Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Σηματων Ελεγχου και Ρομποτικης

Νευρο-ασαφης Ελεγχος και Εφαρμογες
$9^{ο}$ Εξαμηνο

Αναγνώριση Δυναμικού Συστήματος
$1^{\text{Η}}$ Εργασία

Όνομα: Αλέξανδρος Μπενετάτος
ΑΜ: 031 16077

November 14, 2020

Contents

1 Εκφώνηση

Δίνεται ένα μη-γραμμικό διακριτό δυναμικό σύστημα:

$$x_{k+1} = f(x_k),\ x_0 = [-2, 0, -1]^T$$

όπου $f : \mathcal{R}^3 \rightarrow \mathcal{R}^3$ είναι μια άγνωστη, ομαλή και φραγμένη συνάρτηση. Επίσης δίνονται μετρήσεις της απόκρισης του συστήματος για το πρώτα 1000 βήματα στο αρχείο $data\_NN.mat$, καθώς και η γραφική τους αναπαράσταση στο ακόλουθο σχήμα.

1. Να εκπαιδεύσετε ένα νευρωνικό δίκτυο που να προσεγγίζει την άγνωστη συνάρτηση $f(\cdot)$ χρησιμοποιώντας τα δεδομένα που δίνονται στο αρχείο.

2. Να αναπαραστήσετε γραφικά το σφάλμα προσέγγισης του νευρωνικού δικτύου για όλα τα δεδομένα που δίνονται στο αρχείο.

3. Επίσης, να υπολογίσετε, απεικονίσετε γραφικά και αποθηκεύσετε σε ένα αρχείο με όνομα $data\_VAL.mat$ τον πίνακα x_val που θα περιέχει τα πρώτα 200 βήματα, χρησιμοποιώντας το νευρωνικό δίκτυο που εκπαιδεύσατε με αρχική κατάσταση $x_0 = [-1.9, 0, -0.9]^T$.

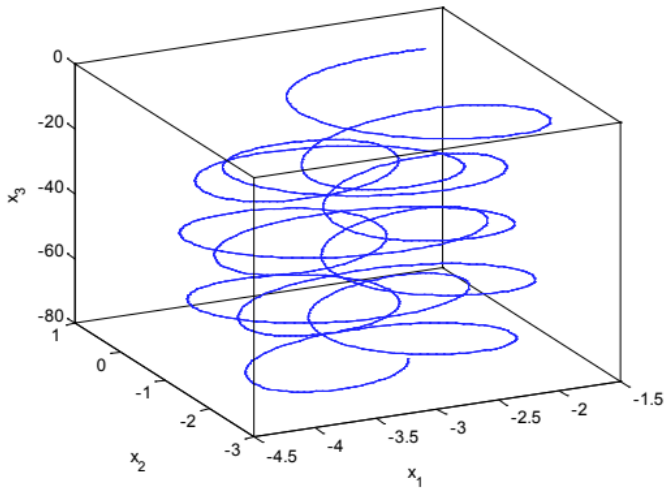

Figure 1: Γραφική απεικόνιση των δεδομένων του αρχείου $data\_NN.mat$

2 Προεπεξεργασία Δεδομένων

2.1 Διαβάζοντας τα δεδομένα

Για την υλοποίηση της άσκησης χρησιμοποιήσαμε *python* και τη βιβλιοθήκη της *pytorch* για την υλοποίηση του νευρωνικού δικτύου.

Ξεκινώντας, το πρώτο πράγμα που κάναμε ήταν να διαβάσουμε τα δεδομένα από το αρχείο και να τα χωρίσουμε σε δεδομένα εισόδου και δεδομένα εξόδου. Ο τρόπος που το κάναμε αυτό είναι πως κάθε δύο διαδοχικά "σημεία" στο αρχείο δεδομένων τα θεωρήσαμε ως ένα ζευγάρι εισόδου-εξόδου και έτσι, για τα 1001 σημεία της συνάρτησης δημιουργήσαμε 1000 ζευγάρια εισόδου εξόδου.

