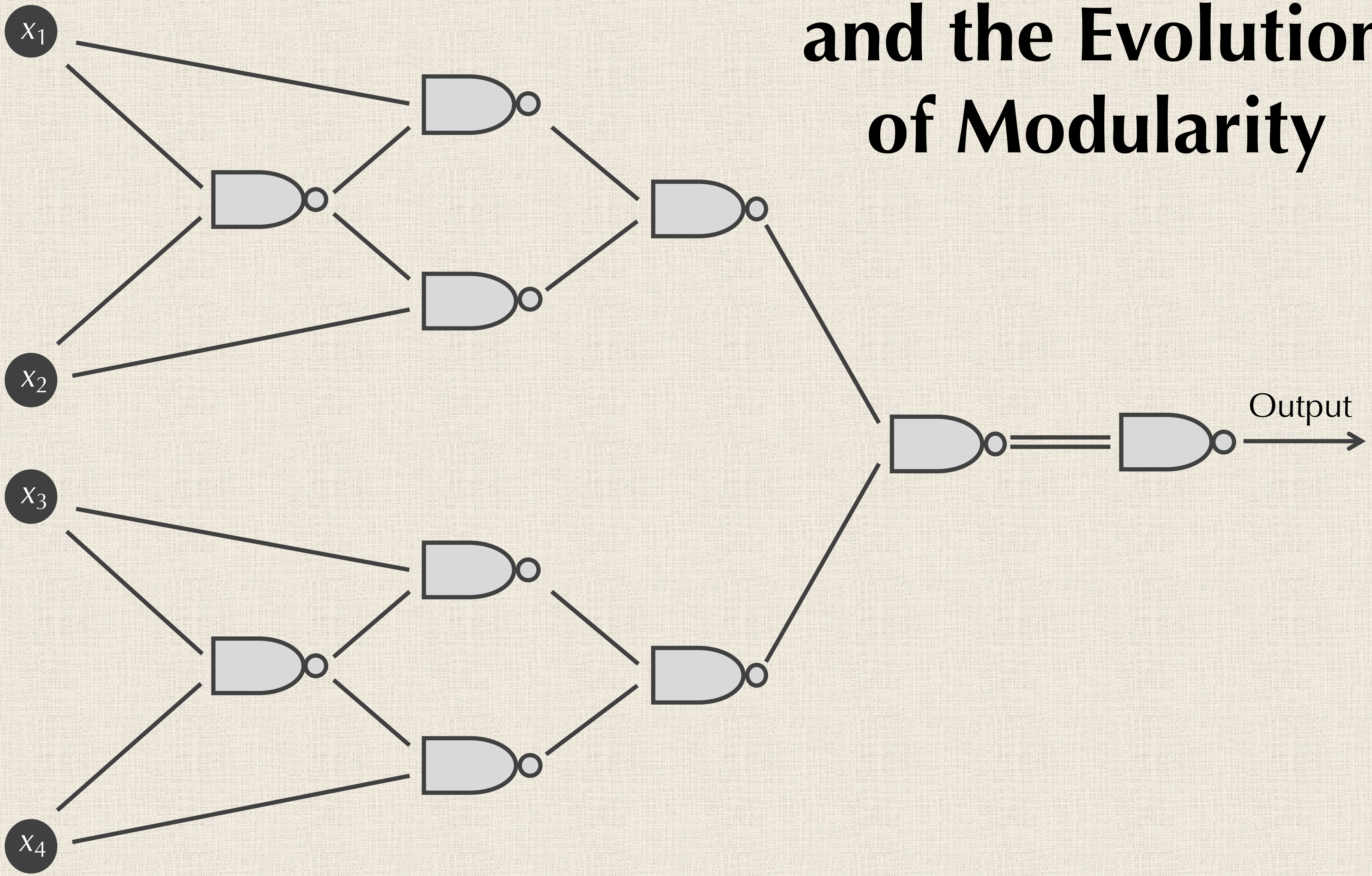
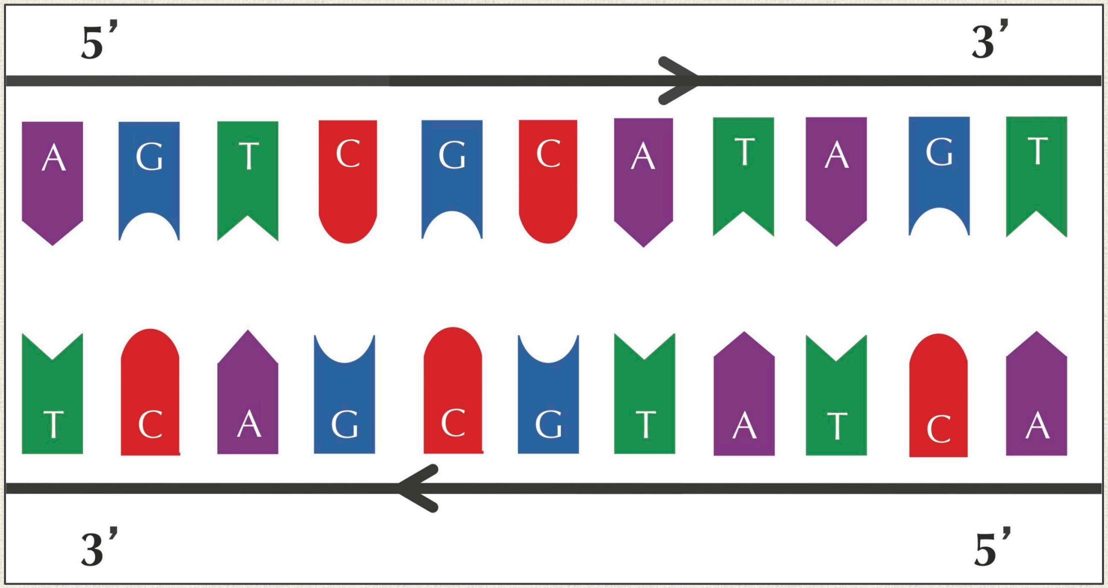
Neural Networks and the Evolution of Modularity



MODULARITY WUT?

Quick Review Question

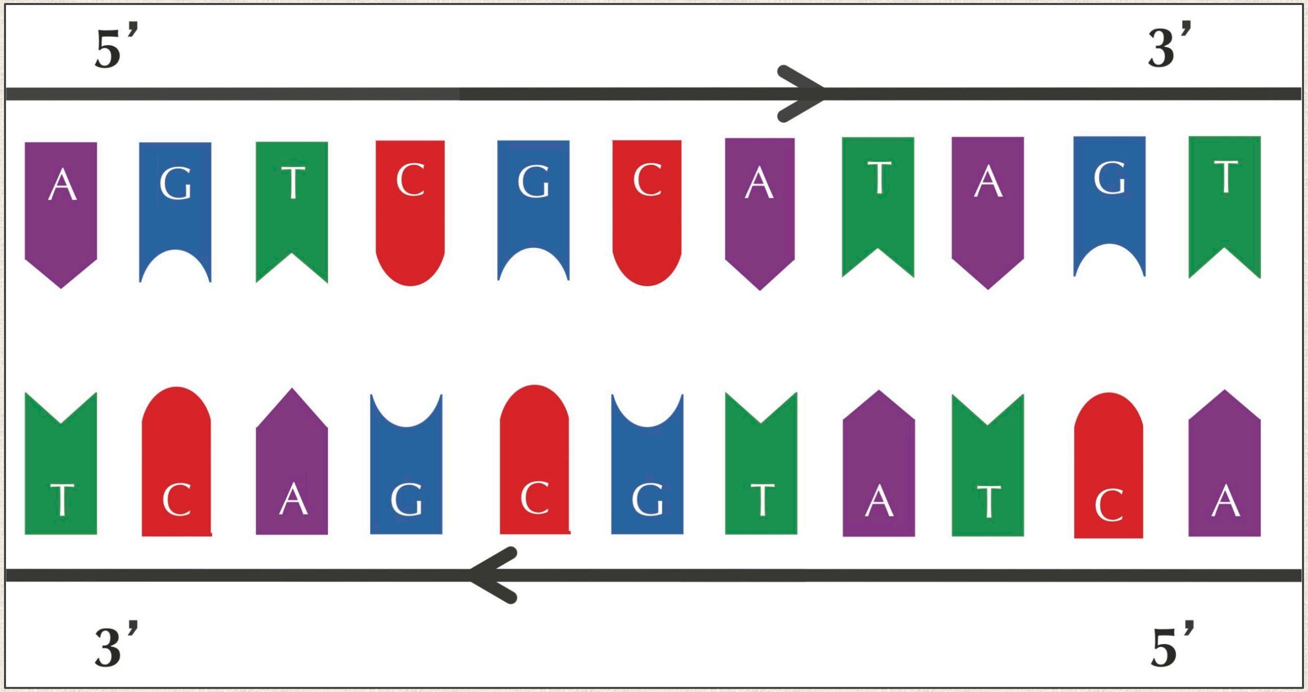


Reverse Complement Problem

- **Input:** A DNA string *s*.
- **Output:** The reverse complement of *s*.

STOP: How would you write code to solve this?

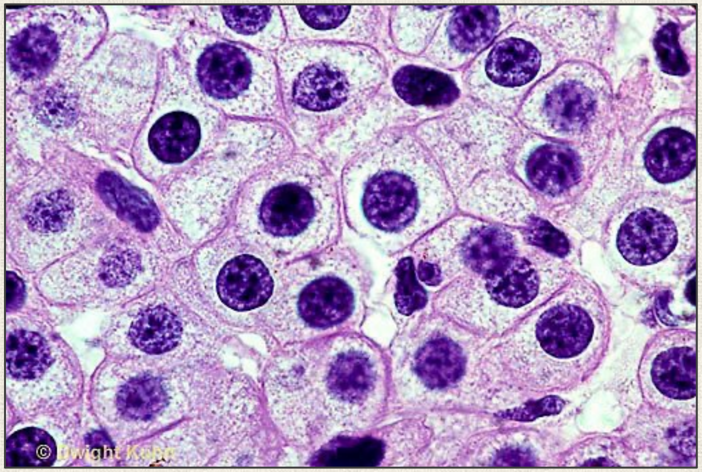
A "Modular" Reverse Complement Function is Best!

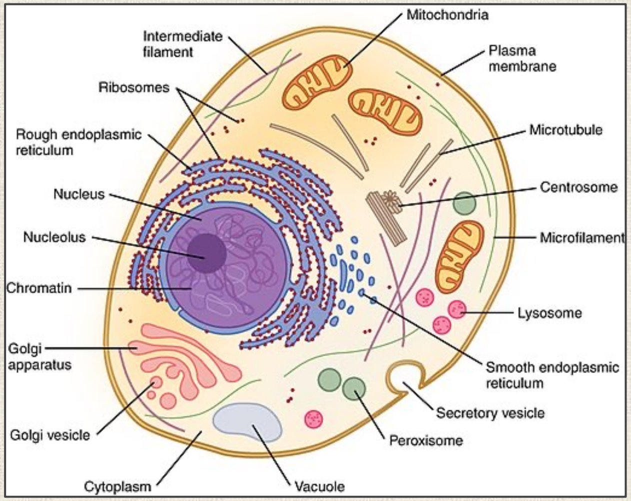


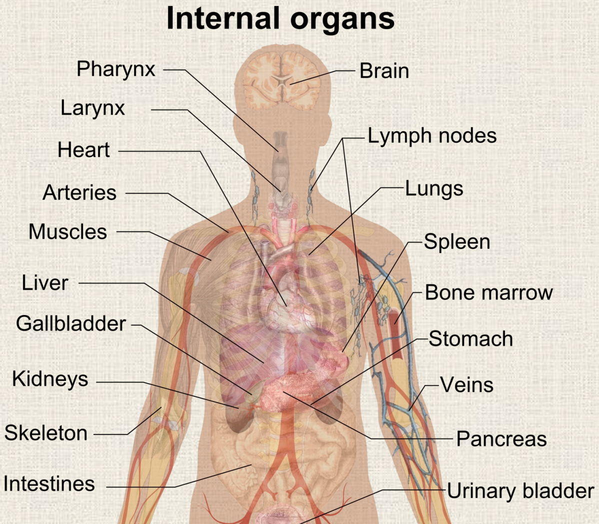

```
ReverseComplement(s)
    return Reverse(Complement(s))
```

STOP: What does it mean for code to be "modular"?

Modularity is everywhere in biology

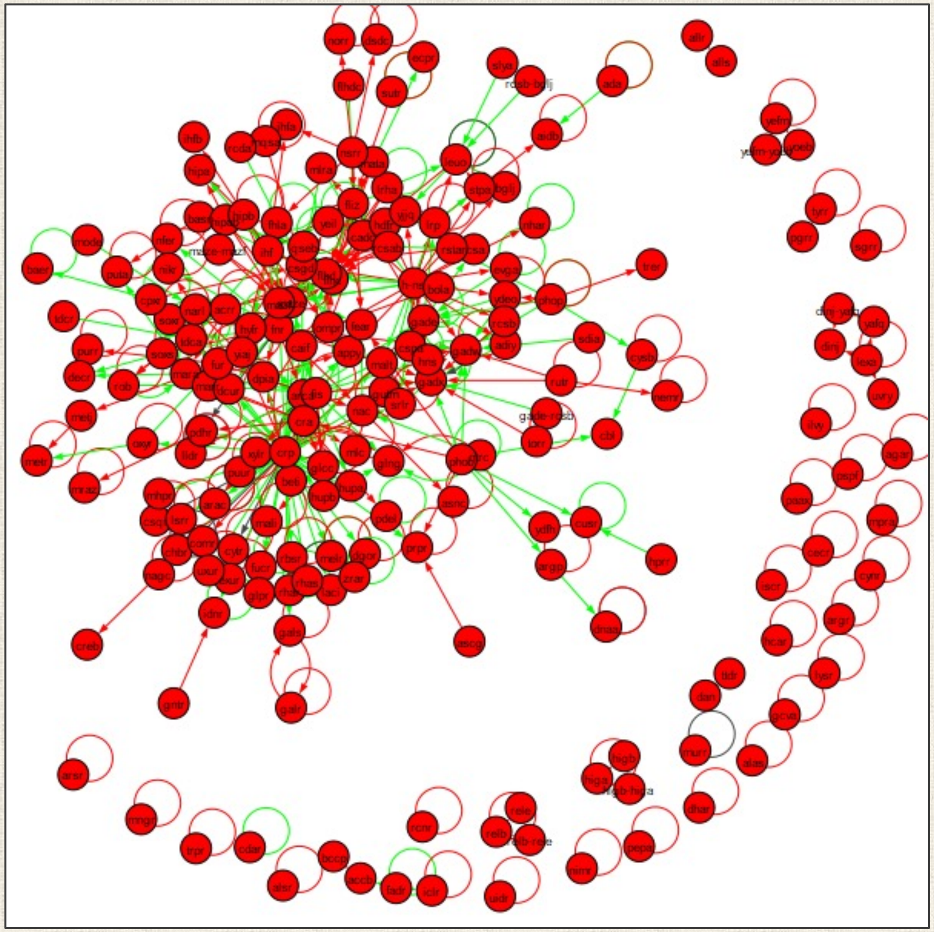




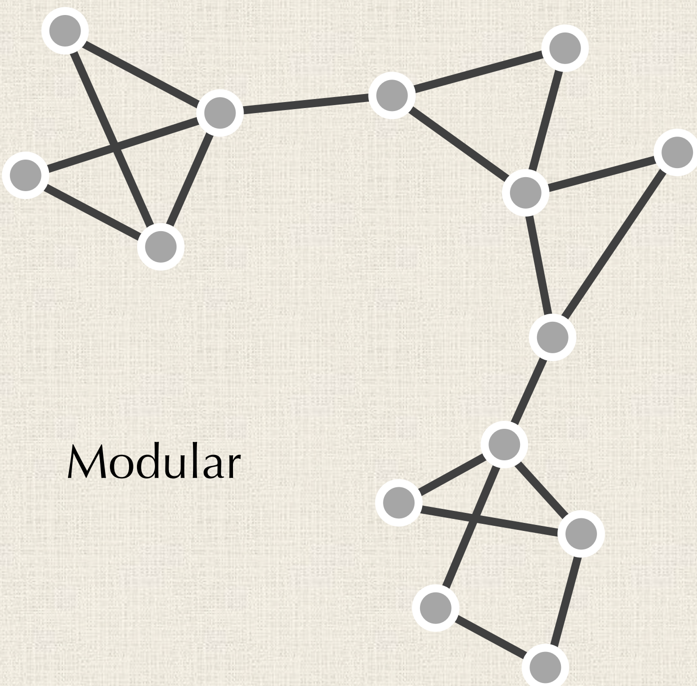


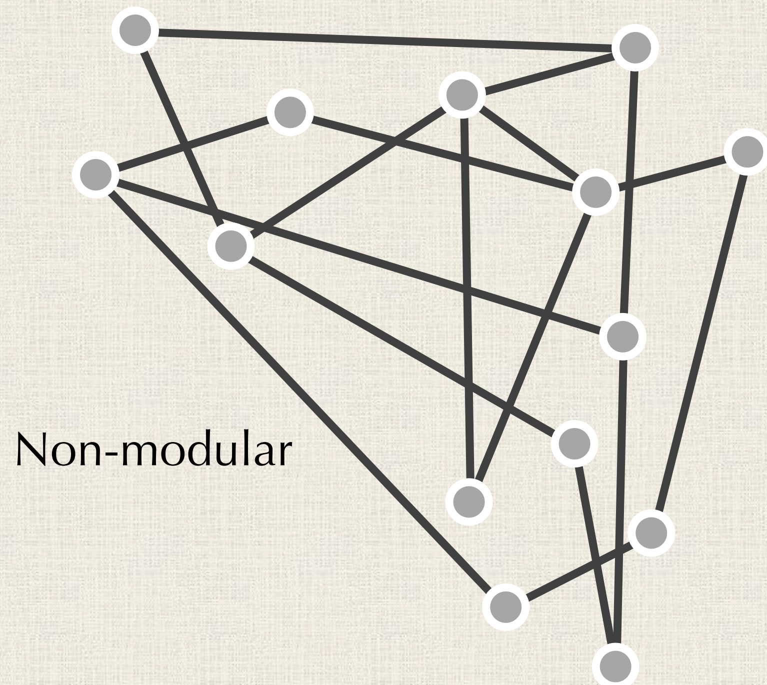
We already know that modularity occurs in biological networks

The "network motifs" that we saw in TF networks are their own form of modularity.



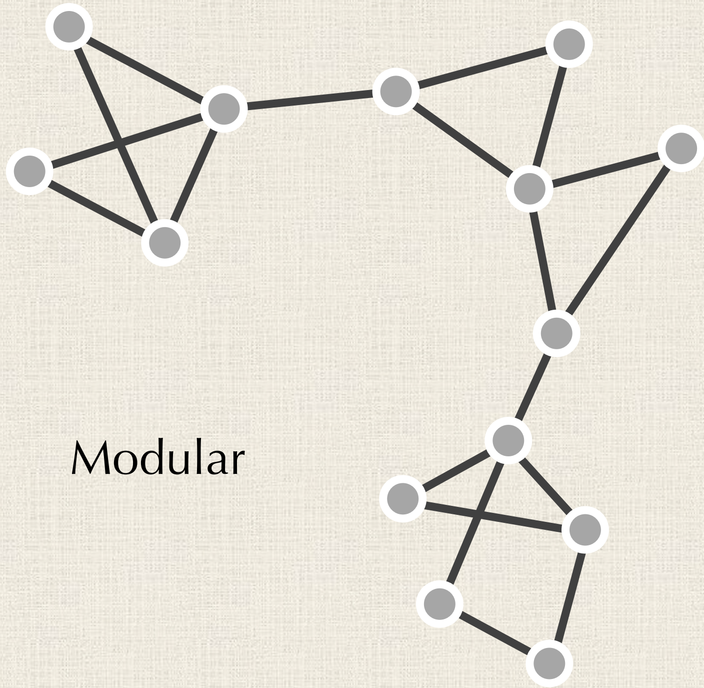
Modularity in Graphs

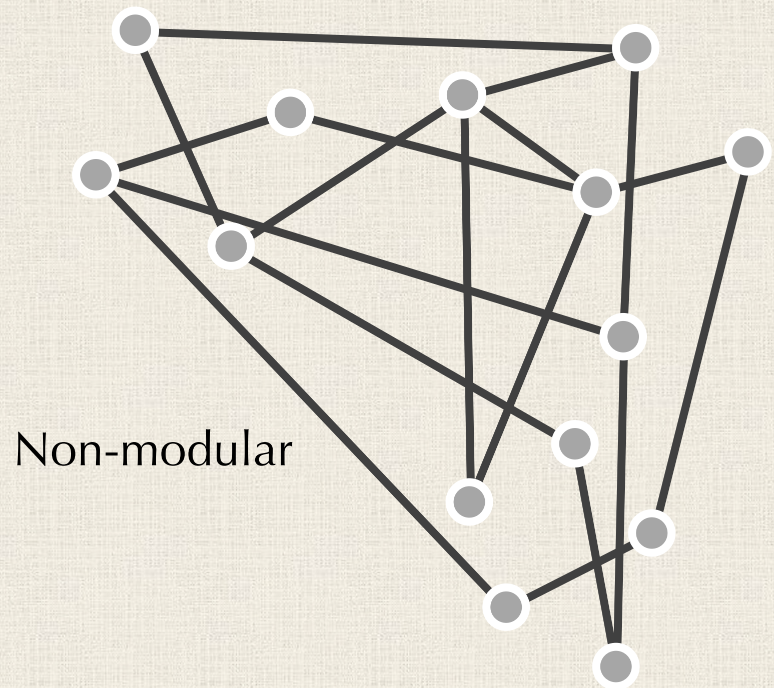




STOP: What should it mean for a graph to be "modular"?

