INT301 W13

SELF-ORGANIZING FEATURE MAP

Biological Motivation

大脑是一个 self-organizing system，它可以通过改变（增加、移除、加强）神经元之间的相互联系进行自我学习。

具有相似功能的神经元被组合在一起。

大脑在“2”-dimensional internal map 中处理来自外部世界的多维信号。

Feature Maps

brain’s self-organization 的结果：

- 在大脑中形成具有线性或平面拓扑的 feature maps（即，它们在一个或两个维度上延伸）

大脑将世界的外部多维表征映射到类似的 1 或 2 维内部表征（internal representation）中。也就是说，大脑以拓扑保存的方式（topology-preserving way）处理外部信号。

Topographic Maps

扩展竞争性学习的想法，以纳入输入和神经元的 neighborhood。

我们希望将 input pattern space 非线性地转换为 output feature space，从而保留输入之间的 neighborhood relationship。

- 附近神经元响应类似输入的 feature map
- 神经元有选择地调整到特定的输入模式，使神经元相对于彼此有序，以便为不同的输入特征创建有意义的坐标系

空间位置指示输入模式的内在统计特征：在输入中相近的，在输出中也要相近。

这个方法的思路就是，对多维的输入进行映射，使它们变成 **1** 或 **2** 维的数据，同时保持它们之间的关系。比如：类似 **(**或相近**)** 的输入都放在一块。

结合竞争和合作机制，使用无监督学习网络生成特征图。

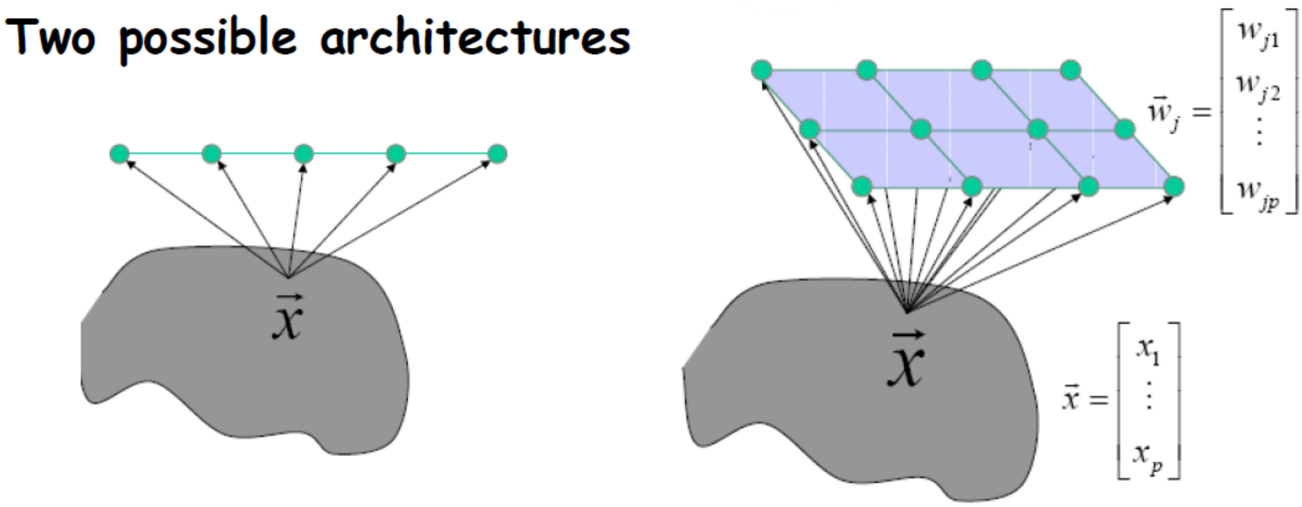
Activity-based self-organization：学习后，会出现 topographic map。但是，输入维度与输出维度相同。

Kohonen 简化了这个模型，并将其称为 Kohonen’s self-organizing map (SOM) algorithm。

- 更通用，因为它可以执行降维 (dimensionality reduction)
- SOM 可以被视为 vector quantization type algorithm (矢量量化类型算法)

Self-Organizing Map (SOM)

SOM 中的想法是将任意维度的输入转换为 1 或 2 维的离散映射。



同样，2层神经元，所有输入都连接到每个输出。输出神经元保持在一维或（通常）2 维的 lattice (晶格) 中，其中晶格中的位置定义了神经元之间的距离。

一旦完成了权重的初始化，算法包括 3 个过程：

1. Competition

   给定一个 input pattern，输出相互竞争，根据一个判别函数 (discriminant function，例如输入向量和权重向量的相似性) 来判断谁是 winner。

