ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Ενότητα 11– Supervised Learning
Neural Networks

Μέθοδοι Μηχανικής Μάθησης στα Χρηματοοικονομικά

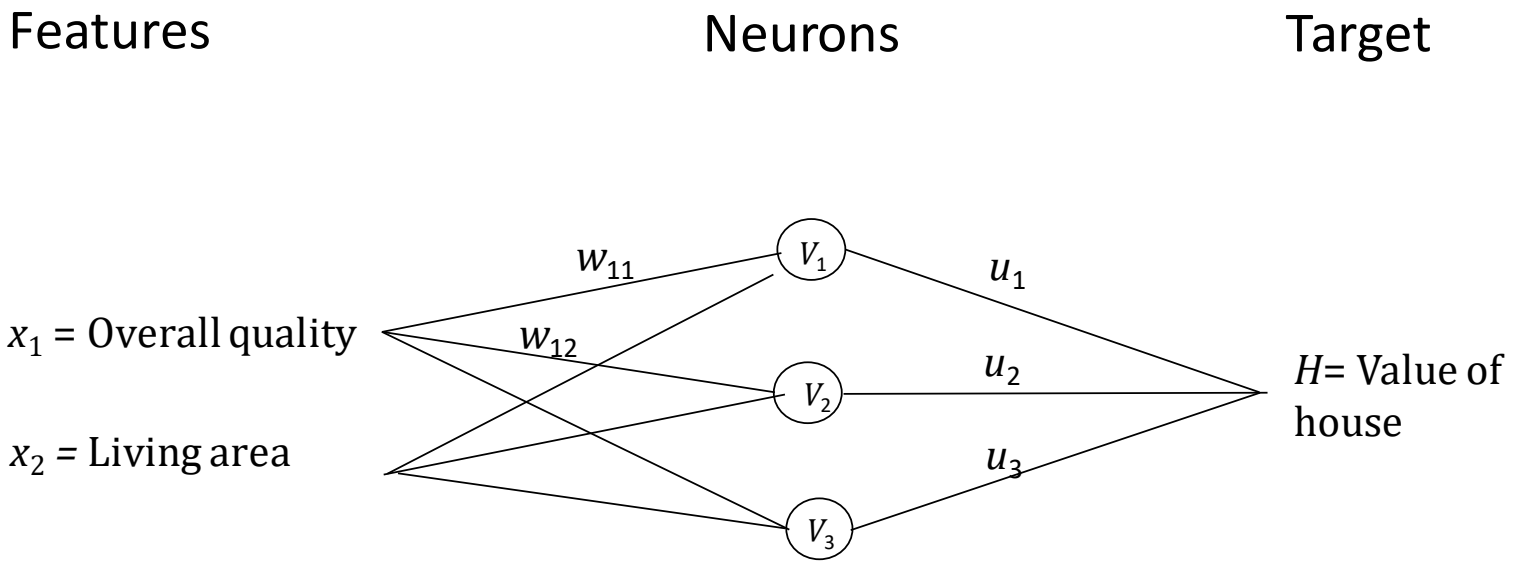
Αθανάσιος Σάκκας, ΟΠΑ

Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANN)

- Τα ANN είναι αλγόριθμοι μηχανικής μάθησης με εφαρμογές σε διάφορες επιστήμες.
- Οι αλγόριθμοι αυτοί μαθαίνουν τις σχέσεις μεταξύ target και features χρησιμοποιώντας ένα δίκτυο από συναρτήσεις.
- Οποιαδήποτε μη γραμμική σχέση μεταξύ target και features μπορεί να εξεταστεί με τη χρήση των ANN.

Στην επόμενη διαφάνεια βλέπουμε ένα απλό ANN εξετάζοντας την περίπτωση του Iowa House με δύο χαρακτηριστικά (features) overall quality και living area.

Ένα απλό ANN



Έχει 3 layers :

1. input layer = 2 features
2. Output layer = Iowa House Value
3. Hidden Layer (k)= 3 neurons

- w_{jk} model parameter (weight) : Συνδέει την τιμή στον k νευρώνα (neuron) με την τιμή x_j του j feature. Συνολικά έχουμε 6 weights.
- u_k model parameter (weight) : Συνδέει την τιμή του target (House Value) με την τιμή στον k νευρώνα (neuron)
- V_k η τιμή στον k νευρώνα (neuron). Υπολογίζεται από το μοντέλο.

Activation Functions - Περιγραφική εισαγωγή

- Η H δεν σχετίζεται άμεσα με τα χαρακτηριστικά (features).
- Υπάρχει hidden layer με νευρώνες (neurons).
- H σχετίζεται με V_k
- Η V_k σχετίζεται με τα χαρακτηριστικά (features). Πως υπολογίζεται? Ας δούμε τα βήματα για τον υπολογισμό της V_1

α. Πολλαπλασιάζουμε κάθε x_j με το weight που συνδέεται με την V_1

και προσθέτουμε τα αποτελέσματα

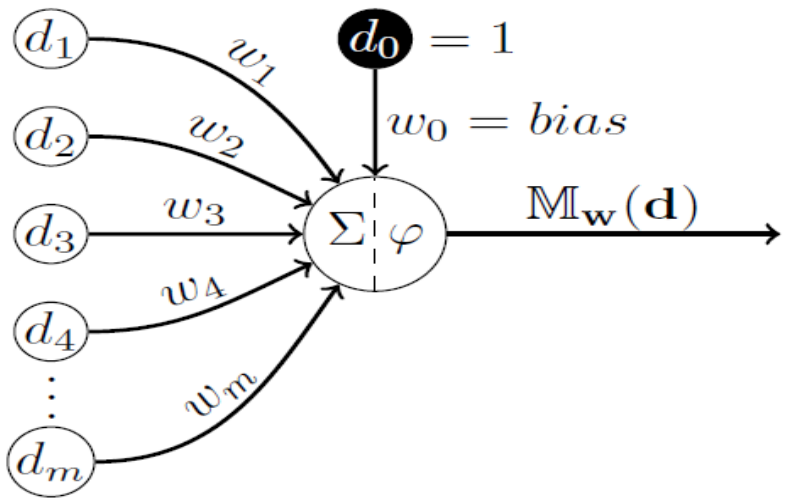
β. Προσθέτουμε έναν σταθερό όρο (bias).

