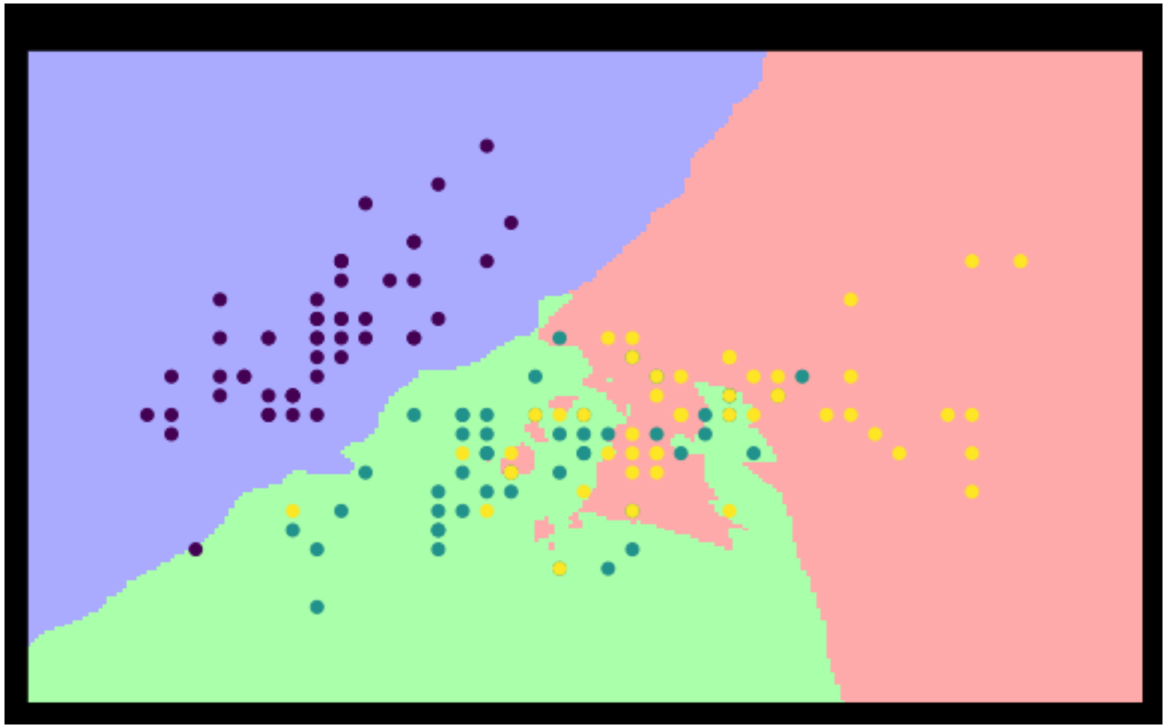
The Application and Theory of KNN Algorithm

