Forms of Learning

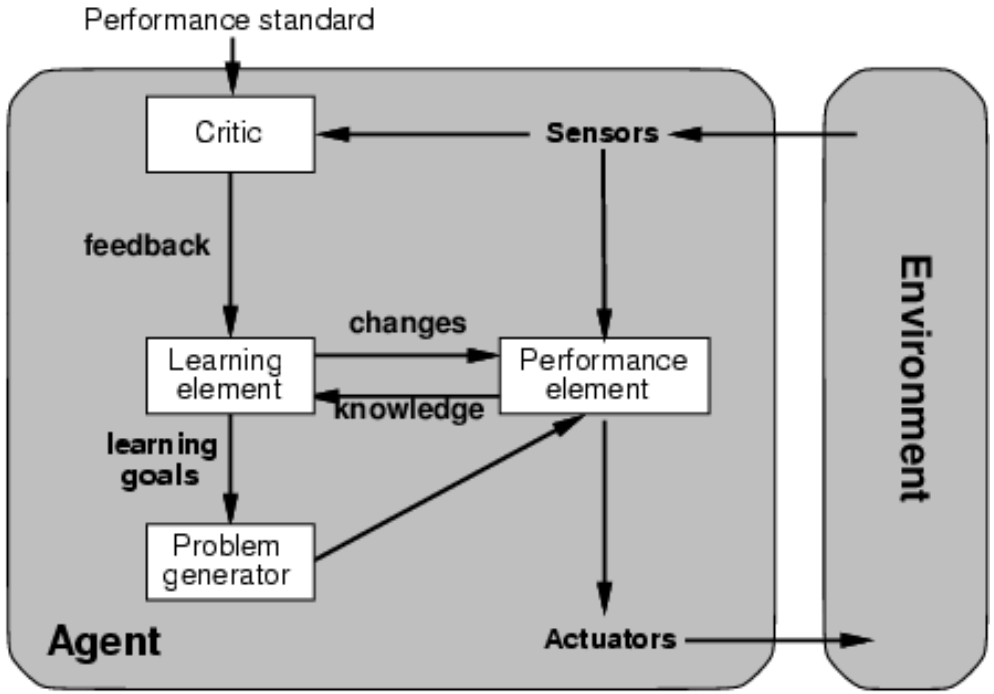
3007/7059 Artificial Intelligence

School of Computer Science
The University of Adelaide

What is learning?

In intelligent agents, percepts should be used not only for acting, but also for improving the agent's ability to act in the future.

A learning agent has a performance element and learning element.



Learning allows an agent to **adapt to new circumstances** and to **detect and extrapolate patterns**.

Forms of Learning

Learning depends on four major factors:

- which component(s) in the performance element are to be improved.
- what prior knowledge the agent already has (i.e, are we building the agent from scratch, or just updating it?).
- the representation of the percept, e.g., numeric, character strings, sets of objects.
- what feedback is available to learn from.

Three main forms of learning in intelligent agents:

- **Supervised learning**.
- **Unsupervised learning**.
- **Reinforcement learning**.

The form of learning to use depends on the **type of feedback**.

Supervised Learning

Given a **training set** of N example **input-output pairs**

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N),$$

where each y_i was generated by an unknown function $y = f(x)$, **discover a function** h that approximates f.

The goal is to **predict the output value** y^* for a new unseen before input x^*.

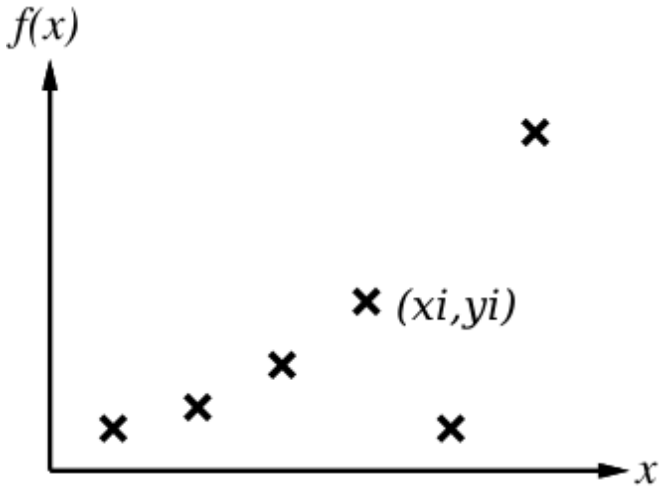
The availability of a **target output value** y_i for each input x_i differentiates supervised learning from other forms of learning.

The function h is called a **hypothesis**. Supervised learning can be viewed as searching in the space of hypotheses for the best h.

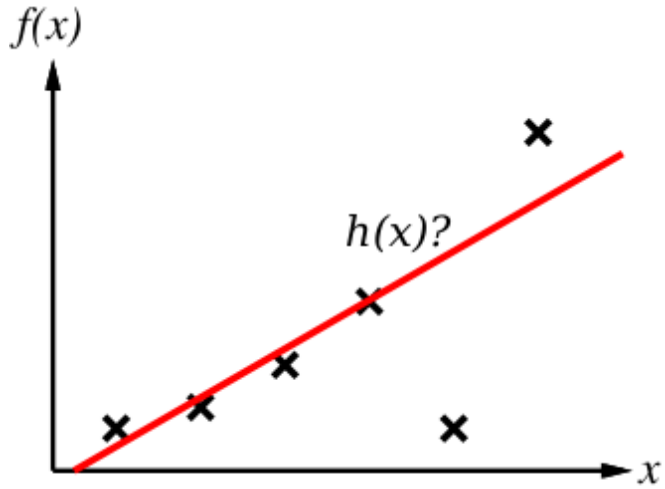
Supervised Learning (cont.)

The values for x_i, y_i can be numeric, discrete, continuous, non-numeric (nominal, symbols, text), etc.

When the target outputs are **continuous** values, we have a **regression** problem.



(a) Training set of input-output pairs. $f(x)$ is unknown.

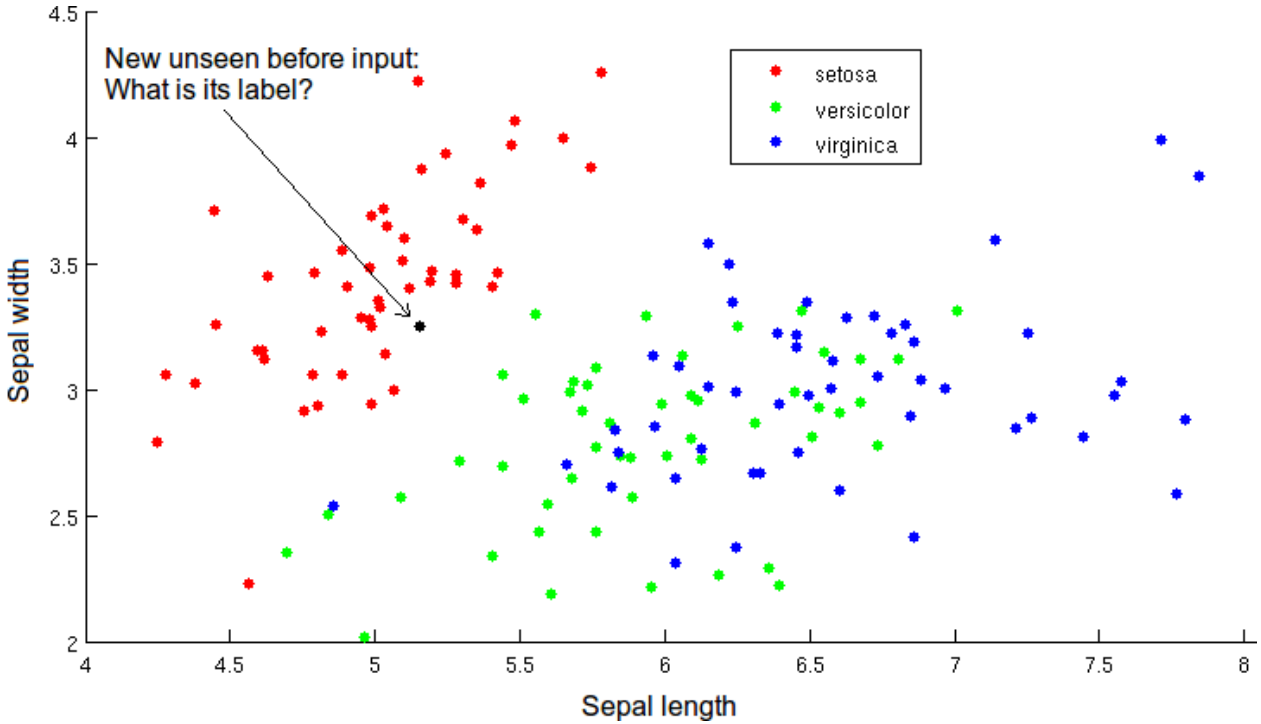


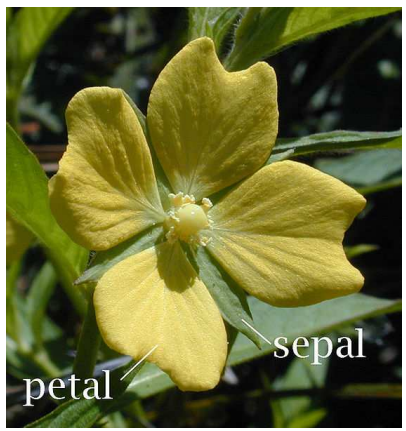
(b) Candidate hypothesis $h(x)$.

Supervised Learning (cont.)

When the target outputs are nominal values or **class labels**, we have a **classification** problem.

Example: Given measurements of sepal length and sepal width, identify the **type** of flower.



