Universitat
de les Illes Balears

Artificial Intelligence

Lesson 4.3: Tree models

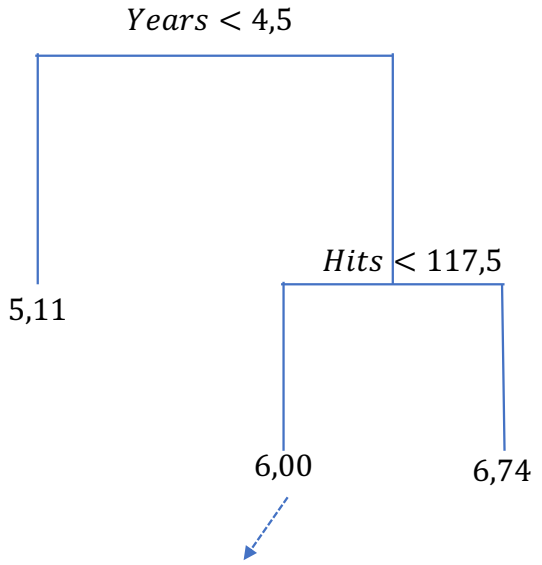
Decision Trees, Bagging Trees, Random Forest


FROM EXECUTIVE PRODUCERS
JONATHAN NOLAN AND LISA JOY
THE
PERIPHERAL
NEW SERIES
OCTOBER 21
prime video

Regression Trees

The figure shows a **regression tree** for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.

At a node, the label (of the form $x_j < t$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $x_j \geq t$.



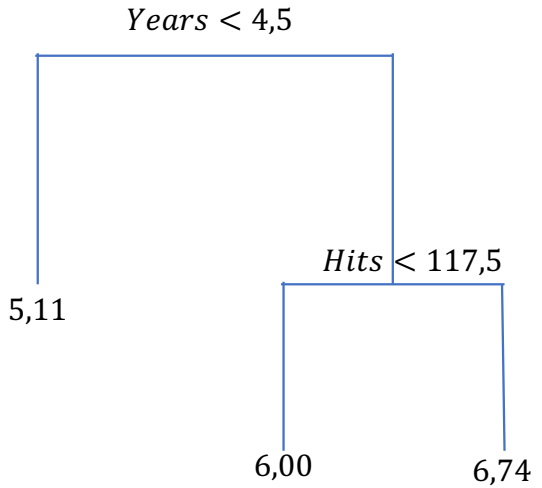
Leaf's value: mean of the response for the observations that fall there

Data: *hitters* (https://rdrr.io/cran/ISLR/man/Hitters.html)

Regression Trees

Explainable

Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players. Given that a player is less experienced, the number of hits that he made in the previous year seems to play little role in his salary. But among players who have been in the major leagues for five or more years, the number of ***hits*** made in the previous year does affect salary, and players who made more hits last year tend to have higher salaries.

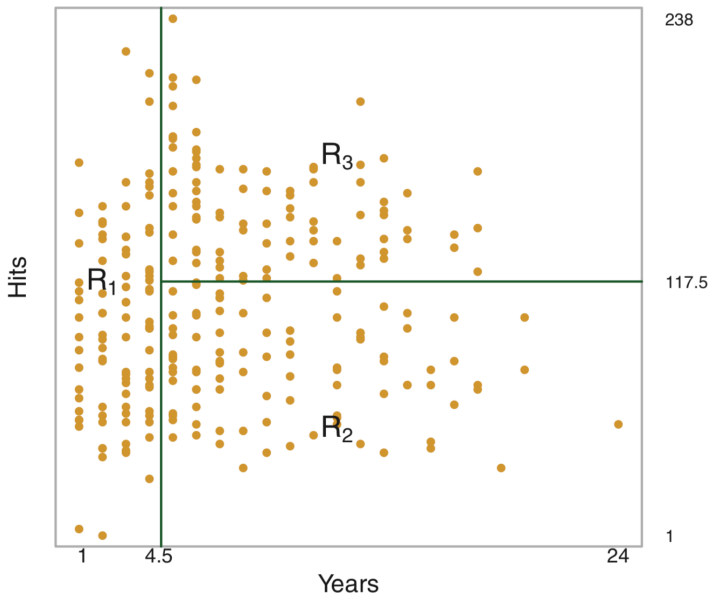


Regression Trees

Graphical representation

The tree stratifies or segments the players into three regions of predictor space: players who have played for four or fewer years, players who have played for five or more years and who made fewer than 118 hits last year, and players who have played for five or more years and who made at least 118 hits last year. These three regions can be written as

