Name: Lanh Nguyen Van
Email: nguyenlanh2580@gmail.com
Phone: +84126 432 2580

REPORT: A Tutorial on Deep Learning

Part 1: Nonlinear Classifiers and The Backpropagation Algorithm

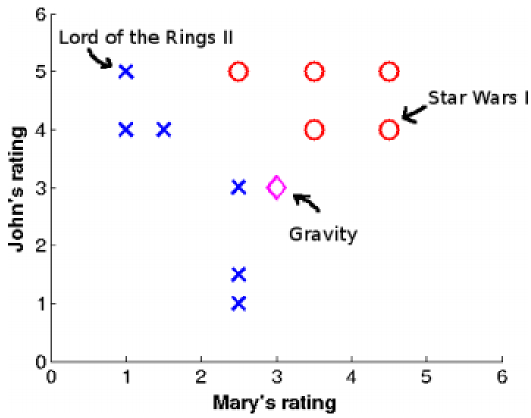
1 Problems:

The purpose of this exercise is for you to understand better the backpropagation algorithm and stochastic gradient descent algorithm. In the following, I will create a dataset similar to the one in the tutorials and a neural network of one hidden layer. Search for WORK to find the places that you're supposed to fill in the necessary code.

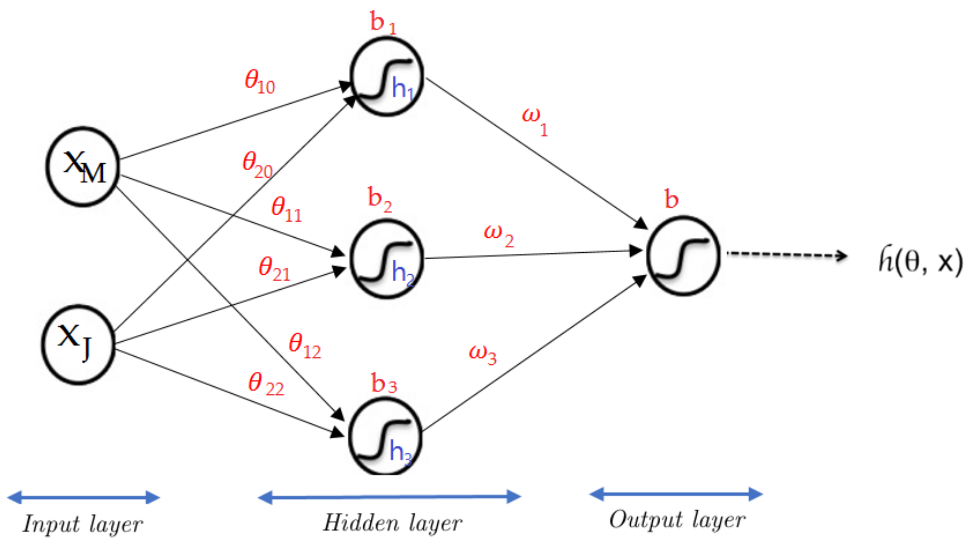
- Ideas:
  - *Input*
    - Dataset, each example is a pair of ratings:

      X = [[1, 5], [2.5, 5], [3.5, 5], [4.5, 5], [1, 4], [1.5, 4], [3.5, 4], [4.5, 4], [2.5, 3], [2.5, 1.5], [2.5, 1]]
    - Each label has a value of either 0 or 1 indicating if I will like the movie or not:

      Y = [0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0]
    - Graph:



  - *Neural Network*



We have:

$$X = \begin{pmatrix} (X_{M1}, X_{J1}) \\ \ldots\ldots \\ \ldots\ldots \\ (X_{Mn}, X_{Jn}) \end{pmatrix}$$

$$Y = \begin{pmatrix} y_0 \\ \ldots \\ \ldots \\ y_n \end{pmatrix}$$

$$W_1 = \begin{pmatrix} (\theta_{10}, \theta_{20}) \\ (\theta_{11}, \theta_{21}) \\ (\theta_{12}, \theta_{22}) \end{pmatrix}$$

$$b_1 = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$W_2 = \begin{pmatrix} (\omega_1) \\ (\omega_2) \\ (\omega_3) \end{pmatrix}$$

$$b_2 = \begin{pmatrix} b \end{pmatrix}$$

which values of θ_{ij} & ω_i are radom(). In code, we call

```
# stochastic gradient descent on dataset X, Y, with learning rate alpha
def sgd(X, Y, alpha):
  W1 = [[random(), random()],
        [random(), random()],
      [random(), random()]]
  b1 = [0, 0, 0];
  W2 = [[random(), random(), random()]]
  b2 = [0];
```