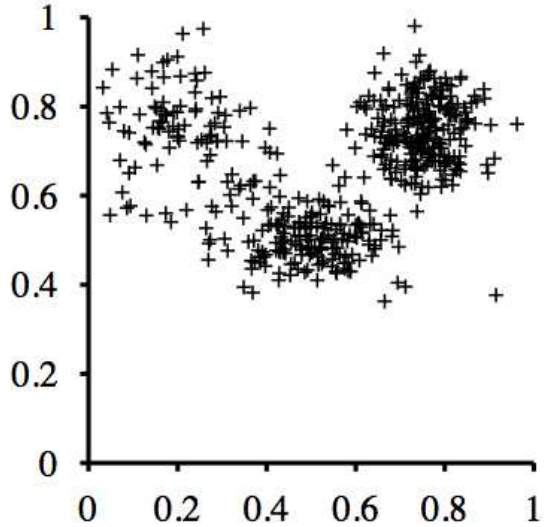


Here each input value $\mathbf{x}_i$ is a two-dimensional vector, while each target value y_i can be setosa, versicolor or virginica.

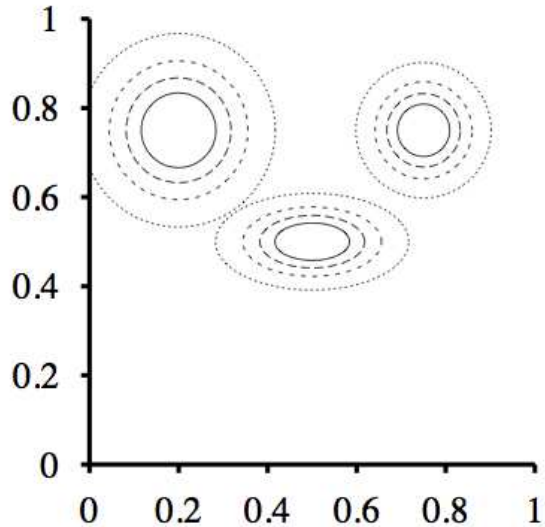
Unsupervised Learning

Learning **patterns** in the input when no specific target output values are supplied.

Example: Finding clusters in 2D input data.



(a) Input data with unknown cluster memberships.



(b) A hypothesis clustering.

Reinforcement Learning

Learning what actions to take in order to maximise some **reward** or **utility**.

The reward (or penalty) is given after a **sequence of actions**. This differs from supervised learning in that explicit input-output examples are not available.

Example 1

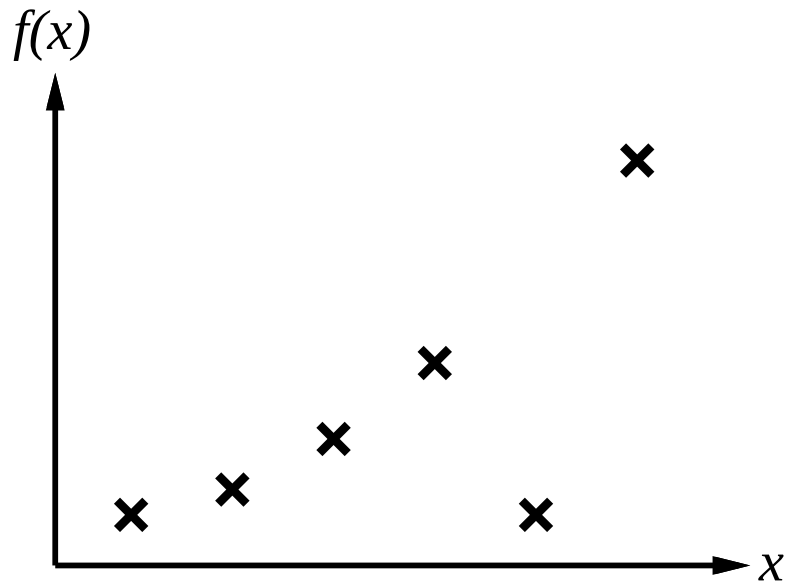
Learning how to play chess without a teacher. The "feedback" or "reward" is unavailable until the end of the game, but you must decide what steps to take in the middle of the game.

Example 2

Learning to play ping pong. You must decide what shot to make before you get to the end of the point which you will either win or lose (reinforcement).

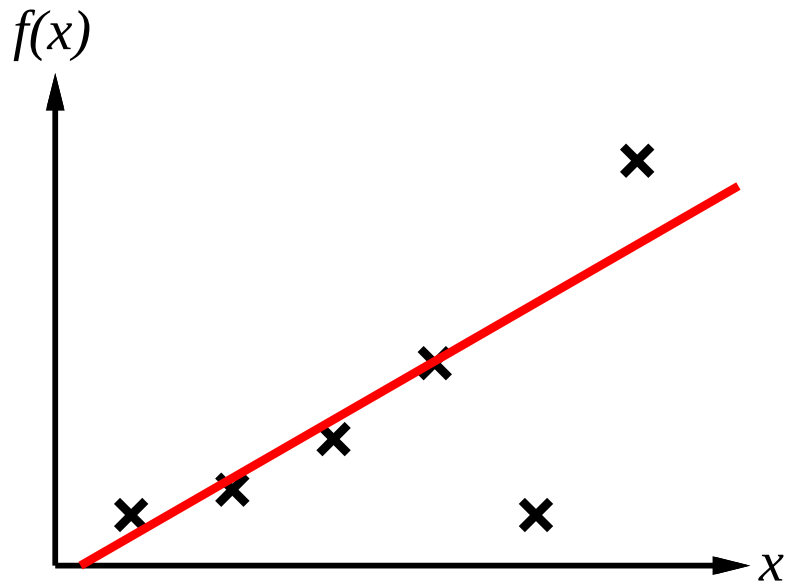
Complexity of hypothesis

Back to the regression problem. What is the best hypothesis?



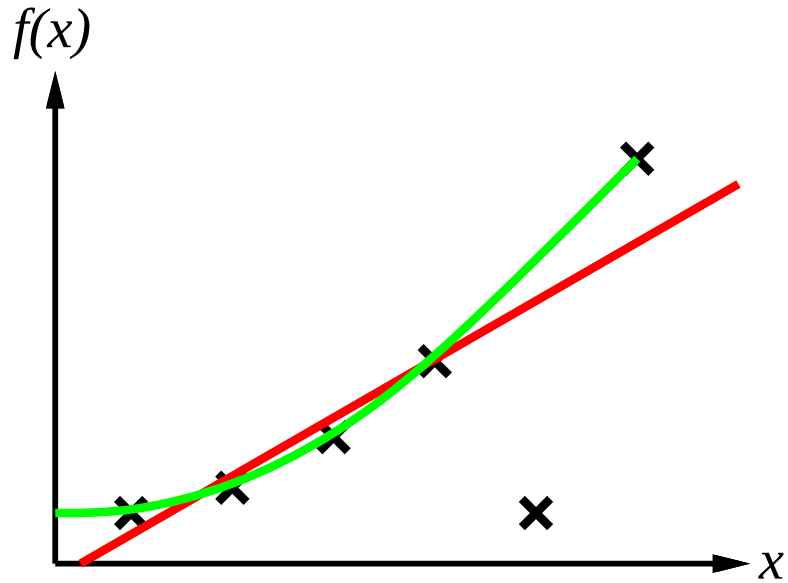
Complexity of hypothesis (cont.)

Use a linear hypothesis? It doesn't agree with all the points...



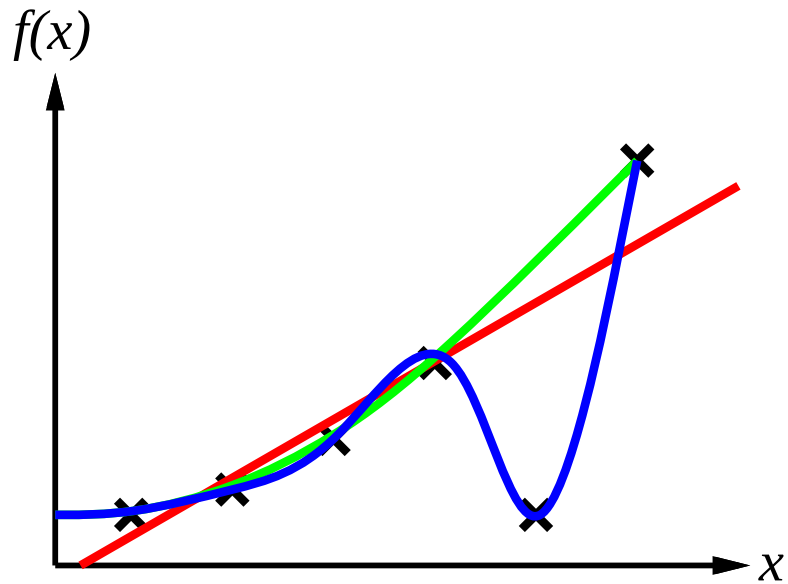
Complexity of hypothesis (cont.)

Use a non-linear polynomial hypothesis? Slightly better...



Complexity of hypothesis (cont.)

