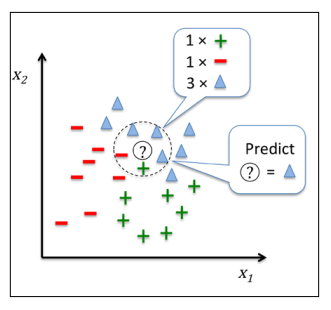
**K-nearest neighhors (KNN)**

The knn can be summarized by the following steps:

1. Choose the number of k and a distance metric
2. Find the k nearest neighbors of the sample that we want to classify.
3. Assign the class label by **majority vote**.

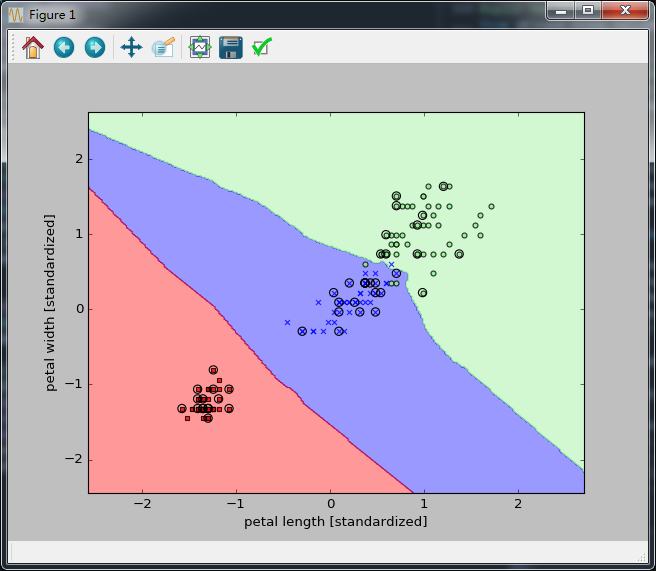


Advantages:

1. the classifer immediately adapts as we collect new training data.

Disadvantages:

1. computational complexity for classifying new samples grows linearly with the number of samples in the training dataset in the worst-case scenario—unless the dataset has very few dimensions (features) and the algorithm has been implemented using effcient data structures such as KD-trees
2. we can't discard training samples since no training step is involved. Thus, storage space can become a challenge if we are working with large datasets.



The right choice of k is crucial to find a good balance between over- and underftting. We also have to make sure that we choose a distance metric that is appropriate for the features in the dataset. Often, a simple Euclidean distance measure is used for real-valued samples, for example, the ﬂowers in our Iris dataset, which have features measured in centimeters. However, if we are using a Euclidean distance measure, it is also important to standardize the data so that each feature contributes equally to the distance. The 'minkowski' distance that we used in the previous code is just a generalization of the Euclidean and Manhattan distance that can be written as follows:

