Neural Networks (part 1)

Most of the materials are taken from here

Nastaran Okati

Outline

- ▷ Introduction
- ▷ Modeling the Neuron
- ▷ Neurons as Classifiers
- ▷ Activation Functions
- ▷ Neural Network Architecture

1.
Introduction

Previously in our course...

- We computed scores for different visual categories given the image using the formula s=Wx, where W was a matrix and x was an input column vector containing all pixel data of the image

- In the case of CIFAR-10, x is a [3072x1] column vector, and W is a [10x3072] matrix, so that the output scores is a vector of 10 class scores.

Simple Neural Network

- An example neural network would instead compute $s = W_2 \max(0, W_1 x)$. Here, W1 could be, for example, a [100x3072] matrix transforming the image into a 100-dimensional intermediate vector. The function max(0,−) is a non-linearity that is applied elementwise.

- There are several choices we could make for the non-linearity

- W2 would then be of size [10x100], so that we again get 10 numbers out that we interpret as the class scores.

Simple Neural Network

- Why should we add non-linearity?

- Two matrices could be collapsed to a single matrix, and therefore the predicted class scores would again be a linear function of the input.

- The parameters W2,W1 are learned with stochastic gradient descent, and their gradients are derived with chain rule and computed with backpropagation.

- An example 3-layer Neural Network?

2.
Modeling the Neuron

Biological Motivation

- The area of Neural Networks has originally been primarily inspired by the goal of modeling biological neural systems.
- The basic computational unit of the brain is a **neuron**.
- Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14} - 10^{15} **synapses**.

Biological Motivation

- Each neuron receives input signals from its dendrites and produces output signals along its (single) axon.
- The axon eventually branches out and connects via synapses to dendrites of other neurons.
- In the computational model of a neuron, the signals that travel along the axons (e.g. $x0$) interact multiplicatively (e.g. $w0x0$) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g. $w0$)

Biological Motivation

- The synaptic strengths (the weights w) are learnable and control the strength of influence (and its direction: excitory (positive weight) or inhibitory (negative weight)) of one neuron on another.

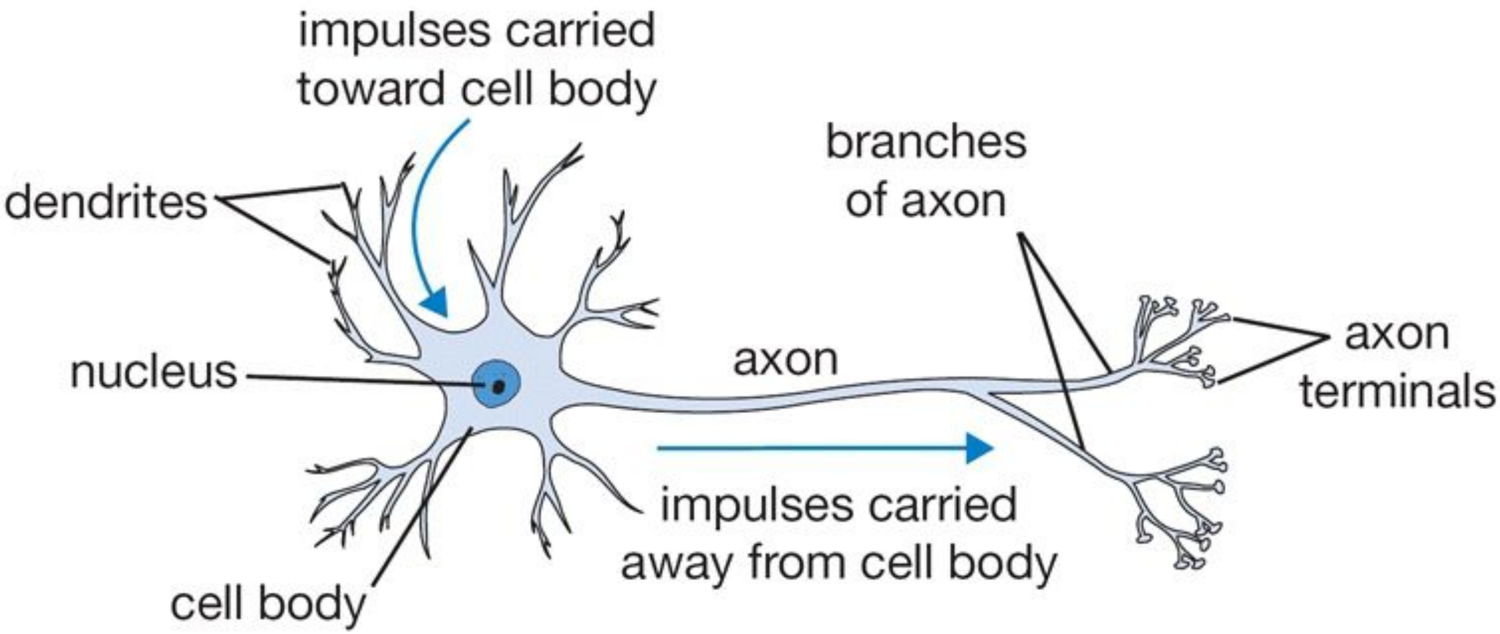
- In the case of CIFAR-10, x is a [3072x1] column vector, and W is a [10x3072] matrix, so that the output scores is a vector of 10 class scores.