2. Cooperation

   Winning 神经元确定 topological neighborhood (拓扑学上的 neighborhood) 的空间位置，其中的输出神经元激发 (excited)

3. Synaptic Adaptation (突触适应)

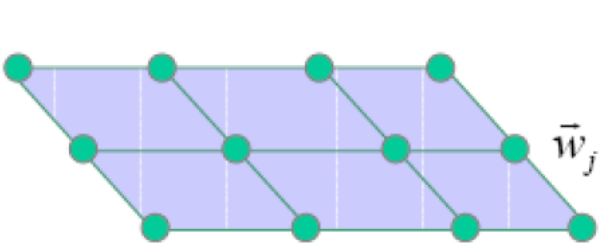
   Excite neurons 调整权重，使判别函数的值增加（类似的输入将导致 winner 的响应增强）

Learning Principle:

竞争性学习，其中胜利"溢出"给 neighbors。

Initialization

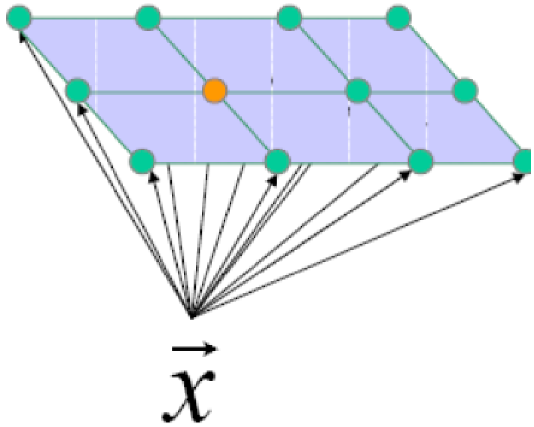
网格 (Grid)：大小和结构先验固定（大多数时候，使用二维网格）



$$\vec{w}_j = \begin{bmatrix} w_{j1} \\ w_{j2} \\ \vdots \\ w_{jp} \end{bmatrix}$$

Competitive Process

activation patterns 的连续输入空间，通过网络中神经元之间的竞争过程映射到神经元的离散输出空间。



Winner neuron

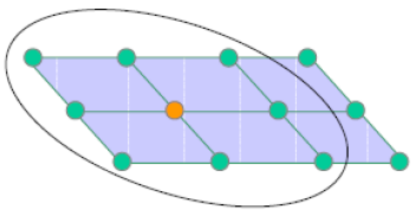
$$= \arg\max_j (w_j^T x)$$

$$= \arg\min_j (\| x - w_j \|)$$

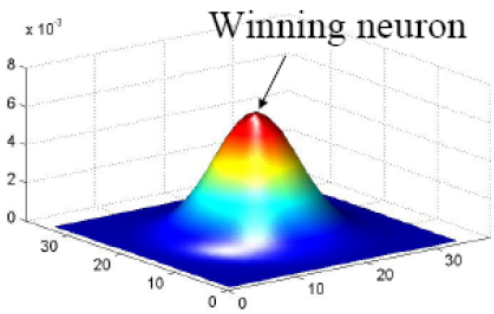
Cooperative Process

获胜的神经元定位了合作神经元的拓扑邻域的中心。

正在放电的神经元更倾向于激发其邻近的神经元，而不是远离它的神经元。



$$h_{j,i} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right)$$



The topological neighborhood $h_{j,i}$:

- 围绕 winning neuron 对称，并在获胜神经元处达到其最大值
- 振幅 (amplitude) 随着横向距离的增加而单调减小

The topological neighborhood $h_{j,i}$ 随时间缩小：

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_1}\right) \qquad h_{j,i}(t) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(t)}\right)$$

