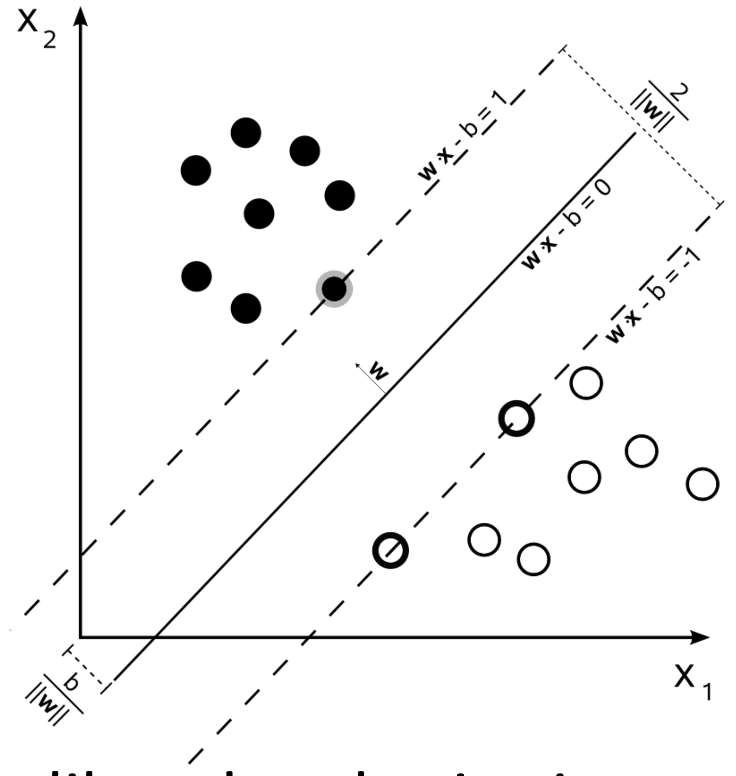
Instance Based Learning

Review of Types of Classifiers



x_2
x_1
$\mathbf{w}\cdot\mathbf{x} - b = 1$
$\mathbf{w}\cdot\mathbf{x} - b = 0$
$\mathbf{w}\cdot\mathbf{x} - b = -1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$

We have studied 3 types of classifiers:

1. Create a model for y (output) as a function of attributes. e.g. Perceptron, ANN, SVM
   $y = \text{sign}(w^T x)$

2. Probabilistic (Generative) classifiers
   $P(Y \mid X) \propto P(X \mid Y) * P(Y)$
   Probability of class membership estimated from likelihood and prior in the training data.

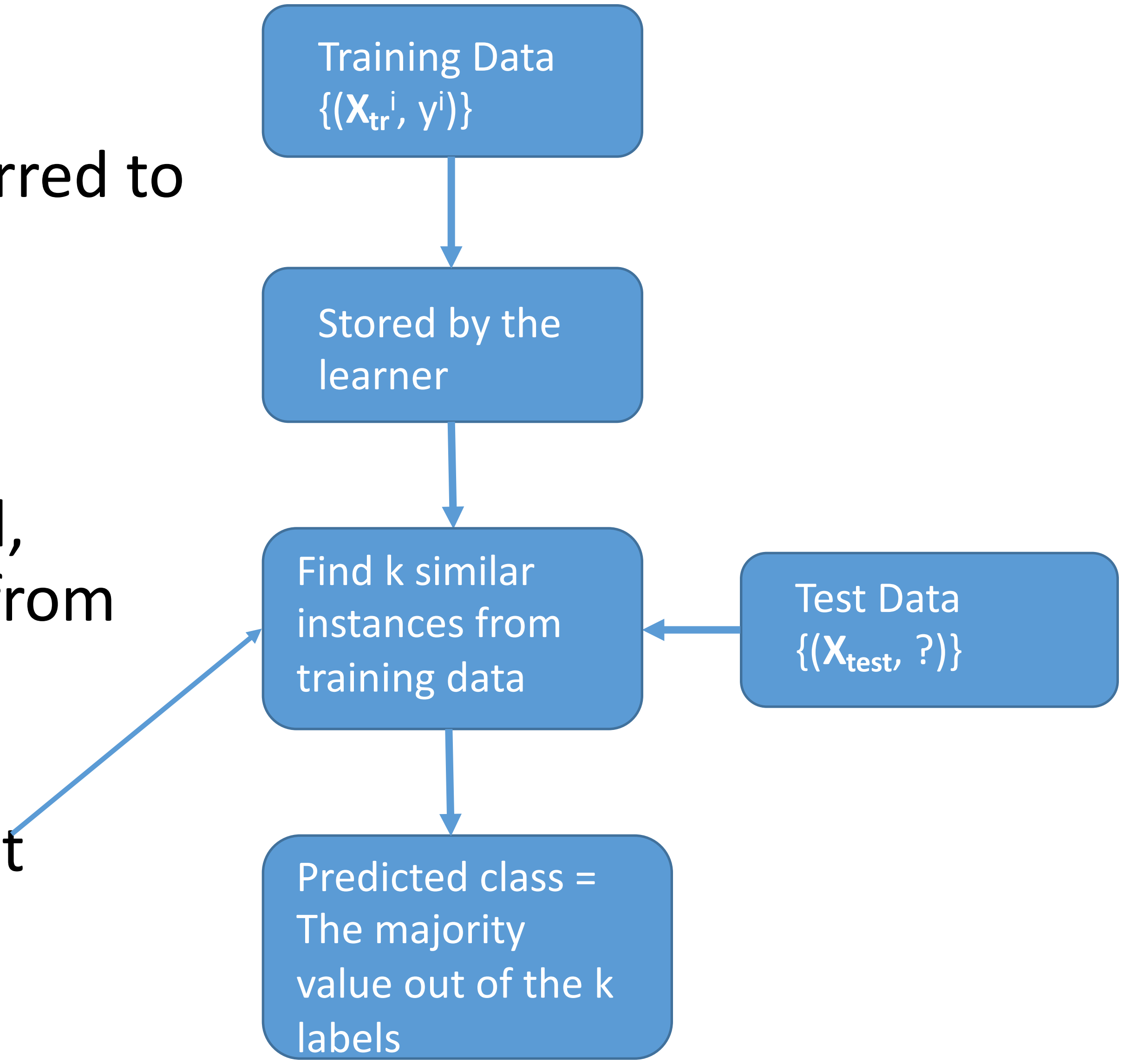
3. Discriminative classifiers –
   Create a model for P(Y|X) – Logistic Regression

