ANALYTIXLABS

Machine Learning: k-Nearest Neighbour (kNN)

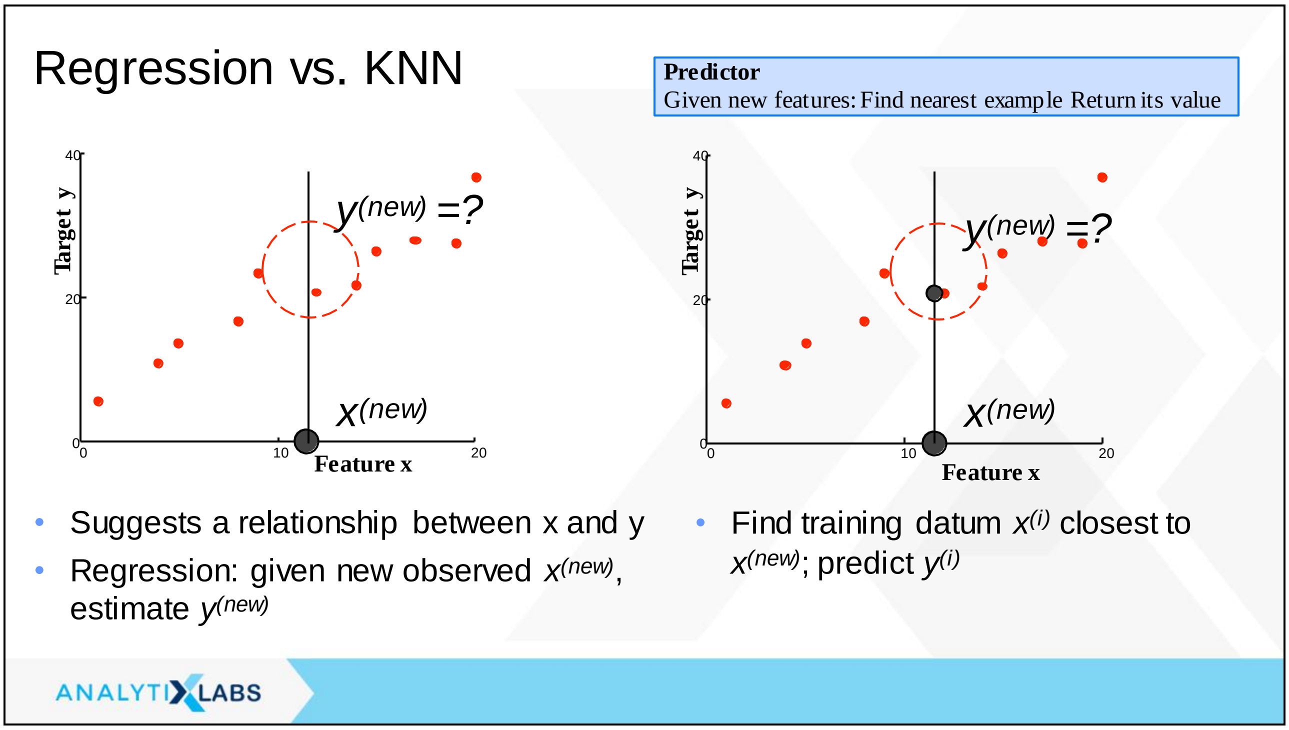
What is KNN?

KNN is non-parametric, instance-based and used in a supervised learning setting

•**Non-parametric** means it makes no explicit assumptions about the functional form of h, avoiding the dangers of mis-modeling the underlying distribution of the data. For example, suppose our data is highly non-Gaussian but the learning model we choose assumes a Gaussian form. In that case, our algorithm would make extremely poor predictions.

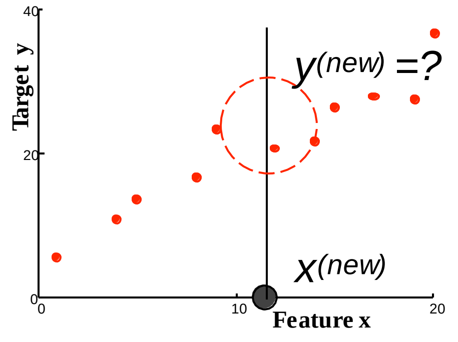
•**Instance-based** learning means that our algorithm doesn't explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as "knowledge" for the prediction phase. Concretely, this means that only when a query to our database is made (i.e. when we ask it to predict a label given an input), will the algorithm use the training instances to spit out an answer.

