LDSSA

SLU14 - k-Nearest Neighbours (kNN)

July 6th and 7th, 2019

1. Introduction

Motivation

- kNN is one of the simplest algorithms, yet very powerful!
- Can be used for Classification and Regression
- It is used a lot in **recommendation algorithms** (we'll see this later!)

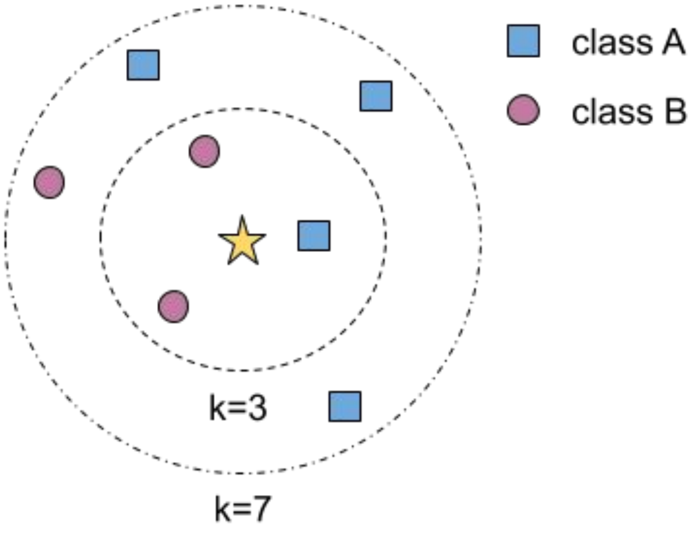
Overview

We'll cover the following topics:

- k-Nearest Neighbours Algorithm
- A Primer on Distance
- Some considerations about kNN
- Using kNN (sklearn)

