2017 Predictive Analytics Symposium

Session 22, TensorFlow (workshop)

Moderator:
Stuart Klugman, FSA, CERA, Ph.D.

Presenter:
Jeff T. Heaton, Ph.D.

SOA Antitrust Compliance Guidelines
SOA Presentation Disclaimer
Session 22: TensorFlow (workshop)

Presented by Jeff Heaton, Ph.D.

September 14, 2017 – SOA Predictive Analytics Symposium.

Washington University in St. Louis
Jeff Heaton, Ph.D.

- Lead Data Scientist at RGA
- Adjunct Instructor at Washington University
- Fellow of the Life Management Institute (FLMI)
- Senior Member of IEEE
- Kaggler (Expert level)
- [http://www.jeffheaton.com](http://www.jeffheaton.com) – (contact info at my website)
T81-558: Applications of Deep Learning

• Course Website: https://sites.wustl.edu/jeffheaton/t81-558/
• Instructor Website: https://sites.wustl.edu/jeffheaton/
• Course Videos: https://www.youtube.com/user/HeatonResearch
Presentation Outline

• TensorFlow as a Compute Graph/Engine
• Keras and TensorFlow
• Keras: Classification
• Keras: Regression
• Keras: Computer Vision and CNN
• Keras: Time Series and RNN
• GPU
TensorFlow as a Compute Graph/Engine
What are Tensors? Why are they flowing?

- Tensor of Rank 0 (or scaler) – simple variable
- Tensor of Rank 1 (or vector) – array/list
- Tensor of Rank 2 (or matrix) – 2D array
- Tensor of Rank 3 (or cube) – 3D array
- Tensor of Rank 4 (tesseract/hypercube) – 4D array
- Higher ranks (hypercube) – nD array
What is a Computation Graph?

```python
import tensorflow as tf

matrix1 = tf.constant([[3., 3.]])
matrix2 = tf.constant([[2.], [2.]])
product = tf.matmul(matrix1, matrix2)

with tf.Session() as sess:
    result = sess.run([product])

print(result)
```

\[
\begin{bmatrix}
3 & 3 \\
\end{bmatrix}
\times
\begin{bmatrix}
2 \\
2 \\
\end{bmatrix}
= \begin{bmatrix}
3 \times 2 + 3 \times 2 \\
\end{bmatrix}
= \begin{bmatrix}
12 \\
\end{bmatrix}
\]
import tensorflow as tf
sess = tf.InteractiveSession()

x = tf.Variable([1.0, 2.0])
a = tf.constant([3.0, 3.0])
x.initializer.run()

sub = tf.subtract(x, a)
print(sub.eval())
# ==> [-2. -1.]

sess.run(x.assign([4.0, 6.0]))
print(sub.eval())
# ==> [1. 3.]
Computation Graph for Mandelbrot Set
Mandelbrot Set Review

• Some point $c$ is a complex number with $x$ as the real part, $y$ as the imaginary part.

• $z_0 = 0$
• $z_1 = c$
• $z_2 = z_1^2 + c$
• ...
• $z_{n+1} = z_n^2 + c$
Mandelbrot Rendering in TensorFlow

xs = tf.constant(Z.astype(np.complex64))
zs = tf.Variable(xs)
ns = tf.Variable(tf.zeros_like(xs, tf.float32))
tf.global_variables_initializer().run()

# Compute the new values of z: z^2 + x
zs_ = zs*zs + xs

# Have we diverged with this new value?
not_diverged = tf.abs(zs_) < 4
step = tf.group(
    zs.assign(zs_),
    ns.assign_add(tf.cast(not_diverged, tf.float32))
)

for i in range(200): step.run()
Keras and TensorFlow
Tools Used in this Presentation

- Anaconda Python 3.6
- Google TensorFlow 1.2
- Keras 2.0.6
- Scikit-Learn
- Jupyter Notebooks
Installing These Tools

• Install Anaconda Python 3.6
• Then run the following:
  – conda install scipy
  – pip install sklearn
  – pip install pandas
  – pip install pandas-datareader
  – pip install matplotlib
  – pip install pillow
  – pip install requests
  – pip install h5py
  – pip install tensorflow==1.2.1
  – pip install keras==2.0.6
Keras and TensorFlow
Anatomy of a Neural Network

• **Input Layer** – Maps inputs to the neural network.
• **Hidden Layer(s)** – Helps form prediction.
• **Output Layer** – Provides prediction based on inputs.
• **Context Layer** – Holds state between calls to the neural network for predictions.
What is Deep Learning

