"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’s 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)
  - Correct answer: the non-Monkee is Roger Noll (a Stanford economist)
  - Now imagine a crowd of 100 people with knowledge distributed as:

    7 know all 3 of the Monkees

    10 know 2 of the Monkees

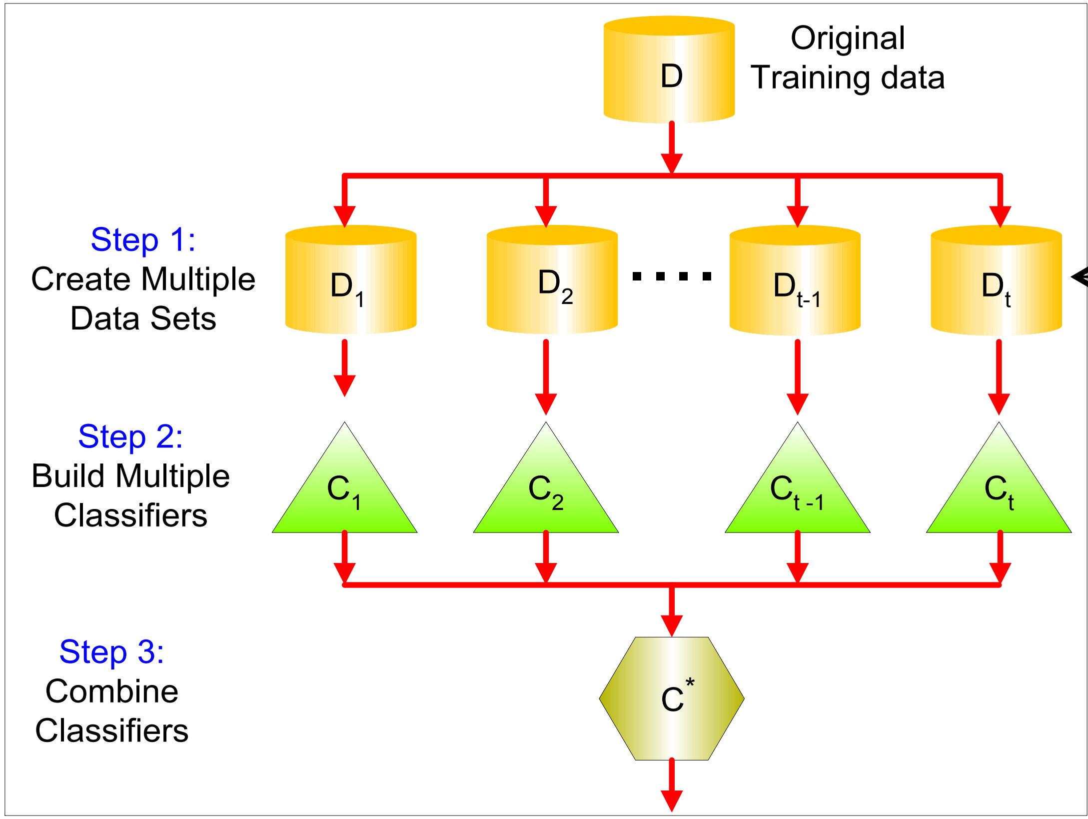
    15 know 1 of the Monkees

    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 ($\longrightarrow$ *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)



Must encode the same distribution as the orig. data set *D*!

Details on the Construction of Random Forests

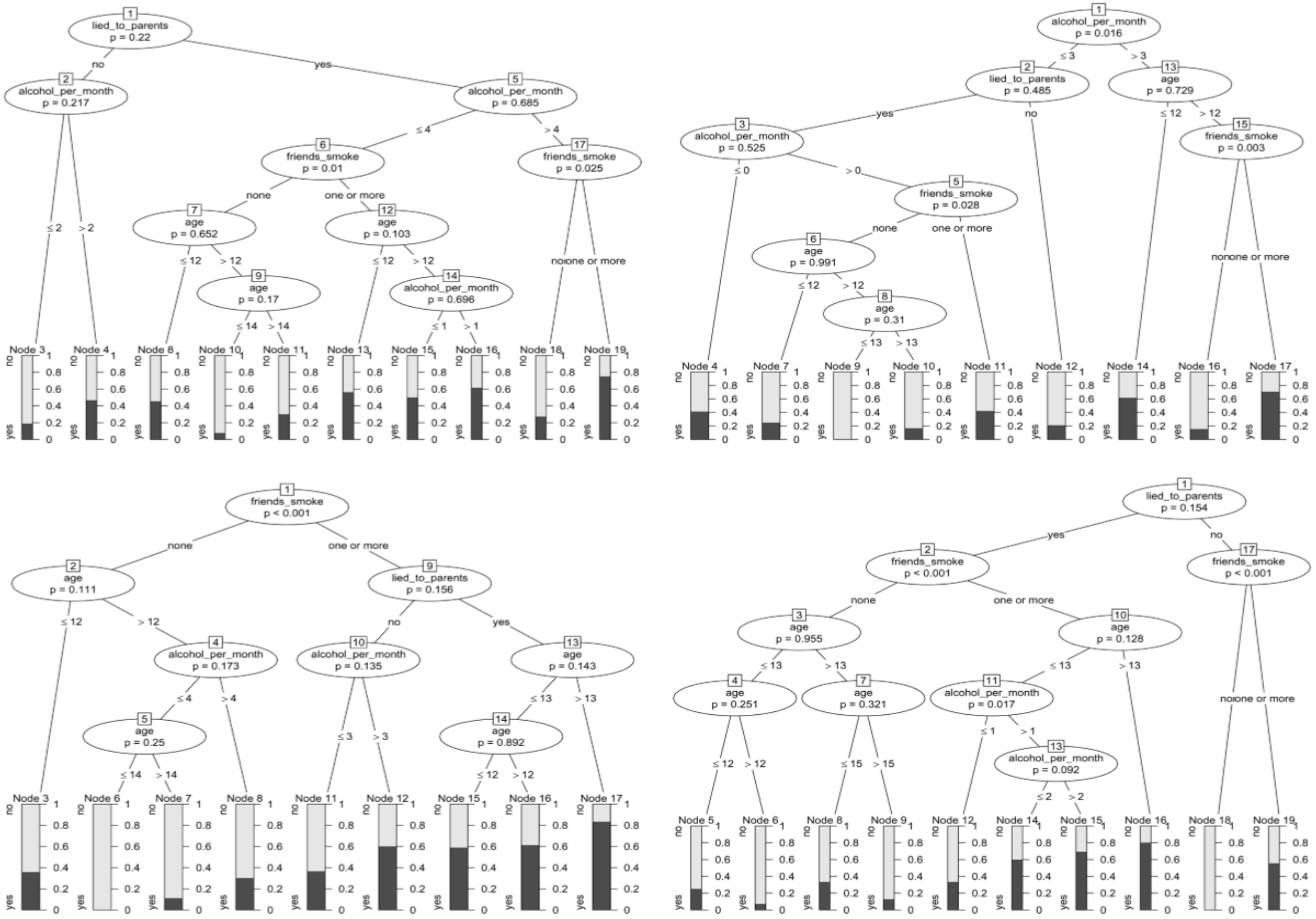
- Learning multiple trees:
  - Generate a number of data sets $\mathcal{L}_1, \mathcal{L}_2, \ldots$ from the original training data $\mathcal{L}$, $\mathcal{L}_i \subset \mathcal{L}$
  - Bootstrapping: randomly draw samples, with replacement, size of new data = size of original data set
  - Subsampling: randomly draw samples, without replacement, size of new data < size of original data set
  - Resulting trees can differ substantially (see earlier slide)
  - 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

