Lecture 1: Machine Learning Problem

Qinfeng (Javen) Shi

30 July 2015

Intro. to Stats. Machine Learning
COMP SCI 4401/7401

Outline of the Course

1. Machine Learning Problem
2. Supervised Learning:
   Classification:
   Support Vector Machines ...
3. Supervised Learning:
   Classification: Boosting
4. Supervised Learning:
   Regression
5. Neural networks and
   Deep learning
6. Dimension reduction
7. Probabilistic Graphical Models
   (PGMs):Representation
8. PGMs: Inference
9. PGMs: Learning parameters
10. PGMs: Learning structures
11. Kernels
12. Learning Theory

Outline of the Course

1. Machine Learning Problem
2. Supervised Learning: Classification: Support Vector Machines ...
3. Supervised Learning: Classification: Boosting
4. Supervised Learning: Regression
5. Neural networks and Deep learning
6. Dimension reduction
7. Probabilistic Graphical Models (PGMs): Representation
8. PGMs: Inference
9. PGMs: Learning parameters
10. PGMs: Learning structures
11. Kernels
12. Learning Theory

Complexity Levels:

- concepts; conceptual
- theory; mathematical
- technique; practical

Table of Contents I

1. Course info

2. Machine Learning
   - 1st Example of Machine Learning
   - What's Machine Learning?
   - Types of Learning
   - Simple is Better

3. Real life problems
   - Typical assumptions
   - Large-scale data
   - Structured data
   - Changing environment

Enrolment

We will use Forum for messages, assignments and slides
Link: https://forums.cs.adelaide.edu.au/login/index.php
Go to Course "COMP SCI 4401 Introduction to Statistical Machine Learning".
For those enrolled in 7401, please go to above on Forum too.

Assessment

The course includes the following assessment components:

- Final written exam at 55% (open book).
- Three assignments at 15% each (report and code).

Required Skills

- Ability to program in Matlab, C/C++ is required.
- Knowing some basic statistics, probability, linear algebra and optimisation would be helpful, but not essential. They will be covered when needed.

Recommended books

1. Pattern Recognition and Machine Learning by Bishop, Christopher M.
2. Kernel Methods for Pattern Analysis by John Shawe-Taylor, Nello Cristianini
3. Convex Optimization by Stephen Boyd and Lieven Vandenberghe

Book 1 is for machine learning in general. Book 2 focuses on kernel methods with pseudo code and some theoretical analysis. Book 3 gives introduction to (Convex) Optimization.

External courses

- Learning from the Data by Yaser Abu-Mostafa in Caltech[1].
- Machine Learning by Andrew Ng in Stanford.
- Machine Learning (or related courses) by Nando de Freitas in UBC (now Oxford).

[1]some pictures and examples here are from Yaser's course

Example: Predicting how a viewer will rate a movie

Netflix

10% improvement

Example: Predicting how a viewer will rate a movie

Netflix

10% improvement = 1 million dollar prize

Example: Predicting how a viewer will rate a movie

Netflix

10% improvement = 1 million dollar prize

The essence of machine learning:

Example: Predicting how a viewer will rate a movie

Netflix

10% improvement = 1 million dollar prize

The essence of machine learning:

- A pattern exists

Example: Predicting how a viewer will rate a movie

Netflix

10% improvement = 1 million dollar prize

The essence of machine learning:

