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/ntum lone
    - “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** ⇔ improve some performance measure (e.g. prediction accuracy)

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

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. :-)

Hsuan-Tien Lin (NTU CSIE)
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

<table>
<thead>
<tr>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>23 years</td>
</tr>
<tr>
<td>gender</td>
<td>female</td>
</tr>
<tr>
<td>annual salary</td>
<td>NTD 1,000,000</td>
</tr>
<tr>
<td>year in residence</td>
<td>1 year</td>
</tr>
<tr>
<td>year in job</td>
<td>0.5 year</td>
</tr>
<tr>
<td>current debt</td>
<td>200,000</td>
</tr>
</tbody>
</table>

**what to learn? (for improving performance):**

‘approve credit card good for bank?’
### Basic Notations

- **input**: $x \in X$ (customer application)
- **output**: $y \in Y$ (good/bad after approving credit card)
- **unknown** underlying pattern to be learned $\iff$ target function: $f : X \rightarrow Y$ (ideal credit approval formula)
- **data** $\iff$ training examples: $D = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \}$ (historical records in bank)
- **hypothesis** $\iff$ skill with hopefully good performance: $g : X \rightarrow Y$ ('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 $\leq$ 2 (really?)
  —all candidate formula being considered: hypothesis set $\mathcal{H}$
  —procedure to learn best formula: algorithm $\mathcal{A}$

\[ \{(x_n, y_n)\} \text{ from } f \xrightarrow{\text{ML (A, H)}} g \]
Practical Definition of Machine Learning

unknown target function
\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]
(ideal credit approval formula)

training examples
\[ \mathcal{D} : (x_1, y_1), \cdots, (x_N, y_N) \]
(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)

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 \( \rightarrow \) **ML** \( \rightarrow \) 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**: 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 \( \approx \) \( \text{recent strength of student} > \text{difficulty of question} \)

- 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)

\textbf{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
Regression for Recommender System (1/2)

- **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**?
Regression for Recommender System (2/2)

A Possible ML Solution

- **pattern:**
  
  \[
  \text{rating} \leftarrow \text{viewer/movie factors}
  \]

- **learning:**
  
  \[
  \text{known rating} \rightarrow \text{learned factors} \\
  \rightarrow \text{unknown rating prediction}
  \]

---

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

**Supervised versus Unsupervised**

**supervised multiclass classification**

**unsupervised** multiclass classification

⇐⇒ ‘clustering’

**Other Clustering Problems**

- articles ⇒ topics
- consumer profiles ⇒ consumer groups

**clustering**: a challenging but useful problem
Learning from Data

Types of Machine Learning

Supervised versus Unsupervised

coin recognition with \( y_n \)

- supervised multiclass classification

coin recognition without \( y_n \)

- unsupervised multiclass classification

\[ \Leftrightarrow \text{’clustering’} \]

Other Clustering Problems

- articles \( \Rightarrow \) topics
- consumer profiles \( \Rightarrow \) consumer groups

clustering: a challenging but useful problem
Semi-supervised: Coin Recognition with Some $y_n$

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 $y_n = \text{sit}$ when $x_n = \text{‘sit down’}$
• but can ‘punish’ to say $\tilde{y}_n = \text{pee is wrong}$

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

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

reinforcement: learn with ‘partial/implicit information’ (often sequentially)
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 $y_n = \text{sit}$ when $x_n = \text{‘sit down’}$
• but can ‘reward’ to say $\tilde{y}_n = \text{sit is good}$

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

• (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
unknown target function $f : \mathcal{X} \rightarrow \mathcal{Y}$
(ideal credit approval formula)

training examples $\mathcal{D} : (x_1, y_1), \cdots, (x_N, y_N)$
(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)

- choose error measure: how $g(x) \approx f(x)$
- decide hypothesis set $\mathcal{H}$
- optimize error and more on $\mathcal{D}$ as $\mathcal{A}$
- pray for generalization: whether $g(x) \approx f(x)$ for unseen $x$
Choose Error Measure