γ. Εφαρμόζουμε μια **activation function** που συνδέει τα παραπάνω.

$$V_k = f(a_k + w_{1k}x_1 + w_{2k}x_2)$$

$$H = f(c + u_1V_1 + u_2V_2 + u_3V_3)$$

- **Το bias** αναφέρεται συχνά ως παράμετρος **πόλωσης** επειδή, ελλείψει οποιασδήποτε άλλης εισόδου, η έξοδος του σταθμισμένου αθροίσματος **πολώνεται ώστε να είναι η τιμή του σταθερού όρου**. Από τεχνικής άποψης, η συμπερίληψη της παραμέτρου πόλωσης ως επιπλέον βάρος σε αυτήν την πράξη αλλάζει τη συνάρτηση από γραμμική στις εισόδους σε **ομοπαράλληλη συνάρτηση** (**affine function**).
- Μια ομοπαράλληλη συνάρτηση αποτελείται από μια γραμμική συνάρτηση ακολουθούμενη από μια μετατόπιση. Αυτό σημαίνει απλώς ότι η αντιστοίχιση που ορίζεται από μια ομοπαράλληλη συνάρτηση από τις εισόδους στις εξόδους είναι γραμμική, αλλά το γράφημα αυτής της αντιστοίχισης δεν περνά απαραίτητα από την αρχή των αξόνων.



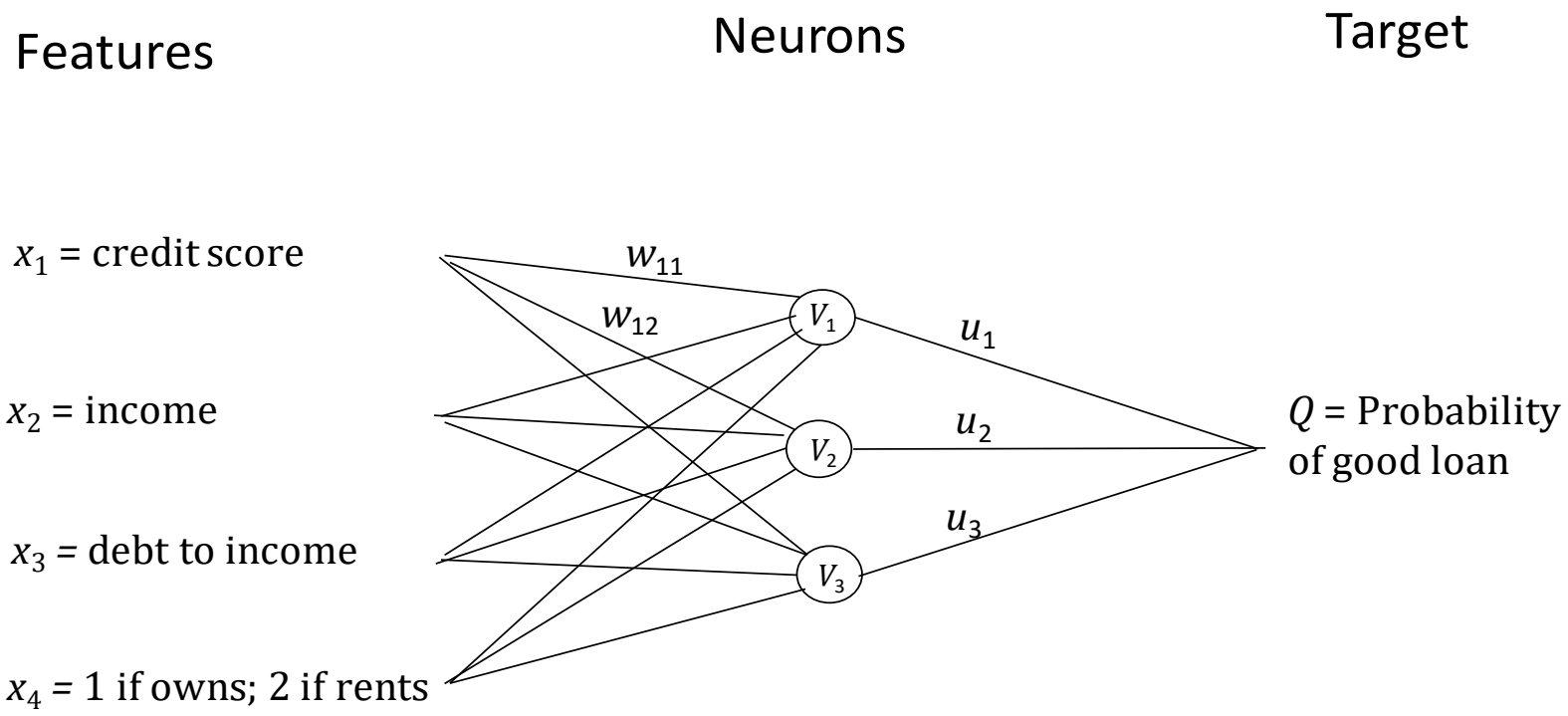
- Για παράδειγμα, εάν παραλείπαμε το εικονικό χαρακτηριστικό **d** [0] και στη συνέχεια πολλαπλασιάζαμε κάθε ένα από τα πραγματικά περι- γραφικά χαρακτηριστικά με ένα βάρος και έπειτα αθροίζαμε τα αποτελέσματα, θα ήταν ισοδύναμο της εφαρμογής μιας γραμμικής συνάρτησης στις εισόδους.

- Το γράφημα αυτής της συνάρτησης θα περνούσε από την αρχή των αξόνων, επειδή δεν υπάρχει τομή με τον άξονα *y* (ή όρος πόλωσης) που να συμπεριλαμβάνεται στον υπολογισμό.

