Machine Learning K-Nearest Neighbours

Francesco Pugliese, PhD
neural1977@gmail.com

We will discuss about…

- ✓ kNN

Introduction

- ✓ Machine Learning algorithms can be divided in parametric and non-parametric models.
- ✓ By exploiting parametric models we are able to estimate model parameters from the training set in order to achieve a function capable to classify new data without requiring the original training set. Typical examples of parametric models are the perceptron, the logistic regression or the linear Support Vector Machine.
- ✓ On the other hand, non-parametric models cannot be characterized by a fixed number of parameters: the number of parameters increases with the number of training set data. So far, two of the examples of non-parametric models that we have seen are: Decision Tree and Random Forest. K-Nearest Neighbours belongs to non-parametric models which is known as Instance-based lLearning. Instance-based Learning models are able to record data from training set and they have a lazy learning.

kNN (k-Nearest Neighbours)

- ✓ Can be used to solve both classification and regression problems.
- ✓ Stores available inputs and classifies new inputs based on a similar measure i.e. the **distance** function.
- ✓ Has found its major application in **statistical estimation** and **pattern recognition**.
- ✓ **Steps for kNN**:
  - ✓ KNN works by finding the distances between a query and all inputs in the data.
  - ✓ Next, it selects a specified number of inputs, say K, closest to the query.
  - ✓ And then it votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

kNN (k-Nearest Neighbours)

- ✓ This Machine Learning algorithm is particularly interesting because is very different from the other algorithms we talked so far.
- ✓ KNN is a typical machine learning system which goes under the category of «lazy» algorithms. It is called lazy for its obvious simplicity, since it does not learn a discrimination function, but it «stores» the training set somehow.
- ✓
- ✓ KNN can be described as the following steps:

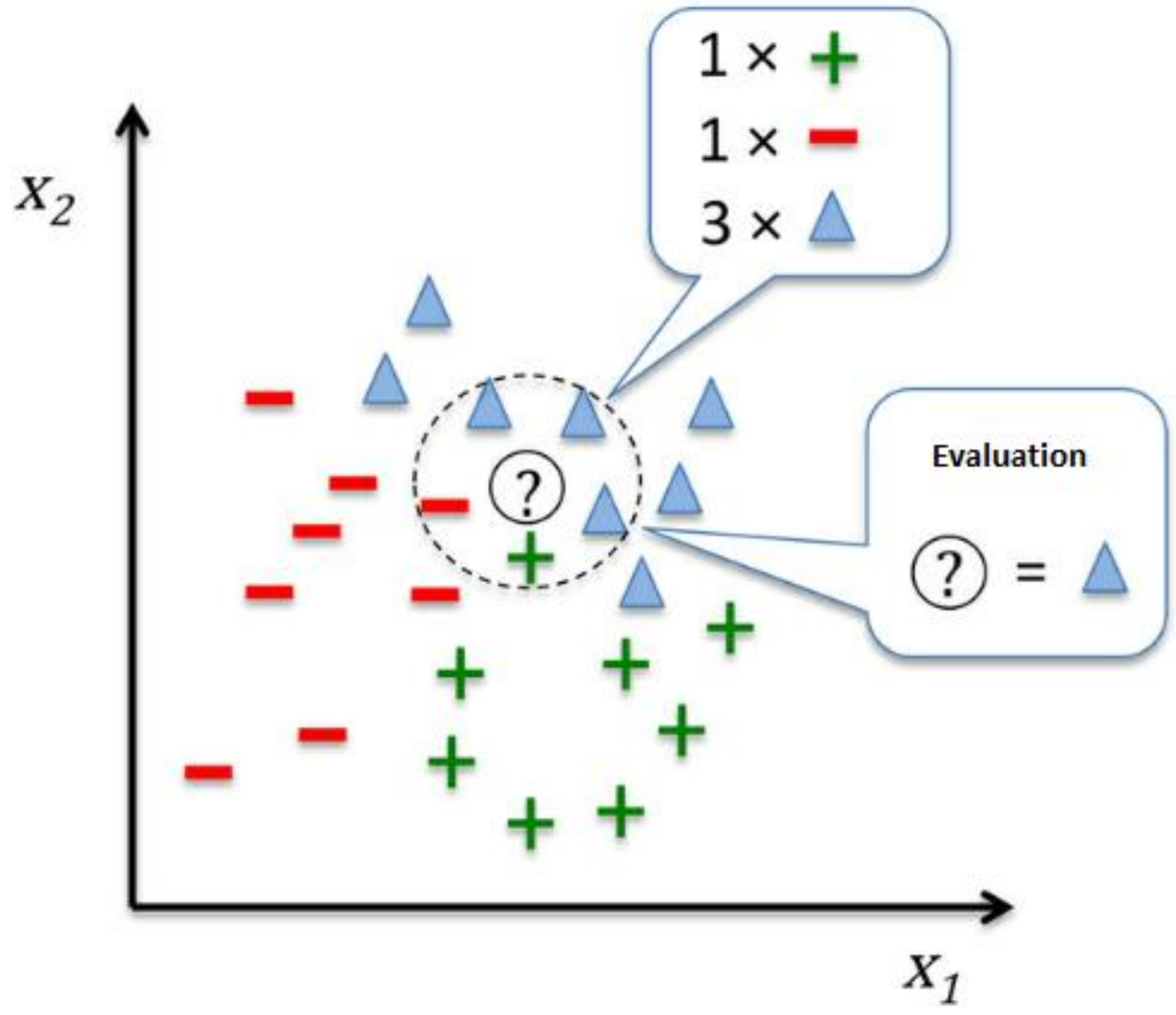
1. Choose the number k and a distance metric
2. Find the k elements nearest to the sample that we are willing to classify
3. Assign a class label according to a majority election.

