http://poloclub.gatech.edu/cse6242

CSE6242 / CX4242: Data & Visual Analytics

Classification Key Concepts

Duen Horng (Polo) Chau

Associate Professor

Associate Director, MS Analytics

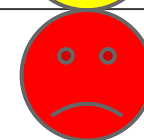
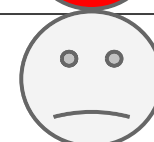
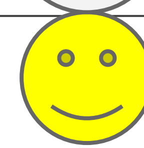
Machine Learning Area Leader, College of Computing

Georgia Tech

Partly based on materials by

Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Parishit Ram, Alex Gray

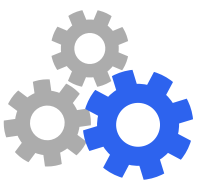
How will I rate "Chopin's 5th Symphony"?

Songs	Like?
Some nights	
Skyfall	
Comfortably numb	
We are young	
...	...
...	...
Chopin's 5th	???

Classification

What tools do you need for classification?

1. **Data** $S = \{(x_i, y_i)\}_{i = 1,\ldots,n}$
   - x_i : data example with d attributes
   - y_i : label of example (what **you** care about)



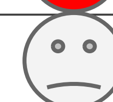
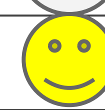
2. Classification **model** $f_{(a,b,c,\ldots.)}$ with some parameters $a, b, c,\ldots$
3. **Loss function** $L(y, f(x))$
   - how to penalize mistakes

Terminology Explanation

data example = data instance
attribute = feature = dimension
label = target attribute

Data $S = \{(x_i, y_i)\}_{i = 1,\dots,n}$

- x_i : data example with d attributes $x_i = (x_{i1}, \dots, x_{id})$
- y_i : label of example

Song name	**Artist**	**Length**	**...**	**Like?**
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	...	??

What is a "model"?

"a simplified representation of reality created to serve a purpose" *Data Science for Business*

Example: maps are abstract models of the physical world

There can be many models!!

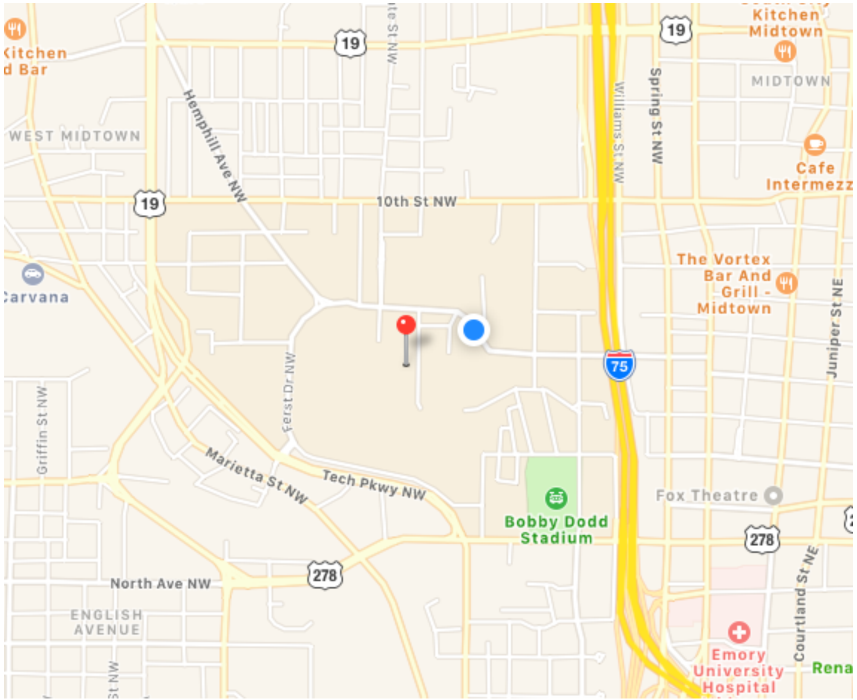
(Everyone sees the world differently, so each of us has a different model.)

In data science, a model is **formula to estimate what you care about**. The formula may be mathematical, a set of rules, a combination, etc.

Training a classifier = building the "model"

How do you learn appropriate values for parameters *a, b, c, ...* ?

Analogy: how do you know your map is a "good" map of the physical world?



