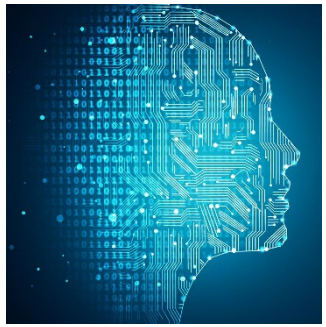
Machine Learning

Decision Trees

Kevin Moon (kevin.moon@usu.edu)

STAT/CS 5810/6655



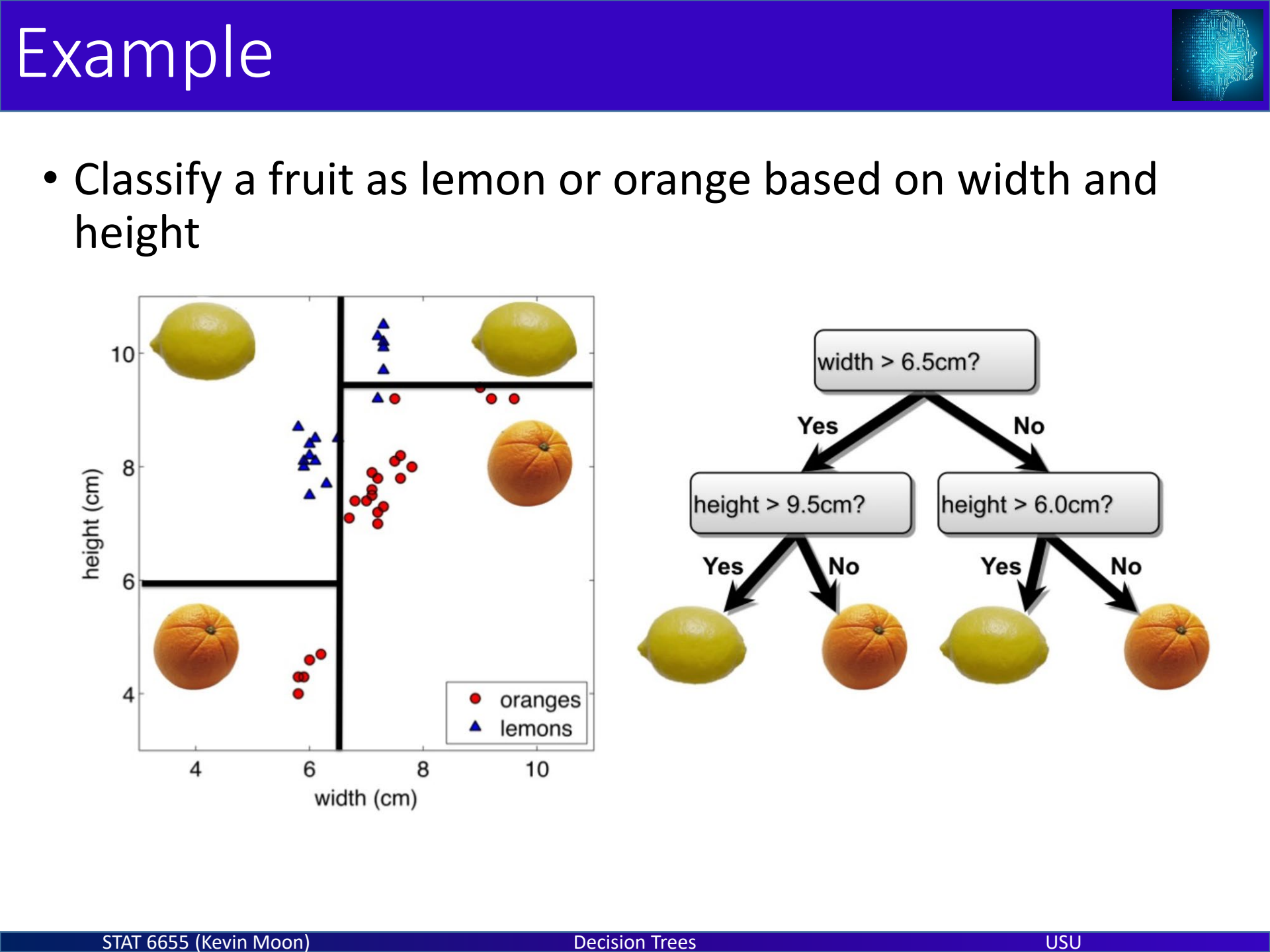
Supervised Learning

- Let's come back to supervised learning
