Neural networks

With the new and improved keras flavor

MATH

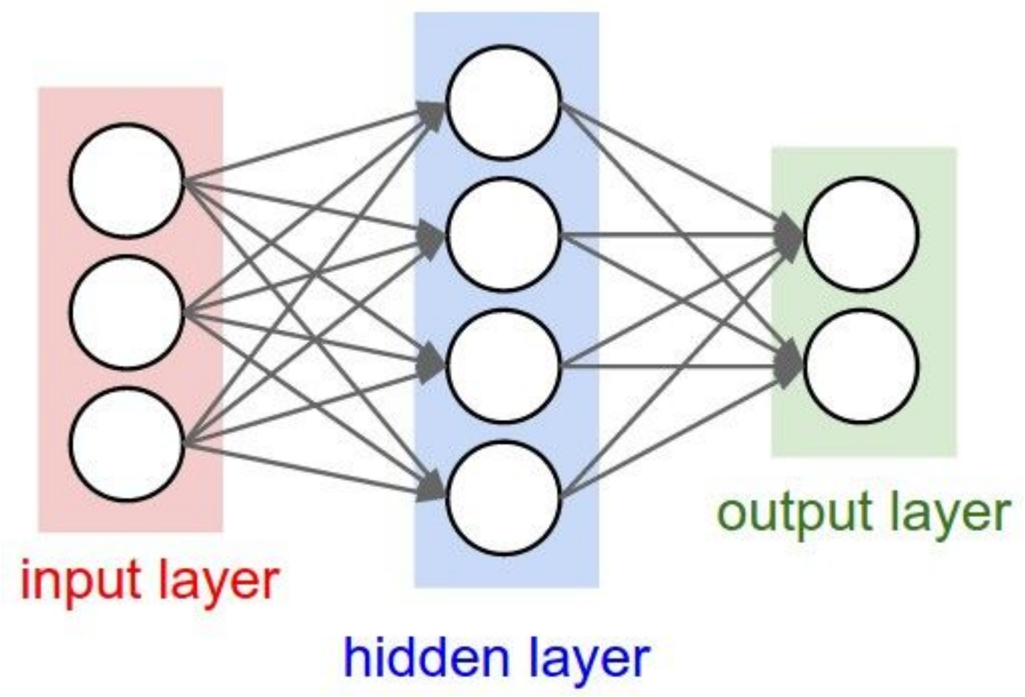
$$\mathbf{z}_1 = \mathbf{x}^T \mathbf{w}_1 + \mathbf{b}_1$$

$$\mathbf{a_1} = \frac{1}{1 + e^{-\mathbf{z}_1}}$$

$$\mathbf{z}_2 = \mathbf{a}_1^T \mathbf{w}_2 + \mathbf{b}_2$$

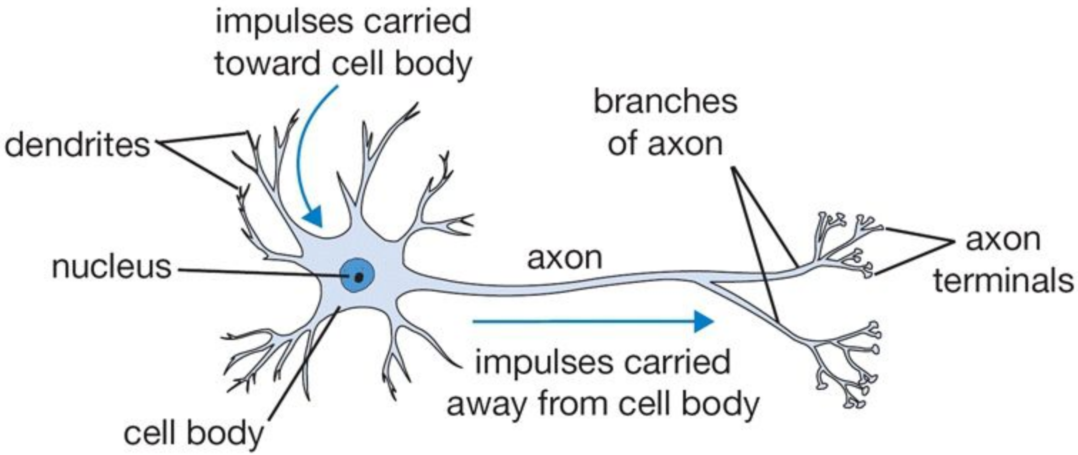
$$\mathbf{o} = \frac{e^{\mathbf{z}_2}}{\sum e^{\mathbf{z}_2}}$$

