

Νευρωνικά Δίκτυα (Πρώτο Παραδοτέο)

PaRT 1. SVM Classifiers

Καλλιμάνης Ιωάννη | 10007 | Ηλεκτρολόγων μηχανικών και μηχανικών ηλεκτρονικών υπολογιστών | [ikallima@ece.auth.gr](mailto:ikallima@ece.auth.gr) | 26-11-2023

Εισαγωγή

Σαν συνέχεια της προηγούμενης προσπάθεια μας να «εκπαιδεύσουμε» τον υπολογιστή να αναγνωρίζει μια εικόνα σε ποια κατηγορία ανήκει από της 10 κατηγορίες της βάσης της CIFAR10, ζητήθηκε να εξετάσουμε τα αποτελέσματα που δίνουν η κατηγορία των Support Vector Machines (SVM) στην βάση μας.

Τα SVM αποτελούν μοντέλα εποπτευόμενης μάθησης που έχουν ως στόχο να μεγιστοποιήσουν το περιθώριο margin μεταξύ των κοντινότερων δειγμάτων των κλάσεων, ώστε ακόμα και αν κάποια από τα δείγματα εισαχθούν με κάποιον θόρυβο στο μοντέλο μας να μην ξεπεράσουν την διαχωριστική γραμμή των κλάσεων και να ταξινομηθούν σωστά. Σημαντικό τους προτέρημα είναι ότι δεν χρειάζονται στην εκπαίδευση όλα τα δείγματα του dataset, παρά μόνο αυτά που επηρεάζουν το margin, δηλαδή αυτά που βρίσκονται πάνω στο περιθώριο, αυτά που βρίσκονται μέσα καθώς και αυτά που ταξινομήθηκαν λανθασμένα. Έτσι εκπαιδεύονται μοντέλα που χρησιμοποιούν στην υλοποίηση σαφώς μικρότερο αριθμό παραδειγμάτων από ότι σε κάποιον MLP γεγονός που τις περισσότερες φορές αντικατοπτρίζεται στην ταχύτητα εκτέλεσης του κώδικα τους.

Ο τρόπος που τα SVM διαχωρίζουν τα δείγματα στις σωστές κλάσεις όταν αυτά δεν είναι γραμμικώς διαχωρίσιμα στο επίπεδο που βρίσκονται είναι μέσω των συναρτήσεων πυρήνα (kernel). Με αυτόν τον τρόπο τα δείγματα μας ανάγονται σε ένα υπερεπίπεδο μεγαλύτερης διάστασης, στο οποίο είναι γραμμικώς διαχωρίσιμα και μεγιστοποιούν το περιθώριο margin.

Περιγραφή κώδικα

Ξεκινώντας την ενασχόληση για την υλοποίηση ενός μοντέλου SVM, μέσω της βιβλιοθήκης sclearn έκανα import την συνάρτηση SVC ώστε να αρχίσω να ελέγχω ποιες πρέπει να είναι οι τιμές των υπερπαραμέτρων που πρέπει να χρησιμοποιήσω ώστε να λάβω την καλύτερη απόδοση.

Στο ξεκίνημα, λοιπόν, της προσπάθειας ώστε να εντοπίσω της καλύτερες υπερπαραμέτρους που μπορώ να χρησιμοποιήσω δεν υλοποίησα καθόλου την αλλαγή των διαστάσεων των δειγμάτων μου μέσω της PCA. Τα αποτελέσματα που έλαβα ακόμα και στην καλύτερη περίπτωση δεν θα τα χαρακτήριζα και ιδανικά. Οι αποδόσεις σε πολλά από τα μοντέλα κυμαίνονταν μεταξύ 20-30% τόσο στο train set όσο και στο test set, πράγμα που από την μία δείχνει μια προσπάθεια εκπαίδευσης καθώς οι ετικέτες εκτίμησης δεν είναι και τόσο τυχαίες (μεγαλύτερες του 10%) αλλά από την άλλη δεν είναι και τόσο υψηλές. Κάποια παραδείγματα είναι:

1). model = SVC(kernel='linear', random\_state=42)

Training Accuracy: 0.2202

Testing Accuracy: 0.2144

2). model = SVC(C=1000, kernel='linear', random\_state=42)

Training Accuracy: 0.22484

Testing Accuracy: 0.2237

3). model = SVC(C=10, kernel='poly', degree=2, random\_state=42)

Training Accuracy: 0.22916

Testing Accuracy: 0.2295

4). model = SVC(C=0.1, kernel='poly', degree=2, random\_state=42) (13)

