When you hear a short snippet of your favorite song. Almost instantly, your brain recalls that song and many related memories, such as attending a recent concert featuring that artist. It seems very natural, after all, people can recall information all the time. However, if you think about it, this problem is computationally non-trivial.

Let's put ourselves in the shoes of evolution and try to come up with an algorithm for the brain to solve it. The first approach that comes to mind is to store a database of all the songs you have heard a large number of times. When an audio fragment of an unknown song is received, we can scan through all the songs in our database, find the one that has a close enough match. However, the search space of every song I've ever heard is astronomically large. Therefore, forming an exhaustive search would be simply impossible.

So how does the brain accomplish this so quickly?

**W**e can draw insights from biology. There is a concept known as Levinthal's paradox of protein folding. As you may know, proteins are long chains of amino acids that fall into specific three-dimensional structures, which determine their function.

Given the number of possibilities, it seems like it would take an astronomical amount of time for a protein to search through all the possible structures to find its correct folded state. Yet, in reality, proteins fall into their native structures in a matter of milliseconds.

After all, the protein molecule is not a computer, so it doesn't do any sort of search. It just falls into the most stable and favorable configuration according to physical laws.

Each possible configuration of the protein chain has a specific potential energy level determined by the sum of all of these atomic interactions. In other words, we can assign a positive number to each state equal to its energy, which is a function in some very high-dimensional space. Let's abstractly visualize it as having just two dimensions. Then the energy function can be thought of as a surface where each point on it represents a possible protein configuration, and the height of the point represents the potential energy of that configuration. **This is the energy landscape.**

For a protein, it would be a complex surface with many peaks and valleys. Now, here is the key point: a protein molecule, like any physical system, tends to minimize its potential energy guided **by the 2nd law of thermodynamics**. It will naturally seek out the configuration that has the lowest possible energy level.

When a protein is folding, it is essentially rolling downhill on the energy landscape, following the steepest path towards the valley. This is why proteins can fall so quickly. They don't need to search. they simply follow the natural tendency of physical systems to minimize their potential energy.

**Now, the core idea is to achieve something similar for the case of memo**ry. Suppose we have a system that can encode information in its states, and each configuration has a specific potential energy determined by the interaction between the states. Then we need to first somehow sculpt the underlying energy landscape so that memories we want to store correspond to these wells in the energy surface.

Second, we need something that would play the role of the 2nd law of thermodynamics and would direct the system towards the nearest local minimum.

Once these two things are achieved, retrieving a memory that is most similar to the input pattern is done. Encode the input pattern initially and Let it run to the equilibrium, Descend into the energy well, from which we can read out the source memory.

Let's consider a set of neurons, abstract units, that can be in one of two possible plus 1 or minus 1. This is a simplified analogy of how nerve cells in the brain encode information; they either generate an impulse at a given point or remain silent. We will focus on the fully connected network. Each neuron has connections to every other neuron. These connections have weights associated with them.

For a pair of units i and j, we denote the connection weight between them as (w{ij}), and the states of neurons themselves as (xi) and (xj).

For simplicity, we will stick to the symmetric weights. In other words, neurons (i) and (j) are connected by the same weight in both directions.

Now that we have a set of neurons linked with each other through weight and connections, let's explore what these connection weights represent.

We can think of each connection as being either happy or unhappy, depending on the states of its neurons. For example, if (w\_{ij}) is a large positive number, it means that neurons \(i\) and \(j\) are closely coupled, and one excites the other. In this case, when one neuron is active, the other tends to be active as well, and when one is silent, the other one is more likely to be silent.

This alignment between these signs can be expressed more concisely using the product \(x\_i \times x\_j\). This product will be positive when both neurons have the same sign and negative when they have different signs. By multiplying this product further by the connection weight, we obtain an expression for the happiness of that connection. For a positive \(w\_{ij}\), happiness will be positive when the product of the two states is positive.

But this is just one edge. We can extend this idea and compute the happiness of the entire network. The larger that number is, the more overall agreement there is between connection weights and states of neurons. Ultimately, we will search for a set of weights that maximize this.

As we discussed previously, we want the Hopfield network to be able to gradually evolve towards an energy minimum. But looking closely at the formula, we can see that the energy value depends both on the states and the weights, so there is a lot of things the system can change.

What exactly is getting adjusted?