• Deep learning is almost always applied to neural networks.
• A deep neural network has more than 2 hidden layers.
• Deep neural networks have existed as long as traditional neural networks.
  – We just did not have a way to train deep neural networks.
• Neural networks have risen three times and fallen twice in their history. Currently, they are on the rise.
The True Believers – Luminaries of Deep Learning

- From left to right:
- Yann LeCun
- Geoffrey Hinton
- Yoshua Bengio
- Andrew Ng
Why Use Deep Learning

• Deep neural networks often accomplish the same task as other models, such as:
  – Support Vector Machines
  – Random Forests
  – Gradient Boosted Machines

• For many problems deep learning will give a less accurate answer than the other models.

• However, for certain problems, deep neural networks perform considerably better than other models.
Why Deep Learning (high y-axis is good)

Andrew Ng on Deep Learning
where AI will learn from untagged data

Learning from tagged data

Deep learning

Older algorithms

Performance vs. Amount of data
Supervised or Unsupervised?

**Supervised Machine Learning**
- Usually classification or regression.
- For an input, the correct output is provided.
- Examples of supervised learning:
  - Propensity to buy
  - Credit scoring

**Unsupervised Machine Learning**
- Usually clustering.
- Inputs analyzed without any specification of a correct output.
- Examples of unsupervised learning:
  - Clustering
  - Dimension reduction
Types of Machine Learning Algorithm

- **Clustering**: Group records together that have similar field values. For example, customers with common attributes in a propensity to buy model.

- **Regression**: Learn to predict a numeric outcome field, based on all of the other fields present in each record. For example, predict the amount of coverage a potential customer might buy.

- **Classification**: Learn to predict a non-numeric outcome field. For example, learn the type of policy an existing customer has a potential of buying next.
## Application of Machine Learning Algorithm

<table>
<thead>
<tr>
<th></th>
<th>Predictive Modeling</th>
<th>Computer Vision</th>
<th>Time Series</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classification</strong></td>
<td>• Intrusion Detection</td>
<td>• Face Recognition</td>
<td>• Buy, Sell, or Hold?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Intrusion Detection</td>
</tr>
<tr>
<td><strong>Regression</strong></td>
<td>• Normal Operating Levels</td>
<td>• Age Determination</td>
<td>• Tomorrow’s opening stock price</td>
</tr>
<tr>
<td><strong>Clustering</strong></td>
<td>• Product Recommendation</td>
<td>• Design Recommendation</td>
<td>• Anomaly Detection</td>
</tr>
</tbody>
</table>
Problems that Deep Learning is Well Suited to

<table>
<thead>
<tr>
<th>Predictive Modeling</th>
<th>Computer Vision</th>
<th>Time Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sepal length</td>
<td>Sepal width</td>
<td>Petal length</td>
</tr>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
</tr>
<tr>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
</tr>
<tr>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
</tr>
<tr>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
</tr>
<tr>
<td>4.4</td>
<td>2.0</td>
<td>1.4</td>
</tr>
<tr>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
</tr>
<tr>
<td>5.4</td>
<td>3.7</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Keras: Classification
The Classic Iris Dataset

• Classic classification problem.
• 150 rows with 4 predictor columns.
• All 150 rows are labeled as a species of iris.
• Three different iris species.
• Created by Sir Ronald Fisher in 1936.
• Predictors:
  – Petal length
  – Petal width
  – Sepal length
  – Sepal width
The Classic Iris Dataset

<table>
<thead>
<tr>
<th>sepal_length</th>
<th>sepal_width</th>
<th>petal_length</th>
<th>petal_width</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>7.0</td>
<td>3.2</td>
<td>4.7</td>
<td>1.4</td>
<td>Iris-versicolor</td>
</tr>
<tr>
<td>6.3</td>
<td>3.3</td>
<td>6.0</td>
<td>2.5</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>6.4</td>
<td>3.2</td>
<td>4.5</td>
<td>1.5</td>
<td>Iris-versicolor</td>
</tr>
<tr>
<td>5.8</td>
<td>2.7</td>
<td>5.1</td>
<td>1.9</td>
<td>Iris-virginica</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

