Ευρωπαϊκή Ένωση
Ευρωπαϊκό Ταμείο Περιφερειακής Ανάπτυξης
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΥΠΟΥΡΓΕΙΟ ΟΙΚΟΝΟΜΙΑΣ & ΑΝΑΠΤΥΞΗΣ
ΕΙΔΙΚΗ ΓΡΑΜΜΑΤΕΙΑ ΕΠΑ & ΤΣ
ΕΠΑνΕΚ 2014-2020
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΤΗΤΑ ΚΑΙΝΟΤΟΜΙΑ
ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ARPOLIS
Augmented Reality Polis Stories

AUGMENTED REALITY POLIS STORIES

ΠΑΡΑΔΟΤΕΟ-4
ΣΥΣΤΗΜΑ ΞΕΝΑΓΗΣΗΣ ΣΤΗΝ ΑΘΗΝΑ
ΠΑΡΑΡΤΗΜΑ – 4.6
ΣΥΣΤΗΜΑ ΣΥΣΤΑΣΕΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΟΥ TENSORFLOW

δι@δρασις
ανάπτυξη εφαρμογών
web • mobile • multimedia

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Εισαγωγή

2 TensorFlow

2.1 Ιστορικό

Το TensorFlow αποτελεί μια ελεύθερη, ανοικτού κώδικα, βασισμένη στην Python machine learning πλατφόρμα η οποία αρχικά αναπτύχθηκε από την Google, κυκλοφόρησε το Νοέμβριο τους 2015 και αποτελεί αυτή τη στιγμή τη πιο δημοφιλή deep learning πλατφόρμα. Η ανάπτυξη του TensorFlow βασίστηκε στην πολυετή εμπειρία της ομάδας της Google Brain, δηλαδή της ομάδας τεχνητής νοημοσύνης της Google, με το πρώτης γενιάς σύστημα DistBelief, το οποίο αποτελεί ένα κατανεμημένο σύστημα εκπαίδευσης νευρωνικών δικτύων που χρησιμοποιούσε η Google από το 2011 (Jeffrey Dean, 2012). Σε αντίθεση με το προαναφερθέν σύστημα, το Tensorflow αναπτύχθηκε για να απλοποιήσει την διαδικασία ανάπτυξης των μοντέλων και για να αποτελέσει ένα πιο ευέλικτο εργαλείο το οποίο παράλληλα θα ικανοποιεί τις απαιτήσεις του φόρτου εργασίας μηχανικής εκμάθησης της Google.

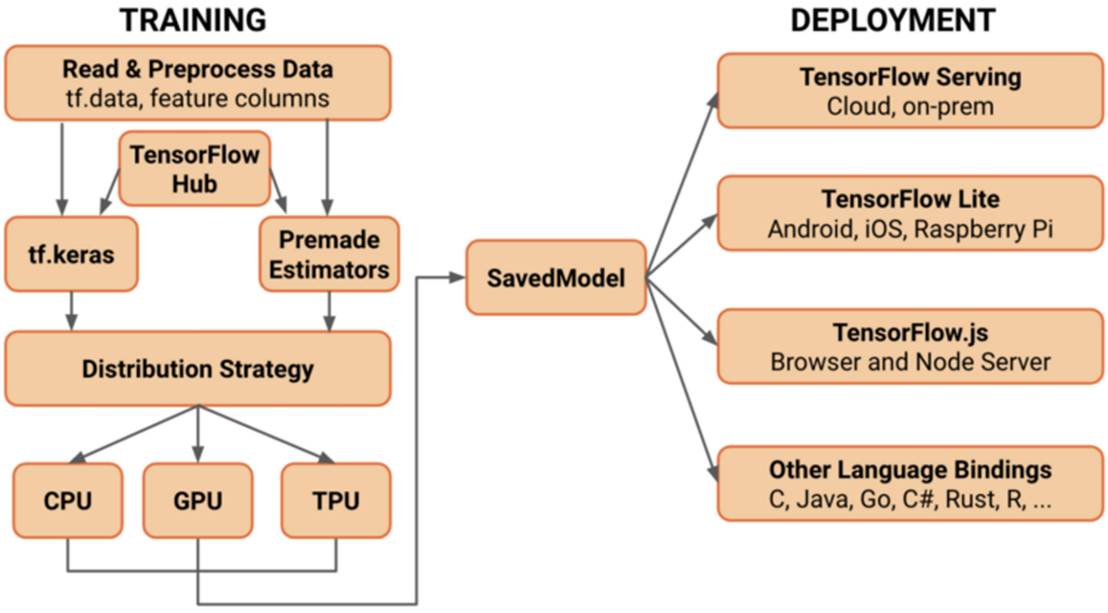
Το Tensorflow συνδυάζει την υπολογιστική άλγεβρα με τεχνικές βελτιστοποίησης για να εκτελεί εύκολα και γρήγορα μαθηματικές εκφράσεις οι οποίες διαφορετικά θα είχαν αυξημένες χρονικές απαιτήσεις λόγω πολυπλοκότητας. Με την εκτέλεση του Tensorflow να μπορεί να κατανέμει τις machine learning διεργασίες σε ποικίλες συσκευές, η απόδοση του αυξάνεται. Αν και το TensorFlow προσφέρει Python και C++ API όπου το πιο διαδεδομένο και πλήρες μπορεί να θεωρηθεί το Keras, έχει αναπτυχθεί σε C/C++ κάτι το οποίο το καθιστά ιδιαίτερα γρήγορο σε σύγκριση με άλλες deep learning βιβλιοθήκες.

Το Keras αποτελεί ένα deep-learning API του Tensorflow (Application Programming Interface ή Διεπαφή Προγραμματισμού Εφαρμογών) για Python το οποίο διευκολύνει τον προσδιορισμό και την εκπαίδευση κάθε είδους μοντέλου «βαθιάς» μάθησης. Το Keras αναπτύχθηκε αρχικά για έρευνα από τον Francois Chollet, ερευνητή την Google AI, 8 μήνες πριν το Tensorflow. Για την εκπαίδευση νευρωνικών δικτύων, όμως, το Keras χρειαζόταν ένα backend, δηλαδή μια υπολογιστική μηχανή για την δημιουργία του γραφήματος/τοπολογίας του δικτύου, για την εκτέλεση των βελτιστοποιήσεων κλπ. Αρχικά το backend του Keras ήταν το Theano, μια βιβλιοθήκη

που επίσης χειρίζεται τανυστές ενώ αργότερα υποστήριξε άλλα backend, όπως το CNTK και το mxnet. Με την κυκλοφορία του Tensorflow, το Keras ξεκίνησε να το υποστηρίζει ως backend το TensorFlow το οποίο σταδιακά έγινε το πιο δημοφιλές backend, με αποτέλεσμα το TensorFlow να είναι το προεπιλεγμένο backend από την έκδοση v1.1.0 του Keras. Από το σημείο αυτό, η χρήση του TensorFlow και του Keras αυξήθηκαν παράλληλα. Την ίδια στιγμή, η χρήστη το API υψηλού επιπέδου Keras γινόταν όλο και πιο δημοφιλές στους χρήστες του TensorFlow. Έτσι με την εισαγωγή του tf.keras στο TensorFlow v1.10.0, έγινε το πρώτο βήμα για την ενσωμάτωση του Keras απευθείας στο πακέτο του TensorFlow. Τον Ιούνιο του 2019 η Google ανακοίνωσε το TensorFlow 2.0 δηλώνοντας ότι το Keras αποτελεί πλέον το επίσημο high level API του TensorFlow για το γρήγορο και εύκολο σχεδιασμό καθώς και για την εκπαίδευση μοντέλων μηχανικής μάθησης.

2.2 Τρόπος Λειτουργίας

Η αρχιτεκτονική του Tensorflow 2.0 παρουσιάζεται στην Εικόνα 1 (Team, 2019).δ



Εικόνα 1: Νέα αρχιτεκτονική του Tensorflow

Στην ενότητα αυτή θα παρουσιαστεί ο τρόπος λειτουργίας του Tensorflow.

2.2.1 Αρχιτεκτονική του Tensorflow

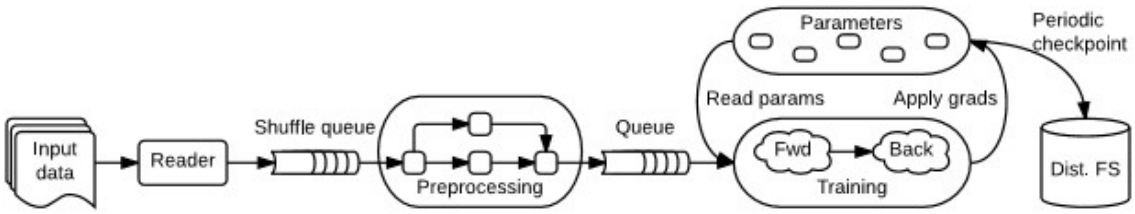
Η αρχιτεκτονική Tensorflow λειτουργεί σε τρία μέρη:

- Προ-επεξεργασία των δεδομένων
- Δημιουργία του μοντέλου
- Εκπαίδευση και αξιολόγηση του μοντέλου

Το Tensorflow πήρε την ονομασία τους από τους τανυστές (tensors) οι οποίοι είναι στην ουσία πολυδιάστατους πίνακες. Ένα μοντέλο Tensorflow, παίρνει ως είσοδο τους εν λόγω «tensors», οι οποίοι εισέρχονται στο ένα άκρο και στη συνέχεια «ρέουν» στο σύστημα που αποτελείται από πολλαπλές μαθηματικές λειτουργίες και εν τέλει «εξέρχονται» από το άλλο άκρο ως έξοδος. Από την προαναφερθείσα λειτουργία, λοιπόν, προκύπτει η ονομασία του TensorFlow, καθώς ένας τανυστής (tensor)

εισέρχεται σε αυτό, ρέει (flows) μέσω μιας σειράς λειτουργιών και στη συνέχεια εξέρχεται από την άλλη πλευρά.

