



Άσκηση 1 - ΕΠΛ 442

Όνομα : Έλια Νικολάου 1012334

Ημερομηνία παράδοσης: 15/10/20

Επεξήγηση – Σκοπός Προγράμματος

Σκοπός του προγράμματος είναι η υλοποίηση ενός Feed-Forward νευρωνικού δικτύου που μπορεί να μάθει να υπολογίζει την exclusive-OR (XOR) συνάρτηση. Ως μέθοδος μάθησης χρησιμοποιείται η Back Propagation.

Επεξήγηση Κλάσεων

1. Drive

Η κλάση αυτή λειτουργεί ως driver. Αποτελεί το σημείο εισόδου του προγράμματος. Εδώ διαβάζονται από τα δοθέντα αρχεία όλοι οι παράμετροι που χρειάζονται για την λειτουργία του προγράμματος. Η Drive κλάση είναι υπεύθυνη να δημιουργήσει το νευρωνικό μας δίκτυο (τύπου) Neurons_Network, και να καλέσει τις συναρτήσεις οι οποίες θα τρέξουν το training και το testing του νευρωνικού δικτύου καθώς επίσης είναι υπεύθυνη για να περάσει τα αποτελέσματα στα αρχεία υπολογίζοντας το error (training & testing) και το success rate (training & testing).

Η Drive παράγει δύο αρχεία

1. successrate.txt => περιλαμβάνει το ποσοστό ορθών αποτελεσμάτων κατά την εκμάθηση (training phase) και το ποσοστό ορθών αποτελεσμάτων κατά τον έλεγχο (testing phase).
2. errors.txt => περιλαμβάνει το λάθος εκμάθησης (training error) στο τέλος κάθε επανάληψης και το λάθος ελέγχου (testing error) στο τέλος κάθε επανάληψης.

2. Neurons Network

Η κλάση αυτή αποτελεί την σημαντικότερη κλάση του προγράμματος μας. Είναι ουσιαστικά ένα πρότυπο ενός νευρωνικού δικτύου. Περιέχει ένα πίνακα από neurons (τύπου neuron), τα οποία χαρακτηρίζονται από το layer τους και τα βάρη τα οποία συνδέουν το κάθε νευρώνα με τους νευρώνες του επόμενου layer. Η κλάση λειτουργεί όπως και ένα νευρωνικό δίκτυο.

Αρχικά κάνει initialize τους νευρώνες του, δίνοντας τους random τιμές καθώς επίσης και το threshold του κάθε νευρώνα με τον ίδιο τρόπο.

Μετά όταν καλείται από το Drive, εκπαιδεύεται και ελέγχεται για συγκεκριμένα inputs και outputs

- > training με την χρήση του Forward_propagation & του Back_propagations (δεξ συναρτήσεις)
- > testing με την χρήση μόνο του Forward_propagation.

3. Layer

Αποτελεί ουσιαστικά βοηθητική κλάση. Δεν έχει κάποια πολύ βασική λειτουργία για το πρόγραμμα, αφού απλά περιέχει μεταβλητές που χρησιμοποιούνται από τις υπόλοιπες κλάσεις. Βασικά περιέχει πληροφορίες για τα επιμέρους layers, τι layers υπάρχουν και πόσους νευρώνες έχει το κάθε layer.

4. Neuron

Η κλάση αυτή αποτελεί πρότυπο ενός νευρώνα. Ένας νευρώνας χαρακτηρίζεται από το output, το bias, τα weights , το προηγούμενο bias και το layer στο οποίο ανήκει. Δεν έχει κάποια συνάρτηση μέσα. Στην δική μας περίπτωση χρησιμοποιείται ως αντικείμενο από την κλάση Neurons_Network.

Λεπτομέρειες Λειτουργίας (instructions)

Το πρόγραμμα μου δεν χρειάζεται κάποια συγκεκριμένη διαχείριση για να λειτουργήσει.

