Announcements

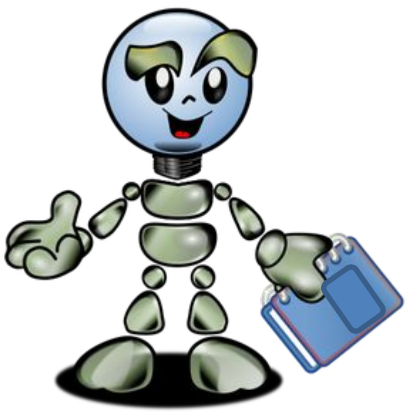
- Pset-3 out, due 03/27
- Quiz-3 will be out tomorrow.

Previously..

- Markov Chain
- Hidden Markov Model
- Decoding HMMs
- Practical scenarios where HMMs are used.

Today

- Intro to neural networks
- Feed-forward networks
- Learning via Backpropagation



Intro to Neural Networks

Motivation

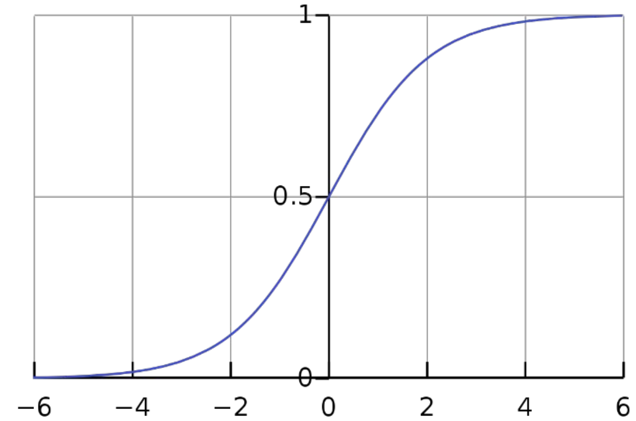
Many slides adapted from Kate Saenko and Brian Kulis

Recall: Logistic Regression

$$g(\theta) = x^T\beta$$

$$P(y = 0|x) = \frac{1}{1 + e^{x^T\beta}}$$

sigmoid/logistic function



$g(\theta)$

Output is probability of label 1 given input

predict "$y = 1$" if $P(y = 0|x) \geq 0.5$

predict "$y = 0$" if $P(y = 0|x) < 0.5$

Image Source: https://en.wikipedia.org/wiki/Sigmoid_function

Do not edit
How to change the design

What are some methods of supporting non-linear decision boundaries?

What are some methods of supporting non-linear decision boundaries?

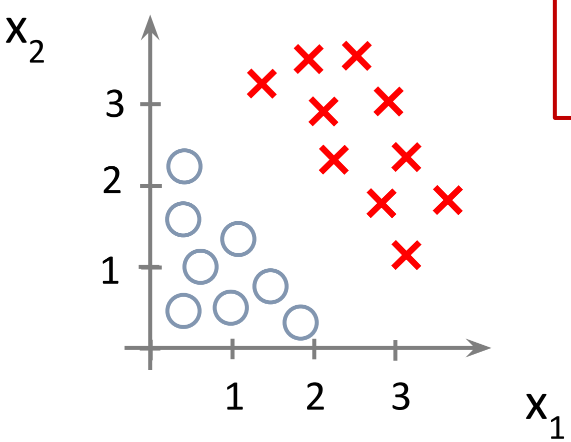
Use non-linear features (eg: log, cosine, polynomial) ✓
80%

Use non-linear kernel functions to project linear features into a non-linear feature space. ✓
84%

Use a single layer perceptron
14%

Use non-linear learning models (SVM, decision trees) ✓
84%

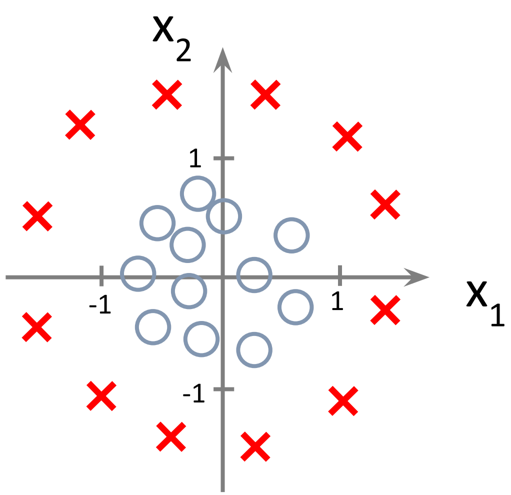
Decision boundary: non-linear features (eg: log, cosine, polynomial)



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundaries



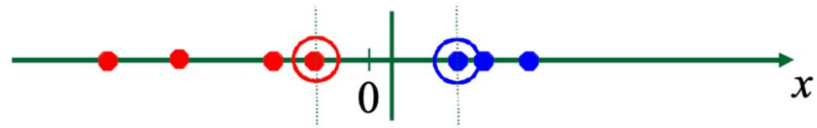
Replace features with nonlinear functions
e.g. log, cosine, or polynomial

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

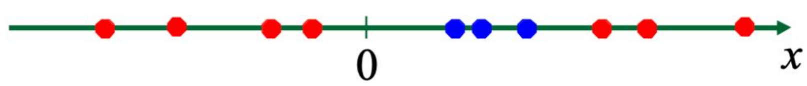
Predict "y = 1" if $-1 + x_1^2 + x_2^2 \geq 0$

Non-linear SVMs

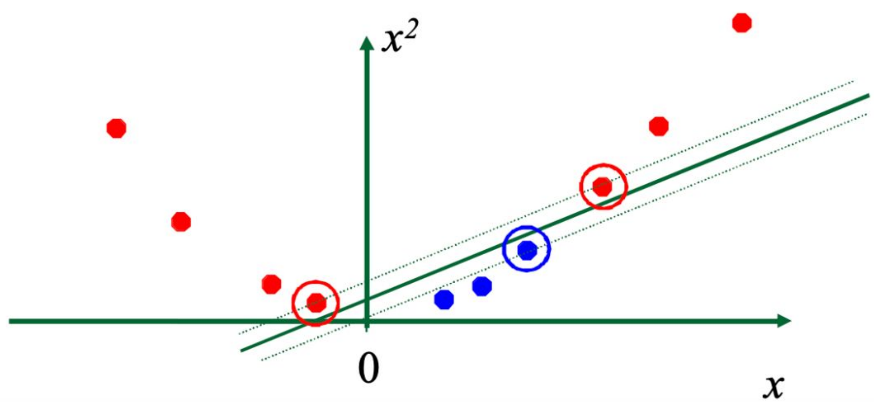
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?



- How about… mapping data to a higher-dimensional space:



Non-linear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(\mathbf{x})$, define a kernel function K such that

Computed between pairs of points.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

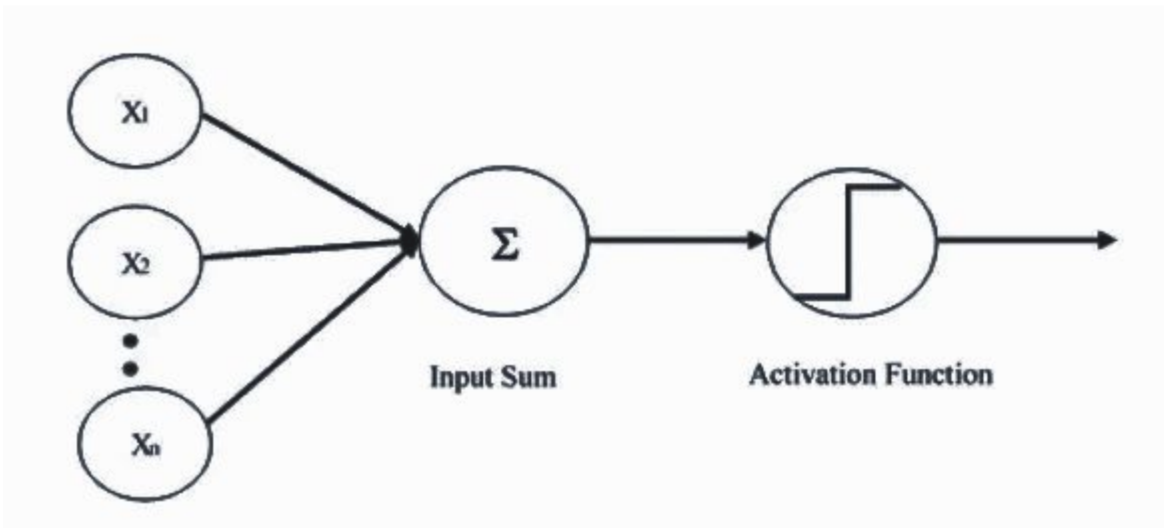
- Gaussian RBF: $K(x_i, x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}\right)$

- Histogram intersection:

$$K(x_i, x_j) = \sum_k \min(x_i(k), x_j(k))$$

Slide inspiration: Kristen Grauman

Single layer perceptron



	Output
$w_1x_1 + w_2x_2 + \ldots + w_nx_n > \theta$ ➡	1
$w_1x_1 + w_2x_2 + \ldots + w_nx_n \leq \theta$ ➡	0

- Does not model non-linear decision boundary.

Limitations of linear models

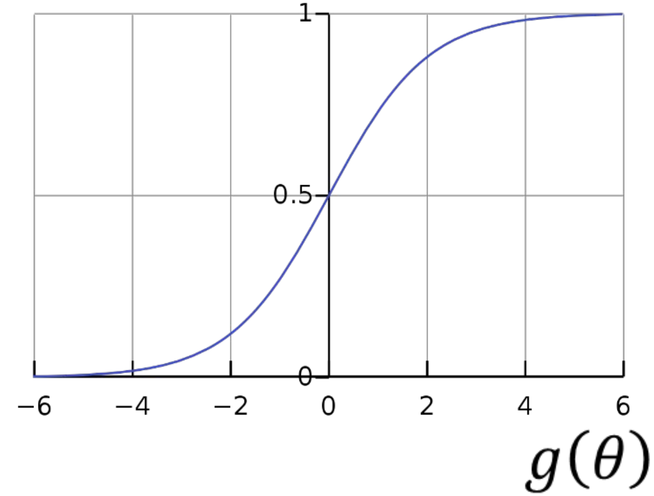
$$g(\theta) = x^T \beta$$

$$P(y = 0|x) = \frac{1}{1 + e^{x^T \beta}}$$

predict “$y = 1$“ if $P(y = 0|x) \geq 0.5$

predict “$y = 0$“ if $P(y = 0|x) < 0.5$

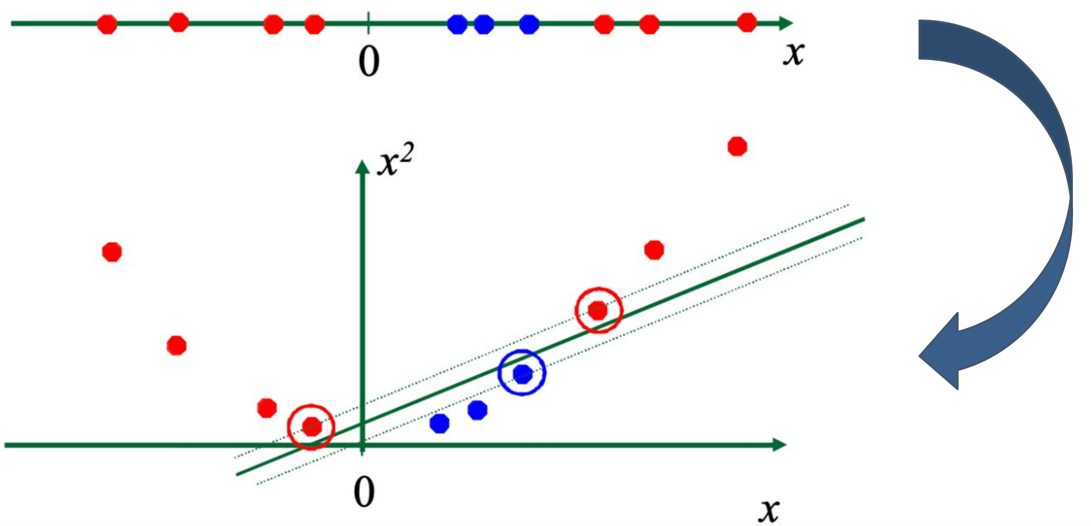
sigmoid/logistic function



- Logistic regression and other linear models cannot handle nonlinear decision boundaries
  - Must use non-linear feature transformations
  - Up to designer to specify which one

Image Source: https://en.wikipedia.org/wiki/Sigmoid_function

Can we instead learn the non-linear transformation?



- Yes, this is what neural networks do!
- A Neural network chains together many layers of “neurons” such as logistic units (logistic regression functions)

Neural Networks

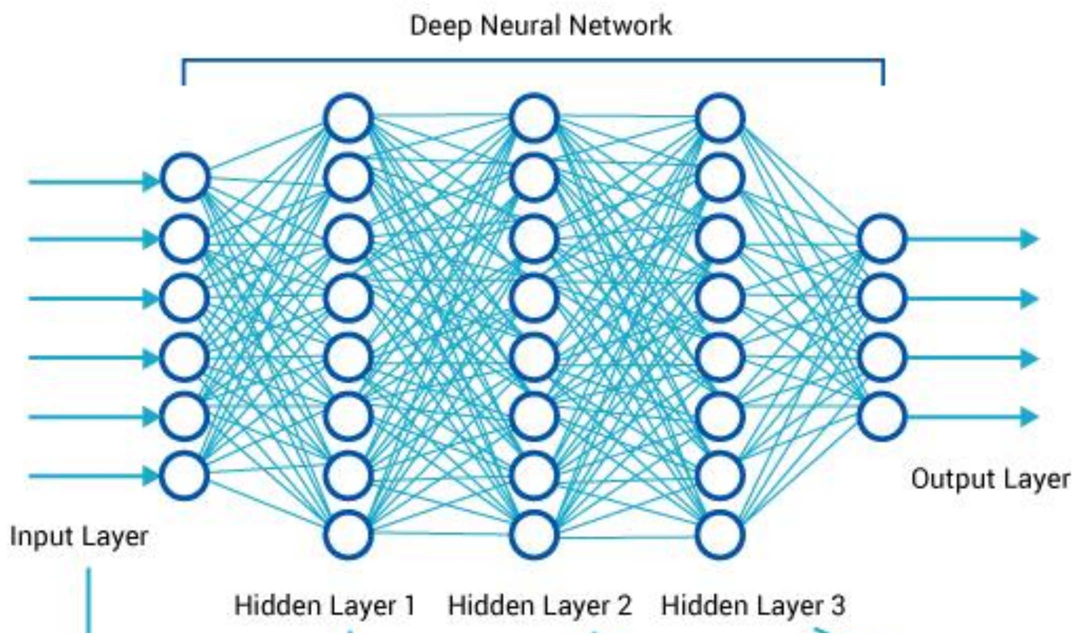


Image: http://www.amax.com/blog/wp-content/uploads/2015/12/blog_deeplearning3.jpg

