Στατιστική Μοντελοποίηση και Αναγνώριση Προτύπων (ΤΗΛ311)

Αναφορά 2ης Σειράς Ασκήσεων

Ανδρεαδάκης Αντώνης 2013030059

1. Στην άσκηση αυτή, ασχοληθήκαμε με την αναλυτική εύρεση της κλίσης σε ένα σύνολο m δεδομένων $\{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots, (\boldsymbol{x}^{(m)}, y^{(m)})\}$, όπου $\boldsymbol{x}^{(i)} \in \mathbb{R}^{n\times 1}$ είναι τα διανύσματα χαρακτηριστικών και $y^{(i)} \in \{0,1\}$ ορίζουν την κλάση κάθε δείγματος (labels).

A.

We know calculate the slope of the loss function $J(\theta)$ w.r.t the learnable parameters θ of our learner.

$$ln\left(h_\theta(x^{(i)})\right) = ln(\frac{1}{1+e^{-\theta^T x^{(i)}}}) = -ln(1+e^{-\theta^T x^{(i)}})$$

$$ln\left(1-h_\theta(x^{(i)})\right) = ln(1-\frac{1}{1+e^{-\theta^T x(i)}}) = ln(\frac{e^{-\theta^T x^{(i)}}}{1+e^{-\theta^T x^{(i)}}}) =$$
$$= ln(e^{-\theta^T x^{(i)}}) - ln(1+e^{-\theta^T x^{(i)}}) = -\theta^T x^{(i)} - ln(1+e^{-\theta^T x^{(i)}})$$

By substitution we get that

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} ln\left(1+e^{-\theta^T x^{(i)}}\right) + (1-y^{(i)})\left(\theta^T x^{(i)} + ln(1+e^{-\theta^T x^{(i)}})\right)\right) =$$

$$\frac{1}{m}\sum_{i=1}^{m}\left(ln\left(e^{\theta^T x^{(i)}}(1+e^{-\theta^T x^{(i)}})\right) - y^{(i)}\theta^T x^{(i)}\right) =$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(ln(1+e^{\theta^T x^{(i)}}) - y^{(i)}\theta^T x^{(i)}\right)$$

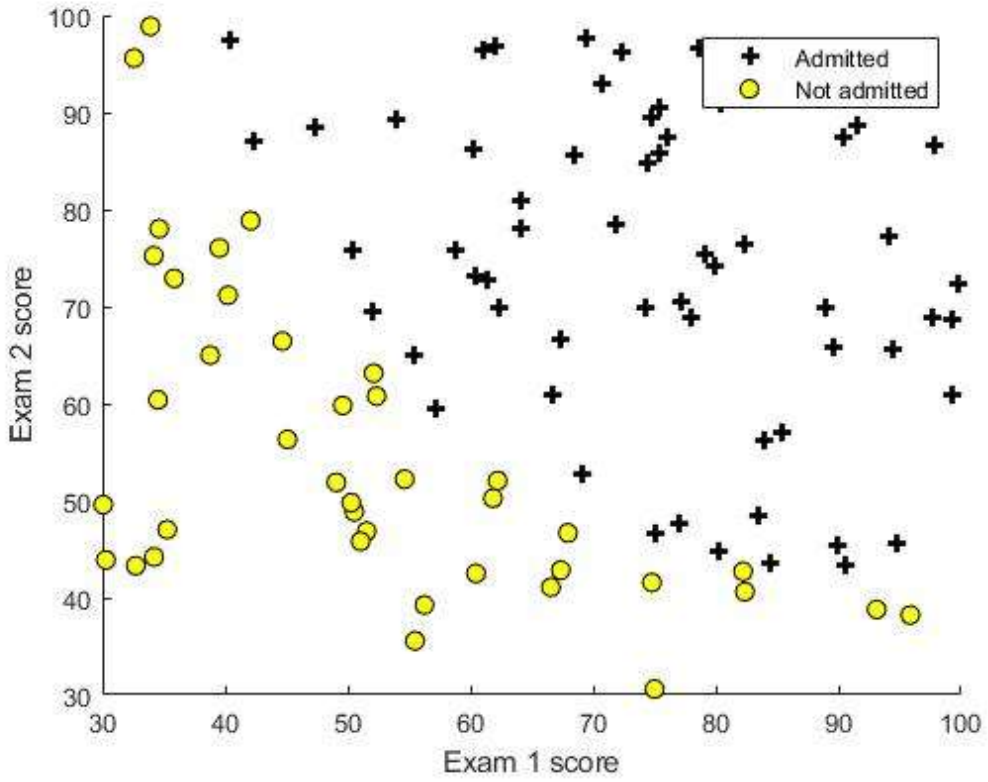
By taking the derivative of the above relation w.r.t the learnable parameters $\theta \in \mathbb{R}^n$ we get that

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{x^{(i)}e^{\theta^T x^{(i)}}}{1+e^{\theta^T x^{(i)}}} - y^{(i)}x^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m}(\frac{1}{1+e^{-\theta^T x^{(i)}}} - y^{(i)})x^{(i)}.$$

Proving that

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

Β. Χρησιμοποιώντας τη λογιστική παλινδρόμηση, για να προβλέψουμε αν ένας μαθητής γίνει δεκτός σε ένα πανεπιστήμιο , με βάση τους βαθμούς του σε 2 εξετάσεις, πήραμε τα παρακάτω αποτελέσματα: Αρχικά προβάλλουμε τα δεδομένα:

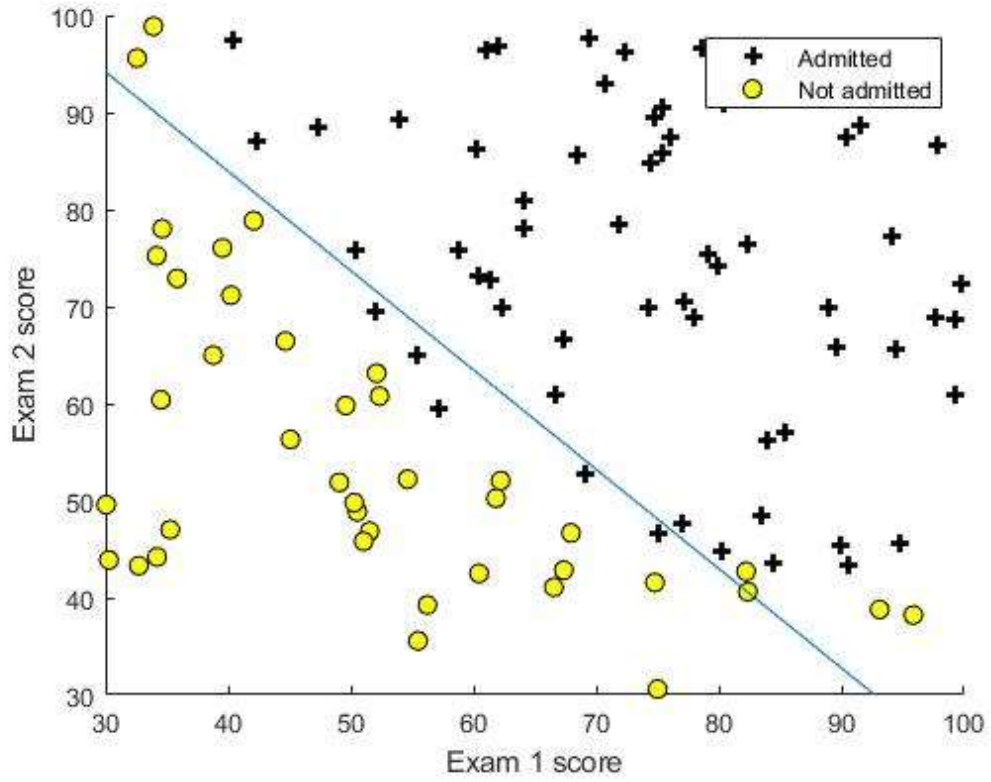


Υλοποιώντας τις συναρτήσεις που μας ζητήθηκαν και εκτελώντας τον κώδικα, είχαμε τα εξής αποτελέσματα:

Κόστος και κλίση:

```
Cost at initial theta (zeros): 0.693147
Gradient at initial theta (zeros):
 -0.100000
 -12.009217
 -11.262842
```

Γράφημα από την plotDecisionBoundary:



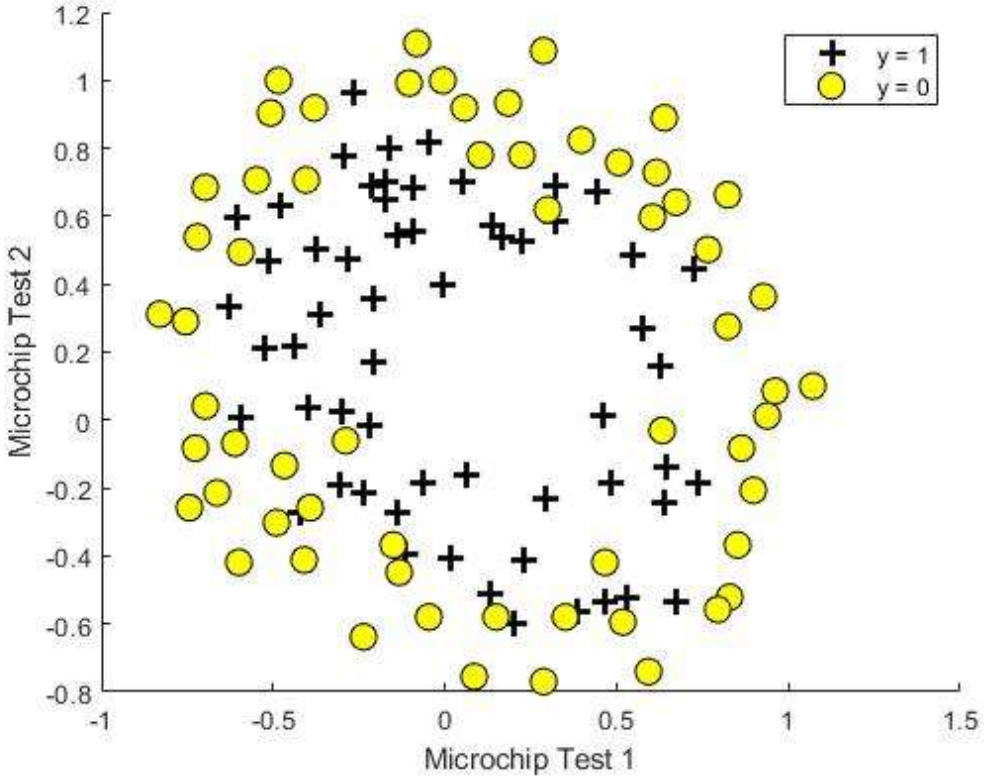
Με βελτιστοποίηση:

```
Cost at theta found by fminunc: 0.203498
theta:
 -25.161343
 0.206232
 0.201472
```

Πρόβλεψη αν ο φοιτητής γίνει δεκτός:

```
Program paused. Press enter to continue.
For a student with scores 45 and 85, we predict an admission probability of 0.776291

Train Accuracy: 89.000000
```

2. Εδώ εφαρμόσαμε ομαλοποιημένη λογιστική παλινδρόμηση, για την πρόβλεψη ελέγχου ποιότητας στα μικροτσιπ μιας μονάδας κατασκευής. Αρχικά προβάλλουμε τα δεδομένα:



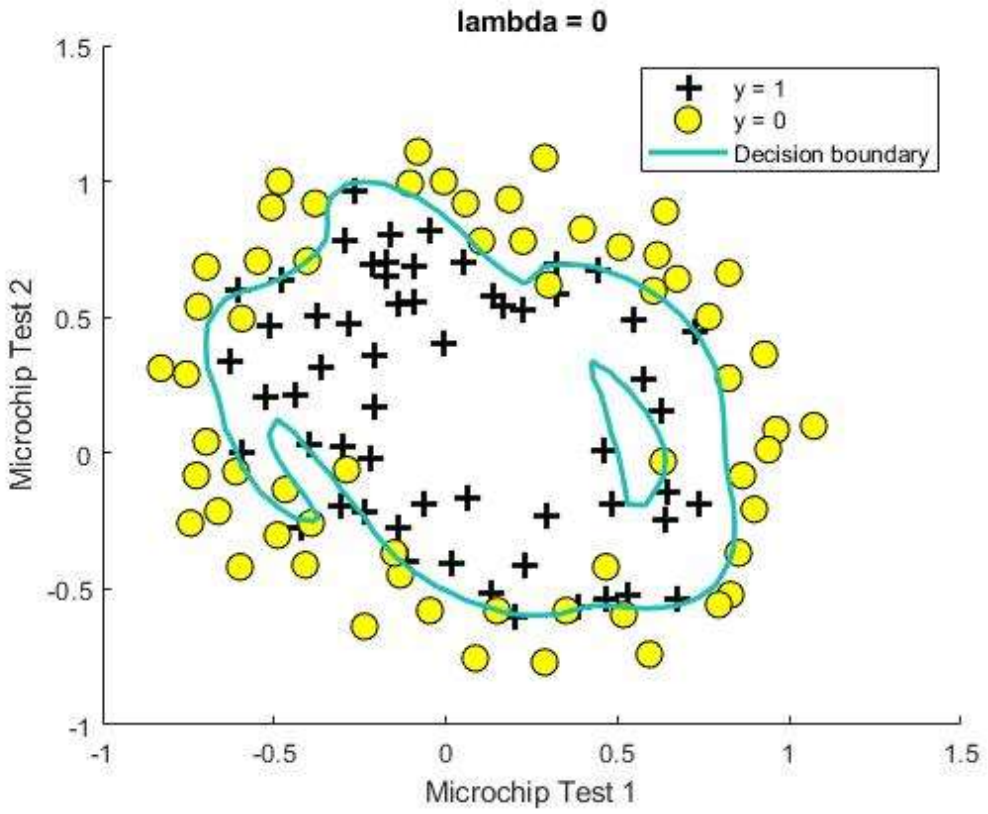
Με μαύρο απεικονίζονται όσα έχουν ελεγχθεί και είναι ποιοτικά, ενώ με κίτρινο όσα δεν είναι ποιοτικά.

