脑启发人工智能导论 Introduction to Brain-Inspired Artificial Intelligence

唐华锦 教授
浙江大学计算机学院
htang@zju.edu.cn
https://person.zju.edu.cn/htang

大作业

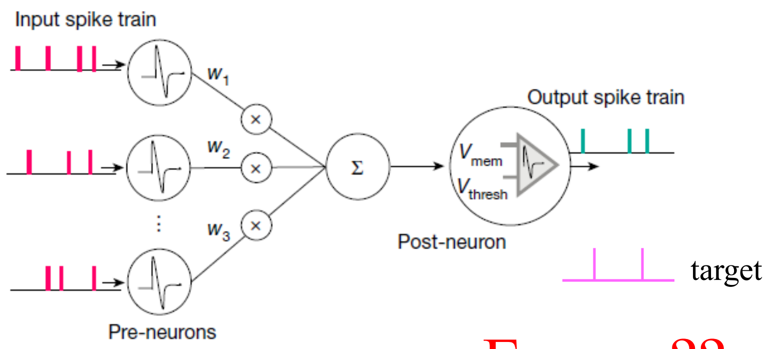
研读一篇或多篇脑启发人工智能（计算神经科学、脑科学、类脑计算、类脑智能、机器人）相关的论文，以**PPT**形式报告，要求包含：

- 阐述其核心思想、主要方法、结果、结论、启发等；
- 结合本课程内容，阐述你对脑启发人工智能（类脑计算）方法、模型、计算架构、系统或机器人等的创新构想。

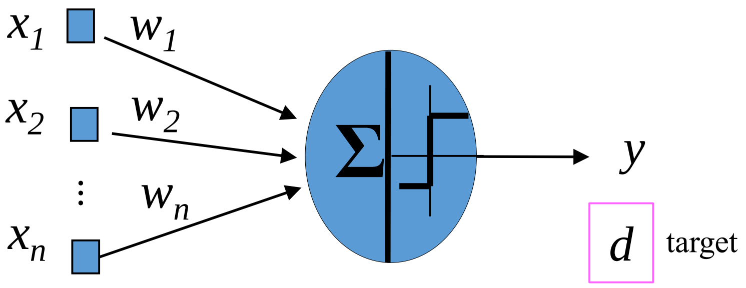
监督式学习
Supervised Learning

Supervised Learning in Spiking Neural Networks

- Encode information as input spikes
- Input spike sequences into spiking neural network to calculate output spikes
- Compare the expected spike sequences with the actual output spike to get the error, and adjust the weight according to the error.



Error= ??



Error=(y-d)^2

脉冲序列的相似性度量（Similarity Measure）

- 在PSD算法中提到了脉冲序列以及其相似性的定义，脉冲序列在脉冲神经网络的监督学习过程中扮演很重要的角色，度量脉冲序列的相似性对于误差计算和梯度推导都有着非常重要的作用。

- 对于脉冲序列的**相似性度量**，目前主要采用的方式分为以下两类：

① 根据时间序列的**脉冲时间间隔**比较两个脉冲序列，其重点在于对**脉冲的发放频率进行局部估计;**

② 根据**脉冲发放的精确时间**比较两个脉冲序列，通过脉冲序列模式的**时间结构特征**进行计算

脉冲序列相似性度量（Similarity Measure）

- 首先对脉冲序列进行数学化定义，这个定义在PSD算法中已经提及了：

$$S(t) = \sum_f \delta(t - t^f)$$

$$\delta(x) = \begin{cases} 1, x = 0 \\ 0, x \neq 0 \end{cases}$$

- 相同的，为了解决狄拉克函数的不连续问题，这里也要对脉冲序列进行卷积：

$$\tilde{S}_i(t) = S_i(t) * K(t)$$

- 这种脉冲序列的表现形式既可以解释为基于突触模型的突触后膜电位，也可以解释为脉冲序列的条件密度函数估计。

脉冲序列内积的定义

- 定义

首先给出脉冲时间的内积，对于两个脉冲对应的发放时间，对于两个脉冲对应的发放时间t^m和t^n,有：

$$\mathrm{K}(t^m，t^n) =< \delta(t - t^m), \delta(t - t^n)>$$

其中$\mathrm{K}(x, y): X * X \to R$是满足对称性和正定性的一个核函数。那么对于脉冲序列s_i, s_j则有

$$F\left(s_i, s_j\right) =< f_{s_i}(t)，f_{s_j}(t)>= \int_{\Gamma} f_{s_i}(t) * f_{s_j}(t)\mathrm{dt}$$

其中 $f(t) = s(t) * h(t) = \sum_{f=1}^{N} h(t - t^f)$表示对脉冲序列采用特定平滑函数处理转化为连续函数的过程（卷积计算）

