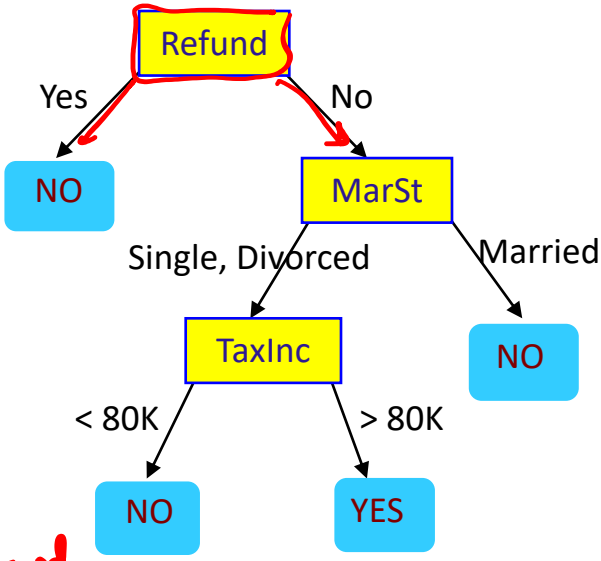
How to learn a decision tree

- Top-down induction [ID3, C4.5, C5, …]

Main loop: C4.5

1. $X \leftarrow$ the "best" decision feature for next *node*
2. Assign X as decision feature for *node*
3. For "best" split of X, create new descendants of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes
6. Prune back tree to reduce overfitting
7. Assign majority label to the leaf node



(don't discard previously used features)

Handling continuous features (C4.5)

Convert continuous features into discrete by setting a threshold.

What threshold to pick?

Search for best one as per information gain. Infinitely many??

Don't need to search over more than ~ n (number of training data),e.g. say X_1 takes values $x_1^{(1)}, x_1^{(2)}, \dots, x_1^{(n)}$ in the training set. Then possible thresholds are

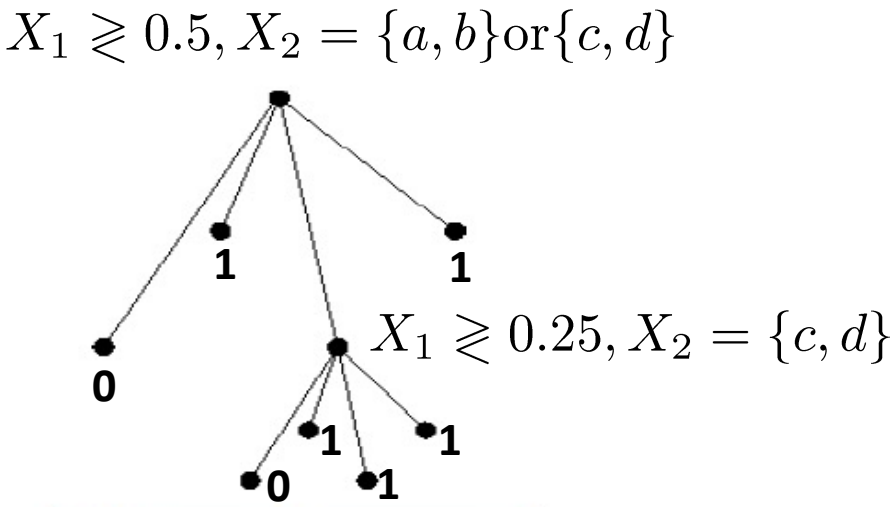
$$[x_1^{(1)} + x_1^{(2)}]/2,\ [x_1^{(2)} + x_1^{(3)}]/2,\ \dots,\ [x_1^{(n-1)} + x_1^{(n)}]/2$$

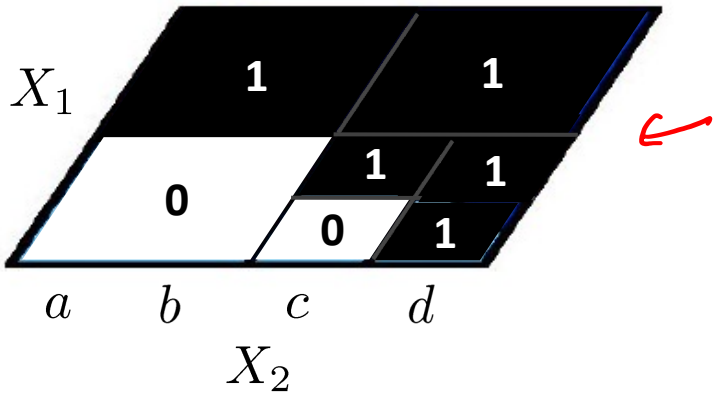
Poll

- What's the maximum depth of a decision tree learnt using ID3?

- What's the maximum depth of a decision tree learnt using C4.5?

Decision Tree more generally…



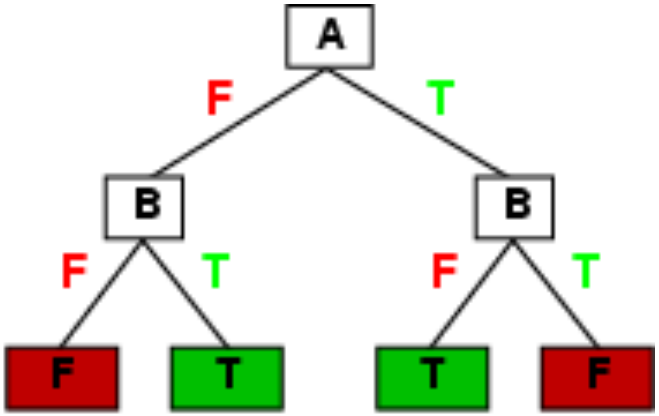


- Features can be discrete, continuous or categorical
- Each internal node: test some set of features $\{X_i\}$
- Each branch from a node: selects a set of value for $\{X_i\}$
- Each leaf node: prediction for Y

Expressiveness of Decision Trees

- Decision trees in general (without pruning) can express any function of the input features.
- E.g., for Boolean functions, truth table row → path to leaf:

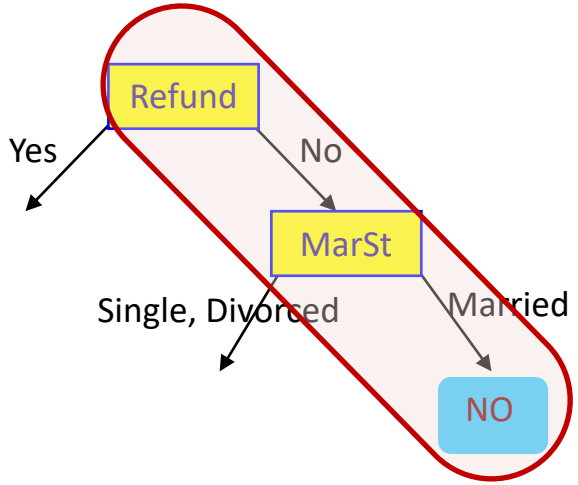
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- There is a decision tree which perfectly classifies a training set with one path to leaf for each example - overfitting
- But it won't generalize well to new examples - prefer to find more compact decision trees

Pruning the tree

- Many strategies for picking simpler trees:
  - Pre-pruning
    - Fixed depth (e.g. ID3)
    - Fixed number of leaves
  - Post-pruning
    - Chi-square test – independence test
      - Convert decision tree to a set of rules
      - Eliminate variable values in rules which are independent of label (using chi-square test for independence)
      - Simplify rule set by eliminating unnecessary rules
  - Information Criteria: MDL(Minimum Description Length)



Information Criteria

- Penalize complex models by introducing cost

$$\hat{f} = \arg\min_T \left\{ \frac{1}{n} \sum_{i=1}^{n} \text{loss}(\hat{f}_T(X_i), Y_i) + \text{pen}(T) \right\}$$