Today's topics

- Instance Based Learning
  - k-Nearest Neighbors
  - Locally Weighted Regression (LWR)
- Practice Questions
- Reading:
  Chapter 8 of Tom Mitchell

Instance Based Learning - Classification

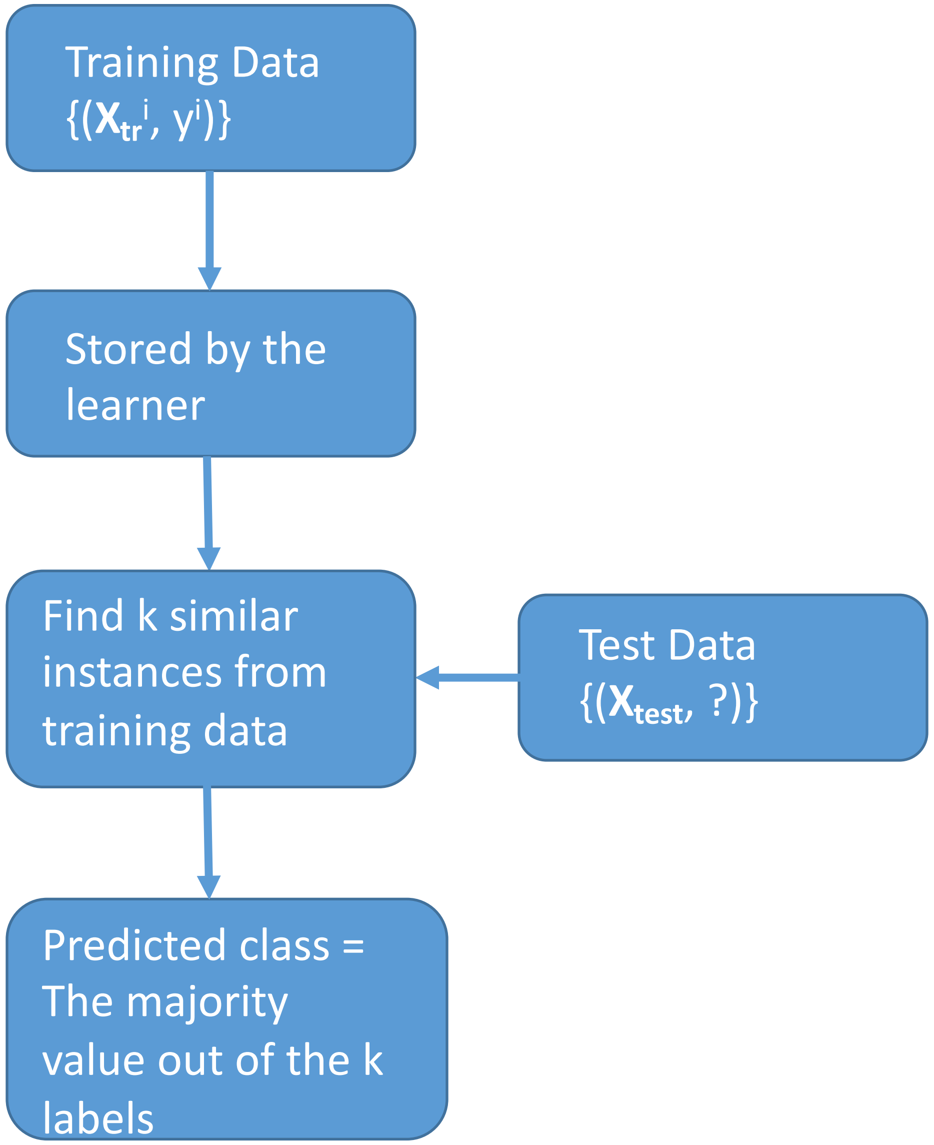
- Learning phase consists of simply storing all the training examples [referred to as instances]
- In fact there is no model creation, just storing the data.
- When a query (test) data is presented, a set of similar instances is retrieved from memory and used to classify the data.
- k-Nearest Neighbors (kNN) is the most popular IBL algorithm.