Πιο αναλυτικά, το Tensorflow χρησιμοποιεί μια δομή που είναι γνωστή ως γράφος ροής δεδομένων (Data flow graph) όπου οι κύριες μονάδες είναι οι κόμβοι και οι ακμές. Οι κόμβοι (Nodes) σε έναν γράφο αναπαριστούν μαθηματικές λειτουργίες (πολλαπλασιασμός, άθροισμα, διαίρεση κλπ.) ενώ οι ακμές (Edges) αναπαριστούν τα δεδομένα που είναι συνήθως πολυδιάστατοι πίνακες ή τανυστές (tensors) οι οποίοι επικοινωνούν μεταξύ τους. Το TensorFlow χρησιμοποιεί ένα γράφο ροής δεδομένων για να αναπαραστήσει όλους τους υπολογισμούς σε έναν αλγόριθμο μηχανικής εκμάθησης, συμπεριλαμβανομένων των μεμονωμένων μαθηματικών λειτουργιών, των παραμέτρων και των κανόνων ενημέρωσής τους καθώς και της προεπεξεργασία που υφίστανται μια είσοδος (input) όπως φαίνεται και στην Εικόνα 2.



Εικόνα 2: Tensorflow dataflow Graph

Ο γράφος έχει πολλά πλεονεκτήματα:

- Έχει δημιουργηθεί ώστε να μπορεί να εκτελείται σε πολλές CPU ή GPU καθώς και σε λειτουργικό σύστημα κινητής συσκευής.
- Η φορητότητα του γράφου επιτρέπει τη διατήρηση των υπολογισμών για άμεση ή μεταγενέστερη χρήση, με άλλα λόγια μπορεί να αποθηκευτεί για εκτέλεση στο μέλλον.
- Εξοικονόμηση υπολογιστικής ισχύος
- Διαίρεση των υπολογισμών σε μικρά, διαφορετικά μέρη

Τα στοιχεία γραφήματος ροής δεδομένων, οι κόμβοι και οι ακμές στο Tensorflow είναι τα εξής (Packtpub, n.d.):

- Nodes (κόμβοι): Κάθε κόμβος αναπαριστά ένα στιγμιότυπο μιας μαθηματικής πράξης. Κάθε τέτοια έχει τουλάχιστον μία είσοδο και καμία ή περισσότερες εξόδους. Κάθε κόμβος στο tensorflow ονομάζεται op (εν συντομία για τη λέξη operation)
- Κάθε κόμβος αναπαριστά ένα στιγμιότυπο μιας μαθηματικής πράξης. Κάθε τέτοια μαθηματική λειτουργία έχει τουλάχιστον μία είσοδο και καμία ή περισσότερες εξόδους.
- Edges (ακμές): οι ακμές μπορούν να είναι είτε κανονικές ακμές (Normal Edges) είτε ειδικές ακμές (Special Edges)
  - Normal Edges: Είναι φορείς δομών δεδομένων (tensors), όπου η έξοδος μιας λειτουργίας (από έναν κόμβο) αποτελεί την είσοδο για μια άλλη λειτουργία.
  - Special Edges: Αυτές οι ακμές δεν είναι φορείς δεδομένων μεταξύ δύο κόμβων, αλλά χρησιμοποιούνται στο γράφημα ροής δεδομένων για τον καθορισμό ποιας λειτουργίας θα εφαρμοστεί πρώτη στα δεδομένα. Αν για παράδειγμα έχουμε δύο κόμβους, τον 1 και τον 2, οι οποίοι συνδέονται με μια ειδική ακμή, σημαίνει ότι το 2 θα ξεκινήσει τη λειτουργία του μόνο όταν τελειώσει η λειτουργία του 1.

Μαθηματικές πράξεις (Operations): Μια τέτοια πράξη αντιπροσωπεύει έναν υπολογισμό, όπως για παράδειγμα άθροισμα ή γινόμενο πινάκων. Ένα operation διαχειρίζεται και δρα σε τανυστές (tensors) διαφορετικών τύπων. Μια λειτουργία λαμβάνει $m \geq 0$ tensors ως είσοδο και παράγει $n \geq 0$ tensors ως έξοδο. Επιπλέον, έχει έναν ονομασμένο "τύπο" (όπως Const, MatMul ή Assign) και μπορεί να έχει μηδενικά ή περισσότερα χαρακτηριστικά χρόνου μεταγλώττισης που καθορίζουν τη συμπεριφορά της. Για παράδειγμα, η απλούστερη λειτουργία Const δεν έχει εισόδους και παράγει μόνο μια έξοδο ως αποτέλεσμα.

Τανυστές (tensors): Στο TensorFlow, όλα τα δεδομένα μοντελοποιούνται ως τανυστές ή τένσορες (δηλαδή πίνακες n διαστάσεων). Σε πολλούς αλγορίθμους μηχανικής μάθησης οι τανυστές αντιπροσωπεύουν τόσο τις εισόδους όσο και τα αποτελέσματα των μαθηματικών λειτουργιών. Για

παράδειγμα, ένας πολλαπλασιασμός πινάκων παίρνει ως είσοδο δύο δισδιάστατους τανυστές και παράγει έναν δισδιάστατο τανυστή. Οι τανυστές, όπως έχει τονιστεί είναι η βασική δομή δεδομένων στο Tensorflow. Οι τανυστές αναπαριστούν την πληροφορία που ρέει μέσα στις ακμές του γράφου ροής δεδομένων και προσδιορίζονται από τρία βασικά χαρακτηριστικά:

1. Βαθμό (rank): περιγράφει τη διάσταση κάθε τανυστή. Ένας δισδιάστατος πίνακας αποτελεί έναν tensor με βαθμό 2 ενώ ένα διάνυσμα έναν tensor με βαθμό 1.
2. Σχήμα (shape): περιγράφει το μέγεθος της κάθε διάτασης του τανυστή (σε ένα δισδιάστατο πίνακα περιγράφει τον αριθμό των γραμμών και στηλών).
3. Τύπο (type): περιγράφει τον τύπο των δεδομένων (int32, float κλπ.) του τανυστή.

Στη συνέχεια παρουσιάζονται παραδείγματα δήλωσης τανυστών σε συνδυασμό με την εικονική αναπαράστασή τους.

1. Τανυστής που περιλαμβάνει μια μοναδική τιμή και έχει τύπο δεδομένων int32

```
rank_0_tensor = tf.constant(4)
print(rank_0_tensor)
```

```
tf.Tensor(4, shape=(), dtype=int32)
```

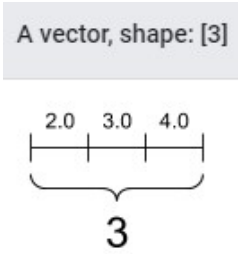
A scalar, shape: []

4

2. Τανυστής διάνυσμα με βαθμό 1 (rank-1) και τύπο δεδομένων float:

```
rank_1_tensor = tf.constant([2.0, 3.0, 4.0])
print(rank_1_tensor)
```

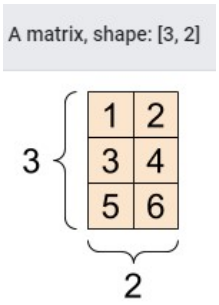
```
tf.Tensor([2. 3. 4.], shape=(3,), dtype=float32)
```



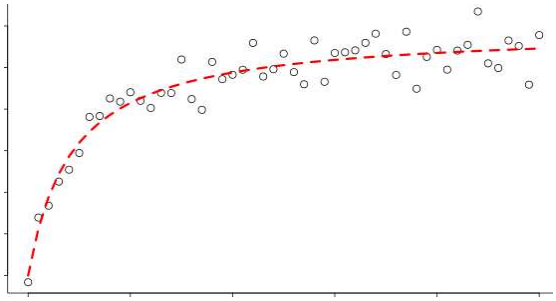
3. Τανυστής δισδιάστατος πίνακας ή με βαθμό 2 ("rank-2") έχει 2 :

```
rank_2_tensor = tf.constant([[1, 2],
                             [3, 4],
                             [5, 6]], dtype=tf.float16)
print(rank_2_tensor)
```

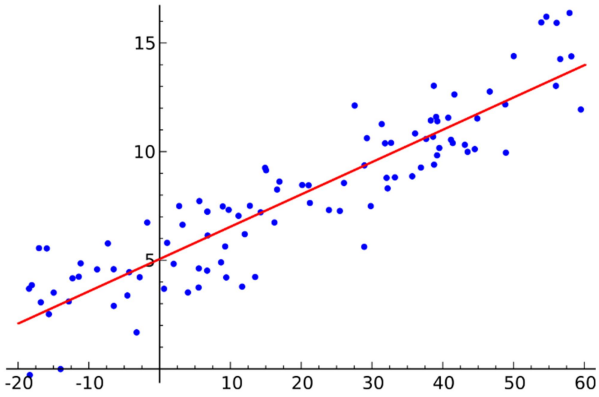
```
tf.Tensor(
[[1. 2.]
 [3. 4.]
 [5. 6.]], shape=(3, 2), dtype=float16)
```



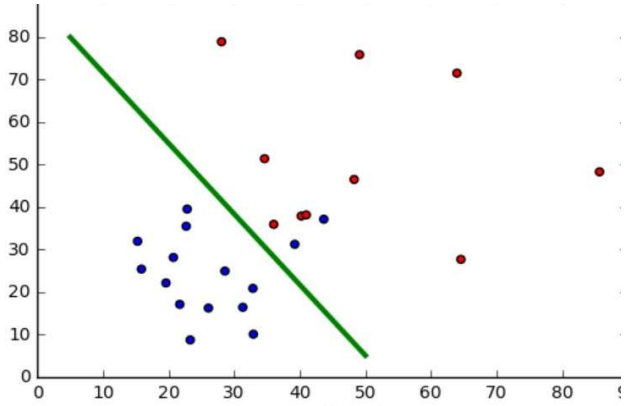
Η TensorFlow σε συνδυασμό με την Keras διευκολύνουν την ανάπτυξη νευρωνικών δικτύων για μηχανική μάθηση, όπως φαίνεται στην Εικόνα 6. Παράλληλα όμως, πέρα από τα νευρωνικά δίκτυα το Tensorflow, μπορεί να χρησιμοποιηθεί και άλλους σκοπούς όπως για να μοντελοποιηθεί η γραμμική/ μη γραμμική παλινδρόμηση (linear/non-linear regression)(Εικόνα 3, Εικόνα 4), λογιστική παλινδρόμηση (logistic regression)(Εικόνα 5) καθώς και άλλα μοντέλα.