Modularity in Graphs





Answer: It should divide into subgraphs so that two nodes from one subgraph are more likely to be connected than two nodes from different subgraphs.

Modular Code is Best, Right?

STOP: Is our `ReverseComplement()` function the best way to reverse complement a string?

```
ReverseComplement(s)
    return Reverse(Complement(s))
```

Not if we care about speed!

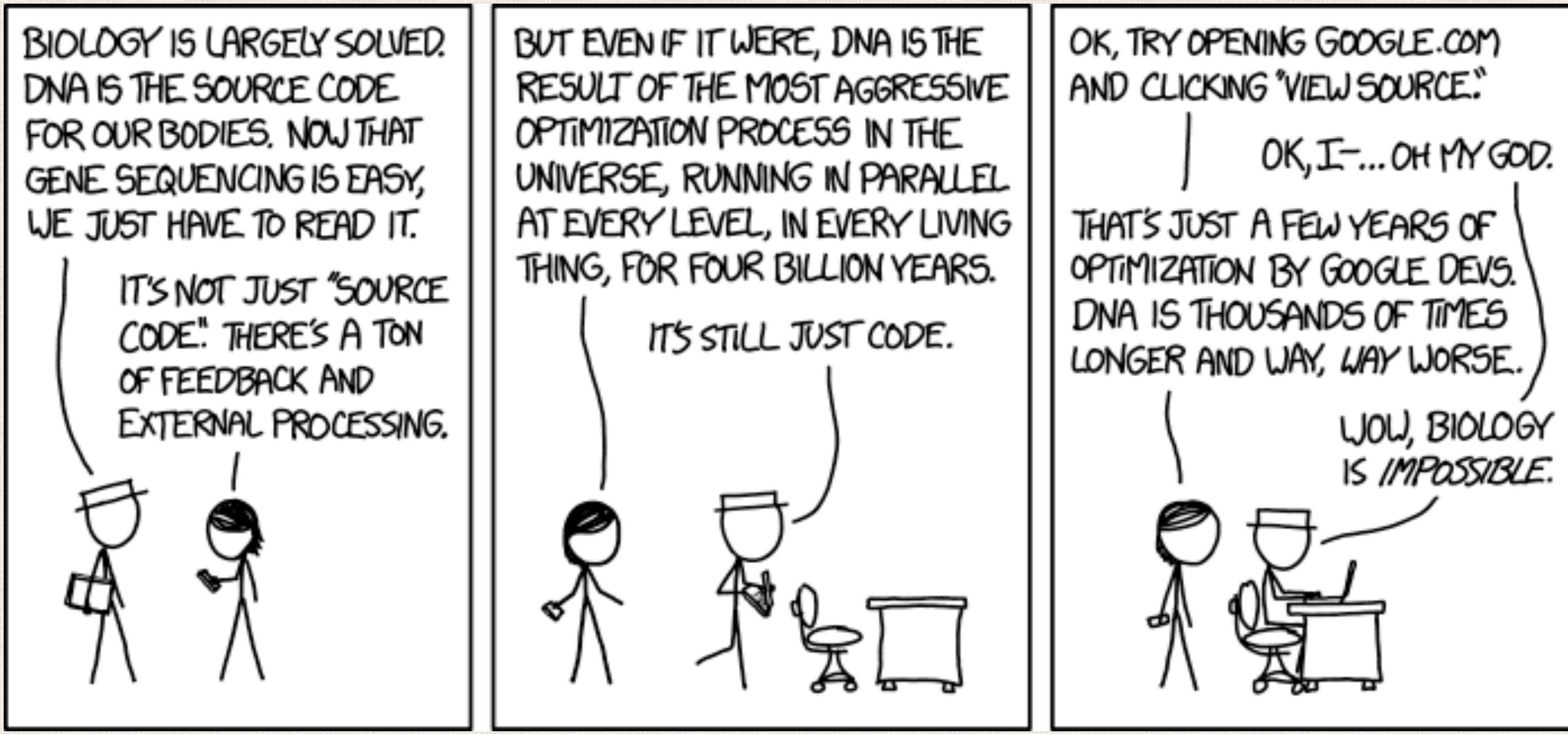
```
ReverseComplement(s):
    revComp = ""

    complementMap = {
        'A': 'T',
        'T': 'A',
        'C': 'G',
        'G': 'C'
    }

    for i = Length(DNAString) – 1 to 0
        currentChar = DNAString[i]
        complementChar = complementMap[currentChar]
        revComp = revComp + ComplementChar

    return revComp
```

Modular code is good practice, but optimized code can be non-modular

```html
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta charset="UTF-8"><meta content="origin" name="referrer"><meta
content="Search the world's information, including webpages, images, videos and more. Google has many special features to help you find exactly what you're
looking for." name="description"><meta content="noodp" name="robots"><meta content="/images/branding/googleg/1x/googleg_standard_color_128dp.png"
itemprop="image"><meta content="origin" name="referrer"><title>Google</title><script nonce="U2J5inrFmU2AB7s/
N08z0Q==">(function(){window.google={kEI:'CYaCXYX-L8-1ggezkIJw',kEXPI:'31',authuser:0,kscs:'790932f9_CYaCXYX-L8-
1ggezkIJw',u:'790932f9',kGL:'US',kBL:'q48m'};google.sn='webhp';google.kHL='en';google.jsfs='Ffpdje';})();(function(){google.lc=[];google.li=0;google.getEI=f
unction(a){for(var b;a&&(!a.getAttribute||!(b=a.getAttribute("eid")));)a=a.parentNode;return b||google.kEI};google.getLEI=function(a){for(var
b=null;a&&(!a.getAttribute||!(b=a.getAttribute("leid")));)a=a.parentNode;return
b};google.https=function(){return"https:"==window.location.protocol};google.ml=function(){return null};google.time=function(){return(new
Date).getTime()};google.log=function(a,b,e,c,g){if(a=google.logUrl(a,b,e,c,g)){b=new Image;var
d=google.lc,f=google.li;d[f]=b;b.onerror=b.onload=b.onabort=function(){delete
d[f]};google.vel&&google.vel.lu&&google.vel.lu(a);b.src=a;google.li=f+1}};google.logUrl=function(a,b,e,c,g){var d="",f=google.ls||"";e||-
1!=b.search("&ei=")||(d="&ei="+google.getEI(c),-1==b.search("&lei=")&&(c=google.getLEI(c))&&(d+="&lei="+c));c="";!e&&google.cshid&&-
1==b.search("&cshid=")&&"slh"!=a&&(c="&cshid="+google.cshid);a=e||"/"+(g||"gen_204")+"?atyp=i&ct="+a+"&cad="+b+d+f+"&zx="+google.time()+c;/^http:/
i.test(a)&&google.https()&&(google.ml(Error("a"),!1,{src:a,glmm:1}),a="");return a};}).call(this);(function(){google.y={};google.x=function(a,b){if(a)var
c=a.id;else{do
c=Math.random();while(google.y[c])}google.y[c]=[a,b];return!1};google.lm=[];google.plm=function(a){google.lm.push.apply(google.lm,a)};google.lq=[];google.lo
ad=function(a,b,c){google.lq.push([[a],b,c])};google.loadAll=function(a,b){google.lq.push([a,b])};}).call(this);google.f={};(function(){google.hs={h:true};}
)();(function(){google.c={};(function(){var f=window.performance;var
g=function(a,b,c){a.addEventListener?a.addEventListener(b,c,!1):a.attachEvent&&a.attachEvent("on"+b,c)};google.timers={};google.startTick=function(a){google
.timers[a]={t:{start:google.time()},e:{},m:{}}};google.tick=function(a,b,c){google.timers[a]||google.startTick(a);c=void 0!==c?c:google.time();b instanceof
Array||(b=[b]);for(var e=0,d;d=b[e++];)google.timers[a].t[d]=c};google.c.e=function(a,b,c){google.timers[a].e[b]=c};google.c.b=function(a){var
b=google.timers.load.m;b[a]&&google.ml(Error("a"),!1,{m:a});b[a]=!0};google.c.u=function(a){var b=google.timers.load.m;if(b[a]){b[a]=!1;for(a in
b)if(b[a])return;google.csiReport()}else google.ml(Error("b"),!1,{m:a})};google.rll=function(a,b,c){var
e=function(d){c(d);d=e;a.addEventListener?a.removeEventListener("load",d,!1):a.attachEvent&&a.detachEvent("onload",d);d=e;a.addEventListener?a.removeEventLi
stener("error",d,!1):a.attachEvent&&a.detachEvent("onerror",d)};g(a,"load",e);b&&g(a,"error",e)};google.aft=function(a){a.setAttribute("data-
iml",google.time())};google.startTick("load");var h=google.timers.load;a:{var k=h.t;if(f){var l=f.timing;if(l){var
m=l.navigationStart,n=l.responseStart;if(n>m&&n<=k.start){k.start=n;h.wsrt=n-m;break
a}}f.now&&(h.wsrt=Math.floor(f.now()))}}google.c.b("pr");google.c.b("xe");}).call(this);})();(function(){var
b=[function(){google.tick&&google.tick("load","dcl")}];google.dclc=function(a){b.length?b.push(a):a()};function c(){for(var
a;a=b.shift();)a()}window.addEventListener?(document.addEventListener("DOMContentLoaded",c,!1),window.addEventListener("load",c,!1)):window.attachEvent&&win
dow.attachEvent("onload",c);}).call(this);(function(){var
```

Here is some HTML source code from google.com.

Much of biology is hyper-optimized …



https://xkcd.com/1605/

… and yet modularity in some contexts must be worth preserving

Although modularity is important to many biological processes, no one built a model in which modularity spontaneously evolved until 2005.

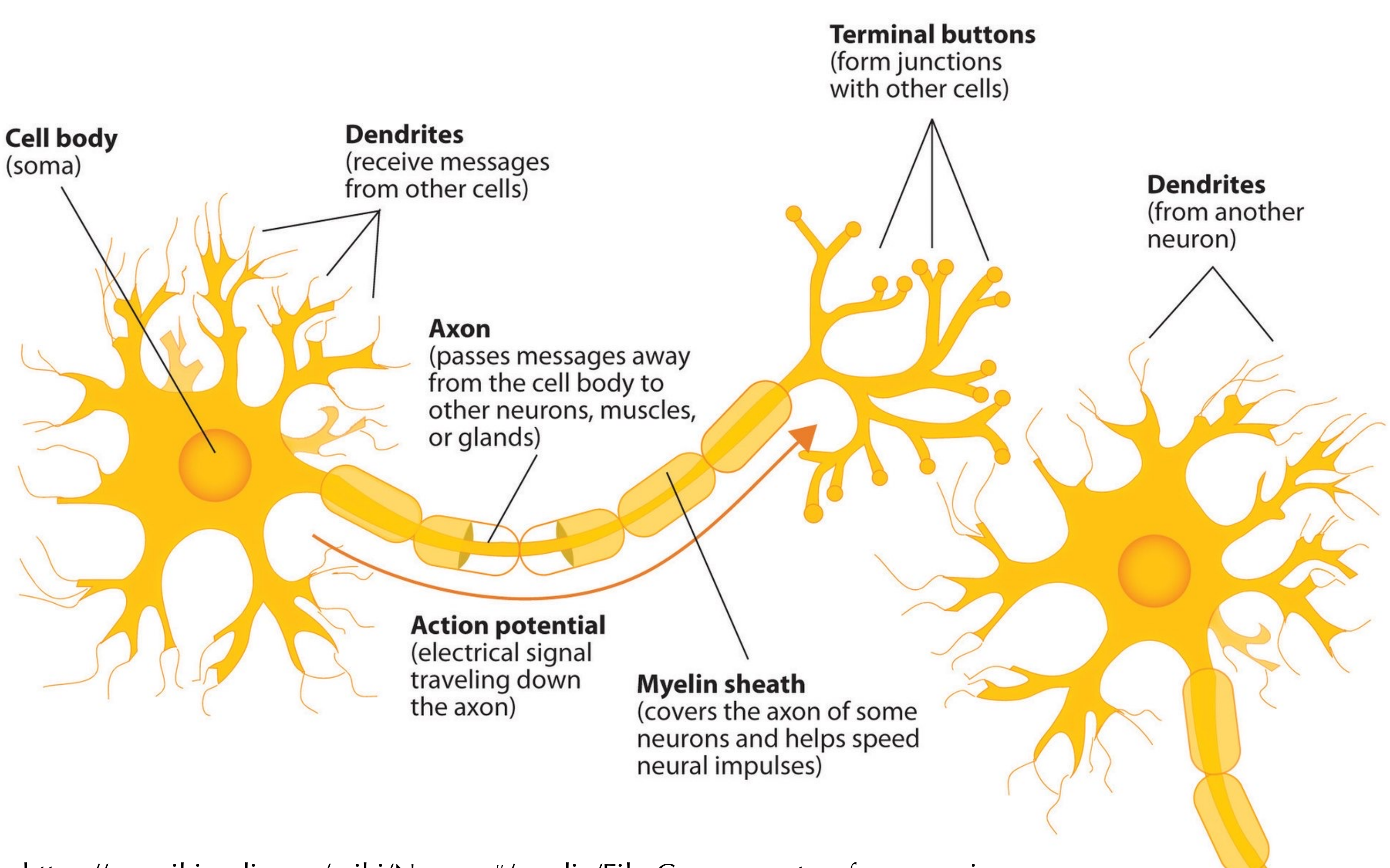
https://www.pnas.org › content

Spontaneous evolution of modularity and network motifs | PNAS

by N Kashtan · 2005 · Cited by 899 — Nadav **Kashtan** and Uri **Alon** ... To understand the origin of **modularity** and network motifs in biology one has to understand how these features ...

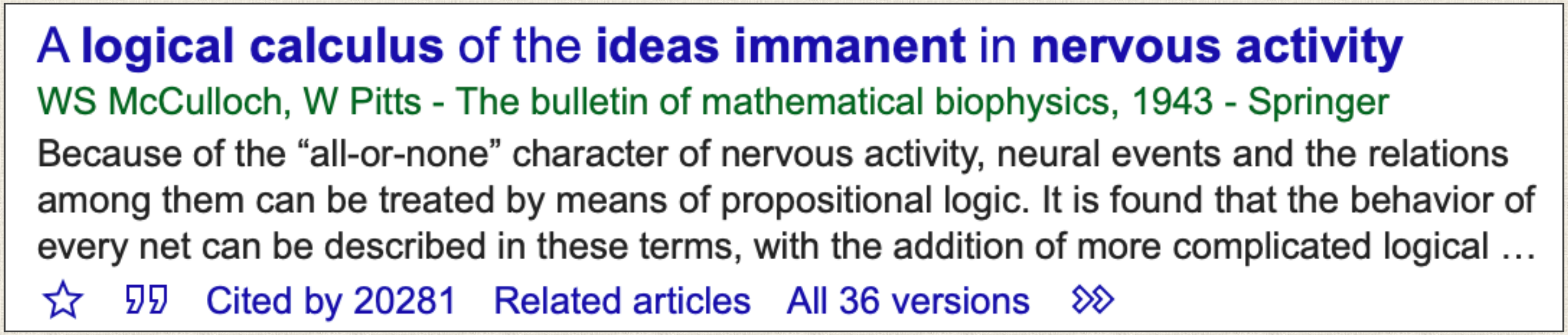
MCCULLOCH-PITTS NEURONS: THE HUMBLE FOUNDATIONS OF AI

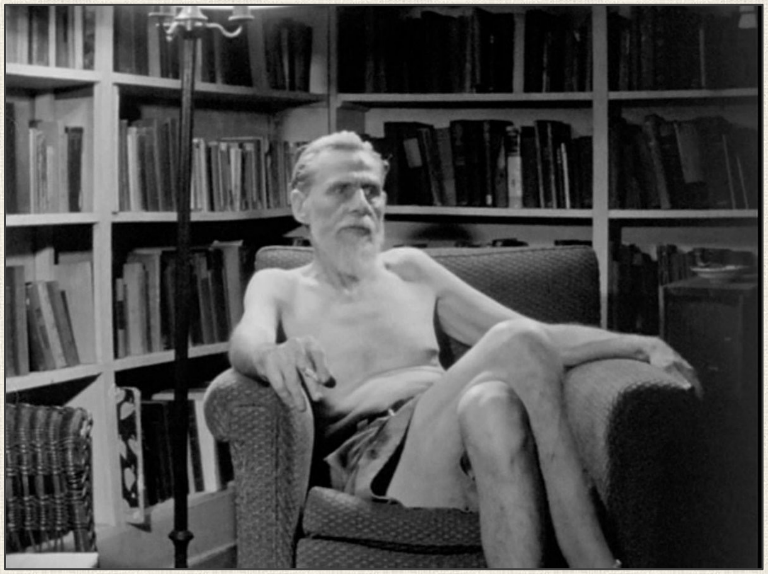
Neurons form a network of cells exchanging information



https://en.wikipedia.org/wiki/Neuron#/media/File:Components_of_neuron.jpg

Hooray for interdisciplinary research


A logical calculus of the ideas immanent in nervous activity
WS McCulloch, W Pitts - The bulletin of mathematical biophysics, 1943 - Springer
Because of the “all-or-none” character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical …
Cited by 20281 Related articles All 36 versions

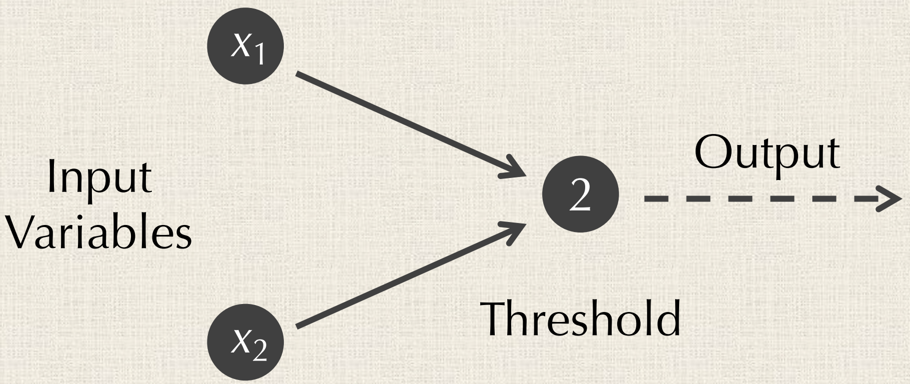


Warren McCulloch

Walter Pitts

McCulloch-Pitts Neurons

A **McCulloch-Pitts (MP) neuron** takes as input n binary variables $x_1, \ldots, x_n$. For a threshold θ, it **fires** (returns 1) if $x_1 + \ldots + x_n \geq \theta$; otherwise, it returns 0.

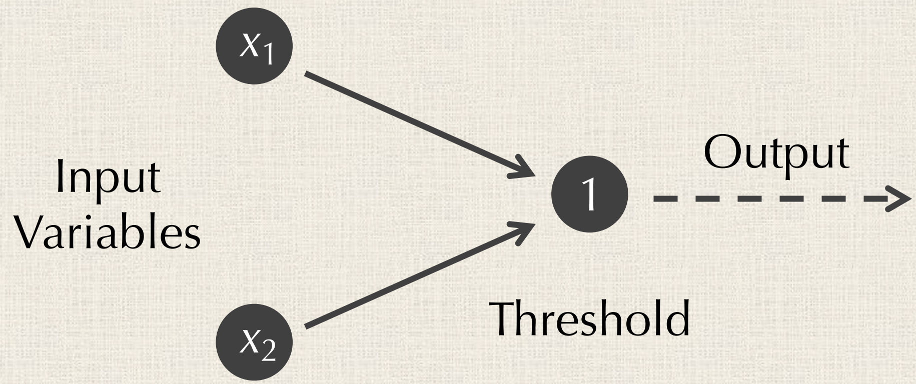


Example: At right is an MP neuron for $n = 2$ and $\theta = 2$.

x_1	x_2	$x_1 + x_2$	**Output**
1	1	2	1
1	0	1	0
0	1	1	0
0	0	0	0

McCulloch-Pitts Neurons

Example: And here is the MP neuron for $n = 2$ and $\theta = 1$.