- 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:

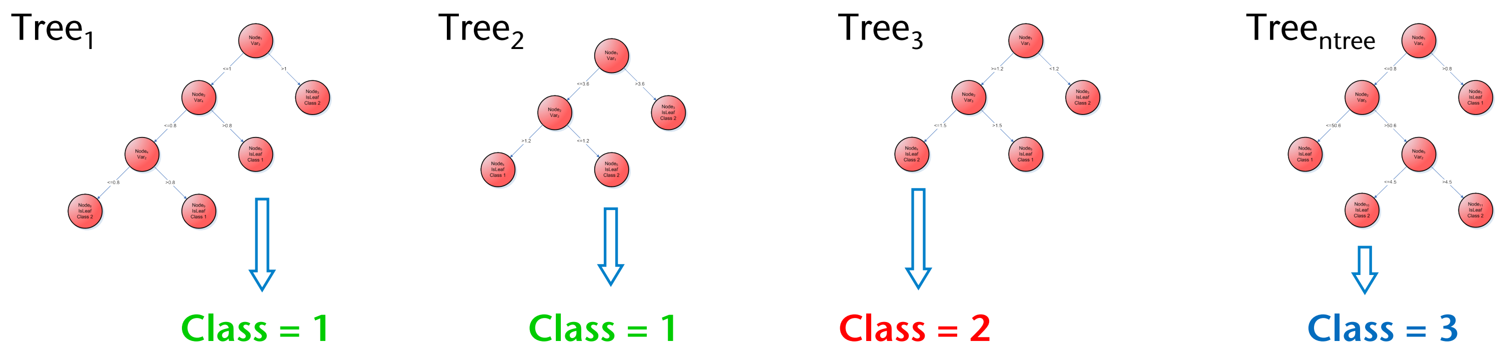
```
input: learning set L
for t = 1...ntree:
   build subset Lt from L by random sampling
   learn tree Tt from Lt:
      at each node:
         randomly choose mtry features
         compute best split from only those features
      grow each tree until leaves are perfectly pure
```

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



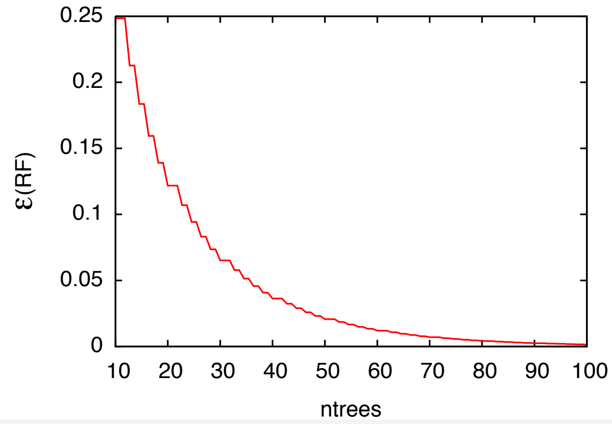
  - Take majority of those class labels
- Sometimes, if labels are numbers, (weighted) averaging makes sense

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:

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

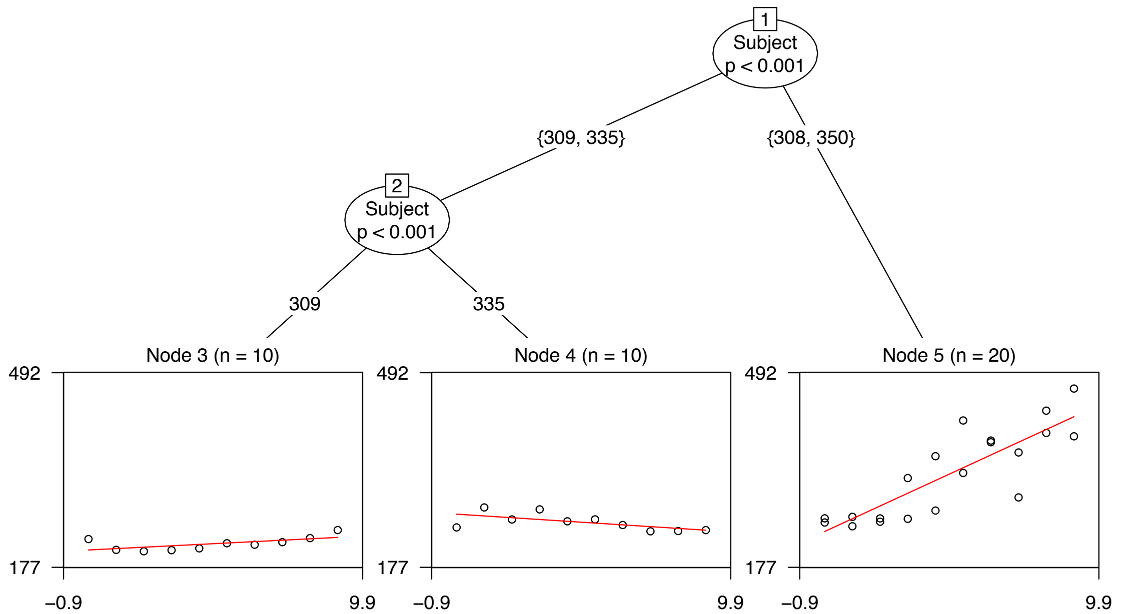
- Example: individual error rate
  $\varepsilon = 0.35 \longrightarrow$ error rate of RF
  $\varepsilon_{\text{RF}} \approx 0.01$



