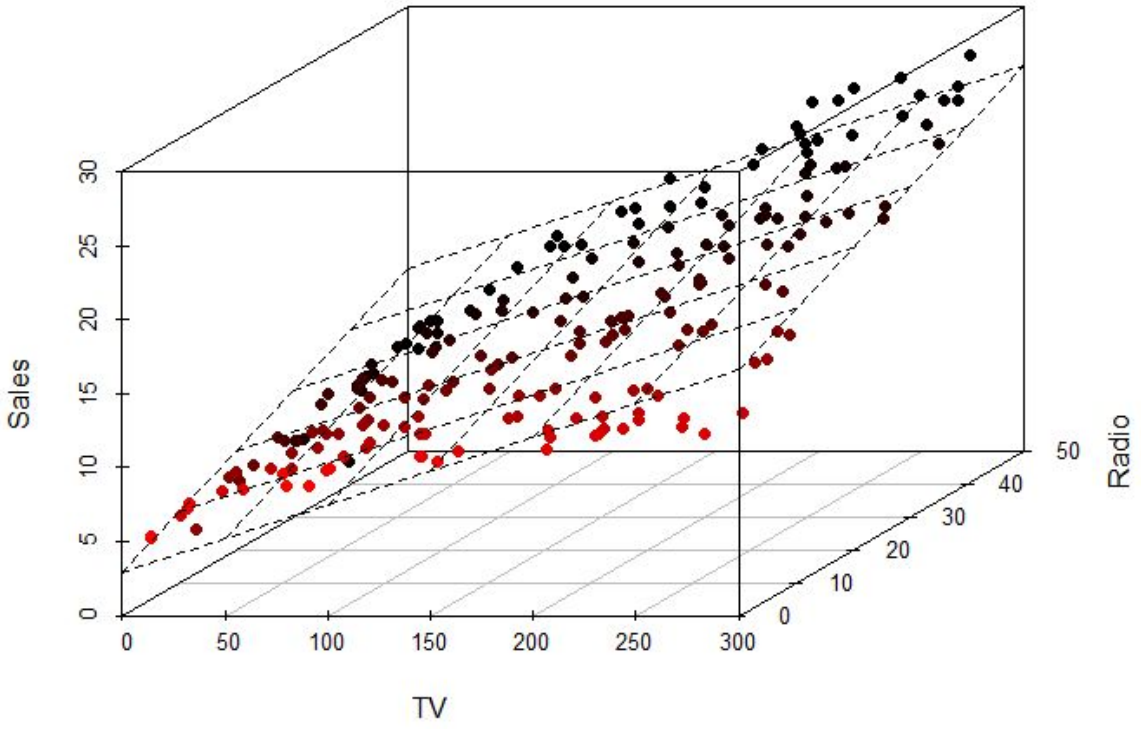
ΑΛΓΟΡΙΘΜΙΚΗ ΕΠΙΧΕΙΡΗΣΙΑΚΗ ΕΡΕΥΝΑ

2019-2020



Φίλιππος Αποστόλου
Ευστράτιος Βαμβουρέλλης

Περιεχόμενα

Εισαγωγή

Machine learning είναι μια υποκατηγορία του Artificial Intelligence. Διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικά με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις. Δεν πρέπει να ταυτοποιούμε τον όρο Machine Learning με τα Neural Networks. Τα NN είναι μια υποκατηγορία του ML, όπως βλέπουμε και στην παρακάτω εικόνα.

Εμείς θα αναλύσουμε κάποιους αλγορίθμους που ανήκουν αποκλειστικά στο ML. Είναι εξελίξεις στατιστικών αλγορίθμων. Ονομαστικά θα μιλήσουμε για το least squares, linear regression, multiple regression, gradient descent.

Παρακάτω βλέπουμε ένα πίνακα από το wikipedia που δείχνει κάποιες σημαντικές ανακαλύψεις στον τομέα του AI. Παρατηρούμε ότι οι πρώτοι αλγόριθμοι ήταν καθαρά στατιστικοί, στην συνέχεια γράφτηκαν οι πρώτοι αλγόριθμοι ML και αργότερα τα πρώτα νευρωνικά δίκτυα. Βλέπουμε οτι το Least Squares έχει ανακαλυφθεί από το 1805! Το gradient descent ανακαλύφθηκε το 1847! Αυτοί οι αλγόριθμοι, αν και παλιοί, ακόμα έχουν πρακτική χρήση σε πολλά γραμμικά προβλήματα και αποτελούν την βάση για άλλους πιο περίπλοκους αλγόριθμους.

Year	Caption	Event
1763	The Underpinnings of Bayes' Theorem	Thomas Bayes's work *An Essay towards solving a Problem in the Doctrine of Chances* is published two years after his death, having been amended and edited by a friend of Bayes, Richard Price.[8] The essay presents work which underpins Bayes theorem.
1805	Least Squares	Adrien-Marie Legendre describes the "méthode des moindres carrés", known in English as the least squares method.[9] The least squares method is used widely in data fitting.
1812	Bayes' Theorem	Pierre-Simon Laplace publishes *Théorie Analytique des Probabilités*, in which he expands upon the work of Bayes and defines what is now known as Bayes' Theorem.[10]

1951	First Neural Network Machine	Marvin Minsky and Dean Edmonds build the first neural network machine, able to learn, the SNARC.[13]
1952	Machines Playing Checkers	Arthur Samuel joins IBM's Poughkeepsie Laboratory and begins working on some of the very first machine learning programs, first creating programs that play checkers.[14]
1957	Perceptron	Frank Rosenblatt invents the perceptron while working at the Cornell Aeronautical Laboratory.[15] The invention of the perceptron generated a great deal of excitement and was widely covered in the media.[16]
1963	Machines Playing Tic-Tac-Toe	Donald Michie creates a 'machine' consisting of 304 match boxes and beads, which uses reinforcement learning to play Tic-tac-toe (also known as noughts and crosses).[17]
1967	Nearest Neighbor	The nearest neighbor algorithm was created, which is the start of basic pattern recognition. The algorithm was used to map routes.[2]
1981	Explanation Based Learning	Gerald Dejong introduces Explanation Based Learning, where a computer algorithm analyses data and creates a general rule it can follow and discard unimportant data.[2]
1982	Recurrent Neural Network	John Hopfield popularizes Hopfield networks, a type of recurrent neural network that can serve as content-addressable memory systems.[29]
1986	Backpropagation	Seppo Linnainmaa's reverse mode of automatic differentiation (first applied to neural networks by Paul Werbos) is used in experiments by David Rumelhart, Geoff Hinton and Ronald J. Williams to learn internal representations.[30]
1989	Reinforcement Learning	Christopher Watkins develops Q-learning, which greatly improves the practicality and feasibility of reinforcement learning.[31]
1995	Random Forest Algorithm	Tin Kam Ho publishes a paper describing random decision forests.[34]
1995	Support Vector Machines	Corinna Cortes and Vladimir Vapnik publish their work on support vector machines.[35][36]
1997	IBM Deep Blue Beats Kasparov	IBM's Deep Blue beats the world champion at chess.[2]

2009	ImageNet	ImageNet is created. ImageNet is a large visual database envisioned by Fei-Fei Li from Stanford University, who realized that the best machine learning algorithms wouldn't work well if the data didn't reflect the real world.[41] For many, ImageNet was the catalyst for the AI boom[42] of the 21st century.
2012	Recognizing Cats on YouTube	The Google Brain team, led by Andrew Ng and Jeff Dean, create a neural network that learns to recognize cats by watching unlabeled images taken from frames of YouTube videos.[46][47]
2014	Leap in Face Recognition	Facebook researchers publish their work on DeepFace, a system that uses neural networks that identifies faces with 97.35% accuracy. The results are an improvement of more than 27% over previous systems and rivals human performance.[48]
2016	Beating Humans in Go	Google's AlphaGo program becomes the first Computer Go program to beat an unhandicapped professional human player[51] using a combination of machine learning and tree search techniques.[52] Later improved as AlphaGo Zero and then in 2017 generalized to Chess and more two-player games with AlphaZero.

Classification vs Regression

Δύο μεγάλες κατηγορίες προβλημάτων στο AI είναι το classification και το regression.

Classification

Στη μηχανική μάθηση (Machine Learning) και στη στατιστική, η ταξινόμηση (classification) είναι το πρόβλημα του προσδιορισμού σε ποιο σύνολο κατηγοριών (υποπληθυσμών) ανήκει μια νέα παρατήρηση, με βάση ένα σετ εκπαίδευσης (training set) των δεδομένων που περιέχει τις παρατηρήσεις (ή περιπτώσεις) των οποίων η κατηγορία μέλους είναι γνωστή. Ένα παράδειγμα θα ήταν η ταξινόμηση ηλεκτρονικού ταχυδρομείου σε "Σπαμ" ή "μη-σπαμ" τάξεις ή μια διάγνωση για έναν δεδομένο ασθενή με βάση τα παρατηρούμενα χαρακτηριστικά του ασθενούς (το φύλο, η πίεση του αίματος, η παρουσία ή απουσία ορισμένων συμπτωμάτων, κτλ.). Η ταξινόμηση είναι ένα παράδειγμα αναγνώρισης μοτίβων (pattern recognition).

Regression

Η παλινδρόμηση(Regression) είναι μια ευρέως χρησιμοποιούμενη στατιστική τεχνική μοντελοποίησης για την έρευνα της συσχέτισης μεταξύ μιας εξαρτημένης μεταβλητής και μιας ή περισσότερων ανεξάρτητων μεταβλητών. Χρησιμοποιείται με σκοπό την εκχώρηση δεδομένων σε μία πραγματική μεταβλητή πρόβλεψης, όπως ισχύει και στην περίπτωση της κατηγοριοποίησης όταν είναι διακριτή, αλλιώς καλείται παλινδρόμηση αν η μεταβλητή είναι συνεχής. Η παλινδρόμηση προϋποθέτει ότι τα σχετικά δεδομένα ταιριάζουν με μερικά γνωστά είδη συνάρτησης και μετά καθορίζει την καλύτερη συνάρτηση αυτού του είδους που μοντελοποιεί τα δεδομένα που έχουν δοθεί. Αποτέλεσμα της παλινδρόμησης όταν χρησιμοποιείται ως τεχνική εξόρυξης δεδομένων (data mining), αποτελεί ένα μοντέλο που χρησιμοποιείται αργότερα για να προβλέψει τις τιμές της κατηγορίας για τα νέα δεδομένα. Τέτοια παραδείγματα εφαρμογής της παλινδρόμησης αποτελεί η πρόβλεψη της ζήτησης για ένα νέο προϊόν ή υπηρεσία συναρτήσει των δαπανών διαφήμισης ή ο υπολογισμός της ταχύτητας του ανέμου σε σχέση με την θερμοκρασία, την υγρασία και την ατμοσφαιρική πίεση του περιβάλλοντος.