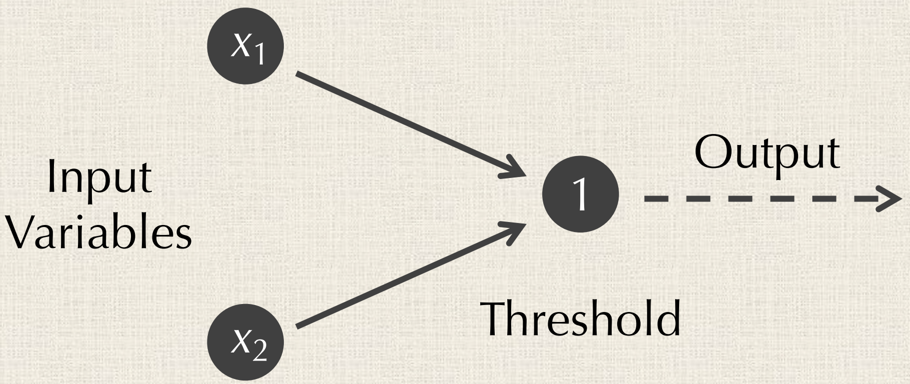


x_1	x_2	$x_1 + x_2$	**Output**
1	1	2	1
1	0	1	**1**
0	1	1	**1**
0	0	0	0

McCulloch-Pitts Neurons

Example: And here is the MP neuron for $n = 2$ and $\theta = 1$.

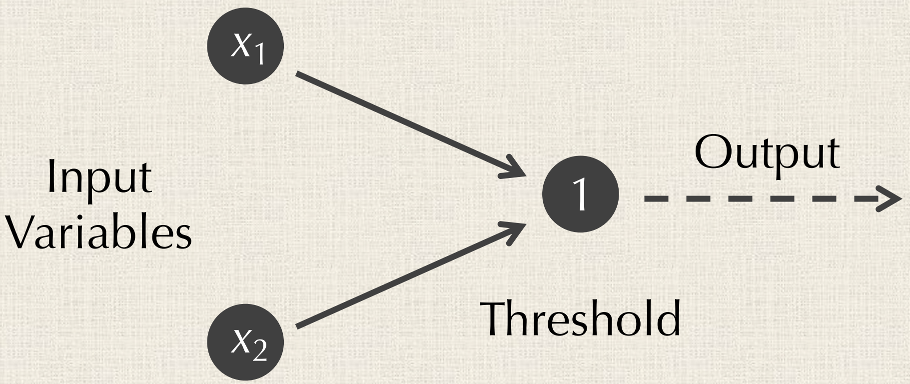
STOP: Do these neurons remind you of anything?



x_1	x_2	$x_1 + x_2$	**Output**
1	1	2	1
1	0	1	1
0	1	1	1
0	0	0	0

McCulloch-Pitts Neurons

Example: And here is the MP neuron for $n = 2$ and $\theta = 1$.



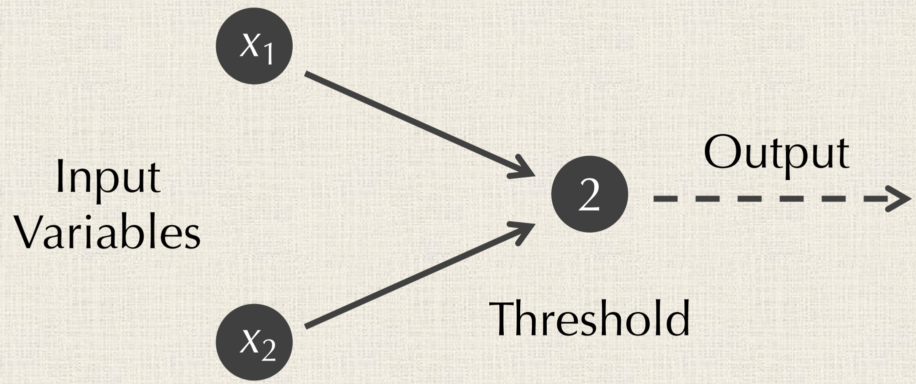
STOP: Do these neurons remind you of anything?

x_1	x_2	$x_1 + x_2$	Output
1	1	2	1
1	0	1	1
0	1	1	1
0	0	0	0

Answer: The output is just $x_1 \vee x_2$.

McCulloch-Pitts Neurons

And the output of the MP neuron when $\theta = 2$ is $x_1 \wedge x_2$.



We say that an MP neuron **represents** a truth table if the inputs and outputs of the neuron and the truth table are the same.

x_1	x_2	$x_1 + x_2$	**Output**
1	1	2	1
1	0	1	0
0	1	1	0
0	0	0	0

A Quick Exercise

Exercise: The `AND` of *n* input variables returns `true` if all of the input variables are `true`, and `false` otherwise; the `OR` of n input variables returns `true` if at least one of them is `true`, and `false` if they are all `false`. Construct MP neurons representing the `AND` and `OR` of *n* binary input variables.

An Even Simpler Logical Connective: NOT

Here is a truth table representing the logical connective `NOT`.

x_1	$\sim x_1$
`true`	`false`
`false`	`true`

An Even Simpler Logical Connective: NOT

Here is a truth table representing the logical connective `NOT`.

x_1	$\sim x_1$
`true`	`false`
`false`	`true`

Theorem: There is no McCulloch-Pitts neuron representing `NOT`.

An Even Simpler Logical Connective: NOT

Here is a truth table representing the logical connective `NOT`.

x_1	$\sim x_1$
`true`	`false`
`false`	`true`

Theorem: There is no McCulloch-Pitts neuron representing `NOT`.

Proof: Assume that there is such an MP neuron with one input variable x_1.

An Even Simpler Logical Connective: NOT

Here is a truth table representing the logical connective `NOT`.

x_1	$\sim x_1$
`true`	`false`
`false`	`true`

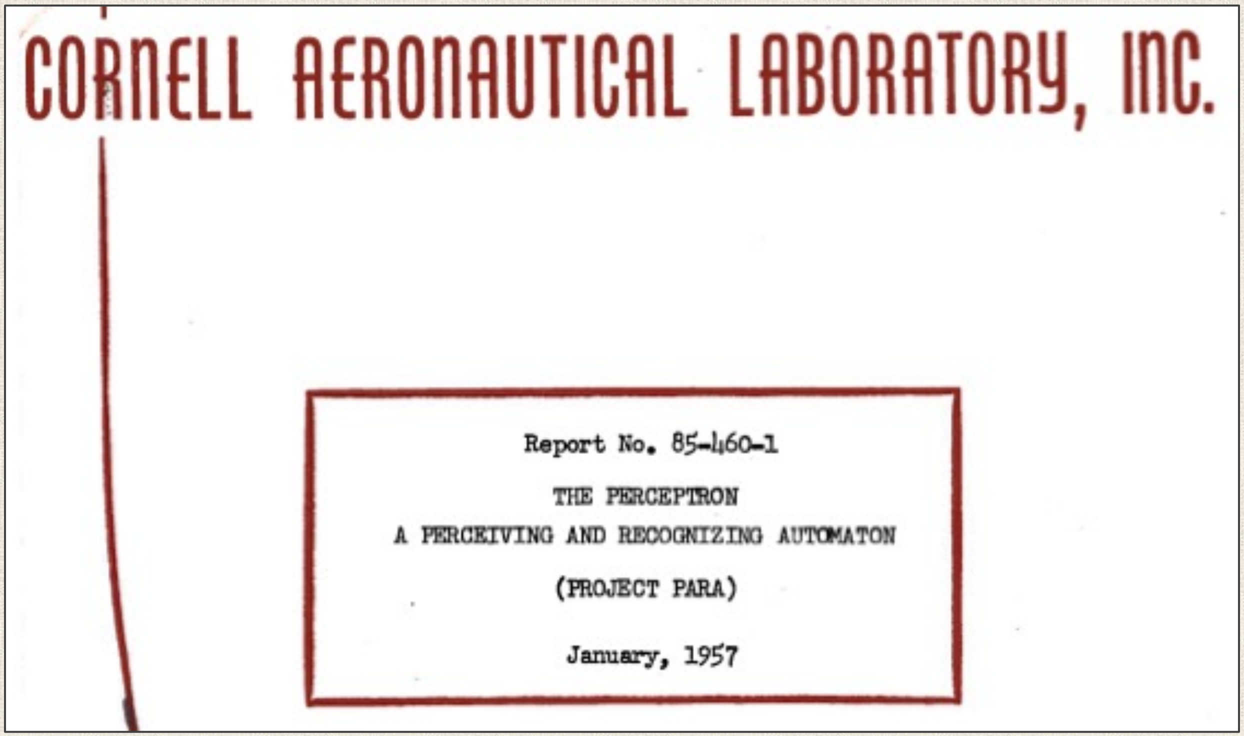
Theorem: There is no McCulloch-Pitts neuron representing `NOT`.

Proof: Assume that there is such an MP neuron with one input variable x_1. There must be some threshold θ such that when $x_1 = 1$, $x_1 < \theta$, and when $x_1 = 0$, $x_1 \geq \theta$. In other words, $1 < \theta \leq 0$, a contradiction. □

FROM MCCULLOCH-PITTS NEURONS TO PERCEPTRONS

Perceptrons Generalize MP Neurons

Perceptron: A neuron having a threshold θ and constants $w_1, w_2, \ldots, w_n$, which fires if and only if $w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \geq \theta$.



CORNELL AERONAUTICAL LABORATORY, INC.

Report No. 85-460-1

THE PERCEPTRON

A PERCEIVING AND RECOGNIZING AUTOMATON

(PROJECT PARA)

January, 1957

Perceptrons Generalize MP Neurons

Perceptron: A neuron having a threshold θ and constants w_1, w_2, ..., w_n, which fires if and only if $w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \geq \theta$.

STOP: Why does a perceptron generalize the MP neuron?

Perceptrons Generalize MP Neurons

Perceptron: A neuron having a threshold θ and constants $w_1, w_2, \ldots, w_n$, which fires if and only if $w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \geq \theta$.

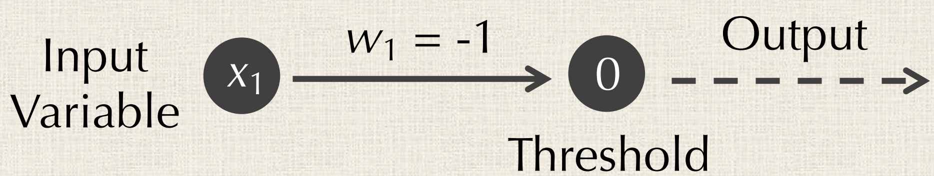
STOP: Why does a perceptron generalize the MP neuron?

Answer: An MP neuron is a perceptron with all weights w_i equal to 1.

Perceptrons Generalize MP Neurons

Perceptron: A neuron having a threshold θ and constants $w_1, w_2, \ldots, w_n$, which fires if and only if $w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \geq \theta$.

Although an MP neuron cannot represent `NOT`, here is a perceptron representing `NOT`.



x_1	$-x_1$	**Output**
1	-1	0
0	0	1

Consider the ambiguity of the word "or"

"Would you like ketchup **or** mustard with your hot dog?"

"Would you like to visit the beach **or** the mountains on vacation?"

Consider the ambiguity of the word “or”

“Would you like ketchup **or** mustard with your hot dog?”

“Would you like to visit the beach **or** the mountains on vacation?”

STOP: What is the difference in “or” in these two questions?

Consider the ambiguity of the word “or”

“Would you like ketchup **or** mustard with your hot dog?”

“Would you like to visit the beach **or** the mountains on vacation?”

STOP: What is the difference in “or” in these two questions?

Answer: The first question implies that *both* options are possible (“and/or”).

Introducing XOR

Exclusive or (XOR): $x_1 \veebar x_2$ is `true` precisely when exactly one of x_1 and x_2 is `true` (i.e., when $x_1 \neq x_2$).

x_1	x_2	$x_1 \vee x_2$	$x_1 \veebar x_2$
`true`	`true`	`true`	`false`
`true`	`false`	`true`	`true`
`false`	`true`	`true`	`true`
`false`	`false`	`false`	`false`

Introducing XOR

Exclusive or (XOR): $x_1 \veebar x_2$ is `true` precisely when exactly one of x_1 and x_2 is `true` (i.e., when $x_1 \neq x_2$).

x_1	x_2	$x_1 \vee x_2$	$x_1 \veebar x_2$
`true`	`true`	`true`	`false`
`true`	`false`	`true`	`true`
`false`	`true`	`true`	`true`
`false`	`false`	`false`	`false`

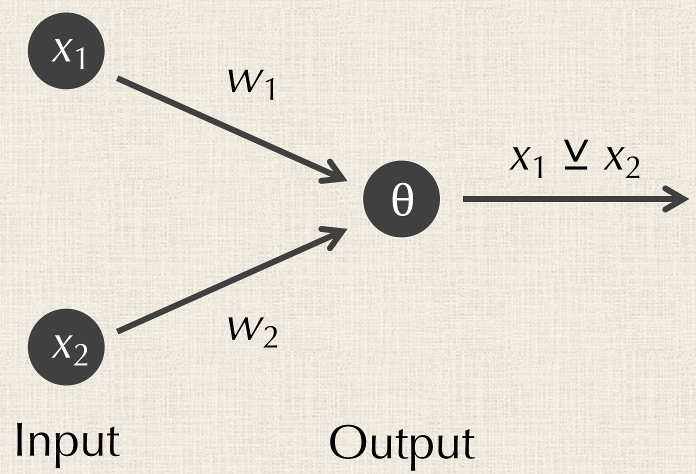
Exercise: Find a perceptron that models $x_1 \veebar x_2$.

Perceptrons have limits too

Theorem: There is no perceptron representing XOR.

Proof: Assume there is, so there must be constants w_1, w_2, such that

- when $x_1 = x_2$, $w_1 \cdot x_1 + w_2 \cdot x_2 < \theta$
- when $x_1 \neq x_2$, $w_1 \cdot x_1 + w_2 \cdot x_2 \geq \theta$

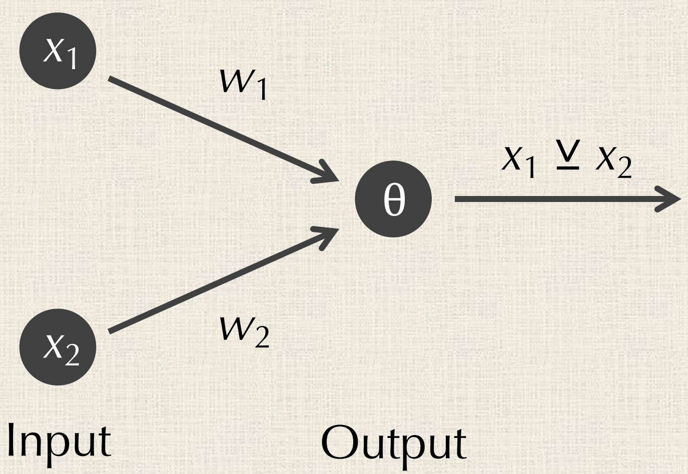


Perceptrons have limits too

Theorem: There is no perceptron representing XOR.

Proof: When $x_1 = x_2$, the neuron doesn't fire, and

$$w_1 \cdot 0 + w_2 \cdot 0 = 0 < \theta$$
$$w_1 \cdot 1 + w_2 \cdot 1 = w_1 + w_2 < \theta$$

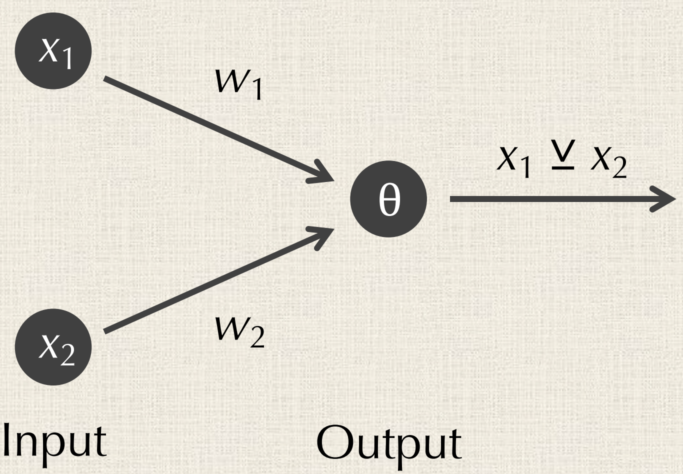


Perceptrons have limits too

Theorem: There is no perceptron representing XOR.

Proof: When $x_1 \neq x_2$, the neuron fires, and

$$w_1 \cdot 1 + w_2 \cdot 0 = w_1 \geq \theta$$
$$w_1 \cdot 0 + w_2 \cdot 1 = w_2 \geq \theta$$



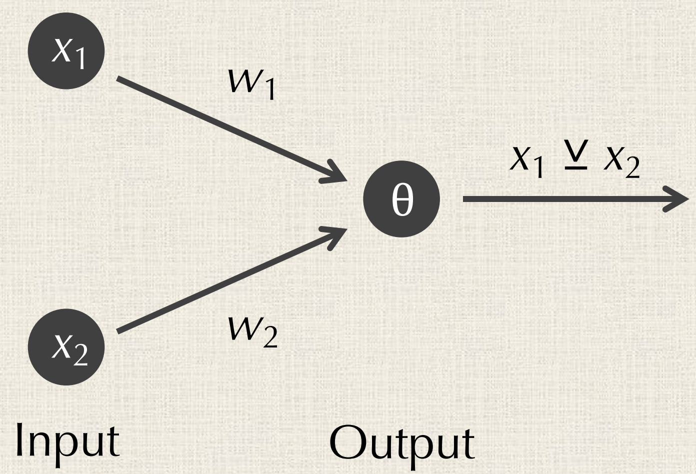
Perceptrons have limits too

Theorem: There is no perceptron representing XOR.

Proof: In summary:

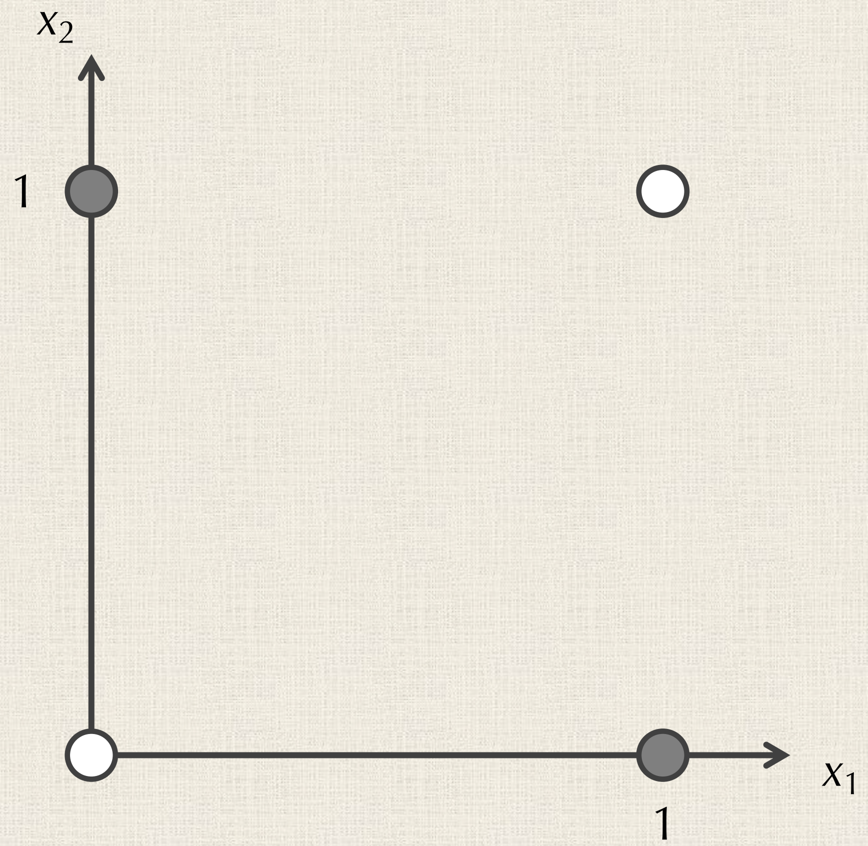
- $w_1 \geq \theta$
- $w_2 \geq \theta$
- $0 < \theta$
- $w_1 + w_2 < \theta$

Adding eqs. 1 and 2 gives $w_1 + w_2 \geq 2\theta$, which contradicts $w_1 + w_2 < \theta$ since θ is positive. □



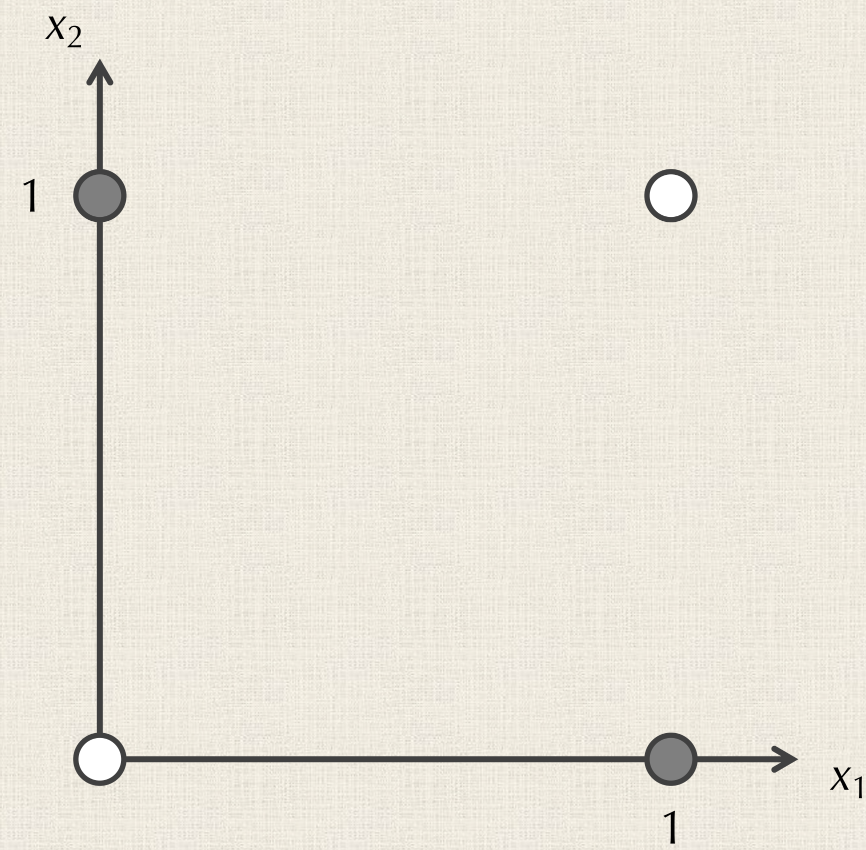
A less rigorous view of this proof

Note: The collection of all points (x_1, x_2) such that $w_1 \cdot x_1 + w_2 \cdot x_2 = \theta$ must form a line. The points such that $w_1 \cdot x_1 + w_2 \cdot x_2 \geq \theta$ fall on one side of this line.