Κάνω όμως την παραδοχή ότι το file parameters.txt είναι σταθερό. (Το έχω ρητά ως μεταβλητή μέσα στο πρόγραμμα)

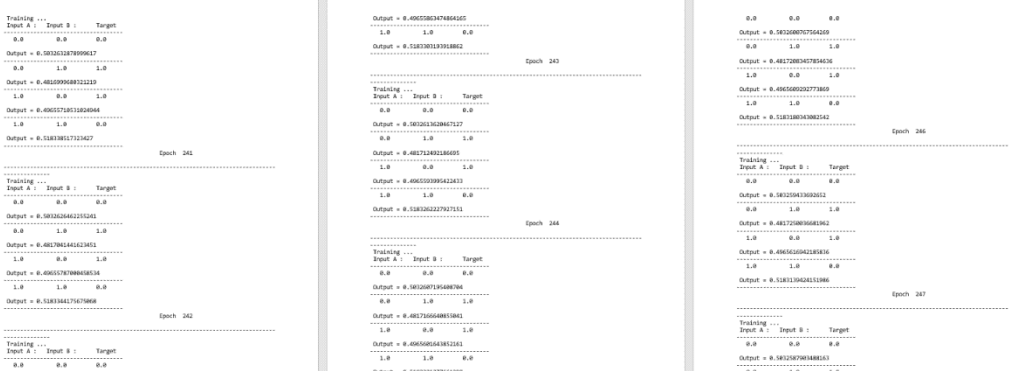
Παραδείγματα Λειτουργίας (simulations)

Δημιούργησα πολλά παραδείγματα αλλάζοντας το learning_rate, το momentum και τα iterations. Τα διάφορα αποτελέσματα μου έδιναν στοιχεία και συμπεράσματα. Ωστόσο ο αριθμός των layer είναι σταθερός | 2 (input_neurons) , 2 (Hidden1_neurons) , 0 (Hidden2_neurons) , 1 (output_neurons) |

Παραθέτω κάποια από τα παραδείγματα από την αρχή ως ένα από τα τελικά «βέλτιστα» μου, με πολύ καλή εκπαίδευση του δικτύου. (Δεν εμφανίζονται ολόκληρα, επειδή τα iterations είναι πολλά)

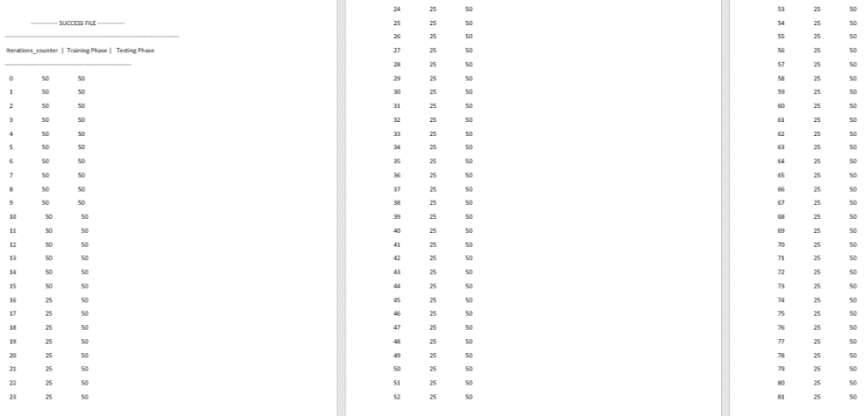
1.	Learning Rate	Momentum	Iterations
	0.3	0.2	200