- A pattern exists
- We cannot pin it down mathematically

Example: Predicting how a viewer will rate a movie

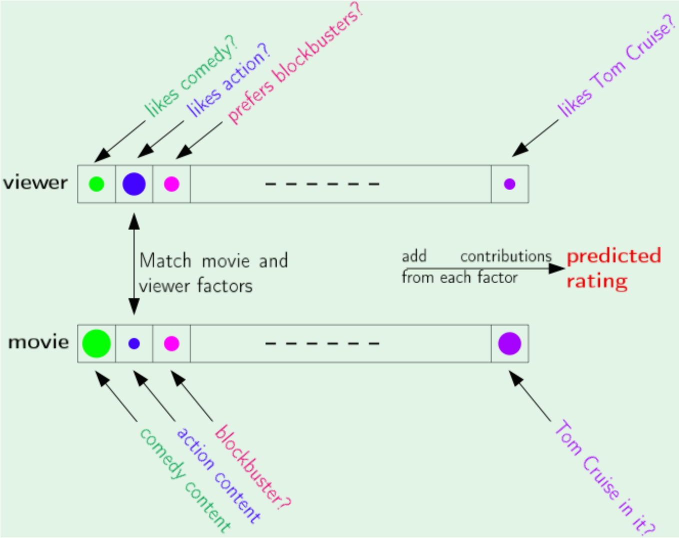
Netflix

10% improvement = 1 million dollar prize

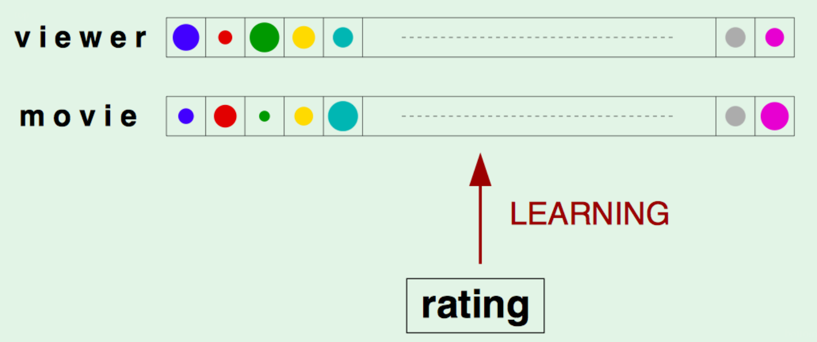
The essence of machine learning:

- A pattern exists
- We cannot pin it down mathematically
- We have data on it

A solution of movie rating



Machine learning approach



Components of Machine Learning

Metaphor: Financial fraud detection (*e.g.* swapping a credit card).

Components of Machine Learning

Metaphor: Financial fraud detection (*e.g.* swapping a credit card).

Card holder' info (age, gender, income, home address, years in residence, years in job, ...)

Components of Machine Learning

Metaphor: Financial fraud detection (*e.g.* swapping a credit card).

Card holder' info (age, gender, income, home address, years in residence, years in job, ...)

Transaction info (where, when, amount, type of purchase, ...)

Components of Machine Learning

Metaphor: Financial fraud detection (*e.g.* swapping a credit card).

Card holder' info (age, gender, income, home address, years in residence, years in job, ...)

Transaction info (where, when, amount, type of purchase, ...)

Decision: a genuine transaction or a fraud?

Components of Machine Learning

Formulation:

- Input: $\mathbf{x} \in \mathcal{X}$ (Card holder's info and transaction's info)[feature]
- Output: $y \in \mathcal{Y}$ (genuine or fraud?)[label]

Components of Machine Learning

Formulation:

- Input: $\mathbf{x} \in \mathcal{X}$ (Card holder's info and transaction's info)[feature]
- Output: $y \in \mathcal{Y}$ (genuine or fraud?)[label]
- Underlying process (unknown) $f : \mathcal{X} \to \mathcal{Y}$ (ideal fraud detection formula)

Components of Machine Learning

Formulation:

- Input: $\mathbf{x} \in \mathcal{X}$ (Card holder's info and transaction's info)[feature]
- Output: $y \in \mathcal{Y}$ (genuine or fraud?)[label]
- Underlying process (unknown) $f : \mathcal{X} \rightarrow \mathcal{Y}$ (ideal fraud detection formula)
- Data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ (historical records)

Components of Machine Learning

Formulation:

- Input: $\mathbf{x} \in \mathcal{X}$ (Card holder's info and transaction's info)[feature]
- Output: $y \in \mathcal{Y}$ (genuine or fraud?)[label]
- Underlying process (unknown) $f : \mathcal{X} \to \mathcal{Y}$ (ideal fraud detection formula)
- Data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^N = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ (historical records)
  $\Downarrow$ Learn
- Decision function $g : \mathcal{X} \to \mathcal{Y}$, such that $g \approx f$. (formula to be used)