A less rigorous view of this proof

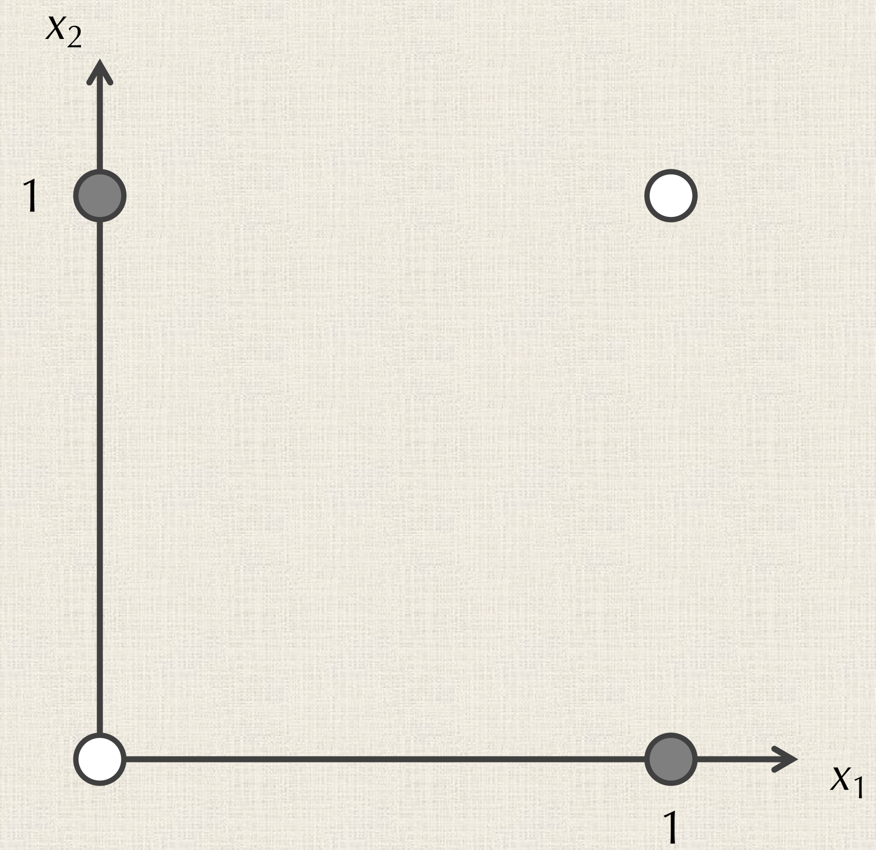
We color the points (x_1, x_2) by whether $x_1 \veebar x_2$ is `true` (black) or `false` (white).



A less rigorous view of this proof

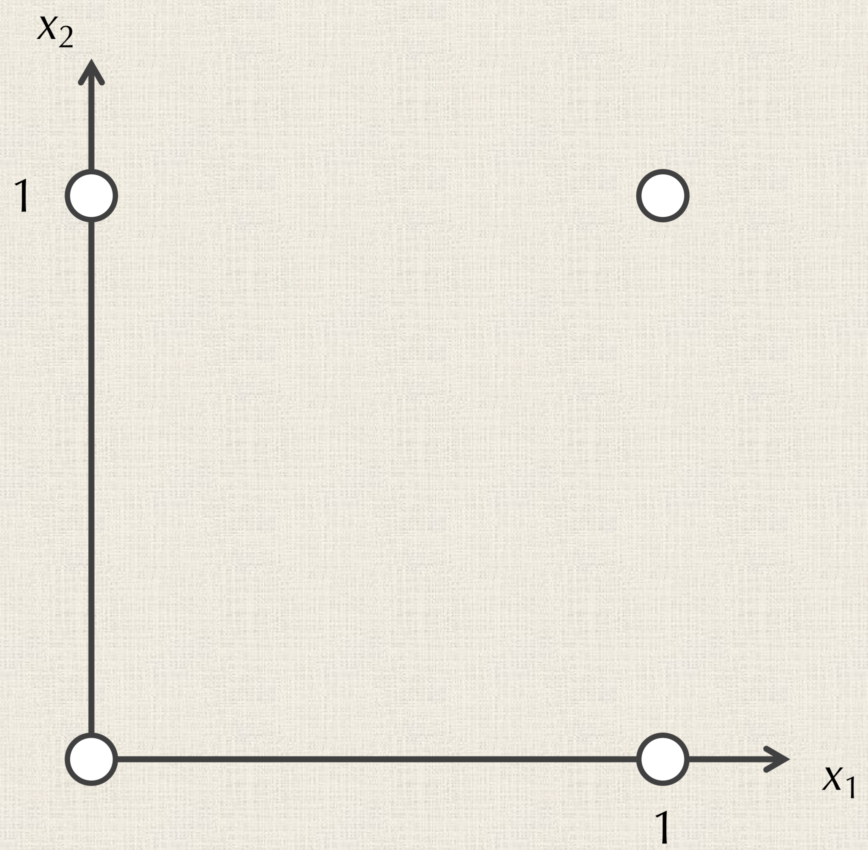
We color the points (x_1, x_2) by whether $x_1 \veebar x_2$ is `true` (black) or `false` (white).

There is no line through the points such that shaded points are on one side; i.e., `XOR` is not **linearly separable**.



Linear Separability of AND and OR

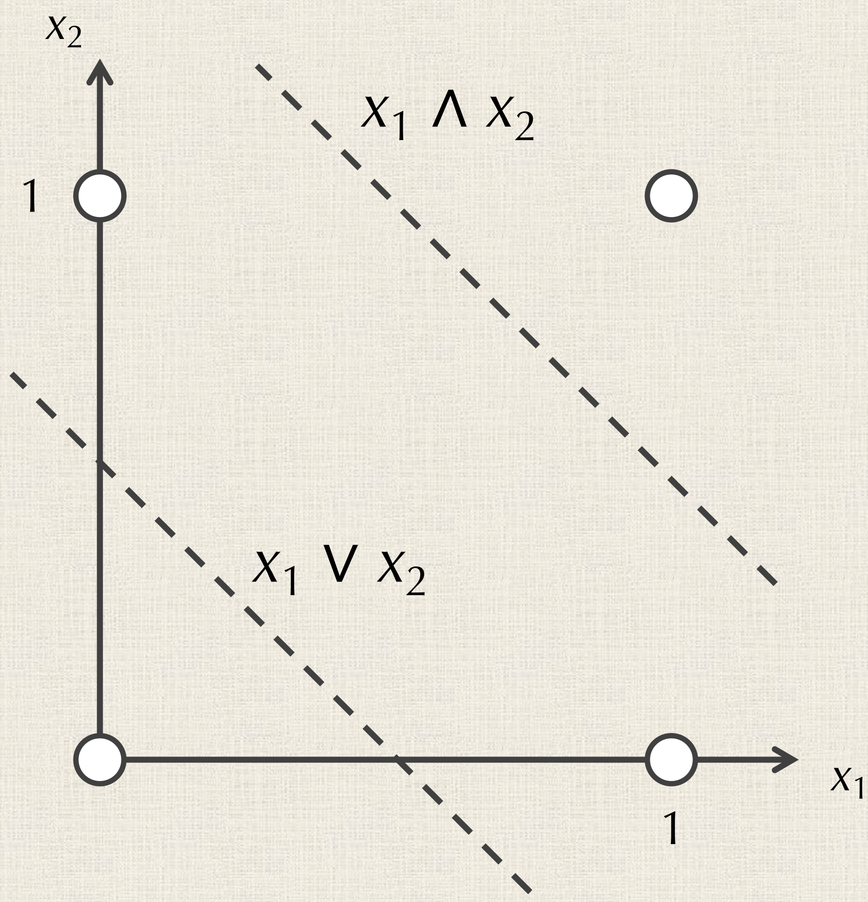
STOP: Draw lines that separate points based on the values of $x_1 \vee x_2$. Do the same for $x_1 \wedge x_2$.



Linear Separability of AND and OR

STOP: Draw lines that separate points based on the values of $x_1 \vee x_2$. Do the same for $x_1 \wedge x_2$.

Answer: Shown at right.

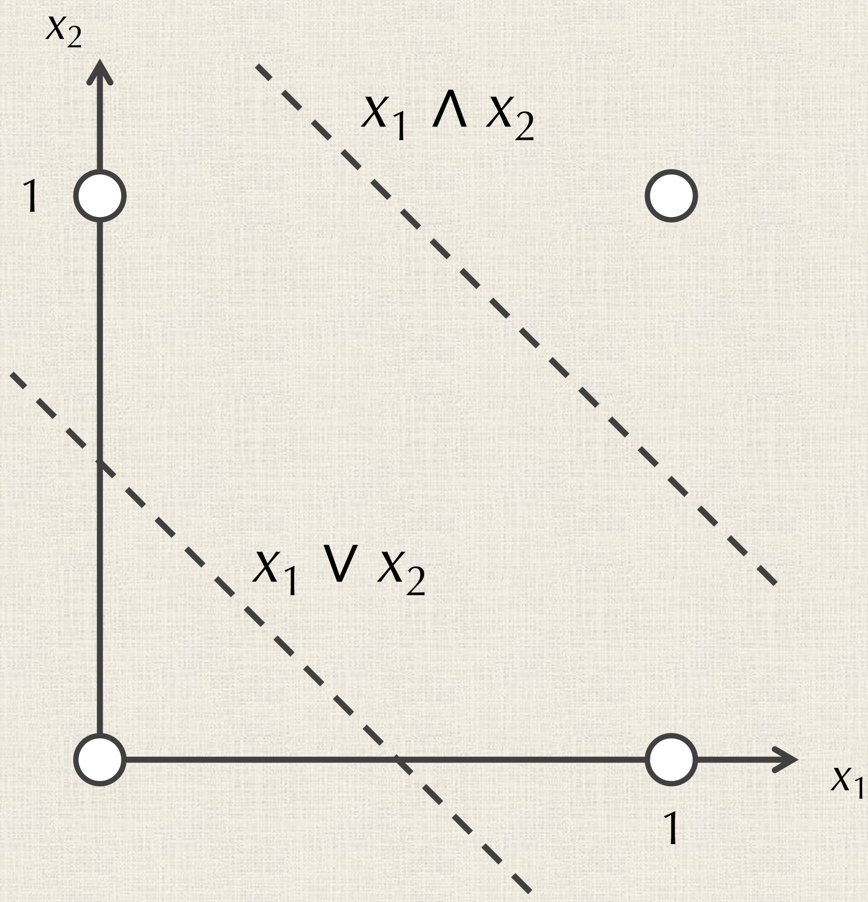


Linear Separability of AND and OR

STOP: Draw lines that separate points based on the values of $x_1 \vee x_2$. Do the same for $x_1 \wedge x_2$.

Answer: Shown at right.

You may be wondering how useful perceptrons can be if they can't model XOR. Sit tight!



A BIT MORE LOGIC

Propositions use logical connectives as building blocks

Proposition: A combination of logical connectives in which outputs of one connective can be used as inputs of another (e.g., $(x_1 \wedge (x_2 \vee \sim x_3)) \veebar (x_4 \vee x_5)$.

Propositions use logical connectives as building blocks

Proposition: A combination of logical connectives in which outputs of one connective can be used as inputs of another (e.g., $(x_1 \wedge (x_2 \vee \sim x_3)) \veebar (x_4 \vee x_5)$.

Example: Truth table below demonstrates one of DeMorgan's Laws: $\sim(x_1 \wedge x_2) \equiv \sim x_1 \vee \sim x_2$.

x_1	x_2	$x_1 \wedge x_2$	$\sim(x_1 \wedge x_2)$	$\sim x_1$	$\sim x_2$	$\sim x_1 \vee \sim x_2$
true	true	true	false	false	false	false
true	false	false	true	false	true	true
false	true	false	true	true	false	true
false	false	false	true	true	true	true

Propositions use logical connectives as building blocks

Note: Here "≡" denotes logical equivalence, meaning that the truth table values are the same.

Example: Truth table below demonstrates one of DeMorgan's Laws: $\sim(x_1 \wedge x_2) \equiv \sim x_1 \vee \sim x_2$.

x_1	x_2	$x_1 \wedge x_2$	$\sim(x_1 \wedge x_2)$	$\sim x_1$	$\sim x_2$	$\sim x_1 \vee \sim x_2$
true	true	true	false	false	false	false
true	false	false	true	false	true	true
false	true	false	true	true	false	true
false	false	false	true	true	true	true

Propositions use logical connectives as building blocks

The expression $\sim(x_1 \wedge x_2)$ is so common that it has its own connective, **NAND** (“not **AND**”): $x_1 \uparrow x_2$.

Example: Truth table below demonstrates one of DeMorgan’s Laws: $\sim(x_1 \wedge x_2) \equiv \sim x_1 \vee \sim x_2$.

x_1	x_2	$x_1 \wedge x_2$	$\sim(x_1 \wedge x_2)$	$\sim x_1$	$\sim x_2$	$\sim x_1 \vee \sim x_2$
true	true	true	false	false	false	false
true	false	false	true	false	true	true
false	true	false	true	true	false	true
false	false	false	true	true	true	true

Let's do a couple of exercises!

The expression $\sim(x_1 \wedge x_2)$ is so common that it has its own connective, **NAND** ("not **AND**"): $x_1 \uparrow x_2$.

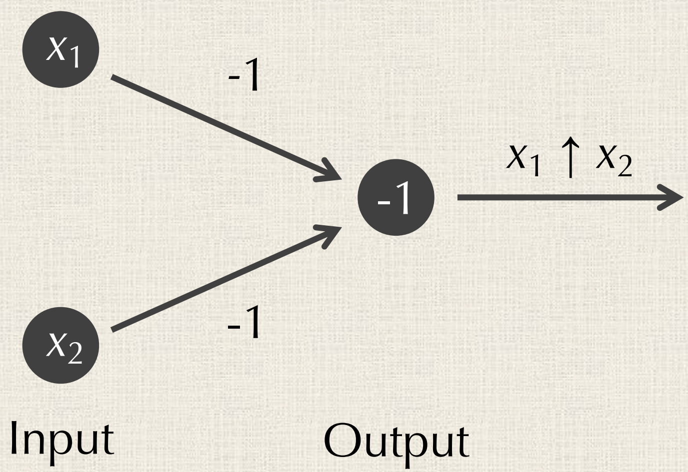
Exercise 1: Find a perceptron representing $x_1 \uparrow x_2$.

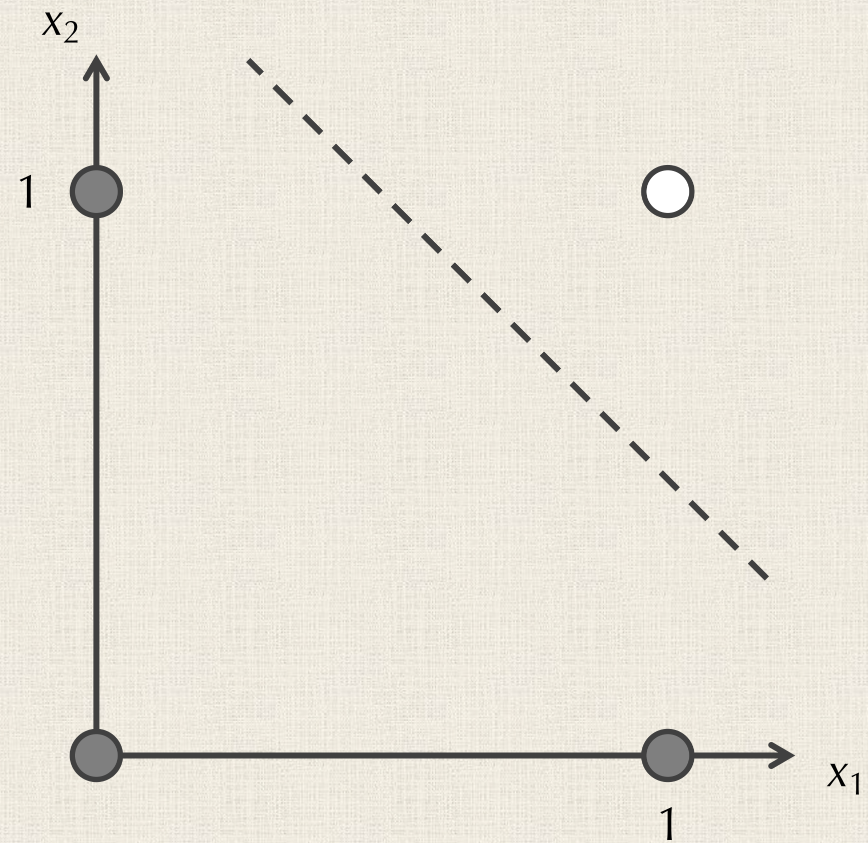
Exercise 2: Find a proposition using connectives other than $\veebar$ that is logically equivalent to $x_1 \veebar x_2$.

LINKING PERCEPTRONS INTO NEURAL NETWORKS MAKES THEM MORE POWERFUL

One solution to exercise 1

Exercise 1: Find a perceptron representing $x_1 \uparrow x_2$.





One solution to exercise 2

Exercise 2: Find a proposition using connectives other than $\veebar$ that is logically equivalent to $x_1 \veebar x_2$.

One common solution is that $x_1 \veebar x_2 \equiv (x_1 \vee x_2) \wedge (\sim x_1 \vee \sim x_2)$, which in turn is just $(x_1 \vee x_2) \wedge (x_1 \uparrow x_2)$.

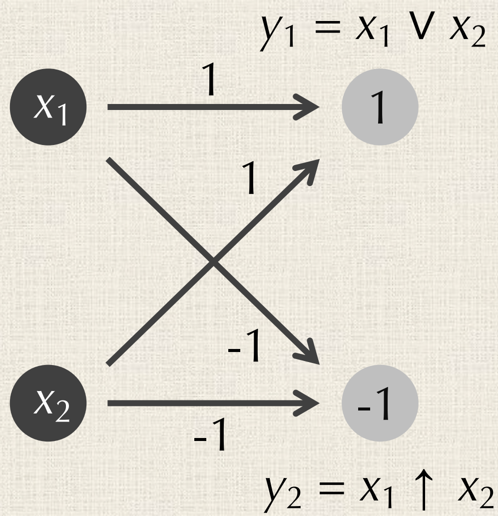
One solution to exercise 2

Exercise 2: Find a proposition using connectives other than $\veebar$ that is logically equivalent to $x_1 \veebar x_2$.

