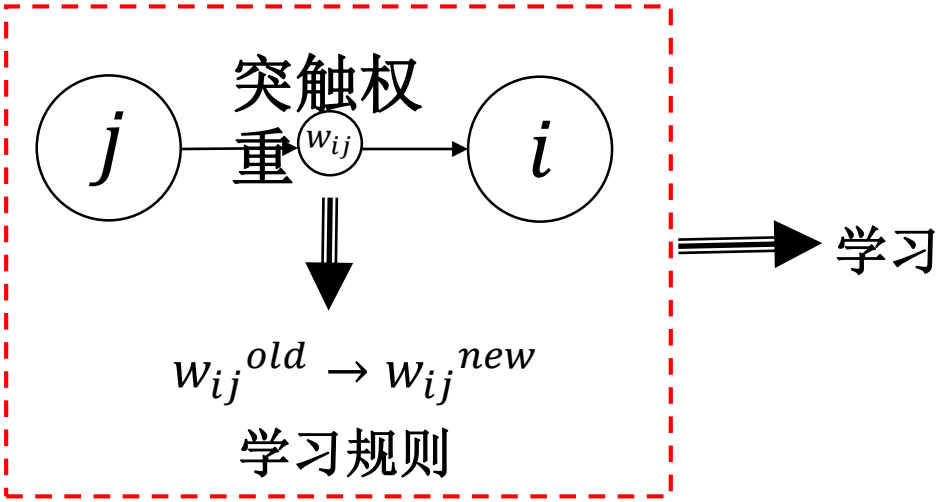
脑启发人工智能导论
Introduction to Brain-Inspired Artificial Intelligence

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

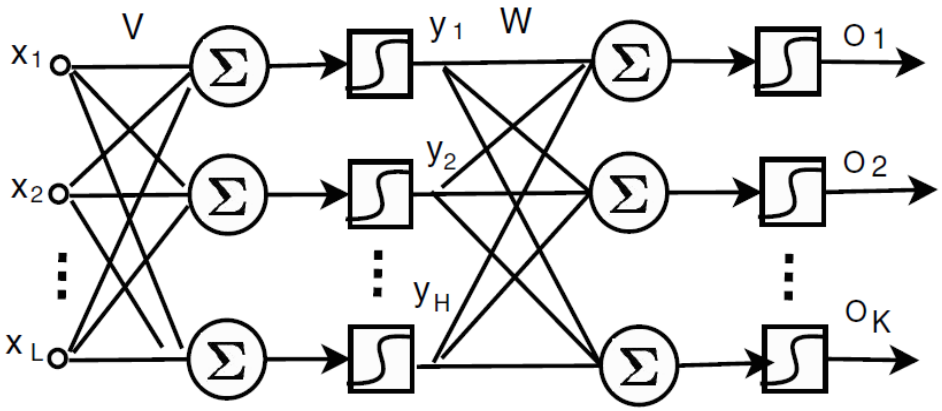
General Learning Process

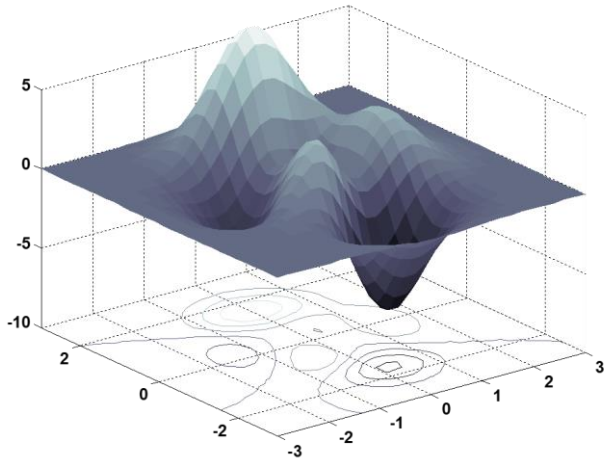
- **突触权重/效能**(synaptic weight/ efficacy):神经网络中，从神经元j到i的突触连接由权重参数w_{ij}表示，该参数决定了传入动作电位（action potential）的突触后响应幅度，可以调整以优化给定任务的网络性能。

- **学习&学习规则：**
  - ✓ 参数自适应过程称为学习；
  - ✓ 调整权重的过程称为学习规则。

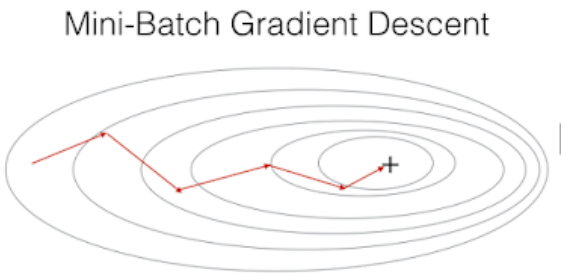


Gradient Descent in Optimization of Multilayer Networks





The idea is to use a gradient descent over the space of weights to find a global minimum (no guarantee).



Gradient Descent Method in Optimization

Back-Propagation Algorithm:

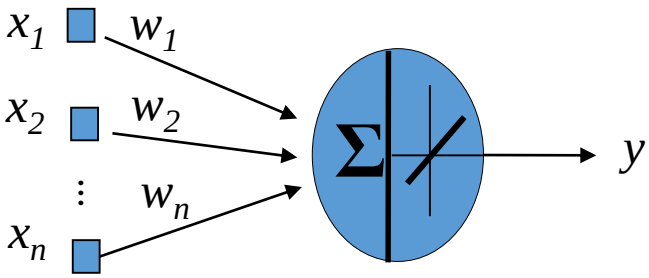
- Goal: To learn the weights for all links in an interconnected multilayer network.

- We begin by defining our measure of error:

$$E(W) = \frac{1}{2}\sum_d \sum_k (t_{kd} - o_{kd})^2 = \frac{1}{2}\sum_{example} (t - o)^2 = \frac{1}{2} Err^2$$

  - k varies along the output nodes and
  - d over the training examples.

Supervised Learning and Gradient Descent



自适应线性神经网ADLINE络
(ADaptive LInear NEuron)

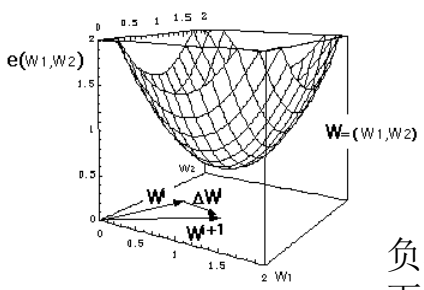
根据误差函数和梯度，计算最优的权值W:

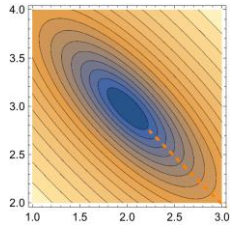
W=？？

Gradient descent, is a more general way of finding the minimum of an error function. It can be used when the error function is much more complicated (e.g. it isn't quadratic), and there is no linear solution.

$$\mathbf{W}^{i+1} = \mathbf{W}^{i} - \eta \left.\frac{\partial e}{\partial \mathbf{W}}\right|_{\mathbf{W}^i}$$

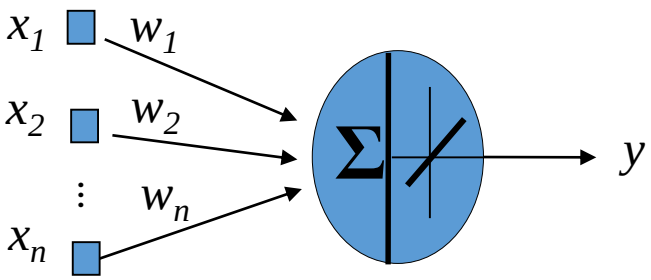
$$\frac{d\mathbf{W}}{dt} = -\frac{\partial e}{\partial \mathbf{W}} = -\nabla e$$





负梯度下降方向（能量函数下降最快的方向）

Supervised Learning and Gradient Descent



自适应线性神经网ADLINE络
(ADaptive LInear NEuron)

计算误差和梯度:

$$e(\mathbf{W}) = \sum_{i=1}^{N} \left| \mathbf{y}_i - \mathbf{W}\mathbf{x}_i \right|^2$$

$$\frac{\partial e}{\partial \mathbf{W}} = -2\sum_{i=1}^{N} \left(\mathbf{y}_i - \mathbf{W}\mathbf{x}_i \right) \mathbf{x}_i^T = 0$$

$$\sum_{i=1}^{N} \mathbf{y}_i \mathbf{x}_i^T - \mathbf{W} \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T = 0$$

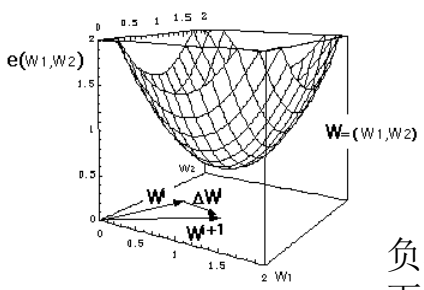
$$W = \sum_{i=1}^{N} \mathbf{y}_i \mathbf{x}_i^T \left(\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T \right)^{-1}$$

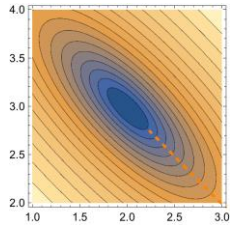
最小均方差方法
(线性回归方法)

Gradient descent, is a more general way of finding the minimum of an error function. It can be used when the error function is much more complicated (e.g. it isn't quadratic), and there is no linear solution.

$$\mathbf{W}^{i+1} = \mathbf{W}^{i} - \eta \left. \frac{\partial e}{\partial \mathbf{W}} \right|_{\mathbf{W}^i}$$

$$\frac{d\mathbf{W}}{dt} = -\frac{\partial e}{\partial \mathbf{W}} = -\nabla e$$





负梯度下降方向（能量函数下降最快的方向）

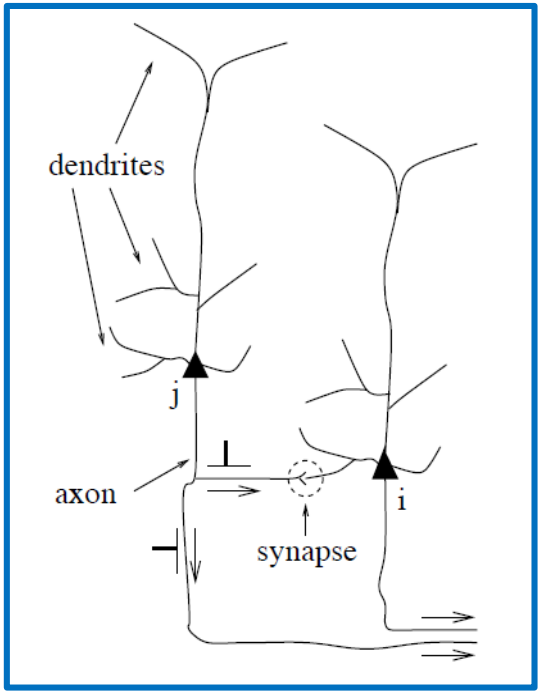
Gradient Descent Based Supervised Learning Algorithms

- Tempotron
- PSD

How to learn the knowledge

- In neural network model, each synapse is characterized by a single constant parameter **wij** that determines the **amplitude of the postsynaptic response to an incoming action potential**.
- The process of parameter adaptation is called **learning** and the algorithm for adjusting the weights is referred to as a **learning rule**.
- Learning is to find the optimal **parameters for a given task.**

Learning Rule → **Optimize the performance of a network**



Tempotron

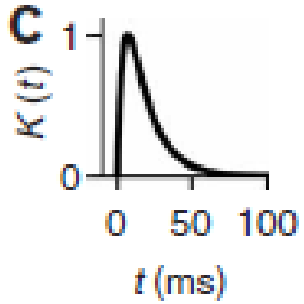
Assuming a leaky integrate-and-fire-model the potential $V(t)$ of the synapse can be described by :

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

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

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

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



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

Gradient Descent Based Learning: Tempotron

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.