Ma Xiaowen

2019/10/11

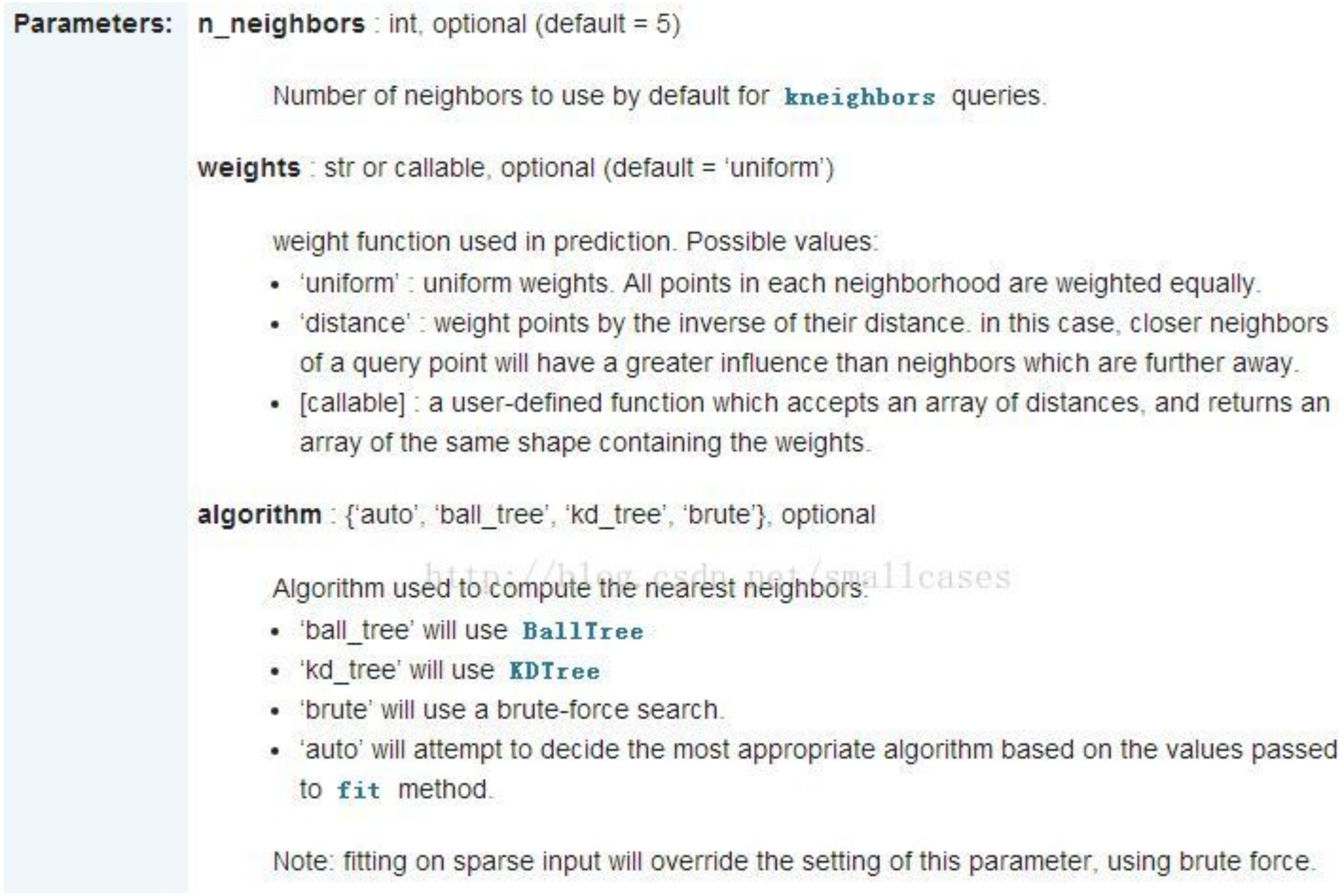
KNN(K-NearestNeighbor)

- For example:



KNN(K-NearestNeighbor)

KNeighborsClassifier*(*n_neighbors, weights, algorithm, leaf_size,p, **kwargs)



Parameters: **n_neighbors** : int, optional (default = 5)

Number of neighbors to use by default for `kneighbors` queries.

weights : str or callable, 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.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

algorithm : {'auto', 'ball_tree', 'kd_tree', 'brute'}, optional

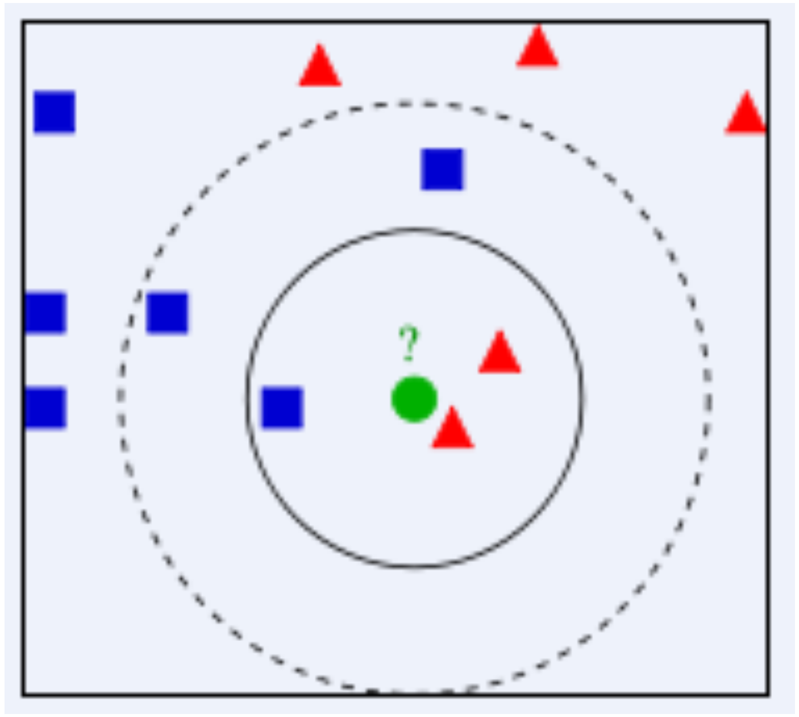
Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`
- 'kd_tree' will use `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.

Note: fitting on sparse input will override the setting of this parameter, using brute force.

