Using CNTK's Python Interface for Deep Learning

dave.debarr (at) gmail.com

slides @ http://cross-entropy.net/PyData

2017-07-05

What drop out called it "deep learning hype" instead of "backpropaganda"?
-- Naomi Saphra / ML Hipster: https://twitter.com/ML_Hipster/status/729487995816935425

Topics to be Covered

- Cognitive Toolkit (CNTK) installation
- What is “machine learning”? [gradient descent example]
- What is “learning representations”?
- Why do Graphics Processing Units (GPUs) help?
- How do we prevent overfitting?
- CNTK Packages and Modules
- Deep learning examples, including Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) examples

What is “Machine Learning”?

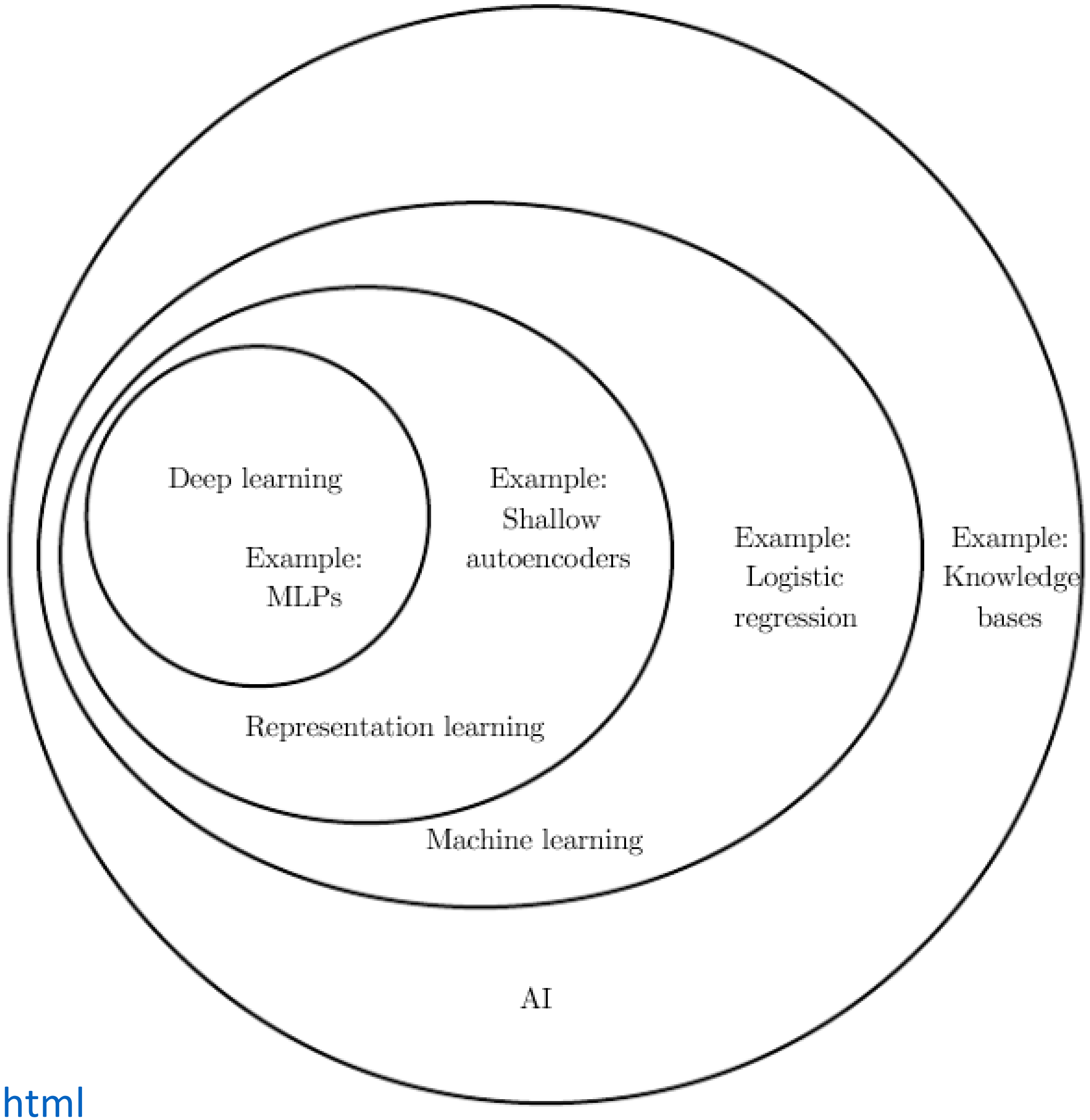
- Using data to create a model to map one-or-more input values to one-or-more output values

- Interest from many groups
  - Computer scientists: “machine learning”
  - Statisticians: “statistical learning”
  - Engineers: “pattern recognition”

Example Applications

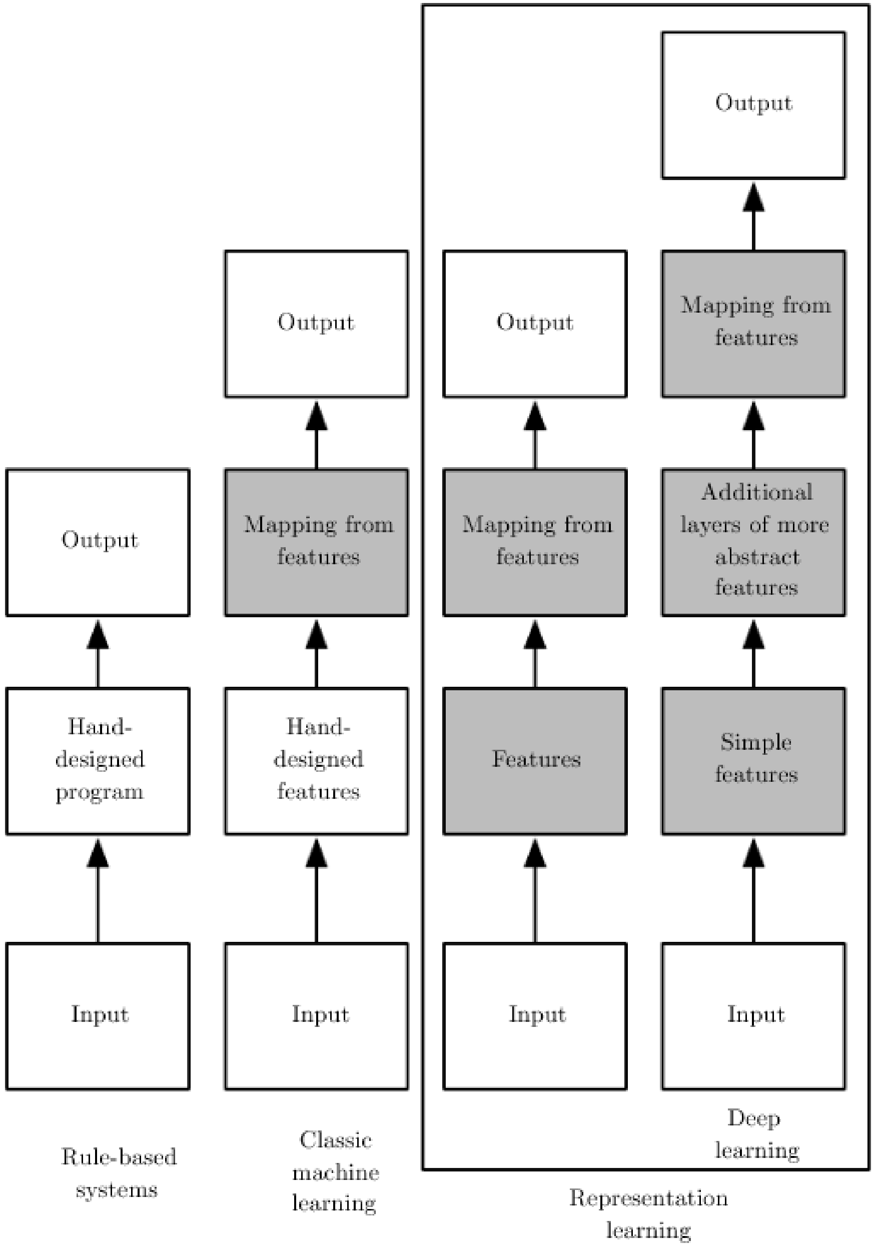
- Object detection
- Speech recognition
- Translation
- Natural language processing
- Recommendations
- Genomics
- Advertising
- Finance
- Security

Relationships



http://www.deeplearningbook.org/contents/intro.html

What is Deep Learning?



http://www.deeplearningbook.org/contents/intro.html

Machine Learning Taxonomy

- Supervised Learning: output is provided for observations used for training
  - Classification: the output is a categorical label [our focus for today is discriminative, parametric models]
  - Regression: the output is a numeric value
- Unsupervised Learning: output is not provided for observations used for training (e.g. customer segmentation)
- Semi-Supervised Learning: output is provided for some of the observations used for training
- Reinforcement Learning: rewards are provided to provide positive or negative reinforcement, with exploration used to seek an optimal mapping from states to actions (e.g. games)

A Word (or Two) About Tensors

- A tensor is just a generalization of an array
- Scalar: a value [float32 often preferred for working with Nvidia GPUs]
- Vector: a one-dimensional array of numbers
- Matrix: a two-dimensional array of numbers
- Tensor: may contain three or more dimensions
  - Array of images with Red Green Blue (RGB) channels
  - Array of documents with each word represented by an “embedding”

A Word (or Two) About Dot Products

- The “dot product” between 2 vectors (one-dimensional arrays of numeric values) is defined as the sum of products for the elements:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

- The dot product measures the similarity between the two vectors
- The dot product is an unnormalized version of the cosine of the angle between two vectors, where the cosine takes on the maximum value of +1 if the two vectors “point” in the same direction; or the cosine takes on the minimum value of -1 if the two vectors “point” in opposite directions

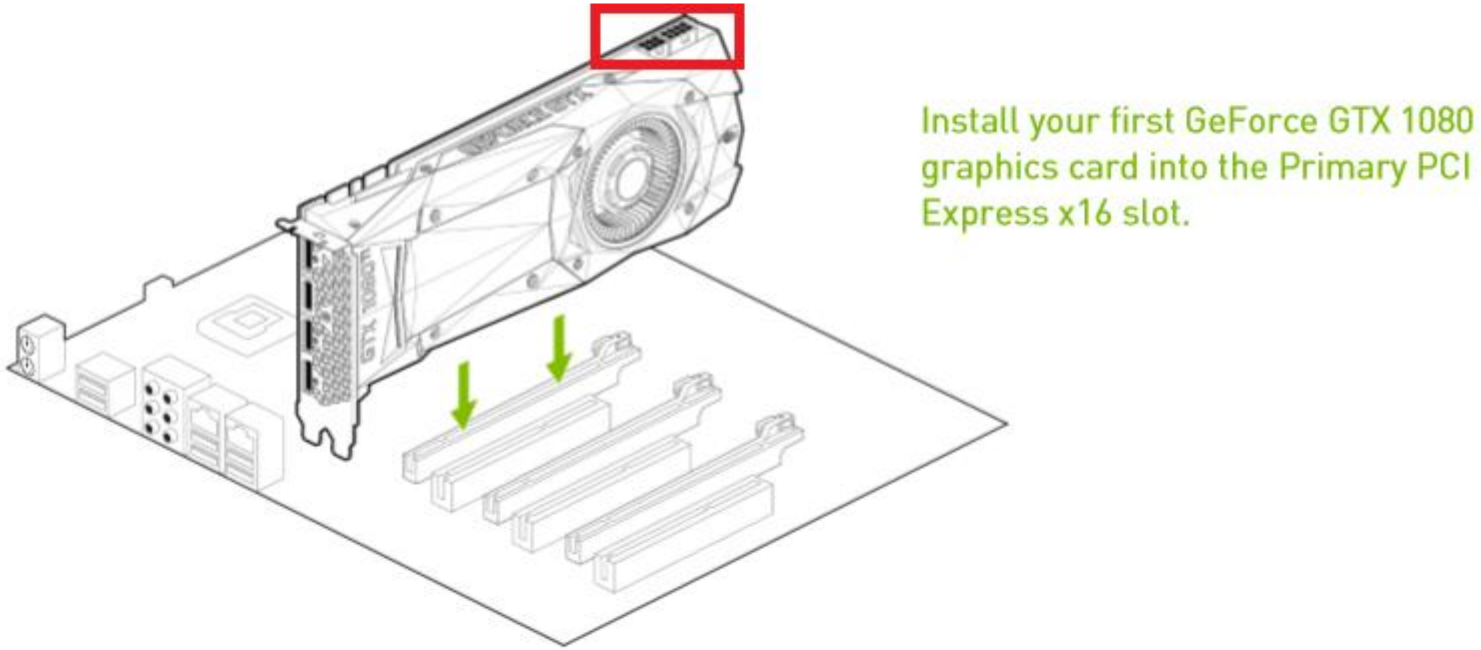
Getting Access to a Platform with a GPU

- Graphics Processing Units (GPUs) often increase the speed of tensor manipulation by an order of magnitude, because deep learning consists of lots of easily parallelized operations (e.g. matrix multiplication)
- GPUs often have thousands of processors, but they can be expensive
  - If you're just playing for a few hours, Azure is probably the way to go [rent someone else's GPU]
  - If you're a recurring hobbyist, consider buying an Nvidia card (cores; memory)
    - GTX 1050 Ti (768; 4GB): $150 [no special power requirements]
    - GTX 1070 (1920; 8GB): $400 [requires a separate power connector]
    - GTX 1080 Ti (3584; 11GB): $700
    - Titan Xp (3840; 12GB): $1200
- Will cover Azure VM here: don't forget to delete it when you're done!

