MEME19903/MECG11103 Predictive Modelling Topic 2b: Supervised Learning: kNN

Dr Liew How Hui

May 2022

Outline

Methods of Classification

- Regression models from statistics
- kNN models
- Naive Bayes models
- Tree-based models

kNN Model

The *k-nearest neighbours* (kNN) algorithm is a non-parametric method used for classification and regression.

Unlike the logistic regression model which 'computes' the coefficients (which are fixed) based on the 'training data'. The kNN model keeps the training data (which becomes the internal parameters which vary in size) and use the "**similar inputs have similar outputs**" assumption to tag the new point a label based on the labels of the k neighbours (k points in the training data which are most similar to the new point).

kNN Model (cont)

Given a positive integer k and an input $\boldsymbol{x}$, the kNN algorithm first identifies the k points in the training data $(\boldsymbol{x}_i, y_i)$ that are "closest" to $\boldsymbol{x}$, represented by $N(\boldsymbol{x})$.

- For kNN classifier, the prediction is

$$h(\boldsymbol{x}) = \text{mode}(\{y'' : (\mathbf{x}'', y'') \in N(\boldsymbol{x})\}),$$

$$\mathbb{P}(Y = j|\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{x}_i \in N(\boldsymbol{x})} I(y_i = j).$$

- For kNN regressor, the prediction is

$$h(\boldsymbol{x}) = \frac{1}{k} \sum_{(\boldsymbol{x}'', y'') \in N(\boldsymbol{x})} y''.$$

kNN Model (cont)

The kNN algorithm fundamentally relies on a "distance" metric.

What is a "distance"?

A *distance* $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a function which satisfies the following conditions: For any $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_k$,

(i) $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$ and $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0$ iff $\boldsymbol{x}_i = \boldsymbol{x}_j$;

(ii) $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$; and

(iii) $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq d(\boldsymbol{x}_i, \boldsymbol{x}_k) + d(\boldsymbol{x}_k, \boldsymbol{x}_j)$ (triangle inequality).

kNN Model (cont)

The most common choice is the *Minkowski distance*:

$$d(\boldsymbol{x}, \boldsymbol{z}) = \|\boldsymbol{x} - \boldsymbol{z}\|_r = \left(\sum_{i=1}^{p} |x_i - z_i|^r \right)^{\frac{1}{r}}, \quad \boldsymbol{x}, \ \boldsymbol{z} \in \mathbb{R}^p. \quad (1)$$

Note that $\|\cdot\|^r$ is called the ℓ^r norm.

When $r = 1$, we have the *Manhattan distance*:
$\|\boldsymbol{x} - \boldsymbol{z}\|_1 = |x_1 - z_1| + |x_2 - z_2| + \cdots + |x_p - z_p|$.

When $r = 2$, we have the *Euclidean distance*:
$\|\boldsymbol{x} - \boldsymbol{z}\|_2 = \sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2 + \cdots + (x_p - z_p)^2}$.

kNN Model (cont)

When $r = \infty$, we have the *Chebyshev distance*:
$\|\boldsymbol{x} - \boldsymbol{z}\|_\infty = \max\{|x_1 - z_1|, |x_2 - z_2|, \cdots, |x_p - z_p|\}$.

The Euclidean distance is more sensitive to outliers than the Manhattan distance. When outliers are rare, the Euclidean distance performs very well and is generally preferred. When the outliers are significant, the Manhattan distance is more stable.

The kNN in R only supports Euclidean distance. The kNN in Python supports general Minkowski distance.

kNN Model (cont)

More Examples of 'Dissimilarity':

Apart from the Minkowski distance function (which only accepts numeric inputs), the following distance functions can be used in kNN:

A) Mahalanobis (knnGarden::knnMCN)

B) Gower (can be used to process mixed numeric and categorical data)

C) Jaccard score (for data that take discrete values, usually 0 and 1, e.g. black and white images) and its extension Tanimoto score (for data that can take on continuous values). They are used in Chemoinformatics, plagiarism detection, thesauras extration, market-basket transactional data, anomalies detection in spatio-temporal data.

kNN Model (cont)

The theoretical foundation behind the kNN model is the Bayes Optimal Classifier theory.

