MATH5743M: Statistical Learning

Dr Seppo Virtanen, School of Mathematics, University of Leeds

Semester 2: 2022

Week 8: Ensemble Learning and Collective Intelligence

Figure 1: Watch this BBC documentary section on the Wisdom of the Crowd here: https://www.youtube.com/watch?v=iOucwX7Z1HU

'In 1906, the great statistician Francis Galton observed a competition to guess the weight of an ox at a country fair. Eight hundred people entered. Galton, being the kind of man he was, ran statistical tests on the numbers. He discovered that the average guess (1,197lb) was extremely close to the actual weight (1,198lb) of the ox. This story was told by James Surowiecki, in his entertaining book The Wisdom of Crowds.' [The Parable of the Ox. John Key, Financial Times 2012]

This famous story illustrates the power of collective wisdom. Crowds acting together can be capable of intelligence that goes beyond what any individual can achieve on their own. Just look at how colonies of termites build fantastically complex nest structures that no individual could conceive of alone. Consider how many individuals can contribute to accurate predictions of the future through financial and betting markets. These are instances where the aggregation of individuals with limited information and/or simple models of the world produces a much richer collective system.

Figure 2: Francis Galton noted how collective intelligence gave an accurate estimate for the weight of a bull.

Fundamentally, all intelligence that we know of is *collective* intelligence. None of the neurons in your brain has any intelligence of its own. Likewise with the transistors on a microchip. It is only through the interactions and aggregation of these fundamental units that intelligence is produced.

Condorcet's Jury Theorem

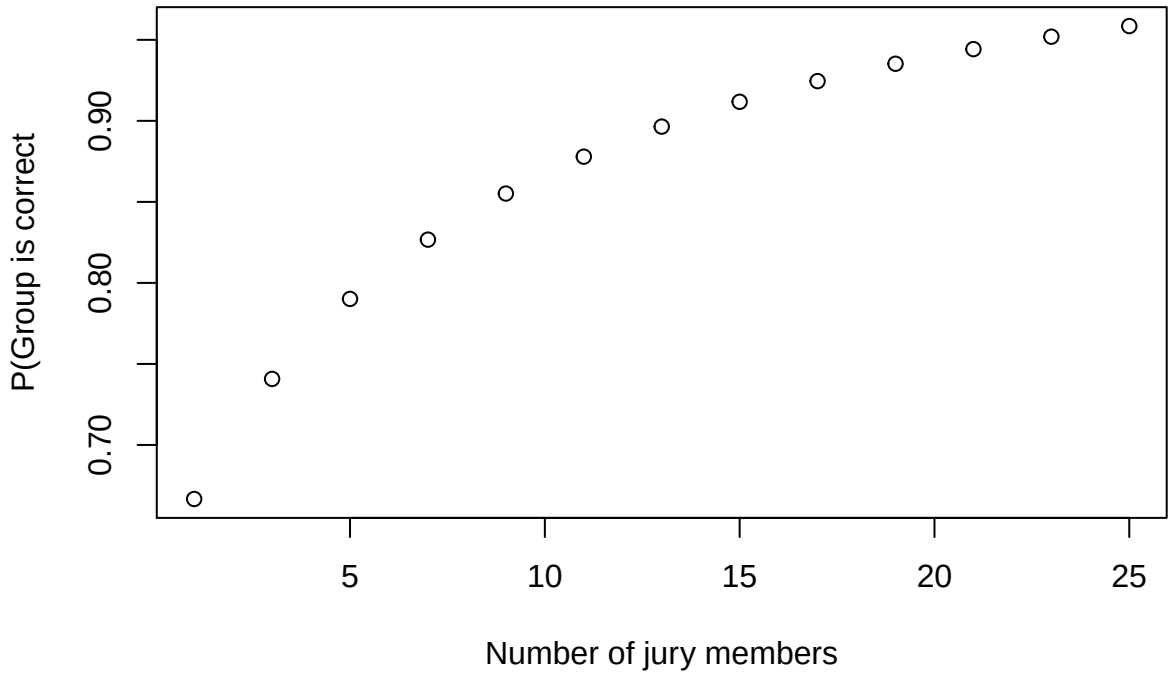
Magna Carta, the fundamental document of British democracy, states that 'No free man shall be captured, and or imprisoned, or disseised of his freehold, and or of his liberties, or of his free customs, or be outlawed, or exiled, or in any way destroyed, nor will we proceed against him by force or proceed against him by arms, but by the lawful judgment of his peers, and or by the law of the land.' Over time this principle has developed into the modern trial by jury, where twelve citizens sit in judgement of the accused and decide on his/her guilt or innocence. But why do we have twelve jury members, rather than just a single individual? One mathematical reason is given by a famous theorem by Condorcet. Consider N individuals asked to decide on the truth of some proposition. Assume that each individual has a probability of being correct p, and that $p > 1/2$ (even a single individual is more likely to be correct than wrong). If these N individuals vote, and the vote of individual i is X_i, with $X_i = 1$ implying the correct decision and $X_i = 0$ the wrong decision, then the probability that the group will make the correct decision is:

