

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ
Χειμερινό Εξάμηνο 2020-21

Προπαρασκευή 1ης Εργαστηριακής Άσκησης:
Οπτική Αναγνώριση Ψηφίων

ΔΙΑΔΙΚΑΣΤΙΚΑ ΕΡΓΑΣΤΗΡΙΟΥ

Εξηγήστε περιεκτικά και επαρκώς την εργασία σας. Κώδικας χωρίς σχόλια δεν θα βαθμολογηθεί. Επιτρέπεται η συνεργασία εντός ομάδων των 2 ατόμων εφόσον φοιτούν στο ίδιο πρόγραμμα σπουδών (είτε ομάδες προπτυχιακών, είτε ομάδες μεταπτυχιακών). Κάθε ομάδα 2 ατόμων υποβάλλει μια κοινή αναφορά που αντιπροσωπεύει μόνο την προσωπική εργασία των μελών της. Αν χρησιμοποιήσετε κάποια άλλη πηγή εκτός των βιβλίων και του εκπαιδευτικού υλικού του μαθήματος, πρέπει να το αναφέρετε. Η παράδοση της αναφοράς και του κώδικα της εργασίας θα γίνει ηλεκτρονικά στο [moodle](#). **Επισημαίνεται ότι απαγορεύεται η ανάρτηση των λύσεων των εργαστηριακών ασκήσεων στο github, ή σε άλλες ιστοσελίδες.**

ΒΟΗΘΗΤΙΚΟ ΥΛΙΚΟ

Στην ακόλουθη [σελίδα](#) μπορείτε να βρείτε βοηθητικό κώδικα σχετικά με τα εργαστήρια. Όπως θα δείτε σας παρέχονται μη-υλοποιημένες συναρτήσεις τις οποίες καλείστε να συμπληρώσετε. Σημειώνουμε πως η υποβολή του αρχείου αυτού (lib.py) **με συμπληρωμένες τις δικές σας υλοποιήσεις είναι απαραίτητη για τη βαθμολόγηση της εργασίας σας**. Στη σελίδα [αυτή](#) μπορείτε επίσης να υποβάλετε απορίες και ερωτήσεις προς τους βοηθούς του μαθήματος με μορφή issues. Ερωτήσεις αναφορικά με το εργαστήριο που θα γίνονται μέσω mail δεν θα λαμβάνουν απάντηση.

ΠΕΡΙΓΡΑΦΗ

Σκοπός είναι η υλοποίηση ενός συστήματος οπτικής αναγνώρισης ψηφίων. Τα δεδομένα προέρχονται από την US Postal Service (γραμμένα στο χέρι σε ταχυδρομικούς φακέλους και σκαναρισμένα) και περιέχουν τα ψηφία από το 0 έως το 9 και διακρίνονται σε train και test.

Τα δεδομένα κάθε αρχείου αναπαριστούν τα περιεχόμενα ενός πίνακα (οι τιμές των στοιχείων του πίνακα διαχωρίζονται με κενό). Κάθε γραμμή αφορά ένα ψηφίο (δείγμα). Οι στήλες αντιστοιχούν στα χαρακτηριστικά (features) που περιγράφουν τα ψηφία. Για παράδειγμα, η τιμή του (i,j) στοιχείου αφορά το j -th χαρακτηριστικό του i -th ψηφίου. Κάθε ψηφίο περιγράφεται από 257 τιμές, εκ των οποίων η πρώτη αντιστοιχεί στο ίδιο το ψηφίο (αν είναι το 0, το 1 κτλ.) και οι υπόλοιπες 256 είναι τα χαρακτηριστικά (features) που το περιγράφουν (grayscale values). Ας φανταστούμε το κάθε ψηφίο να απεικονίζεται σε έναν 16×16 πίνακα αποτελούμενο από 256 κουτάκια ("pixels"). Για να εμφανίζεται το κάθε ψηφίο στην οθόνη "φωτίζεται" ένα σύνολο από τέτοια κουτάκια, με τέτοιο τρόπο ώστε η συνολική εικόνα που βλέπουμε να απεικονίζει το θεωρούμενο ψηφίο. Επειδή τα ψηφία εμφανίζονται σε grayscale, κάθε μία από τις 256 τιμές αντιστοιχεί σε μία απόχρωση μαύρου για το αντίστοιχο "pixel". Στόχος είναι η δημιουργία και αποτίμηση (evaluation) ταξινομητών οι οποίοι θα ταξινομούν κάθε ένα από τα ψηφία που περιλαμβάνονται στα test δεδομένα σε μία από τις δέκα κατηγορίες (από το 0 έως το 9).

