Νευρωνικά Δίκτυα

Εργασία 2

SVM - binary classification

Ηλιάνα Κόγια
(AEM: 10090)
ilianakogia@ece.auth.gr

1 SVM - binary classification

1.1 Dual Problem Soft Margin

Quadratic Programming Problem (Convex) - Kernel trick

$$\begin{aligned} \max_a Q(a) = \quad & \sum_{j=1}^{m} \alpha_j - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j K(x_i, x_j) \alpha_i \alpha_j \\ \text{subject to} \quad & 0 \leq a_i \leq C, \forall i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

Χρησιμοποιούμε την συνάρτηση τετραγωνικού προγραμματισμού solvers.qp της python της βιβλιοθήκης cvxopt και η αντιστοιχία για το συγκεκριμένο πρόβλημα είναι η παρακάτω:
Η μορφή του προβλήματος που δέχεται η cvxopt είναι η ακόλουθη:

Standard QP cvxopt

$$\min_x \frac{1}{2} \mathbf{x}^T \mathbf{P} \mathbf{x} + \mathbf{q}^T \mathbf{x} \tag{1.1}$$

$$\text{subject to } \mathbf{G}\mathbf{x} \leq \mathbf{h} \tag{1.2}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1.3}$$

Soft Margin dual SVM problem:

$$\min_\alpha \frac{1}{2} \alpha^T \mathbf{Q} \alpha - \mathbf{1}^T \alpha \tag{1.4}$$

$$\text{subject to } -\alpha_\mathbf{i} \leq \mathbf{0} \tag{1.5}$$

$$\alpha_\mathbf{i} \leq \mathbf{C} \tag{1.6}$$

$$\mathbf{y^T} \alpha = \mathbf{0} \tag{1.7}$$

με

$$P = Q = yy^T K \tag{1.8}$$

Ο τύπος που χρησιμοποιείται για το bias υπολογίζεται πάνω στα support vector τα οποία προκύπτουν για όλους τους θετικούς συντελεστές Lagrange alpha, δηλαδή είναι:

$$b = \frac{1}{Ns} \sum_{neS} (y_n - \sum_{meS} \alpha_m y_m K(x_m, x_n)) \tag{1.9}$$

1.2 Training

Δοκιμάστηκαν διάφοροι συνδυασμοί των εκάστοτε παραμέτρων για τα Kernel: linear, rbf, polynomial:

Linear Kernel:

$$K(x,y) = x^T y \tag{1.10}$$

RBF Kernel:

$$K(x,y) = \exp(-\gamma \|x-y\|^2) \tag{1.11}$$

Polynomial Kernel:

$$K(x,y) = (x^T y + c)^d \tag{1.12}$$

Σημείωση:

Επιλέχθηκαν για classification οι κλάσεις airplane και automobile της Cifar-10

Τα δεδομένα εκπαίδευσης κανονικοποιήθηκαν στο [-1,1] και οι πίνακες Gram των Kernel υλοποιήθηκαν με την metrics.pairwise της Python για λόγους ταχύτητας.

1.3 Linear Kernel results

C	Train accuracy	Test accuracy	Support Vectors	Time (s)
0.001	0.832	0.824	4635	660
0.01	0.8536	**0.824**	4325	607
0.1	0.8818	0.811	4123	575
0.5	0.8987	0.795	3940	562
1	0.9056	0.7915	3858	552

Παρατηρούμε ότι όσο αυξάνεται η παράμετρος C ο αριθμός των support vector μειώνεται. Επίσης, αυξάνεται η ακρίβεια στο train set ενώ μειώνεται στο test set, δηλαδή έχουμε overfitting όσο μεγαλώνει το C. Όπως είναι αναμενόμενο ο χρόνος είναι μεγαλύτερος όταν έχουμε μεγαλύτερο αριθμό sv, διότι οι υπολογισμοί για το bias και τον προβλέψεων y_{pred} που γίνονται στα support vector και όχι σε όλο το training set είναι περισσότεροι.

