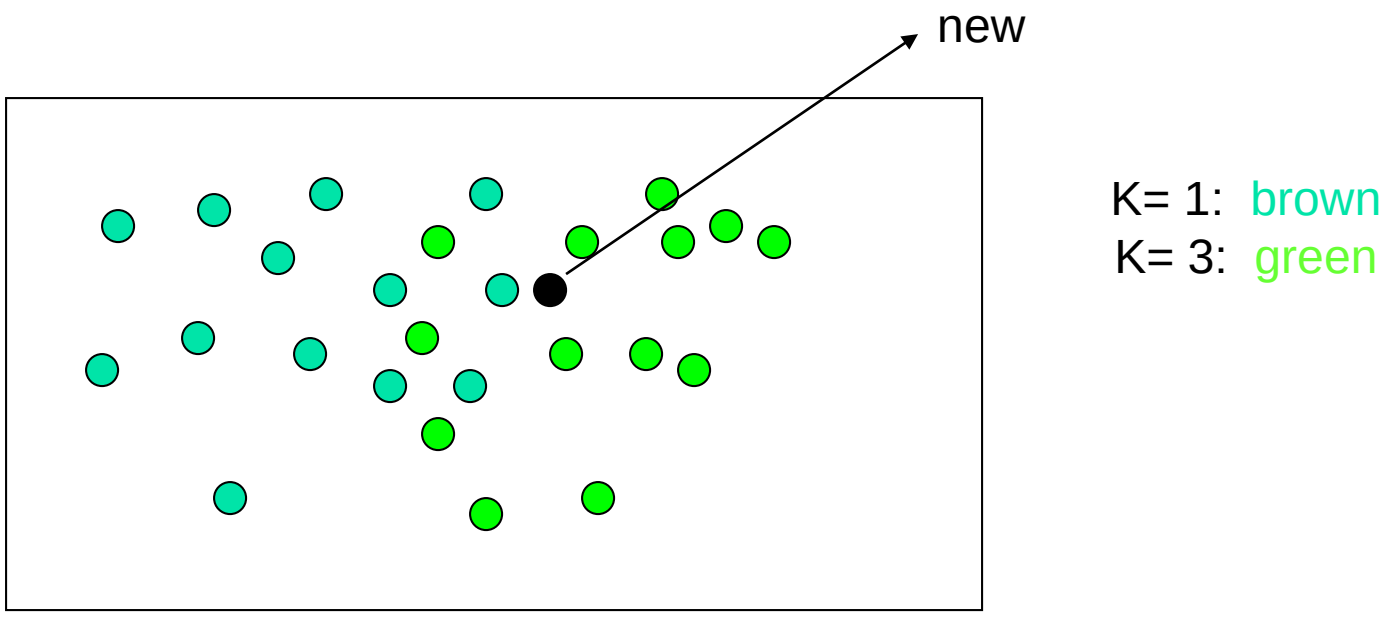
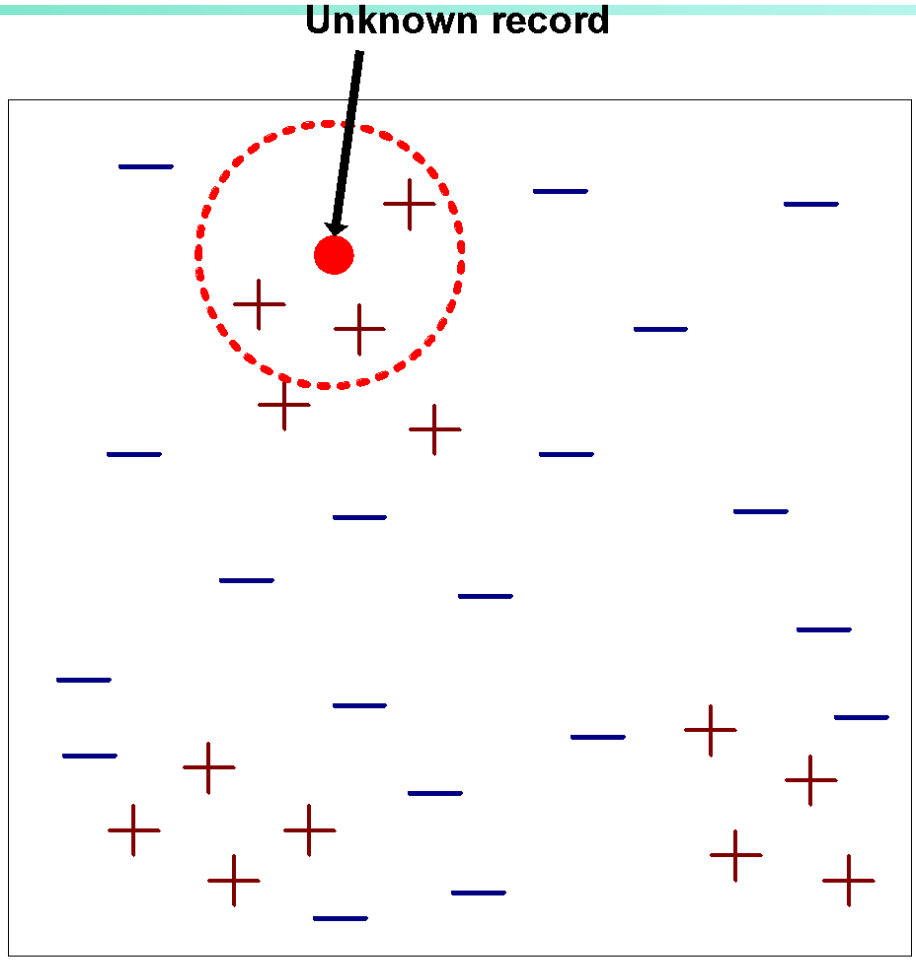
Nearest neighbor method