ΕΚΤΕΛΕΣΗ

Κατεβάστε τα δεδομένα από τις διευκρινίσεις του mycourses.
Χρησιμοποιώντας Python εκτελέστε τα παρακάτω βήματα:

Βήμα 1

Διαβάστε τα δεδομένα από το αρχείο. Τα δεδομένα πρέπει να διαβαστούν σε μορφή συμβατή με το scikit-learn σε 4 πίνακες X_{train} , X_{test} , y_{train} και y_{test} . Ο πίνακας X_{train} περιέχει τα δείγματα εκπαίδευσης (χωρίς τα labels) και είναι διάστασης ($n_{samples_train} \times n_{features}$). Ο y_{train} είναι ένας μονοδιάστατος πίνακας μήκους $n_{samples}$ και περιέχει τα αντίστοιχα labels για τον X_{train} . Αντίστοιχα για τα test δεδομένα.

Βήμα 2

Σχεδιάστε το υπ' αριθμόν 131 ψηφίο, (βρίσκεται στη θέση 131) των train δεδομένων. Υπόδειξη: χρησιμοποιήστε τη συνάρτηση `numpy.reshape` για να οργανώσετε τα 256 χαρακτηριστικά σε ένα πίνακα 16×16 , και τη συνάρτηση `matplotlib.pyplot.imshow` για την απεικόνιση του ψηφίου.

Βήμα 3

Διαλέξτε 1 τυχαίο δείγμα από κάθε label (συνολικά 10 δείγματα). Σχεδιάστε τα σε ένα figure με subplots. (Hint: `fig = plt.figure(); fig.add_subplot(,,)`)

Βήμα 4

Υπολογίστε τη μέση τιμή των χαρακτηριστικών του pixel (10, 10) για το ψηφίο «0» με βάση τα train δεδομένα.

Βήμα 5

Υπολογίστε τη διασπορά των χαρακτηριστικών του pixel (10, 10) για το ψηφίο «0» με βάση τα train δεδομένα.

Βήμα 6

Υπολογίστε τη μέση τιμή και διασπορά των χαρακτηριστικών κάθε pixel για το ψηφίο «0» με βάση τα train δεδομένα.

Βήμα 7

Σχεδιάστε το ψηφίο «0» χρησιμοποιώντας τις τιμές της μέσης τιμής που υπολογίσατε στο Βήμα 6.

Βήμα 8

Σχεδιάστε το ψηφίο «0» χρησιμοποιώντας τις τιμές της διασποράς που υπολογίσατε στο Βήμα 6. Συγκρίνετε το αποτέλεσμα με το αποτέλεσμα του Βήματος 7 και εξηγήστε τυχόν διαφορές.

Βήμα 9

- (α) Υπολογίστε τη μέση τιμή και διασπορά των χαρακτηριστικών για όλα τα ψηφία (0-9) με βάση τα train δεδομένα.
(β) Σχεδιάστε όλα τα ψηφία χρησιμοποιώντας τις τιμές της μέσης τιμής που υπολογίσατε στο Βήμα 9(α).

Βήμα 10

Ταξινομήστε το υπ' αριθμόν 101 ψηφίο των test δεδομένων (βρίσκεται στη θέση 101) σε μία από τις 10 κατηγορίες (κάθε ένα από τα 10 ψηφία, 0-9, αντιπροσωπεύει μία κατηγορία) βάσει της *Ευκλείδειας απόστασης*¹

¹ Ο *Ευκλείδειος ταξινομητής* χρησιμοποιεί τους μέσους όρους κάθε κλάσης (class means) και για κάθε δείγμα υπολογίζει τις ευκλείδειες αποστάσεις από όλα τα class means. Στη συνέχεια ταξινομεί τα δείγματα στην κλάση από της οποίας το μέσο όρο απέχει λιγότερο (minimum Euclidean distance from class means). Ουσιαστικά είναι

(υπόδειξη: χρησιμοποιείτε τις τιμές που υπολογίσατε στο Βήμα 9(α)). Ήταν επιτυχής η ταξινόμηση;

Βήμα 11

(α) Ταξινομήστε όλα τα ψηφία των test δεδομένων σε μία από τις 10 κατηγορίες με βάση την Ευκλείδεια απόσταση.

(β) Υπολογίστε το ποσοστό επιτυχίας για το Βήμα 11(α).