Suppose that we **know** $\mathbb{P}(y|\boldsymbol{x})$ (which is almost never the case), then the "optimal" prediction is

$$y^* = h_{\text{opt}}(\boldsymbol{x}) = \underset{y}{\text{argmax}}\, \mathbb{P}(y|\boldsymbol{x})$$

For this type of classifier, the **error rate** is

$$\epsilon_{BayesOpt} = 1 - \mathbb{P}(y^*|\boldsymbol{x})$$

This is a **lower bound of the error rate**.

kNN Model (cont)

The **upper bound on the error** is given by the constant classifier (in which regardless what input $\boldsymbol{x}$, the classifier just reply the 'same' output).

The following maths theory says that kNN is OK:

Theorem

As $n \to \infty$, the 1-NN error is no more than twice the error of the Bayes Optimal classifier:

$$\epsilon_{\text{BayesOpt}} \leq \epsilon_{NN} \leq 2\epsilon_{\text{BayesOpt}}$$

Similar guarantees hold for $k > 1$.

kNN Model (cont)

The kNN model with Euclidean distance (only) is supported in R by various packages:

```
class::knn(train, test, cl, k = 1, l = 0, prob = FALSE,
  use.all = TRUE)
FNN::knn(train, test, cl, k = 1, prob = FALSE,
  algorithm=c("kd_tree", "cover_tree", "brute"))
FNN::knn.reg(train, test = NULL, y, k = 3,
  algorithm=c("kd_tree", "cover_tree", "brute"))
```

Weighted kNN

The weights for the kNN model are equal. The weighted kNN (wkNN) model introduces a weight the k nearest neighbours in the following way:

1. Let $N(\boldsymbol{x}, k+1)$ be the $k+1$ nearest neighbours to $\boldsymbol{x}$ according to a distance function $d(\boldsymbol{x}, \boldsymbol{x}_i)$.
2. The $(k+1)$th neighbour is used for "standardisation" of the k smallest distances via $D_i = \frac{d(\boldsymbol{x}, \boldsymbol{x}_i)}{d(\boldsymbol{x}, \boldsymbol{x}_{k+1})}$.
3. A weighted majority of the k nearest neighbour

$$\hat{y} = \max_j \left\{ \sum_{i=1}^{k} K(D_i) I(y_i = j) \right\}.$$

Weighted kNN (cont)

The weighted kNN model with Minkowski distance are supported in R and Python respectively.

```
kknn::kknn(formula = formula(train), train, test,
  na.action = na.omit(), k = 7, distance = 2,
  kernel = "optimal", ykernel = NULL, scale=TRUE,
  contrasts = c('unordered' = "contr.dummy",
  ordered = "contr.ordinal"))
```

```
sklearn.neighbors.KNeighborsClassifier(n_neighbors=5,
  *, weights='uniform', algorithm='auto',
  leaf_size=30, p=2, metric='minkowski',
  metric_params=None, n_jobs=None)
```

The options `distance` in R and `p` in Python correspond to the parameter *r* in the Minkowski distance formula (1).

Weighted kNN (cont)

The option `kernel` in R and the option `weights` in Python correspond to the the *kernel (function) K* which satisfies

1. $K(x) \geq 0$ for all $x \in \mathbb{R}$;
2. $K(x)$ is maximum when $x = 0$;
3. $K(x)$ descents monotonously when $x \to \pm\infty$.

A *rectangular kernel* has the form

$$K(x) = \frac{1}{2} I(|x| \leq 1)$$

and when `kernel=rectangular` (R's kknn) or when `weights=uniform` (Python), wkNN are theoretically the same as kNN.

Weighted kNN (cont)

Python only provides the 'distance' kernel $K(x) = 1/|x|$ for wkNN while R provides more options for the kernel:

- inv: $K(x) = 1/|x|$
- triangular: $K(x) = (1 - |x|) \cdot I(|x| \leq 1)$.
- optimal: The number of neighbours used for this kernel should be $\left(\frac{2(d+4)}{d+2}\right)^{\frac{d}{d+4}} k \in [1.2, 2]$, where k is used
- Others: cos, gaussian, rank, epanechnikov (or beta(2,2)), biweight (or beta(3,3)), triweight (or beta(4,4)).

