

Learning In Networks: From Spiking Neural Nets To Graphs

Victor Miagkikh
Machine Learning Specialist,
Cisco Systems Inc.
Ironport business unit

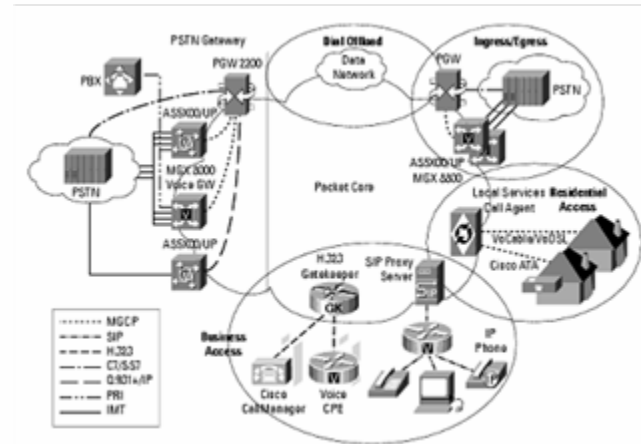
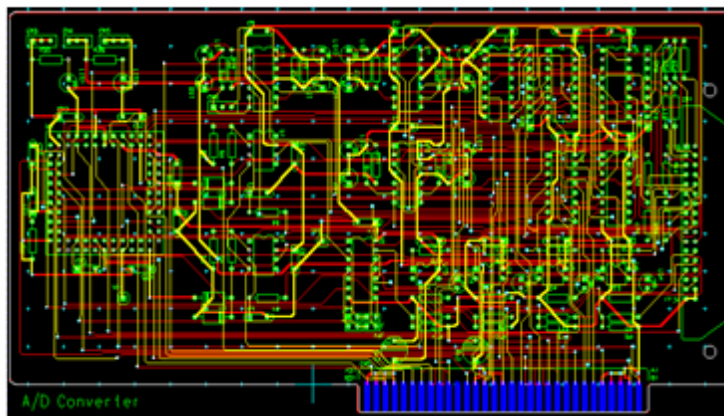
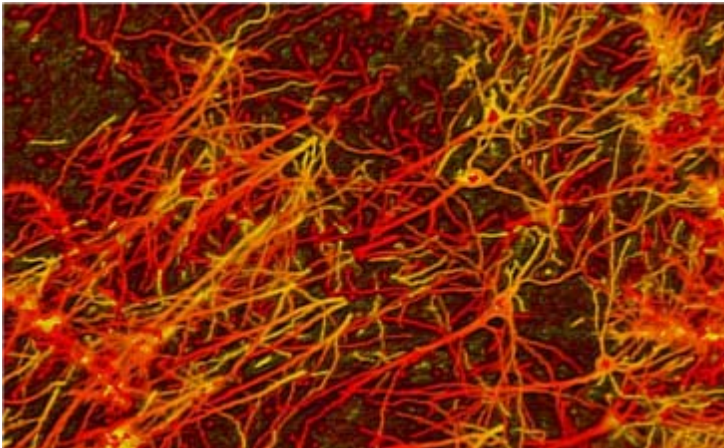


Roadmap

- Introduction to Hebbian learning
- Bee navigation problem
- Introduction to Spiking Neural Networks(SNN)
- rHebb algorithm: Hebbian learning augmented with reward controls plasticity learning principle
- Reinforcement learning applications for movies and stock purchase recommendations
- Methodology for introducing reinforcement learning in networks
- Q&A

Networks

- There are many different kinds of networks around us

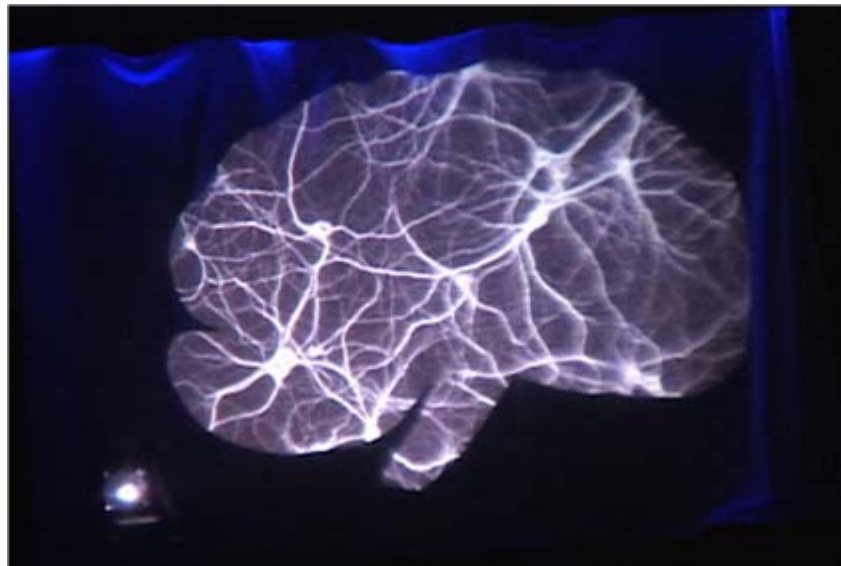


Learning in Networks



- Suppose we would like to add a connection between San Francisco and Orlando. Is it a good thing to do?