Console



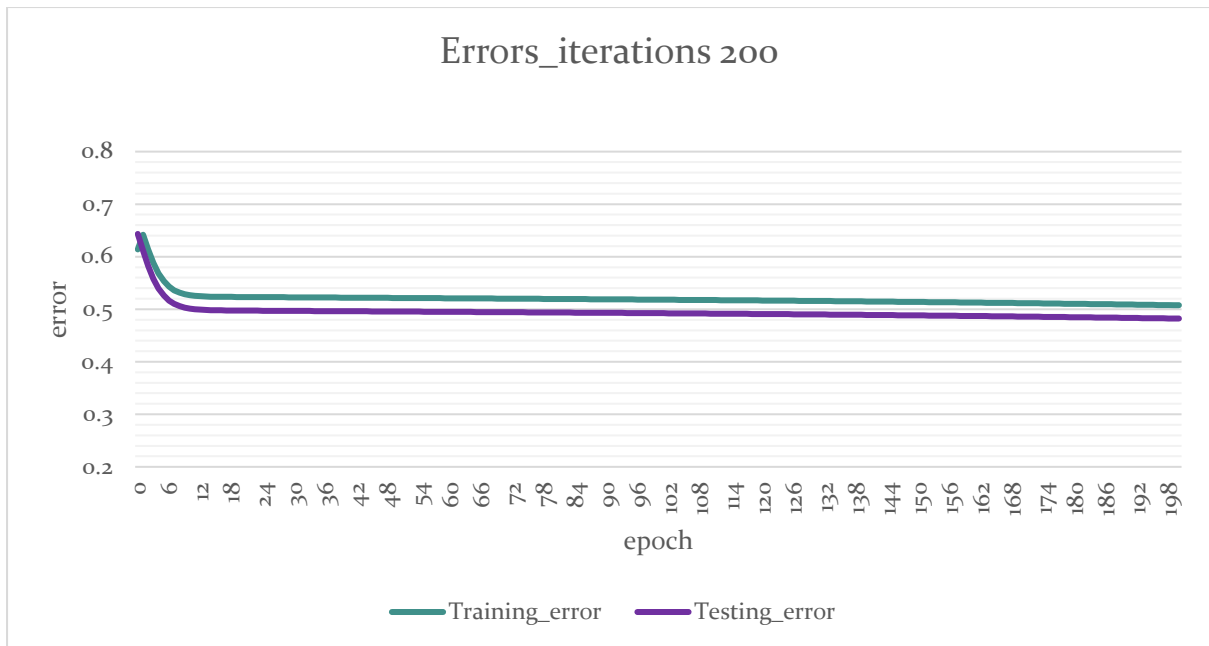
Errors.txt

SuccessRate.txt



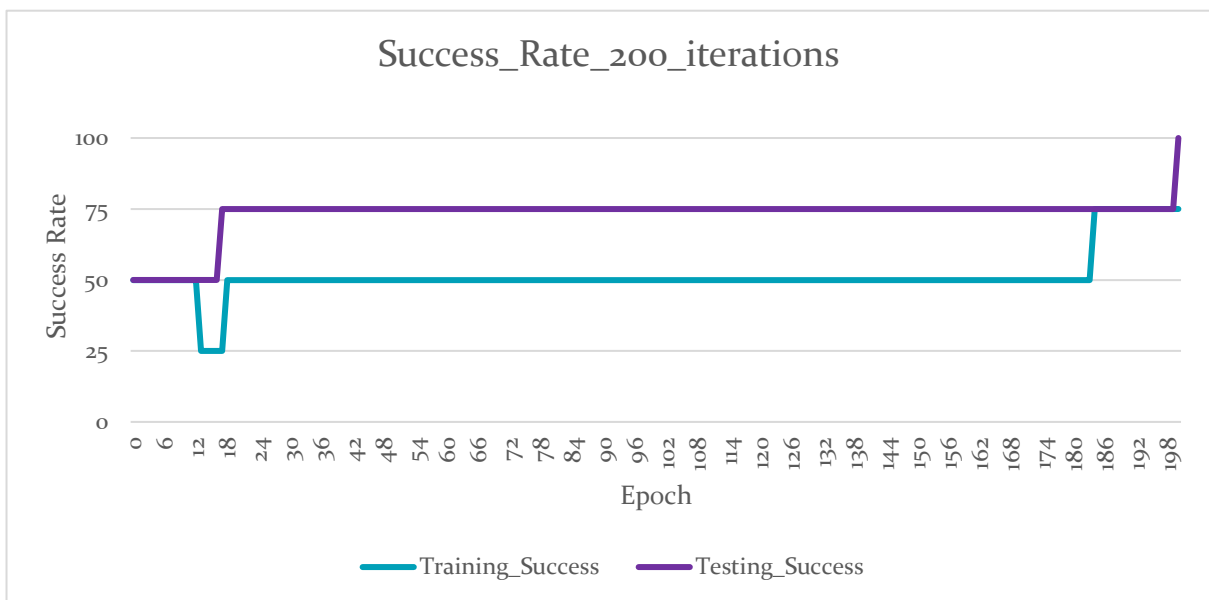
Παρατηρήσεις :