$$\begin{aligned} P(\text{Group is correct}) &= P(\sum_{i=1}^{N} X_i > N/2) \\ &= P(Z > N/2),\ Z = \sum_{i=1}^{N} X_i \sim \mathcal{B}(N, p) \end{aligned} \tag{1}$$

Lets look at how the probability for the group being correct changes as we change the number of people in the jury, assuming that $p = 2/3$ – each individual has a $2/3$ chance of getting the decision correct on their own. We will only look at odd numbered juries, so that there can be no ties

```
N = seq(1, 25, 2)
p = 2/3
```

```
p_group_correct = 1 - pbinom(N/2, N, p)
plot(N, p_group_correct, xlab="Number of jury members", ylab="P(Group is correct")
```



As you can see, the chance that the group is correct increases as the number of jurors is increased. With a jury of 15 the group will get its decision correct over 90% of the time, even though each individual is only right 66% of the time. If we imagine that instead of different jurors, we had a collection of different *classifiers*, this shows how an *ensemble* of many bad classifiers can produce very accurate results.

The Netflix prize

In October 2006, Netflix (then primarily a DVD rental company) launched a competition to seek out the best way of making film recommendations to its customers. The company wanted to predict which films its customers would watch next, based on their previous viewings. A prize of $1,000,000 was offered to any one who could make predictions that were more than 10% more accurate than the company's own software. Many teams of researchers entered the competition and produced very complicated algorithms that improved on Netflix's own, but for a long time no one was able to beat this magic figure of 10%. Eventually, in 2009, two teams finally managed to get beyond this mark. Both were formed by joining together two or more previously unsuccessful teams. The improvement in prediction came as a result of combining their different methods into an ensemble of statistical models. Each statistical model was able to spot patterns in the data that other models had missed; as a result the combined prediction from all of the models put together was able to perform much better than any single model. In this way they acted like Condorcet's jury: each model had a relatively low probability of being correct itself, but the combined predictions were more accurate.