There are essentially two modes of network updates: adjusting the weights and Inference. Once the weights are fixed, to weaken the states of neurons to bring them into greater agreement with the weights corresponds to descending along the energy surface.

Let's take a look at inference first.

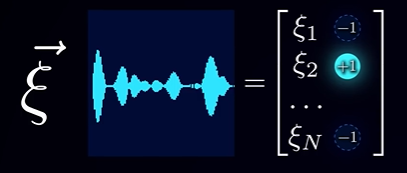
Suppose for a second someone has already set the weights (w\_ij), the neurons themselves with all the connection weights. However, the exact configuration of states, which neurons are active and which are silent, is unknown. How do we find the state pattern that would minimize the total energy? We will start with some initial state and try to lower the energy value by focusing on updating one neuron at a time.

We will calculate the total weighted input from all other neurons. This input is the sum of the states of all other neurons multiplied by their respective connection weights. You can think of this as a voting process: each neuron looks at the states of all other neurons, weighted by the strength of their connections, and decides whether to be active or silent based on the majority vote.

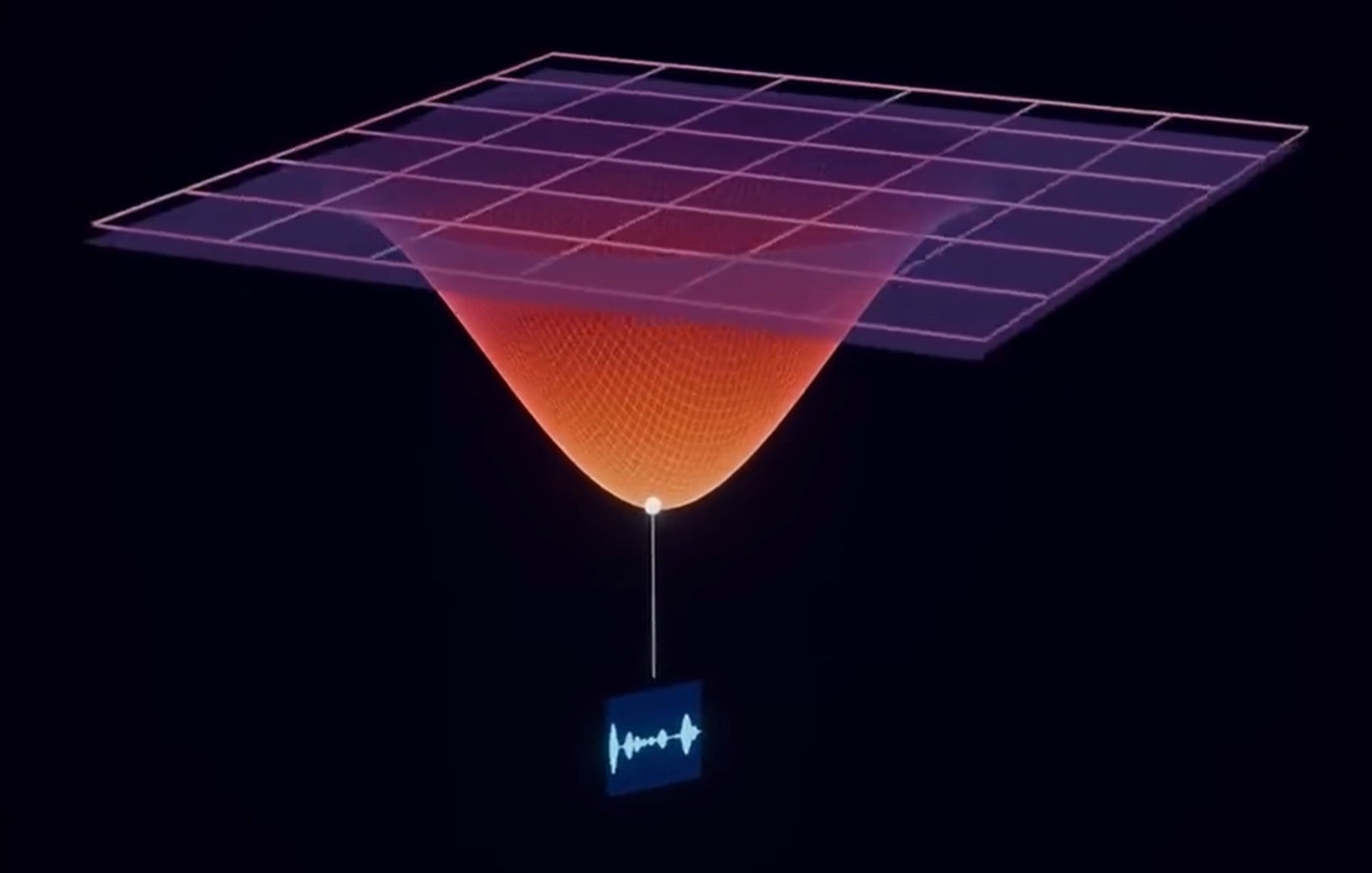
Once we've updated all neurons, we will have completed one iteration of the network inference and decreased the system's energy by a little bit. Keep repeating this process, and the network will gradually minimize the overall energy.

