Python for Engineering Applications

Ryan Kinnear

University of Regina

With support from NSERC - USRA

& Dr. Laforge

What This Is About

- A presentation about solving hard problems with computers
- Focused more on Python as a tool, rather than for building end products
- Anyone studying Engineering, Science, Business or Math could benefit

What am I Going to Talk About?

- A modern perspective on problem solving
- How using Python got me an excellent job offer and impressed my boss
- Show you some stuff that may be useful in class
- I hope to introduce you to and get you thinking about a huge swath of problems and application areas
- Lots of interesting ideas / examples from a broad range of fields
- Not diving too much into specifics

Python

- High level “batteries included” scripting language
  - *Very* easy to learn and use (in direct contrast to C++)
  - Effective for an *extremely* broad set of tasks
  - Supports broad set of programming paradigms
- Yes, it is actually named after the comedy group
  - References to Spam are highly encouraged

Why Am I Doing This?

- I want to share things I've learned
- I hope this information can be useful to others
- I think it is good for my own professional development

- I like to hear myself talk

Why Should You Care?

- Increasing demand for computer proficient engineers
- You won't see much of this in class
- Start to feel like you can *actually do stuff*
- Useful in *every technical discipline*
- Suprisingly easy to impress people that don't know better (e.g. your boss)

Why Should You Care?

- Python is NOTHING like C++
  - Get cool stuff working quickly
  - Useful datastructures built in
  - Syntax is much simpler
  - No pointers!
  - Error messages are **far** more helpful
  - Famous for it's simplicity (in direct contrast to C++)

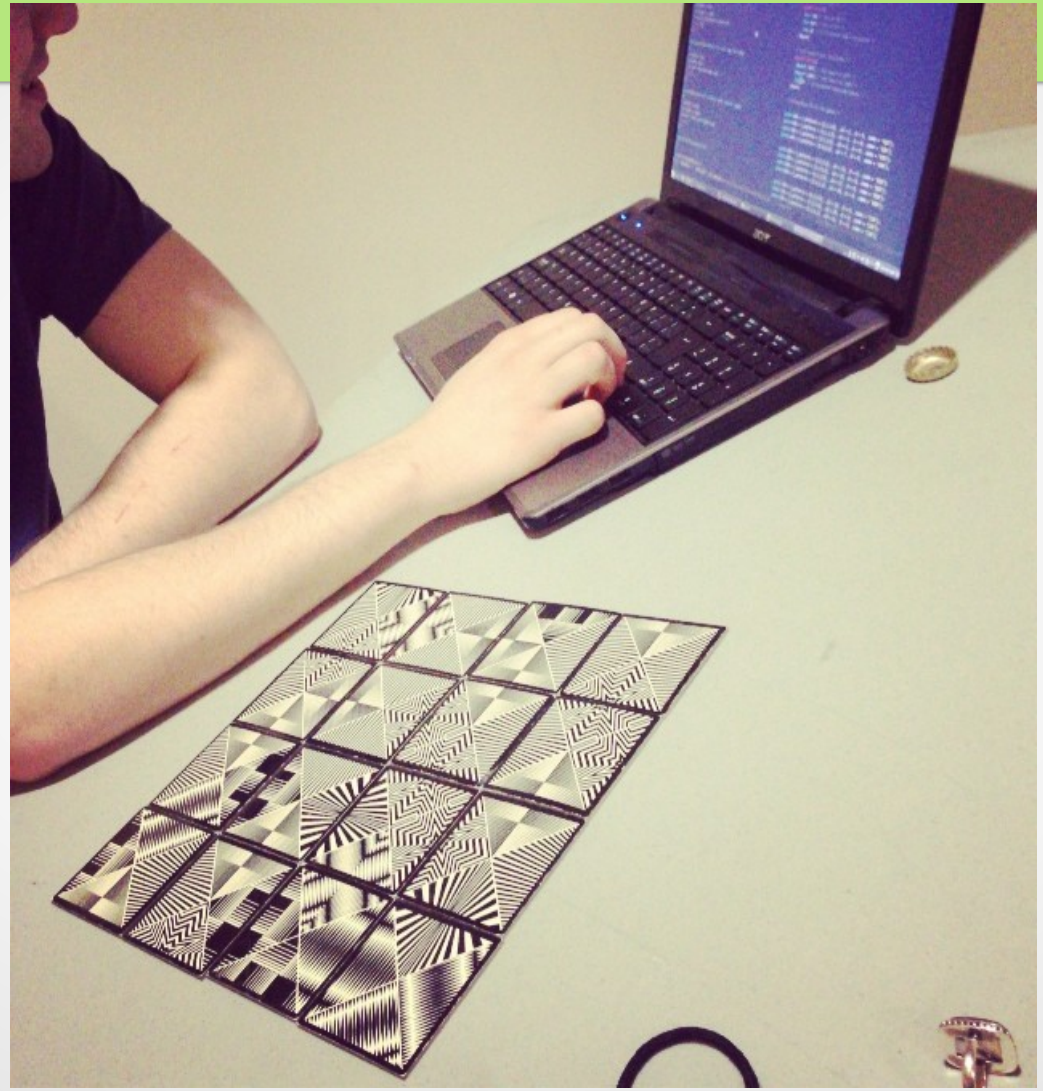
Why Should You Care?

