Representing Data

CS/DSA 5970: Machine Learning Practice
Connecting Real World Data to our ML Tools

Often have a huge disconnect between the two. Our ML tools often rely on:

• Well-defined formatting of the data
• Cut into distinct *examples*. Each example:
  – List of property values. Most often assume each example consists of the same properties.
  – Label / expected output value (for supervised problems)
Connecting Real World Data to our ML Tools

(cont) Our ML tools often assume:

• Properties are numerical
• Statistical independence between the different examples
• All examples are drawn from the same statistical distribution
Connecting Real World Data to our ML Tools

Real world data can:
• Be weakly formatted
• Properties can be enumerated types (e.g., strings such as “circle”, “square”)  
• Values can be incorrect
• Values can be missing
• Different examples can have different properties
• Distribution that we draw examples from can be changing in time
Connecting Real World Data to our ML Tools

Transforming the raw data to a well-formatted form is a key first step:

• This step can take much of our project time, depending on the form of the data

• How careful we are in taking this step can dramatically affect everything else we do

• As a byproduct of this step, it is important to really understand the nature of your data
Roadmap

• Pandas package
  – Importing data from standard formats
  – Data massaging
• Numpy package
  – Efficient representation of numerical data
• Matplotlib package
  – Matlab-like visualization package
Pandas

Toolkit for data handling and analysis

• File I/O, including csv files
• Hooks for visualization
• Basic statistics
• Data selection and massaging
• SQL-type operations
Classes Provided by Pandas

Two primary Python classes:

• **Series**: 1D data
  – Indexed by integer location in the array or by some index variable (index values can be numerical or strings)

• **DataFrame**: 2D data
  – Each dimension indexed by integer index or other index variable
  – Most common for us: examples (rows) x features (columns)
Some Useful DataFrame Operations

- Data exploration:
  - Show row / column index names
  - Compute statistics for individual columns
- Create a new DataFrame that contains a subset of the rows and/or columns
- Remove or repair rows and/or columns that contain invalid data
- Export data to a numpy array for use with ML methods
Numpy

Numerical methods package
• Representation of vectors, matrices, tensors
  – Vector: yet another way of representing a list of numbers
• Implementation of many linear algebra type operations
  – Computing matrix inverses, Singular Value Decomposition …
• Basis for many ML packages, including Scikit-Learn
Real-Time Activity Recognition for Assistive Robotics

OU Crawling Assistant (Kolobe, Fagg, Miller, Ding)

Scientific American (Oct 2016)
Infants Learning to Crawl

• Learning to crawl is in part a reinforcement learning process:
  – Initially: making novel things happen (such as the body rolling or shifting a bit) is rewarding
  – Eventually: it becomes rewarding to grasp toys (or car keys)

• These rewards are important:
  – Practice many types of motor skills
  – Drives the development of spatial skills
Infants at Risk for Cerebral Palsy

- Initial exploratory movements do not result in interesting things happening
- These infants show a dramatic delay in the onset of crawling
- This impacts the learning of other motor skills & the development of spatial skills
SIPPC Crawling Assistant

- 6-Axis Load Cell
- Wide-Angle Cameras
- Vertical Lifts
- EEG Head Net
- Infant Support
- Covered Omni-Wheels

Andrew H. Fagg: Machine Learning Practice
Kinematic Capture Suit

IMU-based kinematic suit
• 12 sensors mounted in suit
• Real-time reconstruction of body posture
• Recognition of crawling-like actions

Southerland (2012)
Infant-Robot Interaction

Three modes of interaction:

- **Force control**: robot velocity is linearly related to ground reaction forces
- **Power steering**: small ground reaction forces produce a substantial robot movement
- **Gesture-based control**: recognized crawling-like movements produce robot movement
Machine Learning Questions

- Predict robot motion from kinematic data
- Predict visual attention from kinematic and robot data
- Predict limb motion from EEG data
- Predict visual attention from EEG data
- …
Introduction to Pandas