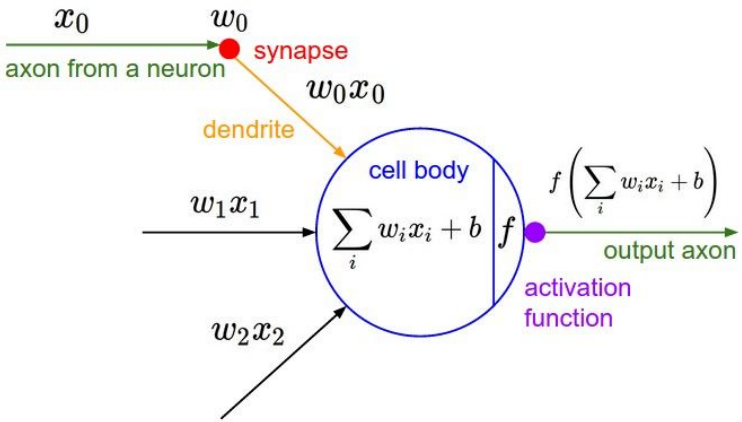
NO MATH



https://cs231n.github.io/neural-networks-1/

Bridging the neural analogy





https://cs231n.github.io/neural-networks-1/

Why neural networks?

Really hard question, but!

1. Versatile
2. Mature
3. Performance
4. Unreasonably stable (good initialization, normalization, regularization)

Lin, H. W., Tegmark, M., & Rolnick, D. (2017). Why Does Deep and Cheap Learning Work So Well? *Journal of Statistical Physics*, *168*(6), 1223–1247. https://doi.org/10.1007/s10955-017-1836-5

Keras

```
from tensorflow import keras



n_hidden = 100

n_classes = 10

model = keras.models.Sequential()

model.add(keras.layers.Flatten(input_shape=(28, 28)))

model.add(keras.layers.Dense(n_hidden, activation="sigmoid"))

model.add(keras.layers.Dense(n_classes, activation="softmax"))
```

Gradient descent

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}(o, t)$$

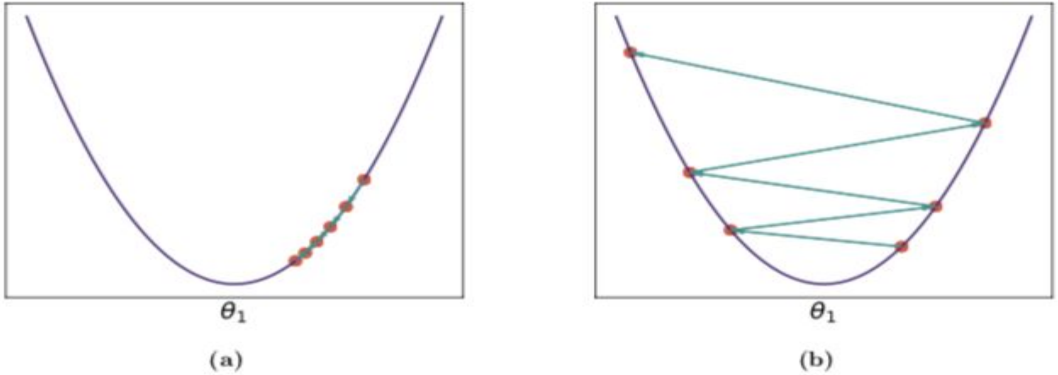


Figure 2.2: Gradient descent on a simple quadratic function showing the effect of too small, **(a)**, and too large, **(b)**, value for the learning rate η

R. solli great unpublished works

Compiling and Fitting

```
model.compile("adam", loss="categorical_crossentropy")

model.fit(x=data, y=targets, metrics="accuracy")

predicted = model.predict(x)
```