Visualizing machine learning models in Jupyter notebooks

JupyterCon 2018
August 23, 2018

Chakri Cherukuri
Senior Researcher
Quantitative Financial Research Group

TechAtBloomberg.com
Outline

• Introduction

• Overview of interactive widgets

• Example with code walk-through
  — Gaussian Density

• Machine learning algorithms/models
  — Kernel Regression
  — K-Means Clustering
  — PCA vs. Autoencoder
  — Tools for deep learning

• Q & A
Interactive widgets

- **ipywidgets**: core UI controls (text boxes, sliders, button etc.)

- **bqplot**: 2D plotting widgets (built on top of the ipywidgets framework)

- **Key takeaways**:  
  — ipywidgets and bqplot can be combined to build interactive plots  
  — Layout classes in ipywidgets can be used to layout/stack widgets  
  — Attributes of widgets can be linked using callbacks and the ‘observe’ method
Local Regression
Local Regression

• Non-parametric method for computing $E[Y|X]$

• Order 0: Local weighted average, using kernel as the weighting function

• Order 1 and above: Local weighted polynomial fit
  — Weighted least squares
  — Weights coming from the kernel function
Local Regression

• Gaussian kernel
  — Infinite support
  — Weights decay exponentially for far away points
  — Hyper-parameter ‘bandwidth’ provides bias-variance tradeoff

• Interactive visualization helps us understand:
  — Impact of ‘bandwidth’ and ‘polynomial order’ on the fit
  — Impact of outliers on the fit
K-Means Clustering
K-Means Clustering
Dimensionality Reduction
U.S. Treasury Yield Curve

- 11 tenors/maturities

- Different shapes
  - Pre-crisis
  - Post-crisis
  - Current
Dimensionality Reduction: PCA

• PCA:
  — Linear combinations of features
  — First few (3-5) factors enough to explain almost all the variance
Dimensionality Reduction: Autoencoder
PCA vs. Autoencoder
Dimension Reduction: AE vs. PCA
Tools for Deep Learning
Deep Learning Tools

• Graphical Wizard for building end-to-end deep learning pipeline
  — Select network parameters
  — Build network architecture
  — Training plots
  — Diagnostic plots
Network Parameters

- **Epochs**: 100
- **Batch Size**: 64
- **Loss**: mse
- **Optimizer**: adam

**Optimizer Parameters**
- \( l = 0.001 \)
- \( \beta_1 = 0.9 \)
- \( \beta_2 = 0.999 \)
- \( \epsilon_{opt} = 1e-8 \)
- \( \text{decay} = 0 \)
## Network Architecture

Here is a table showing the network architecture:

<table>
<thead>
<tr>
<th>Hidden Layers</th>
<th>Inputs</th>
<th>Hidden Layer 1</th>
<th>Hidden Layer 2</th>
<th>Hidden Layer 3</th>
<th>Hidden Layer 4</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>100</td>
<td>60</td>
<td>30</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>

**Activation**
- Hidden Layer 1: relu
- Hidden Layer 2: relu
- Hidden Layer 3: relu
- Hidden Layer 4: relu
- Outputs: linear

**Batch Norm**
- Hidden Layer 1: off
- Hidden Layer 2: off
- Hidden Layer 3: off
- Hidden Layer 4: off

**Dropout prob**
- Hidden Layer 1: 0
- Hidden Layer 2: 0
- Hidden Layer 3: 0
- Hidden Layer 4: 0
- Outputs: 0
Distributions of Weights/Biases/Activations
Diagnostic Plots
Resources

• Widget libraries used to build the applications:
  - ipywidgets: https://github.com/jupyter-widgets/ipywidgets
  - bqplot: https://github.com/bloomberg/bqplot

• Machine Learning libraries
  - scikit-learn: http://scikit-learn.org
  - tensorflow: https://www.tensorflow.org
  - keras: https://keras.io

• Link to the notebooks/code: https://github.com/ChakriCherukuri/jupytercon_2018

• Tech At Bloomberg blog: www.TechAtBloomberg.com
We are hiring!

https://www.bloomberg.com/careers

Questions?

TechAtBloomberg.com