Neural Networks

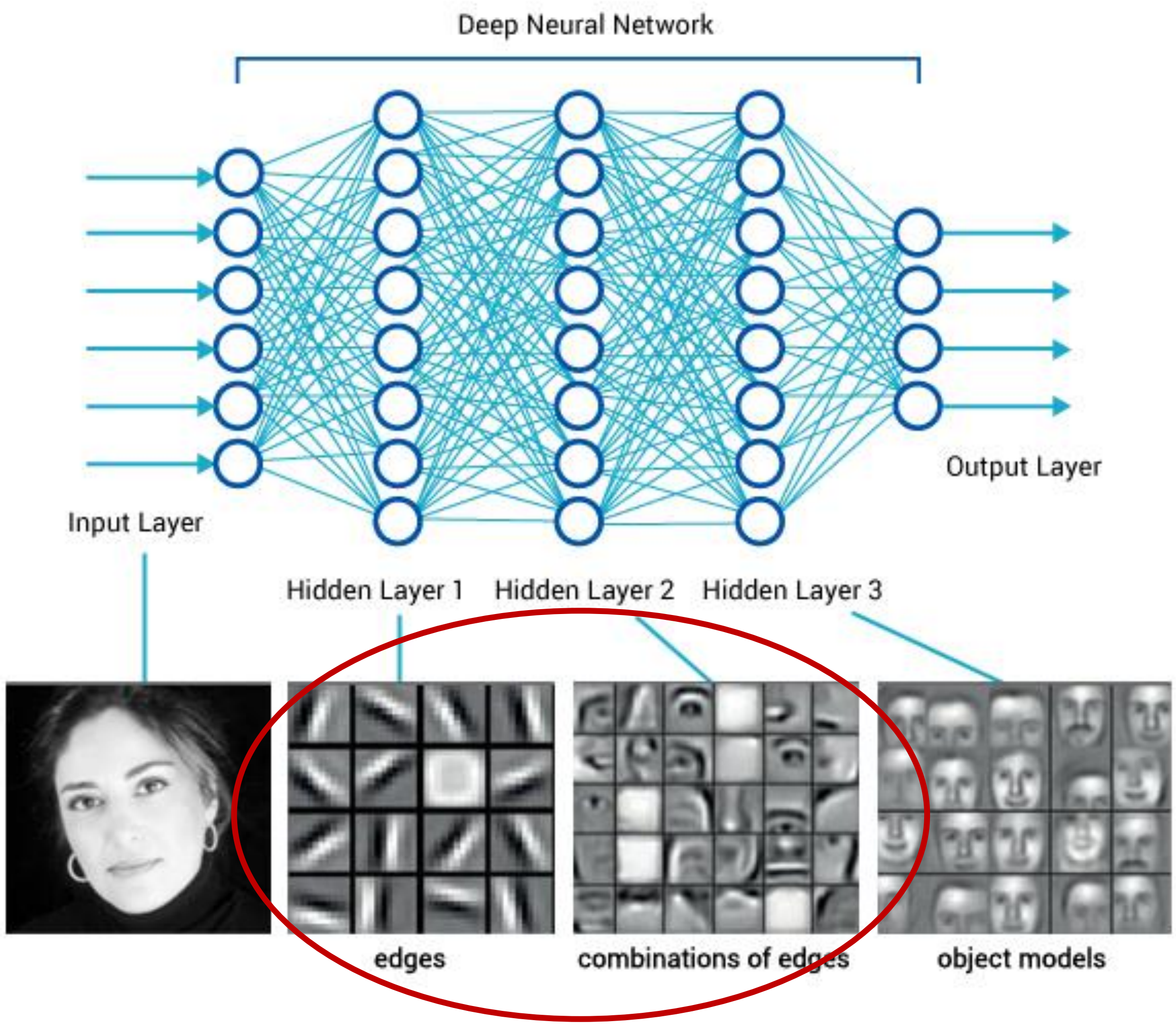
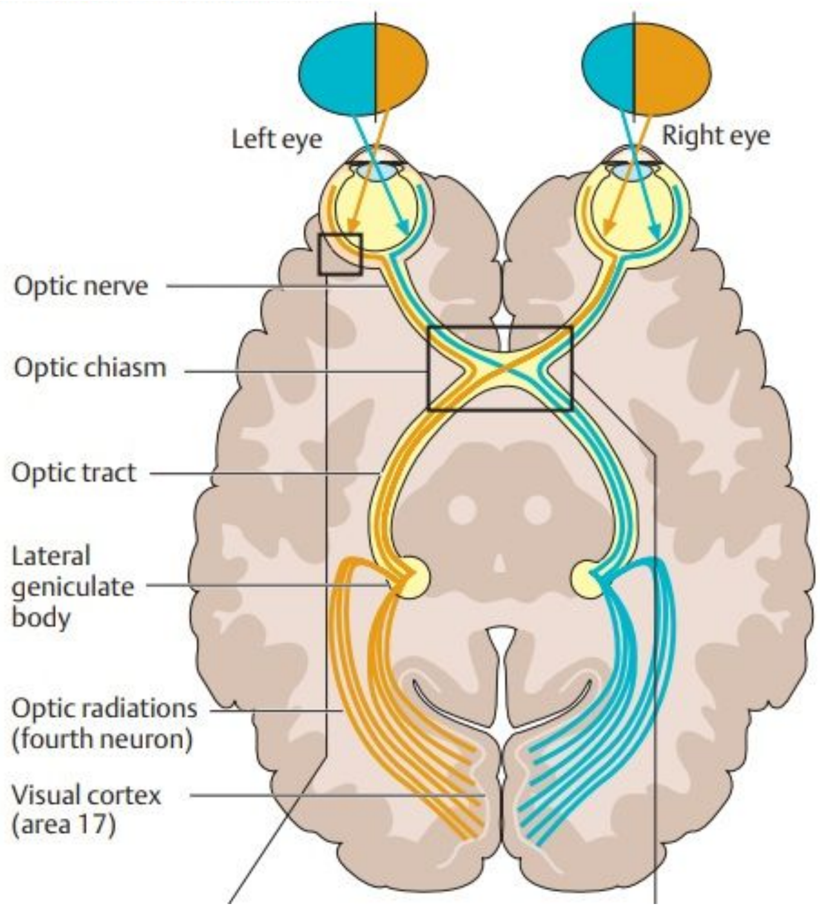


Image: http://www.amax.com/blog/wp-content/uploads/2015/12/blog_deeplearning3.jpg

Taking a step back…

How does our brain perceive the visual world?



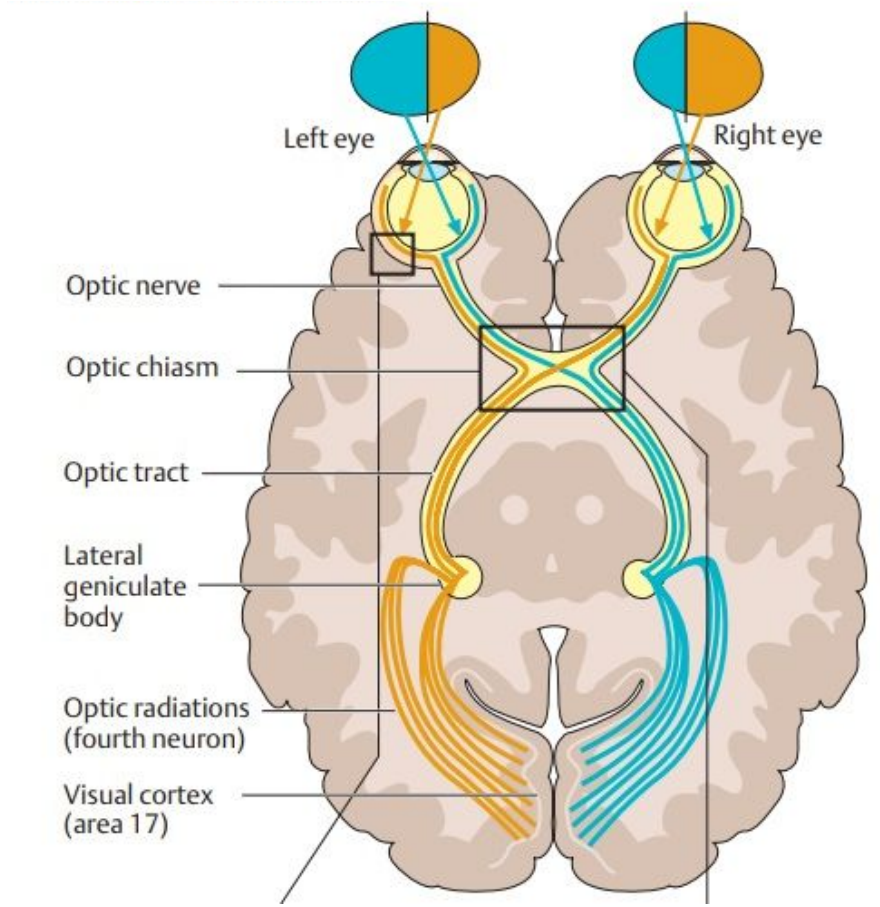
Taking a step back…

How does our brain perceive the visual world?

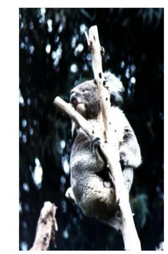
Illumination

Occlusions

Object pose



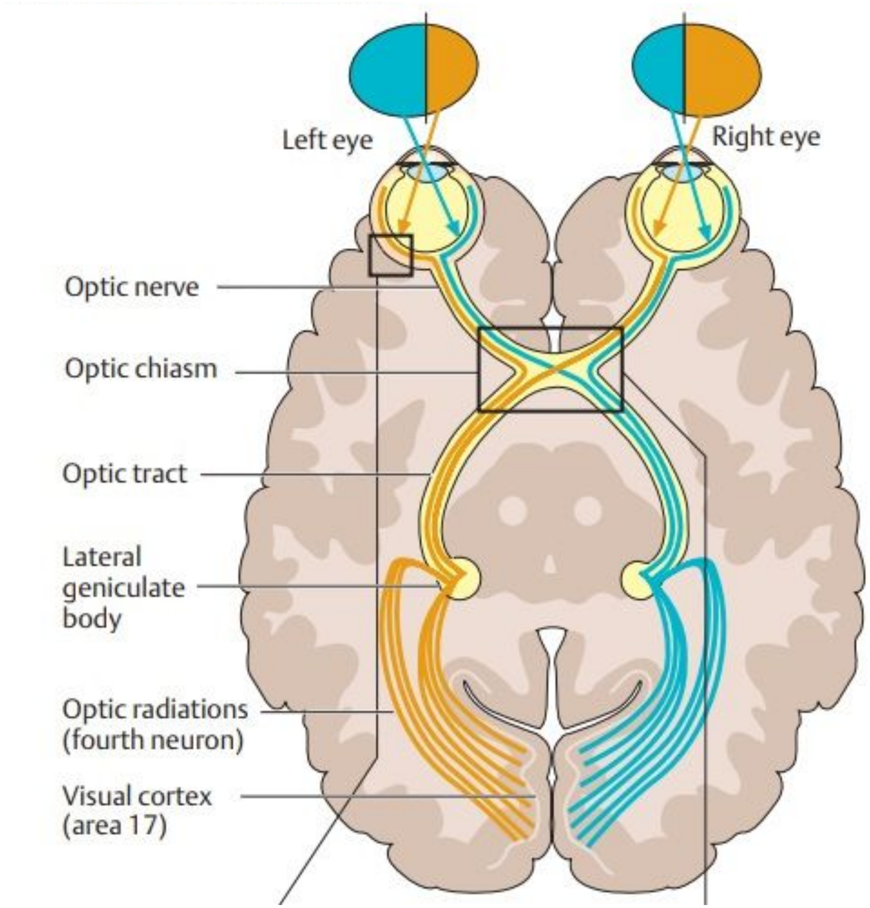
Intra-class appearanc



Viewpoint

Clutter

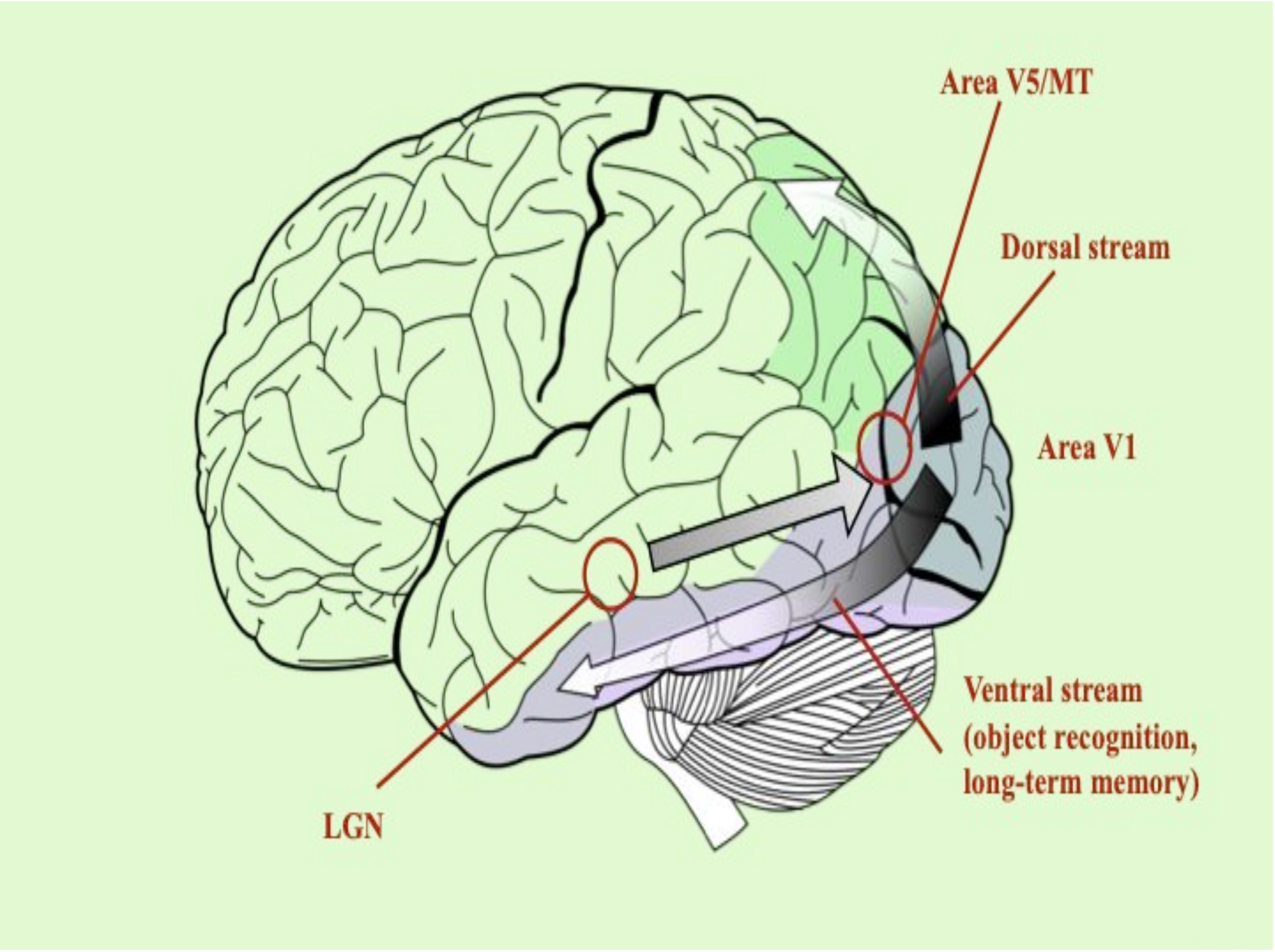
We see with our brains, not with our eyes
– Oliver sacks and others



50% of the brain is dedicated to visual processing!

Slide adapted from: Al Bovik

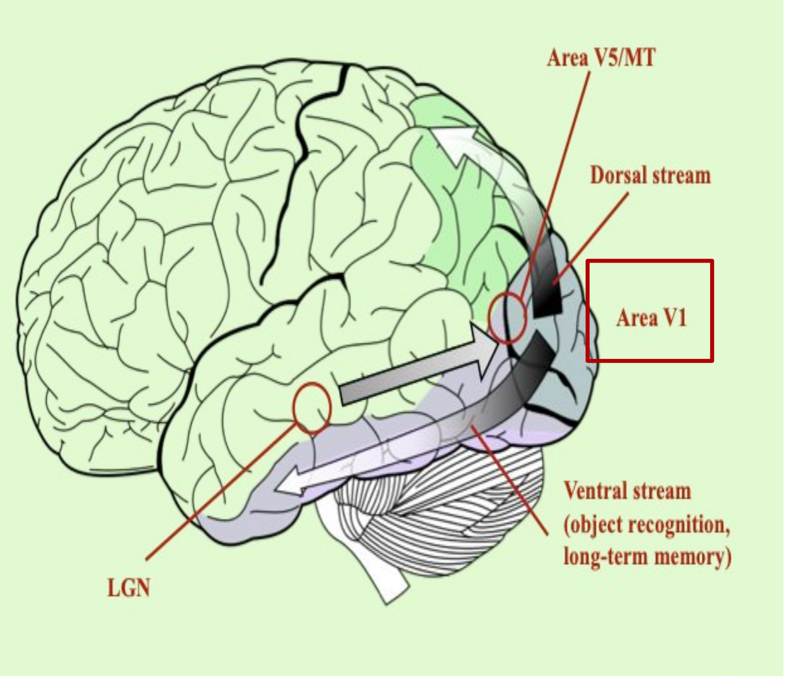
Flow of visual data in the brain



Slide adapted from: Al Bovik

Flow of visual data in the brain

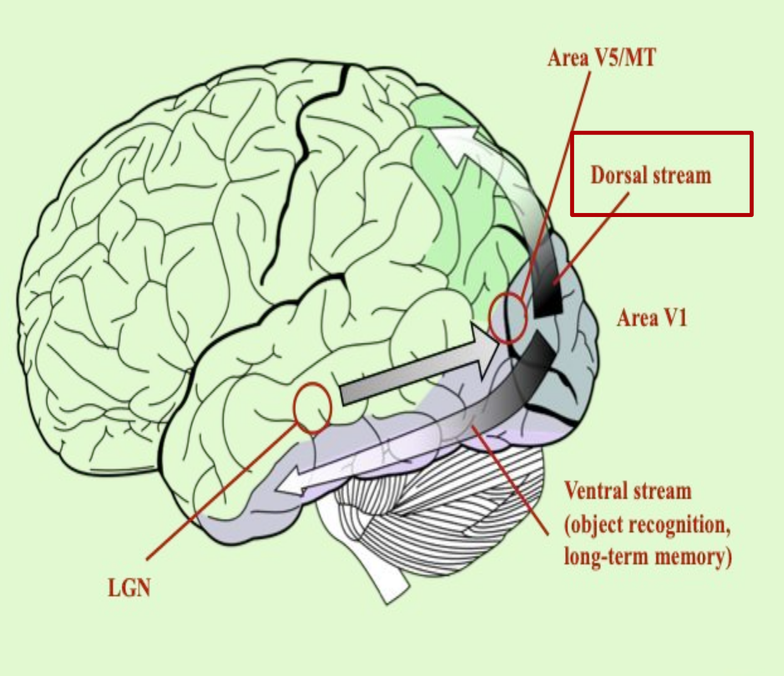
- Area V1 (primary visual cortex): massive multi-scale, multi-orientation decomposition of visual data occurs in space, time, and disparity (depth cue).
  - This rich information is passed en masse to other visual brain centers.