Classification vs Regression

Με λίγα λόγια η διαφορά είναι η εξής:

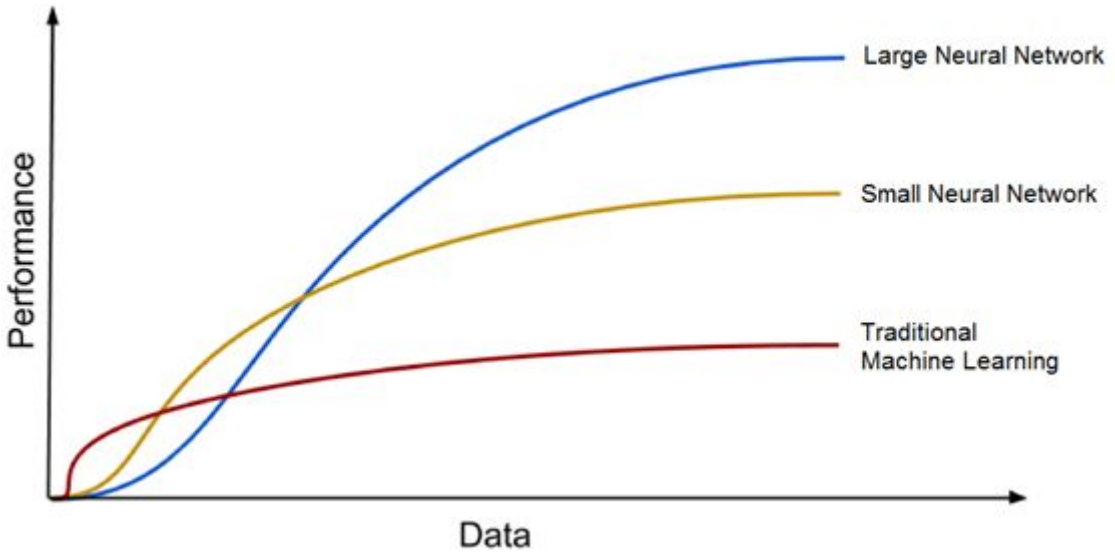
- Στα προβλήματα classification προσδίδουμε μια ταμπέλα σε κάθε αντικείμενο πχ αυτοκίνητο ή πεζός.
- Στα προβλήματα regression προσδίδουμε μια συνεχόμενη τιμή σε κάθε αντικείμενο, συνήθως μια πιθανότητα, πχ είναι 80% αυτοκίνητο και 20% πεζός.

Linear Classifiers

Οι μέθοδοι linear regression έχουν σκοπό την πρόβλεψη της τιμής μιας συνεχής εξαρτημένης μεταβλητής. Υπολογίζουν αυτή την τιμή έχοντας υπόψη ένα γραμμικό συνδυασμό των δεδομένων εκπαίδευσης (training data) . Τα χαρακτηριστικά των δεδομένων ονομάζονται feature vectors. Τέτοιες μέθοδοι είναι χρήσιμες σε πραγματικές εφαρμογές πχ document classification, sentiment analysis και γενικά προβλήματα πολλών μεταβλητών, φτάνοντας σε ακρίβεια τις, πιο καινούργιες ,μη γραμμικές μεθόδους έχοντας μικρότερο χρόνο εκπαίδευσης.

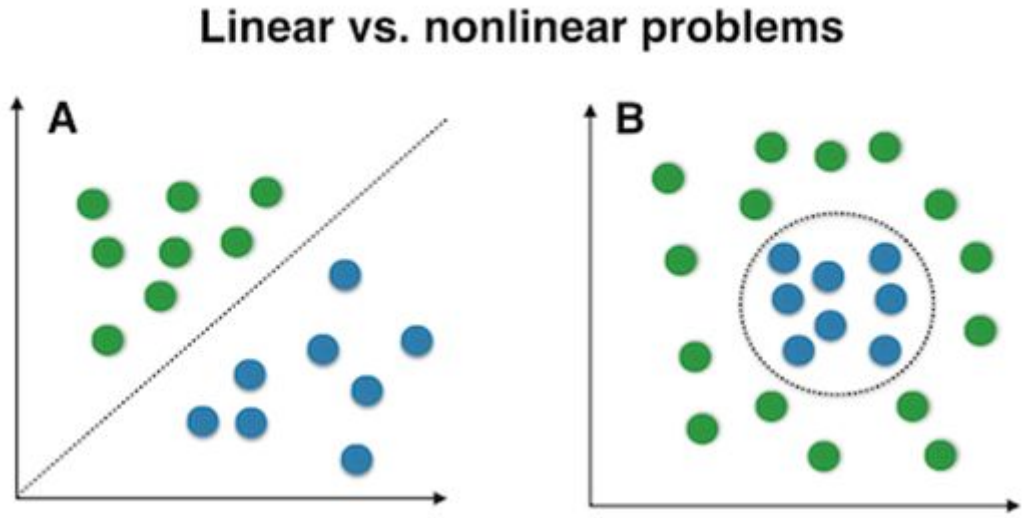
Θα μιλήσουμε για κάποιους βασικούς linear classifiers και μεθόδους linear regression. Γενικά αυτοί οι αλγόριθμοι είναι πιο παλιοί και λιγότερο αποδοτικοί από τα Neural Networks.

Αν δεν έχουμε, όμως, σχετικά μεγάλο αριθμό δεδομένων, αυτοί οι αλγόριθμοι ξεπερνούν τα NN όπως βλέπουμε στο παρακάτω διάγραμμα.



Γιατί να χρησιμοποιήσω linear classifier ?!

Γενικά οι linear classifiers είναι πολύ καλοί για δεδομένα που έχουν γραμμική μορφή, μπορούν να διαχωριστούν από μια ευθεία γραμμή.



Ένα μεγάλο πλεονέκτημα τους ενάντια στα μη γραμμικά μοντέλα (πχ decision trees, knn) είναι ότι είναι πολύ απλοί, έχουν χαμηλό variance! Αυτό σημαίνει ότι είναι σπάνιο και να

κάνουν overfit τα δεδομένα μας. Ακόμα και έτσι, υπάρχουν τεχνικές για να μειώσουμε το variance σε linear classifiers (regularization πχ ridge, lasso, elastic net), τις οποίες θα δούμε παρακάτω.Προφανώς και υπάρχουν αντίστοιχες τεχνικές και σε non-linear classifiers (πχ pruning).

Bias – Variance trade off

Στις στατιστική και στην μηχανική μάθηση, η αντιστάθμιση bias-variance είναι η ιδιότητα ενός συνόλου προγνωστικών μοντέλων, όπου τα μοντέλα με χαμηλότερο bias στην εκτίμηση των παραμέτρων έχουν υψηλότερο variance των εκτιμώμενων παραμετρων μεταξύ δειγμάτων και αντίστροφα. Το δίλημμα bias-variance είναι η σύγκρουση στην προσπάθεια ταυτόχρονης ελαχιστοποίησης αυτών των δύο πηγών σφάλματος που εμποδίζουν τους αλγορίθμους εποπτευόμενης μάθησης (supervised learning) να γενικεύουν πέρα από το σετ εκπαιδεύσεών τους:

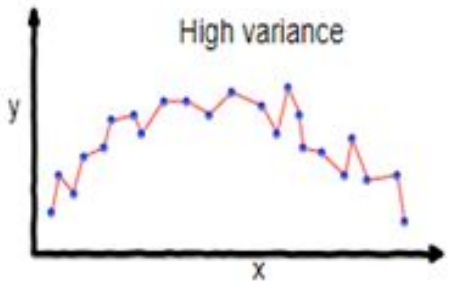
Το σφάλμα πόλωσης (bias error) είναι ένα σφάλμα από εσφαλμένες υποθέσεις στον αλγόριθμο μάθησης. Το υψηλό bias μπορεί να προκαλέσει έναν αλγόριθμο να χάσει τις σχέσεις μεταξύ των χαρακτηριστικών και της προβλεπόμενης εξόδου (underfitting).

Το variance είναι ένα σφάλμα από την ευαισθησία στις μικρές διακυμάνσεις στο σύνολο εκπαίδευσης. Η μεγάλο variance μπορεί να προκαλέσει έναν αλγόριθμο για να μοντελοποιήσει τον τυχαίο θόρυβο στα δεδομένα εκπαίδευσης, παρά τις προβλεπόμενες εξόδους (overfitting).

Το Bias – Variance trade off είναι ένας τρόπος ανάλυσης του αναμενόμενου σφάλματος γενίκευσης του αλγορίθμου μάθησης σε σχέση με ένα συγκεκριμένο πρόβλημα ως άθροισμα τριών όρων, της απόκλισης (bias), της διακύμανσης (variance) και μιας ποσότητας που ονομάζεται μη αναγώγιμο σφάλμα (irreducible error), που προκύπτει από τον θόρυβο στο ίδιο το πρόβλημα.

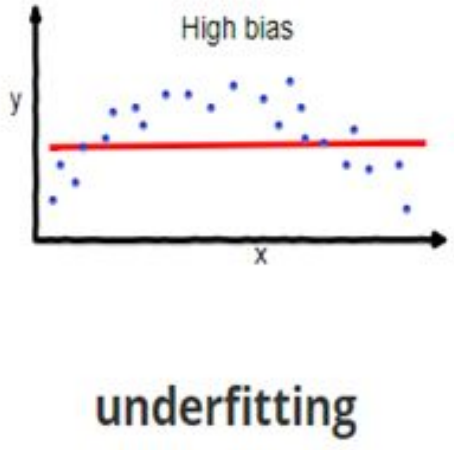
Αυτό το εμπόριο ισχύει για όλες τις μορφές εποπτευόμενης μάθησης (supervised learning): classification,regression και structured output learning.