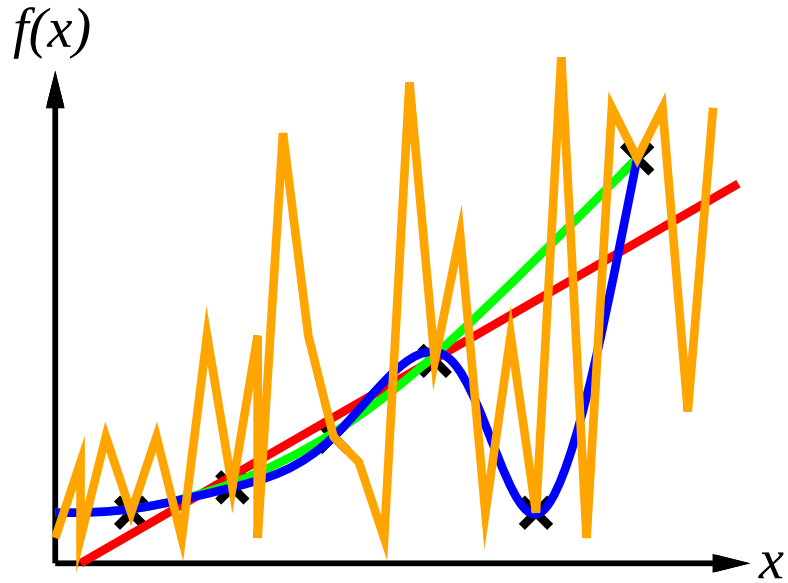
Increase the degree of the polynomial to fit all points.



A hypothesis is **consistent** if it agrees with all the data.

Complexity of hypothesis (cont.)

What about this one? Which hypothesis should we choose?



Occam's Razor

"Plurality should not be posited without necessity."

William of Ockham

"Everything should be made as simple as possible, but not simpler."

Albert Einstein

A hypothesis can be regarded as an explanation of the data.

Occam's Razor states that we should prefer the simplest hypothesis consistent with the data.

If we accept that the data are noisy versions of the true data generated from f, it might be better to fit a simple hypothesis which is not exactly consistent with the data.

This **tradeoff** between hypothesis complexity and goodness-of-fit will crop up again in the learning algorithms we study later.

Nearest Neighbour Method

3007/7059 Artificial Intelligence

School of Computer Science
The University of Adelaide

Introduction

Given a testing/query point, Nearest Neighbours performs a search for the point(s) closest to the testing point in the training data.

Nearest Neighbours is one of the simplest methods for statistical learning. It belongs to a group of techniques called "non-parametric learning methods" because it does not make use of any assumed model for the data.

When properly harnessed Nearest Neighbours can provide excellent classification accuracy despite its simplicity.

K-nn

Nearest Neighbours is primarily a **supervised learning** algorithm.

Let $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be a set of N training data, where $\mathbf{x}_i$ is the i-th feature point, and y_i is the class label of $\mathbf{x}_i$. Note that $\mathbf{x}_i$ can be **multi-dimensional**.

Given a testing point $\mathbf{z}$ we wish to search among X the K points

$$\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_K^*$$

that are closest to $\mathbf{z}$, i.e. the K **nearest neighbours** of $\mathbf{z}$. It is almost always the case that $K < N$.

The testing point $\mathbf{z}$ is then assigned the **mode label** (label with highest count) among its K nearest neighbours:

$$mode(y_1^*, y_2^*, \ldots, y_K^*)$$

A classifier that performs classification in this manner is often called a **K-nn classifier**.

K-nn (cont.)

The concept of "nearness" requires a way of measuring the distance between any two pairs of points. Under K-nn the distance used is frequently the **Euclidean** distance:

$$d(\mathbf{z}, \mathbf{x}) = \sqrt{\sum_{j=1}^{M} (\mathbf{z}[j] - \mathbf{x}[j])^2}$$

This is essentially the **straight line distance** between two points.

Example 1: If $\mathbf{x} = \begin{bmatrix} 0.5 \\ 2 \end{bmatrix}$ and $\mathbf{z} = \begin{bmatrix} 2.3 \\ -1.5 \end{bmatrix}$ then

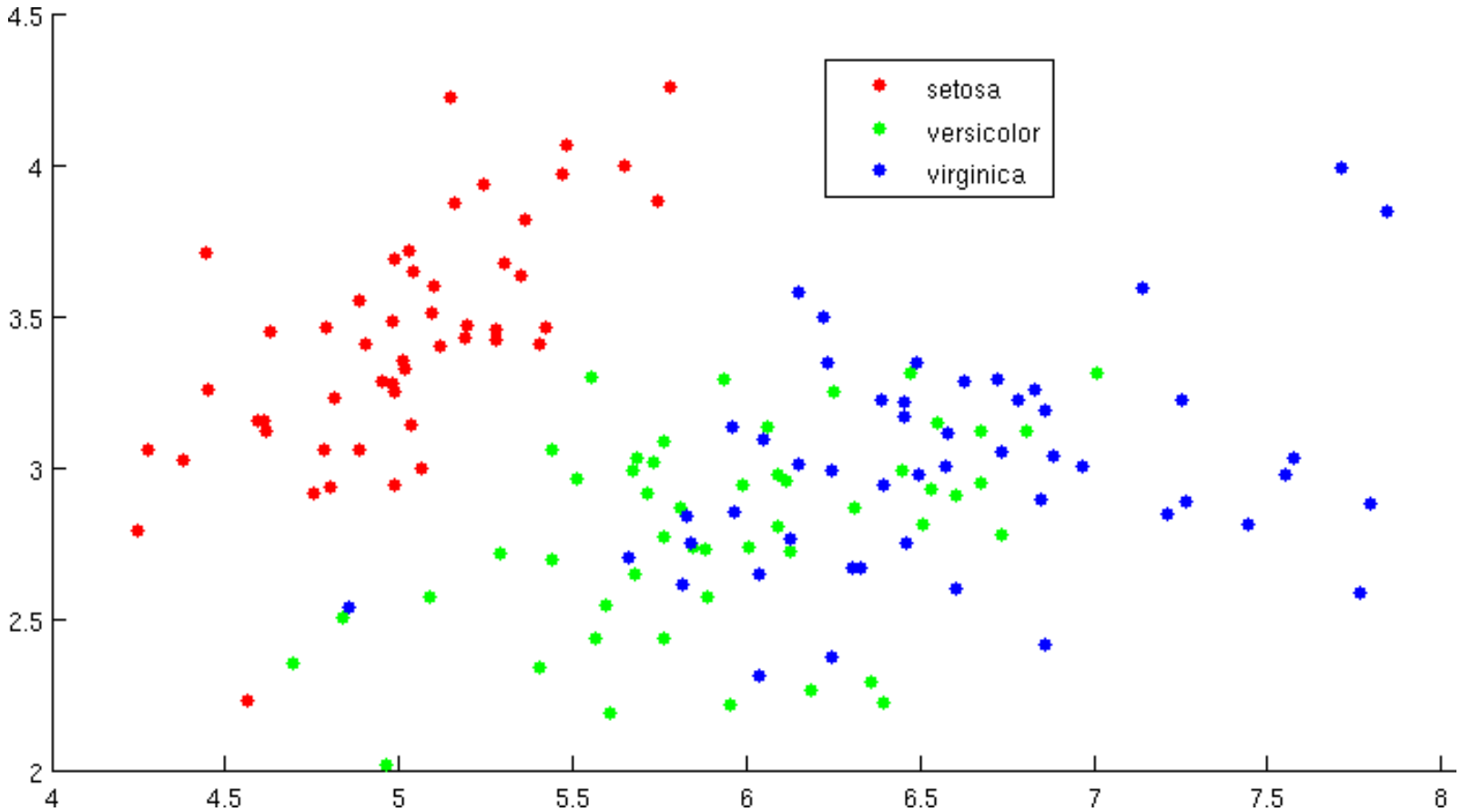
$$d(\mathbf{z}, \mathbf{x}) = \sqrt{(0.5 - 2.3)^2 + (2 - (-1.5))^2} = \sqrt{15.49}$$

Example 2: If $\mathbf{x} = \begin{bmatrix} -12 \\ 0 \\ -9.8 \end{bmatrix}$ and $\mathbf{z} = \begin{bmatrix} 13 \\ -1.5 \\ -7 \end{bmatrix}$ then

$$d(\mathbf{z}, \mathbf{x}) = \sqrt{(-12 - 13)^2 + (0 - (-1.5))^2 + (-9.8 - (-7))^2} = \sqrt{635.09}$$

