Decision Trees, kNN Classifier

Machine Learning 10-601B

Seyoung Kim

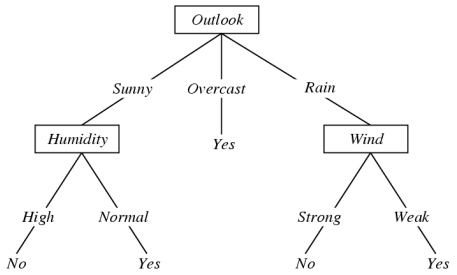
Many of these slides are derived from Tom Mitchell and William Cohen. Thanks!

Beyond linearity

- Decision tree
  - What decision trees are
  - How to learn them

- Nearest neighbor classifier

Decision Tree Learning



Problem Setting:

- Set of possible instances X, Y
  - each instance x in X is a feature vector $x = < x_1, x_2 \ldots x_n >$
  - Y is discrete-valued
- Unknown target function $f : X \rightarrow Y$
- Set of function hypotheses $H = \{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree

Input:

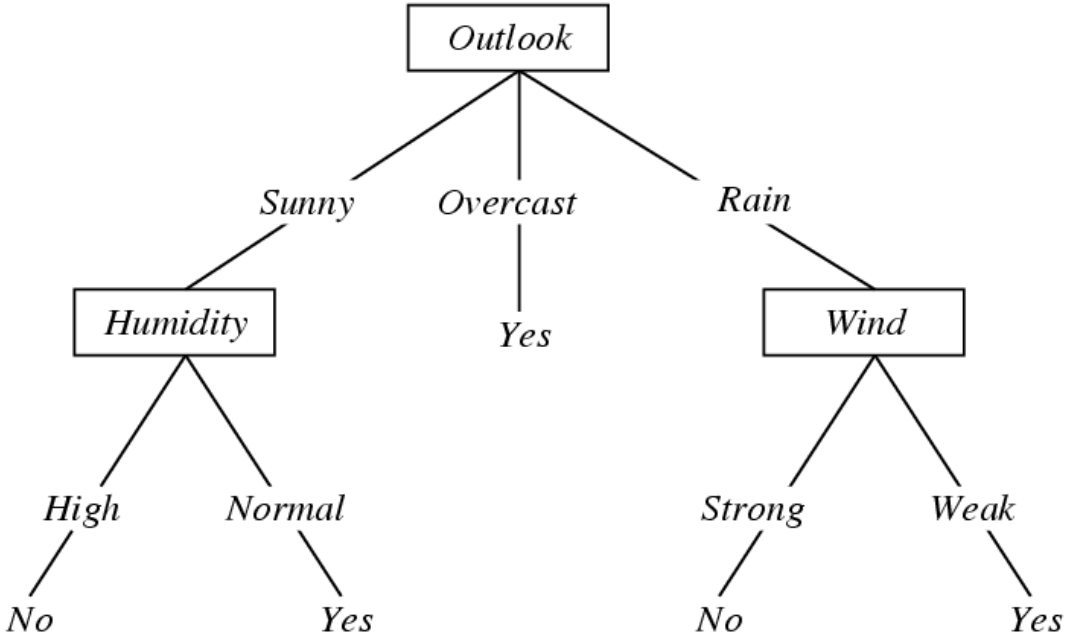
- Training examples $\{<x^{(i)}, y^{(i)}>\}$ of unknown target function f

Output:

- Hypothesis $h \in H$ that best approximates target function f

A Decision tree for

f: <Outlook, Humidity, Wind, Temp> → PlayTennis?



Each internal node: test one discrete-valued attribute X_i

Each branch from a node: selects one value for X_i

Each leaf node: predict Y (or $P(Y|X \in \text{leaf})$)

A Tree to Predict C-Section Risk

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .(
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Decision tree learning

Google

decision tree induction quinlan

Scholar

About 28,100 results (**0.05** sec)

Articles

Case law

My library

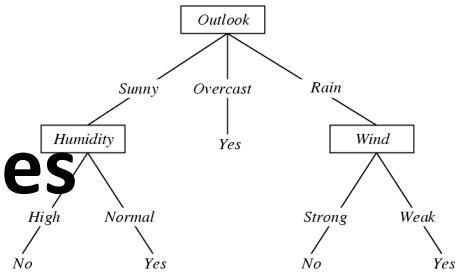
Induction of **decision trees**

JR **Quinlan** - Machine learning, 1986 - Springer

Abstract The technology for building knowledge-based systems by inductive inference from examples has been demonstrated successfully in several practical applications. This paper summarizes an approach to synthesizing **decision trees** that has been used in a variety of **...**

Cited by 16084 Related articles All 74 versions Cite Save More

Motivations for Decision Trees

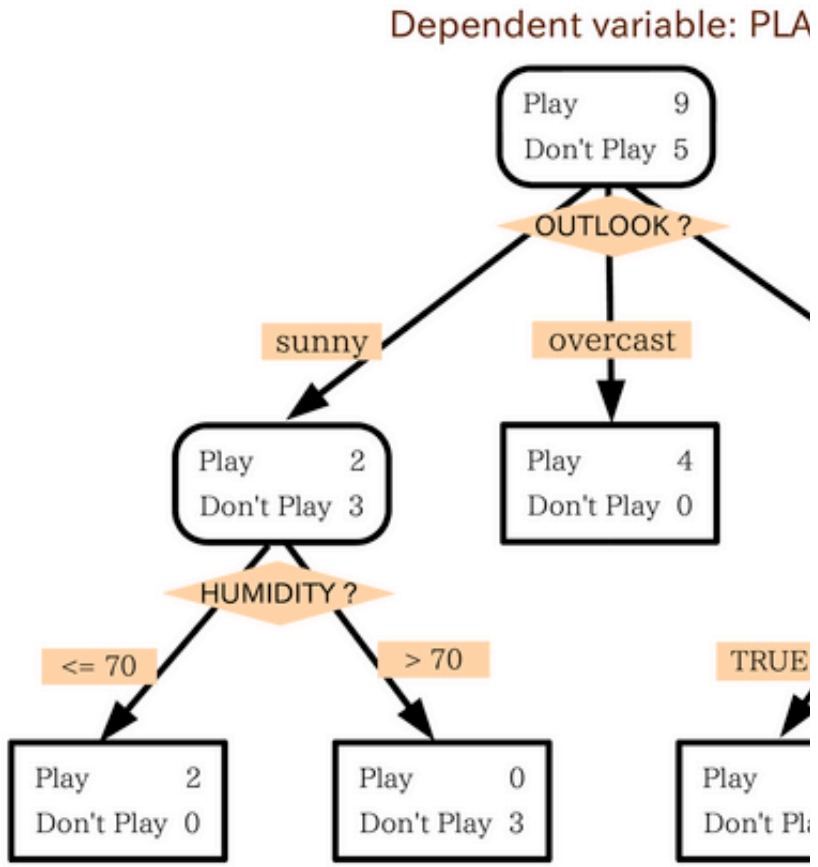


- Often you can find a fairly accurate classifier which is **small and easy to understand**.
  - Sometimes this gives you useful **insight** into a problem
- Sometimes features **interact** in complicated ways
  - Trees can find interactions (e.g., “sunny and humid”) that linear classifiers can’t
- Trees are very **inexpensive at test time**
  - You don’t always **even need to compute all the features** of an example.
  - You can even build classifiers that take this into account….
  - Sometimes that’s important (e.g., “bloodPressure<100” vs “MRIScan=normal” might have different costs to compute).

