Deep Learning

Mohammad Ali Keyvanrad

Lecture 3:A Review of Artificial Neural Networks (2)

OUTLINE

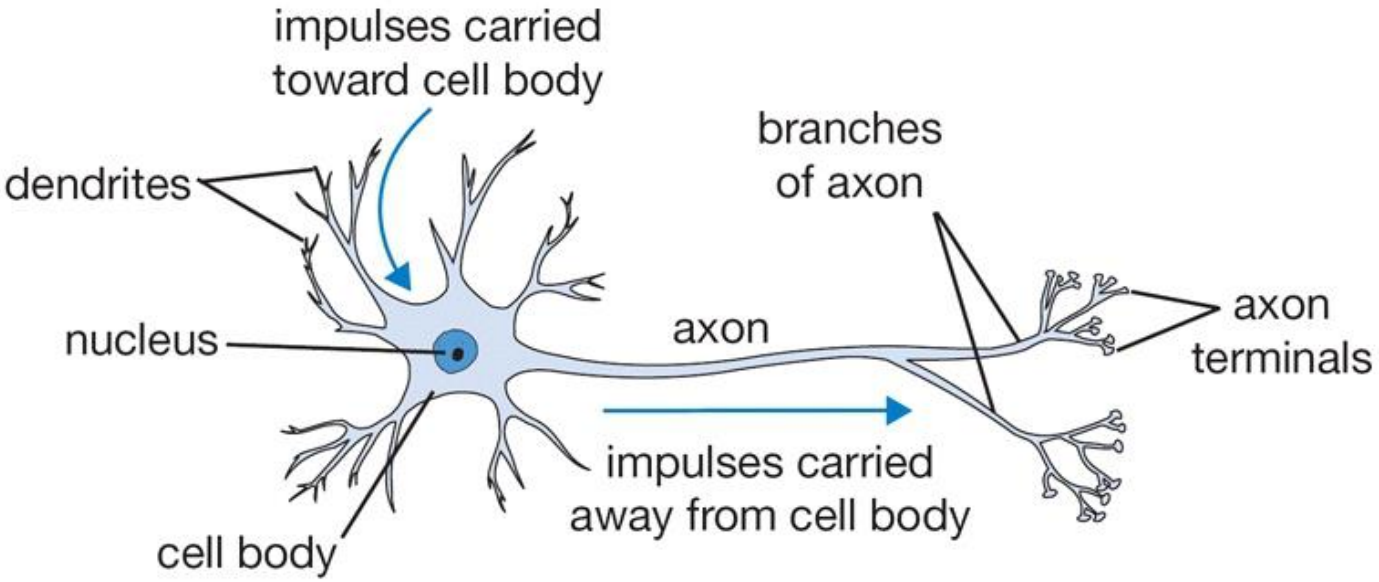
- Activation Functions
  - Sigmoid
  - Tanh
  - ReLU
  - Leaky ReLU
  - ELU
  - Maxout
- Neural Network architectures

OUTLINE

- **Activation Functions**
  - Sigmoid
  - Tanh
  - ReLU
  - Leaky ReLU
  - ELU
  - Maxout
- Neural Network architectures

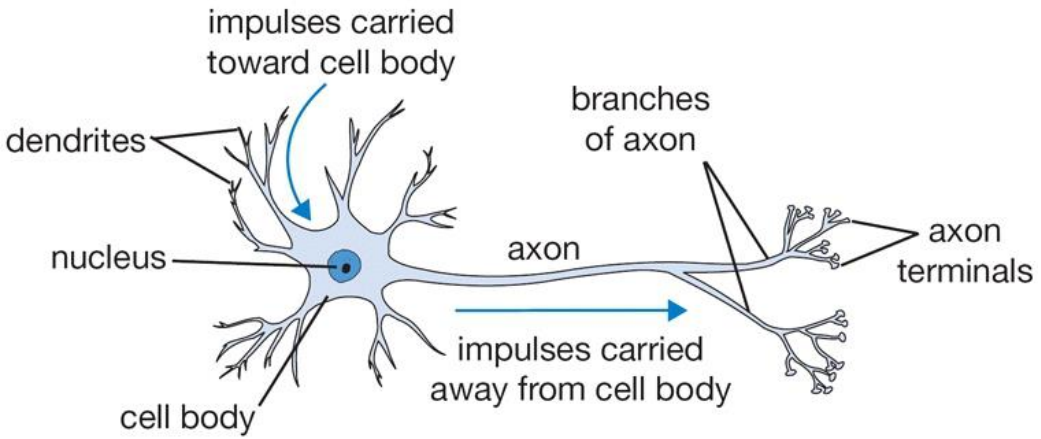
Biological motivation and connections

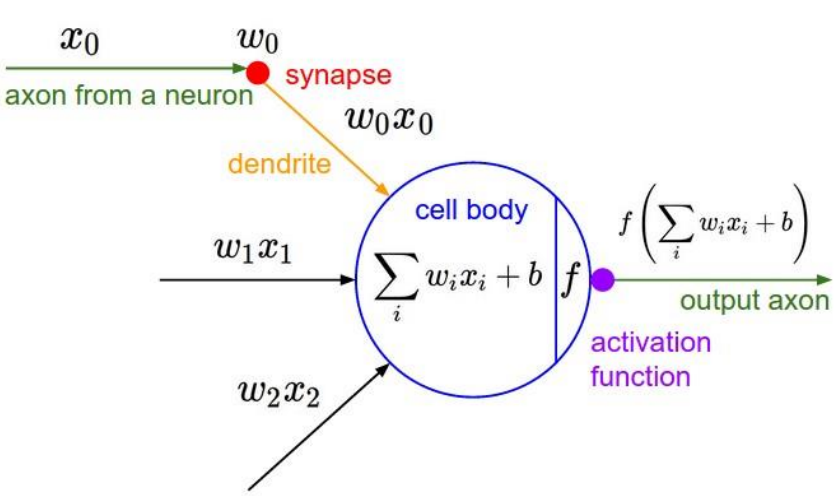
- The basic computational unit of the brain is a neuron.
  - Approximately 86 billion neurons
  - Approximately $10^{14} - 10^{15}$ connections