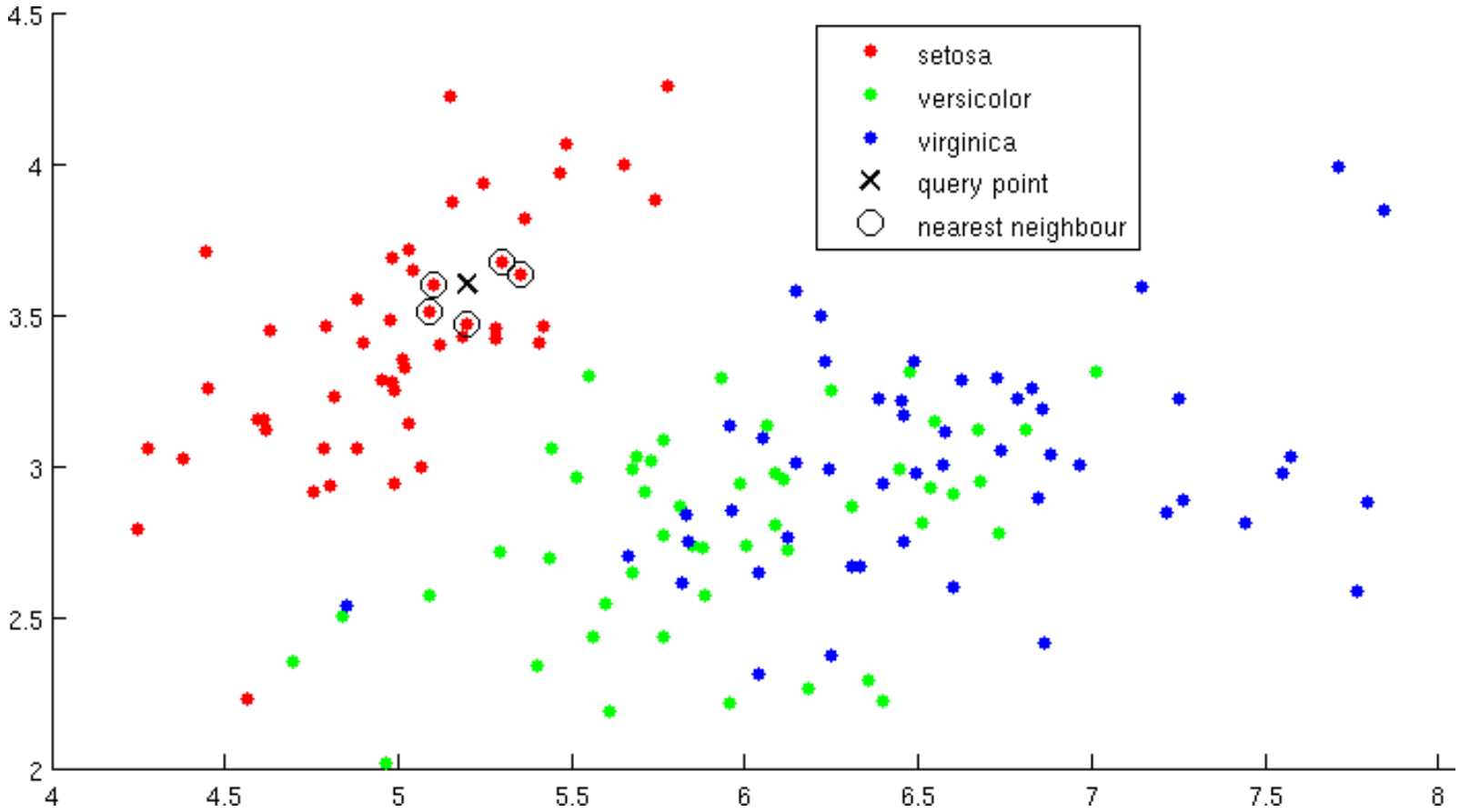
Iris data again

We use the Iris data again. This time we have 3 types of irises: Setosa, Versicolour and Virginica.



Iris data again (cont.)

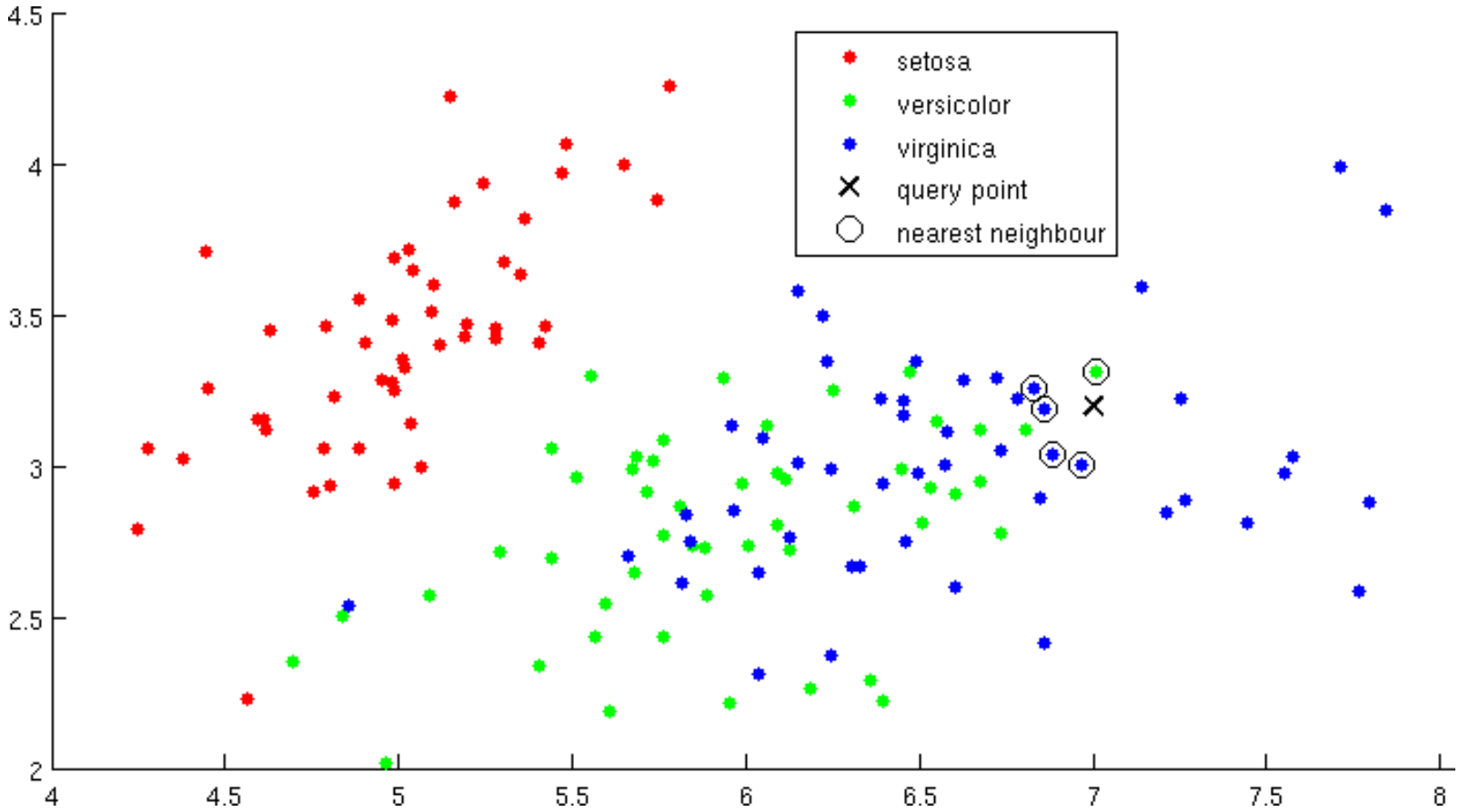
Given a testing point $\mathbf{z} = [\ 5.2 \quad 3.6\ ]^T$, we find its 5 nearest neighbours:



Observe that all the nearest neighbours are labelled Setosas. Thus $\mathbf{z}$ is assigned the label Setosa as well.

Iris data again (cont.)

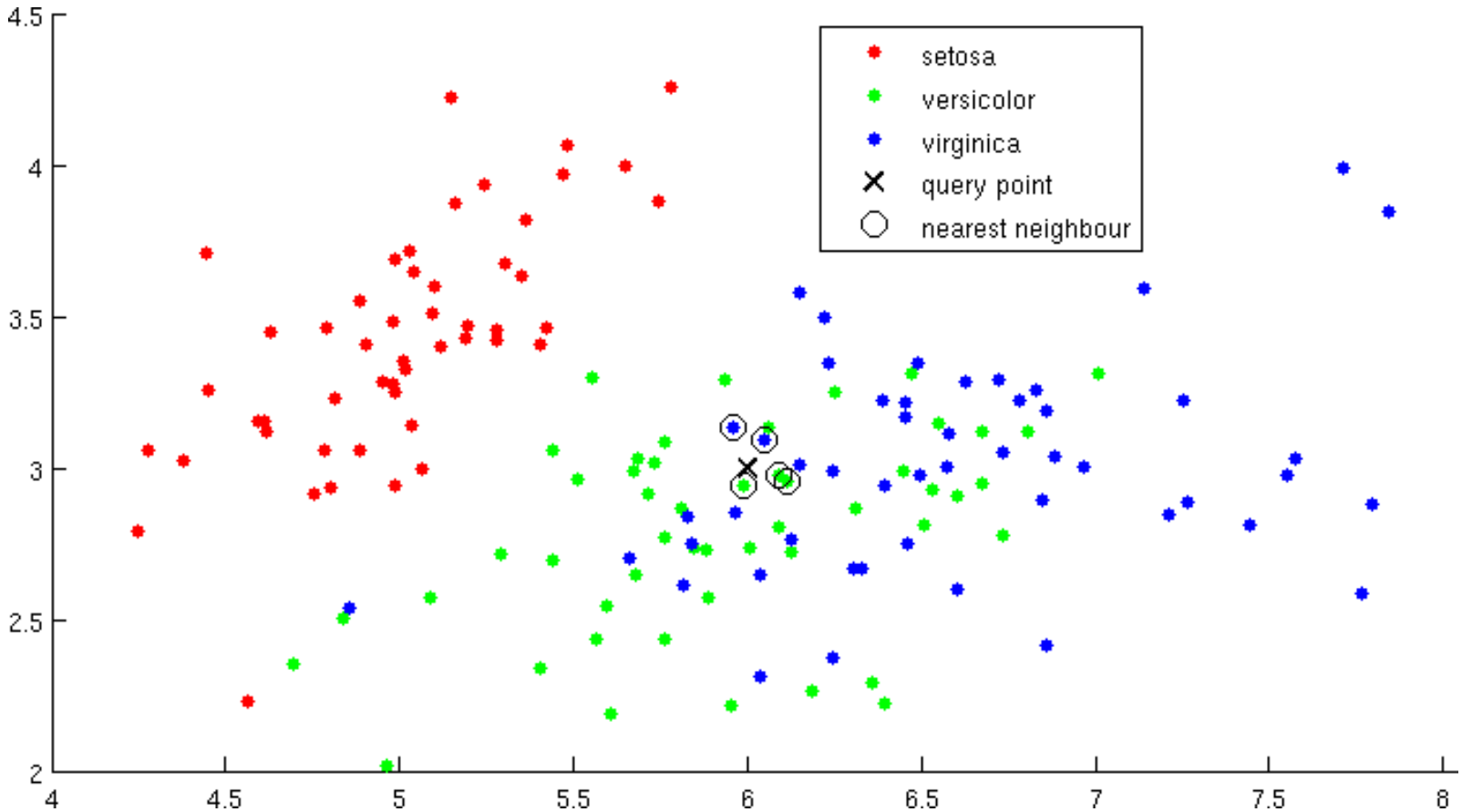
Given another testing point $\mathbf{z} = [\ 7.0 \ \ 3.2\ ]^T$, the 5 nearest neighbours are:



The label for $\mathbf{z}$ is taken as the mode among 1 Versicolour and 4 Virginicas, thus $\mathbf{z}$ is assigned the label of Virginica.

