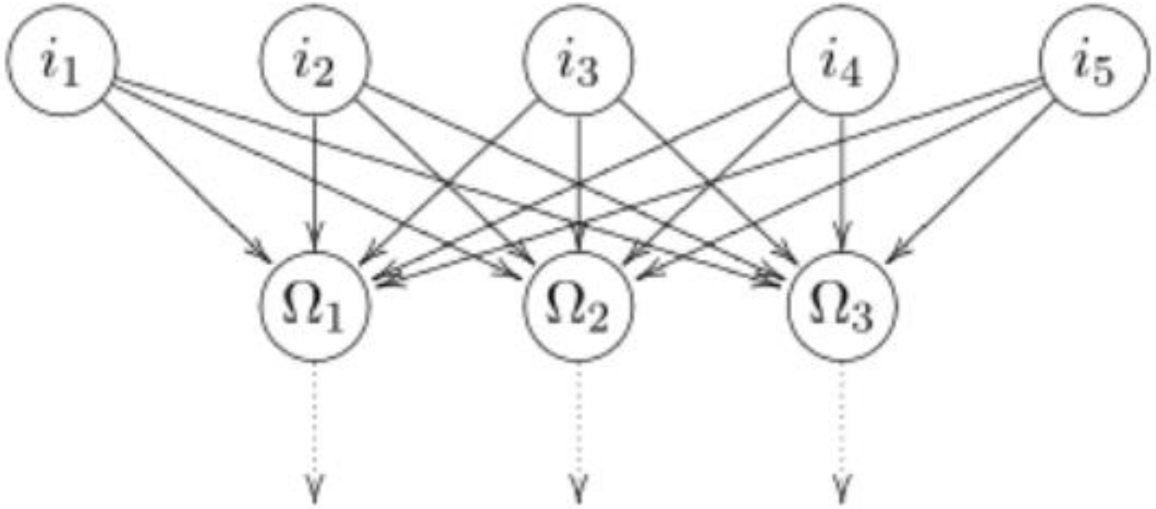
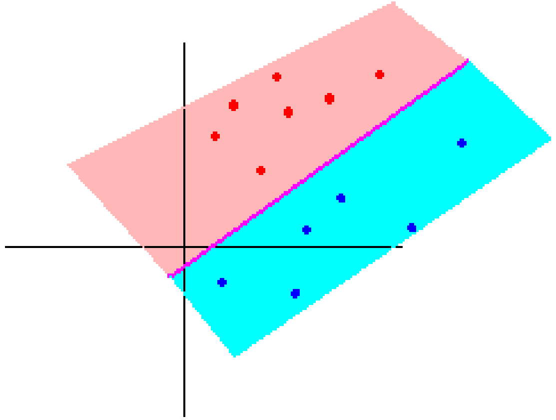
Neural network problems

Outline

1. Single-layer perceptron
   a) Using perceptron learning algorithm
   b) Using delta rule
2. Single-layer perceptron with multiple outputs

1. Single-layer perceptron





Linear separation

1. Single-layer perceptron

- Before using SLP, make sure the data is linearly separable
  - Visualize the data (not possible for more than 2 features)

1. Single-layer perceptron

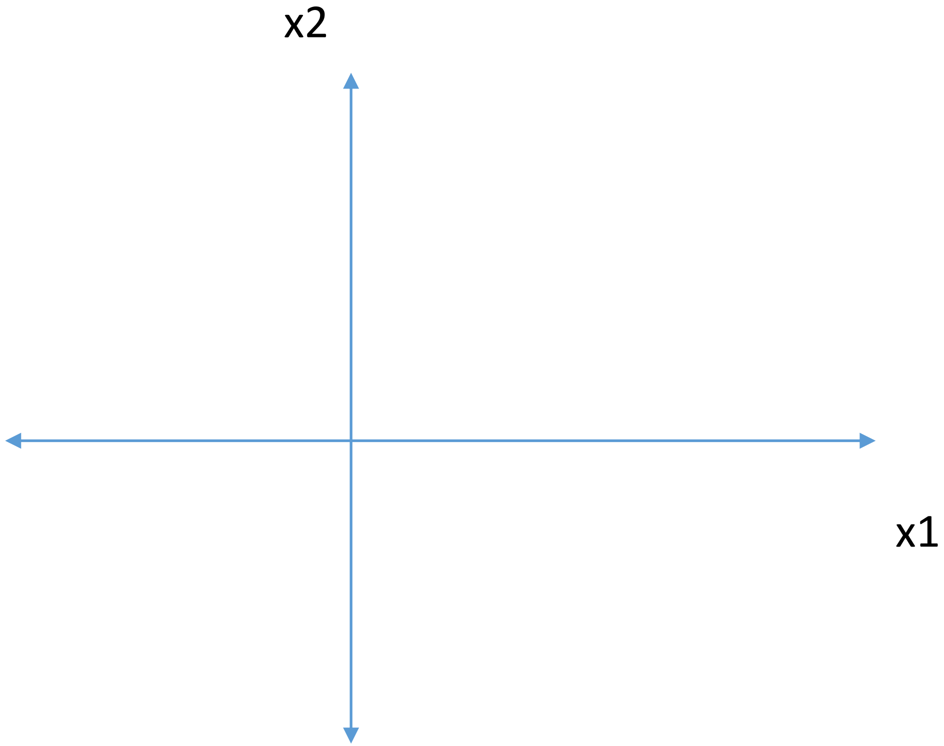
- Visualization example (2 features)

x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1

1. Single-layer perceptron

- Visualization example (2 features)

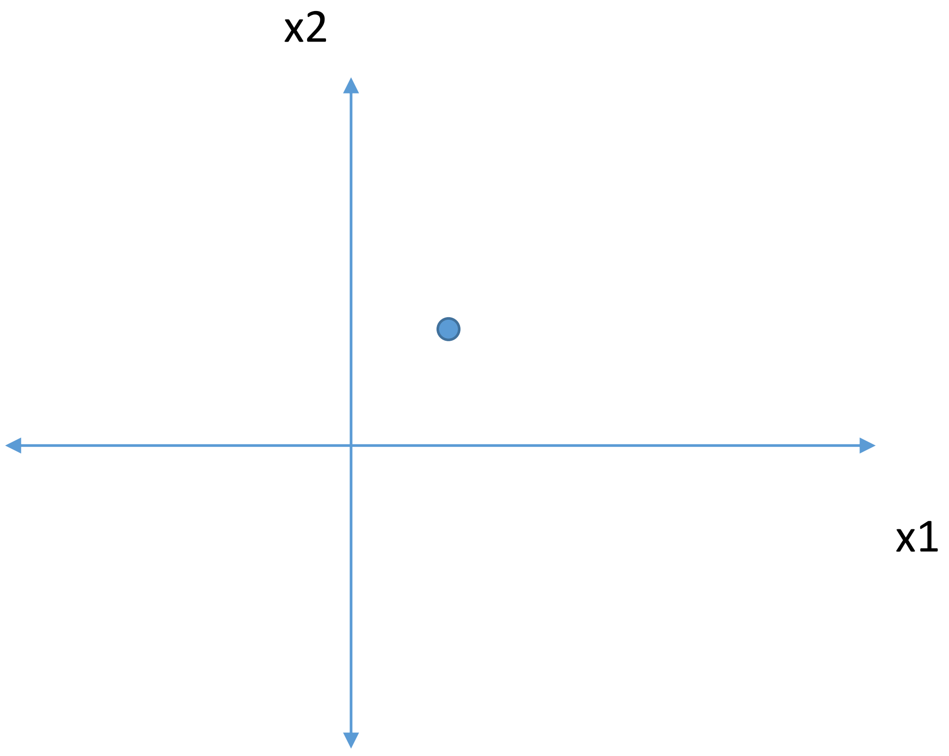
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

