# Neural Networks

A simple neural network is made up of an input layer, hidden layers and an output layers. Our aim is to select correct weights on each edge using iterative methods.

## Introduction

A neural network attempts to replicate the structure of the brain

## Backpropogation

This is a training method, also referred to "the backward propogation of errors". To use this, we first define the following quatities

$$J(y) = (t-y)^2 \text{ the loss function}, \tag{1}$$

$$D_n(y) = \frac{dJ(y)}{dw_n} \text{ the derivative of the loss function} \tag{2}$$

We then perform the following steps for each $(x, t) \in X$

1. Pass $x$ through the neural network and obtain the output $y$
2. Obtain the new weight for each edge $w_n' = \delta w_n = -RD_n(y)$ for a learning rate $R$

## The Iris Dataset

We now train a neural network on the iris dataset

In [6]:
```
import tensorflow

from sklearn.datasets import load_iris


X, y = load_iris(as_frame=True, return_X_y=True)
```

We can now get a training and testing set for our network

In [8]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)
```

In [9]:
```
import tensorflow as tf

train = tf.data.Dataset.from_tensor_slices((X_train, y_train))
test = tf.data.Dataset.from_tensor_slices((X_test, y_test))
```

```
2023-03-16 16:01:31.974239: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No such file or directory
2023-03-16 16:01:31.974261: W tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:265] failed call to cuInit: UNKNOWN ERROR (303)
2023-03-16 16:01:31.974281: I tensorflow/compiler/xla/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (IT085374): /proc/driver/nvidia/version does not exist
2023-03-16 16:01:31.974526: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX2 AVX_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

In [10]:
```
test = test.batch(1)
```

## Creating the Model

In [12]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(10, activation=tf.nn.relu),   # hidden layer
    tf.keras.layers.Dense(10, activation=tf.nn.relu),   # hidden layer
    tf.keras.layers.Dense(3, activation=tf.nn.softmax)  # output layer
])
```

In [13]:
```
model.compile(
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
)
```

In [16]:
```
model.fit(
    train,
    validation_data=test,
    epochs=10
)
```

```
Epoch 1/10
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[16], line 1
----> 1 model.fit(
      2     train,
      3     validation_data=test,
      4     epochs=10
      5 )

File ~/Documents/COMPASS/First Year/TB2/Statistical Computing/Neural Networ
ks/tfvenv/lib/python3.8/site-packages/keras/utils/traceback_utils.py:70, in
 filter_traceback.<locals>.error_handler(*args, **kwargs)
     67     filtered_tb = _process_traceback_frames(e.__traceback__)
     68     # To get the full stack trace, call:
     69     # `tf.debugging.disable_traceback_filtering()`
---> 70     raise e.with_traceback(filtered_tb) from None
     71 finally:
     72     del filtered_tb

File /tmp/__autograph_generated_file_2pc0uek.py:15, in outer_factory.<local
s>.inner_factory.<locals>.tf__train_function(iterator)
     13 try:
     14     do_return = True
---> 15     retval_ = ag__.converted_call(ag__.ld(step_function), (ag__.ld
(self), ag__.ld(iterator)), None, fscope)
     16 except:
     17     do_return = False

ValueError: in user code:

    File "/home/ac18826/Documents/COMPASS/First Year/TB2/Statistical Comput
ing/Neural Networks/tfvenv/lib/python3.8/site-packages/keras/engine/trainin
g.py", line 1249, in train_function  *
        return step_function(self, iterator)
    File "/home/ac18826/Documents/COMPASS/First Year/TB2/Statistical Comput
ing/Neural Networks/tfvenv/lib/python3.8/site-packages/keras/engine/trainin
g.py", line 1233, in step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    File "/home/ac18826/Documents/COMPASS/First Year/TB2/Statistical Comput
ing/Neural Networks/tfvenv/lib/python3.8/site-packages/keras/engine/trainin
g.py", line 1222, in run_step  **
        outputs = model.train_step(data)
    File "/home/ac18826/Documents/COMPASS/First Year/TB2/Statistical Comput
ing/Neural Networks/tfvenv/lib/python3.8/site-packages/keras/engine/trainin
g.py", line 1023, in train_step
        y_pred = self(x, training=True)
    File "/home/ac18826/Documents/COMPASS/First Year/TB2/Statistical Comput
ing/Neural Networks/tfvenv/lib/python3.8/site-packages/keras/utils/tracebac
k_utils.py", line 70, in error_handler
        raise e.with_traceback(filtered_tb) from None
    File "/home/ac18826/Documents/COMPASS/First Year/TB2/Statistical Comput
ing/Neural Networks/tfvenv/lib/python3.8/site-packages/keras/engine/input_s
pec.py", line 250, in assert_input_compatibility
        raise ValueError(

    ValueError: Exception encountered when calling layer 'sequential' (type
```

```
 Sequential).

    Input 0 of layer "dense" is incompatible with the layer: expected min_n
dim=2, found ndim=1. Full shape received: (4,)

    Call arguments received by layer 'sequential' (type Sequential):
      • inputs=tf.Tensor(shape=(4,), dtype=float64)
      • training=True
      • mask=None
```