上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

对传网(CPN)

- **主要内容：CPN的网络结构，正常运行，输入向量的预处理，Kohonen层的训练算法及其权矩阵的初始化方法；Grossberg层的训练；完整的对传网**
- **重点：Kohonen层与Grossberg层的正常运行与训练**
- **难点：Kohonen层的训练算法及其权矩阵的初始化方法**

对传网

1 网络结构

2 网络的正常运行

3 Kohonen层的训练

4 Kohonen层联接权的初始化方法

5 Grossberg层的训练

6 补充说明

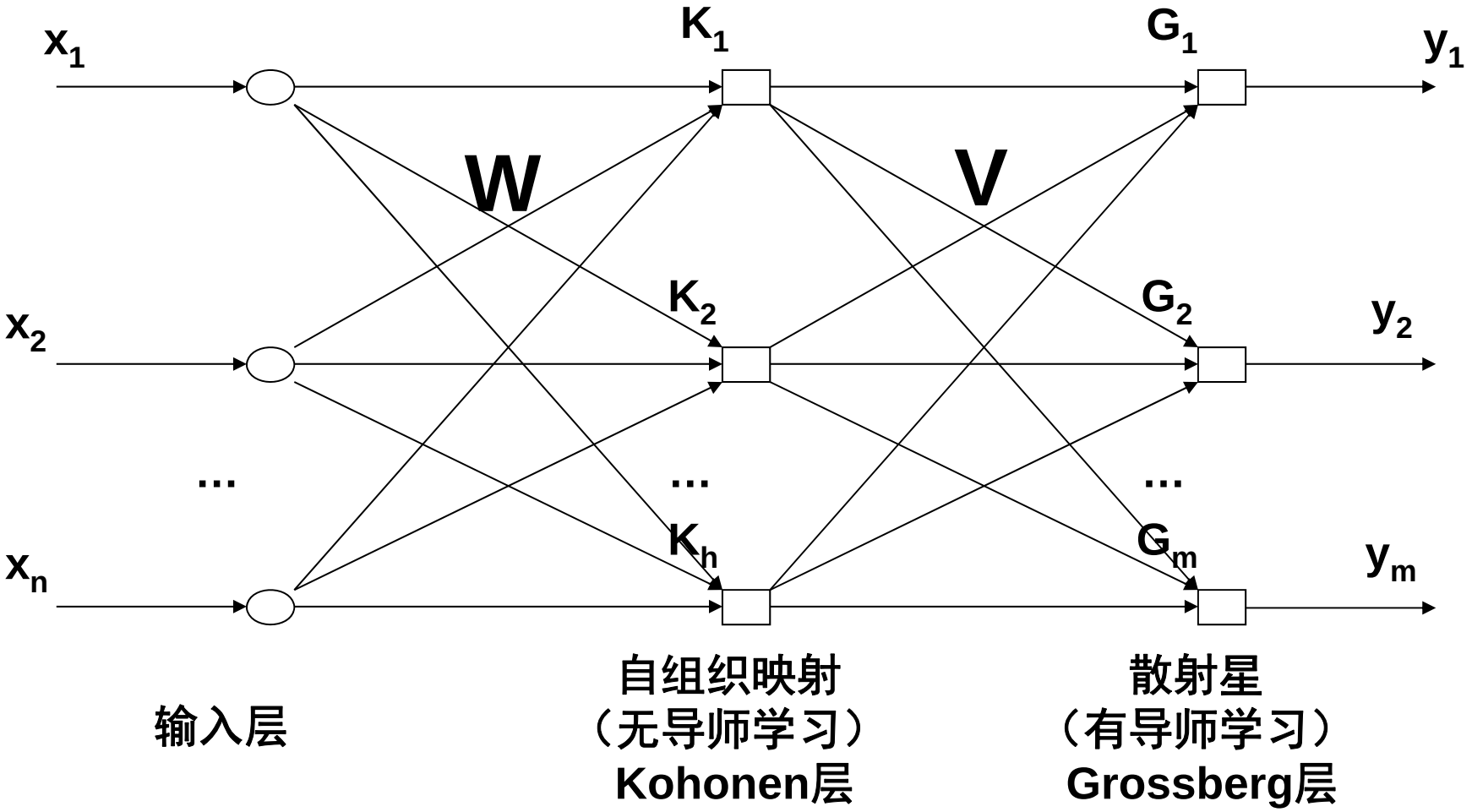
对传网

- Robert Hecht-Nielson 在1987年提出了对传网（Counterpropagation Networks，CPN）。
- CPN为异构网：
  - Kohonen1981年提出的Self-organization map
    - SOM——Kohonen层
  - Grossberg1969年提出的Outstar——Grossberg层
- 训练时间短：BP的1%。应用面：比较窄
- 让网络的隐藏层执行无导师学习，是解决多级网络训练的另一个思路

1 网络结构

- 单向CPN，完整CPN（双向网）
- 除拓扑结构外，网络的运行机制也是确定网络结构和性能的重要因素
- ***网络的层数计算***

1 网络结构


x1
x2
...
xn
K1
K2
...
Kh
G1
G2
...
Gm
y1
y2
ym
W
V
输入层
自组织映射
（无导师学习）
Kohonen层
散射星
（有导师学习）
Grossberg层

1 网络结构

- 以Kohonen层的神经元为“中心”讨论问题
- K_1
  - $W_1=(w_{11}, w_{21}, \ldots, w_{n1})^T$
  - $V_1=(v_{11}, v_{12}, \ldots, v_{1m})$
- K_2
  - $W_2=(w_{12}, w_{22}, \ldots, w_{n2})^T$
  - $V_2=(v_{21}, v_{22}, \ldots, v_{2m})$

  ……
- K_h
  - $W_h=(w_{1h}, w_{2h}, \ldots, w_{nh})^T$
  - $V_h=(v_{h1}, v_{h2}, \ldots, v_{hm})$

