ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

ΒΑΘΙΑ ΜΑΘΗΣΗ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Πορλού Χάιδω

ΣΧΟΛΗ: ΗΜΜΥ ΑΠΘ

ΑΕΜ: 9372

ΕΡΓΑΣΙΑ 2

SUPPORT VECTOR MACHINES (SVM)

Εισαγωγή

Η 2η εργασία που μου ανατέθηκε στο μάθημα Νευρωνικά Δίκτυα – Βαθιά Μάθηση αφορά ένα πρόβλημα ταξινόμησης (classification) 2 (κλασσικά) ή περισσοτέρων κλάσεων με τη χρήση ενός Support Vector Machine (SVM) . Όσον αφορά την εξαγωγή χαρακτηριστικών, επιλέχθηκε η χρήση όλης της εισόδου (ολόκληρης της 32x32 εικόνας), μιας και δεν υπήρχαν προβλήματα υπολογιστικής ισχύος και η μέθοδος χρησιμοποιήθηκε ήδη στην προηγούμενη εργασία. Σημειώνεται επίσης πως επήλθε κανονικοποίηση των στοιχείων στο διάστημα [0, 1].

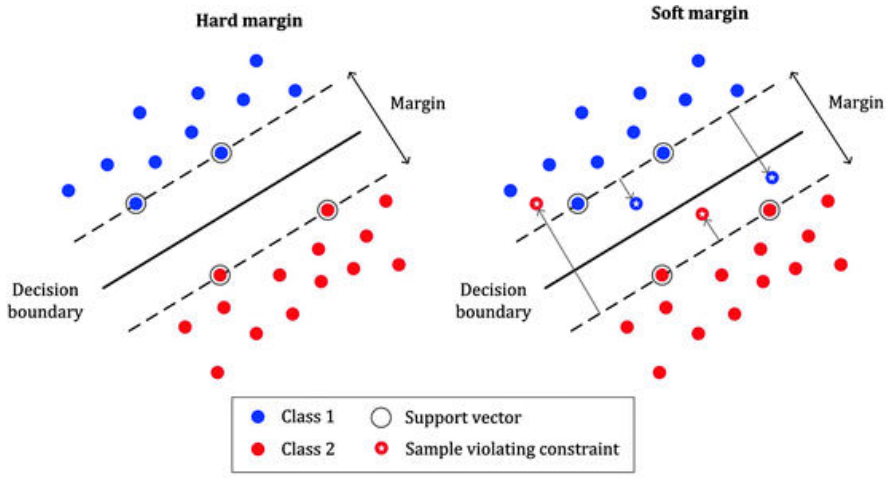
Σαν dataset επιλέχθηκε το cifar-10 dataset, το οποίο περιλαμβάνει 60,000 32x32 εικόνες που μπορούν να κατηγοριοποιηθούν σε 10 κλάσεις: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Ο αρχικός μου στόχος ήταν η κατηγοριοποίηση/διαχωρισμός 2 κλάσεων του dataset (συγκεκριμένα “dog” vs “horse”), η μελέτη όμως συνεχίστηκε με προσπάθεια κατηγοριοποίησης όλων των κλάσεων.

Το dataset είναι ήδη χωρισμένο σε training και testing υποσύνολα, με 50,000 και 10,000 στοιχεία αντίστοιχα. Στο πρόβλημα των 2 τάξεων που προαναφέρθηκε, ο αριθμός των training και testing δεδομένων-εικόνων είναι ίσος με 10000 και 2000 αντίστοιχα (μισά για τη μία κλάση και μισά για την άλλη). Αυτό βοηθάει πάρα πολύ στην υπολογιστική διαδικασία, καθώς κάποια από τα SVM που μελετήθηκαν χρειάσθηκαν πολύ χρόνο για να εκπαιδευθούν. Όσον αφορά το κομμάτι που πραγματεύεται το πρόβλημα ταξινόμησης όλων των κλάσεων, επιλέχθηκαν 10000 τυχαία στοιχεία για το training και 2000 τυχαία για το testing, για λόγους -και πάλι- χρόνου.

Κατηγοριοποίηση 2 κλάσεων (dog & horse) με 3 διαφορετικά μοντέλα SVM

- Γραμμικό (linear) SVM -

Το πρώτο SVM που επέλεξα να μελετήσω -και αδιαμφισβήτητα το απλούστερο- είναι το **γραμμικό (linear)**. Περιληπτικά, για την κατηγοριοποίηση ενός στοιχείου στη μία ή την άλλη κλάση σε ένα εκπαιδευμένο SVM χρησιμοποιείται μία γραμμική συνάρτηση (ευθεία), στα δεξιά και αριστερά της οποίας υπάρχει ένα περιθώριο που ορίζεται από 2 ή περισσότερα στοιχεία που ονομάζονται support vector machines (βλ. εικόνα). Στη δική μας περίπτωση χρησιμοποιείται ένα soft-margin linear SVM, το οποίο σημαίνει πως τα support vectors είναι περισσότερα από 2, διότι οι κλάσεις μας δεν είναι ξεκάθαρα γραμμικά διαχωρίσιμες.