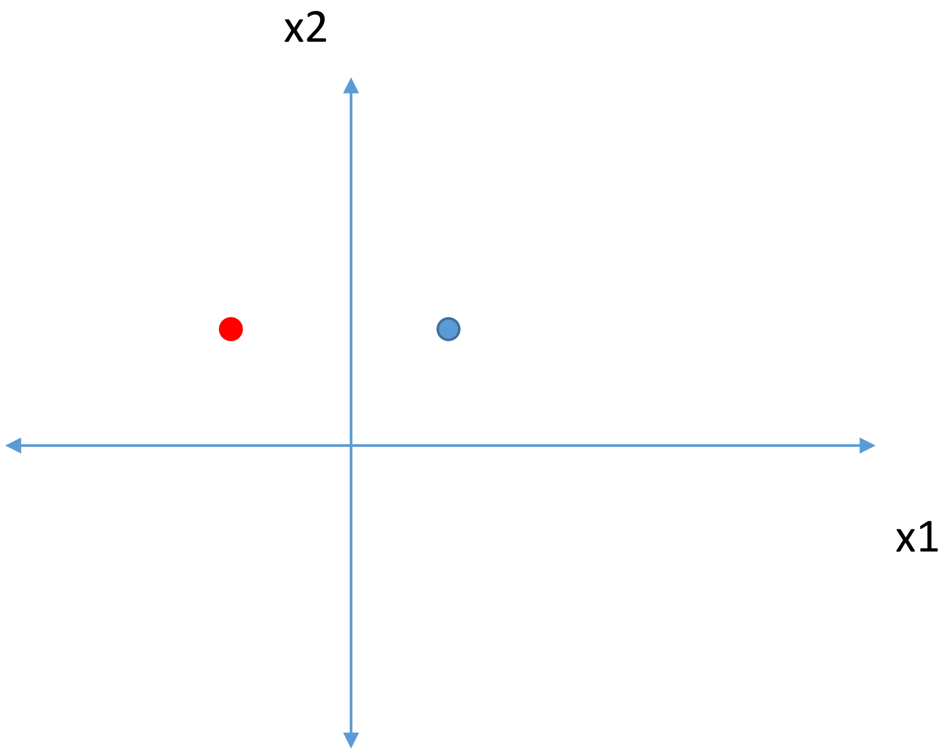
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

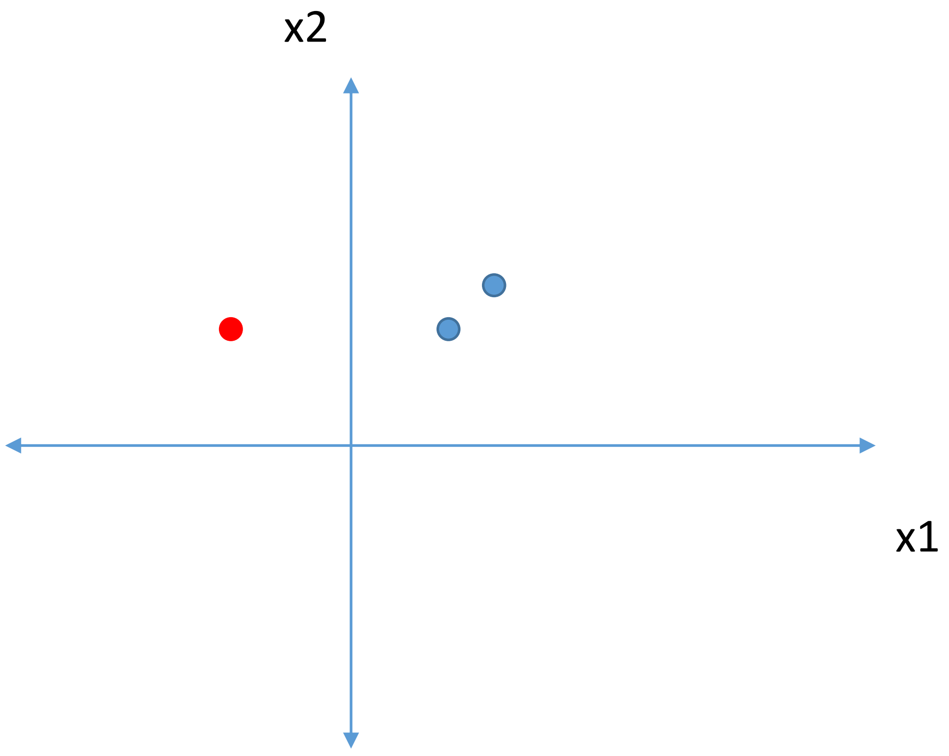
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

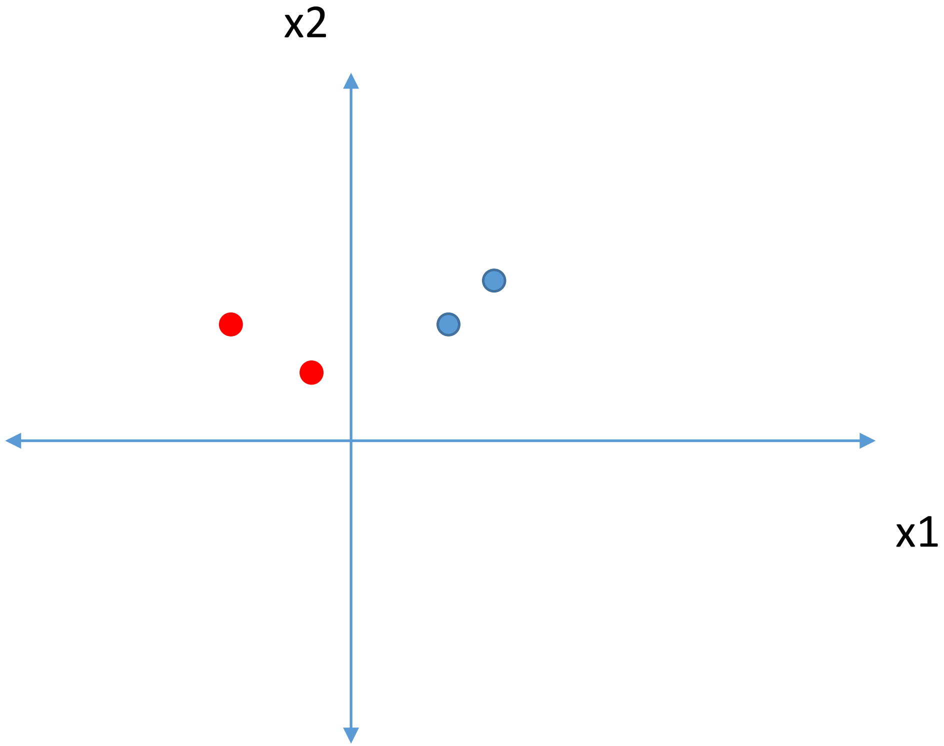
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

