Προπτυχιακό μάθημα: Μηχανική Μάθηση

Τμήμα Μηχανικών Η/Υ & Πληροφορικής, Πανεπιστήμιο Ιωαννίνων,

Ακαδημαϊκό έτος 2021-22

Μέλη Ομάδας :

Χρήστος Καραγιαννίδης , ΑΜ : 4375

Νεφέλη-Ελένη Κατσιλέρου , ΑΜ :4385

1η Σειρά Ασκήσεων

Πρώτη Μέθοδος: Κ--Κοντινότεροι Γείτονες

Η μέθοδος Κ-κοντινότεροι γείτονες (kNN) είναι ένας απλός αλγόριθμος μηχανικής μάθησης που ανήκει στην supervised κατηγορία μεθόδων και χρησιμοποιείτε είτε για κατηγοριοποίηση είτε για προβλήματα παλινδρόμησης. Η μέθοδος αυτή υλοποιείται με βάση την υπόθεση ότι παρόμοια αντικείμενα βρίσκονται σε κοντινή απόσταση. Για την υλοποίηση της χρησιμοποιήσαμε την Ευκλείδεια απόσταση . Πιο συγκεκριμένα :

Η συνάρτηση openTest() ανοίγει το αρχείο test.csv και επιστρέφει τα περιεχόμενα του σε ένα πίνακα . Το αντίστοιχο κάνει και η openTrain() με το αρχείο train.csv . Η convertColorInTest() παίρνει ως όρισμα ένα πίνακα και μετατρέπει τα χρώματα του σε διακριτές τιμές από (0-1) , το ίδιο κάνει και η convertColorAndTypeInTrain() με τη διαφορά ότι αλλάζει και τα είδη των τεράτων σε διακριτές τιμές. Στη συνέχεια έχουμε τη συνάρτηση euclidean_distance που υπολογίζει την απόσταση μεταξύ δύο γραμμών των δύο πινάκων (train , test) για κάθε στήλη τους εκτός από το id με βάση τον μαθηματικό τύπο . Η συνάρτηση get_neighbors βρίσκει την απόσταση μιας γραμμής του test.csv με όλες τις γραμμές του train.csv και επιστρέφει ένα πίνακα με τους number_of_neighbors κοντινότερους γείτονες. Η predict_class βρίσκει τι είδους τέρατα είναι οι περισσότεροι γείτονες και με βάση τι είναι σε πλήθος οι περισσότεροι επιστρέφει το prediction. Η convertToString δίνει στη λίστα με τα predicitions το όνομα του τέρατος ως string. Επίσης η write_csv_with_results() γράφει σε ένα file τα predictions με το format που ζητείτε από το kaggle και οι RunKnnForK() συναρτήσεις τρέχουν τον αλγόριθμο για τις k τιμές που αναφέρονται στην εκφώνηση .

Τα αποτελέσματα της υλοποίησης μας μετά από ανάρτηση στο kaggle είναι τα εξής :

K	Accuracy score	F1 score
1	0.68998	
3	0.68241	
5	0.70888	
7	0.71833	
10	0.69943	

Παρατηρούμε ότι το καλύτερο accuracy score το έχουμε για k=7 .

Δεύτερη Μέθοδος : Νευρωνικά Δίκτυα

Το νευρωνικό δίκτυο είναι ένα κύκλωμα διασυνδεδεμένων νευρώνων εμπνευσμένο από το Κεντρικό Νευρικό Σύστημα , το οποίο προσπαθεί να προσομοιώσει . Τα νευρωνικά δίκτυα είναι εφαρμόσιμα σχεδόν σε κάθε κατάσταση στην οποία ισχύει μια σχέση μεταξύ μεταβλητών πρόβλεψης (εισροές) και προβλεπόμενες μεταβλητές (εκροές), ακόμα και όταν αυτή η σχέση είναι πολύ περίπλοκη για να αποδοθεί με τους συνηθισμένους όρους της «συσχέτισης» ή των «διαφόρων ομάδων». Αποτελούνται από ένα input layer , ένα ή περισσότερα hidden layers και ένα output layer . Κάθε στοιχείο από το input layer συνδέεται με όλους τους νευρώνες από το πρώτο hidden layer και κάθε ακμή έχει ένα βάρος . Η τιμές αυτών των βαρών αποτελούν τη "γνώση" του νευρωνικού δικτύου καθώς μεταβάλλονται κατά τη διάρκεια της εκπαίδευσης. Συγκεκριμένα στην άσκηση μας δημιουργήσαμε δύο μοντέλα , το πρώτο με ένα κρυμμένο επίπεδο και το δεύτερο με δύο . Κάθε νευρώνας έχει ως συνάρτηση ενεργοποίησης τη σιγμοειδή (tanh) και στο επίπεδο εξόδου την softmax. Για την υλοποίηση της χρησιμοποιήσαμε τη βιβλιοθήκη keras .