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

**in-sample (within data)**

$$E_{\text{in}}(g) = \frac{1}{N} \sum_{n=1}^{N} \text{err}(g(x_n), f(x_n))$$

**out-of-sample (future data)**

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

will start from 0/1 error $\text{err}(\tilde{y}, y) = [\tilde{y} \neq y]$ for **classification**
Choose Hypothesis Set (for Credit Approval)

<table>
<thead>
<tr>
<th>age</th>
<th>23 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>annual salary</td>
<td>NTD 1,000,000</td>
</tr>
<tr>
<td>year in job</td>
<td>0.5 year</td>
</tr>
<tr>
<td>current debt</td>
<td>200,000</td>
</tr>
</tbody>
</table>

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

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

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

- $\mathcal{Y} = \{+1(\text{good}), -1(\text{bad})\}$, 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

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

- want: $g \approx f$ (hard when $f$ unknown)
- almost necessary: $g \approx f$ on $\mathcal{D}$, ideally $g(x_n) = f(x_n) = y_n$
- 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**
worked like a charm with < 20 lines!!
—A fault confessed is half redressed. :-)

Hsuan-Tien Lin (NTU CSIE)
worked like a charm with < 20 lines!!
—A fault confessed is half redressed. :-}
worked like a charm with < 20 lines!!
—A fault confessed is half redressed. :-)

Hsuan-Tien Lin (NTU CSIE)
worked like a charm with < 20 lines!!

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

Hsuan-Tien Lin (NTU CSIE)
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. :-)

Hsuan-Tien Lin (NTU CSIE)
worked like a charm with \(<20\) lines!!

—A fault confessed is half redressed. :-)}
worked like a charm with < 20 lines!!
—A fault confessed is half redressed. :-}
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

- alternatives

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

- examples (picture $x_n$, label $y_n$)

learning algorithm

- hypothesis set $\mathcal{H}$

- good hypothesis $g(x) \approx f(x)$

challenge:

see only $\{(x_n, y_n)\}$ without knowing $f$ nor noise

$\Rightarrow$ generalize to unseen $(x, y)$ w.r.t. $f(x)$
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 $A$ takes data $D$ and hypotheses $H$ to get hypothesis $g$

- **Types of Machine Learning**
  - variety of problems almost everywhere

- **Step-by-step Machine Learning**
  - error, hypotheses, optimize, generalize
### Fundamental Machine Learning Models

- Linear Regression
- Logistic Regression
- Nonlinear Transform
- Decision Tree
Fundamental Machine Learning Models :: Linear Regression
Credit Limit Problem

unknown target function
\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]
(ideal credit limit formula)

training examples
\[ \mathcal{D} : (x_1, y_1), \ldots, (x_N, y_N) \]
(historical records in bank)

hypothesis set
\[ \mathcal{H} \]
(set of candidate formula)

learning algorithm
\[ \mathcal{A} \]

final hypothesis
\[ g \approx f \]
('learned' formula to be used)

\[ \mathcal{Y} = \mathbb{R} : \text{regression} \]

| 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
Linear Regression Hypothesis

- 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
Fundamental Machine Learning Models

Linear Regression

Illustration of Linear Regression

\[ \mathbf{x} = (\mathbf{x}) \in \mathbb{R} \]

\[ \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^2 \]

linear regression: find lines/hyperplanes with small residuals
The Error Measure

popular/historical error measure:
squared error \( \text{err}(\hat{y}, y) = (\hat{y} - y)^2 \)

\[
E_{\text{in}}(h\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} (h(x_n) - y_n)^2
\]

\[
E_{\text{out}}(\mathbf{w}) = \mathbb{E}_{(x,y) \sim \mathcal{P}} (\mathbf{w}^T x - y)^2
\]

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

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

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

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

—not possible to ‘roll down’

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

Linear Regression

Linear Regression Algorithm

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

$$
X = \begin{bmatrix}
- & - & x_1^T & - & - \\
- & - & x_2^T & - & - \\
& & \cdots & & \\
- & - & x_N^T & - & - \\
\end{bmatrix}_{N \times (d+1)}
$$

$$
y = \begin{bmatrix}
y_1 \\
y_2 \\
& \cdots \\
y_N \\
\end{bmatrix}_{N \times 1}
$$