- Ωστόσο, αν στη συνέχεια προσθέσουμε τον όρο πόλωσης (την τομή στον άξονα *y*) στα αποτελέσματα της γραμμικής συνάρτησης, μετατοπίζουμε τα αποτελέσματά της μακριά από την αρχή των αξόνων.

- **Συμπεριλαμβάνοντας τον όρο πόλωσης στο σύνολο των βαρών μαζί με ένα εικονικό περιγραφικό χαρακτηριστικό, η συνάρτηση που υλοποιείται πολλαπλασιάζοντας τα βάρη με τα εκτεταμένα περιγραφικά χαρακτηριστικά αποτελεί πλέον ομοπαράλληλη συνάρτηση.**

Παράδειγμα του Lending Club με 4 χαρακτηριστικά



Η αντικειμενική συνάρτηση (objective function) για το νευρωνικό δίκτυο στη διαφάνεια 5 θα πρέπει να αναγνωρίζει ότι το δίκτυο αποδίδει καλά όταν τα δάνεια που αποδείχθηκαν «καλά» στο training set έχουν υψηλή τιμή Q, ενώ τα δάνεια που κατέληξαν σε αθέτηση πληρωμών (default) έχουν χαμηλή τιμή Q. Για τον λόγο αυτό, είναι κατάλληλη η χρήση της αντικειμενικής συνάρτησης που βασίζεται στη μέγιστη πιθανοφάνεια (maximum likelihood). Αυτό σημαίνει ότι επιλέγουμε τις 19 παραμέτρους του μοντέλου έτσι ώστε να μεγιστοποιείται:

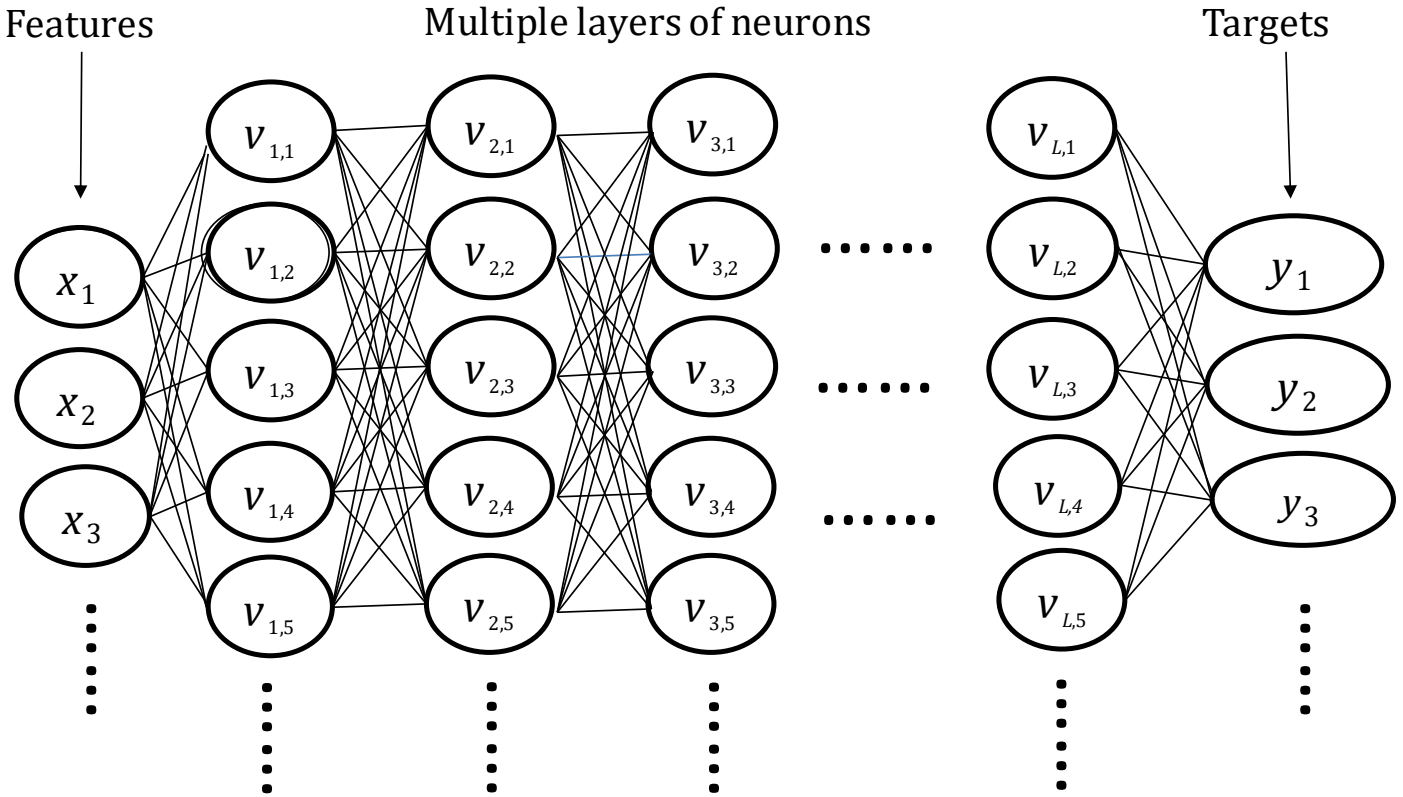
$$\sum_{\substack{\text{Positive}\\ \text{Outcomes}}} \ln(Q) + \sum_{\substack{\text{Negative}\\ \text{Outcomes}}} \ln(1-Q)$$

ή ισοδύναμα, επιλέγουμε τις παραμέτρους του μοντέλου έτσι ώστε να ελαχιστοποιείται η συνάρτηση κόστους:

$$\text{Cost Function} = -\sum_{\substack{\text{Positive}\\ \text{Outcomes}}} \ln(Q) - \sum_{\substack{\text{Negative}\\ \text{Outcomes}}} \ln(1-Q)$$

Η συνάρτηση αυτή λαμβάνει τιμή μηδέν όταν το μοντέλο παρέχει τέλειες προβλέψεις, δηλαδή όταν $Q = 1$ για θετικά αποτελέσματα και $Q = 0$ για αρνητικά αποτελέσματα. Η τιμή της συνάρτησης κόστους αυξάνεται όσο οι προβλέψεις του μοντέλου δυσκολεύονται περισσότερο να διαχωρίσουν καθαρά τα θετικά από τα αρνητικά αποτελέσματα.