Προχωρώντας τον κώδικα έχουμε:

```
Cost at initial theta with lambda = 1 (zeros): 0.693147
Expected cost (approx): 0.693
Gradient at initial theta (zeros) - first five values only:
    0.008475
    0.018788
    0.000078
    0.050345
    0.011501
Expected gradients (approx) - first five values only:
    0.0085
    0.0188
    0.0001
    0.0503
    0.0115
```

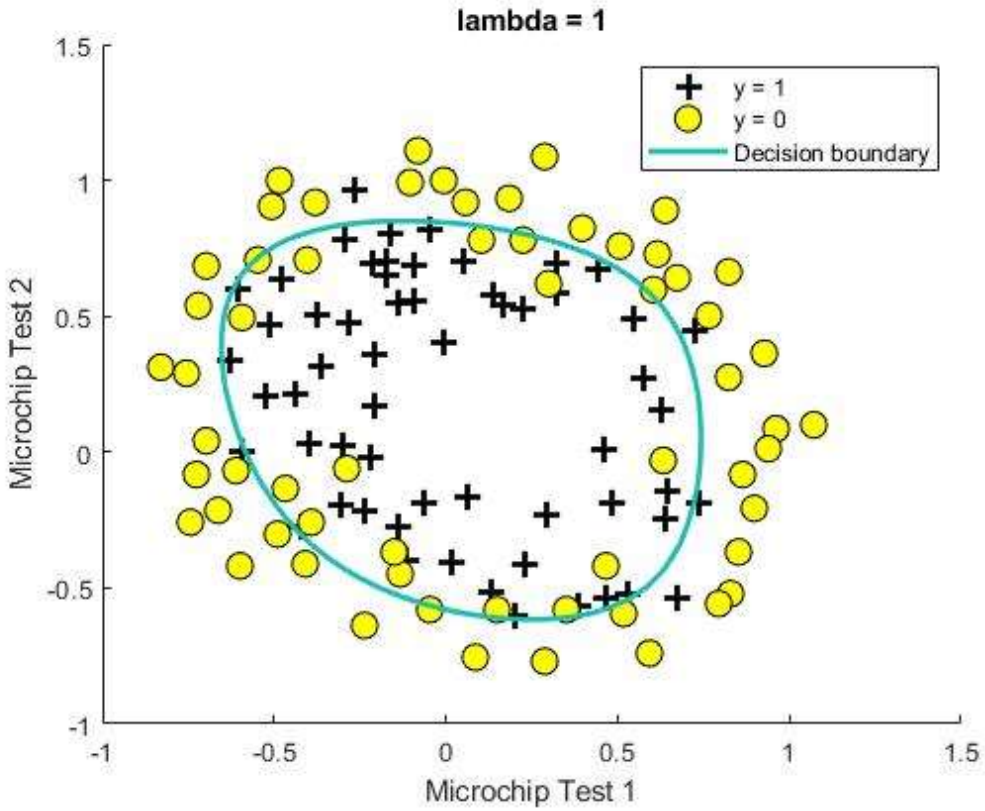
```
Cost at test theta (with lambda = 10): 3.164509
Expected cost (approx): 3.16
Gradient at test theta - first five values only:
    0.346045
    0.161352
    0.194796
    0.226863
    0.092186
Expected gradients (approx) - first five values only:
    0.3460
    0.1614
    0.1948
    0.2269
    0.0922
```

Και τα σύνορα απόφασης για διάφορες τιμές του λ:



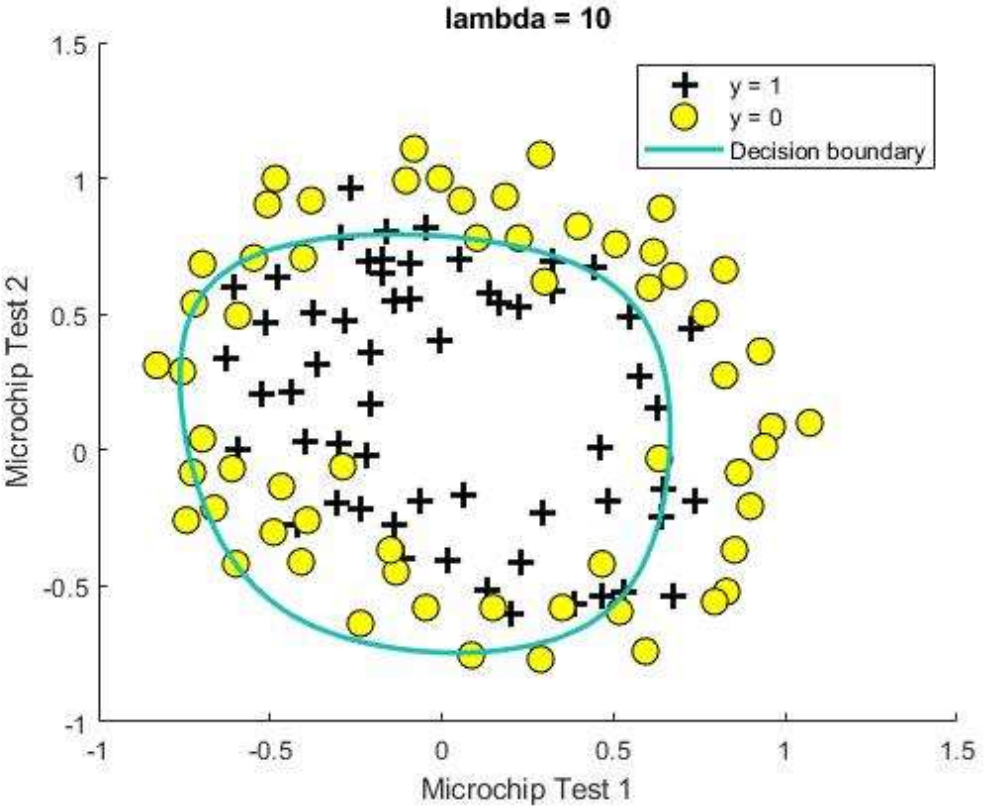

```
Train Accuracy: 88.983051
```

```
Expected accuracy (with lambda = 0): 89(approx)
```



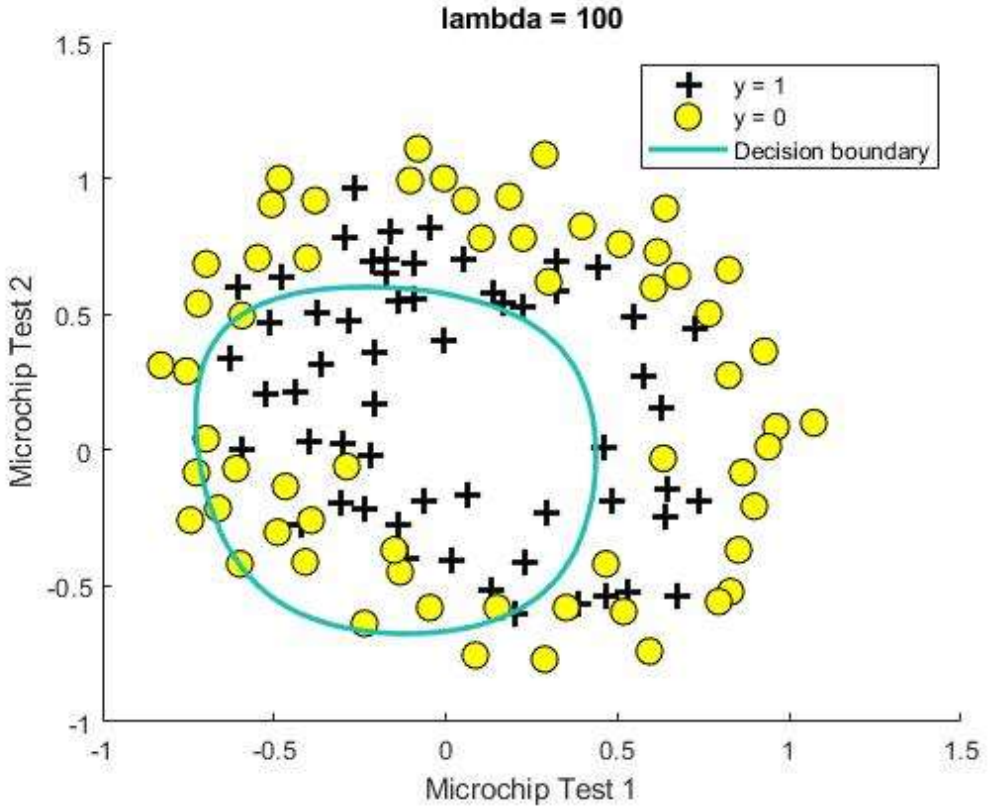

```
Train Accuracy: 83.050847
```

```
Expected accuracy (with lambda = 1): 83.1(approx)
```




```
Train Accuracy: 74.576271
```

```
Expected accuracy (with lambda = 10): 74.6(approx)
```




```
Train Accuracy: 61.016949
```

```
Expected accuracy (with lambda = 1): 61(approx)
```

3. Ο M-L.E. είναι η λύση του ακόλουθου προβλήματος μεγιστοποίησης: $\hat{\lambda} = \arg\max_{\lambda} l(\lambda; x_1, \ldots, x_n)$

Η 1η συνθήκη για μέγιστο είναι: $\frac{d}{d\lambda} l(\lambda; x_1, \ldots, x_n) = 0$

Η 1η παράγωγος της πιθανότητας log-likelihood, σε σχέση με την παράμετρο λ είναι:

$$\frac{d}{d\lambda} l(\lambda; x_1, \ldots, x_n) = \frac{d}{d\lambda}\left(-n\lambda - \sum_{j=1}^{n} \ln(x_j!) + \ln(\lambda) \sum_{j=1}^{n} x_j \right)$$

$$= -n + \frac{1}{\lambda} \sum_{j=1}^{n} x_j$$

Η παράγωγος της πιθανότητας πρέπει να είναι μηδέν, σύμφωνα με την συνθήκη που αναφέρουμε παραπάνω. Οπότε τελικά έχουμε:

$$\lambda = \frac{1}{n} \sum_{j=1}^{n} x_j$$

Μια πιο ακριβής λύση είναι η εξής:

Suppose we have n i.i.d samples $D = \{x_1, ..., x_n\}$ that are generated from a Poisson distribution that is parameterized by λ.

$$p(x|\lambda)\frac{\lambda^x e^{-\lambda}}{x!}, \quad x = 0, 1, 2, ... \quad , \quad \lambda > 0$$

We will attend to find the MLE of parameters λ. The likelihood function is

$$L_D(\lambda) = \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}.$$

The log-likelihood then is

$$l_D(\lambda) = log(\prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}) = \sum_{i=1}^{n} log(\frac{\lambda^{x_i} e^{-\lambda}}{x_i!}) =$$

$$= \sum_{i=1}^{n}(-\lambda + x_i log\lambda - logx_i!) = -n\lambda - \sum_{i=1}^{n} logx_i! + log\lambda \sum_{i=1}^{n} x_i.$$

The Maximum Likelihood Estimation of λ can be found by solving the convex optimization problem $\text{argmax}_\lambda(l_D(\lambda))$. (This optimization problem is convex because we are maximizing a concave function).

$$\nabla_{\lambda_{MLE}} l_D(\lambda) = 0 \Leftrightarrow -n + \frac{1}{\lambda_{MLE}} \sum_{i=1}^{n} x_i = 0$$

$$\Leftrightarrow \lambda_{MLE} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Therefore, λ_{MLE} is just the sample mean of the n observations in the sample.

4. Εδώ, υλοποιήσαμε τον naive Bayes ταξινομητή για αναγνώριση προτύπων. Τα ψηφία 0 έως 9 αντιστοιχούν σε 10 κλάσεις και κάθε δείγμα είναι αναπαράσταση μιας εικόνας με διαστάσεις 28x28. Στην περίπτωσή μας, υποθέτουμε ότι τα χαρακτηριστικά (εικονοστοιχεία-pixels) είναι ανεξάρτητα. Χρησιμοποιώντας την κατανομή Bernoulli, μοντελοποιούμε κάθε Pixel από κάθε κλάση.

A.

$$L_{D_{x^{y_i}}} = P(D_{x^{y_i}}|p^{y_i}) = \prod_{j=1}^{n} P(X = x_j^{y_i}) = (p^{y_i})^k (1-p^{y_i})^{k'}$$

where $k = \sum_{j=1}^{n} x_j^{y_i}$ and $k' = n - k$.

