

Νευρωνικά Δίκτυα 2012-13

Εργαστήριο Εικόνας, Βίντεο και Πολυμέσων

12-11-2012

Τάσος Βενέτης

avenet@image.ntua.gr

Ηλιάννα Κόλλια

ilianna1@gmail.com

Ελένη Τσαλαπάτη

etsalap@image.ntua.gr

Μέρος 1^ο

Εισαγωγή στη MATLAB

Πίνακες και MATLAB

- MATLAB = MATrix LABoratory
- Αποτελεί ισχυρό υπολογιστικό περιβάλλον και ταυτόχρονα γλώσσα προγραμματισμού που χειρίζεται με ευκολία πίνακες και πολύπλοκες πράξεις
- Βασική Δομική Μονάδα: Πίνακες (1-Δ, 2-Δ, 3-Δ,...)
- Κάθε μεταβλητή στη MATLAB αποθηκεύεται σαν ένας πίνακας 1x1
- Η MATLAB είναι case-sensitive, άρα $a \neq A$
- Το στοιχείο στη γραμμή i και στη στήλη j του πίνακα A συμβολίζεται με $A(i,j)$

Πίνακες και MATLAB

- Δημιουργία Πινάκων:
 - Ορισμός όλων των στοιχείων του πίνακα:
 - Αναλυτικά: $A=[1\ 2\ 3\ 4]$, $B=[1\ 2;3\ 4]$
 - Περιγραφικά: $C=[1:10]$, $D = [0:5:100]$
 - Φόρτωμα πίνακα από αρχείο:
 - `load myMatrix.mat`
 - Δημιουργία πίνακα χρησιμοποιώντας μια από τις έτοιμες συναρτήσεις της MATLAB:
 - $E=\text{zeros}(3,4)$, $F=\text{ones}(5,6)$, $G=\text{eye}(10)$
 - $H=\text{rand}(5,2)$
 - Δημιουργία πίνακα με χρήση δικής μας συνάρτησης:
 - $I=\text{myFunction}(E,F)$

Χρήσιμες πράξεις μεταξύ πινάκων

- έστω: $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ $B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$
- τότε: $A + B = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$ $A - B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
 $A * B = \begin{pmatrix} 7 & 10 \\ 15 & 22 \end{pmatrix}$ $A. * B = \begin{pmatrix} 1 & 4 \\ 9 & 16 \end{pmatrix}$
 $A. / B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ $A.^ B = \begin{pmatrix} 1 & 4 \\ 27 & 256 \end{pmatrix}$
 $A' = B' = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$

Χρήσιμες Πράξεις μεταξύ Πινάκων

- Πράξεις με συναρτήσεις:

$$A=[1 \ 2 \ 3] \rightarrow \sin(A) = (0.8415 \quad 0.9093 \quad 0.1411)$$

$$A=[1 \ 2 \ 3] \rightarrow \exp(A) = (2.7183 \quad 7.3891 \quad 20.0855)$$

- Λογικές πράξεις:

$$A = [1 \ 2 \ 1], B = [0 \ 3 \ 1] \rightarrow$$

$$A > B = (1 \quad 0 \quad 0), A \geq B = (1 \quad 0 \quad 1)$$

$$A < B = (0 \quad 1 \quad 0), A \leq B = (0 \quad 1 \quad 1)$$

$$A == B = (0 \quad 0 \quad 1)$$

- Πράξεις Μέγιστου/Ελάχιστου:

$$A = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix} \rightarrow \max(A) = (4 \ 3), \max(\max(A)) = 4$$

Αλληλουχία Πινάκων

- έστω: $A = [1 \ 2 \ 3 \ 4]$
- τότε: $B = [A \ A] \rightarrow B = (1 \ 2 \ 3 \ 4 \ 1 \ 2 \ 3 \ 4)$

$$B = [A; A] \rightarrow B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

Ελεγχος Ροής

- **if** statement **else** statement **end**
- **if** statement **elseif** statement **elseif** ... **else** statement **end**
- **while** statement **end**
- **for** statement **end**
- **switch** switch_expression
 - case** case_expression(s)
statement(s)
 - case** case_expression(s)
statement(s)
 - ...
 - otherwise**
statement(s)**end**
- **break**
- Όχι σε άσκοπη χρήση **for** loops!!!

M-Files

- Δύο τρόποι για τη δημιουργία m-files:
 - File→New→M-File
 - `edit file1.m` [σε command line]
- Ένα M-File μπορεί να είναι:
 - Μια συνάρτηση (function)
 - `function C = myFunction(A,B)`
`C = A + 2*B;`
 - Μια ακολουθία εντολών (script)
 - `A, B`
`C = A + 2*B`

Σχεδίαση Γραφικής Παράστασης με τη χρήση της συνάρτησης `plot`

- Η `plot(x,y)` δέχεται ως είσοδο δύο διανύσματα και εμφανίζει μια γραφική παράσταση του y σε συνάρτηση του x

Παράδειγμα

- Θέλουμε να σχεδιάσουμε το $\sin(x)$ για τιμές του x από 0 έως 2π :

```
t = 0:pi/100:2*pi;
```

```
y = sin(t);
```

```
plot(t,y)
```

Μέρος 2^ο

Multilayer Perceptrons

Μάθηση Νευρωνικών Δικτύων

- Επίτευξη επιθυμητής συμπεριφοράς μέσω ανανέωσης τιμών των συναπτικών βαρών
- Διάφοροι αλγόριθμοι μάθησης
 - Επιβλεπόμενη
 - Ενισχυτική
 - Μη επιβλεπόμενη

Επιβλεπόμενη Μάθηση

- Δάσκαλος – Σύστημα μάθησης

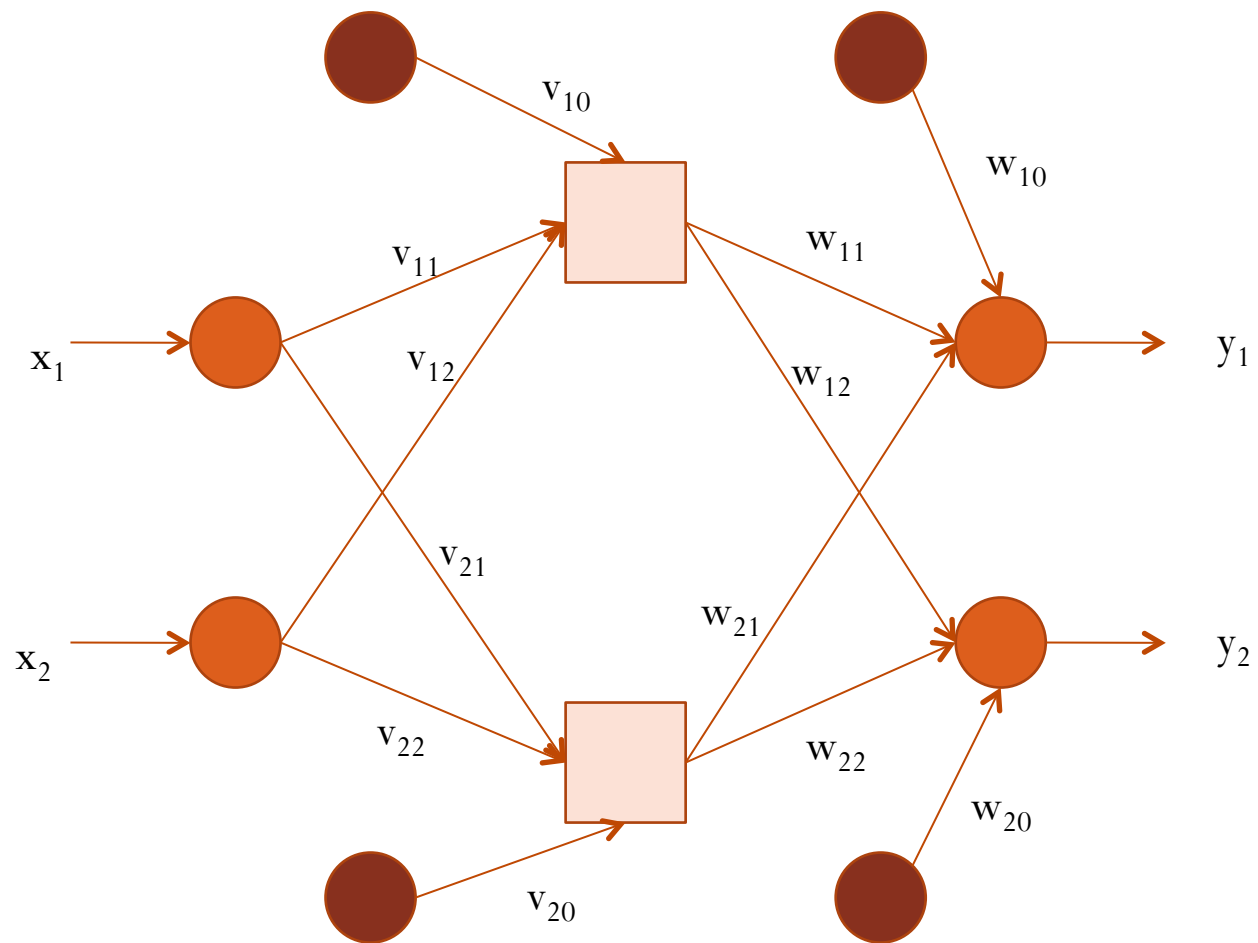
Πολυεπίπεδα Perceptrons (1)

- Γενικά:
 - Ενα από τα πιο συνηθισμένα Νευρωνικά Δίκτυα
 - Δίκτυο πρόσθιας τροφοδότησης (feed forward)
 - Γενίκευση του μονοστρωματικού perceptron
 - Επιβλεπόμενη(supervised) μάθηση
 - Αρχιτεκτονική:
 - 1 επίπεδο εισόδου
 - 1 ή περισσότερα κρυμμένα επίπεδα
 - 1 επίπεδο εξόδου
 - Εκπαίδευση με τον αλγόριθμο ανάστροφης διάδοσης σφάλματος (Back Propagation)

Πολυεπίπεδα Perceptrons (2)

- Ενα πολυεπίπεδο perceptron έχει τα εξής χαρακτηριστικά:
 - Κάθε κρυμμένος νευρώνας περιέχει μια ΜΗ-ΓΡΑΜΜΙΚΗ συνάρτηση ενεργοποίησης (activation function)
 - Η συνάρτηση αυτή είναι ΔΙΑΦΟΡΙΣΙΜΗ
 - Τα κρυμμένα επίπεδα προσδίδουν στο δίκτυο την δυνατότητα να «μάθει» πολύπλοκα πρότυπα
 - Δεν επιτρέπονται συνδέσεις μεταξύ επιπέδων που δεν ανήκουν σε διαδοχικά επίπεδα
 - Δεν επιτρέπονται συνδέσεις μεταξύ νευρώνων του ίδιου επιπέδου

