Session #21&22

CMSC 409:
Artificial Intelligence

http://www.people.vcu.edu/~mmanic/

Virginia Commonwealth University,
Fall 2023,
Dr. Milos Manic
(mmanic@vcu. edu)

1

CMSC 409: Artificial Intelligence
Session # 21&22

Topics for today

- Announcements
- Project 3, lessons learned
- Previous session review
- Error Back Propagation (EBP) algorithm
  - *Derivation*
  - *Learning example*

Upcoming topics…
- Probability Theory
- Subjective probability - Bayes' Rule

2

CMSC 409: Artificial Intelligence

Announcements Session # 21&22

- IMPORTANT:
  - *Course materials (slides, assignments) are copyrighted by instructor & VCU. Sharing/posting/chatGPT/similar is copyright infringement and is strictly prohibited. Such must be immediately reported.*
- Canvas
  - *Prev. session slides updated*
- TAs
  - *Victor Cobilean <cobileanv@vcu.edu>, Harindra Sandun Mavikumbure mavikumbureh@vcu.edu*
  - *TA office hours: Thursdays, 3:30 - 4:30pm (Zoom)*
- Project #3
  - *Deadline was Oct. 26; Review a week from the deadline*
- Project #4
  - *Deadline is Nov. 9*
- Paper (optional)
  - *The 3rd draft due Nov. 3 (noon)*
  - *In addition to previous draft, it should contain a technique (or selection thereof), you plan on using to solve the selected problem (check out the class paper instructions for the 3rd draft)*
  - *The 4th draft (final submission) due Nov. 28*
  - *In addition to previous draft, it should contain a technique (or selection thereof), you plan on using to solve the selected problem (check out the class paper instructions for the 4th draft)*
- Subject line and signature
  - *Please use [CMSC 409] Last_Name Question*

3

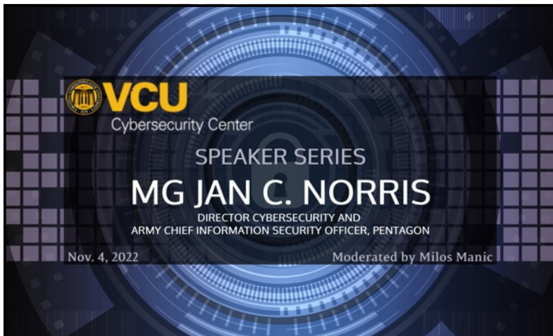
CMSC 409: Artificial Intelligence

Session # 21&22

Announcements, cont.

- Nov. 8, VCU closed
- **Final exam**
  - **Dec. 13 (take home, 48 hr open book exam)**
  - prep examples will be posted prior
- Interest in PhD or MSc program?
- Subject line and signature
  - *Please use [CMSC 409] Last_Name Question*

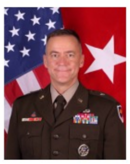
4



2022 Cybersecurity Speaker Series

Cyber security and future workforce in defending information technology infrastructure

November 4, 2022 11 a.m.–Noon (Eastern Time)
Hosted by **Milos Manic, Ph.D., FIEEE, FCCI**
VCU College of Engineering West Hall, Room 106



Major General Jan C. Norris
Deputy Chief Information Officer (IMA)
The Office of the Chief Information Officer
United States Army Reserve

Register at:
https://cybersecurity.vcu.edu

Lessons learned

Project 03

Class Statistics

STATISTICS

COUNT	24
Minimum Value	0
Maximum Value	18
Range	18
Average	14.91
Median	17
Standard Deviation	5.03
Variance	25.38

GRADE DISTRIBUTION

Greater than 100	13
90 - 100	6
80 - 89	3
70 - 79	0
60 - 69	0
50 - 59	0
40 - 49	0
30 - 39	0
20 - 29	0
10 - 19	0
0 - 9	2

Project 3 review

Pr. 3.1 Understand and explore Reinforcement Learning (6pts)