Με λίγα λόγια:



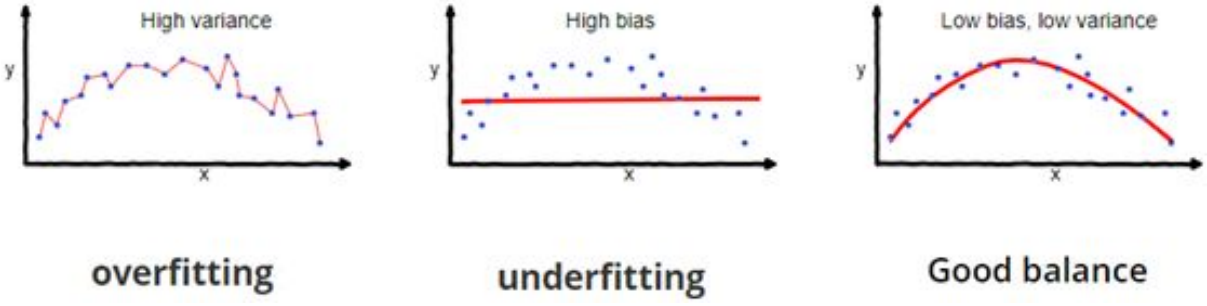
Variance είναι το φαινόμενο που συμβαίνει όταν μια μέθοδος αναπαριστά “τέλεια” τα δεδομένα μας, αλλά δεν προβλέπει καλά

οποιοδήποτε άλλο σετ, γνωστό και ως overfitting. πχ



Bias είναι το φαινόμενο που συμβαίνει όταν μια μέθοδος δεν μπορεί να “εκφράσει” σωστά τα δεδομένα μας, γνωστό και ως underfitting. πχ

Bias-Variance trade off είναι ένας όρος που χρησιμοποιείται πολύ συχνά στο machine learning. Αναφέρεται στην ισορροπία μεταξύ bias και variance που πρέπει να έχει ένα μοντέλο για να θεωρηθεί ακριβές. πχ



Cross Validation

Η διασταυρούμενη επικύρωση (cross validation), μερικές φορές ονομάζεται rotation estimation είναι σαν ένα λίγο πιο περίπλοκο split training. Το cross validation είναι μια ομάδα τεχνικών επικύρωσης μοντέλων για την αξιολόγηση του τρόπου γενίκευσης των αποτελεσμάτων μιας στατιστικής ανάλυσης σε ένα ανεξάρτητο σύνολο δεδομένων. Χρησιμοποιείται κυρίως σε αλγορίθμους όπου ο στόχος είναι πρόβλεψη και κάποιος θέλει να υπολογίσει πόσο ακριβές θα είναι ένα μοντέλο πρόβλεψης στην πράξη. Σε ένα πρόβλημα πρόβλεψης, ένα μοντέλο συνήθως λαμβάνει ένα σύνολο γνωστών δεδομένων (training data) για τα οποία εκτελείται η εκπαίδευση και ένα σύνολο άγνωστων δεδομένων (testing data) με τα οποία δοκιμάζεται το μοντέλο. Ο στόχος του cross validation είναι να ελέγξει την ικανότητα του μοντέλου να προβλέψει νέα δεδομένα που δεν χρησιμοποιήθηκαν για την

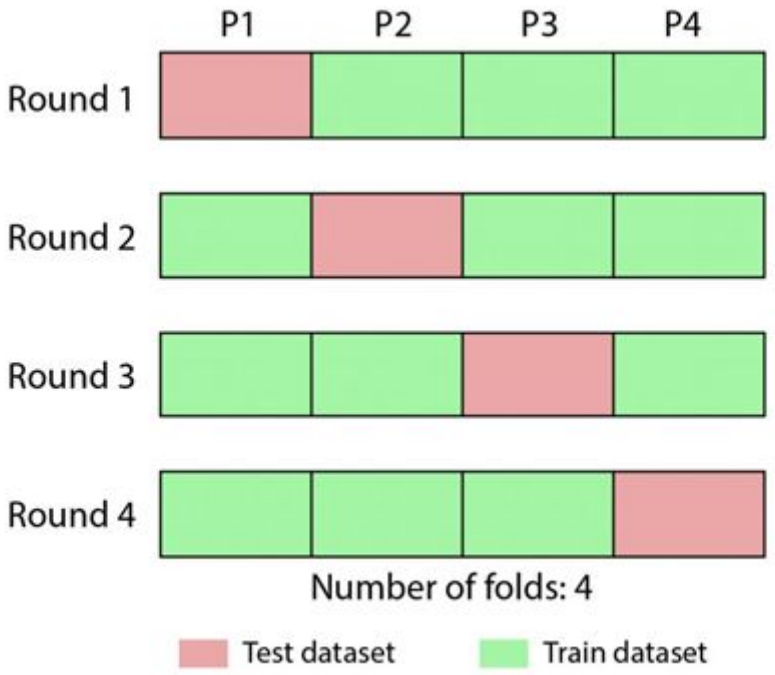
εκπαίδευσή του, προκειμένου να επισημάνει προβλήματα όπως overfitting ή selection bias και να δώσει μια εικόνα για το πώς το μοντέλο θα γενικευτεί σε ένα ανεξάρτητο σύνολο δεδομένων (δηλαδή ένα άγνωστο σύνολο δεδομένων, για παράδειγμα από ένα πραγματικό πρόβλημα).

Ένας κύκλος cross validation περιλαμβάνει τη διαίρεση δείγματος δεδομένων σε συμπληρωματικά υποσύνολα, την εκτέλεση της ανάλυσης σε ένα υποσύνολο (training data) και την επικύρωση της ανάλυσης στο άλλο υποσύνολο (testing data). Για να μειωθεί η διακύμανση, στις περισσότερες μεθόδους εκτελούνται πολλαπλοί cross validation χρησιμοποιώντας διαφορετικά διαμερίσματα και τα αποτελέσματα επικύρωσης συνδυάζονται (π.χ. κατά μέσο όρο) για να δώσουν μια εκτίμηση της προβλεπόμενης απόδοσης του μοντέλου.
Επιπλέων, το cross validation μας βοηθάει να βρούμε τις κατάλληλες τιμές για τις παραμέτρους του αλγορίθμου που χρησιμοποιούμε. Μπορούμε για παράδειγμα, απλά, να δοκιμάσουμε κάποιες τιμές που πιστεύουμε ότι είναι καλές ή να χρησιμοποιήσουμε κάποιον πιο περίπλοκο αλγόριθμο σύγκλισης για τις παραμέτρους του αλγορίθμου μας και τέλος, να αξιολογήσουμε το μοντέλο μας με cross validation.

Τέλος το cross validation μας βοηθάει να διαλέξουμε τον καλύτερο αλγόριθμο για το πρόβλημά μας, καθώς δοκιμάζοντας διαφορετικούς αλγορίθμους μπορούμε να συγκρίνουμε την αποτελεσματικότητά τους.

πχ 4-fold cross validation

Όπως βλέπουμε στην παρακάτω εικόνα έχουμε χωρίσει τα δεδομένα μας σε 4 ισάριθμες τυχαίες ομάδες P1,P2,P3,P4. Θα εκτελέσουμε τον αλγόριθμο που θέλουμε να τσεκάρουμε 4 φορές-γύρους. Κάθε φορά θα τον εκπαιδεύουμε και τεστάρουμε με άλλα δεδομένα. Την πρώτη φορά θα τον εκπαιδεύσουμε με τις πράσινες ομάδες P2,P3,P4 (training data) και θα τον τεστάρουμε με την κόκκινη P1. Αντίστοιχα και για τους άλλους 3 γύρους.



Linear Regression Analysis

Ενίοτε μας ζητείτε ,έχοντας δεδομένα για ένα γεγονός, η μελλοντική του συμπεριφορά. Επίσης η αξιολόγηση των προηγούμενων δεδομένων ως προς το ποσό επηρέασαν το γεγονός με πιο απλά λόγια αν τα δεδομένα που έχουμε δικαιολογούν η συμπεριφορά που ακολουθεί το γεγονός.

Μπορούμε να φτιάξουμε ένα μαθηματικό μοντέλο βασισμένο πάνω σε αυτό το γεγονός χωρίζοντας τα δεδομένα μας ως σημεία στον χώρο και χωρίζοντας τις μεταβλητές μας σε αυτές που χαρακτηρίζουν τη συμπεριφορά του γεγονότος και αυτές που την προκαλούν. Δηλαδή χωρίζουμε τις μεταβλητές σε εξαρτημένες και ανεξάρτητες, για λόγους απλότητας θα αναφερθούμε σε μία εξαρτημένη μεταβλητή η οποία χαρακτηρίζει η συμπεριφορά του γεγονότος.

Αυτή η σχέση μεταξύ της εξαρτημένης μεταβλητής την οποία ονομάζω z και ανεξάρτητων μεταβλητών πού ονομάζω x_i μπορεί να οριστεί και ως μία συνάρτηση πολλών μεταβλητών με είσοδο x_i και έξοδο z.

$$data \Rightarrow z = h(\vec{x}, a^*) \ \ opov \ \vec{x} = (x_1, x_2, \cdots, x_{n-1})$$

Όπου **α*** οι βέλτιστοι παράμετροι - συντελεστές της συνάρτησης .

Η εύρεση ή η προσέγγιση μιας συνάρτησης βάσει των δεδομένων που έχουμε λέγεται παλινδρόμηση και η συνάρτηση πού θα βρούμε ως προσέγγιση λέγεται παλινδρομική(regression) καμπύλη ή best fitting line.

Το βασικό θέμα του regression analysis είναι έχοντας μετρήσεις εξαρτημένων και ανεξάρτητων μεταβλητών να μπορέσω να συμπεράνω τη σχέση, τη συνάρτηση που τις συνδέει . Τα δεδομένα μου είναι οι μετρήσεις ,τα οποία μπορώ να αναπαραστήσω ως σημεία στο χώρο $(x_1, x_2, x_3, \dots, x_{n-1}, z) \in \Re^n$ με τα x_i οι ανεξάρτητες μεταβλητές και το z να είναι η εξαρτημένη.

Αν γνωρίζαμε την ακριβή συνάρτηση h τότε θα μπορούσαμε να προβλέψουμε με ακρίβεια τιμές z ξέροντας τις τιμές των x ,πράγμα πολύ δύσκολο. Παράδειγμα γνωρίζοντας την ατμοσφαιρική πίεση, την υγρασία και την κατεύθυνση των ανέμων θα μπορούσαμε να προβλέψουμε την ακριβή θερμοκρασία που θα έκανε κάθε μέρα .

Η συνάρτηση h δεν έχει κάποια συγκεκριμένη μορφή και τύπου και πολλές φορές είναι σύνθετη και με πολλαπλό τύπο κάνοντας την εύρεση της πολύπλοκη και χρονοβόρα. Ως μία μέθοδος απλοποίησης αυτής της διαδικασίας επιλέγουμε να χάσουμε ακρίβεια ώστε να κερδίσουμε σε απλότητα ,αυτό επιτυγχάνεται χρησιμοποιώντας μία γραμμική προσέγγιση της h , εξού και ο τίτλος linear regression.

$l(\vec{x}) = a_1 + a_2 x_1 + a_3 x_2 + \cdots + a_n x_{n-1}$ *ο τύπος γραμμικής συνάρτησης στο* $\mathbb{R}^n$

Για να βρούμε μία γραμμική συνάρτηση ,η οποία περνάει από όλα τα σημεία(data) που έχουμε, αρκεί να βρούμε την λύση στο γραμμικό σύστημα το οποίο αναπαριστά m εξισώσεις της γραμμικής συνάρτησης με είσοδο ένα σημείο και ως αγνώστους τους a_i συντελεστές .

