2η εργασία Νευρωνικών Δικτύων 2024-2025

Αβραμίδου Αφροδίτη, ΑΕΜ: 10329, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

I. ΠΡΟΛΟΓΟΣ

Ζητούμενο εργασίας: Υλοποίηση αλγορίθμου για Support Vector Machine, το οποίο εκπαιδεύεται για να επιλύει τα προβλήματα:

a) **Διαχωρισμού 2 κλάσεων**
b) **Διαχωρισμού όλων των κλάσεων**

της βάσης δεδομένων Cifar-10. Για την εξαγωγή των χαρακτηριστικών των δειγμάτων έχει εφαρμοστεί σε αυτά ο αλγόριθμος PCA για την μείωση των διαστάσεών τους τόσο όσο να εξακολουθεί να υπάρχει περισσότερο από το 90% της αρχικής πληροφορίας, συγκεκριμένα έχει πραγματοποιηθεί μείωση των features σε 200, κρατώντας το 95% τα αρχικής πληροφορίας. Στην αναφορά περιλαμβάνονται η περιγραφή του αλγορίθμου, ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου για διαφορετικούς πυρήνες τόσο γραμμικό όσο και μη γραμμικούς, εξέταση των παραμέτρων τους για εύρεση της βέλτιστης απόδοσης, παραδείγματα ορθής και λανθασμένης κατηγοριοποίησης των κλάσεων. Τέλος γίνεται σύγκριση της απόδοσης του Support Vector Machine με αυτήν της κατηγοριοποίησης 1 και 3 πλησιέστερου γείτονα (Nearest Neighbor) και πλησιέστερου κέντρου κλάσης (Nearest Class Centroid) καθώς επίσης και με ένα MLP με ένα κρυφό επίπεδο που χρησιμοποιεί Hinge loss για την βελτιστοποίηση.

II. ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

A. Υλοποίηση SVM με χρήση της cvxopt και συγκεκριμένα της μεθόδου cvxopt.solvers.qp() για την επίλυση του προβλήματος τετραγωνικού προγραμματισμού, για την διάκριση μεταξύ των κατηγοριών 0 (αεροπλάνο) και 1 (αυτοκίνητο) :

Εισαγωγή βιβλιοθηκών:
numpy και cvxopt για αριθμητικούς υπολογισμούς και βελτιστοποίηση αντίστοιχα.
TensorFlow για τη φόρτωση των δεδομένων της cifar-10 βάσης δεδομένων.
PCA (Principal Component Analysis) από την Scikit-learn για τη μείωση των διαστάσεων των δεδομένων (δεν θεώρησα ότι χρειάζεται να ξανακάνω υλοποίηση του αλγορίθμου PCA from scratch καθώς αυτό έγινε στα πλαίσια της 1ης εργασίας) .

Κλάση SVM_classifier:

__init__(self, c_parameter): ορίζεται η c_parameter, παράμετρος κανονικοποίησης που ελέγχει το trade-off μεταξύ του περιθωρίου (margin) και της ανοχής των σφαλμάτων ταξινόμησης**.**

