Goals for Splitting Nodes

- We want (summed diversity within children) < (diversity in parent)
- Data points should be
  - Homogeneous (by labels) within leaves
  - Different between leaves
- Goal: try to increase purity within subsets
  - *Optimization* goal in each node: find the *attribute* and a *cutpoint* that splits the set of samples into two subsets with *optimal purity*
  - This attribute is the "most discriminative" one for that data (sub-) set
- Question: what is a good measure of purity for two given subsets of our training set?

Digression: Information Gain in Politics/Journalism

- Politician *X* is accused of doing something wrong
- He is asked (e.g., by journalists): "Did you do it?"
- The opposition (assuming *X* is a member of the ruling party) is asked: "Do you think he did it?"
- The answers are reported in the news ...
- What information do you gain?

Information Gain

- Enter the information theoretic concept of information gain
- Imagine different events:
  - The outcome of rolling a dice = 6
  - The outcome of rolling a *biased* dice = 6
  - Each situation has a different amount of uncertainty whether or not the event will occur
- Information = *amount of reduction in uncertainty* (= amount of surprise if a specific outcome occurs)

- Quiz:
  - I am thinking of an integer number in [1,100]
  - How many yes/no questions do you need at most to find it out?
  - Answer: $\lceil \log_2 100 \rceil = 7$
- Definition Information Value:
  - Given a set *S*, the maximum work required to determine a specific element in *S* by traversing a decision tree is

$$\log_2 |S|$$

  - Call this value the information value of being *told* the element, rather than having to *work* for it (by asking binary questions)

- Let Y be a random variable; then we make one observation of the variable Y (e.g., we draw a random ball out of a box) $\longrightarrow$ value y
- The information we obtain if event "$Y = y$" occurs, i.e., the information value of that event, is

$$I[Y = y] = \log_2\left(\frac{\#\text{ balls in box}}{\#\text{ y's in box}}\right) = \log_2 \frac{1}{p(y)} = -\log p(y)$$

  - "If the probability of this event happening is small and it happens, then the information is large"
- Examples:
  - Observing the outcome of coin flip $\longrightarrow I = -\log \frac{1}{2} = 1$
  - Observing the outcome of dice == 6 $\longrightarrow I = -\log \frac{1}{6} = 2.58$

Entropy

- A random variable Y (= experiment) can assume different values $y_1, \ldots, y_n$ (i.e., the experiment can have different outcomes)
- What is the *average* information we obtain by observing the random variable?
  - In other words: if I pick a value y_i at random, according to their respective probabilities – what is the average number of yes/no question you need to ask to determine it?
  - In probabilistic terms: what is the *expected amount of information*? ⟶ captured by the notion of entropy
- Definition: Entropy

  Let Y be a random variable. The entropy of Y is

$$H(Y) = E[I(Y)] = \sum_i p(y_i) I[Y = y_i] = -\sum_i p(y_i) \log p(y_i)$$

- Interpretation: The number of yes/no questions (= bits) needed *on average* to pin down the value of *y* in a random drawing

- Example: if *Y* can assume 8 values, and all are equally likely, then

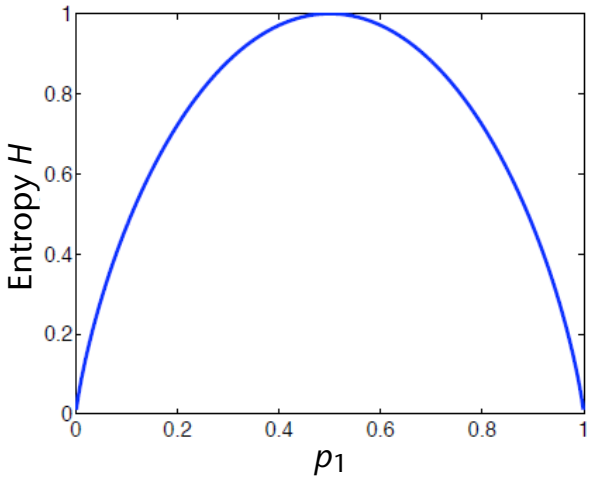
$$H(Y) = -\sum_{i=1}^{8} \frac{1}{8} \log \frac{1}{8} = \log 2^3 = 3 \text{ bits}$$

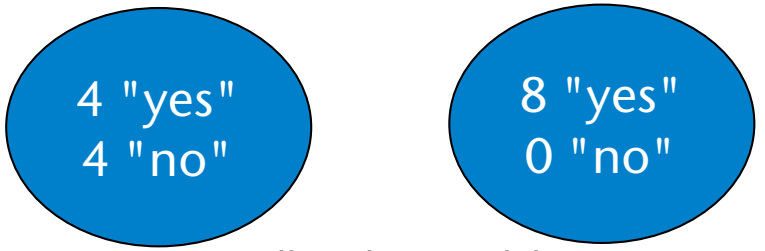
- In general, if there are k different possible outcomes, then

$$H(Y) \leq \log k$$

  - Equality holds when all outcomes are equally likely

- With $k = 2$ (two outcomes), entropy looks like this:
- The more the probability distribution deviates from uniformity, the lower the entropy
- *Entropy* measures the *impurity*:





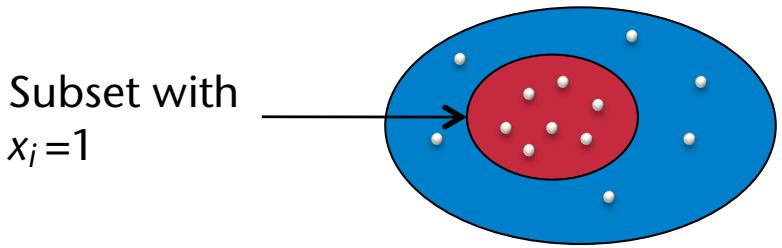
Balls in bin model

This distribution is less uniform
Entropy is lower
The node is purer

Conditional Entropy

- Now consider a random variable Y (e.g., the different classes/labels) with an attribute X (e.g., the first variable, $x_{i,1}$, of the data points, $\mathbf{x}_i$)
  - With every drawing of Y, we also get a value for the associated attribute X
- Assume that X is discrete, i.e., $x_i \in \{1, 2, ..., z\}$
- We now consider only cases of Y that fulfill some *condition*, e.g., x_i=1
- The entropy of Y, provided that it assumes only values with x_i =1:

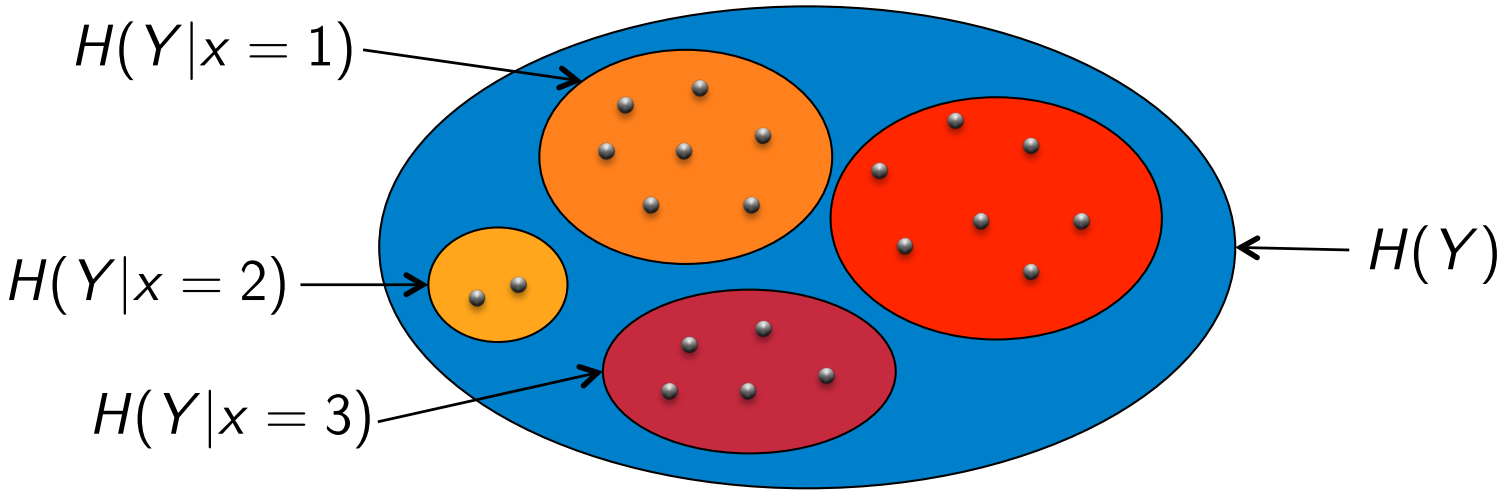
$$H(Y|x_i = 1) = -\sum_i p(y_i|x_i = 1) \log \underbrace{p(y_i|x_i = 1)}_{\text{Probability of } y_i \text{ occurring as a value of } Y, \text{ where we draw } Y \text{ only from the subset that contains only data points that have attribute } x_i=1}$$



- Overall conditional entropy:

$$H(Y|X) = \sum_{k=1}^{z} p(x=k) \cdot H(Y|x=k)$$

$$= -\sum_{k=1}^{z} \underbrace{p(x=k)}_{\text{Probability that the attribute } X \text{ has value } k} \sum_{i} p(y_i|x_i=k) \log p(y_i|x_i=k)$$



Information Gain

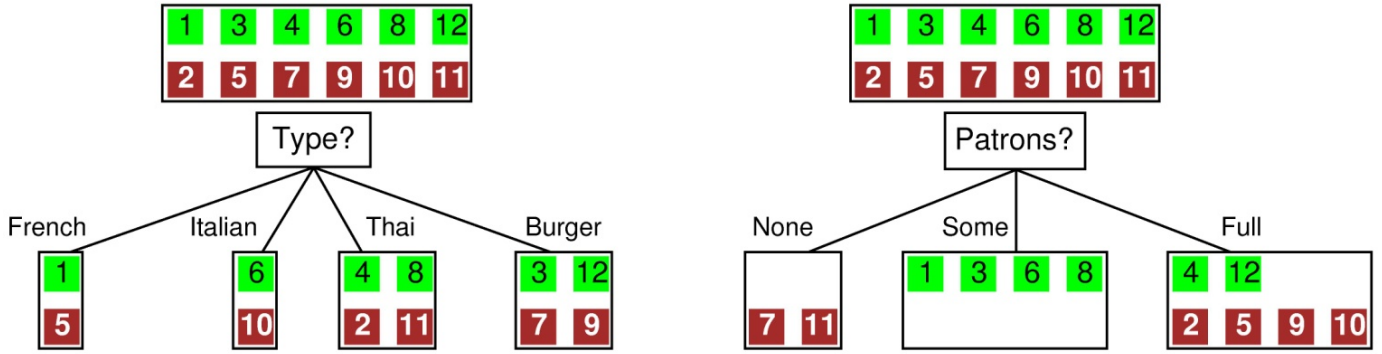
- How much information do we gain *if we disclose the value of one attribute X*?
- Information gain = (information *before* split) – (information *after* split) = *reduction of uncertainty* by knowing attribute *X*
- The information gained by a split in a node of a decision tree:

$$IG(Y, X) = H(Y) - H(Y|X)$$

- Goal: choose the attribute with the largest *IG*
  - In case of scalar attributes, also choose the *optimal cutpoint*