1. Discuss what Reinforcement Learning (RL) is. (2pts)
2. What are the real-world applications which use RL, discuss 2 application areas. (2pts)
3. Discuss the advantages and disadvantages of RL. (2pts)

Best practices:

Provide complete answers:

- Concise and to the point answers
- Try to avoid lengthy and possibly unrelated discussions

Project 3 review

Pr. 3.2 Simulation of a self-driving cab using Q Learning (9 pts)

1. Change the following hyperparameters of the RL pipeline. Make a note of the reward for each combination of the hyperparameters. Discuss how the values of the hyperparameters affect the total reward for the smart cab **(6pts)**
   a. alpha: 0.01, 0.1, 0.9
   b. gamma: 0.40, 0.99
   c. epsilon: 0.5, 1

   Example: You can try:
   - alpha: 0.01
   - gamma: 0.40
   - epsilon: 0.5

   (then change alpha for the same gamma and epsilon, etc.)

2. Discuss the importance of reducing the epsilon parameter in achieving a higher reward **(1pts)**

3. Run the "*Smart_cab_brute_force.ipynb*" and compare and discuss the results with the reinforcement learning results. **(2pts)**

Project 3 review

Pr.3.2
Importance of parameters

$$Q_{t+1}(s + a) = Q_t(s,a) + \alpha(R + \lambda maxQ_t(s + 1, :) - Q_t(s,a))$$

α - Learning Rate:

- How much $R + \lambda maxQ_t(s + 1, :) - Q_t(s,a)$ contributes to *the Q*-values in the next time step, $Qt+1(s + a)$:
  - *Define the portion of reward R, discount factor λ, prev. step Q-value $Q_t(s,a)$, and maxQ*
  - *maxQ or $maxQ_t(s + 1, :)$, evaluates all possible actions for "neighboring" state s+1.*
- Larger alpha -> faster convergence,
  - *but may lead to convergence challenges (the "convergence" here refers to the process of algorithm converging to the maximum reward - R)*
  - *As algorithm goes through iterations, it attempts to maximize reward for given Q(state, action),*
  - *But, with larger α, algorithm may get stuck with lower reward (may not be able to improve reward). But, the process also depends on λ and epsilon...*

Project 3 review

Pr.3.2
Importance of parameters

- λ *[0, 1]* - Discount Factor
  - larger lambda (closer to **1**) - makes agent focus on the long-term effective award;
  - smaller lambda (closer to **0)** - makes agent focus on immediate reward (greedy agent).

Project 3 review

Pr.3.2
Importance of parameters

- $\mathcal{E}_{decay}$*[0, 1]* - decay rate
  - *Epsilon closer to 1* - agent performs more explorations (more random actions)
  - *Epsilon closer to 0* - agent performs more exploitation (follows *Q*-table).

Exploration vs. exploitation

- With exploration, an agent takes random decisions; Useful when learning about the environment.
- With exploitation, an agent takes actions based on what agent already knows (from the Q table)

Project 3 review

Alpha	Gamma	Epsilon	Reward
0.01	0.99	0.5	6.09
0.01	0.99	1	6.8
0.01	0.4	0.5	0.5
0.01	0.4	1	2.4
0.1	0.99	0.5	8.5
0.1	0.99	1	8.7
0.1	0.4	0.5	7.65
0.1	0.4	1	8.2
0.9	0.99	0.5	8.2
0.9	0.99	1	6.9
0.9	0.4	0.5	7.55
0.9	0.4	1	7.3

Project 3 review

Extra Credit: (3pts)
Implement the *epsilon_decay* as:
*epsilon_decay = min_epilson +(max_epilson - min_epilson) * exp (-epsilon_decay_rate * episode)*

using the following hyperparameters:

- min_epilson
- max_epilson
- epsilon_decay_rate

1. Which values of these hyperparameters (alpha, gamma, epsilon, min_epilson, max_epilson, epsilon_decay_rate) result in the highest reward?
2. What is the highest reward?