Variable Importance

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 $\mathbb{R}^d \to \mathbb{R}$ that generalizes the training data
  - Example:



Features and Pitfalls of Random Forests

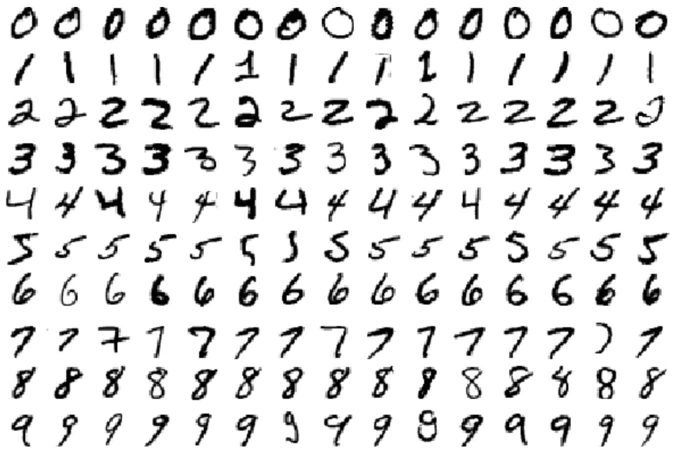
- "Small *n*, large *p*":
  - RFs are well-suited for problems with many more variables (dimensions in the feature space) than observations / training data
- Nonlinear function approximation:
  - RFs can approximate *any* unknown function
- Blackbox:
  - RFs are a black box; it is practically impossible 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:

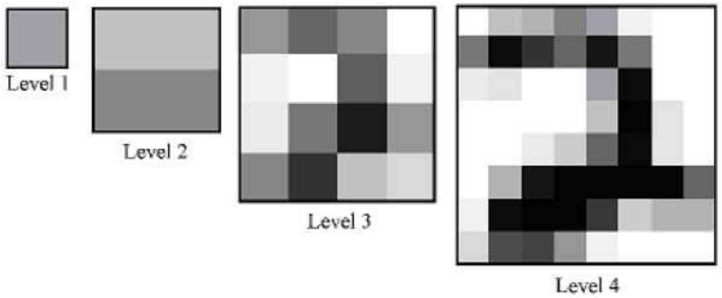
$$H(i,j) = I(i,j) \wedge I(i,j+2)$$
$$V(i,j) = I(i,j) \wedge I(i+2,j)$$
$$N(i,j) = I(i,j) \wedge I(i+2,j+2$$
$$S(i,j) = I(i,j) \wedge I(i+2,j-2$$

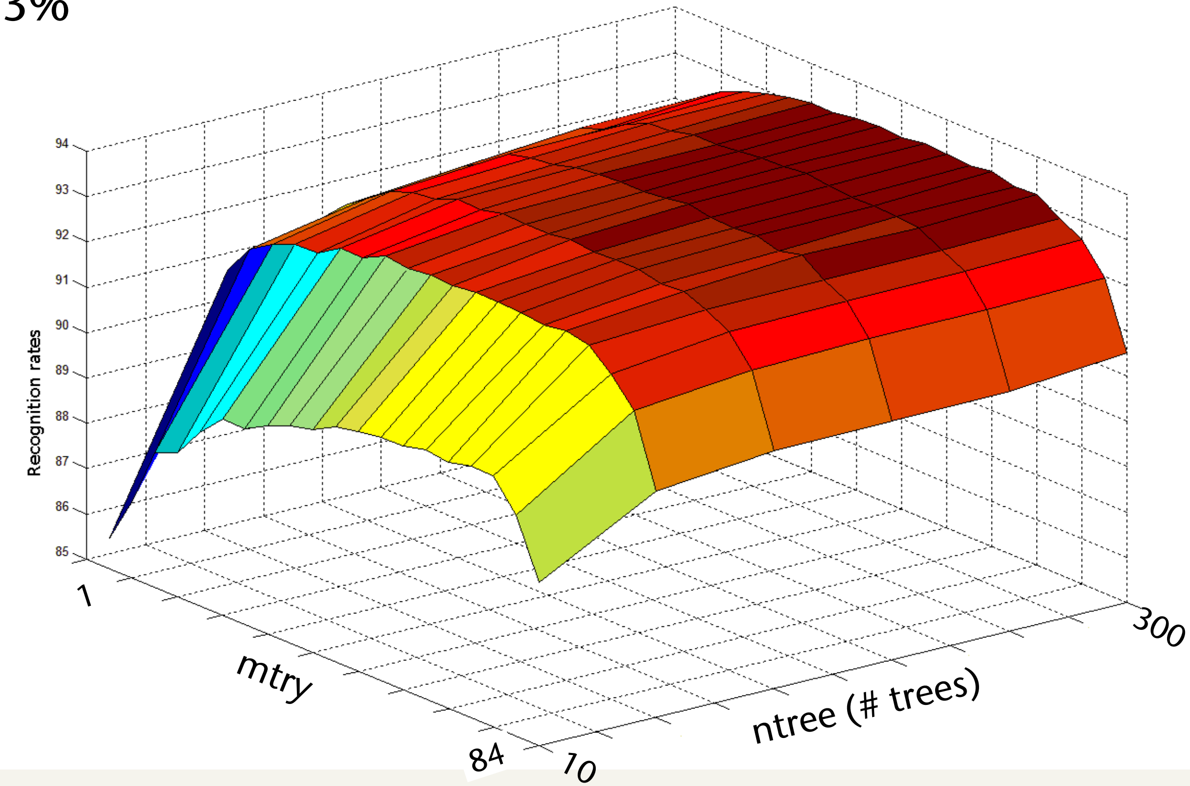
and a few more ...