$X \cdot a = z \quad , X \varepsilon R^{mxn} \ , a \varepsilon R^{nx1} \ , z \varepsilon R^{mx1}$

$$X \cdot a = \begin{bmatrix} 1 & x_{12} & x_{13} & \dots & x_{1n} \\ 1 & x_{22} & x_{23} & \dots & x_{2n} \\ 1 & x_{32} & x_{33} & \dots & x_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_m \end{bmatrix}$$

m: *το πλήθος των σημείων − δεδομένων*

n : *πλήθος των συντελεστών της γραμμικής συνάρτησης*

Overdetermined System

Στην πραγματικότητα δεν είναι ρεαλιστικό να περιμένουμε ότι μία γραμμική συνάρτηση ,που είναι προσέγγιση της κανονικής, θα περνάει από όλα τα σημεία που έχουμε. Αν έχουμε παραπάνω σημεία-δεδομένα(data) από τις μεταβλητές που χαρακτηρίζουν το πρόβλημά μας δηλαδή τις διαστάσεις του χώρου ,υπάρχει περίπτωση να μην ορίζεται γραμμική συνάρτηση τέτοια ώστε να περνάει από όλα τα σημεία (ορίζεται στην περίπτωση που κάποια σημεία είναι ομεία ή γραμμικος συνδυασμός άλλων) , δηλαδή το σύστημα δεν έχει λύση .

$$X \cdot a = z \quad , X \varepsilon R^{mxn} \ , \ a \varepsilon R^{nx1} \ , z \varepsilon R^{mx1} \quad , \quad m > n$$

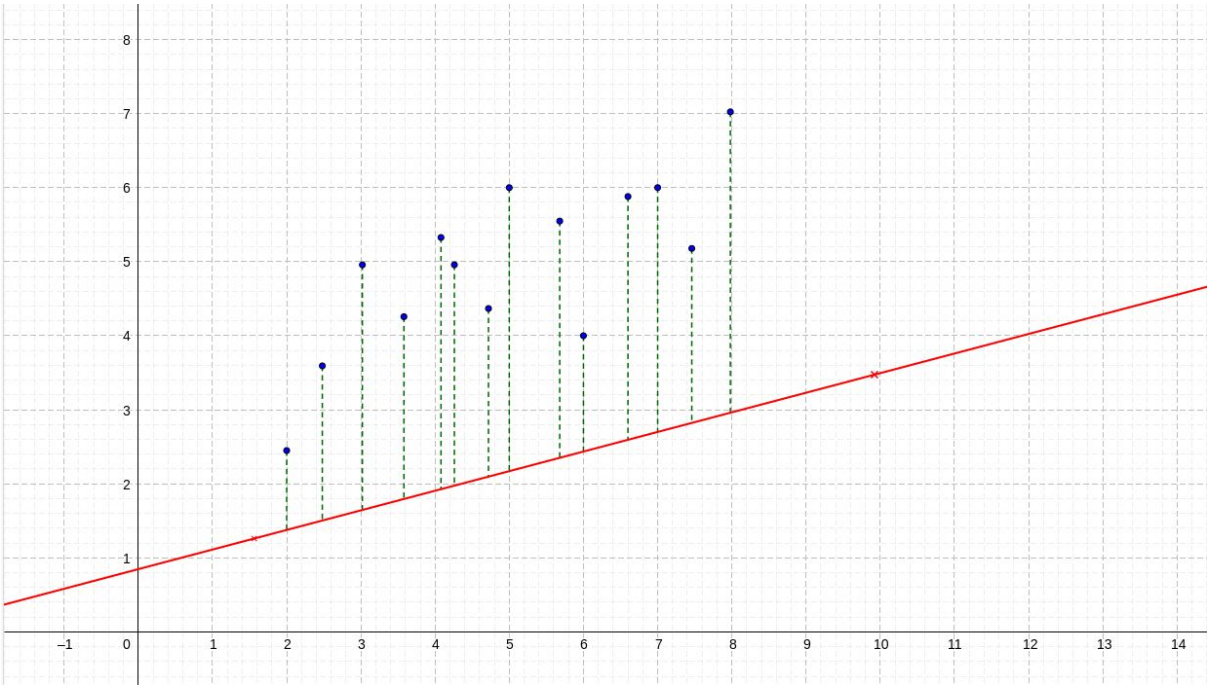
$$X \cdot a = \begin{bmatrix} 1 & x_{12} & x_{13} & \dots & x_{1n} \\ 1 & x_{22} & x_{23} & \dots & x_{2n} \\ 1 & x_{32} & x_{33} & \dots & x_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_m \end{bmatrix}$$

Σε αυτή την περίπτωση δεν έχουμε μία τέλεια λύση οπότε θα αρκεστούμε στο να βρούμε μία γραμμική συνάρτηση ,ώστε τα σημεία της να προσεγγίζουν τα πραγματικά σημεία(data) με το μικρότερο δυνατό σφάλμα (residual).

$$\hat{z}\left(\vec{x} \right) = h\left(\vec{x}, a^{*} \right) + residual$$

$$residual \in \mathbb{R}$$

Στην παρακάτω εικόνα 1.1 παρουσιάζεται με κόκκινο το best fishing line στον δισδιάστατο χώρο και με μπλε σημεία τα δεδομένα μας. Με αυτή την γραμμική συνάρτηση που έχει επιλεχθεί η κατακόρυφη απόσταση των σημείων με αυτήν είναι τα residuals. Γενικά υπάρχουν πολλοί τρόποι να ορίσουμε πόσο καλό fit κάνει η συνάρτηση στα δεδομένα, ο πιο απλός τρόπος είναι να αθροίσουμε όλα τα residuals με σκοπό να ελαχιστοποιήσουμε το άθροισμα.



εικόνα 1.1

Άρα προσπαθούμε να βρούμε τις παραμέτρους μιας γραμμικής συνάρτησης με στόχο να μειώσουμε το συνολικό σφάλμα σε σχέση με τα δεδομένα μας. Αυτό μπορεί να γραφτεί και σαν **πρόβλημα γραμμικού προγραμματισμού**:

$$min \left| z_1 - \hat{z}(\vec{x}_1) \right| + \left| z_2 - \hat{z}(\vec{x}_2) \right| + \cdots + \left| z_m - \hat{z}(\vec{x}_m) \right|$$

$$s.t.\ \ a_i \in \mathbb{R}^n$$

$$\hat{z}(\vec{x}) = a_1 + a_2 x_1 + a_3 x_2 + \cdots + a_n x_{n-1}$$

Αυτό μπορεί να γραφτεί και με μορφή πινάκων:

$$min \left|\left| z - X \cdot a \right|\right|_2 \ , \ a \in \mathbb{R}^n$$

Η προσέγγιση στο πρόβλημα με απόλυτες δεν χρησιμοποιείται λόγω πολυπλοκότητας σαν πρόβλημα γραμμικού και αποφυγής υπολογισμού ριζών στην μορφή πινάκων. Αντί της απόλυτης τιμής χρησιμοποιείται το τετράγωνο της ποσότητας ως ισοδύναμη λύση αφού μία ποσότητα τετράγωνο γίνεται ελάχιστη, όταν ελαχιστοποιείται η απόλυτη τιμή αυτής. Αρα παρουσιάζουν στο ίδιο σημείο ακρότατο, αρα είναι το ισοδύναμο πρόβλημα.

$$min \left(z_1 - \hat{z}(\vec{x}_1) \right)^2 + \left(z_2 - \hat{z}(\vec{x}_2) \right)^2 + \cdots + \left(z_m - \hat{z}(\vec{x}_m) \right)^2$$

$$s.t.\ \ a_i \in \mathbb{R}^n$$

$$\hat{z}(\vec{x}) = a_1 + a_2 x_1 + a_3 x_2 + \cdots + a_n x_{n-1}$$

ή

$$min \left|\left|z - X \cdot a\right|\right|_2^2 \ , \ a \in \ \mathbb{R}^n$$

Αυτή η μέθοδος λέγεται των ελαχίστων τετραγώνων και είναι από τις πιο γνωστές μεθόδους για την εύρεση παλινδρομικής καμπύλη.

Finding The Parameters

Υπάρχουν αρκετοί τρόποι για την εύρεση των παραμέτρων , εμείς θα εστιάσουμε μόνο σε δύο, έναν θεωρητικό τρόπο και έναν επαναληπτικό(προσεγγιστικός).

Εύρεση ολικού ακρότατου:

Αυτός είναι ο πιο ευθύς τρόπος για την εύρεση του ελάχιστου της συνάρτησης κόστους. Χρησιμοποιώντας παράγωγο πολλών μεταβλητών βρίσκουμε την κλήση της συνάρτησης κόστους και λύνουμε την εξίσωση $\nabla \cdot f \ = \mathbf{0}$ για την εύρεση ακροτάτου.

$$min \left|\left|X \cdot a - z\right|\right|_2^2 \ , \ a \in \ \mathbb{R}^n$$

$$\Leftrightarrow min \ (X \cdot a - z)^T (X \cdot a - z) \ , \ \pi\rho\acute{\epsilon}\pi\epsilon\iota \ \nu\alpha \ \beta\rho\acute{\omega} \ \tau\alpha \ \alpha_i : \frac{\mathrm{d}\left[(X \cdot \boldsymbol{a} - z)^T (X \cdot \boldsymbol{a} - z)\right]}{\mathrm{d}\boldsymbol{a}} = 0$$

$$\Leftrightarrow (X \cdot \boldsymbol{a} - z)^T \cdot X + X^T \cdot (X \cdot \boldsymbol{a} - z) = 0 \quad \Leftrightarrow \quad (X \cdot \boldsymbol{a} - z)^T \cdot X + (X \cdot \boldsymbol{a} - z)^{\mathrm{T}} \cdot (A^T)^T = 0$$

$$\Leftrightarrow 2 \cdot (X \cdot \boldsymbol{a} - z)^T \cdot X = 0 \quad \Leftrightarrow \quad X^T \cdot \boldsymbol{a}^{\mathbf{T}} \cdot X - z^T \cdot X = 0$$

$$\Leftrightarrow X^T \cdot X \cdot \boldsymbol{a} = z^T \cdot X \quad \Leftrightarrow \quad \boldsymbol{a} = (X^T \cdot X)^{-1} \cdot (z^T \cdot X)$$

Gradient descent:

Ο αλγόριθμος gradient descent είναι επαναληπτικός με τοπική σύγκλιση και ξεκινάει με μία αρχική τιμή για τις παραμέτρους a_i και χρησιμοποιώντας την κλήση-παράγωγο της συνάρτησης κόστους, η οποία εξαρτάται από αυτές , προσπαθεί να βρει που παρουσιάζει ελάχιστο.Αυτό το κάνει προσπαθώντας σε κάθε επανάληψη να μηδενίσει την τιμή όλων των μερικών παραγωγών της συνάρτησης.