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

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

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

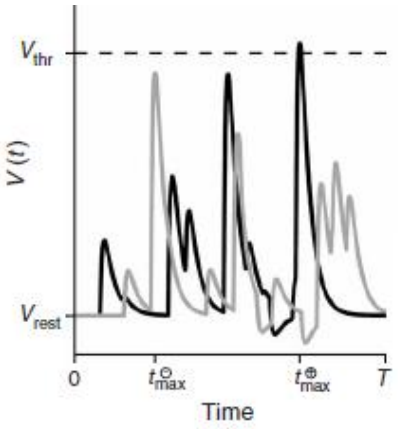
Gütig R, Sompolinsky H. The tempotron: a neuron that learns spike timing–based decisions[J]. Nature neuroscience, 2006, 9(3): 420-428.

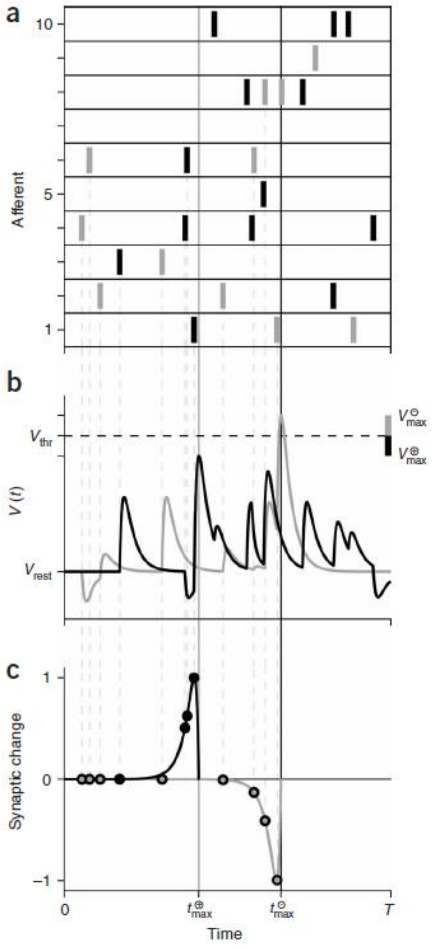
Tempotron Learning Rule

We need to get the t_{max}. The potential V gets the maximal value when its derivative equals to 0:

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

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





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

Leaky Integrate-and-Fire Model

- Leaky Integrate and Fire (LIF) 膜电压变化满足：

$$\tau_{\mathrm{m}} \frac{du(t)}{dt} = -[u(t) - u_{rest}] + R_{\mathrm{m}} I(t)$$

τ_{m} : Membrane time constant

$u(t)$: Membrane potential

u_{rest} : Resting potential

R_{m}: Membrane resistance

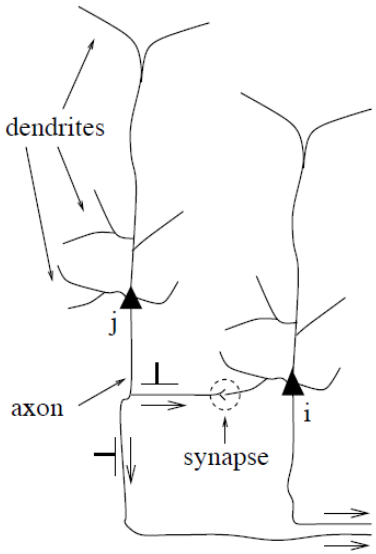
$I(t)$: Total input current

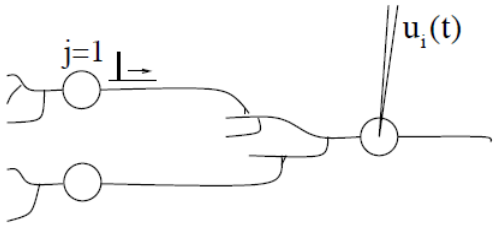
神经动力学

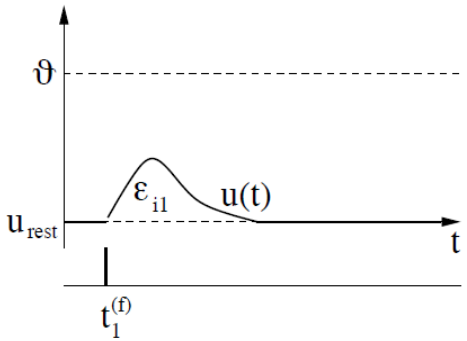
- 定义神经元i的膜电压为$u_i(t)$，其静息态膜电压为u_{rest}；
- $t=0$, 突触前神经元j发放一个脉冲， $t>0$ 突触后电位(PSP)

$$\varepsilon_{ij}(t) = u_i(t) - u_{rest}$$

- 如果PSP是正的，定义为兴奋型（Excitatory PSP），简称EPSP；
  反之定义为抑制型（Inhibitory PSP），简称IPSP。

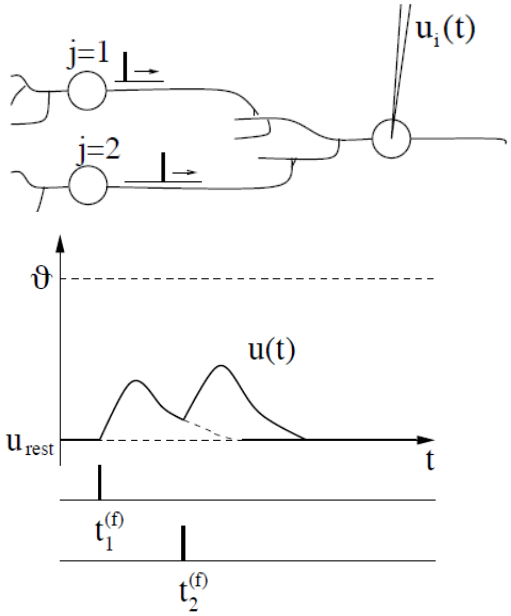


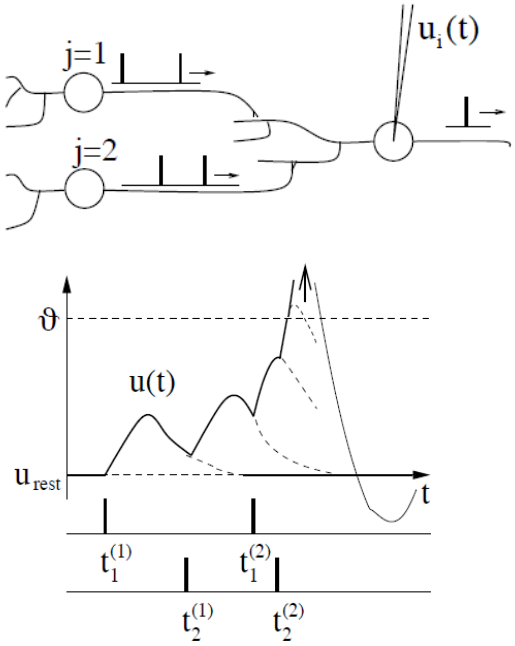




EPSP示意图

脉冲输入对突触后神经元的影响





- 分别定义突触前神经元j， $j=1$，$2\ldots$，和突触后神经元i;
- 突触前神经元j发放的脉冲为$t_j^f: t_j^1, t_j^2, \ldots,$

  突触后膜电压计算公式：

$$u_i(t) = \sum_j \sum_f \epsilon_{ij}\left(t - t_j^{(f)}\right) + u_{rest}$$

突触前神经元的输入激励

总的突触后电流表示为：$I(t)=\sum_j W_j \sum_f \alpha(t-t_j^{(f)})$

$t_j^{(f)}$ 表示第j个突触前神经元的第f个脉冲的时间

$\alpha(t-t_j^{(f)})$ 表示突触后神经元在这个时间过程内能感受到电流

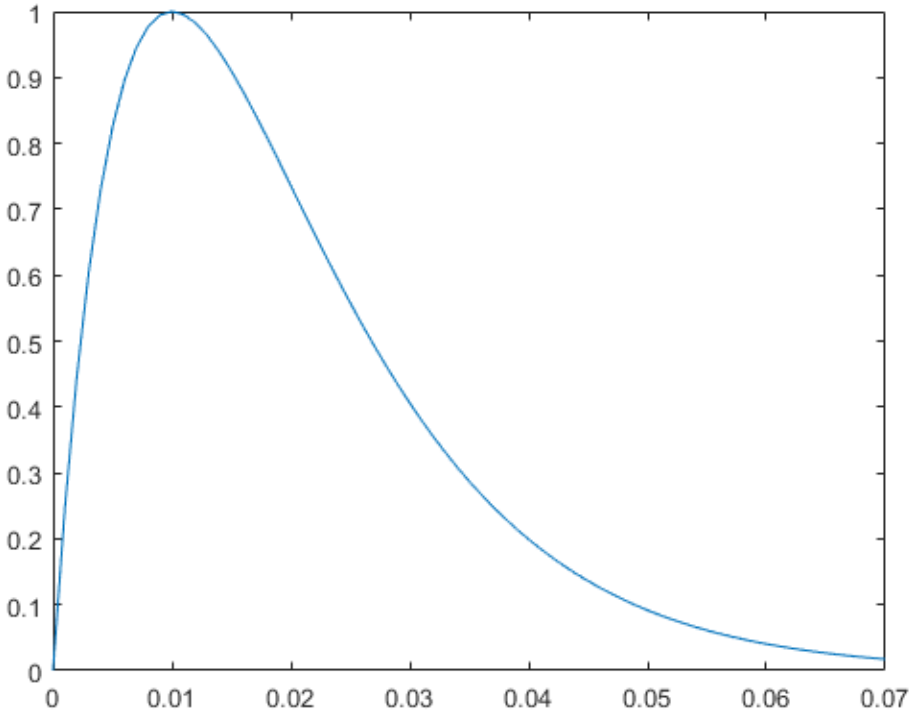
α常见的形式：（1） $\alpha(t)=\alpha\frac{t}{\tau}\exp(1-\frac{t}{\tau})$

（2） $\alpha(t)=\beta\frac{\tau_1}{\tau_2-\tau_1}[\exp(-\frac{t}{\tau_1})-\exp(-\frac{t}{t_2})]$