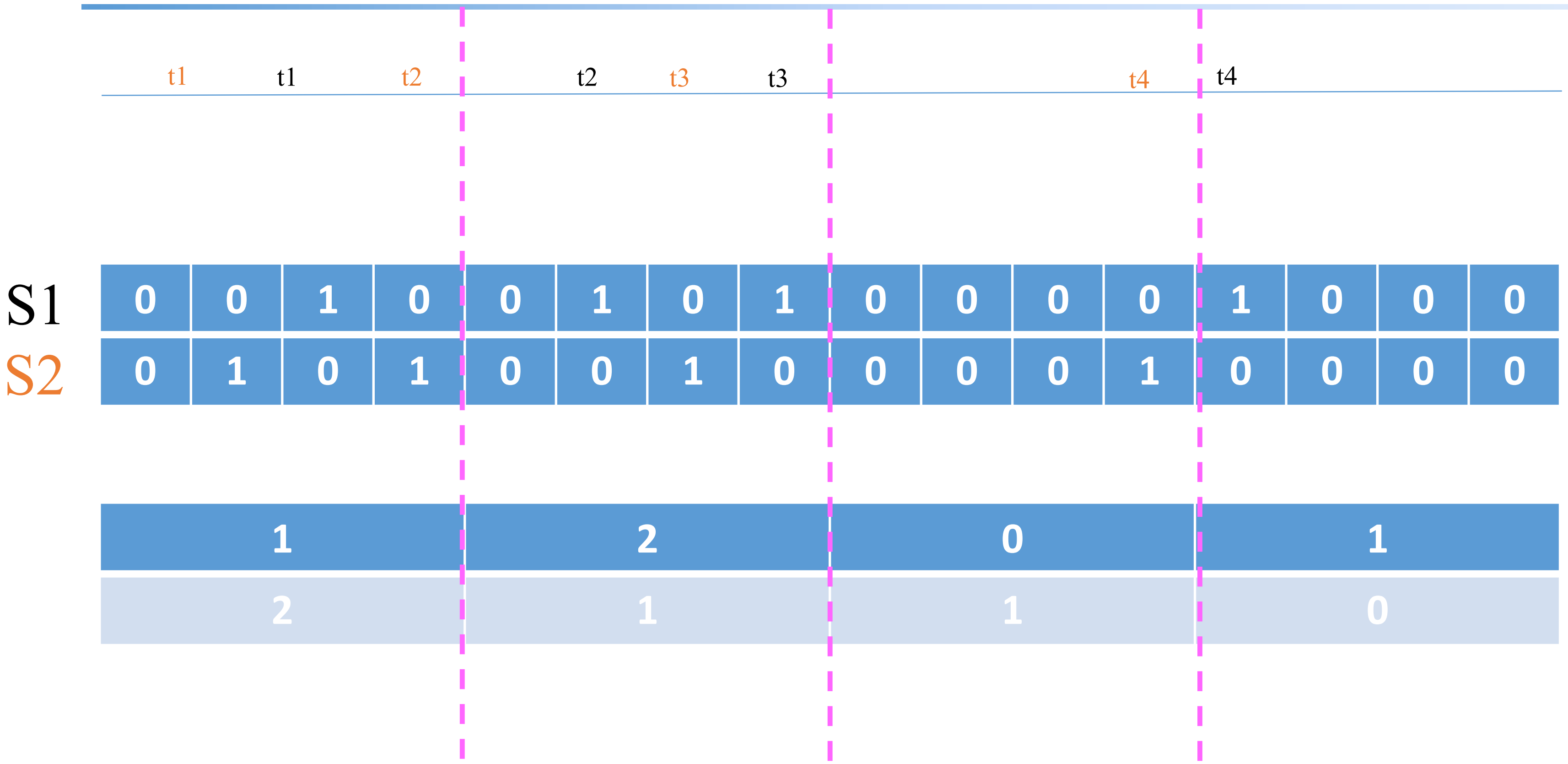
度量方法：基于区域脉冲计数的相似性度量方法

- 对于两个脉冲序列的相似性构造，可以使用一种简单有效的脉冲序列和方法进行构造，将脉冲映射到一个维度有限且固定的欧式空间，然后计算脉冲序列的内积：

$$\mathrm{K}(x,y)=(x-y)^T(x-y),x,y\in R^d$$

- 选择一个连续时间窗口将脉冲序列进行分区，相邻的时间分区对应着不同的维度，则可以将脉冲序列以一定维度的向量形式存储。

度量方法：基于区域脉冲计数的相似性度量方法



S1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0
S2	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0

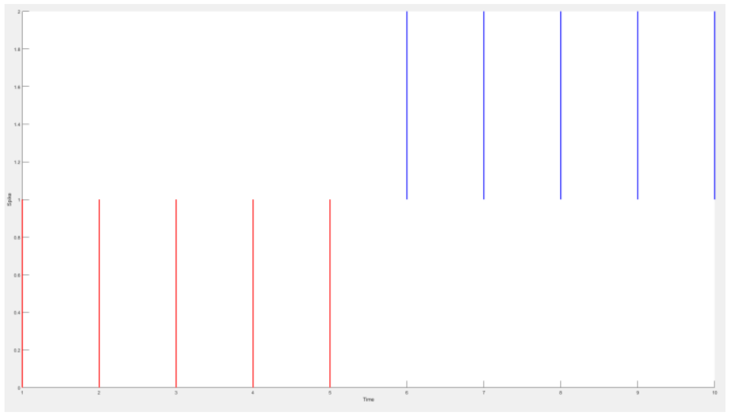
1	2	0	1
2	1	1	0

如上图中，两个不同脉冲序列，下面则是它们在不同时间窗口下转换的向量。

区域脉冲计数仿真代码&结果

平滑区域参数设置为2，时间窗总长度为10

```
function [out]=timeSplit(s1,s2,inner_t,tmax)
%tmax表示最大脉冲发放时间, inner_t表示时间分区长度%
s_t1=zeros(1,fix(tmax/inner_t));
s_t2=zeros(1,fix(tmax/inner_t));
%计算S1时间分区平滑后的特征向量%
for i=1:length(s1)
idx=ceil(s1(i)/inner_t);
s_t1(idx)=s_t1(idx)+1;
end
%计算S2时间分区平滑后的特征向量%
for i=1:length(s2)
idx=ceil(s2(i)/inner_t);
s_t2(idx)=s_t2(idx)+1;
end
%计算S1和S2的特征向量距离%
out=pdist2(s_t1,s_t2);
```



S1,S2完全相同：

```
s1 =

     1     2     3     4     5


s2 =

     1     2     3     4     5


ans =

     0
```

脉冲计数向量

$\Rightarrow$ [2,2,1,0,0]

S1,S2完全不同

```
s1 =

     1     2     3     4     5


s2 =

     6     7     8     9    10


ans =

     4
```

脉冲计数向量

$\Rightarrow$ [2,2,1,0,0]

$\Rightarrow$ [0,0,1,2,2]

度量方法：基于区域脉冲计数的相似性度量方法

- 优点：这是一种非常简单的相似性度量方法，

- 局限：由于很大程度上忽略了脉冲序列的时间结构特性，因而不能对脉冲序列的相似性度量进行准确计算。相比于后续会介绍的VR距离和VP距离，该方法只能简单对脉冲序列的相似性进行大致估算。

度量方法：Victor-Purpura相似性

- Victor-Purpura相似性度量方法是由Victor和Purpura提出的一种基于脉冲时间匹配的脉冲序列相似性度量方法，通常称为VP距离。
- VP距离的核心思想为：计算将一个脉冲变换成另一个脉冲所需的最小代价，与编辑距离（Minimum Edit Distance，MED）十分相似，计算过程有以下三个步骤：
  - 插入一个脉冲，将代价设置为1
  - 删除一个脉冲，将代价设置为1
  - 平移一个脉冲，假设平移时间为Δt，则代价为$\sigma\left(\frac{|\Delta t|}{\tau_q}\right)$
- 其中平移代价函数σ是一个正的递增函数，并且$\sigma(0) = 0$；参数τ_q表示正的时间常量。一般情况下，$\sigma(x) = x$
- VP距离可以定义为：

$$D_{vp}\left(s_i, s_j\right) = \sum_{t_i^m \in s_i^*} 1 + \sum_{t_j^m \in s_j^*} 1 + \sum_{t_i^m \in s - s_i^*, t_j^n \in s - s_j^*} \sigma\left(\frac{\left|t_i^m - t_j^n\right|}{\tau}\right)$$

度量方法：Victor-Purpura相似性

- Victor-Purpura相似性的计算可以归结为三部分求和：插入脉冲的次数、删除脉冲的次数和平移脉冲的代价。

- 由于VP距离是代价最小化的求解过程，因此可以应用动态规划算法对脉冲序列之间的VP距离进行递推计算（可类比编辑距离）。假设$D_{m,n}$表示S_i中含有前m个脉冲的子序列S_i^m与S_j中含有前n个脉冲子序列S_j^n的VP距离，则$D_{m,n}$的递推计算公式为：

$$D_{m,n} = \min\{D_{m-1,n} + 1, D_{m,n-1} + 1, D_{m-1,n-1} + \sigma(\frac{|t_i^m - t_j^n|}{\tau_q})\}$$