Biological Motivation

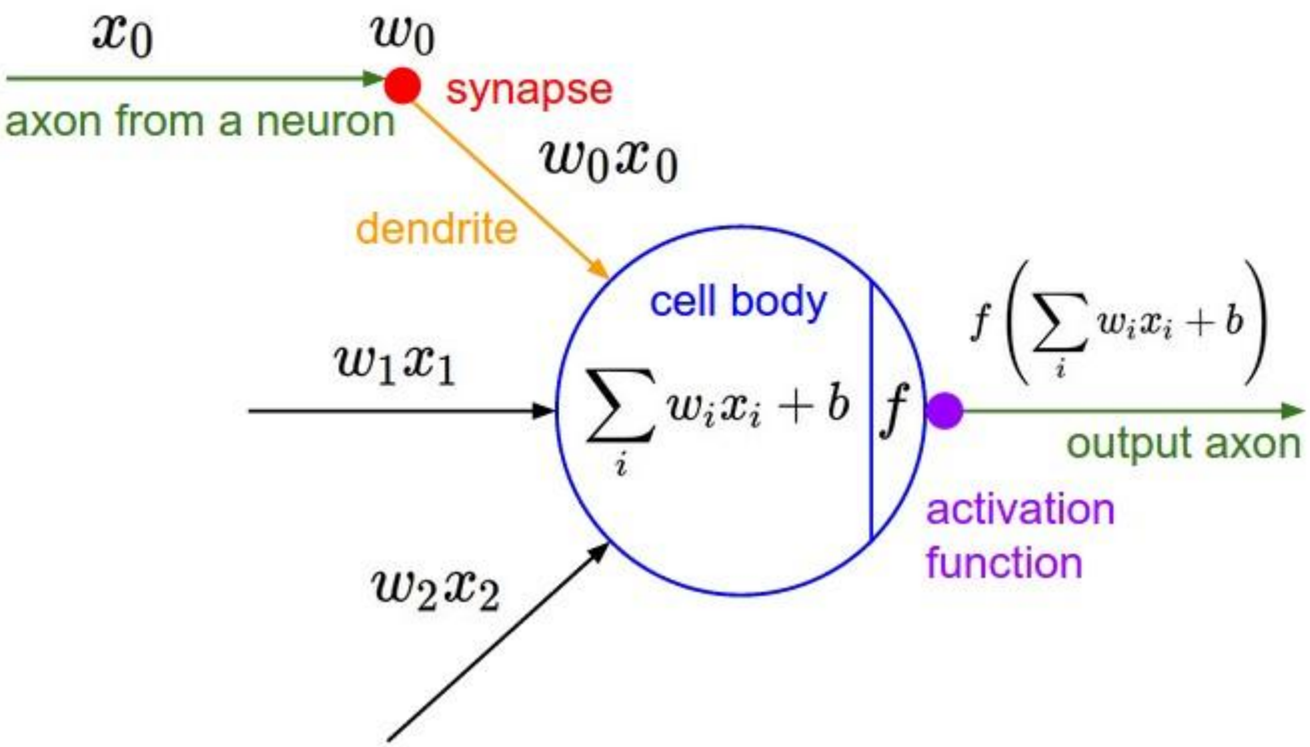
- The dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon.

- In the computational model, we assume that the precise timings of the spikes do not matter, and that only the frequency of the firing communicates information.

Biological Neuron



Mathematical Neuron



Don’t mess with Neuroscientists!

- There are many different types of neurons, each with different properties.
- The dendrites in biological neurons perform complex nonlinear computations.
- The synapses are not just a single weight, they’re a complex non-linear dynamical system.
- The exact timing of the output spikes in many systems is known to be important.

3.

Neurons as Classifiers

Single Neuron as a Linear Classifier

- a neuron has the capacity to “like” (activation near one) or “dislike” (activation near zero) certain linear regions of its input space.
- With an appropriate loss function on the neuron’s output, we can turn a single neuron into a linear classifier.

Binary Softmax Classifier

- we can interpret $\sigma(\sum_i w_i x_i + b)$ to be the probability of one of the classes
  $P(y_i = 1 \mid x_i; w)$

- The probability of the other class would be
  $P(y_i = 0 \mid x_i; w) = 1 - P(y_i = 1 \mid x_i; w)$
  since they must sum to one.

- We can formulate the cross-entropy loss and optimizing it would lead to a binary Softmax classifier (also known as logistic regression).