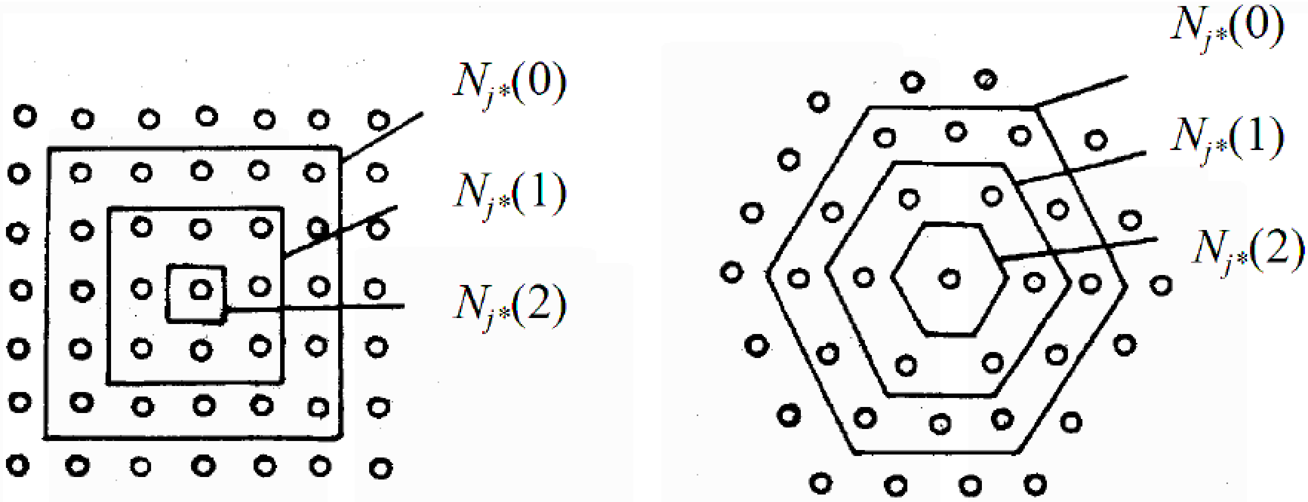
获胜节点的 Neighbors 也允许更新，即使他们没有获胜。

Neighborhood

大量的 neighborhood 会让你有一个良好的开始 (good global ordering)，但可能导致收敛困难 (bad local fit)。

少量的 neighborhood 恰好与上面相反：bad global ordering，good local fit。

因此通过随时间逐渐缩小 neighborhood ，我们可以获得最佳效果。



我们可以选择矩阵型或者六边形型的平面阵列来确定我们的 neighborhood，N_{j*} 的括号里代表的是时间 (随时间缩小 neighborhood)。输入向量 x 同时应用于所有节点。

Adaptive Process

更新与输入相关的权重：

$$\mathrm{w}_j(t+1) = \mathrm{w}_j(t) + \boxed{\eta(t)}\boxed{h_{j,i}}(t)(\mathrm{x} - \mathrm{w}_j(t))$$

Learning rate

Neighborhood function

$$\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_1}\right)$$

$$h_{j,i}(t) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(t)}\right)$$

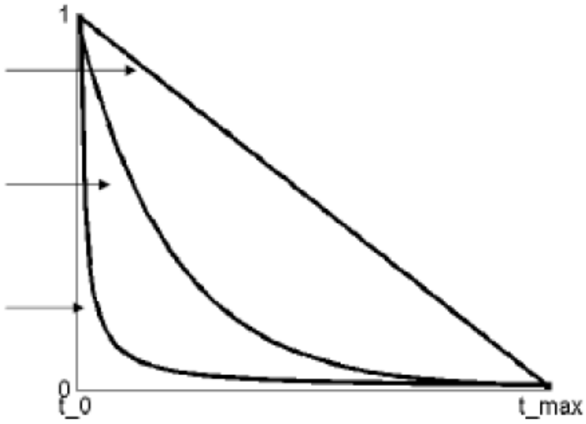
Learning Rate

Possible options:

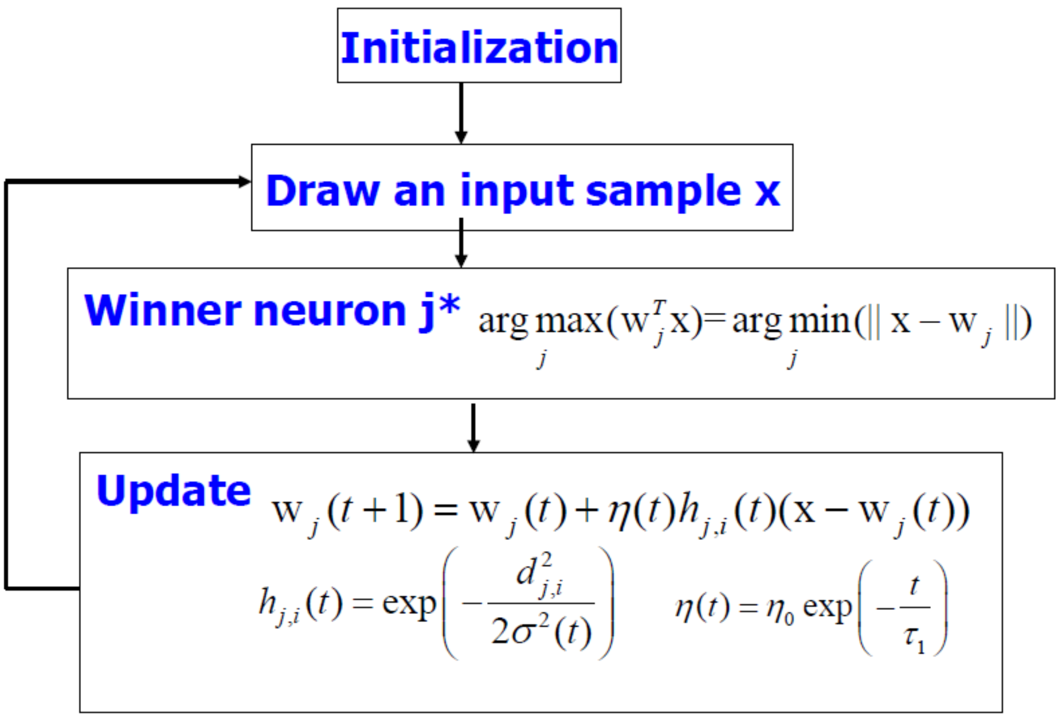
$$\eta(t) = -at + b$$

$$\eta(t) = \exp(-at) + b$$

$$\eta(t) = \frac{1}{at} + b$$



Summary



Property of SOM

- Approximate the input space
- Topological ordering
- Density matching
- Feature selection (features of the underlying distribution)

More on Vector Quantization

在 VQ 技术中，形成了许多局部中心来表示输入向量。

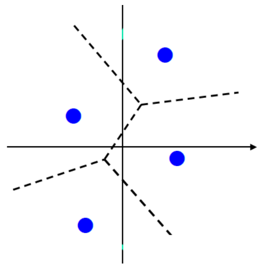
对于拥有 M 个参考向量 (reference vectors，或中心) $\{w_1, \ldots, w_M\}$ 的集合，如果输入向量 x 和 w_k 的距离最小 ($||x - w_k||^2$ 最小)，那么我们就认为 x 和 w_k 最匹配。

参考向量把输入空间 R^L 分成 Voronoi cells/polygons 定义为:

$$V_k = \left\{\mathrm{x} \in R^L \mid \|\mathrm{x} - \mathrm{w}_k\| \leq \|\mathrm{x} - \mathrm{w}_l\|, \forall l\right\}$$

注：上式的意思就是，把 x 分到和它最相似的 Voronoi cell 中。

参考向量的集合叫做 codebook，其中的参考向量叫 code vectors。

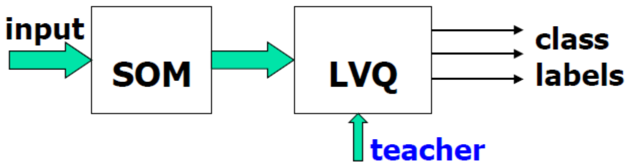


类似上面这样的多边形区域就是 Voronoi cell。

SOM 算法提供了一种以无监督方式计算 Voronoi 向量的近似方法。

Learning Vector Quantizer (LVQ)

