Neural Networks

COMP2002

Lauren Ansell

Introduction

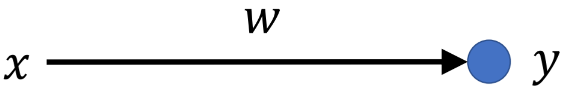
Today's topics:

- Neural networks
- Regression
- Classification
- Neural networks in Scikit

Session learning outcomes - by the end of today's lecture you will be able to:

- Explain the structure of an artificial neural network and describe the process by which one is trained
- Explain how to design a problem to be tackled with neural networks

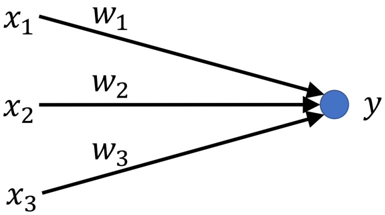
The Perceptron



$$y = f(xw)$$

- The model input is the *x* variable
- The model has a single unit (shown in blue)
- Input is connected to the unit by a weight
- Output is a function of the weighted sum of the input

Single-layer Networks

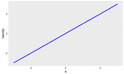


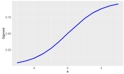
- Multiple inputs – the output is now the sum of the weighted inputs
- K stands for the number of inputs

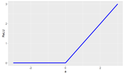
$$y = f(x_1 w_1 + x_2 w_2 + x_3 w_3) \qquad y = f\left(\sum_{k=1}^{K} x_k w_k\right) \tag{1}$$

Activation Functions – What To Use As f?

Identity - $f(a) = a$
Sigmoid - $f(a) = 1/(1 + e^{-a})$
ReLU - $f(a) = \max(0, a)$


(a) Identity


(b) Sigmoid


(c) ReLU

Choosing an activation function – consider:

- The function's affect on the learning process
- What activation function you want to apply to a specific layer of units

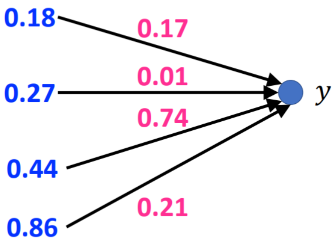
Example

$$a = (0.18 \cdot 0.17) + (0.27 \cdot 0.01) + (0.444 \cdot 0.74) + (0.86 \cdot 0.21) \quad (2)$$

$$y = 1/(1 + e^{a}) \quad (3)$$

$$y = 1/(1 + e^{0.54}) \quad (4)$$

$$y = 0.368 \quad (5)$$



How accurate is the model?

Measuring Error - Regression

To train a neural network you must identify a set of weights (and NN structure?) that minimizes an error function.

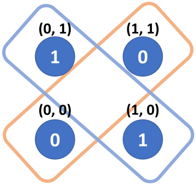
Given a set of targets $\{t_n\}_{n=1}^{N}$ and model outputs $\{y_n\}_{n=1}^{N}$ compute the mean absolute error.

$$\text{MAE} = \frac{\sum_{n=1}^{N} |t_n - y_n|}{N} \tag{6}$$

The SLP on the previous slide predicted a value of $y = 0.368$ for a target value of 0.798, so the error is $|0.368 - 0.798| = 0.43$ – the smaller this value the better.

Other error functions are available: root mean squared error, R^2 ...

SLP Can’t Process XOR

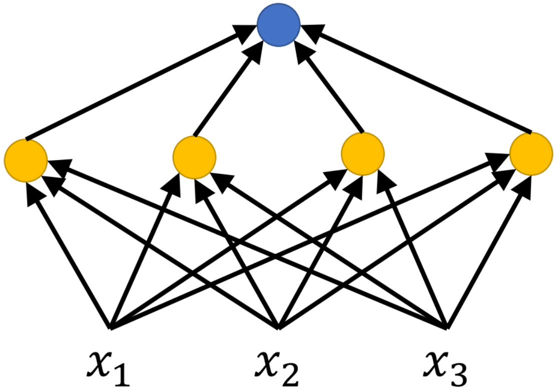


XOR
(exclusive or) is a simple logic problem
- output value depends on the two inputs.

Minsky and Papert
showed in the 1960s that this simple
problem cannot be modelled by a SLP.

Instead we need multiple
layers which can model the non-linearity.

Multi-layer Networks



Hidden layer

Training: Back Propagation

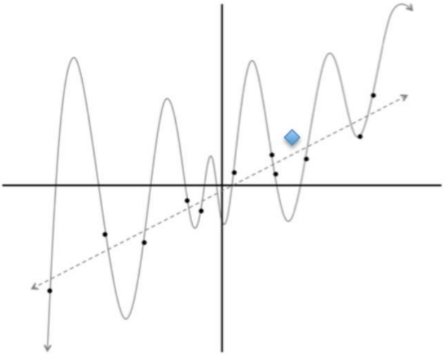
Use model error to update the network weights
Back propagation algorithm

1. Initialise all weights to random values
2. repeat
3. for all training examples do
4. Forward propagation: pass the training examples through the network to the output
5. end for
6. Calculate the network error
7. Back propagation: update each weight based on a learning term derived from the network error
8. until weights converge (or runtime elapses)

Generalisation

The model is trained to follow the training data too closely – it does not generalize to new data.

Prevent with early stopping.

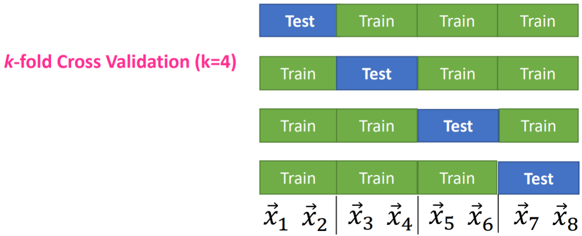


Cross Validation

Need to assess whether the model generalizes – does it work on unseen data?