2. calculate pseudo-inverse

$$
X^\dagger_{(d+1) \times N}
$$

3. return

$$
\mathbf{w}_{LIN} = X^\dagger y_{(d+1) \times 1}
$$

simple and efficient

with **good † routine**
Is Linear Regression a ‘Learning Algorithm’?

\[ \mathbf{w}_{\text{LIN}} = \mathbf{X}^\dagger \mathbf{y} \]

No!
- analytic (closed-form) solution, ‘instantaneous’
- not improving \( E_{\text{in}} \) nor \( E_{\text{out}} \) iteratively

Yes!
- good \( E_{\text{in}} \)?
  - yes, optimal!
- good \( E_{\text{out}} \)?
  - 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)

unknown target
distribution $P(y|x)$
containing $f(x) + \text{noise}$

training examples
$D: (x_1, y_1), \cdots, (x_N, y_N)$

learning
algorithm $\mathcal{A}$

final hypothesis
$g \approx f$

error measure
$\hat{\text{err}}$

hypothesis set
$\mathcal{H}$

binary classification:
ideal $f(x) = \text{sign} \left( P(+1|x) - \frac{1}{2} \right) \in \{-1, +1\}$
because of classification error

<table>
<thead>
<tr>
<th>age</th>
<th>40 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>gender</td>
<td>male</td>
</tr>
<tr>
<td>blood pressure</td>
<td>130/85</td>
</tr>
<tr>
<td>cholesterol level</td>
<td>240</td>
</tr>
<tr>
<td>weight</td>
<td>70</td>
</tr>
</tbody>
</table>

heart disease? yes
Heart Attack Prediction Problem (2/2)

<table>
<thead>
<tr>
<th>age</th>
<th>40 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>gender</td>
<td>male</td>
</tr>
<tr>
<td>blood pressure</td>
<td>130/85</td>
</tr>
<tr>
<td>cholesterol level</td>
<td>240</td>
</tr>
<tr>
<td>weight</td>
<td>70</td>
</tr>
</tbody>
</table>

heart attack? 80% risk

unknown target distribution $P(y|x)$ containing $f(x) +$ noise

training examples $D: (x_1, y_1), \cdots, (x_N, y_N)$

learning algorithm $A$

hypothesis set $\mathcal{H}$

final hypothesis $g \approx f$

error measure $err$

‘soft’ binary classification:

$$f(x) = P(+1|x) \in [0, 1]$$
Soft Binary Classification

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

ideal (noiseless) data

\[
\begin{align*}
(x_1, y'_1) &= 0.9 = P(+1|x_1) \\
(x_2, y'_2) &= 0.2 = P(+1|x_2) \\
\vdots \\
(x_N, y'_N) &= 0.6 = P(+1|x_N)
\end{align*}
\]

actual (noisy) data

\[
\begin{align*}
(x_1, y_1) &= \circ \sim P(y|x_1) \\
(x_2, y_2) &= \times \sim P(y|x_2) \\
\vdots \\
(x_N, y_N) &= \times \sim P(y|x_N)
\end{align*}
\]

same data as hard binary classification, different target function
Fundamental Machine Learning Models

Logistic Regression

Soft Binary Classification

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

<table>
<thead>
<tr>
<th>ideal (noiseless) data</th>
<th>actual (noisy) data</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1, y'_1 = 0.9 = P(+1</td>
<td>x_1)$</td>
</tr>
<tr>
<td>$x_2, y'_2 = 0.2 = P(+1</td>
<td>x_2)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_N, y'_N = 0.6 = P(+1</td>
<td>x_N)$</td>
</tr>
</tbody>
</table>

same data as hard binary classification, different target function
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:

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

<table>
<thead>
<tr>
<th>age</th>
<th>40 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>gender</td>
<td>male</td>
</tr>
<tr>
<td>blood pressure</td>
<td>130/85</td>
</tr>
<tr>
<td>cholesterol level</td>
<td>240</td>
</tr>
</tbody>
</table>
Minimizing $E_{\text{in}}(w)$

A popular error: $E_{\text{in}}(w) = \frac{1}{N} \sum_{n=1}^{N} \ln (1 + \exp(-y_n w^T x_n))$ called cross-entropy derived from maximum likelihood.

