CM50264: Machine Learning 1

Decision trees and random forests

Kwang In Kim
k.kim@bath.ac.uk

Slide acknowledgements:
Neill Campbell, Richard Turner, and Manfred Law

Example machine learning task: Guessing subsequent numbers

1, 3, 5, 7, ? …

Three steps of learning and inference

1. Observe phenomena.
2. Build a model of phenomena.
3. Make a prediction.

This is more or less the definition of natural science [Olivier Bousquet].
The goal of machine learning is to automate this process.

Learning and inference

1, 3, 5, 7, ? …

It might be impossible to estimate a true model from finite observations (data).

It could be still possible to estimate a useful model.

Different flavours of machine learning

- Supervised/unsupervised
  - Supervised algorithms
    - learn from labelled examples:
      pairs of input and the corresponding desired output.
  - Unsupervised algorithms
    - learn from unlabelled examples:
      input data only; no desired outputs.
    - E.g., clustering and dimensionality reduction.
- Classification/regression
  - Classification: output variables take categorical (discrete) class index (e.g., {1, 0}, {Cat, Dog}).
  - Regression: output variables take continuous values (e.g., {0.75, -10.25, …}).

Classification problem

- We are given a set of training data points:

$$D = \{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\} \subset \boldsymbol{X} \times \{0,1\}.$$

- Our goal is to find a rule (e.g., function)

$$f : \boldsymbol{X} \to \{0,1\}$$

such that its output $f(\mathbf{x}^*)$ for an unseen input $\mathbf{x}^* \notin D$ is close to the underlying ground-truth class y^*.

Classification error

- Classification loss (error) per instance:

$$l(f(x), y) = \begin{cases} 0, \text{if } f(x) = y \\ 1, \text{otherwise.} \end{cases}$$

- Overall, classification error:

$$\int l(f(x), y) dx \,.$$

- Mean classification error:

$$\int l\big(f(x), y(x)\big) dP(x, y) \,.$$

Classification examples

$$f: \boldsymbol{X} \rightarrow \{0,1\}$$

- Elements of $\mathbf{x} \in \boldsymbol{X}$ can be continuous or discrete (categorical).
- Outputs are categorical.

- **x**: credit score, own or rent, age, …
  y: load defaults (yes/no).
- **x**: Length, word frequency histogram, …
  y: Email classification (spam/no spam).
- **x**: pixel values of a face image, …
  y: gender.

Image-based gender classification

- Training

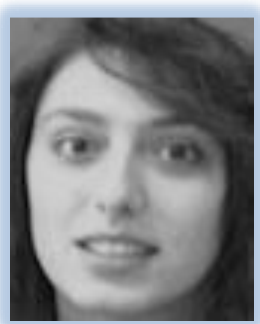
$\mathbf{I}^1$ $\mathbf{I}^2$ $\mathbf{I}^3$ $\mathbf{I}^4$

y^1=1 y^2=0 y^3=0 y^4=1

1: female
0: male

- Testing

$\mathbf{I}^*$=



y^*=?

Image-based gender classification

- Training

I: image

x: representation (or *features*) of **I**

$\mathbf{I}^1=$

$\Rightarrow \mathbf{x}^1 = [x^1{}_1, x^1{}_2, x^1{}_3, \ldots, x^1{}_{n\text{-}1}, x^1{}_n] \in \mathbf{R}^n$

$y^1=1$

- Testing

$\mathbf{I}^*=$

$\Rightarrow \mathbf{x}^* = [x^*{}_1, x^*{}_2, x^*{}_3, \ldots, x^*{}_{n\text{-}1}, x^*{}_n] \in \mathbf{R}^n$

$y^*=?$

Regression

- We are given a set of training data points:

$$D = \{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\} \subset \boldsymbol{X} \times \mathbf{R}.$$

- Our goal is to find a function

$$f: \boldsymbol{X} \to \mathbf{R}$$

such that its output $f(\mathbf{x}^*)$ for an unseen input $\mathbf{x}^* \notin D$ is close to the underlying ground-truth label y^*.

Regression error

- L2-loss: $l(f(x), y) = (f(x) - y)^2$.
- L1-loss: $l(f(x), y) = |f(x) - y|$.
- Mean squared error:

$$\int (f(x) - y)^2 dP(x, y).$$

- Mean absolute deviation:

$$\int |f(x) - y| dP(x, y).$$

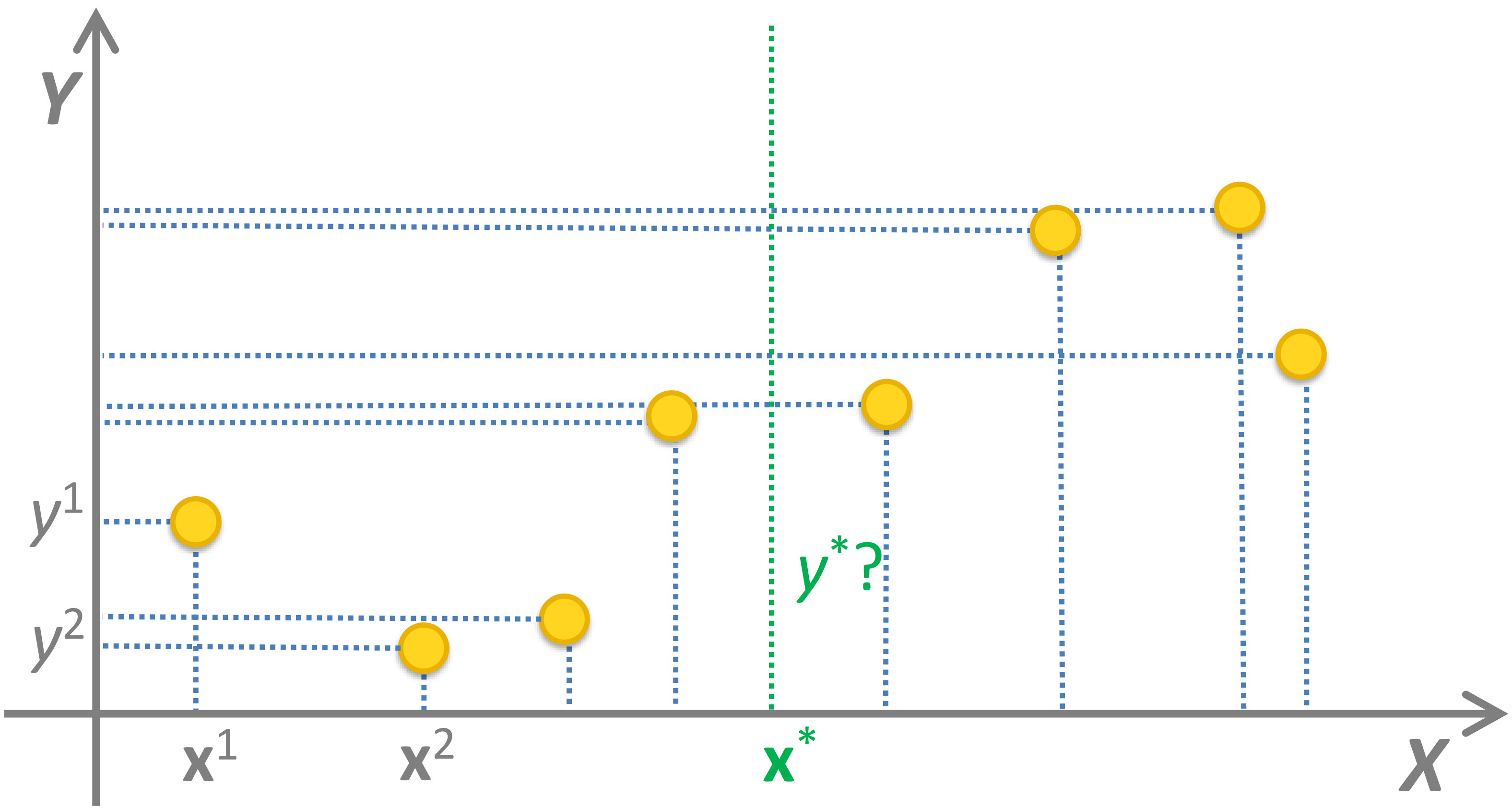
Regression examples

$$f: \boldsymbol{X} \rightarrow \mathbf{R}$$

- Elements of $\mathbf{x} \in \boldsymbol{X}$ can be continuous or discrete (categorical).
- Outputs are continuous.

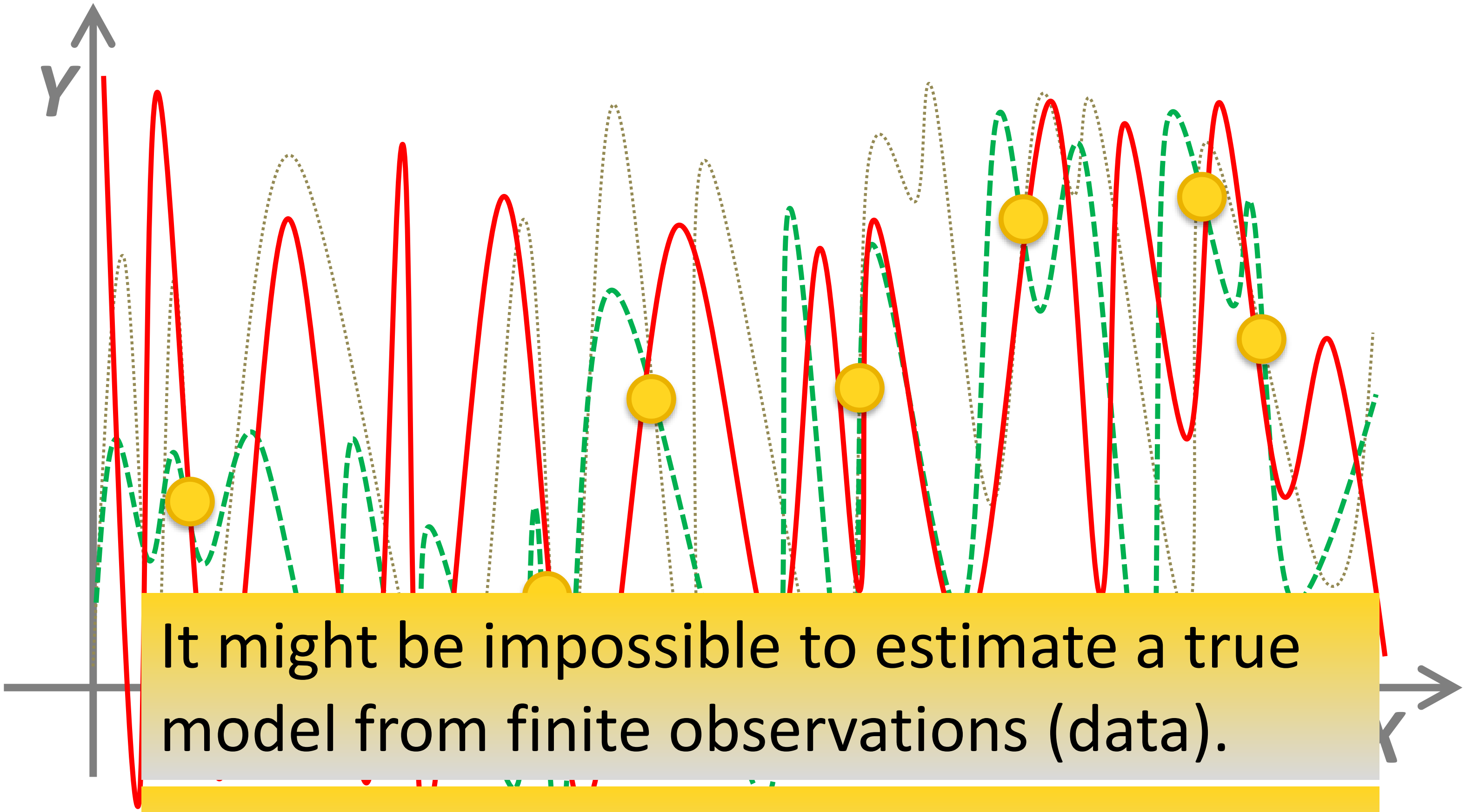
- **x**: age, education, occupation, …
  y: income.
- **x**: size, age, location, …
  y: selling price of houses.
- **x**: pixel values of a face image, …
  y: age.

