Μηχανική Μάθηση 1η Σειρά Ασκήσεων

Ομάδα 30

Μέλη ομάδας :

Α.Μ. 4161, Πουρσανίδης Κωνσταντίνος

Α.Μ. 4117, Μπαλής Κωνσταντίνος

Ξεκινώντας να αναφέρουμε ότι για την υλοποίησή μας χρησιμοποιήσαμε την βιβλιοθήκη Pandas για να διαχειριστούμε τα δεδομένα του training set. Επίσης δημιουργήσαμε μια συνάρτηση printer η οποία μας δημιουργεί τα αρχεία αποτελεσμάτων του test set στη μορφή που μας ζητάει το Kaggle. Για όλες τις μεθόδους αντικαταστήσαμε τις τιμές της $5^{ης}$ στήλης (color) σε συνεχείς τιμές ώστε να τις χρησιμοποιήσουμε κατά την υλοποίηση, όπως αναφέρθηκε και στο μάθημα.

Μέθοδοι Ταξινόμησης :

I. **k-NN Nearest Neighbor με Ευκλείδια απόσταση.**
   Για την υλοποίηση του k-nn χρησιμοποιήσαμε τον έτοιμο classifier της βιβλιοθήκης sk-learn. Στη συνέχεια αλλάξαμε την μεταβλητή n_neighbors στις τιμές που ζητήθηκαν (k = 1, 3, 5, 10) καθώς αυτή η μεταβλητή καθορίζει τους πόσους κοντινότερους γείτονες θα κοιτάξουμε. Εφόσον κάναμε fit τα δεδομένα για το train set στη συνέχεια χρησιμοποιήσαμε την έτοιμη συνάρτηση predict πάνω στο test set η οποία μας επέστρεψε ένα πίνακα με τα αποτελέσματα που βρήκε. Εφόσον ανεβάσαμε τα αποτελέσματα στο Kaggle βρήκαμε τα εξής αποτελέσματα :
   Για k = 1 : Score = 0.68241
   Για k = 3 : Score = 0.66918
   Για k = 5 : Score = 0.69376
   Για k = 10 : Score = 0.71455
   Αυτό που συμπεραίνουμε είναι όσο αυξάνουμε τους γείτονες τόσο καλύτερη είναι η πρόβλεψη, όπως και περιμέναμε από τη θεωρία. Για το k = 3 παρότι βλέπουμε μια μείωση πιστεύουμε πως πρόκειται για κάποια ισοβαθμία με αποτέλεσμα ο

αλγόριθμος να κάνει κάποια λάθος πρόβλεψη.

II. Neural Networks

Για τα νευρωνικά δίκτυα χρησιμοποιήσαμε τον έτοιμο MLPClassifier της βιβλιοθήκης sk-learn όπου σύμφωνα με την εκφώνηση αλλάξαμε τις παραμέτρους activation σε “tanh” ώστε να χρησιμοποιηθεί η σιγμοειδής συνάρτηση ως συνάρτηση ενεργοποίησης. Επίσης αλλάξαμε την μεταβλητή solver σε “sgd” ώστε να χρησιμοποιηθεί ως μέθοδος βελτιστοποίησης η stochastic gradient descent και τέλος θέσαμε τα max iterations σε 10000 γιατί εκεί είδαμε τα καλύτερα αποτελέσματα. Επίσης χρησιμοποιήσαμε την έτοιμη συνάρτηση *predict_proba()* ώστε να δοθούν τα αποτελέσματα ως πιθανότητα να ανήκει ένα δεδομένο σε μια κατηγορία (softmax).