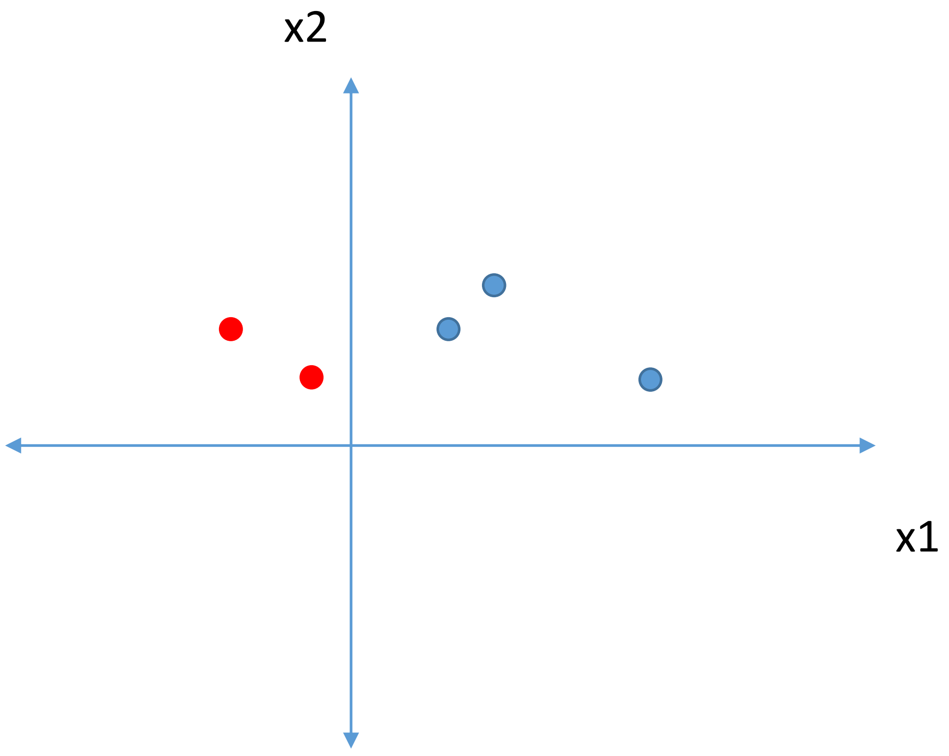
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

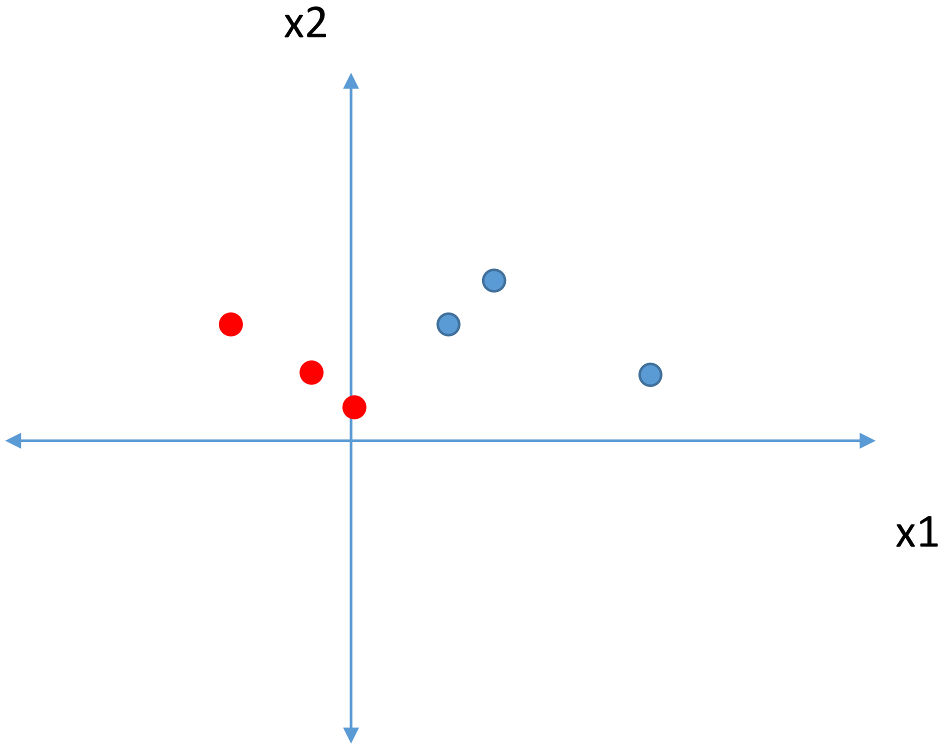
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

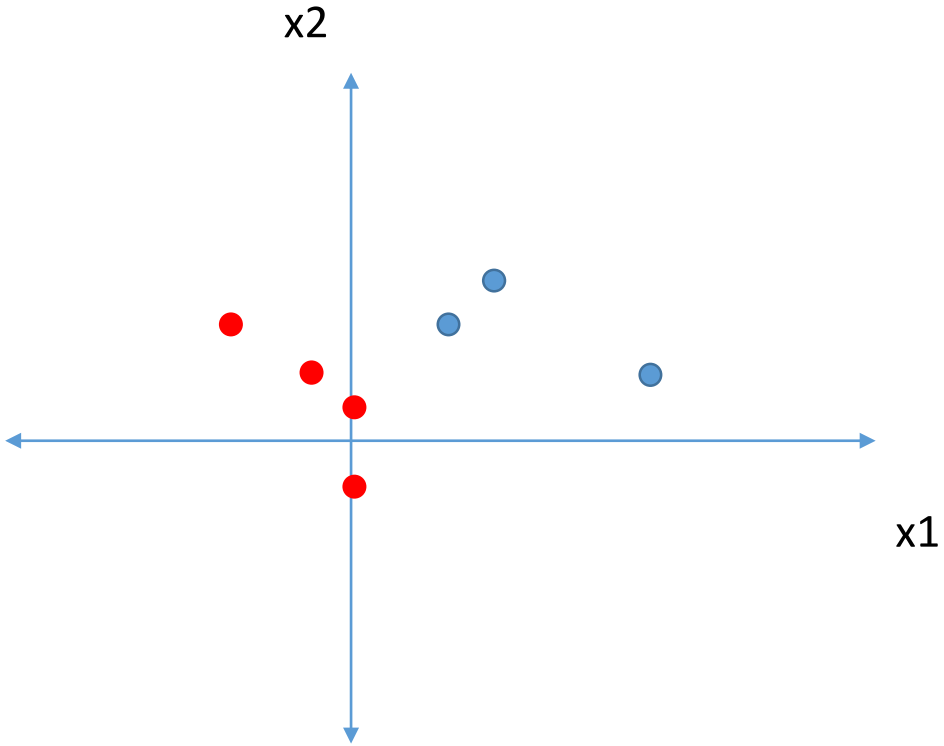
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

