S&DS 265 / 565

Introductory Machine Learning

Neural Networks

October 31

Reminders

- Quiz 4 today
- Assn 4 due next week

For today

- Finishing up topic models – Gibbs sampling
- Neural networks

Outline: Neural networks

- 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

$$P(v \mid w) \propto e^{\beta_v^T \phi(w)}$$

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

The parameters β_v 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 $f(x) = \beta^T x + \beta_0$.

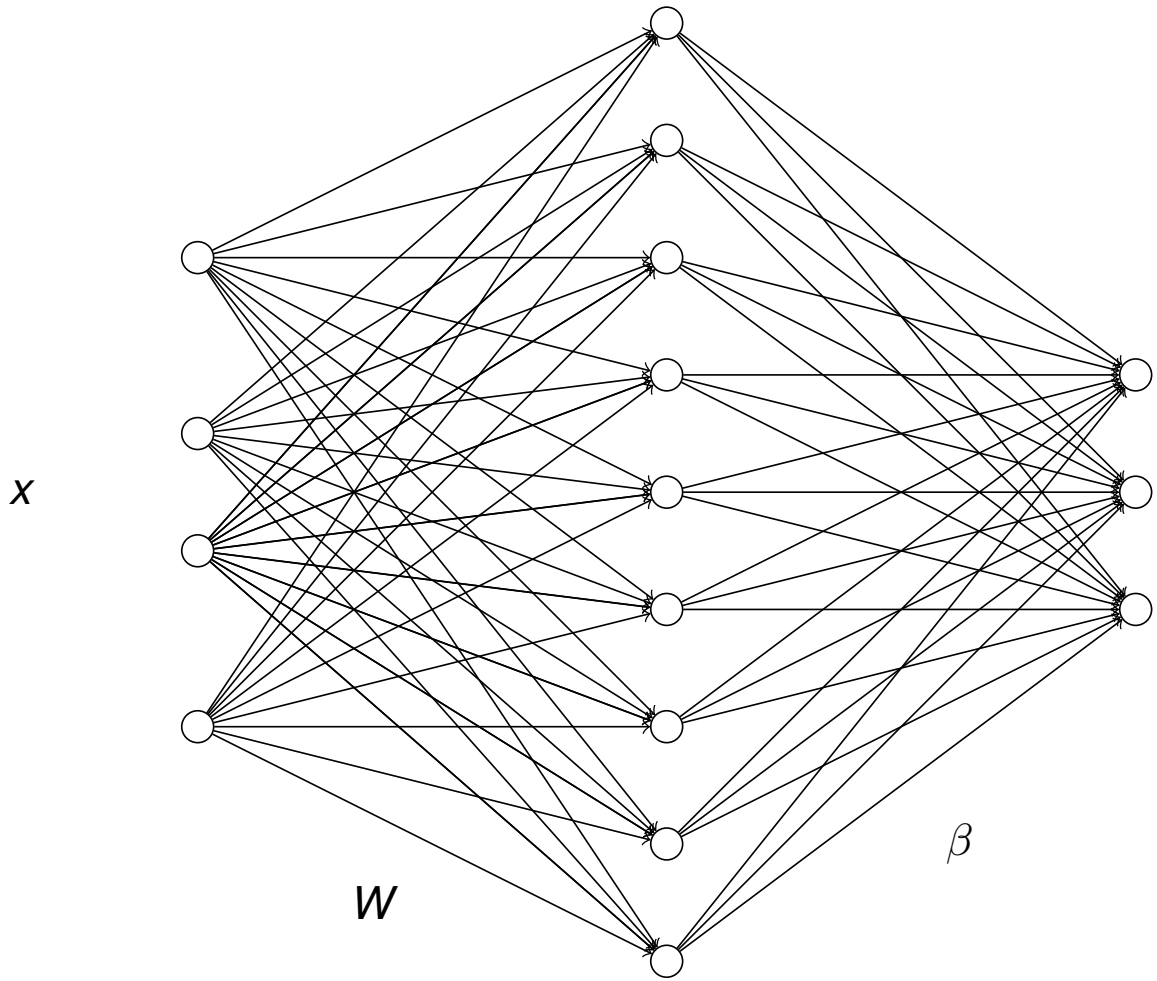
Adding a layer

Loss is

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

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

This can be viewed graphically.