Βήμα 12

Υλοποιήστε τον ταξινομητή ευκλείδειας απόστασης σαν ένα scikit-learn estimator.

(Hint: το αρχείο lib.py που σας δίνεται)

Βήμα 13

α) Υπολογίστε το score του ευκλείδειου ταξινομητή με χρήση 5-fold cross-validation

β) Σχεδιάστε την περιοχή απόφασης του ευκλείδειου ταξινομητή.

γ) Σχεδιάστε την καμπύλη εκμάθησης του ευκλείδειου ταξινομητή (learning curve)

Τα παρακάτω βήματα δεν αποτελούν μέρος της προπαρασκευής.

Βήμα 14

Υπολογίστε τις a-priori πιθανότητες για κάθε κατηγορία (class priors).

Βήμα 15

α) Ταξινομήστε όλα τα ψηφία των test δεδομένων ως προς τις 10 κατηγορίες χρησιμοποιώντας τις τιμές της μέσης τιμής και διασποράς που υπολογίσατε στο Βήμα 9(α), υλοποιώντας έναν *Naive Bayesian ταξινομητή*². Μην χρησιμοποιήσετε έτοιμες υλοποιήσεις. **Η υλοποίηση σας πρέπει να είναι συμβατή με το scikit-learn όπως δείξαμε στο Βήμα 12.**

β) Υπολογίστε το σκορ για το Βήμα 15(α).

γ) Συγκρίνετε την υλοποίησή σας του Naive Bayes με την υλοποίηση του scikit-learn (GaussianNB).

Βήμα 16

Επαναλάβετε το Βήμα 15(α), (β) υποθέτοντας ότι η διασπορά για όλα τα χαρακτηριστικά, για όλες τις κατηγορίες ισούται με 1.

Βήμα 17

Συγκρίνετε την επίδοση των ταξινομητών Naive Bayes, Nearest Neighbors, SVM (με διαφορετικούς kernels).

Μπορείτε να χρησιμοποιήσετε τις υλοποιήσεις του scikit-learn.

μια υποπερίπτωση ενός Bayes Classifier με κανονικές (gaussian) παραμετρικές κατανομές, όπου ο πίνακας συνδιακύμανσης ανάμεσα σε όλες τις κατηγορίες είναι ο μοναδιαίος ενώ οι a-priori πιθανότητες κάθε κλάσης θεωρούνται ίδιες.

² ***Naive Bayes Classifier***: Ένας Bayes Classifier πάνω σε κάποιο σύνολο (π.χ. εικόνων) μοντελοποιεί με βάση τα δείγματα που έχει μια κατανομή (π.χ. κανονική) για κάθε κλάση. Η κατανομή αυτή είναι πολυδιάστατη και αφορά όλα τα εμπλεκόμενα “στοιχεία” κάθε δείγματος για κάθε κλάση (π.χ. pixels κάθε ψηφίου για την κλάση “2”). Με απλά λόγια ο Bayes Classifier θεωρεί ότι τα “στοιχεία” (π.χ. pixels) των δειγμάτων για κάθε κλάση σχετίζονται μεταξύ τους και περιγράφει αυτή τους τη σχέση με μια πολυδιάστατη πιθανοτική κατανομή. Ο Naive Bayes Classifier τώρα, υποθέτει πως τα επιμέρους “στοιχεία” κάθε δείγματος είναι ανεξάρτητα. Αυτό πέραν της μαθηματικής απλούστευσης στην περιγραφή, καταλήγει και σε μικρότερο αριθμό παραμέτρων για να περιγράψει την κάθε κλάση (με πιθανό κόστος στην απόδοση του ταξινομητή). Δηλαδή ένας Naive Bayes Classifier υποθέτει ότι τα pixels μιας εικόνας είναι ασυσχέτιστα και συνεπώς το κάθε ένα μπορεί να περιγραφεί από μια και μόνο πιθανοτική κατανομή (π.χ. μια κανονική κατανομή για κάθε pixel).

Βήμα 18³

Η βασική ιδέα του βήματος αυτού είναι ο συνδυασμός κάποιων ταξινομητών με αρκετά καλή επίδοση με στόχο να επιτευχθεί επίδοση υψηλότερη των επιμέρους επιδόσεων. Αυτή η τεχνική είναι γνωστή ως *ensembling*. Είναι σημαντικό οι ταξινομητές που θα συνδυαστούν να χαρακτηρίζονται από διαφορετικό τύπο λαθών, π.χ., ο ένας ταξινομητής να τείνει να ταξινομεί λάθος το ψηφίο 3, ενώ ένας άλλος να τείνει να ταξινομεί λάθος το ψηφίο 7.