- **Key advantage:** Information redundancy reduction.

Flow of visual data in the brain

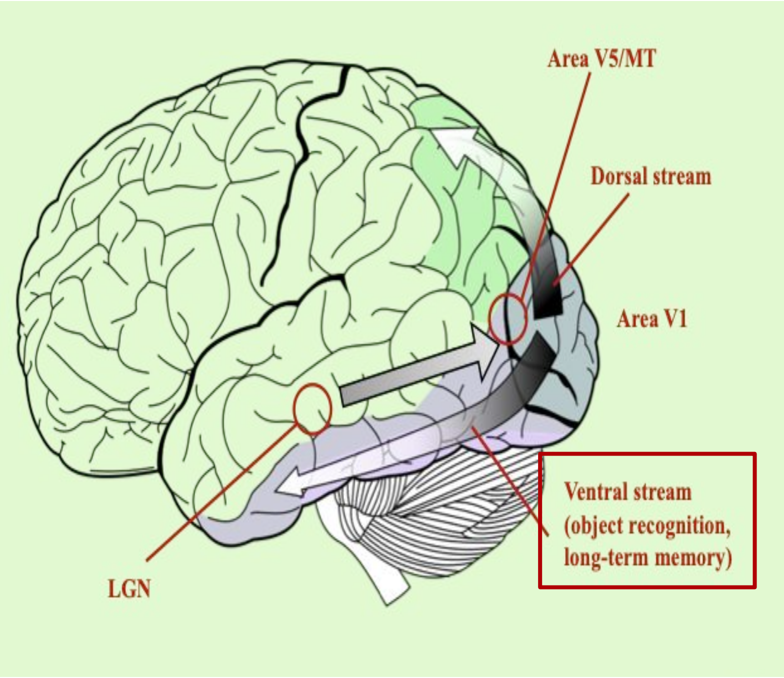
- **Dorsal stream** = “where” pathway of visual information.
- **Tasks:** position, heading, eye vergence and lens control, depth calculation and optical flow.



Slide adapted from: Al Bovik

Flow of visual data in the brain

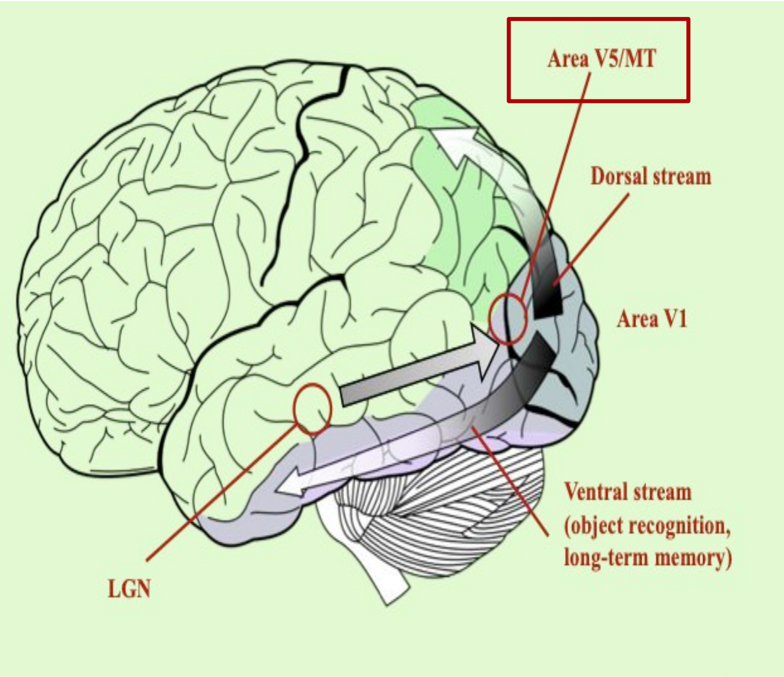
- **Ventral stream** or “what” pathway of visual information.
- **Tasks:** object recognition, long-term memory.



Slide adapted from: Al Bovik

Flow of visual data in the brain

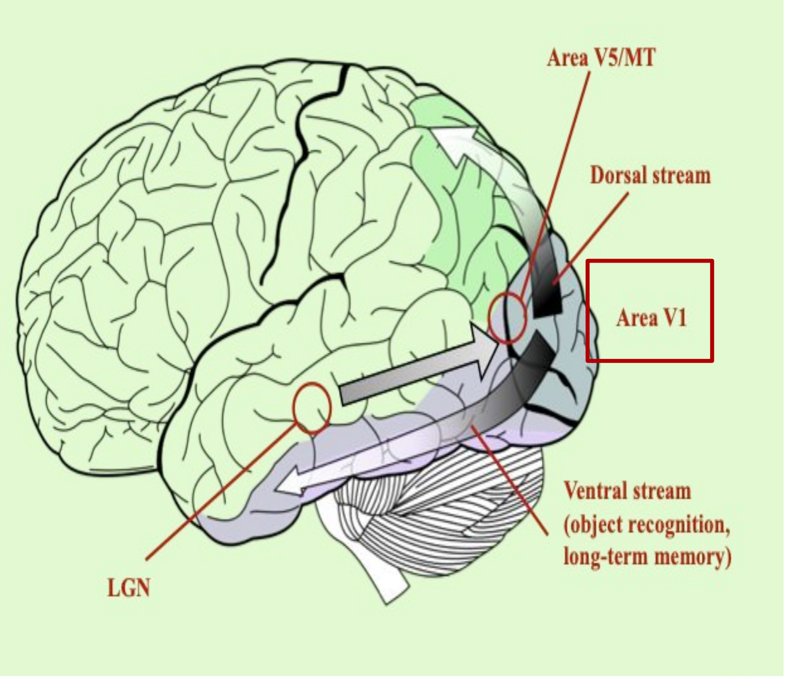
- **Area MT**: Extra-striate cortical area middle temporal (MT) area.
- **Tasks: optical flow (motion) computations**



Slide adapted from: Al Bovik

Types of cortical neurons

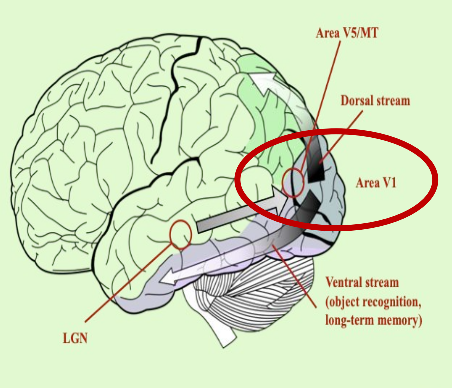
- Area V1 (primary visual cortex): massive multi-scale, multi-orientation decomposition of visual data occurs in space, time, and disparity (depth cue)

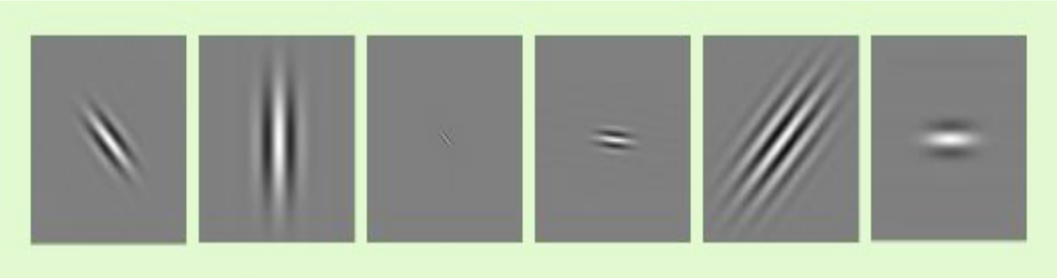


- Three types of cells: **simple cells**, **complex cells**, **hypercomplex cells**

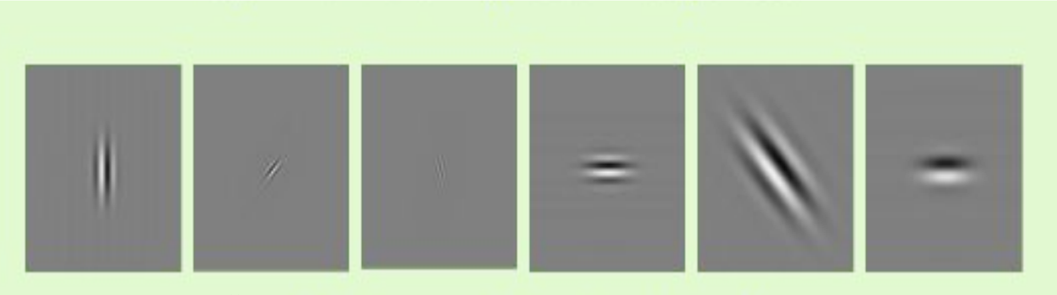
Slide adapted from: Al Bovik

Types of cortical neurons: **Simple cells**





"Bar-sensitive" simple cells



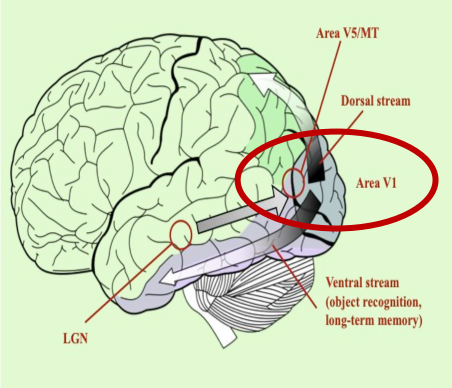
"Edge-sensitive" simple cells

- The response of simple cells correspond to a wide range of orientations, lobe separations, and sizes.

- Laid the foundation for all spatial filters (eg: steerable pyramids, **convolutional filters**)

Slide adapted from: Al Bovik

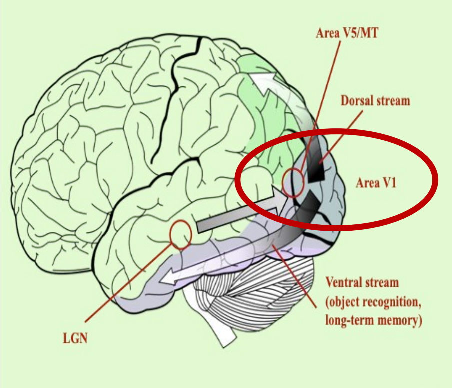
Types of cortical neurons: **Complex cells**



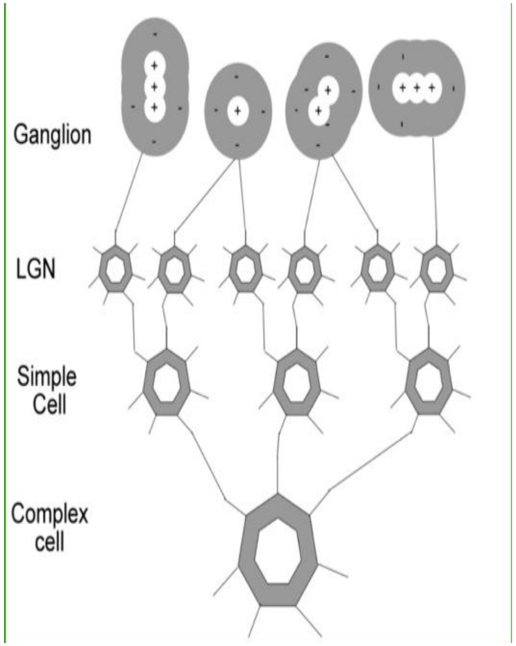
- Less well-understood than simple cells.
- Complex cells receive signals from simple cells and process them in complex, unknown ways.

Slide adapted from: Al Bovik

Types of cortical neurons: **Complex cells**



- Less well-understood than simple cells.
- Complex cells receive signals from simple cells and process them in complex, unknown ways.



- Laid the foundation for introducing **non-linearity** in creative ways in neural networks.

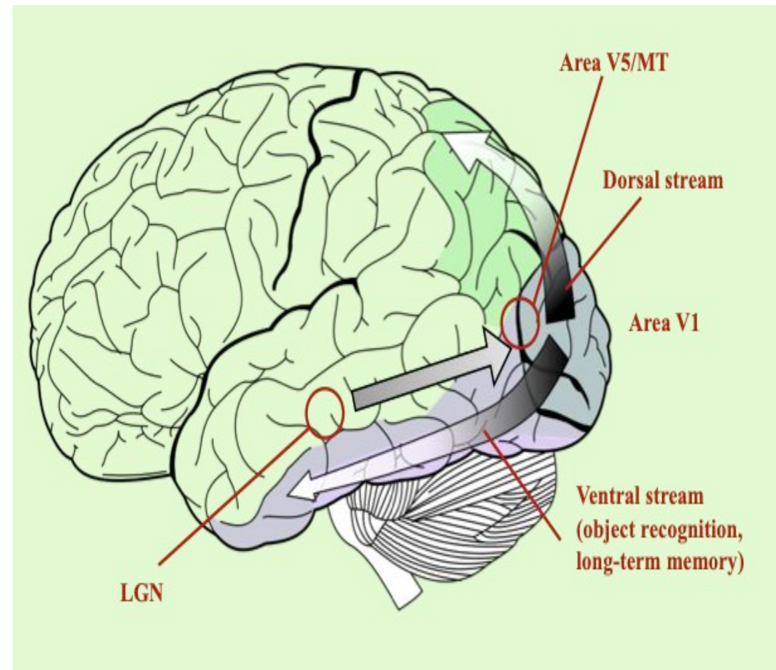
Slide adapted from: Al Bovik

Summary so far

- Area V1 has:
  - Simple cells
  - Complex cells.
  - Hyper-complex cells.

- Dorsal stream: where pathway
- Ventral stream: what pathway

- Area MT : motion perception.



Slide adapted from: Al Bovik

Why study human visual system? (Select all that apply)

Why study human visual system? (Select all that apply)

Marvel at human brains ✓
43%

Develop biologically inspired ML models. ✓
95%

Learn how to process large data effectively ✓
78%

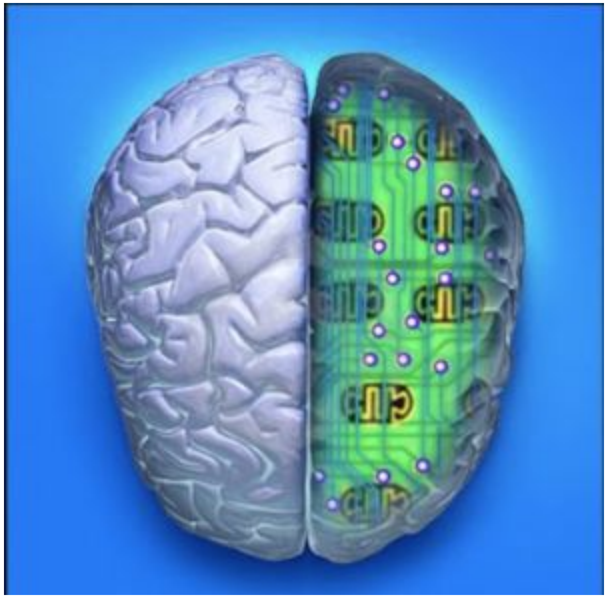
Helps us design effective algorithms to avoid overfitting.
59%

Can we learn features directly?