Παρατηρώ ότι παρόλο που το πρόγραμμα δίνει λίγο καλύτερα αποτελέσματα όσο προχωράνε τα iterations – οι εποχές μου, το δίκτυο ωστόσο δεν εκπαιδεύεται άριστα. Πιθανή Λύση είναι να αυξήσω τον αριθμό των iterations μου και να αλλάξω τις υπόλοιπες μεταβλητές



Παρατηρήσεις :

Παρατηρούμε ότι το Training Error είναι υψηλότερο σε σχέση με το Testing Error πράγμα που σημαίνει ότι το συνολικό σφάλμα είναι πιο μεγάλο (εάν το σκεφτούμε ως απόκλιση, τότε η απόκλιση είναι μεγαλύτερη από το επιθυμητό μας αποτέλεσμα). Παρατηρούμε ωστόσο ότι καθώς αυξάνονται οι εποχές το δίκτυο εκπαιδεύεται λίγο καλύτερα αλλά όχι, δεν εκπαιδεύεται ολόσωστα αφού βλέπουμε ότι παραμένει σταθερό χωρίς μείωση του σφάλματος. Αλλάζοντας τις παραμέτρους μπορούμε να επιτύχουμε μεγάλη ελαχιστοποίηση του σφάλματος.



Παρατηρήσεις :

Παρατηρούμε ότι το Training_Success_Rate είναι χαμηλότερο σε σχέση με το Testing_Success_Rate. Ωστόσο το δίκτυο πετυχαίνει 100% success στα τελευταία iterations, εάν αυξήσουμε το learning rate & το momentum μπορούμε να δούμε ότι το δίκτυο εκπαιδεύεται καλύτερα ως αποτέλεσμα να έχουμε πιο πολύ αποτελεσματικότητα και για το training και το testing.

2. Αλλαγή του αριθμού των iterations , του learning_rate και του momentum

Learning Rate	Momentum	Iterations
0.5	0.5	200

[illegible]

Console

**Καταλήγουμε να έχουμε
τιμές πολύ κοντά στα
επιθυμητά μας
αποτελέσματα**

SUCCESS FILE											
Iterations_counter Training Phase Testing Phase											
0	50	50	168	75	75	401	75	75			
1	50	50	169	75	75	402	75	75			
2	50	50	170	75	75	403	75	75			
3	50	50	171	75	75	404	75	75			
4	50	50	172	75	75	405	75	75			
5	25	50	173	75	75	406	75	75			
6	25	50	174	75	75	407	75	75			
7	0	50	175	75	75	408	75	100			
8	0	50	176	75	75	409	75	100			
9	0	50	177	75	75	410	75	100			
10	0	50	178	75	75	411	75	100			
11	0	50	179	75	75	412	75	100			
12	0	50	180	75	75	413	75	100			
13	0	50	181	75	75	414	75	100			
14	0	50	182	75	75	415	75	100			
15	0	50	183	75	75	416	75	100			
16	0	50	184	75	75	417	100	100			
17	0	50	185	75	75	418	100	100			
18	0	50	186	75	75	419	100	100			
19	0	50	187	75	75	420	100	100			
20	0	50	188	75	75	421	100	100			
21	0	50	189	75	75	422	100	100			
22	0	50	190	75	75	423	100	100			
23	0	50	191	75	75	424	100	100			
24	0	50	192	75	75	425	100	100			
25	0	50	193	75	75	426	100	100			
26	0	50	194	75	75	427	100	100			
27	0	50	195	75	75	428	100	100			
28	0	50	196	75	75	429	100	100			

