http://poloclub.gatech.edu/cse6242

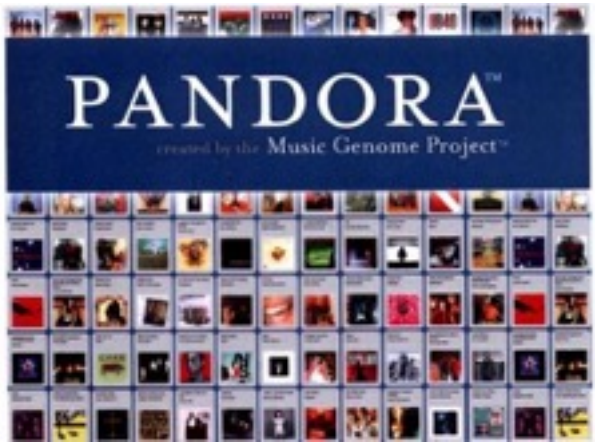
CSE6242 / CX4242: Data & Visual Analytics

Classification

Duen Horng (Polo) Chau
Assistant Professor
Associate Director, MS Analytics
Georgia Tech

Parishit Ram
GT PhD alum; SkyTree

Partly based on materials by
Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Parishit Ram (GT PhD alum; SkyTree), Alex Gray



How will I rate "Chopin's 5th Symphony"?

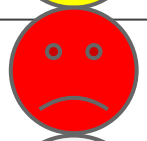
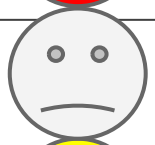
Songs	Label
Some nights	
Skyfall	
Comfortably numb	
We are young	
…	…
…	…
Chopin's 5th	???

Classification

What tools do you need for classification?

1. **Data** $S = \{(x_i, y_i)\}_{i = 1,...,n}$
   - x_i represents each example with d attributes
   - y_i represents the label of each example
2. Classification **model** $f_{(a,b,c,....)}$ with some parameters *a, b, c,...*
   - a model/function maps examples to labels
3. **Loss function** $L(y, f(x))$
   - how to penalize mistakes

Features $x_i = (x_{i1}, \ldots, x_{id})$

Song name	Label	Artist	Length	...
Some nights		Fun	4:23	...
Skyfall		Adele	4:00	...
Comf. numb		Pink Fl.	6:13	...
We are young		Fun	3:50	...
...	...	...	...	...
...	...	...	...	...
Chopin's 5th	??	Chopin	5:32	...

$$x_i = (x_{i1}, \ldots, x_{id}); y_i = \{1, \ldots, m\}$$

Training a classifier (building the "model")

Q: How do you learn appropriate values for parameters *a, b, c, ...* such that

- $y_i = f_{(a,b,c,\ldots)}(x_i)$, $i = 1, \ldots, n$
  - Low/no error on "training data" (songs)
- $y = f_{(a,b,c,\ldots)}(x)$, for any new x
  - Low/no error on "test data" (songs)

Possible A: Minimize $\sum_{i=1}^{n} L(y_i, f_{(a,b,c,..)}(x_i))$ with respect to *a, b, c,...*