通过突触前神经元的电流刺激

（1） $\alpha(t) = \alpha \frac{t}{\tau} \exp(1 - \frac{t}{\tau})$

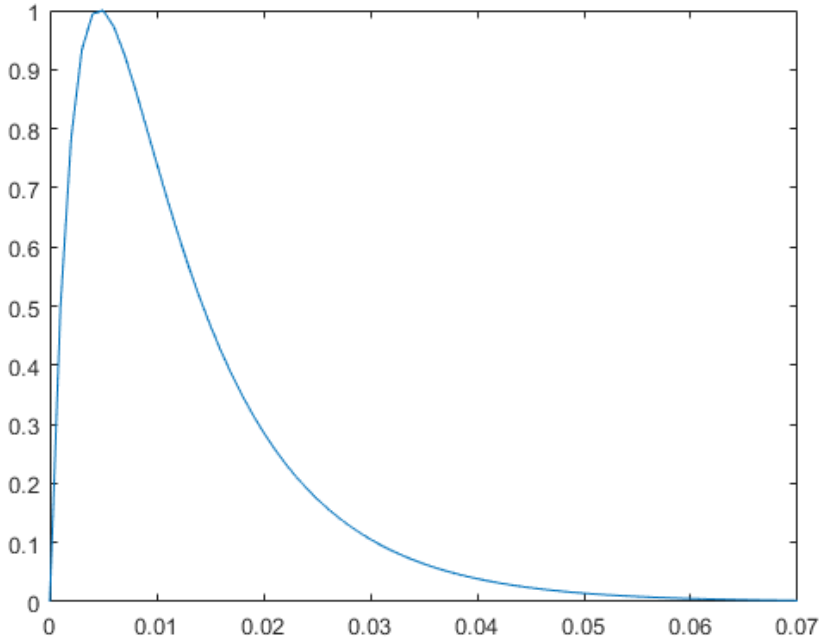
阈值为15mV，τ设为10ms，核函数如下：



通过突触前神经元的电流刺激

(2) $\alpha(t) = \beta \frac{\tau_1}{\tau_2 - \tau_1}[\exp(-\frac{t}{\tau_1}) - \exp(-\frac{t}{t_2})]$

τ_1 设为10ms, τ_2 设为5ms，核函数如下：



The spiking neuron model

The leaky integrate-and-fire (LIF) model is firstly considered

$$\tau_m \frac{dV_m}{dt} = -(V_m - E) + (I_{ns} + I_{syn})R_m$$

- Vm is the membrane potential, E is the resting potential, Ins and Isyn are the background current noise and synaptic current, respectively.
- When Vm exceeds a constant threshold Vthr, the neuron is said to fire, and Vm is reset to Vreset for a refractory period t_ref .
- We set E=Vreset=0mV and Vthr=E+18mV.

The spiking neuron model

For the postsynaptic neuron, we model the input synaptic current as:

$$I_{syn}(t) = \sum_i w_i I^i_{PSC}(t)$$

The postsynaptic current from the corresponding afferent is given by:

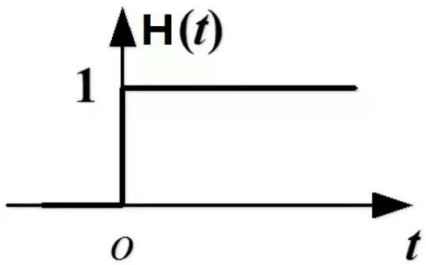
$$I^i_{PSC}(t) = \sum_{t^j} K(t - t^j) H(t - t^j)$$

K denotes a normalized kernel

$$K(t - t^j) = V_0 \cdot \left(\exp\left(\frac{-(t - t^j)}{\tau_s} \right) - \exp\left(\frac{-(t - t^j)}{\tau_f} \right) \right)$$

H(t) refers to the Heaviside function

$$H(t) = \begin{cases} 0, & t < 0, \\ 1, & t \geq 0. \end{cases}$$

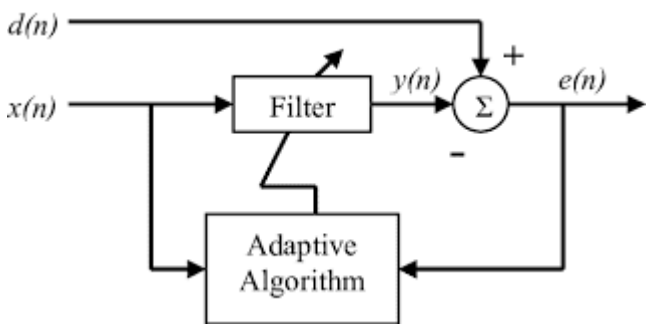


Gradient Descent Based Supervised Learning Algorithms

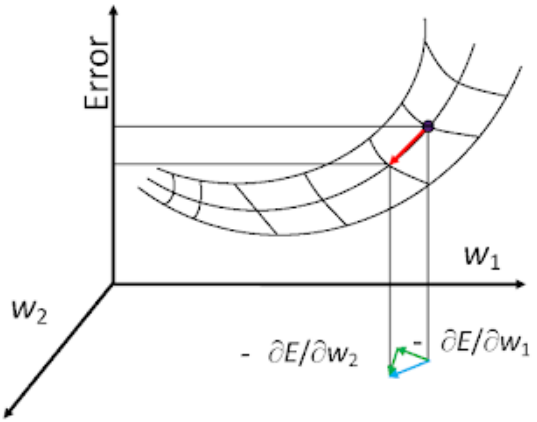
- Tempotron
- PSD

Gradient Descent

- Can we find the optimal W in such a way so as to be biologically plausible, and avoid having to invert a large matrix?



An unknow multi-input single-output dynamic system, analogous to a linear neuron model.



Gradient descent in 2D weight space.

The weight update given by: $$\mathbf{w}(m+1) = \mathbf{w}(m) + \Delta\mathbf{w}(m)$$

It is equivalent to: $$\mathbf{w}(m+1) = \mathbf{w}(m) - \eta\frac{\partial E(\mathbf{w}(m))}{\partial\,\mathbf{w}(m)}$$

Gradient Descent: Widrow-Hoff

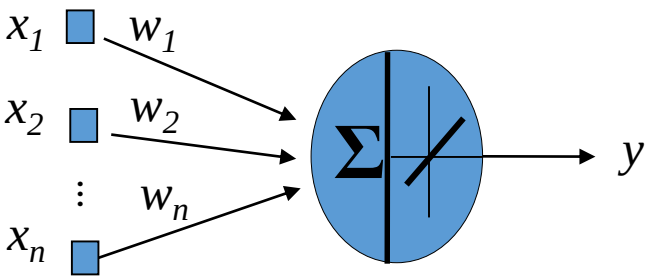
- The **Widrow-Hoff Learning (also called Delta Learning Rule)** provides an answer. The basic idea behind Widrow-Hoff learning is to update W iteratively with each new training pair.

$$e(\mathbf{W}) = \left|\mathbf{y}_i - \mathbf{W}\,\mathbf{x}_i\right|^2$$

$$\frac{\partial e}{\partial \mathbf{W}} = -2\left(\mathbf{y}_i - \mathbf{W}\,\mathbf{x}_i\right)\mathbf{x}_i^T$$

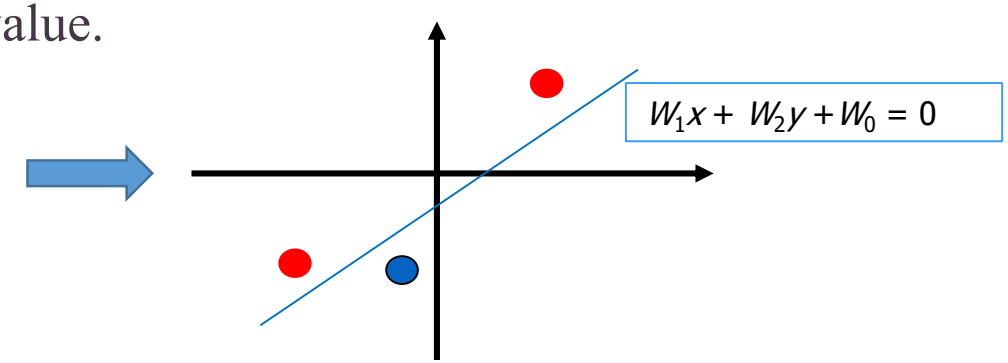
$$\frac{\partial e}{\partial \mathbf{W}} = -\frac{d\mathbf{W}}{dt}$$

$$\mathbf{W}^{i+1} = \mathbf{W}^i + \eta_i\left(\mathbf{y}_i - \mathbf{W}^i\mathbf{x}_i\right)\mathbf{x}_i^T$$



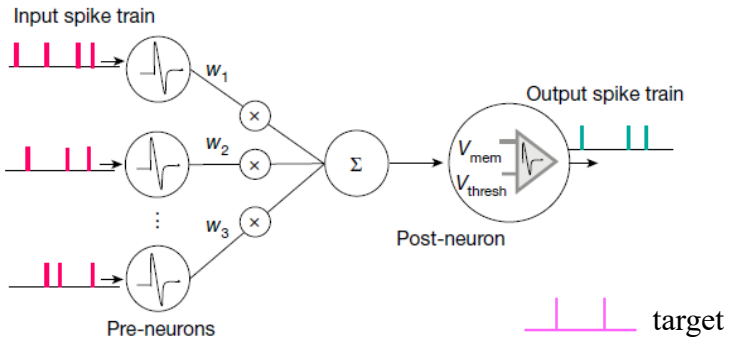
- This learning rule is found to minimize the mean-squared error between the activation and the target value.

可以用来训练线性分类器



Supervised Learning in Spiking Neural Networks

- Set up the input and output pairs, which are spatiotemporal spike patterns.
- Compute the error between the expected spike train and the actual output spike train.
- Minimize the error by adjusting the weights until find the optimal weights.

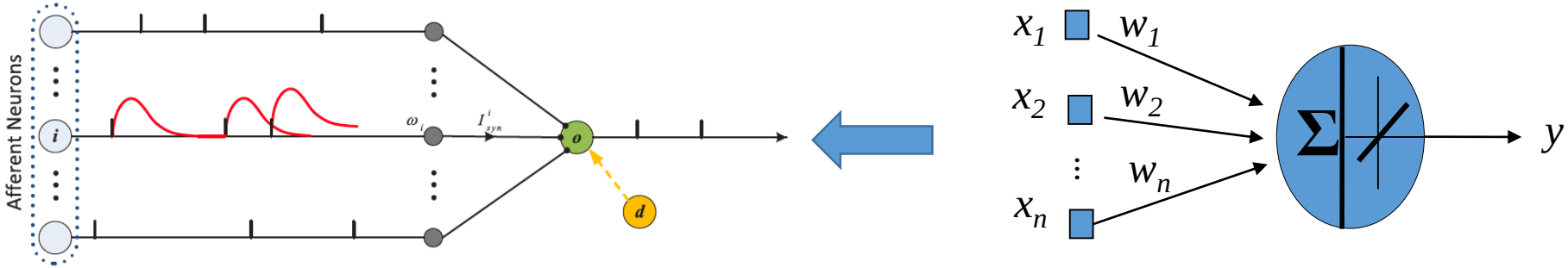


基于突触可塑性，重新思考 Widrow-Hoff 学习规则

For a synapse i, it is defined as:

$$\Delta w_i = \eta x_i (y_d - y_o)$$

η is a positive constant referring to the learning rate, x_i , y_d and y_o refer to the input, the desired output and the actual output, respectively



- In SNN, input and output are represented by spike trains. A spike train is defined as a sequence of impulses triggered by a particular neuron at its firing time. A spike train is expressed in the form of:

$$s(t) = \Sigma_f \delta(t - t^f)$$

where t^f is the f-th firing time, and $\delta(x)$ is the Dirac function

$$\delta(x) = 1 \ (if \ \ x = 0) \text{ or } 0 \ (otherwise).$$

- However, The difference between two spike trains does not define a suitable error landscape which can be minimized by a gradient descent.

