6CCS3ML1 (Machine Learning)

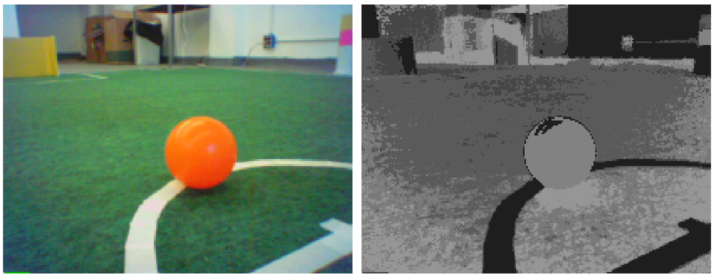
Tutorial 01 — Answers

(Version 1.1)

1. I get to pick examples from the slides.

   (a) The Iris example.
   Just as in the set of questions, the features are measurements that describe each flower (petal area and sepal area, for example), and the task is to identify which species a given Iris flower comes from.

   (b) The image processing example below.

   Here the elements are just pixels, so each is a set of three numbers, reporting the R, G and B values of the pixel (in this case numbers between 0 and 255). The code that generated these examples looked to group into clusters pixels that were within some increment of each other.
   The picture on the right is the result of some post-processing after the clustering was done — each pixel was replaced by the average of the cluster. (The image is grey scale because of laziness on my part — it was much easier to generate grey-scale output than full colour output.)
   The task here is to just find similar pixels in the image. It just happens, because of the strong colours, that these correspond nicely to objects.

2. Again I'll use examples from the lecture.

   (a) An example of binary classification is the breast cancer dataset which we used to talk about performance. The features are elements that can be distinguished from images of a potential tumour such as: radius, texture, perimeter, area and smoothness. The aim of the learning task is to distinguish between malignant and benign growths.

   (b) For multiclass classification I get to use the Iris datset again.

   (c) The image processing example is an example of clustering.

   (d) For regression, a nice example is credit scoring. A number of numerical measures, like income and a record of past missed payments, are fed into a regression model, and this is used to create a numerical score. (That score is then used to make a binary decision — that is whether someone is considered to be a bad risk or not.)
   This is also a good example of a model that can be a “weapon of math destruction”, because some features that have been routinely used in some credit scoring models (for example including post code) have the potential to discriminate against applicants in ways that can cause great harm to their lives.

3. (a) There is no training. Usually training classifier involves processing the training set in some way. In a nearest-neighbour classifier we use the training set raw.

   (b) This is the method on the slides:

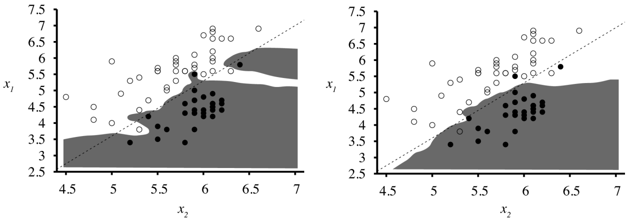
   - The k elements of the training set that are nearest to the new example are identified.
   - The number of these in each class C_1, C_2, ... are counted, call these n_1, n_2 etc.
   - The probability of the new example being in each class is then:

$$
\begin{aligned}
p(C_1) &= \frac{n_1}{k} \\
p(C_2) &= \frac{n_2}{k} \\
&\vdots
\end{aligned}
$$

   - The new example is then assigned to the class with the highest probability.

   (c) Typical behaviour is like the one below:

   - A k-nearest neighbour example:



(Russell & Norvig)

   - $k = 1$ and $k = 5$.
   - $k = 1$ probably overfits here.

   The decision boundary for a 1-NN classifier will tend to hug the examples quite tightly, and so it may well be the case that the 1-NN overfits the data. If this is the case, then a 5-NN classifier will tend to give better performance on test data.

   (a) There are 15 correctly classified examples and 20 examples in total. Thus:

$$
\begin{aligned}
accuracy &= \frac{15}{20} \\
&= 0.75
\end{aligned}
$$

   (b) For the confusion matrix we need to compute the number of true and false positives (TP and FP) and true and false negatives (TN and FN).

   We take 1 to be positive and 0 to be negative. Thus E01 is a true positive, E03 is a false positive, E05 is a true negative and E07 is a false negative. Overall:

$$
\begin{aligned}
TP &= 8 \\
TN &= 7 \\
FP &= 2 \\
FN &= 3
\end{aligned}
$$

   The confusion matrix is then:

		actual	
		yes	**no**
predicted	**yes**	8	2
	no	3	7

(c) Since the largest numbers are along the diagonal, the confusion matrix suggests that the classifer does a good job in the sense that it mainly classifies correctly. The similarity between the off-diagonals suggests that the classifier is pretty balanced – it does not make significantly more mistakes when it classifies positive examples than when it classifies negative examples.

(d)

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ &= \frac{8}{8+2} \\ &= 0.8 \end{aligned}$$

(e)

$$\begin{aligned} recall &= \frac{TP}{TP + FN} \\ &= \frac{8}{8+3} \\ &= 0.73 \end{aligned}$$

(f)

$$\begin{aligned} F_1 &= 2 \times \left(\frac{precision \times recall}{precision + recall} \right) \\ &= \frac{0.8 \times 0.73}{0.8 + 0.73} \\ &= 0.76 \end{aligned}$$

With this example you can see how the F_1 score sits between the precision and recall.

4. (a) For the instance with $x_1 = 3$ and $x_2 = 1$, the distance to each example using Manhattan distance is:

Instance	Class	Distance
X_1	C_1	2
X_2	C_1	2
X_3	C_1	4
X_4	C_1	2
X_5	C_2	7
X_6	C_2	4
X_7	C_2	5
X_8	C_2	3
X_9	C_2	5

The three nearest points are thus X_1, X_2 and X_4 all of which are in C_1, so:

$$P(C_1) = 1, P(C_2) = 0$$

and this example is classified as $C_!$.

(b) For the instance with $x_1 = 4$ and $x_2 = 5$, distances are:

Instance	Class	Distance
X_1	C_1	7
X_2	C_1	5
X_3	C_1	5
X_4	C_1	3
X_5	C_2	4
X_6	C_2	3
X_7	C_2	2
X_8	C_2	2
X_9	C_2	2

Thus the class is C_2.

(c) For the instance with $x_1 = 2$ and $x_2 = 3$, distances are:

Instance	Class	Distance
X_1	C_1	3
X_2	C_1	1
X_3	C_1	1
X_4	C_1	3
X_5	C_2	4
X_6	C_2	1
X_7	C_2	2
X_8	C_2	2
X_9	C_2	4

So, this time things are a bit less clear cut and:

$$P(C_1) = \frac{2}{3}, P(C_2) = \frac{1}{3}$$

so the most likely class is C_1 and this is how a 3-NN classifier would classify this instance.

(d) Finally, for the instance with $x_1 = 4$ and $x_2 = 3$, we have:

Instance	Class	Distance
X_1	C_1	5
X_2	C_1	3
X_3	C_1	3
X_4	C_1	1
X_5	C_2	6
X_6	C_2	3
X_7	C_2	4
X_8	C_2	2
X_9	C_2	2

Here:

$$P(C_1) = \frac{1}{3}, P(C_2) = \frac{2}{3}$$

and this example would be classified as C_2.

Version list

- Version 1.0, January 18th 2020.
- Version 1.1, January 8th 2021.