- Remember that we have labeled data

$$(\boldsymbol{x}_i, y_i)_{i=1}^n$$

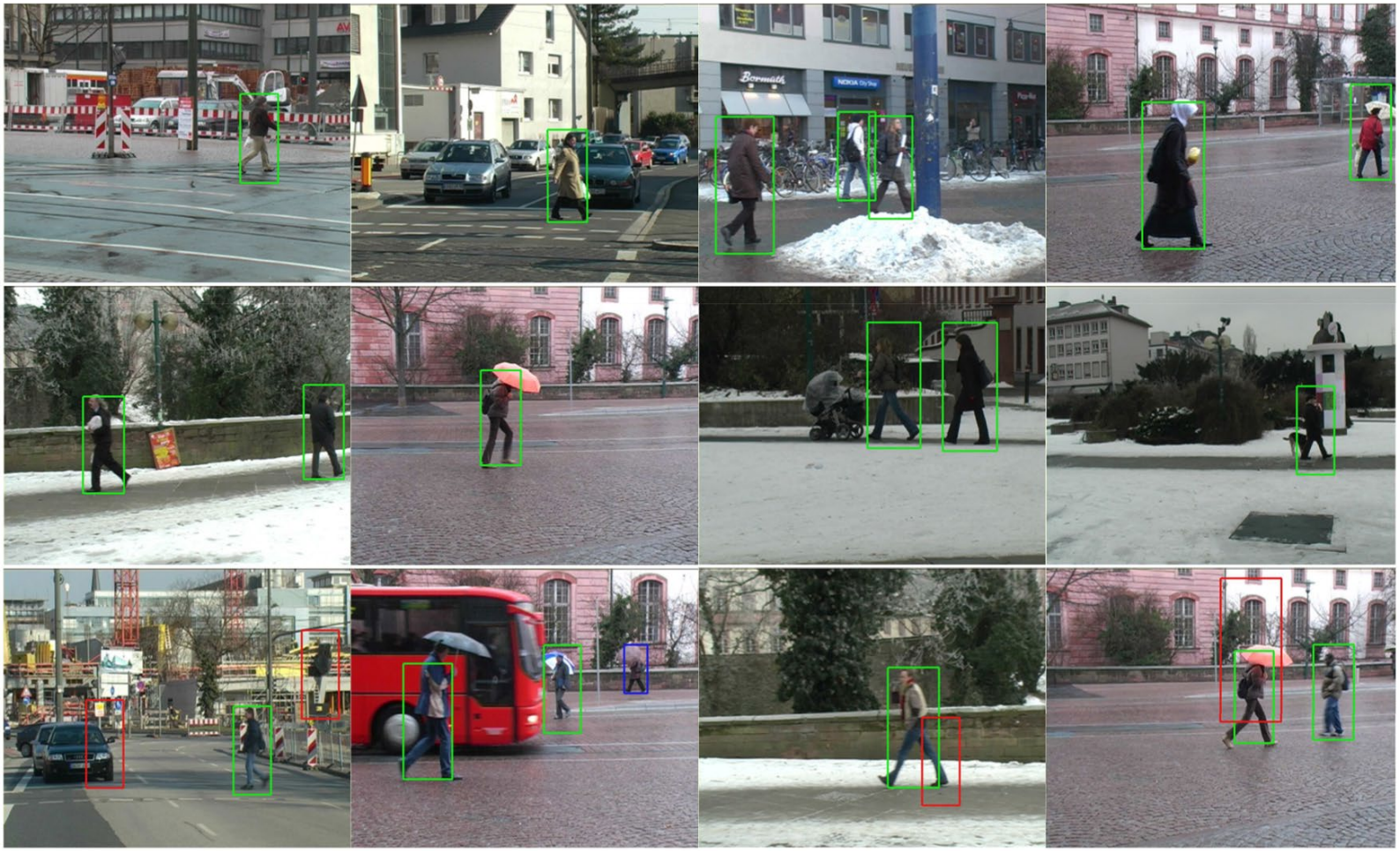
- y_i finite $\Rightarrow$ classification
- $y_i \in \mathbb{R} \Rightarrow$ regression
- We'll consider classification and regression trees (CART)

