Task 1 Computation of the Probabilistic Binary Model

Additional Explanation

This is a simple example of the probabilistic computation.

So far, I think, the computation you have experienced was all deterministic.

You can always get the same result if the inputs are same.

Deterministic Computation

↕

Probabilistic computation (stochastic)

This is your first experience.

You might be confused.

But it is important for AI.

Even for the same inputs, the output can be different.

The output is computed using a die

The result changes by chance

For example, a human action can be different even for the same input

To help your understanding, I like to give you some questions.

You can use a die  A real die with $\boxed{6 \text{ faces}}$.

Q1 : Use this die, and let "0" and "1" with the equal probability $\boxed{0.5}$.

multiply

Your procedure

① Throw a die, and observe the number on the top face.

The number on the top face is two.

We can generate "0" and "1" with the equal probability 0.5.

→ This output ("0" or "1") can be different by chance. Even for the same inputs, the output is different.

② You output "0" when the number on the top face is larger than $\boxed{3}$

"1" when the number of the top face is less than or equal $\boxed{3}$

Q2 : Which number can you use, if you like to

generate { "0" with probability $1/6$
 "1" with probability $\boxed{5/6}$

the real die has $\boxed{6}$ faces

multiply

Generate { "0" if the number on the top face is larger than $\boxed{5}$
 "1" if the number on the top face is less than or equal to $\boxed{5}$.

In case of the ~~turn~~ random number generator.

↳ Artificial die with RAND_MAX faces,

Instead of 6, we multiply "RAND_MAX" to the probability of output "1".

↳ p

P * RAND_MAX.

Generate $\begin{cases} \text{"0" if } ran() > p * RAND\_MAX \\ \text{"1" if } \underline{ran()} \leqq p * RAND\_MAX. \end{cases}$

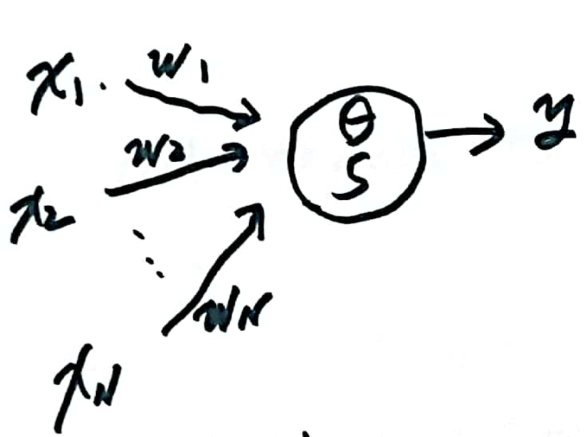
An integer between 0 and RAND_MAX is generated with the uniform probability.

2024. Jan. 9th.

What is the Recurrent Neural Network. (RNN)

We use the deterministic binary model of the neuron or the probabilistic binary model of the neuron.

Special case. ($\alpha \to \infty$)



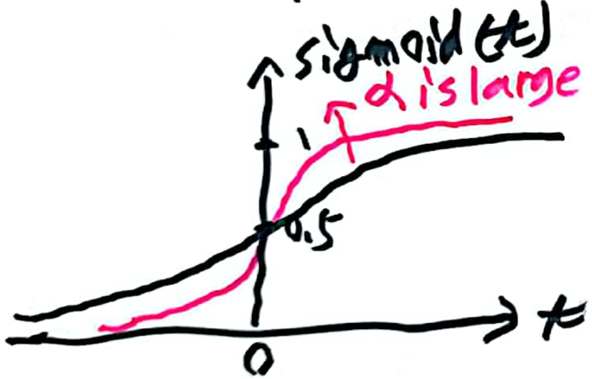
$$S = \sum_{i=1}^{N} w_i x_i$$

$$P = \text{sigmoid}(S - \theta)$$

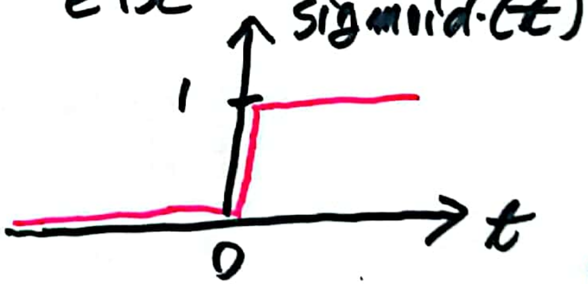
$$\text{sigmoid}(t) = \frac{1}{1 + e^{-\alpha t}}$$