- Directly applying Widrow-Hoff rule is not possible for SNN.

Precise-Spike-Driven (PSD)

Thus, the input, the desired output and the actual output of the spiking neuron are described as:

$$\begin{cases} s_i(t) = \Sigma_f \delta(t - t_i^f) & \textit{the input spike train:} \\ s_d(t) = \Sigma_g \delta(t - t_d^g) & \textit{the desired output:} \\ s_o(t) = \Sigma_h \delta(t - t_o^h) & \textit{the actual output:} \end{cases}$$

The products of Dirac functions are mathematically problematic. To solve this difficulty, we apply an approach called spike convolution.
We convolve the input spike trains.

$$\tilde{s}_i(t) = s_i(t) * \kappa(t)$$

where k(t) is the convolving kernel,

In this case, the convolved signal is in the same form as I_{PSC} in the above. Thus, we use I_{PSC} as the eligibility trace for the weight adaptation. The learning rule becomes:

$$\frac{dw_i(t)}{dt} = \eta[s_d(t) - s_o(t)]I^i_{PSC}(t)$$

It can be seen that the polarity of the synaptic changes depends on three cases:
(1) a positive error (corresponding to a miss of the spike) where the neuron does not spike at the desired time,
(2) a zero error (corresponding to a hit) where the neuron spikes at the desired time, and
(3) a negative error (corresponding to a false-alarm) where the neuron spikes when it is not supposed to.

Yu Q, Tang H, Tan K C, et al. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns[J]. Plos one, 2013, 8(11): e78318.

Precise-Spike-Driven (PSD)

By integrating the gradient equation, we get

$$\Delta w_i = \eta \int_0^{\infty} [s_d(t) - s_o(t)] I_{PSC}^i(t)\, dt =$$

$$\eta[\sum_g \sum_f K(t_d^g - t_i^f) H(t_d^g - t_i^f) - \sum_h \sum_f K(t_o^h - t_i^f) H(t_o^h - t_i^f)]$$

The update of the weight is dependent on

1. the error between the desired output and the actual output,
2. the post-synaptic current.

Yu Q, Tang H, Tan K C, et al. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns[J]. Plos one, 2013, 8(11): e78318.

- Thus, the weight adaptation is triggered by the error between the desired and the actual output spikes, with positive errors causing long-term potentiation and negative errors causing long-term depression. No synaptic change will occur if the actual output spike fires at the desired time.

- The amount of synaptic changes is determined by the current $I_{PSC}^{i}(t)$.

- Each of the variables involved has its own physical meaning.

- Moreover, the weight adaptation only depends on the current states. This is different from rules involving STDP, where both the pre- and post-synaptic spiking times are stored and used for adaptation.

- Thus, it will cost less computational resources.

Yu Q, Tang H, Tan K C, et al. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns[J]. Plos one, 2013, 8(11): e78318.

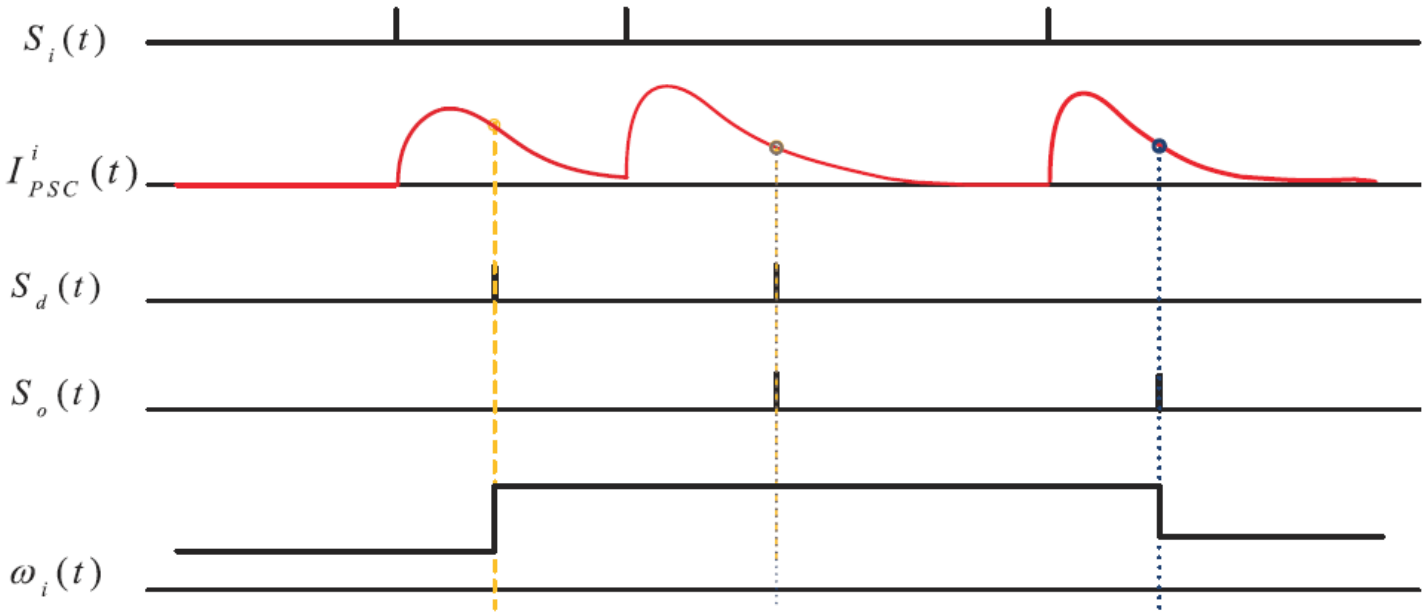


Illustration of the weight adaptation. Si(t) is the presynaptic spike train. Sd (t) and So(t) are the desired and the actual postsynaptic spike train, respectively. IPSC(t) is the postsynaptic current and can be referred to as the eligibility trace for the adaptation of wi(t).

- A positive error, where the neuron does not spike at the desired time, causes synaptic potentiation.
- A negative error, where the neuron spikes when it is not supposed to, results in synaptic depression.
- The amount of adaptation is proportional to the postsynaptic current.
- There will be no modification when the actual output spike fires exactly at the desired time.

Precise-Spike-Driven (PSD)

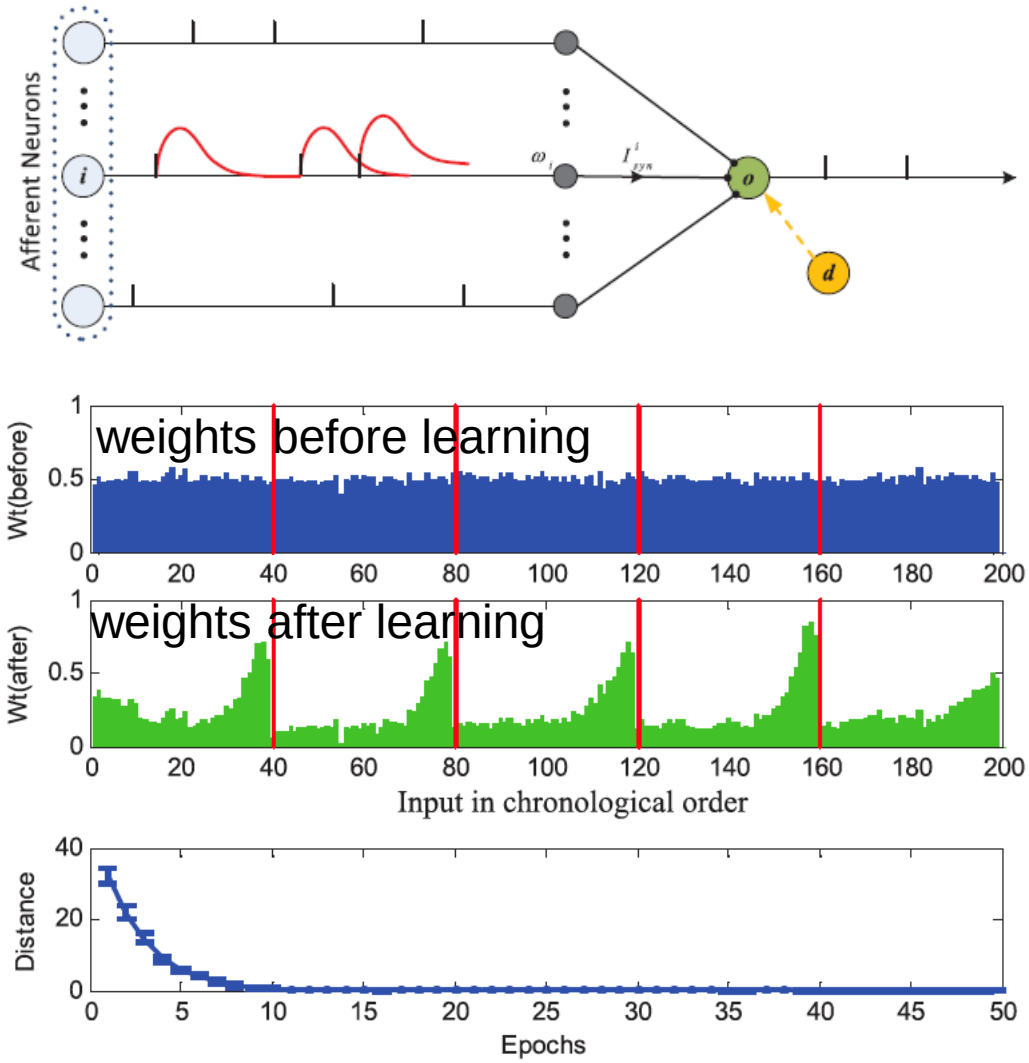
In order to measure the distance between two spike trains, we use the van Rossum metric, with a filter function given by K(t).

$$Dist = \frac{1}{\tau}\int_0^{\infty} [f(t) - g(t)]^2 dt$$

where t is a free parameter (we set t=10 ms here), f (t) and g(t) are filtered signals of the two spike trains.

Note that this distance parameter *Dist* is not involved in the PSD learning rule, but is used for measuring and analyzing the performance of the learning rule.

SNN学习的特点（1）：权值的时间结构分布



把突触前神经元按照脉冲时间先后顺序，从而排列突触的顺序。

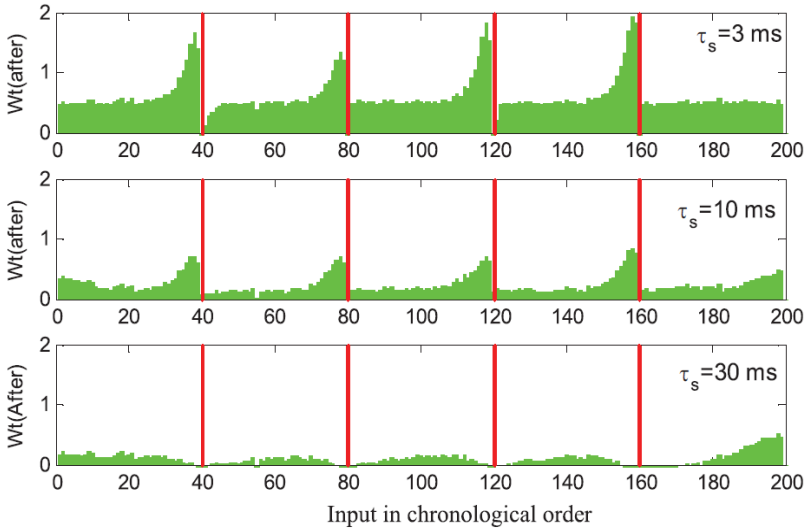
The initial weights are uniformly distributed around 0.5.

