USC Viterbi
School of Engineering

COMPUTATIONAL DEEP LEARNING

EE 541 – UNIT 1

DR. BRANDON FRANZKE

Spring 2023

OUTLINE

- What is Machine Learning
- Setting up your Python Environment
  - Python
  - NumPy: Array processing
  - scikit-learn: core ML algorithms
  - PyTorch: Neural network framework
- Other tools and workflow
- Cloud resources
- Python fundamentals

WHAT IS MACHINE LEARNING?

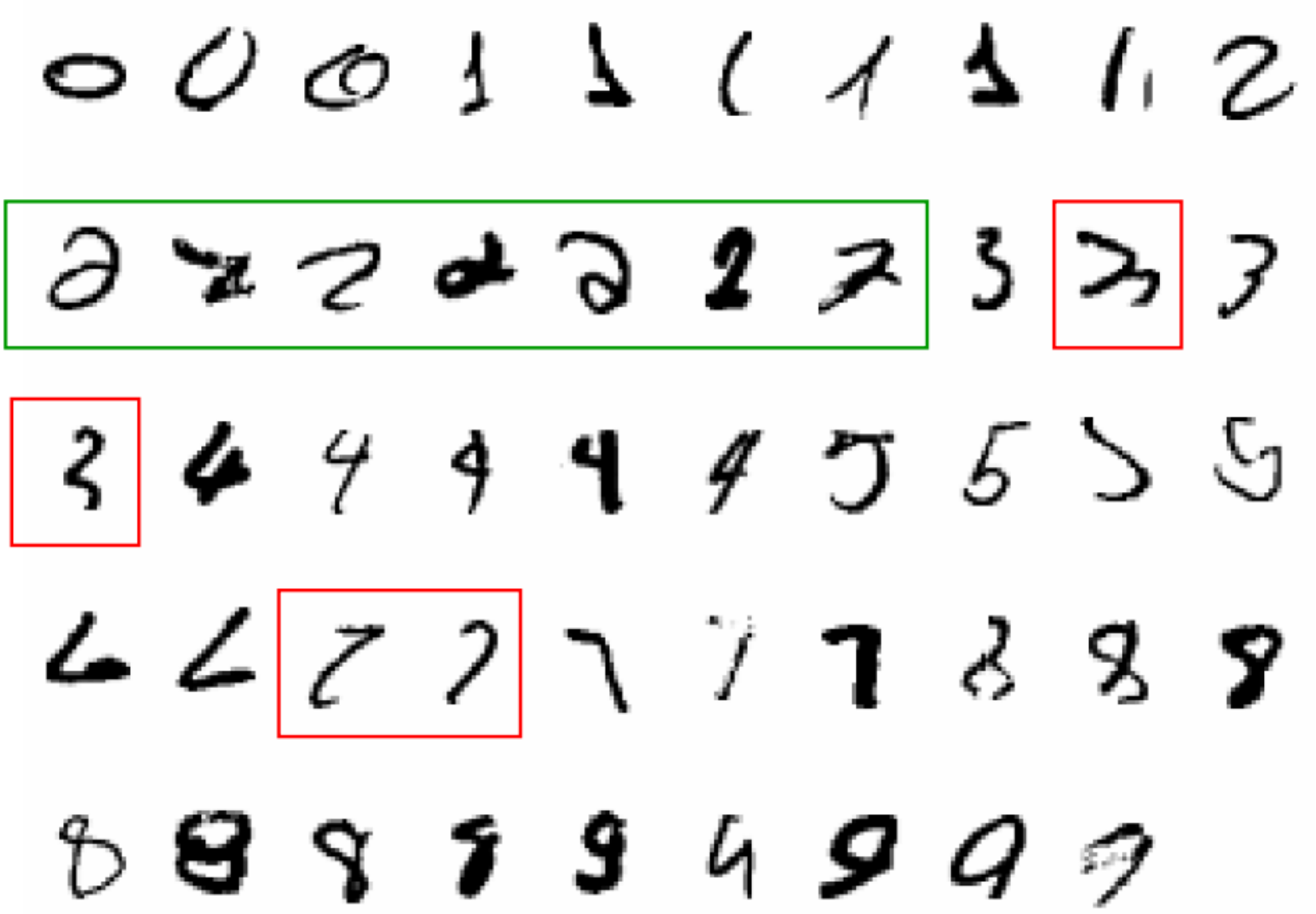
WHAT IS MACHINE LEARNING?

"Learning is any process by which a system improves performance from experience."

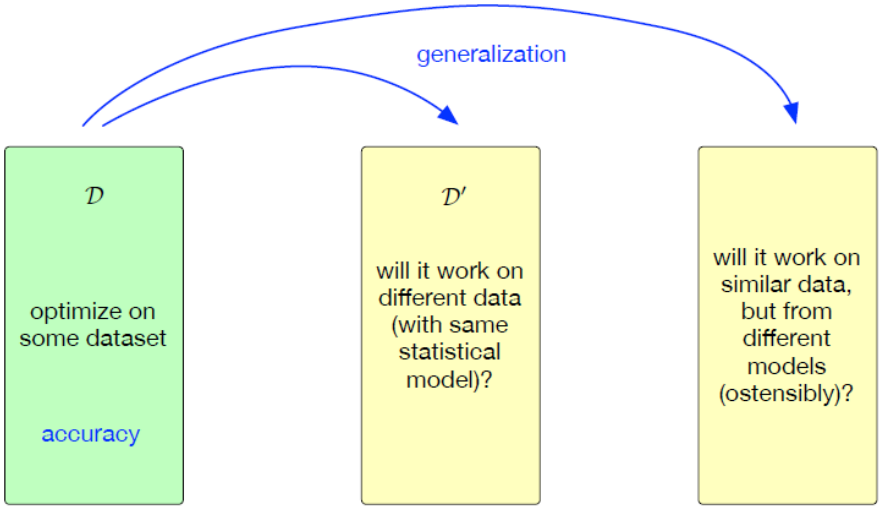
- Herbert Simon

- Machine Learning is the study of algorithms that:
  - improve performance P
  - at given task T
  - with experience E
- Well-defined learning task: $\{P, T, E\}$

EXAMPLE: FIND ALL "2" IN HANDWRITTEN DATA



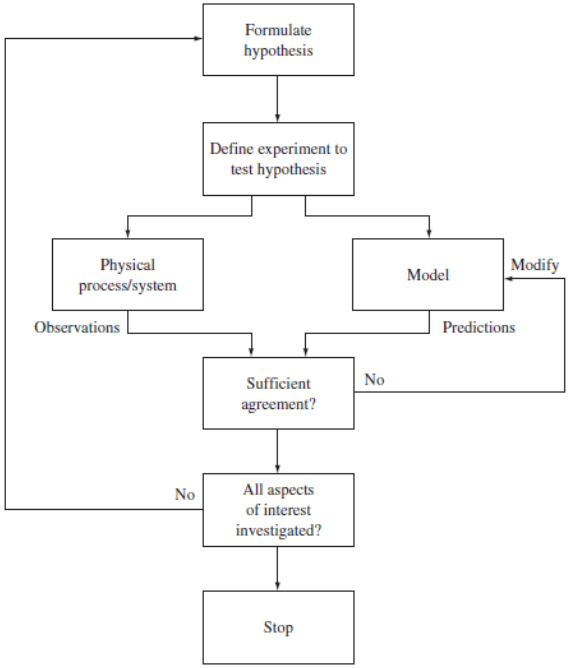
GENERALIZATION IS THE GOAL OF MACHINE LEARNING



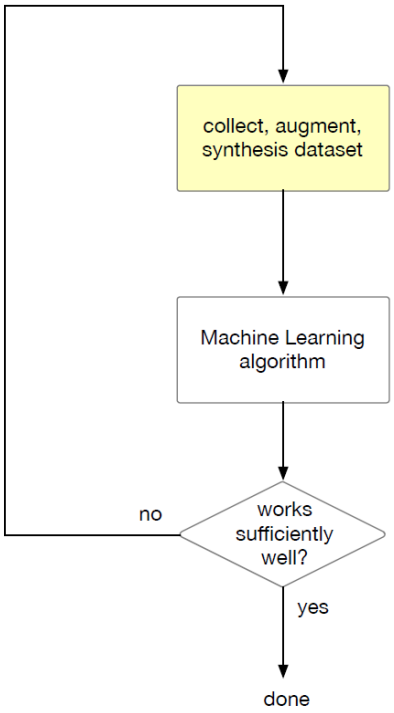
- Do **not** care about the performance on the dataset we have
- **Do care about performance on similar data that has no labels**
- **Accuracy/Generalization trade-off (aka bias-variance trade):**
  - **Generally: optimize accuracy to the extreme reduces capability to generalize**

MACHINE LEARNING ⟹ "*DATA IS* THE STORY"

EXPERIMENT DRIVEN MODELING
(CLASSICAL)

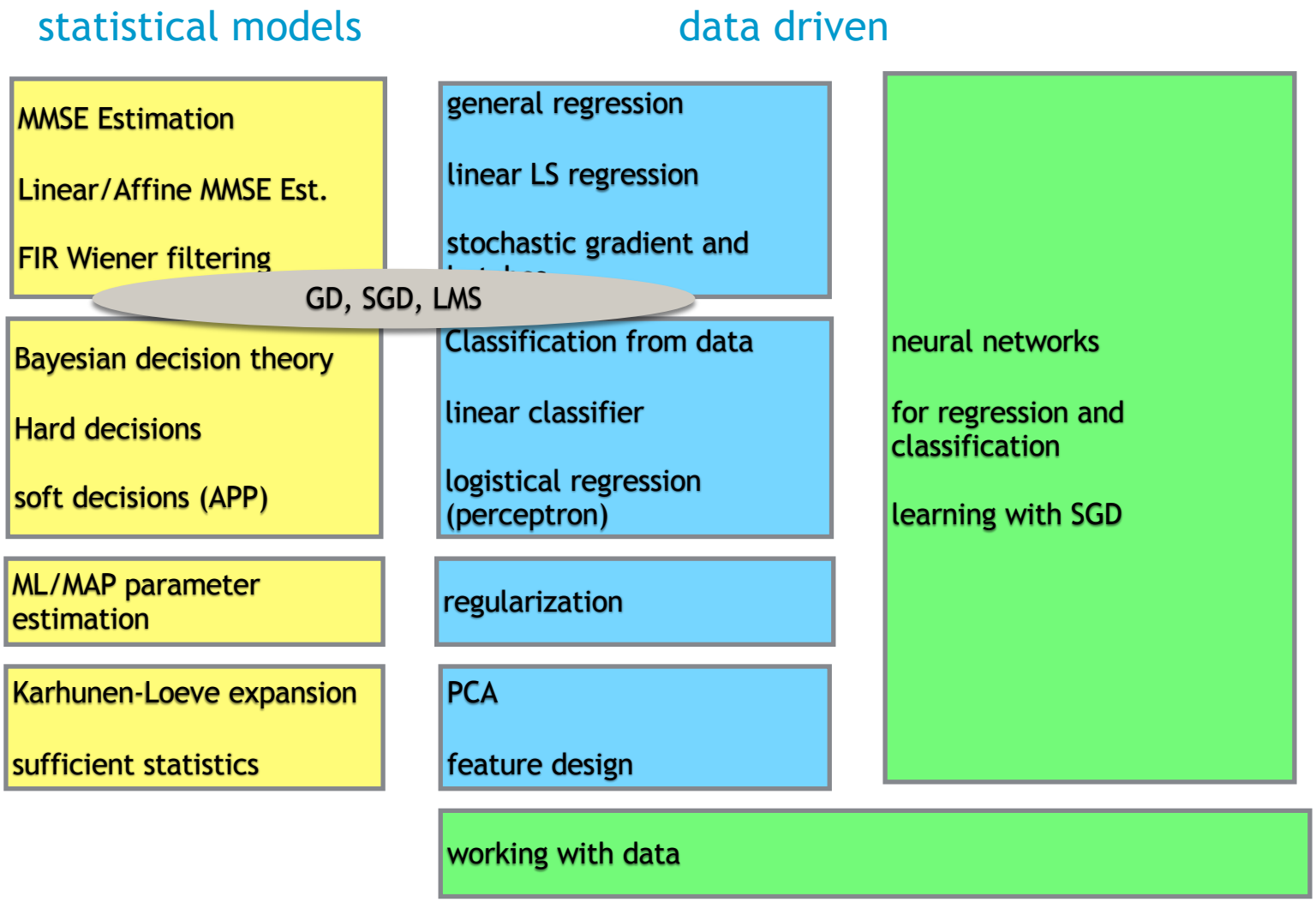


Data driven modeling



A. Leon-Garcia. Probability, Statistics, and Random Processes for EE. 2008.

DETECTION, ESTIMATION, REGRESSION



All models are wrong, but some are useful…

2.3 Parsimony

Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration. On the contrary, following William of Occam he should seek an economical description of natural phenomena. Just as the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity.