Can we replicate human visual system?

3,000 - 30,000 human-recognizable object categories.

30+ degrees of freedom in the pose of objects.



Thousands to millions of pixels in an image

Millions to billions of pixels in a video

From neural responses to spatial filters to features

- **Feature:** A vector representing measurable characteristics of an image (video).
  - ✅ Remove redundant information
  - ✅ Extract useful information

Learning features directly

Can we replicate human visual system?

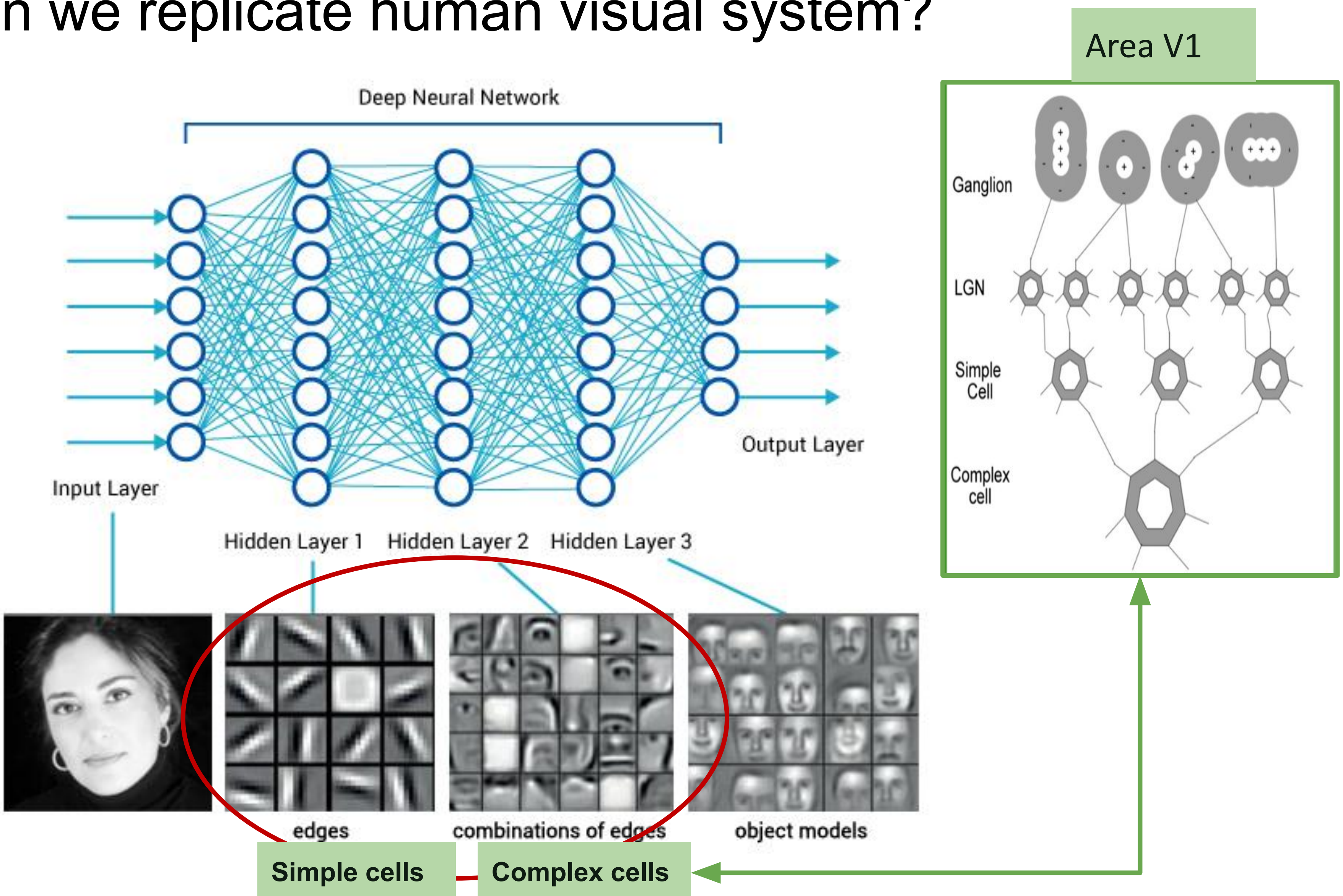
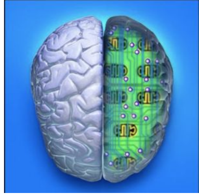


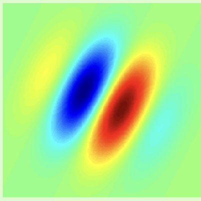
Image: http://www.amax.com/blog/wp-content/uploads/2015/12/blog_deeplearning3.jpg

From neural responses to spatial filters

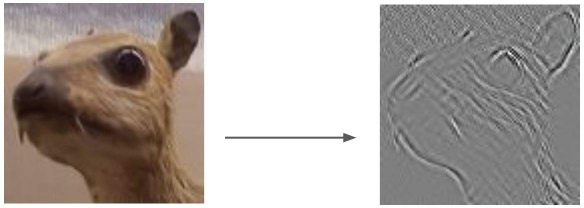


- Kernel = a small matrix mimicking primary and secondary cell receptive fields.
  - Eg: Laplacian of Gaussians (LoG), Derivative of Gaussians (DoG)

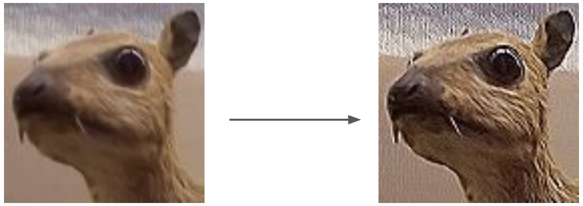
0	-1	0
-1	5	-1
0	-1	0



Edge detection



Sharpening



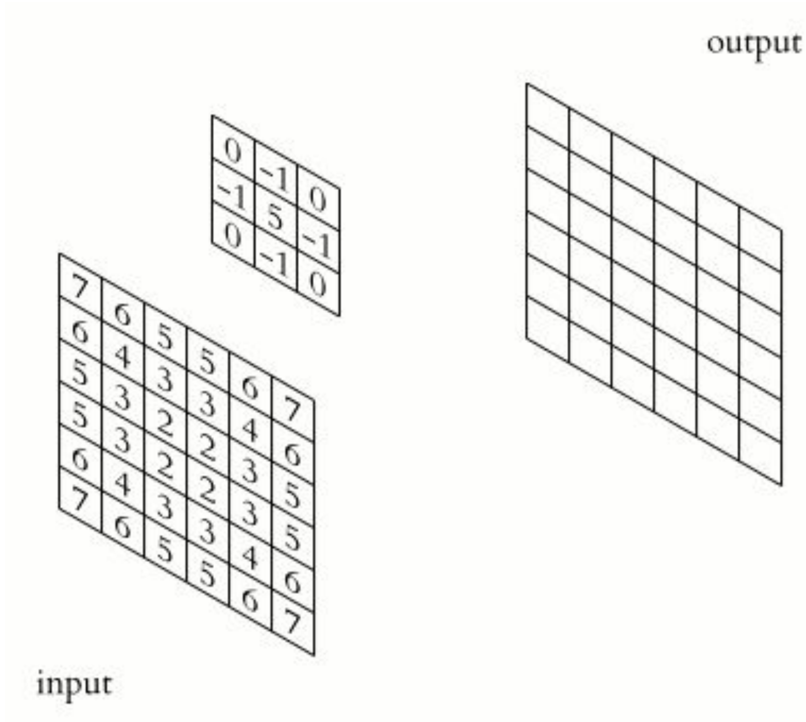
Interesting reads:

[1] J. Daugman, Uncertainty relation for resolution in space, spatial frequency and orientation optimized by visual cortical filters, *Journal of the Optical Society of America A*, July 1985

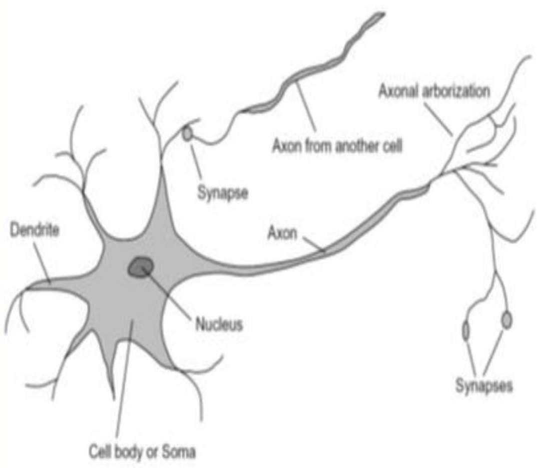
[2] https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

[3] https://www.cns.nyu.edu/~eero/steerpyr/

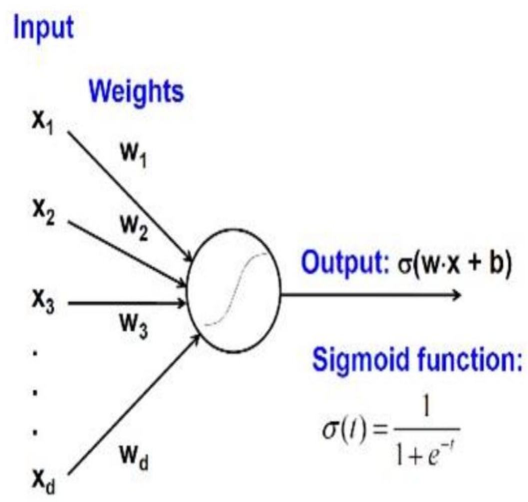
From neural responses to spatial filters



Biological inspiration: Neuron cells



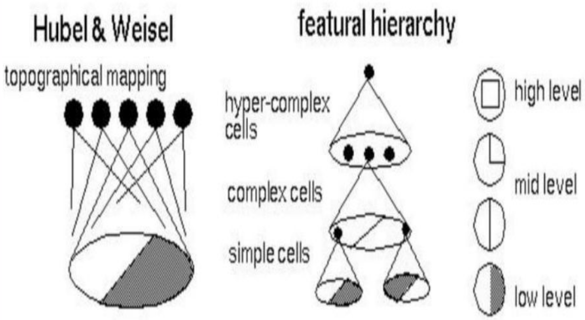
A biological neuron



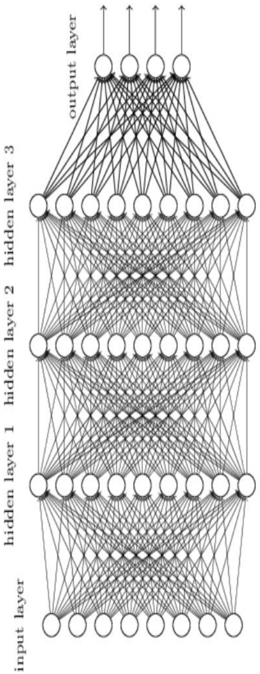
An artificial neuron

- Accept information from multiple inputs.
- Transmit information to other neurons.
- Multiply inputs by weights along edges
- Apply some function to the set of inputs

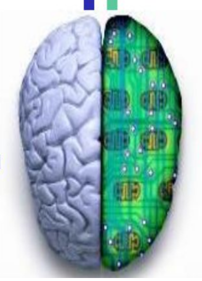
Biological inspiration: Hierarchical neural network



Hubel and Weisel's architecture



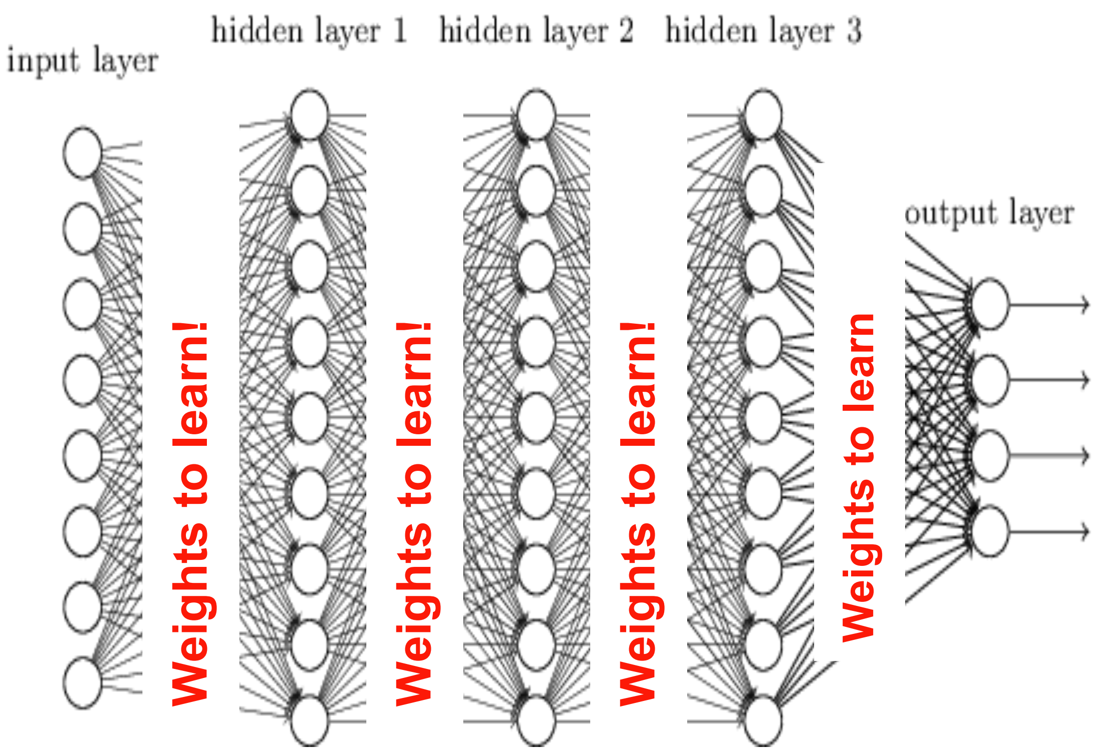
Multi-layer neural network



Slide adapted from: Jia-bin Huang

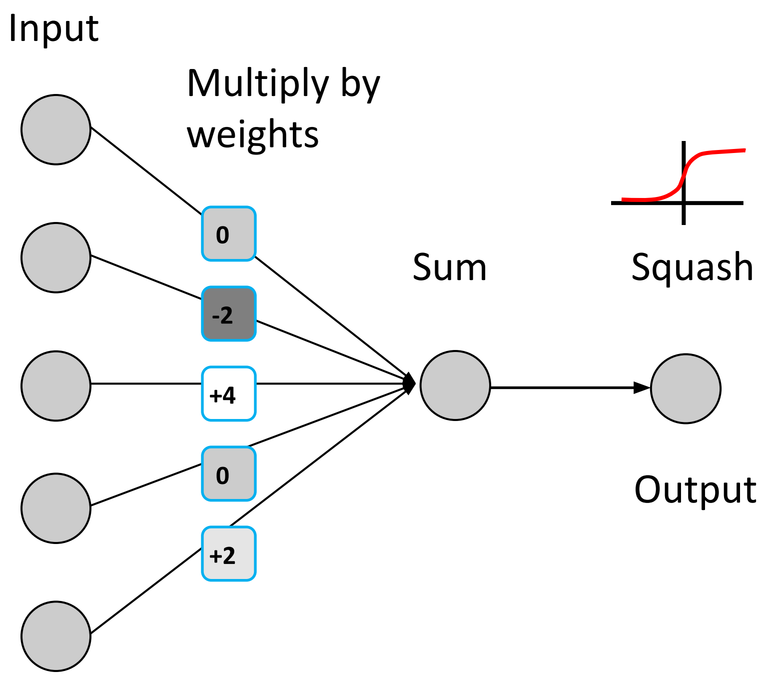
Deep neural networks

- Lots of hidden layers
- Lots of non-linearity
- Depth = power (usually)