Biological motivation and connections

- Each neuron
  - receives input signals from its dendrites
  - produces output signals along its (single) axon
- **Activation function f**
  - the firing rate of the neuron

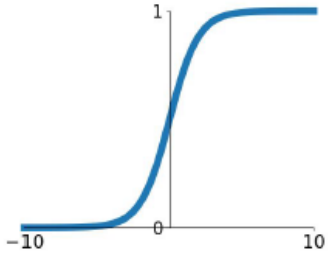




Types of Activation Functions

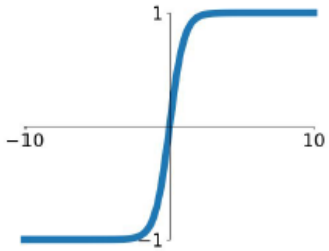
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



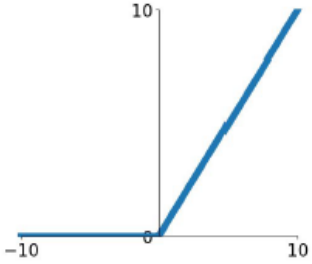
tanh

$$\tanh(x)$$



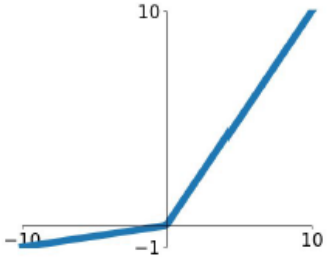
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

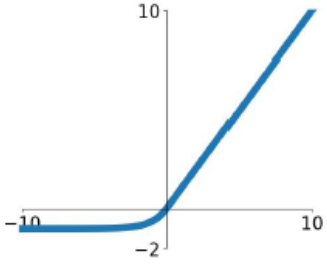


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

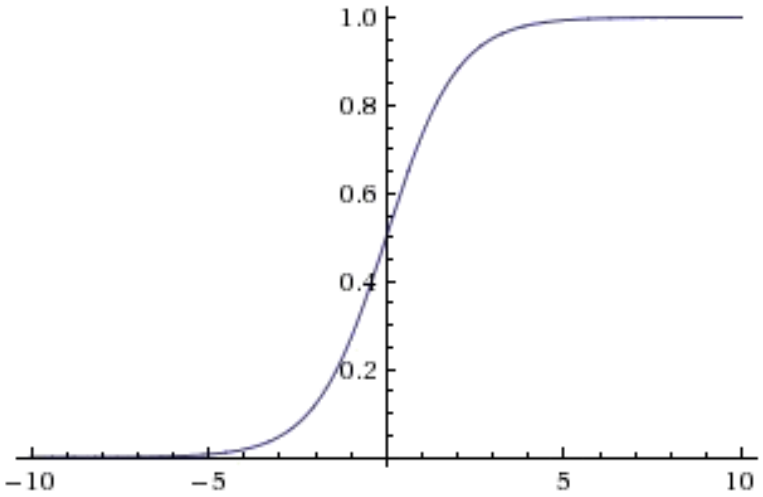


OUTLINE

- Activation Functions
  - **Sigmoid**
  - Tanh
  - ReLU
  - Leaky ReLU
  - ELU
  - Maxout
- Neural Network architectures