- Feature vector for an image = (all pixels, all $H(i,j)$, $V(i,j)$, …)
- Feature space = 852-dimensional = 852 variables per data point

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

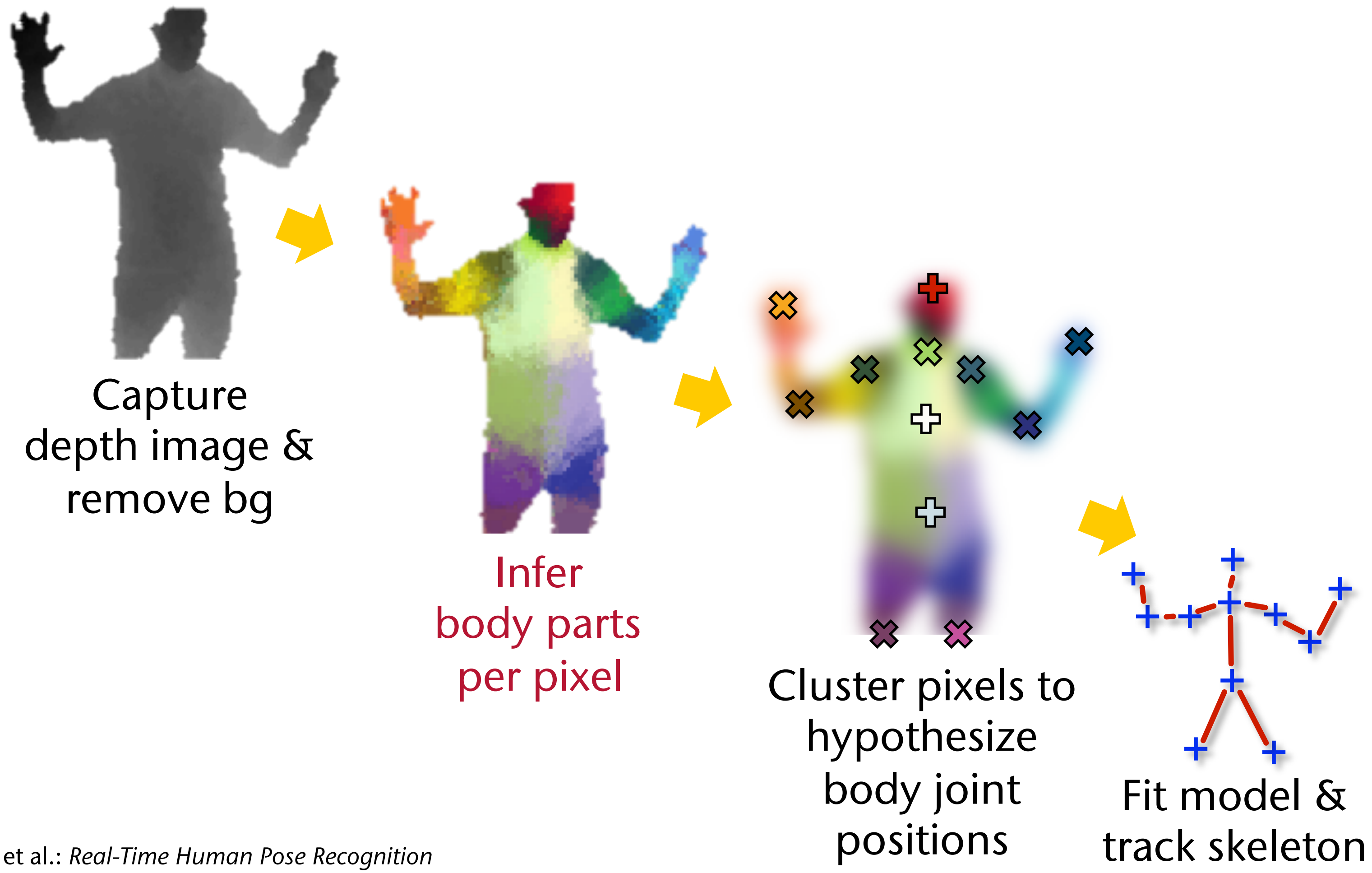
- Other experiments on handwritten digit recognition:
  - Feature vector = all pixels of an image pyramid
  - Recognition rate: ~ 93%
  - Dependence of recognition rate on *ntree* and *mtry*:





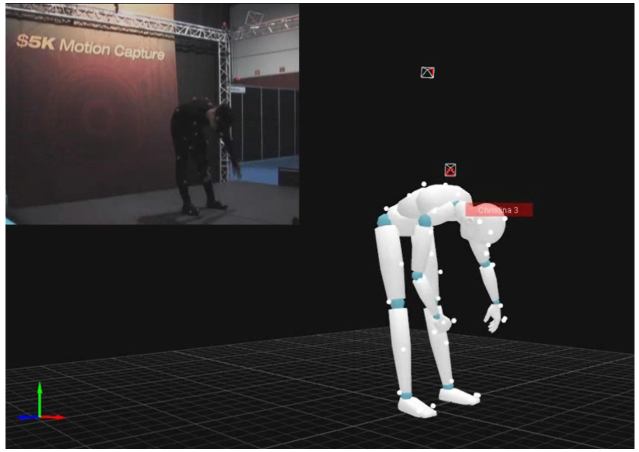
Body Tracking Using Depth Images (Kinect)

- The tracking / data flow pipeline:



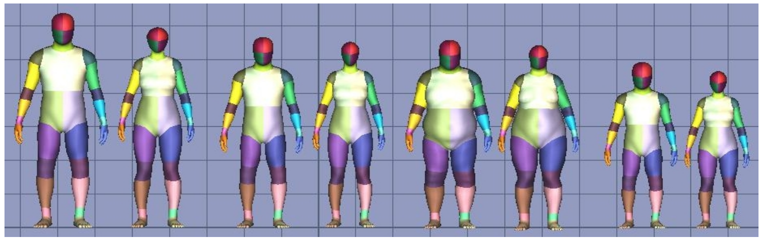
[Shotton et al.: *Real-Time Human Pose Recognition in Parts from Single Depth Images*; CVPR 2011]