LVQ 是一种监督学习技术，它使用类信息稍微移动 Voronoi 向量，以提高分类器决策区域的质量。



LVQ1

从输入空间中随机选取输入向量 x。

如果输入向量 x 和 Voronoi 向量 w 的类标签一致，则 Voronoi 向量 w 沿输入向量 x 的方向移动。

如果输入向量 x 和 Voronoi 向量 w 的类标签不一致，则 Voronoi 向量 w 将远离输入向量 x。

if the winner belongs to the right class

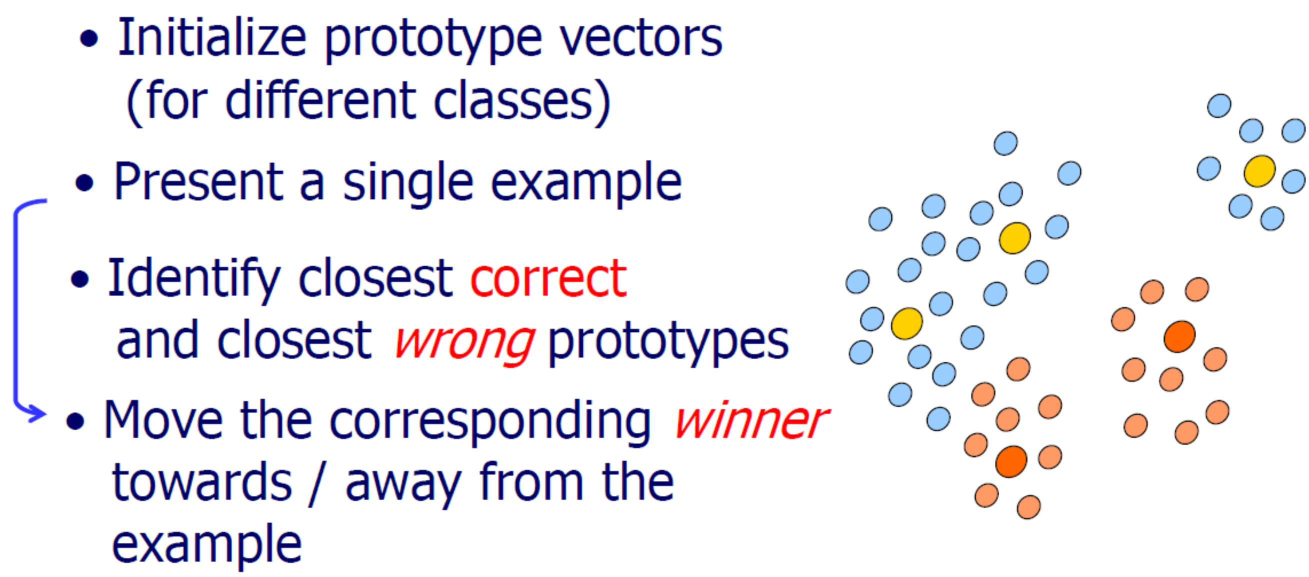
$$\mathrm{w}^{new} = \mathrm{w}^{old} + \eta(\mathrm{x} - \mathrm{w})$$

if the winner belongs to the wrong class

$$\mathrm{w}^{new} = \mathrm{w}^{old} - \eta(\mathrm{x} - \mathrm{w})$$

LVQ2

LVQ2 和 SCL 过程类似，不过它和 LVQ1 的区别在于：LVQ1 只能将 Voronoi 向量远离或接近，而 LVQ2 可以同时远离和接近。



LVQ Discussion

停止条件:

- Codebook vectors 已稳定
- 或已达到最大 epoch 数

Advantages:

- 看起来合理、直观、灵活
- 快速且易于实施
- 经常应用于涉及结构化数据分类的各种问题

Disadvantages:

- 对于重叠类不稳定
- 对初始化非常敏感

ASSOCIATIVE MEMORIES

标准计算机内存是通过分配的地址进行访问的。当用户搜索文件时，CPU 必须将请求转换为数字指令，然后在内存中搜索相应的地址。

计算机的内存通常称为RAM（随机存取存储器）。

Associative Memory & Pattern Association

关联内存 (associative memory) 是一种内容可寻址结构 (content-addressable structure)，它将一组输入模式映射到一组输出模式。

也就是说，内存可以由内容直接访问，而不需要再把请求转换为内存中的物理地址。

关联内存 (associative memory) 通常与关联模式 (pattern association) 有关:

- Associating patterns which are
  - similar

- contrary
- in close proximity (spatial)
- in close succession (temporal)

- Associative recall
  - evoke associated patterns
  - recall a pattern by part of it
  - evoke/recall with incomplete/noisy patterns

Associative Memories (AM)