2 网络的正常运行

2.1 Kohonen层

- “强者占先、弱者退出”（the winner takes all）

$knet_j = XW_j$

$= (x_1, x_2, \ldots, x_n)(w_{1j}, w_{2j}, \ldots, w_{nj})^T$

$= w_{1j}x_1 + w_{2j}x_2 + \ldots + w_{nj}x_n$

向量形式

$KNET = (knet_1, knet_2, \ldots, knet_h)$

2.1 Kohonen层

- K_1，K_2，…，K_h的输出k_1，k_2，…，k_h构成向量 $K=(k_1, k_2, \ldots, k_h)$
- $1 \leqq j \leqq h$

$$k_j = \begin{cases} 1 & knet_j = Max\{knet_1, knet_2, \ldots, knet_h\} \\ 0 & \text{其它} \end{cases}$$

- 几何意义

2.2 Grossberg层

- Grossberg层的每个神经元G_j（$1 \leqq j \leqq m$）

$$gnet_j = K(v_{1j}, v_{2j}, \ldots, v_{hj})^T$$

$$= (k_1, k_2, \ldots, k_h)(v_{1j}, v_{2j}, \ldots, v_{hj})^T$$

$$= k_1 v_{1j} + k_2 v_{2j} + \ldots + k_h v_{hj}$$

唯一输出1的神经元为K_o

$$gnet_j = k_1 v_{1j} + k_2 v_{2j} + \ldots + k_h v_{hj}$$

$$= v_{oj}$$

2.2 Grossberg层

$$GNET=(gnet_1, gnet_2, \ldots, gnet_m)$$
$$=(v_{o1}, v_{o2}, \ldots, v_{om})$$
$$=V_o$$

- **散射星**：V_o的各个分量是从K_o到Grossberg层各神经元的联接权

2.2 Grossberg层

- **CPN用于模式的完善**，此时n=m：接受含有噪音的输入模式(x_1，x_2，…，x_n)，而输出去掉噪音后的模式(v_{o1}，v_{o2}，…，v_{om})
- **对训练启示**
  - W_1，W_2，…，W_h，各类X的共同特征
  - V_1，V_2，…，V_h，X对应的理想输出Y的共同特征