Nvidia GTX 1080 Ti Card

In case you're buying a card …

- Fits in Peripheral Component Interconnect (PCI) Express x16 slot; but …
- fancier cards require separate power connectors

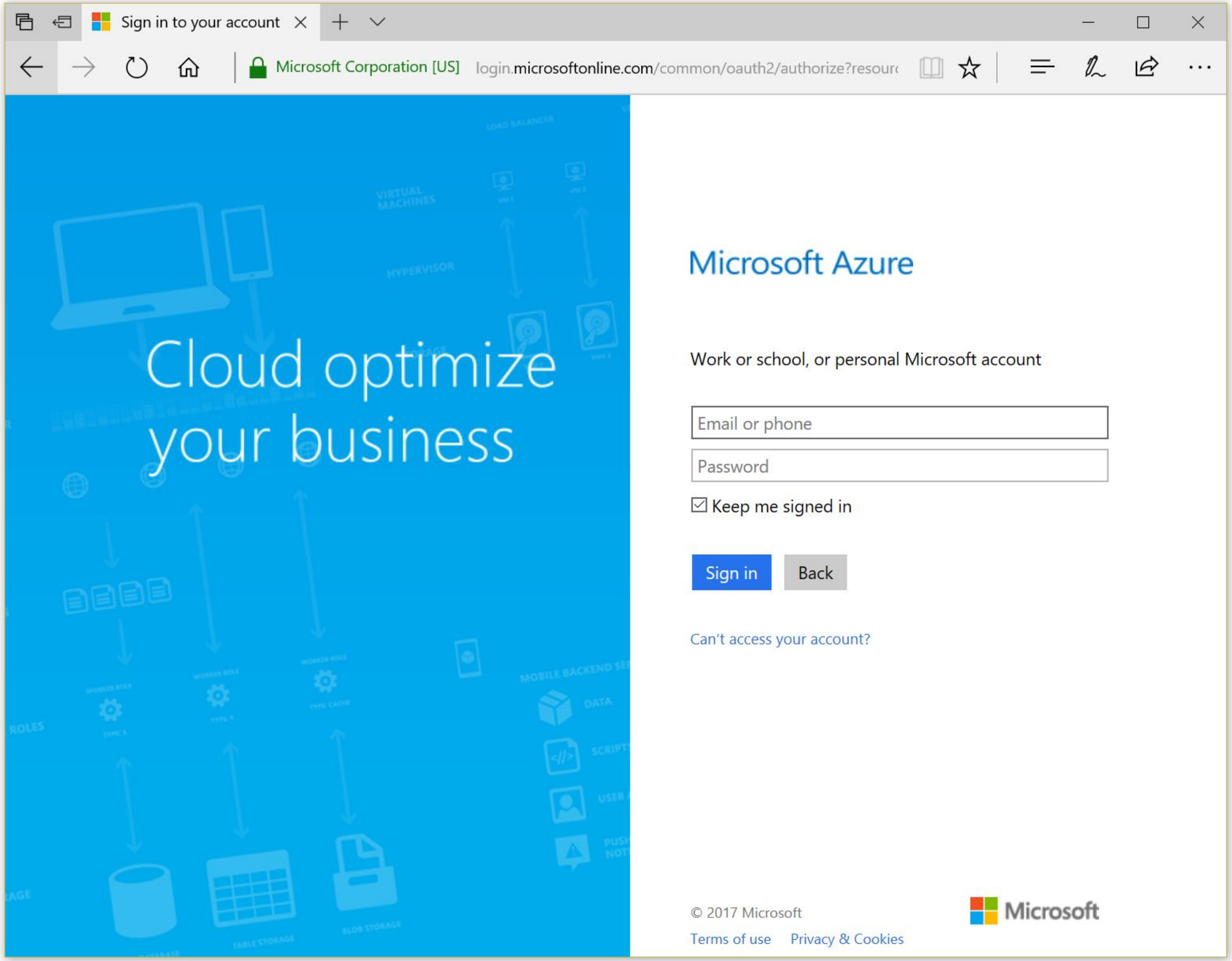


http://www.nvidia.com/content/geforce-gtx/GTX_1080_Ti_User_Guide.pdf

PyData

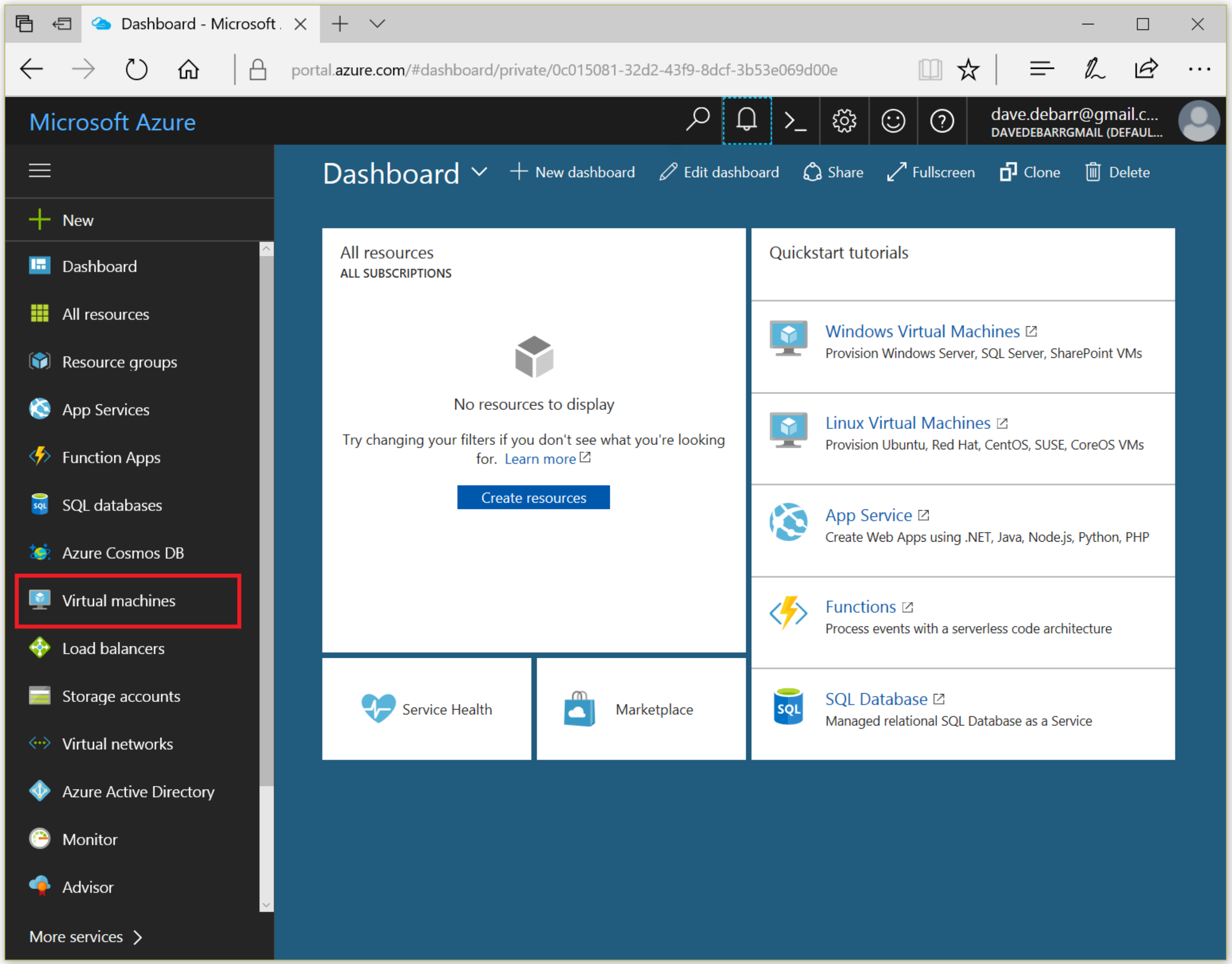
https://azure.microsoft.com/en-us/pricing/details/virtual-machines/windows/
https://azure.microsoft.com/en-us/regions/services/ [NC6 (Ubuntu): $0.9/hour]

Azure: Sign In

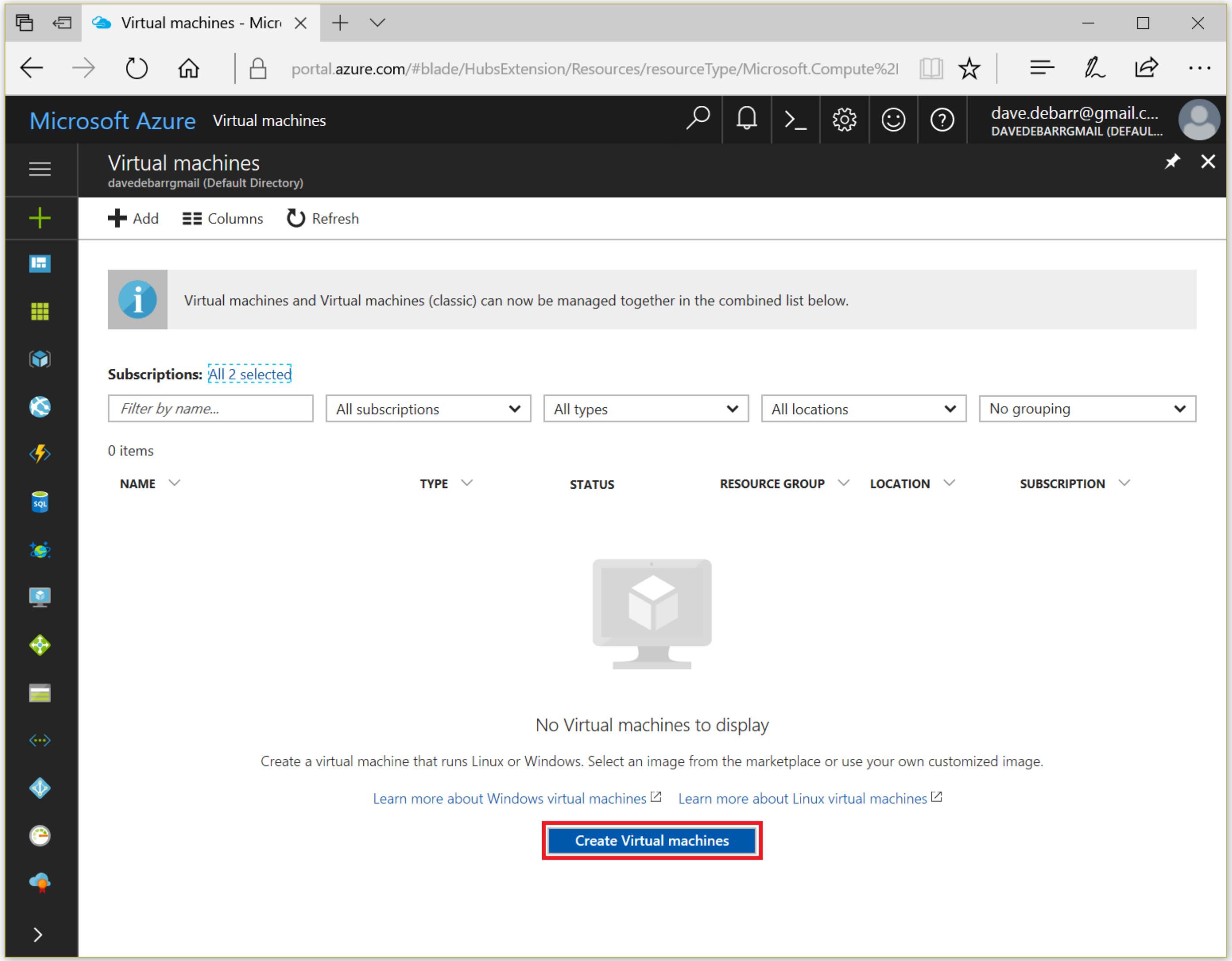


https://portal.azure.com/

Select "Virtual machines" (on the left)

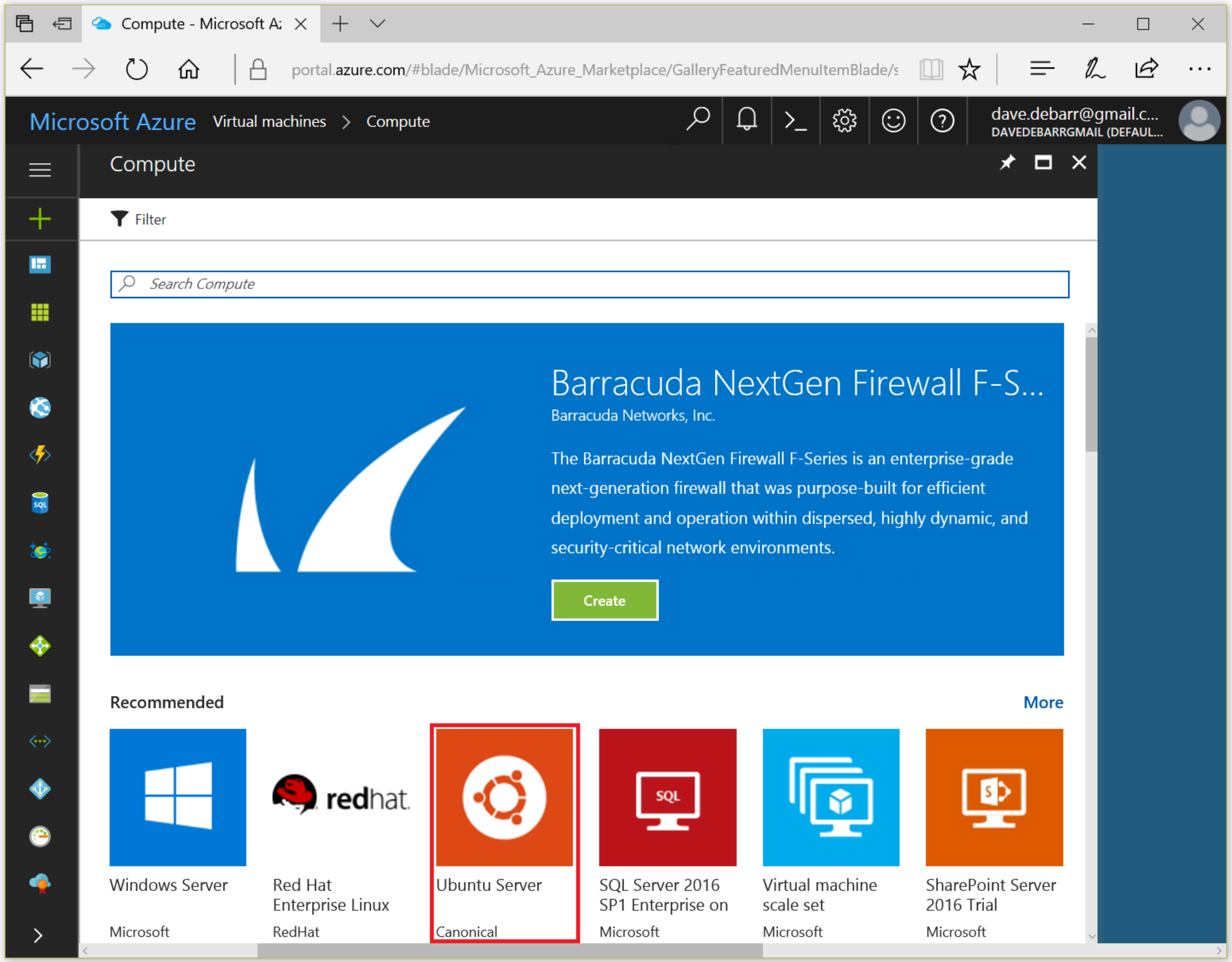


Select “Create Virtual machines”