$$R_1 = \{x_j | years < 4.5\}$$
$$R_2 = \{x_j | years \geq 4.5, hits < 117.5\}$$
$$R_3 = \{x_j | years \geq 4.5, hits \geq 117.5\}$$



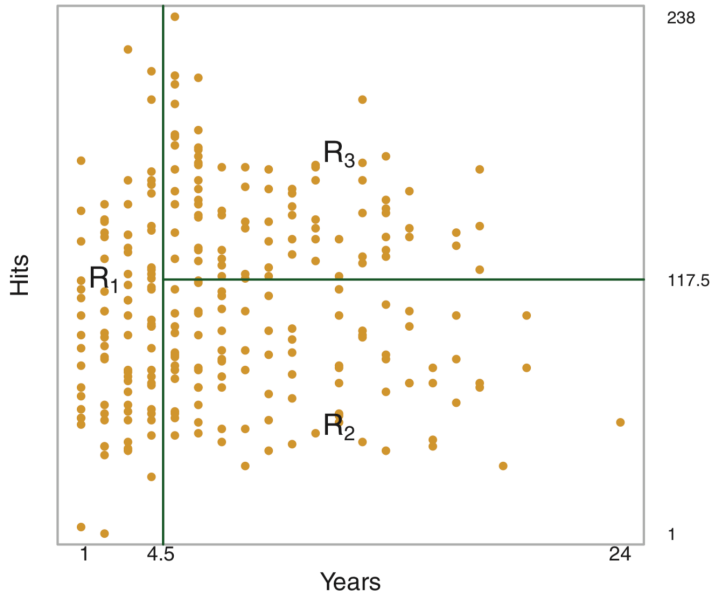
Regression Trees

In order to build a regression tree:

1. Divide the feature space into M distinct and non-overlapping regions:

$$R_1, R_2, R_3, \cdots, R_M$$

2. For every observation, $x^{(i)}$, that falls into the region R_m , we make the same prediction, which is simply the mean of the response values for the training observations in R_m $(\hat{y}_{R_m})$



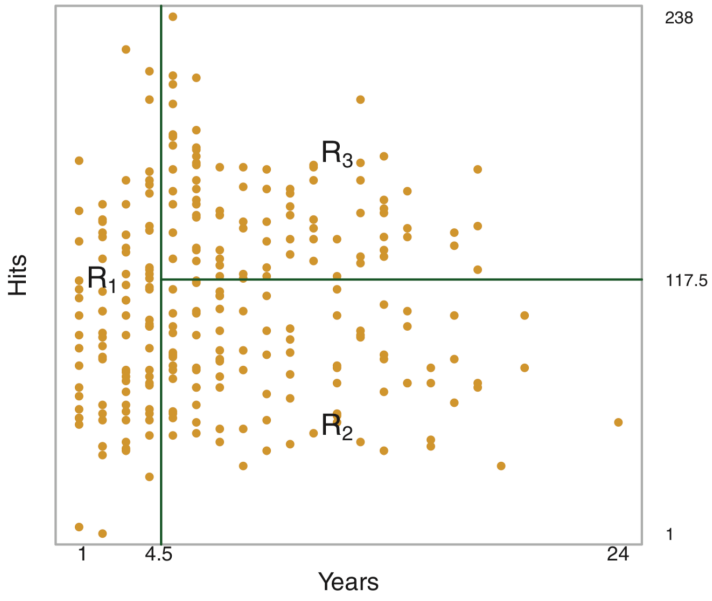
Regression Trees

How do we construct the regions R_m?