Outline

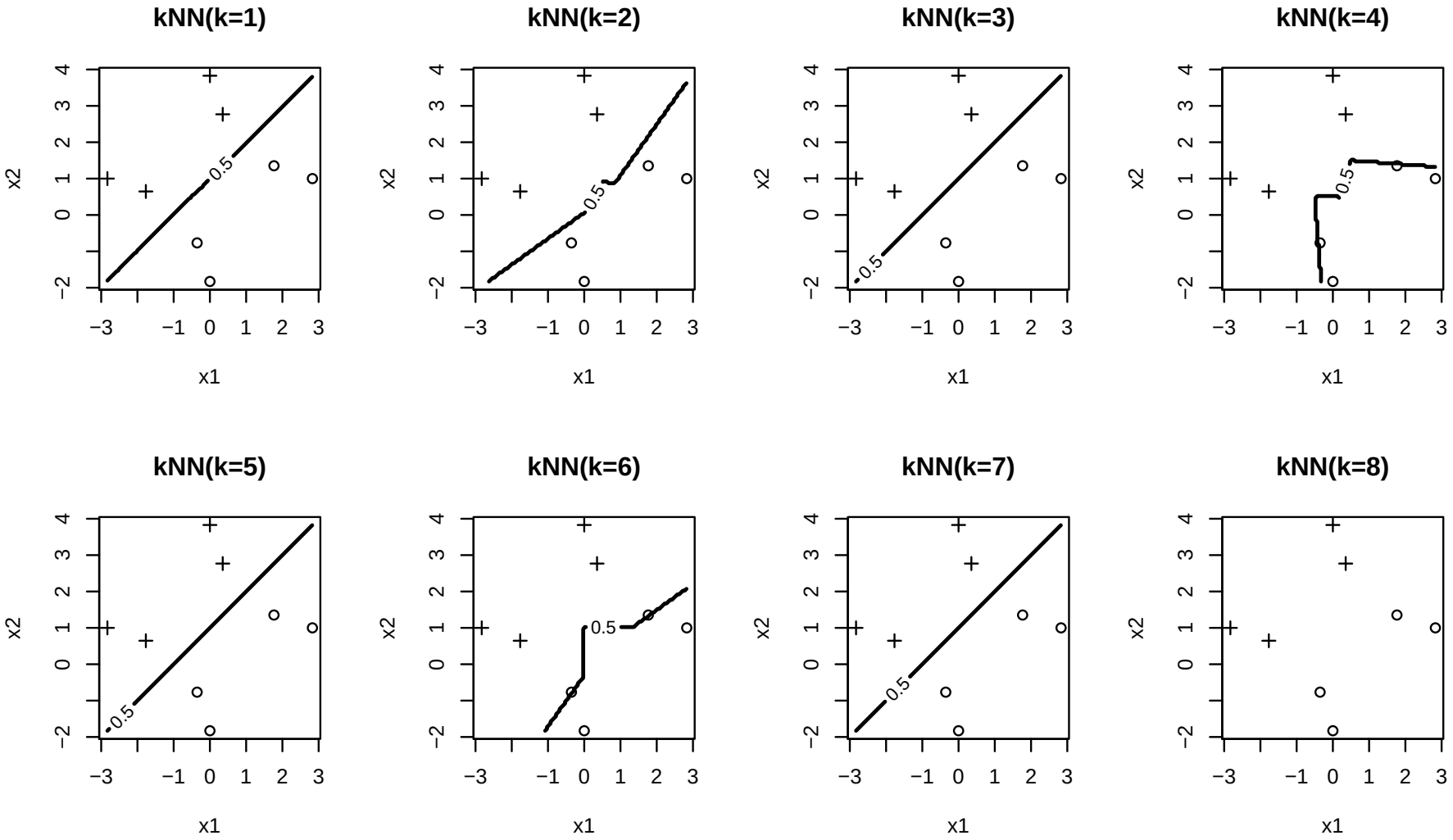
Results Interpretation

The **classifier boundary** of a kNN model is the boundary between different classes. We can think of it as the "boundaries" between "countries", for example, Malaysia has boundaries with Singapore, Thailand and Indonesia.