- The most useful thing I have ever learned
  - Boss at SaskPower amazed at what I could do
  - Offered full time position at SaskPower
  - Used extensively throughout my capstone project
  - Helped a lot throughout my classes
- Exposure to broad application areas
- You can add a ton of buzzwords to your resume

"In Practice"

- Example: Model or estimate the spread of disease
- Example: Statistical quality control of a complex manufacturing or chemical process
- Example: Predict future load on SaskTel's network
- Example: Analyze the stock market
- Example: Model traffic flow/congestion
- Example: Visualize a city's demographics
- Example: Interact automatically with a webservice

Fun Stuff



What is a Programming Language

- Formal language for controlling a computer
- Used to express algorithms in a computer-readable format
- Examples: C, FORTRAN, Python, Ruby, Haskell, LISP, etc...

Final Product, Design Tool, Scripting

- Code as a Final Product (I'm not focusing on this)
  - Web browser, operating system, facebook, word processor…
  - Control system for chemical production, network router
- Programming as a Design Tool
  - Design an electronic filter, model a car's suspension
  - Fit data to a model, work symbolically with complicated equations
- Use as a Scripting Tool
  - Automate mundane spreadsheet tasks, interface automatically with a web service
  - Find / replace over thousands of files

Scripting

- Python is an effective scripting language
- Useful for writing small applications to help you with simple, repetitive, or error prone tasks
- Many applications (e.g. MS Office, AutoCAD) have scripting interfaces – learning Python is a good start to understanding them in general

Examples From Term at SaskPower

- Take an excel sheet with (many) legal land locations (e.g. SE-12-20-33-W1) and convert them to GPS coordinates with http://prairielocator.com/ to import into Google Earth.
- Used AutoCAD's scripting interface to search thousands of drawings for a reference to a specific site, move the file into a folder, and print. Resulted in 200+ printed drawings.
- Using MS office scripting interface, did a find/replace over 300+ word files for an erroneous email address.

Examples From Term at SaskPower

- Scripting Geographic Information Systems (GIS)
- At SaskPower, I had to pull every single transmission line out of a slow GIS database by hand, convert it to GPS coordinates, and load it into GoogleEarth
- This was a painful, error prone task
- ArcGIS for example, supports Python scripting
- See: Python Scripting for ArcGIS, Paul A. Zandbergen

What Makes Python Easy?

```
#One line hello world
print 'Hello, world!'

#Lists can contain anything
#[string, int, float, functions, other lists, any object really
my_list = ["I'm a string!", 4, range, ['nested list'], {'dict': 2}]

#Easy to iterate through objects in a list
for item in my_list:
    print item

#Built in hash tables
Ryan = {'Age': 22, 'Height': 182}
print Ryan['Age']

#-----More Complex------

#"List comprehensions"
#[1, 4, 9, 16]
other_list = [i**2 for i in range(5)]
print other_list

#Functional programming
#[1, 2, 3, 4]
final_list = map(lambda x: x**0.5, other_list)
print final_list
```

Alternatives to Python (and why Python is better)

- Ruby
  - Similar to Python in many ways.
  - Smaller ecosystem for number crunching and scientific tasks
  - Claim to fame is RoR
- MATLAB
  - Non-existant ecosystem for non-scientific tasks
  - Python is as good for many scientific tasks, both use the same backend
  - Expensive and proprietary

Alternatives to Python (and why Python is better)

- Excel
  - Very limited capability
  - Slooooowwwww
  - Expensive
- Others (e.g. Javascript, C#, Java, C++) suffer from some combination of
  - Complicated
  - Immature
  - Limited scope / ecosystem

Python Interpreter

- Lines of code are read, decoded, and executed by an *interpreter* line by line
- Possible to experiment and play with ideas interactively
  - Improves development time, debugging is far easier
- Genearlly slower than compiled code
  - Bindings can alleviate this problem in many cases
  - Programmer time is far more valuable than computer time

IPython Interpreter

- Ipython is a “souped up” Python interpreter
- Deeper introspection
- Built in documentation e.g.

```
In [4]: from numpy import array

In [5]: array?
Type:       builtin_function_or_method
String Form:<built-in function array>
Docstring:
array(object, dtype=None, copy=True, order=None, subok=False, ndmin=0)

Create an array.

Parameters
----------
object : array_like
    An array, any object exposing the array interface, an
    object whose __array__ method returns an array, or any
    (nested) sequence.
dtype : data-type, optional
    The desired data-type for the array.  If not given, then
```

Number Crunching

- Curve fitting, differential equations, optimization, numerical integration, modeling etc…
- Python libraries Numpy and Scipy interface with…
  - BLAS: Free-software FORTRAN code carefully tuned for basic linear algebra operations
  - LAPACK: Similar to BLAS, but higher level operations (e.g. SVD, QR)
  - The same backend (and hence very similar performance) to MATLAB
- Don't use "for" loops for numerics! Use built in matrix / vector operations