Example

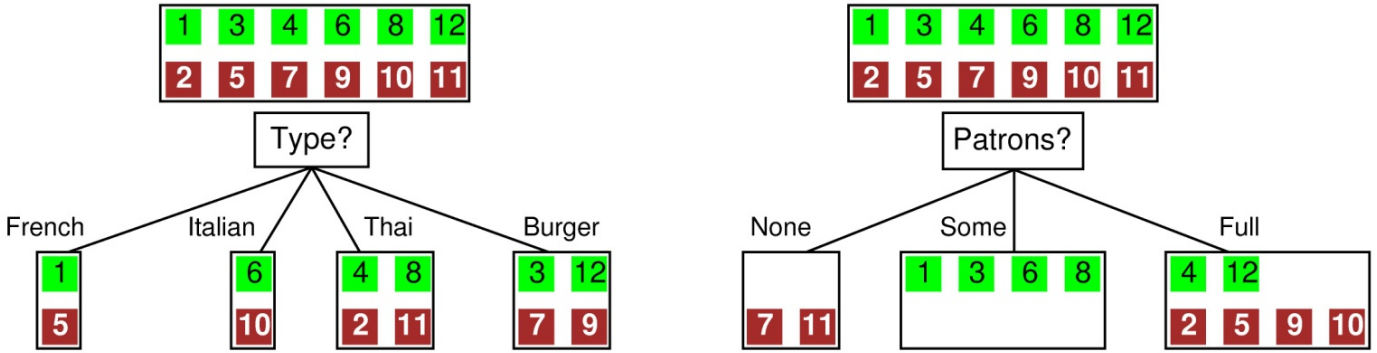
- Consider 2 options to split the root node of the restaurant example



- Random variable $Y \in \{$ "yes", "no" $\}$
- At the root node:

$$H(Y) = p(y = \text{“yes”}) \log \frac{1}{p(y = \text{“yes”})} + p(y = \text{“no”}) \log \frac{1}{p(y = \text{“no”})}$$

$$= \frac{1}{2} \log 2 + \frac{1}{2} \log 2 = 1$$



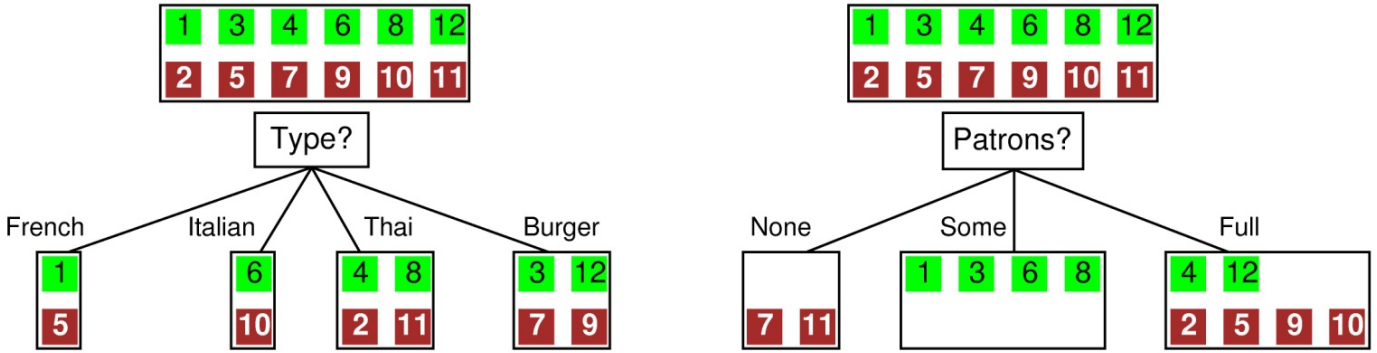
- Conditional entropy for right option:

$$H(Y \mid n) = p(n = \text{“none”}) \cdot H(Y \mid n = \text{“none”}) + p(n = \text{“some”}) \cdot H(Y \mid n = \text{“some”}) + p(n = \text{“full”}) \cdot H(Y \mid n = \text{“full”})$$

where n = the attribute "#patrons" $\in$ { "none", "some", "full" }

$$H(Y|\#\text{patrons}) = \frac{2}{12}\left(p(y = \text{“no”}) \log p(y = \text{“no”}) + p(y = \text{“yes”}) \log p(y = \text{“yes”})\right) + \frac{4}{12}\left(p(y = \text{“no”}) \log p(y = \text{“no”}) + p(y = \text{“yes”}) \log p(y = \text{“yes”})\right) + \frac{6}{12}\left(p(y = \text{“no”}) \log p(y = \text{“no”}) + p(y = \text{“yes”}) \log p(y = \text{“yes”})\right)$$

$$H(Y|\#\text{patrons}) = \frac{2}{12}(1 \log 1 + 0 \log 0) + \frac{4}{12}(0 \log 0 + 1 \log 1) + \frac{6}{12}\left(\frac{4}{6} \log \frac{6}{4} + \frac{2}{6} \log \frac{6}{2}\right)$$



- Conditional entropy for left option:

$$H(Y|\text{type}) = \frac{2}{12}\left(p(y=\text{“no”})\log p(y=\text{“no”}) + p(y=\text{“yes”})\log p(y=\text{“yes”})\right) +$$
$$\frac{2}{12}\left(p(y=\text{“no”})\log p(y=\text{“no”}) + p(y=\text{“yes”})\log p(y=\text{“yes”})\right) +$$
$$\frac{4}{12}\left(p(y=\text{“no”})\log p(y=\text{“no”}) + p(y=\text{“yes”})\log p(y=\text{“yes”})\right) +$$
$$\frac{4}{12}\left(p(y=\text{“no”})\log p(y=\text{“no”}) + p(y=\text{“yes”})\log p(y=\text{“yes”})\right) +$$

$$H(Y|\text{type}) = 2\cdot\frac{2}{12}\left(\frac{1}{2}\log\frac{2}{1} + \frac{1}{2}\log\frac{2}{1}\right) + 2\cdot\frac{4}{12}\left(\frac{2}{4}\log\frac{4}{2} + \frac{2}{4}\log\frac{4}{2}\right)$$

- Compare the information gains:

$$IG(Y, \#\text{patrons}) = H(Y) - H(Y|\#\text{patrons})$$
$$= 1 - 0.585$$

$$IG(Y, \text{type}) = H(Y) - H(Y|\text{type})$$
$$= 1 - 1$$

  - So, the attribute "#patrons" gives us more information about Y

- Compute the *IG* obtained by a split induced by *each attribute*
  - In this case, the optimum is achieved by the attribute "#patrons" for splitting the set of data points in the root

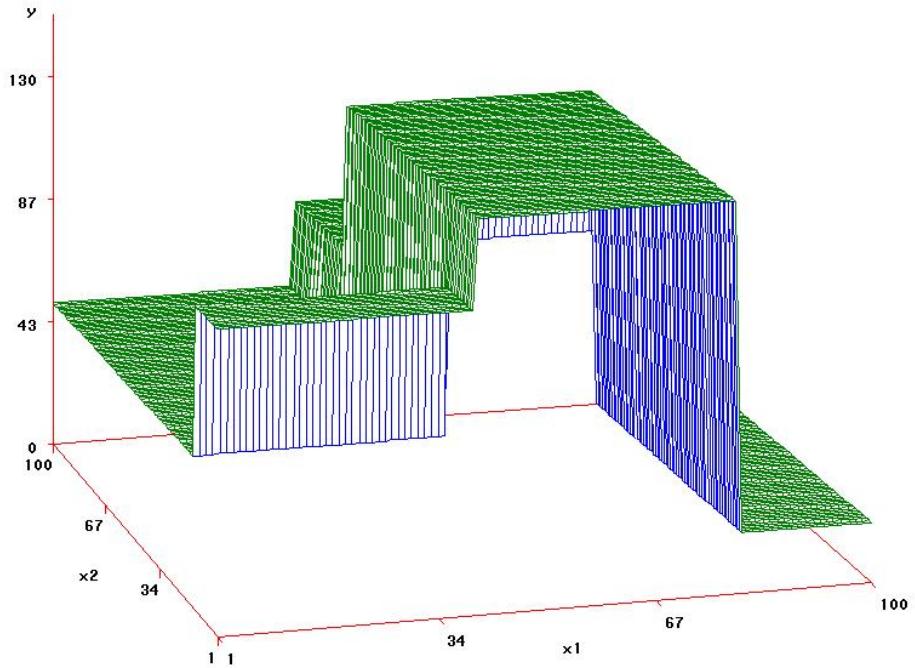
Bits and Pieces

- If there are no attributes left:
  - Can happen during learning of the decision tree, when a node contains data points with same attribute values but different labels
  - This constitutes error / noise
  - Stop construction here, use majority vote (discard erroneous point)
- If there are leaves with no data points:
  - While classifying a new data point
  - Just choose the majority vote of the parent node

Expressiveness of Decision Trees

- Assume all variables (attributes and labels) are Boolean
- What is the class of Boolean functions that can be represented by a decision tree?
- Answer: all Boolean functions!
- Proof (simple):
  - Given any Boolean function
  - Convert it to a truth table
  - Consider each row as a data point, output = label
  - Construct a DT over all data points / rows

- If *Y* is a discrete, numerical variable, then DTs can be regarded as piecewise constant functions over the feature space:

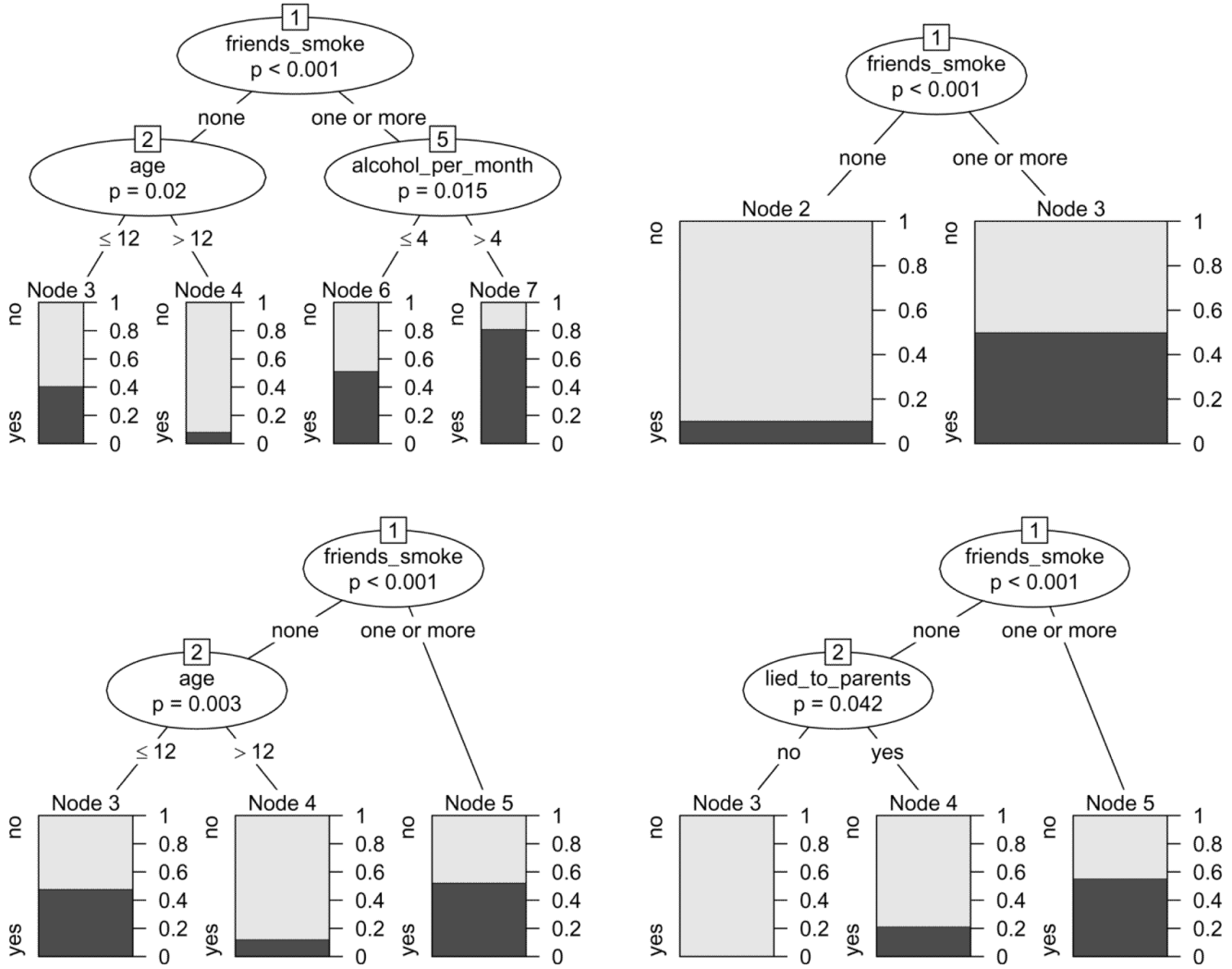


- DTs can approximate *any* function

Problems of Decision Trees

- Error propagation:
  - Learning a DT is based on a series of local decisions
  - What happens, if one of the nodes implements the wrong decision? (e.g., because of an outlier)
  - The whole subtree will be wrong!
- Overfitting: in general, it means the learner performs extremely well on the training data, but very poorly on unseen data ⟶ high generalization error
  - When overfitting occurs, the DT has learned the noise in the data

- Example for the instability of single decision trees:



"The Wisdom of Crowds"

[James Surowiecki, 2004]

- Francis Galton's experience at the 1906 West of England Fat Stock and Poultry Exhibition
- Jack Treynor's jelly-beans-in-the-jar experiment (1987)
  - Only 1 of 56 students' guesses came closer to the truth than the average of the class' guesses
- Who Wants to Be a Millionaire?
  - Call an expert? ⟶ 65% correct
  - Ask the audience? ⟶ 91% correct

- Example (thought experiment):
  "Which person from the following list was *not* a member of the Monkees?"

  (A) Peter Tork (C) Roger Noll

  (B) Davy Jones (D) Michael Nesmith

  - (BTW: Monkeys are a 1960s pop band, comprising 3 band members)
  - Correct answer: the non-Monkee is Roger Noll
  - Now imagine a crowd of 100 people with knowledge distributed as:

    7 know 3 of the Monkees

    10 know 2 of the Monkees

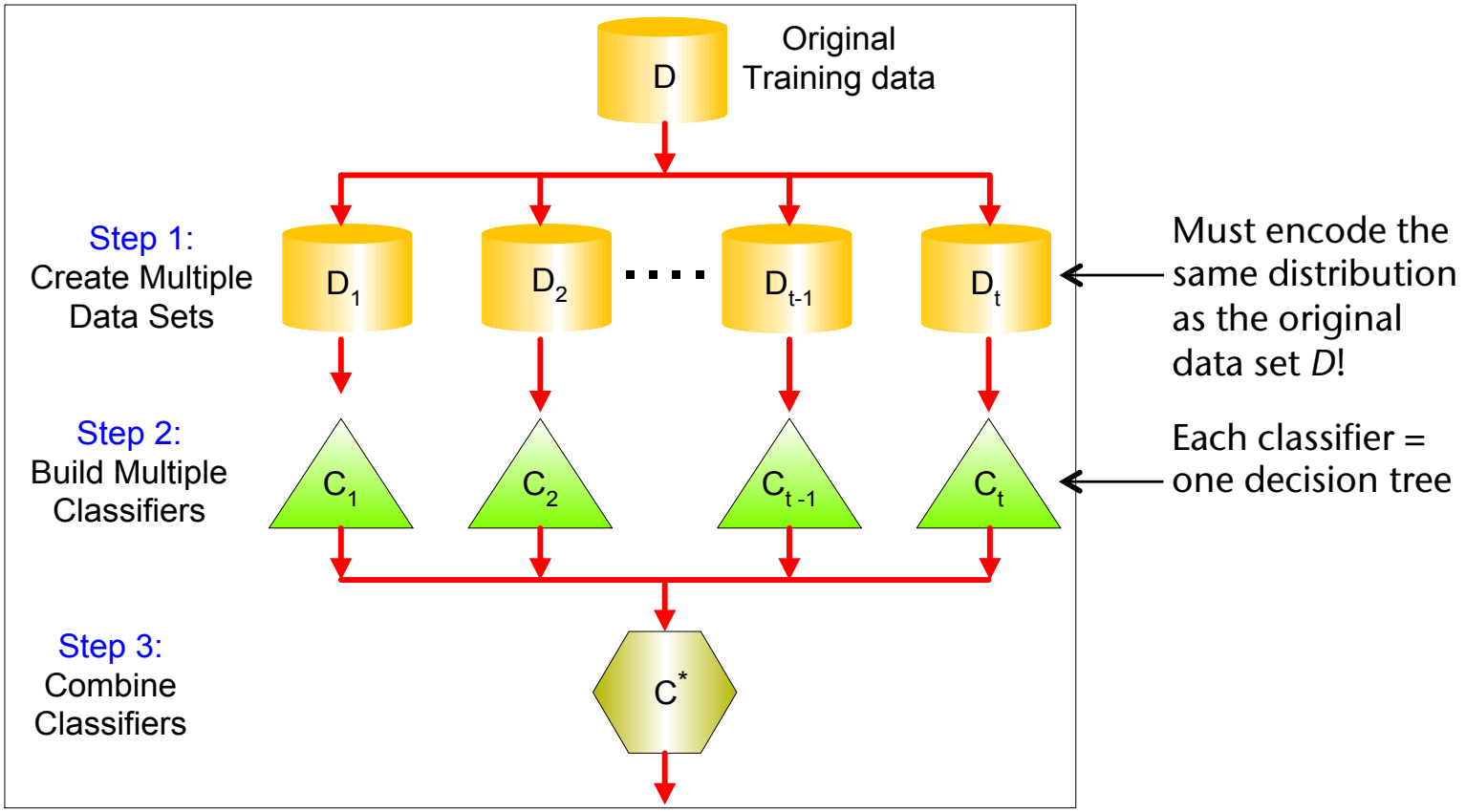
    15 know 1 of the Monkees

    68 have no clue
  - So "Noll" will garner, on average, 34 votes versus 22 votes for each of the other choices

- Implication: one should not expend energy trying to identify an expert within a group but instead rely on the group's collective wisdom
- Counter example:
  - Kindergartners guessing the weight of a 747
- Prerequisites for crowd wisdom to emerge:
  - Opinions must be independent
  - Some knowledge of the truth must reside with some group members (⟶ *weak classifiers*)

The Random Forest Method

- One kind of so-called ensemble (of experts) methods
- Idea: predict class label for unseen data by *aggregating* a set of predictions (= classifiers learned from the training data)



Details on the Construction of Random Forests

- Learning multiple trees:
  - Generate a number of random sub-sets $\mathcal{L}_1, \mathcal{L}_2, \ldots$ from the original training data $\mathcal{L}$, $\mathcal{L}_i \subset \mathcal{L}$. There are basically two methods:

1. Bootstrapping: randomly draw samples, with replacement, size of new data = size of original data set; or,
2. Subsampling: randomly draw samples, without replacement, size of new data < size of original data set

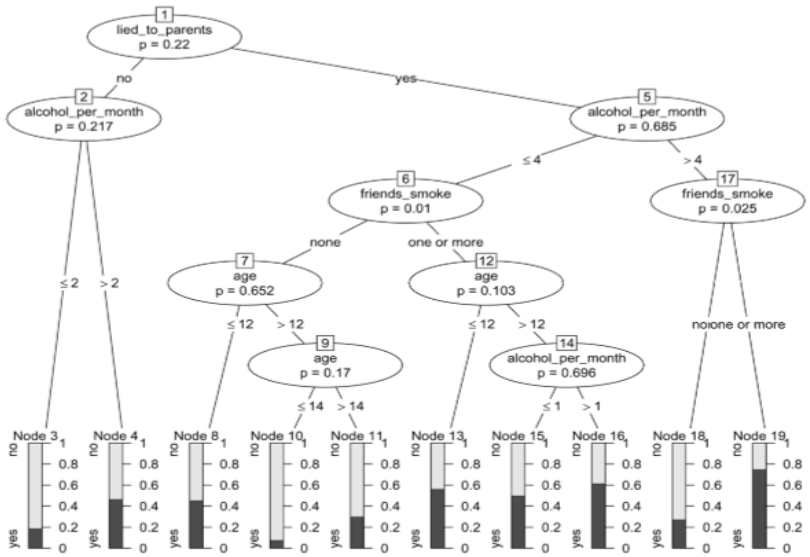
  - New data sets reflect the *same* random process as the orig. data, but they differ slightly from each other and the orig. set due to random variation
  - Resulting trees can differ substantially (see earlier slide)