Most data are not 2D, for example, the iris data. But most analysis will try to reduce to 2D to "visualise" the boundary. We can find a comprehensive analysis of the iris data using R with beautiful diagrams on `https://www.datacamp.com/community/tutorials/machine-learning-in-r`. As for the drawing the kNN boundary between classes, a "contour" plot is required and is illustrated by `https://stats.stackexchange.com/questions/21572/how-to-plot-decision-boundary-of-a-k-nearest-neighbor-c`

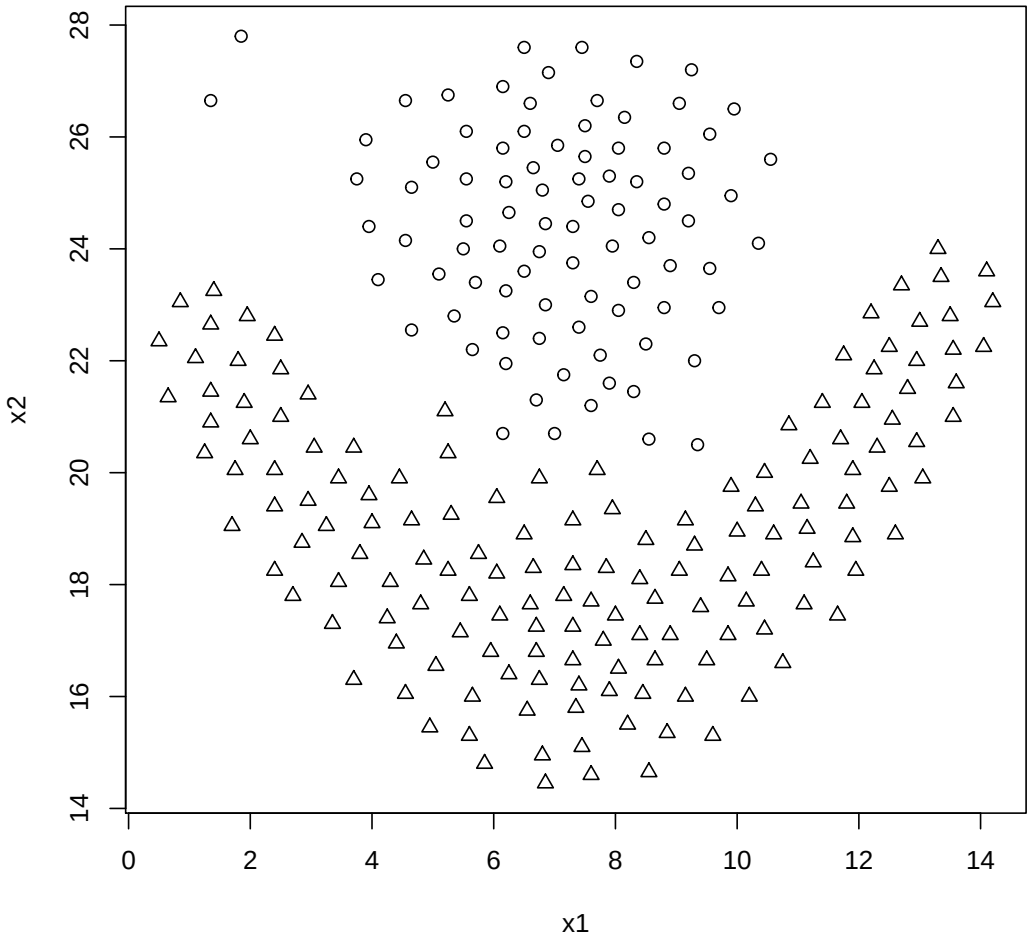
Results Interpretation (cont)

First, consider the boundary of the 'data1' found in your Practical 1 which is symmetrical. Therefore, we "feel" that it is a "line".