Example



- Classify a fruit as lemon or orange based on width and height

Example: Object Detecion

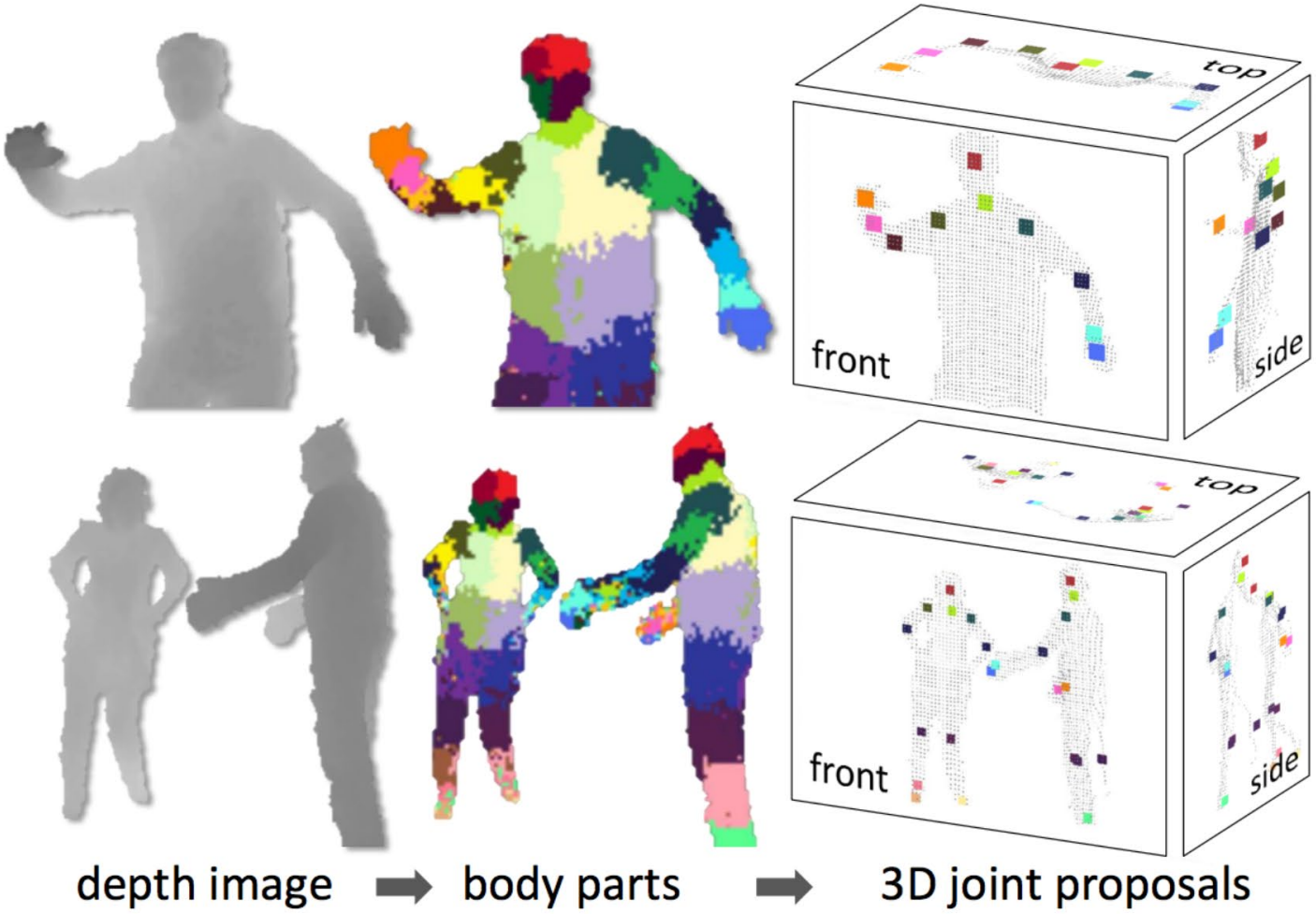


Example: Xbox Kinect

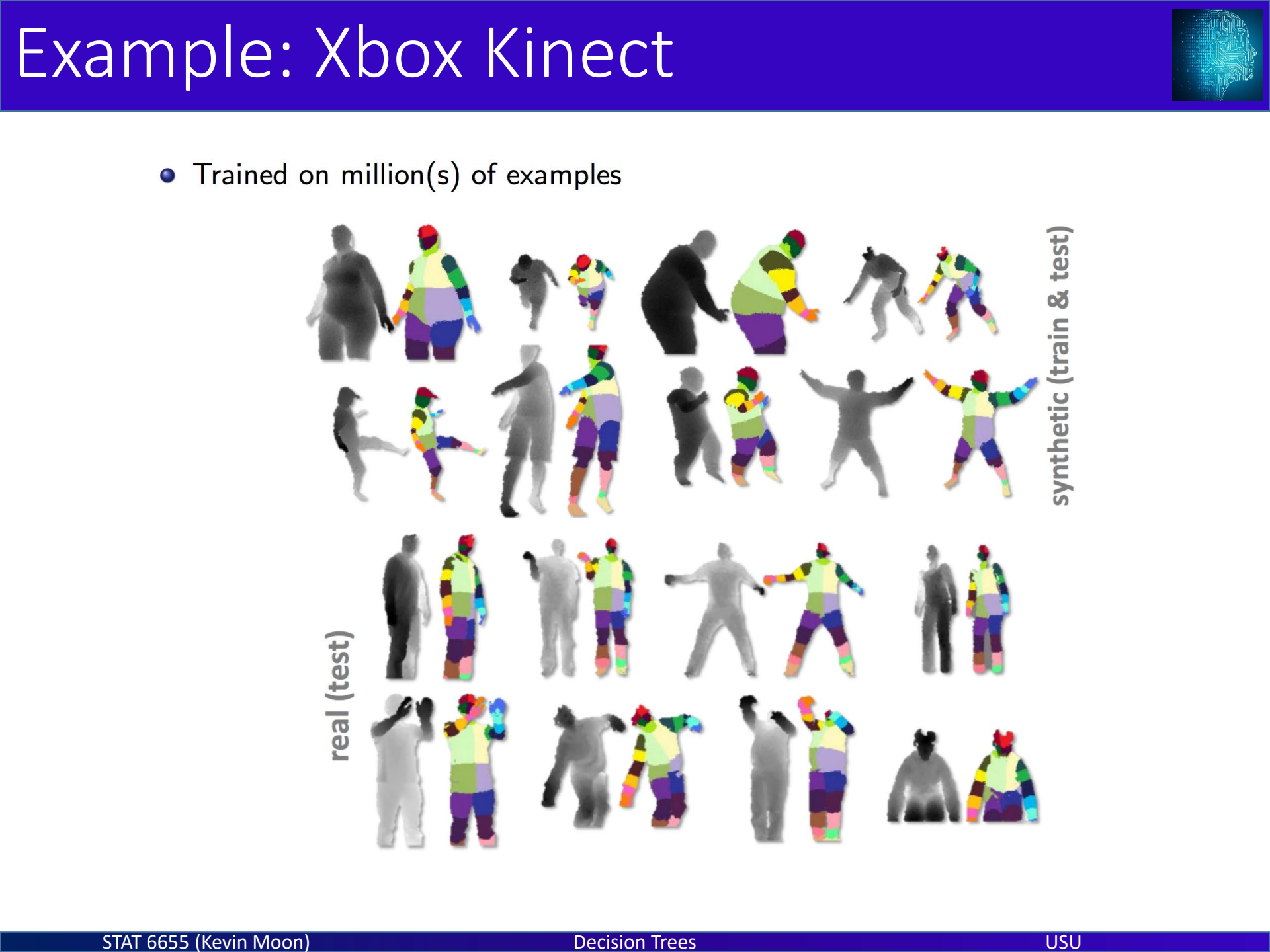
[J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake. Real-Time Human Pose