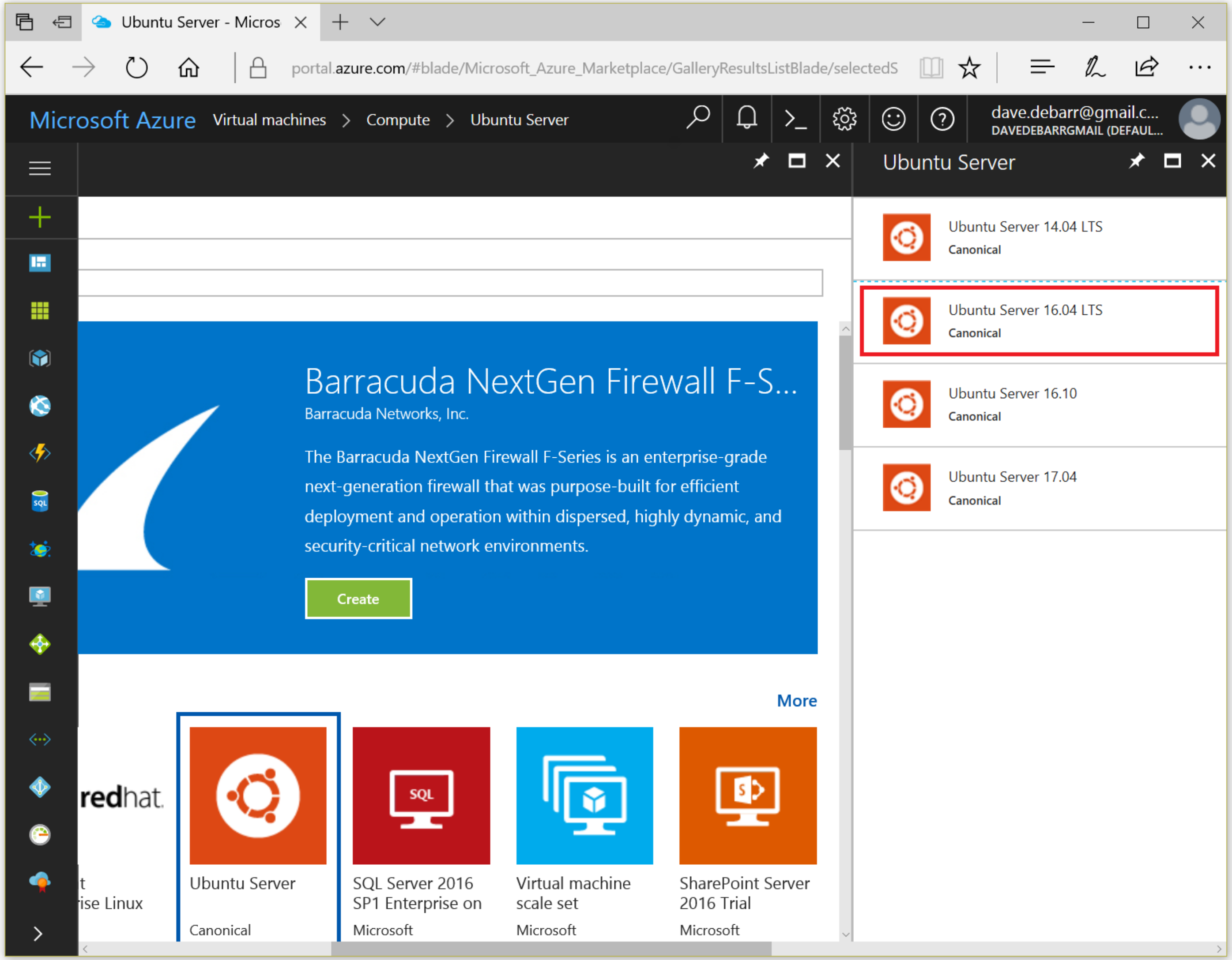


PyData

Select “Ubuntu Server”

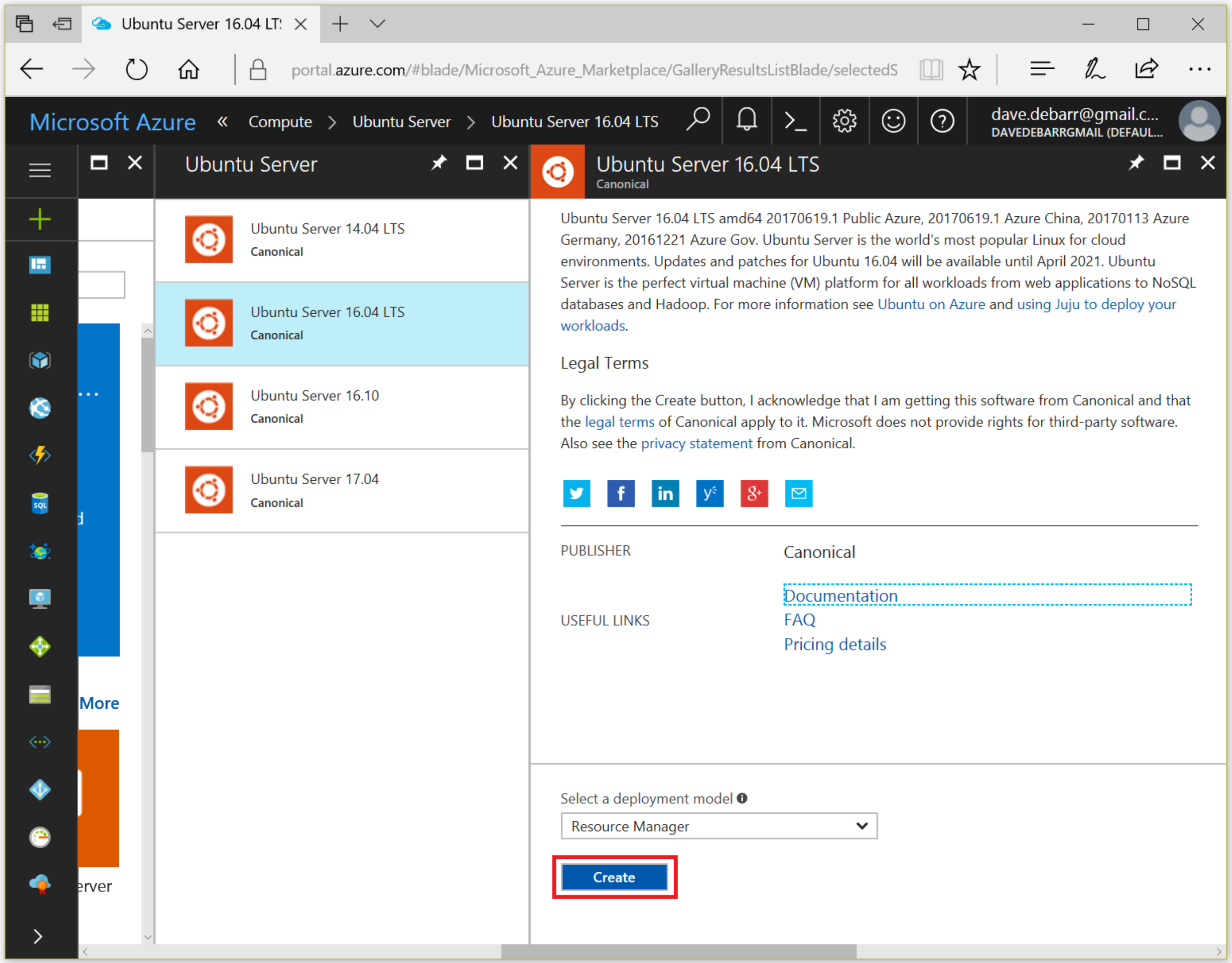


Select “Ubuntu Server 16.04 LTS”