Components of Machine Learning

Formulation:

- Input: $\mathbf{x} \in \mathcal{X}$ (Card holder's info and transaction's info)[feature]
- Output: $y \in \mathcal{Y}$ (genuine or fraud?)[label]
- Underlying process (unknown) $f : \mathcal{X} \to \mathcal{Y}$ (ideal fraud detection formula)
- Data: $\{(\mathbf{x}_i, y_i)\}_{i=1}^N = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ (historical records)
  $\Downarrow$ Learn
- Decision function $g : \mathcal{X} \to \mathcal{Y}$, such that $g \approx f$. (formula to be used)

For a new $\mathbf{x}'$, predict $y' = g(\mathbf{x}')$.

Summarise and question

Summarise Machine Learning in one line?

Summarise and question

Summarise Machine Learning in one line?

Machine Learning

Using data to uncover an unknown underlying process.

Summarise and question

Summarise Machine Learning in one line?

Machine Learning

Using data to uncover an unknown underlying process.

How does it response to the essence of machine learning?
Recall the essence of machine learning:

- A pattern exists
- We cannot pin it down mathematically
- We have data on it

Examples

$\mathcal{X}$ $\mathcal{Y}$

Examples

$$\begin{array}{ccc} \mathcal{X} & & \mathcal{Y} \\ \text{(age, education, occupation, ...)} & \rightarrow & \text{income} > \$50k \text{ p.a.?} \end{array}$$

Examples

$\mathcal{X}$		$\mathcal{Y}$
(age, education, occupation, ...)	$\rightarrow$	income $>$ \$50k p.a.?
	$\rightarrow$	$\{0, 1, ..., 9\}$

Examples

$\mathcal{X}$		$\mathcal{Y}$
(age, education, occupation, ...)	$\rightarrow$	income $>$ \$50k p.a.?
	$\rightarrow$	$\{0, 1, ..., 9\}$
	$\rightarrow$	$\{John, Jenny, ...\}$

Examples

$\mathcal{X}$		$\mathcal{Y}$
(age, education, occupation, ...)	$\rightarrow$	income $>$ \$50k p.a.?
	$\rightarrow$	$\{0, 1, ..., 9\}$
	$\rightarrow$	$\{John, Jenny, ...\}$

To learn decision function $g : \mathcal{X} \rightarrow \mathcal{Y}$. What's g like?

Examples

$\mathcal{X}$		$\mathcal{Y}$
(age, education, occupation, ...)	$\rightarrow$	income $>$ \$50k p.a.?
	$\rightarrow$	$\{0, 1, ..., 9\}$
	$\rightarrow$	$\{John, Jenny, ...\}$

To learn decision function $g : \mathcal{X} \rightarrow \mathcal{Y}$. What's g like?

To show g, we may have to introduce (or refresh) some concepts (vectors, inner products and sign function).

Refresh concepts

Inner product

For vectors $\mathbf{x} = [x^1, x^2, \cdots, x^d]^\top, \mathbf{w} = [w^1, w^2, \cdots, w^d]^\top$, inner product

$$\langle \mathbf{x}, \mathbf{w} \rangle = \sum_{i=1}^{d} x^i w^i.$$

We write $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d$ to say they are d-dimensional real number vectors. We consider all vectors as column vectors by default. $\top$ is the transpose. We also use the matlab syntax that $[x^1; x^2; \cdots; x^d]$ as column vector.

Refresh concepts

Inner product

For vectors $\mathbf{x} = [x^1, x^2, \cdots, x^d]^\top, \mathbf{w} = [w^1, w^2, \cdots, w^d]^\top$, inner product

$$\langle \mathbf{x}, \mathbf{w} \rangle = \sum_{i=1}^{d} x^i w^i.$$

We write $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d$ to say they are d-dimensional real number vectors. We consider all vectors as column vectors by default. $\top$ is the transpose. We also use the matlab syntax that $[x^1;x^2;\cdots;x^d]$ as column vector.

