S&DS 265 / 565

Introductory Machine Learning

Neural Networks

Thursday, November 11

Yale

Reminders

- Quiz 3 next Tuesday: LMs, embeddings, Bayes, TMs
- Assn 5 due today; Assn 6 out
- Questions?

Outline

- Connecting to embeddings and logistic regression
- Alternative view of linear models
- Adding nonlinearities — activation functions
- Biological analogy and inspiration
- Backpropagation — high level
- Examples: Regression

Logistic Regression

Recall:

$$\log\left(\frac{P(y=1\,|\,x)}{P(y=0\,|\,x)}\right) = \beta^T x + \beta_0$$

Equivalently:

$$P(Y=1\,|\,x) \propto e^{\beta^T x + \beta_0}$$

Logistic Regression

In the multi-class case we have

$$P(Y = k \mid x) \propto e^{\beta_k^T x + \beta_{k0}}, \quad k = 1, \ldots, K - 1$$

We can write this in ML terminology as

$$\text{Softmax}\left(\left\{\beta_k^T x + \beta_{k0}\right\}\right)$$

Note: Can also use β_k for $k = \underline{0}, \ldots, K - 1$. This will be "overparameterized"

Logistic Regression

What if x is an image, represented as pixels? It might be hard to get an accurate classifier.

Want to learn *feature representation* $\phi(x)$.

The model becomes

$$P(Y = k \mid x) \propto e^{\beta_k^T \phi(x) + \beta_{k0}}, \quad k = 0, 1, \ldots, K - 1$$

The parameters of ϕ and the parameters β need to be learned/trained.

Word embeddings

Applying this to language modeling, we could have a bigram model

The model becomes

$$P(w_{\text{next}} \mid w_{\text{prev}}) \propto e^{\beta_{w_{\text{next}}}^T \phi(w_{\text{prev}})}$$

where $\phi(w)$ is a learned feature vector or "embedding vector" for each word.

The parameters $\beta_w = \beta(w)$ are also embeddings. By making them the same as ϕ we get the word2vec model.

Starting with regression

For linear regression, our loss function for an example (x, y) is

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}(y - \beta^T x - \beta_0)^2 \\ &= \frac{1}{2}(y - f)^2\end{aligned}$$

where $f = \beta^T x + \beta_0$.

Adding a layer

Loss is

$$\mathcal{L} = \frac{1}{2}(y - f)^2$$

where now $f = \beta^T h + \beta_0$ where $h = Wx + b$.

This can be viewed graphically.

Equivalent to linear model

But this is just a linear model

$$f = \widetilde{\beta}^T x + \widetilde{\beta}_0$$

We get a reparameterization of a linear model; nothing new.

Need to add *nonlinearities*

Nonlinearities

Add nonlinearity

$$h = \phi(Wx + b)$$

applied component-wise.

For regression, the last layer is just linear:

$$f = \beta^T h + \beta_0$$

Nonlinearities

Commonly used nonlinearities:

$$\phi(u) = \tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

$$\phi(u) = \text{sigmoid}(u) = \frac{e^u}{1 + e^u}$$

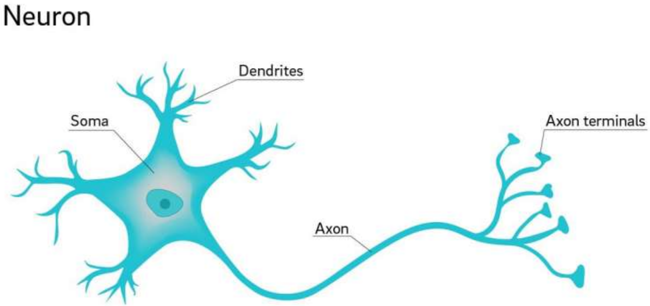
$$\phi(u) = \text{relu}(u) = \max(u, 0)$$

Nonlinearities

So, a neural network is nothing more than a parametric regression model with a restricted type of nonlinearity

Why are they called neural networks?

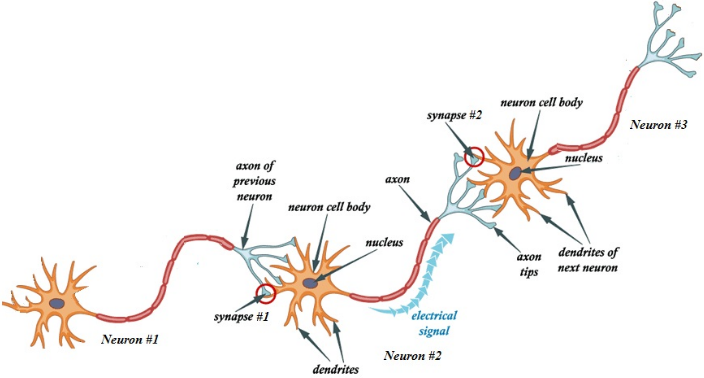
Biological Analogy



Biological Analogy

- The dendrites play the role of inputs, collecting signals from other neurons and transmitting them to the soma, which is the “central processing unit.”
- If the total input arriving at the soma reaches a threshold, an output is generated.
- The axon is the output device, which transmits the output signal to the dendrites of other neurons.