Εικόνα 3: Non linear regression



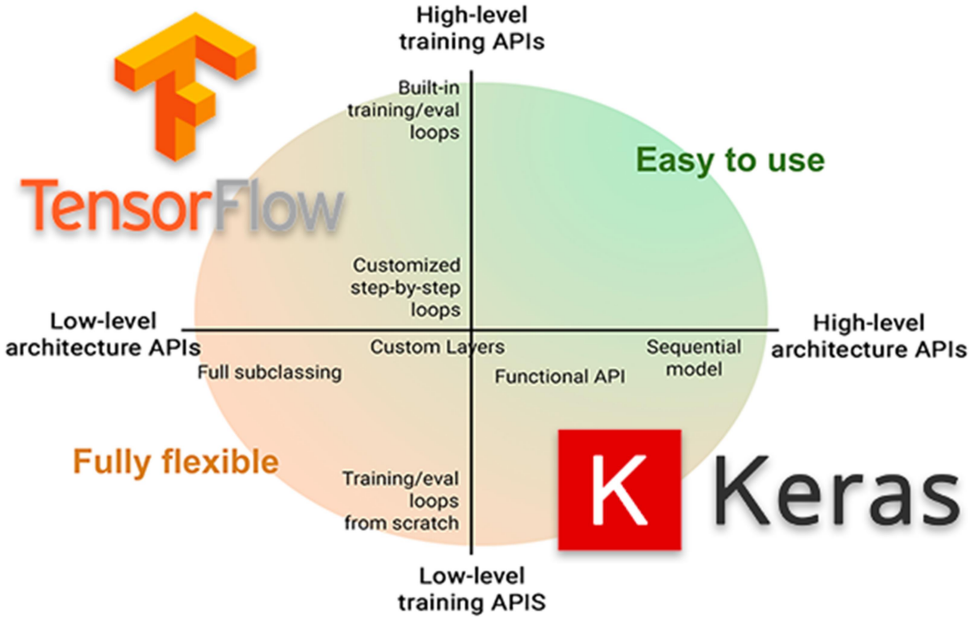
Εικόνα 4: Linear regression



Εικόνα 5: Logistic regression

2.2.2 Tensorflow και Τεχνητά Νευρωνικά Δίκτυα

Όπως προαναφέρθηκε, το TensorFlow σε συνδυασμό με την Keras διευκολύνουν την ανάπτυξη νευρωνικών δικτύων για μηχανική μάθηση. Στην υπο-ενότητα αυτή θα επεξηγηθεί η έννοια των τεχνητών νευρωνικών δικτύων και στη συνέχει πως επιτυγχάνεται στην πράξη η ανάπτυξή τους αξιοποιώντας το Tensorflow.



Εικόνα 6:Το Keras αποτελεί το επίσημο backend του TensorFlow

Με τον όρο νευρωνικό δίκτυο γίνεται αναφορά στη νευροβιολογία, παρόλο όμως ορισμένες έννοιες την «βαθιάς μάθησης» αναπτύχθηκαν εν μέρει αντλώντας έμπνευση από την κατανόηση του εγκεφάλου, τα μοντέλα βαθιάς μάθησης δεν μοντελοποιούν την λειτουργία του εγκεφάλου. Για την ακρίβεια δεν υπάρχουν ενδείξεις ότι ο εγκέφαλος εφαρμόζει κάτι παρόμοιο με τους μηχανισμούς μάθησης που χρησιμοποιούνται στα σύγχρονα deep learning μοντέλα.

Με τον όρο τεχνητό νευρωνικό δίκτυο αναφερόμαστε σε έναν παράλληλο και κατανεμημένο επεξεργαστή ο οποίος αποτελείται από ένα πλήθος ανεξάρτητων υπολογιστικών μονάδων, τους νευρώνες, οι οποίοι είναι διασυνδεδεμένοι μεταξύ τους σε στρώματα – επίπεδα (layers) (Λιβιέρης, 2012).

Η προσέγγιση της μηχανικής μάθησης περιλαμβάνει την χρήση ενός τεχνητού νευρωνικού δικτύου ώστε να «μάθει» ένα σύστημα τον τρόπο με τον οποίο οι είσοδοι σχετίζονται με τις εξόδους. Άρα είναι γνωστοί οι είσοδοι και οι έξοδοι και το σύστημα μαθαίνει τον αλγόριθμο που τις «συνδέει». Αντιθέτως, στον «παραδοσιακό» προγραμματισμό είναι γνωστή η είσοδος και ο αλγόριθμος που παράγει την έξοδο. Ένα νευρωνικό δίκτυο είναι ουσιαστικά μια στοίβα από επίπεδα (layers), όπου κάθε επίπεδο αποτελείται από κάποια μαθηματική λειτουργία και από εσωτερικές μεταβλητές (“weights” και “biases”) οι οποίες ενημερώνονται κατά τη διάρκεια της εκπαίδευσης. Με τον όρο «εκπαίδευση» γίνεται αναφορά στη διαδικασία κατά την οποία το νευρωνικό δίκτυο καλείται επανειλημμένα να αντιστοιχίσει μια είσοδο με μια έξοδο.

Σε ένα τεχνητό νευρωνικό δίκτυο ο τρόπος με τον οποίο οι νευρώνες δομούνται ονομάζεται τοπολογία/ αρχιτεκτονική. Όλοι οι νευρώνες είναι τοποθετημένοι σε layers, όπου οι νευρώνες στο ίδιο στρώμα έχουν την ίδια συνάρτηση ενεργοποίησης και συμπεριφέρονται με τον ίδιο τρόπο. Οι συναρτήσεις ενεργοποίησης που χρησιμοποιούνται συνήθως παρουσιάζονται στον Πίνακας 1 (Βάρσος, 2018). Ένα νευρωνικό δίκτυο έχει τουλάχιστον 2 layers, το input layer και το output layer. Τα ενδιάμεσα στρώματα είναι τα λεγόμενα hidden layers.

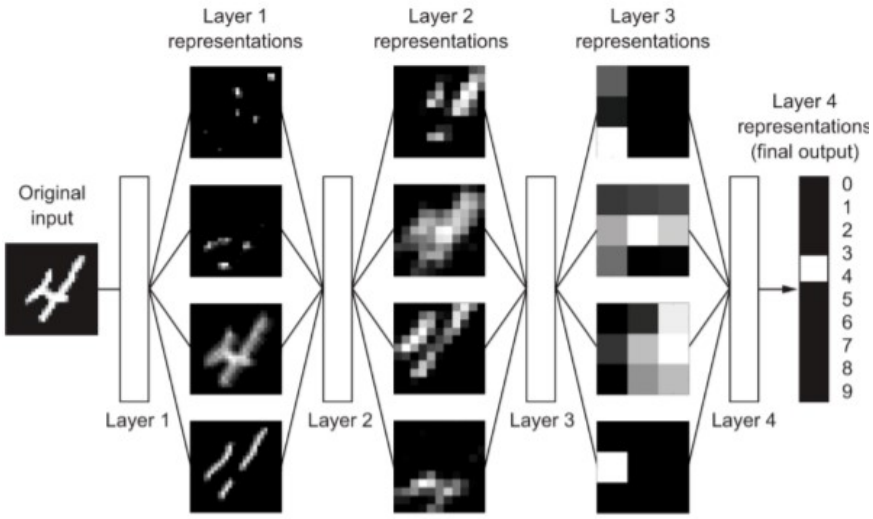
Πίνακας 1: Συναρτήσεις Ενεργοποίησης

Όνομα συνάρτησης	**Συνάρτηση**
Βηματική συνάρτηση	$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$
Γραμμική συνάρτηση	$\underline{f(x) = x}$
Σιγμοειδής συνάρτηση	$f(x) = \frac{1}{1 + e^{-x}}$
Υπερβολική συνάρτηση	$f(x) = tanh(x)$

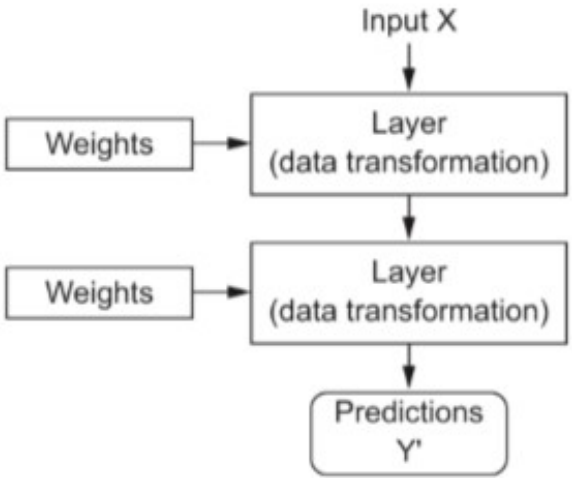
Η επιλογή της συνάρτησης ενεργοποίησης είναι ιδιαίτερα σημαντική για την αποτελεσματικότητα του τεχνητού νευρωνικού δικτύου. Το Tensorflow παρέχει ιδιαίτερη ευελιξία στον χρήστη καθώς του δίνει την δυνατότητα να επιλέξει τη δομή του νευρωνικού δικτύου καθώς και των συναρτήσεων που χρησιμοποιούνται για την επεξεργασία των δεδομένων.