One-dimensional regression



Task: from a set of pairs of input $\mathbf{x}^i$ and the corresponding output y^i, to estimate the output value y^* for the new input $\mathbf{x}^*$.

One-dimensional regression



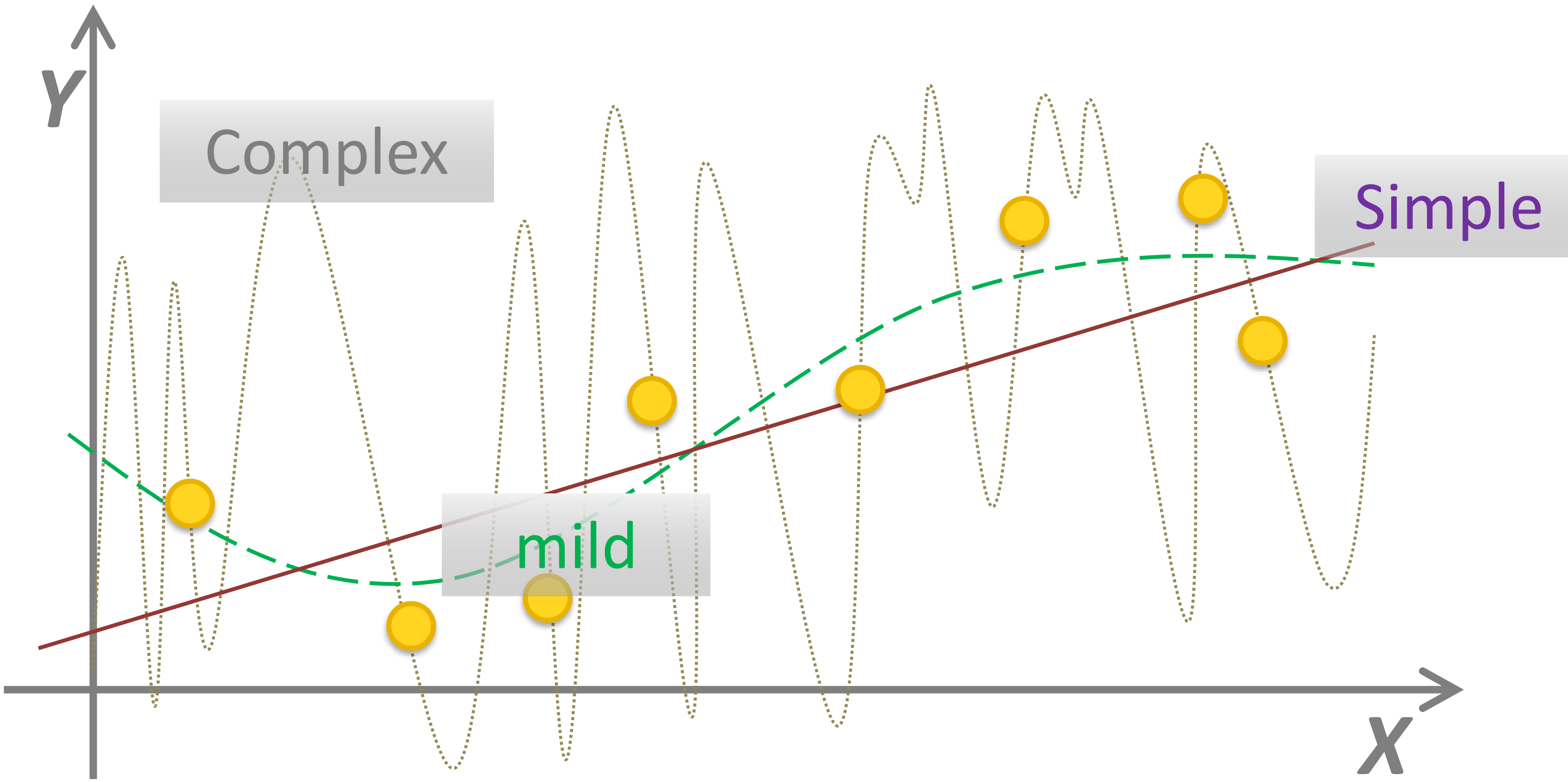
It might be impossible to estimate a true model from finite observations (data).

It could be still possible to estimate a useful model.

Basic idea of machine learning: Keep things simple.

- Aristotle: The best demonstration is the one using the least number of hypotheses.
- Mach: Simple is more economical in terms of number of experiments needed to confirm.
- Occam’s Razor: Entities should not be multiplied beyond necessity.
- Popper: Falsifiability, more empirical content means easier to falsify (require less experiments).
- Einstein: Everything should be as simple as it is but not simpler.

One-dimensional regression



Classification and regression trees

- L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth, 1984.
- L. Breiman, Bagging predictors, *Machine Learning*, 24, 123-140, 1996.
- L. Breiman, Random Forests, *Machine Learning*, 45, 5-32, 2001

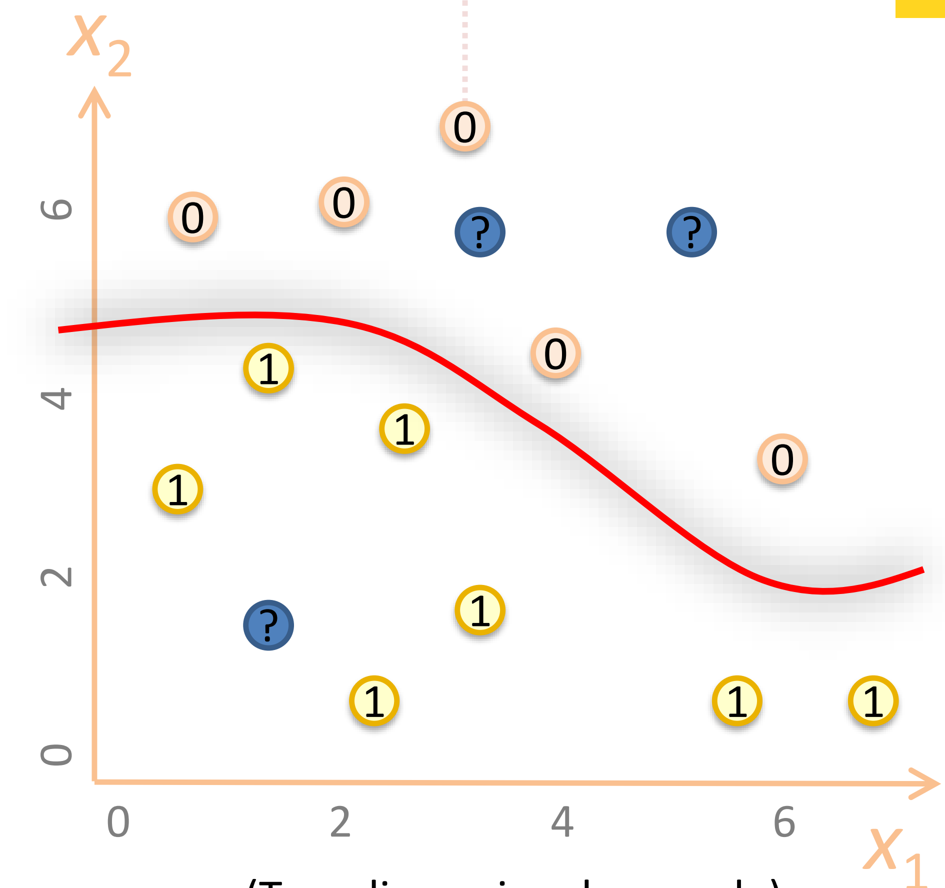
Leo Breiman
[27 Jan. 1928–5 Jul. 2005]

Classification and regression trees

- Training: build a tree given dataset *D*
  - Grow a (binary) tree.
  - At each node, *split* the data into *children* nodes.
  - Splits are chosen using a *splitting criterion*.
  - Bottom nodes are *leaf* nodes.
- Testing: make a prediction for an input $\mathbf{x}^*$
  - Traverse tree from root to a leaf based on splitting criteria.
  - For classification, the predicted class is the most common class in the node (majority vote).
  - For regression, the predicted value is the average outputs for all training data in the node.

Classification = space partitioning

$\mathbf{x}^1 = [x_1^1, x_2^1] \in \mathbf{R}^2$



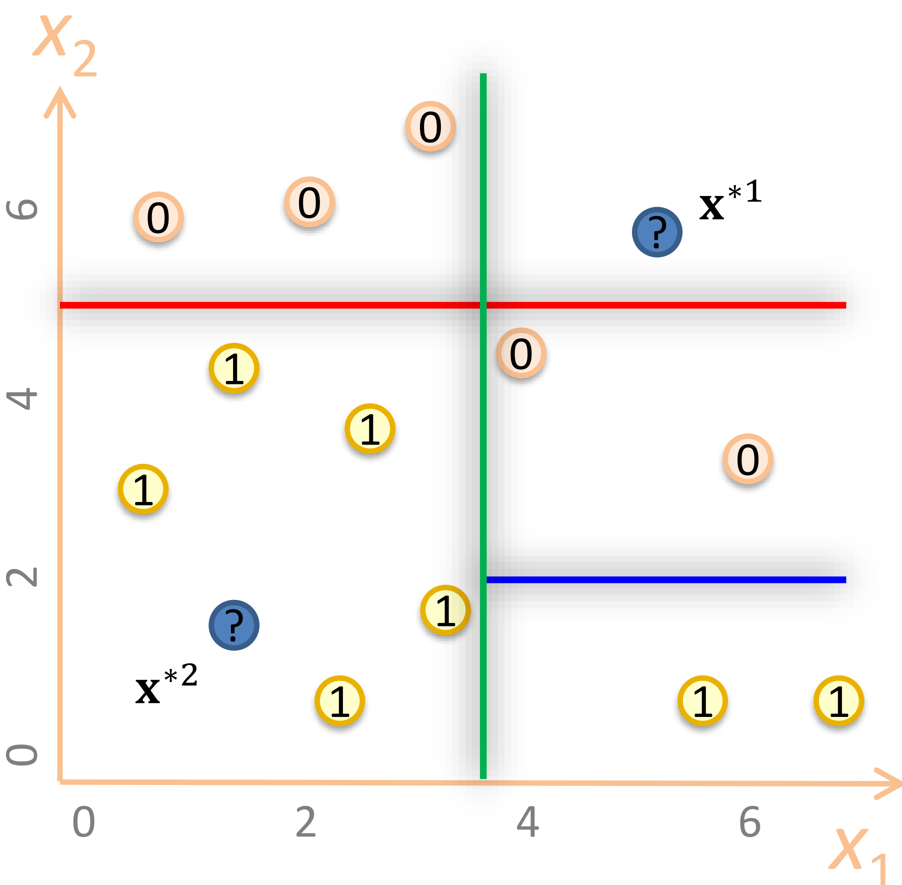
(Two-dimensional example)

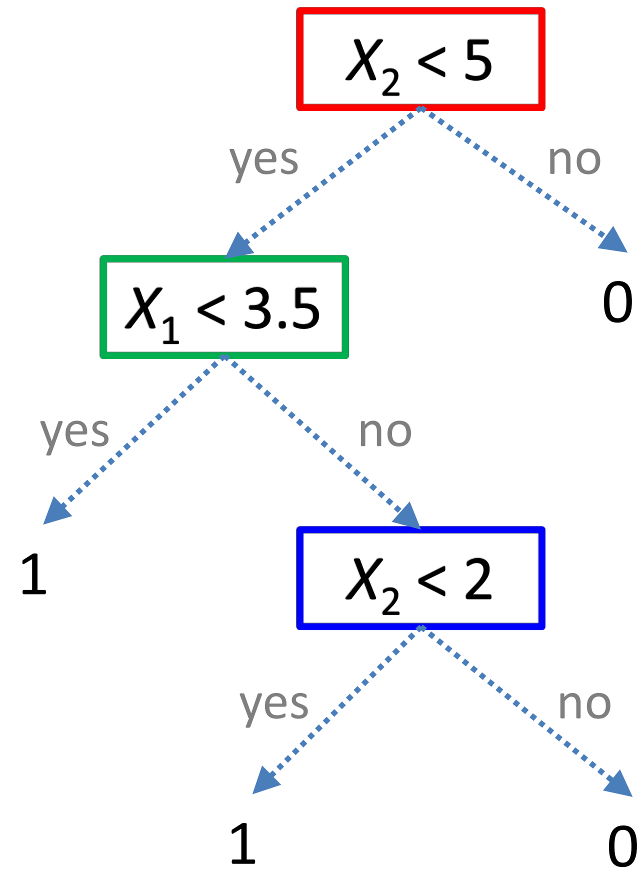
- Input: continuous or discrete *features*: x_1 and x_2.
- Output: categorical decisions (0 or 1).
- Based on training data, we want to predict class labels for every possible inputs

$\Rightarrow$ partition the space.