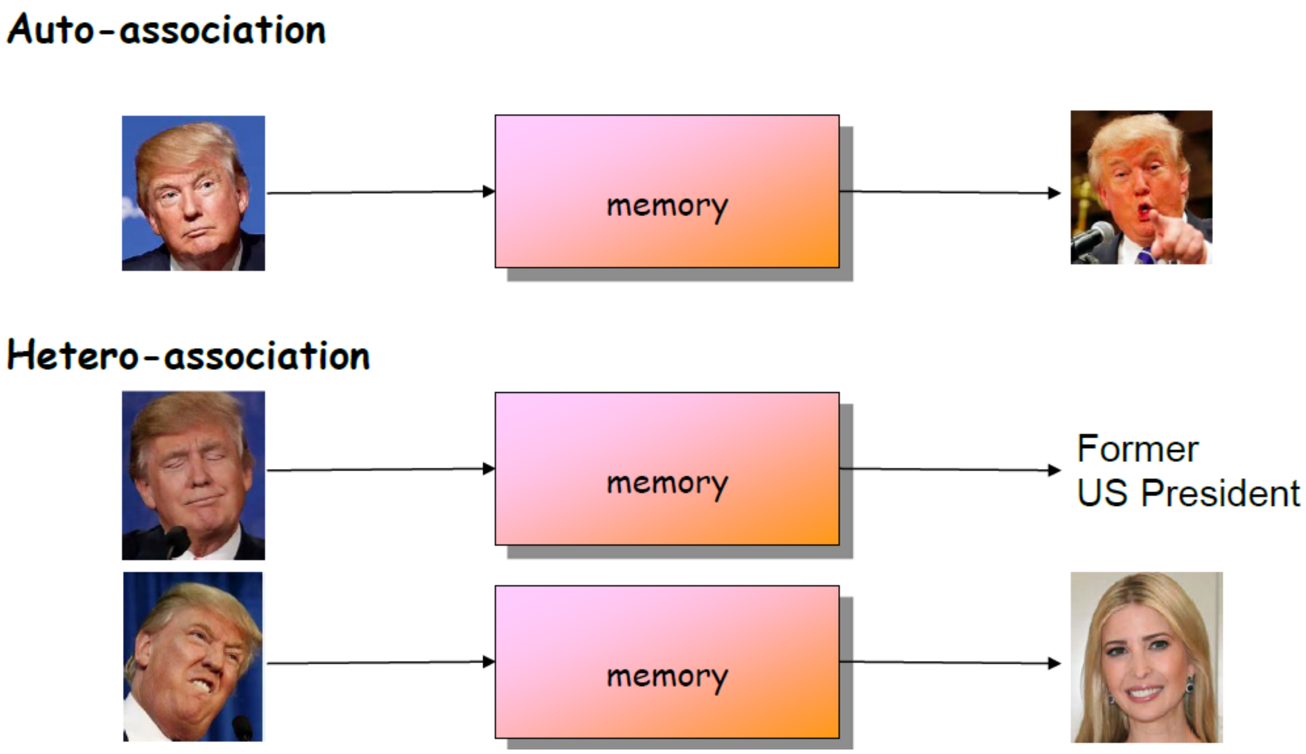
两种类型的 associative memory：auto-associative 和 hetero-associative (异质关联)。

