Machine Learning and Data Analytics
ME 5013- Fall 2019

Lecture 06

- K Nearest Neighbors
- Kernel Regression

The University of Texas at San Antonio™

Adel Alaeddini, PhD
Associate Professor of Mechanical Engineering
Advanced Data Engineering Lab
adel.alaeddini@utsa.edu

Disclosure: the slides are adopted from various online resources

- Tell me about your friends(*who your neighbors are*) and *I will tell you who you are*.

KNN – Different names

- K-Nearest Neighbors
- Memory-Based Reasoning
- Example-Based Reasoning
- Instance-Based Learning
- Lazy Learning

- A powerful regression/classification algorithm used in machine learning.
- K nearest neighbors stores all available cases and regress/classifies new cases based on a similarity measure (e.g distance function)
- A non-parametric lazy learning algorithm (An Instance- based Learning method).

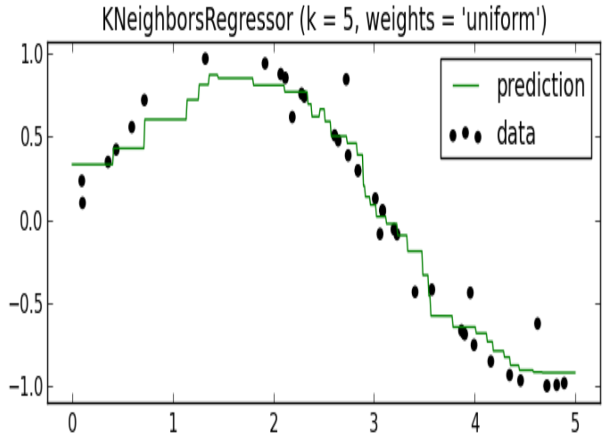
KNN Approach

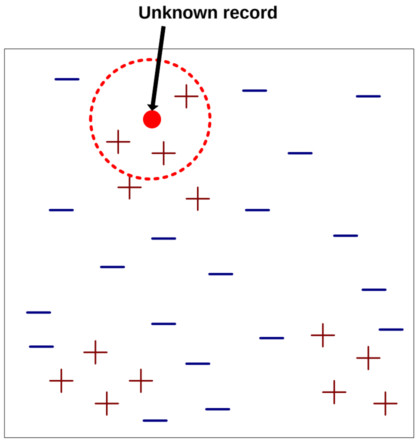
Regression

- An object (a new instance) value is estimated by the (weighted) average of its neighbor value.
- The weight and neighbors are identified based on *a distance function*

Classification

- An object (a new instance) is classified by a majority votes for its neighbor classes. (common class amongst its K nearest neighbors)
- The neighbors are identified based on *a distance function*





Simple distance measure

- **Euclidean Distance:** Simplest, fast to compute

$$d(x,y) = \|x - y\| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- **Cosine Distance:** Good for documents, images, etc.

$$d(x,y) = 1 - \frac{x \cdot y}{\|x\| \|y\|}$$

- **Jaccard Distance:** For set data:

$$d(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$$

- **Hamming Distance:** For string data:

$$d(x,y) = \sum_{i=1}^{n} (x_i \neq y_i)$$

Example

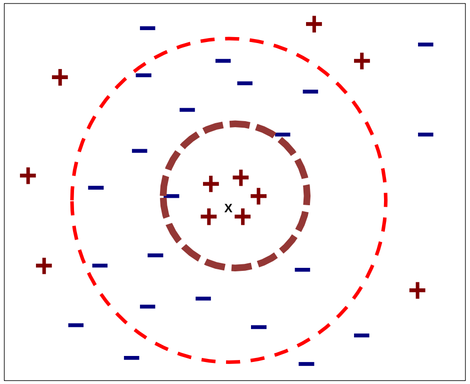
ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

- Calculate the mean value of the *k* nearest training examples:

$$f : \Re^d \rightarrow \Re \qquad \hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

