ETC3555 2018 - Lab 7

Deep neural networks

Cameron Roach and Souhaib Ben Taieb

05 October, 2018

Preliminaries

We will be working with the Keras framework to implement our deep neural network (DNN). Keras is a front end for several DNN frameworks (e.g. TensorFlow, Theano, etc.) which aims to simplify the coding process. Install Keras for `R` by running:

```r
install.packages("keras")
library(keras)
install_keras(method = c("conda"), conda = "auto", tensorflow = "default",
              extra_packages = c("tensorflow-hub"))
```

TensorFlow should automatically be installed and selected as the default backend.

If at any stage you need to retrain your models from scratch make sure to clear your workspace and refresh your R session. Failing to do so will result in continued training of existing models or unexpected errors.

Exercises

Exercise 1

Work through the introduction to Keras tutorial on the RStudio blog located at https://keras.rstudio.com/. This walk-through will have you implement a feed-forward neural network with:

1. 256 node hidden layer with relu activation function
2. dropout layer with 0.4 dropout rate
3. 128 node hidden layer with relu activation function
4. dropout layer with 0.3 dropout rate
5. fully connected (dense) output layer with softmax output.

After completing the tutorial answer the following questions:

a) Why would you use softmax on the output layer and relu on the input?
b) Why different dropout rates (is there any benefit in varying them)?
c) Is the loss lower on the validation set than the training set for early epochs? Does this agree with your intuition? What happens when you remove the dropout layers and retrain? Explain why.
d) Once you figure out (c), can you justify the design choice?
e) Does the training accuracy and validation accuracy cross? Why? If not, why?

Assignment - Question 1

a) Create another DNN as above, but with half as many units in each hidden layer. Assess the performance of this model against your previous model.
b) Now try L1 or L2 regularization with dropout layers removed.

c) Create another model without dropout or regularization and implement early stopping based on the validation loss. Is the loss improved? Is the accuracy improved? How do the three models compare? Produce a plot comparing both the loss and accuracy of each model.

Assignment - Question 2

Create a new network with the following structure:

1. $d^{(1)}$ node hidden layer with relu activation function
2. dropout layer with q dropout rate
3. fully connected (dense) output layer with softmax output.

Use grid search to select optimal $d^{(1)} \in \{16, 32, 64, 128\}$ and $q \in \{0, 0.25, 0.5, 0.75\}$ values. Create a heatmap or similar showing the loss for each combination of hyperparameter values. Which combination gives the best performance? Comment on the suitability of this grid of values.

Assignment - Question 3

Using your optimal hyperparameters from the previous question, try refitting the network using:

- Momentum
- ADAM
- RMSProp.

Comment on how each affects the model accuracy on the test set. Which converges fastest?

Solutions

Exercise 1

(a) Softmax converts to a probability. Relu is just an efficient way to introduce non-linearities into the neural network model. It tends to outperform other activation functions.
(b) Maybe - what did you find?
(c) From FAQ:

> Why is the training loss much higher than the testing loss?
>
> A Keras model has two modes: training and testing. Regularization mechanisms, such as Dropout and L1/L2 weight regularization, are turned off at testing time.
>
> Besides, the training loss is the average of the losses over each batch of training data. Because your model is changing over time, the loss over the first batches of an epoch is generally higher than over the last batches. On the other hand, the testing loss for an epoch is computed using the model as it is at the end of the epoch, resulting in a lower loss.

(d) Dropout is useful when training as a form of regularization. You don't want to apply dropout during the final predictions as it will decrease accuracy.
(e) Overfitting.

Assignment - Question 1

(3 marks)