α = Gain

$$y = \begin{cases} 0 & \text{if } ran() > P * RAND\_MAX \\ 1 & \text{else} \end{cases}$$



$\alpha \to \infty$



P takes only 0 or 1

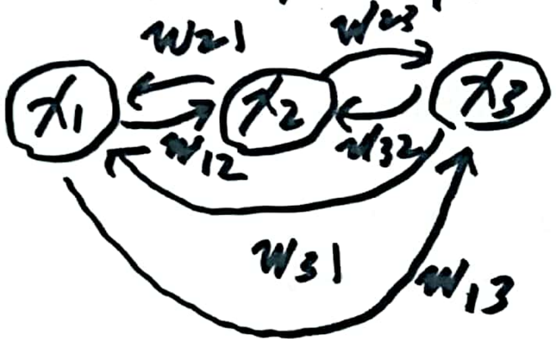
↓

Deterministic Model.

A simple RNN with three neurons.

We have used x_i to mean inputs, but from now, we use x_i to mean the states of neurons.

The output of a neuron i's memorised for ~~the input~~ its use as an input for the next ~~state~~ step.



x_i : state of the i-th neuron.

w_{ij} : Connection from the i-th neuron to the j-th neuron.

① We give initial values to x_1, x_2, x_3

② We update the state of the neuron one by one.

Update x_1 using the current x_2, x_3.

Update x_2 using the current x_1, x_3

⋮

"State" ~~means~~ means a memorized value

Output of a neuron in the previous step

Input for the neuron in the next step.

initial values	0	0	0
update x_1	1	0	0
update x_2	1	0	0
update x_3	1	0	1
update x_1	1	0	1
update x_2	1	1	1
update x_3	1	1	0
	⋮		

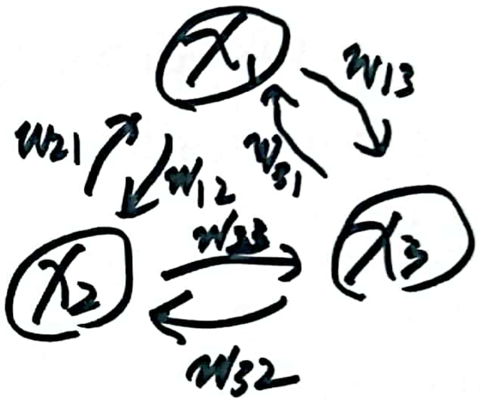
Repeat updating of x_i one by one.

A simple example of RNN application

The winner takes all.

↳ Among the three neurons, only one neuron can take 1 and others ~~x_1 x_2 x_3~~ take 0.

x_1	x_2	x_3	
1	1	1	← Initial values
0	1	1	← Update x_1
0	0	1	← Update x_2



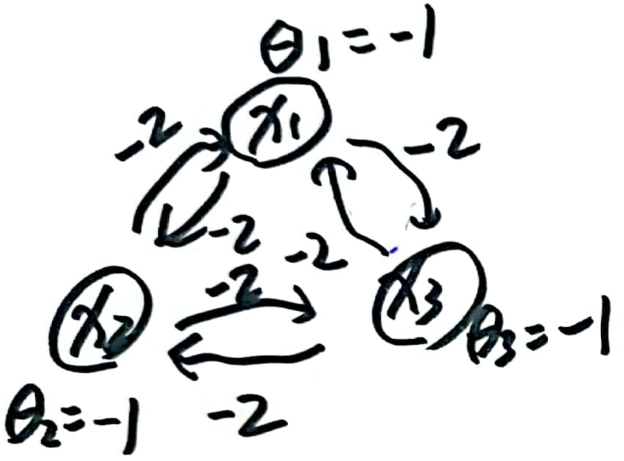
x_3 is the winner.

We can use a probabilistic model, the winner changes by chance.

For this "winner takes all" RNN, we use inhibitory connections.

In our brain, neurons are inhibiting others.

For inhibitory connections, we use negative values -2. $w_{ij} = -2$.



Deterministic Model

x_1	x_2	x_3	
1	1	1	← Initial values
0	1	1	← Update x_1
0	0	1	← Update x_2
0	0	1	← Update x_3

Never change afterward.

weighted sum for x_1

$$s_1 = w_{21}x_2 + w_{31}x_3 = -2*1 + (-2)*1 = -4$$

$$s_1 \leqq \theta_1 \qquad s_1 = -4,\ \theta_1 = -1$$

$$x_1 = 0$$

weighted sum for x_2

$$s_2 = w_{12}x_1 + w_{32}x_3 = -2*0 + (-2)*1 = -2$$

$$s_2 \leqq \theta_2 \qquad s_2 = -2,\ \theta_2 = -1$$

$$x_2 = 0$$

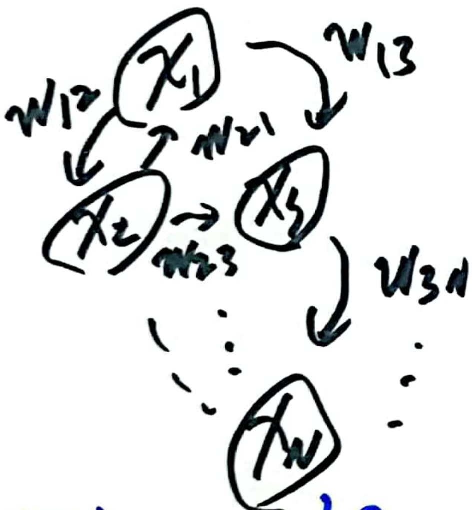
$$s_3 = w_{13}x_1 + w_{23}x_2 = -2*0 + (-2)*0 = 0 > \theta_3 = -1$$

RNN can solve many problems.

Most popular application is the optimization ~~problem~~.

In order to apply RNN to such applications, we need to introduce the idea of the energy.

The energy of a RNN is defined as follows,



$$E(x_1, x_2, x_3, \cdots x_N)$$
$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\, x_i x_j + \sum_{i=1}^{N} \theta_i x_i$$

w_{ij} : connection weight for $i \to j$ connection

θ_i : threshold of i-neuron.

x_i : state of i-neuron.

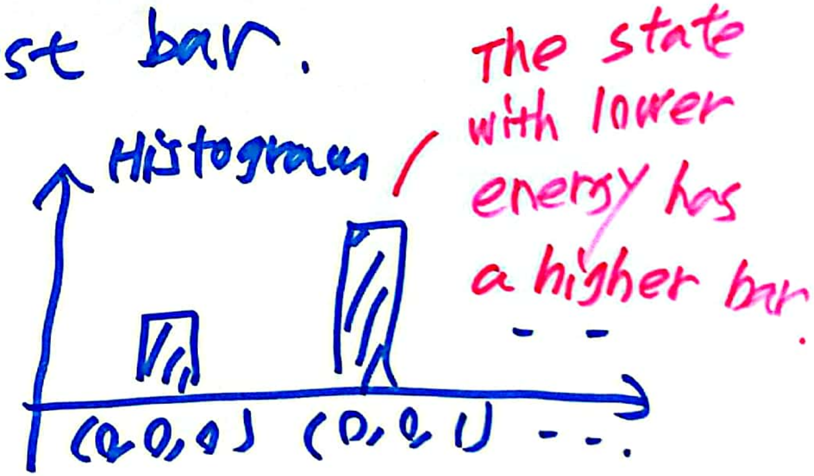
w_{ij} must be symmetric. $w_{ij} = w_{ji}$ $w_{ii} = 0$

This energy always decreases when we update neurons one by one according to the deterministic updating rule.

~~The~~ If we use the probabilistic updating rule, the state for the lowest energy can appear most frequently.

↳ If you make a histogram, the state with the lowest energy has the highest bar.

Graphical Presentation



Task 2

2-1 Make a program for a RNN with three neurons.
Check if it works correctly.
Use the example of the winner takes all.

$$w_{ij} = -2 \quad i = 1\sim3 \quad j = 1\sim3$$

$$\theta_i = -1 \quad i = 1\sim3$$

Give initial values for x_1, x_2 and x_3

$(x_1, x_2, x_3) = (1, 1, 1) \ (1, 1, 0) \ \cdots$

Use the deterministic binary model. → Update neurons one by one

$x_1 \to x_2 \to x_3$ index order

$x_3 \to x_2 \to x_1$ reversed order.