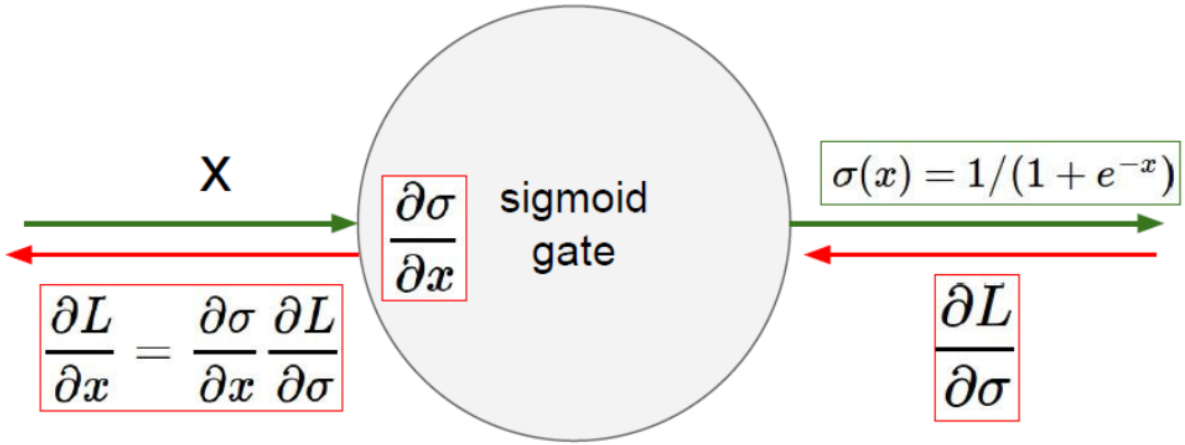
Sigmoid

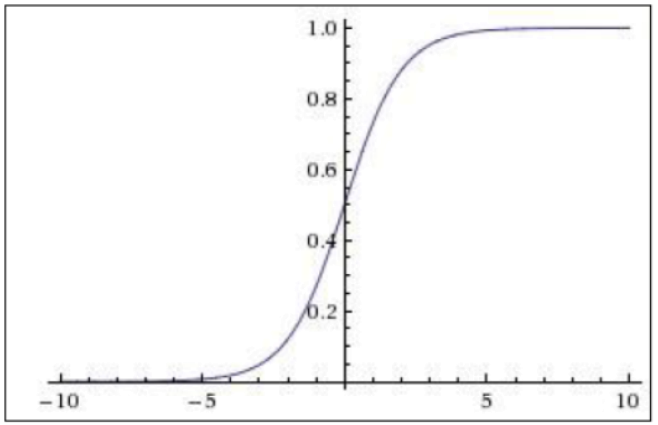
- Mathematical form: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron.



Sigmoid

- **Drawbacks**
  1. Sigmoids saturate and kill gradients
     - if the local gradient is very small, it will "kill" the gradient
     - must pay extra caution for initializing

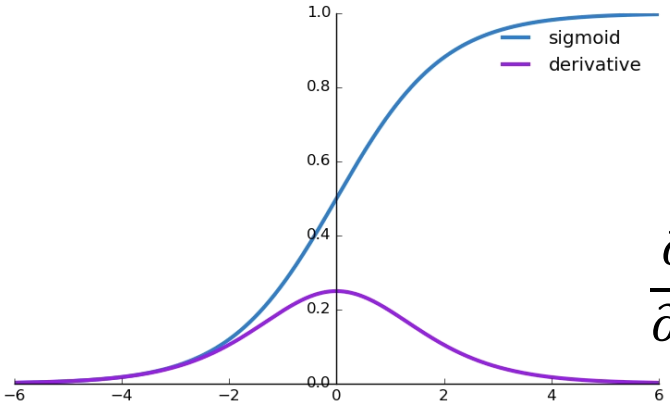




  2. *exp()* is a bit compute expensive

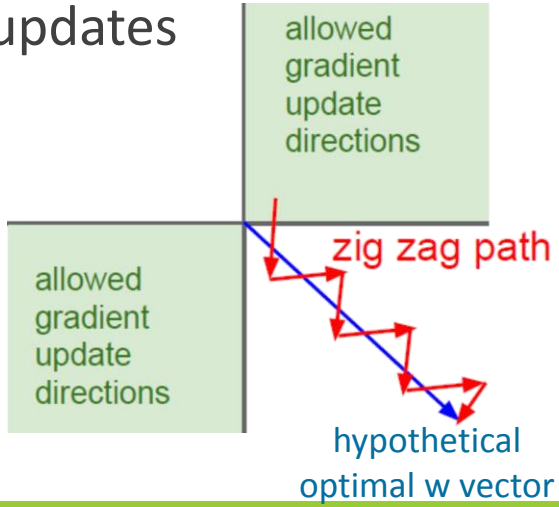
Sigmoid

- **Drawbacks**
  3. Sigmoid outputs are not zero-centered
     - neurons in later layer receiving data that is not zero-centered
     - data coming into a neuron is always positive
     - the gradient on the weights $\boldsymbol{w}$ become either all be positive, or all negative
     - introduce zig-zagging dynamics in the gradient updates



$$f\left(\sum_i w_i x_i + b\right)$$

$$\frac{\partial E}{\partial w_i} = -(t-o)o(1-o)x_i$$

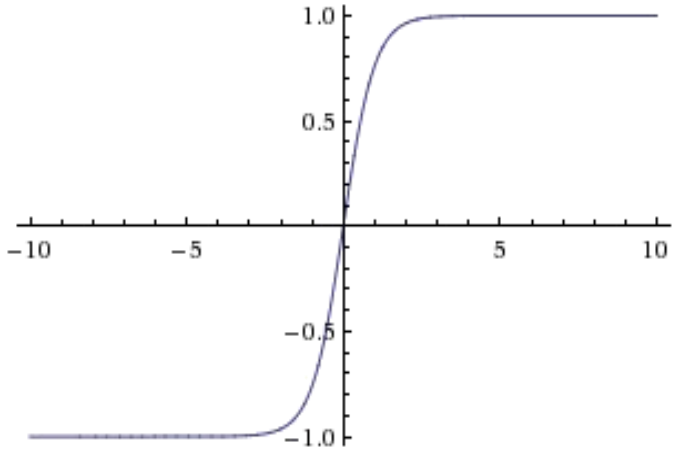


OUTLINE

- Activation Functions
  - Sigmoid
  - **Tanh**
  - ReLU
  - Leaky ReLU
  - ELU
  - Maxout
- Neural Network architectures

Tanh

- Mathematical form: $2\sigma(2x) - 1 = \frac{2}{1+e^{-2x}} - 1$
- Features
  - Squashes numbers to range [-1,1]
  - zero centered (nice)
  - **still kills gradients when saturated**
  - **$tanh()$ is a bit compute expensive**



