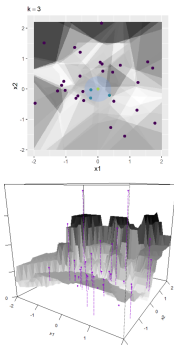


Introduction to Machine Learning

k -Nearest Neighbors

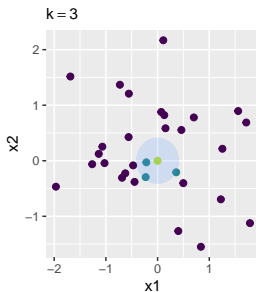
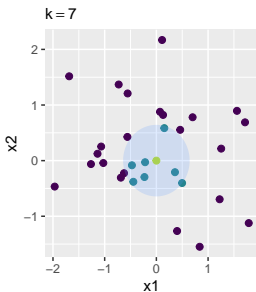
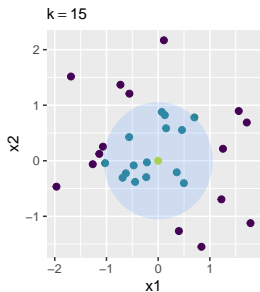


Learning goals

- Understand the basic idea of k -NN for regression and classification
- Understand that k -NN is a non-parametric, local model
- Know different distance measures for different scales of feature variables

K-NEAREST-NEIGHBORS

- k -**NN** can be used for regression and classification.
- Generates "similar" predictions for \mathbf{x} as its k closest neighbors.
- "Closeness" requires a distance or similarity measure.
- The set containing the k closest points $\mathbf{x}^{(i)}$ to \mathbf{x} in the training data is called the k -**neighborhood** $N_k(\mathbf{x})$ of \mathbf{x} .

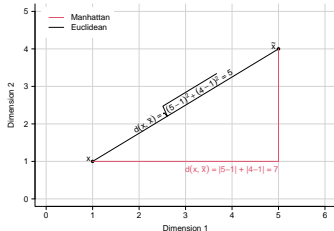


DISTANCE MEASURES

How to calculate distances?

- Most popular distance measure for numerical features:
Euclidean distance
- For two data points \mathbf{x} and $\tilde{\mathbf{x}}$ with p numeric features

- $d_{Euclidean}(\mathbf{x}, \tilde{\mathbf{x}}) = \sqrt{\sum_{j=1}^p (x_j - \tilde{x}_j)^2}$
- $d_{manhattan}(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{j=1}^p |x_j - \tilde{x}_j|$

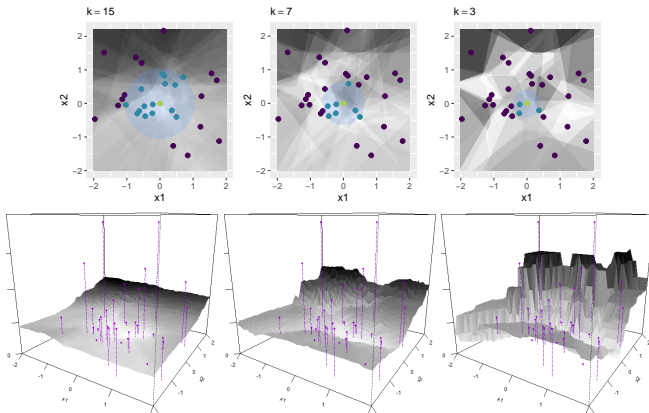


PREDICTION - REGRESSION

Compute for each point the average output y of the k -nearest neighbours in $N_k(\mathbf{x})$:

$$\hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{i:\mathbf{x}^{(i)} \in N_k(\mathbf{x})} y^{(i)} \text{ or } \hat{f}(\mathbf{x}) = \frac{1}{\sum_{i:\mathbf{x}^{(i)} \in N_k(\mathbf{x})} w_i} \sum_{i:\mathbf{x}^{(i)} \in N_k(\mathbf{x})} w_i y^{(i)}$$

with neighbors weighted based on their distance to \mathbf{x} : $w_i = \frac{1}{d(\mathbf{x}^{(i)}, \mathbf{x})}$



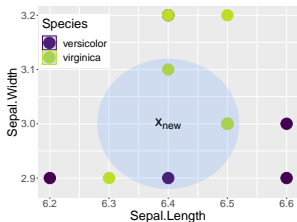
PREDICTION - CLASSIFICATION

For classification in g groups, a majority vote is used:

$$\hat{h}(\mathbf{x}) = \arg \max_{\ell \in \{1, \dots, g\}} \sum_{i: \mathbf{x}^{(i)} \in N_k(\mathbf{x})} \mathbb{I}(y^{(i)} = \ell)$$

And posterior probabilities can be estimated with:

$$\hat{\pi}_{\ell}(\mathbf{x}) = \frac{1}{k} \sum_{i: \mathbf{x}^{(i)} \in N_k(\mathbf{x})} \mathbb{I}(y^{(i)} = \ell)$$