Classification loss function

Most common loss: **0-1 loss function**

$$L_{0-1}(y, f(x)) = \mathbb{I}(y \neq f(x))$$

More general loss functions are defined by a *m x m* cost matrix *C* such that

$$L(y, f(x)) = C_{ab}$$

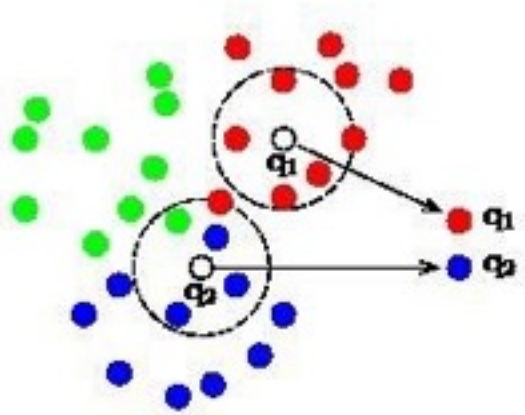
where $y = a$ and $f(x) = b$

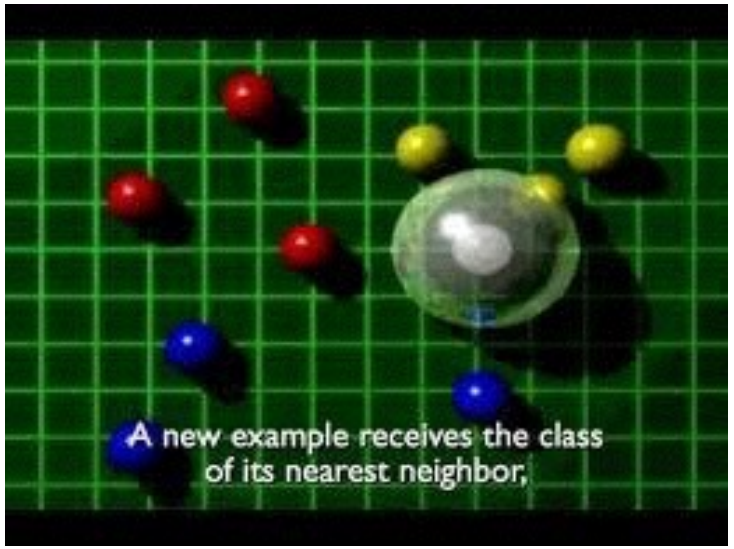
Class	T0	T1
P0	0	C_{10}
P1	C_{01}	0

T0 (true class 0), **T1** (true class 1)

P0 (predicted class 0), **P1** (predicted class 1)

k-Nearest-Neighbor Classifier





The classifier:

f(x) = majority label of the *k* nearest neighbors (NN) of *x*

Model parameters:

- Number of neighbors *k*
- Distance/similarity function *d(.,.)*

But KNN is so simple!

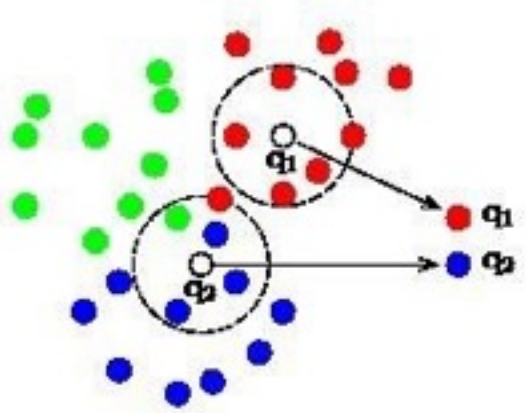
It can work really well! Pandora uses it:
https://goo.gl/foLfMP
(from the book “Data Mining for Business Intelligence”)

k-Nearest-Neighbor Classifier

If *k* and *d(.,.)* are fixed

Things to learn: ?

How to learn them: ?



If *d(.,.)* is fixed, but you can change *k*

Things to learn: ?

How to learn them: ?

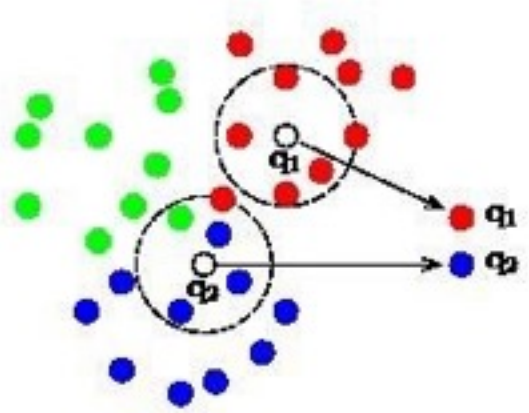
$$x_i = (x_{i1}, \ldots, x_{id}); y_i = \{1, \ldots, m\}$$