One common solution is that $x_1 \veebar x_2 \equiv (x_1 \vee x_2) \wedge (\sim x_1 \vee \sim x_2)$, which in turn is just $(x_1 \vee x_2) \wedge (x_1 \uparrow x_2)$.

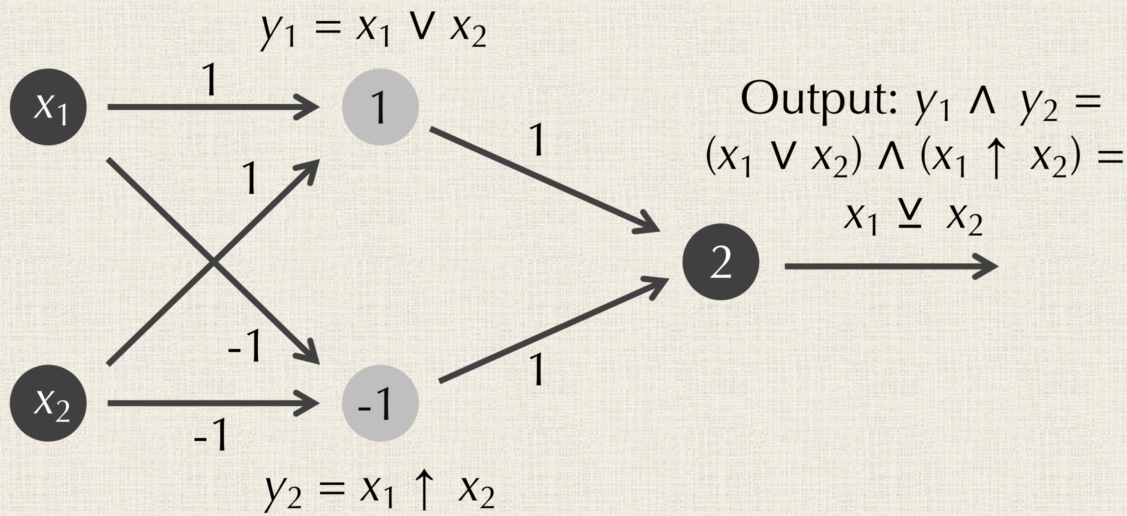
Note: Although we don't have a perceptron representing $\veebar$, we *do* have perceptrons representing $\vee$, $\wedge$, and $\uparrow$ …

Constructing a *network* of perceptrons representing $x_1 \veebar x_2$



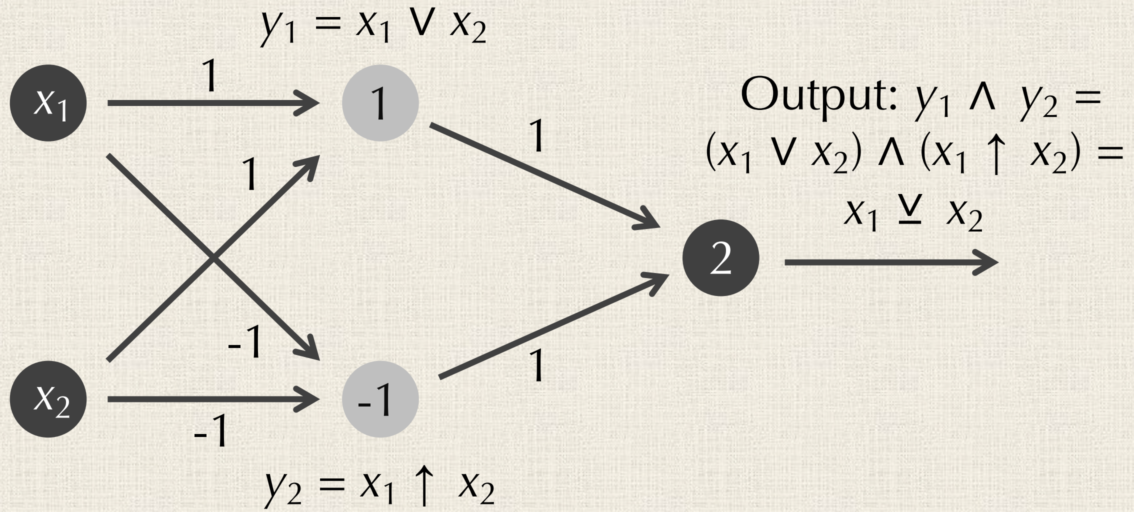
x_1	x_2	$x_1 + x_2$	y_1	$-x_1 - x_2$	y_2
1	1	2	1	-2	0
1	0	1	1	-1	1
0	1	1	1	-1	1
0	0	0	0	0	1

Constructing a *network* of perceptrons representing $x_1 \veebar x_2$



x_1	x_2	$x_1 + x_2$	y_1	$-x_1 - x_2$	y_2	$y_1 + y_2$	**Output**
1	1	2	1	-2	0	1	0
1	0	1	1	-1	1	2	1
0	1	1	1	-1	1	2	1
0	0	0	0	0	1	1	0

Constructing a *network* of perceptrons representing $x_1 \veebar x_2$



Neural network: a network of artificial neurons in which neuron outputs are inputs into other neurons. The above network has a single **hidden layer** of neurons (gray) that are not input variables or output.

THE UNIVERSALITY OF PERCEPTRON NEURAL NETWORKS

Binary Functions

Binary function: a function having n binary variables as input and producing a binary output.

Example: $f(0,0) = 1$; $f(0,1) = 1$; $f(1,0) = 0$; $f(1,1) = 1$.

STOP: How many different binary functions are there with n input variables?

Binary Functions

Binary function: a function having n binary variables as input and producing a binary output.

Example: $f(0,0) = 1$; $f(0,1) = 1$; $f(1,0) = 0$; $f(1,1) = 1$.

STOP: How many different binary functions are there with n input variables?

Answer: There are 2^n different possible inputs. Each input can produce a 1 or 0; therefore, there are 2^{2^n} total binary functions.

Our building blocks can be used to build *any* binary function

Note: this binary function can be represented by the proposition $\sim x_1 \vee x_2$, with 1 = `true` and 0 = `false`.

Example: $f(0,0) = 1$; $f(0,1) = 1$; $f(1,0) = 0$; $f(1,1) = 1$.

Our building blocks can be used to build *any* binary function

Note: this binary function can be represented by the proposition $\sim x_1 \vee x_2$, with 1 = `true` and 0 = `false`.

Example: $f(0,0) = 1$; $f(0,1) = 1$; $f(1,0) = 0$; $f(1,1) = 1$.

Theorem: Any binary function can be represented by some proposition formed by a finite number of the logical connectives $\wedge$, $\vee$, and $\sim$.

Our building blocks can be used to build *any* binary function

Note: this binary function can be represented by the proposition $\sim x_1 \vee x_2$, with 1 = `true` and 0 = `false`.

Example: $f(0,0) = 1$; $f(0,1) = 1$; $f(1,0) = 0$; $f(1,1) = 1$.

Theorem: Any binary function can be represented by some proposition formed by a finite number of the logical connectives $\wedge$, $\vee$, and $\sim$.

Key point: All these connectives can be represented by single perceptrons…

Our building blocks can be used to build *any* binary function

Note: this binary function can be represented by the proposition $\sim x_1 \vee x_2$, with 1 = `true` and 0 = `false`.

Example: $f(0,0) = 1$; $f(0,1) = 1$; $f(1,0) = 0$; $f(1,1) = 1$.

Corollary: Any binary function can be represented by a neural network of finitely many perceptrons.

Key point: All these connectives can be represented by single perceptrons…

The only building block we need is NAND

Recall that $\sim(x_1 \land x_2)$ is abbreviated as $x_1 \uparrow x_2$.

x_1	x_2	$x_1 \uparrow x_2$
1	1	0
1	0	1
0	1	1
0	0	1

The only building block we need is NAND

Recall that $\sim(x_1 \wedge x_2)$ is abbreviated as $x_1 \uparrow x_2$.

x_1	x_2	$x_1 \uparrow x_2$
1	1	0
1	0	1
0	1	1
0	0	1

Theorem: Any binary function can be represented by some proposition formed exclusively by a finite number of $\uparrow$ connectors.

The only building block we need is NAND

Recall that $\sim(x_1 \wedge x_2)$ is abbreviated as $x_1 \uparrow x_2$.

x_1	x_2	$x_1 \uparrow x_2$
1	1	0
1	0	1
0	1	1
0	0	1

Proof: We will show that each of the expressions $\sim x_1$, $(x_1 \wedge x_2)$, and $(x_1 \vee x_2)$ can be represented with just NAND ($\uparrow$) connectors.

The only building block we need is NAND

Recall that $\sim(x_1 \wedge x_2)$ is abbreviated as $x_1 \uparrow x_2$.

$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_1 \uparrow x_2}$
1	1	0
1	0	1
0	1	1
0	0	1

Proof: We will show that each of the expressions $\sim x_1$, $(x_1 \wedge x_2)$, and $(x_1 \vee x_2)$ can be represented with just NAND ($\uparrow$) connectors.

STOP: Find a proposition formed only of $\uparrow$ connectors that is logically equivalent to $\sim x_1$.

The only building block we need is NAND

x_1	x_2	$x_1 \uparrow x_2$
1	1	0
1	0	1
0	1	1
0	0	1

Recall that $\sim(x_1 \wedge x_2)$ is abbreviated as $x_1 \uparrow x_2$.

Proof: We will show that each of the expressions $\sim x_1$, $(x_1 \wedge x_2)$, and $(x_1 \vee x_2)$ can be represented with just NAND ($\uparrow$) connectors.

Answer: $\sim x_1 \equiv x_1 \uparrow x_1$.

The only building block we need is NAND

Recall that $\sim(x_1 \wedge x_2)$ is abbreviated as $x_1 \uparrow x_2$.

x_1	x_2	$x_1 \uparrow x_2$
1	1	0
1	0	1
0	1	1
0	0	1

Proof: We will show that each of the expressions $\sim x_1$, $(x_1 \wedge x_2)$, and $(x_1 \vee x_2)$ can be represented with just NAND ($\uparrow$) connectors.

Exercise: Find propositions of $\uparrow$ connectors that are logically equivalent to $(x_1 \wedge x_2)$ and $(x_1 \vee x_2)$.

The only building block we need is NAND

x_1	x_2	$x_1 \wedge x_2$	$x_1 \uparrow x_2$	$(x_1 \uparrow x_2) \uparrow (x_1 \uparrow x_2)$
1	1	**1**	0	**1**
1	0	**0**	1	**0**
0	1	**0**	1	**0**
0	0	**0**	1	**0**

x_1	x_2	$x_1 \vee x_2$	$x_1 \uparrow x_1$	$x_2 \uparrow x_2$	$(x_1 \uparrow x_1) \uparrow (x_2 \uparrow x_2)$
1	1	**1**	0	0	**1**
1	0	**1**	0	1	**1**
0	1	**1**	1	0	**1**
0	0	**0**	1	1	**0**

The only building block we need is NAND

Recall that $\sim(x_1 \land x_2)$ is abbreviated as $x_1 \uparrow x_2$.

$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_1 \uparrow x_2}$
1	1	0
1	0	1
0	1	1
0	0	1

Theorem: Any binary function can be represented by some proposition formed exclusively by a finite number of $\uparrow$ connectors.

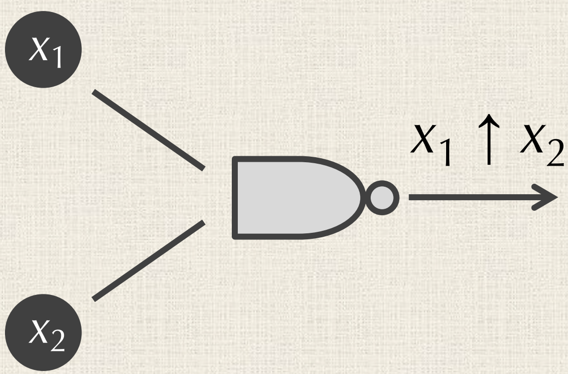
STOP: Now that we have proven this theorem, what is the corollary?

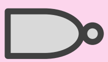
The only building block we need is NAND

x_1	x_2	$x_1 \uparrow x_2$
1	1	0
1	0	1
0	1	1
0	0	1

Recall that $\sim(x_1 \wedge x_2)$ is abbreviated as $x_1 \uparrow x_2$.

Corollary: Any binary function can be represented by a neural network of NAND perceptrons.



Note:  is called a NAND gate.

MODELING THE EVOLUTION OF BIOLOGICAL MODULARITY

Returning to our original question

Can we build a (simple) model in which modularity spontaneously evolves as an optimal solution?

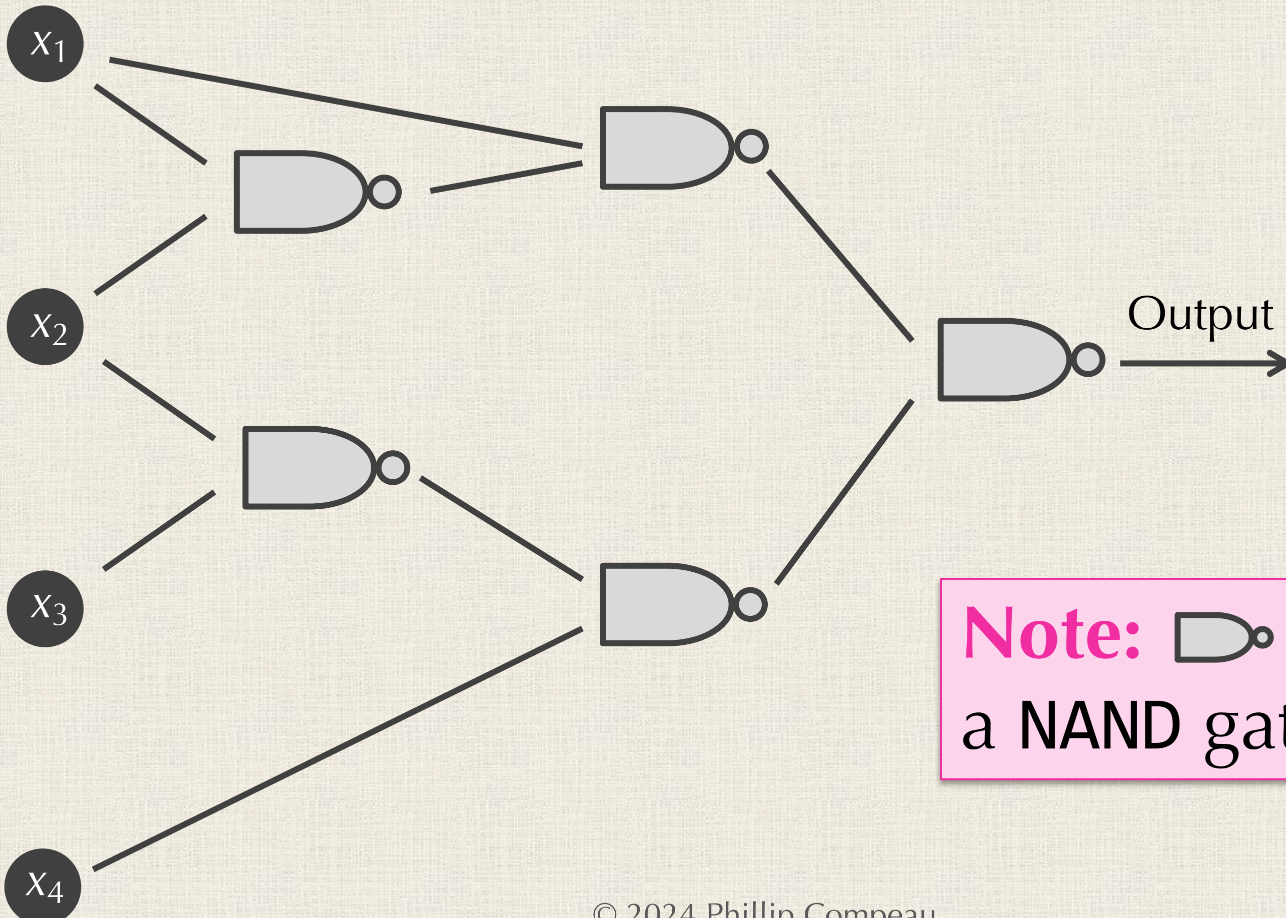
https://www.pnas.org › content

Spontaneous evolution of modularity and network motifs | PNAS

by N Kashtan · 2005 · Cited by 899 — Nadav **Kashtan** and Uri **Alon** ... To understand the origin of **modularity** and network motifs in biology one has to understand how these features ...

The Kashtan-Alon Model

Organisms: all 4-input networks of **NAND** perceptrons



Note: is called a **NAND** gate.

The Kashtan-Alon Model

Organisms: all 4-input networks of `NAND` perceptrons

Goal (G): correctly "compute" as many inputs as possible for the proposition $(x_1 \veebar x_2) \wedge (x_3 \veebar x_4)$.

The Kashtan-Alon Model

Organisms: all 4-input networks of `NAND` perceptrons

Goal (G): correctly "compute" as many inputs as possible for the proposition $(x_1 \veebar x_2) \wedge (x_3 \veebar x_4)$.

STOP: How many different choices of input are there for this proposition?

The Kashtan-Alon Model

Organisms: all 4-input networks of **NAND** perceptrons

Goal (G): correctly "compute" as many inputs as possible for the proposition $(x_1 \veebar x_2) \wedge (x_3 \veebar x_4)$.

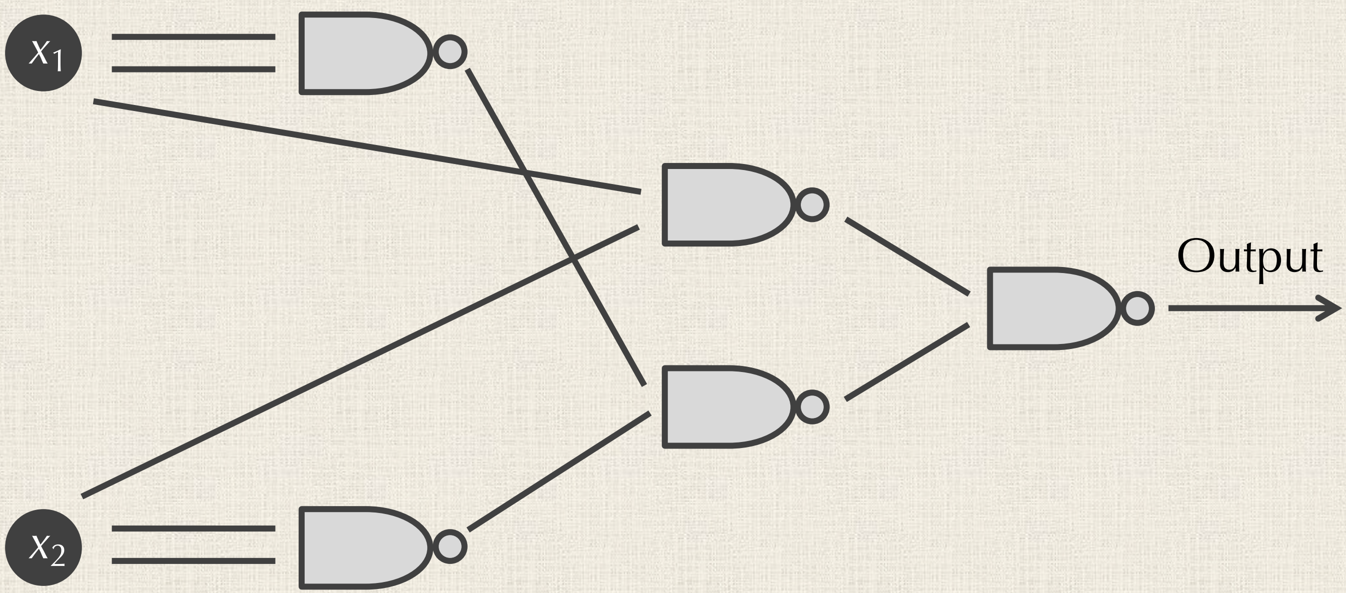
STOP: How many different choices of input are there for this proposition?

Answer: Two possibilities for each variable, so $2^4 = 16$.