Training Data $\{(\mathbf{X}_{tr}^i, y^i)\}$
Stored by the learner
Find k similar instances from training data
Test Data $\{(\mathbf{X}_{test}, ?)\}$
Predicted class = The majority value out of the k labels

k- Nearest Neighbors (kNN)

- What happens during testing phase?
  - k similar items are scanned
  - their class labels are identified
  - majority of the class labels gives the predicted class label.
- What are the parameters?
  - Similarity metric
  - Value of k


Training Data $\{(\mathbf{X}_{tr}^i, y^i)\}$
Stored by the learner
Find k similar instances from training data
Test Data $\{(\mathbf{X}_{test}, ?)\}$
Predicted class = The majority value out of the k labels

k-NN Algorithm

Training data: Labeled instances {$\mathbf{X}^{\mathbf{i}}$, y^i}

Training Phase: Just store the examples

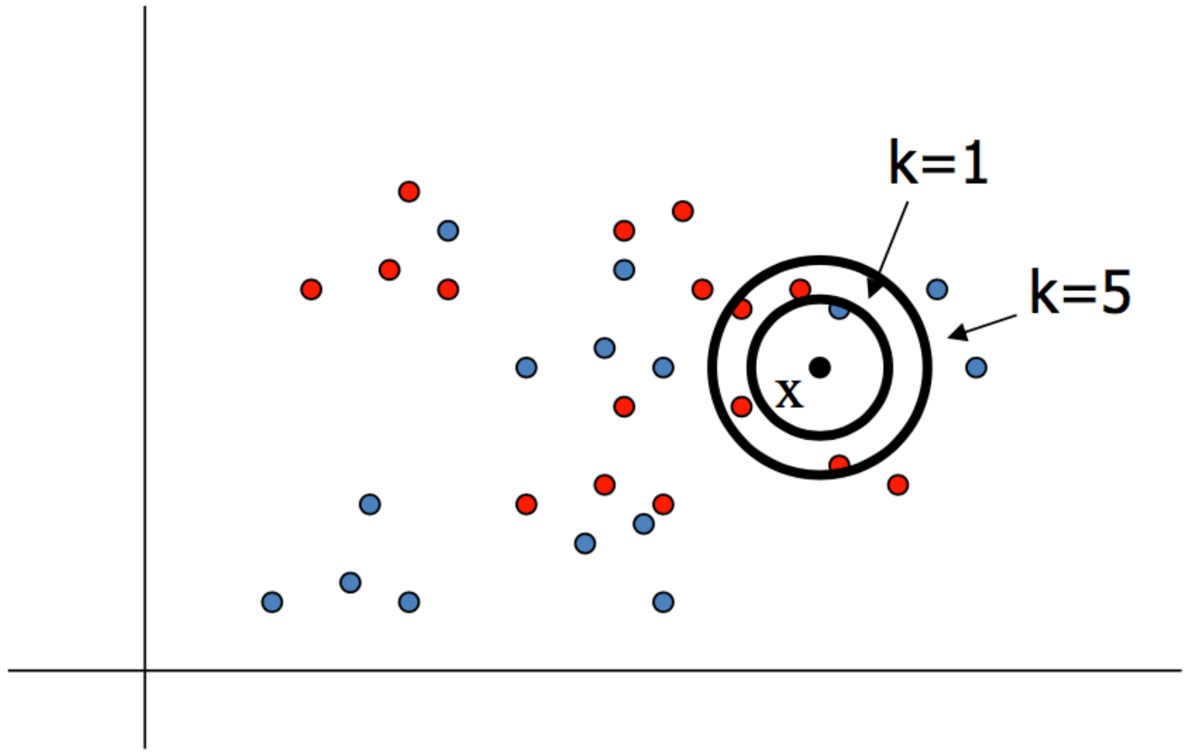
Test Phase:
Given a test instance $\mathbf{X}^{\mathbf{q}}$, take a vote among its k-nearest neighbors for the class value:

$$\hat{f} = majority(f(X^i))$$

where $i \in Neighbors(K)$

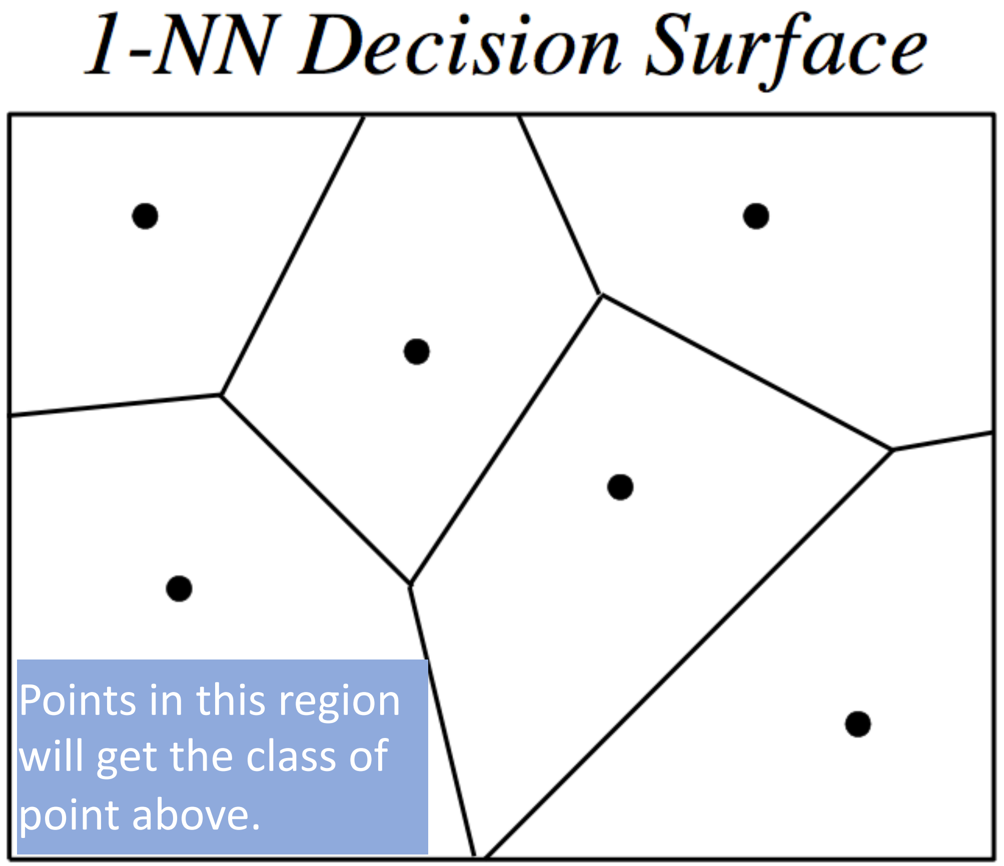
Example

- Image on the right shows a k-NN example with k = 1 and k = 5.