Number Crunching

- The magic lies in storing numbers in efficient numpy arrays

```
In [5]: t = linspace(0, 1) #Break "time" into small chunks in computer memory

In [6]: print t
[ 0.          0.02040816  0.04081633  0.06122449  0.08163265  0.10204082
  0.12244898  0.14285714  0.16326531  0.18367347  0.20408163  0.2244898
  0.24489796  0.26530612  0.28571429  0.30612245  0.32653061  0.34693878
  0.36734694  0.3877551   0.40816327  0.42857143  0.44897959  0.46938776
  0.48979592  0.51020408  0.53061224  0.55102041  0.57142857  0.59183673
  0.6122449   0.63265306  0.65306122  0.67346939  0.69387755  0.71428571
  0.73469388  0.75510204  0.7755102   0.79591837  0.81632653  0.83673469
  0.85714286  0.87755102  0.89795918  0.91836735  0.93877551  0.95918367
  0.97959184  1.        ]

In [7]:

In [7]: x = cos(2*pi*t) #A cosine wave over time

In [8]: print x
[ 1.          0.99179001  0.96729486  0.92691676  0.8713187   0.80141362
  0.71834935  0.6234898   0.51839257  0.40478334  0.28452759  0.1595999
  0.03205158 -0.09602303 -0.22252093 -0.34536505 -0.46253829 -0.57211666
 -0.67230089 -0.76144596 -0.8380881  -0.90096887 -0.94905575 -0.98155916
 -0.99794539 -0.99794539 -0.98155916 -0.94905575 -0.90096887 -0.8380881
 -0.76144596 -0.67230089 -0.57211666 -0.46253829 -0.34536505 -0.22252093
 -0.09602303  0.03205158  0.1595999   0.28452759  0.40478334  0.51839257
  0.6234898   0.71834935  0.80141362  0.8713187   0.92691676  0.96729486
  0.99179001  1.        ]
```

Optimization

- In math class you deal with trivial optimization problems
  - This is because you have to solve them by hand e.g.

$$\min_x f(x) = 2x^2 - 3x + 1$$

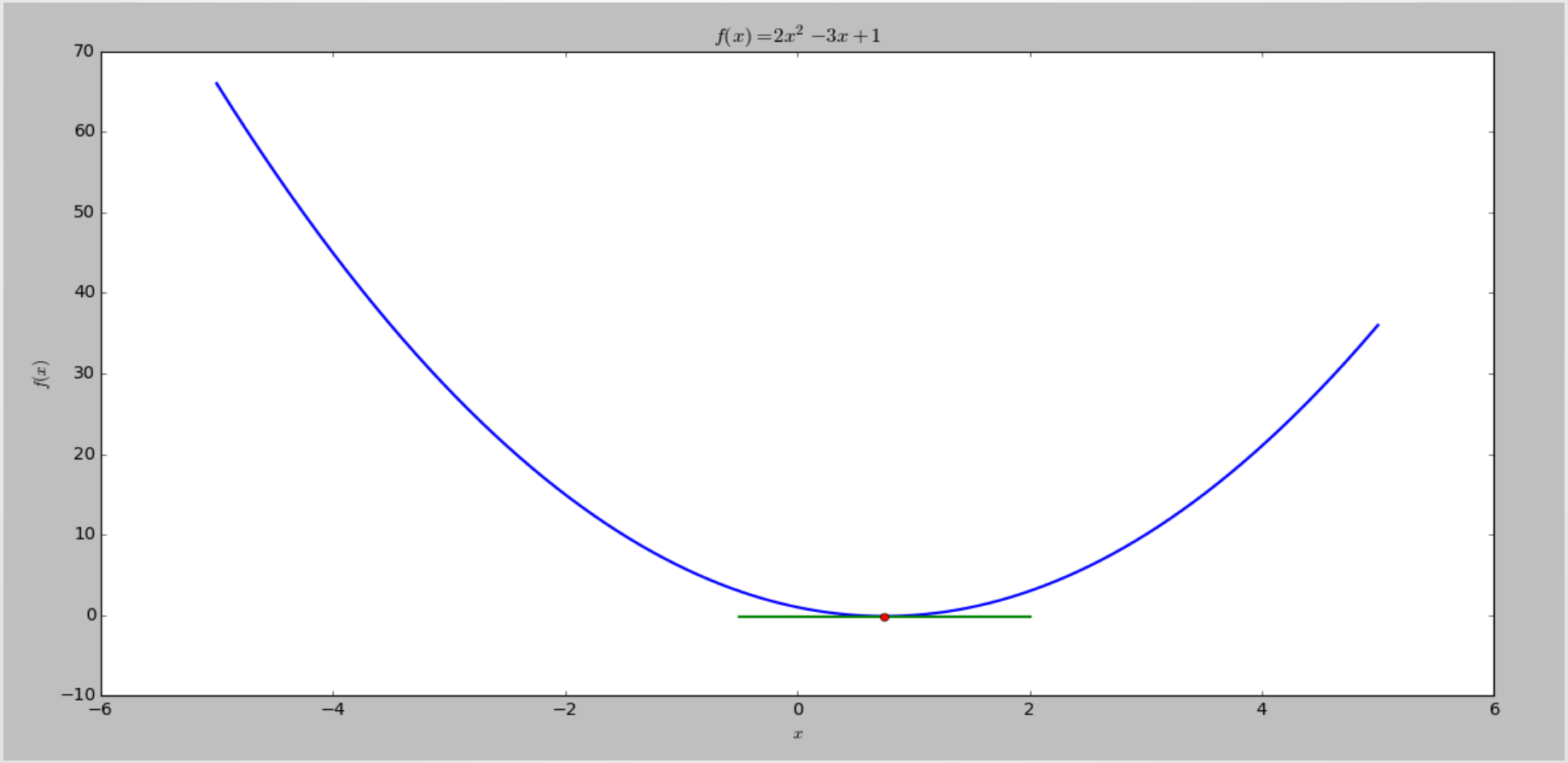
Simply differentiate and find the roots of the resulting equation...