Neural Networks



- There is no external analyst in the brain who changes connections between neurons. What principles make learning possible?

Hebbian Learning



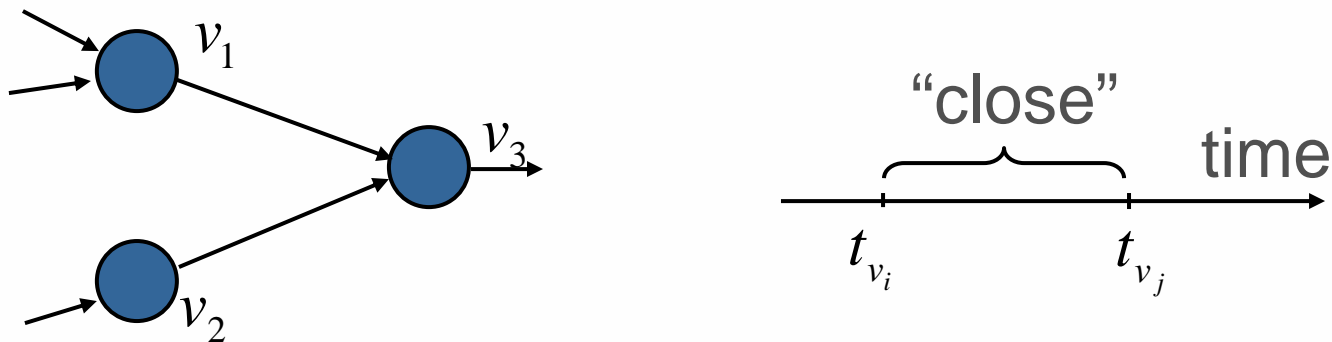
Donald Olding Hebb
1904-1985

- Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability....
When an axon of cell *A* is near enough to excite a cell *B* and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that *A*'s efficiency, as one of the cells firing *B*, is increased. (Hebb, *The organization of behavior*, 1949)

Hebbian Learning (Cont.)

- In other words: if two neurons fire “close in time” then strength of synaptic connection between them increases.

$$\Delta w_{ij}(t) = \eta v_i v_j g(t_{v_i}, t_{v_j})$$

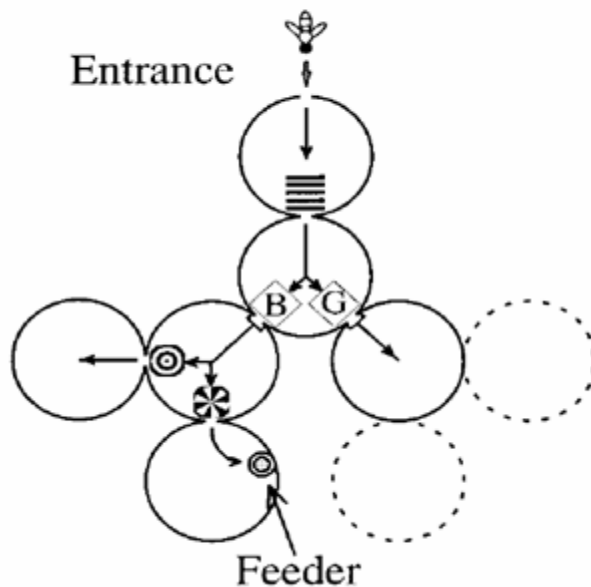


- Weights reflect correlation between firing events.

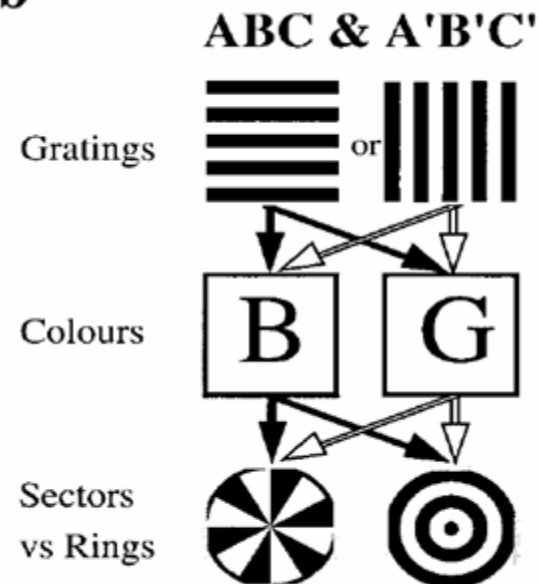
Bee Navigation Problem

- A bee has to correlate signs in order to find the feeder (S. W. Zhang et al., Honeybee Memory: Navigation by Associative Grouping and Recall of Visual Stimuli, 1998).