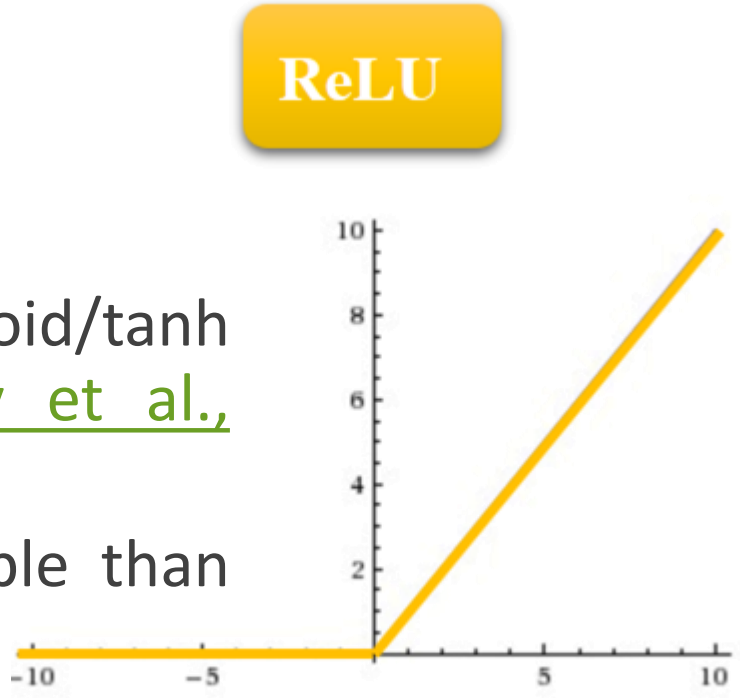
In practice the ***tanh*** *non-linearity is always preferred to the* ***sigmoid*** *nonlinearity*

OUTLINE

- Activation Functions
  - Sigmoid
  - Tanh
  - **ReLU**
  - Leaky ReLU
  - ELU
  - Maxout
- Neural Network architectures

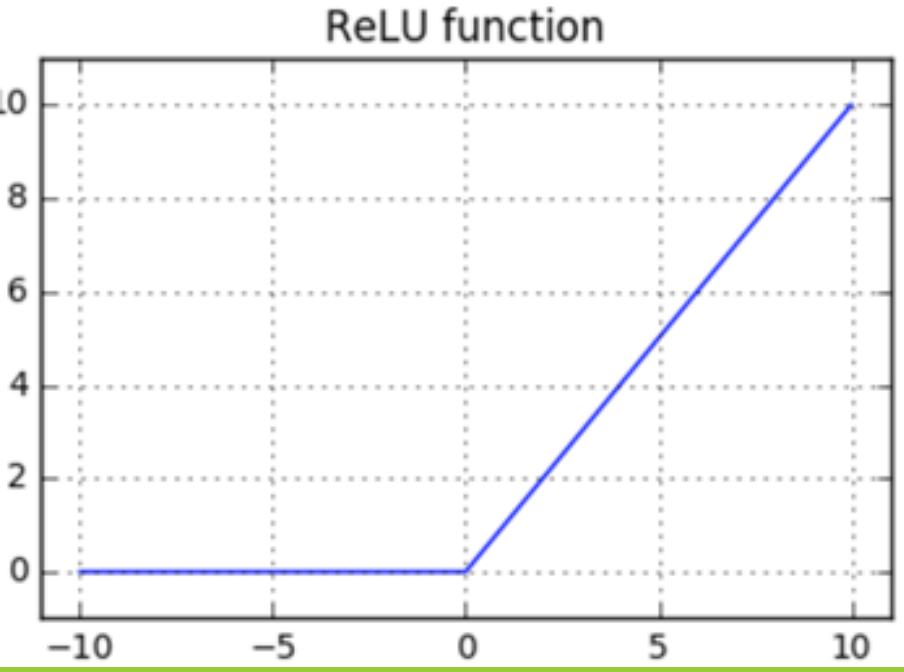
ReLU (Rectified Linear Unit)

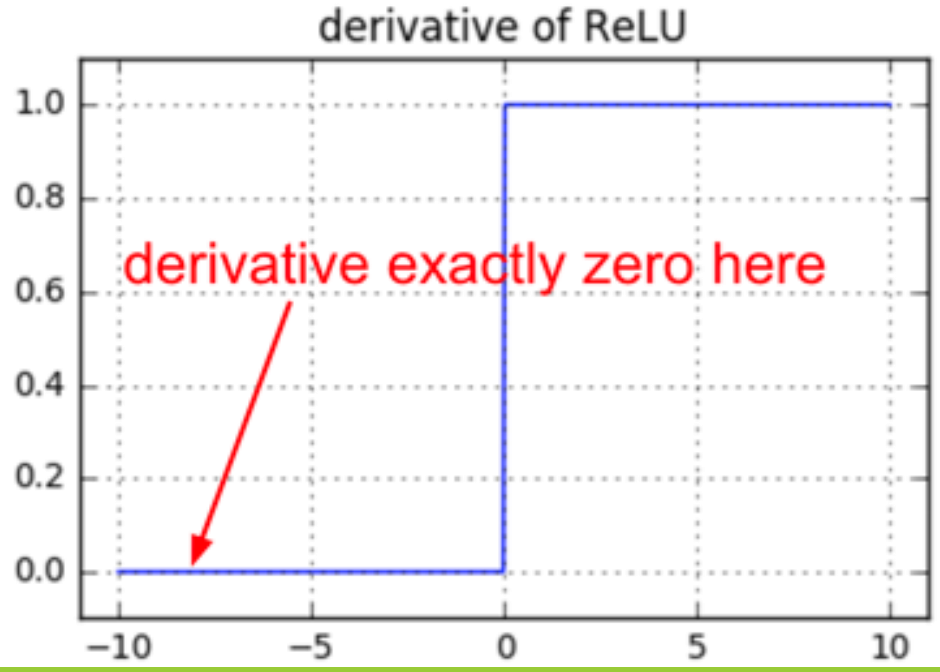
- Mathematical form: $f(x) = \max(0, x)$
- Features
  - Maps numbers to range $[0, +\infty]$
  - Does not saturate (in +region)
  - Very computationally efficient
  - Converges much faster than sigmoid/tanh in practice (e.g. **6x**) [Krizhevsky et al., 2012]
  - Actually more biologically plausible than sigmoid