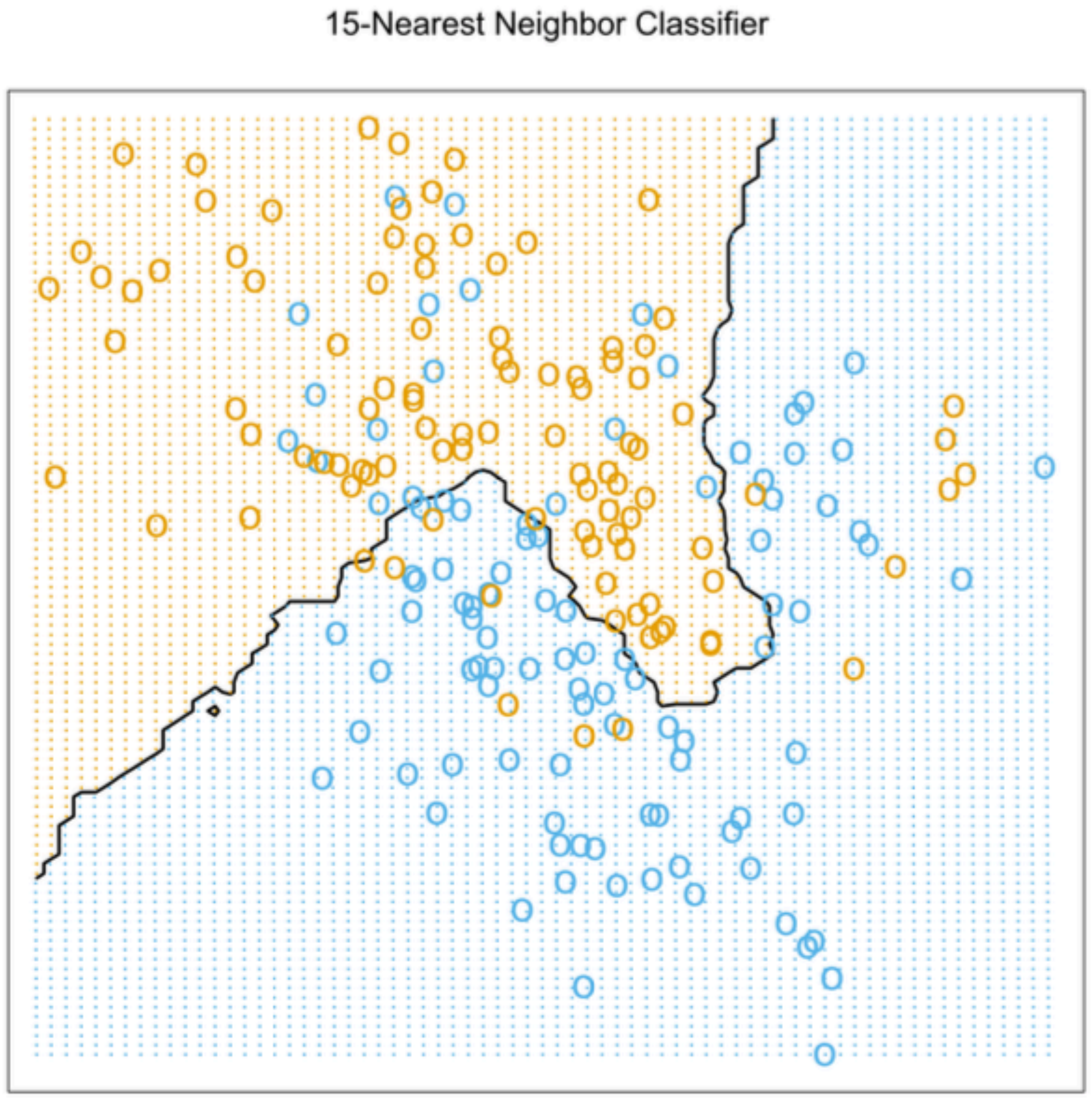
Decision Boundary

- The decision boundary for a 1-NN case can be shown as:
- This forms a Voronoi diagram,
  Defn: partitioning of a plane into part based on distance to specific points.



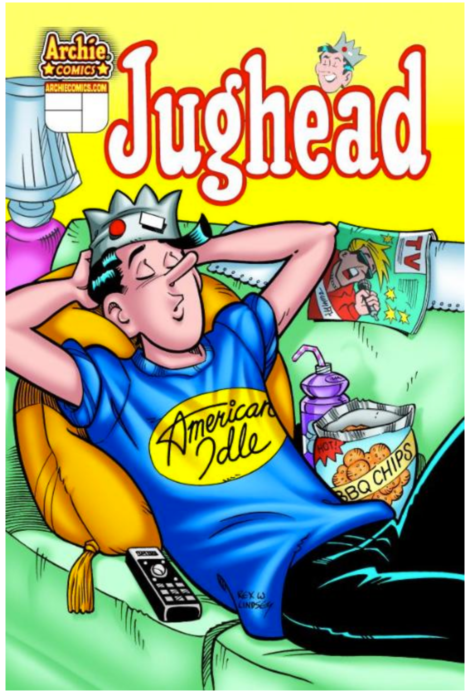
Example

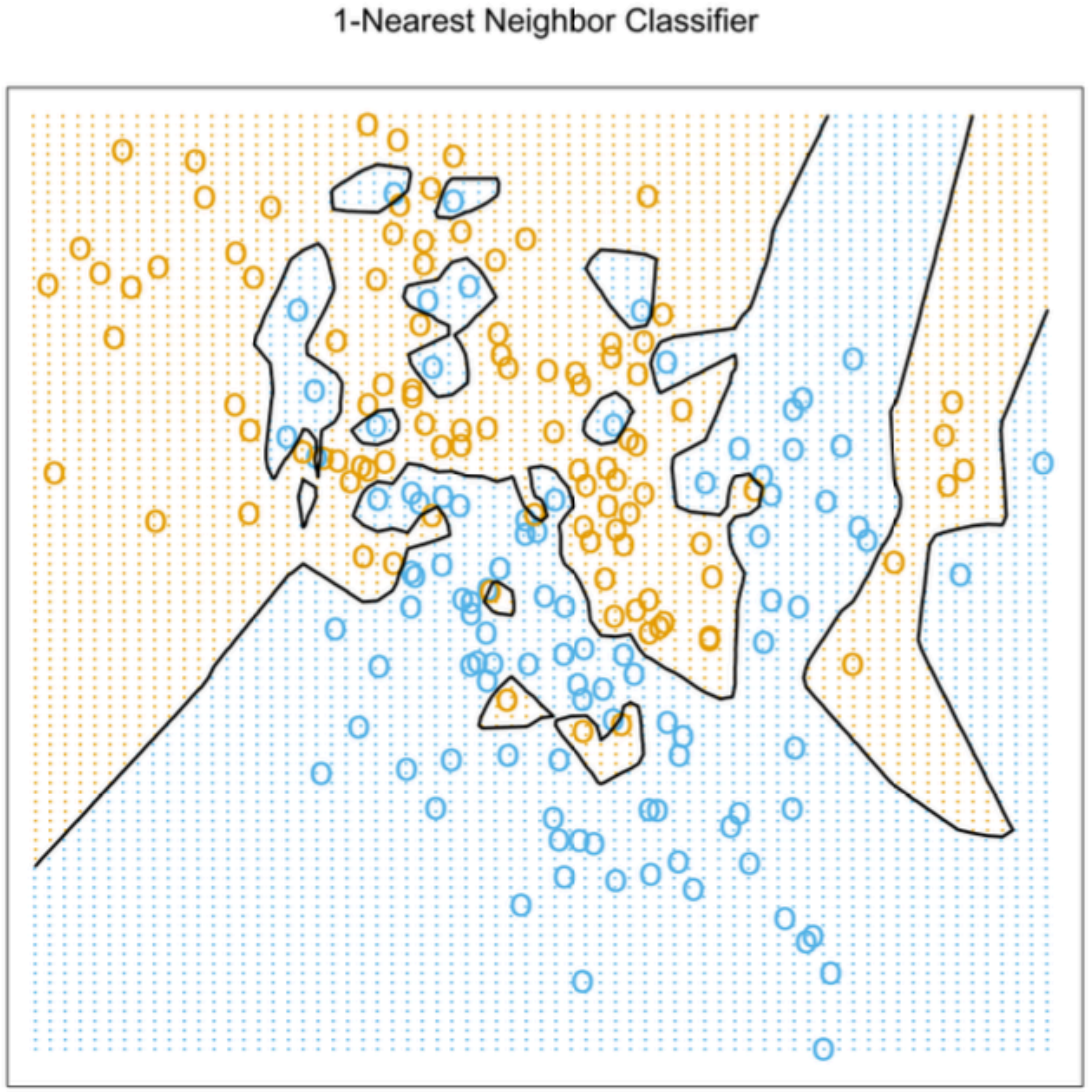
- Image on the right shows a k-NN example with k = 15.



