|  |  |
| --- | --- |
| Poseidon:Users:egpapado:Documents:MyDocuments: NTUA Συμβούλιο Ιδρύματος:Υποδομή ΣΙ:Web Site:pyrforos-digamma.eps | **ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**  ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΩΝ ΚΑΤΕΡΓΑΣΙΩΝ |

Τεχνητά Νευρωνικά Δίκτυα στα Συστήματα Κατεργασιών

**3D printing optimization**

**genetic algorithms**

Αντώνιος Καντούνιας (mc18004)

Ελευθέριος Καντούνιας (mc18025)

AΘHNA 2022

Περιεχόμενα

[1 Εισαγωγή 4](#_Toc119635500)

[1.1 Ιδέα 4](#_Toc119635501)

[1.2 Αρχή λειτουργίας 4](#_Toc119635502)

[1.3 Σκοπός εργασίας 5](#_Toc119635503)

[2 προεπεξεργασία πειραματικών δεδομένων 6](#_Toc119635504)

[2.1 Εισαγωγή 6](#_Toc119635505)

[2.2 Στοιχεία θεωρίας 6](#_Toc119635506)

[2.3 Υλοποίηση 6](#_Toc119635507)

[3 Συνδυασμός των παραμέτρων εισόδου 7](#_Toc119635508)

[3.1 Εισαγωγή 7](#_Toc119635509)

[3.2 Στοιχεία θεωρίας 7](#_Toc119635510)

[3.2.1 Ανάλυση ευαισθησίας 7](#_Toc119635511)

[3.2.2 Fuzzy Curves 7](#_Toc119635512)

[3.2.3 MSE 7](#_Toc119635513)

[3.2.4 Συσχέτιση μέσο απλής γραμμικής παλινδρόμησης 7](#_Toc119635514)

[3.3 Υλοποίηση 8](#_Toc119635515)

[3.3.1 Μεθοδολογία 8](#_Toc119635516)

[3.3.2 Αποτελέσματα 8](#_Toc119635517)

[4 Χωρισμός δεδομένων / επικύρωση 10](#_Toc119635518)

[4.1 Εισαγωγή 10](#_Toc119635519)

[4.2 Cross-Validation techniques 10](#_Toc119635520)

[4.2.1 Hold-out cross-validation (early stopping) 10](#_Toc119635521)

[4.2.2 K-fold cross-validation 10](#_Toc119635522)

[4.3 Χωρισμός δεδομένων 11](#_Toc119635523)

[5 κριτήριο ελέγχου της απόδοσης 12](#_Toc119635524)

[5.1 Εισαγωγή 12](#_Toc119635525)

[5.2 Στοιχεία θεωρίας 12](#_Toc119635526)

[5.3 Υλοποίηση 12](#_Toc119635527)

[6 Αρχιτεκτονική νευρωνικών δικτύων 13](#_Toc119635528)

[6.1 Εισαγωγή 13](#_Toc119635529)

[6.2 πιθανές προσεγγίσεις 13](#_Toc119635530)

[6.3 Υλοποίηση 13](#_Toc119635531)

[6.3.1 Αποτελέσματα raw data 14](#_Toc119635532)

[6.3.2 Αποτελέσματα processed data 16](#_Toc119635533)

[7 Βιβλιογραφία 18](#_Toc119635534)

[Παράρτημα 19](#_Toc119635535)

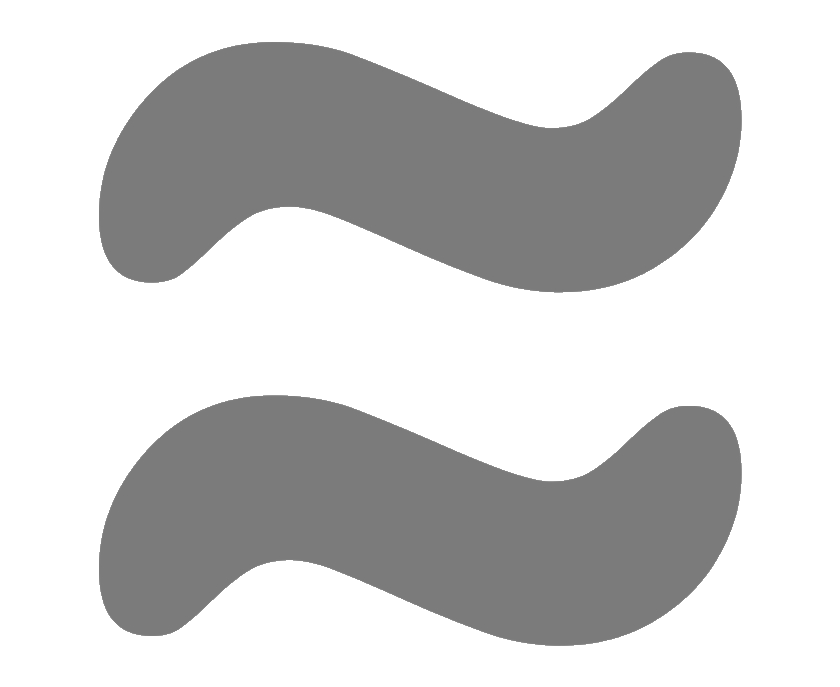
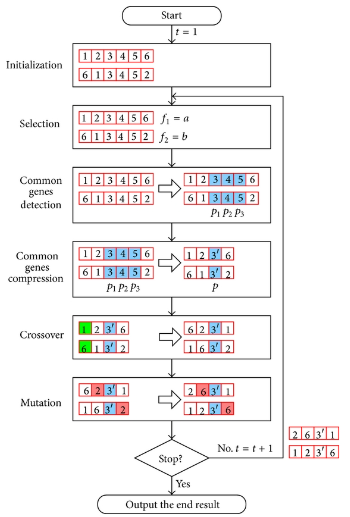
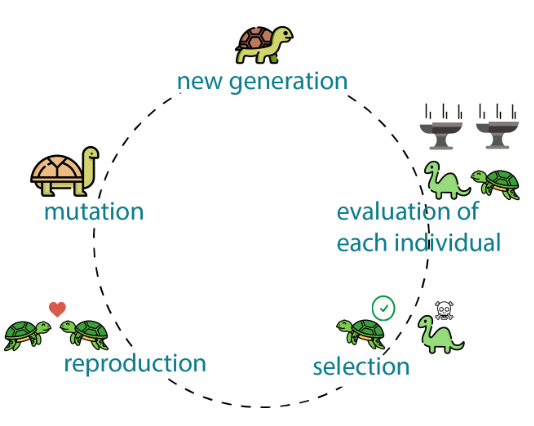
# Εισαγωγή