- Implement the code-behind:

– Task #1: Fill in your gradient computation here. In preferences document page 3,4, we have:
<u>**In second layer:**</u> $W_2(\omega)$

$$
\begin{aligned}
W_{2grad} = \Delta\omega &= \frac{\partial J(\omega, b)}{\partial W_2} && (1)\\
&= \frac{\partial}{\partial W_2}\left(h(h_2^{(i)};\omega, b) - y^{(i)}\right)^2 && (2)\\
&= 2\left(h(h_2^{(i)};\omega, b) - y^{(i)}\right)\frac{\partial}{\partial W_2}\left(h(h_2^{(i)};\omega, b) - y^{(i)}\right) && (3)\\
&= 2\left(g(\omega^T h_2^{(i)} + b) - y^{(i)}\right)\frac{\partial}{\partial W_2}g(\omega^T h_2^{(i)} + b) && (4)\\
&= 2\left(g(\omega^T h_2^{(i)} + b) - y^{(i)}\right)\frac{\partial g(\omega^T h_2^{(i)} + b)}{\partial(\omega^T h_2^{(i)} + b)}.\frac{\partial(\omega^T h_2^{(i)} + b)}{\partial\omega} && (5)\\
&= 2\left[g(\omega^T h_2^{(i)} + b) - y^{(i)}\right]\left[1 - g(\omega^T h_2^{(i)} + b)\right]g(\omega^T h_2^{(i)} + b).h_2 && (6)\\
&= 2 \quad . \quad diff \quad . \quad dh3 \quad . \quad h2 && (7)\\
&= \quad b2grad \quad . \quad h2 && (8)
\end{aligned}
$$

where:

$$
\begin{aligned}
J(\omega, b) &= \sum_{i=1}^{m}(h(h_2^{(i)};\omega, b) - y^{(i)})^2 && (9)\\
h_2 = g(\theta^T x^{(i)} + b) &= \frac{1}{1 + exp(-(\theta^T x^{(i)} + b))} && (10)
\end{aligned}
$$

<u>**In first layer:**</u> $W_1(\theta)$

$$
\begin{aligned}
W_{1grad} = \Delta\omega &= \frac{\partial J(\omega, b)}{\partial W_1} && (11)\\
&= \frac{\partial}{\partial W_1}\left(h(h_2^{(i)};\omega, b) - y^{(i)}\right)^2 && (12)\\
&= 2\left(h(h_2^{(i)};\omega, b) - y^{(i)}\right)\frac{\partial}{\partial W_1}\left(h(h_2^{(i)};\omega, b) - y^{(i)}\right) && (13)\\
&= 2\left(g(\omega^T h_2^{(i)} + b) - y^{(i)}\right)\frac{\partial}{\partial W_1}g(\omega^T h_2^{(i)} + b) && (14)\\
&= 2\left(g(\omega^T h_2^{(i)} + b) - y^{(i)}\right)\frac{\partial g(\omega^T h_2^{(i)} + b)}{\partial(\omega^T h_2^{(i)} + b)}.\frac{\partial(\omega^T h_2^{(i)} + b)}{\partial W_1} && (15)\\
&= 2\left[g(\omega^T h_2^{(i)} + b) - y^{(i)}\right]\left[1 - g(\omega^T h_2^{(i)} + b)\right]g(\omega^T h_2^{(i)} + b).\frac{\partial(\omega^T h_2^{(i)} + b)}{\partial W_1} && (16)\\
&= \quad 2. \quad diff. \quad dh3 \quad .\frac{\partial(\omega^T h_2^{(i)} + b)}{\partial\theta} && (17)\\
&= 2. \quad diff. \quad dh3 \quad .\frac{\partial(\omega^T h_2^{(i)} + b)}{\partial g(\theta^T x^{(i)} + b)}.\frac{\partial g(\theta^T x^{(i)} + b)}{\partial(\theta^T x^{(i)} + b)}.\frac{\partial(\theta^T x^{(i)} + b)}{\partial\theta} && (18)\\
&= 2. \quad diff. \quad dh3. \quad W_2.\left[1 - g(\theta^T x^{(i)} + b)\right]g(\theta^T x^{(i)} + b).\frac{\partial(\theta^T x^{(i)} + b)}{\partial\theta} && (19)\\
&= 2. \quad diff. \quad dh3. \quad W_2. \quad dh2. \quad X && (20)\\
&= \quad b1grad \quad . \quad X && (21)
\end{aligned}
$$

```
# Backward pass to compute the gradient for every parameter
def backprop(Yi, Xi, W1, b1, W2, b2):
  # forward pass to compute the activation at each layer
  [J, diff, h1, h2, dh2, h3, dh3] = forprop(Yi, Xi, W1, b1, W2, b2)
  ############################################
  # WORK: Fill in your gradient computation here

  temp2 = scale(2 * diff, dh3)
  b2 = temp2
  temp2 = scale(temp2[0], h2)
  N = [];
  N.append(temp2)

  M = scale(2 * diff * dh3[0], W2[0])
  M = ew_dot(M, dh2)
  b1 = M
  M = cross(M, Xi)

  ############################################
  # Right now i am just setting the gradient the same as the
  # parameters. Replace the following 4 lines with your own backprop
  # computation (could be longer than 4 lines).
  W2grad = N;
  W1grad = M;
  b2grad = b2;
  b1grad = b1;
  return [J, W1grad, b1grad, W2grad, b2grad]
```