Iris data again (cont.)

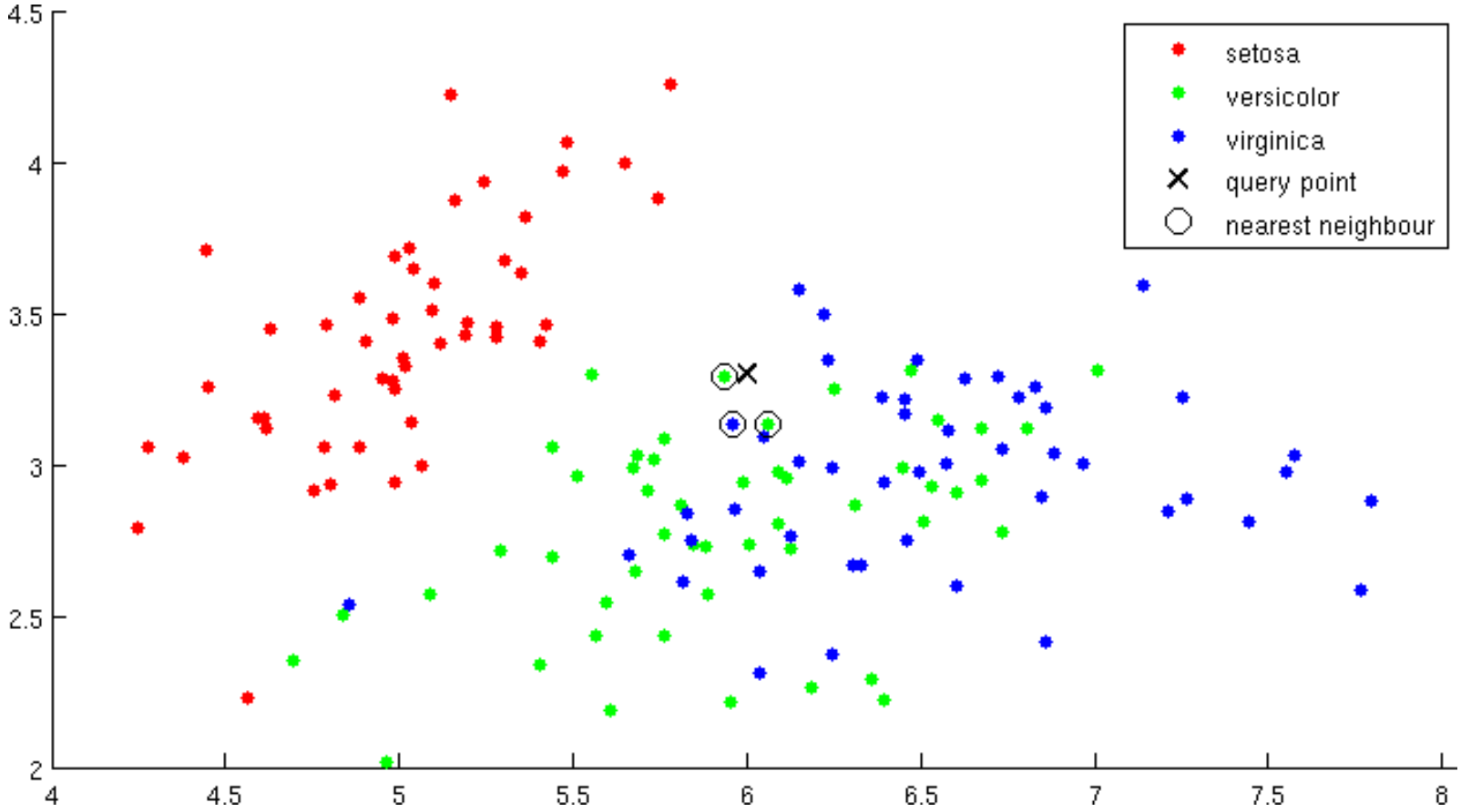
Similarly, the testing point $\mathbf{z} = [\ 6.0 \quad 3.0\ ]^T$ with 5 nearest neighbours is then labelled as Versicolour:



K is a free parameter

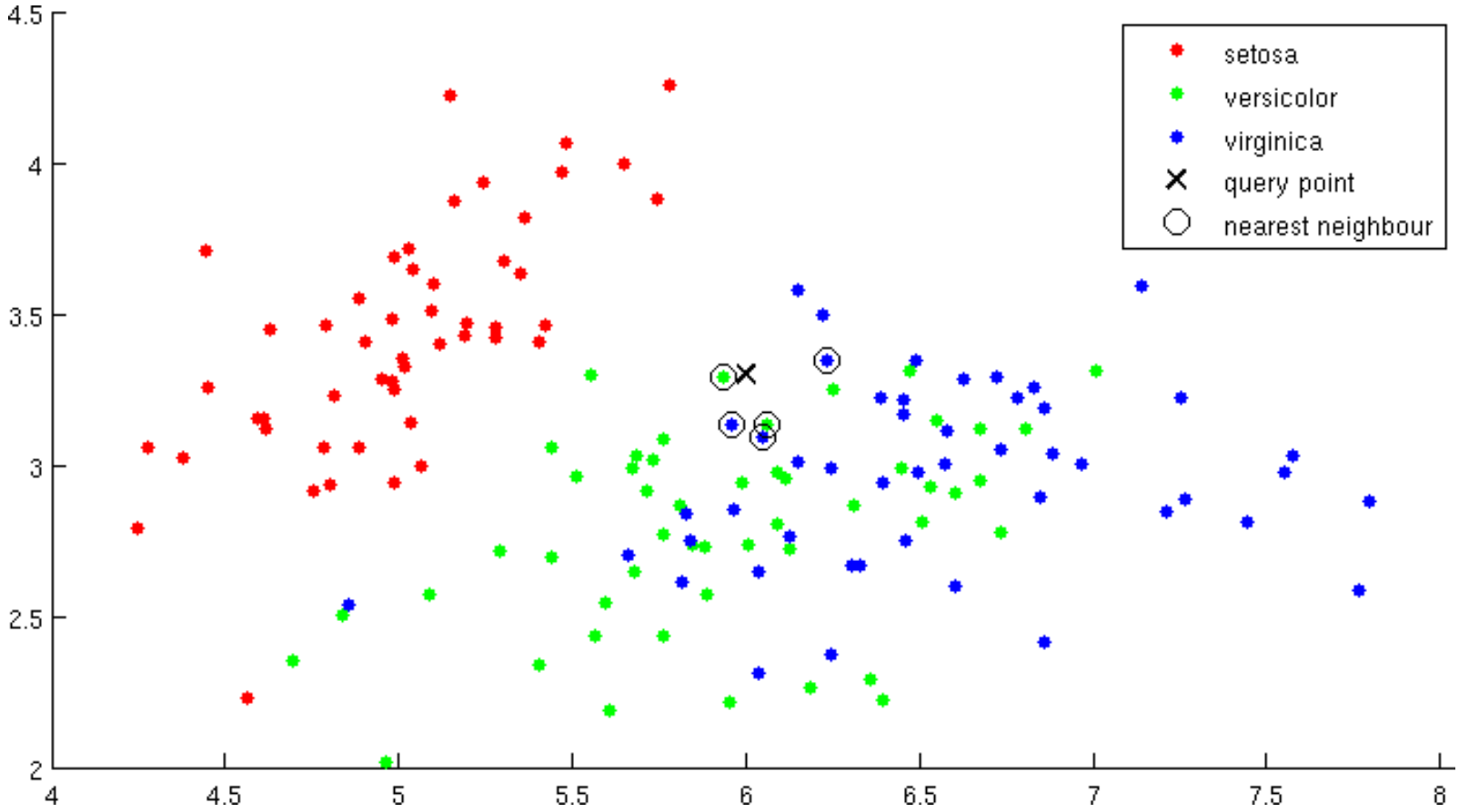
It is obvious that using different values of K will give different labelling results, especially at the boundaries between classes.

For example, for the testing point $\mathbf{z} = [\ 6.0 \quad 3.3\ ]^T$ with $k = 3$ gives the label of Versicolour.



K is a free parameter (cont.)