2.4 Worrying Selectively

Since all models are wrong the scientist must be alert to what is importantly wrong. It is inappropriate to be concerned about mice when there are tigers abroad.

George Box

"Science and statistics" Journal of the American Statistical Association. 1976.

LEARNING TASKS

- Facial recognition
  - training data: collection of images and labels
  - evaluation: correct label for new images

- Text/document classification
  - training data: labeled training documents (*e.g.*, webpages)
  - evaluation: correct label for new documents

LEARNING TASKS

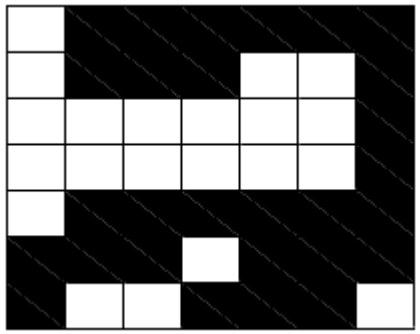
- Computer vision
  - Tesla AI day: https://www.youtube.com/watch?v=j0z4FweCy4M&t=0s
- Natural Language Processing (NLP)
- Recommender systems
- Generating Patterns
  - “Deep fakes”
  - Style transfer
- Anomaly detection
  - Fraud and unusual credit card activity
- Market prediction
- Annotating large datasets

MACHINE LEARNING – THE PROCESS

- Consider a (biased) set of possibilities
  - *hypothesis class*
- Adjust predictions using available examples
  - *estimation*
- Rethink set of possibilities
  - *model selection*

EXAMPLE – THE DATA *DOMAIN*

Input

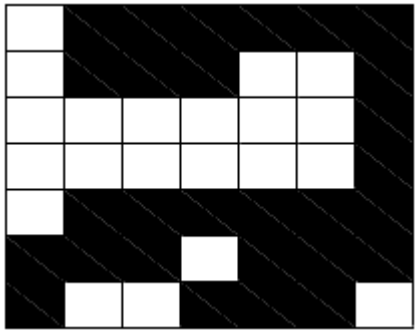
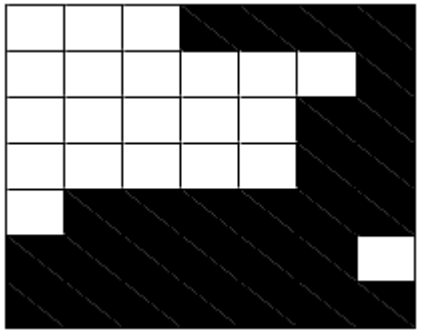


Label

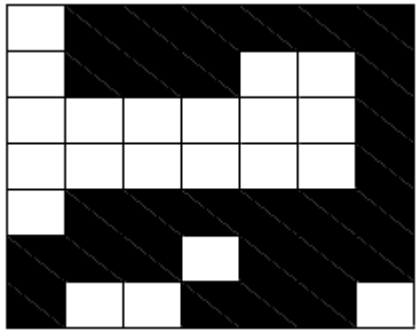
"GOOD"

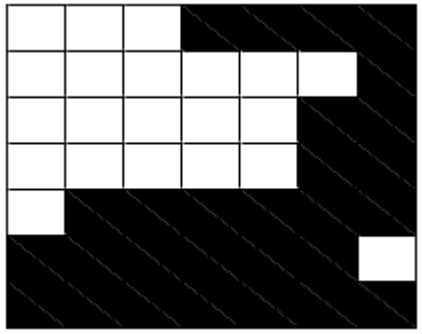
The choice of how to represent input is very important

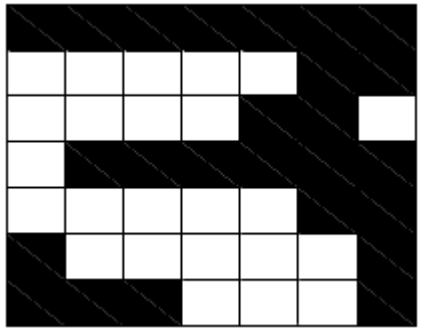
EXAMPLE – THE DATA *DOMAIN*

Input	Label
	"GOOD"
	"GOOD"

EXAMPLE – THE DATA *DOMAIN*







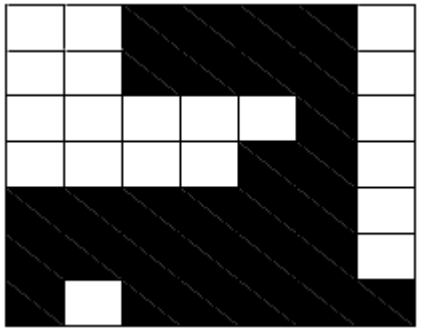
Label

"GOOD"

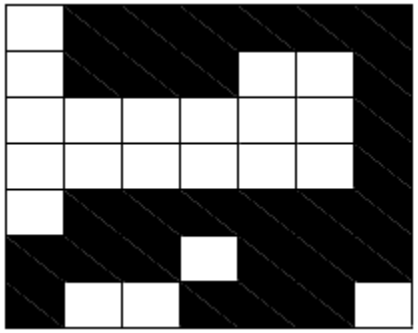
"GOOD"

"FAIL"

?



MANY WAYS TO REPRESENT THE SAME DATA



"GOOD"

```
01111110111001000000010000
00101111111110111001110
```

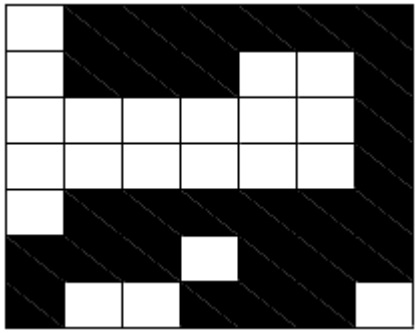
bit string

The choice of how to represent input is very important

DATA REPRESENTATION AND HYPOTHESIS CLASS

Representation: data are binary vectors, length $d = 49$.

$$\boldsymbol{x} = \begin{matrix} 0111111011100100000010000 \\ 0010111111111011111001110 \end{matrix}$$



$$\boldsymbol{y} \in \{-1, +1\}$$

Hypothesize mapping data to label using *linear classifier*.

$$\hat{y} = \text{sign}(\boldsymbol{\theta} \cdot \boldsymbol{x}) = \text{sign}(\theta_1 x_1 + \cdots + \theta_d x_d)$$

θ: Parameters to *learn*

TWO LEARNING METHODS

- **Explicit (closed-form)** calculate θ using many samples.

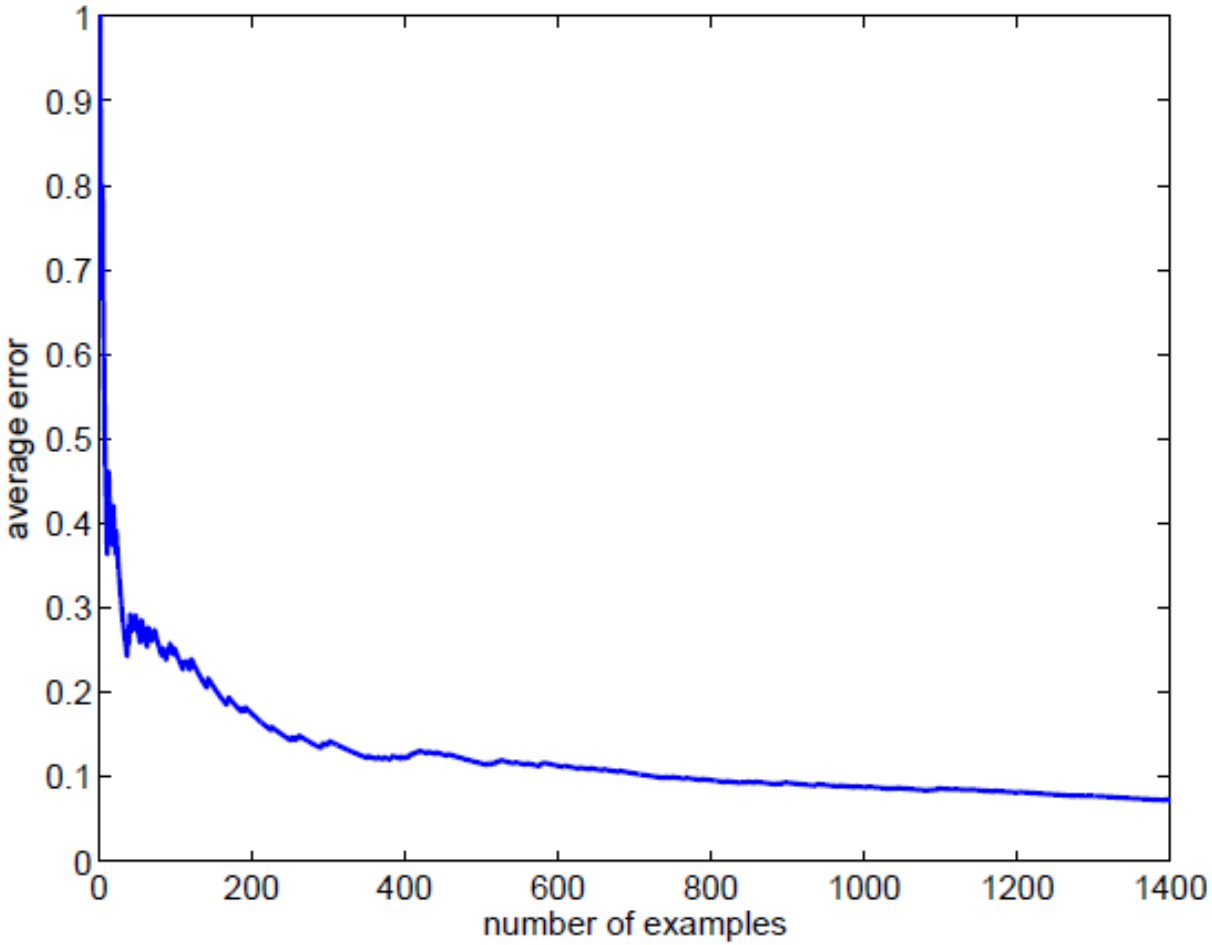
  Choose θ to minimize an *error-function* (cost function).

- **Implicit (sequential)** guess θ_1. Evaluate model.