One way of reaching the goal

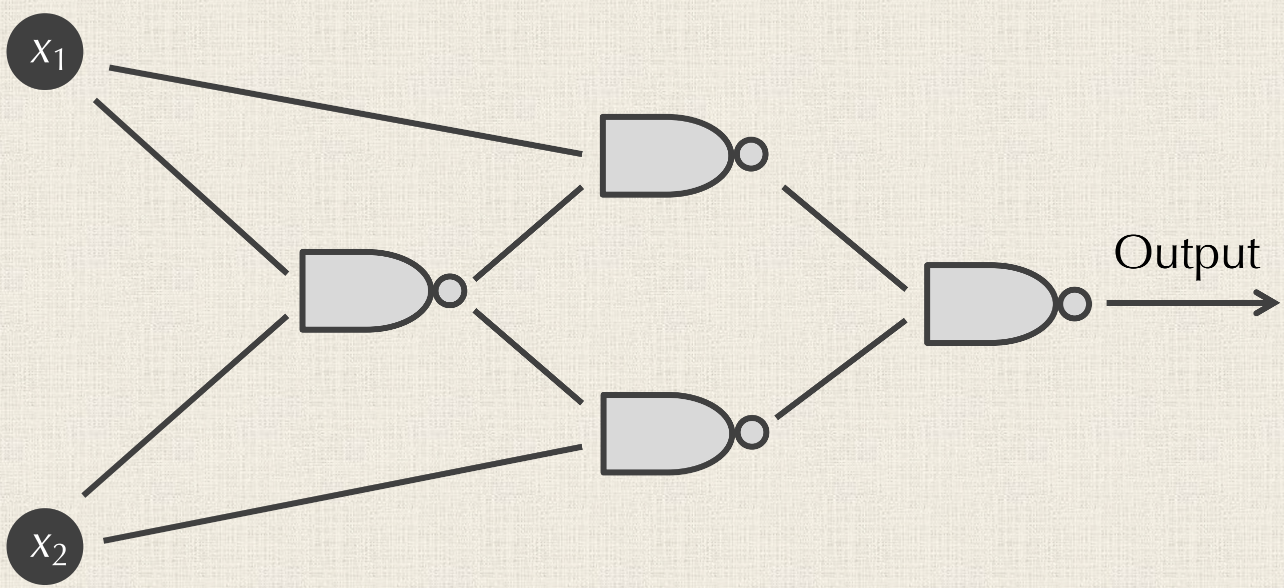
Recall that $(x_1 \veebar x_2) \equiv (x_1 \vee x_2) \wedge (x_1 \uparrow x_2)$.

By the theorem from previously, there is some neural network of **NAND** gates for $(x_1 \vee x_2) \wedge (x_1 \uparrow x_2)$.



One way of reaching the goal

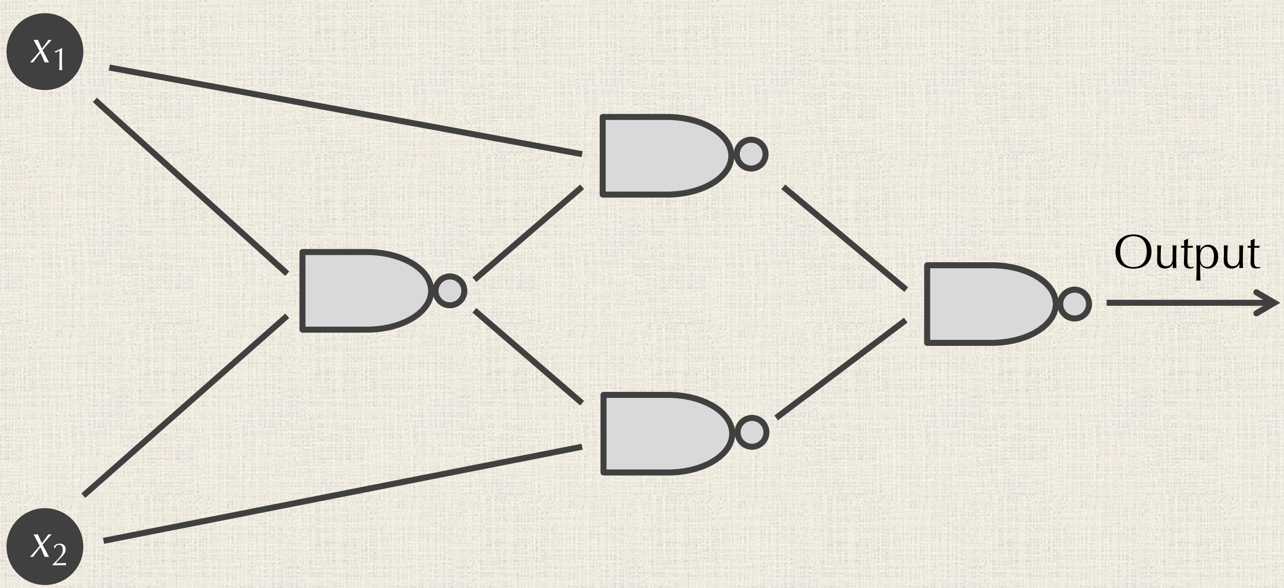
And yet there is a simpler network for $x_1 \veebar x_2$, which is $[x_1 \uparrow (x_1 \uparrow x_2)] \uparrow [x_2 \uparrow (x_1 \uparrow x_2)]$, as shown below.



One way of reaching the goal

And yet there is a simpler network for $x_1 \veebar x_2$, which is $[x_1 \uparrow (x_1 \uparrow x_2)] \uparrow [x_2 \uparrow (x_1 \uparrow x_2)]$, as shown below.

Key point: we should prioritize this smaller network because it would be easier to have evolved.



One way of reaching the goal

And yet there is a simpler network for $x_1 \veebar x_2$, which is $[x_1 \uparrow (x_1 \uparrow x_2)] \uparrow [x_2 \uparrow (x_1 \uparrow x_2)]$, as shown below.

Key point: we should prioritize this smaller network because it would be easier to have evolved.

To prefer a smaller network over a larger network, Kashtan and Alon defined a **fitness function** for a network as the fraction of the 16 input assignments whose output matches the goal G, minus a small positive ε times the number m of NAND gates.

The Kashtan-Alon Algorithm

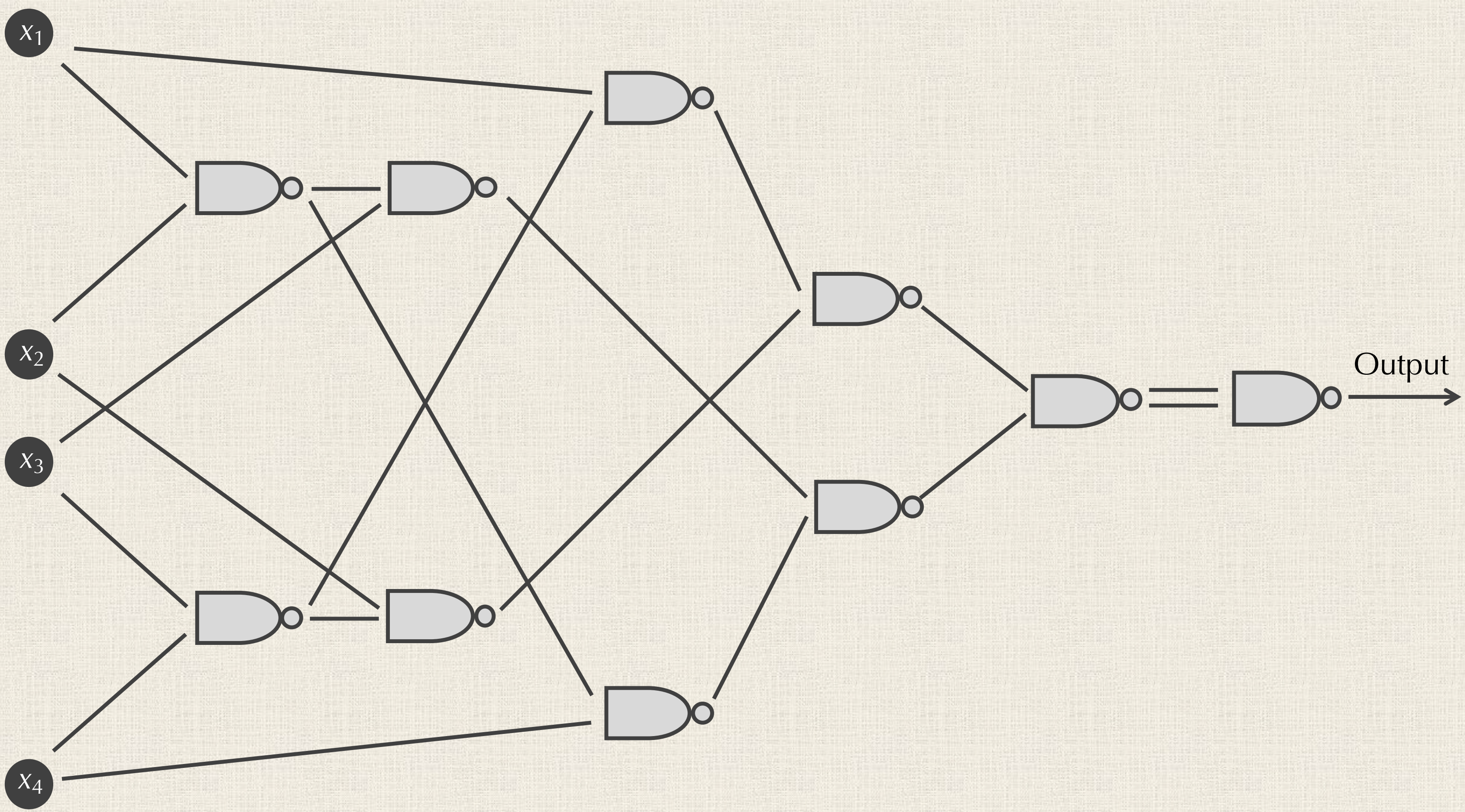
1. Construct 100 random initial networks.
2. Run the following algorithm for 10,000 "generations".
   1. Consider only the 50 networks having highest fitness.
   2. Use these networks to produce 100 "children" networks that have mutations compared to the parent networks.
3. At the end, return the network(s) having maximum fitness as the winner(s).

The Kashtan-Alon Algorithm

1. Construct 100 random initial networks.
2. Run the following algorithm for 10,000 "generations".
   1. Consider only the 50 networks having highest fitness.
   2. Use these networks to produce 100 "children" networks that have mutations compared to the parent networks.
3. At the end, return the network(s) having maximum fitness as the winner(s).

This type of search heuristic, which mimics evolution, is called a **genetic algorithm**.

Our winner isn't very modular… ☹



Life changes, and fitness should change too

Key point: a more realistic model of a competitive landscape would use a *variable* fitness function.

Life changes, and fitness should change too

Key point: a more realistic model of a competitive landscape would use a *variable* fitness function.

Previous goal (G): correctly "compute" as many inputs as possible for $(x_1 \veebar x_2) \wedge (x_3 \veebar x_4)$.

Life changes, and fitness should change too

Key point: a more realistic model of a competitive landscape would use a *variable* fitness function.

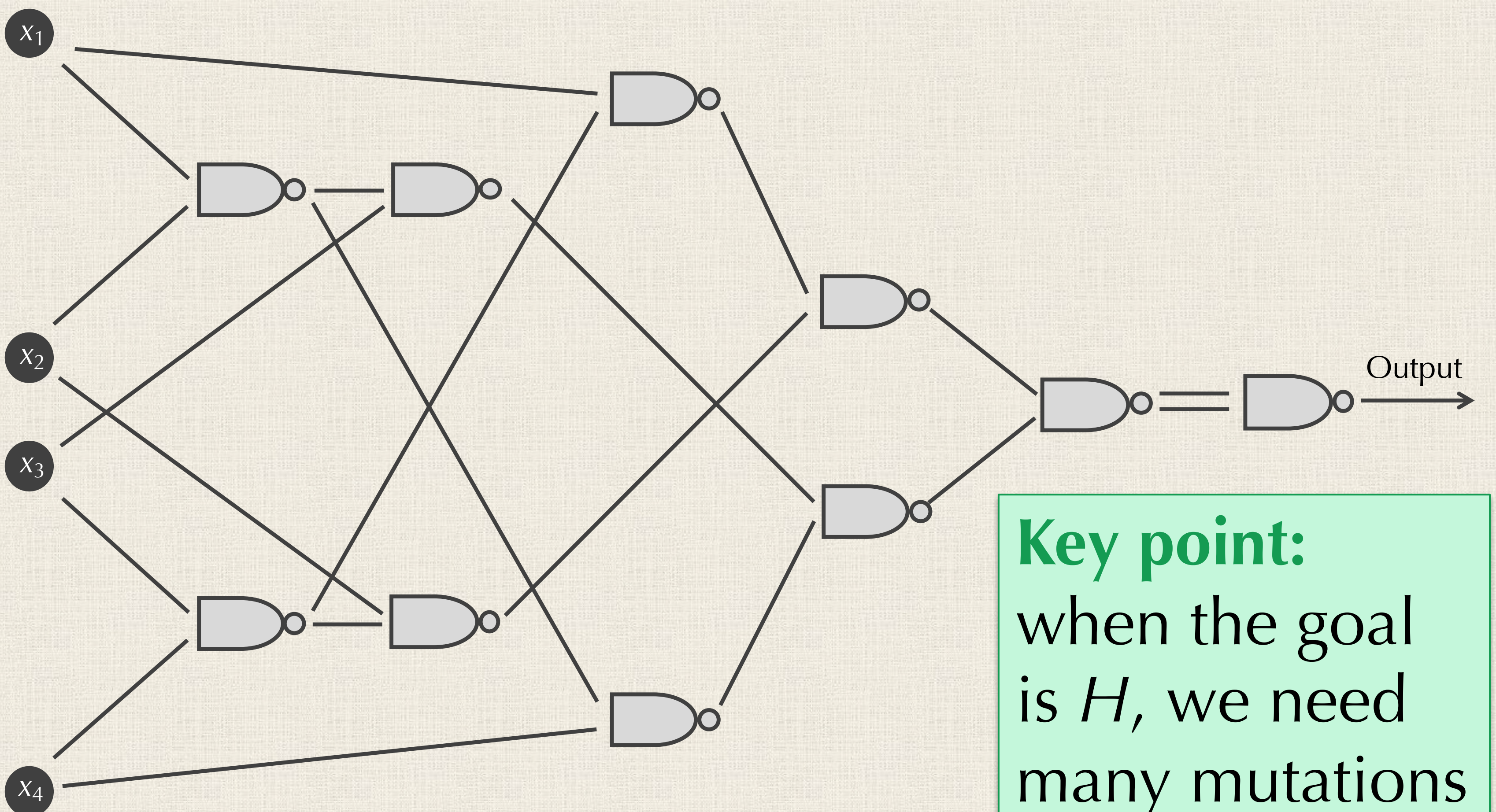
Previous goal (G): correctly "compute" as many inputs as possible for $(x_1 \veebar x_2) \wedge (x_3 \veebar x_4)$.

Alternate goal (H): correctly "compute" as many inputs as possible for $(x_1 \veebar x_2) \vee (x_3 \veebar x_4)$.

Adapting the algorithm to incorporate *variable* fitness

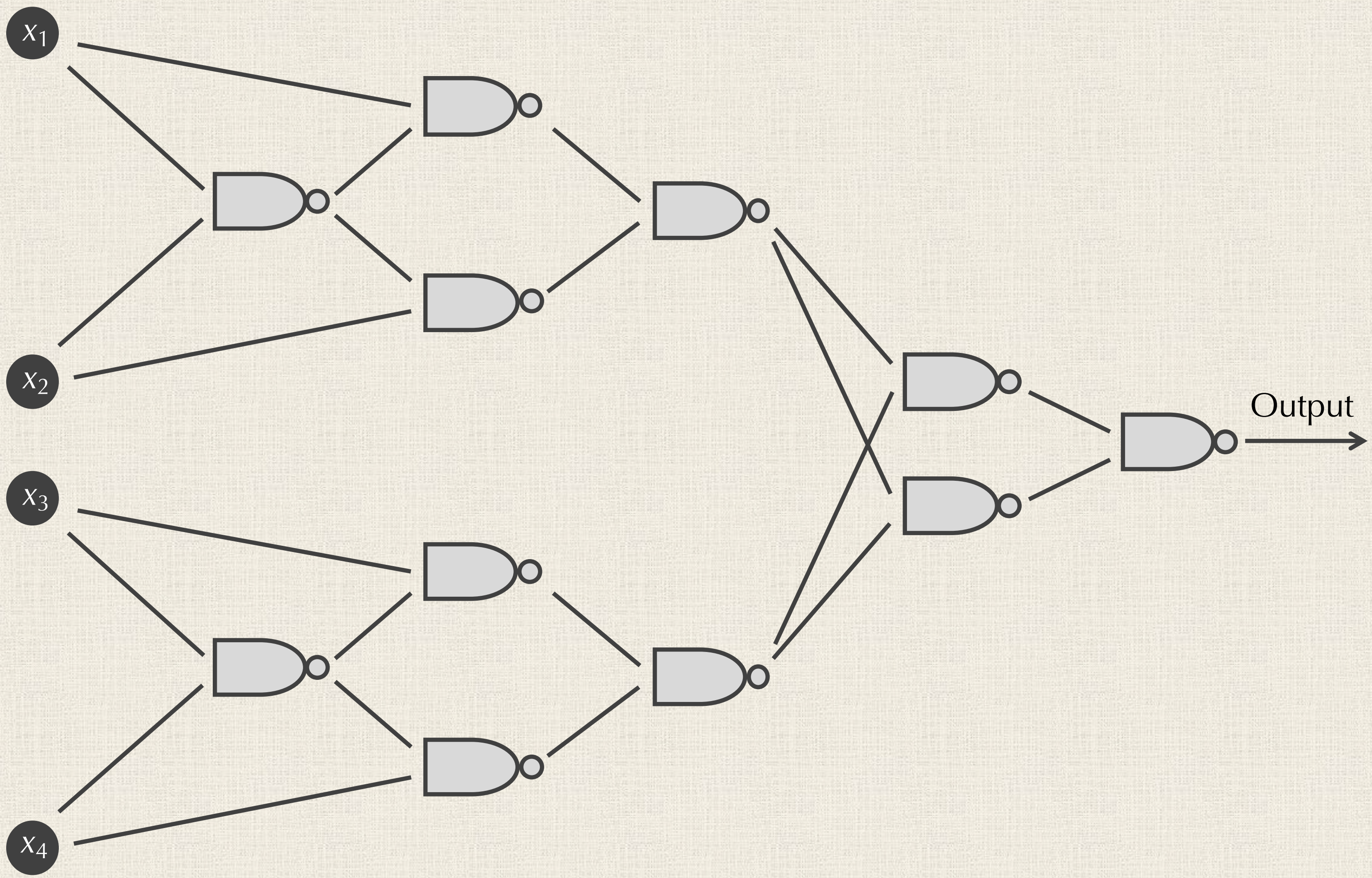
1. Construct 100 random initial networks.
2. Run the following algorithm for 10,000 "generations".
   1. Consider only the 50 networks having highest fitness.
   2. Use these networks to produce 100 "children" networks that have mutations compared to the parent networks.
   3. Every e generations ($e = 20$ in original paper), switch the goal function from G to H or vice-versa.
3. At the end, return the network(s) having maximum fitness as the winner(s).