3 Kohonen层的训练

3.1 输入向量的预处理

单位化处理

$$X=(x_1,\ x_2,\ \ldots,\ x_n)$$

$$X'=(x_1',\ x_2',\ \ldots,\ x_n')$$

$$=(x_1/\|X\|,\ x_2/\|X\|,\ \ldots,\ x_n/\|X\|)$$

3.2 训练

算法1 Kohonen层训练算法

1 对所有的输入向量，进行单位化处理；

2 对每个样本（X，Y）执行下列过程

2.1 for j=1 to h do 根据相应式子计算$knet_j$；

2.2 求出最大的$knet_o$:

2.2.1 max=$knet_1$； o=1

2.2.2 for j=1 to h do

if $knet_j$>max then {max=$knet_j$； o=j}；

算法 1 Kohonen层训练算法

3 计算K

3.1 for j=1 to h do k_j=0；

3.2 k_o=1；

4 使W_o更接近X：$W_o^{(new)}=W_o^{(old)}+\alpha(X- W_o^{(old)})$;

5 对$W_o^{(new)}$进行单位化处理

$W_o^{(new)}=W_o^{(old)}+\alpha\ (X-\ W_o^{(old)})$

$\alpha \in (0,\ 1)$

$$W_o^{(new)}=W_o^{(old)}+\alpha\ (X-\ W_o^{(old)})$$

$$=W_o^{(old)}+\alpha\ X-\alpha\ W_o^{(old)}$$

$$X-W_o^{(new)}=X-[W_o^{(old)}+\alpha\ (X-\ W_o^{(old)})]$$

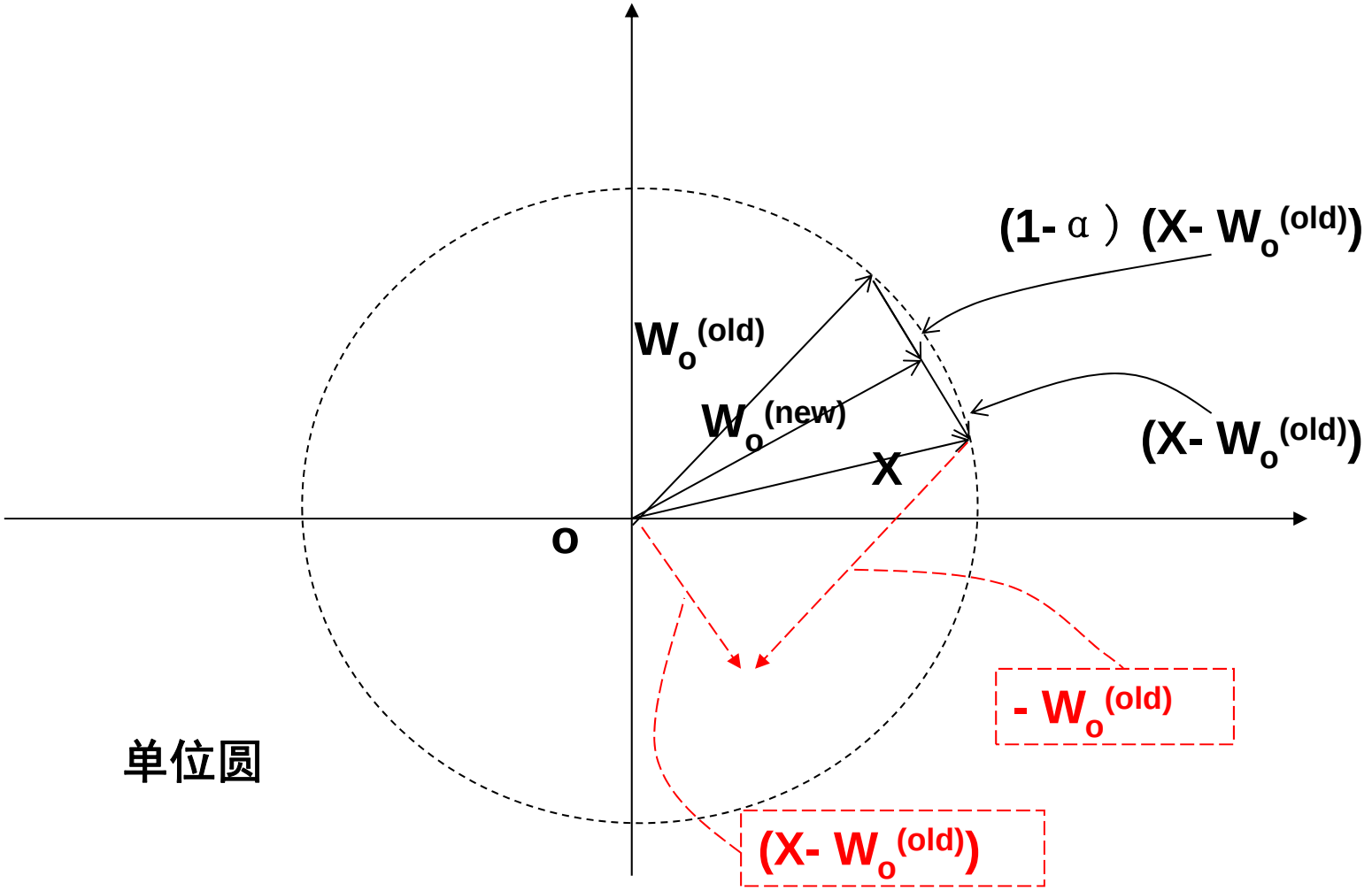
$$=X-W_o^{(old)}-\alpha\ X+\alpha\ W_o^{(old)}$$