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 $f(x) = b$

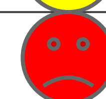
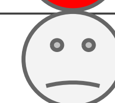
Class	P0	P1
T0	0	C_{10}
T1	C_{01}	0

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

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

An ideal model should correctly estimate:

- known or seen data examples’ labels
- unknown or unseen data examples’ labels

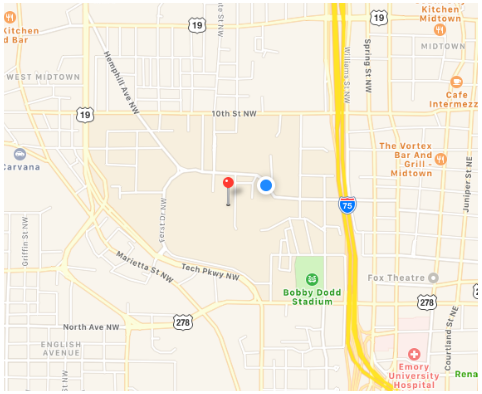
Song name	Artist	Length	...	Like?
Some nights	Fun	4:23	...	
Skyfall	Adele	4:00	...	
Comf. numb	Pink Fl.	6:13	...	
We are young	Fun	3:50	...	
...	...	...	...	...
...	...	...	...	...
Chopin's 5th	Chopin	5:32	...	??

Training a classifier = building the "model"

Q: How do you learn appropriate values for parameters *a, b, c, ...* ?
(Analogy: how do you know your map is a "good" map?)

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

It is very easy to achieve perfect classification on training/seen/known data. Why?



If your model works really well for *training* data, but poorly for *test* data, your model is "overfitting".

How to avoid overfitting?

Example: one run of *5-fold* cross validation

You should do a **few runs** and **compute the average**
(e.g., error rates if that's your evaluation metrics)

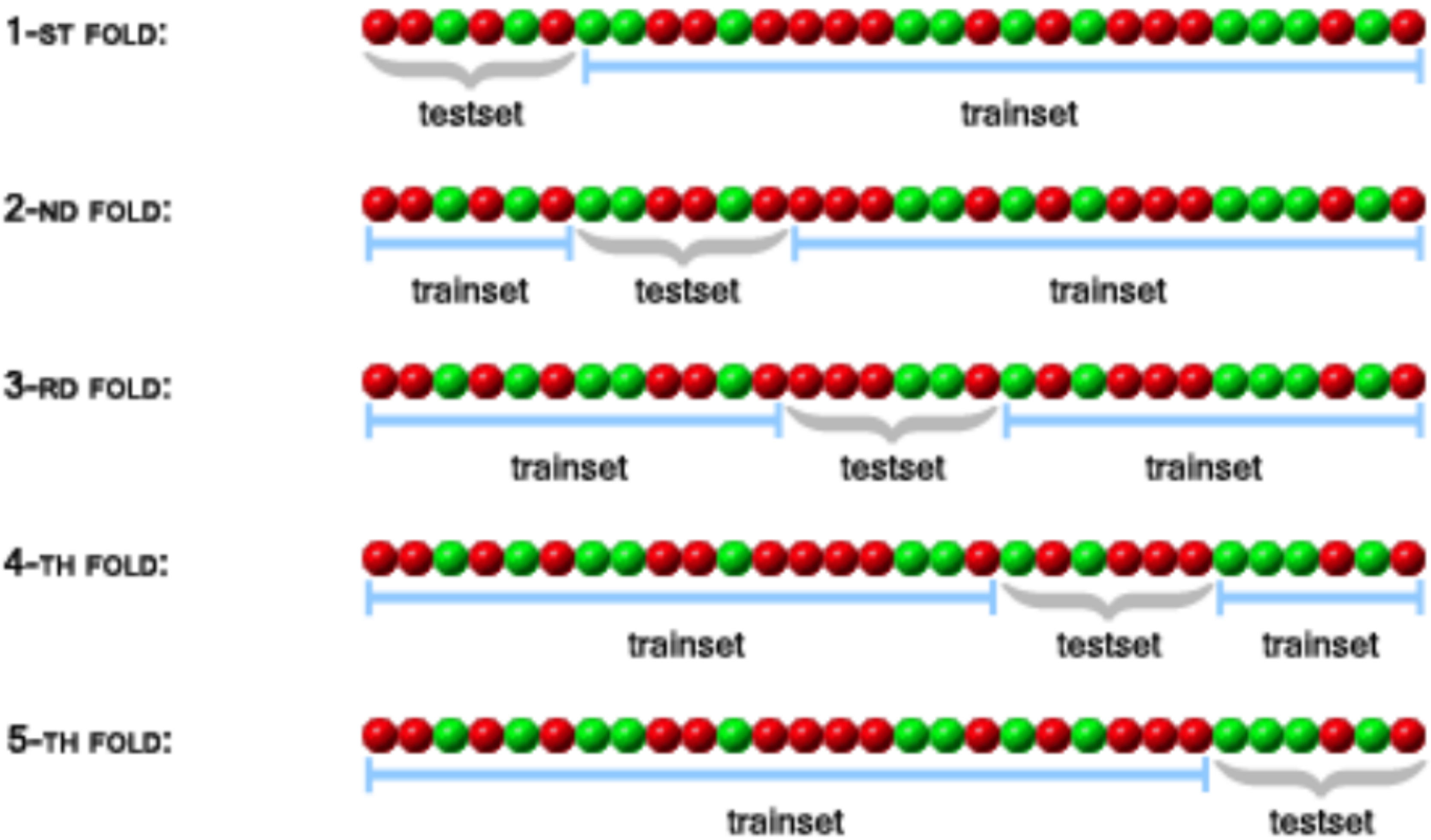
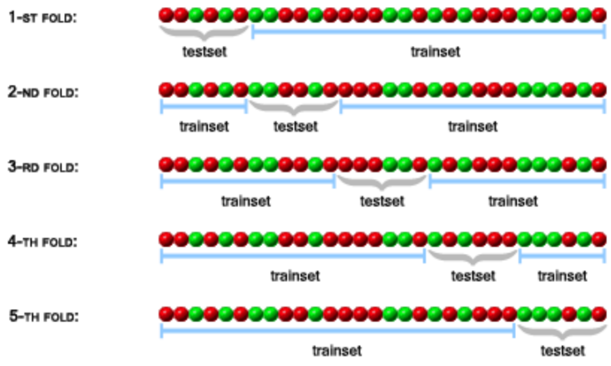