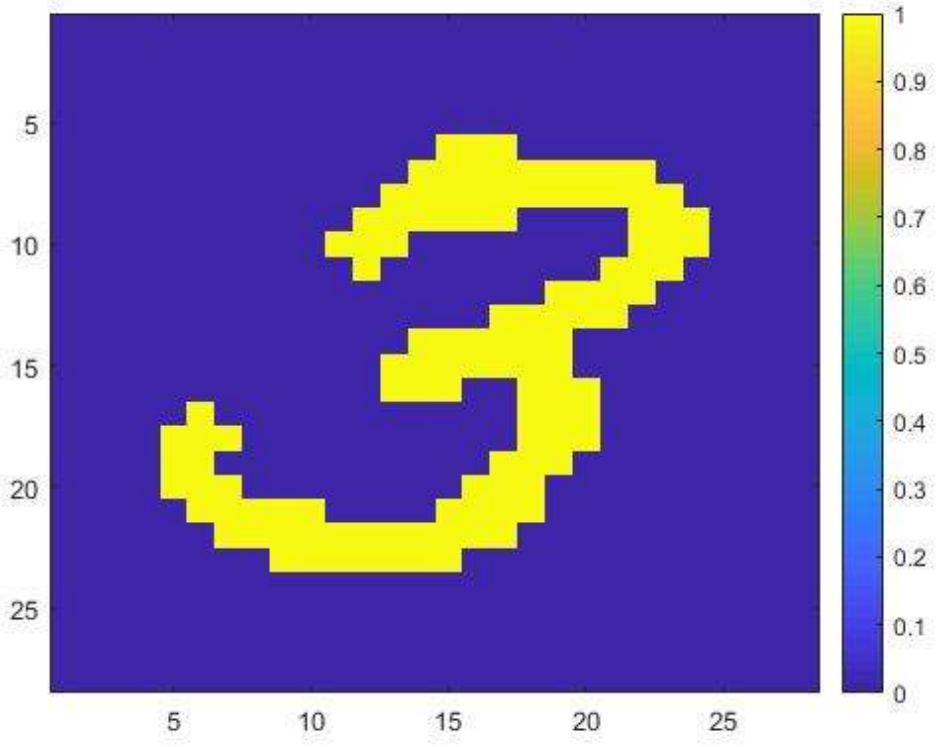
$$l_{D_{x^{y_i}}}(p^{y_i}) = log((p^{y_i})^k (1-p^{y_i})^{k'}) = klog(p^{y_i}) + k'log(1-p^{y_i})$$

$$\nabla_{p_{MLE}^{y_i}} l_{D_{x^{y_i}}}(p^{y_i}) = 0 \Leftrightarrow \frac{k}{p_{MLE}^{y_i}} - \frac{k'}{1-p_{MLE}^{y_i}} = 0$$

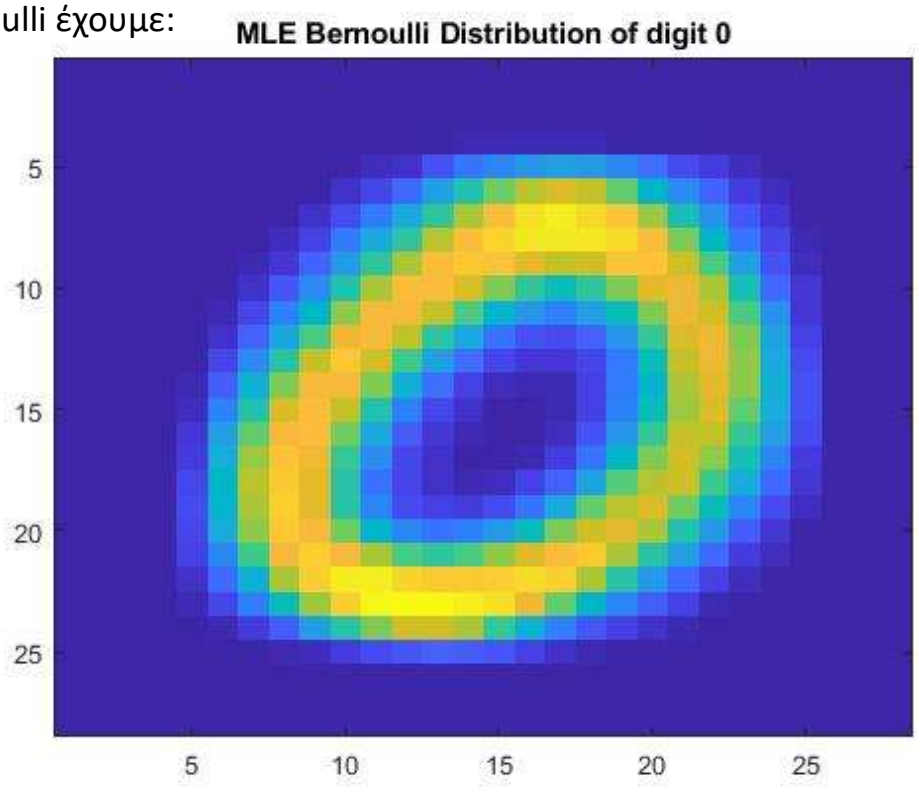
$$\frac{k - kp_{MLE}^{y_i} - k'p_{MLE}^{y_i}}{p_{MLE}^{y_i}(1-p_{MLE}^{y_i})} = 0 \Leftrightarrow p_{MLE}^{y_i} = \frac{k}{k+k'}$$

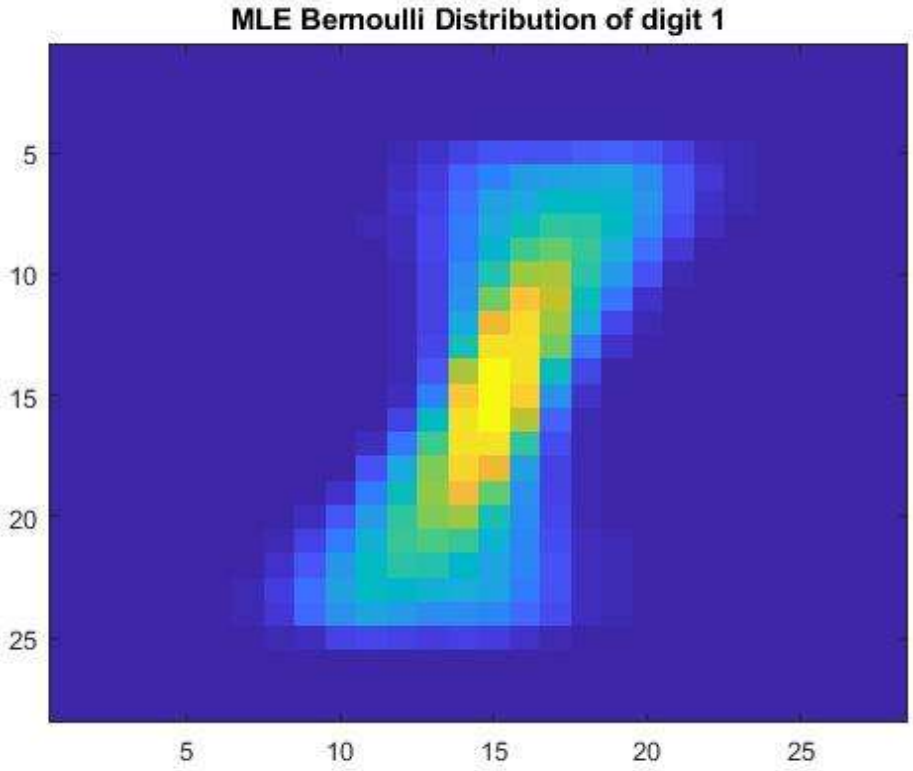
$$p_{MLE}^{y_i} = \frac{1}{n}\sum_{j=1}^{n} x_j^{y_i}$$

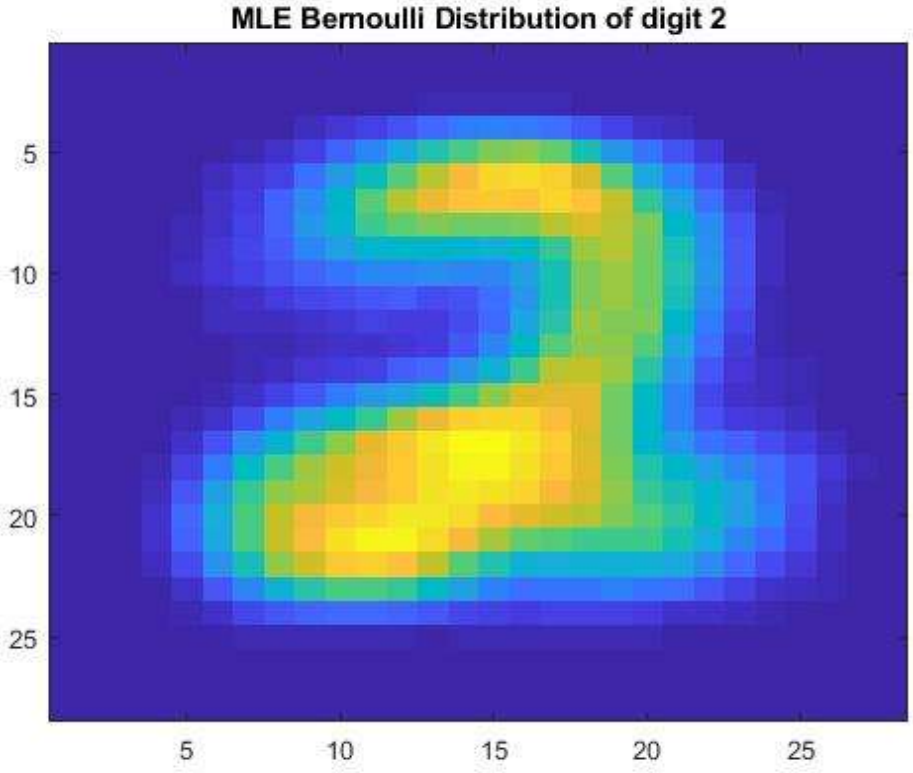
B. Παρακάτω στα γραφήματα, χρησιμοποιώντας τα δοσμένα δείγματα εκπαίδευσης, για κάθε ψηφίο χρησιμοποιούμε τους εκτιμητές μέγιστης πιθανοφάνειας.
Τρέχοντας τον κώδικα που δόθηκε ως παράδειγμα είχαμε την εξής εικόνα:

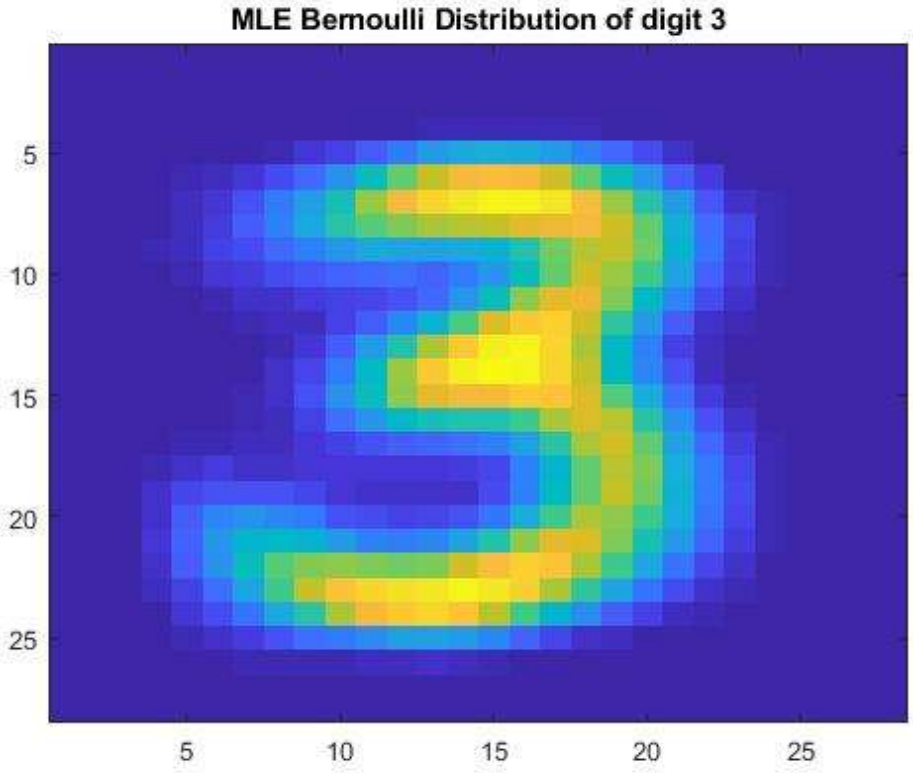


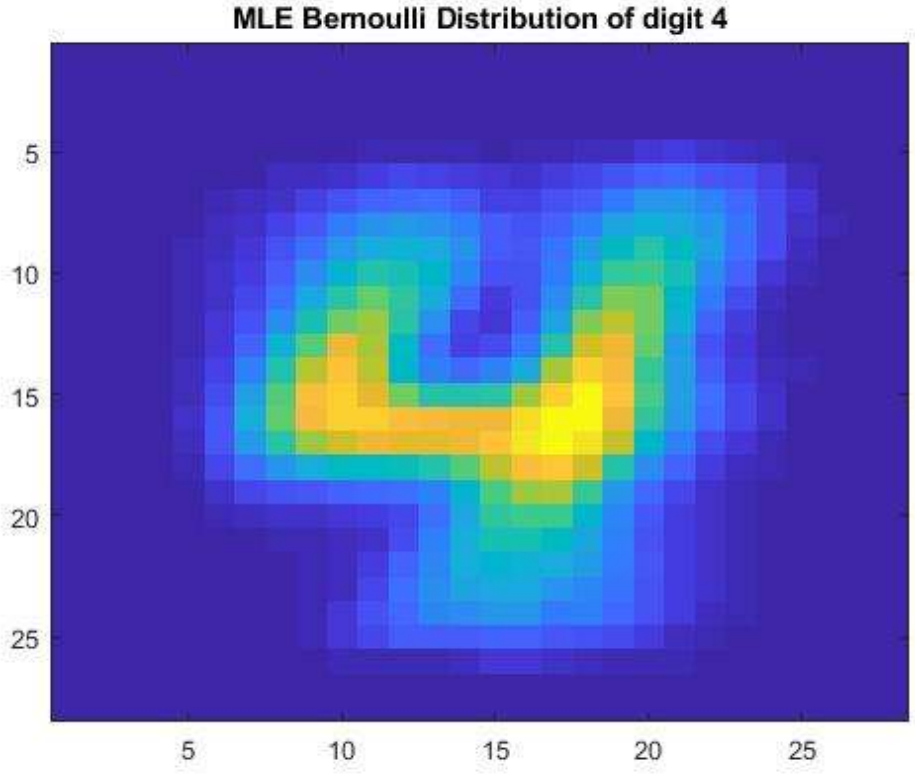
Μετά την εκπαίδευση κάθε ψηφίου, χρησιμοποιώντας την κατανομή Bernoulli έχουμε:

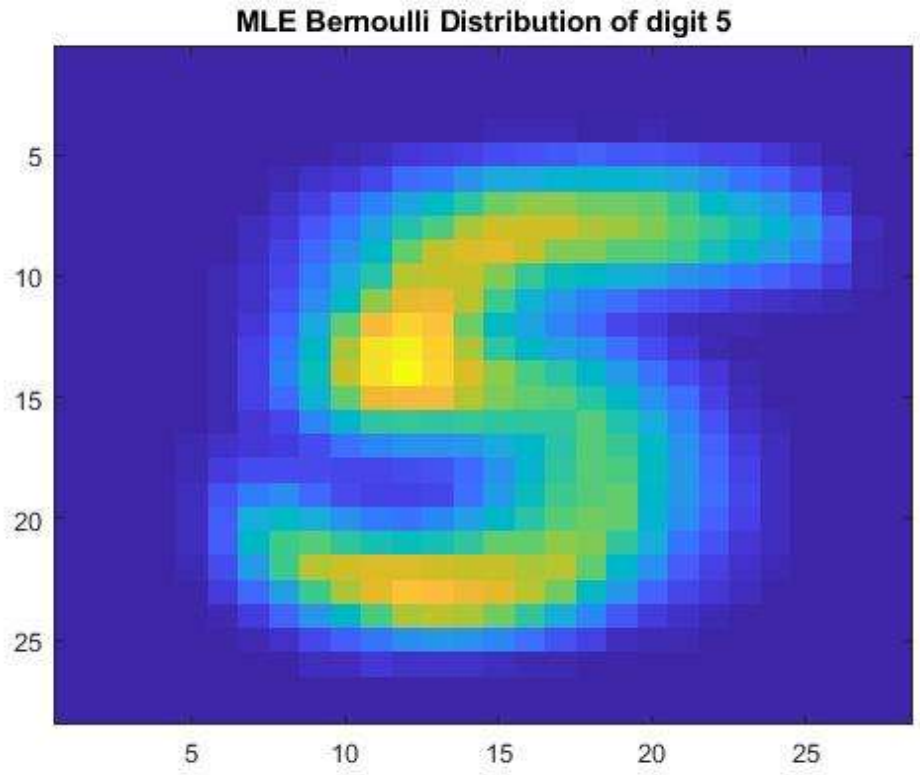


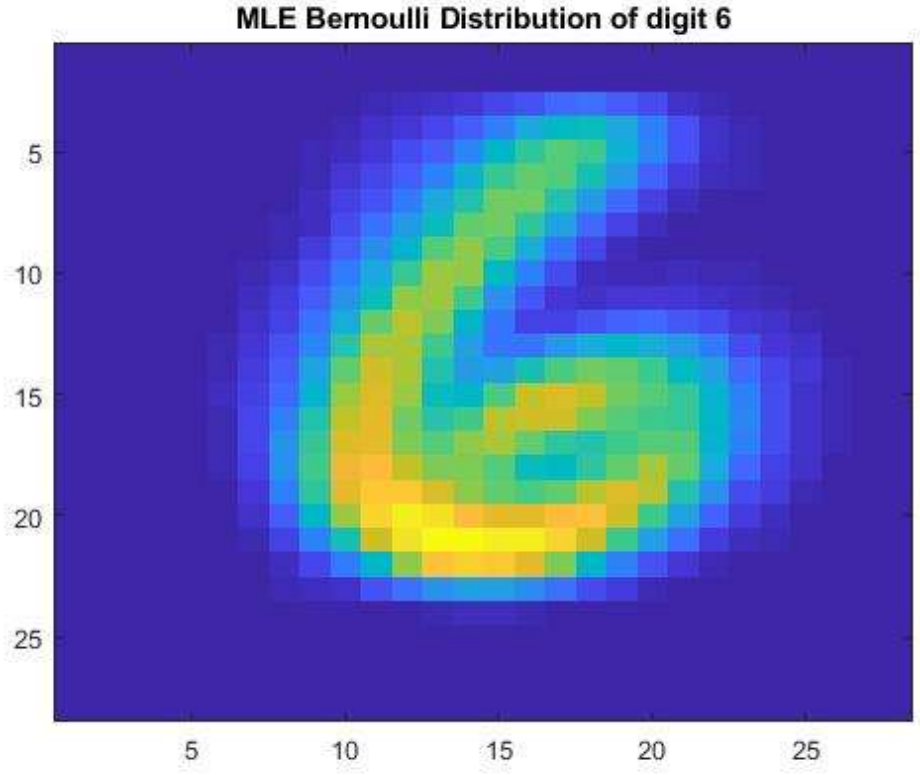


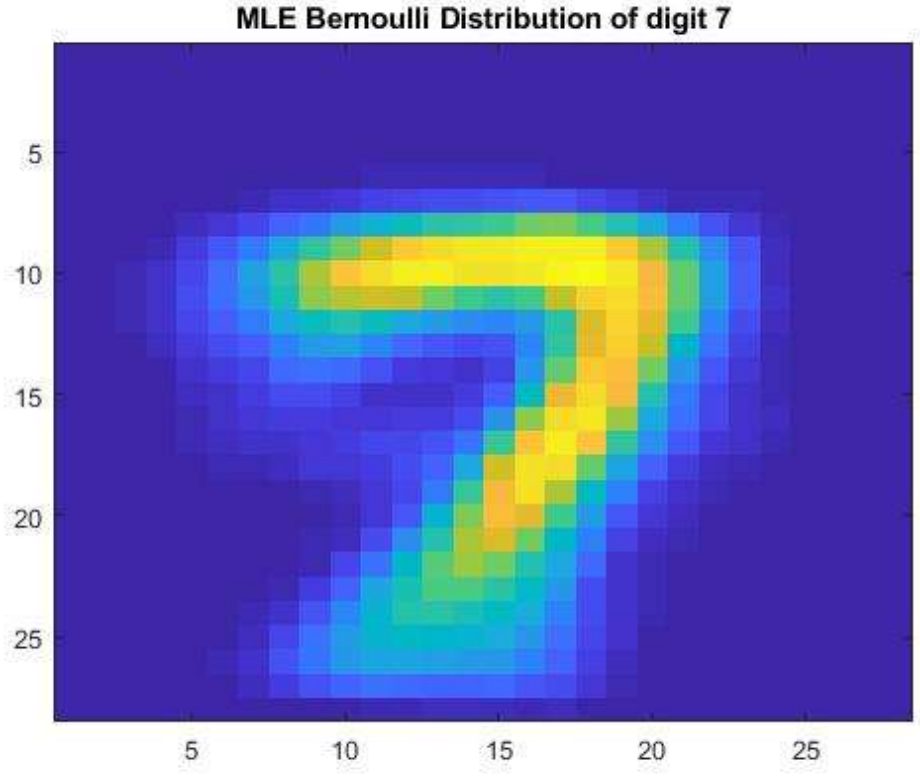

MLE Bernoulli Distribution of digit 2
5
10
15
20
25
5
10
15
20
25

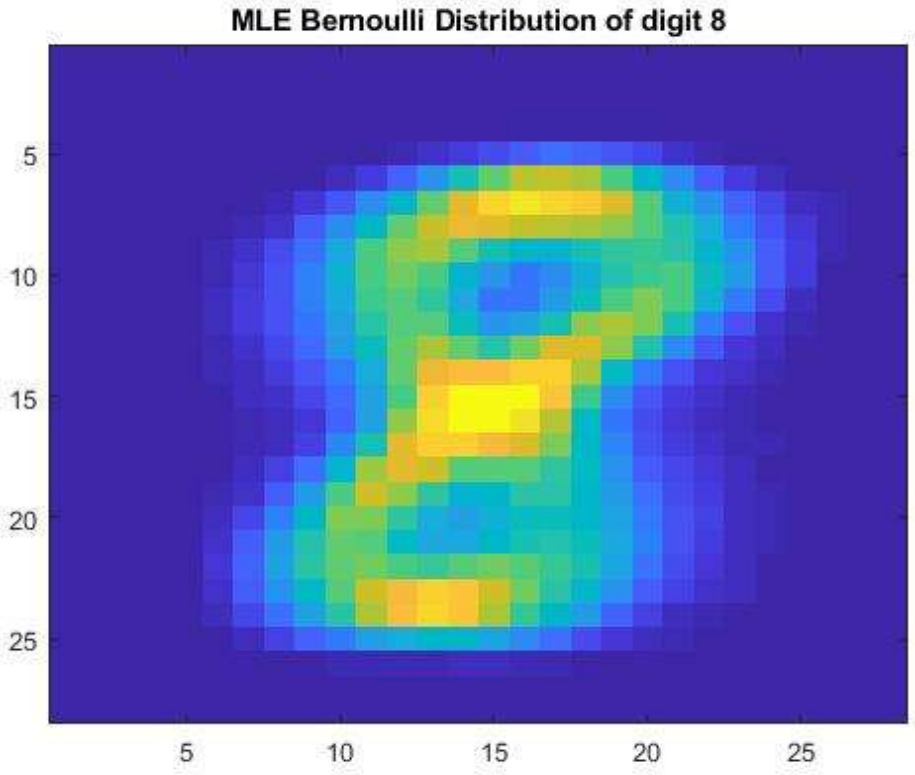

MLE Bernoulli Distribution of digit 3
5
10
15
20
25
5
10
15
20
25

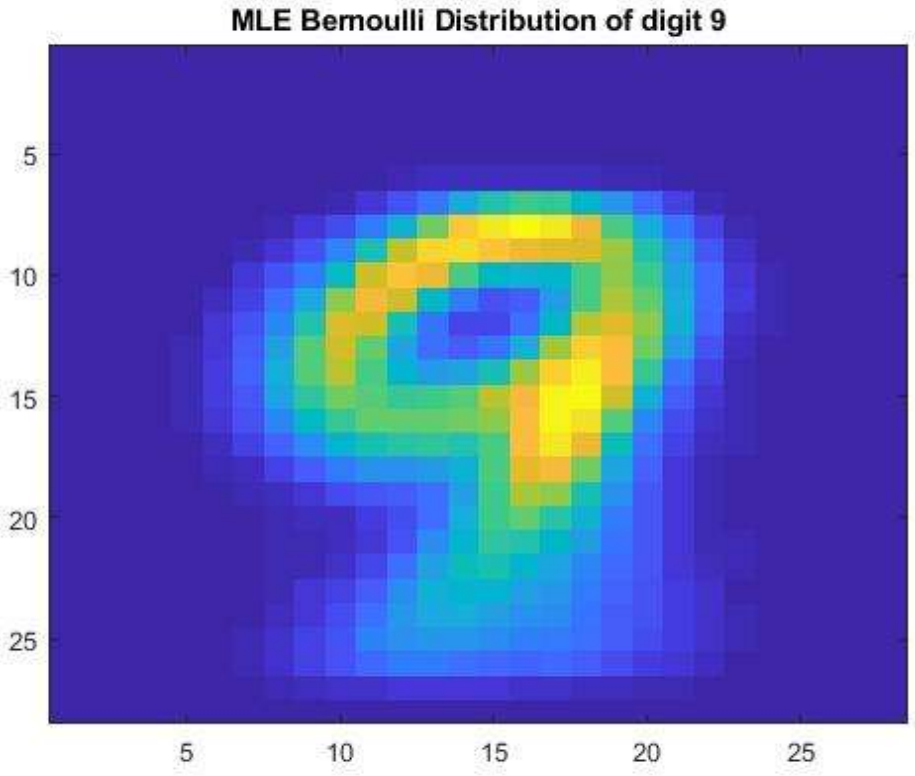

MLE Bernoulli Distribution of digit 4
5
10
15
20
25
5
10
15
20
25


MLE Bernoulli Distribution of digit 5
5
10
15
20
25
5
10
15
20
25


MLE Bernoulli Distribution of digit 6
5
10
15
20
25
5
10
15
20
25


MLE Bernoulli Distribution of digit 7
5
10
15
20
25
5
10
15
20
25


MLE Bernoulli Distribution of digit 8
5
10
15
20
25
5
10
15
20
25


MLE Bernoulli Distribution of digit 9
5
10
15
20
25
5
10
15
20
25

C. Ακρίβεια ταξινόμησης:

(matlab code) …`(correct/(label*size(test0,1))`:

```
Accuracy after testing digits 0 : 0.718
Accuracy after testing digits 1 : 0.858
Accuracy after testing digits 2 : 0.64533
Accuracy after testing digits 3 : 0.577
Accuracy after testing digits 4 : 0.58
Accuracy after testing digits 5 : 0.51033
Accuracy after testing digits 6 : 0.53314
Accuracy after testing digits 7 : 0.54675
Accuracy after testing digits 8 : 0.50978
Accuracy after testing digits 9 : 0.53
Accuracy: 0.53
```

(matlab code) …`(correct/(label*size(test0,2))`:

```
Accuracy after testing digits 0 : 0.45791
Accuracy after testing digits 1 : 0.54719
Accuracy after testing digits 2 : 0.41156
Accuracy after testing digits 3 : 0.36798
Accuracy after testing digits 4 : 0.3699
Accuracy after testing digits 5 : 0.32547
Accuracy after testing digits 6 : 0.34001
Accuracy after testing digits 7 : 0.34869
Accuracy after testing digits 8 : 0.32511
Accuracy after testing digits 9 : 0.33801
Accuracy: 0.53
```

(matlab code) …`(correct/(label*size(test0,3))`:

```
Accuracy after testing digits 0 : 359
Accuracy after testing digits 1 : 429
Accuracy after testing digits 2 : 322.6667
Accuracy after testing digits 3 : 288.5
Accuracy after testing digits 4 : 290
Accuracy after testing digits 5 : 255.1667
Accuracy after testing digits 6 : 266.5714
Accuracy after testing digits 7 : 273.375
Accuracy after testing digits 8 : 254.8889
Accuracy after testing digits 9 : 265
Accuracy: 0.53
```

(matlab code) …`(correct/(label*size(test0,4))`:

```
Accuracy after testing digits 0 : 359
Accuracy after testing digits 1 : 429
Accuracy after testing digits 2 : 322.6667
Accuracy after testing digits 3 : 288.5
Accuracy after testing digits 4 : 290
Accuracy after testing digits 5 : 255.1667
Accuracy after testing digits 6 : 266.5714
Accuracy after testing digits 7 : 273.375
Accuracy after testing digits 8 : 254.8889
Accuracy after testing digits 9 : 265
Accuracy: 0.53
```

(matlab code) …`(correct/(label*size(test0,9))`:

```
Accuracy after testing digits 0 : 359
Accuracy after testing digits 1 : 429
Accuracy after testing digits 2 : 322.6667
Accuracy after testing digits 3 : 288.5
Accuracy after testing digits 4 : 290
Accuracy after testing digits 5 : 255.1667
Accuracy after testing digits 6 : 266.5714
Accuracy after testing digits 7 : 273.375
Accuracy after testing digits 8 : 254.8889
Accuracy after testing digits 9 : 265
Accuracy: 0.53
```

Όταν το σύνολο των δειγμάτων αλλάζει από size(test0,1) σε size(test0,2) … size(test0,9) η ακρίβεια της ταξινόμησης παραμένει σταθερή 0.53. Αντιθέτως, η ακρίβεια για το κάθε ψηφίο (0…9) διαφοροποιείται. Συγκεκριμένα, για τις 2 πρώτες περιπτώσεις που αναφέρουμε έχουμε μικρή μεταβολή προς τα κάτω. Μόλις η «παράμετρος» i (στην έκφραση: size(test0, i), i=1…9) γίνει 3, η διαφοροποίηση είναι ακραία και παραμένει σταθερή.