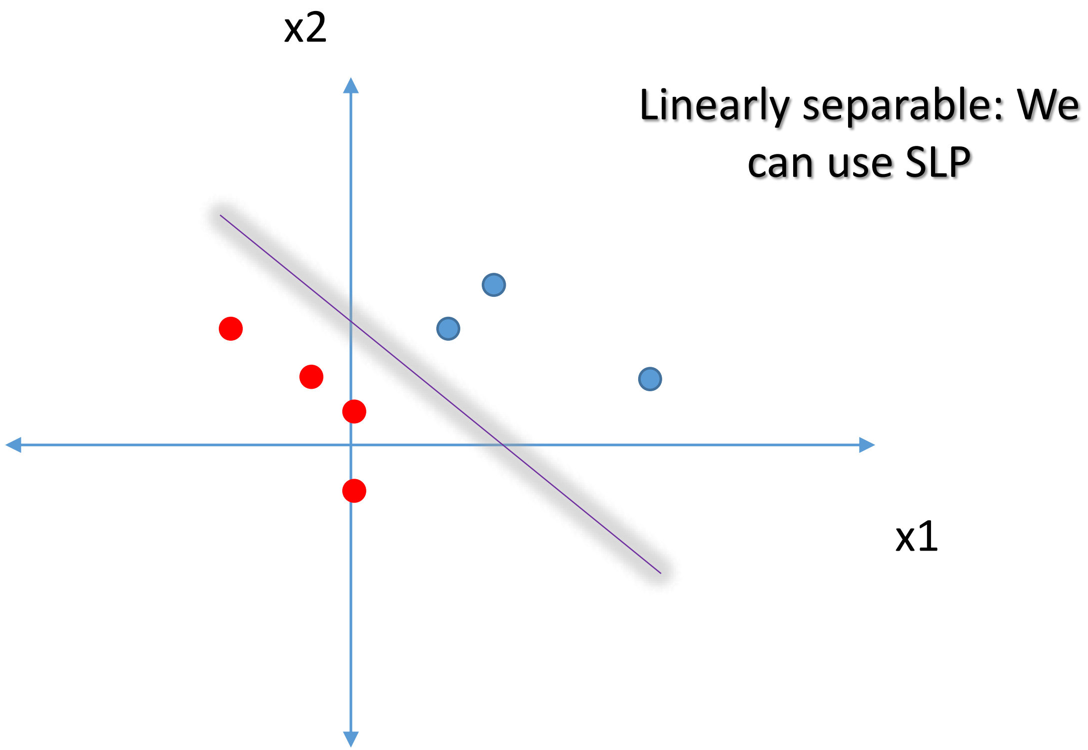
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

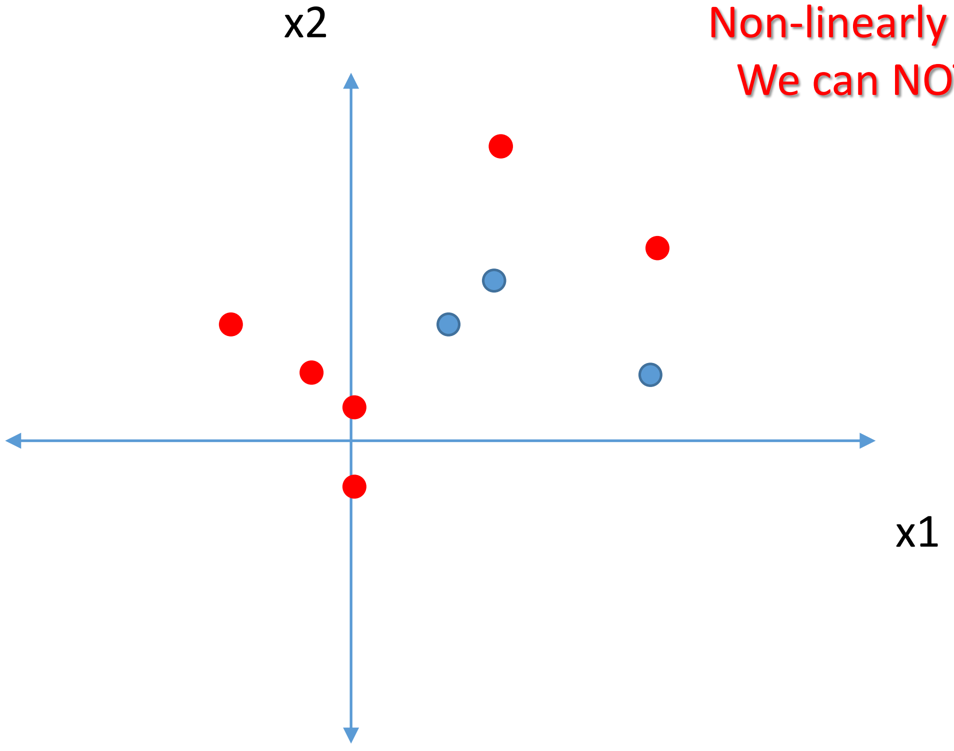
x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1



1. Single-layer perceptron

- Visualization example (2 features)

x1	x2	t
2	3	0
-3	3	1
3	4	0
-1	2	1
7	2	0
0	1	1
0	-2	1
3	8	1
7	5	1



Non-linearly separable:
We can NOT use SLP

1. Single-layer perceptron

- Visualization example (1 feature)

x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1

1. Single-layer perceptron

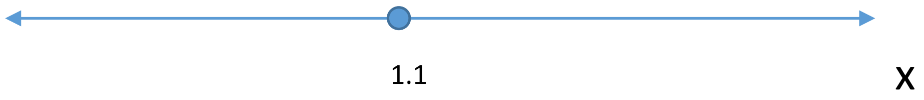
- Visualization example (1 feature)

x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1

1. Single-layer perceptron

- Visualization example (1 feature)

x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1



1. Single-layer perceptron

- Visualization example (1 feature)

x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1

1. Single-layer perceptron

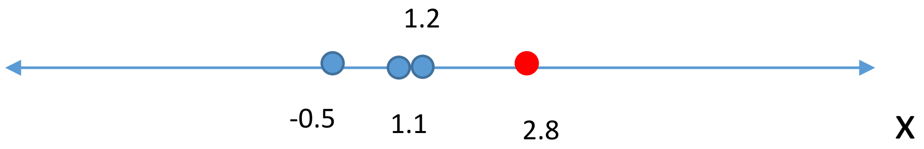
- Visualization example (1 feature)

x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1

1. Single-layer perceptron

- Visualization example (1 feature)

