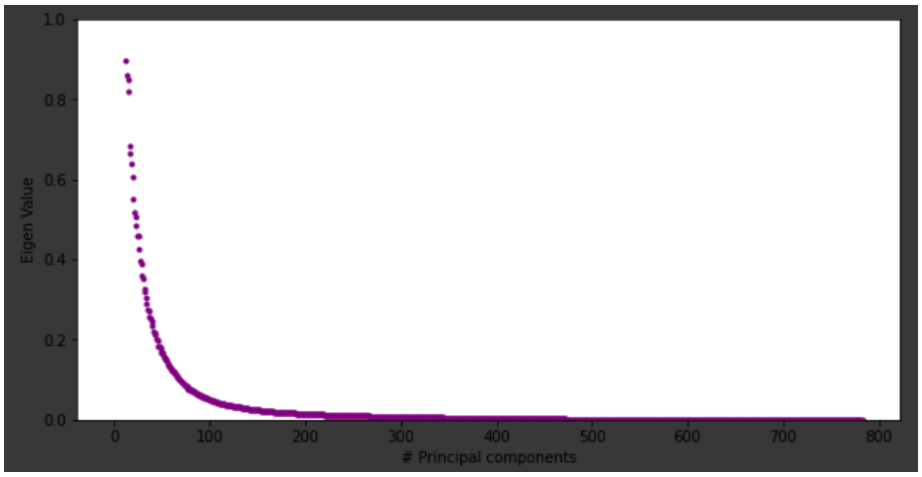
Νευρωνικά Δίκτυα

2η Εργασία

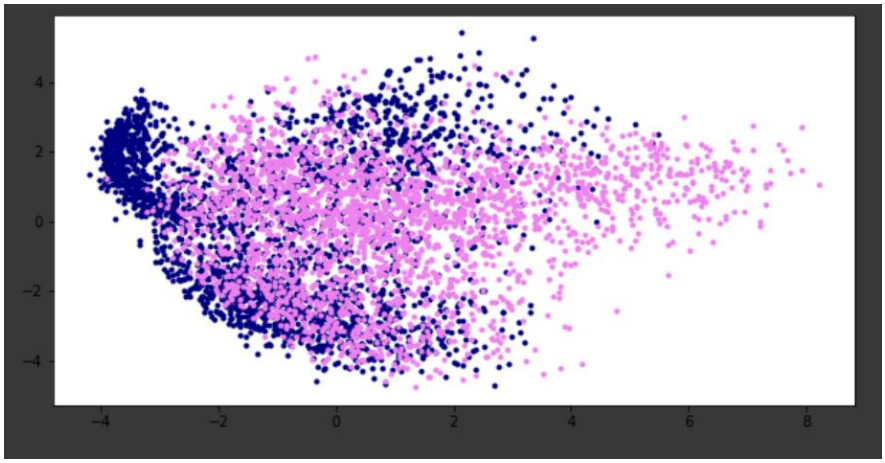
Στην εργασία χρησιμοποιείται η βάση δεδομένων MNIST, αλλά σε αυτήν την εργασία θα προσπαθήσουμε να χωρίσουμε τα δεδομένα μας σε 2 κλάσεις, δηλαδή σε ψηφία που είναι άρτια και σε ψηφία που είναι περιττά χρησιμοιώντας μια μηχανή διανυσμάτων υποστηρίξης. Έτσι αφού κατέβουν τα δεδομένα, γίνεται μία προ-επεξεργασία τους, μετατρέποντας τις ετικέτες των εικόνων σε τύπο δεδομένων int, και δημιουργώντας δύο κλάσεις, με ετικέτα 1 γα τα άρτια ψηφία και -1 για τα περιττά. Έπειτα κανονικοποιούνται τα δεδομένα διαιρώντας με 255, ώστε οι τιμές να είναι ανάμεσα στο 0 και το 1. Τέλος, αλλάζουμε το σχήμα των δεδομένων σε 60000, 784 για τα δεδομένα εκπαίδευσης και 10000, 784 για δεδομένα του τεστ.