Example: Xbox Kinect

- Classifying body parts



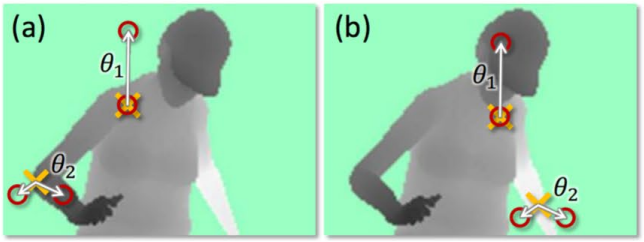
Example: Xbox Kinect

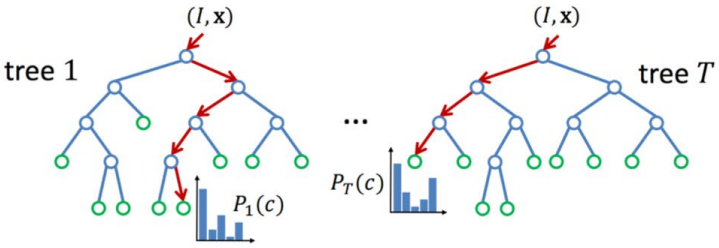


- Trained on million(s) of examples

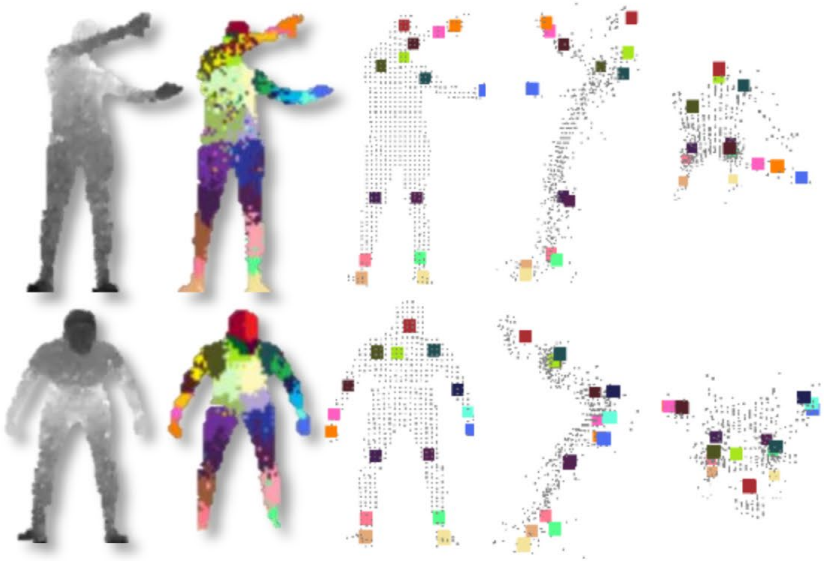
Example: Xbox Kinect

- Trained on million(s) of examples





- Results:

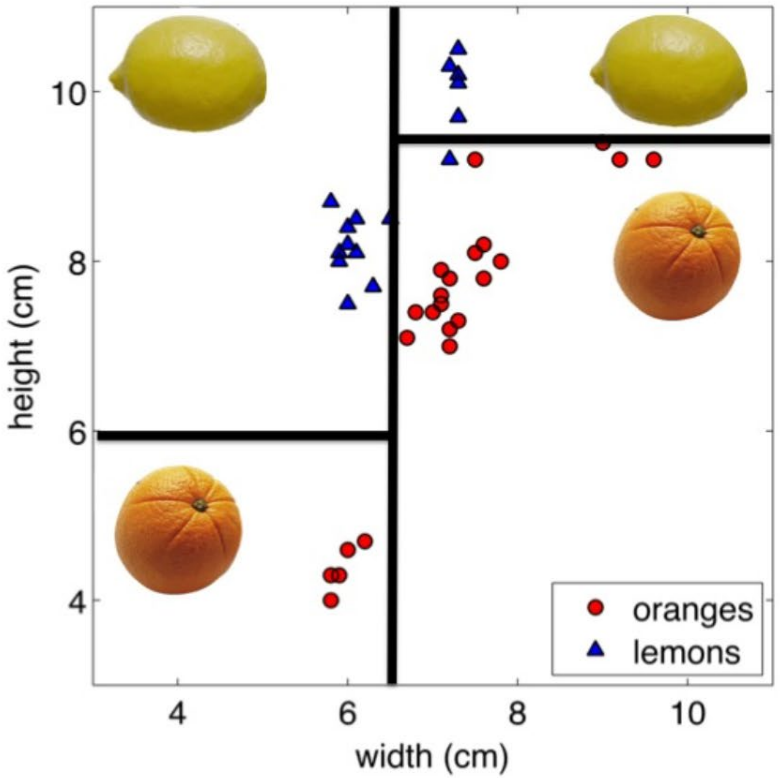


Decision Trees

- In short, a decision tree asks a series of simple questions about $\boldsymbol{x}$ and then predicts a label for $\boldsymbol{x}$
- Decision trees can also be applied to regression
  - Labels at the "leaf nodes" are replaced with real numbers
- Other generalizations
  - Splits that involve more than one feature
  - Splits with more than two outcomes
  - The above splits can more easily lead to overfitting
- For simplicity, we'll assume only binary splits on a single feature