The Training Data



Record mocap
500k frames
distilled to 100k poses

Retarget to several models



Render (depth, body parts) pairs

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 $\mathbf{x}$

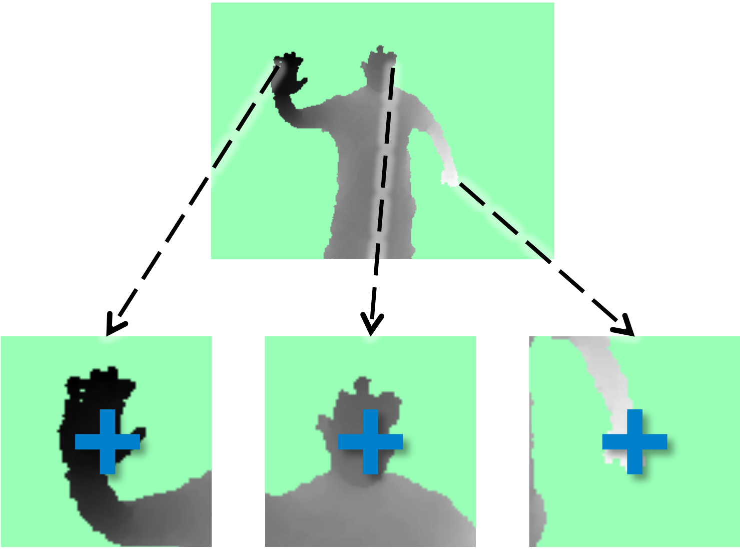
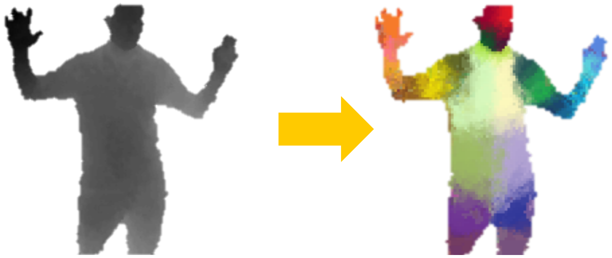


image windows move with classifier

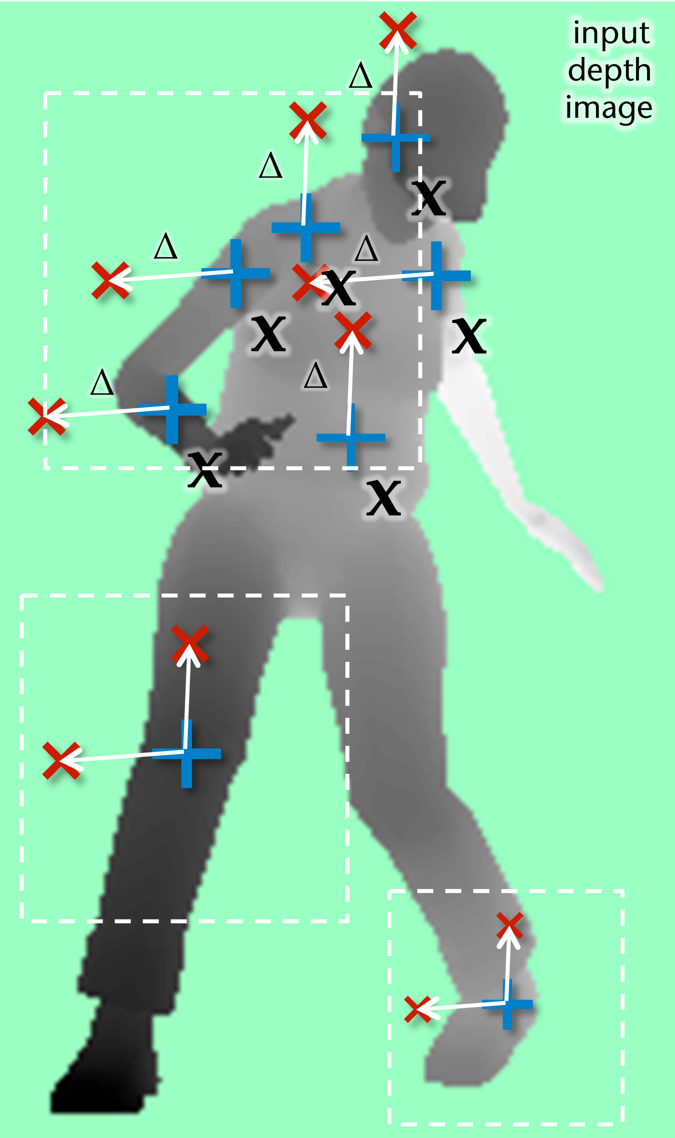


Fast Depth Image Features

- 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:

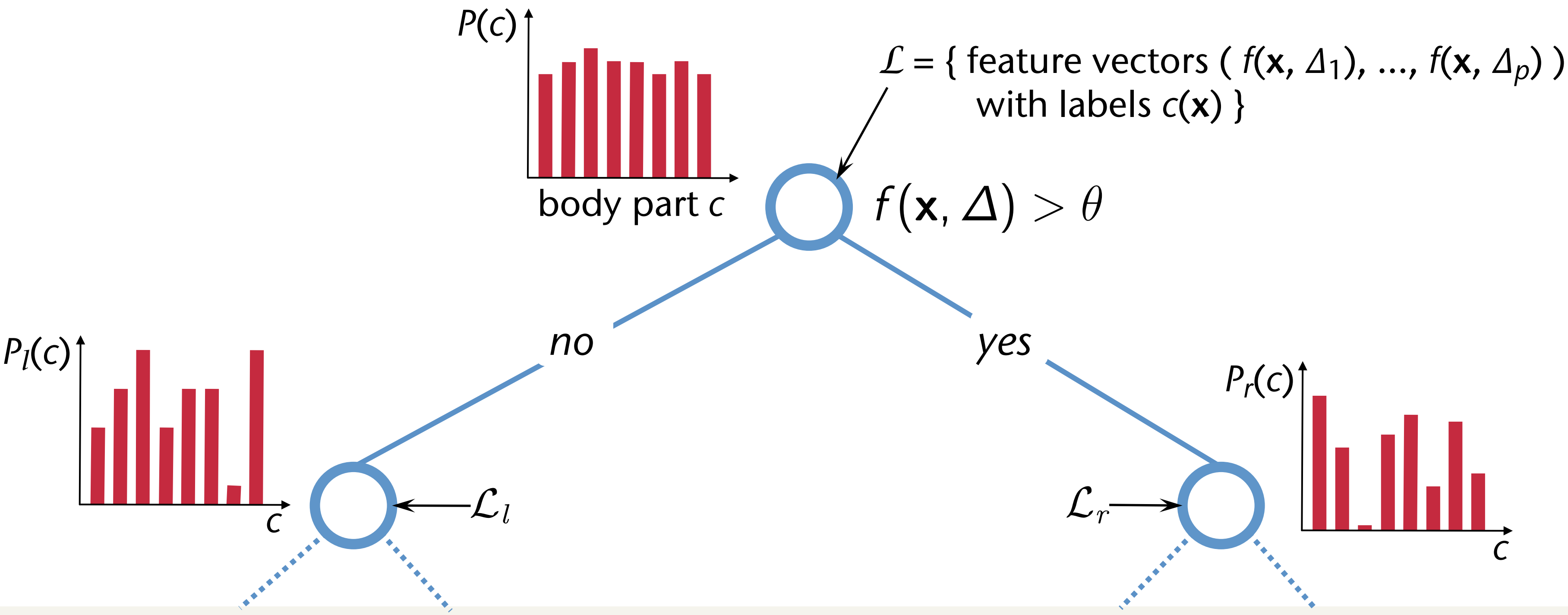
  $$f(\mathbf{x}, \Delta) = D(\mathbf{x}) - D\left(\mathbf{x} + \frac{\Delta}{D(\mathbf{x})}\right)$$