– Task #2: Fill in your stochastic updates here.

To minimize the above function, we can iterate through the examples and slowly update the parameters θ, ω and b in the direction of minimizing each of the small objective J. Concretely, we can update the parameters in the following manner:

$$W_2 = W_2 - \alpha \Delta W_2; \qquad (\omega = \omega - \alpha \Delta \omega) \tag{22}$$

$$W_1 = W_1 - \alpha \Delta W_1; \qquad (\theta = \theta - \alpha \Delta \theta) \tag{23}$$

$$b_i = b_i - \alpha \Delta b_i; \tag{24}$$

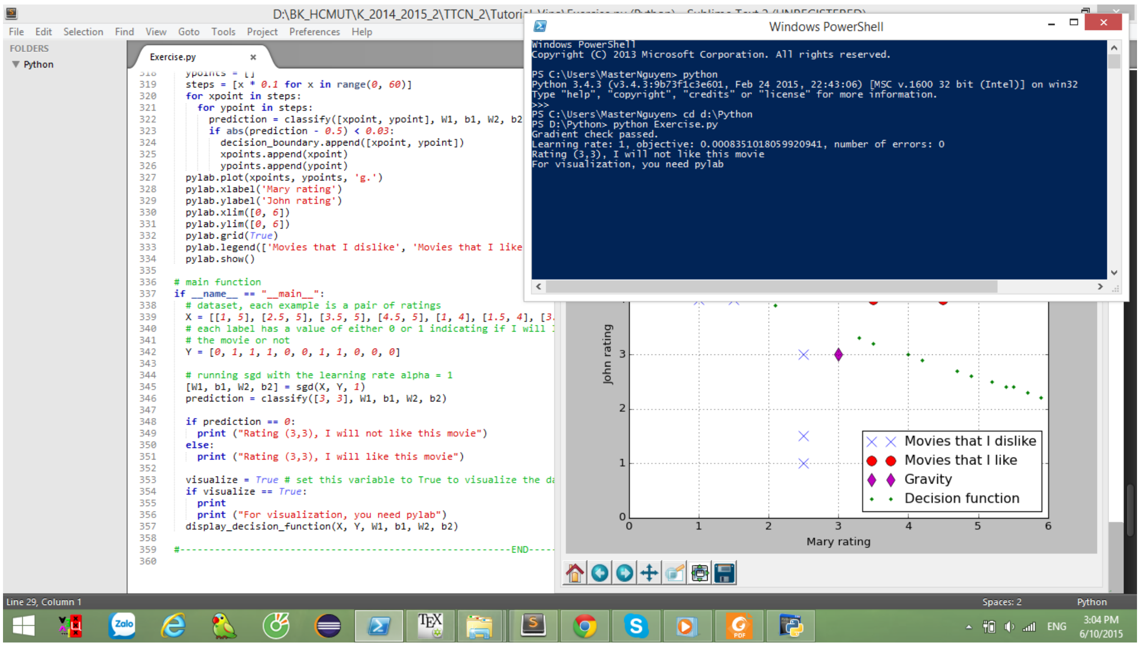
```
for i in range(10000):
  # Sample one example from the dataset
  index = randint(0, len(X) - 1)  # this is the index of the example

  # Run backprop to compute the gradient
  [J, W1grad, b1grad, W2grad, b2grad] = backprop(Y[index], X[index], W1, b1, W2, b2)
  # Use the computed gradient to update parameters
  ############################################
  # WORK: Fill in your stochastic updates here

  W1grad = scale_matrix(-alpha, W1grad)
  b1grad = scale(-alpha, b1grad)
  W2grad = scale_matrix(-alpha, W2grad)
  b2grad = scale(-alpha, b2grad)

  ############################################
  # For now, I just keep the parameters to be the same. Replace the
  # following 4 lines with your own updates.
  W1 = add_matrix(W1, W1grad)
  W2 = add_matrix(W2, W2grad)
  b1 = add(b1, b1grad)
  b2 = add(b2, b2grad)
```