ReLU (Rectified Linear Unit)

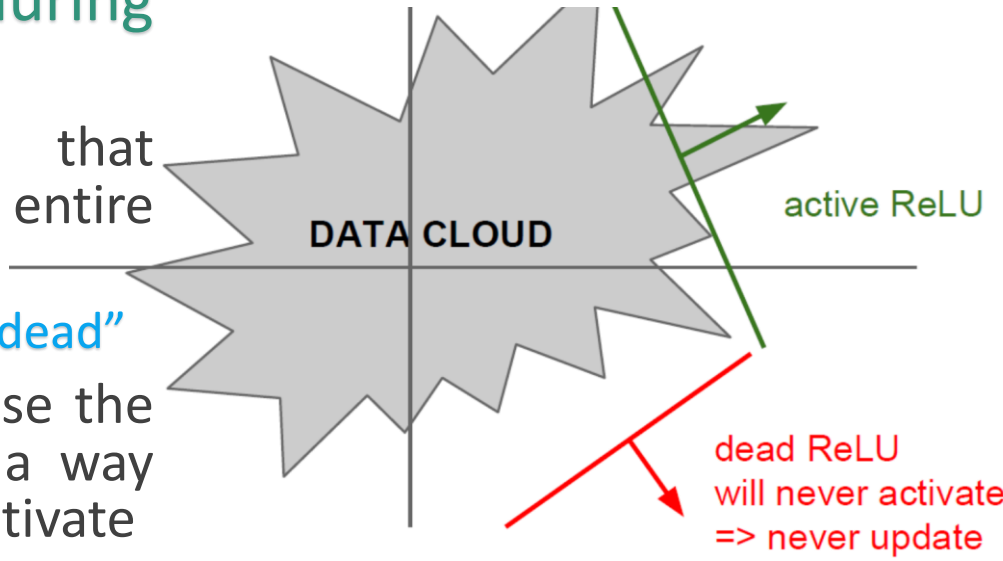
- **Drawbacks**
  1. Not zero-centered output
  2. Saturate (in -region)





ReLU (Rectified Linear Unit)

- **Drawbacks**
  3. ReLU units can "die" during training
     - Dead neurons: neurons that never activate across the entire training dataset
       - 40% of your network can be "dead"
     - A large gradient could cause the weights to update in such a way that the neuron will never activate
     - With a proper setting of the learning rate this is less frequently an issue.

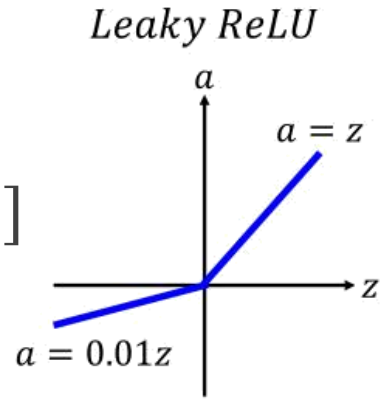


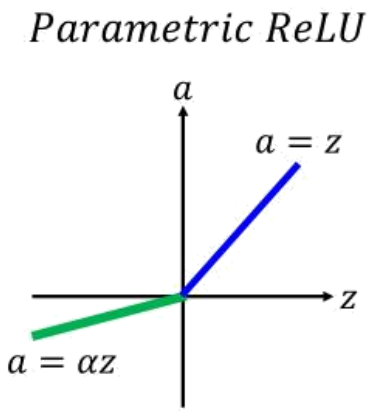
OUTLINE

- Activation Functions
  - Sigmoid
  - Tanh
  - ReLU
  - **Leaky ReLU**
  - ELU
  - Maxout
- Neural Network architectures

Leaky ReLU and Parametric ReLU

- Mathematical form:
  - LReLU: $f(x) = \max(0.01x, x)$
  - PReLU: $f(x) = \max(\alpha x, x)$
- Features
  - Maps numbers to range $[-\infty, +\infty]$
  - Does not saturate
  - Computationally efficient
  - Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
  - Will not "die"
  - Some people report success but the results are not always consistent





α also learned by gradient descent

[Mass et al., 2013]

[He et al., 2015]

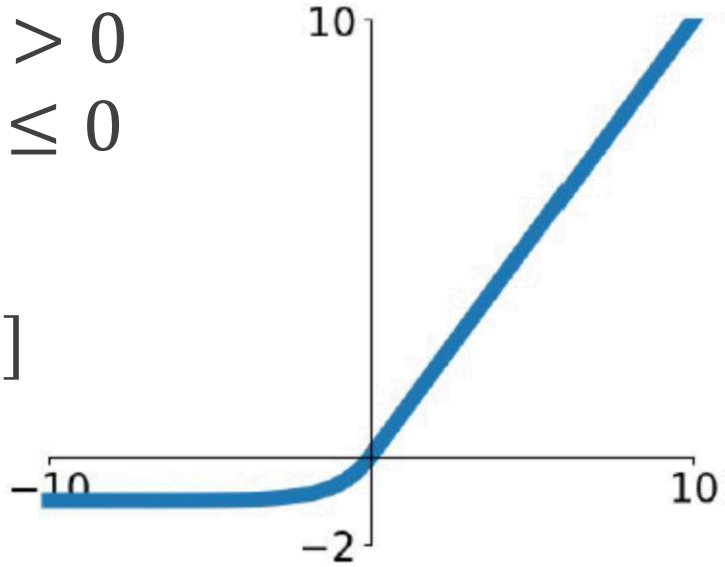
OUTLINE

- Activation Functions
  - Sigmoid
  - Tanh
  - ReLU
  - Leaky ReLU
  - **ELU**
  - Maxout
- Neural Network architectures