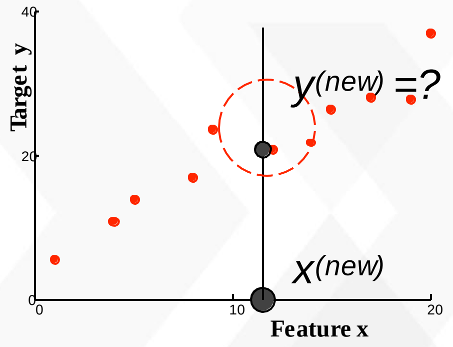
ANALYTIXLABS



Regression vs. KNN

Predictor
Given new features: Find nearest example Return its value

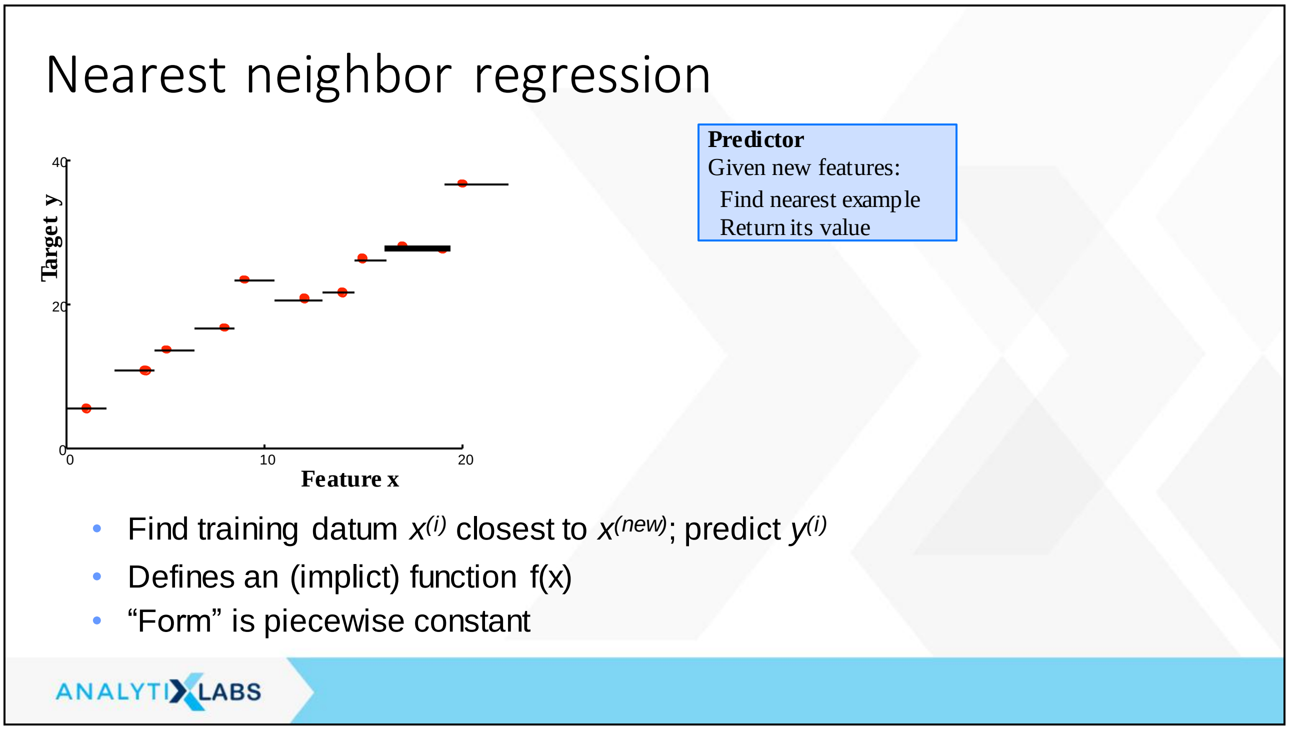




- Suggests a relationship between x and y
- Regression: given new observed $x^{(new)}$, estimate $y^{(new)}$