Υπάρχουν πολλοί τύποι αρχιτεκτονικών νευρωνικών δικτύων. Ωστόσο, ανεξάρτητα από την αρχιτεκτονική, τα μαθηματικά που περιέχει (δηλαδή ποιοι υπολογισμοί εκτελούνται και με ποια σειρά) δεν τροποποιούνται κατά τη διάρκεια της εκπαίδευσης σε αντίθεση με τις εσωτερικές μεταβλητές που τροποποιούνται. Ένας αλγόριθμος μηχανικής μάθησης θα μπορούσε πολύ αφαιρετικά να οριστεί ως μια συνάρτηση που συντονίζει κάποιες μεταβλητές ώστε να αντιστοιχίσει σωστά κάποια δεδομένα εισόδου με κάποια δεδομένα εξόδου.

Στις εικόνεςΕικόνα 7 και Εικόνα 8 παρουσιάζεται σχηματικά πως ένα νευρωνικό δίκτυο μετασχηματίζει την μια εικόνα ενός χειρόγραφου αριθμού για να αναγνωρίσει τον εν λόγω αριθμό (Chollet, 2020).



Εικόνα 7:Νευρωνικό δίκτυο για την αναγνώριση χειρόγραφου αριθμού

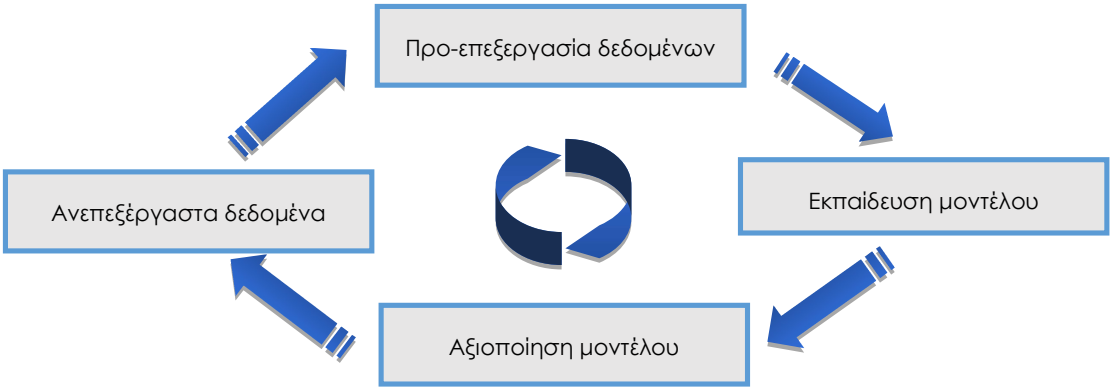


Εικόνα 8: Στόχος είναι η εύρεση των κατάλληλων τιμών για τα βάρη.

2.2.3 Δημιουργία ενός machine learning μοντέλου

Στην υποενότητα αυτή παρουσιάζονται η βασική δομή ανάπτυξης ενός μοντέλου μηχανικής μάθησης αξιοποιώντας το Tensorflow. Συνήθως, η διαδικασία ανάπτυξης ενός συστήματος μηχανικής μάθησης περιλαμβάνει την συλλογή και προεπομασία

των δεδομένων, την εκπαίδευση του μοντέλου με τα προαναφερθέντα δεδομένα και τέλος την αξιοποίηση του εκπαιδευμένου μοντέλου.



Εικόνα 9: ML Lifecycle

Κατά την ανάπτυξη ενός machine learning μοντέλου με το Tensorflow και το Keras πραγματοποιούνται τα παρακάτω πέντε βήματα :

1. **Ορισμός του μοντέλο**

   Ο ορισμός του μοντέλου αρχικά τον προσδιορισμό του τύπου του και στη συνέχεια την επιλογή της αρχιτεκτονικής/τοπολογίας του δικτύου. Ένα μοντέλο «βαθιάς» μάθησης αποτελεί ένα γράφημα από επίπεδα (layers). Στο βήμα αυτό καθορίζονται τα layers του μοντέλου, διαμορφώνεται ο αριθμός των nodes και η συνάρτηση ενεργοποίησης του κάθε επιπέδου και πραγματοποιείται οι σύνδεση των layers μεταξύ τους. Υπάρχουν τρεις διαφορετικοί τρόποι με τους οποίους μπορούν να δημιουργηθούν τα μοντέλα με το με Keras και το TensorFlow 2.0.

   - *Sequential API:* αποτελεί τον ευκολότερο τρόπος δημιουργίας μοντέλων με το Keras.
   - *Functional API:* χρησιμοποιείται για την ανάπτυξη πιο σύνθετων μοντέλων, ιδίως για μοντέλα με πολλαπλές εισόδους ή εξόδους.
   - *Mode Subclassing*: χρησιμοποιείται για την ανάπτυξη πλήρως προσαρμόσιμων μοντέλων.

Sequential μοντέλα (Σειριακό μοντέλα)

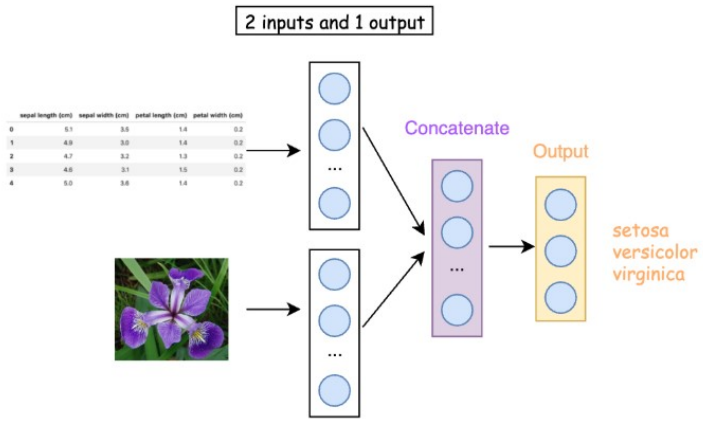
Τα σειριακά μοντέλα περιλαμβάνουν τον ορισμό μια Sequential κλάσης και την προσθήκη επιπέδων στο μοντέλο ένα προς ένα, με γραμμικό τρόπο. Με άλλα λόγια ένα σειριακό μοντέλο είναι κατάλληλο για μια απλή στοίβα επιπέδων όπου κάθε επίπεδο έχει ακριβώς έναν τανυστή ως εισόδου και έναν τανυστή ως εξόδου. Παρακάτω παρουσιάζεται η αρχιτεκτονική ενός sequential μοντέλου.

```
model = models.Sequential([
  layers.Dense(512, activation='relu'),
  layers.Dense(10, activation='softmax')
])
```

Όπως έχει προαναφερθεί, το βασικό δομικό στοιχείο των τεχνητών νευρωνικών δικτύων είναι το layer, το οποίο μπορεί να θεωρηθεί ως μια μονάδα επεξεργασίας των δεδομένων που τα «φιλτράρει» ώστε να εξέρχονται από αυτό σε μια «χρήσιμη» μορφή. Στον παραπάνω κώδικα, το μοντέλο αποτελείται από δύο στρώματα, τα οποία είναι «πυκνά» συνδεδεμένα νευρωνικά στρώματα (τα λεγόμενα dense layers). Το softmax layer, το δεύτερο δηλαδή στρώμα του μοντέλου θα επιστρέψει μια σειρά από 10 probability scores (άθροισμα σε 1). Αναφερόμενοι στο παράδειγμα που παρουσιάζεται στην Εικόνα 7, το κάθε σκορ θα αποτελεί την πιθανότητα η εικόνα να αντιστοιχεί στον κάθε έναν από τους 10 αριθμούς (0 έως 9).

Functional Model API

Τα functional μοντέλα μπορεί να είναι πιο περίπλοκα αλλά την ίδια στιγμή είναι και πιο ευέλικτα. Το Functional API διευκολύνει τον χειρισμό πολλαπλών εισόδων και εξόδων σε ένα μοντέλο, κάτι το οποίο δεν συμβαίνει με το Sequential API. Τρία διαφορετικά σενάρια είναι ένα μοντέλο με 2 εισόδους και 1 έξοδο, ε'ενα μοντέλο με 1 είσοδο και 2 εξόδους και ένα μοντέλο με 2 εισόδους και 2 εξόδους. Στην Εικόνα 10 (Chen, 2020) παρουσιάζεται η μορφή ενός τεχνητού νευρωνικού δικτύου αντίστοιχο με αυτό που αναπτύχθηκε και παρουσιάζεται στο συγκεκριμένο παράρτημα, με 2 εισόδους και μια έξοδο.