Binary SVM Classifier

- We could attach a max-margin hinge loss to the output of the neuron and train it to become a binary Support Vector Machine.

4.
Activation Functions

Commonly Used Activation Functions

- Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it.

- Commonly used activation functions are:
  - Sigmoid
  - Tanh
  - ReLU
  - Leaky ReLU
  - Maxout

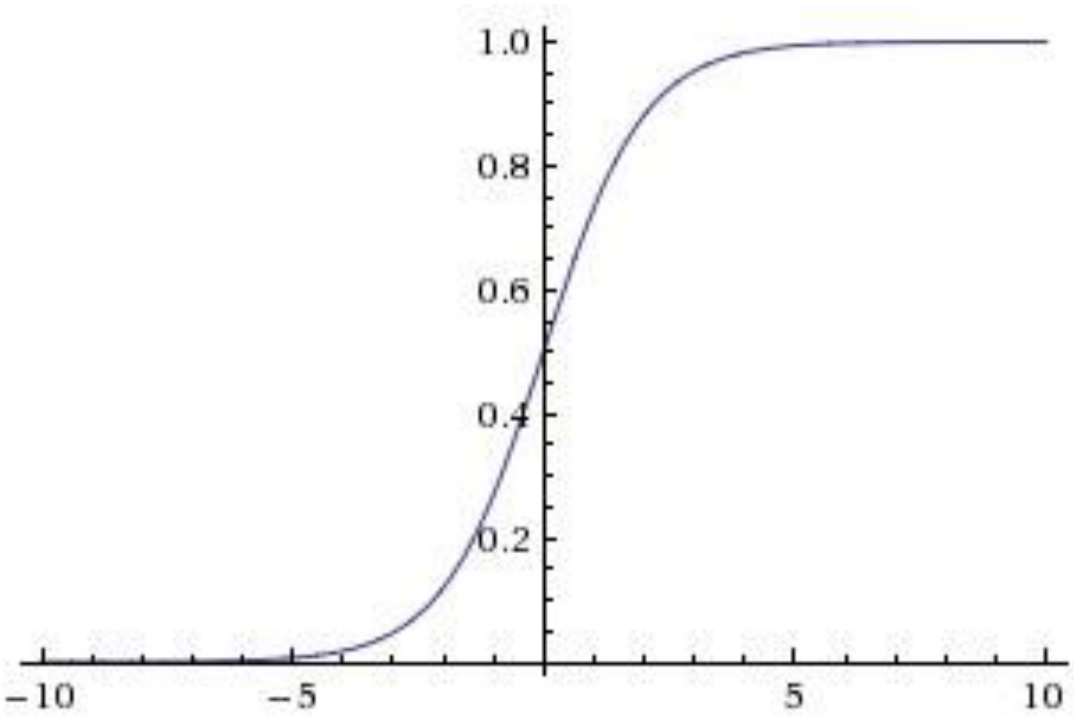
Sigmoid

- It takes a real-valued number and "squashes" it into range between 0 and 1.

- large negative numbers become 0 and large positive numbers become 1

- Drawbacks:
  - Sigmoids saturate and kill gradients.
  - Sigmoid outputs are not zero-centered.

Sigmoid



Sigmoid kills the gradient

- When the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero.

- During backpropagation, this (local) gradient will be multiplied to the gradient of this gate's output for the whole objective.

- If the local gradient is very small, it will effectively "kill" the gradient and almost no signal will flow through the neuron to its weights and recursively to its data

Sigmoid outputs are not zero centered

- Neurons in later layers of processing in a Neural Network would be receiving data that is not zero-centered

