Summer AI

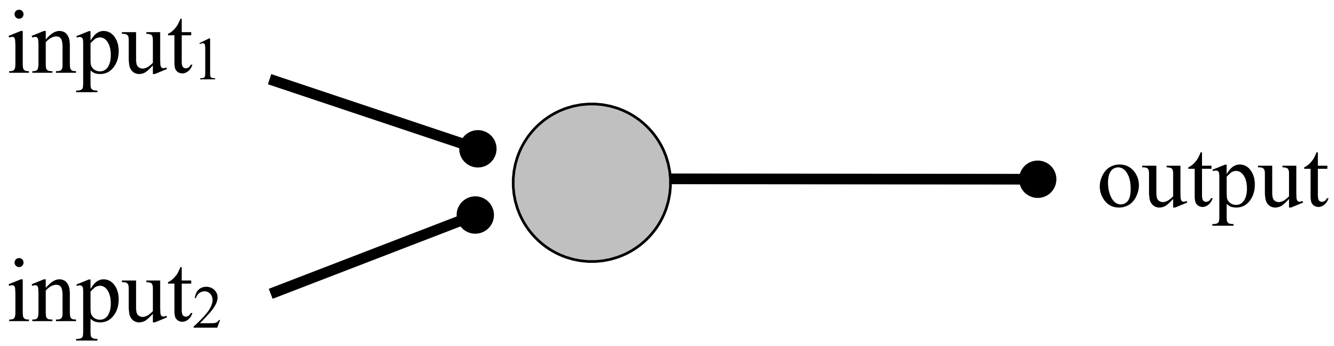
Teaching a very very simple neuron

Today

Let’s teach our simple neuron …

Remember …

Q: Consider the Perceptron



$\text{input}_1 = 10$ $\quad$ $w_1 = -0.5$

$\text{input}_2 = 8$ $\quad$ $w_2 = 1$

What is the activity (x)?

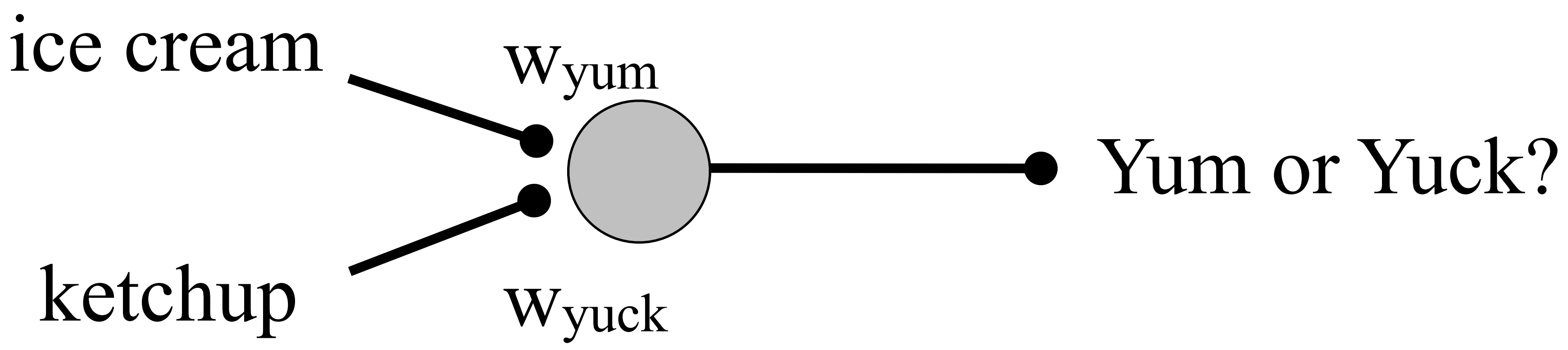
Perceptron: a classifier

Let’s examine a perceptron in action …

Specifically, let’s use a perceptron to **classify** some data.

Eat it?