Decision tree example

Note: there’s no need to achieve zero training error.





Another example

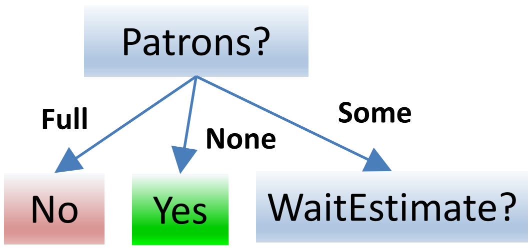
Our training data: Discrete + continuous input features, binary outputs.

Example	Attributes										Goal
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	*WillWait*
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	
$\mathbf{x}^1$	*Yes*	*No*	*No*	*Yes*	*Some*	£££	*No*	*Yes*	*French*	0-10	*Yes*
$\mathbf{x}^2$	*Yes*	*No*	*No*	*Yes*	*Full*	£	*No*	*No*	*Thai*	30-60	*No*
$\mathbf{x}^3$	*No*	*Yes*	*No*	*No*	*Some*	£	*No*	*No*	*Burger*	0-10	*Yes*
$\mathbf{x}^4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	£	*No*	*No*	*Thai*	10-30	*Yes*
$\mathbf{X}^5$	*Yes*	*No*	*Yes*	*No*	*Full*	£££	*No*	*Yes*	*French*	>60	*No*
$\mathbf{x}^6$	*No*	*Yes*	*No*	*Yes*	*Some*	££	*Yes*	*Yes*	*Italian*	0-10	*Yes*
$\mathbf{x}^7$	*No*	*Yes*	*No*	*No*	*None*	£	*Yes*	*No*	*Burger*	0-10	*No*
$\mathbf{x}^8$	*No*	*No*	*No*	*Yes*	*Some*	££	*Yes*	*Yes*	*Thai*	0-10	*Yes*
$\mathbf{x}^9$	*No*	*Yes*	*Yes*	*No*	*Full*	£	*Yes*	*No*	*Burger*	>60	*No*
$\mathbf{x}^{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	£££	*No*	*Yes*	*Italian*	10-30	*No*
$\mathbf{x}^{11}$	*No*	*No*	*No*	*No*	*None*	£	*No*	*No*	*Thai*	0-10	*No*
$\mathbf{x}^{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	£	*No*	*No*	*Burger*	30-60	*Yes*

y^1

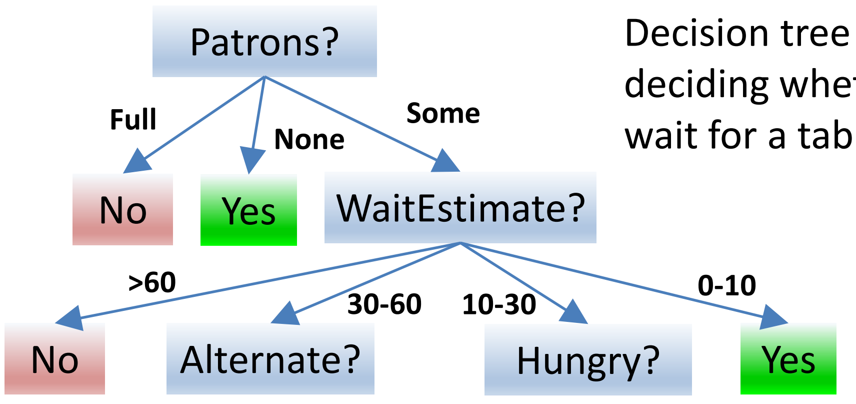
Goal predicate (classification)

Decision tree (one possibility)



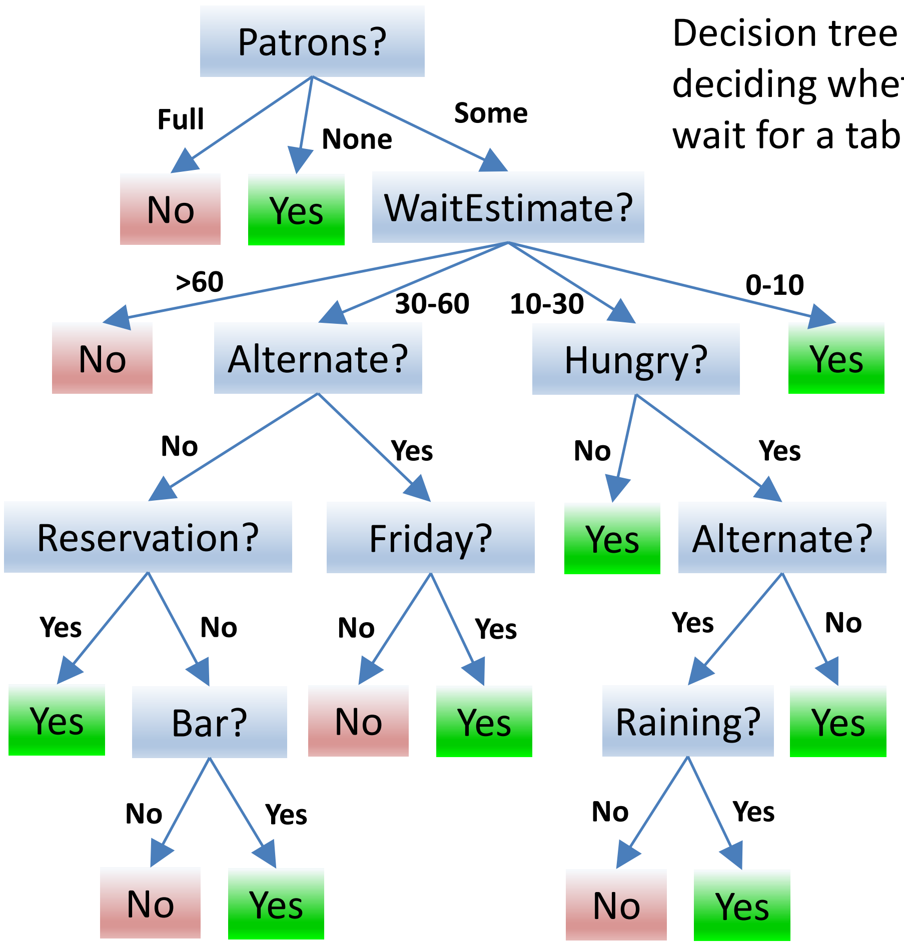
Decision tree for deciding whether to wait for a table

Decision tree (one possibility)



Decision tree for deciding whether to wait for a table

Decision tree (one possibility)

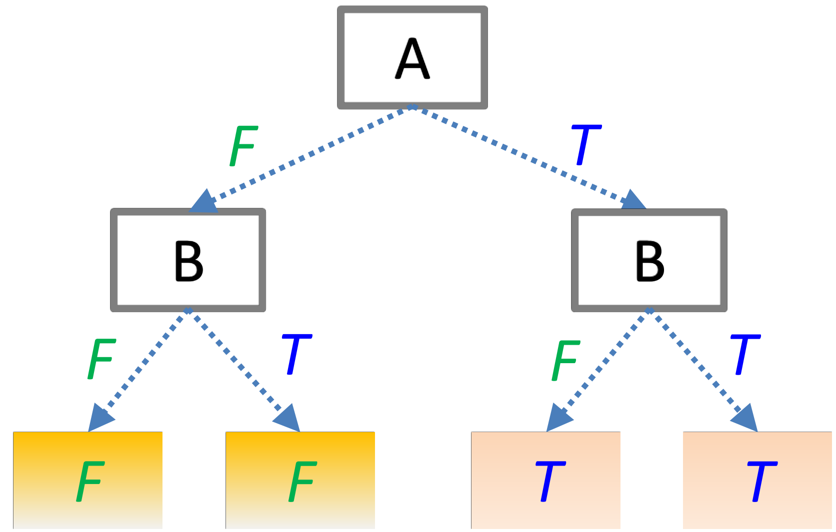


Decision tree for deciding whether to wait for a table

- Represents a Boolean function (goal predicate: WillWaitForTable()).
- Leaf nodes are Boolean values: final decisions (classifications).
- Internal nodes (*split nodes*) are tests of an attribute or feature.

Truth tables and decision trees

A	B	A xor B
F	*F*	*F*
F	*T*	*T*
T	*F*	*T*
T	*T*	*F*



- A truth table row = path in tree from root to leaf.
- Trivially, there is a consistent decision tree for any data set with one path to leaf for each example (zero-training error)
  - But most likely won't generalise to new examples.
- And such a tree is much larger than necessary, so we prefer to find more compact decision trees.