Decision Tree Learning Algorithm

Decision tree learning algorithms

1. Given dataset D:
   - return *leaf(y)* if all examples are in the same class *y* … or nearly so
   - pick the best split, on the best attribute *a*
     - *a* or *not(a)*
     - $a=c_1$ or $a=c_2$ or …
     - $a<\vartheta$ or $a \geq \vartheta$
     - *a in* $\{c_1,\ldots,c_k\}$ or not
   - split the data into $D_1, D_2, \ldots D_k$ and recursively build trees for each subset
2. "Prune" the tree



Most decision tree learning algorithms

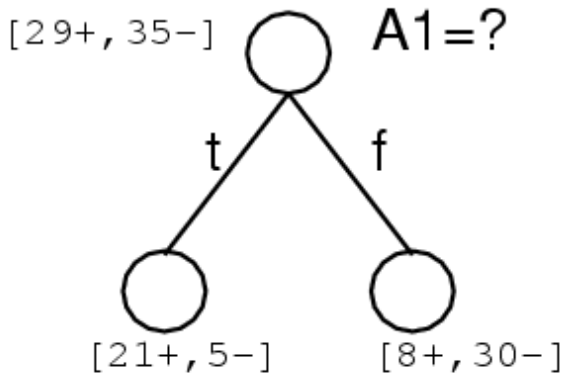
1. Given dataset D:
   - return *leaf(y)* if all examples are in the same class *y* … or nearly so...
   - pick the **best** split, on the best attribute *a*
     - $a=c_1$ or $a=c_2$ or …
     - $a<\vartheta$ or $a \geq \vartheta$
     - *a* or *not(a)*
     - *a in* $\{c_1,\ldots,c_k\}$ or not
   - split the data into D_1 ,D_2 , … D_k and recursively build trees for each subset
2. "Prune" the tree

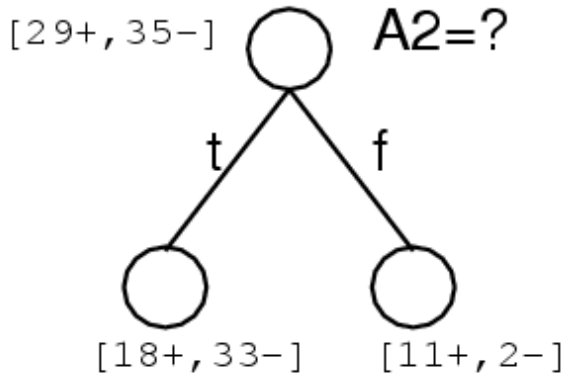
Popular splitting criterion: try to lower *entropy* of the *y* labels on the resulting partition

- i.e., prefer splits that have very skewed distributions of labels

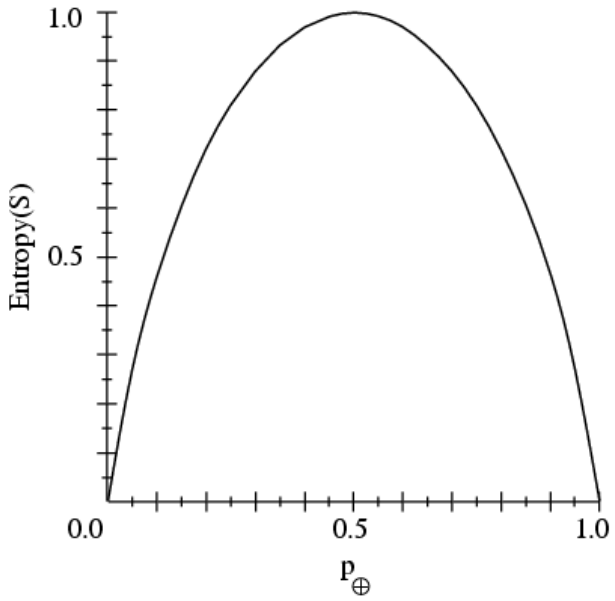
Picking the Best Split

- Which attribute is the best?





Sample Entropy



- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$H(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Entropy

Entropy $H(X)$ of a random variable X

# of possible values for X

$$H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

Why? Information theory:

- Most efficient possible code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random X is:

$$\sum_{i=1}^{n} P(X=i)(-\log_2 P(X=i))$$

Entropy

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)$$

Specific conditional entropy $H(X|Y=v)$ of X given $Y=v$:

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v)\log_2 P(X=i|Y=v)$$

Conditional entropy $H(X|Y)$ of X given Y :

$$H(X|Y) = \sum_{v \in values(Y)} P(Y=v)H(X|Y=v)$$

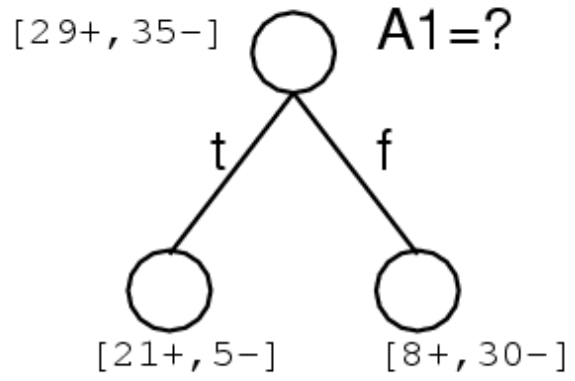
Mutual information (aka Information Gain) of X and Y :

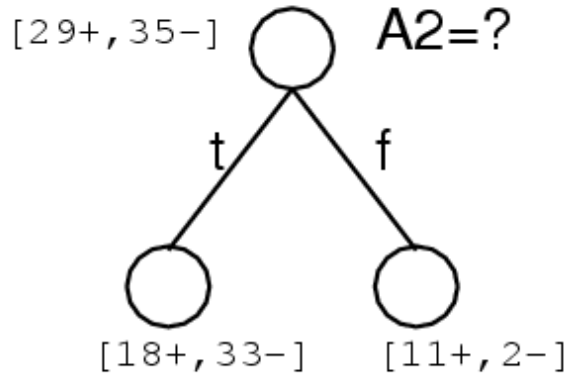
$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Information Gain is the mutual information between input attribute A and target variable Y

Information Gain is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A

$$Gain(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y|A)$$



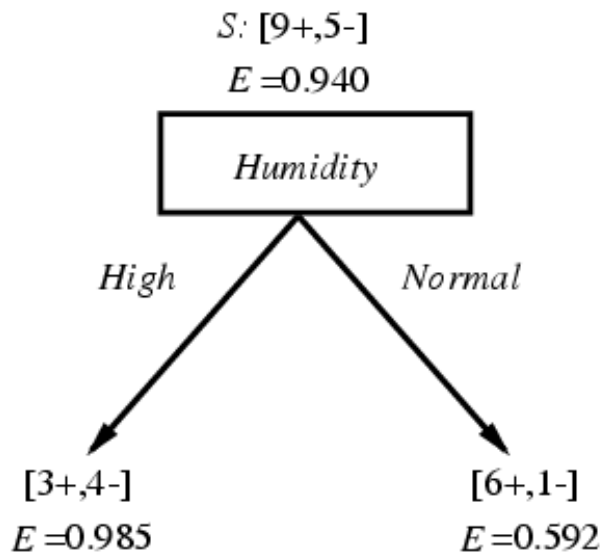


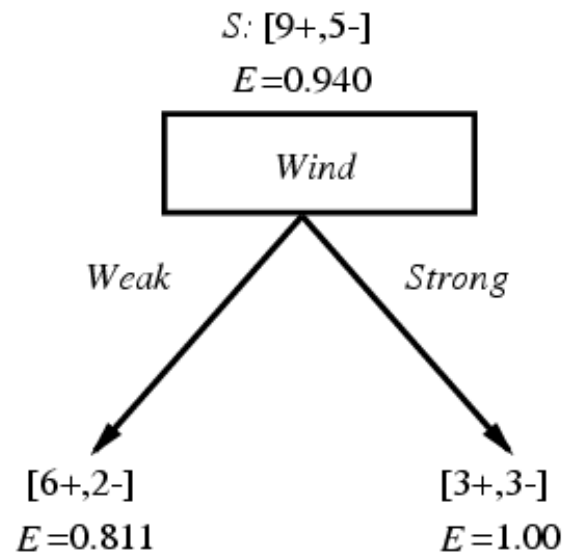
Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Selecting the Next Attribute

Which attribute is the best classifier?