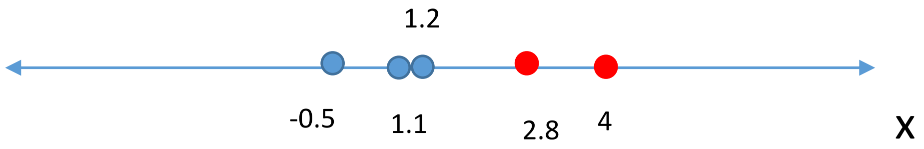
x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1



1. Single-layer perceptron

- Visualization example (1 feature)

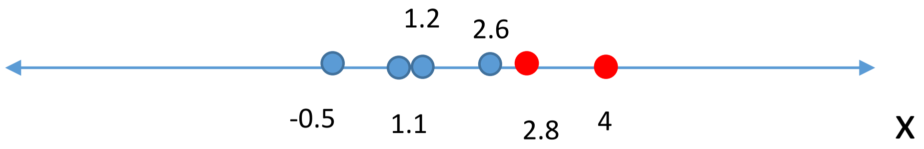
x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1



1. Single-layer perceptron

- Visualization example (1 feature)

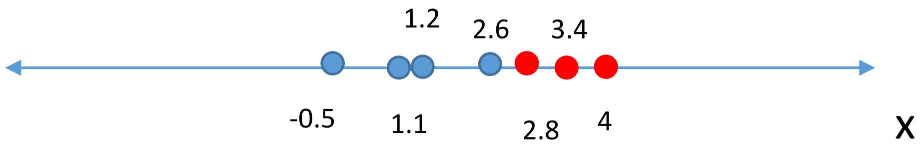
x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1



1. Single-layer perceptron

- Visualization example (1 feature)

x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1

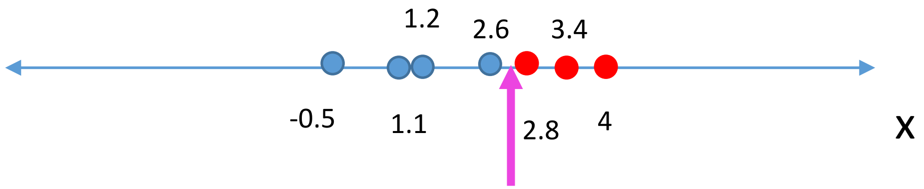


1. Single-layer perceptron

- Visualization example (1 feature)

x	t
1.1	0
2.8	1
-0.5	0
1.2	0
4	1
2.6	0
3.4	1

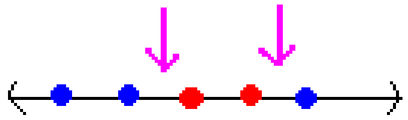
Note: in 1d, SLP is a point separator



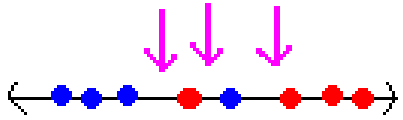
We can separate the two classes with one point: We can use SLP

1. Single-layer perceptron

- Visualization example (1 feature)



Need at least 2 points:
Can't use SLP



Need at least 3 points:
Can't use SLP

1. a) Perceptron learning algorithm

```
while ∃p ∈ P and error too large do
  Input p into the network, calculate output y {P set of training patterns}
  for all output neurons Ω do
    if y_Ω = t_Ω then
      Output is okay, no correction of weights
    else
      if y_Ω = 0 then
        for all input neurons i do
          w_{i,Ω} := w_{i,Ω} + o_i {...increase weight towards Ω by o_i}
        end for
      end if
      if y_Ω = 1 then
        for all input neurons i do
          w_{i,Ω} := w_{i,Ω} − o_i {...decrease weight towards Ω by o_i}
        end for
      end if
    end if
  end for
end while
```

1. a) Perceptron learning algorithm

```
while ∃p ∈ P and error too large do
  Input p into the network, calculate output y {P set of training patterns}
  for all output neurons Ω do
    if y_Ω = t_Ω then
      Output is okay, no correction of weights
    else
      if y_Ω = 0 then
        for all input neurons i do
          w_{i,Ω} := w_{i,Ω} + o_i {...increase weight towards Ω by o_i}
        end for
      end if
      if y_Ω = 1 then
        for all input neurons i do
          w_{i,Ω} := w_{i,Ω} − o_i {...decrease weight towards Ω by o_i}
        end for
      end if
    end if
  end for
end while
```

1. a) Perceptron learning algorithm

```
while ∃p ∈ P and error too large do
  Input p into the network, calculate output y {P set of training patterns}
  for all output neurons Ω do
    if y_Ω = t_Ω then
      Output is okay, no correction of weights
    else
      if y_Ω = 0 then
        for all input neurons i do
          w_{i,Ω} := w_{i,Ω} + o_i {...increase weight towards Ω by o_i}
        end for
      end if
      if y_Ω = 1 then
        for all input neurons i do
          w_{i,Ω} := w_{i,Ω} − o_i {...decrease weight towards Ω by o_i}
        end for
      end if
    end if
  end for
end while
```

1. a) Perceptron learning algorithm

$$w_{i,\Omega} := w_{i,\Omega} + o_i$$

$$w_{i,\Omega} := w_{i,\Omega} - o_i$$

1. a) Perceptron learning algorithm

$$w_{i,\Omega} := w_{i,\Omega} + o_i$$

$$w_{i,\Omega} := w_{i,\Omega} - o_i$$

Step size depends on Oi.