The elements of collective intelligence

1. Diversity
2. Independence

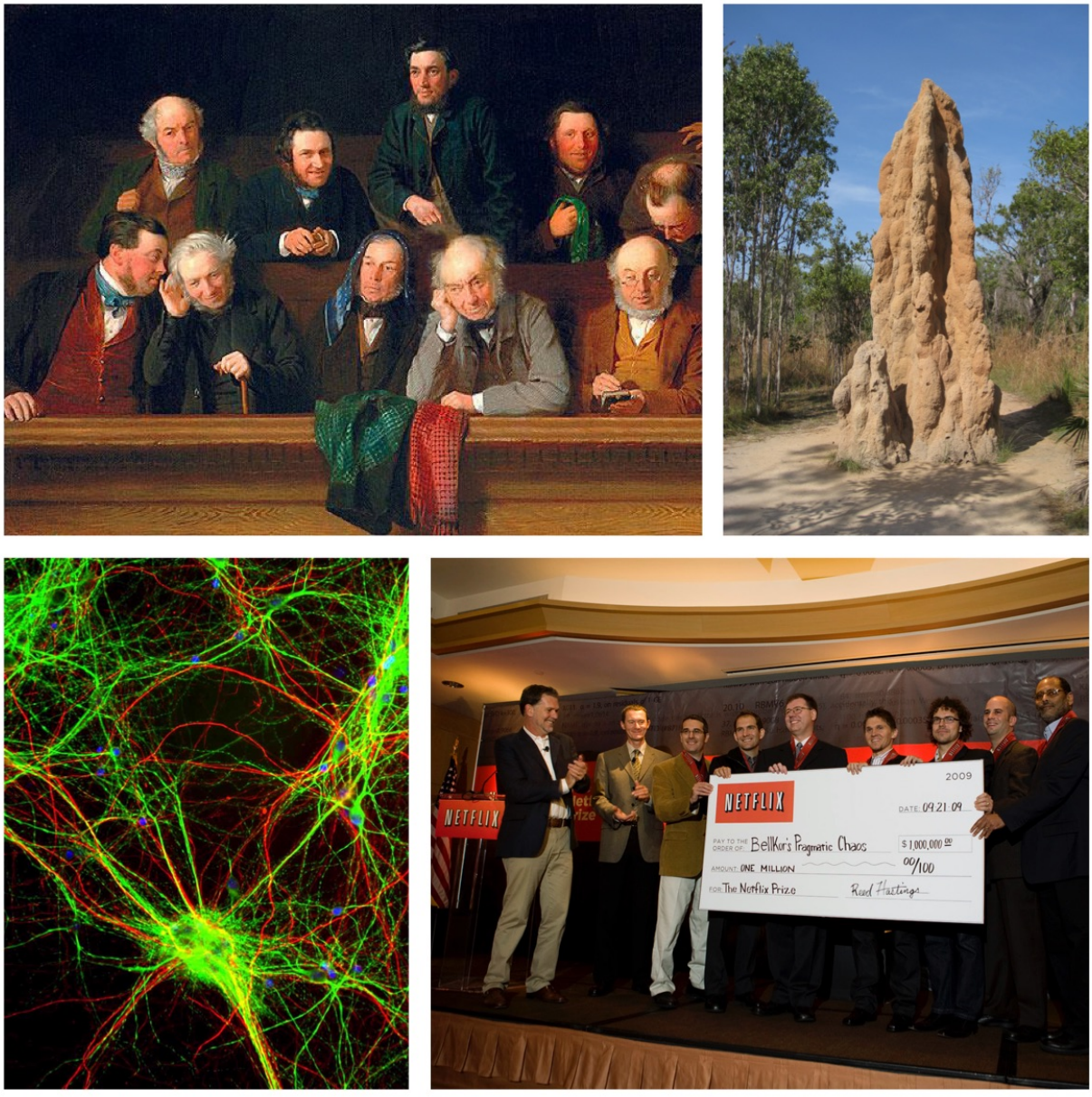


Figure 3: Examples of collective intelligence: juries, insect colonies, neurons in the brain and the ensemble of statistical models that won the Netflix prize

3. Decentralisation
4. Aggregation

Bootstrap Aggregating (BAGGING)

One powerful yet simple method for exploiting the power of ensemble learning is known as Bootstrap Aggregating (often shortened to BAGGING).

Bootstrapping

Bootstrapping is a method for generating new data sets from an original data source.