SuccessRate.txt

***** ENDS FILE *****				50	0.54812835149015	0.48066876581262	1	0.51447765739125	0.460919917224195
iterations_count	Training error	Testing error							
0	0.613769194203	0.604812671823	0.604812671823	50	0.48066876581262	1	0.51447765739125	0.460919917224195	
1	0.604812671823	0.59806228065	0.59806228065	51	0.48066876581262	2	0.51447765739125	0.460919917224195	
2	0.59806228065	0.592668158742	0.592668158742	52	0.48066876581262	3	0.51447765739125	0.460919917224195	
3	0.592668158742	0.58727512864	0.58727512864	53	0.48066876581262	4	0.51447765739125	0.460919917224195	
4	0.58727512864	0.5818821214966	0.5818821214966	54	0.48066876581262	5	0.51447765739125	0.460919917224195	
5	0.5818821214966	0.576487125240623	0.576487125240623	55	0.48066876581262	6	0.51447765739125	0.460919917224195	
6	0.576487125240623	0.571092127913824	0.571092127913824	56	0.48066876581262	7	0.51447765739125	0.460919917224195	
7	0.571092127913824	0.5656971263526	0.5656971263526	57	0.48066876581262	8	0.51447765739125	0.460919917224195	
8	0.5656971263526	0.56030613526	0.56030613526	58	0.48066876581262	9	0.51447765739125	0.460919917224195	
9	0.56030613526	0.5549151263526	0.5549151263526	59	0.48066876581262	10	0.51447765739125	0.460919917224195	
10	0.5549151263526	0.5495241263526	0.5495241263526	60	0.48066876581262	11	0.51447765739125	0.460919917224195	
11	0.5495241263526	0.5441331263526	0.5441331263526	61	0.48066876581262	12	0.51447765739125	0.460919917224195	
12	0.5441331263526	0.5387421263526	0.5387421263526	62	0.48066876581262	13	0.51447765739125	0.460919917224195	
13	0.5387421263526	0.5333511263526	0.5333511263526	63	0.48066876581262	14	0.51447765739125	0.460919917224195	
14	0.5333511263526	0.5279601263526	0.5279601263526	64	0.48066876581262	15	0.51447765739125	0.460919917224195	
15	0.5279601263526	0.5225691263526	0.5225691263526	65	0.48066876581262	16	0.51447765739125	0.460919917224195	
16	0.5225691263526	0.5171781263526	0.5171781263526	66	0.48066876581262	17	0.51447765739125	0.460919917224195	
17	0.5171781263526	0.5117871263526	0.5117871263526	67	0.48066876581262	18	0.51447765739125	0.460919917224195	
18	0.5117871263526	0.5063961263526	0.5063961263526	68	0.48066876581262	19	0.51447765739125	0.460919917224195	
19	0.5063961263526	0.5010051263526	0.5010051263526	69	0.48066876581262	20	0.51447765739125	0.460919917224195	
20	0.5010051263526	0.4956141263526	0.4956141263526	70	0.48066876581262	21	0.51447765739125	0.460919917224195	
21	0.4956141263526	0.4902231263526	0.4902231263526	71	0.48066876581262	22	0.51447765739125	0.460919917224195	
22	0.4902231263526	0.4848321263526	0.4848321263526	72	0.48066876581262	23	0.51447765739125	0.460919917224195	
23	0.4848321263526	0.4794411263526	0.4794411263526	73	0.48066876581262	24	0.51447765739125	0.460919917224195	
24	0.4794411263526	0.4740501263526	0.4740501263526	74	0.48066876581262	25	0.51447765739125	0.460919917224195	
25	0.4740501263526	0.4686591263526	0.4686591263526	75	0.48066876581262	26	0.51447765739125	0.460919917224195	
26	0.46865912								