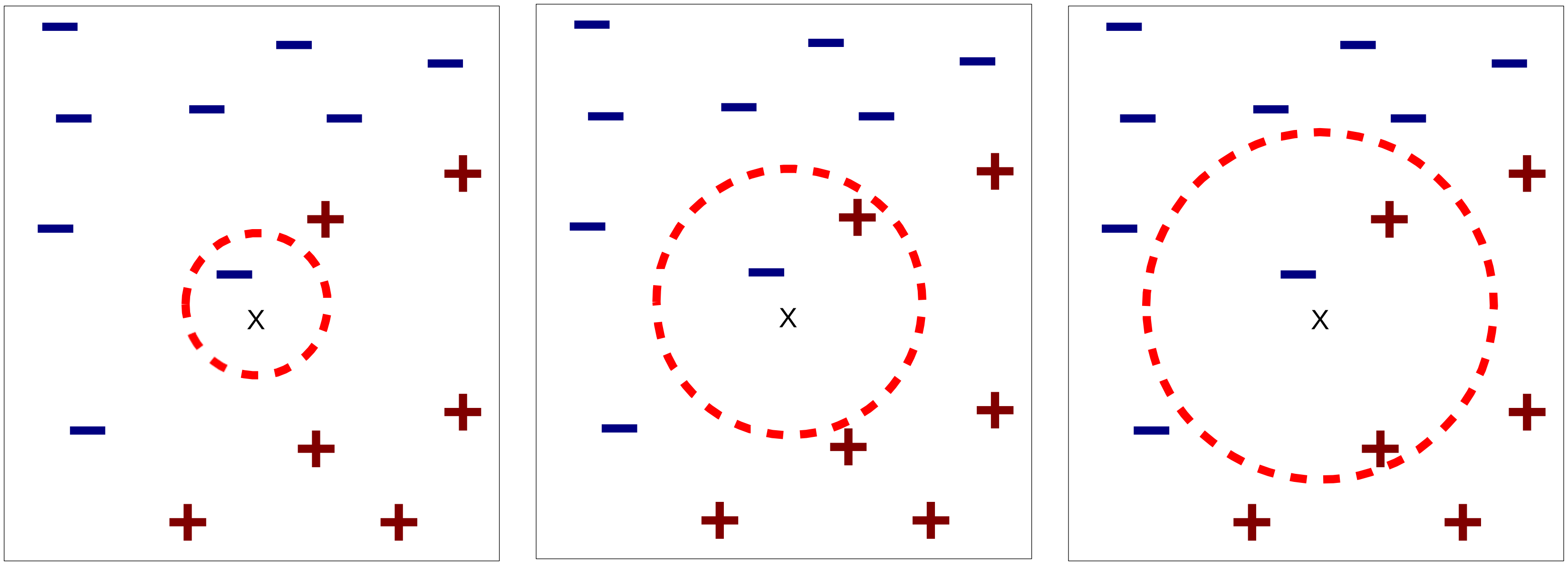
How to choose K?

- If K is too small it is sensitive to noise points.
- Larger K works well. But too large K may include majority points from other classes.



- Rule of thumb is K < sqrt(n), n is number of examples.

How to choose K?



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Feature Normalization

- Distance between neighbors could be dominated by some attributes with relatively large numbers.
- Arises when two features are in different scales.
- Important to normalize those features.
  - Mapping values to numbers between (0,1) ?

$$z_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$$

  - Mapping values to numbers between (-1 ,1) ?

$$z_i = \frac{x_i - \text{Average }(x_i)}{\text{Range}(x_i)}$$

KNN with weights

- Refinement to KNN is to weight the contribution of each *k* neighbor according to the distance to the query point x_q
- Greater weight to closer neighbors

Weight function

$$w_i = \begin{cases} \dfrac{1}{d(x_q, x_i)^2} & if \quad x_q \neq x_i \\ 1 & else \end{cases}$$

For continuous target functions

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

For discrete target functions

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\arg\max} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

Advantage and Disadvantages

Advantage

- Very simple and intuitive.
- Requires little tuning
- Can be applied to the data from any distribution.
- Good performance if the number of samples is large enough (Often performs quite well!

