ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Ενότητα 9 – Supervised Learning
Neural Networks

Μέθοδοι Μηχανικής Μάθησης στα Χρηματοοικονομικά

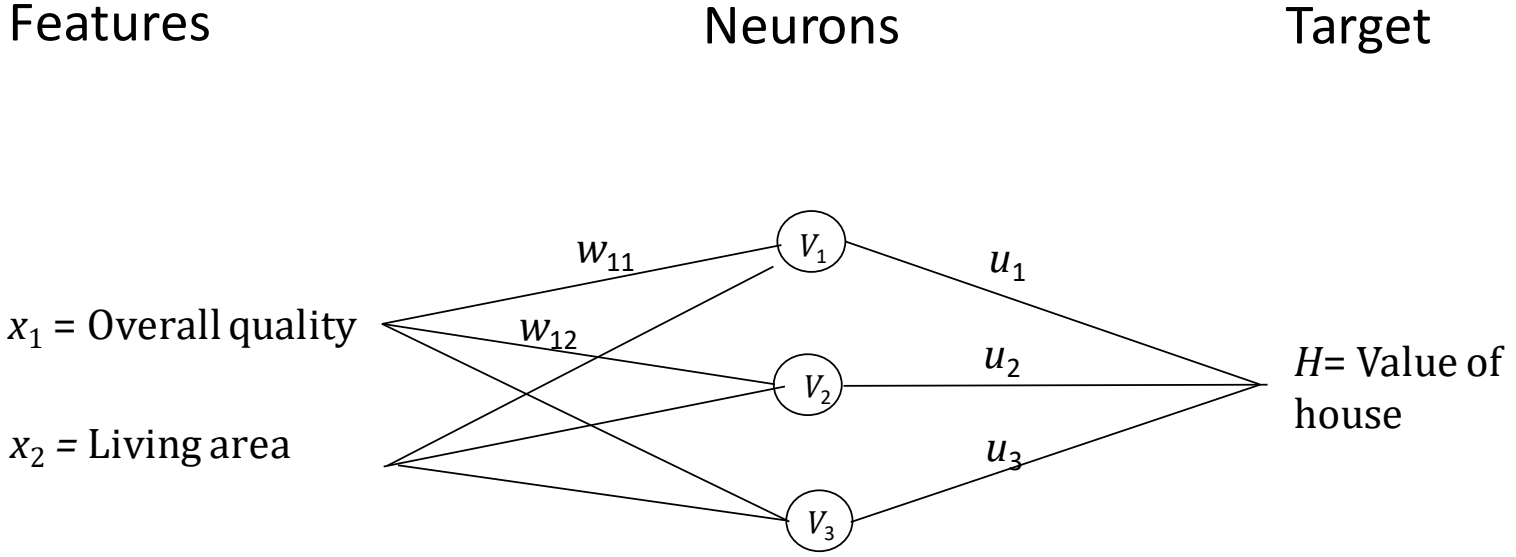
Αθανάσιος Σάκκας, Επ. Καθηγητής, ΟΠΑ