...
Are the Iris Data Predictive?
Keras Classification: Load and Train/Test Split

```python
path = "./data/"

filename = os.path.join(path,"iris.csv")
df = pd.read_csv(filename,na_values=['NA','?'])

species = encode_text_index(df,"species")
x,y = to_xy(df,"species")

# Split into train/test
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.25, random_state=42)
```
model = Sequential()
model.add(Dense(10, input_dim=x.shape[1],
kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3,
patience=5, verbose=1, mode='auto')

model.fit(x,y,validation_data=(x_test,y_test),callbacks=[monitor],verbose=2,epochs=1000)
Keras Classification: Build NN and Fit

# Evaluate success using accuracy
# raw probabilities to chosen class (highest probability)
pred = model.predict(x_test)

def pred = np.argmax(pred, axis=1)

y_compare = np.argmax(y_test, axis=1)
score = metrics.accuracy_score(y_compare, pred)
print("Accuracy score: {}".format(score))
Keras: Regression
Predict a Car’s Miles Per Gallon (MPG)

- Classic regression problem.
- Target: mpg
- Predictors:
  - cylinders
  - displacement
  - horsepower
  - weight
  - acceleration
  - year
  - origin
  - name
### Predict a Car’s Miles Per Gallon (MPG)

<table>
<thead>
<tr>
<th>mpg</th>
<th>cylinders</th>
<th>displacement</th>
<th>horsepower</th>
<th>weight</th>
<th>acceleration</th>
<th>year</th>
<th>origin</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>8</td>
<td>307</td>
<td>130</td>
<td>3504</td>
<td>12</td>
<td>70</td>
<td>1</td>
<td>chevrolet malibu</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>350</td>
<td>165</td>
<td>3693</td>
<td>11.5</td>
<td>70</td>
<td>1</td>
<td>buick skylark 320</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>318</td>
<td>150</td>
<td>3436</td>
<td>11</td>
<td>70</td>
<td>1</td>
<td>plymouth satellite</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>304</td>
<td>150</td>
<td>3433</td>
<td>12</td>
<td>70</td>
<td>1</td>
<td>amc rebel sst</td>
</tr>
<tr>
<td>17</td>
<td>8</td>
<td>302</td>
<td>140</td>
<td>3449</td>
<td>10.5</td>
<td>70</td>
<td>1</td>
<td>ford torino</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>429</td>
<td>198</td>
<td>4341</td>
<td>10</td>
<td>70</td>
<td>1</td>
<td>ford galaxie 500</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>454</td>
<td>220</td>
<td>4354</td>
<td>9</td>
<td>70</td>
<td>1</td>
<td>chevrolet impala</td>
</tr>
</tbody>
</table>
Regression Models - MPG

• Models such as GBM or Neural Network can predict the MPG to around +/-2.7 accuracy.
• Result of regression can be given in equation form (though not as accurate as a model):

\[
mpg = 0.002 \left( acc + \frac{1}{3}(-dsp - 1) - wgt \right) + 29.6
\]
Keras Regression: Load and Train/Test Split

```python
path = "./data/"

filename_read = os.path.join(path,"auto-mpg.csv")
df = pd.read_csv(filename_read,na_values=['NA','?'])

cars = df['name']
df.drop('name',1,inplace=True)
missing_median(df, 'horsepower')
x,y = to_xy(df,"mpg")
```
Keras Regression: Build and Fit

```python
model = Sequential()
model.add(Dense(10, input_dim=x.shape[1],
kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal'))
model.compile(loss='mean_squared_error', optimizer='adam')
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3,
patience=5, verbose=1, mode='auto')
model.fit(x,y,validation_data=(x_test,y_test),callbacks=[monitor],verbose=2,epochs=1000)
```
Keras Regression: Predict and Evaluate

# Predict
pred = model.predict(x_test)

# Measure RMSE error. RMSE is common for regression.
score = np.sqrt(metrics.mean_squared_error(pred, y_test))
print("Final score (RMSE): {}".format(score))
Preparing Data for Predictive Modeling is Hard

- The iris and MPG datasets are nicely formatted.
- Real world data is a complex mix of XML, JSON, textual formats, binary formats, and web service accessed content (the variety V in “Big Data”).
- More complex security data will be presented later in this talk.
Keras: Computer Vision and CNN
Predicting Images: What is Different?