Disadvantage

- Prediction accuracy can quickly degrade when number of attributes grows (>20).
- Fooled by irrelevant features (attributes)
- Need distance/similarity measure and attributes that “match” target function.
- Must make a pass through the entire dataset for each classification. This can be prohibitive for large data sets.
- Choosing k may be tricky.
- Need large number of samples for accuracy.

Advantage and Disadvantages

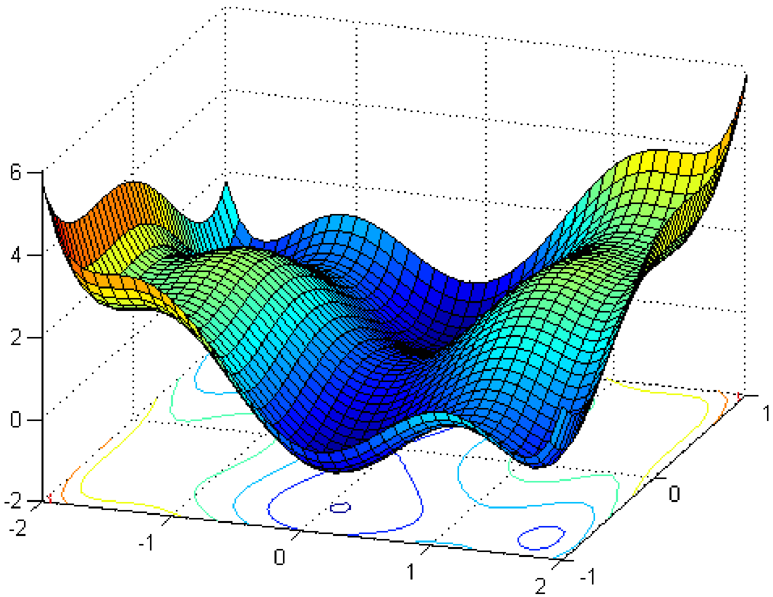
Advantage

- Very simple and intuitive.
- Requires little tuning
- Can be applied to the data from any distribution.
- Good performance if the number of samples is large enough (Often performs quite well!

Disadvantage

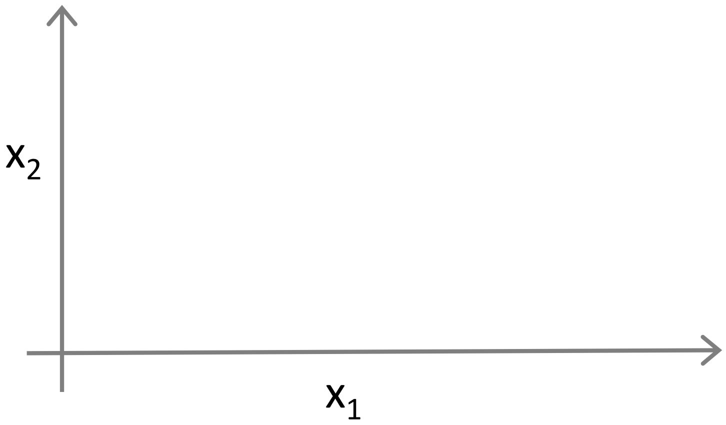
- Prediction accuracy can quickly degrade when number of attributes grows (>20).
- Fooled by irrelevant features (attributes)
- Need distance/similarity measure and attributes that "match" target function.
- Must make a pass through the entire dataset for each classification. This can be prohibitive for large data sets.
- Choosing k may be tricky.
- Need large number of samples for accuracy.

Non-linear structure



Is there a different / better choice of the features f_1 , f_2,…?