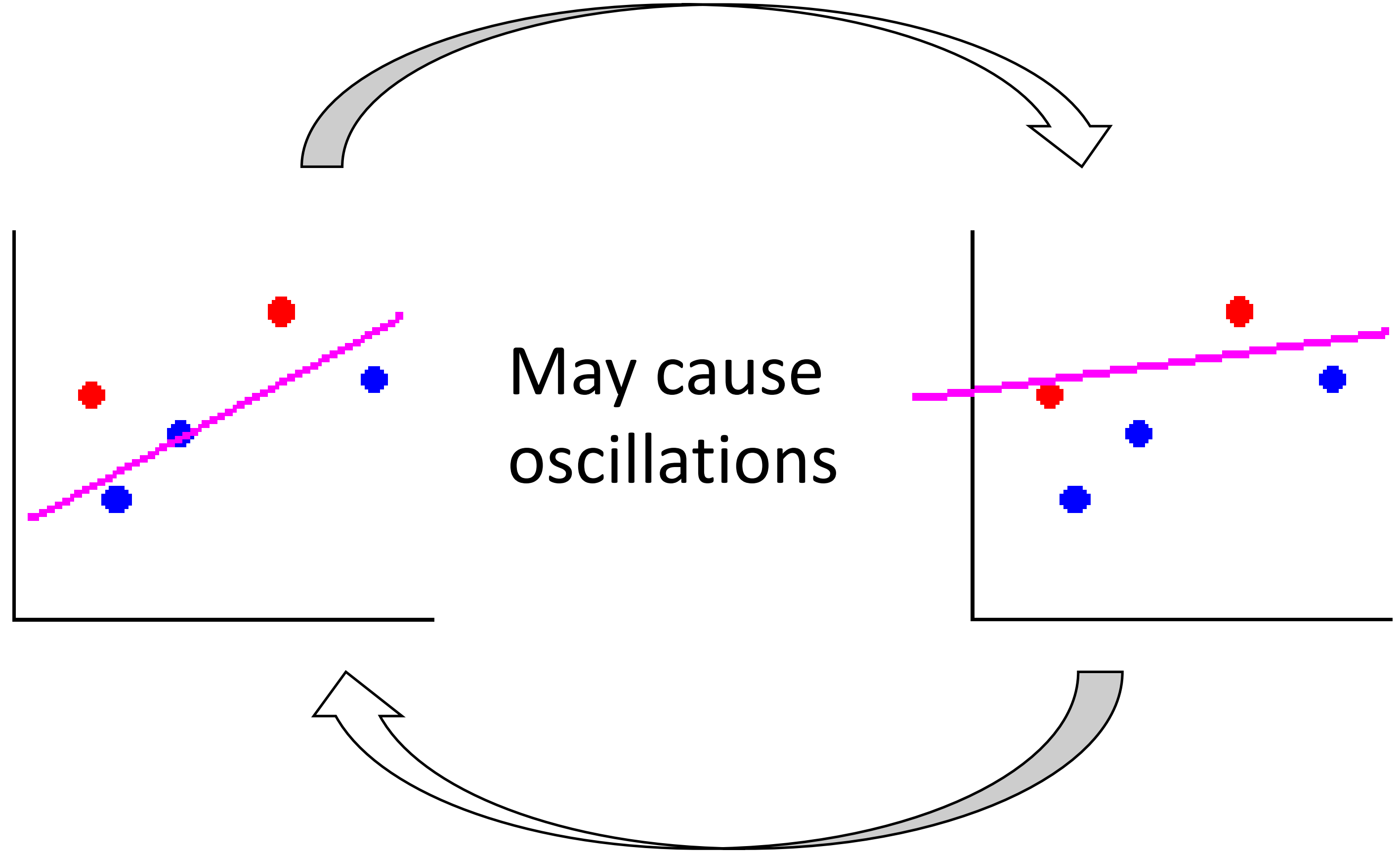
Not controlled because Oi can be large.

1. a) Perceptron learning algorithm

while $\exists p \in P$ **and** error too large **do**
  Input p into the network, calculate output y $\{P$ set of training patterns$\}$
  for all output neurons Ω **do**
    if $y_\Omega = t_\Omega$ **then**
      Output is okay, no correction of weights
    else
      if $y_\Omega = 0$ **then**
        for all input neurons i **do**
          $w_{i,\Omega} := w_{i,\Omega} + o_i$ $\{$...increase weight towards Ω by $o_i\}$
        end for
      end if
      if $y_\Omega = 1$ **then**
        for all input neurons i **do**
          $w_{i,\Omega} := w_{i,\Omega} - o_i$ $\{$...decrease weight towards Ω by $o_i\}$
        end for
      end if
    end if
  end for
end while

It assumes the output is either 0 or 1

1. a) Perceptron learning algorithm



1. a) Perceptron learning algorithm example

x1	x2	t
0	0	0
0	1	1
1	0	1
1	1	1

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0							0
0	1							1
1	0							1
1	1							1

Add new columns

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1						0
0	1	1						1
1	0	1						1
1	1	1						1

Bias node always
produces 1

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2			0
0	1	1						1
1	0	1						1
1	1	1						1

Put initial weights (given)

If not given: assume random weights
(but not 0)

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2		0
0	1	1						1
1	0	1						1
1	1	1						1

Calculate net = x1*w1 + x2*w2 + bias*w_bias

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1						1
1	0	1						1
1	1	1						1

Calculate y =

1 if net >= threshold,

0 if net < threshold

Threshold should be given. If not, assume random threshold

Here we assume threshold = 0.1 → net < threshold

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1						1
1	0	1						1
1	1	1						1

y = t ? yes

Weights will not be changed

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2			1
1	0	1						1
1	1	1						1

Use the same weights for next pattern

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0		1
1	0	1						1
1	1	1						1

Calculate net = x1*w1 + x2*w2 + bias*w_bias

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1						1
1	1	1						1

net < 0.1 → y = 0

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1						1
1	1	1						1

y != t

We need to change weights

y = 0   we want y = 1   increase weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8			1
1	1	1						1

w1 := w1 + x1
w2 := w1 + x1
w_bias := w_bias + bias

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8	0.9		1
1	1	1						1

Calculate net = x1*w1 + x2*w2 + bias*w_bias

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8	0.9	1	1
1	1	1						1

net >= 0.1 → y = 1

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8	0.9	1	1
1	1	1	0.1	1.2	0.8			1

y = t

Don't change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8	0.9	1	1
1	1	1	0.1	1.2	0.8	2.1		1

Calculate net = x1*w1 + x2*w2 + bias*w_bias

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8	0.9	1	1
1	1	1	0.1	1.2	0.8	2.1	1	1

net >= 0.1 → y = 1

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8	0.9	1	1
1	1	1	0.1	1.2	0.8	2.1	1	1
Weights for next epoch			0.1	1.2	0.8			

y = t

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8	0.9	1	1
1	1	1	0.1	1.2	0.8	2.1	1	1
Weights for next epoch			0.1	1.2	0.8			

1 Epoch complete:

But we still have 1 error

We need to run another epoch

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	1.2	0.8	0.9	1	1
1	1	1	0.1	1.2	0.8	2.1	1	1
Weights for next epoch			0.1	1.2	0.8			

Use these as initial weights for next epoch

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8			0
0	1	1						1
1	0	1						1
1	1	1						1

New epoch with initial weights from previous slide

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1						1
1	0	1						1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1	0.1	1.2	-0.2			1
1	0	1						1
1	1	1						1

y != t   y = 1 and we want y = 0

Decrease weights:

w1   := w1 - x1
w2   := w1 - x1
w_bias := w_bias - bias

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1	0.1	1.2	-0.2	1	1	1
1	0	1						1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1	0.1	1.2	-0.2	1	1	1
1	0	1	0.1	1.2	-0.2			1
1	1	1						1

y = t

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1	0.1	1.2	-0.2	1	1	1
1	0	1	0.1	1.2	-0.2	-0.1	0	1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1	0.1	1.2	-0.2	1	1	1
1	0	1	0.1	1.2	-0.2	-0.1	0	1
1	1	1	1.1	1.2	0.8			1

y = 0 and we want y = 1

Increase weights:

w1 := w1 + x1
w2 := w1 +x1
w_bias := w_bias + bias

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1	0.1	1.2	-0.2	1	1	1
1	0	1	0.1	1.2	-0.2	-0.1	0	1
1	1	1	1.1	1.2	0.8	3.1	1	1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1	0.1	1.2	-0.2	1	1	1
1	0	1	0.1	1.2	-0.2	-0.1	0	1
1	1	1	1.1	1.2	0.8	3.1	1	1
Weights for next epoch			1.1	1.2	0.8			

y = t

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	0.1	1.2	0.8	0.8	1	0
0	1	1	0.1	1.2	-0.2	1	1	1
1	0	1	0.1	1.2	-0.2	-0.1	0	1
1	1	1	1.1	1.2	0.8	3.1	1	1
Weights for next epoch			1.1	1.2	0.8			

