ΠΜΣ-07 Τεχνητή Νοημοσύνη και Εφαρμογές στο Διαδίκτυο των Πραγμάτων
Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

1η Εργασία

Έκδοση 2023-1.0

Διδάσκων: Χρήστος Δίου
Επικουρική Διδασκαλία: Βασίλης Γκολέμης

1 Εισαγωγή

Στην εργασία αυτή θα ασχοληθούμε με την ανάπτυξη μοντέλων κατηγοριοποίησης (classification) για δεδομένα που μας δίνονται σε μορφή πίνακα (tabular data). Στόχος είναι η εξοικείωσή σας με όλα τα βήματα που απαιτεί ένα πρόβλημα Μηχανικής Μάθησης, δηλαδή (α) φόρτωση και προετοιμασία των δεδομένων, (β) ανάπτυξη μοντέλων πρόβλεψης, (γ) αξιολόγηση των προτεινόμενων λύσεων, καθώς και η συγγραφή αναφοράς για την παρουσίαση των αποτελεσμάτων. Θα ασχοληθούμε με το πρόβλημα της αυτόματης αναγνώρισης ανθρώπινης δραστηριότητας (human activity recognition). Στο πρόβλημα αυτό λαμβάνουμε ως είσοδο σήματα από αισθητήρες (πχ. γυροσκόπιο, επιταχυνσιόμετρο) ενός wearable (πχ. smartphone, smartwatch) και ως έξοδο προβλέπουμε τη δραστηριότητα του χρήστη (πχ. αν τρέχει, αν περπατάει, αν ξαπλώνει κλπ.). Η αυτόματη αναγνώριση της ανθρώπινης δραστηριότητας είναι ένα ιδιαίτερα διαδεδομένο πρόβλημα (τα περισσότερα smartwatches έχουν ενσωματωμένη κάποια εφαρμογή που το επιλύει) καθώς δίνει τη δυνατότητα στους χρήστες να παρακολουθούν χρήσιμα στατιστικά του τρόπου ζωής τους, όπως για παράδειγμα πόσα βήματα κάνουν ανά ημέρα.

1.1 Σύντομη Περιγραφή του Προβλήματος

Δίνονται οι καταγραφές 30 ατόμων σε ένα σύνολο από CSV (comma-separated values) αρχεία. Τα άτομα είναι ηλικίας από 19 έως 48 ετών και εκτελούν δραστηριότητες ενώ φορούν ένα smartphone στην μέση τους. Οι δραστηριότητες που εκτελούν είναι οι εξής: walking, walking upstairs, walking downstairs, sitting, standing, laying. Αυτές οι 6 καταστάσεις αποτελούν και τα labels του προβλήματος. Καθόλη τη διάρκεια του πειράματος καταγράφονται από βίντεο, το οποίο μετά χρησιμοποιείται για το labeling. Λεπτομέρειες σχετικά με τα δεδομένα μπορείτε να βρείτε στο αποθετήριο UCI. Στα δεδομένα έχουν ήδη εκτελεστεί κάποια βήματα προεπεξεργασίας.

Η εργασία σας θα πρέπει να υλοποιηθεί σε γλώσσα Python, χρησιμοποιώντας τις βιβλιοθήκες Numpy, Pandas, Scikit-learn και Tensorflow. Επομένως, θα χρειαστεί να δημιουργήσετε ένα `Python` περιβάλλον με τις παραπάνω βιβλιοθήκες. Για αυτόν τον σκοπό σας παρέχουμε ένα αρχείο requirements.txt με όλα τα απαραίτητα πακέτα. Σημ: Αν προτιμάτε, μπορείτε να χρησιμοποιήσετε και Pytorch αντί για Tensorflow.

Θα πρέπει να παραδώσετε ένα αρχείο `<name>.zip`, όπου `<name>` το επίθετό σας, το οποίο θα περιέχει: (α) τον κώδικα της υλοποίησής σας, είτε σε ένα αρχείο .py (python script) είτε σε ένα αρχείο .ipynb (Jupyter Notebook) και (β) ένα αρχείο PDF με την αναφορά σας.