$$\frac{d}{dx} f(x) = 4x - 3 = 0$$

$$x = \frac{3}{4}$$

Optimization

- Here is a plot of the previous problem and the optimum point



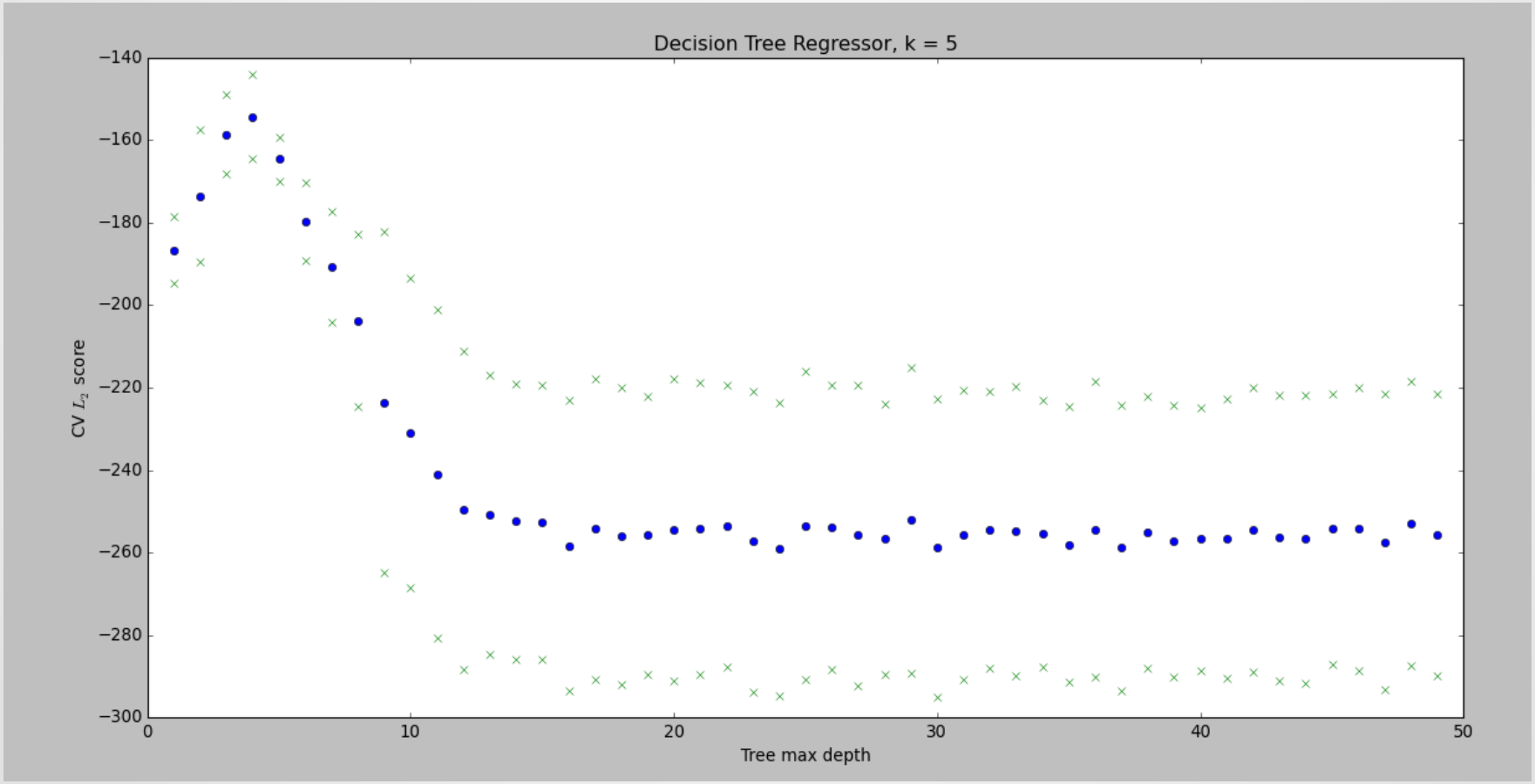
Real Optimization Problems

- Predict load on a power network and use this prediction to schedule generators and transmission lines
- Find an investment portfolio with the maximum expected return and minimum variance
- Efficiently schedule jobs onto computer nodes
- Efficiently schedule work crews on a large job site
- Fitting a ton of data to some model (machine learning)
- Design high frequency electronic filters (Part of my NSERC project)

Example: Predicting Titanic survival

- Competition problem from www.kaggle.com
  - Some problems here worth $100,000, some people solve them with Python. E.g. Detecting diabetic retinopathy
  - Possible source of a really cool 4th year project!
- Given a dataset of titanic passengers including their age, gender, ticket price, name, and whether or not they survived.
- Create a model from this data to predict who would have survived out of a new dataset.