$$=X(1-\alpha\ )-W_o^{(old)}(1-\alpha\ )$$

$$=(1-\alpha\ )\ (X-W_o^{(old)})$$

由$0<(1-\alpha\ )<1$，$W_o^{(new)}$比$W_o^{(old)}$更接近X

$$W_o^{(new)} = W_o^{(old)} + \alpha\,(X - W_o^{(old)})$$



学习率α

- 训练初期，α一般取0.7左右，它将随着训练进展不断变小
- α过大可能导致有的**X**被放入错误的类中；使训练陷入抖动
- 根据**X**的分布决定**W**的初值：防止类过小和过大

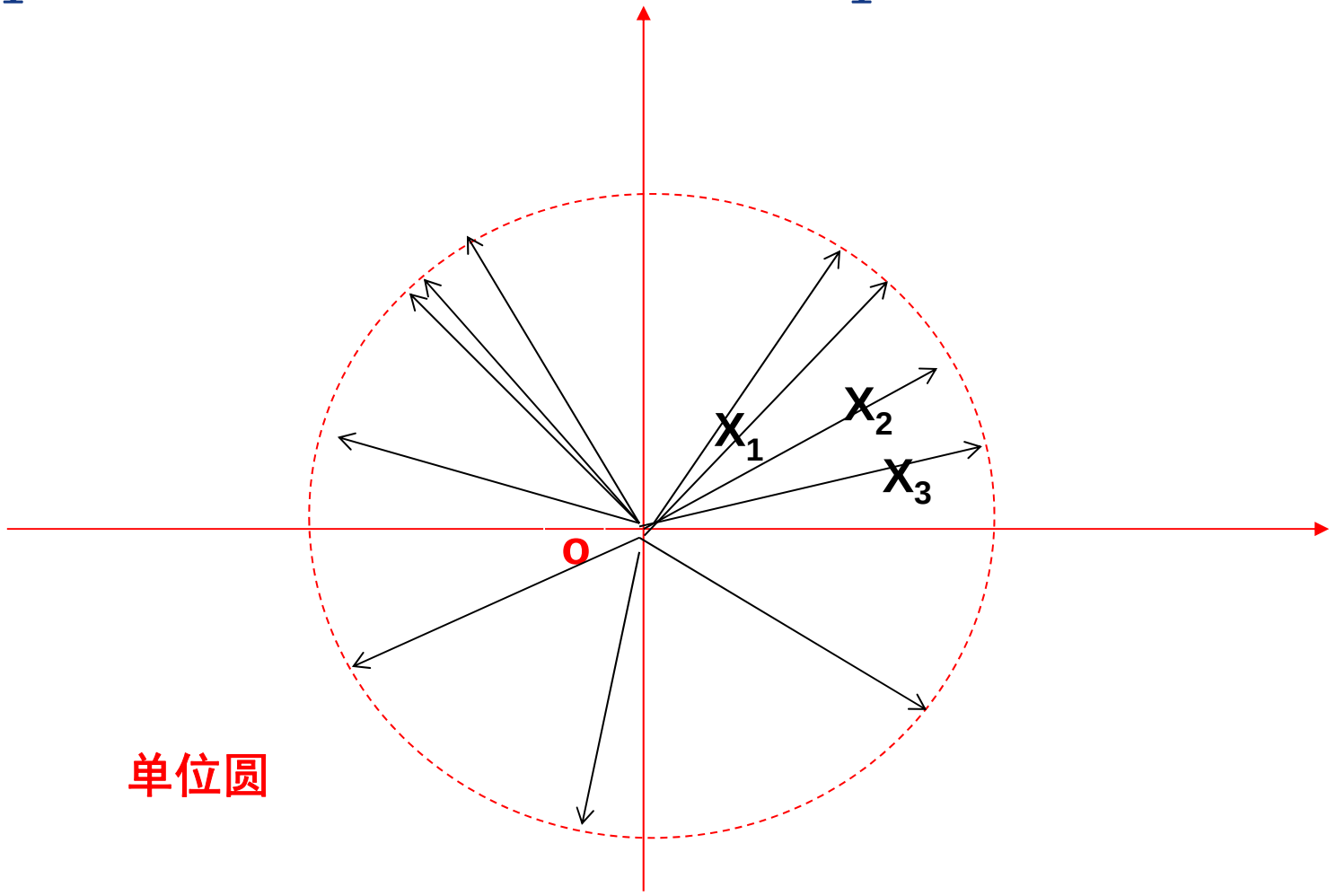
启发

- 一般来说，一个类含有许多向量。这个类对应的$\mathbf{W}_j$应该是样本集中这一类向量（输入向量部分）的平均值。
- 事先给问题一个粗略分类，并从这个分类中提取一个较有代表性的向量构成样本集
- 启发我们采用训练和直接设定权向量的方式来完成该层的训练。

4 Kohonen层联接权初始化

- 理想情况下，W_1，W_2，…，W_h的初值应该依照样本集中的输入向量的分布来确定
- 样本集中的输入向量的分布并不是均匀的

X_i的非均匀分布要求W_i非均匀分布



凸状组合法

取$w_{ij}=\frac{1}{\mathrm{sqrt}(n)}$

将输入向量

$X=(x_1, x_2, \ldots, x_n)$

变换为

$X'=(x_1', x_2', \ldots, x_n')$

其中

$$x_j' = \lambda x_j + \frac{1-\lambda}{\sqrt{n}}$$

凸状组合法

在训练的初期阶段，λ的值非常小，使得

$$X \approx (\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \cdots, \frac{1}{\sqrt{n}})$$

随着训练的进行，λ趋近于1，从而使$\mathbf{X}'$趋近于$\mathbf{X}$，进而$\mathbf{W}_j$趋近于一组$\mathbf{X}$的平均值。