So far, we haven't talked about how we come up with a set of connection weights in the first place.

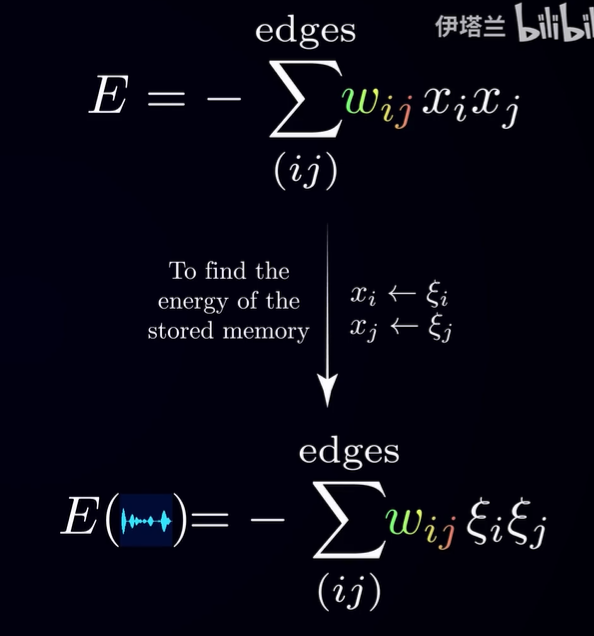
Let's consider memorizing a single pattern of states. the network has a single global minimum, one energy well, and converges to the same pattern no matter where you initialize it. Let's denote the template pattern that we'd like to store as ξ, which is a vector packing these states of all neurons,



and x\_i refers to the state of the i-th neuron in the network in general.

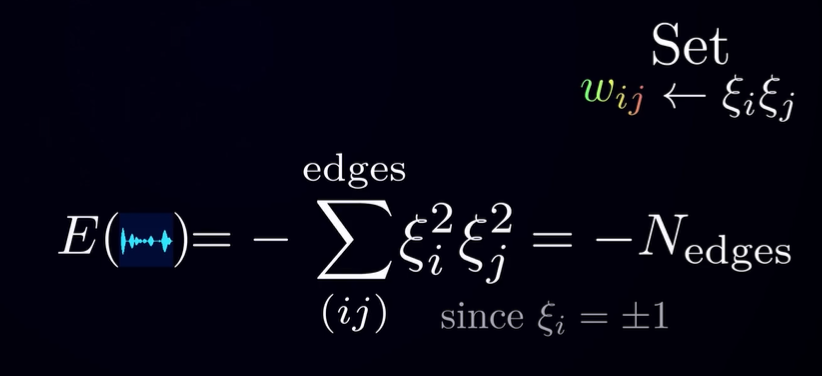


We want to set w\_{ij} so that this quantity would be at its minimal value. If we plug (x\_i = ξ\_i), we get the equation for the energy of the reference pattern as a function of weights, which we want to turn into a global minimum.