The goal is to find regions $R_1, R_2, R_3, \cdots, R_M$ that minimize the squared error, given by

$$\sum_{m=1}^{M} \sum_{i \in R_m} \left(y^{(i)} - \hat{y}_{R_m}\right)^2$$

It is computationally infeasible to consider every possible partition of the feature space into M regions



Recursive binary splitting

This top-down, greedy approach, begins at the top of the tree (a single region) and then successively splits the feature space.

It is **greedy** because at each step of the tree-building process, the best split is made at that particular step.

Repeat the process, in order to split the data further so as to minimize the error within each of the resulting regions.

The process continues until a stopping criterion is reached

Select the feature x_j and the cutpoint t such that splitting the feature space into the regions

$$R_1(j,t) = \{x | x_j < t\}, \quad R_2(j,t) = \{x | x_j \geq t\}$$

To seek the value of j and t that minimize the equation

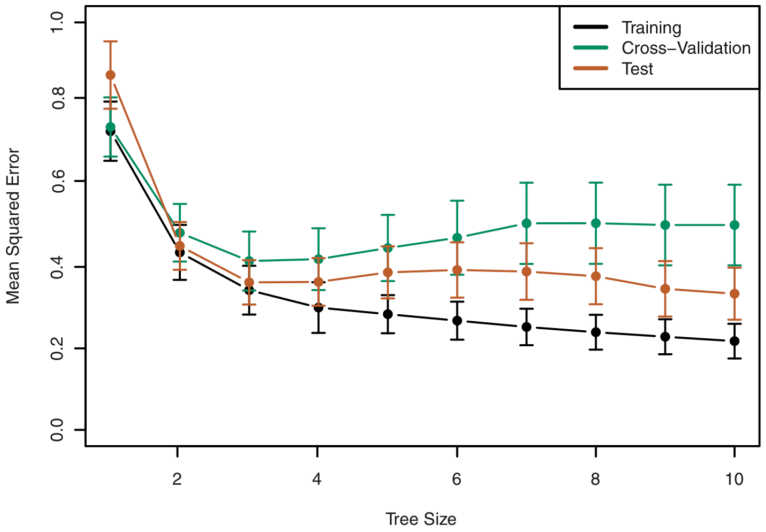
$$\sum_{i \in R_1} \left(y^{(i)} - \hat{y}_{R_1}\right)^2 + \sum_{i \in R_2} \left(y^{(i)} - \hat{y}_{R_2}\right)^2$$

Tree Pruning

The recursive binary splitting may produce good predictions on the training set, but is likely to **overfit** the data, leading to poor test set performance.

A strategy is to grow a very large tree, and then prune it back in order to obtain a subtree. The goal is to select a subtree that leads to the lowest test error rate.

Given a subtree, we can estimate its test error using **cross-validation** & weakest link pruning.



Classification Trees

For a **classification tree**, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.

Therefore, in interpreting the results, we are often interested not only in the class prediction corresponding to a particular terminal node region, but also in the class proportions among the training observations that fall into that region.

An example:

Develop an AI application that advises whether we should wait for a table at a restaurant or not.

