14.3

Python Tools for Machine Learning

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)

What is 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

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 …

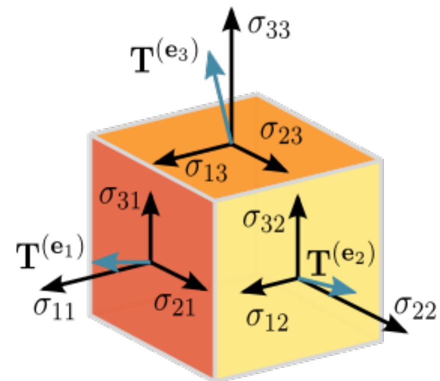
Structured lists of numbers

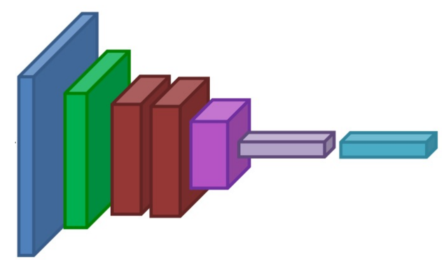
- Vectors
- Matrices
- **Images**
- Tensors
- Convolutional Neural Networks

NumPy Arrays Can Represent …

Structured lists of numbers

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





NumPy Arrays, Basic Properties

```
>>> 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

SciPy

- SciPy builds on the NumPy array object
- Adds additional mathematical functions and *sparse arrays*
- **Sparse array:** one where most elements = 0
- An efficient representation only implicitly encodes the non-zero values
- Access to a missing element returns 0

SciPy sparse array use case

- NumPy and SciPy arrays are numeric
- We can represent a document's content by a vector of features
- Each feature is a possible word (aka term)
- A feature's value might be any of:
  - **TF** term frequency: the number of times a term occurs in the document;
  - **TF-IDF** term frequency normalized by IDF (inverse document frequency) to favor uncommon words
  - and may be normalized by document length as well

SciPy sparse array use case

- Only model 50k most frequent words found in a document collection, ignoring others
- Assign each unique word an index (e.g., dog:137)
  - Build python dict **w** from vocabulary, so w['dog']=137
- The sentence "the dog chased the cat"
  - Would be a *numPy vector* of length 50,000
  - Or a *sciPy sparse vector* of length 4
- An 800-word news article may only have 100 unique words; [The Hobbit](#) has about 8,000

More on SciPy

SciPy.org

SciPy.org | Docs | SciPy v1.4.1 Reference Guide

SciPy Tutorial

- Introduction
- Basic functions
- Special functions (scipy.special)
- Integration (scipy.integrate)
- Optimization (scipy.optimize)
- Interpolation (scipy.interpolate)
- Fourier Transforms (scipy.fft)
- Signal Processing (scipy.signal)
- Linear Algebra (scipy.linalg)
- Sparse eigenvalue problems with ARPACK
- Compressed Sparse Graph Routines (scipy.sparse.csgraph)
- Spatial data structures and algorithms (scipy.spatial)
- Statistics (scipy.stats)
- Multidimensional image processing (scipy.ndimage)
- File IO (scipy.io)

See the SciPy tutorial Web pages