Από τα 60000 δείγματα εκπαίδευσης κρατώ τα πρώτα 5000, και από αυτά του τεστ τα πρώτα 700, έτσι ώστε να μην έχω τόσο μεγάλο σετ δεδομένων. Ελέγχω αν είναι ίσα μοιρασμένα στις δύο κλάσεις διαπιστώνοντας πως για τα άρτια έχω 2465 δείγματα ενώ για τα περιττά 2535.

Στη συνέχεια εφαρμόζω σε αυτά ανάλυση κυρίων συνιστωσών κρατώντας 200 κύριες συνιστώσες, αφού όπως φαίνεται στο διάγραμμα παρακάτω η περισσότερη πληροφορία βρίσκεται στις πρώτες 200 από αυτές.



Παρακάτω φαίνεται το διάγραμμα διασποράς των σημείων των 2 πρώτων κύριων συνιστωσών.



Στη συνέχεια, τα δεδομένα παιρνιούνται τόσο στους k κοντινότερους γείτονες, όσο και στο κοντινότερο κέντρο, και τα αποτελέσματα είναι τα εξής:

K-NN για k=1: accuracy 95.14%, χρόνος εκτέλεσης 29 δευτερόλεπτα

K-NN για k=3: accuracy 95%, χρόνος εκτέλεσης 31 δευτερόπελτα

Nearest Centroid: accuracy 79.85%, χρόνος εκτέλεσης 0.03 δευτερόλεπτα

Παρατηρούμε πως ο αλγόριθμος κ κοντινότερων γειτόνων τα πηγαίνει καλύτερα από τον αλγόριθμο κοντινότερων κέντρων, και για τα 2 κ, ωστόσο είναι περισσότερο χρονοβόρος.

Για κ=1 ο αλγόριθμος παρουσιάζει ελάχιστα αυξημένη ακρίβεια από ότι για κ=3, και στις δύο περιπτώσεις υψηλή.

Περνώντας στη μηχανή διανυσμάτων υποστήριξης, υλοποιούνται 3 πυρήνες, ένας γραμμικός ένας πολυωνυμικός και ένας gaussian (rbf). Η κλάση SVM δέχεται σαν όρισμα την παράμετρο κανονικοποίησης C, η οποία αποτελεί trade off ανάμεσα στο να έχουμε μεγάλο περιθώριο και τη σωστή ταξινόμηση των εκπαιδευόμενων δεδομένων. Ουσιαστικά το C παίζει ρόλο στον αριθμό των λάθος ταξινομήσεων που επιτρέπονται. Αφού αποτελεί trade off, το πλάτος του περιθωρίου μικραίνει όσο η τιμή του C μεγαλώνει. Επίσης, αν δεν υπάρχει καθόλου C τότε έχουμε hard margin, δηλαδή δεν επιτρέπονται καθόλου λάθος ταξινομήσεις, αλλιώς έχουμε soft margin.

Έπειτα στην συνάρτηση fit της κλάσης, δίνονται τα δεδομένα και οι αντίστοιχες ετικέτες τους. Φτίαχνεται ο πίνακας του πυρήνα, ο οπόιος περνάει στις ανάλογες συναρτήσεις του πυρήνα τα δείγματα σειρά-σειρά, και έχει διάσταση NxN, όπου N ο αριθμός των δειγμάτων.

Μετά χρησιμοποιείται το πακέτο cvxopt της python για να λύσει το πρόβλημα τετραγωνικού προγραμματισμού που έχει δημιουργηθεί. Πιο αναλυτικά, θέλουμε να μεγιστοποιήσουμε το περιθώριο που θα χωρίζει τα δεδομένα μας, δηλαδή να ελαχιστοποιήσουμε τη δεύτερη νόρμα του διανύσματος των βαρών, με τον περιορισμό να μην υπάρχουν σημεία ανάμεσα. Το πρόβλημα αυτό μετατρέπεται σε διπλό πρόβλημα lagrange. Τα SVM μπορούν να λύσουν και μη γραμμικά προβλήματα κατηγοριοποίησης με τη χρήση των συναρτήσεων πυρήνα. Με αυτές, τα δείγματα των δεδομένων μεταφέρονται σε ένα χώρο μεγαλύτερης διάστασης, όπου μπορούν να ταξινομηθούν γραμμικά. Ουσιαστικά η συνάρτηση που θέλουμε να βελτιστοποιήσουμε είναι η $L_d = \sum a_i - \frac{1}{2}\sum a_i a_j y_i y_j (x_i \cdot x_j)$ όπου $(x_i \cdot x_j)$ είναι το