- $E_{\text{in}}(w)$: continuous, differentiable, twice-differentiable, convex
- How to minimize? Locate valley

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

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 \( \eta \): (assumed) positive

gradient descent: \( \mathbf{v} \propto -\nabla E_{\text{in}}(\mathbf{w}_t) \)
Fundamental Machine Learning Models

Putting Everything Together

Logistic Regression Algorithm

initialize \( w_0 \)
For \( t = 0, 1, \ldots \)

1. compute

\[
\nabla E_{\text{in}}(w_t) = \frac{1}{N} \sum_{n=1}^{N} \theta \left( -y_n w_t^T x_n \right) (-y_n x_n)
\]

2. update by

\[
w_{t+1} \leftarrow w_t - \eta \nabla E_{\text{in}}(w_t)
\]

...until \( \nabla E_{\text{in}}(w_{t+1}) \approx 0 \) or enough iterations

return last \( w_{t+1} \) 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 error = 0/1

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

**linear regression**

\[ h(\mathbf{x}) = s \]

friendly error = squared

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

**logistic regression**

\[ h(\mathbf{x}) = \theta(s) \]

plausible error = 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 $s = w^T x$

but limited ...

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

how to break the limit of linear hypotheses
• $\mathcal{D}$ not linear separable

• but **circular separable** by a circle of radius $\sqrt{0.6}$ centered at origin:

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

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

Hsuan-Tien Lin  (NTU CSIE)
Circular Separable and Linear Separable

\[ h(x) = \text{sign} \begin{pmatrix} 0.6 & 1 \ \tilde{w}_0 & z_0 \\ \tilde{w}_1 & z_1 \\ \tilde{w}_2 & z_2 \end{pmatrix} \]

\[ = \text{sign} (\tilde{w}^T z) \]

- \{ (x_n, y_n) \} circular separable \implies \{ (z_n, y_n) \} linear separable
- \( x \in \mathcal{X} \xleftarrow{\Phi} z \in \mathcal{Z} \) : (nonlinear) feature transform \( \Phi \)

\[ \text{circular separable in } \mathcal{X} \implies \text{linear separable in } \mathcal{Z} \]
General Quadratic Hypothesis Set

a ‘bigger’ $\mathcal{Z}$-space with $\Phi_2(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

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

$$\mathcal{H}_{\Phi_2} = \left\{ h(x) : h(x) = \tilde{h}(\Phi_2(x)) \text{ for some linear } \tilde{h} \text{ on } \mathcal{Z} \right\}$$

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

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

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

include lines and constants as degenerate cases
**Good Quadratic Hypothesis**

\[ \mathcal{Z}\text{-space} \]
- perceptrons
- good perceptron
- separating perceptron

\[ \mathcal{X}\text{-space} \]
- quadratic hypotheses
- good quadratic hypothesis
- separating quadratic hypothesis

- want: get **good perceptron** in \( \mathcal{Z}\text{-space} \)
- known: get **good perceptron** in \( \mathcal{X}\text{-space} \) with data \( \{(x_n, y_n)\} \)

solution: get **good perceptron** in \( \mathcal{Z}\text{-space} \) with data \( \{(z_n = \Phi_2(x_n), y_n)\} \)
The Nonlinear Transform Steps

1. Transform original data \( \{(x_n, y_n)\} \) to \( \{(z_n = \Phi(x_n), y_n)\} \) by \( \Phi \)

2. Get a good perceptron \( \tilde{w} \) using \( \{(z_n, y_n)\} \) and your favorite linear algorithm \( A \)

3. Return \( g(x) = \text{sign} \left( \tilde{w}^T \Phi(x) \right) \)
Nonlinear Model via Nonlinear $\Phi + \text{Linear Models}$

two choices:
- feature transform $\Phi$
- linear model $A$, not just binary classification

