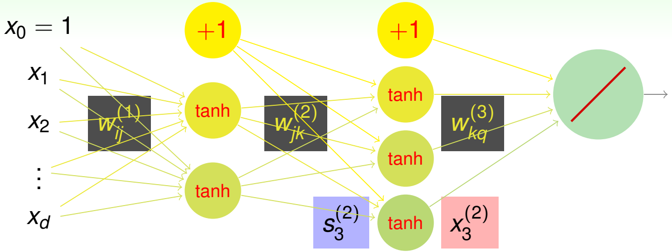
1. obtain g_t by $\mathcal{A}(\{(\mathbf{x}_n, y_n - s_n)\})$ where $\mathcal{A}$ is a (squared-error) regression algorithm
   —**such as 'weak' C&RT?**
2. compute $\alpha_t = \text{OneVarLinearRegression}(\{(g_t(\mathbf{x}_n), y_n - s_n)\})$
3. update $s_n \leftarrow s_n + \alpha_t g_t(\mathbf{x}_n)$

return $G(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})$

GBDT: 'regression sibling' of AdaBoost + decision tree
—**very popular in practice**

Modern Machine Learning Models :: Deep Learning

Physical Interpretation of Neural Network



- each layer: **pattern feature extracted** from data, **remember? :-)**
- how many neurons? how many layers?
  —more generally, **what structure?**
  - subjectively, **your design!**
  - objectively, **validation, maybe?**

structural decisions:
key issue for applying NNet

Shallow versus Deep Neural Networks

shallow: few (hidden) layers; deep: many layers

Shallow NNet

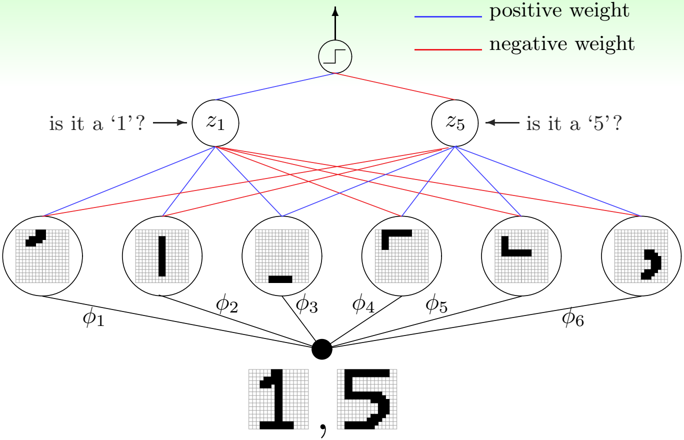
- more **efficient** to train (◯)
- **simpler** structural decisions (◯)
- theoretically **powerful enough** (◯)

Deep NNet

- **challenging** to train ($\times$)
- **sophisticated** structural decisions ($\times$)
- **'arbitrarily' powerful** (◯)
- more **'meaningful'?** (see next slide)

deep NNet (**deep learning**)
gaining attention in recent years

Meaningfulness of Deep Learning



- **'less burden'** for each layer: **simple** to **complex** features
- natural for **difficult** learning task with **raw features**, like **vision**

deep NNet: currently popular in **vision/speech/**...

Challenges and Key Techniques for Deep Learning

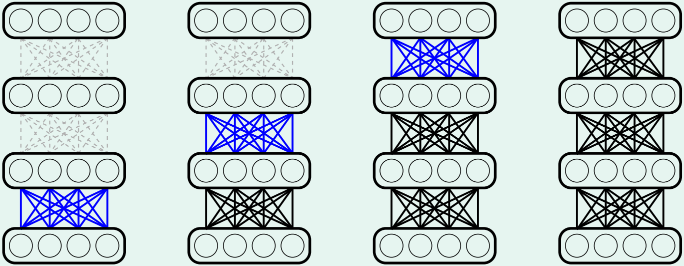
- difficult **structural decisions**:
  - subjective with **domain knowledge**: like **convolutional NNet** for images
- high **model complexity**:
  - no big worries if **big enough data**
  - **regularization** towards noise-tolerant: like
    - **dropout** (tolerant when network corrupted)
    - **denoising** (tolerant when input corrupted)
- hard **optimization problem**:
  - **careful initialization** to avoid bad local minimum: called **pre-training**
- huge **computational complexity** (worsen with **big data**):
  - novel hardware/architecture: like **mini-batch with GPU**

IMHO, careful **regularization** and **initialization** are key techniques

A Two-Step Deep Learning Framework

Simple Deep Learning

1. for $\ell = 1, \ldots, L$, **pre-train** $\left\{w_{ij}^{(\ell)}\right\}$ assuming $w_*^{(1)}, \ldots w_*^{(\ell-1)}$ fixed



(a) (b) (c) (d)

2. **train with backprop** on **pre-trained** NNet to **fine-tune** all $\left\{w_{ij}^{(\ell)}\right\}$

different deep learning models deal with the steps somewhat differently

Mini-Summary

Modern Machine Learning Models

- Support Vector Machine
  large-margin boundary ranging from linear to non-linear
- Random Forest
  uniform blending of many many decision trees
- Adaptive (or Gradient) Boosting
  keep adding simple hypotheses to gang
- Deep Learning
  neural network with deep architecture and careful design

Thank you!!