Project 3 review

- **Extra Credit**

```python
# Implementing the Q Learning Algorithm:

for episode in range(total_ep):

  # Reset Environment:
  state = env.reset()
  step = 0
  done = False

  for step in range(max_steps):

    # Choose an action a in the current world state(s) (step 3)
    # First we randomize a number
    exp_exp_tradeoff = random.uniform(0, 1)

    # If this number > greater than epsilon --> exploitation (taking the biggest q value for the current state):
    if exp_exp_tradeoff > epsilon:
      action = np.argmax(q_table[state, :])

    # Else, doing random choice:
    else:
      action = env.action_space.sample()

    # Take the action (a) and observe the outcome state (s') and the reward (r)
    new_state, reward, done, info = env.step(action)

    # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
    q_table[state, action] = q_table[state, action] + lr * (reward + gamma *
                                np.max(q_table[new_state, :]) - q_table[state, action])

    # Our new state:
    state = new_state

    # If done True, finish the episode:
    if done == True:
      break

  # Reduce epsilon (because we need less and less exploration):
  epsilon = min_epsilon + (max_epsilon – min_epsilon) *np.exp(-decay_rate*episode)
  # Increment number of episodes:
  episode += 1


print("Training finished.\n")
```

Exploitation vs. Exploration

- *As agent learns, epsilon parameter gets smaller*
- *Through the process, an agent gradually moves from exploration to exploitation*

During submission

- Project deliverable should be a zip file containing:
  - Written report (pdf) with answers to all of the questions above.
  - Updated Source code in Ipython notebook file (.ipynb)
- Submit your zip file to Canvas. Please name the zip file as GroupName_Project3.zip.

Error Back Propagation (EBP) algorithm

- *Derivation*
- *Learning example*
- *Heuristic approaches to Error Back Propagation modifications*
  - *variable gain, alpha, weight rescaling,*
  - *momentum,*
  - *search along the gradient*
  - *Quickprop, RPROP, Delta-Bar-Delta, Back Percolation*

Error-Back Propagation algorithm

- Werbos 1974,
- Later Rumelhart & McClelland 1986.

Error-Backpropagation Algorithm (EBP)

- *Breakthrough, finally annulling Minsky & Papert's influence from their book "Perceptrons" 1969.*
- *Problems:*
  - *Convergence difficulties*
  - *Oscillations*
  - *Speed*

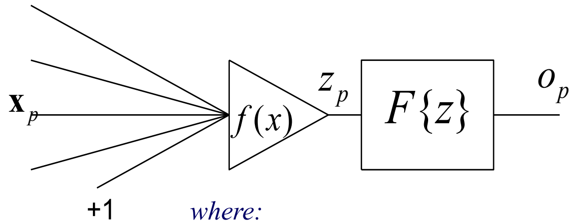
[Werbos 74] Paul J. Werbos. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, 1974.

[Werbos 94] Paul John Werbos, "The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting, Wiley-Interscience, January 1994.

[Rumelhart 86] Rumelhart, D.E., McClelland, J.L., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1, Cambridge, Ma, MIT Press, 1986.

EBP Error Back Propagation algorithm

single output



where:
f(x) – activation function
z_p – neuron outputs
x_p – input pattern
$F\{z\}$ – non-linear function based on activation function of several neurons

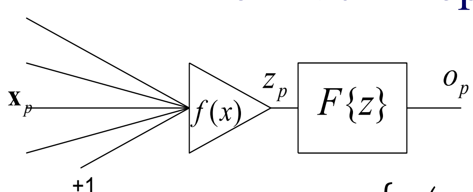
For each pattern p, network output is:

$$o_p = F\Big\{ f\big(w_1 x_{p1} + w_2 x_{p2} + \cdots + w_n x_n\big)\Big\}$$

Total error: $TE = \sum_{p=1}^{np} \left[d_p - o_p\right]^2$

19

EBP Error Back Propagation algorithm



single output case…

$$TE = \sum_{p=1}^{np} \left[d_p - o_p\right]^2$$

$$o_p = F\Big\{ f\big(w_1 x_{p1} + w_2 x_{p2} + \cdots + w_n x_{pn}\big)\Big\}$$

Consider a single weight w_i. The gradient of TE along w_i is:

$$\frac{d(TE)}{dw_i} = -\sum_{p=1}^{np} \left[2\left(d_p - o_p\right) \frac{do_p}{dz_p} \frac{dz_p}{dnet_p} \frac{dnet_p}{dw_i} \right]$$