Remember our Perceptron food critic,



We had to choose the weights to get it to work …

What if we don't want to choose the weights?

Instead, we want the Perceptron to learn on its own …

Perceptron training

To train our perceptron,

- We'll provide our perceptron with inputs & correct answer.
- The perceptron will compare its guess with the correct answer.
  - If the perceptron makes an incorrect guess,
    then it can learn from it's mistake

↑ **adjust its weights**

Let's do it ….

Perceptron training

Perceptron training in 5 steps:

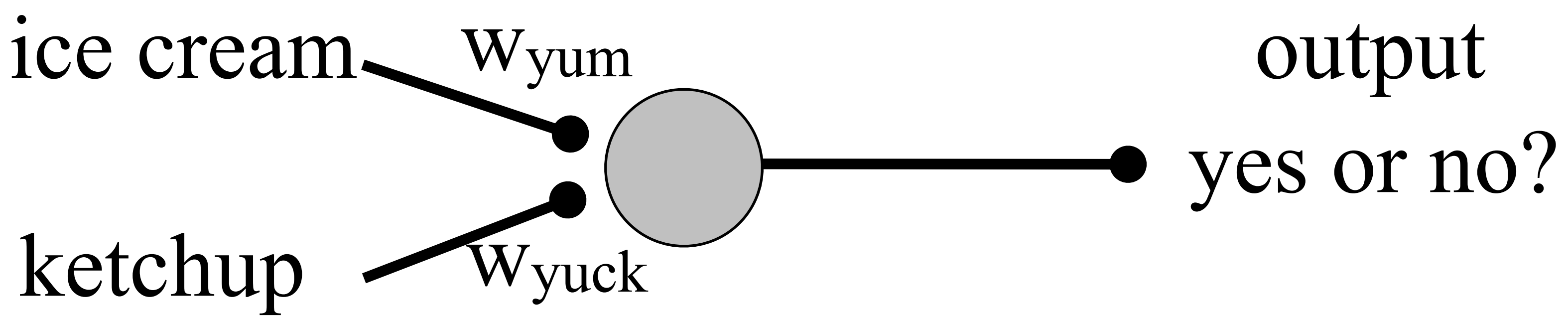
1. Provide perceptron with inputs and known answer.
2. Ask perceptron to guess an answer.
3. Compute the error: does perceptron get answer right or wrong?
4. Adjust weights according to the error. **Learning!**
5. Return to Step 1 and repeat.

Note: We know how to do Step 2, consider other steps …

forward propagation

Perceptron: a classifier

Consider the perceptron:



ice cream w_{yum}

ketchup w_{yuck}

output yes or no?

Two inputs: the things to eat

Computes an output: output = $\{0, 1\}$

interpret as “Yuck!”

interpret as “Eat it”