Image credit: http://stats.stackexchange.com/questions/1826/cross-validation-in-plain-english

Cross validation



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

K-fold cross-validation

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

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

- test sets of size 1

Example:
k-Nearest-Neighbor classifier

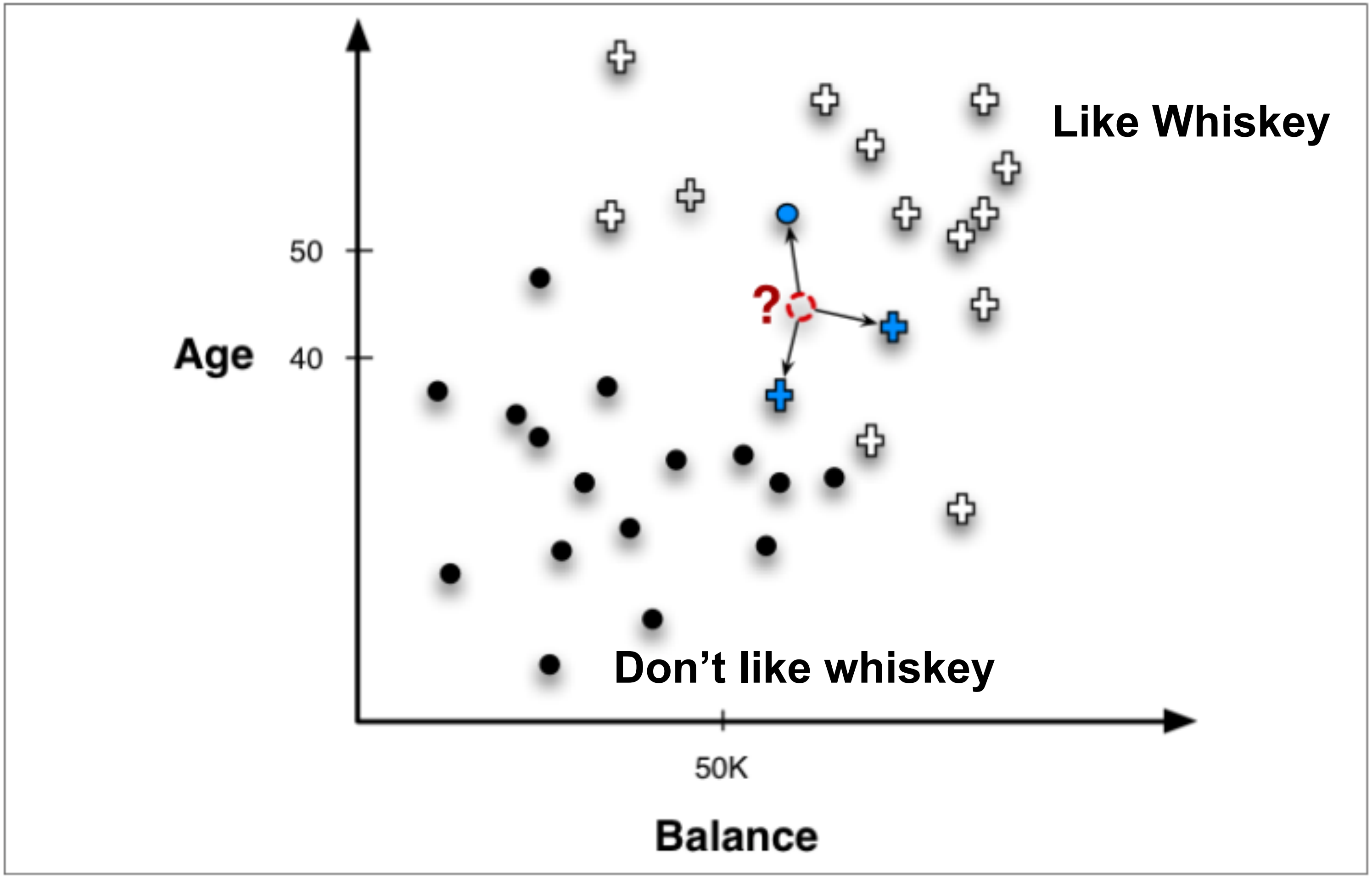


Figure 6-2. Nearest neighbor classification. The point to be classified, labeled with a question mark, would be classified + because the majority of its nearest (three) neighbors are +.

Image credit: Data Science for Business

But k-NN is so simple!

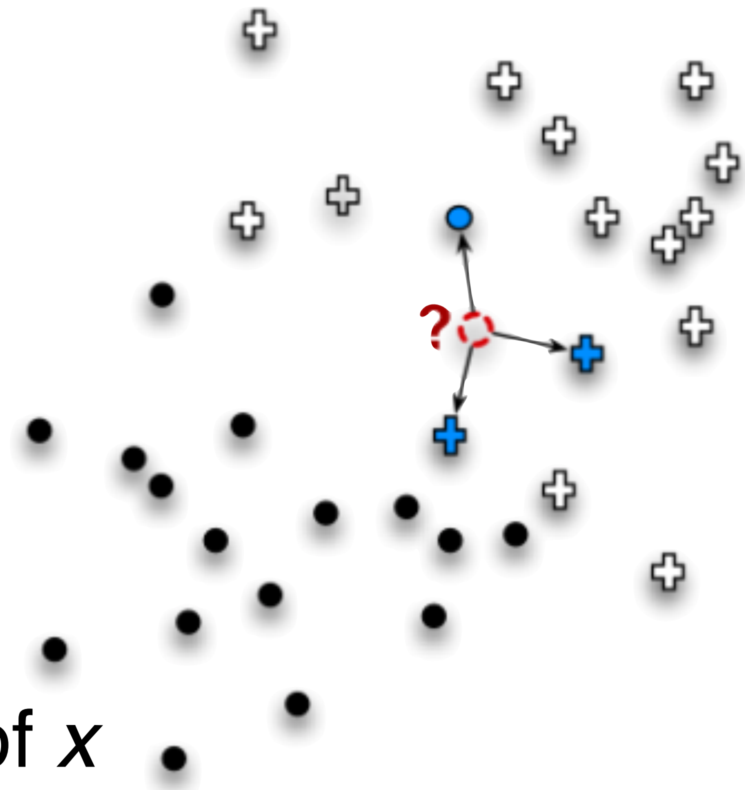