fit(self, X, Y): υλοποίηση της διαδικασίας εκπαίδευσης του SVM χρησιμοποιώντας τον δυαδικό τύπο μέσω προβλήματος τετραγωνικού προγραμματισμού (QP)**.** Τα ορίσματα εισόδου είναι τα features X και οι ετικέτες Y που έχουν μετασχηματιστεί από {0,1} σε {−1,1} για να είναι συμβατές με το δυαδικό SVM.
Ορίζω τον γραμμικό πυρήνα ως $K(xi,xj)=xi\cdot xj^T$. Έπειτα ορίζω τους πίνακες για το QP:
P: Περιέχει το γινόμενο $yi\cdot yj\cdot K(xi,xj)$
q: Διανύσμα με −1 για κάθε δείγμα $(-\sum \alpha i)$
G, h: Περιορισμοί ανισότητας (για $0\leq \alpha i\leq C0$).
A, b: Περιορισμός ισότητας $(\sum \alpha i\cdot yi=0)$
Χρησιμοποιώ την μέθοδο cvxopt.solvers.qp προκειμένου να βρω τους πολλαπλασιαστές Lagrange α. Τελικά το αποτέλεσμα α που παίρνω είναι ένα διάνυσμα που καθορίζει τα support vectors. Έπειτα υπολογίζονται το διάνυσμα των βαρών w ως $w= \sum \alpha iyixi$ και το bias b ως: $b = \frac{1}{|S|}\sum_{i\in S}^{n}(yi-< w,xi >)$ (μέση τιμή των διανυσμάτων $yi-w\cdot xi$).
Να σημειωθεί ότι το πρόβλημα τετραγωνικού προγραμματισμού που καλείται να λύσει το SVM είναι ένα πρόβλημα μεγιστοποίησης, αλλά οι QP επιλυτές όπως το cvxopt μπορούν μόνο να ελαχιστοποιήσουν. Για να μετατρέψω μια μεγιστοποίηση σε ελαχιστοποίηση, αντιστρέφω το πρόσημο της συνάρτησης. Έτσι έχω: $-\sum \alpha k + 1/2\sum \alpha k \alpha l Qkl$. Σε αυτή τη μορφή: ο τετραγωνικός όρος $1/2\alpha^T P\alpha$ (όπου P=Q) παραμένει ο ίδιος., ενώ ο γραμμικός όρος $\sum \alpha k$ αποκτά αρνητικό πρόσημο. Έτσι, όταν ορίζουμε το q για τον λύτη QP: $q= -1\cdot 1$.

predict(self, X): Υπολογισμός του ορίου απόφασης $f(x)=w\cdot x-b$ για κάθε δείγμα. Επιστρέφει τις προβλεπόμενες ετικέτες ως 0 ή 1 βάσει του προσήμου της f(x) (0 για αρνητικό πρόσημο και 1 για θετικό).

Προεπεξεργασία δεδομένων:

1. Φόρτωση των δεδομένων εκπαίδευσης και δοκιμής της Cifar-10 μέσω της TensorFlow.
2. Μετατροπή των πινάκων που περιέχουν τις ετικέτες της βάσης δεδομένων από δισδιάστατους σε μονοδιάστατους (για συμβατότητα με το SVM).
3. Φιλτράρισμα των κλάσεων της Cifar-10, έτσι ώστε να έχω μόνο τις κλάσεις 0 (αεροπλάνο) και 1 (αυτοκίνητο).

4. Κανονικοποίηση των τιμών των pixel στο διάστημα [0,1].
5. Μετασχηματισμός των εικόνων από 2D πίνακες σε διανύσματα χαρακτηριστικών (flattening).
6. Τυποποίηση (standardize) των δεδομένων αφαιρώντας από αυτά την μέση τιμή τους και διαιρώντας με την τυπική τους απόκλιση έτσι ώστε να έχουν μέση τιμή 0 και διασπορά 1.
7. Μείωση των διαστάσεων των δεδομένων από 3072 σε 200 με χρήση PCA.

Εκπαίδευση και Αξιολόγηση:
Δημιουργείται ένα αντικείμενο SVM_classifier με παράμετρο C=1 και γίνεται κλήση της μεθόδου fit() για εκπαίδευση του SVM στα δεδομένα εκπαίδευσης. Στην συνέχεια γίνεται η αξιολόγηση του SVM υπολογίζοντας την ακρίβεια εκπαίδευσης και δοκιμής με την βοήθεια της μεθόδου predict(), η οποία προβλέπει τις ετικέτες για τα δεδομένα εκπαίδευσης και δοκιμής και στην συνέχεια αυτές συγκρίνονται με τις πραγματικές ετικέτες Τέλος με χρήση της μεθόδου np.mean() υπολογίζεται το ποσοστό των ορθών προβλέψεων προκειμένου να καθοριστεί η ακρίβεια ως:

$$Ακρίβεια = \frac{Σωστές\ προβλέψεις}{Σύνολο\ προβλέψεων}\ x\ 100\%$$

B. Χρήση της κλάσης SVC της scikit-learn βιβλιοθήκης, η οποία παρέχει ένα ήδη υλοποιημένο SVM που υποστηρίζει διαφορετικούς πυρήνες και υπερπαραμέτρους, για δυαδική κατηγοριοποίηση μεταξύ των κλάσεων 0 (αεροπλάνο) και 1 (αυτοκίνητο):
(γίνεται αναφορά μόνο στα σημεία του κώδικα που αλλάζουν σε σχέση με την προηγούμενη υλοποίηση)

Εισαγωγή βιβλιοθηκών:
Εκτός των ήδη προαναφερθέντων στην πρώτη υλοποίηση, γίνεται αντικατάσταση της cvxopt με την SVC της scikit-learn.