(goal is to minimize TE)

20

EBP Error Back Propagation algorithm

The gradient of error function with respect to weights:

$$\frac{d(TE)}{dw_i} = -\sum_{p=1}^{np}\left[2(d_p - o_p)\frac{do_p}{dz_p}\frac{dz_p}{dnet_p}\frac{dnet_p}{dw_i}\right]$$

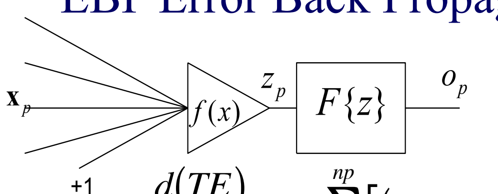
$$\frac{d(TE)}{dw_i} = -2\sum_{p=1}^{np}\left[(d_p - o_p)F'\{z_p\}\cdot f'(net_p)\cdot x_{pi}\right]$$

$\frac{do_p}{dz_p}$ *- gain of the function F through the network*

$\frac{dnet_p}{dw_i}$ *- change of net with regards to weight*

$\frac{dz_p}{dnet_p}$ *- slope of activation function*

EBP Error Back Propagation algorithm



one output case…

$$\frac{d(TE)}{dw_i} = -2\sum_{p=1}^{np}\left[(d_p - o_p)F'\{z_p\}f'(net_p)x_{pi}\right]$$

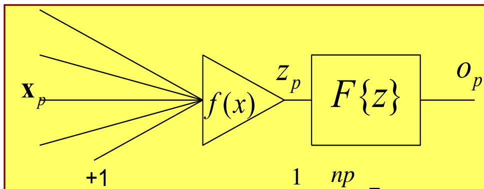
To move towards the minimum, the weight change should be opposite to the gradient change (direction).

$$\Delta w_{pi} \sim 2\sum_{p=1}^{np}\left[(d_p - o_p)\cdot F'\{z_p\}\cdot f'(net_p)\cdot x_{pi}\right]$$

…which can be extended for all weights of this neuron (α instead of ~):

$$\Delta \mathbf{w}_p = \alpha\cdot\sum_{o=1}^{no}\sum_{p=1}^{np}\left[(d_{op} - o_{op})\cdot F'\{z_p\}\cdot f'(net_p)\cdot \mathbf{x}_p\right]$$

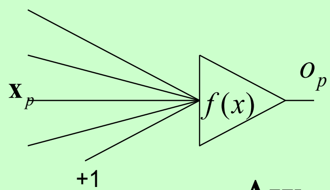
EBP Error Back Propagation algorithm



Error Back Propagation for a single output case …

$$\Delta\mathbf{w}_p = \alpha \cdot \sum_{o=1}^{1}\sum_{p=1}^{np}\left[\left(d_{op} - o_{op}\right)\cdot F'\{z_p\}\cdot f'(net_p)\cdot\mathbf{x}_p\right]$$

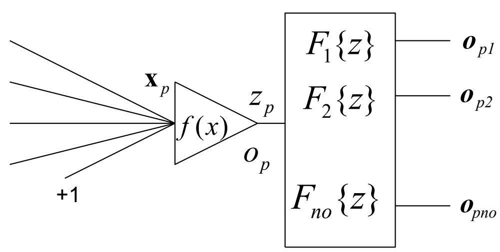
Note the similarity!!!