Αυτό γίνεται με τη χρήση της σταθεράς learning rate η οποία ορίζεται πριν αρχίσει ο επαναληπτικός αλγόριθμος και αντιπροσωπεύει το μέγεθος του βήματος που θα κάνουμε. Αν το βήμα ήταν σταθερό τότε πιθανόν ο αλγόριθμος μας να ήταν πολύ αργός, για αυτό το λόγο πολλαπλασιάζουμε το learning rate με την κλίση της καμπύλης στο σημείο που

βρισκόμαστε αυτή την επανάληψη. Με αυτό τον τρόπο επιτυγχάνουμε μεγαλύτερα βήματα όταν η παραγωγός έχει μεγάλη τιμή και πιο μικρά βήματα όταν πλησιάζουμε την να μηδενίσουμε. Συνήθως επιλέγεται μεγάλο learning rate για τις πρώτες προσπάθειες καθώς δίνει ικανοποιητικές λύσεις για οποιαδήποτε καμπύλη διατρέχει ο αλγόριθμος, σε αντίθεση με ένα μικρό learning rate το οποίο θα καθυστερήσει πολύ τους υπολογισμούς.

$$a_1^{(i+1)} = a_1^{(i)} - (learning\ rate) \cdot \frac{\partial f\left(a_1^{(i)}, a_2^{(i)}, \cdots, a_n^{(i)}\right)}{\partial a_1}$$

$$a_2^{(i+1)} = a_2^{(i)} - (learning\ rate) \cdot \frac{\partial f\left(a_1^{(i)}, a_2^{(i)}, \cdots, a_n^{(i)}\right)}{\partial a_2}$$

$$\vdots$$

$$a_n^{(i+1)} = a_n^{(i)} - (learning\ rate) \cdot \frac{\partial f\left(a_1^{(i)}, a_2^{(i)}, \cdots, a_n^{(i)}\right)}{\partial a_n}$$

Σε πιο απλή μορφή οπου το a είναι διάνυσμα όλων των a_i

$$a^{(i+1)} = a^{(i)} - (learning\ rate)\ \nabla \cdot f\left(a^{(i)}\right)$$

Όπου f η συνάρτηση κόστους που προσπαθούμε να ελαχιστοποιήσουμε, στην περίπτωση πού είμαστε είναι :

$$f\left(\vec{\mathrm{a}}\right) = \left(z_1 - \hat{z}\left(\vec{x}_1\right)\right)^2 + \left(z_2 - \hat{z}\left(\vec{x}_2\right)\right)^2 + \cdots + \left(z_m - \hat{z}\left(\vec{x}_m\right)\right)^2$$

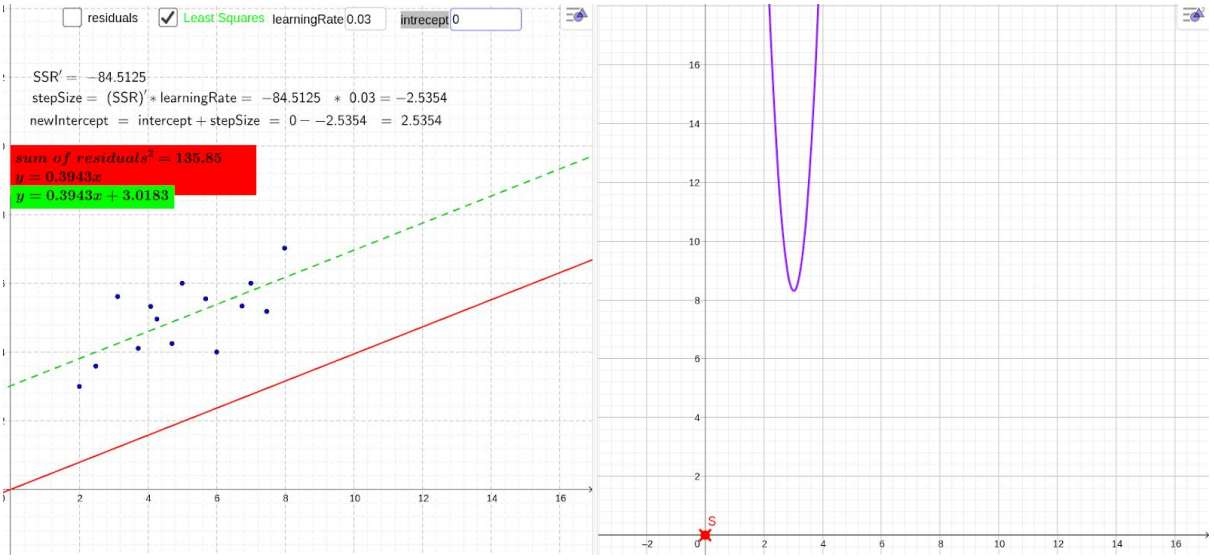
$$\hat{z}\left(\vec{x}\right) = a_1 + a_2 x_1 + a_3 x_2 + \cdots + a_n x_{n-1}$$

Στην υλοποίηση αυτού του αλγόριθμου δεν είναι ρεαλιστικό να περιμένουμε ότι θα βρεθεί λύση ,όποτε έχουμε συνθήκες τερματισμού όπως μέγιστος αριθμός επαναλήψεων και ελάχιστη διαφορά από το 0 δηλαδή μία μικρή τιμή ε>0 για την οποία ισχύει :

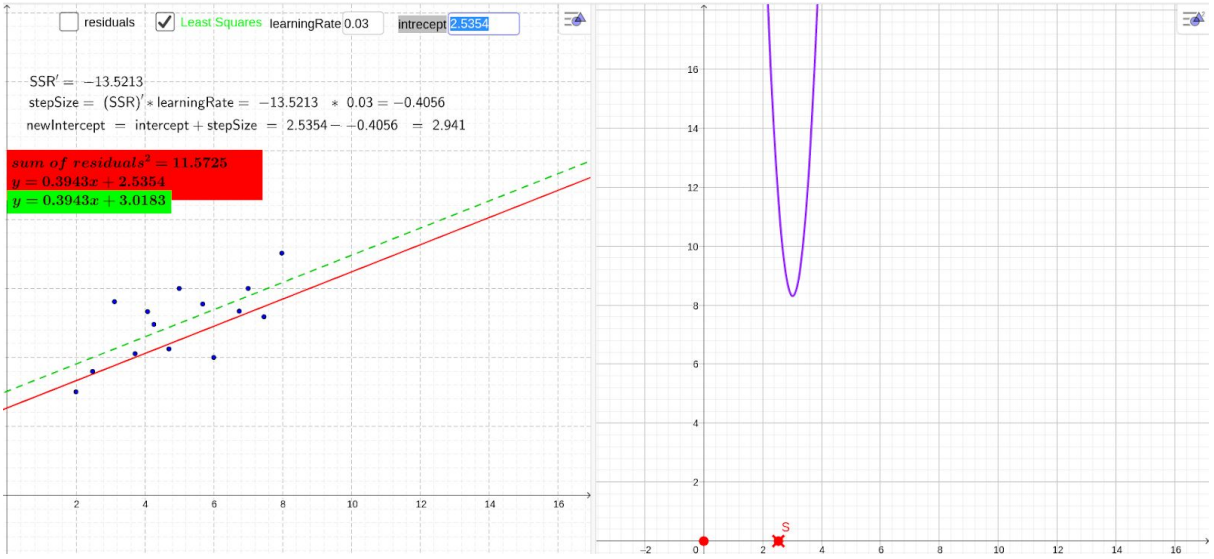
$$\left|\nabla \cdot f\left(\vec{a}^{(i)}\right)\right| < \varepsilon \quad , \varepsilon > 0$$

Η εικόνες(1.2.1 - 1.2.6) περιγράφει τον αλγόριθμο gradient descent με την υπόθεση ότι έχουμε δεδομένο το β της καμπύλης. Η εικόνα χωρίζει χωρίζεται σε δύο πάνελ το δεξί δείχνει την συνάρτηση των summed squared residuals(SSR) ως προς το intercept δηλαδή το β στην εξίσωση ευθείας y = αx+β. Στο κάτω μέρος του πάνελ φαίνονται οι τιμές του SSR για διάφορα intercept που δοκιμάζονται κατά τη διάρκεια αλγόριθμου, με μηδέν να είναι η αρχική τιμή και με S να συμβολίζεται το τελευταίο intercept πού χρησιμοποιούμε. Γραφικά είναι ξεκάθαρο πως η συνάρτηση SSR παρουσιάζει ελάχιστο κοντά στο 3. Στο αριστερό πάνελ φαίνονται τα data points καθώς και με κόκκινο το fitting line πού αντιστοιχεί στο intercept αυτής της επανάληψης, ενώ με πράσινο είναι το best fishing line που θέλουμε να

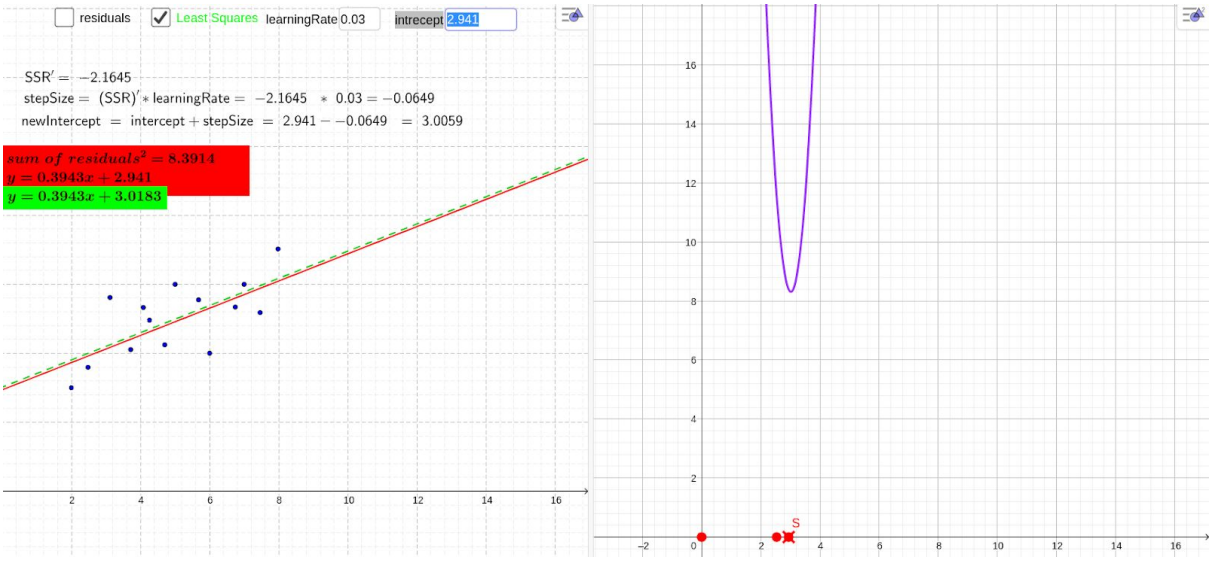
προσεγγίσουμε. Ακριβώς παραπάνω είναι γραμμένοι οι υπολογισμοί που κάνει ο αλγόριθμος σε κάθε βήμα.