A General Neural Network



- Η αρχιτεκτονική δικτύου που φαίνεται στο αποτελεί παράδειγμα **δικτύου προς τα εμπρός τροφοδότησης** (**feedforward network**).
- Το **βάθος (depth)** ενός νευρωνικού δικτύου ισούται με το πλήθος των κρυφών στρώσεων συν τη στρώση εξόδου.

Στην αναφορά Cybenko (1988) αποδείχτηκε ότι ένα δίκτυο με τρεις στρώσεις (επεξεργασίας) νευρώνων (δηλαδή δύο κρυφές στρώσεις και μία στρώση εξόδου) μπορεί να προσεγγίσει οποιαδήποτε συνάρτηση με αυθαίρετη ακρίβεια. Έτσι, εδώ ορίζουμε ως δύο το ελάχιστο πλήθος κρυφών στρώσεων που είναι απαραίτητες για να θεωρηθεί ένα δίκτυο βαθύ. Υπό αυτόν τον ορισμό, το δίκτυο θα περιγράφονταν ως βαθύ. Ωστόσο, τα περισσότερα βαθιά δίκτυα έχουν πολύ περισσότερες από δύο κρυφές στρώσεις. Σήμερα ορισμένα βαθιά δίκτυα έχουν δεκάδες ή και εκατοντάδες στρώσεις

- Υπάρχουν βάρη (weights) που συνδέονται με καθεμία από τις γραμμές στο Σχήμα στη διαφάνεια 7. Οι συναρτήσεις ενεργοποίησης (activation functions) χρησιμοποιούνται, όπως αναφέρθηκε προηγουμένως, για:

Α.Να συνδέουν τις τιμές των νευρώνων του πρώτου hidden layer με τις τιμές των χαρακτηριστικών (features), δηλαδή να συνδέουν τα V_{1k}με τα x_j.

Β.Να συνδέουν τις τιμές ενός νευρώνα στο hidden layer $l+1$με τις τιμές των νευρώνων στο hidden layer $l(1 \leq l \leq L-1)$.

Γ. Να συνδέουν τις τιμές-στόχους (target values) με τις τιμές του τελευταίου hidden layer, δηλαδή να συνδέουν τα yμε τα V_{Lk}.

Activation Functions (1)

- Μία activation function συσχετίζει τιμές σε έναν νευρώνα με έναν γραμμικό συνδυασμό τιμών στο προηγούμενο layer.

$V_{k,j} = f(y)$ where $y = a + b_1 V_{k-1,1} + b_2 V_{k-1,2} + \cdots$

- Examples of activation functions are:

  Linear: $f(y) = y$

  Sigmoid: $f(y) = \frac{1}{1+e^{-y}}$ (gives values between 0 and 1)

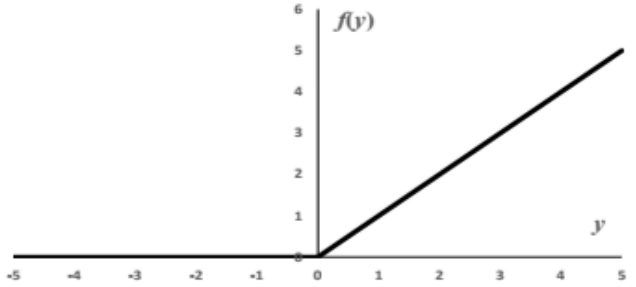
  Hyperbolic tangent: $f(y) = \frac{e^{2y}-1}{e^{2y}+1}$ (gives values between -1 and 1)

  ReLU: $f(y) = \max(y, 0)$

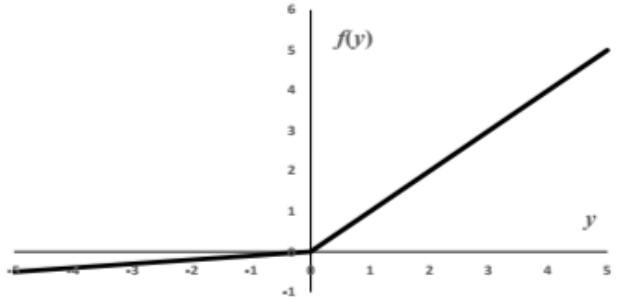
  Leaky ReLU: : $f(y) = \max(y, 0)$ if $y \geq 0$ and max (ay,0) if $y < 0$

Activation Functions (2)

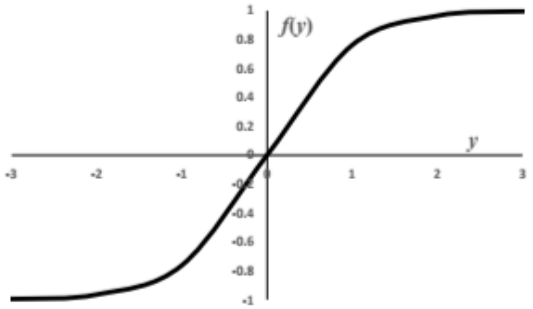
(a) ReLU



(b) Leaky ReLU



(c) Hyperbolic Tangent



Activation Functions (3)

- Η ίδια activation function χρησιμοποιείται συνήθως σε όλο το δίκτυο, εκτός πιθανώς όταν συσχετίζονται οι τιμές στο output layer με τιμές στο προηγούμενο layer.

- Όταν **το target είναι μια αριθμητική εκτίμηση (numerical estimate),** η τελική activation είναι συνήθως $f(y) = y$.