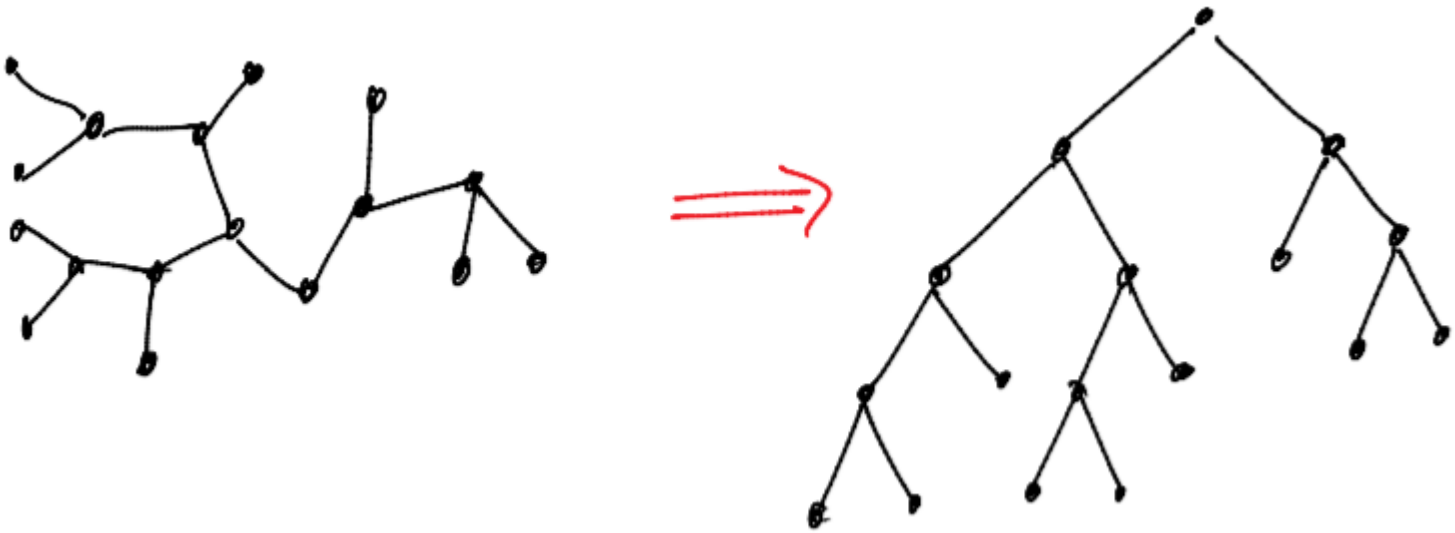
Some terminology

- Each decision tree is associated with a partition of the feature space
  - The elements of this partition are called cells
- Recall that a graph is a collection of nodes, some of which are connected by edges
  - The degree of a node is the number of edges incident on that node



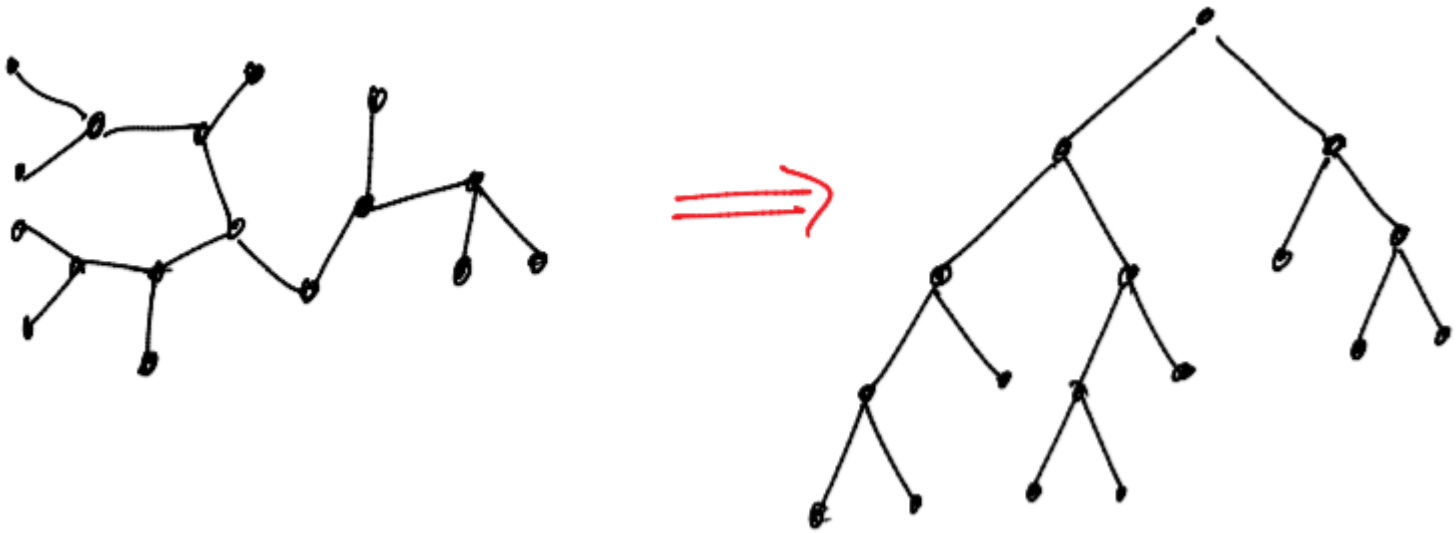
Some terminology

- A tree is a connected graph with no cycles
- A rooted binary tree is a tree where one node, called the root, has degree 2 and all other nodes have either degree 1 or degree 3
  - Degree 1 nodes are called leaf or terminal nodes
  - All other nodes are called internal nodes