εσωτερικό γινόμενο των δύο διανυσμάτων χαρακτηριστικών. Αν μετατρέψουμε σε φ, που είναι μία συνάρτηση που αντιστοιχεί τα δεδομένα σε άλλο χώρο διαστάσεων, τότε πρέπει να υπολογίσουμε $(\varphi(x_i) \cdot \varphi(x_j))$ πράγμα αδύνατο. Χάρη στο κόλπο με τον πυρήνα έχουμε τη συνάρτηση πυρήνα k(x_i, x_j)= $(\varphi(x_i) \cdot \varphi(x_j))$, δεν χρειάζεται καν να υπολογίσουμε τη φ.

Όπως φαίνεται παραπάνω, αντί να ελαχιστοποιήσουμε προς w, b με περιορισμό που περιέχει lagrange πολλαπλασιαστές, μπορούμε να μεγιστοποιήσουμε ως προς αυτούς με περιορισμό τις σχέσεις που ίσχυαν για το w και το b. Έτσι ανεξαρτητοποιούμαστε και από τα δύο αυτά. Αφού βρεθεί η λύση, παίρνουμε τους πολλαπλασιαστές lagrange που είναι μεγαλύτεροι από το 0, και αυτοί αντιστοιχούν στα διανύσματα υποστήριξης. Έτσι η διάσταση της λύσης έχει μειωθεί. Έχοντας τώρα τους πολλαπλαστιαστές, η συνάρτηση απόφασης θα είναι:

$$f(x) = sign(\sum_{i,j=1}^{M} a_i y_i \, k(x_i, x_j) + b)$$

Οπότε αφού στη συνάρτηση fit υπολογίζεται το bias, στη συνάρτηση predict δίνονται τα τεστ δεδομένα, έτσι ώστε βρίσκοντας το πρόσημο της παραπάνω συνάρτησης, να βρεθεί η κλάση κάθε αγνώστου τεστ δείγματος.

Δοκιμάζονται λοιπόν πολλές διαφορετικές περιπτώσεις, για διαφορετικούς πυρήνες, διαφορετικά C, και διαφορετικές παραμέτρους του πολυωνυμικού και gaussian πυρήνα.

Αρχικά για γραμμικό πυρήνα και για μικρή τιμή c=0.01, η ακρίβεια στην εκπαίδευση είναι 90% ενώ για τον έλεγχο είναι 88.8%. Ο χρόνος εκτέλεσης της μηχανής είναι 4.1 λεπτά.

Για γραμμικό πυρήνα και για τιμή c=0.1, η ακρίβεια στην εκπαίδευση είναι 91.4% ενώ για τον έλεγχο είναι 89%. Ο χρόνος εκτέλεσης της μηχανής είναι 4 λεπτά. Φαίνεται πως με την αύξηση του c, αυξήθηκε και λίγο η ακρίβεια.

Έπειτα, για πολυωνυικό πυρήνα 2^{ου} βαθμού και τιμή c=1, η ακρίβεια στην εκπαίδευση είναι 100% ενώ για τον έλεγχο είναι 95.7%. Ο χρόνος εκτέλεσης της μηχανής είναι 4.3 λεπτά.

Για πολυωνυικό πυρήνα 3^{ου} βαθμού και τιμή c=1, η ακρίβεια στην εκπαίδευση είναι 99.9% ενώ για τον έλεγχο είναι 97.1%. Ο χρόνος εκτέλεσης της μηχανής είναι 4.3 λεπτά περίπου.

