INTRO TO

MACHINE LEARNING

@mihahribar

TOSHL FINANCE

Taught by: Andrew Ng, Co-founder, Coursera; Adjunct Professor, Stanford University; formerly head of Baidu AI Group/Google Brain

Coursework

Each course is like an interactive textbook, featuring pre-recorded videos, quizzes and projects.

Help from Your Peers

Connect with thousands of other learners and debate ideas, discuss course material, and get help mastering concepts.

Certificates

Earn official recognition for your work, and share your success with friends, colleagues, and employers.

https://www.coursera.org/learn/machine-learning

Maths!

Gradient descent

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$(j = \cancel{0}, 1, 2, 3, \ldots, n)$

$\}$

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$1 - \alpha \frac{\lambda}{m} < 1$

0.99

$\theta_j \times 0.99$

Collaborative filtering optimization objective

- Given $x^{(1)}, \ldots, x^{(n_m)}$, estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

- Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, estimate $x^{(1)}, \ldots, x^{(n_m)}$:

$$\min_{x^{(1)},\ldots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2$$

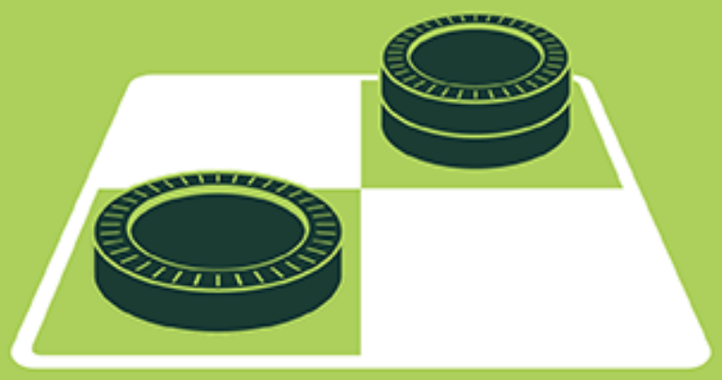
Minimizing $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ simultaneously:

$$J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)}\\ \theta^{(1)},\ldots,\theta^{(n_u)}}} J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$$

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

Machine learning begins to flourish.

DEEP LEARNING

Deep learning breakthroughs drive AI boom.

1950's 1960's 1970's 1980's 1990's 2000's 2010's

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

“Field of study that gives computers the ability to learn without being explicitly programmed.”

– Arthur Samuel (1959)

“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

– Tom Mitchell (1998)

“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.”

Example: spam email classification based on user input

T: Classifying emails as spam/not spam

E: Users marking emails as spam/not spam

P: Number of emails correctly marked as spam

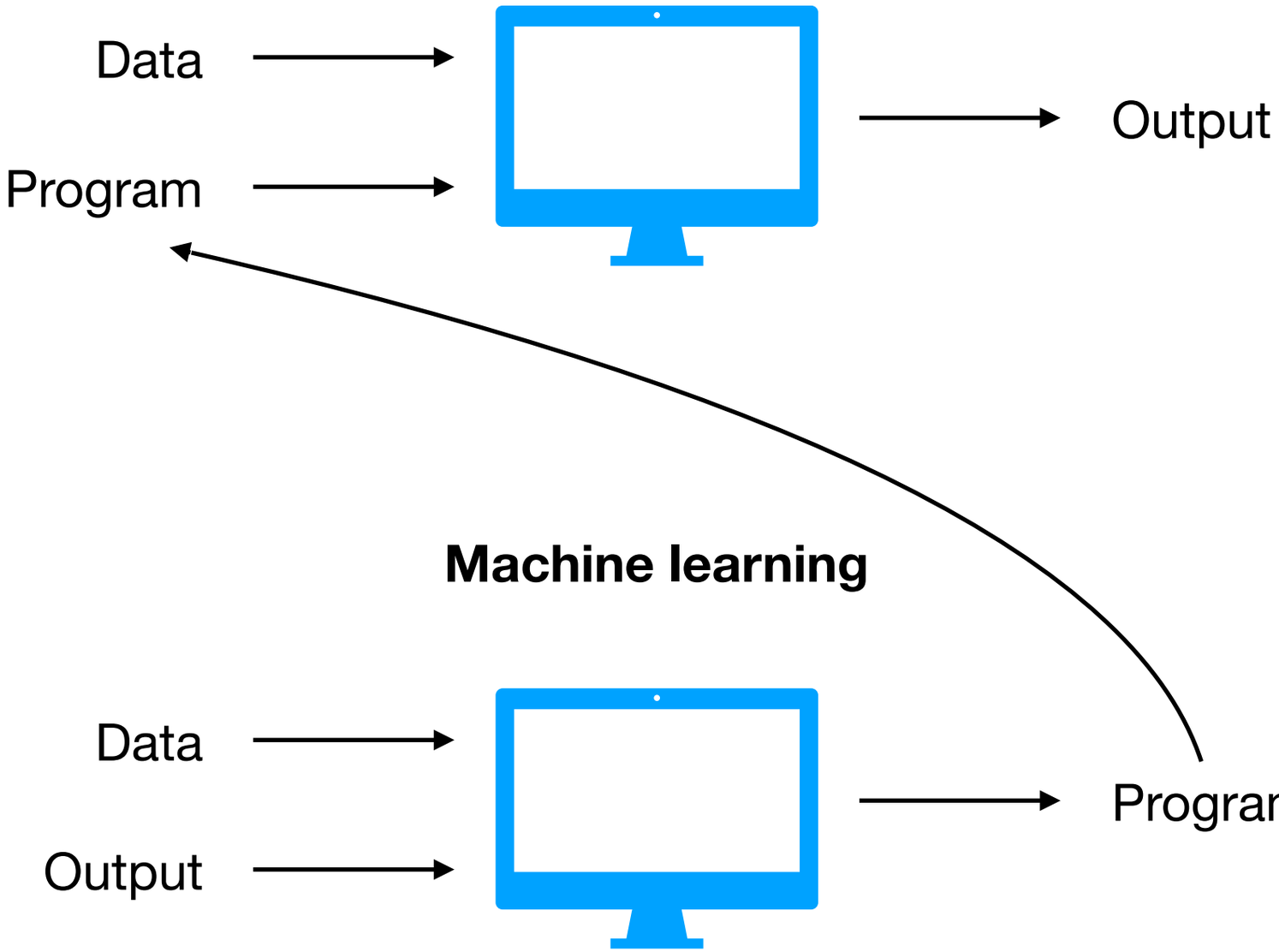
BASED ON YOUR INTERNET HISTORY, YOU MIGHT BE DUMB ENOUGH TO ENJOY EXTREME SPORTS.

CLICK HERE TO BUY A TICKET TO BASE JUMP FROM THE INTERNATIONAL SPACE STATION.

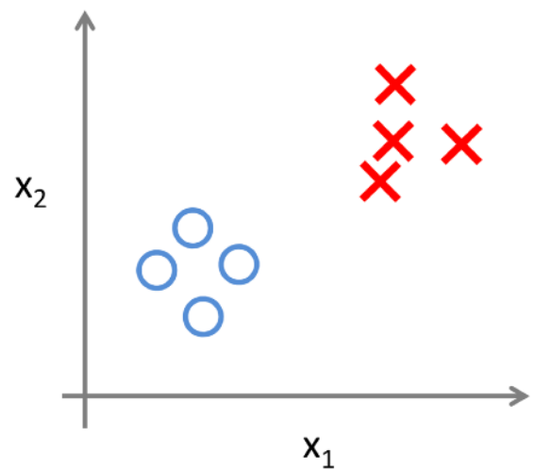
I THINK THE INTERNET IS TRYING TO KILL ME.

WE CALL IT "MACHINE LEARNING."

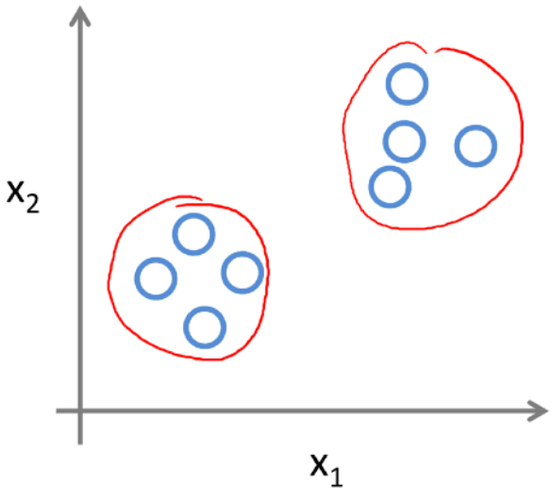
Traditional programming



- **Supervised learning**
  Computer is presented with example inputs and their desired outputs and the goal is to learn a general rule that maps inputs to outputs



- **Unsupervised learning**
  No labels are given to the learning algorithm, leaving it on its own to find structure in its input.

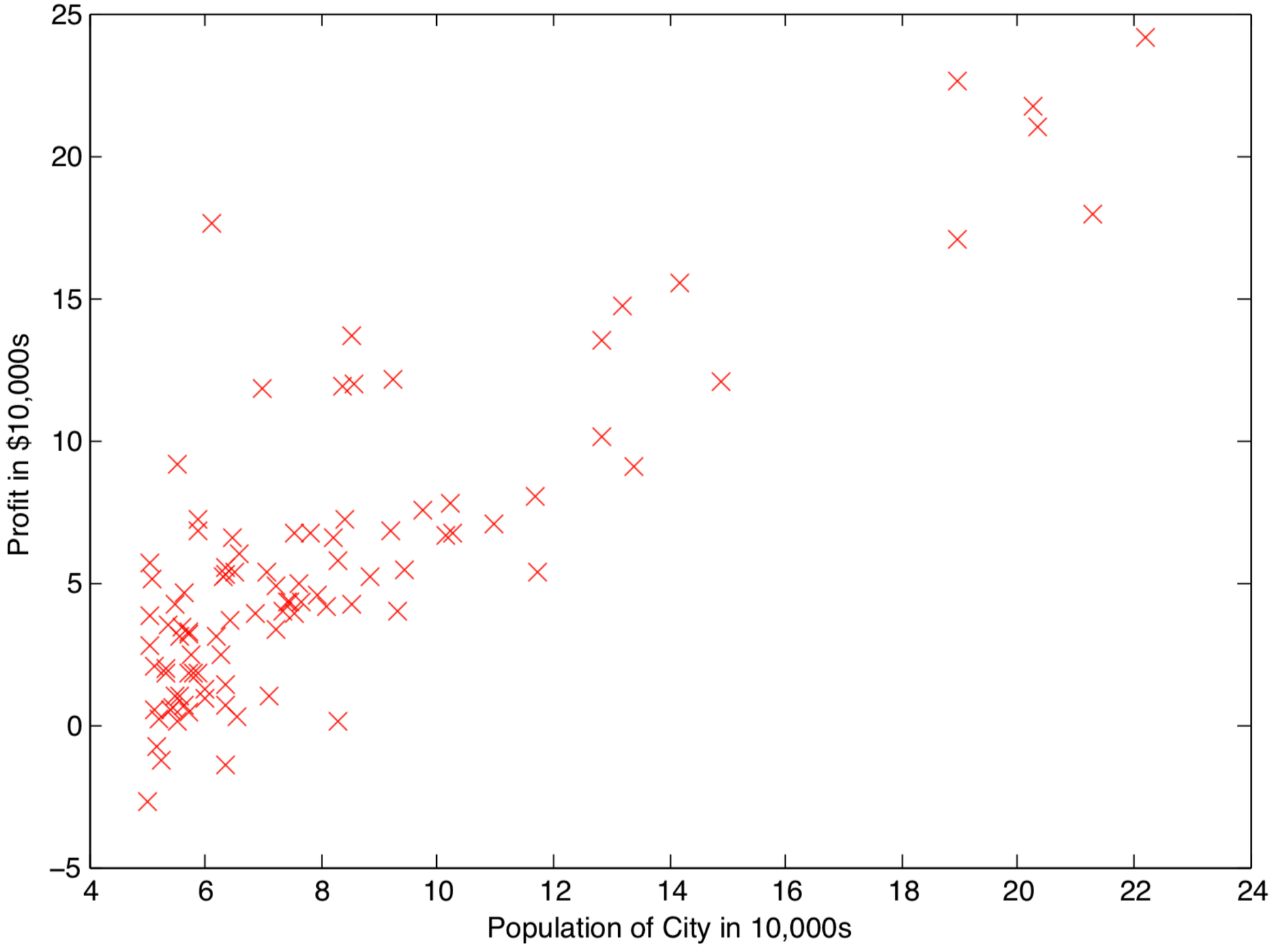


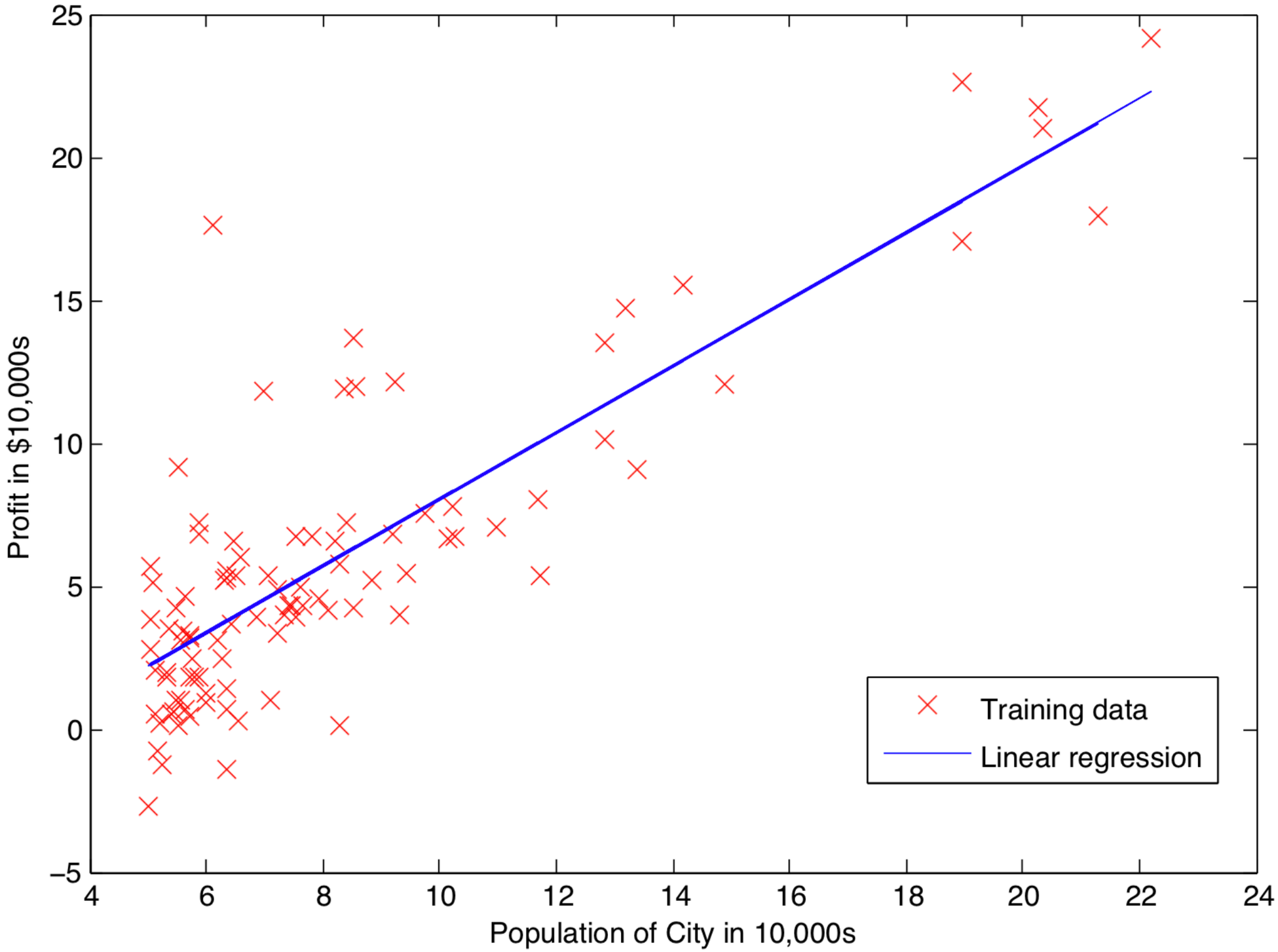
Supervised learning

- Linear regression
- Logistic regression
- Support Vector Machines
- Neural networks

Unsupervised learning

- K-means
- Anomaly detection
- Neural networks
- Recommender systems


Profit in $10,000s
Population of City in 10,000s
25
20
15
10
5
0
−5
4
6
8
10
12
14
16
18
20
22
24


25
20
15
10
5
0
−5
4
6
8
10
12
14
16
18
20
22
24
Profit in $10,000s
Population of City in 10,000s
Training data
Linear regression

