# CIS192 Python Programming

## Machine Learning in Python

Harry Smith

University of Pennsylvania

October 18, 2017

# Outline



# Installation & Access

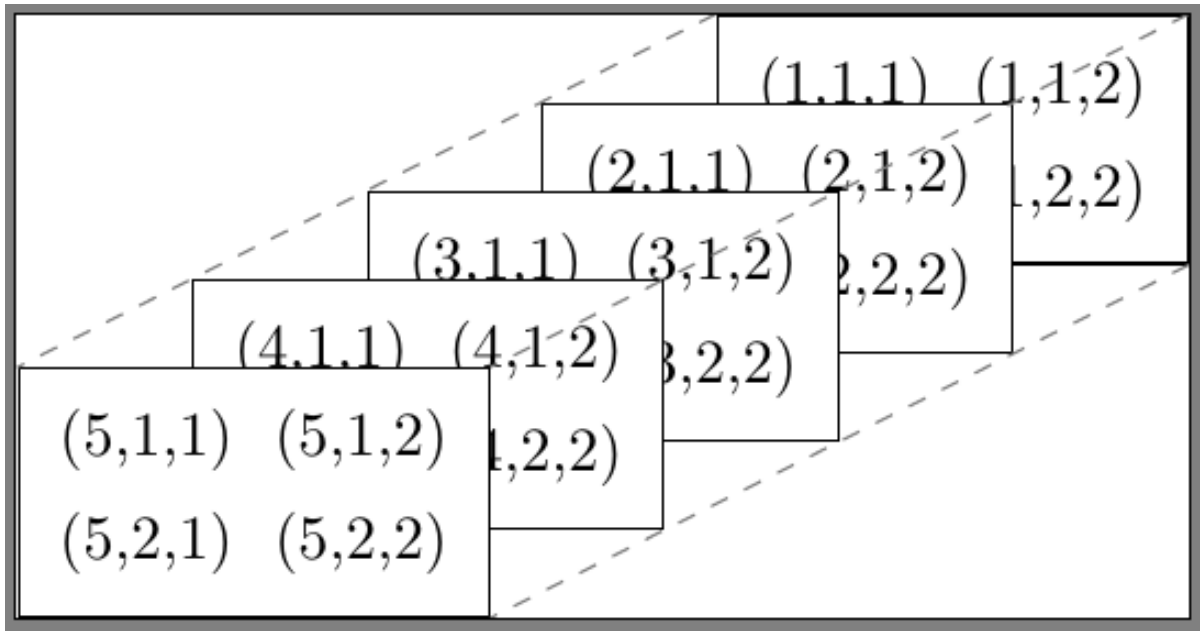
**It is strongly recommended that you have the VM installed for this assignment.**

If you need to install on your own device:
`pip3 install -U numpy scipy matplotlib ipython[notebook] scikit-learn`
Everybody install these packages now.

# Numpy and Scipy

- Libraries for sophisticated mathematics and mathematical computing in Python.
- Include libraries for linear algebra.
- Optimized for efficient machine learning.
- Useful for slicing, dicing, and comparing.

# Visual Reference for Understanding 3D Matrices




# Matplotlib

- Library for plotting and visualization.
- We will look at this in much more depth next week, but we'll use it first today.
- Use it to look at data before attempting machine learning techniques.
- Interfaces well with the IPython (an alternative shell for Python development).

# Scikit Learn

- A machine learning library for Python.
- Uses numpy and scipy.
- Comes with a broad array of built in machine learning algorithms
- Source of some good toy datasets!

# Outline



# Reducing Objects to Data

- Interesting machine learning can be done on people, cars, airline flights, restaurants, pictures, polls, trends, etc.
- How to represent diverse and mixed information?
- Representation is at the core of performing machine learning.

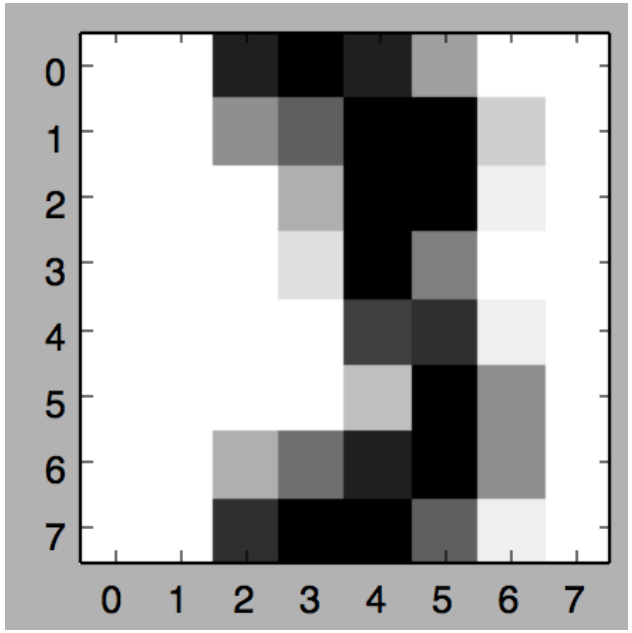
# The Iris Dataset

The well-known Iris dataset (https://archive.ics.uci.edu/ml/datasets/Iris) models Irises as four simple numbers:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

# The Digits Dataset

What is the best way to represent a hand-drawn digit that fits in a small square?
How should we visualize this? How should we let a computer think about it?




# Bags of Words

We often represent text as a “bag of words”. That is, we record the number of times each word appears in document in an array. This allows us to compare the arrays against each other using standard methods.

# Tf-Idf

The *term frequency-inverse document frequency* transformer also uses bag of words representation, but scales words according to how common they are in the broader corpus, preventing too much matching on “and”, “the” etc.

1. Bonus points for appearing a lot in a given document
2. Penalties for appearing a lot in the entire *corpus*.
3. Even if ’and’ appears more than any other word in an email, it’s not a very helpful word for figuring anything out about the email

# Outline



# Coats and Bars

Suppose you didn't know the meaning of "boat" or "car" but you had a stack of photographs of boats and cars.

Could you somehow sort them into two stacks, which corresponded to boats and cars?

# K-Means

- `cluster.KMeans()`
  - Parameter: `n_clusters` - specifies the number of clusters desired.
- Randomly assign initial position for each cluster.
- Repeat until stable:
  - Assign every point to its closest cluster $c_i$.
  - Move $c_i$ to the center of points that are assigned to it.

  Every point is labeled with its cluster.

# Principal Component Analysis

- How can we reduce a data point into its most basic components?
- We really only care about the features of the data that might distinguish them from each other, so why not focus here?
- PCA takes data from a set and transforms each point into a set of orthogonal components that explain most of the variance of the data.

# Other Models

- Gaussian Mixture Models general K-Means - K-means assumes clusters look like circle, where GMM can handle arbitrary elliptic clusters.
- Affinity Propagation ( (cluster.AffinityPropagation()) identifies *exemplars* - points that can stand in for a given cluster. It may vary the number of clusters depending on the exemplars it finds.