a



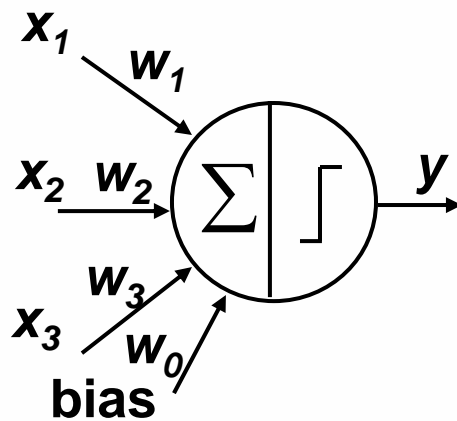
b



Spiking Neural Networks

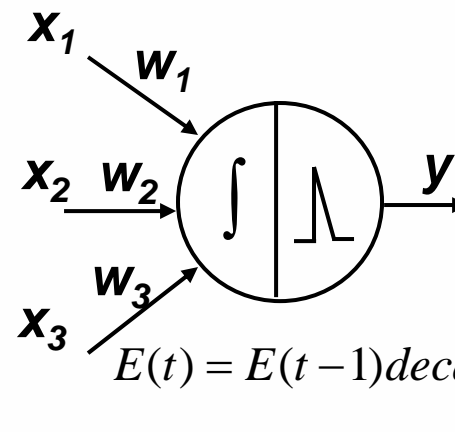
- Short term memory is needed in order to solve this problem.
- Spiking Neural Networks (SNN) have “built in” short term memory.

Perceptron



$$y = \text{Activation_Function}\left(\sum w_i x_i\right)$$

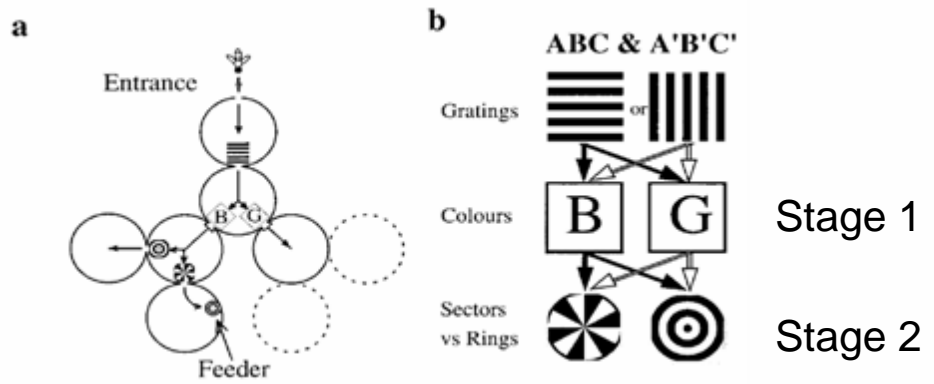
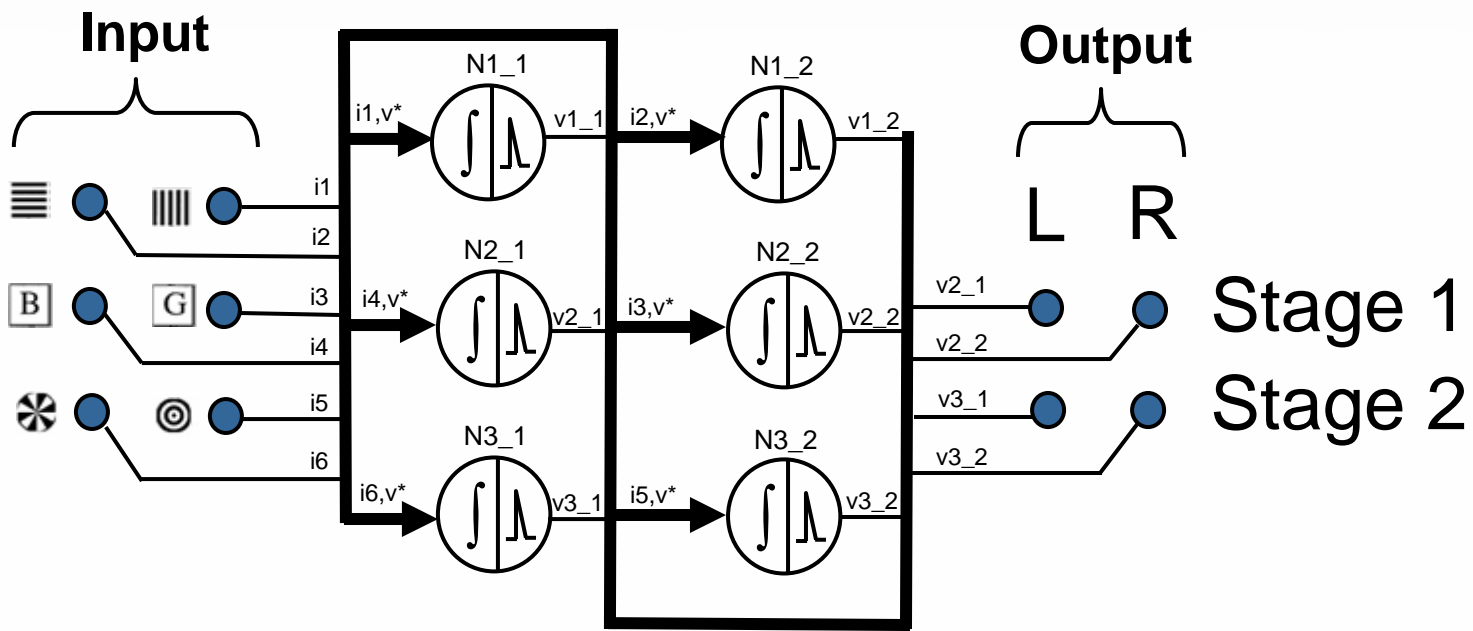
Spiking Neuron