x
W
β

Equivalent to linear model

But this is just a linear model

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

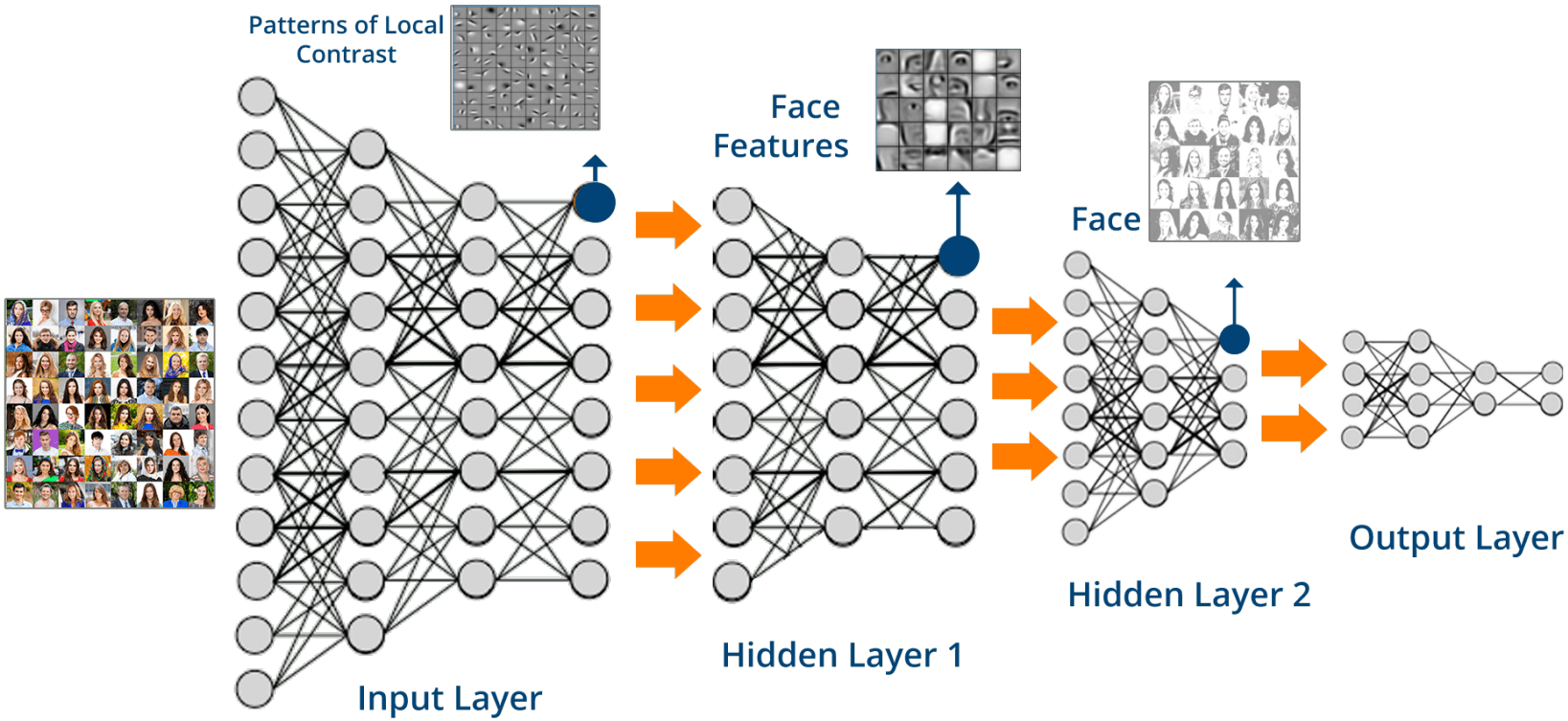
We get a reparameterization of a linear model; nothing new.

Need to add *nonlinearities*

Neural networks

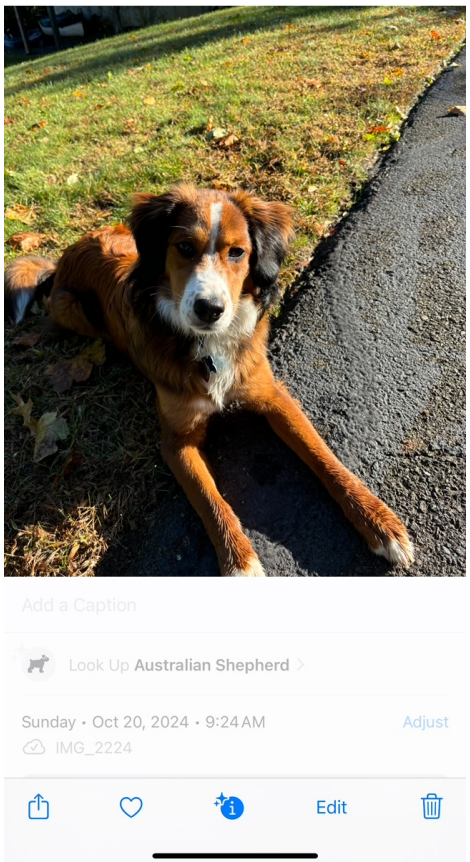
- Linear models are usually too simple to give good prediction rules
- Nonlinear models allow us to fit the data better—but run the risk of *overfitting*
- Neural networks are kind of in-between linear regression and k-nearest neighbors
- *Neural networks are layered linear models with carefully restricted nonlinearities*