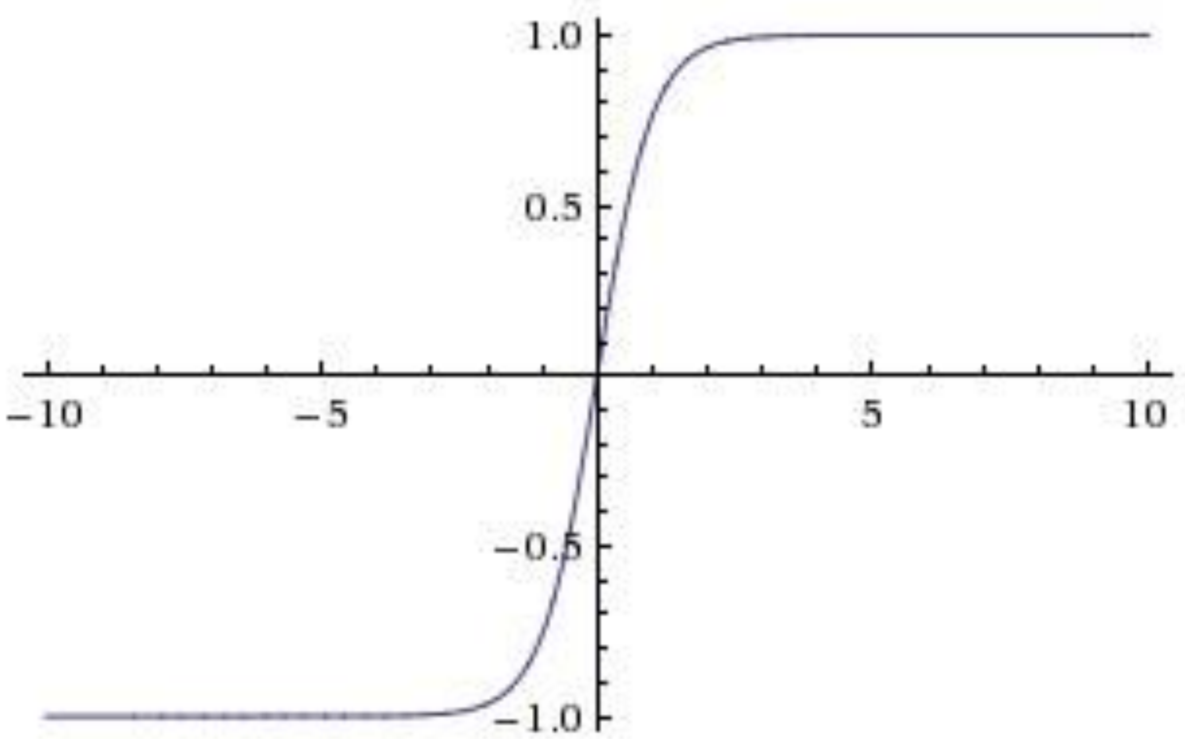
- If the data coming into a neuron is always positive then the gradient on the weights w will during backpropagation become either all positive, or all negative (depending on the gradient of the whole expression f).

- This could introduce undesirable zig-zagging dynamics in the gradient updates for the weights.

Tanh

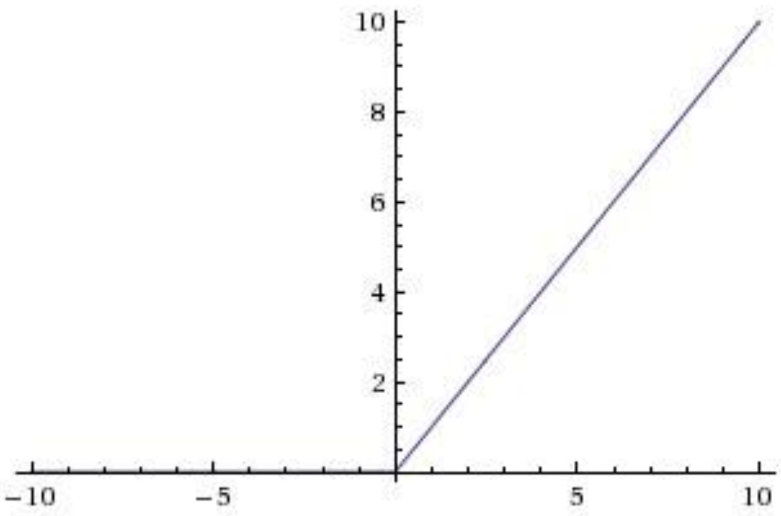
- It squashes a real-valued number to the range [-1, 1].
- Like the sigmoid neuron, its activations saturate, but unlike the sigmoid neuron its output is zero-centered.
- tanh non-linearity is always preferred to the sigmoid nonlinearity.

Tanh



ReLU

- The Rectified Linear Unit has become very popular in the last few years.
- It computes the function f(x)=max(0,x).



ReLU Pros

- Greatly accelerate the convergence of stochastic gradient descent compared to the sigmoid/tanh functions. (due to its linear, non-saturating form.)

- Compared to tanh/sigmoid neurons that involve expensive operations the ReLU can be implemented by simply thresholding a matrix of activations at zero.

ReLU Cons

- ReLU units can be fragile during training and can “die”
- A large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any datapoint again.
- If this happens, then the gradient flowing through the unit will forever be zero from that point on.
- How to fix this?

Leaky ReLU

- One attempt to fix the “dying ReLU” problem

- Instead of the function being zero when x < 0, a leaky ReLU will instead have a small negative slope (of 0.01, or so).

$$f(x) = \mathbb{1}(x < 0)(\alpha x) + \mathbb{1}(x >= 0)(x)$$

Maxout

- Maxout neuron generalizes the ReLU and its leaky version

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

- both ReLU and Leaky ReLU are a special case of this form

- The Maxout neuron enjoys all the benefits of a ReLU unit and does not have its drawbacks

- What is the drawback?