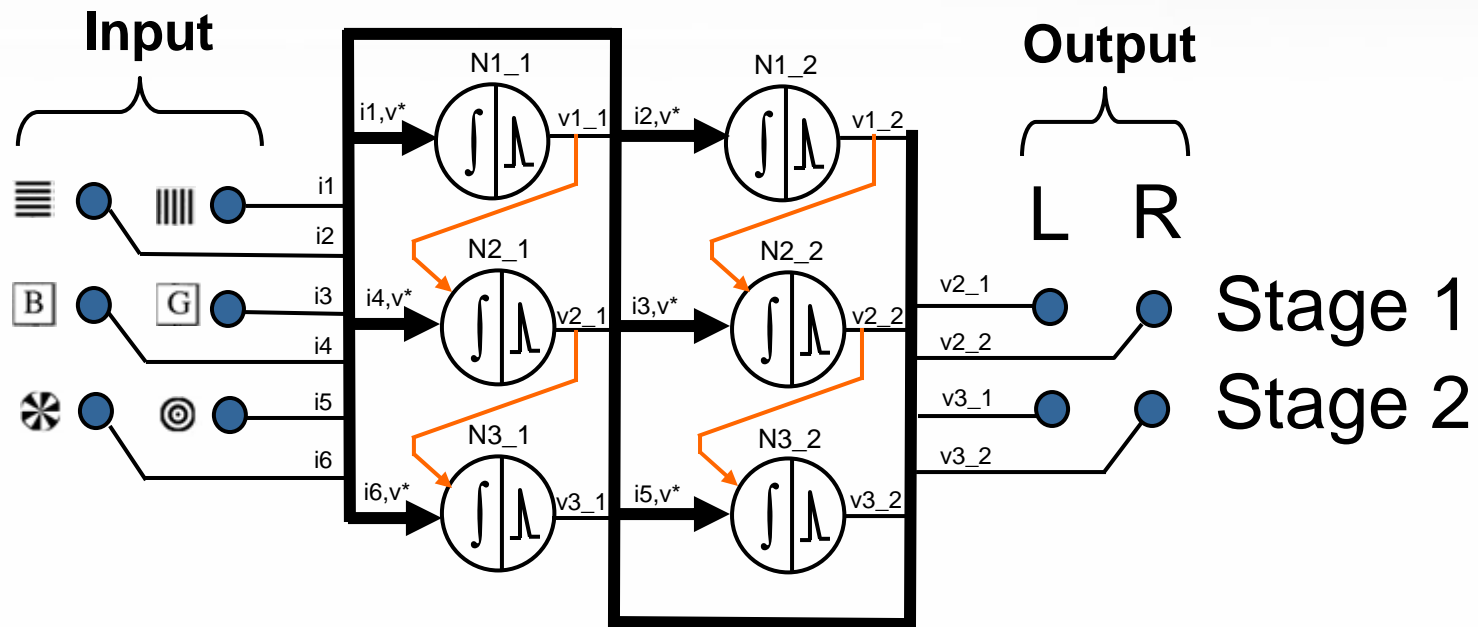
$$E(t) = E(t-1)\text{decay} + \int_{t-1}^t \sum w_i x_i$$