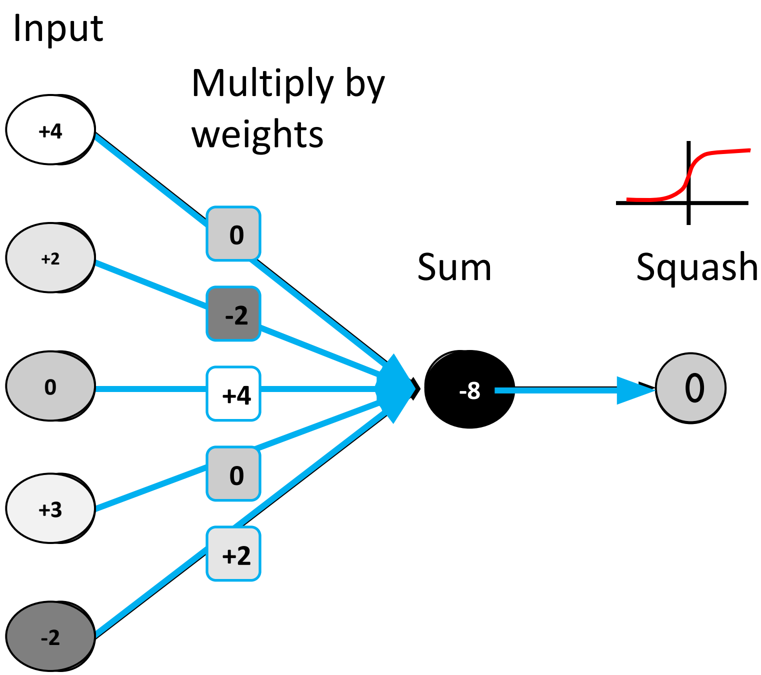


Slide adapted from: Adriana Kovashka

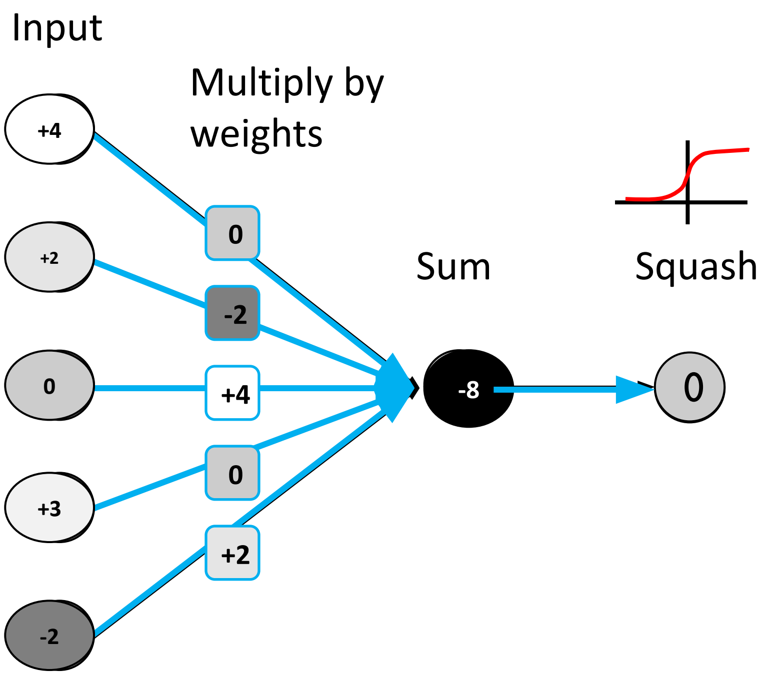
Logistic Unit as Artificial Neuron



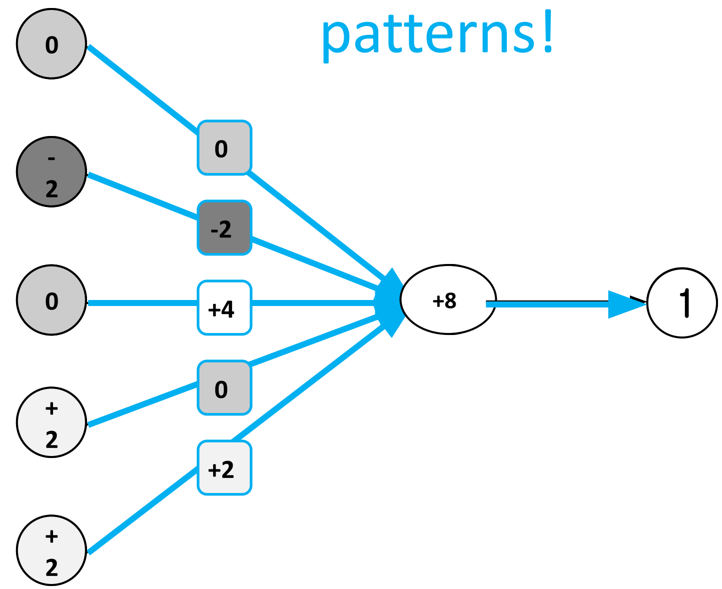
Logistic Unit as Artificial Neuron



Logistic Unit as Artificial Neuron

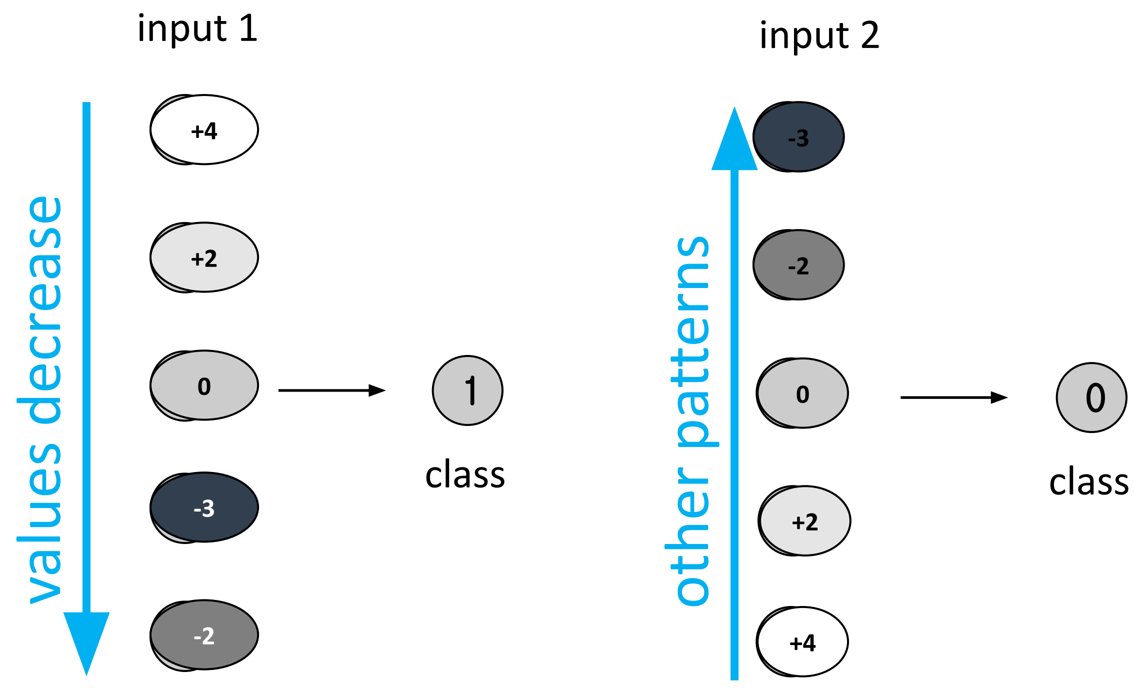


Neurons learn patterns!

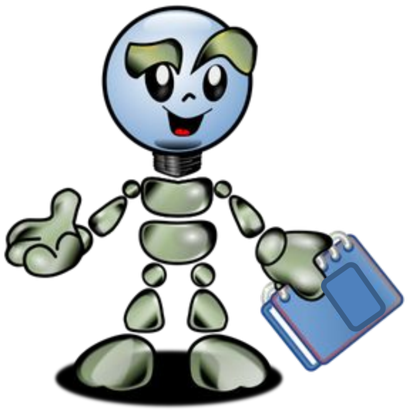


Artificial Neuron Learns Patterns

- Classify input into class 0 or 1
- Teach neuron to predict correct class label
- Detect presence of a simple "feature"