KNN(K-NearestNeighbor)

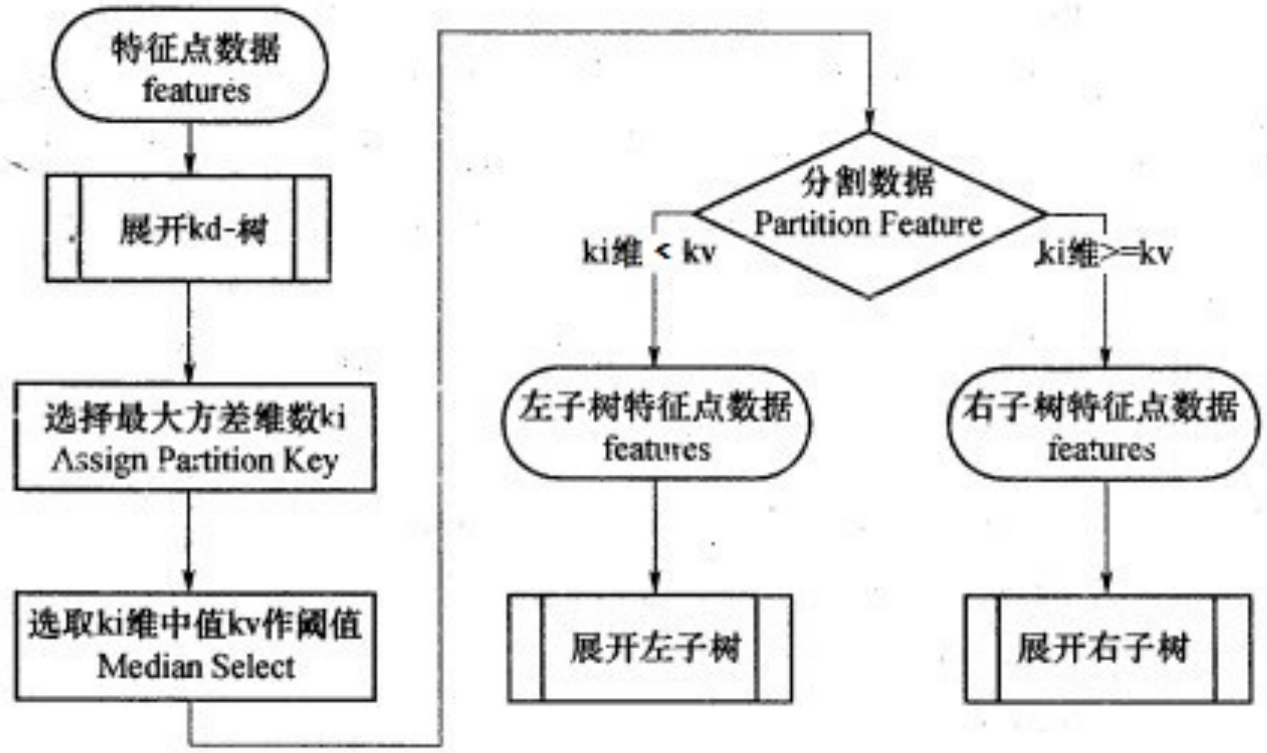
n_neighbors=3 or 5 ?



KNN(K-NearestNeighbor)