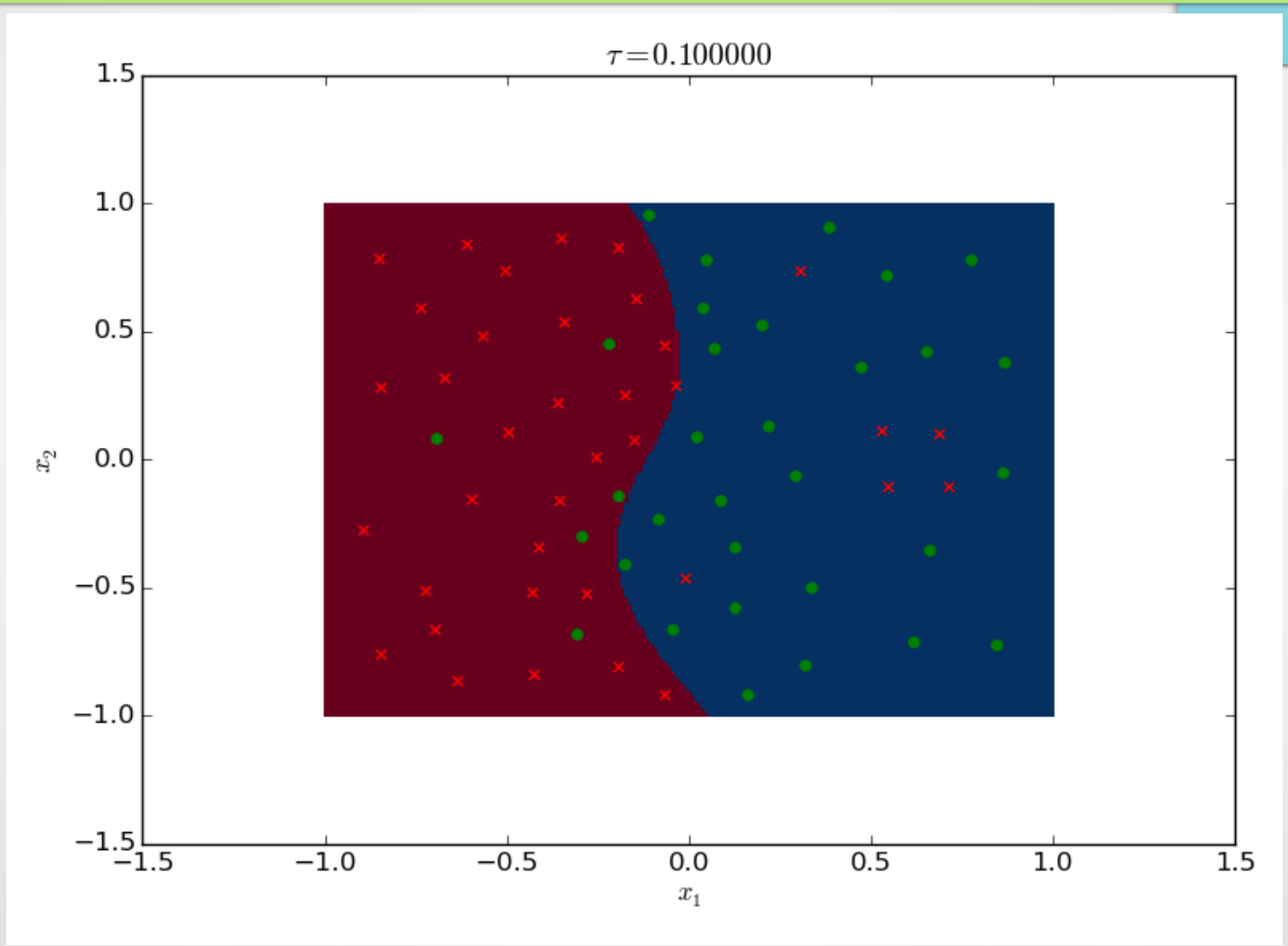
Predicting Titanic survival



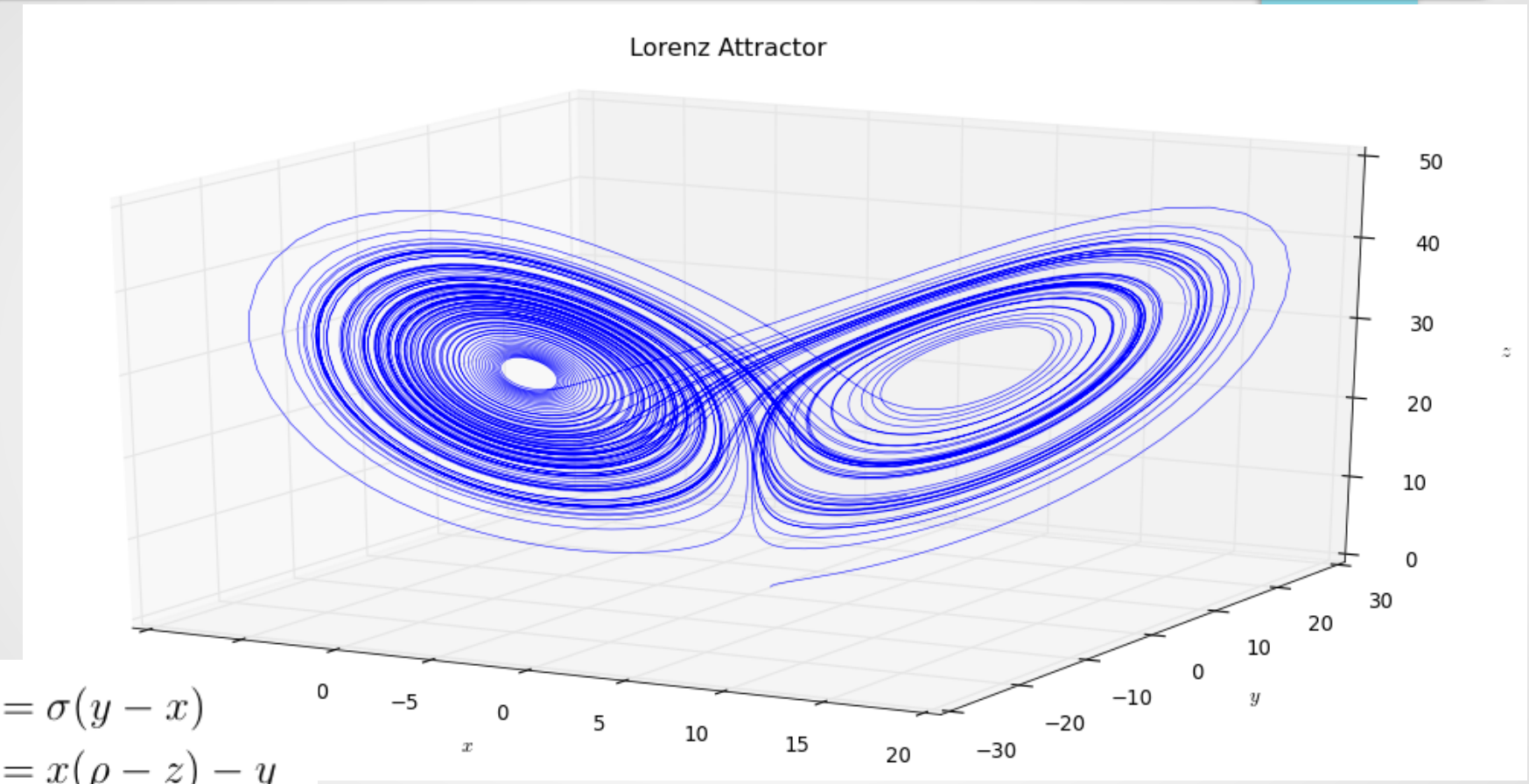
Plotting

- MatplotLib provides great ways to visualize data and plot functions
  - It is again similar to MATLAB in many ways
- Example: Plot the decision boundary for a binary classifier.