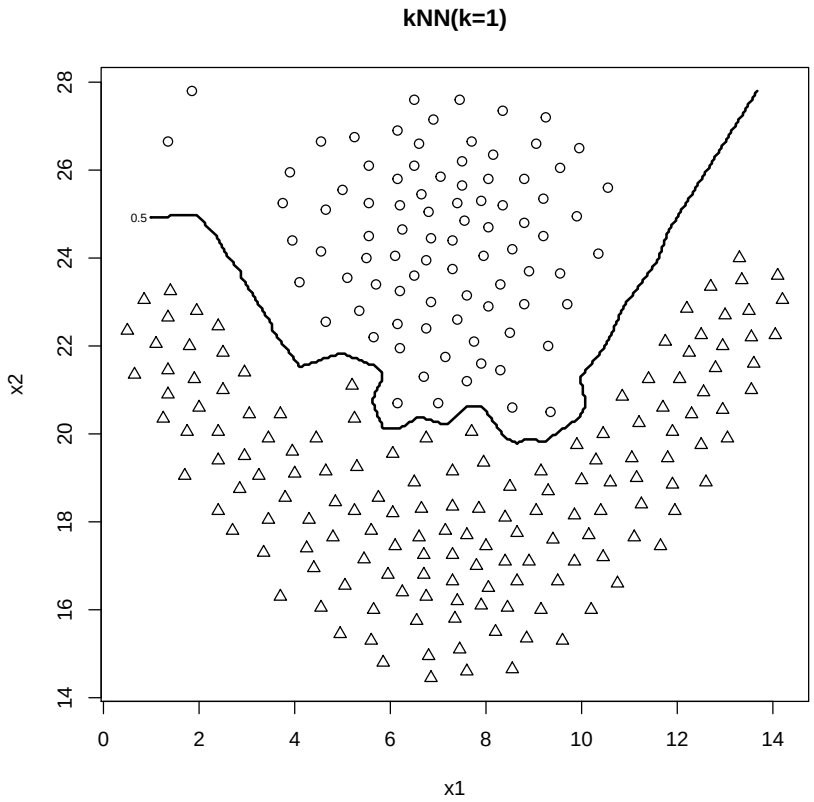


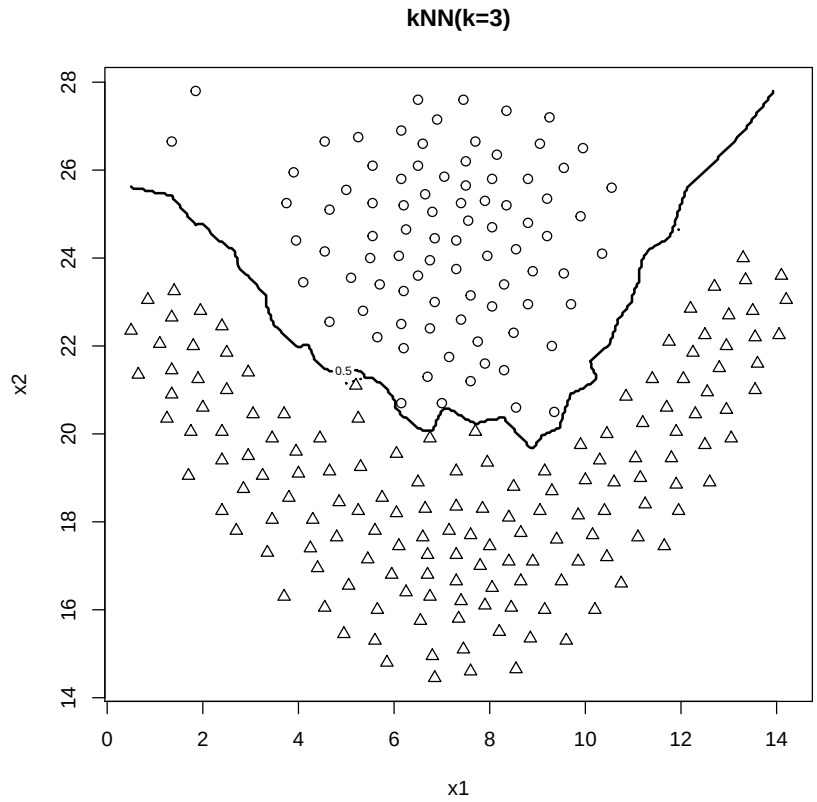
Results Interpretation (cont)

Now, we investigate a “nonlinear boundary” data.

