Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Αυτοματοποιημένη Διαχείριση Πόρων με Χρήση Τεχνικών Πρόβλεψης Χρονοσειρών και Βαθιά Ενισχυτική Μάθηση

Μελέτη και υλοποίηση

Διπλωματικη Εργασια

της

ΕΛΕΝΗΣ-ΙΩΑΝΝΑΣ ΚΟΥΛΕΤΟΥ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2021

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Αυτοματοποιημένη Διαχείριση Πόρων με Χρήση Τεχνικών Πρόβλεψης Χρονοσειρών και Βαθιά Ενισχυτική Μάθηση

Μελέτη και υλοποίηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΕΛΕΝΗΣ-ΙΩΑΝΝΑΣ ΚΟΥΛΕΤΟΥ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Μαρτίου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Νεκτάριος Κοζύρης	Γεώργιος Γκούμας	Ιωάννης Κωνσταντίνου
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Π.Θ.

Αθήνα, Μάρτιος 2021

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Ελένη-Ιωάννα Κουλέτου

11η Μαρτίου 2021

Περίληψη

Τα τελευταία χρόνια το cloud computing είναι ένας από τους πιο επιδραστικούς κλάδους της επιστήμης των υπολογιστών. Οι cloud υπηρεσίες γίνονται ολοένα και πιο δημοφιλείς και ο φόρτος εργασίας των παρόχων συνεχώς και αυξάνεται. Γι΄ αυτό τον λόγο, η διαχείριση των πόρων τείνει να γίνεται επιτακτική ανάγκη. Η αποτελεσματική διαχείριση πόρων έτσι ώστε να καλύπτονται οι ανάγκες τόσο των παρόχων όσο και των πελατών αποτελεί κυρίαρχο ζήτημα στον ερευνητικό τομέα. Έχουν αναπτυχθεί διαφορετικές τεχνικές για να λυθεί αυτό το ζήτημα. Στην παρούσα διπλωματική, ασχοληθήκαμε με τον συνδυασμό δύο τεχνικών για την δυναμική αυτοματοποίηση των πόρων. Η πρώτη αφορά την πρόβλεψη χρονοσειρών φόρτου εργασίας. Σε αυτόν τον τομέα, υλοποιήσαμε έξι διαφορετικά μοντέλα για την πρόβλεψη χρονοσειρών (ARIMA, Prophet, LSTM, GRU, CNN, Autoencoders), τα οποία συγκρίναμε με τρεις μετρικές σφαλμάτων (μέσο τετραγωνικό σφάλμα, ρίζα του μέσου τετραγωνικού σφάλματος και μέσο απόλυτο σφάλμα). Με μικρή διαφορά, καλύτερο αποδείχτηκε το LSTM. Η δεύτερη αφορά την αυτοματοποιημένη διαχείριση πόρων με χρήση ενός πράκτορα βαθιάς ενισχυτικής μάθησης. Υλοποιήσαμε και πειραματιστήκαμε με δύο διαφορετικές μορφές χρονοσειρών (ένα απλό ημίτονο και μια πιο περίπλοκη χρονοσειρά) και με δύο διαφορετικής μορφής συναρτήσεις επιβράβευσης. Σε κάθε περίπτωση, ο πράκτορας μας λειτουργούσε αρκετά ικανοποιητικά. Στη συνέχεια, συνδυάσαμε τις παραπάνω τεχνικές. Σχεδιάσαμε, λοιπόν, ένα νέο σύστημα το οποίο βασίζεται στον πράκτορα της βαθιάς ενισχυτικής μάθησης αλλά ταυτόχρονα λαμβάνει μια έξτρα πληροφορία για την μελλοντική κατάσταση του περιβάλλοντος μέσω του μοντέλου πρόβλεψης, ώστε να αποφασίσει ποια δράση θα πραγματοποιήσει. Συγκρίναμε το νέο μας μοντέλο με τον απλό πράκτορα βαθιάς ενισχυτικής μάθησης και συμπεράναμε ότι αντιλαμβάνεται πιο γρήγορα τις αλλαγές του φόρτου εργασίας και δρα πιο άμεσα. Αυτό είναι κάτι που περιμέναμε δεδομένου ότι, πλέον, ο πράκτορας μας για να πάρει μια απόφαση λαμβάνει υπόψη του τόσο την τωρινή κατάσταση όσο και μια πρόβλεψη για την μελλοντική κατάσταση του συστήματος. Τέλος, εξετάσαμε και αναφέρουμε μελλοντικές επεκτάσεις του συστήματος μας, ώστε να γίνει ακόμα πιο αποδοτικό.

Λέξεις Κλειδιά

Ελαστικότητα, Υπολογιστικό Νέφος, Διαχείριση Πόρων, Πρόβλεψη χρονοσειρών, ARIMA, Prophet, Μηχανική Μάθηση, Αναδρομικά Νευρωνικά Δίκτυα, LSTM, GRU, Συνελικτικά Νευρωνικά Δίκτυα, Αυτοκωδικοποιητές, Βαθιά ενισχυτική μάθηση

Abstract

In recent years, cloud computing has been one of the most influential areas of computer science. Cloud services are becoming more and more popular and the workload of providers is constantly increasing. For this reason, resource management tends to become an imperative. Effective resource management to meet the needs of both providers and customers is a major issue in the research sector. Different techniques have been developed to resolve this issue. In this thesis, we dealt with the combination of two techniques for dynamic resource automation. The first concerns the workload time-series prediction. In this area, we implemented six different time-series prediction models (ARIMA, Prophet, LSTM, GRU, CNN, Autoencoders), which we compared with three error metrics (mean squared error, root mean squared error and mean absolute error). With a little difference, LSTM proved to be the best. The second concerns the automated resource management using a deep reinforcement learning agent. We implemented and experimented with two different time-series types (a simple sine and a more complex time-series) and with two different forms of reward functions. In any case, our agent worked quite satisfactorily. Then, we combined the above techniques. So, we designed a new system that relies on the deep reinforcement learning agent, but, at the same time, receives extra information about the future state of the environment through the prediction model, to decide what action to take. We compared our new system with the simple deep reinforcement learning agent and concluded that it perceives workload changes faster and acts more directly. This is something we have been waiting for since, now, our agent to make a decision takes into account both the current state and a forecast for the future state of the system. Finally, we examined and report future extensions of our system to make it even more efficient.

Keywords

Elasticity, Cloud Computing, Resource Management, Time-Series Prediction, ARIMA, Prophet, Machine Learning, Recurrent Neural Networks, LSTM, GRU, Convolutional Neural Networks, AutoEncoders, Deep Reinforcement Learning

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Κοζύρη Νεκτάριο για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο Εργαστήριο Υπολογιστικών Συστημάτων. Επίσης, ευχαριστώ ιδιαίτερα τον υποψήφιο Δρ. Χαλβαντζή Νικόλαο για την πολύ καλή καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου για την εμψύχωση και ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια καθώς και τους φίλους μου, οι οποίοι ήταν κοντά μου σε κάθε δυσκολία.

Αθήνα, Μάρτιος 2021

Ελένη-Ιωάννα Κουλέτου

Περιεχόμενα

Περίληψη **1**

Abstract **3**

Ευχαριστίες **7**

1 Εισαγωγή **15**
1.1 Αντικείμενο της διπλωματικής 16
1.2 Συνεισφορά (Σχεδιασμός - Υλοποίηση - Πειραματική αξιολόγηση) 16
1.3 Οργάνωση του τόμου 17

2 Θεωρητικό υπόβαθρο **19**
2.1 Cloud Computing 19
2.2 Τεχνικές προβλέψεων χρονοσειρών 21
2.2.1 Εισαγωγή στις χρονοσειρές 21
2.2.2 Box-Jenkins Μοντέλα -ARIMA 23
2.2.3 Facebook Prophet 24
2.3 Μηχανική μάθηση 27
2.3.1 Είδη Μηχανικής Μάθησης 28
2.3.2 Τεχνητά Νευρωνικά Δίκτυα 28
2.3.3 Βαθιά Μάθηση (Deep Learning) 30
2.3.4 Συναρτήση Κόστους (Loss Function) 30
2.3.5 Αλγόριθμος Κατάβασης Κλίσης (Gradient descent) 31
2.3.6 Αλγόριθμος Οπισθοδιάδοσης Σφάλματος (Backpropagation) 33
2.3.7 Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks) 33
2.3.8 Long Short-Term Memory (LSTM) 35
2.3.9 Gated Recurrent Units (GRU) 36
2.3.10 Multilayer Perceptron 37
2.3.11 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) . . . 38
2.3.12 Αυτοκωδικοποιητής (Autoencoder) 39
2.3.13 Ενισχυτική Μάθηση (Reinforcement Learning) 41
2.3.14 Μαρκοβιανή Διαδικασία Αποφάσεων (Markov Decision Process) 42
2.3.15 Μάθηση Χρονικών Διαφορών (Temporal Difference Learning) 44
2.3.16 ε-greedy Policy 44
2.3.17 Q-Learning 45
2.3.18 Βαθιά Ενισχυτική Μάθηση 45

3 Σχετικές εργασίες **49**

4 Περιγραφή, Σχεδίαση και Υλοποίηση **53**
4.1 Αρχιτεκτονική συστήματος 53
4.1.1 Υποσύστημα Πρόβλεψης Χρονοσειράς 53
4.1.2 Υποσύστημα Δυναμικής Διαχείρισης Πόρων 54
4.2 Σχεδίαση Προτεινόμενου Συστήματος 54
4.3 Υλοποίηση Συστήματος 55

5 Πειραματική Αξιολόγηση **57**
5.1 Πλαίσιο Πειραματικής Αξιολόγησης 57
5.2 Μορφή χρονοσειράς φόρτου εργασίας 58
5.3 Αποτελέσματα Πρόβλεψης χρονοσειρών 59
5.3.1 ARIMA 60
5.3.2 Prophet 61
5.3.3 LSTM 63
5.3.4 GRU 64
5.3.5 CNN 65
5.3.6 Autoencoder-LSTM 66
5.3.7 Σύγκριση μεθόδων 67
5.4 Αποτελέσματα Δυναμικής Διαχείρισης Πόρων με χρήση DQN 69
5.5 Αποτελέσματα Προτεινόμενου Συστήματος 73

6 Επίλογος **79**
6.1 Συμπεράσματα 79
6.2 Μελλοντικές Επεκτάσεις 80

Βιβλιογραφία **86**

Κατάλογος Σχημάτων

4.1 Αρχιτεκτονική προτεινόμενου συστήματος 55

Κατάλογος Εικόνων

2.1 Τρόπος λειτουργίας του Prophet 25
2.2 Παράδειγμα ανάλυσης χρονοσειράς με το Facebook Prophet 27
2.3 Νευρώνας 29
2.4 Τεχνητός νευρώνας 29
2.5 Ορισμός του Κρυφού Επιπέδου 30
2.6 Gradient descent calculation 32
2.7 Gradient descent on non-convex example 32
2.8 Backpropagation 34
2.9 Αρχιτεκτονική των αναδρομικών νευρωνικών δικτύων 34
2.10 LSTM unit 35
2.11 LSTM vs GRU 37
2.12 Παράδειγμα συνέλιξης 38
2.13 Single autoencoder 40
2.14 Agent-Environment 42
2.15 Παράδειγμα MDP 43
2.16 Q-Learning vs Deep Q-Learning 46

5.1 Τελικό διάγραμμα χρονοσειράς φόρτου εργασίας 59
5.2 Εστιασμένο διάγραμμα χρονοσειράς σε περίοδο 1000 ωρών 59
5.3 Fit ARIMA (1,1,0) one time Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη 60
5.4 ARIMA (1,1,0) rolling forecasting Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη 61
5.5 Fit Prophet one time Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη 62
5.6 Prophet rolling forecasting Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη 62
5.7 Test LSTM Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη, Κίτρινο: τα πρώτα 24 σημεία 64
5.8 Test GRU Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη, Κίτρινο: τα πρώτα 24 σημεία 65
5.9 Test CNN Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη, Κίτρινο: τα πρώτα 24 σημεία 66
5.10 Test Autoencoder-LSTM Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη, Κίτρινο: τα πρώτα 24 σημεία 67
5.11 Σύγκριση μεθόδων με mse, rmse, mae 67

5.12 Πείραμα για το RL component με load: σταθερό ημίτονο 70
5.13 Πείραμα για το RL component με load: synthetic dataset 71
5.14 Πείραμα για το RL component με load: σταθερό ημίτονο και γκαουσιανή συνάρτηση επιβράβευσης 72
5.15 Πείραμα για το RL component με load: synthetic dataset και γκαουσιανή συνάρτηση επιβράβευσης 72
5.16 Πείραμα για το RL component + predictor με load: synthetic dataset 74
5.17 Πείραμα για το DERP με load: synthetic dataset 75
5.18 Συνδυασμός δύο μεθόδων. Μπλε: DERP και Πράσινο: Προτεινόμενο σύστημα 76
5.19 Εστίαση στα 500-600 test steps. Μπλε: DERP και Πράσινο: Προτεινόμενο σύστημα 77
5.20 Εστίαση στα 600-700 test steps. Μπλε: DERP και Πράσινο: Προτεινόμενο σύστημα 77

Κεφάλαιο 1

Εισαγωγή

Τα υπολογιστικά νέφη χρησιμοποιούνται όλο και περισσότερο την τελευταία δεκαετία. Μεγάλες επιχειρήσεις όπως η Google, Amazon, IBM και η Microsoft προσφέρουν Cloud υπηρεσίες και οι πελάτες τους αυξάνονται συνεχώς. Τι είναι, όμως, το cloud περιβάλλον; Όλοι μας έχουμε χρησιμοποιήσει της υπηρεσίες της Google όπως το Google Docs για να γράψουμε ένα έγγραφο ή το Google Drive για να αποθηκεύσουμε τα αρχεία μας. Αυτές οι υπηρεσίες δεν λειτουργούν με υπολογιστικούς πόρους του δικού μας μηχανήματος, αλλά μέσω του διαδικτύου δεσμεύουν απομακρυσμένους υπολογιστικούς πόρους τους οποίους μας διαθέτουν για να υλοποιήσουμε την εργασία που θέλουμε. Πολλές επιχειρήσεις επιλέγουν την χρησιμοποίση cloud υπηρεσιών, για να υλοποιήσουν τις εργασίες τους, για πολυάριθμους λόγους. Ο πρώτος λόγος είναι ότι δεν χρειάζεται να επενδύσουν σε αγορά υπερσύγχρονου και ακριβού εξοπλισμού για την εκτέλεση απαιτητικών εργασιών. Λόγω του τεράστιου μεγέθους των δεδομένων και την χρονική και χωρική πολυπλοκότητα των σύγχρονων συστημάτων, πολλές εργασίες χρειάζονται υπερσύγχρονους επεξεργαστές, επιταχυντές και μεγάλη χωρική ικανότητα για να καταφέρουν να διαχειριστούν την δουλειά τους. Κάτι τέτοιο μπορεί να οδηγήσει τις επιχειρήσεις να χρειάζεται να σπαταλήσουν μεγάλο μέρος του τζίρου τους για υπολογιστικά συστήματα, κάτι που μπορούν να αποφύγουν ¨ενοικιάζοντας¨ εικονικά μηχανήματα από Cloud υπηρεσίες. Ένας δεύτερος λόγος είναι ότι εκτός από τα έξοδα που απαιτούνται για την απόκτηση εξοπλισμού, απαλάσσονται και από το κόστος συντήρησης και λειτουργίας μεγάλων υπολογιστικών συστημάτων, το οποίο αναλαμβάνουν οι εταιρείες ή οργανισμοί που διαθέτουν τους υπολογιστικούς πόρους προς μίσθωση.

Ας σκεφτούμε, όμως, και την πλευρά των επιχειρήσεων που προσφέρουν αυτούς τους πόρους. Ένα μεγάλο πρόβλημα που αντιμετωπίζουν είναι η διαχείριση των υπολογιστικών πόρων αποδοτικά, ώστε αφενώς να μην δυσαρεστήσουν τους πελάτες τους, αφετέρου να έχουν όσο το δυνατόν περισσότερα έσοδα. Μια απλή σκέψη είναι να γίνεται μια στατική κατανομή πόρων, έτσι ώστε να υπάρχει ένας συγκεκριμένος αριθμός από μηχανήματα, με συγκεκριμένα χαρακτηριστικά μνήμης, αποθηκευτικού χώρου κτλ. Αυτό αντιμετωπίζει αρκετές δυσκολίες κυρίως γιατί το περιβάλλον cloud έχει διαφορετικό εισερχόμενο φόρτο εργασίας ανάλογα με το έτος, τον μήνα, την ημέρα της εβδομάδας και την ώρα. Λόγω διαφορετικών μοτίβων πρόσβασης, η στατική κατανομή αντιμετωπίζει πρόβλημα ταλάντωσης κατά τη χρήση πόρων. Κατά συνέπεια, αρκετές cloud υπηρεσίες παρέχουν άλλους τρόπους για να διαχειρίζονται τους πόρους δυναμικά. Αυτή η δυναμική διαχείριση πόρων ορίζει μια νέα έννοια στα cloud περιβάλλοντα, την έννοια της ελαστικότητας. Οι πόροι μπορούν να αποκτηθούν ή να

απελευθερωθούν σύμφωνα με την απαίτηση χρήσης της εφαρμογής. Οι πάροχοι υπηρεσιών cloud όπως το Google App Engine και το AWS EC2 παρέχουν τη δυνατότητα ελαστικότητας, η οποία θα μπορούσε να αυξήσει τη χρήση πόρων και να βελτιώσει την ποιότητα των υπηρεσιών (QoS) για τους χρήστες. Χάρη στην ελαστική ιδιότητα των υπολογιστικών νεφών, η ποσότητα των δεσμευμένων πόρων μπορεί να αυξομειώνεται δυναμικά. Η αυτοματοποίηση αυτής της διαδικασίας είναι μια πρόκληση που ερευνάται ιδιαίτερα τα τελευταία χρόνια.

1.1 Αντικείμενο της διπλωματικής

Αντικείμενο της διπλωματικής είναι η αυτοματοποιημένη διαχείριση πόρων σε ελαστικά περιβάλλοντα υπολογιστικών νεφών. Το πρόβλημα της διαχείρισης πόρων σε περιβάλλοντα υπολογιστικών νεφών απασχολεί την ερευνητική κοινότητα εδώ και αρκετά χρόνια, λόγω της πολυπλοκότητάς του. Είναι εξαιρετικά δύσκολο να αυτοματοποιηθεί η διαδικασία κλιμάκωσης έτσι ώστε να αποδοθεί η ακριβής ποσότητα πόρων για να παρέχεται η απαιτούμενη ποιότητα υπηρεσιών (QoS). Ο πάροχος πρέπει να διαχειρίζεται τους πόρους του με τέτοιον τρόπο ώστε να αποφεύγει την υπερβολική αλλά και την ανεπαρκή παροχή (over-provisioning, under-provisioning).

Υπάρχουν δύο είδη αυτόματης κλιμάκωσης: η αντιδραστική (reactive) κατανομή πόρων, η οποία ξεκινά όταν προκύπτει η ανάγκη για ανακατανομή, και η προληπτική (provision) κατανομή πόρων, η οποία προγραμματίζεται πριν από την ανάγκη. Απαιτείται μια ισχυρή προσέγγιση για την εξισορρόπηση της αντιστάθμισης για τη συμφωνία επιπέδου εξυπηρέτησης πελατών και τη σχέση κόστους - αποτελεσματικότητας. Η δυσκολία που αντιμετωπίζουν οι reactive τεχνικές είναι ότι δεν μπορούν να διαχειριστούν μεγάλες αλλαγές στο φόρτο εργασίας άμεσα. Αυτό έχει ως αποτέλεσμα, οι πελάτες να μένουν στην αναμονή μέχρι να ολοκληρωθεί η κλιμάκωση και τελικά να έχουν αρνητική εικόνα για τον πάροχο. Αντίθετα, οι προληπτικοί μηχανισμοί μπορούν να ξεκινήσουν να κάνουν ανακατανομή πόρων πριν χρειαστεί έτσι ώστε οι πόροι να είναι άμεσα διαθέσιμοι την ώρα που έχουν προβλέψει αύξηση του φόρτου εργασίας. Μην ξεχνάμε όμως ότι το μέλλον κάποιες φορές είναι απρόβλεπτο και μπορεί να μην ακολουθήσει το μοτίβο που έχει υπολογίσει το μοντέλο πρόβλεψης.

1.2 Συνεισφορά (Σχεδιασμός - Υλοποίηση - Πειραματική αξιολόγηση)

Οι πιο δημοφιλείς τρόποι για αυτόματη κλιμάκωση είναι: η κλιμάκωση με βάση ένα συγκεκριμένο όριο (threshold-based), μοντέλα ουράς (queuing models), μεθόδους πρόβλεψης χρονοσειρών (time-series forecasting) και ενισχυτική μάθηση (reinforcement learning). Στην παρούσα διπλωματική, συνδυάζουμε την τεχνική της πρόβλεψης χρονοσειρών με την βαθιά ενισχυτική μάθηση. Πιο συγκεκριμένα:

- Εξετάσαμε το πρόβλημα πρόβλεψης χρονοσειρών φόρτου εργασίας με περιοδική συμπεριφορά.
- Υλοποιήσαμε και αξιολογήσαμε διάφορα state-of-art μοντέλα πρόβλεψης χρονοσειρών.

- Υλοποιήσαμε και αξιολογήσαμε ένα σύστημα βασισμένο σε βαθιά ενισχυτική μάθηση για την αυτοματοποιημένη διαχείριση πόρων.
- Σχεδιάσαμε και υλοποιήσαμε ένα νέο σύστημα βασισμένο σε βαθιά ενισχυτική μάθηση το οποίο χρησιμοποιεί τις παραπάνω μεθόδους πρόβλεψης.
- Αποδείξαμε πειραματικά ότι έχει καλύτερη συμπεριφορά από υπάρχουσες λύσεις.

1.3 Οργάνωση του τόμου

Η εργασία αυτή είναι οργανωμένη σε έξι κεφάλαια: Στο Κεφάλαιο 2 δίνεται το θεωρητικό υπόβαθρο. Αρχικά περιγράφονται βασικές έννοιες των υπολογιστικών νεφών, στην συνέχεια, μαθηματικοποιημένες τεχνικές πρόβλεψης χρονοσειρών και τέλος οι βασικές έννοιες της μηχανικής μάθησης με ιδιαίτερη βάση στα νευρωνικά δίκτυα που χρησιμοποιούνται στην πρόβλεψη χρονοσειρών και την ενισχυτική μάθηση. Στο Κεφάλαιο 3 περιγράφονται οι σχετικές με το θέμα εργασίες. Στο Κεφάλαιο 4 παρουσιάζεται η ανάλυση, η σχεδίαση και η υλοποιήση του συστήματος, δηλαδή η περιγραφή των υποσυστημάτων και των εφαρμογών του. Η περιγραφή της υλοποίησης του συστήματος και τα προγραμματιστικά εργαλεία. Στο Κεφάλαιο 5 παρουσιάζεται ο έλεγχος καλής λειτουργίας του συστήματος αρχικά του υποσυστήματος πρόβλεψης χρονοσειρών, στην συνέχεια του υποσυστήματος βαθιάς ενισχυτικής μάθησης και τέλος το υβριδικό σύστημα. Τέλος στο Κεφάλαιο 6 δίνεται η συνολική αξιολόγηση του υβριδικού συστήματος, καθώς και μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζονται αναλυτικά οι βασικές τεχνολογίες που έχουν σχέση με την εργασία αυτή. Πιο συγκρεκριμένα, αναλύονται οι βασικές εννοιες για τα περιβάλλοντα υπολογιστικών νεφών, οι τεχνικές πρόβλεψης χρονοσειρών, τόσο με στατιστικά μοντέλα, όσο και με χρήση μηχανικής μάθησης και οι βασικές έννοεις της ενισχυτικής και βαθιάς ενισχυτικής μάθησης.

2.1 Cloud Computing

Είτε λέγεται cloud computing είτε on-demand computing, software as a service ή Internet as a platform, το κοινό στοιχείο είναι μια μετατόπιση στη γεωγραφία του υπολογισμού. Όταν δημιουργείται ένα υπολογιστικό φύλλο με την υπηρεσία Google Docs, τα κύρια στοιχεία του λογισμικού βρίσκονται σε αόρατους υπολογιστές, όπου είναι άγνωστοι, πιθανώς διασκορπισμένοι σε όλες τις ηπείρους[1].
Το cloud computing χωρίζεται σε τρεις μεγάλες κατηγορίες υπηρεσιών:

- **Infrastructure as a service (IaaS)** είναι μια υπηρεσία υπολογιστικού νέφους στην οποία ένας προμηθευτής παρέχει στους χρήστες πρόσβαση σε υπολογιστικούς πόρους όπως διακομιστές, αποθήκευση και δικτύωση. Οι οργανισμοί χρησιμοποιούν τις δικές τους πλατφόρμες και εφαρμογές μέσα στην υποδομή ενός παρόχου υπηρεσιών.

- **Platform as a service (PaaS)** είναι μια υπηρεσία υπολογιστικού νέφους που παρέχει στους χρήστες ένα περιβάλλον cloud στο οποίο μπορούν να αναπτύξουν, να διαχειριστούν και να παραδώσουν εφαρμογές. Εκτός από την αποθήκευση και άλλους πόρους υπολογιστών, οι χρήστες μπορούν να χρησιμοποιήσουν μια σειρά από προεγκατεστημένα εργαλεία για να αναπτύξουν, να προσαρμόσουν και να δοκιμάσουν τις δικές τους εφαρμογές.

- **Software as a service (SaaS)** είναι μια υπηρεσία υπολογιστικού νέφους που παρέχει στους χρήστες πρόσβαση σε λογισμικό που βασίζεται σε cloud ενός προμηθευτή. Οι χρήστες δεν εγκαθιστούν εφαρμογές στις τοπικές τους συσκευές. Αντ άυτού, οι εφαρμογές βρίσκονται σε ένα απομακρυσμένο δίκτυο cloud στο οποίο έχετε πρόσβαση μέσω του διαδικτύου ή ενός API. Μέσω της εφαρμογής, οι χρήστες μπορούν να αποθηκεύουν και να αναλύουν δεδομένα και να συνεργάζονται σε έργα.

Cloud Elasticity

Η ελαστικότητα (elasticity) είναι ο βαθμός στον οποίο ένα σύστημα είναι σε θέση να προσαρμοστεί στις αλλαγές του φόρτου εργασίας (workload) με την παροχή και την απόσυρση πόρων με αυτόνομο τρόπο, έτσι ώστε σε κάθε χρονική στιγμή οι διαθέσιμοι πόροι να ταιριάζουν με την τρέχουσα ζήτηση όσο το δυνατόν πιο στενά[2].

Οφέλη της ελαστικότητας

Ο στόχος της παροχής πόρων είναι ο εντοπισμός και η παροχή των κατάλληλων πόρων στους κατάλληλους φόρτους εργασίας εγκαίρως, έτσι ώστε οι εφαρμογές να μπορούν να χρησιμοποιούν αποτελεσματικά τους πόρους. Με άλλα λόγια, το ποσό των πόρων θα πρέπει να είναι ελάχιστο για ένα φόρτο εργασίας να διατηρεί ένα επιθυμητό επίπεδο ποιότητας υπηρεσίας ή να μεγιστοποιεί την απόδοση (ή να ελαχιστοποιεί το χρόνο ολοκλήρωσης του φόρτου εργασίας) ενός φόρτου εργασίας. Για καλύτερη παροχή πόρων, απαιτείται καλύτερη χαρτογράφηση πόρων εργασίας. Ο στόχος της παροχής πόρων είναι να ανιχνεύσει τον πιο προσαρμοστικό και ικανό φόρτο εργασίας που υποστηρίζει τον προγραμματισμό πολλαπλών φόρτων εργασίας, ώστε να είναι αρκετά ικανός να ικανοποιεί διαφορετικές απαιτήσεις QoS όπως CPU utilization, διαθεσιμότητα, αξιοπιστία, ασφάλεια κ.λπ. Επομένως, η παροχή πόρων λαμβάνει υπόψη τον χρόνο εκτέλεσης κάθε ξεχωριστού φόρτου εργασίας, αλλά το πιο σημαντικό, η συνολική απόδοση βασίζεται επίσης στον τύπο φόρτου εργασίας, δηλαδή ετερογενείς (διαφορετικές απαιτήσεις QoS) και ομοιογενείς (παρόμοιες απαιτήσεις QoS) [3].

Η ελαστικότητα[4] στοχεύει στο να ταιριάξει το ποσό των πόρων που διατίθενται σε μια υπηρεσία με το ποσό των πόρων που πραγματικά χρειάζεται, αποφεύγοντας την υπερβολική παροχή ή την υποεκτίμηση.

- Η υπερβολική παροχή (**over-provisioning**), δηλαδή η κατανομή περισσότερων πόρων από ό,τι απαιτείται, θα πρέπει να αποφεύγεται, καθώς ο πάροχος υπηρεσιών πρέπει συχνά να πληρώνει για τους πόρους που διατίθενται στην υπηρεσία. Για παράδειγμα, μια εξαιρετικά μεγάλη εικονική μηχανή του Amazon Elastic Compute Cloud M4 κοστίζει 0,239 $ / ώρα. Εάν μια υπηρεσία έχει εκχωρήσει δύο εικονικές μηχανές όταν απαιτείται μόνο μία, ο πάροχος υπηρεσιών σπαταλά 2.095 $ κάθε χρόνο. Επομένως, τα έξοδα του παρόχου υπηρεσιών είναι υψηλότερα από το βέλτιστο και το κέρδος τους μειώνεται.