Second epoch done

We still have 2 errors

We need to run another epoch

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8			0
0	1	1						1
1	0	1						1
1	1	1						1

Starting third epoch

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1						1
1	0	1						1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1	1.1	1.2	-0.2			1
1	0	1						1
1	1	1						1

Decrease weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1						1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2			1
1	1	1						1

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1	1.1	1.2	-0.2			1

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1	1.1	1.2	-0.2	2.1	1	1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1	1.1	1.2	-0.2	2.1	1	1
Weights for next epoch			1.1	1.2	-0.2			

Don't change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	0.8	0.8	1	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1	1.1	1.2	-0.2	2.1	1	1
Weights for next epoch			1.1	1.2	-0.2			

We still have one error

We need to run another epoch

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2			0
0	1	1						1
1	0	1						1
1	1	1						1

Fourth epoch

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1						1
1	0	1						1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1	1.1	1.2	-0.2			1
1	0	1						1
1	1	1						1

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1						1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2			1
1	1	1						1

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1						1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1	1.1	1.2	-0.2			1

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1	1.1	1.2	-0.2	2.1	1	1

Calculate net and y

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1	1.1	1.2	-0.2	2.1	1	1
Weights for next epoch			1.1	1.2	-0.2			

Don’t change weights

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	**0**	**0**
0	1	1	1.1	1.2	-0.2	1	**1**	**1**
1	0	1	1.1	1.2	-0.2	0.9	**1**	**1**
1	1	1	1.1	1.2	-0.2	2.1	**1**	**1**
Weights for next epoch			1.1	1.2	-0.2			

Fourth epoch done

No errors → Stop training

1. a) Perceptron learning algorithm

x1	x2	bias	w1	w2	w_bias	net	y	t
0	0	1	1.1	1.2	-0.2	-0.2	0	0
0	1	1	1.1	1.2	-0.2	1	1	1
1	0	1	1.1	1.2	-0.2	0.9	1	1
1	1	1	1.1	1.2	-0.2	2.1	1	1
Weights for next epoch			1.1	1.2	-0.2			

Final weights

1. b) SLP using delta rule

- Same as the previous example. Just updating weights is different

$$w_{i,\Omega} \coloneqq w_{i,\Omega} + \eta \, o_i \, (t_\Omega - y_\Omega)$$

- For previous example:
  - w1 := w1 + η * x1 * (t – y)
  - w2 := w2 + η * x2 * (t – y)
  - w_bias := w_bias + η * bias * (t – y)

1. b) SLP using delta rule

- Same as the previous example. Just updating weights is different

$$w_{i,\Omega} := w_{i,\Omega} + \eta\, o_i\, (t_\Omega - y_\Omega)$$

- For previous example:
    - w1 := w1 + η * x1 * (t – y)
    - w2 := w2 + η * x2 * (t – y)
    - w_bias := w_bias + η * bias * (t – y)

The term “bias” always equals 1 (can be omitted)

1. b) SLP using delta rule

- Same as the previous example. Just updating weights is different

$$w_{i,\Omega} := w_{i,\Omega} + \eta\, o_i\, (t_\Omega - y_\Omega)$$

- For previous example:
  - w1 := w1 + η * x1 * (t – y)
  - w2 := w2 + η * x2 * (t – y)
  - w_bias := w_bias + η * bias * (t – y)

This is the learning rate (a given constant). If not given, assume a value between 0.01 and 0.9

1. b) SLP using delta rule

- Same as the previous example. Just updating weights is different

$$w_{i,\Omega} := w_{i,\Omega} + \eta \, o_i \, (t_\Omega - y_\Omega)$$

- For previous example:
  - w1 := w1 + η * x1 * (t – y)
  - w2 := w2 + η * x2 * (t – y)
  - w_bias := w_bias + η * bias * (t – y)

We always add
(even if y > t)

But how do we
decrease weights?

1. b) SLP using delta rule

- Same as the previous example. Just updating weights is different

$$w_{i,\Omega} := w_{i,\Omega} + \eta\, o_i\, (t_\Omega - y_\Omega)$$

- For previous example:
  - w1 := w1 + η * x1 * (t – y)
  - w2 := w2 + η * x2 * (t – y)
  - w_bias := w_bias + η * bias * (t – y)

If y > t, this term will be negative, causing weights to be decreased

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2			0
0	1	1						1
1	0	1						1
1	1	1						1

Same example using delta rule

Assume learning rate = 0.1

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1						1
1	0	1						1
1	1	1						1

Calculating net and y is not different

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2			1
1	0	1						1
1	1	1						1

If we try to update weights: (even though y = t)

w1 := w1 + 0.1 * x1 * (t - y)
w2 := w2 + 0.1 * x2 * (t - y)
wb := wb + 0.1 * bias * (t - y)

(t - y) = 0 so the weights will not be changed

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1						1
1	1	1						1

Calculate y and net

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	0.3	-0.1			1
1	1	1						1

update weights:

w1 := w1 + 0.1 * x1 * (t - y) → w1 := 0.1 + 0.1 * 0 * 1 → 0.1
w2 := w2 + 0.1 * x2 * (t - y) → w2 := 0.2 + 0.1 * 1 * 1 → 0.3
wb := wb + 0.1 * bias * (t - y) → wb := -0.2 + 0.1 * 1 * 1 → -0.1

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	0.3	-0.1	0	0	1
1	1	1						1

Calculate net and y

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	0.3	-0.1	0	0	1
1	1	1	0.2	0.3	0			1

update weights:

w1 := w1 + 0.1 * x1 * (t - y) → 0.1 + 0.1 * 1 * 1 → 0.2
w2 := w2 + 0.1 * x2 * (t - y) → 0.3 + 0.1 * 0 * 1 → 0.3
wb := wb + 0.1 * bias * (t - y) → -0.1 + 0.1 * 1 * 1 → 0

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	0.3	-0.1	0	0	1
1	1	1	0.2	0.3	0	0.5	1	1

Calculate net and y

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	0.3	-0.1	0	0	1
1	1	1	0.2	0.3	0	0.5	1	1
Weights for next epoch:			0.2	0.3	0			

Weights will not be changed

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.1	0.2	-0.2	-0.2	0	0
0	1	1	0.1	0.2	-0.2	0	0	1
1	0	1	0.1	0.3	-0.1	0	0	1
1	1	1	0.2	0.3	0	0.5	1	1
Weights for next epoch:			0.2	0.3	0			

