

## *Practical 1*

### *Jumping Rivers*

#### *Building a first model*

The **jrpytensorflow** package has some concentric circle data which can be loaded with

```
import jrpytensorflow
```

```
X, y = jrpytensorflow.datasets.load_circles()
```

- Create an exploratory visualisation of the data
- Create a logistic regression model for this problem
- Compile and run a simple training routine to fit this model.
- How many predictions did you get correct?
- Assuming that your model object is called `model`, you can visualise the predicted probability space with the following code.

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
x1 = np.linspace(-1.5, 1.5, 100)
grid = np.array([(x, y) for x in x1 for y in x1])
output = np.array(model(grid))
plt.figure()
plt.pcolormesh(x1, x1, output.reshape(100, 100))
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='black')
plt.xlim([-1.5, 1.5])
plt.ylim([-1.5, 1.5])
plt.show()
```

- It is very hard to classify this dataset with logistic regression using only the input variables provided. Can you improve performance by introducing some additional features.

#### *Optional*

- Create a flexible logistic regression class that could be integrated with a `sklearn.Pipeline` for predicting a binary output