Στη συνέχεια, χωρίσαμε αυτά τα δεδομένα σε *test* δεδομένα και *train* δεδομένα για την εκπαίδευση του μοντέλου μας. Ο τρόπος που το πραγματοποιήσαμε αυτό είναι είναι να επιλέξουμε τυχαία έναν ποσοστών των συνολικών δεδομένων ως δεδομένων ελέγχου (*test data*) και όλων των υπολοίπων ως δεδομένων εκπαίδευσης (*train data*). Στη περίπτωσή μας, μάλιστα, επιλέξαμε αυτό το ποσοστό ως 30% των συνολικών δεδομένων.

Τέλος, για να εκπαιδεύσουμε με αυτά τα δεδομένα κάποιο μοντέλο με τη βοήθεια της *pytorch* θα πρέπει να κάνουμε *wrap* τα δεδομένα μας σε μια κλάση *torch.utils.data.Dataset*. Μάλιστα, σε αυτή τη κλάση θα δώσουμε και σαν ορίσματα συναρτήσεις οι οποίες θα εφαρμόζουν κάποιον μετασχηματισμό τόσο των δεδομένων εισόδου όσο και των δεδομένων εξόδου.

2.2 Επιλογή συνάρτησης μετασχηματισμού εισόδου και εξόδου

Για τις συναρτήσεις μετασχηματισμού των δεδομένων εισόδου αλλά και εξόδου είχαμε αρκετές επιλογές καθώς και αρκετούς συνδιασμούς μεταξύ τους. Συγκεκριμένα, για καθένα απο τα δεδομένα εισόδου και εξόδου μπορούσαμε να τα:

a) να μην εφαρμόσουμε κανένα μετασχηματισμό

b) κανονικοποιήσουμε (*normalize*) ώστε να έχουν τιμές μεταξύ 0 και 1

c) κανονικοποιήσουμε (*normalize*) ώστε να έχουν τιμές μεταξύ 0 και 10

d) κανονικοποιήσουμε (*normalize*) ώστε να έχουν τιμές μεταξύ -1 και 1

e) κανονικοποιήσουμε (*normalize*) ώστε να έχουν τιμές μεταξύ -5 και 5

f) κάνουμε ομοιόμορφα (*standarize*) ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1

g) κάνουμε ομοιόμορφα (*standarize*) ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 10

(η επιλογή του 10 και του 5 υποδεικνύει απλά *scaling* των τιμών που, για ομοιομορφία ανάμεσα στους μετασχηματισμούς επιλέχθηκε ως 10 για όλους)

Τελικά, δοκιμάζοντας όλους του συνδυασμούς και εκπαιδεύοντας το δίκτυο (όπως θα αναλύσουμε παρακάτω) για καθένα τους υπολογίσαμε τα παρακάτω:

Input Transf	*Output Transf*	*Best MSE*	*Epoch of Best MSE*	*Mean MSE* (140 *best*)	*std MSE* (140 *best*)
norm	*no transf*	0.8531	287	67.296	87.150
norm	*norm*	4.6417	290	46.745	43.368
norm	*norm scaled*	0.6054	214	19.521	13.943
norm	*stan*	6.9114	212	27.019	13.085
norm	*stand scaled*	0.7570	178	113.288	83.338
norm	$-1, 1$ *norm*	2.1285	296	8.310	4.099
norm	$-1, 1$ *norm scaled*	0.7188	168	11.043	8.581
stand	*no transf*	2.4977	135	154.997	168.083
stand	*norm*	75.6459	178	$88{\cdot}10^{13}$	$568{\cdot}10^{13}$
stand	*norm scaled*	11.0633	250	42.880	18.151
stand	*stand*	153.2007	4	266.759	46.099
stand	*stand scaled*	0.9872	271	19.510	12.659
stand	$-1, 1$ *norm*	80.0935	286	148.052	55.657
stand	$-1, 1$ *norm scaled*	1.7514	252	34.884	20.227

Table 1: Αποτελέσματα από την εκπαίδευση του δικτύου για διαφορετικούς μετασχηματισμούς εισόδου-εξόδου

