ECON 480-3
LECTURE 11: A PRIMER ON RANDOM FORESTS

Ivan A. Canay
Northwestern University

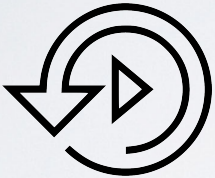
Past & Future

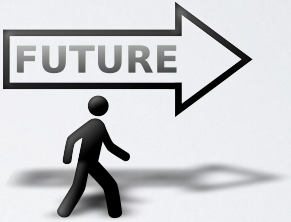
LAST CLASS

- The Regression Discontinuity Design
- Sharp and Fuzzy RDD
- Bandwidth Choice
- Matching Estimators

TODAY

- Regression Tress
- Classification Tress
- Random Forests





Leo Breiman, 1928-2005

- 1954: PhD Berkeley (mathematics)
- 1960-1967: UCLA (mathematics)
- 1969-1982: Consultant
- 1982-1993: Berkeley (statistics)
- 1984: Classification & Regression Trees (with Friedman, Olshen, Stone)
- 1996: Bagging
- 2001: Random Forests

Setup

- Let (Y, X) be a random vector where $Y \in \mathbf{R}$ and $X \in \mathbf{R}^k$.
- Start focusing on $k = 2$ for simplicity and $X \in [0, 1]^2$.
- Let P be the distribution of (Y, X).
- We are interested in the **conditional mean** of Y given X.

$$g(x) = E[Y|X = x] \ .$$

- Let $\{(Y_1, X_1), \ldots, (Y_n, X_n)\}$ be an i.i.d. sample from P.
- **Today**:
  - Regression Trees
  - Bagging
  - Random Forests
- When Y is **discrete**: classification trees are more appropriate (CART)

- Tree-based methods partition the X space into a set of **rectangles**.
- Let's denote these **rectangles** by
$$\{R_m : 1 \leqslant m \leqslant M\}\,.$$
- They then fit a very simple model: usually, **a constant**,
$$\hat{g}(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\}\,. \tag{1}$$
- With **continuous** Y, c_m is usually the average of Y conditional on $X \in R_m$,
$$c_m = \frac{\sum_{i=1}^{n} Y_i I\{X_i \in R_m\}}{\sum_{i=1}^{n} I\{X_i \in R_m\}}\,.$$
- **Main issue:** How to find "good" rectangles R_m.

Two Partitions of X

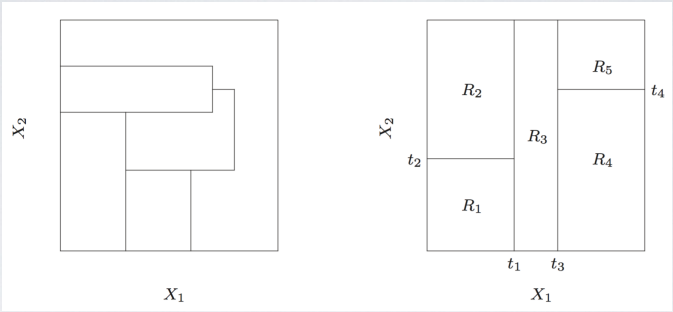


FIGURE: $X = (X_1, X_2) \in [0,1]^2$. Left: generic partition. Right: Tree partition

- **Left Panel**
  - Each partition line has a simple description, $X_1 = c$
  - However, resulting regions are hard to describe
- **Right Panel**
  - Arise from recursive binary partitions
  - First split $X_1 = t_1$. Then, for the region $X_1 \leqslant t_1$, we split at $X_2 = t_2$ and the region $X_1 > t_1$ is split at $X_1 = t_3$. Etc...

Recursive Binary Partitions: Trees

- **Trees**: split the space of X into recursive binary partitions.
- Terminal nodes, or "leaves", correspond to the regions $R_1, \ldots, R_m$.
- Key advantage $\Rightarrow$ interpretability
  - Partition of X fully described by a single tree.
  - In higher dimensions regions are hard to describe, but "trees" are alway easy.