k-Nearest-Neighbor Classifier

If *k* and *d(.,.)* are fixed

Things to learn: Nothing

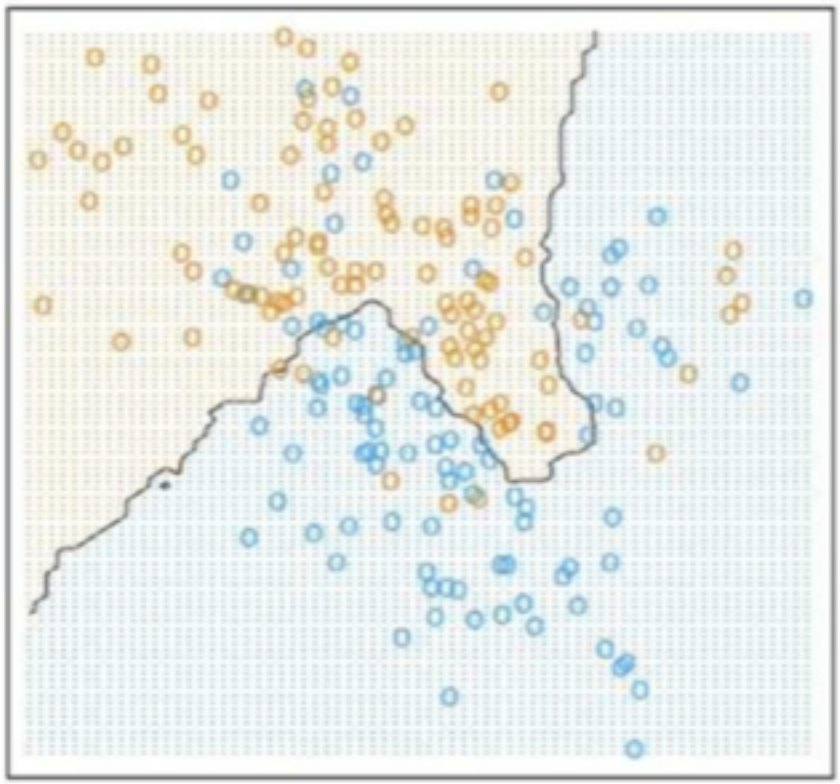
How to learn them: N/A



If *d(.,.)* is fixed, but you can change *k*

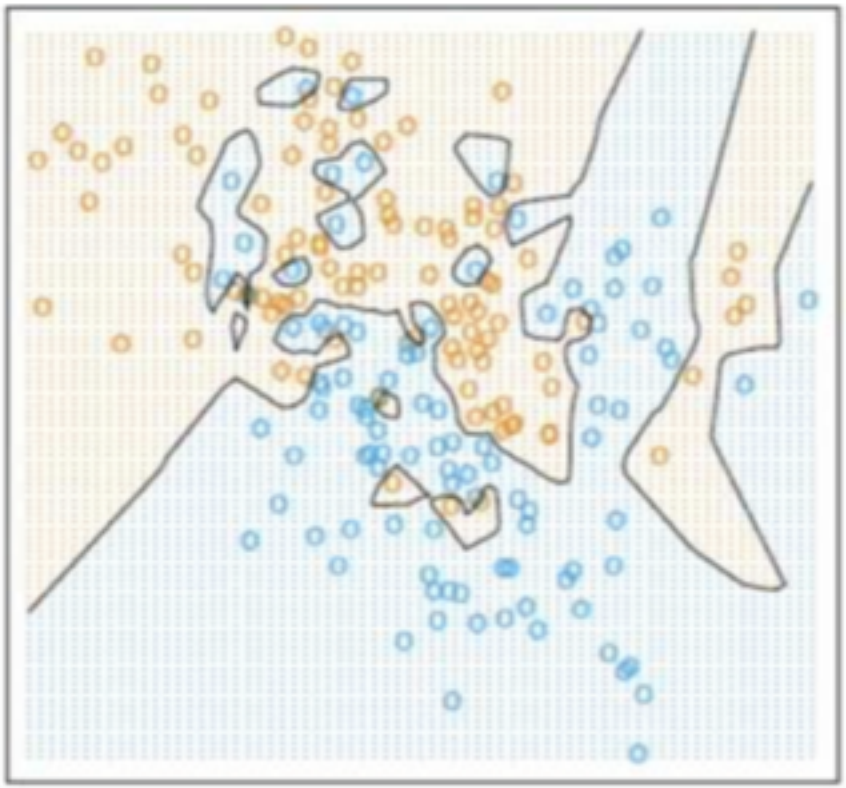
Selecting *k*: Try different values of *k* on some hold-out set

15-NN



Pretty good!