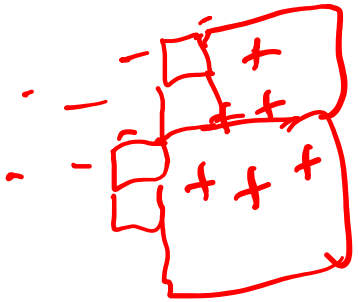
log likelihood cost

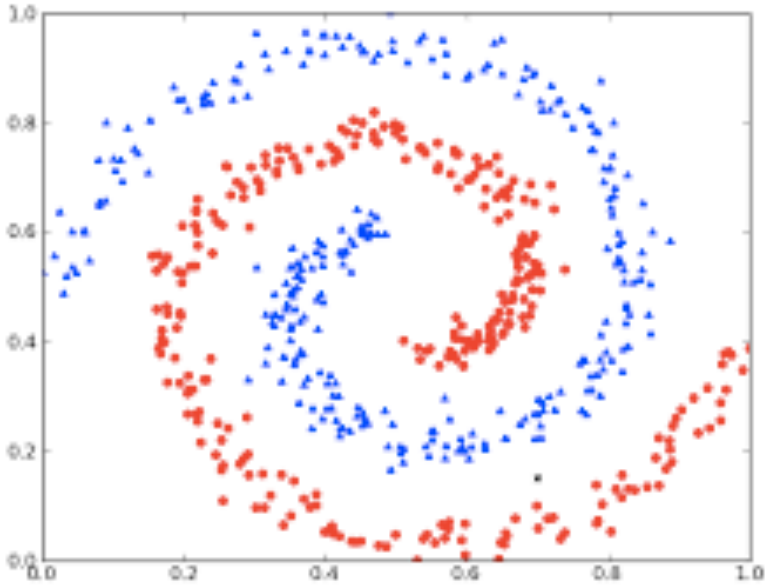
$$\text{loss}(\hat{f}_T(X_i), Y_i) = (\hat{f}_T(X_i) - Y_i)^2 \quad \text{regression}$$
$$= 1_{\hat{f}_T(X_i) \neq Y_i} \quad \text{classification}$$

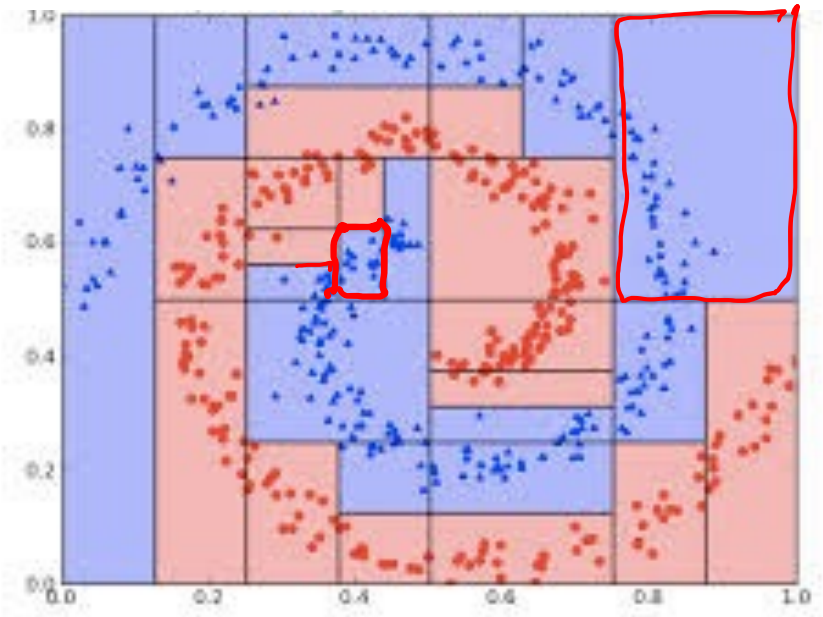
$\text{pen}(T) \propto |T|$ penalize trees with more leaves

CART – optimization can be solved by dynamic programming

Example of 2-feature decision tree classifier







larger bins ⇒ low variance, high bias

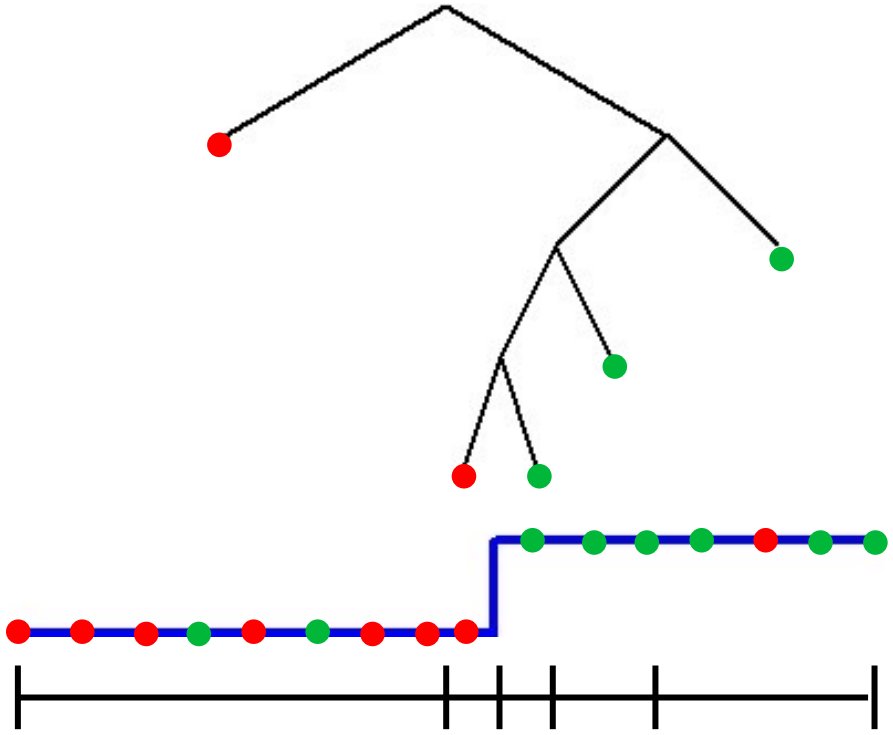
smaller bins ⇒ high variance, low bias

cs.uchicago.edu

How to assign label to each leaf

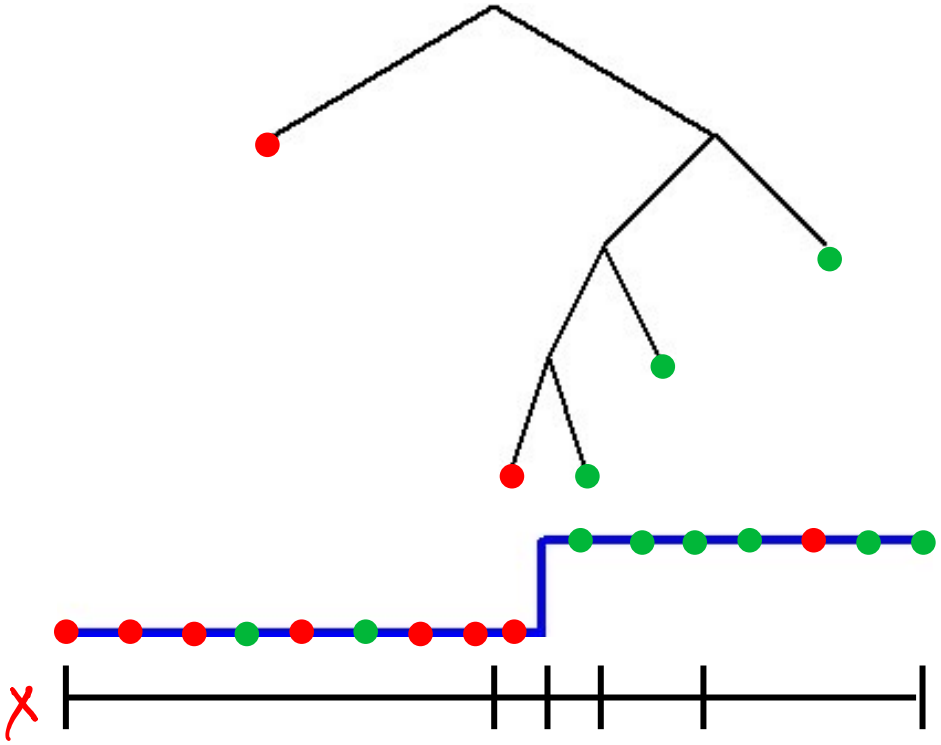
Classification – Majority vote

Regression – ?