It can work really well! **Pandora** (acquired by SiriusXM) uses it or has used it: https://goo.gl/foLfMP
(from the book “Data Mining for Business Intelligence”)

Image credit: https://www.fool.com/investing/general/2015/03/16/will-the-music-industry-end-pandoras-business-mode.aspx

What are good models?

Simple (few parameters)	Effective	🤗
Complex (more parameters)	Effective (if significantly more so than simple methods)	🤔
Complex (many parameters)	Not-so-effective	😱

k-Nearest-Neighbor Classifier



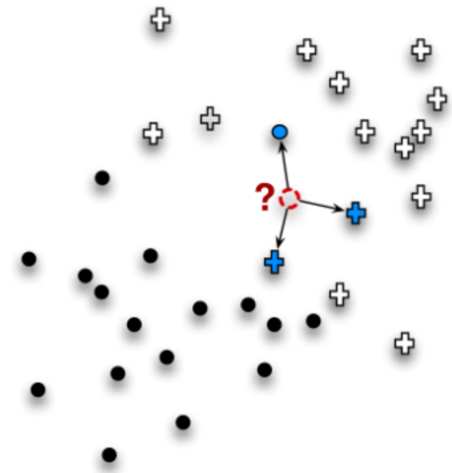
The classifier:

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

Model parameters:

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

k-Nearest-Neighbor Classifier



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

Things to learn: ?

