kNN and Regression

Naren Ramakrishnan

Recap

- We now know two ML algorithms!
  - Decision Trees and Naïve Bayes
    - Pretty powerful approaches in their own right
    - Form the basis of more powerful methods
      - e.g., Decision Trees -> Random Forests
      - e.g., Naïve Bayes -> Bayesian Networks
- Next
  - We will focus on methods especially suited when features are real-valued
    - Target class is nominal: classification
    - Target class is real-valued: regression

What’s common between DT and NB

Trg set

f1	f2	f3	Class
			c1
			c1
			c2
			c3
			c1

Model

Test set

f1	f2	f3	Class
			?
			?
			?
			?
			?

+ Model

f1	f2	f3	Class
			c3
			c4
			c1
			c2
			c2

Eager vs Lazy algorithms

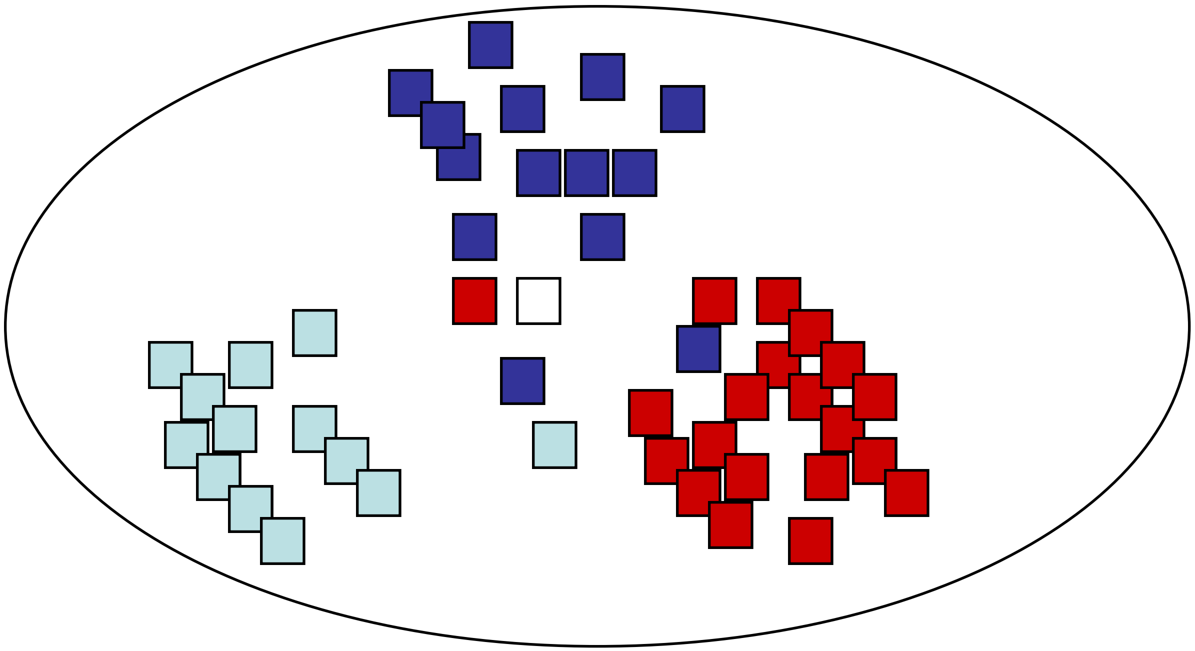
- A model-based algorithm is
  - "eager" by definition
  - Most of the hard work is done during the learning phase
- A lazy learner doesn't do much learning
  - Most of the hard work is done during the evaluation/application phase

Classical lazy algorithm

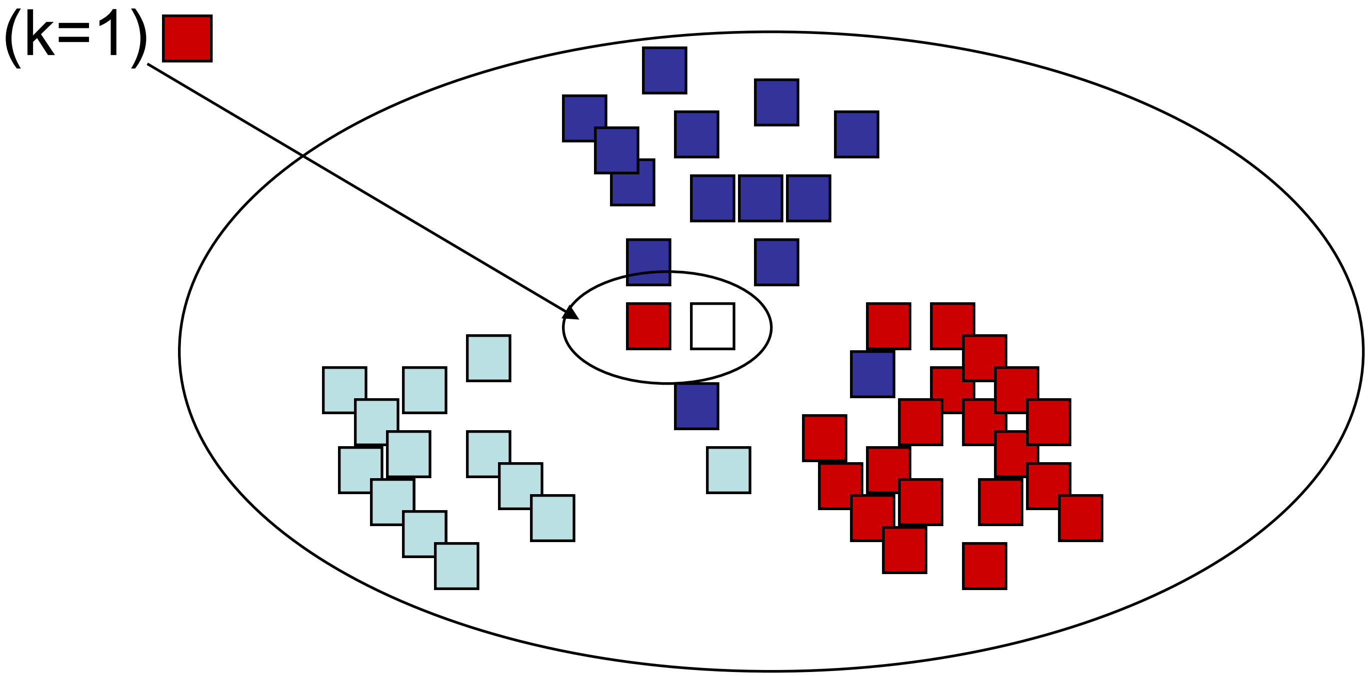
- Nearest neighbor
  - Also called “instance based learning”
  - Also referred to as a non-parametric method

Example input

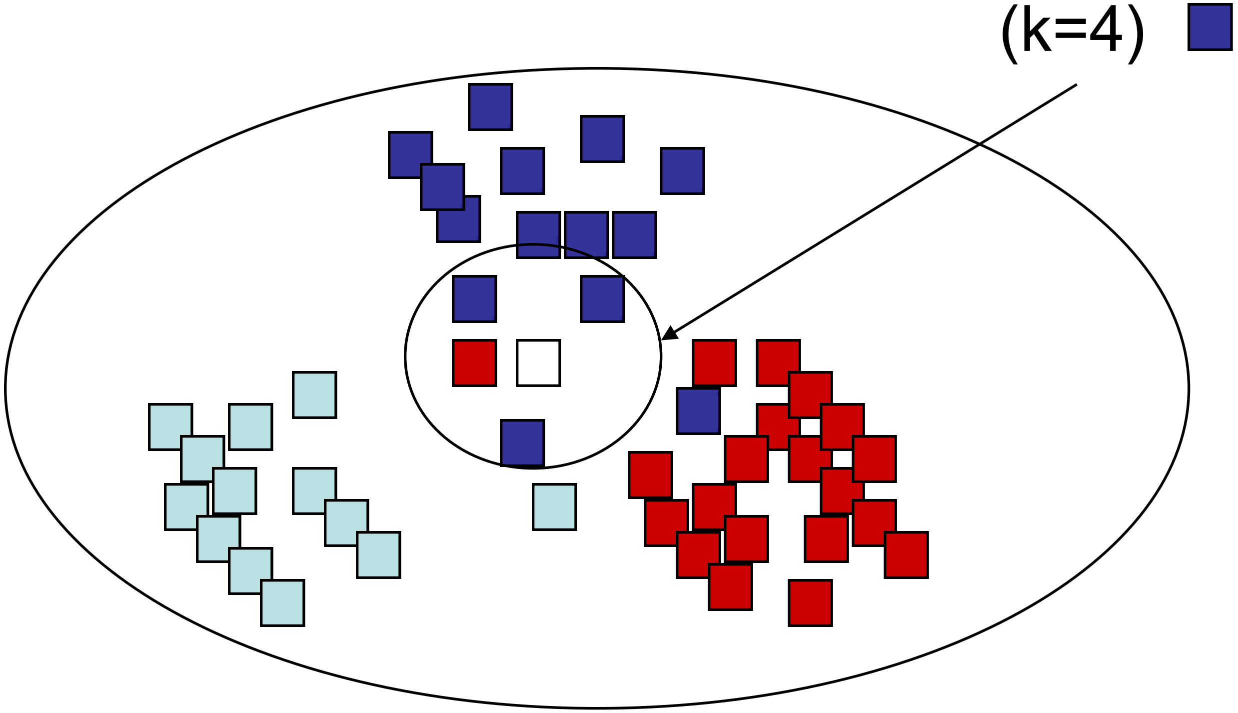
- Consider a real-valued set of attributes
  - Two dimensional