Pandora’s box :-):
- can now freely do quadratic PLA, quadratic regression, cubic regression, ..., polynomial regression
more generally, not just polynomial:

raw (pixels) \rightarrow \text{domain knowledge} \rightarrow \text{concrete (intensity, symmetry)}

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

**Computation/Storage Price**

\[ \Phi_Q(x) = \left( 1, \ x_1, x_2, \ldots, x_d, \ x_1^2, x_1 x_2, \ldots, x_d^2, \ \ldots, \ x_1^Q, x_1^{Q-1} x_2, \ldots, x_d^Q \right) \]

- 1 + \tilde{d} dimensions
- \( \tilde{w}_0 \) + others
- \# ways of \( \leq Q \)-combination from \( d \) kinds with repetitions

\[ \binom{Q+d}{Q} = \binom{Q+d}{d} = O(Q^d) \]

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

Q large \( \Rightarrow \) difficult to compute/store

AND curve too complicated
which one do you prefer? :-)

- $\Phi_1$ ‘visually’ preferred
- $\Phi_4$: $E_{\text{in}}(g) = 0$ but overkill

how to pick $Q$?

**model selection** (to be discussed) important
Fundamental Machine Learning Models :: Decision Tree
**Decision Tree for Watching MOOC Lectures**

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

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

**Decision tree:** arguably one of the most **human-mimicking models**
Recursive View of Decision Tree

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

Recursive View

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

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

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

Tree Diagram:
- quitting time?
  - < 18:30
    - has a date?
      - true
        - deadline?
          - Y
            - > 21:30
              - between < −2 days
                - N
                - Y
              - between > 2 days
                - N
                - Y
        - false
          - > 21:30
            - between < −2 days
              - N
              - Y
        - < 18:30
          - between > 21:30
            - N
            - Y
          - between < −2 days
            - N
            - Y
      - between > 2 days
        - N
        - Y
A Basic Decision Tree Algorithm

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

function DecisionTree(data \( D = \{ (x_n, y_n)\}_{n=1}^{N} \))

if termination criteria met
    return base hypothesis \( g_t(x) \)
else
    1. learn branching criteria \( b(x) \)
    2. split \( D \) to \( C \) parts \( D_c = \{ (x_n, y_n) : b(x_n) = c \} \)
    3. build sub-tree \( G_c \leftarrow \text{DecisionTree}(D_c) \)
    4. return \( G(x) = \sum_{c=1}^{C} [b(x) = c] G_c(x) \)

four choices: number of branches, branching criteria, termination criteria, & base hypothesis
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™ of California Statistical Software
Fundamental Machine Learning Models

Decision Tree

A Simple Data Set

Hsuan-Tien Lin (NTU CSIE)
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
Fundamental Machine Learning Models

Decision Tree

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 secret ‘force’ to enrich your model
- **Decision Tree**
  - human-like explainable model learned recursively
Hazard of Overfitting

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,

\[ E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{8}{N} \ln \left( \frac{4(2N)^{d_{\text{VC}}(\mathcal{H})}}{\delta} \right)} \]

\[ \Omega: \text{price of using } \mathcal{H} \]

\( d_{\text{VC}}(\mathcal{H}) \): complexity of \( \mathcal{H} \),
\[ \approx \# \text{ of parameters to describe } \mathcal{H} \]

- \( d_{\text{VC}} \uparrow: E_{\text{in}} \downarrow \text{ but } \Omega \uparrow \)
- \( d_{\text{VC}} \downarrow: \Omega \downarrow \text{ but } E_{\text{in}} \uparrow \)
- best \( d_{\text{VC}}^* \) in the middle

**powerful \( \mathcal{H} \) not always good!**
Bad Generalization

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

bad generalization: low $E_{in}$, high $E_{out}$
Bad Generalization and Overfitting