## Ιδέα

H βελτιστοποίηση σαν ευρύτερη έννοια απαρτίζεται από μαθηματικές ή αλγοριθμικές μεθόδους οι οποίες στόχο έχουν την επίλυση ποσοτικών προβλημάτων με βέλτιστο τρόπο. Προβλήματα στα οποία τέτοιοι μέθοδοι βρίσκουν εφαρμογή σε, φυσική, βιολογία, μηχανική, οικονομικά, επιχειρήσεις και αλλά. Evolutionary Algorithms (EA) αποτελούν αλγορίθμους οι οποίοι είναι εμπνευσμένοι από τις μεθόδους βελτιστοποίησης που συναντάται στην φύση όπως, particle swarm, ant colony, genetic algorithm, biogeography-based optimisation. Oι αλγόριθμοι αυτοί χαρακτηρίζονται από robustness ενώ ή στοχαστικότατα που τους διέπει εγγυάται την ικανότητα εύρεσης βέλτιστης λύσης. Το σημαντικότερο μειονέκτημα τους, είναι το μεγάλο υπολογιστικό κόστος που απαιτείτε για την υλοποίηση τους. Στην συγκεκριμένη εργασία θα υλοποιηθεί συγκεκριμένα, ίσως η πιο διαδεδομένη μέθοδος, των γενετικών αλγορίθμων.

## Αρχή λειτουργίας

Ο γενετικός αλγόριθμος βασίζεται στην δαρβινική θεωρεία. Η διαδικασία εκκινεί από την τυχαία δημιουργία χρωμοσωμάτων – γονέων (1η γενιά). Τα γονίδια αποτελούν τα δομικά στοιχεία ενός χρωμοσώματος και ουσιαστικά εκφράζουν τις αριθμητικές τιμές των προς βελτιστοποίηση design variables. Στο κάθε χρωμόσωμα αντιστοιχίζεται μια τιμή καταλληλόλητας (fitness value) σύμφωνα με την απόδοση τους στο πρόβλημα (τιμή της αντικειμενικής συνάρτησης). Τα καλύτερα χρωμοσώματα έχουν περισσότερες πιθανότητες να αναπαραχθούν, δηλαδή να ανταλλάξουν πληροφορίες δημιουργώντας καινούργια χρωμοσώματα με μεικτά γονίδια των δυο. Το αποτέλεσμα της αναπαραγωγής είναι τα offsprings, των οποίον τα γονίδια διατρέχουν μικρή πιθανότητα μετάλλαξης. Στην περίπτωση που τα offsprings είναι λιγότερα, το πληθυσμιακό χάσμα καλύπτεται με χρωμοσώματα προηγούμενης γενιάς. Συνήθως επιλέγονται τα καλύτερα (elitist strategy).



## Σκοπός εργασίας

Στην παρούσα εργασία, στόχος είναι η εύρεση των βέλτιστων παραμέτρων εκτύπωσης :

α) Πάχος κάθε στρώματος

β) Πάχος τοιχώματος

γ) Ποσοστό γεμίσματος

δ) Γεωμετρικό μοτίβο γεμίσματος

ε) Θερμοκρασία του ακροφυσίου

στ) Θερμοκρασία της πλατφόρμας εκτύπωσης

ζ) Ταχύτητα εκτύπωσης

η) Υλικό

θ) Ταχύτητα του ανεμιστήρα ψύξης

Οι παραμέτροι αυτές θα επιλεχθούν με στόχο την βελτιστοποίηση των :

1. Τραχύτητα επιφάνειας
2. Όριο αντοχής
3. Επί τοις εκατό επιμήκυνση

Το μοντέλο το οποίο θα παίξει τον ρόλο της αντικειμενικής συνάρτησης, είναι το τεχνητό νευρωνικό δίκτυο που κατασκευάστηκε και εκπαιδεύτηκε στην προηγούμενη εργασία. Η βελτιστοποίηση θα γίνει με χρήση γενετικού αλγόριθμου, του οποίου η απόδοση (ποιότητα λύσης , ταχύτητα σύγκλησης) θα διερευνηθεί συναρτήσει των βασικών του παραμέτρων:

* Μέγεθος πληθυσμού
* Αριθμός γενεών
* Ποσοστό ελιτισμού
* Μέθοδος μετάλλαξης
* Μέθοδος διασταύρωσης

Τέλος θα παρουσιαστεί η επίδραση της συνάρτησης καταλληλότητας στο τελικό βέλτιστο αποτέλεσμα .

# Βασικά στοιχεία γενετικών αλγορίθμων

## Εισαγωγή

Οι γενετικοί αλγόριθμοι αποτελούνται από κάποια κύρια χαρακτηριστικά στα οποία πρέπει να δίνεται βάση κατά τη χρήση τους προκειμένου αυτοί να καθίστανται λειτουργικοί και αποδοτικοί.

## επιλογή παραμέτρων

Οι επιλογή των παραμέτρων του αλγορίθμου είναι μείζονος σημασίας για την καλή του λειτουργεία. Παράμετροι οι οποίες είναι σταθερές στον χρόνο επιλέγονται εκ των προτέρων. Στρατηγικές δειγματοληψίας όπως, latin hypercube sampling, grid search, καθώς και χρήση στατιστικών μεθόδων μπορούν να πλαισιώσουν αυτήν την προσπάθεια [1].Η στρατηγική Meta-Genetic algorithm κάνει optimization τις παραμέτρους του αλγόριθμού, Ωστόσο το κόστος της είναι απαγορευτικό [2].