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

- *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

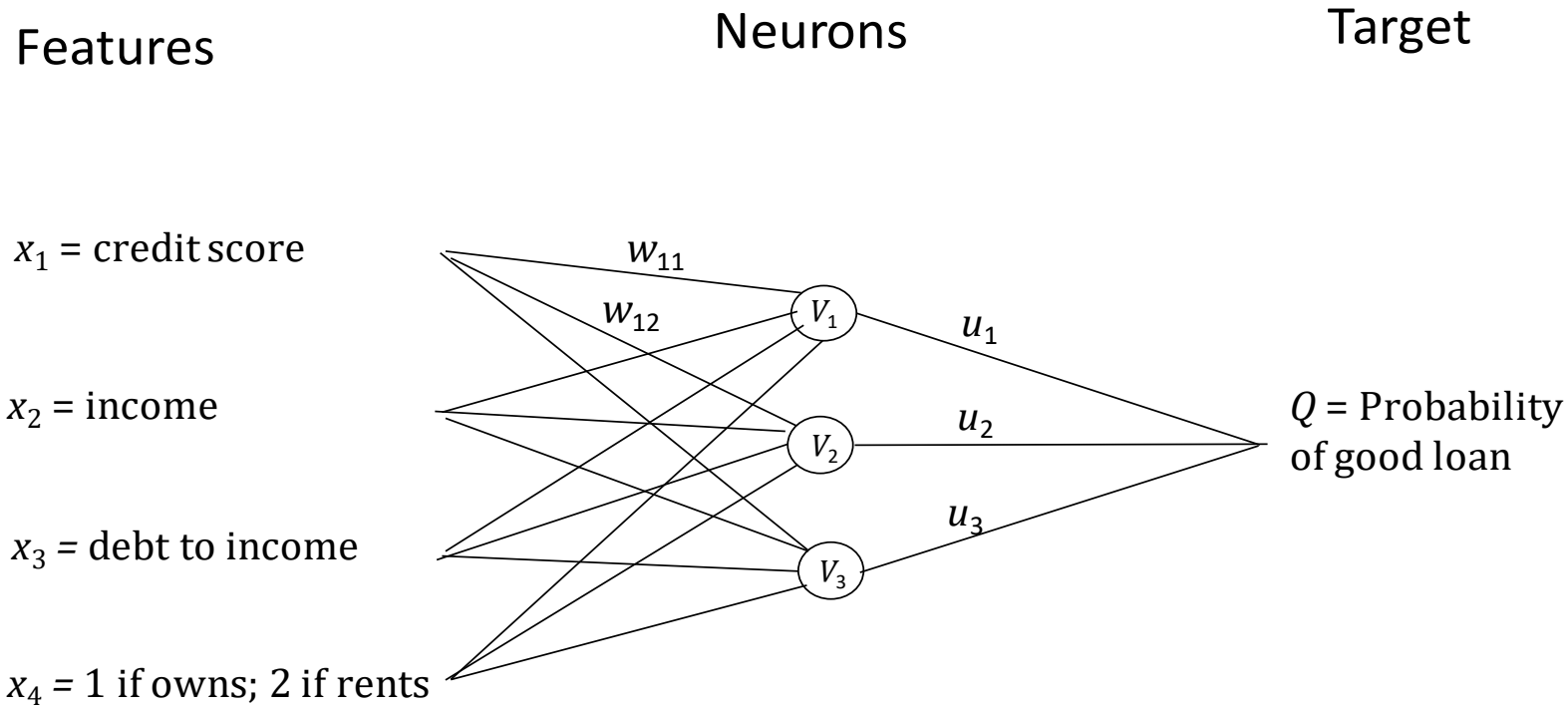
β. Προσθέτουμε έναν σταθερό όρο (bias).