Training Accuracy: 0.1636

Testing Accuracy: 0.164

5). model = SVC(C=1000, kernel='rbf', random\_state=42, gamma='auto', decision\_function\_shape='ovo')

Training Accuracy: 0.205

Testing Accuracy: 0.2023

6). model = SVC(C=1000, kernel='sigmoid', random\_state=42, gamma='auto')

Training Accuracy: 0.2096

Testing Accuracy: 0.2207

Συνεχίζοντας λοιπόν την αναζήτηση μου για την αναβάθμιση των αποτελεσμάτων μου, ο καθηγητής στο μάθημα του αναφέρθηκε στην έννοια του Principal Component Analysis (PCA), δηλαδή σε έναν τρόπο μείωσης της διάστασης των δειγμάτων μου με τέτοιον τρόπο ώστε τα δείγματα μου να έχουν στην έξοδο ικανοποιητικό ποσοστό της αρχικής μου πληροφορίας, δηλαδή της «πιθανότητας» να λάβω εκ νέου από τα καινούργια μου δείγματα με την μικρότερη πληροφορία την πληροφορία που είχα και προηγουμένως. Έτσι, λοιπόν, δοκίμασα με την μέθοδο της PCA από την sklearn.decomposition να δω ποιο είναι το ποσοστό της πληροφορίας που λαμβάνω από τα νέα δείγματα με βάση την καινούργια τους διάσταση. Με τον τρόπο αυτόν έριξα την προσοχή μου σε δύο τιμές:

100 : 0.90106225

250 : 0.95736456

Μόνο με 100 νέα χαρακτηριστικά από τα συνολικά 3072 μπορώ να λάβω το 90% της πληροφορίας που είχα προηγουμένως. Έτσι έκανα μετατροπή των δεδομένων μου με καινούργια διάσταση 100% και δοκίμασα εκ νέου κάποια από τα παραδείγματα που είδα στην αρχή με την διαφορά να είναι πλέον κάπως αισθητή, όπως:

7). model = SVC(C=1000, kernel='rbf', random\_state=42, gamma='auto') (παρόμοιο με το 5).)

Train Accuracy: 29.9040%

Test Accuracy: 29.1000%

8). model = SVC(C=100, kernel='rbf', random\_state=42)

Train Accuracy: 99.6820%

Test Accuracy: 54.0400%

(Εδώ αν και τα αποτελέσματα είναι εντυπωσιακά για το training accuracy παρατηρείται μεγάλο overfit των δεδομένων στα δεδομένα εκπαίδευσης)

Γρήγορα όμως σε συζητήσεις με τους συμφοιτητές μου και με το εργαστήριο που ακολούθησε διαπίστωσα ότι εκπαίδευση δεν χρειάζεται να γίνεται σε ολόκληρο το training dataset, αλλά μπορεί κάλλιστα να δοκιμαστεί και σε ένα υποσύνολο του. Έτσι δημιούργησα ένα νέο dataset εκπαίδευσης με ίδια ποσοστά των δειγμάτων της κάθε κλάσης (10% για το καθένα) απλά αντί για συνολικά 5000 παραδείγματα για την κάθε κλάση είχα μονάχα 1000. Με τον τρόπο αυτό αυξήθηκε κατά πολύ η ταχύτητα του μοντέλου μου, ρίχνοντας παράλληλα και τις αποδόσεις. Σχετικά παραδείγματα:

9). model = SVC(C=100, kernel='rbf', random\_state=42)

Train Accuracy: 99.6820%

Test Accuracy: 54.0400%

Με την χρήση ενός mini-batch εκπαίδευσης

Train Accuracy: 56.4840%

Test Accuracy: 45.3600%

10). model = SVC(C=10, kernel='rbf', random\_state=42)

Train Accuracy: 92.2300%

Test Accuracy: 56.1700%

Train Accuracy: 56.9040%

Test Accuracy: 46.8400%

11). model = SVC(C=1, kernel='rbf', random\_state=42)

Train Accuracy: 65.4700%

Test Accuracy: 54.0000%

Train Accuracy: 50.3760%

Test Accuracy: 47.2600%

Παρόλα αυτά, προτίμησα να ακολουθήσω την λογική του mini-batch καθώς παρατηρώ μικρότερες διαφορές μεταξύ train και test accuracy ώστε εν τέλη να καταλήξω στο ιδανικό για την περίπτωση μου. Έτσι δοκίμαζα τυχαία μοντέλα με διαφορετικές παραμέτρους το καθένα όπως:

12). SVC(C=10, kernel='linear', random\_state=42)

Train Accuracy: 39.3980%

Test Accuracy: 37.8300%

(\*Σε αυτό το σημείο θα ήταν καλό να παραθέσω ένα σχόλιο για τα μοντέλα με kernel=’linear’ στην CIFAR10 τουλάχιστον. Τα μοντέλα αυτά είναι πιο αργά σε σχέση με τα υπόλοιπα. Ειδικά σε περιπτώσεις όπου C>1 να καθυστερούν εκπληκτικά πολύ τα αποτελέσματα τους, τα οποία δεν είναι και τόσο ικανοποιητικά όπως δηλώνει το παραπάνω παράδειγμα. Για C=100 το πρόγραμμα έκανα παραπάνω από 8 ώρες να δώσει αποτέλεσμα, που εν τέλη δεν έλαβα καθώς αποφάσισα να διακόψω το πρόγραμμα για να δοκιμάσω κάτι άλλο ίσως και καλύτερο)

13). model = SVC(kernel='poly', degree=6, random\_state=42)

Train Accuracy: 29.8480%

Test Accuracy: 23.8500%

14). model = SVC(kernel='poly', degree=2, random\_state=42)

Train Accuracy: 41.7680%

Test Accuracy: 39.5500%

15). model = SVC(kernel='poly', degree=2, random\_state=42, coef0=0.5)

Train Accuracy: 49.4100%

Test Accuracy: 46.9000%

16). model = SVC(C=10, kernel='poly', degree=2, random\_state=42, coef0=0.5)

Train Accuracy: 54.1440%

Test Accuracy: 46.5900%

17). model = SVC(C=10, kernel='poly', degree=2, random\_state=42, gamma=0.5, coef0=0.5, probability=True)

Train Accuracy: 53.0180%

Test Accuracy: 41.2300%

18). model = SVC(C=10, kernel='rbf', random\_state=42)

Train Accuracy: 57.0920%

Test Accuracy: 47.2100%

Τελικά άλλαξα λίγο τον κώδικα μου, στην PCA να παίρνω το 90% της πληροφορίας λαμβάνοντας ως καινούργια διάσταση των δειγμάτων μου 99 χαρακτηριστικά και μέσω GridSearchCV να καταλήξω στις καλύτερες υπερπαραμέτρους του μοντέλου που θα παραδώσω.

Τελικό Μοντέλο

Στο τελικό μοντέλο είναι προσαρμοσμένα 2 μοντέλα. Ένα rbf και ένα πολυωνυμικό.

1). Για το rbf η τιμές των υπερπαραμέτρων που χρησιμοποιήθηκαν είναι: C=10, decision\_function\_shape=’ovo’ και ενώ δοκιμάστηκε και η probability=True καθώς αποτελούσε ιδανική παράμετρο σύμφωνα με το GridSearch δεν προτιμήθηκε στο τελικό μοντέλο καθώς καθυστερούσε το τελικό μοντέλο, ειδικά στην περίπτωση που το SVM εκπαιδευόταν σε ολόκληρο το dataset εκπαίδευσης (χρειαζόταν πάνω από 10000 επαναλήψεις για να δώσει αποτέλεσμα). Τα αποτελέσματα που έλαβα είναι:

-Για τυχαία 5000 παραδείγματα εκπαίδευσης:

Training process (5000 the size of the mini batch): 2.770853281021118

Mini-batch training set Accuracy: 97.4800%

Whole training set Accuracy: 49.1980%

Test Accuracy: 44.2200%

-Για τυχαία 10000 παραδείγματα εκπαίδευσης:

Training process (10000 the size of the mini batch): 10.014667510986328

Mini-batch training set Accuracy: 96.2700%

Whole training set Accuracy: 56.9020%

Test Accuracy with tolerance: 47.8200%

-Και συνολικά για την εκπαίδευση σε ολόκληρο το dataset εκπαίδευσης

Training process: 423.3248574733734

Training set Accuracy: 92.1620%

Test Accuracy : 56.2400%

2). Έναν πολυωνυμικό μοντέλο με C=1, coef0=0.5, degree=3, tol=0.01, decision\_function\_shape=’ovo’ είναι καλό να εξεταστεί καθώς μας παρέχει παρόμοια αποτελέσματα με το προηγούμενο rbf με ακρετά μικρότερο χρόνο εκπαίδευσης(4.8 λεπτα έναντι 7,05). (Αξίζει να σημειωθεί ότι ακόμα ένα πολυωνιμικό είναι κοντά σε αυτές τις τιμές, ελάχιστα πιο κάτω από το προηγούμενο με C=10, coef=0.1, degree=2, decision\_function\_shape=’ovo’)