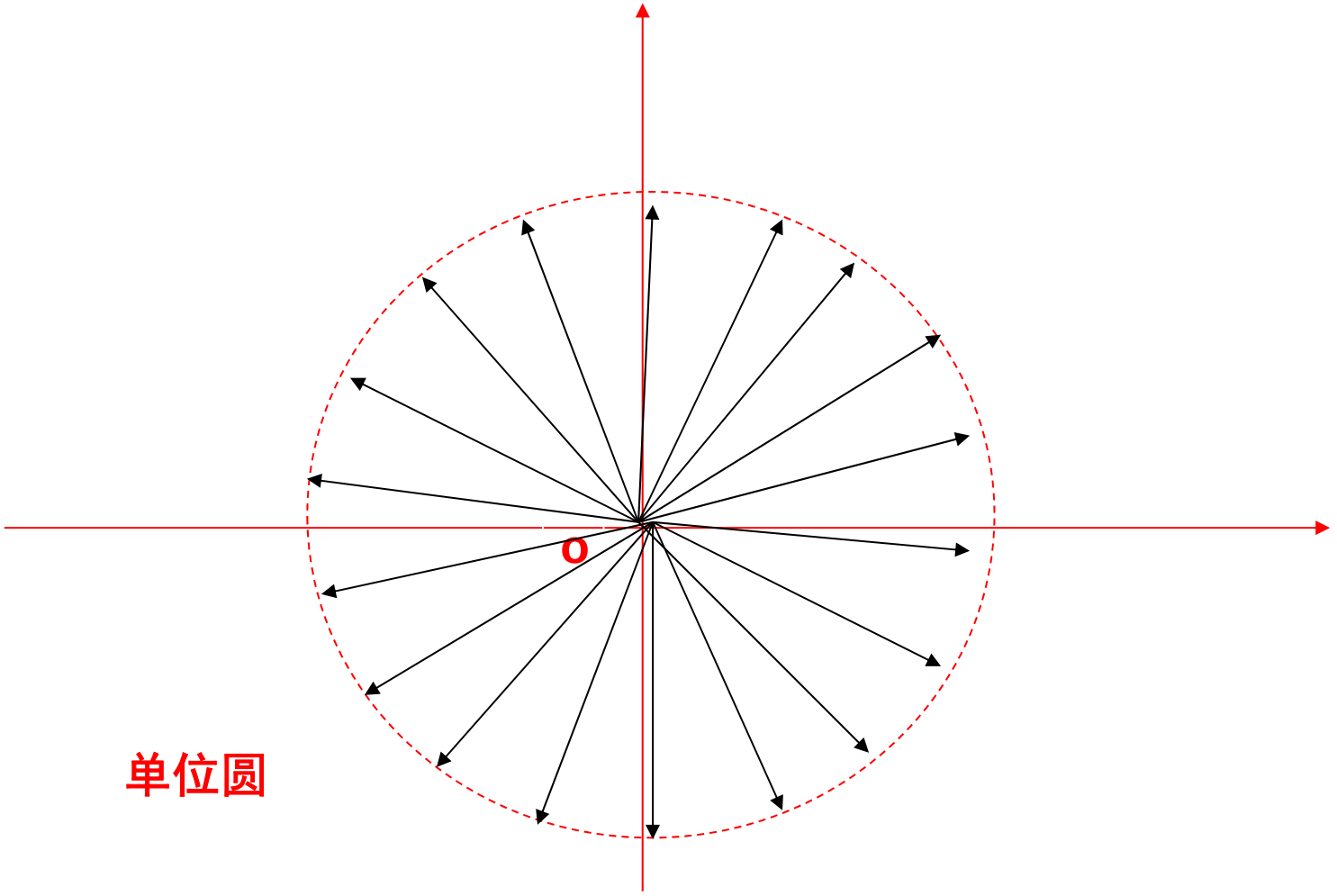
W需要追踪一个变化的目标

添加噪音法

- 在输入向量中加进适当的随机噪音，使输入向量的分布均匀。训练中逐渐去掉噪音
- W_j不断地调整自己的“运动方向”，去追踪其不断变化的目标。试验表明，这种方法的收敛速度比凸状组合法更慢。

W也需要追踪一个变化的目标

X在加噪音后变成均匀分布的



单位圆

初期全调法

- **Kohonen**层训练的初期，对应一个输入向量，允许多个神经元同时处于激发状态。逐渐减少被激发的神经元的最大个数或者逐渐提高阈值，最后达到对一个输入向量，只有一个神经元激发
- **要解决的问题**
  - 问题调整的范围的度量。