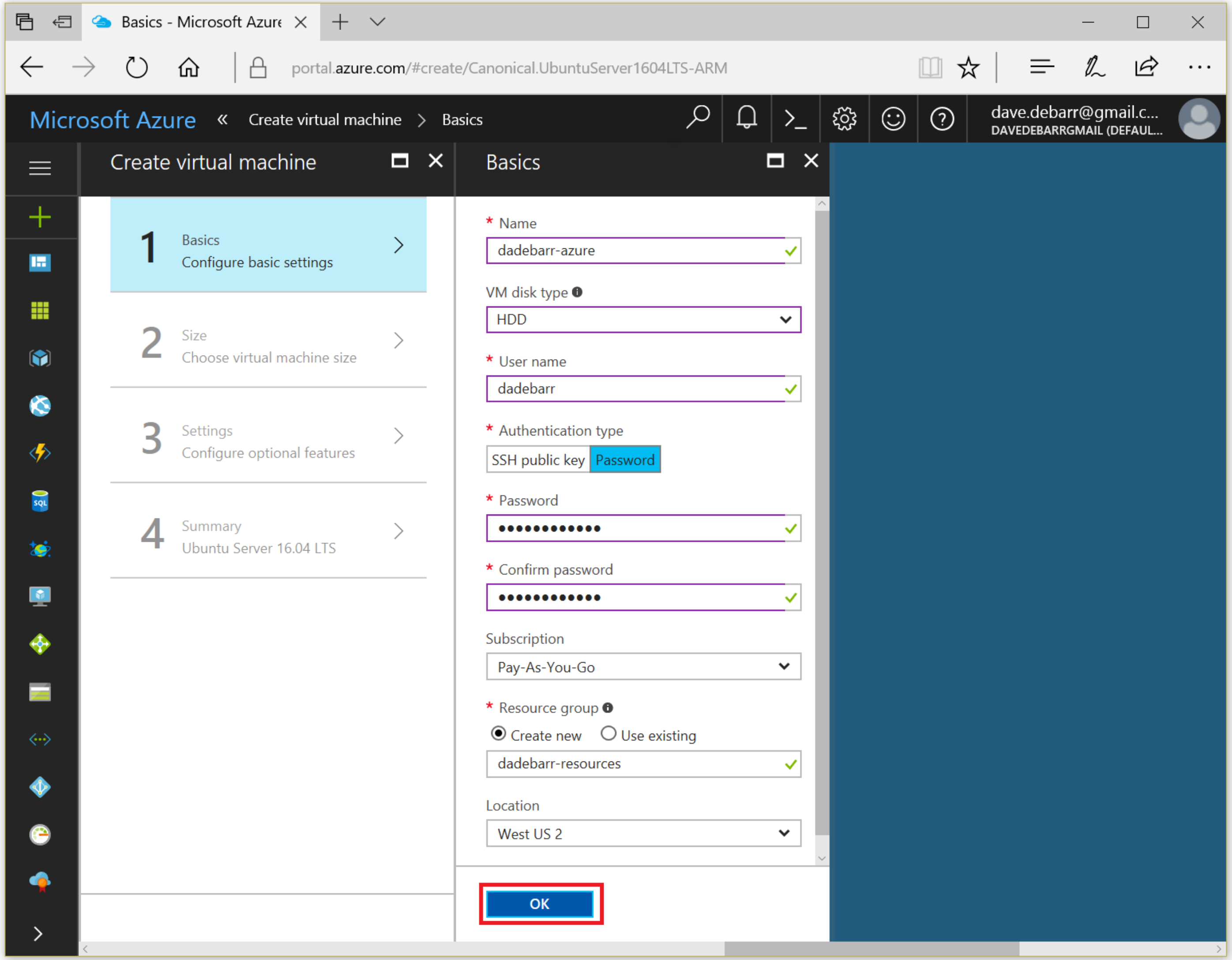


LTS: Long Term Support

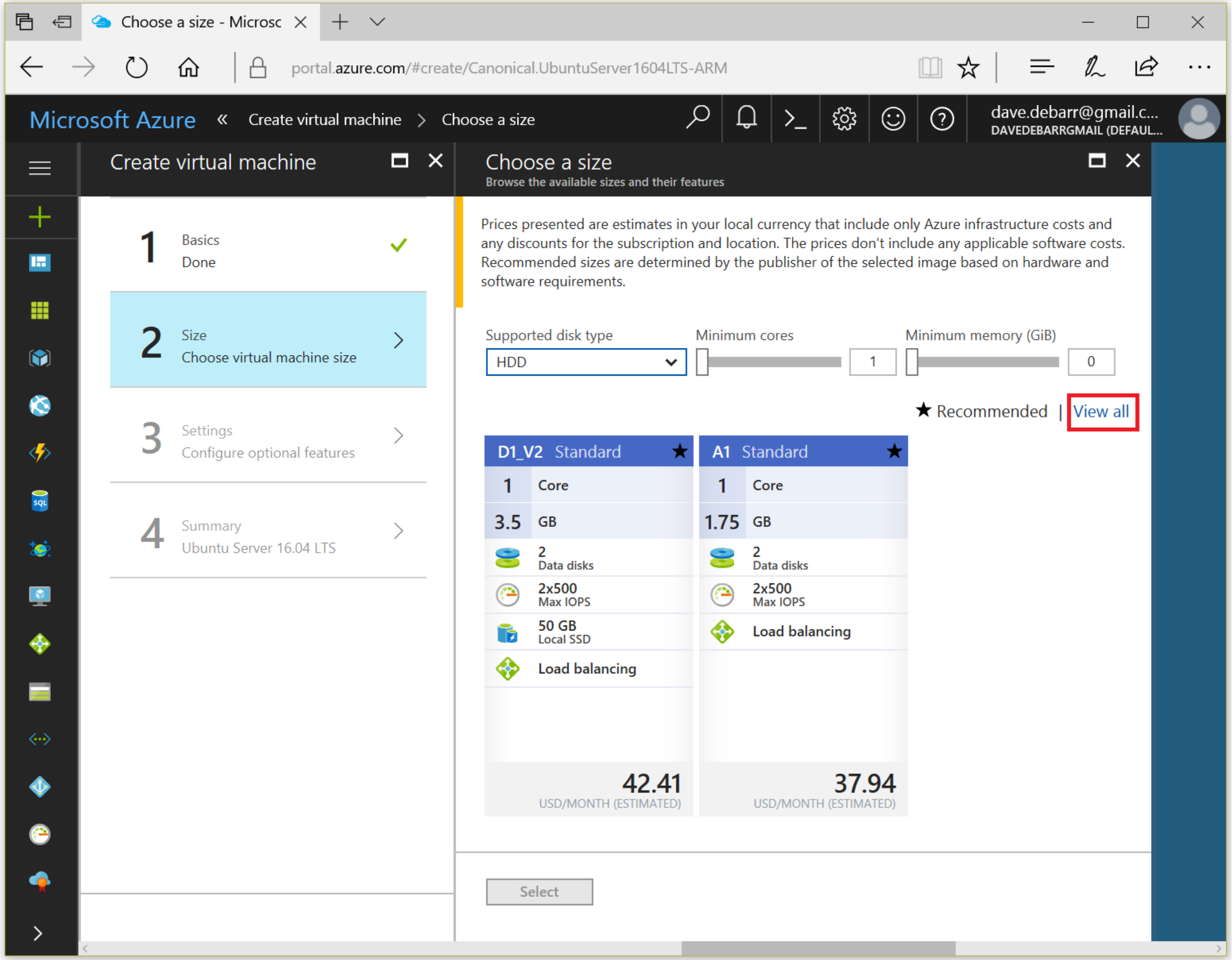
Select the “Create” Button



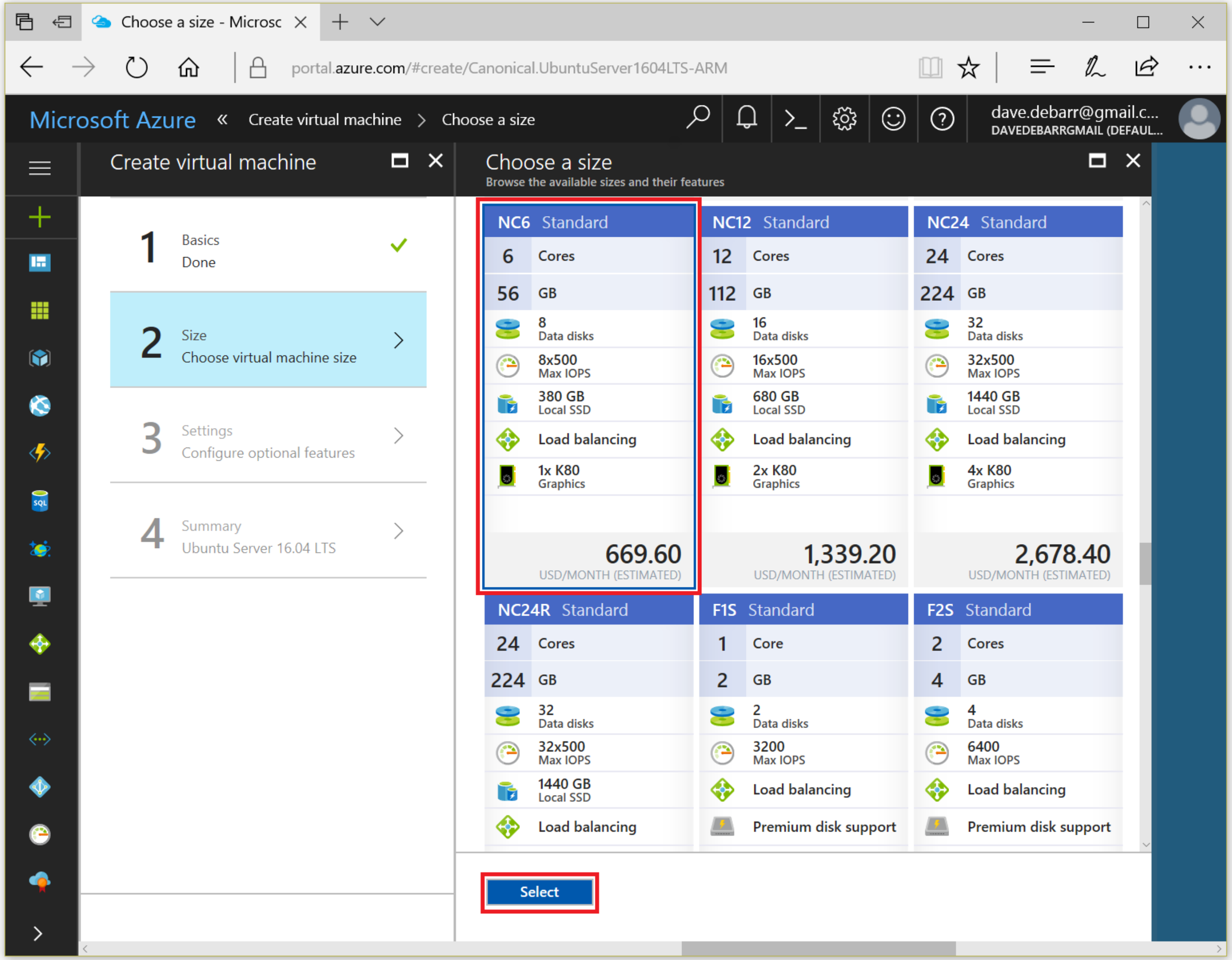
Configure the Virtual Machine



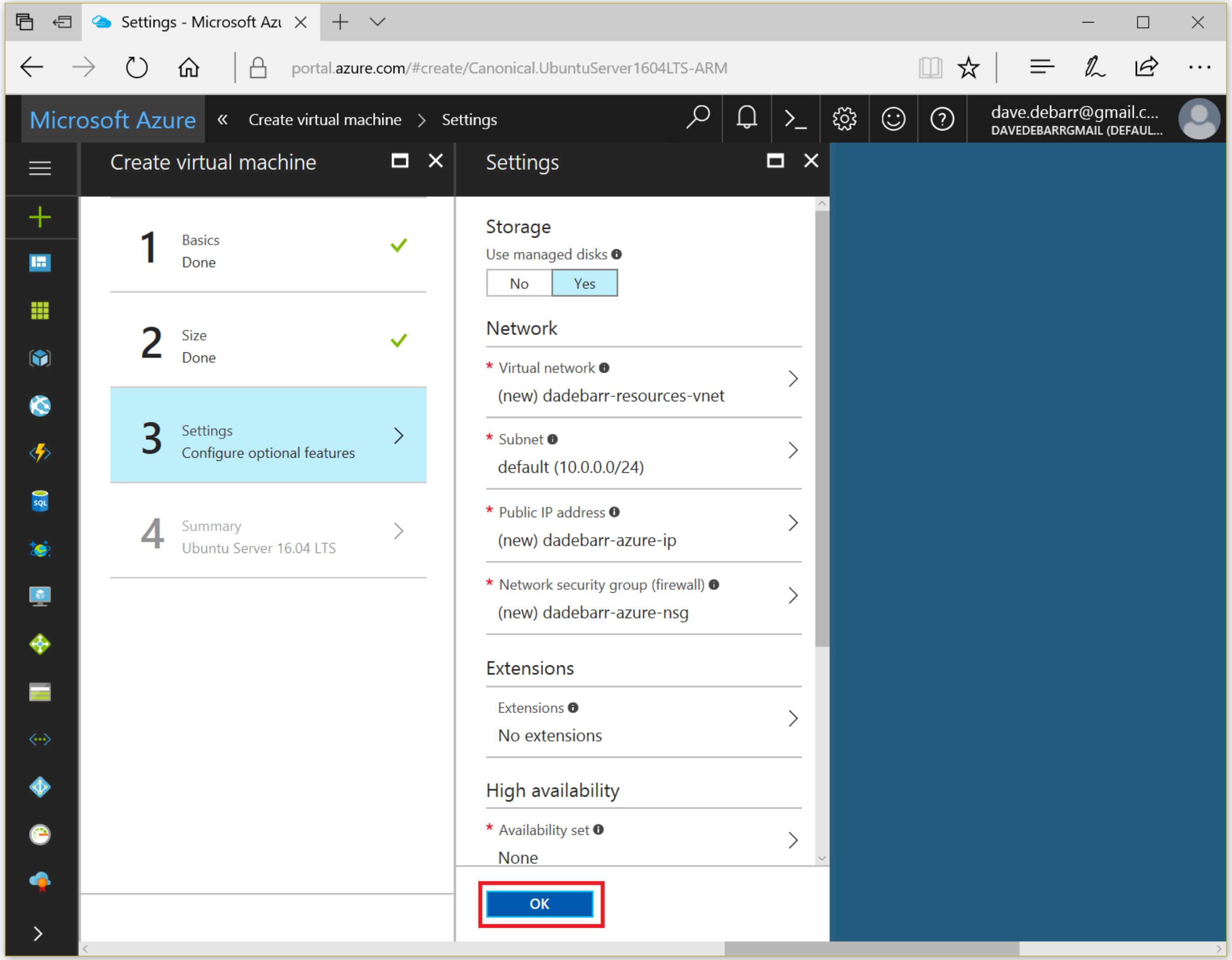
Select "View all" (on the right)



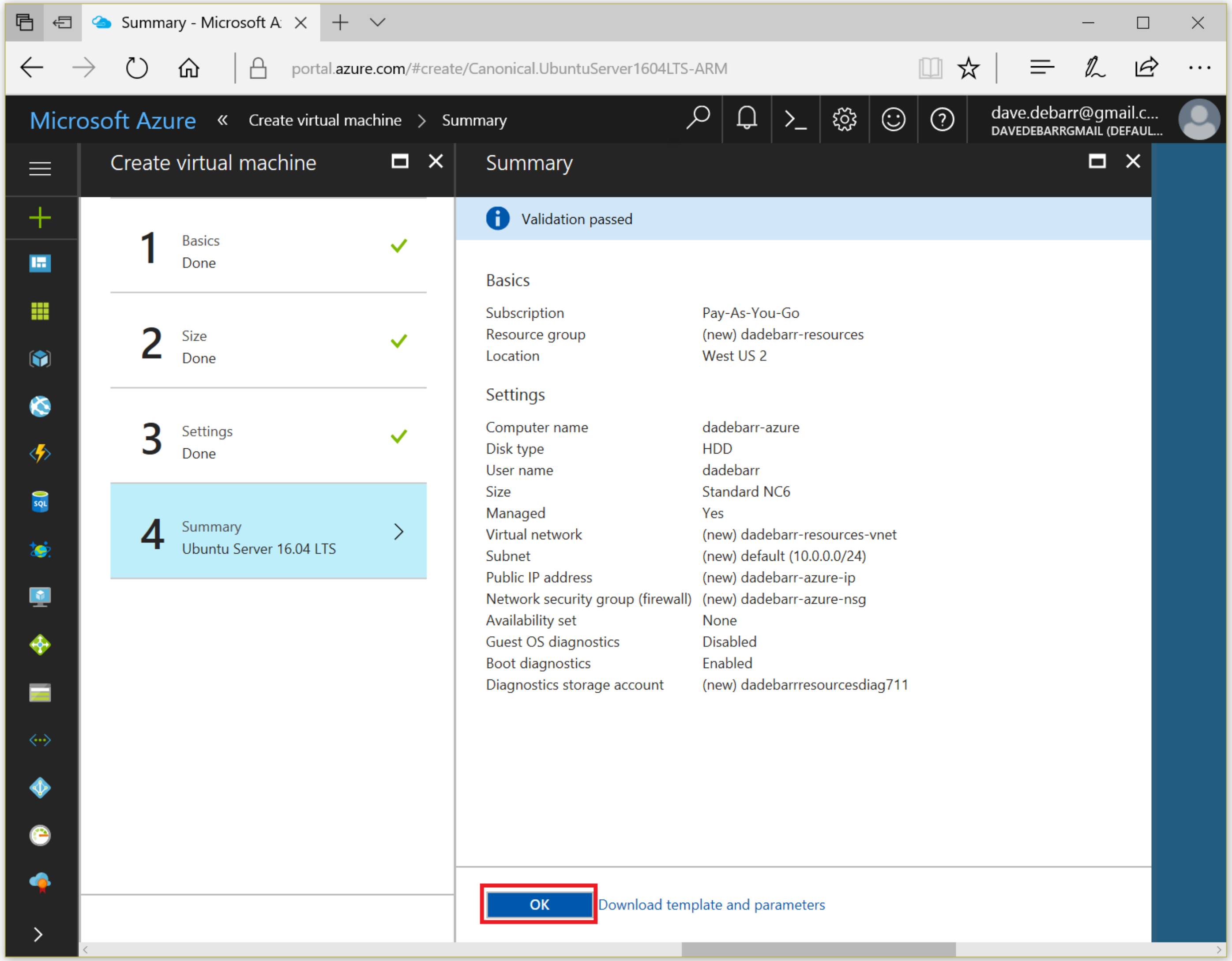
Select “NC6” Virtual Machine (VM)



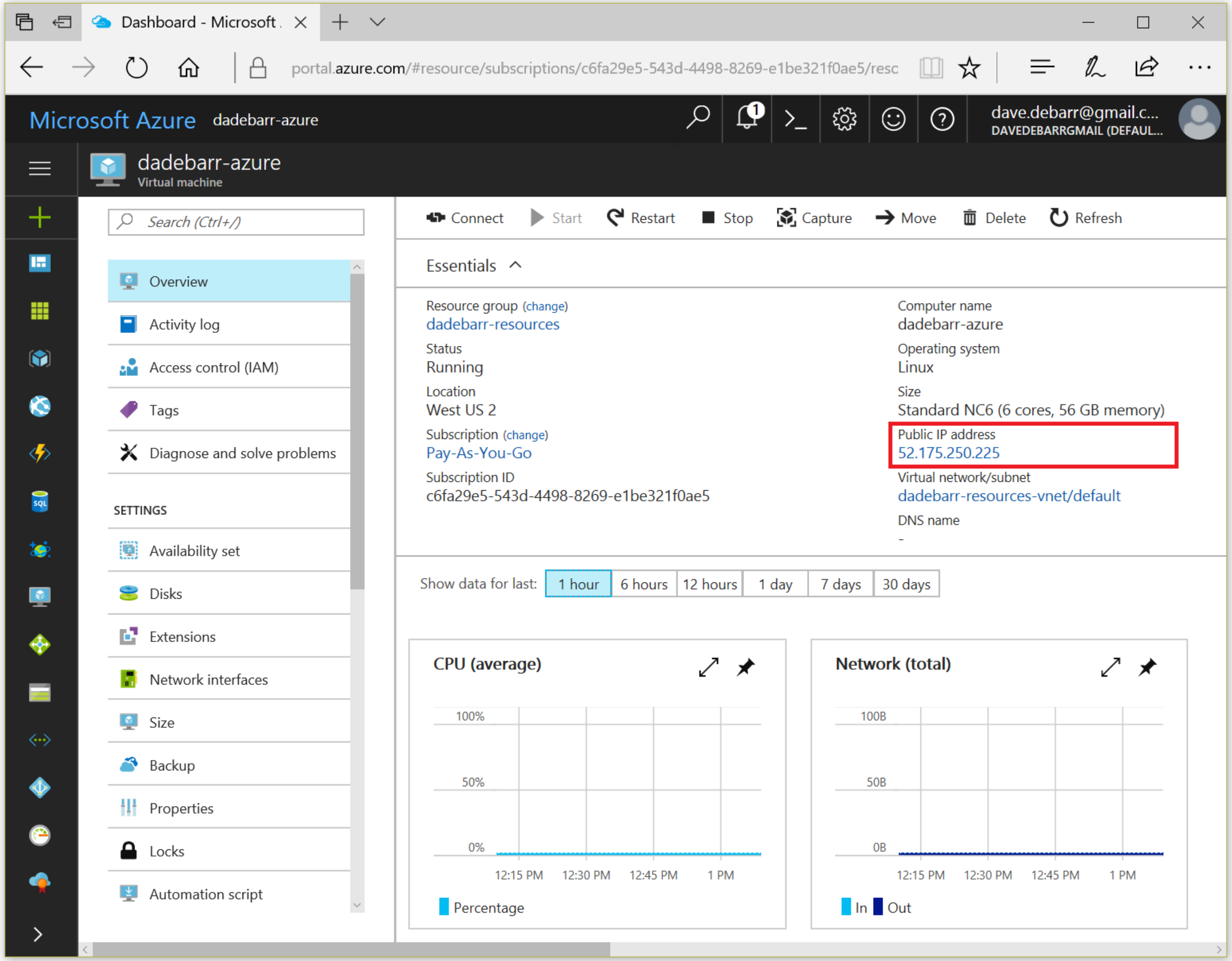
Configure “Settings”



Acknowledge “Summary”



Take Note of "Public IP address"



Install Support Software

https://docs.microsoft.com/en-us/azure/virtual-machines/linux/n-series-driver-setup#install-cuda-drivers-for-nc-vms

- Download PuTTY [secure shell (ssh) software: optional (client)]
  - ftp://ftp.chiark.greenend.org.uk/users/sgtatham/putty-latest/w32/putty-0.69-installer.msi
  - When using ssh, check the "Connection > SSH> X11: Enable X11 Forwarding" option
- Download Xming X Server for Windows [optional (client)]
  - https://sourceforge.net/projects/xming/files/latest/download
- Configure the Nvidia driver [required (server)]

```
CUDA_REPO_PKG=cuda-repo-ubuntu1604_8.0.61-1_amd64.deb
wget -O /tmp/${CUDA_REPO_PKG} \
   http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/${CUDA_REPO_PKG}
sudo dpkg -i /tmp/${CUDA_REPO_PKG}
rm -f /tmp/${CUDA_REPO_PKG}
sudo apt-get update
sudo apt-get install cuda-drivers
sudo apt-get install cuda
```