Exponential Linear Units (ELU)

- Mathematical form:

$$f(x) = \begin{cases} x & x > 0 \\ \alpha(\exp(x) - 1) & x \leq 0 \end{cases}$$



- Features
  - Maps numbers to range $[-\alpha, +\infty]$
  - All benefits of ReLU
  - Closer to zero mean outputs
  - Negative saturation regime compared with Leaky ReLU adds some robustness to noise

[Clevert et al., 2015]

OUTLINE

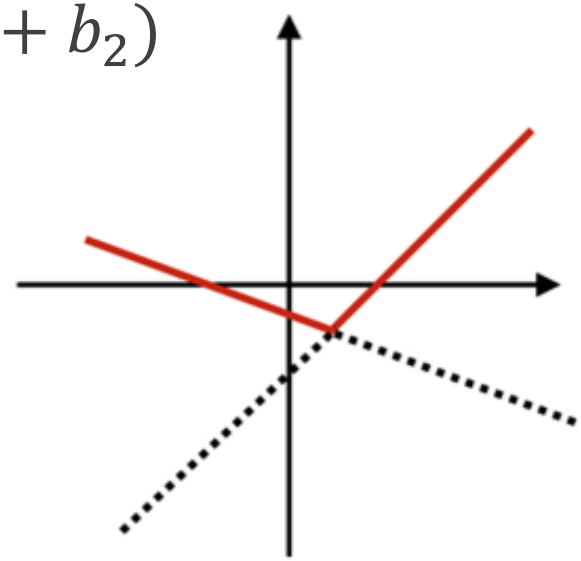
- Activation Functions
  - Sigmoid
  - Tanh
  - ReLU
  - Leaky ReLU
  - ELU
  - **Maxout**
- Neural Network architectures

Maxout

- Mathematical form:

$$f(x) = max(w_1^T x + b_1, w_2^T x + b_2)$$

- Features
  - Maps numbers to range $[?, +\infty]$
  - Generalizes ReLU and Leaky ReLU
  - Linear Regime!
  - Does not saturate!
  - Does not die!
  - **Doubles the number of parameters/neuron**

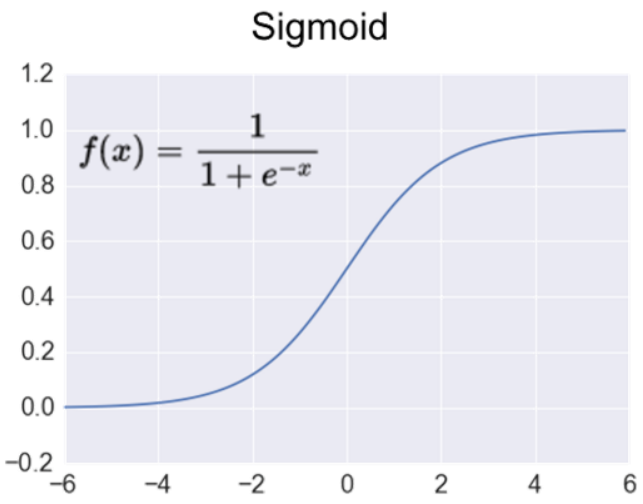


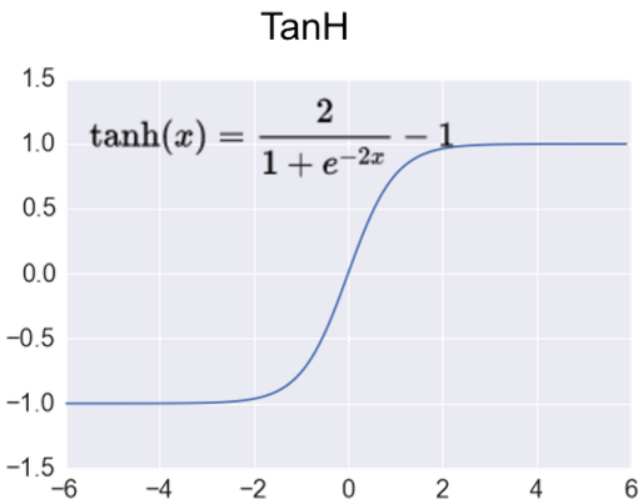
Maxout (n=2)

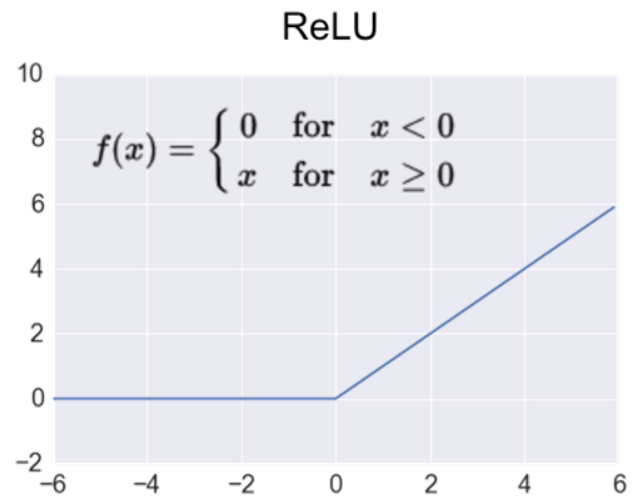
[Goodfellow et al., 2013]