Causal weight distribution after learning. All the afferent neurons are chronologically sorted according to their spike time. The target spikes (red lines) are overlayed on the weights figure according to their time

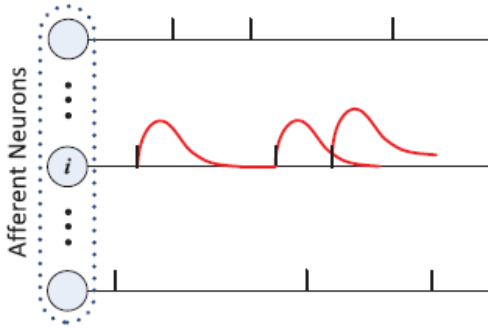
The averaged distance between the actual spike train and the desired spike train along the learning process.

SNN学习的特点（1）：权值的时间结构分布

PSP 影响权值的时间结构分布



$$K(t-t^j)=V_0\cdot(\exp(\frac{-(t-t^j)}{\tau_s})-\exp(\frac{-(t-t^j)}{\tau_f}))$$

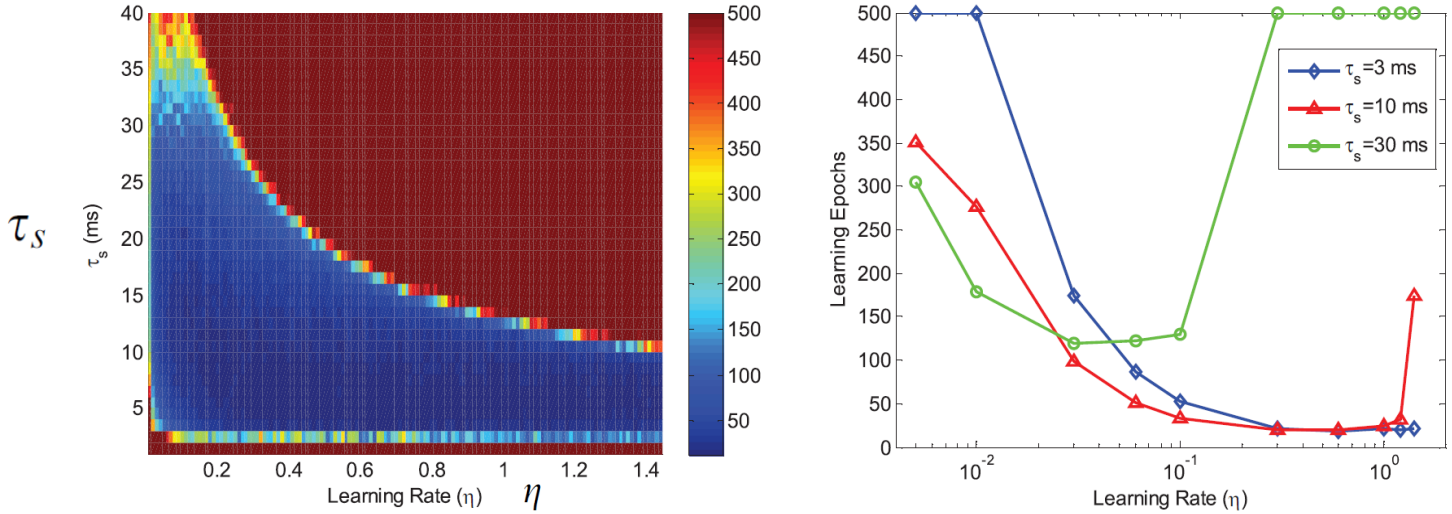


Small τ_S results in strong causal weight distribution.

- As a decay constant, τ_S is an important parameter involved in the postsynaptic current. It determines how long a presynaptic spike will still have causal effect on the postsynaptic neuron.
- A smaller τ_S can result in a very sharp distribution, higher value result in a flat distribution.
- With a larger τ_S ., a wider range of causal neighbors can contribute to generating the desired spikes, and therefore a lower synaptic strength will be sufficient.

SNN学习的特点（2）：学习率和时间常数的共同影响

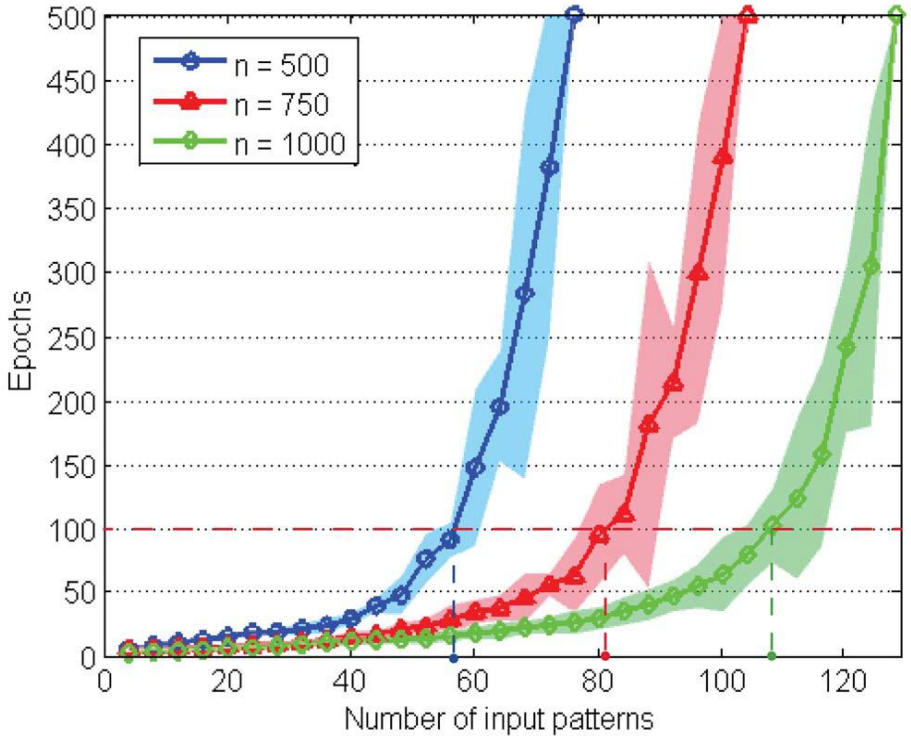
η and τ_s jointly affect the learning performance



- With a fixed τ_S , a larger η results in a faster learning speed; but when η is increased above a critical value (e.g., 0.1 for ts~30 ms), the learning will slow down or even fail.
- For smal η , a larger τ_S leads to a faster learning, however, for large η, a larger τ_S has the opposite effect.
- As a consequence, when τ_S is set in a suitable range (e.g., [5,15] ms), a wide range of η can result in a fast learning speed (e.g., below 100 epochs).

SNN学习的特点（3）：突触数量对学习容量的影响

除了学习精度，学习容量是另一种重要的学习性能指标。

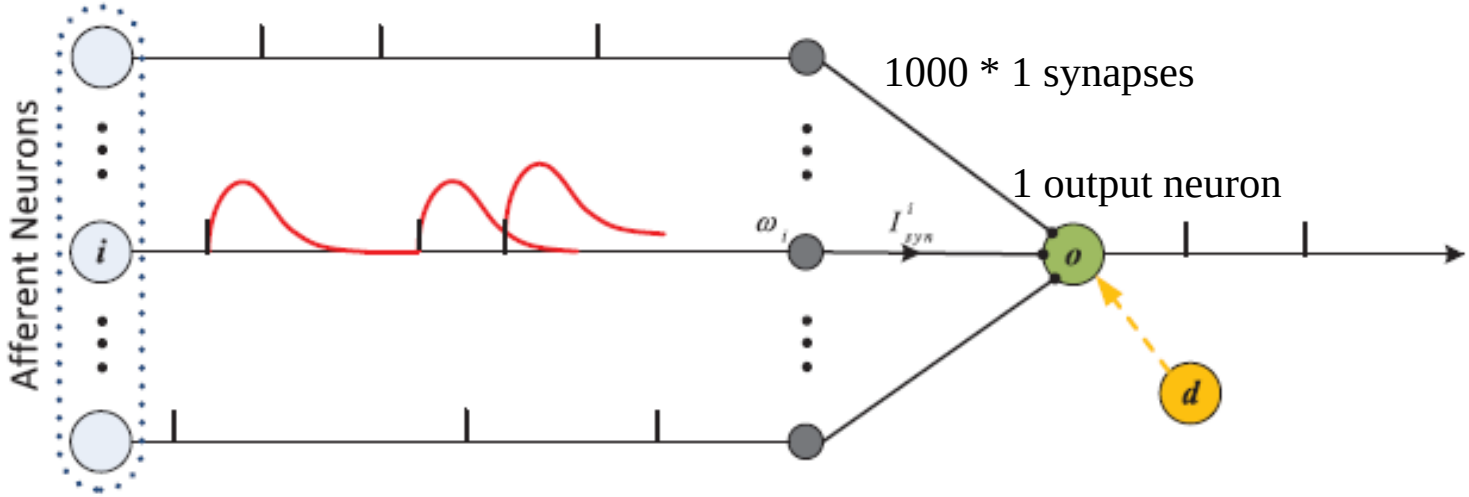


The neuron is trained to memorize all patterns correctly in a maximum number of 500 epochs. The reaching points of 500 epochs are regarded as failure of the learning. The cases of 500, 750 and 1000 synapses are denoted by blue, red and green parts, respectively.

It shows that a larger number of synapses leads to a bigger memory capacity for the same neuron.

Precise-Spike-Driven (PSD) simulation in matlab

We shows the simulation of Learning Hetero-Association of Spatiotemporal Spike Patterns. The network structure is illustrated as followed:

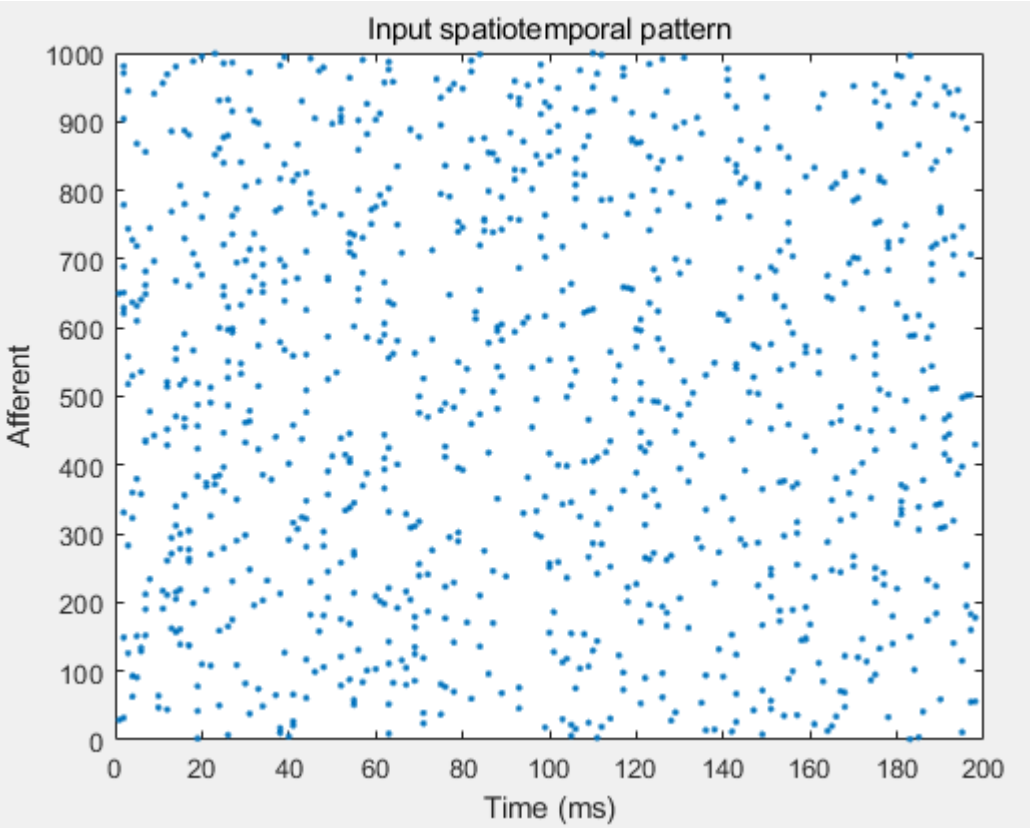


1000 input neurons

Yu Q, Tang H, Tan K C, et al. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns[J]. Plos one, 2013, 8(11): e78318.