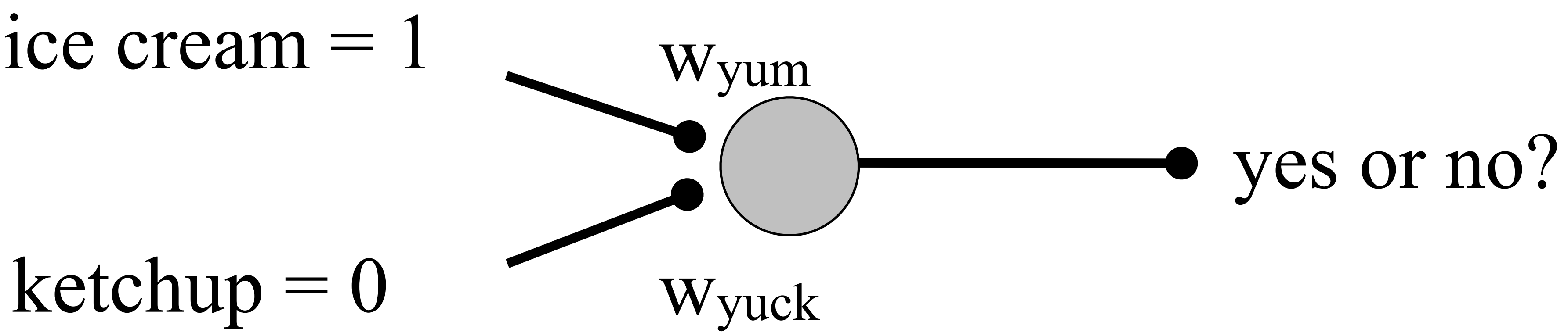
Weights: w_{yum}, w_{yuck}

We’ll need to specify those …

Perceptron classifier #1

We'd like to classify inputs as Yum or Yuck

Consider this input (ice cream alone):



ice cream = 1

ketchup = 0

w_{yum}

w_{yuck}

yes or no?

Q: What weights? To start let's choose: $w_{yum}=1$, $w_{yuck}=1$

Q: What is the output?

$$\text{ice cream} * w_{yum} + \text{ketchup} * w_{yuck} = 1 * 1 + 0 * 1 = 1 > 0$$

so, output = 1

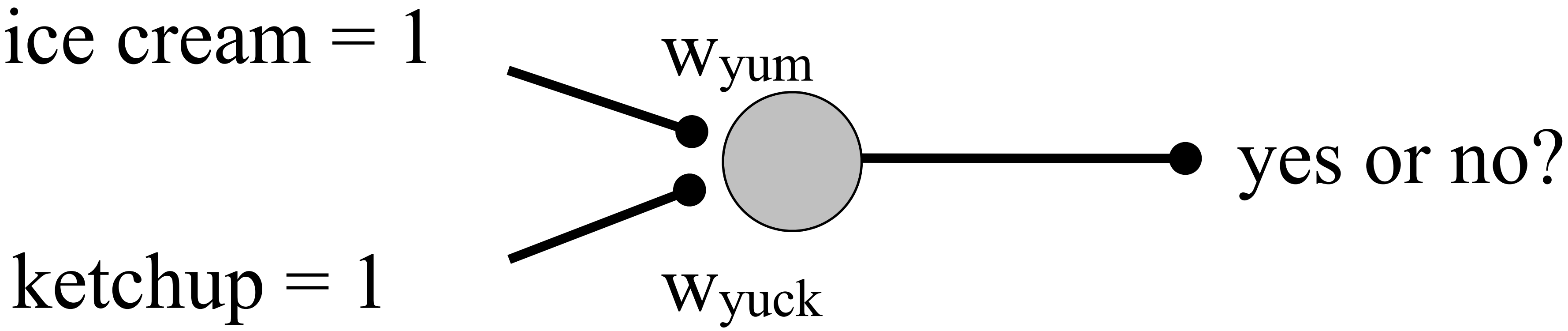
interpret as "Yum!"

Perceptron <u>succeeds</u>!

Perceptron classifier #1

We'd like to classify inputs as Yum or Yuck

Consider this input:



ice cream = 1

ketchup = 1

w_{yum}

w_{yuck}

yes or no?

Keep weights fixed at $w_{yum}=1$, $w_{yuck}=1$

Q: What is the output?