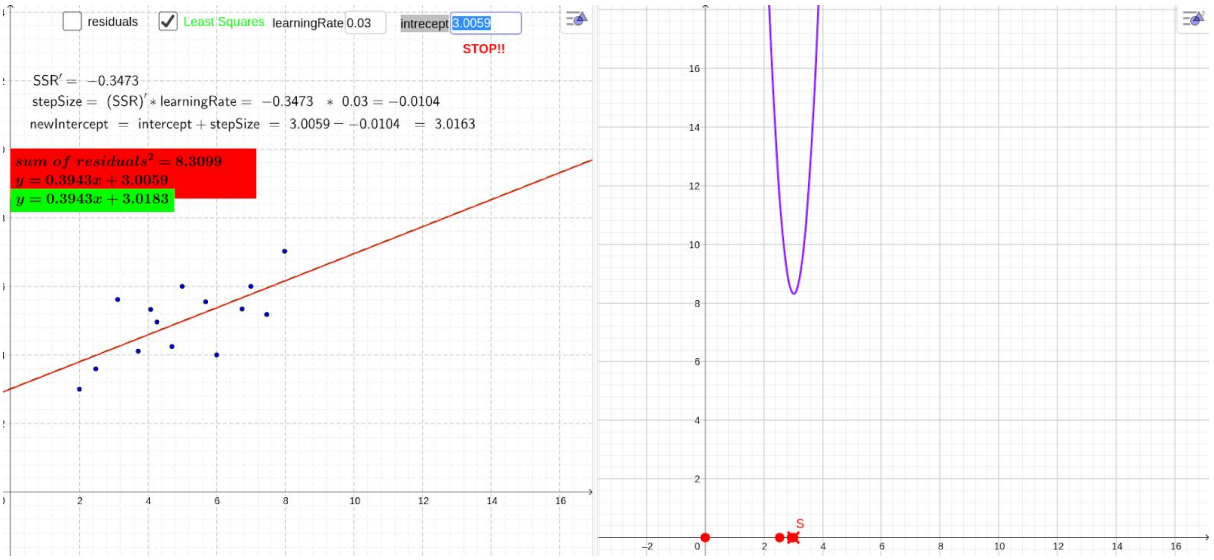
εικόνα 1.3.1 good learning rate



εικόνα 1.3.2 good learning rate

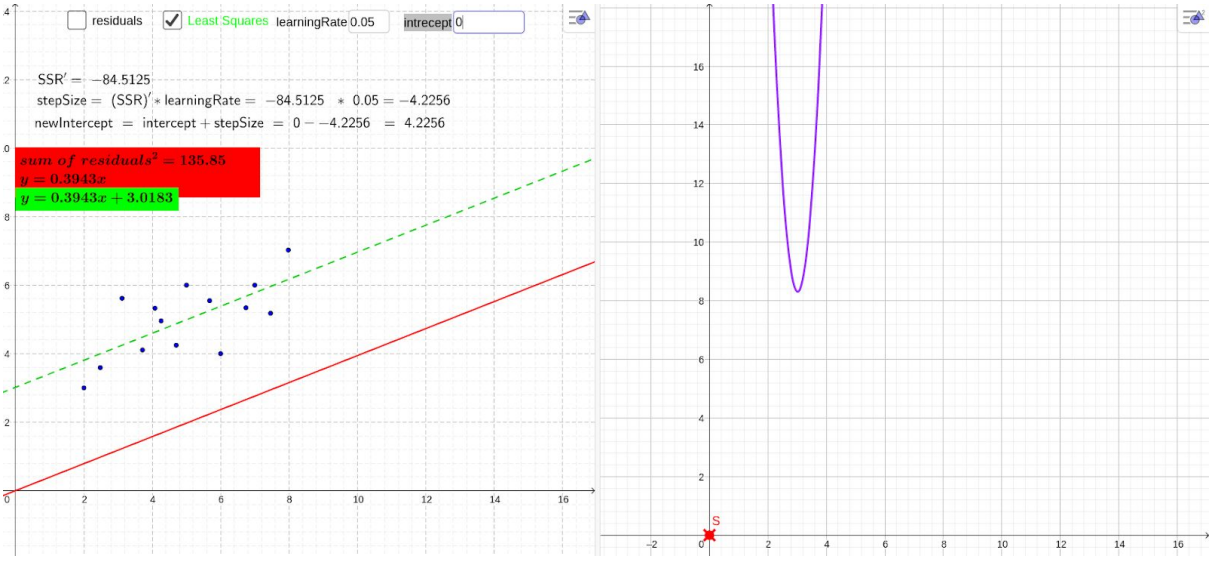


εικόνα 1.3.3 good learning rate

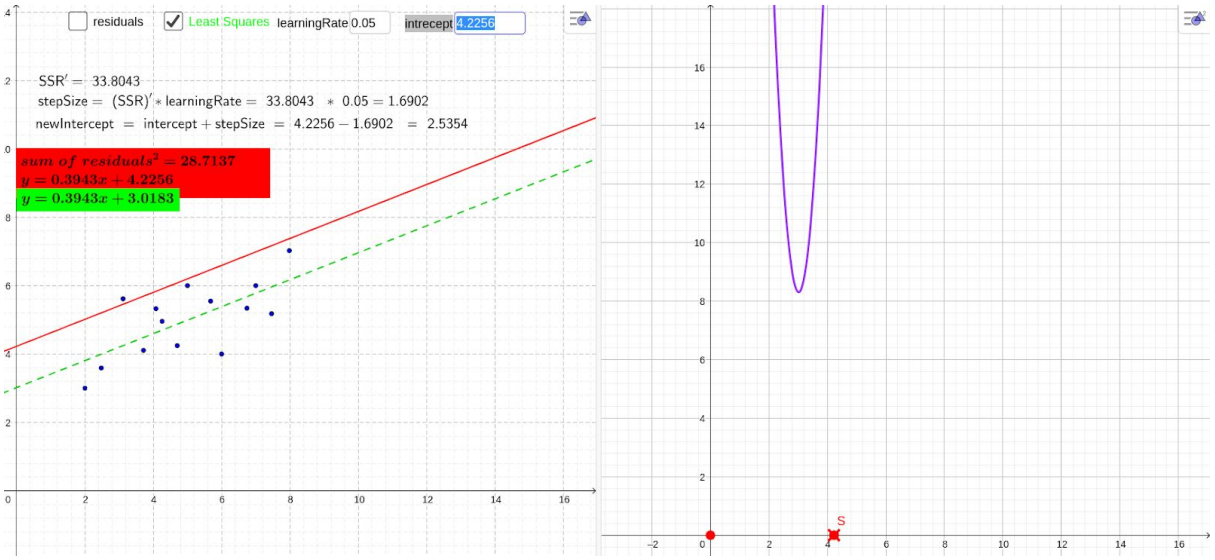


1.3.4 good learning rate

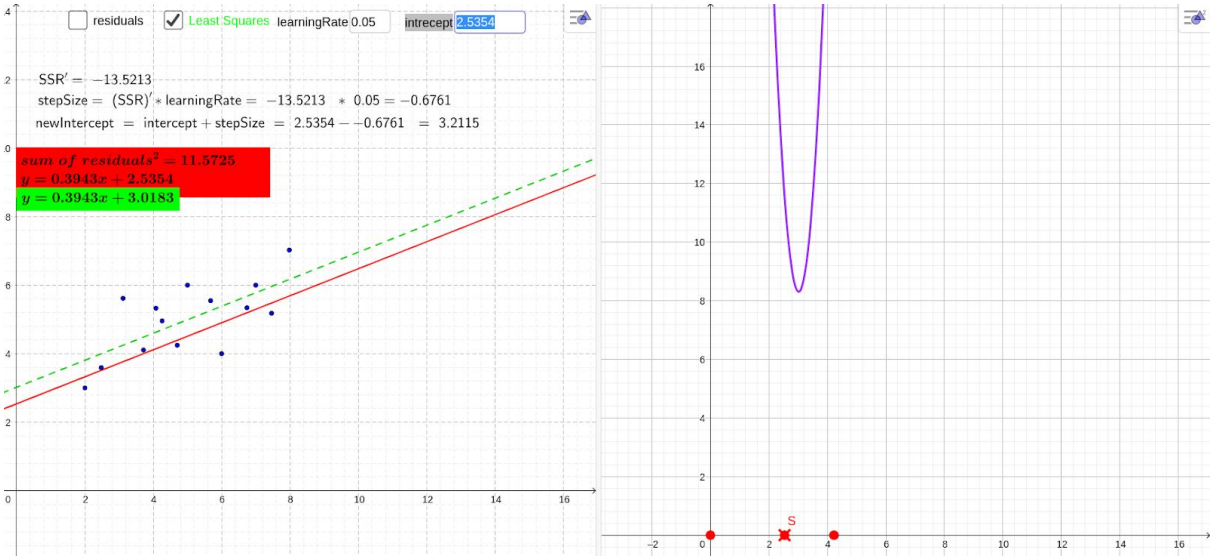
Με μια μικρή αλλαγή κατα 0.02 του learning rate αλλάζει την ακρίβεια και την ταχύτητα σύγκλισης.