Final words on activations

- It is very rare to mix and match different types of neurons in the same network, even though there is no fundamental problem with doing so.
- Use the ReLU non-linearity, be careful with your learning rates and possibly monitor the fraction of “dead” units in a network.
- Give Leaky ReLU or Maxout a try. Never use sigmoid. Try tanh, but expect it to work worse than ReLU/Maxout.

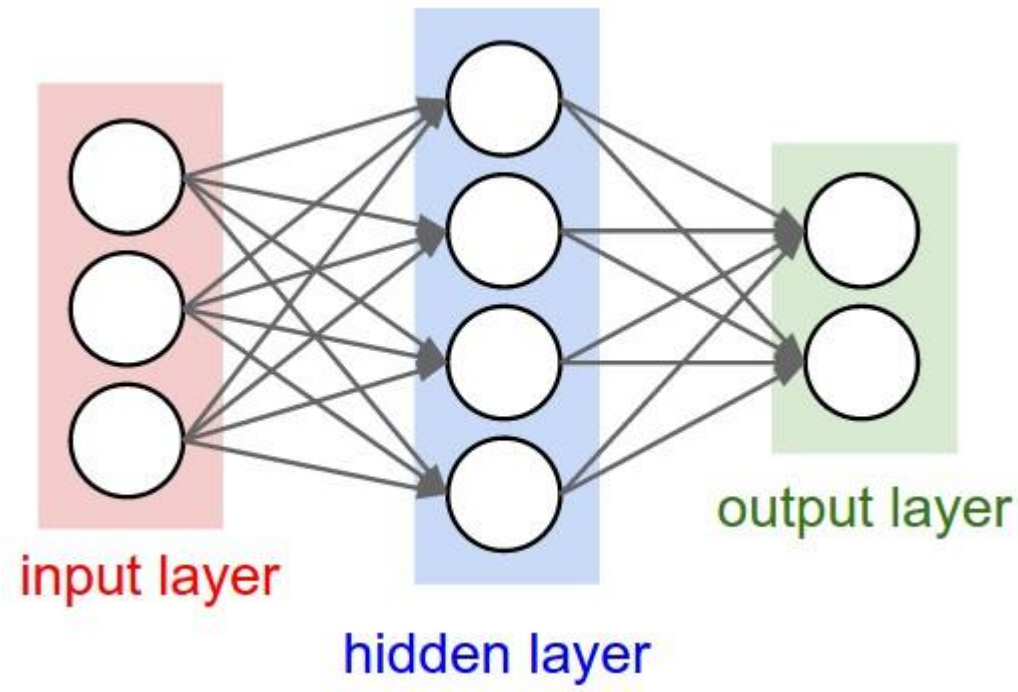
5.