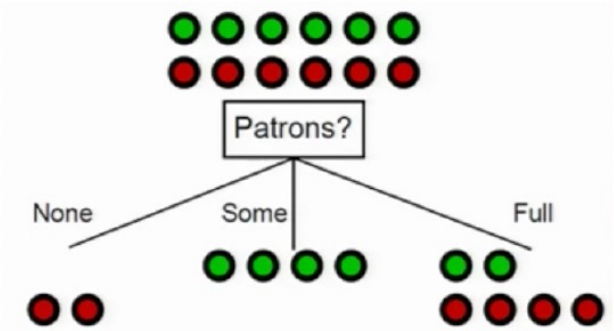
An example

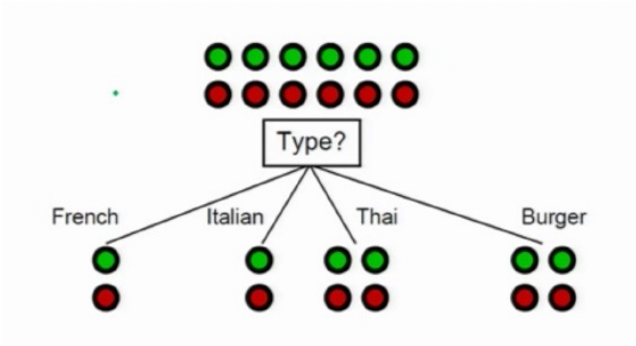
1. **Alternate**: whether there is a suitable alternative restaurant nearby.
2. **Bar**. whether the restaurant has a comfortable bar area to wait in.
3. **Fri/Sat:** true on Fridays and Saturdays.
4. **Hungry**: whether we are hungry.
5. **Patrons**: how many people are in the restaurant (values are None, Some, and Full).
6. **Price**: the restaurant's price range ($, $$, $$$).
7. **Raining**: whether it is raining outside.
8. **Reservation**: whether we made a reservation.
9. **Type**: the kind of restaurant (French, Italian, Thai, or Burger).
10. **WaitEstimate**: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Classification: which feature?

Which feature would you pick? Why?





Which feature is more informative?

Classification Trees: error measures

The squared error cannot be used as a criterion for making the binary splits.

In practice, two other measures are preferable:

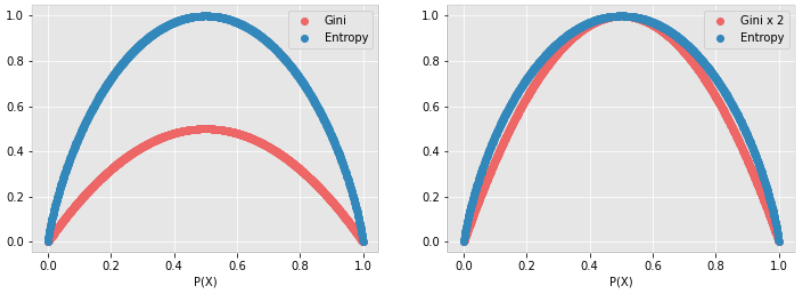
- Entropy

$$E = -\sum_{c=1}^{C} p_{mc} \log p_{mc}$$

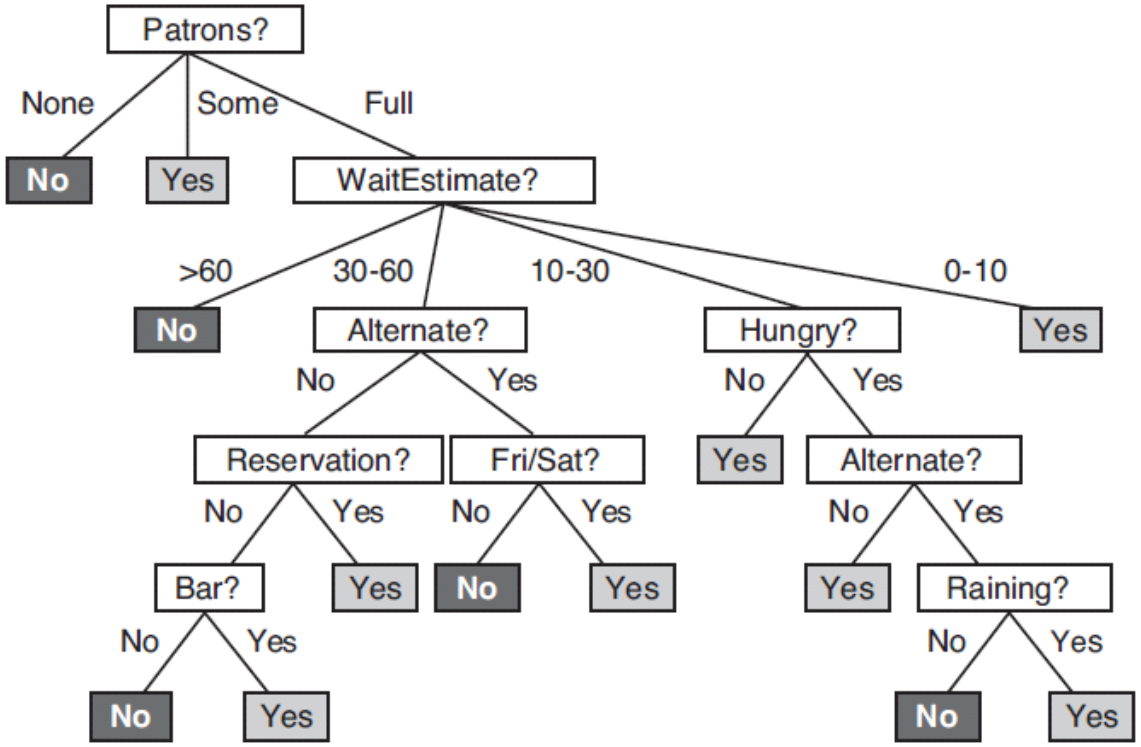
- Gini index

$$G = \sum_{c=1}^{C} p_{mc}(1 - p_{mc})$$

Here p_{mc} represents the proportion of training observations in the region R_m that are from the class c.



Exxample: learning result

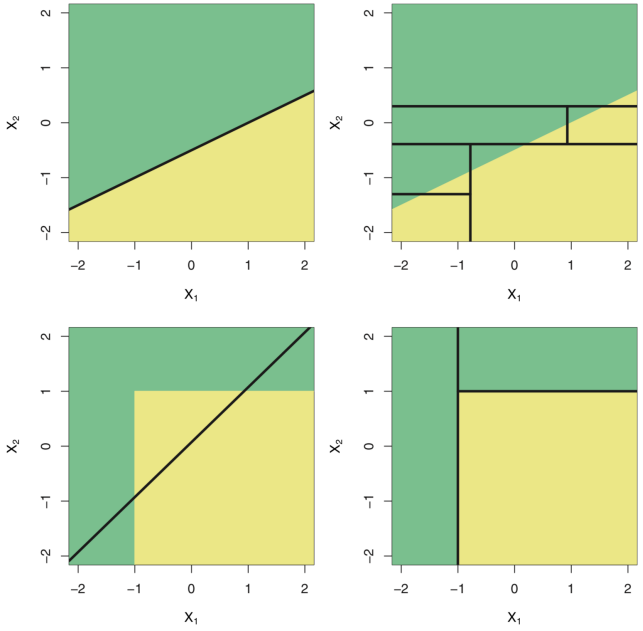


CART vs Linear models

Prediction functions:

$$h_\theta(x) = g\left(\sum_{j=0}^{n} \theta_j x_j\right)$$

$$h_t(x) = \hat{y}_{R_m} \cdot 1\{\mathrm{x} \in R_m\}$$



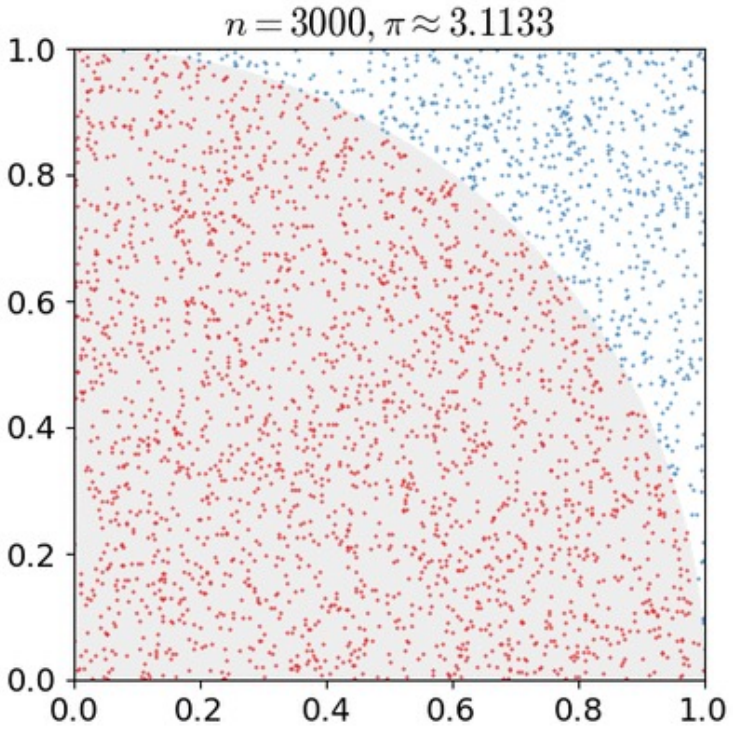
Which model is better?