Εκπαίδευση του SVM:
Καθορίζω τιμές για τις υπερπαραμέτρους: C, η παράμετρος κανονικοποίησης που καθορίζει τον συμβιβασμό ανάμεσα στην μεγιστοποίηση του περιθωρίου (margin) και την ελαχιστοποίηση του σφάλματος κατηγοριοποίησης και gamma:
Τα βασικά ορίσματα που δέχεται η SVC είναι: C, kernel και ανάλογα τον πυρήνα μπορεί να έχω και επιπρόσθετες παραμέτρους όπως ο βαθμός του πολυωνύμου για πολυωνυμικό πυρήνα. Για την συγκεκριμένη υλοποίηση, έχω κρατήσει σταθερή την παράμετρο C=1 και έχω δοκιμάσει τους ακόλουθους τύπους πυρήνων:

a) Linear (γραμμικός)
b) Polynomial (πολυωνυμικός) διαφορετικών βαθμών
c) RBF (Radio Basis Function)
d) Sigmoid (σιγμοειδής)

a) Μορφή: $K(x,x') = x \cdot x' + c$. Οι παράμετροι είναι οι:
c (coef0): σταθερά που μπορεί να επηρεάσει το (μη γραμμικό) όριο απόφασης προκαλώντας την μετατόπισή του και προσθέτει έτσι ευελιξία στον πυρήνα.

b) Μορφή: $K(x,x') = (x \cdot x' + c)^d$. Οι παράμετροι είναι οι:
c (coef0): σταθερά που μπορεί να επηρεάσει το όριο απόφασης.
d: ο βαθμός του πολυωνύμου που ελέγχει την πολυπλοκότητα του πυρήνα.

c) Μορφή: $K(x,x') = \exp(-\gamma \|x - x'\|2)$. Η παράμετρος είναι το **γ** (στον κώδικα του fine tuning αναφέρεται ως gamma), που καθορίζει την επίδραση ενός μεμονωμένου δείγματος εκπαίδευσης στο όριο απόφασης.

d) Μορφή: $K(x,x') = \tanh(\gamma \cdot x \cdot x'^T + c)$. Οι παράμετροι είναι οι:
- γ (gamma): Ελέγχει το πόσο κάθε μεμονωμένο δείγμα εκπαίδευσης επηρεάζει το όριο απόφασης.
- c (coef0): Ελέγχει την μετατόπιση, προσθέτοντας ευελιξία στο όριο απόφασης και μπορεί να κάνει το μοντέλο περισσότερο ή λιγότερο πολύπλοκο.

Εξέταση της απόδοσης για c=1.0 για διαφορετικούς πυρήνες (Kernels):

Kernel	Training accuracy (%)	Test accuracy (%)	Runtime (sec)
Linear	82.94	82.60	851.60
Poly (2nd degree)	89.77	85.70	22.66
Poly (3rd degree-default)	93.02	86.00	36.10
Poly (4th degree)	88.84	77.60	25.66
Poly (5th degree)	90.37	78.45	43.43
Poly (10th degree)	77.87	60.95	30.91
RBF	94.10	90.10	39.69
sigmoid	67.95	68.60	26.54

Πίνακας 1

Fine tuning των παραμέτρων (C, βαθμός πολυωνύμου, σταθερά) για βέλτιστη απόδοση:
Για τον σκοπό αυτό χρησιμοποιώ την GridSearchCV της βιβλιοθήκης scikit-learn η οποία αυτοματοποιεί τη διαδικασία ρύθμισης των υπερπαραμέτρων, εκτελώντας εξαντλητική αναζήτηση σε ένα καθορισμένο χώρο υπερπαραμέτρων με χρήση cross-validation για να βρει τον συνδυασμό υπερπαραμέτρων που μεγιστοποιεί την απόδοση για το κάθε μοντέλο.
Ορίζεται ένα πλέγμα (param_grid) με τις τιμές των