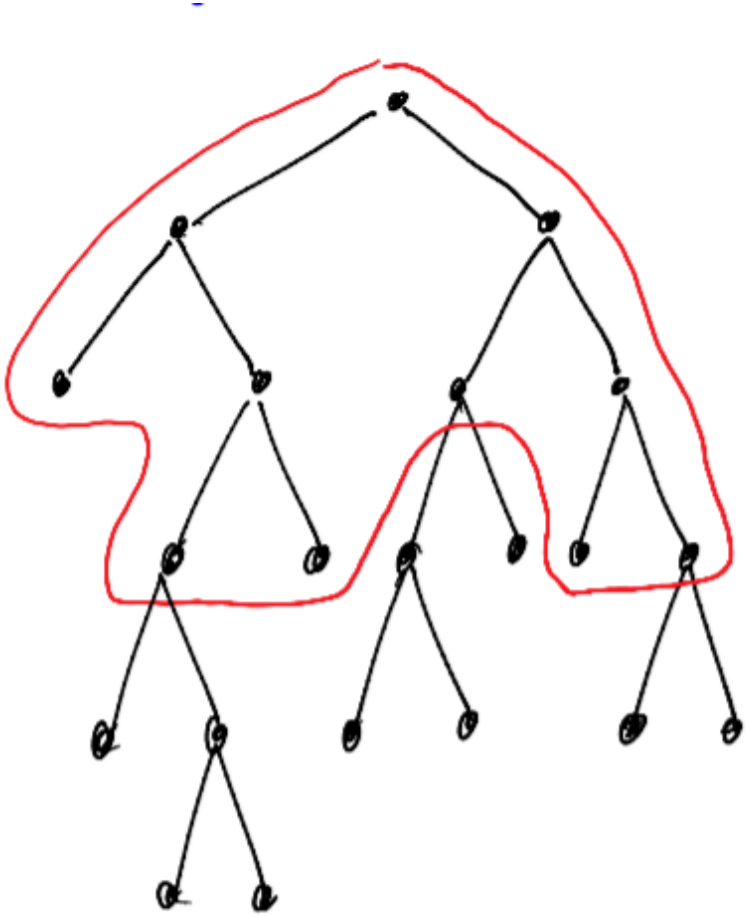
Some terminology

- The depth of a node is the number of edges between that node and the root
- The parent of a node is the neighbor of the node whose depth is one less
- Two nodes are siblings if they have the same parent



Some terminology

- A subtree is a subgraph that is also a tree
- A rooted binary subtree is a subtree that contains the root and is such that if any non-root node is included, so is its sibling
- A binary decision tree is a rooted binary tree where each internal node is associated with a binary classifier and each leaf node with a label



Learning Decision Trees

- Let $\mathcal{T} =$ set of all binary decision trees
- Basic strategy: regularized ERM

$$\min_{T \in \mathcal{T}} \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, T(\boldsymbol{x}_i)\right) + \lambda |T|$$

- L is an appropriate loss, e.g. 0-1 loss for classification or squared error loss for regression
- $|T| =$ number of leaf nodes
- Unfortunately, this is intractable and so a different procedure is typically employed (next slide)

Learning Procedure

1. Grow a very large tree T_0 using a greedy approach
2. Prune T_0 by solving

$$\min_{T \in \mathcal{T}_0} \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, T(\boldsymbol{x}_i)\right) + \lambda |T|$$

where $\mathcal{T}_0$ is the set of all binary decision trees based on rooted binary subtrees of T_0

- Let's talk about each of these steps in detail

Growing a Decision Tree

- A greedy algorithm is an algorithm that attempts to solve a global optimization problem by taking the locally optimal choice at each stage
- T_0 is constructed using a greedy approach

1. Start at root node (entire feature space)
2. Decide whether to stop growing tree. If yes, assign a label. Stop
3. If no, consider all possible ways to split the data reaching the current node and select the best one
4. For each branch of the split, create a new node and go to Step 2

Growing a Decision Tree

1. Start at root node (entire feature space)
2. Decide whether to stop growing tree. If yes, assign a label. Stop
3. If no, consider all possible ways to split the data reaching the current node and select the best one
4. For each branch of the split, create a new node and go to Step 2

- To implement this strategy, we need:

a) List of possible splits

b) Labelling rule

c) Rule for stopping splitting

d) Rule for selecting the best split

Growing a Decision Tree

1. Start at root node (entire feature space)
2. Decide whether to stop growing tree. If yes, assign a label. Stop
3. If no, consider all possible ways to split the data reaching the current node and select the best one
4. For each branch of the split, create a new node and go to Step 2

- To implement this strategy, we need:

a) List of possible splits

- For continuous features, splits have the form $X^{(j)} \leq t?, j = 1, \ldots, d$
- n data points $\Rightarrow$ only $n - 1$ values need to be considered for each j
- For categorical or discrete features, other simple splits can be used

$$X^{(j)} = \text{blue}?$$

Growing a Decision Tree

1. Start at root node (entire feature space)
2. Decide whether to stop growing tree. If yes, assign a label. Stop
3. If no, consider all possible ways to split the data reaching the current node and select the best one
4. For each branch of the split, create a new node and go to Step 2

- To implement this strategy, we need:

b) Labelling rule

- Majority vote over the data reaching the given node for classification
- For regression, take the average y_i over the node