初期全调法

另一种实现

- 在训练的初期，算法不仅调整“获胜”的神经元对应的权向量，而且对其它的权向量也作适当的调整。随着训练的推进，被调整的范围逐渐缩小，直到最终只有“获胜”的神经元对应的权向量才被调整

要解决的问题

- 问题调整的范围的度量。
- 其它的权向量的“适当调整”

DeSieno法

- 当某一个权向量所获得的匹配向量超过给定的数（1/h）后，它的阈值就被临时提高
- **问题**：当最应该被某个神经元对应的权向量匹配的输入向量在较后的时候被输入时，它可能被拒绝，从而造成网络精度的损失
- **Kohonen [1988]**：在一个被完全训练过的网中，随机选取的输入向量与任何给定权向量是最接近的概率是1/h
  - 按均匀分布初始化的权向量具有相同被匹配概率

5 Grossberg层的训练

- 训练
  - 标量形式

$$v_{oj} = v_{oj} + \alpha\ (y_j - v_{oj})$$

  - 向量形式

$$V_o^{(new)} = V_o^{(old)} + \alpha\ (Y - V_o^{(old)})$$

- 比较

$$W_o^{(new)} = W_o^{(old)} + \alpha\ (X - W_o^{(old)})$$

Kohonen层

算法2 CPN训练算法一

0 对W、V进行初始化；

1 对所有的输入向量，进行单位化处理；

2 对每个样本（X，Y）执行下列过程

2.1 for j=1 to h do 根据$knet_j=XW_j$计算$knet_j$；

2.2 求出最大的$knet_o$：

2.2.1 $max=knet_1$； o=1;

2.2.2 for j=1 to h do

2.2.2.1 if $knet_j>max$ then {$max=knet_j$； o=j}；

算法2 CPN训练算法一

2.3 计算K：

2.3.1 for j=1 to h do k_j=0；

2.3.2 k_o=1；

2.4 使W_o更接近X：

$W_o^{(new)}=W_o^{(old)}+\alpha\ (X-W_o^{(old)})$；

2.5 对$W_o^{(new)}$进行单位化处理；

2.6 使V_o更接近Y：

$V_o^{(new)}=V_o^{(old)}+\alpha\ (Y-V_o^{(old)})$。

算法3 CPN训练算法二

- 对应Kohonen的每一个K_i，它将代表一组输入向量，所以希望这个K_i对应的V_i能代表这组输入向量对应的输出向量的平均值。

0　对W、V进行初始化；

0′ 清空Kohonen层各神经元对应的纪录表：

for j=1 to h do $SK_j=\Phi$；

1 对所有的输入向量，进行单位化处理；

算法3 CPN训练算法二

2 对每个样本（X_s，Y_s）执行下列过程

2.1 for j=1 to h do

2.1.1 根据相应式子计算$knet_j$;

2.2 求出最大的$knet_o$:

2.2.1 max=$knet_1$； o=1;

2.2.2 for j=1 to h do

2.2.2.1 if $knet_j$>max then

{max=$knet_j$； o=j};