υπερπαραμέτρων (τύπος πυρήνα, παράμετρος gamma, βαθμός πολυωνύμου κλπ.) στο οποίο κάθε συνδυασμός τους αποτελεί μία πιθανή διαμόρφωση του μοντέλου.
Η GridSearchCV παίρνει ως βασικά της ορίσματα τα εξής:
estimator= {το όνομα του στιγμιότυπου του SVM που έχω δημιουργήσει πριν την κλήση της μεθόδου}
param_grid={το πλέγμα που έχω ορίσει παραπάνω}
cv={αριθμός των υποσυνόλων-folds για το cross validation}.
Στην συγκεκριμένη υλοποίηση έχω επιλέξει cv=5, δηλαδή τα δεδομένα εκπαίδευσης χωρίζονται σε 5 υποσύνολα από τα οποία τελικά τα 4 χρησιμοποιούνται για εκπαίδευση και το 1 για επιβεβαίωση (validation data).
Ο συνδυασμός υπερπαραμέτρων που δίνει το υψηλότερο cross-validation score είναι αυτός που επιλέγεται ως ο βέλτιστος. Η μέθοδος grid_search.best_params_ επιστρέφει τον συνδυασμό αυτό και η grid_search.best_score_ την cross-validation ακρίβεια που του αντιστοιχεί. Τέλος υπολογίζεται και η ακρίβεια των δεδομένων ελέγχου για το SVM με τις βέλτιστες υπερπαραμέτρους.
Για τα διαφορετικά είδη πυρήνων έχω:

b)
```
param_grid = {
   'kernel': ['poly'],            # Polynomial kernel
    'degree': [2, 3, 4, 5],       # Degree of the polynomial kernel
    'coef0': [0, 1, 10],          # Constant term
}

# Create the SVM
model svm = SVC()
# Perform GridSearchCV
 grid_search = GridSearchCV(estimator=svm,
param_grid=param_grid, cv=5) grid_search.fit(x_train_pca,
y_train)
```

Αποτελέσματα:
Best Parameters: {'C': 1, 'coef0': 1, 'degree': 3, 'kernel': 'poly'}
Best Cross-Validation Accuracy: 0.89
Test Accuracy of Best Model: 90.70%

Έτσι για πολυωνυμικό πυρήνα παρατηρώ από τον πίνακα 1 ότι για βαθμό πολυωνύμου ίσο με 3, η απόδοση είναι η μέγιστη δυνατή και η περαιτέρω αύξησή του μειώνει την απόδοση, πιθανόν λόγω overfitting. Το γεγονός αυτό επιβεβαιώνει και το tuning των παραμέτρων, όπου όπως φαίνεται καλύτερη απόδοση παρουσιάζεται για πυρήνα πολυώνυμο τρίτου βαθμού, παράμετρο κανονικοποίησης του SVM C ίση με 1 και σταθερά coef0 ίση με 1.

c)
```
param_grid = {
   'kernel': ['rbf'],             # Radial basis function kernel
   'C': [0.1, 1, 10, 100],        # Regularization parameter
   'gamma': [0.001, 0.01, 0.1, 1, 10]  # Kernel coefficient
}

svm = SVC()
grid_search=GridSearchCV(estimator=svm,
param_grid=param_grid, cv=5)
grid_search.fit(x_train_pca, y_train)
```

Αποτελέσματα:
Best Parameters: {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
Best Cross-Validation Accuracy: 0.89
Test Accuracy of Best Model: 90.65%
Runtime: 1015.54 seconds

Επομένως για RBF πυρήνα η μέγιστη απόδοση επιτυγχάνεται για παράμετρο κανονικοποίησης του SVM C ίση με 10 και για παράμετρο γ ίση με 0.001.

d)
```
param_grid = {
   'kernel': ['sigmoid'],         # Sigmoid kernel
   'C': [0.1, 1, 10, 100],  # Regularization parameter
   'gamma': [0.001, 0.01, 0.1, 1, 10],  # Hyperparameter
for kernel coefficient
   'coef0': [0, 0.1, 1]  # Bias term
}
```

Αποτελέσματα:
Best Parameters: {'C': 0.1, 'coef0': 0.1, 'gamma': 0.001, 'kernel': 'sigmoid'}
Best Cross-Validation Accuracy: 0.64
Test Accuracy of Best Model: 66.25%
Runtime: 1229.02 seconds

Επομένως για Σιγμοειδή πυρήνα η απόδοση γίνεται η μέγιστη δυνατή για C ίσο με 0.1, coef0 ίσο με 0.1 και gamma ίσο με 0.001.