Delta rule for single neuron…

$$\Delta\mathbf{w}_p = \alpha\sum_{p=1}^{np}\left[\left(d_p - o_p\right)f'\left(net_p\right)\mathbf{x}_p\right]$$

EBP Error Back Propagation algorithm



multiple output case…

where:
no – number of outputs

The weight increment for the whole network is a **superposition** of weight modifications from errors on all outputs:

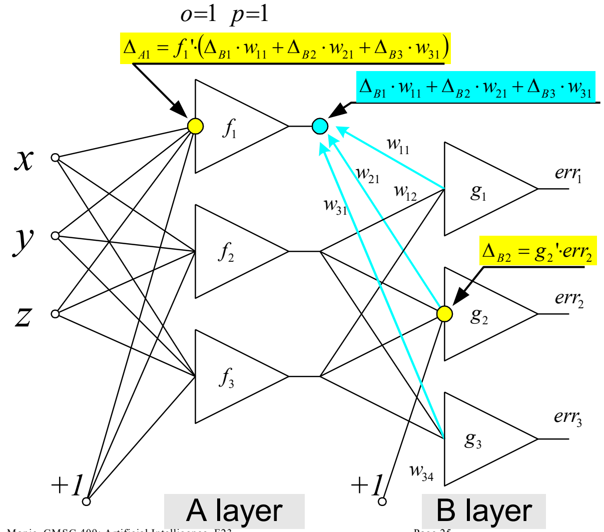
$$\Delta\mathbf{w}_p = \alpha \cdot \sum_{o=1}^{no}\sum_{p=1}^{np}\left[\left(d_{op} - o_{op}\right)\cdot F'\{z_p\}\cdot f'(net_p)\cdot\mathbf{x}_p\right]$$

…weight update by each error from each output….

EBP Error Back Propagation algorithm

$$\Delta \mathbf{w}_p = \alpha \cdot \sum_{o=1}^{no} \sum_{p=1}^{np} \left[(d_{op} - o_{op}) \cdot F'\{z_p\} \cdot f'(net_p) \cdot \mathbf{x}_p \right]$$

$$\Delta \mathbf{w} = \alpha \cdot \Delta \cdot \mathbf{x}$$



- *Example of 2-layer network*
- *Forward computation & backward computation*
- *Error is computed back (back propagated)*

25

Error Back Propagation (EBP) algorithm

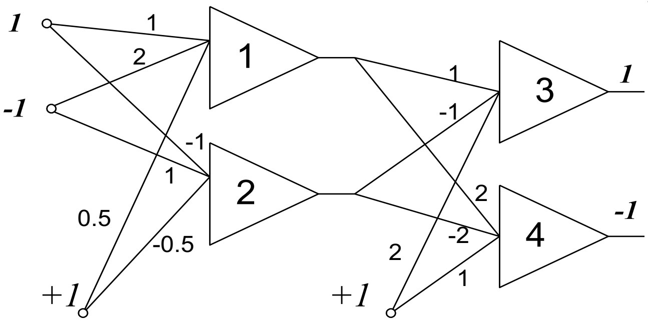
- *Derivation*
- *Learning example*
- *Heuristic approaches to Error Back Propagation modifications*
  - *variable gain, alpha, weight rescaling,*
  - *momentum,*
  - *search along the gradient*
  - *Quickprop, RPROP, Delta-Bar-Delta, Back Percolation*

26

EBP Error Back Propagation algorithm

EXAMPLE:

$k = 0.5 \quad \alpha = 1.0$



- *Example of 2-layer network*
- *We will process single pattern*
- *with parameters chosen as:*

$$\mathrm{x} = \begin{bmatrix} +1 \\ -1 \end{bmatrix} \quad \mathrm{W} = \begin{bmatrix} +1 & +2 & +.5 \\ -1 & +1 & -.5 \\ +1 & -1 & +2 \\ +2 & -2 & +1 \end{bmatrix} \quad \mathrm{d} = \begin{bmatrix} +1 \\ -1 \end{bmatrix}$$