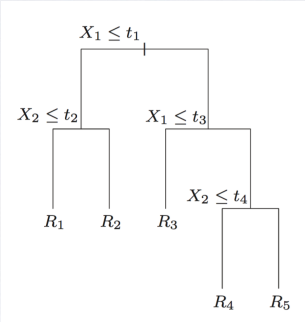


FIGURE: Tree in the previous example

Example: California Housing

- **Data**: each of 20,460 neighborhoods (1990 census block groups) in California.
- **Response variable**: Y is the median house value in each neighborhood.
- The are a total of eight predictor variables (or covariates)
  - Median income of neighborhood
  - Median house age
  - Housing features: average number of rooms and bedrooms.
  - Housing density: number of houses
  - Average occupancy in each house
  - Location of each neighborhood (longitude and latitude)

Example: House Prices

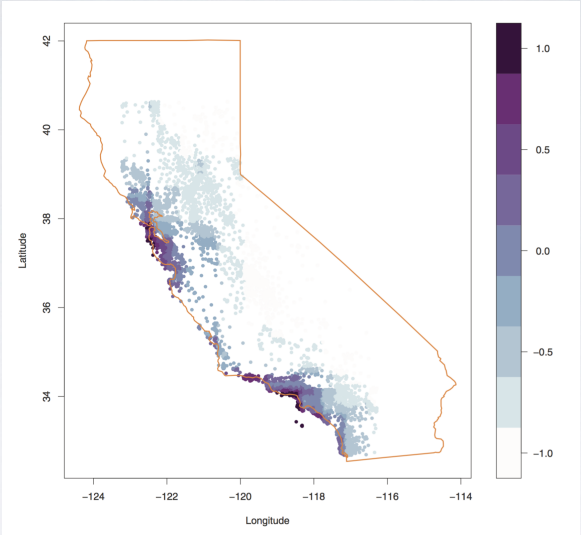


FIGURE: California housing prices (relative to median)

Example: California Housing

- **Dataset**: median house prices by location in California (Longitude and latitude).

```
  MedianHouseValue MedianIncome MedianHouseAge Latitude Longitude
1           452600       8.3252             41    37.88   -122.23
2           358500       8.3014             21    37.86   -122.22
3           352100       7.2574             52    37.85   -122.24
4           341300       5.6431             52    37.85   -122.25
```

- **Goal**: grow a regression tree as a function of geographic coordinates.

- R has several packages for trees: `Tree` being a simple one.

```
require(tree)
calif = read.table("cadata.dat",header=TRUE)
treefit = tree(log(MedianHouseValue) ~ Longitude+Latitude)
plot(treefit)
text(treefit,cex=0.75)
```

Example: House Prices

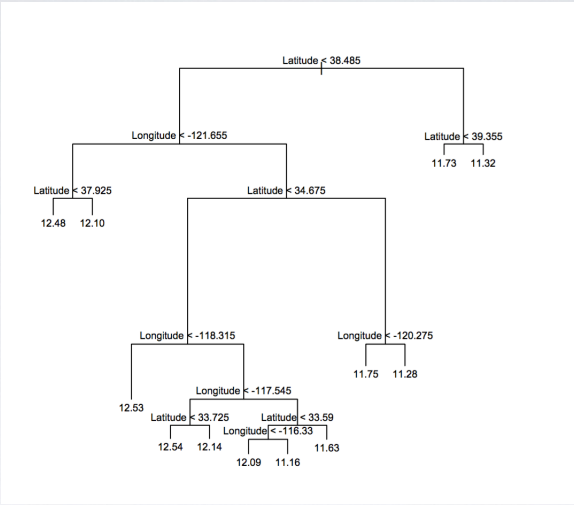


FIGURE: Regression tree for predicting California housing prices from geographic coordinates.