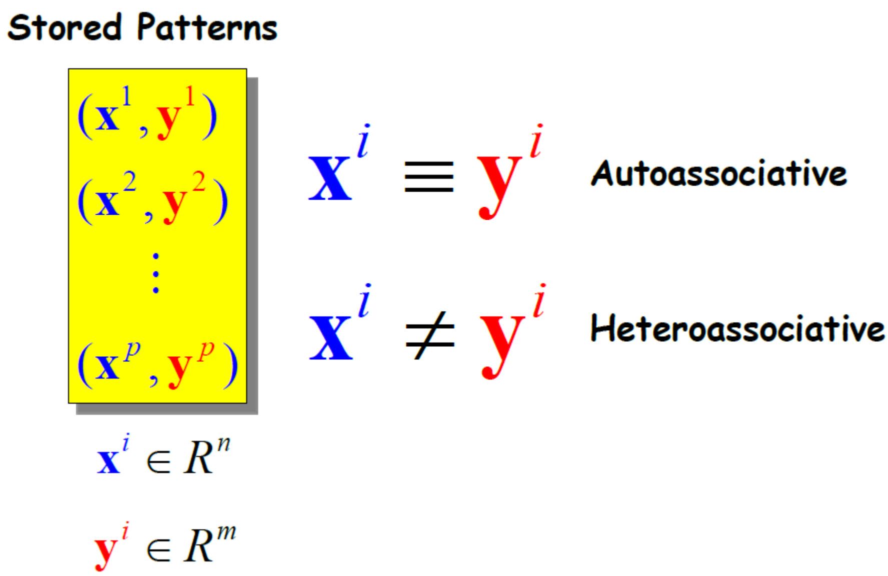
- Auto-association

  检索以前存储的与当前模式最相似的模式

- Hetero-association

  检索到的模式不仅在内容上，而且在类型和格式上也可能与输入模式不同

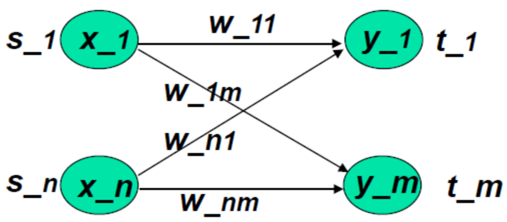




Simple AM

Network structure: single layer

- 一个非线性 unit 的输出层和一个输入层
- 类似于简单的分类网络



通过训练 pattern pairs {s:t}，得到一组权重 w_ij，从而使输入 s 时可以得到输出为 t。

Learning algorithm

和 Hebbian learning 类似。

Algorithm: (bipolar or binary patterns，双极性 (0/1 或 ON/OFF) 或二进制 (+1/-1) 模式)

- 对于 training sample $s : t$ $\Delta w_{ij} = s_i \cdot t_j$
- 如果输入和输出均为 ON（二进制）或具有相同的符号（双极性），则 Δw_{ij} 增加
- 如果初始化 $\Delta w_{ij} = 0$，在更新所有 P 个 training patterns 之后：

$$w_{ij} = \sum_{p=1}^{P} s_i(p) t_j(p) \quad W = \{w_{ij}\}$$

- 它不是通过迭代更新来获得 W，而是可以通过从训练集中计算 s 和 t 的外积（outer product）得到

Outer product：

- 让 s 和 t 变成行向量，然后对于特定的训练对 s:t

$$\Delta W(p) = s^T(p)\cdot t(p) = \begin{bmatrix} s_1 \\ \\ \\ s_n \end{bmatrix} [t_1, \ldots\ldots t_m] = \begin{bmatrix} s_1 t_1 \ldots\ldots s_1 t_m \\ s_2 t_1 \ldots\ldots s_2 t_m \\ \\ s_n t_1 \ldots\ldots s_n t_m \end{bmatrix} = \begin{bmatrix} \Delta w_{11} \ldots\ldots \Delta w_{1m} \\ \\ \\ \Delta w_{n1} \ldots\ldots \Delta w_{nm} \end{bmatrix}$$

and $W(P) = \sum_{p=1}^{P} s^T(p)\cdot t(p)$

我们使用 training sample 进行 recall：

- 使用 s(k) 作为输入，希望输出为 t(k)，e.g. $f(s(k)W) = t(k)$

$$\begin{aligned} s(k)W &= s(k)\sum_{p=1}^{P} s^T(p)\,t(p) = \sum_{p=1}^{P} s(k)\cdot s^T(p)\cdot t(p) \\ &= s(k)\,s^T(k)\,t(k) + \sum_{p\neq k} s(k)\,s^T(p)\,t(p) \\ &= \underbrace{\|s(k)\|^2\; t(k)}_{\text{principal term}} + \underbrace{\sum_{p\neq k} s(k)\,s^T(p)\,t(p)}_{\text{cross-talk term}} \end{aligned}$$

principal term 给出了 s(k) 和 t(k) 的关联。

Cross-talk term 表示 s(k):t(k) 与其他训练对之间的相关性。当 cross-talk 很大时，s(k) 会想起 t(k) 以外的其他东西 (principal term 和 cross-talk 会得到两个值，当 cross-talk 占比大时，会使结果误差大，即想起其他东西)。

如果所有 s(p) 垂直于其他 training sample，则 $s(k)\cdot s^T(p) = 0$，那么除了 s(k):t(k) 之外，没有其他样本对结果有贡献。不过，在 n 维空间中最多存在 n 个正交向量 (orthogonal vectors)。

当 p 增加时，cross-talk 增加 (相加的东西多了，结果自然变大)。

Example 1: hetero-associative

- Binary pattern pairs $\boldsymbol{s}:\boldsymbol{t}$ with $|\boldsymbol{s}| = 4$ and $|\boldsymbol{t}| = 2$.
- Total weighted input to output units: $y\_in_j = \sum_i x_i w_{ij}$
- Activation function: threshold

$$y_j = \begin{cases} 1 & \text{if} \quad y\_in_j > 0 \\ 0 & \text{if} \quad y\_in_j \le 0 \end{cases}$$