α) Επιλέξτε κάποιους από τους ταξινομητές που χρησιμοποιήσατε στα προηγούμενα βήματα. Χρησιμοποιήστε το `VotingClassifier` του `scikit-learn` για να τους συνδυάσετε σε *hard* ή *soft voting*. Αυτός ο μετα-ταξινομητής συνδυάζει τους επιμέρους ταξινομητές βάζοντάς τους να ψηφίσουν για το αποτέλεσμα. Πρέπει να επιλέξετε μονό αριθμό ταξινομητών. Γιατί;

β) Επιλέξτε έναν ταξινομητή από τα προηγούμενα βήματα και χρησιμοποιήστε τον `BaggingClassifier` για να δημιουργήσετε ένα *ensemble*. Η τεχνική *bagging*, αφορά στο χωρισμό του (training) dataset σε τυχαία υποσύνολα (με πιθανές επικαλύψεις) και την εφαρμογή ενός ταξινομητή σε κάθε ένα από αυτά. Η τελική απόφαση βγαίνει μέσω ψηφοφορίας ή μέσου όρου των προβλέψεων των επιμέρους ταξινομητών. Ο συνδυασμός αυτής της τεχνικής με *Decision Trees* μας δίνει τον ταξινομητή *Random Forest*.

γ) Σχολιάστε τα αποτελέσματα.

Βήμα 19 (Extra credit 10% για τους προπτυχιακούς. Υποχρεωτικό για τους μεταπτυχιακούς και ΥΔ)

Σε αυτό το βήμα θα κάνουμε μια εισαγωγή στα νευρωνικά δίκτυα και στη βιβλιοθήκη `PyTorch`.

α) Υλοποιήστε έναν `dataloader` για να αναλάβει την ανάγνωση των δεδομένων και τον χωρισμό σε *batches*.

Ακολουθήστε αυτές τις οδηγίες (https://pytorch.org/tutorials/beginner/data_loading_tutorial.html)

β) Υλοποιήστε ένα *fully connected* νευρωνικό δίκτυο σε `PyTorch` σαν μια υποκλάση της `nn.Module`. και εκπαιδεύστε το στα δεδομένα. Πειραματιστείτε με τον αριθμό των νευρώνων, τον αριθμό των *layers* και τον τύπο των μη γραμμικών *actications*.

(οδηγίες: https://pytorch.org/tutorials/beginner/examples_nn/two_layer_net_module.html)

γ) Γράψτε τον κώδικα για την εκπαίδευση και το *evaluation* του νευρωνικού ,συμβατή με το `scikit-learn` βλ. Βήμα 12). Χωρίστε το dataset σε *train* και *validation* για το *training*.

δ) Αξιολογήστε την επίδοση του νευρωνικού στα δεδομένα *test*.

ΠΑΡΑΔΟΤΕΑ

Το τελικό παραδοτέο αφορά όλα τα βήματα (προπαρασκευή + εργαστήριο).

(1) Σύνομη αναφορά (σε **.pdf**) που θα περιγράφει τη διαδικασία που ακολουθήθηκε σε κάθε βήμα, καθώς και τα σχετικά αποτελέσματα.

(2) Κώδικας `python` (**.py**) συνοδευόμενος από σχόλια.

(3) Το αρχείο **lib.py** με συμπληρωμένες τις δικές σας υλοποιήσεις

Συγκεντρώστε τα (1), (2) και (3) σε ένα *.zip* αρχείο το οποίο πρέπει να αποσταλεί μέσω του `mycourses.ntua.gr` εντός της καθορισμένης προθεσμίας. **Εαν κάποιο .zip δεν περιέχει και τα 3 αρχεία δεν θα βαθμολογηθεί.**

Εάν κάνετε χρήση notebook, μπορείτε στη συνέχεια να κάνετε export του notebook (.ipynb) σε .py μορφή και να μας αποστείλετε και τα δύο αυτά αρχεία.

³ *Ensemble/Bagging/Boosting*: Πρόκειται για τεχνικές που συνδυάζουν ταξινομητές (και όχι μόνο) με σκοπό να πετύχουν καλύτερα αποτελέσματα. Παραπέμπουμε σε σχετική βιβλιογραφία όπως αυτή αναγράφεται στο `mycourses` και συγκεκριμένα [DHS] Ενότητα 9.5, [Bishop] Κεφ 14.