# ETC3250/5250: Neural networks 2

## Semester 1, 2020

Professor Di Cook

Econometrics and Business Statistics
Monash University

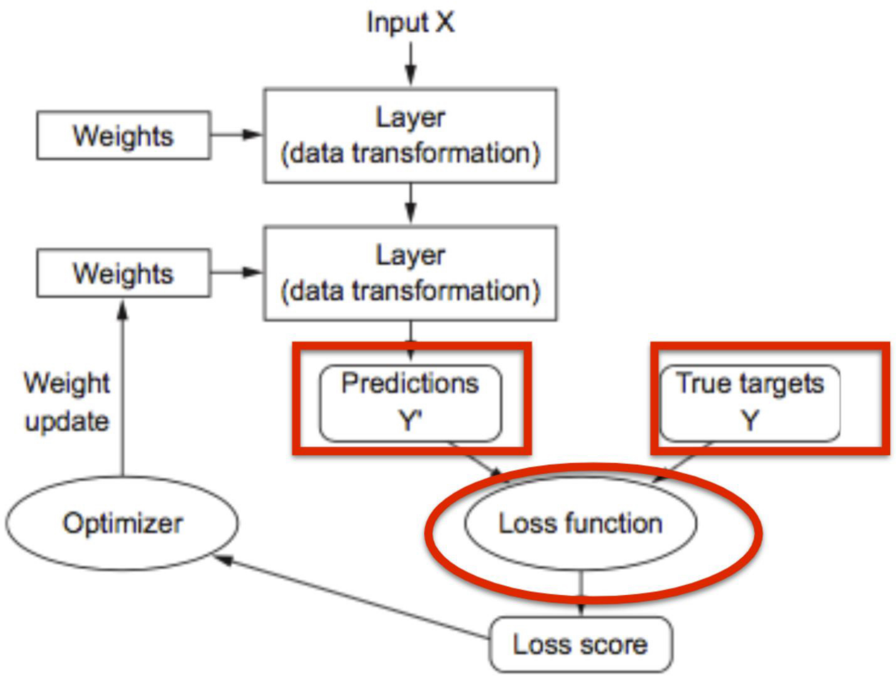
Week 8 (b)

# 3. Feedback Mechanism

# Compiling the model

Now that we have a model architecture in place - how will the model *learn* from the data? To do this, we need to specify a **loss function** and **optimiser** to use during training.

- 📊 The *loss function* (also called objective function) helps measure performance. For regression you may use the MSE, for classification you may use cross entropy.
- 📊 The *optimiser* controls which optimisation algorithm is implemented in our NN.


Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Weight
update
Predictions
Y'
True targets
Y
Optimizer
Loss function
Loss score

# Compiling the model in R

In R, we pipe our model to the `compile` function. This is all done in place and is not assigned to an object!

```
model %>%   compile(
    loss = 'categorical_crossentropy',
    optimizer = "rmsprop",
    metrics = c('accuracy')
  )
```

# 4. Model Training

# Model training

Now that we have created the model specification, we are ready to give it some data! We can use the `fit` function in `keras` to achieve this.

```
fit <- model %>% fit(
  x = mnist_x,
  y = mnist_y,
  batch_size = 512,
  epochs = 10
)
```

Note - `batch_size` refers to the number of samples fed into the model at a time, and `epoch` refers to how many times we will transverse the input data.

# Model training

Now that we have created the model specification, we are ready to give it some data! We can use the `fit` function in `keras` to achieve this.

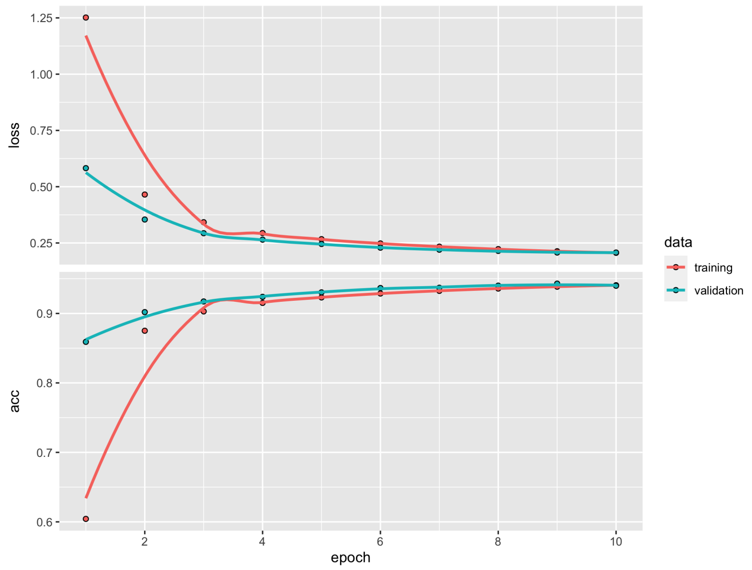
Additionally, we can hold out data in `validation_split` to validate that we are not *overfitting* to out data.

```
fit <- model %>% fit(
  x = mnist_x,
  y = mnist_y,
  batch_size = 512,
  epochs = 10,
  validation_split = 0.2,
  verbose = FALSE
)
```