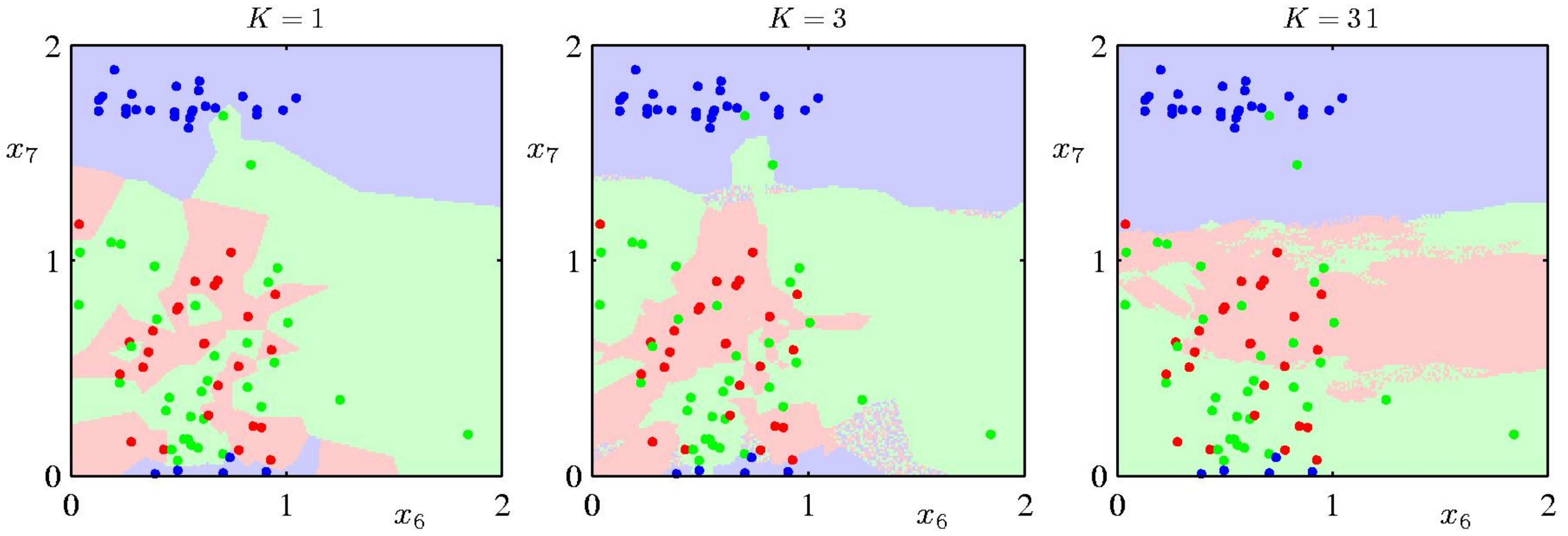
Basic Idea (k=1)



Basic Idea (k=4)



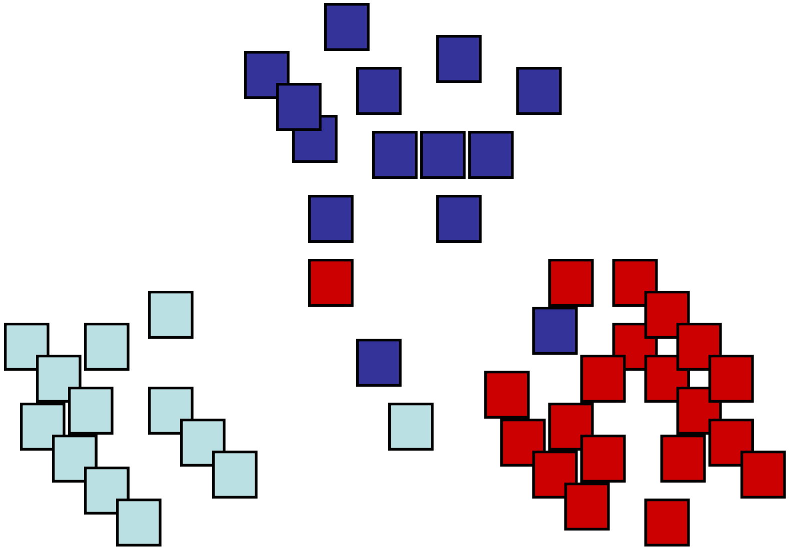
kNN as a smoother



How do we choose k?

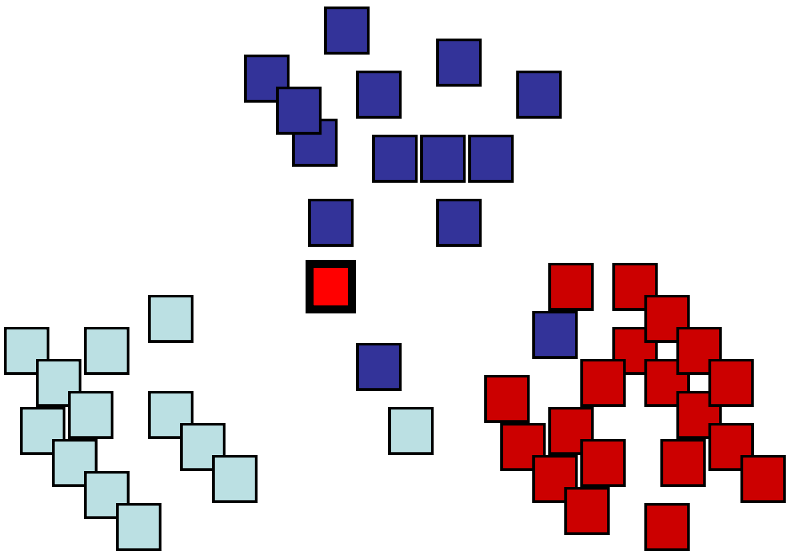
- Divide training examples into two sets
  - A training set (80%) and a validation set (say 10%) – test set is separate (another 10%)
- Predict the class labels for validation set by using the examples in training set
- Choose the number of neighbors *k* that maximizes the classification accuracy
  - => Cross-validation

Leave-One-Out Method



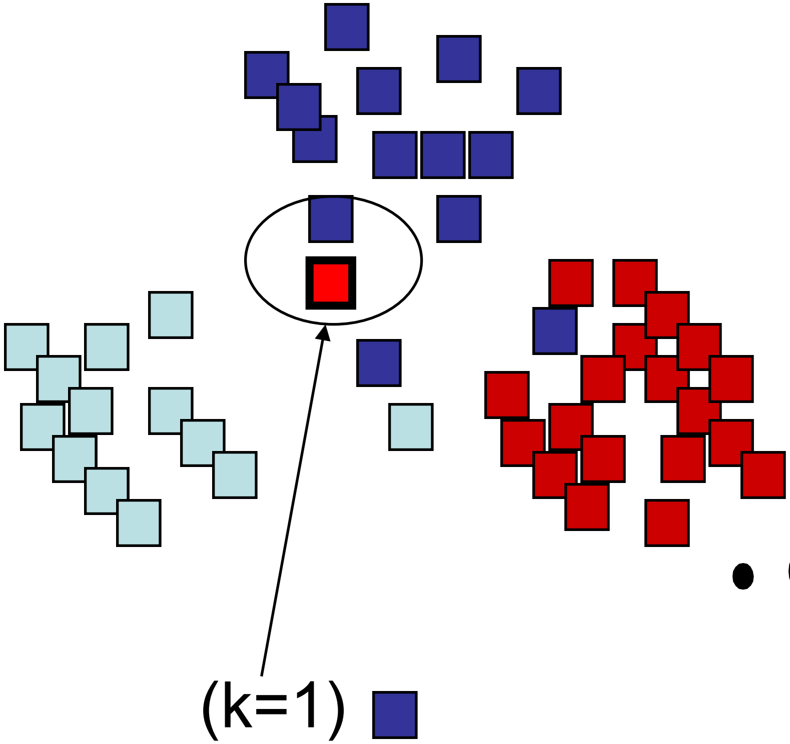
- For $k = 1, 2, \ldots, K$
  - $err(k) = 0$
  - For $i = 1, 2, \ldots, n$
    - Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$ using the remaining data points
    - $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
- Output $k^* = \arg\min_{1 \leq k \leq K} err(k)$

