STAT430: Machine Learning for Financial Data

LarryHua.com/teaching

Spring 2019

Anatomy of a neural network

Core components of a neural network:

- Layers: which are combined into a network (or model)
- Input/features/covariates and targets/labels/response variables
- Loss function: the quantity that will be minimized during training
- Optimizer: a specific variant of stochastic gradient descent, determines how the model is updated based on the feedback from the loss function

Layers

- Some basic layers in Keras:
  - Simple vector data: 2D tensors of shape (samples, features) | densely connected layers
    - `layer_dense`: densely connected layers
  - Sequence data: 3D tensors of shape (samples, timesteps, features) | recurrent layers
    - `layer_lstm`: recurrent layers with long short-term memory
    - `layer_gru`: recurrent layers with gated recurrent unit
  - Image data: 4D tensors | 2D convolution layers
    - `layer_conv_2d`
- There are a lot more!
  - Keras cheatsheet

Layer compatibility

- Every layer only accepts input tensors of a certain shape and returns output tensors of a certain shape
- Example:

```
model <- keras_model_sequential() %>%
layer_dense(units = 32, input_shape = c(784)) %>%
layer_dense(units = 32)
```

  - `units` and `input_shape` are the shapes w/o the sample axis for output and input, respectively
  - Only the first layer needs `input_shape`
- Pipe operator: `%>%`
  - From the `magrittr` package
  - Pass the value on its left as the **first argument** to the function on its right
  - `layer_dense(object, units, ... )`, object: model or layer object

Models, Loss functions

- A deep-learning model is a directed, acyclic graph of layers
- Picking the right network architecture is more an art than a science
- Multiple loss functions can be employed for multiple outputs, but losses have to be combined into a scalar value for gradient descents
  - Cross entropy are often used for a classification problem
    - `model %>% compile(loss = 'categorical_crossentropy', ...)`
    - `model %>% compile(loss = 'sparse_categorical_crossentropy', ...)`, when labels are integers and not one-hot encoding
  - MSE for a regression problem
    - `model %>% compile(loss = 'mse', ...)`

Develop models with Keras

- Two approaches:
  - Use `keras_model_sequential()`: a linear stacks of layers
  - Easier and more readable

```
model <- keras_model_sequential() %>%
layer_dense(units = 32, activation = "relu", input_shape = c(784)) %>%
layer_dense(units = 10, activation = "softmax")
```

  - Use functional API: completely arbitrary architectures
  - Much more flexible

```
input_tensor <- layer_input(shape = c(784))
output_tensor <- input_tensor %>%
layer_dense(units = 32, activation = "relu") %>%
layer_dense(units = 10, activation = "softmax")
model <- keras_model(inputs = input_tensor, outputs = output_tensor)
```

Compile and fit models with Keras

- Use `compile` to configure learning process

```
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy') # monitor performance
)
```

- Use `fit` to configure run settings and fit model

```
his <- model %>% fit(
  x_train, y_train,
  epochs = 30, batch_size = 128, # epochs: number of passes through the full training set
  validation_split = 0.2 # 20% from _train used for validation
)
```

- Try R
- Back to Course Scheduler