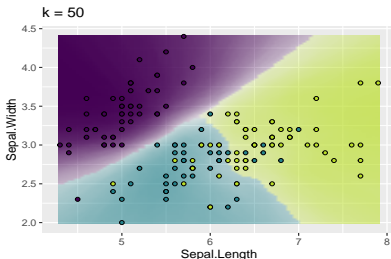
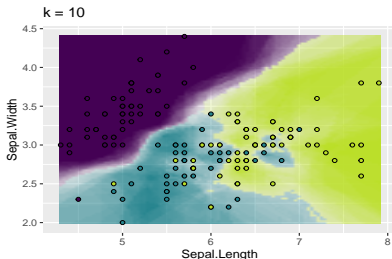
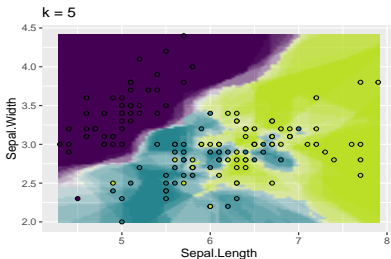
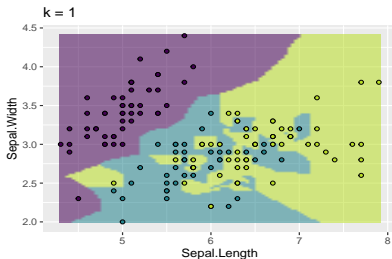
	SL	SW	Species	dist
52	6.4	3.2	versicolor	0.200
59	6.6	2.9	versicolor	0.224
75	6.4	2.9	versicolor	0.100
76	6.6	3.0	versicolor	0.200
98	6.2	2.9	versicolor	0.224
104	6.3	2.9	virginica	0.141
105	6.5	3.0	virginica	0.100
111	6.5	3.2	virginica	0.224
116	6.4	3.2	virginica	0.200
117	6.5	3.0	virginica	0.100
138	6.4	3.1	virginica	0.100
148	6.5	3.0	virginica	0.100

Example with subset of iris data ($k = 3$)

$$\hat{\pi}_{setosa}(\mathbf{x}_{new}) = \frac{0}{3} = 0\%, \quad \hat{\pi}_{versicolor}(\mathbf{x}_{new}) = \frac{1}{3} = 33\%, \quad \hat{\pi}_{virginica}(\mathbf{x}_{new}) = \frac{2}{3} = 67\%,$$

$$\hat{h}(\mathbf{x}_{new}) = virginica$$

K-NN: FROM SMALL TO LARGE K



Complex, local model vs smoother, more global model

K-NN SUMMARY

- k -NN is a lazy classifier, it has no real training step, it simply stores the complete data - which are needed during prediction.
- Hence, its parameters are the training data, there is no real compression of information.
- As the number of parameters grows with the number of training points, we call k -NN a non-parametric model
- k -NN is not based on any distributional or functional assumption, and can, in theory, model data situations of arbitrary complexity.
- The smaller k , the less stable, less smooth and more “wiggly” the decision boundary becomes.
- Accuracy of k -NN can be severely degraded by the presence of noisy or irrelevant features, or when the feature scales are not consistent with their importance.

STANDARDIZATION AND WEIGHTS

- **Standardization:** Features in k-NN are usually standardized or normalized. If two features have values on a very different range, most distances would place a higher importance on the one with a larger range, leading to an imbalanced influence of that feature.
- **Importance:** Sometimes one feature has a higher importance (maybe we know this via domain knowledge). It can now manually be upweighted to reflect this.

$$d_{Euclidean}^{weighted}(\mathbf{x}, \tilde{\mathbf{x}}) = \sqrt{\sum_{j=1}^p w_j (x_j - \tilde{x}_j)^2}$$

- If these weights would have to be learned in a data-driven manner, we could only do this by hyperparameter tuning in k-NN. This is inconvenient, and Gaussian processes handle this much better.

GOWER DISTANCE

- A weighted mean of univ. distances in the j -th feature.
- It can handle categoricals, missings, and different ranges.

$$d_{gower}(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{\sum_{j=1}^p \delta_{x_j, \tilde{x}_j} \cdot d_{gower}(x_j, \tilde{x}_j)}{\sum_{j=1}^p \delta_{x_j, \tilde{x}_j}}.$$

- $\delta_{x_j, \tilde{x}_j}$ is 0 or 1. It's 0 if j -th feature is *missing* in at least one observation, or when the feature is asymmetric binary (where “1” is more important than “0”) and both values are zero. Otherwise 1.
- $d_{gower}(x_j, \tilde{x}_j)$: For nominals it's 0 if both values are equal and 1 otherwise. For integers and reals, it's the absolute difference, divided by range.

GOWER DISTANCE

Example of Gower distance with data on sex and income:

index	sex	salary
1	m	2340
2	w	2100
3	NA	2680

$$d_{gower}(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{\sum_{j=1}^p \delta_{x_j, \tilde{x}_j} \cdot d_{gower}(x_j, \tilde{x}_j)}{\sum_{j=1}^p \delta_{x_j, \tilde{x}_j}}$$

$$d_{gower}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \frac{1 \cdot 1 + 1 \cdot \frac{|2340 - 2100|}{|2680 - 2100|}}{1 + 1} = \frac{1 + \frac{240}{580}}{2} = \frac{1 + 0.414}{2} = 0.707$$

$$d_{gower}(\mathbf{x}^{(1)}, \mathbf{x}^{(3)}) = \frac{0 \cdot 1 + 1 \cdot \frac{|2340 - 2680|}{|2680 - 2100|}}{0 + 1} = \frac{0 + \frac{340}{580}}{1} = \frac{0 + 0.586}{1} = 0.586$$

$$d_{gower}(\mathbf{x}^{(2)}, \mathbf{x}^{(3)}) = \frac{0 \cdot 1 + 1 \cdot \frac{|2100 - 2680|}{|2680 - 2100|}}{0 + 1} = \frac{0 + \frac{580}{580}}{1} = \frac{0 + 1.000}{1} = 1$$