  Update/adjust parameters if wrong:

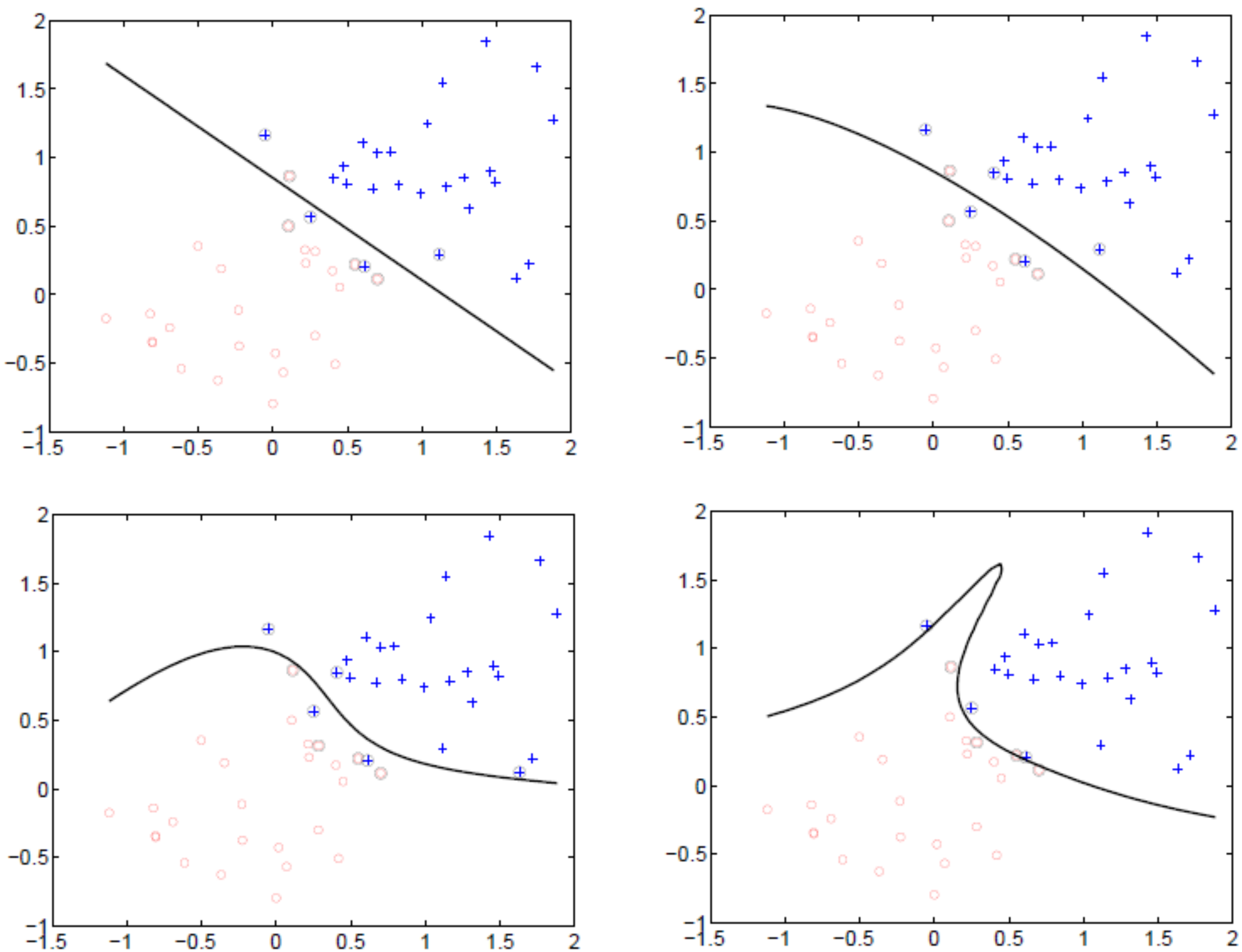
$$\theta_{i+1} = \theta_i + y\, x_i$$

MODEL PERFORMANCE



Average error starts high. Model *learns* as it sees more examples

SUCCESS = ACCURATELY *PARTITION* THE DOMAIN



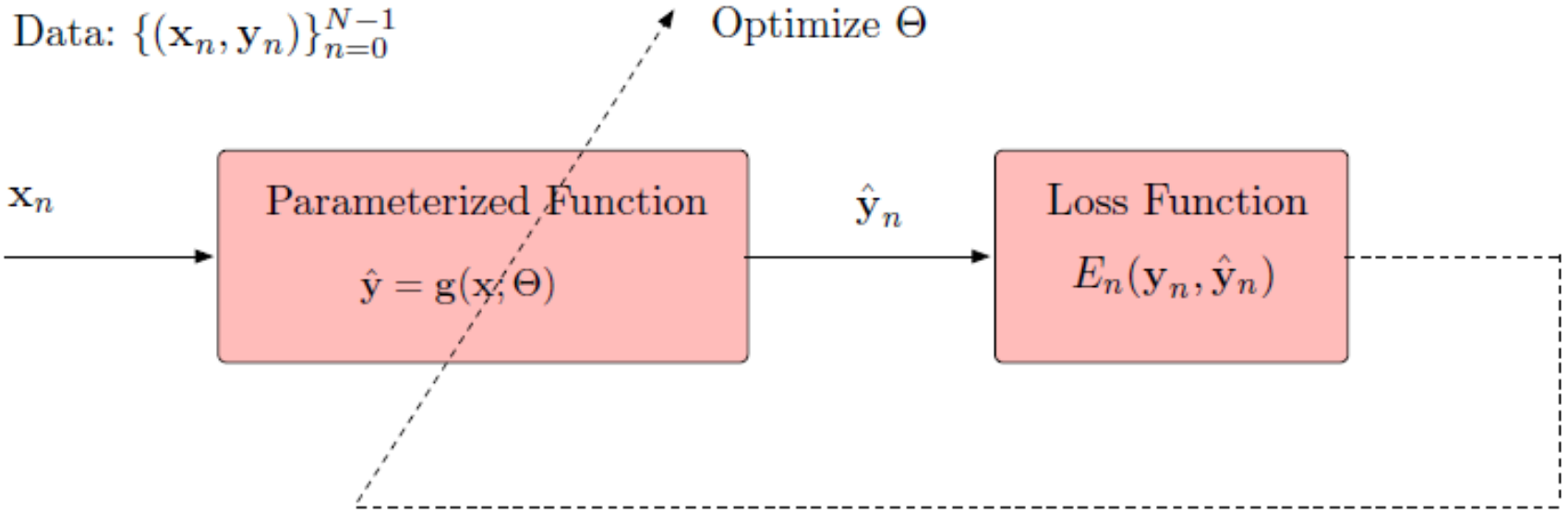
Lower *order* models are simpler but may inherently unable to perfectly partition

Higher *order* models can represent more complicated decision boundaries

THREE TYPES OF ML

1. SUPERVISED LEARNING

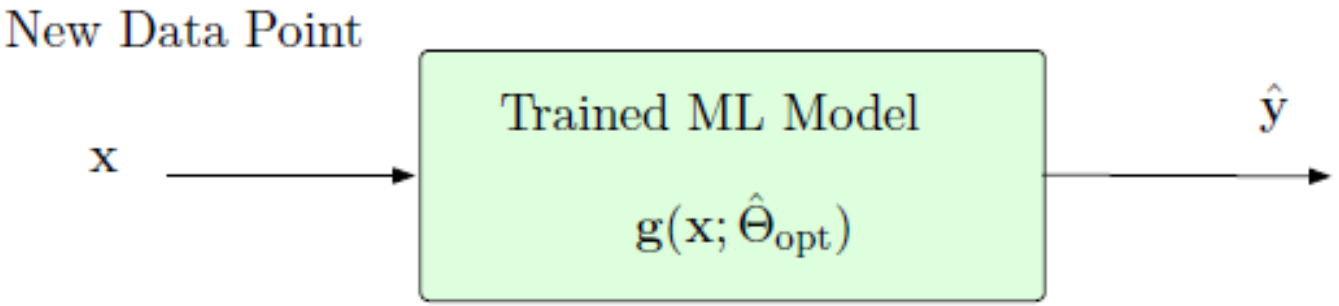
Training requires input and (desired) output pairs



Training = show data, compare prediction to actual, update θ

1. SUPERVISED LEARNING

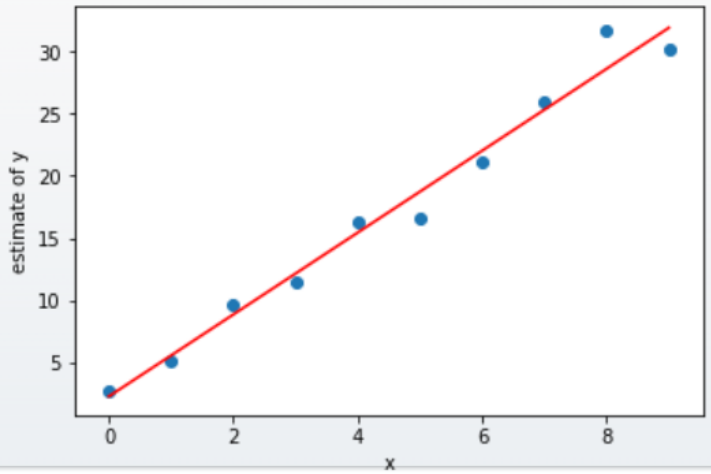
Inference mode = predict new data



- Examples:
  - Automatic speech recognition
  - Image classification
  - Signal filtering & processing

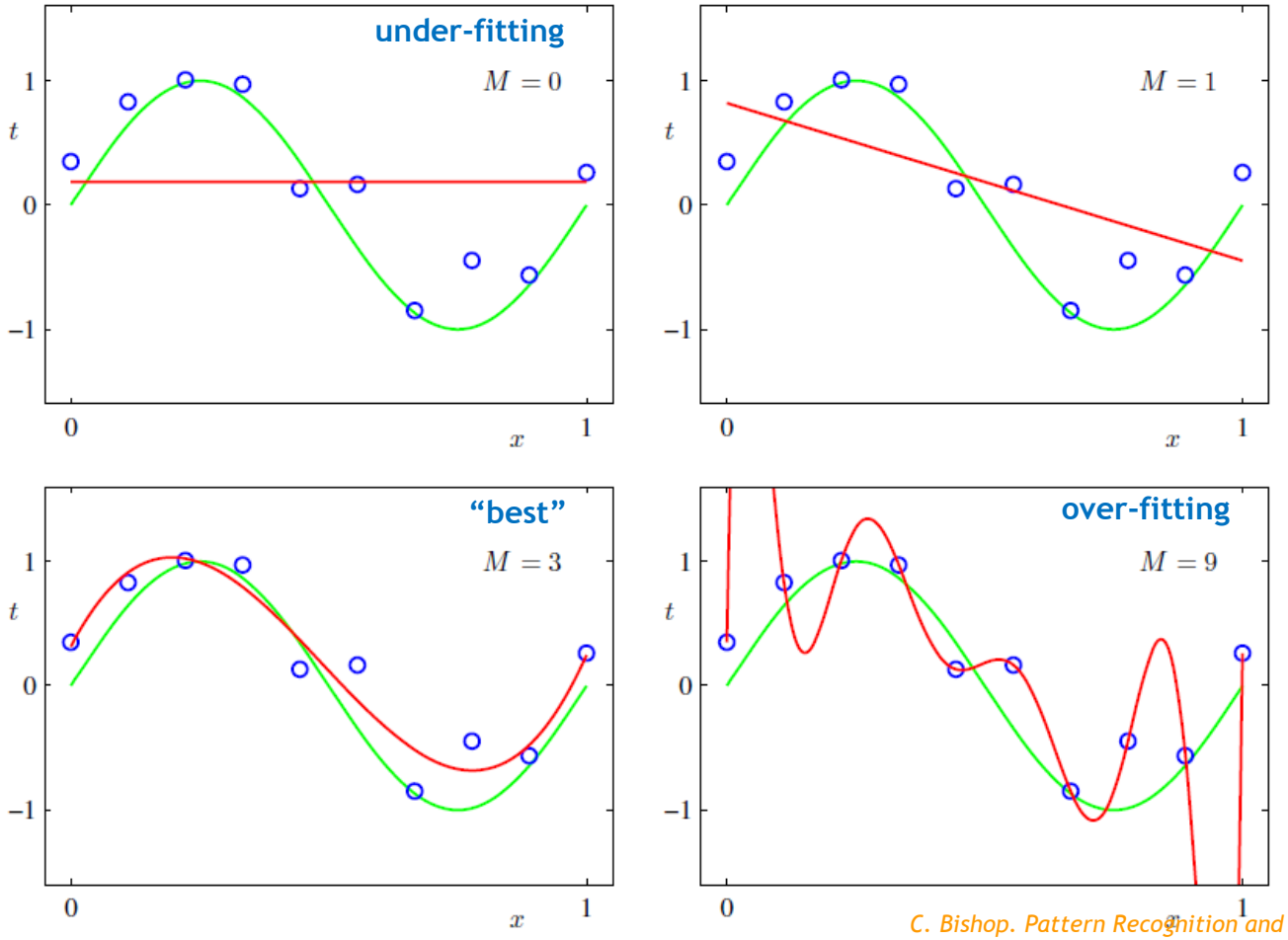
REGRESSION (CURVE-FITTING) FROM DATA

```
from scipy import stats
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(10)
y = 3*x+4
y =  y + np.random.normal(0,2,10)
slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
y_hat = intercept + slope * x

fig = plt.figure()
plt.plot(x,y_hat, color='r')
plt.scatter(x,y)
plt.xlabel("x")
plt.ylabel("estimate of y")
#axes = plt.gca()
#axes.set_xlim([-1, 4])
```