Στην μελέτη χρησιμοποιήθηκε (από την βιβλιοθήκη sklearn) η συνάρτηση SVC(), αν και δοκιμάσθηκε και η linearSVC(), η οποία δεν χρησιμοποιήθηκε τελικά λόγω μεγάλης αύξησης στο χρόνο εκπαίδευσης. Στο παρακάτω πινακάκι παρουσιάζονται τα αποτελέσματα για την ακρίβεια στο testing dataset, καθώς και ο χρόνος της κάθε εκτέλεσης, για διαφορετικά C.

C	Testing Accuracy	Time (s)	Library
0.001	0.7470	168.84579181671143	SVC
0.01	0.7520	145.4804117679596	SVC
0.1	0.7515	144.38744139671326	SVC
1	0.7305	161.20866990089417	SVC
10	0.7045	821.6471834182739	SVC

Το C, η γνωστή παράμετρος σφάλματος, δείχνει στο εκπαιδευόμενο SVM πόσο πολύ πρέπει να αποφευχθεί το misclassification των δειγμάτων. Ένα μεγάλο C δίνει ένα μικρότερο margin αν αυτό κατηγοριοποιεί σωστότερα τα δεδομένα, ενώ ένα μικρότερο C μπορεί να δώσει ένα μεγαλύτερο margin, ακόμα κι αν αυτό δεν είναι το “βέλτιστο”. Για το λόγο αυτό, για παράδειγμα, στο πείραμα μας με C = 1 έχουμε 4951 support vectors, ενώ με C = 10 έχουμε 4679 support vectors, το οποίο μεταφράζεται σε μεγαλύτερο και μικρότερο margin αντίστοιχα (περισσότερα support vectors/στοιχεία εντός του margin = μεγαλύτερο margin).

Σχολιάζεται εδώ πως με τη βιβλιοθήκη LinearSVC() το SVM δεν μπορούσε να συγκλίνει για C > 0.1 (με τον standard αριθμό από iterations = 1000), και η αύξηση του αριθμού των iterations οδηγούσε σε μεγάλους χρόνους, και αυτός είναι ο λόγος που τα αποτελέσματα που παρουσιάζονται είναι τελικά μόνο από τη βιβλιοθήκη SVC(), χάριν καλύτερης σύγκρισης.

Καλύτερο accuracy στην ταξινόμηση των 2 κλάσεων φαίνεται να έχουμε όταν C = 0.01, κάτι ιδιαίτερα θετικό, καθώς ο χρόνος εκπαίδευσης του SVM που το συνοδεύει είναι αρκετά καλός και δεν αυξάνει πολύ όπως με C = 10. Φαίνεται λοιπόν εδώ πως το μικρότερο C λύνει καλύτερα το πρόβλημα (με μεγαλύτερο margin), το οποίο είναι σχετικά λογικό, καθώς ξέρουμε από την αρχή πως το πρόβλημα δεν είναι γραμμικά διαχωρίσιμο, οπότε θα έχουμε misclassification όπως και να έχει (εννοώντας πως το μικρότερο margin πιθανότατα να μην επηρεάζει τόσο το πρόβλημα μας).

Μιας και τα δεδομένα μας είναι ξεκάθαρα μη γραμμικά διαχωρίσιμα, το επόμενο λογικό βήμα είναι να προχωρήσουμε στη χρήση συναρτήσεων πυρήνα (kernels), οι οποίες είναι μη γραμμικές. Μιλάμε δηλαδή για έναν μη γραμμικό μετασχηματισμό, μετά τη χρήση του οποίου τα δεδομένα μας θα είναι πια γραμμικά διαχωρίσιμα, με αποτέλεσμα να μπορούμε να κάνουμε καλύτερη κατηγοριοποίηση τους.

- **Πολυωνυμικό (polynomial) SVM** -

Το δεύτερο μοντέλο που μελετήθηκε αφορά ένα μη γραμμικό SVM, και συγκεκριμένα το **πολυωνυμικό (polynomial)**. Οι παράμετροι που άλλαξαν τιμές στα πειράματα είναι:

- Το C (βάρος του κόστους των λαθών ταξινόμησης, όπως πριν)
- Ο βαθμός του πολυωνύμου
- Το gamma (με default gamma = 'scale')

Στους παρακάτω πίνακες φαίνονται όλες οι αλλαγές των παραπάνω, μαζί με τις τιμές του testing accuracy και του χρόνου κάθε εκπαίδευσης.

C	Degree	Gamma	Accuracy	Time (s)
0.0001	2	scale	0.6365	130.74317312240
0.001	2	scale	0.6800	115.31123757362
0.01	2	scale	0.7520	98.432552814483
0.1	2	scale	0.8000	79.721715927124
1	2	scale	0.8235	72.171987056732
10	2	scale	0.8030	91.293151855468
100	2	scale	0.7865	96.262943983078

C	Degree	Gamma	Accuracy	Time (s)
0.1	2	0.001	0.7270	108.10703277587
1	2	0.001	0.7790	88.581116676330
0.1	2	0.01	0.8205	74.692308187484
1	2	0.01	0.8190	77.535934686660
0.1	2	0.1	0.7955	97.463131904602
1	2	0.1	0.7865	96.734083890914
0.1	2	1	0.7865	96.127807617187
1	2	1	0.7865	94.743685722351
0.1	2	10	0.7865	97.329569101333
1	2	10	0.7865	95.718275308609