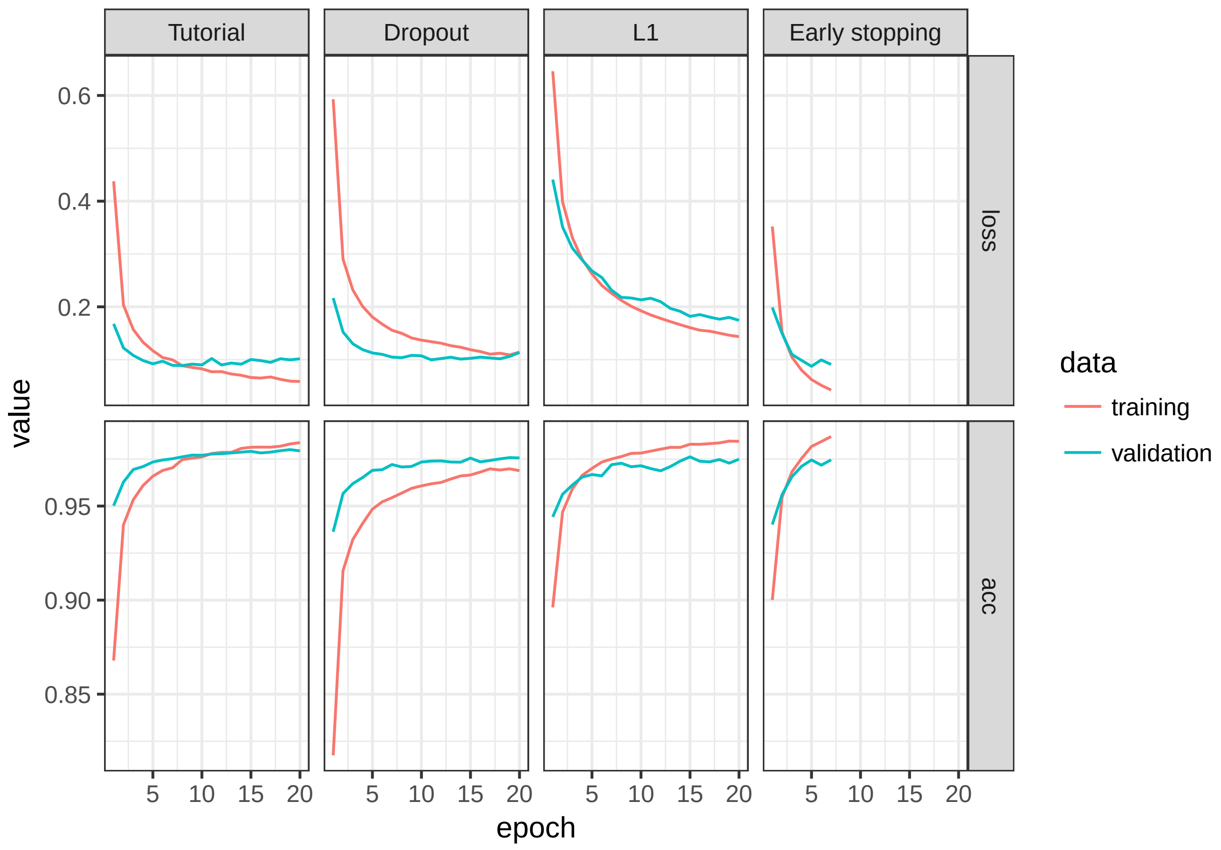
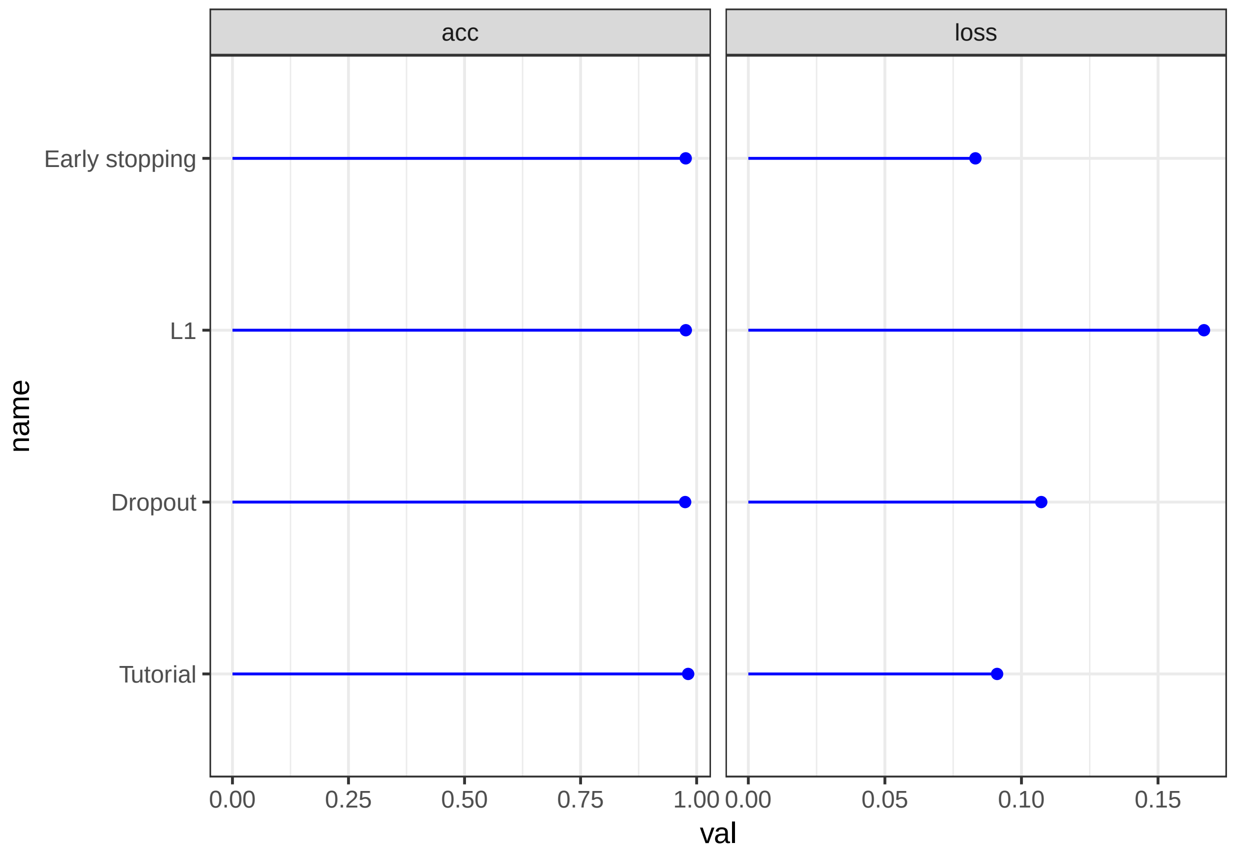