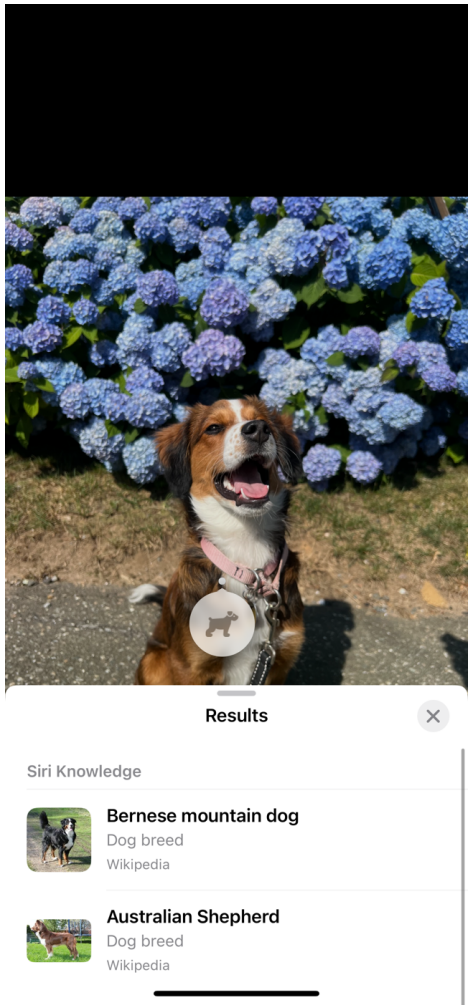
Deep neural networks

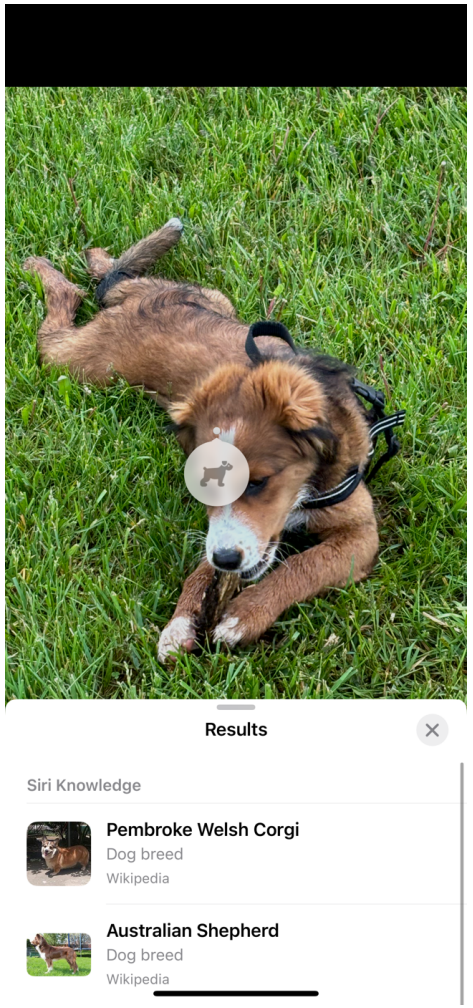


- Heuristics motivated from simplified view of the brain
- A particular form of nonlinear classification/regression

