Machine Learning MIPT
Introduction (seminar 1)

Ashuha Arseniy, Alexey Dral

ars.ashuha@gmail.com

February 9, 2016

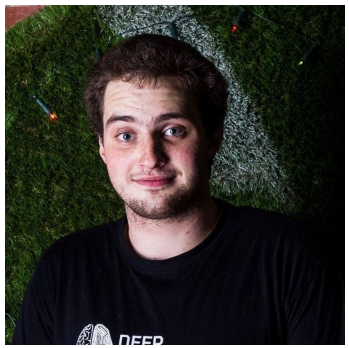
Plan

1. Introduction
   - Who we are?
   - What Machine Learning is?
2. How to pass the course?
3. Optimization, Linear algebra and Probability and review
4. Data scientist tools (python and numpy review)

Introduction

Course authors

- Arseniy Ashuha



Master student at MIPT, Data scientist at Rambler&CO

- Supervised by: Alexey Dral

Assistant Prof. MIPT, Amazon

Machine learning in general

We have a visible varibles X and we want to predict a hiden varibles T.

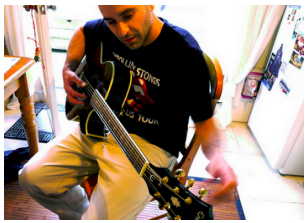
- Classification

 cat ? dog

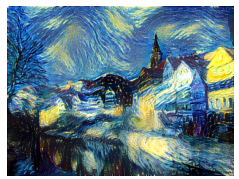
- Regression

 76.23

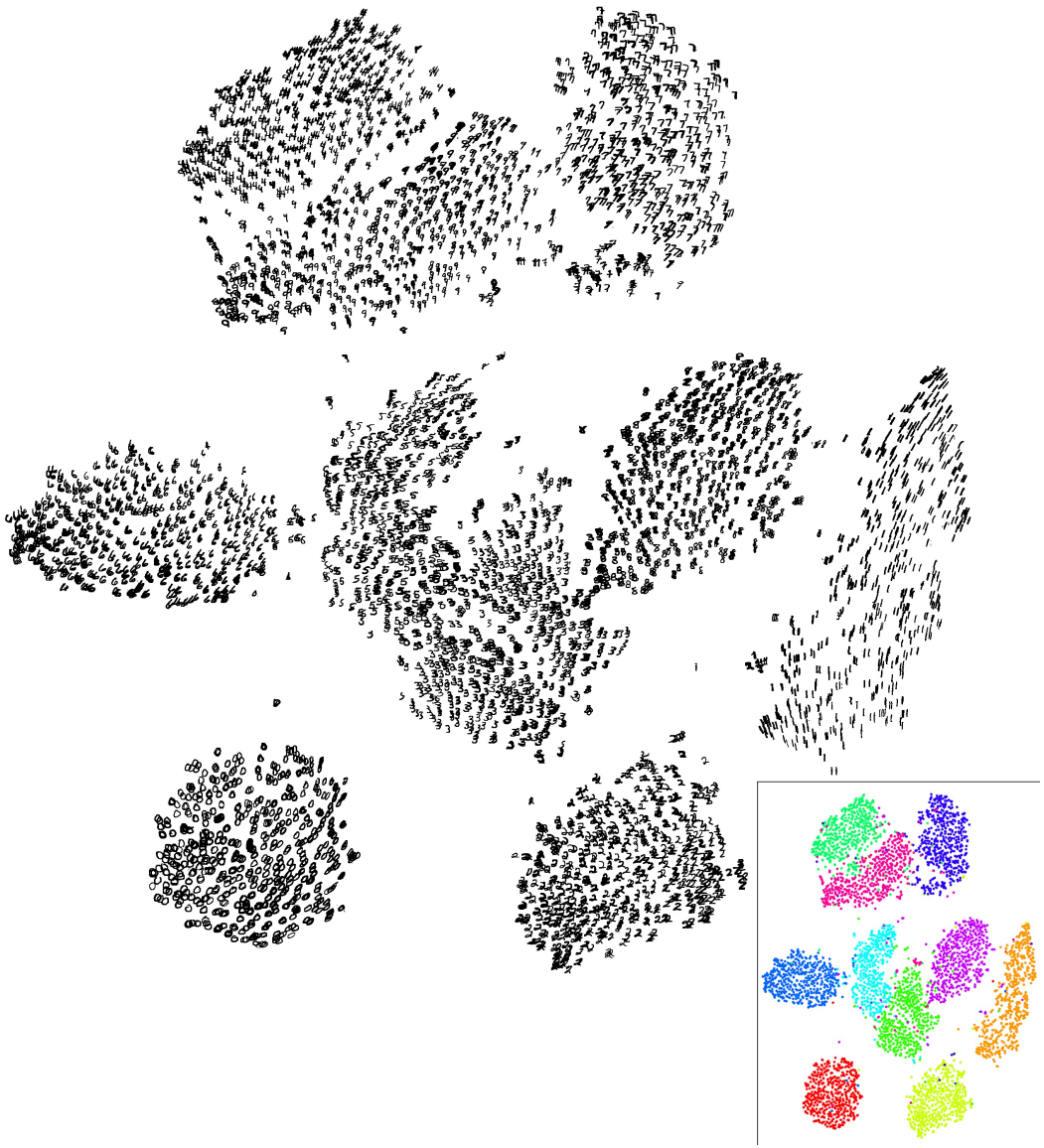
- Phrase prediction

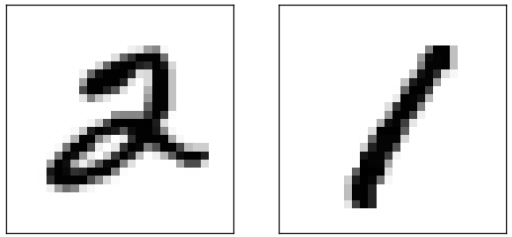
 the guy sitting on chair and tune his guitar

- Some art

( , ) want to generate 

Supervised / Unupervised





How to pass the course?

What will you learn after course

- How to look for ml problem in real live
- How to solve ml problem
  - What to do if train set is absent?
  - What to do if you have small/much data?
  - How to estimate quality of your model
- How to code it by yourself

How to pass the course?

- Practical HW (2 credits per HW) :
  Bonus:
  - 1.0, 0.75, 0.5 for 1-3 positions
  - -0.2 each day after deadline, but not more -1.0

  1. KNN picture classification code yourself
  2. Trees, Forest salary prediction code yourself
  3. Linear, Dimension reduction picture classification part code yourself
  4. All Methods age prediction
- Practice class works (only bonus credits ☺):
  1. Bayesian classification
  2. Neural nets
- Theory credit (2 credit in time, 1 after)