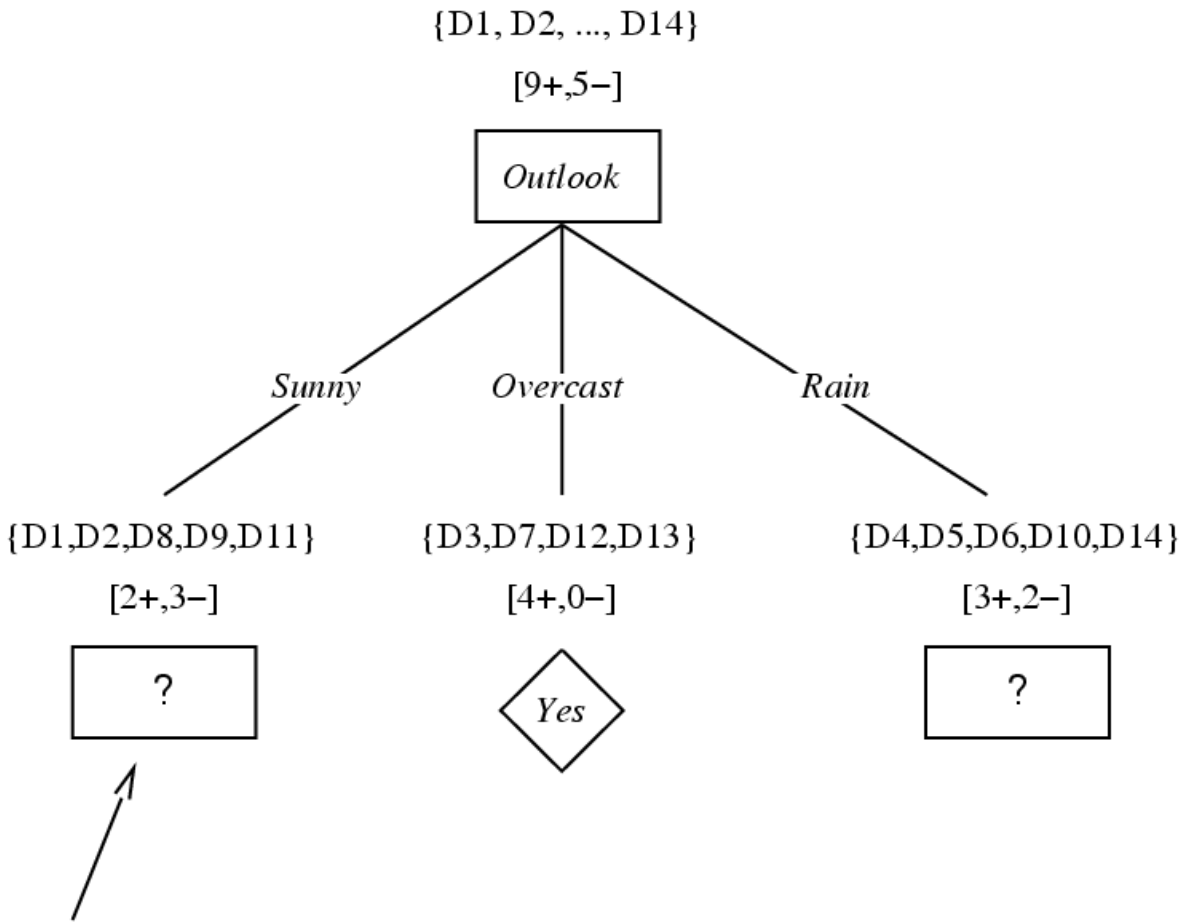


Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

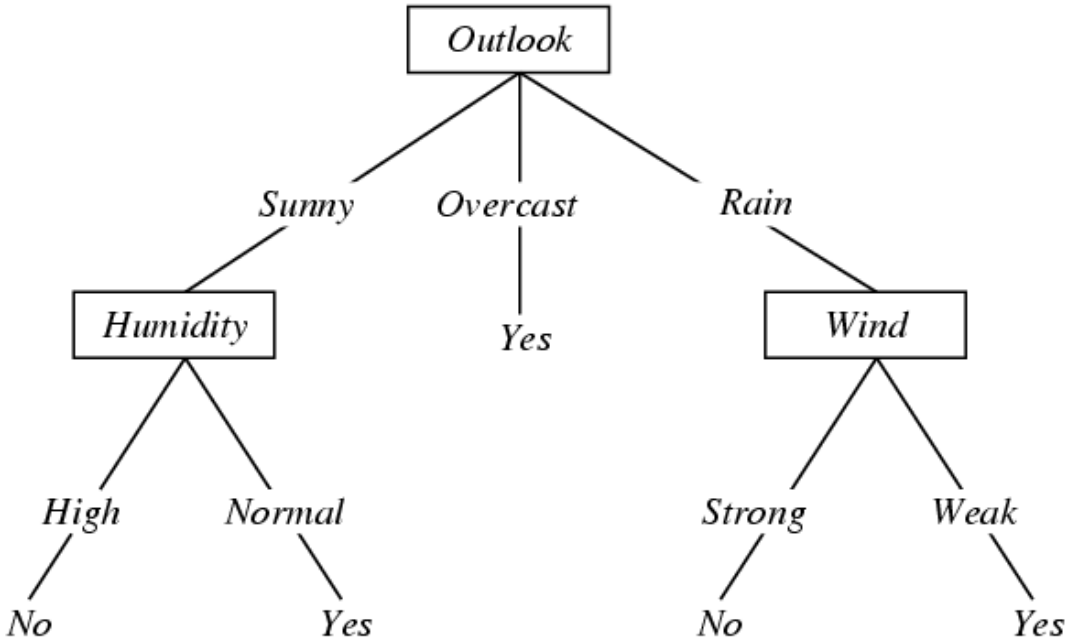
Gain (S_{sunny}, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Overfitting in Decision Trees

Consider adding noisy training example #15:

$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$

What effect on earlier tree?



Overfitting

Consider a hypothesis h and its

- Error rate over training data: $error_{train}(h)$
- True error rate over all data: $error_{true}(h)$