Neural Network Architectures

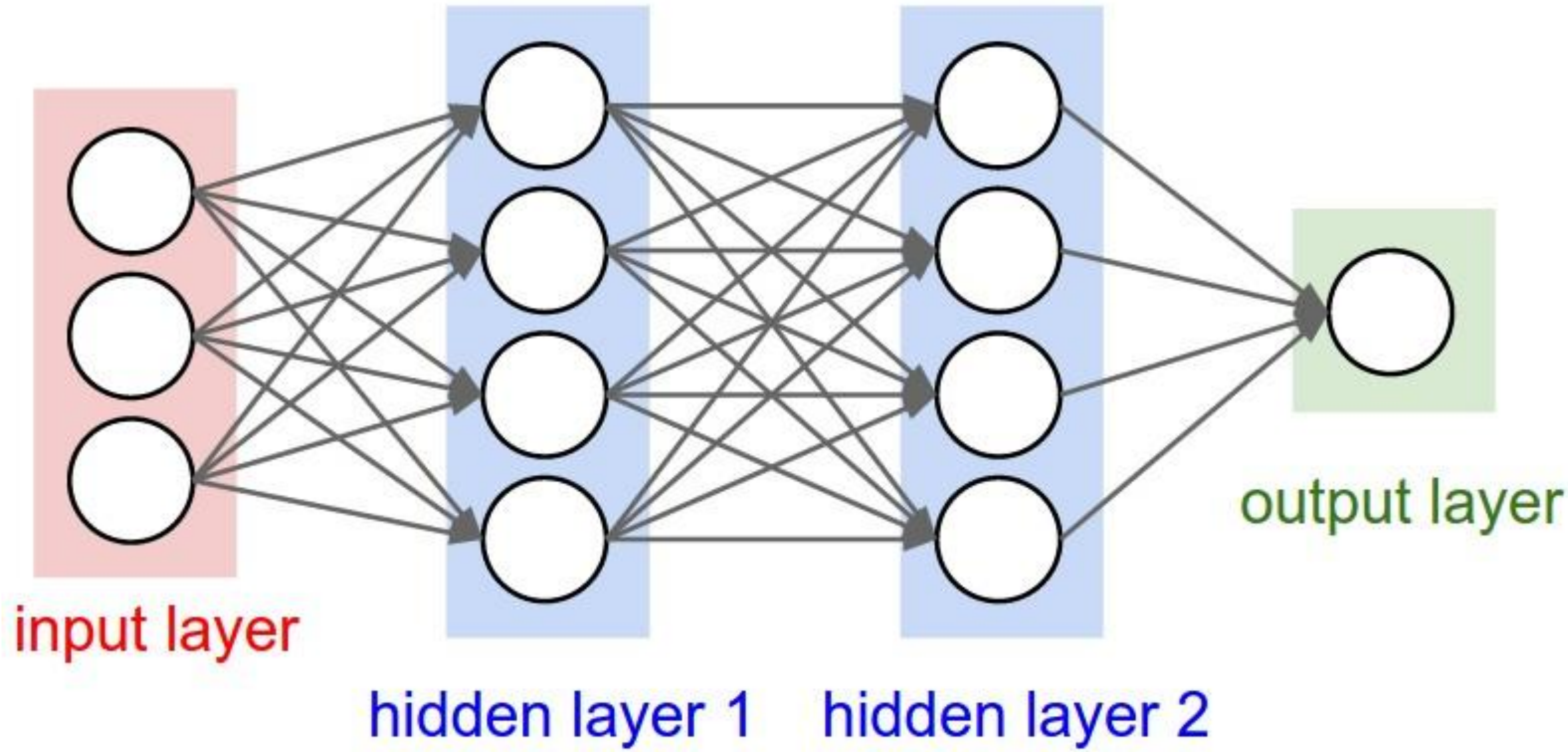
Neural Networks as graphs

- Neural Networks are modeled as collections of neurons that are connected in an acyclic graph
- The outputs of some neurons can become inputs to other neurons
- Neural Network models are often organized into distinct layers of neurons.
- The most common layer type is the fully-connected layer

A 2-layer NN



A 3-layer NN



Output Layer

- The output layer neurons most commonly do not have an activation function

- The last output layer is usually taken to represent the class scores (e.g. in classification), which are arbitrary real-valued numbers, or some kind of real-valued target (e.g. in regression).

Sizing of NN

- The two metrics that people commonly use to measure the size of neural networks are the number of neurons, or more commonly the number of parameters.

- What is the size of the two networks mentioned previously?

- Modern Convolutional Networks contain on orders of 100 million parameters and are usually made up of approximately 10-20 layers (deep learning)

Feed-forward computation

- Layered structure makes it very simple and efficient to evaluate Neural Networks using matrix vector operations.

- What are the vectors and matrices of the previous example?

- What are the learnable parameters?

- Notice that the final Neural Network layer usually doesn't have an activation function

Representational Power

- NNs define a family of functions that are parameterized by the weights of the network
- NNs with at least one hidden layer are universal approximators
- Given any continuous function $f(x)$ and some $\epsilon>0$, there exists a NN, $g(x)$ with one hidden layer such that $\forall x, |f(x)-g(x)|<\epsilon$.
- In other words, the neural network can approximate any continuous function.

Why more layers?!

- the fact that a two-layer NN is a universal approximator is a weak statement in practice.
- There are many universal approximators which are useless in ML
- NNs work well because they compactly express smooth functions that fit well with the data and are easy to learn
- Empirically deeper networks can work better than a single-hidden-layer networks, although their representational power is equal.

Why more layers?!

- In practice it is often the case that 3-layer neural networks will outperform 2-layer nets, but going even deeper (4,5,6-layer) rarely helps much more.

- In contrast, in Convolutional Networks, depth has been found to be an extremely important component for a good recognition system (e.g. on order of 10 learnable layers).

- Images contain hierarchical structure (e.g. faces are made up of eyes, which are made up of edges, etc.), so several layers of processing make intuitive sense for this data domain.