$$Result = sum \cdot [Theory\ credit \geq 1.0]$$

course wiki: MIPT ML 2016 Spring

Linear algebra, probability and optimization review

Optimization

1. Usually we want to optimize function L – how lot error algorithm do

$$L(w, data) \to \min_w$$

2. Optimize it by gradient desent

   chose initial approximation of w *and make small step by* $-grad(w)$

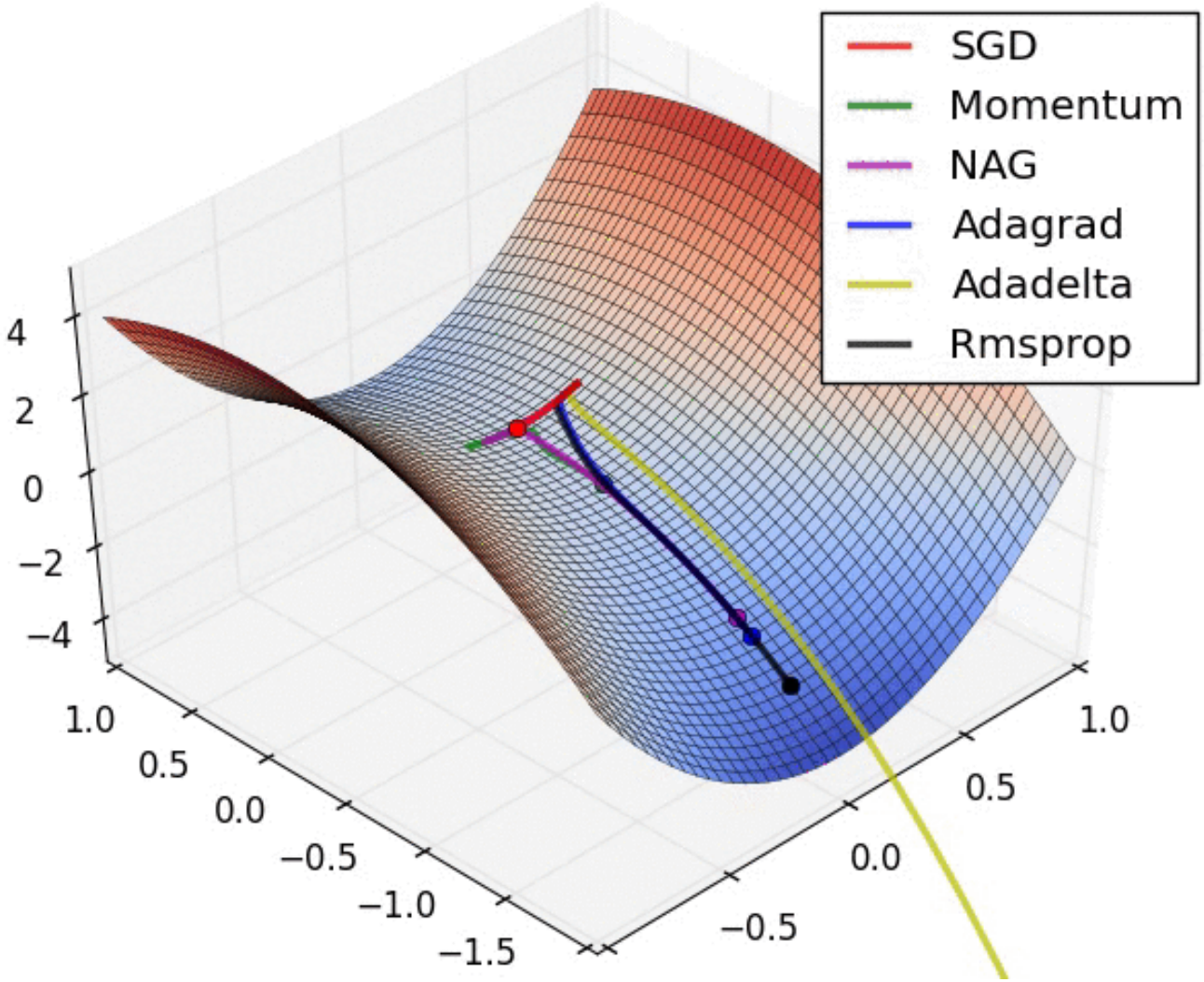
3. Or stochastic gradient desent

$$grad(L) = \frac{d}{dw} L(w, data) \approx \frac{d}{dw} L(w, random(X))$$

4. Linear restriction r(w) e.g $w_i \geq 0$, $\sum w_i = 1$ – w is equal probability for example

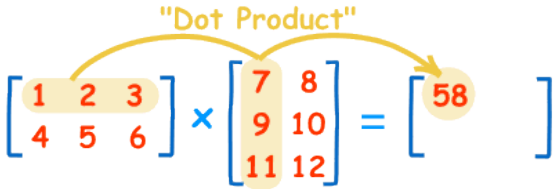
$$L(w, data) + \sum \lambda_i \cdot w_i + m \cdot (\sum w_i - 1) \to \min_w$$

Optimization



Linear algebra

- Sometimes we can express our algorithm in term of the Matrix (Tensor) multiplication – n^3 very fast operation ☺