Kernel



Given x, compute new feature depending on proximity to landmarks $l^{(1)}, l^{(2)}, l^{(3)}$

Kernels and Similarity

$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$$

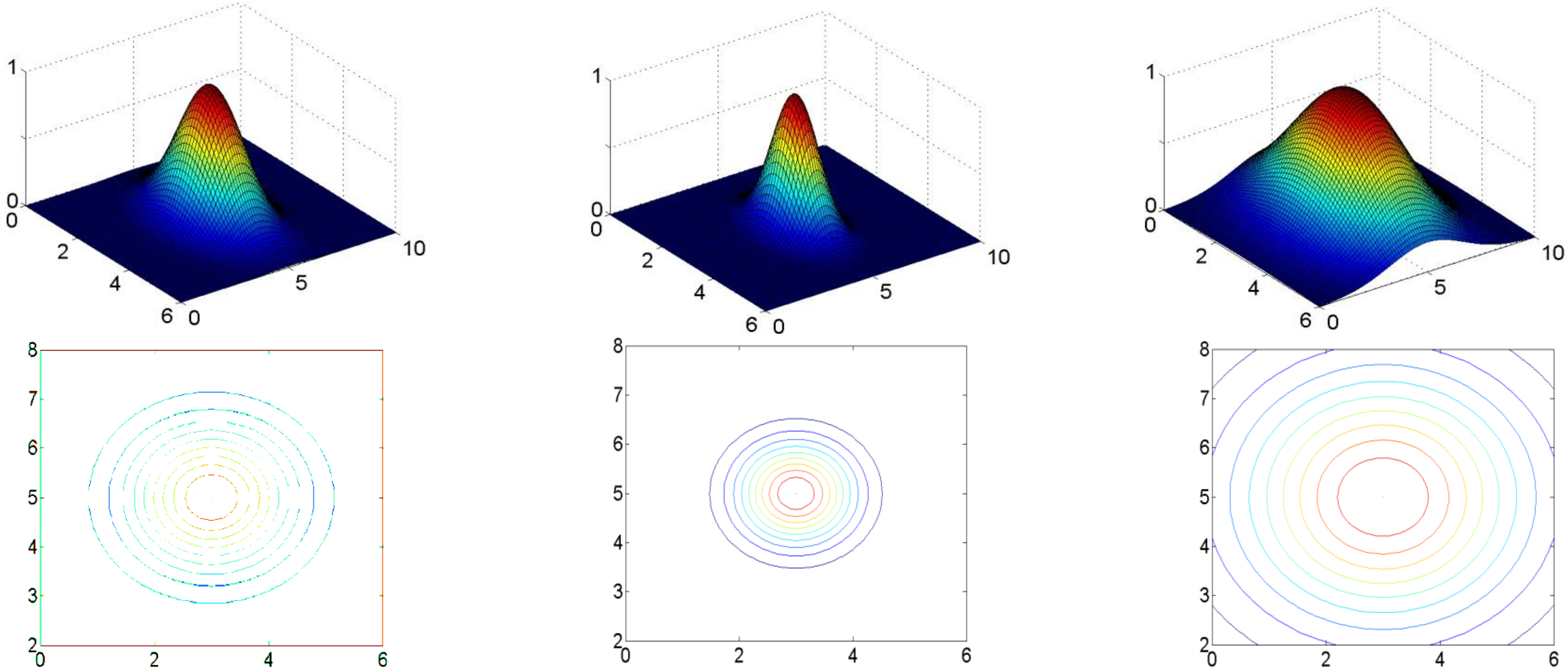
If $x \approx l^{(1)}$:

If x if far from $l^{(1)}$:

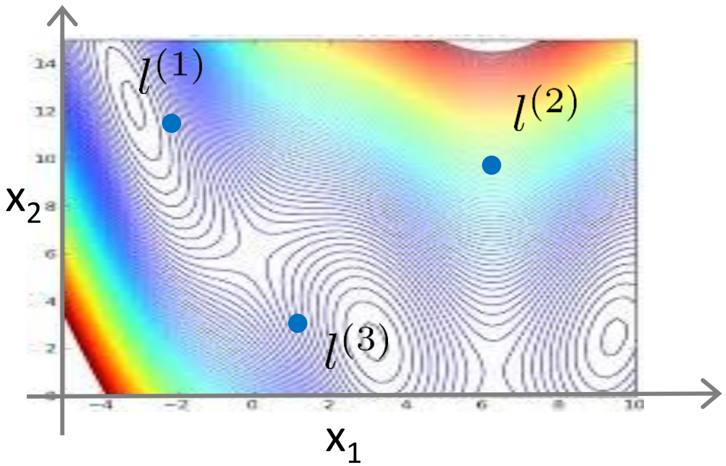
Example:

$$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

$$\sigma^2 = 1 \qquad \sigma^2 = 0.5 \qquad \sigma^2 = 3$$



Choosing the landmarks



Where to get $l^{(1)}, l^{(2)}, l^{(3)}, \ldots$?

Given x:

$$f_i = \text{similarity}(x, l^{(i)})$$
$$= \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right)$$

Choosing the landmarks

SVM with Kernels

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$,
choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$.

Given example x:

$$f_1 = \text{similarity}(x, l^{(1)})$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$\ldots$$

For training example $(x^{(i)}, y^{(i)})$:

The Nadaraya-Watson estimator

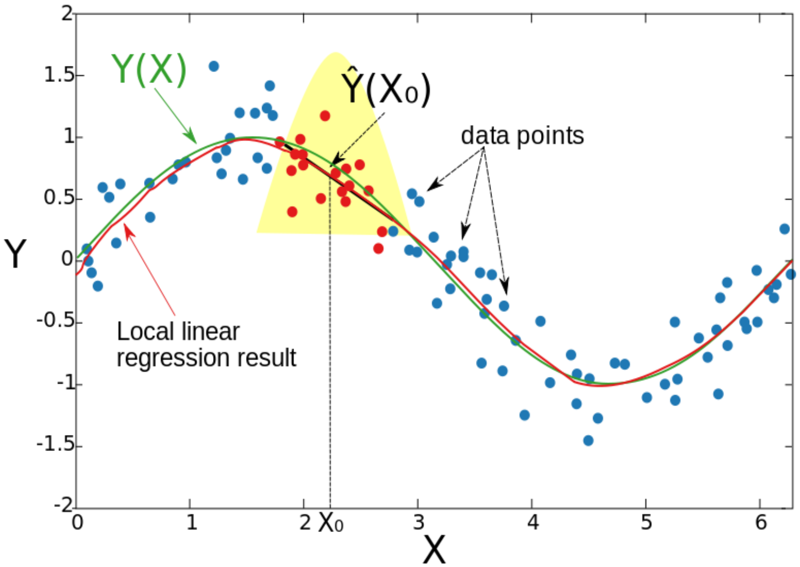
$$\widehat{m}(x) = \frac{n^{-1}\sum_{i=1}^{n} K_h\,(x - X_i)Y_i}{n^{-1}\sum_{i=1}^{n} K_h(x - X_i)}$$

Rewrite the Nadaraya-Watson estimator

$$\widehat{m}(x) = \frac{1}{n}\sum\nolimits_{i=1}^{n}\left(\frac{K_h\,(x - X_i)}{n^{-1}\sum_{i=1}^{n} K_h(x - X_i)}\right)Y_i$$

$$= \frac{1}{n}\sum_{i=1}^{n} W_{hi}(x)Y_i$$

- Weighted (local) average of *Yi*
- *h* determines the degree of smoothness.
- *h (also known as bandwidth) is half-width of the window centered on **x**.*
- *Nearest neighbor method can be used to determine h, i.e. 60% of data*



Relation to regression

The result is a **weighted least squares estimator**

$$\hat{\beta}(x) = (\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{W}\mathbf{Y}$$

where $\mathbf{X} = \begin{pmatrix} 1 & X_1 - x & \dots & (X_1 - x)^p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_n - x & \dots & (X_n - x)^p \end{pmatrix}$, $\mathbf{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}$ and

$$\mathbf{W} = \begin{pmatrix} K_h(x - X_1) & 0 & \dots & 0 \\ 0 & K_h(x - X_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K_h(x - X_n) \end{pmatrix}$$

Note:
This estimator varies with x (in contrast to parametric least squares)