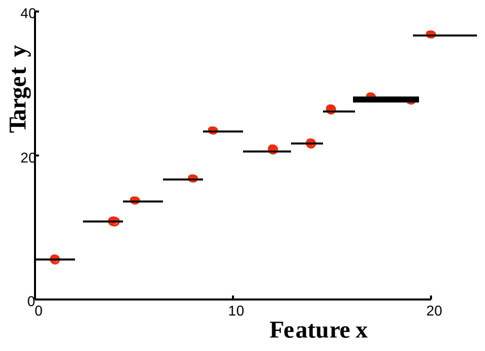
- Find training datum $x^{(i)}$ closest to $x^{(new)}$; predict $y^{(i)}$

ANALYTIXLABS



Nearest neighbor regression

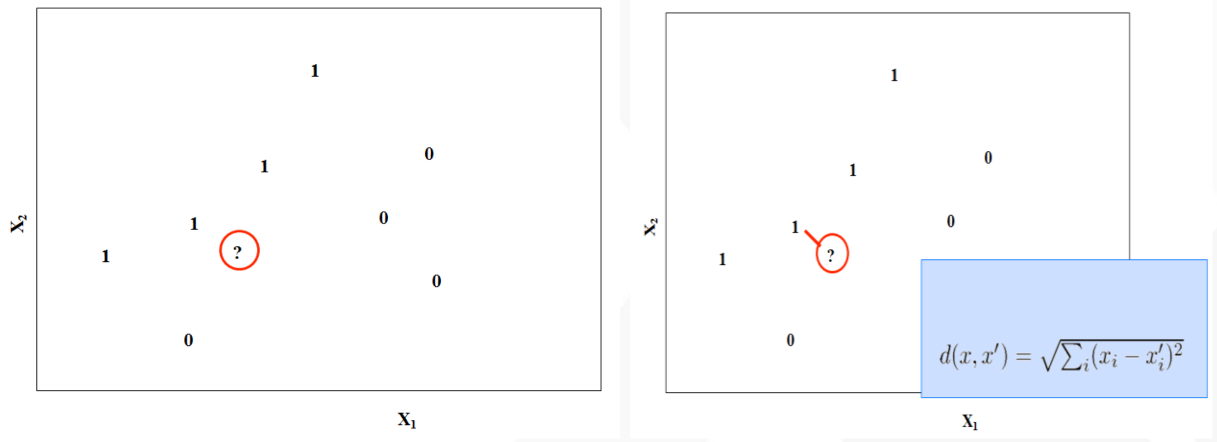
Predictor
Given new features:
Find nearest example
Return its value



- Find training datum $x^{(i)}$ closest to $x^{(new)}$; predict $y^{(i)}$
- Defines an (implict) function f(x)
- “Form” is piecewise constant

ANALYTIXLABS

Nearest neighbor classifier



ANALYTIXLABS

K-Nearest Neighbor (kNN) Classifier

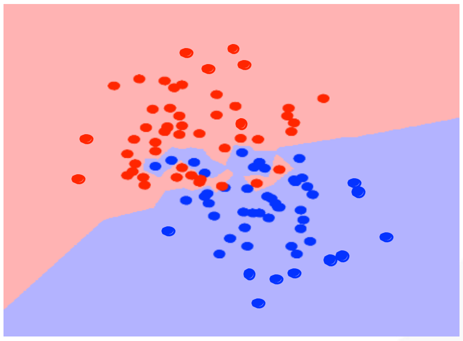
- Find the k-nearest neighbors to x̲ in the data
  - i.e., rank the feature vectors according to Euclidean distance
  - select the k vectors which are have smallest distance to x̲
- Regression
  - Usually just average the y-values of the k closest training examples
- Classification
  - ranking yields k feature vectors and a set of k class labels
  - pick the class label which is most common in this set (vote)
  - classify x̲ as belonging to this class
  - **Note: for two-class problems, if k is odd (k=1, 3, 5, …) there will never be any "ties"**
- Training is trivial: just use training data as a lookup table, and search to classify a new datum