CUDA: Compute Unified Device Architecture

nvidia-smi

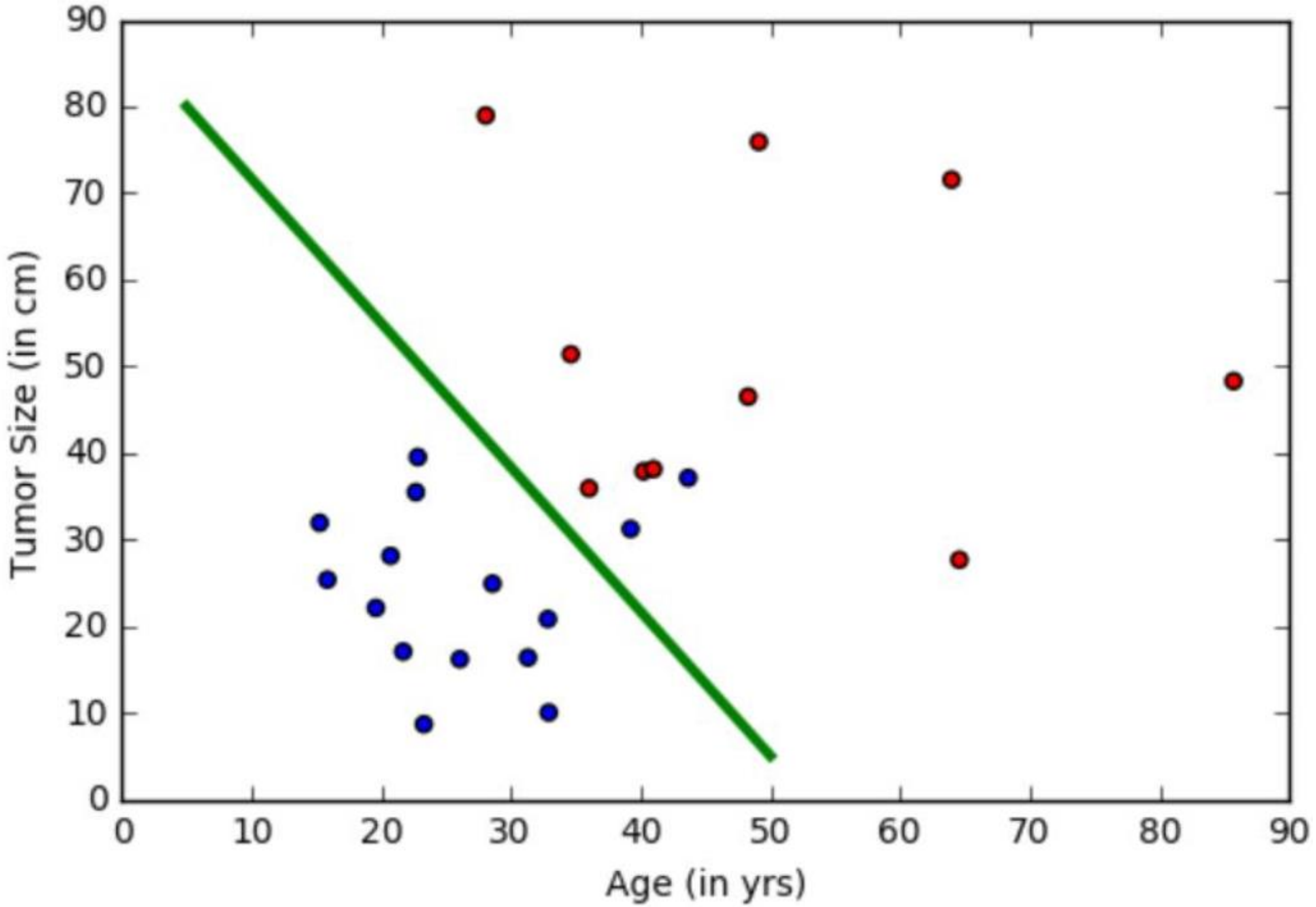
```
dadebarr@dadebarr-azure:~$ nvidia-smi
Sun Jul  2 20:50:59 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.66                 Driver Version: 375.66                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | A2B4:00:00.0     Off |                    0 |
| N/A   39C    P0    72W / 149W |      0MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

NC6 has access to one of the two Nvidia K80 GPUs: 2496 cores; 12 GB memory
https://images.nvidia.com/content/pdf/kepler/Tesla-K80-BoardSpec-07317-001-v05.pdf

SMI: System Management Interface

Logistic Regression Tutorial Example

https://gallery.cortanaintelligence.com/Collection/Cognitive-Toolkit-Tutorials-Collection



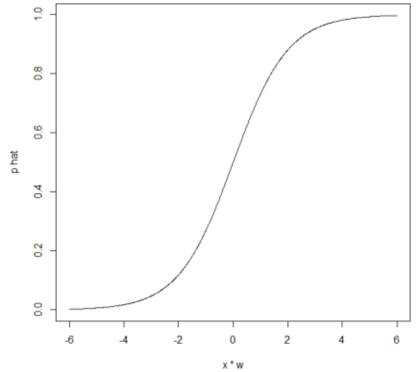
Logistic Regression

- Logistic regression is a shallow, linear model
  - Consists of a single “layer” with a single “sigmoid” activation function
  - Cross entropy is used as a loss function: the objective function used to drive “training” (i.e. updating the weights)
- We will use Stochastic Gradient Descent (SGD) in our example today, because this is the core learning method used for training deep learning models; but most “logistic regression” packages use a method known as Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization [an approximation of Iteratively Reweighted Least Squares (IRLS)]

PyData

The Logistic Regression Model

The "sigmoid" function is used to map input features to a predicted probability of class membership



$$\hat{p} = \frac{1}{1 + exp(-\boldsymbol{x}^T\boldsymbol{w})}$$

… where …

- $\boldsymbol{x}^T\boldsymbol{w}$ is a "dot product", a measure of the similarity between two vectors; an unnormalized measure of the cosine of the angle between the feature vector and the model's weight vector [the weight vector points in the direction of the "positive" class]
- $\hat{p}$ is an estimate of the probability that the input vector belongs to the positive class

Learning by Gradient Descent

- The gradient of the loss function is used to update the weights of the model

- The gradient of the loss function tells us how to maximize the loss function, so the negative of the gradient is used to minimize the loss function

The Cross Entropy Loss Function

- This function is used to measure the dissimilarity between two distributions
- In the context of evaluating pattern recognition models, we are using this function to measure the dissimilarity of the target class indicator and the predicted probability for the target class

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

https://www.kaggle.com/wiki/LogLoss

Gradient Descent for Logistic Regression (1/4)

The cross entropy function, the function used for evaluating the quality of a prediction, can be expressed as …

$$
\begin{aligned}
&-\log\left(\Pr\left(y_i^* = 1 \mid \mathbf{x}_i; \mathbf{w}\right)\right) \\
&= -\log\left(\left(\frac{1}{1+\exp\left(-\mathbf{x}_i^T\mathbf{w}\right)}\right)^{y_i^*}\left(1-\frac{1}{1+\exp\left(-\mathbf{x}_i^T\mathbf{w}\right)}\right)^{\left(1-y_i^*\right)}\right) \\
&= -\log\left(\frac{1}{1+\exp\left(-y_i\mathbf{x}_i^T\mathbf{w}\right)}\right) \\
&= \log\left(1+\exp\left(-y_i\mathbf{x}_i^T\mathbf{w}\right)\right)
\end{aligned}
$$

$$y_i = \{-1, +1\}$$

$$y_i^* = \frac{y_i + 1}{2}$$

Gradient Descent for Logistic Regression (2/4)

The derivative of the loss function with respect to a parameter indicates how to update a weight to optimize the loss function …

$$\nabla_{\mathbf{w}} \log\left(1 + \exp\left(-y_i \mathbf{x}_i^T \mathbf{w}\right)\right)$$

$$= \left[\frac{\partial}{\partial w_1} \log\left(1 + \exp\left(-y_i \mathbf{x}_i^T \mathbf{w}\right)\right) \quad \cdots \quad \frac{\partial}{\partial w_p} \log\left(1 + \exp\left(-y_i \mathbf{x}_i^T \mathbf{w}\right)\right)\right]$$

[the machine “learns” by updating the weights to minimize the loss function]

Gradient Descent for Logistic Regression (3/4)

So we update a weight by subtracting the product of the input feature value and the difference between the predicted probability and the class membership indicator …

$$\frac{\partial}{\partial w_i} \log\left(1+\exp\left(-y_i \hat{f}\left(x_i\right)\right)\right)$$

$$= \frac{\partial}{\partial \hat{f}\left(x_i\right)} \log\left(1+\exp\left(-y_i \hat{f}\left(x_i\right)\right)\right) \frac{\partial}{\partial w_i} \hat{f}\left(x_i\right)$$

$$= \frac{\partial}{\partial \hat{f}\left(x_i\right)} \log\left(1+\exp\left(-y_i \hat{f}\left(x_i\right)\right)\right) \frac{\partial}{\partial w_i} x_i w_i$$

$$= \left(\frac{1}{1+\exp\left(-y_i \hat{f}\left(x_i\right)\right)} - y_i^*\right) x_i$$

Gradient Descent for Logistic Regression (4/4)

Showing steps of differentiation for completeness …

$$\frac{\partial}{\partial \hat{f}(x_i)} \log\left(1+\exp\left(-y_i \hat{f}(x_i)\right)\right)$$

$$= \frac{1}{1+\exp\left(-y_i \hat{f}(x_i)\right)} \left(\frac{\partial}{\partial \hat{f}(x_i)} 1 + \frac{\partial}{\partial \hat{f}(x_i)} \exp\left(-y_i \hat{f}(x_i)\right) \right)$$

$$= \frac{1}{1+\exp\left(-y_i \hat{f}(x_i)\right)} \left(0 + \exp\left(-y_i \hat{f}(x_i)\right) \frac{\partial}{\partial \hat{f}(x_i)} \left(-y_i \hat{f}(x_i)\right) \right)$$

$$= \frac{1}{1+\exp\left(-y_i \hat{f}(x_i)\right)} \left(0 + \exp\left(-y_i \hat{f}(x_i)\right) (-y_i) \right)$$

$$= -y_i \frac{\exp\left(-y_i \hat{f}(x_i)\right)}{1+\exp\left(-y_i \hat{f}(x_i)\right)}$$

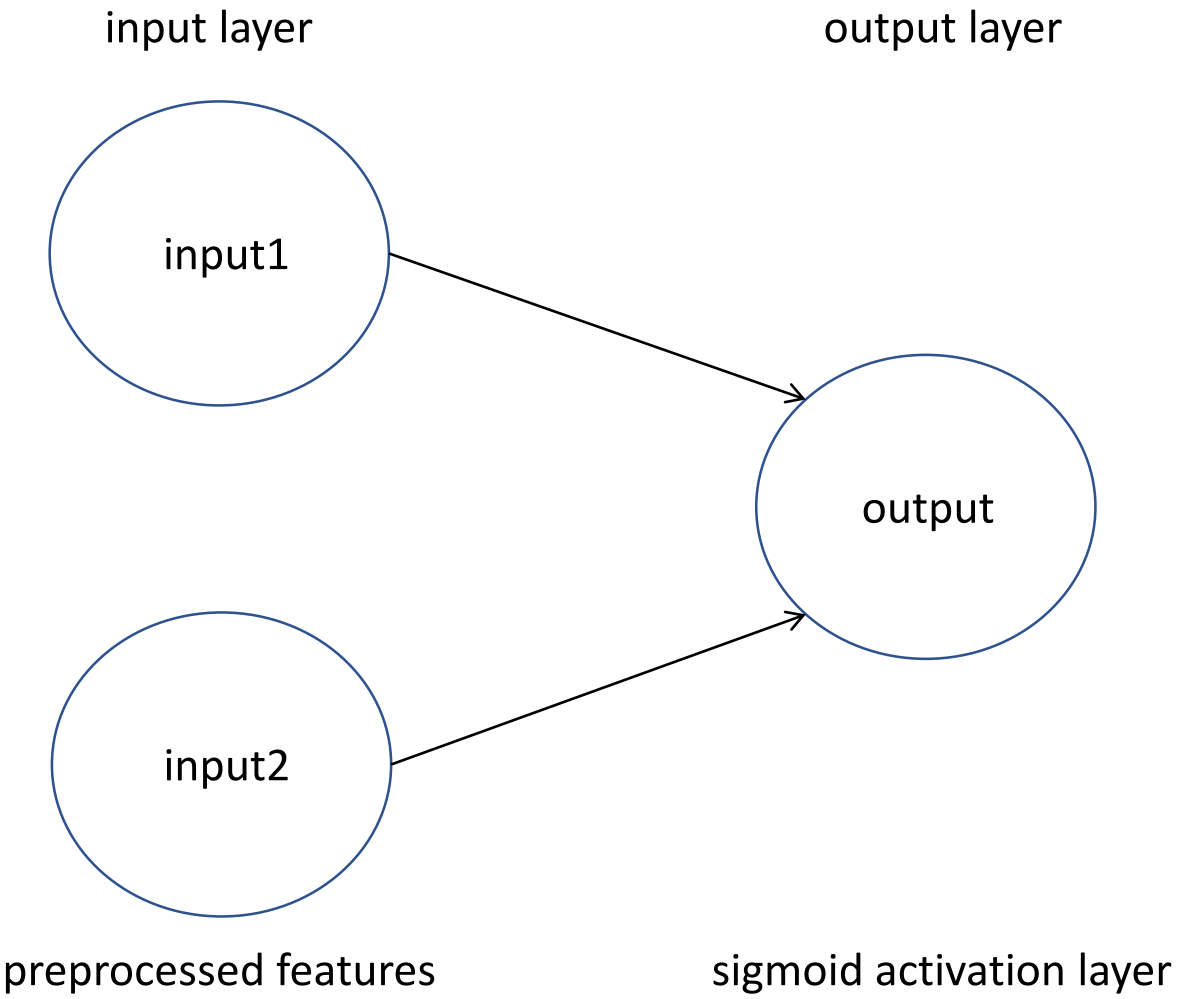
$$= -y_i \frac{1}{1+\exp\left(y_i \hat{f}(x_i)\right)}$$

$$= -y_i \left(1 - \frac{1}{1+\exp\left(-y_i \hat{f}(x_i)\right)} \right)$$

$$= \frac{1}{1+\exp\left(-y_i \hat{f}(x_i)\right)} - y_i^*$$

http://www.derivative-calculator.net/

Logistic Regression Example



Simple SGD in Python

- $HOME/anaconda3/bin/jupyter notebook
- http://cross-entropy.net/PyData/
- 01_SGD.ipynb

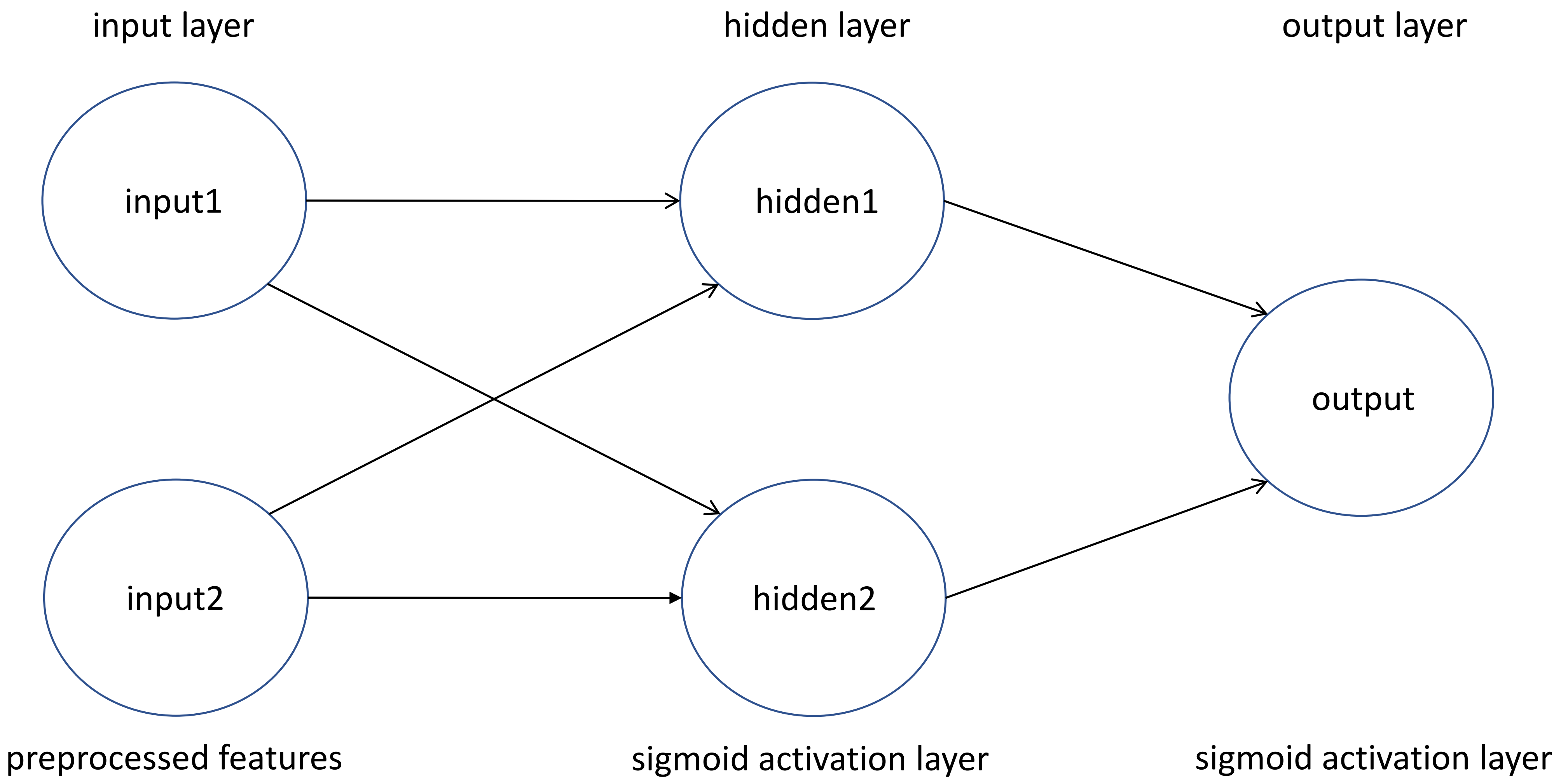
Stratifying Gradient Descent

- Stochastic Gradient Descent (SGD): a randomly selected training set observation is used to update the weights of the model

- Batch Gradient Descent: all training set observations are used to update the weights of the model [better updates but more computationally intensive than SGD]

- Mini-Batch Stochastic Gradient Descent: a subset of the training set is used to update the weights of the model [a compromise; this is the most popular version]

Multi-Layer Perceptron (MLP) Example



Simple MLP in Python

- 02_Backpropagation.ipynb

Backpropagation Description

After the forward computation, compute the gradient on the output layer:

$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y})$

for $k = l, l-1, \ldots, 1$ **do**

Convert the gradient on the layer's output into a gradient into the pre-nonlinearity activation (element-wise multiplication if f is element-wise):

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$

Compute gradients on weights and biases (including the regularization term, where needed):

$\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g} + \lambda \nabla_{\boldsymbol{b}^{(k)}} \Omega(\theta)$

$\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\, \boldsymbol{h}^{(k-1)\top} + \lambda \nabla_{\boldsymbol{W}^{(k)}} \Omega(\theta)$

Propagate the gradients w.r.t. the next lower-level hidden layer's activations:

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)\top}\, \boldsymbol{g}$

end for

http://www.deeplearningbook.org/contents/mlp.html

Install CNTK

https://docs.microsoft.com/en-us/cognitive-toolkit/Setup-Linux-Binary-Manual