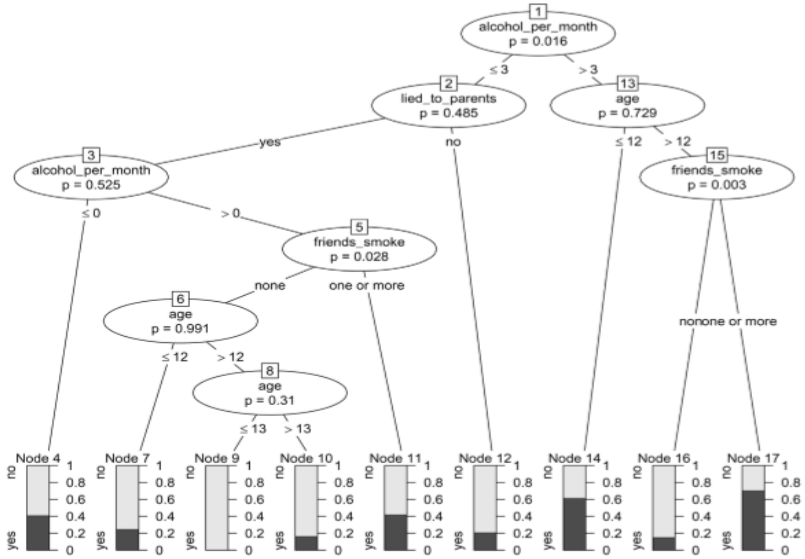
- Growing the trees:
  - Each tree is grown without any stopping criterion, i.e., until each leaf contains data points of only *one single* class
  - At *each node*, a random subset of attributes (= predictor variables/ features) is preselected; *only from those,* the one with the best information gain is chosen
    - NB: an individual tree is *not just a DT over a subspace of feature space*!
- Naming convention for 2 essential parameters:
  - Number of trees = *ntree*
  - Size of random subset of variables/attributes = *mtry*
- Rules of thumb:
  - *ntree* = 100 ... 300
  - *mtry* = sqrt(*d*) , with *d* = dimensions of the feature space

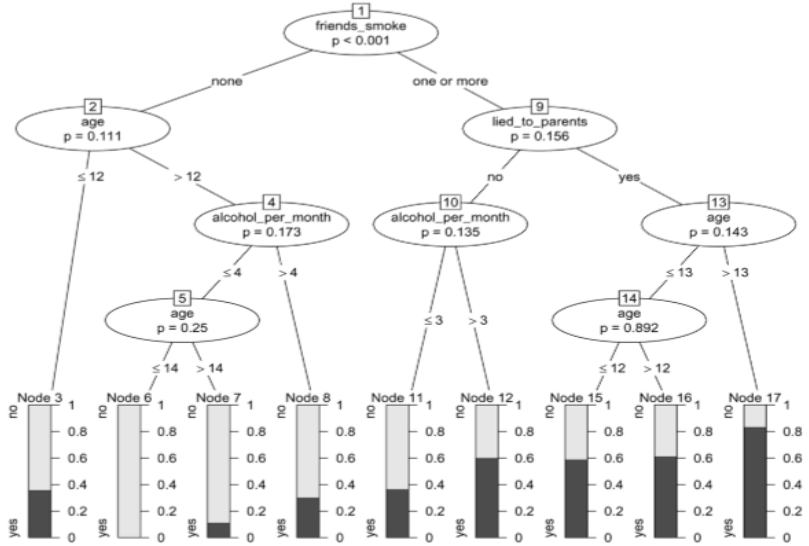
- The learning algorithm:

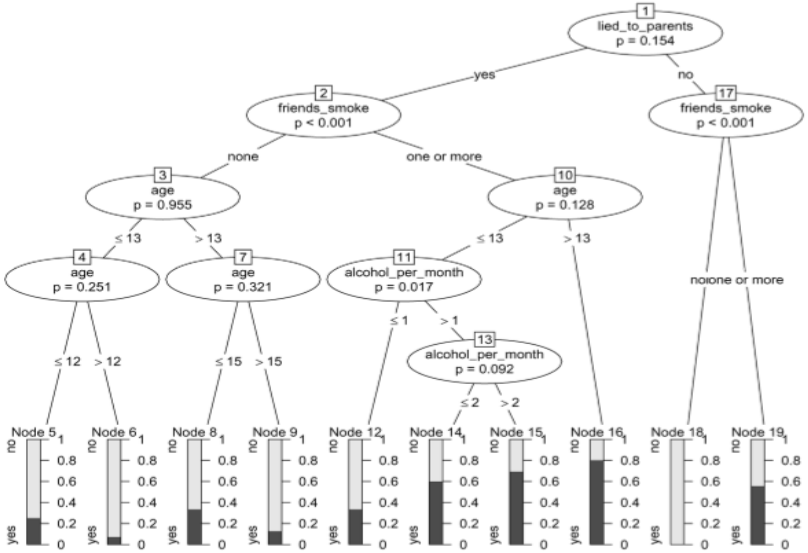
```
input: learning set L
for t = 1...ntree:
   build subset Lt from L by random sampling
   learn tree Tt from Lt:
      at each node:
         randomly choose mtry features
         compute best split from only those features
      grow each tree until leaves are perfectly pure
```

A Random Forest Example for the Smoking Data Set



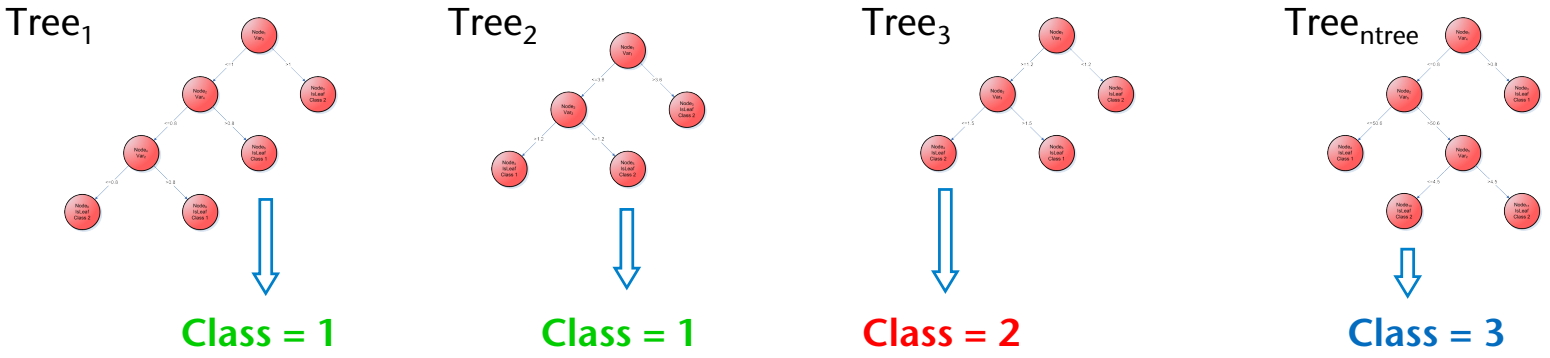






Using a Random Forest for Classification

- With a new data point:
  - Traverse each tree individually using that point
  - Gives *ntree* many class labels



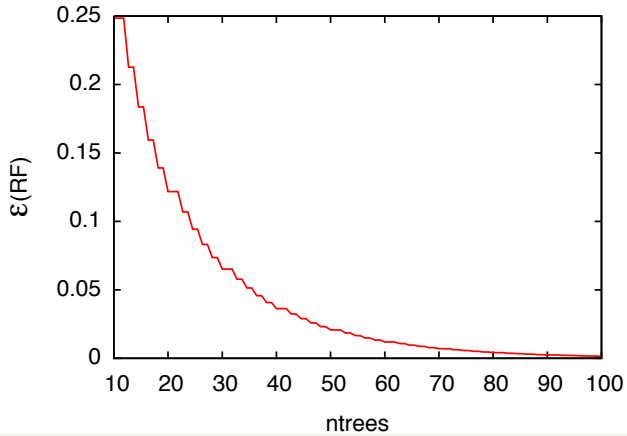
  - Take majority of those class labels
- Sometimes, if labels are numbers, (weighted) averaging makes sense

Why does it Work?

- Make following assumptions:
  - The RF has *ntree* many trees (classifiers)
  - Each tree has an error rate of ε
  - All trees are perfectly independent! (no correlation among trees)
- Probability that the RF makes a wrong prediction:

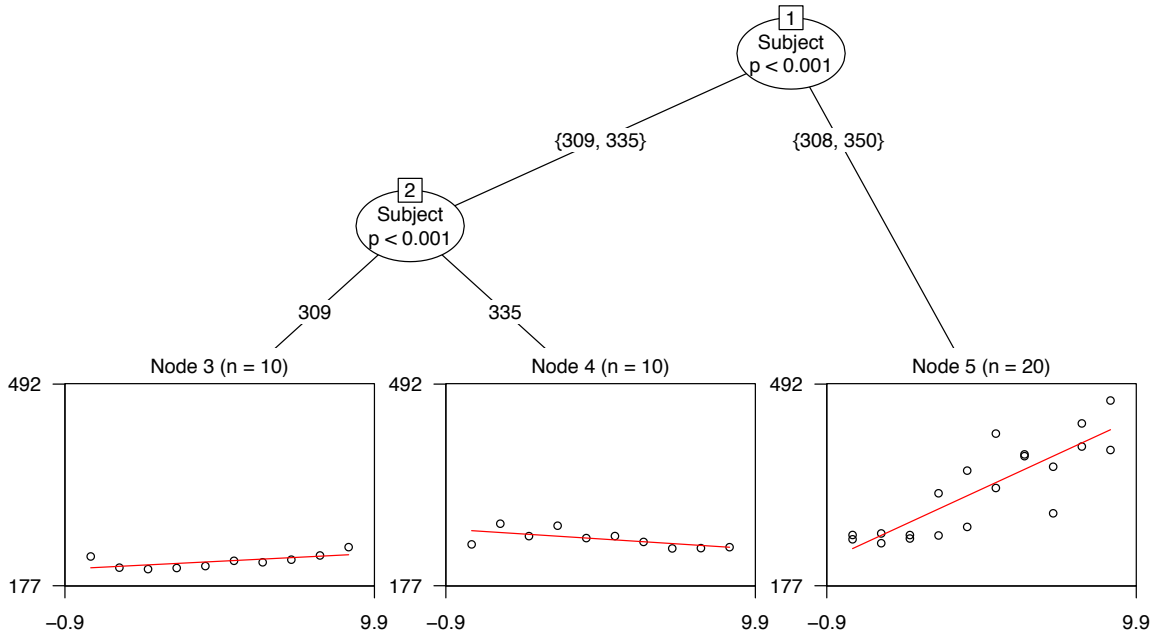
$$\varepsilon_{\text{RF}} = \sum_{i=\lceil \frac{ntree}{2} \rceil}^{ntree} \binom{ntree}{i} \varepsilon^i (1-\varepsilon)^{(ntree-i)}$$