- Assume: data generated with known model + noise
- Guess: at model in practice (*e.g.*, linear)
- Choose: parameters to minimize error

COMPLEX HYPOTHESIS CLASS LEADS TO OVER-FITTING



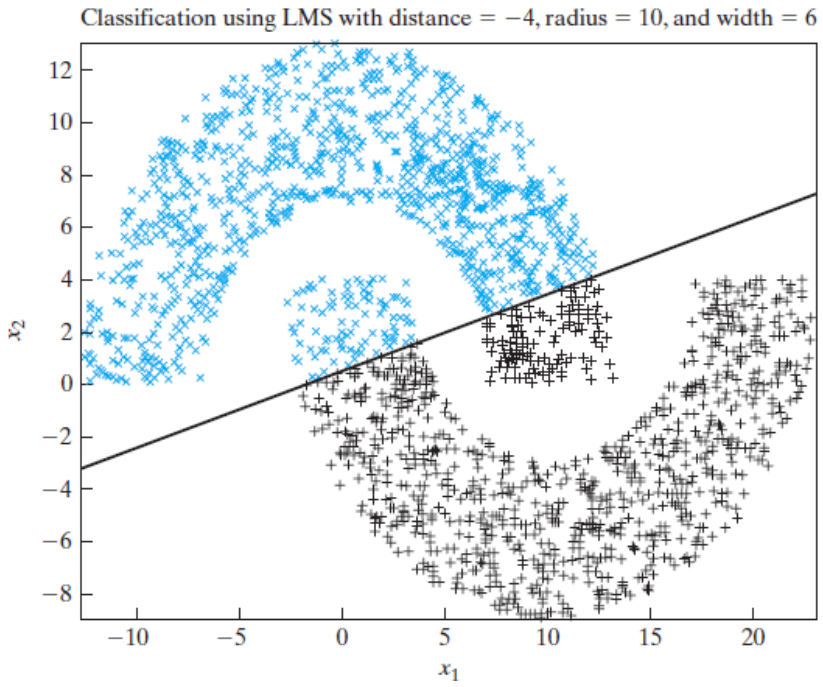
Bias-Variance trade-off (EE 503)

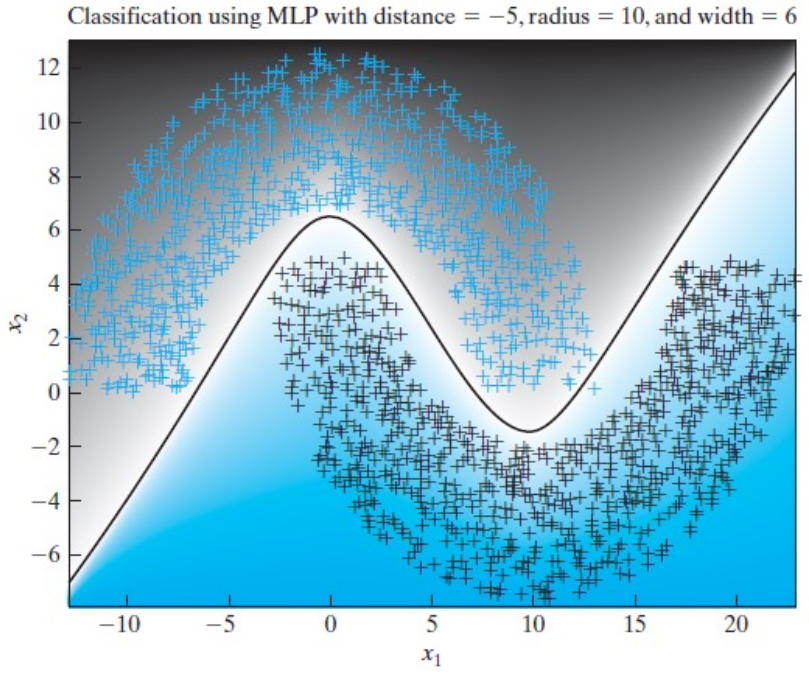
$\text{MSE} = \text{V} + \text{Bias}^2$

C. Bishop. Pattern Recognition and Machine Learning. 2006.

Choosing the right model (hypothesis) is challenging

DECISION BOUNDARIES

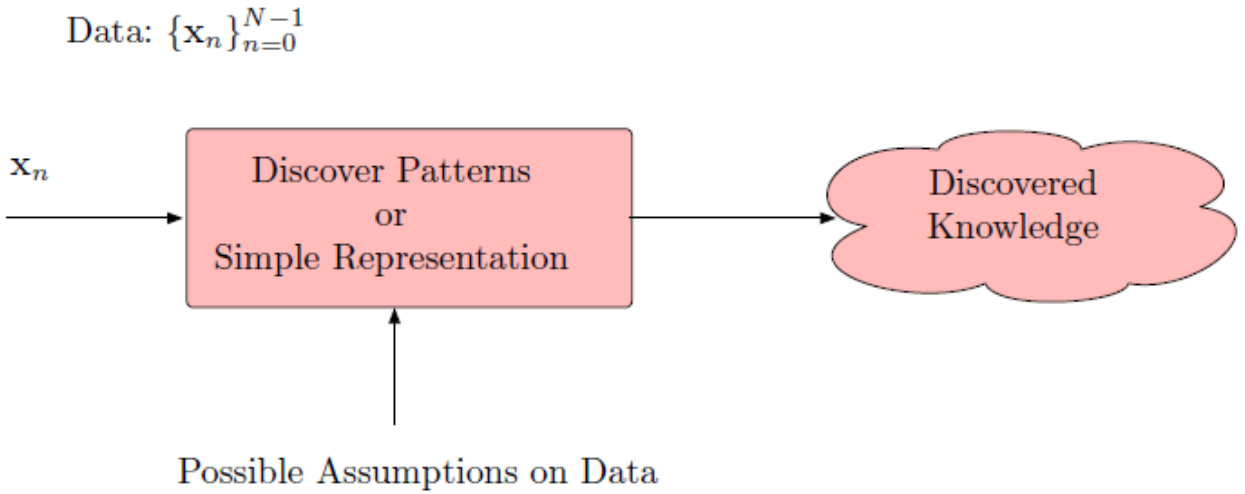




S. Haykin. Neural networks. 1999.

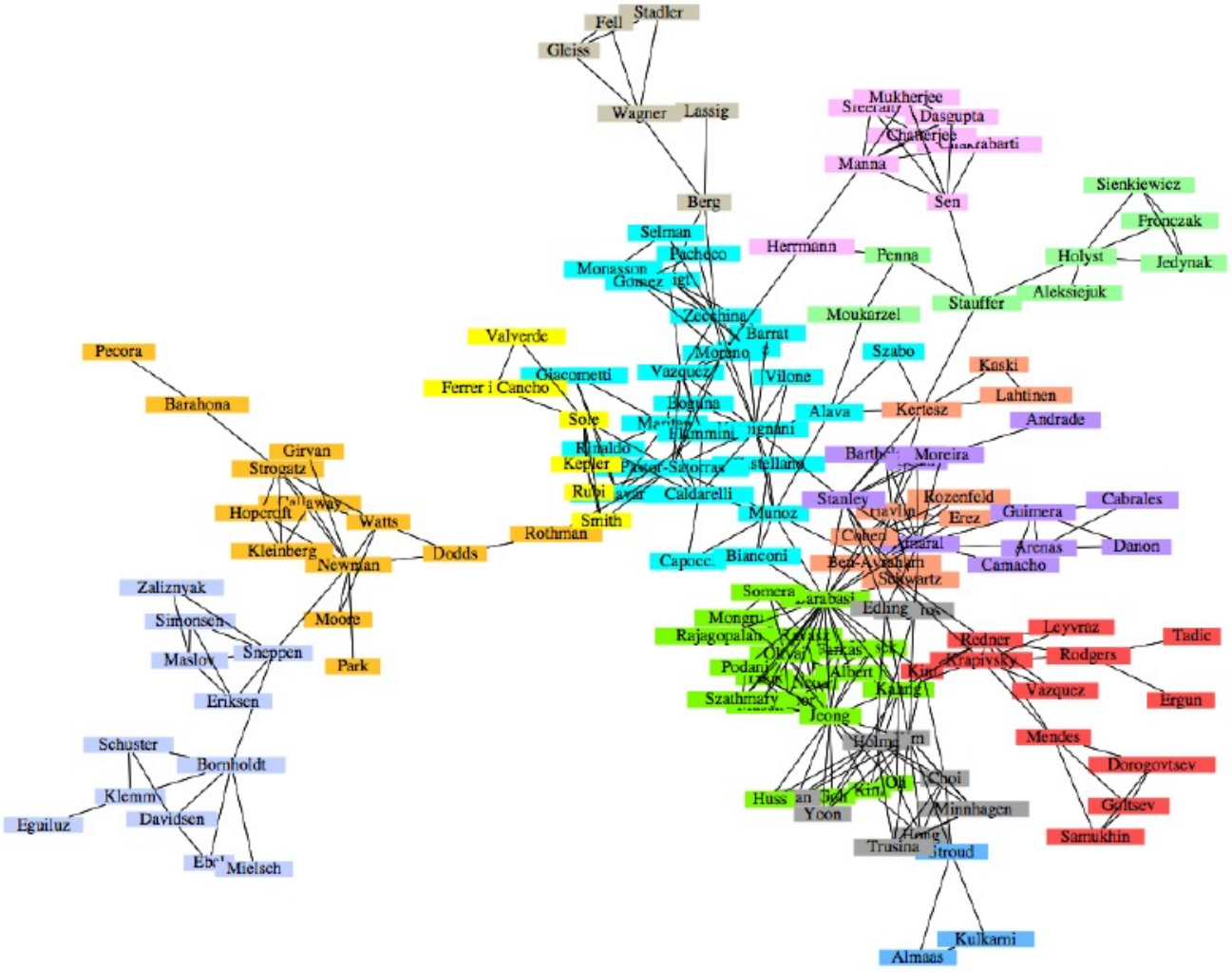
2. UNSUPERVISED LEARNING

Training requires input but not desired output



- Examples:
  - Community detection in social network
  - Political donor analysis / targeted advertisement
  - Sorting photo by subject (unknown subject-set)