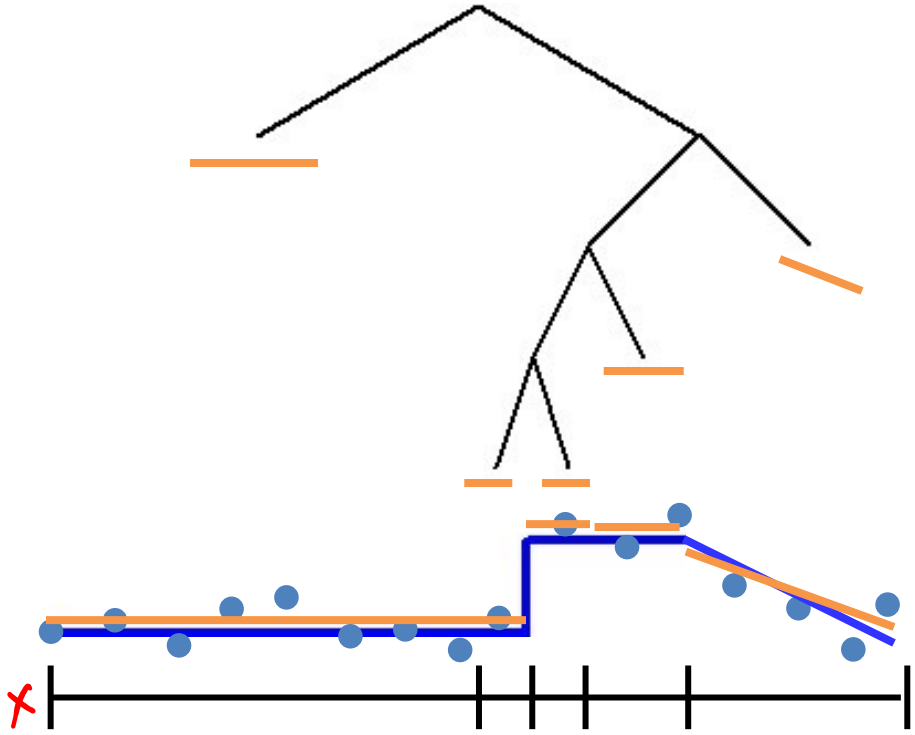


How to assign label to each leaf

Classification – Majority vote



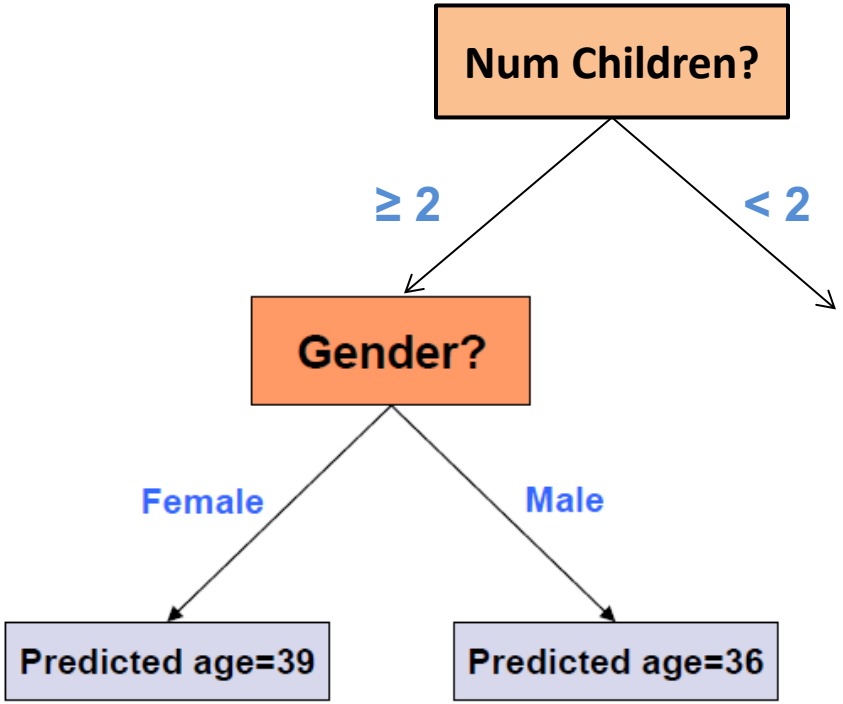
Regression – Constant/ Linear/Poly fit



Regression trees

$X^{(1)}$ .... $X^{(p)}$ Y

Gender	Rich?	Num. Children	# travel per yr.	Age
F	No	2	5	38
M	No	0	2	25
M	Yes	1	0	72
:	:	:	:	:



Average (fit a constant) using training data at the leaves

What you should know

- Decision trees are one of the most popular data mining tools
  - Simplicity of design
  - Interpretability
  - Ease of implementation
  - Good performance in practice (for small dimensions)
- Information gain to select attributes (ID3, C4.5,…)
- Decision trees will overfit!!!
  - Must use tricks to find "simple trees", e.g.,
    - Pre-Pruning: Fixed depth/Fixed number of leaves
    - Post-Pruning: Chi-square test of independence
    - Complexity Penalized/MDL model selection
- Can be used for classification, regression and density estimation too

