Νευρωνικά Δίκτυα-Βαθιά Μάθηση
Δεύτερη Εργασία

Ονοματεπώνυμο: Αθανάσιος Γιαπουτζής

ΑΕΜ: 3589

Η παρακάτω έκθεση είναι χωρισμένη σε δύο μέρη. Στο πρώτο κομμάτι γίνεται συνοπτική ανάλυση της βάσης *MNIST* η οποία χρησιμοποιήθηκε για το testing και training των αλγορίθμων/SVM ενώ στο δεύτερο κομμάτι αναλύεται εκτενώς το υλοποιημένο SVM και συγκρίνεται με τους δύο κατηγοριοποιητές.

1 MNIST

Όπως προαναφέρθηκε, για την εκπαίδευση και το testing του δικτύου χρησιμοποιήθηκε η βάση MNIST. Αποτελείται από 60000 εικόνες ψηφίων για εκπαίδευση και 10000 για testing. Η αρχική αναπαράστατη τους γίνεται με μορφή τρισδιάστου πίνακα, δηλαδή με πίνακα 60000 θέσεων ο οποίος περιέχει δισδιάστατους πίνακες διαστάσεων 28x28. Κάθε κελί των παραπάνω δισδιάστατων πινάκων περιέχει τιμές από το 0 εως το 255, με το 0 να δηλώνει το μαύρο και το 255 το άσπρο.

2 SVM Analysis

Για την υλοποίηση του SVM χρησιμοποιήθηκε η βιβλιοθήκη sklearn.
Τα δεδομένα της βάσης φορτώνονται και τροποποιούνται κατάλληλα ώστε να γίνουν επεξεργάσιμα απο το δίκτυο, μετατρέπονται δηλαδή σε διανύσματα διάστασης 1x784. Στην συνέχεια κανονικοποιούμε τις τιμές. Τέλος χωρίζεται το training data set σε validation-training
Η υλοποίηση των παραπάνω με κώδικα είναι η εξής:

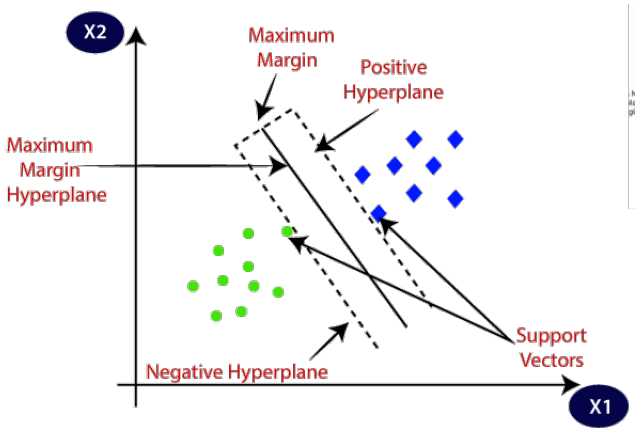
```
    #Load Data
    (train_images, train_labels), (test_images, test_labels) =
    mnist.load_data()

    #Reshaping data
    train_images = np.reshape(train_images, (-1, 784))
    test_images = np.reshape(test_images, (-1, 784))

    #Normalizing to avoid high gradient values
    train_images = train_images.astype('float32') / 255
```

```
10      test_images = test_images.astype('float32') / 255
11
12      #Split train and test for evaluating purposes
13      X_train, X_eval, y_train, y_eval = train_test_split(
        train_images, train_labels, train_size=0.7, stratify=
        train_labels)
```

Η σωστή επιλογή μοντέλου έγινε έπειτα από εκτενή χρήση cross validation για την κατάλληλη επιλογή των παραμέτρων(kernel, c parameter, gamma parameter).

Συνοπτική ανάλυση του αλγορίθμου

Τα SVM(Support Vector Machine) αποτελούν έναν αλγόριθμο της μηχανικής μάθησης ο οποίος χρησιμοποιείται για την επίλυση classification και regression προβλημάτων. Στόχος του αλγορίθμου είναι η εύρεση ενός υπερεπιπέδου σε έναν ν-διάστατο χώρο στον οποίο θα γίνεται σωστός διαχωρισμός των δειγμάτων σε κλάσεις. Ο SVM διαλέγει τα πιο ακριανά σημεία της κάθε κλάσης τα οποία βοηθούν στην δημιουργία του υπερεπιπέδου. Αυτά τα σημεία ονομάζονται Support Vectors.