Με βάση λοιπόν τα παραπάνω αποτελέσματα επιλέξαμε να εφαρμόσουμε κανονικοποίηση των δεδομένων εισόδου μεταξύ του 0 και του 1 ενώ για τα δεδομένα εξόδου τα κανονικοποιήσαμε μεταξύ του 0 και του 20. Το *scaling* κάνει αρκετή διαφορά καθώς μας βοηθάει όταν υπολογίζουμε το $MSE\ loss$ να λαμβάνουμε ακόμα και για σχετικά μικρά σφάλματα φαινομενικά ικανοποιητικές τιμές ώστε να εκπαιδεύσουμε το νευρωνικό. Αυτό κάνει αρκετή διαφορά ιδιαίτερα λόγο της συνιστώσας x_3 των δεδομένων που, όταν κανονικοποιηθεί, μοιάζει σχεδόν με γραμμική συνάρτηση. Φυσικά, θα μπορούσαμε να αλλάξουμε το $loss\ function$ ώστε να περιλαμβάνει κάποιο *scaling* αλλά τελικά ήταν πιο εύκολο να το ενσωματώσουμε στον μετασχηματισμό των δεδομένων.

3 Εκπαίδευση του Δικτύου

3.1 Επιλέγοντας *loss function* και *optimizer*

Έχοντας ένα πρόβλημα *regression* ουσιαστικά οι επιλογές μας για την *loss function* ήταν το μέσο τετραγωνικό σφάλμα (MSE) καθώς και το μέσο απόλυτο σφάλμα (MAE). Γενικότερα, τόσο το MSE όσο και το MAE θα έβγαζαν ικανοποιητικά αποτελέσματα, ωστόσο, το MSE δημιουργεί κατά κανόνα έναν πιο ομαλό και κυρτό χώρο όπου υπολογίζουμε τα *gradients* σε σχέση με το MAE και για αυτό θεωρήθηκε προτιμότερο.

Όσον αφορά τον *optimizer* επιλέξαμε έναν που έχει δυναμικό *learning rate*, που το *learning rate* δηλαδή αλλάζει όσο περνάνε οι εποχές και μάλιστα ανεξάρτητα ανά παράμετρο. Ο πιο γνωστός, ίσως, τέτοιος *optimizer* είναι ο *Adam*, ωστόσο, επιλέξαμε μια παραλλαγή του, τον *Adabound*, ο οποίος θέτει ορισμένα δυναμικά όριο στις τιμές των *learning rate* οδηγώντας, ουσιαστικά, από τον *Adam* στον κλασικό SGD όταν αυτά τα όρια μικρύνουν αρκετά οδηγώντας σε καλύτερά αποτελέσματα.

3.2 Το νευρωνικό δίκτυο

Για την αρχιτεκτονική του νευρωνικού δικτύου η λογική ήταν να ξεκινήσουμε από ένα δίκτυο με ένα μόνο *hidden layer* νευρώνων και αν αυτό δεν έχει αρκετό *learning capacity* να προχωρήσουμε σε πιο βαθιά δίκτυα. Ωστόσο, με ένα μόνο *layer* μεγέθους 8 νευρώνων επιτύχαμε εξαιρετικά αποτελέσματα στα *training data* συνεπώς δεν υπήρχε πρόβλημα μαθησιακής χωρητικότητας του δικτύου.

Για συνάρτηση ενεργοποίησης χρησιμοποιήσαμε την $ReLU$. Κάποιοι από τους λόγους που η $ReLU$ προτιμάται πια για όλα σχεδόν τα SOA μοντέλα, τουλάχιστον για τα *hidden layers*, έχουν να κάνουν με υπολογιστική απλότητα της, την γραμμική, εκτός του μηδενός, συμπεριφορά της (μειώνει το πρόβλημα των εξαφανιζόμενων *gradients*) και την αραιότητα (*sparsity*) της αφού επιτρέπει εύκολα σε νευρώνες του δικτύου να πάρουν μηδενικές τιμές δημιουργώντας αραιές αναπαραστάσεις (για παράδειγμα, αν θεωρήσουμε τις τιμές εξόδου των νευρώνων ενός *layer* σαν ένα διάνυσμα, όσο πιο πολλά μηδενικά έχουμε σε αυτό τόσο πιο αραιή αναπαράσταση έχουμε) που φαίνεται να είναι ευεργετικές για το νευρωνικό.