Example: $a = [1; 3; 1.5], b = [2; 1; 1]$. $\langle a, b \rangle =?$

Refresh concepts

Inner product

For vectors $\mathbf{x} = [x^1, x^2, \cdots, x^d]^\top, \mathbf{w} = [w^1, w^2, \cdots, w^d]^\top$, inner product

$$\langle \mathbf{x}, \mathbf{w} \rangle = \sum_{i=1}^{d} x^i w^i.$$

We write $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d$ to say they are d-dimensional real number vectors. We consider all vectors as column vectors by default. $\top$ is the transpose. We also use the matlab syntax that $[x^1; x^2; \cdots; x^d]$ as column vector.

Example: $a = [1; 3; 1.5], b = [2; 1; 1]$. $\langle a, b \rangle = ?$
$= 1 \times 2 + 3 \times 1 + 1.5 \times 1 = 6.5$

Refresh concepts

Sign function

For any scalar $a \in \mathbb{R}$,

$$\text{sign}(a) = \begin{cases} 1 & \text{if } a > 0 \\ -1 & \text{otherwise} \end{cases}$$

Refresh concepts

Sign function

For any scalar $a \in \mathbb{R}$,

$$\text{sign}(a) = \begin{cases} 1 & \text{if } a > 0 \\ -1 & \text{otherwise} \end{cases}$$

Examples: $\text{sign}(20) = ?$, $\text{sign}(-5) = ?$, $\text{sign}(0) = ?$.

Refresh concepts

Sign function

For any scalar $a \in \mathbb{R}$,

$$\text{sign}(a) = \begin{cases} 1 & \text{if } a > 0 \\ -1 & \text{otherwise} \end{cases}$$

Examples: $\text{sign}(20) = ?$, $\text{sign}(-5) = ?$, $\text{sign}(0) = ?$.
$\text{sign}(20) = 1$, $\text{sign}(-5) = -1$, $\text{sign}(0) = -1$.

Decision functions

Typical decision functions for classification [2] :

$$\text{Binary-class} \quad g(\mathbf{x};\mathbf{w}) = \operatorname{sign}(\langle \mathbf{x}, \mathbf{w} \rangle).$$

$$\text{Multi-class} \quad g(\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}}(\langle \mathbf{x}, \mathbf{w}_y \rangle).$$

where $\mathbf{w}, \mathbf{w}_y$ are the parameters, and $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d, \mathbf{w}_y \in \mathbb{R}^d$.

[2] for $b \in \mathbb{R}$, more general form $\langle \mathbf{x}, \mathbf{w} \rangle + b$ can be rewritten as $\langle [\mathbf{x}; 1], [\mathbf{w}; b] \rangle$

Decision functions

Typical decision functions for classification [2] :

$$\text{Binary-class} \quad g(\mathbf{x};\mathbf{w}) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle).$$

$$\text{Multi-class} \quad g(\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}}(\langle \mathbf{x}, \mathbf{w}_y \rangle).$$

where $\mathbf{w}, \mathbf{w}_y$ are the parameters, and $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d, \mathbf{w}_y \in \mathbb{R}^d$.
Example 1: $\mathbf{x} = [1; 3.5], \mathbf{w} = [2; -1]$. $g(\mathbf{x};\mathbf{w}) = ?$

[2] for $b \in \mathbb{R}$, more general form $\langle \mathbf{x}, \mathbf{w} \rangle + b$ can be rewritten as $\langle [\mathbf{x}; 1], [\mathbf{w}; b] \rangle$

Decision functions

Typical decision functions for classification [2] :

$$\text{Binary-class} \quad g(\mathbf{x};\mathbf{w}) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle).$$

$$\text{Multi-class} \quad g(\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}}(\langle \mathbf{x}, \mathbf{w}_y \rangle).$$

where $\mathbf{w}, \mathbf{w}_y$ are the parameters, and $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d, \mathbf{w}_y \in \mathbb{R}^d$.
Example 1: $\mathbf{x} = [1; 3.5], \mathbf{w} = [2; -1]$. $g(\mathbf{x};\mathbf{w}) =?$
$= \text{sign}(1 \times 2 + 3.5 \times (-1)) = \text{sign}(-1.5) = -1$.