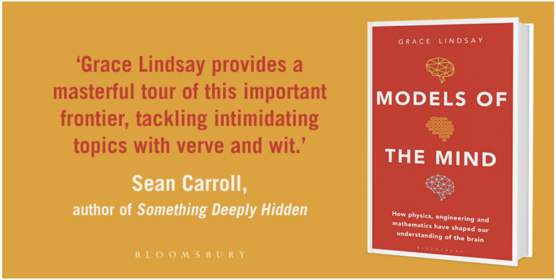
Biological Analogy



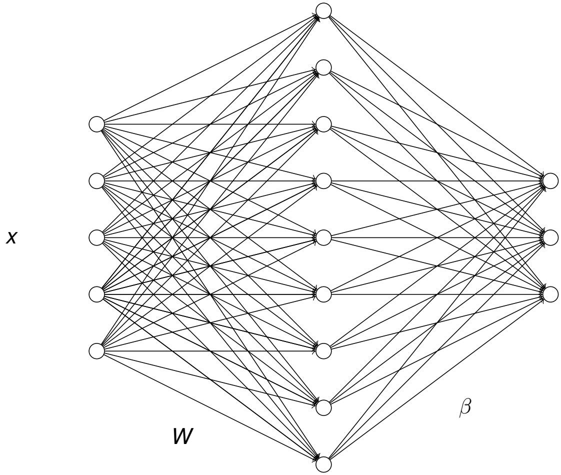
Plug: Grace Lindsay’s book

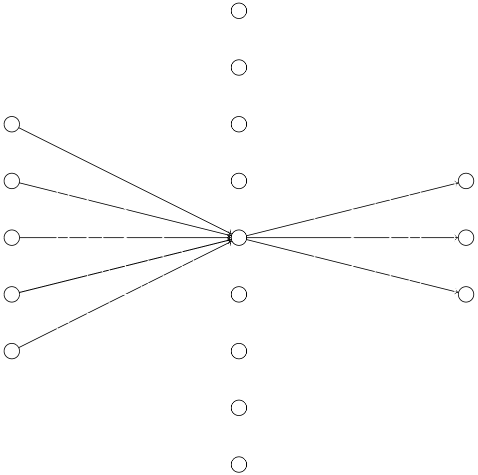
February 10, 2021 / neurograce

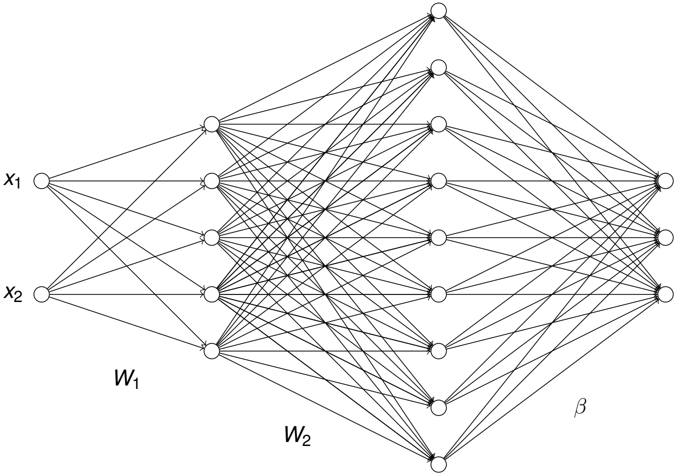
Models of the Mind: How Physics, Engineering and Mathematics Have Shaped Our Understanding of the Brain

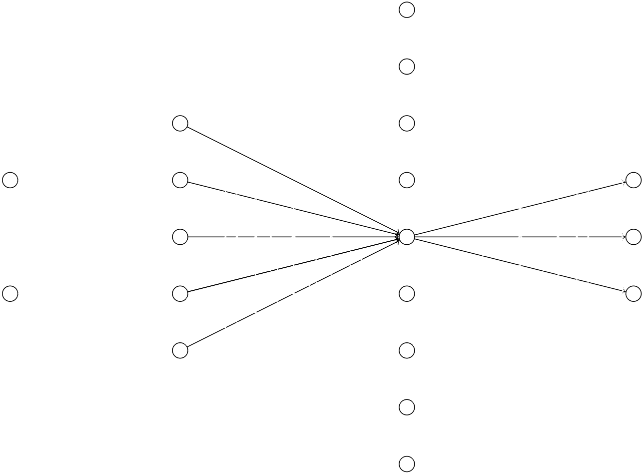


I wrote a book!


x
W
β




x_1
x_2
W_1
W_2
β



Training

- The parameters are trained by stochastic gradient descent.
- To calculate derivatives we just use the chain rule, working our way backwards from the last layer to the first.

Training

- For the last layer, $\mathcal{L} = \frac{1}{2}(y - f)^2$ and

$$\frac{\partial \mathcal{L}}{\partial f} = -(y - f)$$

- Next, we compute

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \beta} &= \frac{\partial \mathcal{L}}{\partial f} \frac{\partial f}{\partial \beta} \\
&= -(y - f) \frac{\partial f}{\partial \beta} \\
&= -(y - f) h
\end{aligned}$$

We'll go further next time

Summary

- For complex data we may want to learn features of the inputs
- This representation can be part of a logistic regression model
- Features that are linear transforms followed by a nonlinearity form the building blocks of (artificial) neural networks
- Based on a crude analogy with neurons in biological brains
- Trained using stochastic gradient descent
- Can be thought of as a particular type of nonlinear regression model