2. UNSUPERVISED LEARNING



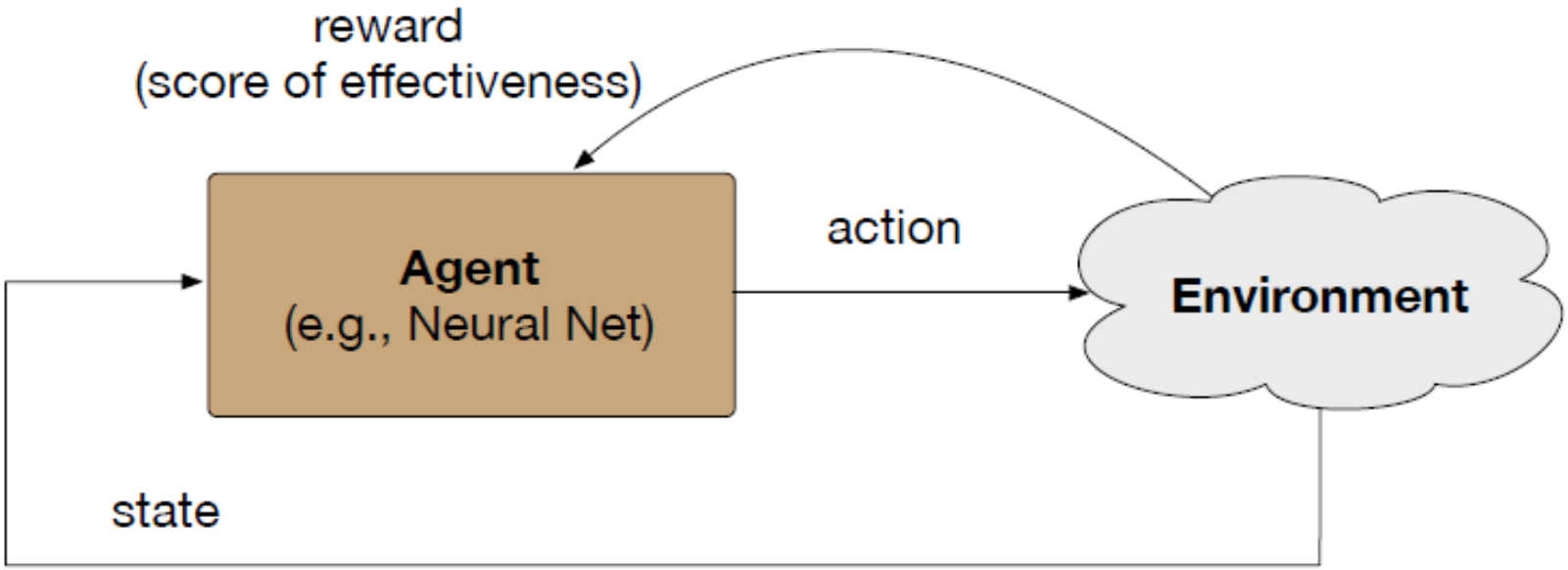
A co-authorship network of physicists and applied mathematicians working on networks

Tightly-knit subgroups become evident within the network structure

Unsupervised ML to discover but probably not identify topics

D. Easley, J. Kleinberg. Networks, Crows, and Markets: Reasoning about a Highly Connected World. 2010.

3. REINFORCEMENT LEARNING



- Examples:
  - Autonomous navigation systems
  - Game playing

NN MODELING

NEURAL NETWORKS DEFINE A RICH *HYPOTHESIS SET*

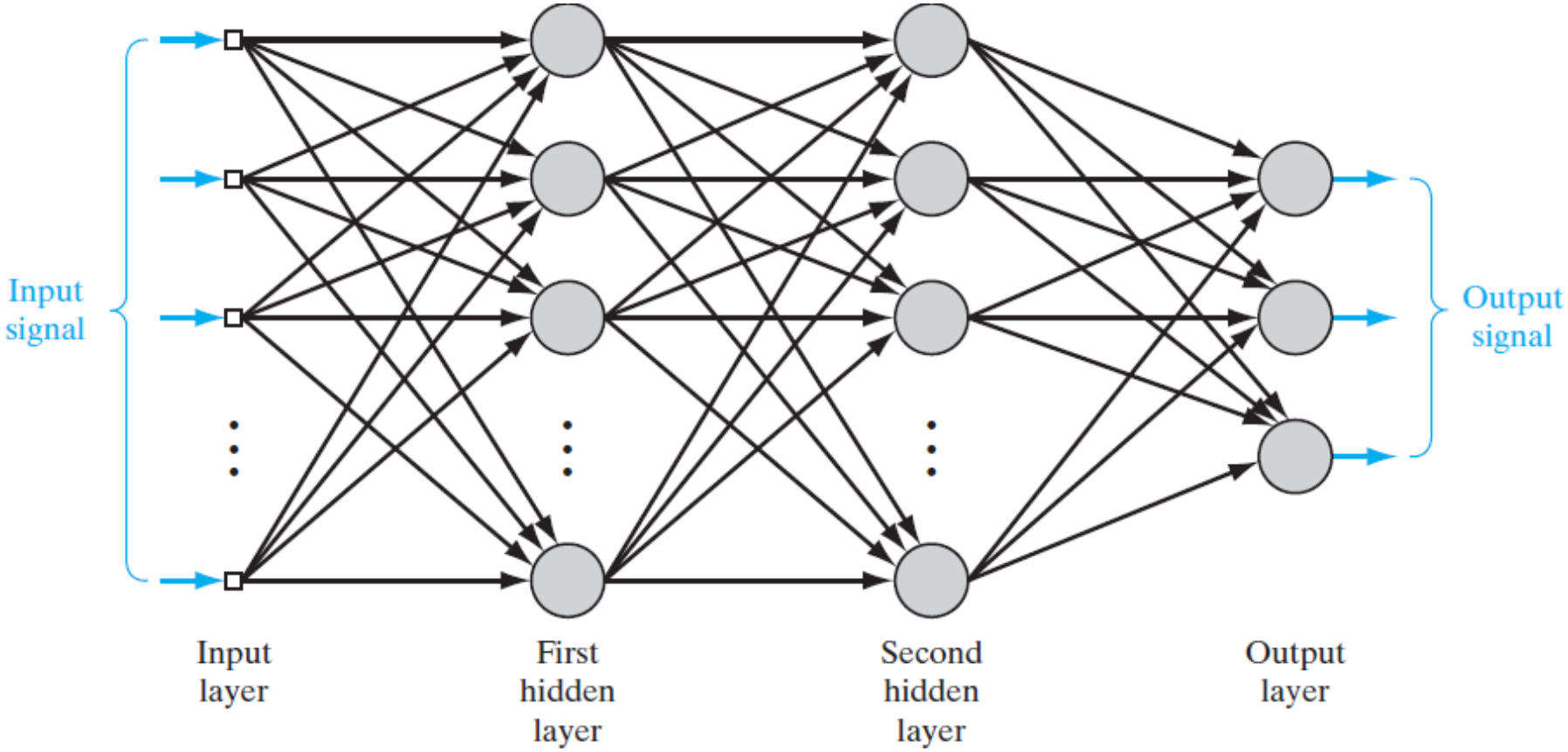


FIGURE 4.1 Architectural graph of a multilayer perceptron with two hidden layers.

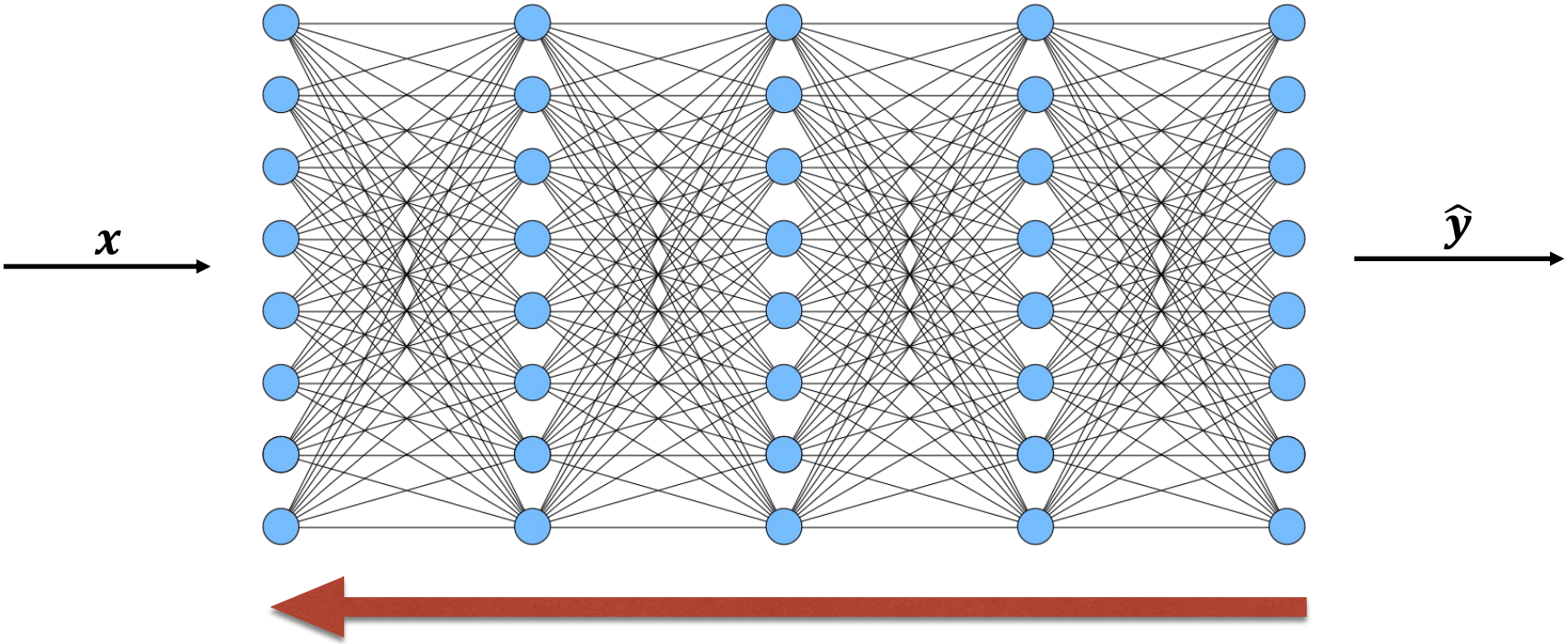
S. Haykin. Neural networks. 1999.

MULTILAYER PERCEPTRON NETWORKS (MLPS)

Forward propagation (inference and training)

$$\boldsymbol{a}^{(l)} = \underline{h}\left(\boldsymbol{W}_l \boldsymbol{a}^{l-1} + \boldsymbol{b}_l\right) \qquad \Theta = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{L}$$

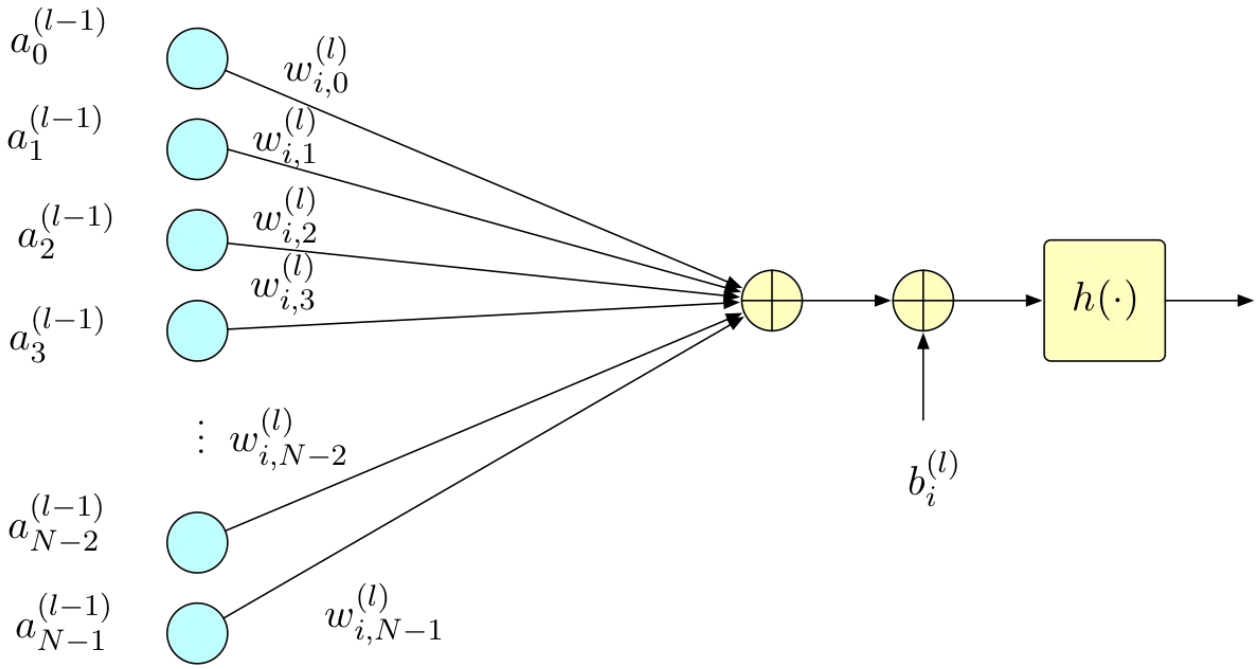
(trainable parameters)