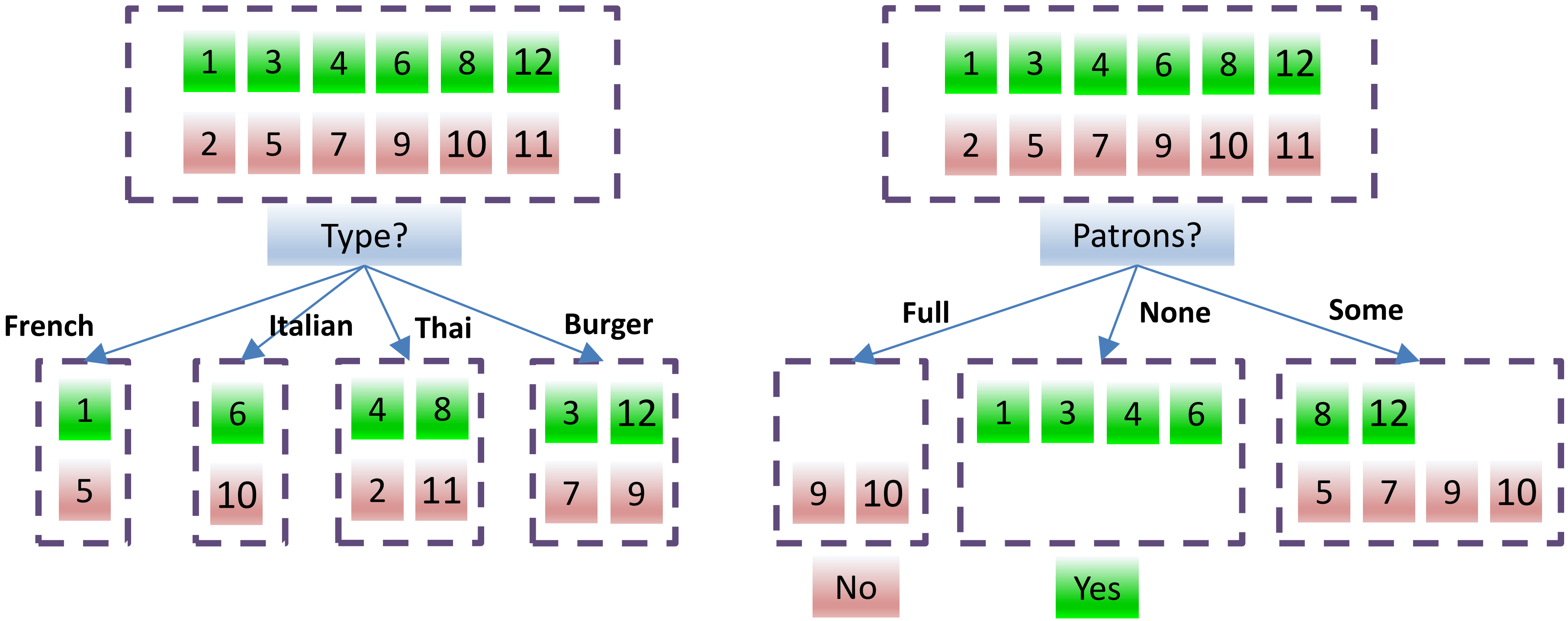
Inducing decision trees from examples

Example	Attributes										Goal
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	*WillWait*
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	
$\mathbf{x}^1$	*Yes*	*No*	*No*	*Yes*	*Some*	£££	*No*	*Yes*	*French*	0-10	*Yes*
$\mathbf{x}^2$	*Yes*	*No*	*No*	*Yes*	*Full*	£	*No*	*No*	*Thai*	30-60	*No*
$\mathbf{x}^3$	*No*	*Yes*	*No*	*No*	*Some*	£	*No*	*No*	*Burger*	0-10	*Yes*
$\mathbf{x}^4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	£	*No*	*No*	*Thai*	10-30	*Yes*
$\mathbf{X}^5$	*Yes*	*No*	*Yes*	*No*	*Full*	£££	*No*	*Yes*	*French*	>60	*No*
$\mathbf{x}^6$	*No*	*Yes*	*No*	*Yes*	*Some*	££	*Yes*	*Yes*	*Italian*	0-10	*Yes*
$\mathbf{x}^7$	*No*	*Yes*	*No*	*No*	*None*	£	*Yes*	*No*	*Burger*	0-10	*No*

Goal predicate (classification)

- Use the set of examples as a *training data* to create a decision tree.
- A trivial solution:
  - Build a tree that has one path for each example.
- *Occam's Razor* (or regularization theory):
  - The most likely hypothesis is the simplest one that is consistent with the data.

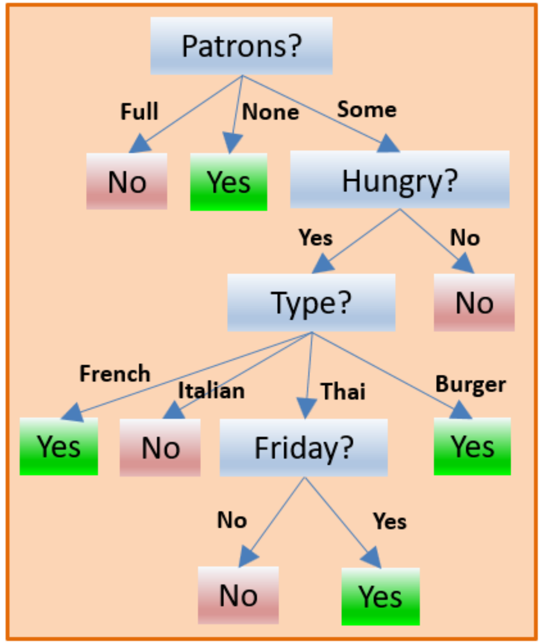
Simple tree means small tree



Guidelines for finding a small decision tree:

- Test the most important feature first.
- If you have only one type of example, return a leaf.
- Otherwise, choose the next most important feature.

Decision tree learning

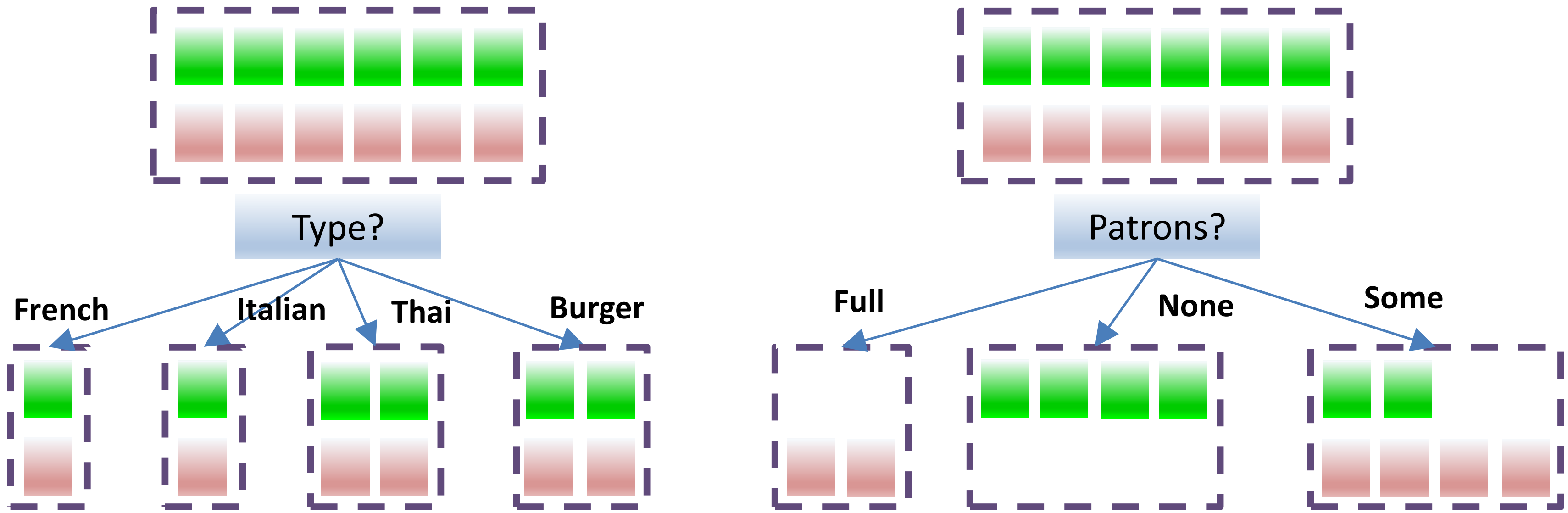


- Following this method, we generate the tree on the left.
- But the examples were generated by the original tree on the right.
- There is nothing wrong with the learning method: The method generates a hypothesis that matches the examples, not necessarily the underlying function
  - May be much simpler.
  - May uncover unexpected regularities.

How to decide which attribute to split on?

Idea: a good attribute should reduce *uncertainty*.

- E.g., splits the examples into subsets that are (ideally) `all positive’ or `all negative’



For [**Type?**], *to wait or not* is still at 50%.

[**Patrons?**] is a better choice.

Information

x: rain tomorrow?

$P(x\text{=YES})$	$P(x\text{=NO})$
0.9	0.1

- A rare event carries more information.
  - Information of \`x=YES' ~ $1/P(x\text{=YES})$.
  - Information of \`x=NO' ~ $1/P(x\text{=NO})$.
- Formally, *information* of each event is defined as
  - Information of \`x=YES' = $\log_2(1/P(x\text{=YES})) = -\log_2(P(x\text{=YES}))$.
  - Information of \`x=NO' = $\log_2(1/P(x\text{=NO})) = -\log_2(P(x\text{=NO}))$.

Quantifying the uncertainty: *Entropy*

x: rain tomorrow?

$P(x\text{=YES})$	$P(x\text{=NO})$
0.9	0.1

- Information of events
  - Information of \`x=YES' = $\log_2(1/P(x\text{=YES})) = -\log_2(P(x\text{=YES}))$.
  - Information of \`x=NO' = $\log_2(1/P(x\text{=NO})) = -\log_2(P(x\text{=NO}))$.
- *Entropy* $H(x)$ of a variable (or a question) x, is defined as the average information of all events:
  $H(x)$
  $= P(x\text{=YES}) \cdot -\log_2(P(x\text{=YES})) + P(x\text{=NO}) \cdot -\log_2(P(x\text{=NO}))$
  $= -P(x\text{=YES}) \log_2(P(x\text{=YES})) - P(x\text{=NO}) \log_2(P(x\text{=NO}))$
  $= -0.9 \log_2(0.9) - 0.1 \log_2(0.1) = 0.47$ bit.

Entropy

z: blood type?

$P(z\text{=O})$	$P(z\text{=A})$	$P(z\text{=B})$	$P(z\text{=AB})$
0.44	0.42	0.10	0.04

Entropy of *z:* average information of all events:

$H(z)$

$$
\begin{aligned}
=&-P(z\text{=O}) \log_2(P(z\text{=O})) \\
&-P(z\text{=A}) \log_2(P(z\text{=A})) \\
&-P(z\text{=B}) \log_2(P(z\text{=B})) \\
&-P(z\text{=AB}) \log_2(P(z\text{=AB}))
\end{aligned}
$$

$= -0.44 \log_2(0.44) -0.42 \log_2(0.42) -0.10 \log_2(0.10) -0.04 \log_2(0.04)$

$= 1.56$ bit.

Entropy example

x: rain tomorrow?

$P(x\text{=YES})$	$P(x\text{=NO})$
0.9	0.1

- $H(x)$
  $= - P(x\text{=YES}) \log_2(P(x\text{=YES})) - P(x\text{=YES}) \log_2(P(x\text{=NO}))$
  $= - 0.9 \log_2(0.9) - 0.1 \log_2(0.1) = 0.47$ bit.

x: rain tomorrow?

$P(x\text{=YES})$	$P(x\text{=NO})$
0.5	0.5

- $H(x)$
  $= - P(x\text{=YES}) \log_2(P(x\text{=YES})) - P(x\text{=YES}) \log_2(P(x\text{=NO}))$
  $= - 0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1$ bit.

Entropy example

x: rain tomorrow?

$P(x\text{=YES})$	$P(x\text{=NO})$
A	1-A

A=0.5:

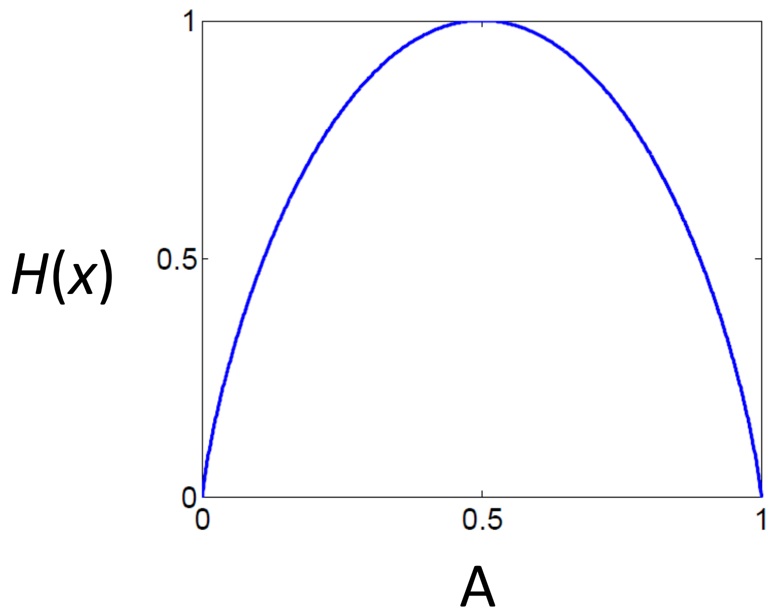
$P(x\text{=YES})$	$P(x\text{=NO})$
0.5	0.5

$H(x)$: 1 bit.