2 Demo

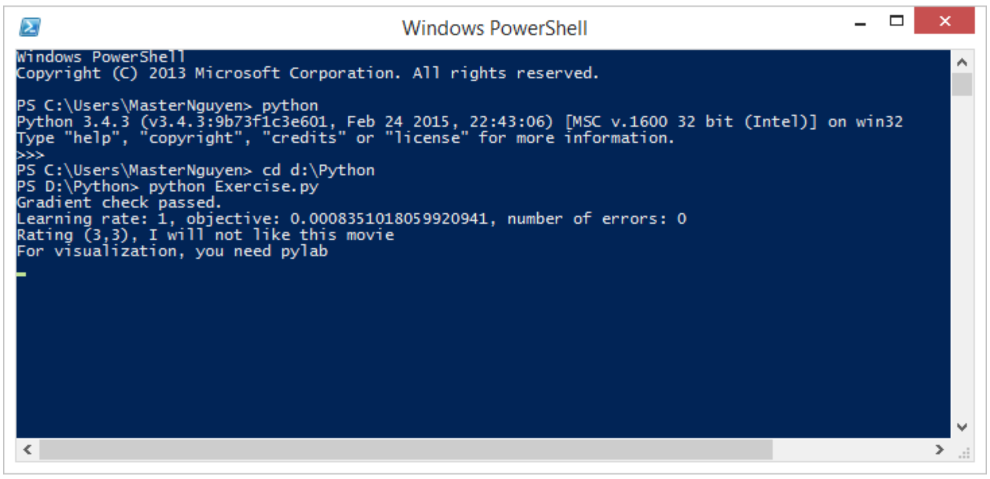
Introduction

1. Using **Python version 3.4.3**: Some syntax in old file can error. So, I changed its. For example:
   - *print* $s \Longrightarrow$ *print (s)*
   - *xrange(len(W2))* $\Longrightarrow$ *range(len(W2))*
2. Editor: Sublime.
3. Comiler: Window PowerShell.

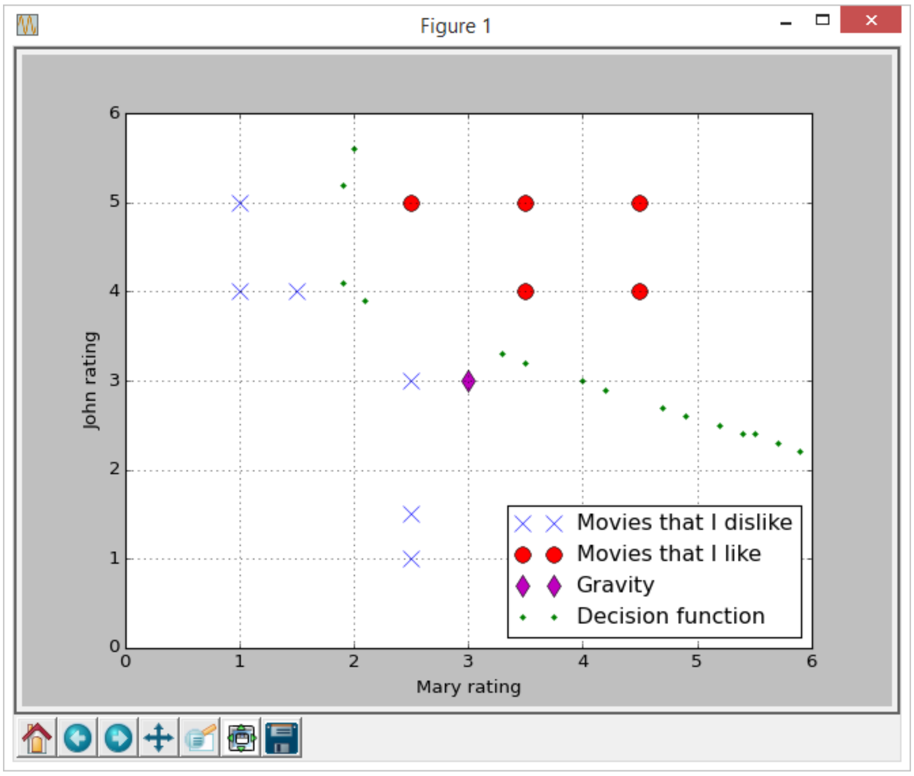
- *Note: You should prepare a good environment (libraries, compiler..) if you run it again.*