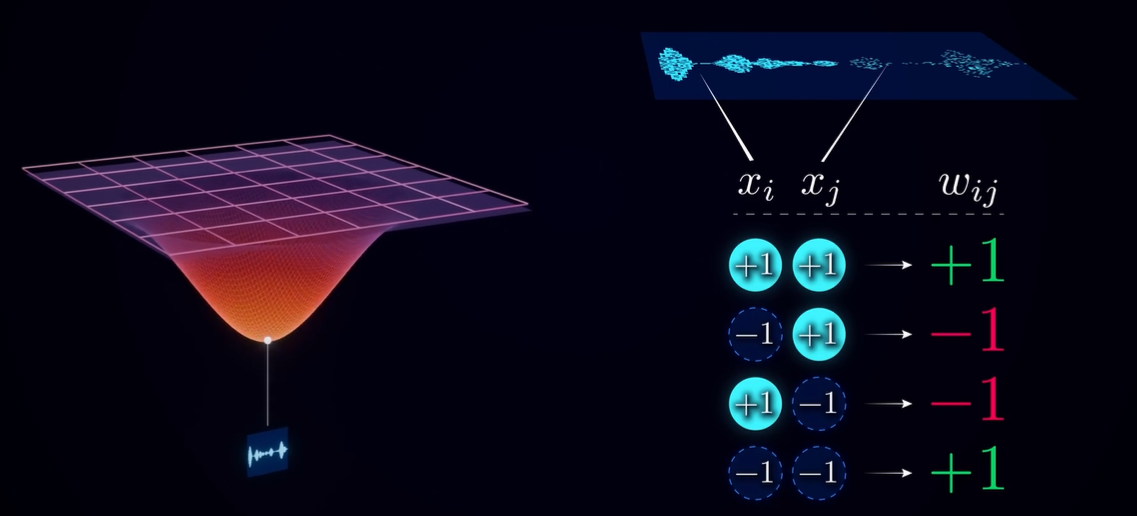


Now, intuitively, the lowest possible energy is obtained when all the connection weights fully align with the state pairs.

When we have just a single pattern, All we need is to set the weight (w\_{ij}) to be the product of the corresponding pair of states in the memory pattern. This way, every connection is satisfied, and the energy of the network when it is in the state ξ becomes the negative of the total number of edges. Any flip of a neuron would increase the energy, thus making it a stable state.

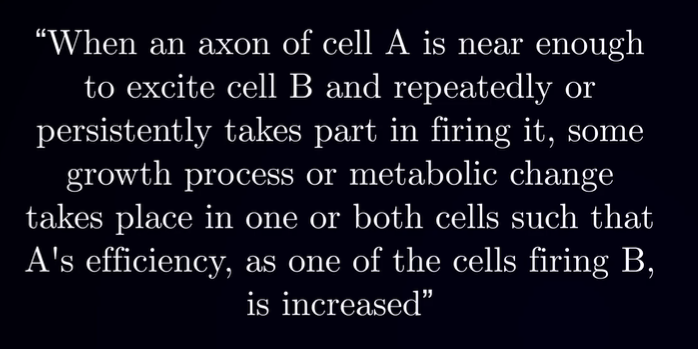


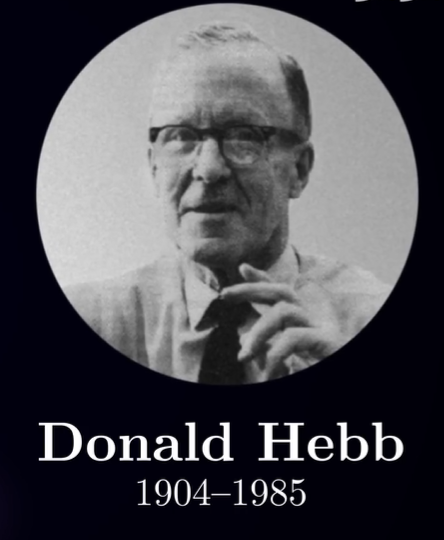
I want to reiterate the crucial point here: if we want to come up with a set of weights that would dig an energy well around some pattern, then all we need to know are the pairwise relationships between states in that pattern. If the two neurons are active together in the source memory, strengthening the connection between them lowers the Hopfield energy of that memory, effectively storing it in the weights for associative recall.