- Majority vote within the k nearest neighbors



K= 1: brown
K= 3: green

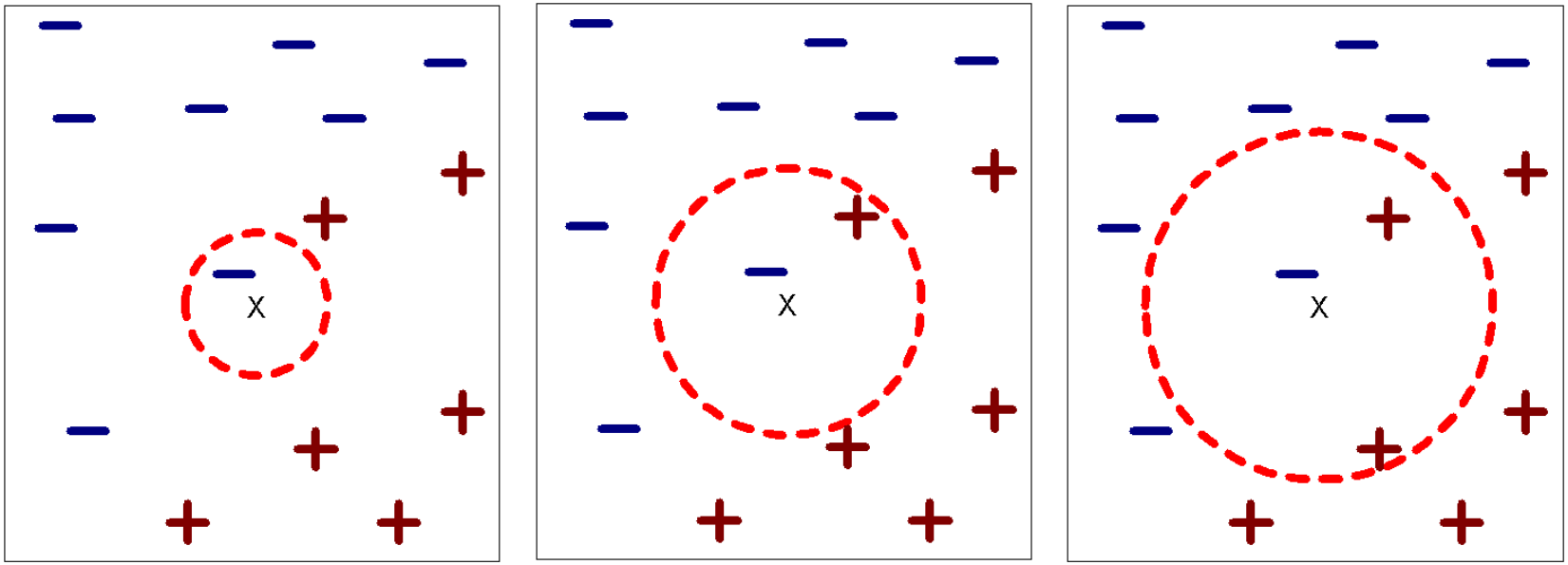
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