- **Στο classification,** το output layer είναι η πιθανότητα ενός θετικού αποτελέσματος. Η sigmoid function χρησιμοποιείται συνήθως για να συσχετίσει τις τιμές στο output layer με τις τιμές του προηγούμενου layer.

Activation Functions (4)

Στόχος είναι η **επιλογή των weights και των biases για** την **ελαχιστοποίηση του mse** (mean squared error) ή του **mae** (mean absolute error) στην περίπτωση μιας πρόβλεψης (**prediction**) **- cost/loss function.**

ή

για τη μεγιστοποίηση της πιθανότητας (**maximize likelihood**) σε περίπτωση ταξινόμησης (**classification**).

Ένα πιθανό σύνολο εξισώσεων για το παράδειγμα Iowa House price (13 παράμετροι)

$$V_1 = \frac{1}{1 + \exp(-a_1 - w_{11}x_1 - w_{21}x_2)}$$

$$V_2 = \frac{1}{1 + \exp(-a_2 - w_{12}x_1 - w_{22}x_2)}$$

$$V_3 = \frac{1}{1 + \exp(-a_3 - w_{13}x_1 - w_{23}x_2)}$$

$$H = c + u_1V_1 + u_2V_2 + u_3V_3$$

Ένα πιθανό σύνολο εξισώσεων για το παράδειγμα του Lending Club (19 παράμετροι)

$$V_1 = \frac{1}{1 + \exp(-a_1 - w_{11}x_1 - w_{21}x_2 - w_{31}x_3 - w_{41}x_4)}$$

$$V_2 = \frac{1}{1 + \exp(-a_2 - w_{12}x_1 - w_{22}x_2 - w_{32}x_3 - w_{42}x_4)}$$

$$V_3 = \frac{1}{1 + \exp(-a_3 - w_{13}x_1 - w_{23}x_2 - w_{32}x_3 - w_{41}x_4)}$$

$$Q = \frac{1}{1 + \exp(-c - u_1V_1 - u_2V_2 - u_3V_3)}$$

Universal Approximation Theorem

- Οποιαδήποτε συνεχής συνάρτηση μπορεί να προσεγγιστεί με αυθαίρετη ακρίβεια με ένα hidden layer (K. Hornik: *Neural Networks*, 1991, 4:251-257).

- Αλλά αυτό μπορεί να απαιτεί έναν πολύ μεγάλο αριθμό νευρώνων.

- Η χρήση πολλών hidden layers μπορεί να είναι υπολογιστικά πιο αποτελεσματική.

Αριθμός παραμέτρων

- Τα νευρωνικά δίκτυα μπορούν να έχουν πολύ μεγάλο αριθμό παραμέτρων.

- Εάν υπάρχουν F features, H hidden layers, M neurons σε κάθε hidden layer και T targets ο αριθμός των παραμέτρων είναι

$$(F+1)M+M(M+1)(H-1)+(M+1)T$$

- Το νευρωνικό δίκτυο της IOWA HOUSE PRICE έχει 13 παραμέτρους και το νευρωνικό δίκτυο του LENDING CLUB έχει 19 παραμέτρους.

- Αλλά στην πράξη ένα μικρό νευρωνικό δίκτυο μπορεί να έχει 4 χαρακτηριστικά, 3 hidden layers, 30 neurons ανά layers και ένα target για συνολικά 2.041 παραμέτρους.

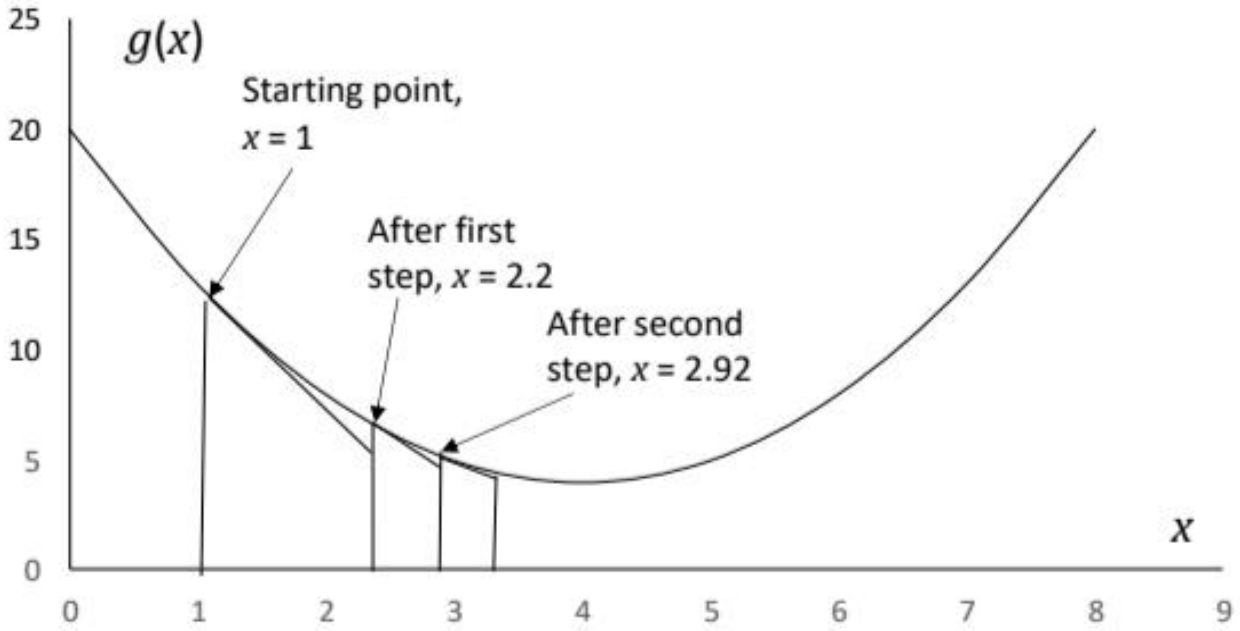
- Τα μεγαλύτερα νευρωνικά δίκτυα έχουν δεκάδες, ή και εκατοντάδες, χιλιάδες παραμέτρους.