Εικόνα 10: Παράδειγμα νευρωνικού δικτύου με 2 εισόδους και 1 έξοδο

Στα μοντέλα αυτά η έξοδος ενός επιπέδου αποτελεί την είσοδο για ένα άλλο επίπεδο. Παρακάτω παρουσιάζεται ο ορισμός ενός μοντέλου τέτοιου τύπου. Αρχικά, προσδιορίζεται μέσω της κλάσης Input ένα επίπεδο εισόδου. Στη συνέχεια ένα πλήρως διασυνδεδεμένο (fully connected) επίπεδο μπορεί να συνδεθεί με την με το input layer και στο επόμενο βήμα να συνδεθεί με τον ίδιο τρόπο σε ένα output layer. Μόλις συνδεθούν όλα τα layers με τον προαναφερθέντα τρόπο μπορεί να οριστεί ένα αντικείμενο Μοντέλου (model object) και να καθοριστούν τα επίπεδα εισόδου και εξόδου.

```
x_in = Input(shape=(8,))

x = Dense(10)(x_in)

x_out = Dense(1)(x)

# Προσδιορισμός του model object

model = Model(inputs=x_in, outputs=x_out)
```

Model Subclassing

Ο τελευταίος τρόπος ανάπτυξης ενός μοντέλου Machine Learning με το Keras και το TensorFlow 2.0 ονομάζεται Model Subclassing. Αυτός ο τρόπος κατασκευής μοντέλων προσφέρει «low level» έλεγχο τόσο στην κατασκευή όσο και στη λειτουργία ενός μοντέλου. Η δομή ενός τέτοιου μοντέλου φαίνεται στη συνέχεια:

```
class CustomModel(Model):
    def __init__(self, **kwargs):
        super(CustomModel, self).__init__(**kwargs)
        self.dense1 = Dense(5, activation='relu', )
        self.dense2 = Dense(10, activation='relu')
        self.dense3 = Dense(3, activation='softmax')
    def call(self, inputs):
        x = self.dense1(inputs)
        x = self.dense2(x)
        return self.dense3(x)
my_custom_model = CustomModel(name='my_custom_model')
```

Αν και το Model Subclassing δίνει χαμηλού επιπέδου έλεγχο τόσο στην κατασκευή όσο και στη λειτουργία ενός μοντέλου, η ευελιξία του μπορεί να οδηγήσει σε περισσότερα σφάλματα. Επιπλέον, η χρήση του είναι πολύ πιο δύσκολη από αυτή του του Sequential API και του Functional API. Το Model Subclassing χρησιμοποιείται κυρίως από ερευνητές για την εφαρμογή πρωτοπόρων και πολύπλοκων αρχιτεκτονικών.

2. Μεταγλώττιση του μοντέλου

Όταν η αρχιτεκτονική του μοντέλου έχει οριστεί καλείται η μέθοδος compile στην οποία ορίζονται τρεις παράμετροι:

- optimizer : Ο optimizer ή αλλιώς βελτιστοποιητής είναι ο μηχανισμός μέσω του οποίου το μοντέλο θα ενημερωθεί βάσει των δεδομένων εκπαίδευσης που δέχεται ως είσοδο ώστε να βελτιώσει την απόδοσή του.
- loss : Η συνάρτηση που πρέπει να ελαχιστοποιείται κατά τη βελτιστοποίηση. Αποτελεί έναν τρόπο με τον οποίο το μοντέλο μετράει την απόδοσή του κατά τη διάρκεια της εκπαίδευσης.
- metrics: Χρησιμοποιείται για την παρακολούθηση της διαδικασίας εκπαίδευσης και επαλήθευσης.

Μια πιθανή μέθοδος compile για το μοντέλο που παρουσιάζεται στην εικόνα Εικόνα 7 είναι η εξής:

```
model.compile(optimizer='rmsprop',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

3. Εκπαίδευση του μοντέλου

Το επόμενο βήμα αφορά την εκπαίδευση του μοντέλου, η οποία επιτυγχάνεται με την μέθοδο fit. Η ονομασία της εν λόγω μεθόδου προκύπτει μεταφορικά από το γεγονός ότι προσπαθούμε να «ταιριάξουμε-χωρέσουμε» το μοντέλου στα δεδομένα εκπαίδευσης. Η διαδικασία της εκπαίδευσης απαιτεί και τον περισσότερο χρόνο (από λίγα λεπτά έως και μέρες) και δέχεται τα παρακάτω ορίσματα:

- epochs : Η μάθηση οργανώνεται σε εποχές (epochs). Με τον όρο «εποχή» γίνεται αναφορά σε μια επανάληψη σε όλο τον όγκο των δεδομένων εισόδου. Είναι προφανές ότι όσο μεγαλύτερη είναι αυτή η τιμή τόσο περισσότερος χρόνος απαιτείται κατά την εκπαίδευση.
- batch_size : Το μοντέλο τα τεμαχίζει σε (μικρότερες) "δεσμίδες" τα δεδομένα εισόδου και εκτελεί επαναλήψεις πάνω σε αυτές κατά τη διαδικασία της εκμάθησης. Το όρισμα αυτό καθορίζει των αριθμό των δειγμάτων που θα περιέχει η κάθε «δεσμίδα».

  Για να γίνει πιο κατανοητή αυτή η έννοια θα δοθεί ένα παράδειγμα. Άν υποθέσουμε ότι το συνολικό πλήθος δειγμάτων εκπαίδευσης είναι 1050 οριστεί το batch_size ίσο με 100, ο αλγόριθμος θα παίρνει τα πρώτα 100 δείγματα από το πλήρες σύνολο δεδομένων εκπαίδευσης και θα εκπαιδεύσει με αυτά το δίκτυο. Στη συνέχεια, παίρνει τα δεύτερα 100 δείγματα (από το 101 έως 200) και εκπαιδεύει ξανά το δίκτυο κοκ. Τα πλεονεκτήματα που προκύπτουν από τον ορισμό του batch_size είναι ότι η μνήμη που απαιτείται είναι λιγότερη, καθώς και ο χρόνος εκπαίδευσης. Από την άλλη πλευρά, ένα μειονέκτημα που προκύπτει όταν ορίζεται το batch_size είναι ότι το αποτέλεσμα μπορεί να μην είναι τόσο ικανοποιητικό.

- validation_data : Χρησιμοποιώντας τα δεδομένα επαλήθευσης που του δίνονται, το μοντέλο μπορεί να παρακολουθήσει την απόδοσή του στο τέλος κάθε «εποχής». Επίσης, πέρα από το εν λόγω όρισμα μπορούν να οριστεί η μεταβλητή validation_split η οποία παίρνει μια τιμή από το 0 έως το 1 η οποία ορίζει τα δεδομένα που αντί για εκπαίδευση θα αξιοποιηθούν για αξιολόγηση.

```
model.fit(data, labels, epochs=10, batch_size=32,
          validation_data=(val_data, val_labels))
```

Κατά την εκπαίδευση του μοντέλου, υπάρχει μια έννοια που αξίζει να τονιστεί, καθώς είναι ιδιαίτερα σημαντική για την παραγωγή προβλέψεων. Η έννοια αυτή είναι το «overfitting» . Ουσιαστικά, «overfitting» προκύπτει όταν το μοντέλο ταιριάζει υπερβολικά στα δεδομένα εκπαίδευσης. Κάτι τέτοιο όμως μπορεί να φέρει αρνητικά αποτελέσματα καθώς ο στόχος είναι η ανάπτυξη μοντέλων που «γενικεύονται» καλά σε ένα σύνολο δεδομένων εκπαίδευσης.

Το αντίθετο του overfitting ονομάζεται underfitting το οποίο επίσης θέλουμε να αποφευχθεί. Το underfitting συμβαίνει όταν υπάρχει ακόμη περιθώριο βελτίωσης του μοντέλου σε σχέση με τα δεδομένα εκπαίδευσης. Αυτό μπορεί να συμβεί για διάφορους λόγους όπως για παράδειγμα εάν το μοντέλο δεν είναι αρκετά ισχυρό, εάν είναι υπερβολικά ρυθμισμένο ή απλά δεν έχει εκπαιδευτεί αρκετά. Με άλλα λόγια το underfitting μεταφράζεται σε ένα μοντέλο το οποίο δεν έχει «μάθει» τα μοτίβα (patterns) που υπάρχουν στα δεδομένα εκπαίδευσης.

4. Αξιολόγηση του μοντέλου

Η αξιολόγηση του μοντέλου έχει ως στόχο την επαλήθευση της αποτελεσματικότητάς του. Για να γίνει η εν λόγω αξιολόγηση αρχικά απαιτείται ο ορισμός ενός συνόλου δεδομένων, τα οποία δεν έχουν χρησιμοποιηθεί στη διαδικασία της εκπαίδευσης. Με τον τρόπο αυτό, εκτιμάται η απόδοση του μοντέλου όταν του ζητείται να παράξει προβλέψεις με άγνωστα δεδομένα. Η ταχύτητα της αξιολόγησης του μοντέλου είναι ανάλογη με την ποσότητα των

δεδομένων που του δίνονται, παραμένει όμως πολύ ταχύτερη από την εκπαίδευση καθώς το μοντέλο δεν αλλάζει.

```
loss = model.evaluate(X, y, verbose=0)
```

5. Παραγωγή προβλέψεων με το μοντέλο

Η παραγωγή προβλέψεων είναι το τελευταίο βήμα στον κύκλο ζωής ενός μοντέλου και ουσιαστικά ο λόγος για τον οποία παράχθηκε το μοντέλο. Για να γίνουν προβλέψεις, είναι αναμενόμενο να απαιτούνται καινούρια δεδομένα, για τα οποία το μοντέλο δεν έχει εκπαιδευτεί. Προβλέψεις μπορούν να γίνουν είτε απευθείας μετά την εκπαίδευση του μοντέλου, είτε αφού σωθεί το μοντέλο.

