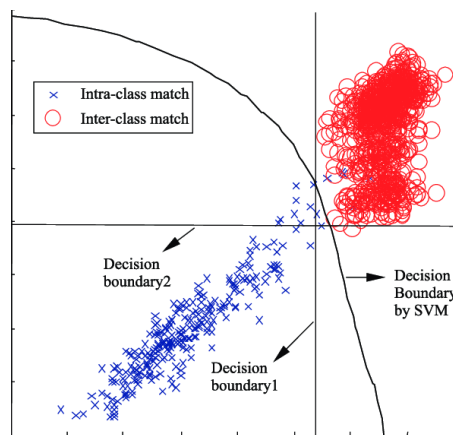



Διναύσματα Μηχανικής Υποστήριξης για  
Διαχωρισμό Εικόνων σε Κλάσεις  
Support Vector Machines for Multiclass  
Classification - CIFAR10



Αριστοστέλειο Πανεπιστήμιο  
Θεσσαλονίκης  
10ο Εξάμηνο ECE AUTH  
7ο Εξάμηνο CSD AUTH  
Νευρωνικά Δίκτυα - Βαθιά Μάθηση

Χάρης Φίλης AEM: 9449  
GitHub Repository Link :   
GitHub Repository Folder :   
Update branch: 

December, 2022

## Contents

1	Εισαγωγή	3
2	Αλγόριθμος K πλησιέστερων γειτόνων	3
3	Μηχανές Διανυσμάτων Υποστήριξης ( <i>SVM</i> )	4
3.1	Κατηγοριοποιητής τριών κλάσεων - <i>3 class Support Vector Classifier(SVC)</i>	4
3.2	Προεπεξεργασία Δεδομένων- <i>Data Preprocessing</i>	4
3.3	<i>SVM compare</i>	5
3.3.1	Σχολιασμός Χρόνου	6
4	Διαγράμματα Αποτελεσμάτων	6
4.1	<i>Linear Kernel Models</i>	6
4.1.1	Accuracy over C	6
4.2	<i>RBF Kernel Models</i>	6
4.2.1	Training-Evaluation Times	6
4.3	<i>RBF Kernel Models</i>	7
4.3.1	Accuracy over Gamma	7
4.3.2	Training-Evaluation Times	7
4.3.3	Confusion Matrix-Heatmap	7
4.4	Τελικά Σχόλια	7

## List of Figures

1	Classification report k=3	3
2	Classification results nearest class centroids	4
3	Confusion Matrix correct = true_label incorrect=predicted_label	5
4	x:C parameters y: Accuracies	6
5	Timings Linear	6
6	x:Gamma Values y: Accuracies	7
7	Timings RBF	7