Use cross-validation:

1. Train the model on p% of the data
2. Test the model on (100 – p)% of the data – this data is unseen during training



Other strategies available, e.g. leave-one-out cross validation

Regression - Example: California Housing Data

Predict the median house value in California districts.

The dataset consists of 20,640 observations.

1. House block longitude
2. House block latitude
3. Median house age
4. Total number of rooms
5. Total number of bedrooms
6. Population
7. Households
8.Median income
9. Median house value
10. Ocean proximity

Normalisation

Place all inputs on the same range.

Various strategies – in this case, all of the inputs are transformed to have the same statistical properties.

Could normalize based on maximum/minimum values to place between 0 and 1.

(d) Raw values

(e) Normalised values

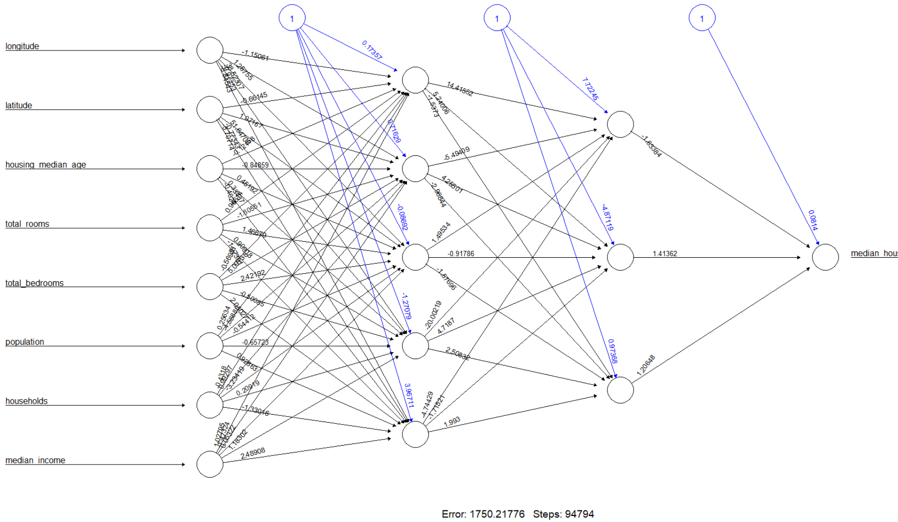
Fitting The Regression Model

Use all the variables except ocean proximity to predict the median house value.

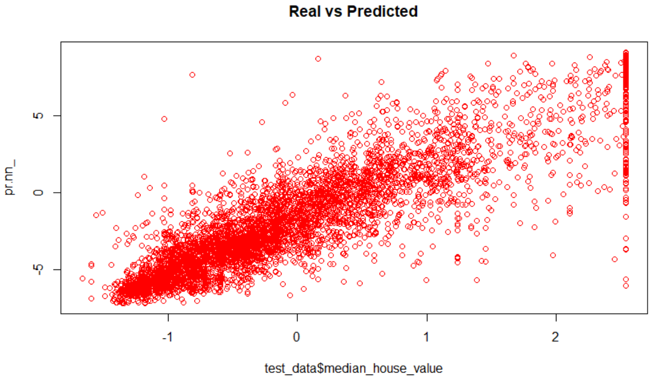
Uses the Sigmiod function as the activation function.

There are 2 hidden layers, one with 5 neurons and one with 3.

The Neural Network



Assessing The Results



We can see that the predictions made by the neural network general form a straight line.

There is a lot of variation shown in the plot- what could we do to improve the prediction?

Classification - Example: Iris Data

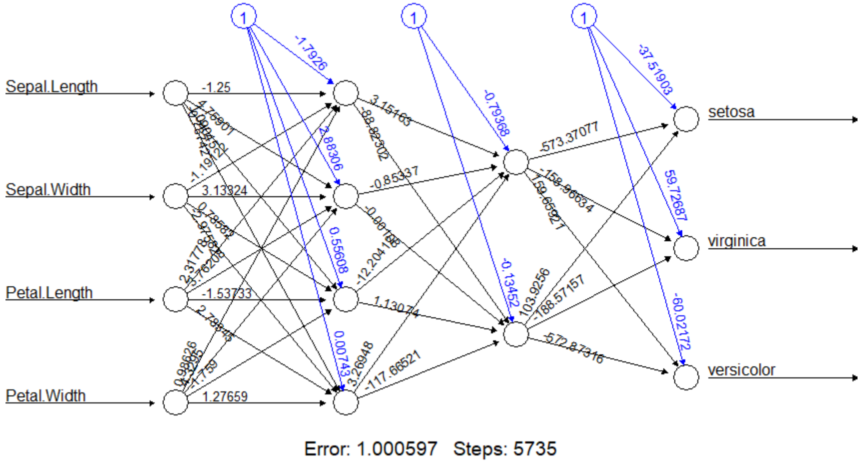
Dataset containing 150 observation that are split into 3 classes.

Use a neural network to classify the observations.

Using the Sigmoid function as the activation function.

There are 2 hidden layers, one with 4 neurons and one with 2.

The Neural Network



Assessing The Results

Actual Label	Prediction Label setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	7	0
virginica	0	0	8

Neural Networks In Scikit

```
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error

regressor = MLPRegressor()
regressor.fit(scaled, targets)
outputs = regressor.predict(scaled)
print(mean_absolute_error(targets, outputs))
```

- Scikit provides neural networks for both classification and regression
- Standard pipeline for using them (and other ML tools):
  1. Instantiate the object
  2. Train the model with **fit** method
  3. Obtain predictions with **predict** method
  4. Evaluate (e.g. **mean_absolute_error** in this case)

Neural networks

- Comprised of units connected by weights
- Must be trained