A=1:

$P(x\text{=YES})$	$P(x\text{=NO})$
1	0

$H(x)$: 0 bit.



Entropy example

z: blood type?

$P(z\text{=O})$	$P(z\text{=A})$	$P(z\text{=B})$	$P(z\text{=AB})$
1	0	0	0

- $H(z)$
  $= - 1 \log_2(1) - 0 \log_2(0) - 0 \log_2(0) - 0 \log_2(0) = 0$ bit.

z: blood type?

$P(z\text{=O})$	$P(z\text{=A})$	$P(z\text{=B})$	$P(z\text{=AB})$
1/4	1/4	1/4	1/4

- $H(z)$
  $= - 0.25 \log_2(0.25) - 0.25 \log_2(0.25) - 0.25 \log_2(0.25) - 0.25 \log_2(0.25) = 2$ bits.

Entropy = Uncertainty

- Entropy of a question (or random variable) *x* with possible answers $\{A^1, \ldots, A^n\}$:

$$H(x) = \sum_{i=1}^{n} -P(\mathrm{A}^i)\log_2 P(\mathrm{A}^i).$$

- Entropy of *x* measures how even the probability distribution of *x* is.

$P(x=\text{YES})$	$P(x=\text{NO})$
0.5	0.5

$H(x)$: 1 bit.

$P(x=\text{YES})$	$P(x=\text{NO})$
1	0

$H(x)$: 0 bit.

Entropy = Uncertainty

- Entropy measures the amount of uncertainty in a probability distribution
  - Perfectly even distribution: completely uncertain about the possible outcomes.
  - Uneven distribution: less uncertain.
  - Perfectly uneven (deterministic) distribution: completely certain about the possible outcomes.
- Entropy of *x* measures how even the probability distribution of *x* is.

$P(x=\text{YES})$	$P(x=\text{NO})$
0.5	0.5

$H(x)$: 1 bit.

$P(x=\text{YES})$	$P(x=\text{NO})$
1	0

$H(x)$: 0 bit.

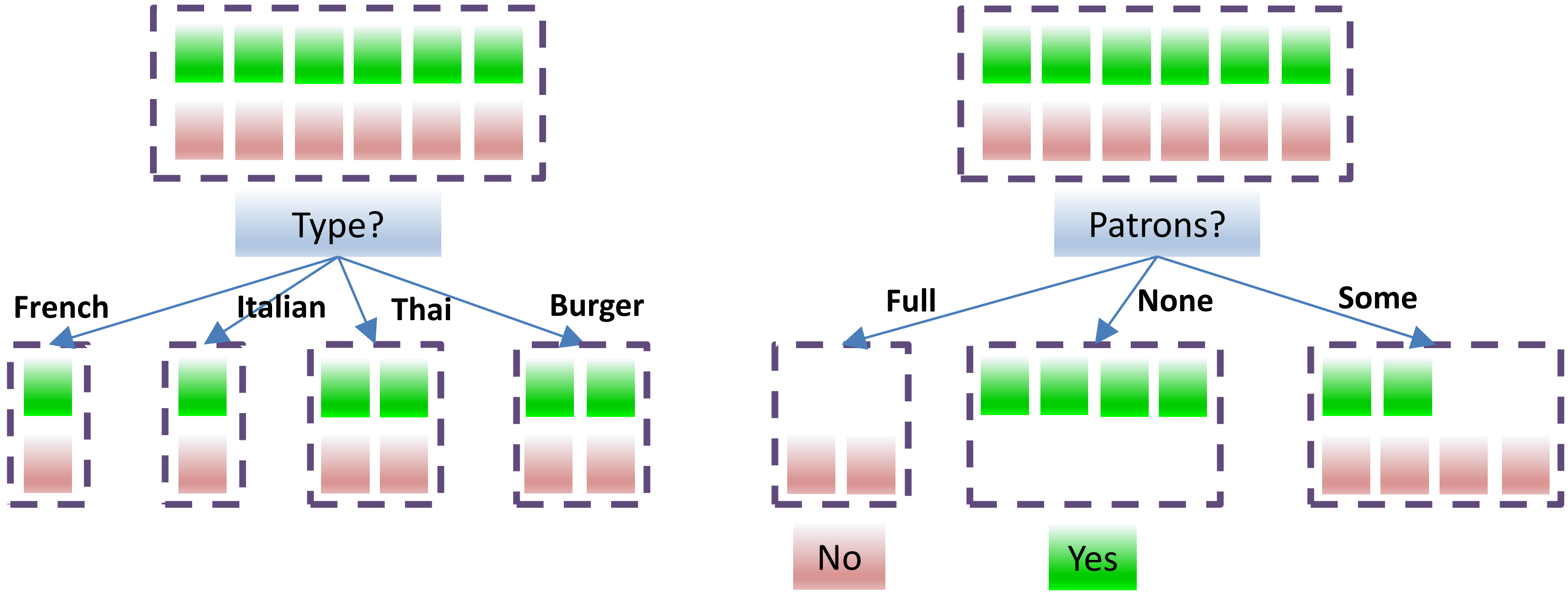
Choosing an attribute to split on

- Idea: a good attribute should reduce uncertainty and result in *gain in information*.
- How much information do we gain if we disclose the value of some attribute?
- Answer:

  uncertainty before – uncertainty after

  ⇔ entropy before – entropy after.

Back at the restaurant



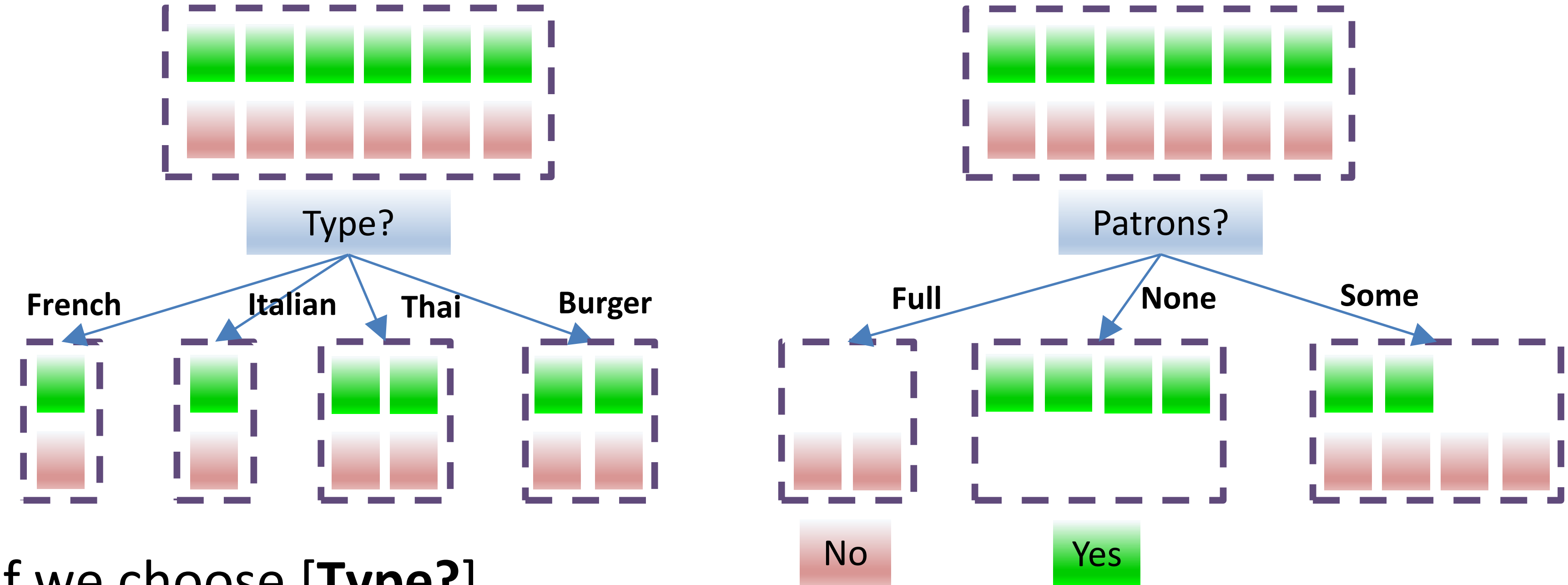
x: wait or not?

Before choosing an attribute (6 YES and 6 NO):

$H(x) = -6/12 \log_2(6/12) - 6/12 \log_2 (6/12) = 1$ bit.

There is `1 bit of uncertainty'.

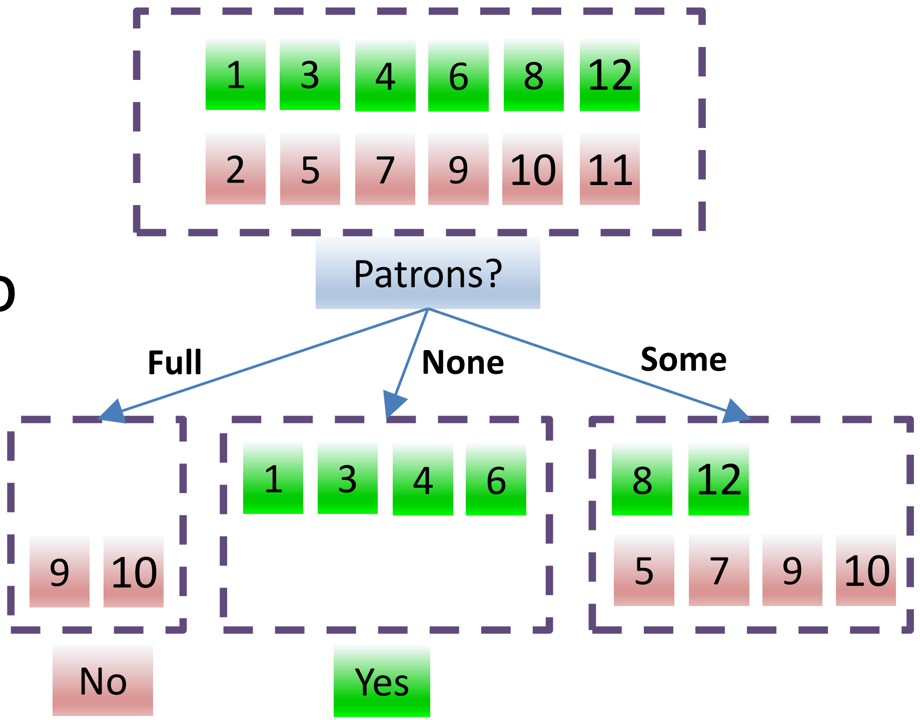
Back at the restaurant



- If we choose [**Type?**]
  - Go along branch *French*: We have entropy = 1 bit; similarly for the others.
  - Information gain = 1-1 = 0 along any branch.
- If we choose [**Patrons?]**
  - In branch *Full* and *None* entropy = 0.
  - For branch *Some*, entropy = -2/6 $\log_2$ (2/6) - 4/6 $\log_2$(4/6) = 0.92 bit.
- So choosing [**Patrons?**] gains more information.

Entropy across branches

- How do we combine entropy of different branches?
  - Compute average entropy.
- Weight entropies according to probabilities of branches
  - P(*None*) = 2/12 = 1/6.
  - P(*Some*) = 4/12 = 1/3.
  - P(*Full*) = 6/12 = 1/2.