If $E(t) > \text{threshold}$ -> emit spike; $E(t+1) = 0$

SNN Solution



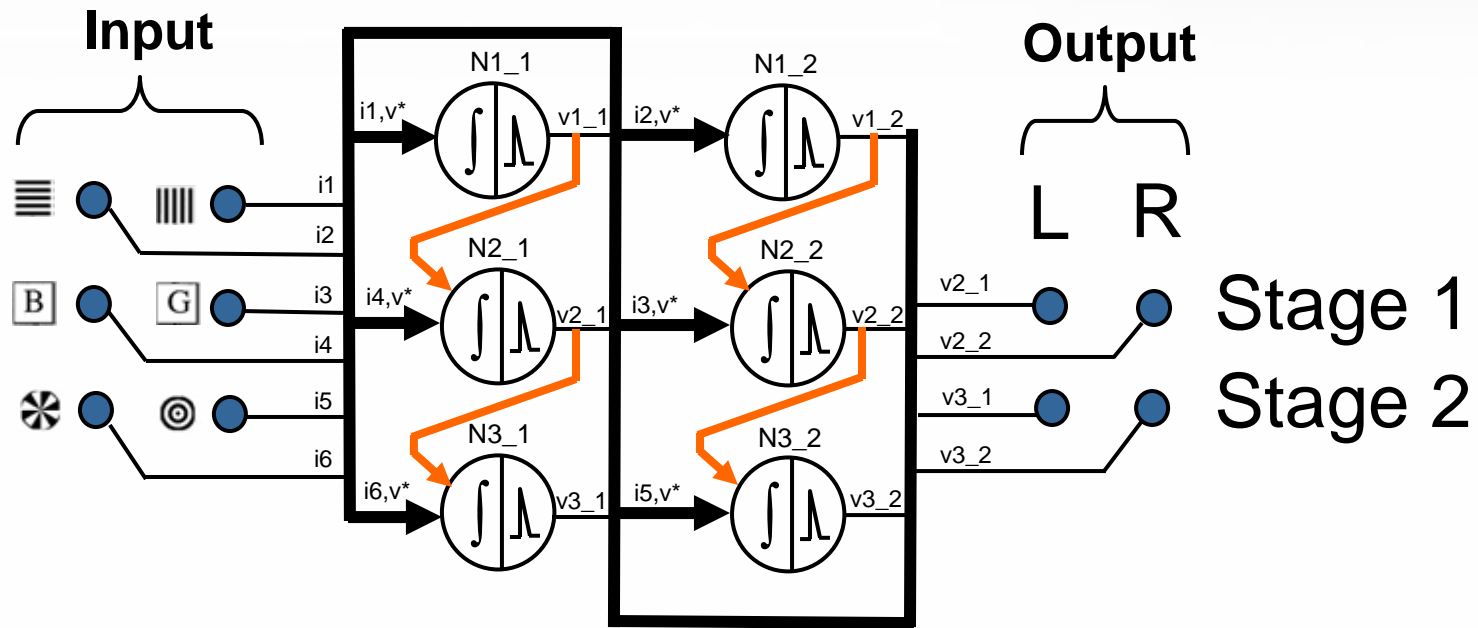
Evolution of Weights



Iteration 25

	N1_1	N1_2	N2_1	N2_2	N3_1	N3_2
N1_1	N/A	0.0872	0.5188	0.1881	1.0356	0.2256
N1_2	0.0872	N/A	0.17953	0.4338	0.0831	0.4935
N2_1	0.5188	0.17953	N/A	0.0931	2.4867	0.2107
N2_2	0.1881	0.4338	0.0931	N/A	0.1325	0.31944
N3_1	1.0356	0.0831	2.4867	0.1325	N/A	0.1296
N3_2	0.2256	0.4935	0.2107	0.31944	0.1296	N/A

Evolution of Weights (Cont.)



Iteration 50

	N1_1	N1_2	N2_1	N2_2	N3_1	N3_2
N1_1	N/A	0.0743	0.9007	0.1819	2.6617	0.2256
N1_2	0.0743	N/A	0.0973	1.7223	-0.6328	0.6212
N2_1	0.9007	0.0973	N/A	0.0652	8.579	0.2107
N2_2	0.2107	1.7223	0.0652	N/A	0.0453	0.4476
N3_1	0.6212	-0.6328	8.579	0.0453	N/A	0.0978
N3_2	0.2256	0.6212	0.2107	0.4476	0.0978	N/A

rHebb Algorithm

Assumption#1 - Hebbian Learning:

The strength of the connections between the neurons that fire “close in time” increases (subject to assumption#2).

$$\Delta w_{ij} = \eta \cdot f(v_i, v_j, \Delta t_{ij}) = \eta \cdot v_i \cdot v_j \cdot g(\Delta t_{ij}) = \eta \cdot v_i \cdot v_j \cdot e^{-k_1 \Delta t_{ij}}$$

v_i, v_j outputs of neurons i and j

Assumption#2

Δt_{ij} - difference in time between firings of neurons i and j

Reward controls plasticity (the ability to change weights)

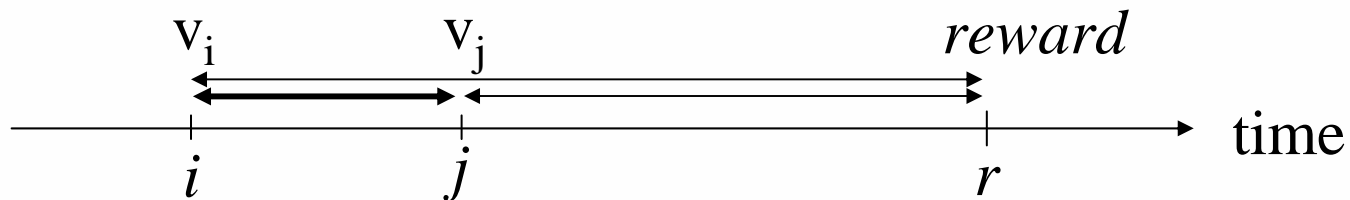
Therefore, the changes in weights as defined by Hebbian learning occur not all the time but are controlled in **sign** and **intensity** by the reward.

$$\Delta w_{ij} = \eta \cdot \text{reward} \cdot v_i \cdot v_j \cdot g(\Delta t_{ij}) = \eta \cdot \text{reward} \cdot v_i \cdot v_j \cdot e^{-k_1 \Delta t_{ij}}$$

rHebb Algorithm (Cont.)