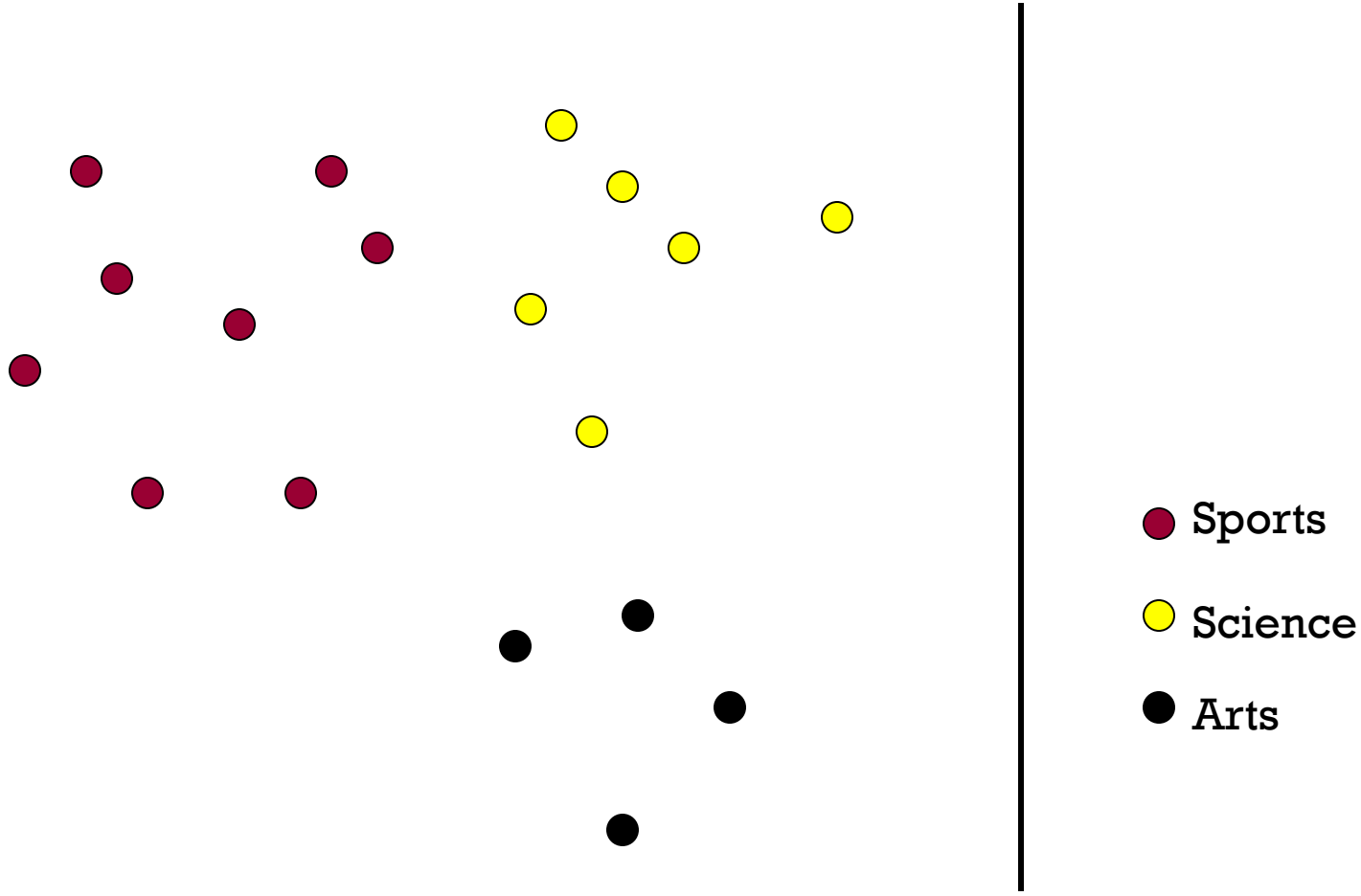
features ≡ attributes

k-NN classifier
Nonparametric kernel regression

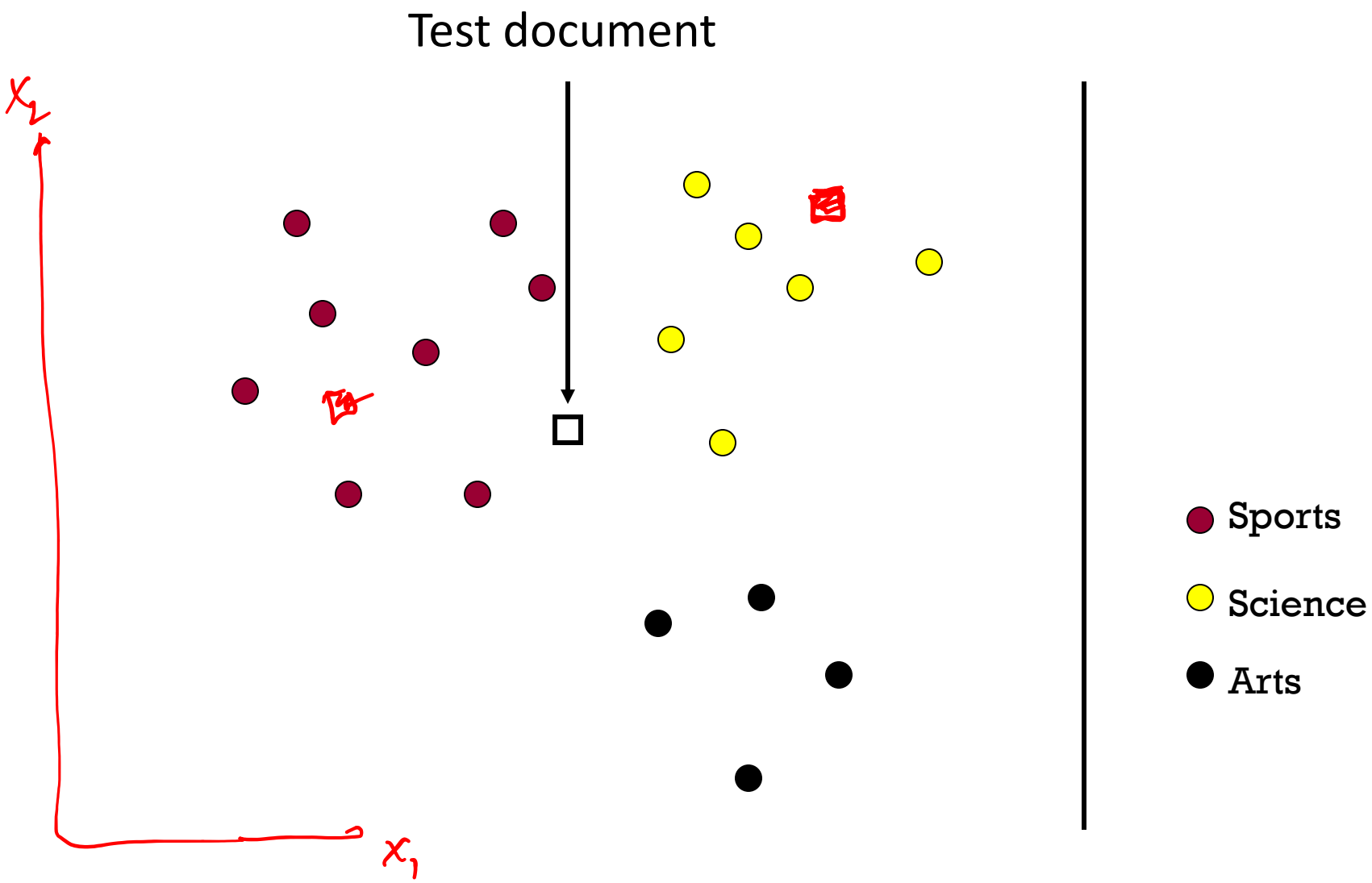
Aarti Singh

Feb 27, 2023
Machine Learning 10-701

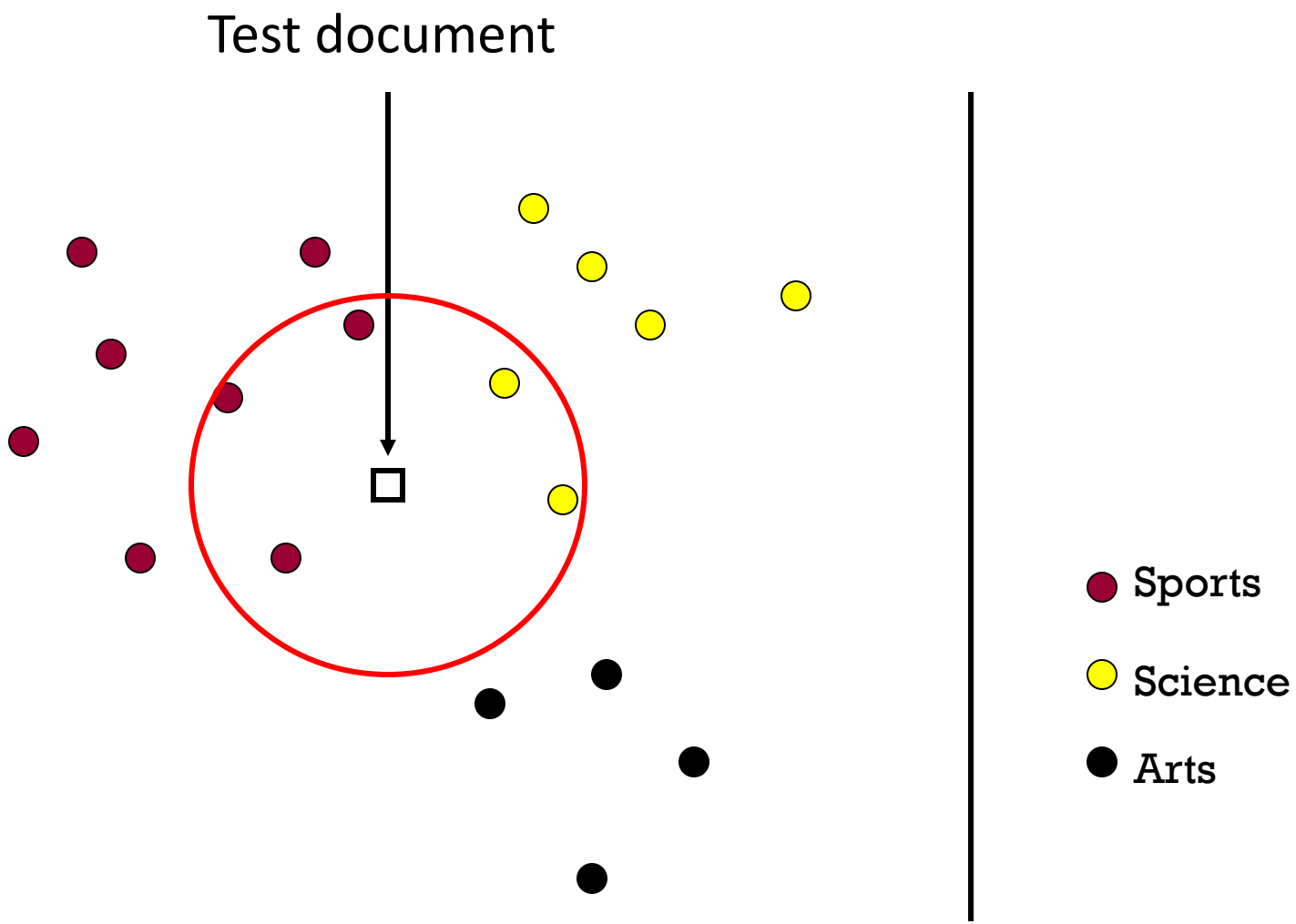
k-NN classifier



k-NN classifier



k-NN classifier (k=5)



What should we predict? … Average? Majority? Why?

k-NN classifier

- Optimal Classifier: $f^*(x) = \arg\max_y P(y|x)$
  $= \arg\max_y P(x|y)P(y)$

- k-NN Classifier: $\hat{f}_{kNN}(x) = \arg\max_y \hat{P}_{kNN}(x|y)\hat{P}(y)$
  $= \arg\max_y k_y$

$\hat{P}_{kNN}(x|y) = \frac{k_y}{n_y}$ → # training pts of class y amongst k NNs of x