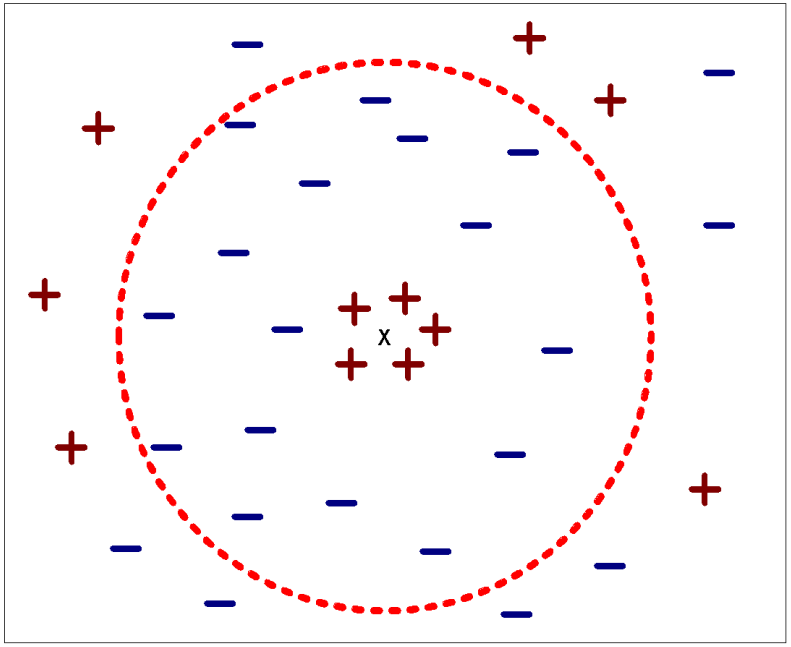


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

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

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation…

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

KNN

n_neighbors

int, optional (default = 5)

Number of neighbors to use by default

weights

optional (default = 'uniform')

weight function used in prediction. Possible values:

'uniform' : uniform weights. All points in each neighborhood are weighted equally.

'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

KNN

algorithm

optional Algorithm used to compute the nearest neighbors:

'ball_tree' will use :class:'BallTree'

'kd_tree' will use :class:'KDTree'

'brute' will use a brute-force search.

'auto' will attempt to decide the most appropriate algorithm based on the values passed to fit method.

KNN

leaf_size

int, optional (default = 30)

Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.

KNN

p

integer, optional (default = 2)

Power parameter for the Minkowski metric.

When p = 1, Manhattan_distance

When p = 2, Euclidean_distance

KNN

metric

String, default 'minkowski'

the distance metric to use for the tree. The default metric is minkowski, and with p=2 is equivalent to the standard Euclidean metric.

metric_params

dict, optional (default = None)

Additional keyword arguments for the metric function.