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

bad generalization: low $E_{in}$, high $E_{out}$; **overfitting**: lower $E_{in}$, higher $E_{out}$
Cause of Overfitting: A Driving Analogy

<table>
<thead>
<tr>
<th>learning</th>
<th>driving</th>
</tr>
</thead>
<tbody>
<tr>
<td>overfit</td>
<td>commit a car accident</td>
</tr>
<tr>
<td>use excessive $d_{vc}$</td>
<td>‘drive too fast’</td>
</tr>
<tr>
<td>noise</td>
<td>bumpy road</td>
</tr>
<tr>
<td>limited data size $N$</td>
<td>limited observations about road condition</td>
</tr>
</tbody>
</table>

let’s ‘visualize’ overfitting
Impact of Noise and Data Size

reasons of serious overfitting:

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

overfitting ‘easily’ happens
(more on ML Foundations, Lecture 13)
Hazard of Overfitting

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

<table>
<thead>
<tr>
<th>Learning</th>
<th>Driving</th>
</tr>
</thead>
<tbody>
<tr>
<td>overfit</td>
<td>commit a car accident</td>
</tr>
<tr>
<td>use excessive $d_{vc}$</td>
<td>‘drive too fast’</td>
</tr>
<tr>
<td>noise</td>
<td>bumpy road</td>
</tr>
<tr>
<td>limited data size $N$</td>
<td>limited observations about road condition</td>
</tr>
</tbody>
</table>

### Practical Techniques
- start from simple model
- data cleaning/pruning
- data hinting
- regularization
- validation
- drive slowly
- use more accurate road information
- exploit more road information
- put the brakes
- monitor the dashboard

all very **practical** techniques to combat overfitting
Hazard of Overfitting ::
Data Manipulation and Regularization
• 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

- 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**

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

**VC Bound**

\[ E_{\text{out}}(w) \leq E_{\text{in}}(w) + \Omega(\mathcal{H}) \]

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

minimizing \( E_{\text{aug}} \):

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

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

how to choose?
validation!
Hazard of Overfitting :: Validation
Model Selection Problem

Hazard of Overfitting

Validation

• given: $M$ models $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$, each with corresponding algorithm $A_1, A_2, \ldots, A_M$
• goal: select $\mathcal{H}_{m^*}$ such that $g_{m^*} = A_{m^*}(D)$ is of low $E_{\text{out}}(g_{m^*})$
• unknown $E_{\text{out}}$, as always :-)
• arguably the most important practical problem of ML

which one do you prefer? :-)

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

Hsuan-Tien Lin  (NTU CSIE)
**Validation Set** $\mathcal{D}_{\text{val}}$

- $\mathcal{D}_{\text{val}} \subset \mathcal{D}$: called **validation set**—‘on-hand’ simulation of test set
- to connect $E_{\text{val}}$ with $E_{\text{out}}$:
  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

\[
E_{\text{out}}(g_m) \leq E_{\text{val}}(g_m) + 'small'
\]
Hazard of Overfitting

Validation

Model Selection by Best $E_{\text{val}}$

\[ m^* = \arg\min_{1 \leq m \leq M} (E_m = E_{\text{val}}(A_m(D_{\text{train}}))) \]

- generalization guarantee for all $m$:
  \[ E_{\text{out}}(g_{m^*}(D)) \leq E_{\text{val}}(g_{m^*}) + \text{'small'} \]

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

\[
E_{\text{out}} \left( \begin{pmatrix} g_{m^*} \\ A_{m^*}(D) \end{pmatrix} \right) \leq E_{\text{out}} \left( \begin{pmatrix} g_{m^*} \\ A_{m^*}(D_{\text{train}}) \end{pmatrix} \right)
\]

\[
E_{\text{out}}(g_{m^*}) \leq E_{\text{out}}(g_{m^*}) \leq E_{\text{val}}(g_{m^*}) + \text{'small'}
\]
**V-fold Cross Validation**

making validation more stable

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

\[
\begin{array}{cccccc}
\mathcal{D}_1 & \mathcal{D}_2 & \mathcal{D}_3 & \mathcal{D}_4 & \mathcal{D}_5 & \mathcal{D}_6 & \mathcal{D}_7 & \mathcal{D}_8 & \mathcal{D}_9 & \mathcal{D}_{10} \\
\text{train} & \text{validate} & \text{train} \\
\end{array}
\]