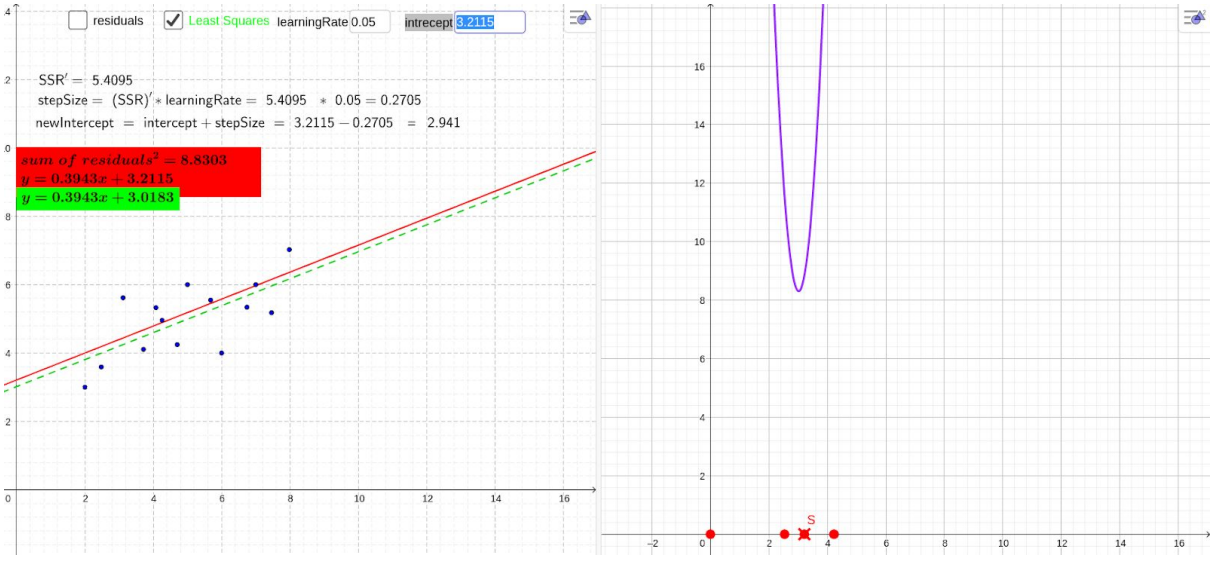
1.2.1 bad learning rate



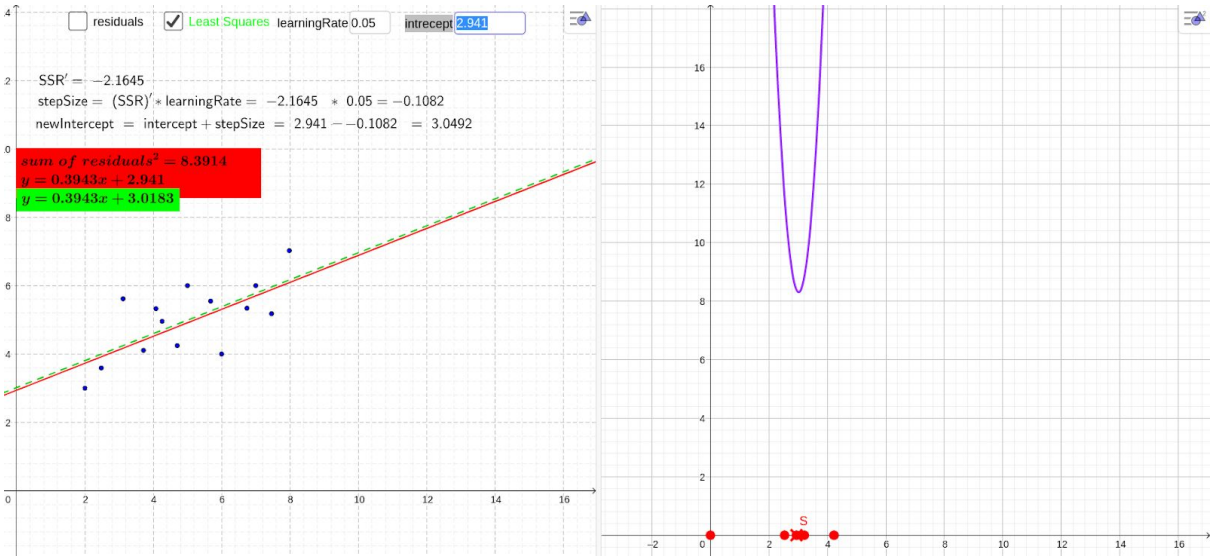
1.2.2 bad learning rate



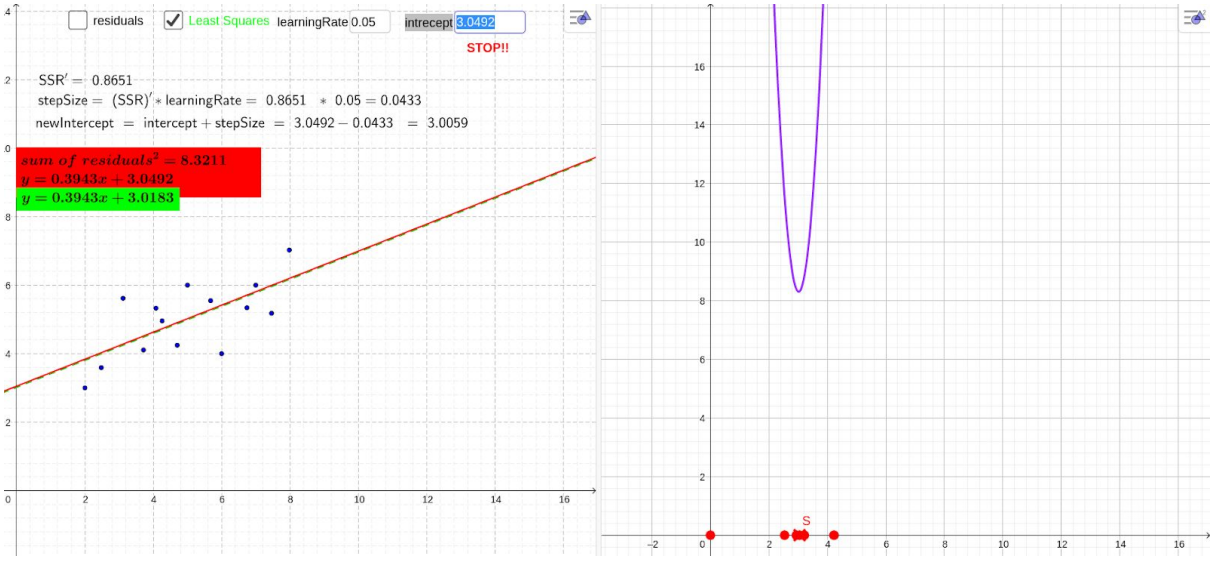
1.2.3 bad learning rate



1.2.4 bad learning rate



1.2.5 bad learning rate



1.2.6 bad learning rate

Παρατηρήσεις οι εικόνες 1.2 και 1.3 είναι για να τονίσουν τη σημασία του learning rate, καθώς στις πρώτες εικόνες έχουμε περισσότερες επαναλήψεις σε σχέση με τις επόμενες, οι οποίες έχουν λιγότερες επαναλήψεις και στο τέλος καταλήγουν σε μία πιο ακριβής λύση.

Τέλος ο αλγόριθμος αυτός είναι αρκετά χρήσιμος όταν δεν μπορεί να βρεθεί ακριβής ρίζα με τον πιο πάνω αναλυτικό τρόπο , αλλά κάθε επανάληψη έχει μεγάλο πλήθος από πράξεις καθώς εξαρτάται από το πλήθος των δεδομένων μας ,το οποίο συνήθως είναι μεγάλο. Χρήσιμη γίνεται η τροποποίηση του αλγόριθμου ώστε να μειωθούν οι πράξεις.

Αλγόριθμος : gradient descent

*Βήμα*1: *ανάθεση αρχικής τιμής* $a^{(0)}$ *και* $i = 0$ *και* $f(\vec{a}) = \left(z_1 - \hat{z}(x_1)\right)^2 + \cdots + \left(z_n - \hat{z}(x_n)\right)^2$

*Βήμα*2: *υπολογισμός* $\nabla \cdot f$ *και* $\nabla \cdot f(a^{(i)})$

*Βήμα*3: *υπολογισμός του νέου* $a^{(i+1)} = a^{(i)} - \gamma \cdot \nabla \cdot f(a^{(i)})$

Αν $i >$ *max iterations*

Return $a^{(i+1)}$

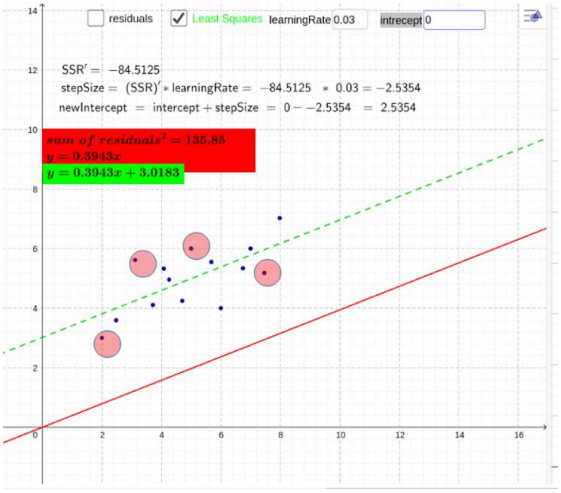
Αν $\left| a^{(i+1)} \right| <$ *tolerance*

Return $a^{(i+1)}$

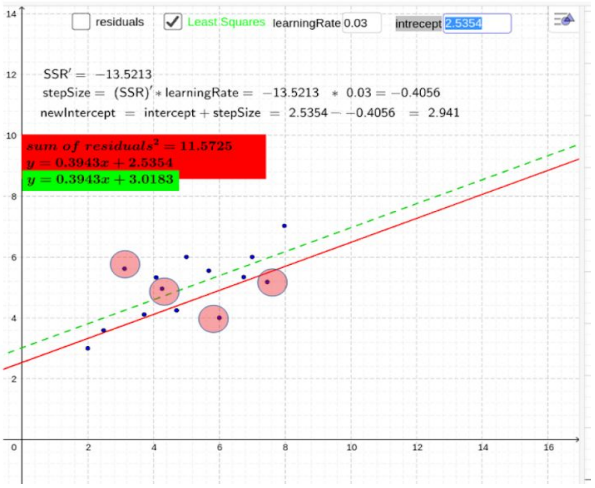
goto Βήμα 2

Stochastic gradient descent:

Αυτή η μέθοδος χρησιμοποιεί τον παραπάνω αλγόριθμο με τη διαφορά ότι σε κάθε επανάληψη για τον υπολογισμό του $a^{(i)}$ επιλέγεται σταθερό πλήθος όρων που είναι υψωμένοι στο τετράγωνο δηλαδή ο αλγόριθμος μας λαμβάνει υπόψη του σταθερό πλήθος από τα σημεία- δεδομένα. Με αυτό τον τρόπο επιτυγχάνουμε ταχύτερη εκτέλεση με αντάλλαγμα την ακρίβεια . Ιδιαίτερο ενδιαφέρον δείχνει η μέθοδος στην περίπτωση ύπαρξης clusters.



εικόνα 1.4



εικόνα 1.5

Στην παραπάνω εικόνα (1.4) εμφανίζεται ένα παράδειγμα του stochastic gradient descent, όπου στο πρώτο βήμα λαμβάνονται υπόψη μόνο τέσσερα τυχαία δεδομένα κι έπειτα στο επόμενο αλλα τέσσερα(1.5), για τον υπολογισμό της κλίσης της συνάρτησης κόστους.

$Αλγόριθμος$: $stochastic\ gradient\ descent$

$Βήμα 1$: $ανάθεση\ αρχικής\ τιμής\ \ a^{(0)}\ και\ i = 0\ και\ k$

$Βημα 2$: $επιλογή\ k\ τυχαίων\ σημείων\ απο\ το\ dataset$

$Βήμα 3$: $υπολογισμός\ της\ f(\vec{a}) = \left(z_1 - \hat{z}(x_1)\right)^2 + \cdots + \left(z_k - \hat{z}(x_k)\right)^2$

$υπολογισμός\ \nabla \cdot f\ \ και\ \nabla \cdot f(a^{(i)})$

$Βήμα 4$: $υπολογισμός\ του\ νέου\ \ a^{(i+1)} = a^{(i)} - \gamma \cdot \nabla \cdot f(a^{(i)})$

$Αν\ i > max\ iterations$

$Return\ \ a^{(i+1)}$

$Αν\ \left|a^{(i+1)}\right| < tolerance$

$Return\ \ a^{(i+1)}$

$goto\ Βήμα\ 2$

Κώδικας python

Έχω βρει δυο ανοιχτά datasets, ένα για classification και ένα για regression. Έχω γράψει κάποιο ενδεικτικό κώδικα στο Jupyter notebook python για να εκπαιδεύσω και να τεστάρω τους αλγόριθμους multiple linear regression, lasso regression, Ridge regression, elastic net regression, SVM, logistic regression.

Παρακάτω έχω κάποιες πληροφορίες για κάθε dataset κατευθείαν από την σελίδα τους. Παρατηρούμε ότι κάποια στοιχεία μπορεί να λείπουν ή να έχουν inf values. Υπάρχουν πολλοί τρόποι να το χειριστούμε αυτό (π.χ. απόρριψη τιμών) εγώ επέλεξα να βάλω τον μέσο όρο της στήλης.

Pima Indians Diabetes Dataset

The Pima Indians Diabetes Dataset involves predicting the onset of diabetes within 5 years in Pima Indians given medical details.