```
sudo apt-get install openmpi-bin
wget https://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86_64.sh
/bin/bash Anaconda3-4.1.1-Linux-x86_64.sh
    [press Enter]
    [press the spacebar]
    [Enter "yes" to access the license terms]
    [press Enter to accept the default directory for installation: $HOME/anaconda3]
    [Enter "yes" to prepend python to your program search path: $HOME/anaconda3/bin]
pip install https://cntk.ai/PythonWheel/GPU/cntk-2.0-cp35-cp35m-linux_x86_64.whl
sudo apt-get install chromium-browser
```

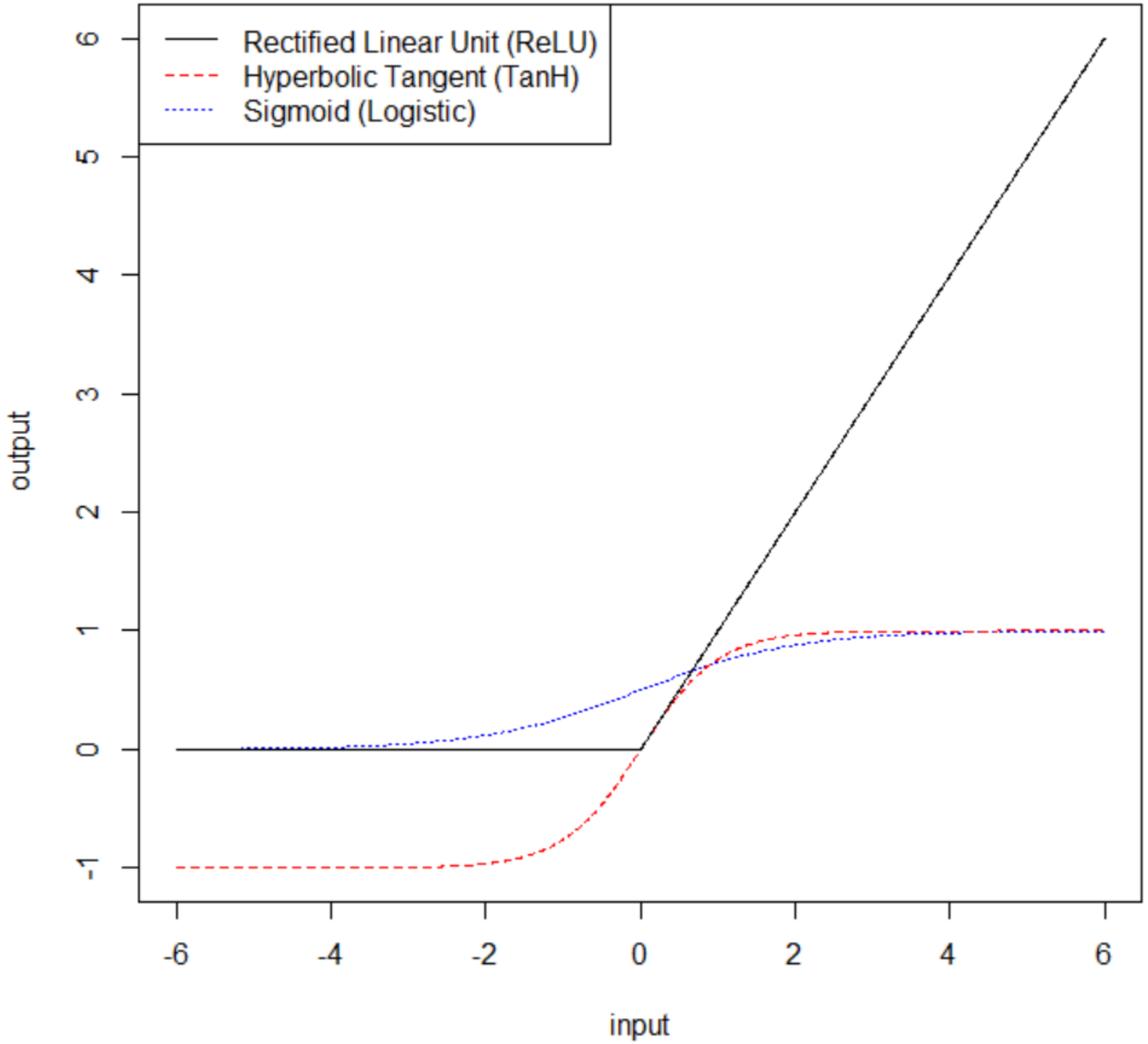
MLP Example

- 03_MLP_CNTK.ipynb

Learning Representations

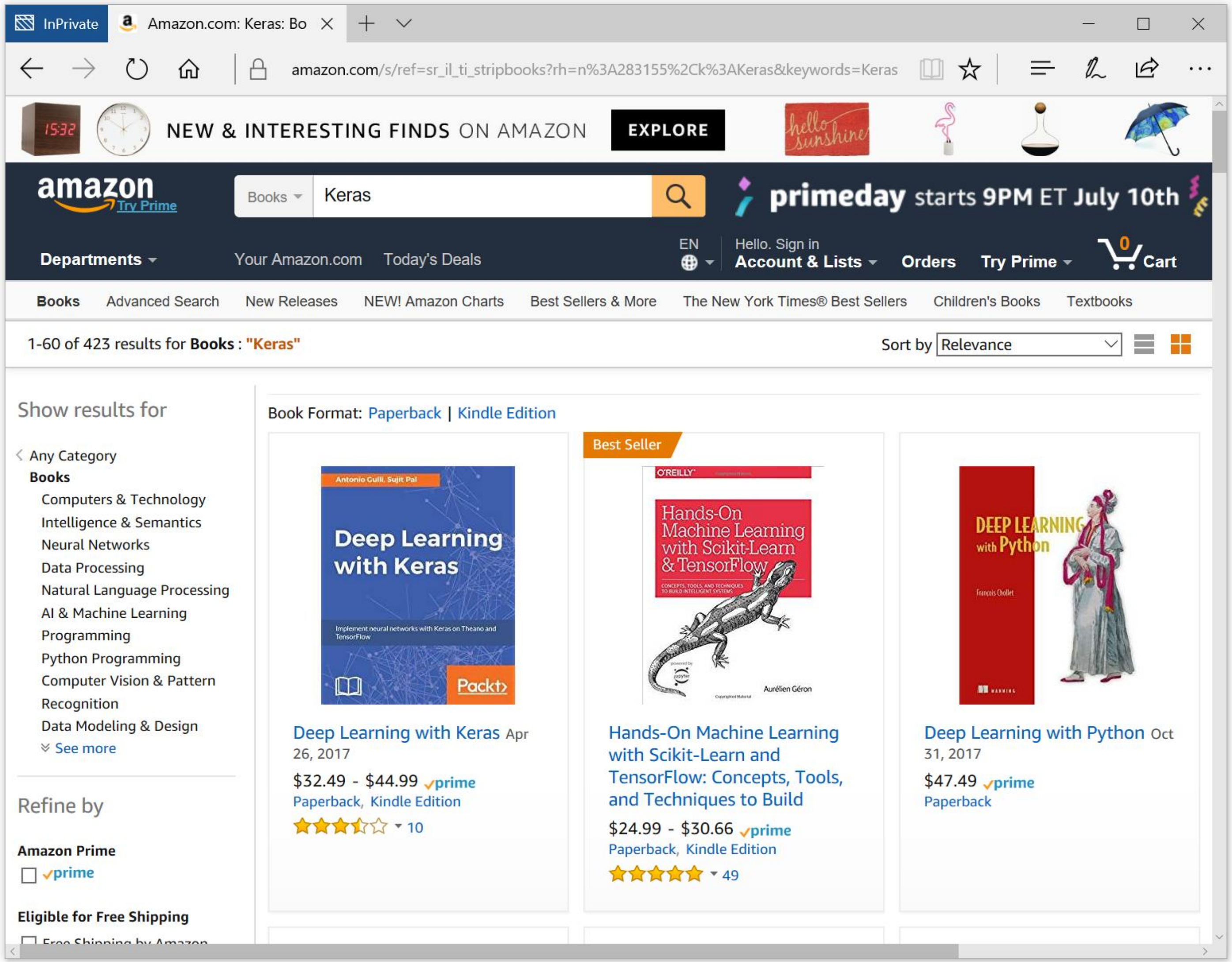
- You could turn the classification problem from the Simple MLP Example into a linearly separable problem by manually generating an interaction feature (input1 * input2); but it’s convenient to have the computer do the work for us (as shown in the Simple MLP Example)
- Deep learning models, neural networks with more than one hidden layer, allow the computer to create a hierarchy of features
- For perceptual problems, such as computer vision and speech recognition, deep learning is providing features that make the model’s performance comparable to a human’s performance (for the specified task)

Activation Functions



Why Consider Keras?

We didn't find results for "**CNTK**" in Books.



PyData

Install Keras