- Η ανεπαρκής παροχή (**under-provisioning**), δηλαδή η κατανομή λιγότερων πόρων από ό, τι απαιτείται, πρέπει να αποφευχθεί, διαφορετικά η υπηρεσία δεν μπορεί να εξυπηρετήσει τους χρήστες της με μια καλή υπηρεσία. Στο παραπάνω παράδειγμα, η ανεπαρκής παροχή του ιστότοπου μπορεί να τον κάνει να φαίνεται αργό ή απρόσιτο. Οι χρήστες του Διαδικτύου σταματούν τελικά να έχουν πρόσβαση σε αυτό, έτσι, ο πάροχος υπηρεσιών χάνει πελάτες. Μακροπρόθεσμα, το εισόδημα του παρόχου θα μειωθεί, γεγονός που μειώνει επίσης το κέρδος τους.

Auto-scaling

Οι μηχανισμοί Auto-scaling μπορούν να χωριστούν σε δύο κατηγορίες, τους proactive και τους reactive[5].

1. Οι reactive μηχανισμοί παρακολουθούν συνεχώς το σύστημα και ενεργοποιούν μια συγκεκριμένη δράση κλιμάκωσης όταν πληρούται μια συγκεκριμένη κατάσταση (π.χ. παροχή ή αφαίρεση ενός δεδομένου αριθμού πόρων όταν μια συγκεκριμένη μέτρηση είναι υψηλότερη ή χαμηλότερη από ένα συγκεκριμένο όριο). Το κύριο πρόβλημα με τους reactive μηχανισμούς είναι ότι ο χρόνος αντίδρασης (ο χρόνος που έχει παρέλθει από την ανίχνευση της κατάστασης σκανδάλης έως ότου οι πόροι είναι έτοιμοι για χρήση) μπορεί να είναι ανεπαρκής για να αποφευχθεί η υπερφόρτωση του συστήματος. Επιπλέον, αυτοί οι μηχανισμοί μπορούν να προκαλέσουν αστάθεια του συστήματος λόγω της συνεχούς διακύμανσης των κατανεμημένων πόρων.

2. Οι proactive (ή predicted) μηχανισμοί προσπαθούν να προβλέψουν την ποσότητα των πόρων που απαιτούνται κατά την επόμενη χρονική περίοδο, με βάση στατιστικά ή μαθηματικά μοντέλα παρατηρούμενων φόρτων εργασίας και μετρήσεων συστήματος είτε τεχνικές ενισχυτικής μάθησης είτε μεθόδους βασισμένες σε ουρές (Queuing Network).

2.2 Τεχνικές προβλέψεων χρονοσειρών

2.2.1 Εισαγωγή στις χρονοσειρές

Η ανάλυση και η πρόβλεψη χρονοσειρών είναι ένα αντικείμενο μελέτης που αποκτά όλο και μεγαλύτερη δυναμική και συγκεντρώνει την αυξανόμενη προσοχή της επιστημονικής κοινότητας σε πολλούς και διαφορετικούς τομείς, όπως την επεξεργασία σήματος, την πρόγνωση καιρού, την σεισμική πρόβλεψη, το ηλεκτροεγκεφαλογράφημα αλλά και την πρόβλεψη των μελλοντικών τιμών ζήτησης των υπολογιστικών πόρων.

Χρονοσειρά (Time-series)

Με τον όρο χρονοσειρά ή χρονολογική σειρά ορίζουμε μια ακολουθία που λαμβάνεται σε διαδοχικά ισαπέχουσες χρονικές στιγμές $x_t : t = 0, 1, 2, ..,$ όπου κάθε x_t εκφράζει κατά την χρονική στιγμή t, την κατάσταση ενός συστήματος το οποίο εξελίσσεται στο χρόνο. Τα διαστήματα μεταξύ δύο χρονικών στιγμών μπορεί να είναι ωριαία, ημερήσια, μηνιαία, ετήσια ή οποιασδήποτε άλλης χρονικής συχνότητας.

Οι χρονοσειρές μπορούν να αποσυντεθούν σε τέσσερα στοιχεία, καθένα από τα οποία εκφράζει μια συγκεκριμένη πτυχή της κίνησης των τιμών των χρονοσειρών. Τα τέσσερα αυτά ποιοτικά χαρακτηριστικά είναι τα εξής[6]:

1. **Τάση** (Trend), η οποία περιγράφει την κίνηση κατά μήκος του όρου. Αποτελεί μια μακροπρόθεσμη αύξηση ή μείωση του επιπέδου των τιμών.

2. **Εποχιακές παραλλαγές** (Seasonal variations), οι οποίες αντιπροσωπεύουν εποχιακές αλλαγές.

3. **Κυκλικές διακυμάνσεις** (Cyclical fluctuations), οι οποίες αντιστοιχούν σε περιοδικές αλλά όχι εποχιακές διακυμάνσεις, δηλαδή οι τιμές αυξομειώνονται, αλλά όχι σε σταθερές περιόδους.

4. **Ακανόνιστες παραλλαγές** (Irregular variations), οι οποίες είναι άλλες τυχαίες πηγές παραλλαγών της σειράς.

Δεν είναι απαραίτητο μία χρονοσειρά να αποτελείται και από τα τέσσερα παραπάνω χαρακτηριστικά. Για παράδειγμα, υπάρχουν χρονοσειρές που είναι απόλυτα τυχαίες και αυτές τις ονομάζουμε λευκό θόρυβο (white noise) και είναι στάσιμες.

Στασιμότητα (Stationarity)

Μια χρονοσειρά y_t θεωρείται στάσιμη (stationary), όταν για κάθε s, της κατανομής $(y_t, ..., y_{t+s})$ δεν επηρεάζεται από το χρόνο t. Επομένως, μια χρονοσειρά που περιέχει τάση, εποχιακές παραλλαγές ή/και κυκλικές διακυμάνσεις δεν είναι στάσιμη, ενώ μια χρονοσειρά που περιέχει μόνο θόρυβο θεωρείται στάσιμη, αφού δεν σχετίζεται με τον χρόνο.

Αυτοσυσχέτιση (Autocorrelation)

Όπως η συσχέτιση μετρά την γραμμική σχέση μεταξύ δύο μεταβλητών, η αυτοσυσχέτιση μετρά τη γραμμική σχέση μεταξύ των καθυστερημένων τιμών (lagged values) μιας χρονοσειράς. Το διάγραμμα αυτοσυσχέτισης χρησιμοποιείται για να προσδιορίσει πόσο τυχαίο είναι ένα σύνολο δεδομένων. Στην περίπτωση τυχαίων δεδομένων, οι τιμές αυτοσυσχέτισης πλησιάζουν το μηδέν για όλες τις τιμές με καθυστέρηση στο χρόνο, διαφορετικά, μία ή περισσότερες τιμές αυτοσυσχέτισης πλησιάζουν το 1 ή το 1. Στην πλοκή αυτοσυσχέτισης, ο οριζόντιος άξονας αντιπροσωπεύει τις χρονικές καθυστερήσεις. Οι τιμές στον κατακόρυφο άξονα υπολογίζονται χρησιμοποιώντας τον συντελεστή αυτοσυσχέτισης [7]

$$R_h = \frac{C_h}{C_0}$$

όπου C_h είναι η συνάρτηση αυτοδιακύμανσης (auto-covariance function) η οποία ορίζεται ως:

$$C_h = \frac{1}{N}\sum_{t=1}^{N}(X_t - \bar{X})(X_{t+\tau} - \bar{X}),$$

όπου N είναι ο αριθμός των δειγμάτων, $\bar{X}$ είναι ο μέσος των δειγμάτων $X_t, t = 1...N$ και τ η χρονική καθυστέρηση (time lag).

$$C_0 = \frac{1}{N}\sum_{t=1}^{N}(X_t - \bar{X})^2$$

.

Μερική αυτοσυσχέτιση (Partial autocorrelation)

Η μερική αυτοσυσχέτιση [7] στο τ είναι η αυτοσυσχέτιση μεταξύ X_t και $X_{t-\tau}$ που υπολογίζεται μόνο από την καθυστέρηση των παραπάνω τ-1.

Επιπλέον, είναι σημαντικό να αναφερθεί οτι υπάρχουν πολλά διαφορετικά κίνητρα για τα οποία αναλύουμε τις χρονοσειρές. Στο πλαίσιο της στατιστικής, της οικονομετρίας, της ποσοτικής χρηματοδότησης, της σεισμολογίας, της μετεωρολογίας και της γεωφυσικής, ο

πρωταρχικός στόχος της ανάλυσης χρονοσειρών είναι η πρόβλεψη. Στο πλαίσιο της επεξεργασίας σήματος, της μηχανικής ελέγχου και της μηχανικής επικοινωνίας χρησιμοποιείται για την ανίχνευση και εκτίμηση σημάτων, ενώ στο πλαίσιο της εξόρυξης δεδομένων, της αναγνώρισης προτύπων και της μηχανικής μάθησης η ανάλυση των χρονοσειρών μπορεί να χρησιμοποιηθεί για την ομαδοποίηση (clustering), για την ταξινόμηση, για την ανίχνευση ανωμαλιών (anomaly detection) καθώς και την πρόβλεψη (prediction)) πρόγνωση (forecasting) μελλοντικών καταστάσεων.

Η πρόβλεψη χρονοσειρών αποτελεί ένα μέρος μελέτης της στατιστικής και των πιθανοτήτων και έχει αναπτυχθεί ένας μεγάλος αριθμός από στατιστικά μοντέλα. Επιπρόσθετα, αλγόριθμοι και μοντέλα μηχανικής μάθησης έχει αποδειχτεί ότι μπορούν να χρησιμοποιηθουν για την προβλεψη χρονοσειρών παρόλο που ο αρχικός σκοπός δημιουργίας τους μπορεί να μην ήταν αυτός. Στην συνέχεια, παρατίθονται και αναλύονται τα μοντέλα που χρησιμοποιήθηκαν και στο πειραματικό μέρος της Διπλωματικής εργασίας.

2.2.2 Box-Jenkins Μοντέλα -ARIMA

Ο Box-Jenkins πρότεινε μια ομάδα στατιστικών μοντέλων που είναι ευρέως γνωστά για την πρόβλεψη χρονοσειρών. Το πιο δημοφιλές μοντέλο είναι βασισμένο στην αυτοπαλινδρόμηση (autoregression) AR, στην ολοκλήρωση (integration) I και στον κινητό μέσο όρο (moving average) MA[8]. Δημιουργήθηκε και χρησιμοποείται για πρόβλεψη μίας χρονοσειράς (univariate time-series forecasting), αλλά υπάρχουν και επεκτάσεις για την πρόβλεψη πολλαπλών χρονοσειρών (multivariate time-series forecasting). Τα βασικά μοντέλα που δημιουργήθηκαν από τον Box-Jenkins είναι τα εξής:

- **AutoRegressive (AR(p)) Model**: Σε αυτό το μοντέλο, η έξοδος του y_t είναι γραμμικά εξαρτώμενη από τις προηγούμενες p τιμές της χρονοσειράς ($y_{t-1}, .., y_{t-p}$) και κάποιον λευκό θόρυβο e_t. Επομένως, το autoregressive μοντέλο p-τάξης AR(p) υπολογίζει την επόμενη κατάσταση/έξοδο ως[9]: $y_t = a_1 y_{t-1} + a_2 y_{t-2} + ... + a_p y_{t-p} + e_t$, όπου e_t: λευκός θόρυβος και $(a_1, a_2, ..., a_p)$: οι παράμετροι του παραδείγματος που εκπαιδεύονται ανάλογα με την χρονοσειρά. Η τάξη p που περιγράφει καλύτερα μία χρονοσειρά μπορεί να βρεθεί μέσω του διαγράμματος μερικής αυτοσυσχέτισης. Πιο συγκεκριμένα, προτίνεται η τιμή το διάγραμμα μερικής αυτοσυσχέτισης πέφτει κάτω από το σημαντικό επίπεδο σε τ = p + 1 [7]. Επίσης, εάν το διάγραμμα αυτοσυσχέτισης παρουσιάζει τιμές κοντά στο μηδέν καταλαβαίνουμε ότι η συνάρτηση δεν παρουσιάζει περιοδικότητα και είναι μόνο λευκός θόρυβος e_t.

- **Moving Average (MA(q)) Model**: Θεωρώντας μια τυχαία διαδικασία e_t με μηδενικό μέσο όρο και διακύμανση σ^2, η έξοδος y_t του μοντέλου κινητού μέσου q-τάξης μπορεί να γραφεί ως[9]: $y_t = e_t + b_1 e_{t-1} + b_2 e_{t-2} + ... + b_q e_{t-q}$, όπου $(b_1, b_2, ..., b_q)$: οι παράμετροι του παραδείγματος που εκπαιδεύονται ανάλογα με την χρονοσειρά. Από το διάγραμμα μερικής αυτοσυσχέτισης συμπεραίνουμε ότι ο αριθμός καθυστερήσεων πριν οι τιμές αυτοσυσχέτισης μειωθούν κάτω από το σημαντικό επίπεδο ορίζουν την τιμή του χ για το κινούμενο μέσο όρο[7].

- **AutoRegressive Moving Average (ARMA(p,q)) Model**: Το μοντέλο ARMA εκτελεί τις

δύο παραπάνω στατιστικές διαδικασίες. Συμπερασματικά, η έξοδος y_t του μοντέλου (p,q)-τάξης περιγράφεται με την εξής εξίσωση: $y_t = a_1 y_{t-1} + a_2 y_{t-2} + ... + a_p y_{t-p} + e_t + b_1 e_{t-1} + b_2 e_{t-2} + ... + b_q e_{t-q}$, όπου e_t: λευκός θόρυβος και $(a_1, a_2, ..., a_p)$, $(b_1, b_2, ..., b_q)$: οι παράμετροι των μοντέλων AR, MA αντίστοιχα.

- **AutoRegressive Integrated Moving Average (ARIMA(p,d,q)) Model**: Σαν εξέλιξη των παραπάνω μοντέλων που απαιτούν οι χρονοσειρές να είναι στάσιμες, δημιουργήθηκε το μοντέλο ARIMA. Η διαφορά με τα προηγούμενα είναι ο όρος I που δηλώνει την διαφόριση της χρονοσειράς d φορές μέχρι να γίνει στάσιμη. Η έξοδος y_t του μοντέλου (p,d,q)-τάξης περιγράφεται με την εξής εξίσωση: $y_t^{(d)} = a_1 y_{t-1}^{(d)} + a_2 y_{t-2}^{(d)} + ... + a_p y_{t-p}^{(d)} + e_t + b_1 e_{t-1} + b_2 e_{t-2} + ... + b_q e_{t-q}$

- **Seasonal AutoRegressive Integrated Moving Average (SARIMA(p,d,q)(P,D,Q)S) Model**: Για να λαμβάνεται υπόψη και η εποχικότητα (seasonality) δημιουργήθηκε αυτή η επέκταση. Οι επιπλέον παράμετροι του μοντέλου ορίζουν την εποχικότητα. Πιο συγκεκριμένα, P: το πλήθος των εποχικών autoregressive όρων, D: το πλήθος των εποχιακών διαφορίσεων και Q: το πλήθος των εποχικών όρων κινητού μέσου και S: ο αριθμός των χρονικών βημάτων στη χρονική περίοδο μιας εποχής.

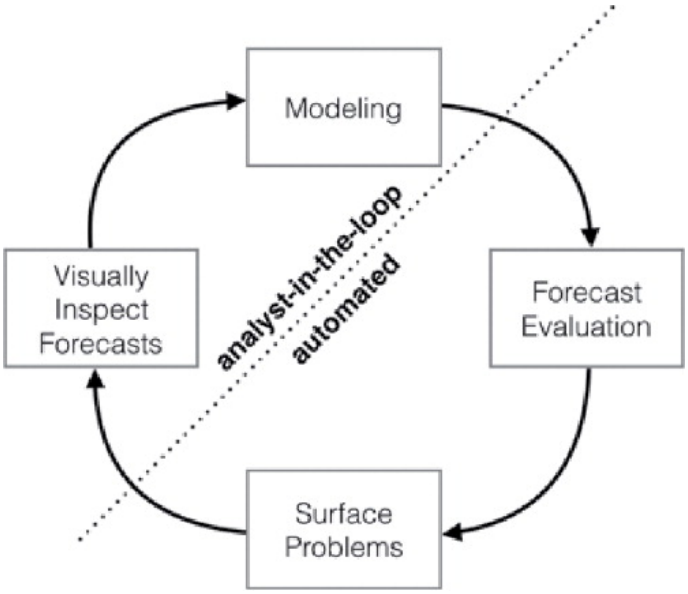
Επιπρόσθετα, υπάρχουν τα αντίστοιχα μοντέλα με κατάληξη -X (exogenous) (όπως το ARIMAX, ARMAX) στα οποία λαμβάνονται υπόψη τα εξωτερικά χαρακτηριστικά της χρονοσειράς καθώς και τα μοντέλα με πρόθεμα V- (Vector) (όπως το VARIMA, VARIMAX) που χρησιμοποιούνται για (multivariate time-series forecasting).

Το δύσκολο μέρος αυτών των μοντέλων είναι να βρεθούν οι κατάλληλες τιμές για τις παραμέτρους τους ώστε το μοντέλο να προσαρμόζεται, όσο καλύτερα γίνεται, στην κάθε χρονοσειρά. Εξ' αιτίας αυτού του προβλήματος έχουν δημιουργηθεί μοντέλα που κάνουν αναζήτηση των πιο αποδοτικών υπερπαραμέτρων (hyperparameter tuning), όπως το μοντέλο auto-arima[10].

2.2.3 Facebook Prophet

Το μοντέλο αυτό σχεδιάστηκε από την Facebook[11] στις αρχές του 2017 και είναι διαθέσιμο open-source σε python[12] και σε R[13]. Η μέθοδος που προτείνανε αποσυνθέτει την χρονοσειρά σε τρία βασικά χαρακτηριστικά την τάση (trend), εποχικότητα (seasonality) και διακοπές (holidays). Επομένως, η χρονοσειρά μπορεί να περιγραφτεί ως εξής: $y(t) = g(t) + s(t) + h(t) + e_t$, όπου η $g(t)$ αναπαριστά το trend που είναι μη-περιοδικές αλλαγές της χρονοσειράς, η $s(t)$ αναπαριστά τις περιοδικές αλλαγές (ημερήσια, εβδομαδιαία ή/και ετήσια εποχικότητα), η $h(t)$ αναπαριστά την επίδραση των διακοπών οι οποίες επιδρούν στην χρονοσειρά με ακανόνιστο τρόπο και e_t αναπαριστά τυχόν ιδιοσυγκρασιακές αλλαγές που δεν καλύπτονται από το μοντέλο. Όσον αφορά τον επιπλέον όρο των διακοπών που δεν υπήρχε στο μοντέλο ARIMA, δινεται η δυνατότητα να οριστούν οι περίοδοι που μία χρονοσειρά μπορεί να αναπτύξει ανωμαλίες (outliers) για μία μέρα ή για μικρό χρονικό διάστημα, οι οποίες είναι εύκολο να προβλεφθούν. Για παράδειγμα, η περίοδος χριστουγεννιάτικων διακοπών σε ένα περιβάλλον υπολογιστικών νεφών που χρησιμοποιούν οι φοιτητές ενός τμήματος ίσως επιδράσει αρνητικά στο workload γιατί οι φοιτητές θα είναι σε μια κατάσταση ξεκούρασης,

ενώ η περίοδος πριν ή μετά την εξεταστική που οι φοιτητές παραδίδουν εργασίες (και αυτή η περίοδος είναι τις ίδιες μέρες κάθε χρόνο) θα έχουν προβλεπόμενα υψηλότερο από το συνηθισμένο workload. Το κύριο χαρακτηριστικό του μοντέλου είναι ότι είναι αυτοματοποιημένο και έχουν οριστεί κάποιοι ¨αναλυτές¨ που εκπαιδεύονται και αναλύουν τα αποτελέσματα και διαμορφώνουν κατάλληλα τις τελικές προβλέψεις του μοντέλου.



Εικόνα 2.1: *Τρόπος λειτουργίας του Prophet*

Το μοντέλο έχει παρόμοια χαρακτηριστικά με το Generalized additive model (GAM), μια κατηγορία μοντέλων παλινδρόμησης (regression models) με δυνητικά μη γραμμικά smoothers που εφαρμόζονται στους regressors. Εδώ χρησιμοποιείται μόνο ο χρόνος ως regressor αλλά πιθανώς πολλές γραμμικές και μη γραμμικές συναρτήσεις του χρόνου ως components. Η μοντελοποίηση εποχικότητας ως πρόσθετο συστατικό είναι η ίδια προσέγγιση με την εκθετική εξομάλυνση. Η πολλαπλή εποχικότητα, μπορεί να επιτευχθεί μέσω λογαριθμικού μετασχηματισμού (log transformation). Το μοντέλο για να υπολογίσει την τάση χρησιμοποιεί σημεία ελέγχου checkpoints S που επιλέγονται αυτόματα μέσω του αναλυτή του. Πιο συγκεκριμένα, τα χαρακτηριστικά του υπολογίζονται με τους παρακάτω τρόπους:

1. Η τάση υπολογίζεται με δύο τρόπους, ένα κορεσμένο μοντέλο ανάπτυξης (saturating growth model), και ένα μερικά γραμμικο μοντέλο (piecewise linear model).

   (α΄) Το saturating growth model μέσω σημείων ελέγχου checkpoints S $s_j = 1, ..., S$, υπολογίζει την τάση μέ την συνάρτηση:

   $$g(t) = \frac{C(t)}{1 + exp(-(k + a(t)^T\delta)(t - (m + a(t)^T\gamma))}, \text{οπου}$$

   - C(t)): η δεδομένη χωρητικότητα (carrying capacity),
   - δ: ένα διάνυσμα προσαρμογών τιμών (a vector of rate adjustments) $\delta \in R^S$
   - a: $a_j(t) = \begin{matrix} 1 & t \geqslant s_j, \\ 0 & otherwise. \end{matrix}$,
   - k + $a(t)^T\delta$: ο ρυθμός ανάπτυξης ((growth rate) σε κάθε checkpoint,

- m: μια παράμετρος μετατόπισης offset parameter, πρέπει επίσης να προσαρμοστεί για να συνδέσει τα τελικά σημεία των τμημάτων,
- $\gamma_j = (s_j - m - \sum_{i<j} \gamma_i)(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l \leq j} \delta_l})$, για την σωστή προσαρμογή των checkpoints.

(β') Το piecewise linear model μέσω σημείων ελέγχου checkpoints S $s_j = 1, ..., S$, υπολογίζει την τάση μέ την συνάρτηση:

$$g(t) = (k + a(t)^T \delta)t + (m + a(t)^T \gamma), \text{ όπου}$$

k, m, δ όπως ορίστηκαν και στο παραπάνω μοντέλο, όμως το $\gamma_j = s_j \delta_j$, για να γίνει η συνάρτηση συνεχής.

2. Η εποχικότητα ορίζεται μέσω των σειρών Fourier:

$$s(t) = \sum_{n=1}^{N} (a_n cos(\frac{2\pi nt}{P}) + b_n sin(\frac{2\pi nt}{P}))$$

Ανάλογα πόσα seasonality υπάρχουν στην χρονοσειρά τόσες σειρές Fourier προστίθονται στο $s(t)$.

3. Οι διακοπές και τα events που εμπεριέχονται ορίζονται σε έναν πινακα $Z(t)$ και αντιστοιχίζονται με χώρες προέλευσης. Ο χρήστης είτε μπορεί να ορίσει μία χώρα και να συμπεριλάβει στο μοντέλο του τις διακοπές που βρίσκονται στον πίνακα $Z(t)$, είτε να ορίσει έναν δικό του πίνακα που ορίζει τις ημερομηνίες που θέλει. Ο τύπος των διακοπών, λοιπόν, είναι:

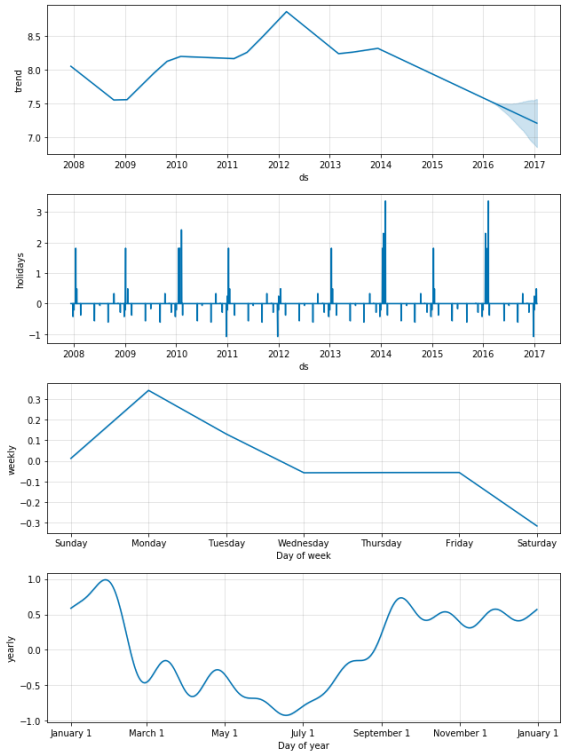
$$h(t) = Z(t)\kappa, \text{ όπου}$$

κ: ακολουθεί κανονική κατανομή με μηδενική μέση τιμή και διασπορά υ^2

Είναι, επίσης, σημαντικό να αναφερθεί ότι παρουσιάζονται πλεονεκτήματα έναντι του μοντέλου ARIMA, τα οποία είναι τα εξής[11]:

- Ευελιξία: Το μοντέλο προσαρμόζεται με ευκολία στη εποχικότητα σε πολλές διαφορετικές περιόδους. Επίσης, μπορούμε να αφήσουμε τον αναλυτή να κάνει διαφορετικές παραδοχές σχετικά με τις τάσεις.
- Σε αντίθεση με τα μοντέλα ARIMA, οι μετρήσεις δεν χρειάζεται να είναι συνεχόμενες καθώς μπορεί να διαχειριστεί και τα διαστήματα που οι τιμές λείπουν (missing values).
- Η εκπαίδευση είναι πολύ γρήγορη.
- Το μοντέλο πρόβλεψης έχει εύκολα ερμηνεύσιμες παραμέτρους που μπορούν να αλλάξουν από τον αναλυτή για να επιβάλουν υποθέσεις στην πρόβλεψη. Επιπλέον, οι αναλυτές συνήθως έχουν εμπειρία με την παλινδρόμηση και είναι εύκολα σε θέση να επεκτείνουν το μοντέλο ώστε να συμπεριλάβει νέα στοιχεία.

Οι αναλυτές παίζουν σπουδαίο ρόλο στην πρόβλεψη χρονοσειρών διότι βοηθούν το μοντέλο να γίνει πιο αυτοματοποιημένο. Συγκρίνουν τα αποτελέσματα του prophet με τα baseline μοντέλα και αν υπάρχουν μεγάλες διαφορές προσαρμόζουν την τάση ή την εποχικότητα αν



Εικόνα 2.2: *Παράδειγμα ανάλυσης χρονοσειράς με το Facebook Prophet*

χρειάζεται. Επιπλέον, αν υπάρχουνε μεγάλα σφάλματα αφαιρούν τους outliers καθώς και κατανοούν αν υπάρχει μεγάλη διαφορά στην μορφή των δεδομένων και φτιάχνουν νέα checkpoints και κατευθύνουν το μοντέλο να συμπεριφερθεί διαφορετικά απο πριν χωρίζοντας το πρόβλημα σε δύο μέρη. Υπάρχουν περιπτώσες που δεν μπορούν να διορθωθούν εύκολα, αλλά τα περισσότερα από τα ζητήματα που έχει εξεταστεί το μοντέλο μπορούν να διορθωθούν καθορίζοντας checkpoints και αφαιρώντας τους outliers. Ένα επιπλέον χαρακτηριστικό του μοντέλου είναι ότι μπορείς να προσθέσεις και άλλες χρονοσειρές οι οποίες έχουν αλληλεξάρτηση με την χρονοσειρά που μελετάς, ως extra regressors.

2.3 Μηχανική μάθηση

Η μηχανική μάθηση είναι ένας κλάδος της επιστήμης υπολογιστών που είναι υποσύνολο της τεχνητής νοημοσύνης. Το 1959, ο Άρθουρ Σάμουελ ορίζει τη μηχανική μάθηση ως ¨Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί¨. Στην συνέχεια, ο Tom M. Mitchell πρότεινε έναν πιο επίσημο ορισμό που χρησιμοποιείται ευρέως: ¨ Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία Ε ως προς μια κλάση εργασιών Τ και ένα μέτρο επίδοσης Π, αν η επίδοσή του σε εργασίες της κλάσης Τ, όπως αποτιμάται από το μέτρο Ρ, βελτιώνεται με την εμπειρία Ε¨.

Πέρα από τα μοντέλα που βασίζονται στην στατιστική ανάλυση και αποσύνθεση χρονοσειρών, μελέτες έχουν δείξει ότι η μηχανική μάθηση μπορεί να ανταπεξέλθει ισάξια και κάποιες φορές καλύτερα στην πρόβλεψη μελλοντικών καταστάσεων. Το βασικότερο πλεονέκτημα που έχουν τα νευρωνικά δίκτυα είναι ότι μπορούν να εκπαιδευτούν εύκολα σε

μη-γραμμικά μοντέλα. Επιπλέον, μέσω της μηχανικής μάθησης έχει αποδειχθεί οτι μπορεί να γίνει αυτοματοποιημένη διαχείριση πόρων σε υπολογιστικά περιβάλλοντα.

2.3.1 Είδη Μηχανικής Μάθησης

- **Επιβλεπόμενη Μάθηση** (Supervised Learning): Ο αλγόριθμος εκπαιδεύεται μέσω ενός συνόλου από παραδείγματα όπου γνωρίζουμε για κάθε είσοδο την επιθυμητή έξοδο κατασκευάζοντας μία συνάρτηση που τα συσχετίζει. Στόχος είναι η γενίκευση της συνάρτησης αυτής για εισόδους με άγνωστη έξοδο. Τα προβλήματα που μελετούνται με αυτό το είδος είναι προβλήματα ταξινόμησης (Classification) και παλινδρόμησης (Regression).

