Decoding GPT: Background on Neural Networks

Samy Wu Fung, Daniel McKenzie, Michael Ivanitskiy

2024-01-11

Neurons, to a Neuroscientist

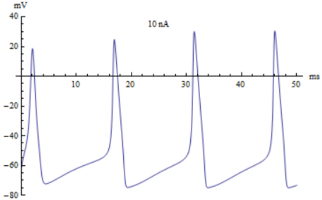
Hodgkin-Huxley model of a neuron is the gold standard:

$$I = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} + \bar{g}_{\mathrm{K}} n^4 (V_m - V_K) + \bar{g}_{\mathrm{Na}} m^3 h (V_m - V_{Na}) + \bar{g}_l (V_m - V_l)$$

$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n$$

$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m$$

$$\frac{dh}{dt} = \alpha_h(V_m)(1-h) - \beta_h(V_m)h$$



Intuition:

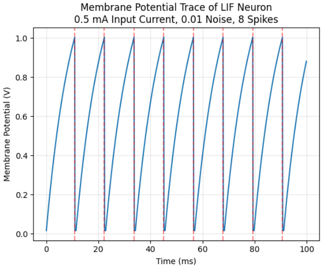
- input current I raises the membrane potential
- if V crosses a threshold, a *spike* occurs (voltage shoots up) due to Na^+ ion channels opening, and ions flowing into the cell
- spike causes K^+ ion channels to let K^+ out of the cell, which drives the voltage down

Leaky Integrate and Fire model is a simplified version:

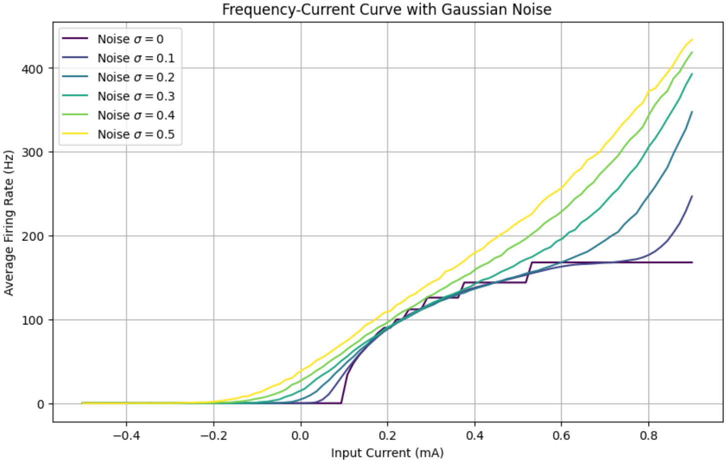
$$\frac{dV}{dt} = -\frac{1}{\tau_m}(V - V_{\text{rest}}) + \frac{I}{C}$$

- V: membrane potential
- τ_m: membrane time constant
- V_{rest}: resting membrane potential
- I: input current
- C: membrane capacitance

At every timestep, if $V > V_{\text{rest}}$, count that as a *spike*, and set $V = V_{\text{reset}}$



- "rate coding" is the assumption that neurons encode information in their firing rate
- neurons are often characterized by their "frequency-current" (fI) curve



Even further simplification: the “Perceptron”

- we want to build *networks* of neurons
- assume that the input current is the linear (weighted) sum of upstream neurons
- run the input through a nonlinear “activation function” σ to get the output

$$y = \sigma\left(\sum_{i \in \mathbb{N}_d} w_i x_i + b\right)$$

where the dimension d is the number of inputs, $b \in \mathbb{R}$ is the “bias” term, and $w \in \mathbb{R}^d$ is the “weight” vector.

This is a “single-layer perceptron”. If the activation function is:

- the identity, this is linear regression
- a step function, this is a binary classifier

Constructing a DNN

- "DNN": Deep Neural Network
- "MLP": Multi-Layer Perceptron (same thing)

with a single hidden layer:

$$h(x) = W_1 \cdot \sigma \left(W_0 \cdot x + b_0 \right) + b_1$$

with multiple layers:

- weights $W = [W_0, W_1, \ldots, W_n]$
- biases $b = [b_0, b_1, \ldots, b_n]$:

$$h(x) = \Lambda^{y}_{i \in \mathbb{N}_n} \sigma \left(W_i \cdot y_{i-1} + b_i \right)$$

NNs to a Neuroscientist

("NN" meaning "artificial neural network")

- NNs are simplified models of biological neural networks
- Each neuron in an ANN represents a rate-coded neuron in the brain
- connection structure doesn't match:
  - the brain is locally dense and globally sparse
  - NNs are dense between consecutive layers and have no connections otherwise

Why the nonlinearity?

We're already making so many approximations, why not drop the activation function σ too?

Consider:

$$h(x) = W_1 \cdot (W_0 \cdot x + b_0) + b_1$$
$$h(x) = (W_1 W_0 \cdot x + W_1 b_0) + b_1$$
$$h(x) = W_1 W_0 \cdot x + W_1 b_0 + b_1$$

If we define $W_* = W_1 W_0$ and $b_* = W_1 b_0 + b_1$, then we can rewrite this as:

$$h(x) = W_* \cdot x + b_*$$

So,

$$h(x) = W_1 \cdot (W_0 \cdot x + b_0) + b_1 = W_* \cdot x + b_*$$

This is just a single-layer perceptron! We haven't gained any expressive power by stacking layers without nonlinearities – we still just have a linear model.

NNs to a Mathematician

- ANNs are affine transformations with nonlinearities in between
- Can represent a wide range of mathematical functions, given sufficient depth and neuron count

which functions can we represent, exactly, and why?

Universal Function Approximation Theorem

Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x. Then σ is not polynomial if and only if for every $n \in \mathbb{N}, m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n, f \in C(K, \mathbb{R}^m), \varepsilon > 0$ there exist $d \in \mathbb{N}, W_0 \in \mathbb{R}^{d \times n}, b \in \mathbb{R}^d$, $W_1 \in \mathbb{R}^{m \times d}$ such that

$$\sup_{x \in K} \| f(x) - h(x) \| < \varepsilon$$

where

$$h(x) = W_1 \cdot (\sigma \circ (W_0 \cdot x + b))$$

- σ is our element-wise activation function – it must be *non-polynomial,* which means it cant be affine either!
- $h(x)$ is our single hidden layer neural network, with a hidden layer of size d
- f is our function we are trying to model, and its domain of interest K must be **compact** – closed and *bounded*
    - but K can still be as big as we want it to be
- for any continuous f, there exists a $d \in \mathbb{N}$ such that our neural network is within ε of f
- similar result exists for making the network deep instead of wide