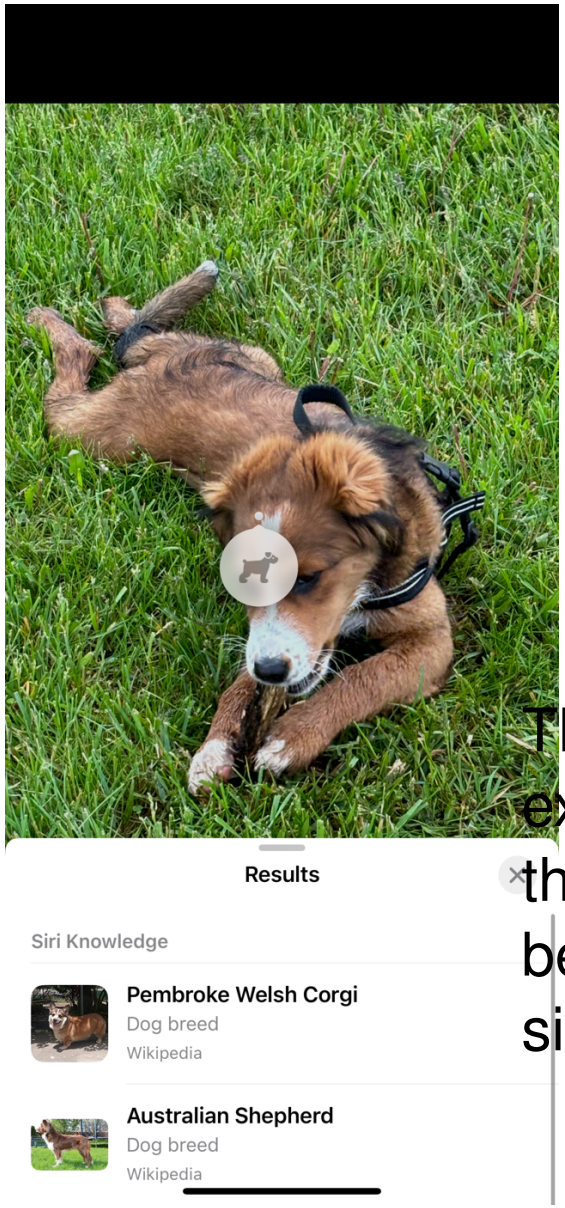
Everyday example







Everyday example

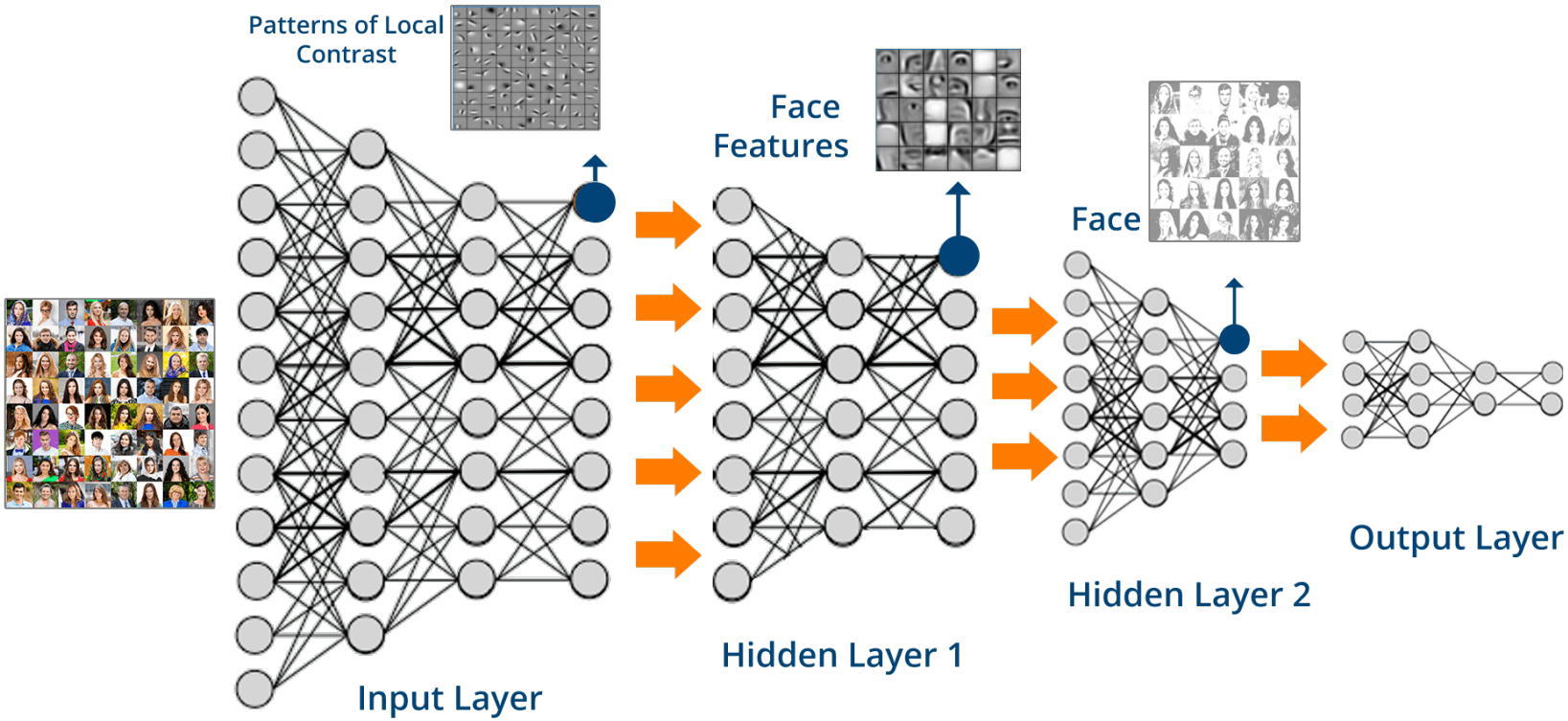


The neural network extracts “features” of the image that can be used to find similar images

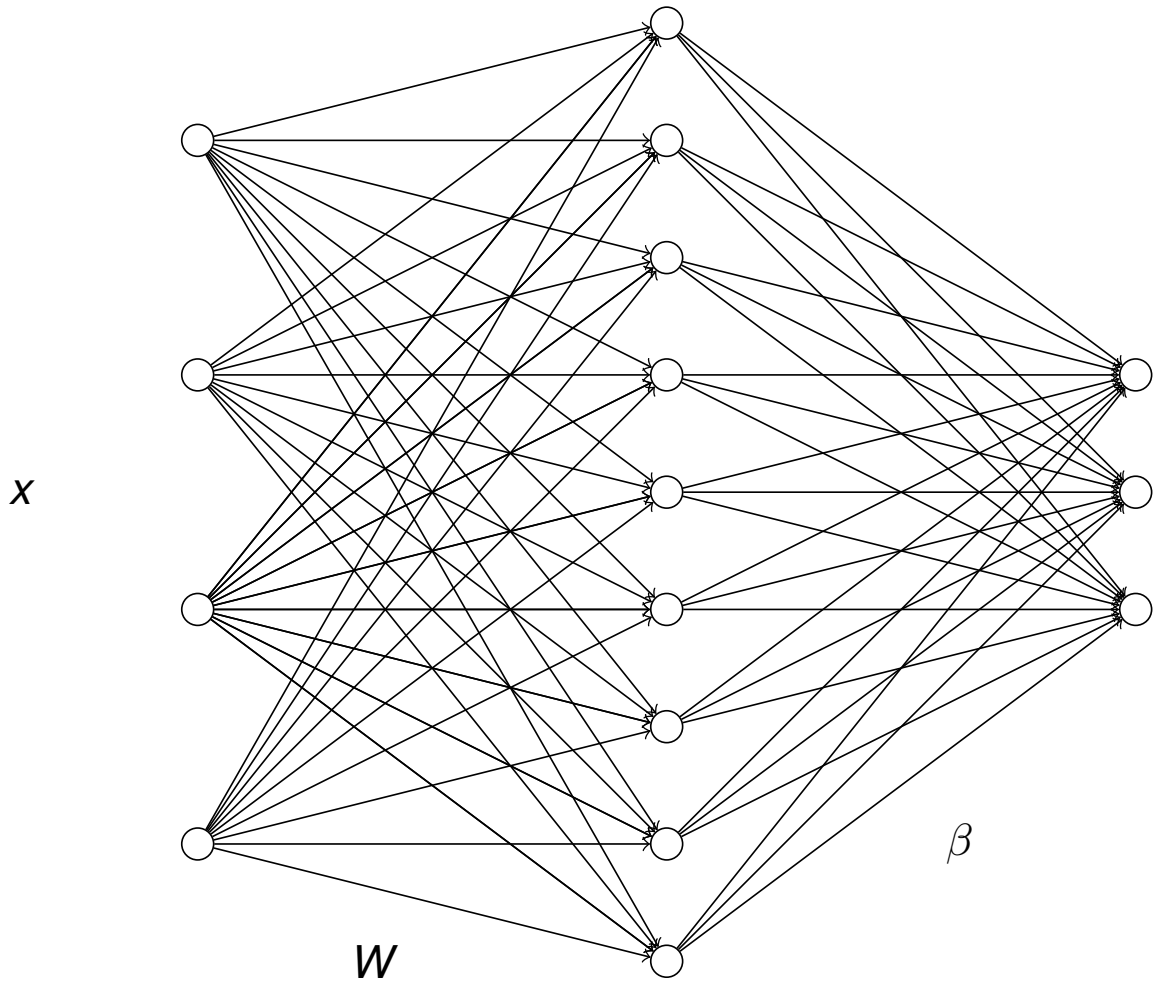
Everyday example

The neural network extracts “features” of the image that can be used to find similar images