ice cream $*$ w_{yum} + ketchup $*$ w_{yuck} $= 1 * 1 + 1 * 1 = 2 \quad > 0$

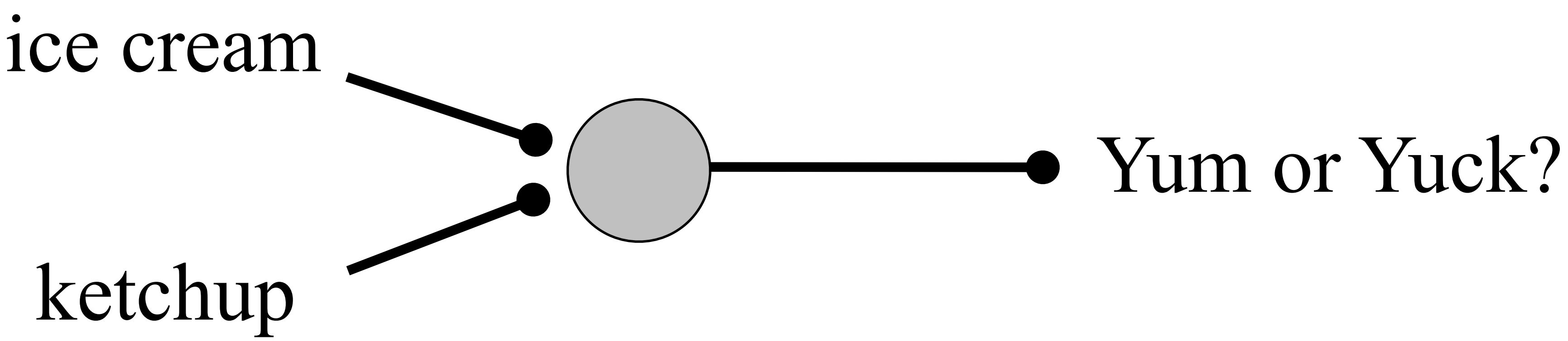
so, output = 1

interpret as "Yum!"

Perceptron fails!

Perceptron classifier #2: Summary

Summary of perceptron classifier:



For inputs (ice cream, ketchup) ask the perceptron:

Is it Yum (output 1) or is it Yuck (output 0)?

Q: Will the perceptron get classification right?

A: If we’re lucky, then maybe … but we need to train it!

Perceptron training: Step 3

Consider Step 3. *Compute the error*

Q: What is the perceptron’s error?

Let’s define it:

Difference between desired answer and perceptron’s guess.

Error = **Desired output - Perceptron output**

In our case: {0, 1} {0, 1}

The output has only 2 possible states (Yum or Yuck).

Perceptron training: Step 3

Let’s make a table of possible error values:

Desired output	Perceptron output	Error	
0 Yuck	0 Yuck	0	ok!
0 Yuck	1 Yum	-1	:(
1 Yum	0 Yuck	1	:(
1 Yum	1 Yum	0	ok!

Note: the error is 0 when perceptron guesses the correct output

the error is +1 or -1 when perceptron guesses the wrong output

Next step: use the error to adjust the weights …

Perceptron training: Step 4

Consider Step 4. *Adjust all weights according to the error.*

The error determines how weights should be adjusted.

Let's define the change in weight:

$$\triangle \text{ weight} = \text{Error} * \text{Input}$$

Then, to update the weight:

$$\begin{aligned} \text{New weight} &= \text{weight} + \triangle \text{ weight} \\ &= \text{weight} + \text{Error} * \text{Input} \end{aligned}$$

Note: The error determines how the weight should be adjusted

big error — big change in weight

Perceptron training: Step 4

So, for our perceptron to learn:

– adjust the weights according to the error.

We'll also include a **learning constant**:

Compute this for Step 4:

New weight = weight + Error * Input * Learning Constant

When learning constant is big: weights change more drastically.