```
git clone https://github.com/fchollet/keras
cd keras
python setup.py install
export KERAS_BACKEND=cntk
cd examples
python mnist_mlp.py
```

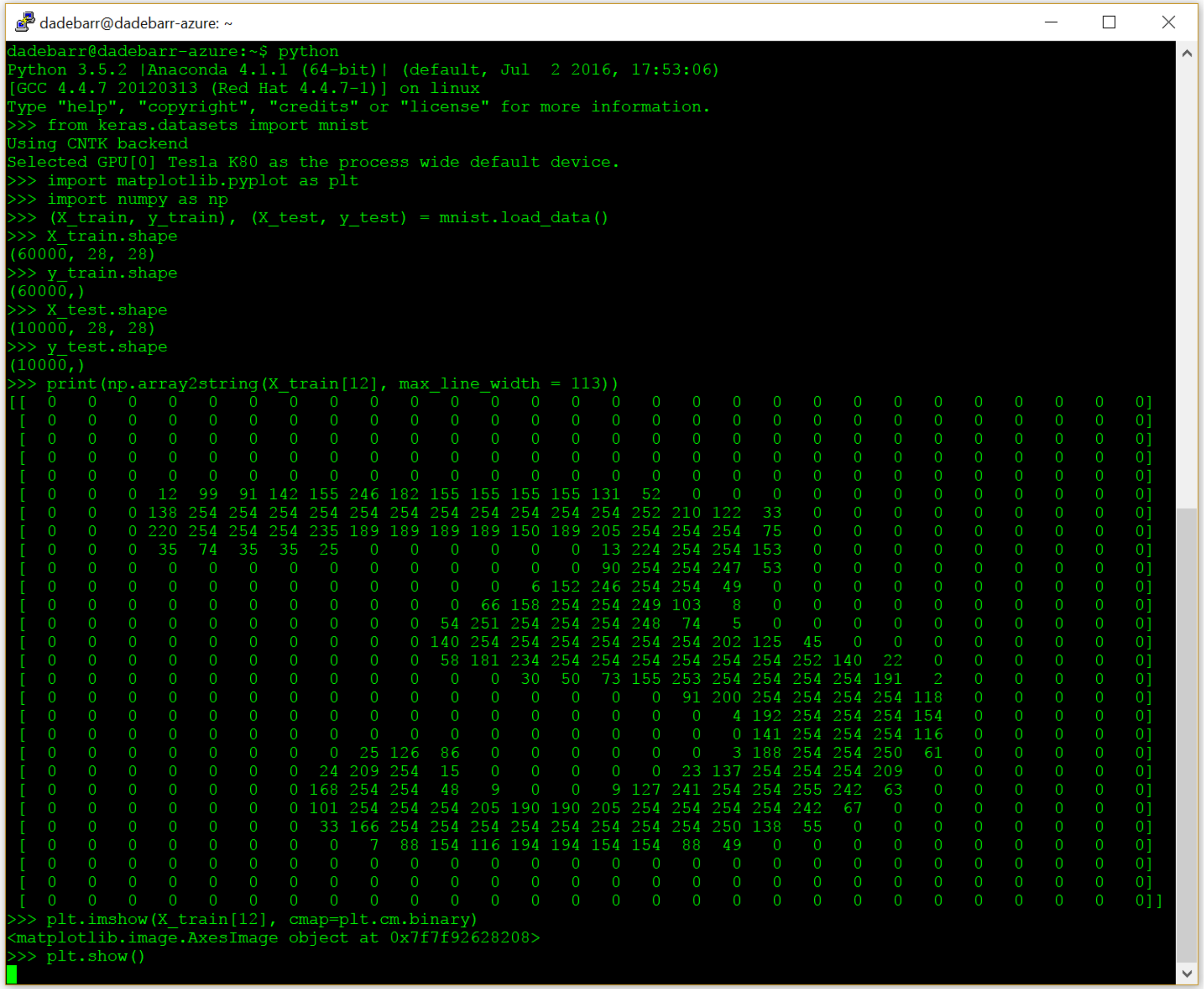
Documentation: https://keras.io/

git clone https://github.com/PacktPublishing/Deep-Learning-with-Keras.git

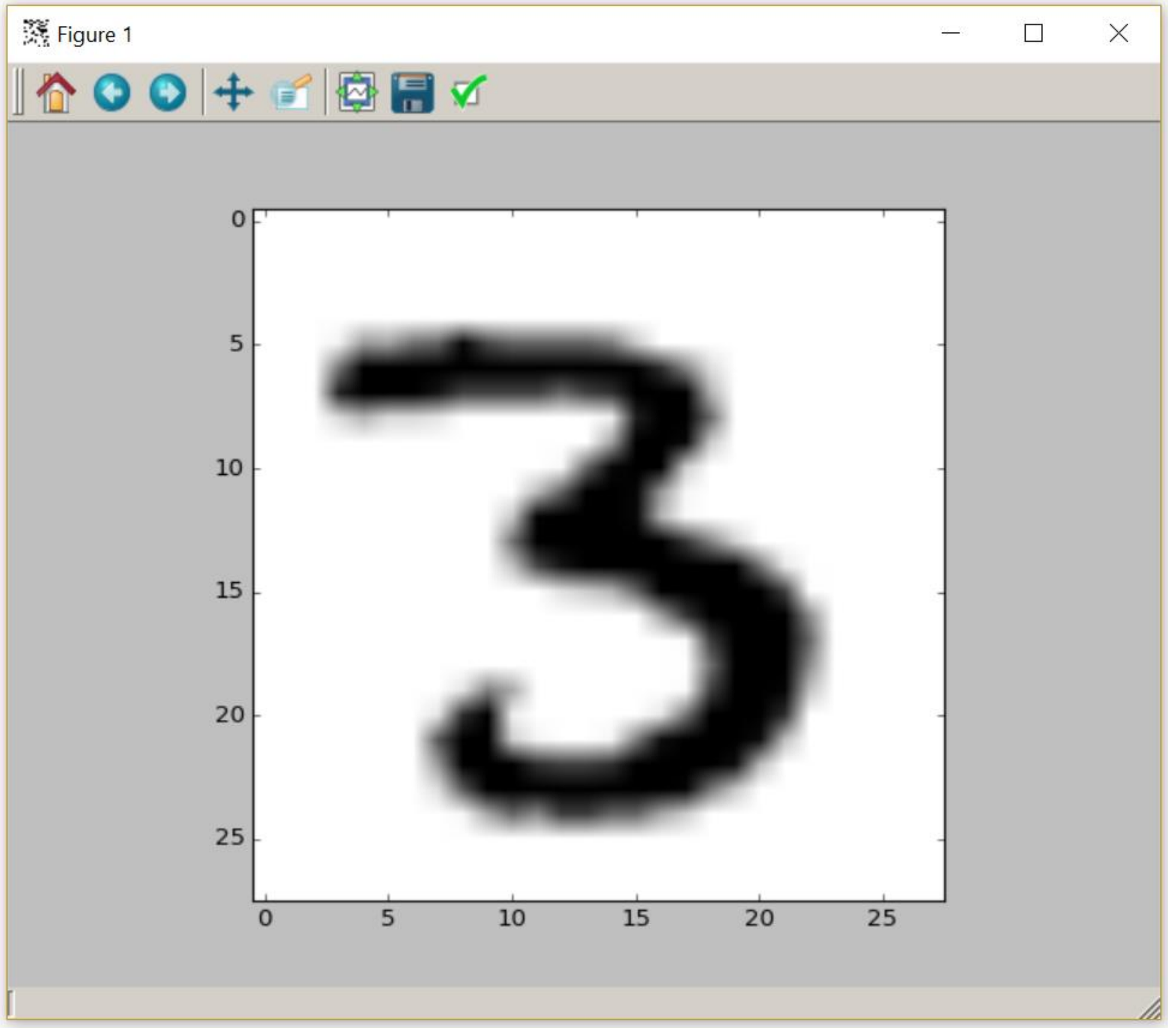
PyData

MNIST Data

Modified National Institutes of Standards and Technology data:
http://yann.lecun.com/exdb/mnist/
http://yann.lecun.com/exdb/lenet/



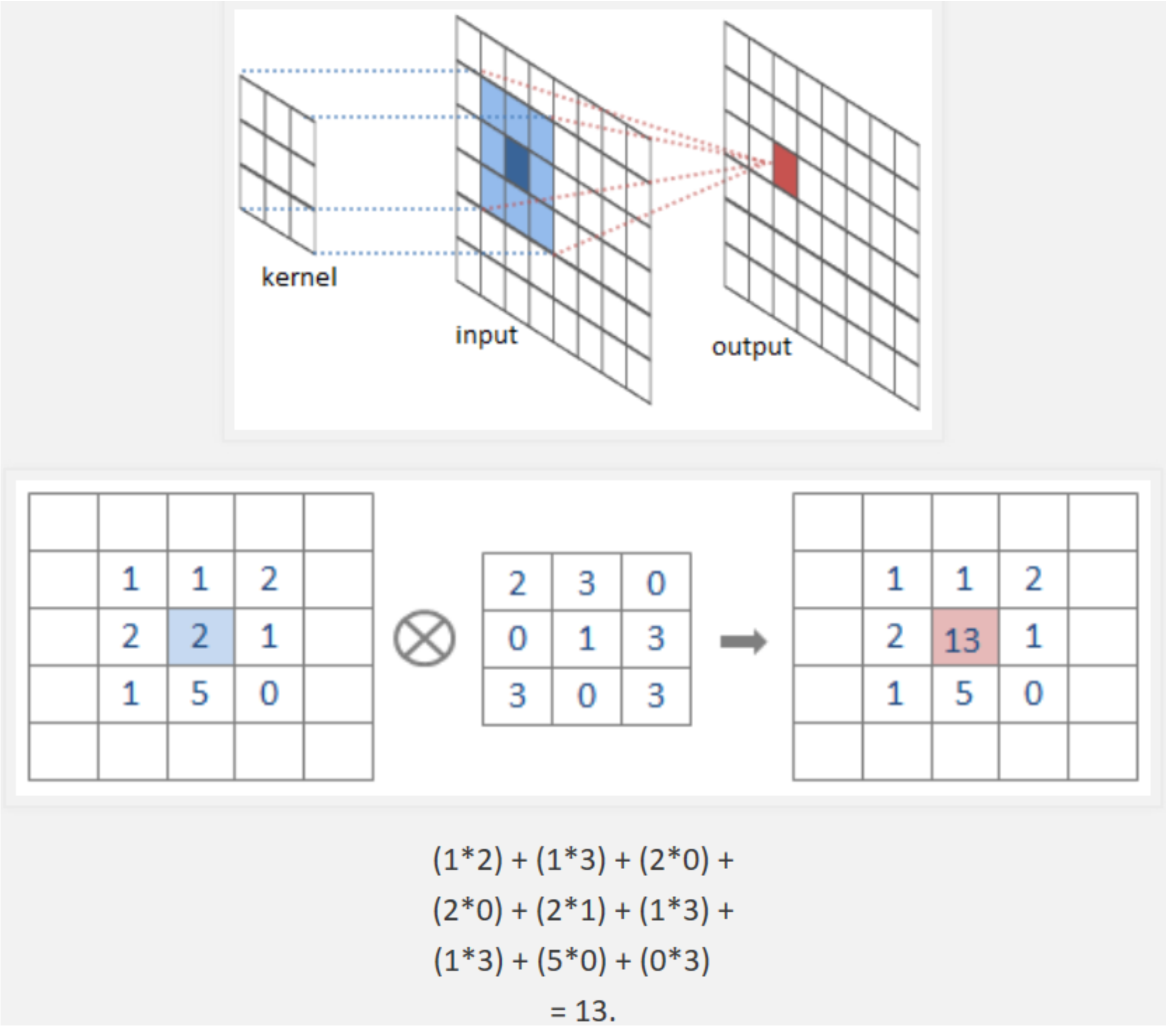
```
dadebarr@dadebarr-azure: ~
dadebarr@dadebarr-azure:~$ python
Python 3.5.2 |Anaconda 4.1.1 (64-bit)| (default, Jul  2 2016, 17:53:06)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from keras.datasets import mnist
Using CNTK backend
Selected GPU[0] Tesla K80 as the process wide default device.
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> (X_train, y_train), (X_test, y_test) = mnist.load_data()
>>> X_train.shape
(60000, 28, 28)
>>> y_train.shape
(60000,)
>>> X_test.shape
(10000, 28, 28)
>>> y_test.shape
(10000,)
>>> print(np.array2string(X_train[12], max_line_width = 113))
[[   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0  12  99  91 142 155 246 182 155 155 155 155 131  52   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0 138 254 254 254 254 254 254 254 254 254 254 254 252 210 122  33   0   0   0   0   0   0   0   0   0]
 [   0   0   0 220 254 254 254 235 189 189 189 189 150 189 205 254 254 254  75   0   0   0   0   0   0   0   0   0]
 [   0   0   0  35  74  35  35  25   0   0   0   0   0   0  13 224 254 254 153   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0  90 254 254 247  53   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   6 152 246 254 254  49   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0  66 158 254 254 249 103   8   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0  54 251 254 254 254 248  74   5   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0 140 254 254 254 254 254 254 202 125  45   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0  58 181 234 254 254 254 254 254 254 252 140  22   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0  30  50  73 155 253 254 254 254 254 191   2   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  91 200 254 254 254 254 118   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   4 192 254 254 254 154   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 141 254 254 254 116   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0  25 126  86   0   0   0   0   0   0   3 188 254 254 250  61   0   0   0   0   0]
 [   0   0   0   0   0   0   0  24 209 254  15   0   0   0   0   0  23 137 254 254 254 209   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0 168 254 254  48   9   0   0   9 127 241 254 254 255 242  63   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0 101 254 254 254 205 190 190 205 254 254 254 254 242  67   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0  33 166 254 254 254 254 254 254 254 254 250 138  55   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   7  88 154 116 194 194 154 154  88  49   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
>>> plt.imshow(X_train[12], cmap=plt.cm.binary)
<matplotlib.image.AxesImage object at 0x7f7f92628208>
>>> plt.show()
```



MNIST

- 04_MNIST_LR.ipynb
- 05_MNIST_MLP.ipynb
- 06_MNIST_MLP_Dropout.ipynb
- 07_MNIST_MLP_RMSProp.ipynb
- 08_MNIST_CNN.ipynb

Convolution Example

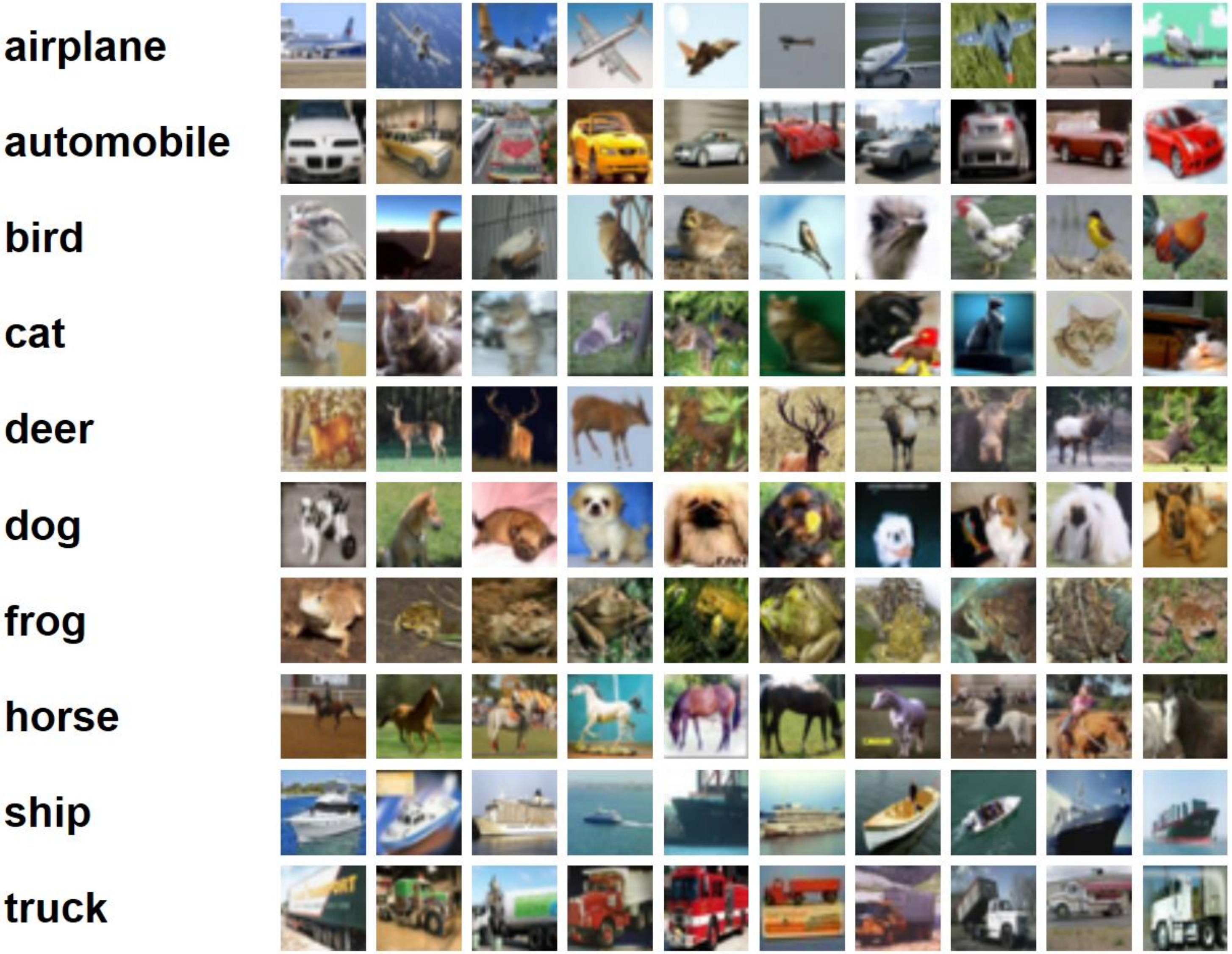


The output response map quantifies the filter's response at locations within the image

http://intellabs.github.io/RiverTrail/tutorial/

CIFAR 10 Data

Canadian Institute For Advanced Research (CIFAR):
http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html



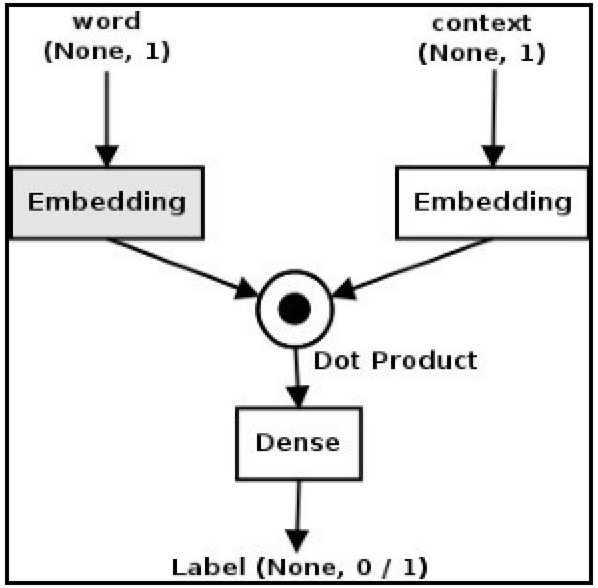
CIFAR10

- 09_CIFAR10_CNN.ipynb

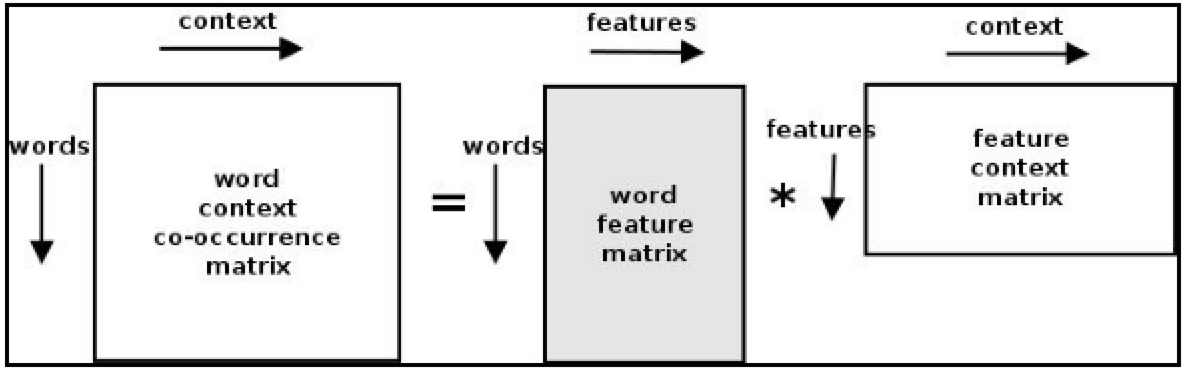
Text Classification

- 10_Reuters_MLP.ipynb