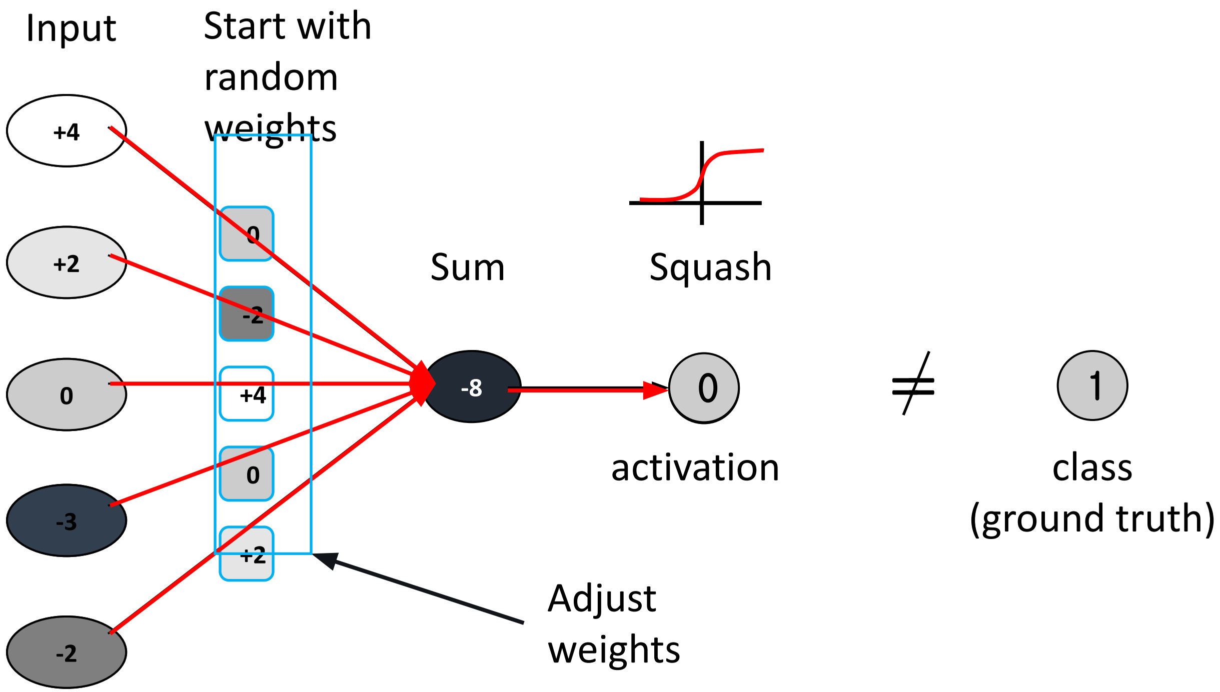
Example



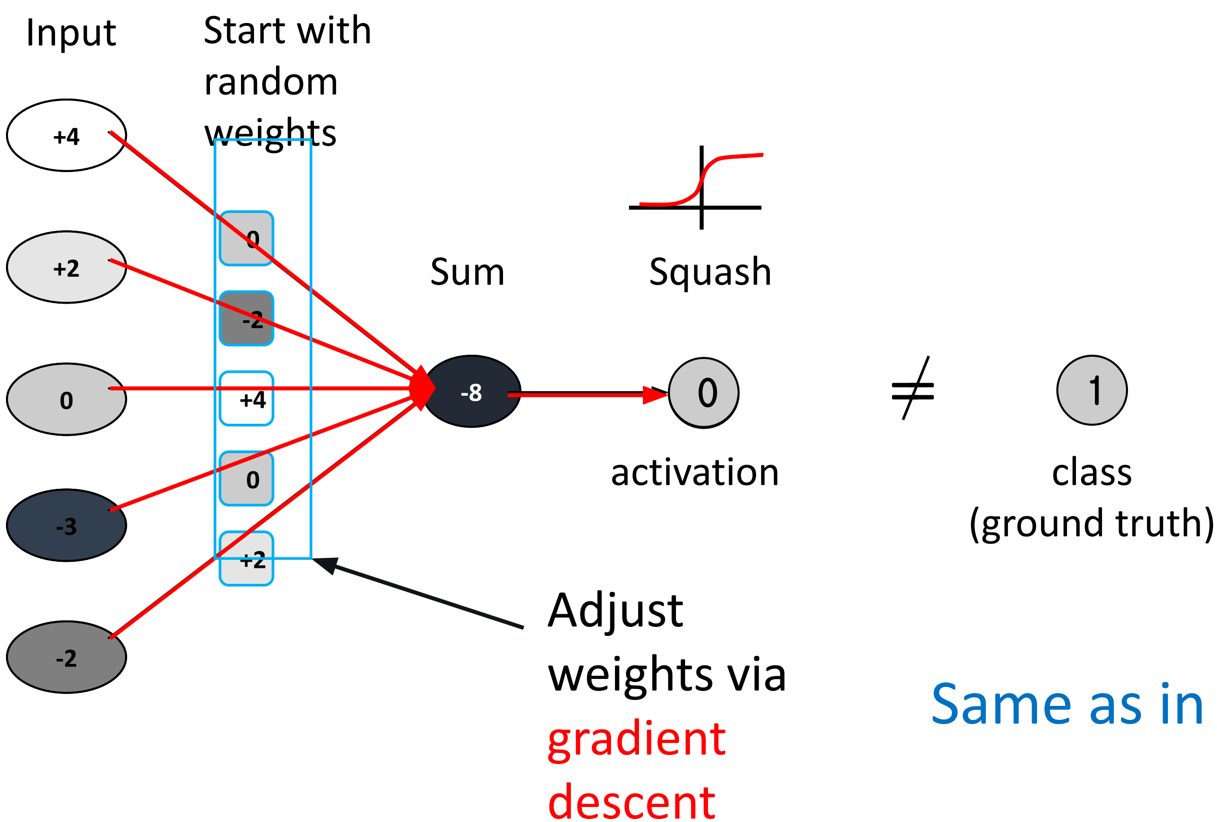
Neural Networks: Learning

Intuition

Artificial Neuron: Learning

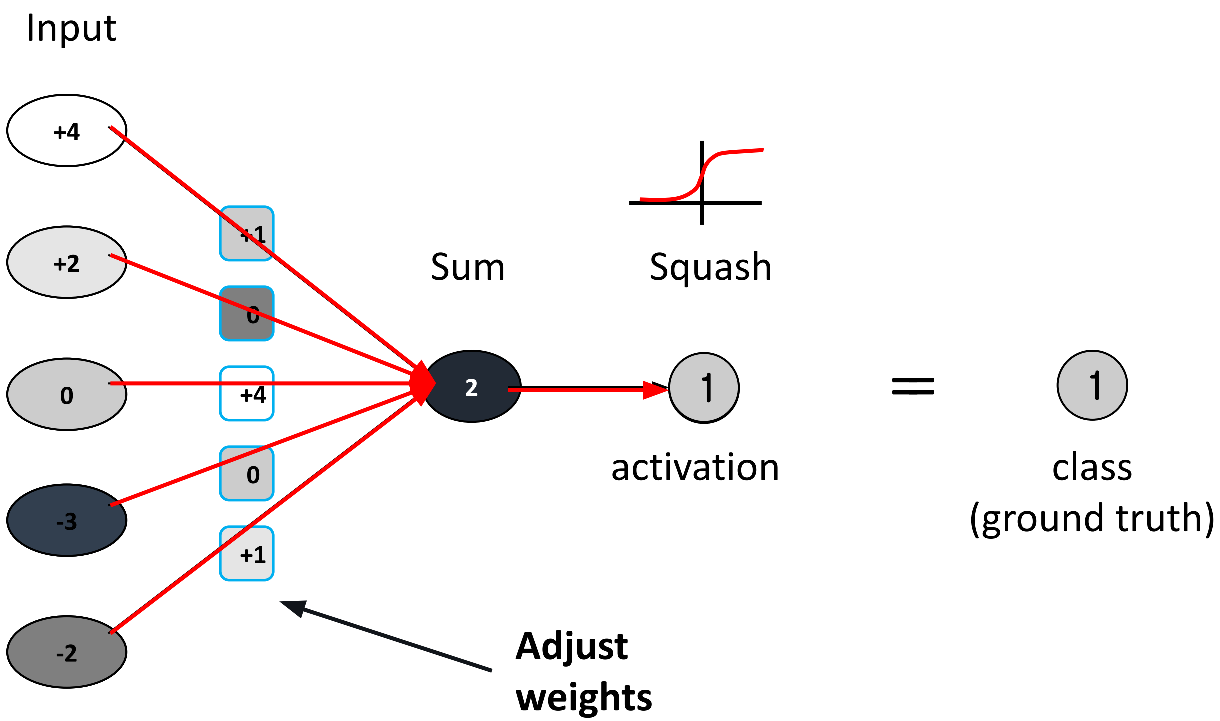


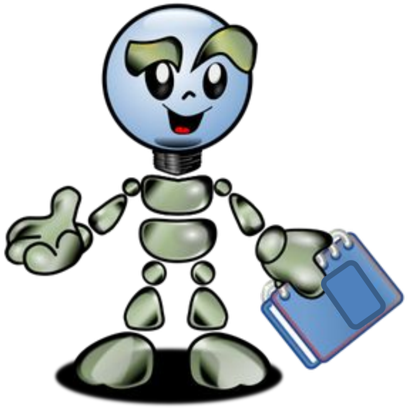
Artificial Neuron: Learning



Same as in logistic regression

Artificial Neuron: Learning

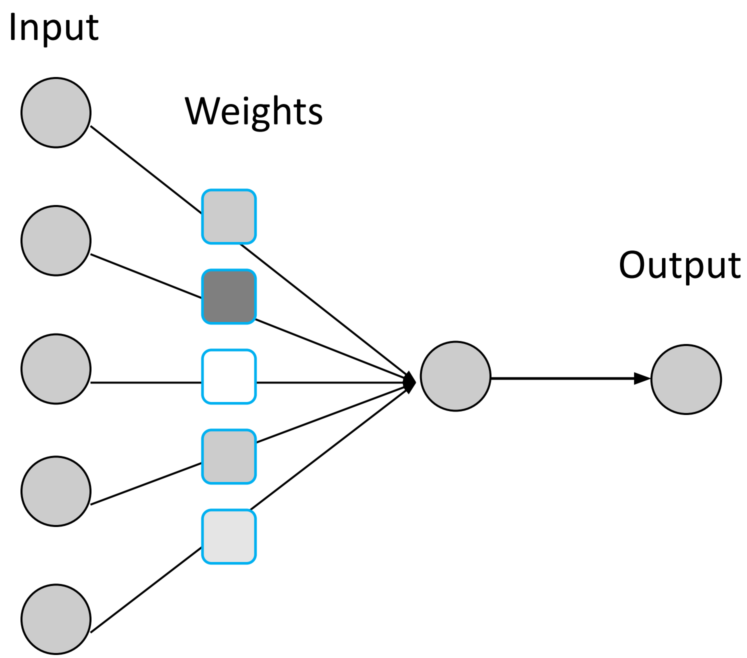




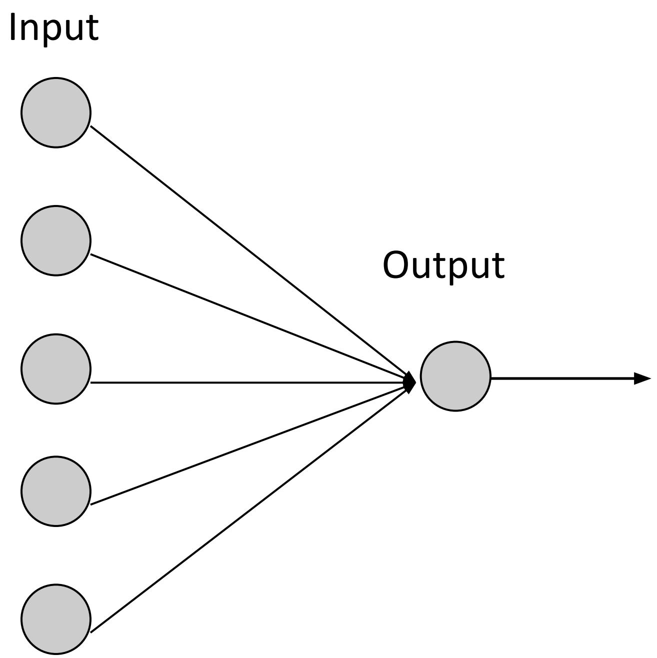
Neural Networks: Learning

Multi-layer network

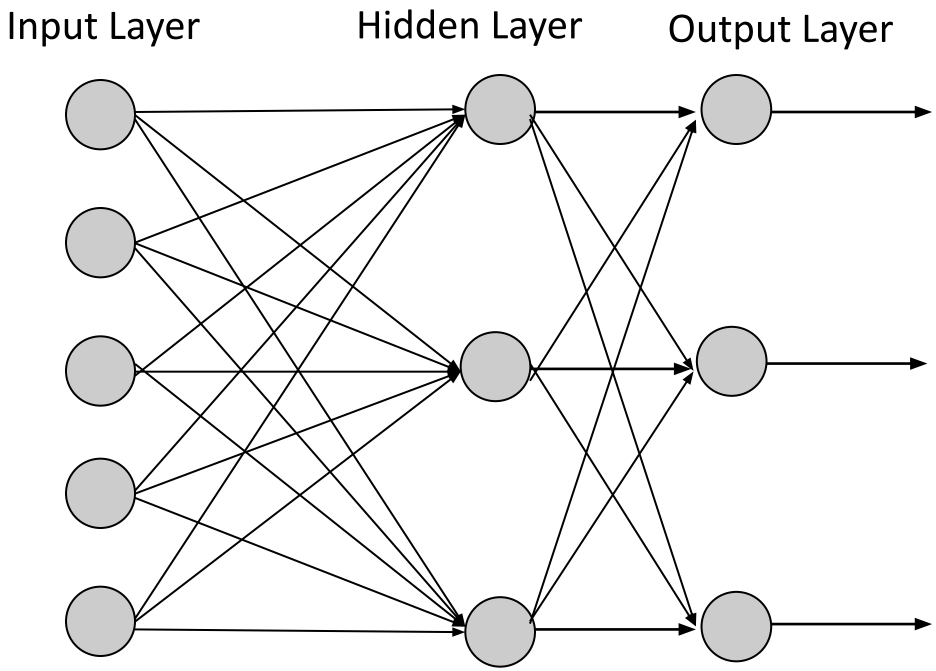
Artificial Neuron: simplify



Artificial Neural Network



Single Neuron

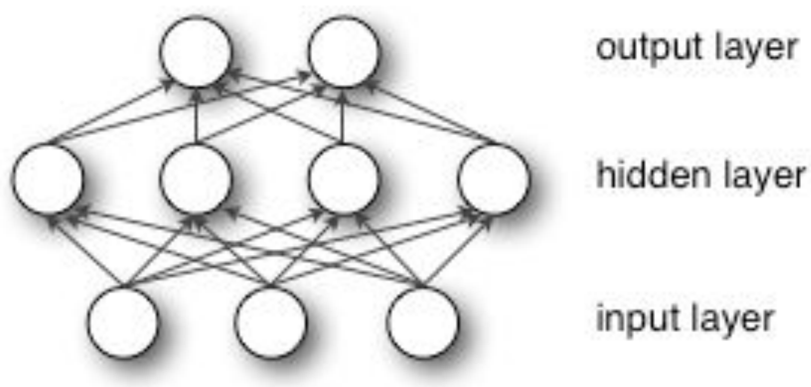


Neural Network

Deep Network: many hidden layers

Multi-layer perceptron (MLP)

- Just another name for a feed-forward neural network



How is a multilayer perceptron different from a single layer perceptron? Select all that apply

How is a multilayer perceptron different from a single layer perceptron? Select all that apply

Single layer perceptron cannot take non-linear features as input
19%

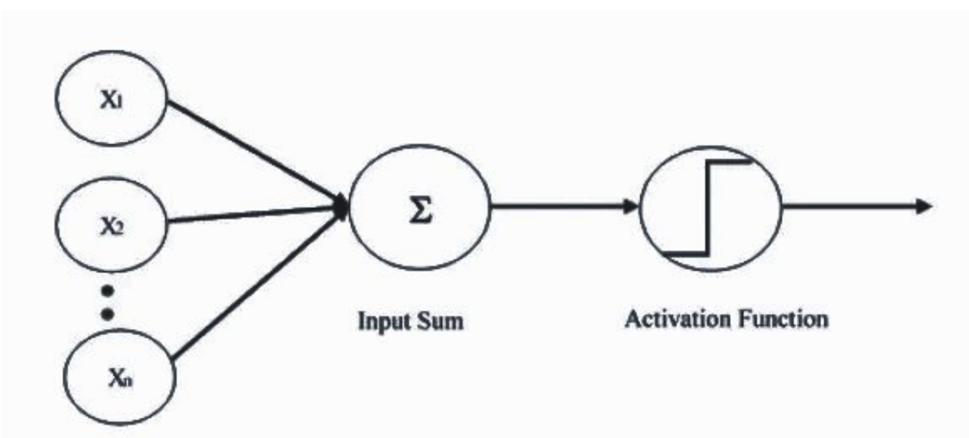
Single layer perceptron has only a single hidden layer
31%

Single layer perceptron has no hidden layers ✓
48%

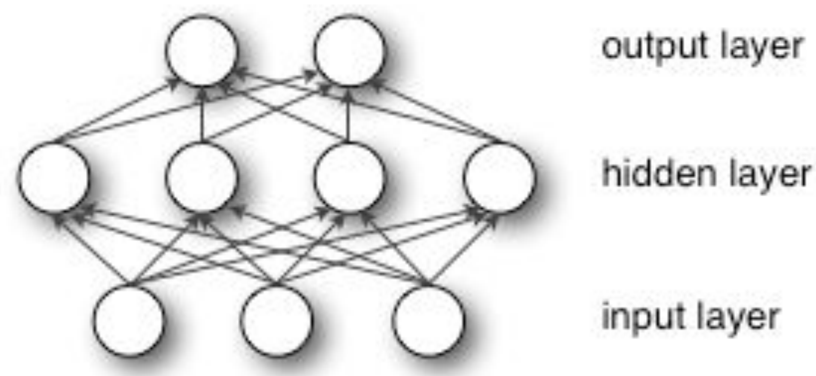
Multi layer perceptron cannot process non-linear features as input
2%

Multi-layer perceptron (MLP)

- Just another name for a feed-forward neural network
- Logistic regression is a special case of the MLP with no hidden layer

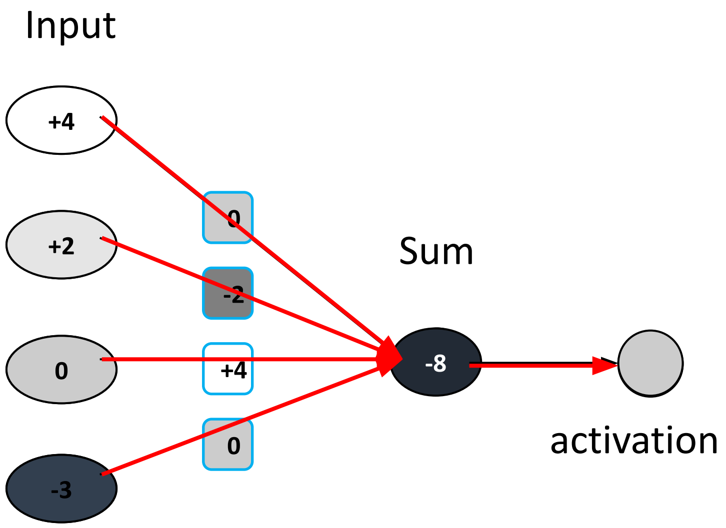


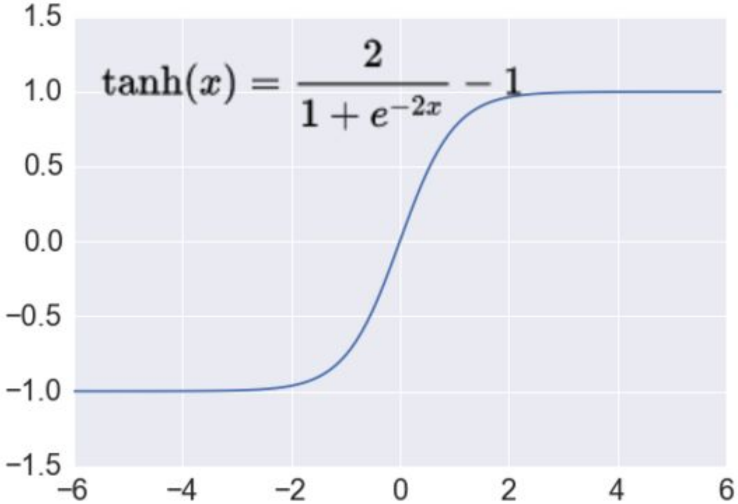
Logistic regression or single layer perceptron



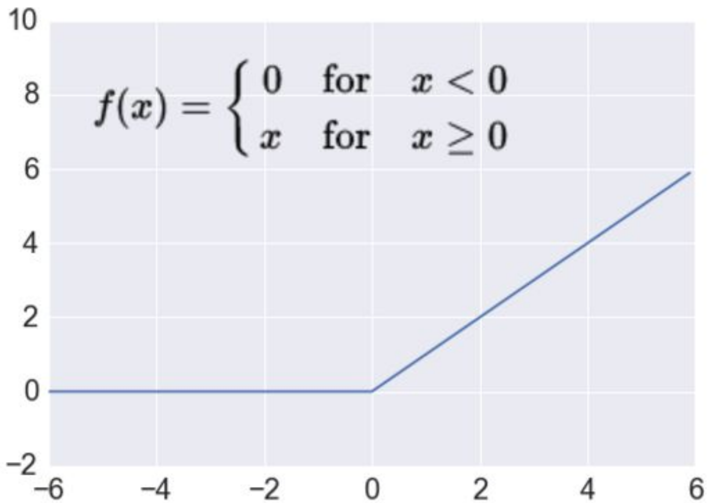
Other Non-linearities

- Also called activation functions





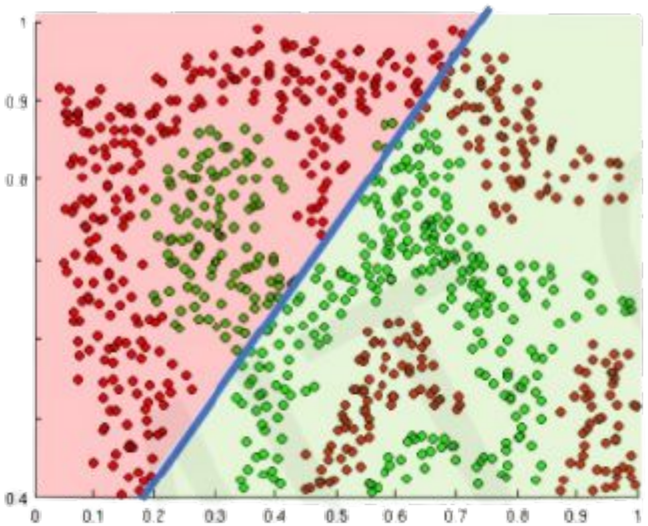
tanh



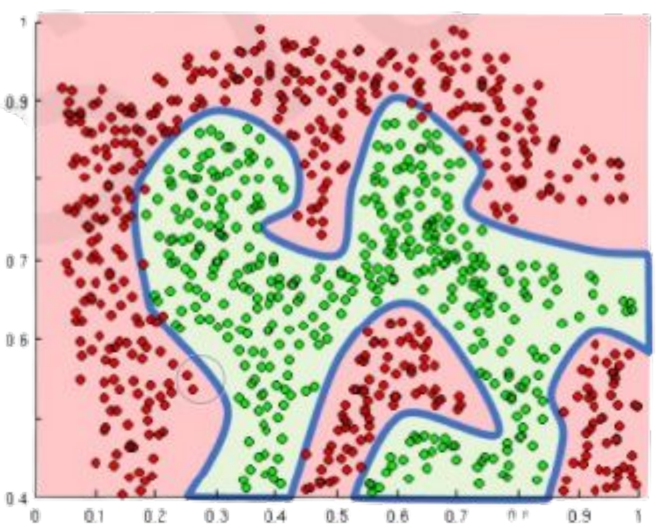
ReLU

Importance of Non-linearities

- The purpose of activation functions is to introduce non-linearities into the network.



Linear activation functions produce linear decisions no matter the network size.



Non-linearities allow us to approximate arbitrarily complex functions.

