Keras

Class	09_deep_learning
Created	@Jan 25, 2021 10:43 AM
Materials	
Reviewed	☐
Type	

Keras

1) What is Keras?

> 💡 "Deep learning for humans. Keras is an API designed for human beings, not machines."

> 💡 "The purpose of Keras is to be a model-level framework, providing a set of "Lego blocks" for building Deep Learning models in a fast and straightforward way."

- It is fundamentally an API to other software that is designed for fast and efficient calculations using tensors (array of matrices).
- It used to be an API to several backends like Tensorflow (Google) and Theano (University of Montreal)
- Along the way, Keras added to the core Tensorflow library as prime API to access Tensorflow functionality through Python code
- That is why we will install `tensorflow` and use `tensorflow.keras` instead of `keras` directly.

2) How to install Tensorflow

2.1) Create a virtual environment

```
conda create -n <your_project_name> pip python==3.8
```

2.2) Activate your virtual environment

```
conda activate <your_project_name>
```

2.3) Pip install tensorflow

```
pip install --upgrade pip
pip install tensorflow
```

3) Keras "Lego blocks"

This way we will not have to write the model ourselves.

3.1) Model

```
from tensorflow.keras.models import Sequential
```

This `class` allows us to construct deep learning models with a sequential order of layers.

3.2) Layers

```
from tensorflow.keras.layers import Dense
```

A `Dense` layer is just a fully connected layer. The only model architecture that you have seen so far. You are getting to know other architectures later this week.

3.3) Define Model

Use `Sequential` and `Dense` to define a model architecture.

```
model = Sequential([
          Dense(units=2, activation='sigmoid', input_shape=(2,)),
          Dense(units=1, activation='sigmoid')
])
```

3.4) Compile the Model

Configures the model for training. The parameters to be chosen here are:

- `optimizer` - The algorithm with which the model is trained
- `loss` - The loss function used by the algorithm to train the model
- `metrics` - Metrics for the model evaluation

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

3.5) Fit the model

Parameters for the model fitting are:

- `X`
- `y`
- `epochs` - # of iterations on the whole training data
- `batch_size` - # of training data points to use at once for training
- `validation_split` - fraction

```
model.fit(x=X, y=y, epochs=200, batch_size=32)
```

3.6) Evaluate the model

```
model.evaluate(X, y)
```

3.7) Model summary

After you define the model, model.summary() provides you with a really nice overview of the model.

```
model.summary()
```

3.8) Get weights

```
model.get_weights()
```