Backward propagation (training)

Learn the trainable parameters using SGD and the chain-rule

MLP FORWARD PROPAGATION DETAILS



$$a_i^{(l)} = h\left(\left[\mathbf{w}_i^{(l)}\right]^{\mathsf{t}} \mathbf{a}^{(l-1)} + b_i^{(l)}\right)$$

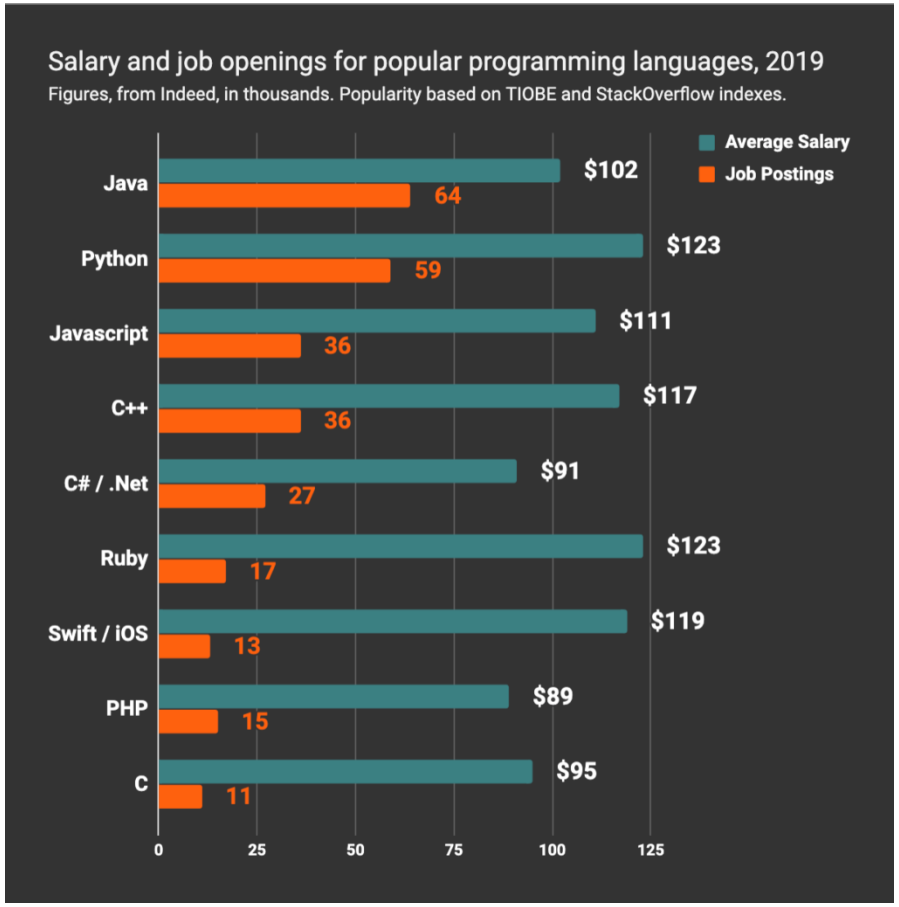
processing at the i^{th} neuron (node) at layer l

YOUR PYTHON ENVIRONMENT

WHY PYTHON? WHY NOT MATLAB?

- Python does almost everything MATLAB does – but it is open source and free
  - Interpreted language (script) like MATLAB
  - Vectorized operations
  - Plotting
- A complete programming language
  - Classes/Objects, Inheritance, Modules
  - Data handling, web-accessing
  - Extensive libraries for numerical computation and ML/DL
    - numpy, scipy, scikit-learn, NLTK, Keras, Tensorflow, Pytorch, etc.

WHY PYTHON?



Source: https://www.codeplatoon.org/the-best-paying-and-most-in-demand-programming-languages-in-2019/

10 Best Programming Language to Learn in 2020

With time old programming languages become obsolete while new programming languages are launched, but they never gain traction. A common question amongst beginners (and coders alike) is the programming language they should invest learning in, that is in demand, stable outlook, and plenty of jobs.

Here, is a list of top 10 languages that you should learn -

1) Python

Created: Python language developed by Guido van Rossum. It was first released in 1991.

Pros:

- Supports multiple systems and platforms
- Object-Oriented Programming (OOPs) driven.
- Helps to improve Programmer's Productivity
- Allows you to scale even the most complex applications with ease
- Extensive Support Libraries

Cons:

- Note ideal for Mobile Computing
- Python's database access layer is bit underdeveloped and primitive.

Source: https://www.guru99.com/best-programming-language.html

SETTING UP PYTHON ENVIRONMENT

EE541 will use Python 3.8

Anaconda

- Install most everything you ever need (and usually more!)
- Bundles some IDEs and Jupyter tools:
  - Spyder
  - Jupyter notebook
  - Qt Console

pyenv

- Install only minimal dependencies
- Good professional option when working with others
  - Manages library versions w/o ambiguity

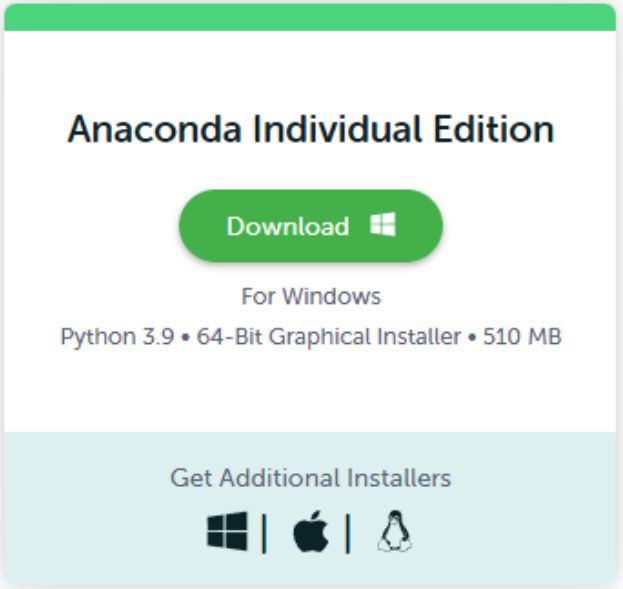
you may have a system Python, best practice is not to modify that and instead do a separate install using one of the above

PYTHON WORKFLOWS: ANACONDA

Individual Edition

Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

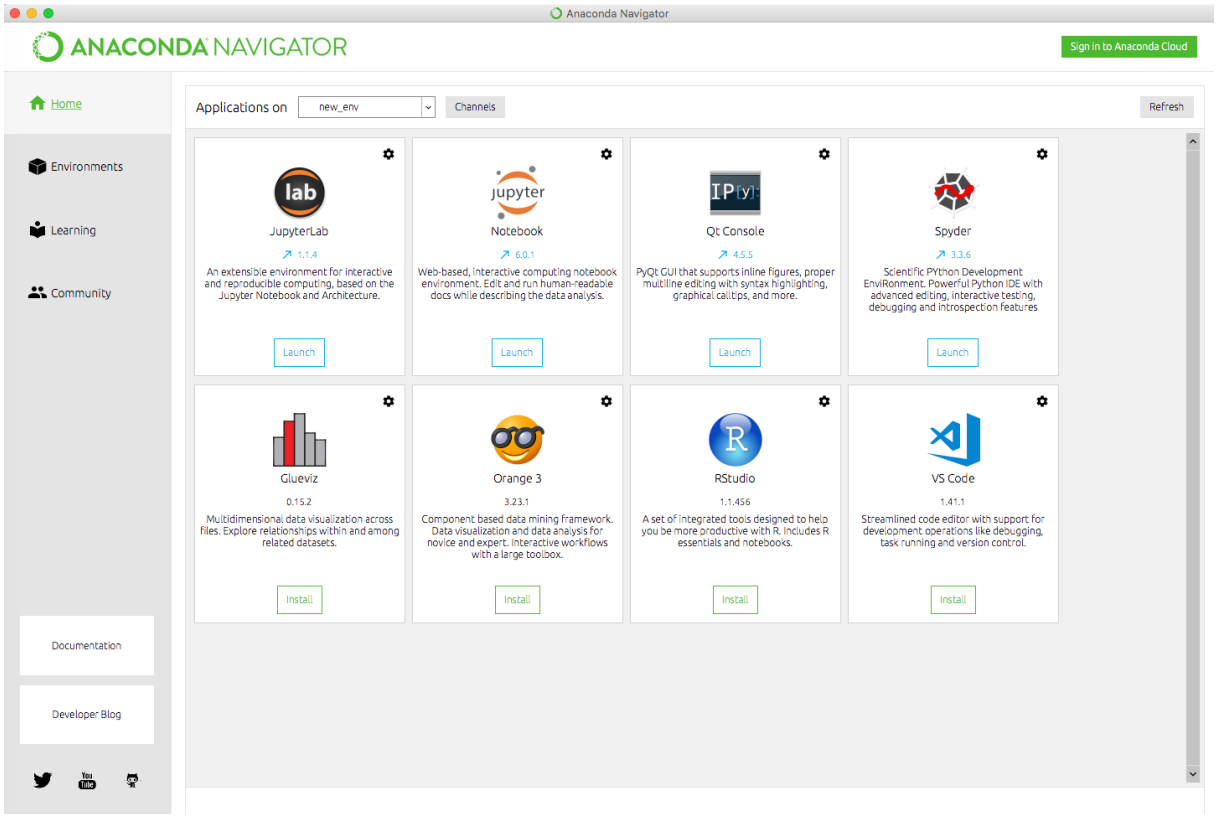


Open Source

Conda Packages

Manage Environments

PYTHON WORKFLOWS: ANACONDA



Positives

Installs almost everything you will ever need

conda package manager and update system

has some useful packages — starts with navigator

Negatives

takes about 8 GB

Installs a lot stuff you may never use

Not as simple to control package/library versions when working in teams

PYTHON WORKFLOWS: ANACONDA

After installing Anaconda, open **conda** terminal

```
conda create -n ee541_demo python=3.8 anaconda
```

wait… type y and wait…

```
conda activate ee541_demo
```

```
python –version
```

```
  Python 3.8.8
```

You can now work inside virtual environment "ee541_demo"