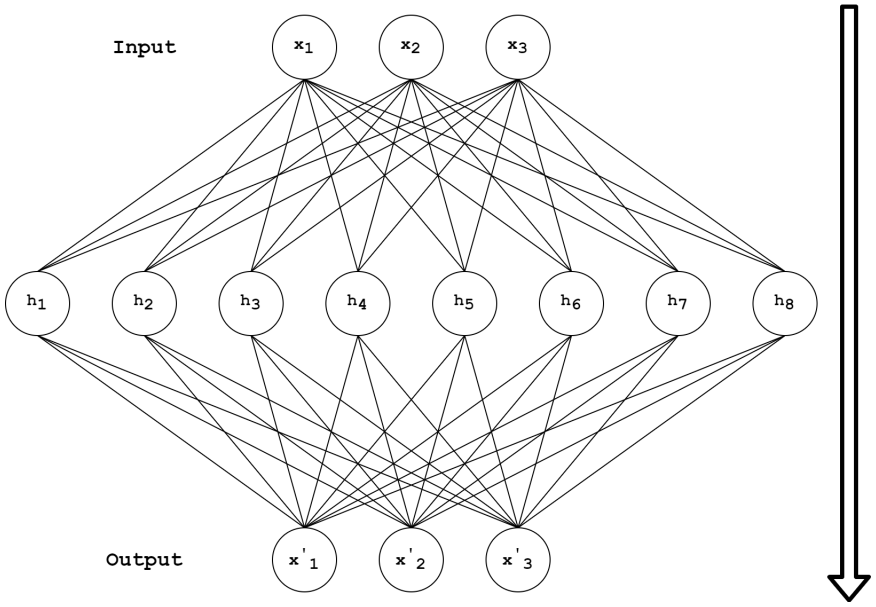

Figure 2: Γραφική απεικόνιση της αρχιτεκτονικής του νευρωνικού δικτύου

3.3 Επιλέγοντας το καλύτερο μοντέλο

Καθώς εκπαιδεύουμε το μοντέλο για την ελαχιστοποίηση του σφάλματος στην πρόβλεψη του επόμενου σημείου είναι πιθανό το μοντέλο που καταλήγουμε στο τέλος της εκπαίδευσης να μην είναι το καλύτερο μοντέλο που συναντήσαμε κατά τη διάρκεια της εκπαίδευσης. Έτσι, αρχικά είχαμε επιλέξει να αποθηκεύουμε κάθε φορά το μοντέλο με το ελάχιστο *loss*. Ωστόσο, παρατηρήσαμε πως το μοντέλο αυτό δεν ήταν το μοντέλο με το ελάχιστο *loss* κατά την πρόβλεψη ενός σημείου και από αυτή τη πρόβλεψη του επόμενου σημείο κ.ο.κ. για τα 1000 σημεία για παράδειγμα των δεδομένων μας.

Συνεπώς, για να βελτιστοποιήσουμε την απόδοση του δικτύου κατά τη συνεχόμενη πρόβλεψη σημείων επιλέξαμε σε κάθε εποχή να κάνουμε *evaluate* το δίκτυο με βάσει την πρόβλεψη 200 σημείων δεδομένου μόνο ενός (του πρώτου) και να κρατάμε το μοντέλο που τελικά κατάφερε να έχει τη μικρότερη απόκλιση από τα δεδομένα μας.