C	Degree	Gamma	Accuracy	Time (s)
0.1	3	scale	0.8290	73.194113492965
1	3	scale	0.8165	74.060700654983
0.1	4	scale	0.8230	74.356865882873
1	4	scale	0.8165	73.230432510375

Αρχικά, αξίζει να σημειωθεί πως επιλέχθηκε να μελετηθεί κυρίως το kernel με διωνυμική συμπεριφορά χάριν απλότητας και καλύτερης κατανόησης της μορφής του. Η πρώτη φάση πειραμάτων έγινε με επιλογή του gamma ως 'scale'. Το gamma σαν ορισμός δείχνει την μεγαλύτερη ή μικρότερη "επιρροή" ενός στοιχείου στη διαδικασία του training (με μικρό gamma να δείχνει μεγάλη επιρροή και αντίστοιχα μεγάλο gamma, μικρότερη). Η επιλογή της τιμής του ως scale (τιμή ανάλογη του training dataset -> {1 / (n_features * X.var)}) είναι συνήθως η καλύτερη, καθώς προσαρμόζεται στα δεδομένα και τις ανάγκες τους.

Έτσι, με βαθμό πολυωνύμου = 2 και gamma = 'scale', μεταβάλλουμε το C στις τιμές [0.0001, 0.001, 0.1, 1, 10, 100] ώστε να κατανοήσουμε σε ποια τάξη μεγέθους έχουμε καλύτερα αποτελέσματα. Η καλύτερη ακρίβεια φαίνεται να επιτυγχάνεται με C = 1 (σχετικά αναμενόμενο, μιας και αυτή είναι και η default τιμή του στη βιβλιοθήκη που χρησιμοποιήθηκε). Παρατηρείται ότι όσο αυξάνεται το C μέχρι την τιμή 1, το accuracy αυξάνεται επίσης, μέχρι που φτάνει στο μέγιστο του και αρχίζει να ξαναπέφτει, όχι όμως τόσο κατακόρυφα όσο ανέβηκε (τιμές 10 & 100). Παρατηρείται επίσης πως ο χρόνος εκπαίδευσης μικραίνει μέχρι C = 1, και μετά αρχίζει να μεγαλώνει ξανά. Αυτό είναι ένα ευχάριστο συμπέρασμα, μιας και καλύτερη απόδοση είχαμε όταν C = 1. Τέλος, στο χρόνο ανάλογα με το C δε βλέπουμε κάποια ανοδική ή καθοδική τάση, αλλά σύμφωνα με κάποια μικρή έρευνα στο διαδίκτυο φαίνεται long term ότι η αύξηση του C σε μεγαλύτερες τάξεις μεγέθους οδηγεί σε μικρότερους χρόνους overall.

Στη συνέχεια, η παράμετρος που μελετήθηκε είναι το gamma, με τιμές του C ίσες με 0.1 και 1 κάθε φορά (λόγω της απόδοσης τους αλλά και του μικρότερου χρόνου που πέτυχαν στα προηγούμενα πειράματα). Οι τιμές του gamma που χρησιμοποιήθηκαν είναι οι [0.001, 0.01, 0.1, 1, 10]. Το gamma, όπως αναλύθηκε και παραπάνω, δείχνει το βαθμό επιρροής του κάθε στοιχείου του training set στην όλη διαδικασία (μεγάλο gamma -> μικρότερη επιρροή). Η καλύτερη απόδοση φαίνεται να επιτυγχάνεται με gamma = 0.01, και είναι περίπου 82% (παρόμοια με παραπάνω για gamma = 'scale'). Καταλήγουμε δηλαδή πως έχουμε καλύτερα αποτελέσματα όταν η επίδραση στο training είναι μεγαλύτερη παρά μικρότερη. Επίσης σχολιάζεται πως η απόδοση παραμένει σταθερή από ένα σημείο και μετά (gamma = 1 και μετά). Αυτή η σταθεροποίηση σημαίνει πως η επιρροή του κάθε στοιχείου μικραίνει τόσο που το μοντέλο δε μπορεί να συγκλίνει σε διαφορετικό σημείο και να βελτιωθεί κι άλλο κάθε φορά, εννοώντας πως η σφαίρα επιρροής του συμπεριλαμβάνει μόνο τον εαυτό του (το οποίο οδηγεί με σιγουριά και σε overfitting). Για τις ανάγκες του παραπάνω συμπεράσματος εξαγάγαμε το training (σε αντίθεση με το testing πριν) accuracy, που φαίνεται στον παρακάτω πίνακα.

C	Gamma	Accuracy	Training Accuracy
0.1	0.1	0.7955	0.9997
0.1	1	0.7865	1
0.1	10	0.7865	1
1	0.1	0.7865	1
1	1	0.7865	1
1	10	0.7865	1

Επιβεβαιώνεται λοιπόν το αρχικό μας σχόλιο πως πρόκειται για overfitting, καθώς έχουμε training accuracy = 100%, κάτι το οποίο πρέπει να αποφεύγεται σε μοντέλα μηχανικής μάθησης όπως αυτό που μελετάμε.