[2] for $b \in \mathbb{R}$, more general form $\langle \mathbf{x}, \mathbf{w} \rangle + b$ can be rewritten as $\langle [\mathbf{x}; 1], [\mathbf{w}; b] \rangle$

Decision functions

Typical decision functions for classification [2] :

$$\text{Binary-class} \quad g(\mathbf{x}; \mathbf{w}) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle).$$

$$\text{Multi-class} \quad g(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}}(\langle \mathbf{x}, \mathbf{w}_y \rangle).$$

where $\mathbf{w}, \mathbf{w}_y$ are the parameters, and $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d, \mathbf{w}_y \in \mathbb{R}^d$.
Example 1: $\mathbf{x} = [1; 3.5], \mathbf{w} = [2; -1]$. $g(\mathbf{x}; \mathbf{w}) = ?$
$= \text{sign}(1 \times 2 + 3.5 \times (-1)) = \text{sign}(-1.5) = -1.$
Example 2:
$\mathbf{x} = [1; 3.5], \mathbf{w}_1 = [2; -1], \mathbf{w}_2 = [1; 2], \mathbf{w}_3 = [3; 2], y = 1, 2, 3.$
$g(\mathbf{x}; \mathbf{w}) = ?$

[2] for $b \in \mathbb{R}$, more general form $\langle \mathbf{x}, \mathbf{w} \rangle + b$ can be rewritten as $\langle [\mathbf{x}; 1], [\mathbf{w}; b] \rangle$

Decision functions

Typical decision functions for classification [2] :

$$\text{Binary-class} \quad g(\mathbf{x}; \mathbf{w}) = \operatorname{sign}(\langle \mathbf{x}, \mathbf{w} \rangle).$$

$$\text{Multi-class} \quad g(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}}(\langle \mathbf{x}, \mathbf{w}_y \rangle).$$

where $\mathbf{w}, \mathbf{w}_y$ are the parameters, and $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d, \mathbf{w}_y \in \mathbb{R}^d$.
Example 1: $\mathbf{x} = [1; 3.5], \mathbf{w} = [2; -1]$. $g(\mathbf{x}; \mathbf{w}) = ?$
$= \operatorname{sign}(1 \times 2 + 3.5 \times (-1)) = \operatorname{sign}(-1.5) = -1.$
Example 2:
$\mathbf{x} = [1; 3.5], \mathbf{w}_1 = [2; -1], \mathbf{w}_2 = [1; 2], \mathbf{w}_3 = [3; 2], y = 1, 2, 3.$
$g(\mathbf{x}; \mathbf{w}) = ?$ $\langle \mathbf{x}, \mathbf{w}_1 \rangle = -1.5, \langle \mathbf{x}, \mathbf{w}_2 \rangle = 8, \langle \mathbf{x}, \mathbf{w}_3 \rangle = 10$. Thus
$g(\mathbf{x}; \mathbf{w}) = \operatorname{argmax}_{y \in \{1,2,3\}} \langle \mathbf{x}, \mathbf{w}_y \rangle = 3.$

[2] for $b \in \mathbb{R}$, more general form $\langle \mathbf{x}, \mathbf{w} \rangle + b$ can be rewritten as $\langle [\mathbf{x}; 1], [\mathbf{w}; b] \rangle$

To learn Parameters

Typical decision functions for classification [3] :

$$\text{Binary-class} \quad g(\mathbf{x}; \mathbf{w}) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle).$$

$$\text{Multi-class} \quad g(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}}(\langle \mathbf{x}, \mathbf{w}_y \rangle).$$

where $\mathbf{w}, \mathbf{w}_y$ are the parameters, and $\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^d, \mathbf{w}_y \in \mathbb{R}^d$.

Parameter estimation

To learn g is to learn $\mathbf{w}$ or $\mathbf{w}_y$.

[3] for $b \in \mathbb{R}$, more general form $\langle \mathbf{x}, \mathbf{w} \rangle + b$ can be rewritten as $\langle [\mathbf{x}; 1], [\mathbf{w}; b] \rangle$

Break

Take a break ...