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

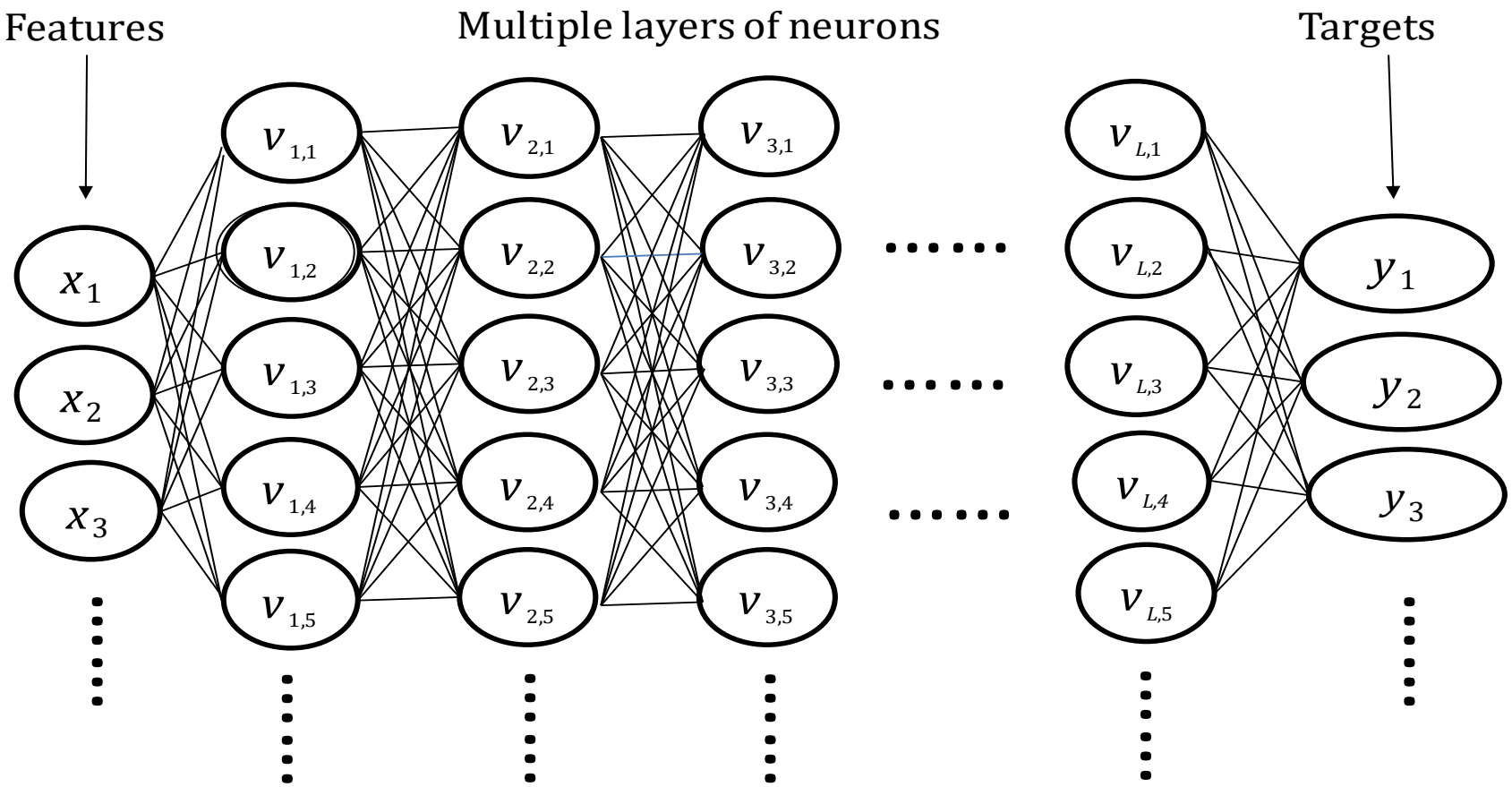
$$V_k = f(a_k + w_{1k}x_1 + w_{2k}x_2)$$

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

Παράδειγμα του Lending Club με 4 χαρακτηριστικά



A General Neural Network



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)

- 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 (3)

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.