Precise-Spike-Driven (PSD) simulation in matlab

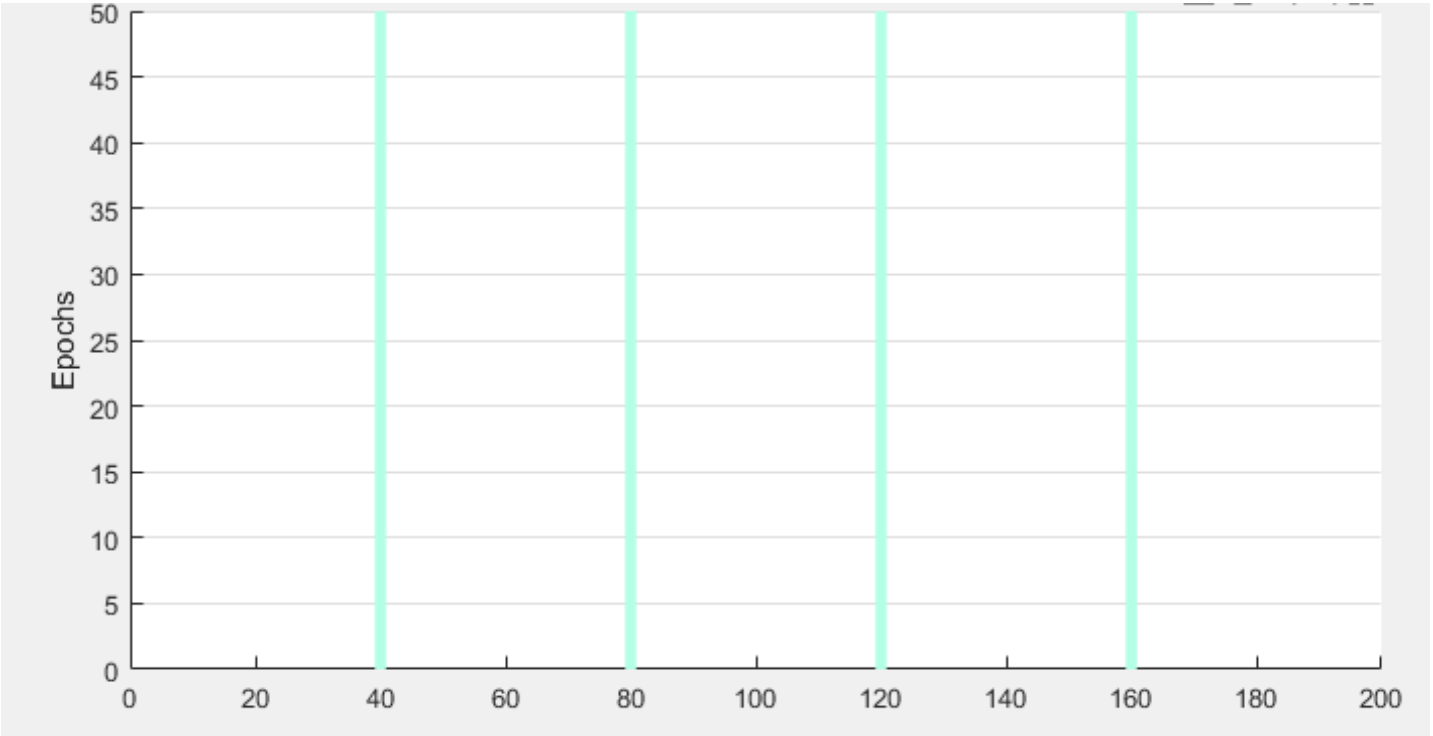
We uniformly generate the input pattern:



Yu Q, Tang H, Tan K C, et al. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns[J]. Plos one, 2013, 8(11): e78318.

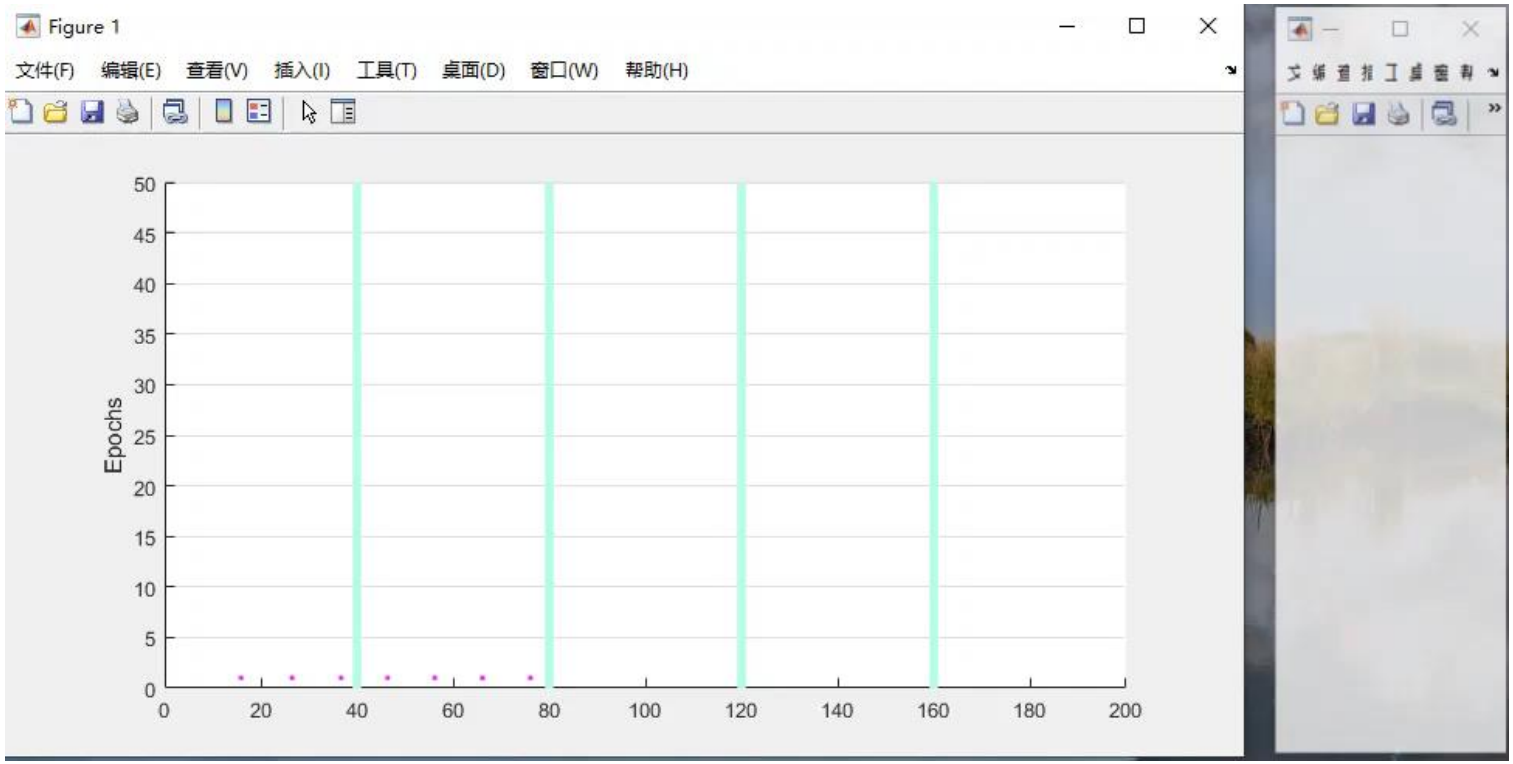
Precise-Spike-Driven (PSD) simulation in matlab

Light blue bars in the middle means the target time that neuron reproduces spikes.



Precise-Spike-Driven (PSD) simulation in Matlab

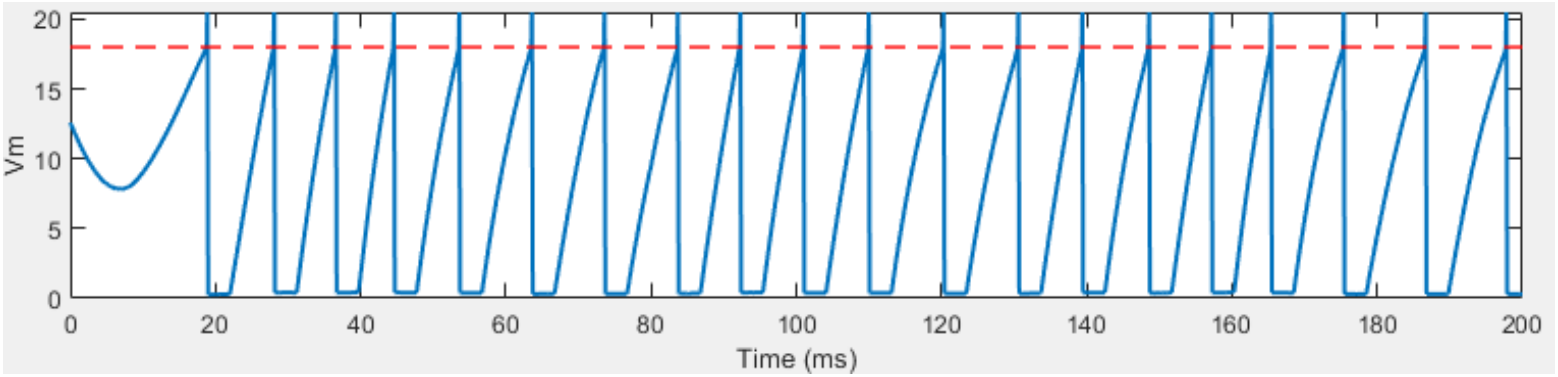
This video shows the feasibility of the PSD rule to train the neuron to reproduce a desired spike train. The right figure shows the distance between the actual output spike train and the target spike train. After several learning epochs, the neuron can successfully spike at the target time. In other words, the proposed rule is able to train the neuron to associate the input spatiotemporal pattern with a desired output spike train within several training epochs. The information of the input pattern is stored by a specified spike train.