In the following picture we see the way a new data dot is assigned to the triangles class according of a majority election among the 5 closest neigbours.

kNN Algorithm

- ✓ Load the data
- ✓ Initialize k to a chosen number of neighbours in the data
- ✓ For each example in the data, calculate the distance between the query example and the current input from the data
- ✓ Add that distance to the index of input to make an ordered collection
- ✓ Sort the ordered collection of distances and indices in ascending order grouped by distances
- ✓ Pick the first K entries from the sorted collection
- ✓ Get the labels of the selected K entries
- ✓ If regression, return the **mean** of the K labels; If classification, return the **mode** of the K labels

kNN (k-Nearest Neighbours)



kNN (k-Nearest Neighbours)

- ✓ According to the chosen distance metric, the KNN algorithm finds the k samples from the training set which are closer (more similar) to the point we intend to classify. The class label of the new point is determined by means of a majority election among the k-nearest neighbours.

- ✓ The main upside of this approach (based on the memorization) is the fact that classfier adapts immediately while we collect new training data.

- ✓ However, the downside is its computational complexity for the classification of new samples, the complexity linearly increases with the number of samples within the training set (in the worst case), unless the dataset is made of a low number of dimensions (features) and the algorithm is implemented with data structures called KD-trees.

- ✓ Moreover, we cannot remove training samples, since there is not a training step. Therefore, the memory space can become a problem if we have to address a huge dataset.

kNN (k-Nearest Neighbours)