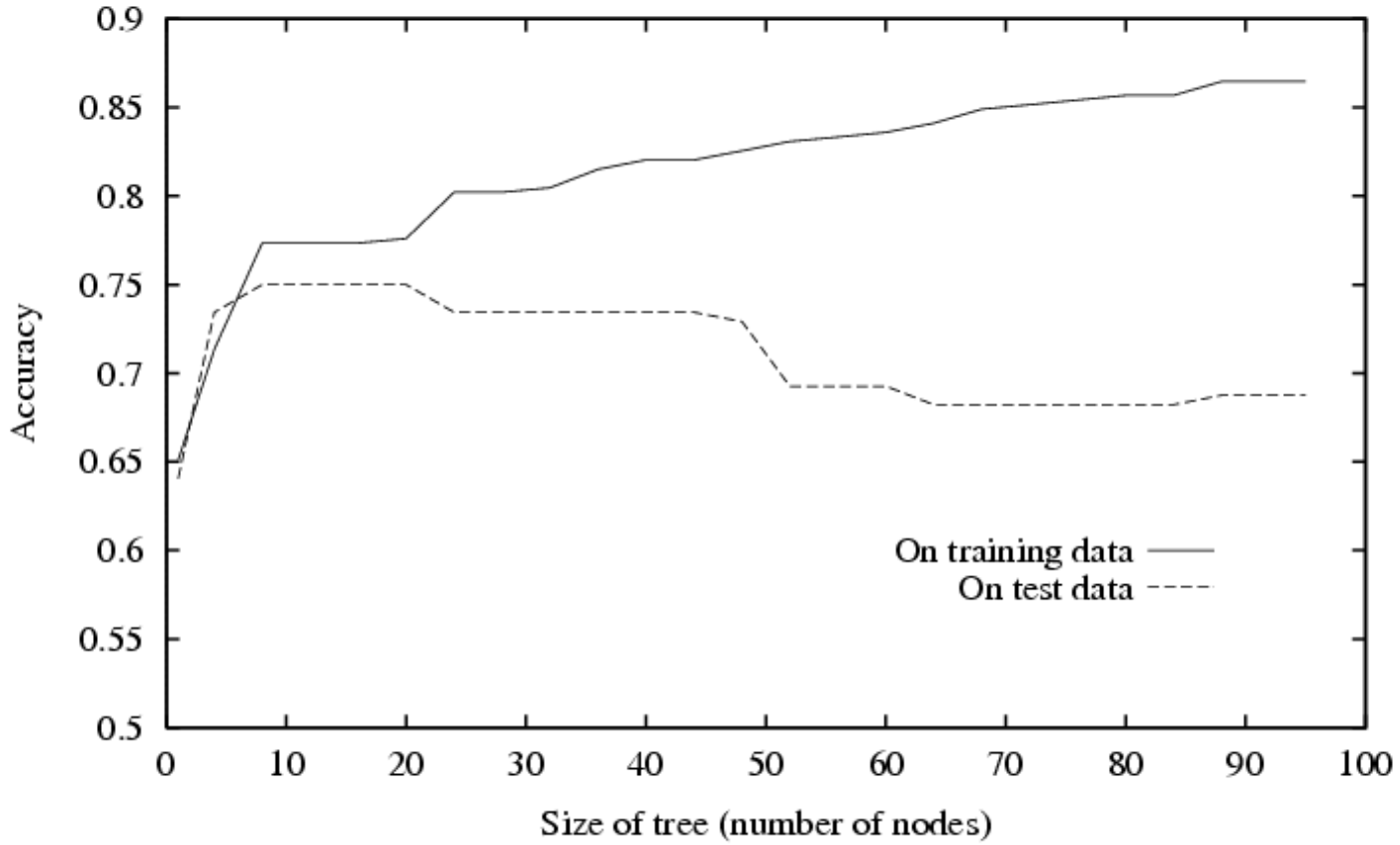
We say h overfits the training data if

$$error_{true}(h) > error_{train}(h)$$

Amount of overfitting =

$$error_{true}(h) - error_{train}(h)$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- MDL: minimize
  $size(tree) + size(misclassifications(tree))$

Reduced-Error Pruning

Split data into *training* and *validation* set

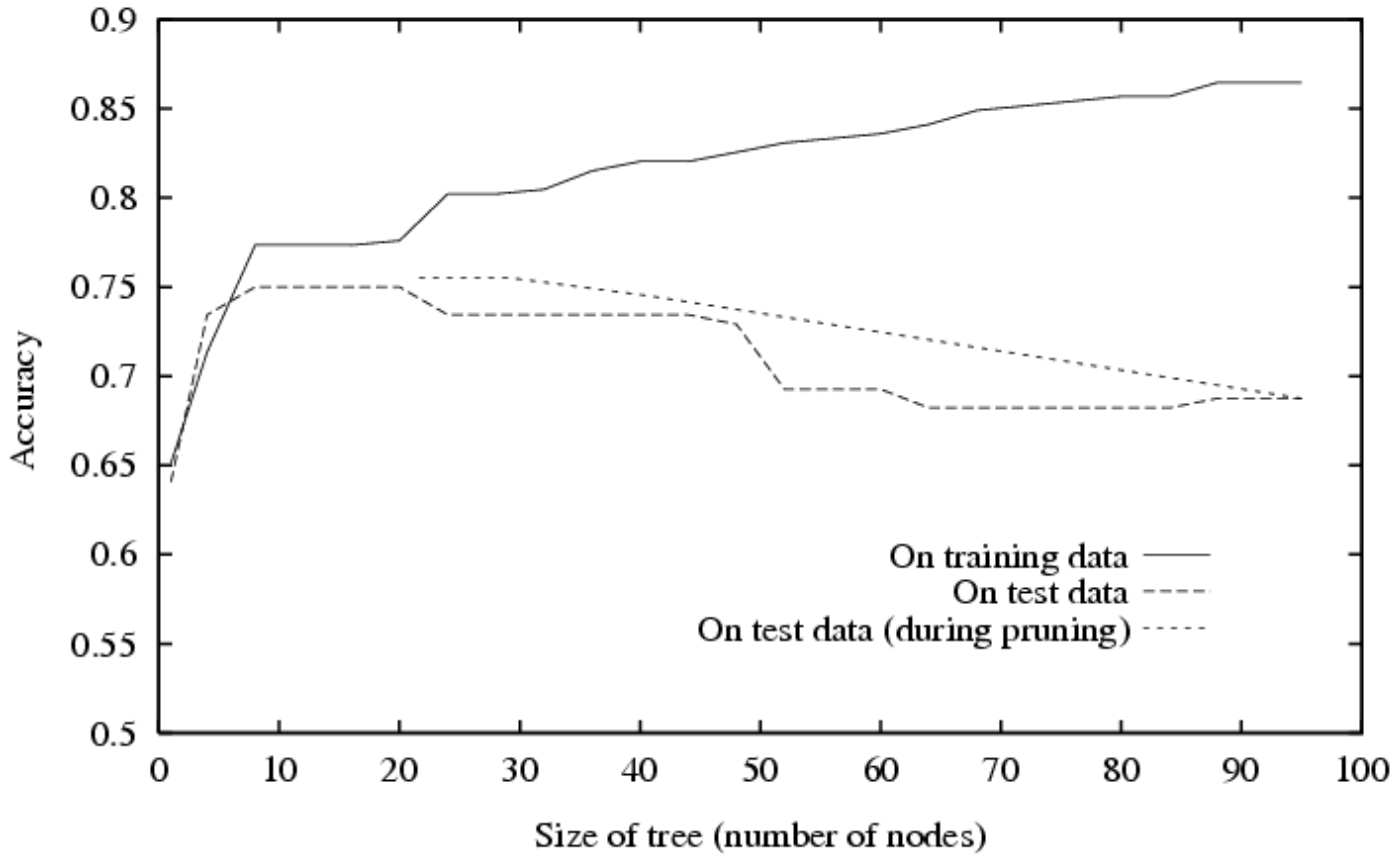
Create tree that classifies *training* set correctly

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?

Effect of Reduced-Error Pruning



Continuous Valued Attributes

Create a discrete attribute to test continuous

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use $GainRatio$ instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