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

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

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

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
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 $\Phi_2$: $z = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$, $d_{VC} = 6$
- or $z = (1, x_1^2, x_2^2)$, $d_{VC} = 3$, after visualizing?
- or better $z = (1, x_1^2 + x_2^2)$, $d_{VC} = 2$?
- or even better $z = (\text{sign}(0.6 - x_1^2 - x_2^2))$?

—careful about your brain's 'model complexity'

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

Hsuan-Tien Lin (NTU CSIE)
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)**
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

\[
\begin{align*}
\max_w \quad & \text{fatness}(w) \\
\text{subject to} \quad & w \text{ classifies every } (x_n, y_n) \text{ correctly} \\
\text{fatness}(w) &= \min_{n=1,\ldots,N} \text{distance}(x_n, w)
\end{align*}
\]

- fatness: formally called \textit{margin}
- correctness: \( y_n = \text{sign}(w^T x_n) \)

initial goal: find \textit{largest-margin} \textit{separating} hyperplane
Motivation: Large-Margin Separating Hyperplane

max \( \frac{\text{margin}(w)}{w} \)

subject to every \( y_n w^T x_n > 0 \)

\[ \text{margin}(w) = \min_{n=1,\ldots,N} \text{distance}(x_n, w) \]

- fatness: formally called \text{margin}
- correctness: \( y_n = \text{sign}(w^T x_n) \)

initial goal: find \text{largest-margin separating} hyperplane
Modern Machine Learning Models

Support Vector Machine

Soft-Margin Support Vector Machine

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

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

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

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

linear: ‘best’ linear classification model;
non-linear: ‘leading’ non-linear classification model **for mid-sized data**
Gaussian kernel $K(x, x') = \exp(-\gamma \|x - x'\|^2)$

$$g_{\text{SVM}}(x) = \text{sign} \left( \sum_{SV} \alpha_n y_n K(x_n, x) + b \right)$$

$$= \text{sign} \left( \sum_{SV} \alpha_n y_n \exp\left(-\gamma \|x - x_n\|^2\right) + b \right)$$

- linear combination of Gaussians centered at SVs $x_n$
- also called Radial Basis Function (RBF) kernel

Gaussian SVM:
find $\alpha_n$ to combine Gaussians centered at $x_n$
& achieve large margin in infinite-dim. space
## Support Vector Mechanism

<table>
<thead>
<tr>
<th>large-margin hyperplanes</th>
<th>not many</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ higher-order transforms with kernel trick</td>
<td>sophisticated</td>
</tr>
<tr>
<td>+ noise tolerance of soft-margin</td>
<td></td>
</tr>
</tbody>
</table>

- transformed vector $z = \Phi(x) \Rightarrow$ efficient kernel $K(x, x')$
- store optimal $w \Rightarrow$ store a few SVs and $\alpha_n$

new possibility by Gaussian SVM: infinite-dimensional linear classification, with generalization ‘guarded by’ large-margin :-(
Practical Need: Model Selection

- large $\gamma \rightarrow$ sharp Gaussians $\rightarrow$ ‘overfit’?
- complicated even for $(C, \gamma)$ of Gaussian SVM
- more combinations if including other kernels or parameters

how to select? validation :-)

Hsuan-Tien Lin (NTU CSIE)
Step-by-step Use of SVM

strongly recommended: ‘A Practical Guide to Support Vector Classification’

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 \((\gamma, C)\)
4. use the best \((\gamma, C)\) 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(D)
For t = 1, 2, ..., T
  1. request size-N' data \( \tilde{D}_t \) by bootstrapping with \( D \)
  2. obtain tree \( g_t \) by DTREE(\( \tilde{D}_t \))
return \( G = \text{Uniform}(\{g_t\}) \)

function DTREE(D)
if termination
  return base \( g_t \)
else
  1. learn \( b(x) \) and split \( D \) to \( D_c \) by \( b(x) \)
  2. build \( G_c \leftarrow \text{DTREE}(D_c) \)
  3. return \( G(x) = \sum_{c=1}^{C} [b(x) = c] G_c(x) \)