Leave-One-Out Method



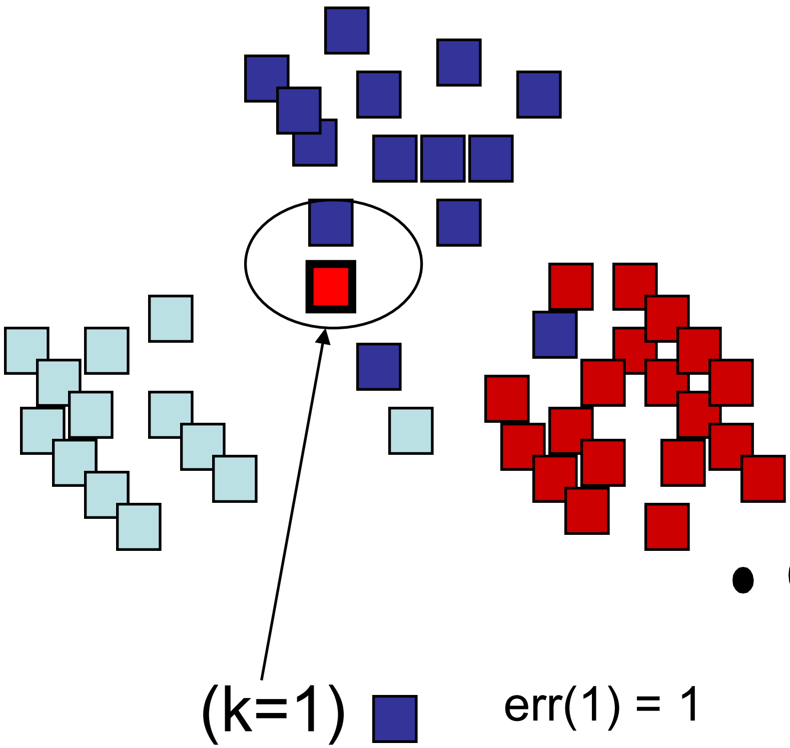
- For $k = 1, 2, \ldots, K$
  - $err(k) = 0$
  - For $i = 1, 2, \ldots, n$
    - $*$ Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$ using the remaining data points
    - $*$ $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
- Output $k^* = \arg\min_{1 \leq k \leq K} err(k)$

Leave-One-Out Method



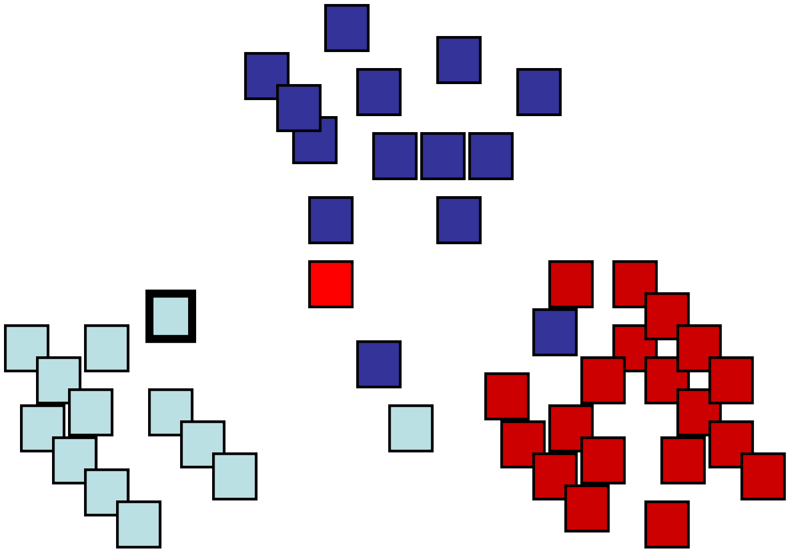
- For $k = 1, 2, \ldots, K$
  - $err(k) = 0$
  - For $i = 1, 2, \ldots, n$
    - Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$ using the remaining data points
    - $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
- Output $k^* = \arg\min_{1 \le k \le K} err(k)$

Leave-One-Out Method



- For $k = 1, 2, \ldots, K$
  - $err(k) = 0$
  - For $i = 1, 2, \ldots, n$
    - Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$ using the remaining data points
    - $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
- Output $k^* = \arg\min_{1 \leq k \leq K} err(k)$

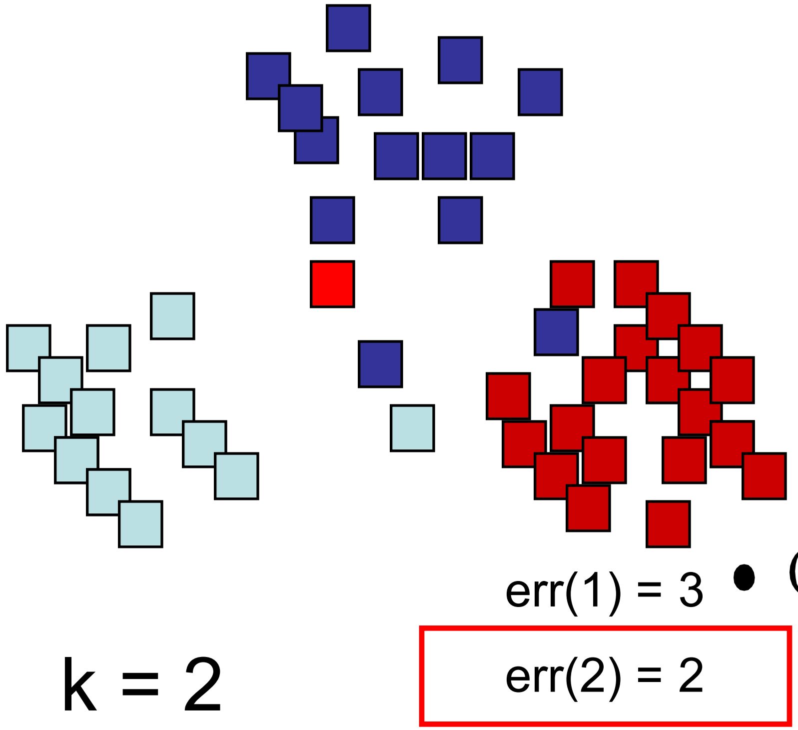
Leave-One-Out Method



err(1) = 1

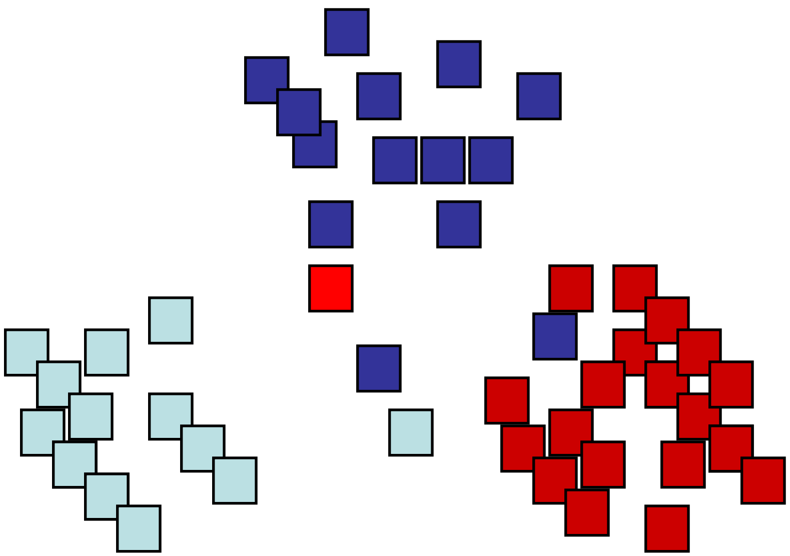
- For $k = 1, 2, \ldots, K$
  - $err(k) = 0$
  - For $i = 1, 2, \ldots, n$
    - $*$ Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$ using the remaining data points
    - $*$ $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
- Output $k^* = \underset{1 \leq k \leq K}{\arg\min}\, err(k)$

Leave-One-Out Method



- For $k = 1, 2, \ldots, K$
  - $err(k) = 0$
  - For $i = 1, 2, \ldots, n$
    - Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$ using the remaining data points
    - $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
- Output $k^* = \arg\min_{1 \leq k \leq K} err(k)$

Leave-One-Out Method



- For $k = 1, 2, \ldots, K$
  - $err(k) = 0$
  - For $i = 1, 2, \ldots, n$
    - Predict the class label $\widehat{y}_i$ for $\mathbf{x}_i$ using the remaining data points
    - $err(k) = err(k) + 1$ if $\widehat{y}_i \neq y_i$
- Output $k^* = \arg\min_{1 \leq k \leq K} err(k)$