A sample of the first 5 rows is listed below.

6,148,72,35,0,33.6,0.627,50,1

1,85,66,29,0,26.6,0.351,31,0

8,183,64,0,0,23.3,0.672,32,1

1,89,66,23,94,28.1,0.167,21,0

0,137,40,35,168,43.1,2.288,33,1

Relevant Information:Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. ADAP is an adaptive learning routine that generates and executes digital analogs of perceptron-like devices. It is a unique algorithm; see the paper for details.

Number of Instances: 768

Number of Attributes: 8 plus class

For Each Attribute: (all numeric-valued)

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Missing Attribute Values: Yes

Class Distribution: (class value 1 is interpreted as "tested positive for diabetes")

Class Value	Number of instances
0	500
1	268

Brief statistical analysis:

Attribute number:	Mean:	Standard Deviation:
1.	3.8	3.4
2.	120.9	32.0
3.	69.1	19.4
4.	20.5	16.0
5.	79.8	115.2
6.	32.0	7.9
7.	0.5	0.3
8.	33.2	11.8

Κώδικας:

In [1]:
```
import pandas as pd
import re
import string
import pickle
import csv
import numpy as np

# dataset
# https://machinelearningmastery.com/standard-machine-learning-datasets/

# read data
Location = r'/home/stratos/Desktop/diabetes'
df = pd.read_csv(Location,names=['times pregnant',
                                 'Plasma glucose concentration ',
                                 'blood pressure','Triceps skinfold thickness',
                                 'serum insulin','Body mass index',
                                 'Diabetes pedigree function','age','will have
diabetes'],
                 error_bad_lines=False, sep=',')

df
```

Out[1]:

	times pregnant	Plasma glucose concentration	blood pressure	Triceps skinfold thickness	serum insulin	Body mass index	Diabetes pedigree function	age	will have diabetes
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [2]:
```
# fill nan colums with mean
df=df.replace([np.inf, -np.inf], np.nan)
df=df.fillna(df.mean())

print("size="+str(df.shape))
```

```
size=(768, 9)
```

In [3]:
```
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

train, test = train_test_split(df, test_size=0.2)

x_train=train.loc[:, train.columns != 'will have diabetes']
y_train=train['will have diabetes']

x_test=test.loc[:, test.columns != 'will have diabetes']
y_test=test['will have diabetes']
```

In [4]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(dual=False,solver='liblinear')

lr.fit(x_train,y_train)
prediction = lr.predict(x_test)
print("LogisticRegression")
score=f1_score(np.asarray(y_test),prediction.flatten(),average='macro')*100
print(str(score)+" %")
```

```
LogisticRegression
76.75120772946859 %
```

In [5]:
```
from sklearn.svm import LinearSVC
svm = LinearSVC(dual=False)

svm.fit(x_train,y_train)
prediction = svm.predict(x_test)
print("LogisticRegression")
score=f1_score(np.asarray(y_test),prediction.flatten(),average='macro')*100
print(str(score)+" %")
```

```
LogisticRegression
77.03320739709683 %
```

Life Expectancy

Content

The project relies on accuracy of data. The Global Health Observatory (GHO) data repository under World Health Organization (WHO) keeps track of the health status as well as many other related factors for all countries The data-sets are made available to public for the purpose of health data analysis. The data-set related to life expectancy, health factors for 193 countries has been collected from the same WHO data repository website and its corresponding economic data was collected from United Nation website. Among all categories of health-related factors only those critical factors were chosen which are more representative. It has been observed that in the past 15 years , there has been a huge development in health sector resulting in improvement of human mortality rates especially in the developing nations in comparison to the past 30 years. Therefore, in this project we have considered data from year 2000-2015 for 193 countries for further analysis. The individual data files have been merged together into a single data-set. On initial visual inspection of the data showed some missing values. As the data-sets were from WHO, we found no evident errors. Missing data was handled in R software by using Missmap command. The result indicated that most of the missing data was for population, Hepatitis B and GDP. The missing data were from less known countries like Vanuatu, Tonga, Togo, Cabo Verde etc. Finding all data for these countries was difficult and hence, it was decided that we exclude these countries from the final model data-set. The final merged file(final dataset) consists of 22 Columns and 2938 rows which meant 20 predicting variables. All predicting variables was then divided into several broad categories:Immunization related factors, Mortality factors, Economical factors and Social factors.

Acknowledgements

The data was collected from WHO and United Nations website with the help of Deeksha Russell and Duan Wang.

Inspiration

The data-set aims to answer the following key questions: 1. Does various predicting factors which has been chosen initially really affect the Life expectancy? What are the predicting variables actually affecting the life expectancy? 2. Should a country having a lower life expectancy value(<65) increase its healthcare expenditure in order to improve its average lifespan? 3. How does Infant and Adult mortality rates affect life expectancy? 4. Does Life Expectancy has positive or negative correlation with eating habits, lifestyle, exercise, smoking, drinking alcohol etc. 5. What is the impact of schooling on the lifespan of humans? 6. Does Life Expectancy have positive or negative relationship with drinking alcohol? 7. Do

densely populated countries tend to have lower life expectancy? 8. What is the impact of Immunization coverage on life Expectancy?

5 first rows:

1	Afghanistan	2015	Developing	65	263	62	0.01	71.27962362	65	1154	19.1	83	6	8.16	65	0.1	584.25921	33736494	17.2	17.3
2	Afghanistan	2014	Developing	59.9	271	64	0.01	73.52358168	62	492	18.6	86	58	8.18	62	0.1	612.696514	327582	17.5	17.5
3	Afghanistan	2013	Developing	59.9	268	66	0.01	73.21924272	64	430	18.1	89	62	8.13	64	0.1	631.744976	31731688	17.7	17.7
4	Afghanistan	2012	Developing	59.5	272	69	0.01	78.1842153	67	2787	17.6	93	67	8.52	67	0.1	669.959	3696958	17.9	18
5	Afghanistan	2011	Developing	59.2	275	71	0.01	7.097108703	68	3013	17.2	97	68	7.87	68	0.1	63.537231	2978599	18.2	18.2

In [1]:
```
import pandas as pd
import re
import string
import pickle
import csv
import numpy as np

# dataset
# https://www.kaggle.com/kumarajarshi/life-expectancy-who?fbclid=IwAR1oHJhXt1PjDfcTfj_R4ANlE-KwgmD___O_wvsJea_lRfGXYO3OxpxfAEs

# read data
Location = r'/home/stratos/Desktop/Life_Expectancy_Data.csv'
df = pd.read_csv(Location, error_bad_lines=False, sep=',')

df
```

Out[1]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0
...	...	...	...	...	...	...	...	...	...
2933	Zimbabwe	2004	Developing	44.3	723.0	27	4.36	0.000000	68.0
2934	Zimbabwe	2003	Developing	44.5	715.0	26	4.06	0.000000	7.0
2935	Zimbabwe	2002	Developing	44.8	73.0	25	4.43	0.000000	73.0
2936	Zimbabwe	2001	Developing	45.3	686.0	25	1.72	0.000000	76.0
2937	Zimbabwe	2000	Developing	46.0	665.0	24	1.68	0.000000	79.0

2938 rows × 22 columns

In [2]:
```
# replace countries with numbers
countries=pd.unique(df['Country'])
index=dict(zip(countries, range(len(countries))))
df['Country']=df['Country'].replace(index)

# replace status with numbers
countries=pd.unique(df['Status'])
index=dict(zip(countries, range(len(countries))))
df['Status']=df['Status'].replace(index)

# drop country, status nan rows
df['Country']=df['Country'].replace([np.inf, -np.inf], np.nan)
df['Country']=df['Country'].dropna()
df['Status']=df['Status'].replace([np.inf, -np.inf], np.nan)
df['Status']=df['Status'].dropna()

# fill nan colums with mean
df=df.replace([np.inf, -np.inf], np.nan)
df=df.fillna(df.mean())

print("size="+str(df.shape))
```

```
size=(2938, 22)
```

In [3]:
```
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
import sklearn.metrics as metrics

train, test = train_test_split(df, test_size=0.2)

x_train=train.loc[:, train.columns != 'Life expectancy ']
y_train=train['Life expectancy ']

x_test=test.loc[:, test.columns != 'Life expectancy ']
y_test=test['Life expectancy ']
```

In [4]:
```
from sklearn import linear_model
regr = linear_model.LinearRegression()

regr.fit(x_train, y_train)
prediction = regr.predict(x_test)
print("least squares")
print(metrics.mean_absolute_error(y_test, prediction))
print(metrics.mean_squared_error(y_test, prediction))
```

```
least squares
3.179632446809571
18.535849825375227
```

In [5]:
```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=1e-10)

ridge.fit(x_train,y_train)
prediction = ridge.predict(x_test)
print("Ridge λ="+str(1e-10))
print(metrics.mean_absolute_error(y_test, prediction))
print(metrics.mean_squared_error(y_test, prediction))
```

```
Ridge λ=1e-10
3.1796324468105595
18.535849825382172
```

```
/home/stratos/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/rid
ge.py:147: LinAlgWarning: Ill-conditioned matrix (rcond=4.18279e-18): result
may not be accurate.
  overwrite_a=True).T
```

In [6]:
```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=1.0)

ridge.fit(x_train,y_train)
prediction = ridge.predict(x_test)
print("Ridge λ="+str(1.0))
print(metrics.mean_absolute_error(y_test, prediction))
print(metrics.mean_squared_error(y_test, prediction))
```

```
Ridge λ=1.0
3.1814080818165196
18.54616453438569
```

```
/home/stratos/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/rid
ge.py:147: LinAlgWarning: Ill-conditioned matrix (rcond=4.31647e-18): result
may not be accurate.
  overwrite_a=True).T
```

In [7]:
```
from sklearn.linear_model import Lasso
ridge = Lasso(tol=0.1)

ridge.fit(x_train,y_train)
prediction = ridge.predict(x_test)
print("Lasso λ="+str(0.1))
print(metrics.mean_absolute_error(y_test, prediction))
print(metrics.mean_squared_error(y_test, prediction))
```

```
Lasso λ=0.1
3.2820920573209005
19.730010714932146
```

```
/home/stratos/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/coo
rdinate_descent.py:475: ConvergenceWarning: Objective did not converge. You
might want to increase the number of iterations. Duality gap: 21326.00898600
2213, tolerance: 21218.054175887937
  positive)
```

Βιβλιογραφικές Αναφορές

Αναφορές

1.https://www.khanacademy.org/math/linear-algebra/alternate-bases/orthogonal-projections/v/linear-algebra-least-squares-approximation

2.https://math.stackexchange.com/questions/2829437/understanding-the-absolute-value-of-a-matrix

3.https://people.sc.fsu.edu/~jpeterson/linear_least_squares.pdf

4.http://youtube.com/watch?v=fkS3FkVAPWU

5.http://en.wikipedia.org/wiki/Matrix_calculus

6.http://en.wikipedia.org/wiki/Gradient_descent

7.http://stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec6.pdf

8.https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.names

9.https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

10.https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model