## 1 Εισαγωγή

Αντικείμενο της παρούσας εργασίας, ήταν η συγγραφή ενός προγράμματος που υλοποιεί ένα *Support Vector Machine (SVM)*, το οποίο επιλύει το πρόβλημα της κατηγοριοποίησης τριών κλάσεων. Το *dataset* που επιλέχθηκε είναι το *CIFAR10*, το οποίο έχει RGB εικόνες διάστασης  $32 \times 32 \text{px}$  και σκοπός των μοντέλων που δημιουργήθηκαν ήταν να διαχωρίσουν τις εικόνες σε όλες τις κλάσεις τους, αλλά αυτό το πείραμα που έτρεξε κυρίως (λόγω περιορισμένου χρόνου και υποχρεώσεων) είναι στο αρχείο *svm-Compare\_3ClassClassifier.py* και έχει κυρίως ως στόχο τον διαχωρισμό 3 κλάσεων του *dataset* και συγκεκριμένα των κλάσεων σκύλος,γάτα,άλογο. Παράλληλα έγινε επιλογή ενός πολύ μικρού υποσυνόλου *training* 3000 δειγμάτων και για *validation* 1000 δειγμάτων για να μην καθυστερήσει η εκπαίδευση αντιπροσωπευτικού set μοντέλων με γραμμικό ή *rbf kernel*. Τέλος αξίζει να σημειωθεί πως δημιουργήθηκαν διάφορα μοντέλα όπως δύο *from scratch SVM* το ένα που χρησιμοποιεί γραμμικό *kernel* στο *kernel trick* ενώ το άλλο *rbf kernel*, ένα *Linear SVM* που κάνει χρήση της βιβλιοθήκης *LibLinear* της *sklearn* και 3 μοντέλα που χρησιμοποιούν την *LibSVM* και το ένα από αυτά υλοποιεί *PCA* ενώ το άλλο κάνει μετατοπίζει την λειτουργικότητα στο *intel dataset*. Τέλος έγινε και ένα *script* βελτιστοποίησης των υπερπαραμέτρων (*model\_hyperparameter\_tunning.py*) των μοντέλων χρησιμοποιώντας την μέθοδο *cross validation* για το ανακάτεμα και το σπάσιμο του *dataset* και τον αλγόριθμο *grid search* για την διερεύνηση βέλτιστων παραμέτρων. Δυστυχώς το μόνο που πρόλαβα να τρέξω (στα πλαίσια του χρόνου που είχα) ήταν το *svmCompare* το οποίο υλοποιεί μοντέλα για διαχωρισμό 3 κλάσεων όπως ειπώθηκε. Ίσως στην παρουσίαση να έχω αποτελέσματα και από άλλα μοντέλα.

## 2 Αλγόριθμος K πλησιέστερων γειτόνων

Σαν προπαρασκευαστική εργασία είχαμε τρέξει το αλγόριθμο *KNN* ο οποίος είναι ένας απλός σχετικά αλγόριθμος κατηγοριοποίησης δεδομένων με βάση το πόσο κοντά βρίσκονται τα δείγματα όσον αφορά την ευκλείδεια απόστασή τους. Ο συγκεκριμένος αλγόριθμος όπως είχαμε δείξει και σε προηγούμενες εργασίες δεν λειτουργεί με βάση να βελτιώνει αυτή την κατηγοριοποίηση που κάνει και πετυχαίνει αρκετά χαμηλά επίπεδα ακρίβειας στην κατηγοριοποίηση ενός σύνθετου *dataset* εικόνων όπως είχαμε δείξει και βλέπουμε παρακάτω για πλήθος τριών γειτόνων.

	precision	recall	f1-score	support
0	0.20	0.65	0.30	57
1	0.96	0.02	0.03	41
2	0.99	0.12	0.19	51
3	0.14	0.12	0.13	49
4	0.97	0.20	0.11	44
5	0.35	0.14	0.20	50
6	0.95	0.02	0.03	56
7	0.12	0.02	0.04	40
8	0.43	0.16	0.23	57
9	0.00	0.00	0.00	59
accuracy			0.15	512
macro avg	0.15	0.15	0.12	512
weighted avg	0.15	0.15	0.12	512

(c) K = 3

Figure 1: Classification report k=3

Αντίστοιχα και ίσως χειρότερα, είναι και τα αποτελέσματα του αλγορίθμου *Nearest Class Centroid*, ο οποίος λειτουργεί με πιο απλή λογική συγκρίνοντας με ένα προεπιλεγμένο κέντρο νέα δείγματα

Ωστόσο, όπως θα δούμε και στην συνέχεια επειδή τα μοντέλα που εκπαιδεύτηκαν με

Time elapsed: 10.648506787155151 s

	precision	recall	f1-score	support
0	0.10	0.07	0.08	57
1	0.12	0.05	0.07	41
2	0.22	0.14	0.17	51
3	0.18	0.14	0.16	49
4	0.11	0.30	0.16	44
5	0.17	0.12	0.14	50
6	0.29	0.09	0.14	56
7	0.17	0.19	0.18	48
8	0.24	0.42	0.31	57
9	0.39	0.41	0.40	59
accuracy			0.20	512
macro avg	0.20	0.19	0.18	512
weighted avg	0.21	0.20	0.19	512