Example: House Prices

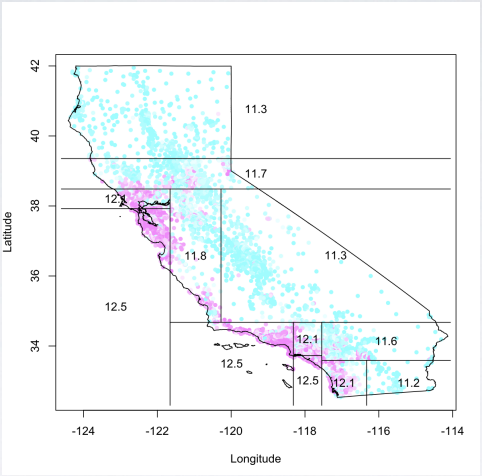


FIGURE: Map of actual median house prices (color-coded by decile, darker being more expensive), and the partition of the `treefit` tree (12 leaves)

QUESTIONS?

How to grow a tree

- The algorithm needs to automatically decide on the **splitting variables** and **split points**, and also what topology (shape) the tree should have.
- To judge whether a given tree is good or bad we need a **criterion function**.
- Suppose we have a partition with M **regions**: $R_1, \ldots, R_M$
- Consider the estimator we discussed before

$$\hat{g}(x) = \sum_{m=1}^{M} \hat{c}_m I\{x \in R_m\} .$$

- We may choose $\hat{c}_m$ in order to minimize some criteria: e.g., sum of squares,

$$\sum_{i=1}^{n} (Y_i - \hat{g}(x))^2 .$$

How to grow a tree

- **Criterion**: minimize the sum of squares. Easy to see that $\hat{c}_m$ is just the average of Y_i in region R_m,

$$\hat{c}_m = \frac{\sum_{i=1}^{n} Y_i I\{X_i \in R_m\}}{\sum_{i=1}^{n} I\{X_i \in R_m\}} = \frac{1}{N_m} \sum_{X_i \in R_m} Y_i \,,$$

  where $N_m = \sum_{i=1}^{n} I\{X_i \in R_m\}$.

- **Next**: choose the number M and partition $R_1, \ldots, R_M$ that deliver the minimum value of

$$\sum_{i=1}^{n} (Y_i - \hat{g}(x))^2 \,.$$

- **Result**: Best binary partition in terms of minimum sum of squares.

- **Problem**: This is an NP-Hard problem.

- The common get around is to use a **greedy algorithm**

Definition (Greedy Algorithm)

- Consider splitting variable j and split point s. Define

$$R_1(j,s) = \{X | X_j \leqslant s\} \quad \text{and} \quad R_2(j,s) = \{X | X_j > s\}$$

- Seek j and s that solve

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] .$$

- For any choice of j and s, the inner minimization is solved by

$$\hat{c}_a = \frac{\sum_{i=1}^n Y_i I\{X_i \in R_a(j,s)\}}{\sum_{i=1}^n I\{X_i \in R_a(j,s)\}} \text{ for } a \in \{0,1\} .$$

- For each variable j, determination of split point s can be done quickly.
- Scanning all covariates can be done quickly too $\Rightarrow$ best (j,s).
- Given the best split, repeat the splitting process on each of the two regions.
- The process is repeated again on the resulting regions, etc.

How large should we grow a tree?

- Tree size is a tuning parameter governing the model's complexity
- Optimal tree size should be adaptively chosen from the data.
- Naive approach: split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold.
- Short-sighted: a seemingly worthless split might lead to a very good split below it.
- Preferred strategy: grow a large tree T_0, stopping the splitting process only when some minimum node size (say 5) is reached. Then prune this tree using cost-complexity pruning.
- Pruning??? 😱

Definition (Pruning)

To prune a tree T in a (non-terminal) node t means that t becomes a leaf node and all descendants of t are removed.
The resulting tree is called a **subtree**.