If we have n data entries, indexed as $y_1, \ldots, y_n$ and $x_1, \ldots, x_n$, we can generate a new data set by sampling from these indices uniformly with replacement. Lets see how to do this in R using the faithful data set

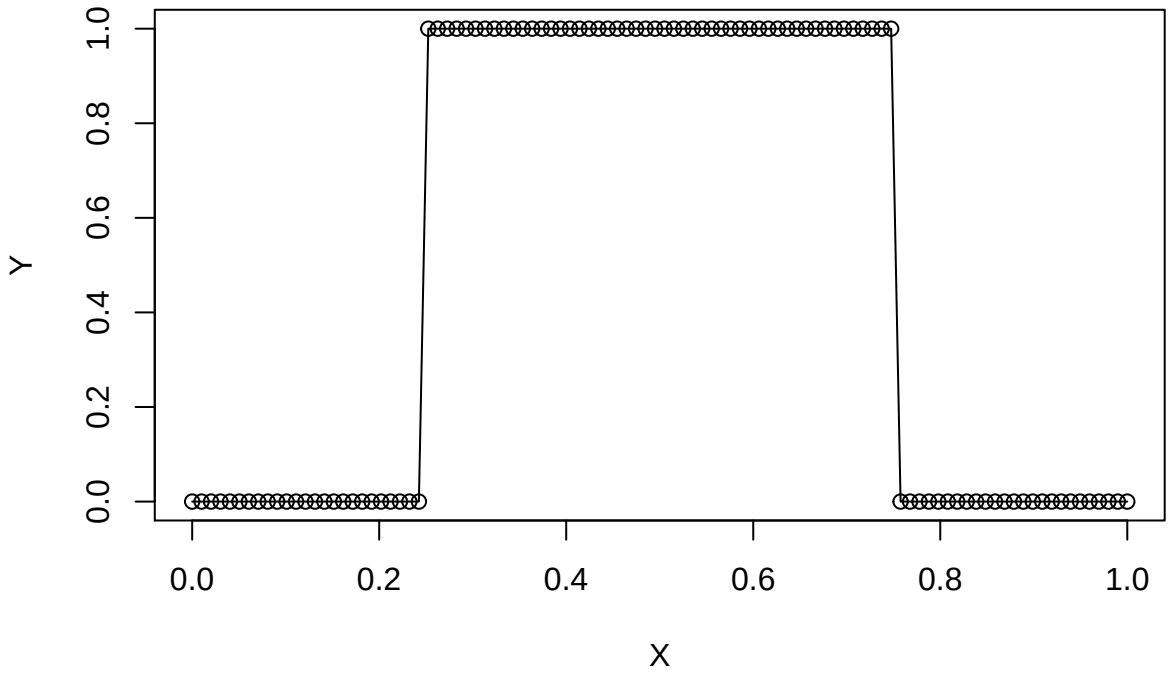
```
data = faithful
bootstrap_indices = sample(length(faithful), length(faithful), replace=TRUE)
data_bootstrap = data[bootstrap_indices, ]
```

databootstrap is now a bootstrap sample from the original data.

BAGGING of logistic regressions

There is little to be gained from BAGGING linear regressions. This is because the sum of two linear functions is another linear function. However, for non-linear functions there is much more potential. For example, the sum of many logistic functions is not in general another logistic function. Therefore we will use logistic regression to demonstrate the power of BAGGING. We'll define an underlying probability function that is impossible for a single logistic regression to capture properly. Then we'll show how summing over many logistic regression models trained on different sampled subsets of the data can outperform a single model.

```
#define a complex probability function
myfunction <- function(x){0+1*(x>0.25 & x < 0.75)}

#Generate some data
X = seq(0,1, length.out=100)
P = myfunction(X)
Y = as.numeric(runif(100)<P)

#Plot the data with the actual probability
plot(X, Y)
lines(X, P)
```




```r
#Make a data frame and split into training and test data
mydata = data.frame(Y, X)
test_idx = sample(dim(mydata)[1], 20)
test_data = mydata[test_idx, ]
train_data = mydata[-test_idx, ]

#Fit a logistic model to the training data
myModel = glm(Y ~ X, data=train_data, family=binomial)

#Make a prediction for the probability of the test data being 1
prediction = predict(myModel, newdata=test_data, type="response")

#Evaluate the predictive log-likelihood
testLL = sum(log(prediction[which(test_data$Y==1)])) +
  sum(log(1-prediction[which(test_data$Y==0)]))


#BAGGING!

#Make 1000 bootstrap samples
#Fit a logistic model to each
#Record prediction onto test data and aggregate
bsPrediction = rep(0, dim(test_data)[1])
for (i in 1:1000){
  bs_idx = sample(dim(train_data)[1], 20, replace=TRUE)
  bs_data = train_data[bs_idx,]
  bsModel = glm(Y ~ X, data=bs_data, family=binomial)
  bsPrediction = bsPrediction + predict(bsModel, newdata=test_data, type="response")/1000

}
```