It depends on the problem at hand: Linear vs Non-linear feature space

Advantages and Disadvantages

- Machine performance: fast learning & prediction algorithms (Real-time)
- Bias: Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the data set prior to fitting with the decision tree
- High Variance: Tends to overfitting and they can be unstable because small variations in the data might result in a completely different tree being generated
- Human: decision trees more closely mirror human decision-making
- Explainable; Trees are very easy to explain to people

Monte Carlo method



Bagging (bootstrap aggregation)

CART suffers of high variance… however by means of **bootstrapping** we can reduce the variance by taking many training sets from the dataset, build a separate model using each training set, and average the resulting predictions.

To apply **bagging** to regression trees, we simply construct **T** regression trees using **T** bootstrapped training sets, and average the resulting predictions.

These trees are grown deep, and are not pruned. Hence each individual tree has high variance, but low bias. Averaging these **T** trees reduces the variance.

Calculate, $h_1(x), h_2(x), \ldots, h_T(x)$ using T separate training sets, and average them in order to obtain a single low-variance statistical learning model

$$h(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

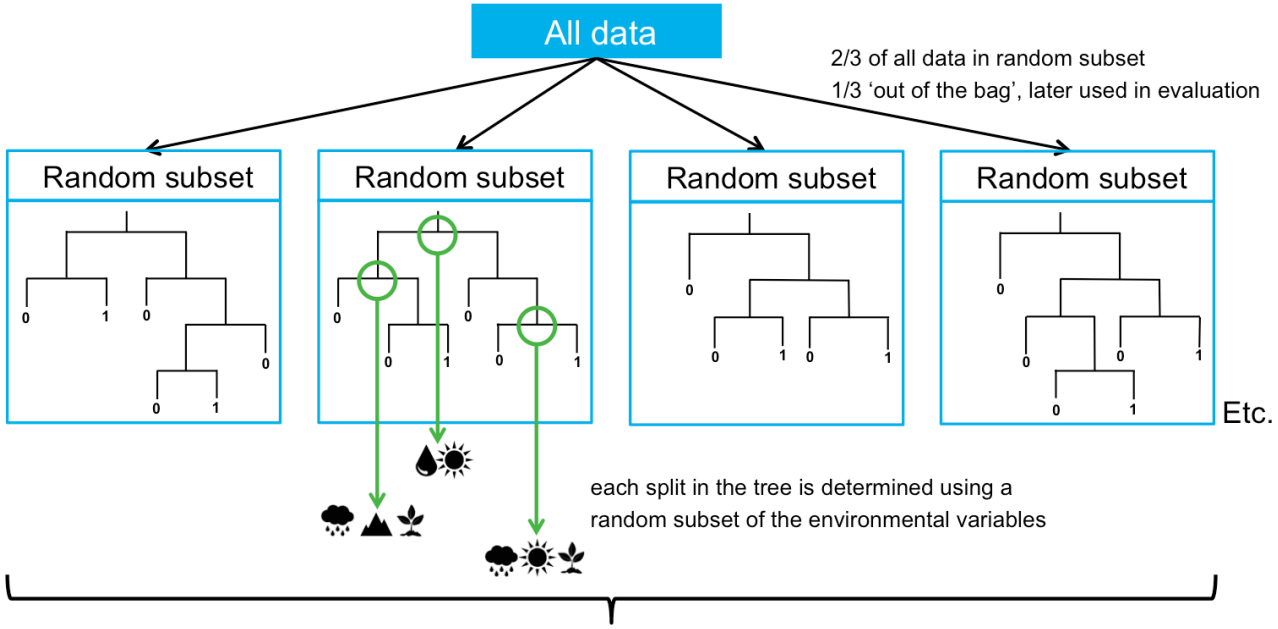
Quiz: how it works for classification trees?