Average entropy

$= P(Full)\, H(Full) + P(None)\, H(None) + P(Some)\, H(Some)$

$= 1/6 \times 0 + 1/3 \times 0 + 1/2 \times 0.92$

$= 0.46$ bit.

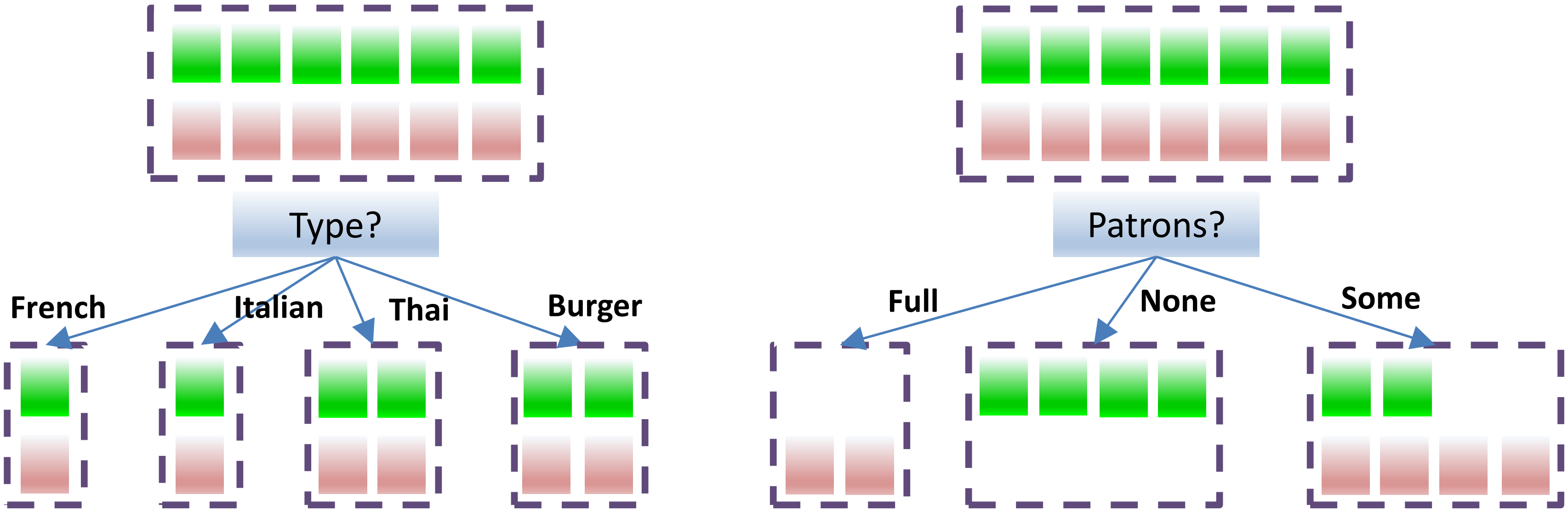
Information gain

Information gain (*IG*) or reduction in entropy from using feature *A*:

IG(*A*) = Entropy before
– average entropy after choosing A.

Our strategy: Choose the feature with the largest *IG*.

Information Gain in our example



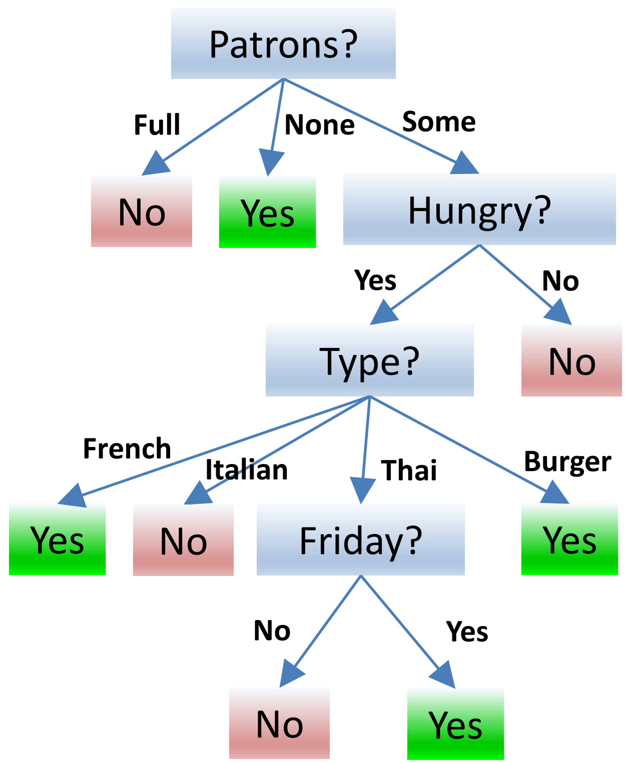
$$\mathrm{IG}(Patrons) = 1 - \left[\frac{1}{6} H(Full) + \frac{1}{3} H(None) + \frac{1}{2} H(Some) \right] = 0.54 \text{ bit.}$$

$$\mathrm{IG}(Type) = 1 - \left[\frac{1}{6} H(French) + \frac{1}{6} H(Italian) + \frac{1}{3} H(Thai) + \frac{1}{3} H(Burger) \right] = 0 \text{ bit.}$$

Patrons has the highest IG of all attributes.

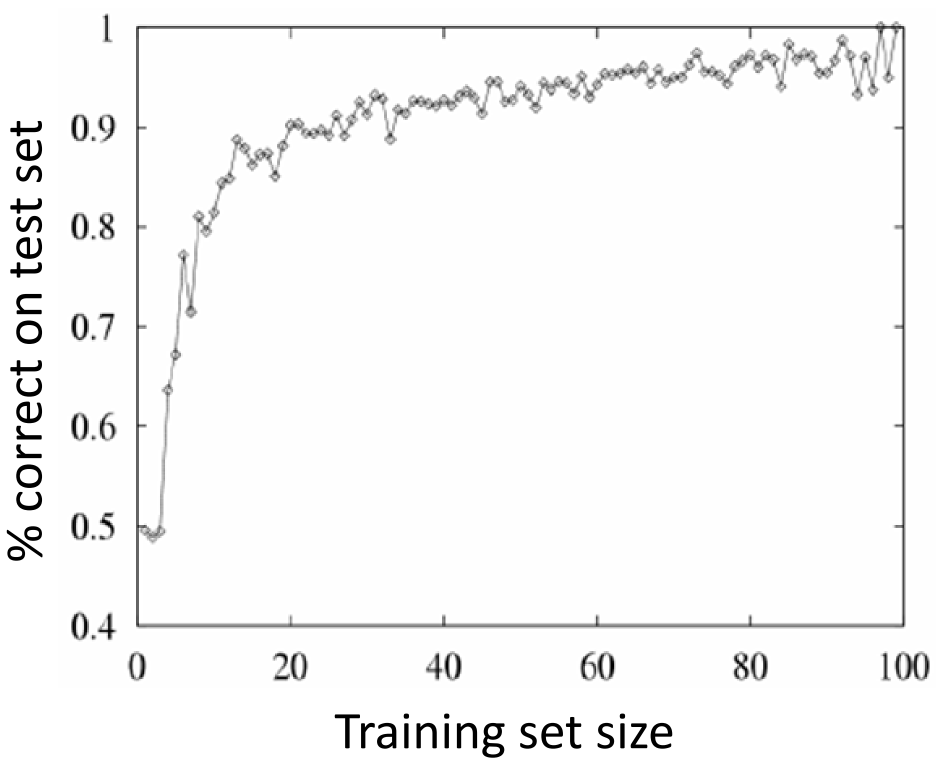
So we choose *Patrons* as the root.

Learned decision tree



- After root node, we repeat process of `choosing an attribute to split on' for each sub-tree, until we get to leaf nodes that are completely or approximately classified to one class.
- In this case, the learned tree is much simpler than the original tree.

Assessing the performance of a learning algorithm



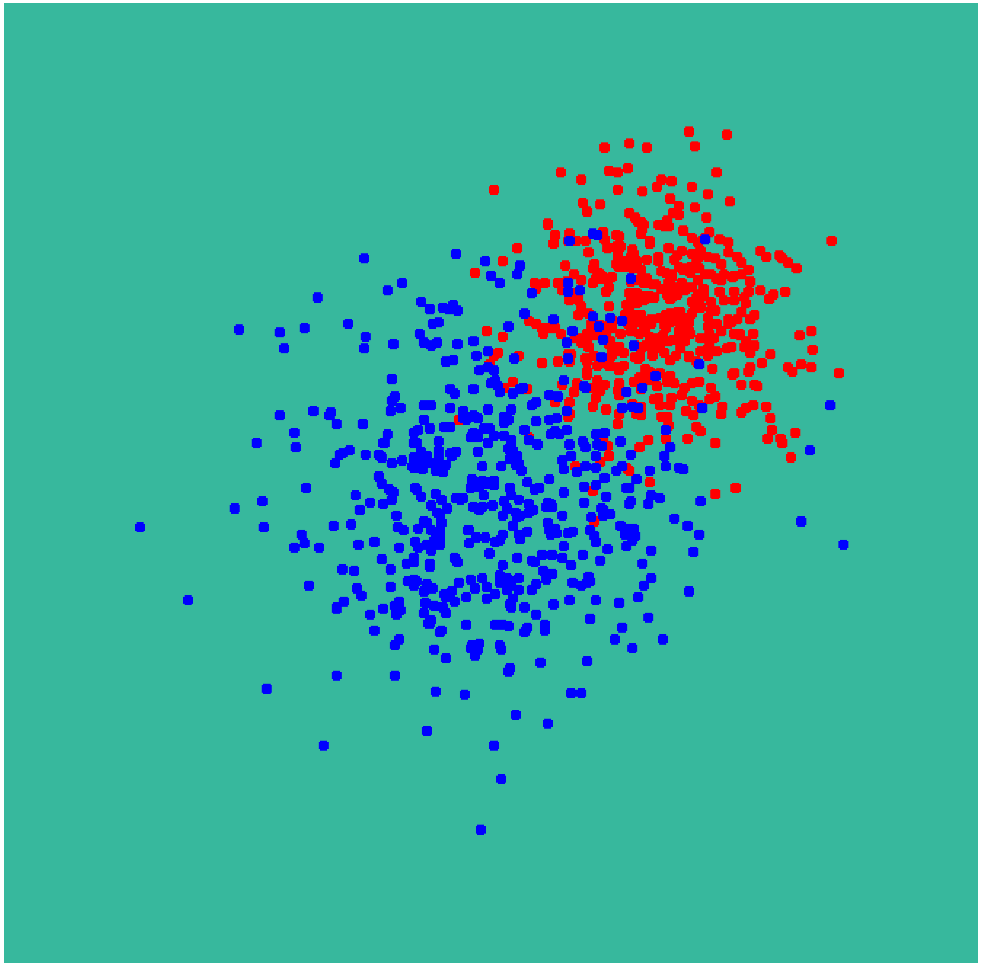
- Randomly divide the examples into a training set and a test set.
- Plot shows a learning curve.

