# Δεύτερη Σειρά Ασκήσεων

Η προθεσμία για την δεύτερη σειρά ασκήσεων είναι την Τετάρτη 18 Δεκεμβρίου, 11:55 μ.μ. Παραδώστε Notebooks με τον κώδικα και τις αναφορές σε .ipynb **και** .html μορφή. Για την Ερώτηση 1 μπορείτε να παραδώσετε και pdf με την απόδειξη, ή φωτογραφίες από χειρόγραφα. Για καθυστερημένες υποβολές ισχύει η πολιτική στην σελίδα του μαθήματος. Η παράδοση θα γίνει μέσω του ecourse. Λεπτομέρειες στη σελίδα Ασκήσεις του μαθήματος. Η άσκηση είναι **ατομική**.

## Ερώτηση 1

Α. Μία εκθετική κατανομή ορίζεται με συνάρτηση πυκνότητας πιθανότητας $f(x) = \lambda e^{-\lambda x}$, για $x \geq 0$, όπου $\lambda$ είναι η παράμετρος της κατανομής. Σας δίνεται ένα σύνολο από παρατηρήσεις $X = \{x_1, \dots, x_n\}$, $x_i \geq 0$, που έχουν παραχθεί από μία εκθετική κατανομή. Χρησιμοποιήστε την Maximum Likelihood Estimation τεχνική που περιγράψαμε στην τάξη για να βρείτε την παράμετρο της κατανομής που ταιριάζει (fits) τα δεδομένα των παρατηρήσεων.

Β. Υποθέστε ότι οι παρατηρήσεις $X = \{x_1, \dots, x_n\}$ έχουν παραχθεί από ένα μείγμα εκθετικών κατανομών, $L_1, L_2$, με παραμέτρους $\lambda_1, \lambda_2$, και πιθανότητες μίξης (mixture probabilities) $\pi_1, \pi_2$. Θα χρησιμοποιήσετε τον EM αλγόριθμο για να υπολογίσουμε τις παραμέτρους $\Theta = (\lambda_1, \lambda_2, \pi_1, \pi_2)$ του mixture μοντέλου, όπως κάναμε και για την περίπτωση της μίξης από Gaussian κατανομές. Στο Μ βήμα, υποθέτουμε ότι έχουμε τις πιθανότητες ανάθεσης $P(L_k|x_i)$, για $k = 1,2$ και $i = 1, \dots, n$, και θέλουμε να υπολογίσουμε τις παραμέτρους $\Theta$. Δώστε τις εξισώσεις για τα $\lambda_1, \lambda_2, \pi_1, \pi_2$ και τους υπολογισμούς με τους οποίους τις παρήγατε.

**Υπόδειξη**: Θα σας βοηθήσει να διαβάσετε τις σημειώσεις του Άρη Αναγνωστόπουλου που είναι στην σελίδα του μαθήματος για την περίπτωση της μίξης των Gaussians, τις οποίες παρουσιάσαμε στο μάθημα.

## Ερώτηση 2

Ο στόχος αυτής της άσκησης είναι να πειραματιστείτε με αλγορίθμους για συστήματα συστάσεων και να εξασκηθείτε στην διαχείριση πινάκων μέσα από τις βιβλιοθήκες numpy και scipy που έχουμε μάθει, καθώς και με μια απλή χρήση του clustering.

Θα χρησιμοποιήσετε δεδομένα από το MovieLens dataset με ratings χρηστών για ταινίες. Περιέχει ratings από 943 χρήστες για 1682 ταινίες. Τα δεδομένα σας δίνονται προεπεξεργασμένα στη σελίδα του μαθήματος. Το αρχείο data-train.csv περιέχει τα ratings τα οποία θα χρησιμοποιήσετε για να δημιουργήσετε τα συστήματα συστάσεων. Το αρχείο data-test.csv περιέχει τα δεδομένα που θα χρησιμοποιήσετε για να αξιολογήσετε τα συστήματα συστάσεων που θα δημιουργήσετε. Και τα δύο αρχεία περιέχουν τριάδες της μορφής: user_id, movie_id, rating. Τα ids των χρηστών και ταινιών ξεκινάνε από το 0, το οποίο θα σας φανεί χρήσιμο για να φορτώσετε τα δεδομένα σε numpy πίνακες. Τα ratings είναι ακέραιες τιμές τις οποίες για να δουλέψουν κάποιες συναρτήσεις θα πρέπει να τις μετατρέψετε σε floats.