Τελευταία πειράματα όσον αφορά το πολυωνυμικό kernel είναι εκείνα με μεγαλύτερο βαθμό πολυωνύμου (degree = 3 & degree = 4). Κρατάμε το gamma = 'scale' και το C στις τιμές 0.1 και 1 όπως πριν, για χάρη σύγκρισης και καλύτερων αποτελεσμάτων. Το accuracy σε όλες τις περιπτώσεις παραμένει πολύ καλό (~82%), φτάνοντας και τα υψηλότερα του διωνυμικού kernel. Οι χρόνοι επίσης παραμένουν σταθερά χαμηλοί (~73-74 sec), οπότε συμπεραίνουμε πως και αυτές οι επιλογές θα ήταν κατάλληλες για ένα μοντέλο όπως αυτό που μελετάμε.

- **Radial Basis Function (RBF) SVM** -

Τελευταία συνάρτηση πυρήνα που μελετήθηκε στο πρόβλημα κατηγοριοποίησης 2 κλάσεων είναι η radial basis function, η οποία είναι της μορφής:

$$K(x, x') = e^{-\gamma||x-x'||^2}$$

Στο παρακάτω πινακάκι φαίνονται τα διάφορα πειράματα που διενεργήσαμε, αλλάζοντας ξανά τα C και gamma όπως και πριν.

C	Gamma	Accuracy	Time (s)
0.001	scale	0.6530	150.9930021762848
0.01	scale	0.6840	136.1552219390869
0.1	scale	0.7795	105.5902261734008
1	scale	0.8425	84.71050882339478
10	scale	0.8385	177.9095807075500
100	scale	0.8350	168.1049981117248

C	Gamma	Accuracy	Time (s)
0.1	0.01	0.7800	117.16019439697266
1	0.01	0.8405	102.12415289878845
0.1	0.1	0.5060	140.39141035079956
1	0.1	0.5700	247.82502913475037
0.1	1	0.5015	143.42579221725464
1	1	0.5025	142.70944142341614

0.1	10	0.5490	148.4633116722107
1	10	0.5495	144.63145232200623

Τα συμπεράσματα στα οποία καταλήγουμε όσον αφορά το C είναι πως το μεγαλύτερο (δηλαδή το μικρότερο θεωρητικά margin) μας δίνει εδώ καλύτερα αποτελέσματα. Πιο συγκεκριμένα, το C = 1 έχει το μεγαλύτερο accuracy = 84.25%, με τα επόμενα C (10, 100) να μην πέφτουν όμως μακριά από αυτό (πάντα για gamma = 'scale').

Όσον αφορά την παράμετρο gamma, που δείχνει όπως προ είπαμε την επιρροή κάθε στοιχείου εισόδου, το μικρότερο φαίνεται να δίνει καλύτερα αποτελέσματα. Για gamma = 0.01 λοιπόν έχουμε 78% και 84% accuracy για C = 0.1 και C = 1 αντίστοιχα. Όταν το gamma αυξάνεται εδώ η απόδοση πέφτει αρκετά απότομα, κάτι που δε συμβαίνει στις πολυωνυμικές συναρτήσεις πυρήνα. Αυτό ερμηνεύεται πιθανότατα μέσω της μορφής της ίδιας της RBF (το gamma βρίσκεται στον εκθέτη εκθετικής συνάρτησης, κι έτσι πιθανώς επηρεάζει περισσότερο το αποτέλεσμα).

Σχολιάζεται, τέλος, πως ο χρόνος εκπαίδευσης του RBF kernel στην καλύτερη περίπτωση (C = 1) είναι αποδεκτός (~84 sec), γενικότερα όμως είναι αρκετά αυξημένος σε σχέση με το πολυωνυμικό kernel (περίπου διπλάσιος σε πολλές περιπτώσεις). Όσον αφορά τη σύγκριση με το χρόνο του γραμμικού SVM από την άλλη, τα δύο kernels (πολυωνυμικό και RBF) παρουσιάζουν σίγουρα τα αποτελέσματα γρηγορότερα. Η παρατήρηση αυτή είναι βέβαια αναμενόμενη, καθώς γνωρίζουμε από την αρχή πως το πρόβλημα δεν είναι γραμμικά διαχωρίσιμο, με αποτέλεσμα ένα γραμμικό μοντέλο να αντιμετωπίζει δυσκολίες στη σύγκλιση, οι οποίες οδηγούν και σε μεγαλύτερους χρόνους.

Στη συνέχεια παρουσιάζονται τα αποτελέσματα για k-nearest neighbor και nearest centroid αλγορίθμους, στο ίδιο πρόβλημα κατηγοριοποίησης 2 κλάσεων του cifar-10, ώστε να προχωρήσουμε σε σύγκριση με τα παραπάνω μοντέλα SVM.