1-NN

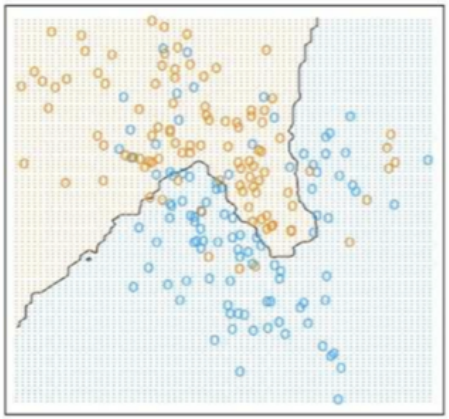


Overfitted

How to find the best k in K-NN?

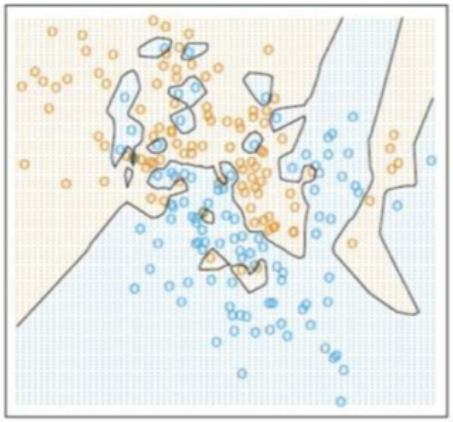
Use cross validation.

15-NN



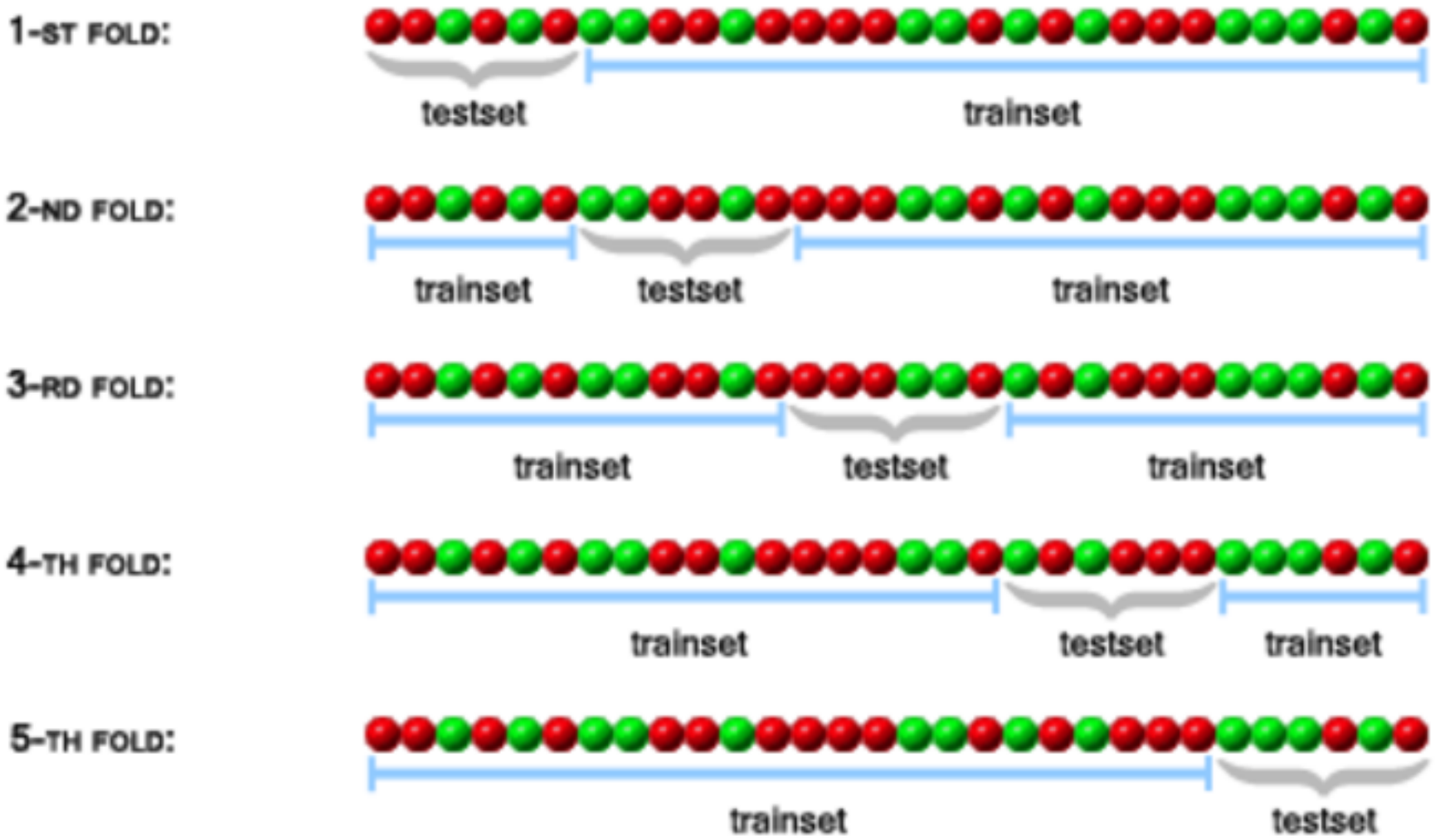
Pretty good!