Assumption#3

Temporal credit assignment



$$\Delta w_{ij} = \eta \cdot \text{reward} \cdot v_i \cdot v_j \cdot g(\Delta t_{ij}) \cdot h(\Delta t_{ir}, \Delta t_{jr}) =$$
$$\Delta w_{ij} = \eta \cdot \text{reward} \cdot v_i \cdot v_j \cdot e^{-k_1 \Delta t_{ij}} \cdot e^{-k_2 (\Delta t_{ir} + \Delta t_{jr})}$$

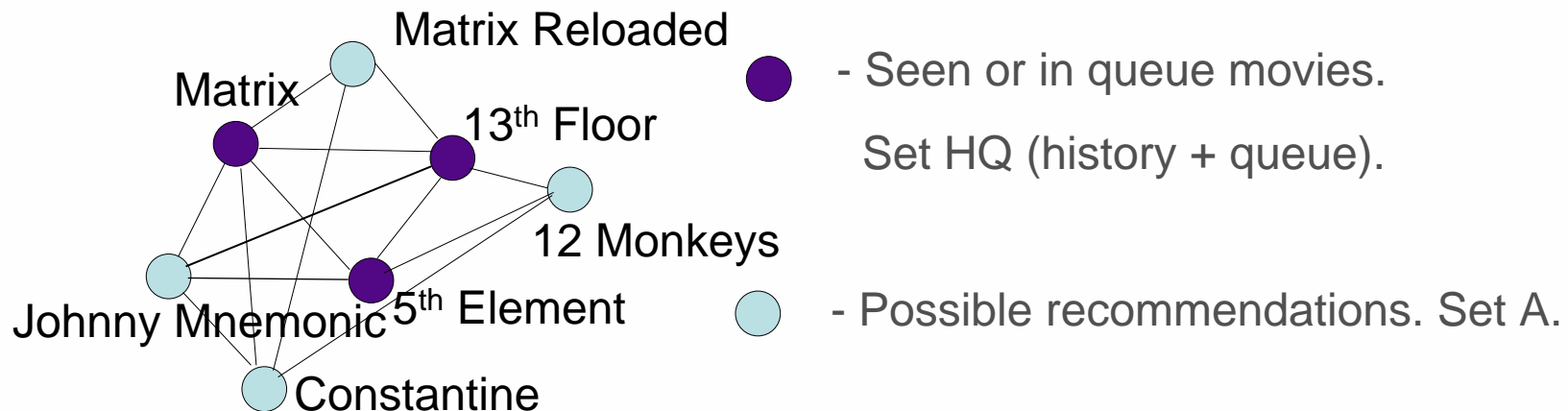
$$\Delta w_{ij} = \eta \cdot \text{reward} \cdot v_i \cdot v_j \cdot e^{-k_1 \Delta t_{ij} - k_2 \Delta t_{ir} - k_2 \Delta t_{jr}}$$

rHebb Algorithm (Cont.)

- Weights in rHebb reflect not correlations, but utilities.
- Related work: C.M. Pennartz, Reinforcement Learning by Hebbian Synapses with Adaptive Thresholds, Neurocince, 1997 (Caltech).
- rHebb is biologically plausible.
- Could “reward control plasticity” learning axiom be used for training not only spiking neural networks, but other kinds of networks?

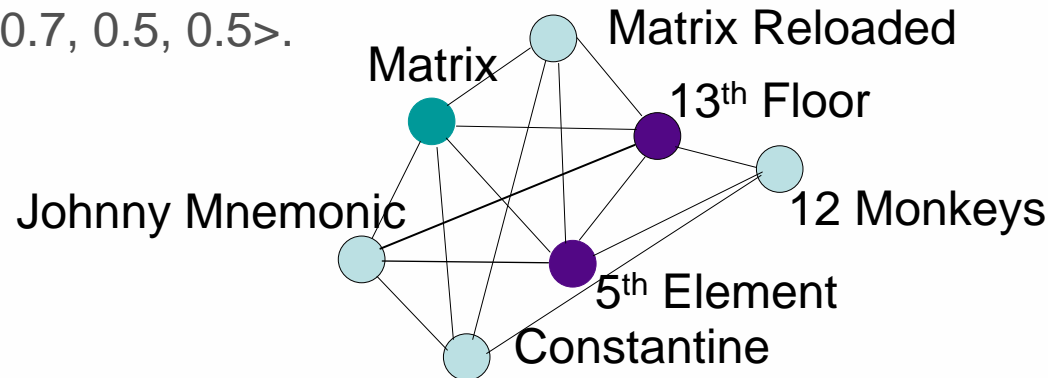
Movie Recommendation Problem

Problem: given a list of favorite movies, a wish list, and demographic information (age, gender, zip code) try to come up with a movie recommendation list.



