CMSC 471

ML : Tools

(in Python)

Motivation

- Machine learning involves working with data
  - analyzing, manipulating, transforming, …
- More often than not, it's numeric or has a natural numeric representation
- Natural language text is an exception, but this too can have a numeric representation
- A common data model is as a N-dimensional matrix or tensor
- These are supported in Python via libraries

Motivation

- Python is a great language, but slow compared to Java, C, and many others
- Python packages are available to represent, manipulate and visualize matrices
- We'll briefly review numpy and scipy
  - Needed to create or access datasets for ML training, evaluation and results
- And touch on pandas (data analysis and manipulation) and matplotlib (visualization)

Numpy

- Numpy stands for numerical python.
  - Open-Source library
  - Useful for crunching **numbers**.
- Why **Numpy**?
  - NumPy supports features needed for ML
    - Typed N-dimensional arrays (matrices/tensors)
    - Fast numerical computations (matrix math)
    - High-level math functions
  - Python does numerical computations slowly and lacks an efficient matrix representation
  - 1000 x 1000 matrix multiply
    - Python triple loop takes > 10 minutes!
    - Numpy takes ~0.03 seconds
- To install Numpy:
  - `pip install numpy`

Resources:
https://www.w3schools.com/python/numpy/default.asp

NumPy Arrays, Basic Properties

```
>>> import numpy as np
>>> a= np.array([[1,2,3],[4,5,6]],dtype=np.float32)
>>> print(a.ndim, a.shape, a.dtype)
2  (2, 3)  float32
>> print(a)
[[1. 2. 3.]
 [4. 5. 6.]]
```

Arrays:

1. Can have any number of dimensions, including zero (a scalar)
2. Are **typed**: np.uint8, np.int64, np.float32, np.float64
3. Are **dense:** each element of array exists and has the same type

NumPy Array Indexing, Slicing

```
a[0,0]  # top-left element
a[0,-1] # first row, last column
a[0,:]  # first row, all columns
a[:,0]  # first column, all rows
a[0:2,0:2]  # 1st 2 rows, 1st 2 columns
```

Notes:

- Zero-indexing
- Multi-dimensional indices are comma-separated)
- Python notation for slicing

Pandas

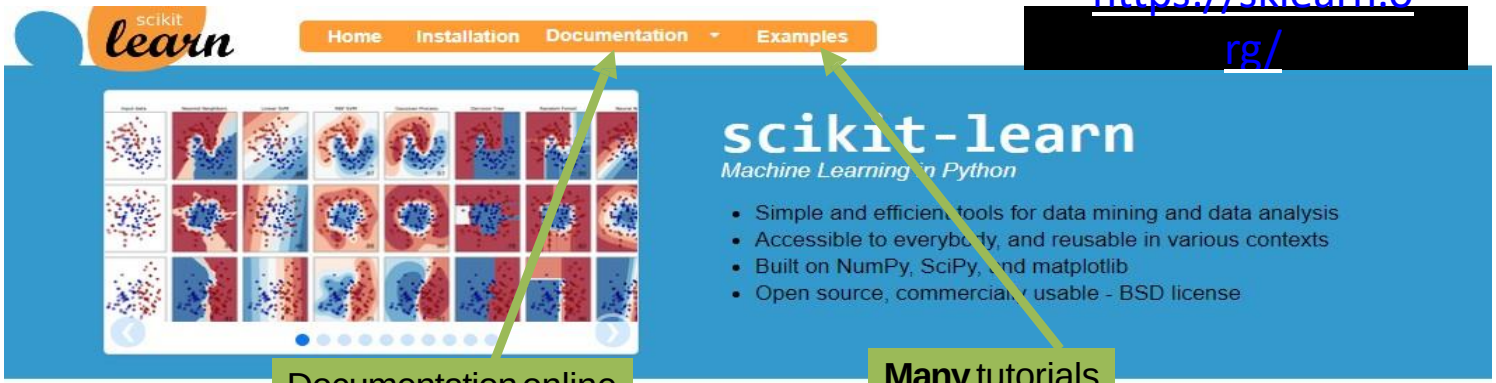
- Python opensource library for data analysis
- Pandas is structured around **DataFrame** objects.
- DataFrame can be used to load your data and pandas function can help you manipulate the data.
- Some of the things that Pandas support:
  - **Reading and writing** data from different formats such as .csv, .txt, .xlsx, SQL, etc.
  - **Merging and joining** of data sets
  - Data alignment and handling of missing data, data type etc.
- To install Pandas:
  - `pip install pandas`
- To import Pandas:
  - `import pandas as pd`

- Pandas complete documentation:
  - https://pandas.pydata.org/

SciKit Learn

- Scikit-Learn (**sklearn**) is an open-source library for Machine Learning in Python.
- Complete Documentation:
  - https://scikit-learn.org/stable/
- User Guide has list of most ML models that you will be needing to get started

https://sklearn.org/



Documentation online

Many tutorials

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

Scikit Learn

- Example of linear model

```
>>> from sklearn import linear_model
>>> reg = linear_model.LinearRegression()
>>> reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
```

Import model

Create instance

Fit data and train

Preprocessing data

- Clean the data:
  - Pick relevant features
  - Change data type
  - Fill in Null Values; Clean garbage values etc.
- Convert Categorical Variables into Numerical Variables
  - sklearn.preprocessing:
    - Includes function for binarizing data, label encoding, one hot encoding…
  - Using pandas:
    - `pd.Categorical(pd.factorize(…)[0])`

Test-Train Split

- You can use Numpy or Pandas to randomly select datapoints to create your training and test data
  - Remember test and training data must be disjoint
- Sklearn also has helpful functions to split your dataset: Model Selection
  - Train_test_split

```
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
```

Evaluation

- Common evaluation metrics can be found in [sklearn.metrics]

```
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]
>>> accuracy_score(y_true, y_pred)
0.5
>>> accuracy_score(y_true, y_pred, normalize=False)
2
```

Implementing ML in Python

- Numpy, Pandas and Scikit Learn **work together.**
- Default implementation:
  1. Load/clean/manipulate your data using Pandas
  2. Translate your Pandas DataFrame into a Numpy array
  3. Feed it to Scikit Learn function(s)

Example Implementation

Iris Data Classification with Random Forest Classifier : here