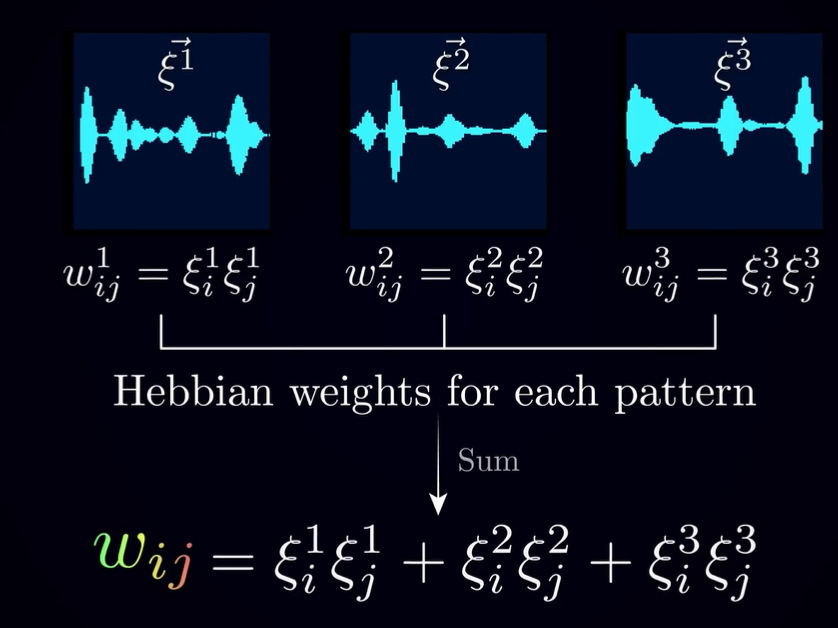
And in fact, what we just did is known as the Hebbian learning rule.

"Neurons that fire together wire together."

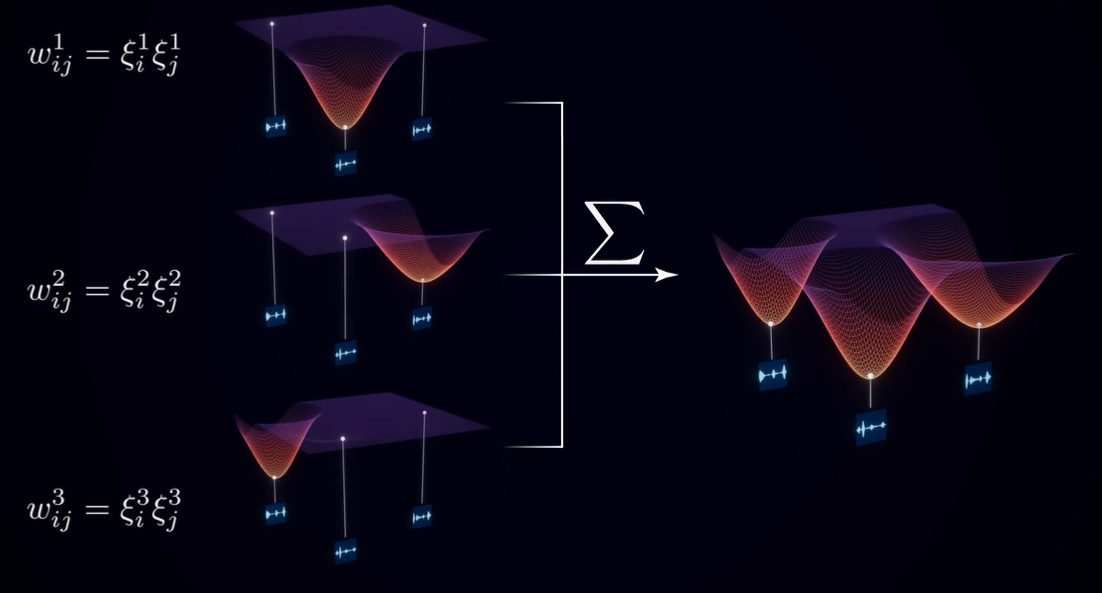




Great, so we found a way to make a single pattern a stable state of the network. But we want to store multiple patterns. Here's the key idea: we can simply sum the weights we would get for each pattern separately. So if we have three patterns, we can set the weights according to the following equation:



What this will do is turn each of the patterns into a local minimum. But, Intuitively, if the patterns you want to store are very different, then if you first independently dig energy wells around each of them and simply add the energy landscapes together, the result will have local minima in the same three valleys.



And this brings us to the limitation of the Hopfield networks: At some point, if we try to store too many patterns, the network will fail to converge to a stored pattern reliably and recall weird in-between kinds of memories.



However, to this day, they provide a powerful and model of associative memory, a simple network of neuron-like units that can store and retrieve patterns through purely local learning and inference rules. Hopfield networks have laid the groundwork for more advanced energy-based models.