Deep neural networks



- Heuristics motivated from simplified view of the brain
- A particular form of nonlinear classification/regression


x
W
β

Nonlinearities

Add *fixed nonlinearity*, for example

$$h = \max(Wx + b, 0)$$

applied component-wise to each "neuron"

- Typically the last layer is just linear
- So, a neural net is a linear model, but with complex *predictor variables* or "features" that are automatically learned from data

Nonlinearities

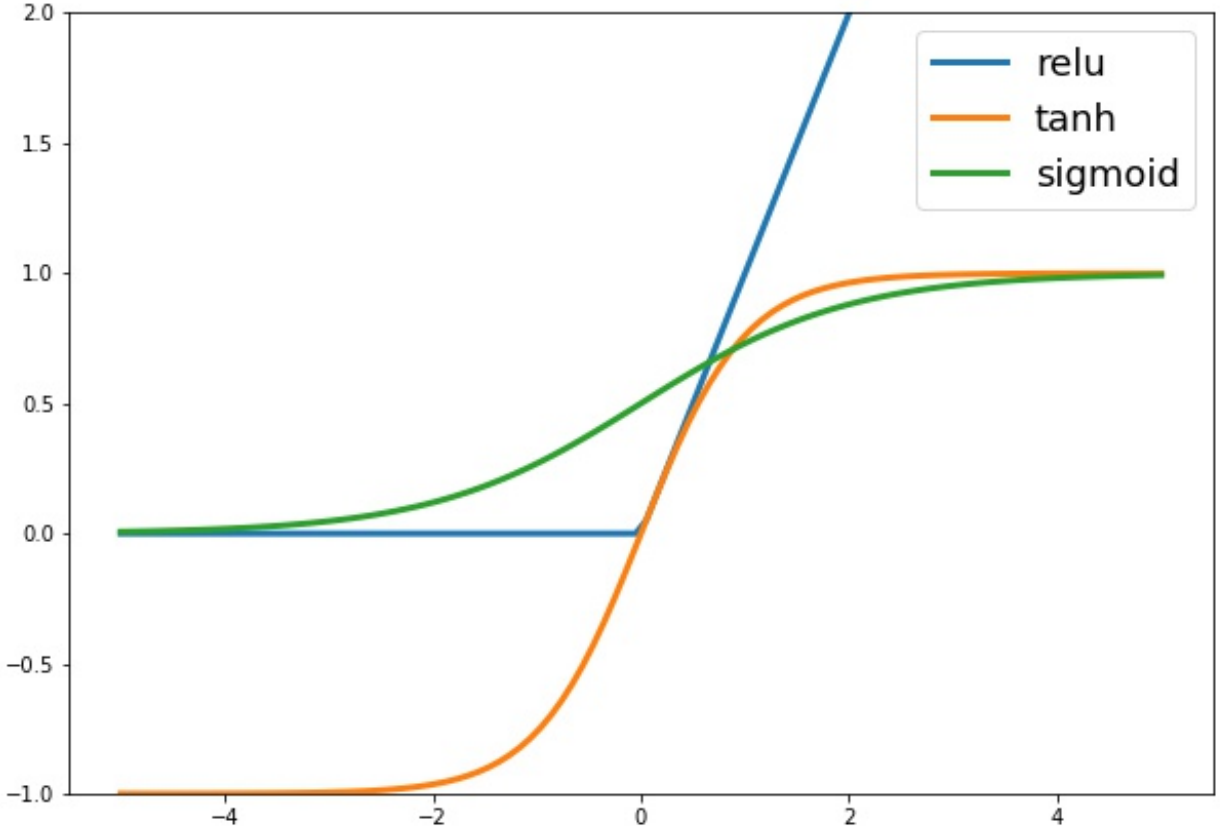
Commonly used nonlinearities:

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

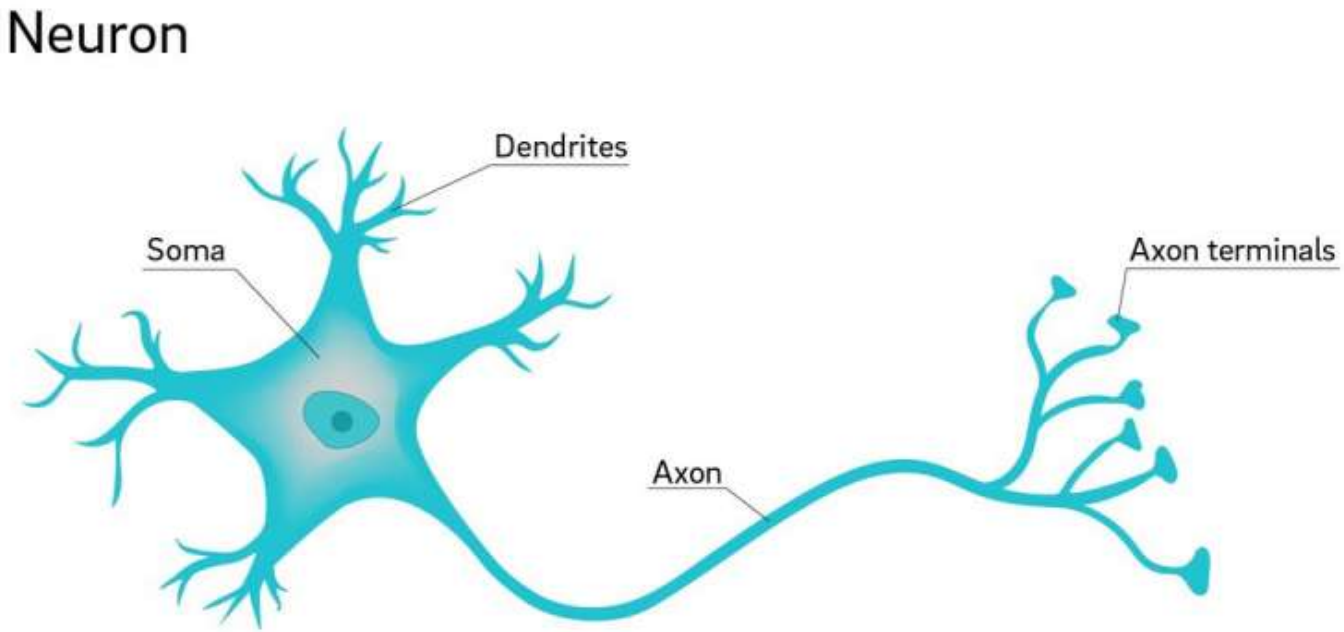
$$\phi(u) = \text{sigmoid}(u) = \frac{1}{1 + e^{-u}}$$

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

Nonlinearities

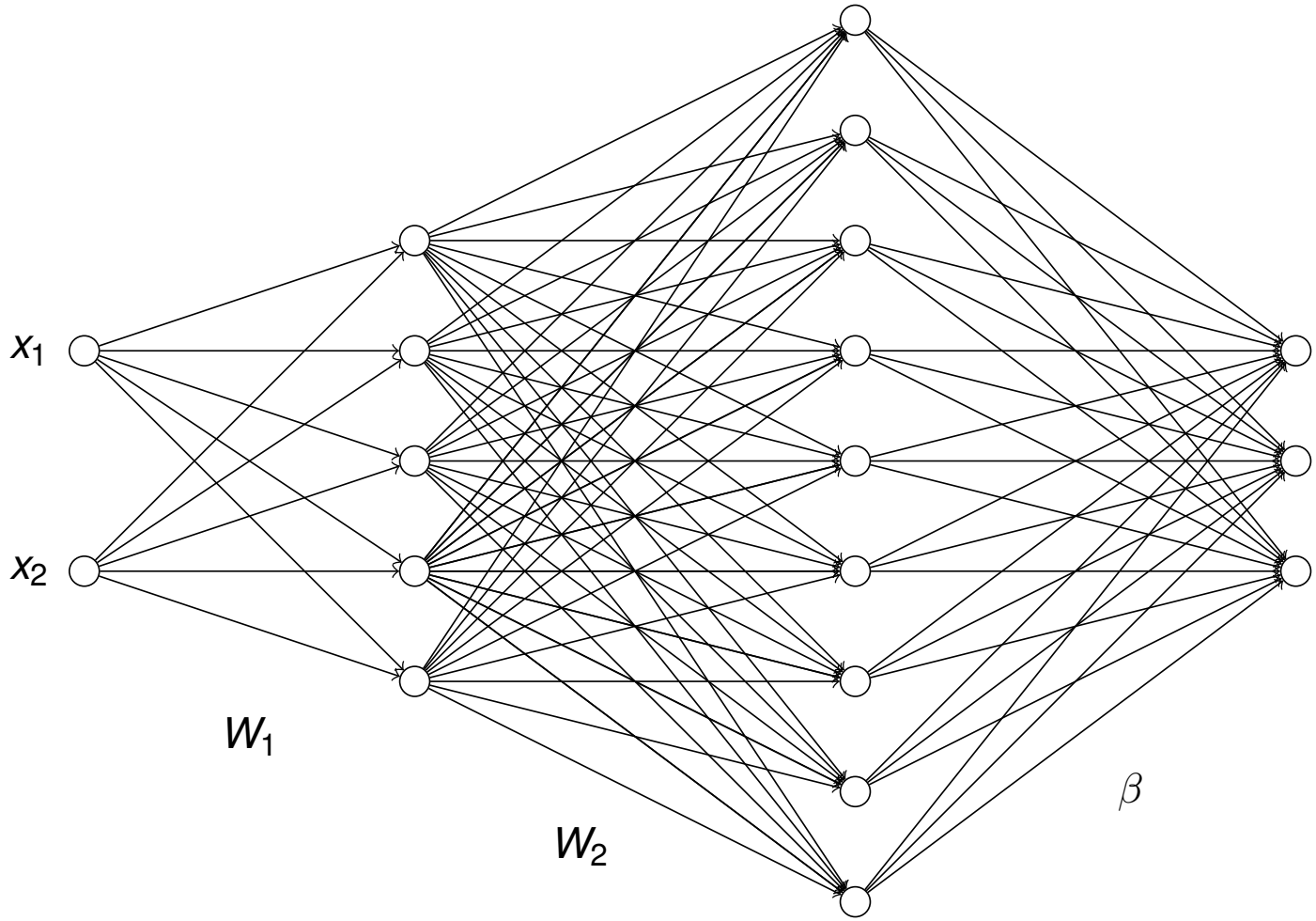


Biological analogy



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 — this is the nonlinearity!
- The axon is the output device, which transmits the output signal to the dendrites of other neurons.