Linear regression

hypotheses

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

cost function

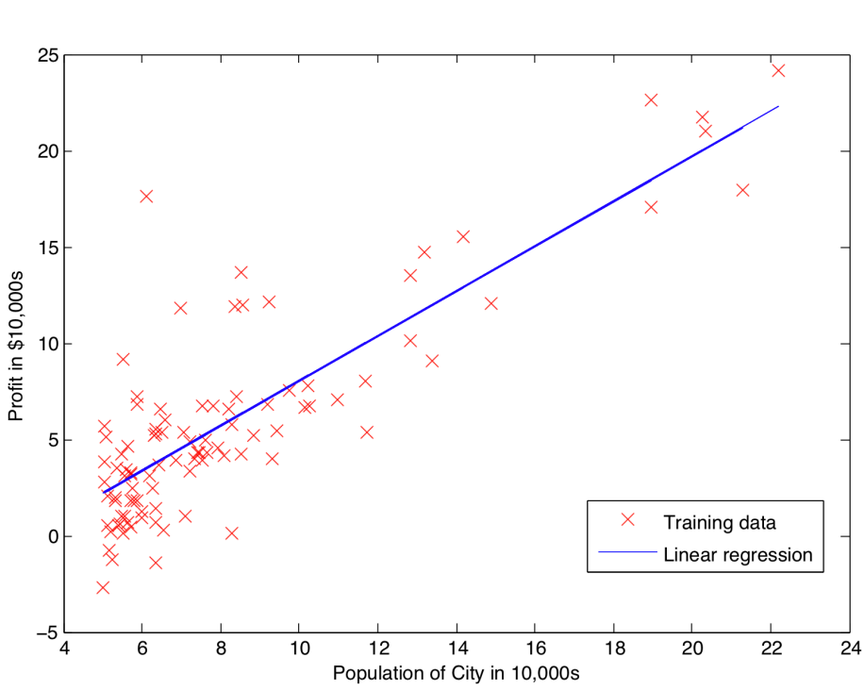
$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2.$$

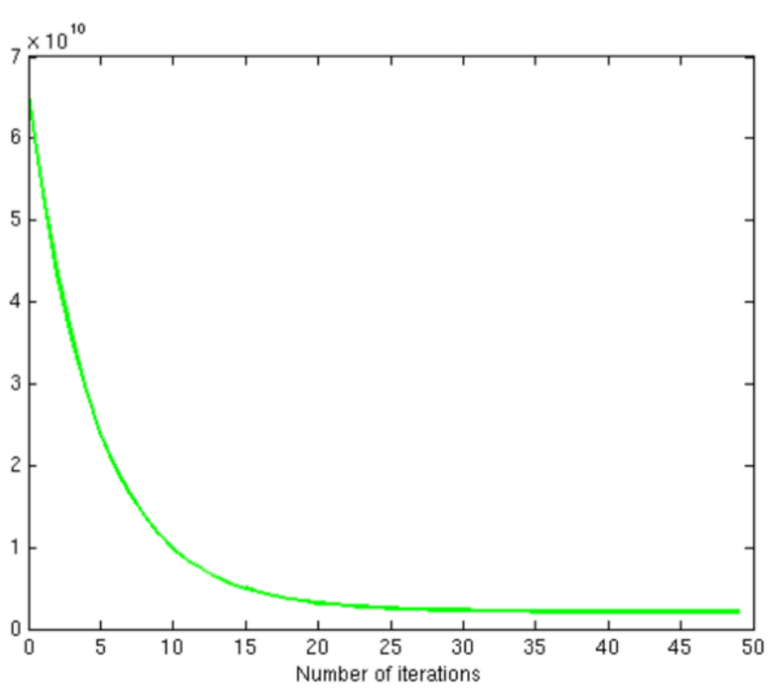
gradient descent

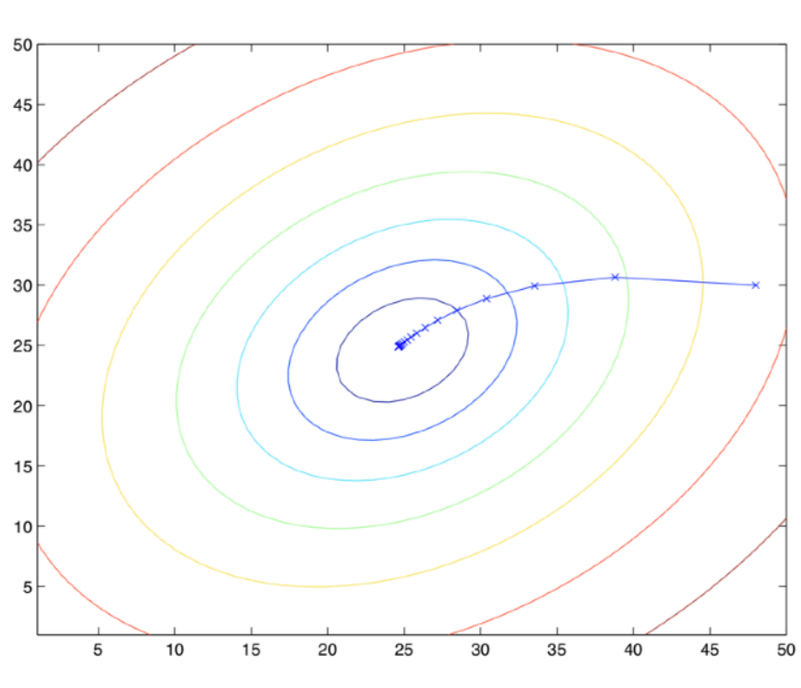
Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

}







1 iteration

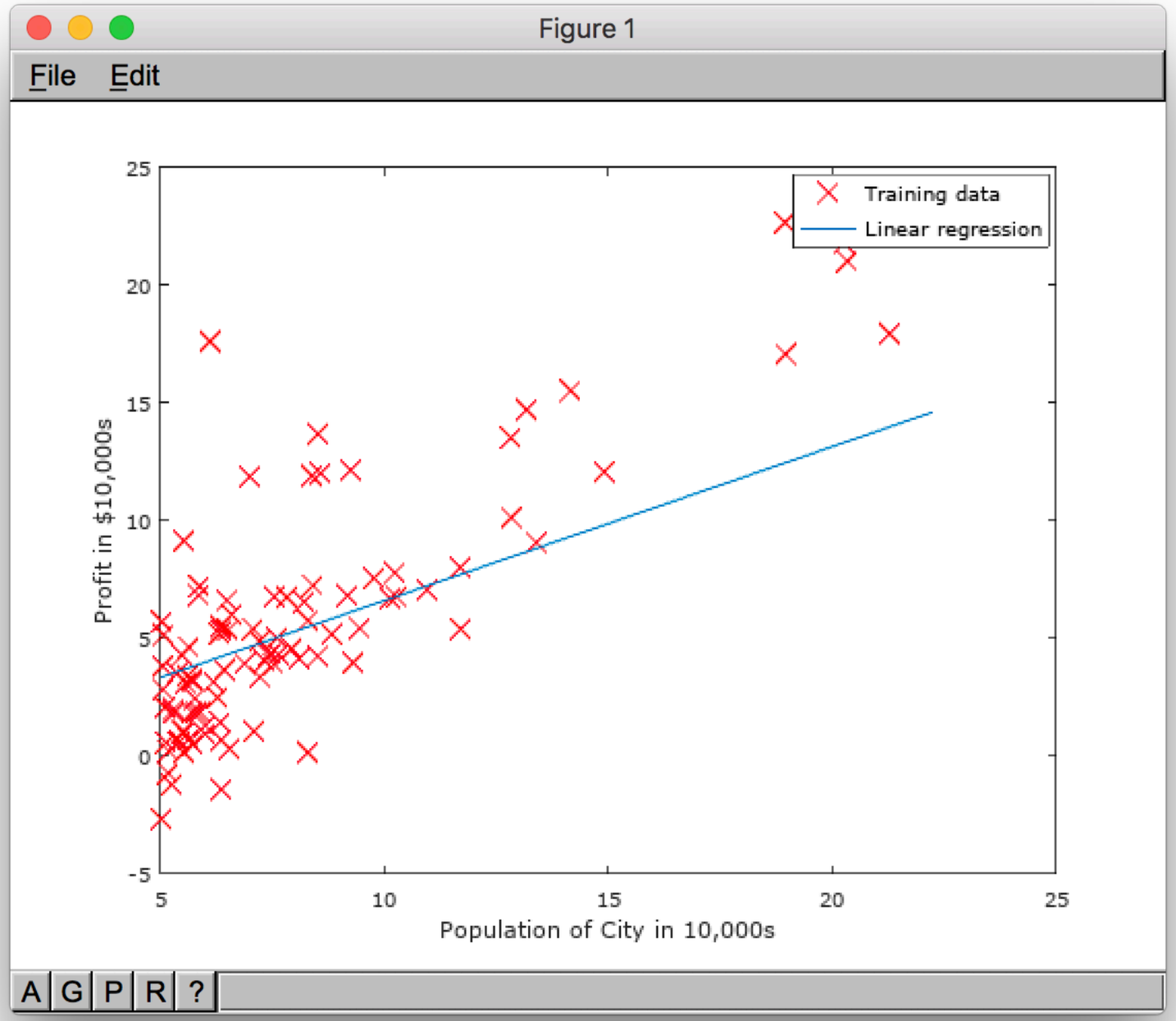
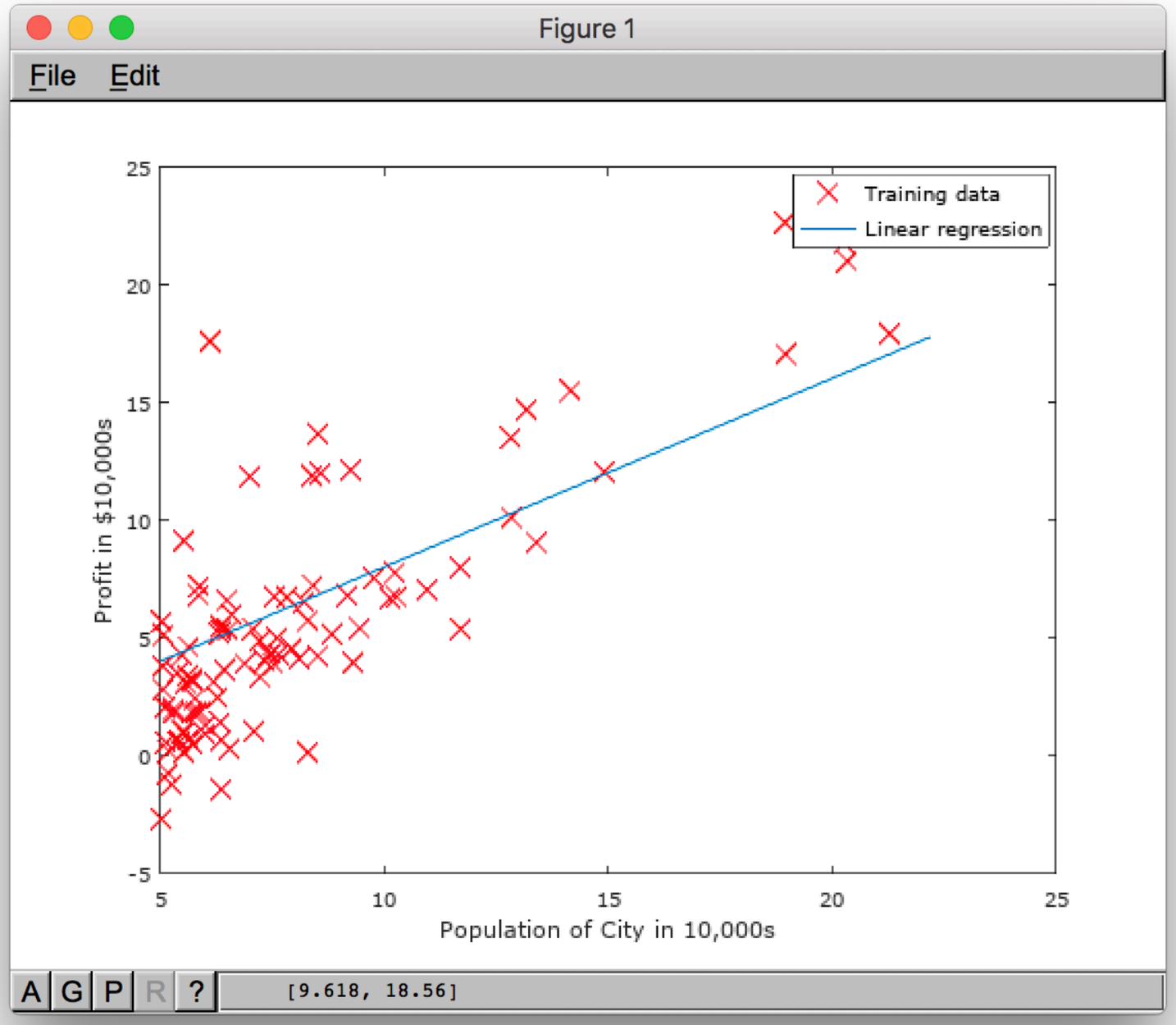
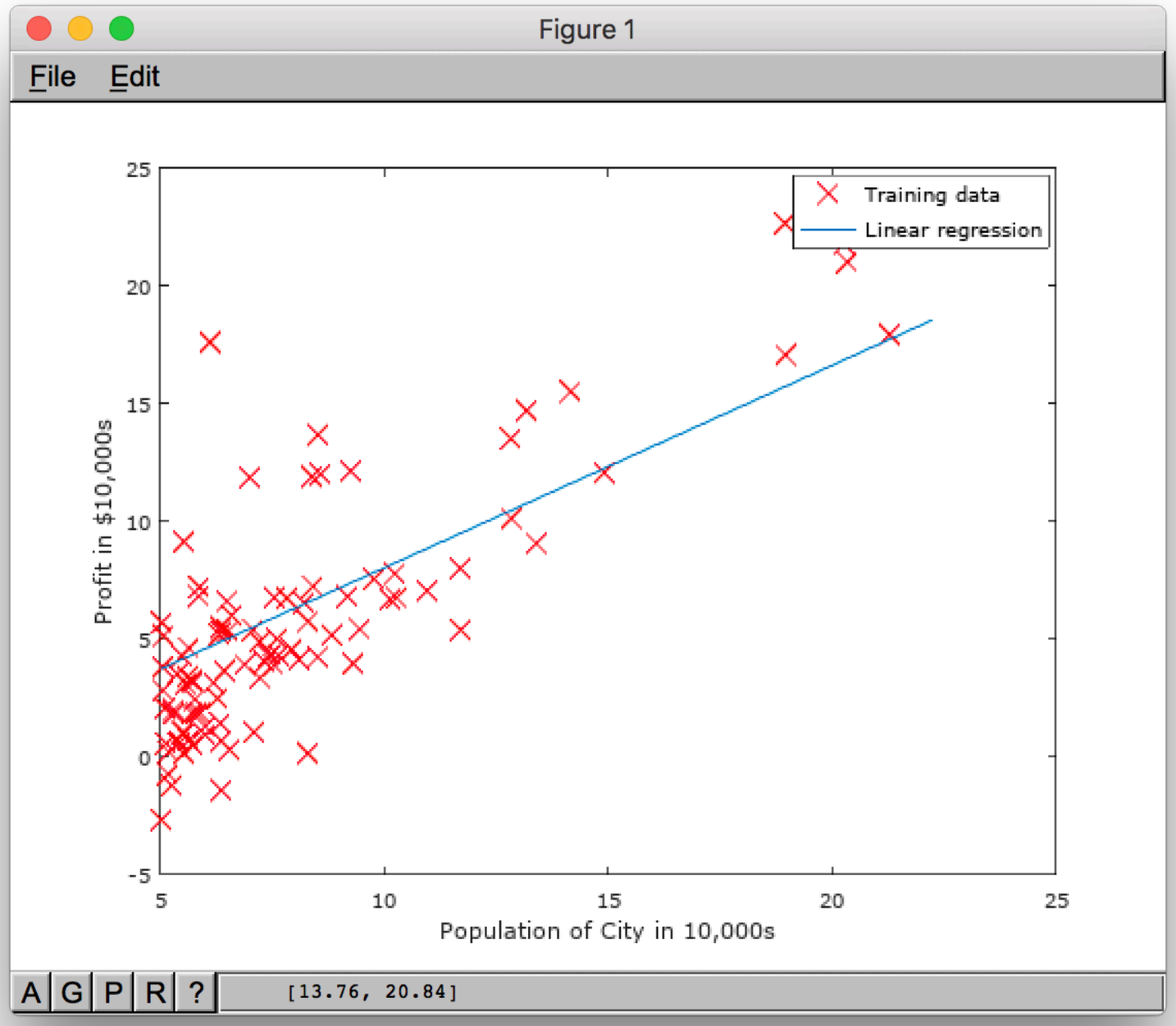


Figure 1
File Edit
Training data
Linear regression
Profit in $10,000s
Population of City in 10,000s
25
20
15
10
5
0
-5
5
10
15
20
25
A G P R ?

