Exercise Collection – k-NN

Contents

Lecture exercises

Exercise 1: k-NN with Manhattan Distance

a) Let the two-dimensional feature vectors in the following figure be instances of two different classes (triangles and circles). Classify the point (7, 6) – represented by a square in the picture – with a k-NN classifier using $L1$ norm (Manhattan distance):

$$d_{\text{Manhattan}}(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{j=1}^{p} |x_j - \tilde{x}_j|.$$

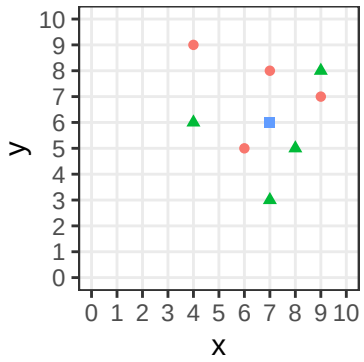
As a decision rule, use the unweighted number of the individual classes in the k-neighborhood, i.e., assign the point to the class that represents most neighbors.

i) $k = 3$

ii) $k = 5$

iii) $k = 7$

```
## Warning:  `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



b) Now consider the same constellation but assume a regression problem this time, where the circle-shaped points have a target value of 2 and the triangles have a value of 4.

Again, predict for the square point (7, 9), using both the *unweighted* and the *weighted* mean in the neighborhood (still with Manhattan distance).

i) $k = 3$

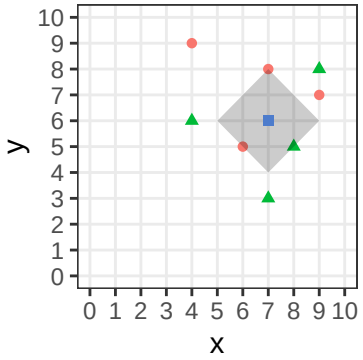
ii) $k = 5$

iii) $k = 7$

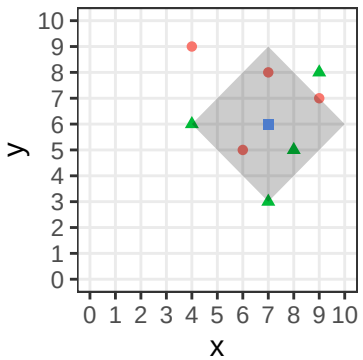
Solution 1:

a) k-NN classification

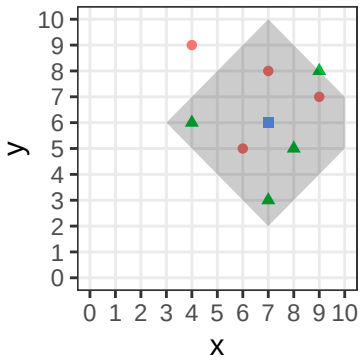
i) $k = 3 \implies$ 2 circles and 1 triangle, so we predict "circle".



b) $k = 5 \implies$ 3 circles and 3 triangles, we have to specify beforehand what to do in case of a tie.



c) $k = 7 \implies$ 3 circles and 4 triangles, so we predict "triangle".



b) k-NN regression

We now consider both unweighted and weighted predictions. Recall that weights are computed based on the distance between the point of interest and its respective neighbors. With the Manhattan, or "city block" metric, the distance can be read from the plot by walking along the grid lines (shortest way). For example, in the 3-neighborhood, all points have a distance of 2 from our square, so all get weights $\frac{1}{2}$.

i) $k = 3$

$$\hat{y} = \frac{2+2+4}{3} = \frac{8}{3} \approx 2.67$$
$$\hat{y}_{\text{weighted}} = \frac{\frac{1}{2}\cdot 2 + \frac{1}{2}\cdot 2 + \frac{1}{2}\cdot 4}{\frac{3}{2}} = \frac{8}{3} \approx 2.67$$

ii) $k = 5$

$$\hat{y} = \frac{3\cdot 2 + 3\cdot 4}{6} = 3$$
$$\hat{y}_{\text{weighted}} = \frac{\frac{1}{2}\cdot 2 + \frac{1}{2}\cdot 2 + \frac{1}{3}\cdot 2 + \frac{1}{2}\cdot 4 + \frac{1}{3}\cdot 4 + \frac{1}{3}\cdot 4}{\frac{5}{2}} = \frac{44}{15} \approx 2.93$$

iii) $k = 7$

$$\hat{y} = \frac{3\cdot 2 + 4\cdot 4}{7} = \frac{22}{7} \approx 3.14$$
$$\hat{y}_{\text{weighted}} = \frac{\frac{1}{2}\cdot 2 + \frac{1}{2}\cdot 2 + \frac{1}{3}\cdot 2 + \frac{1}{2}\cdot 4 + \frac{1}{3}\cdot 4 + \frac{1}{3}\cdot 4 + \frac{1}{4}\cdot 4}{\frac{11}{4}} = \frac{100}{33} \approx 3.03$$

Exercise 2: k-NN from Scratch

Implement a simple version of a K-nearest-neighbour classifier.

```
myknn = function(target, traindata, testdata, k)
```

The function should return a factor of predicted classes from `testdata`. `target` is the name of the target variable in both `data.frames`. Some hints:

- Your function only needs to work for numeric features.
- Use the euclidean distance.
- Do not overengineer your solution. Keep it simple and do not think too much about efficiency.

Test your implementation on the `iris` data set for $k = 1, 2, 7$. Split your data set 10 times in $\frac{2}{3}$ training and $\frac{1}{3}$ test data. Measure training and test error in each split.

Solution 2:
See R file

Further exercises

Ideas & exercises from other sources