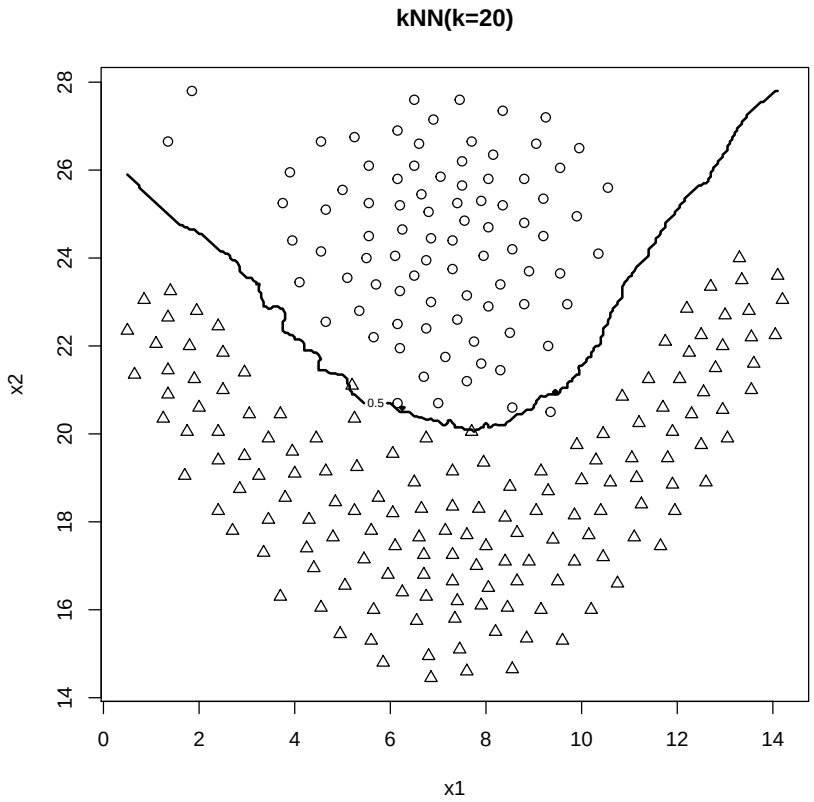


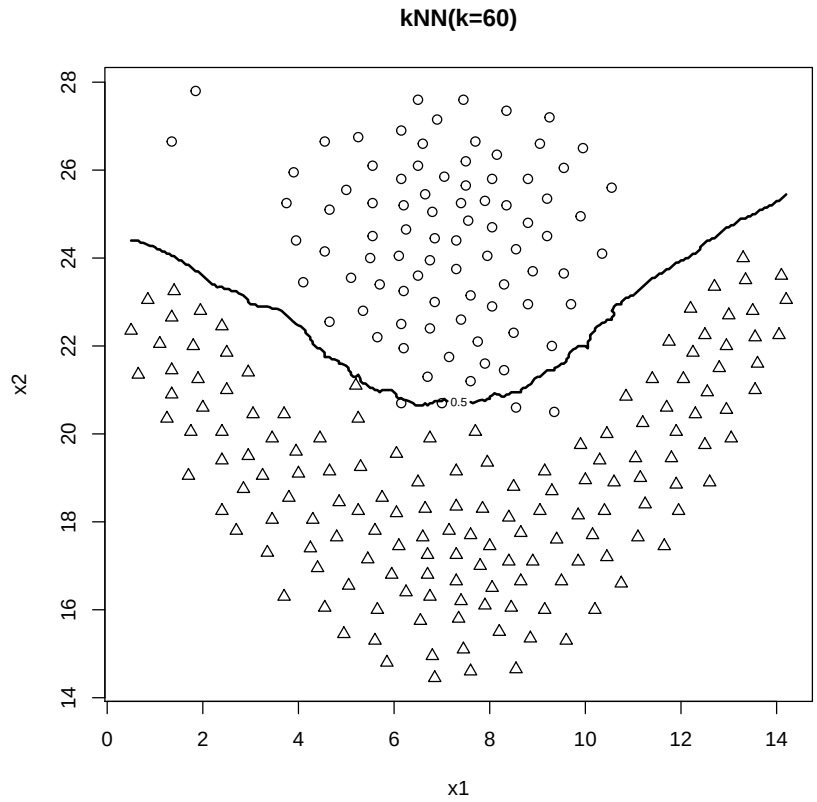
Results Interpretation (cont)