- **Μη επιβλεπόμενη Μάθηση** (Unsupervised Learning): Ο αλγόριθμος εκπαιδεύεται με unlabeled δεδομένα με σκοπό να βρει κοινά χαρακτηριστικά μεταξύ τους. Τα κύρια προβλήματα που μελετούνται με μη επιβλεπόμενη μάθηση είναι η συσταδοποίηση (Clustering) και η μείωση διαστατικότητας (Dimensionality Reduction).

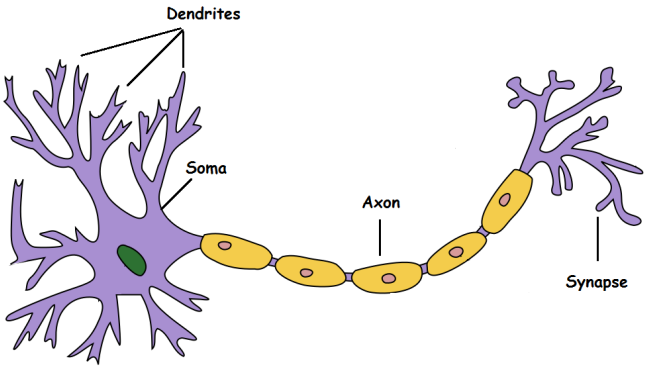
- **Ενισχυτική Μάθηση** (Reinforcement Learning): Η λειτουργία της διαφέρει από τις προηγούμενες και θεωρείται ότι είναι αυτή που προσεγγίζει περισσότερο τον ανθρώπινο εγκέφαλο. Η μάθηση βασίζεται στην αλληλεπίδραση του υποκειμένου με το περιβάλλον του αναπτύσσοντας εμπειρία των κινήσεων του. Αναπτύσσει στρατηγικές ώστε να καταφέρει να επιτύχει τους στόχους του μέσω εμπειρίας από τα λάθη του έχοντας, ως στόχο την μεγιστοποίηση του βραβείου του (reward). Χρησιμοποιείται κυρίως σε προβλήματα σχεδιασμού όπως ο έλεγχος κίνησης ρομπότ καθώς και παιχνίδια στρατηγικής.

2.3.2 Τεχνητά Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα δημιουργήθηκαν για να προσομοιάσουν την λειτουργία των ανθρώπινων εγκεφάλων. Αναμφίβολα, ο εγκέφαλος είναι το πιο μυστήριο όργανο του σώματός μας και αυτό που έχουμε δυσκολευτεί περισσότερο να αποκρυπτογραφήσουμε. Το βασικό μέρος της λειτουργίας του εξαρτάται από τους νευρώνες. Κάθε νευρώνας συνδέεται με χιλιάδες άλλους νευρώνες και ανταλλάσει μηνύματα με αυτούς μέσω διαφόρων ισχύων συνάψεων. Όσο πιο πολλά μαθαίνουμε οι συνάψεις των συγκεκριμένων νευρώνων που συμμετείχαν στην εκμάθηση ενισχύονται. Ένας νευρώνας έχει δύο καταστάσεις. Μία κατά την οποία στέλνει έναν παλμό, και μία που δεν στέλνει σήμα. Κάθε νευρώνας έχει αρκετές συνάψεις διαφορετικής ισχύος (βάρη) ως εισόδους και μία έξοδο που εξαρτάται από τις εισόδους. Ένας νευρώνας είναι μια πολύ μικρή υπολογιστική μονάδα, τόσο σε μέγεθος όσο και σε ικανότητα υπολογισμών.

Το μοντέλο ενός τεχνητού νευρώνα (Perceptron) που αποτελεί την βάση για την σχεδίαση μιας μεγάλης οικογένειας νευρωνικών δικτύων δομείται με τρία βασικά στοιχεία[14]:

1. Ένα σύνολο συνάψεων (ή διασυνδέσεων), κάθε μία εκ των οποίων χαρακτηρίζεται από το δικό του βάρος ή δύναμη που συμβολίζουμε συνήθως με το γράμμα w. Ανόμοια με το βάρος μίας σύναψης στον ανθρώπινο εγκέφαλο, το βάρος ενός τεχνητού νευρώνα μπορεί να λαμβάνει τόσο αρνητικές όσο και θετικές τιμές.

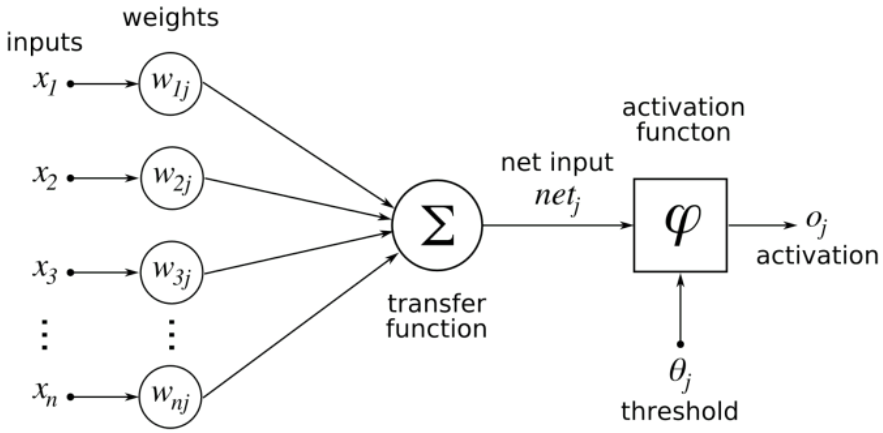


Εικόνα 2.3: *Νευρώνας*

2. Έναν αθροιστή για την άθροιση των σημάτων εισόδου, σταθμισμένων από τα αντίστοιχα συναπτικά βάρη του νευρώνα.

3. Μια συνάρτηση ενεργοποίησης φ (activation function) για τον περιορισμό του πλάτους του σήυματος εξόδου ενός νευρώνα.

Επιπρόσθετα, κάποια νευρωνικά δίκτυα περιέχουν και μια εξωτερικά εφαρμοζόμενη πόλωση b_k η οποία προκαλεί θετική ή αρνητική προκατάληψη (bias) στο αποτέλεσμα της συνάρτησης ενεργοποίησης.



Εικόνα 2.4: *Τεχνητός νευρώνας*

Ένας απλός τεχνητός νευρώνας με μαθηματικούς όρους μπορεί να περιγραφεί από το ακόλουθο ζεύγος εξισώσεων[14].

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

,

$$y_k = \varphi(u_k + b_k)$$

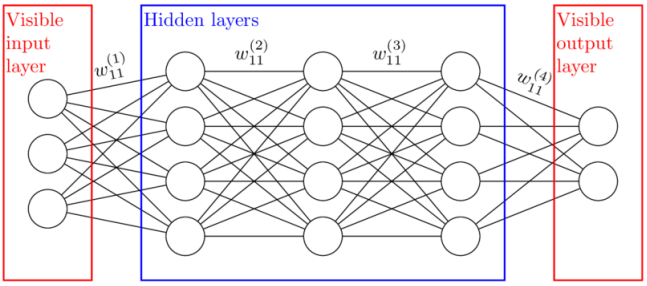
Οι συναρτήσεις ενεργοποίησης ορίζουν την έξοδο ενός νευρώνα. Κάποιες από τις πιο συναρτήσεις που χρησιμοποιούνται πιο συχνά είναι οι ακόλουθες:

- Sigmoid Function: $g(x) = \frac{1}{1+e^{-x}}$

- Hyperbolic Tangent Function (tanh): $g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Rectified Linear Function (ReLU): $g(x) = \begin{matrix} 0 & x < 0, \\ x & x \geq 0. \end{matrix}$
- Softmax Function: $g(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$, με x ένα διάνυσμα με μέγεθος K ίσο με τον αριθμό εισόδων.

2.3.3 Βαθιά Μάθηση (Deep Learning)

Η βαθιά μάθηση (Deep Learning) είναι ένα υποπεδίο της μηχανικής μάθησης που επιχειρεί να μάθει υψηλού επιπέδου άντληση δεδομένων χρησιμοποιώντας ένα πλήθος από διαφορετικές αρχιτεκτονικές. Πρόκειται για μια αναδυόμενη προσέγγιση και έχει εφαρμοστεί ευρέως σε παραδοσιακούς τομείς τεχνητής νοημοσύνης. Υπάρχουν κυρίως τρεις σημαντικοί λόγοι για την άνθηση της βαθιάς μάθησης σήμερα: οι δραματικά αυξημένες δυνατότητες επεξεργασίας τσιπ (π.χ. μονάδες GPU), το σημαντικά μειωμένο κόστος του υλικού υπολογιστών και οι σημαντικές εξελίξεις στους αλγόριθμους μηχανικής μάθησης[15]. Ένα σήμα από τη στιγμή της εισόδου σε ένα βαθιά νευρωνικό δίκτυο δέχεται πολλούς μετασχηματισμούς μέχρι την έξοδο. Ο αριθμός των μετασχηματισμών, ουσιαστικά, καθορίζεται από τον αριθμό των κρυφών επιπέδων (hidden layers),όπως φαίνεται και στην Εικόνα 2.5, με τα οποία είναι δομημένος ένας Deep Learning αλγόριθμος.



Εικόνα 2.5: *Ορισμός του Κρυφού Επιπέδου*

2.3.4 Συναρτήση Κόστους (Loss Function)

Ο ρόλος της συνάρτησης κόστους/μετρικής σφάλματος είναι να συγκρίνει τις τιμές εξόδου ενός τεχνητού νευρωνικού δικτύου με τις αναμενόμενες (πραγματικές) τιμές που έχουν οριστεί κατά την εκπαίδευση. Η επιλογή της συνάρτησης κόστους είναι σημαντική γιατί καθορίζει και πόσο αποδοτική είναι η εκπαίδευση του δικτύου. Οι πιο διαδεδομένες συναρτήσεις κόστους είναι οι εξής:

- Mean Squared Error: Χρησιμοποείται κυρίως στα προβλήματα παλινδρόμησης. Υπολογίζει το μέσο τετραγωνικό σφάλμα μεταξύ των προβλέψεων και των πραγματικών τιμών.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2,$$

όπου Y_i πραγματικές τιμές και $\hat{Y}_i$ οι προβλέψεις του μοντέλου την δεδομένη στιγμή.

- Root Mean Squared Error: Υπολογίζει την ρίζα του MSE. Χρησιμοποείται περισσότερο γιατί έτσι το σφάλμα έχει πιο ρεαλιστική τιμή, μιας και το MSE υπολογίζει τον μέσο όρο των τεταγώνων,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2},$$

όπου Y_i πραγματικές τιμές και $\hat{Y}_i$ οι προβλέψεις του μοντέλου την δεδομένη στιγμή.

- Mean Absolute Error: Χρησιμοποιείται και αυτή για προβλήματα παλινδρόμησης όμως πιο σπάνια σε σχέση με την Mean Squared Error. Θυμίζει την Mean Squared Error όμως αυτή η συνάρτηση υπολογίζει το μέσο όρο της απόλυτης τιμής των σφαλμάτων.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|,$$

όπου Y_i πραγματικές τιμές και $\hat{Y}_i$ οι προβλέψεις.

- Cross-Entropy Loss: Χρησιμοποιείται κυρίως σε προβλήματα ταξινόμησης. Συγκρίνει τις πιθανοτικές κατανομές των προβλέψεων και των πραγματικών τιμών. Όταν οι δύο κατανομές αποκλίνουν το σφάλμα αυξάνεται, ενώ όταν ταυτίζονται το σφάλμα τείνει στο μηδέν.

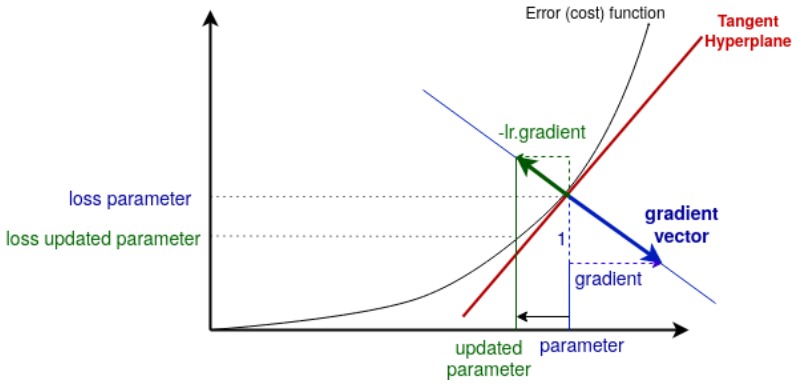
$$CrossEntropy(P, Q) = -\sum P(x)log(Q(x)),$$

όπου $P(x)$ η κατανομή των πραγματικών τιμών και $Q(x)$ η κατανομή των προβλεπόμενων τιμών.

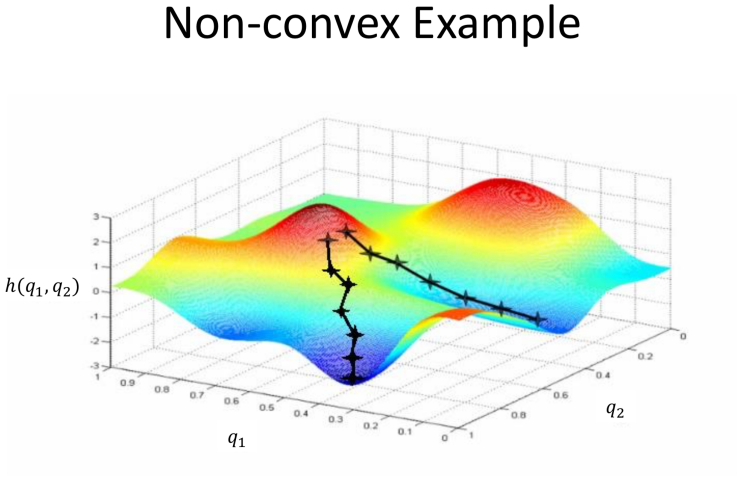
2.3.5 Αλγόριθμος Κατάβασης Κλίσης (Gradient descent)

Ο Gradient descent είναι ένας από τους πιο δημοφιλείς αλγόριθμους για την πραγματοποίηση βελτιστοποίησης και μακράν ο πιο κοινός τρόπος βελτιστοποίησης νευρωνικών δικτύων. Ταυτόχρονα, κάθε state-of-the-art βιβλιοθήκη Deep Learning περιλαμβάνει υλοποιήσεις διαφόρων αλγορίθμων για τη βελτιστοποίηση της κατάβασης κλίσης. Αυτοί οι αλγόριθμοι, ωστόσο, χρησιμοποιούνται συχνά ως βελτιστοποιητές μαύρου κουτιού, καθώς είναι δύσκολο να βρεθούν πρακτικές εξηγήσεις για τα πλεονεκτήματα και τις αδυναμίες τους.

Ένα νευρωνικό δίκτυο για να αξιολογήσει την εκμάθηση των παραμέτρων του ποσοτικοποιεί το σφάλμα του σε κάθε βήμα εκπαίδευσης. Συγκεκριμένα, μετράει μέσω μιας συνάρτησης κόστους την διαφορά μεταξύ της εκτιμώμενης τιμής του νευρωνικού και της πραγματικής τιμής. Ο αλγόριθμος Gradient descent προσπαθεί να βελτιστοποιήσει τις παραμέτρους του δικτύου μέσω της ελαχιστοποίησης της συνάρτησης κόστους $J(\theta)$. Για να υλοποιηθεί η ελαχιστοποίηση υπολογίζεται η κλίση της συνάρτησης, ώστε να μπορούμε να αποφασίσουμε σε ποια κατεύθυνση πρέπει να κινηθούμε. Το πρόβλημα που εμφανίζεται με την εκμάθηση μέσω του αλγορίθμου Gradient descent είναι ότι ο αλγόριθμος σταματάει στα τοπικά ελάχιστα στις μη-κυρτές καμπύλες και δεν βρίσκει την βέλτιστη λύση (ολικό ελάχιστο).



Εικόνα 2.6: *Gradient descent calculation*



Εικόνα 2.7: *Gradient descent on non-convex example*

Υπάρχουν τρία ήδη αλγορίθμων Gradient descent τα οποία διαφέρουν ως προς το πόσα δεδομένα χρησιμοποιούμε για τον υπολογισμό της διαβάθμισης της αντικειμενικής συνάρτησης[16].

1. **Batch Gradient Descent**:

   Αυτή είναι η πιο παραδοσιακή υλοποίηση του αλγορίθμου. Υπολογίζει την κλίση της συνάρτησης κόστους σε όλο το σύνολο δεδομένων εκπαίδευσης.

   $$\vartheta = \vartheta - \eta \nabla_{\vartheta} J(\vartheta)$$

   Δεδομένου ότι πρέπει να υπολογίσουμε τις διαβαθμίσεις για ολόκληρο το σύνολο δεδομένων για να εκτελέσουμε μόνο μία ενημέρωση, αυτή η εκδοχή μπορεί να είναι πολύ αργή και δυσδιάκριτη για σύνολα δεδομένων που δεν χωρά στη μνήμη. Η σύγκλιση επηρεάζεται από τον βαθμό εκμάθησης η (learning rate) που καθορίζει πόσο μεγάλο άλμα κάνουμε. Με αυτήν την μέθοδο είναι εγγυημένο ότι θα συγκλίνει σε ολικό ελάχιστο για κυρτές επιφάνειες και σε τοπικό ελάχιστο για μη-κυρτές.

2. **Stochastic Gradient Descent**:

   Ο Stochastic Gradient Descent (SGD) σε αντίθεση μια την πρώτη ενημέρωση παρα-

μέτρων για κάθε δείγμα εκπαίδευσης $x(i)$ και ετικέτα $y(i)$:

$$\vartheta = \vartheta - \eta \nabla_{\vartheta} J(\vartheta; x(i); y(i))$$

Η Batch Gradient Descent εκτελεί περιττούς υπολογισμούς για μεγάλα σύνολα δεδομένων, καθώς υπολογίζει τις κλίσεις για παρόμοια παραδείγματα πριν από κάθε ενημέρωση παραμέτρων. Η SGD καταργεί αυτήν την απόλυση εκτελώντας μία ενημέρωση κάθε φορά. Είναι λοιπόν συνήθως πολύ πιο γρήγορο και μπορεί επίσης να χρησιμοποιηθεί για μάθηση ταυτόχρονα (online). Αυτή η μέθοδος μπορεί να προσπεράσει κάποια τοπικά ελάχιστα, αν και έχει αποδειχτεί ότι για μικρό βαθμό εκμάθησης έχει ίδια συμπεριφορά με τον Batch Gradient Descent.

3. **Mini-Batch Gradient Descent**:

   Ο Mini-Batch Gradient Descent είναι τελικά η καλύτερη εκδοχή των δύο προηγούμενων. Εκτελεί μια ενημέρωση για κάθε μίνι-παρτίδα n παραδειγμάτων εκπαίδευσης.

$$\vartheta = \vartheta - \eta \nabla_{\vartheta} J(\vartheta; x(i : i+n); y(i : i+n))$$

   Με αυτόν τον τρόπο, μειώνονται οι πιθανότητες για σύγκλιση σε τοπικό ελάχιστο και γι' αυτό τον χρησιμοποιούν οι περισσότερες βιβλιοθήκες deep learning.

Το πρόβλημα που εμφανίζεται με αυτόν τον αλγόριθμο είναι ότι εαν η κλίση μηδενιστεί δεν μπορούν να ανανεωθούν οι παράμετροι του. Αυτό το πρόβλημα ονομάζεται εξαφάνιση παραγώγου (vanishing gradient).

2.3.6 Αλγόριθμος Οπισθοδιάδοσης Σφάλματος (Backpropagation)

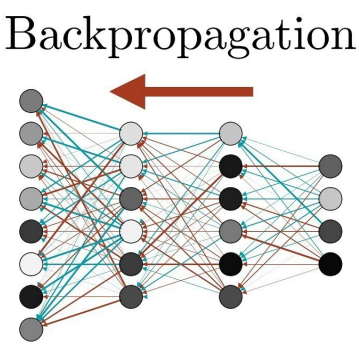
Η εκπαίδευση ενός νευρωνικού δικτύου χρησιμοποιεί τον αλγόριθμο οπισθοδρομικής διάδοσης (backpropagation). Συγκεκριμένα, πραγματοποιούνται δύο φάσεις[14]:

1. Στην πρώτη φάση όπου εξελίσσεται προς τα εμπρός, τα συναπτικά βάρη του δικτύου είναι σταθερά και το σήμα εισόδου διαδίδεται διαμέσου των επιπέδων του δικτύου προς την έξοδο.

2. Στην δεύτερη φάση υπολογίζεται το σφάλμα συγκρίνοντας το επιθυμητό αποτέλεσμα με το αποτέλεσμα που έχει παραχθεί στην πρώτη φάση, το σφάλμα αυτό διαδίδεται με αντίθετη κατεύθυνση, επίπεδο προς επίπεδο, μέχρι την είσοδο, προσαρμόζοντας ταυτόχρονα τα συναπτά βάρη του δικτύου.

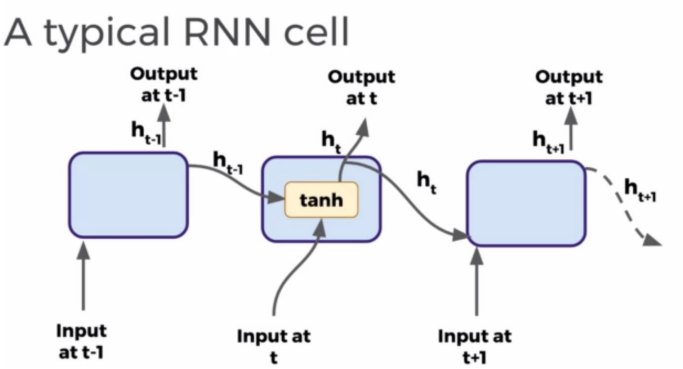
2.3.7 Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks)

Τα Αναδρομικά Νευρωνικά Δίκτυα είναι σχεδιασμένα με τέτοιον τρόπο ώστε να αποθηκεύουν και να επεξεργάζονται ιστορική πληροφορία. Γι' αυτόν τον λόγο είναι και τα πιο κατάλληλα για τις χρονοσειρές κάθε είδους. Τα προβλήματα που χρησιμοποιούν αναδρομικά νευρωνικά δίκτυα στην υλοποίηση τους είναι κυριως η αναγνώριση φωνής (speech recognition), η μετάφραση (translation) , η δημιουργία λεζάντας μίας εικόνας (image captioning),



Εικόνα 2.8: *Backpropagation*

η μεταφορά από βίντεο σε κείμενο (video-to-text), τα αυτοματοποιημένα μηχανήματα απαντήσεων (chatbots), και την πρόβλεψη χρονοσειρών (time-series forecasting).



Εικόνα 2.9: *Αρχιτεκτονική των αναδρομικών νευρωνικών δικτύων*

Η αρχιτεκτονική των αναδρομικών δικτύων βασίζεται στην σύνδεση προηγούμενων εισόδων με την τρέχουσα είσοδο του νευρωνικού δικτύου μέσω των κρυμμένων επιπέδων του. Με αυτόν τον τρόπο, το τεχνητό νευρωνικό δίκτυο αποκτά μίας μορφής μνήμη και έχει την αντίληψη του χρόνου γι' αυτό και χρησιμοποιείται σε χρονοεξαρτώμενα προβλήματα. Πιο φορμαλιστικά, ένα RNN ενός επιπέδου που την χρονική στιγμή t δέχεται είσοδο x_t παράγει την έξοδο y_t με την εξίσωση:

$$y_t = f(x_t, y_{t-1} = f(x_t, f(x_{t-1}, y_{t-2})) = \ldots = f(x_t, f(x_{t-1}, \ldots, f(x_1, y_0)))$$

Γενικότερα:

$$y_t = \phi(Wx_t + Uy_{t-1} + b),$$

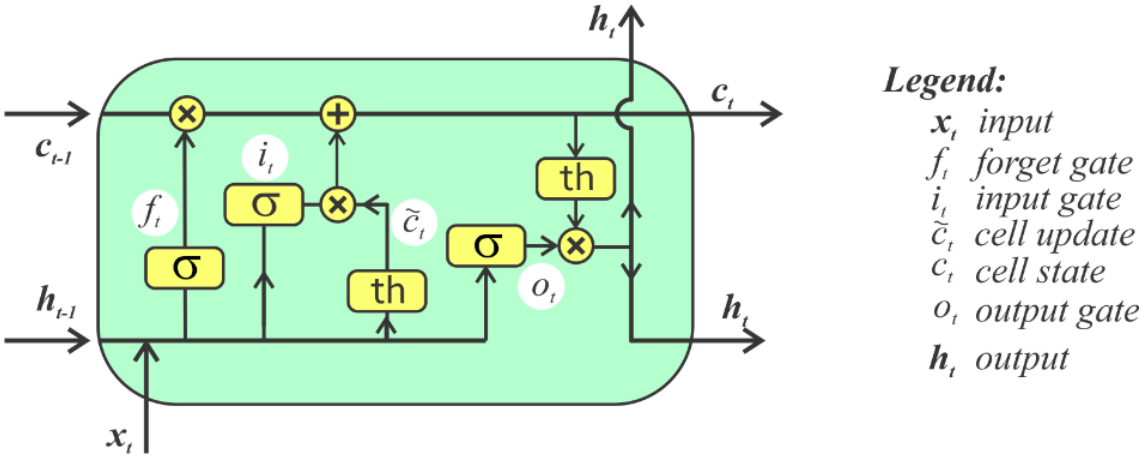
όπου:

- ϕ: συνάρτηση ενεργοποίησης,
- W: διανύσματα βαρών που επιδρούν στην είσοδο x_t,
- U: διάνυσμα βαρών που επιδρά στην έξοδο της προηγούμενης χρονικής στιγμής y_{t-1}
- b : διάνυσμα πόλωσης

Το πρόβλημα που αντιμετωπίζεται στα RNN είναι ότι δεν μπορούν να εκπαιδευτούν μέσω του αλγορίθμου gradient descent λόγω του vanishing gradient. Στην εκπαίδευση μέσω του backpropagation, εάν το αποτέλεσμα του προηγούμενου στρώματος στο τρέχον στρώμα είναι μικρό, τότε η τιμή κλίσης θα είναι μικρή. Εάν η κλίση του προηγούμενου στρώματος είναι μικρότερη, τότε η κλίση του τρέχοντος στρώματος θα είναι ακόμη μικρότερη. Αυτό κάνει τις κλίσεις να συρρικνώνονται εκθετικά καθώς προχωράμε πίσω. Μικρότερη κλίση σημαίνει ότι δεν θα επηρεάσει την ενημέρωση βάρους. Λόγω αυτού, το δίκτυο δεν μαθαίνει το αποτέλεσμα των προηγούμενων εισόδων. Έτσι, προκαλείται το πρόβλημα της βραχυπρόθεσμης μνήμης (short-term memory problem). Για να λυθεί αυτό το πρόβλημα δημιουργήθηκαν δύο αναδρομικά μοντέλα που θα παρουσιάσουμε στην συνέχεια το Long Short-Temp Memory (LSTM) και το Gated Recurrent Units (GRU).

2.3.8 Long Short-Term Memory (LSTM)

Οι Sepp Hochreiter, Jürgen Schmidhuber πρότειναν το 1997 την αρχιτεκτονική του LSTM[17] ώστε να αντιμετωπίσουν το πρόβλημα του vanishing gradient των RNNs. Ένα LSTM ακολουθεί την βασική αρχιτεκτονική των αναδρομικών νευρωνικών δικτύων προσθέτοντας εσωτερικούς μηχανισμούς-πύλες (gated cells). Πιο συγκεκριμένα οι πύλες που χρησιμοποιούνται είναι οι ακόλουθες:



Εικόνα 2.10: *LSTM unit*

1. Forget Gate:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

Το πρώτο βήμα του LSTM unit είναι να αποφασίσει ποια κομμάτια της πληροφορίας θα ξεχάσει. Λαμβάνει σαν είσοδο την έξοδο της προηγούμενης κατάστασης h_{t-1} και την είσοδο x_t και επιστρέφει μία τιμή μεταξύ του 0 και του 1 για κάθε cell state C_{t-1}.

2. Input Gate:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$\hat{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)$$

Το επόμενο βήμα είναι να αποφασίσει ποια νέα πληροφορία θα αποθηκευτεί στο cell state. Αυτό γίνεται με δύο βήματα: αρχικά, αποφασίζει ποιες τιμές θα ανανεωθούν (i_t) και στην συνέχεια δημιουργεί τον πίνακα $\hat{C}_t$ που θα αποθηκευτεί στο cell state.

3. Cell state update:

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t$$

Τώρα θα γίνει η ανανέωση του cell state απο C_{t-1} σε C_t λαμβάνοντας υπόψη τα προηγούμενα.

4. Output Gate:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

Σε αυτό το βήμα αποφασίζεται ποια μέρη του cell state θα λάβουν μέρος στην έξοδο του νευρώνα.

5. Hidden State:

$$h_t = o_t * tanh(C_t)$$

Τέλος, υπολογίζεται η έξοδος μέσω της συνάρτησης ενεργοποίησης tanh και το αποτέλεσμα της προηγούμενης πύλης.