err(1) = 3

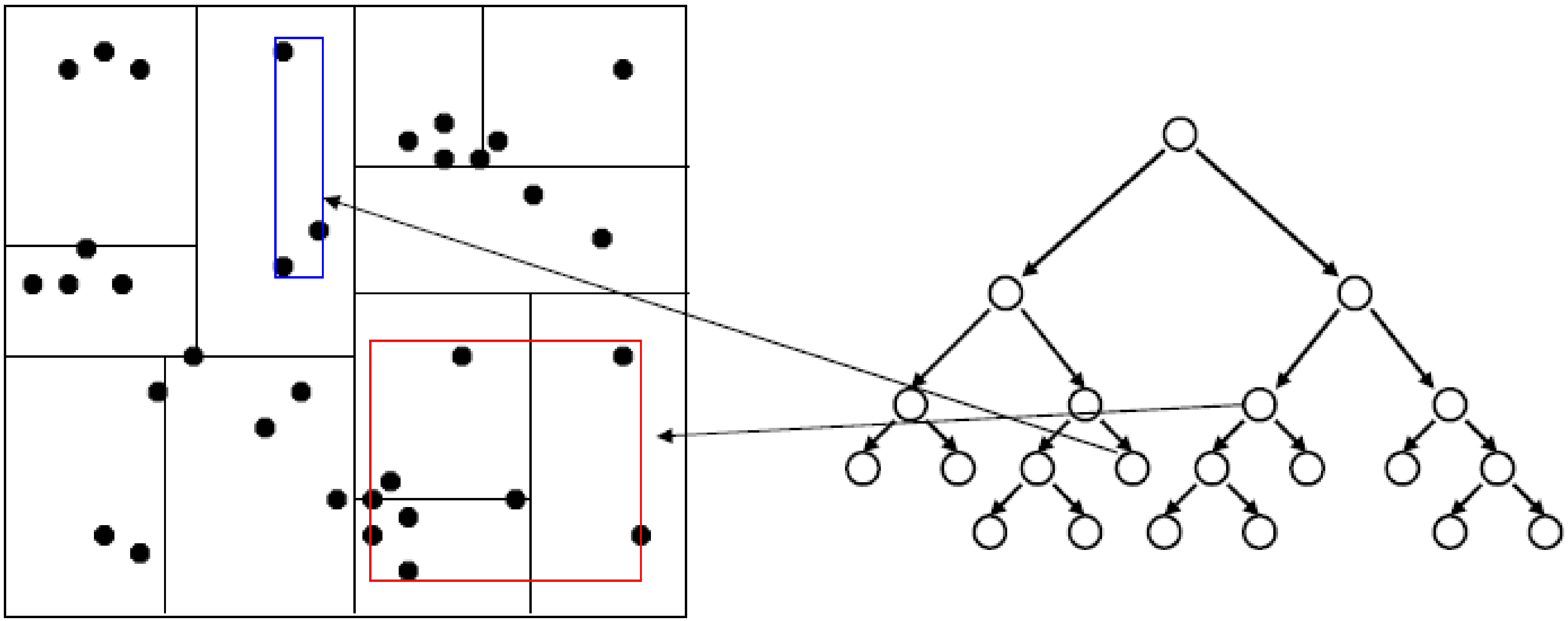
err(2) = 2

err(3) = 6

When should we use kNN?

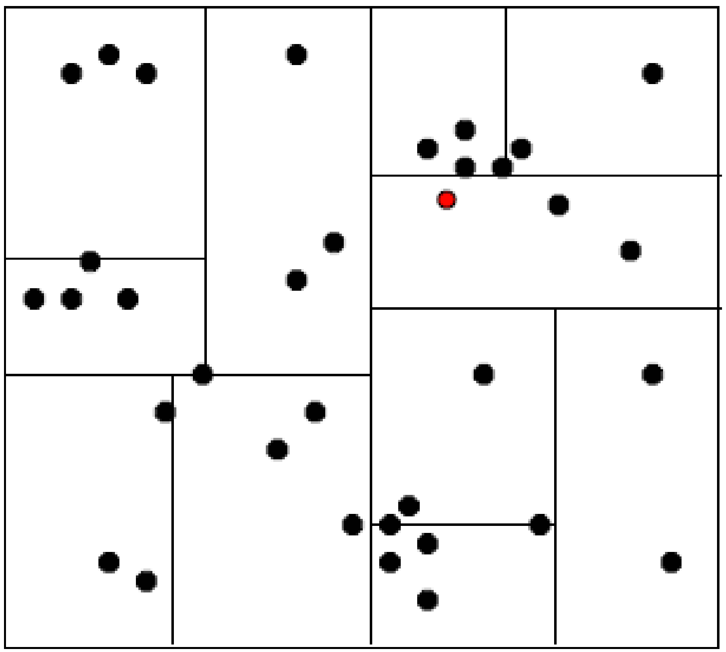
- Advantages
  - Can learn very complex target functions with arbitrary boundaries
  - Fast training time (duh!)
- Disadvantages
  - Can get easily bogged down by noise (e.g., irrelevant attributes)
  - Slow at classification time (duh!)

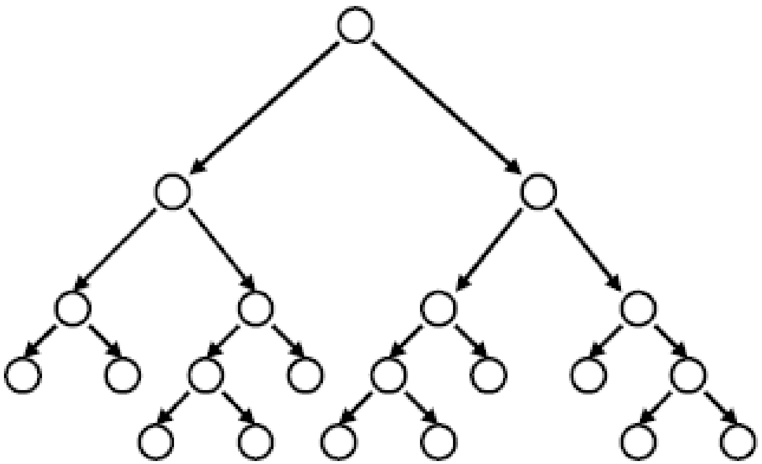
Speeding up kNN (kd-tree)



- Each node contains
  - Children information
  - The tightest box that bounds all the data points within the node.

