Pandas and Scikit-Learn

Andrew H. Fagg
Symbiotic Computing Laboratory

Is Jupyter Working?

Test Data Sets

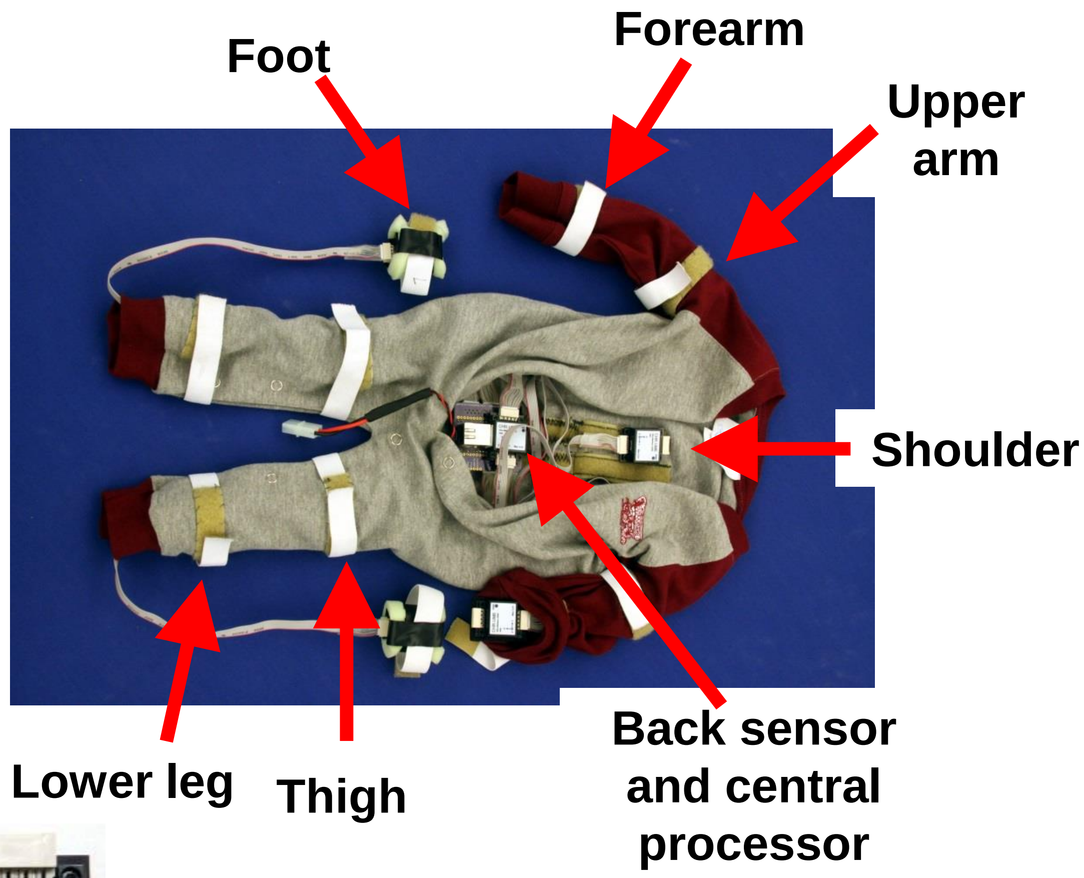
/home2/fagg/datasets

- book/housing/: Housing dataset from the book
- baby1/: Infant kinematic datasets
  - k1: basic table
  - k2: much larger table, including some robot information

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)

SIPPC Crawling Assistant

6-Axis Load Cell

Wide-Angle Cameras

EEG Head Net

Vertical Lifts

Infant Support

Covered Omni-Wheels

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

Python Lists

Python mechanism for implementing arrays

- Zero-indexed
- Bounds checking
- Elements can contain arbitrary data (including other arrays)

```
b = (2, 4, 7, 8, 1, 'foo', 'bar', 42)
```

Python Lists

```
b = (2, 4, 7, 8, 1, 'foo', 'bar', 42)
b[3]
```
8
```
b[6]
```
‘bar’

```
# Reslicing
b[2:4]
```
(7, 8)

Fundamental Data Structure in Python: Dictionaries

Implementation of a map

- Map contains a set of keys (keys are unique)
- Each key has arbitrary data associated with it

```
c = {0: 'zero', 5: 'five', 'foo': 'bar', 'baz': (42, 37)}
```

Dictionaries

```
c = {0: 'zero', 5: 'five', 'foo': 'bar', 'baz': (42, 37)}
c[0]
```
‘zero’
```
c[1]
```
KeyError
```
c[‘foo’]
```
‘bar’
```
c[‘bar’]
```
KeyError
```
c[‘baz’]
```
(42, 37)

Python Objects

Proper objects in the *object oriented programming* sense

- Instance variables: state describing the object
- Instance methods: methods that can be executed with respect to the object
- Underlying representation is a dictionary
  - Python is happy to allow us to make use of this fact…

An Example …

```
class testClass:
    def __init__(self):
        self.name = 'foo'
        self.value = 5
    def increment(self):
        self.value = self.value + 1
```

Constructor

Initialize instance variables

Another instance method

Using our Class

```
# Create a new instance
a = testClass()
a.value
5
a.increment()
a.value
6
a.name
‘foo’
```

Using our Class

```
a.increment
```

```
<bound method testClass.increment of
<__main__.testClass object at
0x7f7480431c88>>
```

- When you want to call a method, make sure you include the parens!