Θα υλοποιήσετε και θα τεστάρετε διαφορετικούς αλγόριθμους συστάσεων. Ο στόχος μας είναι για κάθε ζευγάρι χρήστη-ταινίας $(u, m)$ στα test δεδομένα να υπολογίσουμε ένα score που είναι όσο πιο κοντά γίνεται στην πραγματική βαθμολογία του χρήστη για την ταινία. Για την αξιολόγηση θα χρησιμοποιήσετε το RMSE (Root Mean Square Error). Αν $r_1, r_2, \dots, r_n$ είναι τα ratings που θέλουμε να προβλέψουμε, και $p_1, p_2, \dots, p_n$ είναι οι προβλέψεις ενός αλγορίθμου, το RMSE του αλγορίθμου ορίζεται ως

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - p_i)^2}$$

**Υπόδειξη:** Το RMSE υπάρχει υλοποιημένο στην βιβλιοθήκη metrics της scikit-learn.

**Βήμα 1**: Οι δύο πρώτοι αλγόριθμοι που θα δοκιμάσουμε είναι ο **User Average (UA)** και ο **Item Average (IA)**. Ο πρώτος για κάθε ζευγάρι $(u, m)$ προβλέπει τη μέση βαθμολογία του χρήστη $u$, ενώ ο δεύτερος τη μέση βαθμολογία του αντικειμένου (ταινία) $m$. Η μέση βαθμολογία του χρήστη υπολογίζεται μόνο από τα αντικείμενα τα οποία έχει βαθμολογήσει. Αντίστοιχα η μέση βαθμολογία ενός κινητού υπολογίζεται μόνο από τους χρήστες που το έχουν βαθμολογήσει. Για το βήμα αυτό σας προτείνεται να κρατήσετε τα δεδομένα σε dataframes. Για να πάρετε όλους τους βαθμούς της άσκησης η υλοποίηση σας θα πρέπει να γίνει χρησιμοποιώντας εντολές της βιβλιοθήκης pandas, χωρίς να διατρέξετε τις γραμμές του dataframe με for loop. Υπολογίστε και αναφέρετε το RMSE για τους δύο αλγορίθμους.

**Βήμα 2:** Ο επόμενος αλγόριθμος που θα δοκιμάσετε είναι ο **Singular Value Decomposition (SVD)**. Για την υλοποίηση του αλγορίθμου θα φορτώσετε το train dataset σε ένα **αραιό** user-movie πίνακα $R$. Η πρόβλεψη του rating για το ζεύγος $(u, m)$ θα γίνει χρησιμοποιώντας η τιμή $R_k[u, m]$ του rank-$k$ approximation πίνακα $R_k$ του πίνακα $R$ για διάφορες τιμές του $k$.

Υπολογίστε το Singular Value Decomposition για $K = 20$. Στη συνέχεια για όλα τα $k \in [1 \dots 20]$, για κάθε ζευγάρι $(u, m)$ στα test δεδομένα, υπολογίσετε την τιμή $R_k[u, m]$. Δεν θα δημιουργήσετε τον πίνακα $R_k$ ο οποίος είναι πυκνός. Η τιμή $R_k[u, c]$ μπορεί να υπολογιστεί ως

$$R_k[u, c] = U_k[u, :]\Sigma_k V_k^T[:, c]$$

Όπου $U_k[u, :]$ είναι η γραμμή $u$ του πίνακα $U_k$ με τα $k$ πρώτα left singular vectors, $\Sigma_k$ είναι ο διαγώνιος πίνακας με τα $k$ μεγαλύτερα singular values και $V_k^T[:, c]$ είναι η στήλη $c$ του ανάστροφου πίνακα $V_k$ με τα $k$ πρώτα right singular vectors. Το «πρώτα» αναφέρεται στις μεγαλύτερες singular values και τα αντίστοιχα διανύσματα. Προσέξτε να πάρετε τα διανύσματα και τις τιμές με φθίνουσα σειρά. Ανάλογα με την βιβλιοθήκη που θα χρησιμοποιήσετε τα μεγαλύτερα singular values μπορεί να είναι τα τελευταία. Αν η βαθμολογία που υπολογίζει ο αλγόριθμος είναι μεγαλύτερη από 5 ή μικρότερη από μηδέν κάνετε “clip” την βαθμολογία σε 5 και 0 αντίστοιχα.

Υπολογίστε το RMSE για κάθε $k \in [1, K]$ και κάνετε μια γραφική παράσταση του RMSE ως προς το $k$. Αναφέρετε την τιμή του $k$ που ο αλγόριθμος πετυχαίνει το ελάχιστο RMSE και ποιο είναι αυτό.