Random Forest

Random Forest provide an improvement over bagged trees by way of a small tweak that decorrelates the trees:

"when building the decision trees in a bagging approach, each time a split in a tree is considered, a random sample of features is chosen as split candidates from the full set of *p* predictors. The split is allowed to use only one of those features. A fresh sample of features is taken at each split (typically we choose $\sqrt{n}$)"

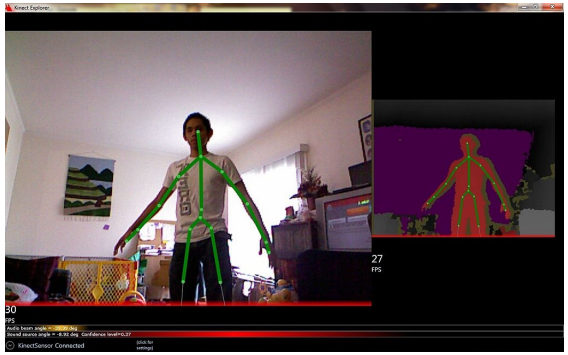
Random Forest



> find the set of predictor variables that produce the strongest classification model

Application: Project Natal

https://youtu.be/p2qlHoxPioM



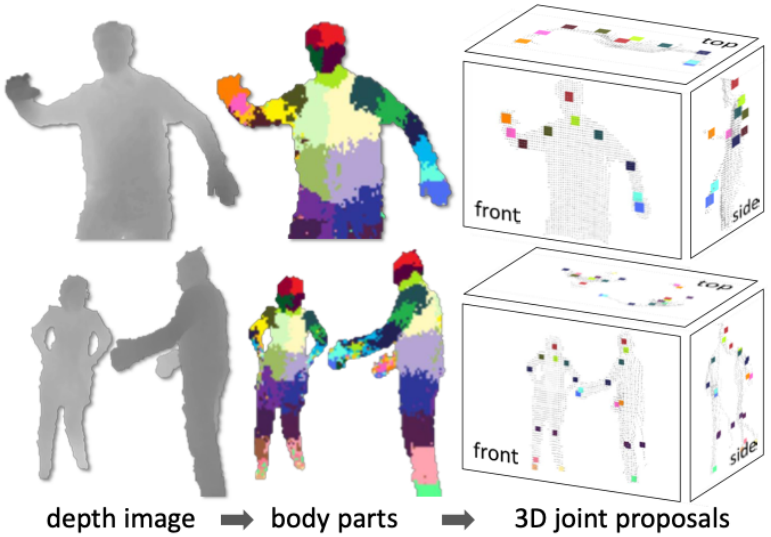
Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton Andrew Fitzgibbon Mat Cook Toby Sharp Mark Finocchio
Richard Moore Alex Kipman Andrew Blake
Microsoft Research Cambridge & Xbox Incubation

Abstract

We propose a new method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information. We take an object recognition approach, designing an intermediate body parts representation that maps the difficult pose estimation problem into a simpler per-pixel classification problem. Our large and highly varied training dataset allows the classifier to estimate body parts invariant to pose, body shape, clothing, etc. *Finally we generate confidence-scored 3D proposals of several body joints by reprojecting the classification result and finding local modes.*

The system runs at 200 frames per second on consumer



https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf

Model design

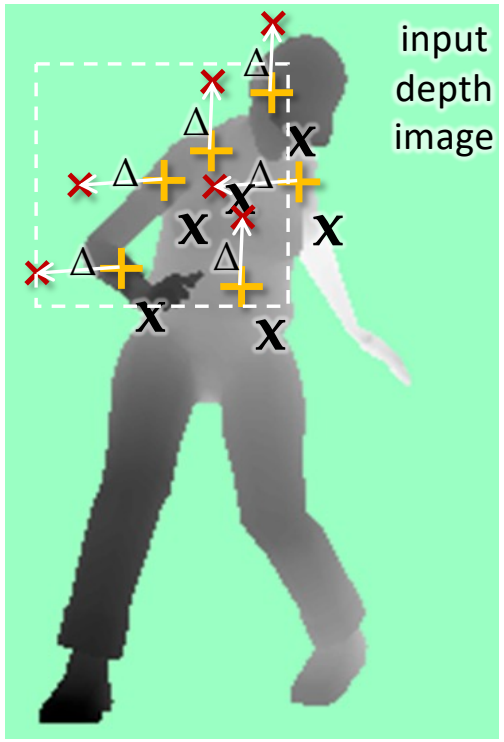
Training:

- 3 trees,
- 20 deep,
- 300k training images per tree,
- 2000 training example pixels per image,
- 2000 candidate features, and
- 50 candidate thresholds per feature.