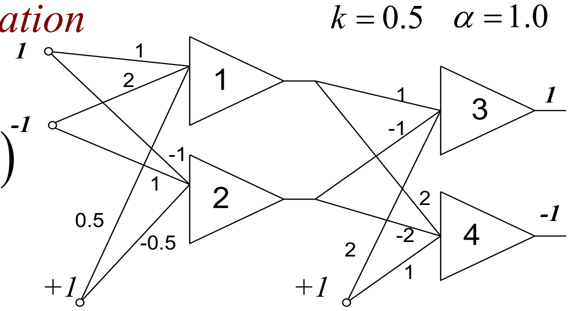
EBP Error Back Propagation algorithm

example

1. Forward Computation

$k = 0.5 \quad \alpha = 1.0$

$$net = \sum w_i x_i$$
$$out = \tanh(k \cdot net)$$



$net_1 = 1 \cdot 1 - 1 \cdot 2 + 1 \cdot 0.5 = -0.5;$ $\quad out_1 = \tanh(0.5 \cdot net_1) = -0.2449$

$net_2 = -2.5;$ $\quad out_2 = \tanh(0.5 \cdot net_2) = -0.8483$

now propagate these values to a 2nd layer……………………………………………

$net_3 = 1 \cdot (-0.2449) - 1 \cdot (-0.8483) + 2 \cdot 1 = 2.6034;$ $\quad out_3 = 0.86215$

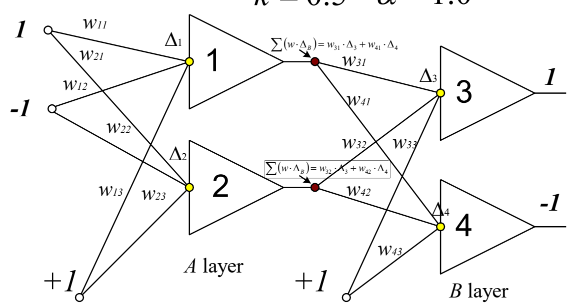
$out_4 = 0.8017$

EBP Error Back Propagation algorithm
example

$k = 0.5 \quad \alpha = 1.0$

2. Error Computation

$$d = \begin{bmatrix} +1 \\ -1 \end{bmatrix}$$



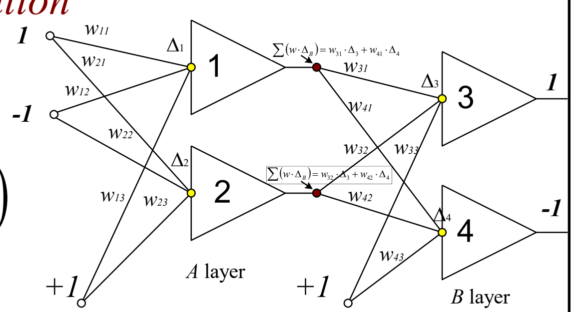
$$error = desired - actual\ output$$

$$err_3 = 1 - 0.86215 = 0.1378$$
$$err_4 = -1 - 0.8017 = -1.8017$$

EBP Error Back Propagation algorithm
example

$k = 0.5 \quad \alpha = 1.0$

3. Error back propagation through last layer



soft activation function:

$$f' = gain \cdot \left(1 - out^2\right)$$
$$\Delta_B = f' \cdot err$$

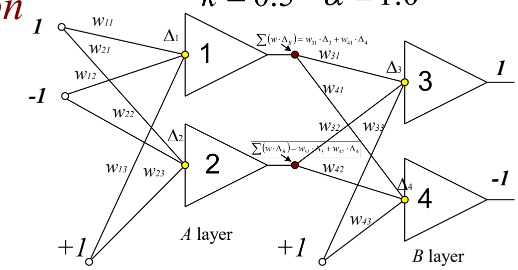
$$f_3' = 0.5 \cdot \left(1 - 0.86215^2\right) = 0.1283$$
$$f_4' = 0.5 \cdot \left(1 - 0.8017^2\right) = 0.1786$$

$$\Delta_3 = 0.1283 \cdot 0.1378 = 0.0177$$
$$\Delta_4 = 0.1786 \cdot (-1.8017) = -0.3218$$