Όπως κάθε νευρωνικό έχει βάρη τα οποία εκπαιδεύονται. Αυτά είναι τα W_f, W_i, W_c, W_o που αναφέρονται παραπάνω. Τα συνολικά των LSTMs καταλήγουν να είναι πολυάριθμα όσο αυξάνονται τα κρυμμένα επίπεδα του. Παρόλα αυτά λόγω της πύλης forget gate δεν υπάρχει το πρόβλημα με το vanishing gradient που σημαίνει ότι το LSTM μπορεί να μάθει εργασίες που απαιτούν μνήμη από γεγονότα που συνέβησαν χιλιάδες ή ακόμα και εκατομμύρια χρονικά βήματα (time steps) νωρίτερα.

2.3.9 Gated Recurrent Units (GRU)

Το 2014 προτάθηκε από τον Kyunghyun Cho και την ομάδα του[18] μία άλλη αρχιτεκτονική αναδρομικών νευρωνικών δικτύων. Το μοντέλο τους βασίζεται στο LSTM, επειδή και αυτό έχει πύλες που διαχειρίζεται την πληροφορία σε κάθε νευρώνα, όμως δεν έχουν ξεχωριστά κελιά για να αποθηκεύουν το cell state. Επίσης, οι πύλες τους είναι λιγότερες, μιας και λείπει η πύλη output gate και η forget gate, τις οποίες αντικαθιστα η πύλη update gate.

Πιο αναλυτικά η διαδικασία που χρησιμοποιεί το GRU είναι οι παρακάτω:

1. Update Gate:

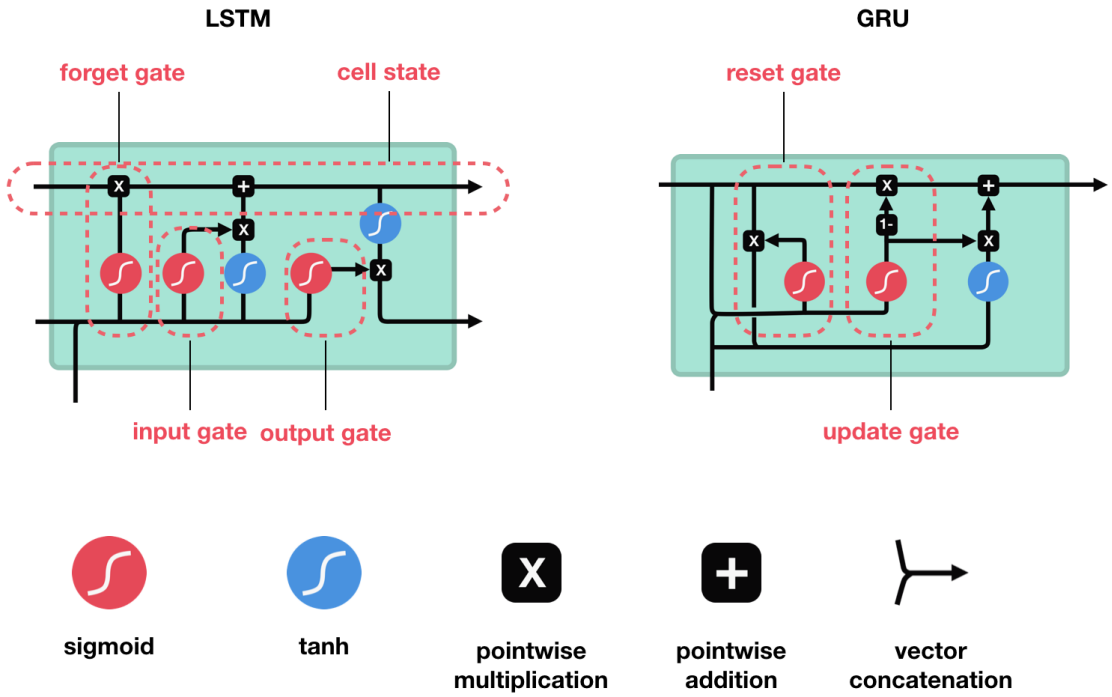
$$z_t = \sigma(W_s[h_{t-1}, x_t])$$

Αποφασίζει εάν η κατάσταση του κελιού πρέπει να ενημερωθεί με την υποψήφια κατάσταση (τρέχουσα τιμή ενεργοποίησης) ή όχι.

2. Reset Gate:

$$r_t = \sigma(W_r[h_{t-1}, x_t])$$

χρησιμοποιείται για να αποφασίσει εάν η προηγούμενη κατάσταση κελιού είναι σημαντική ή όχι. Η πύλη αυτή δεν χρησιμοποιείται στην απλή μορφη του GRU.



Εικόνα 2.11: *LSTM vs GRU*

3. Candidate Activation:

$$\hat{h}_t = tanh(W[r_t * h_{t-1}, x_t])$$

Εκτελείται η συνάρτηση ενεργοποίησης tanh.

4. Output:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t$$

Η τελική κατάσταση κελιού εξαρτάται από την update gate. Μπορεί ή όχι να ενημερωθεί με την candidate activation. Αφαιρεί κάποιο περιεχόμενο από την τελευταία κατάσταση του κελιού και γράφει κάποιο νέο περιεχόμενο.

Μετά από πειράματα έχει φανεί ότι το GRU τις περισσότερες φορές είναι ισάξιο με το LSTM. Δεδομένου όμως του γεγονότος ότι το GRU έχει λιγότερες παραμέτρους και δεν έχει το cell state, προτιμάται λόγω χαμηλότερης χρήσης της μνήμης και γιατί εκπαιδεύεται πιο γρήγορα.

2.3.10 Multilayer Perceptron

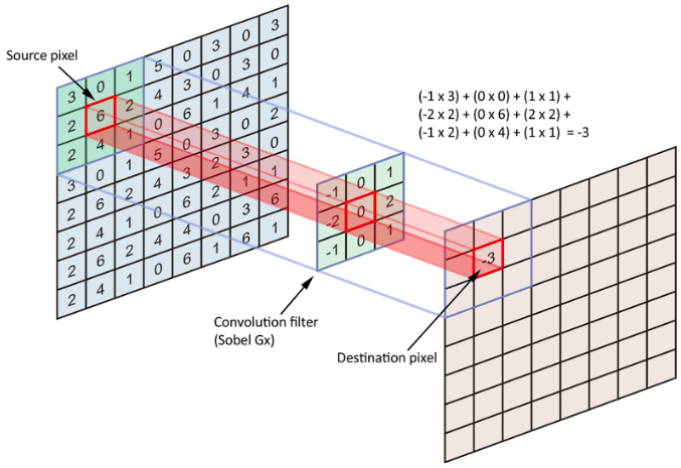
Η πιο γνωστή μορφή νευρωνικών δικτύων είναι τα Perceptrons. Η σύνδεση μεταξύ δύο ή περισσοτέρων Perceptrons δημιουργεί μια συνάρτηση που έχει την δυνατότητα να λύσει μη γραμμικά διαχωρίσιμα προβλήματα (όπως το πρόβλημα XOR). Στα βαθιά νευρωνικά δίκτυα συχνά χρησιμοποιούνται πλήρως συνδεδεμένα Perceptrons. Χωρίζονται σε επίπεδα layers και κάθε επίπεδο συνδέεται πλήρως με το αμέσως επόμενο. Αυτή η αρχιτεκτονική λέγεται Multi-layer Perceptron ή MLP. Ένα MLP αποτελείται από τουλάχιστον τρία επίπεδα κόμβων: ένα επίπεδο εισόδου, ένα κρυφό επίπεδο και ένα επίπεδο εξόδου. Εκτός από τους

κόμβους εισόδου, κάθε κόμβος είναι ένας νευρώνας που χρησιμοποιεί μια συνάρτηση μη γραμμικής ενεργοποίησης. Το MLP χρησιμοποιεί τον αλγόριθμο backpropagation (2.3.6) για εκπαίδευση. Τα πολλαπλά επίπεδα και η μη γραμμική ενεργοποίηση διακρίνουν το MLP από ένα απλό γραμμικό perceptron. Ένα παράδειγμα MLP είναι αυτό που απεικονίζεται στην Εικόνα 2.5.

2.3.11 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks)

Τα συνελικτικά νευρωνικά δίκτυα (CNN ή ConvNet) είναι μια ειδική κατηγορία νευρωνικών δικτύων τα οποία χρησιμοποιούνται κυρίως στην ανάλυση/κατηγοριοποίηση εικόνων και βίντεο. Ωστόσο, συνελικτικά δίκτυα έχουν χρησιμοποιηθεί και στην ανάλυση και πρόβλεψη χρονοσειρών[19][20]. Η αρχιτεκτονική τους παρομοιάζεται με τα MLP(2.3.10) όμως, όπως λέει και το όνομα τους χρησιμοποιούν την μαθηματική πράξη της συνέλιξης.

Κάθε επίπεδο συνέλιξης χρησιμοποιεί ένα σύνολο από φίλτρα, τα οποία ¨σαρώνουν¨ κατά πλάτος και κατά ύψος τον πίνακα εισόδου, με χρήση της μαθηματικής πράξης της συνέλιξης. Έτσι εντοπίζονται συγκεκριμένα χαρακτηριστικά ή μοτίβα που παρουσιάζονται στα δεδομένα εισόδου. Ο νέος πίνακας που δημιουργείται ονομάζεται χάρτης χαρακτηριστικών (feature map).



Εικόνα 2.12: *Παράδειγμα συνέλιξης*

Ένα συνελικτικό επίπεδο αποτελείται από τα παρακάτω χαρακτηριστικά:

1. Μέγεθος φίλτρου-kernel size: Ορίζει το μέγεθος του convolutional filter, όπως φαίνεται και στην Εικόνα 2.12.

2. Αριθμός φίλτρων-filters : Ορίζει το πλήθος των φίλτρων που θα χρησιμοποιηθούν για την εξαγωγή χαρακτηριστικών.

3. Βήμα-stride: Ρυθμίζει το βήμα μετακίνησης του φίλτρου. Όσο μεγαλύτερο το βήμα τόσο μικρότερη η διάσταση εξόδου.

4. Γέμισμα περιθωρίου-padding: Ορίζει το μέγεθος του γεμίσματος γύρω από τις άκρες της εισόδου.

Το μέγεθος της εξόδου ενός συνελικτικού επιπέδου καθορίζεται από όλα τα παραπάνω χαρακτηριστικά με βάση την σχέση:

$$\frac{N + 2P - F}{S} + 1, \text{όπου:}$$

- N = Διαστάσεις εισόδου
- P = Γέμισμα περιθωρίου (padding)
- F = Διαστάσεις φίλτρου
- S = Βήμα (stride)

Τα συνελικτικά δίκτυα, τις περισσότερες φορές, μετά από το συνελεκτικό επίπεδο κάνουν μια υποδειγματοληψία του υπολογισμού. Αυτή την διαδικασία ονομάζουμε pooling. Τα pooling layers μειώνουν τις διαστάσεις των δεδομένων συγκεντρώνοντας τις εξόδους των νευρώνων σε ομάδες κρατώντας τα πιο σημαντικά χαρακτηριστικά. Το local pooling κάνει μικρές ομαδοποιήσεις, συνήθως 2 x 2. Αντίθετα, το global pooling επιδρά σε όλους τους νευρώνες του convolutional layer. Η τεχνική του pooling, υπολογίζει τη μέγιστη τιμή (max pooling) ή τον μέσο όρο (average pooling) των τιμών του πίνακα. Η μείωση των διαστάσεων οδηγεί σε λιγότερους υπολογισμούς και μειώνει σε μεγάλο βαθμό την πιθανότητα υπερεκπαίδευσης του μοντέλου. Ωστόσο, πρέπει να είμαστε προσεκτικοί με την χρήση τους, γιατί μπορεί να χαθεί σημαντική πληροφορία, όταν τα προσθέτουμε αλόγιστα.

Το τελευταίο στάδιο ενός συνελικτικού δικτύου αποτελείται από το πλήρως συνδεδεμένο επίπεδο (fully-connected layer). Σε αυτό το στάδιο, αφού έχουν συλλεχθεί τα σημαντικά χαρακτηριστικά της εισόδου, συνδυάζονται έτσι ώστε να υπολογιστεί η τελική έξοδος.

Στην περίπτωση μας, δίνοντας μια χρονοσειρά σαν είσοδο υπολογίζεται η συνέλιξη ανάλογα με τα χαρακτηριστικά του πρώτου σταδίου (convolutional layer). Έτσι, λαμβάνουμε τα πιθανά μοτίβα που εμφανίζονται στην χρονοσειρά. Στην συνέχεια, κρατάμε την πιο σημαντική πληροφορία μέσω του δεύτερου σταδίου (pooling layer) και στο τέλος υπολογίζουμε την πρόβλεψη μας μέσω του fully-connected layer.

2.3.12 Αυτοκωδικοποιητής (Autoencoder)

Ο Αυτοκωδικοποιητής (Autoencoder) είναι ένα τεχνητό νευρωνικό δίκτυο το οποίο έχει ίδια είσοδο και έξοδο. Στην ουσία, προσπαθεί να μάθει την είσοδο έτσι ώστε να την παράγει στην έξοδο. Χρησιμοποιείται κυρίως σε προβλήματα μείωσης διαστάσεων (dimensionality reduce), information retrieval tasks[21] και data denoising. Ωστόσο, γίνεται αναφορά στους αποκωδικοποιητές και σε προβλήματα πρόβλεψης χρονοσειρών[22],[23],[24],[25] καθώς και σε προβλήματα anomaly detection[26].

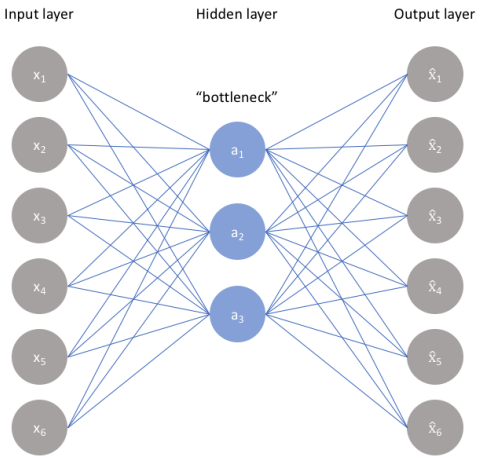
Single Autoencoder

Ένας απλός αυτοκωδικοποιητής (single autoencoder) αποτελείται από μια διαδοχικά συνδεδεμένη διάταξη τριών επιπέδων όπως φαίνεται και στην εικόνα 2.13:

1. το στρώμα εισόδου (Input layer)

2. το κρυφό στρώμα (Hidden layer)

3. το στρώμα εξόδου (ή ανακατασκευή) (Output layer)



Εικόνα 2.13: *Single autoencoder*

Ο στόχος της εκπαίδευσης του single autoencoder είναι να ελαχιστοποιηθεί το σφάλμα μεταξύ του διανύσματος εισόδου και του διανύσματος ανακατασκευής. Το πρώτο βήμα της εμπρόσθιας διάδοσης του single AE είναι η αντιστοίχηση της εισόδου στο κρυφό στρώμα (μειώνοντας την διαστατικότητα), ενώ το δεύτερο βήμα είναι η ανακασκευή της εισόδου χαρτογραφώντας τον κρυφό στρώμα στο στρώμα εξόδου ή ανακατασκευής. Τα δύο βήματα μπορούν να διατυπωθούν ως εξής[22]:

$$a(x) = f(W_1 x + b_1)$$

$$x' = f(W_2 a(x) + b_2)$$

όπου $x \in R^k$ διάνυσμα εισόδου και $x' \in R^k$ διάνυσμα ανακατασκευής, $a(x)$ το κρυφό επίπεδο, W_1, W_2 τα βάρη του κρυφού επιπέδου και της ανακατασκευής και b_1, b_2 τα bias αντίστοιχα.

Stacked autoencoder

Ο stacked autoencoder είναι μια ιεραρχημένη στοίβα από single autoencoders. Η διαδικασία εκπαίδευσης μοντέλου περιλαμβάνει άπληστη εκπαίδευση σε επίπεδο στρώματος για την ελαχιστοποίηση του σφάλματος μεταξύ των διανυσμάτων εισόδου και εξόδου. Το επόμενο επίπεδο του autoencoder είναι το κρυφό στρώμα του προηγούμενου, με καθένα από τα επίπεδα να εκπαιδεύεται από αλγόριθμο καθόδου κλίσης χρησιμοποιώντας μια συνάρτηση βελτιστοποίησης. Για την πρόβλεψη χρονοσειρών χρησιμοποιούνται κυρίως LSTM-SAE αλλά έχουν προταθεί και υβριδικά μοντέλα όπως μοντέλα με encoder συνελεκτικό νευρωνικό δίκτυο (Convolutional Neural Network) και decoder LSTM[19].

2.3.13 Ενισχυτική Μάθηση (Reinforcement Learning)

Η ενισχυτική μάθηση είναι ένας κλάδος της μηχανικής μάθησης που βασίζεται στην αλληλεπίδραση ενός πράκτορα (agent) με ένα περιβάλλον (environment), μέσω συγκεκριμένων δράσεων (actions) που πραγματοποιεί. Οι ενέργειες αυτές μεταβάλουν την κατάσταση (state) αποδίδοντας μία ανταμοιβή (reward), όπως φαίνεται στην Εικόνα 2.14. Πιο συγκεκριμένα, τα κύρια στοιχεία του reinforcement learning είναι τα ακόλουθα[27]:

1. **Environment:** Αποτελεί οποιοδήποτε σύστημα ανάλογα με το πρόβλημα (όπως οικονομικά και τεχνικά συστήματα). Εξαρτάται από τα ιστορικά στοιχεία και τα actions που εκτελούνται μέσω του agent. Σε κάθε αλληλεπίδραση, στέλνει ένα reward R_t στον agent που χρησιμεύει ως κριτήριο αξιολόγησης για τη δράση του agent στην τελευταία κατάσταση του συστήματος. Η κατάσταση (state) s_t του συστήματος μπορεί να είναι διακριτή ή συνεχής.

2. **Agent:** Λειτουργεί ως ελεγκτής του συστήματος. Μπορεί τουλάχιστον να παρατηρήσει εν μέρει την κατάσταση του συστήματος λαμβάνοντας observations x_t. Χρησιμοποιώντας αυτές, αλληλεπιδρά με το environment εκτελώντας actions και με αντάλλαγμα ανακτώντας reward R_{t+1}, τις οποίες μπορεί να χρησιμοποιήσει για να βελτιώσει την πολιτική της.

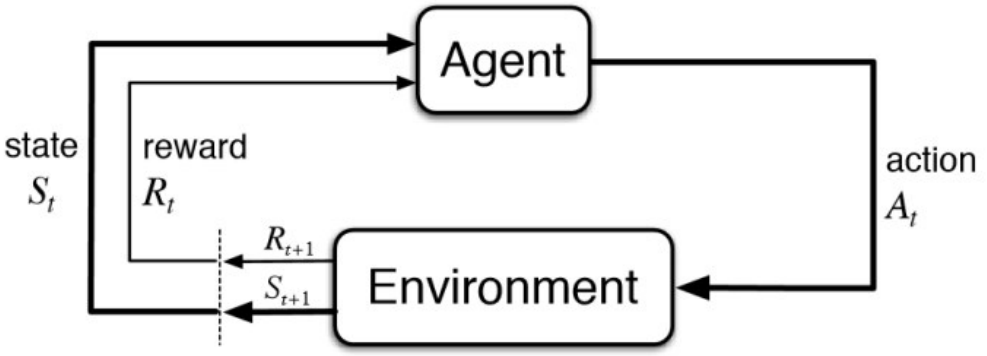
3. **Action:** Το action επηρεάζει την ανάπτυξη του environment. Ουσιαστικά αναπαριστούν την επιλεγμένη αλλαγή στο state του συστήματος. Τα actions μπορούν να περιοριστούν ανάλογα με το είδος του προβλήματος. Παραδείγματα από actions είναι η αγορά ή η πώληση προϊόντος ή στην περίπτωση μας η αυξηση ή η μείωση από resources.

4. **Policy:** Η αντιστοίχιση από το κάθε state του συστήματος σε κάθε action γίνεται με βάση μίας πολιτικής (policy) π. Ακολουθώντας μια πολιτική σχηματίζεται μια ακολουθία ενεργειών και αντικατροπτίζει τη συμπεριφορά του agent σε μια δεδομένη στιγμή. Για τις περισσότερες εφαρμογές κάποιος ενδιαφέρεται να ελέγξει ένα σύστημα για μια συγκεκριμένη χρονική περίοδο και όχι απλά για την επόμενη στιγμή. Επομένως, αντί για βελτιστοποίηση ενός βήματος, κάποιος προσπαθεί να καθορίσει μια βέλτιστη πολιτική σε σχέση με έναν δεδομένο συνολικό στόχο, ανάλογα με τη reward function.

5. **Reward:** Η reward function καθορίζει τον συνολικό στόχο του συστήματος. Απεικονίζει την άμεση ανταμοιβή που λαμβάνει ο agent για την εκτέλεση του συγκεκριμένου action στο δεδομένο state συστήματος. Το μοντέλο βάση και της πολιτικής κοιτάζει την μακροπρόθεσμα καλύτερη ανταμοιβή και όχι αυτή που την επόμενη στιγμή θα φέρει το καλύτερο αποτέλεσμα και στην συνέχεια μπορεί να έχει πολύ μικρότερες τιμές.

Exploration/Exploitation

Το δίλημμα της "εξερεύνησης - εκμετάλλευσης". Όταν ο πράκτορας ξεκινά να αλληλεπιδρά με το περιβάλλον δε γνωρίζει τίποτα για αυτό και κατά συνέπεια δρα στην τύχη.



Εικόνα 2.14: *Agent-Environment*

Καθώς όμως συνεχίζεται αυτή η διαδικασία αρχίζει και συσσωρεύει κάποια γνώση για το περιβάλλον και την οποία είναι σε θέση να εκμεταλλευτεί ώστε να λάβει τις αποφάσεις του. Σε ποιο βαθμό όμως μπορεί να εκμεταλλεύεται την υφιστάμενη γνώση και σε ποιο συνεχίζει την εξερεύνηση·[28]

- Από τη μία η άμεση εκμετάλλευση μπορεί να οδηγήσει σε μη βέλτιστες αποφάσεις καθώς είναι πιθανόν να υπάρχουν καταστάσεις του περιβάλλοντος που δεν έχουμε εξερευνήσει ακόμα και οι οποίες να οδηγούν σε καλύτερα αποτελέσματα.
- Από την άλλη η συνέχιση της εξερεύνησης σημαίνει πως θυσιάζουμε ένα ποσό ανταμοιβής που μπορούμε άμεσα να λάβουμε.

2.3.14 Μαρκοβιανή Διαδικασία Αποφάσεων (Markov Decision Process)

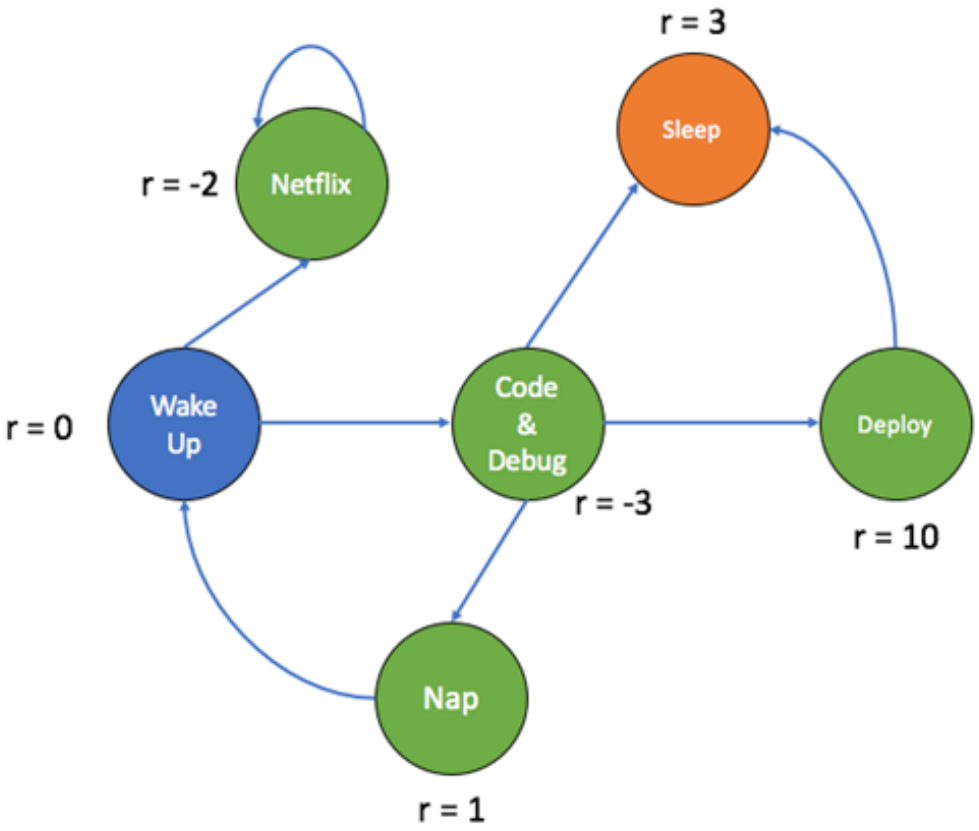
Η Μαρκοβιανή διαδικασία αποφάσεων (MDP) αποτελεί την μαθηματική βάση στα περισσότερα θεωρητικά προβλήματα RL. Μία MDP ορίζεται ως ένα σύνολο (S, A, P, R, γ) όπου:

- S ένα πεπερασμένο σύνολο από states,
- A ένα πεπερασμένο σύνολο από actions,
- P η πιθανότητα με δεδομένο state s, action a την χρονική στιγμη t να μεταβεί στο state s' $Pr(s_{t+1} = s' | s_t = s, a_t = a)$,
- R η ανταμοιβή που λαμβάνεται μετά από την αλλαγή κατάστασης από s_t σε s_{t+1} εξαίτίας του action a $R_s = E[R_{t+1} | s_t = s, a_t = a]$,
- γ ένας παράγοντας έκπτωσης (discount factor) όπου $\gamma \in [0, 1]$

Στη θεωρία πιθανοτήτων και στην στατιστική, ο όρος **Markov property** αναφέρεται στην ιδιότητα μιας στοχαστικής διαδικασίας χωρίς μνήμη ("Future is Independent of the past given the present"[29]), η οποία ορίζεται ως:

$$Pr(s_{t+1} | s_1, s_2, ..., s_t) = Pr(s_{t+1} | s_t)$$

όπου $Pr(s_1, s_2, ..., s_t) > 0$



Εικόνα 2.15: *Παράδειγμα MDP*

Ο στόχος του MDP για συνεχή προβλήματα (episodic and continuous tasks) είναι να μεγιστοποιήσει την ανταμοιβή:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Στην συνέχεια, αναφέρονται κάποιες σημαντικές συναρτήσεις του MDP[29].

Συνάρτηση πολιτικής (Policy Function)

Η συνάρτηση πολιτικής είναι μια πιθανοτική κατανομή στα actions $a \in A$ για κάθε state $s \in S$. Εάν ένας πράκτορας την στιγμή t ακολουθεί την πολιτική $\pi(a|s)$ είναι η πιθανότητα ότι ο πράκτορας να αναλάβει δράση a στο συγκεκριμένο χρονικό βήμα t.

$$\pi(s|a) = P[A_t = a|S_t = s]$$

Συνάρτηση κατάστασης-τιμής (State-Value Function)

Η συνάρτηση κατάστασης-τιμής βρίσκει την τιμή σε μία κατάσταση. Η τιμή σε μία κατάσταση s, όταν ο πράκτορας ακολουθεί μια πολιτική π, ορίζεται ως $V_\pi(s)$. είναι η αναμενόμενη επιστροφή ξεκινώντας από το ς και ακολουθώντας μια πολιτική π για τις επόμενες καταστάσεις, μέχρι να φτάσουμε στην κατάσταση τερματισμού. Μπορούμε να το διατυπώσου-

με ως:

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$

Χρησιμοποιώντας την συνάρτηση Bellman μπορούμε να βρούμε την βέλτιστη συνάρτηση κατάστασης-τιμής $V^*_\pi(s)$ η οποία είναι:

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(s_{t+1})|S_t = s]$$

Συνάρτηση κατάστασης-δράσης (State-Action value Function or Q-Function)

Αυτή η συνάρτηση καθορίζει το πόσο καλό είναι για τον πράκτορα να αναλάβει δράση a σε μια κατάσταση s με πολιτική π. Μπορούμε να το διατυπώσουμε ως:

$$Q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a]$$

Όμοια με πριν, μέσω της συνάρτησης Bellman μπορούμε να βρούμε την βέλτιστη συνάρτηση κατάστασης-δράσης $Q^*_\pi(s)$ η οποία είναι:

$$Q_\pi(s, a) = E_\pi[R_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})|S_t = s, A_t = a]$$

2.3.15 Μάθηση Χρονικών Διαφορών (Temporal Difference Learning)

Ο Richard S. Sutton το 1988 εισήγαγε το Temporal Difference Learning - TD[30]. Είναι μια προσέγγιση για να μάθουμε πώς να προβλέψουμε μια ποσότητα που εξαρτάται από τις μελλοντικές τιμές ενός δεδομένου σήματος. Το όνομα TD προέρχεται από τη χρήση αλλαγών ή διαφορών, σε προβλέψεις για διαδοχικά χρονικά βήματα για την προώθηση της μαθησιακής διαδικασίας. Η πρόβλεψη σε οποιοδήποτε δεδομένο χρονικό βήμα ενημερώνεται για να την φέρει πιο κοντά στην πρόβλεψη της ίδιας ποσότητας στο επόμενο βήμα. Είναι μια εποπτευόμενη μαθησιακή διαδικασία στην οποία το εκπαιδευτικό σήμα για μια πρόβλεψη είναι μια μελλοντική πρόβλεψη. Οι αλγόριθμοι TD χρησιμοποιούνται συχνά στην εκμάθηση ενίσχυσης για να προβλέψουν ένα μέτρο του συνολικού ποσού ανταμοιβής που αναμένεται στο μέλλον, αλλά μπορούν επίσης να χρησιμοποιηθούν για την πρόβλεψη και άλλων ποσοτήτων. Έχουν επίσης αναπτυχθεί αλγόριθμοι συνεχούς χρόνου TD[31].