Modified Class

```
class testClass:
    def __init__(self):
        self.name = 'foo'
        self.value = 5
    def increment(self):
        self.value = self.value + 1

    def __getitem__(self, i):
        if i == 0:
            return self.name
        elif i == 1:
            return self.value
        else:
            return None
```

Allows array-like access

Using the New Access Method

```
a = testClass()
a[1]
5
a.increment()
a[1]
6
a[0]
'foo'
```

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

Data Structures

Two primary Python classes:

- Series: 1D data
  - Indexed by integer location in the array or by some index variable (which can have string values)
- DataFrame: 2D data
  - Each dimension indexed by integer index or other index variable

- Live demo…

- Live demo continued:
  - Plotting with Pandas & specifying horizontal axis variable

- Pipeline demo

Model Construction (Learning)

We want the "best" model as possible. One approach:

- Use the available data to select model parameters that optimize some performance metric
- Deploy the model

Model Construction (Learning)

How do we know that the model is really all that good?

Model Construction (Learning)

How do we know that the model is really all that good?

- We don't: our model could very well have ***overfit*** the data

Model Learning

Goal: we want our models to perform well on future data sets

- Our challenge is how to measure this *now*, so that we can make proper decisions about which model or model parameterization to choose
- Note the relationship with scientific theories: a good scientific theory is one that can make predictions about future experiments

Model Learning

- Future data are (we assume) statistically independent of the data we have to construct our model from
    - But: we assume that they come from the same distribution
- Our approach is to simulate future data: hold out some of the available data from the model building (training) process
    - After training, we then use this *test data set* to measure the difference between model predictions and truth

But is it that simple?

But is it that simple?

There is typically more than one model

- Different model forms / training procedures
- Different hyper-parameters
  - Learning rates, kernel sizes …

There could be *many* such choices (especially in the hyper-parameters)

One possible solution…

Pick the model form and hyper-parameters with the highest test set performance

One possible solution…

Pick the model form and hyper-parameters with the highest test set performance

- Because we are using the test set data to make this choice, we only know how the selected model will perform on *this* test data set…
  - It does not tell us about the future!
- Another take: the choice of “best” model (i.e., **model selection**) is another part of the model learning process
  - If the test set is about simulating future experience, then we should not use it for model selection, either

Training Set for Model Selection

What about using training set performance for model selection?

Training Set for Model Selection

What about using training set performance for model selection?

- We are back to our overfitting problem

A Step Back to the Science Side…

We are often wanting to answer the question: what is the best model form or learning algorithm?

- Another way to look at it: I hypothesize that my algorithm is better than your algorithm
- We assume already the “best” choice for hyper-parameters for each one
- Typically the number of model forms/algorithms is much smaller than the number of hyper-parameter choices

We will separate these questions in the learning and testing procedure

Model Learning and Selection Solution

- **Training data set**: use to choose model parameters
- **Validation data set**: for a give model form / algorithm, used to select the best hyper-parameters
- **Test data set**: use to compare form / algorithm

These different data sets must be statistically independent from one-another

Another Dimension

Model construction and evaluation is a statistical process

- Variations in the data that are available
- Some learning algorithms make random decisions

Another Dimension

Model construction and evaluation is a statistical process

- Variations in the data that are available
- Some learning algorithms make random decisions

We would like to same something more general than “this is the best choice for this model form / algorithm and this data”

Dealing with the Statistical Nature of Learning

Approach:

- For a given model form and parameter choices, don't construct a single model: construct N of them
- Measure performance for all N
- When comparing two different model forms or parameter sets, we can now ask a statistical question: are the performance distributions statistically different from one another?

Dealing with the Statistical Nature of Learning

Approach:

- For many hypothesis test types, we need to assume independence of each of the N performance measures
- Technically, this means that the training/validation/test data sets must be statistically independent from one-another
- But: this means that we need N times more data

Dealing with the Statistical Nature of Learning: Practice

Often, gathering more data is very expensive

- Instead, let's be clever in how we select our training/validation/test data

N-Fold Cross-Validation

Approach presented in chapter 2:

- Cut your data into two pieces: test data set and “other”
- Cut the “other” into N separate folds
- Construct N models

N-Fold Cross-Validation

- Construct N models:
  - Model 0: folds [0, 1, … N-2] for training; fold N-1 for validation
  - Model 1: folds [1, 2, … N-1] for training; fold 0 for validation
  - ….
- Select model hyper-parameters based on average validation set performance

N-Fold Cross-Validation

- Use test data set to measure performance of all N models for the selected hyper-parameters
- Use mean of performance across the N to compare model forms

Single Test Set Problem

- The N performance measures are not independent of one-another
- Our typical hypothesis testing methods will not apply here

Alternative N-Fold Cross-Validation

- Cut full data set into N folds
- Construct N models:
  - Model 0: folds [0, 1, … N-3] for training; fold N-2 for validation; fold N-1 for testing
  - Model 1: folds [1, 2, … N-2] for training; fold N-1 for validation; fold 0 for testing
  - ….

- Testing folds are independent of one-another. Hence, performance metrics are independent (somewhat)

Alternative N-Fold Cross-Validation

- Use validation folds for hyper-parameter selection
- Only after hyper-parameters are chosen, examine test set performance
- Use test set performance to compare model forms / algorithms

The Right Choice?

Because my typical use case is comparing multiple model forms, the latter is the proper way to proceed for my work

Yet Another Dimension: Hyper-Parameter Selection

Two possible approaches using the validation data set:

- For each model, pick the hyper-parameters that maximize its validation performance
- Pick the hyper-parameters that maximize the average validation performance

Latter tends to give more stable results

Classifiers

Classifiers

Given some example, which discrete case does it belong to?

Classifiers

Different types of classifiers

- Some directly emit the class
  - Example: in some decision trees, a leaf is associated with a specific class
- Many classifier types represent an intermediate score
  - Decision about the class is a function of the score (or scores)
  - In particular, we will have some decision boundary (a threshold) that distinguishes between one class and the other
  - How do we choose this threshold?