- Example: Plot some particular solution for a differential equation

Example: Binary Classifier



Example: Lorenz Attractor

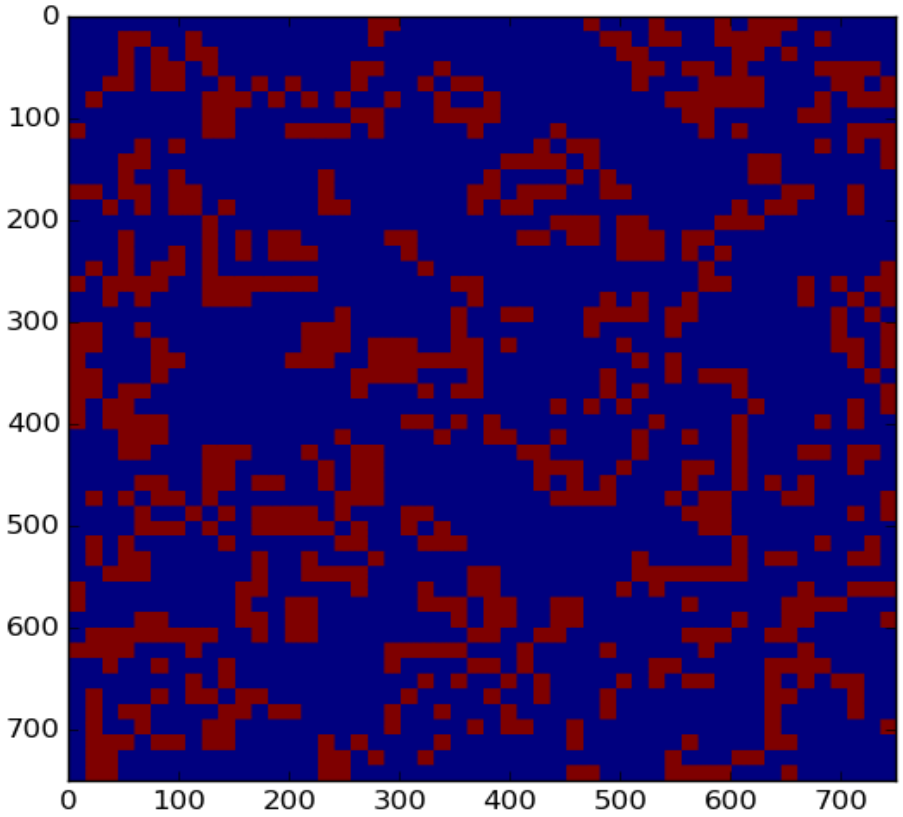


$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = x(\rho - z) - y$$
$$\dot{z} = xy - \beta z$$