Αναλυτικότερα :

Η openFile() ανοίγει τα αρχεία train.csv και test.csv και επιστρέφει έναν πίνακα . Η convertColorInTrain() αλλάζει τα χρώματα των τεράτων σε διακριτές τιμές και τον τύπο τους σε onehot encoding . Η convColor() μετατρέπει μόνο το χρώμα σε αριθμό . Ανοίγουμε τα δυο αρχεία και μετατρέπουμε τα χρώματα τους σε floats και το type στο train.csv σε float. Χωρίζουμε το test.csv σε δυο πίνακες , ο ένας έχει μόνο τα id και ο άλλος όλα τα υπόλοιπα χαρακτηριστικά και στο train.csv κρατάμε ξεχωριστά το type σε ένα πίνακα y και όλα τα υπόλοιπα σε ένα πίνακα x.

Με το model=Sequential() του λέμε τι είδους μοντέλο θα έχουμε και με τα model.add() δίνουμε αρχικά το επίπεδο εισόδου , στη συνέχεια το πόσα κρυμμένα επίπεδα θα έχει με βάση την εκφώνηση και τέλος το επίπεδο εξόδου. Η συνάρτηση model.compile() χειρίζεται το loss που συμβαίνει κατά τη διαδικασία του training και χρησιμοποιεί τον optimizer SGD που μας ζητήθηκε για να το βελτιώσει . Στη συνέχεια κάνουμε fit το μοντέλο δίνοντας του τους πίνακες x και y με τα περιεχόμενα που αναφέραμε παραπάνω . Με τη συνάρτηση model.evaluate() υπολογίζουμε το accuracy και τέλος με το model.predict() υπολογίζουμε τα predictions που θα γίνουν . Η συνάρτηση makeFinalFile() μετατρέπει τα predictions στο σωστό format για το kaggle .

Η αξιολόγηση :

1 κρυμμένο επίπεδο	Accuracy score	F1 score
K=50	0.72022	
K=100	0.71644	
K=200	0.71266	

2 κρυμμένα επίπεδα	Accuracy score	F1 score
(K1,K2)=(50,25)	0.71455	
(K1,K2)=(100,50)	0.73156	
(K1,K2)=(200,100)	0.72967	

Όπως φαίνεται παραπάνω , το μοντέλο με τα δύο κρυμμένα επίπεδα αποδίδει καλύτερα.

Τρίτη Μέθοδος : Support Vector Machines

Στη μηχανική μάθηση, οι μηχανές διανυσμάτων υποστήριξης είναι εποπτευόμενα μοντέλα μάθησης με σχετικούς αλγορίθμους μάθησης που αναλύουν δεδομένα για τα ταξινόμηση και ανάλυση παλινδρόμησης και βασίζονται σε πυρήνες. Κατά τη ταξινόμηση των δεδομένων υποθέτουμε ότι ορισμένα δεδομένα ανήκουν σε μία από δύο κλάσεις (binary) και ο στόχος είναι να αποφασίσουμε σε ποια κλάση θα ανήκει το νέο σημείο δεδομένων. Εμείς δεδομένου ότι έχουμε να κατηγοριοποιήσουμε περισσότερες από δύο τάξεις θα εφαρμόσουμε την τεχνική One-vs-All . Στον SVM ένα σημείο θεωρείται ως ένα p-διάστατο διάνυσμα και θέλουμε να γνωρίζουμε αν μπορούμε να διαχωρίσουμε τέτοια σημεία με ένα (p-1)-διάστατο υπερεπίπεδο. Στην άσκηση μελετήσαμε α)την γραμμική συνάρτηση πυρήνα και β)Gaussian συνάρτηση πυρήνα. Για τη υλοποίηση της μεθόδου χρησιμοποιήσαμε τη βιβλιοθήκη sklearn . Πιο συγκεκριμένα :

Η openTrain() ανοίγει το train.csv αρχείο και επιστρέφει έναν πίνακα με τα περιεχόμενα του, το ίδιο κάνει και η openTest() με το test.csv . Η convertColorInTrain μετατρέπει τα χρώματα και τον τύπο των τεράτων σε διακριτές τιμές ενώ η convertColorInTest μόνο τα χρώματα . Φορτώνουμε τους δυο πίνακες train και test, κρατάμε ξεχωριστά τα id και τα υπόλοιπα δεδομένα . Χωρίζουμε το train σε Xtrain και ytrain και χρησιμοποιήσαμε τις κατάλληλες συναρτήσεις(και για linear kernel και για rbf kernel) από τη βιβλιοθήκη sklearn για να κάνουμε train το μοντέλο , τα predictions καθώς και για να βρούμε το accuracy και f1 score του . Η makeFinalFile() μετατρέπει τα predictions στο σωστό format για το kaggle.