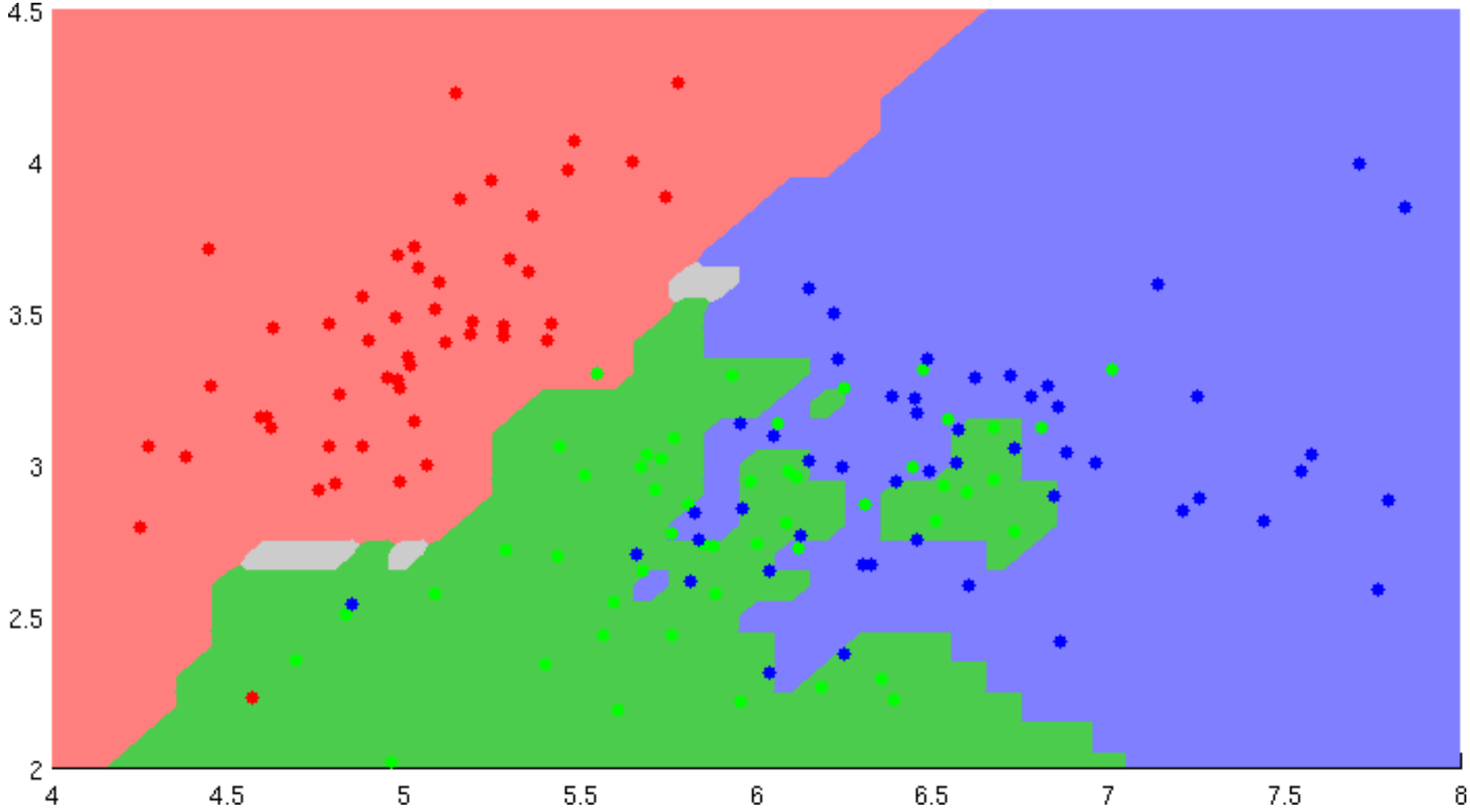
Testing the same point of $\mathbf{z} = [\ 6.0 \quad 3.3\ ]^T$ with $k = 5$ however gives the label of Virginica.



For most problems, setting the appropriate K is crucial. One often tunes K in a **trial-and-error** fashion (another drawback of K-nn).

Effects of different K's

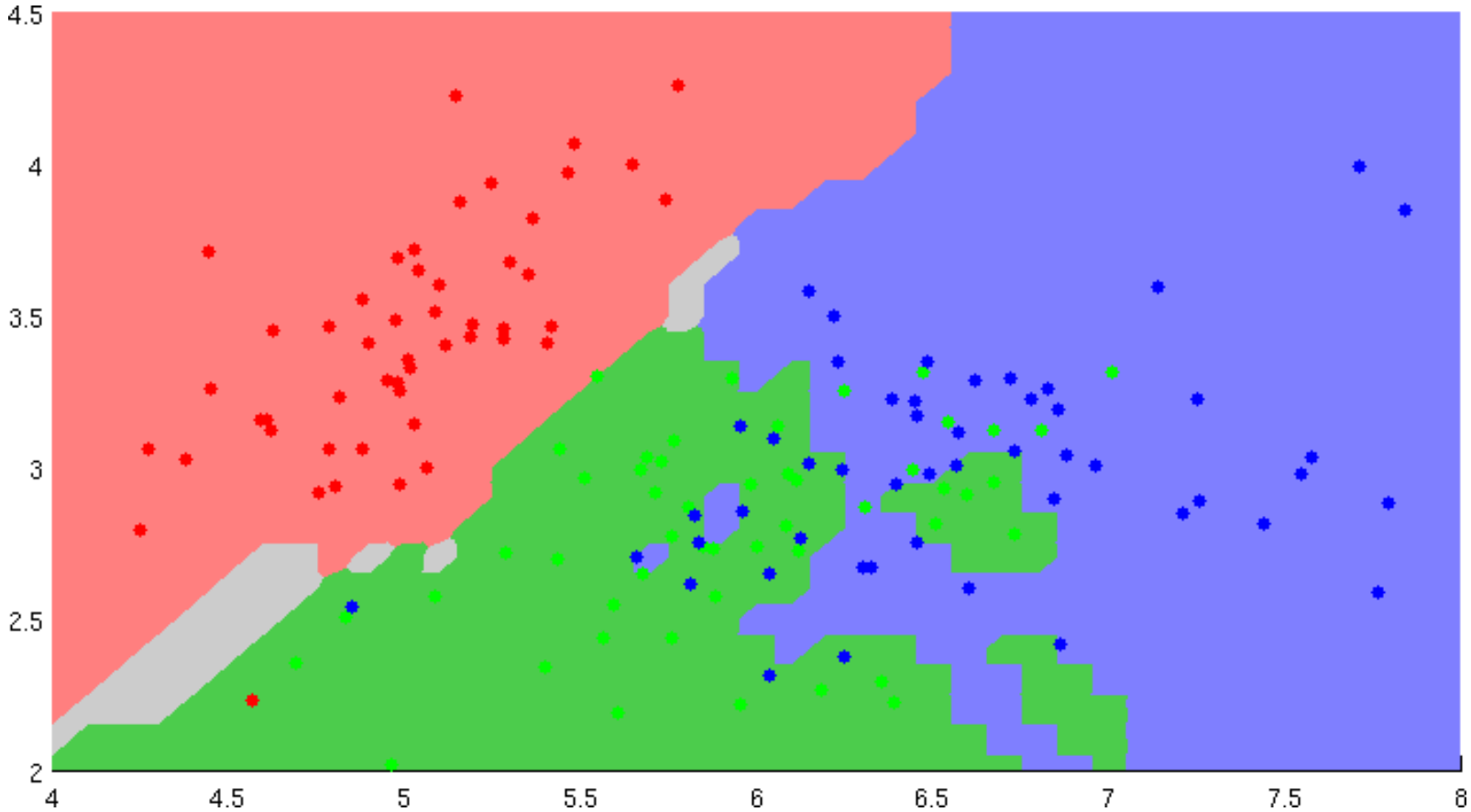
One way of visualising the effects of different K's is to plot the **decision areas** for the data. We subject every location (at suitable increments) to the K-nn classifier and set the colour accordingly. Here are the decision areas for $K = 3$.



Coloured grey are areas with no classification, i.e. **no unique mode labels**.

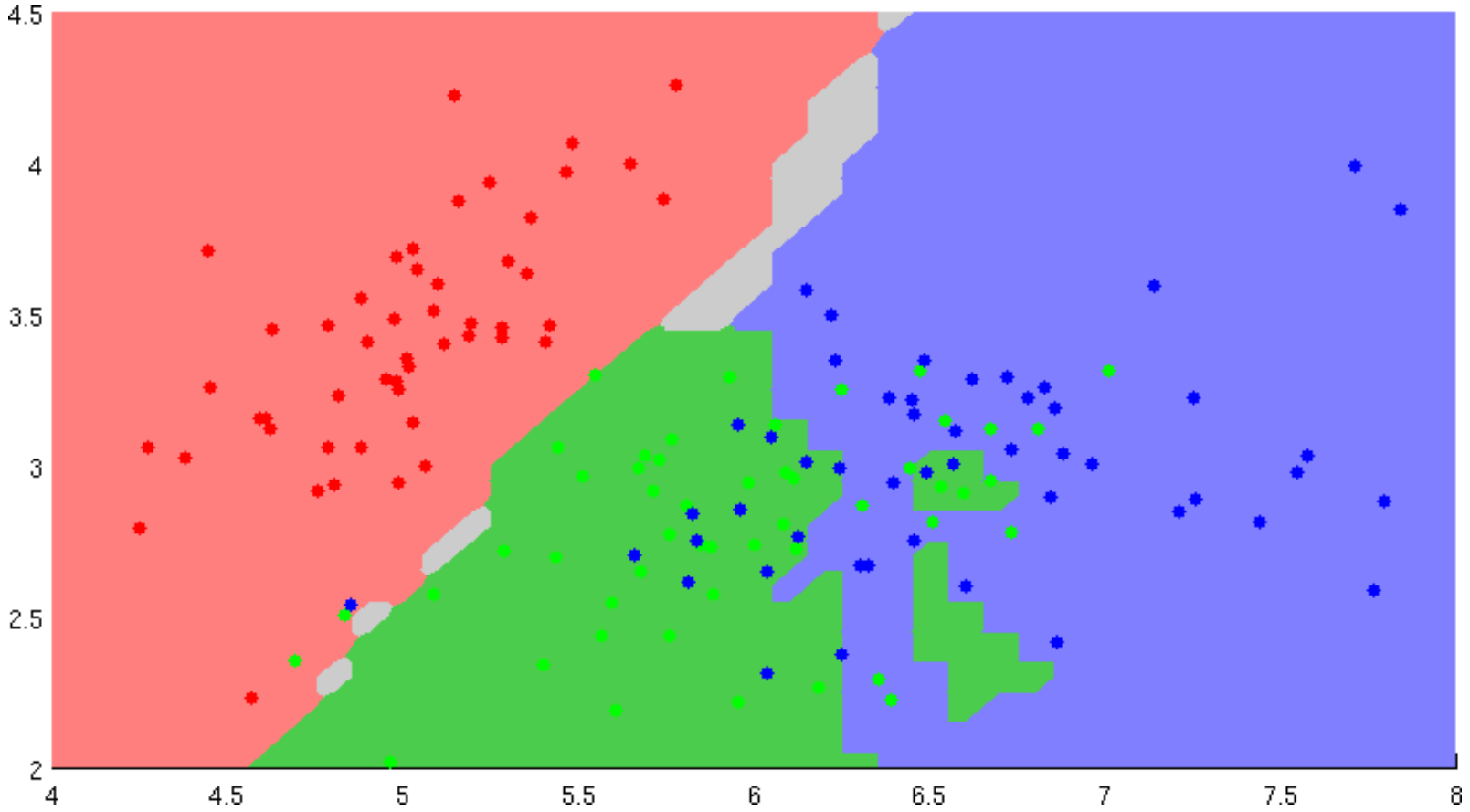
Effects of different K's (cont.)