Figure 2: Classification results nearest class centroids

τον αλγόριθμο του SVM δεν εκπαιδεύτηκαν σε μεγάλο σύνολο δεδομένων καθώς και επίσης ο ίδιος ο αλγόριθμος δεν αποτελεί *state of art solution* στο πρόβλημα όπως τα MLP που είδαμε ήδη τα αποτελέσματα που προέκυψαν είναι αρκετά απογοητευτικά σε επίπεδα που φτάνουμε και τις επιδόσεις του NCC. Βέβαια όπως θα φανεί αργότερα, αν και ο K-NN προσφέρει μεγαλύτερη ακρίβεια, δεν προσφέρει την βέλτιστη ταχύτητα για την πρόβλεψη νέων δεδομένων (μιας και είναι άπληστος αλγόριθμος) σε σχέση με ένα SVM. Η πολυπλοκότητα του KNN είναι άμεσα εξαρτώμενη και μεγαλώνει όσο μεγαλώνει το training set.

### 3 Μηχανές Διανυσμάτων Υποστήριξης (SVM)

Για την κατασκευή των μηχανών διανυσμάτων υποστήριξης της εργασίας χρησιμοποιήθηκε τόσο η *sklearn* (το κομμάτι που χρησιμοποιεί την *LibSVM* αλλά και το κομμάτι που λύνει το πρόβλημα στο *primal space* *LibLinear* με την διαμάχη των *svm* με την ταχτική *one versus rest*) όσο και *from scratch* προσπάθειες υλοποίησης.

#### 3.1 Κατηγοριοποιητής τριών κλάσεων - 3 class Support Vector Classifier(SVC)

Λόγων περιορισμένου χρόνου ο τύπος των κατηγοριοποιητών που εξετάστηκε είναι προϊόν της *LibSVM* που χρησιμοποιεί η *sklearn* και δεν λύνει το πρόβλημα στο *primal space* αλλά μετασχηματίζει τα δεδομένα ή μάλλον χρησιμοποιεί το *kernel trick* συγκρίνοντας τις ομοιότητες των δεδομένων σε υψηλότερο χώρο όπου το πρόβλημα είναι γραμμικά διαχωρίσιμο χωρίς να μετασχηματίζει τα δεδομένα. (Σε περίπτωση που κριθεί απαραίτητο για λόγους πληρότητας παρουσίασης να ολοκληρωθούν τα πειράματα αυτό θα γίνει χωρίς βαθμολογικό αντίκτυπο όμως).

#### 3.2 Προεπεξεργασία Δεδομένων- Data Preprocessing

Στον φάκελο *auxiliary\_funcs* βρίσκονται βοηθητικές συναρτήσεις για την λήψη, χειρισμό, προ επεξεργασία και αποτύπωση δεδομένων σε διαγράμματα. Εδώ βρίσκεται και το αρχείο κώδικα *data\_preprocessing.py* το οποίο υλοποιεί την συνάρτηση *data\_preproc*. Η παρούσα συνάρτηση είναι υπεύθυνη για την κλήση της συνάρτησης λήψης δεδομένων και την κανονικοποίηση των δεδομένων όσο και για τον μετασχηματισμό τους σε *principal components* αν αυτό κριθεί απαραίτητο για να λυθεί το πρόβλημα σε μικρότερο χώρο δεδομένων PCA (αυτό γίνεται με τον PCA μετασχηματισμό που προσφέρει η *sklearn.preprocessing*). Η συνάρτηση αυτή παίρνει διαχωρισμένα τα δεδομένα σε *train* και *test* σύνολα το οποίο γίνεται με συνάρτηση, *train\_test\_split*, που