Gradient Descent και NN

- Όταν τα νευρωνικά δίκτυα χρησιμοποιούνται για supervised learning, η ελαχιστοποίηση της αντικειμενικής συνάρτησης πραγματοποιείται με τη χρήση του αλγορίθμου gradient descent.
- Ας υποθέσουμε ότι υπάρχουν N παράμετροι. Μια αντικειμενική συνάρτηση (π.χ. mse ή μια βασισμένη σε εκτιμήσεις μέγιστης πιθανότητας) ελαχιστοποιείται για training set data.
- Ένας gradient descent algorithm ξεκινά με ένα σύνολο τιμών για τις παραμέτρους N, υπολογίζει την κατεύθυνση της πιο απότομης καθόδου στην κοιλάδα, κάνει ένα βήμα, υπολογίζει μια νέα κατεύθυνση της πιο απότομης κατάβασης, κάνει ένα άλλο βήμα και ούτω καθεξής.
- Οι μερικές παράγωγοι σε σχέση με τις παραμέτρους υπολογίζονται με μια διαδικασία γνωστή ως **backpropagation**. Αυτό περιλαμβάνει την επαναλειτουργία μέσω του δικτύου χρησιμοποιώντας τον κανόνα της αλυσίδας.
- Το μέγεθος του βήματος καθορίζεται από το “learning rate”. Εάν το βήμα είναι πολύ μικρό, ο αλγόριθμος θα είναι πολύ αργός. Εάν είναι πολύ μεγάλο, ενδέχεται να υπάρξουν ταλαντώσεις:

$$\text{Increase in variable} = -\text{learning rate} \times \text{gradient}$$

Πολύ απλό παράδειγμα:
Υπολογισμός της τιμής του x που ελαχιστοποιεί το y όταν $y=x^2-8x+20$



Learning Rate= 0.2

Iteration	x	Gradient	Change in x
0	1.000	-6.000	1.200
1	2.200	-3.600	0.720
2	2.920	-2.160	0.432
3	3.352	-1.296	0.259
4	3.611	-0.778	0.156
5	3.767	-0.467	0.093
6	3.860	-0.280	0.056
7	3.916	-0.168	0.034
8	3.950	-0.101	0.020
9	3.970	-0.060	0.012
10	3.982	-0.036	0.007
11	3.989	-0.022	0.004
12	3.993	-0.013	0.003
13	3.996	-0.008	0.002

Learning Rate= 0.02

Iteration	x	Gradient	Change in x
0	1.000	-6.000	0.120
1	1.120	-5.760	0.115
2	1.235	-5.530	0.111
3	1.346	-5.308	0.106
4	1.452	-5.096	0.102
5	1.554	-4.892	0.098
6	1.652	-4.697	0.094
7	1.746	-4.509	0.090
8	1.836	-4.328	0.087
9	1.922	-4.155	0.083
10	2.006	-3.989	0.080
11	2.085	-3.829	0.077
12	2.162	-3.676	0.074
13	2.235	-3.529	0.071

Learning Rate = 1.2

Iteration	x	Gradient	Change in x
0	1.000	-6.000	7.200
1	8.200	8.400	-10.080
2	-1.880	-11.760	14.112
3	12.232	16.464	-19.757
4	-7.525	-23.050	27.660
5	20.135	32.269	-38.723
6	-18.589	-45.177	54.213
7	35.624	63.248	-75.898
8	-40.274	-88.547	106.257
9	65.983	123.966	-148.760
10	-82.776	-173.553	208.263
11	125.487	242.974	-291.569
12	-166.082	-340.163	408.196
13	242.114	476.229	-571.475

Backpropagation (Οπισθοδιάδοση)

- Σε ένα νευρωνικό δίκτυο, υπάρχουν πολλά βάρη (weights) και σταθεροί όροι (biases) που πρέπει να προσδιοριστούν.
- Ο υπολογισμός της κλίσης (gradient) της αντικειμενικής συνάρτησης ως προς κάθε μία από αυτές τις παραμέτρους, ώστε να χρησιμοποιηθεί ο αλγόριθμος gradient descent, φαίνεται αρχικά να είναι μία αδύνατη διαδικασία.
- Ευτυχώς, έχει αναπτυχθεί μία αποτελεσματική μέθοδος συντόμευσης αυτής της διαδικασίας.
- Η μέθοδος αυτή είναι γνωστή ως **backpropagation (οπισθοδιάδοση)** και περιλαμβάνει τον υπολογισμό των απαραίτητων gradients ξεκινώντας από το τέλος του δικτύου και προχωρώντας προς την αρχή.
- Με βάση τον κανόνα αλυσίδας (chain rule), ο οποίος αποτελεί έναν τρόπο υπολογισμού μερικών παραγώγων (ή gradients) σύνθετων συναρτήσεων, αν για παράδειγμα η συνάρτηση f είναι συνάρτηση της g, και η g είναι συνάρτηση της x, τότε μπορούμε να χρησιμοποιήσουμε τον κανόνα αλυσίδας για να υπολογίσουμε τη μερική παράγωγο της σύνθετης συνάρτησης $f\big(g(x)\big)$ ως προς το x.

- Στο backpropagation εφαρμόζουμε επαναλαμβανόμενα τον κανόνα αλυσίδας, ώστε να υπολογίσουμε τη μερική παράγωγο του τετραγωνικού σφάλματος (ή οποιασδήποτε άλλης συνάρτησης απώλειας) για μία πρόβλεψη ως προς τα biases και τα weights του δικτύου.
- Η μερική παράγωγος της αντικειμενικής συνάρτησης ως προς ένα bias ή ένα weight προκύπτει ως το άθροισμα των επιμέρους μερικών παραγώγων που υπολογίζονται για κάθε πρόβλεψη.