	Time (s)	**Accuracy**
k-nearest neighbor (k=1)	2.13547682762146	0.7180
k-nearest neighbor (k=3)	2.26756858825683	0.7395
nearest centroid	1.11625123023986	0.6390

Οι παραπάνω αλγόριθμοι δίνουν πολύ καλά αποτελέσματα, ειδικά αν συγκρίνουμε το χρόνο που κάνουν να ολοκληρωθούν σε σχέση με τα SVM counterparts τους. Ο καλύτερος αλγόριθμος από τους 3 (k-nearest neighbor με k = 3) έχει accuracy ίσο με 73.95%, μόλις 10% λιγότερο από το καλύτερο SVM που μελετήσαμε, με μία σημαντική διαφορά: παρουσιάζει αυτό το αποτέλεσμα 40-50 φορές γρηγορότερα. Συμπεραίνεται λοιπόν πως οι παραπάνω αλγόριθμοι (και ειδικά ο k-nearest neighbor) δίνουν πολύ καλό accuracy στο πρόβλημα

κατηγοριοποίησης 2 κλάσεων και πρέπει σίγουρα να εξετάζονται σε παρόμοια προβλήματα πριν τη μετάβαση σε περιπλοκότερες λύσεις.

Κατηγοριοποίηση 10 κλάσεων (όλο το dataset) με 3 διαφορετικά μοντέλα SVM

Αφού μελετήθηκε η κατηγοριοποίηση για 2 κλάσεις του dataset (horse & dog), διαδικασία πιο intuitive σε ένα SVM, το επόμενο βήμα ήταν η κατηγοριοποίηση όλων των κλάσεων με τη βοήθεια παρόμοιων μοντέλων.

Επί της ουσίας, ο τρόπος με τον οποίο λειτουργεί αυτή η διαδικασία είναι με την δημιουργία (εσωτερικά) από τη συνάρτηση SVC() πολλών μικρότερων One-Vs-One classifiers για όλους τους συνδυασμούς των κλάσεων, οι οποίοι μετά “ενώνονται” και παρουσιάζουν μια μέση απόδοση όλου του μοντέλου. Μεμονωμένα, η “ένωση” αυτή γίνεται με τη μορφή “ψήφων”, εννοώντας πως όταν πρέπει να αποφασιστεί η κλάση ενός στοιχείου του testing set, όλοι οι διαφορετικοί One-Vs-One classifiers που το αφορούν καταλήγουν στην πρόβλεψη μιας κλάσης, και εκείνη με τις περισσότερες προβλέψεις επιλέγεται ως η πιο πιθανή.

Παρακάτω παρουσιάζονται σε πίνακες τα διάφορα πειράματα που διενεργήθηκαν κατά τη διαδικασία της μελέτης κατηγοριοποίησης δεδομένων εισόδου σε όλες τις κλάσεις του cifar-10. Υπενθυμίζεται πως έγινε επιλογή μόνο 10000 training δεδομένων και 2000 testing δεδομένων, καθώς αλλιώς ο χρόνος παρουσίαζε μεγάλη αύξηση.

LINEAR SVM

C	Accuracy	Time (s)
0.001	0.3815	161.9136013984680
0.01	0.3950	142.0474944114685
0.1	0.3690	150.5944437980651
1	0.3355	190.4883549213409
10	0.3140	243.9135737419128
100	0.3165	245.1096408367157

POLYNOMIAL KERNEL SVM

C	Degree	Gamma	Accuracy	Time (s)
0.001	2	scale	0.2525	226.3784701824188
0.01	2	scale	0.3610	185.9478256702423
0.1	2	scale	0.4170	153.6823928356170

1	2	scale	0.4385	143.6533143520355
10	2	scale	0.4275	166.6594498157501
100	2	scale	0.4095	167.2917962074279

C	Degree	Gamma	Accuracy	Time (s)
0.1	2	0.001	0.3240	193.46909141540527
1	2	0.001	0.3980	155.11192297935486
0.1	2	0.01	0.4370	138.66879296302795
1	2	0.01	0.4340	139.27232122421265
0.1	2	0.1	0.4110	162.62125945091248
1	2	0.1	0.4100	161.11308121681213
0.1	2	1	0.4100	164.71191501617432
1	2	1	0.4100	160.73708939552307
0.1	2	10	0.4100	163.0132131576538
1	2	10	0.4100	160.1169831752777

C	Degree	Gamma	Accuracy	Time (s)
0.1	3	scale	0.4390	138.154256105422
1	3	scale	0.4365	144.837911367416
0.1	4	scale	0.4260	134.264888525009
1	4	scale	0.4275	133.538266897201

RBF KERNEL SVM

C	Gamma	Accuracy	Time (s)
0.001	scale	0.0975	257.6466314792633
0.01	scale	0.2180	247.22992372512817
0.1	scale	0.3865	198.19889450073242
1	scale	0.4665	166.07999205589294
10	scale	0.4710	199.11207818984985
100	scale	0.4575	200.21591567993164

C	Gamma	Accuracy	Time (s)
0.1	0.01	0.3635	205.570125341415
1	0.01	0.4735	192.158151388168
0.1	0.1	0.0975	246.046138048172
1	0.1	0.1950	260.529940605163
0.1	1	0.0975	244.041933059692
1	1	0.0975	267.115957498550
0.1	10	0.0975	253.211651802063
1	10	0.0975	271.280808210372

Πρώτο σχόλιο, όσον αφορά το γραμμικό SVM, είναι πως η απόδοση είναι αρκετά κακή, δεν αγγίζει δηλαδή το 40%. Το καλύτερο accuracy πάντως (39.5%) επιτυγχάνεται με C = 0.01, όπως και στην προηγούμενη μελέτη για κατηγοριοποίηση 2 μόνο κλάσεων. Στο ίδιο C παρατηρείται και ο μικρότερος χρόνος εκπαίδευσης, κάτι προφανώς θετικό.