First epoch done

We have 2 errors

We need to run another epoch

1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.2	0.3	0			0
0	1	1						1
1	0	1						1
1	1	1						1

Second epoch

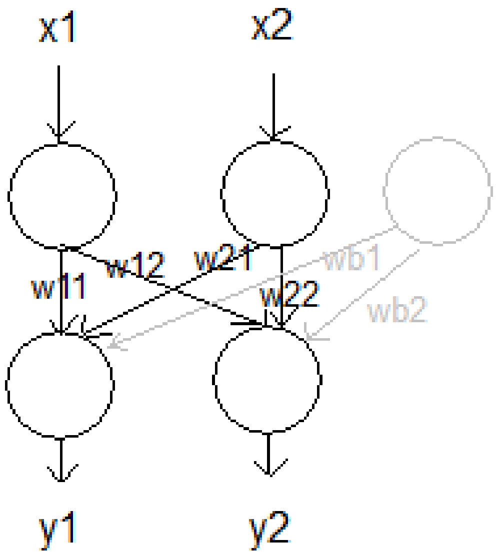
1. b) SLP using delta rule

x1	x2	bias	w1	w2	wb	net	y	t
0	0	1	0.2	0.3	0	0	**0**	**0**
0	1	1	0.2	0.3	0	0.3	**1**	**1**
1	0	1	0.2	0.3	0	0.2	**1**	**1**
1	1	1	0.2	0.3	0	0.5	**1**	**1**
Weights for next epoch:			0.2	0.3	0			

Second epoch has no errors → stop training

2. SLP with multiple outputs

x1	x2	t1	t2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1



The two output neurons can have different threshold values

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0									0	0
0	1									1	0
1	0									1	0
1	1									1	1

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1			0	0
0	1									1	0
1	0									1	0
1	1									1	1

Initial weights

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0		0	0
0	1									1	0
1	0									1	0
1	1									1	1

net1 = w11 * x1 + w21 * x2 + b1
net1 = 0.1 * 0 + 0.2 * 0 - 0.2 = -0.2

Assume threshold1 = 0.1,
threshold2 = 1

net1 >= 0.1 ? → y = 1
else? → y = 0

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1									1	0
1	0									1	0
1	1									1	1

net2 = w12 * x1 + w22 * x2 + b2
net2 = 0.2 * 0 + 0.3 * 0 + 1 = 1

net2 >= 1 ? → y = 1
else? → y = 0

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2						1	0
1	0									1	0
1	1									1	1

y1 is OK

don’t change its weights

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2						1	0
1	0									1	0
1	1									1	1

y2 is wrong

update its weights:
(We can either use perceptron learning algorithm or delta rule)

Assume we are using delta rule, $\eta = 0.1$

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9			1	0
1	0									1	0
1	1									1	1

w12 := w12 + 0.1 * x1 * (t2 – y2) → 0.2 + 0.1 * 0 * -1 → 0.2
w22 := w22 + 0.1 * x2 * (t2 – y2) → 0.3 + 0.1 * 0 * -1 → 0.3
wb2 := wb2 + 0.1 * (t2 – y2) → 1 + 0.1 * -1 → 0.9

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0		1	0
1	0									1	0
1	1									1	1

calculate net1, y1

net1 = 0.1 * 0 + 0.2 * 1 – 0.2 = 0

net1 < 0.1 → y1 = 0

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0									1	0
1	1									1	1

calculate net2, y2

net2 = 0.2 * 0 + 0.3 * 1 + 0.9 = 1.2   net2 >= 1 → y2 = 1

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1						1	0
1	1									1	1

Update weights of y1

w11 = 0.1 + 0.1 * 0 * (1 - 0) = 0.1
w21 = 0.2 + 0.1 * 1 * (1 – 0) = 0.3
wb1 = -0.2 + 0.1 * (1 – 0) = -0.1

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8			1	0
1	1									1	1

Update weights of y2

w12 = 0.2 + 0.1 * 0 * (0 - 1) = 0.2
w22 = 0.3 + 0.1 * 1 * (0 – 1) = 0.2
wb2 = 0.9 + 0.1 * (0 – 1) = 0.8

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8	0		1	0
1	1									1	1

Calculate net1 and y1

net1 = 0.1 * 1 + 0.3 * 0 – 0.1 = 0 → y1 = 0

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8	0	1	1	0
1	1									1	1

Calculate net2 and y2

net2 = 0.2 * 1 + 0.2 * 0 + 0.8 = 1 → y2 = 1

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8	0	1	1	0
1	1	0.2	0.3	0						1	1

Update weights of y1

w11 = 0.1 + 0.1 * 1 * (1 - 0) = 0.2
w21 = 0.3 + 0.1 * 0 * (1 – 0) = 0.3
wb1 = -0.1 + 0.1 * (1 – 0) = 0

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8	0	1	1	0
1	1	0.2	0.3	0	0.1	0.2	0.7			1	1

Update weights of y2

w12 = 0.2 + 0.1 * 1 * (0 - 1) = 0.1
w22 = 0.2 + 0.1 * 0 * (0 - 1) = 0.2
wb2 = 0.8 + 0.1 * (0 - 1) = 0.7

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8	0	1	1	0
1	1	0.2	0.3	0	0.1	0.2	0.7	1		1	1

Calculate net1 and y1

net1 = 0.2 * 1 + 0.3 * 1 + 0 = 0.5 → y1 = 1

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8	0	1	1	0
1	1	0.2	0.3	0	0.1	0.2	0.7	1	1	1	1

Calculate net2 and y2

net2 = 0.1 * 1 + 0.2 * 0.7 + 0 = 1 → y2 = 1

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8	0	1	1	0
1	1	0.2	0.3	0	0.1	0.2	0.7	1	1	1	1
Next weights		0.2	0.3	0	0.1	0.2	0.7				

both y1 and y2 are OK

Don’t update weights

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.1	0.2	-0.2	0.2	0.3	1	0	1	0	0
0	1	0.1	0.2	-0.2	0.2	0.3	0.9	0	1	1	0
1	0	0.1	0.3	-0.1	0.2	0.2	0.8	0	1	1	0
1	1	0.2	0.3	0	0.1	0.2	0.7	1	1	1	1
Next weights		0.2	0.3	0	0.1	0.2	0.7				

We need to run another epoch

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.2	0.3	0	0.1	0.2	0.7			0	0
0	1									1	0
1	0									1	0
1	1									1	1

Second epoch

2. SLP with multiple outputs

x1	x2	w11	w21	wb1	w12	w22	wb2	y1	y2	t1	t2
0	0	0.2	0.3	0	0.1	0.2	0.7	0	0	0	0
0	1	0.2	0.3	0	0.1	0.2	0.7	1	0	1	0
1	0	0.2	0.3	0	0.1	0.2	0.7	1	0	1	0
1	1	0.2	0.3	0	0.1	0.2	0.7	1	1	1	1
Next weights		0.2	0.3	0	0.1	0.2	0.7				

Second epoch