Η δυναμική αλλαγή ορισμένων παραμέτρων (π.χ. mutation rate) κατά την βελτιστοποίηση μπορεί να βελτιώσει εσθήτα την διαδικασία, καθώς η προσαρμογή γίνεται με βάση το feedback που λαμβάνεται από την συμπεριφορά του προβλήματος. Σημαντικές τέτοιου είδους τεχνικές είναι, ο κανόνας του Rechenberg [3] και η στρατηγική Self-adaptation [4], η οποίες επιδρούν στα cross over και mutation χαρακτηριστικά πριν τα εφαρμόσουν στους απογόνους.

## Πολυτροπικά συστήματα

Η εύρεση λύσης σε προβλήματα που χαρακτηρίζονται από μεγάλες διακυμάνσεις και πολλά τοπικά βέλτιστα, αποτελεί απαιτητική διαδικασία για τους αλγορίθμους βελτιστοποίησης. Μέθοδοι με βάση το μέγεθος του πληθυσμού καθώς και των έλεγχο των παραμέτρων, παρέχουν ικανοποιητικά αποτελέσματα. Τα λεγόμενα restart strategy’s πυροδοτούν πολλαπλά ξεκινήματα από διαφορετικά σημεία του χώρου των λύσεων [5]. Το global mutation control, το οποίο αναφέρθηκε και παραπάνω επίσης βρίσκει εφαρμογή σε τέτοιου είδους προβλήματα προσαρμόζοντας τα explore και exploit χαρακτηριστικά του αλγορίθμου. Ένας άλλος αποτελεσματικός τρόπος ώστε να διατηρείτε η ποικιλομορφία των λύσεων και να μην παγιδεύονται τοπικά, είναι η χρήση κοινών fitnesses σε ομάδες λύσεων. Ακόμα η novelty search μέθοδος επιτυγχάνει τον στόχο της, επικροτώντας λύσεις με βάση το πόσο καινοτομικά χαρακτηριστικά έχουν. Τέλος στρατηγικές Niching, συνήθως μέσο clustering αλγορίθμων, εστιάζουν στην εύρεση όσο το δυνατών περισσοτέρων ‘’λεκανών’’ στο πρόβλημα, ενώ στην συνέχεια υλοποιείτε μετέπειτα έρευνα σε αυτές [6].

Όπως φαίνεται τα προβλήματα με τέτοιου είδους χαρακτηριστικά, είναι προσεγγίσιμα μέσο διαφόρων μεθόδων τον οποίων το κοινό και απαραβίαστο χαρακτηριστικό είναι το trade off που πρέπει να γίνει μεταξύ exploration και exploitation.

## Εισαγωγή περιορισμών

Τα περισσότερα πραγματικά πρόβλημα βελτιστοποίησης βασίζονται στην ύπαρξη περιορισμών. Ο πιο απλός και επιτυχημένος μηχανισμός εισαγωγής περιορισμών, είναι το death penalty. Ο συγκεκριμένος μηχανισμός διαγράφει οποιαδήποτε λύση δεν τηρεί τους περιορισμούς ώσπου να βρει κάποια που να το κάνει. Τα penalty functions έχουν την δυνατότητα να δουλεύουν εντός του χώρου των πειρασμών. Κατευθύνουν το πρόβλημα στον επιτρεπόμενο χώρο λύσεων μειώνοντας το fitness λύσεων που δεν τηρούν τους περιορισμούς. Η ένταση που συμβαίνει αυτό καθορίζεται από το penalty factor [7]. Τα decoder functions έχουν την δυνατότητα να μετατρέπουν ένα περίπλοκο περιορισμένο χώρο λύσης σε έναν πιο εύκολο μετασχηματισμένο χώρο, όπου η αναζήτηση είναι λιγότερο δύσκολη [8]. Μια άλλη τεχνική αποτελεί την αντιμετώπιση των περιορισμών σαν objectives και την διαχείριση του προβλήματος σαν multi-objective. Δημιουργούνται έτσι pareto στα οποία ο χρήστης μπορεί να διαλέξει την λύση που του κάνει [9]. Τέλος η επιστράτευση metamodel για το detection λύσεων που πιθανός έγκειται σε περιορισμούς, μπορεί να μειώσει σημαντικά τα function calls που οδηγούν σε ανέφικτες λύσεις.

## Προβλήματα πολλαπλών στόχων

Τα περισσότερα προβλήματα στον πραγματικό κόσμο είναι πολύπλοκα και το ενδιαφέρον βρίσκεται στην βελτίωση προδιαγραφών του. Αν οι προδιαγραφές αυτές είναι αντικρουόμενες, τότε επιβάλλεται η χρήση multi-objective optimization (έτσι η λύση αποτελεί ένα pareto και δεν είναι μονοσήμαντη). Ο NSGA-ΙΙ μεγιστοποιεί την απόσταση μεταξύ των λύσεων στο pareto ώστε να αποφευχθεί ο συνωστισμός. Το rake selection παρόμοια με τον NSGA-ΙΙ κατασκευάζει κάποιες απέχουσες ευθείες με βάση τις οποίες γίνεται το selection [10]. Μια άλλη τεχνική αποτελεί το hypervolume indicator. Το hypervolume indicator είναι το εμβαδόν που σχηματίζεται με βάση ένα επιλεγμένο reference point (οριακές τιμές). Η μεγιστοποίηση του σπρώχνει προς τα μέσα το pareto των λύσεων [11]. Τέλος είναι εφικτή η χρήση υβριδικών μεθόδων οι οποίες επωφελούνται από τα χαρακτηριστικά τον επιμέρους μεθόδων (π.χ. rakes in solution space μαζί με hypervolume maximization) .

## Χρήση μηχανικής μάθησης

Η μηχανική μάθηση είναι κομμάτι ενός μεγάλου μέρους μεθοδολογιών σχετικές με, πρόβλεψη μοντέλων και μετατροπή χώρων μεγάλων διαστάσεων σε μικρότερους. Η Χρήση meta-models μπορεί να εξοικονομήσει evaluations αντικαθιστώντας την αντικειμενική και τις συναρτήσεις περιορισμών όπου χρειάζεται [12]. Τέλος unsupervised μέθοδοι βρίσκουν εφαρμογή σε dimension reduction και clustering [13].Γενικότερα το υβρίδιο γενετικών αλγορίθμων και μηχανικής μάθησης είναι πλέον το state of the art .