Growing a Decision Tree

1. Start at root node (entire feature space)
2. Decide whether to stop growing tree. If yes, assign a label. Stop
3. If no, consider all possible ways to split the data reaching the current node and select the best one
4. For each branch of the split, create a new node and go to Step 2

- To implement this strategy, we need:

c) Rule for stopping splitting

- Common strategy: split until each leaf node contains a single data point

Growing a Decision Tree

1. Start at root node (entire feature space)
2. Decide whether to stop growing tree. If yes, assign a label. Stop
3. If no, consider all possible ways to split the data reaching the current node and select the best one
4. For each branch of the split, create a new node and go to Step 2

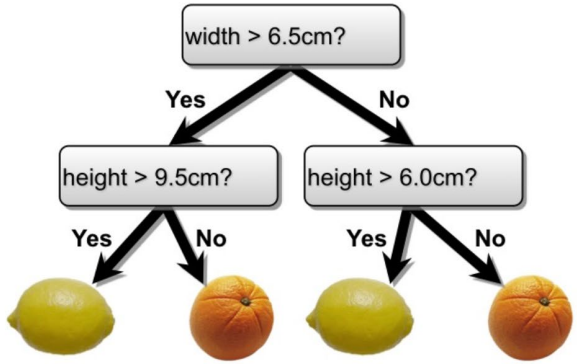
- To implement this strategy, we need:

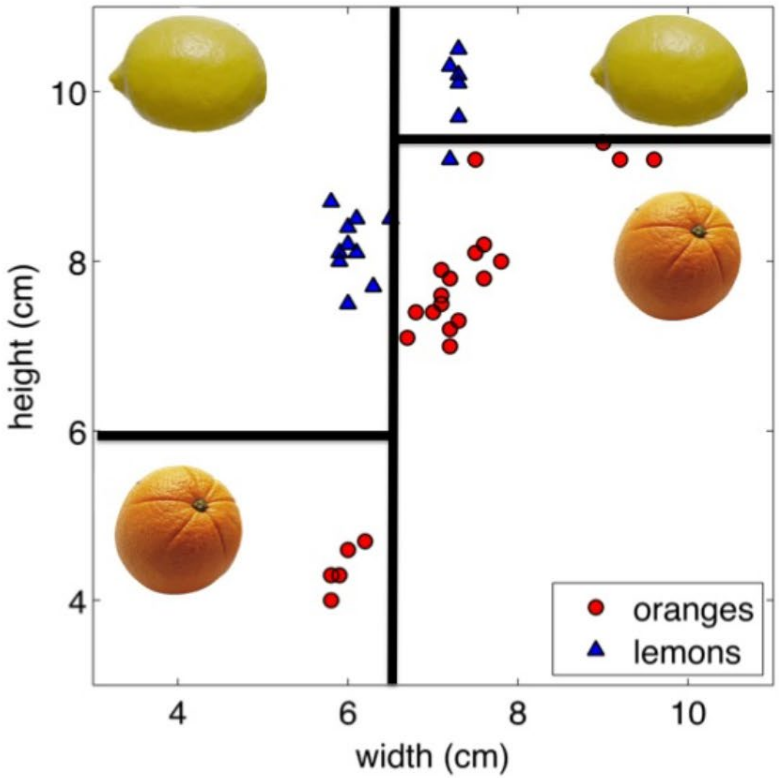
d) Rule for selecting the best split

- Let's go over this in detail

Split Selection with Impurity Measures

- Let's focus on binary classification
- Suppose N is a leaf node at some stage in the growing process
  - Think of N as a cell in a partition of the feature space or the training points in that cell
- Intuitively, a good split leads to children that are more homogeneous or pure than their parent





Impurity Measure

- Assume class labels are $\{0,1\}$
- Denote

$$q := \frac{|\{i: \boldsymbol{x}_i \in N, y_i = 0\}|}{|\{i: \boldsymbol{x}_i \in N\}|}$$

- This ratio defines a probability distribution of the labels
- An impurity measure is a function $i(N)$ s.t.
  - $i(N) \geq 0$ with equality iff N consists of a single class
  - A larger value of $i(N)$ indicates that the distribution defined by q is closer to the uniform distribution