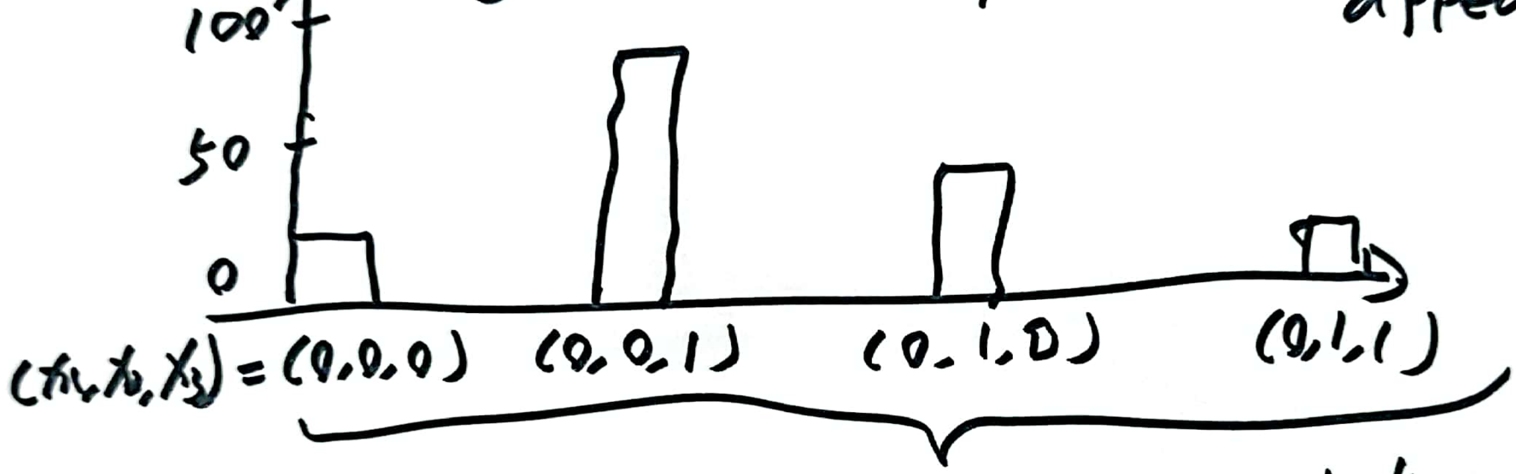
Check the result.

2-2 Use the probabilistic model for 2-1

Repeat updating many times. 100 times

1000 times 10,000 times.

Histogram (How many times does (x_1, x_2, x_3) appears?)



$2 \times 2 \times 2 = 8$ states.

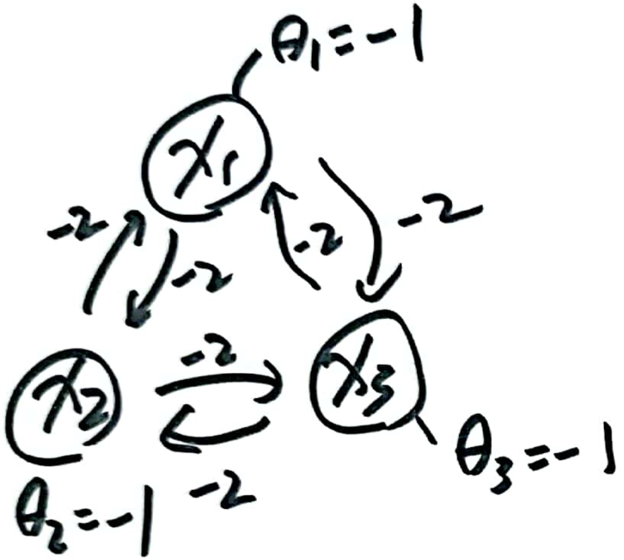
Initial values also influene the results

Be careful about the gain α.

if α is large, the probabilistic model can be close to the deterministic model, So the updating order can influence the results

In Task 2-2. please check if the state with larger energy value has the lower histogram bar.

For example, the energy of the winner takes all model is computed as follows.



$$
\begin{aligned}
E(x_1, x_2, x_3) &= -\frac{1}{2}\big(-2 * x_1 * x_2 - 2 * x_1 * x_3 - 2 * x_2 * x_3 \\
&\qquad -2 * x_2 * x_1 - 2 * x_3 * x_1 - 2 * x_3 * x_2\big) \\
&\qquad + (-1) * x_1 + (-1) * x_2 + (-1) * x_3 \\
&= +2 * x_1 * x_2 + 2 * x_1 * x_3 + 2\, x_2 * x_3 \\
&\qquad - x_1 - x_2 - x_3
\end{aligned}
$$

Task 2-3

When you update neurons one by one in Task 2-1, please check the decrease of the energy.

Deterministic Model