Python ML Tools
Motivation

• Python is a great language, but slow compared to Java, C, and many others
• Python packages are available to represent and operate on matrices
• We’ll briefly review numpy and scipy
• You need some familiarity to be able to create or access datasets for training, evaluation and results
What is Numpy?

• NumPy supports features needed for ML
  Typed multi-dimensional arrays (matrices)
  – Fast numerical computations (matrix math)
  – High-level math functions
• Python does numerical computations slowly
• 1000 x 1000 matrix multiply
  – Python triple loop takes > 10 min.
  – Numpy takes ~0.03 seconds
NumPy Arrays Can Represent ...

Structured lists of numbers

- Vectors
- Matrices
- Images
- Tensors
- Convolutional Neural Networks

\[
\begin{bmatrix}
  p_x \\
  p_y \\
  p_z
\end{bmatrix}
\]

\[
\begin{bmatrix}
  a_{11} & \cdots & a_{1n} \\
  \vdots & \ddots & \vdots \\
  a_{m1} & \cdots & a_{mn}
\end{bmatrix}
\]
NumPy Arrays Can Represent ...

Structured lists of numbers

- Vectors
- Matrices
- Images
- Tensors
- Convolutional Neural Networks
NumPy Arrays Can Represent ... 

Structured lists of numbers

• Vectors
• Matrices
• Images
• Tensors
• Convolutional Neural Networks
**NumPy Arrays, Basic Properties**

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

1. Arrays can have any number of dimensions, including zero (a scalar).
2. Arrays are typed: np.uint8, np.int64, np.float32, np.float64
3. Arrays are dense. Each element of the array exists and has the same type.
NumPy Array Indexing, Slicing

\[ x[0,0] \quad \# \text{top-left element} \]
\[ x[0,-1] \quad \# \text{first row, last column} \]
\[ x[0,:] \quad \# \text{first row (many entries)} \]
\[ x[ :, 0] \quad \# \text{first column (many entries)} \]

Notes:

– Zero-indexing
– Multi-dimensional indices are comma-separated (i.e., a tuple)
ScyPy

• SciPy builds on the NumPy array object
• Adds additional mathematical functions and sparse arrays
• **Sparse array** is one where most elements = 0
• An efficient representation only explicitly encode the non-zero values
• Access to a missing element returns 0
SciPy Sparse Array Use Case (1)

• NumPy and SciPy arrays are numeric
• We can represent a document’s content by an vector of features
• Each feature is a possible word
• A feature’s value is:
  – TF: number of times it occurs in the document;
  – TF-IDF: ... normalized by how common the word is
  – And maybe normalized by document length
SciPy Sparse Array Use Case (2)

• We may be interested only in the 50,000 most frequent words found in a large document collection and ignore others

• We assign each English word a number

• The sentence “the dog chased the cat”
  – Would be a numPy vector of length 50,000
  – Or a sciPy sparse vector of length 4

• A 800 words news article may only have 100 unique words; The Hobbit has about 8,000