Κοιτώντας συνολικά την εικόνα των αποτελεσμάτων για τις αποδόσεις στους διαφορετικούς τύπους πυρήνων, παρατηρώ ότι την καλύτερη απόδοση παρουσιάζει ο Πολυωνυμικός βαθμού 3, με ακρίβεια ελέγχου που φτάνει το 90.70%, ενώ πολύ κοντά βρίσκεται και ο RBF πυρήνας με 90.65% ακρίβεια ελέγχου.
Γενικά οι βέλτιστες τιμές αποδόσεων επιτυγχάνονται για σχετικά μικρή τιμή της παραμέτρου C (που ωστόσο ποικίλλει ανάλογα με το είδος του πυρήνα) και για πολύ μικρή τιμή της παραμέτρου γ (όπου υπάρχει), ίση με 0.001. Χαμηλές τιμές για την C οδηγούν σε πιο ομαλότερο όριο απόφασης, καθώς το μοντέλο δέχεται μεγαλύτερη «ποινή» (penalty) σε κάθε λανθασμένη κατηγοριοποίηση των δεδομένων εκπαίδευσης και προσπαθεί να βρει ένα πιο απλό όριο απόφασης. Ταυτόχρονα οι χαμηλές τιμές της παραμέτρου γ, η οποία ελέγχει την κλίση της συνάρτησης του πυρήνα (Kernel function), υποδεικνύουν έναν ανθεκτικό πυρήνα, όπου η επίδραση κάθε δεδομένου εκπαίδευσης απλώνεται σε μεγαλύτερη έκταση και δεν είναι τόσο τοπική.

C. Χρήση SVM της κλάσης SVC της scikit-learn βιβλιοθήκης για κατηγοριοποίηση πολλαπλών κλάσεων (Multiclass classification):
Η τροποποίηση που κάνω στον κώδικα είναι η αφαίρεση του βήματος που φιλτράρει τις κλάσεις επιλέγοντας μόνο τις 2 πρώτες, έτσι ώστε πλέον το μοντέλο να χρησιμοποιεί και τις 10 κλάσεις της Cifar-10 βάσης δεδομένων.

Αποτελέσματα που αναμένω ως προς τα εξής:

Πολυπλοκότητα: Πιο πολύπλοκο πρόβλημα από την δυαδική κατηγοριοποίηση, οπότε αναμένω χαμηλότερη ακρίβεια αφού το μοντέλο πρέπει να διακρίνει ανάμεσα σε 10 διαφορετικές κατηγορίες κλάσεων αντί για μόνο 2.
Σφάλμα: Στην κατηγοριοποίηση πολλών κλάσεων το σφάλμα είναι πιο κατανεμημένο καθώς το μοντέλο προσπαθεί να διαχειριστεί πολλαπλές κατηγορίες, ενώ στην δυαδική κατηγοριοποίηση το σφάλμα είναι πιο περιορισμένο καθώς οι λανθασμένες κατηγοριοποιήσεις περιορίζονται μόνο σε 2 κατηγορίες.
Χρόνος εκπαίδευσης: Στην κατηγοριοποίηση πολλών κλάσεων η εκπαίδευση διαρκεί περισσότερο καθώς το SVM πρέπει βρει τα όρια απόφασης για 10 κλάσεις αντί για μόνο 2. Έτσι η χρονική πολυπλοκότητα της εκπαίδευσης αυξάνει με τον αριθμό των κλάσεων.