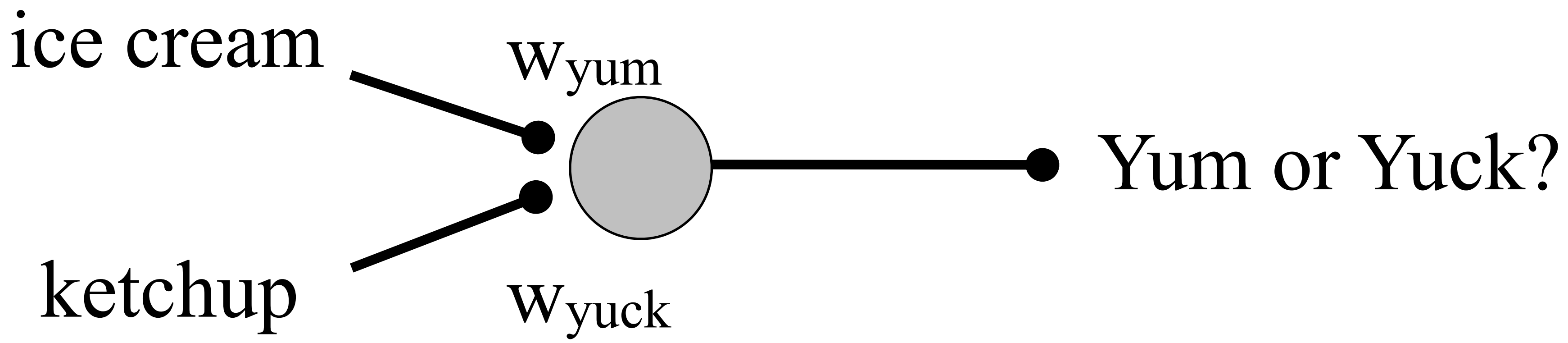
- Learn more quickly.

When learning constant is small: weights change more slowly.

- Improve accuracy

Perceptron training: by-hand

Let’s train the perceptron …



ice cream

w_{yum}

ketchup

w_{yuck}

Yum or Yuck?

Initialize:

All weights = 1

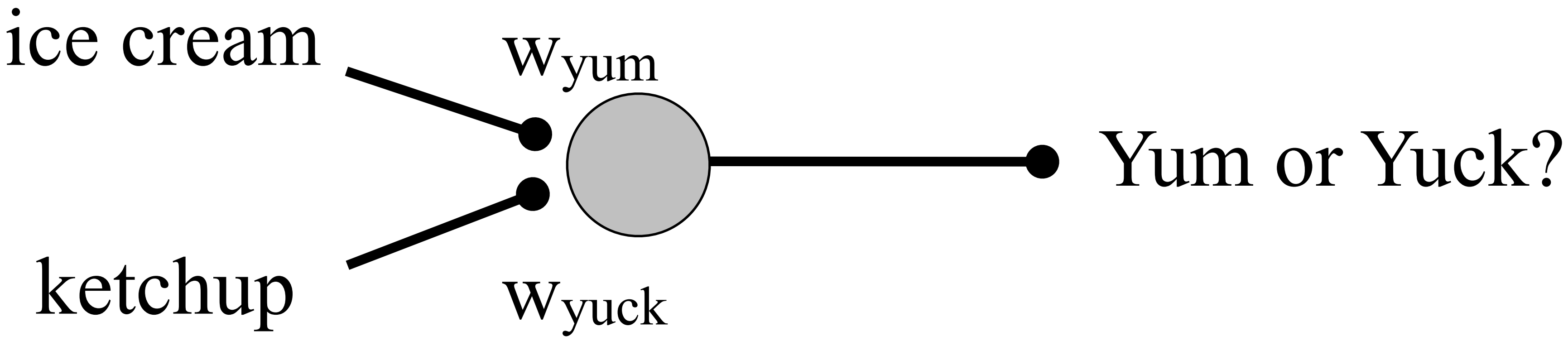
Learning constant = 0.01

ice cream without ketchup = Yum!
ice cream with ketchup = Yuck!