**Υπόδειξη**: Σε κανένα σημείο δεν θα πρέπει να δημιουργήσετε ένα πυκνό πίνακα $N \times M$. Ο πίνακας $R$ θα πρέπει να είναι αραιός, και όπως λέει παραπάνω η εκφώνηση, και δεν πρέπει να δημιουργήσετε τον πίνακα $R_k$. Θα χάσετε μονάδες αν δημιουργήσετε πυκνούς πίνακες.

**Βήμα 4**: Στη συνέχεια θα υλοποιήσετε τον **User-Based Collaborative Filtering (UCF)** αλγόριθμο. Για το σκοπό αυτό θα χρησιμοποιήσετε πάλι τον **αραιό** user-movie πίνακα $R$. Ο UCF αλγόριθμος έχει μια παράμετρο $k$, που είναι ο αριθμός των όμοιων χρηστών που κοιτάει. Για να υπολογίσετε την τιμή για ένα ζευγάρι $(u, m)$ υπολογίστε το σύνολο $N_k(u, m)$ με τους $k$ πιο όμοιους χρήστες με τον χρήστη $u$ οι οποίοι έχουν βαθμολογήσει την ταινία $m$. Στη συνέχεια χρησιμοποιήστε την εξής εξίσωση για την πρόβλεψη σας:

$$p(u, m) = \frac{\sum_{u' \in N_k(u,m)} s(u, u') R[u', m]}{\sum_{u' \in N_k(u,m)} s(u, u')}$$

Στην εξίσωση $s(u, u')$ είναι το cosine similarity μεταξύ των χρηστών $u$ και $u'$.

Για να πάρετε όλες τις μονάδες θα πρέπει να υλοποιήσετε τον αλγόριθμο χρησιμοποιώντας μόνο μεθόδους διαχείρισης πινάκων και διανυσμάτων της numpy ή scipy. Σε κανένα σημείο του προγράμματος δεν πρέπει να υπολογίσετε κάποιο πυκνό πίνακα $N \times M$, και σε κανένα σημείο δεν πρέπει να χρησιμοποιήσετε for loop για να διατρέξετε τους χρήστες ή τις ταινίες στα training data. Για ευκολία σας η υλοποίηση σπάει στα εξής βήματα.

**Βήμα 4.1**. Στο βήμα αυτό θα εξασκηθείτε στην χρήση των μεθόδων της numpy και scipy, και θα υλοποιήσετε κάποιες (κατά κύριο λόγο απλές) συναρτήσεις, τις οποίες μπορείτε να χρησιμοποιήσετε αργότερα (είτε να τις καλέσετε, είτε τον κώδικα τους).

Συνάρτηση **user_mean(R,u)**: Παίρνει σαν όρισμα τον πίνακα $R$, και ένα χρήστη $u$ και επιστρέφει την μέση τιμή των ratings του $u$.

Συνάρτηση **item_users(R,m)**: παίρνει σαν όρισμα τον πίνακα $R$ και την ταινία $m$ και επιστρέφει τους χρήστες που έχουν βαθμολογήσει την ταινία $m$.

Συνάρτηση **similar_users(R,u,m_users,k)**: Παίρνει σαν όρισμα τον πίνακα $R$, τον χρήστη $u$, ένα np.array $m\_users$ με ids χρηστών, και μια τιμή $k$. Υπολογίζει την ομοιότητα (cosine similarity) μεταξύ του $u$ και των χρηστών στο $m\_users$, και επιστρέφει τους $k$ πιο όμοιους χρήστες ταξινομημένους ως προς την ομοιότητα τους με τον $u$, και το διάνυσμα με τις ομοιότητες. Η συνάρτηση δεν επιτρέπεται να υπολογίσει τον user-user πίνακα με όλες τις ομοιότητες.

Συνάρτηση **compute_score(k_ratings,k_similarities)**: Παίρνει σαν όρισμα ένα np.array διάνυσμα $k\_ratings$ με τις βαθμολογίες από $k$ χρήστες για μια ταινία $m$, και ένα np.array διάνυσμα $k\_similarities$ με την ομοιότητα τους με τον χρήστη $u$ και υπολογίζει και επιστρέφει το score $p(u, m)$ όπως σας δίνεται παραπάνω.

**Βήμα 4.2**. Υλοποιήστε τον αλγόριθμο UCF. Δημιουργήστε μια συνάρτηση η οποία παίρνει σαν είσοδο ένα ζευγάρι $(u, m)$ και μια τιμή $k$ και κάνει τα εξής βήματα:

1. Βρίσκει τους χρήστες που έχουν βαθμολογήσει την ταινία $m$. Αν δεν υπάρχουν τέτοιοι χρήστες επιστρέφει την μέση τιμή των ratings του χρήστη $u$.