- 11_Newsgroups_GloVe_CNN.ipynb

word2vec embeddings

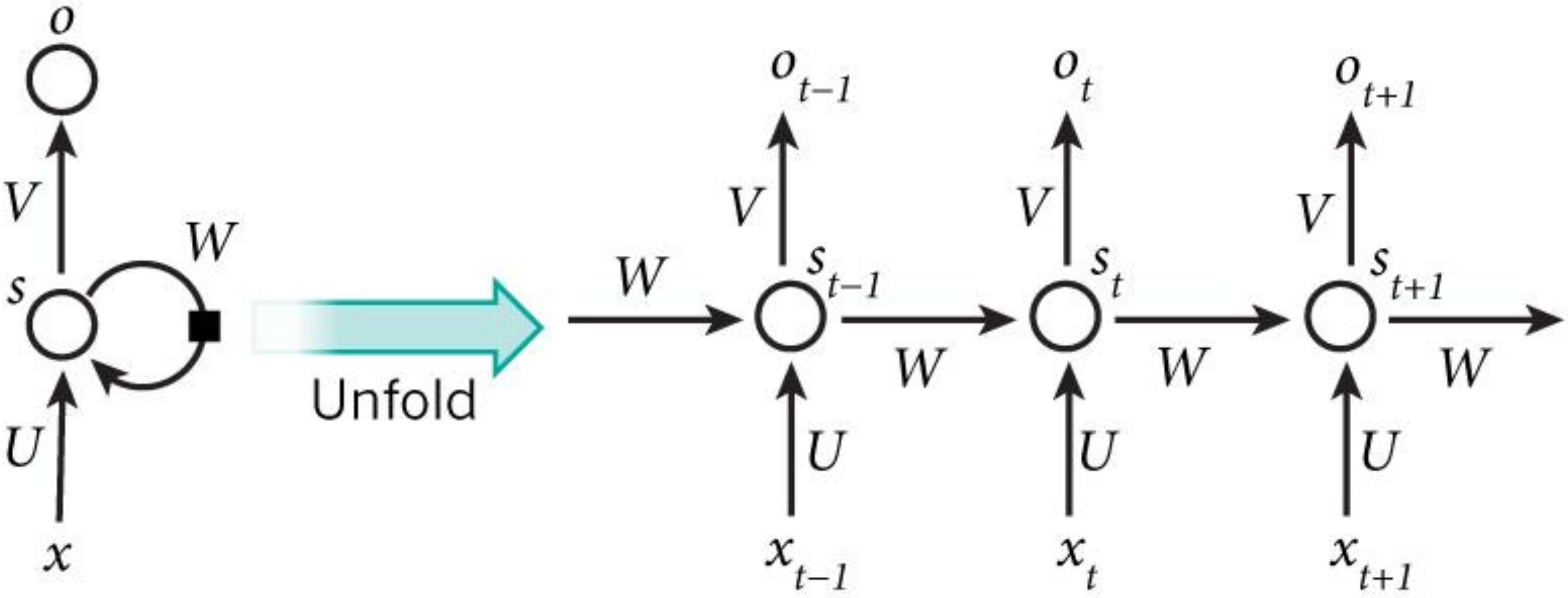


Global Vector (GloVe) embeddings



Example: embedding(king) - embedding(man) + embedding(woman) == embedding(queen)

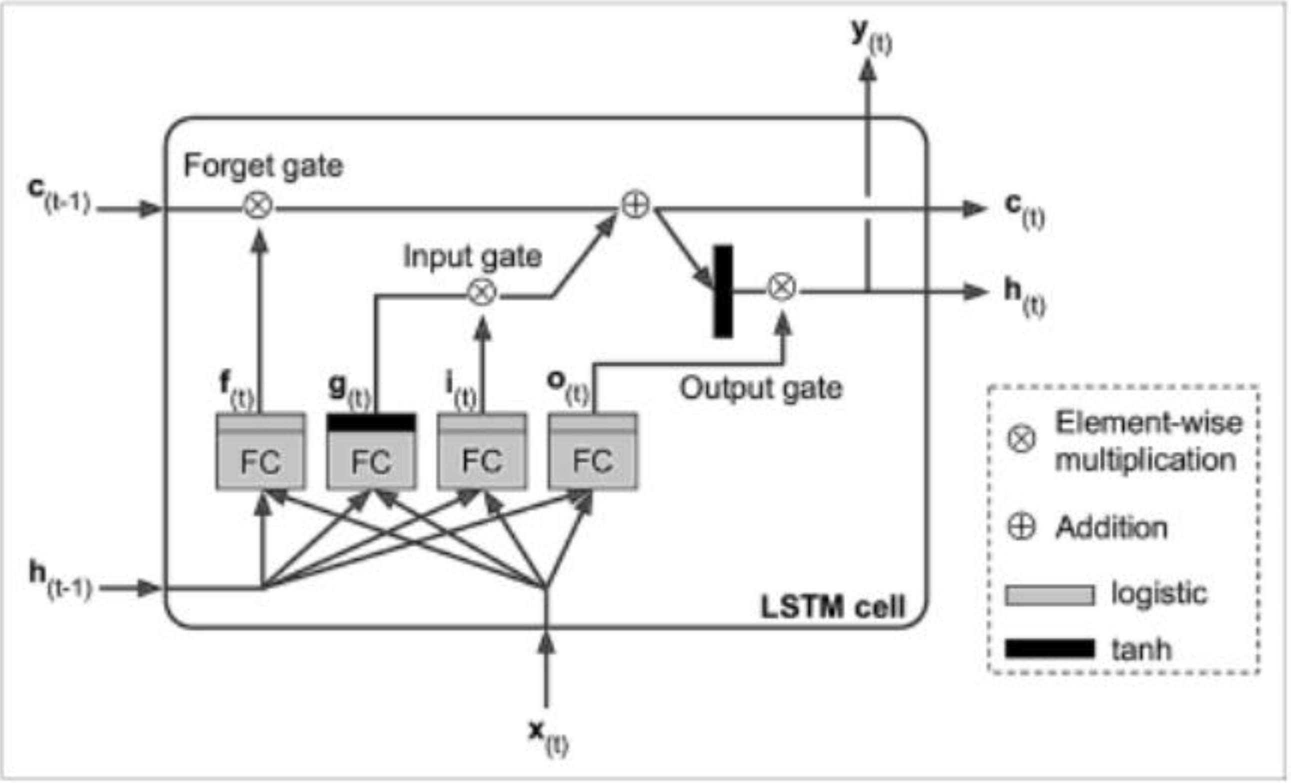
Simple Recurrent Neural Network Example



$s_t = f(Ws_{t-1} + Ux_t)$
$o_t = g(Vs_t)$

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Long Short-Term Memory (LSTM) Cell



$$
\begin{aligned}
\mathbf{i}_{(t)} &= \sigma\left(\mathbf{W}_{xi}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i\right) \\
\mathbf{f}_{(t)} &= \sigma\left(\mathbf{W}_{xf}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f\right) \\
\mathbf{o}_{(t)} &= \sigma\left(\mathbf{W}_{xo}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o\right) \\
\mathbf{g}_{(t)} &= \tanh\left(\mathbf{W}_{xg}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g\right) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh\left(\mathbf{c}_{(t)}\right)
\end{aligned}
$$

Hands-On Machine Learning with Scikit-Learn and TensorFlow

Text Continued

- 12_IMDB_LSTM.ipynb
- 13_IMDB_LSTM_Bidirectional.ipynb
- 14_IMDB_FastText.ipynb

Recap of Stuff We Covered

- Brief Intro
- Setting Up an Azure VM with a GPU; and installing GPU drivers, CNTK, and Keras
- Bunch of Examples, including both Feedforward and Recurrent Neural Networks

1. SGD
2. Backpropagation
3. MLP CNTK
4. MNIST LR
5. MNIST MLP
6. MNIST MLP Dropout
7. MNIST MLP RMSProp
8. MNIST CNN
9. CIFAR10 CNN
10. Reuters MLP
11. Newsgroups GloVe CNN
12. IMDB LSTM
13. IMDB LSTM Bidirectional
14. IMDB FastText

CNTK References

- Python API Documentation: https://cntk.ai/pythondocs/cntk.html
  - cntk.layers
  - cntk.ops
  - cntk.train.trainer
  - cntk.learners
  - cntk.losses
  - cntk.metrics
- Stack OverFlow: http://stackoverflow.com/search?q=cntk (note CNTK tag)

PyData

Other Stuff to Check Out

- keras/examples/babi_memnn.py
  - trains a memory network on the bAbI dataset for reading comprehension
  - bAbI: "baby", with A.I. capitalized (https://research.fb.com/projects/babi/)
- AN4 Alphanumeric Data Classification
  - git clone https://github.com/Microsoft/CNTK.git
  - cd CNTK/Examples/Speech/AN4/Python
  - python HTK_LSTM_Truncated_Distributed.py
- Kaggle competitions
  - Ensembling of diverse models; e.g. an ensemble that includes both a wide, shallow network and a narrow, deep network

References

- Applied Deep Learning
  - https://www.manning.com/books/deep-learning-with-python
  - https://www.packtpub.com/big-data-and-business-intelligence/deep-learning-keras
- Theoretical Deep Learning
  - http://www.deeplearningbook.org/
- Applied Machine Learning
  - http://www.statlearning.com/
  - http://statweb.stanford.edu/~tibs/ElemStatLearn/
- Theoretical Machine Learning
  - https://mitpress.mit.edu/books/machine-learning-0

Appendix Material

Derivative of a Sigmoid Function

From the Simple MLP Example …

$$\frac{\partial}{\partial \hat{f}(x)} \frac{1}{1 + \exp\left(-\hat{f}(x)\right)}$$

$$= -\frac{1}{\left(1 + \exp\left(-\hat{f}(x)\right)\right)^2} \frac{\partial}{\partial \hat{f}(x)} \left(1 + \exp\left(-\hat{f}(x)\right)\right)$$

$$= -\frac{1}{\left(1 + \exp\left(-\hat{f}(x)\right)\right)^2} \left(\frac{\partial}{\partial \hat{f}(x)} 1 + \frac{\partial}{\partial \hat{f}(x)} \exp\left(-\hat{f}(x)\right)\right)$$

$$= -\frac{1}{\left(1 + \exp\left(-\hat{f}(x)\right)\right)^2} \left(0 + \exp\left(-\hat{f}(x)\right) \frac{\partial}{\partial \hat{f}(x)} \left(-\hat{f}(x)\right)\right)$$

$$= -\frac{1}{\left(1 + \exp\left(-\hat{f}(x)\right)\right)^2} \left(\exp\left(-\hat{f}(x)\right)(-1)\right)$$

$$= \frac{1}{\left(1 + \exp\left(-\hat{f}(x)\right)\right)^2} \exp\left(-\hat{f}(x)\right)$$

$$= \frac{1}{1 + \exp\left(-\hat{f}(x)\right)} \frac{\exp\left(-\hat{f}(x)\right)}{1 + \exp\left(-\hat{f}(x)\right)}$$

$$= \frac{1}{1 + \exp\left(-\hat{f}(x)\right)} \left(1 - \frac{1}{1 + \exp\left(-\hat{f}(x)\right)}\right)$$