- 公式中，VP距离的递推初始值为$D_{m,0} = 0, D_{0,n} = 0$，因为对应的两个子序列S_i^m或S_j^n为空集。
- 计算出VP距离D_{VP}后，再进行归一化，即可得到脉冲序列的相似度公式：

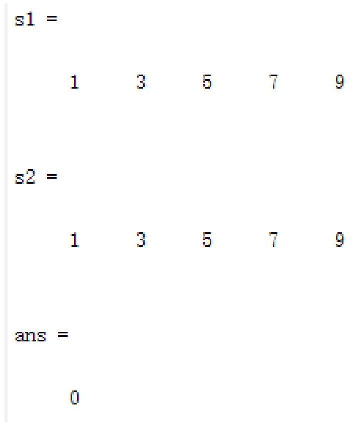
$$D_{NVP}(S_i, S_j) = \frac{N_i + N_j - D_{VP}(S_i, S_j)}{N_i + N_j}$$

N_i， N_j分别是两个脉冲序列包含的脉冲个数。

VP相似性仿真代码&结果：

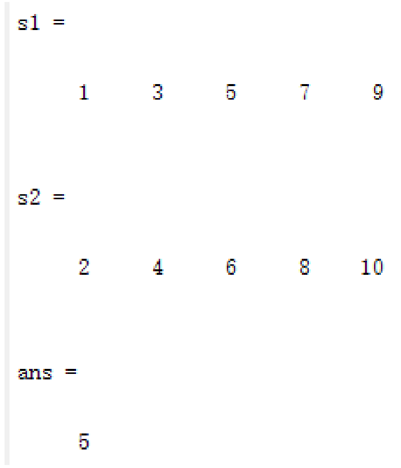
```
function  [out]=VP(s1,s2,tau)
len1=length(s1);
len2=length(s2);
D=zeros(len1,len2);
%初始化递推数组%
for m=1:len1
    for n=1:len2
        D(m,n)=1000000;
    end
End
for m=1:len1
    for n=1:len2
        %边界条件计算%
        if m == 1 && n==1
            D(m,n)=min(n+1,m+1);
            D(m,n)=min(D(m,n),abs(s1(m)-s2(n))/tau);
            continue
        end
        if m==1 && n~=1
            D(m,n)=min(n+1,D(m,n-1)+1);
            D(m,n)=min(D(m,n),n-1+abs(s1(m)-s2(n))/tau);
            continue
        end
        if m~=1 && n==1
            D(m,n)=min(D(m-1,n)+1,m+1);
            D(m,n)=min(D(m,n),m-1+abs(s1(m)-s2(n))/tau);
            continue
        end
        %递推计算%
        D(m,n)=min(D(m-1,n)+1,D(m,n-1)+1);
        D(m,n)=min(D(m,n),D(m-1,n-1)+abs(s1(m)-s2(n))/tau);
    end
end
```

S1, S2完全相同：



```
s1 =

     1     3     5     7     9


s2 =

     1     3     5     7     9


ans =

     0
```

S1, S2完全不同：



```
s1 =

     1     3     5     7     9


s2 =

     2     4     6     8    10


ans =

     5
```

度量方法：van Rossum相似性

- 除了VP距离，van Rossum相似性也常被用于度量脉冲序列相似性，相比于VP距离，van Rossum相似性更简单、更直观，它是一种基于**指数滤波**脉冲序列的欧氏距离。

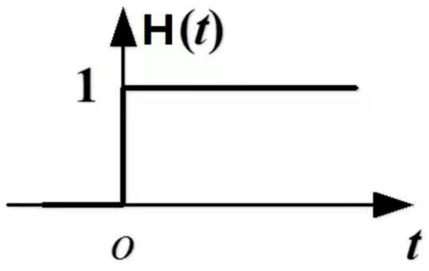
- 首先将脉冲序列s进行指数滤波转换：

$$\tilde{S}(t) = \sum_{f=1}^{N} h(t - t^f)$$

式中，N为脉冲序列包含的脉冲数目。

- h(s)为指数函数，表达式为：

$$h(s) = e^{\left(-\frac{s}{\tau}\right)} H(s)$$



式中，H(s)表示阶跃函数，与PSD算法中的阶跃函数相同，τ表示时间常数，控制指数函数的衰变率。

度量方法：van Rossum相似性

- 根据指数滤波转换后的脉冲序列，可以定义van Rossum距离：

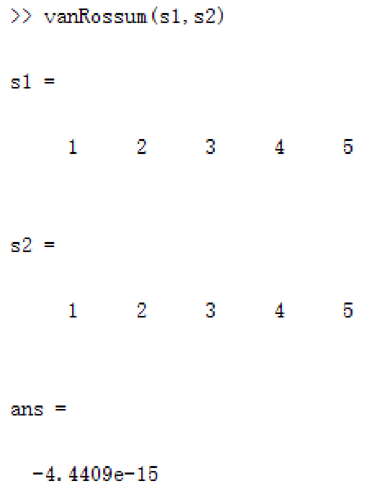
$$D_{VR}(S_i, S_j) = \frac{1}{\tau}\int_0^{\infty} [\tilde{S}_i(t) - \tilde{S}_j(t)]^2 dt$$

- 当两个脉冲序列的脉冲时间非常接近时，指数滤波函数转换的脉冲序列之间的差异非常小，这时两个脉冲序列的VR距离也就非常小。VR距离主要度量两个脉冲序列的脉冲时间差异，比VP距离更加简单更加直观。

VR相似性仿真代码&结果：

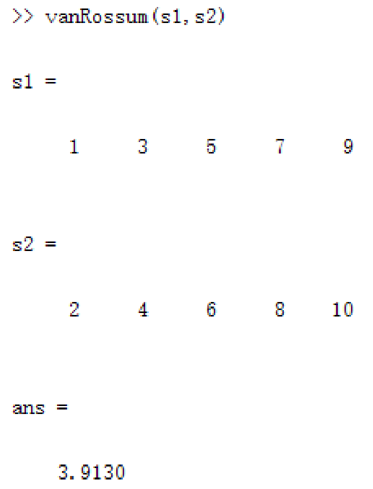
```
function [out]=vanRossum(s1, s2)
ans=0;
tmp1=0;
tmp2=0;
tau=0.5;
%计算S1与S2的脉冲时刻差异%
for i=1:length(s1)
    for j=1:length(s2)
        tmp2=tmp2+lap(s1(i)-s2(j),tau);
    end
end
%计算S1本身的脉冲时刻差异%
for i=1:length(s1)
    for j=1:length(s1)
        tmp1=tmp1+lap(s1(i)-s1(j),tau);
    end
end
%计算S2本身的脉冲时刻差异%
for i=1:length(s2)
    for j=1:length(s2)
        tmp1=tmp1+lap(s2(i)-s2(j),tau);
    end
end
out=tmp1/2-tmp2;
```