Η απλή μορφή το TD συμβολίζεται ως TD(0) και υπολογίζει το state-value V με τον παρακάτω τρόπο:

$$V(S_t) = V(S_t) + a[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

όπου α: learning rate, που καθορίζει σε ποιο βαθμό λαμβάνουμε υπόψιν τις νέες ανανεώσεις, γ: discount factor

Το TD χρησιμοποιεί την εξίσωση βελτιστοποίησης Bellman για να εκτιμήσει την τιμή και, στη συνέχεια, ενημερώνει την εκτιμώμενη τιμή με την τιμή-στόχο.

2.3.16 ε-greedy Policy

Η πολιτική αυτή έχει φτιαχτεί για να ξεπεραστεί το δίλημμα exploration/exploitation και χρησιμοποιείται συνήθως στο Q-Learning. Με αυτήν την τεχνική σε κάθε βήμα ο αλγόριθμος

με μια πιθανότητα ε λαμβάνει αποφάσεις τυχαία, ενώ διαφορετικά δρα άπληστα επιλέγοντας το action που εκτιμά ότι προσφέρει μεγαλύτερο reward[28].

2.3.17 Q-Learning

Ο αλγόριθμος Q-Learning[32] είναι ο πιο γνωστός αλγόριθμος reinforcement learning. Μπορούμε να πούμε ότι είναι μια μορφή ασύγχρονου δυναμικού προγραμματισμού. Χρησιμοποιεί την μέθοδο μάθησης χρονικών διαφορών (temporal difference learning) και την ε-greedy policy. Αναλυτικά ο αλγόριθμος φαίνεται παρακάτω (Αλγόριθμος 2.1).

Αλγοριθμος 2.1: *Αλγόριθμος Q-Learning*

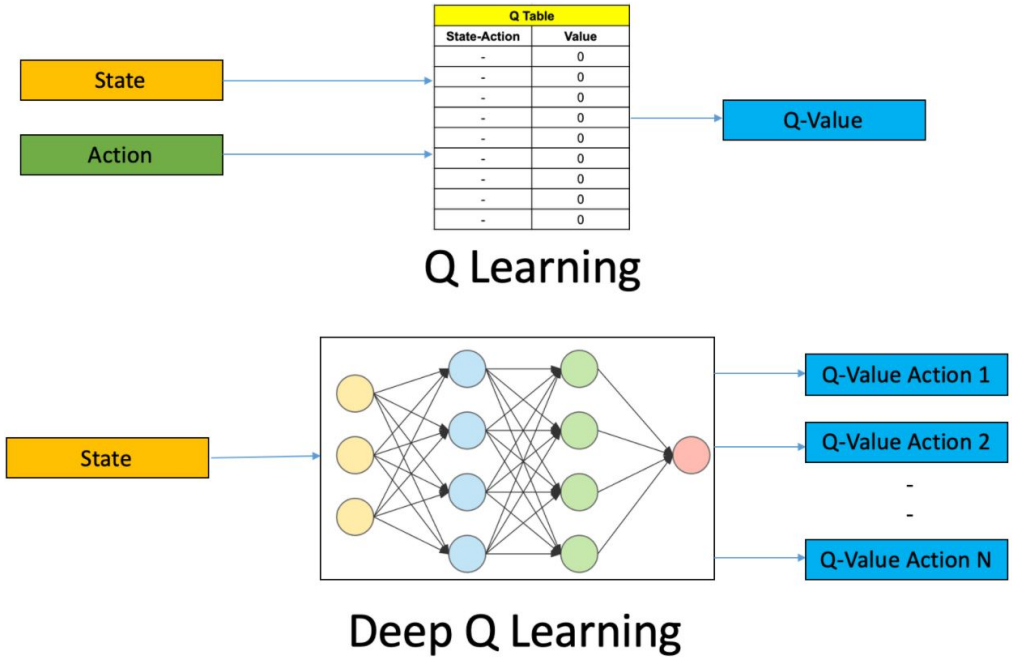
Αρχικοποίησε τον πίνακα Q με μηδενικά
for episode=1, to M **do**
  Πάρε το current state από το environment s_t
  for t=1, T **do**
    Με πιθανότητα ε διάλεξε ένα τυχαίο action a_t
    αλλιώς επέλεξε $a_t = max_a Q^*(s_t, a)$
    Εκτέλεσε το action a_t και υπολόγισε το reward r_t και το next state s_{t+1}
    Εκτέλεσε τον κανόνα:
    $Q(s_t, a_t) = Q(s_t, a_t) + a[r_t + \gamma max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)]$
    Θέσε $s_t = s_{t+1}$
  end for
end for

Πρέπει να σημειωθεί ότι ο αλγόριθμος υποθέτει ένα πεπερασμένο πλήθος καταστάσεων για την αναπαράσταση του περιβάλλοντος, οι οποίες αναπαρίσταται με έναν πίνακα στη μνήμη. Γίνεται εύκολα αντιληπτό ότι κάτι τέτοιο είναι αδύνατο για μεγάλο αριθμό καταστάσεων. Επιπλέον, η συγκεκριμένη λογική δεν εκμεταλλεύεται τις δυνατότητες γενίκευσης που υπάρχουν, επιχειρώντας μια σημειακή εκτίμηση της κάθε κατάστασης. Στην συνέχεια, θα παρουσιαστεί μια νέα μορφή του αλγορίθμου με χρήση νευρωνικών δικτύων.

2.3.18 Βαθιά Ενισχυτική Μάθηση

Λόγω των περιορισμών που εμφανίζονται στον αλγόριθμο Q-learning σχετικά με την μεγάλη μνήμη που χρειάζεται και τον μεγάλο χρόνο για να εξερευνήσει όλες τις καταστάσεις, η DeepMind το 2013[33] πρότεινε μια νέα μορφή του αλγορίθμου με χρήση τεχνητών νευρωνικών δικτύων. Ο νέος αλγόριθμος φτιάχτηκε αρχικά για τα παιχνίδια Atari 2600 τα οποία δεν μπορούσαν να υλοποιηθούν με απλό Q-learning λόγω του πλήθους των πιθανών καταστάσεων τους στο χώρο. Το Deep Q-Learning, υλοποιεί την Q-value function με χρήση ενός νευρωνικού δικτύου αντί για έναν πίνακα περασμένων καταστάσεων, όπως φαίνεται και στην Εικόνα 2.16. Στην περίπτωση των Atari 2600 χρησιμοποιήθηκαν συνελικτικά νευρωνικά δίκτυα (λόγω των εικόνων που λάμβανε σαν είσοδο), αλλά γενικά μπορούμε να χρησιμοποιήσουμε κάθε είδος νευρωνικού δικτύου (όπως MLP ή αναδρομικά δίκτυα).

Πιο αναλυτικά, τα στάδια που υλοποιούνται με χρήση deep Q-learning networks- DQN είναι τα εξής:



Εικόνα 2.16: *Q-Learning vs Deep Q-Learning*

1. Όλες οι προηγούμενες καταστάσεις αποθηκεύονται στην μνήμη ή σε έναν buffer
2. Το επόμενο action ορίζεται με βάση το μεγαλύτερο output του Q-network
3. Η συνάρτηση σφάλματος (loss function) είναι συνήθως η mean squared error η οποία συγκρίνει το predicted Q-value και το target Q-value Q^*. Αυτό είναι ένα βασικό πρόβλημα παλινδρόμησης. Ωστόσο, δεν γνωρίζουμε το target ή actual value εδώ μιας και ασχολούμαστε με πρόβλημα reinforcement learning. Χρησιμοποιούμε, λοιπόν, την Q-value update equation που προέρχεται από την Bellman equation ως εξής:

$$Q^*(s_t, a_t) = Q^*(s_t, a_t) + a[R_{t+1} + \gamma max_a Q^*(s_{t+1}, a) - Q^*(s_t, a_t)]$$

Επιπλέον, γίνεται χρήση και της τεχνικής του experience replay, όπου έχουμε έναν replay memory και από αυτόν επιλέγουμε mini batches με τα οποία εκπαιδεύουμε το νευρωνικό δίκτυο. Χρησιμοποιούμε δηλαδή τον αλγόριθμο Mini-Batch Gradient Descent που αναφέραμε παραπάνω. Ο πλήρης αλγόριθμος παρουσιάζεται στον Αλγόριθμο 2.2

Αλγοριθμος 2.2: *Αλγόριθμος Deep Q-Learning*

Αρχικοποίησε το νευρωνικό Q-network με τυχαία βάρη ϑ
Αρχικοποίησε την replay memory D με μέγεθος N
for episode=1, to M **do**
  Πάρε το current state από το environment s_t
  for t=1, T **do**
    Με πιθανότητα ε διάλεξε ένα τυχαίο action a_t
    αλλιώς επέλεξε $a_t = max_a Q^*(s_t, a; \vartheta)$
    Εκτέλεσε το action a_t και υπολόγισε το reward r_t και το next state s_{t+1}
    Αποθήκευσε στην μνήμη το transition (s_t, a_t, r_t, s_{t+1})
    Θέσε $s_t = s_{t+1}$
    Διαλέγουμε random minibatch of k transitions (s_j, a_j, r_j, s_{j+1}) από το D
    Θέσε $y_t = \begin{cases} r_j & \text{for terminal states,} \\ r_j + \gamma max_{a'} Q^*(s_{j+1}, a'; \vartheta) & \text{for non terminal states.} \end{cases}$
    Εκτέλεσε το gradient descent step on $(y_j - Q(s_j, a_j; \vartheta))^2$
  end for
end for

Κεφάλαιο 3

Σχετικές εργασίες

Στο κεφάλαιο αυτό αρχικά γίνεται μια περιγραφή των σχετικών εργασιών με βάση την πρόβλεψη χρονοσειρών και την ελαστικότητα στα υπολογιστικά νέφη.

Η δυναμική χρονοδρομολόγηση πόρων έχει ερευνηθεί από την σκοπιά της πρόβλεψης του φόρτου εργασίας σε πραγματικό χρόνο. Παρουσιάζουμε, λοιπόν, κάποιες τεχνικές που εφαρμόστηκαν είτε σε χρονοσειρές φόρτου εργασίας είτε σε άλλα προβλήματα για την πρόβλεψη χρονοσειρών.

Βασικά μοντέλα πρόβλεψης που χρησιμοποιούνται για το φόρτο εργασίας σε υπολογιστικά περιβάλλοντα υλοποιούνται εδώ [9] και εφαρμόζονται σε πραγματικά δεδομένα (Google cluster data, Intel Netbatch logs). Πιο συγκεκριμένα, υλοποιούνται τα μοντέλα : Αυτοπαλινδρομικό μοντέλο πρώτης τάξης, Μοντέλο κινητού μέσου όρου πρώτης τάξης, Απλή εκθετική εξομάλυνση, Διπλή εκθετική εξομάλυνση, Μέθοδος ETS, Αυτοματοποιημένη μέθοδος ARIMA και η Μέθοδος αυτοπαλινδρόμησης νευρικού δικτύου (ένα πλήρως συνδεδεμένο τριών επιπέδων νευρωνικό δίκτυο). Για να υπολογιστεί η ακρίβεια των μοντέλων εφαρμόστηκαν με δύο τρόπους. Στον πρώτο τρόπο (out-of-sample forecasting), εκπαιδεύονται τα μοντέλα με ιστορικά στοιχεία και προβλέπουμε τις επόμενες καταστάσεις με βάση αυτά. Αντίθετα, ο δεύτερο τρόπο (rolling forecast origin cross-validation) βασίζεται στην τεχνική του κυλιόμενου παραθύρου, όπου τα μοντέλα εκπαιδεύονται με τα τελευταία k στοιχεία, υπολογίζονται τα υπόλοιπα h και μετακινεί το παράθυρο. Παρατηρήθηκε ότι με λίγα μόνο ιστορικά δεδομένα (10 με 15 τιμές) τα μοντέλα βγάζουν αρκετά ακριβή αποτελέσματα, κυρίως για δύο ή τρεις περιόδους στο μέλλον.

Επιπλέον, μια μελέτη πρόβλεψης ηλιακής ενέργειας [24] πρότεινε μια μορφή αυτοκωδικοποιητή τον οποίο ονόμασε Auto-LSTM που αποτελείται από έναν Autoencoder και ένα LSTM το οποίο είχε καλύτερα αποτελέσματα από το MLP, LSTM, Deep Belief Network.

Το 2018 δημοσιεύτηκε μία άλλη έρευνα για την πρόβλεψη χρονοσειρών για τη φωτοβολταϊκή ηλιακή ενέργεια και το φορτίο ηλεκτρικής ενέργειας για την επόμενη μέρα[34]. Συγκρίθηκαν τρία διαφορετικά μοντέλα νευρωνικών δικτύων (συνελικτικά νευρωνικά δίκτυα-CNN, multilayer perceptron και LSTM). Τα αποτελέσματα απέδειξαν ότι το CNN και το Multilayer perceptron είχαν καλύτερα αποτελέσματα από το LSTM.

Επιπρόσθετα, το θέμα πρόβλεψης χρονοσειρών με δεδομένα τιμών bitcoin αναλύθηκε με χρήση των μοντέλων ARIMA και Facebook Prophet[35]. Αποδείχθηκε ότι το Prophet υπερτερεί κατά πολύ από το ARIMA. Στην συνέχεια, μια άλλη έρευνα της πρόβλεψης χρονοσειρών [36] σε δεδομένα τιμών του bitcoin έδειξε ότι το GRU έχει καλύτερη επίδοση από το LSTM,

όμως το ARIMA είχε ακόμα καλύτερα αποτελέσματα τόσο από άποψη ακρίβειας όσο και από άποψη χρόνου.

Άλλη μια πρόσφατη έρευνα του αφορά πρόβλεψη φόρτου εργασίας χρονοσειρών σε συστήματα αποθήκευσης προτείνει το CrystalLP[37], μια μέθοδο βαθιάς μάθησης. Αναλυτικότερα, για την πρόβλεψη χρησιμοποιεί αναδρομικά δίκτυα μακράς-βραχείας μνήμης - LSTM και τα συνδυάζει με επεξεργασία δεδομένων. Τα στάδια υλοποιήσης του CrystalLP είναι: αρχικά, η συλλογή δεδομένων για τον φόρτο εργασίας, η προ-επεξεργασία των δεδομένων (ομαλοποίηση των δεδομένων), η πρόβλεψη με βάση των LSTMs και η μετα-επεξεργασία των δεδομένων. Η πρόβλεψη γίνεται με βήμα 1 τιμή, χρησιμοποιώντας ένα κυλιόμενο παράθυρο. Το μοντέλο συγκρίθηκε με το ARIMA, SVR και το απλό-RNN και αποδείχθηκε ότι έχει καλύτερη απόδοση.

Επίσης, η αυτοματοποιημένη διαχείριση πόρων έχει ερευνηθεί και με την χρήση της ελαστικότητας και της ενισχυτικής μάθησης.

Ο Tiramola [38] είναι μία υπηρεσία που χρησιμοποιείται για να αυξομειώνει αυτόματα το μέγεθος των υπολογιστικών συστάδων σε περιβάλλοντα υπολογιστικών νεφών. Ο Tiramola σχεδιάστηκε για να διαχειρίζεται αυτόματα και σε πραγματικό χρόνο την αύξηση ή την μείωση των υπολογιστικών πόρων, σε NoSQL συστάδες, ανάλογα με τις προδιαγραφές και τις απαιτήσεις του εκάστοτε χρήστη. Η λήψη αποφάσεων σχετικά με τις προτεινόμενες ενέργειες γίνεται μοντελοποιώντας τη συστάδα ως μια μαρκοβιανή διαδικασία αποφάσεων που προσδιορίζει συνεχώς την πιο επωφελής δράση ανάλογα με την συνάρτηση επιβράβευσης που ορίζει ο χρήστης. Το πρόβλημα με αυτήν την υλοποίηση ήταν ότι στις μαρκοβιανές διαδικασίες ο χώρος καταστάσεων είναι διακριτές τιμές και οι παράμετροι εισόδου είναι συνεχείς μεταβλητές. Αυτό οδηγεί σε έναν πολύ μεγάλο χώρο καταστάσεων που είναι δύσκολο να διαχειριστεί η υπηρεσία.

Με βάση την αρχιτεκτονική του Tiramola έγιναν περαιτέρω έρευνες ώστε να βρεθεί μια πιο αποδοτική τεχνική λήψης αποφάσεων. Το 2017[39] προτάθηκε ένα άλλο σύστημα αποφάσεων που βασίζεται σε μια προσαρμοσμένη μορφή ενισχυτικής μάθησης. Η προσέγγιση αυτή προτείνει έναν αλγόριθμο που μοντελοποιεί το περιβάλλον σαν μια μαρκοβιανή διαδικασία και χρησιμοποιεί δέντρα αποφάσεων χωρίζοντας δυναμικά τον χώρο κατάστασης όταν χρειάζεται, σύμφωνα με τις οδηγίες της συμπεριφοράς του συστήματος. Με αυτόν τον τρόπο μπορεί να γενικεύσει τις καταστάσεις του περιβάλλοντος ώστε να λειτουργεί για μεγαλύτερο χώρου καταστάσεων σε σχέση με την αρχική υλοποίηση.

Εν συνεχεία των παραπάνω, το 2018 προτάθηκε το DERP[40], το οποίο είναι μια πρωτοποριακή υπηρεσία δυναμικής διαχείρισης υπολογιστικών πόρων με χρήση βαθιάς ενισχυτικής μάθησης. Λαμβάνοντας υπόψιν τον αλγόριθμο βαθιάς ενισχυτικής μάθησης που παρουσίασε η DeepMind[33] δημιουργήθηκαν τρεις διαφορετικοί πράκτορες (Single Deep Q-Learning, Full Deep Q-Learning, Double Deep Q-Learning) που αναλάβαναν την λήψη αποφάσεων στο περιβάλλον των υπολογιστικών νεφών. Αποδείχτηκε ότι αυτή η τεχνική ήταν περισσότερο αποδοτική από τις προηγούμενες.

Μια ακόμα προσέγγιση με ενισχυτική μάθηση είναι αυτή [41], η οποία δημιουργεί έναν κατανεμμημένο ελεγκτή ενισχυτικής μάθησης που διαχειρίζεται αποδοτικότερα τους πόρους. Συγκεκριμένα είναι σε θέση όχι μόνο να αυξήσει τους πόρους πιο γρήγορα για να καλύψει

την αυξανόμενη ζήτηση αλλά και να τους μειώσει ωστε να ελαχιστοποιθεί η πλεονάζουσα χρήση πόρων και ο διακομιστής να εξοικονομήσει τρέχον κόστος.

Τέλος, προτάθηκε ένα σύστημα[42] που χρησιμοποίησε τον πράκτορα βαθιάς ενισχυτικής μάθησης σε συνδυασμό με έναν ελεγκτή για να μπορέσει να βελτιστοποιήσει την κατανάλωση ενέργειας και την εκτέλεση εργασιών έχοντας εμφανή καλύτερες επιδόσεις σε σύγκριση με τον βασικό αλγόριθμο Deep Q-Learning. Έτσι κατανοούμε ότι ο πράκτορας βαθιάς ενισχυτικής μάθησης μπορεί να συνδυαστεί με διαφορετικές τεχνικές και δημιουργήσει πιο αποδοτικά αποτελέσματα.

Κεφάλαιο 4

Περιγραφή, Σχεδίαση και Υλοποίηση

Στο κεφάλαιο αυτό παρουσιάζεται η μελέτη που έγινε για την υλοποίηση του συστήματος. Αρχικά, περιγράφεται η αρχιτεκτονική του συστήματος και γίνεται ο διαχωρισμός του στα επιμέρους υποσυστήματα. Στην συνέχεια, αναφέρεται ο τρόπος υλοποίησης του συστήματος.

4.1 Αρχιτεκτονική συστήματος

Το περιβάλλον λειτουργίας του προβλήματος μας αποτελείται από ένα πλήθος παραμέτρων περιβάλλοντος οι οποίοι σχετίζονται τόσο με τα εσωτερικά χαρακτηριστικά της συστάδας της cloud υπηρεσία μας όσο και με εξωτερικούς παράγοντες όπως είναι εισερχόμενος φόρτος εργασίας. Δεδομένου ότι ο εξωτερικός παράγοντας του φόρτου εργασίας δεν έχει κάποια άμεση σχέση με τις σταθερές παραμέτρους της υπηρεσίας μας όπως το συνολικό capacity ή την συνολική CPU που διατείθονται από το cloud cluster και εξαρτάται μόνο από την συμπεριφορά των πελατών, μπορούμε να θεωρήσουμε το φόρτο εργασίας σαν μια χρονοσειρά. Υπάρχουν πολλές παρόμοιες προσεγγίσεις σε αυτό το πρόβλημα που έχουν αποδείξει ότι ο φόρτος εργασίας ακολουθεί κάποια μοτίβα σε χρονικές περιόδους και μπορεί έτσι να προβλεφθεί η επόμενη κατάσταση του με μοντέλα πρόβλεψης χρονοσειρών.

Πιο συγκεκριμένα, λοιπόν, το σύστημα μας βασίζεται σε δύο υποσυστήματα. Το ένα είναι η πρόβλεψη χρονοσειρών το οποίο χρησιμοποιούμε για να προβλέψουμε την κατάσταση του cloud περιβάλλοντός μας και το άλλο είναι η δυναμική διαχείριση πόρων με χρήση ενισχυτικής μάθησης, ώστε να προσθέτουμε ή να αφαιρούμε πόρους ανάλογα με την κατάσταση μας. Δεδομένου ότι έχουμε μία μελλοντική κατάσταση από το πρώτο υποσύστημα, η δυναμική διαχείριση πόρων έχει μία επιπλέον πληροφορία για το περιβάλλον του προβλήματος ώστε να αποτρέψει κάποιο scaling που δεν χρειάζεται να γίνει ή να προβλέψει από πριν μια μελλοντική αύξηση ή μείωση των πόρων ώστε να καλυφθούν οι ανάγκες των πελατών γρηγορότερα και με λιγότερο κόστος.

4.1.1 Υποσύστημα Πρόβλεψης Χρονοσειράς

Αρχικά, τα δύο υποσυστήματα μπορούν να μελετηθούν ξεχωριστά. Σχετικά με την πρόβλεψη χρονοσειρών υλοποιήθηκαν τα εξής μοντέλα:

- ARIMA

- Prophet
- LSTM
- GRU
- CNN
- AutoEncoder-LSTM

Στόχος τους είναι να μας δώσουν πληροφορίες για την μελλοντική κατάσταση του cloud περιβάλλοντος μας, η οποία εξαρτάται από εξωγενείς παράγοντες.

4.1.2 Υποσύστημα Δυναμικής Διαχείρισης Πόρων

Σχετικά με το σύστημα δυναμικής διαχείρισης πόρων υλοποιήθηκε ένα σύστημα βαθιάς ενισχυτικής μάθησης που ακολουθεί το αλγόριθμο Deep Q-Learning Network όπως ορίστηκε και στο Κεφάλαιο 2 (Αλγόριθμος 2.2).
Οι μεταβλητές που ορίζουν το περιβάλλον του συστήματος είναι οι εξής:

- Συνολικό workload
- Συνολικό capacity
- Ποσοστό χρησιμοποίησης της CPU
- Ποσοστό των read loads
- Ποσοστό της ελεύθερης RAM
- Το storage capacity του cluster
- Αριθμός δεσμευμένων CPU
- Αριθμός δεσμευμένων (VMs)
- Τα I/O events σε κάθε cluster

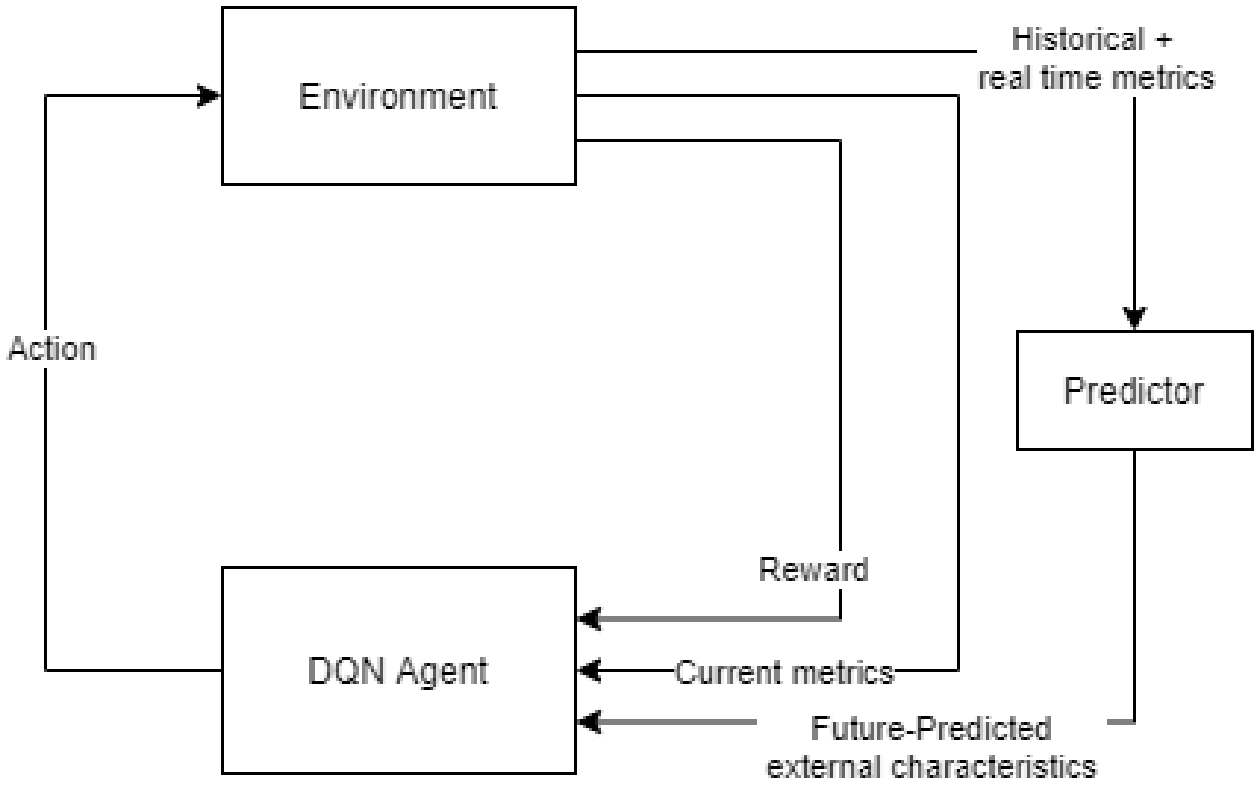
Έχει ήδη αποδειχτεί ότι ένα σύστημα με όλες τις παραπάνω μεταβλητές μπορεί να λειτουργήσει αποδοτικά στην σχετική βιβλιογραφία (DERP [40]).

4.2 Σχεδίαση Προτεινόμενου Συστήματος

Το προτεινόμενο σύστημα είναι ένας συνδυασμός των δύο παραπάνω συστημάτων. Αφού βρούμε τον predictor που είναι πιο αποδοτικός στα ιστορικά δεδομένα του cloud περιβάλλοντός μας, τον χρησιμοποιούμε στο υποσύστημα δυναμικής διαχείρισης πόρων. Πιο συγκεκριμένα, στις μεταβλητές περιβάλλοντος πλέον μπορούμε να προσθέσουμε και την πρόβλεψη του predictor για την επόμενη κατάσταση των μετρικών του περιβάλλοντος την οποία θα χρησιμοποιήσουμε για τον υπολογισμό της συνάρτησης Q-value. Η νέα κατάσταση state διαμορφώνεται με τα εξής χαρακτηριστικά:

- Τις μετρικές την δεδομένη στιγμή
- Τις μετρικές από την πρόβλεψη του predictor για την επομένη χρονική στιγμή
- Αριθμός των πόρων (VMs) που είναι δεσμευμένα εκείνη την στιγμή.

Ο νέος αλγόριθμος που προτείνεται, λοιπόν, είναι ο Αλγόριθμος 2.2 με διαφοροποίηση στις μεταβλητές του state όπου παίρνει πληροφορία και για την επόμενη κατάσταση μέσω του predictor. Το σύστημα μας λειτουργεί με την αρχιτεκτονική που απεικονίζεται στην Εικόνα 4.1.



Σχήμα 4.1: *Αρχιτεκτονική προτεινόμενου συστήματος*

4.3 Υλοποίηση Συστήματος

Το σύστημα υλοποιήθηκε σε Python 3.7.4, χρησιμοποιώντας τις βιβλιοθήκες:

- statsmodels.tsa για τα μοντέλα ARIMA,
- fbprophet για το Prophet,
- google tensorflow για τα νευρωνικά δίκτυα,
- sklearn για τον υπολογισμό των μετρικών αξιολόγησης των predictors,
- numpy για χρήση πινάκων και
- matplotlib.pyplot για απεικόνιση των δεδομένων

Κεφάλαιο 5

Πειραματική Αξιολόγηση

Στο κεφάλαιο αυτό γίνεται ο έλεγχος καλής λειτουργίας του συστήματος.