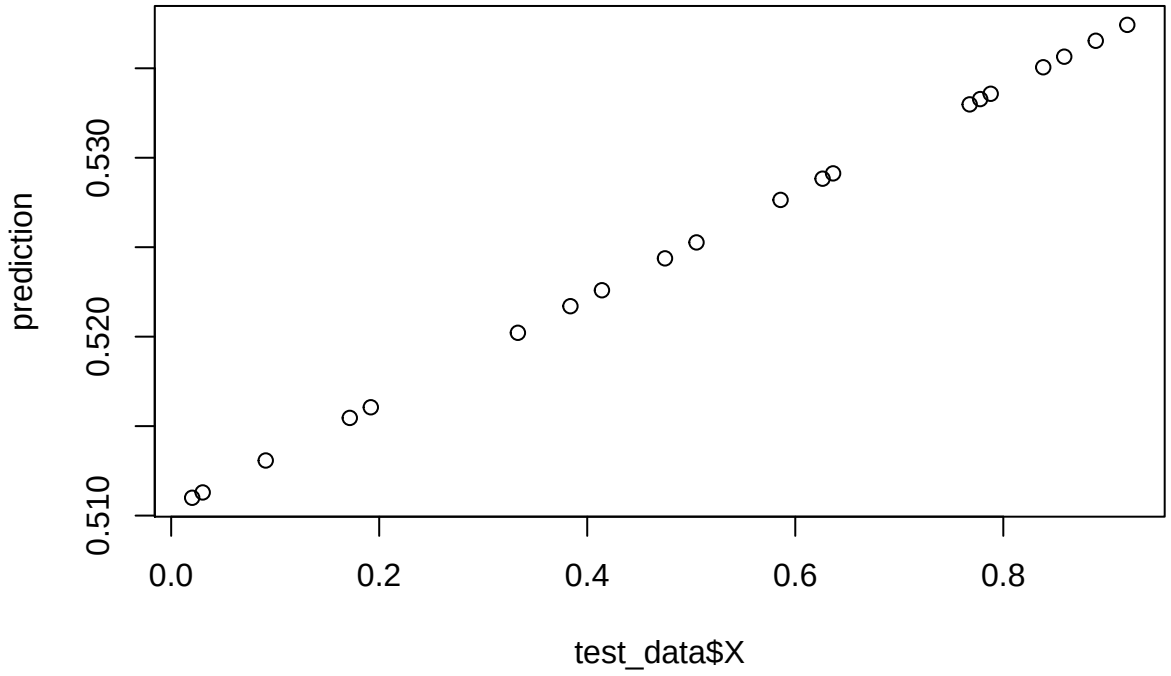
```r
#Get the predictive likelihood of the bagged model
bstestLL = sum(log(bsPrediction[which(test_data$Y==1)])) +
  sum(log(1-bsPrediction[which(test_data$Y==0)]))

#Print the results
print(testLL)
```

```
## [1] -14.11583
```