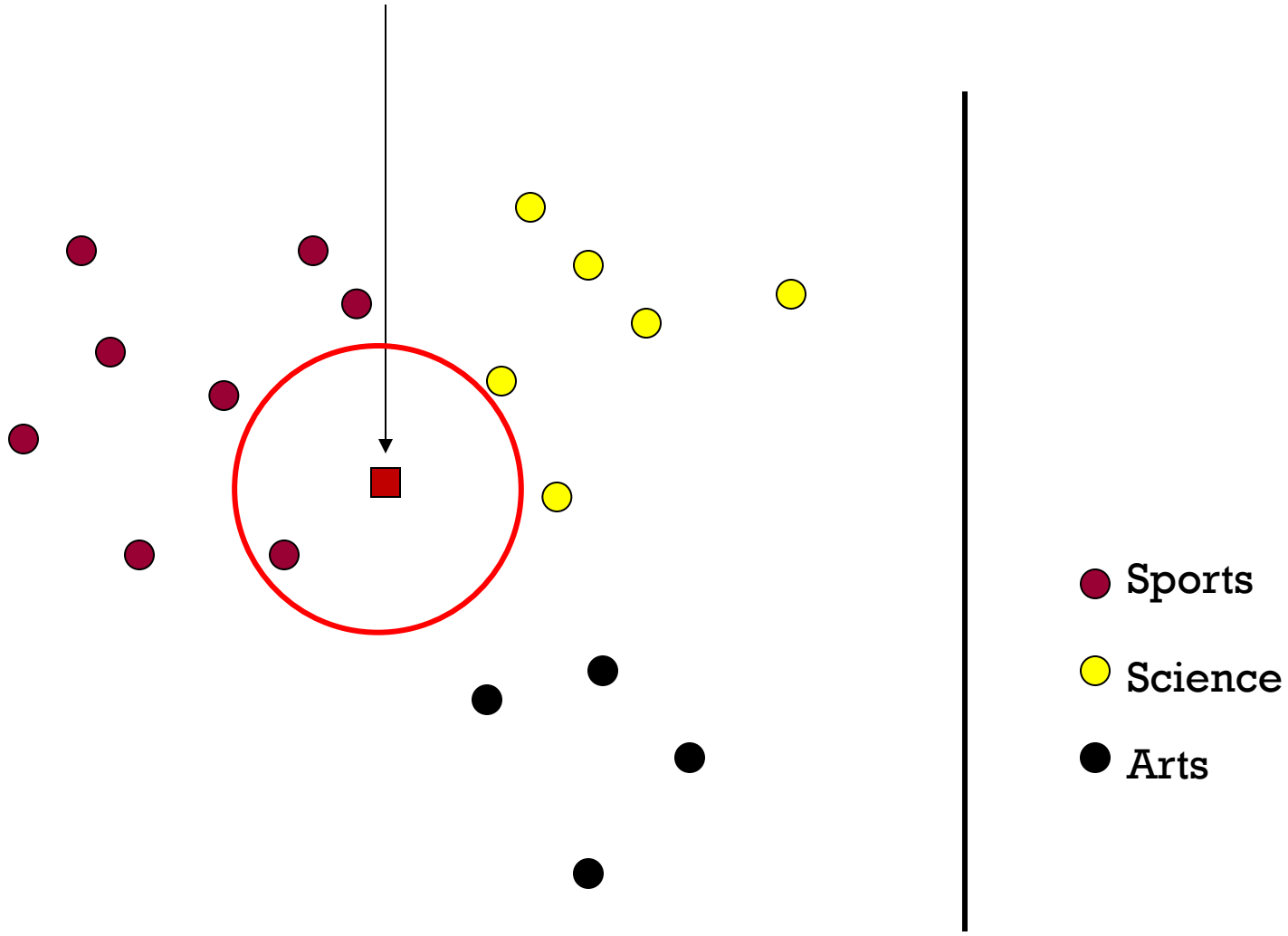
→ # total training pts of class y

$\sum_y k_y = k$

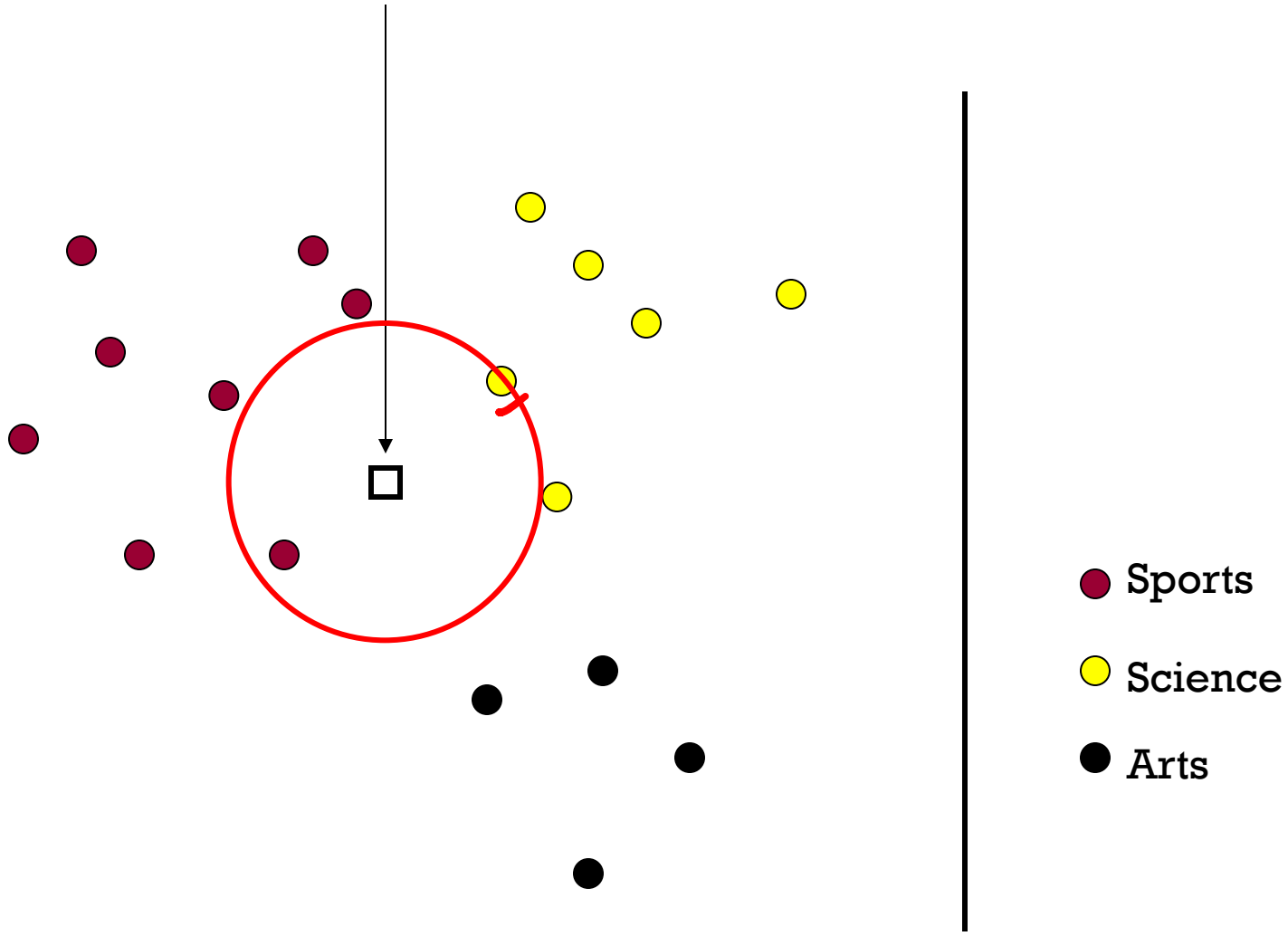
$\hat{P}(y) = \frac{n_y}{n}$ ← no. of training examples that have label y