# Ανάλυση παραμέτρων

## Εισαγωγή

Στο πλαίσιο της παρούσας εργασίας έγινε η διερεύνηση των βασικότερων παραμέτρων οι οποίες θεωρούνται η κυριότεροι παράγοντες που καθορίζουν την απόδοση ενός γενετικού αλγορίθμου. Οι παραμέτροι που εξετάσθηκαν δίνονται ονομαστικά:

* Crossover population to total population ratio (Ποσοστό πληθυσμού που διασταυρώνεται)
* Elitism (Πιθανότητα διασταύρωση ελιτιστικών λύσεων)
* Population size (Μέγεθος πληθυσμού)
* Mutation probability rate (Πιθανότητα μετάλλαξης)

## Υλοποίηση

Σημειώνεται ότι βελτιστοποίηση έγινε με χρήση αντικειμενικής συναρτήσεις προκύπτει από το άθροισμα αδιαστατοποιημένων προδιαγραφών αντοχής, ποιότητας επιφανείας, και επιμήκυνσης, πολλαπλασιασμένων με το βάρος 1.

Στο πλαίσιο της μοντελοποιήσης των μεταβλητών σχεδιασμού έχουμε ότι οι αριθμητικές παραμέτροι εισήχθησαν στην βελτιστοποίηση ως doubles ενώ οι μη αριθμητικές παραμέτροι ως boolean values.

Οι περιορισμοί που τέθηκαν αποτελούν ήταν οι εξής:

1. Οι προδιαγραφές strength και roughness παίρνουν πάντα θετικές τιμές.
2. Οι προδιαγραφές που προκύπτουν από το νευρωνικό δίκτυο δεν μπορούν σε τιμές να υπερβαίνουν τις μέγιστες και τις ελάχιστες τιμές που συναντώνται στην database.

Και οι δύο παραπάνω περιορισμοί σε περίπτωση που δεν τηρούνται, υποδηλώνουν ότι το νευρωνικό δίκτυο χαρακτηρίζεται από ελλιπή εκπαίδευση στην συγκεκριμένη περιοχή και ότι τα αποτελέσματα δεν μπορούν να θεωρηθούν έμπιστα για την συνέχεια της βελτιστοποίησης στην συγκεκριμένη περιοχή. Επομένως τέτοιες περιοχές αποκρύπτονται.

## Crossover population to total population ratio

Η γενική ιδέα της επιλογής ενός κατάλληλου ποσοστού πληθυσμού μια γενιάς που πρόκειται να διασταυρωθεί για την παραγωγή των απογόνων της επόμενης γενιάς έγκειται στο κατά πόσο υπάρχει επαρκής πλουραλισμός των απογόνων από την μια γενιά στην επόμενη.

Εικόνα 1: Παρουσίαση της πορείας της βελτιστοποίησης αναλόγως το ποσοστό των γονέων που διασταυρώνονται.

Παρατηρούμε ότι καλύτερη συμπεριφορά σύγκλησης παρέχει η επιλογή της παραμέτρου crossover στο 90%. Η συγκεκριμένη επιλογή φαίνεται να δίνει καλύτερα αποτελέσματα και κατά την διάρκεια της σύγκλησης αλλά και στο πέρας αυτής.

Σε γενικές γραμμές δεν μπορούμε να εξάγουμε κάποιον γενικό κανόνα σχετικά με την καταλληλότερη στρατηγική διασταύρωσης καθώς παρατηρούμε ότι και η επιλογή της παραμέτρου στο 60% δίνει επίσης καλά αποτελέσματα.

## Elitism

Το ποσοστό του ελιτισμού για τον αλγόριθμο καθορίζει το κατά πόσο η απόγονοι που ο αλγόριθμος επιλέγει να διασταυρώσει έχουν δώσει περισσότερο αξιόλογα αποτελέσματα. Ο ελιτισμός στην ουσία καθορίζει πόσο στοχαστικός η αιτιοκρατικός είναι ο αλγόριθμός μας, με όλα τα θετικά και αρνητικά που εμπεριέχουν η δύο αυτές στρατηγικές βελτιστοποίησης.



Εικόνα 2: Παρουσίαση της πορείας της βελτιστοποίησης για της διάφορες επιλογές ελιτισμού

Παρατηρούμε ότι υψηλό ποσοστό ελιτισμού δίνει αρχικά ικανοποιητική πορεία στην βελτιστοποίηση αλλά οδηγεί σε τοπικά ακρότατα από τα οποία ο αλγόριθμος δεν μπορεί να ξεφύγει.

Αντίθετα για μικρό ποσοστό ελιτισμού έχουμε περισσότερο αργή πορεία, αρχικά, βελτιστοποίησης αλλά το αποτέλεσμα στο οποίο ο αλγόριθμος συγκλίνει τελικά είναι καλύτερο.

## Population size

Για συγκεκριμένη υπολογιστική ισχύ που διαθέτουμε το population size μας καθορίζει στην ουσία πόσες γενιές μπορούμε να πράξουμε. Όσο περισσότερες γενιές θέλουμε να πράξουμε τόσο η δυναμική της κάθε γενιάς μειώνεται.

Εικόνα 3: Παρουσίαση της πορείας σύγκλησης για ορισμένη υπολογιστική ισχύ. Και μεταβλητό μέγεθος πληθυσμών.

Παρατηρούμε ότι όσο το μέγεθος των πληθυσμών αυξάνει τόσο ο αριθμός των παραγόμενων γενεών μειώνεται.

Τελικά βέλτιστη συμπεριφορά σε ότι αφορά το τελικό αποτέλεσμα δίνει η επιλογή επαρκώς μεγάλου μεγέθους πληθυσμών (75 άτομα). Αυτό βεβαία δεν θεωρείτε σκόπιμο να υπερβαίνει ένα όριο (100 άτομα).

## Mutation probability rate

Τέλος εξετάζεται η επίδραση του mutation probability στην πορεία της βελτιστοποίησης. Όσο μεγαλύτερο το mutation probability τόσο πιο στοχαστική καθιστάτε η λειτουργεία του αλγορίθμου.



Εικόνα 4: Παρουσίαση της πορείας της βελτιστοποίησης για διάφορα mutation probabilities

Στο συγκεκριμένο πρόβλημα φαίνεται ότι το mutation δεν έχει ιδιαίτερη επίδραση στην πορεία και το αποτέλεσμα της βελτιστοποίησης.