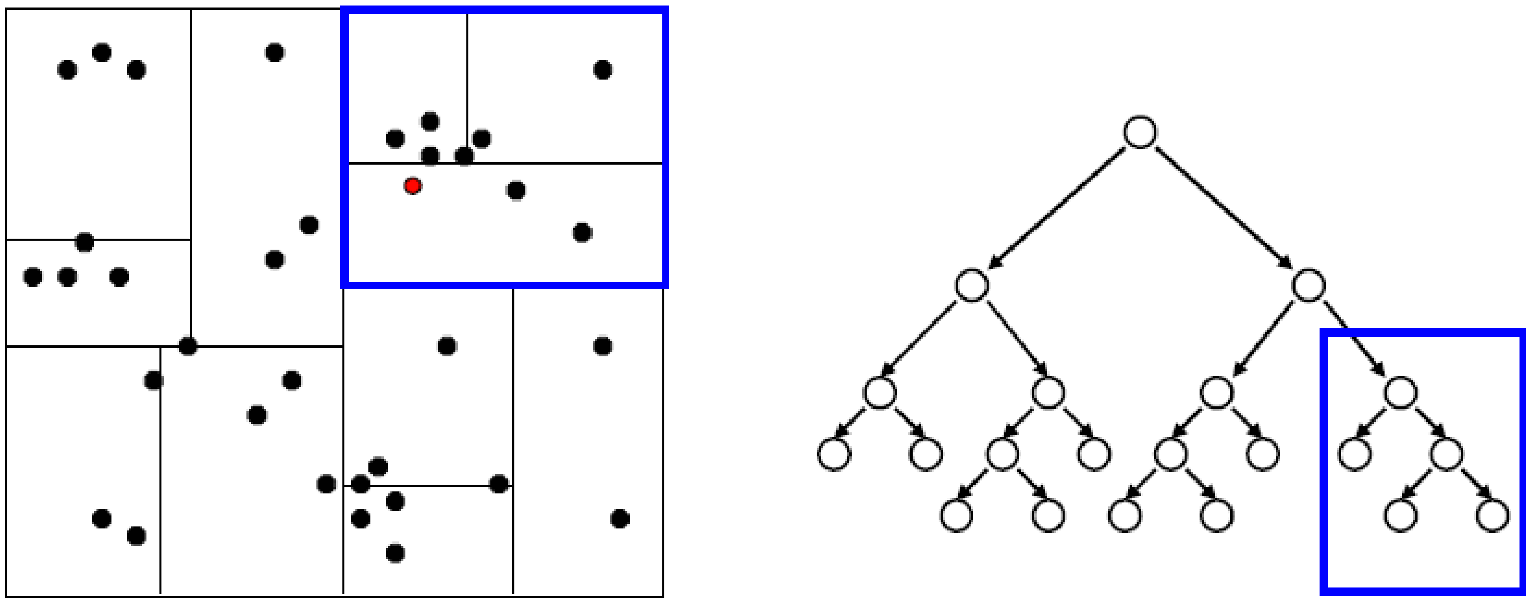
NN Search by kd-tree





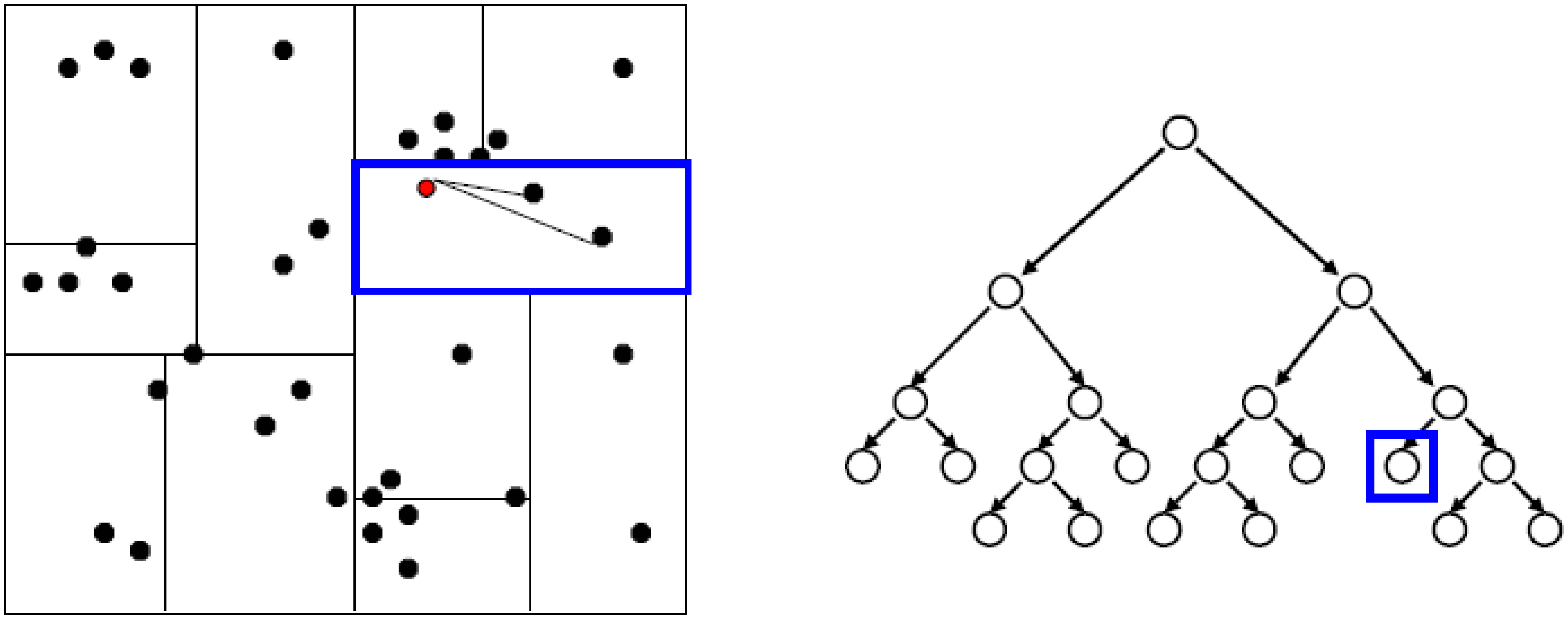
We traverse the tree looking for the nearest neighbor of the query point.

NN Search by kd-tree



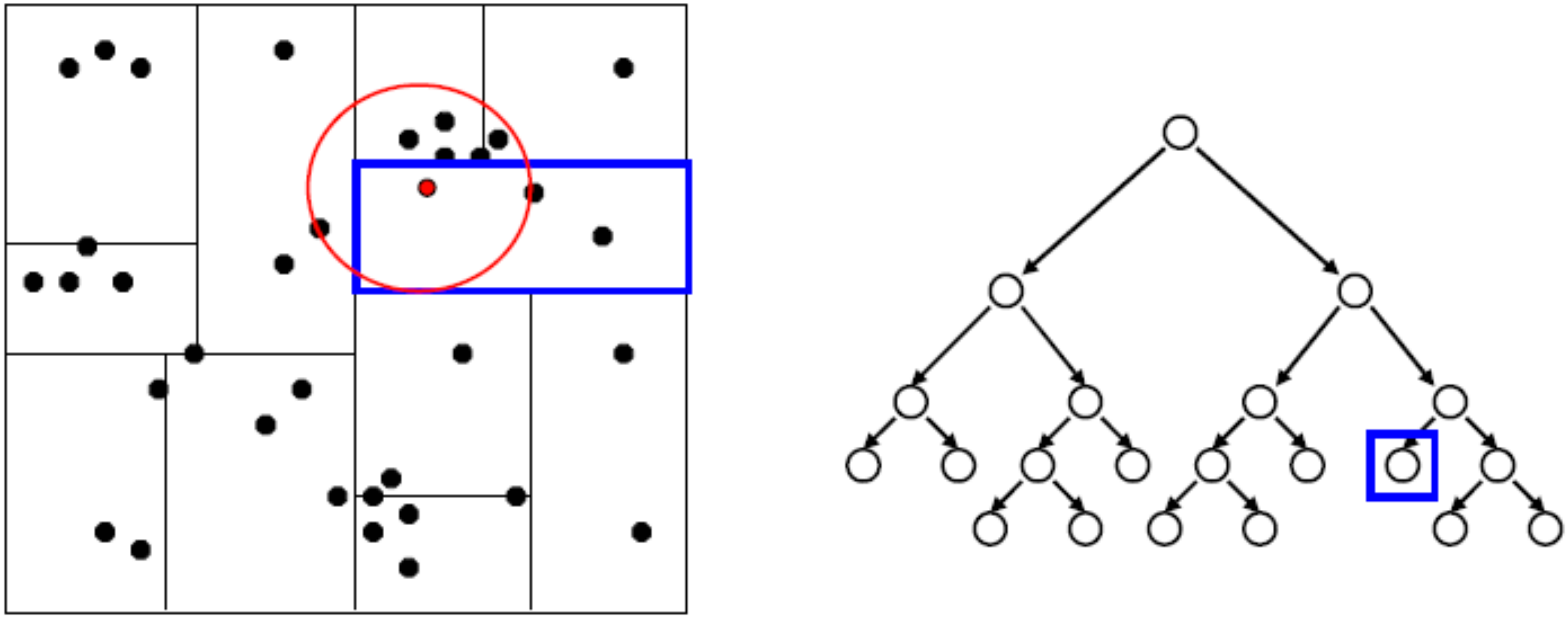
Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

NN Search by kd-tree



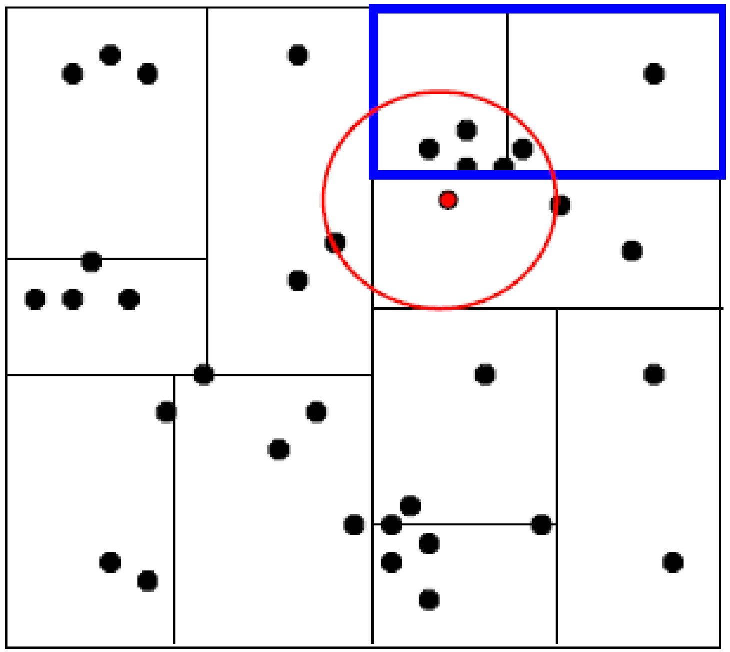
When we reach a leaf node: compute the distance to each point in the node.

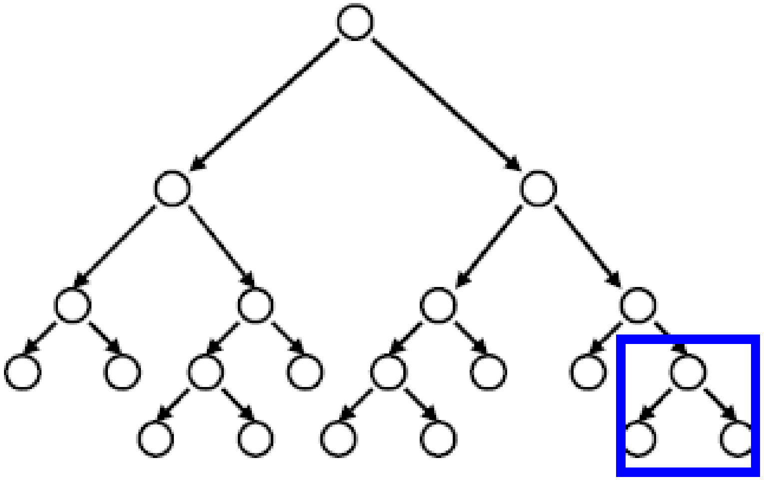
NN Search by kd-tree



When we reach a leaf node: compute the distance to each point in the node.