ANALYTIXLABS

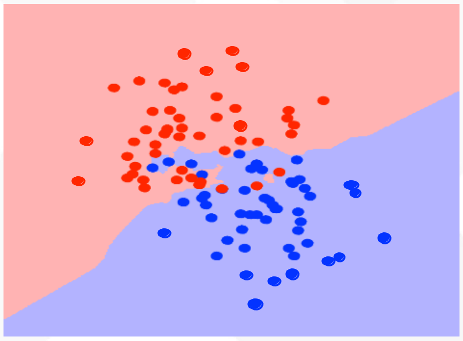
kNN Decision Boundary

- Piecewise linear decision boundary
- Increasing k simplifies decision boundary
  - Majority voting means less emphasis on individual points

K = 1



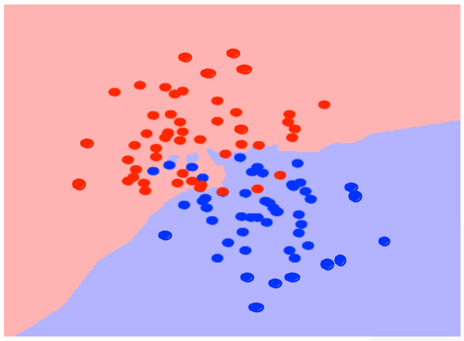
K = 3



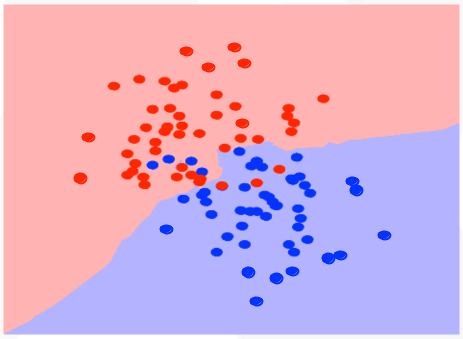
kNN Decision Boundary

- Recall: piecewise linear decision boundary
- Increasing k simplifies decision boundary
  - Majority voting means less emphasis on individual points

K = 5



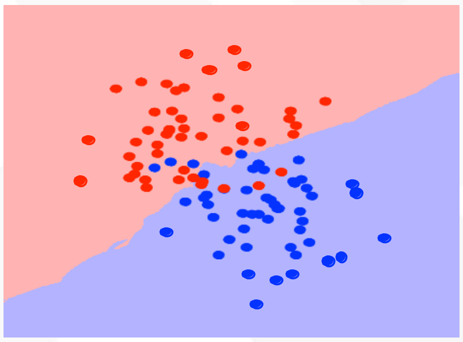
K = 7



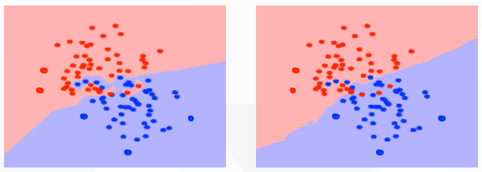
kNN Decision Boundary

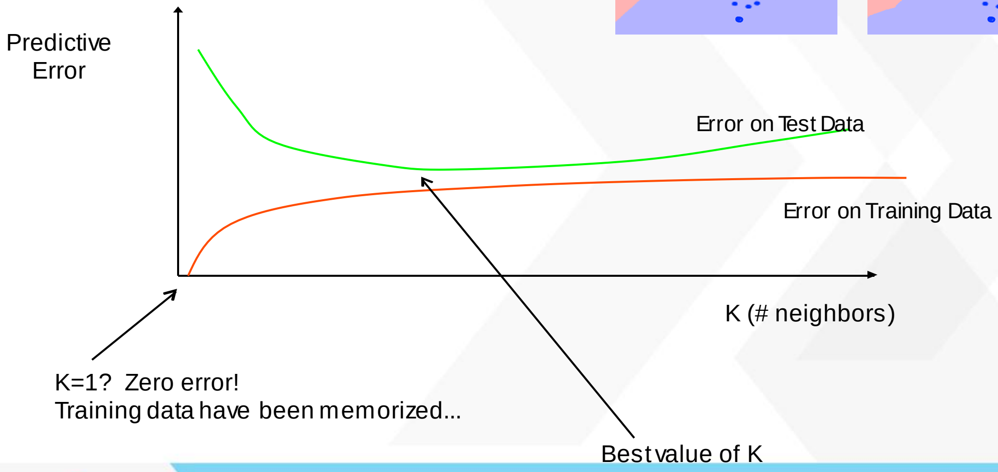
- Recall: piecewise linear decision boundary
  - Increasing k simplifies decision boundary
    - Majority voting means less emphasis on individual points