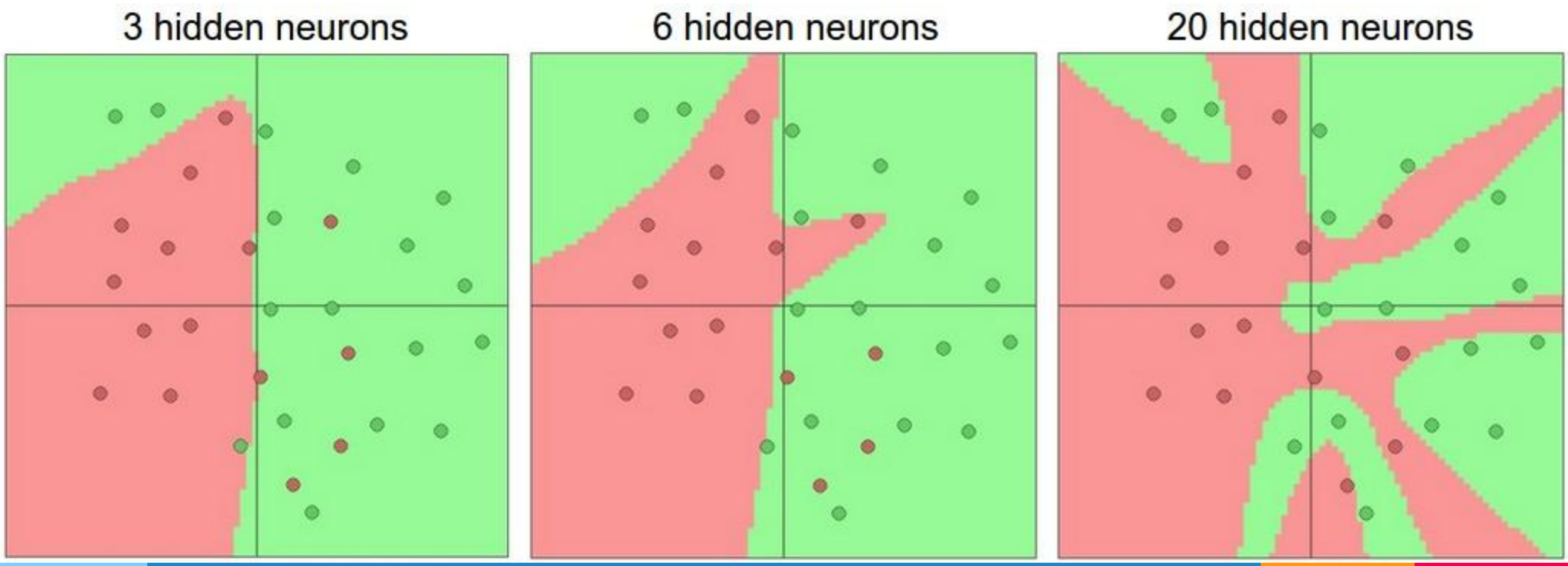
Setting number of layers

- What architecture should we use? How large should each layer be?
- As we increase the size and number of layers in a Neural Network, the capacity of the network increases.
- That is, the space of representable functions grows since the neurons can collaborate to express many different functions

Effect of capacity

- Neural Networks with more neurons can express more complicated functions

- Which one is better?



Generalization

- Should we use as many layers as possible?
- Overfitting occurs when a model with high capacity fits the noise in the data instead of the underlying relationship
- The model with 20 hidden neurons fits all the training data but at the cost of segmenting the space into many disjoint red and green decision regions.
- The model with 3 hidden neurons only classifies the data in broad strokes

How many layers?!

- Should we use as few layers as possible?
- It seems that smaller NNs can be preferred if the data is not complex enough to prevent overfitting
- This is incorrect - there are many other preferred ways to prevent overfitting in NNs (such as L2 regularization, dropout, input noise)
- In practice, it is always better to use these methods to control overfitting instead of the number of neurons.

How many layers?!

- Smaller networks are harder to train with local methods such as Gradient Descent
- Their loss functions have few local minima, but it turns out that many of these minima are easier to converge to, and that they are bad
- Bigger NNs contain significantly more local minima, but these minima turn out to be much better in terms of their actual loss.

How many layers?!

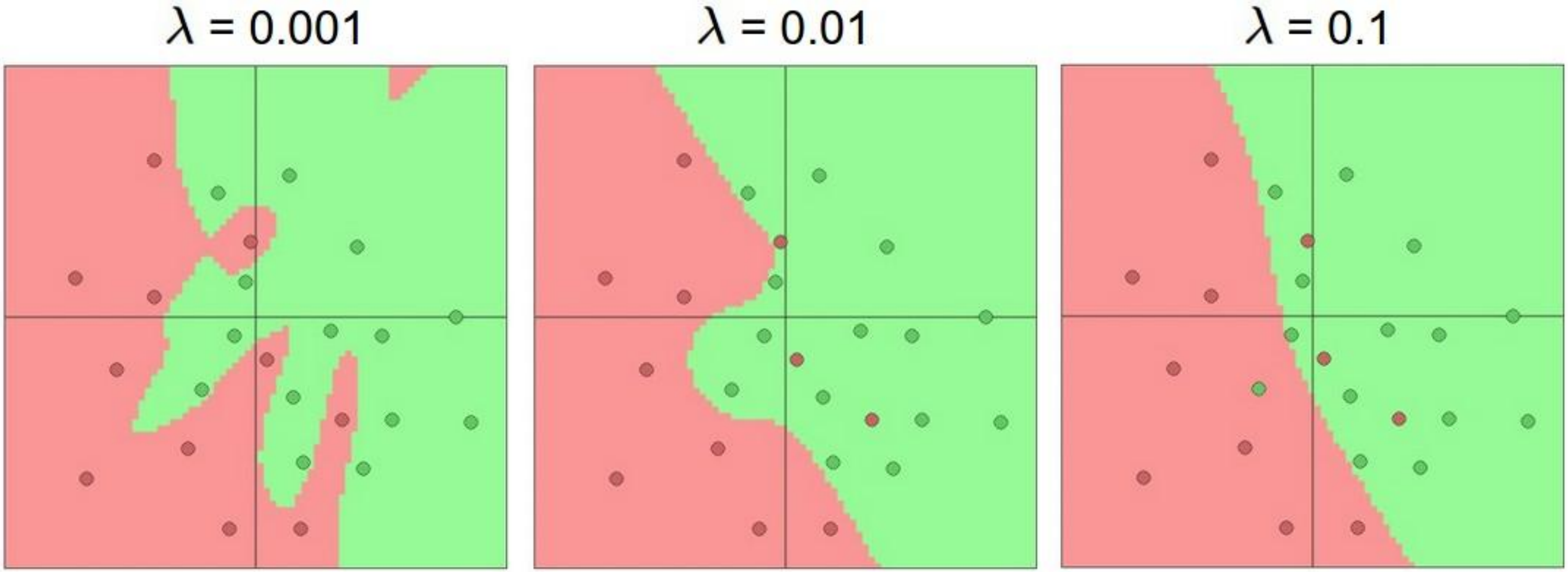
- In practice, if you train a small network the final loss can display a good amount of variance - in some cases you get lucky and converge to a good place but in some cases you get trapped in one of the bad minima.