Table 1: Loss and accuracy for Keras models

name	loss	acc
Tutorial	0.0910859	0.9817
Dropout	0.1072502	0.9753
L1	0.1668341	0.9767
Early stopping	0.0831466	0.9765



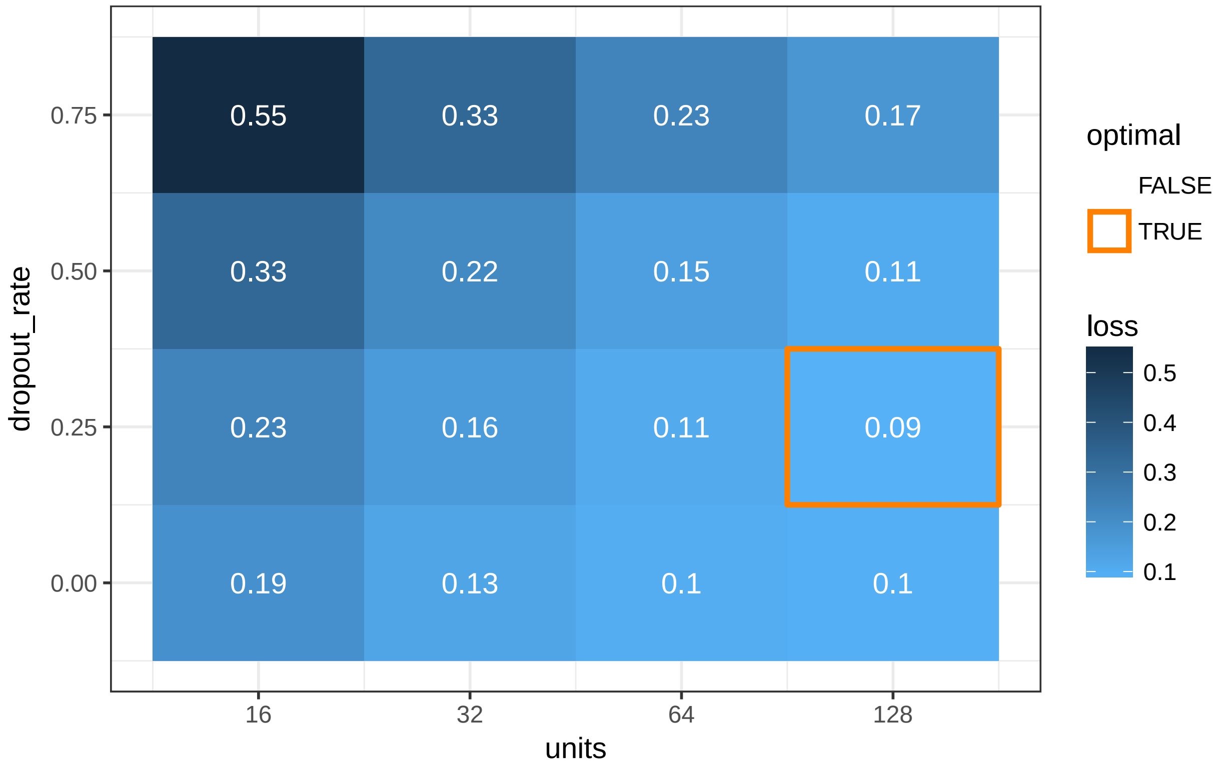
Assignment - Question 2

(4 marks)

We can see that the optimal performance using 5-fold cross validation on our training set occurs at the boundary of our subset of the hyperparameter space. If we increase the size of our hyperparameter space we observe a better combination of values.