Συνεχίζοντας με το πολυωνυμικό kernel, και ξεκινώντας με του 2^{ου} βαθμού, οι αποδόσεις του μοντέλου βελτιώνονται, με αποκορύφωμα το accuracy = 43.85% όταν C = 1 (με gamma = 'scale' όπως σε όλα τα αρχικά πειράματα). Ακριβώς το ίδιο C έδωσε την καλύτερη απόδοση και στα πειράματα 2 κλάσεων. Παρατηρείται λοιπόν (σχετικά αναμενόμενα) ότι οι ίδιες παράμετροι λειτουργούν καλύτερα και στο πρόβλημα της κατηγοριοποίησης όλων των κλάσεων. Η απόδοση φυσικά πέφτει αισθητά λόγω της δυσκολίας κατηγοριοποίησης σε 10 κλάσεις, αλλά κατά τα άλλα τα αποτελέσματα φαίνεται να είναι παρόμοια.

Παρόμοια είναι και τα συμπεράσματα από τη μελέτη της επιρροής του gamma, όπου το καλύτερο accuracy παρατηρείται για gamma = 0.01, και από gamma = 0.1 και μετά ξεκινάει η υπερεκπαίδευση του μοντέλου. Ακόμα και αν η απόδοση είναι σχετικά καλή, το overfitting είναι κάτι που θέλουμε σίγουρα να αποφύγουμε, οπότε το πιο πιθανό είναι πως δε θα λαμβάναμε υπόψιν μας gamma μεγαλύτερα από αυτή την τιμή σε τέτοιου είδους (και παρεμφερή) προβλήματα.

Τέλος, τα πολυωνυμικά kernel 3^{ου} και 4^{ου} βαθμού συνεχίζουν να δίνουν καλές αποδόσεις (τις καλύτερες από όλα τα πολυωνυμικά μοντέλα που μελετήσαμε), σε σχετικά καλούς χρόνους. Ο χρόνος εκπαίδευσης στο πρόβλημα των 10 κλάσεων είναι περίπου διπλάσιος σε πολλές περιπτώσεις, κάτι αναμενόμενο μιας και υπάρχει αυξημένη δυσκολία. Πιο συγκεκριμένα, και όσον αφορά τα polynomial kernels, τα 3^{ου} και 4^{ου} βαθμού έχουν τους καλύτερους χρόνους εκπαίδευσης, κάτι το οποίο "συμφέρει", μιας και παρουσιάζουν τις καλύτερες αποδόσεις.

Πέρα από τα 2 προαναφερθέντα μοντέλα, μελετήθηκε φυσικά και το RBF kernel, με ανάμεικτα αποτελέσματα. Αρχικά, δοκιμάσθηκαν διάφορες τιμές στην παράμετρο C, και παρατηρήθηκε

πως όσο αυξάνεται, αυξάνεται και το accuracy του μοντέλου (με μικρά fluctuations). Καλύτερη ακρίβεια έχουμε όταν C = 10 (47.1%) το οποίο, σε αντίθεση με τα παραπάνω, δεν αντιστοιχεί στην καλύτερη περίπτωση RBF για κατηγοριοποίηση 2 κλάσεων (C = 1). Επίσης σχολιάζεται πως ο χρόνος εκπαίδευσης στο RBF kernel είναι κάπως αυξημένος σε σχέση με τις πολυωνυμικές συναρτήσεις, παρουσιάζει όμως κατά περιπτώσεις αρκετά καλύτερη (+5%) απόδοση, οπότε πιθανώς να αξίζει. Τέλος, οι διαφορετικές τιμές του gamma (πέρα από gamma = 0.01 που δίνει αποδεκτά αποτελέσματα) παρουσιάζουν πολύ κακή, σταθερή απόδοση (σίγουρα συνδυασμένη με overfitting). Η επιλογή του gamma σε κάποια άλλη τιμή λοιπόν (και όχι gamma = 'scale') δεν προσφέρει σχεδόν τίποτα και κατά πάσα πιθανότητα θα απορριπτόταν από την αρχή.

Παρακάτω παρουσιάζονται τα αποτελέσματα για k nearest neighbor και nearest centroid αλγορίθμους, στο ίδιο πρόβλημα classification του cifar-10. Σημειώνεται πως εδώ η εκπαίδευση έγινε με ολόκληρο το training και testing dataset (50000 & 10000 στοιχεία αντίστοιχα) καθώς δεν παρουσιαζόταν πρόβλημα χρόνου.