S1, S2完全相同：



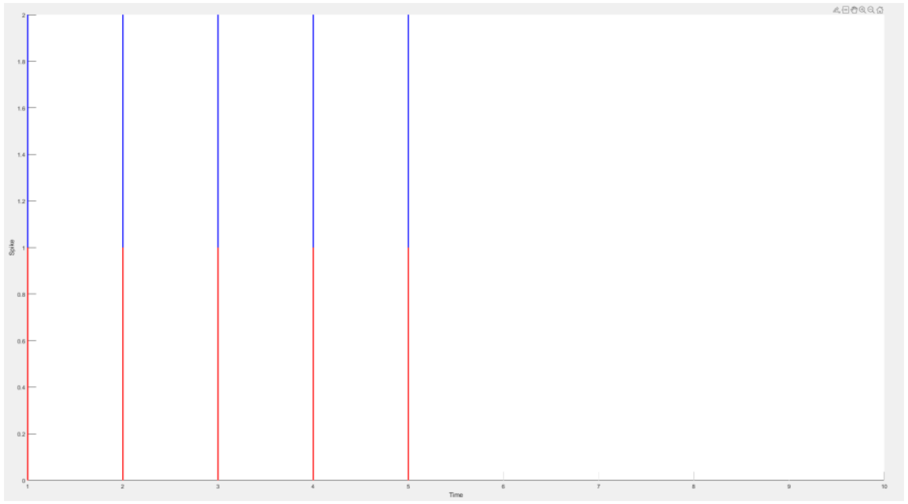
```
>> vanRossum(s1,s2)

s1 =

     1     2     3     4     5


s2 =

     1     2     3     4     5


ans =

  -4.4409e-15
```

S1, S2完全不同：



```
>> vanRossum(s1,s2)

s1 =

     1     3     5     7     9


s2 =

     2     4     6     8    10


ans =

    3.9130
```

VR相似性与VP相似性仿真对比：

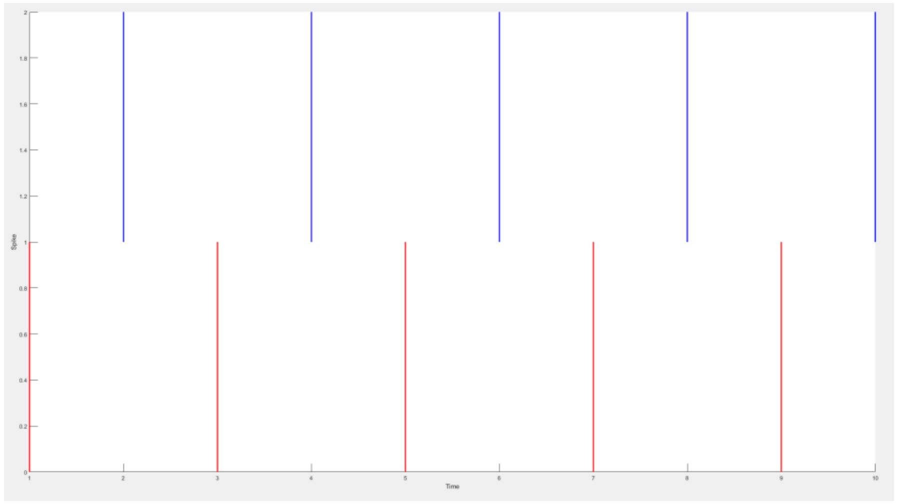


S1=S2=[1, 2, 3, 4, 5],

计算结果：

- VP(S1, S2)=0
- VR(S1, S2)=0

S1用红线表示，S2用蓝线表示



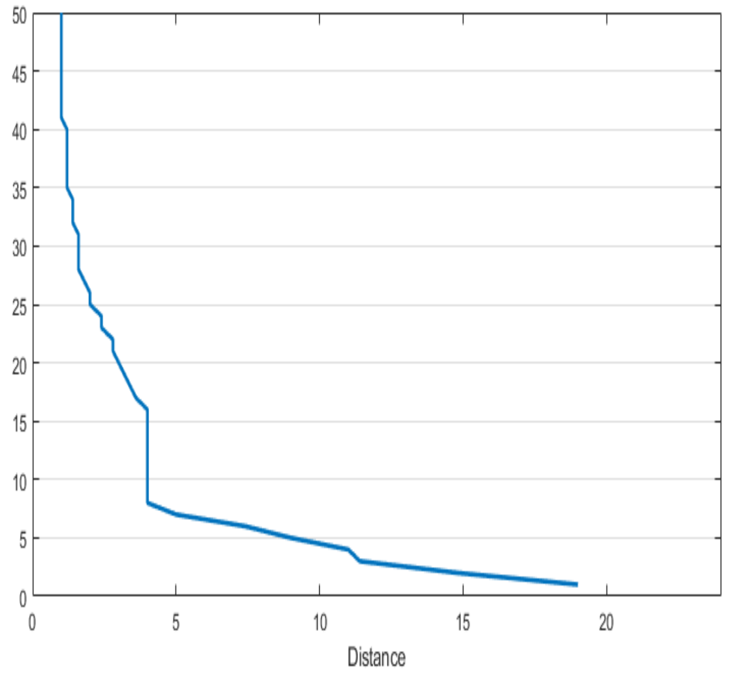
S1=[1, 3, 5, 7, 9],
S2=[2, 4, 6, 8, 10];