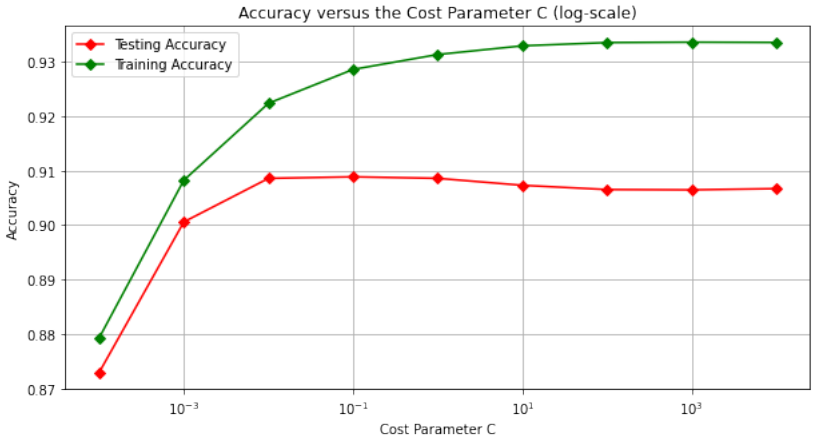


Επιλογή της παραμέτρου C

Η παράμετρος C καθορίζει το πόσο καλό θα είναι το μοντέλο στον διαχωρισμό των κλάσεων. Για μεγάλες τιμές της παραμέτρου ο αλγόριθμος θα επιλέξει το μικρότερο υπερεπιπέδο έτσι ώστε να γίνεται σωστά ο διαχωρισμός των δεδομένων. Αντίθετα ένα μικρό C οδηγεί στην επιλογή ενός μεγαλύτερου υπερεπιπέδου ακόμα και αν αυτό το υπερεπιπέδο κάνει λανθασμένες επιλογές στο classification των δεδομένων.
Για την σωστή επιλογή της παραμέτρου υλοποιήθηκε ένας επαναληπτικός βρόγχος στον οποίο δοκιμάζονταν διαφορετικές τιμές του C, χρησιμοποιώντας γραμμικό SVM. Η υλοποίηση των παραπάνω με κώδικα είναι η εξής:

```
for c in [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]:
  svm = SVC(kernel='linear', C=c)
  svm.fit(X_train, y_train)
  coef = svm.coef_

  acc_tr = accuracy_score(y_train, svm.predict(X_train))

  acc_pred = accuracy_score(y_eval, svm.predict(X_eval))

  coefficient.append(coef)
  training_acc.append(acc_tr)
  test_acc.append(acc_pred)
```

Στην συνέχεια κάνοντας plot τα αποτελέσματα παρατηρούμε ότι για τιμές μεγαλύτερες του 10 το μοντέλο αρχίζει και υπερεκπαιδεύεται.



Επομενώς επιλέγουμε $c = 1$ διότι εκεί το μοντέλο πετυχαίνει το καλύτερο testing accuracy.

Επιλογή της παραμέτρου Gamma

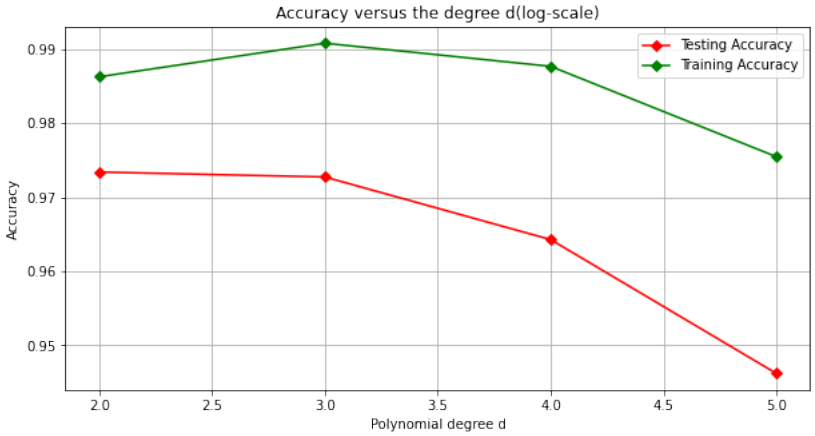
Στην συνέχεια επιλέγουμε την παράμετρο gamma πραγματοποιόντας την ίδια διαδικασία με πριν. Δημιουργούμε έναν επαναληπτικό βρόγχο στον οποίο δοκιμάζονται διαφορετικές τιμές της παραμέτρου, ενώ κρατάμε αποτελέσματα για training/testing accuracy τα οποία φαίνονται στον παρακάτων πίνακα. Επιπλέον η παράμετρος gamma έχει νόημα μόνο για SVMs τα οποία χρησιμοποιούν RBF kernel.