```
predictions = model.predict([test.user_id, test.poi_id])
```

2.3 Πλεονεκτήματα

Υπάρχει μια πληθώρα πλεονεκτημάτων που καθιστά το tensorflow ένα ιδανικό εργαλείο για την ανάπτυξη εφαρμογών τεχνητής νοημοσύνης και machine learning. Αρχικά, είναι σημαντικό να τονιστεί ότι το TensorFlow είναι κάτι περισσότερο από μια απλή βιβλιοθήκη. Είναι, ουσιαστικά μια πλατφόρμα, που αποτελείται από ένα τεράστιο αριθμό από components (συστατικά στοιχεία), ορισμένα εκ των οποίων έχουν αναπτυχθεί από την Google ενώ άλλα από τρίτους. Για παράδειγμα, υπάρχει TF-Agents το οποία είναι μια reinforcement learning library στο tensorflow που διευκολύνει το σχεδιασμό, την εφαρμογή και τη δοκιμή νέων reinforcement learning αλγορίθμων RL (Google, 2020). Άλλα παραδείγματα είναι το TF-Serving για παραγωγή, υπάρχει το TF-Hub και άλλα τα οποία όλα μαζί καλύπτουν ένα πολύ ευρύ φάσμα περιπτώσεων περιπτώσεις .

Επιπλέον, ένα άλλο πλεονέκτημα του Tensorflow είναι πως τα μοντέλα του μπορούν να εξαχθούν και να εκτελεστούν σε μια πληθώρα συστημάτων. Για παράδειγμα μπορούν να εξαχθούν σε C ++, σε JavaScript (για εφαρμογές που βασίζονται σε πρόγραμμα περιήγησης) ή TFLite (για εφαρμογές που εκτελούνται σε κινητές συσκευές) κ.λπ.

Το tensorflow παρέχει μεγάλη ευελιξία και μπορεί να αξιοποιηθεί για μια πληθώρα έργων όπως η ταξινόμηση και η επεξεργασία εικόνων, παραγωγή συστάσεων κ.α. Παραδείγματα χρήσης της βιβλιοθήκης αυτής είναι η Airbnb, που την αξιοποιεί για την κατηγοριοποίηση εικόνων και τον εντοπισμό αντικειμένων σε αυτές για να βελτιώσει το User Experience. Ένα άλλο παράδειγμα είναι η Airbus η οποία επεξεργάζεται εικόνες από δορυφόρους για διάφορους σκοπούς όπως ο εντοπισμός παράνομης ανοικοδόμησης κ.α. ενώ το Twitter το οποίο χρησιμοποιεί το λογισμικό αυτό για την εξατομικευμένη ταξινόμηση των tweets που παρουσιάζονται στον χρήστη.

3 Σύστημα Συστάσεων ARPolis

Στην ενότητα αυτή παρουσιάζεται ένα πρότυπο μοντέλο που δημιουργήθηκε αξιοποιώντας το tensorflow. Το εν λόγω μοντέλο αξιοποιεί μια collaborative filtering μέθοδο για να παράγει προβλέψεις και εν τέλει συστάσεις στο χρήστη. Θα παρουσιαστεί στη συνέχεια η λογική του μοντέλου σε συνδυασμό με τον κώδικα που χρησιμοποιήθηκε. Πιο συγκεκριμένα, τα δεδομένα που αξιοποιούνται αφορούν τις βαθμολογίες ενός χρήστη για συγκεκριμένα αξιοθέατα – σημεία ενδιαφέροντος (Points of Interest). Τα δεδομένα αυτά, στο μεγαλύτερο ποσοστό τους αξιοποιούνται για να εκπαιδευτεί το μοντέλο, και ένα πολύ μικρό μέρος από αυτά για να επαληθευθεί η αποτελεσματικότητα του μοντέλου.

3.1 Δημιουργία του machine learning μοντέλου για παραγωγή συστάσεων

Για την δημιουργία ενός συστήματος συστάσεων αναπτύχθηκαν δύο διαφορετικής αρχιτεκτονικής μοντέλα αξιοποιώντας το Tensorflow σε συνδυασμό με το Keras. Όπως έχει ήδη αναφερθεί, το Το tf.keras καθιστά το TensorFlow ευκολότερο στη χρήση, χωρίς να θυσιάζει τη προσαρμοστικότητα και την απόδοση. Τα μοντέλα αναπτύχθηκαν με τη χρήση του Keras functional API.

Στόχος του κάθε μοντέλου είναι να αξιοποιηθεί σε ένα σύστημα συστάσεων στον τουρισμό. Για να επιτευχθεί αυτό, το μοντέλο θα εκπαιδευτεί σε ένα πλήθος αξιολογήσεων που έχουν πραγματοποιήσει τουρίστες σε σημεία ενδιαφέροντος. Έστω ότι τα δεδομένα αυτά είναι ένας πίνακας που περιλαμβάνει τρεις στήλες το poi_id, το οποίο είναι ένας μοναδικός αριθμός που αντιστοιχεί σε ένα σημείο ενδιαφέροντος, το user_id, που είναι ένας μοναδικός αριθμός για τον κάθε χρήστη και το rating που αποτελεί την αξιολόγηση του κάθε χρήστη για ορισμένα αξιοθέατα και είναι ένας ακέραιος από το 1 έως το 5. Στον Πίνακας 2 φαίνεται ένα μικρό δείγμα της μορφής που έχουν τα δεδομένα που αξιοποιεί το μοντέλο για εκπαίδευση.

Πίνακας 2: Παράδειγμα μορφής πίνακα

```
1
2 dataset=pd.read_csv('/content/drive/My Drive/ratings.csv')
3 print(dataset)
```

```
        poi_id  user_id  rating
0            1      314       5
1            1      439       3
2            1      588       5
3            1     1169       4
4            1     1185       4
...        ...      ...     ...
981751   10000    48386       5
981752   10000    49007       4
981753   10000    49383       5
981754   10000    50124       5
981755   10000    51328       1

[981756 rows x 3 columns]
```

Για να μπορέσει να εκπαιδευτεί επιτυχώς ένα μοντέλο θα πρέπει το πλήθος των δεδομένων να είναι αρκετά μεγάλο. Από την άλλη πλευρά το μοντέλο δε θα πρέπει να εκπαιδευτεί «υπερβολικά» για να αποφευχθεί το overfitting που παρουσιάστηκε στην προηγούμενη ενότητα. Τα δεδομένα που προαναφέρθηκαν είναι κοντά στις 900.000 και χρησιμοποιούνται κατά κύριο λόγο για training, με ένα μικρό ποσοστό να χρησιμοποιείται για να επαληθευτεί η ικανοποιητική λειτουργία του μοντέλου.

ΑΝΑΓΝΩΣΗ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΤΗΝ ΕΚΠΑΙΔΕΥΣΗ/ΑΞΙΟΛΟΓΗΣΗ ΤΟΥ ΜΟΝΤΕΛΟ

```
#Τα δεδομένα φορτώνονται από ένα csv αρχείο χρησιμοποιώντας τη βιβλιοθήκη Pandas

dataset = pd.read_csv('data/ratings.csv')

#Στο σημείο αυτό το dataset χωρίζεται σε training data και σε test data.

train, test = train_test_split(dataset, test_size=0.00002)



#Οι παρακάτω δύο μεταβλητές θα εμπεριέχουν τον αριθμό των χρηστών και των σημείων ενδιαφέροντος που υπάρχουν στο csv αρχείο. n_users= 53424, n_poi= 10000

n_users = len(dataset.user_id.unique())