CS/DSA 5970: Machine Learning Practice
Pandas Roadmap

- Importing data from Comma Separated Values (CVS) file
- Exploring data
- Indexing rows and columns
Pandas

• Live example
Pandas: Basic Plotting

CS/DSA 5970: Machine Learning Practice
• Live example
Introduction to Numpy

CS/DSA 5970: Machine Learning Practice
Numpy Rodmap

• Transforming Pandas data to a Numpy matrix
• Indexing Numpy matrices
• Combining vectors to create a matrix
• Live example
Visualization with Matplotlib

CS/DSA 5970: Machine Learning Practice
Matplotlib Roadmap

- Creating temporal figures
- Creating scatter plots
- Tuning the display of figure elements
- Subplots
- Repairing a Pandas dataset & visualizing the results
• Live example
Pipelines

CS/DSA 5970: Machine Learning Practice
Pipelines

• Data processing often involves multiple computational steps, only some of which involve ML.
• The Scikit-Learn Pipeline class provides a clean interface for expressing these steps.
  – Each step (or pipeline element) is implemented by a class that adheres to a standard interface.
  – This allows us to mix-and-match elements for different purposes.
A pipeline element class is some combination of:

• Estimator
• Transformer
• Predictor
Estimator: given a dataset, compute some measure or some model parameters

• Implements the fit() method
  – Takes as input one or two datasets (input data & desired output)

• Our ML methods are estimators
Flavors of Pipeline Element Classes

Transformer: modifies a dataset in some way

• Implements the transform() method
  – Takes as input one dataset
  – Returns a dataset

• Transformers can be used to clean a dataset before it is used by a ML method
Flavors of Pipeline Element Classes

Predictor: predicts some quantity given a dataset
• Implements the predict() method
  – Takes as input one dataset and returns a different dataset
• Implements a score() method that evaluates a prediction
  – Takes as input an input dataset and an expected output dataset
  – Returns a score
Pipeline Notes

• Pipeline elements are classes in and of themselves
• The Pipeline class is also a pipeline elements
  – So, we can nest pipelines!
• Python classes can inherit from multiple classes
  – An element can be both an Estimator and a Predictor
• Datasets are generally Pandas objects or Numpy tensors
  – A particular pipeline element will use only one type as an input and one type as an output
• Live example
Creating Pipeline Elements

CS/DSA 5970: Machine Learning Practice
• Live example
Creating Pipeline Element Classes

CS/DSA 5970: Machine Learning Practice
• Live example
Pipeline Example: Computing Derivatives

CS/DSA 5970: Machine Learning Practice
Computing Derivatives

Numerical differentiation of a timeseries $\mathbf{x}$:

• For each time $t$:

$$\dot{x}[t] \approx \frac{x[t + 1] - x[t]}{\Delta t}$$

• Often will want to include some filtering to address the discrete nature of the data (though we won’t do this here)
• Live example
Pipeline Example: Linear Imputer

CS/DSA 5970: Machine Learning Practice
Linear Imputer

For our implementation: we will take advantage of the DataFrame.interpolate() method
• Live example
Pipeline Example: Building a New Pipeline

CS/DSA 5970: Machine Learning Practice
• Live example
Representing Categorical Data

CS/DSA 5970: Machine Learning Practice
Handling Categorical Data

• Discrete, finite set of values
  – Most often the different values are strings or symbols
  – Also known as an enumerated type

• Most ML algorithms only address numerical data, so need some way of transforming from categorical values to some numerical representation
Handling Categorical Data

Often done in stages:

• Identify the set of possible categorical values
• Transform these values into an integer index
  – Order is arbitrary
• Transform the integer index into a 1-hot encoding
  – Array of bits: one bit per possible index value
  – For a given categorical value, only one bit is one and all others are zeros
• Different from book: use OneHotEncoder to do all of this!
• Live example
Example: Adding Data to a DataFrame

CS/DSA 5970: Machine Learning Practice
Example: Adding Data to a DataFrame

Our example:
• Create a discrete label as a function of Z
• Convert discrete label to a 1-Hot Encoding
• Add these columns to the original DataFrame
• Live example