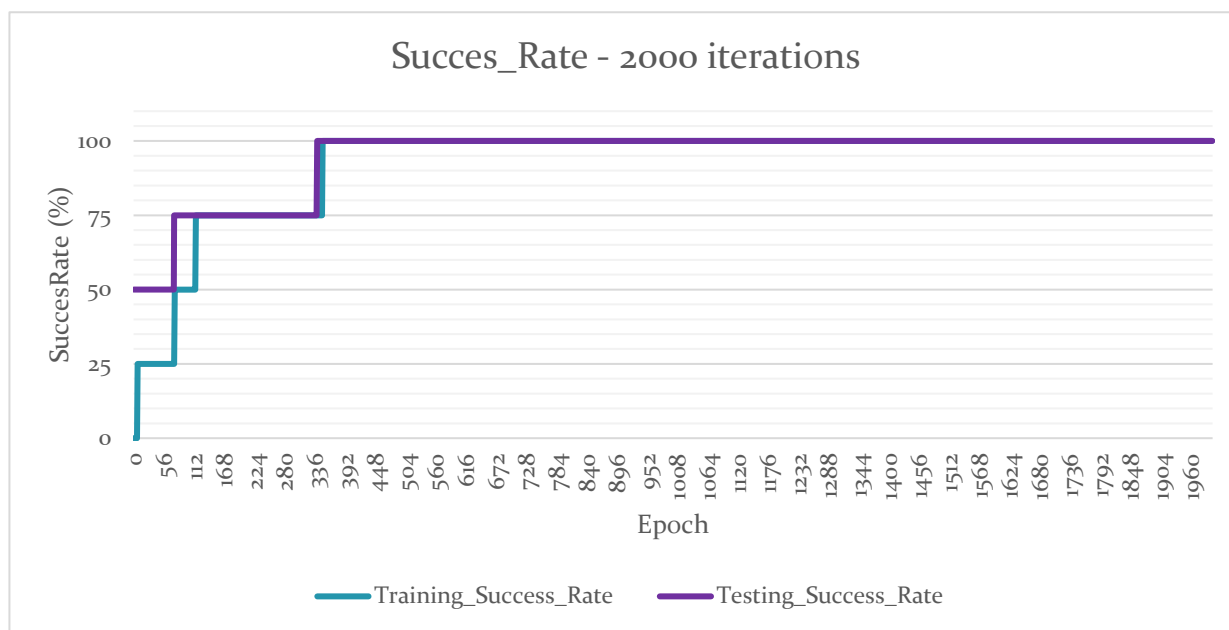
Errors.txt

Παρατηρήσεις :