# Βιβλιογραφία

1. W. de Landgraaf, A. Eiben, and V. Nannen. Parameter calibration using meta-algorithms. In IEEE Congress on Evolutionary Computation (CEC), pages 71–78, 2007
2. W. A. de Landgraaf, A. E. Eiben, and V. Nannen. Parameter calibration using meta-algorithms. In IEEE Congress on Evolutionary Computation (CEC), pages 71–78, 2007.
3. I. Rechenberg. Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Fromman-Holzboog, 1971.
4. H. Schwefel. Evolution and Optimum Seeking. Wiley, New York, 1995.
5. I. Loshchilov. CMA-ES with restarts for solving CEC 2013 benchmark problems. In IEEE Congress on Evolutionary Computation (CEC), pages 369–376, 2013.
6. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In International Conference on Knowledge Discovery and Data Mining (KDD), pages 226–231. AAAI Press, 1996.
7. J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In D. B. Fogel, editor, IEEE Congress on Evolutionary Computation (CEC), pages 579–584, Orlando, Florida, 1994. IEEE Press
8. V. N. Vapnik. The Nature of Statistical Learning Theory. Springer, New York, 1995.
9. X. Li, S. Zeng, S. Qin, and K. Liu. Constrained optimization problem solved by dynamic constrained NSGA-III multiobjective optimizational techniques. In IEEE Congress on Evolutionary Computation (CEC), pages 2923–2928, 2015.
10. O. K ramer and P. Koch. Rake selection: A novel evolutionary multi-objective optimization algorithm. In Advances in Artificial Intelligence (KI), pages 177–184, 2009.
11. N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective Selection based on Dominated Hypervolume. European Journal of Operational Research, 181(3):1653–1669, 2007.
12. O. Kramer. Local fitness meta-models with nearest neighbor regression. In Applications of Evolutionary Computation (EvoStar), pages 3–10, 2016.
13. I. T. Jolliffe. Principal component analysis. Springer Series in Statistics. Springer, New York u.a., 1986.

**Κώδικες**

function [Network,networkPath] = generateNetwork(Input,Output,Architecture,varargin)

## Add parameters

p = inputParser;

p.addParameter('dirDB','C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Networks');

p.parse(varargin{:})

dirDB = p.Results.dirDB;

## Generate network

trainFunction = Architecture.trainFunction;

hiddenLayers = Architecture.hiddenLayers;

Net = feedforwardnet(hiddenLayers,trainFunction);

Net.divideParam.trainRatio = Architecture.devideParameters(1);

Net.divideParam.valRatio = Architecture.devideParameters(2);

Net.divideParam.testRatio = Architecture.devideParameters(3);

Net.performFcn = Architecture.performFcn;

Net.trainParam.lr = Architecture.learningRate;

% Net.trainParam.lr\_inc = Architecture.learningRateIncreament;

% Net.trainParam.lr\_dec = Architecture.learningRateDecreament;

Net.trainParam.epochs = Architecture.epochs;

## Train the network

vectorInput = Input.Vector;

vectorOutput = Output.Vector;

[trainedNet,trainRecord] = train(Net,vectorInput,vectorOutput);

## Assign values to network

Save data

Network.Architecture = Architecture;

Network.Input = Input;

Network.Output = Output;

% Save initial network

Network.Net = Net;

% Save trained network

Network.TrainedNet = trainedNet;

Network.TrainedRecord = trainRecord;

Network.verificationPerformance = trainRecord.best\_vperf;

Network.bestEpoch = trainRecord.best\_epoch;

% Generate network file name

Network.FileName = ['TF\_',trainFunction,'\_PF\_',Architecture.performFcn,'\_',replace(num2str(Architecture.devideParameters\*100),' ','')];

for iHiddenLayer = 1:length(hiddenLayers)

Network.FileName = [Network.FileName,'\_',num2str(hiddenLayers(iHiddenLayer))];

end

% Generate path

networkPath = dirDB;

Network.networkPath = networkPath;

end

clear

dirPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 1'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 2'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 3'};

inputDataPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed1.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed2.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed3.mat'};

dataUsedPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\J1.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\J2.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\J3.mat'};

outputDataPath = "C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\OutputDataNorm.mat";

inputNames = { 'Layer height [mm]'

'Wall thickness [mm]'

'Infill density [%]'

'Infill patern'

'Nozzle temperature [C]'

'Bed temperature [C]'

'Print speed [mm/sec]'

'Material'

'Fan speed [%]'};

outputNames = { 'Roughness [μm]'

'Tension strength [Mpa]'

'Elongation [%]'};

DevideParameters = {[0.7,0.15,0.15]

[0.75,0.15,0.1]

[0.8,0.1,0.1]};

for iDevide = 1:length(DevideParameters)

for iPath = 1:length(inputDataPaths)

dirDB = [dirPaths{iPath},'\Networks'];

dirPerf = [dirPaths{iPath},'\Performance'];

load(outputDataPath)

load(inputDataPaths{iPath})

load(dataUsedPaths{iPath})

Input.Names = inputNames(J)';

Input.Vector = InputDataNormRed';

Output.Names = outputNames{iPath};

Output.Vector = OutputDataNorm(:,iPath)';

Architecture.trainFunction = 'traingdx'; % https://www.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html

Architecture.devideParameters = DevideParameters{iDevide}; % You can change this

Architecture.performFcn = 'mse';

Architecture.learningRate = 1e-3;

% Architecture.learningRateIncreament = 1;

% Architecture.learningRateDecreament = 1;

Architecture.epochs = 15000;

layersVector = 2:4:128;

verificationPerformance = zeros(length(layersVector),length(layersVector));

bestEpoch = zeros(length(layersVector),length(layersVector));

for iFirstLayer = 1:length(layersVector)

fristLayer = layersVector(iFirstLayer);

for iSecondLayer = 1:length(layersVector)

secondLayer = layersVector(iSecondLayer);

Architecture.hiddenLayers = [fristLayer,secondLayer];

Network = generateNetwork(Input,Output,Architecture,'dirDB',dirDB);

mkdir(dirDB)

save([dirDB,'\',Network.FileName],'Network')