Εξέταση της απόδοσης για c=1.0 για διαφορετικούς πυρήνες (Kernels):

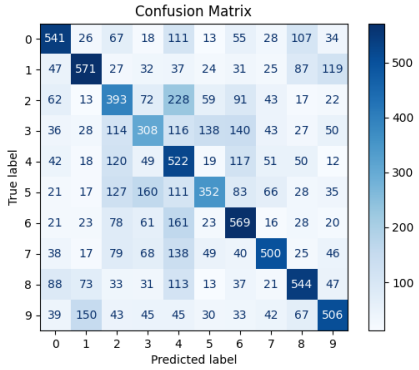
Kernel	Training accuracy (%)	Test accuracy (%)	Runtime (sec)
Linear	44.54	42.11	81452.60
Poly (2nd degree)	58.51	48.06	718.60
Poly (3rd degree)	68.26	47.14	672.79
Poly (4th degree)	63.29	38.56	922.93
Poly (5th degree)	62.95	35.48	799.92
Poly (10th degree)	51.36	22.13	854.94
RBF	68.60	54.60	480.34
sigmoid	23.42	23.97	414.29

Πίνακας 2

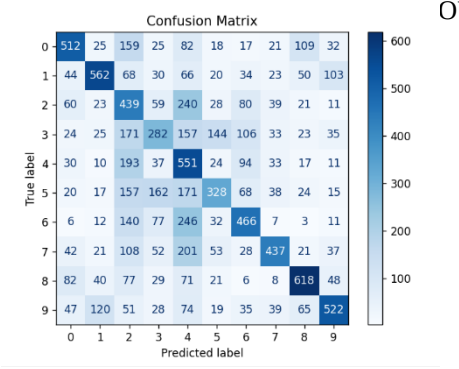
Παρατηρώ ότι η καλύτερη απόδοση δοκιμής που παίρνω είναι για RBF πυρήνα-ίση με 54.60%- ενώ ακολουθούν αυτές για πολυωνυμικό πυρήνα 2ου και 3ου βαθμού με αποδόσεις για τα δεδομένα δοκιμής ίσες με 48.06 και 47.14 αντίστοιχα. Αξίζει να σημειωθεί ότι ο χρόνος εκτέλεσης του αλγορίθμου για γραμμικό πυρήνα είναι εμφανώς πολύ μεγαλύτερος από αυτούς για όλες τις υπόλοιπες περιπτώσεις πυρήνων (περίπου 22.5 ώρες!) και με όχι τόσο καλά ποσοστά απόδοσης συγκριτικά με τους 3 προαναφερθέντες πυρήνες.

Ένα ακόμη σχόλιο που θα μπορούσε να γίνει σχετικά με τα αποτελέσματα του γραμμικού πυρήνα συγκριτικά αυτά στην περίπτωση της κατηγοριοποίησης 2 κλάσεων που παρουσίασα στις υποενότητες A και B, συγκεκριμένα των κλάσεων 0 (αεροπλάνο) και 1 (αυτοκίνητο) οι οποίες δεν μοιάζουν ιδιαίτερα μεταξύ τους. Λόγω αυτού το SVM είναι σε θέση να τις διαχωρίσει ευκολότερα από κάποιες άλλες που εμφανίζουν περισσότερα κοινά σημεία όπως για παράδειγμα οι κλάσεις 3 (γάτα) και 5 (σκύλος). Αυτό είναι εμφανές καθώς σε αυτή την περίπτωση η απόδοση του γραμμικού πυρήνα παρουσιάζεται αρκετά αυξημένη (82.90%). Αντίθετα στο πρόβλημα κατηγοριοποίησης πολλαπλών κλάσεων που παρουσιάζω στην τρέχουσα υποενότητα, η απόδοση του γραμμικού πυρήνα έχει πέσει μόλις στο 42.11%, γεγονός που κάνει αυτό το είδος του πυρήνα ακατάλληλο για τον διαχωρισμό των κλάσεων.
Τέλος προφανώς οι χρόνοι εκτέλεσης των αλγορίθμων για όλους τους διαφορετικούς πυρήνες σε αυτή την περίπτωση κατηγοριοποίησης των κλάσεων είναι πολύ μεγαλύτερος σε σχέση με αυτούς στην δυαδική κατηγοριοποίηση. Αυτό είναι απολύτως λογικό και αναμενόμενο καθώς στην δυαδική κατηγοριοποίηση το SVM χρειάζεται να υπολογίσει μόνο έναν διαχωριστικό υπερεπίπεδο που διαχωρίζει 2 κλάσεις. Επίσης η επίλυση του προβλήματος βελτιστοποίησης είναι απλούστερη και ταχύτερη αφού αφορά λιγότερους περιορισμούς και μικρότερο αριθμό μεταβλητών. Όσον αφορά την κατηγοριοποίηση πολλαπλών κλάσεων, αυτή χρησιμοποιεί είτε την One-vs-Rest προσέγγιση, όπου δημιουργεί ένα SVM για κάθε κλάση, όπου κάθε μοντέλο διαχωρίζει τη συγκεκριμένη κλάση από όλες τις υπόλοιπες. Έτσι στην cifar-10 απαιτούνται 10 SVM μοντέλα, που συνεπάγεται πολύ περισσότερους υπολογισμούς τόσο για την εκπαίδευση όσο και για την πρόβλεψη. Η άλλη εναλλακτική προσέγγιση που μπορεί να χρησιμοποιηθεί είναι η One-vs-One, η οποία δημιουργεί ένα SVM για κάθε ζεύγος κλάσεων και συνεπώς στην cifar-10 όπου υπάρχουν $10\times(10-1)/2=45$ ζεύγη κλάσεων, απαιτούνται 45 SVM μοντέλα. Αυτό αυξάνει εκθετικά την πολυπλοκότητα και τον χρόνο εκτέλεσης.