Παράδειγμα



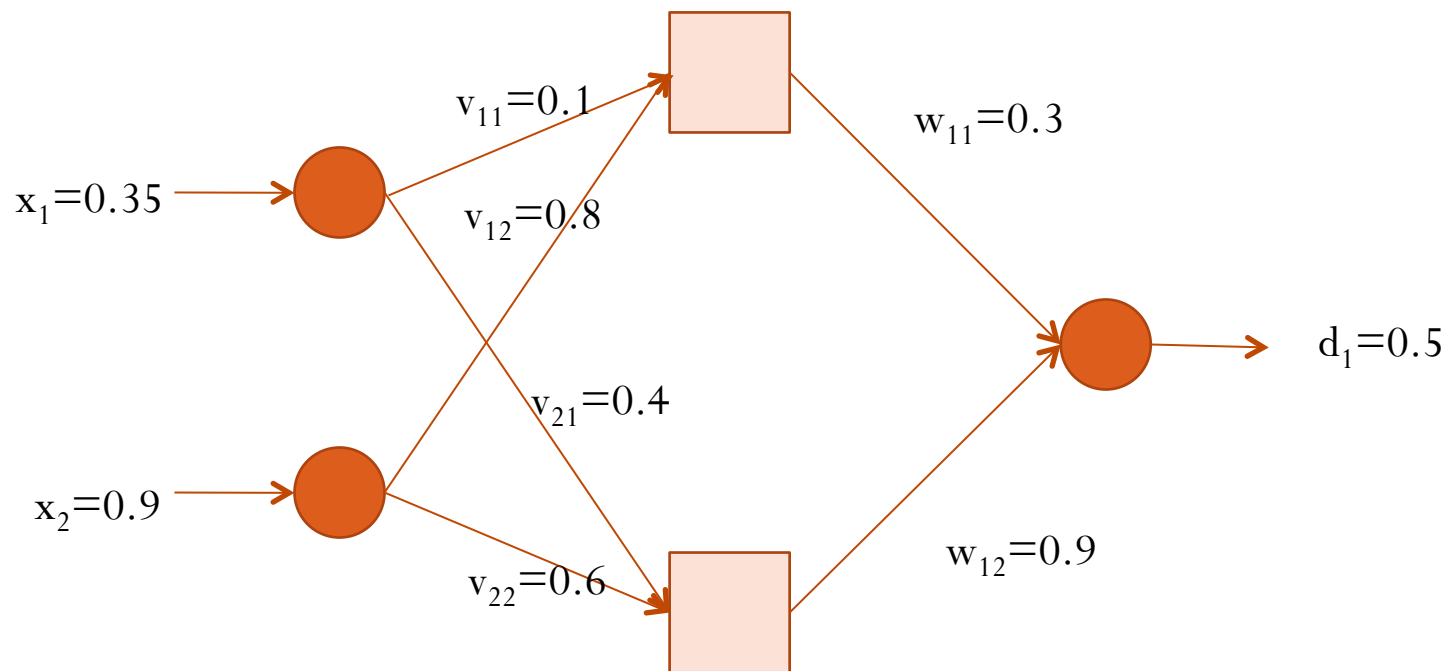
Back Propagation

- Αλγόριθμος Επιβλεπόμενης Μάθησης
- Διαδικασία Εκπαίδευσης
 - Αρχικοποιούμε τυχαία τα βάρη
 - Δίνουμε την είσοδο
 - Υπολογίζουμε την έξοδο
 - Προσαρμόζουμε τα συναπτικά βάρη ώστε να ελαχιστοποιηθεί το μέσο τετραγωνικό σφάλμα σε σχέση με την επιθυμητή έξοδο
- Τερματισμός Εκπαίδευσης
 - Μικρό σφάλμα
 - Αριθμός εποχών

Γενίκευση

- Ικανότητα ταξινόμησης προτύπων που δεν έχουν παρουσιαστεί ποτέ
- Όταν το δίκτυο «υπερεκπαιδεύει» χάνει την ικανότητα για γενίκευση
- Early Stopping
 - Δεδομένα εκπαίδευσης
 - Δεδομένα επαλήθευσης
 - Δεδομένα ελέγχου
 - Εκπαίδευση μόνο με τα δεδομένα εκπαίδευσης
 - Υπολογισμός σφάλματος για δεδομένα επαλήθευσης
 - Όταν το σφάλμα αρχίσει να αυξάνεται σταματάμε

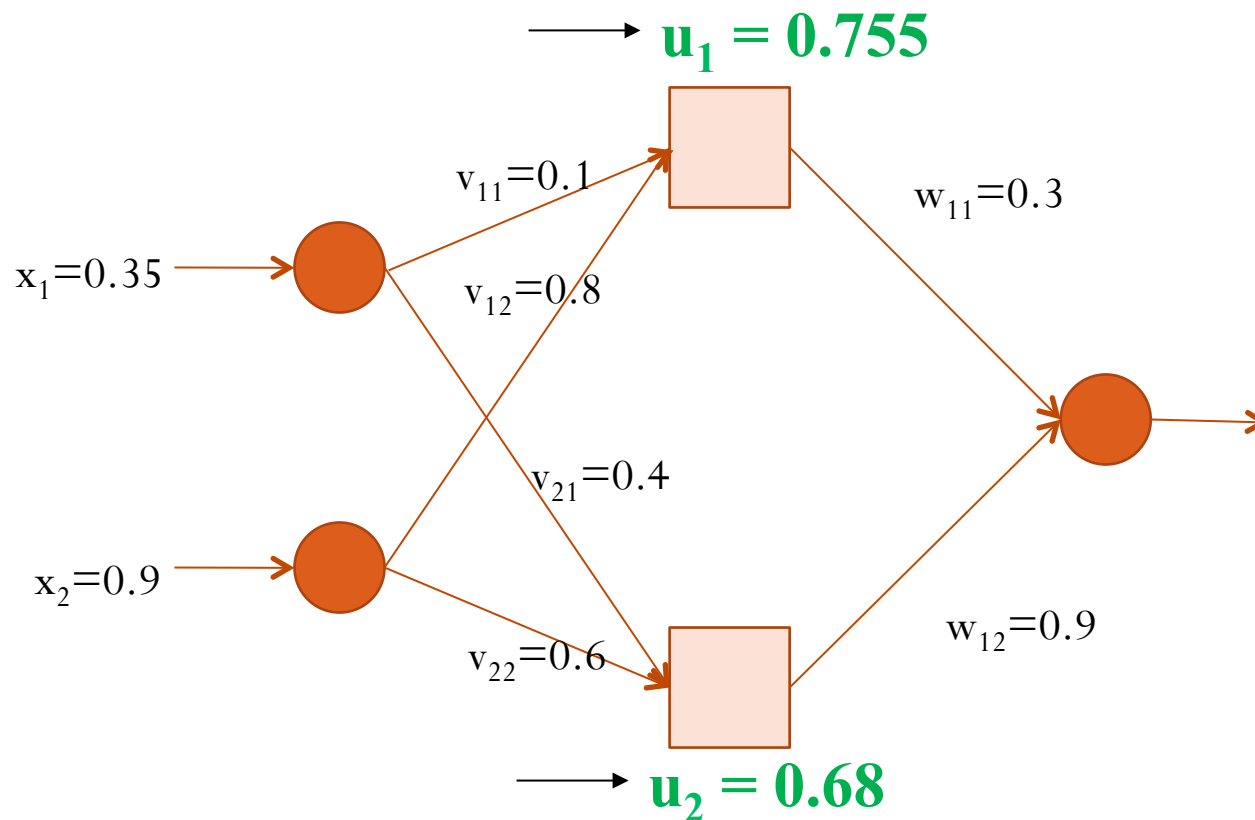
Παράδειγμα



Είσοδος: $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.35 \\ 0.9 \end{pmatrix}$ Επιθυμητή Εξοδος: $(d_1) = (0.5)$

Παράδειγμα

$$u_k = \sum_i w_{ki} \cdot x_i$$



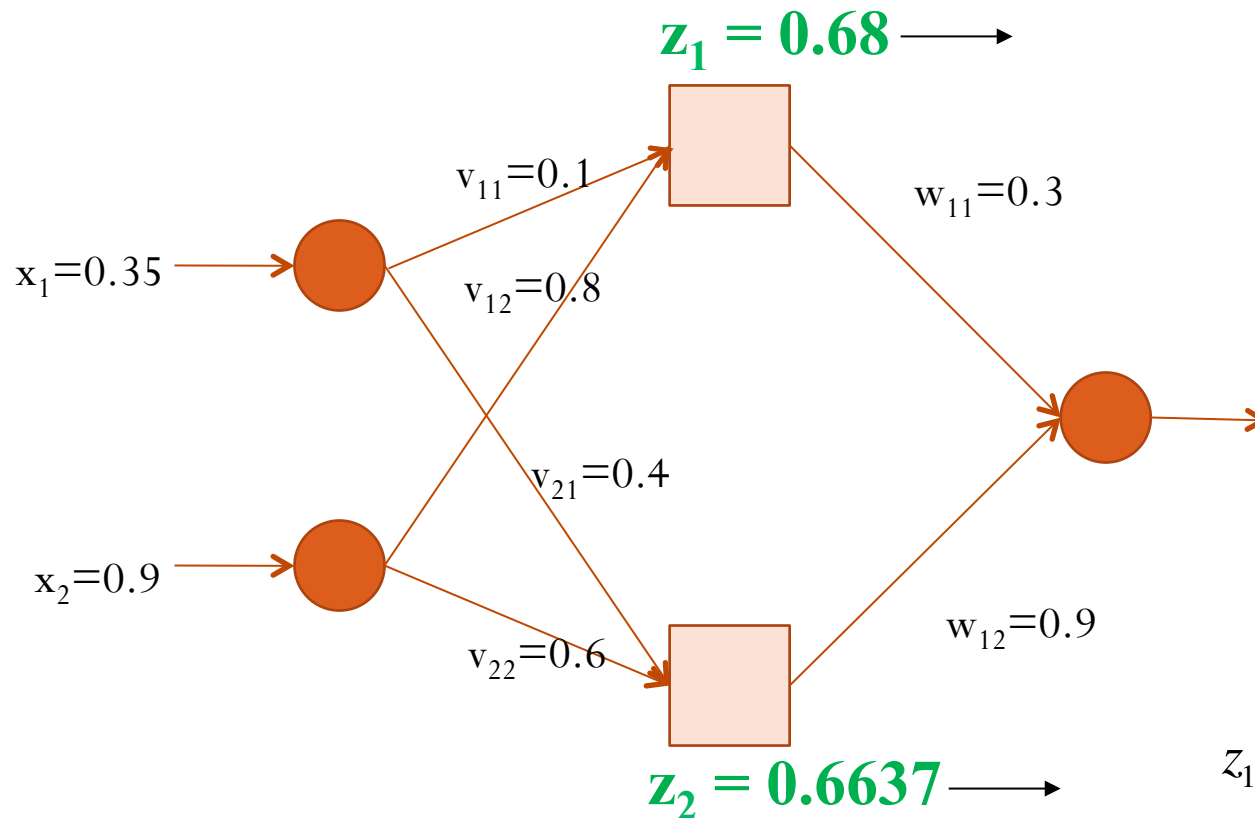
- Πέρασμα κατά την **ΕΥΘΕΙΑ** φορά
- Υπολογίζουμε τις ενεργοποιήσεις 1^{ου} επιπέδου

$$u_1 = v_{11} \cdot x_1 + v_{12} \cdot x_2 = 0.1 \cdot 0.35 + 0.8 \cdot 0.9 = 0.755$$

$$u_2 = v_{21} \cdot x_1 + v_{22} \cdot x_2 = 0.4 \cdot 0.35 + 0.6 \cdot 0.9 = 0.68$$

Παράδειγμα

$$y_k = f(u_k) = \frac{1}{1 + e^{-u_k}}$$



- Υπολογίζουμε τις εξόδους του πρώτου επιπέδου, περνώντας τις ενεργοποιήσεις που υπολογίσαμε προηγουμένως στη συνάρτηση ενεργοποίησης:

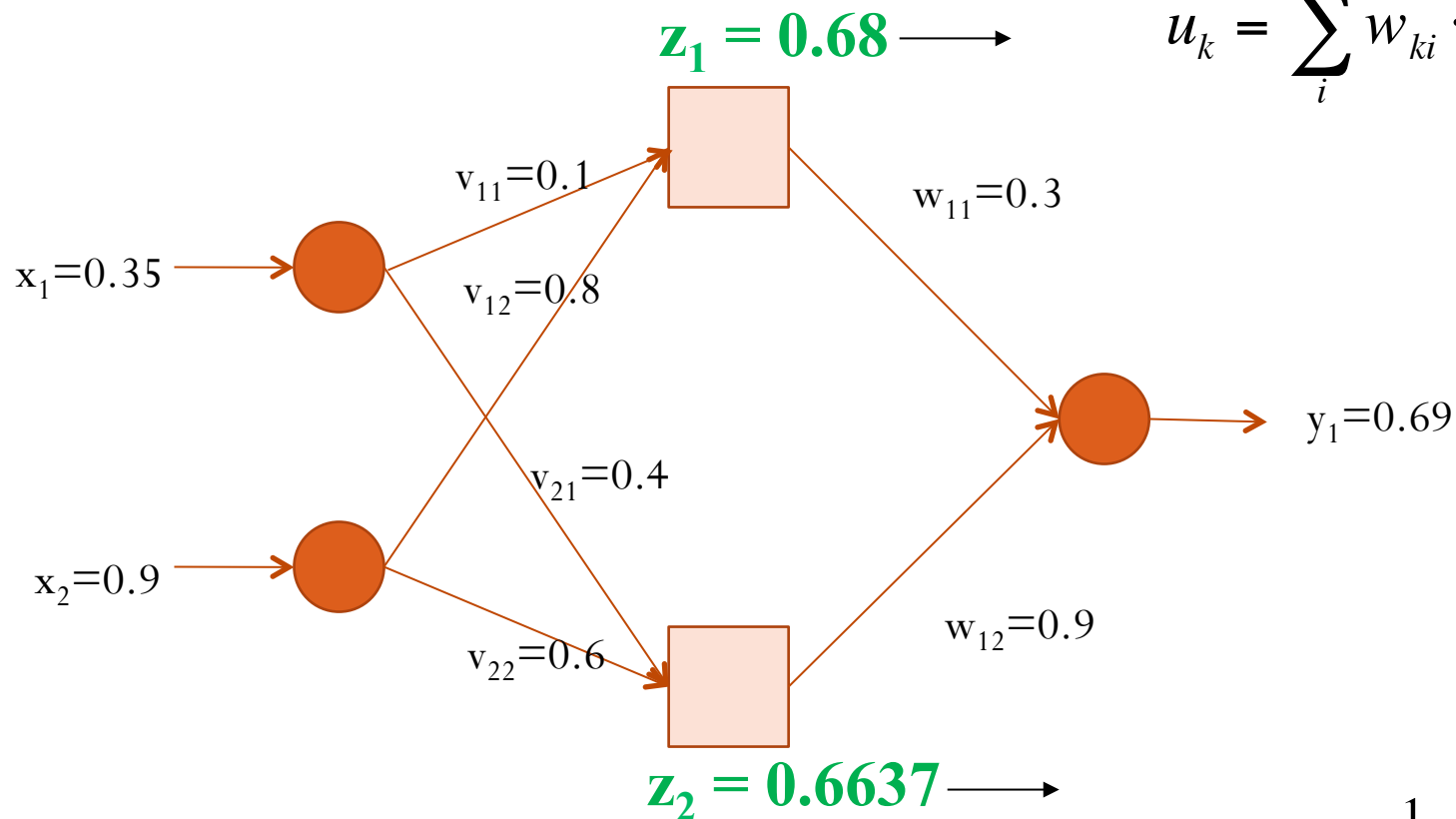
$$z_1 = \frac{1}{1 + e^{-0.755}} = 0.68$$

$$z_2 = \frac{1}{1 + e^{-0.68}} = 0.6637$$

Παράδειγμα

$$y_k = f(u_k) = \frac{1}{1 + e^{-u_k}}$$

$$u_k = \sum_i w_{ki} \cdot x_i$$

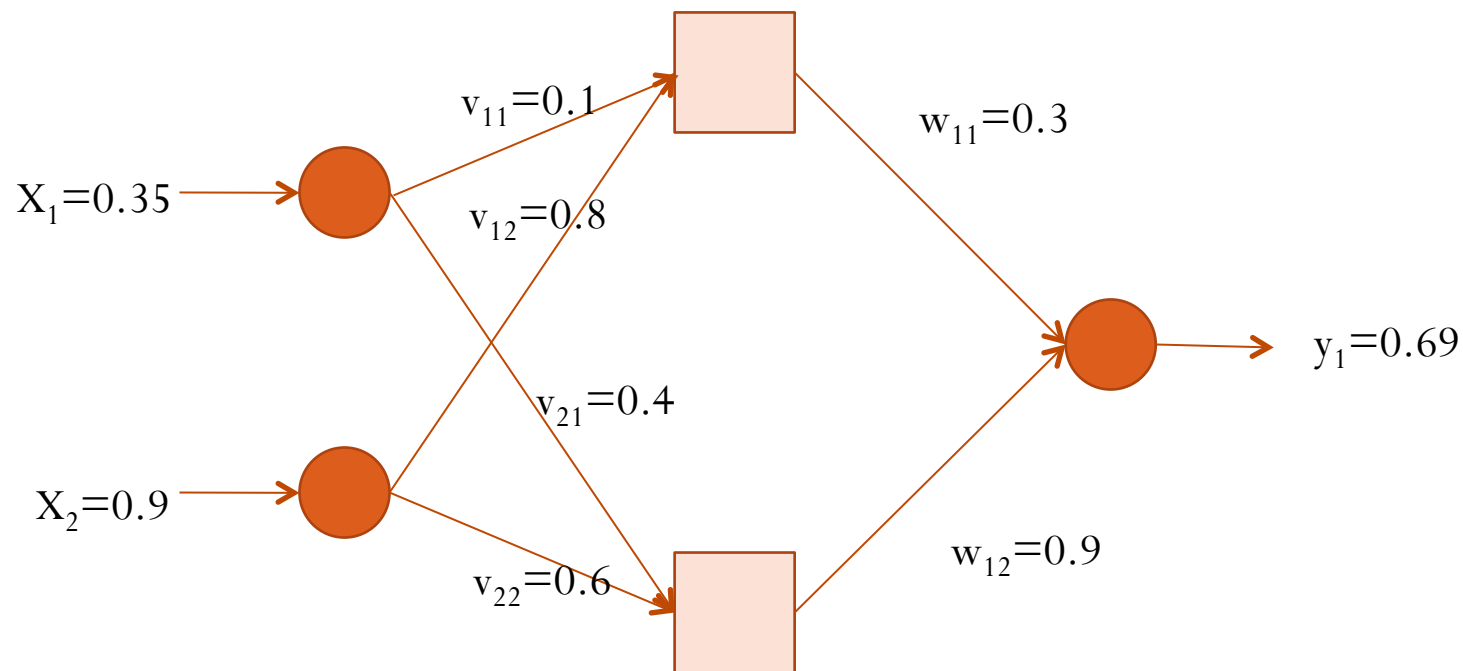


- Υπολογίζουμε την έξοδο του δευτέρου επιπέδου:

$$y_1 = \frac{1}{1 + e^{-0.8013}} = 0.69$$

Παράδειγμα

✖ The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and



- Υπολογίζουμε το σφάλμα στην έξοδο:

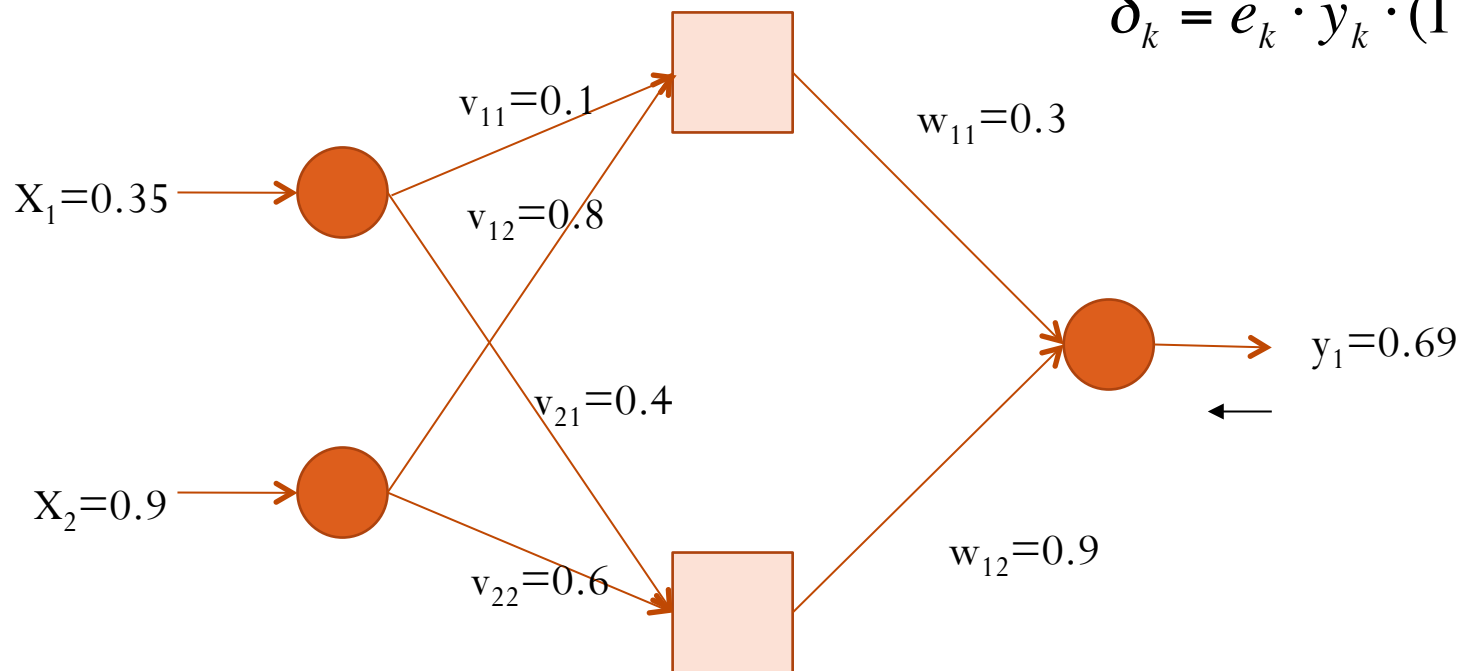
✖ The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

Παράδειγμα

$$e_k = d_k - y_k$$

$$\delta_k = e_k \cdot f'(u_k)$$

$$\delta_k = e_k \cdot y_k \cdot (1 - y_k)$$



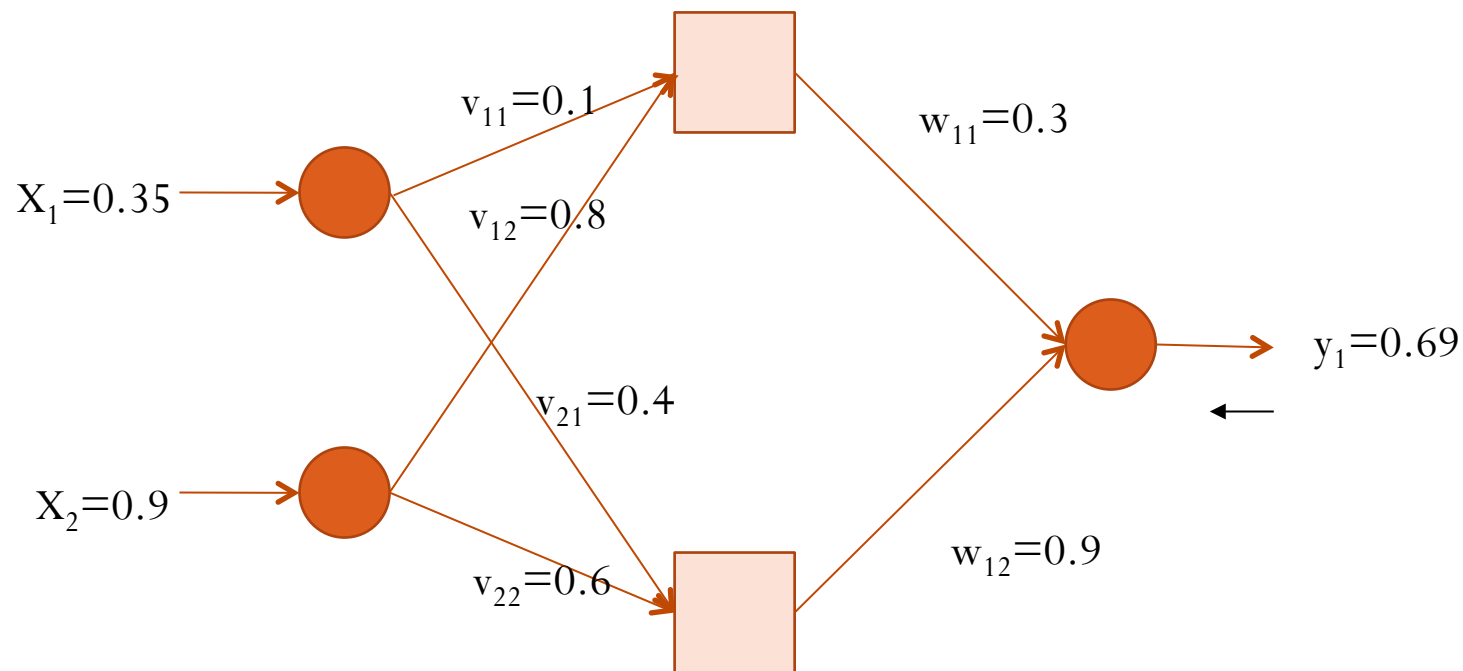
- Υπολογίζουμε το σφάλμα στην έξοδο και την τοπική κλίση:

$$e_1 = d_1 - y_1 = 0.5 - 0.69 = -0.19$$

$$\delta_1 = -0.19 \cdot 0.69 \cdot (1 - 0.69) = -0.0406$$

Παράδειγμα

$$\Delta w_{ki} = \gamma \cdot \delta_k \cdot y_i$$

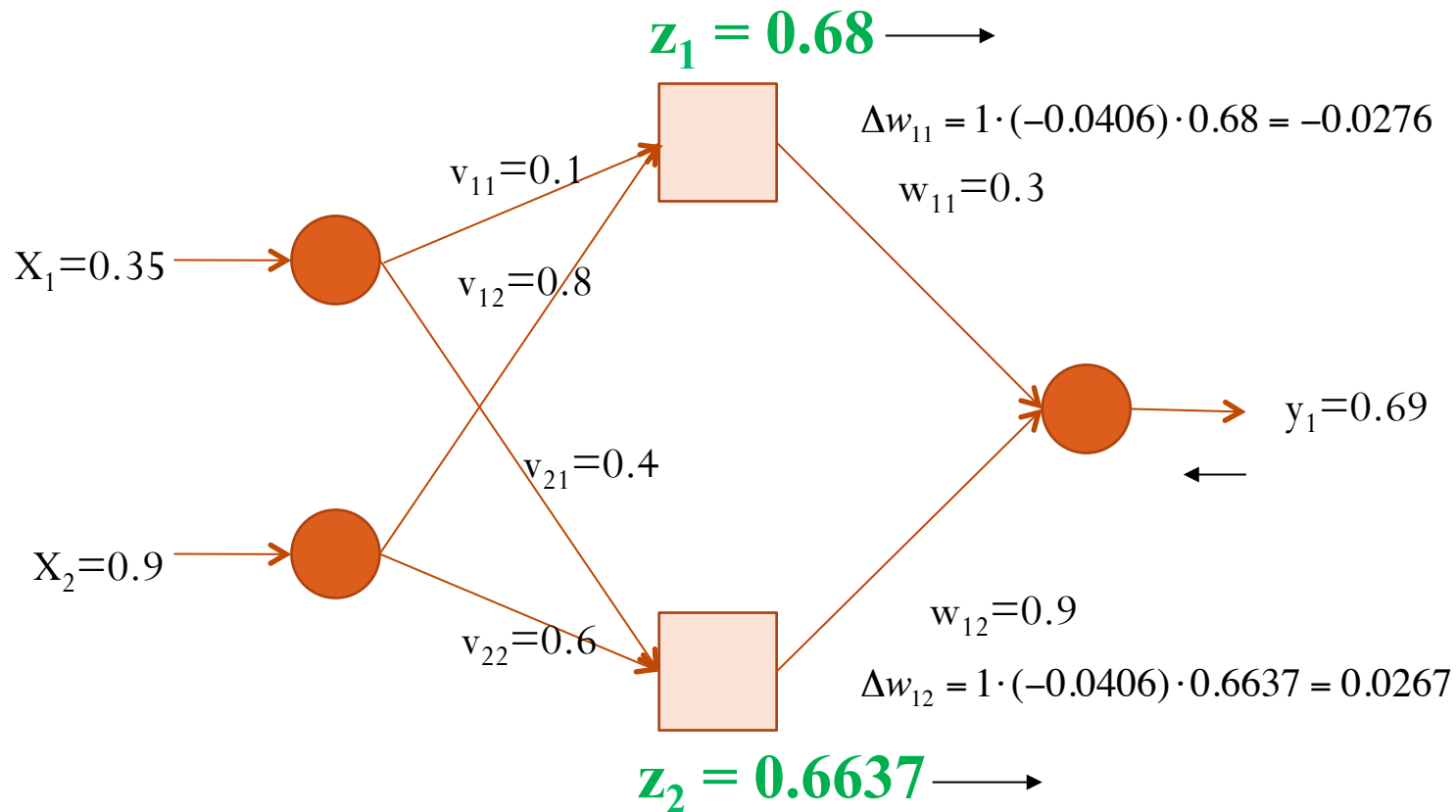


- Υπολογίζουμε το : Δw_{ki}

$$\delta_1 = -0.0406$$

Παράδειγμα

$$\Delta w_{ki} = \gamma \cdot \delta_k \cdot y_i$$

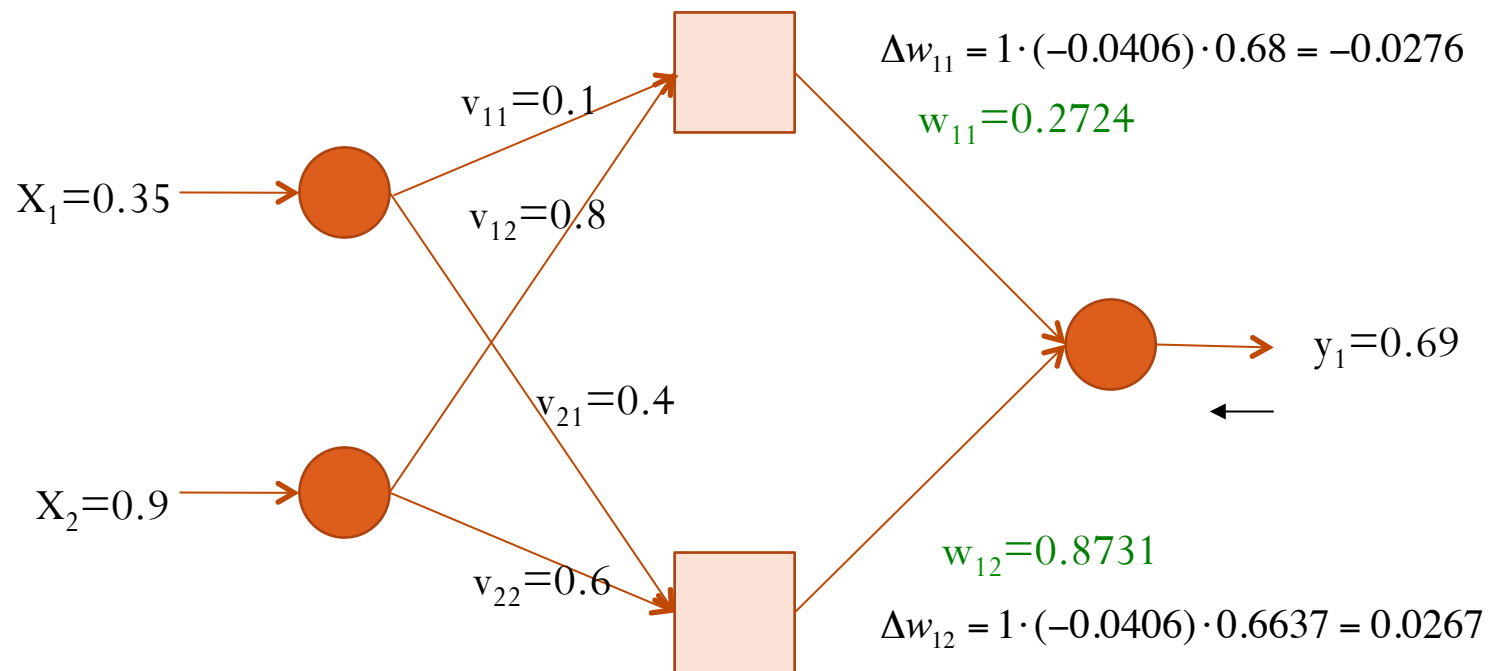


- Υπολογίζουμε το : Δw_{ki}

$$\delta_1 = -0.0406$$

Παράδειγμα

$$\Delta w_{ki} = \gamma \cdot \delta_k \cdot y_i$$



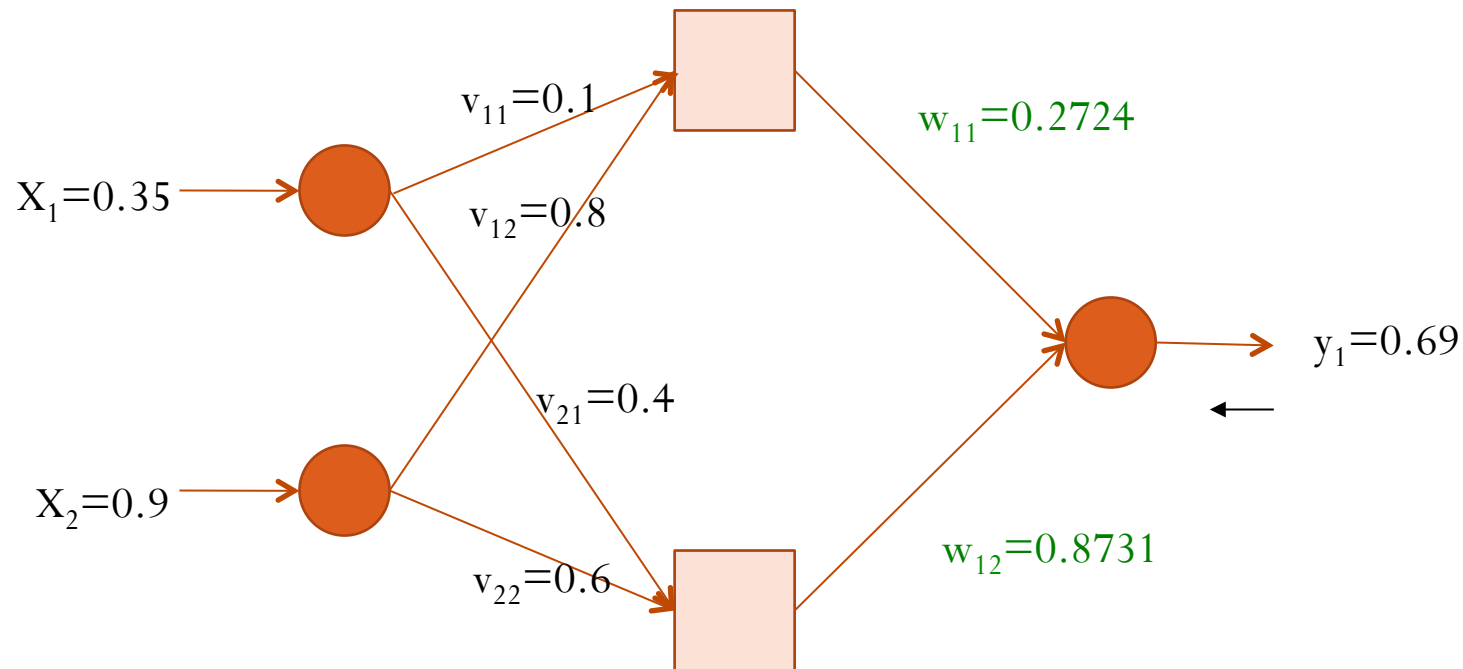
- Ανανεώνουμε τα βάρη

$$\delta_1 = -0.406$$

Παράδειγμα

$$\Delta w_{ji} = \gamma \cdot \delta_j \cdot y_i$$

$$\delta_j = y_j \cdot (1 - y_j) \cdot \sum_m \delta_m \cdot w_{mj}$$



- Υπολογίζουμε το σφάλμα προς τα πίσω

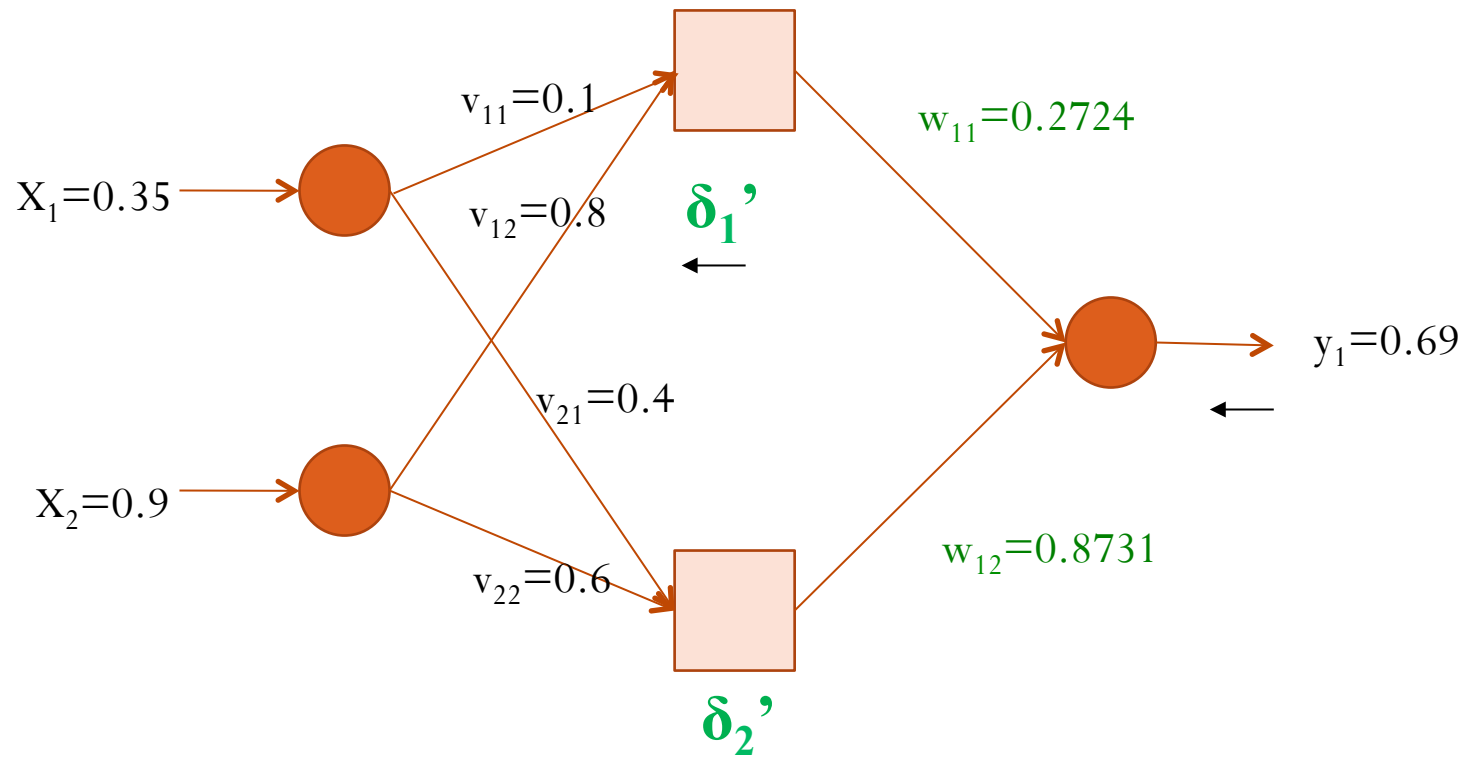