C	Gamma	Train	Test
1	0.01	0.977	0.951
1	0.1	1.000	0.868
1	1	1.000	0.157
1	10	1.000	0.133
1	100	1.000	0.133

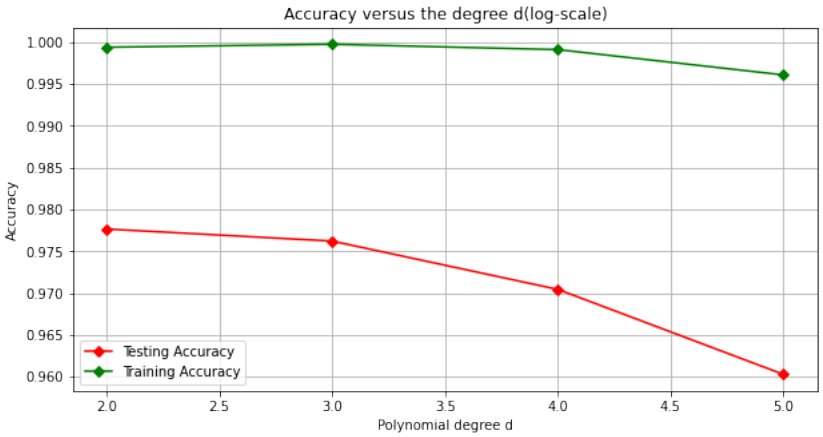
Παρατηρώντας τον πίνακα, είναι εύκολο να αντιληφθούμε ότι ναι μεν αυξάνοντας την τιμή της παραμέτρου πετυχαίνουμε καλύτερα αποτελέσματα στην εκπαίδευση, αλλά η ακρίβεια στο test είναι απογοητευτικά χαμηλή μετά από ένα σημείο. Επιλέγουμε άρα, $Gamma = 0.1$

Επιλογή kernel

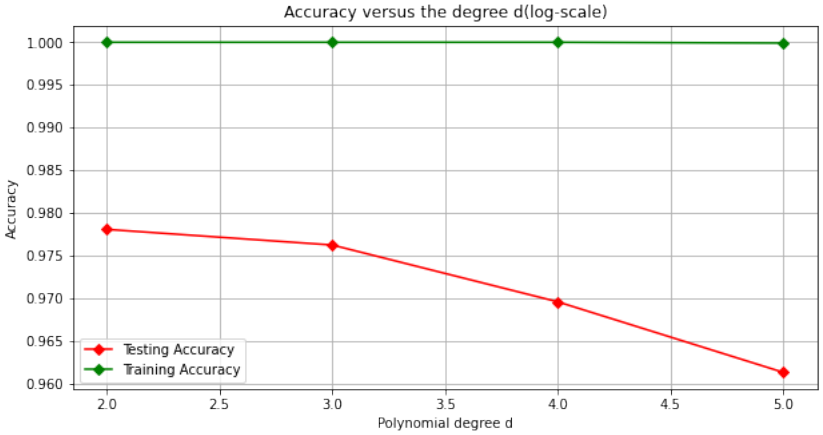
Οι επιλογές των δύο προηγούμενων παραμέτρων έγιναν με την χρήση γραμμικού SVM για την παράμετρο C, και με RBF kernel για την παραμέτρο $Gamma$. Επόμενο kernel είναι το πολυωνυμικό, το οποίο έδωσε τα εξής αποτελέσματα για διαφορετικούς βαθμούς του πολυωνυμου.