With the static goal G, we found a non-modular solution

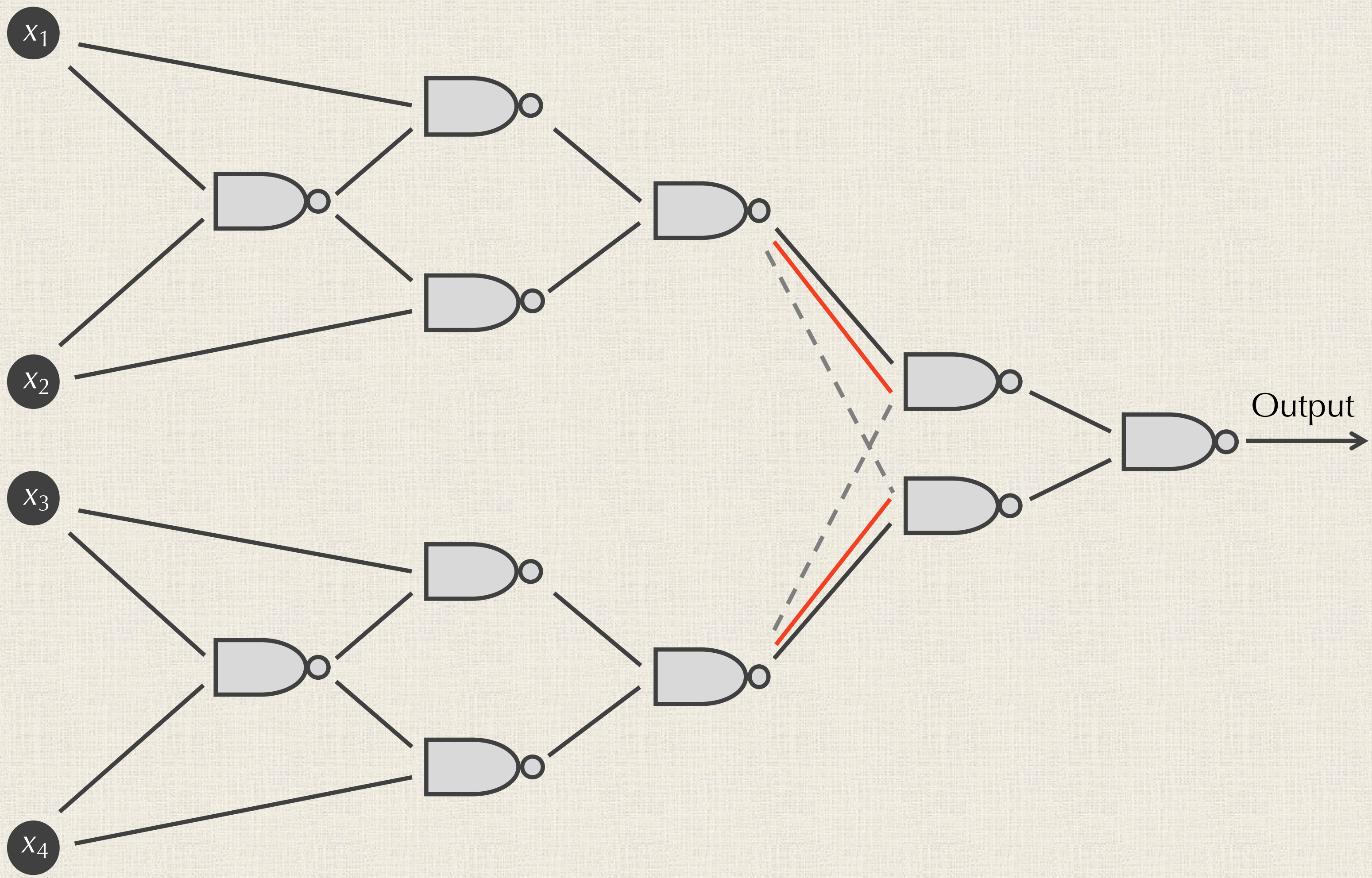


Key point: when the goal is H, we need many mutations to this network.

Dynamic fitness leads to a modular solution to G in ~5000 generations



Switching the goal to H yields a very slightly different modular solution



A great idea leads to more questions

1. What is the extent to which real fitness functions reward modularity?
2. What are the limits of modularity in biology?
3. And what happens when we start building models of consciousness that are more advanced than the neural networks presented here?

EPILOGUE: ~~PRACTICAL APPLICATIONS OF NEURAL NETWORKS~~ AI MAGIC IN 20 MINUTES

Many problems can be framed as classification

Classification Problem

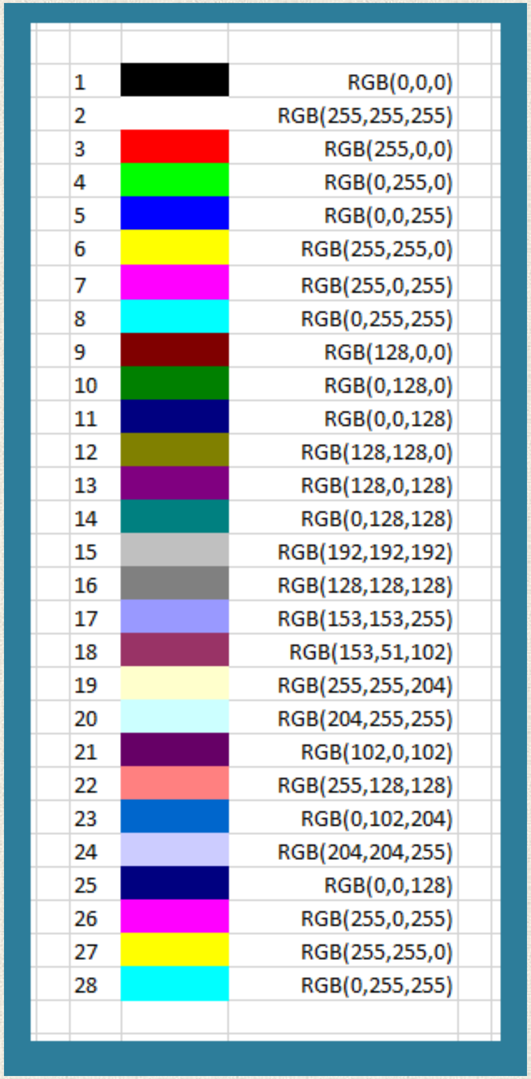
- **Input:** A collection of data divided into a training set and a test set. Each training data point is labeled into one of k classes.
- **Output:** a predictive labeling of all the points in the test set into one of k classes.

Example: Our data might be images of skin lesions, which we want to classify as non-neoplastic, a benign tumor, or malignant (cancer).

Converting data into a manageable form

We then need to *vectorize* our data in some way, converting each object into a collection of variables.

Example: If each image has n pixels, then each pixel has three RGB values, representing the amount of red, green, and blue in each pixel. This produces $3n$ 0-1 decimal values for each image.

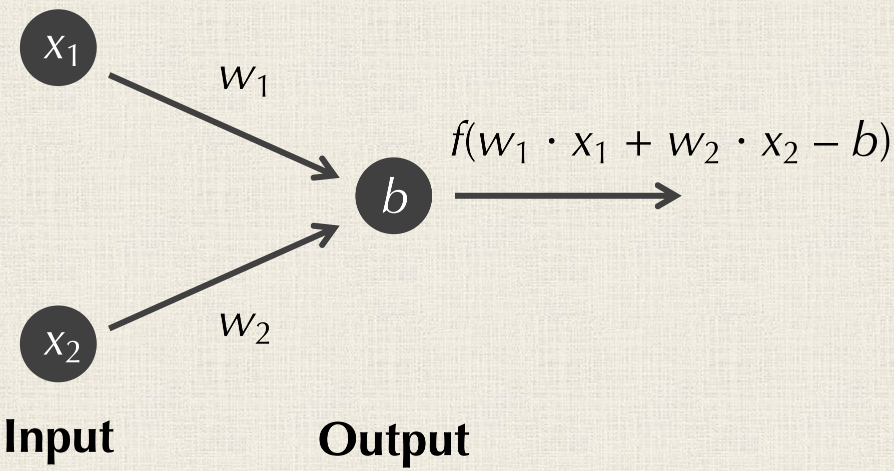


1	RGB(0,0,0)
2	RGB(255,255,255)
3	RGB(255,0,0)
4	RGB(0,255,0)
5	RGB(0,0,255)
6	RGB(255,255,0)
7	RGB(255,0,255)
8	RGB(0,255,255)
9	RGB(128,0,0)
10	RGB(0,128,0)
11	RGB(0,0,128)
12	RGB(128,128,0)
13	RGB(128,0,128)
14	RGB(0,128,128)
15	RGB(192,192,192)
16	RGB(128,128,128)
17	RGB(153,153,255)
18	RGB(153,51,102)
19	RGB(255,255,204)
20	RGB(204,255,255)
21	RGB(102,0,102)
22	RGB(255,128,128)
23	RGB(0,102,204)
24	RGB(204,204,255)
25	RGB(0,0,128)
26	RGB(255,0,255)
27	RGB(255,255,0)
28	RGB(0,255,255)

https://excelatfinance.com/xlf/xlf-colors-1.php

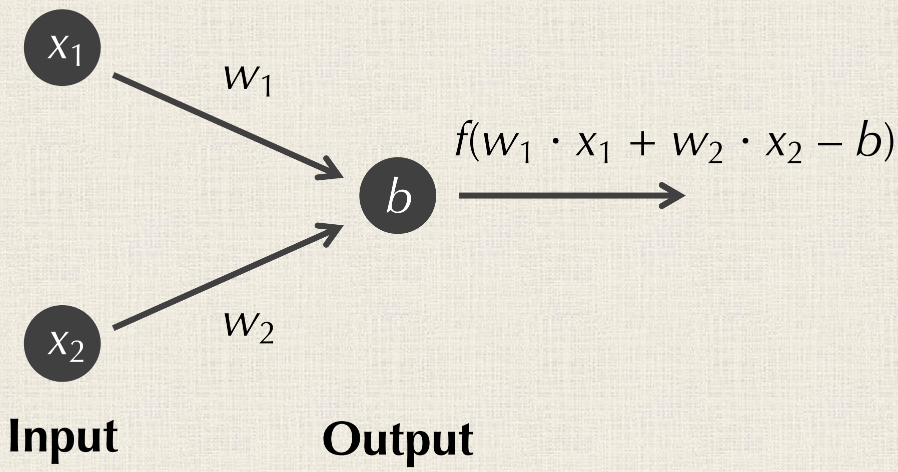
Generalizing neural networks

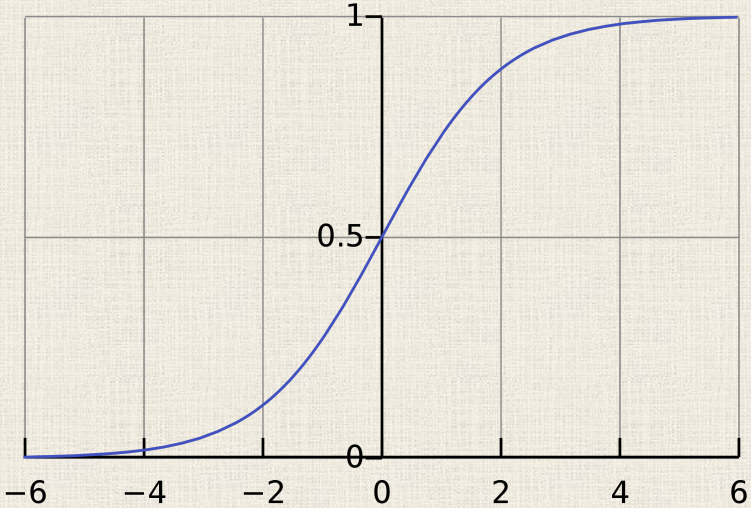
A generalized neuron allows n arbitrary decimal inputs (often between 0 and 1) and fires $f(w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n - b)$ for an **activation function** f and a constant **bias** b.



Generalizing neural networks

One common activation function is the **logistic function:** $f(x) = 1/(1 + e^{-x})$, shown below.

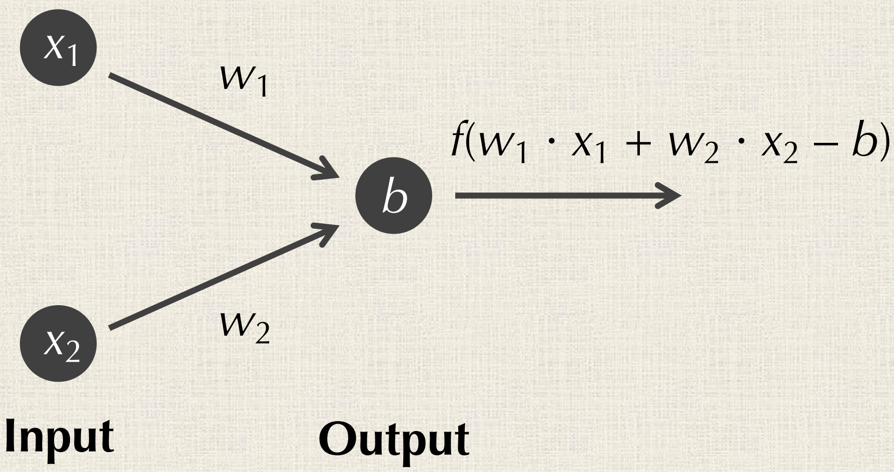




https://en.wikipedia.org/wiki/Logistic_function

Generalizing neural networks

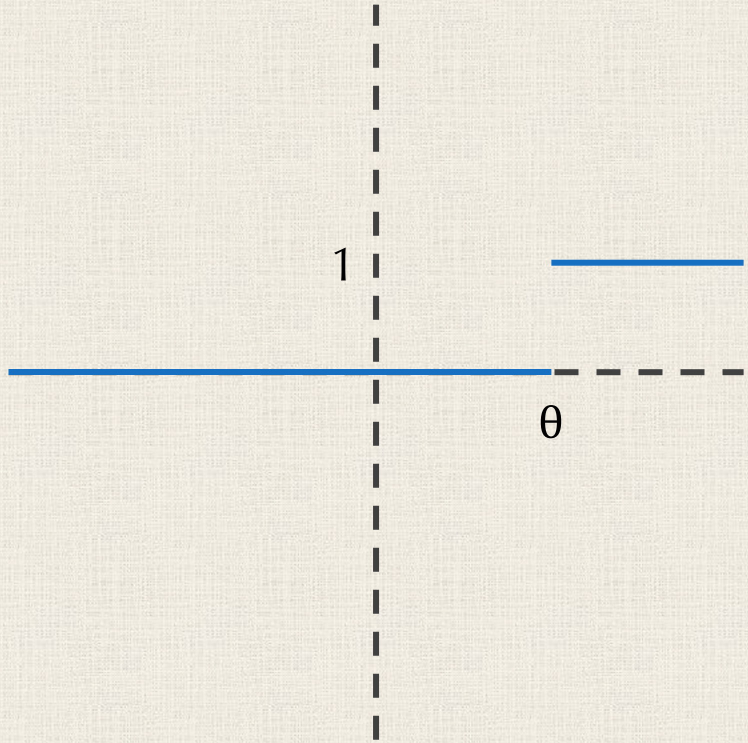
STOP: What was the "activation function" that we were using with perceptrons?



https://en.wikipedia.org/wiki/Logistic_function

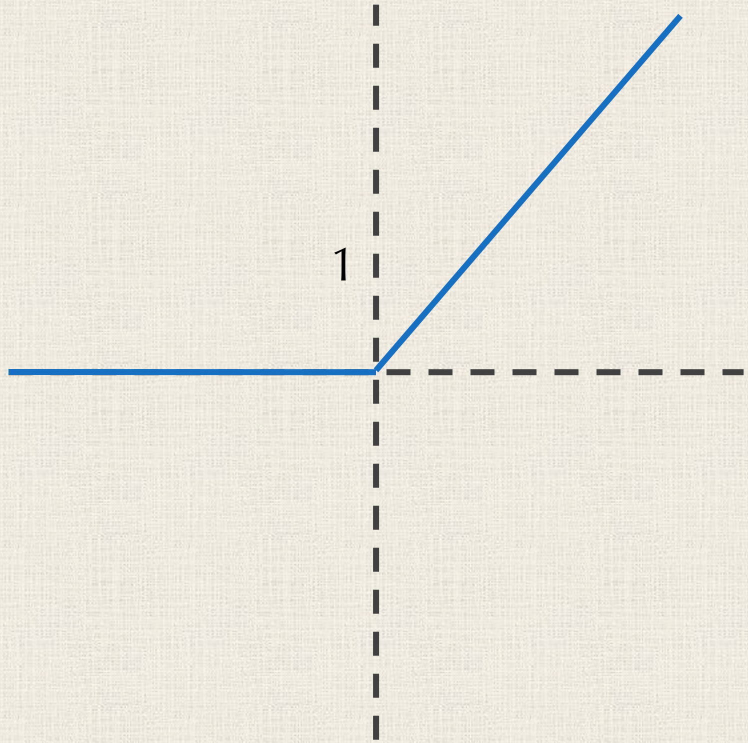
Generalizing neural networks

Answer: The "step function" $S(x)$ that outputs 1 if x is $\geq \theta$ and outputs 0 if $x < \theta$.

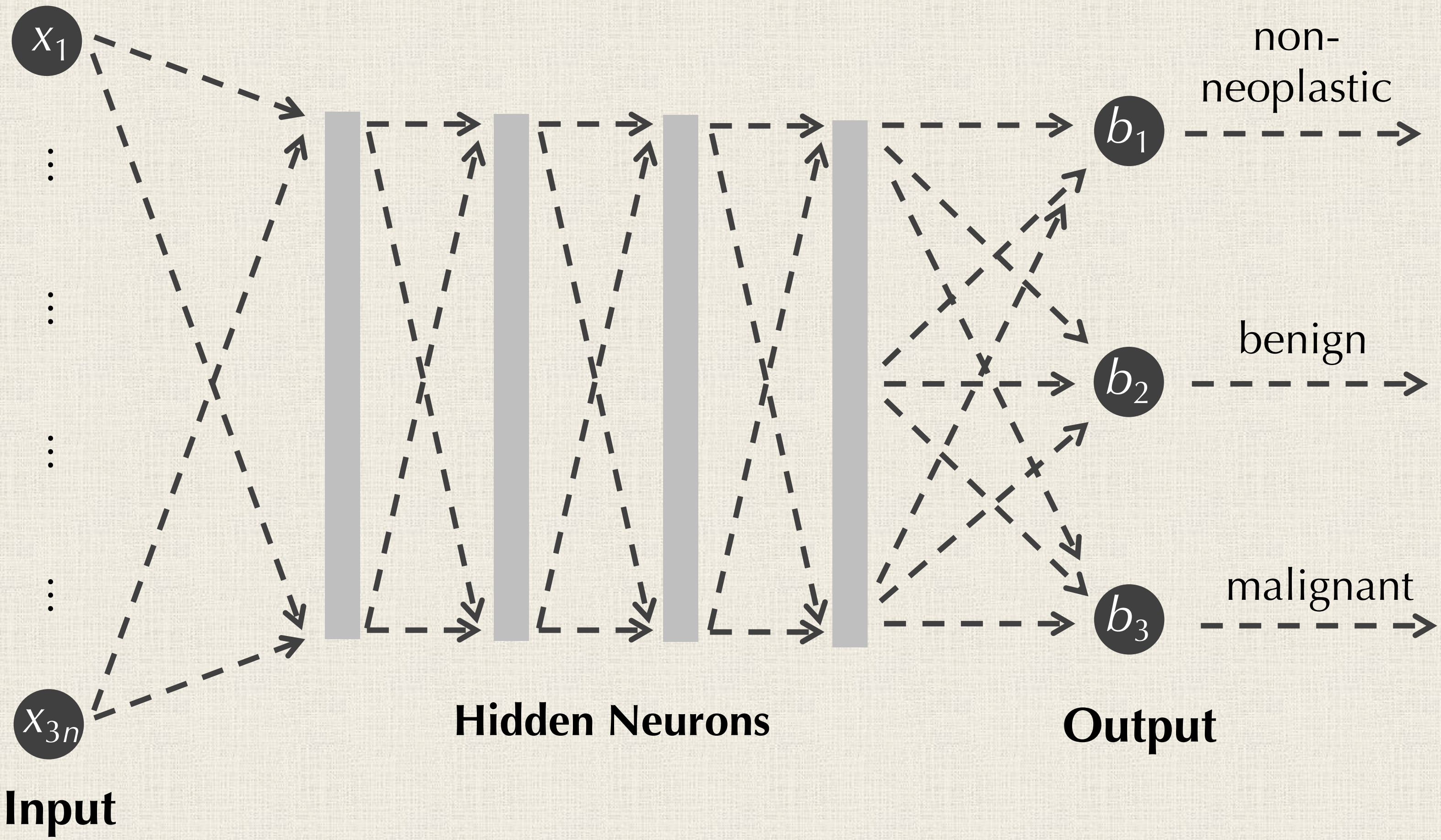


Generalizing neural networks

Note: even though it's simple, researchers now often use a "rectifier" function: $f(x) = \max(0, x)$.