1-NN



Overfitted

Example, evaluate k = 1 (in K-NN) using 5-fold cross-validation



Cross-validation (C.V.)

1. Divide your data into n parts
2. Hold 1 part as “test set” or “hold out set”
3. Train classifier on remaining n-1 parts “training set”
4. Compute test error on test set
5. Repeat above steps n times, once for each n-th part
6. Compute the average test error over all n folds
   (i.e., cross-validation test error)

Cross-validation variations

Leave-one-out cross-validation (LOO-CV)

- test sets of size 1

K-fold cross-validation

- Test sets of size *(n / K)*
- K = 10 is most common (i.e., 10 fold CV)

$x_i = (x_{i1}, \ldots, x_{id}); y_i = \{1, \ldots, m\}$

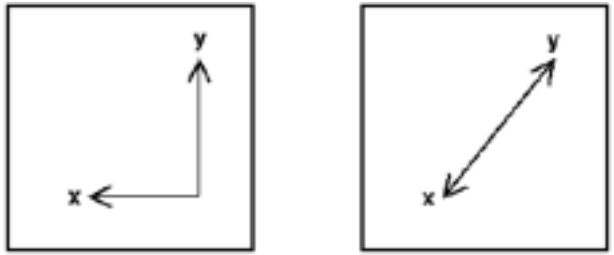
k-Nearest-Neighbor Classifier

If *k* is fixed, but you can change *d(.,.)*

Things to learn: ?

How to learn them: ?

Cross-validation: ?



Possible distance functions:

- Euclidean distance: $\|x_i - x_j\|_2 = \sqrt{(x_i - x_j)^\top (x_i - x_j)}$
- Manhattan distance: $\|x_i - x_j\|_1 = \sum_{l=1}^{d} |x_{il} - x_{jl}|$
- …