5.1 Πλαίσιο Πειραματικής Αξιολόγησης

Η αξιολόγηση του συστήματος μας έγινε με έναν αριθμό από προσομοιώσεις στο εικονικό περιβάλλον Google Colab[43]. Σε πρώτη φάση, αναζητήσαμε στο διαδίκτυο πραγματικά δεδομένα από φόρτο εργασίας σε συστάδες υπολογιστικών νεφών, αλλά δεν καταφέραμε να βρούμε κάποια βάση δεδομένων που θα μπορούσαμε να χρησιμοποιήσουμε στο πείραμα μας. Έτσι, αποφασίσαμε να φτιάξουμε μια συνθετική χρονοσειρά φόρτου εργασίας που να προσομοιάζει μια πραγματική χρονοσειρά, έχοντας αρκετές τυχαίες αυξομειώσεις στην εποχικότητα των δεδομένων της. Την μορφή της χρονοσειράς παρουσιάζουμε στο Κεφάλαιο 5.2. Στην συνέχεια, υλοποιήσαμε έξι διαφορετικά μοντέλα πρόβλεψης χρονοσειρών. Τα δύο από αυτά βασίζονται στην στατιστική ανάλυση των χρονοσειρών, χωρίζοντας τες σε επιμέρους χαρακτηριστικά και τα υπόλοιπα είναι μορφές τεχνητών νευρωνικών δικτύων που έχει αποδειχτεί ότι μπορούν να προβλέψουν χρονοσειρές. Στα στατιστικά μοντέλα υλοποιήσαμε δύο τρόπους πρόβλεψης. Μία που εκπαιδεύει τα μοντέλα σε συγκεκριμένα ιστορικά στοιχεία και προβλέπει τις K επόμενες στιγμές και μία που λαμβάνει υπόψη μόνο τις τελευταίες N μετρήσεις και υπολογίζει την αμέσως επόμενη. Τα νευρωνικά δίκτυα με την σειρά τους υλοποιήθηκαν όλα με τον ίδιο τρόπο. Τα μοντέλα λαμβάνουν σαν είσοδο τις τελευταίες K=24 τιμές της χρονοσειράς και προβλέπουν την αμέσως επόμενη τιμή του φόρτου εργασίας. Στο επόμενο βήμα, αξιολογήσαμε αυτά τα μοντέλα με βάση κάποιες μετρικές (MSE, RMSE, MAE) που αναφέραμε και στο θεωρητικό υπόβαθρο (Κεφάλαιο 2.3.4) και επιλέξαμε το πιο ακριβές μοντέλο πρόβλεψης που θα χρησιμοποιήσουμε και στο τελικό μας σύστημα.

Η επόμενη φάση των προσομοιώσεων μας βασίζεται στην υλοποίηση βαθιάς ενισχυτικής μάθησης για την δυναμική διαχείριση υπολογιστικών πόρων. Δεδομένου ότι είναι ήδη αποδεδειγμένο ότι ένα σύστημα με βαθιά ενισχυτική μάθηση μπορεί να λειτουργήσει αποτελεσματικά με μεγάλο πλήθος παραμέτρων περιβάλλοντος (από το DERP[40]), αποφασίσαμε στις προσομοιώσεις μας να κρατήσουμε μόνο τις μεταβλητές του φόρτου εργασίας και του αριθμού των ενεργών εικονικών μηχανών (VMs). Πειραματιστήκαμε με διαφορετικές μορφές φόρτου εργασίας (μία με μια απλή ημιτονοειδής κυματομορφή και με ένα μέρος της συνθετικής χρονοσειράς μας) και δύο διαφορετικής μορφής συναρτήσεις επιβράβευσης. Αποδείξαμε ότι το σύστημα μας λειτουργεί αποδοτικά σε όλες τις παραπάνω περιπτώσεις και είναι στην

ευχέρεια του χρήστη να αποφασίσει ποια συνάρτηση επιβράβευσης θα χρησιμοποιήσει ανάλογα με το ποια θεωρεί οτι καλύπτει καλύτερα τις απαιτήσεις του.

Στο τέλος, υλοποιήσαμε το προτεινόμενο μας σύστημα, το οποίο συνδυάζει τις δύο παραπάνω τεχνικές. Το σύστημα μας βασίζεται και αυτό στην βαθιά ενισχυτική μάθηση όμως διαφοροποιείται από αυτήν γιατί για να λάβει μία απόφαση δεν του αρκεί μόνο η μέτρηση της τιμής του φόρτου εργασίας και των ενεργών εικονικών μηχανών αλλά και η πρόβλεψη της επόμενης κατάστασης που λαμβάνεται από το μοντέλο πρόβλεψης μας. Με αυτόν τον τρόπο, θέλουμε να δούμε αν το σύστημα μας μπορεί να προλάβει κάποια αυξομείωση του φόρτου εργασίας που δεν μπορεί προβλέψει η απλή βαθιά ενισχυτική μάθηση και να προσαρμοστεί γρηγορότερα στις αλλαγές. Για να αξιολογήσουμε την απόδοση του συγκρίναμε το σύστημα που λειτουργεί μόνο με την βαθιά μηχανική μάθηση με το προτεινόμενο υβριδικό σύστημα μας (με τις ίδιες υπερπαραμέτρους και συναρτήση επιβράβευσης) και αναλύσαμε τις μικρές, αλλά σημαντικές διαφοροποιήσεις τους.

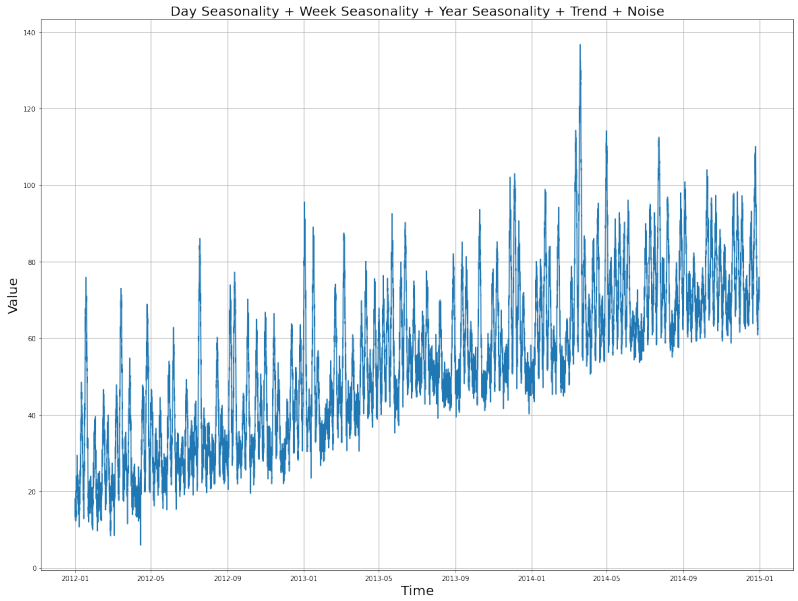
5.2 Μορφή χρονοσειράς φόρτου εργασίας

Οι χρονοσειρές που χρησιμοποιούμε είναι ένα ημίτονο για την απλή μορφή και μία πιο περίπλοκη μορφή η οποία παρουσιάζει πολλαπλές μορφές εποχιακότητας, μία γραμμική ανοδική τάση καθώς επίσης προστέθηκε και λευκός θόρυβος.Αναλυτικότερα, το συγκεκριμένο συνθετικό dataset απεικονίζει τις μετρήσεις του φόρτου εργασίας ανά ώρα. Ακολουθεί μια εποχιακότητα τόσο ως προς τις ημέρες, τις εβδομάδες και τα χρόνια. Θεωρήσαμε δηλαδή ότι:

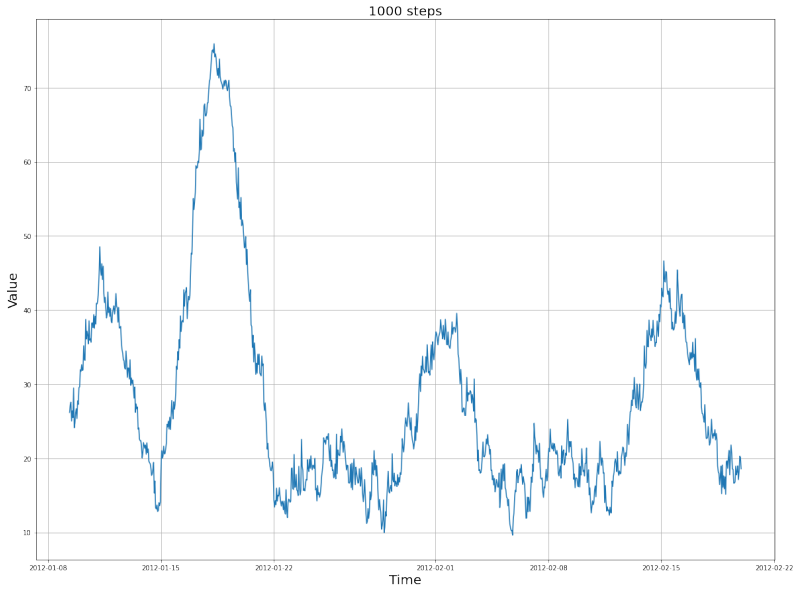
- Κάθε μέρα υπάρχει ένα ημιτονοειδές μοτίβο με σταθερή περίοδο (24 ώρες) και τυχαίου πλάτους κορυφής, το οποίο είναι διαφορετικό για [0,12] και για [12,24] ώρες.
- Κάθε εβδομάδα υπάρχει ένα γκαουσιανό μοτίβο με σταθερή περίοδο (7 ημέρες) και τυχαίο ύψος, διαφορετικό για κάθε εβδομάδα.
- Κάθε χρόνο υπάρχει ενα γκαουσιανό μοτίβο με σταθερή περίοδο (1 χρόνου) και τυχαίου υψους, διαφορετικό για κάθε χρόνο.
- Μία γραμμική ανοδική τάση καθ' όλη την διάρκεια.
- Έναν λευκό θόρυβο.

Η συνολική χρονοσειρά, λοιπόν, προέκυψε από την πρόσθεση όλων των παραπάνω χαρακτηριστικών. Το συνολικό διάγραμμα για τρία χρόνια διαμορφώνεται όπως φαίνεται και στην Εικόνα 5.1. Τέλος, για να απεικονιστεί καλύτερα η χρονοσειρά παρουσιάζουμε στην Εικόνα 5.2 ένα δείγμα 1000 ωρών.

Είναι ξεκάθαρο ότι η χρονοσειρά αυτή δεν είναι εύκολο να αναλυθεί και να προβλεφθεί με ευκολία. Ο πρώτος μας στόχος, επομένως, είναι να βρούμε ένα αποδοτικό μοντέλο που να μπορεί να προβλέψει αυτού του είδους τις χρονοσειρές.



Εικόνα 5.1: *Τελικό διάγραμμα χρονοσειράς φόρτου εργασίας*



Εικόνα 5.2: *Εστιασμένο διάγραμμα χρονοσειράς σε περίοδο 1000 ωρών*

5.3 Αποτελέσματα Πρόβλεψης χρονοσειρών

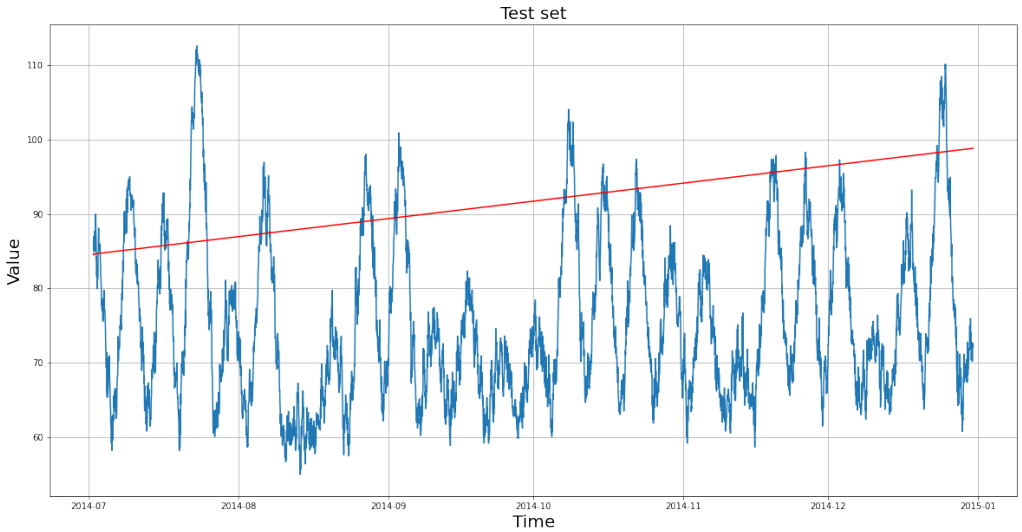
Για την πρόβλεψη της χρονοσειρά χρησιμοποιήσαμε τα μοντέλα που αναφέραμε στην προηγούμενη ενότητα. Σε αυτήν την ενότητα θα παρουσιαστούν αναλυτικά όλα τα χαρακτηριστικά των μοντέλων και τα αποτελέσματά τους.

Η χρονοσειρά αποτελείται από τρία έτη. Την χωρίσαμε σε δύο κομμάτια, το train που αποτελεί τα πρώτα 2,5 χρόνια και το test που αποτελεί το άλλον μισό χρόνο. Τα μοντέλα μας θα προσπαθήσουν να προβλέψουν το test set παίρνοντας πληροφορίες από το training set.

5.3.1 ARIMA

Δεδομένου ότι η χρονοσειρά που μελετάμε στην ουσία δεν έχει κάποιο σταθερό seasonality μιας και το πλάτος κάθε μοτίβου είναι τυχαίο και δεν είναι στάσιμη, αφού παρουσιάζει και τάση, τα μοντέλα ARIMA είναι δύσκολο να μπορέσουν να την αναλύσουν. Γενικά, τα μοντέλα ARIMA μπορούν να προβλέψουν το μέλλον μιας χρονοσειράς μακροπρόθεσμα. Αυτό όμως γίνεται όταν η χρονοσειρά πληροί τις προυποθέσεις των μοντέλων ώστε να μπορεί να αναλυθεί σε τρεις συνιστώσεις που αναφέραμε στο θεωρητικό μέρος (τάση, εποχικότητα και ανωμαλίες).

Αρχικά, δοκιμάστηκαν τα μοντέλα ARIMA - Box-Jenkins τα οποία εκπαιδεύτηκαν με το training set και προέβλεψαν τον επόμενο μισό χρόνο. Οι υπερπαράμετροι που δοκιμάσαμε ήταν οι συνδυασμοί p=[1,2,4,12,24], d=[1], q=[0,1,2]. Το αποτελέσμα του καλύτερου μοντέλου φαίνεται στην Εικόνα 5.3 καθώς και οι μετρικές αξιολόγησης των δύο από τα μοντέλα που δοκιμάστηκαν.



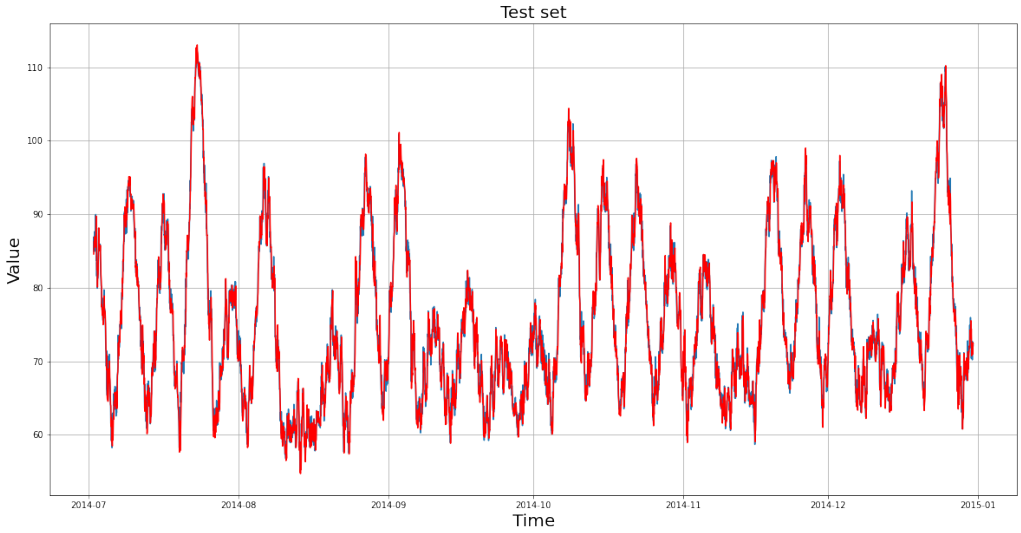
Εικόνα 5.3: *Fit ARIMA (1,1,0) one time Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη*

	MSE	RMSE	MAE
ARIMA (1,1,0)	398.162	19.954	17.705
ARIMA (24,1,0)	793.098	28.162	25.937

Τα αποτελέσματα των μοντέλων δεν είναι ενθαρρυντικά, αλλά το περιμέναμε για αυτού του είδους την χρονοσειρά που περιέχει πολύ μεγάλο θόρυβο και δυσδιάκριτη περιοδικότητα.

Μια άλλη τεχνική που έχει αποδειχτεί ότι βγάζει πολύ καλά αποτελέσματα είναι το rolling forecasting. Σε αυτήν την τεχνική το μοντέλο εκπαιδεύεται συνεχώς λαμβάνοντας υπόψιν την αμέσως προηγούμενη πληροφορία. Με αυτήν την τεχνική προβλέπουμε το άμεσο μέλλον με βάση το άμεσο παρελθόν. Πειραματιστήκαμε με μέγεθος buffer = [1 ημέρα, 2ημέρες, 1 εβδομάδα, 2 εβδομάδες]. Στην περίπτωση μας καλύτερα αποτελέσματα πήραμε κρατώντας τις τελευταίες 2 βδομάδες σε έναν buffer και δοκιμάσαμε τα μοντέλα που αναφέραμε και

παραπάνω, τα οποία είχαν πολύ καλές επιδόσεις. Οι μετρικές για τα τρία καλύτερα μοντέλα φαίνονται στην συνέχεια, καθώς και η Εικόνα 5.4 δείχνει το αποτέλεσμα του καλύτερου μοντέλου.



Εικόνα 5.4: *ARIMA (1,1,0) rolling forecasting Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη*

	MSE	RMSE	MAE
ARIMA (1,1,0)	3.175	1.782	1.395
ARIMA (2,1,0)	3.411	1.847	1.451
ARIMA (4,1,0)	3.482	1.866	1.455

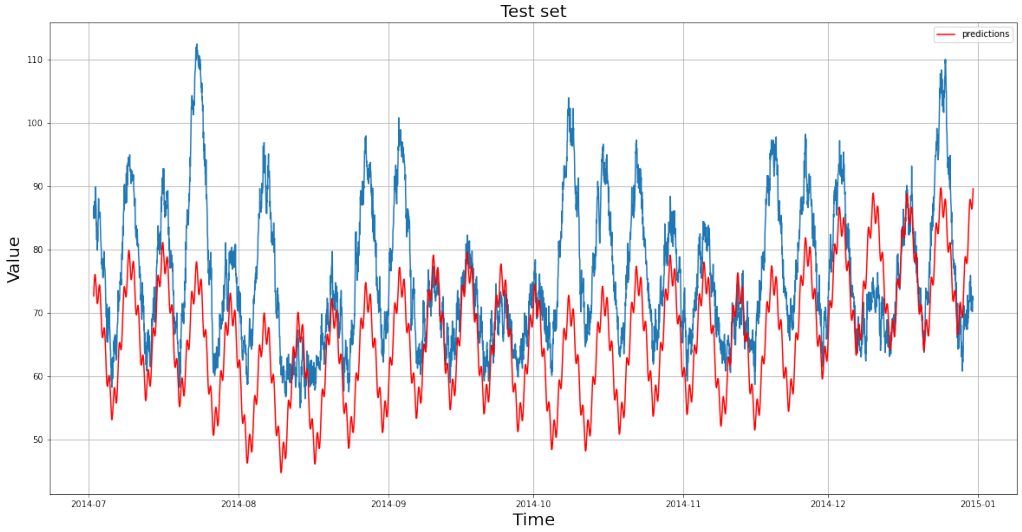
Φαίνεται ότι η τεχνική αυτή είναι πολύ αποδοτική και έχει προταθεί και για το workload prediction[7].

5.3.2 Prophet

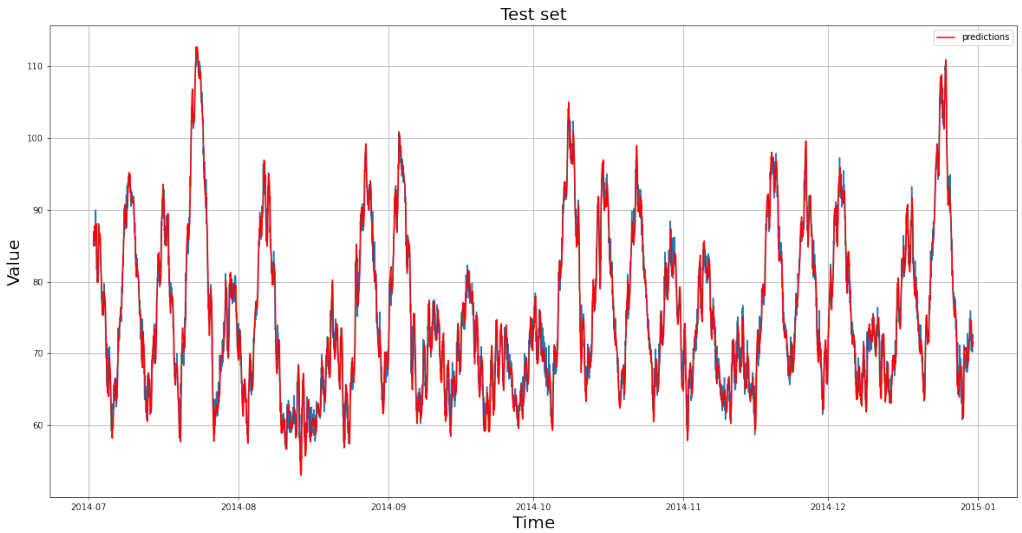
Αντίστοιχα με τα ARIMA μοντέλα δοκιμάσαμε τις δύο διαφορετικές τεχνικές και για το Prophet.

Με την πρώτη τεχνική, δηλαδή εκπαιδεύοντας το μοντέλο με τα δεδομένα των 2,5 χρόνων και κάνοντας forecast τον επόμενο μισό, όπως φαίνεται και στην Εικόνα 5.5 προσπαθεί να πλησιάσει την πραγματική χρονοσειρά αλλά δεν μπορεί να καταλάβει τις αλλαγές των κορυφών γι' αυτό και έχει ένα σταθερό πλάτος, πιθανώς κοντά στην μέση τιμή των τυχαίων κορυφών. Μπορούμε να συμπεράνουμε ότι τα αποτελέσματα μπορεί να μην είναι τόσο αποτελεσματικά αλλά πλησιάζουν τις πραγματικές τιμές σε αντίθεση με τα μοντέλα ARIMA που τραβούσαν απλά μια ευθεία γραμμή.

Εφαρμόσαμε, στην συνέχεια, την τεχνική του rolling forecasting στο prophet έχοντας αποθηκεύσει στον buffer τις τιμές του προηγούμενου μήνα. Τα αποτελέσματα φαίνονται στην Εικόνα 5.6.



Εικόνα 5.5: *Fit Prophet one time Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη*



Εικόνα 5.6: *Prophet rolling forecasting Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη*

Prophet	MSE	RMSE	MAE
one time fitting	156.925	12.527	10.347
rolling forecasting	4.368	2.09	1.645

Στην συνέχεια, δοκιμάσαμε διάφορα μοντέλα τεχνητών νευρωνικών δικτύων. Η εκπαίδευση των τεχνητών νευρωνικών δικτύων είναι συγκεκριμένη. Εκπαιδεύονται στο training set σε ένα σύνολο από εποχές και εξετάζεται η αποτελεσματικότητα τους στο test set. Αυτός ο διαχωρισμός γίνεται γιατί τα νευρωνικά δίκτυα όταν εκπαιδεύονται σε συγκεκριμένα δεδομένα για πολλές εποχές τείνουν να υπερπροσαρμόζονται σε αυτά και να μην μπορούν να βγάλουν σωστά αποτέλεσματα σε δεδομένα που δεν έχουν ξαναντικρίσει. Επιπλέον, κατά την περίοδο της εκπαίδευσης, χωρίσαμε το train set σε train-validation με ποσοστό 80%-20%, έτσι ώστε σε κάθε εποχή να παρακολουθούμε το επίπεδο της υπερεκπαίδευσης. Εαν το train loss μειώνεται και το validation loss αυξάνεται καταλαβαίνουμε ξεκάθαρα ότι το μοντέλο μας αρχίζει να υπερεκπαιδεύεται.

5.3.3 LSTM

Το μοντέλο LSTM που βρέθηκε να έχει καλύτερες επιδόσεις αποτελείται από τα εξής στοιχεία:

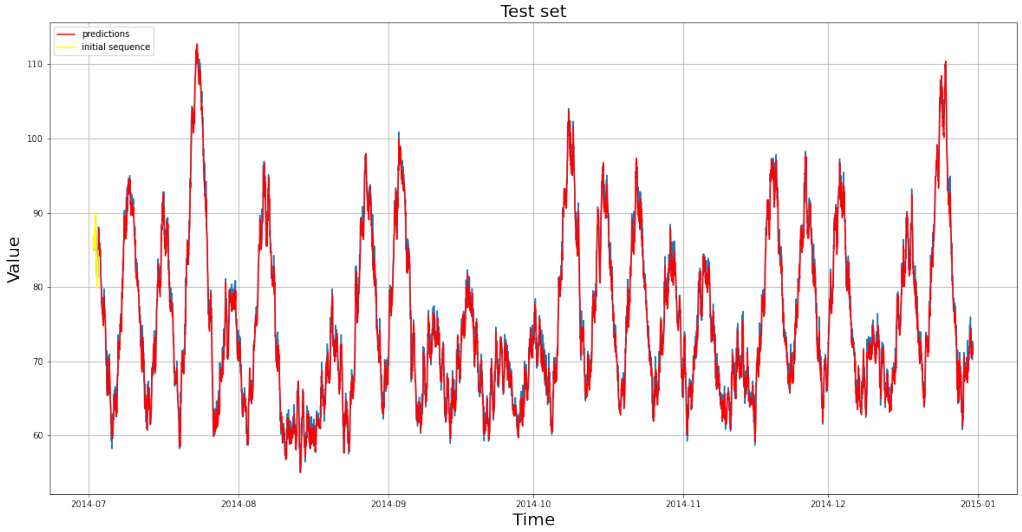
- LSTM layer με 24 units, relu activation function και μέγεθος εισόδου = (24,1)
- Dense layer με 1 unit και relu activation function

Πιο συγκεκριμένα, παίρνει σαν είσοδο τις προηγούμενες 24 τιμές και επιστρέφει την πρόβλεψη του για την επόμενη τιμή του workload. Η εκπαίδευση έγινε με τις υπερπαραμέτρους:

- batch size 128
- epochs 25
- optimizer Adam
- loss function MSE

Τα αποτελέσματα του test set φαίνονται στην Εικόνα 5.7 και οι μετρικές στο παρακάτω πινακάκι.

	MSE	RMSE	MAE
LSTM	2.907	1.705	1.335



Εικόνα 5.7: *Test LSTM Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη, Κίτρινο: τα πρώτα 24 σημεία*

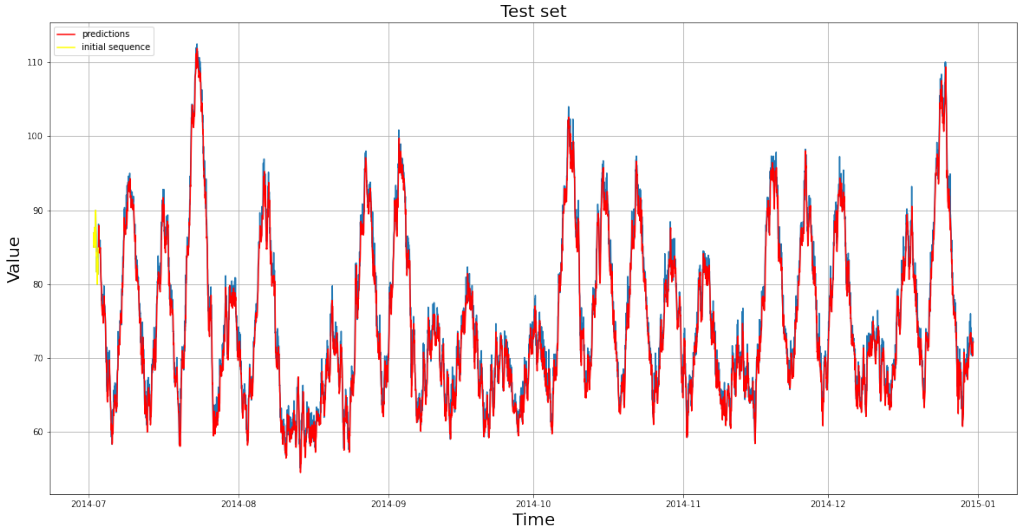
5.3.4 GRU

Αντίστοιχα με τα LSTM, έχουμε το μοντέλο GRU:

- GRU layer με 24 units, relu activation function και μέγεθος εισόδου = (24,1)
- Dense layer με 1 unit και relu activation function
- batch size 128
- epochs 20
- optimizer Adam
- loss function MSE

Το GRU μοντέλο εκπαιδεύεται σε λιγότερες εποχές από το LSTM επειδή παρατηρήθηκε αύξηση στο validation loss το οποίο μας οδηγεί στο συμπέρασμα ότι το μοντέλο αρχίζει να υπερεκπαιδεύεται. Αυτό ήταν αναμενόμενο γιατί όπως αναφέραμε και στο θεωρητικό υπόβαθρο το GRU περιέχει λιγότερα βάρη σε σχέση με το LSTM. Τα αποτελέσματα του φαινονται στην Εικόνα 5.8.