Pruning

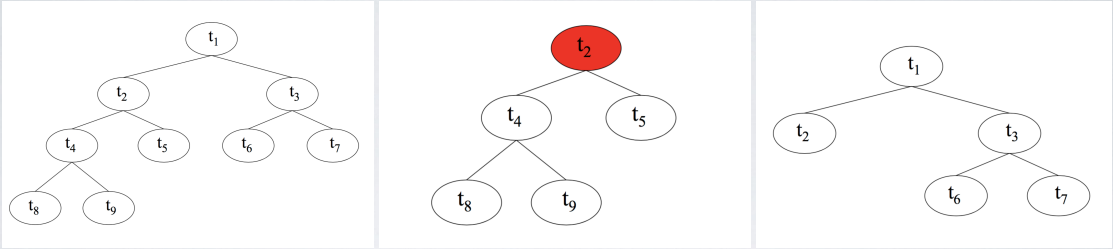


FIGURE: Pruning a Tree - original (left), branch (center), subtree (right)

- Problem: If T is large, there are many subtrees.
- Also, the larger the tree the better the fit (i.e., sum-of-squares)
- Cost-Complexity: penalize the size/complexity of the tree:
  - It avoids getting trees that are unnecessarily large.
  - It reduces the number of subtrees to consider.

Cost-Complexity pruning

- Let $|T|$ denote the **number of terminal nodes** (indexed by m) in T and define

$$N_m = \sum_{i=1}^{n} I\{X_i \in R_m\}$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{X_i \in R_m} Y_i$$

$$Q_m(T) = \frac{1}{N_m} \sum_{X_i \in R_m} (Y_i - \hat{c}_m)^2 .$$

Definition (Cost Complexity Criterion)

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T| . \tag{2}$$

- **Idea**: For given α, find the subtree $T_\alpha \subseteq T$ to minimize $C_\alpha(T)$
- **Note**: $\alpha = 0$ leads to T_0, as expected.

Weakest Link Pruning

- For each α: there is a **unique** smallest subtree T_α that minimizes $C_\alpha(T)$.
- To find T_α we use **weakest link pruning**:
  - Successively collapse the internal node that produces the smallest per-node increase $\sum_{m=1}^{|T|} N_m Q_m(T)$
  - Continue until producing the single-node (root) tree.
  - The approach delivers a sequence of subtrees

$$T_0, T_1, T_2, \ldots, T_p$$

- Gives a (finite) sequence of subtrees that must contain T_α.
- **Importantly!** this holds for every value of α!
- In addition, for $\alpha > \alpha'$ it can be shown that $T_\alpha \subseteq T_{\alpha'}$.
- The last facts deliver an efficient algorithm to find the **smallest** minimizing subtrees at different values of α.
- The parameter α can then be chosen by Cross Validation $\Rightarrow \hat{\alpha}$
- The resulting tree is $T_{\hat{\alpha}}$.

QUESTIONS?

Classification Trees

- Suppose the outcome is a **classification** outcome taking values $1, 2, ..., K$.
- **Tree algorithm**: the criteria for splitting nodes and pruning the tree changes.
- In a node m, representing a region R_m with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{X_i \in R_m} I\{Y_i = k\}$$

  be the **proportion of class** k observations in node m.

- We classify the observations in node m to the **majority class** in node m,

$$k(m) = \arg\max_k \hat{p}_{mk}$$

- **Different measures** $Q_m(T)$: of node impurity include Misclassification Error,

$$Q_m(T) = \frac{1}{N_m} \sum_{X_i \in R_m} I\{Y_i \neq k(m)\} = 1 - \hat{p}_{mk(m)} \ ,$$

  Gini Index, and Cross-Entropy. The last two are differentiable; hence amenable to optimization.

Tree Instability

- **Problem** trees have high variance.
- Often a small change in the data can result in a very different series of splits, making interpretation somewhat precarious.
- The major reason for this instability is the hierarchical nature of the process: the effect of an error in the top split is propagated down to all of the splits below it.
- One can alleviate this to some degree by trying to use a more stable split criterion, but the inherent instability is not removed.
- It is the price to be paid for estimating a simple, tree-based structure from the data.
- **Bagging** averages many trees to reduce this variance.