(α) *με 1 κρυμμένο επίπεδο και Κ κρυμμένους νευρώνες*
Με 1 κρυμμένο επίπεδο τα αποτελέσματα που βρήκαμε είναι :
Για k = 50 : Score = 0.71455
Για k = 100 : Score = 0.70321
Για k = 200 : Score = 0.71077

(β) *με 2 κρυμμένα επίπεδα αποτελούμενο από Κ1 και Κ2 νευρώνες*
Με 2 κρυμμένα επίπεδα τα αποτελέσματα που βρήκαμε είναι :
Για k1 = 50, k2 = 25 : Score = 0.72400
Για k1 = 100, k2 = 50 : Score = 0.72022
Για k1 = 200, k2 = 100 : Score = 0.71644

Συμπεραίνοντας παρατηρούμε ότι τα καλύτερα αποτελέσματα δίνονται για μικρό αριθμό νευρώνων καθώς όσο αυξάνουμε τους νευρώνες παρατηρούμε χαμηλότερα αποτελέσματα που πιστεύουμε ότι οφείλονται στο φαινόμενο της υπερεκπαίδευσης.

III. Support Vector Machines (SVM).

Για την υλοποίηση των Support Vector Machines χρησιμοποιήσαμε τον έτοιμο classifier της βιβλιοθήκης sk-learn. Για την 1η περίπτωση αλλάξαμε την παράμετρο kernel σε “linear”

ώστε να χρησιμοποιηθεί η γραμμική συνάρτηση πυρήνα ενώ για την 2η αλλάξαμε την παράμετρο σε “rbf” για την Gaussian. Και στις 2 περιπτώσεις θέσαμε την παράμετρο *decision_function_shape* σε “ovr” ώστε να χρησιμοποιήσουμε την στρατηγική one-versus-all. Τα αποτελέσματα που βρήκαμε στην περίπτωση της Linear είναι 0.73345 ενώ για την Gaussian μετά από δοκιμές που κάναμε γύρο από την περιοχή του 0.2 (1/num_of_features, στην περίπτωσή μας 5 άρα 1/5=0.2) στην παράμετρο *gamma* (Kernel coefficient) 0.73534 για gamma = 0.3 και 0.4.

IV. Naïve Bayes.

- Αρχικά δουλεύοντας πάνω στο training set βρήκαμε για κάθε χαρακτηριστικό τη μέση τιμή και τη διακύμανση χρησιμοποιώντας έτοιμες συναρτήσεις της βιβλιοθήκης Pandas. Επίσης κάναμε το ίδιο και για κάθε κλάση. Στη συνέχεια δημιουργήσαμε ένα dictionary με κλειδιά του τύπους τεράτων και values ανά δυάδες για κάθε χαρακτηριστικό τη μέση τιμή και διακύμανση του. Έπειτα για κάθε αντικείμενο στο test set καλούμε την συνάρτηση *determine_class()* η οποία επιστρέφει ένα dictionary με κλειδιά τις κλάσεις των τεράτων και τιμές την πιθανότητα να ανήκει το εκάστοτε αντικείμενο του test set στην αντίστοιχη κλάση. Στη συνάρτηση *determine_class()* υπολογίζουμε την πιθανότητα ένα αντικείμενο του test set να ανήκει σε κάθε κλάση ξεχωριστά. Ο τρόπος που υπολογίζουμε αν ένα αντικείμενο ανήκει σε μια κλάση είναι με βάση τον τύπο **P(class|data) = P(X|class) * P(class)** όπου το P(class) υπολογίζεται διαιρώντας το πλήθος των εγγραφών της κλάσεις στο training set με τον συνολικό αριθμό εγγραφών και στη συνέχεια χρησιμοποιώντας την συνάρτηση *calculate_probability()* υπολογίζουμε την Gaussian κατανομή πιθανότητας με βάση την συνάρτηση **(1 / sqrt(2 * π) * σ) * exp(-((x-μ)^2 / (2 * σ^2)))** για κάθε χαρακτηριστικό ώστε να βρούμε το **P(Xi|class)**. Το αποτέλεσμα που βρήκαμε είναι 0.73534.

Τελικά για την καλύτερη μέθοδο ταξινόμησης προκύπτει ισοβαθμία μεταξύ της Naive Bayes και Gaussian SVM με gamma 0.3 ή 0.4.

Ευχαριστούμε για το χρόνο σας.