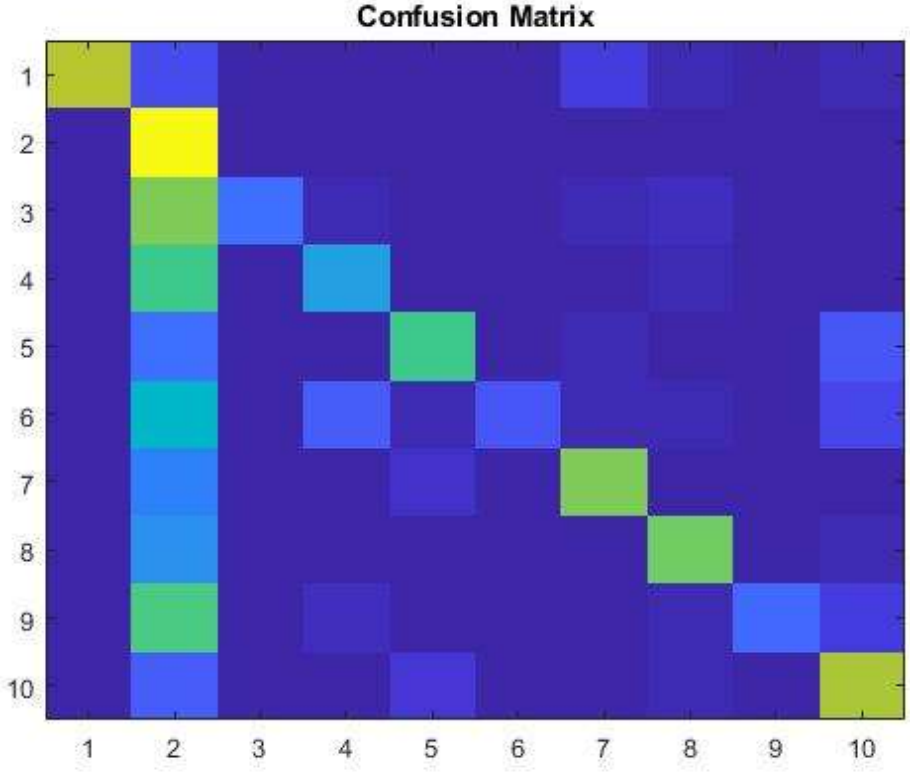
Ο αρχικός πίνακας είναι:

```
confusion_matrix =

   359    64     0     1     2     6    40    10     5    13
     0   499     0     0     0     0     1     0     0     0
     7   329   110    13     4     0    11    19     4     3
     0   294     0   186     1     2     1    11     1     4
     0   112     0     0   296     0    11     1     0    80
     5   231     0    87    14    81    12    14     0    56
     4   135     1     0    24     0   335     0     0     1
     1   162     0     0     5     0     0   321     1    10
     5   298     0    22     5     2     7    11   107    43
     1    90     0     4    32     0     3    13     1   356
```

Ο πίνακας διαμορφώνεται ως εξής:

```
confusion_matrix =

    0.7180    0.1280         0    0.0020    0.0040    0.0120    0.0800    0.0200    0.0100    0.0260
         0    0.9980         0         0         0         0    0.0020         0         0         0
    0.0140    0.6580    0.2200    0.0260    0.0080         0    0.0220    0.0380    0.0080    0.0060
         0    0.5880         0    0.3720    0.0020    0.0040    0.0020    0.0220    0.0020    0.0080
         0    0.2240         0         0    0.5920         0    0.0220    0.0020         0    0.1600
    0.0100    0.4620         0    0.1740    0.0280    0.1620    0.0240    0.0280         0    0.1120
    0.0080    0.2700    0.0020         0    0.0480         0    0.6700         0         0    0.0020
    0.0020    0.3240         0         0    0.0100         0         0    0.6420    0.0020    0.0200
    0.0100    0.5960         0    0.0440    0.0100    0.0040    0.0140    0.0220    0.2140    0.0860
    0.0020    0.1800         0    0.0080    0.0640         0    0.0060    0.0260    0.0020    0.7120
```

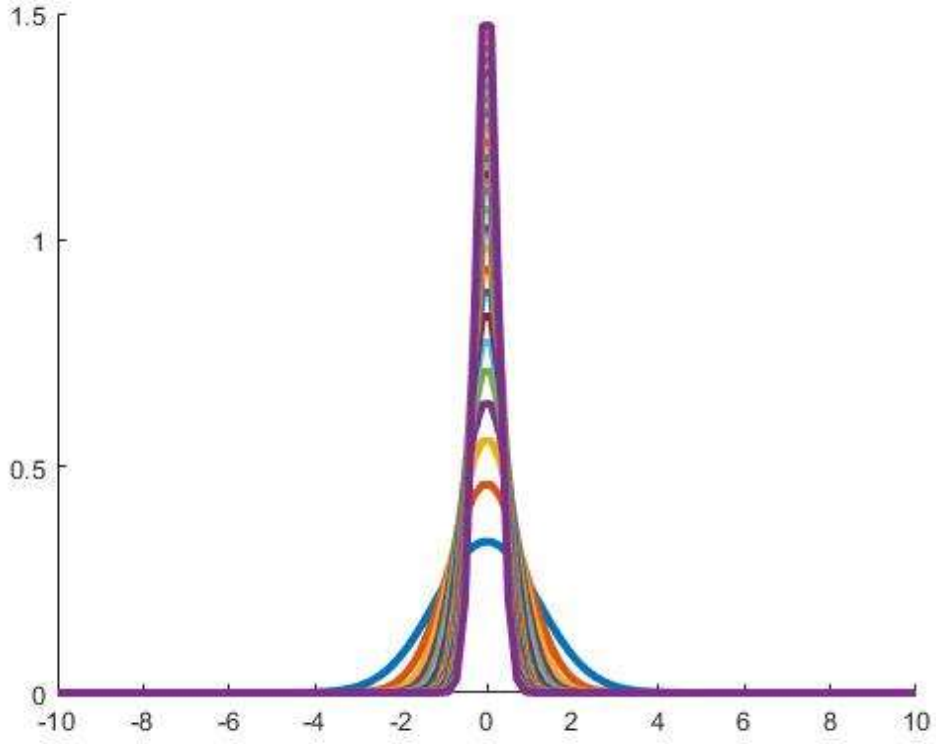
Και τελικά η γραφική απεικόνιση είναι η παρακάτω:



Παρατηρούμε, ότι όσες τιμές του πίνακα confusion_matrix είναι μικρότερες από 0.05(περίπου), απεικονίζονται με μπλε (σκούρο ή λίγο ανοικτό), ενώ όσες είναι από 0.1 και πάνω αρχίζουν και διαφοροποιούν το χρώμα με το οποίο αναπαρίστανται σε κάθε pixel (γαλάζιο, λαχανί, κίτρινο).

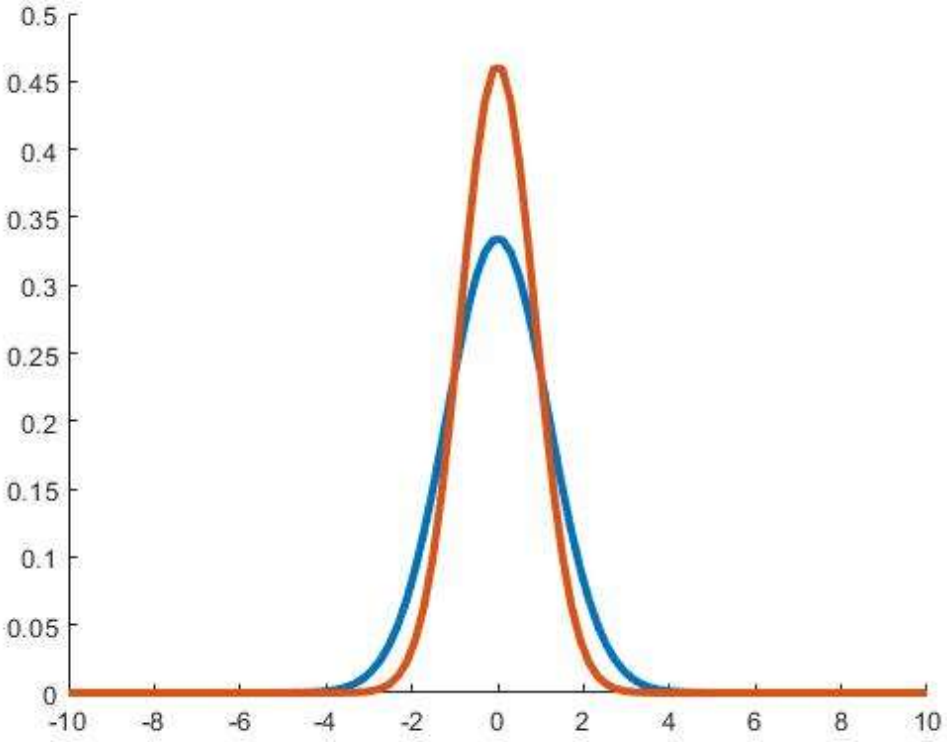
5. Σε αυτή την άσκηση, μελετήσαμε την Bayesian εκτίμηση του μ. Σε ένα σύνολο H(n) με n = 25 μετρήσεις, υποθέτοντας ότι η συνάρτηση πυκνότητας πιθανότητας των μετρήσεων είναι Gaussian, με γνωστή τυπική απόκλιση $\sigma = 1.25$ και μέση τιμή $\mu_0 = 0$.

Α. Για τις διάφορες τιμές του σ_0^2, υλοποιήσαμε σε κοινό γράφημα τη δεσμευμένη πυκνότητα πιθανότητας $p(\mu|H(n))$, καθώς το n μεταβάλλεται από 1 έως 25. Αρχικά, υλοποιώντας για όλες τις τιμές του πίνακα H είχαμε:

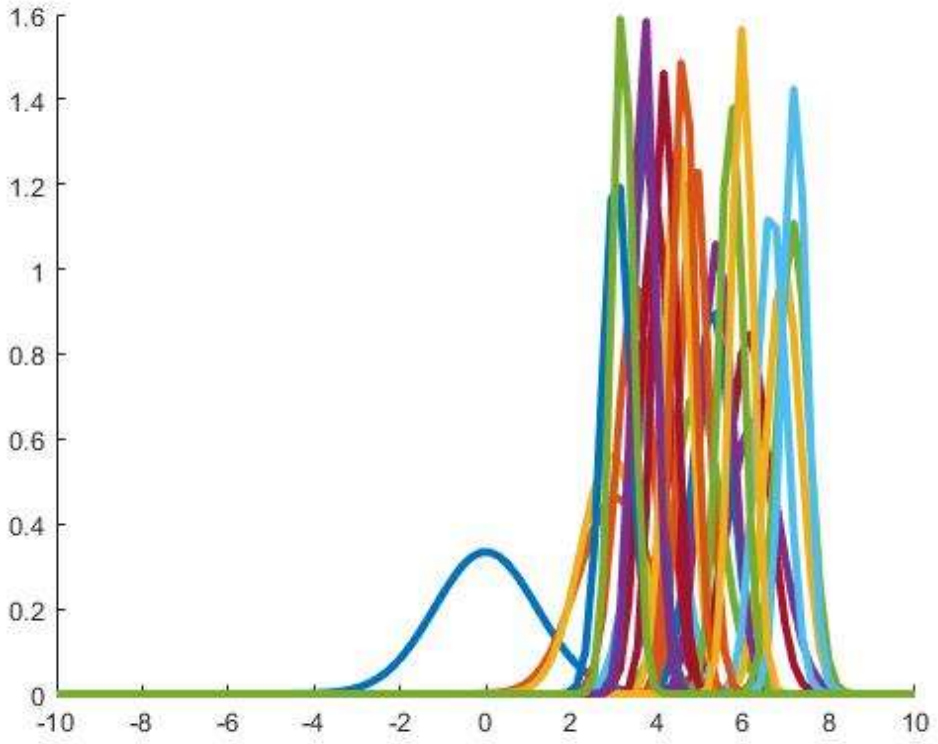


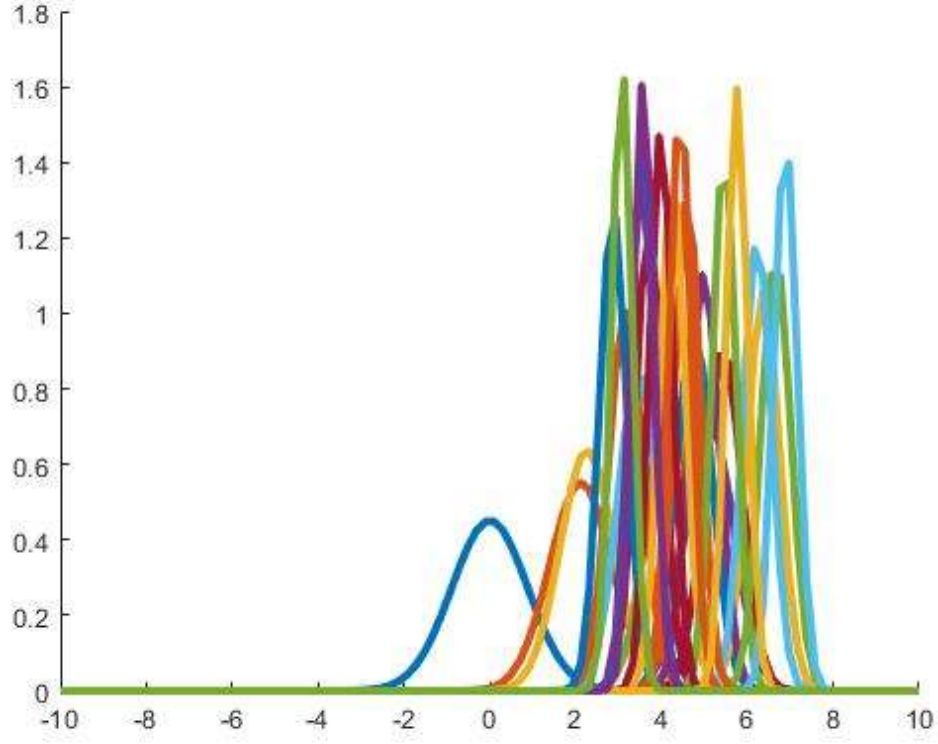
Τελικά, το ζητούμενο γράφημα είναι:

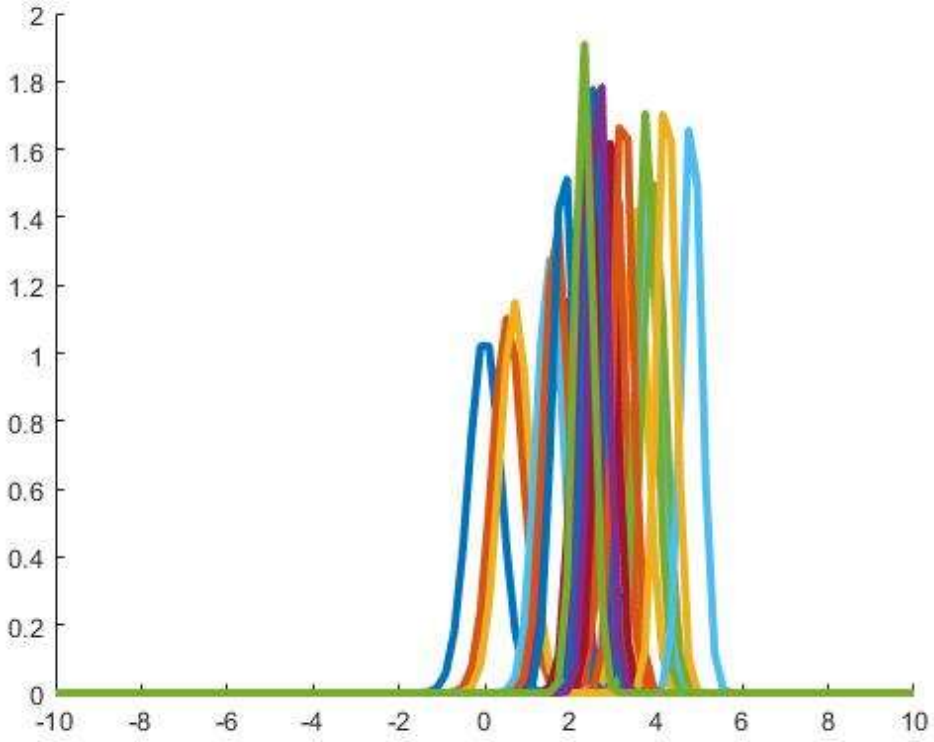
Όταν $\sigma_0^2 = 10 * \sigma^2$:

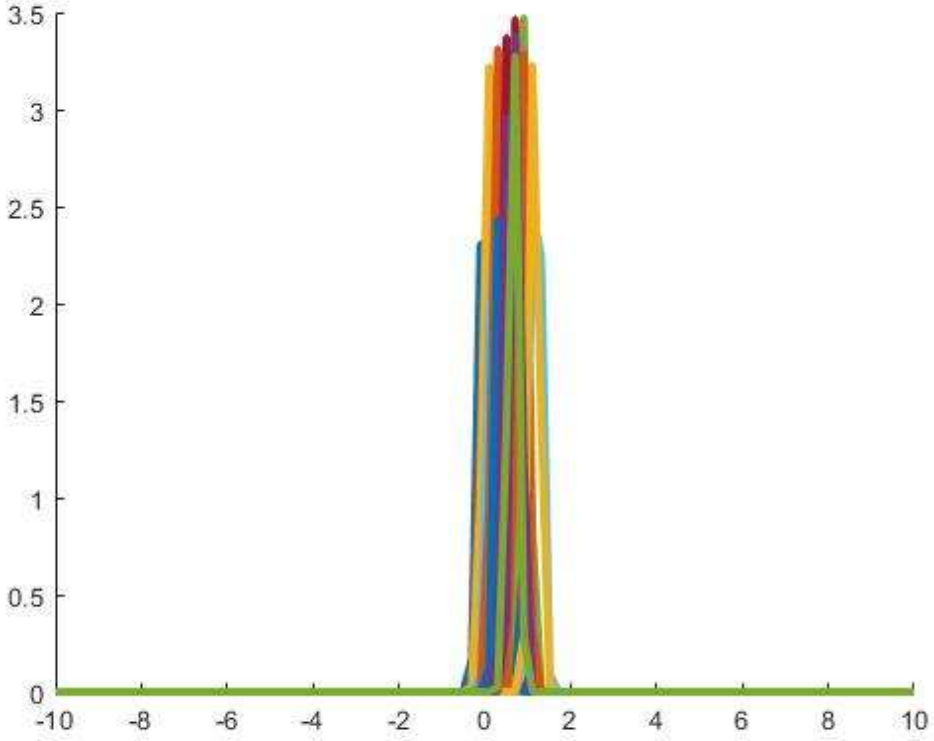


Β. Οι δεσμευμένες πιθανότητες p($x|H(n)$), για όλες τις τιμές του πίνακα Η είναι:



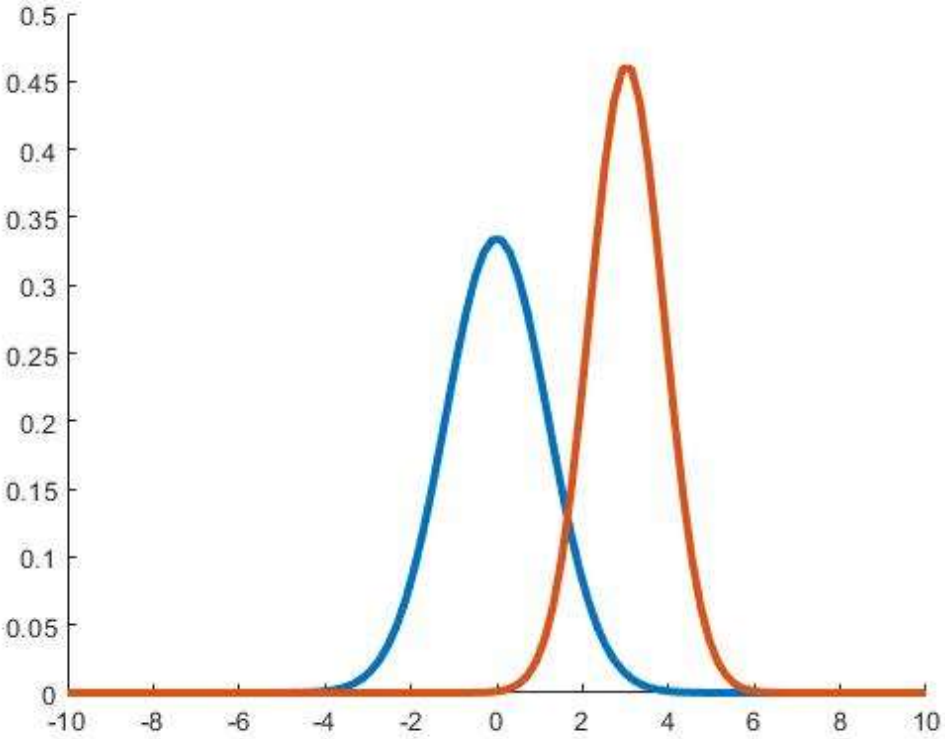




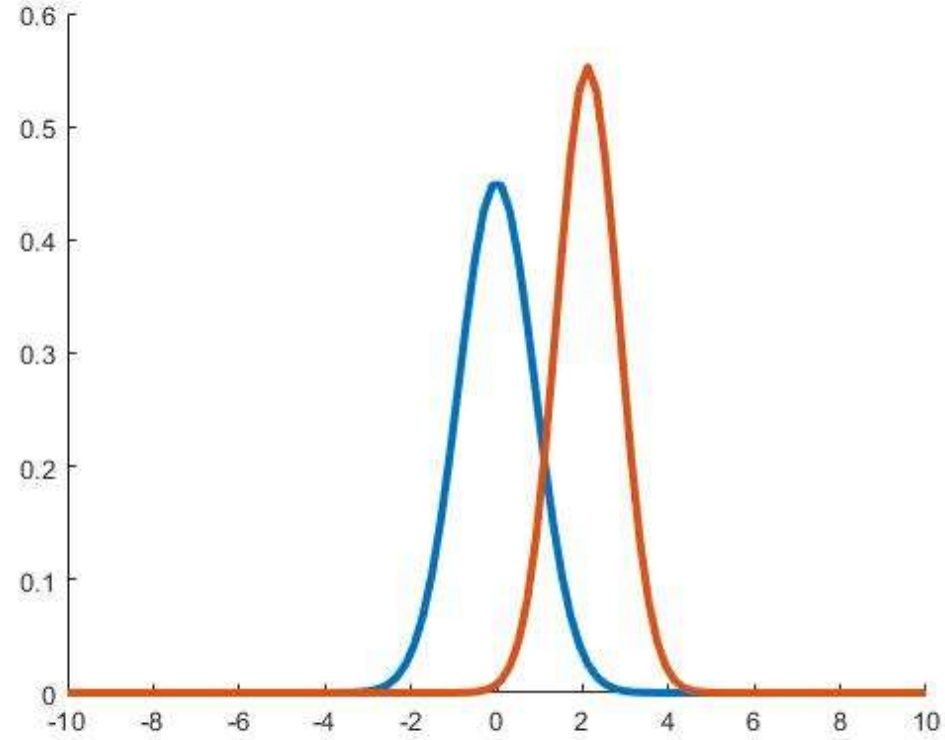


Τα ζητούμενα γραφήματα όμως είναι:

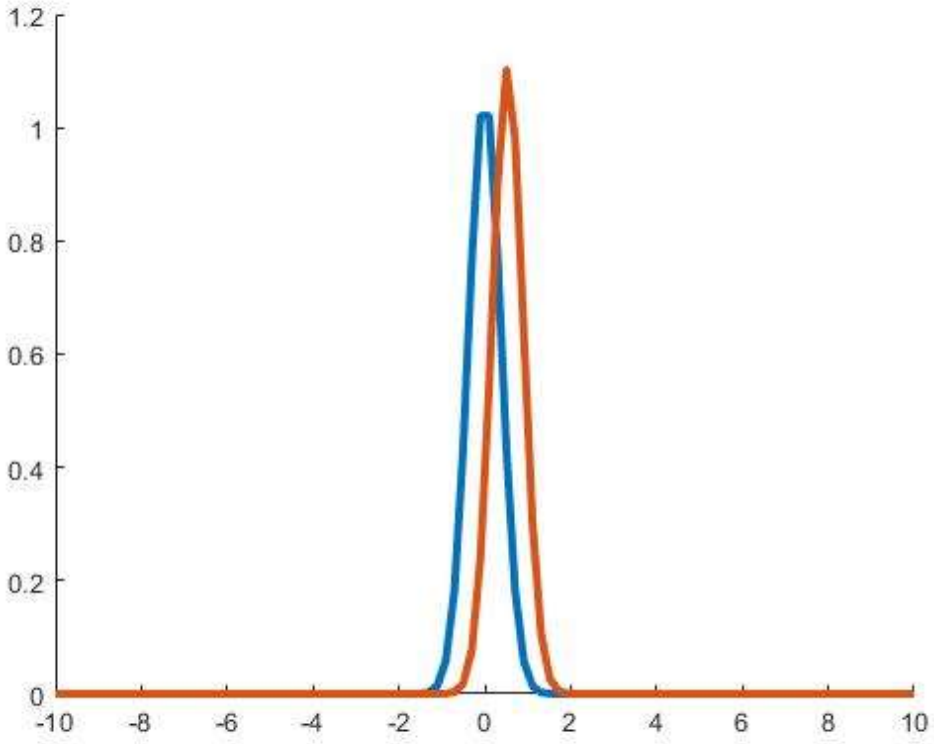
Όταν $\sigma_0^2 = 10 * \sigma^2$:



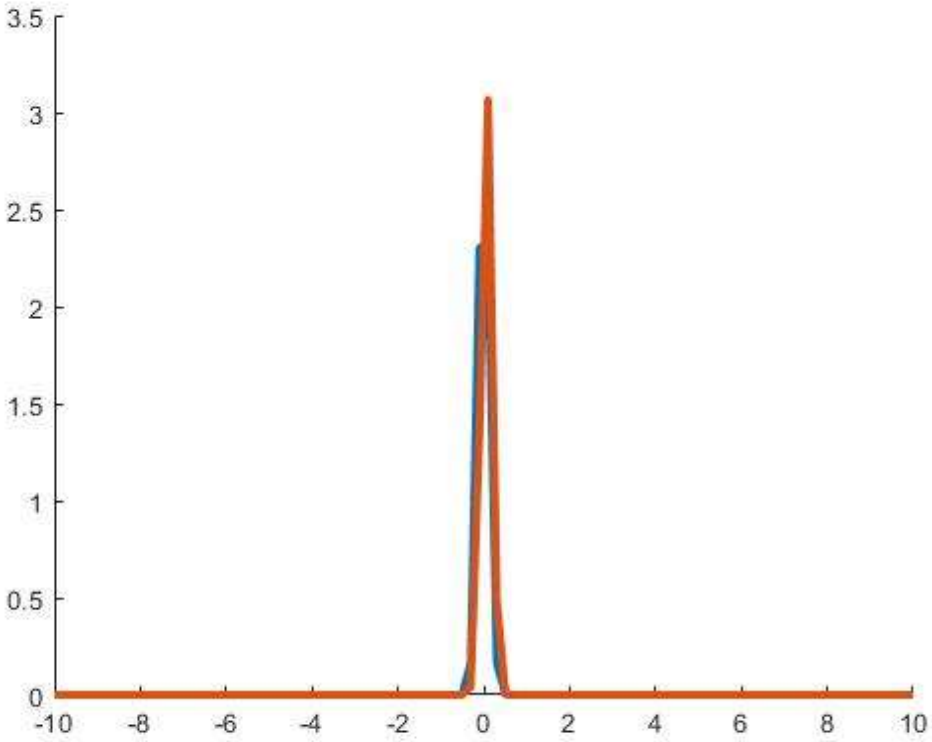
Όταν $\sigma_0^2 = \sigma^2$:



Όταν $\sigma_0^2 = 0.1 * \sigma^2$:



Όταν $\sigma_0^2 = 0.01 * \sigma^2$:



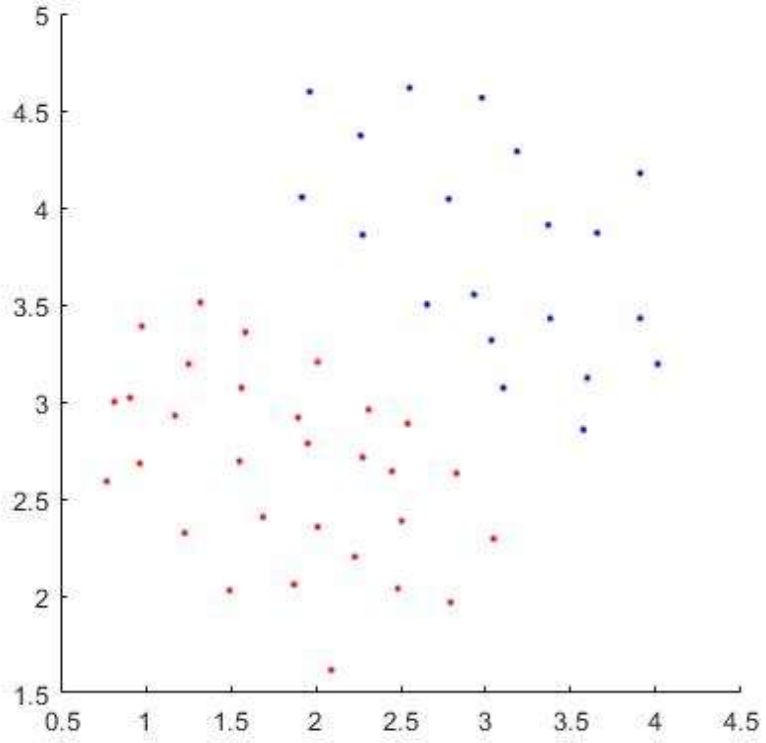
Αποδεικνύεται από τα γραφήματα, ότι όταν το σ_0^2 είναι μεγάλο, η εκτίμηση της μέσης τιμής συγκλίνει γρηγορότερα στο μέσο όρο του δείγματος των μετρήσεων του χρόνου. Όταν το σ_0^2 είναι μικρό, η επίδραση του μέσου όρου του δείγματος στην εκτίμηση της μέσης τιμής είναι λιγότερη, κάνοντάς την να παραμένει κοντά στο $\mu_0 = 0$.

6.

7.

Στην άσκηση αυτή, ασχοληθήκαμε με ένα πρόβλημα βελτιστοποίησης με περιορισμούς, χρησιμοποιώντας Support Vector Machines. Θέλουμε να προβλέψουμε τις τιμές $y^{(i)}$ από τις αντίστοιχες τιμές $x^{(i)}$, $i\epsilon\{1,\ldots,n\}$. Αποθηκεύουμε όλα τα δεδομένα σε ένα πίνακα X, όπου οι γραμμές είναι τα δείγματα και οι στήλες τα χαρακτηριστικά.

Αρχικά προβάλλουμε τα δεδομένα:



Μέρος 1°: Γραμμικά διαχωρίσιμα δείγματα.

Η Λαγκρανζιανή του δυικού προβλήματος είναι

$$\tilde{L}(\lambda) = \sum_{i=1}^{n} \lambda^{(i)} - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda^{(i)} \lambda^{(j)} y^{(i)} y^{(j)} K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$$

όπου $K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \boldsymbol{x}^{(i)} \cdot \left(\boldsymbol{x}^{(j)}\right)^{T}$

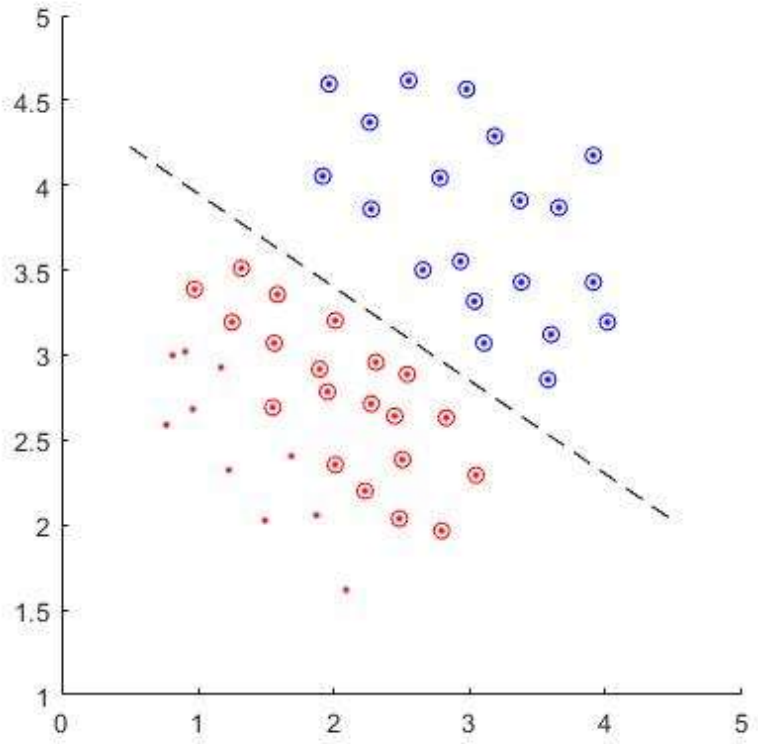
την οποία **μεγιστοποιούμε** κάτω από τους περιορισμούς

$$\lambda^{(i)} \geq 0, \quad i = 1, \dots, n$$

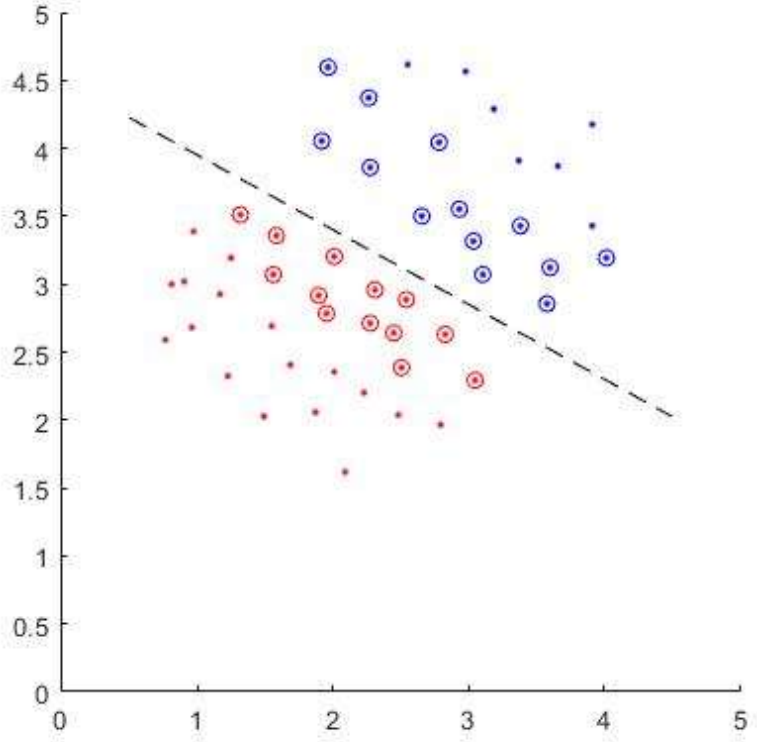
$$\sum_{i=1}^{n} \lambda^{(i)} y^{(i)} = 0$$

Υλοποιώντας τα ερωτήματα, προέκυψαν οι εξής γραφικές:

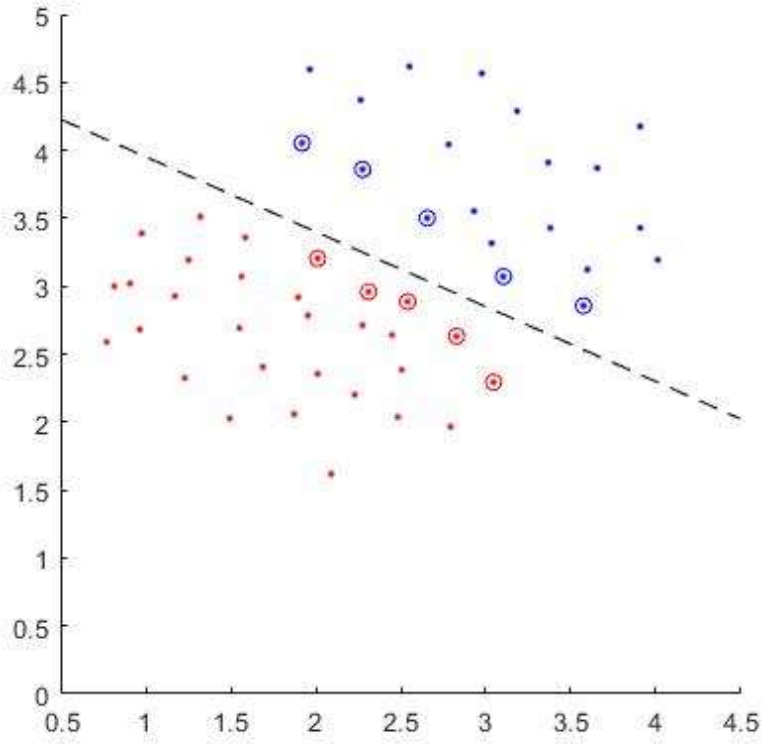
C = 0.01



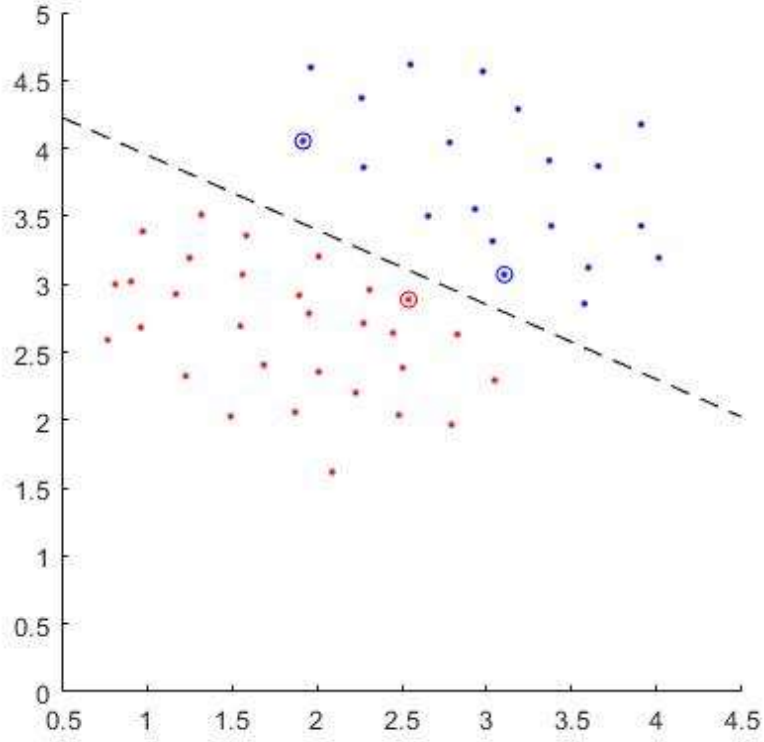
C = 0.1


5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5

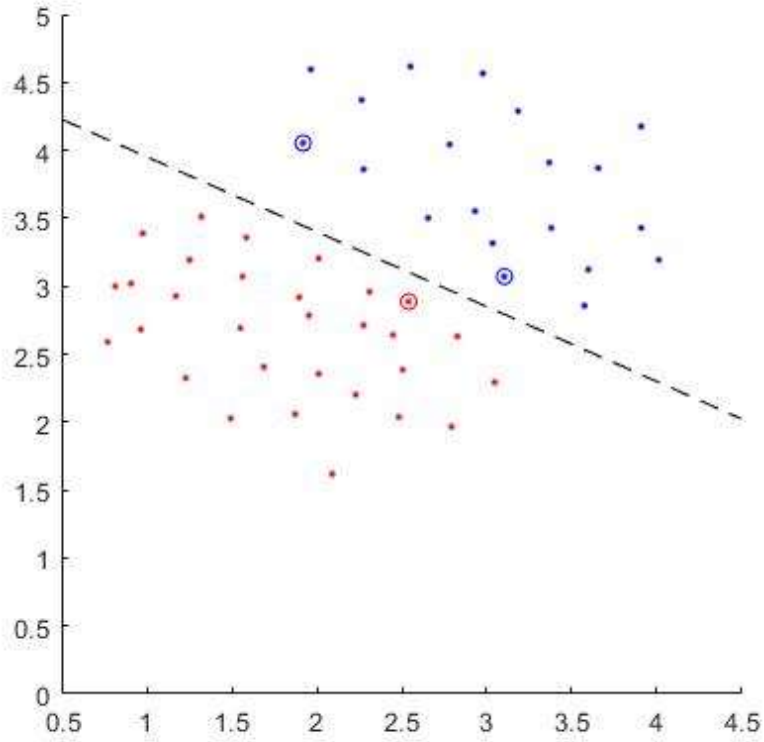
C = 1


5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5

C = 10



C = 100



Και για μεγαλύτερο C, παραμένει η ίδια γραφική.