计算结果：

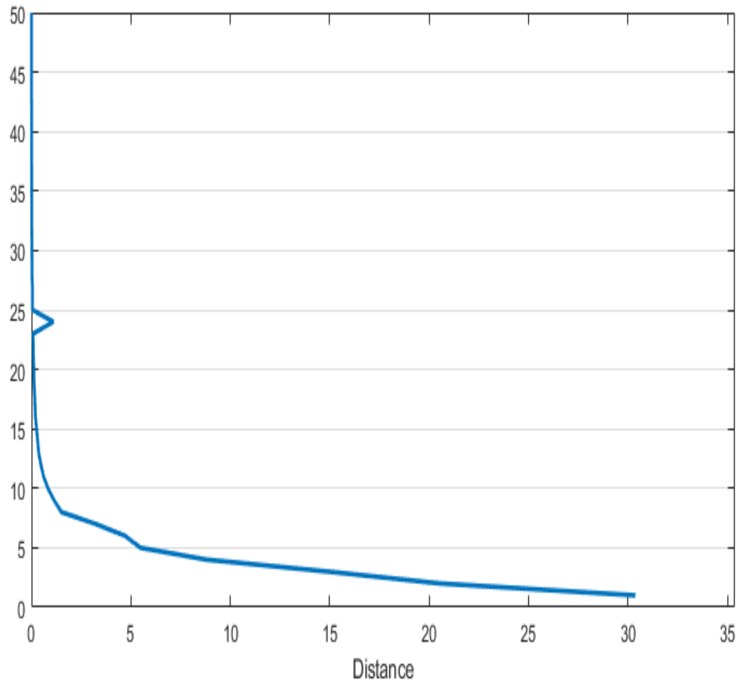
- VP(S1, S2)=5
- VR(S1, S2)=3.913

VP距离与VR距离

- 实例（在psd训练过程中）



VP距离



VR距离

VR相似性与VP相似性仿真对比：

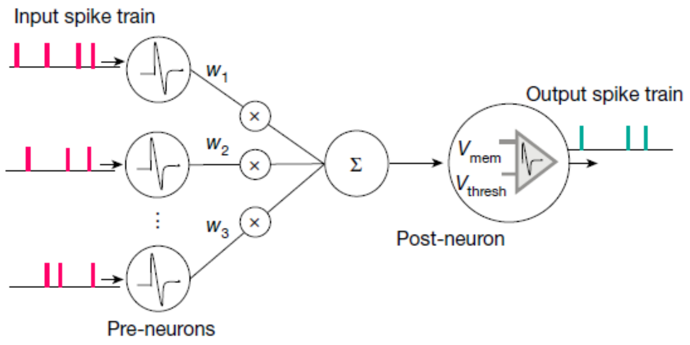
❑ 根据定义与仿真结果，可以总结出VP距离与VR距离主要有两点不同：

1. VP距离定义的是两个脉冲序列之间通过删除、平移、增加三种操作达到一致所需要的最少步数，VP距离的值域为[0,脉冲序列长度]，VP距离可以通过归一化计算转化成相似度定义；

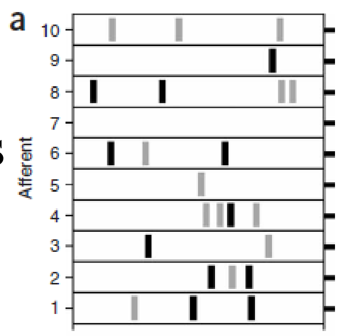
2. VR距离的定义更侧重于计算脉冲序列之间的脉冲发放时刻差异，通过卷积积分进行计算，总体上比VP距离更加直观。

脉冲神经元的监督式学习
Supervised Learning for Spiking Neurons

A typical supervised learning model



Input patterns



输入到输出
的映射

mapping

firing

Category 1

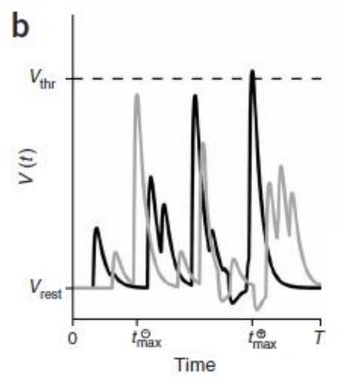
no firing

Category 2

Target pattern

- 根据输入模式是否能激励输出神经元准确发放对应的脉冲模式，调整电压，通过调整权值来实现。

- 调整权值需要合适的训练算法。



Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

Tempotron

Assuming a leaky integrate-and-fire-model the potential $V(t)$ of the synapse can be described by ：

$$V(t) = \sum_i \omega_i \sum_{t_i} K(t - t_i) + V_{rest},$$

where t_i denotes the spike time of the i-th afferent synapse with synaptic weight ω_i and V_{rest} the resting potential.

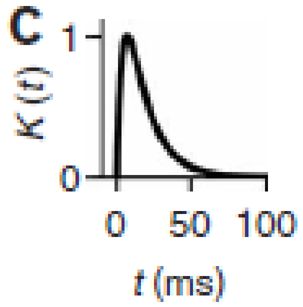
Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

Tempotron

$K(t-t_i)$ describes the postsynaptic potential (PSP) elicited by each incoming spike:

$$K(t-t_i)=\begin{cases} V_0[\exp(-(t-t_i)/\tau)-\exp(-(t-t_i)/\tau_s)] & t \geq t_i \\ 0 & t < t_i \end{cases}$$

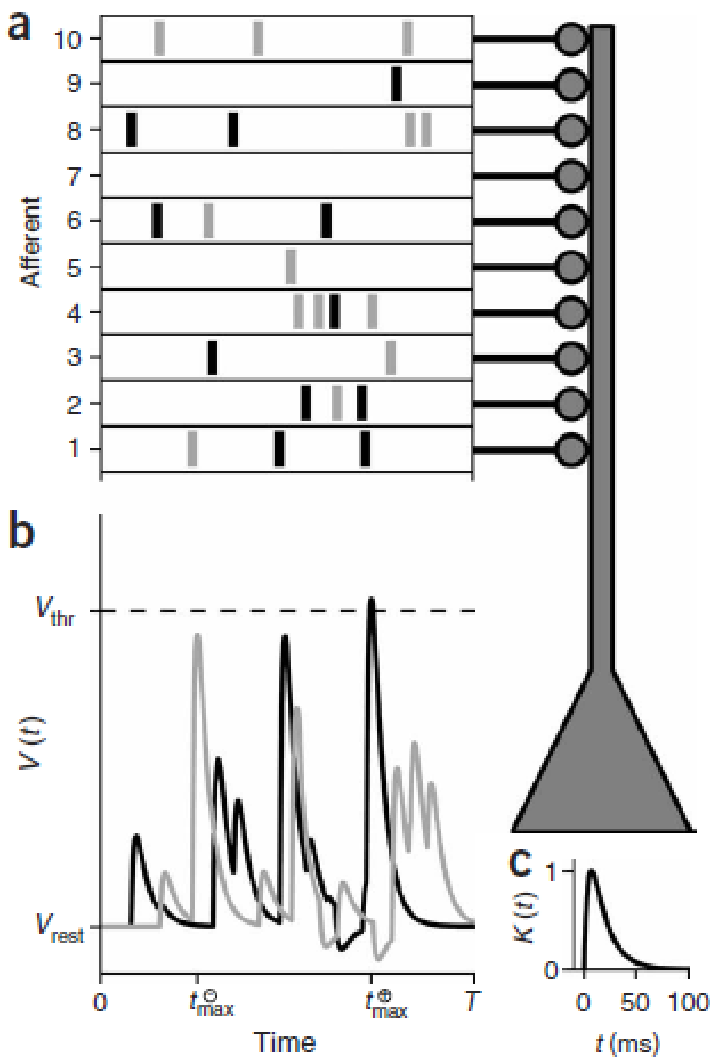
with parameters τ and τ_s denoting decay time constants of the membrane integration and synaptic currents. The factor V_0 is used for the normalization of the PSP kernels. When the potential crosses the firing threshold V_{th} the potential is reset to its resting value by shunting all incoming spikes.



Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

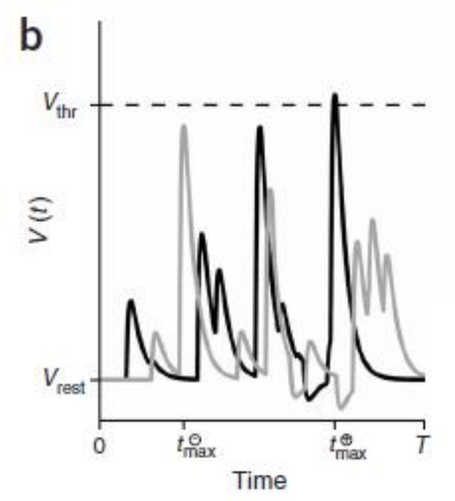
Tempotron Classification result

- Consider a binary classification , we can use one neuron as output neuron， if the output neuron fires, it means 1 else, it means 0.
- As illustrated in right figure.
- (a): Category ⊕ (黑色模式) means belonging to this category, ⊖（灰色模式）means not belonging to this category; (b):Resulting postsynaptic voltage traces V(t). Maximal voltages were reached at $t_{max}^{\oplus}$ and $t_{max}^{\ominus}$ respectively. Because $V(t_{max}^{\oplus}) > V_{thr} > V(t_{max}^{\ominus})$ with V_{thr}(dashed horizontal line) denoting the spike threshold, both patterns are classified correctly.
- Inputs arriving after a threshold crossing (black trace)are shunted.



Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

Tempotron Learning Rule



So, how to learn a binary classification of the input patterns is needed.

① First obtain the maximum membrane potential time points t_{max} of category 1 and category 2 respectively.

② Suppose that when the sample is of category 1. If the output neuron emits spikes, nothing needs to to. If the output neuron does not fire, then find the maximum membrane potential V(t_{max}) and increase the weight to increase V(t_{max}) .

Tempotron Learning Rule

③ If the true category of the sample is category 2, after inputting the neural network, the maximum membrane potential of the output neuron in the current window time range is less than the threshold potential, then no spike is fired and no synaptic weight needs to be updated. When the maximum membrane potential is greater than the threshold potential, the weights need to be weaken.

Based on the above weight updating rule, we can get the Tempotron Learning Rule.

Tempotron Learning Rule

The Tempotron only updates weights when an error occurs based on gradient descent. The loss function is described by:

$$E_{\pm} = \pm\left(V_{\text{thr}} - V\left(t_{\max}\right)\right)\Theta\left(\pm\left(V_{\text{thr}} - V\left(t_{\max}\right)\right)\right)$$

$$\Theta(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

Here t_{max} denotes the time at which the postsynaptic potential $V(t)$ reaches its maximal value.

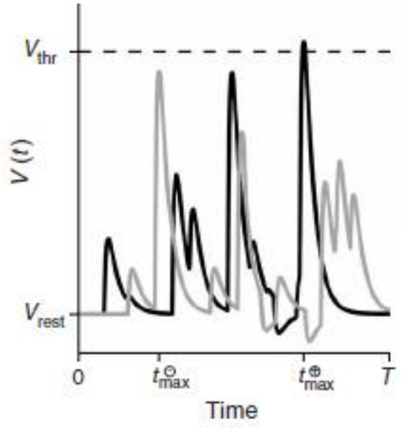
How to get the t_{max} ? The potential V gets the maximal value when its derivative equals to 0:

From

$$\boxed{V(t) = \sum_i \omega_i \sum_{t_i} K(t - t_i) + V_{rest},}$$

we get

$$\boxed{t_{max} = \frac{\tau\tau_s}{\tau - \tau_s}\left(\ln\frac{\tau}{\tau_s} + \ln\frac{\sum \omega_i \exp(\frac{t_i}{\tau})}{\sum \omega_i \exp(\frac{t_i}{\tau_s})}\right)}$$



Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

Tempotron Learning Rule

Loss function ’ derivative of weight is

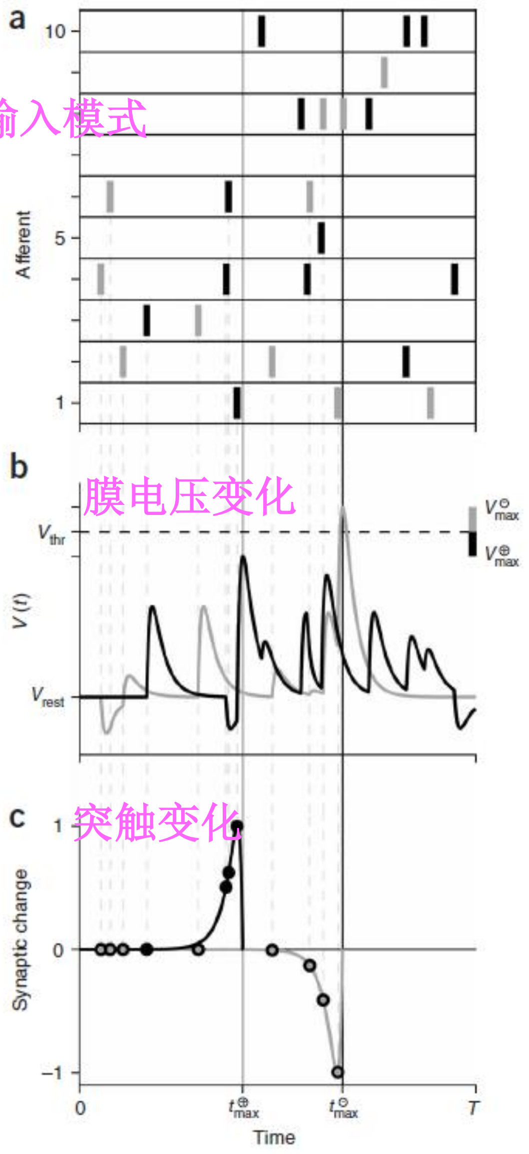
$$-\frac{\mathrm{d}E_{\pm}}{\mathrm{d}\omega_i} = \pm \sum_{t_i < t_{\max}} K(\Delta t_i) \pm \frac{\partial V(t_{\max})}{\partial t_{\max}} \frac{dt_{\max}}{d\omega_i}$$

The potential V gets the maximal value when its derivative equals to 0.
So the gradients of weights can be computed as

$$\Delta\omega_i = \lambda \sum_{t_i < t_{\max}} K(t_{\max} - t_i)$$

Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

Training performance analysis



(a): 10 afferent inputs in two input patterns, ⊕ (black) and ⊖ (gray);

(b):Resulting postsynaptic voltage traces V(t). Maximal voltages were reached at , $t_{max}^{\oplus}$ and $t_{max}^{\ominus}$ respectively.

Because $V(t_{max}^{\oplus}) < V_{thr} < V(t_{max}^{\ominus})$ with V_{thr}(dashed horizontal line) denoting the spike threshold, both patterns generate an error. Inputs arriving after a threshold crossing (gray trace) are shunted.