TLDR; In practice

- What neuron type should I use?
  - Use ReLU. Be careful with your learning rates
  - Try out Leaky ReLU / Maxout / ELU
  - Try out tanh but don't expect much
  - Don't use sigmoid







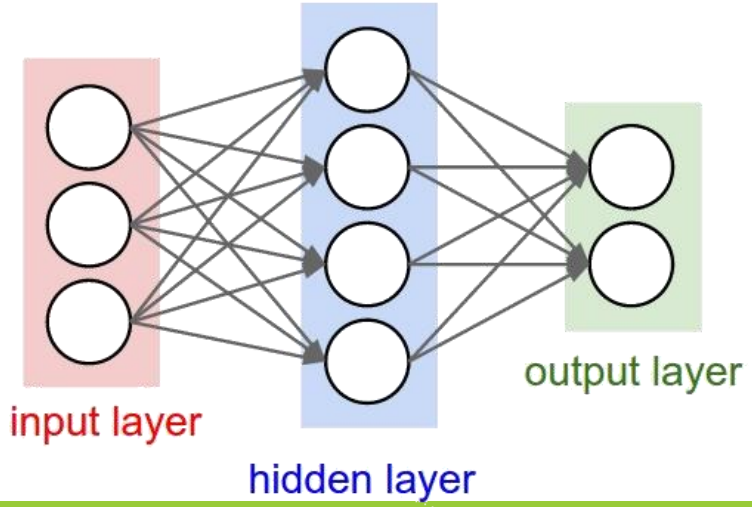
OUTLINE

- Activation Functions
  - Sigmoid
  - Tanh
  - ReLU
  - Leaky ReLU
  - ELU
  - Maxout
- **Neural Network architectures**

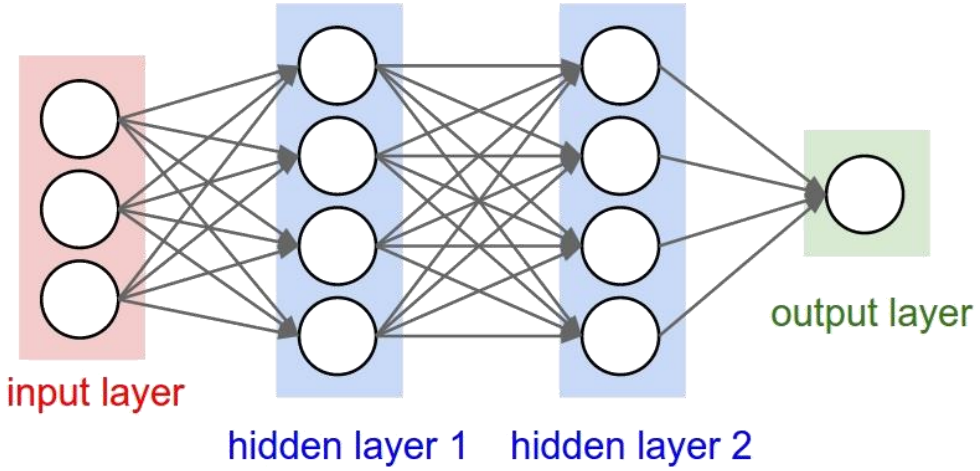
Layer-wise organization

- N-layer neural network
  - we do not count the input layer

A 2-layer Neural Network (3-4-2)
4 + 2 = 6 neurons (not counting the inputs)
[3 x 4] + [4 x 2] = 20 weights
4 + 2 = 6 biases
26 learnable parameters



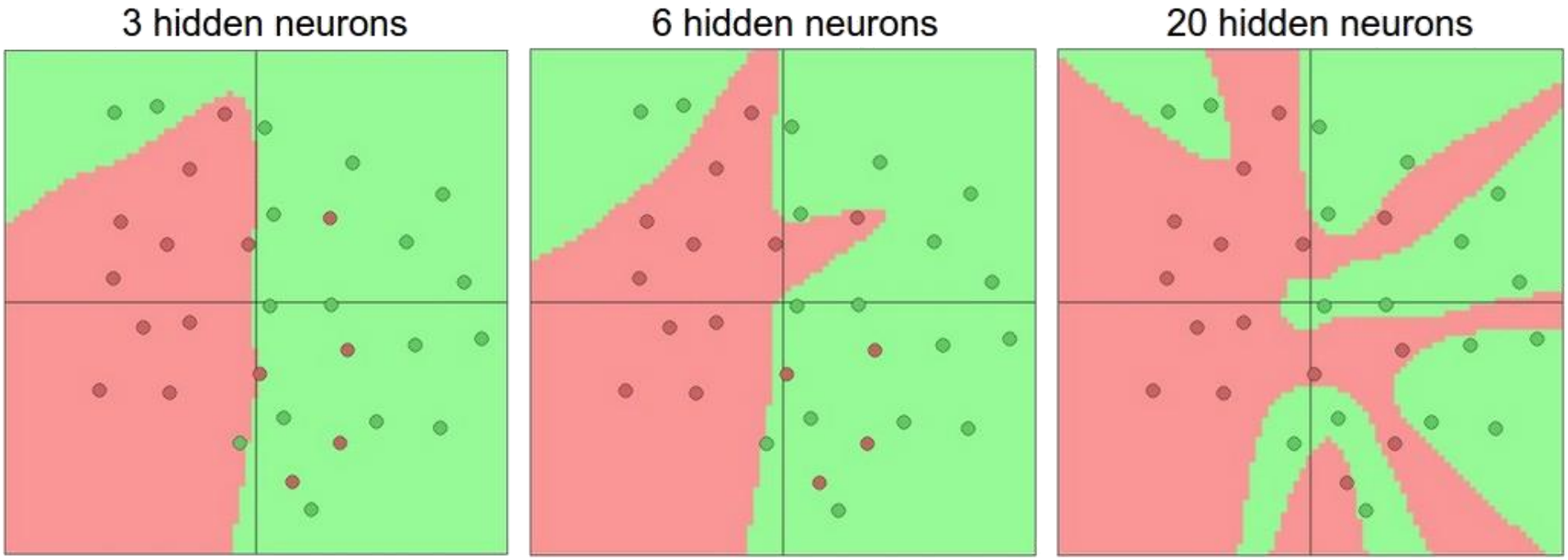
A 3-layer Neural Network (3-4-4-1)
4 + 4 + 1 = 9 neurons
[3 x 4] + [4 x 4] + [4 x 1] = 12 + 16 + 4 = 32 weights
4 + 4 + 1 = 9 biases
41 learnable parameters