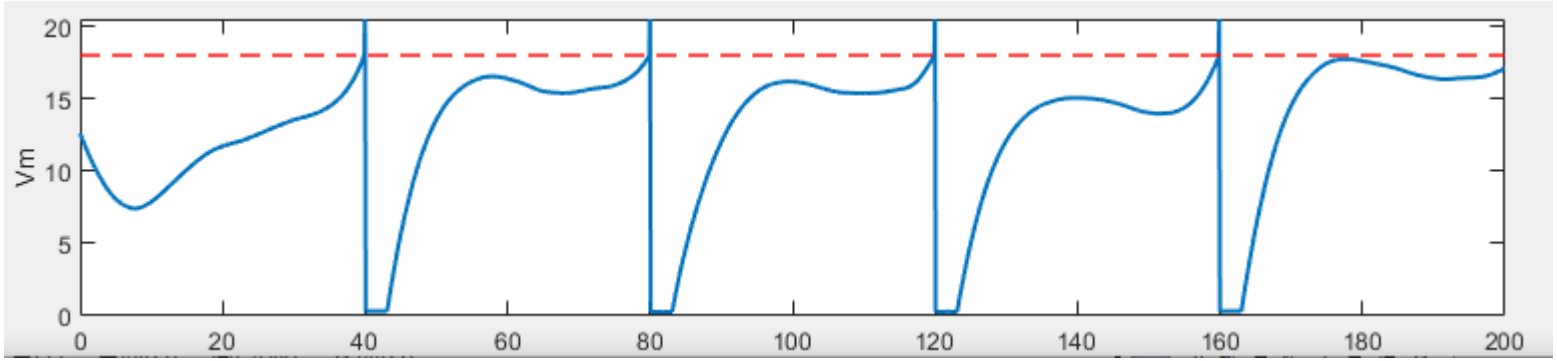
Precise-Spike-Driven (PSD) simulation in Matlab

The following figures show the dynamics of the neuron's potential before and after learning, respectively:

Before training



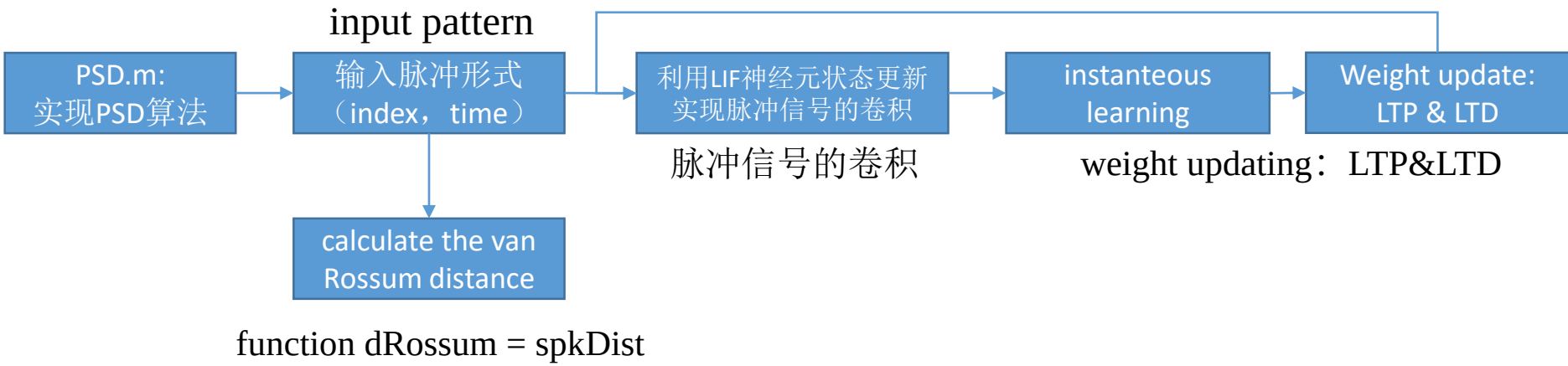
After training



Yu Q, Tang H, Tan K C, et al. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns[J]. Plos one, 2013, 8(11): e78318.

PSD code in matlab

代码结构示意图



function dRossum = spkDist

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

PSD code in matlab

Hyperparameter configuration

```
clear; close all;
nAfferents=1000; % Number of inputs
ptnTime=200;     % Pattern time (ms)
dt = 0.1;        % step time - resolution(ms)
```

```
%---- Neuron parameters ---%
tau_m = 10;      % (ms)
Rm = 1;          % (Mohm)
Vr = 0;          % rest potential (also as
the reset potential) (mV)
Vthr = 18;       % threshold (mV)
t_ref = 3;       % refectory time (ms)
wmin=0; wmax=6;  % synaptic weight range (nA)
tau1=tau_m; %%tau_s
tau2=tau1/4; %%tau_f
Sc1 = exp(-dt/tau1);
Sc2 = exp(-dt/tau2);
V0 = 1/max(exp(-(0:dt:5*tau1)/tau1)-exp(-
(0:dt:5*tau1)/tau2)); % normalization factor
eta = 0.01*wmax; % learning rate
%---- end Neuron parameters ---%
```

input pattern

```
% uniformly generated an input pattern
spkPtn=[round(1+rand(nAfferents,1)*(ptnTime-
2)) (1:nAfferents)'];
% plot the input pattern
plot(spkPtn(:,1),spkPtn(:,2),'.');
xlabel('Time (ms)'); ylabel('Afferent');
title('Input spatiotemporal pattern');
```

```
% number of spikes in the target train
kTgtSpks = 4;
% target spike train (ms)
targetPtn =
round((1:kTgtSpks)/(kTgtSpks+1)*ptnTime);
Wts= 0.5+0.2*randn(nAfferents,1);  % initial
weights
maxEpoch=50;
% initial kernel
K1 = zeros(size(Wts));
K2 = zeros(size(Wts));
Ins = 0;
Dist = zeros(1,maxEpoch);% initial van Rossum
distance
% monitor of the membrane potentials
beginVm=zeros(1, round(ptnTime/dt));
endVm = 0*beginVm;
```

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

PSD code in matlab

The training process

```
for iepoch=1:maxEpoch % training epoch
    firings = [];      % neuron's actual firing times
    %----- reset states --------
    Vm = 0.7*Vthr;
    K1 = 0* K1;
    K2 = 0* K2;
    PSC = K1-K2;   % post synaptic current
    %----- end reset states ----

    for t=dt:dt:ptnTime
        Dw = zeros(size(Wts)); % instanteous learning

        %--- caculate current states ---
        Isyn = Wts'*PSC;
        Vm = Vm + (dt/tau_m)*(Vr-Vm + Rm*(Ins+Isyn));
        SpkTimesIdx = abs(spkPtn(:,1)-t)<0.1*dt;
        SpkAfrnt = spkPtn(SpkTimesIdx,2);
        % current firing afferents
        if ~isempty(SpkAfrnt)
            K1(SpkAfrnt) = K1(SpkAfrnt) + V0;
            K2(SpkAfrnt) = K2(SpkAfrnt) + V0;
        end
        PSC = K1-K2;
        %--- end of caculate current states ---
```

脉冲信号的卷积

The weight updating process

```
if iepoch==1 % record the potentials
    beginVm(round(t/dt))=Vm;
elseif iepoch==maxEpoch
    endVm(round(t/dt))=Vm;
end

noRefFlag = isempty(firings)||t-firings(end)>t_ref;
if ~noRefFlag % in refractory period
    Vm = Vr;
end
% actual firing & instanteous update LTD
if Vm>Vthr
    if iepoch==1
        beginVm(round(t/dt))=1.2*Vm;
    elseif iepoch==maxEpoch
        endVm(round(t/dt))=1.2*Vm;
    end
    Vm = Vr;
    Dw =Dw - eta*PSC;     % LTD
     end
% desired spike time & LTP
 if ~isempty(find(abs(targetPtn-t)<0.1*dt, 1))
           Dw = Dw + eta*PSC; % LTP
       end
       Wts = Wts + Dw;  % instanteous learning
       Wts = max(wmin, min(wmax,Wts) );
       K1 = Sc1*K1;
       K2 = Sc2*K2;
   end
    % calculate the van Rossum distance
    Dist(iepoch) = spkDist(targetPtn,firings,ptnTime,10);
end
```

LTD

LTP

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

PSD code in matlab

Calculate the van Rossum distance

```
function dRossum=spkDist(tSpike1,tSpike2,tmax,tau1)
% spike distance: Euclidean distance of two filtered
spike trains with a kernel
    tau2=tau1/4;
    dt = tau1/100;
    tk = 0:dt:5*tau1;
    t = 0:dt:tmax;

    spiketrain1 = zeros(size(t));
    spiketrain2 = zeros(size(t));
    spiketrain1(fix(tSpike1/dt)) = 1;
    spiketrain2(fix(tSpike2/dt)) = 1;

    Kernel = exp(-tk/tau1)-exp(-tk/tau2);
    Kernel = Kernel/max(Kernel);
    cSpike1 = conv(Kernel,spiketrain1);
    cSpike2 = conv(Kernel,spiketrain2);

    dRossum =  dt/tau1 * sum((cSpike1-cSpike2).^2) ;
end
```

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

Precise-Spike-Driven (PSD)

Characteristics of PSD:

- Based on a Hebbian interpretation of the Widrow-Hoff rule and kernel function convolution
- For spike sequence generation and classification
- Spatio-temporal input and output spike patterns
- Batch or incremental learning

$$\Delta w_i = \eta x_i (y_d - y_o)$$

Widrow-Hoff rule

$$\frac{dw_i(t)}{dt} = \eta [s_d(t) - s_o(t)] I^i_{PSC}(t)$$

PSD learning algorithm

Yu Q, Tang H, Tan K C, et al. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns[J]. Plos one, 2013, 8(11): e78318.

SNN vs ANN

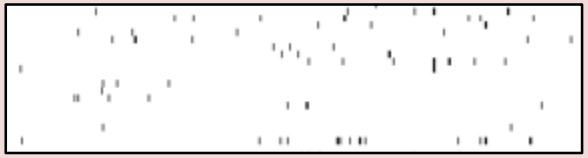
ANN | **SNN** | **Difficults**

Information Representation

Continuous Values

Temporal coding

Spiking Patterns



- **Precise and sparse in time**

Rate coding

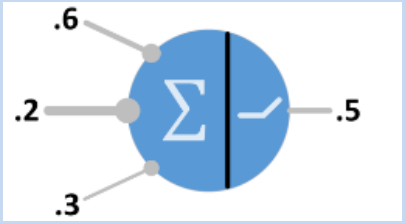
- **High information redundancy**

How to encode the information into the Spiking Patterns?

