Multivariate Distances

22/05

In the last 2 lectures, we will discuss unsupervised learning again. We will talk about algorithms for **clustering** data. Clusters are collections of points which are close to each other, so we need a notion of “close”.

Unfortunately, it is not obvious what the appropriate notion of “close” might be. So far, we have been implicitly using **Euclidean Distance**.

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + \ \ldots + (x_n - y_n)^2}$$

We have already seen that PCA can go wrong if you don’t scale the variables. The same thing happens with Euclidean distance:

	(100, 1)	(102, 10)	(110, 1)
(100, 1)	0	sqrt(85)	sqrt(100)
(102, 10)		0	sqrt(145)
(110, 1)			0

(110, 1) ought to be closer to (100, 1) than (102, 10). We can deal with this by scaling the variables.

$$d(x, y) = \sqrt{(\frac{x_1 - y_1}{\sigma_1})^2 + \ldots + (\frac{x_n - y_n}{\sigma_n})^2}$$

where the σ are the standard deviations of the individual variables in the data set.

This works, but it doesn't take account of correlations between variables. We do not want to say that two things are very different if they differ slightly in many different aspects, all of which are similar to one another.

This problem can be overcome by using **Mahalanobis Distance**.

$$d(x, y) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})'\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{y})}$$

where Σ^{-1} is the inverse of the (sample) variance-covariance matrix of the data. *[Question: where have you already seen it in this course?] Note when the variance-covariance matrix is diagonal, you get the same distance as before.*

This is a *statistical distance*; it only makes sense to talk about the Mahalanobis distance between two points in the context of them being in a data set.

Discrete variables

Example:

petal colour	*sepal colour*	*leaf shape*
orange	orange	acute
red	orange	oval
blue	blue	oval

simple distance: count 3–(how many features in common). *[But problem: isn't orange more similar to red than blue? Is this really a categorical variable?]*

Presence-absence data

Site	Flounder	Halibut	Sole	Brill	Turbot	Plaice	Dab
A	1	0	1	0	0	0	1
B	1	0	0	0	0	0	1
C	1	1	1	0	1	1	0
D	0	1	1	1	1	1	0
E	0	1	0	0	0	0	0

	B1	B0
A1	2	1
A0	0	4

Simple matching: (2+4)/7
Dice-Sorensen: 2*2/(2*2+1+0)
Jaccard: 2/(2+0+1)

Measures of **similarity** ; distance can be obtained in an appropriate way, e.g. via 1-x.

Mixed data

e.g. Hair colour, eye colour, height, weight, BMI.

Distance should be some combination of categorical + continuous variables, but it is very hard to see how to combine them.

In practice, different subjects (biology, botany, archaeology…) have their preferred notions of distance to use in particular situations. There is no simple recipe for a notion of distance that always works.

Nearest Neighbours

One reason why distance is important is that it can be used to decide which points are close together. This gives rise to a classification technique called **nearest neighbours.** Given a test instance, simply find the training instance which is closest to it, and classify it into the appropriate class.

Elaboration: *k*-nearest neighbours (knn): find the k closest training instances, and classify into the majority class.

Example:

	Weight	Ph
Apple	29	3.1
Apple	26	3.4
Orange	25	3.0
Orange	22	3.0
Pear	30	3.6
Pear	29	3.2
Pear	20	3.4

Ex: Use 1-nn to classify a fruit with weight = 26 and Ph = 3.2 using Mahalanobis distance

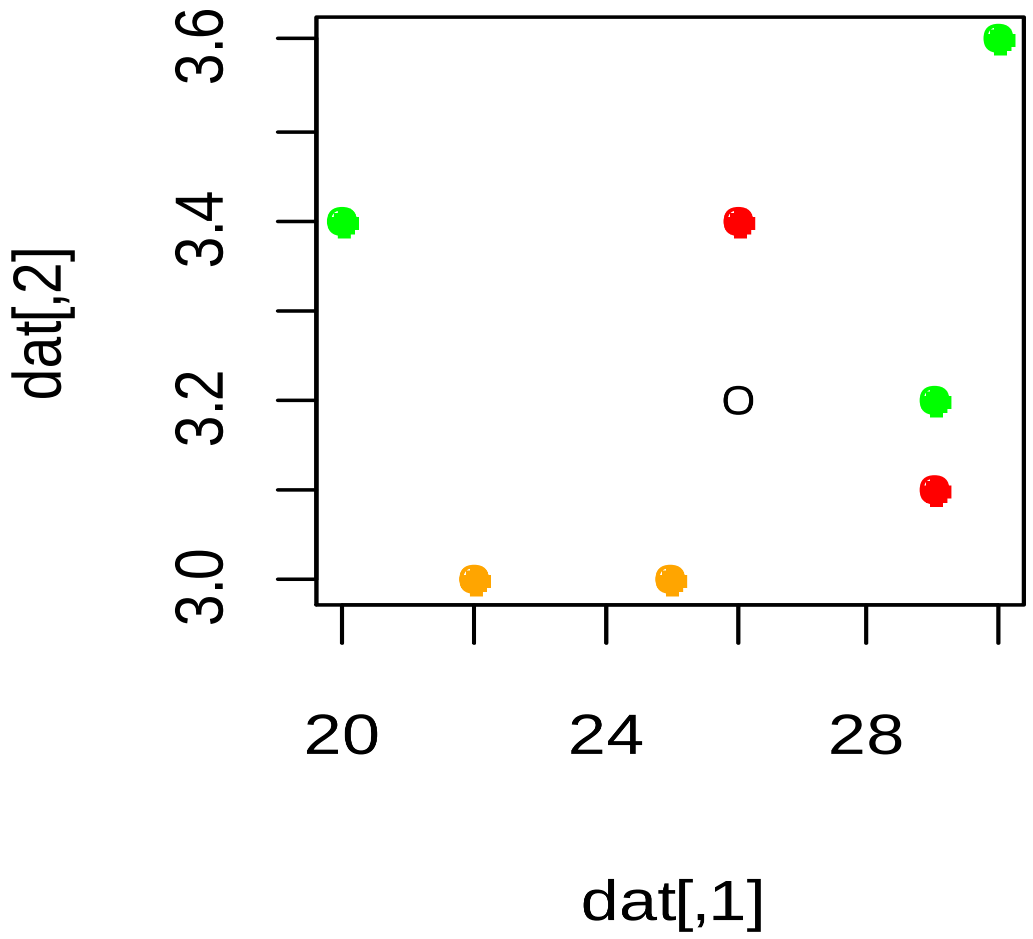
```
> V
        [,1]      [,2]
[1,] 86.857143 1.1428571
[2,]  1.142857 0.3171429
> solve(V)
           [,1]        [,2]
[1,]  0.01208624 -0.04355401
[2,] -0.04355401  3.31010453
```

Distances of new example from training data:

```
0.4098897 0.3638739 0.3564671 0.5060608 0.7639527 0.3298123
0.8197794
```

Answer: it’s closest to example 6 and would be classified as **pear** (intuitively reasonable.) But if we used 3-nn, it would be a tie between the 3 classes, and 4-nn would give **apple**.

Here, we see one of the drawbacks of knn; lengthy calculations are required.


3.6
3.4
3.2
3.0
dat[,2]
20
24
28
dat[,1]
o

- knn is easy to understand.
- if the notion of distance is chosen wisely, the error of knn is no more than twice the Bayes rate. This can be used to estimate the Bayes rate.

- Intensive computation is required to apply knn.
- All of the training data needs to be stored in order to classify new observations.
- You still need to choose the value of k. (See the example.)
- knn breaks down in high dimensions (the **curse of dimensionality**.)