2. Υπολογίζει την ομοιότητα αυτών των χρηστών με τον χρήστη $u$ και κρατάει τους $k$ πιο όμοιους χρήστες. Αν υπάρχουν λιγότεροι από $k$ χρήστες που έχουν βαθμολογήσει την ταινία $m$, χρησιμοποιήστε τους όλους.
3. Χρησιμοποιείστε τα διανύσματα με τις ομοιότητες και τις βαθμολογίες για τους $k$ πιο όμοιους χρήστες, και υπολογίστε την βαθμολογία με την παραπάνω εξίσωση. Αν η ομοιότητα του χρήστη $u$ με όλους τους χρήστες είναι μηδέν επιστρέψετε μηδέν.

Τρέξτε τον αλγόριθμο για $k = [1,2,3,5,10,20,30,40,50,60,70,80,90,100]$ και φτιάξετε μια γραφική παράσταση με το RMSE ως συνάρτηση του $k$. Αναφέρετε την τιμή του $k$ που ο αλγόριθμος πετυχαίνει το ελάχιστο RMSE και ποιο είναι αυτό.

Αν τρέξετε για όλες τις τιμές του $k$, ένα τρέξιμο του αλγορίθμου θα πάρει ~5 λεπτά. Λάβετε υπόψη σας το υπολογιστικό κόστος του αλγορίθμου στην υλοποίηση σας. Αν θέλετε να κάνετε πολλά εξερευνητικά πειράματα χρησιμοποιήστε κάποιο sample των test δεδομένων. Τα αποτελέσματα που θα αναφέρετε θα πρέπει να είναι στο σύνολο των δεδομένων.

**Βήμα 4.3:** Τροποποιήστε τον κώδικα σας και δημιουργήστε μια συνάρτηση η οποία παίρνει σαν όρισμα ένα ζευγάρι $(u, m)$, για μια λίστα από τις τιμές για το $k$ υπολογίζει τις βαθμολογίες για το $(u, m)$ για όλες τις τιμές του $k$. Η συνάρτηση θα πρέπει να εκμεταλλεύεται το γεγονός ότι πολύς από τον υπολογισμό είναι κοινός για διαφορετικά $k$. Συγκρίνεται το χρόνο που παίρνει αυτή η υλοποίηση σε σχέση με αυτή στο βήμα 4.2

**Βήμα 5:** Αφού ολοκληρώσετε τα Βήματα 1-4, κάνετε μια συγκριτική αξιολόγηση των αλγορίθμων. Φτιάξετε ένα πίνακα που να έχει όλους τους αλγορίθμους, και το καλύτερο error που επιτυγχάνει ο κάθε αλγόριθμος, και σχολιάστε τα αποτελέσματα.

Παραδώσετε ένα notebook με τον κώδικα σας και την αναφορά με τις παρατηρήσεις σας για τα αποτελέσματα. Στο notebook το κάθε βήμα θα πρέπει να έχει δική του επικεφαλίδα.

**Bonus 1:** Υλοποιήστε την παραλλαγή του UCF αλγορίθμου που αφαιρούμε την μέση τιμή από τα ratings του κάθε χρήστη και προβλέπουμε την απόκλιση από μέση τιμή.

**Bonus 2:** Υλοποιήστε τον αλγόριθμο **Item-Based Collaborative Filtering (ICF)**. Ο αλγόριθμος είναι παρόμοιος με αυτόν στο Βήμα 4, απλά δουλεύετε με στήλες αντί για γραμμές.

**Υποδείξεις**

Οι παρακάτω συναρτήσεις μπορεί να σας φανούν χρήσιμες:

- Οι πράξεις μεταξύ πινάκων και διανυσμάτων με τη βιβλιοθήκη numpy μερικές φορές επιστρέφουν διανύσματα που μπορεί να έχουν διαφορετική μορφή απ ότι θα θέλατε οπότε θα πρέπει να είσαστε προσεκτικοί. Οι μέθοδοι reshape και flatten μπορεί να σας βοηθήσουν.
- Η μέθοδος nonzero σας δίνει τις μη μηδενικές τιμές σε ένα numpy διάνυσμα ή πίνακα.
- Η εντολή argsort της numpy σας δίνει την σειρά των indices ενός διανύσματος όπως προκύπτει μετά την ταξινόμηση των τιμών του.
- Για το RMSE μπορείτε να χρησιμοποιήσετε τη μέθοδο της sklearn.