- Example:
  ntree = 60,
  individual error rate $\varepsilon = 0.35 \longrightarrow$
  error rate of RF $\varepsilon_{\text{RF}} \approx 0.01$



Variants of Random Forests

- Regression trees:
  - Variable *Y* (dependent variable) is continuous
    - I.e., no longer a class label
  - Goal is to learn a function $\mathbb{R}^d \rightarrow \mathbb{R}$ that generalizes the training data
  - Example:



Features and Pitfalls of Random Forests

- "Small *n*, large *p*":
  - RFs are well-suited for problems with many more variables (dimensions in the feature space) than observations / training data
- Nonlinear function approximation:
  - RFs can approximate *any* unknown function
- Blackbox:
  - RFs are a black box; it is practically impossible to obtain an analytic function description, or gain insights in predictor variable interactions
- The "XOR problem":
  - In an XOR truth table, the two variables show no effect at all
    - With either split variable, the information gain is 0
  - But there is a perfect interaction between the two variables
  - Random pre-selection of *mtry* variables can help

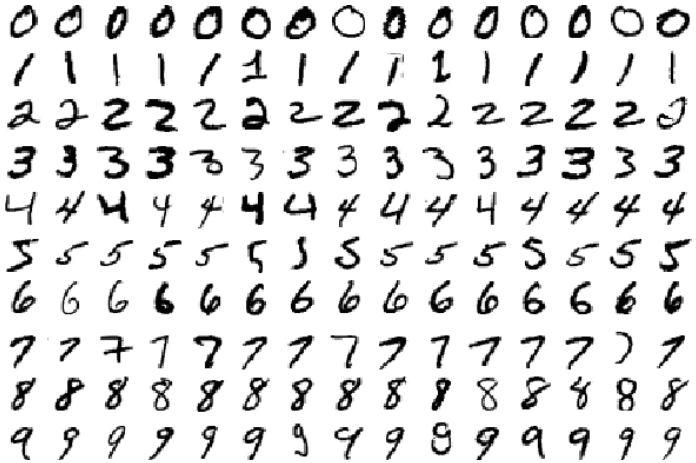
- Out-of-bag error estimation:
  - For each tree T_i, a training data set $\mathcal{L}_i \subset \mathcal{L}$ was used
  - Use $\mathcal{L} \setminus \mathcal{L}_i$ (the out-of-bag data set) to test the prediction accuracy
- Handling missing values:
  - Occasionally, some data points contain a missing value for one or more of its variables (e.g., because the corresponding measuring instrument had a malfunction)
  - When information gain is computed, just omit the missing values
  - During splitting, use a surrogate that best predicts the values of the splitting variable (in case of a missing value)

- Randomness:
  - Random forests are truly random
  - Consequence: when you build two RFs with the same training data, you get slightly different classifiers/predictors
    - Fix the random seed, if you need reproducible RFs
  - Suggestion: if you observe that two RFs over the same training data (with different random seeds) produce noticeably different prediction results, and different variable importance rankings, then you should adjust the parameters *ntree* and *mtry*

- Do random forests overfit?
  - The evidence is inconclusive (with some data sets it seems like they could, with other data sets it doesn't)
  - If you suspect overfitting: try to build the individual trees of the RF to a smaller depth, i.e., not up to completely pure leaves

Application: Handwritten Digit Recognition

- Data set:
  - Images of handwritten digits
  - Normalization: 20x20 pixels, binary images
  - 10 classes



- Naïve feature vectors (data points):
  - Each pixel = one variable ⟶ 400-dim. feature space over {0,1}
  - Recognition rate: ~ 70-80 %
- Better feature vectors by domain knowledge:
  - For each pixel $I(i,j)$ compute:

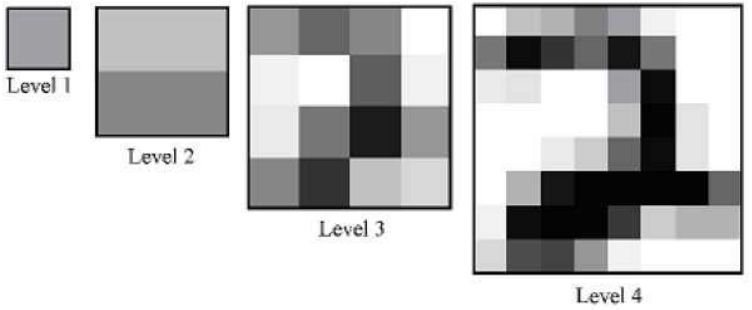
$$H(i,j) = I(i,j) \wedge I(i,j+2)$$
$$V(i,j) = I(i,j) \wedge I(i+2,j)$$
$$N(i,j) = I(i,j) \wedge I(i+2,j+2)$$
$$S(i,j) = I(i,j) \wedge I(i+2,j-2)$$

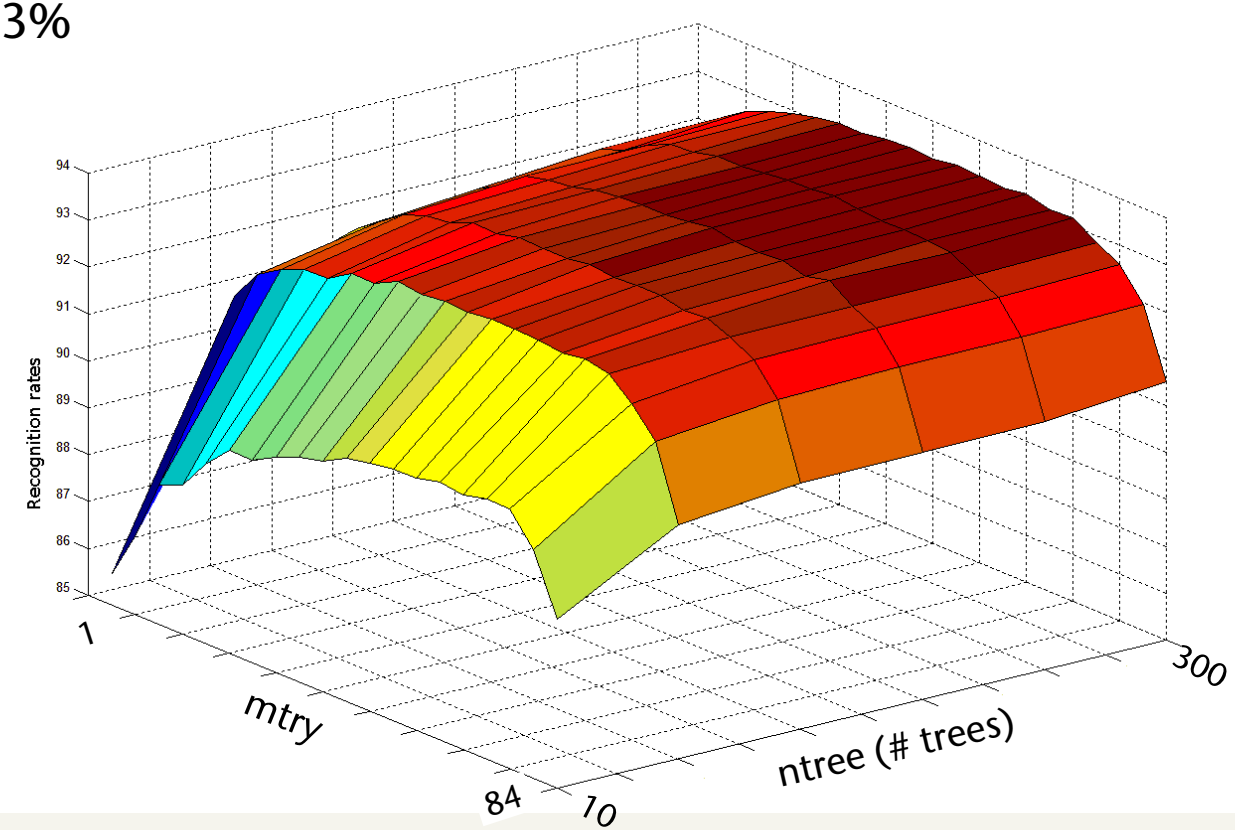
and a few more ...

- Feature vector for an image = (all pixels $I(i,j)$, all $H(i,j)$, $V(i,j)$, ...)
- Feature space = ca. 1400-dimensional = 1400 variables per data point

- Classification accuracy = ~93%
  - Caveat: it was a precursor of random forests

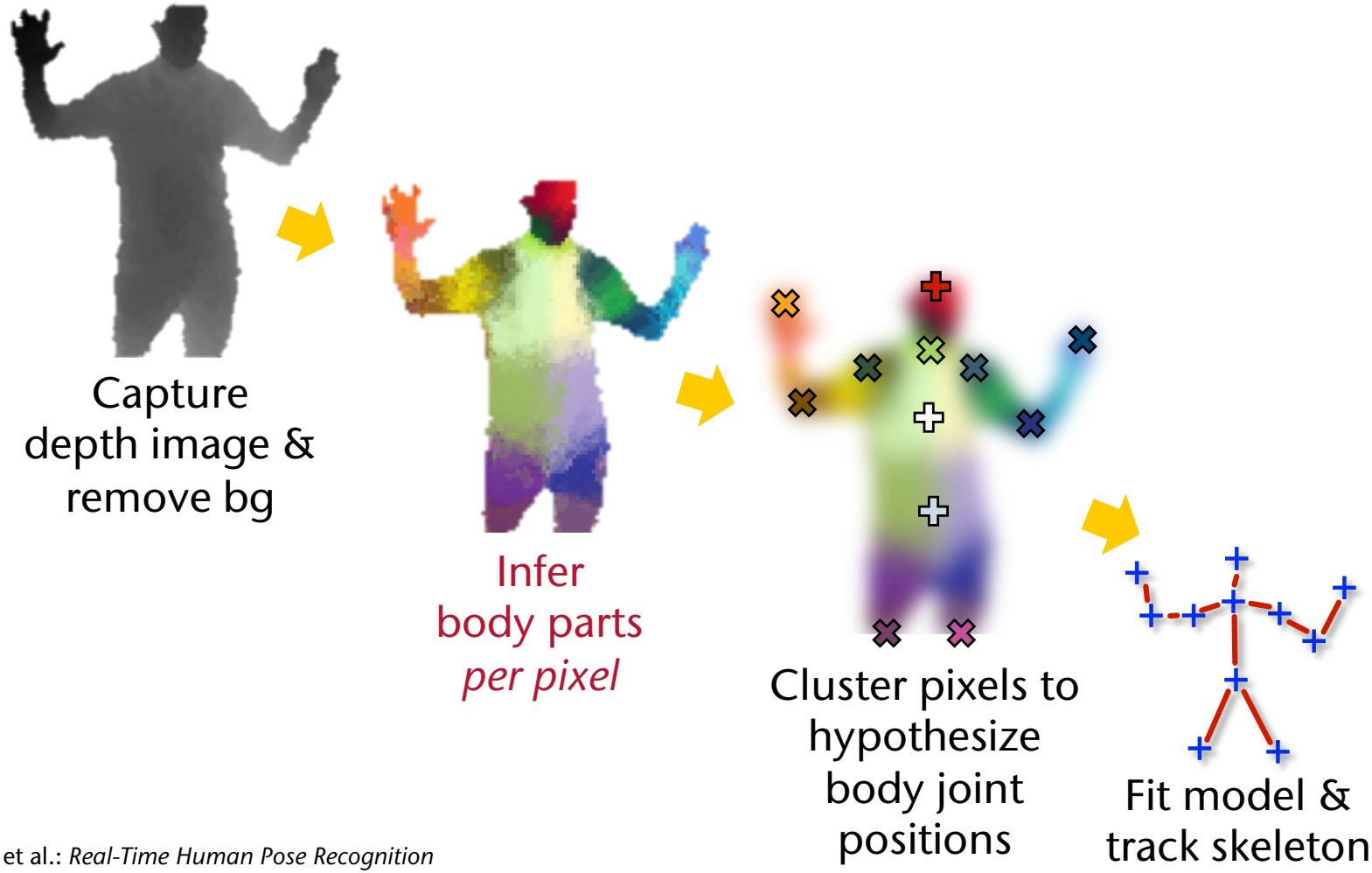
- Other experiments on handwritten digit recognition:
  - Feature vector = all pixels of an image pyramid
  - Recognition rate: ~ 93%
  - Dependence of recognition rate on *ntree* and *mtry*:





Body Tracking Using Depth Images (Kinect)

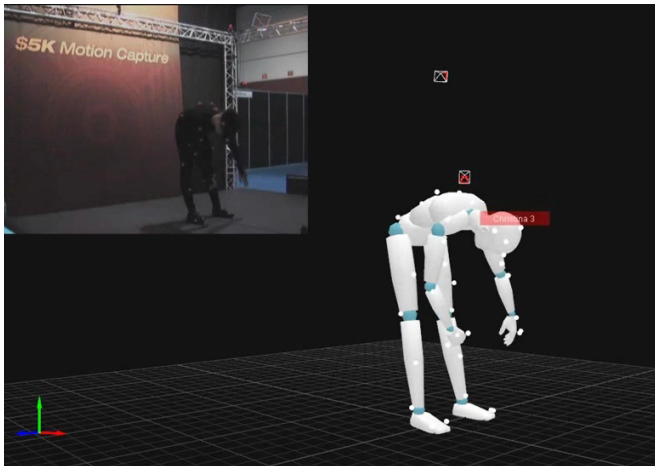
- The tracking / data flow pipeline:



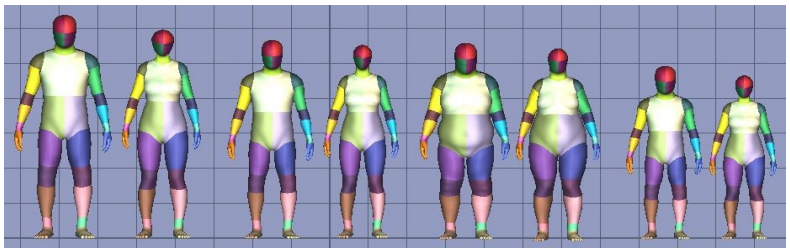
[Shotton et al.: *Real-Time Human Pose Recognition in Parts from Single Depth Images*; CVPR 2011]

The Training Data

Record mocap
500k frames
distilled to 100k poses



Retarget to several models



Render models: store depth & body part ID

Synthetic vs Real Data

synthetic
(train & test)

real
(test)

For each pixel in the depth image, we know its correct class (= label).
Sometimes, such data is also called ground truth data.

Classifying Pixels

- Goal: for each pixel determine the most likely body part (head, shoulder, knee, etc.) it belongs to
- Classifying pixels = compute probability $P(c_{\mathbf{x}})$ for pixel $\mathbf{x} = (x,y)$, where $c_{\mathbf{x}}$ = body part
- Task: learn classifier that returns the most likely body part class $c_{\mathbf{x}}$ for every pixel $\mathbf{x}$
- Idea: consider a neighborhood around $\mathbf{x}$ (moving window)

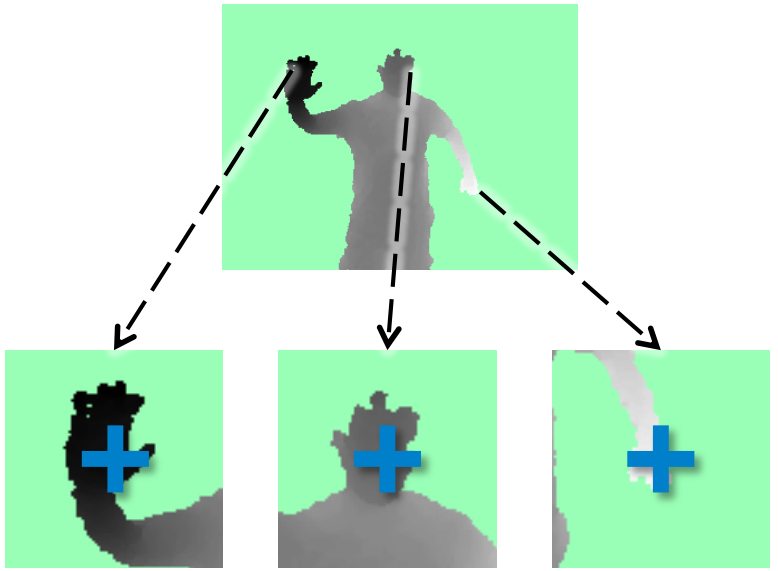
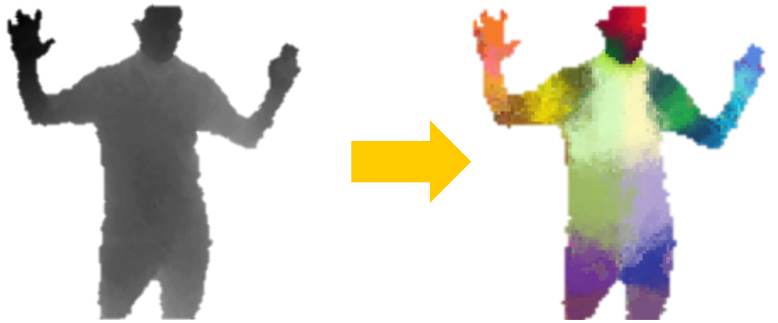


image windows move with classifier



Fast Depth Image Features

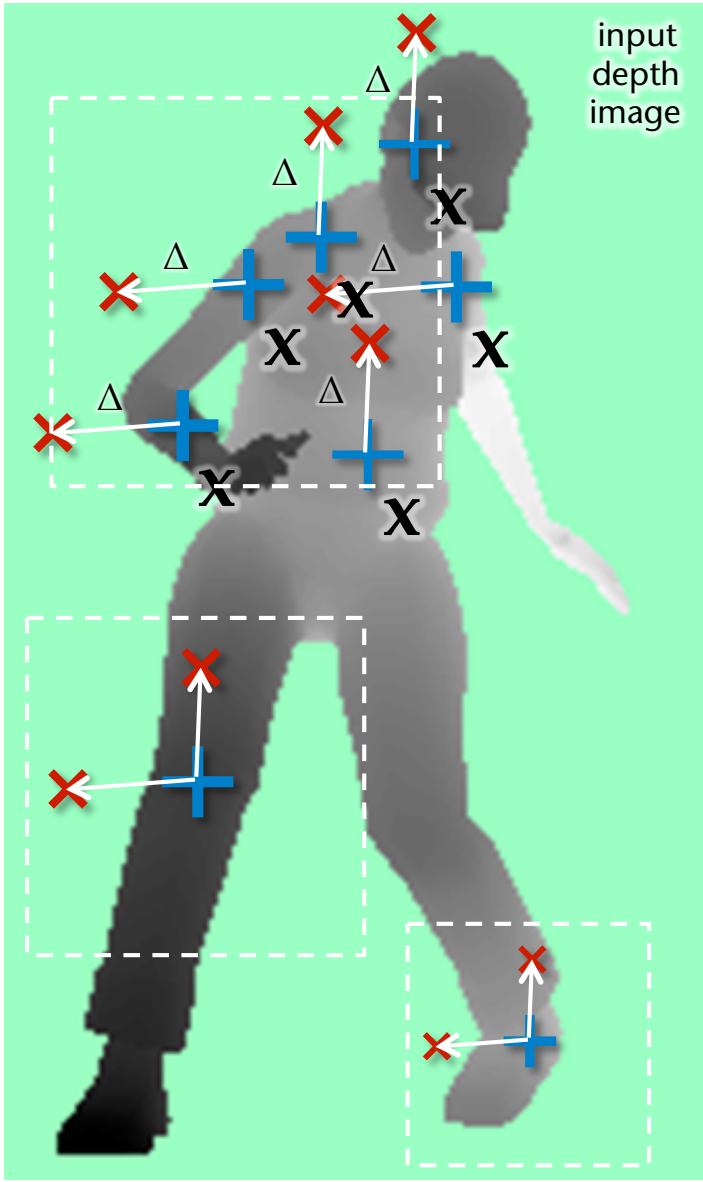
- For a given pixel, consider all depth comparisons inside a window
- The *feature vector* for a pixel **x** are all *feature variables* obtained by all possible depth comparisons inside the window:

$$f(\mathbf{x}, \Delta) = D(\mathbf{x}) - D(\mathbf{x} + \frac{\Delta}{D(\mathbf{x})})$$

where D = depth image,
$\Delta = (\Delta_x, \Delta_y)$ = offset vector,
and D(background) = large constant