Bagging (Bootstrap Aggregating)

- Bagging stands for **Bootstrap Aggregating**
- **Idea**: In situations where we have an estimator $\hat{g}(x)$ that has possibly high variance, we could reduce the variability by averaging the same estimator over bootstrap samples.
- Let $\{(Y_1^{*,b}, X_1^{*,b}), \ldots, (Y_n^{*,b}, X_n^{*,b})\}$ be a bootstrap sample from $\hat{P}_n$, the empirical distribution of the original sample $\{(Y_1, X_1), \ldots, (Y_n, X_n)\}$
- Index the bootstrap samples by $b = 1, \ldots, B$ and let $\hat{g}^{*,b}(x)$ denote the estimate of $g(x)$ using the bth bootstrap sample.
- The **bagging estimate** is defined by

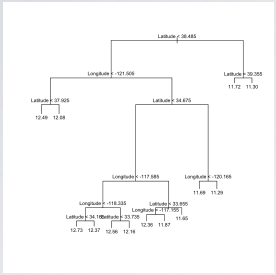
$$\hat{g}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{g}^{*,b}(x) \ . \tag{3}$$

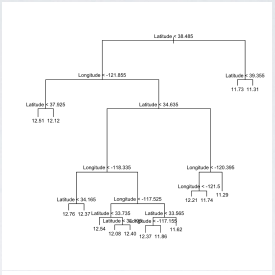
- Note: the idea of bagging can be applied to any estimator (not necessarily tress)

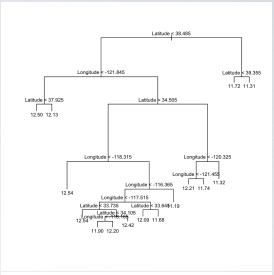
Bagging with Trees

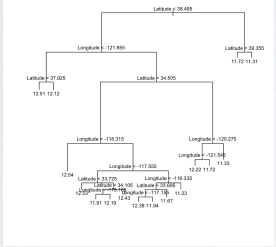
- Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes.
- The bagged estimate is the average prediction at x from these B trees.
- For classification problems, the bagged classifier selects the class with the most "votes" among the B trees.
- Bagging can dramatically reduce the variance of unstable procedures like trees, leading to improved prediction.
- Under square-loss: averaging reduces variance and leaves bias unchanged.
- Several packages in `R` for bagging **CART** (classification and regression trees).