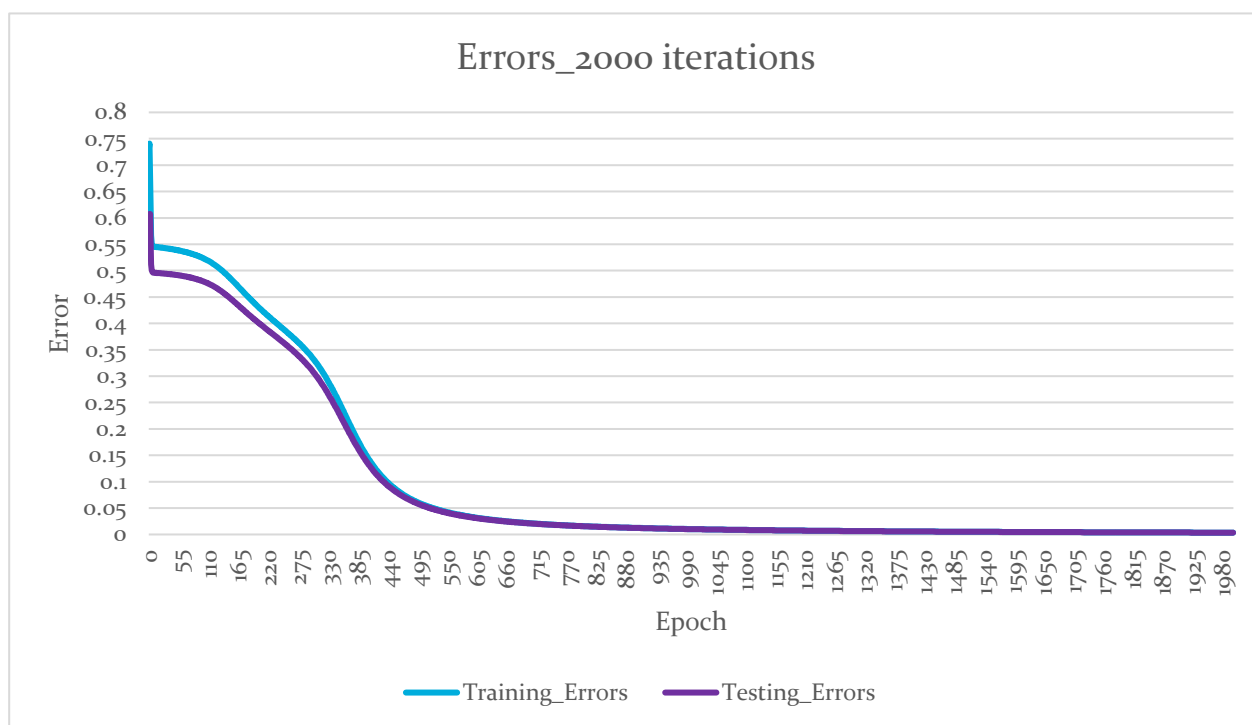
Εάν δούμε το output από το screenshot της κονσόλας παρατηρούμε ότι είναι πολύ κοντά στα επιθυμητά μας αποτελέσματα (κύκλος) . Επιπλέον, παρατηρούμε ότι τα success rate φτάνουν στο μέγιστο δυνατό 100% και τα errors ελαχιστοποιούνται σε μεγάλο βαθμό. Άρα κάνουμε εδώ την παραδοχή ότι το learning_rate & το momentum κάνουν μεγάλη αλλαγή καθώς αυξάνονται.

Γραφικές Παραστάσεις

= για τα αποτελέσματα της γραφικής πήρα τις τιμές της τελευταίας μου εκτέλεσης, όπου δεδομένα είναι το πιο πάνω παράδειγμα



Παρατηρούμε ότι το Training_Success_Rate είναι χαμηλότερο σε σχέση με το Testing_Success_Rate. Παρατηρούμε ωστόσο ότι καθώς αυξάνονται οι εποχές το δίκτυο εκπαιδεύεται πολύ καλά σε σημείο που έχει τα ίδια αποτελέσματα με το Testing_Success_Rate στο μέγιστο 100% - βρίσκει δηλαδή όλα τα επιθυμητά outputs.



Παρατηρούμε ότι το Training Error είναι υψηλότερο σε σχέση με το Testing Error. Παρατηρούμε ωστόσο ότι καθώς αυξάνονται οι εποχές το δίκτυο εκπαιδεύεται πολύ καλά σε σημείο που έχει τα ίδια αποτελέσματα με το Testing_Error όπου το error ελαχιστοποιείται.