$$x_i = (x_{i1}, \ldots, x_{id}); y_i = \{1, \ldots, m\}$$

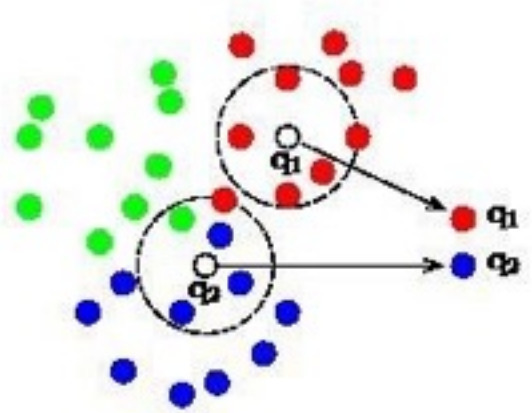
k-Nearest-Neighbor Classifier

If *k* is fixed, but you can change *d(.,.)*

Things to learn: distance function *d(.,.)*

How to learn them: optimization

Cross-validation: any regularizer you have on your distance function



Summary on k-NN classifier

- Advantages
  - Little learning (unless you are learning the distance functions)
  - quite powerful in practice (and has theoretical guarantees as well)
- Caveats
  - Computationally expensive at test time

Reading material:

- ESL book, Chapter 13.3 http://www-stat.stanford.edu/~tibs/ElemStatLearn/printings/ESLII_print10.pdf
- Le Song's slides on kNN classifier http://www.cc.gatech.edu/~lsong/teaching/CSE6740/lecture2.pdf

Points about cross-validation

Requires extra computation, but gives you information about expected test error

LOO-CV:

- Advantages
  - Unbiased estimate of test error (especially for small *n*)
  - Low variance
- Caveats
  - Extremely time consuming

Points about cross-validation

K-fold CV:

- Advantages
  - More efficient than LOO-CV
- Caveats
  - *K* needs to be large for low variance
  - Too small *K* leads to **under-use** of data, leading to higher bias
- Usually accepted value ***K* = 10**

Reading material:

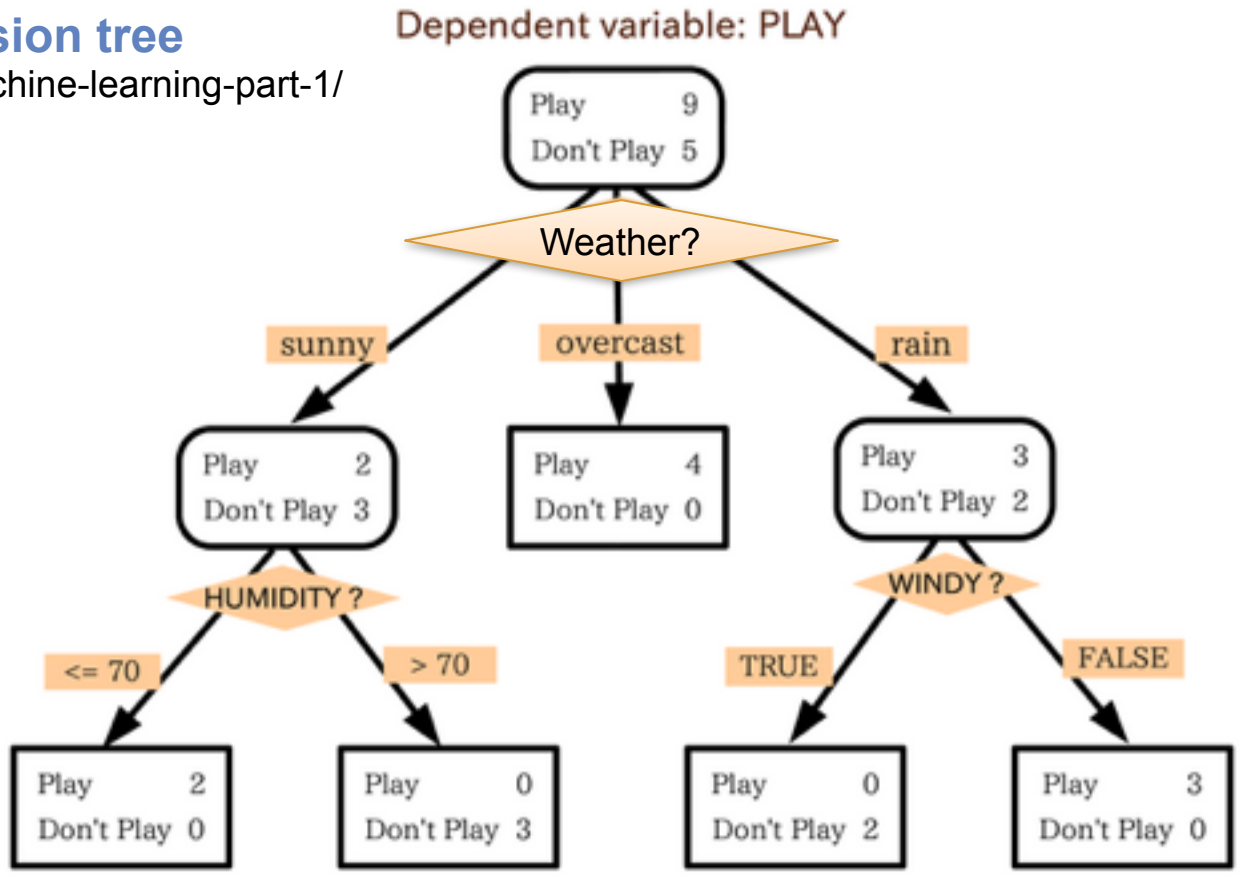
- ESL book, Chapter 7.10 http://www-stat.stanford.edu/~tibs/ElemStatLearn/printings/ESLII_print10.pdf
- Le Song's slides on CV http://www.cc.gatech.edu/~lsong/teaching/CSE6740/lecture13-cv.pdf