Noise and overfitting

- In the presence of noise, some feature vectors will have multiple examples with conflicting results:

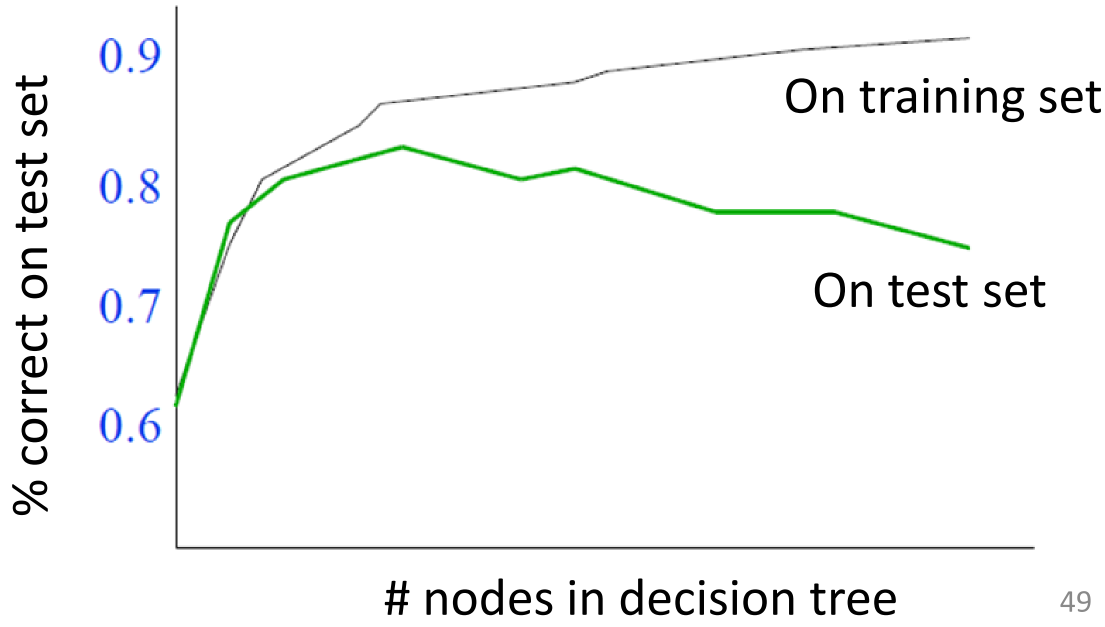
  Different outcomes for the same inputs.
- We must be careful to avoid finding meaningless regularity in the data (*overfitting*)
  - Every time I roll the dice with my left hand it comes up heads.
  - I always encounter less traffic on Mondays (but I'm always late on Mondays).
- *Regularization*: Sometimes, it is good to stop expanding the tree even when training error > 0.

If data are noisy, achieving zero training error would lead to poor generalization.



When to stop?

- Decision trees will overfit
- Strategies for constructing simpler trees
  - Minimum number of data points per leaf.
  - Fixed maximum depth.

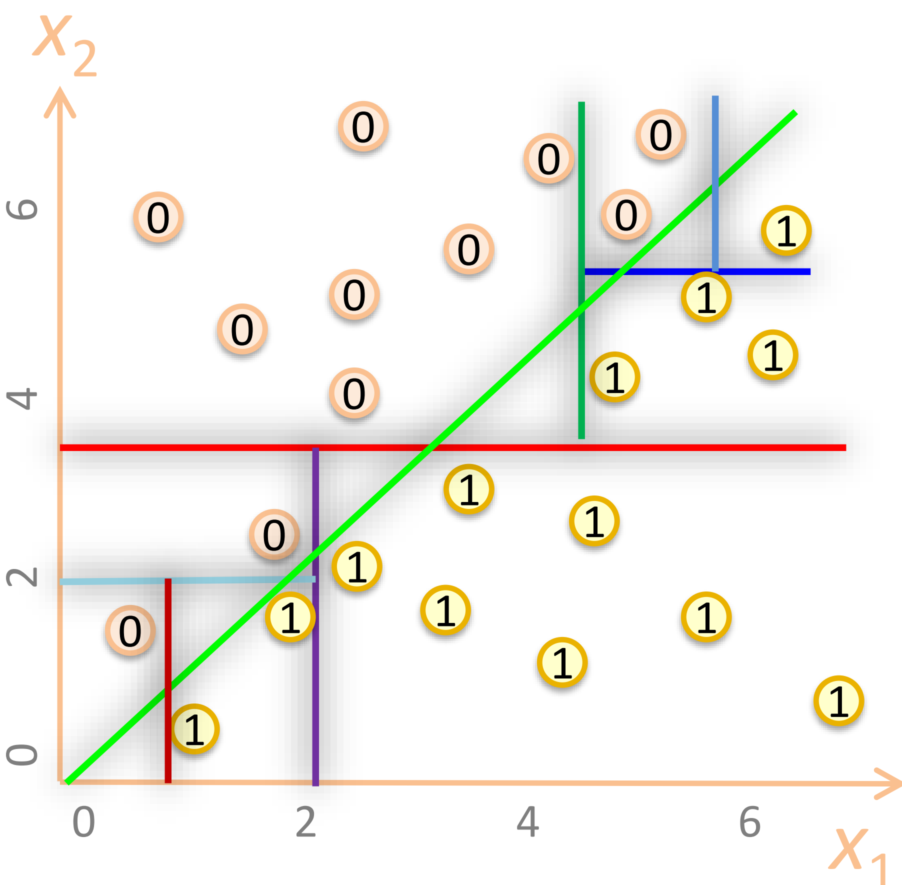


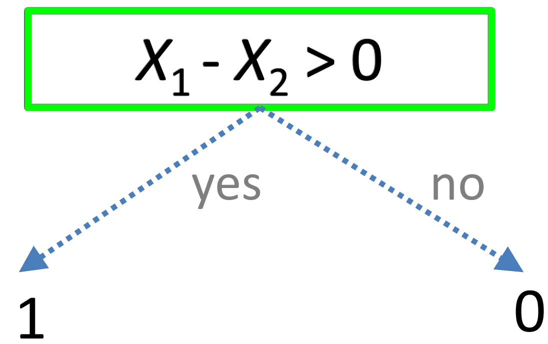
Decision trees examples

- **Designing oil platform equipment**
  - GASOIL (1986 – BP).
  - Designing gas-oil separation systems for offshore platforms.
  - 2500 rules.
  - Would have taken 10 person-years to build by hand.
  - Decision tree took 100 person-days to implement and train.

- **Learning to Fly**
  - C4.5 (1992 - Sammut et al.).
  - Cessna on a flight simulator.
  - Observe 3 human pilots make 30 assigned flights.
  - Create training example every time a control is touched.
  - Flies better than the human instructors!
    - (allows generalisation across errors).

A hard case for simple features





Complex split functions vs. complex tree.

Other splitting criteria

- Entropy (for classification)

$$H(x) = \sum_{i=1}^{n} -P\left(\mathrm{A}^i\right)\log_2 P\left(\mathrm{A}^i\right).$$

- Gini impurity (for classification)

$$I_G(x) = \sum_{i=1}^{n} P\left(\mathrm{A}^i\right)\left(1 - P\left(\mathrm{A}^i\right)\right).$$

- Residual sum of squares (for regression)

$$\mathrm{RSS}(x) = \sum_{i=1}^{n}\left(\overline{\mathrm{A}} - \mathrm{A}^i\right)^2.$$

$\overline{\mathrm{A}}$: mean *y* value.

Decision trees summary

- Efficient learning algorithm.
- Handle both discrete and continuous inputs and outputs.
- Robust to outliers.
- Automatically ignore irrelevant features: no need for feature selection.
- Decision trees are usually interpretable.
- Strategies to terminate building the tree
  - Too deep and we overfit (doesn't generalize well to unseen data).
  - Too shallow and we underfit (poor performance again).
  - (Hyper-)parameters: maximum depth, # data points per leaf, etc.

Random forests (RFs)

$$D = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\} \subset \mathbf{R}^n \times \{0,1\}$$

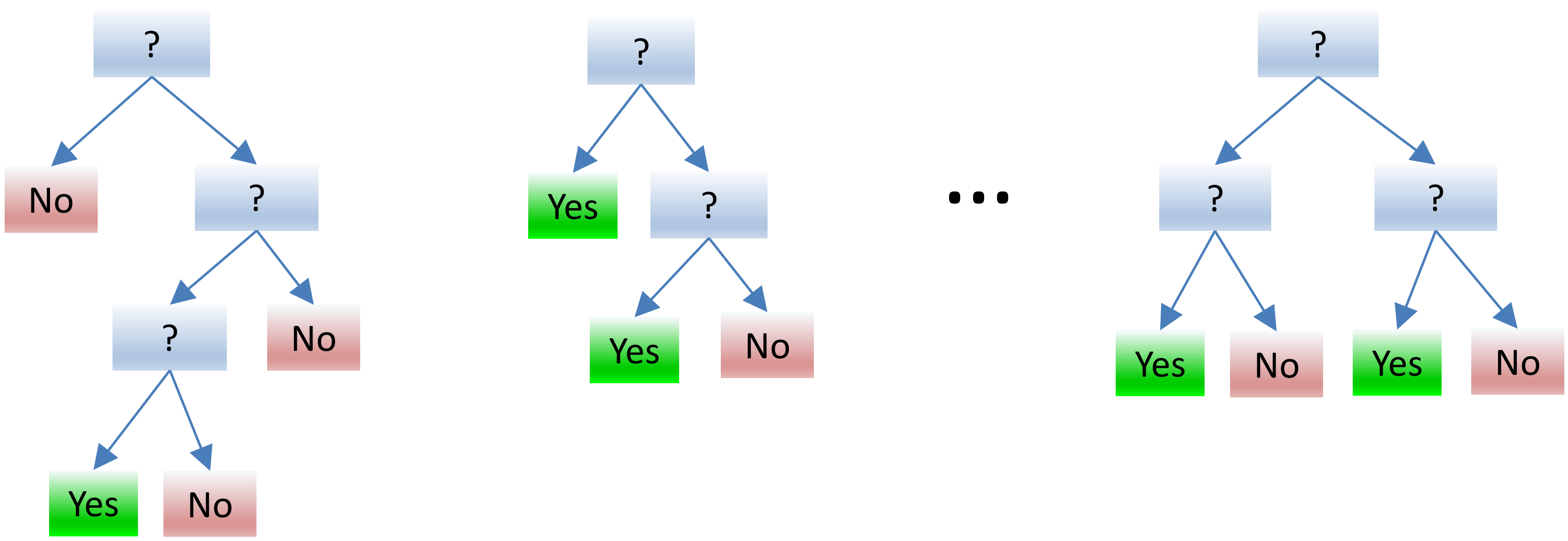
Ensemble method – averages over multiple, diverse classification trees (a *forest*).

- For each tree, randomly draw a subset (of size $L < N$) of the original training set D.
- At each node, randomly sample $p < n$ attributes and choose the best among them.
- Aggregate across trees (majority vote or average $\rightarrow$ mixture model).
- Avoids over-fitting and computationally efficient (easy to parallelize).
- Random forests are usually not interpretable.