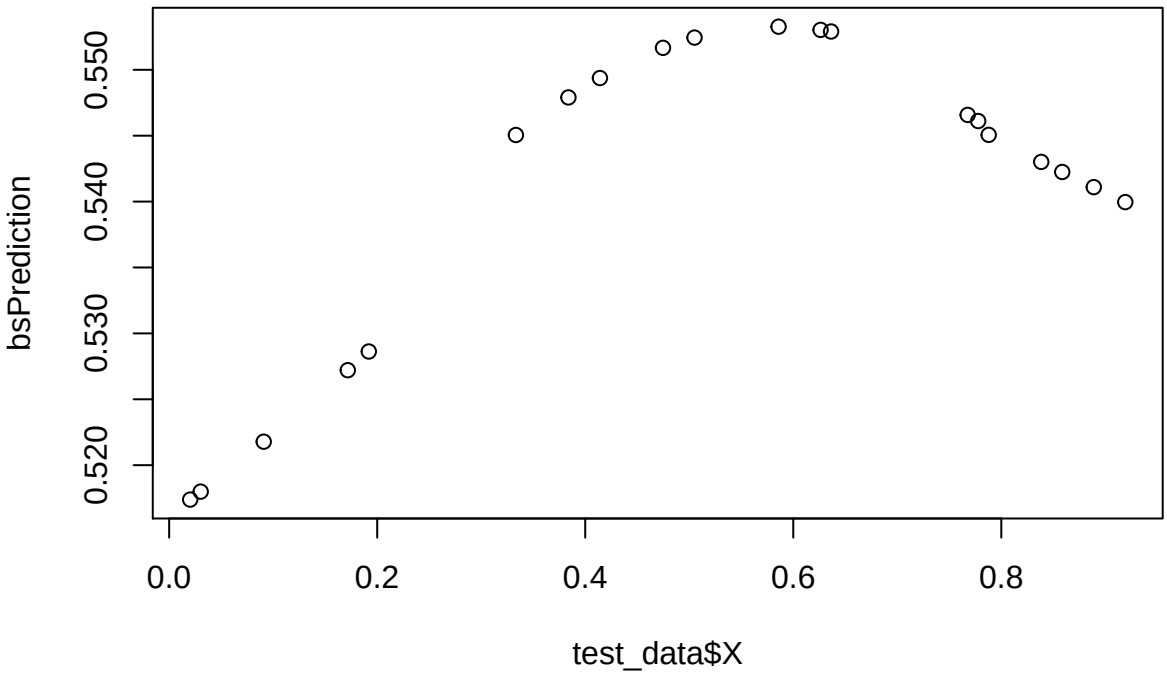
```r
print(bstestLL)
```

```
## [1] -13.95809
```

```r
plot(test_data$X, prediction)
```




```r
plot(test_data$X, bsPrediction)
```



In this case a single logistic regression cannot capture the fact that the probability is non-monotonic in X. By learning from different sampled subsets of the data, the BAGGED logistic regressions capture different features of the underlying relationship, in this case the rise in probability at $X = 0.25$ and the decrease at $X = 0.75$.

Ways to improve the performance of BAGGING

Previously, we learnt about the key principles of collective intelligence. As a reminder, these were:

- Diversity
- Independence
- Decentralisation
- Aggregation

This reminds us that if we want a collective or ensemble to perform well, we should try to increase the diversity and independence of the different models within it, and the information they can draw on. One way we can do that is by decreasing the overlap in the data each model sees the training set.

Going further

Maximising the performance of statistical ensembles is an active line of research. Algorithms known as BOOSTING aim to improve on BAGGING by choosing data that is currently not well captured by the ensemble to train the next model. Bayesian model averaging and Bayesian model combination aim to find better ways to aggregate existing models, rather than taking a simple average. My own research looks at how distributed experts (be they models or people) can be incentivised to find the best information to maximise collective accuracy. Research in this area is of great interest both academically and industrially, as companies look to utilise the computing power and time of the many computers distributed across the world.