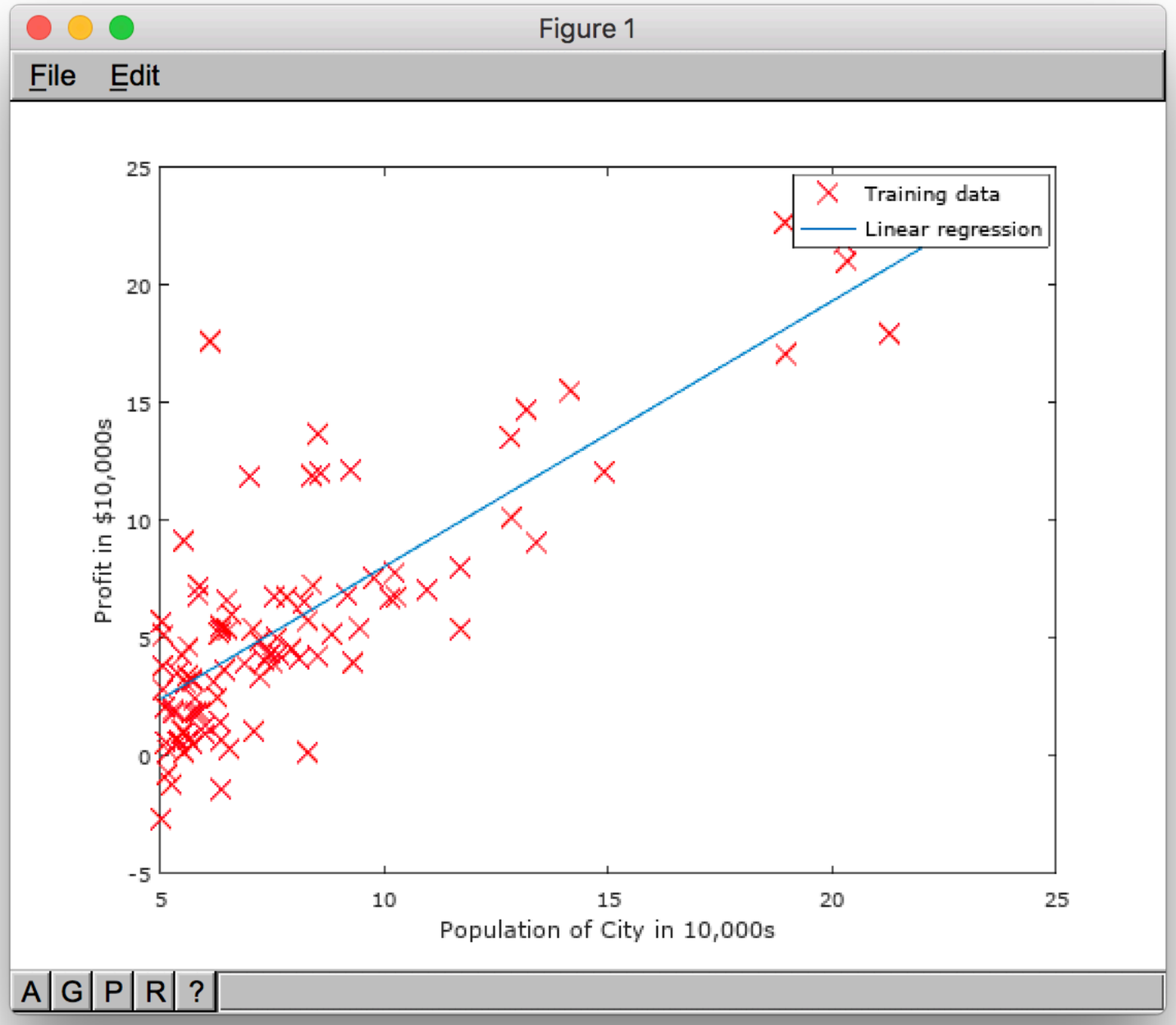
10 iterations



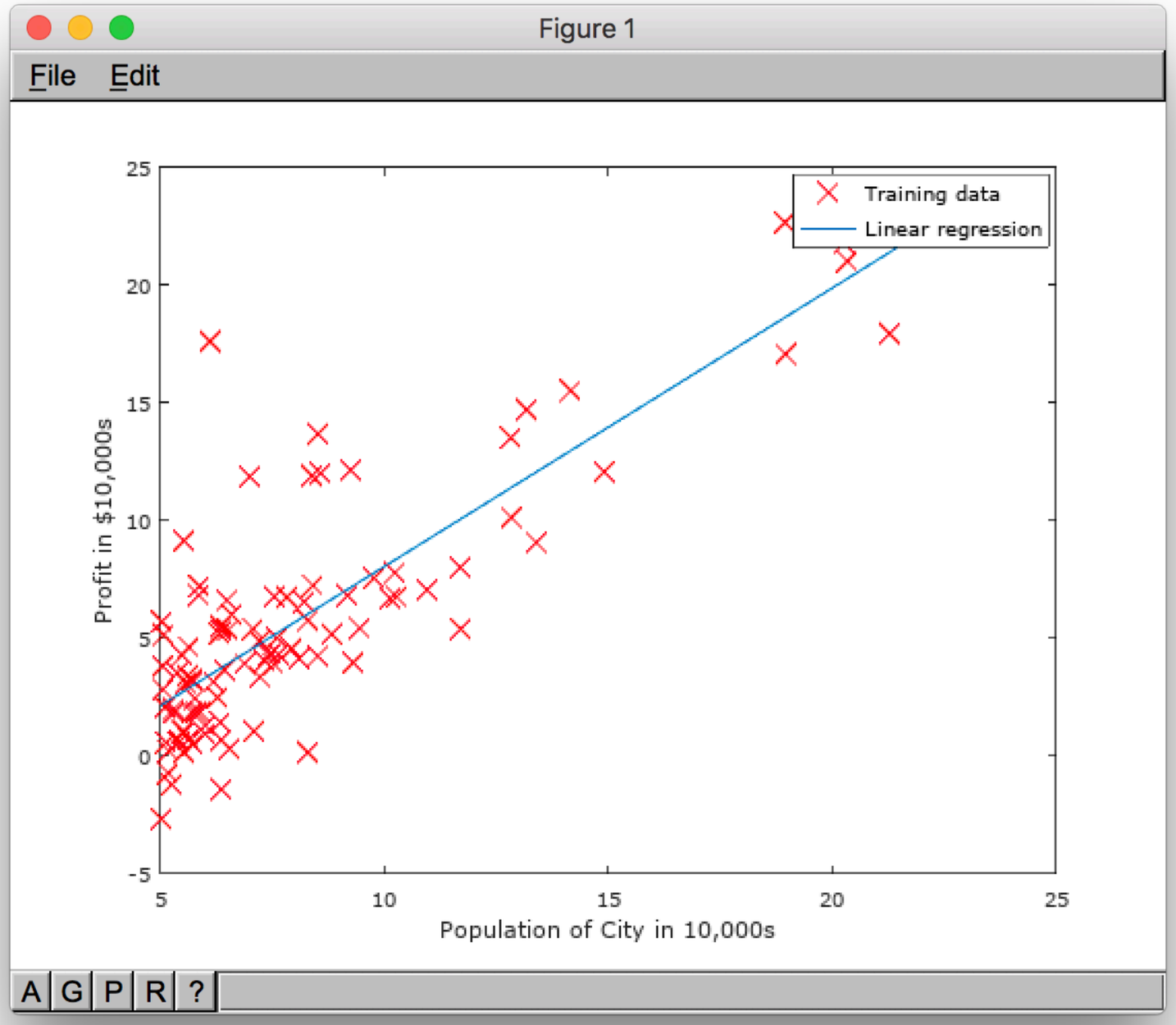
100 iterations



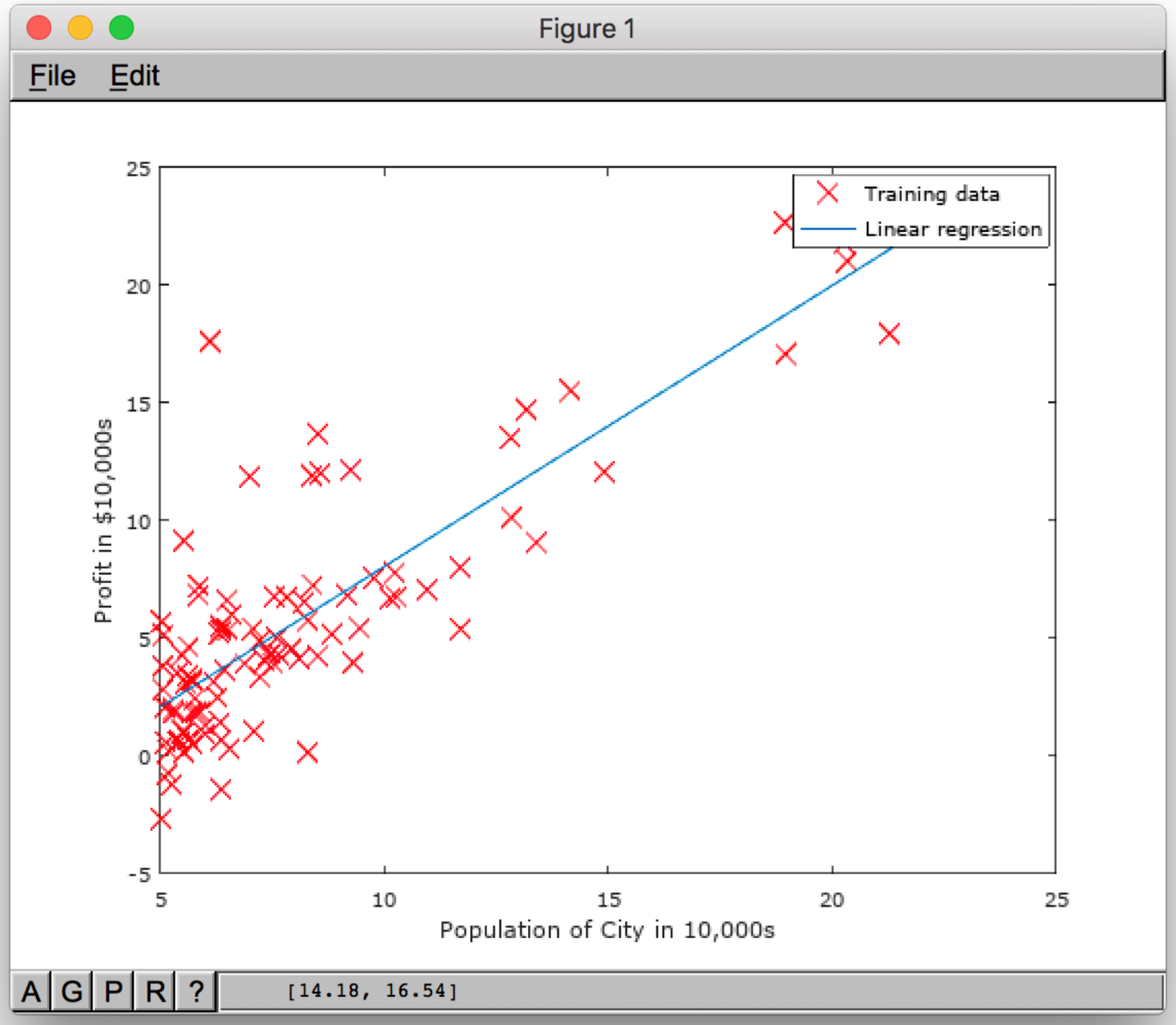
1000 iterations



2000 iterations

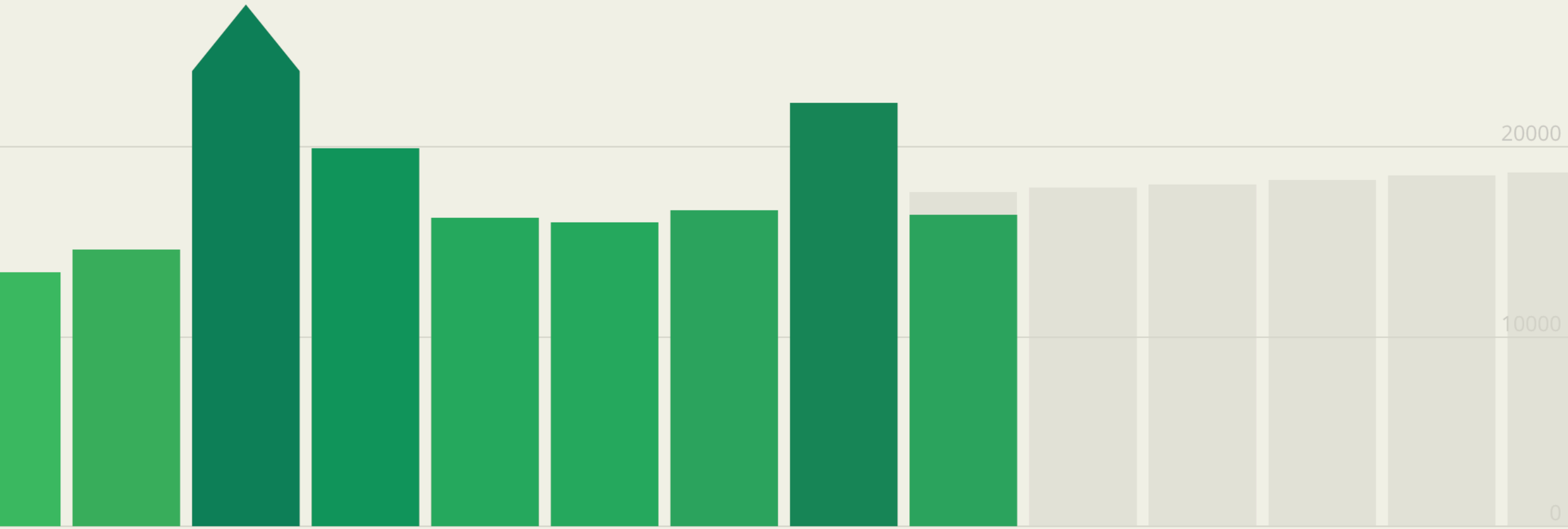


10000 iterations



February

Balance | Expenses | Incomes | Net Worth



Jul | Aug | Sep | Oct | Nov | Dec | Jan | Feb | Mar | Apr | May | Jun

Normal equation

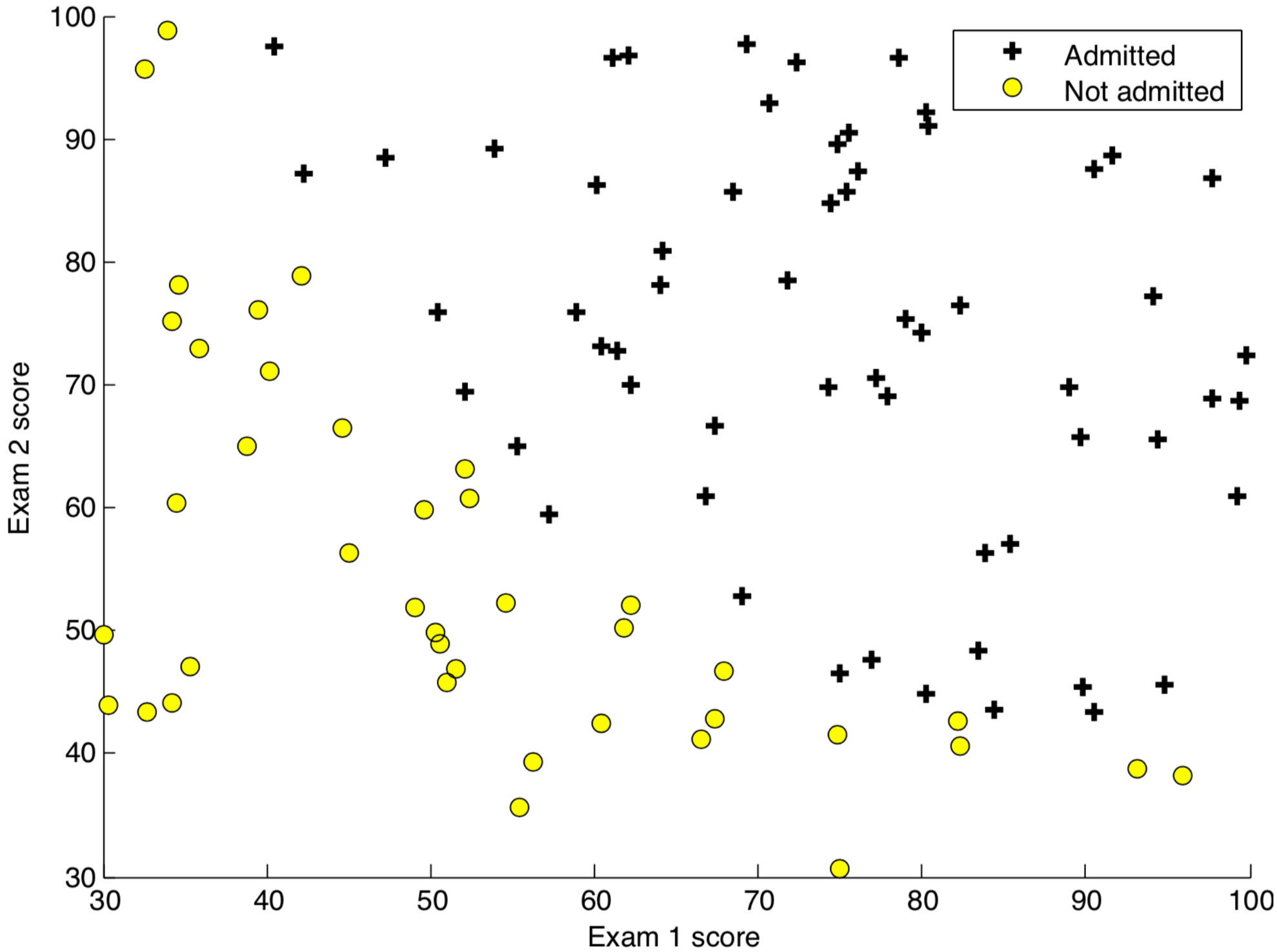
$$\theta = \left(X^T X\right)^{-1} X^T \vec{y}.$$