algorithm：{‘brute’, ‘kd_tree’, ‘ball_tree’}

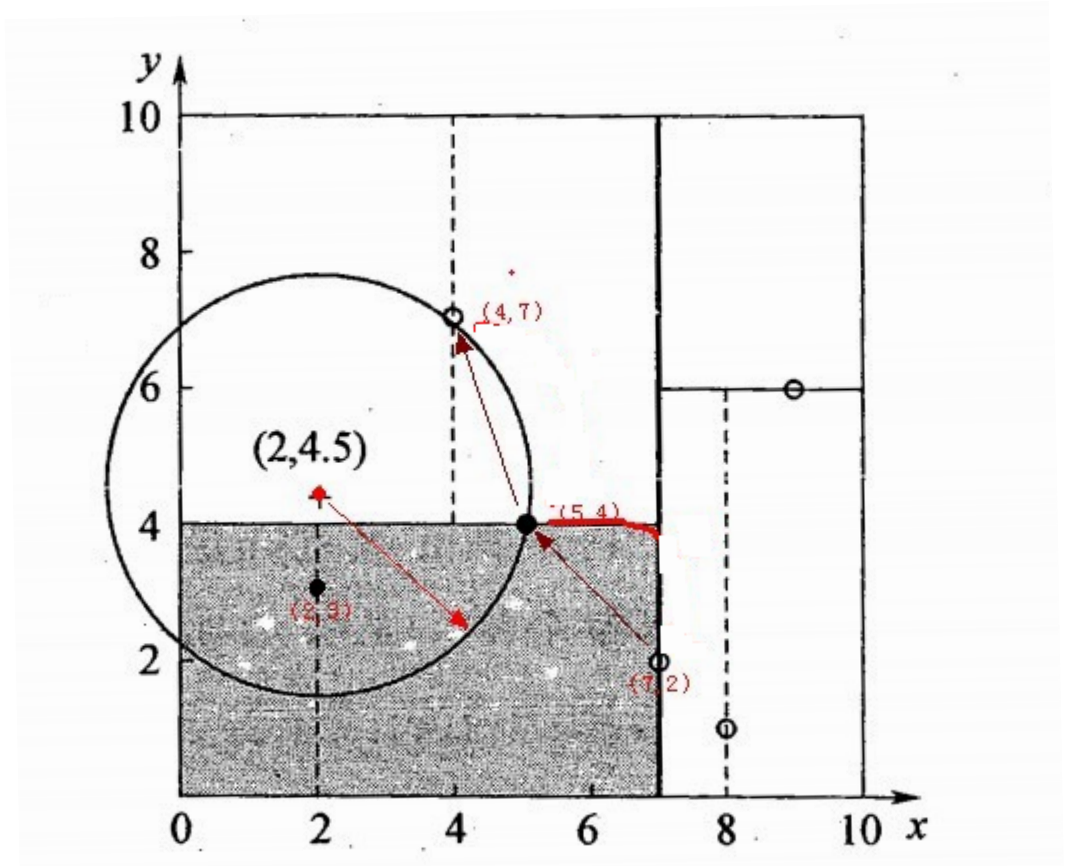
- ***kd_tree***



KNN(K-NearestNeighbor)

algorithm：{‘brute’, ‘kd_tree’, ‘ball_tree’}

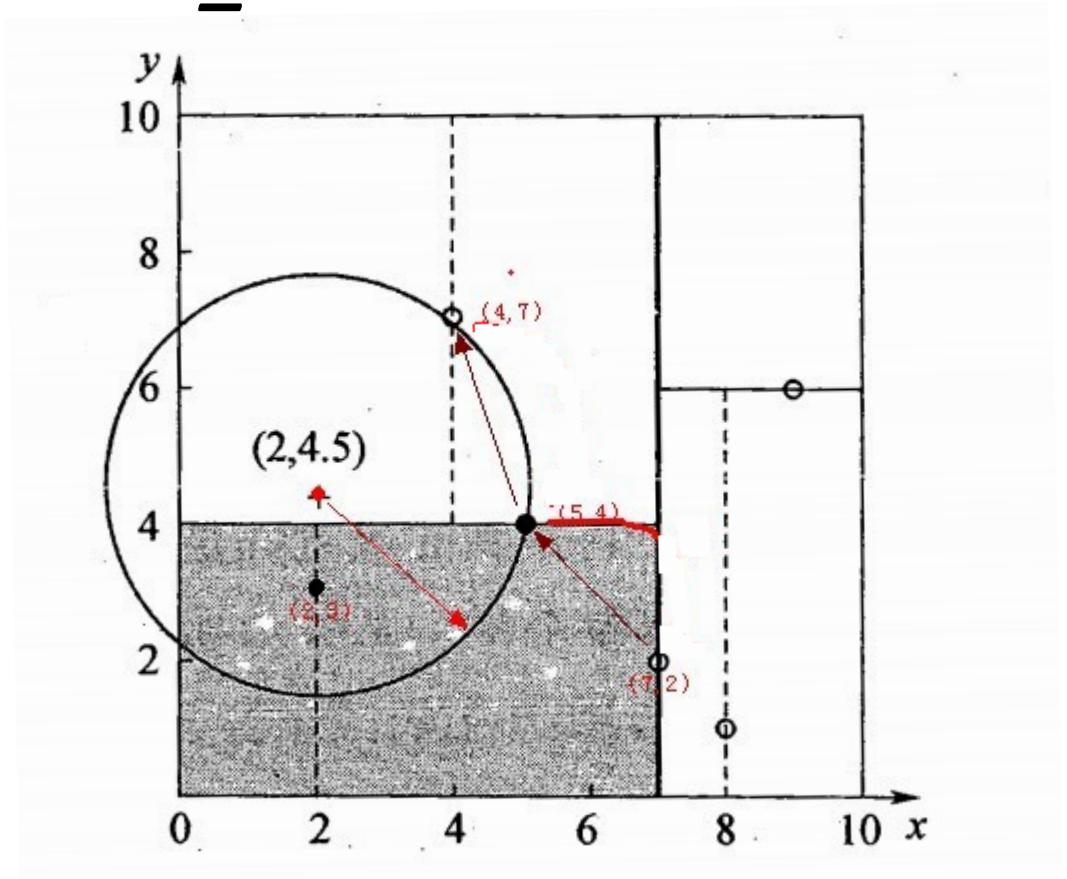
- ***kd_tree***



KNN(K-NearestNeighbor)

algorithm：{'brute', 'kd_tree', 'ball_tree'}

- ***kd_tree→ball_tree***



THANK YOU~