Artificial Neural Network:

general notation

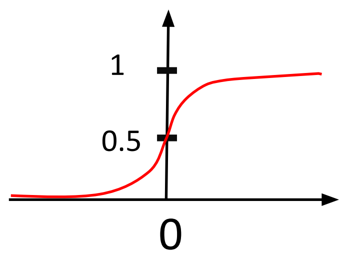
input

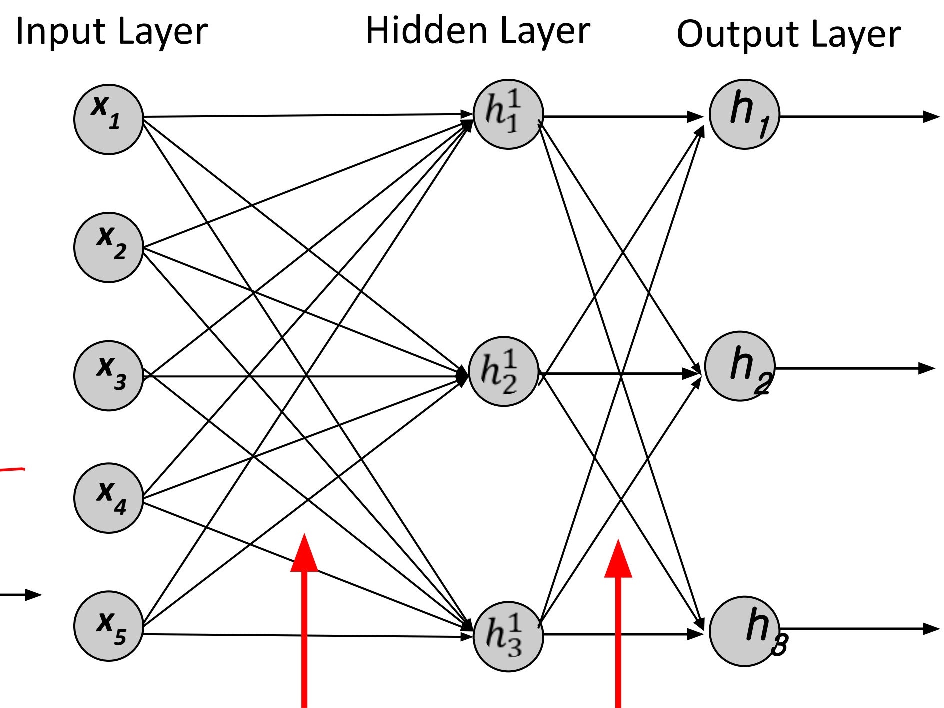
$$x = \begin{bmatrix} x_1 \\ \cdots \\ x_5 \end{bmatrix}$$

hidden layer activations

$$h^i = g(\Theta^{(i)} x)$$

$$g(z) = \frac{1}{1 + \exp(-z)}$$



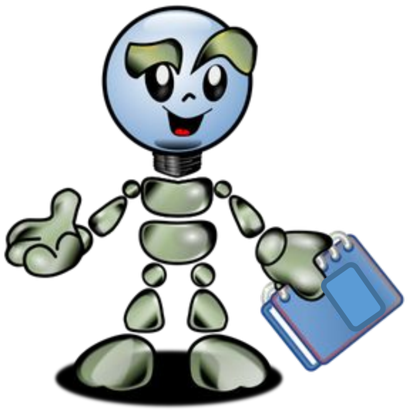


output

$$h_\Theta(\mathrm{x}) = g(\Theta^{(2)} a)$$

weights

$$\Theta^{(1)} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{15} \\ \vdots & \ddots & \vdots \\ \theta_{31} & \cdots & \theta_{35} \end{pmatrix} \qquad \Theta^{(2)} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{13} \\ \vdots & \ddots & \vdots \\ \theta_{31} & \cdots & \theta_{33} \end{pmatrix}$$



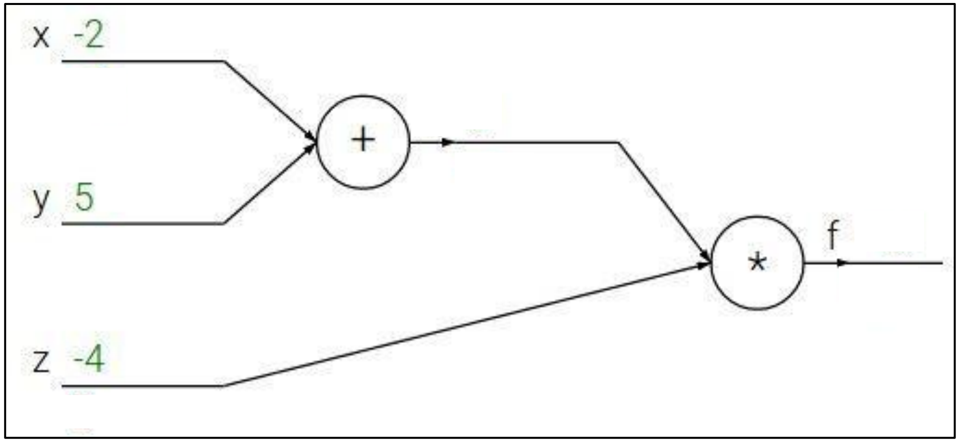
Neural Networks: Learning

Backpropagation

Chain Rule with a Computational Graph

$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

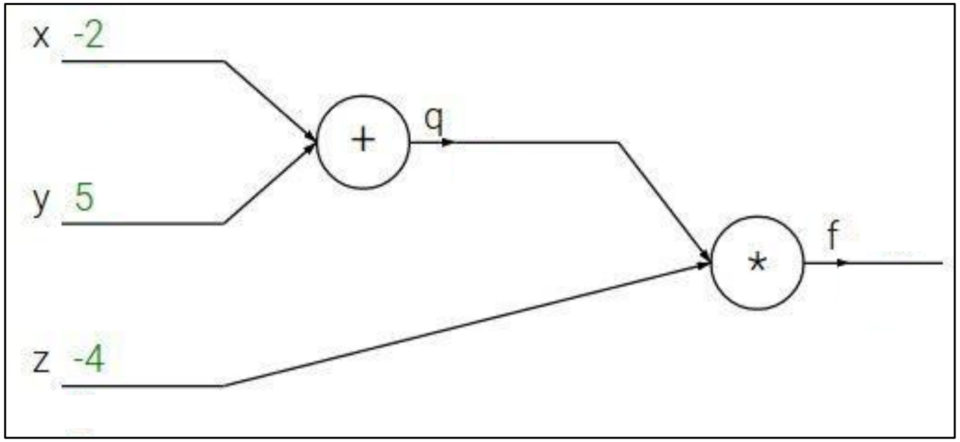


Example from http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture04.pdf

Chain Rule with a Computational Graph

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Computation Graph: Forward

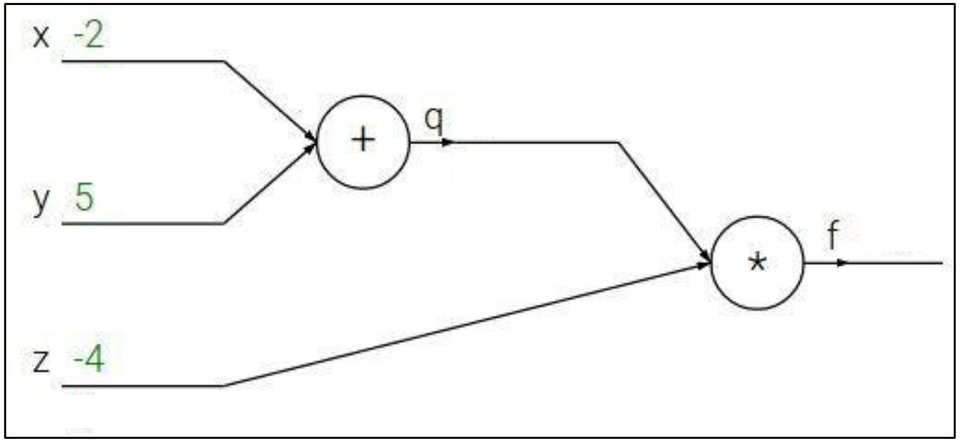
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



compute values

Compute q and f, enter slido in the next slide ⏰

Compute q and f

Compute q and f

q = 3, f = 1

8%

q = 3, f = -12

84%

q = 7, f = -28

2%

q = 10, f = -14

6%

Computation Graph: Backward

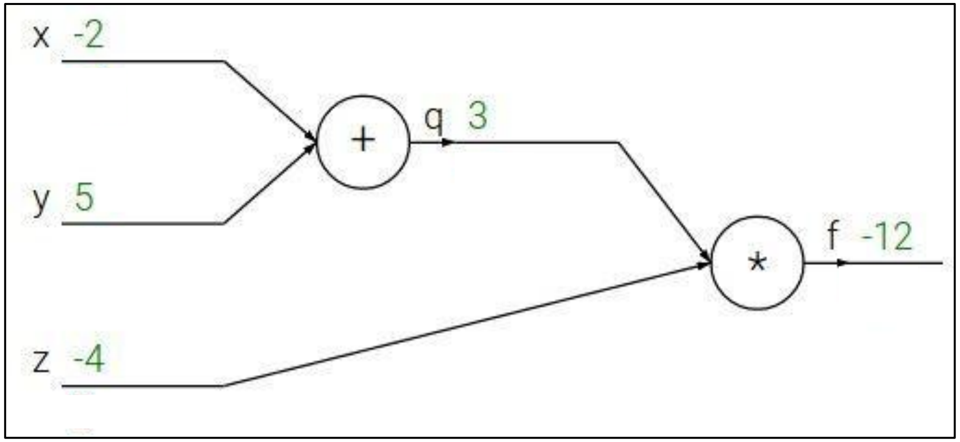
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



compute values

Computation Graph: Backward

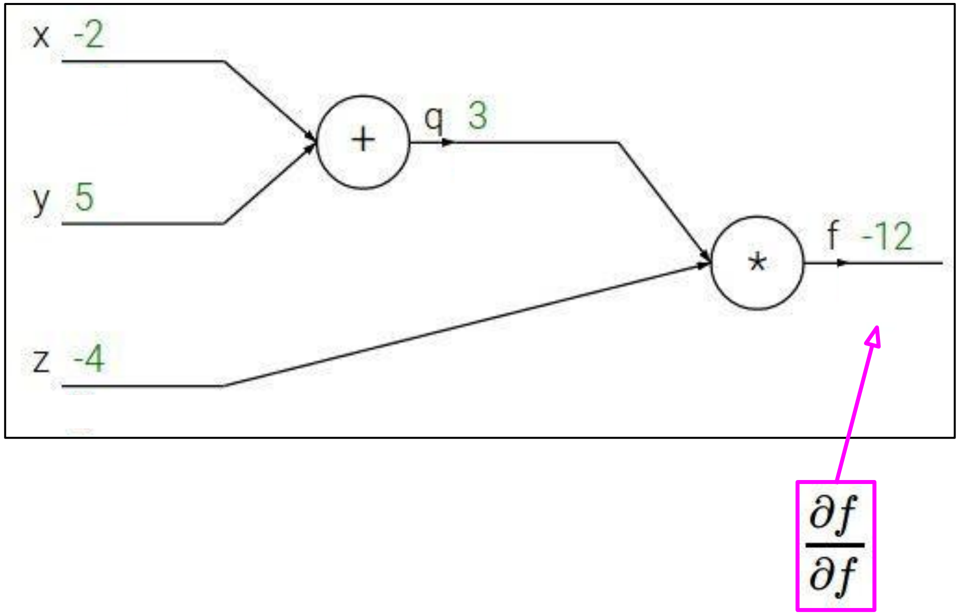
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



compute gradients

Computation Graph: Backward

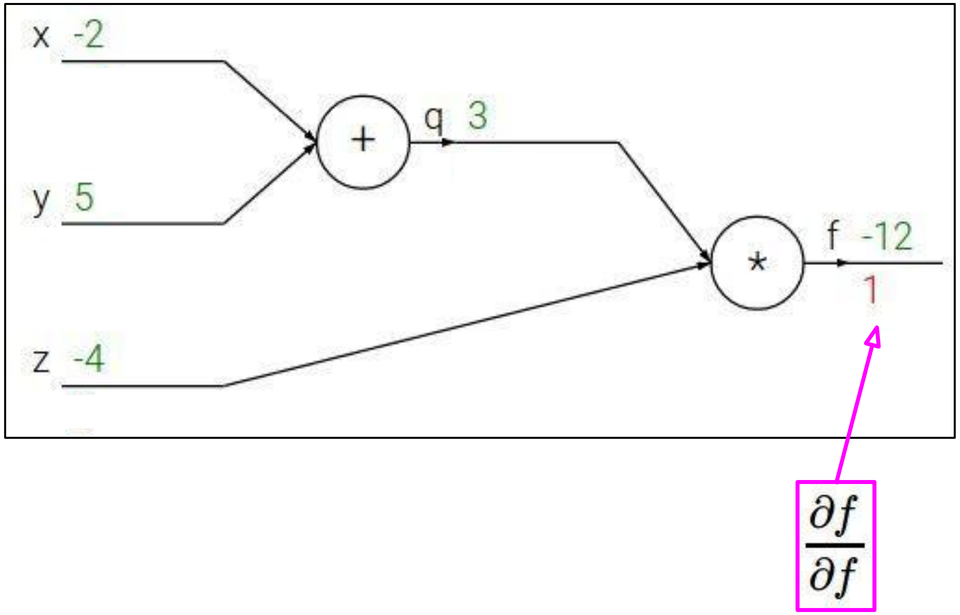
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Computation Graph: Backward

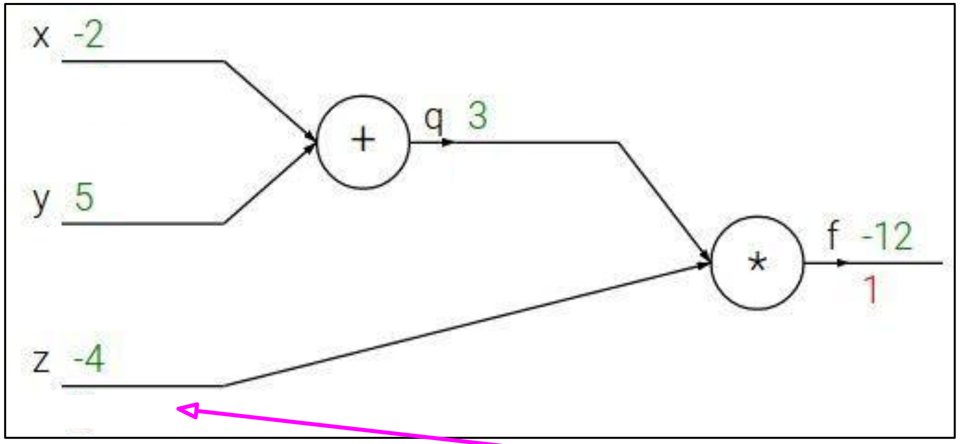
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

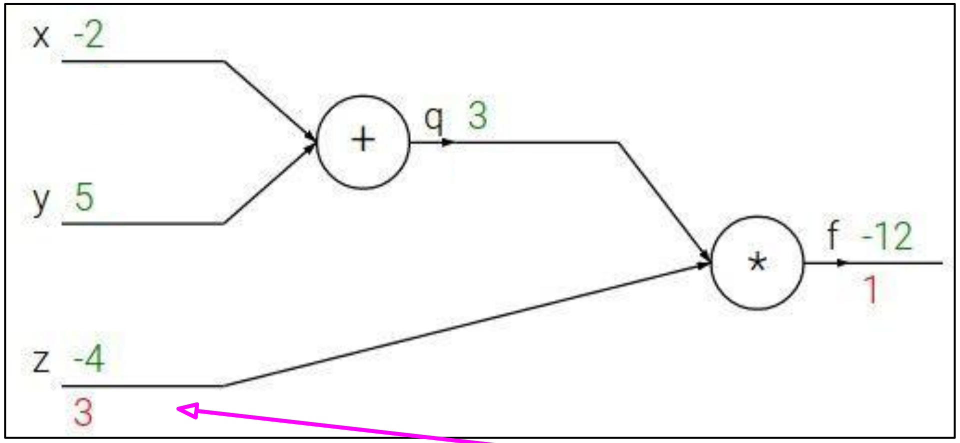


$$\boxed{\frac{\partial f}{\partial z}}$$

Computation Graph: Backward

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \boxed{\frac{\partial f}{\partial z} = q}$$