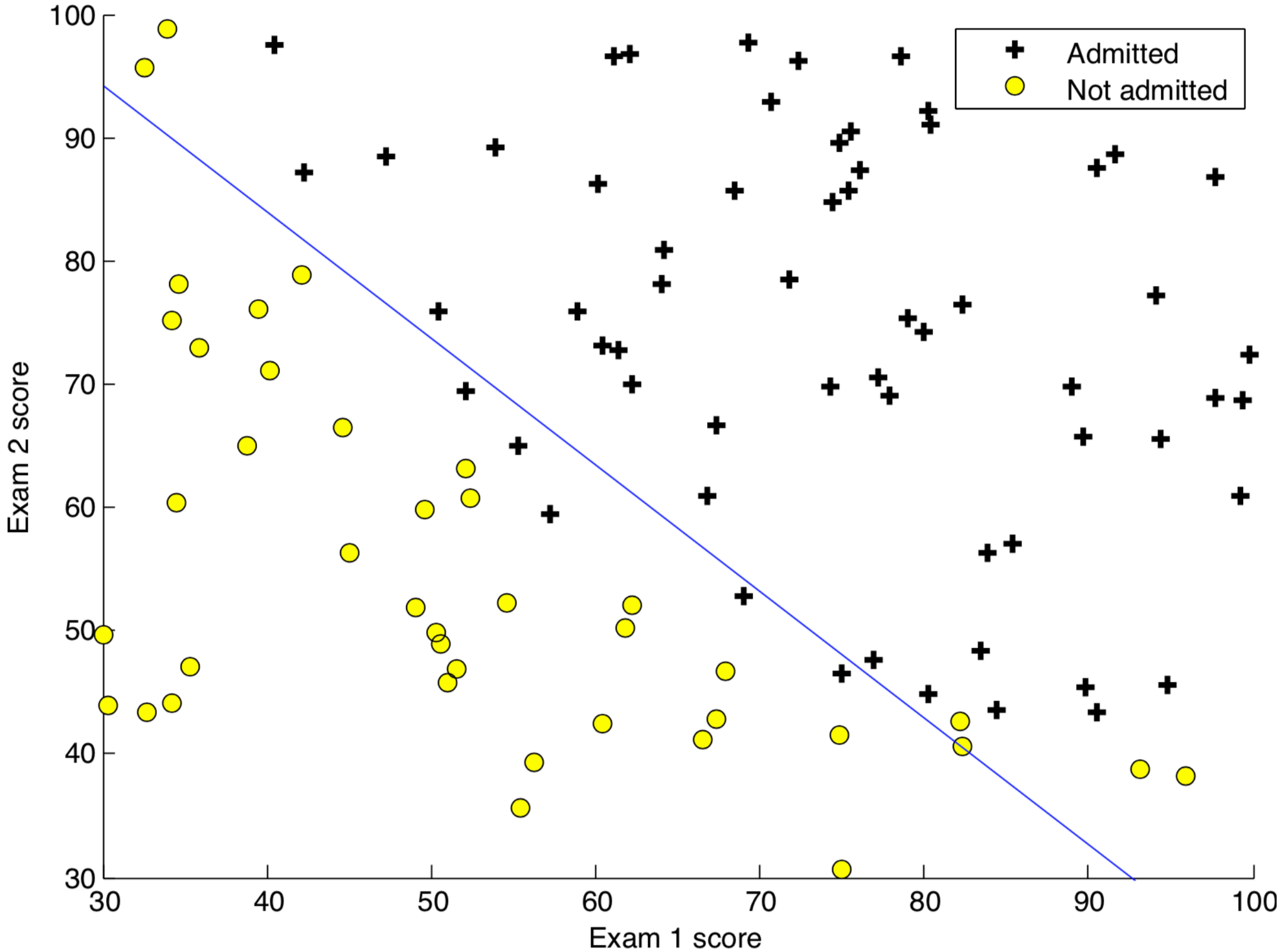
- computed in one step
- slow if n is large
- need to compute inv(X'X) - very slow
- problem if X'X not invertible
- prune outliers

```
import org.apache.commons.math3.linear.*;

private RealVector linearRegression(List<BigDecimal> list)
{
        double[] vector = new double[list.size()];
        double[][] matrix = new double[list.size()][2];
        // fill in vector and matrix
        RealMatrix X = MatrixUtils.createRealMatrix(matrix);
        RealVector y = MatrixUtils.createRealVector(vector);
        return MatrixUtils
                        .inverse(X.transpose().multiply(X))
                        .multiply(X.transpose())
                        .operate(y);
}
```

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$


Admitted
Not admitted
Exam 2 score
Exam 1 score
100
90
80
70
60
50
40
30
30
40
50
60
70
80
90
100


Admitted
Not admitted
Exam 2 score
Exam 1 score
100
90
80
70
60
50
40
30
30
40
50
60
70
80
90
100

Logistic regression

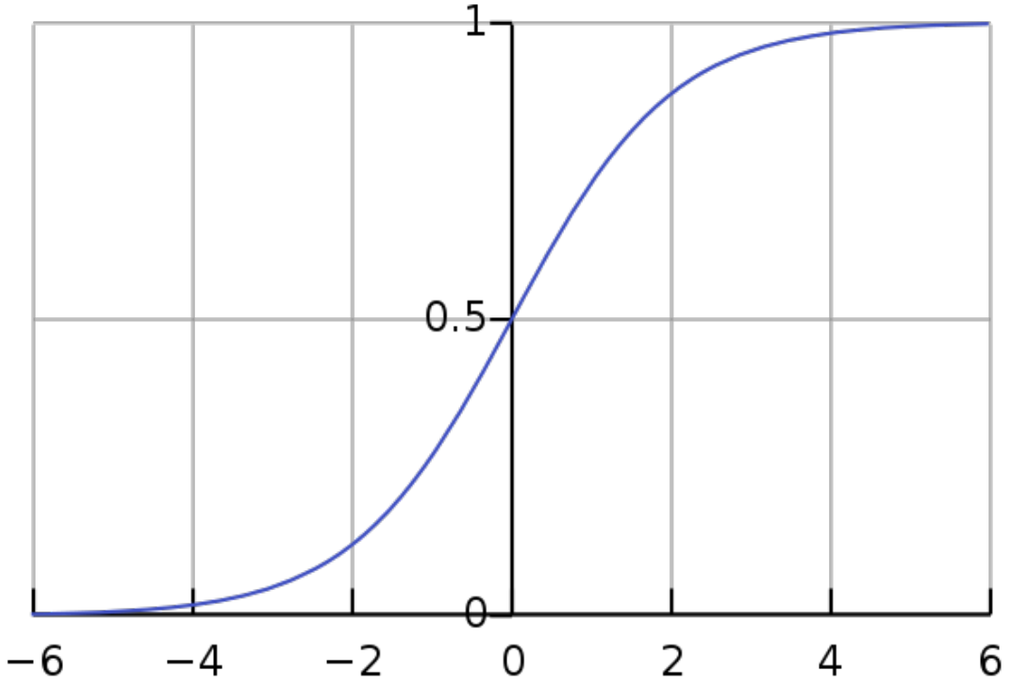
hypotheses

$$h_\theta(x) = g(\theta^T x),$$

$$g(z) = \frac{1}{1+e^{-z}}.$$

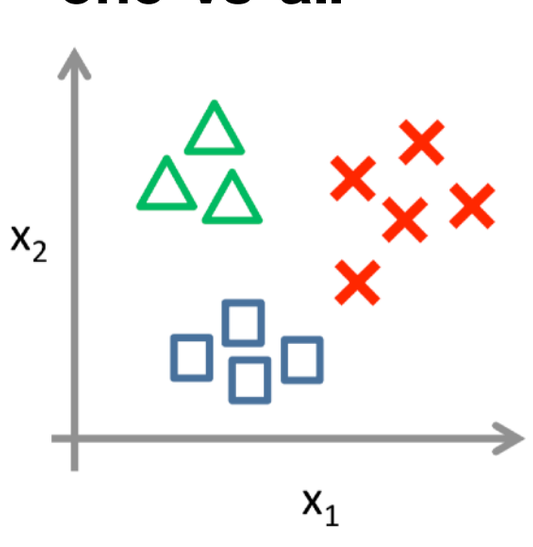
cost function

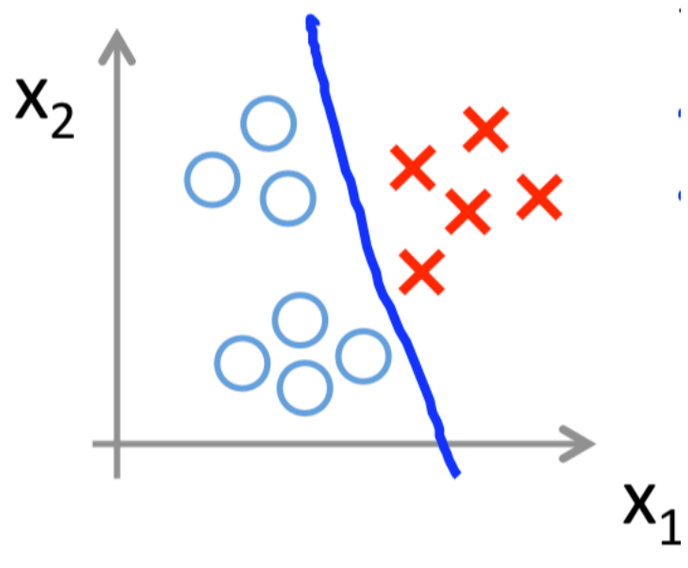
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$



$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

one-vs-all





$$\max_i h_\theta^{(i)}(x)$$

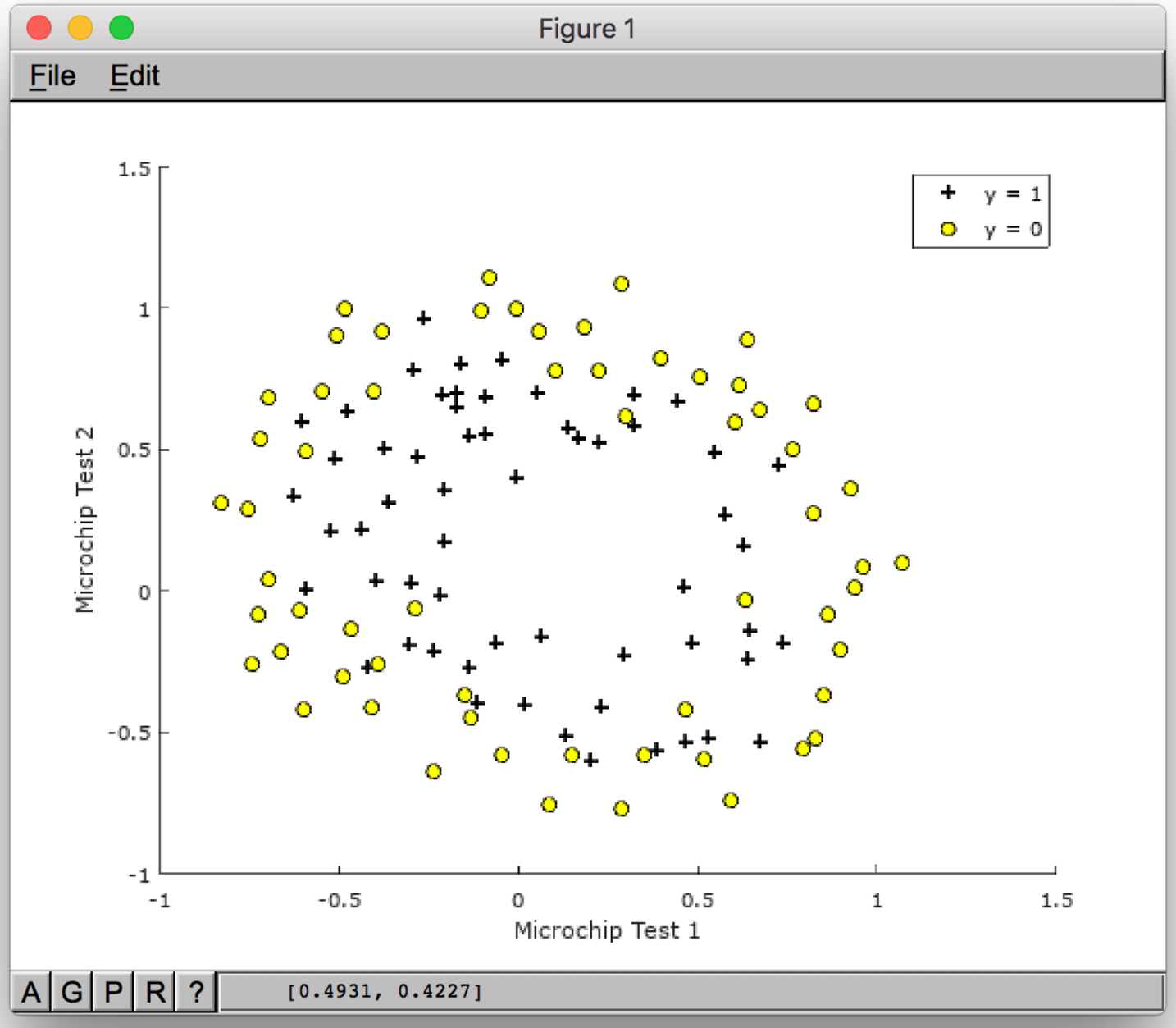


Figure 1
File Edit
y = 1
y = 0
Microchip Test 2
Microchip Test 1
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5
A G P R ?
[0.4931, 0.4227]

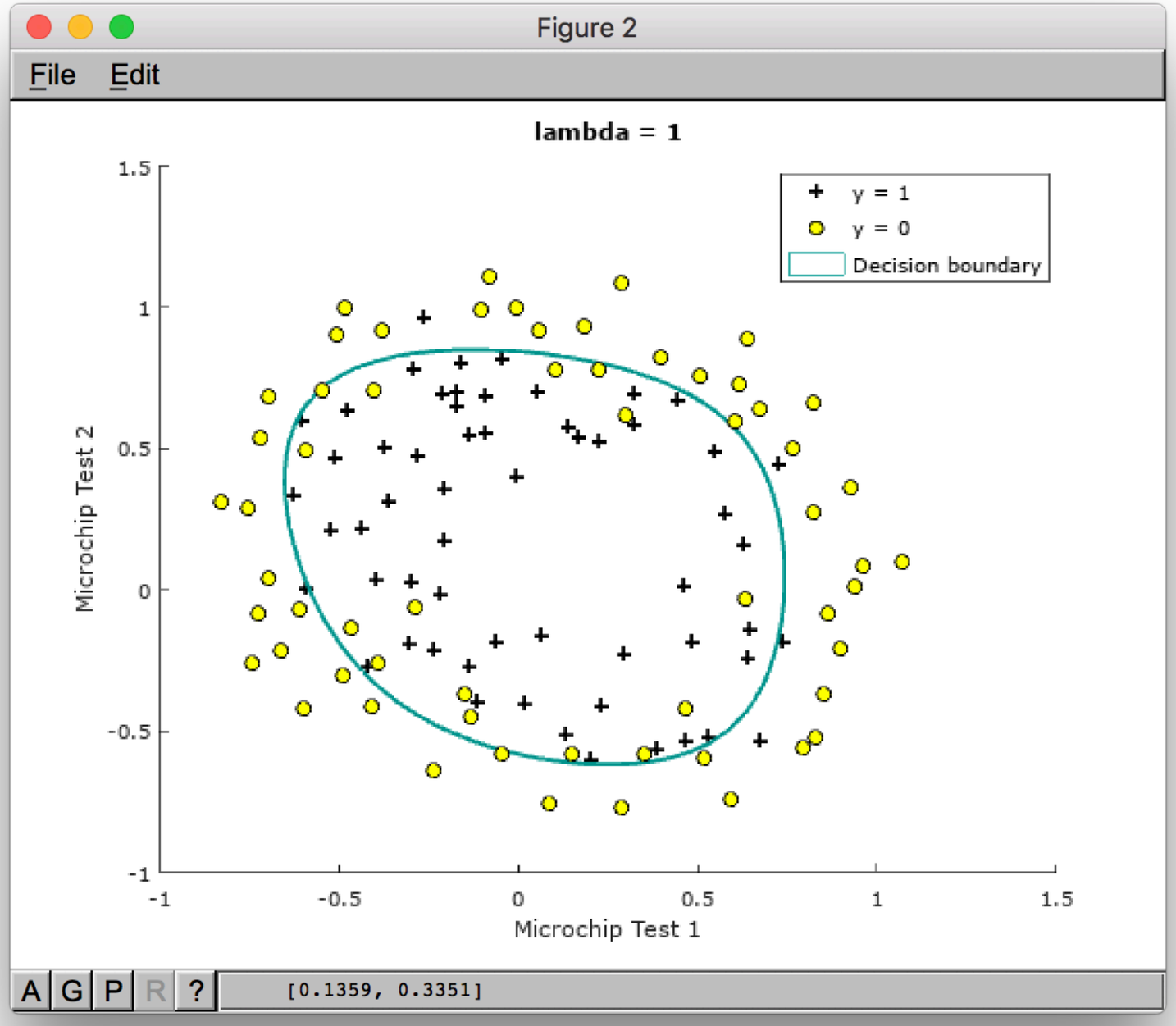


Figure 2
File Edit
lambda = 1
y = 1
y = 0
Decision boundary
Microchip Test 2
Microchip Test 1
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5
A G P R ?
[0.1359, 0.3351]