← total no. of training examples

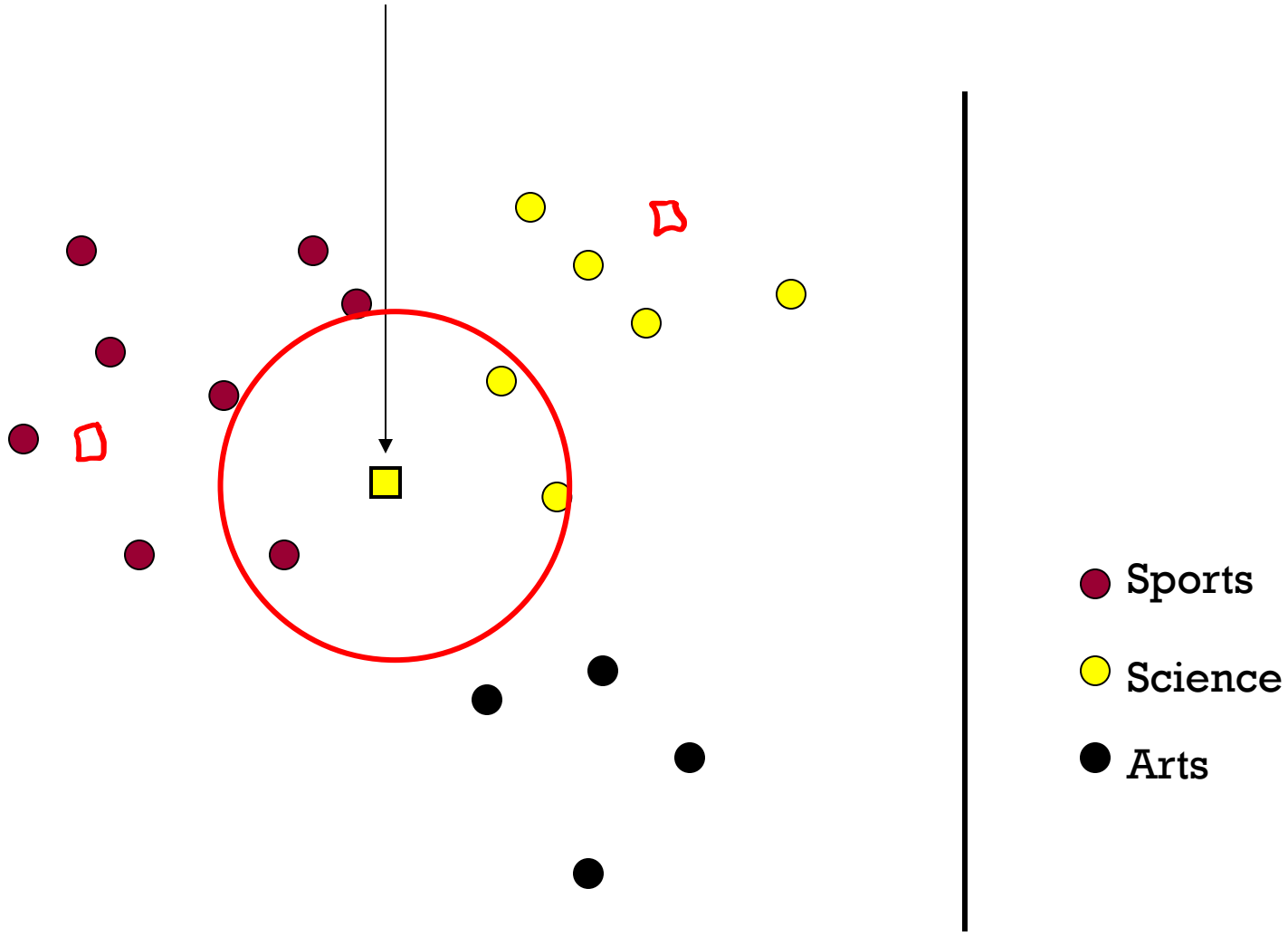
1-Nearest Neighbor (kNN) classifier



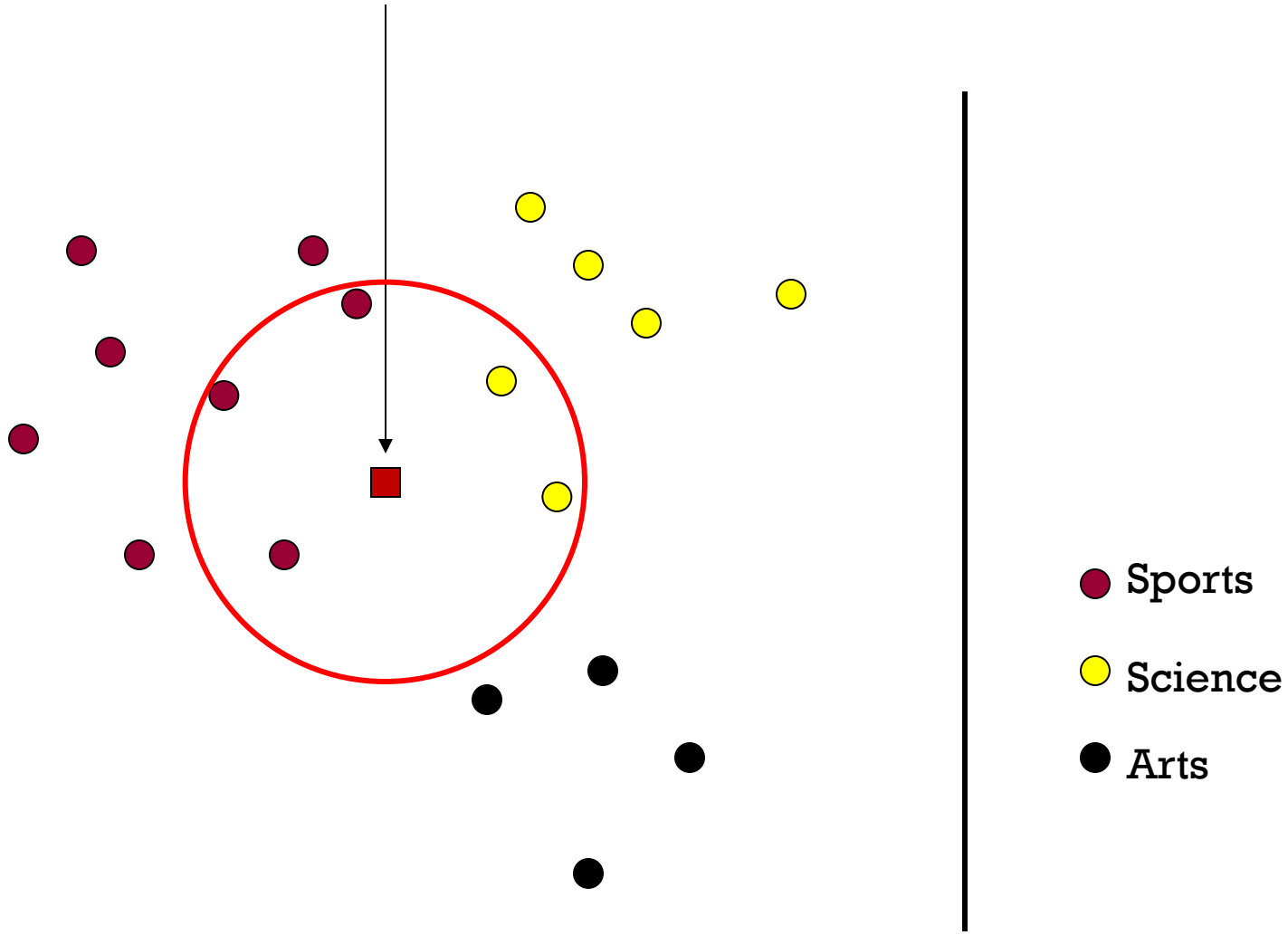
2-Nearest Neighbor (kNN) classifier



3-Nearest Neighbor (kNN) classifier

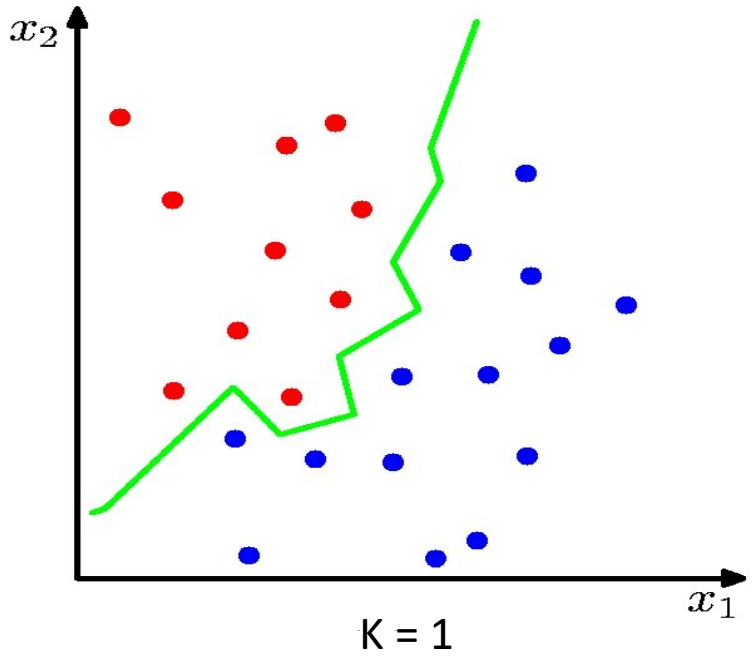


5-Nearest Neighbor (kNN) classifier



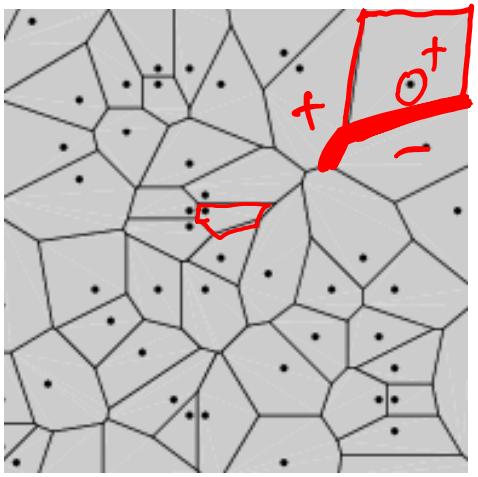
What is the best k?

1-NN classifier decision boundary



K = 1

Voronoi Diagram



larger k → smoother decision boundary

As k increases, boundary becomes smoother (less jagged).

What is the best k?

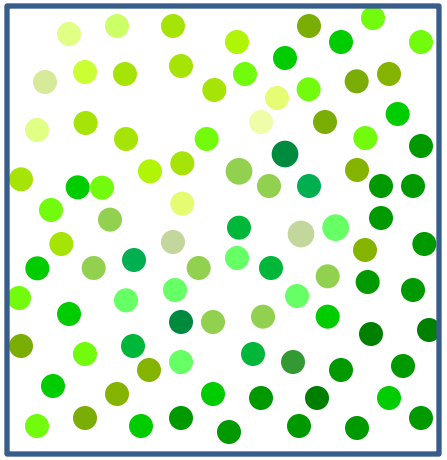
Approximation vs. Stability (aka Bias vs Variance) Tradeoff

- Larger K => predicted label is more stable (low variance) but potentially less accurate (high bias)

- Smaller K => predicted label can approximate best classifier well given enough data (low bias) but predict label is unstable (high variance)

Local Kernel Regression

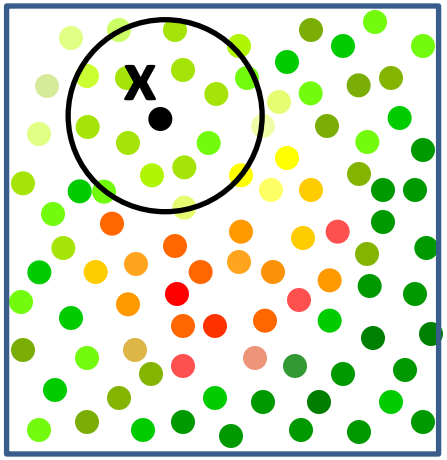
- What is the temperature in the room?



$$\widehat{T} = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

Average

at location x?