2 Φόρτωση Δεδομένων

Για την φόρτωση των δεδομένων, θα χρησιμοποιήσετε το βοηθητικό script utils.py που σας έχει δοθεί. Πρώτα, θα αποσυμπιέσετε τα δεδομένα και έπειτα θα τα φορτώσετε στην μνήμη μέσω της εντολής `utils.load_data()`. Αν λοιπόν υποθέσουμε ότι αποσυμπιέσατε τα δεδομένα στον φάκελο `./data/`, εκτελέστε το παρακάτω κομμάτι κώδικα:

```
from utils import load_data
X_train, y_train, X_test, y_test = load_data("./data/")
```

Επιβεβαιώστε ότι τα `np.array` έχουν φορτωθεί επιτυχημένα και έχουν τις αναμενόμενες διαστάσεις:

```
print(X_train.shape) # shape: (7352, 128, 9)
print(y_train.shape) # shape: (7352,)
print(X_test.shape)  # shape: (2947, 128, 9)
print(y_test.shape)  # shape: (2947,)
```

Για τα πειράματά μας θα χρειαστούμε τα δεδομένα σε δύο επιπλέον παραλλαγές. Τα δεδομένα εισόδου (`X_train, X_test`) έχουν τη μορφή χρονοσειρών, δηλαδή 128 δείγματα επί 9 χαρακτηριστικά (features) ανά δείγμα. Αυτή η μορφή είναι χρήσιμη εάν αξιοποιήσουμε τη χρονική συσχέτιση που έχει από τη φύση του το σήμα, όπως θα κάνουμε στο Κεφάλαιο 6. Στα προηγούμενα Κεφάλαια (4, 5) θα αγνοήσουμε την χρονική συσχέτιση και θα αντιμετωπίσουμε το κάθε δείγμα και το κάθε χαρακτηριστικό ως ένα ξεχωριστό feature. Επομένως, θα δημιουργήσουμε δύο `np.arrays` με $128 * 9 = 1152$ features. Αυτή η διαδικασία ονομάζεται flattening και επιτυγχάνεται με το παρακάτω κομμάτι κώδικα:

```
# shape: (7352, 1152)
X_train_flat = utils.flatten_X(X_train)
# shape: (2947, 1152)
X_test_flat = utils.flatten_X(X_test)
```

Επίσης, τα `y_train, y_test` κωδικοποιούν τα labels με έναν integer από το set $\{0, 1, 2, 3, 4, 5\}$. Στα περισσότερα προβλήματα classification προτιμούμε την κωδικοποίηση "one hot encoding". Για αυτόν τον σκοπό, θα δημιουργήσουμε δύο νέα `np.arrays` που θα περιέχουν τα labels με αυτήν την κωδικοποίηση:

```
# shape: (7352, 6)
y_train_one_hot = utils.int_to_one_hot(y_train, n_classes=6)
# shape: (2947, 6)
y_test_one_hot = utils.int_to_one_hot(y_test, n_classes=6)
```

Είστε πλέον έτοιμοι για την επίλυση του προβλήματος!

3 Επεξεργασία και ανάλυση δεδομένων EDA

4 Γραμμικό Μοντέλο πρόβλεψης

4.1 Με χρήση του scikit-learn

Αναπτύξτε ένα γραμμικό μοντέλο χρησιμοποιώντας το scikit-learn, αξιοποιώντας όσα μάθαμε στο μάθημα. Ίσως χρειαστεί να πειραματιστείτε λίγο με τις παραμέτρους του αλγόριθμου βελτιστοποίησης. Για τη δημιουργία του γραμμικού μοντέλου πρόβλεψης μπορείτε να χρησιμοποιήσετε την κλάση:

```
sklearn.linear_model.LogisticRegression()
```

Δώστε ιδιαίτερη προσοχή στη μορφή των δεδομένων που πρέπει να χρησιμοποιηθούν, δηλαδή `X_train` ή `X_train_flat` και `y_train` ή `y_train_one_hot` .

4.2 Με χρήση του tensorflow

Αναπτύξτε τώρα ένα γραμμικό μοντέλο χρησιμοποιώντας το tensorflow. Ως γνωστόν, ένα γραμμικό μοντέλο είναι ουσιαστικά ένα νευρωνικό δίκτυο χωρίς κανένα κρυφό επίπεδο (hidden layer). Για την δημιουργία και την εκπαίδευση του μοντέλου μπορείτε να χρησιμοποιήσετε τα παρακάτω με τα **κατάλληλα ορίσματα**:

```
model = models.Sequential()
layers.Dense()
model.compile()
model.fit()
model.predict()
```

4.3 Αξιολόγηση Μοντέλου

Για την αξιολόγηση του μοντέλου θα χρησιμοποιήσουμε τα παρακάτω:

- τον πίνακα σύγχυσης (confusion matrix)
- την ακρίβεια (Accuracy)
- το f1-score

Για τα παραπάνω μπορείτε να χρησιμοποιήσετε τις ρουτίνες του πακέτου `sklearn`:

```
sklearn.metrics.confusion_matrix()
sklearn.metrics.f1_score()
sklearn.metrics.accuracy_score()
```

Με βάση τα παραπάνω αποτελέσματα, απαντήστε στις παρακάτω τρεις ερωτήσεις:

- ποια η πιθανότητα το μοντέλο να είναι εύστοχο σε μια τυχαία μελλοντική πρόβλεψη;
- αν το μοντέλο προέβλεψε για κάποιον χρήστη ότι περπατάει, ποια η πιθανότητα να είναι εύστοχη η πρόβλεψη;
- αν περπατάω, ποια η πιθανότητα να δω την ένδειξη "περπάτημα" αν κοιτάξω το ρολόι; (εναλλακτικά: αν περπατάω, ποια η πιθανότητα το μοντέλο να έχει προβλέψει σωστά);

Συμβουλή: Επειδή η αξιολόγηση όλων των μοντέλων θα γίνει με τον ίδιο τρόπο, θα ήταν βολικό να υλοποιήσετε μία συνάρτηση που επιστρέφει όλα τα παραπάνω αποτελέσματα και να την επαναχρησιμοποιείτε στα ερωτήματα των παρακάτω κεφαλαίων.

5 Πλήρως συνδεδεμένο νευρωνικό δίκτυο

Στο βήμα αυτό θα υλοποιήσετε ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο (fully connected) νευρωνικό δίκτυο. Πειραματιστείτε με διάφορες αρχιτεκτονικές νευρωνικού δικτύου. Μεταξύ άλλων πειραματιστείτε με το πλήθος των κρυφών επιπέδων (βάθος), τον αριθμό των νευρώνων σε κάθε επίπεδο (πλάτος), τη χρήση διαφορετικού ρυθμού εκμάθησης και διαφορετικού optimizer (SGD, Adam). Μπορείτε προαιρετικά να εξετάσετε και τη χρήση dropout ή ομαλοποίησης για την αποφυγή της υπερεκπαίδευσης, καθώς και τη χρήση "learning rate schedules" όπου ο ρυθμός εκμάθησης μεταβάλλεται κατά τη διάρκεια της εκπαίδευσης. Αξιολογήστε το μοντέλο/τα μοντέλα σας όπως ακριβώς στο Κεφάλαιο 4.3.

Όπως και πριν, οι παρακάτω εντολές μπορεί να σας φανούν χρήσιμες:

```
model = models.Sequential()
layers.Dense()
model.compile()
model.fit()
model.predict()
```

6 Συνελικτικό νευρωνικό δίκτυο

Στο βήμα αυτό θα υλοποιήσετε ένα συνελικτικό νευρωνικό δίκτυο (convolutional neural network). Σε αντίθεση με τις προηγούμενες περιπτώσεις, εδώ αντιμετωπίζουμε την πληροφορία εισόδου ως σήμα, δηλαδή θεωρούμε ότι έχουμε μια χρονοσειρά με 128 διαδοχικά σημεία όπου το καθένα περιγράφεται απο 9 χαρακτηριστικά (features). Επομένως χρησιμοποιείστε τους πίνακες `X_train_flat` και `X_test_flat` ως είσοδο στο μοντέλο.

Πειραματιστείτε με διάφορες αρχιτεκτονικές συνελικτικών νευρωνικών δικτύου. Μεταξύ άλλων πειραματιστείτε με το πλήθος των κρυφών επιπέδων (βάθος), το μέγεθος του συνελικτικού πυρήνα (kernel size) σε κάθε επίπεδο, τη χρήση διαφορετικού ρυθμού εκμάθησης και διαφορετικού optimizer (SGD, Adam). Μπορείτε προαιρετικά να εξετάσετε και τη χρήση dropout ή ομαλοποίησης για την αποφυγή της υπερεκπαίδευσης,

καθώς και τη χρήση “learning rate schedules” όπου ο ρυθμός εκμάθησης μεταβάλλεται κατά τη διάρκεια της εκπαίδευσης.

Αξιολογήστε το μοντέλο/τα μοντέλα σας όπως ακριβώς στο Κεφάλαιο 4.3.

Οι παρακάτω εντολές μπορεί να σας φανούν χρήσιμες:

```
model = models.Sequential()
layers.Conv1D()
layers.MaxPooling1D()
layers.Flatten()
layers.Dense()
model.compile()
model.fit()
model.predict()
```

7 Random Forest

Συμβαίνει συχνά σε δεδομένα πίνακα (tabular data) οι μέθοδοι συνόλου (ensemble methods) να αποδίδουν καλύτερα, ιδιαίτερα αυτές που βασίζονται σε δέντρα αποφάσεων, όπως οι μέθοδοι Random Forest και EXTRA trees και σε περιπτώσεις με σχετικά λίγα δείγματα εκπαίδευσης. Αρκεί να δημιουργήσετε ένα αντικείμενο `sklearn.ensemble.Randomforestclassifier()` παρόμοια με τα παραδείγματα που είδαμε στο μάθημα για τα γραμμικά μοντέλα στο `scikit-learn`. Αξιολογήστε το μοντέλο σας όπως ακριβώς στο Κεφάλαιο 4.3.