κάνει χρήση *masking* 0.8-0.2 για να χωρίσει τα δεδομένα. Το κομμάτι της κανονικοποίησης το αναλαμβάνει ο *StandardScaler* της *sklearn.preprocessing* ο οποίος προσαρμόζεται στα δεδομένα και τα κανονικοποιεί με βάση την τυπική απόκλιση τους και την μέση τιμή τους. Αυτό γίνεται για να εξαλείψουμε διαφορές εικόνων που έχουν να κάνουν και με τα χαρακτηριστικά της εικόνας και πιθανόν της λήψης της (*exposure, contrast*) τα οποία επηρεάζουν το *pixel value*. Στην συνέχεια, στις ετικέτες εφαρμόζεται κωδικοποίηση τύπου *one\_hot* με συνάρτηση που υλοποιώ παραπάνω για να μειωθεί η υπολογιστή πολυπλοκότητα στην σύγκριση των *labels* με την έξοδο του δικτύου. Τέλος τα δεδομένα φορτώνονται σε *DataFrames* που είναι η βασική κλάση της βιβλιοθήκης *pandas* ώστε να έχουμε ένα πιο σαφές αποτύπωμα των δεδομένων μιας και έχουμε και κατηγορικά δεδομένα. Τέλος δίνεται ένα παράδειγμα μια συνάρτησης που βρήκα όταν προσπαθούσα να μειώσω την διάσταση των δεδομένων η οποία δεν χαλάει την εντροπία του *subset* που δημιουργείται (δεν πρόλαβα να τη χρησιμοποιήσω ωστόσο μάλλον αυτή θα βελτίωνε τα αποτελέσματα μιας και ο τρόπος που παίρνω μέρος του *dataset* δεν γίνεται με σεβασμό της διατήρησης της εντροπίας των δεδομένων).

### 3.3 SVM compare

Το script *svmCompare\_3ClassClassifier.py* κάνει χρήση του SVC μοντέλου της *sklearn*. Τα υπόλοιπα μοντέλα δυστυχώς άλλοτε είχαν *bugs* είτε δεν πρόλαβα να τα τρέξω. Αρχικά εισάγονται τα *DataFrames* με την παραπάνω συνάρτηση. Σε αυτή την περίπτωση το SVM δεν μπορεί να διαχειριστεί πολλά δεδομένα για αυτό γίνεται περιορισμός του *dataset* όπως είχε αναφερθεί στην εισαγωγή (αντίθετα το *LinearSVC* θα μπορούσε να τα διαχειριστεί). Εκπαιδεύεται ένα αρχικό μοντέλο ( $C = 1$ , *kernel* = 'linear') με γραμμικό *kernel* σε αυτά τα δεδομένα και στην συνέχεια αξιολογείται. Το αποτέλεσμα είναι απογοητευτικό με 9% επιτυχία αναγνώρισης μιας από τις τρεις κλάσεις. Ενώ ο *confussion matrix* είναι μόνο με *false negative*. Έτσι στην συνέχεια εκπαιδεύον-

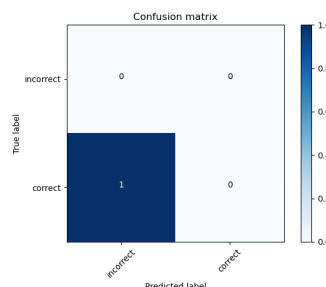


Figure 3: Confussion Matrix correct = true\_label incorrect=predicted\_label

ται πολλά μοντέλα με διαφορετικές ανοχές  $C$  για το γραμμικά μοντέλα και για τα μοντέλα που χρησιμοποιούν ως *kernel* το *rbf* γίνονται διαφορετικά μοντέλα με διαφορετικό *gamma* της *gaussian* κατανομής. Τα υπόλοιπα αρχεία που έχω στο φάκελο *experiments* ήταν αρχεία με μοντέλα και πειράματα που είχα σκοπό να τρέξω. Καθώς και θα ήθελα να τρέξω *test scripts* για τα μοντέλα *from scratch* καθώς και για το *LinearSVC*.