Παρατηρούμε ότι το μοντέλο δίνει τα καλύτερα αποτελέσματα για βαθμό πολυωνυμου ίσο με 2. Αυξάνοντας τον βαθμό βελτιώνουμε την ακρίβεια της εκπαίδευσης, αλλά ταυτόχρονα το μοντέλο υπερεκπαιδεύεται οδηγώντας σε κακά αποτελέσματα στο testing.
Για $C = 10$ έχουμε:



Ενώ θα περιμέναμε η αύξηση του βαθμού να δώσει καλύτερα αποτελέσματα στο testing κάτι τέτοιο δεν συμβάινει καθώς το μοντέλο υπερεκπαιδεύεται.
Τέλος δοκιμάζουμε για $C = 100$:



Το μοντέλο παπαγαλίζει πολύ γρήγορα οδηγώντας σε πολύ κακά αποτελέσματα στο testing.
Καταλήγουμε άρα στο ότι το μοντέλο δίνει τα καλύτερα αποτελέσματα όταν ο βαθμός του πολυωνύμου ισούται με 2 και το C είναι ίσο με 1.

Παρουσίαση αποτελεσμάτων - Επιλογή μοντέλου

Συνοψίζοντας:

Kernel	Gamma	C	Degree	Train	Test
Linear	-	1	-	0.928	0.909
RBF	1	0.01	-	0.977	0.951
Poly	-	1	2	0.988	0.973

Σύμφωνα με τον παραπάνω πίνακα επιλέγουμε το πολυωνυμικό kernel δευτέρου βαθμού με $C = 1$.
Σειρά έχει η εκπαίδευση του μοντέλου και με τα 60000 δείγματα.
Το μοντέλο πετυχαίνει ακρίβεια:

```
Training Accuracy: 0.9875
```

Ενώ στο testing πετυχαίνει ακρίβεια:

```
Testing Accuracy: 0.9774
```

Επιπλέον μπορούμε να εξάγουμε και το classification report του μοντέλου το όποιο θα μας δώσει μια καλύτερη εικόνα ως προς το πως τα πηγαίνει.

	precision	recall	f1-score	support
0	0.98	0.99	0.99	980
1	0.98	0.99	0.99	1135
2	0.98	0.98	0.98	1032
3	0.98	0.98	0.98	1010
4	0.97	0.98	0.98	982
5	0.98	0.97	0.98	892
6	0.98	0.98	0.98	958
7	0.97	0.97	0.97	1028
8	0.97	0.97	0.97	974
9	0.97	0.96	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Σύγκριση SVM με τους κατηγοριοποιητές

Τα τρία παρακάτω screenshots αποτελούν τα classification reports των τριών κατηγοριοποιητών.
K-Nearest Neigbors(1):

	precision	recall	f1-score	support
0	0.98	0.99	0.99	980
1	0.97	0.99	0.98	1135
2	0.98	0.96	0.97	1032
3	0.96	0.96	0.96	1010
4	0.97	0.96	0.97	982
5	0.95	0.96	0.96	892
6	0.98	0.99	0.98	958
7	0.96	0.96	0.96	1028
8	0.98	0.94	0.96	974
9	0.96	0.96	0.96	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

K-Nearest Neigbors(3):

	precision	recall	f1-score	support
0	0.97	0.99	0.98	980
1	0.96	1.00	0.98	1135
2	0.98	0.97	0.97	1032
3	0.96	0.97	0.96	1010
4	0.98	0.97	0.97	982
5	0.97	0.96	0.96	892
6	0.98	0.99	0.98	958
7	0.96	0.96	0.96	1028
8	0.99	0.94	0.96	974
9	0.96	0.96	0.96	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

Nearest Centroid:

	precision	recall	f1-score	support
0	0.91	0.90	0.90	980
1	0.77	0.96	0.86	1135
2	0.88	0.76	0.81	1032
3	0.77	0.81	0.78	1010
4	0.80	0.83	0.81	982
5	0.75	0.69	0.72	892
6	0.88	0.86	0.87	958
7	0.91	0.83	0.87	1028
8	0.79	0.74	0.76	974
9	0.77	0.81	0.79	1009
accuracy			0.82	10000
macro avg	0.82	0.82	0.82	10000
weighted avg	0.82	0.82	0.82	10000

Ο αλγόριθμος KNN για 1 και 3 γείτονες έδωσε πολύ καλά αποτελέσματα φτάνοντας στο 97% accuracy, ενώ ο NC ήταν αρκετά χειρότερος με μόλις 82%. Καταλήγουμε άρα στο ότι το SVM αποτελεί την καλύτερη λύση για την επίλυση του συγκεκριμένου προβλήματος classification.

3 Πηγές

https://en.wikipedia.org/wiki/Support_vector_machine

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