Sequential models with Keras

Max Pumperla - Skymind

skymind

Two types of Keras models

- Sequential models: `Sequential`

skymind

Two types of Keras models

- Sequential models: `Sequential`
- Simply stack layers sequentially

skymind

Two types of Keras models

- Sequential models: `Sequential`
- Simply stack layers sequentially
- Focus of this lecture

skymind

Two types of Keras models

- Sequential models: `Sequential`
- Simply stack layers sequentially
- Focus of this lecture
- Non-sequential models: `Model`

skymind

Two types of Keras models

- Sequential models: `Sequential`
- Simply stack layers sequentially
- Focus of this lecture
- Non-sequential models: `Model`
- More general models, uses a functional API

skymind

Two types of Keras models

- Sequential models: `Sequential`
- Simply stack layers sequentially
- Focus of this lecture
- Non-sequential models: `Model`
- More general models, uses a functional API
- Will be introduced later on

skymind

Keras layers

- Core abstraction for every model

skymind

Keras layers

- Core abstraction for every model
- In sequential models layers have `input`, `output`, `input_shape` and `output_shape`

skymind

Keras layers

- Core abstraction for every model
- In sequential models layers have `input`, `output`, `input_shape` and `output_shape`
- Can get weights as list of numpy arrays: `layer.get_weights()`

skymind

Keras layers

- Core abstraction for every model
- In sequential models layers have `input`, `output`, `input_shape` and `output_shape`
- Can get weights as list of numpy arrays: `layer.get_weights()`
- Set layer weights with `layer.set_weights(weights)`

skymind

Keras layers

- Core abstraction for every model
- In sequential models layers have `input`, `output`, `input_shape` and `output_shape`
- Can get weights as list of numpy arrays: `layer.get_weights()`
- Set layer weights with `layer.set_weights(weights)`
- Each layer has a defining configuration, `layer.get_config()`

skymind

Building a sequential model

- Instantiate a `Sequential` model

skymind

Building a sequential model

- Instantiate a `Sequential` model
- Add layers to it one by one using `add`

skymind

Building a sequential model

- Instantiate a `Sequential` model
- Add layers to it one by one using `add`
- Compile the model with a loss function, an optimizer and optional evaluation metrics

skymind

Building a sequential model

- Instantiate a `Sequential` model
- Add layers to it one by one using `add`
- Compile the model with a loss function, an optimizer and optional evaluation metrics
- Use data to `fit` the model

skymind

Building a sequential model

- Instantiate a `Sequential` model
- Add layers to it one by one using `add`
- Compile the model with a loss function, an optimizer and optional evaluation metrics
- Use data to `fit` the model
- Evaluate model, persist or deploy model, start new experiment etc.

skymind

Compiling models I: loss functions

```
# Option 1: Importing from loss module (preferred)
from keras.losses import mean_squared_error
model.compile(loss=mean_squared_error, optimizer=...)

# Option 2: Using strings
model.compile(loss='mean_squared_error', optimizer=...)
```

skymind

Compiling models I: optimizers

```
# Option 1: Load optimizers from module (preferred)
from keras.optimizers import SGD

sgd = SGD(lr=0.01,          # learning rate >= 0
          decay=1e-6,       # lr decay after updates
          momentum=0.9)     # Momentum parameter used for SGD
model.compile(loss=..., optimizer=sgd)


# Option 2: pass string (default parameters will be used)
model.compile(loss=..., optimizer='sgd')
```

skymind

Fit, evaluate and predict

```
# Fit a model to train data
model.fit(x_train, y_train,
          batch_size=32,
          epochs=10,
          validation_data=(x_val, y_val))

# Evaluate on test data
evaluate(x_test, y_test, batch_size=32)

# Predict labels
predict(x_test, batch_size=32)
```