Παράδειγμα



The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

$$\Delta w_{ji} = \gamma \cdot \delta_j \cdot y_i$$

$$\delta_j = y_j \cdot (1 - y_j) \cdot \sum_m \delta_m \cdot w_{mj}$$



The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

Παράδειγμα

$$\Delta v_{11} = 1 \cdot (-0.0024) \cdot 0.35 = -0.00084$$

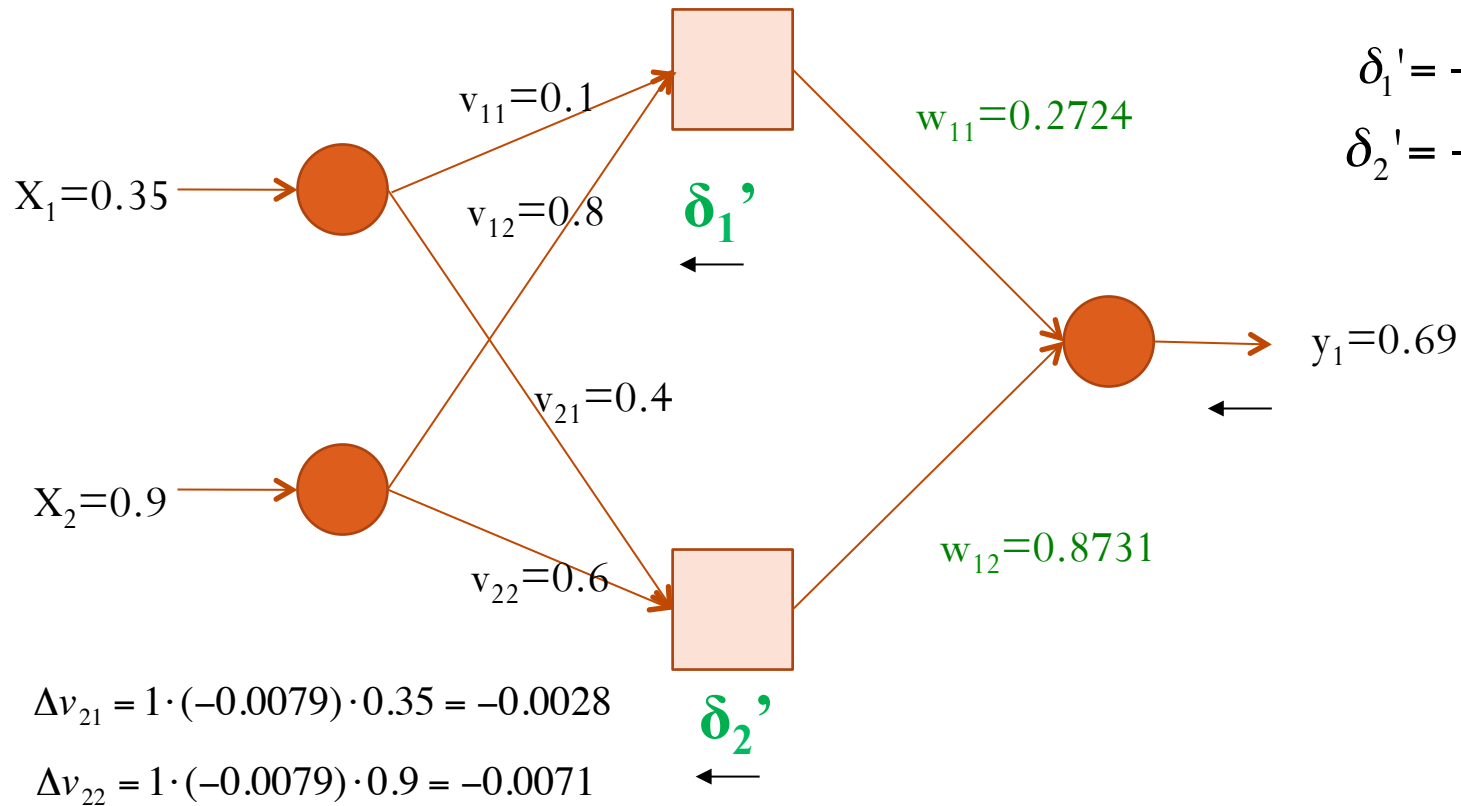
$$\Delta v_{12} = 1 \cdot (-0.0024) \cdot 0.9 = -0.0022$$

$$\Delta w_{ji} = \gamma \cdot \delta_j \cdot y_i$$

$$\delta_j = y_j \cdot (1 - y_j) \cdot \sum_m \delta_m \cdot w_{mj}$$

$$\delta_1' = -0.0024$$

$$\delta_2' = -0.0079$$



Παράδειγμα

$$\Delta v_{11} = 1 \cdot (-0.0024) \cdot 0.35 = -0.00084$$

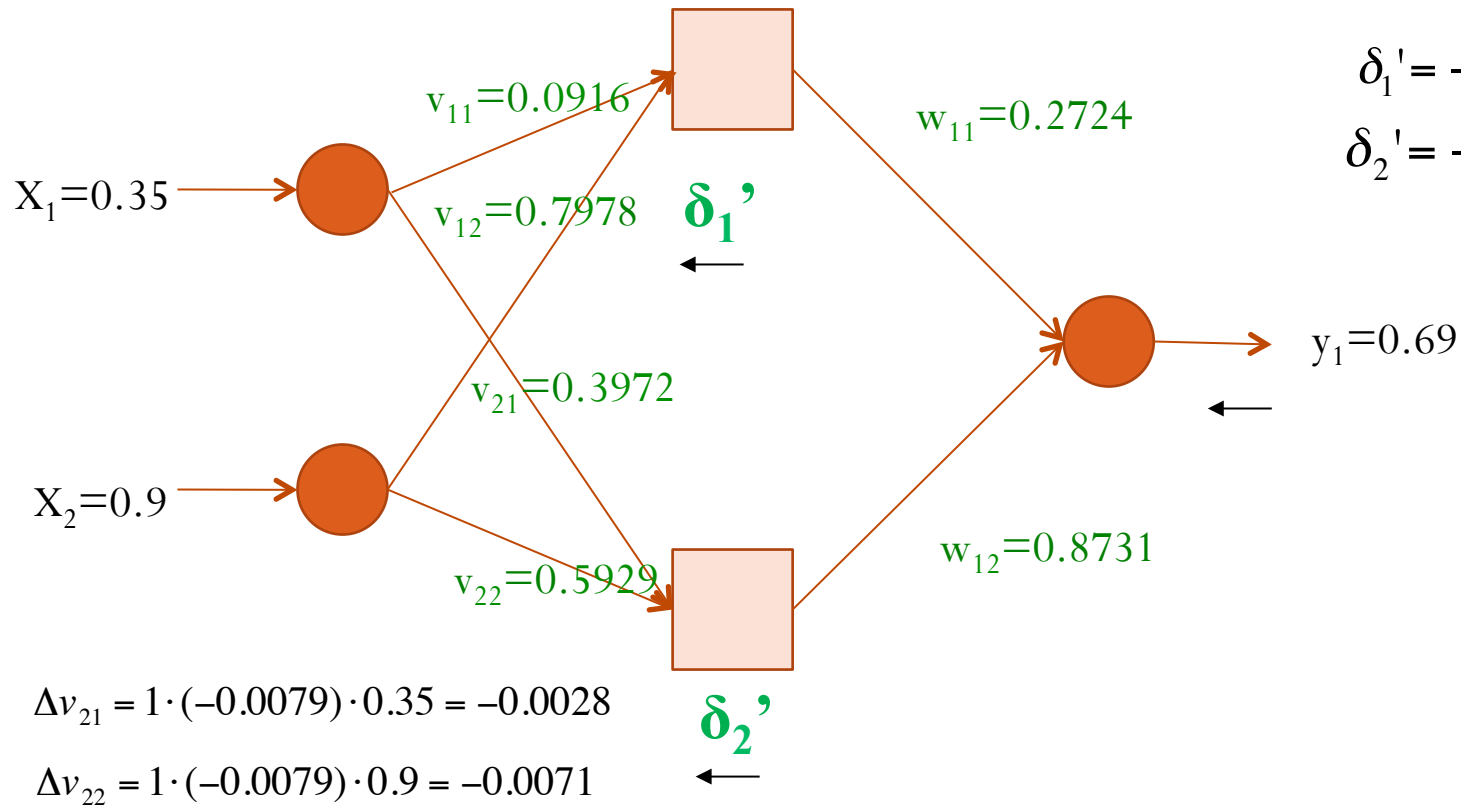
$$\Delta v_{12} = 1 \cdot (-0.0024) \cdot 0.9 = -0.0022$$