EBP Error Back Propagation algorithm
example

4. Error back propagation through the first layer

$k = 0.5 \quad \alpha = 1.0$



$$f' = gain \cdot \left(1 - out^2\right)$$

$$\Delta_A = f' \cdot \sum \left(w \cdot \Delta_B\right)$$

$$f_1' = 0.5 \cdot \left(1 - 0.2449^2\right) = 0.4700$$
$$f_2' = 0.5 \cdot \left(1 - 0.8483^2\right) = 0.1402$$

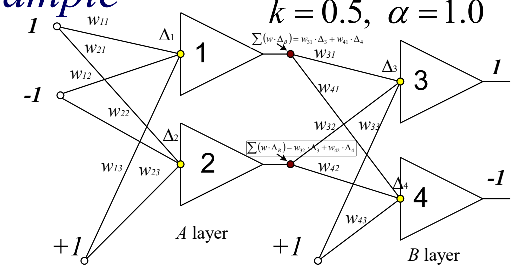
$$\Delta_1 = f_1' \cdot \left(w_{31} \cdot \Delta_3 + w_{41} \cdot \Delta_4\right)$$
$$\Delta_2 = f_2' \cdot \left(w_{32} \cdot \Delta_3 + w_{42} \cdot \Delta_4\right)$$

$$\Delta_1 = 0.4700 \cdot \left(1 \cdot 0.0177 + 2 \cdot \left(-0.3218\right)\right) = -0.2942$$
$$\Delta_2 = 0.1402 \cdot \left(\left(-1\right) \cdot 0.0177 + \left(-2\right) \cdot \left(-0.3218\right)\right) = 0.0878$$

EBP Error Back Propagation algorithm
example

$k = 0.5, \quad \alpha = 1.0$

5. Calculate layer A weight update



$$\Delta \mathbf{w} = alpha \cdot \Delta \cdot \mathbf{x}$$

$$\Delta_1 = -0.2942 \quad \begin{cases} \Delta w_{11} = 1.0 \cdot \left(-0.2942\right) \cdot 1 = -0.2942 \\ \Delta w_{12} = 1.0 \cdot \left(-0.2942\right) \cdot \left(-1\right) = 0.294 \\ \Delta w_{13} = 1.0 \cdot \left(-0.2942\right) \cdot 1 = -0.294 \end{cases} \quad \textit{Neuron 1}$$

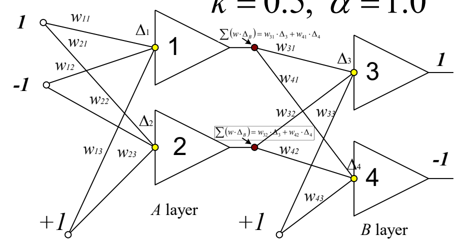
$$\Delta_2 = 0.0878 \quad \begin{cases} \Delta w_{21} = 1.0 \cdot 0.0878 \cdot 1 = 0.0878 \\ \Delta w_{22} = 1.0 \cdot 0.0878 \cdot \left(-1\right) = -0.0878 \\ \Delta w_{23} = 1.0 \cdot 0.0878 \cdot 1 = 0.0878 \end{cases} \quad \textit{Neuron 2}$$

EBP Error Back Propagation algorithm *example*

$k = 0.5, \ \alpha = 1.0$

6. Perform layer A weight update

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$$



Neuron 1

$$\begin{cases} w_{11} = w_{11} + \Delta w_{11} = 1 - 0.2942 = 0.7058 \\ w_{12} = w_{12} + \Delta w_{12} = 2 + 0.2942 = 2.2942 \\ w_{13} = w_{13} + \Delta w_{13} = 0.5 - 0.2942 = 0.2058 \end{cases}$$