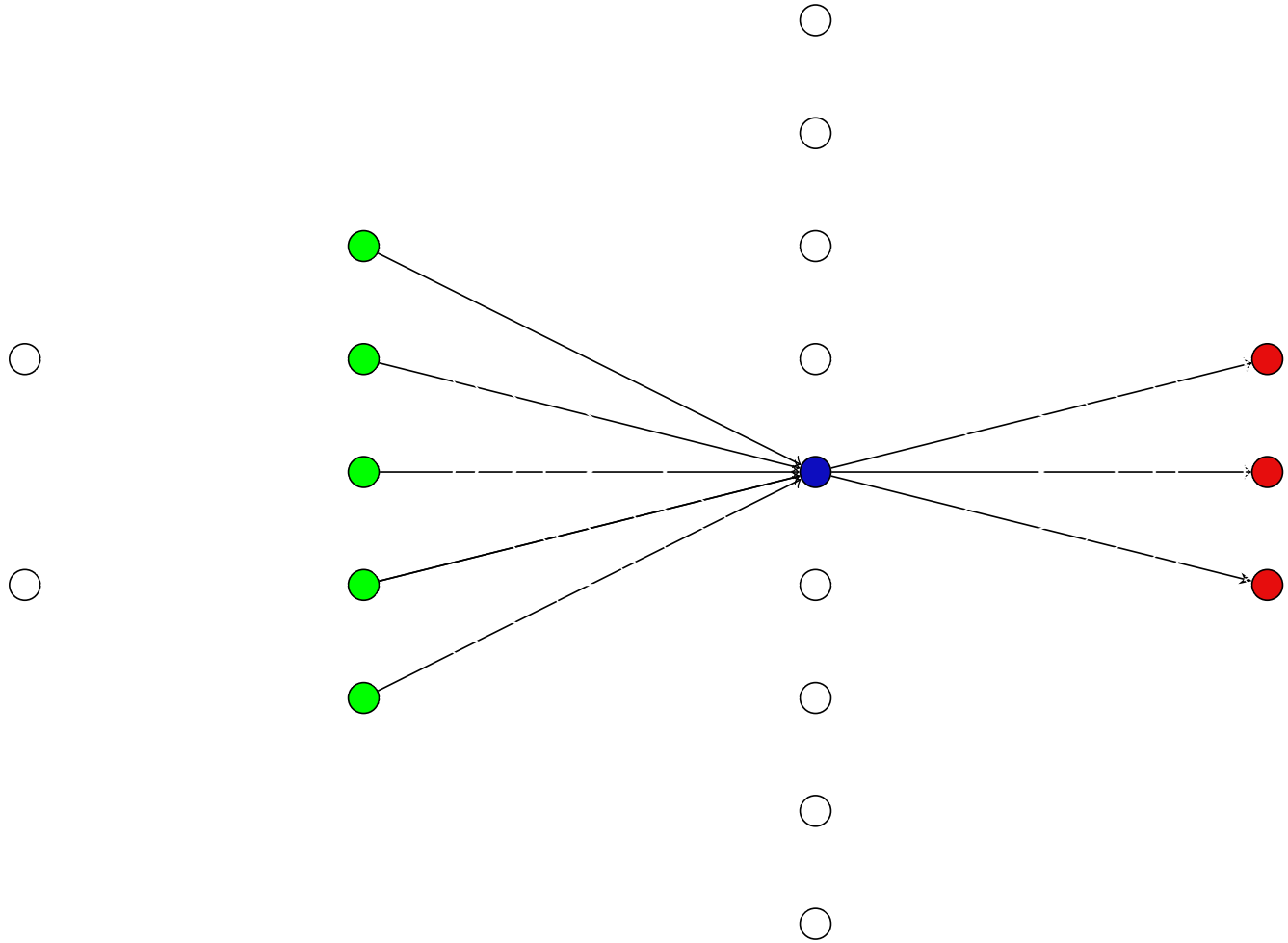


inputs

4 neurons
layer 1

9 neurons
layer 2

3 outputs



upstream neurons
(dentrites)

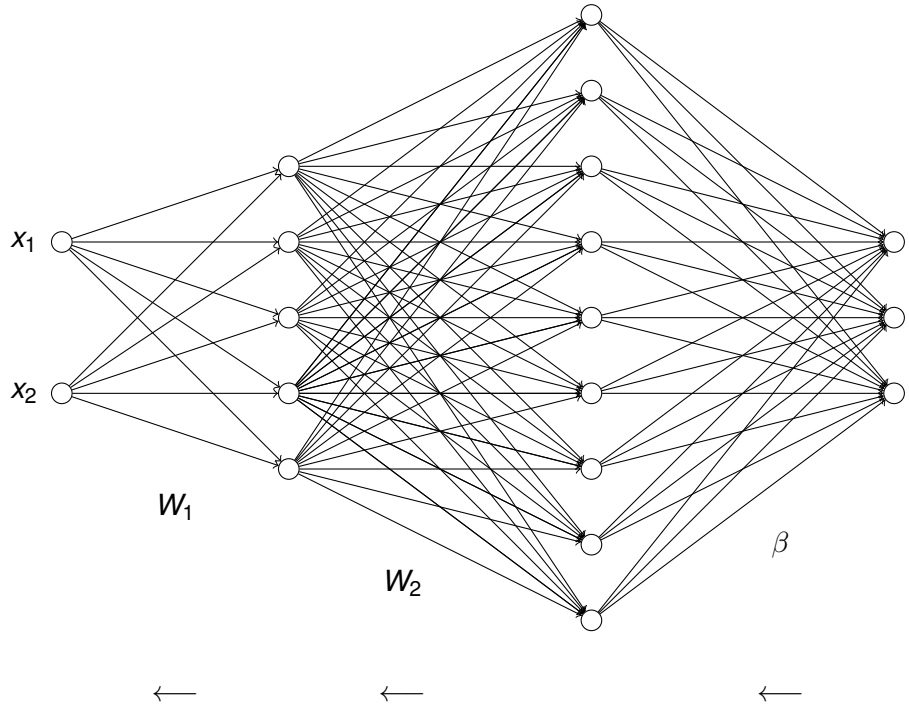
nonlinearity applied
(soma)