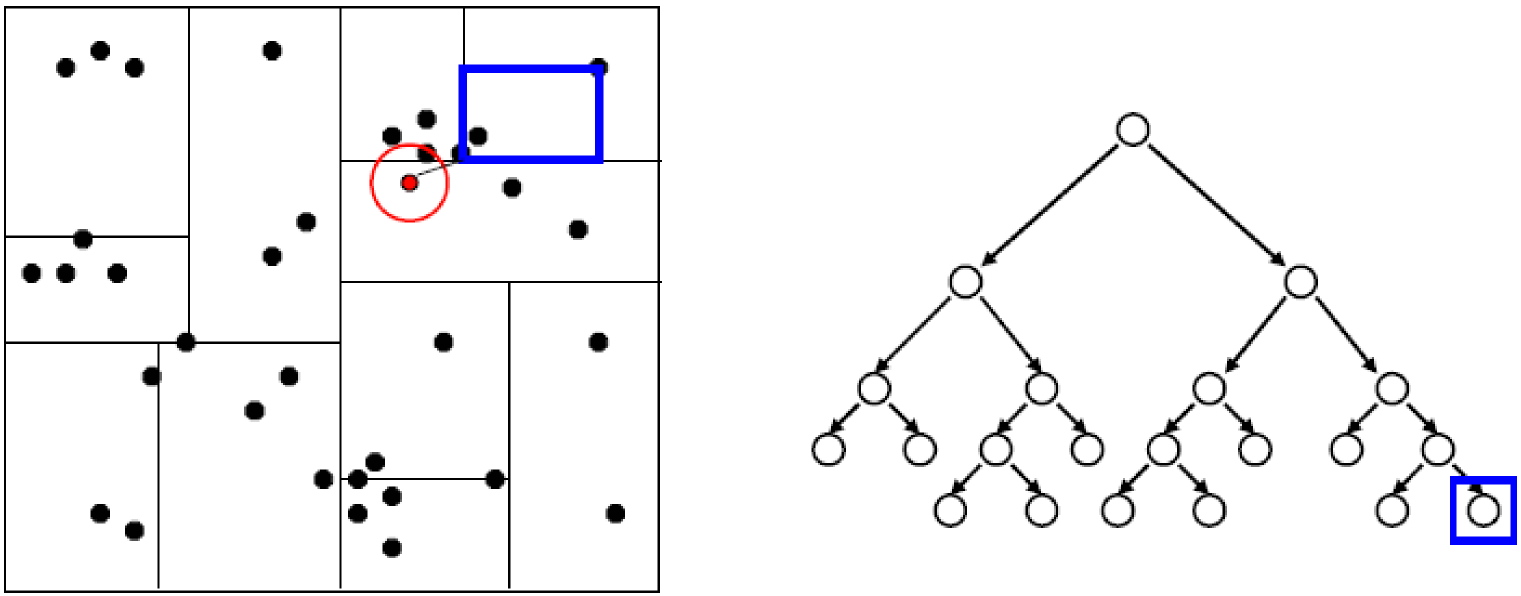
NN Search by kd-tree





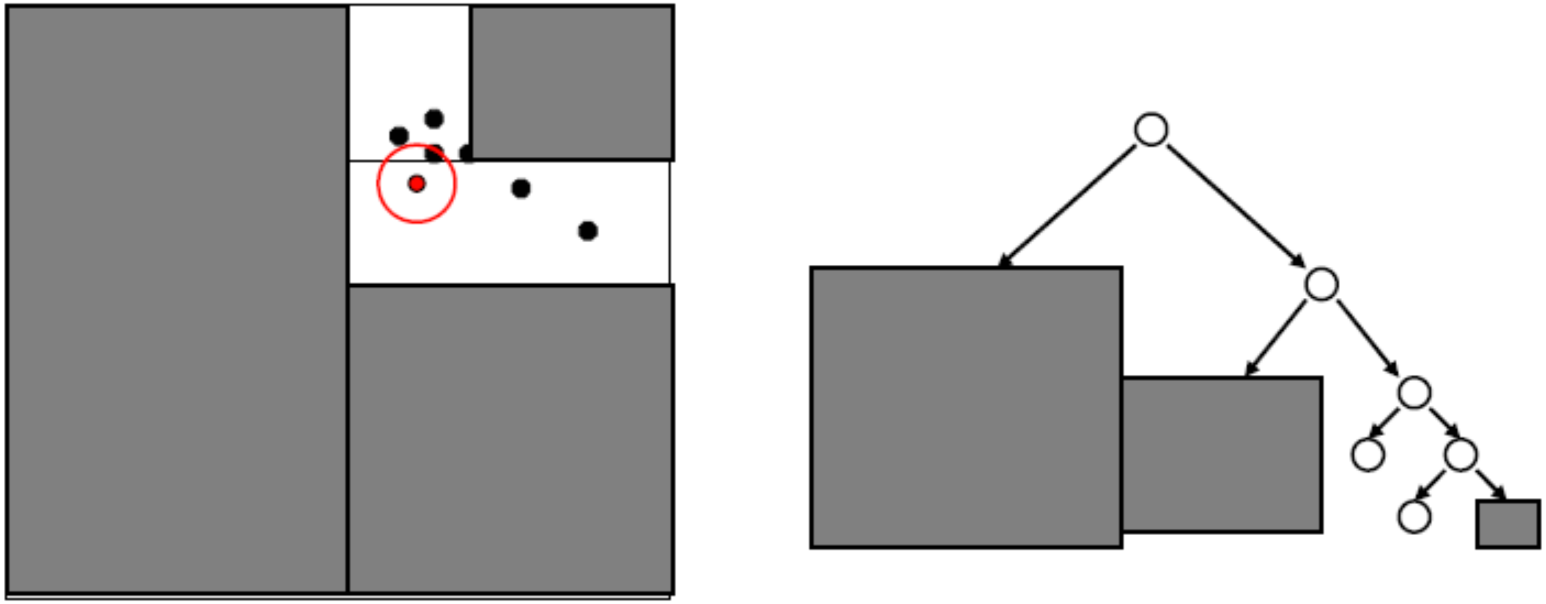
Then we can backtrack and try the other branch at each node visited.

NN Search by kd-tree



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

NN Search by kd-tree



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

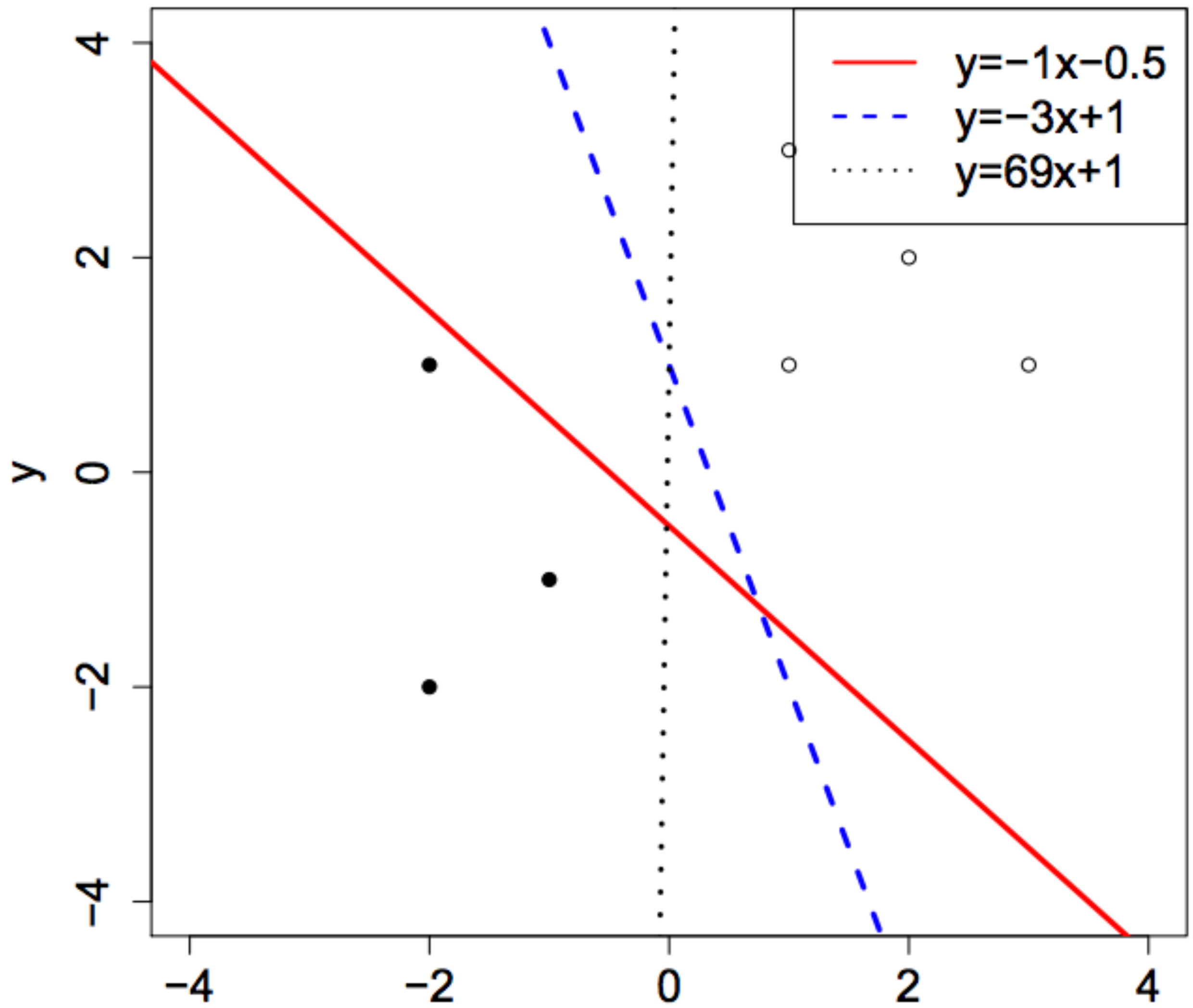
Curse of dimensionality

- If only 2 attributes (out of say 30) are relevant to the target class, kNN is easily misled by the high-dimensionality of data
- Solution
  – Do feature selection before applying kNN

Other configurations of kNN

- Instead of choosing top-k nearest neighbors
  - Pick nearest neighbors within some distance
- Other ways of aggregating responses instead of simple majority
  - Weighted k-nearest neighbors

Linear classifier



What forms a “model” in a linear classifier?