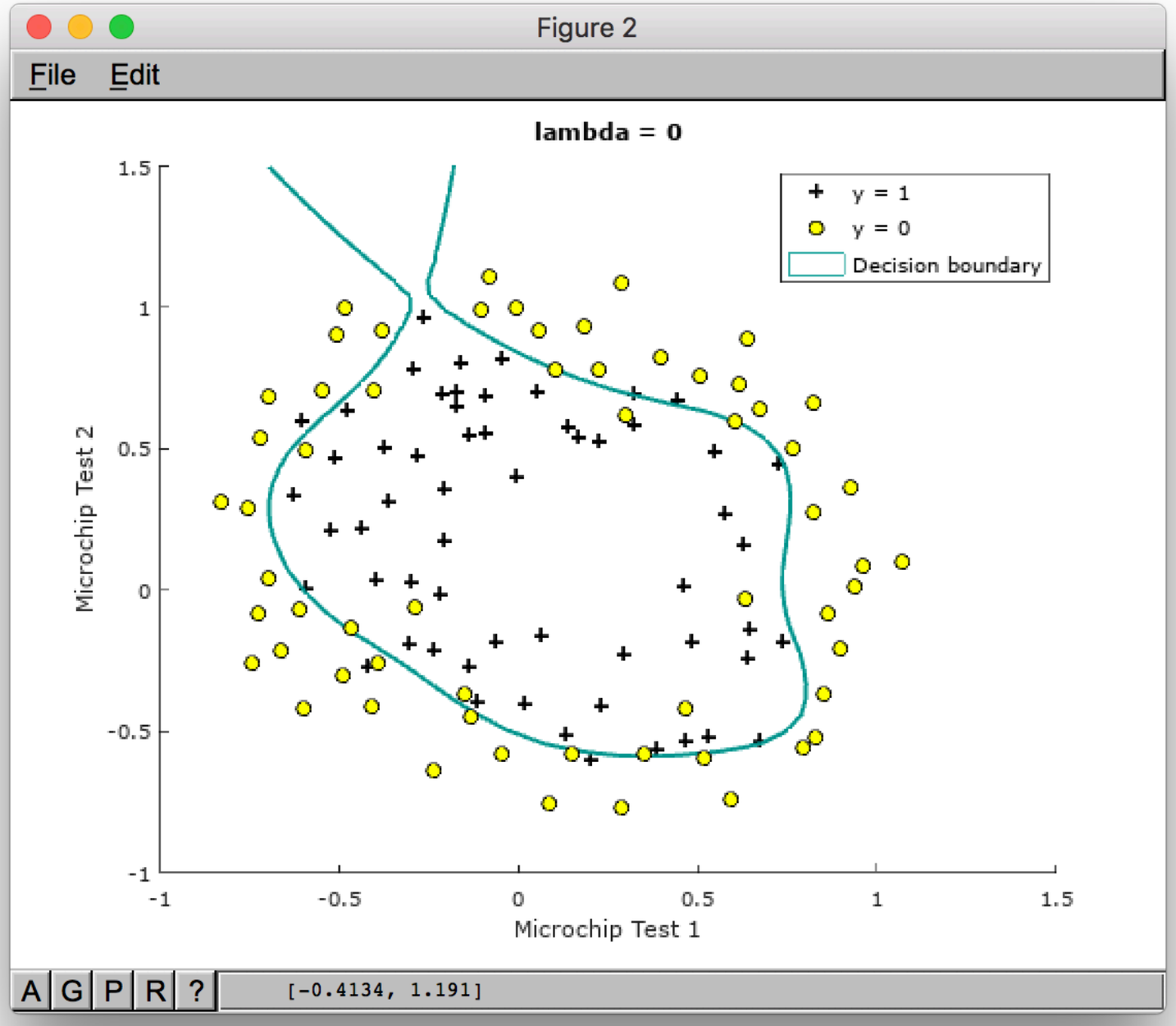


Figure 2
File Edit
lambda = 0
y = 1
y = 0
Decision boundary
Microchip Test 2
Microchip Test 1
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5
A G P R ?
[-0.4134, 1.191]

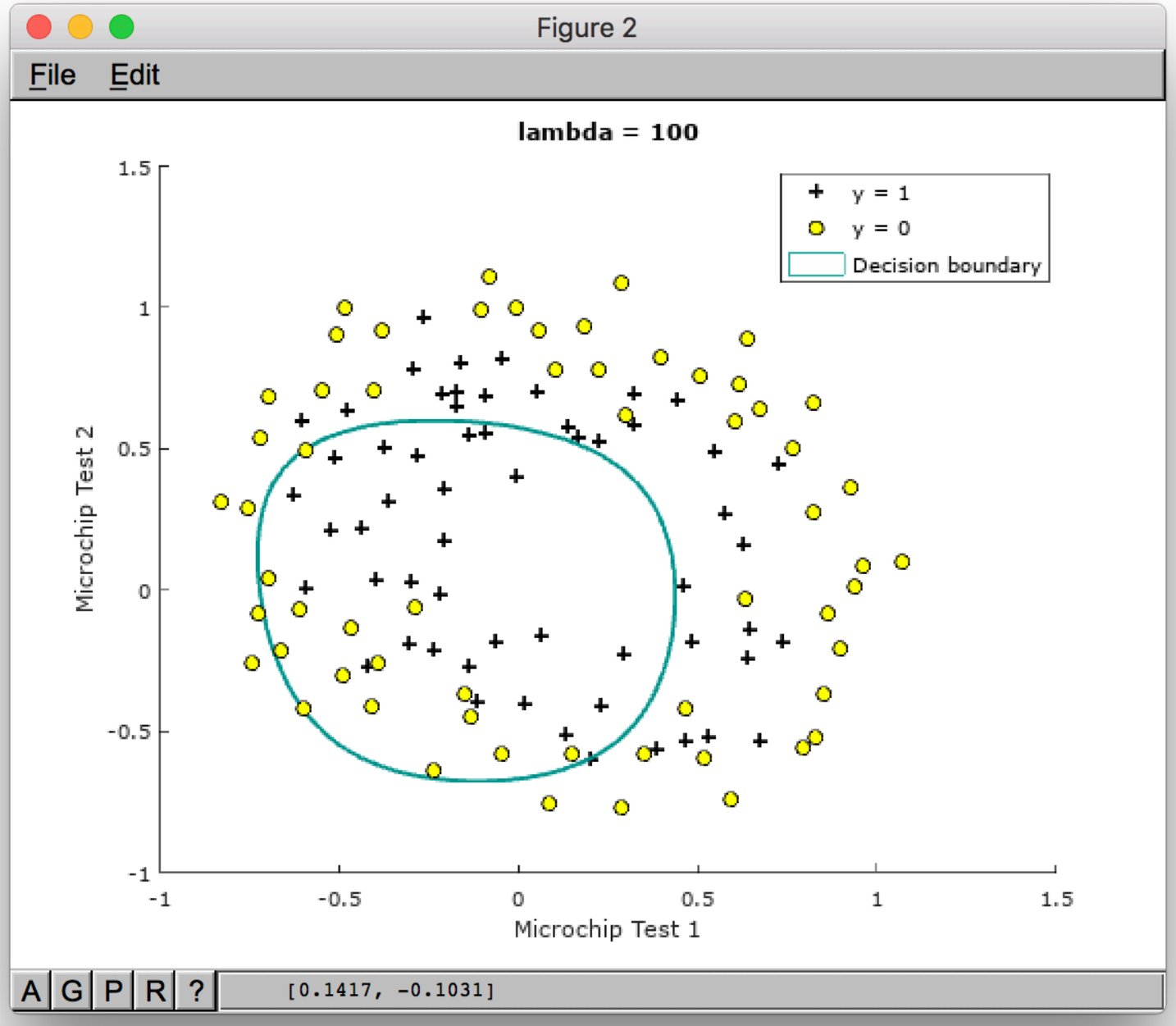


Figure 2
File Edit
lambda = 100
y = 1
y = 0
Decision boundary
Microchip Test 2
Microchip Test 1
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5
A G P R ?
[0.1417, -0.1031]

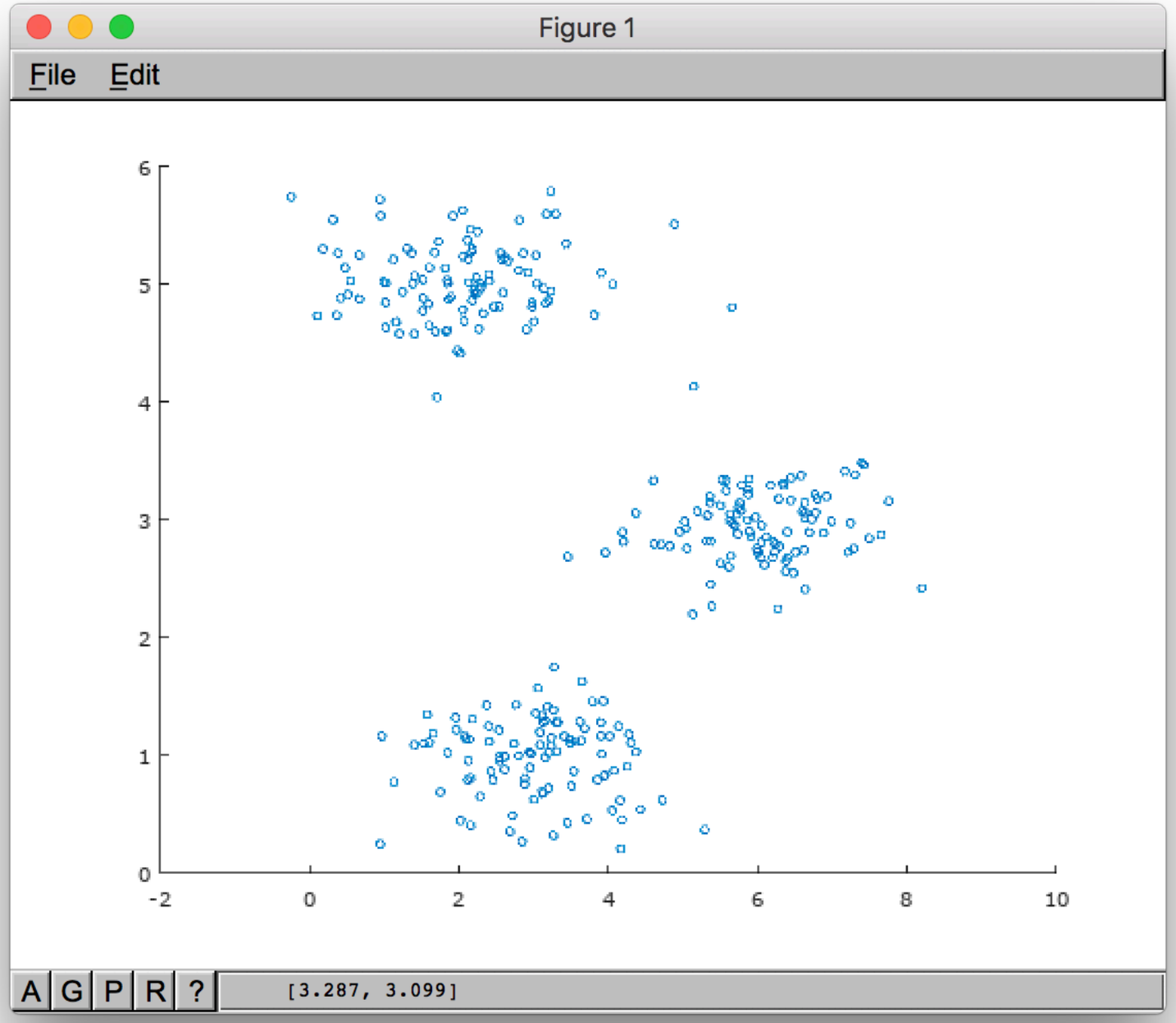


Figure 1
File Edit
6
5
4
3
2
1
0
-2
0
2
4
6
8
10
A G P R ?
[3.287, 3.099]

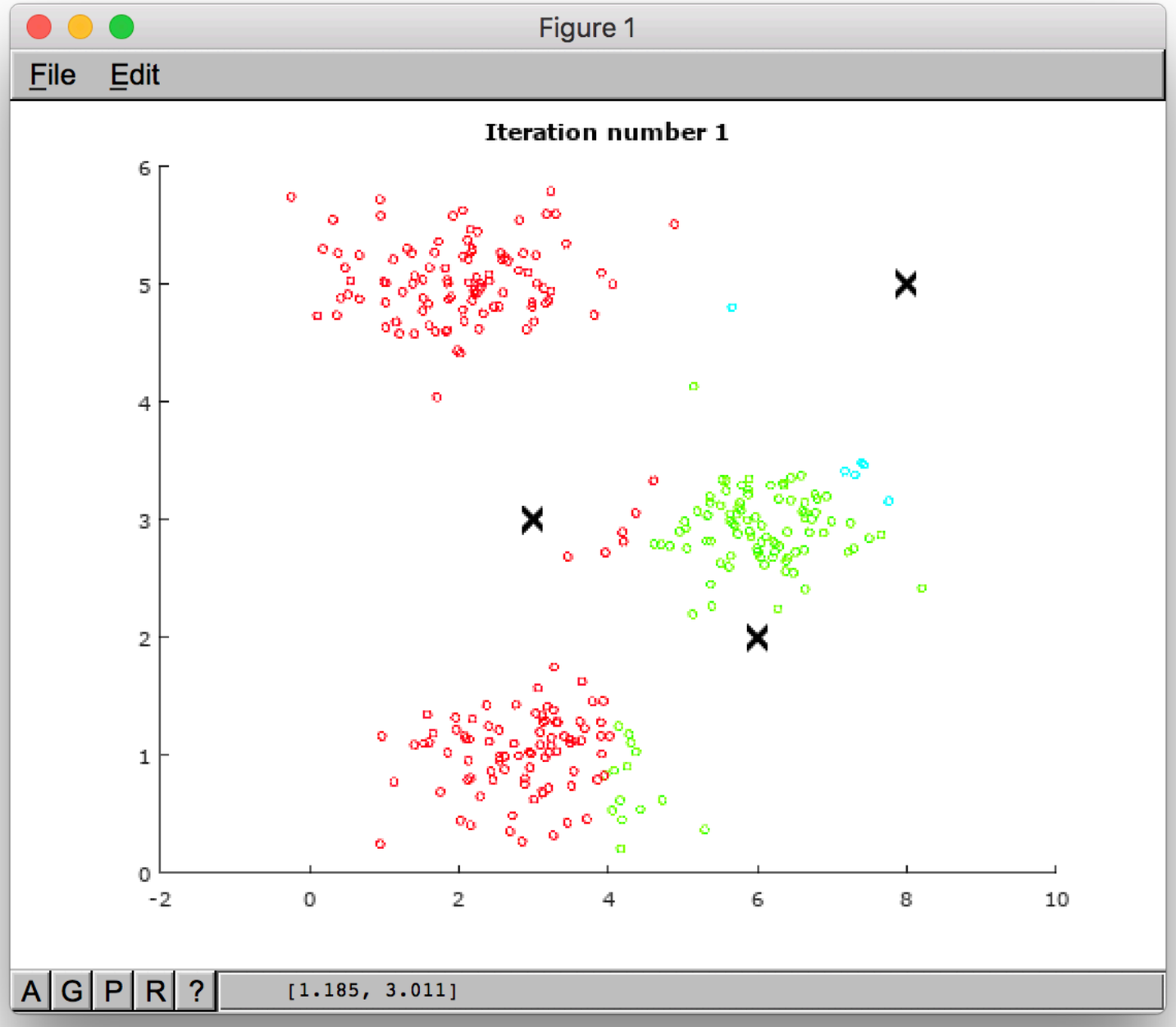


Figure 1
File Edit
Iteration number 1
6
5
4
3
2
1
0
-2
0
2
4
6
8
10
A G P R ?
[1.185, 3.011]