K = 25



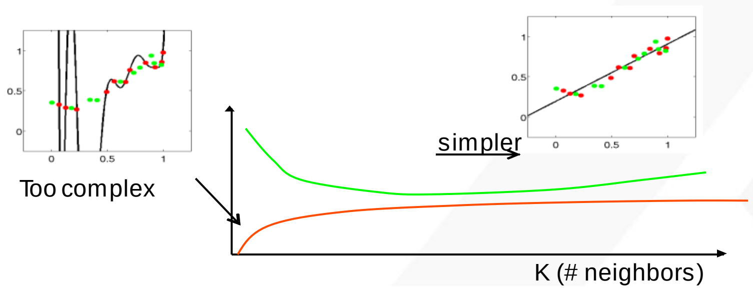
Error rates and K





Complexity & Overfitting

- Complex model predicts all training points well
- Doesn't generalize to new data points
- K=1 : perfect memorization of examples (complex)
- K=M : always predict majority class in dataset (simple)
- Can select K using validation data, etc.



K-Nearest Neighbor (kNN) Classifier

- Theoretical Considerations
  - as k increases
    - w e are averaging over more neighbors
    - the effective decision boundary is more smooth
  - as N increases, the optimal k value tends to increase
  - k=1, m increasing to infinity : error < 2x optimal

- Extensions of the Nearest Neighbor classifier
  - weighted distances
    - e.g., if some of the features are more important
    - e.g., if features are irrelevant

$$d(x, x') = \sqrt{\sum_i w_i (x_i - x'_i)^2}$$

  - fast search techniques (indexing) to find k-nearest neighbors in d-space

Pros and Cons of KNN

Pros:

- K-nearest neighbor algorithm is simple to understand and easy to implement.
- With zero to little training time
- KNN works just as easily with multiclass data sets whereas other algorithms are hardcoded for the binary setting.
- The non-parametric nature of KNN gives it an edge in certain settings where the data may be highly "unusual".

Cons:

- KNN algorithm is the computationally expensive testing phase which is impractical in industry settings. Note the rigid dichotomy between KNN and the more sophisticated Neural Network which has a lengthy training phase albeit a **very fast** testing phase.
- Furthermore, KNN can suffer from skewed class distributions. For example, if a certain class is very frequent in the training set, it will tend to dominate the majority voting of the new example (large number = more common).
- Finally, the accuracy of KNN can be severely degraded with high-dimension data because there is little difference between the nearest and farthest neighbor.

Improvements

There are many ways in which the KNN algorithm can be improved.

- A simple and effective way to remedy skewed class distributions is by implementing **weighed voting**. The class of each of the K neighbors is multiplied by a weight proportional to the inverse of the distance from that point to the given test point. This ensures that nearer neighbors contribute more to the final vote than the more distant ones.
- **Changing the distance metric** for different applications may help improve the accuracy of the algorithm. (i.e. Hamming distance for text classification)
- **Rescaling your data** makes the distance metric more meaningful. For instance, given 2 features height and weight, an observation such as x=[180,70]x=[180,70] will clearly skew the distance metric in favor of height. One way of fixing this is by column-wise subtracting the mean and dividing by the standard deviation. Scikit-learn's normalize() method can come in handy.
- **Dimensionality reduction** techniques like PCA should be executed prior to applying KNN and help make the distance metric more meaningful.
- **Approximate Nearest Neighbor** techniques such as using *k-d trees* to store the training observations can be leveraged to decrease testing time. Note however that these methods tend to perform poorly in high dimensions (20+). Try using **locality sensitive hashing (LHS)** for higher dimensions.