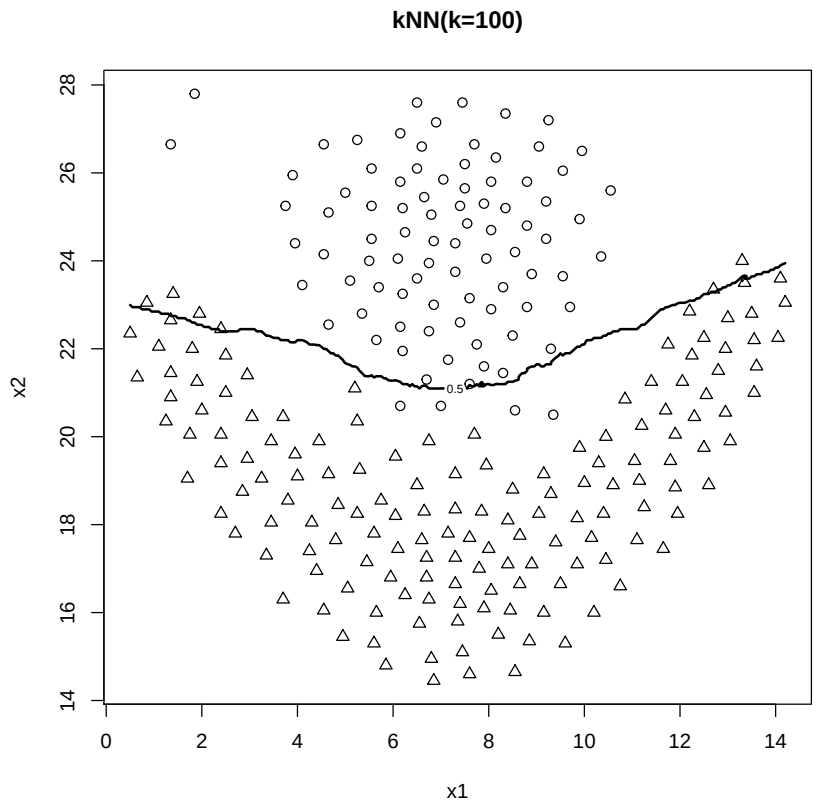


Results Interpretation (cont)





Results Interpretation (cont)



Conclusion: The choice of k depends upon the data.
In general,

- k small, decision boundary is over "flexible" $\Rightarrow$ kNN classifier is low bias, high variance.
- k large, decision boundary is less flexible, "less noise", "smoother", more "linear" $\Rightarrow$ kNN classifier is low-variance, high bias.

Outline

Models Comparison

For kNN (as well wkNN), the key to the 'best' model relies on

- the right number of neighbours k (which controls the decision boundary);
- the right distance measure (and data pre-scaling using standard deviation or min-max scaling)

The Euclidean distance is usually fine. What is more important is the scaling of the data so that the standard deviation of all predictors are similar.

We usually vary the **hyperparameter** *k* and use the performance measure (e.g. accuracy) to find the best model.

Models Comparison (cont)

Similar to Logistic Regression (LR) introduced in the previous week, we can also add or remove predictors X_i and compare models but this is not really a good idea because unlike which has model measurements such as AIC. kNN does not have model measurements and performance measures are the only tools we have.

The recommendation for model selection depends on the size of the dataset:

- large dataset: 3-way holdout (train-validation-test split) should be used.
- small dataset: K-fold CV / LOOCV with independent test set should be used.

Outline

Case Study 1

Example 1: A sport school would like to group their new enrolled students into 2 groups, as according to the existing students' weight and height. The weight and height of 7 existing students with group are shown in the table below.

Student	Weight (kg)	Height (cm)	Group
A	29	118	A
B	53	137	B
C	38	127	B
D	49	135	B
E	28	111	A
F	24	111	A
G	30	121	A

Case Study 1 (cont)

(a) Perform and use $k = 3$-NN method (with Euclidean distance) to predict which group the following students will be grouped into, based on **cut-off of 0.7** on group A.

Student	Weight (kg)	Height (cm)
H	35	120
I	47	131
J	22	115
K	38	119
L	31	136

Class Discussion with Excel.

Case Study 1 (cont)

(b) The actual groups of the students are {A, B, A, B, B} for students {H, I, J, K, L} respectively. Construct a confusion matrix and calculate the accuracy measurements for a cut-off of 0.5 and a cut-off of 0.7.

(c) Use R to confirm the calculations in part (b) above.

Class Discussion with Excel and R.

Case Study 2

The *kNN regressor* applies "mean" instead of "mode" to "predict" the output.

Example 2: (Exam SRM Study Manual, p225, Q15.10) A continuous variable Y is modelled as a function of X using kNN with $k = 3$. With the following data:

X	5	8	15	22	30
Y	4	1	10	16	30

Calculate the fitted value of Y at $X = 12$. Try write an R script to perform the calculation.

Class Discussion with Excel and R.