Example: Lorenz Attractor code

```
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

import numpy as np
from scipy.integrate import odeint

def lorenz(x, t0, s = 10, r = 28, b = 2.667):
  '''
  Lorenz system gradient
  '''
  x0_dot = s*(x[1] - x[0])
  x1_dot = r*x[0] - x[1] - x[0]*x[2]
  x2_dot = x[0]*x[1] - b*x[2]
  return x0_dot, x1_dot, x2_dot

x0 = np.array([1.0, 0.2, 0.1])
t = np.linspace(0, 100, 10001)
y = odeint(lorenz, x0, t)

fig = plt.figure()
ax = fig.gca(projection = '3d')

ax.plot(y[:, 0], y[:, 1], y[:, 2], linewidth = 0.5)
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
ax.set_zlabel('$z$')
ax.set_title('Lorenz Attractor')

plt.show()
```

Example: Game of Life



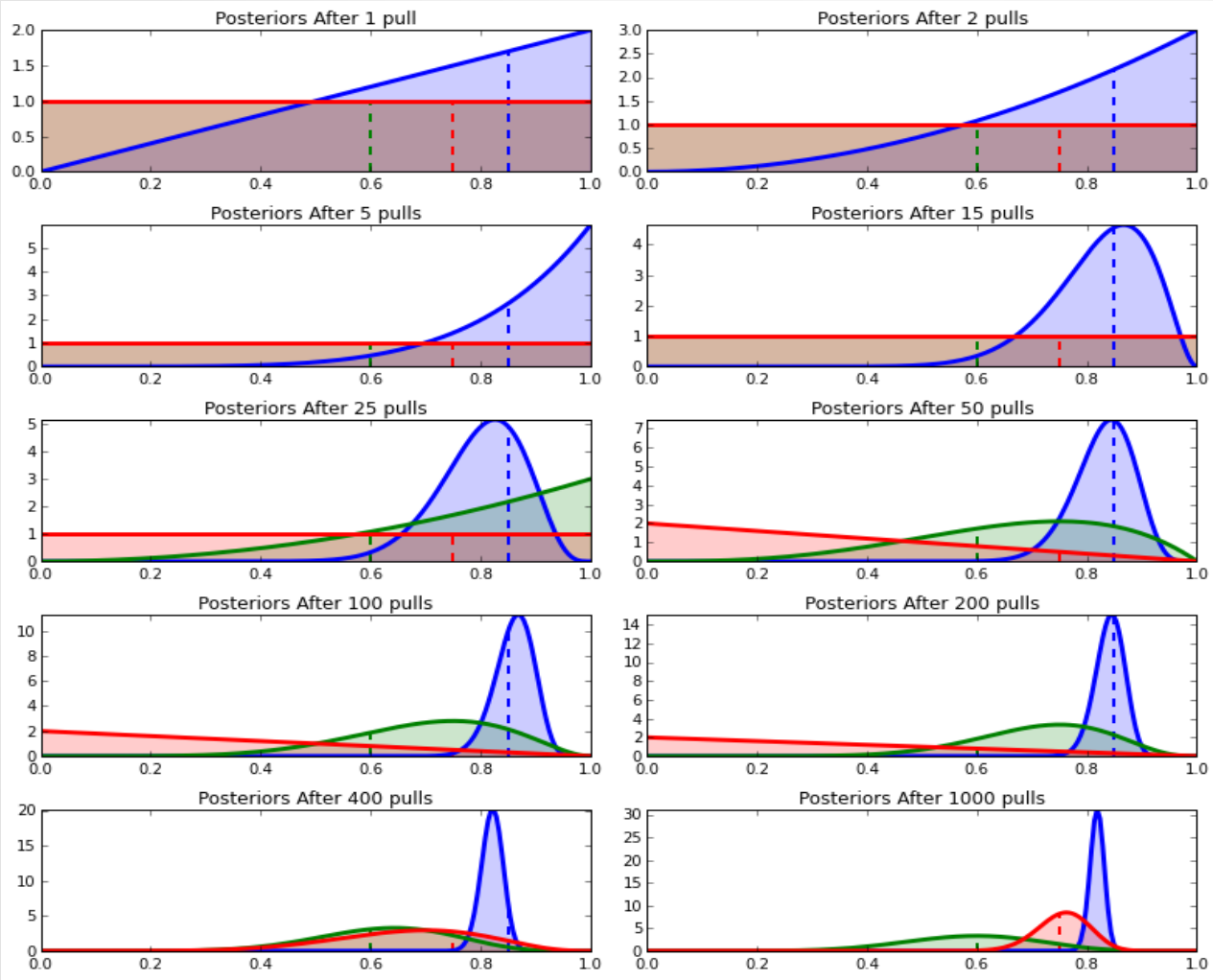
Example: Multi Armed Bandits

- You are in a casino in a room with a bunch of one armed bandits (slot machines)
- Each one has a different probability of paying out
- Your goal is to make money as quickly as possible
- You need to explore the machines to find the better ones, but also exploit the ones you know are good
- Very hard, and very practical problem