% For each network save minimum mean squared error

verificationPerformance(iFirstLayer,iSecondLayer) = Network.verificationPerformance;

bestEpoch(iFirstLayer,iSecondLayer) = Network.bestEpoch;

end

end

[optimumFirstIdx,optimumSecondIdx] = find(min(min(verificationPerformance))==verificationPerformance);

optimumLayerFirst = layersVector(optimumFirstIdx);

optimumLayerSecond = layersVector(optimumSecondIdx);

Performance.Verification = verificationPerformance;

Performance.layersVector = layersVector;

Performance.optimumLayerFirst = optimumLayerFirst;

Performance.optimumLayerSecond = optimumLayerSecond;

Performance.bestEpoch = bestEpoch;

Performance.trainFunction = Architecture.trainFunction;

Performance.performFcn = Architecture.performFcn;

Performance.devideParameters = Architecture.devideParameters;

mkdir(dirPerf)

save([dirPerf,'\','TF\_',Architecture.trainFunction,'\_PF\_',Architecture.performFcn,'\_',replace(num2str(Architecture.devideParameters\*100),' ','')],'Performance')

end

end

clear

dirPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Problem 1'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Problem 2'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Problem 3'};

inputDataPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNorm.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNorm.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNorm.mat'};

outputDataPath = "C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\OutputDataNorm.mat";

inputNames = { 'Layer height [mm]'

'Wall thickness [mm]'

'Infill density [%]'

'Infill patern'

'Nozzle temperature [C]'

'Bed temperature [C]'

'Print speed [mm/sec]'

'Material'

'Fan speed [%]'};

outputNames = { 'Roughness [μm]'

'Tension strength [Mpa]'

'Elongation [%]'};

DevideParameters = {[0.7,0.15,0.15]

[0.75,0.15,0.1]

[0.8,0.1,0.1]};

for iDevide = 1:length(DevideParameters)

for iPath = 1:length(inputDataPaths)

dirDB = [dirPaths{iPath},'\Networks'];

dirPerf = [dirPaths{iPath},'\Performance'];

load(outputDataPath)

load(inputDataPaths{iPath})

Input.Names = inputNames';

Input.Vector = InputDataNorm';

Output.Names = outputNames{iPath};

Output.Vector = OutputDataNorm(:,iPath)';

Architecture.trainFunction = 'traingdx'; % https://www.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html

Architecture.devideParameters = DevideParameters{iDevide}; % You can change this

Architecture.performFcn = 'mse';

Architecture.learningRate = 1e-3;

% Architecture.learningRateIncreament = 1;

% Architecture.learningRateDecreament = 1;

Architecture.epochs = 15000;

layersVector = 2:4:128;

verificationPerformance = zeros(length(layersVector),length(layersVector));

bestEpoch = zeros(length(layersVector),length(layersVector));

for iFirstLayer = 1:length(layersVector)

fristLayer = layersVector(iFirstLayer);

for iSecondLayer = 1:length(layersVector)

secondLayer = layersVector(iSecondLayer);

Architecture.hiddenLayers = [fristLayer,secondLayer];

Network = generateNetwork(Input,Output,Architecture,'dirDB',dirDB);

mkdir(dirDB)

save([dirDB,'\',Network.FileName],'Network')

% For each network save minimum mean squared error

verificationPerformance(iFirstLayer,iSecondLayer) = Network.verificationPerformance;

bestEpoch(iFirstLayer,iSecondLayer) = Network.bestEpoch;

end

end

[optimumFirstIdx,optimumSecondIdx] = find(min(min(verificationPerformance))==verificationPerformance);

optimumLayerFirst = layersVector(optimumFirstIdx);

optimumLayerSecond = layersVector(optimumSecondIdx);

Performance.Verification = verificationPerformance;

Performance.layersVector = layersVector;

Performance.optimumLayerFirst = optimumLayerFirst;

Performance.optimumLayerSecond = optimumLayerSecond;

Performance.bestEpoch = bestEpoch;

Performance.trainFunction = Architecture.trainFunction;

Performance.performFcn = Architecture.performFcn;

Performance.devideParameters = Architecture.devideParameters;

mkdir(dirPerf)

save([dirPerf,'\','TF\_',Architecture.trainFunction,'\_PF\_',Architecture.performFcn,'\_',replace(num2str(Architecture.devideParameters\*100),' ','')],'Performance')

end

end

performanceDirs = [

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 3\Performance\TF\_traingdx\_PF\_mse\_801010.mat"

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 3\Performance\TF\_traingdx\_PF\_mse\_751510.mat"

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 3\Performance\TF\_traingdx\_PF\_mse\_701515.mat"

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 2\Performance\TF\_traingdx\_PF\_mse\_801010.mat"

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 2\Performance\TF\_traingdx\_PF\_mse\_751510.mat"

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 2\Performance\TF\_traingdx\_PF\_mse\_701515.mat"

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 1\Performance\TF\_traingdx\_PF\_mse\_801010.mat"

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 1\Performance\TF\_traingdx\_PF\_mse\_751510.mat"

"C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 1\Performance\TF\_traingdx\_PF\_mse\_701515.mat"

];

problems = {

'Elongation'

'Elongation'

'Elongation'

'Tension strength'

'Tension strength'

'Tension strength'

'Roughness'

'Roughness'

'Roughness'

};

bestResultsMatrix = cell(3,3);

bestResult = zeros(3,3);

bestEpocheMatrix = zeros(3,3);

for iPerformance = 1:length(performanceDirs)

load(performanceDirs(iPerformance))

Performance.OptimumPerformance = Performance.Verification(Performance.layersVector==Performance.optimumLayerFirst,Performance.layersVector==Performance.optimumLayerSecond);