  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



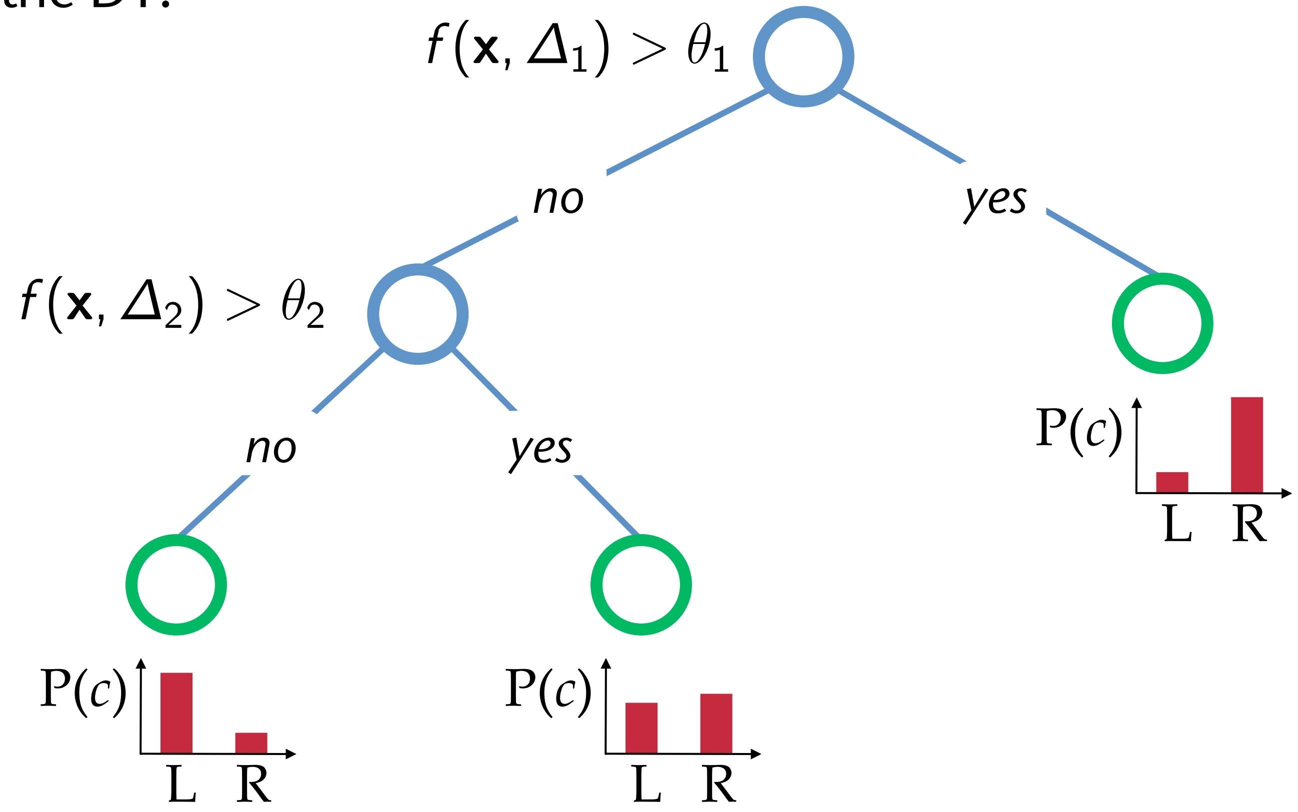
Training of a Single Decision Tree

- 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

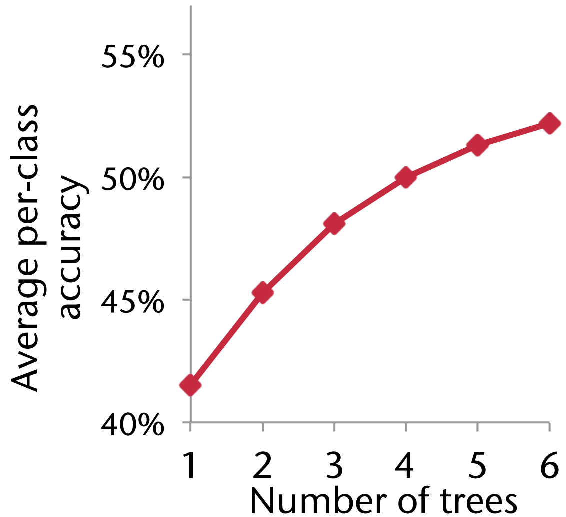
- Toy example: distinguish left (L) and right (R) sides of the body
- Note: each node only needs to store Δ and θ!
- For every pixel **x** in the depth image,
  we traverse the DT:



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 vector is never explicitly constructed (only conceptually)

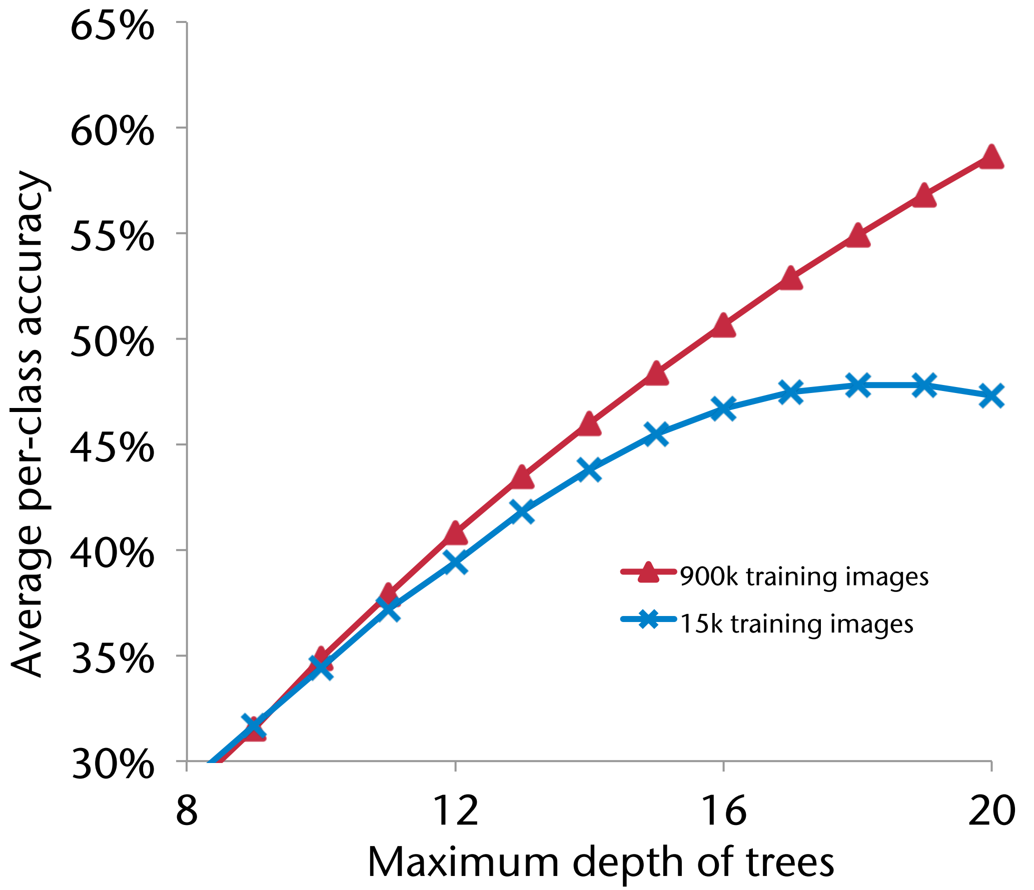
ground truth



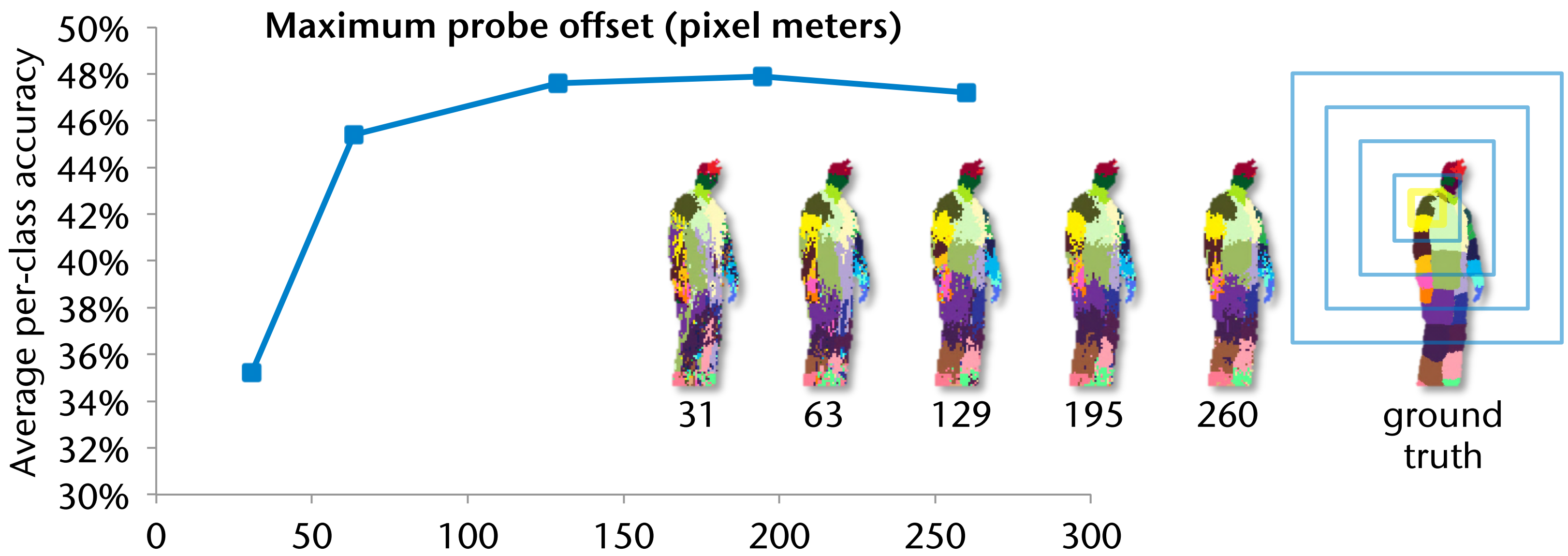
inferred body parts (most likely)