Black and gray thick vertical lines indicate the cost terms:

$$V_{max} - V_{thr} \text{ and } V_{thr} - V_{max}$$

associated with the ⊕ and ⊖ patterns, respectively.

(c): Resulting synaptic changes depend on presynaptic spike times (circles) relative to the corresponding voltage maximum.

Thin dashed vertical lines in (a–c) mark presynaptic spike times.

Output coding

- Since the output neuron has only two states: output 1 and no output 0, how to represent output multiple categories when applied to classification tasks? We can use the binary coding method.
- If the number of categories is 26, which is between 16 (2^4) and 32 (2^5), then 26 categories can be represented by 5-bit binary numbers so the number of output neurons is 5.
- For example, 00000 means A, and 00001 means B, and so on.

Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

Tempotron code in matlab——二分类

The main process

```
close all;
clear all;clc;
max_time = 500e-3; % 时间窗口=0.5s
thr=1; % 膜电位阈值=1V
n=2; % 类别数量=2
Aff=10; % 传入神经元数量=10
% 两类随机脉冲模式
spike{1,1} =
{{[1,70],[1,300],[1,420],[3,300],[3,420],
[5,150],[6, 200],[7,250],[7,
280],[7,350],[8, 390],[9, 350],[10,
150]},1};

spike{2,1} =
{{[2,390],[3,50],[3,180],[5,80],[5,300],[
7,280],[8,120],[9,260],[9,330],[10,230],[
10,340]},0};

% 随机生成初始权重
weight = rand(1,Aff);
```

```
w2=[]; % 记录每次更新的权重
acc = 0; % 准确率
count = 0; % 记录迭代次数
while acc ~= 1 % 迭代训练直到正确率是1
    for i=1:1:n % 遍历所有模式
       [spike_list,w]=tempotron(spike{i},thr,max_time,
1,weight); % 调用Tempotron训练函数
        weight= w;
    end
    % 每2次迭代测试1次精度
    if mod(count,2) == 0
       acc=test(spike,thr,max_time,weight); % 调用测试函数
       disp([count,acc]);
       w2=[w2;weight];
    end
    count = count+1;
end
```

Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

Tempotron code in matlab

The training process

```
function [psp_spike,w]=tempotron
(spike_list,thr,max_time,isTrain,weight)
% spike_list 单个脉冲模式+标签
% thr 膜电位阈值
% max_time 最大时间窗口
% isTrain 是否训练
% weight 训练权重
% psp_spike 输出脉冲时间
% w 更新后权重
spike = spike_list{1}; % 输入脉冲模式
target_label = spike_list{2}; % 类别标签
Aff = max(spike(:,2));
[size_spike,~] = size(spike);
dt=1e-3; % 时间分辨率 1ms
total_time=max_time; % 最大时间窗口
tau1=10e-3; % 时间常数1=10ms
tau2=5e-3; % 时间常数2=5ms
tau_m = 50e-3; % 膜电位时间常数=50ms
ref_time= 100e-3; % 不应期时间 100ms
V0 = 1/max(exp(-(0:dt:total_time)/tau1)-exp(-
(0:dt:total_time)/tau2)); % PSP核函数归一化
v_thr =thr;
v_rest = -100e-3; % 复位膜电压
learning_rate=0.1; % 学习率=0.1
new_spike_time = total_time; % 记录脉冲发放时刻
v = zeros(1,total_time/dt); % 记录每个时刻的膜电位
I_syn=zeros(1,total_time/dt); % 记录每个时刻的突触电流
v(1)=v_rest; % 记录初始膜电位=复位膜电压
psp_spike = [];
```

```
for t=dt:dt:total_time
    %% LIF 神经元模型动力学计算
    v(round(t/dt)+1) = v(round(t/dt));
    % 判断不应期：在不应期内则突触电流置零；不在不应期则计算突触电流
    if t>new_spike_time && t<new_spike_time+ref_time % 不应期内
        I_syn(round(t/dt))=0;
    else
        I_syn(round(t/dt))=0; % 非不应期
        for i=1:1:size_spike % 遍历所有输入脉冲
            input = cell2mat(spike(i));
            index = input(1); % 突触前脉冲地址
            t_spike = input(2); % 突触前脉冲发放时间
            w=weight(round(index));
            % 计算PSP核函数K(t-t_spike)
            if t>=t_spike
                K = V0*(exp(-(t-t_spike)/tau1)-exp(-(t-t_spike)/tau2));
            else
                K=0;
            end
            I_syn(round(t/dt))= I_syn(round(t/dt))+w*K; % 计算突触电流I_syn
        end
    end
    % 计算膜电位增量dv
    dv=dt*(-v(round(t/dt))+v_rest+I_syn(round(t/dt)))/tau_m;
    % 更新膜电位
    v(round(t/dt)+1) = v(round(t/dt))+dv;
    % 判断是否发放脉冲，复位
    if v(round(t/dt)+1)>=v_thr && ~(t>new_spike_time && t<new_spike_time+ref_time)
        new_spike_time = t; % 更新突触后脉冲发放时刻
        psp_spike = [psp_spike,t]; % 记录脉冲发放时间
    end
end
```

Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

Tempotron code in matlab

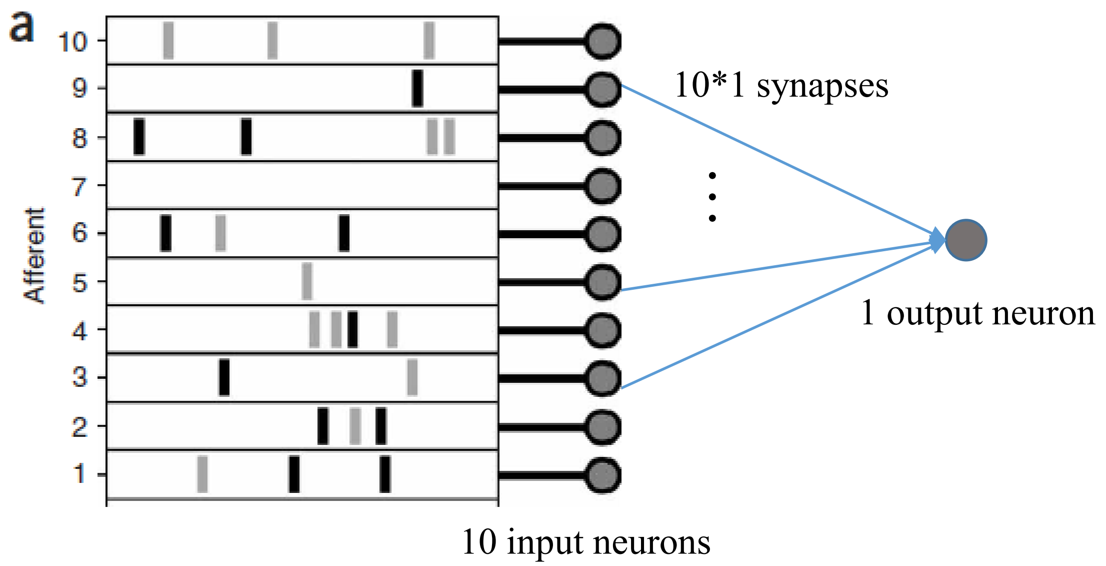
The training process

```
%% 找到最大膜电位，并判断发放状态
[v_max,v_max_index]=max(v);
if v_max>=v_thr
    label=1; % 发放
else
    label=0; % 不发放
end
%% 当发放错误时,调整权重
% 错误情况: target=1, label=0（没发放）; target=0, label=1 (发放),
if (isTrain==1)&&(label~=target_label)
    % 找到最大膜电位的时刻，计算该时刻之前的PSP作为权重调整量
    t_max = v_max_index*dt; % 最大膜电位时刻 t_max
    dw = zeros(1,Aff);
    % 计算该时刻之前的PSP
    for i=1:1:size_spike
        input = cell2mat(spike(i));
        index = input(1); % 突触前脉冲地址
        t_spike = input(2); % 突触前脉冲发放时间
        if t_max>=t_spike
            K = V0*(exp(-(t_max-t_spike)/tau1)-exp(-(t_max-t_spike)/tau2));
        else
            K=0;
        end
        dw(index) = dw(index) + K;
    end
    dw = learning_rate * dw;
    %% 判断更新方向，更新权重
    if target_label==1
        weight = weight + dw;
    else
        weight = weight - dw;
    end
end
w = weight;
```

```
%% 绘制突触后膜电位与训练后权重
subplot(1,4,1)
scatter(spike(:,1),spike(:,2),2)

subplot(1,4,2)
title('I');
plot(dt:dt:total_time, I_syn)
subplot(1,4,3)
plot(dt:dt:total_time, v(1:total_time/dt))
hold on;
axis([0 1 v_rest-0.1 v_thr*1.1])
plot([0, max_time],[v_rest,v_rest])
plot([0, max_time],[v_thr,v_thr])
hold off;
% [~,n] = size(psp_spike);
% stem(psp_spike,1.1*v_thr*ones(n),'--k')
subplot(1,4,4)
% plot(w)
histogram(w,20)
end
```

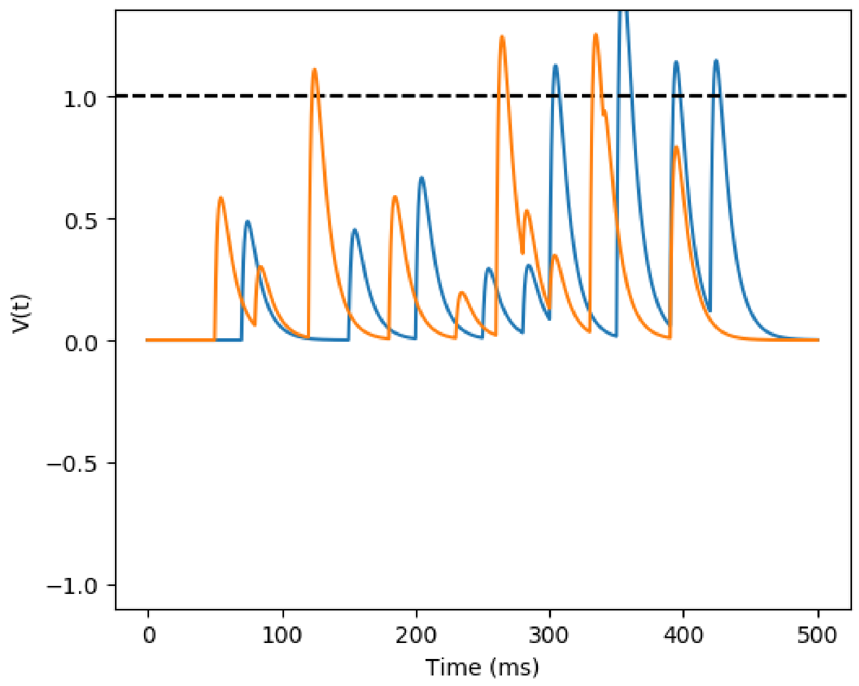
Tempotron simulation in matlab

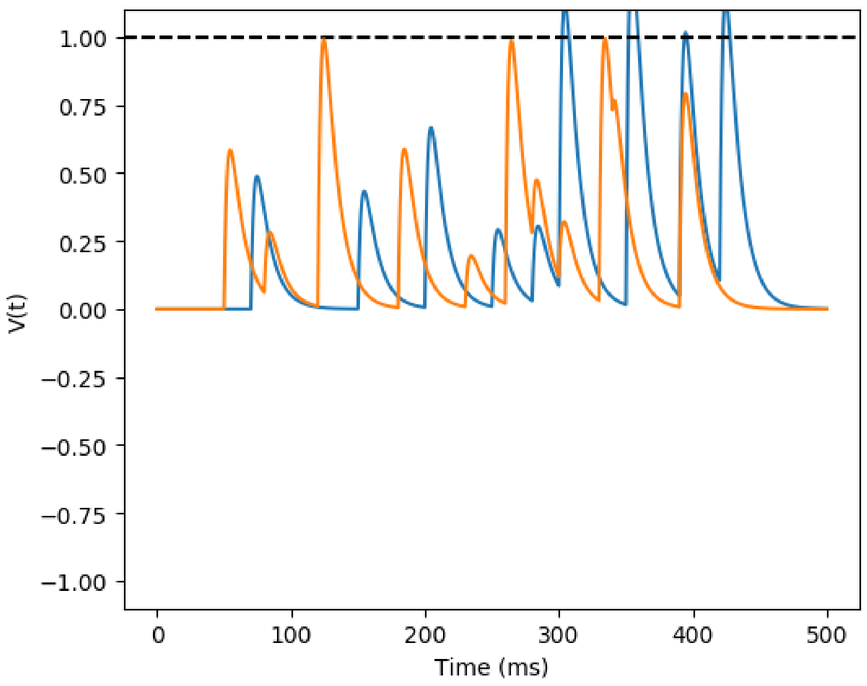
We use the matlab to simulate the Tempotron process. The network structure is :



We randomly generate spike patterns which belong to two categories respectively.

Tempotron simulation in Matlab





The left figure is before training, and the right figure is after training. The red line shows the changes in the potential of neurons that not belong to this category, and the blue line shows the changes in the potential of neurons that belong to this category. It can be clearly seen that those that do not belong to this category no longer reach the threshold, and those belonging to this category reached the threshold.

Tempotron

Characteristics of Tempotron:

- Minimizes an error based on the membrane potential
- Classification of spatio-temporal spike patterns using single neuron
- Spatio-temporal spike pattern classification
- Binary output (spike or no spike), not suitable for spike sequence learning
- Both suitable for online (incremental) and batch learning