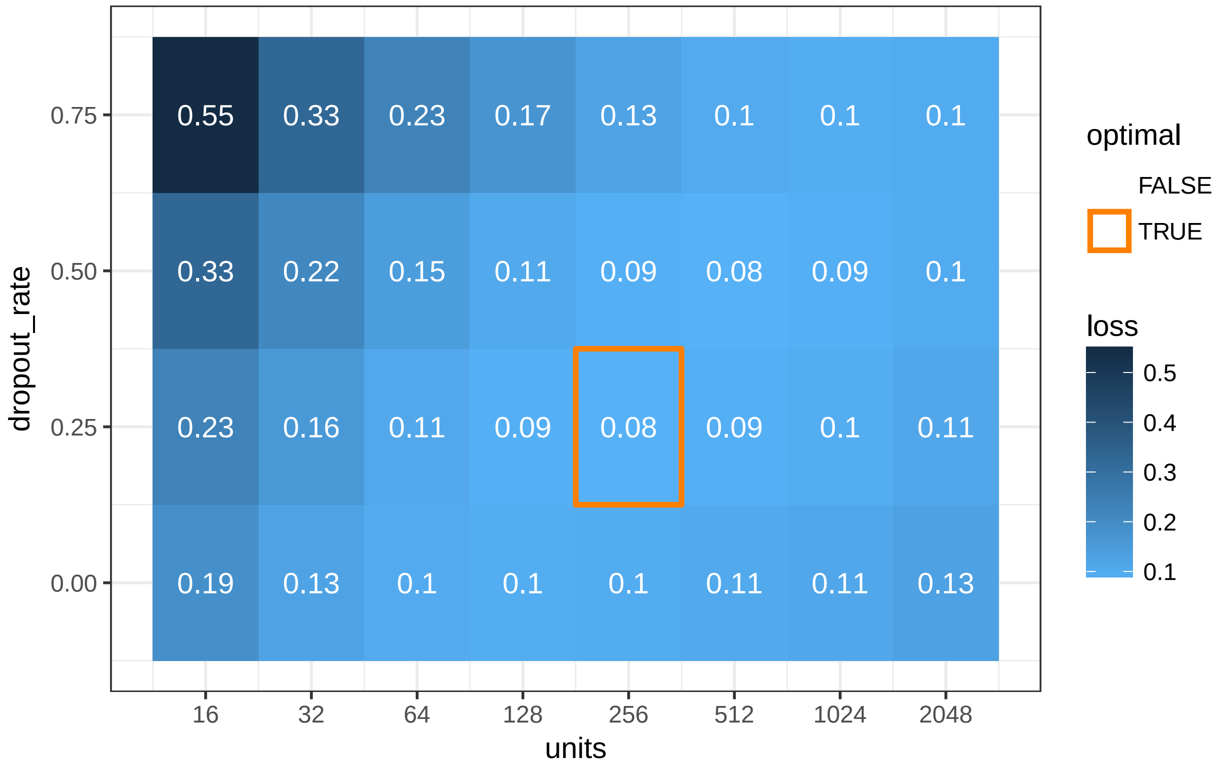
Grid search for optimal dropout rate and number of units

Based on 5−fold cross validation



Grid search for optimal dropout rate and number of units

Based on 5−fold cross validation



Assignment - Question 3

(3 marks)

This is just a matter of updating the `optimizer` argument in `compile`. For example:

```
compile(loss = "categorical_crossentropy",
        optimizer = optimizer_sgd(momentum = 0.3),
        metrics = c("accuracy"))
```

You'll likely see that momentum is fastest, but doesn't reach the same accuracy. To reach same accuracy it could run for more epochs, but then might not be the fastest anymore. RMSProp and ADAM reach a given accuracy (say 95%) in fewer epochs. You can use `system.time()` or `microbenchmark` from the `microbenchmark` package to time how long each takes.