	Time (s)	**Accuracy**
k-nearest neighbor (k=1)	21.1209619	0.3539
k-nearest neighbor (k=3)	22.2508304	0.3303
nearest centroid	1.37121844	0.2774

Η απόδοση και των τριών αλγορίθμων δεν είναι πολύ κακή, στην καλύτερη περίπτωση μάλιστα (k-nearest neighbor με k = 1) πλησιάζει κάποια από τα παραπάνω πειράματα, σε αρκετά μικρότερο χρόνο. Θα ήταν επιθυμητή λοιπόν η περαιτέρω εξέταση του προβλήματος και η σύγκριση των μεθόδων, ειδικά αν ο μειωμένος χρόνος έπαιζε σημαντικό ρόλο στη μελέτη overall.

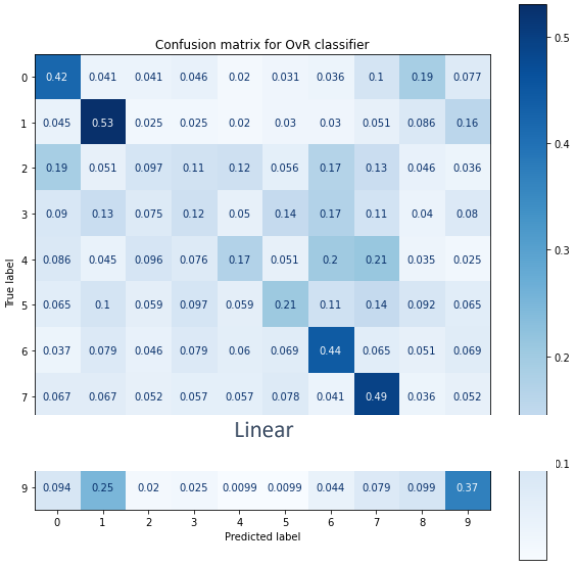
One-Vs-Rest (OVR) αντιμετώπιση του προβλήματος κατηγοριοποίησης

Σαν τελευταίο κομμάτι της εργασίας έγινε μια προσπάθεια δημιουργίας ενός One-Vs-Rest classifier. Αναφέρθηκε ήδη πως η εκπαίδευση στην βιβλιοθήκη SVC(), επειδή είναι βασισμένη στην libsvm, χρησιμοποιεί την One-Vs-One τεχνική για να δώσει αποτελέσματα. Εδώ, αφού χρησιμοποιούμε την βιβλιοθήκη SVC για να ορίσουμε τις παραμέτρους του μοντέλου, το περνάμε σε έναν OneVsRestClassifier, και μέσω αυτού προχωράμε στο training/fit. Σημειώνεται εδώ πως δεν προχώρησα με σιγουριά σε αυτή τη διαδικασία και δεν γνωρίζω αν λειτουργεί όπως αναμένεται, αλλά ήθελα να συμπεριληφθεί στην εργασία.

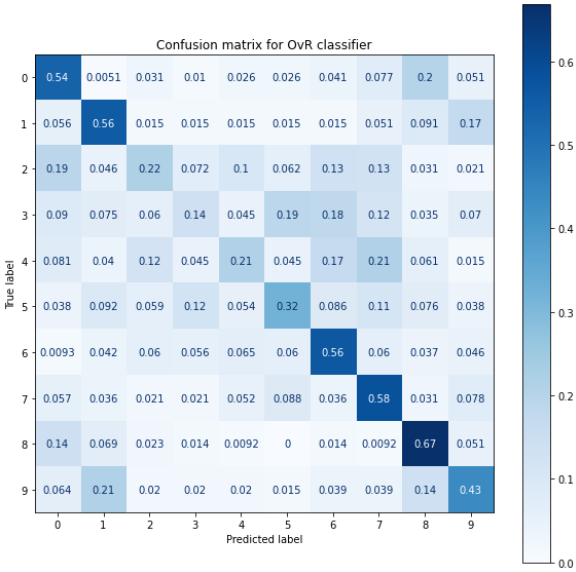
Η διαφορά του One-Vs-Rest classifier είναι πως πλέον δεν δημιουργούνται 1v1 classifiers μεταξύ όλων των κλάσεων, αλλά δημιουργείται ένας για κάθε κλάση, ο οποίος αποφασίζει αν κάθε στοιχείο ανήκει σε αυτή την κλάση ή όχι. Εκεί λοιπόν που πριν είχαμε 45 "εσωτερικούς" κατηγοριοποιητές, τώρα έχουμε 10.

Έγιναν μερικά πειράματα πάνω στην παραπάνω διαδικασία, με διαφορετικών ειδών SVM, από τα οποίο προέκυψαν confusion matrices για την καλύτερη κατανόηση των αποτελεσμάτων. Οι 10 classifiers, λοιπόν, και τα ποσοστά επιτυχίας τους, φαίνονται σε αυτά τα confusion matrices. Παρατηρείται, όπως ήταν και αναμενόμενο, πως δεν προβλέπονται όλες οι κλάσεις με τον ίδιο βαθμό ευκολίας. Στον παρακάτω πίνακα εμφανίζονται οι πληροφορίες και τα αποτελέσματα κάθε μοντέλου.