Neuron 2

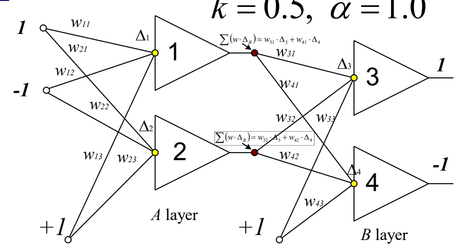
$$\begin{cases} w_{21} = w_{21} + \Delta w_{21} = -1 + 0.0878 = -0.9122 \\ w_{22} = w_{22} + \Delta w_{22} = 1 - 0.0878 = 0.9122 \\ w_{23} = w_{23} + \Delta w_{23} = -0.5 + 0.0878 = -0.4122 \end{cases}$$

EBP Error Back Propagation algorithm *example*

$k = 0.5, \ \alpha = 1.0$

7. Calculate layer B weight update

$$\Delta\mathbf{w} = alpha \cdot \Delta \cdot \mathbf{x}$$



Neuron 3

$\Delta_3 = 0.0177$

$$\begin{cases} \Delta w_{31} = 1.0 \cdot 0.0177 \cdot (-0.2449) = -0.0043 \\ \Delta w_{32} = 1.0 \cdot 0.0177 \cdot -(0.8483) = -0.0150 \\ \Delta w_{33} = 1.0 \cdot 0.0177 \cdot 1 = 0.0177 \end{cases}$$

Neuron 4

$\Delta_4 = -0.3218$

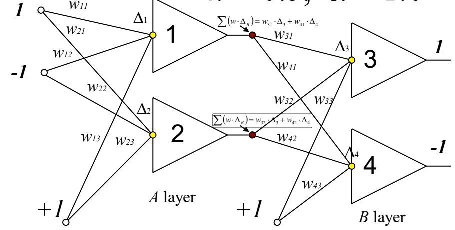
$$\begin{cases} \Delta w_{41} = 1.0 \cdot (-0.3218) \cdot (-0.2449) = 0.0788 \\ \Delta w_{42} = 1.0 \cdot (-0.3218) \cdot (-0.8483) = 0.2730 \\ \Delta w_{43} = 1.0 \cdot (-0.3218) \cdot 1 = -0.3218 \end{cases}$$

EBP Error Back Propagation algorithm *example*

$k = 0.5, \ \alpha = 1.0$

8. Perform layer B weight update

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$$



Neuron 3

$$\begin{cases} w_{31} = w_{31} + \Delta w_{31} = 1 - 0.0043 = 0.9957 \\ w_{32} = w_{32} + \Delta w_{32} = -1 - 0.0150 = -1.0150 \\ w_{33} = w_{33} + \Delta w_{33} = 2 + 0.0177 = 2.0177 \end{cases}$$

Neuron 4

$$\begin{cases} w_{41} = w_{41} + \Delta w_{41} = 2 + 0.0788 = 2.0788 \\ w_{42} = w_{42} + \Delta w_{42} = -2 + 0.2730 = -1.7270 \\ w_{43} = w_{43} + \Delta w_{43} = 1 - 0.3218 = 0.6782 \end{cases}$$

EBP Error Back Propagation algorithm *example*

Algorithm steps:

- *repeat the procedure for each pattern*
- *after applying all the patterns, start next iteration*
- *exit the algorithm once the TE is < req.error*

Problems:

- *Oscillations*
- *Slow convergence or no convergence*
- *Flat spot, local minima traps*

Things to remember…

- **EBP introduced…**
  - *Multi-layer, multi-output architecture*
  - *mapping of n-dimensional to m-dimensional vector*
  - *soft-activation function (historically)*

- **EBP represented…**
  - *A go-to algorithm for a long time (still in use, new life with deep learning)*
  - *Robust and reliable choice (though not a snake oil, e.g. slow/no convergence, oscillations, etc.)*