Keras

Import

There are lots of things in Keras that you might want to import. For now, we'll stick with the following submodules:

```
from keras import models
from keras import layers
```

Sequential Models

Sequential models are built up as a sequence of layers leading from the input layer to the output layer.

Overall Structure

To specify a sequential model in Keras, do the following steps:

1. Create a Sequential model object

```
my_model = models.Sequential()
```

2. Add layers to the model
   - Use the `add` method on your model object
   - The first argument is a layer object – for now, we know about Dense layers:
     - Dense means all units in the previous layer feed into the next layer
     - First argument is the number of units in the new layer you're adding
     - activation is the name of the activation function to use.

   - Your first layer must specify an `input_shape` – how many features?

```
my_model.add(layers.Dense(1, activation = 'sigmoid', input_shape = (2,)))
# potentially add more layers here
```

3. Compile the model, specifying:
   - The algorithm to use to minimize the loss/cost function: in this class, we'll discuss 'sgd', 'RMSprop', and 'adam'
   - The loss function to use: so far, we have discussed 'binary_crossentropy', 'categorical_crossentropy', 'sparse_categorical_crossentropy', and 'mean_squared_error'
   - Any measures of performance you want to compute on the training and validation sets: for classification tasks, this is typically ['accuracy']. For regression tasks, you might not include anything.

```
my_model.compile(optimizer = 'sgd', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

4. Run the estimation process, specifying:
   - training set features
   - training set responses
   - validation_data – a tuple of training set features and responses
   - epochs and batches – we will discuss more soon, but essentially how many passes through the training data you want to make during the estimation process and how many training set observations are processed at once.

```
my_model.fit(train_x, train_y,
  validation_data = (X_val, y_val),
  epochs = 1000, batch_size = 32)
```

5. If you decide you like your model, you might be ready to get test set predictions

```
my_model.evaluate(X_test, y_test)
y_hat = my_model.predict(X_test)
```