Types of Learning

1. Supervised Learning
2. Unsupervised Learning
3. Semi-supervised Learning

Supervised Learning

Definition

Given input-output data pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ sampled from an unknown but fixed distribution $p(\mathbf{x}, y)$, the goal is to learn $g : \mathcal{X} \to \mathcal{Y}$, $g \in \mathcal{G}$ s.t. $p(g(\mathbf{x}) \neq y)$ is small.

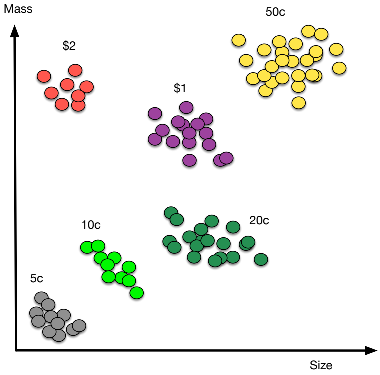
$p(g(\mathbf{x}) \neq y)$ (*i.e.* expected testing error) is generalisation error.

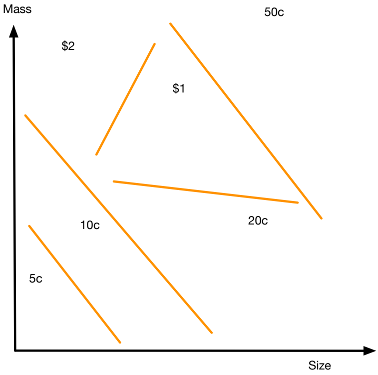
Supervised Learning

Coin recognition (vending machines and parking meters).

Supervised Learning

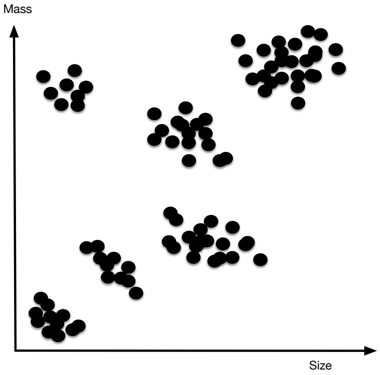
We have (input, correct output) in the training data.





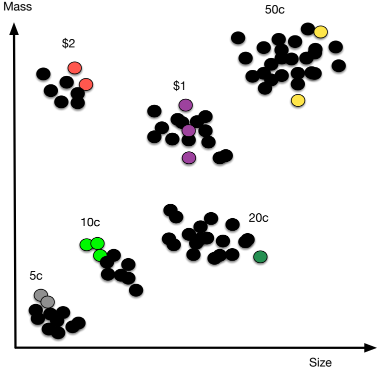
Unsupervised Learning

Instead of (input, correct output), we have (input, ?).



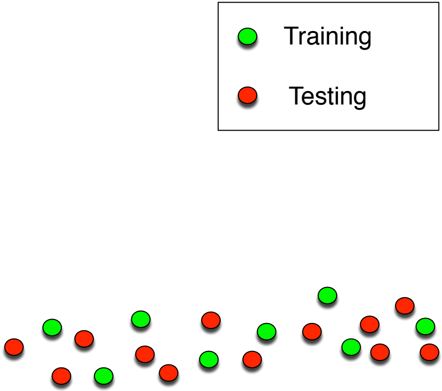
Semi-supervised Learning

We have some (input, correct output), and some (input, ?).



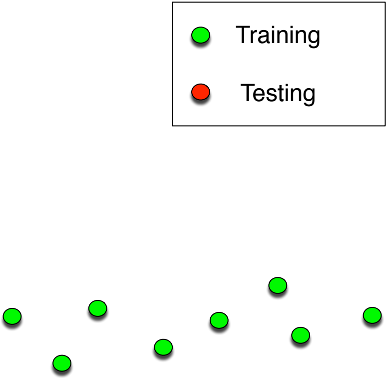
Overfitting

Fitting the training data too well cause a problem.