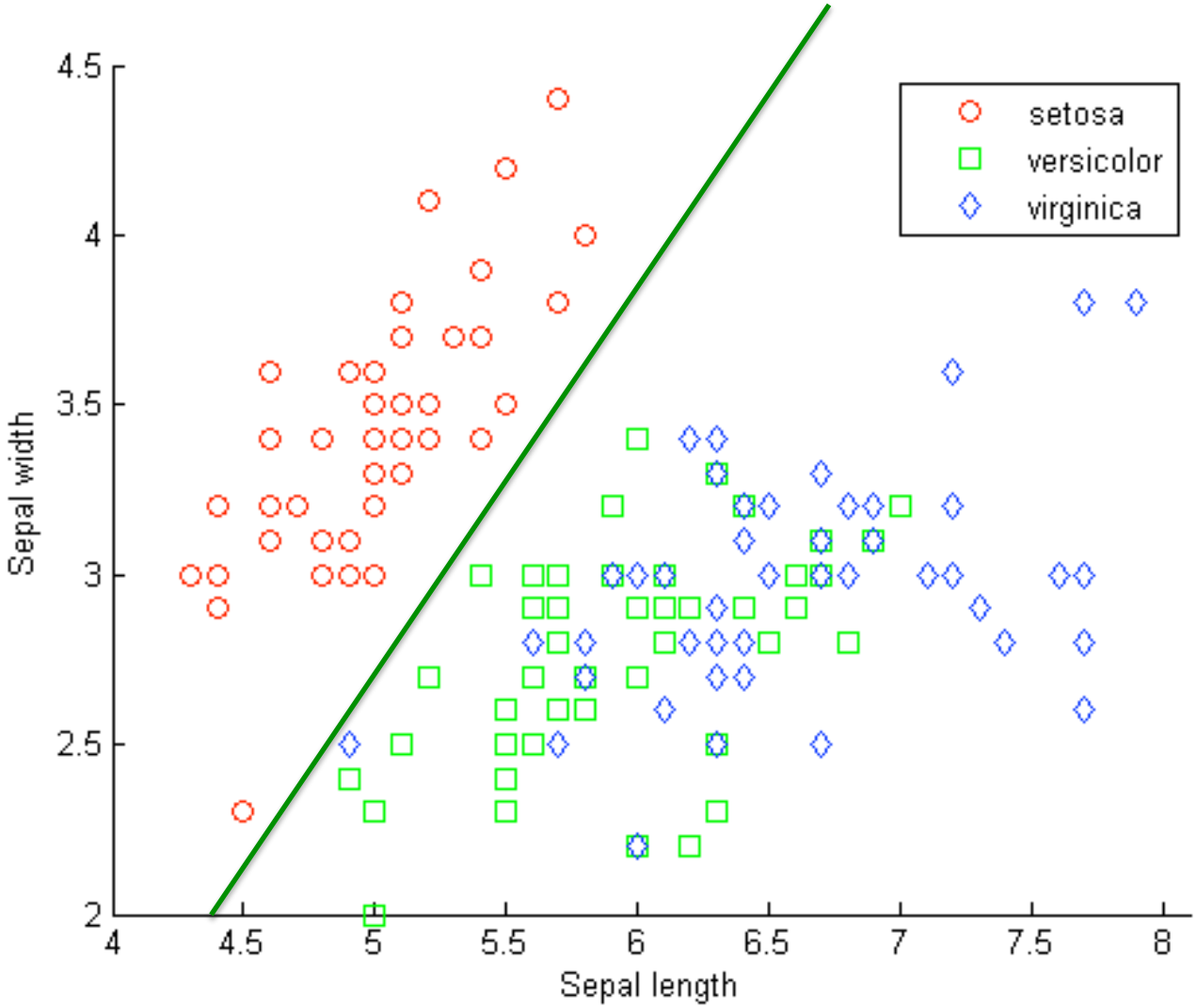
$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

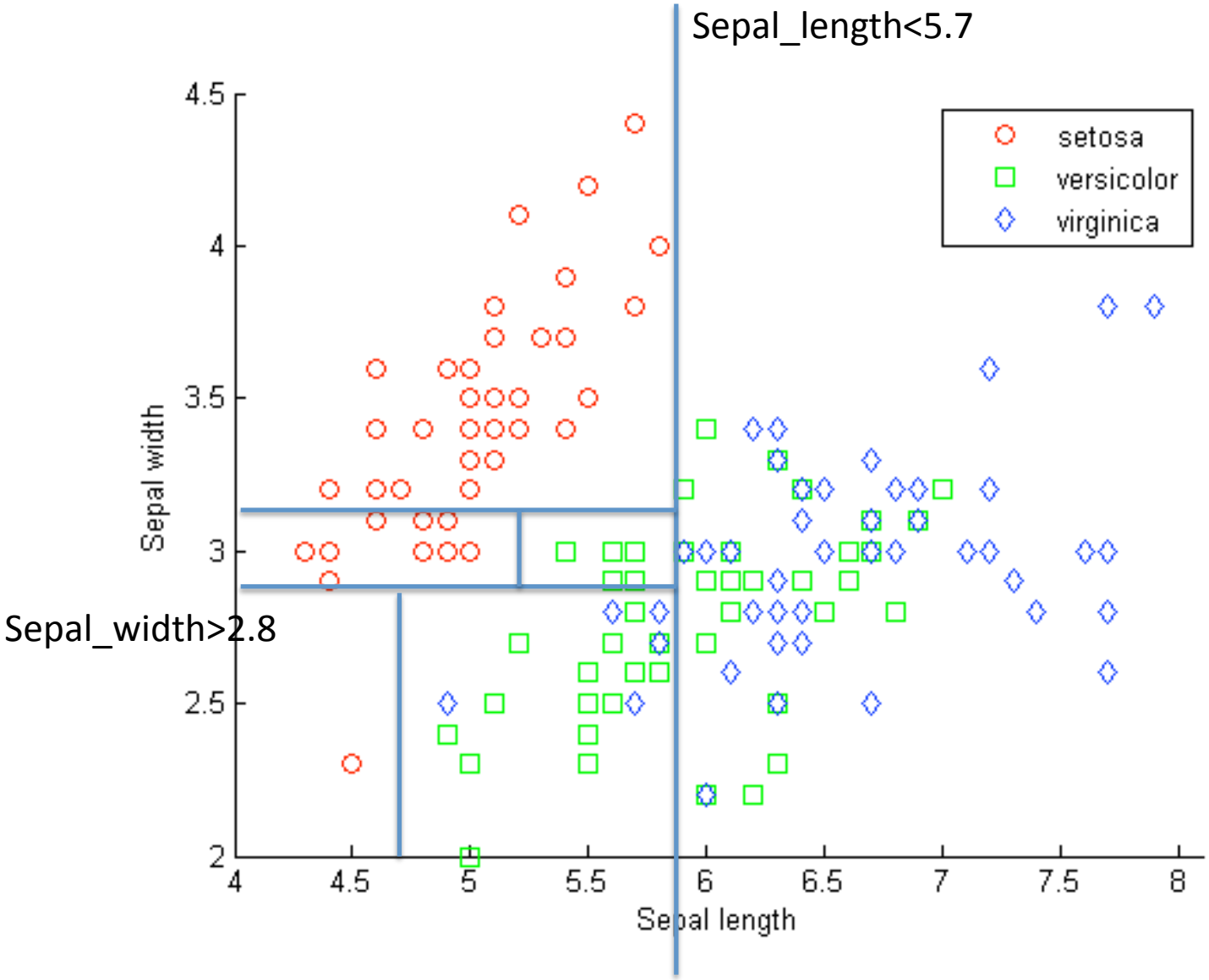
Decision Tree and Linear Classifier

- Decision trees don't (typically) improve over linear classifiers when you have lots of features
- Sometimes fail badly on problems that linear classifiers perform well on
  - here's an example