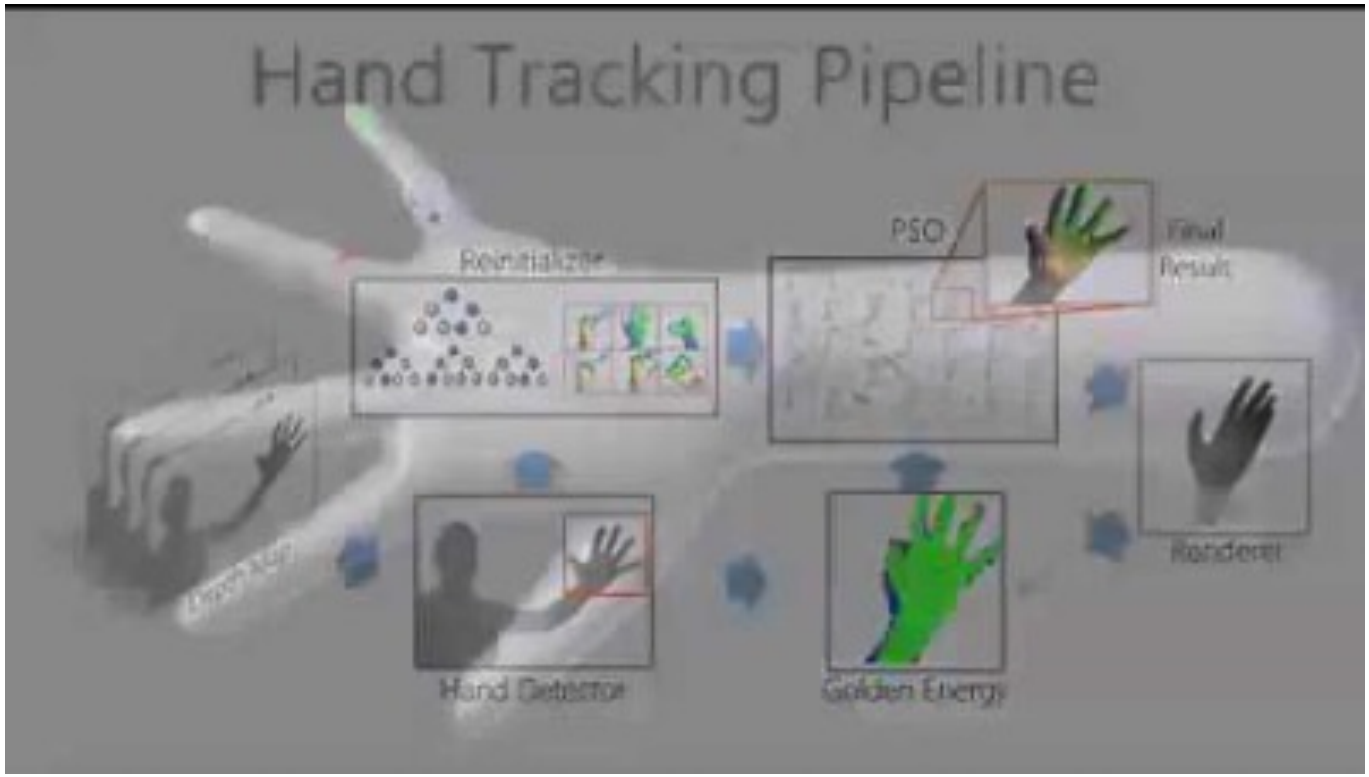
Prediction: 5ms.

https://youtu.be/IntbRsi8lU8

Features



An extension: 3D hand tracking



https://www.youtube.com/watch?v=ESvyoVEwPrc