Multi Armed Bandit Applications

- Clinical trials: Find the best treatment while letting few people perish
- Advertising: Find the most effective ad, while still using ones that work
- Networks: Find the best network route or channel while still getting your data through
- Finding a mate: Find the best boyfriend/girlfriend while …

Solution in Python "Bayesian Bandits''



From the freely available book – **'Bayesian Methods for Hackers'** by Cameron Davidson-Pilon

Symbolic Math

- SymPy – a computer algebra system for Python (similar to Maple)
- Very complex equations can be dealt with in closed form
- Trigonometry, calculus, special functions, differential equations, combinatorics… all in closed form
- e.g. This integral was evaluated via Sympy and printed to LaTeX

$$\int x(ax+b)^{\alpha}\,\mathrm{d}x = \frac{(ax+b)^{\alpha}\left(a^2\alpha x^2 + a^2x^2 + a\alpha bx - b^2\right)}{a^2\left(\alpha^2+3\alpha+2\right)}$$

Examples

- Taylor Series

```
In [26]: expr = exp(x)*cos(2*pi*log(x))

In [27]: pprint(expr)
 x
e ·cos(2·π·log(x))

In [28]: series = expr.series(x, 1, 3) #First 3 terms of Taylor series around x = 1

In [29]: pprint(series)
                          2 ⎛         2   e⎞    ⎛       3        ⎞
e + e·(x - 1) + (x - 1) ·⎜- 2·e·π  + ─⎟ + O⎝(x - 1) ; x → 1⎠
                            ⎝              2⎠
```

- Limits

```
In [61]: pprint(expr)
⎛    π⎞
⎜x - ─⎟·tan(x)
⎝    2⎠

In [62]: expr.limit(x, pi/2, '+')
Out[62]: -1

In [63]: expr.limit(x, pi/2, '-')
Out[63]: -1
```

- Differential Equations

```
In [237]: eq = Eq(fpp + 2*zeta*w0*fp + w0**2*f - 3) #Harmonic oscillator

In [238]: pprint(dsolve(eq, f))
```

$$f(x) = C_1 \cdot e^{x\cdot\left(-\omega_0\cdot\zeta - \sqrt{\zeta^2 - 1}\cdot|\omega_0|\right)} + C_2 \cdot e^{x\cdot\left(-\omega_0\cdot\zeta + \sqrt{\zeta^2 - 1}\cdot|\omega_0|\right)} + \frac{3}{\omega_0^2}$$

Other Stuff

- Pandas: Library for efficiently working with labeled and time series data (e.g. financial data, data on individuals, sales over time …)
- Sklearn: Library implementing large number of high quality mahine learning algorithms (regression/classificiation, clustering, visualization, etc…)
- Skimage: Library for image processing
- Scipy.signal: Various signal processing methods (e.g. filter design, LTI systems). I used this extensively for my fourth year project

Other Stuff

- Django: Popular web framework similar to Ruby on Rails
- Theano: GPU computing, C code generation
- PyQt: Develop GUIs
- Cython: Write C extensions and bindings
- BeautifulSoup: HTML parsing
- Scrapy: Web crawling
- PyMC: Markov Chain Monte Carlo (MCMC) methods

Possible Thoughts on Your Mind

- "This stuff is not useful, I never see problems where I would need to make use of any of this stuff "
  - I would suggest that this is because the people who assign you problems assume you aren't aware of any of these tools
- "Using this is probably much harder than you make it seem…"
  - You are right, it can be challenging. But, who will pay you to do stuff that is easy?

Possible Thoughts on Your Mind

- "Am I ever going to use this?"
  - Maybe, maybe not. But, it's another skill to set you apart from others in a competitive job market.
- "I don't have time to learn this stuff…"
  - Yes you do.
- "I hated my programming classes, programming *#@$^& sucks!"
  - Python is 100x easier than C++, and you can get exciting results 100x faster

Possible Thoughts on Your Mind

- “This is so cool omg!’’
  - Yes.

How to Get Started

- Anaconda Python distribution (Windows, OSX, Linux)
- You will be well served to use a Unix-like environment
  - Mac OS
  - Linux Virtual Machine inside Windows
  - Dual boot Linux - Ubuntu and Linux Mint are very user-friendly
  - Not strictly necessary, everything still works on Windows
- Unix was designed for programmers by programmers, it should make your life easier in the long run (my opinion...)

Summary

- Python is a phenomenal tool with a huge ecosystem of useful libraries
- Excellent starting place to learn about scientific computing and scripting languages
- There are specialized tools that are better than Python for specialized tasks – but learning Python gives you a tool that is still perfectly capable for most tasks
- Very easy, and very powerful – especially as a student
- Skills will carry over to other languages / environments

How to Get Started

- Huge amount of resources on the web
- **docs.python.org/2/tutorial/introduction.html (Informal intro to Python)**
- Anaconda scientific python distribution
- Google search "SciPy''
- Check out www.kaggle.com
- Solve some problems on www.projecteuler.net
- Email me: Ryan@Kinnear.ca (seriously)
- QUESTIONS?