Random forests (RFs)

$$D = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\} \subset \mathbf{R}^n \times \{0,1\}$$

$D_1 \subset D$ $\quad$ $D_2 \subset D$ $\quad \cdots \quad$ $D_s \subset D$



T_1 $\quad$ T_2 $\quad$ T_s

RF hyper-parameters

- # trees, maximum depth of each tree, split function, etc.
- These hyper-parameters have to be decided before we train RFs.
- Ideally, we would tune them based on the expected error:

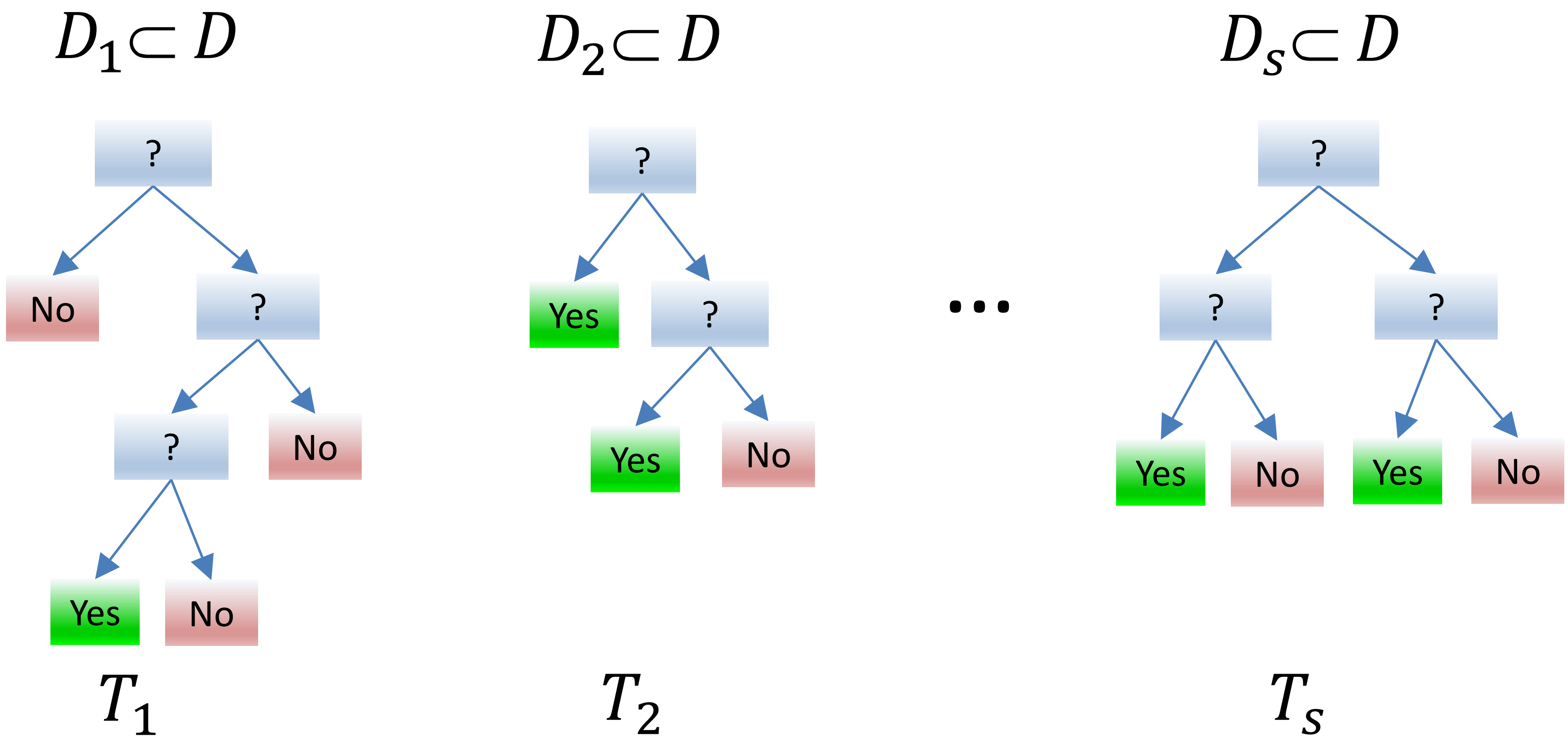
$$\int l\big(f(x), y(x)\big)dP(x, y)$$

- But we do not have access to the data generation process.

Estimates of predictive accuracy

- Training error
  - Use the accuracy on the training set as an estimate of generalization error.
  - Typically, too optimistic.
- Cross-validation
  - Randomly select a training set, use the rest as the test set.
  - 2-fold cross-validation
    - Divide the data at random into 2 pieces, D_1 and D_2.
    - Train the predictor on D_1 and test it on D_2.
    - Train the predictor on D_2 and test it on D_1.
    - Calculate the average over splits.
  - 10-fold cross-validation.

Out-of-bag error



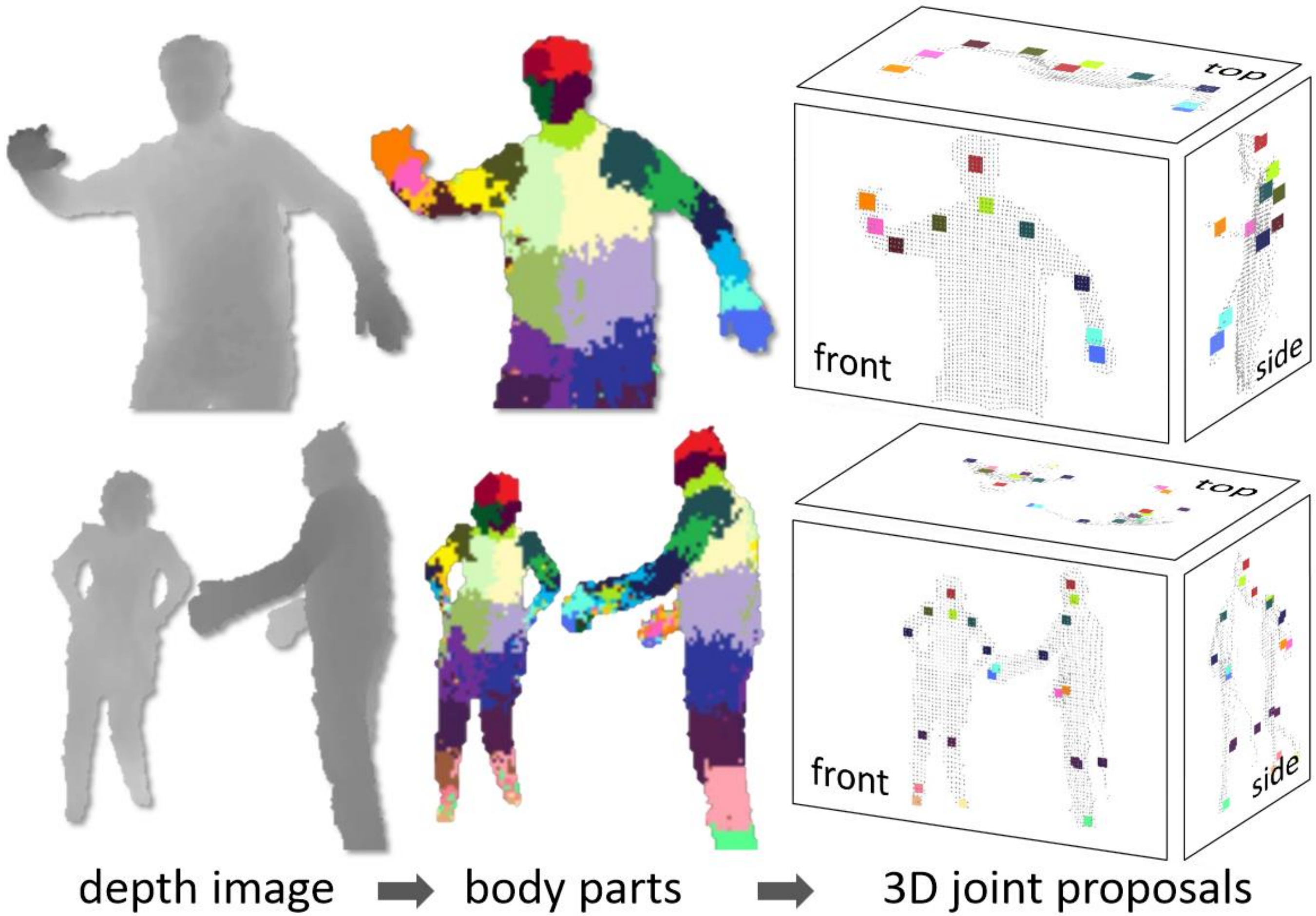
- Each tree T_i is trained on $D_i \subset D$; It can be validated on $D \backslash D_i$.
- Out-of-bag estimate: aggregated validation error for all trees.

Demo

Motion classification example

[Big League Sports]

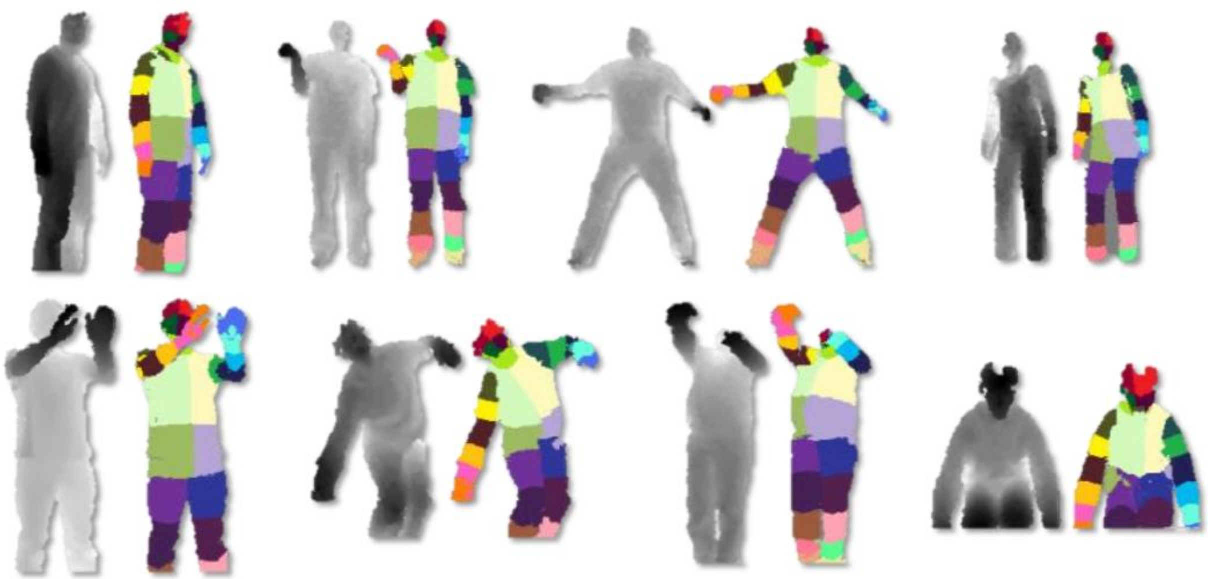
Body pose estimation in MS Kinect



[Shotton et al. 2011]

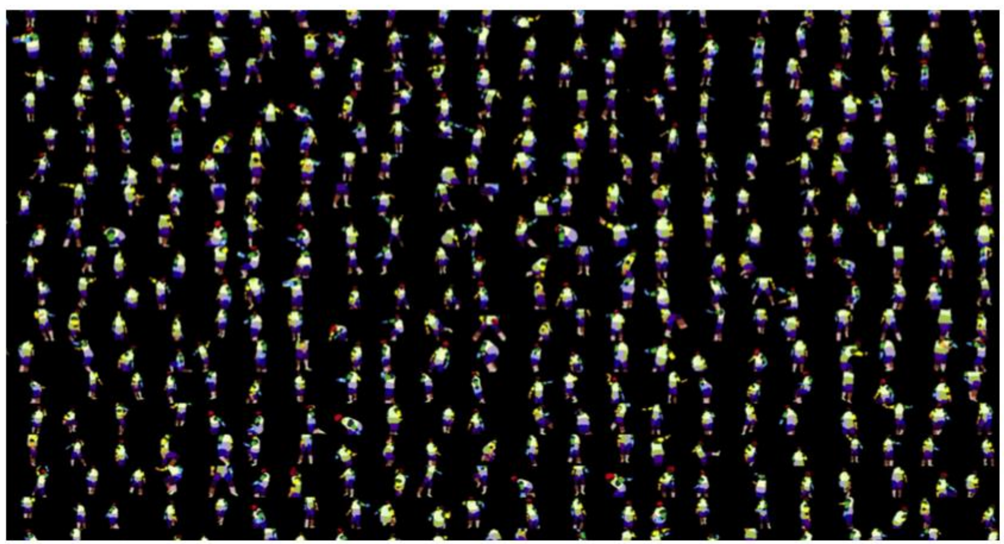
Body pose estimation in MS Kinect

Synthetic training data



Real testing data

Body pose estimation in MS Kinect



1 million test images, 1 day using a 1000 core cluster

Body pose estimation in MS Kinect

Depth image features