- We will usually use classification, though regression is still an option.
- The input to the neural network is now 3D (height, width, color).
- Data are not transformed, no zscores or dummy variables.
- Processing time is usually much longer.
- We now have different layer types: dense layers (just like before), convolution layers and max pooling layers.
- Data will no longer arrive as CSV files. TensorFlow provides some utilities for going directly from image to the input of a neural network.
Sources of Image Data: CIFAR10 and CIFAR100
Sources of Image Data: ImageNet
Sources of Training Data: The MNIST Data Set
Recognizing Digits
Convolutional Neural Networks (CNN)

A LeNET-5/CNN Network (LeCun, 1998)

**Dense Layers** - Fully connected layers.

**Convolution Layers** - Used to scan across images.

**Max Pooling Layers** - Used to downsample images.

**Dropout Layer** - Used to add regularization.
Loading the Digits

(x_train, y_train), (x_test, y_test) = mnist.load_data()
print("Shape of x_train: {}".format(x_train.shape))
print("Shape of y_train: {}".format(y_train.shape))
print()
print("Shape of x_test: {}".format(x_test.shape))
print("Shape of y_test: {}".format(y_test.shape))

⇒ Shape of x_train: (60000, 28, 28)
⇒ Shape of y_train: (60000,)

⇒ Shape of x_test: (10000, 28, 28)
⇒ Shape of y_test: (10000,)
Display a Digit

```python
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
digit = 101  # Change to choose new digit
a = x_train[digit]
plt.imshow(a, cmap='gray', interpolation='nearest')
print("Image (#{}): Which is digit '{}".format(digit, y_train[digit]))
```
Build the CNN Network

```python
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```
Fit and Evaluate

```python
model.fit(x_train, y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=2,
    validation_data=(x_test, y_test))

score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss: {}\n'.format(score[0]))
print('Test accuracy: {}\n'.format(score[1]))

➢ Test loss: 0.03047790436172363
➢ Test accuracy: 0.9902
➢ Elapsed time: 1:30:40.79 (for CPU, approx 30 min GPU)
```
Keras: Time Series and RNN
How is a RNN Different?

- RNN = Recurrent Neural Network.
- LSTM = Long Short Term Memory.
- Most models will always produce the same output for the same input.
- Previous input does not matter to a non-recurrent neural network.
- To convert today’s temperature from Fahrenheit to Celsius, the value of yesterday’s temperature does not matter.
- To predict tomorrow’s closing price for a stock you need more than just today’s price.
- To determine if a packet is part of an attack, previous packets must be considered.
How do LSTM’s Work?

- The LSTM units in a deep neural network are short-term memory.
- This short term memory is governed by 3 gates:
  - Input Gate: When do we remember?
  - Output Gate: When do we act?
  - Forget Gate: When do we forget?
Sample Recurrent Data: Stock Price & Volume

\[
x =\[
\begin{array}{llllll}
[32,1383], [41,2928], [39,8823], [20,1252], [15,1532], \\
[35,8272], [32,1383], [41,2928], [39,8823], [20,1252], \\
[37,2738], [35,8272], [32,1383], [41,2928], [39,8823], \\
[34,2845], [37,2738], [35,8272], [32,1383], [41,2928], \\
[32,2345], [34,2845], [37,2738], [35,8272], [32,1383],
\end{array}
\]
\]

\[
y = [1,
    -1,
    0,
    -1,
    1]
\]
LSTM Example

```python
max_features = 4  # 0,1,2,3 (total of 4)
x = [
    [[0],[1],[1],[0],[0],[0]],
    [[0],[0],[0],[2],[2],[0]],
    [[0],[0],[0],[0],[3],[3]],
    [[0],[2],[2],[0],[0],[0]],
    [[0],[0],[3],[3],[0],[0]],
    [[0],[0],[0],[0],[1],[1]]
]
x = np.array(x,dtype=np.float32)
y = np.array([1,2,3,2,3,1],dtype=np.int32)
```
Build a LSTM

```python
model = Sequential()
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2, input_dim=1))
model.add(Dense(4, activation='sigmoid'))

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

def runit(model, inp):
    inp = np.array(inp, dtype=np.float32)
    pred = model.predict(inp)
    return np.argmax(pred[0])

print(runit(model, [[[0],[0],[0],[0],[3],[3]]]))
\rightarrow 3

print(runit(model, [[[4],[4],[0],[0],[0],[0]]]))
\rightarrow 4
GPU’s and Deep Learning
Low Level GPU Frameworks

• CUDA
  CUDA is NVidia's low-level GPGPU framework.

• OpenCL
  An open framework supporting CPU’s, GPU’s and other devices. Managed by the Khronos Group.
Thank you!

• Jeff Heaton
• http://www.jeffheaton.com