- highly parallel/efficient to learn
- inherit pros of C&RT
- eliminate cons of fully-grown tree
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 $\Phi(\mathbf{x}) = (x_{i_1}, x_{i_2}, x_{i_d'})$

with $d' < d$ for $g(\Phi(\mathbf{x}))$

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
Modern Machine Learning Models

Random Forest

A Complicated Data Set

\[ g_t \left( N' = N/2 \right) \quad G \text{ with first } t \text{ trees} \]
A Complicated Data Set

\[ g_t \left( N' = \frac{N}{2} \right) \quad G \text{ with first } t \text{ trees} \]
A Complicated Data Set

\[ g_t \left( N' = \frac{N}{2} \right) \]

\[ G \text{ with first } t \text{ trees} \]
Modern Machine Learning Models

Random Forest

A Complicated Data Set

\[ g_t \left( N' = N/2 \right) \]

\[ G \text{ with first } t \text{ trees} \]
A Complicated Data Set

\[ g_t (N' = N/2) \]

\[ G \text{ with first } t \text{ trees} \]

‘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/6y2hZ

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/7jw7Gp

APAL
https://flic.kr/p/jzPYN

APAL
https://flic.kr/p/jzScif
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!)

Mr. Roboto.
https://flic.kr/p/i5BN85

Richard North
https://flic.kr/p/bHhPkB

Richard North
https://flic.kr/p/d8tGou

Emilian Vicol
https://flic.kr/p/bpmGXW

Robert 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

Hsuan-Tien Lin (NTU CSIE)

Quick Tour of Machine Learning
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.
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.
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.
Modern Machine Learning Models

Adaptive (or Gradient) Boosting

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

initially

[Diagram of a simple data set with red 'x' and blue 'o' markers]
A Simple Data Set

t = 1
Adaptive (or Gradient) Boosting

A Simple Data Set

$t = 2$
A Simple Data Set

t = 3
Modern Machine Learning Models

Adaptive (or Gradient) Boosting

A Simple Data Set

t = 4

Hsuan-Tien Lin (NTU CSIE)

Quick Tour of Machine Learning

119/128
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 \( A(\{(x_n, y_n - s_n)\}) \) where \( A \) is a (squared-error) regression algorithm
   —such as ‘weak’ C&RT?

2. compute \( \alpha_t = \text{OneVarLinearRegression}(\{(g_t(x_n), y_n - s_n)\}) \)

3. update \( s_n \leftarrow s_n + \alpha_t g_t(x_n) \)

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

**GBDT:** ‘regression sibling’ of AdaBoost + decision tree
—very popular in practice
Modern Machine Learning Models :: Deep Learning
• 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**
Modern Machine Learning Models

Deep Learning

Shallow versus Deep Neural Networks

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

<table>
<thead>
<tr>
<th>Shallow NNet</th>
<th>Deep NNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>more <strong>efficient</strong> to train (○)</td>
<td><strong>challenging</strong> to train (×)</td>
</tr>
<tr>
<td><strong>simpler</strong> structural decisions (○)</td>
<td><strong>sophisticated</strong> structural decisions (×)</td>
</tr>
<tr>
<td>theoretically <strong>powerful enough</strong> (○)</td>
<td>‘arbitrarily’ <strong>powerful</strong> (○)</td>
</tr>
<tr>
<td></td>
<td>more ‘<strong>meaningful</strong>’? (see next slide)</td>
</tr>
</tbody>
</table>

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/...
Modern Machine Learning Models

Deep Learning

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** $\{w^{(\ell)}_{ij}\}$ assuming $w^{(1)}_*, \ldots, w^{(\ell-1)}_*$ **fixed**

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

Different deep learning models deal with the steps somewhat differently.
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!!