n_poi = len(dataset.poi_id.unique())
```

Η «ανάγνωση» και προσωρινή αποθήκευση των δεδομένων σε ένα πλαίσιο δεδομένων (dataframe) γίνεται με τη βιβλιοθήκη Pandas, που αποτελεί μια βιβλιοθήκη Python με πολλές χρήσιμες λειτουργίες για φόρτωση επεξεργασία δομημένων δεδομένων. Στη συνέχεια, τα δεδομένα αυτά θα χρησιμοποιηθούν για να εκπαιδευτούν

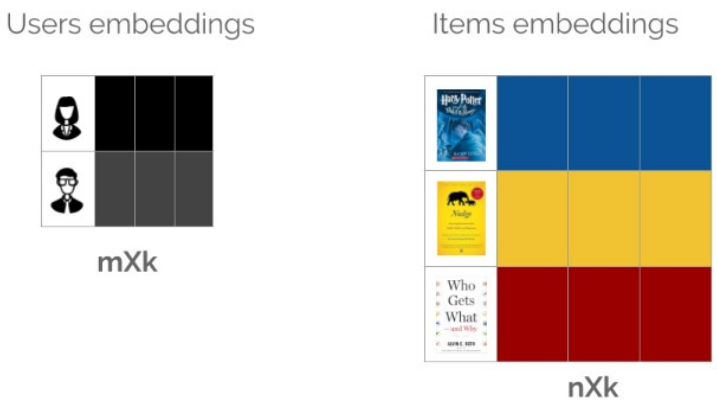
δύο διαφορετικά μοντέλα, με διαφορετική αρχιτεκτονική. Και τα δυο μοντέλα χρησιμοποιούν «embeddings» για την αναπαράσταση των δεδομένων. Στόχος είναι να παρουσιαστούν τόσο τα μοντέλα σε συνδυασμό με τις μεθόδους που ακολουθούν καθώς και οι προβλέψεις που παράγουν. Οι δύο μέθοδοι είναι οι εξής:

- Matrix factorization
- Tabular data method

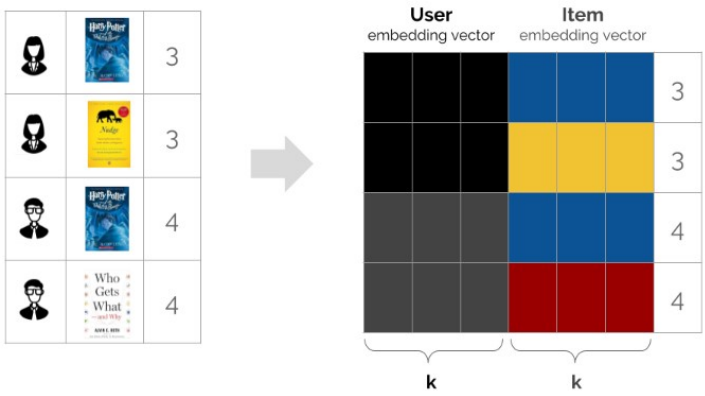
Όπως προαναφέρθηκε, τα μοντέλα αναπτύχθηκαν με τη χρήση του Keras functional API. Σε αντίθεση με το Sequential API , για την ανάπτυξη ενός μοντέλου χρησιμοποιώντας το Functional API πρέπει πρώτα να οριστεί ένα αυτόνομο στρώμα εισόδου (input layer) που καθορίζει το «σχήμα» των δεδομένων που εισάγονται στο μοντέλο. Το embedding layer αποτελεί το δεύτερο στρώμα του δικτύου μετά το (input layer). Στο στρώμα αυτό γίνεται η χαρτογράφηση διακριτών αντικειμένων σε ένα διάνυσμα συνεχών τιμών. Στη συγκεκριμένη περίπτωση τα εν λόγω αντικείμενα είναι τα αναγνωριστικά των σημείων ενδιαφέροντος και τα αναγνωριστικά των χρηστών. Η προαναφερθείσα λειτουργία μπορεί να χρησιμοποιηθεί για να εντοπιστούν τυχών ομοιότητες μεταξύ των διακριτών αντικειμένων, που δεν θα ήταν εμφανή στο μοντέλο χωρίς την χρήση των embedding layers. Τα διανύσματα ενσωμάτωσης (embedding vectors) ενημερώνονται κατά την εκπαίδευση του δικτύου. Η μεταβλητή features_num ορίζει τα latent factors τόσο για τους χρήστες όσο και τα σημεία ενδιαφέροντος.

3.1.1 Μέθοδος Tabular data

Έστω ότι έχουμε τους παρακάτω πίνακες που φαίνονται στην Εικόνα 11. Τα διανύσματα που σχετίζονται με τον κάθε χρήστη και το κάθε αντικείμενο στη μέθοδο tabular data (μέθοδος πίνακα δεδομένων) συνδυάζονται σε έναν πίνακα με δομή user_vector + poi_vector ο οποίος φαίνεται στην Εικόνα 12 .



Εικόνα 11;User/item embeddings



Εικόνα 12: Tabular data

ΔΗΜΙΟΥΡΓΙΑ INPUT & EMBEDDING LAYERS

```
# Users embedding features.
user_input = Input(shape=[1])
user_embedding = Embedding(n_users + 1, features_num)(user_input)
user_vector = Flatten()(user_embedding)


# Pois embedding features.
poi_input = Input(shape=[1])
poi_embedding = Embedding(n_poi + 1, features_num)(poi_input)
poi_vector = Flatten()(poi_embedding)
```

```
# Συνδυασμένα δεδομένα εισόδου
concatenated = Concatenate()([poi_vector, user_vector])
```

Το επόμενο βήμα είναι η δημιουργία του μοντέλου. Η οικοδόμηση του μοντέλου της βαθιάς μάθησης ακολουθεί σε γενικές γραμμές ένα παράδειγμα το οποίο παρέχεται στους επίσημους ιστοχώρους του Keras και του Tensorflow (Tensorflow, n.d.) (Keras, n.d.). Αναλυτικά, τα κύρια στοιχεία του μοντέλου βαθιάς μάθησης που θα χρησιμοποιηθεί είναι τα εξής:

- 3 Πλήρως Συνδεδεμένα επίπεδα (Dense Layers)
- 2 στρώματα απόσυρσης (dropout). Στο επίπεδο αυτό ένα ποσοστό νευρώνων τυχαία αποσύρεται, δηλαδή ορίζεται τιμή της εξόδου τους το 0, με σκοπό την αποφυγή της υπερεκπαίδευσης (overfitting). Στο παρακάτω μοντέλο αποσύρεται το 10% των δεδομένων εισόδου.

Κάθε στιγμιότυπο ενός layer, όπως φαίνεται, δέχεται ως είσοδο έναν τανυστή και επιστρέφει έναν τανυστή, ο οποίος αποτελεί την είσοδο του επόμενου layer.

ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ ΜΟΝΤΕΛΟΥ

```
# Δημιουργία μοντέλου
layer1 = Dense(128, activation='relu')(concatenated)
dropout1 = Dropout(0.1)(layer1)
layer2 = Dense(64, activation='relu')(dropout1)
dropout2 = Dropout(0.1)(layer2)

# Ορίζεται το επίπεδο εξόδου που περιλαμβάνει μόνο μια έξοδο.
output = Dense(1)(dropout2)
```

Μετά την σύνδεση όλων των layers με τον προαναφερθέντα τρόπο μπορεί να οριστεί ένα αντικείμενο Μοντέλου (model object) και να καθοριστούν τα επίπεδα εισόδου και εξόδου όπως φαίνεται παρακάτω.

```
model = Model([user_input, poi_input], output)
```

Μετά τον ορισμό της αρχιτεκτονικής του μοντέλου καλείται οι μέθοδος compile και στην συνέχεια ξεκινάει η εκπαίδευση του μοντέλου.

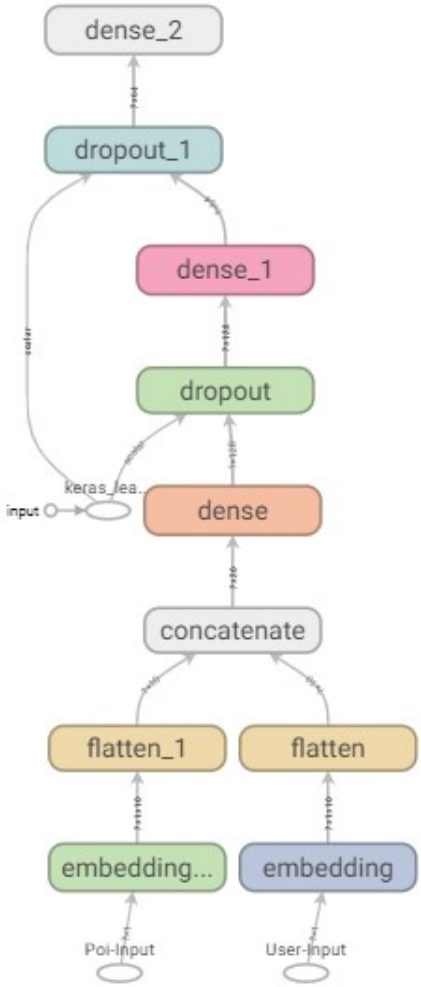
ΜΕΤΑΓΛΩΤΤΙΣΗ & ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ

```
model.compile('adam', 'mean_squared_error')

# Εκπαίδευση του μοντέλου

model.fit([train.user_id, train.poi_id],

                    train.rating, epochs=15,callbacks=[tensorboard_callback])
```



Στη συνέχεια παρουσιάζεται ένα στιγμιότυπο της διαδικασίας εκπαίδευσης του μοντέλου. Δύο τιμές που εμφανίζονται κατά τη διάρκεια εκπαίδευσης του μοντέλου και σχετίζονται με την απόδοσή του είναι το loss και το accuracy. Όπως φαίνεται από την εικόνα, όσο εκπαιδεύεται το μοντέλο τόσο μικραίνει και το loss. Επίσης, το επιθυμητό είναι να βελτιώνεται η ακρίβεια του μοντέλου με βάση τα δεδομένα εκπαίδευσης. Στο παρακάτω στιγμιότυπο όμως δεν παρουσιάζεται η τιμή accuracy. Αυτό συμβαίνει γιατί η μεταβλητή «ακρίβειας» πρέπει να χρησιμοποιείται σε μοντέλα ταξινόμησης καθώς λειτουργεί μόνο με ακριβείς αντιστοιχίες (y_true == y_pred). Σε μοντέλα αντίστοιχα με αυτό που παρουσιάζεται εδώ, όπου οι προβλέψεις είναι ένας

οποιοσδήποτε πραγματικός αριθμός από το 1 έως το 5 , η πιθανότητα να προβλεφθεί η ακριβής τιμή είναι ιδιαίτερα σπάνια και δεν σχετίζεται καθόλου με την απόδοση του μοντέλου.