$$x_i = (x_{i1}, \ldots, x_{id}); y_i = \{1, \ldots, m\}$$

Decision trees (DT)

Visual introduction to decision tree

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/



The classifier:

$f_T(x)$: majority class in the leaf in the tree T containing x

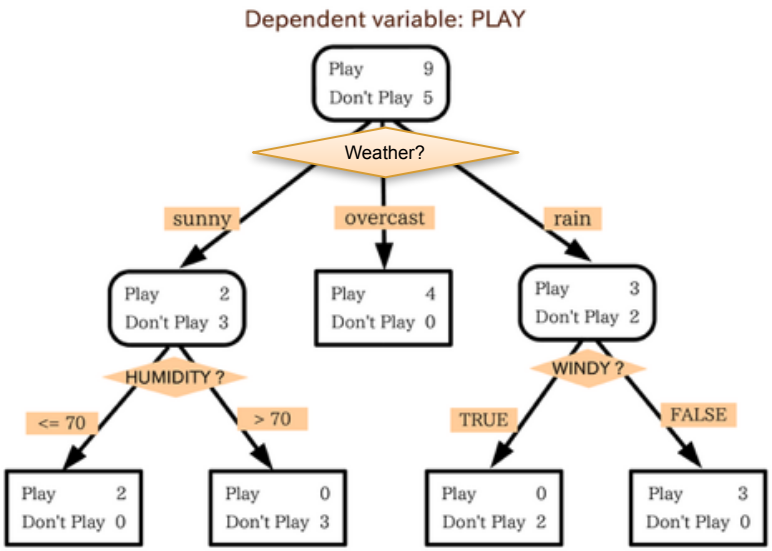
Model parameters: The tree structure and size

Decision trees

Things to learn: ?

How to learn them: ?

Cross-validation: ?



Learning the Tree Structure

Things to learn: the tree structure

How to learn them: (greedily) minimize the overall classification loss

Cross-validation: finding the best sized tree with *K*-fold cross-validation

Decision trees

Pieces:

1. Find the best split on the chosen attribute
2. Find the best attribute to split on
3. Decide on when to stop splitting
4. Cross-validation

