Representation learning

- Deep networks internally build representations of patterns in the data
- Partially replace the need for feature engineering
- Subsequent layers build increasingly sophisticated representations of raw data

Backpropagation Algorithm

- Go back one layer at a time
- Gradients for weight is product of:
  1. Node value feeding into that weight
  2. Slope of loss function w.r.t node it feeds into
  3. Slope of activation function at the node it feeds into

Model Specification

```
In [1]: import numpy as np

In [2]: from keras.layers import Dense

In [3]: from keras.models import Sequential

In [4]: predictors = np.loadtxt('predictors_data.csv', delimiter=',')

In [5]: n_cols = predictors.shape[1]

In [6]: model = Sequential()

In [7]: model.add(Dense(100, activation='relu', input_shape = (n_cols,)))

In [8]: model.add(Dense(100, activation='relu'))

In [9]: model.add(Dense(1))
```

Compiling and Fitting a Model

```
In [6]: model.compile(optimizer='adam', loss='mean_squared_error')

In [7]: model.fit(predictors, target)
```

Classification

```
In[1]: from keras.utils import to_categorical

In[2]: data = pd.read_csv('basketball_shot_log.csv')

In[3]: predictors = data.drop(['shot_result'], axis=1).as_matrix()

In[4]: target = to_categorical(data.shot_result)

In[5]: model = Sequential()

In[6]: model.add(Dense(100, activation='relu', input_shape = (n_cols,)))

In[7]: model.add(Dense(100, activation='relu'))

In[8]: model.add(Dense(100, activation='relu'))

In[9]: model.add(Dense(2, activation='softmax'))

In[10]: model.compile(optimizer='adam', loss='categorical_crossentropy',
   ...:               metrics=['accuracy'])

In[11]: model.fit(predictors, target)
```

Making Predictions

```
In [1]: from keras.models import load_model

In [2]: model.save('model_file.h5')

In [3]: my_model = load_model('my_model.h5')

In [4]: predictions = my_model.predict(data_to_predict_with)

In [5]: probability_true = predictions[:,1]
```

Vanishing Gradient

- Occurs when many layers have very small slopes (e.g. due to being on flat part of tanh curve)
- In deep networks, updates to backprop were close to 0

Dying Neuron Problem – ReLU

- Once a node starts always getting negative inputs
  - It may continue only getting negative inputs
- Contributes nothing to the model
  - "Dead" neuron

Early Stopping

```
In [3]: from keras.callbacks import EarlyStopping

In [4]: early_stopping_monitor = EarlyStopping(patience=2)

In [5]: model.fit(predictors, target, validation_split=0.3, epochs=20,
   ...:           callbacks = [early_stopping_monitor])
```

Model Capacity

- Start with a small network
- Gradually increase capacity
- Keep increasing capacity until validation score is no longer improving