n_jobs

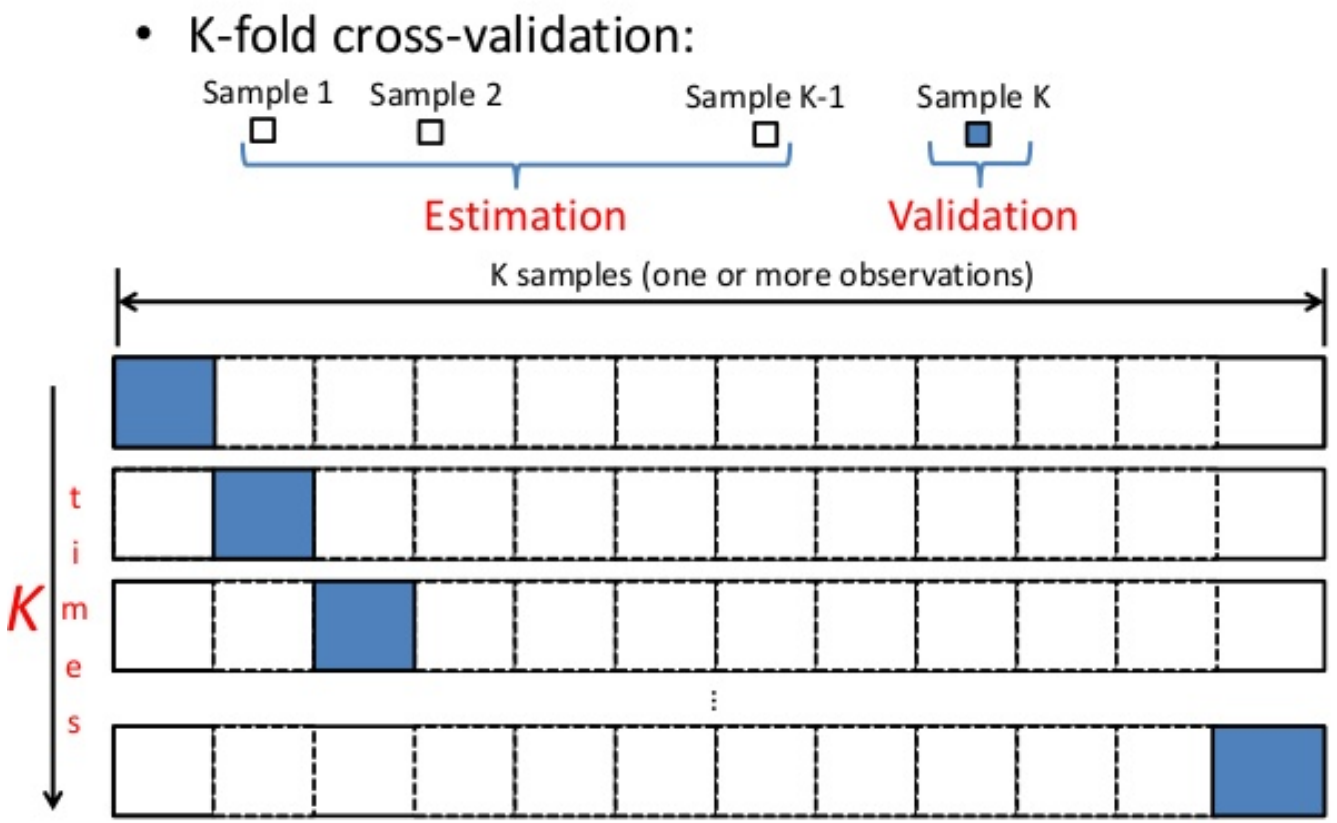
int, optional (default = 1)

The number of parallel jobs to run for neighbors search. If '-1', then the number of jobs is set to the number of CPU cores.

KNN

K is always selected as an odd number

Parameter Tuning with Cross Validation



Parameter Tuning with Cross Validation

k-fold cross validation (*the k is totally unrelated to the K in KNN*) involves randomly dividing the training set into k groups or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k-1$ folds.

The misclassification rate is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error which are then averaged out.

Pros and Cons of KNN

Pros

- one of the most attractive features of the K-nearest neighbor algorithm is that is simple to understand and easy to implement.
- KNN works just as easily with multiclass data sets whereas other algorithms are hardcoded for the binary setting.
- Finally, as we mentioned earlier, the non-parametric nature of KNN gives it an edge in certain settings where the data may be highly "unusual".

Cons

- One of the obvious drawbacks of the KNN algorithm is the computationally expensive testing phase which is impractical in industry settings.
- 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
- Finally, the accuracy of KNN can be severely degraded with high-dimension data because there is little difference between the nearest and farthest neighbor.