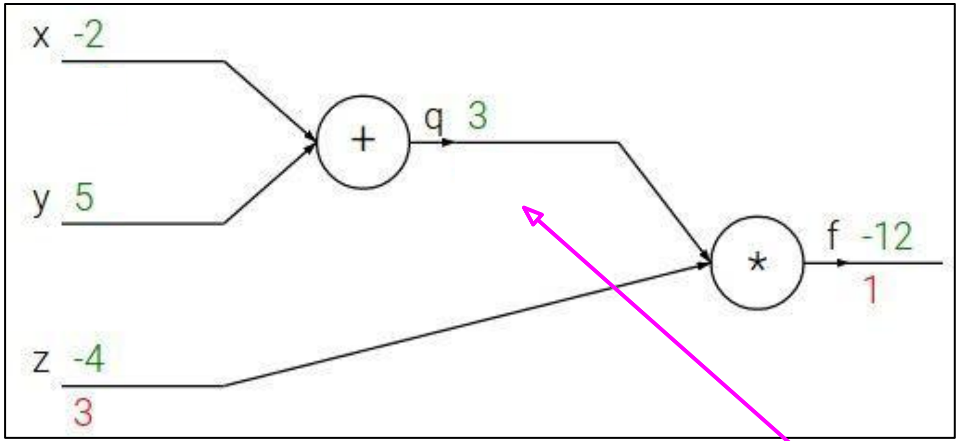
$$\boxed{\frac{\partial f}{\partial z}}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Computation Graph: Backward

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

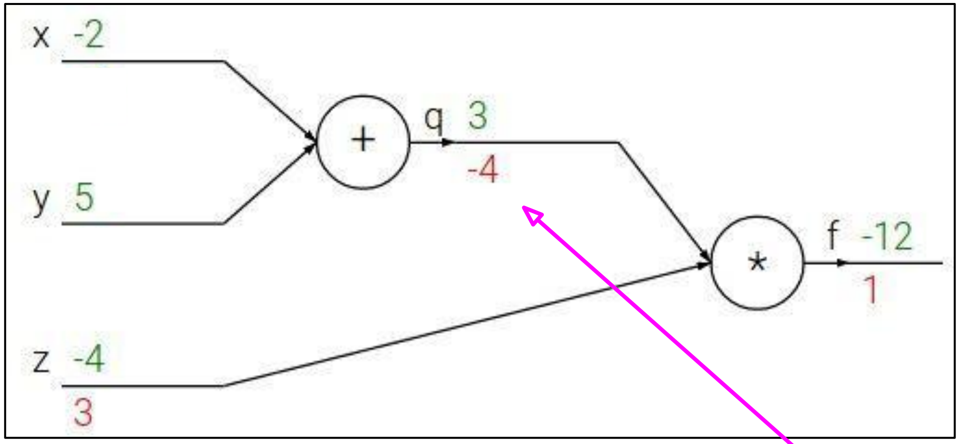
$$\boxed{\frac{\partial f}{\partial q}}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Computation Graph: Backward

$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \boxed{\frac{\partial f}{\partial q} = z}, \frac{\partial f}{\partial z} = q$$

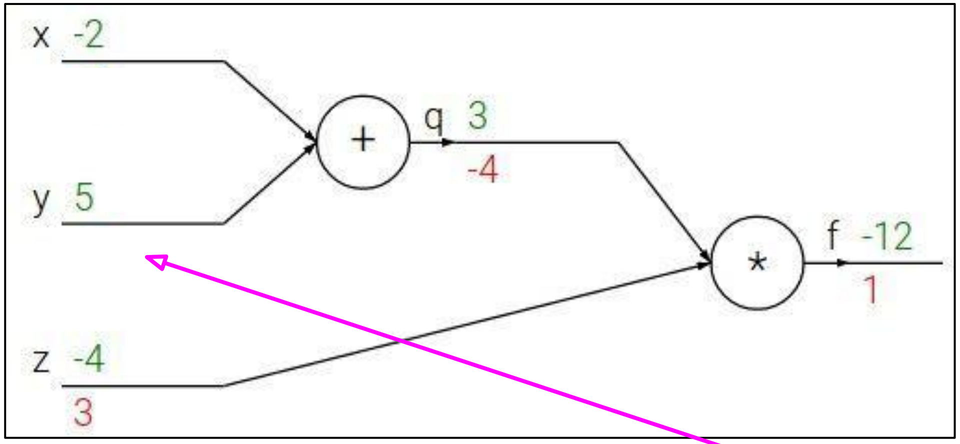
$$\boxed{\frac{\partial f}{\partial q}}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Computation Graph: Backward

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial y}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Computation Graph: Backward

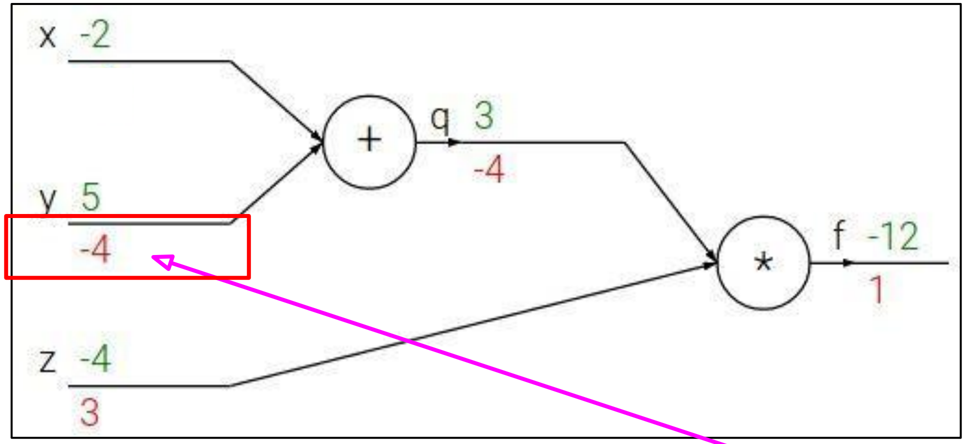
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \boxed{\frac{\partial q}{\partial y} = 1}$$

$$f = qz \qquad \boxed{\frac{\partial f}{\partial q} = z}, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial y}}$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}}$$

-4 X 1

Computation Graph: Backward

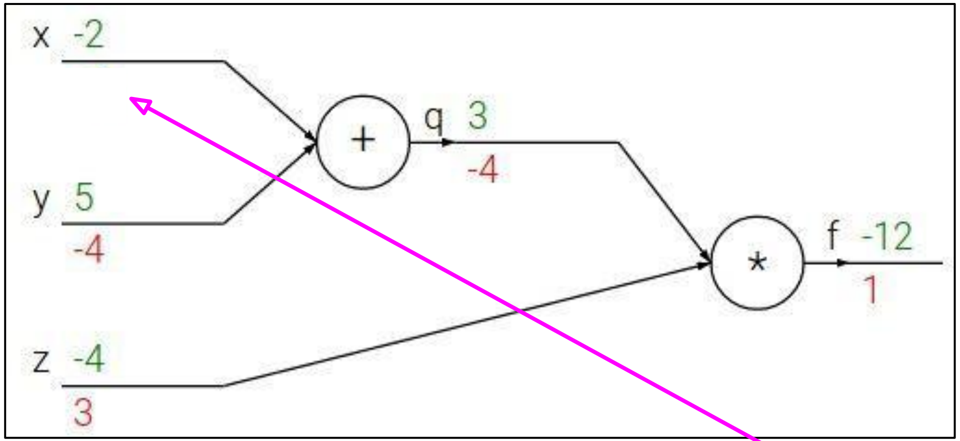
$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial x}}$$

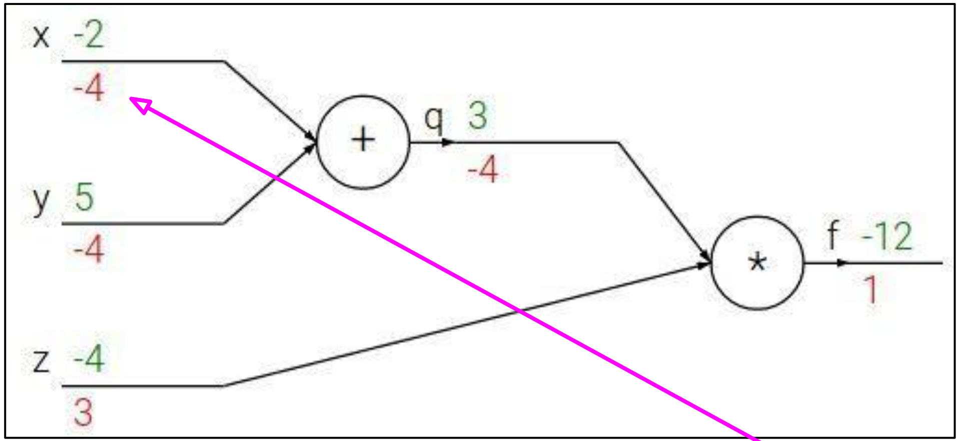
What is df/dx?

What is df/dx?

Computation Graph: Backward

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \boxed{\frac{\partial q}{\partial x} = 1}, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \boxed{\frac{\partial f}{\partial q} = z}, \frac{\partial f}{\partial z} = q$$

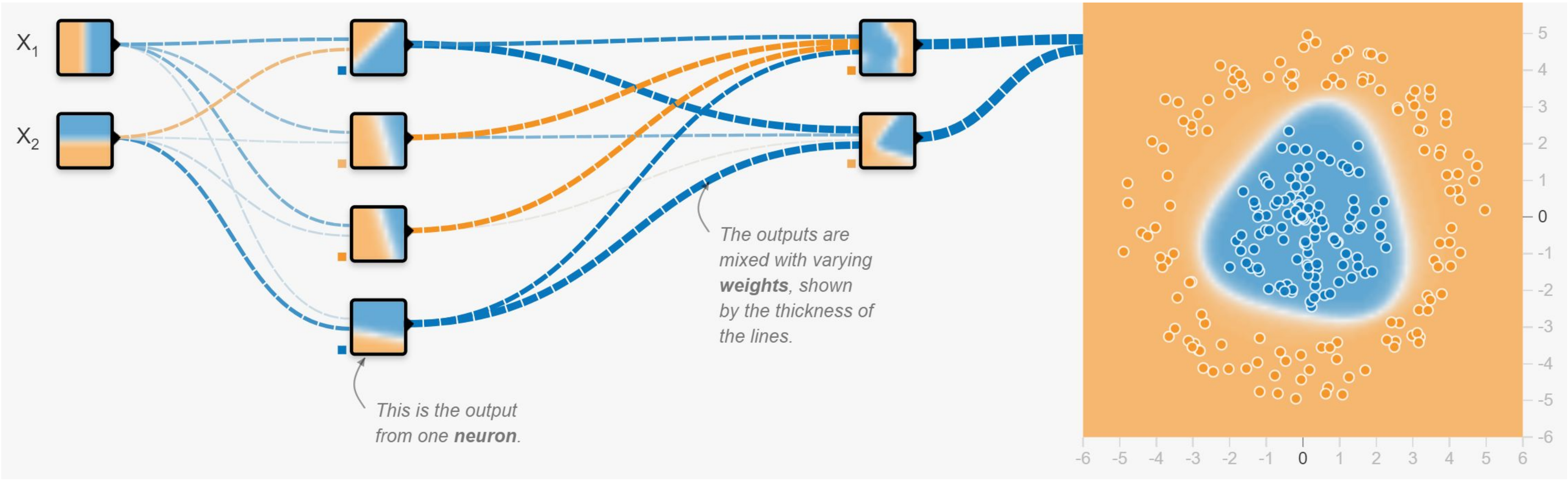
$$\boxed{\frac{\partial f}{\partial x}}$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

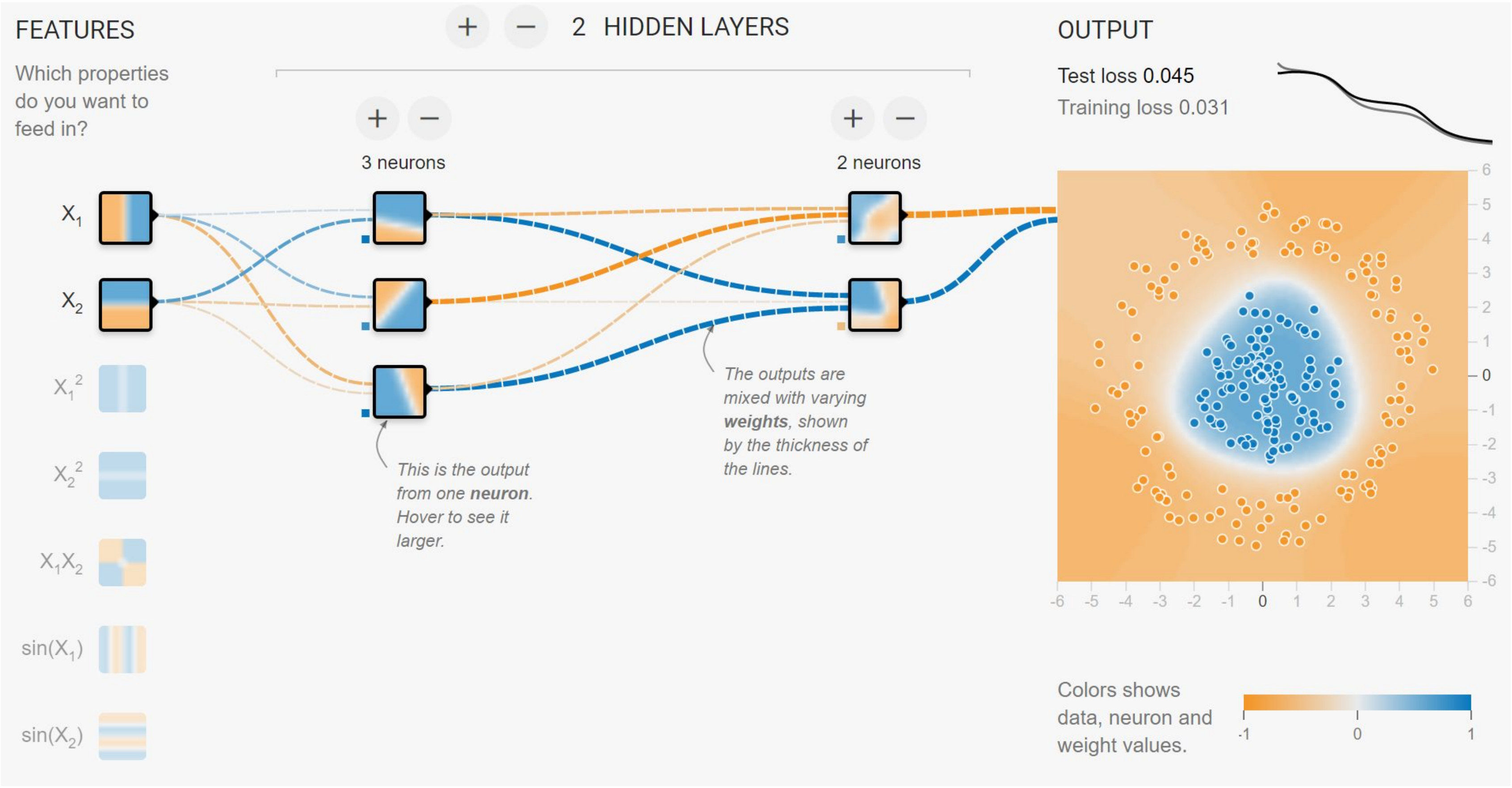
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Example



X_1
X_2
The outputs are mixed with varying **weights**, shown by the thickness of the lines.
This is the output from one **neuron**.
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
6

Training a neural net: Demo



Tensorflow playground

X: image input
Y: output

How do we train deep neural networks?

- Goal: find a set of weights (**W**)
    - $f(\mathbf{W}, \mathbf{x}_i) \sim y_i$
- How? Iteratively minimize *a **loss function*** (defined on **W**).
    - Update the weights using gradient descent and backpropagation.

- **Some practical tips:** 

X: image input
Y: output

How do we train deep neural networks?

- Goal: find a set of weights (**W**)
  - f(**W**, **x**i) ~ yi
- How? Iteratively minimize *a* ***loss function*** (defined on **W**).
  - Update the weights using gradient descent and backpropagation.
- **Some practical tips:**
  - **Normalize** your (real-valued) data.
  - **Decay** the learning rate as you get closer to the optimum.
  - **Regularize** for stability: Weight momentum, exponential average of previous gradients, dropout.
  - **Hyper-parameters** carefully by observing the behavior on val data.

Next Class

Neural Networks II:

Data augmentation, normalization layers, convolutional networks

Reading: Forsyth Ch 18.1.3-18.1.4, 18.1.6, 18.1.9