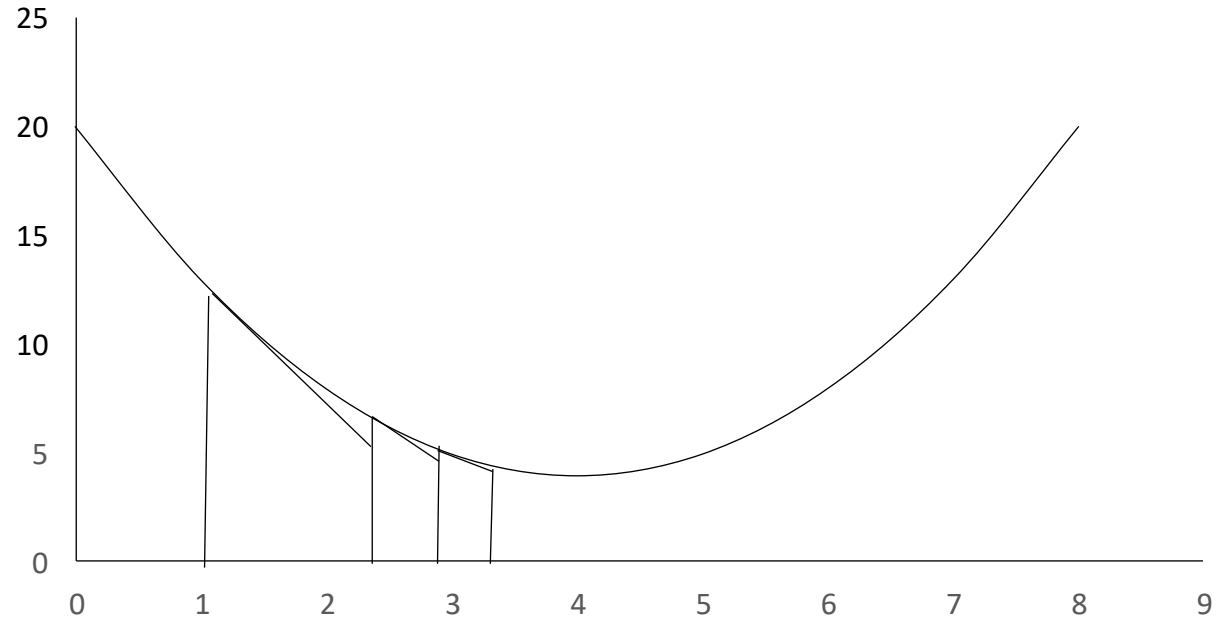
Gradient Descent και NN

- 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:

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

Δείτε το παράδειγμα *9. Salary -Age example Gradient Descent*

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

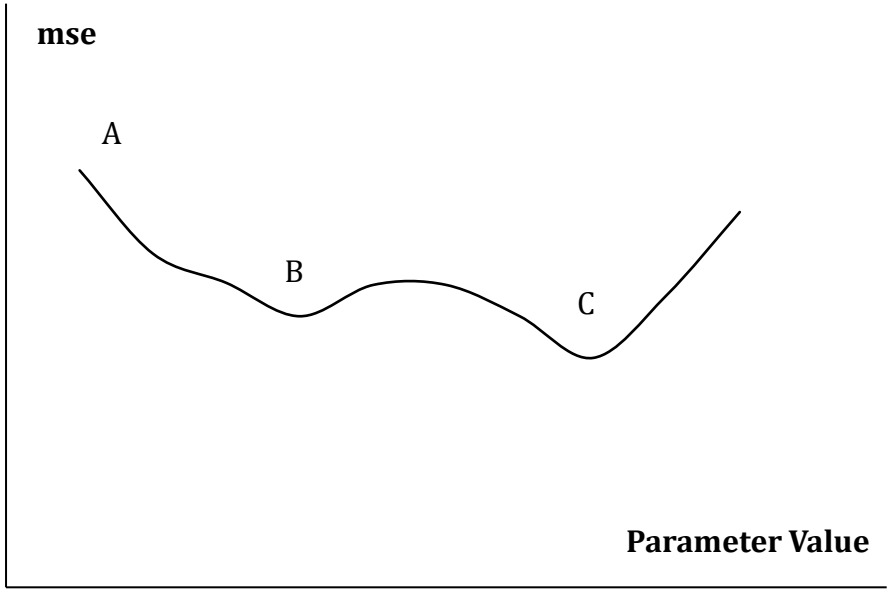
Iteration	x	Gradient	Change in x
0	1.000	-6.000	0.120
1	1.120	-5.760	0.115
2	1.235	-5.530	0.111
3	1.346	-5.308	0.106
4	1.452	-5.096	0.102
5	1.554	-4.892	0.098
6	1.652	-4.697	0.094
7	1.746	-4.509	0.090
8	1.836	-4.328	0.087
9	1.922	-4.155	0.083
10	2.006	-3.989	0.080
11	2.085	-3.829	0.077
12	2.162	-3.676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

Local Minima

Είναι σημαντικό να προσπαθήσετε να αποφύγετε τα 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.

- Όταν το cost function για το validation set αρχίζει να χειροτερεύει, σταματάμε.