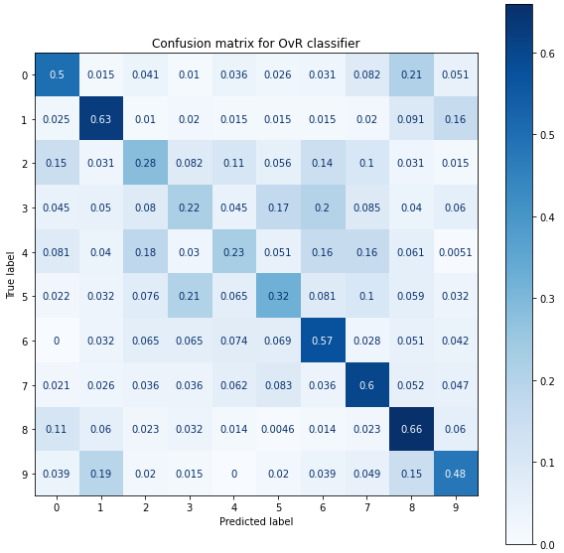
Model	C	Gamma	Accuracy	Time (s)	Accuracy (OvO)
linear	0.01	scale	0.3370	455.82400655746	0.3950
poly (degree = 2)	0.01	scale	0.4265	406.93791604042	0.3610
poly (degree = 3)	0.1	scale	0.4530	446.80841016769	0.4390
RBF	10	scale	0.4770	618.33813452720	0.4710



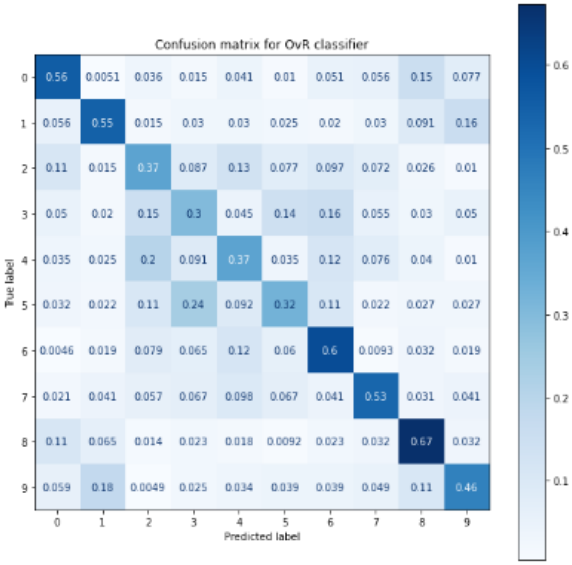
Linear



Polynomial, degree = 2



Polynomial, degree = 3



RBF

Αρχικά, σημειώνεται πως επιλέχθηκαν τα C = 0.01 και gamma = 'scale' διότι με βάση το προηγούμενο κομμάτι της εργασίας είναι αυτά που δίνουν τα καλύτερα (ή κοντά στα καλύτερα αποτελέσματα) -εκτός από το 3^{ου} βαθμού πολυώνυμο που επιλέχθηκε C=0.1 και το RBF που επιλέχθηκε C = 10.

Όσον αφορά την απόδοση, παρατηρούμε καλύτερα απόδοση στα πολυωνυμικά μοντέλα σε σχέση με το One-Vs-One approach, και χειρότερη στα γραμμικά και RBF SVMs. Είτε στην μία είτε στην άλλη περίπτωση όμως φαίνεται ξεκάθαρα πως ο χρόνος εκπαίδευσης όταν επιλέγουμε OvR classifiers είναι πολύ μεγαλύτερος, σε κάποιες περιπτώσεις μέχρι και 3 φορές. Συμπεραίνουμε λοιπόν πως (με τις συγκεκριμένες παραμέτρους αλλά πιθανότατα και με άλλες) ο χρόνος είναι ένα χαρακτηριστικό που "απαγορέυει" αυτού του είδους την προσέγγιση του προβλήματος, ακόμα και όταν δίνει καλύτερα αποτελέσματα και προβλέψεις.

Τέλος, από τα confusion matrices παρατηρούμε πως γενικότερα πιο καλή συμπεριφορά (με καλύτερα ποσοστά επιτυχίας) παρουσιάζει το RBF, το οποίο φαίνεται να είναι πιο σταθερό overall. Αυτή η παρατήρηση επιβεβαιώνεται και από την τιμή της απόδοσης, η οποία είναι αρκετά υψηλή (47.7%) και αγγίζει το 50%, τιμή-στόχο στο συγκεκριμένο πρόβλημα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

https://scikit-learn.org/stable/modules/svm.html#svm

https://stackoverflow.com/questions/4214868/an-example-using-python-bindings-for-svm-library-libsvm

https://towardsdatascience.com/support-vector-machines-explained-with-python-examples-cb65e8172c85

https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

https://github.com/mok232/CIFAR-10-Image-Classification

https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

https://stats.stackexchange.com/questions/270187/svm-why-does-the-number-of-support-vectors-decrease-when-c-is-increased

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

https://towardsdatascience.com/svm-classifier-and-rbf-kernel-how-to-make-better-models-in-python-73bb4914af5b

https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

https://www.sciencedirect.com/science/article/abs/pii/S002002551631043X

https://www.machinecurve.com/index.php/2020/11/11/creating-one-vs-rest-and-one-vs-one-svm-classifiers-with-scikit-learn/