Impurity Measure Examples

1. Entropy

$$i(N) = -[q \log q + (1-q) \log(1-q)]$$

2. Gini index (typically the default)

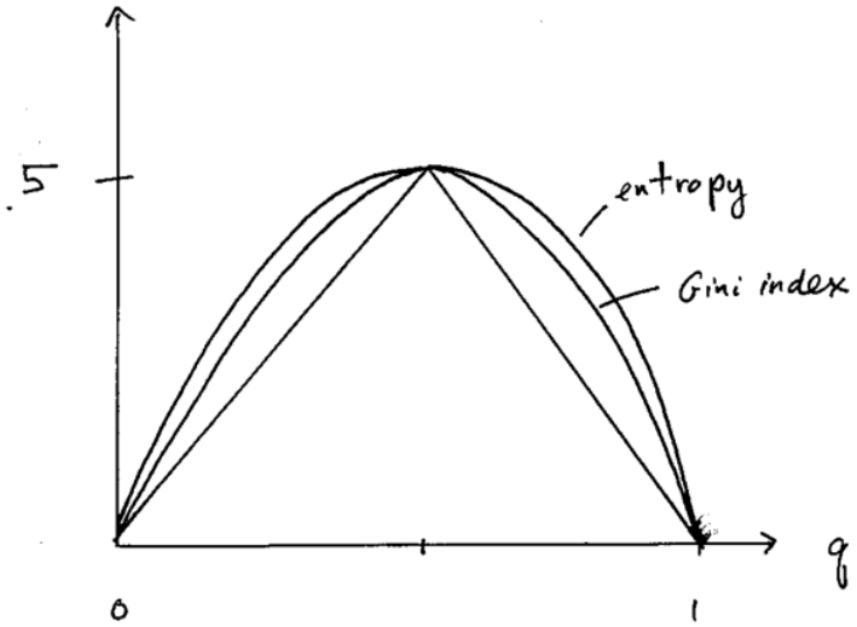
$$i(N) = 2q(1-q)$$

3. Misclassification rate of majority vote classifier

$$i(N) = \min(q, 1-q)$$

Impurity Measure Examples

1. Entropy
   $i(N)$
   $= -[q \log q$
   $+ (1-q) \log(1-q)]$
2. Gini index
   $i(N) = 2q(1-q)$
3. Misclassification rate of majority vote classifier
   $i(N) = \min(q, 1-q)$



Selecting the best split

- Let's maximize the decrease in impurity to choose the best split
- Let N_1 and N_2 be two children of N
- Define $p(N_1) = |N_1|/|N|$ and $p(N_2) = |N_2|/|N|$
  - $|N|$ = # of training points in the cell corresponding to N
- The decrease in expected impurity:

$$i(N) - [p(N_1)i(N_1) + p(N_2)i(N_2)]$$

- When i is entropy, this is called the information gain

Concave impurity measures

Proposition: If i is concave as a function of q, then $i(N) - [p(N_1)i(N_1) + p(N_2)i(N_2)] \geq 0 \; \forall N_1, N_2$. If i is strictly concave and $N_1 \neq \phi \neq N_2$ (ϕ is the empty set) then equality holds iff $q = q_1 = q_2$ where q_i is the proportion of training points of class 0 in N_i.

- This result explains why strictly concave impurity measures are preferred
- The result generalizes easily to multiclass classification

Proof

Let $i(N) = \psi(q)$ where ψ is concave. Then

$$i(N) = \psi(q) = \psi(p(N_1)q_1 + p(N_2)q_2)$$

$$\geq p(N_1)\psi(q_1) + p(N_2)\psi(q_2)$$

$$= p(N_1)i(N_1) + p(N_2)i(N_2).$$

The inequality comes from Jensen's inequality applied to ψ which holds with equality iff the q_i terms are equal and $p(N_i) > 0$ for $i = 1,2$ (this follows from the fact that the N_i are nonempty). Therefore, $q = q_1 = q_2$.

Pruning

- To solve

$$\min_{T \in \mathcal{T}_0} J(T) \coloneqq \frac{1}{n} \sum_{i=1}^{n} L\big(y_i, T(\boldsymbol{x}_i)\big) + \lambda |T|$$

we rely on the fact that the objective function is additive:

- Let $\pi(T) = \{A_1, A_2\}$ be the partition of the feature space corresponding to the children of the root node. Then

$$J(T) = \sum_{A \in \pi(T)} J(T_A)$$

where T_A is the subtree rooted at A

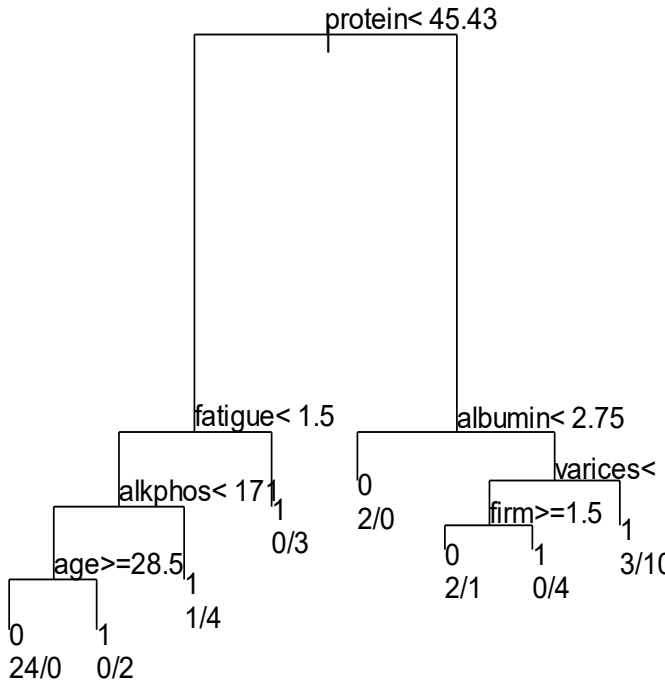
- This suggests a recursive algorithm
- Alternatively, an efficient bottom-up dynamic programming algorithm can be used to solve the pruning problem.

Question

- Why should we grow a tree and then prune it as opposed to growing and just stopping when the decrease in impurity is negligible?
- **Answer**: because of ancillary splits
  - These are splits that have no value by themselves but that enable useful splits later on.

Final Thoughts

- Advantages of decision trees
  - Interpretable
  - Automatic variable selection
  - Can handle nonlinear interactions
  - Rapid evaluation
  - Easily handle categorical data and multiple classes
- Disadvantages of decision trees
  - Greedy growing is suboptimal
  - Unstable: a slight perturbation of training data could drastically change the learned tree
  - Nonsmooth decision boundaries
- The latter two issues can be addressed using ensemble methods (i.e. random forests)



Further Reading

- Ripley, Pattern Recognition and Neural Networks, 1996
- ISL Section 8.1
- ESL Section 9.2