$$\Delta w_{ji} = \gamma \cdot \delta_j \cdot y_i$$

$$\delta_j = y_j \cdot (1 - y_j) \cdot \sum_m \delta_m \cdot w_{mj}$$

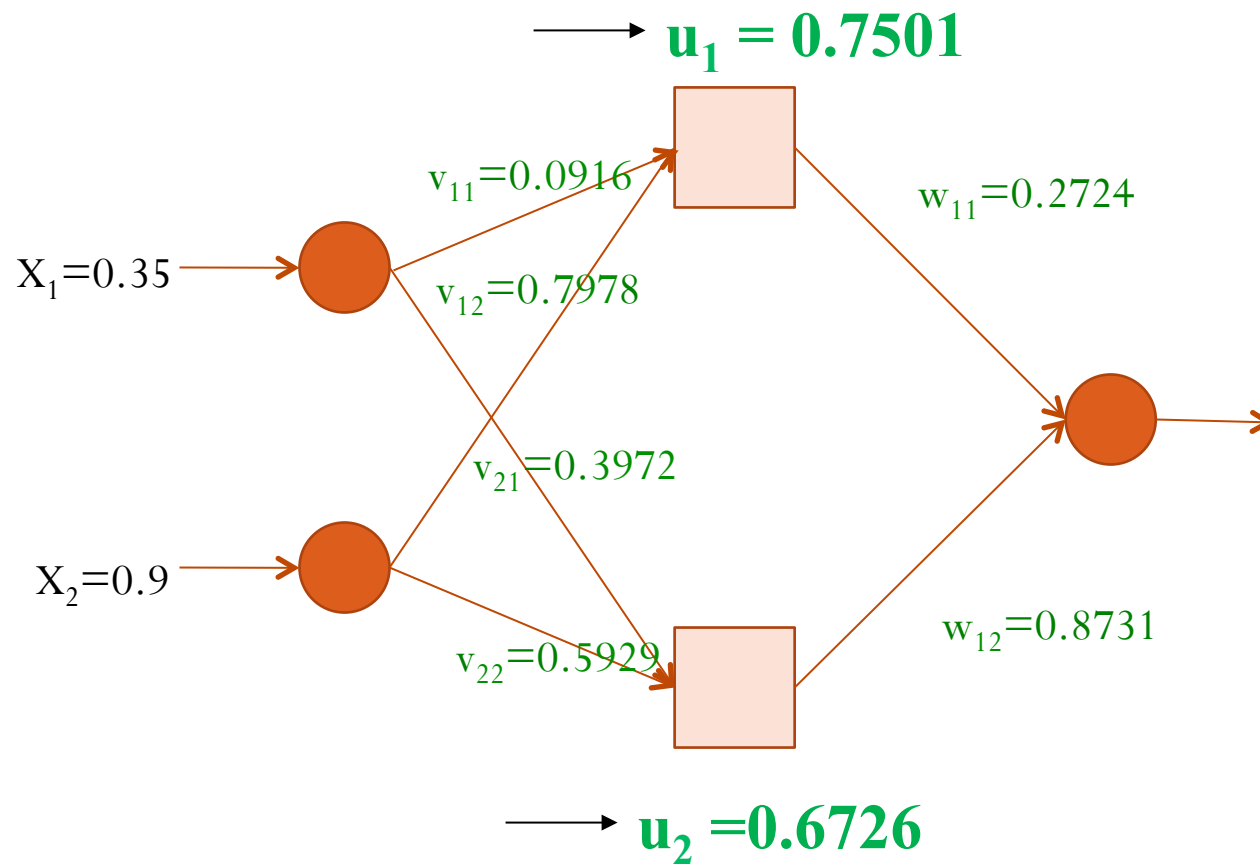
$$\delta_1' = -0.0024$$

$$\delta_2' = -0.0079$$



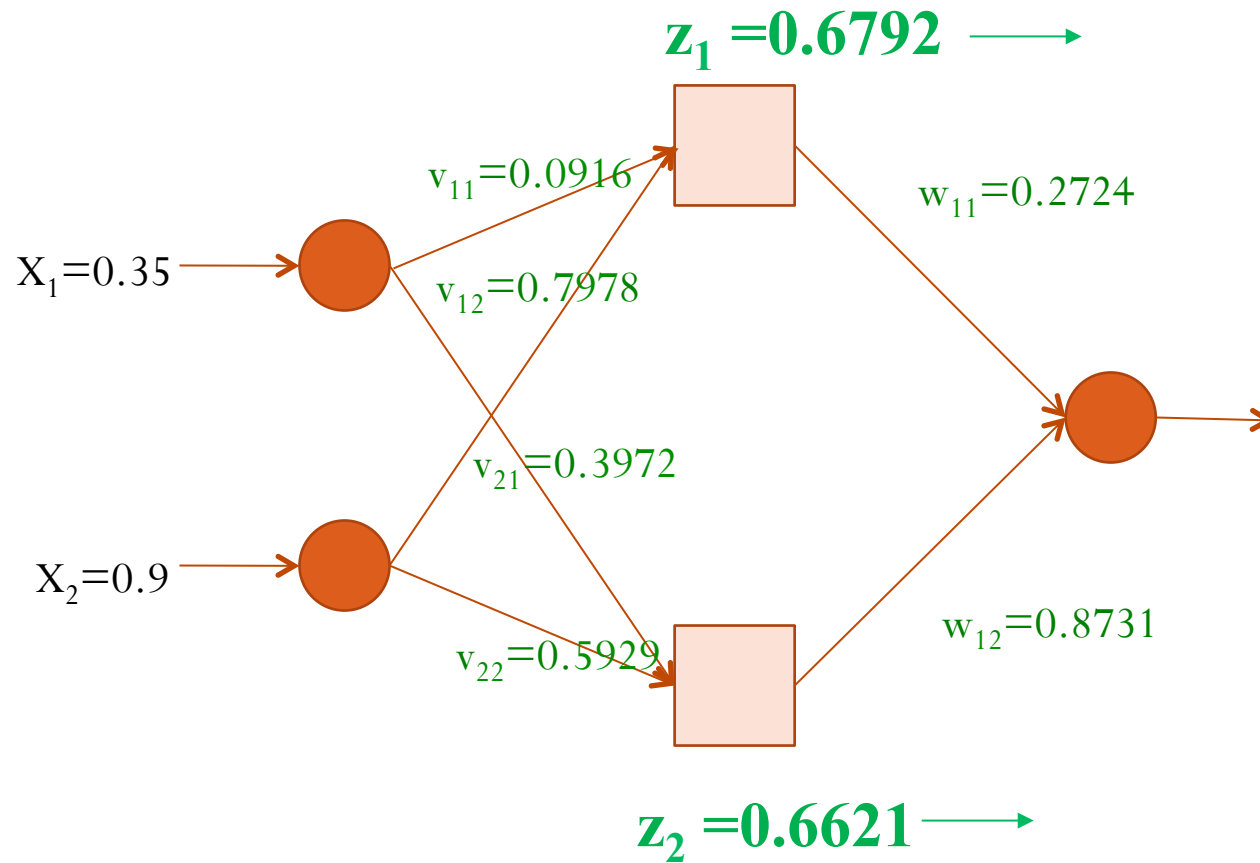
Παράδειγμα

$$u_k = \sum_i w_{ki} \cdot x_i$$



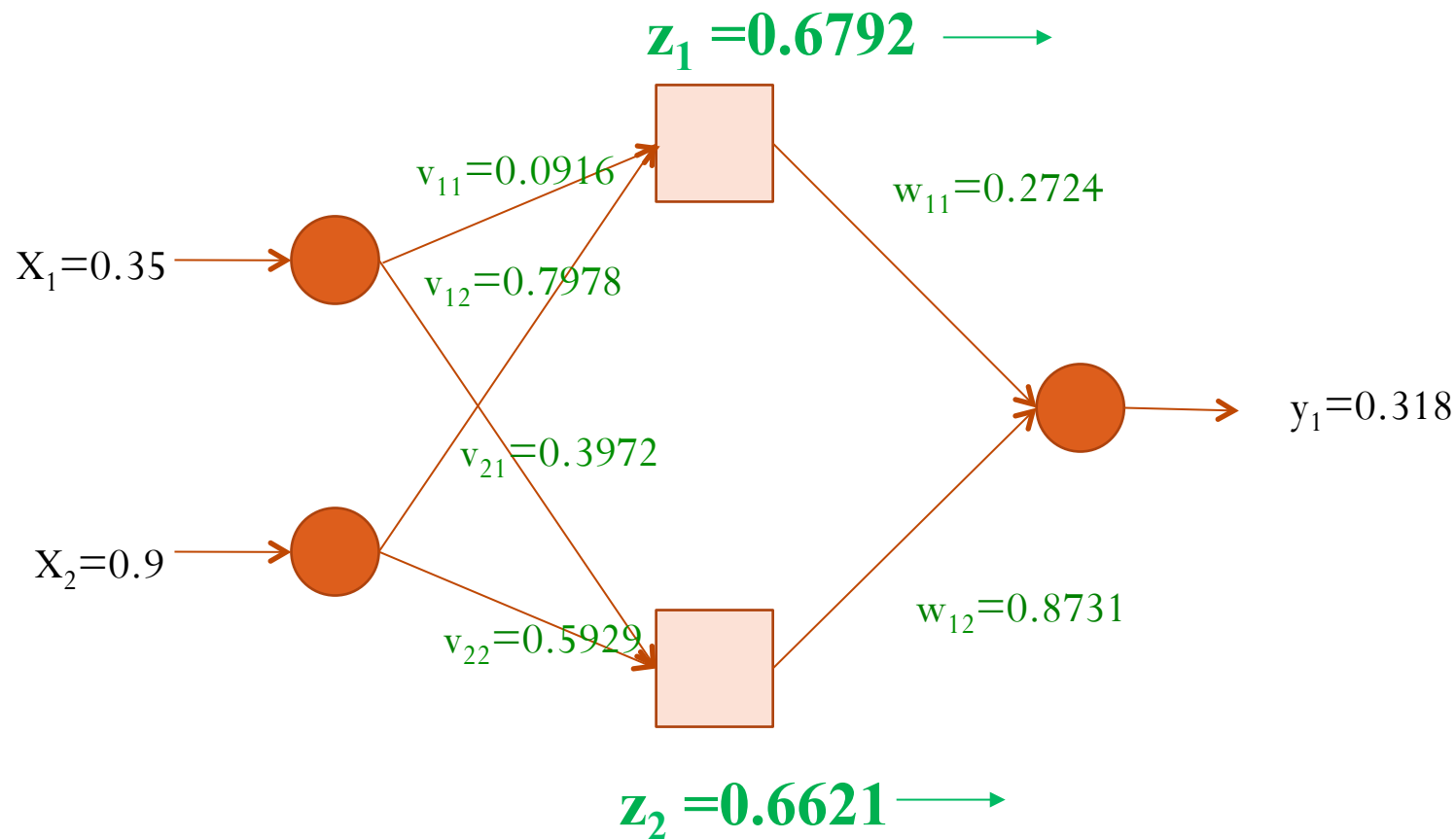
Παράδειγμα

$$y_k = f(u_k) = \frac{1}{1 + e^{-u_k}}$$



Παράδειγμα

$$y_k = f(u_k) = \frac{1}{1 + e^{-u_k}}$$



Το σφάλμα από 0.19 έχει γίνει 0.182

Μέρος 3^ο

Εισαγωγή στην άσκηση

Άσκηση 1



Άσκηση 1

Τμηματοποίηση



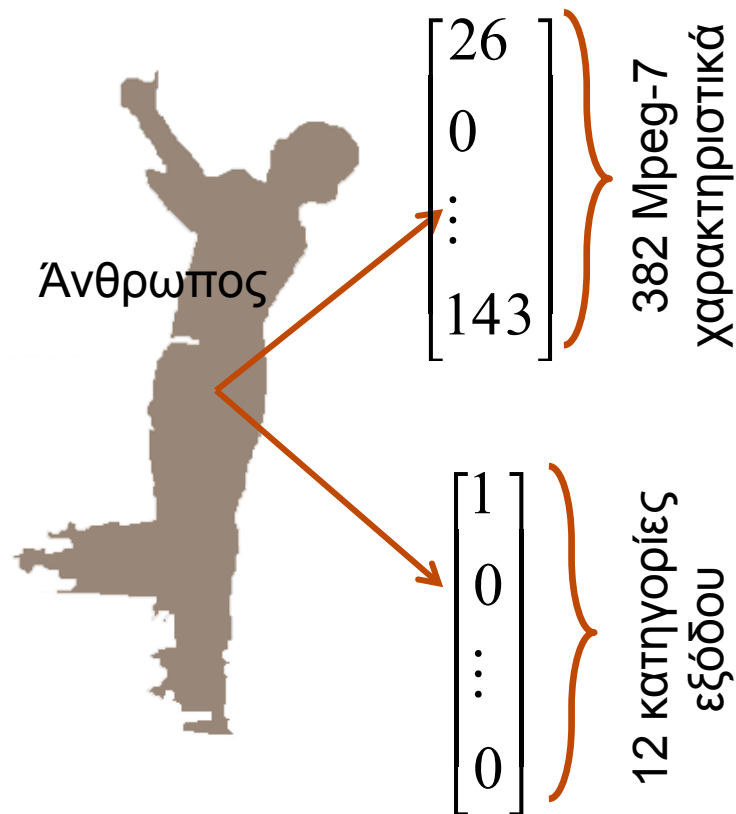
Άσκηση 1

Χαρακτηρισμός κάθε τμήματος της εικόνας



Κατηγορίες: Άνθρωπος, Άμμος, Κτήριο, Πεζοδρόμιο, Θάλασσα, Ουρανός, Κύμα, Φυτό, Γρασίδι, Δέντρο, Κορμός, Χώμα

Άσκηση 1



- Για κάθε τμήμα εικόνας δημιουργείται ένα διάνυσμα εισόδου και ένα διάνυσμα εξόδου
 - **Διάνυσμα εισόδου:** Προκύπτει από την εξαγωγή των MPEG-7 χαρακτηριστικών του τμήματος της εικόνας (Τα χαρακτηριστικά προκύπτουν από την επεξεργασία της εικόνας και μπορεί να σχετίζονται με στοιχεία όπως το χρώμα, η υφή κ.ο.κ.)
 - **Διάνυσμα εξόδου:** 12-διάστατο διάνυσμα που έχει την τιμή 1 μόνο στην γραμμή που αντιστοιχεί στην έννοια άνθρωπος και 0 στις άλλες γραμμές.

Άσκηση 1

Στόχος της άσκησης

- Δημιουργία Νευρωνικού Δικτύου
 - Είσοδος: MPEG-7 χαρακτηριστικά που περιγράφουν το τμήμα μιας εικόνας
 - Εξοδος: Ενα διάνυσμα 12 διαστάσεων όπου η μεγαλύτερη τιμή σε κάθε διάσταση θα προσδιορίζει σε ποια κατηγορία ανήκει το τμήμα
- Δεδομένα
 - Δεδομένα Εκπαίδευσης Νευρωνικού Δικτύου
 - TrainData: Κάθε στήλη περιέχει τα MPEG-7 χαρακτηριστικά του τμήματος μίας εικόνας
 - TrainDataTargets: Το διάνυσμα εξόδου που προσδιορίζει σε ποια κατηγορία ανήκει το συγκεκριμένο τμήμα
 - Δεδομένα Αξιολόγησης Νευρωνικού Δικτύου
 - TestData, TestDataTargets: αντίστοιχη δομή με τα προηγούμενα δεδομένα

Άσκηση 1-Βήμα 1

Επιλογή παρεμφερή αριθμού
δεδομένων από κάθε
κατηγορία

- Αρχικά έχουμε τα Mpeg-7 χαρακτηριστικά και τις αντίστοιχες κατηγορίες για 3510 τμήματα εικόνων.
- Μετά την επεξεργασία θα επιλέξουμε στήλες έτσι ώστε να έχουμε παρεμφερή αριθμό δεδομένων για κάθε κατηγορία.
- Δηλαδή x τμήματα που περιγράφουν άνθρωπο, x τμήματα που περιγράφουν ουρανό

TrainData

26	2	L	0
0	0	L	0
L	L	O	L
143	97	L	79

TestData

0	0	L	2
0	0	L	0
L	L	O	L
73	14	L	135

TrainDataTargets

0	0	L	0
0	0	L	0
1	1	L	0
0	0	L	0
0	0	L	1
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0

TestDataTargets

1	1	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	1
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0

Άσκηση 1-Βήμα 1

- Αν θέλω για παράδειγμα να επιλέξω x εισόδους/εξόδους από κάθε κατηγορία:

```
index1=find(TrainDataTargets(1,:),x);....
```

```
index12=find(TrainDataTargets(12,:),x);
```

```
%Αποθηκεύει στον πίνακα index τις 20 πρώτες στήλες που περιέχουν μονάδα στην τρίτη σειρά
```

```
TrainDataTargets1=TrainDataTargets(:,index1);...
```

```
TrainData1=TrainData(:,index1);...
```

```
%Αποθηκεύει από τους πίνακες TrainDataTargets,TrainData μόνο τις στήλες που προσδιορίζονται από το index
```

```
TrainData=[TrainData1,...,TrainData12];
```

```
TrainDataTargets=[TrainDataTargets1,...,TrainDataTargets12];
```

```
%Με αυτόν τον τρόπο έχουμε αποθηκεύσει 20 εισόδους/εξόδους από κάθε κατηγορία
```

- Στη συνέχεια θα πρέπει να ανακατέψω τα δεδομένα

```
order=randperm(indexesSize)
```

```
TrainData=TrainData(:,order);
```

```
TrainDataTargets=TrainDataTargets(:, order);
```

Άσκηση 1-Βήμα 2

Removeconstantrows

- Διαγράφουμε από τον πίνακα TrainData τις γραμμές που είναι σταθερές για όλες τις στήλες

TrainData

26	2	L	0
0	0	L	0
L	L	O	L
143	97	L	79

TestData

0	0	L	2
0	0	L	0
L	L	O	L
73	14	L	135

TrainDataTargets

0	0	L	0
0	0	L	0
1	1	L	0
0	0	L	0
0	0	L	1
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0

TestDataTargets

1	1	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	1
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0

Άσκηση 1-Βήμα 2

Removeconstantrows