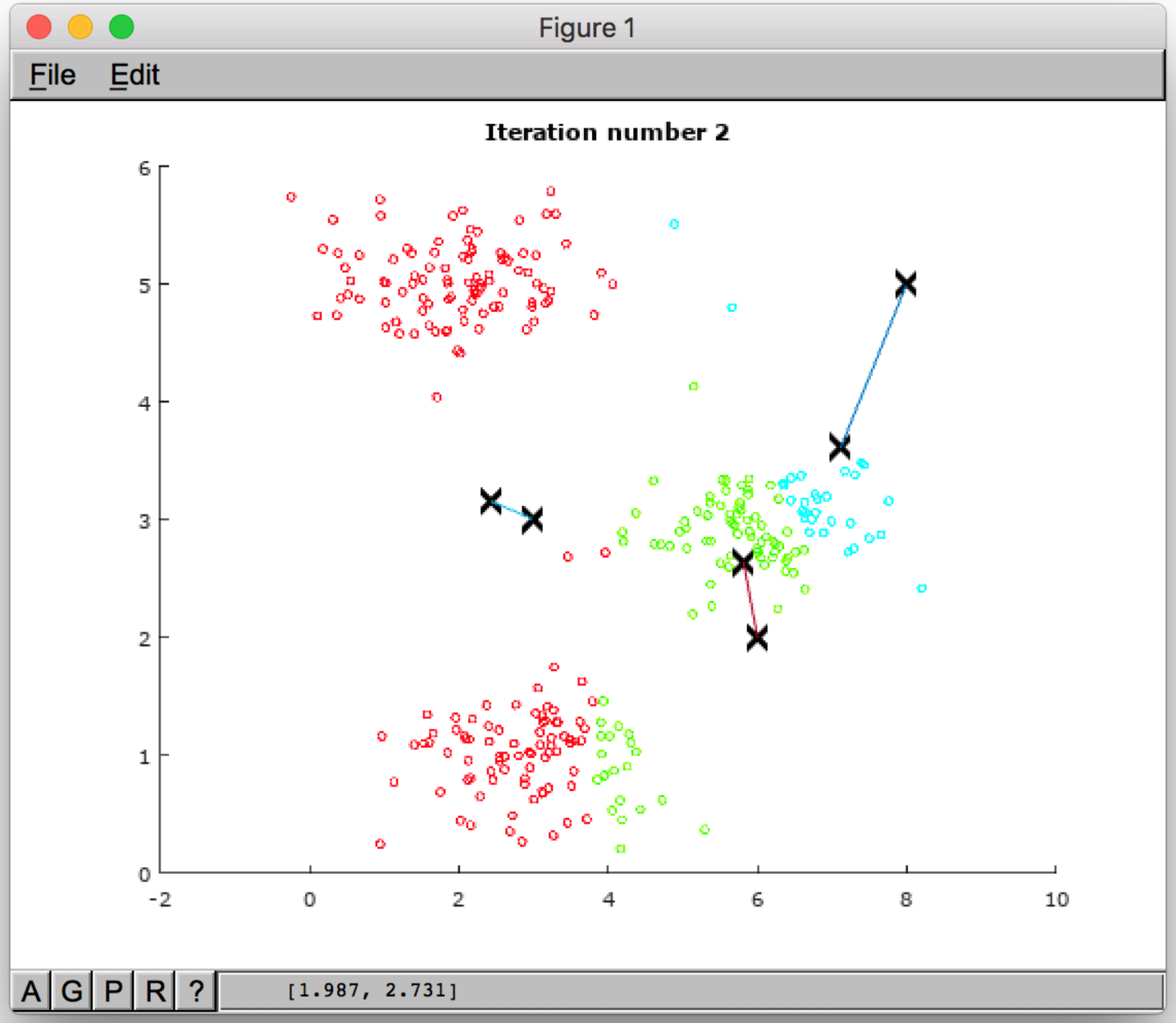


Figure 1
File Edit
Iteration number 2
6
5
4
3
2
1
0
-2
0
2
4
6
8
10
A G P R ?
[1.987, 2.731]

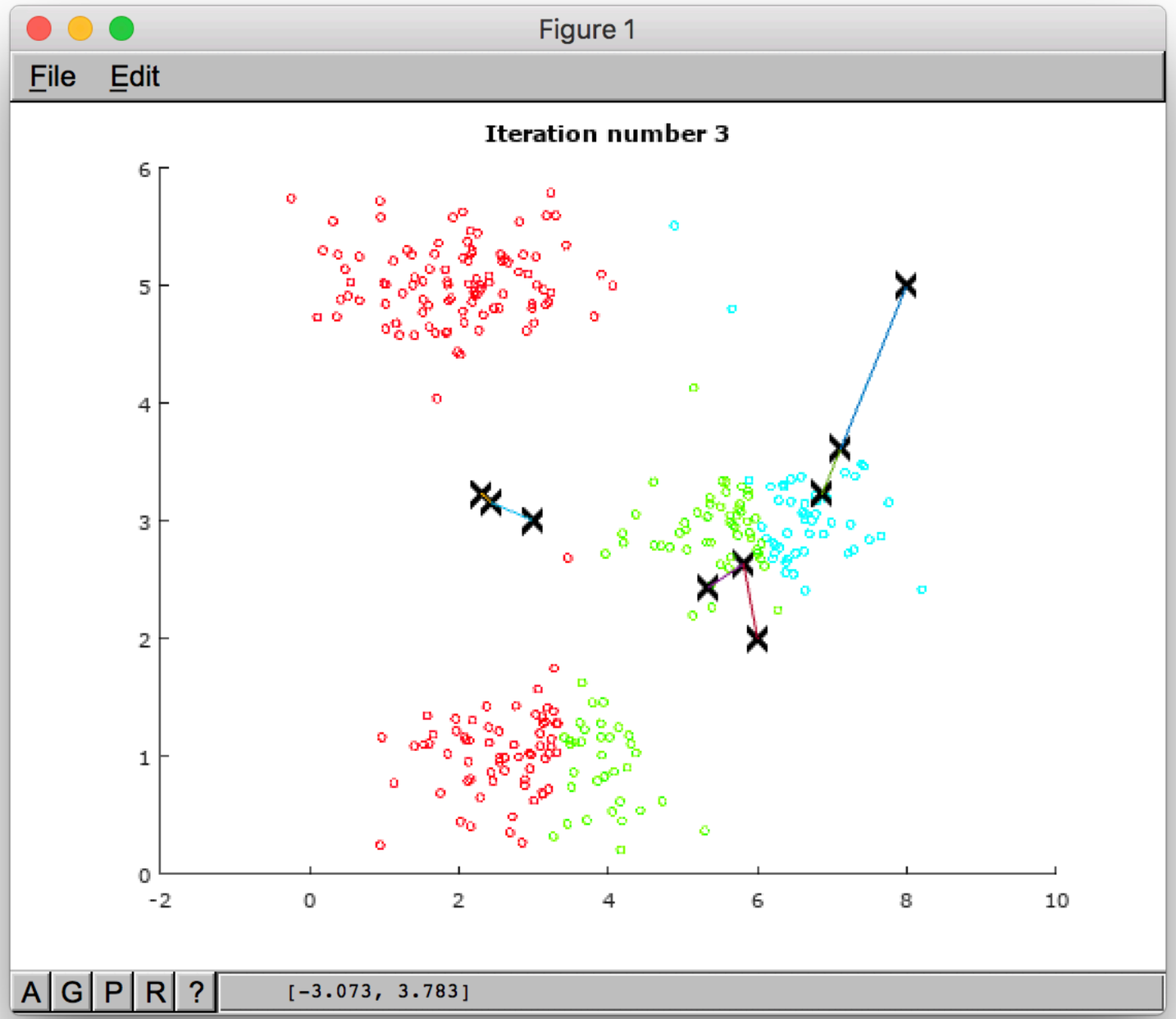


Figure 1
File Edit
Iteration number 3
6
5
4
3
2
1
0
-2
0
2
4
6
8
10
A G P R ?
[-3.073, 3.783]

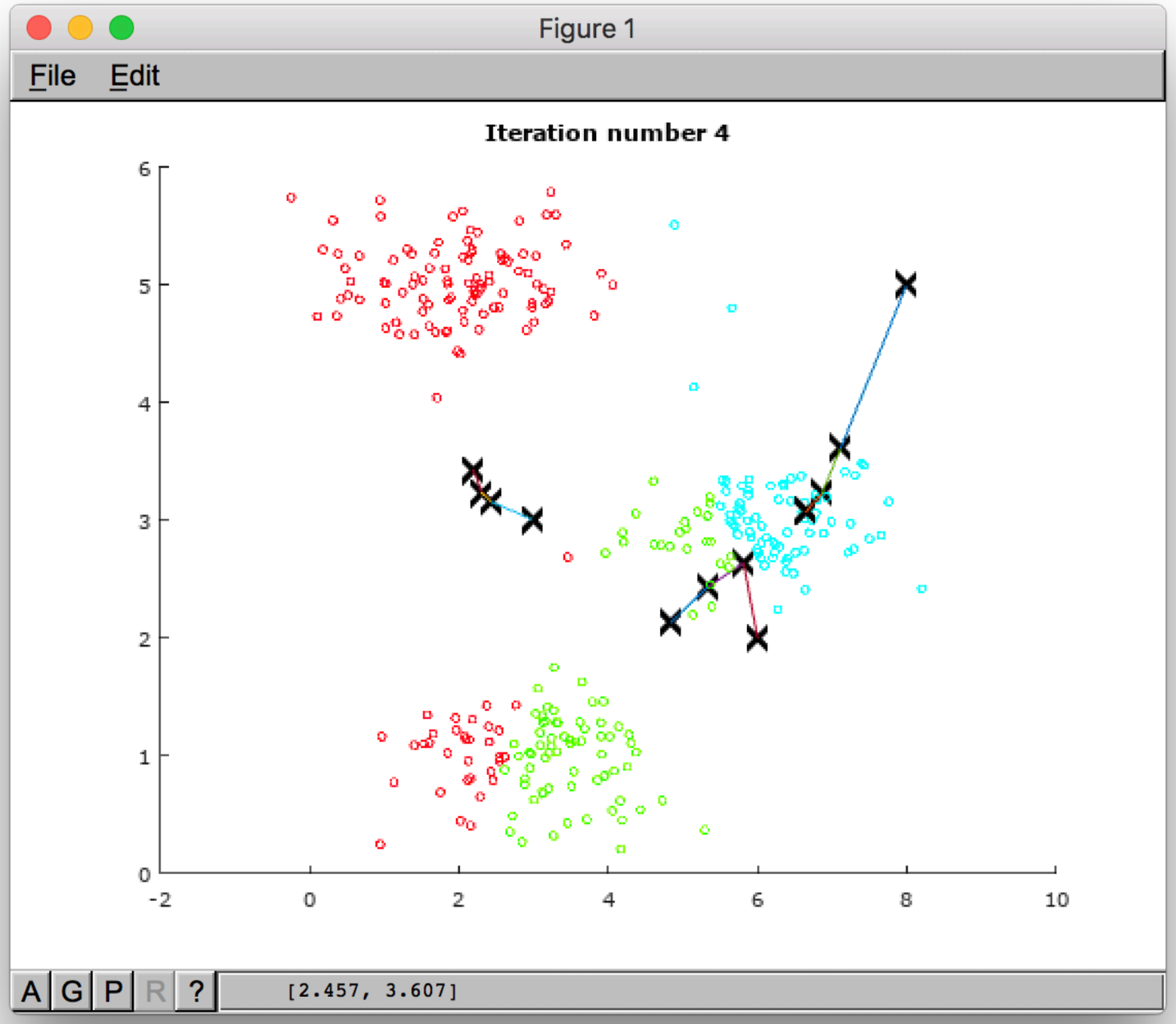


Figure 1
File Edit
Iteration number 4
6
5
4
3
2
1
0
-2
0
2
4
6
8
10
A G P R ?
[2.457, 3.607]

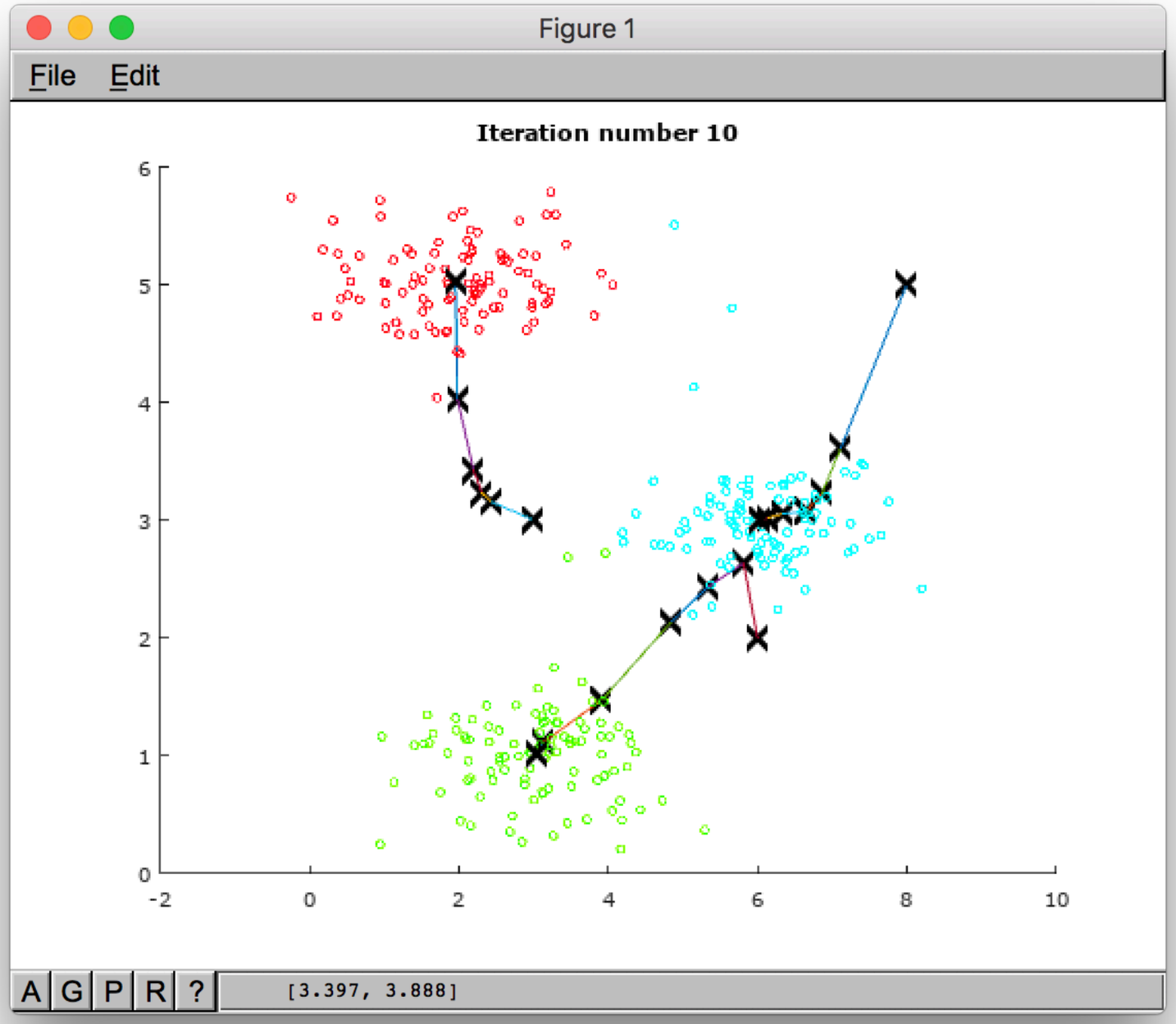


Figure 1
File Edit
Iteration number 10
6
5
4
3
2
1
0
-2
0
2
4
6
8
10
A G P R ?
[3.397, 3.888]

K-means

```
% Initialize centroids
centroids = kMeansInitCentroids(X, K);
for iter = 1:iterations
    % Cluster assignment step: Assign each data point to the
    % closest centroid. idx(i) corresponds to cˆ(i), the index
    % of the centroid assigned to example i
    idx = findClosestCentroids(X, centroids);

    % Move centroid step: Compute means based on centroid
    % assignments
    centroids = computeMeans(X, idx, K);
end
```