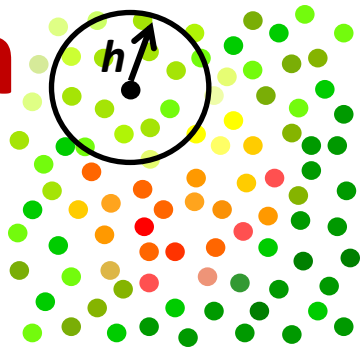


$$\widehat{T}(x) = \frac{\sum_{i=1}^{n} Y_i 1_{||X_i - x|| \le h}}{\sum_{i=1}^{n} 1_{||X_i - x|| \le h}}$$

$n_{x,h}$

"Local" Average

Local Kernel Regression



Global average $w_i = \frac{1}{n} \Rightarrow \sum_i w_i = 1$

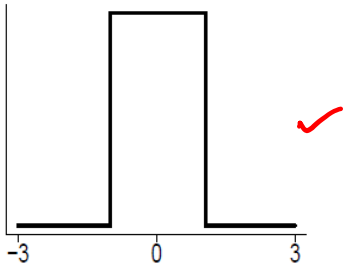
- Nonparametric estimator
- Nadaraya-Watson Kernel Estimator

$$\hat{f}_n(X) = \sum_{i=1}^{n} w_i Y_i \quad \text{Where} \quad w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

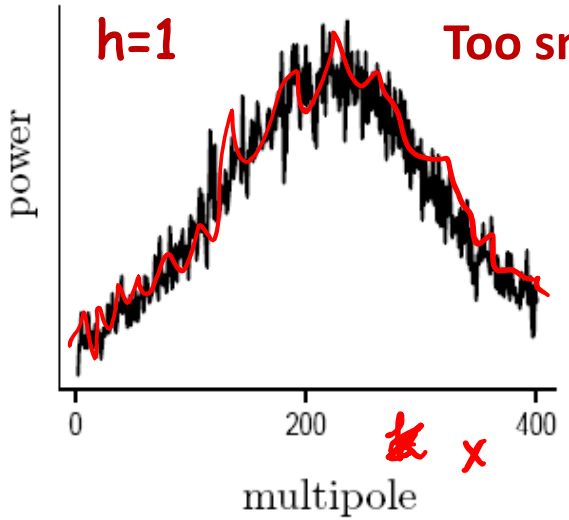
$\Rightarrow \sum_i w_i = 1$

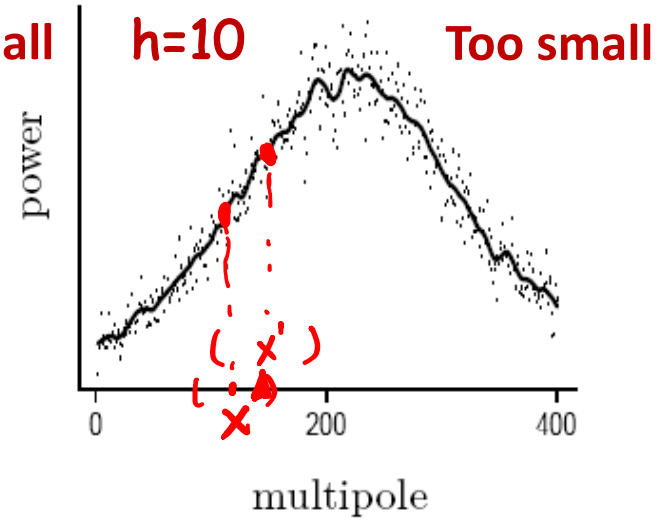
- Weight each training point based on distance to test point
- Boxcar kernel yields local average

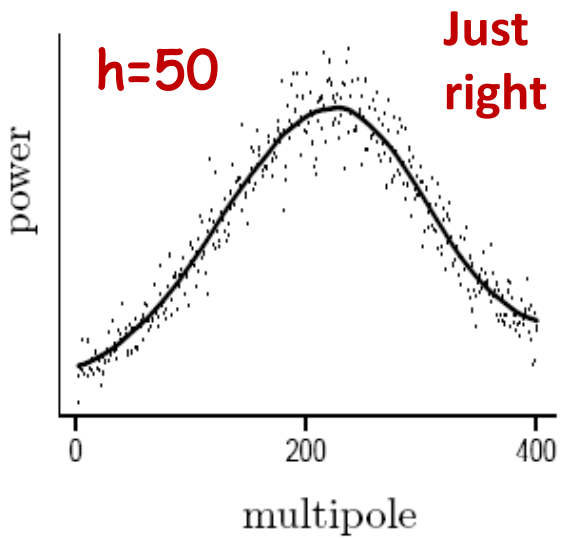
$$\text{boxcar kernel}: \quad K(x) = \frac{1}{2} I(x),$$



Choice of kernel bandwidth h







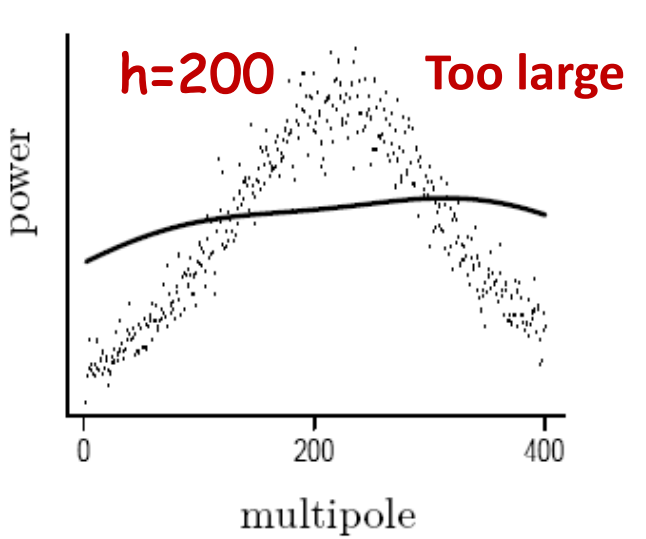


Image Source: Larry's book – All of Nonparametric Statistics

Kernel Regression as Weighted Least Squares

$$\min_f \sum_{i=1}^{n} w_i (f(X_i) - Y_i)^2$$

Weighted Least Squares

$$w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

Kernel regression corresponds to locally constant estimator obtained from (locally) weighted least squares

i.e. set $f(X_i) = \beta$ (a constant)

Kernel Regression as Weighted Least Squares

set $f(X_i) = \beta_x$ (a constant)

$$\min_{\beta} \sum_{i=1}^{n} w_i(\beta_x - Y_i)^2 \qquad w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

β_x: constant

$J(\beta)$

$$\frac{\partial J(\beta_x)}{\partial \beta_x} = 2\sum_{i=1}^{n} w_i(\beta_x - Y_i) = 0$$

Notice that $\sum_{i=1}^{n} w_i = 1$

$$\Rightarrow \hat{f}_n(X) = \hat{\beta} = \sum_{i=1}^{n} w_i Y_i$$

$\sum_{i=1}^{n} w_i \beta_n = \sum_{i=1}^{n} w_i Y_i$

$\Rightarrow \beta_x = \frac{\sum_i w_i Y_i}{\sum w_i}$, $\sum w_i = 1$

Local Linear/Polynomial Regression

$$\min_f \sum_{i=1}^{n} w_i(f(X_i) - Y_i)^2 \qquad w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

Weighted Least Squares

Local Polynomial regression corresponds to locally polynomial estimator obtained from (locally) weighted least squares

i.e. set $f(X_i) = \beta_0 + \beta_1(X_i - X) + \frac{\beta_2}{2!}(X_i - X)^2 + \cdots + \frac{\beta_p}{p!}(X_i - X)^p$

(local polynomial of degree p around X)

$f(x_i) = \beta$ $\qquad$ $f(x_i) = \beta_0 + \beta_1(x_i - x)$

Summary

- Non-parametric approaches

Four things make a nonparametric/memory/instance based/lazy learner:

1. *A distance metric, dist(x,X_i)*
   Euclidean (and many more)
2. *How many nearby neighbors/radius to look at?*
   k, Δ/h
3. *A weighting function (optional)*
   W based on kernel K
4. *How to fit with the local points?*
   Average, Majority vote, Weighted average, Poly fit

Summary

- Parametric vs Nonparametric approaches
  - Nonparametric models place very mild assumptions on the data distribution and provide good models for complex data

    Parametric models rely on very strong (simplistic) modeling assumptions

  - Nonparametric models typically require storage and computation of the order of entire data set size.

    Parametric models, once fitted, are much more efficient in terms of storage and computation.