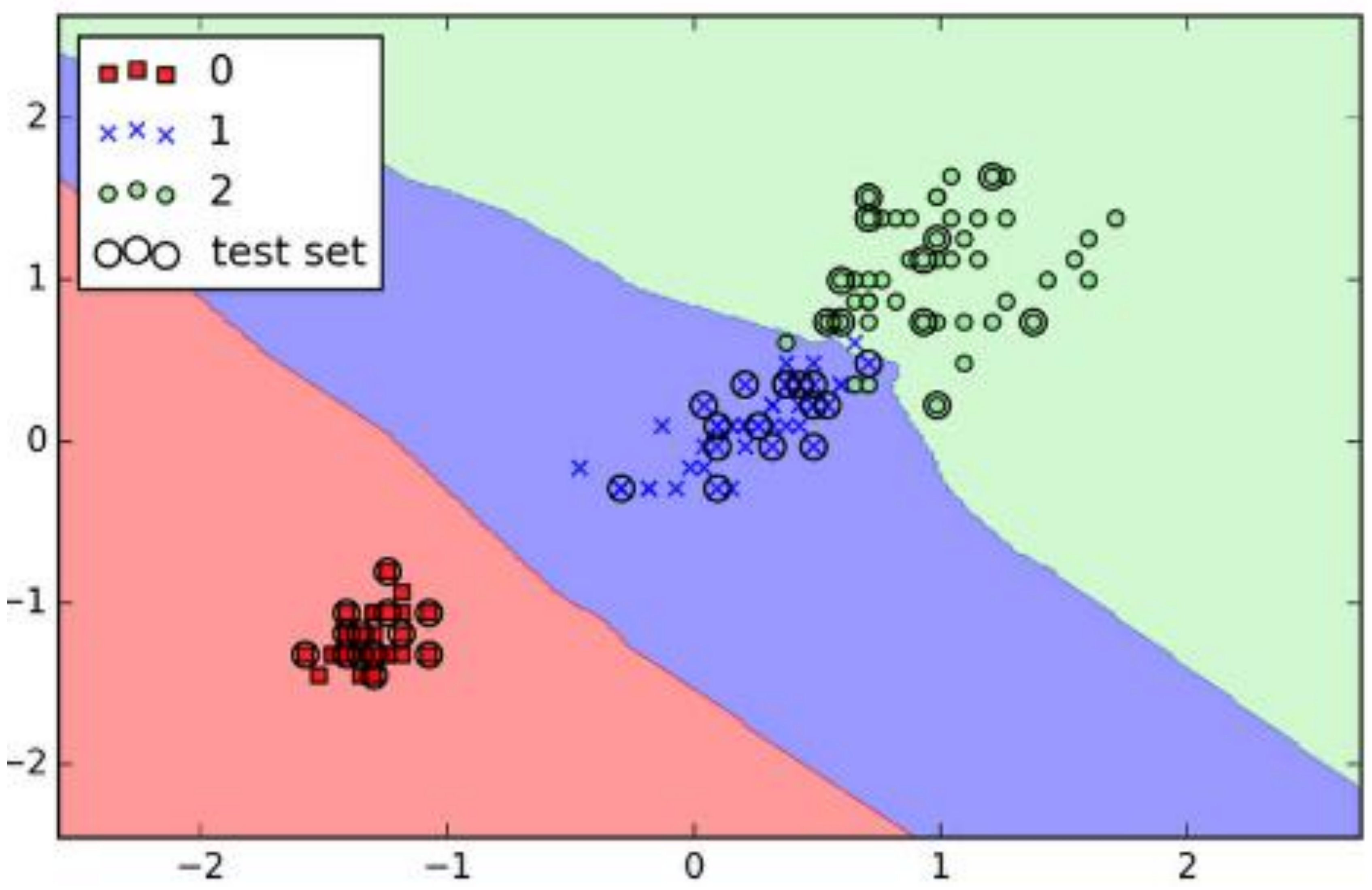
The following code implements a KNN model in scikit-learn exploiting an euclidean distance metric:

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski’)
knn.fit(X_train_std, y_train)
plot_decision_regions(X_combined_std, y_combined, classifier=knn, test_idx=range(105,150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.show()
```

Specifing the 5 closest neighbours in the model KNN for the Iris Datase, we achieve a soft decisonal as we can see in the following figure.

kNN (k-Nearest Neighbours)



kNN (k-Nearest Neighbours)

- ✓ In the case of one node, the scikit-learn implementation of the KNN algorithm will prefer the neighbours, which have a minor distance from the sample.
- ✓ If neighbours have a similar distance, the algorithm will choose the class label appearing as the first within the training set.
- ✓ The correct choice of k is critical to find a good balance between over-fitting and under-fitting issues.
- ✓ Moreover, we must be sure to select the right distance metric which is appropriate for the features of the dataset. Often, for rea world samples, for instance Iris dataset flowers, where features are measured in centimeters, we can use a simple «euclidean distance».
- ✓ However, if we adopt a measure of the euclidean distance, it is important to standardize data such as every feature affects uniformly to the distance. ‘Minkowski’ distance that we used in the previous code is simply a generalization of Euclidean and Manhattan distance.

$$d\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}\right) = \sqrt[p]{\sum_{k} \left| x_k^{(i)} - x_k^{(j)} \right|^p}$$

kNN (k-Nearest Neighbours)

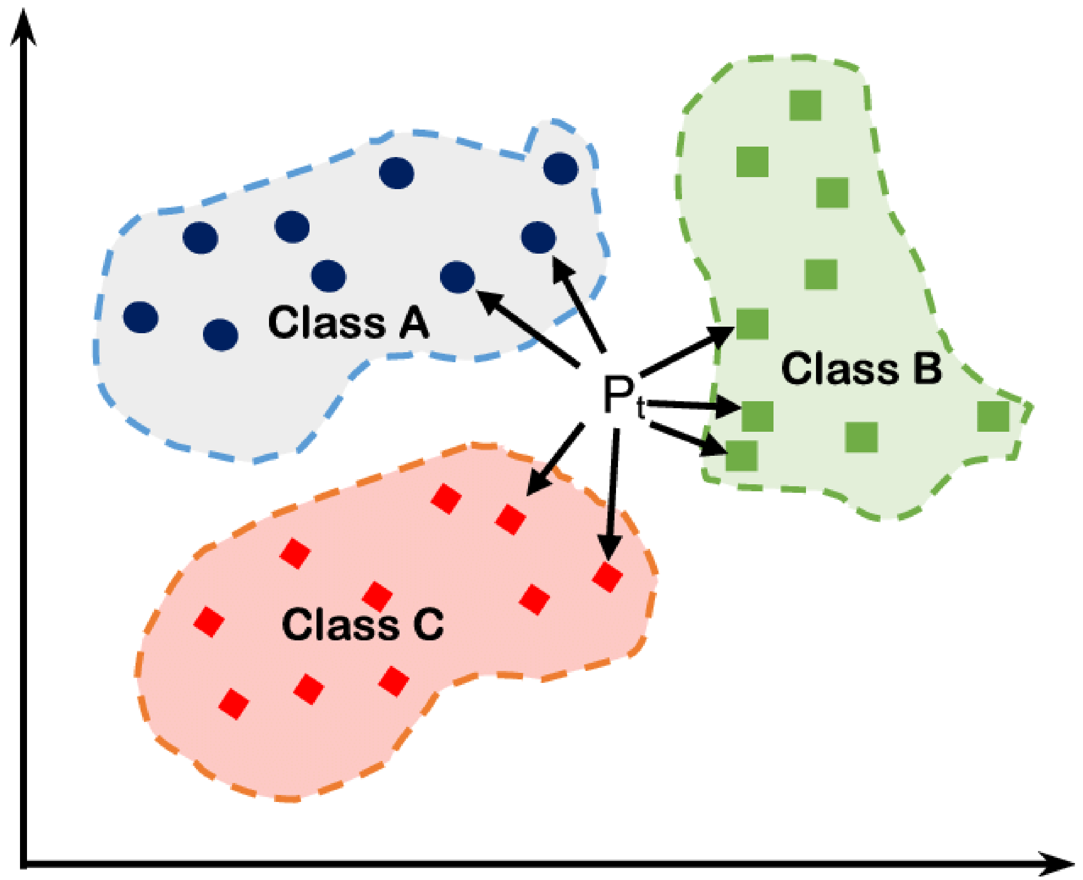
- ✓ This becomes an euclidean distance when the parameter p=2 or the missing distance when p=1.

- ✓ Many other distance evaluations are available on scikit-learn and can be used with the metric parameter. They are listed in: http://scikitlearn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html.

- ✓ It is very important to mention that KNN is affected by the overtting problem, because of dimensionality. It is an effect where the feature space becams more and more sparse due to the size of the fixed training set. Intuitively, we can consider the also nearest neighbours are too far in a space with a huge size, to provide a good estimation.

- ✓ Regularization is a technique for logistic regression to reduce overfitting, however in models with no regularization methods such as decision trees or KNN, we can use techniques to select some features and dimensionality reduction in order to escape from the issue of dimensionality.

kNN



Applications of kNN

- ✓ Fingerprint detection
- ✓ Forecasting stock market
- ✓ Currency exchange rate
- ✓ Bank bankruptencies
- ✓ Credit rating and Loan management
- ✓ Money laundering analyses
- ✓ Estimate the amount of glucose in the blood of a diabetic person from the IR absorption spectrum of that person’s blood.
- ✓ Identify the risk factors for a cancer based on clinical & demographic variables.

References

- ✓ https://towardsdatascience.com/top-10-algorithms-for-machine-learning-beginners-149374935f3c
- ✓ J. H. Friedman, J. L. Bentley e R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. "ACM Transactions on Mathematical Software (TOMS)", 3(3):209–226, 1977

Francesco **Pugliese**