Here are the decision areas for $K = 5$.



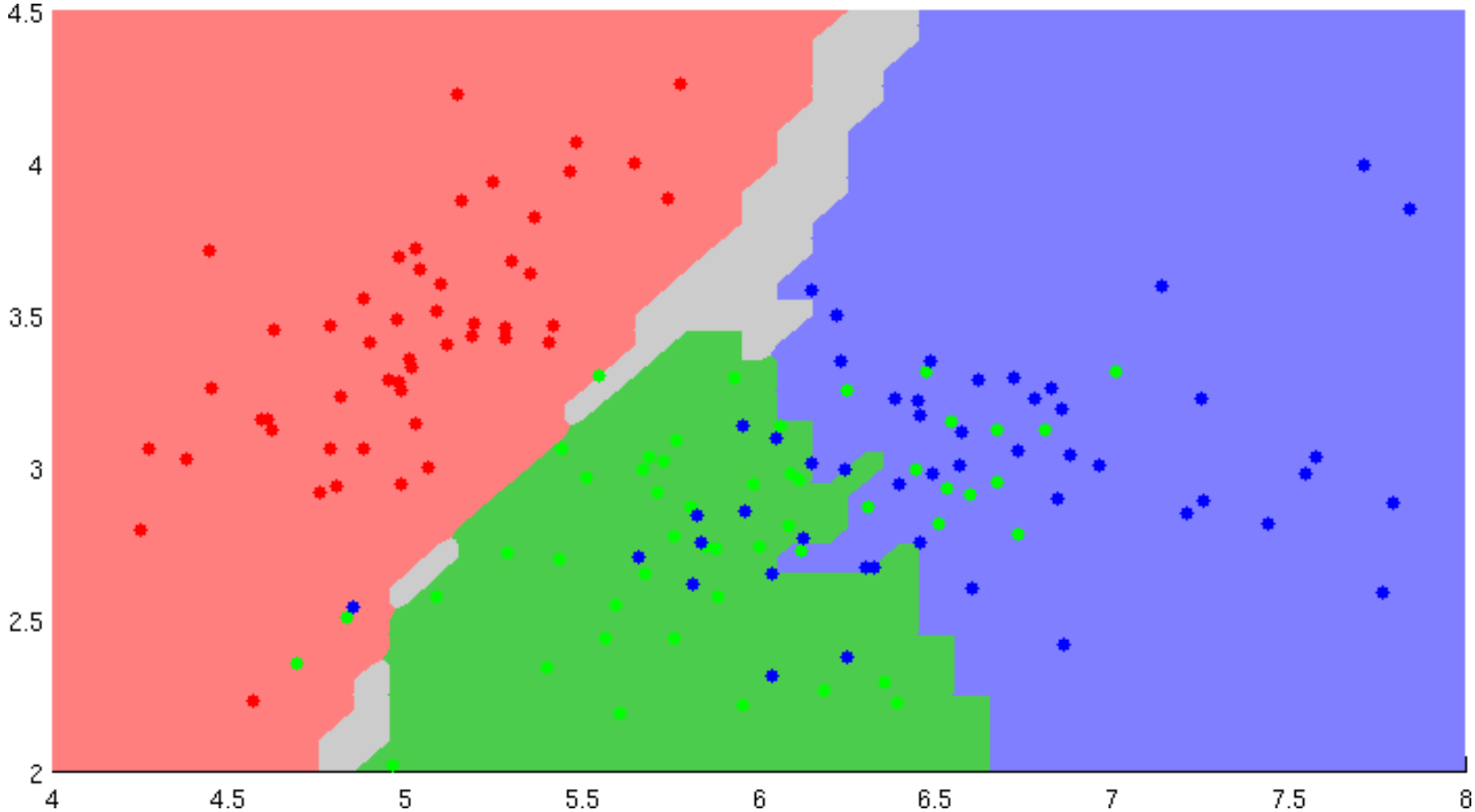
Effects of different K's (cont.)

Here are the decision areas for $K = 15$.



Effects of different K's (cont.)

Here are the decision areas for $K = 25$.



It can be seen that as K gets larger, the "holes" start to fill-in, but the grey areas grow bigger.

The K-nn algorithm with exhaustive search

Require: Training data $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ and a testing point $\mathbf{z}$. Both $\mathbf{x}_i$ and $\mathbf{z}$ are of M-dimensions. Desired value of K.

1: $nndis[k] \longleftarrow \infty$ (or any large number) for $k = 1, \ldots, K$
2: **for** i=1,2,...,N **do**
3: $d \longleftarrow 0$
4: **for** j=1,2,...,M **do**
5: $d \longleftarrow d + (\mathbf{z}[j] - \mathbf{x}_i[j])^2$
6: **end for**
7: **for** k=1,2,...,K **do**
8: **if** $d < nndis[k]$ **then**
9: **for** $l = K-1, K-2, \ldots, k$ **do**
10: $nndis[l+1] \longleftarrow nndis[l]$
11: $nn[l+1] \longleftarrow nn[l]$
12: $nnlab[l+1] \longleftarrow nnlab[l]$
13: **end for**
14: $nndis[k] \longleftarrow d$
15: $nn[k] \longleftarrow \mathbf{x}_i$
16: $nnlab[k] \longleftarrow y_i$
17: **break**
18: **end if**
19: **end for**
20: **end for**
21: **return** The mode value of $nnlab$.

The K-nn algorithm with exhaustive search (cont.)

Lines 3–6: Distance computation.

Lines 7–18: Updating the K nearest neighbours found so far.

Note that the algorithm is actually computing the **squared** Euclidean distance $d(\mathbf{z}, \mathbf{x})^2$. This does not affect the results at all since

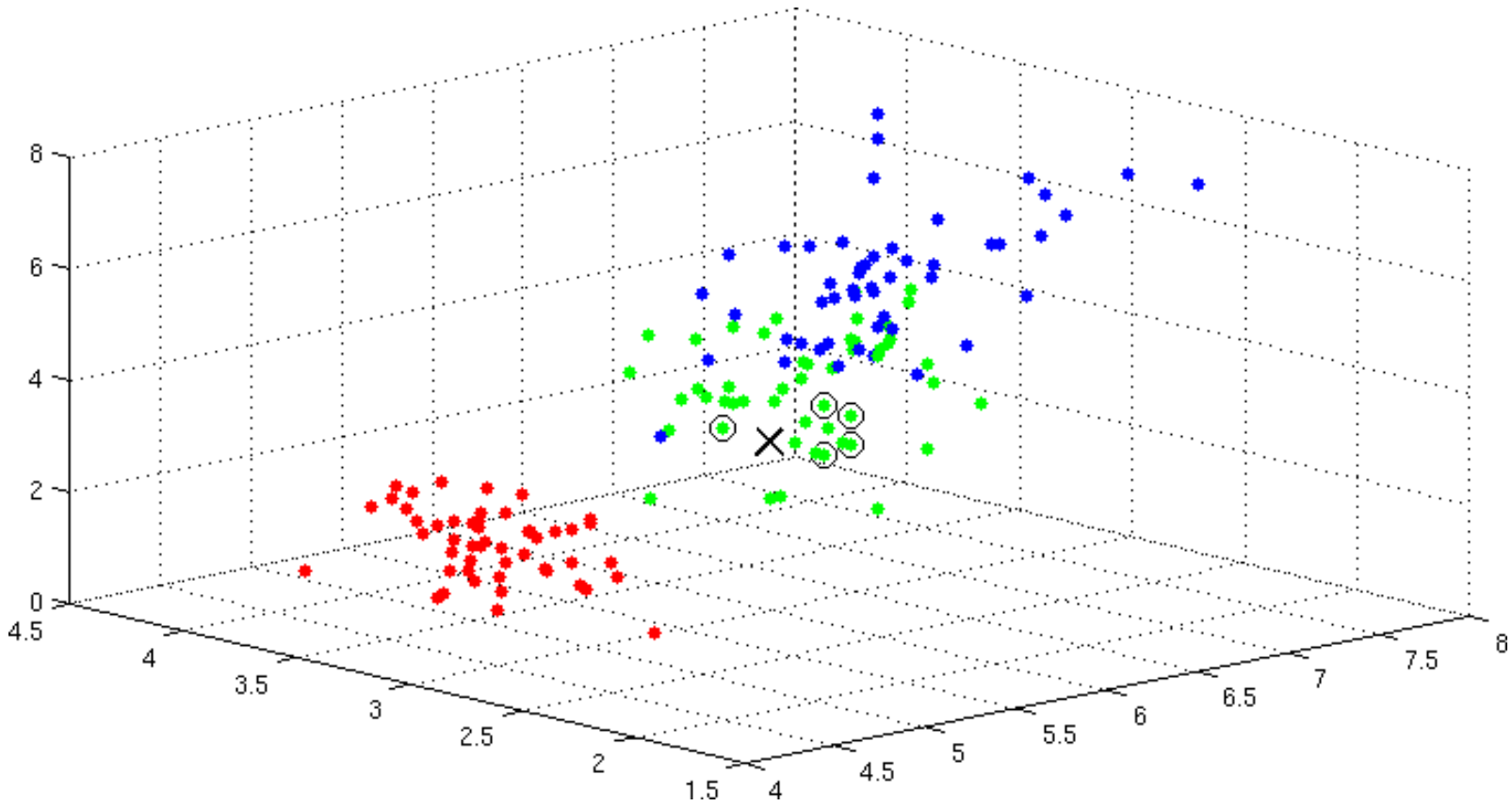
$$d(\mathbf{z}, \mathbf{x}_1) \leq d(\mathbf{z}, \mathbf{x}_2) \implies d(\mathbf{z}, \mathbf{x}_1)^2 \leq d(\mathbf{z}, \mathbf{x}_2)^2$$