- Διαγράφουμε από τον πίνακα TrainData τις γραμμές που είναι σταθερές για όλες τις στήλες
- Τις ίδιες γραμμές πρέπει να τις διαγράψουμε και από τον πίνακα TestData χρησιμοποιώντας την συνάρτηση removeconstantrows με παράμετρο 'apply'

TrainData

26	2	L	0
0	0	L	0
L	L	O	L
143	97	L	79

TestData

0	0	L	2
0	0	L	0
L	L	O	L
73	14	L	135

TrainDataTargets

0	0	L	0
0	0	L	0
1	1	L	0
0	0	L	0
0	0	L	1
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0

TestDataTargets

1	1	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	1
0	0	L	0
0	0	L	0
0	0	L	0
0	0	L	0

Άσκηση 1-Βήμα 2

Processpca

- Εφαρμόζουμε τη συνάρτηση `processpca` στον πίνακα `TrainData` με παράμετρο τέτοια ώστε να προκύψει ένας πίνακας με περίπου 20 γραμμές

```
[TrainData, ps]=  
    processpca(TrainData,0.002)
```

$$\begin{matrix} & \text{TrainData} & & \text{TestData} \\ 382 & \left[\begin{array}{cccc} 26 & 2 & L & 0 \\ 0 & 0 & L & 0 \\ L & L & O & L \\ 143 & 97 & L & 79 \end{array} \right] & & 382 \left[\begin{array}{cccc} 0 & 0 & L & 2 \\ 0 & 0 & L & 0 \\ L & L & O & L \\ 73 & 14 & L & 135 \end{array} \right] \end{matrix}$$

TrainDataTargets TestDataTargets

$$\begin{matrix} \left(\begin{array}{cccc} 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 1 & 1 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 1 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \end{array} \right) & & \left(\begin{array}{cccc} 1 & 1 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 1 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \end{array} \right) \end{matrix}$$

Άσκηση 1-Βήμα 2

Processpca

- Εφαρμόζουμε τη συνάρτηση `processpca` στον πίνακα `TrainData` με παράμετρο τέτοια ώστε να προκύψει ένας πίνακας με περίπου 20 γραμμές

`[TrainData, ps]=
processpca(TrainData,0.002)`

- Εφαρμόζουμε τον ίδιο μετασχηματισμό και στα `TestData` με παράμετρο `'apply'` στη συνάρτηση `processpca`

$$\begin{array}{cc}
 \text{TrainData} & \text{TestData} \\
 20 \left[\begin{pmatrix} -3.66 & -1.27 & L & -4.26 \\ -3.64 & -6.25 & L & 8.13 \\ L & L & O & L \\ -1.32 & 1.41 & L & -0.56 \end{pmatrix} \right. & 20 \left[\begin{pmatrix} 0 & 0 & L & 2 \\ 0 & 0 & L & 0 \\ L & L & O & L \\ 73 & 14 & L & 135 \end{pmatrix} \right.
 \end{array}$$

`TrainDataTargets` `TestDataTargets`

$$\begin{array}{cc}
 \begin{pmatrix} 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 1 & 1 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 1 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \end{pmatrix} & \begin{pmatrix} 1 & 1 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 1 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & L & 0 \end{pmatrix}
 \end{array}$$

Άσκηση 1-Βήμα 2

Processpca

- Εφαρμόζουμε τη συνάρτηση `processpca` στον πίνακα `TrainData` με παράμετρο τέτοια ώστε να προκύψει ένας πίνακας με περίπου 20 γραμμές

`[TrainData, ps]=
processpca(TrainData,0.002)`

- Εφαρμόζουμε τον ίδιο μετασχηματισμό και στα `TestData` με παράμετρο `'apply'` στη συνάρτηση `processpca`

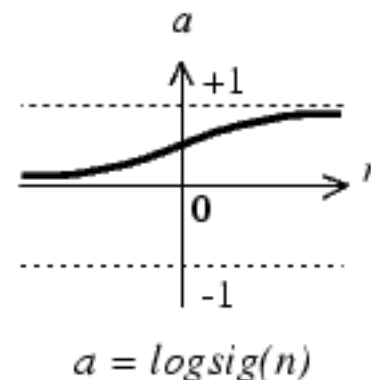
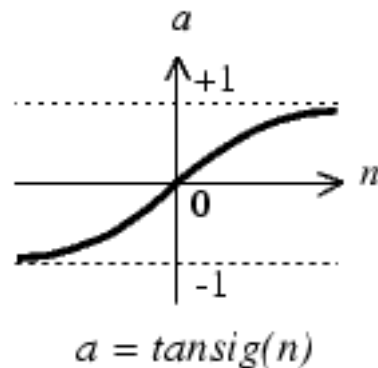
TrainData					TestData						
20	{	-3.66	-1.27	L	-4.26	20	{	-2.60	15.9	L	-2.08
		-3.64	-6.25	L	8.13			-1.15	-4.68	L	4.61
		L	L	O	L			L	L	O	L
		-1.32	1.41	L	-0.56			-0.93	-0.30	L	-0.07

TrainDataTargets					TestDataTargets						
20	{	0	0	L	0	20	{	1	1	L	0
		0	0	L	0			0	0	L	0
		1	1	L	0			0	0	L	0
		0	0	L	0			0	0	L	0
		0	0	L	1			0	0	L	0
		0	0	L	0			0	0	L	0
		0	0	L	0			0	0	L	0
		0	0	L	0			0	0	L	0
		0	0	L	0			0	0	L	1
		0	0	L	0			0	0	L	0
		0	0	L	0			0	0	L	0
		0	0	L	0			0	0	L	0
		0	0	L	0			0	0	L	0
		0	0	L	0			0	0	L	0
		0	0	L	0			0	0	L	0

Άσκηση 1-Βήμα 3

Επιλογή κατάλληλης συνάρτησης ενεργοποίησης

- Με δεδομένο ότι η συνάρτηση `newff` που χρησιμοποιείται για την δημιουργία του νευρωνικού δικτύου κανονικοποιεί τα δεδομένα σε τιμές από $[-1,1]$ να διαλέξετε ποια από τις συναρτήσεις `'tansig'`, `'logsig'` είναι καταλληλότερη για συνάρτηση ενεργοποίησης στο επίπεδο της εξόδου. Στα κρυμμένα επίπεδα θεωρείστε την `'tansig'` σαν συνάρτηση ενεργοποίησης.