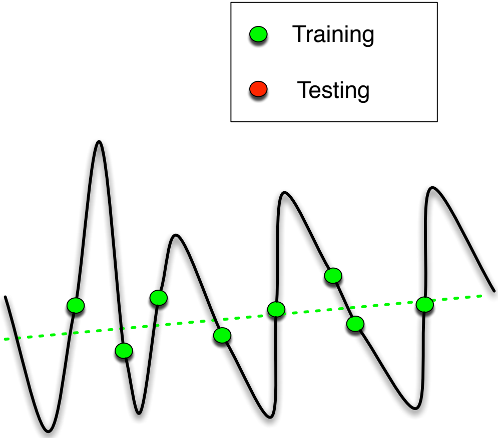
Overfitting

Train on training data (testing data are hidden from us).



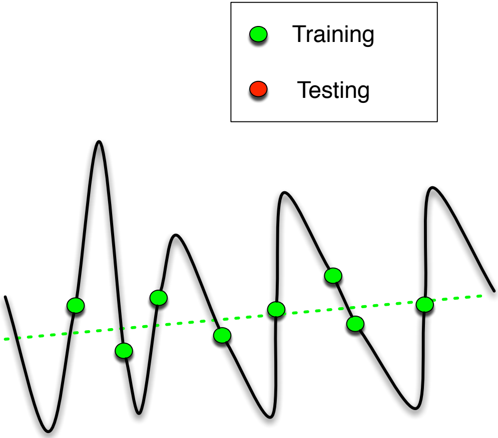
Overfitting

Two possible models. Which model fits the training data better?



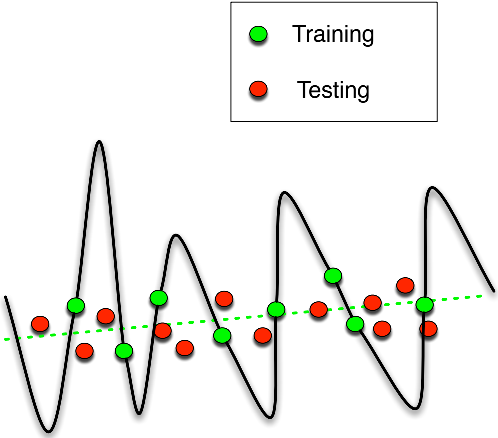
Overfitting

Two possible models. Which model fits the testing data better?



Overfitting

Reveal the testing data.



Occam's Razor

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

Occam's Razor

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

Two questions:

1. What does it mean for a model to be simple?

Occam's Razor

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

Two questions:

1. What does it mean for a model to be simple?
2. Why simpler is better?

Simpler means less complex

Model complexity – two types:

1. complexity of the function g: order of a polynomial, MDL
   - a straight line (order 0 or 1) is simpler than a quadratic function (order 2).
   - computer program: 100 bits simpler than 1000 bits
2. complexity of the space $\mathcal{G}$: $|\,\mathcal{G}\,|$, VC dimension, noise-fitting, ...
   - Often used in proofs.

Simpler is better

1. What do you mean by "better"?
   - smaller generalisation error (*e.g.* smaller expected testing error).
2. Why simpler is better?
   - Practically implemented by regularisation techniques, which will be covered in Lecture 2.
   - Theoretically answered by generalisation bounds, which will be covered in Learning Theory in Lecture 12.

Typical assumptions

1. Small-scale data
   - Model fits in the memory
   - Data fit in the memory or at least the disk
   - Computer is fast enough
2. $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ are independent and identically distributed (i.i.d.) samples from $p(\mathbf{x}, y)$
3. Underlying process ($f(\mathbf{x})$ or $p(\mathbf{x}, y)$) unknown but fixed

In real life things are more complex

1. ~~Small-scale~~ data
   - Large-scale $\rightarrow$ Random Projection

In real life things are more complex

1. ~~Small-scale~~ data
   - Large-scale $\rightarrow$ Random Projection
2. $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ are ~~independent~~ and ~~identically~~ distributed (i.i.d.) samples from $p(\mathbf{x}, y)$
   - Correlated $\rightarrow$ Structured Learning and Graphical Models

In real life things are more complex

1. ~~Small-scale~~ data
   - Large-scale → Random Projection
2. $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ are ~~independent~~ and ~~identically~~ distributed (i.i.d.) samples from $p(\mathbf{x}, y)$
   - Correlated → Structured Learning and Graphical Models
3. Underlying process unknown but ~~fixed~~
   - Changing environment → Online Learning (with Structured Data)

Large-scale data

Assumption 1: ~~Small-scale~~ data.