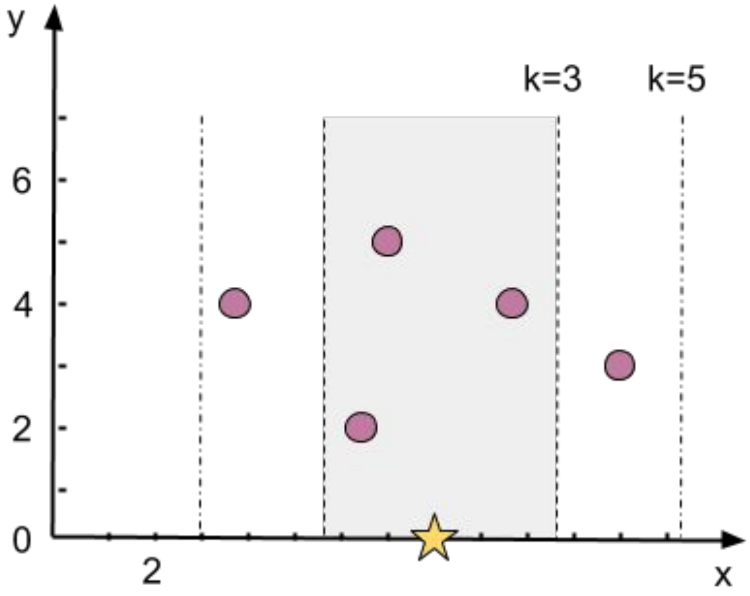
2. Topic Explanation

kNN intuition



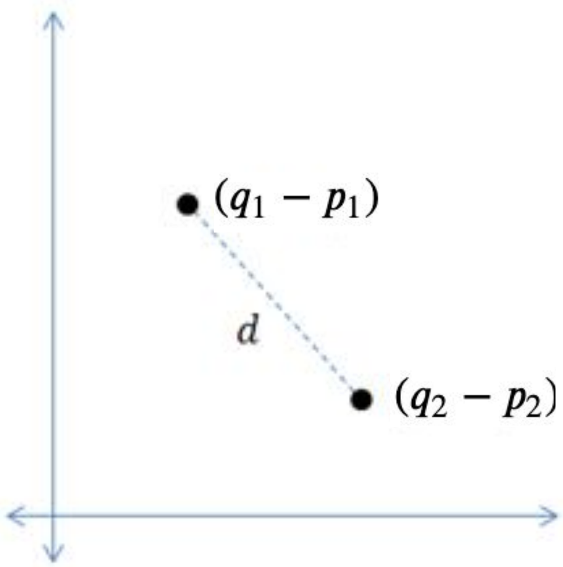
Classification with kNN

kNN intuition



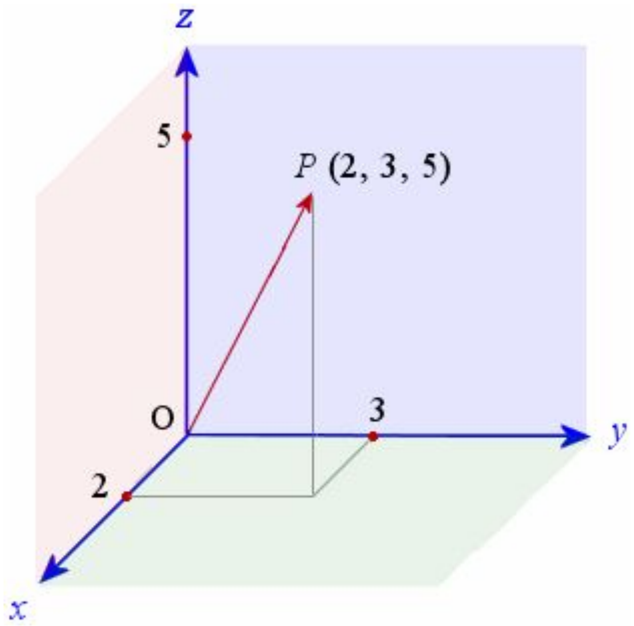
Regression with kNN

Euclidean distance



$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

Euclidean distance



$$d(\mathbf{O}, \mathbf{P}) = \sqrt{(o_1 - p_1)^2 + (o_2 - p_2)^2 + (o_3 - p_3)^2}$$

Euclidean distance - more dimensions

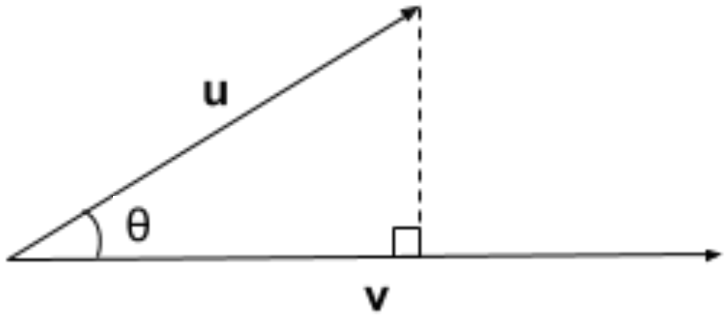
$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Dot product - v1

$$\mathbf{u} \,.\, \mathbf{v} = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n$$

Dot product - v2

$$\mathbf{u} \,.\, \mathbf{v} = |\mathbf{u}|\ |\mathbf{v}|\ cos(\theta)$$



Cosine distance

$$cos(\theta) = \frac{\mathbf{u}\ .\ \mathbf{v}}{|\mathbf{u}|\ |\mathbf{v}|}$$

$$cos\_dist(\mathbf{u}, \mathbf{v}) = 1 - cos(\mathbf{u}, \mathbf{v})$$

Recap

- **Euclidean distance** measures a physical distance
- **Dot product** measures how much two vectors point in the same direction, weighted by the vectors' norms
- **Cosine distance** measures how much two vectors point in the same direction

Considerations about kNN

kNN is non parametric

- No a priori assumptions on the model structure
- The model is determined from the data
- The more data, the better!

Considerations about kNN

No learning!

- No need to learn parameters
- Just does comparisons
- Lazy method: does nothing until prediction phase

Considerations about kNN

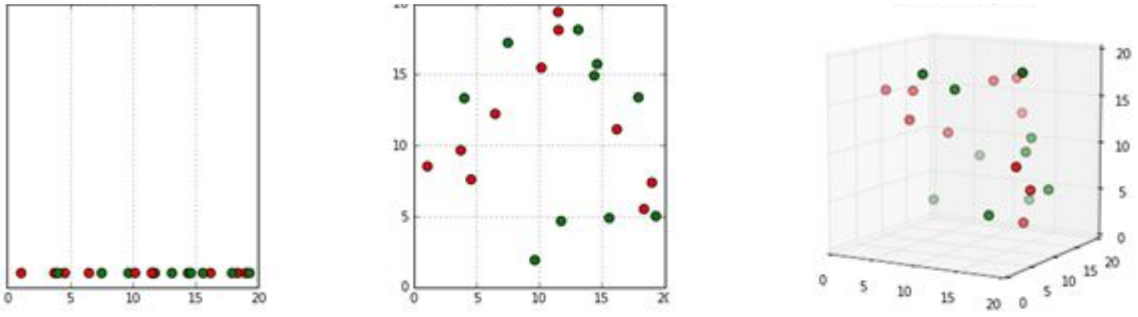
k needs to be tuned

- Higher k has less noise
- Too large k leads to bad predictions!
- We’ll see later how to do this systematically

Considerations about kNN

Curse of dimensionality

- The more dimensions, the more sparse the space gets
- Points don’t have close neighbours anymore
- Assumption that close points are similar doesn’t stand anymore



Considerations about kNN

Slow with large dataset

- Computing distances may be an expensive operation
- 1 prediction = computing as many distances as points in the dataset
- Approximate nearest neighbour search to the rescue

Using kNN in sklearn

```
from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier(n_neighbors=5)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
from sklearn.neighbors import KNeighborsRegressor

reg = KNeighborsRegressor()
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
```

3. Recap

Recap

- kNN is simple!
- Can be used for classification, regression, and others!
- We need to choose the appropriate distance function
- The vanilla version has some caveats (slow in big datasets, and curse of dimensionality)
- The basis of many algorithms used in practice