### 3.3.1 Σχολιασμός Χρόνου

Δεδομένου του περιορισμένου συνόλου δεδομένων που χρησιμοποιήθηκε για εκπαίδευση και αξιολόγηση ο χρόνος που χρειάζεται για εκπαίδευση ενός τέτοιου μοντέλου δεν ξεπερνά τα 40 δευτερόλεπτα ενώ ο χρόνο που απαιτείται για την αξιολόγηση είναι 10 δευτερόλεπτα

## 4 Διαγράμματα Αποτελεσμάτων

### 4.1 Linear Kernel Models

#### 4.1.1 Accuracy over C

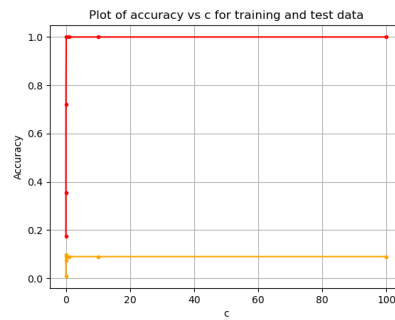
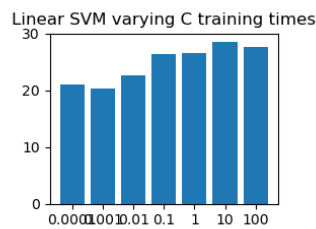


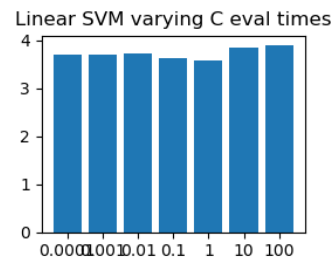
Figure 4: x:C parameters y: Accuracies

### 4.2 RBF Kernel Models

#### 4.2.1 Training-Evaluation Times



(a) Training x:C y:Time(sec)



(b) Eval x:C y:Time(sec)

Figure 5: Timings Linear

### 4.3 RBF Kernel Models

#### 4.3.1 Accuracy over Gamma

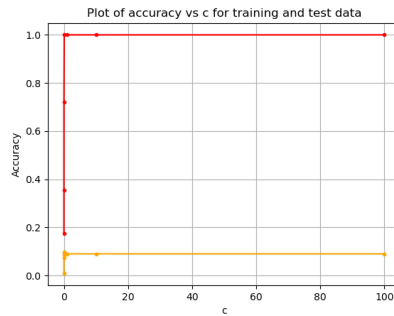
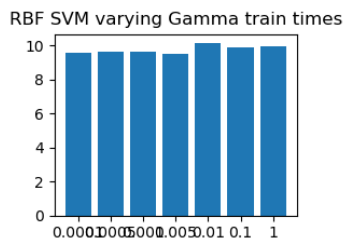


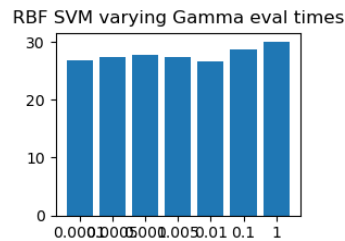
Figure 6: x:Gamma Values y: Accuracies

#### 4.3.2 Training-Evaluation Times

#### 4.3.3 Confusion Matrix-Heatmap



(a) Training x:Gamma Vals  
y:Time(sec)



(b) Eval x:Gamma Vals y:Time(sec)

Figure 7: Timings RBF

### 4.4 Τελικά Σχόλια

Παρατηρούμε ξεκάθαρο *overfitting* των μοντέλων και αδυναμία πρόβλεψης δειγμάτων στο *test set* σε σημεία που αναρωτιέμαι ότι κάτι έχει πάει λάθος στον κώδικα. Βέβαια το σύνολο δεδομένων ήταν πολύ μικρό και αυτό δικαιολογεί τα αποτελέσματα. Έτσι θα μπορούσα να χρησιμοποιήσω το *LinearSVC* μοντέλο που έχει μεγαλύτερη ευελιξία σε πολλά δεδομένα και να πετύχω καλύτερα αποτελέσματα με μεγαλύτερο μέγεθος *training* και *test set*. Λόγω του μικρού συνόλου δεδομένων βλέπουμε και στο *rbf kernel* χειρότερα ακόμη αποτελέσματα κατά το τρέξιμο του κώδικα που είναι πιο εμφανείς οι τιμές των μετρικών.