- T – train matrix, $\hat{T}$ – predict matrix, we can use $norms(T - \hat{T})$

$$||A||_F = \sqrt{\sum_i \sum_j A_{ij}^2}$$

- If Det = 0 a lot of LA algorithms work poor

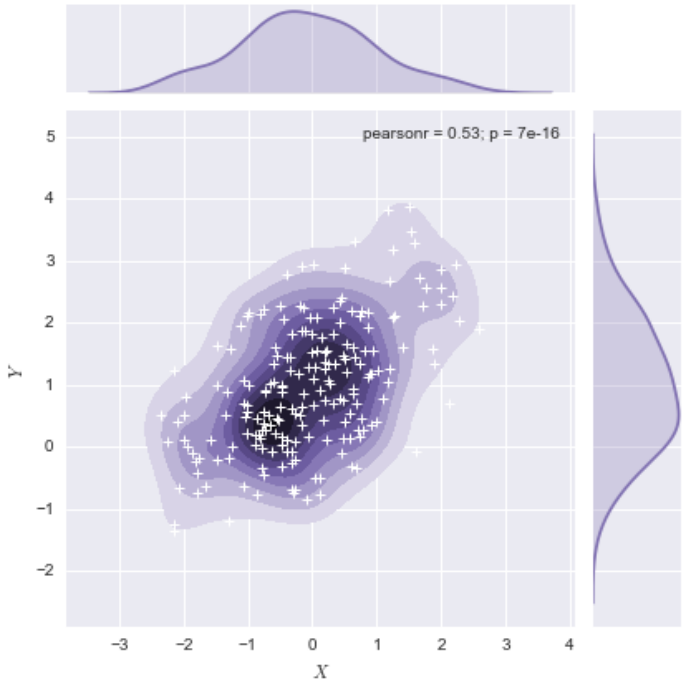
$$det \begin{bmatrix} 1 & 2 \\ 1 & 2 + 1e^{-7} \end{bmatrix} \approx 0.000001 \quad cond \begin{bmatrix} 1 & 2 \\ 1 & 2 + 1e^{-7} \end{bmatrix} \approx 10000004$$

- Matrix calculus

$$L(W) = W_{11}^2 + W_{22} \qquad \frac{d}{dW} L = \begin{bmatrix} 2 \cdot W_{11} & 0 \\ 0 & 1 \end{bmatrix}$$

Probability

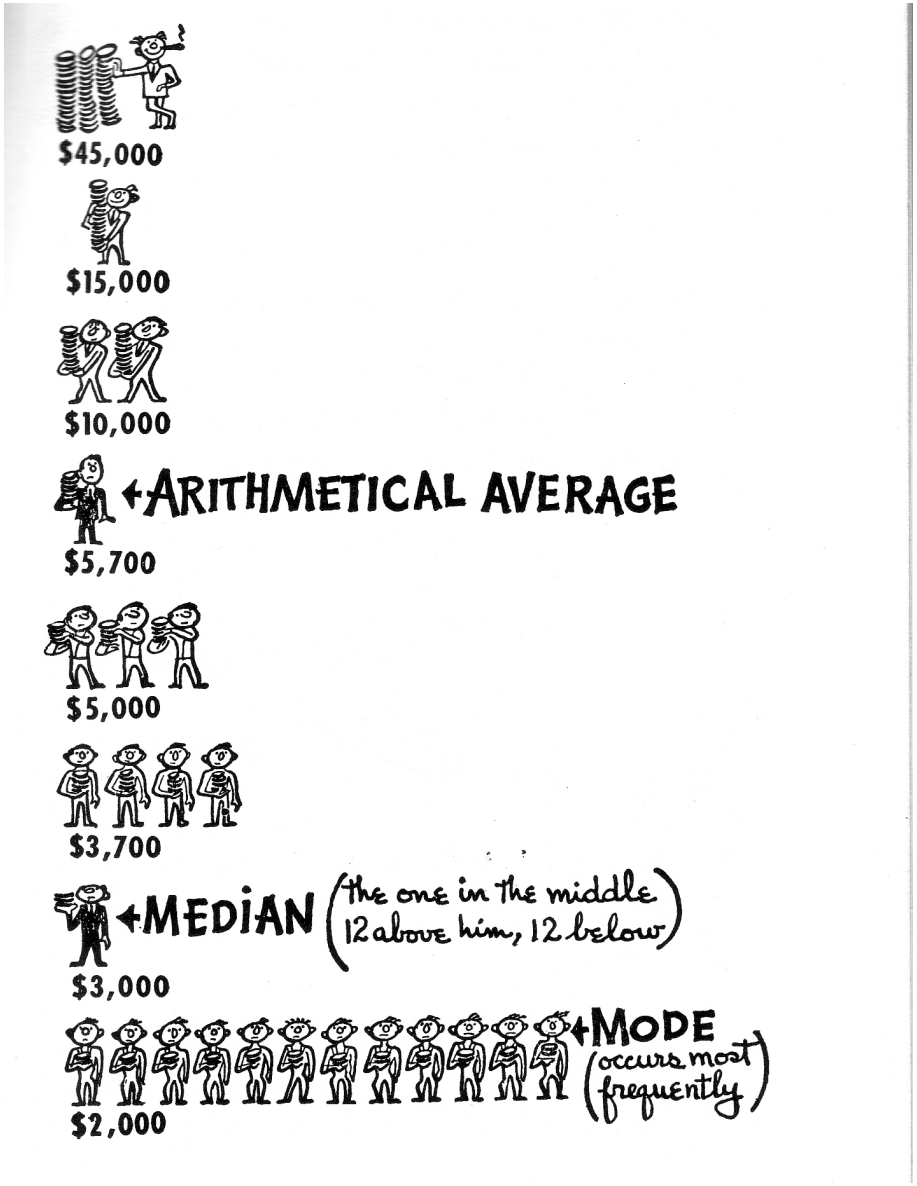
- Probability Mass Function + Conditional distribution + Correlation