How to learn them: ?

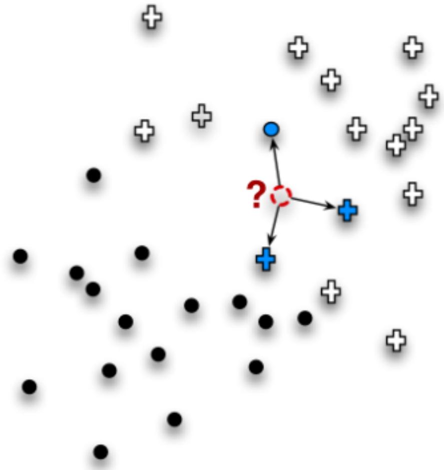
If *d(.,.)* is fixed, but you can change *k*

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

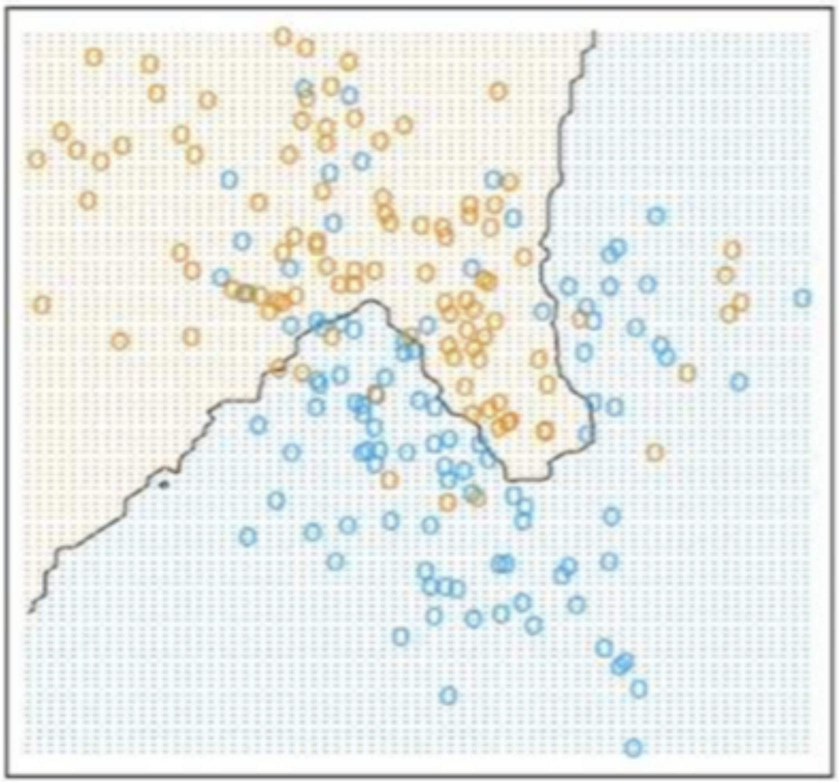
If *d(.,.)* is fixed, but you can change *k*

Selecting *k*: How?

How to find **best k** in k-NN?

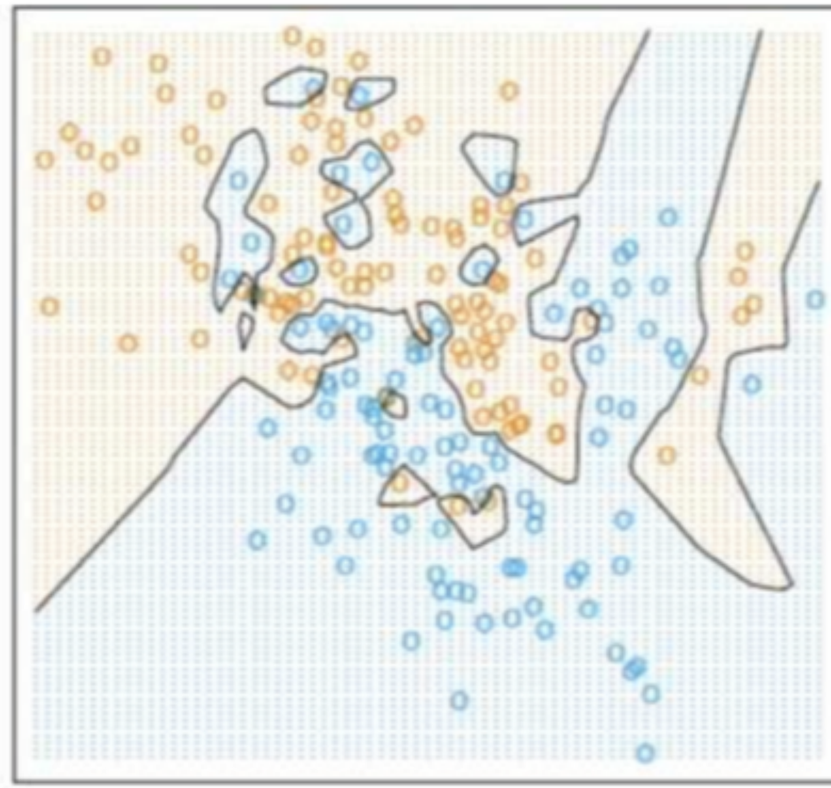
Use **cross validation (CV)**.