Reinforcement Learning in Movie Network

- Let's assume that for some movies in H set we have personal ratings. For combined HQ set we can introduce a reward function $G(HQ_i)$ that returns 0.5 for unseen, and neutral movies, +1 for the best movie ever made, and 0 for the worst movie according to a personal taste.
- For example: $H = \langle 13^{\text{th}} \text{ Floor}, 5^{\text{th}} \text{ Element} \rangle$, $Q = \langle \text{Matrix} \rangle$.
 $G(HQ_i) = \langle 0.7, 0.5, 0.5 \rangle$.



- Let's introduce the set R that is composed of movies that our system recommended. For example above, let $R = Q = \langle \text{Matrix} \rangle$
- At some point user watches a movie in R or some other movie not in R . If user watches a movie in R system gets a reward $G(HQ_i)$. If not then system gets reward = - small_penalty. Thus, we can get a sequence of $\langle m, \text{reward} \rangle$ pairs for each user. Let's call this set T .

Reinforcement Learning in Movie Network (Cont.)

- The set T could be used to adjust connections in C_{ij} in reinforcement learning mode. Consider a $\langle m, reward \rangle$ pair. Given m we know average cumulative flow through each edge and vertex to all vertices in H set. Let's denote them as $fn(i, m, H)$ and $fe(i, j, m, H)$.

- Then we can update edges as follows:

$$\Delta C_{ij} = \eta_e \cdot reward \cdot fe(i, j, m, H)$$

- And vertices:

$$\Delta C_{ii} = \eta_n \cdot reward \cdot fn(i, m, H)$$

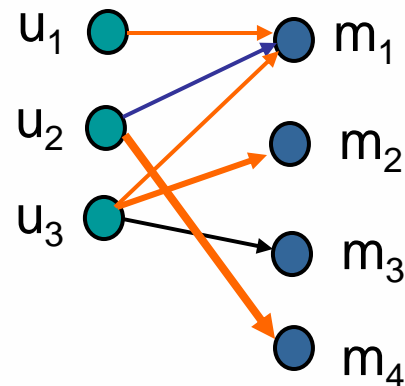
- Thus, we got updated C_{ij} and we can do it for entire population, a segment or a customer.

Reinforcement Learning in Networks

- Generally, flow in network and Hebbian learning are examples of eligibility trace $e(t)$ that is a degree of participation of each functional unit in decision making that caused the action.
- $e(t) * reward$ pattern is an example of doing structural credit assignment in a learner. It's very intuitive: distribute reward according to participation.
- Structural credit assignment is different in each kind of learner, but the pattern $e(t) * reward$ is a good rule of thumb to introduce RL.
- In general, RL can have some place when the only feedback that we have from environment is a reinforcement signal that doesn't allow the use of supervised learning.

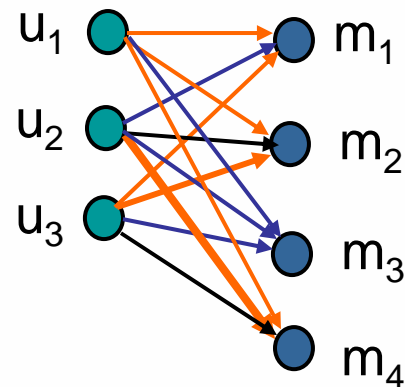
An Approach for Solving NetFlix Problem

- Bipartite graph representation:



	m1	m2	m3	m4
u1	0.3	0	0	0
u2	-0.1	0	0	0.9
u3	0.1	0.4	-0.2	0

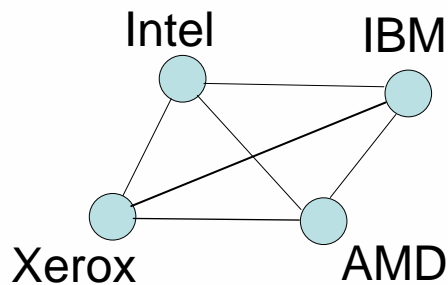
- After finding network equilibrium:



	m1	m2	m3	m4
u1	0.3	0.1	-0.1	0.2
u2	-0.1	0.1	0.15	0.9
u3	0.1	0.4	-0.2	0.05

Example: Influence Networks for Economy

- Consider the following network where nodes represent companies and connections represent mutual influences. Stock price could be considered as the output of a company node similar to the output of a neuron.
- We can apply Hebbian learning = the strength of connection between two neurons that fire at the same time increases.