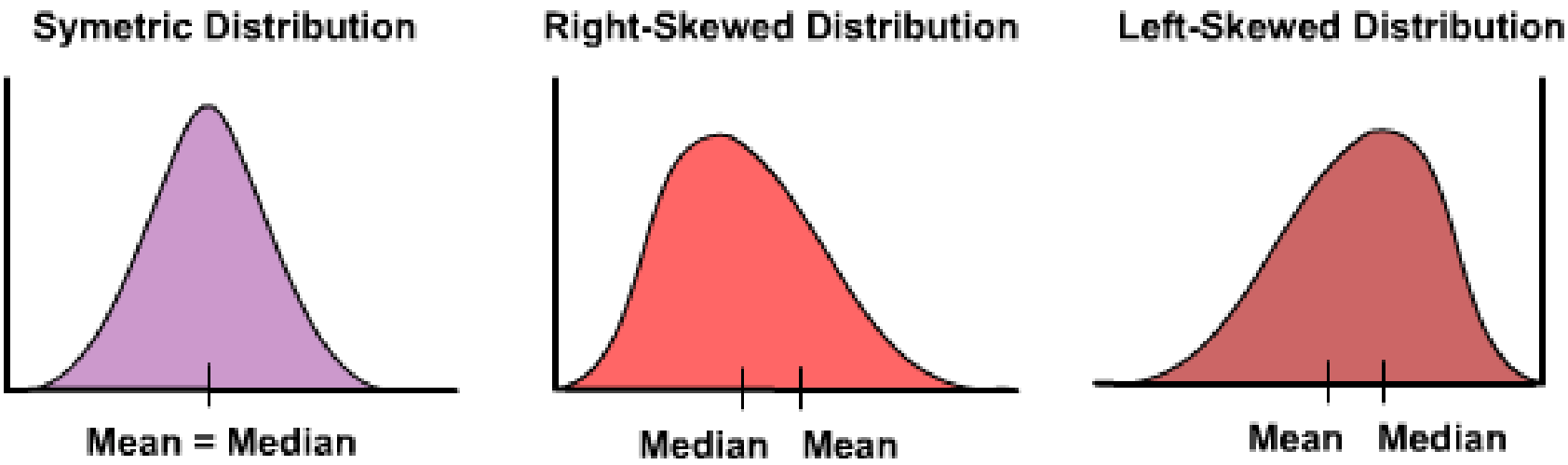
- Bayesian rule, $p(T, X)$ – best approximater, but we cant find it general

$$P(X|T) = \frac{P(T|X)P(X)}{P(T)}$$

Probability: Means



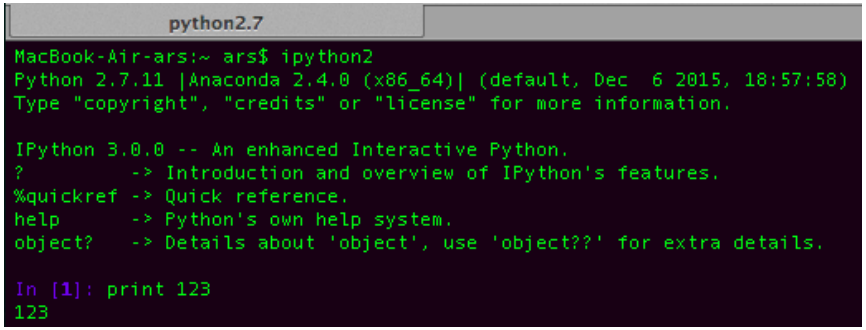
Probability: Means



Data scientist tools

Python

1. Python – интерпретируемый язык, для быстрой разработки.
2. IPython



3. Ipython Notebook + PyCharm + Anaconda

Python: Useful packages

1. numpy/scipy – эффективная реализация массивов
2. pandas – работа с данными: чтение, join, ...
3. matplotlib/seaborn – мощная библиотека для графиков.

