

# ETC3250/5250: Neural networks 2

Semester 1, 2020

Professor Di Cook

Econometrics and Business Statistics  
Monash University

Week 8 (b)

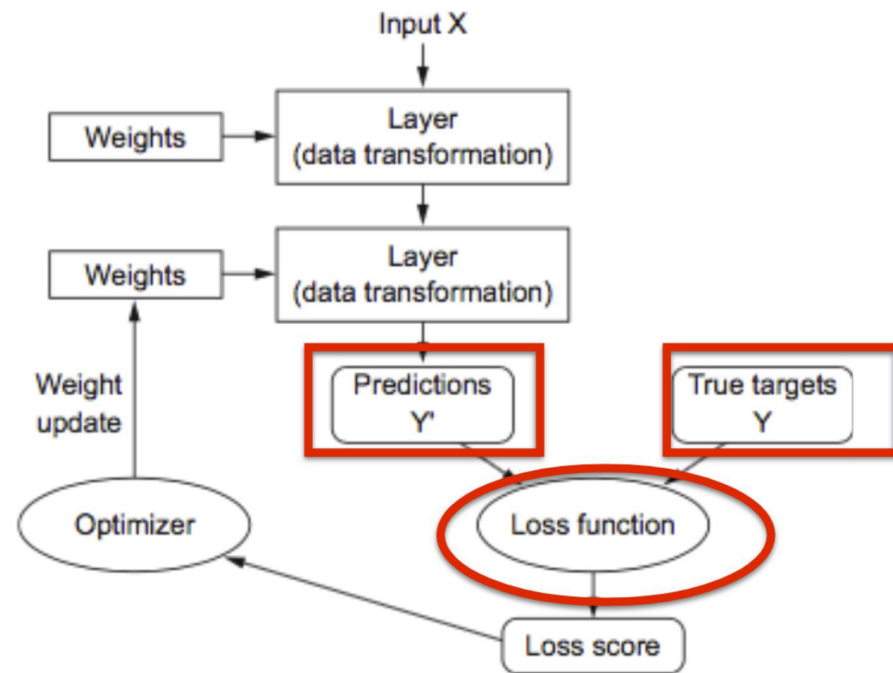
### 3. Feedback Mechanism

## 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.



## 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, which can manage the computational load, and model stability, and `epoch` refers to how many iterations through the entire training 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 our data.

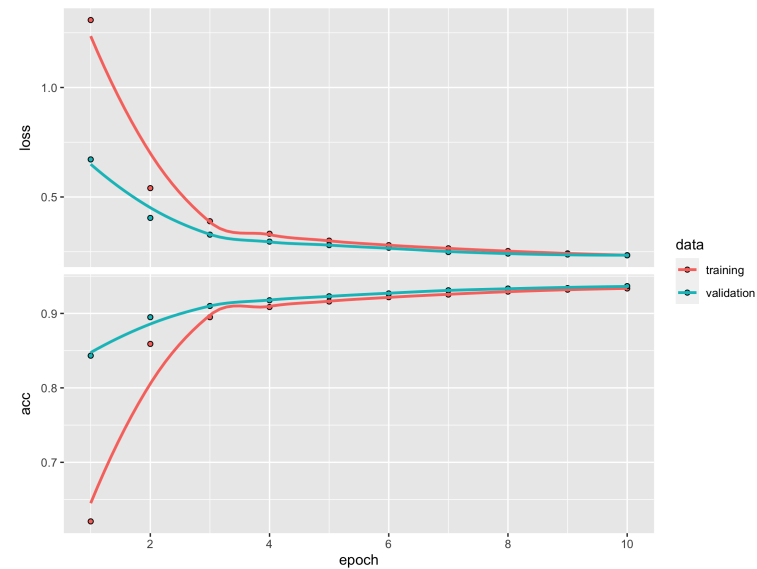
```
fit <- model %>% fit(  
  x = mnist_x,  
  y = mnist_y,  
  batch_size = 512,  
  epochs = 10,  
  validation_split = 0.2,  
  verbose = FALSE  
)
```



# 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  969    0   12    1    2    9   20    3    6   12
##          1    0 1101    2    1    1    4    3    8    2    5
##          2    0    3  946   15    7    4   13   27    4    1
##          3    2    3   10  918    0   19    2    7   14    9
##          4    0    1   11    0  929    5    9    6    8   36
##          5    4    2    4   37    1  810   24    1   26   10
##          6    1    3    7    0    5    5  879    0    4    0
##          7    1    1    9   14    5    6    0  961    6   15
##          8    3   21   31   23   13   25    8    2  902   15
##          9    0    0    0    1   19    5    0   13    2  906
```

## 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-parameterisation, and over-fitting.

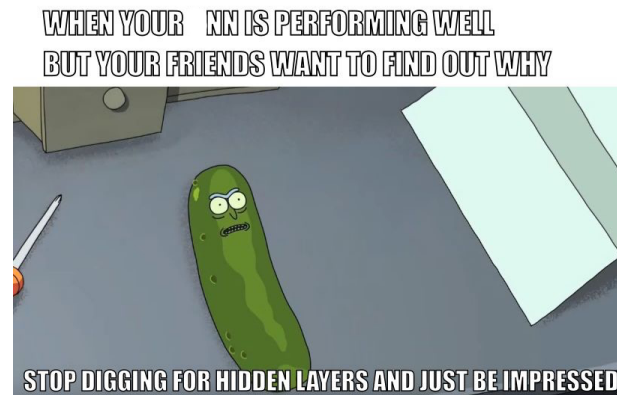
More on regularisation next week

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 over-parameterised).

📊 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.

**When someone asks you why you're not using a neural network model to solve your problem**



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 conjunction with GPUs to parallelise computations.





*Trains the model for 2.8 hours*



*forgets to save the weights to the disk*

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.



This work is licensed under a Creative Commons Attribution-  
ShareAlike 4.0 International License.