```
serveruser@0de53738c2fb:~/aipolis$ python saveaspbsuccess.py
Train on 981736 samples
Epoch 1/15
981736/981736 [==============================] - 51s 52us/sample - loss: 0.9757
Epoch 2/15
981736/981736 [==============================] - 51s 51us/sample - loss: 0.7274
Epoch 3/15
981736/981736 [==============================] - 50s 51us/sample - loss: 0.6795
Epoch 4/15
981736/981736 [==============================] - 50s 51us/sample - loss: 0.6438
Epoch 5/15
981736/981736 [==============================] - 50s 51us/sample - loss: 0.6108
Epoch 6/15
981736/981736 [==============================] - 50s 51us/sample - loss: 0.5755
Epoch 7/15
981736/981736 [==============================] - 48s 49us/sample - loss: 0.5378
Epoch 8/15
981736/981736 [==============================] - 61s 62us/sample - loss: 0.5032
Epoch 9/15
981736/981736 [==============================] - 57s 58us/sample - loss: 0.4739
Epoch 10/15
981736/981736 [==============================] - 59s 60us/sample - loss: 0.4501
Epoch 11/15
981736/981736 [==============================] - 58s 59us/sample - loss: 0.4308
Epoch 12/15
981736/981736 [==============================] - 50s 51us/sample - loss: 0.4151
Epoch 13/15
736512/981736 [=====================>........] - ETA: 13s - loss: 0.3936
```

Η τιμή loss όσο βελτιώνεται (δηλαδή μικραίνει), βελτιστοποιείται το μοντέλο. Αυτή τιμή είναι ιδιαίτερα χρήσιμη όταν εκπαιδεύονται πολλά παρόμια μοντέλα και το loss παραμένει το ίδιο. Αυτό μπορεί να μεταφραστεί ότι το μοντέλο σταμάτησε να σημειώνει πρόοδο, οπότε δεν χρειάζεται περισσότερο training.

3.2 Αξιοποίηση ενός machine learning μοντέλου

Μετά την εκπαίδευση και τον πειραματισμό με ένα machine learning μοντέλο ένα εξίσου σημαντικό στάδιο είναι η μετάβασή του σε μια μορφή αξιοποιήσιμη από μια εφαρμογή. Η διαδικασία που είναι γνωστή στα αγγλικά ως "serving machine learning models" είναι ουσιαστικά η λήψη ενός εκπαιδευμένου μοντέλου και η διάθεσή του για την εξυπηρέτηση αιτημάτων πρόβλεψης. Ορισμένοι τρόποι που μπορεί να επιτευχθεί αυτό παρουσιάζονται στη συνέχεια.

1. **TensorFlow Serving**

Ένας από τους πιο δημοφιλείς και εύκολους τρόπους αξιοποίησης μοντέλων μηχανικής μάθησης μπορεί να επιτευχθεί με τη χρήση του TensorFlow Serving και του Docker. Το Tensorflow Serving αποτελεί ένα API που έχει σχεδιαστεί από την Google για την χρήση συστημάτων μηχανικής εκμάθησης στην «παραγωγή» (production).,

Το TensorFlow Serving χρησιμοποιεί τη μορφή SavedModel για τα machine learning μοντέλα. Το εν λόγω format επιτρέπει τόσο την αξιοποίηση όσο και την τροποποίηση των μοντέλων από τις εφαρμογές που τα «χρησιμοποιούν». Ένα ιδιαίτερα χρήσιμο χαρακτηριστικό τους tensorflow serving είναι ότι παρέχει τη δυνατότητα αποθήκευσης και της έκδοσης του μοντέλου. Ένας σύνηθες προσδιορισμός της εκάστοτε έκδοσης του μοντέλου είναι χρησιμοποιώντας ένα timsestamp. Έτσι το σύστημα που αξιοποιεί το εκάστοτε μοντέλο μπορεί να προσδιορίσει ποια έκδοσή του θέλει να «φορτώσει». Αν δεν το κάνει θα έχει πρόσβαση στην τελευταία έκδοση (by default).

2. **Importing a Keras model into TensorFlow.js**
   Επιτρέπει την αξιοποίηση των εκπαιδευμένων μοντέλων σε περιβάλλοντα που υποστηρίζουν JavaScript, όπως a web browsers ή server side μέσω Node.js. Πρέπει να σημειωθεί εδώ ότι το TensorFlow.js υποστηρίζει επίσης τον καθορισμό μοντέλων σε JavaScript και την εκπαίδευσή τους απευθείας στον Web browser χρησιμοποιώντας ένα API παρόμοιο με το Keras.
3. **Publishing Machine Learning API με Python FlaskKeras H5 format**

3.3 Docker

Το Docker είναι ένα open source εργαλείο που που επιτρέπει στους χρήστες να δημιουργούν, να διαχειρίζονται, να αναπτύσσουν και να αναπαράγουν εφαρμογές χρησιμοποιώντας "containers". Τα "containers" μπορούν να θεωρηθούν ως ένα περιορισμένο περιβάλλον που φιλοξενεί οτιδήποτε απαιτείται για την εκτέλεση μιας εφαρμογής σε επίπεδο λειτουργικού συστήματος. Αυτό σημαίνει ότι κάθε εφαρμογή που αναπτύσσεται χρησιμοποιώντας το Docker «ζει» σε ένα δικό της περιβάλλον και οι απαιτήσεις της αντιμετωπίζονται ξεχωριστά.

Το εργαλείο αυτό είναι ιδιαίτερα χρήσιμο καθώς δίνει τη δυνατότητα εκτέλεσης εφαρμογές σε ένα απομονωμένο εικονικό περιβάλλον, με τα χαρακτηριστικά που απαιτούνται, χωρίς να επιβαρύνεται ο εκάστοτε υπολογιστής. Επιπλέον, διευκολύνει τη μεταφορά εφαρμογών από έναν χρήστη σε έναν άλλο χωρίς να απαιτούνται περαιτέρω εγκαταστάσεις προγραμμάτων, βιβλιοθηκών κλπ. Επίσης, αποτελεί ένα ιδιαίτερα ελαφρύ εργαλείο το οποίο δεν προκαταλαμβάνει μνήμη στο σύστημα, καθώς τα «containers» καταλαμβάνουν μόνο τους απαραίτητους πόρους του συστήματος, δυναμικά.

3.4 Tensorboard

4 Βιβλιογραφία

Chollet, F. (2020). *Deep Learning with Python, Second Edition.* Manning Publications.

Google. (2020). *TensorFlow*. Ανάκτηση από Tf Agents: https://www.tensorflow.org/agents

Jeffrey Dean, G. S. (2012). Large scale distributed deep networks. *25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12)*. *Curran Associates Inc., Red Hook*, (σσ. 1223–1231). USA.

Keras. (χ.χ.). *Keras Functional Api*. Ανάκτηση από https://keras.io/guides/functional_api/

Packtpub. (χ.χ.). Ανάκτηση από Data Flow Graphs: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781786468574/1/ch01lvl1sec13/data-flow-graphs

Team, T. (2019, Ιανουαρίου 14). *What's coming in TensorFlow 2.0*. Ανάκτηση από https://medium.com/tensorflow/whats-coming-in-tensorflow-2-0-d3663832e9b8

Tensorflow. (χ.χ.). *Σύνοψη Keras*. Ανάκτηση από https://www.tensorflow.org/guide/keras/overview?hl=el

Βάρσος, Α. (2018, Οκτώβριος). Αυτόματη ταξινόμηση κειμένων φυσικής. Πάτρα,: ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ, ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ.

Λιβιέρης, Ι. Ε. (2012). Μη Γραμμικές Μέθοδοι Συζυγών Κλίσεων για Βελτιστοποίηση και. Πανεπιστήμιο Πατρών.

Παράρτημα: Κώδικας μοντέλου συστάσεων

```
from tensorflow.keras.models import Model

from tensorflow.python.saved_model import builder as saved_model_builder

from tensorflow.keras.layers import Input, Embedding, Flatten, Dense, Concatenate, Dropout

from sklearn.model_selection import train_test_split

import tensorflow as tf

import pandas as pd

import os

import calendar;

import time;

import tempfile

from tensorflow.python.keras.callbacks import TensorBoard

from time import time

import datetime

import tensorboard

import numpy as np

import matplotlib.pyplot as plt


os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

dataset = pd.read_csv('data/ratings.csv')

train, test = train_test_split(dataset, test_size=0.00002)


n_users = len(dataset.user_id.unique())

n_poi = len(dataset.poi_id.unique())
```

```
log_dir = "logsave/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)


features_num = 10

# Users embedding features.

user_input = Input(shape=[1], name="User-Input")

user_embedding = Embedding(n_users + 1, features_num)(user_input)

user_vector = Flatten()(user_embedding)


# Pois embedding features.

poi_input = Input(shape=[1],name="Poi-Input")

poi_embedding = Embedding(n_poi + 1, features_num)(poi_input)

poi_vector = Flatten()(poi_embedding)


# Concatenate features.

concatenated = Concatenate()([poi_vector, user_vector])


# Create model

layer1 = Dense(128, activation='relu')(concatenated)

dropout1 = Dropout(0.1)(layer1)

layer2 = Dense(64, activation='relu')(dropout1)

dropout2 = Dropout(0.1)(layer2)

output = Dense(1)(dropout2)

model = Model([user_input, poi_input], output)

model.compile('adam', 'mean_squared_error')
```

```
# Train.
model.fit([train.user_id, train.poi_id],
                    train.rating, epochs=15,callbacks=[tensorboard_callback])

# 5 top predictions for user_id =1
poi_data = np.array(list(set(dataset.poi_id)))

user = np.array([1 for i in range(len(poi_data))])
predictions = model.predict([user, poi_data])
predictions = np.array([a[0] for a in predictions])
recommended_poi_ids = (-predictions).argsort()[:5]
print(recommended_poi_ids)
print(predictions[recommended_poi_ids])

# Evaluate.
predictions = model.predict([test.user_id, test.poi_id])
[print(predictions[i], test.rating.iloc[i]) for i in range(0, len(test))]
model.save('saved_model_aipolis.h5')

MODEL_DIR = tempfile.gettempdir()
version = 1
export_path = os.path.join(MODEL_DIR, str(version))
print('export_path = {}\n'.format(export_path))

tf.keras.models.save_model(
```

```
    model,
    export_path,
    overwrite=True,
    include_optimizer=True,
    save_format=None,
    signatures=None,
    options=None
)


print('\nSaved model:')
ts = calendar.timegm(time.gmtime())
tf.saved_model.save(model, "./model_aipolis/" + str(ts))
```