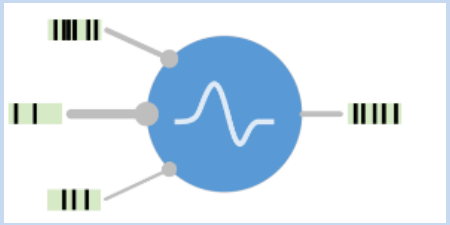
- **Avoid long time simulation**
- **Avoid losing information**

Neural Computation

Output values obtained by nonlinear function of the weighted inputs



Output spikes obtained by membrane temporal dynamics of the weighted inputs



- **Large number of parameters:**

Difficulties of spiking neurons

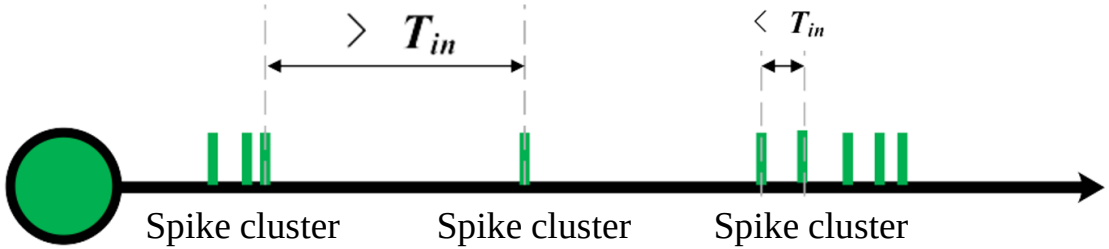
- **Complex temporal dynamics**
- **Sensitive to parameters values**

Backpropagation in SNN

- **STCA: Spatio-Temporal Credit Assignment with Delayed Feedback in Deep Spiking Neural Networks[1]**

A new spatio-temporal error backpropagation policy by defining a temporal based loss function, which is able to credit the network losses to spatial and temporal domains simultaneously.
We first present the new definition: **spike cluster**.

- **spike cluster**: If the intervals between any adjacent spikes in a spike train C are less than a predefined constant T_{in}, and the intervals between spikes in C and other outside spikes are greater than T_{in}, we define C as a spike cluster.



Utilizing this definition, we can set a **more flexible** objective for the output spiking neuron: when the predictive features are recognized, the neuron should issue one spike cluster instead of a fixed number of spikes. Hence, the goal of our loss function is to make **the actual number of spike clusters** equals to **the number of desired features.**

[1] Gu, Pengjie & Xiao, Rong & Pan, Gang & Tang, Huajin. STCA: Spatio-Temporal Credit Assignment with Delayed Feedback in Deep Spiking Neural Networks.
[2] Qiang Yu, Haizhou Li, and Kay C Tan. Spike timing or rate? neurons learn to make decisions for both through threshold-driven plasticity.

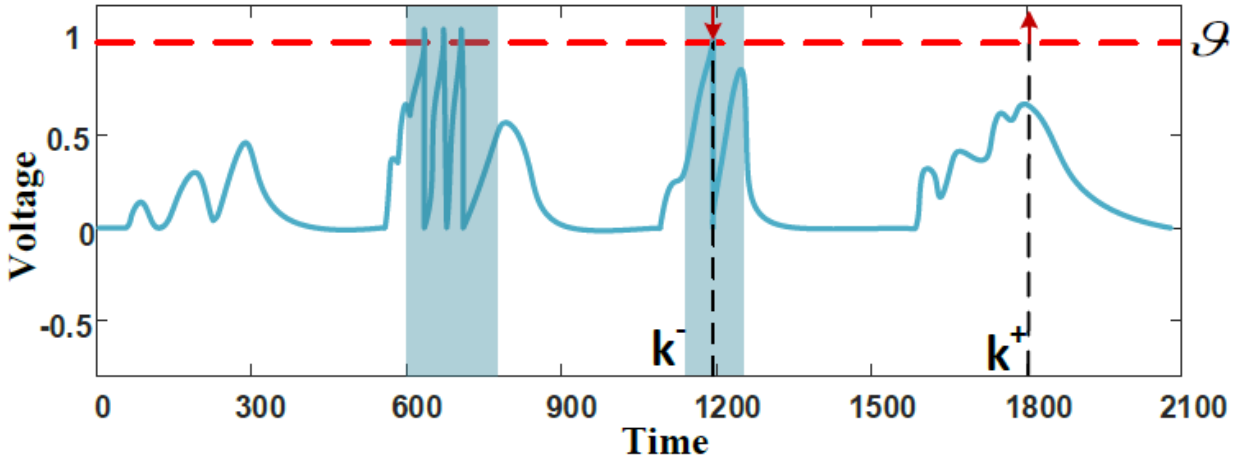
Backpropagation in SNN

- **STCA: Spatio-Temporal Credit Assignment with Delayed Feedback in Deep Spiking Neural Networks[1]**

Expecting that neuron will fire one or more spike cluster when the sample label is equal to 1 but keep silent otherwise, the loss function in classification problem is defined as:

$$L = \begin{cases} \vartheta - V^N[k^+], & N_o = 0 \; and \; y_s = 1 \\ V^N[k^-] - \vartheta, & N_o > 0 \; and \; y_s = 0 \\ 0, & otherwise \end{cases}$$

k+ and k- are both time point to assign temporal credit:



k+ is the time point where the voltage is the highest maximum of subthreshold voltage;
k- is the last spike time of the spike cluster which contains the fewest spikes.

[1] Gu, Pengjie & Xiao, Rong & Pan, Gang & Tang, Huajin. STCA: Spatio-Temporal Credit Assignment with Delayed Feedback in Deep Spiking Neural Networks.
[2] Qiang Yu, Haizhou Li, and Kay C Tan. Spike timing or rate? neurons learn to make decisions for both through threshold-driven plasticity.

Backpropagation in SNN

- **STCA: Spatio-Temporal Credit Assignment with Delayed Feedback in Deep Spiking Neural Networks[1]**

The spiking neuron model is C-LIF neuron model with event-driven property[2] to make error BP effective in both spatial and temporal domains.

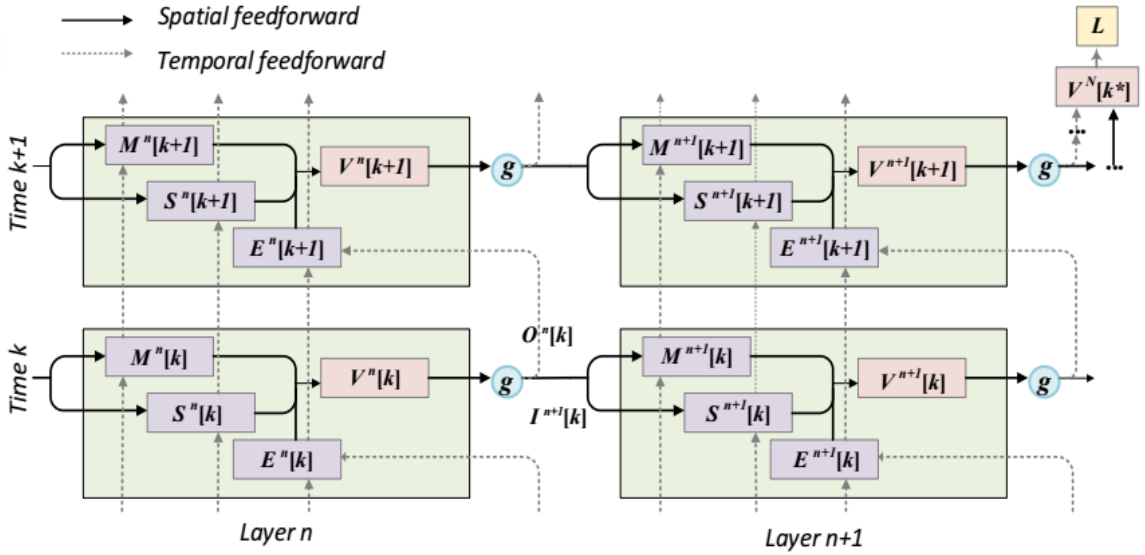


Figure : **The spatio-temporal feedforward of C-LIF neurons.** Each green box indicates a C-LIF neuron at a certain spatio-temporal state. The solid arrows and the dotted arrows indicate the directions of spatial feedforward and temporal feedforward, respectively. The error can be backpropagated to preceding layers and time points along the reverse directions of arrows.

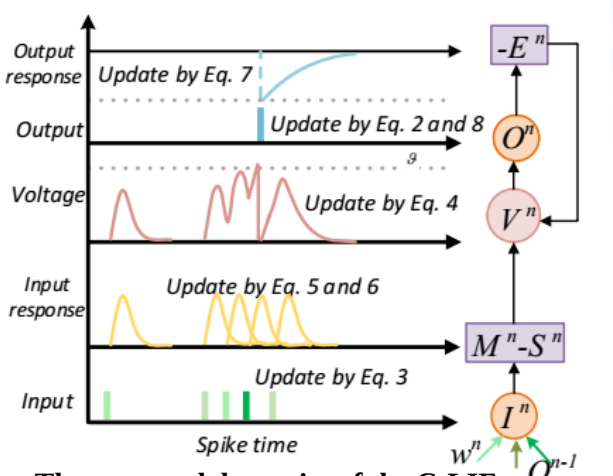


Figure: The temporal dynamics of the C-LIF neuron. (right) The relationships among the variables of the neuron. (left) The traces of these variables and their corresponding equations. The voltage $V^n$is the superposition of the input response (M^n- S^n) and the output response $-E^n$. With the voltage exceeding the threshold v, the neuron will fire an output spike and refrain its voltage by the output response. I^n is the weighted summation of O^{n-1} and will induce the input response.

[1] Gu, Pengjie & Xiao, Rong & Pan, Gang & Tang, Huajin. STCA: Spatio-Temporal Credit Assignment with Delayed Feedback in Deep Spiking Neural Networks.
[2] Qiang Yu, Haizhou Li, and Kay C Tan. Spike timing or rate? neurons learn to make decisions for both through threshold-driven plasticity.

Backpropagation in SNN

- **STCA: Spatio-Temporal Credit Assignment with Delayed Feedback in Deep Spiking Neural Networks[1]**

Based on **the temporal loss function** and the C-LIF model, we can backpropagte the error signal to preceding layers and previous time points accurately.

To handle the non-differentiable property, **the rectangular function** h(.) is adopted to be surrogate of the derivative of the spike function:

$$\frac{\partial g}{\partial V} \approx h(V) = \frac{1}{\alpha} sign(|V - \vartheta| < \frac{\alpha}{2})$$

Thus, we can compute the gradients as follow:

$$\frac{\partial L}{\partial I^n[k]} = \frac{\partial L}{\partial M^n[k]} \frac{\partial M^n[k]}{\partial I^n[k]} + \frac{\partial L}{\partial S^n[k]} \frac{\partial S^n[k]}{\partial S^n[k]} = \frac{\partial L}{\partial M^n[k]} + \frac{\partial L}{\partial S^n[k]}$$

$$\begin{aligned} \frac{\partial L}{\partial V^n[k]} &= \frac{\partial L}{\partial E^n[k+1]} \frac{\partial E^n[k+1]}{\partial O^n[k]} \frac{\partial O^n[k]}{\partial V^n[k]} + \frac{\partial L}{\partial I^{n+1}[k]} \frac{\partial I^n[k+1]}{\partial O^n[k]} \frac{\partial O^n[k]}{\partial V^n[k]} \\ &= (\frac{\partial L}{\partial E^n[k+1]} \vartheta + \frac{\partial L}{\partial I^{n+1}[k]} w^{n+1}) h(V^n[k]) \end{aligned}$$

Note that $\frac{\partial L}{\partial I^n[k]}$ are the gradients of the input signals across all preceding layers and previous time points.

Further, the gradients of synaptic weights can be computed as:

$$\nabla w^n = \sum_{k=1}^{k^*} \delta^n[k] \frac{\partial I^n[k]}{\partial w^n} = V_0 \sum_{k=1}^{k^*} \delta^n[k] O^{n-1}[k]$$


ArtYouhua


Thank You!