-Για τυχαία 5000 παραδείγματα εκπαίδευσης:

Training process (5000 the size of the mini batch): 1.5385587215423584

Mini-batch training set Accuracy: 82.9400%

Whole training set Accuracy: 47.4580%

Test Accuracy: 43.6900%

-Για τυχαία 10000 παραδείγματα εκπαίδευσης:

Training process (10000 the size of the mini batch): 5.8571343421936035

Mini-batch training set Accuracy: 80.6100%

Whole training set Accuracy: 53.4740%

Test Accuracy: 47.1300%

-Και συνολικά για την εκπαίδευση σε ολόκληρο το dataset εκπαίδευσης

Training process: 293.58865761756897

Training set Accuracy: 74.9660%

Test Accuracy : 54.8500%

Συμπεράσματα

Συγκριτικά, λοιπόν, με το MLP που υλοποίησα στο προηγούμενο παραδοτέο παρατηρώ ότι έχω από την μια χειρότερα αποτελέσματα στην απόδοση του test set (της τάξης του 10% max) αλλά τα αποτελέσματα τα λαμβάνω πάρα πολλές φορές πιο γρήγορα. Προτιμότερο δηλαδή από τον MLP from scratch, όχι όμως από ένα CNN (το υλοποίησα από περιέργεια στο ενδιάμεσο διάστημα των δύο εργασιών) όπου τα αποτελέσματα του ήταν τρομακτικά καλά και δεν θεωρώ την χρονική διαφορά τόσο σημαντική για τέτοια αποτελέσματα (Train Accuracy: 99.808%, Testing Accuracy: 97.38%).

Παράμετροι που ελέγχθηκαν

**C**: παράμετρος ρύθμισης

Ρυθμίζει μια παράμετρο «πειθαρχίας» του μοντέλου. Μεγάλες τιμές του C οδηγούν σε μικρότερο περιθώριο margin αλλά με μεγαλύτερη ακρίβεια στα δεδομένα εκπαίδευσης.

**gamma**: συντελεστής πυρήνα (kernel)

Ορίζει πόσο σημαντική είναι η επίδραση ενός παραδείγματος εκπαίδευσης.

**coef0**: Ανεξάρτητος όρος της συνάρτησης πυρήνα

Κατέχει ρόλο μόνο στις συναρτήσεις πυρήνα τύπου ‘poly’ και ‘sigmoid’.

**degree**: βαθμός πολυωνύμου

Κατέχει ρόλο μόνο στην συνάρτηση πυρήνα τύπου poly και ορίζει το βαθμό της συνάρτησης αυτής.

**decision\_function\_shape**: Σχήμα συνάρτησης απόφασης

Χρησιμοποιείται για την επιλογή στρατηγικής σε πρόβλημα με πολλές κλάσεις. Η "ovo" χρησιμοποιεί έναν δυαδικό ταξινομητή για κάθε ζευγάρι κλάσεων, ενώ η "ovr" χρησιμοποιεί έναν δυαδικό ταξινομητή για κάθε κλάση.

**tol**: Απόλυτη τιμή για το κριτήτιο εξόδου

Η τιμή αυτή χρησιμοποιείται ως κριτήριο για την τέλος της εκπαίδευσης όταν η μεταβολή της συνάρτησης στόχου είναι μικρότερη από αυτή την τιμή.

**probability**: Εκτίμηση Πιθανοτήτων

Εάν οριστεί ως True το μοντέλο θα ενεργοποιήσει την εκτίμηση πιθανοτήτων. Αυτό μπορεί να επιβραδύνει τη διαδικασία εκπαίδευσης αλλά επιτρέπει πρόβλεψη πιθανοτήτων

**max\_iter**: Μέγιστος αριθμός επαναλήψεων

Αν δεν τερματίσει ο κώδικας από το κριτήριο που αναφέρθηκε παραπάνω θα τερματίσει όταν φθάσει τον μέγιστο αριθμό επαναλήψεων.

Σημείωση

Επιπλέον να σημειώσω ότι το Train Accuracy που αναφέρω παραπάνω, εκτός από την ενότητα -Τελικό Μοντέλο- αναφέρεται στο συνολικό training set και όχι σε αυτό το mini batch που εκπαιδεύτηκε το εκάστοτε μοντέλο