Настоятельные рекомендации по изучению Python stackoverflow

1. google-styleguide.googlecode.com/svn/trunk/pyguide.html
2. www.tutorialspoint.com/python/python_quick_guide.htm
3. O'REALLY Learning Python, 5th Edition

Solve ML Problem Stages

- Understand, what we want and how ml can help us – use your brain
- Get data – use your brain
- Read data – pandas
- Visualise and clean data – mathplotlib, pandas, numpy
- Build model – ???
- Estimate quality – ???

Pandas

```
In [1]: import pandas as pd
```

```
In [2]: df = pd.read_csv('sal.csv')
```

```
In [3]: df
```

Out[3]:

	id	sex	salary	education
0	1	m	10000	high
1	2	m	5000	mid
2	3	f	7000	high
3	4	m	1000	none

```
In [4]: df.salary.mean()
```

```
Out[4]: 5750.0
```

```
In [12]: df.query('salary>=7000')
```

Out[12]:

	id	sex	salary	education
0	1	m	10000	high
2	3	m	7000	high

```
In [37]: df.sex = df.sex.apply(lambda x: 0 if x == 'm' else 1)
         df.education = df.education.apply(
             lambda x: 1 if x == 'high' else 0)
```

```
In [41]: df
```

Out[41]:

	id	sex	salary	education
0	1	0	10000	1
1	2	0	5000	0
2	3	1	7000	1
3	4	0	1000	0

```
In [38]: X = df[[u'sex', u'education']]
         T = df[u'salary']
```

```
In [39]: X.values
```

```
Out[39]: array([[0, 1],
                [0, 0],
                [1, 1],
                [0, 0]])
```

```
In [40]: T.values
```

```
Out[40]: array([10000,  5000,  7000,  1000])
```

Links: Pandas in 10 minurtes, prof. Dyakonov materials: pandas tutorial

Python: Numpy is fast

```
In [1]: import numpy as np

In [2]: A, B = np.random.random((1000, 1000)), np.random.random((1000, 1000))

In [3]: %time C = A.dot(B)
CPU times: user 129 ms, sys: 5.8 ms, total: 135 ms
Wall time: 70.4 ms

In [4]: %time C = [[sum([A[i, j]*B[j, k] for j in xrange(1000)])
for i in xrange(1000)] for k in xrange(1000)]
CPU times: user 12min 28s, sys: 961 ms, total: 12min 29s
Wall time: 12min 29s

In [5]: a = np.random.randint(0, 10, 100000000)

In [6]: %time a = np.argsort(a)
CPU times: user 11 s, sys: 2.86 s, total: 13.9 s
Wall time: 14.7 s
```

Links: Stanford numpy tutorial, scipy tutorial

mathplotlib

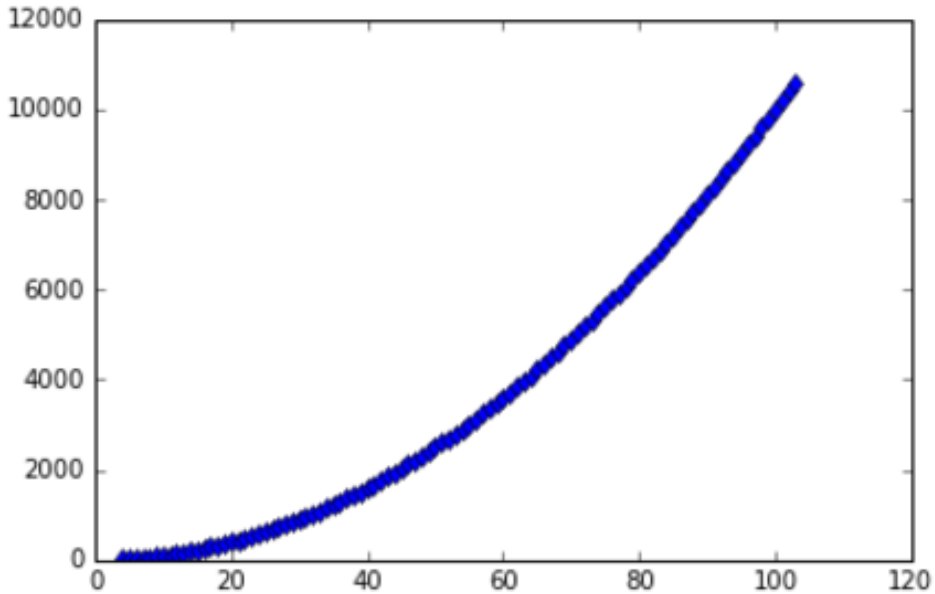
In [ ]:
```
import numpy as np
import matplotlib as pylab

%pylab inline
```