15-NN



Pretty good!

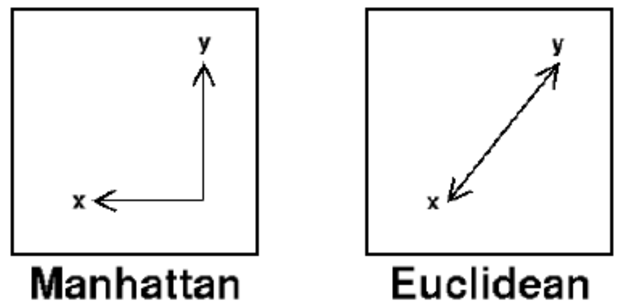
1-NN



Overfitted

k-Nearest-Neighbor Classifier

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



Possible distance functions:

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

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

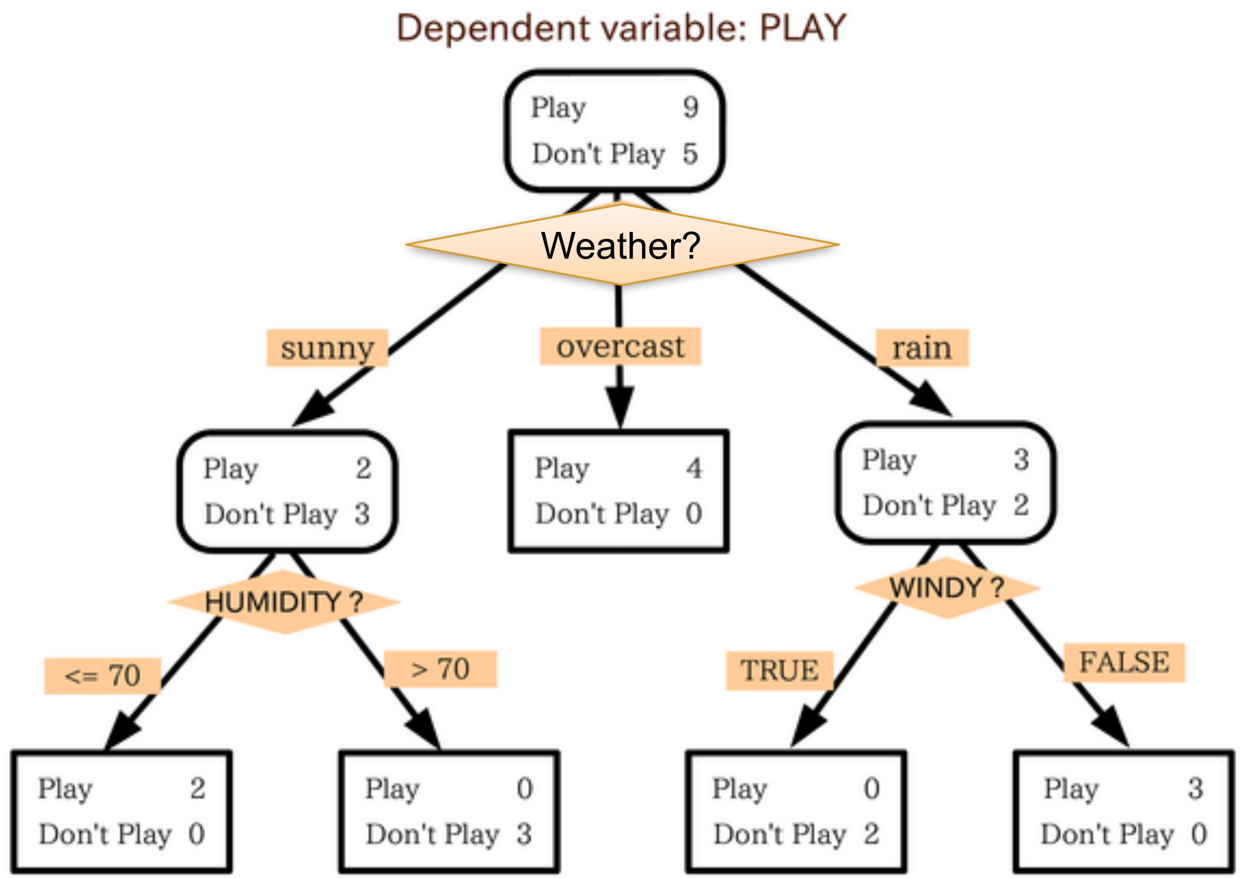
Summary on k-NN classifier

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

Reading material:

- The Elements of Statistical Learning (ESL) book, Chapter 13.3
  https://web.stanford.edu/~hastie/ElemStatLearn/

Decision trees (DT)



The classifier:

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

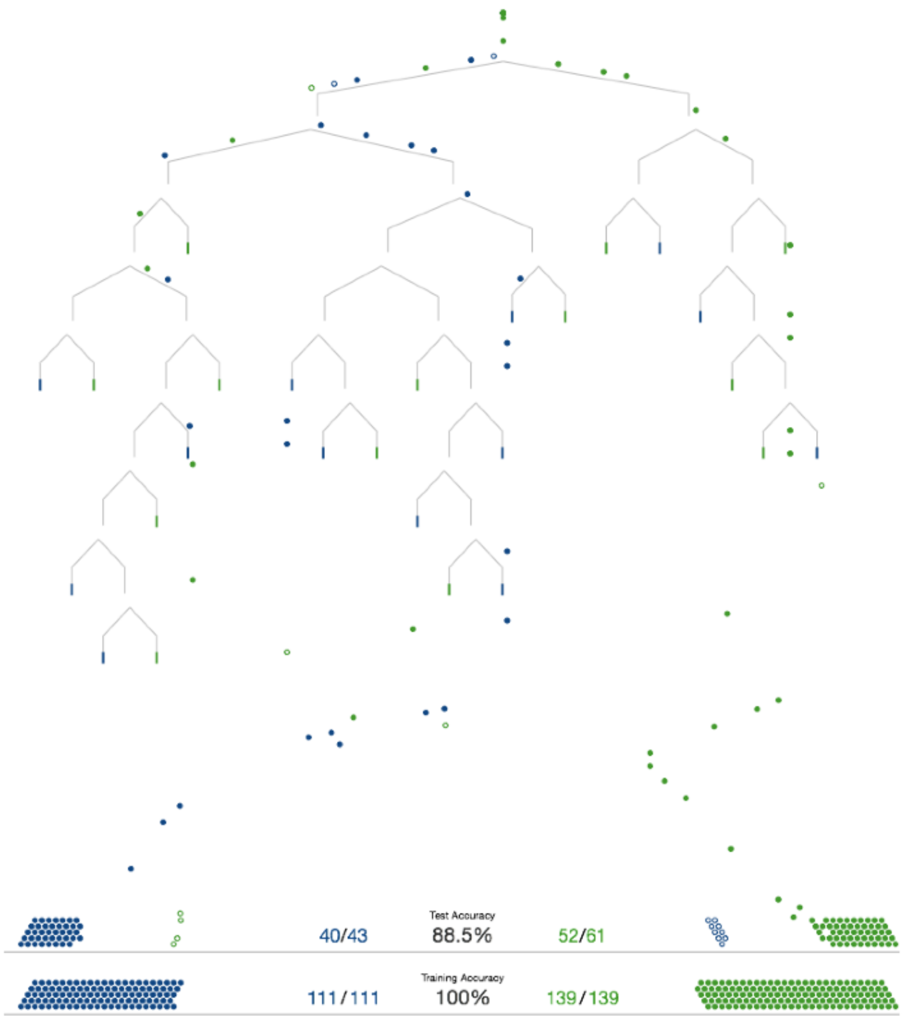
Model parameters: The tree structure and size

Highly recommended!