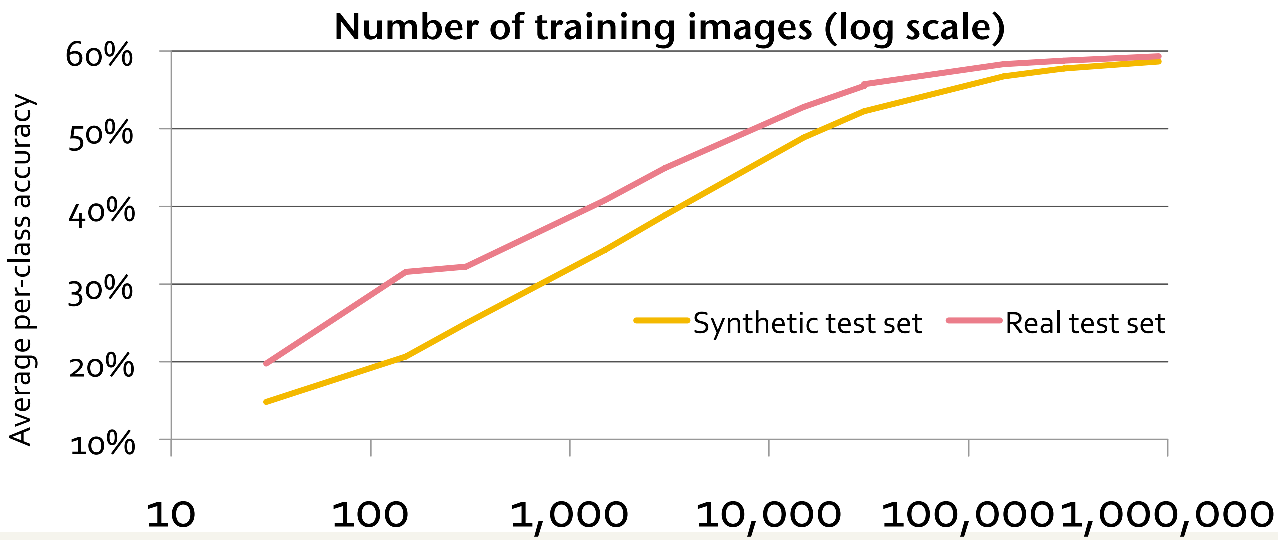
1 tree 3 trees 6 trees

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

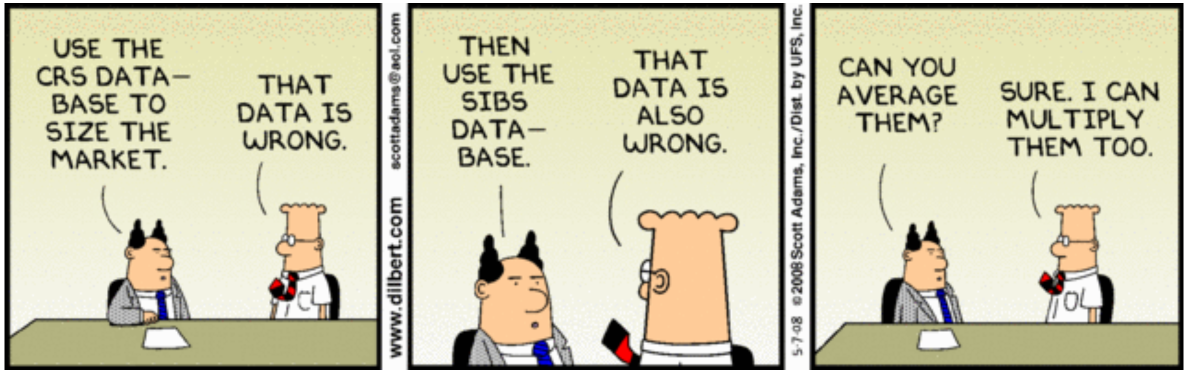


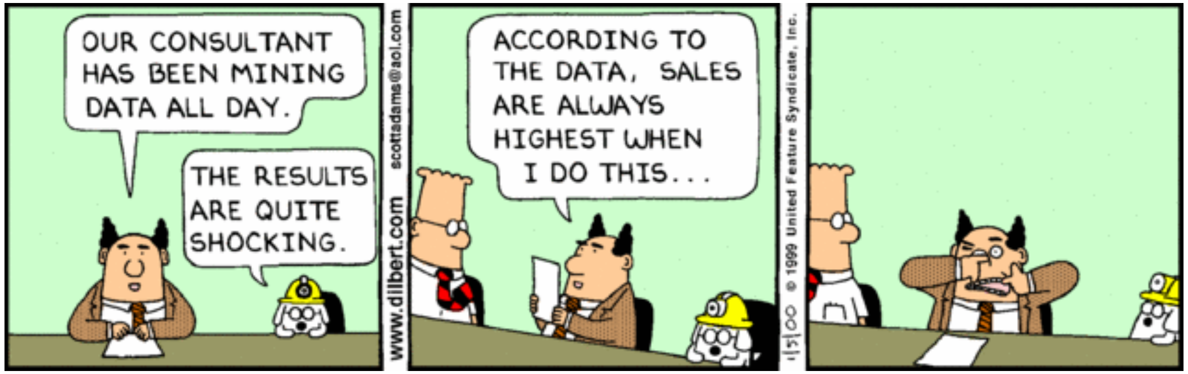
More Parameters





Implementing Decision Trees and Forests on a GPU - Sharp, ECCV 2008
Papers/Massively\ Parallel\ Algorithms/Random\ Forests


USE THE CRS DATA-BASE TO SIZE THE MARKET.
THAT DATA IS WRONG.
THEN USE THE SIBS DATA-BASE.
THAT DATA IS ALSO WRONG.
CAN YOU AVERAGE THEM?
SURE. I CAN MULTIPLY THEM TOO.
www.dilbert.com scottadams@aol.com
5-7-08 ©2008 Scott Adams, Inc./Dist. by UFS, Inc.


OUR CONSULTANT HAS BEEN MINING DATA ALL DAY.
THE RESULTS ARE QUITE SHOCKING.
ACCORDING TO THE DATA, SALES ARE ALWAYS HIGHEST WHEN I DO THIS...
www.dilbert.com scottadams@aol.com
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.
I GET ONE OF THOSE EMAILS EVERY TIME I LEAVE YOUR CUBICLE. DID YOU THINK I WOULDN'T NOTICE THE CORRELATION?
CORRELATION DOES NOT IMPLY CAUSATION.
SEND
Dilbert.com DilbertCartoonist@gmail.com
11-28-11 ©2011 Scott Adams, Inc./Dist. by Universal Uclick