Numpy / Scipy

RJ Nowling

Python Data Types

- int ("grows" into a long)
- float (double precision)
- complex (real and imag doubles)
- str (string)
- bool (Boolean)

What is Numpy?

- Matrix Library
- Memory-efficient data structures -- arrays
  - Used in scikit-learn, matplotlib, others
- Expressive API for indexing and operations
- Time-efficient algorithms
  - Calls C and Fortran libraries where possible

How Numpy is Used – Plotting

- Variables stored as 1D arrays

```
plt.plot(xs, ys)
plt.scatter(xs, ys)
```

How Numpy is Used – Modeling

- Modeling (regression, classification, etc.)
- Model independent variables stored as a matrix (X)
  - Each row is an observation
  - Each column is an independent variable
- Dependent variables stored as a 1D vector (y)

```
rf = RandomForestClassifier()
rf.fit(X, y)
pred_y = rf.predict(X)
```

Iris Data Set Example

Memory Overhead

- Python doesn't have primitive types like Java
- Everything is an object
- Objects have several required fields:
  - Number of references (for garbage collection)
  - Enumeration for object type
  - Data size
  - Actual data value (e.g., long, double)
- This abstraction makes coding easier (e.g., heterogenous lists) but numerical computing less space efficient

Numpy Array Data Structure

- Multidimensional (1, 2, 3, etc.)
- Only stores 1 type (e.g., int, float) at a time
- Allocates a large block of contiguous memory
  - Very little memory overhead
  - Single object for array itself
  - Rest of space is for storing values directly
- Greater control over types
  - e.g., unsigned ints, 2, 4, and 8 bytes
  - 4 or 8 byte floats
- No efficient operation for easily appending data – fixed size

Creating Numpy Arrays

- From Python Lists

```
array = np.array([1, 2, 3, 4, 5], dtype=np.float32)
```

- Using special functions
  - fill with all zeros

```
array = np.zeros(4, dtype=np.int32)
```

  - fill with all ones

```
array = np.ones((4, 5), dtype=np.float64)
```

Numpy Attributes

- Dimensions

```
array.ndim – property, int
```

- Shape – length of each dimension

```
array.shape – property, tuple of ints
```

- Data Type

```
array.dtype – property, constant
```

Indexing

- Single element (returns a scalar)

```
array[0]
array[1, 2]
array[1, 2, 3, 4, 5]
```

- Selecting a single dimension

```
array[:, 5, :] – returns a 2D array
```

Indexing

- Subset of elements

```
array[[7, 3, 5, 3]]
array[:, [2, 3, 4]]
```

- Reversing a 1D array

```
array[::-1]
```

Selecting entries

- Selecting entries that match a given value

```
mask = array == 1
array[mask]

mask = array > 5
array[mask]
```

Reshaping

Scikit Learn functions expect the variable matrix (often denoted X) to be 2 dimensional. If you have a single variable, a 1D array is often created. So, this needs to be reshaped into a 2D array before passing to Scikit Learn.

```
X = np.array([1, 2, 3])
X.shape == (3,)

X = X.reshape(:, np.newaxis)
X.shape == (3, 1)
```

Joining Arrays

Sometimes, it's easier to create variables through separate processes and then join them into a single matrix at the end. The horizontal stack function makes this easy:

```
X1 = np.zeros((4, 5))
X2 = np.zeros((4, 7))
X = np.hstack([X1, X2])
X.shape == (4, 12)
```

Note that the input matrices must have the same number of rows!