算法3 CPN训练算法二

2.3 计算K：

2.3.1 for j=1 to h do k_j=0；

2.3.2 k_o=1；

2.4 使W_o更接近X_s：

$W_o^{(new)}=W_o^{(old)}+\alpha\ (X_s- W_o^{(old)})$；

2.5 对$W_o^{(new)}$进行单位化处理；

2.6 将Y_s放入SK_o：

$SK_o=SK_o \cup \{Y_s\}$；

3 for j=1 to h do

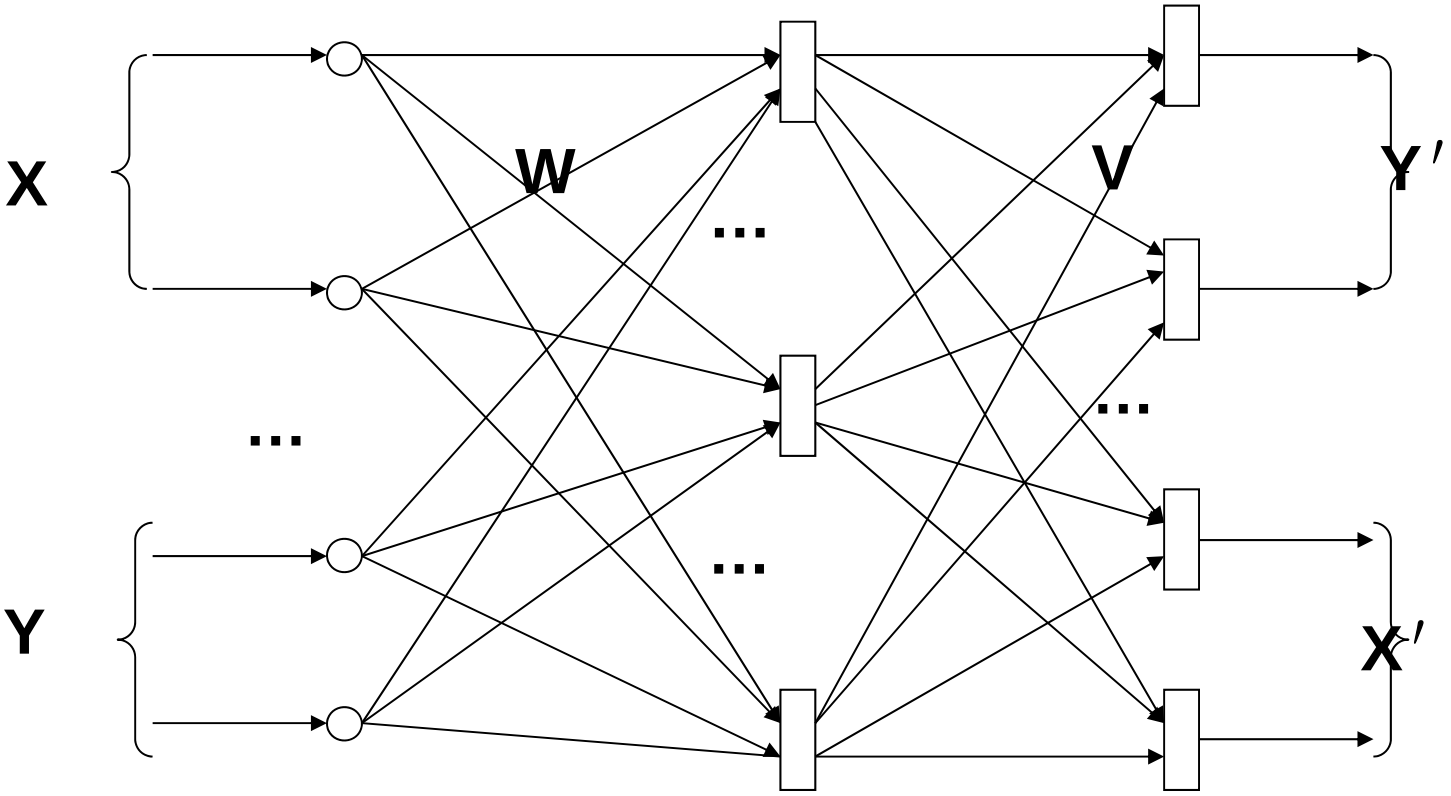
V_j= SK_j中各向量的平均值

算法的进一步优化

- 集合变量SK_1， SK_2 ，…，SK_h改为其它存储量更小，而且更容易实现的变量
- 在X_s激发K_o时，Y_s被放入到SK_o中
  - 会不会出现一个向量被放入多个SK中的问题
  - 如何解决

6 补充说明

1、全对传网



2、非简单工作方式

- 对给定的输入向量，**Kohonen**层各神经元可以给出不同的输出
- 输出作为修改因子
  - 对应神经元**Kohonen**层、**Grossberg**层的权向量
  - 输出值较大的，表明该输入向量与该神经元对应的类较接近，它对应的权向量的修改量就大
  - 输出值较小的，表明该输入向量与该神经元对应的类较远，它对应的权向量的修改量就小。