Input: stock price $s(t)$ for any company c_i

Output: Strength of influence connections $w(c_i, c_j)$

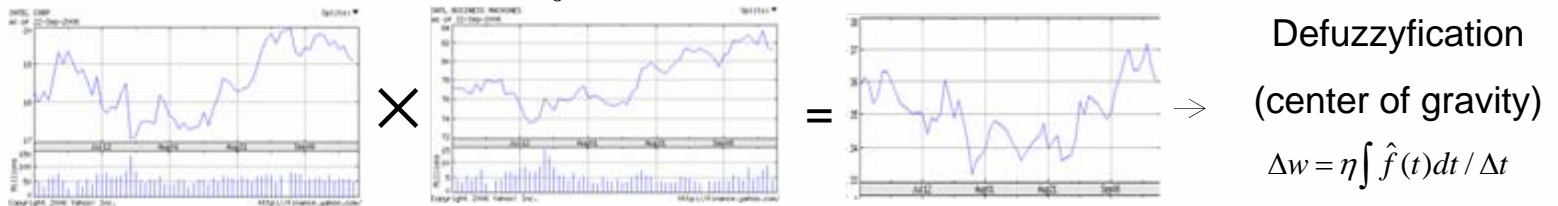
Learning Rule (Hebbian learning):

$$\Delta w_{c_i c_j}(\Delta t) = \eta(\Delta s_i(\Delta t) \cdot \Delta s_j(\Delta t))$$

Example: Influence Networks for Economy (Cont.)

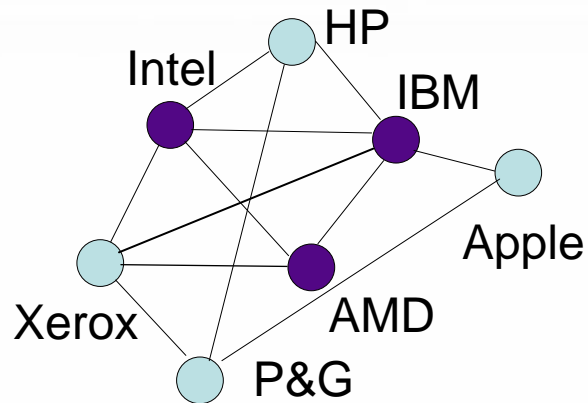
- We can easily extend simple Hebbian Learning to “Fuzzy” Hebbian Learning:

$$\Delta w = \eta \int_0^{\Delta t} f_{c_i}(t) f_{c_j}(t) dt / \Delta t$$



- We can train network for various delta t . The meaning of weight becomes timed correlation of stock price.

Portfolio Analyses



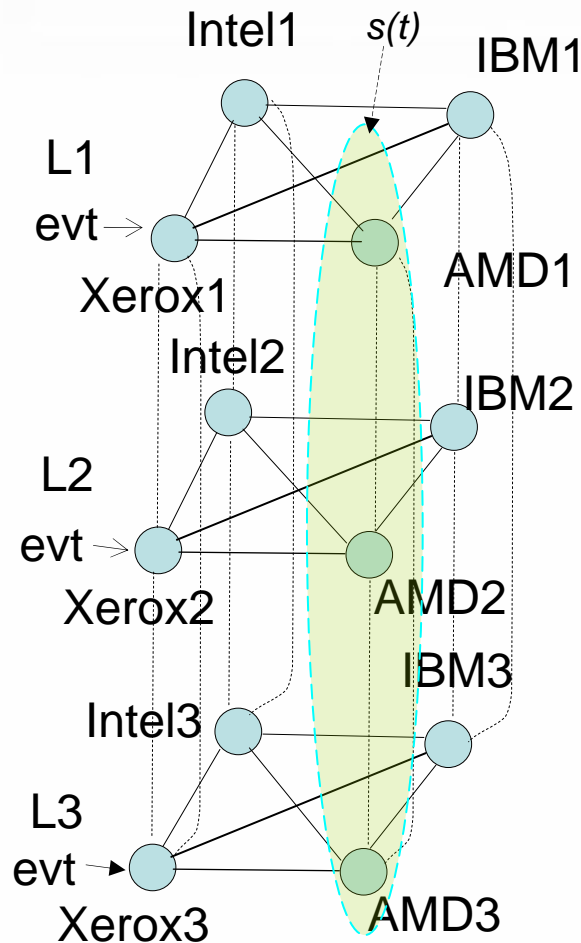
● - some stock in portfolio

○ - no stock in portfolio

In general: portfolio is represented by weights on vertices

- Characteristics we can compute: diversity, coverage, competition within portfolio, company influence.
- Maximum flow between any two vertices could be used to calculate cumulative influence.
- Key idea in stock purchase recommendations: if a fluctuation in a stock price of one company is observed then the network can tell the stock price of which companies will start to fluctuate.

Multi Aspect Influence Model



Stock price could be seen as a form of reinforcement signal reflecting future expected reward of a company allowing the use of reinforcement learning.

Each “plane” L_i models separate dimension of influences. It can represent involvement of a company in different sectors of economy. In general, each plane could be a result of decomposition into some other more abstract basis.

Each company node c_j exist on each L_i . Forming vertical subnetworks that could be used to model transformations within a company.

Reward Controls Plasticity Learning Principle

- $-e(t)$ * *error* pattern used in supervised learning for long time.
- $e(t)$ * *reward* is a generic principle to introduce reinforcement learning.
- $e(t)$ could be any causality inducing principle, not necessarily Hebbian learning.
- There is huge area of applicability. Depending on application, reward could be defined as number of users clicking on online ad or number of messages caught by anti-spam system.
- Q&A