Workspace.

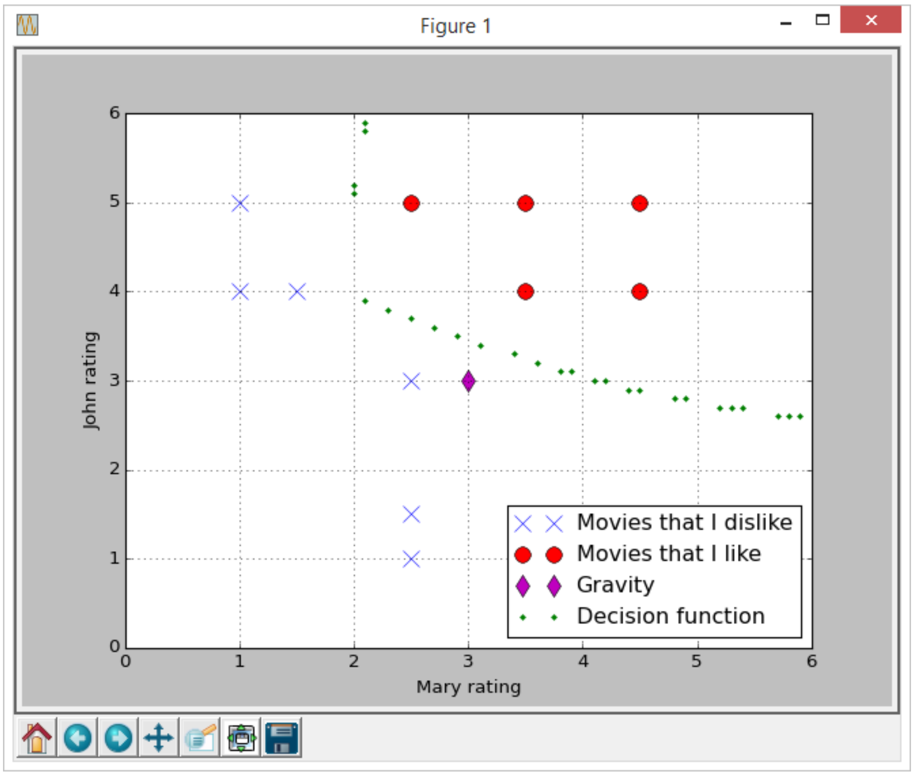


Window Powershell.



Display Graph by source code Python.

<u>There are a number of different types of graphs which using a random values:</u>