- *Answer*: The geometry of the line (in 2D) or plane (in higher dimensions)
  - The coefficients of the linear expression of the plane

Some math

- A hyperplane is typically given by

$$w_1x_1 + w_2x_2, \ldots, +w_dx_d + w_0 = 0$$

- Slope of the hyperplane defined by
  - $(w_1, w_2, \ldots, w_d)$
- Intercept of the hyperplane defined by
  - w_0

Using a hyperplane as a classifier

- Let

$$g(\mathbf{x}) = w_1 x_1 + w_2 x_2, \ldots, + w_d x_d + w_0$$

- Then

$$y = \text{sign}(g(\mathbf{x})) = \begin{cases} +1 & \text{if } g(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

All this sounds good, but..

- How do we learn a classifier?
  - Learning here means “estimating the weights **w**”
- Solution:
  - Mistake-driven approach
  - Guess all weights
  - Repeat
    - See which examples you make a mistake on
    - Nudge/adjust the weights in the direction to reduce the error

Does this work in practice?

- Yes, but (a big BUT)
  - Only if the examples are linearly separable in the first place
    - Difficult to know this in high dimensions!
- What do you do if examples are not linearly separable?
  - Come up with more complex constructs
    - One hyperplane = one neuron
    - Lots of hyperplanes = Neural network
    - Zillions of hyperplanes = Deep neural network

What have we learned so far?

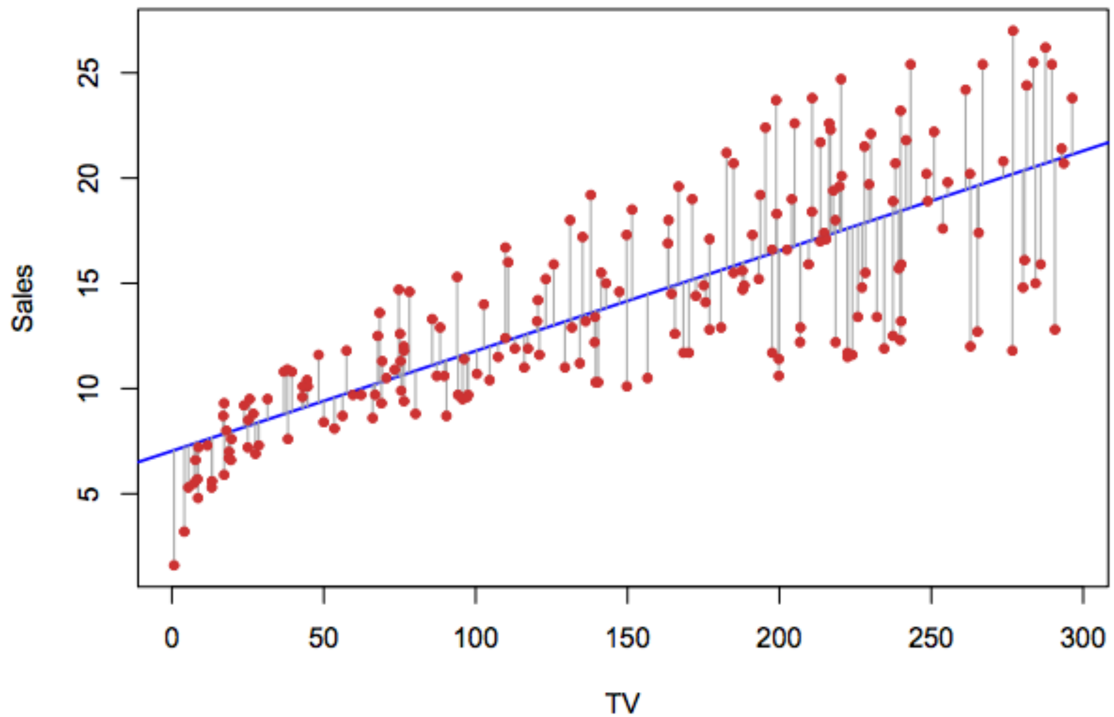
- Classifiers
  - Decision Trees
  - Naïve Bayes
  - Nearest neighbors
  - Linear classifiers
- More fancy methods
  - Logistic regression classifier
    - A marriage of Naïve Bayes and linear classifiers
  - Support vector machines (SVMs)
    - Fancier, more robust, linear classifiers

Regression

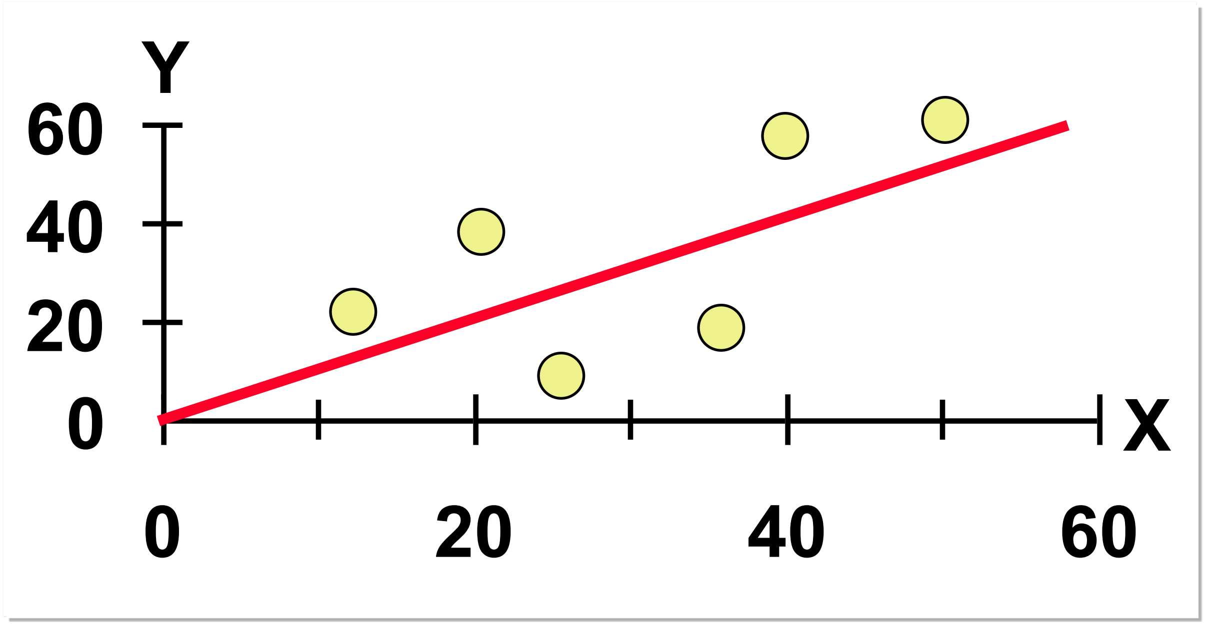
- The output/target variable is not a discrete quantity (like PlayTennis) but a continuous outcome
- Many existing methods of classification can be adapted to regression

Linear regression

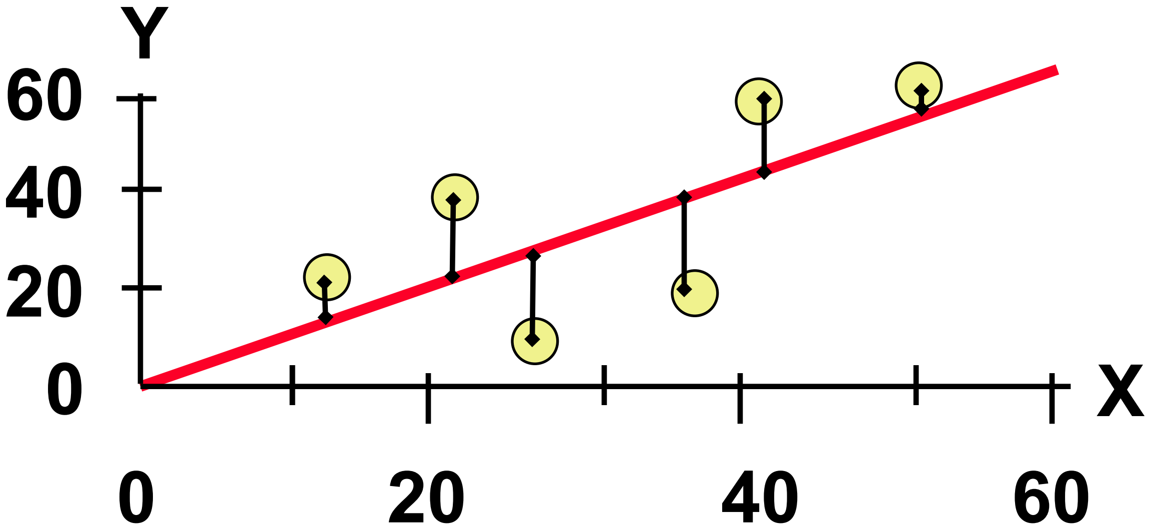
- Relationship between TV advertising budget and sales
  - One dependent var and one independent var



What does it mean to draw a line through the points?