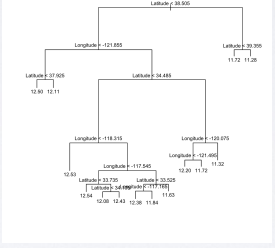
Bagging: Housing data











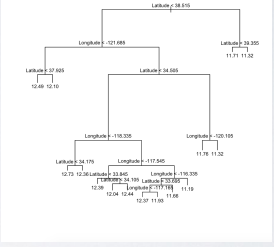


FIGURE: Trees for 6 bootstrap samples: CA housing data

Bagging: Housing data

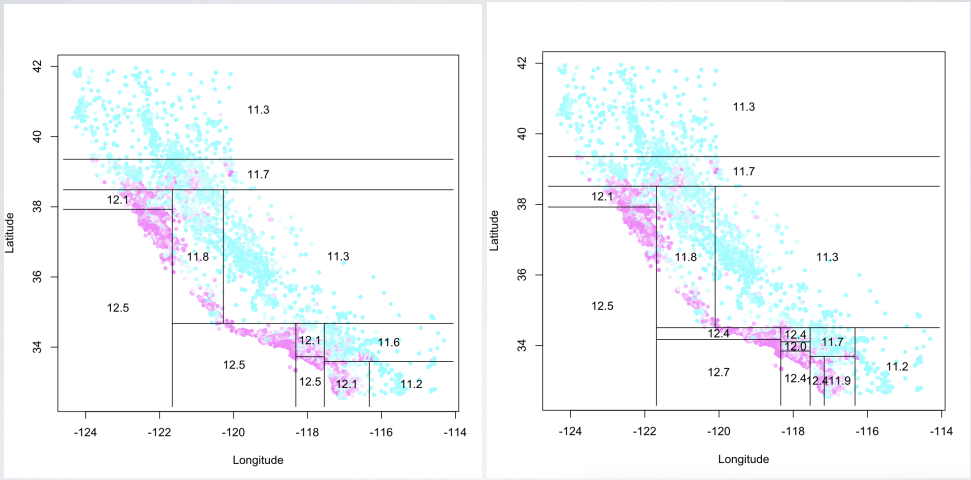


FIGURE: Map of actual median house prices (color-coded by decile, darker being more expensive). Left (original). Right (6th bootstrap sample in previous slide)

QUESTIONS?

Random Forests

- **Random forests** (Breiman, 2001) is a substantial modification of bagging that builds a large collection of **de-correlated trees**, and then averages them.
- On many problems the performance of random forests is very similar to **boosting**, which in turn is an improved version of bagging (not covered here)
- Random forest are simpler to train and tune and, as a consequence, are very popular.
- They can be implemented in a variety of packages, including `randomForests` in `R`.

Correlated vs Uncorrelated Trees

- The essential idea in **bagging** is to average many noisy but approximately unbiased models to reduce the variance.
- Since trees are notoriously noisy, they benefit greatly from the averaging.
- An average of B i.i.d. random variables, each with variance σ^2, has variance
$$\frac{1}{B}\sigma^2 .$$
- With n.i.i.d. (identically distributed, but not independent) with positive pairwise correlation ρ, the variance of the average is
$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 .$$
- As B increases, the second term disappears, but the first remains, and hence the size of the **correlation** of pairs of bagged trees **limits the benefits** of averaging.
- **Random forests**: improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much.

Random Forests: Algorithm

Random Forests algorithm

1. For $b = 1, \ldots, B$
   - 1.1 Draw a bootstrap sample of size n from the training data.
   - 1.2 Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{\min}$ is reached.
     - 1.2.1 Select s variables at random from the k variables.
     - 1.2.2 Pick the best variable/split-point among the s.
     - 1.2.3 Split the node into two daughter nodes
   - 1.3 Output the ensemble of trees $\{T_b : 1 \leqslant b \leqslant B\}$.

To make a prediction at a new point x:

$$\text{Regression} \quad \hat{g}_{\text{rf}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{g}_b(x)$$

$$\text{Classification} \quad \hat{C}_{\text{rf}}(x) = \text{majority vote}\{C_b(x) : 1 \leqslant b \leqslant B\}\,.$$

Comments

- The main feature of a random forest is that, when growing a tree on a bootstrapped dataset:

  Before each split, select $s < k$ of the input variables at random as candidates for splitting.

- Intuitively, reducing s will reduce the correlation between any pair of trees in the ensemble, and hence reduce the variance of the average.

- "Suggested" values of s are $\sqrt{k}$ for classification and $k/3$ for regression.

- Not all estimators can be improved by shaking up the data like this.

- Highly nonlinear estimators, such as trees, benefit the most.

QUESTIONS?

Random Forests vs Boosting: Housing data

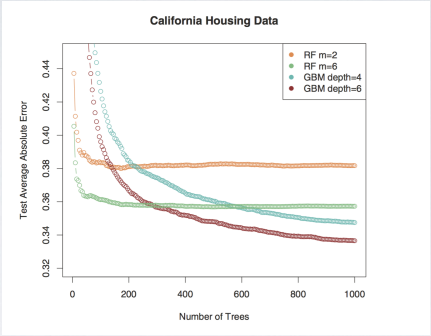


FIGURE: Random forests compared to gradient boosting on the California housing data. The curves represent mean absolute error on the test data as a function of the number of trees in the models.