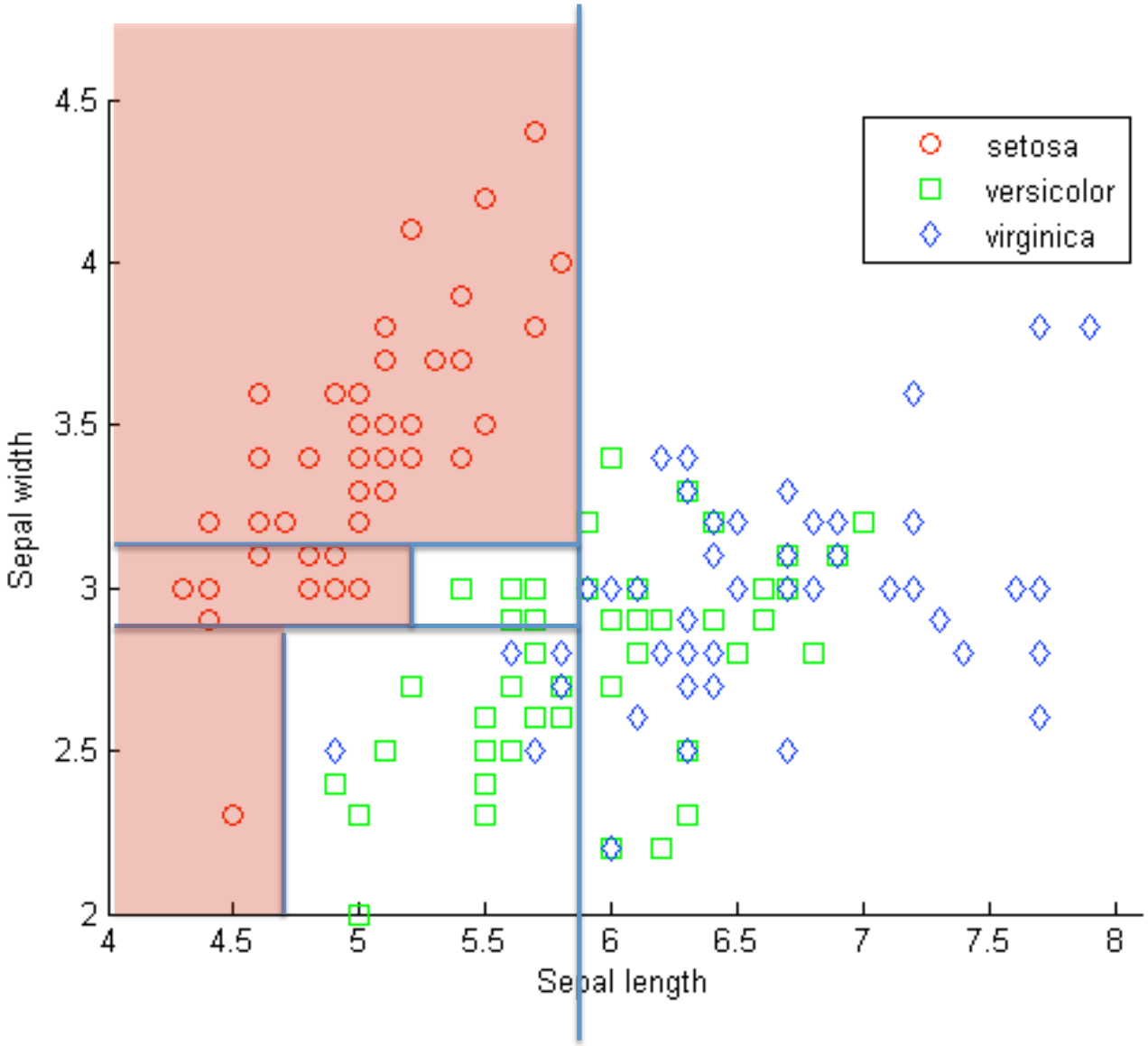
Another view of a decision tree



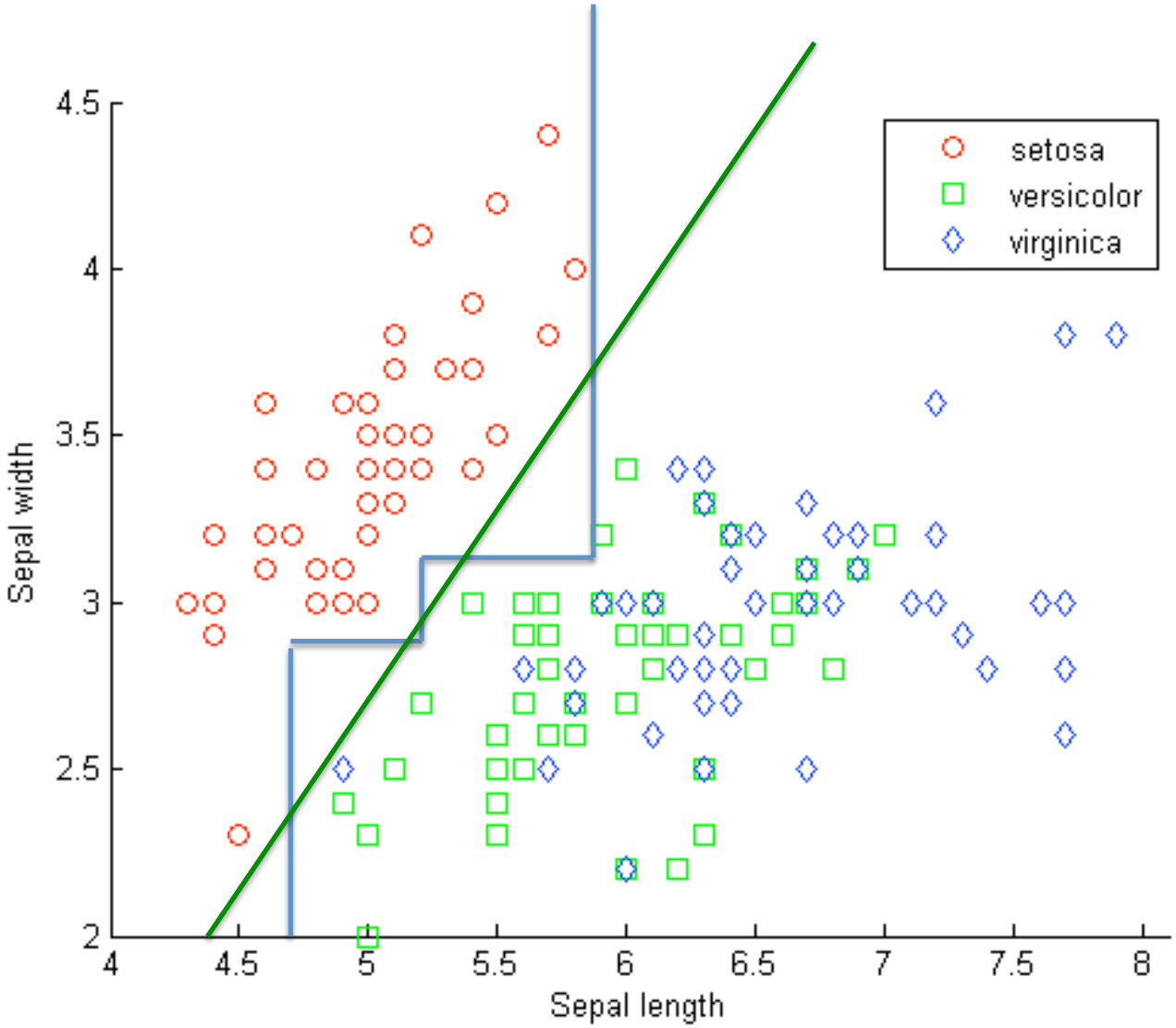
Another view of a decision tree



Another view of a decision tree



Another view of a decision tree



Questions to think about (1)

- Consider target function f: <x1,x2> → y, where x1 and x2 are real-valued, y is boolean. What is the set of decision surfaces describable with decision trees that use each attribute at most once?

Questions to think about (2)

- What is the relationship between learning decision trees, and learning IF-THEN rules

One of 18 learned rules:

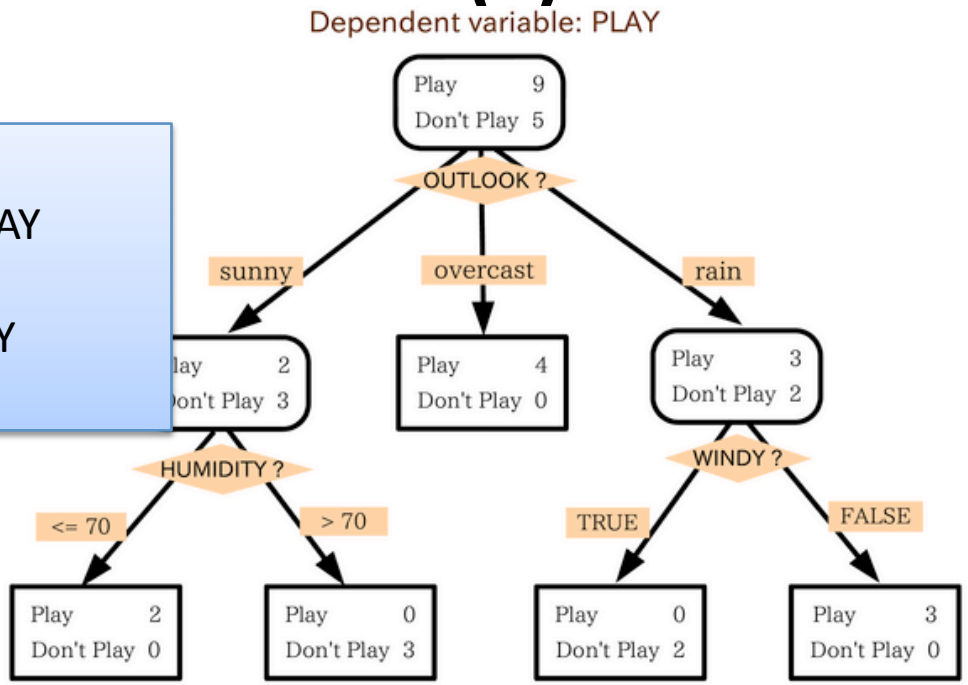
```
If   No previous vaginal delivery, and
     Abnormal 2nd Trimester Ultrasound, and
     Malpresentation at admission
Then Probability of Emergency C-Section is 0.6

 Over training data: 26/41 = .63,
 Over test data: 12/20 = .60
```

Questions to think about (3)

if O=sunny and H<= 70 then PLAY
else if O=sunny and H>70 then DON'T_PLAY
else if O=overcast then PLAY
else if O=rain and windy then DON'T_PLAY
else if O=rain and !windy then PLAY

One rule per leaf in the tree



Simpler rule set

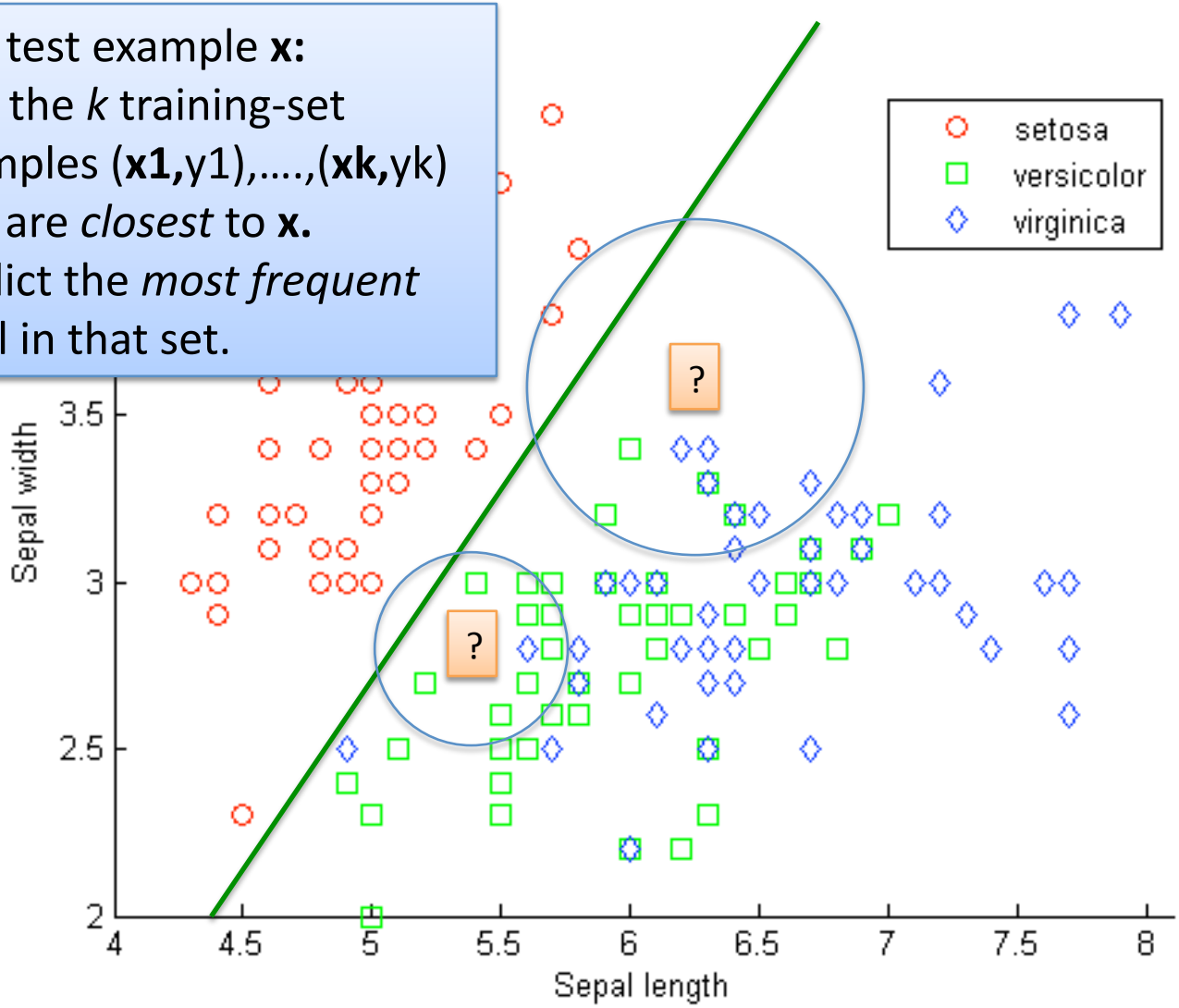
if O=sunny and H> 70 then DON'T_PLAY
else if O=rain and windy then DON'T_PLAY
else PLAY

Nearest Neighbor Learning

k-nearest neighbor learning

Given a test example **x:**

1. Find the *k* training-set examples (**x1,**y1),….,(**xk,**yk) that are *closest* to **x.**
2. Predict the *most frequent* label in that set.



Breaking it down:

- To train:
  - save the data
- To test:
  - For each test example **x:**

Very fast!

…you might build some indices….

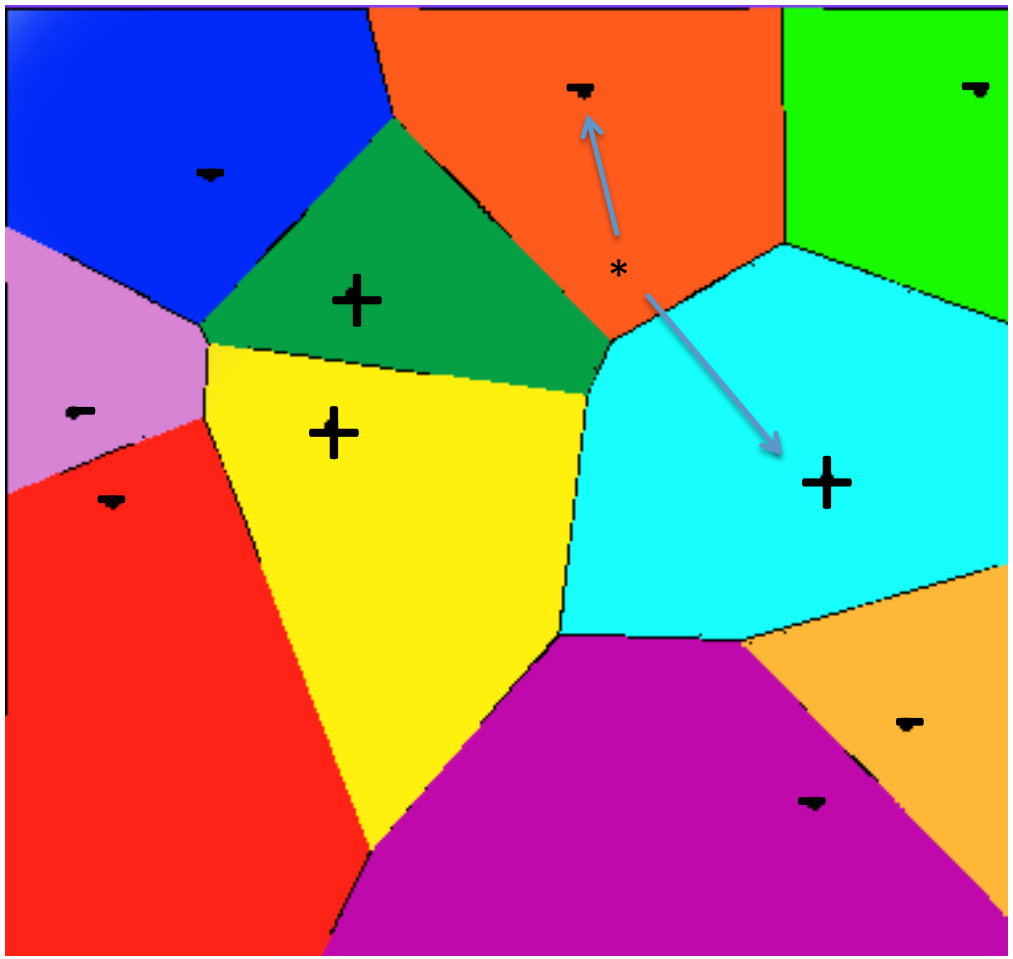
1. Find the *k* training-set examples (**x1,**y1), ….,(**xk,**yk) that are *closest* to **x.**
2. Predict the *most frequent* label in that set.