Ωστόσο, με αυτό τον τρόπο, κατά μια έννοια, χρησιμοποιούμε και τα δεδομένα δοκιμής (*test data*) στην εκπαίδευση το οποίο είναι λάθος. Αν θέλαμε να είμαστε 100% τυπικοί, θα έπρεπε να χωρίζαμε στην αρχή τα δεδομένα που έχουμε

σε *train* και *test* όχι τυχαία αλλά τα πρώτα 700 σαν δεδομένα εκπαίδευσης και τα υπόλοιπα 300 σαν δεδομένα δοκιμής ή απλά με τρόπο ώστε τα 200 πρώτα σημεία των δεδομένων να ανήκουν στα *train data*. Έτσι, στις συνεχόμενες προβλέψεις που αναφέραμε θα ελέγχαμε τη πρόβλεψη 200 σημείων δεδομένου του πρώτο σε σχέση με τα 200 από *train* δεδομένα που είχαμε.

Καθώς, όμως, είχαμε ήδη γράψει τον κώδικα για όλα τα παραπάνω, και εξ' αιτίας της, ουσιαστικά, μηδενικής διαφοροποίησης που θα είχε το αποτέλεσμα επιλέξαμε να μην αλλάξουμε όλο τον κώδικα ξανά για αυτό.

4 Αξιολόγηση Μοντέλου

4.1 Απλή πρόβλεψη επομένου για κάθε σημείο

Ακολουθώντας, λοιπόν, την παραπάνω διαδικασία, εκπαιδεύσαμε το μοντέλο για 300 εποχές και κρατήσαμε αυτή που είχε το ελάχιστο *loss* υπό την έννοια που περιγράψαμε παραπάνω. Αυτό το μοντέλο ήταν το μοντέλο της εποχής 151 με όπου το σφάλμα για τα *test data* είναι $MSE = 0.0023$.

Στο επόμενο διάγραμμα, λοιπόν, απεικονίζουμε την επίδοση του νευρωνικού κατά τη πρόβλεψη, για καθένα από τα σημεία των δεδομένων εισόδου, του επόμενου σημείου (για όλα τα 1000 σημεία των δεδομένων εισόδου). Στην τελευταία γραμμή φαίνεται ο λογάριθμος του *error* ανά συνιστώσα (θεωρώ $x = x_1, y = x_2, z = x_3$), στην προτελευταία γραμμή βλέπουμε τις τιμές κάθε που προβλέψαμε καθώς και της πραγματικές τιμές στο ίδιο διάγραμμα για κάθε συνιστώσα χωριστά ενώ στη πρώτη γραμμή βλέπουμε στο αριστερό διάγραμμα τις προβλέψεις, στο μεσαίο τις πραγματικές τιμές ενώ στο δεξιά τον λογάριθμο του *error* για όλες τις συνιστώσες.

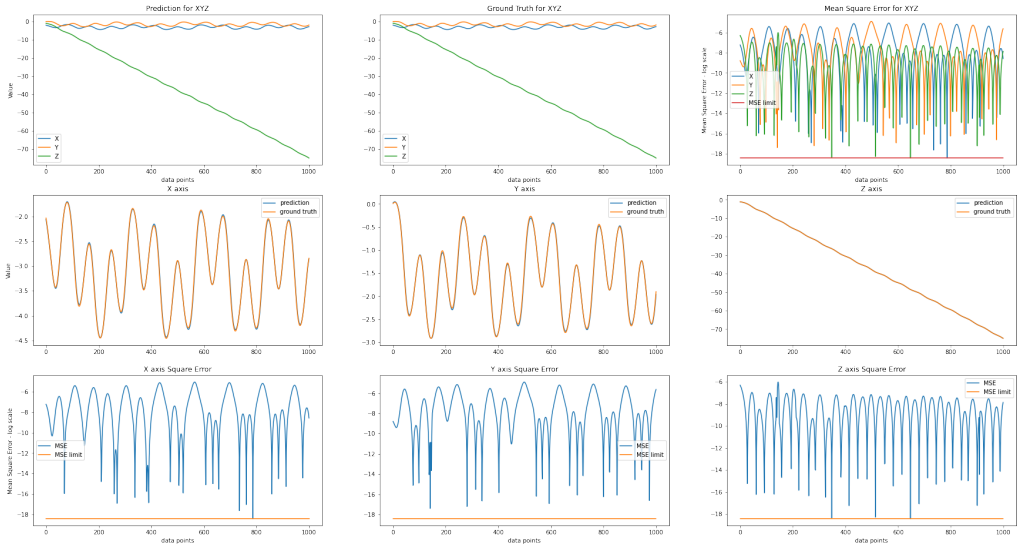

Figure 3: Γραφική απεικόνιση τον προβλέψεων και το σφαλμάτων από την πρόβλεψη ενός μόνο μελλοντικού σημείου κάθε φορά