Performance.solustionBestEpoche = Performance.bestEpoch(Performance.layersVector==Performance.optimumLayerFirst,Performance.layersVector==Performance.optimumLayerSecond);

fontSize = 15;

figure

contourf(Performance.layersVector,Performance.layersVector,Performance.Verification',0:0.1:1,'ShowText','on')

c = colorbar;

c.Limits = [0,1];

c.Title.String = [Performance.performFcn,' [%]'];

c.Title.FontSize = 15;

titleMsg = sprintf([problems{iPerformance},' problem\n','Mean squared error validation performance vs Layer architecture\nTrain function = ',Performance.trainFunction,'\nData division: ',replace(num2str(Performance.devideParameters\*100),' ','-')]);

title(titleMsg,'fontsize',fontSize)

xlabel('Number of neurons at 1st layer','fontsize',fontSize)

ylabel('Number of neurons at 2nd layer','fontsize',fontSize)

hold on

scatter(Performance.optimumLayerFirst,Performance.optimumLayerSecond,100,'filled','MarkerEdgeColor',[0.7,0,0],'MarkerFaceColor',[0.7,0,0])

txtMessage = sprintf('Optimum first layer = %d\nOptimum second layer = %d\n Error = %.4f [%%]',Performance.optimumLayerFirst,Performance.optimumLayerSecond,Performance.OptimumPerformance);

txt = text(Performance.optimumLayerFirst,Performance.optimumLayerSecond+2,txtMessage,'fontsize',15,'color',[0.7,0,0]);

txt.HorizontalAlignment = 'center';

txt.VerticalAlignment = 'bottom';

txt.FontWeight = 'bold';

bestResultsMatrix{ceil(iPerformance/3),mod(iPerformance+2,3)+1} = [Performance.optimumLayerFirst,Performance.optimumLayerSecond]; % x data split, y problem

bestResult(ceil(iPerformance/3),mod(iPerformance+2,3)+1) = Performance.OptimumPerformance;

bestEpocheMatrix(ceil(iPerformance/3),mod(iPerformance+2,3)+1) = Performance.solustionBestEpoche;

end

## Find optimum network for each problem

OptimumNetworksArchitecture = {bestResultsMatrix{(bestResult'==min(bestResult'))'}};

OptimumNetworksDataDivHelp = ones(3,1)\*[1:3];

OptimumNetworksDataDiv = OptimumNetworksDataDivHelp((bestResult'==min(bestResult'))');

pathProblem = {

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 3'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 2'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 1'

};

dataDivisionName = {

'801010'

'751510'

'701515'

};

for iProblem = 1:3

networkFileName = [pathProblem{iProblem},'\Networks\','TF\_traingdx\_PF\_mse\_',dataDivisionName{OptimumNetworksDataDiv(iProblem)},'\_',num2str(OptimumNetworksArchitecture{iProblem}(1)),'\_',num2str(OptimumNetworksArchitecture{iProblem}(2)),'.mat'];

load(networkFileName)

save([pathProblem{iProblem},'\','optimumNetwork.mat'],'Network')

end

FONTSIZE = 15;

pathProblem = {

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 1'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 2'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 3'

};

inputDataPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed1.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed2.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed3.mat'};

outputNames = { 'Roughness [μm]'

'Tension strength [Mpa]'

'Elongation [%]'};

outputDataPath = "C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\OutputDataNorm.mat";

for iProblem = 1:length(pathProblem)

load(outputDataPath)

load(inputDataPaths{iProblem})

InputVector = InputDataNormRed';

OutputVectorReal = OutputDataNorm(:,iProblem)';

load([pathProblem{iProblem},'\optimumNetwork.mat']);

OutputVectorNeural = sim(Network.TrainedNet,InputVector);

figure

barData = [OutputVectorReal;OutputVectorNeural]';

bar(barData)

titleMsg = sprintf(['Optimum network ',replace(outputNames{iProblem},'%','%%'),'\nExperimental output data vs Predicted output data for the given data samples','\nNetwork: ',replace(Network.FileName,'\_','-')]);

title(titleMsg,'fontsize',FONTSIZE)

xlabel('Data sample','fontsize',FONTSIZE)

ylabel('Min square error','fontsize',FONTSIZE)

legend('Experimental data','Predicted data')

end

clear all

clc

## συσχέτιση μέσο απλής γραμμικής παλινδρόμησης %%

Data = xlsread("Data.xlsx");

xlData = readtable("Data.xlsx");

sz = size(xlData);

for i = 1:sz(1)

if string(xlData{i,4}) == 'grid'

Data(i,4) = 1;

else

Data(i,4) = 0;

end

end

sz = size(xlData);

for i = 1:sz(1)

if string(xlData{i,8}) == 'pla'

Data(i,8) = 1;

else

Data(i,8) = 0;

end

end

sz = size(Data);

szIn = 9;

szOut = sz(2)-szIn;

for i=1:sz(2);

MaxMin(i) = (max(Data(:,i))-min(Data(:,i)));

Min(i) = min(Data(:,i));

save('MaxMin.mat','MaxMin')

save('Min.mat','Min')

DataNorm(:,i) = (Data(:,i)-min(Data(:,i)))/(max(Data(:,i))-min(Data(:,i)));

end

InputDataNorm = DataNorm(1:sz(1),1:szIn);

OutputDataNorm = DataNorm(1:sz(1),szIn+1:sz(2));

save('InputDataNorm.mat','InputDataNorm')

save('OutputDataNorm.mat','OutputDataNorm')

outNum = 3 ;

for i=1:outNum

A = [ones(sz(1),1),InputDataNorm];

Y = OutputDataNorm(:,i);

b = lsqr(A,Y);

bCo = b(2:szIn+1);

bCoNorm(:,i) = bCo/max(abs(bCo));

end

## Reduction %%

i=1;j=1;

for i = 1:(sz(2)-szIn)

InputDataNormRed = [];

J = [];

for j = 1:length(bCoNorm(:,i))

if abs(bCoNorm(j,i)) >= 0.1

InputDataNormRed = [InputDataNormRed,InputDataNorm(:,j)];

J = [J,j] ;

else

end

save(['InputDataNormRed',num2str(i),'.mat'],'InputDataNormRed')

save(['J',num2str(i),'.mat'],'J')

end

end

function [InputData] = deNormalize(OutMat,InMat,MaxMin,Min,Jindex)

load(OutMat) %OutputDataNorm.mat

load(InMat) %InputDataNorm.mat

load(MaxMin) %MaxMin.mat

load(Min) %Min.mat

load(Jindex) %J1.mat

szOut = 3;

szIn = 9;

for i = 1:szOut

OutputData(:,i) = MaxMin(szIn+i)\*OutputDataNorm(:,i) + Min(szIn+i);

end

for i = 1:length(J)