This is the relationship we want our perceptron to learn …

Perceptron training: by-hand

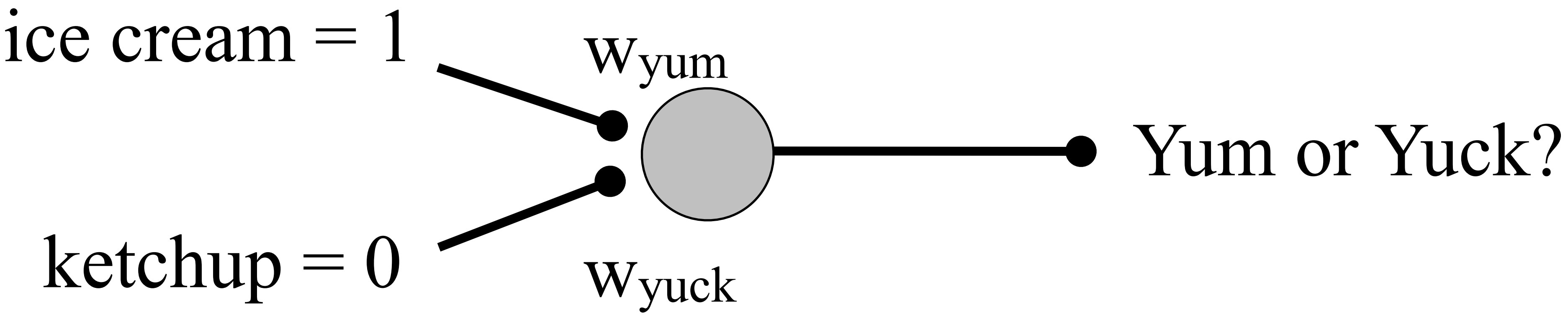
Step 1: *Provide perceptron with inputs and known answer.*



ice cream = 1 ketchup = 0 Yum!

Perceptron training: by-hand

Step 2. *Ask perceptron to guess an answer.*



ice cream = 1

ketchup = 0

w_{yum}

w_{yuck}

Yum or Yuck?

Compute weighted summed inputs:

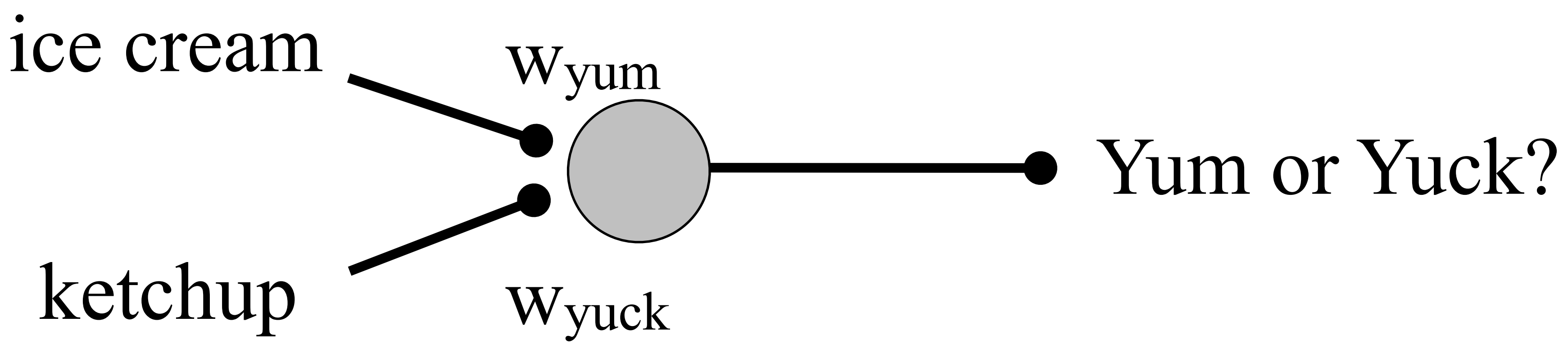
$$\text{ice cream} * w_{yum} + \text{ketchup} * w_{yuck} = \quad 1 * 1 \ + 0 * 1 \qquad = 1$$

$$\text{So, ice cream} * w_{yum} + \text{ketchup} * w_{yuck} \quad > 0$$

$$\text{So, } \boxed{\text{output} = \ 1}$$

Perceptron training: by-hand

Step 3. *Compute the error.*



Perceptron output = 1 (Perceptron: “Yum!”)