- Random forests stabilize at about 200 trees, while at 1000 trees boosting continues to improve. Boosting has several tuning parameters.
- Boosting outperforms random forests here. More similar in other cases.
- Note: $s = 6$ performs better than the default value $\lfloor s/3 \rfloor = 2$.
- Unlike Boosting (which is sequential), RF grows trees in **parallel**.

Application: Microsoft Kinect

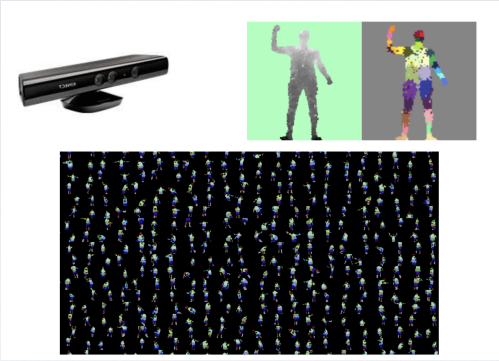


FIGURE: Body part classification in the kinect. 1 million test images

- Algorithm trained on synthetic data (computer graphics)
- Depth image capture: for each pixel computes relative depth (relative to two random directions)
- Goal: quickly classify joints (head, elbows, etc)
- Relatively easy to update algorithm with new images
- Note: no training in real time (training takes a long time) - only classification
- 1 million images: 1 day in 1,000 core cluster.

Application: segmentation of brain tumors

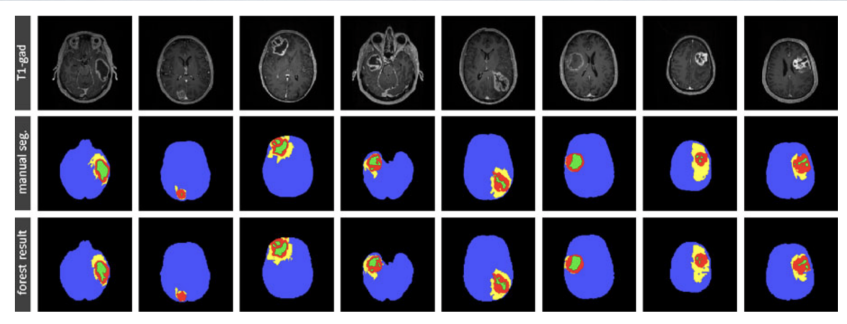


FIGURE: Examples of results on eight (previously unseen) test patients. Results are obtained by a forest trained on 30 patients. The automatic segmentation results (bottom row) look very similar to the manual, ground truth segmentations (middle row)

- Delineation of tissue components is crucial for radiotherapy and surgery planning and is currently performed manually in a labor intensive fashion.
- Here we compare MRI segmentation done by a radiotherapist with that of a random forests algorithm (with some modifications not discussed here).

Conclusions

- The methods we discussed are a type of **non-parametric regression**
- They are about predicting Y given X and involve tuning parameters.
- Most of the **"buzz"** about these machine learning techniques come from accuracy in practical prediction problems (Kinect, Netflix contests, online searches, image recognition, etc)
- The theoretical properties of $\hat{g}_{\text{rf}}(x)$ still being **developed**.
  1. Recent papers by Athey, Tibshirani, Wager, and others are making progress.
  2. Methods suffer from the **curse of dimensionally**, but may work better in some families of DGPS.

$$g(x_1, x_2, x_3, x_4) = h_4(h_1(x_2, x_4), h_2(x_1), h_3(x_3)) = h_4 \circ h_3 \circ h_2 \circ h_4 \Rightarrow d^* = 2 < d = 4$$

Schmidt-Hieber. "Nonparametric regression using deep neural networks with ReLU activation function." Ann. Statist. 2020.

- Economics is less about prediction than other sciences $\Rightarrow$ but causal parameters usually involve conditional expectations so...

THE END

Midterm Exam

Min	24
25th Q	52
Median	61
Mean	64
75th Q	75
Max	100

Table: Distribution of grades
Midterm Exam 2021