Example

- Image on the right shows a k-NN example with k = 1.
- Point to note:
  You can get really complex decision boundaries by this very simple model.
- It pays to be lazy sometimes.





Distance Metric

- k-NN is critically dependent on the distance metric.
- For a function D to be used as a distance metric, it should satisfy certain properties:
  - ❑ $D(a,b) \geq 0$ (Positivity)
  - ❑ $D(a,b) = 0 \text{ iff } a = b$ (Reflexive)
  - ❑ $D(a,b) = D(b,a)$ (Symmetric)
  - ❑ For any other vector c,
    $D(a,b) + D(b,c) \geq D(a,c)$ (Triangle Inequality)

- What are some of the commonly used distance metrics?

Similarity Metrics

Some commonly used similarity metrics for vectors **X**^{i} and **X**^{j} having d dimensions each:

- Euclidean distance:

$$D(\boldsymbol{X}^i, \boldsymbol{X}^j) = \left[\sum_{a=1}^{d} \left(x_a^i - x_a^j\right)^2\right]^{1/2}$$

- Manhattan distance:

$$D(\boldsymbol{X}^i, \boldsymbol{X}^j) = \left[\sum_{a=1}^{d} \left|x_a^i - x_a^j\right|^1\right]^1$$

Example:

If X1 = $(2, -1)^T$ and X2 = $(-2, 2)^T$

Euclidean Distance (D)=

$$\left[\left(2-(-2)\right)^2 + (-1-2)^2\right]^{\frac{1}{2}} = 5$$

Manhattan Distance (D)=

$$[|2-(-2)| + |-1-2|] = 7$$

Similarity Metrics

Both of these are generalization of the Minkowski distance (or norm)

- Minkowski distance of order p:

$$L_p = D(\boldsymbol{X}^i, \boldsymbol{X}^j) = \left[\sum_{a=1}^{d} \left|x_a^i - x_a^j\right|^p\right]^{1/p}$$

- p can range from 1 to ∞.

Trivia: p cannot be less than 1, otherwise it would violate triangle inequality.
It's fun exercise to show this. Try it on your own.
Hint: Set a = (0, 0) b = (1, 1) and c = (0, 1) and p = ½
Does it still satisfy the triangle inequality?

A word of caution

- For these distance metrics to work nicely, the attributes must be scaled before using them.
- We have done this earlier in neural net training.
- In many cases, you might want to weight each attribute differently. This is called weighted distance.
- Example of weighted Euclidean distance:

$$D_w\left(\boldsymbol{X}^i, \boldsymbol{X}^j\right) = \left[\sum_{a=1}^{d} w_a\left(x_a^i - x_a^j\right)^2\right]^{1/2}$$

where w_a is weight for the ath attribute

Example:

Data for UTD students' GPA (attribute) and getting internship (class) is presented below:

GPA	2.6	2.8	2.85	3.1	3.2	3.3	3.4	3.55	3.6	3.7	3.75	4.0	4.0
Internship	0	1	1	0	1	1	0	1	0	1	0	0	1