In [2]:
```
x = np.arange(0, 100) + 3*np.random.normal()
y = x**2 + 1
```

In [3]:
```
pylab.plot(x, y, 'd')
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x10468fed0>]
```



seaborn

seaborn 0.7.0 API Tutorial Gallery Site Page

Scatterplot Matrix

scipy

```
>>> from scipy.optimize import minimize, rosen, rosen_der
```

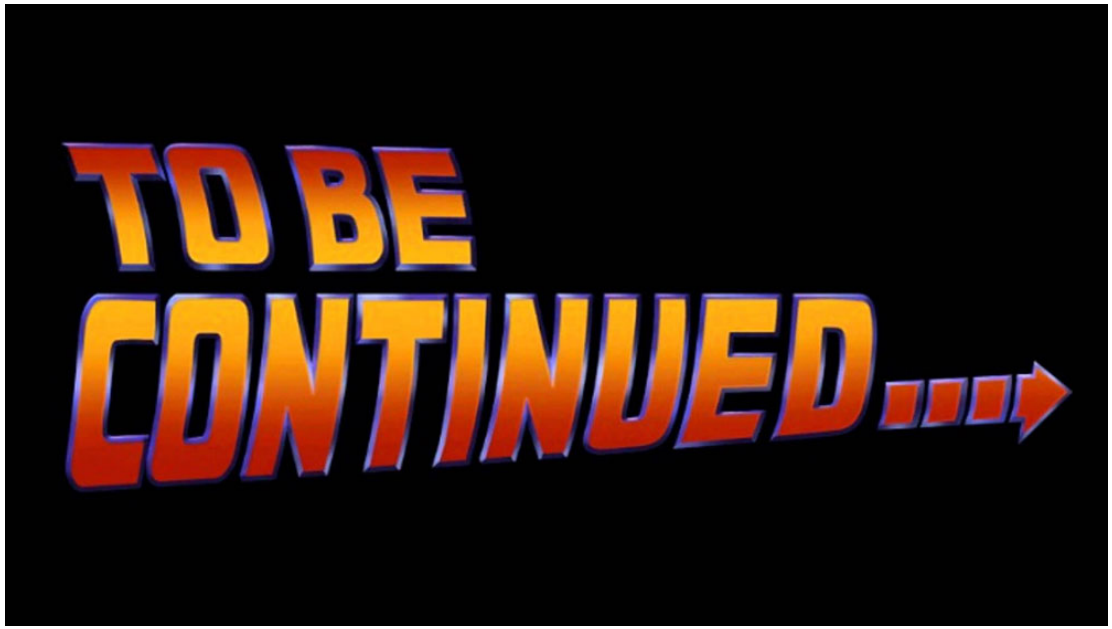
A simple application of the *Nelder-Mead* method is:

```
>>> x0 = [1.3, 0.7, 0.8, 1.9, 1.2]
>>> res = minimize(rosen, x0, method='Nelder-Mead')
>>> res.x
[ 1.  1.  1.  1.  1.]
```

Now using the *BFGS* algorithm, using the first derivative and a few options:

```
>>> res = minimize(rosen, x0, method='BFGS', jac=rosen_der,
...                options={'gtol': 1e-6, 'disp': True})
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations: 52
         Function evaluations: 64
         Gradient evaluations: 64
>>> res.x
array([ 1.  1.  1.  1.  1.])
```

learn model, estimate quality



Administrative

https://goo.gl/UvtlO9

- HW0: Install python, try libs, try wiki + piazza
- KNN HW will be announced next week