STAT430: Machine Learning for Financial Data

LarryHua.com/teaching

Spring 2019

Monitoring Deep-learning Models

Keras callback functions - checkpoint

- Model checkpoint: save the current weights of the model at different points during training
    - `callback_model_checkpoint()`
    - `save_best_only = TRUE`: the latest best model won't be overwritten

```
checkPoint <- callback_model_checkpoint(filepath = "saved_model.h5",
                                        monitor = "val_acc", save_best_only = TRUE)
model %>% fit_generator( ..., callbacks = list(checkPoint), ...)
```

Keras callback functions - early stop

- Early stop: interrupt training when the it is no longer improving (and saving the best model obtained during training)
  - `callback_early_stopping()`
  - `restore_best_weights = TRUE`: restore model weights from the epoch with the best value of the monitored quantity

```
earlyStop <- callback_early_stopping(monitor = "val_acc", patience = 1)
model %>% fit_generator( ..., callbacks = list(earlyStop), ...)
```

  - note that, if monitored quantity is based on "validation", validation data should be put into `fit` or `fit_generator`

Keras callback functions - adjust parameters

- dynamically adjust the value of certain parameters during training
  - reduce learning rate when no improvement: `callback_reduce_lr_on_plateau()`
    - learning rate being too large leads to swinging around optimum
  - `factor=0.1`: new learning rate = lr × 0.1

```
reduceLr <- callback_reduce_lr_on_plateau(monitor = "val_acc", factor = 0.1, patience = 3)
model %>% fit_generator( ..., callbacks = list(reduceLr), ...)
```

  - schedule learning rates along epoch: `callback_learning_rate_scheduler()`

```
schedule <- function(epoch,lr) (lr)*(0.75^(floor(epoch/2)))
scheduleLr <- callback_learning_rate_scheduler(schedule)
model %>% fit_generator( ..., callbacks = list(scheduleLr), ...)
```

Keras callback functions - loggers

- save all metrics such as acc, loss, val_acc, val_loss for each epoch (starting from 0)
- `callback_csv_logger(csv_fileName)`

Customized callbacks

- creating a new R6 class that inherits from the KerasCallback class
- initialize an instance of such a class (eg, `hisAfterBatch` in the following)
- metrics being monitored are saved in the instance (eg, `hisAfterBatch`)

```
afterBatch <- R6::R6Class("LossHistory", inherit = KerasCallback,
                       public = list(losses <- NULL,
                                     on_batch_end = function(batch, logs = list())
                                       {
                                          self$losses <- c(self$losses, logs[["loss"]])
                                       }
                                    )
                       )
hisAfterBatch <- afterBatch$new()
model %>% fit_generator( ..., callbacks = list(hisAfterBatch), ...)
```

Customized callbacks

- implement any number of the following transparently named methods, which are called at various points during training
  - on_epoch_begin(epoch, logs): Called at the start of every epoch
  - on_epoch_end(epoch, logs): Called at the end of every epoch
  - on_batch_begin(batch, logs): Called right before processing each batch
  - on_batch_end(batch, logs): Called right after processing each batch
  - on_train_begin(logs): Called at the start of training
  - on_train_end(logs): Called at the end of training
- The callback has access to the following attributes automatically
  - self$model: the Keras model being trained
  - self$params: Named list with training parameters (verbosity, batch size, number of epochs, and so on)
- Try R
- Back to Course Scheduler