# Model training

We can plot the accuracy and loss of the neural network using the `plot` function.

```
plot(fit)
```



# Predict test set

```
mnist_ts_yp <- predict_classes(model, mnist_ts_x)
table(mnist_ts_yp, mnist_ts_y)
```

```
##              mnist_ts_y
## mnist_ts_yp    0    1    2    3    4    5    6    7    8    9
##           0  960    0   13    3    1    7   10    1    7    9
##           1    0 1109    3    0    0    1    3    7    2    5
##           2    0    2  940   13    1    0    4   15    5    1
##           3    1    3   14  906    0   19    1    5   12    7
##           4    0    1   10    1  926    4    7    6   13   28
##           5    7    0    2   33    1  827   14    0   15    4
##           6    5    3   11    2   12   11  914    0    7    1
##           7    2    4    9   13    2    3    0  967    8    6
##           8    4   13   30   28    4   10    4    1  894    4
##           9    1    0    0   11   35   10    1   26   11  944
```

# Additional thoughts - regularisation

Place constraints on model complexity. Can use a $L_1$ or $L_2$ penalty to add a cost to the size of the node weights.

$$RSS + \lambda \sum_k w_k^2$$

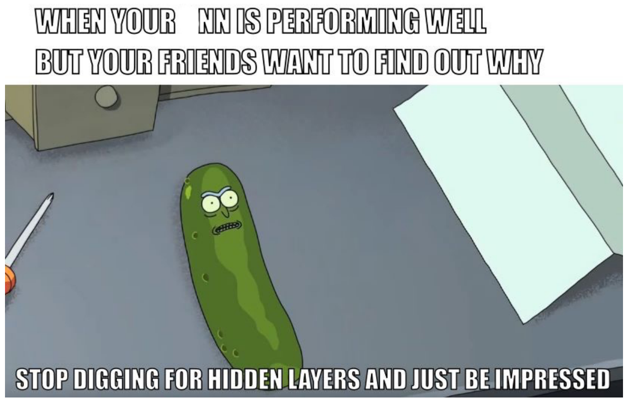
where $w$ indicates the set of weights in the model, labelled $\alpha, \beta$ earlier. Forces some of the weights to zero (or close to), to alleviate over-parametrization, and over-fitting.

More on regularisation next week

So why don't we use neural networks for all machine learning problems?

# Minimal interpretability

- Core concept of prediction vs inference.
- Neural networks are seen as a black box type of model, with limited information provided to as how the neural net is making decisions. (*Contrast this to trees, or logistic regression, say*)




Source: Machine Learning Memes for Convolutional Teens

## Data intensive

- Deep learning algorithms don't work well when the number of features is larger than the number of observations (highly overparameterised).
- If we only have a limited number of training data points, the model can potentially overfit and fit very closely to the training data whilst lacking predictive performance for new data.


Source: Machine Learning Memes for Convolutional Teens

# Computationally intensive

- Many calculations are required to estimate all of the parameters in many neural networks (the one we have shown today is quite basic ).
- Deep learning involves huge amounts of matrix multiplications and other operations.
- Often used in conjuction with GPUs to paralellise computations.


Source: Machine Learning Memes for Convolutional Teens

# Resources

- Neural Networks: A Review from a Statistical Perspective
- A gentle journey from linear regression to neural networks
- McCulloch-Pitts Neuron -- Mankind's First Mathematical Model Of A Biological Neuron
- Hands on Machine Learning with R - Deep Learning

## 👩‍💻 Made by a human with a computer, with help from Sarah Romanes

Slides at https://iml.numbat.space.

Code and data at https://github.com/numbats/iml.

Created using R Markdown with flair by **xaringan**, and **kunoichi** (female ninja) style.