	MSE	RMSE	MAE
GRU	3.179	1.783	1.380



Εικόνα 5.8: *Test GRU Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη, Κίτρινο: τα πρώτα 24 σημεία*

5.3.5 CNN

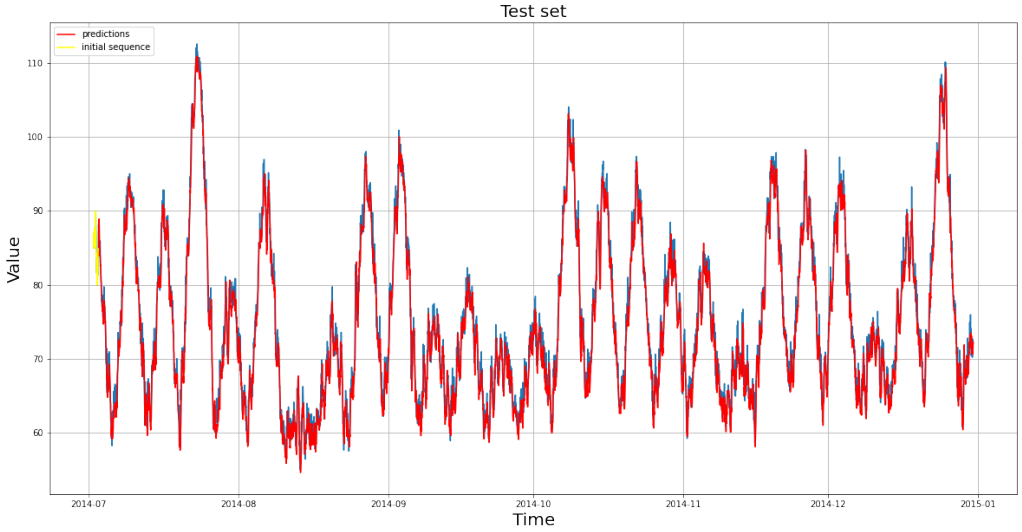
Το CNN που ήταν πιο αποδοτικό στην χρονοσειρά μας αποτελείται από τα παρακάτω επίπεδα:

- Conv1D filters:64, kernel size:2, strides:1, activation:relu
- MaxPooling1D pool size:2
- Flatten
- Dense units:50, activation:relu
- Dense units:1, activation:relu

Και τις υπερπαραμέτρους:

- batch size 64
- epochs 80
- optimizer Adam
- loss function MSE

	MSE	RMSE	MAE
CNN	3.659	1.913	1.483



Εικόνα 5.9: *Test CNN Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη, Κίτρινο: τα πρώτα 24 σημεία*

5.3.6 Autoencoder-LSTM

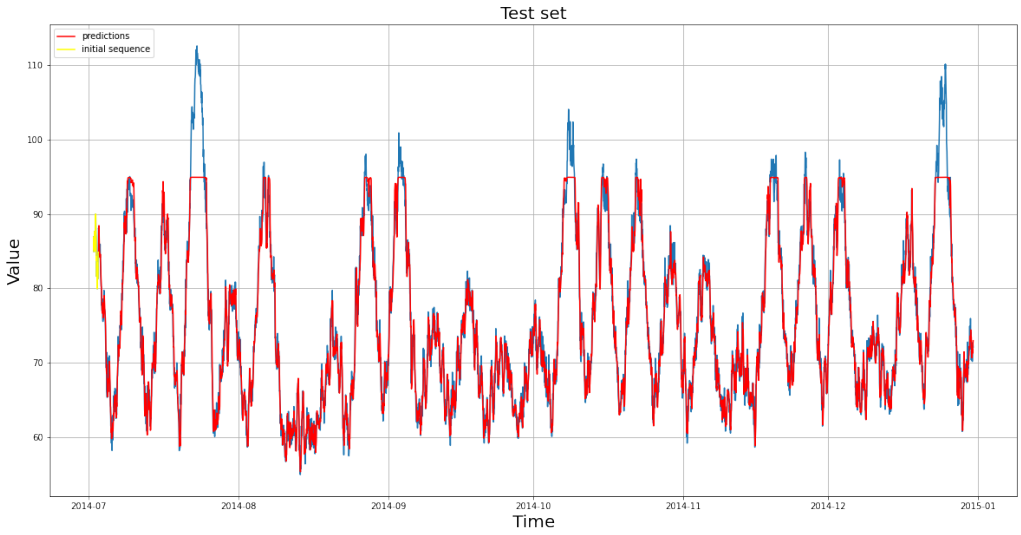
Για τον Autoencoder-LSTM χρησιμοποιήθηκε ένας Autoencoder (που επιλέξαμε να αποτελείται απο 3 LSTM για κωδικοποίηση και 3 για αποκωδικοποίηση) και στην συνέχεια, ένα επιπλέον LSTM για να κάνει την πρόβλεψη μαζί με ένα Dense layer για την έξοδο. Τα units του autoencoder δοκιμάστηκαν σε διαφορετικές τιμές ([126,64,32],[96,48,24],[48,24,12],[24,12,6] για κάθε ένα LSTM του κωδικοποιητή και αποκωδικοποιητή αντίστοιχα). Η περίπτωση που παρουσίασε τα καλύτερα αποτελέσματα αποτελείται από τα παρακάτω επίπεδα:

- LSTM layer units=48, και μέγεθος εισόδου = (24,1) Encoder1
- LSTM layer units=24 Encoder2
- LSTM layer units=12 Encoder3
- RepeatVector n=24 Bottleneck layer
- LSTM layer units=12 Decoder1
- LSTM layer units=24 Decoder2
- LSTM layer units=48 Decoder3
- LSTM layer units=28 LSTM predictor
- Dense layer με 1 unit και relu activation function

Οι υπερπαράμετροι που εκπαιδεύτηκε το σύστημα είναι οι εξής:

- batch size 128
- epochs 150

- optimizer Adam
- loss function MSE

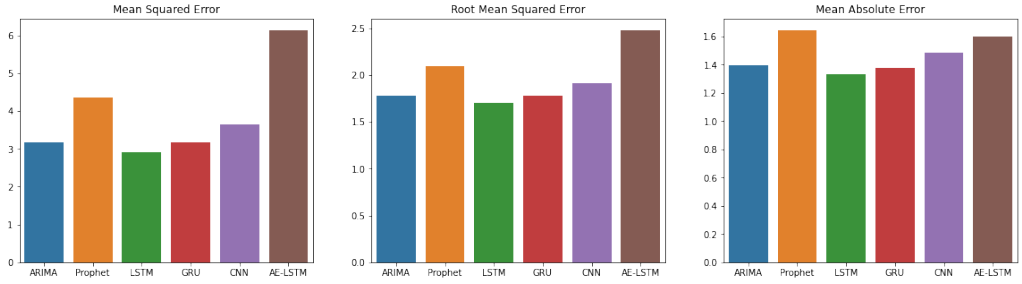


Εικόνα 5.10: *Test Autoencoder-LSTM Μπλε: η πραγματική χρονοσειρά, Κόκκινο: η πρόβλεψη, Κίτρινο: τα πρώτα 24 σημεία*

	MSE	RMSE	MAE
AE-LSTM	6.14	2.478	1.597

5.3.7 Σύγκριση μεθόδων

Αν εξαιρέσουμε τις αρχικές μετρήσεις των στατιστικών μοντέλων που προβλέπουν μακροπρόθεσμα, οι οποίες απέτυχαν, όλες οι άλλες μέθοδοι έχουν ιδιαίτερα καλά αποτελέσματα προβλέποντας με μεγάλη ακρίβεια την επόμενη τιμή της χρονοσειράς. Τα συγκεντρωτικά αποτελέσματα τους βρίσκονται στην Εικόνα5.11.



Εικόνα 5.11: *Σύγκριση μεθόδων με mse, rmse, mae*

Στην δική μας χρονοσειρά, που είναι ιδιαίτερα περίπλοκη, καλύτερη επίδοση έχει το LSTM το οποίο και θα χρησιμοποιήσουμε και στο προτεινόμενο σύστημα μας. Τα επόμενα μοντέλα που έχουν πολύ καλά αποτελέσματα είναι τα ARIMA, GRU με μικρή διαφορά και μετέπειτα το CNN, Prophet, AutoEncoder-LSTM. Αυτό που παρατηρούμε είναι ότι ο AutoEncoder-LSTM δεν μπορεί να προσαρμοστεί στην τάση της χρονοσειράς. Παρουσιάζει

ένα ανώτατο όριο πρόβλεψης το οποίο λόγω της τάσης του test set το ξεπερνάει και δημιουργούνται σφάλματα. Γι΄ αυτό το λόγο φαίνεται να έχει χειρότερα αποτέλεσματα. Τα μοντέλα ARIMA, LSTM είναι τα πιο δημοφιλή στην πρόβλεψη χρονοσειρών και αποδείξανε και σε αυτήν την περίπτωση περίπλοκης χρονοσειράς ότι λειτουργούν εξαιρετικά.

Προτείνεται, ωστόσο, να δοκιμαστούν όλοι οι παραπάνω μέθοδοι για την εκάστοτε χρονοσειρά του cloud περιβάλλοντος που θέλουμε να εφαρμόσουμε το προτεινόμενο σύστημα. Αφού υλοποιηθούν όλες οι μέθοδοι πρόβλεψης, να επιλεχτεί ο predictor με τα καλύτερα αποτελέσματα. Σίγουρα, κάθε χρονοσειρά έχει τα δικά της ιδιαίτερα χαρακτηριστικά και ανάλογα με αυτά θα αποδειχθεί κάποιο μοντέλο πιο ακριβές σε σχέση με τα υπόλοιπα. Τέλος, στην επιλογή του predictor πρέπει να λάβουμε υπόψιν μας τον χρόνο που κάνει κάθε μοντέλο για να προβλέψει την επόμενη τιμή. Στην περίπτωση των στατιστικών μοντέλων, γίνεται εκπαίδευση των μοντέλων on-time και αυτό πιθανόν να παίρνει περισσότερο χρόνο από την πρόβλεψη ενός νευρωνικού που έχει ήδη εκπαιδευτεί. Ο χρόνος πρόβλεψης εξαρτάται άμεσα με το πόσα ιστορικά δεδομένα χρειάζεται το κάθε μοντέλο για να λειτουργήσει αποδοτικά.

5.4 Αποτελέσματα Δυναμικής Διαχείρισης Πόρων με χρήση DQN

Το σύστημα που χρησιμοποιήσαμε προσομοιάζει σε μεγάλο βαθμό το DERP [40]. Στην δική μας υλοποίηση για την απλοποίηση του προβλήματος σε πρώτη φάση ασχοληθήκαμε μόνο με το workload και τον αριθμό των δεσμευμένων πόρων, ώστε να δούμε αν έχουμε καλύτερες αποδόσεις αρχικά με μία χρονομεταβαλλόμενη μεταβλητή και στην συνέχεια το πρόβλημα μπορεί να γενικευτεί και με όλες τις υπόλοιπες.

Το νευρωνικό που υπολογίζει τα Q-values είναι 3-layer fully connected με Dense layers. Το καθένα αποτελείται από 64, 128, 256 νευρώνες αντίστοιχα και από μια relu activation function. Διαλέξαμε να πειραματιστούμε με το ίδιο νευρωνικό του DERP [40] μιας και ήταν αποδεδειγμένο ότι είναι αποδοτικό, ώστε να συγκρίνουμε τα αποτελέσματα μας με το προτεινόμενο μας σύστημα. Είναι σημαντικό να σημειωθεί ότι ενθαρρύνουμε να χρησιμοποιηθούν και διαφορετικής μορφής νευρωνικά δίκτυα τα οποία πιθανώς να έχουν και καλύτερα αποτελέσματα από το δοκιμασμένο. Το νευρωνικό εκπαιδεύεται με την μέθοδο των mini batches χρησιμοποιώντας την τεχνική του experience replay και το ε-policy ώστε να αντιμετωπιστεί το δίλημμα exploration/exploitation.

Οι πιθανές ενέργειες (actions) του συστήματος είναι τρεις:

1. να προστεθεί ένα VM
2. να αφαιρεθεί ένα VM
3. να παραμείνει ο ίδιος αριθμός από VMs

Οι υπερπαράμετροι που χρησιμοποιήθηκαν για την εκπαίδευση του Deep Reinforcement Learning είναι οι παρακάτω:

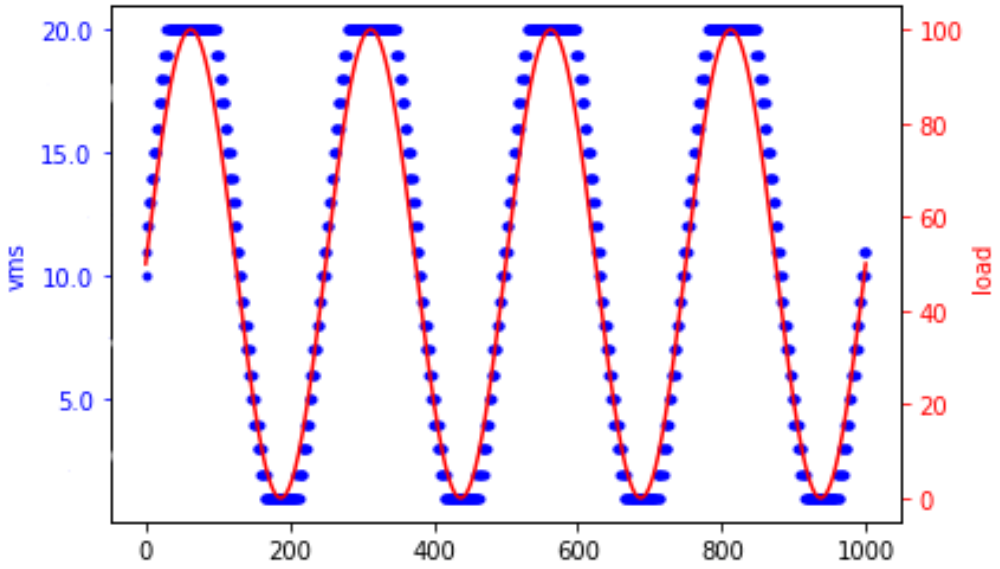
- Μέγεθος μνήμης 2000
- Mini-batch size 128
- Episodes 200
- Pre-training steps 1000
- Training steps 250 (για κάθε episode)
- Testing steps 1000
- Algorithm Simple DQN
- Learning rate 0.0001
- Discount factor 0.95
- Optimizer RMSprop
- Epsilon = [start : 1, end : 0.01, decay : 0.05]

Σημαντικό ρόλο για το σύστημα δυναμικής διαχείρισης έχει η συνάρτηση που υπολογίζει το βραβείο (reward). Ανάλογα με τον τρόπο που θέλουμε να διαχειριστούμε το περιβάλλον μας υλοποιούμε και την reward function. Σίγουρα σπουδαίος παράγοντας είναι η χωρητικότητα του περιβάλλοντος. Πρέπει να βρεθεί μία συνάρτηση επιβράβευσης που δίνει μεγαλύτερο βραβείο όταν εκμεταλλευται ολόκληρο τον διαθέσιμο χώρο και να αυξάνει ή να μειώνει τον αριθμό των εικονικών μηχανών ανάλογα με το φόρτο εργασίας.

Σε πρώτη φάση για να δούμε αν λειτουργεί σωστά το σύστημα μας δοκιμάσαμε τα παρακάτω χαρακτηριστικά που προσομοιάζουν το DERP (ίδιας μορφής συνάρτηση επιβράβευσης και φόρτος εργασίας) προσαρμοσμένα στα δικά μας δεδομένα.

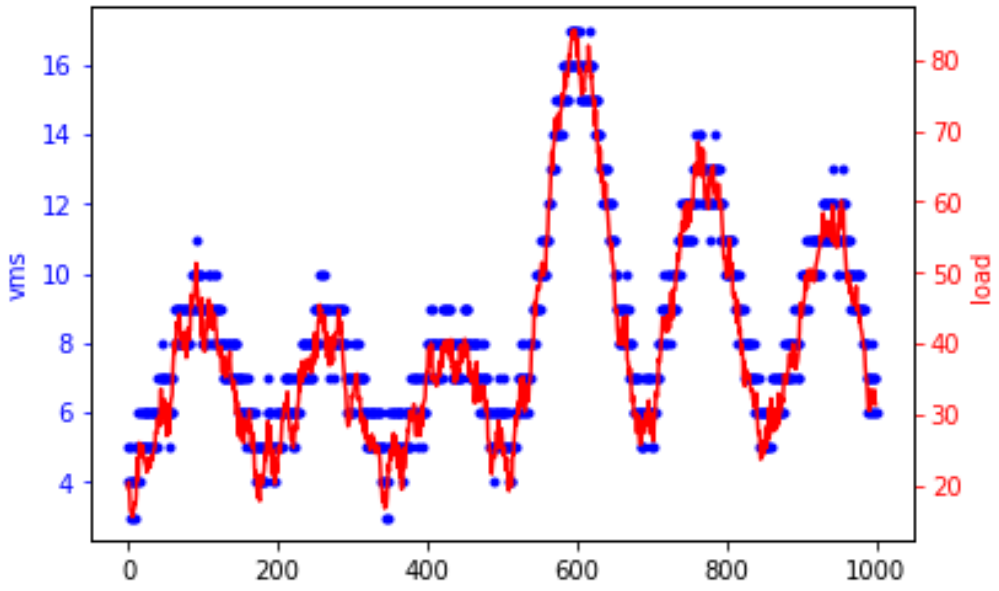
- $capacity(t) = 8 * vms(t)$
- $load(t) = 50 + 50sin(\frac{2\pi t}{250})$
- $r(t) = min(capacity(t+1), load(t+1)) - 3 * vms(t+1)$
- Διαθέσιμα VMs: 1-20

Το αποτέλεσμα φαίνεται στην Εικόνα 5.12.



Εικόνα 5.12: *Πείραμα για το RL component με load: σταθερό ημίτονο*

Στην συνέχεια, εκπαιδεύσαμε το σύστημα με ένα διάστημα 250 τιμών, με 200 επεισόδια της συνθετικής χρονοσειράς που αναφέραμε παραπάνω, το τεστάραμε σε ένα τυχαίο διάστημα 1000 τιμών και λάβαμε την παρακάτω Εικόνα 5.13.



Εικόνα 5.13: *Πείραμα για το RL component με load: synthetic dataset*

Τέλος, σκεφτήκαμε ότι για να εκμεταλλεύονται αποδοτικά τον διαθέσιμο χώρο των εικονικών μηχανών θα μπορούμε να χρησιμοποιήσουμε και άλλου είδους συνάρτηση επιβράβευσης. Η συνάρτηση επιβράβευσης που προτείνουμε έχει την μορφή γκαουσιανής συνάρτησης και ορίζεται ως:

$$r(t) = aexp(-\frac{(load(t+1) - capacity(t+1))^2}{2(\sigma(t))^2})$$

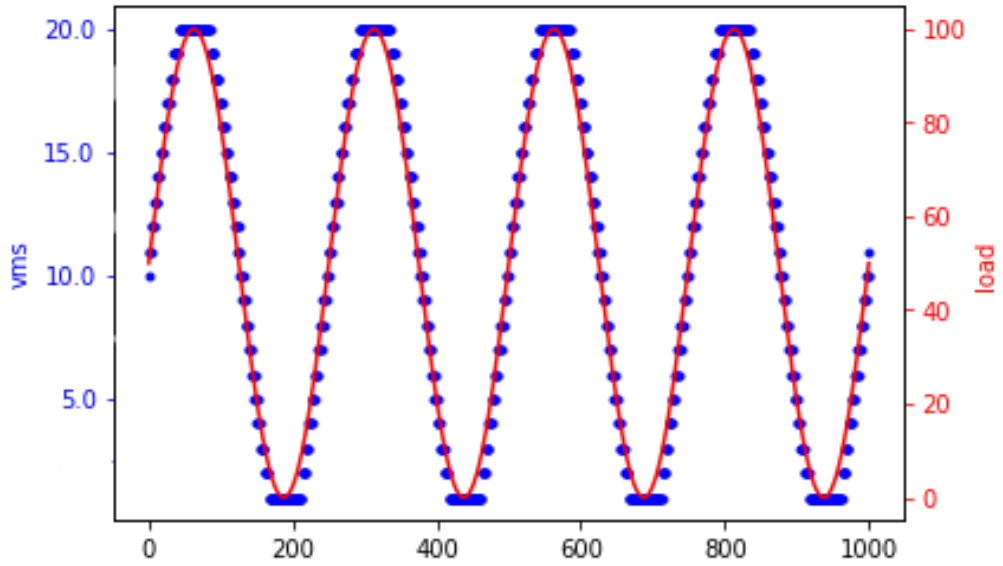
με:

- α: σταθερή τιμή που διαλέγει ο χρήστης ανάλογα με το πόση βαρύτητα θέλει να δώσει στην επιλογή του load να είναι κοντά στο capacity και
- σ: η τιμή που καθορίζει πόσο απόλυτος είναι ο χρήστης στην επιλογή του (μπορεί να είναι είτε μια σταθερή τιμή, είτε να εξαρτάται από το load, capacity).

Πιο συγκεκριμένα, αυτή η συνάρτηση μεγιστοποιεί την επιβράβευση όταν το load είναι κοντά με το capacity και την μειώνει ανάλογα με το πόσο του έχει υποδείξει ο χρήστης μέσω του σ. Όσο μικρότερο είναι το σ τόσο πιο απότομα μειώνεται η επιβράβευση. Μια τέτοια συνάρτηση μπορεί να χρησιμοποιηθεί από έναν πάροχο που θέλει να μεγιστοποιήσει την πιθανότητα το load να είναι κοντά με το capacity που έχει ορίσει.
Η προσομοίωση που υλοποιήσαμε χρησιμοποιούσε την r(t) που ορίσαμε παραπάνω με

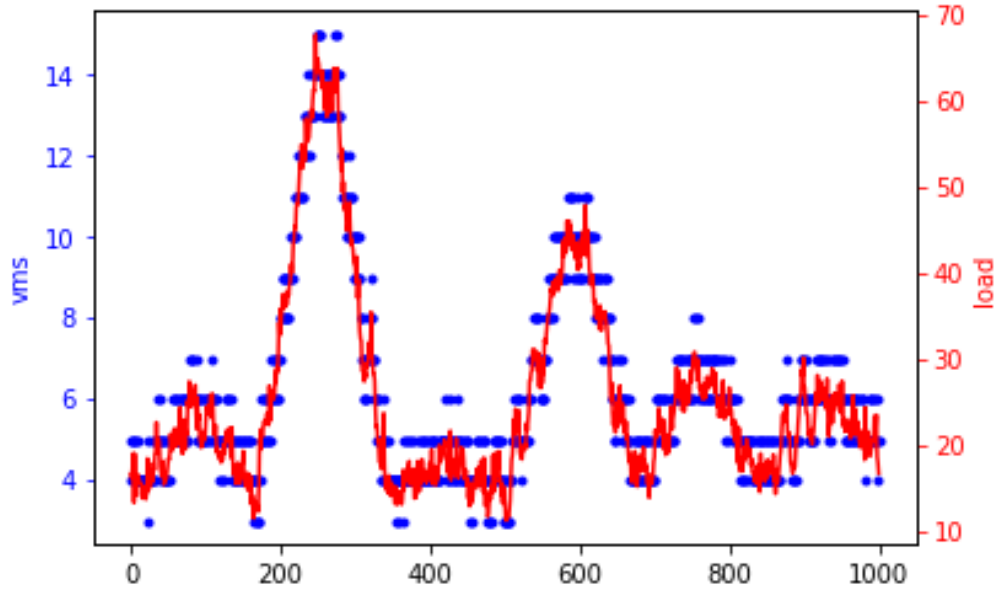
- $\sigma(t) = \frac{\beta}{load(t+1) - capacity(t+1)}$,
- $capacity(t) = 5vms(t)$,
- α=β=10 και
- $load(t) = 50 + 50sin(\frac{2\pi t}{250})$

Το αποτέλεσμα φαίνεται στην Εικόνα 5.14.



Εικόνα 5.14: *Πείραμα για το RL component με load: σταθερό ημίτονο και γκαουσιανή συνάρτηση επιβράβευσης*

Τέλος, μία προσομοίωση με τα ίδια στοιχεία με παραπάνω αλλά με load μέρος της συνθετικής χρονοσειράς Εικόνα 5.15.



Εικόνα 5.15: *Πείραμα για το RL component με load: synthetic dataset και γκαουσιανή συνάρτηση επιβράβευσης*

Με τα αποτελέσματα που φαίνονται στις Εικόνες 5.12, 5.13, 5.14, 5.15 μπορούμε να συμπεράνουμε ότι το σύστημα μας προσαρμόζεται αρκετά καλά τόσο σε σταθερές χρονοσειρές load όσο και σε πιο περίπλοκες και με τις δύο μορφές συναρτήσεων επιβράβευσης.

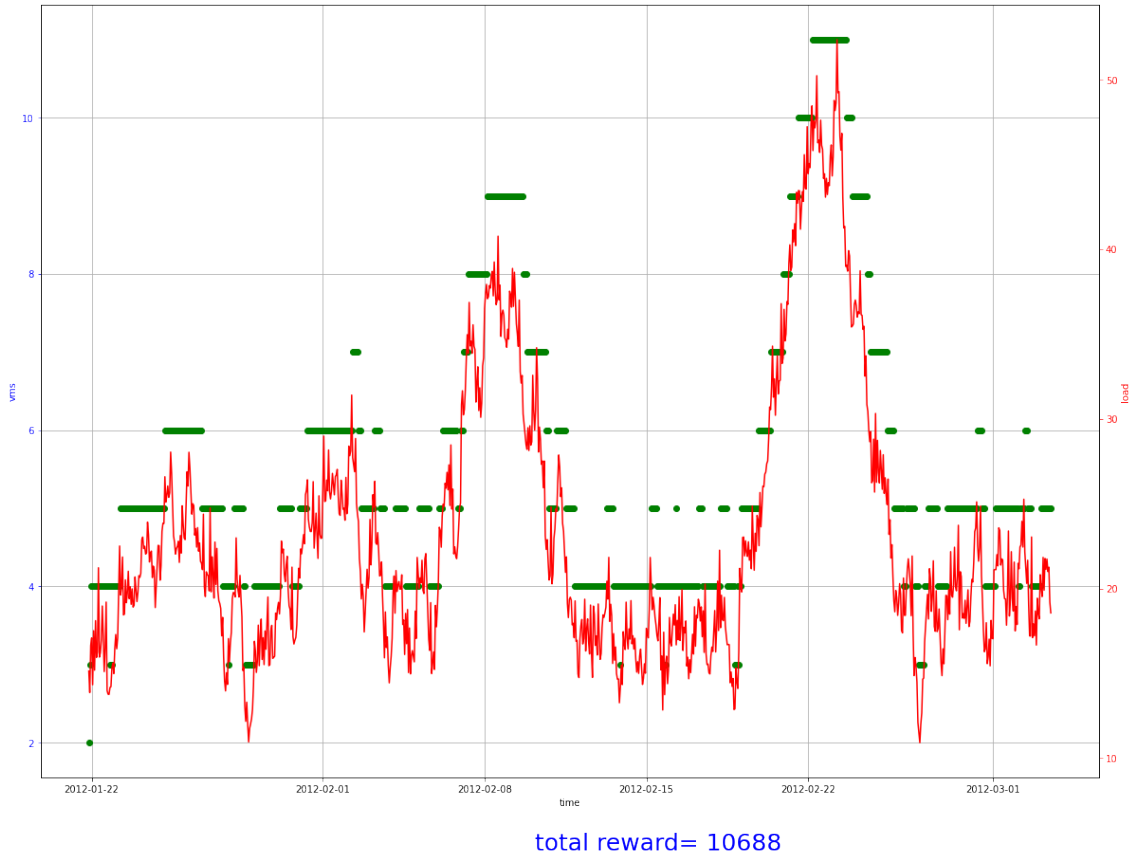
5.5 Αποτελέσματα Προτεινόμενου Συστήματος

Έπειτα, δεδομένης της καλής λειτουργίας του συστήματος αποφασίσαμε να το εξελίξουμε χρησιμοποιώντας το προτεινόμενο σύστημα μας με τον συνδυασμό των δύο παραπάνω μεθόδων μηχανικής μάθησης.

Το προτεινόμενο σύστημά μας, όπως προαναφέραμε, αποτελείται από το σύστημα δυναμικής διαχείρισης πόρων με χρήση ενισχυτικής μάθησης που δοκιμάσαμε παραπάνω προσθέτοντας στις μεταβλητές περιβάλλοντος και τις τιμές των μετρικών που (στο πείραμα μας την τιμή του load) που προβλέπει ο καλύτερος predictor (LSTM).