Μετά την αξιολόγηση των αποτελεσμάτων βλέπουμε ότι η Γραμμική συνάρτηση πυρήνα αποδίδει ελάχιστα καλύτερα από την Gaussian.

	Accuracy score	F1 score
Linear kernel	0.73156	
Gauss kernel	0.72778	

Τέταρτη Μέθοδος : Naive Bayes

Ο Naive Bayes είναι μια απλή τεχνική για την κατασκευή ταξινομητών: μοντέλα που αντιστοιχούν class labels σε περιπτώσεις προβλήματος, οι οποίες αναπαρίστανται ως διανύσματα τιμών χαρακτηριστικών, όπου οι class labels προέρχονται από κάποιο πεπερασμένο σύνολο. Δεν υπάρχει ένας μόνο αλγόριθμος για την εκπαίδευση τέτοιων ταξινομητών, αλλά μια οικογένεια αλγορίθμων που βασίζονται σε μια κοινή αρχή: όλοι οι ταξινομητές Naive Bayes υποθέτουν ότι η τιμή ενός συγκεκριμένου χαρακτηριστικού είναι ανεξάρτητη από την τιμή οποιουδήποτε άλλου χαρακτηριστικού, δεδομένης της μεταβλητής κλάσης. Έστω ότι έχουμε ένα δειγματικό χώρο δεδομένων που αποτελείται από K ασυμβίβαστες κατηγορίες . Κάθε μία από αυτές έχει μια αδέσμευτη πιθανότητα που αντικατοπτρίζει τη βαρύτητα που έχει αυτή η κατηγορία στο σύνολο δεδομένων. Επιπρόσθετα κάθε κατηγορία έχει μία υπο-συνθήκη κατανομή η οποία προσδιορίζει ένα κανόνα δεδομένων της κατηγορίας και περιγράφεται από τη συνάρτηση πυκνότητας πιθανότητας . Στον κώδικα μας χρησιμοποιήσαμε normal distribution για όλα τα χαρακτηριστικά κάνοντας τις απαραίτητες τροποποιήσεις για το 5ο χαρακτηριστικό διακριτής τιμής.

Πιο συγκεκριμένα :

Η συνάρτηση openTest() ανοίγει το αρχείο test.csv και επιστρέφει τα περιεχόμενα του σε ένα πίνακα . Το αντίστοιχο κάνει και η openTrain() με το αρχείο train.csv . Η convertColorAndTypeInTrain() αλλάζει τα χρώματα και τα είδη των τεράτων σε διακριτές τιμές. Η convertColorInTest() μετατρέπει μόνο το χρώμα σε αριθμό . Ανοίγουμε τα αρχεία test.csv και train.csv και αποθηκεύουμε τα δεδομένα τους σε δυο πίνακες . Αρχικά η συνάρτηση

separate_by_class() χωρίζει το σύνολο των δεδομένων και επιστρέφει ένα λεξικό με διαχωρισμένα δεδομένα με βάση το type τους. Η stdev() υπολογίζει τη διακύμανση και η mean() το μέσο όρο που χρειάζονται στον συγκεκριμένο αλγόριθμο . Η summarize_dataset υπολογίζει το mean standard deviasion και το πλήθος στοιχείων για κάθε στήλη του συνόλου δεδομένων . Στη συνέχεια η summarize_by_class() χωρίζει το σύνολο δεδομένων με βάση το type και μετά υπολογίζει τα στατιστικά τους για κάθε γραμμή . Η calculate_probability_gauss επιστρέφει το αποτέλεσμα της κανονικής κατανομής . Η calculate_class_probabilities καλεί την calculate_probablity_gauss για κάθε γραμμή του κάθε type και υπολογίζει την πιθανότητα για το καθένα. Η convertToName μετατρέπει τον τύπου του τέρατος από float σε string . Η predict() προβλέπει τον τύπο τέρατος για μια γραμμή και η NaiveBayes() καλεί την predict() για κάθε γραμμή του αρχείου tests.csv. Τέλος η makeFinalFile() φτιάχνει τα predictions για το σωστό format στο kaggle.

Το αποτέλεσμα είναι το εξής :

Naive Bayes	Accuracy score	F1 score
	0.74102	

Σημειώσεις:

1) Σε καμία από τις 4 ασκήσεις δεν έχουμε βάλει το F1 score καθώς δεν ξέραμε πόσα TP TN FP και FN είχαμε αφού το Kaggle.com μας επέστρεφε μόνο του το accuracy score.