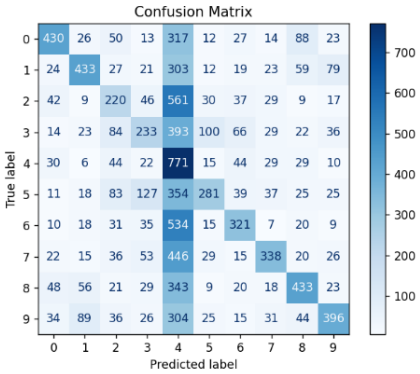
Παραδείγματα ορθής και λανθασμένης κατηγοριοποίησης των κλάσεων, όπως αυτά φαίνονται μέσω των Confusion Matrices, για τα διαφορετικά είδη πυρήνων:



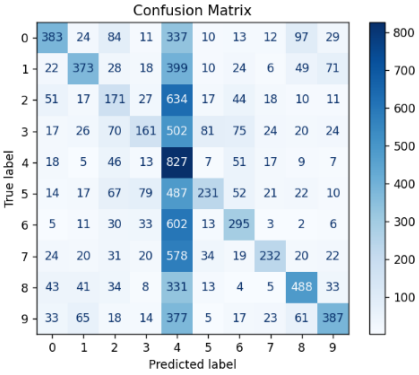
2nd degree Polynomial Kernel



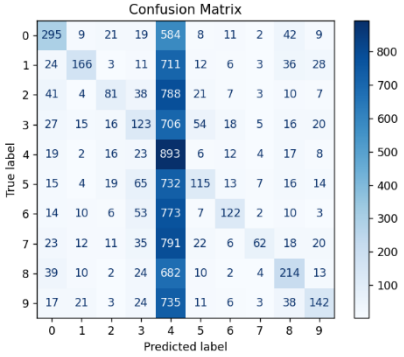
3rd degree Polynomial Kernel



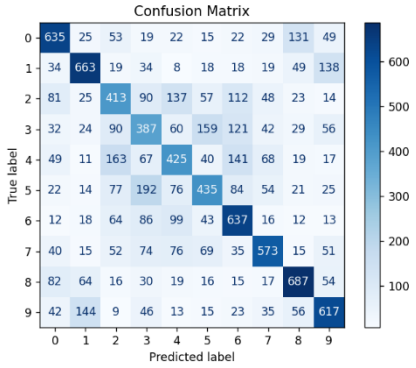
4th degree Polynomial Kernel



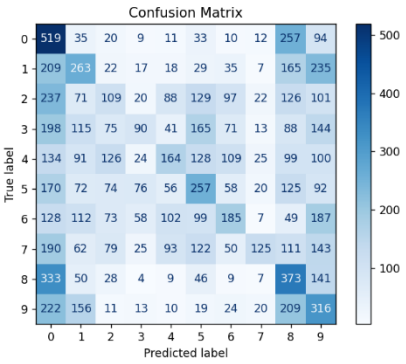
5th degree Polynomial Kernel



10th degree Polynomial Kernel



RBF Kernel



Sigmoid Kernel

Από τα παραπάνω Confusion Matrices επιβεβαιώνεται το γεγονός ότι για πολυωνυμικό πυρήνα 2ου και 3ου βαθμού, καθώς και για RBF πυρήνα έχω τις περισσότερες ορθές κατηγοριοποιήσεις, ενώ τα μοντέλα με το χειρότερο test accuracy είναι αυτά του πολυωνυμικού πυρήνα 10ου βαθμού (πολύ μεγάλος) και του σιγμοειδή.

III. ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΜΕ ΠΡΟΗΓΟΥΜΕΝΕΣ ΥΛΟΠΟΙΗΣΕΙΣ

Σύγκριση SVM με MLP με ένα κρυφό επίπεδο που χρησιμοποιεί Hinge loss για την βελτιστοποίηση:

Το MLP που είχα υλοποιήσει στα πλαίσια της 1ης εργασίας χρησιμοποιεί ως συνάρτηση κόστους την cross-entropy loss function.

Για ένα δείγμα το Hinge Loss ορίζεται ως: $L(y,f(x)) = \max(0,1-y*f(x))$, όπου το $y*f(x)$ μετράει τις παραβιάσεις του margin για την πρόβλεψη f(x). Αν αυτό είναι ≥ 1, η πρόβλεψη είναι ορθή και βρίσκεται στην σωστή πλευρά του περιθωρίου, με αποτέλεσμα μηδενικό loss. Αντίθετα αν είναι <1, η πρόβλεψη είτε είναι λανθασμένη είτε βρίσκεται στην λάθος πλευρά του περιθωρίου, οδηγώντας σε θετικό loss, ανάλογο της απόστασης από το margin.