- Weights are computed by Hebbian rule (sum of outer products of all training pairs)

$$W = \sum_{p=1}^{P} s_i^T(p) t_j(p)$$

- Training samples:

	s(p)	t(p)
p=1	(1 0 0 0)	(1, 0)
p=2	(1 1 0 0)	(1, 0)
p=3	(0 0 0 1)	(0, 1)
p=4	(0 0 1 1)	(0, 1)

$$s^T(1)\cdot t(1) = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 0\\0 & 0\\0 & 0\end{pmatrix} \qquad s^T(2)\cdot t(2) = \begin{pmatrix}1\\1\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\1 & 0\\0 & 0\\0 & 0\end{pmatrix}$$

$$s^T(3)\cdot t(3) = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 0\\0 & 0\\0 & 1\end{pmatrix} \qquad s^T(4)\cdot t(4) = \begin{pmatrix}0\\0\\1\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 0\\0 & 1\\0 & 1\end{pmatrix}$$

Computing the weights $W = \sum_{p=1}^{P} s_i^T(p) t_j(p)$ $\quad W = \begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}$

x=(1 0 0 0)

$$(1\ \ 0\ \ 0\ \ 0)\begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}=(2\ \ 0)$$

$y_1=1,\quad y_2=0$

x=(0 1 0 0) similar to S(1) and S(2)

$$(0\ \ 1\ \ 0\ \ 0)\begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}=(1\ \ 0)$$

$y_1=1,\quad y_2=0$

x=(0 1 1 0)

$$(0\ \ 1\ \ 1\ \ 0)\begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}=(1\ \ 1)$$

$y_1=1,\quad y_2=1$

(1 0 0 0), (1 1 0 0) class (1, 0)
(0 0 0 1), (0 0 1 1) class (0, 1)

(0 1 1 0) is not sufficiently similar to any class

因此，该算法重要的是根据所有 training examples 计算出 W，再根据 W 分类。

Example 2: auto-associative

该方法和 hetero-associative nets 类似，除了 t(p) =s(p) (下面可以看到)。

用于通过其嘈杂或不完整的版本 (noisy or incomplete version) 来 recall 模式。(模式完成/模式恢复，pattern completion/pattern recovery)

- A single pattern ***s*** = (1, 1, 1, -1) is stored (weights computed by Hebbian rule – outer product)

$$\boldsymbol{W}=\begin{bmatrix}1 & 1 & 1 & -1\\1 & 1 & 1 & -1\\1 & 1 & 1 & -1\\-1 & -1 & -1 & 1\end{bmatrix}$$

training pat.	$(1\,1\,1\,-1)\cdot\boldsymbol{W}=(4\,4\,4\,-4)\rightarrow(1\,1\,1\,-1)$
noisy pat	$(-1\,1\,1\,-1)\cdot\boldsymbol{W}=(2\,2\,2\,-2)\rightarrow(1\,1\,1\,-1)$
missing info	$(0\,0\,1\,-1)\cdot\boldsymbol{W}=(2\,2\,2\,-2)\rightarrow(1\,1\,1\,-1)$
more noisy	$(-1\,-1\,1\,-1)\cdot\boldsymbol{W}=(0\,0\,0\,0)$ not recognized

- W 始终是对称矩阵
- 当存储多个模式时，对角线元素将主导计算（= P）
- 当 P 很大时 (模式很多)，W 接近单位矩阵。这会导致 input = output，这可能无法储存任何模式。模式校正能力丢失

- 将对角线元素替换为 0：

$$W'=\begin{bmatrix}0 & 1 & 1 & -1\\ 1 & 0 & 1 & -1\\ 1 & 1 & 0 & -1\\ -1 & -1 & -1 & 0\end{bmatrix}\qquad \begin{array}{r}(1\ \ 1\ \ 1\ \ -1)W'=(3\ \ 3\ \ 3\ \ -3)\rightarrow(1\ \ 1\ \ 1\ \ -1)\\ (-1\ \ 1\ \ 1\ \ -1)W'=(3\ \ 1\ \ 1\ \ -1)\rightarrow(1\ \ 1\ \ 1\ \ -1)\\ (0\ \ 0\ \ 1\ \ -1)W'=(2\ \ 2\ \ 1\ \ -1)\rightarrow(1\ \ 1\ \ 1\ \ -1)\\ (-1\ \ -1\ \ 1\ \ -1)W'=(1\ \ 1\ \ -1\ \ 1)\rightarrow\ \ wrong\end{array}$$