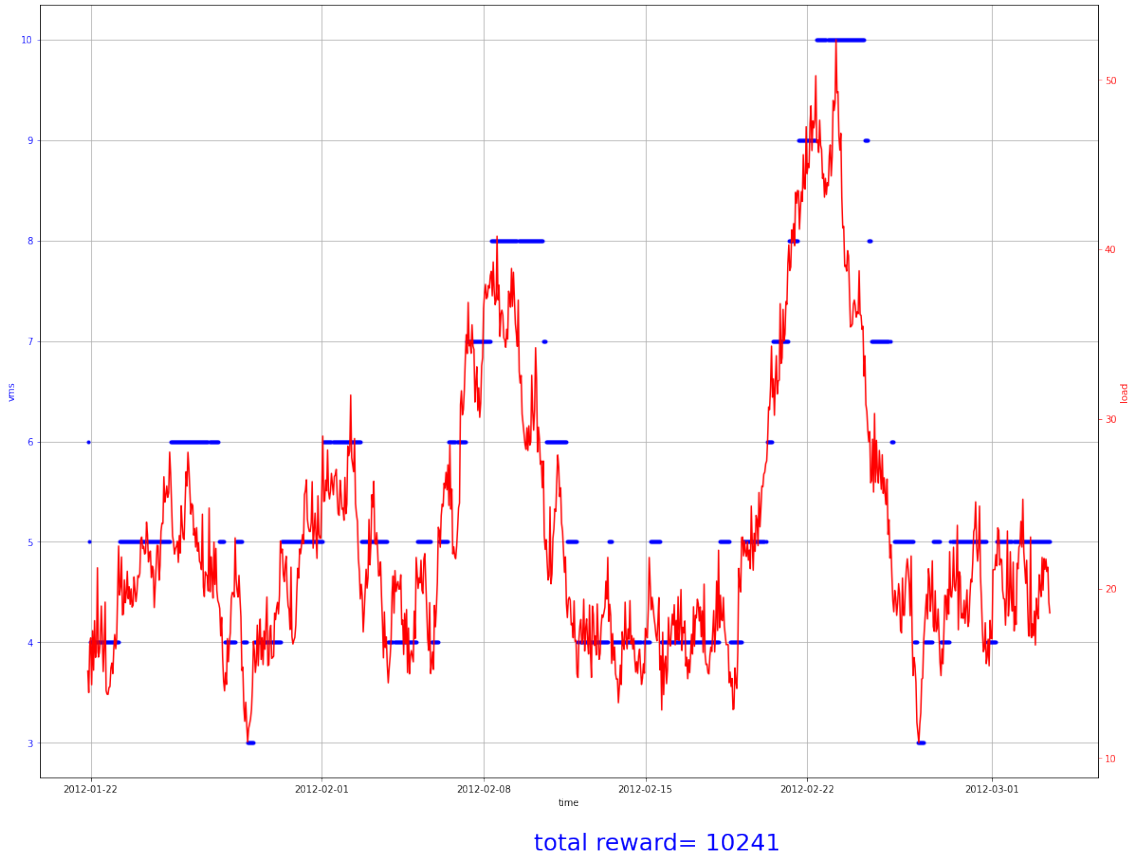
Δεδομένου ότι το load μας ακολουθεί την λογική της συνθετικής χρονοσειράς μας, εκπαιδεύσαμε εξωτερικά το LSTM μας με μια τέτοιου είδους χρονοσειρά. Μετά, το προσθέσαμε στο σύστημα μας, ώστε σε κάθε χρονική στιγμή όπου υπολογίζονται οι μετρικές του περιβάλλοντος μας να το καλούμε δίνοντας του σαν είσοδο τις τελευταίες 24 τιμές του load (συμπεραλαμβανομένου και της τωρινής τιμής) και να μας δίνει την πρόβλεψη του για την επόμενη κατάσταση και να την προσθέτουμε στο state μας.

Το συνολικό μας σύστημα το εκπαιδεύσαμε σε 250 training steps της συνθετικής χρονοσειράς μας, για 200 επεισόδια, με μνήμη μεγέθους 2000 και mini batch size 128. Η συνάρτηση reward που χρησιμοποιήσαμε ήταν η $r(t) = min(avms(t+1), load(t+1)) - \beta * vms(t+1)$, α=4, β=2 και είχαμε το αποτέλεσμα που φαίνεται στην Εικόνα 5.16.



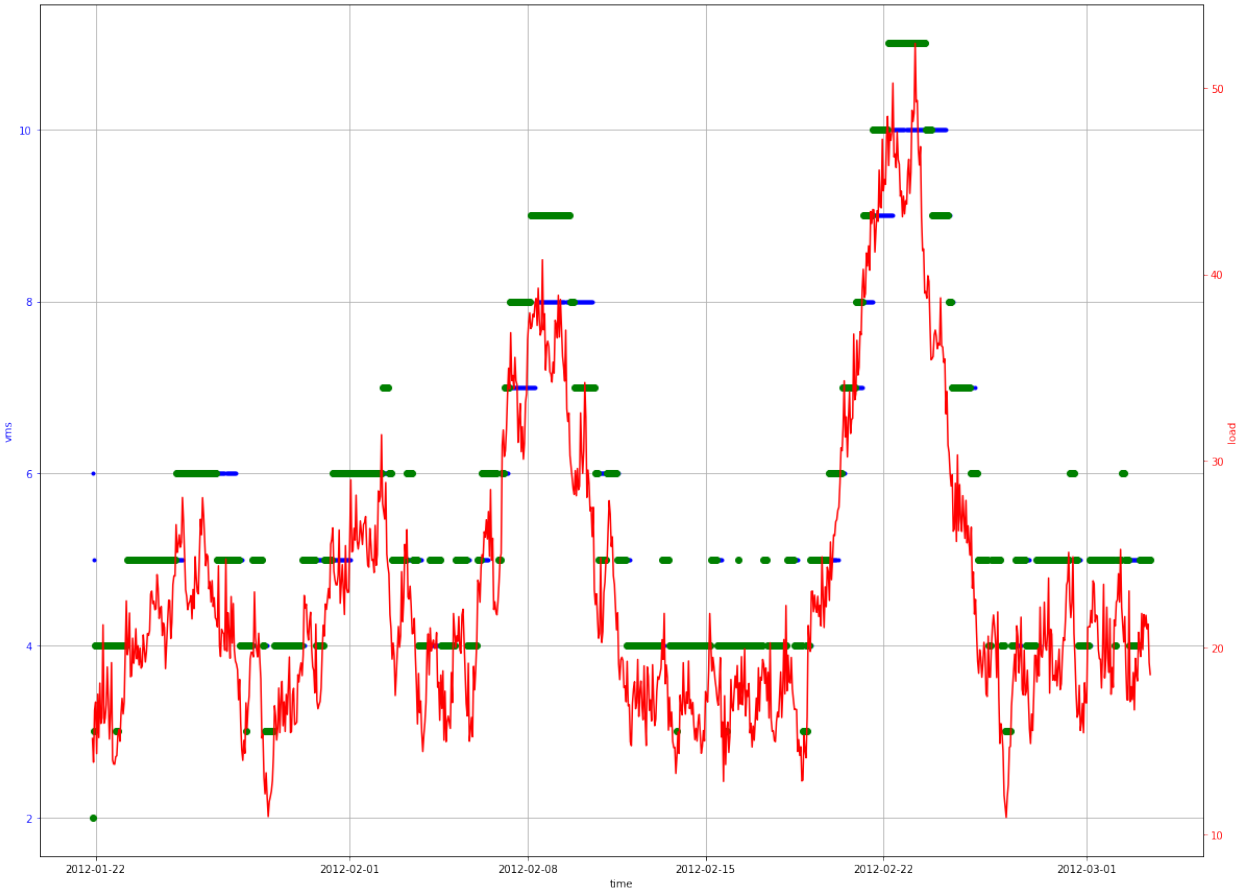
Εικόνα 5.16: *Πείραμα για το RL component + predictor με load: synthetic dataset*

Παρατηρούμε ότι το μοντέλο μας έχει και αυτό πολύ καλές επιδόσεις. Για να έχουμε μια ολοκληρομένη άποψη σχετικά με το αν το μοντέλο μας υπερτερεί σε σχέση με το προηγούμενο που δεν χρησιμοποιούσε τον predictor τρέξαμε με τα ίδια ακριβώς δεδομένα, τις ίδιες υπερπαραμέτρους και reward function και το προηγούμενο σύστημα. Το αποτέλεσμα φαίνεται στην Εικόνα 5.17



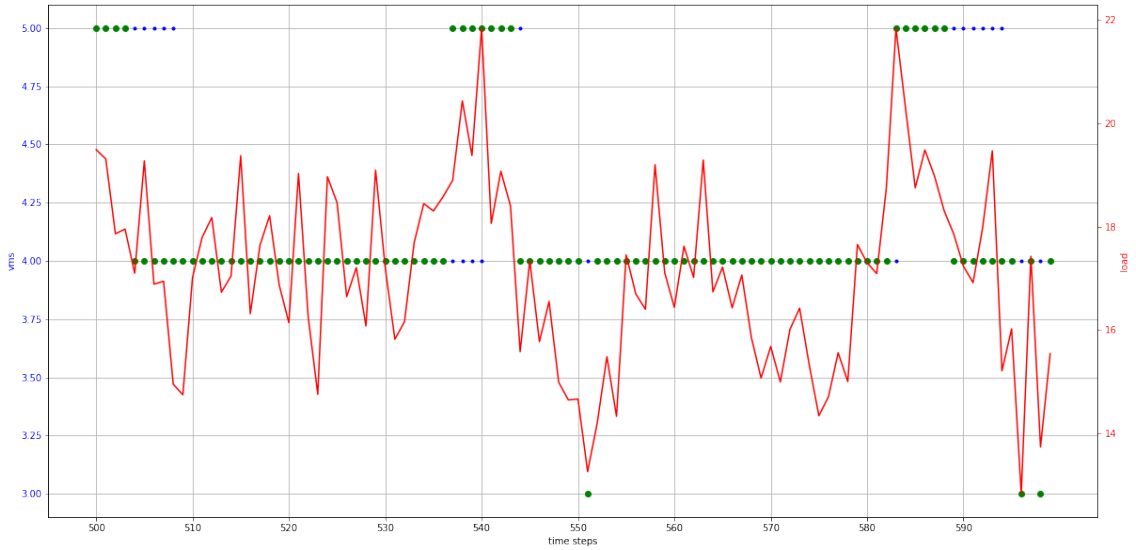
Εικόνα 5.17: *Πείραμα για το DERP με load: synthetic dataset*

Το πρώτο πράγμα που παρατηρούμε είναι ότι η συνολική ανταμοιβή στο σύστημα με την πρόβλεψη χρονοσειράς είναι μεγαλύτερη από το σύστημα που χρησιμοποιεί μόνο ενισχυτική μάθηση. Αυτό μας δείχνει ότι το προτεινόμενο σύστημα επέλεξε πιο αποδοτικά τον αριθμό των υπολογιστικών πόρων που θα έπρεπε να χρησιμοποιηθούν. Σε κοινό διάγραμμα τα αποτελέσματα μας είναι στην Εικόνα 5.18.

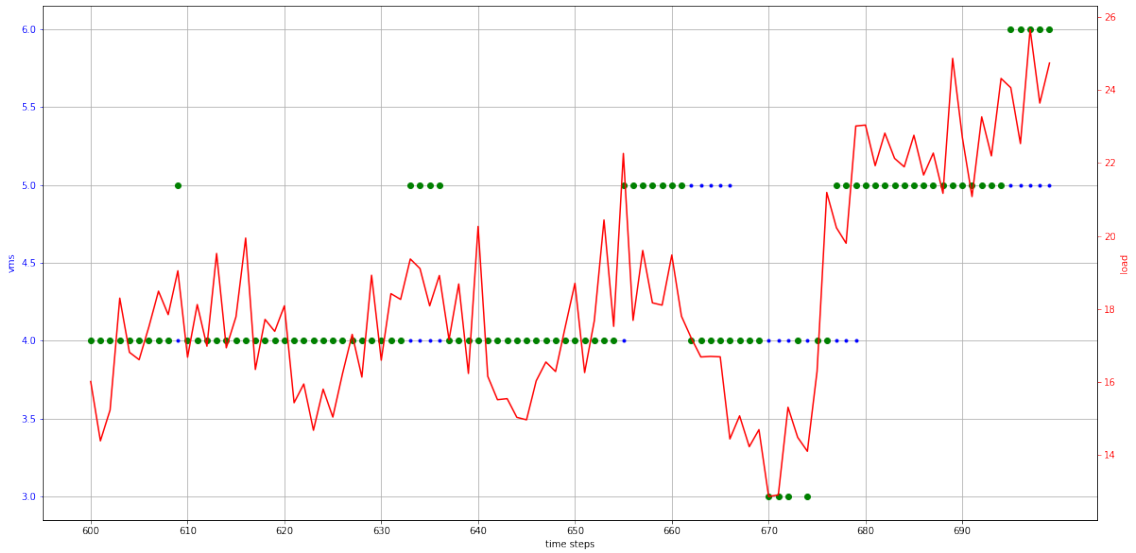


Εικόνα 5.18: *Συνδυασμός δύο μεθόδων. Μπλε: DERP και Πράσινο: Προτεινόμενο σύστημα*

Με πρώτη ματιά, παρατηρούμε ότι υπάρχουν αρκετές ομοιότητες στα δύο αποτελέσματα, όμως είναι εμφανές ότι το σύστημα με την πρόβλεψη έχει πιο γρήγορη ανταπόκριση στις αλλαγές του φόρτου εργασίας. Ας δούμε όμως λεπτομερώς τις διαφορές των δύο συστημάτων. Στις Εικόνες 5.19 (500-600 test steps), 5.20 (600-700 test steps) φαίνεται πιο καθαρά η διαφορά στην λειτουργία των δύο συστημάτων.



Εικόνα 5.19: *Εστίαση στα 500-600 test steps. Μπλε: DERP και Πράσινο: Προτεινόμενο σύστημα*



Εικόνα 5.20: *Εστίαση στα 600-700 test steps. Μπλε: DERP και Πράσινο: Προτεινόμενο σύστημα*

Αφού μελετήσουμε τις δύο εικόνες, μπορούμε να καταλάβουμε ότι ο predictor βοήθησε το περιβάλλον διαχείρισης να είναι πιο οικονομικό (δεδομένου ότι σε μεγάλες μειώσεις του load ανταποκρίθηκε άμεσα μειώνοντας τους πόρους) και πιο κοντά στις ανάγκες των πελατών (αυξάνοντας real time τους πόρους, όταν το load αυξάνεται πολύ). Αναμέναμε μια τέτοια απόδοση γιατί το σύστημα μας εκτός από την πληροφορία της χρονικής στιγμής t που λαμβάνεται στο απλό σύστημα ενισχυτικής μάθησης, λαμβάνει και μια πρόβλεψη για την χρονική στιγμή t+1. Σίγουρα η πρόβλεψη μας δεν είναι απόλυτα ακριβής, όμως είναι αρκετά κοντά στις πραγματικές τιμές της χρονοσειράς και προβλέπει πιο γρήγορα κάποια αύξηση ή μείωση του φόρτου εργασίας. Με αυτόν τον τρόπο, επιτυγχάνουμε γρηγορότερη προσαρμογή στις αλλαγές του περιβάλλοντος. Έτσι, γλυτώνουμε επιπλέον έξοδα των υπηρεσιών των υπολο-

γιστικών νεφών, μειώνοντας τις περιπτώσεις υπερ-πρόβλεψης και ικανοποιούμε πιο άμεσα τους πελάτες, μειώνοντας τις περιπτώσεις υπο-πρόβλεψης. Επεκτάσεις του συστήματος μας για ακόμα καλύτερα αποτελέσματα παρουσιάζονται στο Κεφάλαιο 6.2.

Κεφάλαιο 6

Επίλογος

Στο κεφάλαιο αυτό παρουσιάζονται τα συμπεράσματα και η αξιόγηση του συστήματος μας και κάποιες προτάσεις για μελλοντικές επεκτάσεις.

6.1 Συμπεράσματα

Σε αυτήν την εργασία προτείνουμε μια νέα μορφή συστήματος διαχείρισης υπολογιστικών πόρων που συνδυάζει δύο προυπάρχουσες τεχνικές. Υπάρχουν πολλές υπηρεσίες που βασίζουν την κλιμάκωση του cloud περιβάλλοντος τους στην πρόβλεψη του φόρτους εργασίας τους με βάση ιστορικά δεδομένα και άλλες οι οποίες χρησιμοποιούν τεχνικές ενισχυτικής μάθησης όπως μαρκοβιανές διαδικασίες, τεχνικές Q-Learning και βαθιάς ενισχυτικής μάθησης με χρήση της ελαστικότητας του περιβάλλοντος. Εμείς υλοποιήσαμε ένα μοντέλο που λαμβάνει πληροφορία και από τις δύο αυτές τεχνικές για να αποφασίσει την επόμενη κίνηση του. Ουσιαστικά, το μοντέλο μας βασίζεται σε έναν πράκτορα βαθιάς ενισχυτικής μάθησης που εκτός από τα δεδομένα της κατάστασης του συστήματος λαμβάνει και μία επιπλέον πληροφορία για ένα χρονικό βήμα παραπάνω (με χρήση ενός μοντέλου πρόβλεψης χρονοσειρών που έχουμε ήδη εκπαιδεύσει με ιστορικά δεδομένα). Αναλυτικά, το σύστημα μας έχει τα παρακάτω πλεονεκτήματα:

- Ο πράκτορας έχει περισσότερα δεδομένα για το περιβάλλον (παρελθόν, παρόν, μέλλον) του και έτσι έχει την δυνατότητα να αποφασίζει πιο αποδοτικά την ενέργεια του.
- Η πρόταση μας βελτιώνει την αποδοτικότητα του DERP σε βαθμό που είναι σημαντικό για τα περιβάλλοντα υπολογιστικών νεφών. Ακόμα και αν η συμπεριφορά τους φαίνεται να μοιάζει στο μεγαλύτερο βαθμό, το σύστημα μας προσαρμόζεται άμεσα στις αλλαγές του φόρτου εργασίας (αφού τις έχει προβλέψει σε προηγούμενο χρόνο), σε αντίθεση, με την υλοποίηση που προσομοιάζει το DERP που αργεί κατά δύο-τέσσερα βήματα να καταλάβει την αλλαγή.
- Το σύστημα μας μπορεί να λειτουργήσει αποδοτικά για περίπλοκες χρονοσειρές φόρτου εργασίας, που αναπτύσουν συχνές αυξομειώσεις και θόρυβο κάτι που δείχνει ότι μπορεί να προσαρμοστεί σε οποιαδήποτε είδους χρονοσειρά φόρτου εργασίας.
- Δεν χρειάζεται μεγάλη χρήση μνήμης για την υλοποίηση του, δεδομένου ότι χρησιμοποιεί νευρωνικά δίκτυα, το μόνο που χρειάζεται είναι η αποθήκευση των βαρών του

κάθε νευρωνικού δικτύου.

Συμπερασματικά, το σύστημα μας λαμβάνει υπόψιν τις απαιτήσεις του πελάτη και του κατόχου του περιβάλλοντος υπολογιστικών νεφών και εκτελεί την όσο το δυνατόν καλύτερη δράση (action).

6.2 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας θα μπορούσε να βελτιωθεί και να επεκταθεί περαιτέρω, τουλάχιστον ως προς πέντε κατευθύνσεις. Συγκεκριμένα, αναφέρονται τα ακόλουθα:

- Βελτίωση της μορφής των μοντέλων πρόβλεψης ώστε να υπολογίζουν δύο ή παραπάνω χρονικές στιγμές στο μέλλον. Με αυτόν τον τρόπο, το σύστημα θα έχει ακόμα μεγαλύτερη πληροφορία για το μέλλον και θα μπορεί να αυτοματοποιηθεί ακομά περισσότερο προβλέποντας μελλοντικές αυξομειώσεις του φόρτου εργασίας πιο έγκαιρα. Ωστόσο, όσο απομακρυνόμαστε από την τωρινή χρονική στιγμή, τόσο το ποσοστό σφάλματος της πρόβλεψης αυξάνεται. Γι' αυτόν τον λόγο, θα προτείναμε να υπολογίζονται το πολύ πέντε επόμενες στιγμές, ώστε το σύστημα να μην καταλήξει να λειτουργεί με χρήση εσφαλμένων προβλέψεων.

- Αύξηση των παραμέτρων του περιβάλλοντος του συστήματος. Έχοντας ως γνώμονα την προγενέστερη έρευνα (DERP) το σύστημα θα μπορούσε να λειτουργήσει με παραπάνω από μια χρονοσειρές. Μελετώντας τις επιπλέον χρονοσειρές, όπως την χρησιμοποίηση CPU, RAM, την χωρητικότητα ή οποιαδήποτε άλλη μεταβλητή, θα μπορούσαμε να προσθέσουμε έναν κατάλληλο predictor για κάθε μία ή να συγκεκτρώσουμε όλες τις χρονοσειρές και να τις διαχειριστούμε σαν ένα πρόβλημα multivariate time-series forecasting. Στην συνέχεια, ο στόχος είναι να βρεθεί κάποιο άλλο βέλτιστο μοντέλο να αντικαταστήσει τον predictor μας. Με έναν multivariate time-series predictor, για κάθε χρονική στιγμή, θα έχουμε μια πρόβλεψη για την τιμή όλων των μεταβλητών περιβάλλοντος, χρησιμοποιώντας κάποιες εξαρτήσεις μεταξύ τους.

- Βελτίωση του μοντέλου βαθιάς ενισχυτικής μάθησης χρησιμοποιώντας Double Deep Q-Learning ή Dueling Deep Q-Learning τα οποία γενικά έχουν αποδειχτεί ότι έχουν καλύτερες επιδόσεις από το Simple Deep Q-Learning.

- Βελτίωση του τρόπου εκπαίδευσης του predictor. Στην υλοποίηση μας το μοντέλο πρόβλεψης εκπαιδεύεται σε ένα μέρος ιστορικών δεδομένων πριν προσθεθεί στον πράκτορα ενισχυτικής μάθησης. Σε μία άλλη υλοποίηση θα μπορούσαμε την ώρα που εκπαιδεύουμε το μοντέλο ενισχυτικής μάθησης με τυχαίες κινήσεις να γίνεται και η εκπαίδευση του μοντέλου πρόβλεψης. Με αυτόν τον τρόπο, βέβαια, πρέπει να προσέξουμε καλά και το δίλημμα exploration-exploitation έτσι ώστε ο πράκτορας μας να μην μάθει με χρήση λανθασμένων προβλέψεων.

- Αύξηση των προτεινόμενων κινήσεων του πράκτορα. Οι ενέργειες που μπορεί να κάνει ο πράκτορας μας είναι τρείς, αύξηση ή μείωση των υπολογιστικών πόρων κατά ένα

και η παραμονή στην ίδια κατάσταση. Θα μπορούσαμε να έχουμε την δυνατότητα να αυξάνουμε ή να μειώνουμε μεγαλύτερο αριθμό υπολογιστικών πόρων όταν λαμβάνουμε μια πολύ μεγάλη αυξομείωση του φόρτου εργασίας.

Βιβλιογραφία

[1] Brian Hayes. *Cloud computing*, 2008.

[2] Nikolas Roman Herbst, Samuel Kounev και Ralf Reussner. *Elasticity in cloud computing: What it is, and what it is not. 10th International Conference on Autonomic Computing ({ICAC} 13)*, σελίδες 23–27, 2013.

[3] Sukhpal Singh και Inderveer Chana. *Cloud resource provisioning: survey, status and future research directions*. *Knowledge and Information Systems*, 49(3):1005–1069, 2016.

[4] *Elastic Cloud Computing*. `https://en.wikipedia.org/wiki/Elasticity_(cloud_computing)`. Ημερομηνία πρόσβασης: 4-3-2021.

[5] Rafael Moreno-Vozmediano, Rubén S Montero, Eduardo Huedo και Ignacio M Llorente. *Efficient resource provisioning for elastic Cloud services based on machine learning techniques*. *Journal of Cloud Computing*, 8(1):1–18, 2019.

[6] *Time Series*, σελίδες 536–539. Springer New York, New York, NY, 2008.

[7] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan και Rajkumar Buyya. *Workload prediction using ARIMA model and its impact on cloud applications' QoS*. *IEEE transactions on cloud computing*, 3(4):449–458, 2014.

[8] George EP Box, Gwilym M Jenkins, Gregory C Reinsel και Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[9] Carlos Vazquez, Ram Krishnan και Eugene John. *Time Series Forecasting of Cloud Data Center Workloads for Dynamic Resource Provisioning*. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 6(3):87–110, 2015.

[10] Christoph Bergmeir, Rob J Hyndman και José M Benítez. *Bagging exponential smoothing methods using STL decomposition and Box-Cox transformation*. *International journal of forecasting*, 32(2):303–312, 2016.

[11] Sean J. Taylor και Benjamin Letham. *Forecasting at Scale*. *The American Statistician*, 72(1):37–45, 2018.

[12] *Prophet: Automatic Forecasting Procedure*. `https://pypi.org/project/fbprophet/`. Ημερομηνία πρόσβασης: 19-02-2021.

[13] *Prophet: Automatic Forecasting Procedure*. https://cran.r-project.org/web/packages/prophet/. Ημερομηνία πρόσβασης: 19-02-2021.

[14] Simon S Haykin και others. *Νευρωνικά Δίκτυα και Μηχανική Μάθηση/Simon Haykin.*, 2010.

[15] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu και Michael S Lew. *Deep learning for visual understanding: A review*. *Neurocomputing*, 187:27–48, 2016.

[16] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. *arXiv preprint arXiv:1609.04747*, 2016.

[17] Sepp Hochreiter και Jürgen Schmidhuber. *Long short-term memory*. *Neural computation*, 9(8):1735–1780, 1997.

[18] Kyunghyun Cho, Bartvan Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk και Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. *CoRR*, αβσ/1406.1078, 2014.

[19] Aniekan Essien και Cinzia Giannetti. *A deep learning model for smart manufacturing using convolutional LSTM neural network autoencoders*. *IEEE Transactions on Industrial Informatics*, 16(9):6069–6078, 2020.

[20] Anastasia Borovykh, Sander Bohte και Cornelis W Oosterlee. *Conditional time series forecasting with convolutional neural networks*. *arXiv preprint arXiv:1703.04691*, 2017.

[21] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[22] Wei Bao, Jun Yue και Yulei Rao. *A deep learning framework for financial time series using stacked autoencoders and long-short term memory*. *PloS one*, 12(7):ε0180944, 2017.

[23] Tiago Prado Oliveira, Jamil Salem Barbar και Alexsandro Santos Soares. *Multilayer perceptron and stacked autoencoder for Internet traffic prediction*. *IFIP International Conference on Network and Parallel Computing*, σελίδες 61–71. Springer, 2014.

[24] André Gensler, Janosch Henze, Bernhard Sick και Nils Raabe. *Deep Learning for solar power forecasting—An approach using AutoEncoder and LSTM Neural Networks*. *2016 IEEE international conference on systems, man, and cybernetics (SMC)*, σελίδες 002858–002865. IEEE, 2016.

[25] Xueheng Qiu, Le Zhang, Ye Ren, Ponnuthurai N Suganthan και Gehan Amaratunga. *Ensemble deep learning for regression and time series forecasting*. *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, σελίδες 1–6. IEEE, 2014.

[26] HD Nguyen, Kim Phuc Tran, S Thomassey και M Hamad. *Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management*. *International Journal of Information Management*, σελίδα 102282, 2020.

[27] Anton Maximilian Schäfer. *Reinforcement learning with recurrent neural networks*. 2008.

[28] ΝΙΚΗΦΟΡΟΣ Ι. ΜΑΝΔΗΛΑΡΑΣ. *Σχεδιασμός και υλοποίηση ευφυούς μηχανισμού διαμοιρασμού κοινόχρηστων πόρων, σε πολυπύρηνα συστήματα, με χρήση βαθιάς ενισχυτικής μάθησης*. Διπλωματική εργασία, Εθνικό Μετσόβιο Πολυτεχνείο, 2020.

[29] *Introduction to Markov Decision Process*. `https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da`. Ημερομηνία πρόσβασης: 23-02-2021.

[30] Richard S Sutton. *Learning to predict by the methods of temporal differences*. *Machine learning*, 3(1):9–44, 1988.

[31] *Temporal difference learning*. `http://www.scholarpedia.org/article/Temporal_difference_learning`. Ημερομηνία πρόσβασης: 23-02-2021.

[32] Christopher JCH Watkins και Peter Dayan. *Q-learning*. *Machine learning*, 8(3-4):279–292, 1992.

[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra και Martin Riedmiller. *Playing atari with deep reinforcement learning*. *arXiv preprint arXiv:1312.5602*, 2013.

[34] Irena Koprinska, Dengsong Wu και Zheng Wang. *Convolutional neural networks for energy time series forecasting*. *2018 international joint conference on neural networks (IJCNN)*, σελίδες 1–8. IEEE, 2018.

[35] Işil Yenidoğan, Aykut Çayir, Ozan Kozan, Tuğçe Dağ και Çiğdem Arslan. *Bitcoin forecasting using ARIMA and prophet*. *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, σελίδες 621–624. IEEE, 2018.

[36] Peter T Yamak, Li Yujian και Pius K Gadosey. *A comparison between arima, lstm, and gru for time series forecasting*. *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, σελίδες 49–55, 2019.

[37] Li Ruan, Yu Bai, Shaoning Li, Shuibing He και Limin Xiao. *Workload time series prediction in storage systems: a deep learning based approach*. *Cluster Computing*, σελίδες 1–11, 2021.

[38] Ioannis Konstantinou, Evangelos Angelou, Dimitrios Tsoumakos, Christina Boumpouka, Nectarios Koziris και Spyros Sioutas. *Tiramola: elastic nosql provisioning through a cloud management platform*. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, σελίδες 725–728, 2012.

[39] Konstantinos Lolos, Ioannis Konstantinou, Verena Kantere και Nectarios Koziris. *Elastic resource management with adaptive state space partitioning of Markov Decision Processes*. *arXiv preprint arXiv:1702.02978*, 2017.

[40] Constantinos Bitsakos, Ioannis Konstantinou και Nectarios Koziris. *Derp: A deep reinforcement learning cloud system for elastic resource provisioning*. *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, σελίδες 21–29. IEEE, 2018.

[41] Seyed Mohammad Reza Nouri, Han Li, Srikumar Venugopal, Wenxia Guo, Ming-Yun He και Wenhong Tian. *Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications*. *Future Generation Computer Systems*, 94:765–780, 2019.

[42] Zhiping Peng, Jianpeng Lin, Delong Cui, Qirui Li και Jieguang He. *A multi-objective trade-off framework for cloud resource scheduling based on the deep Q-network algorithm*. *Cluster Computing*, σελίδες 1–15, 2020.

[43] *Research Google Colaboratory*. `https://colab.research.google.com`. Ημερομηνία πρόσβασης: 4-3-2021.