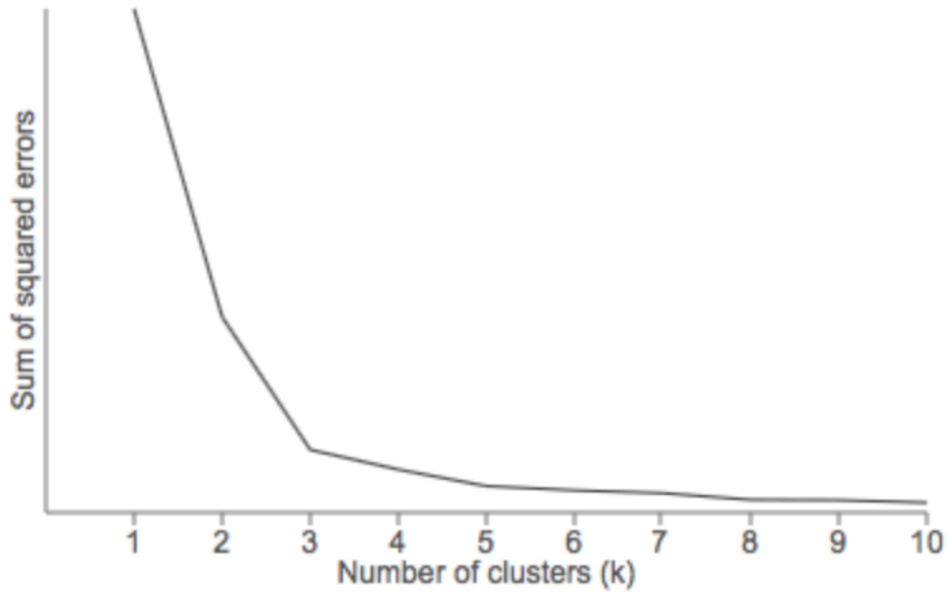
find closest centroids

$$c^{(i)} := j \quad \text{that minimizes} \quad ||x^{(i)} - \mu_j||^2,$$

compute means

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x^{(i)}$$

elbow method



I LOVE IT WHEN

YOU CALL ME BIG DATA

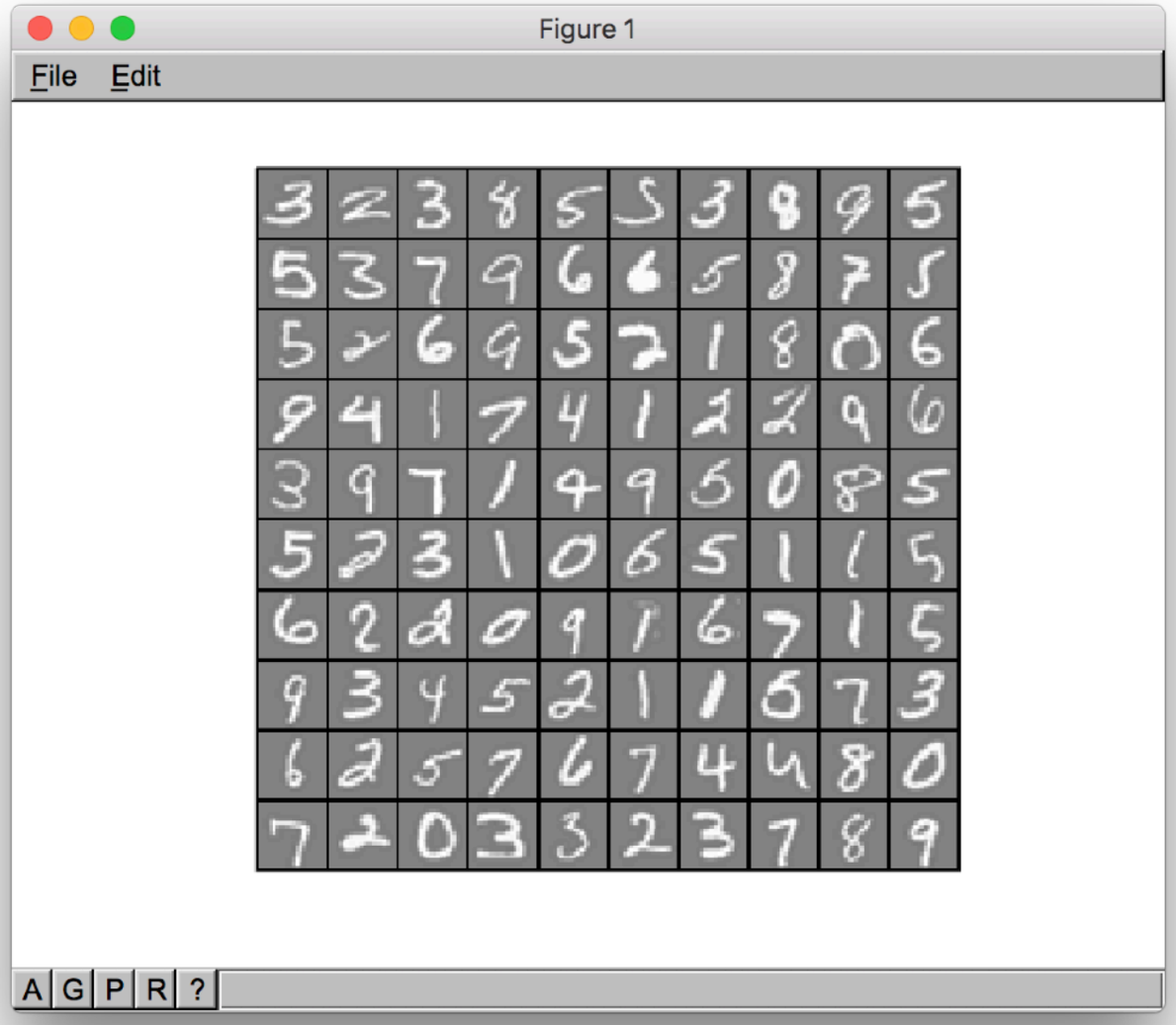
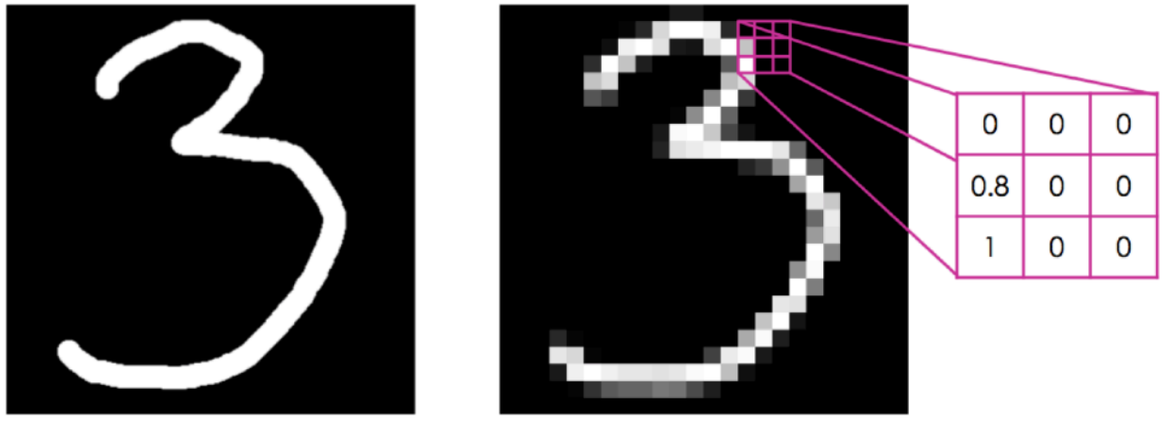
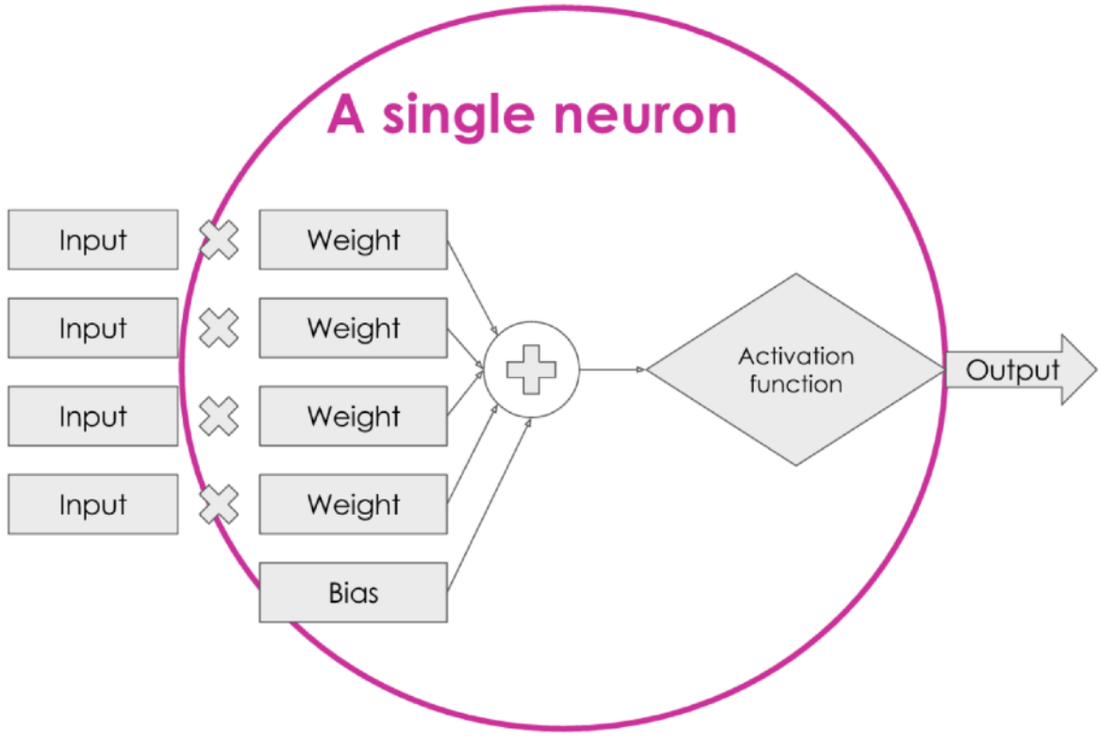
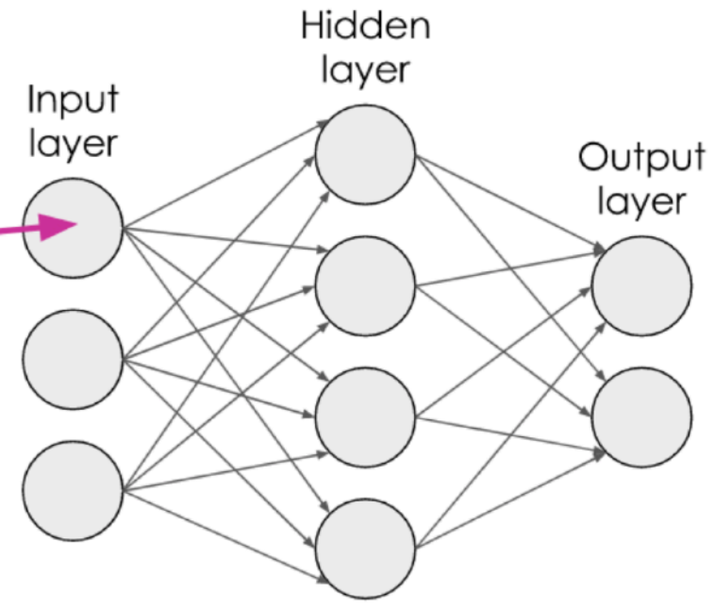


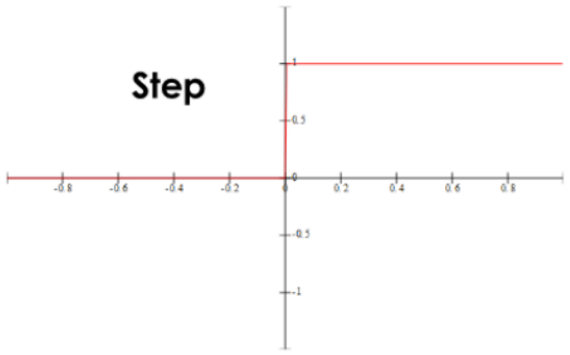
Figure 1
File Edit
A G P R ?


0 0 0
0.8 0 0
1 0 0

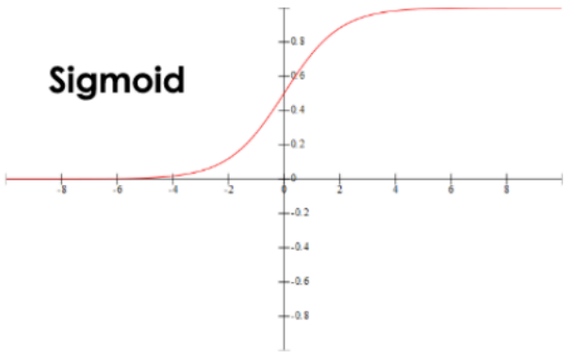


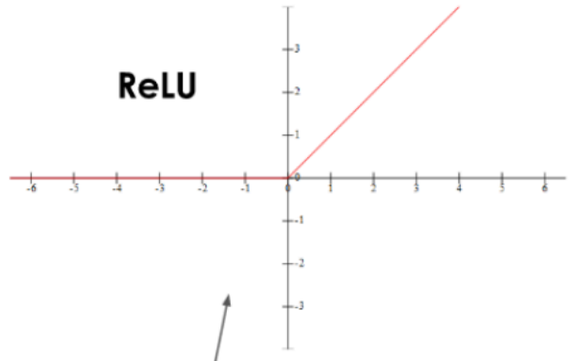
Each of these blobs is a neuron





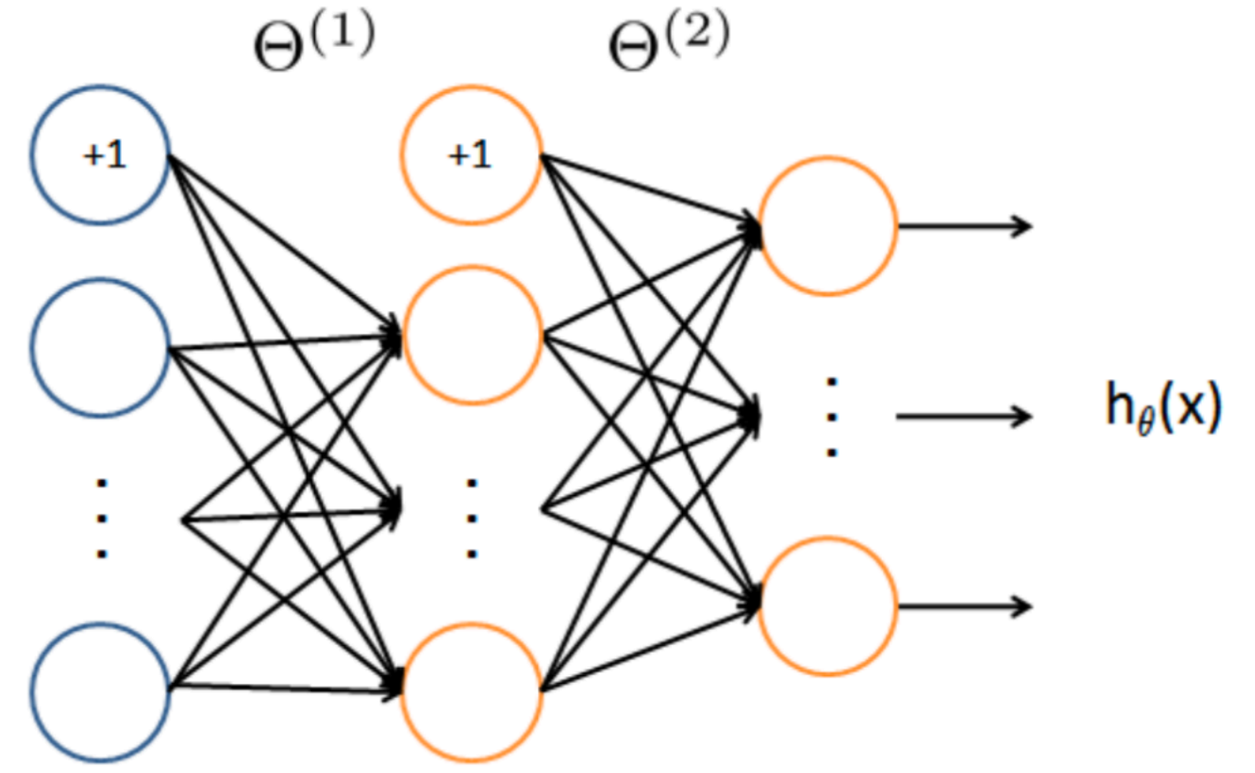
Popular historically





Popular these days

Neural network model



$a^{(1)} = x$
(add $a_0^{(1)}$)

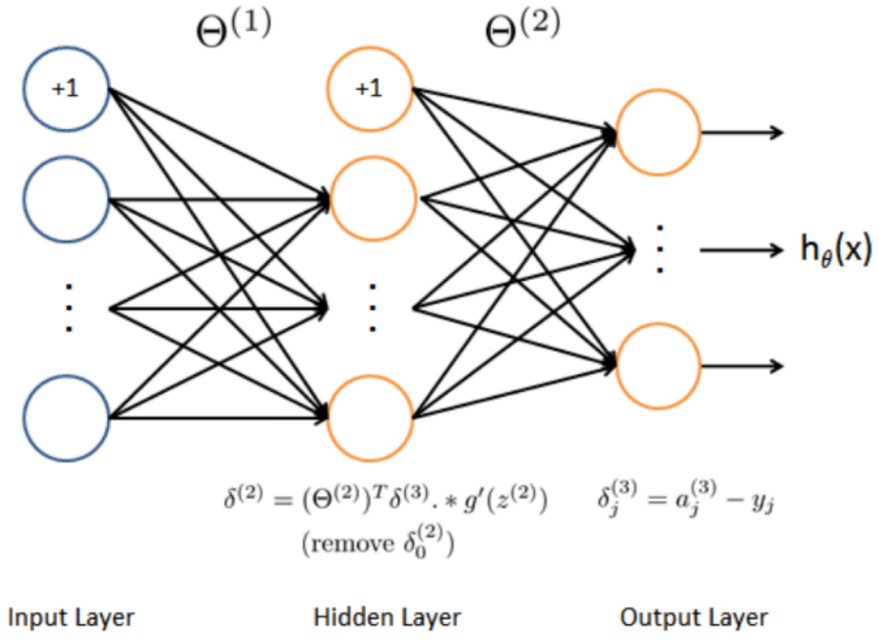
$z^{(2)} = \Theta^{(1)} a^{(1)}$
$a^{(2)} = g(z^{(2)})$
(add $a_0^{(2)}$)