- Το βασικό σημείο είναι ότι ο υπολογισμός των μερικών παραγώγων για καθεμία από τις συναρτήσεις που συμμετέχουν σε ένα νευρωνικό δίκτυο είναι σχετικά απλός.
- Ο κανόνας αλυσίδας (chain rule) παρέχει έναν τρόπο γρήγορου συνδυασμού αυτών των απλών μερικών παραγώγων, ώστε να προκύψουν οι απαραίτητες μερικές παράγωγοι της αντικειμενικής συνάρτησης.

- Από τις συναρτήσεις ενεργοποίησης (activation functions) που εξετάσαμε, οι linear, sigmoid και tanh είναι παραγωγίσιμες σε κάθε σημείο.

- Οι ReLU και Leaky ReLU δεν είναι παραγωγίσιμες όταν η τιμή του ορίσματος είναι ίση με μηδέν.

- Ωστόσο, αυτό δεν δημιουργεί πρόβλημα, επειδή η πιθανότητα το όρισμα να είναι ακριβώς μηδέν είναι ουσιαστικά μηδενική.

Παραλλαγές της βασικής μεθόδου

- Τα νευρωνικά δίκτυα μαθαίνουν πιο αποτελεσματικά όταν τα χαρακτηριστικά (features) που εισάγονται στο μοντέλο γίνονται scaled. Επιπλέον, συνήθως είναι επιθυμητή κάποια μορφή regularization.
  - Παρόμοια με τη γραμμική παλινδρόμηση, το L1 regularization περιλαμβάνει την προσθήκη στην αντικειμενική συνάρτηση ενός σταθερού πολλαπλασιαστή επί του αθροίσματος των απόλυτων τιμών όλων των weights. Αντίστοιχα, το L2 regularization περιλαμβάνει την προσθήκη ενός σταθερού πολλαπλασιαστή επί του αθροίσματος των τετραγώνων όλων των weights στην αντικειμενική συνάρτηση. Ανάλογα με ό,τι συμβαίνει στη γραμμική παλινδρόμηση, το L1 regularization μηδενίζει ορισμένα weights, ενώ το L2 regularization μειώνει το μέσο μέγεθος των weights.
- Όπως αναφέρθηκε προηγουμένως, υπάρχουν συχνά δεκάδες ή και εκατοντάδες χιλιάδες παράμετροι σε ένα μοντέλο. Αυτό σημαίνει ότι αναζητούμε ένα ελάχιστο σε έναν χώρο πολλών διαστάσεων. Για να βελτιωθεί η λειτουργία του βασικού αλγορίθμου gradient descent, έχουν αναπτυχθεί διάφορες παραλλαγές του. Παραδείγματα είναι τα εξής:

- **Mini-batch stochastic gradient descent**:

Η μέθοδος αυτή χωρίζει τυχαία το training dataset σε μικρότερα υποσύνολα, γνωστά ως mini-batches. Αντί να χρησιμοποιούνται όλα τα δεδομένα εκπαίδευσης για τον υπολογισμό των gradients, οι παράμετροι του μοντέλου ενημερώνονται με βάση τα gradients που υπολογίζονται από ένα μόνο mini-batch κάθε φορά. Η διαδικασία αυτή είναι υπολογιστικά πιο αποδοτική. **Ένα *epoch* αντιστοιχεί σε ένα σύνολο επαναλήψεων που χρησιμοποιεί μία φορά ολόκληρο το training set.**

- **Gradient descent with momentum**:

Η μέθοδος αυτή υπολογίζει το gradient ως έναν εκθετικά φθίνοντα κινητό μέσο όρο προηγούμενων gradients. Η προσέγγιση αυτή βοηθά στην επιτάχυνση της μάθησης προς κατευθύνσεις όπου το gradient είναι σχετικά σταθερό.

- **Gradient descent with adaptive learning rates**:

Όπως είδαμε, είναι σημαντικό να επιλέγεται ένα κατάλληλο learning rate. Οι μέθοδοι adaptive learning rates προσαρμόζουν δυναμικά το learning rate κατά τη διάρκεια της εκπαίδευσης ώστε να βελτιώνεται η σύγκλιση του αλγορίθμου όπου learning rate που είναι πολύ μικρό οδηγεί στο να απαιτούνται πολλά epochs ώστε να επιτευχθεί ένα ικανοποιητικό αποτέλεσμα.

Αντίθετα, ένα learning rate που είναι πολύ μεγάλο μπορεί να προκαλέσει ταλαντώσεις (oscillations) και κακή απόδοση του μοντέλου. Οι παράμετροι του μοντέλου ενδέχεται να επωφελούνται από διαφορετικά learning rates σε διαφορετικά στάδια της εκπαίδευσης. Ένας δημοφιλής αλγόριθμος προσαρμοστικού learning rate είναι ο **Adam** (adaptive moment estimation). Ο Adam χρησιμοποιεί τόσο momentum όσο και έναν εκθετικά φθίνοντα μέσο όρο των προηγούμενων τετραγώνων των gradients.

•**Learning rate decay**:

Ένα μεγάλο learning rate μπορεί να λειτουργεί αποτελεσματικά στα αρχικά στάδια της εκπαίδευσης, όταν το μοντέλο βρίσκεται μακριά από το ελάχιστο της αντικειμενικής συνάρτησης, ενώ ένα μικρότερο learning rate είναι συχνά πιο κατάλληλο όταν το μοντέλο πλησιάζει στο ελάχιστο. Εκτός από τη χρήση adaptive learning rates, μπορεί επομένως να είναι χρήσιμο να μειώνεται σταδιακά το learning rate καθώς προχωρά η εκπαίδευση του αλγορίθμου. Η διαδικασία αυτή είναι γνωστή ως **learning rate decay**.