Prediction is relatively slow (compared to a linear classifier or decision tree)

What is the decision boundary for 1-NN?

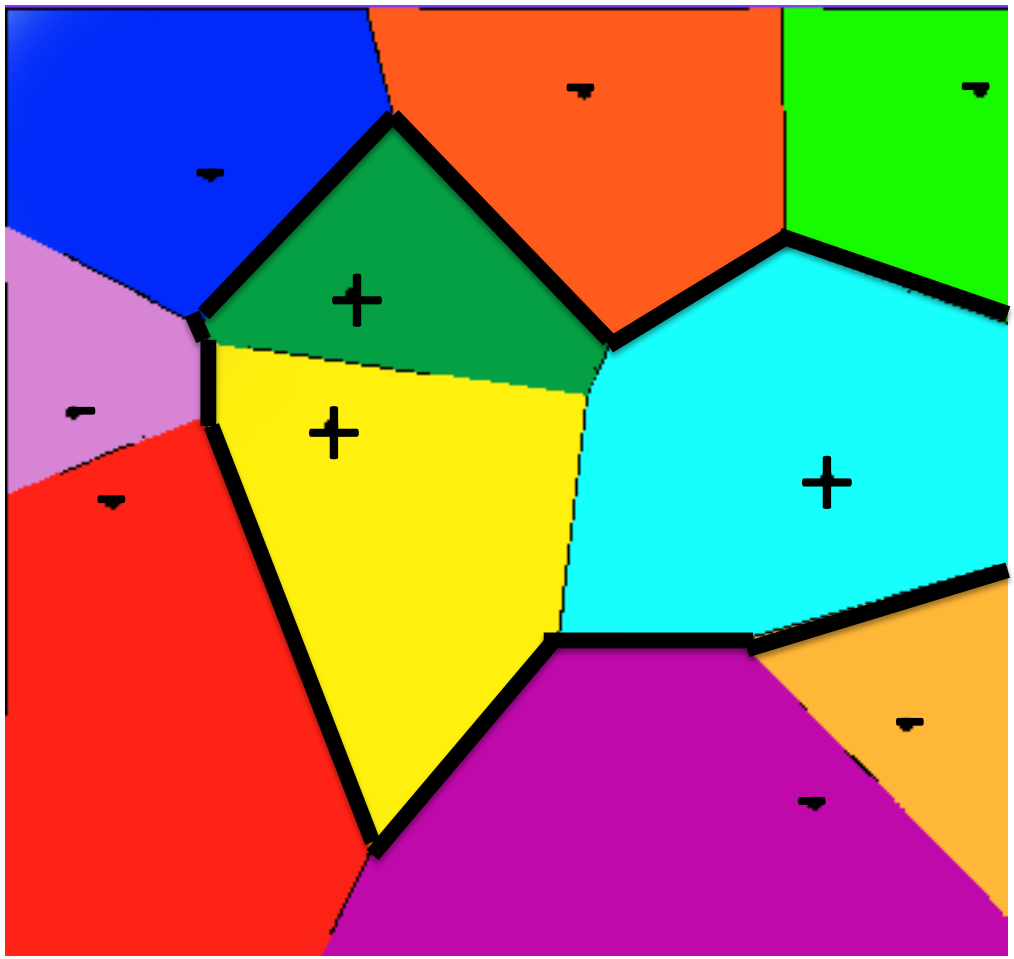
Each cell C_i is the set of all points that are closest to a particular example $\mathbf{x}_i$

Voronoi Diagram



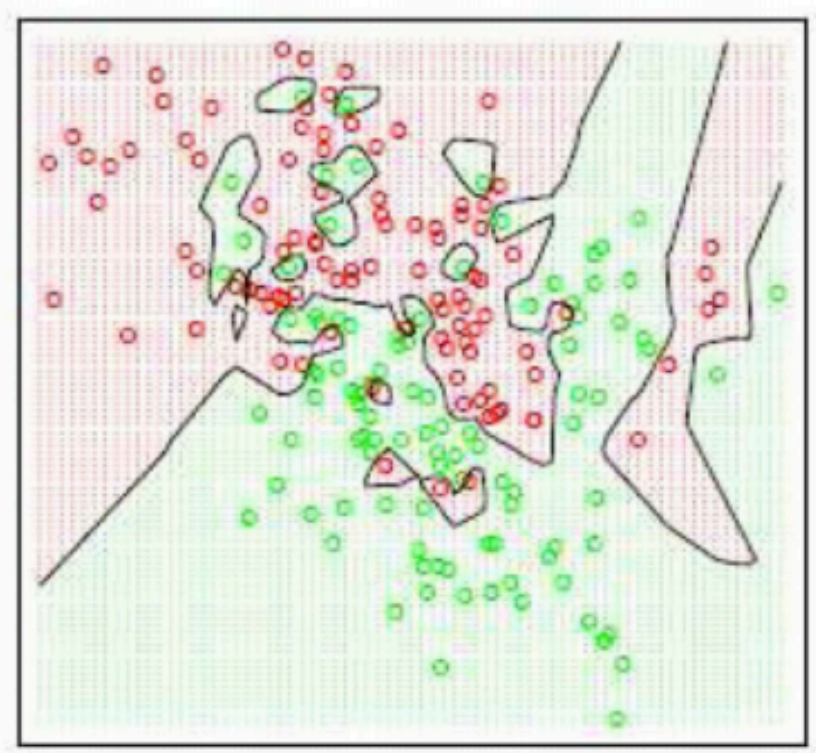
What is the decision boundary for 1-NN?

Voronoi Diagram



Effect of k on decision boundary

k=1



Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Some common variants

- Distance metrics:
  - Euclidean distance: **||x1 - x2||**
  - Cosine distance: 1 - **<x1,x2>/||x1||*||x2||**
    - this is in [0,1]
- Weighted nearest neighbor:
  - Instead of most frequent *y* in k-NN predict

$$\text{argmax}_y \sum_{(\mathbf{x}_i, y) \in kNN(\mathbf{x})} sim(\mathbf{x}_i, \mathbf{x})$$

You should know:

- Well posed function approximation problems:
  - Instance space, X
  - Sample of labeled training data { $\langle x^{(i)}, y^{(i)} \rangle$ }
  - Hypothesis space, H = { f: X→Y }
  - Learning is a search/optimization problem over H

- Decision tree learning
  - Greedy top-down learning of decision trees (ID3, C4.5, ...)
  - Overfitting and tree/rule post-pruning
  - Extensions…

- kNN classifier
  - Non-linear decision boundary
  - Low-cost training, high-cost prediction