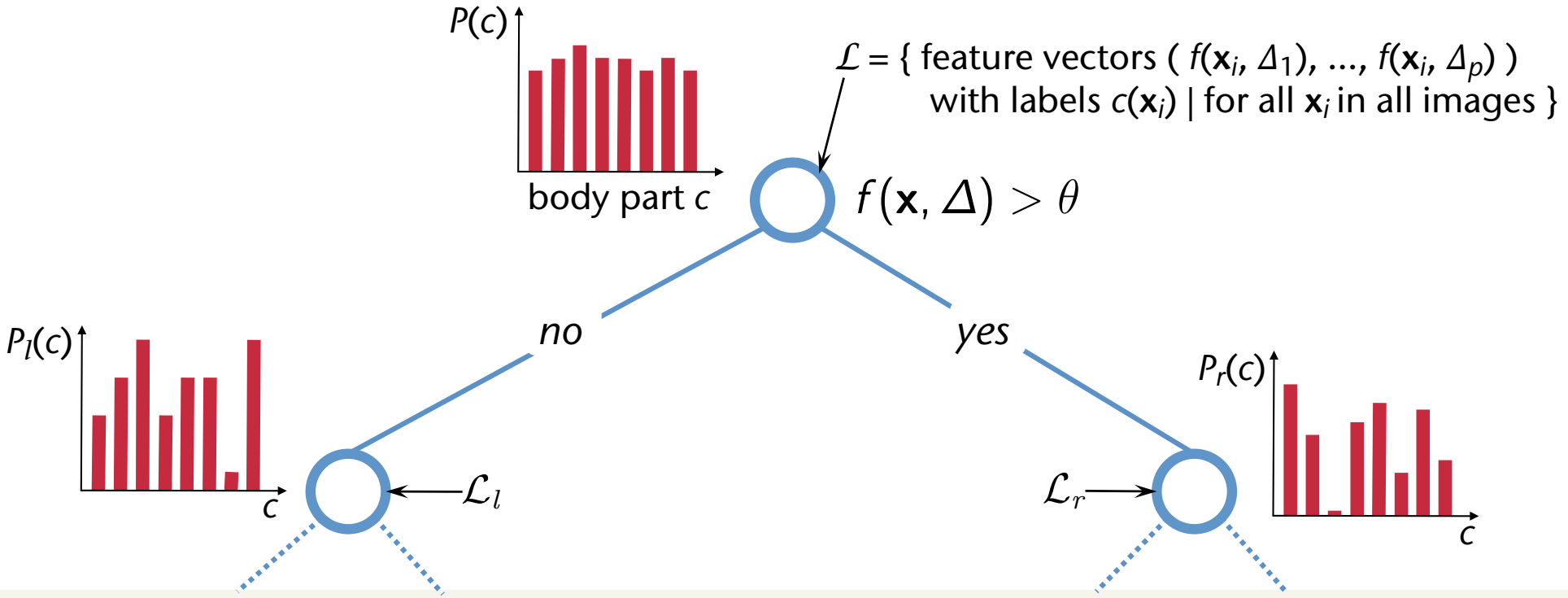
  - Note: scale Δ by 1/depth of **x**, so that the window shrinks with distance

- Features are very fast to compute



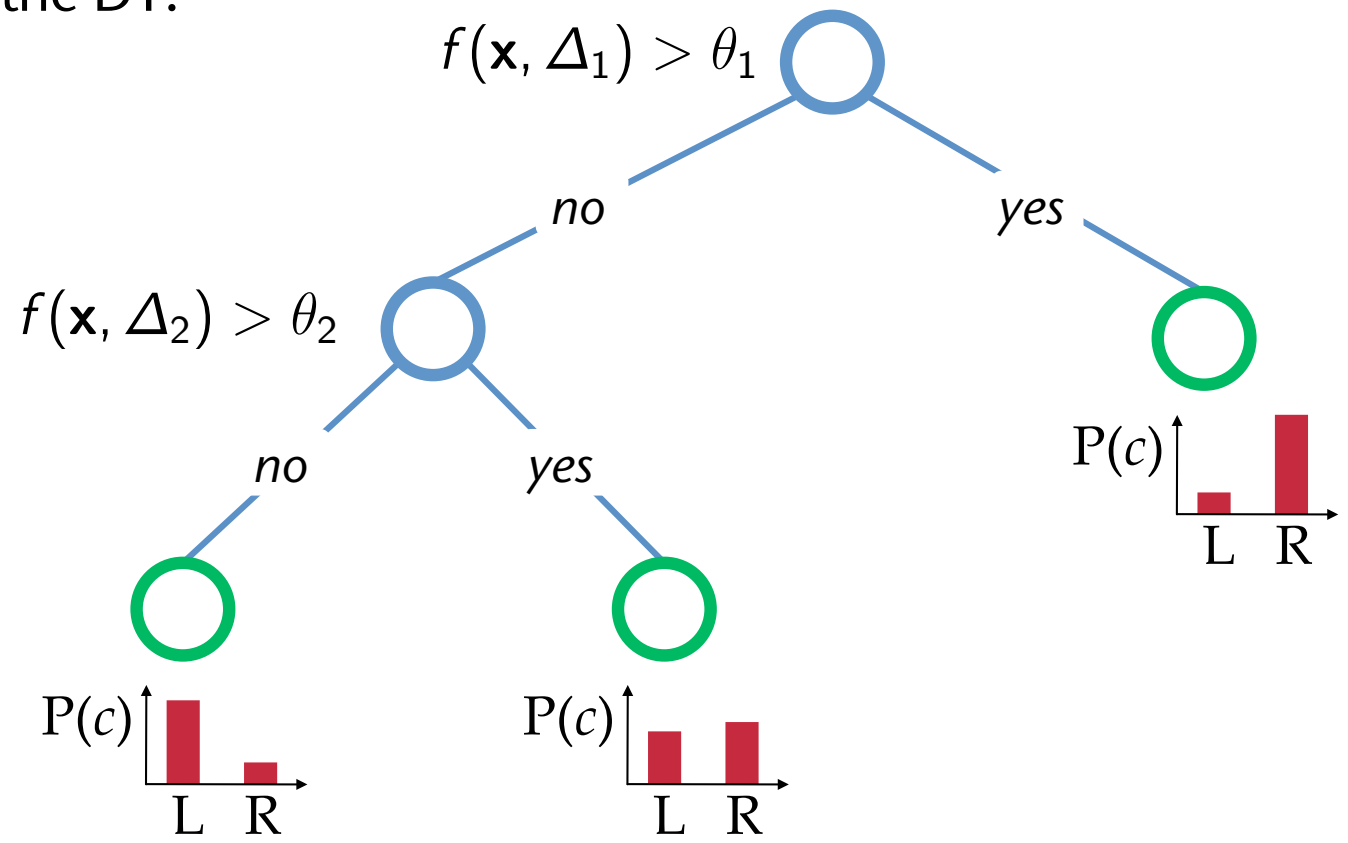
Training of a Single Decision Tree

- The training set $\mathcal{L}$ (conceptually) = all features (= all $f(\mathbf{x}, \Delta)$) of all pixels (= feature vectors) of all training images, together with the correct labels
- Training a decision tree amounts to finding that Δ and θ such that the information gain is maximized



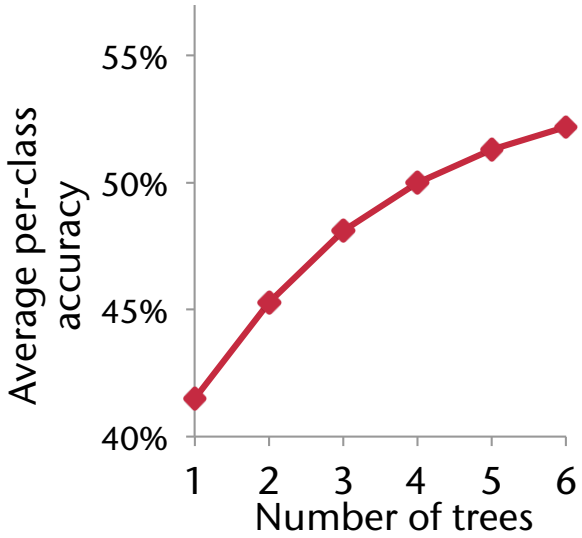
Classification of a Pixel at Runtime

- Toy example: distinguish left (L) and right (R) sides of the body
- Note: each node only needs to store Δ and θ!
- For every pixel **x** in the depth image, we traverse the DT:



Training a Random Forest

- Train *ntree* many trees, for each one introduce lots of randomization:
  - Random subset of pixels of the training images (~ 2000)
  - At each node to be trained, choose a random set of *mtry* many (Δ, θ) values
- Note: the complete feature vectors are never explicitly constructed (only conceptually)



ground truth

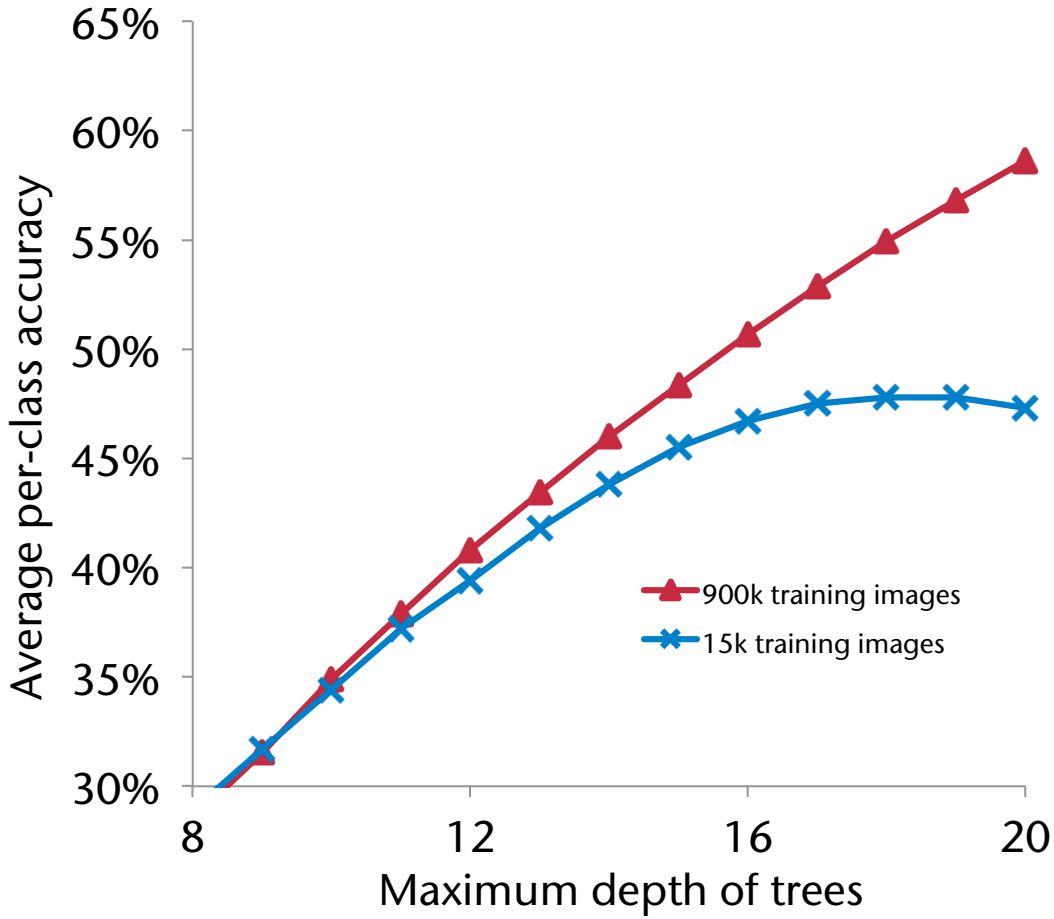
inferred body parts (most likely)