4.2 Πρόβλεψη 200 σημείων δεδομένου ενός

Στο επόμενο διάγραμμα, απεικονίζουμε την επίδοση του νευρωνικού κατά τη πρόβλεψη 200 σημείων το ένα μετά το άλλο δεδομένου μόνο του πρώτου σημείου. Για αυτή τη περίπτωση, το σφάλμα ανάμεσα στις προβλέψεις και στις πραγματικές τιμές είναι σαφώς μεγαλύτερο και ίσο με $MSE = 0.9738$.

Στην τελευταία γραμμή φαίνεται το *error* ανά συνιστώσα (θεωρώ $x = x_1, y = x_2, z = x_3$), στην προτελευταία γραμμή βλέπουμε τις τιμές κάθε που προβλέψαμε καθώς και της πραγματικές τιμές στο ίδιο διάγραμμα για κάθε συνιστώσα χωριστά ενώ στη πρώτη γραμμή βλέπουμε στο αριστερό διάγραμμα τις προβλέψεις, στο μεσαίο τις πραγματικές τιμές ενώ στο δεξιά το *error* για όλες τις συνιστώσες.

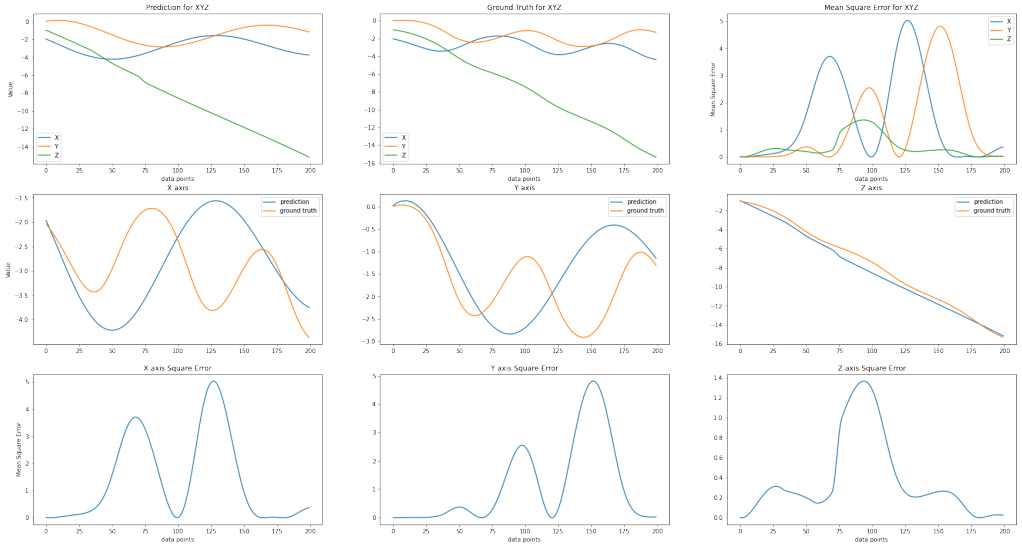

Figure 4: Γραφική απεικόνιση τον προβλέψεων και το σφαλμάτων από την πρόβλεψη 200 σημείων δεδομένου ενός

5 Κώδικας

Μπορείτε να βρείτε ένα *repository* με όλο τον κώδικα της άσκησης καθώς και τα αρχεία εισόδου-εξόδου στο ακόλουθο *link*:

https://github.com/alex-bene/neurofuzzy-control/tree/master/Function Approximation Neural Network - 1st Assignment