Case Study 3

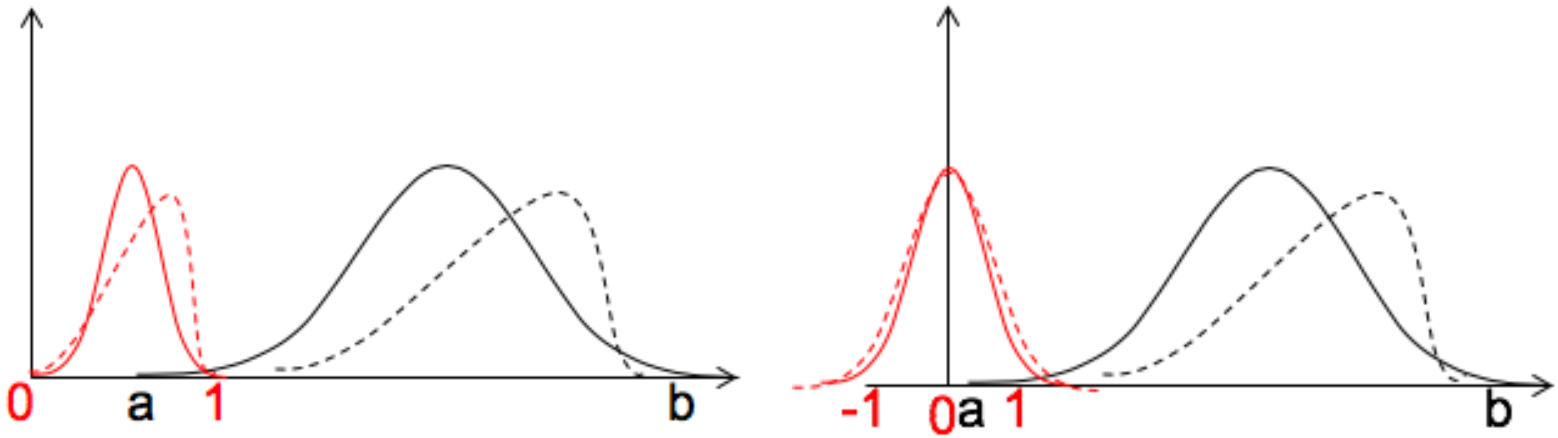
Feature scaling / Standardisation: Put all variables into the "similar" range, the variables are equally weight. This is essential in kNN to prevent the classification / regressions are dominated by columns with large variations.

Two popular methods:

- Min-Max Normalisation (Rescaling):
  $M(X_{ij}) = \frac{X_{ij} - X_{\text{min,j}}}{X_{\text{max,j}} - X_{\text{min,j}}}$
- Standard Scaler / Standardisation:
  $M(X_{ij}) = \frac{X_{ij} - \overline{X}_j}{s_{X,j}}$

Case Study 3 (cont)

The following diagrams illustrate the effects of applying the min-max scaling and the standardisation to a data in the range a to b:



The former transforms to the range [0, 1] while the later transforms to a “standard normal distributed” range.

Case Study 3 (cont)

Example 3: Given the training data and testing data in **Example** 1, apply the standardisation and then use the $k = 3$-NN method (with Euclidean distance) to predict which group the students will be grouped into.

Solution

Student	Weight (kg)	Height	Norm_Wgt	Norm_Hgt	Group
A	29	118	-0.6113	-0.4587	A
B	53	137	1.5284	1.3355	B
C	38	127	0.1910	0.3912	B
D	49	135	1.1717	1.1467	B
E	28	111	-0.7005	-1.1197	A
F	24	111	-1.0571	-1.1197	A
G	30	121	-0.5222	-0.1754	A
Mean	35.8571	122.8571			
Std. dev	11.2165	10.5898			

Case Study 3 (cont)

Solution (cont)

$k = 3$:

		H	I	J	K	L
	Group	Distance	Distance	Distance	Distance	Distance
A	A	0.5673	2.0205	0.6854	0.8079	1.7091
B	B	2.2699	0.7792	3.4575	2.1628	1.9637
C	B	0.7131	0.8869	1.8218	0.7554	1.0544
D	B	1.8879	0.4177	3.0596	1.8013	1.6076
E	A	1.0544	2.5370	0.6548	1.1686	2.3759
F	A	1.2977	2.7878	0.4177	1.4590	2.4419
G	A	0.4557	1.7857	0.9109	0.7378	1.4193
$P(Y = A)$		0.6667	0.0000	1.0000	0.6667	0.3333
Cut-off = 0.5	$\hat{y}$	A	B	A	A	B

Outline

Lab Practice on Classification Method 2

`s42_knn.R`