Desired output = 1 (Us: Yum!)

Error = **Desired output - Perceptron output**

= 1 - 1

= 0 No error, perceptron guess is correct.

Perceptron training: by-hand

Step 4. *Adjust all weights according to the error.*

New weight = weight + Error * Input * Learning Constant

w_{yum} : 1 + 0 * 1 * 0.01 = 1

w_{yuck} : 1 + 0 * 0 * 0.01 = 1

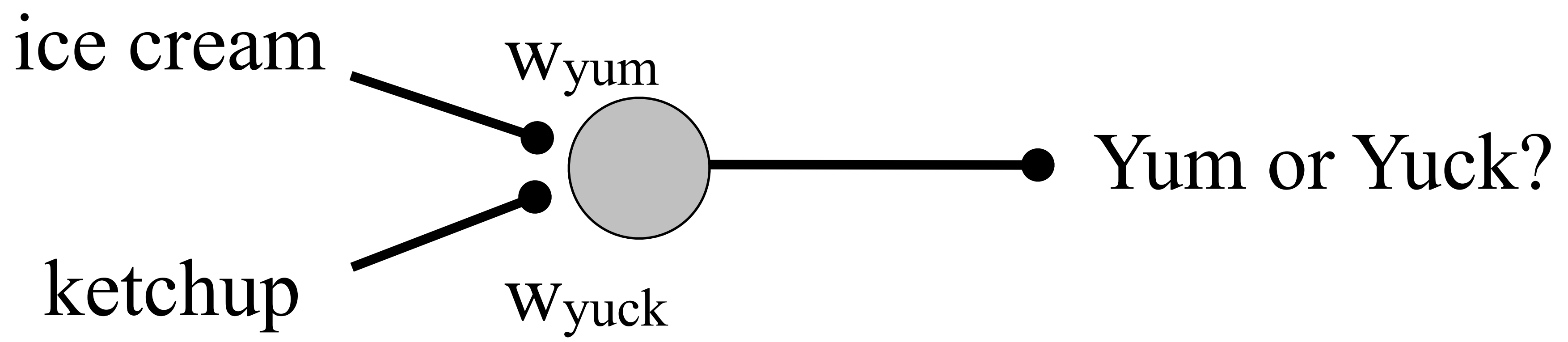
No change in weights!

Q: Our Perceptron is already “smart enough”?

Step 5. *Return to Step 1 and repeat …*

Perceptron training: by-hand

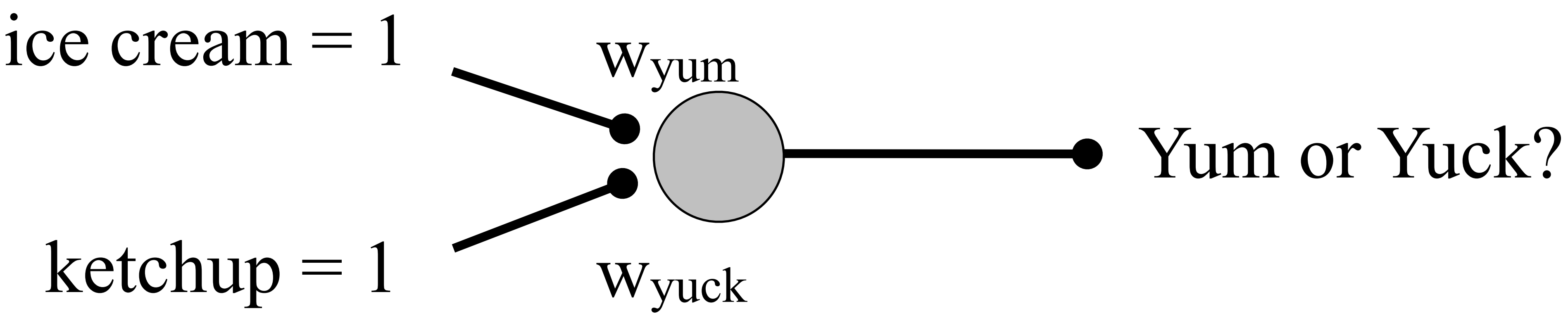
Step 1: *Provide perceptron with inputs and known answer.*



ice cream = 1 ketchup = 1 Yuck!