Μέρος 2°: Μεταβλητές περιθωρίου (slack variables).

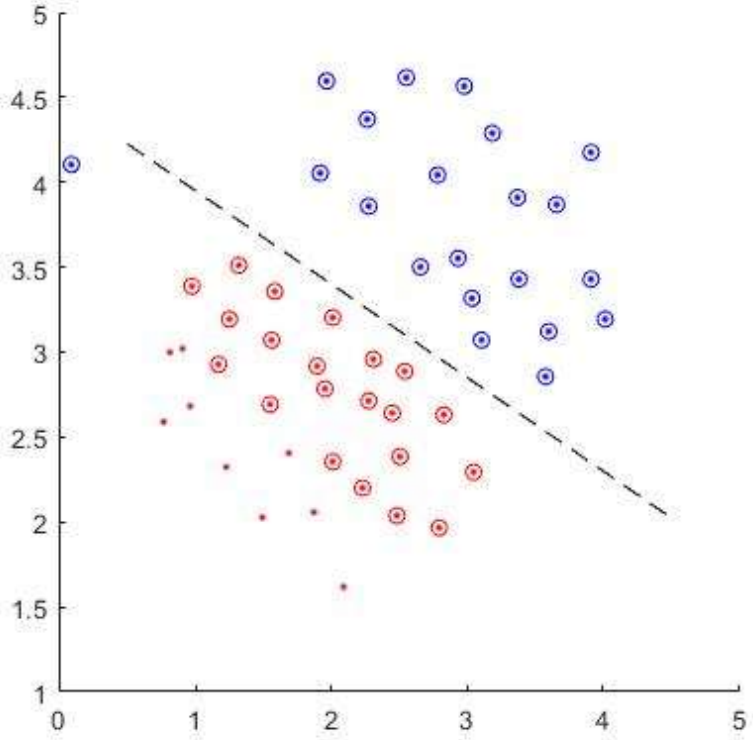
Όταν υπάρχει θόρυβος στα δεδομένα χρησιμοποιούμε μεταβλητές περιθωρίου για να γίνονται ανεκτά κάποια παραδείγματα στην λάθος πλευρά της επιφάνειας απόφασης. Η παράμετρος που καθορίζει το επίπεδο ανοχής συνήθως συμβολίζεται με C και παίρνει τιμές στο διάστημα $(0, +\infty)$, με 0^+ να αντιστοιχεί σε μέγιστη ανοχή και $+\infty$ σε μηδενική ανοχή.

Η Λαγκρανζιανή του δυικού προβλήματος είναι ίδια με αυτή του πρώτου μέρους αλλά την **μεγιστοποιούμε** κάτω από τους παρακάτω περιορισμούς:

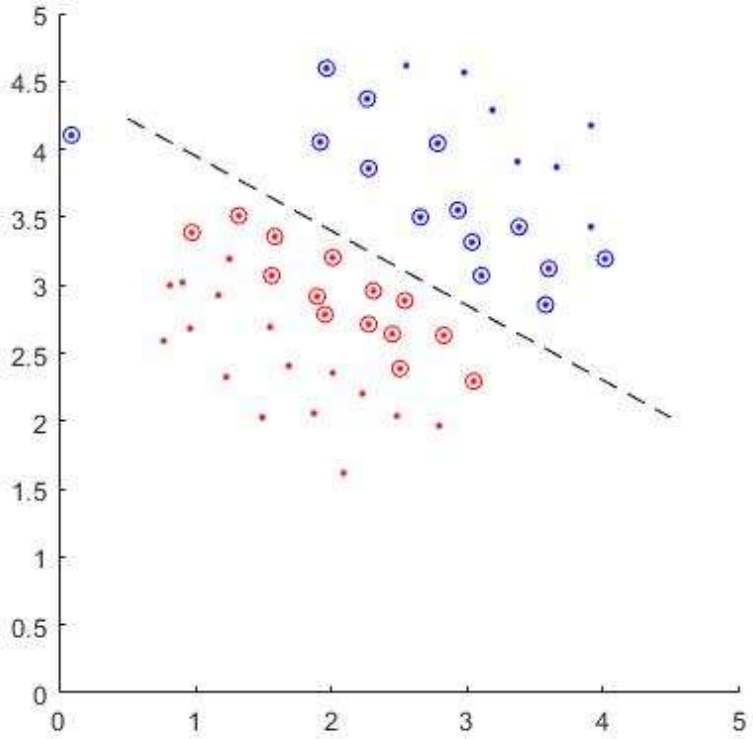
$$0 \leq \lambda^{(i)} \leq C, \quad i = 1, \dots, n$$

Επαναλάβαμε ό,τι και στο προηγούμενο μέρος, αλλά για όλα τα δεδομένα του αρχείου και πήραμε τα εξής αποτελέσματα:

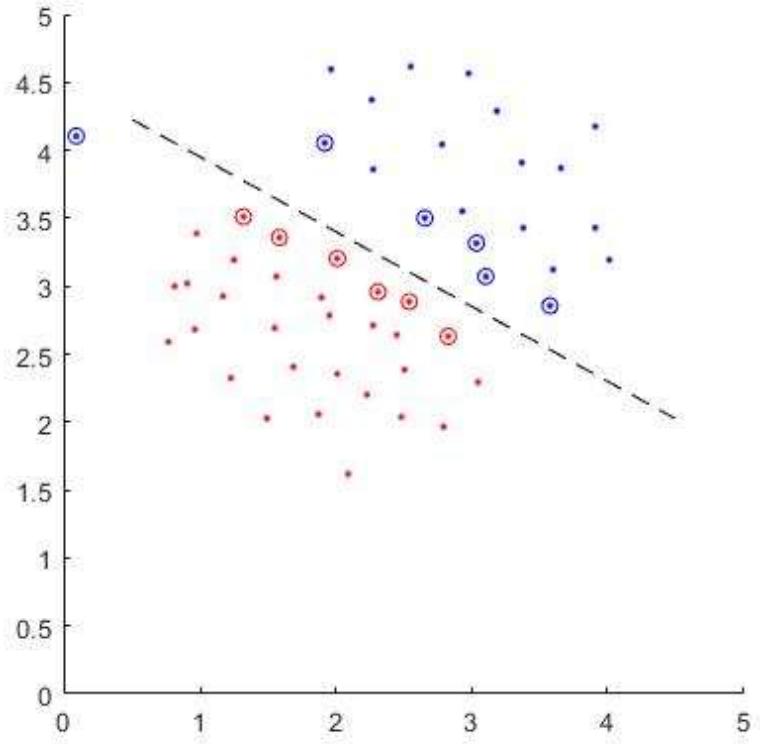
C = 0.01



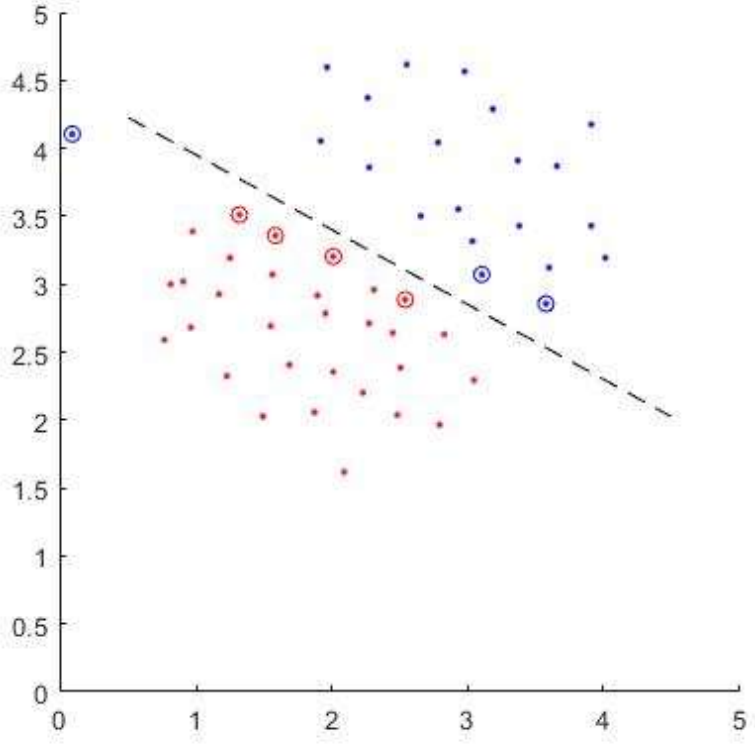
C = 0.1



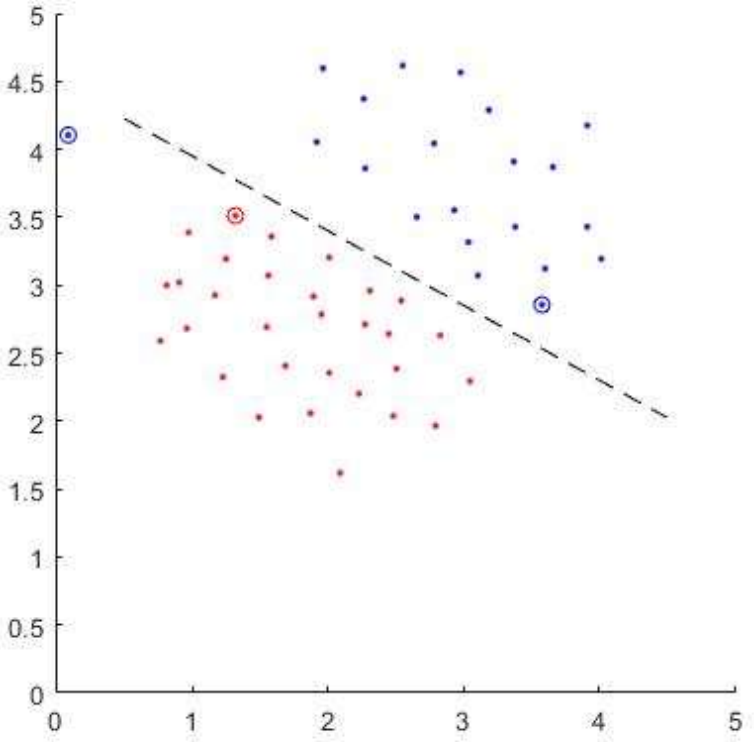
C = 1



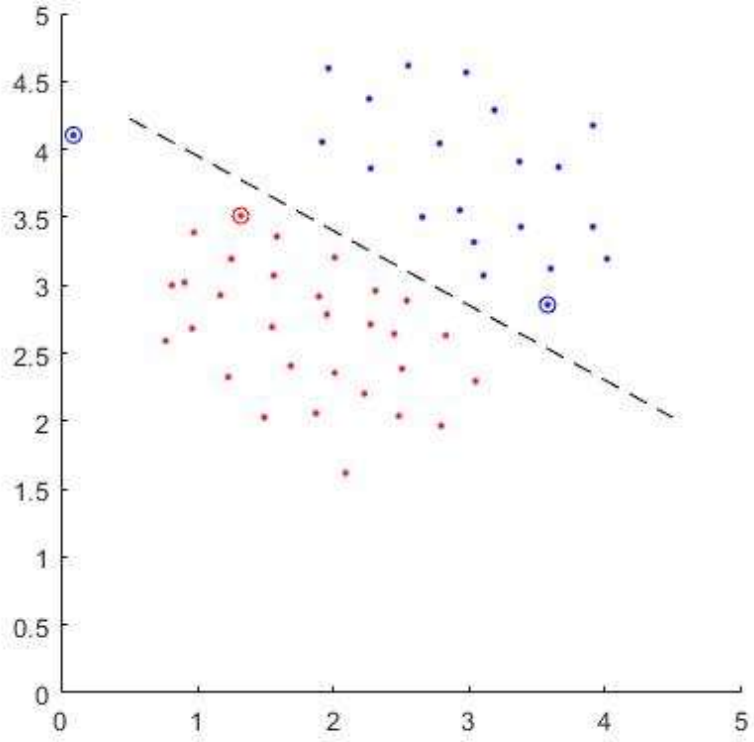
C = 10



C = 100



C = 1000



Ομοίως κι εδώ για μεγαλύτερη τιμή του C, το γράφημα παραμένει σταθερό.