InputData(:,i) = MaxMin(J(i))\*InputDataNorm(:,J(i)) + Min(J(i));

end

end

clear

dirPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 1'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 2'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\DataBase\Reduced Problem 3'};

inputDataPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed1.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed2.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\InputDataNormRed3.mat'};

dataUsedPaths = { 'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\J1.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\J2.mat'

'C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\J3.mat'};

outputDataPath = "C:\Users\Antonis Kantounias\Documents\ergasies\inteligentMachiningSystems\excersise1\InputData\OutputDataNorm.mat";

inputNames = { 'Layer height [mm]'

'Wall thickness [mm]'

'Infill density [%]'

'Infill patern'

'Nozzle temperature [C]'

'Bed temperature [C]'

'Print speed [mm/sec]'

'Material'

'Fan speed [%]'};

outputNames = { 'Roughness [μm]'

'Tension strength [Mpa]'

'Elongation [%]'};

DevideParameters = {[0.7,0.15,0.15]

[0.75,0.15,0.1]

[0.8,0.1,0.1]};

for iDevide = 1:length(DevideParameters)

for iPath = 1:length(inputDataPaths)

dirDB = [dirPaths{iPath},'\Networks'];

dirPerf = [dirPaths{iPath},'\Performance'];

load(outputDataPath)

load(inputDataPaths{iPath})

load(dataUsedPaths{iPath})

Input.Names = inputNames(J)';

Input.Vector = InputDataNormRed';

Output.Names = outputNames{iPath};

Output.Vector = OutputDataNorm(:,iPath)';

Architecture.trainFunction = 'traingdx'; % https://www.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html

Architecture.devideParameters = DevideParameters{iDevide}; % You can change this

Architecture.performFcn = 'mse';

Architecture.learningRate = 1e-3;

% Architecture.learningRateIncreament = 1;

% Architecture.learningRateDecreament = 1;

Architecture.epochs = 15000;

layersVector = 2:4:128;

verificationPerformance = zeros(length(layersVector),length(layersVector));

bestEpoch = zeros(length(layersVector),length(layersVector));

for iFirstLayer = 1:length(layersVector)

fristLayer = layersVector(iFirstLayer);

for iSecondLayer = 1:length(layersVector)

secondLayer = layersVector(iSecondLayer);

Architecture.hiddenLayers = [fristLayer,secondLayer];

Network = generateNetwork(Input,Output,Architecture,'dirDB',dirDB);

mkdir(dirDB)

save([dirDB,'\',Network.FileName],'Network')

% For each network save minimum mean squared error

verificationPerformance(iFirstLayer,iSecondLayer) = Network.verificationPerformance;

bestEpoch(iFirstLayer,iSecondLayer) = Network.bestEpoch;

end

end

[optimumFirstIdx,optimumSecondIdx] = find(min(min(verificationPerformance))==verificationPerformance);

optimumLayerFirst = layersVector(optimumFirstIdx);

optimumLayerSecond = layersVector(optimumSecondIdx);

Performance.Verification = verificationPerformance;

Performance.layersVector = layersVector;

Performance.optimumLayerFirst = optimumLayerFirst;

Performance.optimumLayerSecond = optimumLayerSecond;

Performance.bestEpoch = bestEpoch;

Performance.trainFunction = Architecture.trainFunction;

Performance.performFcn = Architecture.performFcn;

Performance.devideParameters = Architecture.devideParameters;

mkdir(dirPerf)

save([dirPerf,'\','TF\_',Architecture.trainFunction,'\_PF\_',Architecture.performFcn,'\_',replace(num2str(Architecture.devideParameters\*100),' ','')],'Performance')

end

end

clear all

clc

## συσχέτιση μέσο απλής γραμμικής παλινδρόμησης %%

Data = xlsread("Data.xlsx");

xlData = readtable("Data.xlsx");

sz = size(xlData);

for i = 1:sz(1)

if string(xlData{i,4}) == 'grid'

Data(i,4) = 1;

else

Data(i,4) = 0;

end

end

sz = size(xlData);

for i = 1:sz(1)

if string(xlData{i,8}) == 'pla'

Data(i,8) = 1;

else

Data(i,8) = 0;

end

end

sz = size(Data);

szIn = 9;

szOut = sz(2)-szIn;

for i=1:sz(2);

MaxMin(i) = (max(Data(:,i))-min(Data(:,i)));

Min(i) = min(Data(:,i));

save('MaxMin.mat','MaxMin')

save('Min.mat','Min')

DataNorm(:,i) = (Data(:,i)-min(Data(:,i)))/(max(Data(:,i))-min(Data(:,i)));

end

InputDataNorm = DataNorm(1:sz(1),1:szIn);

OutputDataNorm = DataNorm(1:sz(1),szIn+1:sz(2));

save('InputDataNorm.mat','InputDataNorm')

save('OutputDataNorm.mat','OutputDataNorm')

outNum = 3 ;

for i=1:outNum

A = [ones(sz(1),1),InputDataNorm];

Y = OutputDataNorm(:,i);

b = lsqr(A,Y);

bCo = b(2:szIn+1);

bCoNorm(:,i) = bCo/max(abs(bCo));

end

Warning: Column headers from the file were modified to make them valid MATLAB

identifiers before creating variable names for the table. The original column

headers are saved in the VariableDescriptions property.

Set 'VariableNamingRule' to 'preserve' to use the original column headers as

table variable names.

lsqr converged at iteration 10 to a solution with relative residual 0.19.

lsqr converged at iteration 10 to a solution with relative residual 0.28.

lsqr converged at iteration 10 to a solution with relative residual 0.28.

## Reduction %%

i=1;j=1;

for i = 1:(sz(2)-szIn)

InputDataNormRed = [];

J = [];

for j = 1:length(bCoNorm(:,i))

if abs(bCoNorm(j,i)) >= 0.1

InputDataNormRed = [InputDataNormRed,InputDataNorm(:,j)];

J = [J,j] ;

else

end

save(['InputDataNormRed',num2str(i),'.mat'],'InputDataNormRed')

save(['J',num2str(i),'.mat'],'J')

end

end