Perceptron training: by-hand

<u>Step 2</u>. *Ask perceptron to guess an answer.*



ice cream = 1

ketchup = 1

w_{yum}

w_{yuck}

Yum or Yuck?

Compute weighted summed inputs:

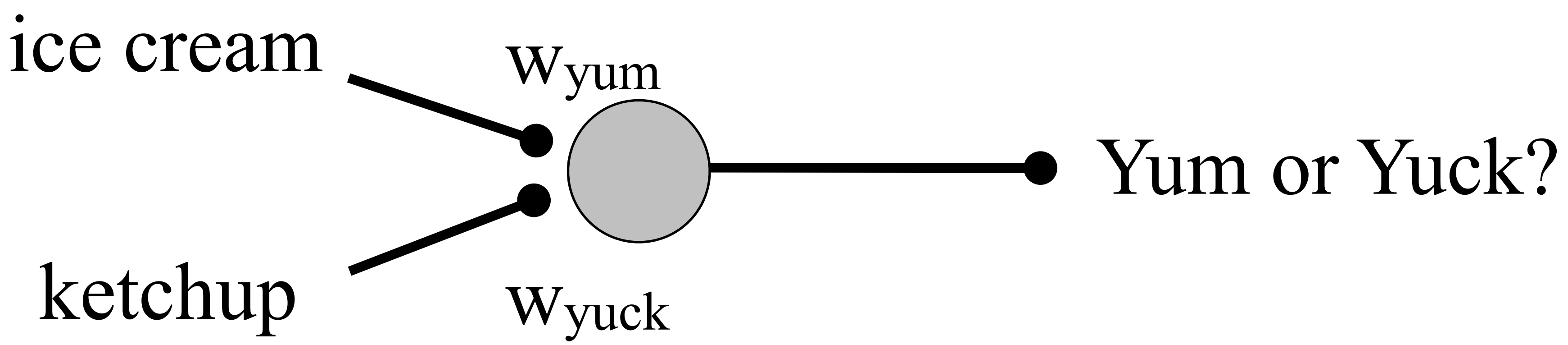
$$\text{ice cream} * w_{yum} + \text{ketchup} * w_{yuck} = \quad 1 * 1 \; + 1 * 1 \qquad = 2$$

So, $\text{ice cream} * w_{yum} + \text{ketchup} * w_{yuck} \quad > 0$

So, $\boxed{\text{output} = \;1}$

Perceptron training: by-hand

Step 3. *Compute the error.*



Perceptron output = 1 (Perceptron: "Yum!")

Desired output = 0 (Us: Yuck!)

Error = **Desired output - Perceptron output**

= 0 - 1

= -1 Error, the perceptron guess is wrong.

Perceptron training: by-hand

Step 4. *Adjust all weights according to the error.*

New weight = weight + Error * Input * Learning Constant

w_{yum} : $1 + -1 * 1 * 0.01 = 0.99$

w_{yuck} : $1 + -1 * 1 * 0.01 = 0.99$

We’ve changed the weights!

Q: Our Perceptron is already “smart enough”?

A: No, our Perceptron is “getting smarter”

Perceptron training: by-hand

Step 5. *Return to Step 1 and repeat …*

In fact, repeat the entire process 1000 times (or more).

Each time:

- Choose a combination (ice cream & ketchup)
- Determine if it's Yum or Yuck
- Ask the perceptron.
- Adjust the weights.

Q: Could you do this by hand?

Q: Would you do this by hand?

Next

Let’s get Python to do it!