What constitutes a good fit?

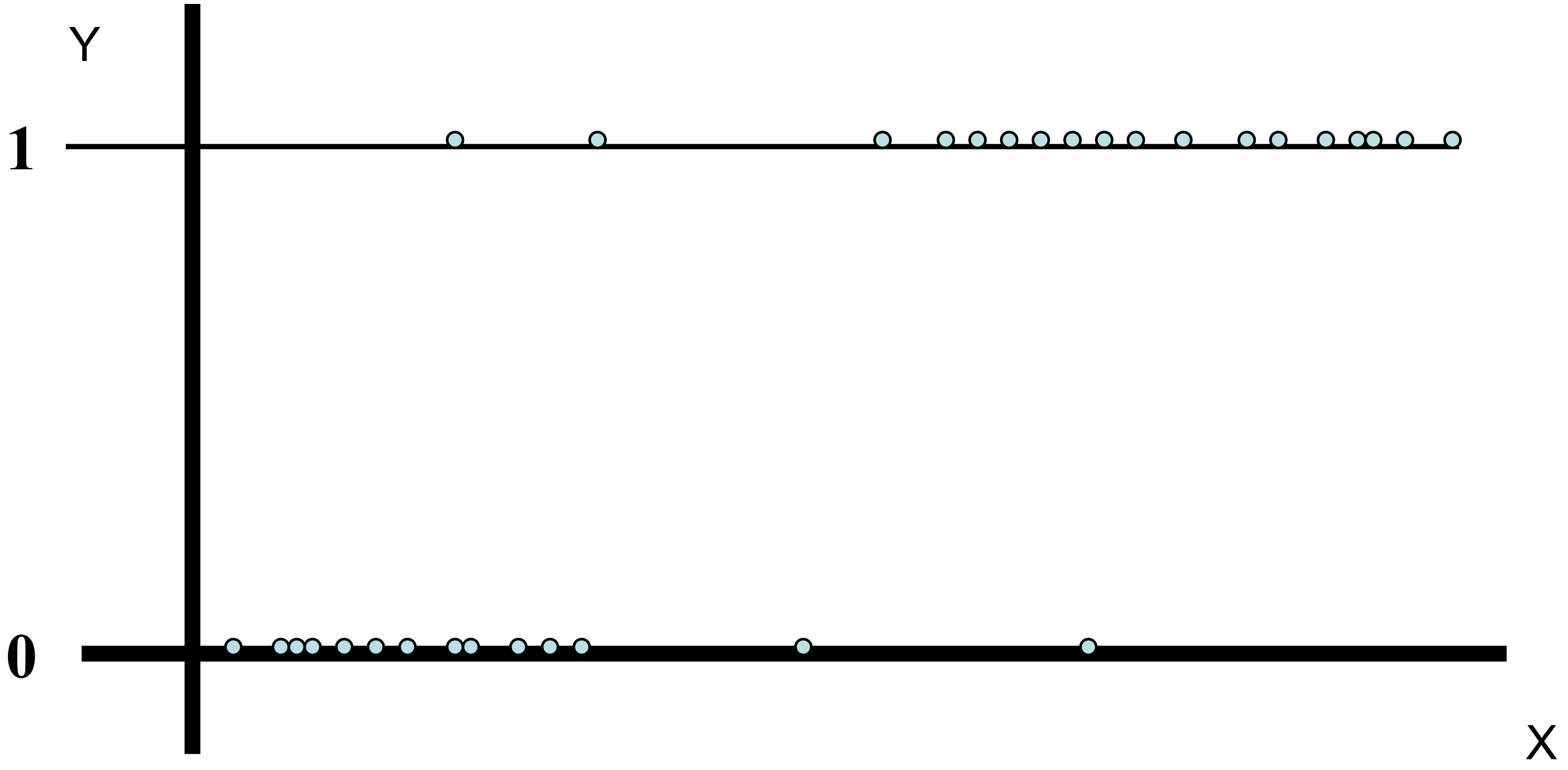


Ordinary least squares fit

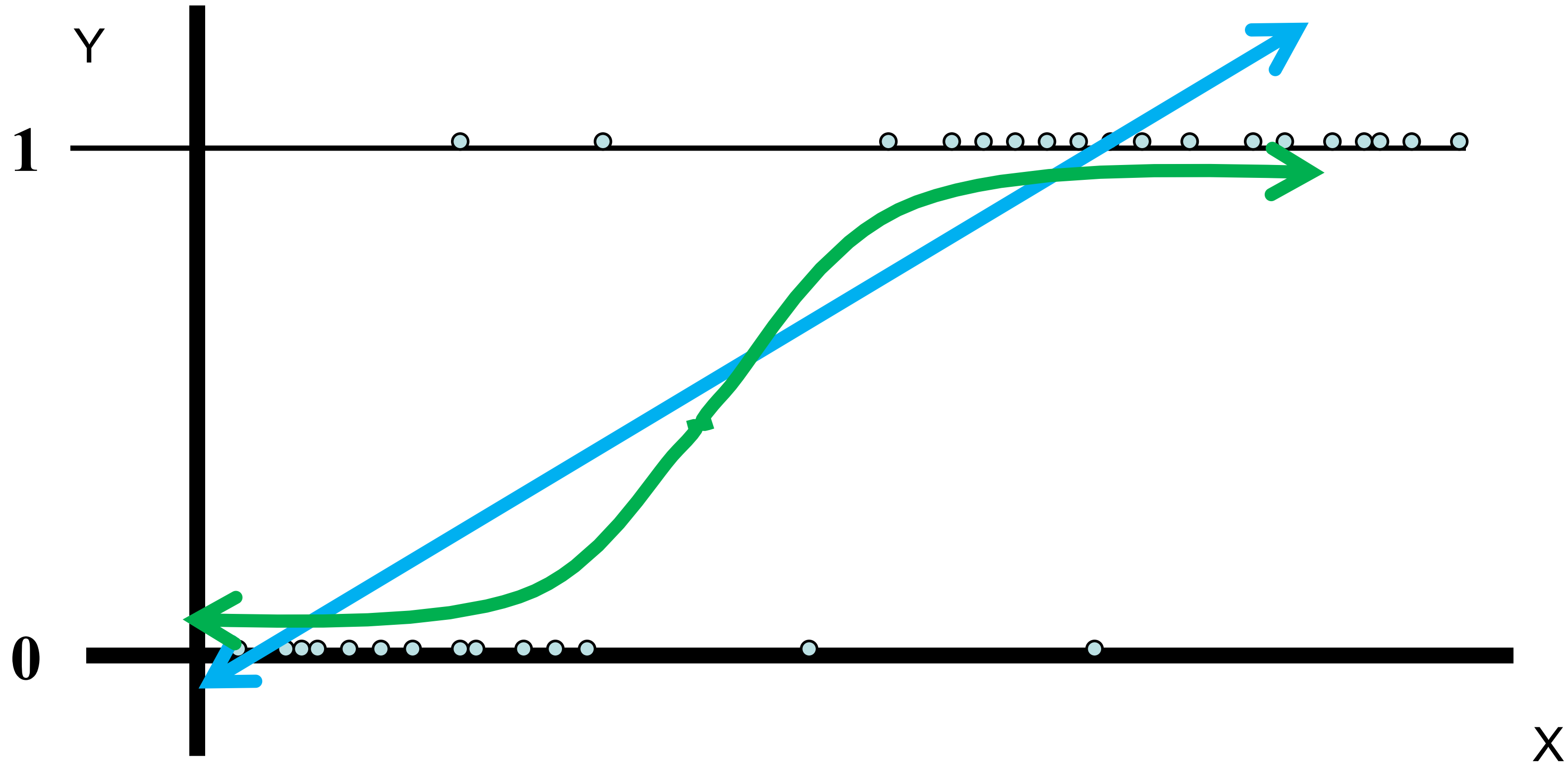
- Find the coefficients of the hyperplane such that
  - Sum of squared errors from predicted to actual values is minimized
  - Also called “linear least squares”
- What is ***ordinary*** about this?
  - The way we measure discrepancy is ordinary
  - Can instead use perpendicular distances!

Logistic regression

- Suitable for binary regression
  - (e.g., predicting probabilties)



How logistic regression works



Logistic regression details

- "Logit" model
  - $\ln[p/(1-p)] = \alpha + \beta X$
- p is the probability that the event occurs
  - p/(1-p) is the "odds ratio"
  - LHS is the log odds ratio
  - RHS is a regular linear expression for a plane

kNN regression

- Is exactly what it sounds like! ☺
- Combine predictions for nearby points
  - Averaging
  - Interpolation
  - Local linear regression
  - Local weighted regression

What we have learned thus far

- Lazy methods
  - k-nearest neighbors (both classification and regression)
- Linear methods
  - Again both classification and regression