`conda deactivate` to quit virtual environment

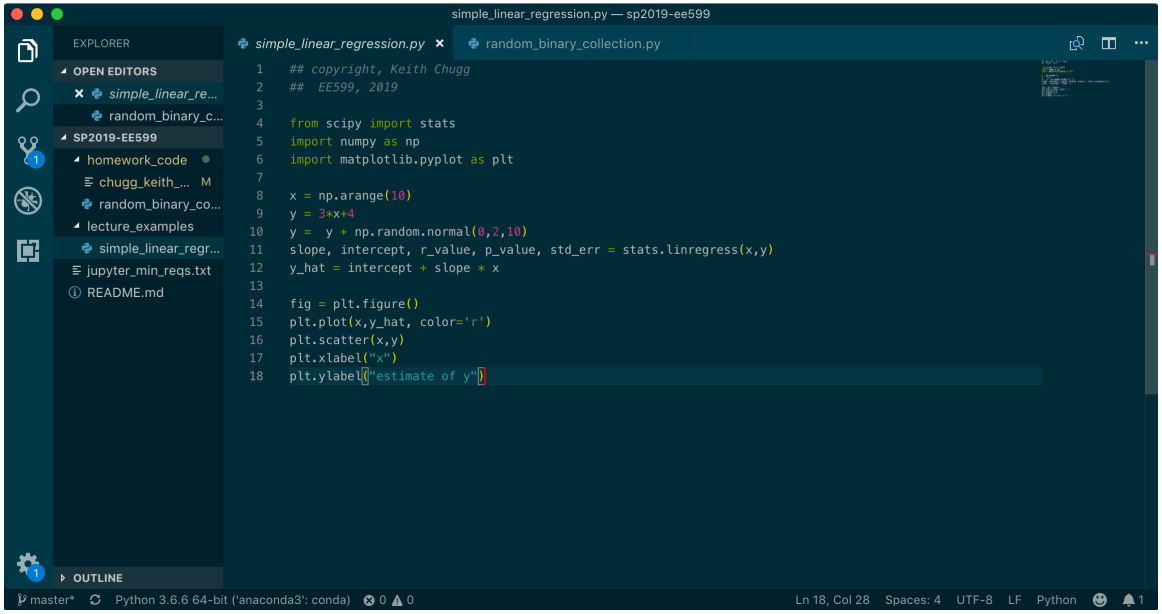
CONDA ENVIRONMENT

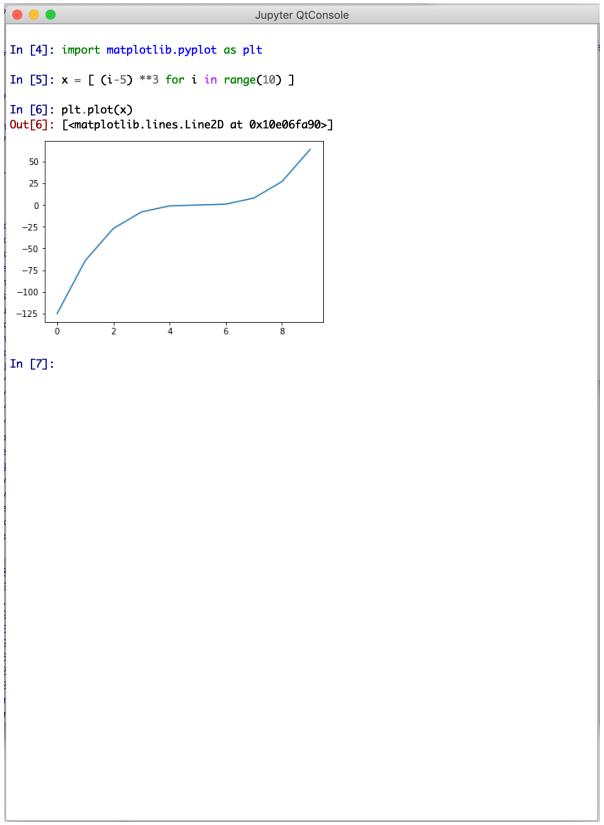
```
conda create --name ee541_work python=3.8
conda activate ee541_work
conda install numpy scipy scikit-learn matplotlib tqdm opencv pandas
# mac (mps)
conda install pytorch torchvision torchaudio -c pytorch
# pc (with cuda)
# conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia
conda install tensorboard
conda install jupyter seaborn h5py
pip install pytorch_model_summary torchsummary flashtorch torchviz


jupyter notebook
# http://localhost:8888/


tensorboard --logdir=runs
# http://localhost:6006/
```

EDITORS, IDES, AND DEBUGGERS

- Language sensitive text editors
  - Sublime text, **Microsoft Code**, BBEdit, etc.
- Some have additional features
  - Python Debugger – e.g. breakpoints, variable inspection, etc.
  - Git integration, extension system, latex, support for other languages, etc.
- All of these can be used with Anaconda and/or pyenv
  - qtConsole, CLI, jupyter notebook

PYTHON WORKFLOWS: MSCODE + QTCONSOLE





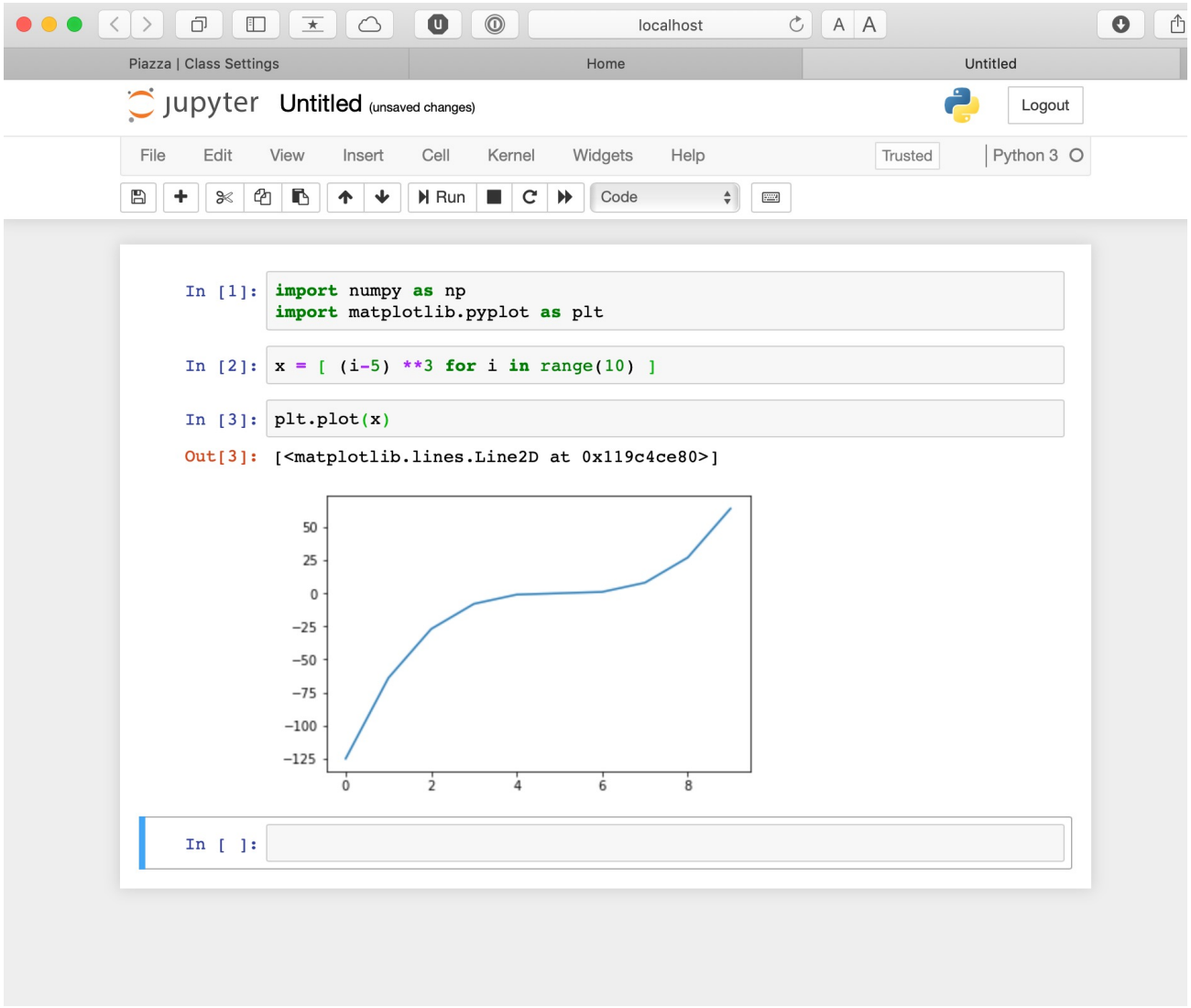
MS Code can be used a text editor in this workflow and/or you can run and debug inside MS Code — i.e., it is an IDE

OTHER WORKFLOWS

all of these can be used with **Anaconda** install or **pyenv**

- Jupyter Notebook
  - Web-based interactive development
  - Good for teaching or showing results to others
- Jupyter qtConsole
  - “Smart terminal” where you can plot
  - Good command history and completion
- CLI
  - Useful for “headless” operation – e.g. AWS, google cloud
  - Plots have to be saved and viewed elsewhere

PYTHON WORKFLOWS: JUPYTER NOTEBOOKS



JUPYTER NOTEBOOK – GETTING STARTED

Installation

Conda environment :
conda install -c conda-forge notebook

Pip environment :
pip install notebook

Starting jupyter:

Go to terminal (Mac/Linux) or Command prompt (Windows) and type: jupyter notebook

JUPYTER PROJECT

- **Non-profit** open-source project developed in 2014
- Supports **interactive data science** and **scientific computing** across multiple programming languages .
- **Jupyter Notebook**: a Web-based application for documenting live-code with narrative text, equations and visualizations.
- **Detailed information**: **https://jupyter.org/**

WHAT IS NOTEBOOK ?

- A **single document** for following operations:
  - **Running code**
  - **Displaying outputs (images/plots)**
  - **Adding explanations (text blocks /formulas)**
- Major part of **data science workflow** at companies
- Easier to share notebooks when working with others for data analysis

jupyter nbviewer

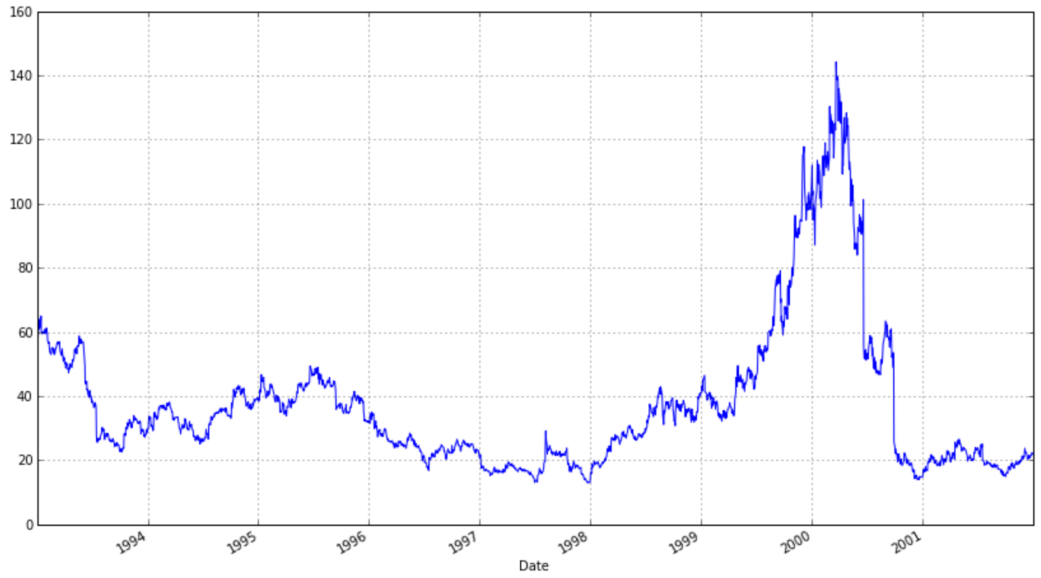
JUPYTER FAQ

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.figsize(14,8)