Note that the algorithm is **exhaustively** computing the distance between $\mathbf{z}$ and all the other points in the training data. This might not be efficient (as we shall see later).

It is also interesting to note that K-nn is a "learning algorithm" that actually **does not involve learning/training**. Once a set of labelled training data is available you can start testing a new point.

The K-nn algorithm with exhaustive search (cont.)

Note also that the algorithm is applicable to data in any number of dimensions. Here's an example in 3 dimensions:

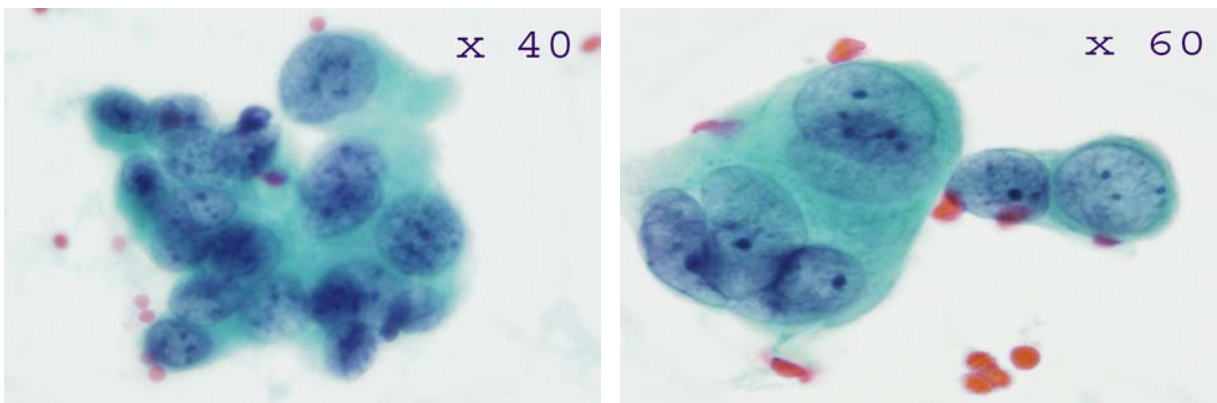


Observe that the classes are more separable with an additional feature (petal length).

Breast cancer detection

One major application of statistical learning techniques is in medical diagnosis, e.g. breast cancer detection.

One of the ways to check for breast cancer is to retrieve cells just at the bottom of the skin and to examine the digital image of the specimen under a microscope:



Several attributes of the cells (e.g. thickness, uniformity of size) are then measured and used as features to distinguish cells corresponding to benign and malignant tumours.

Breast cancer detection (cont.)

For example, the 9 **attributes** or **features** below can be measured from each cell:

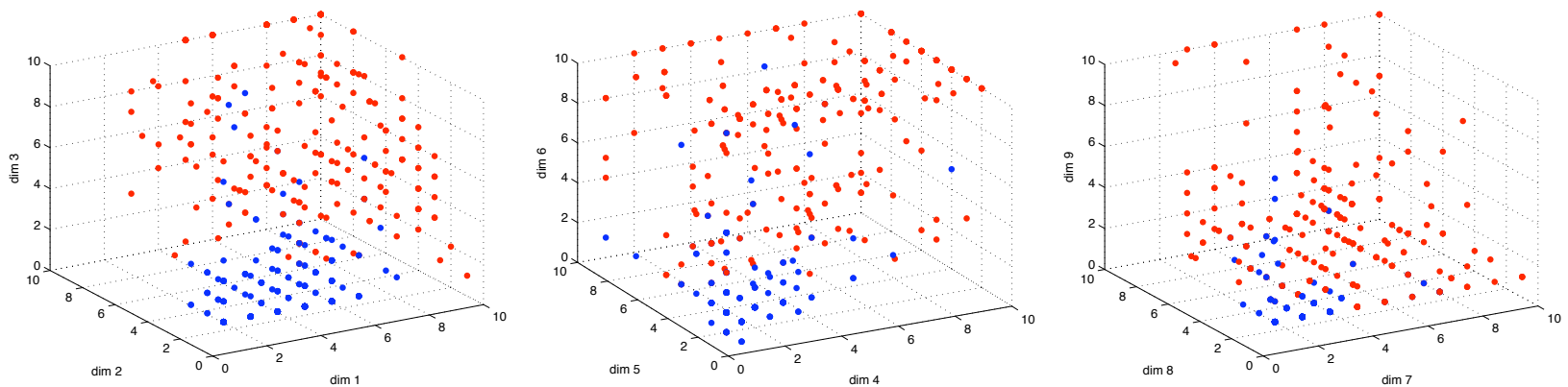
1. Clump Thickness
2. Uniformity of Cell Size
3. Uniformity of Cell Shape
4. Marginal Adhesion
5. Single Epithelial Cell Size
6. Bare Nuclei
7. Bland Chromatin
8. Normal Nucleoli
9. Mitoses

This produces a training set $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ where each $\mathbf{x}_i$ is a 9-dimensional vector of numeric values, while each y_i can take the nominal value benign (negative or 0) or malignant (positive or 1).

Breast cancer detection (cont.)

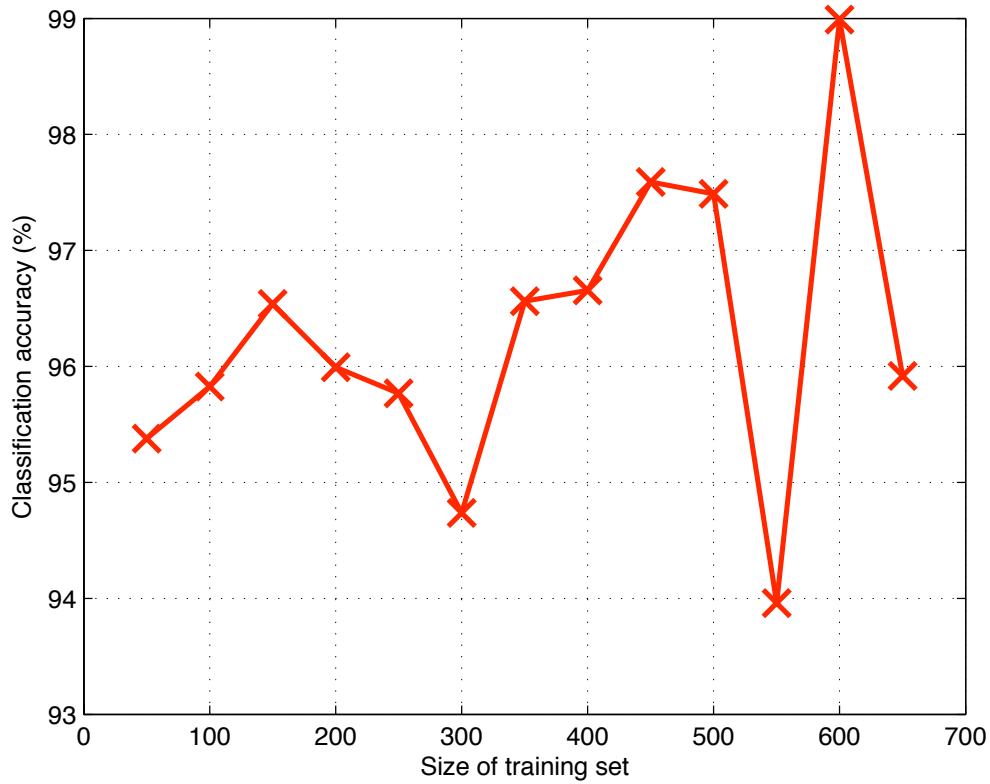
A well known breast cancer detection data is from the University of Wisconsin, where a total of 699 cases of real-life people are included. Each case contains measurements of the 9 features.

We can plot the data 3 features at once in the following 3D plots, where red is malignant while blue is benign:



Breast cancer detection (cont.)

Learning curve for the Wisconsin data with $K = 5$ for K-nn:



Concluding remarks

Knn is one of the simplest methods for supervised learning. Given a classification task, one should test with Knn first as the **baseline**, before attempting more complicated methods.

However **computational efficiency is an issue**. Among the 3 constants N, M and K that affect the total running time of the algorithm, N and M (the number of training data and the dimension of the data) usually have the largest effects.

In many practical problems N can be massive (possibly in the range of millions), while M can also be large (in the range of millions in real world training sets).