Visual Introduction to Decision Tree

Building a tree to distinguish homes in New York from homes in San Francisco

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

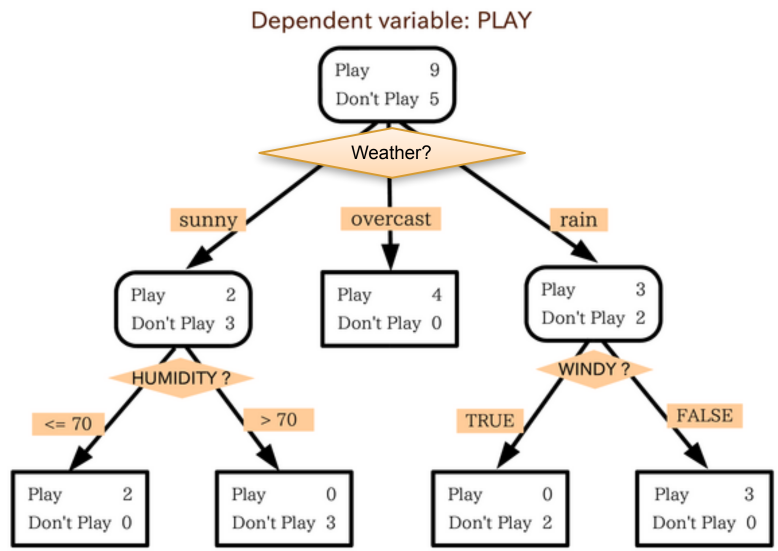


Decision trees

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

Highly recommended lecture slides from CMU

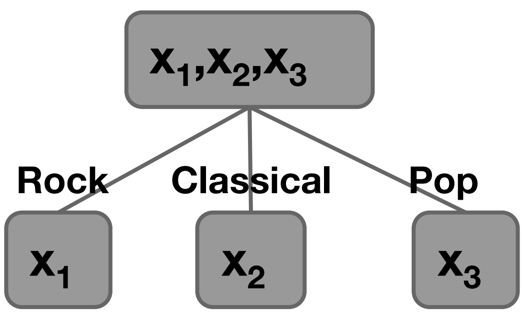
http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-s06/www/DTs.pdf

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

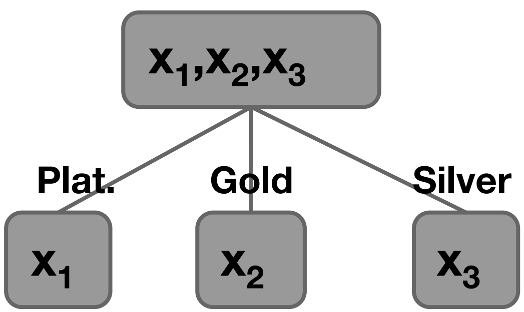
Choosing the split point

Split types for a selected attribute j:

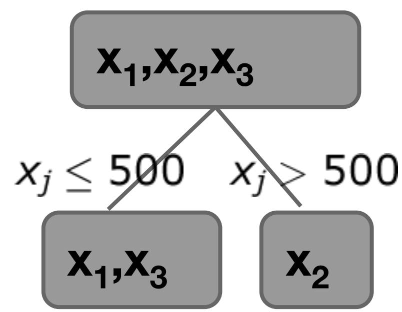
1. Categorical attribute (e.g. “genre”)
   x_{1j} = Rock, x_{2j} = Classical, x_{3j} = Pop
2. Ordinal attribute (e.g., “achievement”)
   x_{1j}=Platinum, x_{2j}=Gold, x_{3j}=Silver
3. Continuous attribute (e.g., song duration)
   $x_{1j} = 235$, $x_{2j} = 543$, $x_{3j} = 378$



Split on genre



Split on achievement



Split on duration

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

Choosing the split point

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")

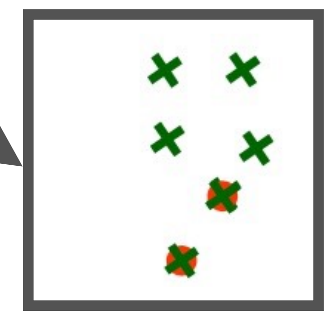
Excellent refresher on **information gain**: using it pick **splitting attribute** and **split point** (for that attribute)

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

PDF page 7 to 21

When to stop splitting? Common strategies:

1. Pure and impure leave nodes
   - All points belong to the same class; OR
   - All points from one class completely overlap with points from another class (i.e., same attributes)
     - Output majority class as this leaf's label



2. Node contains points fewer than some threshold
3. Node purity is higher than some threshold
4. Further splits provide no improvement in *training loss*
   ($loss(T) <= loss(T_L) + loss(T_R)$)

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

Parameters vs Hyper-parameters

Example **hyper-parameters** (need to experiment/try)

- k-NN: k, similarity function
- Decision tree: #node,
- Can be determined using **CV** and **optimization strategies**, e.g., “grid search” (fancy way to say “try all combinations”), random search, etc. (http://scikit-learn.org/stable/modules/grid_search.html)

Example **parameters**
(can be “learned” / “estimated” / “computed” directly from data)

- Decision tree (entropy-based):
  - which attribute to split
  - split point for an attribute

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

- “Too basic” — but OK if it works!
- Training can be very expensive
- Cross-validation is hard (node-level CV)