Setting number of layers and their sizes

Size and number of layers ↑ ⇒ Capacity of the network ↑

- It is easier to overfit the training data

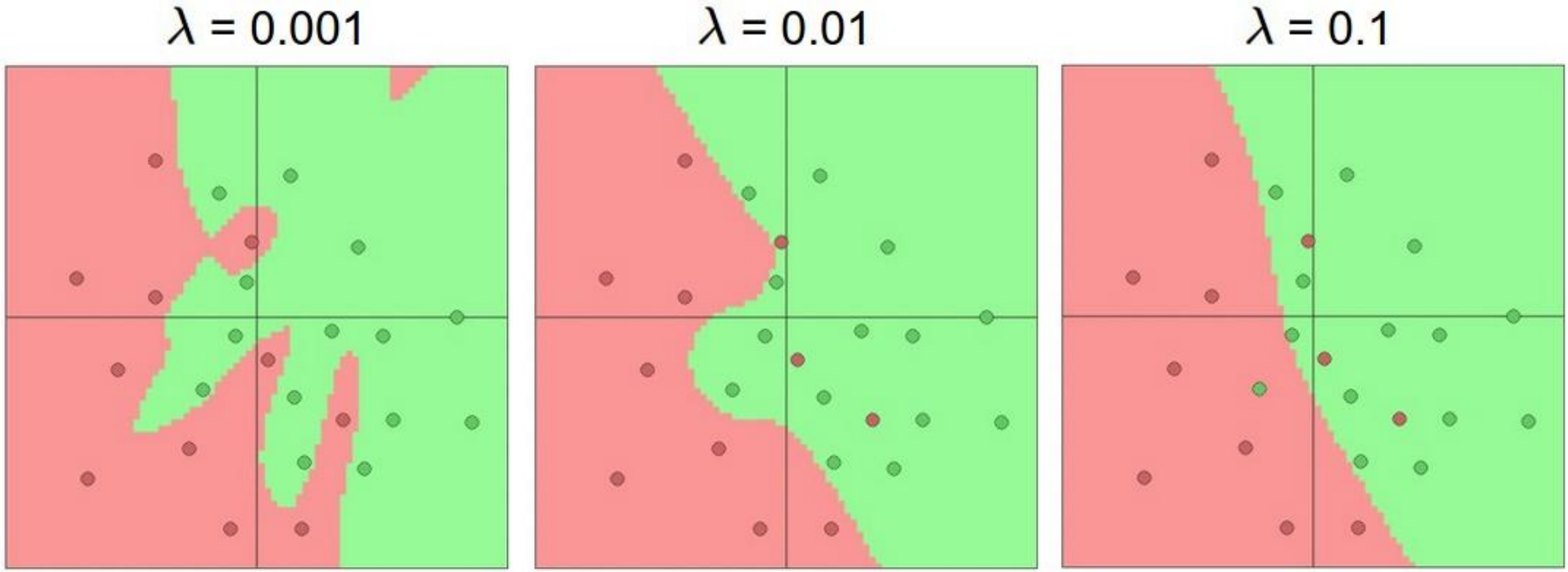


Setting number of layers and their sizes

- It seems that smaller neural networks can be preferred if the data is not complex
  - **This is incorrect**
- There are many other preferred ways to prevent overfitting in Neural Networks
  - L2 regularization
  - Dropout
  - Input noise

Setting number of layers and their sizes

- The effects of regularization strength
  - Each neural network above has 20 hidden neurons



References

- Stanford "Convolutional Neural Networks for Visual Recognition" course (Training Neural Networks, part I)

امام علی (ع):

اِقْبَلْ عُذْرَ أَخِیکَ، وَ إِنْ لَمْ یَکُنْ لَهُ عُذْرٌ فَالْتَمِسْ لَهُ عُذْراً

عذر برادرت را بپذیر و اگر عذری نداشت، عذری برایش بتراش.

Accept your brother's apology, and even if he has no excuse, you bring him an excuse.

بحارالأنوار، ج ۷۴، ص ۱۶۵