- **Gradient descent with dropouts**:

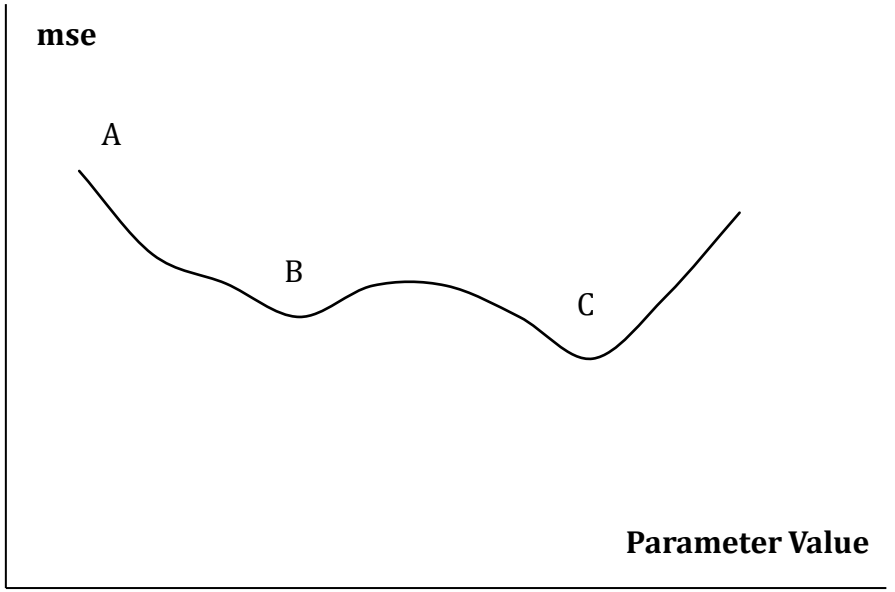
Ο αλγόριθμος εκπαίδευσης μπορεί να δοκιμάζει διαφορετικές αρχιτεκτονικές δικτύου αφαιρώντας τυχαία (δηλαδή αγνοώντας) νευρώνες κατά τη διάρκεια της εκπαίδευσης. Επιλέγεται ένα ποσοστό dropout p(συχνά χρησιμοποιείται $p = 0.5$). Σε κάθε επανάληψη, κάθε νευρώνας έχει πιθανότητα p να αγνοηθεί. Η διαδικασία αυτή συμβάλλει στην αποφυγή του overfitting που προκαλείται από αλληλεξαρτήσεις μεταξύ των νευρώνων (για παράδειγμα, όταν μία μεγάλη θετική τιμή σε έναν νευρώνα αντισταθμίζεται από μία μεγάλη αρνητική τιμή σε κάποιον άλλο νευρώνα). Στο τέλος της εκπαίδευσης, τα weights πρέπει να πολλαπλασιαστούν με το ποσοστό dropout p, ώστε να ληφθεί υπόψη ότι η προσωρινή αφαίρεση ορισμένων νευρώνων κατά τη διάρκεια της εκπαίδευσης οδήγησε σε μεγαλύτερες εκτιμήσεις για τα υπόλοιπα weights.

Local Minima

Είναι σημαντικό να προσπαθήσετε να αποφύγετε τα τοπικά ελάχιστα

Η επιφάνεια πάνω στην οποία αναζητά λύση ο αλγόριθμος gradient descent σπάνια είναι ομαλή. Συχνά υπάρχουν τοπικά ελάχιστα (local minima). Εξετάστε, για παράδειγμα, την περίπτωση του διπλανού σχήματος. Αν ξεκινήσουμε από το σημείο A, είναι πιθανό να καταλήξουμε στο τοπικό ελάχιστο στο σημείο B, ενώ το καλύτερο (global) ελάχιστο βρίσκεται στο σημείο C.

Αποδεικνύεται ότι ορισμένες από τις παραλλαγές της βασικής μεθόδου που μόλις αναφέρθηκαν βοηθούν στην αποφυγή των τοπικών ελαχίστων. Αν αρχικά επιλεγεί ένα μεγάλο learning rate, ο αλγόριθμος μπορεί να «πηδήξει πάνω από» ορισμένα τοπικά ελάχιστα. Αν χρησιμοποιούνται mini-batches, ο αλγόριθμος εξερευνά περισσότερο τον χώρο αντί να κατευθύνεται άμεσα προς ένα ελάχιστο. Σε ορισμένες περιπτώσεις, το learning rate αυξάνεται περιοδικά ώστε να ενθαρρύνεται ο αλγόριθμος να «ξεφεύγει» από τοπικά ελάχιστα.



Stopping Rule

- Επειδή οι εφαρμογές έχουν πολλές παραμέτρους, είναι σημαντικό να χρησιμοποιήσετε έναν κανόνα διακοπής για να αποφύγετε over-fitting.
- **Υπολογίζουμε το cost function για το validation set ταυτόχρονα με το training set,** μετά από κάθε epoch εκπαίδευσης
- **Όταν το cost function για το validation set αρχίζει να χειροτερεύει, σταματάμε**.
- Στη συνέχεια, ο αλγόριθμος επιστρέφει στο μοντέλο που παρήγαγε τη χαμηλότερη τιμή της συνάρτησης κόστους για το validation set.
- Καθώς η εκπαίδευση προχωρά, το λογισμικό του νευρωνικού δικτύου πρέπει επομένως να αποθηκεύει τα weights και τα biases που έχουν οδηγήσει στη μικρότερη τιμή της συνάρτησης κόστους μέχρι εκείνη τη στιγμή για το validation set.

- Όταν η συνάρτηση κόστους του validation set δεν παρουσιάζει βελτίωση για έναν συγκεκριμένο αριθμό epochs (ο οποίος ορίζεται από τον χρήστη), ο αλγόριθμος σταματά.
- Στο TensorFlow, ο συγκεκριμένος αριθμός epochs αναφέρεται ως **“patience”.** Για παράδειγμα, αν το patience οριστεί ίσο με 500, τότε το νευρωνικό δίκτυο σταματά όταν δεν έχει υπάρξει βελτίωση στη συνάρτηση κόστους του validation set για 500 διαδοχικά epochs και το μοντέλο που χρησιμοποιείται είναι εκείνο που είχε δημιουργηθεί 500 epochs νωρίτερα.