- Για την υλοποίηση ενός MLP με τροποποίηση της συνάρτησης κόστους έτσι ώστε να χρησιμοποιεί την Hinge

Loss, για την κατηγοριοποίηση όλων των κλάσεων της cifar-10 πήρα:
Epoch 300/300: Loss = 2.0490, Train Accuracy = 57.85%, Test Accuracy = 48.80%.
Για την SVM υλοποίηση, η μέγιστη απόδοση ήταν για πολυωνυμικό τρίτου βαθμού πυρήνα, ίση με 90.70% δηλαδή πολύ μεγαλύτερη σε σχέση με αυτή που παίρνω από το MLP.

- Για την υλοποίηση ενός MLP με τροποποίηση της συνάρτησης κόστους έτσι ώστε να χρησιμοποιεί την Hinge Loss, για την κατηγοριοποίηση όλων των κλάσεων της cifar-10 πήρα:
  Epoch 300/300: Loss = 2.0512, Train Accuracy = 57.65%, Test Accuracy = 48.69%
  Για την SVM υλοποίηση, η μέγιστη απόδοση ήταν για RBF πυρήνα, ίση με 54.60 %. (για παράμετρο C=1).

Σύγκριση SVM με κατηγοριοποίηση 1 και 3 πλησιέστερου γείτονα (Nearest Neighbor) και πλησιέστερου κέντρου κλάσης (Nearest Class Centroid):

Για τις αποδόσεις των παραπάνω αλγορίθμων, οι οποίοι μελετήθηκαν στην ενδιάμεση εργασία, είχα τα εξής αποτελέσματα:
Nearest Class Centroid: 27.74%
3-NearestNeighbor: 33.27%
1-NearestNeighbor: 35.39%
Συγκριτικά με την βέλτιστη απόδοση που πήρα με το SVM, αυτή ήταν 54.60% για RBF πυρήνα (εξετάστηκε μόνο για σταθερή παράμετρο κανονικοποίησης C και ίση με 1, που ωστόσο δεν αποτελεί την βέλτιστη τιμή της, με τις βέλτιστες παραμέτρους της η απόδοση θα ήταν ακόμη υψηλότερη). Συνεπώς όπως αναμένονταν η απόδοση του SVM είναι υψηλότερη από τους 3 παραπάνω αλγορίθμους κατηγοριοποίησης.