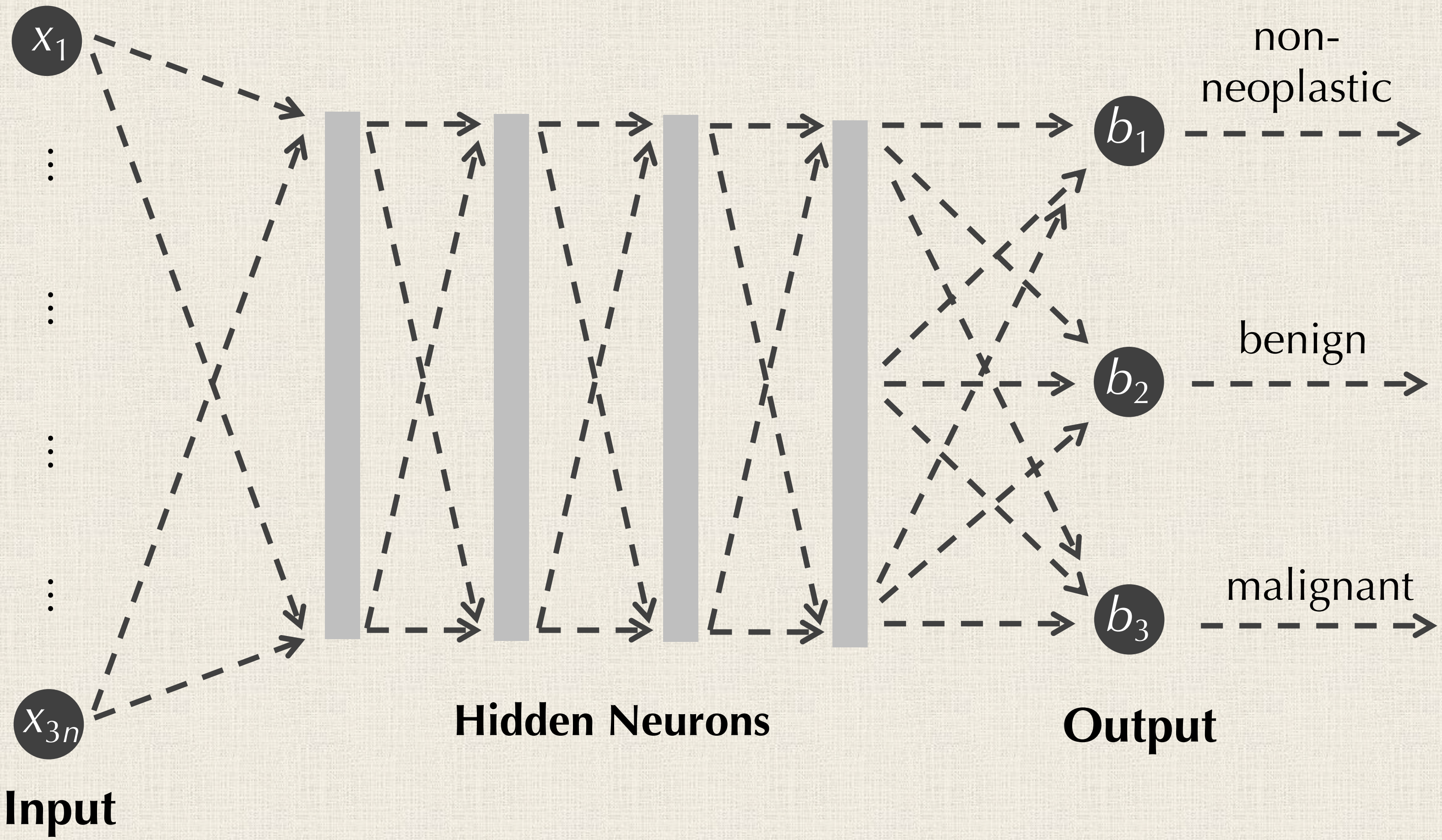


We then build some gigantic network with several hidden layers



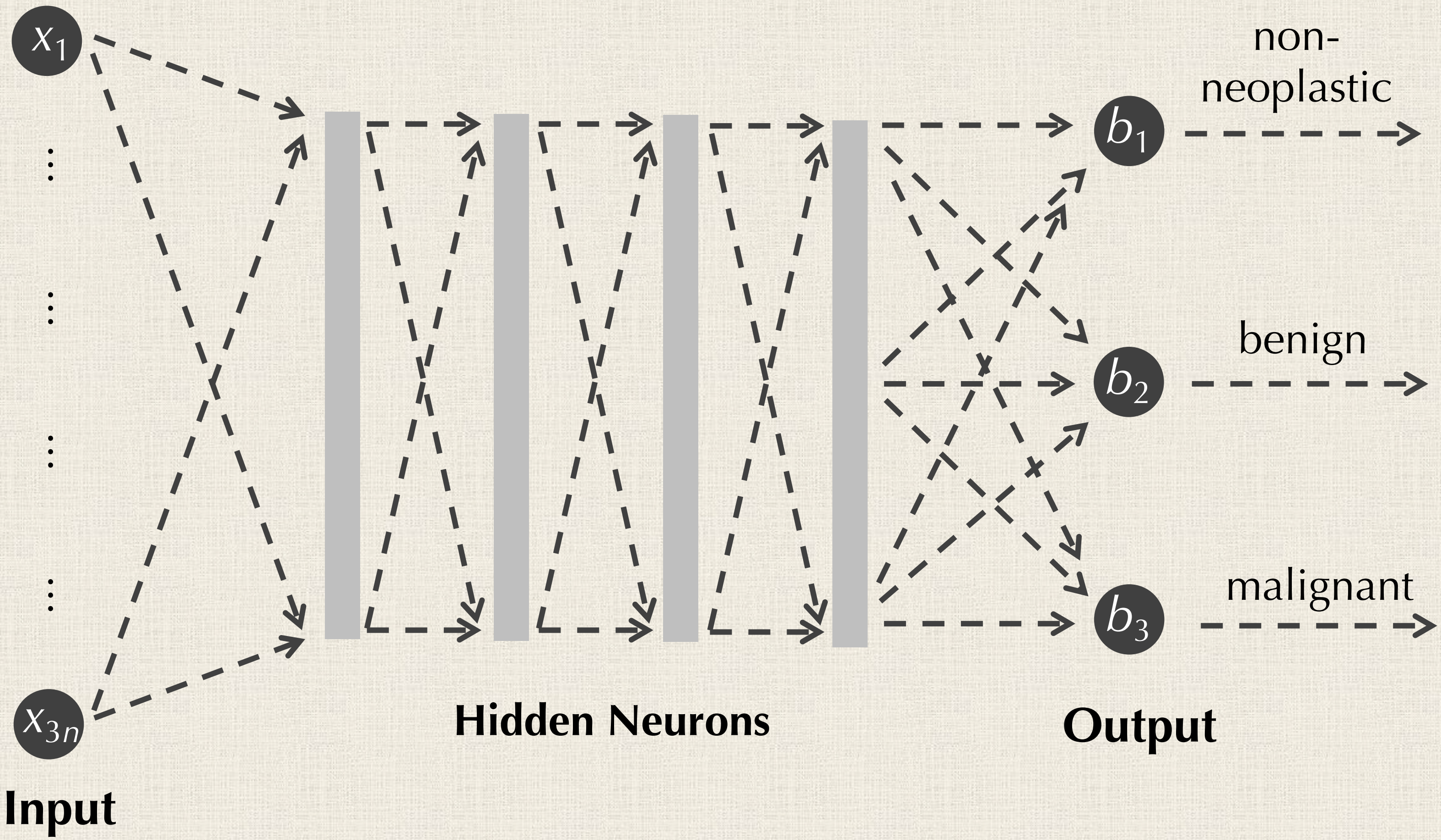
Congrats! You are now a **deep learning** expert.

We then build some gigantic network with several hidden layers



For a data value x, its output is a vector $P(x)$.

We then build some gigantic network with several hidden layers



We want $P(x)$ for a benign image *similar* to (0, 1, 0).

We have a lot of freedom in parameter selection

Note: For every neuron in our network, all of the input weights w_i are parameters.

Network Parameter Learning Problem

- **Input:** A collection of vectorized data and a neural network.
- **Output:** a collection of weights and biases that minimizes the average RMSD between an object x's correct label vector, $L(x)$, and the prediction from the network, $P(x)$, over all objects x.

We have a lot of freedom in parameter selection

STOP: Does “distance between two vectors” ring any bells?

Network Parameter Learning Problem

- **Input:** A collection of vectorized data and a neural network.
- **Output:** a collection of weights and biases that minimizes the average RMSD between an object *x*'s correct label vector, $L(x)$, and the prediction from the network, $P(x)$, over all objects *x*.

We have a lot of freedom in parameter selection

Answer: RMSD is one way of quantifying this distance.

Network Parameter Learning Problem

- **Input:** A collection of vectorized data and a neural network.
- **Output:** a collection of weights and biases that minimizes the average RMSD between an object x's correct label vector, $L(x)$, and the prediction from the network, $P(x)$, over all objects x.

We have a lot of freedom in parameter selection

STOP: What kind of computational problem is this?

Network Parameter Learning Problem

- **Input:** A collection of vectorized data and a neural network.
- **Output:** a collection of weights and biases that minimizes the average RMSD between an object x's correct label vector, $L(x)$, and the prediction from the network, $P(x)$, over all objects x.

We have a lot of freedom in parameter selection

Answer: It's an optimization problem, where the search space is the collection of weights/biases.

Network Parameter Learning Problem

- **Input:** A collection of vectorized data and a neural network.
- **Output:** a collection of weights and biases that minimizes the average RMSD between an object x's correct label vector, $L(x)$, and the prediction from the network, $P(x)$, over all objects x.

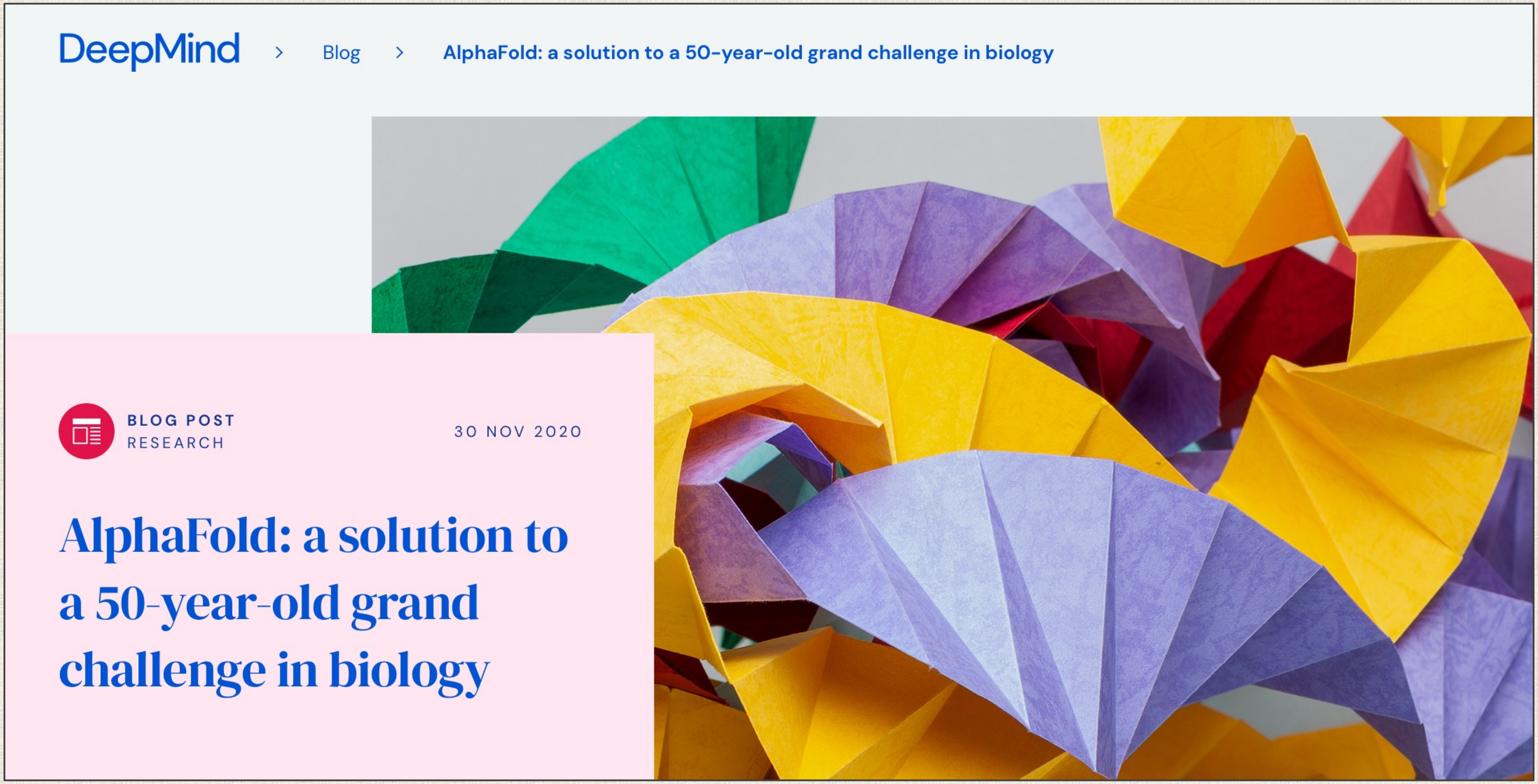
We have a lot of freedom in parameter selection

Note: Much of deep learning is just "build a big network and apply a local search heuristic".

Network Parameter Learning Problem

- **Input:** A collection of vectorized data and a neural network.
- **Output:** a collection of weights and biases that minimizes the average RMSD between an object x's correct label vector, $L(x)$, and the prediction from the network, $P(x)$, over all objects x.

Still, deep learning can be impressive…



… and a fancier version of our skin lesion network was a real paper!

https://www.nature.com › letters › article

Dermatologist-level classification of skin cancer with ... - Nature

by A Esteva · 2017 · Cited by 5697 — Using a single **convolutional neural network** trained on general **skin** lesion **classification**, we match the performance of at least 21 **dermatologists** tested across three critical diagnostic tasks: keratinocyte **carcinoma classification**, **melanoma classification** and **melanoma classification** using dermoscopy.

STOP: Any guesses on how accurate their algorithm was?

... and a fancier version of our skin lesion network was a real paper!

https://www.nature.com › letters › article

Dermatologist-level classification of skin cancer with ... - Nature

by A Esteva · 2017 · Cited by 5697 — Using a single **convolutional neural network** trained on general **skin** lesion **classification**, we match the performance of at least 21 **dermatologists** tested across three critical diagnostic tasks: keratinocyte **carcinoma classification**, **melanoma classification** and **melanoma classification** using dermoscopy.

STOP: Any guesses on how accurate their algorithm was?

Answer: Around 70% accurate, compared to 67% accuracy for a dermatologist.

Deep Learning + CB = 0 Great Ideas?

Royal Society
https://royalsocietypublishing.org › doi › rsif.2017.0387

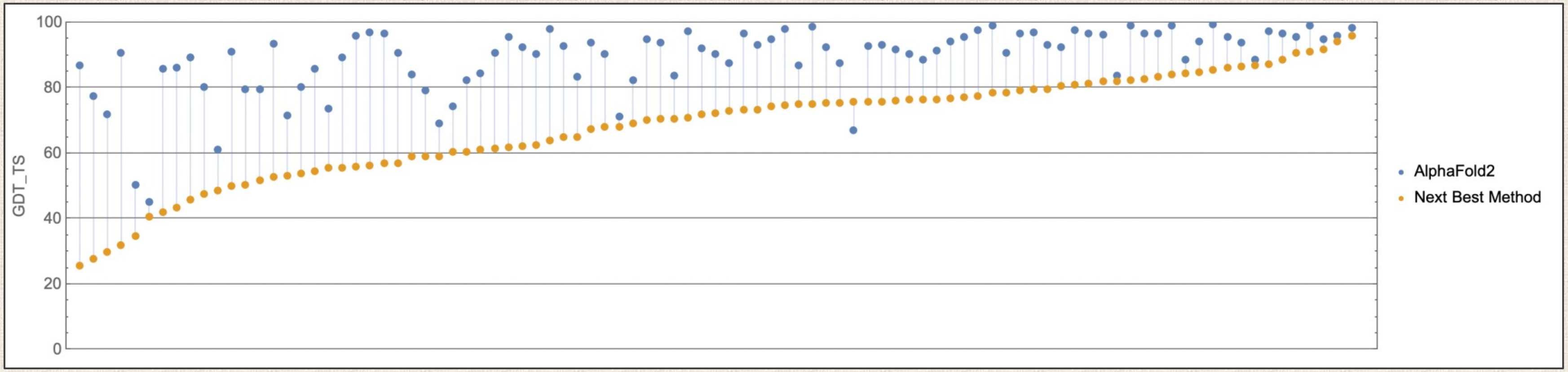
Opportunities and obstacles for deep learning in biology and ...

by T Ching · 2018 · Cited by 1906 — We examine applications of **deep learning** to a variety of biomedical problems—patient classification, fundamental **biological** processes and ...

Abstract · Deep learning and patient... · Deep learning to study the... · Conclusion

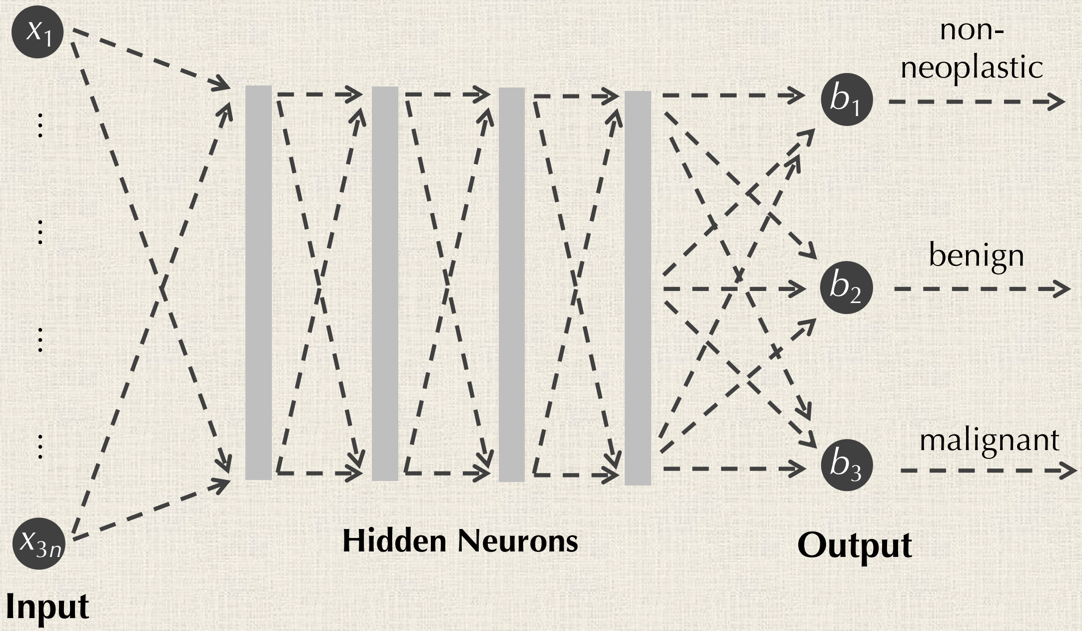
"Following from an extensive literature review, we find that deep learning has yet to revolutionize biomedicine or definitively resolve any of the most pressing challenges in the field, but promising advances have been made on the prior state of the art."

This Might Not Age the Best!



Source: Mohammed AlQuraishi, https://bit.ly/39Mnym3.

… but is this really a model of *intelligence*?

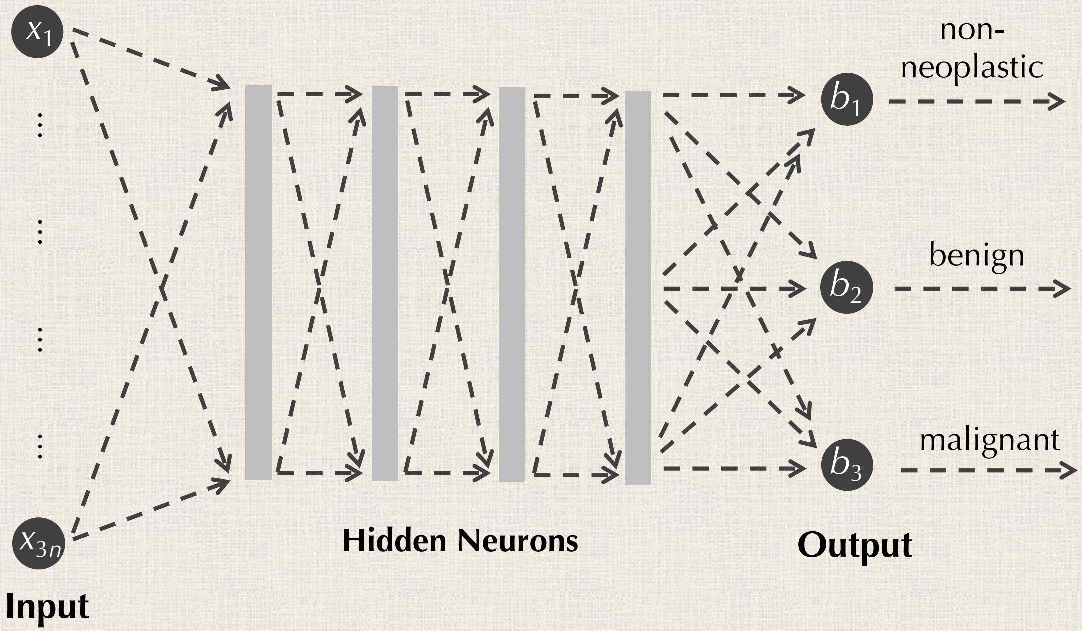


$\neq$

https://www.reddit.com/r/MachineLearning/comments/2fxi6v/ama_michael_i_jordan/

“Let's not impose artificial constraints based on cartoon models of topics in science that we don't yet understand.” – Michael I. Jordan, 2014

… but is this really a model of *intelligence*?



$\neq$

Idea: if nature is good at solving problems, why don't we study the algorithms that it has developed over the course of evolution?