- If you train a large network you'll start to find many different solutions, but the variance in the final achieved loss will be much smaller.

- In other words, all solutions are about equally as good, and rely less on the luck of random initialization.

How to control overfitting?

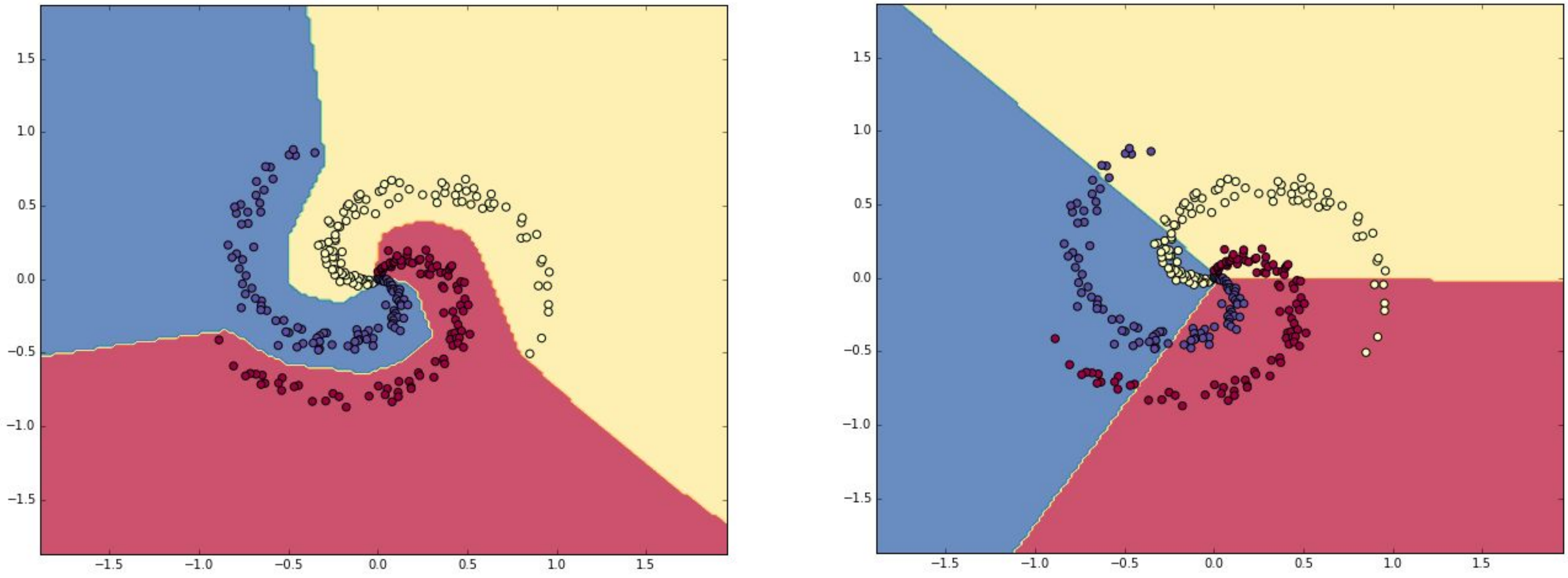
- The regularization strength is the preferred way to control the overfitting of a neural network.



Final word on sizing of NNs

You should not be using smaller networks because you are afraid of overfitting. Instead, you should use as big of a neural network as your computational budget allows, and use other regularization techniques to control overfitting.

Linear Classifier vs NN



How to prepare data for Neural Networks?

Wait until next session! 😉

Thanks!

Any questions?