- Web topic classification: 4.4 million data, input vector 1.8 million dimensions, and output $7k$ classes?
- $\operatorname{argmax}_{y \in \mathcal{Y}}(\langle \mathbf{x}, \mathbf{w}_y \rangle)$? No! "store all $\mathbf{w}_y$" $\approx 100G$ memory.

Large-scale data

Assumption 1: ~~Small-scale~~ data.

- Web topic classification: 4.4 million data, input vector 1.8 million dimensions, and output $7k$ classes?
- $\operatorname{argmax}_{y \in \mathcal{Y}}(\langle \mathbf{x}, \mathbf{w}_y \rangle)$? No! "store all $\mathbf{w}_y$" $\approx 100G$ memory.
- Our methods:
  - Loading data, training and testing on 804,414 news articles to predict the topics in 25.16s!
  - Training 4.4 million data in 0.5 hours (normally 2000 days).

Structured data

Assumption 2: $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ are ~~independent~~.

Structured data

Assumption 2: $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ are ~~independent~~.

Figure : Tennis action recognition

Most likely actions $= \mathrm{argmax}_{y_1,y_2,y_3,y_4} P(y_1, y_2, y_3, y_4 | Image)$.

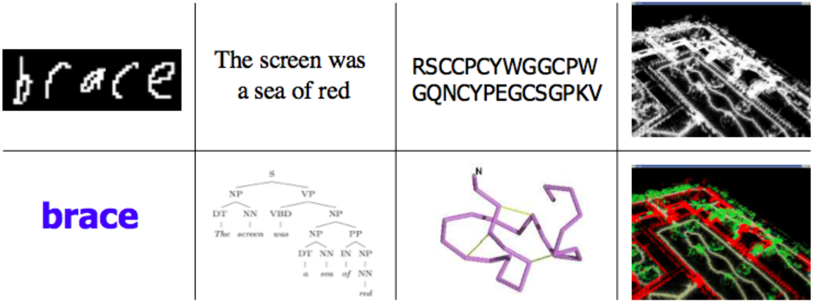
Structured data

Assumption 2: $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ are ~~independent~~.

Figure : Tennis action recognition

Most likely actions $= \mathrm{argmax}_{y_1,y_2,y_3,y_4} P(y_1, y_2, y_3, y_4 | Image)$.
$\mathbf{y} = (y_1, y_2, y_3, y_4)$ is a structure of an array.

Structured data



[4]

Structured output: a sequence, a tree, or a network, ...

[4] courtesy of B. Taskar

Online Learning

Assumption 3 fails: Underlying process changes. +
Assumption 1 fails too. *i.e.* We have Large-scale data.

Online Learning

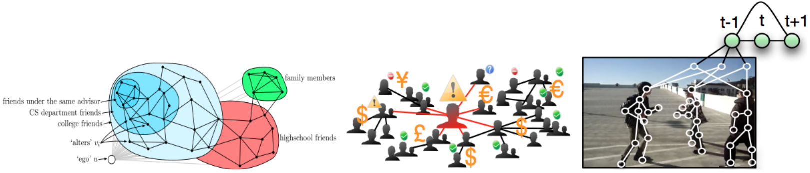
Assumption 3 fails: Underlying process changes. $+$
Assumption 1 fails too. *i.e.* We have Large-scale data.
$\Downarrow$
Online Learning (OL): predicting answers for a sequence of questions.

- processing one datum at a time (theoretical guarantee)
- no assumption on underlying process being fixed

Online Learning

Assumption 3 fails: Underlying process changes. +
Assumption 1 fails too. *i.e.* We have Large-scale data.
$\Downarrow$
Online Learning (OL): predicting answers for a sequence of questions.

- processing one datum at a time (theoretical guarantee)
- no assumption on underlying process being fixed
  Problem 1: it does not scale for structured data.



(a) Social networks (b) Fraud detection (c) Activity recognition

That's all

Thanks!