1 tree

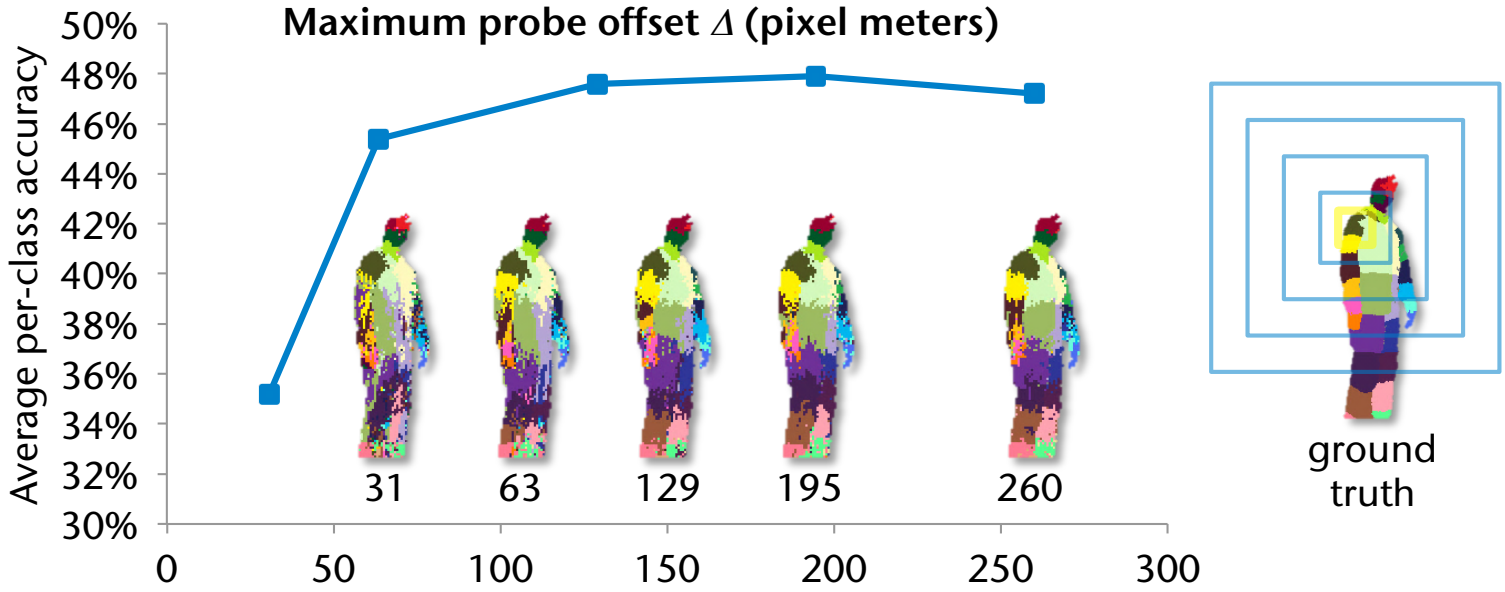
3 trees

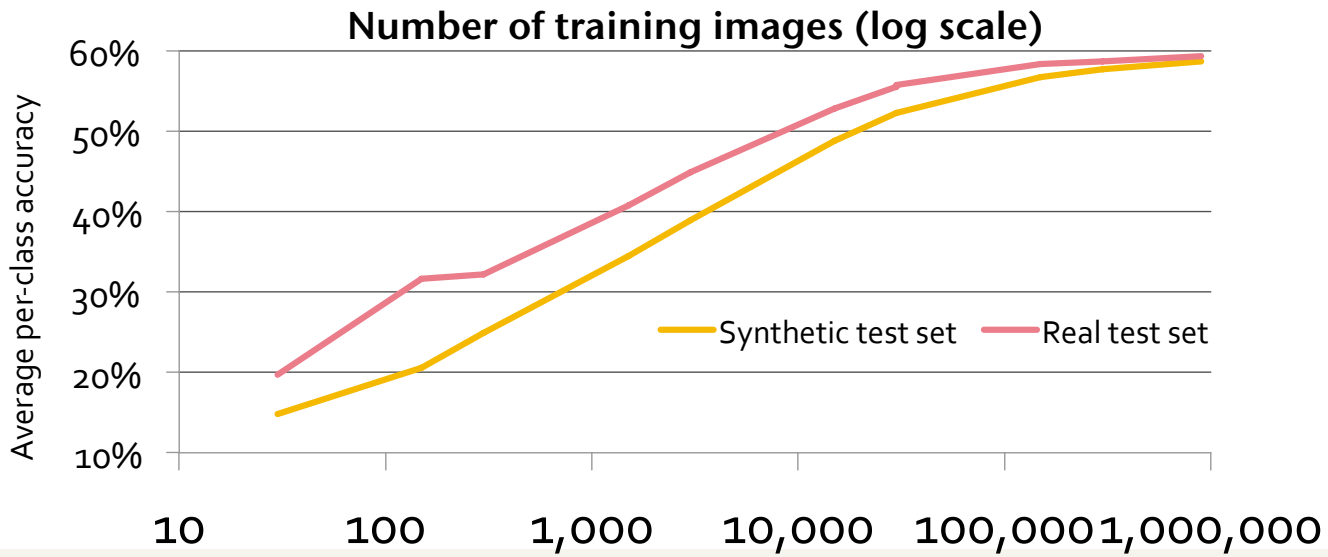
6 trees

- Depth of trees: check whether it is really best to grow all DTs in the RF to their maximum depth



More Parameters

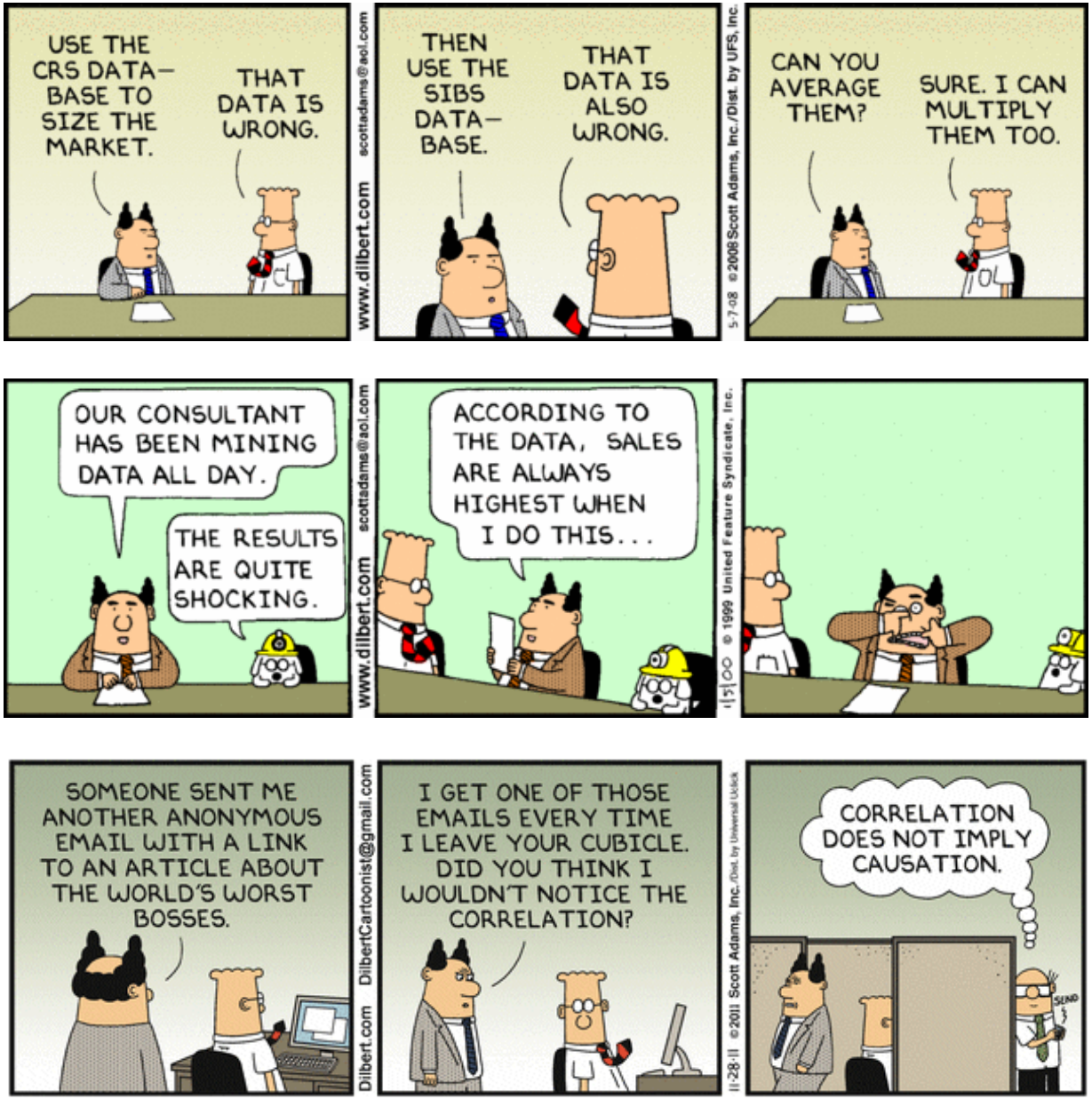




Video

Input depth image (bg removed)

Inferred body parts posterior


USE THE CRS DATA-BASE TO SIZE THE MARKET.
THAT DATA IS WRONG.
www.dilbert.com scottadams@aol.com
THEN USE THE SIBS DATA-BASE.
THAT DATA IS ALSO WRONG.
5-7-08 ©2008 Scott Adams, Inc./Dist. by UFS, Inc.
CAN YOU AVERAGE THEM?
SURE. I CAN MULTIPLY THEM TOO.
OUR CONSULTANT HAS BEEN MINING DATA ALL DAY.
THE RESULTS ARE QUITE SHOCKING.
www.dilbert.com scottadams@aol.com
ACCORDING TO THE DATA, SALES ARE ALWAYS HIGHEST WHEN I DO THIS...
1-5-00 © 1999 United Feature Syndicate, Inc.
SOMEONE SENT ME ANOTHER ANONYMOUS EMAIL WITH A LINK TO AN ARTICLE ABOUT THE WORLD'S WORST BOSSES.
Dilbert.com DilbertCartoonist@gmail.com
I GET ONE OF THOSE EMAILS EVERY TIME I LEAVE YOUR CUBICLE. DID YOU THINK I WOULDN'T NOTICE THE CORRELATION?
11-28-11 ©2011 Scott Adams, Inc./Dist. by Universal Uclick
CORRELATION DOES NOT IMPLY CAUSATION.
SEND