Άσκηση 1-Βήμα 4

Δημιουργία νευρωνικού δικτύου

- Δημιουργούμε ένα νευρωνικό με 2 κρυμμένα επίπεδα που το πρώτο έχει 10 και το δεύτερο 15 νευρώνες. Η συνάρτηση ενεργοποίησης για κάθε κρυμμένο επίπεδο είναι η `tansig` και για το κρυμμένο επίπεδο η `purelin`.

```
net=newff(TrainData,TrainDataTargets,[10 15])
```

- Χωρίζουμε τα δεδομένα μας σε ένα σύνολο εκπαίδευσης και ένα σύνολο επαλήθευσης έτσι ώστε να χρησιμοποιηθεί η μέθοδος `Early Stopping` για τον τερματισμό της εκπαίδευσης του νευρωνικού

```
net.divideParam.trainRatio=0.8  
net.divideParam.valRatio=0.2  
net.divideParam.testRatio=0
```

- Επιλέγω μέγιστο αριθμό εποχών για την εκπαίδευση τις 300

```
net.trainParam.epochs=300
```

Άσκηση 1-Βήμα 4

Εκπαίδευση νευρωνικού δικτύου

- Εκπαιδεύω το νευρωνικό δίκτυο χρησιμοποιώντας την συνάρτηση `train` με εισόδους τα `TrainData` και `TrainDataTargets`.

```
net=train(net,TrainData,TrainDataTargets)
```

- Σκοπός της εκπαίδευσης είναι να μάθει το νευρωνικό δίκτυο τα δεδομένα που βρίσκονται στον πίνακα `TrainData` και σε κάθε είσοδο από το `TrainData` να βγάζει την αντίστοιχη έξοδο από το `TrainDataTargets`.

Άσκηση 1-Βήμα 4

Αξιολόγηση νευρωνικού δικτύου

- Αξιολογώ το νευρωνικό δίκτυο χρησιμοποιώντας τα δεδομένα των πινάκων TestData και TestDataTargets.
- Εφαρμόζω τα δεδομένα του TestData στο νευρωνικό δίκτυο και στην έξοδο παίρνω τον πίνακα TestDataOutput



$\text{TestDataOutput} = \text{sim}(\text{net}, \text{TestData})$

Άσκηση 1-Βήμα 4

Αξιολόγηση νευρωνικού δικτύου

- Χρησιμοποιώντας την συνάρτηση `eval_Accuracy_Precision_Recall` που μου δίνεται από τη σελίδα του μαθήματος υπολογίζω το accuracy το precision και το recall για τα TestData:

```
[accuracy,precision,recall]=  
    eval_Accuracy_Precision_Recall(TestDataOutput,TestDataTargets)
```

Άσκηση 1-Βήμα 5

Μελέτη της επίδρασης του αριθμού των νευρώνων και των επιπέδων

- Εργαζόμενος όπως στο Βήμα 4 υπολογίζω τα accuracy, τα precision και recall για διάφορους συνδυασμούς αριθμού νευρώνων με ένα ή δύο κρυμμένα επίπεδα.
- Επιλέγω τον συνδυασμό και το νευρωνικό που μου δίνει τα καλύτερα αποτελέσματα.

Άσκηση 1-Βήματα 6,7

- Για τα βήματα αυτά εργαζόμαστε αλλάζοντας κάποιες από τις παραμέτρους του νευρωνικού δικτύου με τα καλύτερα χαρακτηριστικά που κατασκευάσαμε στο βήμα 5.
- Η αξιολόγηση γίνεται με παρόμοιο τρόπο.

Μέρος 4^ο

Μέθοδος Ανάλυσης σε Κύριες Συνιστώσες

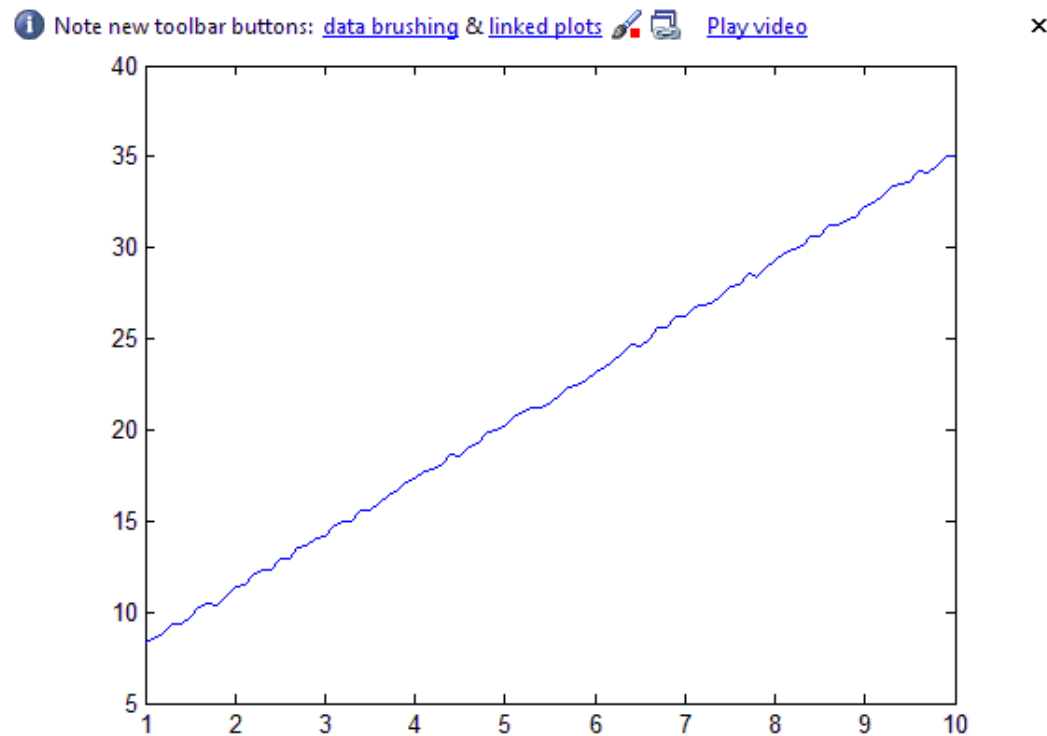
Ανάλυση σε Κύριες Συνιστώσες (processpca)

- Μας επιτρέπει σε ένα σύνολο δεδομένων να μειώσουμε τις διαστάσεις των δεδομένων χωρίς να χάσουμε σημαντική πληροφορία.
- Χρησιμοποιεί έναν ορθογώνιο μετασχηματισμό για να μετατρέξει ένα σύνολο συσχετιζόμενων τιμών σε γραμμικά ασυσχέτιστες

Ανάλυση Κυρίων Συνιστωσών

Παράδειγμα


- $x=1:0.1:10; y=3*x+5+\text{rand}(1,\text{length}(x))*0.5;$
- $A=[x;y]; \text{plot}(A(1,:), A(2,:));$

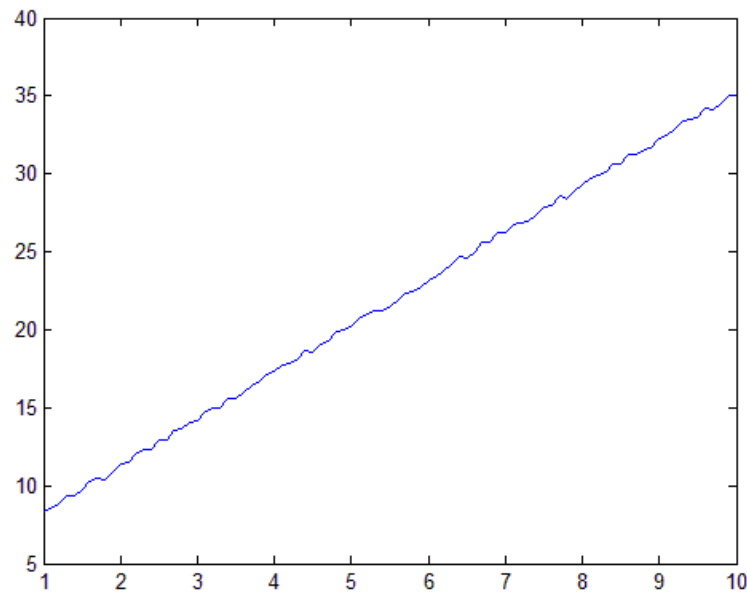



Ανάλυση Κυρίων Συνιστωσών

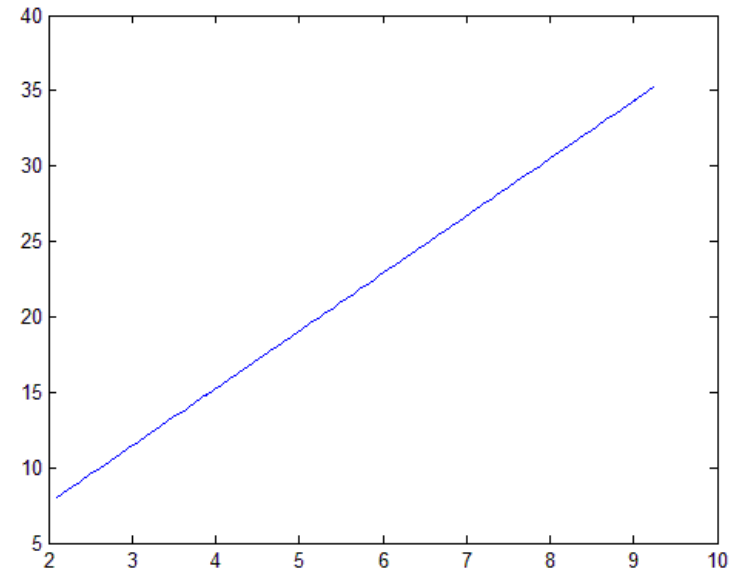
Παράδειγμα

- `[A2,ps]=processpca(A,0.1);`
- `A_new=processpca('reverse',A2,ps);`
- `plot(A_new(1,:), A_new(2,:));`

Note new toolbar buttons: [data brushing](#) & [linked plots](#)   [Play video](#)



Note new toolbar buttons: [data brushing](#) & [linked plots](#)   [Play video](#)



Ανάλυση Κυρίων Συνιστωσών

Παράδειγμα

- Επαναλάβετε τη διαδικασία για $y=x.^2$;
- Τί παρατηρείτε;