COMP9417 - Machine Learning
Tutorial: Nonparametric Modelling

Question 1. Expressiveness of Trees

Give decision trees to represent the following Boolean functions, where the variables `A`, `B`, `C` and `D` have values `t` or `f`, and the class value is either `True` or `False`. Can you observe any effect of the increasing complexity of the functions on the form of their expression as decision trees ?

(a) $A \wedge \neg B$

Solution:

```
A = t:
|    B = f:  True
|    B = t:  False
A = f:  False
```

(b) $A \vee [B \wedge C]$

Solution:

$A \vee [B \wedge C]$

```
A = t:  True
A = f:
|    B = f:  False
|    B = t:
|    |    C = t:  True
|    |    C = f:  False
```

(c) A XOR B

Solution:

```
A = t:
|    B = t:  False
|    B = f:  True
A = f:
|    B = t:  True
|    B = f:  False
```

(d) $[A \wedge B] \vee [C \wedge D]$

Solution:

```
A = t:
|    B = t:  True
|    B = f:
|    |    C = t:
|    |    |    D = t:  True
|    |    |    D = f:  False
|    |    C = f:  False
A = f:
|    C = t:
|    |    D = t:  True
|    |    D = f:  False
|    C = f:  False
```

Notice the *replication* effect of repeated subtrees as the target expression becomes more complex, for example, in the tree for d. This is a situation where the hypothesis class of models (here, decision trees) can fit any Boolean function, but in order to represent the function the tree may need to be very complex. This makes it hard to learn, and will require a lot data !

Question 2. Decision Tree Learning

(a) Assume we learn a decision tree to predict class Y given attributes A, B and C from the following training set, with no pruning.

A	B	C	Y
0	0	0	0
0	0	1	0
0	0	1	0
0	1	0	0
0	1	1	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	0	1
1	1	1	0
1	1	1	1

What would be the training set error for this dataset ? Express your answer as the number of examples out of twelve that would be misclassified.

Solution:
2. There are two pairs of examples with the same values for attributes A, B and C but a different (contradictory) value for class Y. One example from each of these pairs will always be misclassified (noise).

(b) One nice feature of decision tree learners is that they can learn trees to do *multi-class* classification, i.e., where the problem is to learn to classify each instance into exactly one of $k > 2$ classes.

Suppose a decision tree is to be learned on an arbitrary set of data where each instance has a discrete class value in one of $k > 2$ classes. What is the maximum training set error, expressed as a fraction, that any dataset could have ?

Solution:
First consider the case where all k class values are evenly distributed, so there are $\frac{1}{k}$ examples of each class in the training set. In the worst case a decision tree can be learned that predicts one of the k classes for all examples. This will get $\frac{1}{k}$ of the training set correct, and make $1 - \frac{1}{k} = \frac{k-1}{k}$ mistakes as a fraction of the training set.
If any one class has more than $\frac{1}{k}$ examples then the worst case decision tree is guaranteed to predict that class, which will reduce the error since that class now represents more than $\frac{1}{k}$ of the training set.

Question 3. Linear Smoothing

In the lab this week we introduced linear smoothing, also known as kernel smoothing, and we implement it from scratch and apply it to a simulated dataset. The following figure is taken from The Elements of Statistical Learning by Hastie, Tibshirani and Friedman and is an excellent portrayal of the linear smoother at work. The data are simulated from the following data generating process:

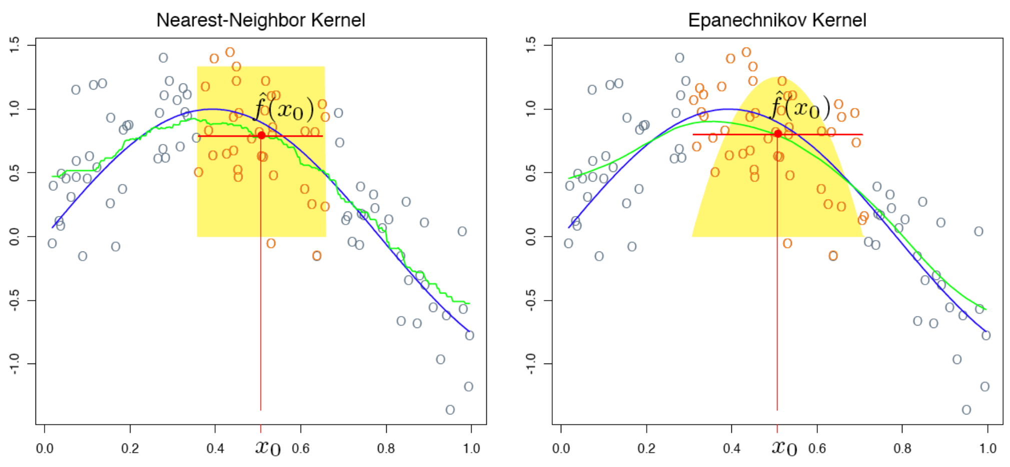
$$y = \sin(4x) + \epsilon, \qquad \epsilon \sim N(0, 1/3).$$

On the left, we see the 'nearest-neighbour' kernel at work, which we refer to as the boxcar kernel in the lab, and on the right we see the Epanechnikov kernel, also introduced in the lab. We include their definitions here:

$$\begin{aligned} K(u) &= \mathbf{1}\{|u| \leq 1/2\} && \text{box-car kernel} \\ K(u) &= \mathbf{1}\{|u| \leq 1\}\frac{3}{4}(1-u^2) && \text{Epanechnikov kernel} \end{aligned}$$

Recall also from the lab that a linear smoother prediction takes the form:

$$\hat{f}(x_0) = \sum_{i=1}^{n} \frac{K\left(\frac{\|X_i - x_0\|_2}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{\|X_j - x_0\|_2}{h}\right)} Y_i.$$



Review the lab Linear smoothing section and then write down a few lines describing what is happening in the figure. Be sure to describe in detail what each of the following represents:

(i) the blue curve

(ii) the black scatter

(iii) the red scatter

(iv) the yellow region

(v) the horizontal red line

(vi) the red point on the horizontal red line

(vii) the green curve

Solution:

(i) the blue curve: this is the true function $f(x) = \sin(4x)$ used in the data generating process.

(ii) the black scatter: these are the sampled points from the true function with added noise. The kernels chosen only look at the points that satisfy a particular condition (they need to be near enough to the query point x_0). Specifically, in the box-car kernel, the numerator is 1 if $\|X_i - x_0\|_2/h \leq 0.5$ and it is zero otherwise. Similarly, in the Epanechnikov case, the numerator is non-zero only if $\|X_i - x_0\|_2/h \leq 1$. This is just a smoothed version of kNN regression.

(iii) the red scatter These are the points that meet the conditons of the kernel, and therefore are used in estimating the prediction of the input point x_0. In other words, the black scatter has no contribution to the prediction $\hat{f}(x_0)$.

(iv) the yellow region This shows the important distinction between the kernels as it represents the weights assigned to the red points. In the left figure, the weight assigned to all red points is uniform, so nearby points and far away points (from x_0) all contribute equally to $\hat{f}(x_0)$. On the right, nearby points have a higher weight (more contribution) than far away points, which seems more reasonable.

(v) the horizontal red line The red scatter is the neighbourhood of x_0 based on the kernel K in a sense, and the red horizontal line shows the side length of the largest rectangle that contains this neighbourhood.

(vi) the red point on the horizontal red line This is the prediction $\hat{f}(x_0)$.

(vii) the green curve This is the fit of running the linear smoother over all points in the domain.