from zipline.algorithm import TradingAlgorithm
from zipline.transforms import MovingAverage, batch_transform
from zipline.utils.factory import load_from_yahoo
```

In [3]:
```
data = pd.load('talk_px.dat') #data = load_from_yahoo(stocks=['AAPL', 'PEP', 'KO']); data.save('talk_px.dat')
```

In [4]:
```
data['AAPL'].plot()
```

Out[4]:
```
<matplotlib.axes.AxesSubplot at 0xa80784c>
```

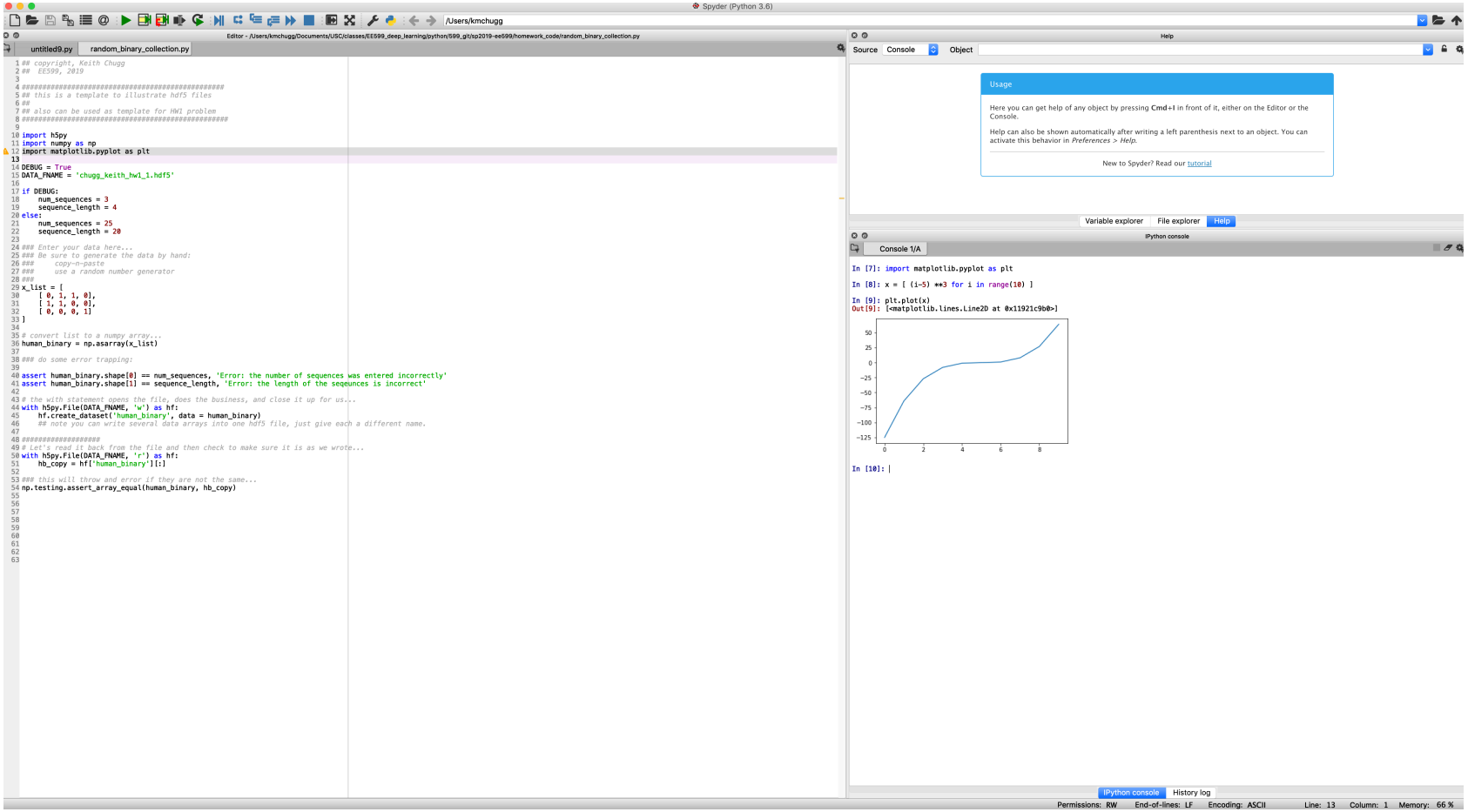


In [5]:
```
# Simplest possible algorithm
class BuyApple(TradingAlgorithm): # inherit from TradingAlgorithm
    def handle_data(self, data): # overload handle_data() method
        self.order('AAPL', 1) # stock ='AAPL') to order and amount (=1 shares)
```

Sample
(https:/
ecki/396

Host
https
ki/A-
Note

PYTHON WORKFLOWS: ANACONDA-SPYDER



Spyder IDE included with Anaconda:

- \+ *integrated with Jupiter qtConsole, includes debugger, simple, closest to Matlab*
- \- *editor is slow for larger files, similar flow can be replicated with editor of choice*

PYTHON W NSOLE

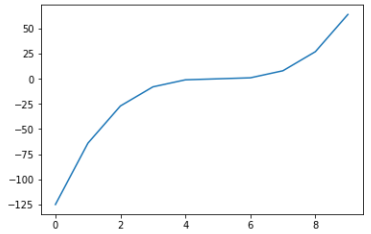
random_binary_collection.py — UNREGISTERED

```python
## copyright, Keith Chugg
##  EE599, 2019

################################################
## this is a template to illustrate hdf5 files
##
## also can be used as template for HW1 problem
################################################

import h5py
import numpy as np
import matplotlib.pyplot as plt

DEBUG = True
DATA_FNAME = 'chugg_keith_hw1_1.hdf5'

if DEBUG:
    num_sequences = 3
    sequence_length = 4
else:
    num_sequences = 25
    sequence_length = 20

### Enter your data here...
### Be sure to generate the data by hand:
###     copy-n-paste
###     use a random number generator
###
x_list = [
    [ 0, 1, 1, 0],
    [ 1, 1, 0, 0],
    [ 0, 0, 0, 1]
]

# convert list to a numpy array...
human_binary = np.asarray(x_list)

### do some error trapping:

assert human_binary.shape[0] == num_sequences, 'Error: the number of sequences was entered incorrectly'
assert human_binary.shape[1] == sequence_length, 'Error: the length of the seqeunces is incorrect'

# the with statement opens the file, does the business, and close it up for us...
with h5py.File(DATA_FNAME, 'w') as hf:
    hf.create_dataset('human_binary', data = human_binary)
    ## note you can write several data arrays into one hdf5 file, just give each a different name.

###################
# Let's read it back from the file and then check to make sure it is as we wrote...
with h5py.File(DATA_FNAME, 'r') as hf:
    hb_copy = hf['human_binary'][:]

### this will throw and error if they are not the same...
np.testing.assert_array_equal(human_binary, hb_copy)
```

On master* in sp2019-ee599, Line 1, Column 1 — Spaces: 4 — Python

Jupyter QtConsole

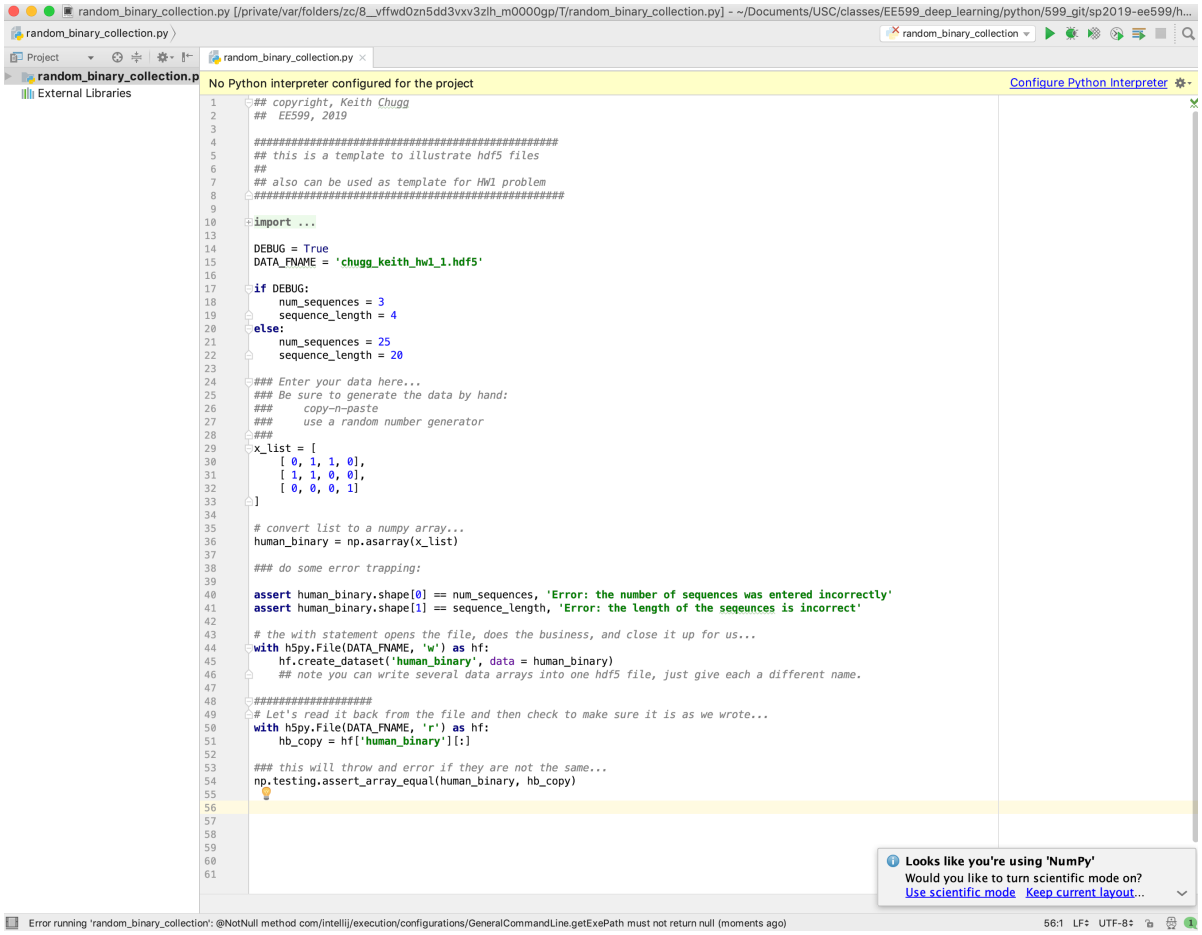
```
In [4]: import matplotlib.pyplot as plt

In [5]: x = [ (i-5) **3 for i in range(10) ]

In [6]: plt.plot(x)
Out[6]: [<matplotlib.lines.Line2D at 0x10e06fa90>]
```




```
In [7]:
```

not tied together, simple text editor + qtConsole

PYTHON WORKFLOWS: PYCHARM



\+ *more polished than Spyder, more widely used. Supports pyenv and Anaconda*

\- *Slows down when using Anaconda install because it indexes libraries*

PYTHON WORKFLOWS: PYENV VIRTUAL ENVIRONMENT

- To install pyenv, check its websites
  - https://github.com/pyenv/pyenv
  - https://github.com/pyenv/pyenv-virtualenv
- The second link has a good description of how to:
- install various Python versions (e.g., 2.7.x, 3.6.9, 3.7.1)
- create, manage and delete virtual-environments on top of these base Python installs

PYENV VIRTUAL ENVIRONMENT

```
pyenv install -l                                  list available python base python versions
pyenv install 3.6.9                               install python 3.6.9 as a base version
pyenv virtualenv 3.6.9 new_env2                   create a new virtualenv new_env2
pyenv activate new_env2          activate new_env2— this is your current python environment
pip install -r requirements.txt                  install libraries from a specified list
pip install numpy                                 add other libraries as needed
pip freeze                                        list the libraries installed in new_env2
pip freeze > new_requirements.txt                 save the new set-up to replicate elsewhere


pyenv virtualenvs                                 list all your virtaulenvs
pyenv uninstall new_env2                          deletes the virtual environment
```

Issue with many open source solutions is minor incompatibilities by version

This approach allows you to set up and replicate on multiple machines the same python set-up (including all versions)

GOOGLE COLAB / KAGGLE / AMAZON SAGEMAKER

- Directly **execute code** in browser.
- Can save as notebook files (.ipynb) in **google drive**.
- Pull code and models from GitHub
- Notebooks execute on **cloud servers** (access to **GPUs** and **TPUs**)

DEEP LEARNING IN THE CLOUD

- Google Colab
- Google Cloud GPU
  - Apply for $300 credit
- Kaggle
  - 30 hours/week
- Gradient
- Amazon SageMaker Studio Lab
- Saturn Cloud