Highly recommended lecture slides from CMU

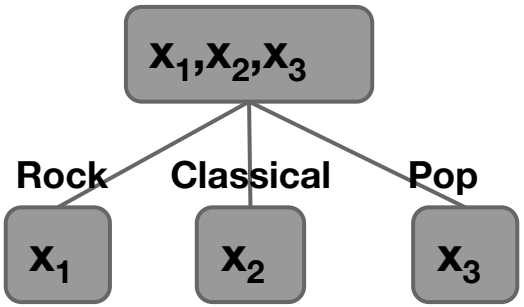
http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-s06/www/DTs.pdf

$$x_i = (x_{i1}, \ldots, x_{id}); y_i = \{1, \ldots, m\}$$

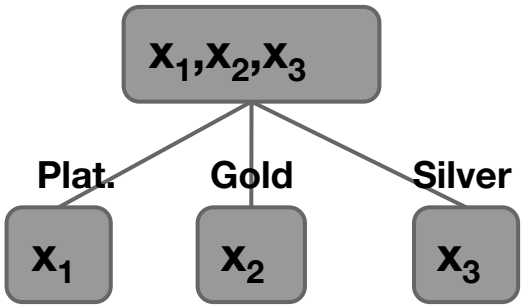
Choosing the split

Split types for a selected attribute j:

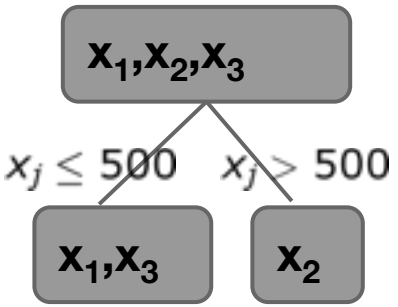
1. Categorical attribute (e.g. "genre")
   x_{1j} = Rock, x_{2j} = Classical, x_{3j} = Pop
2. Ordinal attribute (e.g., "achievement")
   x_{1j}=Platinum, x_{2j}=Gold, x_{3j}=Silver
3. Continuous attribute (e.g., song duration)
   x_{1j} = 235, x_{2j} = 543, x_{3j} = 378



Split on genre



Split on achievement



Split on duration

$$x_i = (x_{i1}, \ldots, x_{id}); y_i = \{1, \ldots, m\}$$

Choosing the split

At a node T for a given attribute d,

select a split s as following:

$$min_s\ loss(T_L) + loss(T_R)$$

where $loss(T)$ is the loss at node T

Common node loss functions:

- Misclassification rate
- Expected loss
- Normalized negative log-likelihood (= cross-entropy)

More details on loss functions, see Chapter 3.3:
http://www.stat.cmu.edu/~cshalizi/350/lectures/22/lecture-22.pdf

Choosing the attribute

Choice of attribute:

1. Attribute providing the maximum improvement in training loss
2. Attribute with highest information gain

   (mutual information)

Intuition: an attribute with highest information gain helps most rapidly describe an instance (i.e., most rapidly reduces "uncertainty")

More details about information gain
https://en.wikipedia.org/wiki/Information_gain_in_decision_trees

When to stop splitting?

1. Homogenous node (all points in the node belong to the same class OR all points in the node have the same attributes)
2. Node size less than some threshold
3. Further splits provide no improvement in training loss
   (*loss(T) <= loss(T_L) + loss(T_R)*)

Controlling tree size

In most cases, you can drive training error to zero (how? is that a good thing?)

What is wrong with really deep trees?

- Really high "variance"

What can be done to control this?

- *Regularize* the tree complexity
  - Penalize complex models and prefers simpler models

Look at Le Song's slides on the decomposition of error in bias and variance of the estimator http://www.cc.gatech.edu/~lsong/teaching/CSE6740/lecture13-cv.pdf

Summary on decision trees

- Advantages
  - Easy to implement
  - Interpretable
  - Very fast test time
  - Can work seamlessly with mixed attributes
  - Works quite well in practice
- Caveats
  - Can be too simplistic (but OK if it works)
  - Training can be very expensive
  - Cross-validation is hard (node-level CV)

Final words on decision trees

Reading material:

- ESL book, Chapter 9.2 http://www-stat.stanford.edu/~tibs/ElemStatLearn/printings/ESLII_print10.pdf
- Le Song's slides http://www.cc.gatech.edu/~lsong/teaching/CSE6740/lecture6.pdf