Για τον gaussian πυρήνα με παράμετρο γάμμα ίση με 0.02. Η παράμετρος αυτή είναι κάτι σαν την εξάπλωση του πυρήνα και δηλαδή της περιοχής απόφασης. Όταν είναι μικρή, τότε η περιοχή απόφασης είναι ευρεία. Αντίθετα, όταν η παράμετρος είναι υψηλή, τότε μετατρέπονται κάποια όρια απόφασης γύρω από τα σημεία δεδομένων. Η ακρίβεια στην εκπαίδευση είναι 100% ενώ στον έλεγχο 98.2%. Ο χρόνος εκτέλεσης αντιστοιχεί σε 6.5 λεπτά.

Αν αυξήσουμε το γάμμα σε 2, χωρίς τη χρήση C, τότε η ακρίβεια πέφτει αρκετά και ισούται με 52.14%. Πολλές φορές με την αύξηση του γάμμα, πετυχαίνεται υπερεκπαίδευση των δεδομένων.

Εκτός από την υλοποίηση του svm, δοκιμάζω και κάποιες έτοιμες βιβλιοθήκες για SVM, όπως το libsvm και το SVC του sklearn για να συγκρίνω τα αποτελέσματα.

Όσον αφορά το libsvm, δοκιμάζοντας να κατηγοριοποιήσουμε τα ψηφία μας με τη χρήση πολυωνυμικού πυρήνα 3^{ου} βαθμού και c=10 τα αποτελέσματα είναι για την εκπαίδευση 77.1% και για τον έλεγχο 73.14%, με χρόνο εκτέλεσης 33 δευτερόλεπτα, αρκετά γρηγορότερα.

Για γραμμικό πυρήνα με c=10 τα αποτελέσματα είναι για την εκπαίδευση 94.1% και για τον έλεγχο 87.5%, με χρόνο εκτέλεσης 31 δευτερόλεπτα. Αν αλλάξουμε σε c =1, τότε η ακρίβεια πέφτει σε 93.5% και παραμένει σχεδόν ίδια για τον έλεγχο.

Για gaussian πυρήνα με c=1 τα αποτελέσματα είναι 90.4% ακρίβεια στην εκπαίδευση και 89.14% στον έλεγχο.

Δοκιμάζουμε και το sklearn, πρώτα για γραμμικό πυρήνα με c=0.1, το αποτέλεσμα είναι 92.3% στην εκπαίδευση και 88.7% στον έλεγχο σε μόλις 17 δευτερόλεπτα.

Για gaussian πυρήνα με γάμμα=2 και c=10 η ακρίβεια της εκπαίδευσης είναι 100% ενώ του ελέγχου 52% παρόμοια με την υλοποίηση του svm που περιγράφηκε πάνω.

Τέλος για πολυωνυμικό πυρήνα $2^{ου}$ βαθμού με c=1 έχουμε 94.2% για την εκπαίδευση και 92.5% για τον έλεγχο σε 25 δευτερόλεπτα. Για τον ίδιο, βάζοντας το C=100, αυξάνονται οι επιδόσεις σε 100% και 95.8% σε 10 δευτερόλεπτα. Αυτό εξηγείται καθώς όσο μεγαλύτερο c, τόσο μεγαλύτερη ακρίβεια.

Σε σχέση με τους ταξινομητές που περιγράφτηκαν πριν, δηλαδή τους κοντινότερους γείτονες και τα κοντινότερα κέντρα κλάσης, η απόδοση σε γενικές γραμμές κρίνεται παρόμοια, αν και κάποιες φορές είναι μεγαλύτερη στα svm, εκτός από συγκεκριμένες τιμές παραμέτρων που επιλέχθηκαν οι οποίες μείωσαν την επίδοση τους. Ωστόσο η υλοποιημένη svm μειονεκτεί σε χρόνο εκτέλεσης τόσο σε σχέση με τους ταξινομητές, αλλά και με το svm του sklearn.

Ο κώδικας βρίσκεται στο link:

https://colab.research.google.com/drive/1ZfxpDCyqPFiOlBGhp73eADnl23_IHpwn#scrollTo=kQ9rNbMW0oHO

Κώστογλου Σοφία 3114