Using Manhattan distance, what will be the prediction of k-NN for a student with GPA of 3.5 in the following cases:

1. k = 1
2. k = 3
3. k = 5

Example:

Data for UTD students' GPA (attribute) and getting internship (class) is presented below:

GPA	2.6	2.8	2.85	3.1	3.2	3.3	3.4	3.55	3.6	3.7	3.75	4.0	4.0
Internship	0	1	1	0	1	1	0	1	0	1	0	0	1

Using Manhattan distance, what will be the prediction of k-NN for a student with GPA of 3.5 in the following cases:

1. k = 1: Nearest neighbor: {(3.55, 1)} => Majority class = 1
2. k = 3: Nearest neighbors: {(3.55, 1), (3.4, 0), (3.6, 0) } => Majority class = 0
3. k = 5: Nearest neighbors: {(3.55, 1), (3.4, 0), (3.6, 0), (3.3, 1), (3.6, 1) }
   => Majority class = 1

Handout

- Let's practice some questions from the handout.

Handout

- Let's practice some questions from the handout.
- Realization:
  - The calculations become more and more tedious as k increases and more so when dimensions increase.
  - More work is done during the testing phase than during the training phase.

Advantages and Disadvantages of k-NN

Advantages:

- Training is very fast
- Can learn complex target functions easily
- Easy to program

Disadvantages:

- Slow at query time
- Lots of storage and processing in memory
- Doesn't scale well in higher dimensions Curse of Dimensionality
- Easily tricked by noisy data items and irrelevant attributes
- Value of k can vary results significantly

k-NN for Continuous Output

- We presented k-NN for classification, but it can easily be used to approximate functions where the output is continuous.
- How? Simply replace the majority function by the average function:
  Instead of:

$$\hat{f} = majority(f(X^i))$$

where $i \in Neighbors(K)$

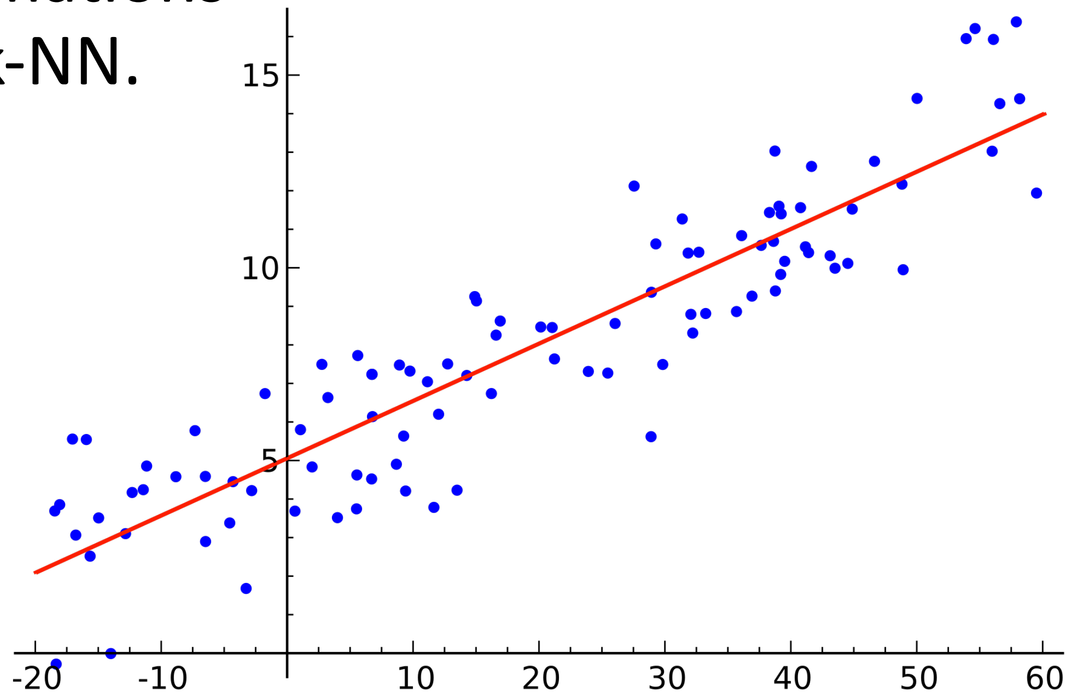
Use this:

$$\hat{f} = \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

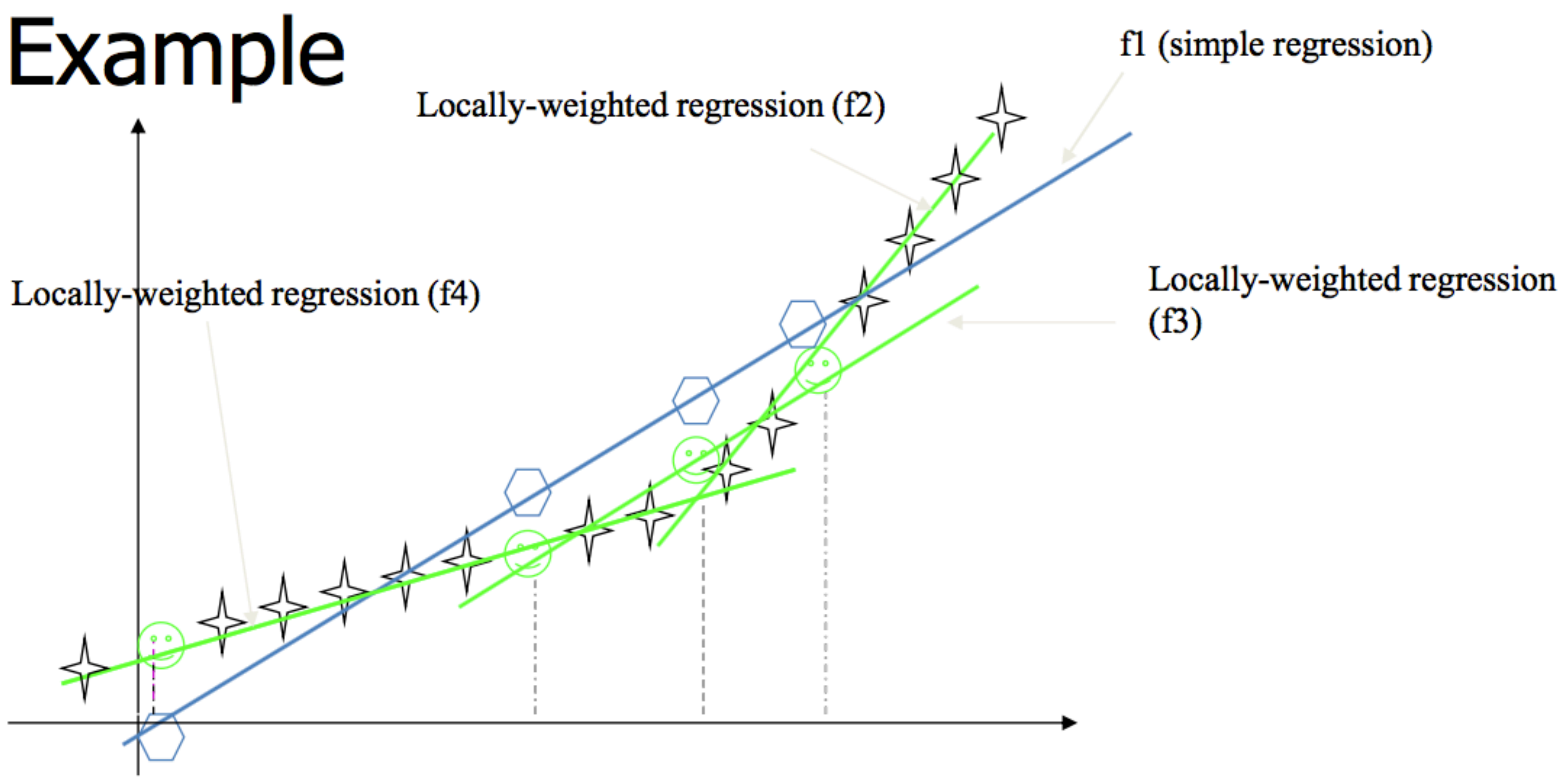
where $i \in Neighbors(K)$

Locally Weighted Local Regression

- In typical regression, we create a global model for the data.
- A better option? Creating local approximations using the idea of "neighborhood" and k-NN.
- The local functions produce piecewise approximation to the global function.



LWR Example



✧ Training data

⬡ Predicted value using simple regression

☺ Predicted value using locally weighted (piece-wise) regression

Yike Guo, Advanced Knowledge Management, 2000

Lazy vs Eager Learners

- Lazy: Wait for query before generalizing i.e. going beyond training data.
  -> Model doesn't propose a function until a query point arrives.
  -> When a query point arrives, a local search for the best function is done and it is outputted as the generalization for the test query.
  -> Model is adaptable.
- Eager: After all training data is received, a generalization function is outputted.
  -> ID3 algorithm creates a decision tree (generalization model) after incorporating all training data.
  -> A generalized linear regression outputs an approximation function after seeing all the training data.
  -> One model for all the query instances.

Parametric vs Non-Parametric Methods

- Parametric: A particular functional form is assumed, e.g., linear, Gaussian distribution, naïve Bayes
  -> may have high bias because the real data may not obey the assumed functional form.
  -> Parameter estimation is required.
  -> Advantage of simplicity – easy to estimate and interpret

- Non-Parametric: Distribution or density estimate is data-driven and relatively few assumptions are made a priori about the functional form.
  -> Parameter estimation is not as involved.

Thought Questions

- Can you think of a lazy algorithm for decision tree construction?
- Will it help if we are able to pre-process and remove irrelevant or redundant attributes?
- What is the time complexity of k-NN?