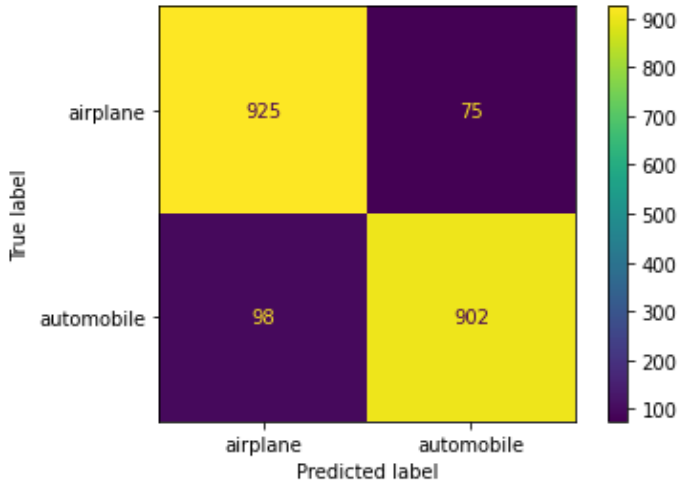
1.4 RBF Kernel results

gamma	C	Train accuracy	Test accuracy	Support Vectors	Time (s)
0.0004	0.1	0.8326	0.844	5574	1477
	1	0.893	0.879	4285	1078
	10	0.9639	**0.9135**	**3681**	933
	100	0.9997	0.907	3720	974
	1000	1.0	0.903	3689	1041

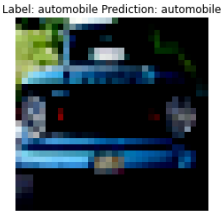
gamma	C	Train accuracy	Test accuracy	Support Vectors	Time (s)
0.004	0.1	0.8496	0.8325	7444	2049
	1	0.9929	**0.892**	**6819**	1837
	10	1.0	0.8975	7199	1995
	100	1.0	0.8975	7228	2022
	1000	1.0	0.8975	7191	2072

Για τιμή του γ επιλέγεται αρχικά η τιμή 0.0004 που βρίσκεται εντός των 3/10000 και 6/10000 (numOfsumples = 10000) και παρατηρούμε ότι πετυχαίνουμε σχετικα καλή ακρίβεια, ενώ έπειτα αυξάνοντας το γ κατα μία ταξη για ίδιες τιμές του C προκύπτει μικρότερη ακρίβεια.
Για σταθερό γ διαπιστώνουμε ότι όσο αυξάνεται το C ο αριθμός των sv μπορεί είτε να μειώνεται είτε να αυξάνεται
Για γ = 0.004 και C >10 δεν παρατηρείται διαφορά στην ακρίβεια και στον αριθμό των sv.

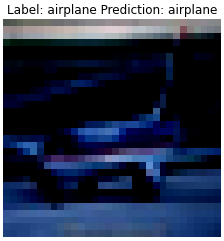
Το μοντέλο που επιτυγχάνει καλύτερη ακρίβεια **0.9135** και ταυτόχρονα μια ανεκτή τιμή gap μεταξύ train και test data (καλό generalization), αλλά και αριθμό sv που δεν υπερβαίνει το 50% των training δειγμάτων(=10,000) είναι για τις παραμέτρους γ **= 0.0004** και **C = 10**. Ο αντίστοιχος confusion matrix:



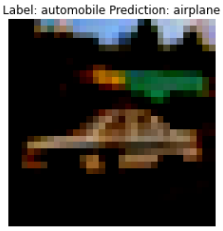
Παρατίθενται ακόμα κάποια παραδείγματα σωστής και εσφαλμένης ταξινόμησης:
automobile true:



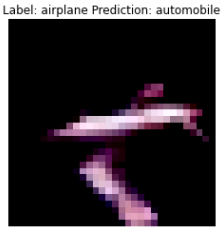
airplane true:



automobile but predicted airplane:



airplane but predicted automobile



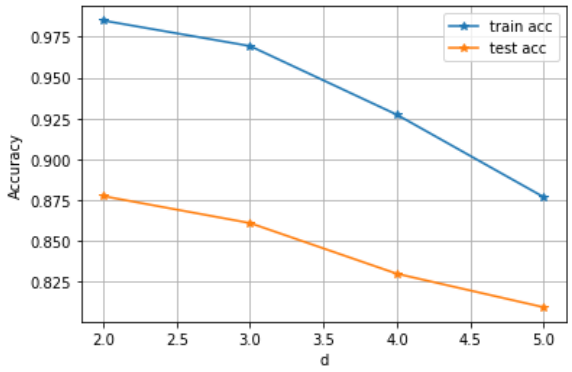
automobile but predicted airplane:

1.5 Polynomial Kernel results

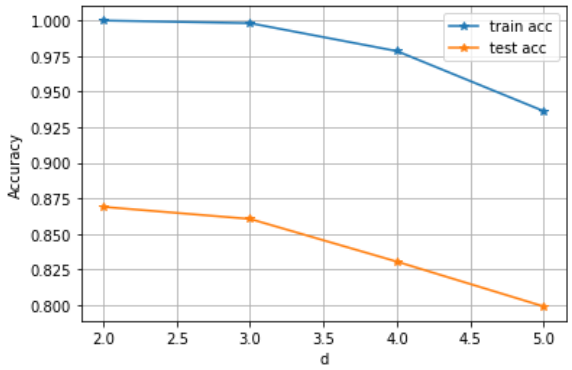
C	coef	gamma	degree	Train accuracy	Test accuracy	Support Vectors	Time (s)
100	0	1/3072	2	0.9849	**0.8775**	4319	772
			3	0.9693	0.861	5202	973
			4	0.9273	0.83	7120	1270
			5	0.8769	0.8095	7873	1413
C	**coef**	**gamma**	**degree**	**Train accuracy**	**Test accuracy**	**Support Vectors**	**Time (s)**
1000	0	1/3072	2	1.0	**0.869**	4138	777
			3	0.9981	0.8605	4541	855
			4	0.9785	0.8305	5727	1012
			5	0.9363	0.799	6504	1181
C	**coef**	**gamma**	**degree**	**Train accuracy**	**Test accuracy**	**Support Vectors**	**Time (s)**
10000	0	1/3072	2	1	**0.87**	4212	862
			3	1	0.859	4463	980.
			4	0.9979	0.8355	6526	1238
			5	0.9812	0.798	8436	1672

Παρατηρούμε ότι όσο αυξάνεται ο βαθμός του polynomial kernel και για σταθερό C εχουμε αρκετά μεγάλο αριθμό sv >50% του training set για d = 4 ή 5
Επιπλέον παρατίθενται τα διαγράμματα των accuracy συναρτήσει των βαθμών d για σταθερο C

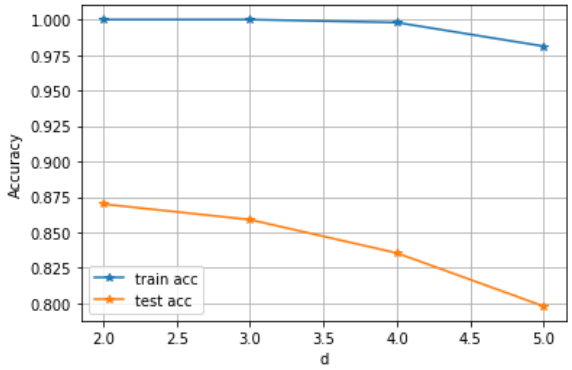
C = 100



C = 1000



C = 10000



Σημειώνεται ότι μικρότερο overfitting έχουμε για C = 100.

Παρατηρήση:

Μπορούμε να επιτύχουμε παρόμοιες τιμές accuracy για διαφορετικά Kernel και για διαφορετικές παραμέτρους ωστόσο διαφέρει ο αριθμός των support vector και έτσι η πολυπλοκότητα του μοντέλου.

2 Compare SVM - kNN - NearestCentroid

Εξετάστηκε το ίδιο πρόβλημα classification (airplane/automobile) και είχε τα ακόλουθα αποτελέσματα για το test set:
Accuracy:

kNN		Nearest Centroid
k = 1	k = 3	72.30
71.5	69.45	

Το καλύτερο μοντέλο που επιτύχαμε όσον αφορά το accuracy με το SVM ήταν για το test set 91.35 % με χρήση RBF Kernel και παραμέτρους $\gamma = 0.0004$ και $C = 10$.
Διαπιστώνουμε ότι με το νευρωνικό επιτυγχάνεται φανερά καλύτερη απόδοση, σε σχέση με τους παραπάνω δύο αλγορίθμους, ωστόσο με μεγαλύτερη πολυπλοκότητα.