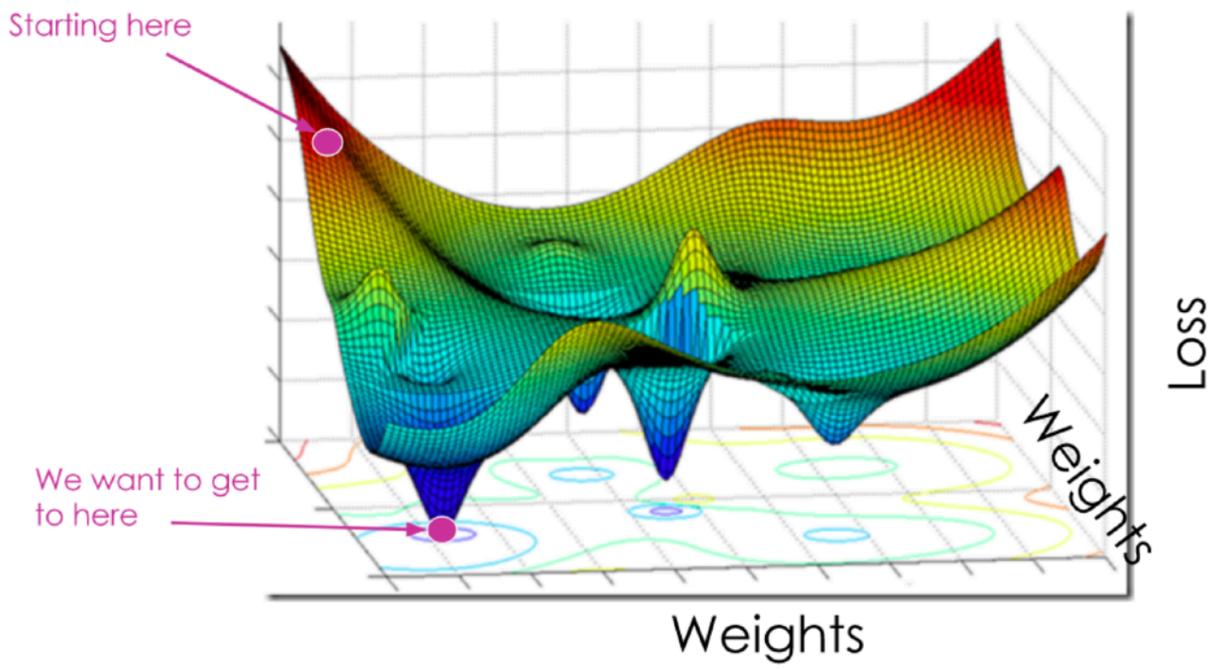
$z^{(3)} = \Theta^{(2)} a^{(2)}$
$a^{(3)} = g(z^{(3)}) = h_\theta(x)$

Input Layer | Hidden Layer | Output Layer

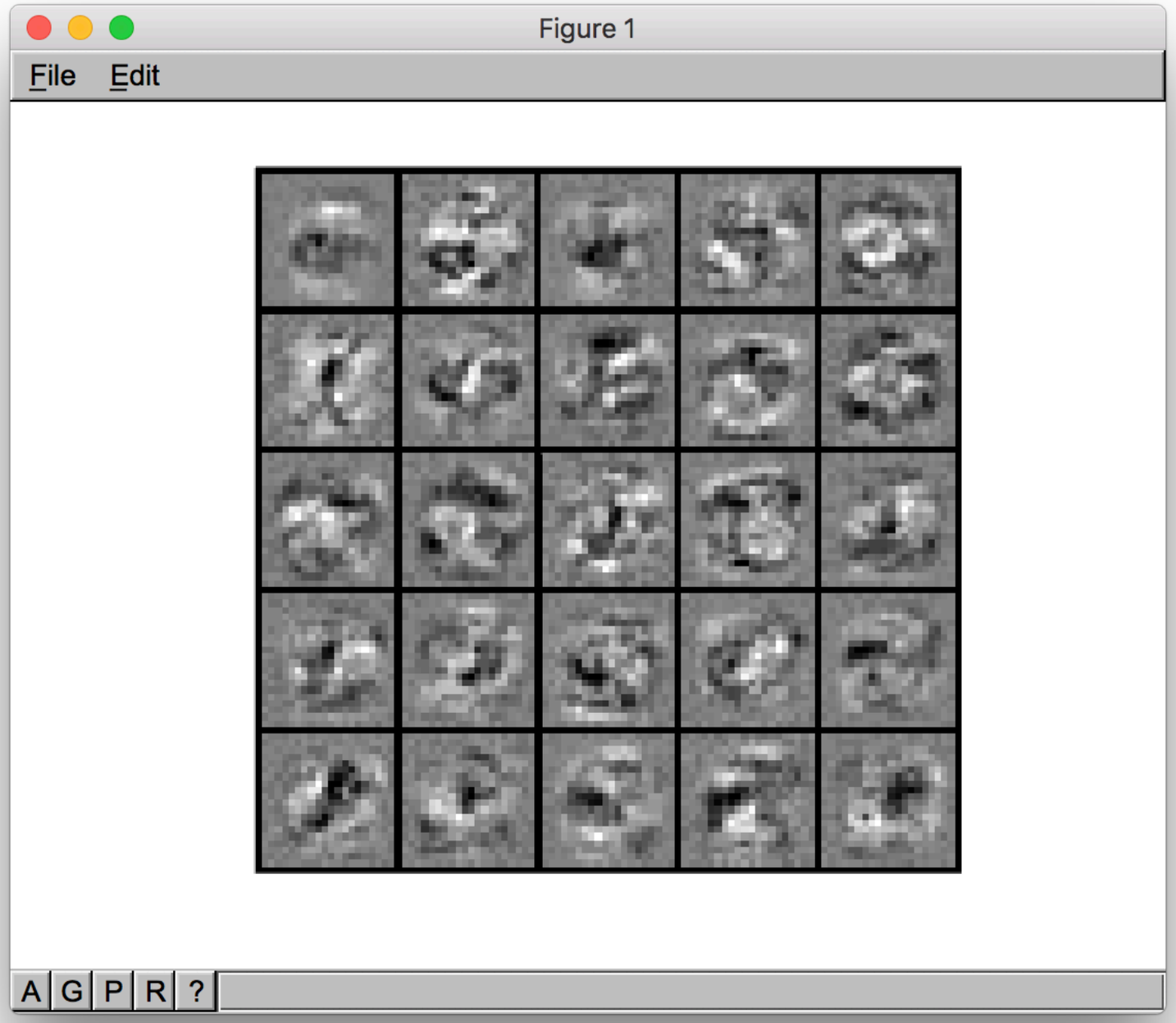
Training a neural network

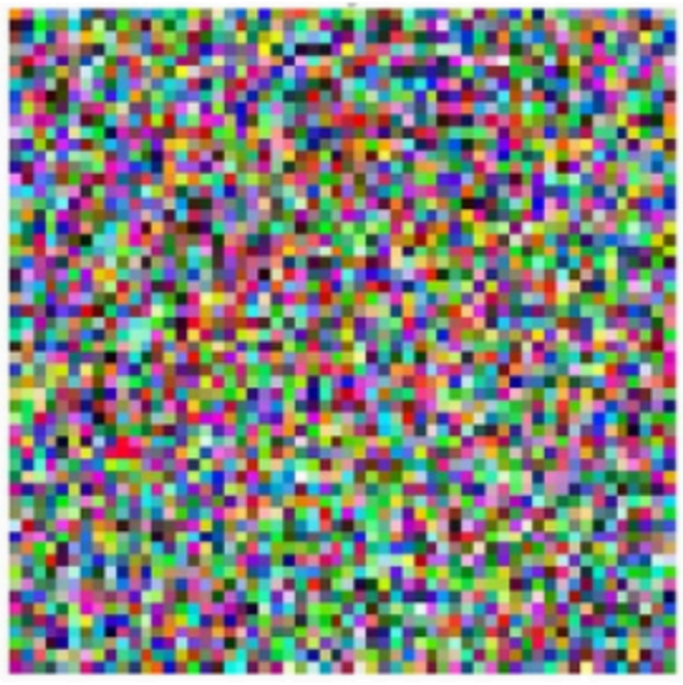
- Randomly initialise the network weights and biases
- For every piece of training data, feed it into the network
- Check whether the network gets it right
- If not, how wrong was it? Or, how right was it?
- Nudge the weights a little to increase the probability of a correct answer
- Repeat



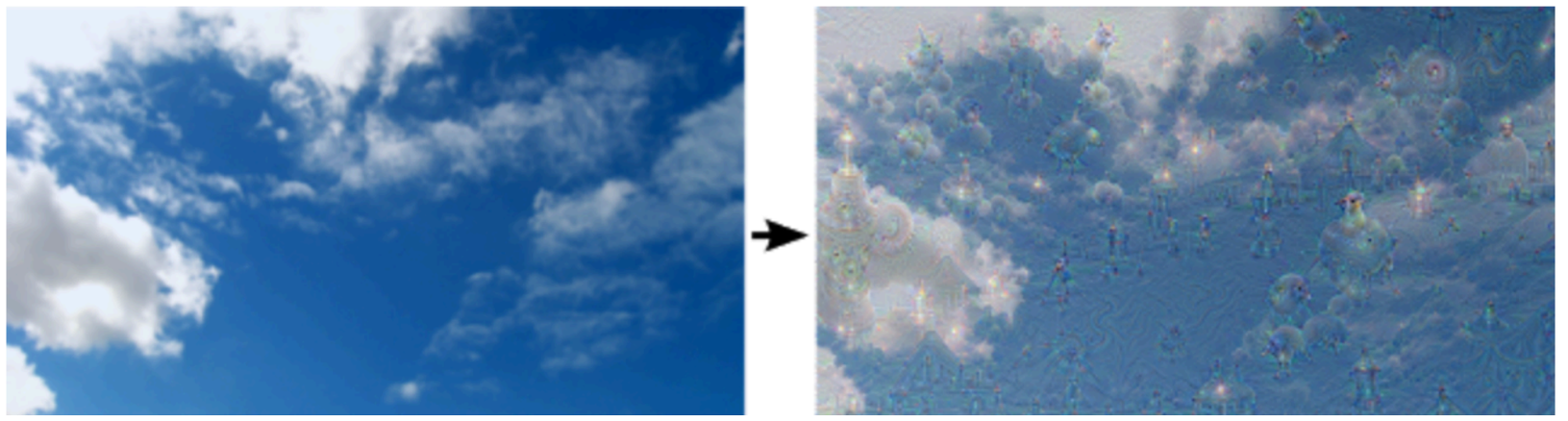


50 iterations, 96.1% accuracy



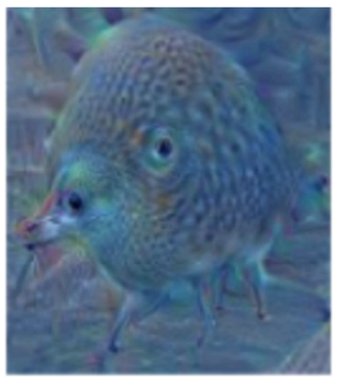


100.0% Goldfish



"Admiral Dog!"

"The Pig-Snail"



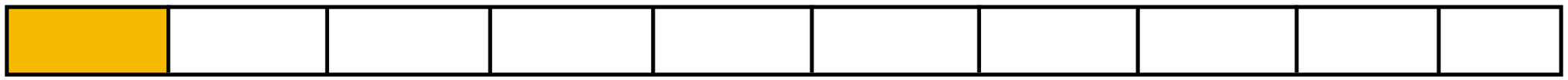
"The Camel-Bird"

"The Dog-Fish"

Neural net "dreams"— generated purely from random noise, using a network trained on places by MIT Computer Science and AI Laboratory. See our Inceptionism gallery for hi-res versions of the images above and more (Images marked "Places205-GoogLeNet" were made using this network).

Model evaluation

K-fold Cross Validation



Run k testing experiments

- pick testing set
- train
- test on testing set

Average test results

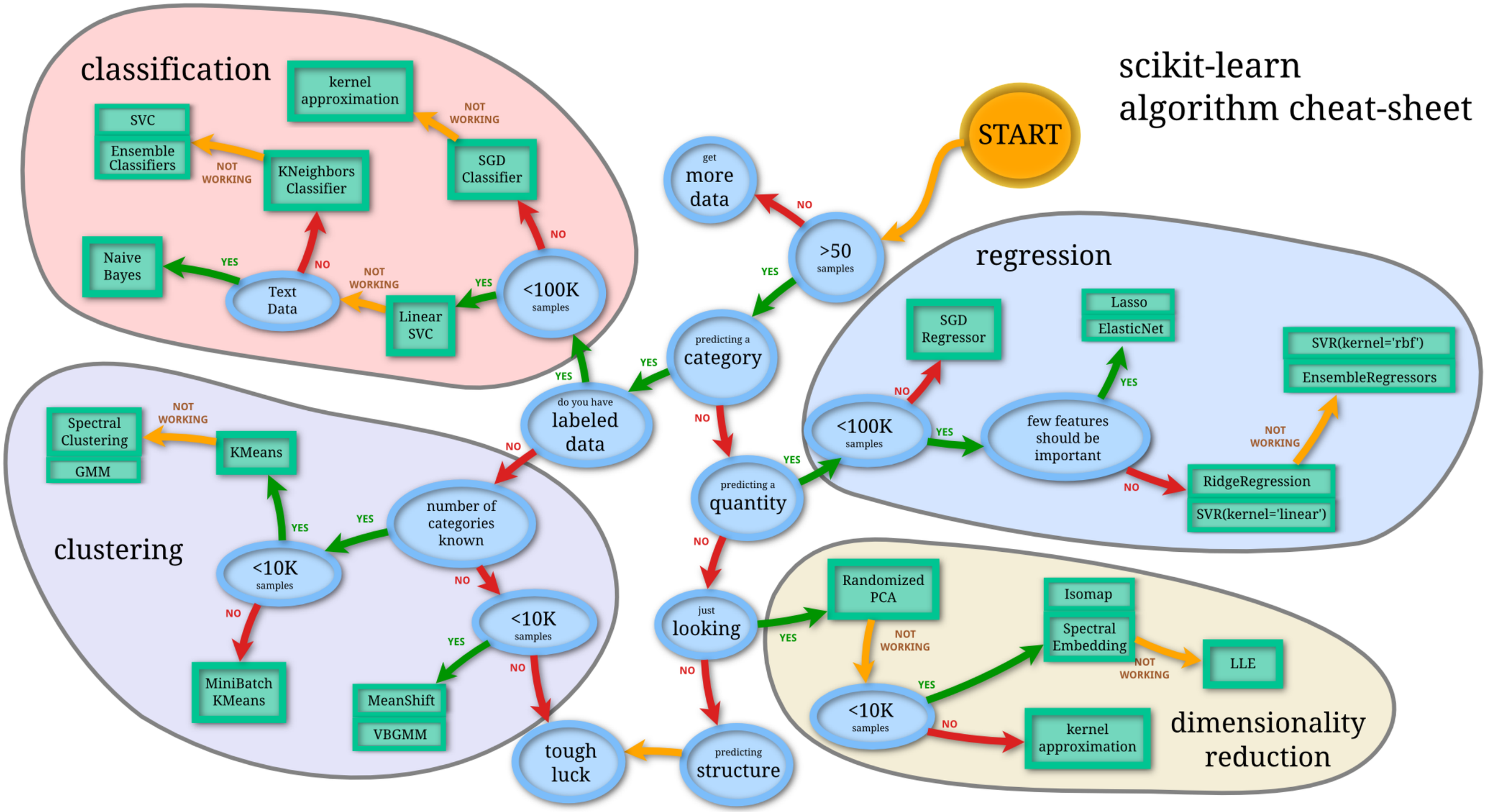
Debugging a learning algorithm

- Get more training examples
- Try smaller set of features
- Try getting additional features
- Try adding polynomial features
- Try decreasing regularisation parameter
- Try increasing regularisation parameter

1. Gathering data
2. Data preparation
3. Choosing a model
4. Training
5. Evaluation
6. Parameter tuning
7. Prediction

Repeat

scikit-learn algorithm cheat-sheet



https://www.coursera.org/learn/machine-learning

http://cs229.stanford.edu/syllabus.html

https://www.coursera.org/learn/neural-networks

https://www.coursera.org/specializations/deep-learning

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/

https://medium.com/manchester-futurists/demystifying-deep-neural-nets-efb726eae941

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

http://setosa.io/ev/

https://dataskeptic.com

http://lineardigressions.com