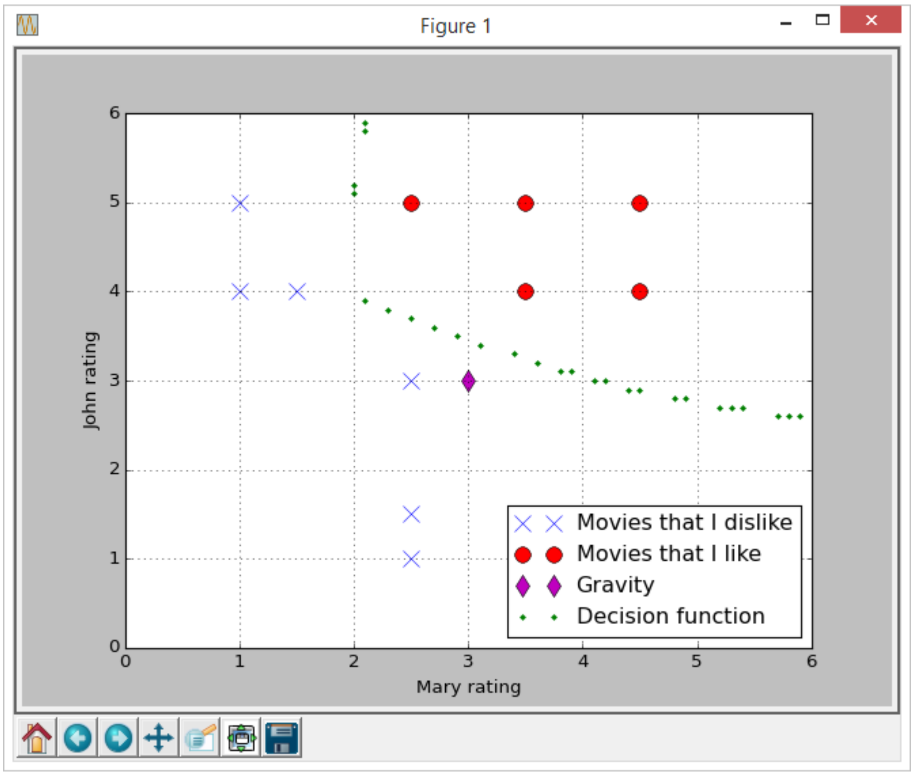


Figure 1
6
5
4
3
2
1
0
John rating
0
1
2
3
4
5
6
Mary rating
Movies that I dislike
Movies that I like
Gravity
Decision function

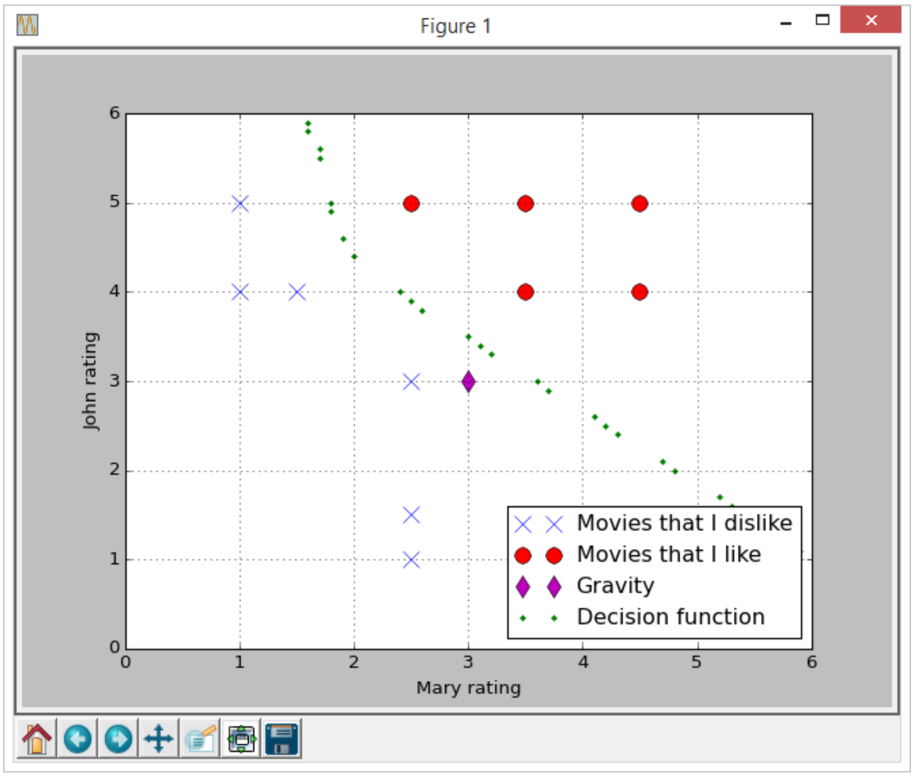


Figure 1
6
5
4
3
2
1
0
John rating
0
1
2
3
4
5
6
Mary rating
Movies that I dislike
Movies that I like
Gravity
Decision function

References

[1] Quoc Le's Lectures on Deep Learning.

[2] https://www.youtube.com/watch?v=GlcnxUlrtek, Neural Networks Demystified [Part 4- Backpropagation].

[3] https://www.coursera.org/learn/machine-learning/home/welcome, Machine Learning by Stanford University.

[4] http://andrew.gibiansky.com/, Quick coding intro to Neural Networks.

[5] http://aimotion.blogspot.com/2011/10/machine-learning-with-python-linear.html, Machine Learning with Python - Linear Regression.