downstream neurons
(axon terminals)

Training

- The parameters are trained by “stochastic gradient descent”
  - Stream through the data, using small batches
  - Add a little bit to each network weight — proportional to how sensitive the predictions are to that weight (derivative)
  - Derivatives calculated with automatically — no math!
  - Stop if overfitting is detected!

High level idea: Backpropagation



Start at last layer, send error information back to previous layers

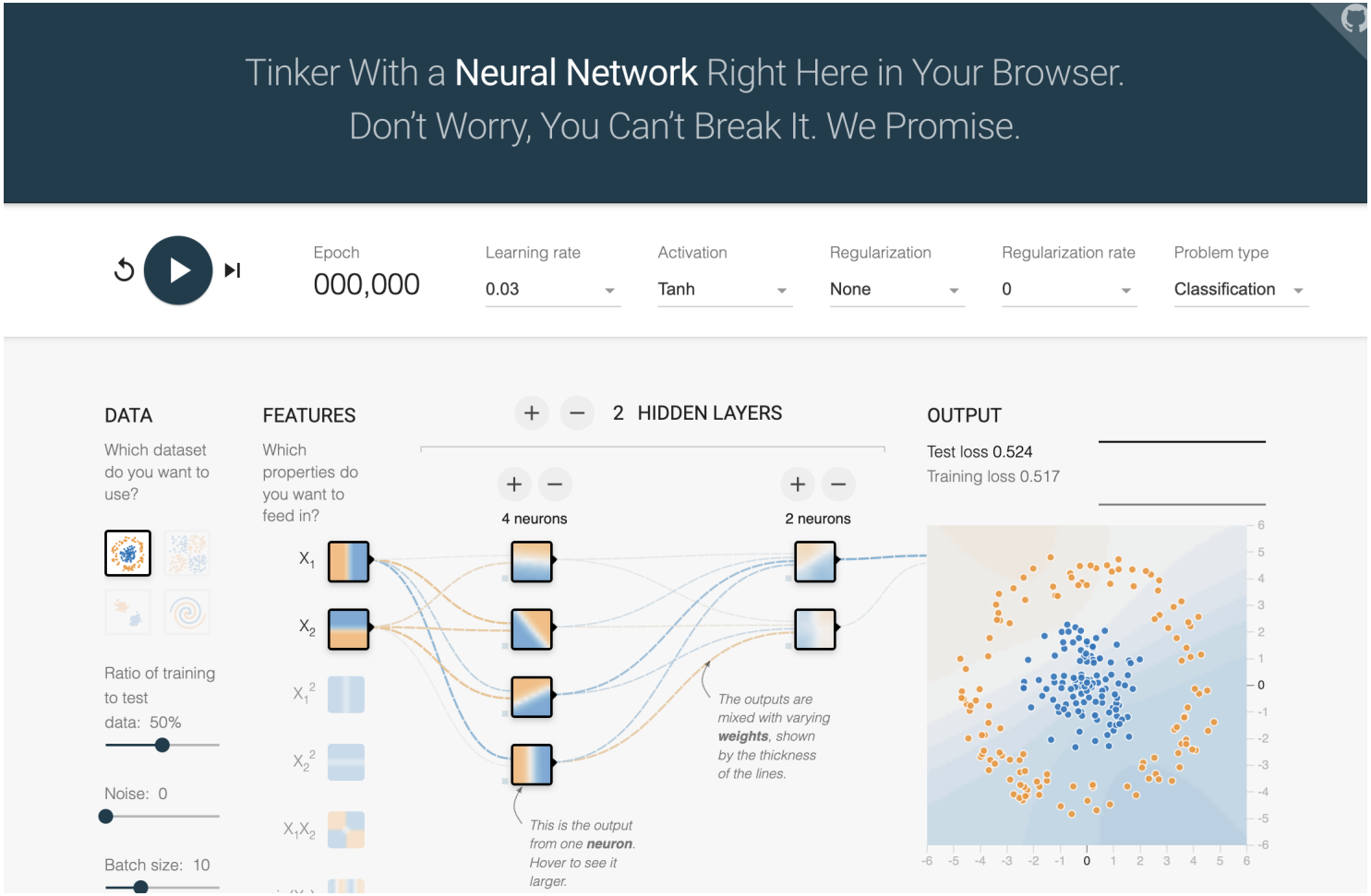
---

First described by Geoff Hinton, who won the Nobel Prize in Physics a couple weeks ago.

These types of neural networks are called "multilayer perceptrons" (MLPs)

They can be surprisingly robust to overfitting!

Demo



https://playground.tensorflow.org/

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