COMP 433: Introduction to Deep Learning

Practical Matters

- Labs ~10 Labs, 4 to be submitted
- 2 Problem sets
- 3 Quizzes
- Final Project — to be assigned by Week 5
- Textbook: Deep Learning by Goodfellow, Courville, Bengio — freely available online
- Dive into Deep Learning, freely available online

More Practical Matters

- Lab 1 is posted
- Office hours
  - Live office hours are Monday at 4-4:45 in room ER 958
  - By appointment on MS Teams
- MS Teams
  - Join MS Teams, link on moodle
  - Some announcements will be made there

Evaluation

- Assignments: 30%
  - Programming assignments (roughly 80% of assignments)
  - Some theoretical and written questions
- Project: 30%
- Quizzes: 30%
  - 3 Quizzes
- Labs: 10%
  - Only first 4 are graded

Pre-requisites

- Linear Algebra, Multivariable Calculus, Probability and Statistics, Algorithms
- Knowledge of *some* programming language
- Key concepts reviewed when appropriate

Outline

- Machine Learning Foundations
- Introduction to Neural Networks
- Backpropagation and Automatic Differentiation Software
- Optimization for Deep Learning
- Practical Training Recipes
- Convolutional Neural Networks
- RNNs and sequence models
- Attention and Self-Attention
- Multi-Task and Transfer Learning
- Deep Generative Models
- Self-supervised Learning

What You Will Learn From This Class

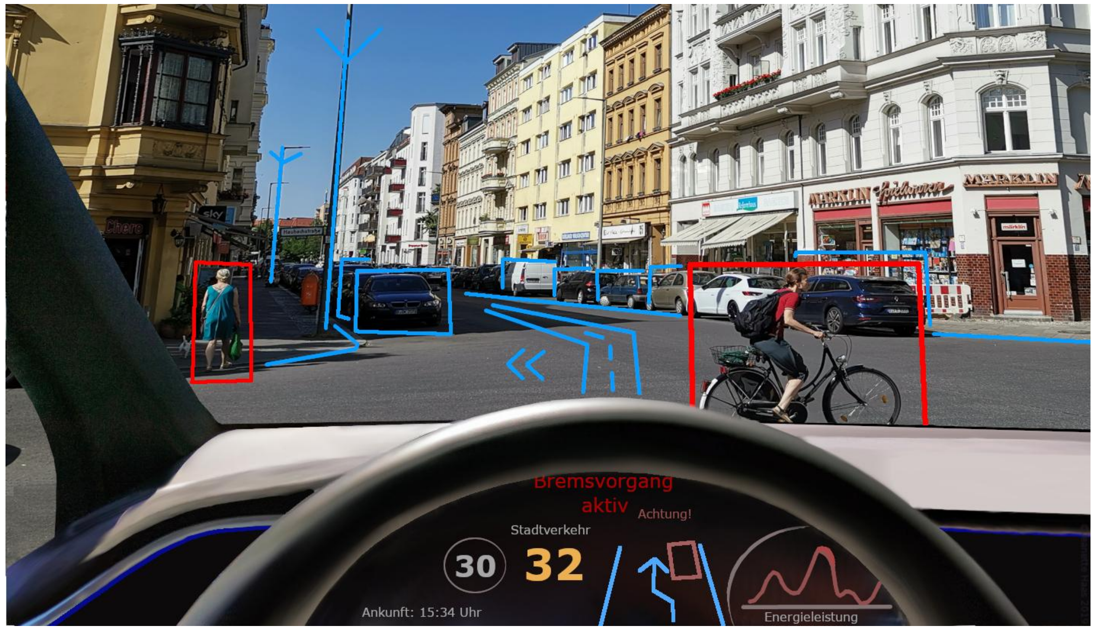
- In-depth practical and (some) theoretical understanding of the building blocks of deep learning models
- Hone or introduce important technologies like Python, Numpy, Pytorch
- Overview of some cutting edge techniques

Concepts to Reinforce or Learn

- Covered in lecture:
  - Basic linear algebra concepts
  - Relevant machine learning concepts
- From labs and additional material:
  - Python - one of the most used programming languages in the world
  - Using Tensor/Matrix libraries - e.g. Numpy

Applications

Autonomous Driving



Speech Recognition

Machine Translation

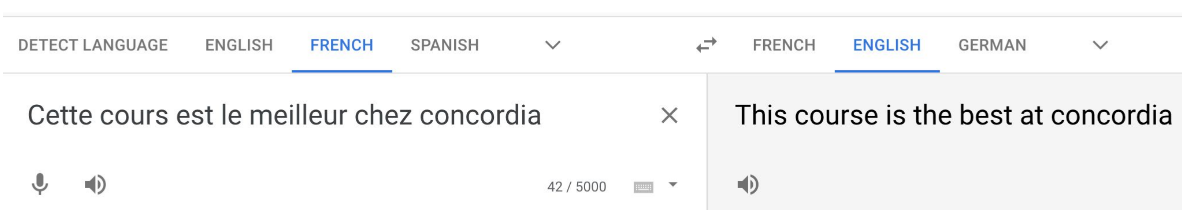


Image generation with prompts

Text Prompt

an armchair in the shape of an avocado. . . .

AI Generated images

https://openai.com/research/dall-e

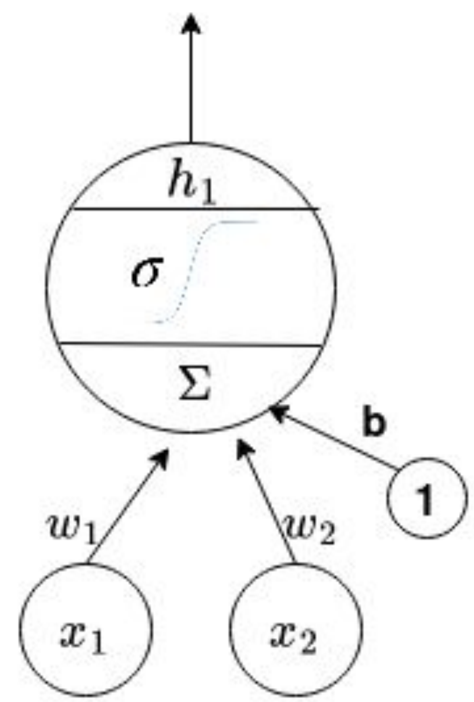
Deep Learning

- Subset of machine learning
- The main object of study is the Neural Network
- Associated with modular but powerful/expressive framework for creating predictive models

Neural Networks

- Simple computation blocks

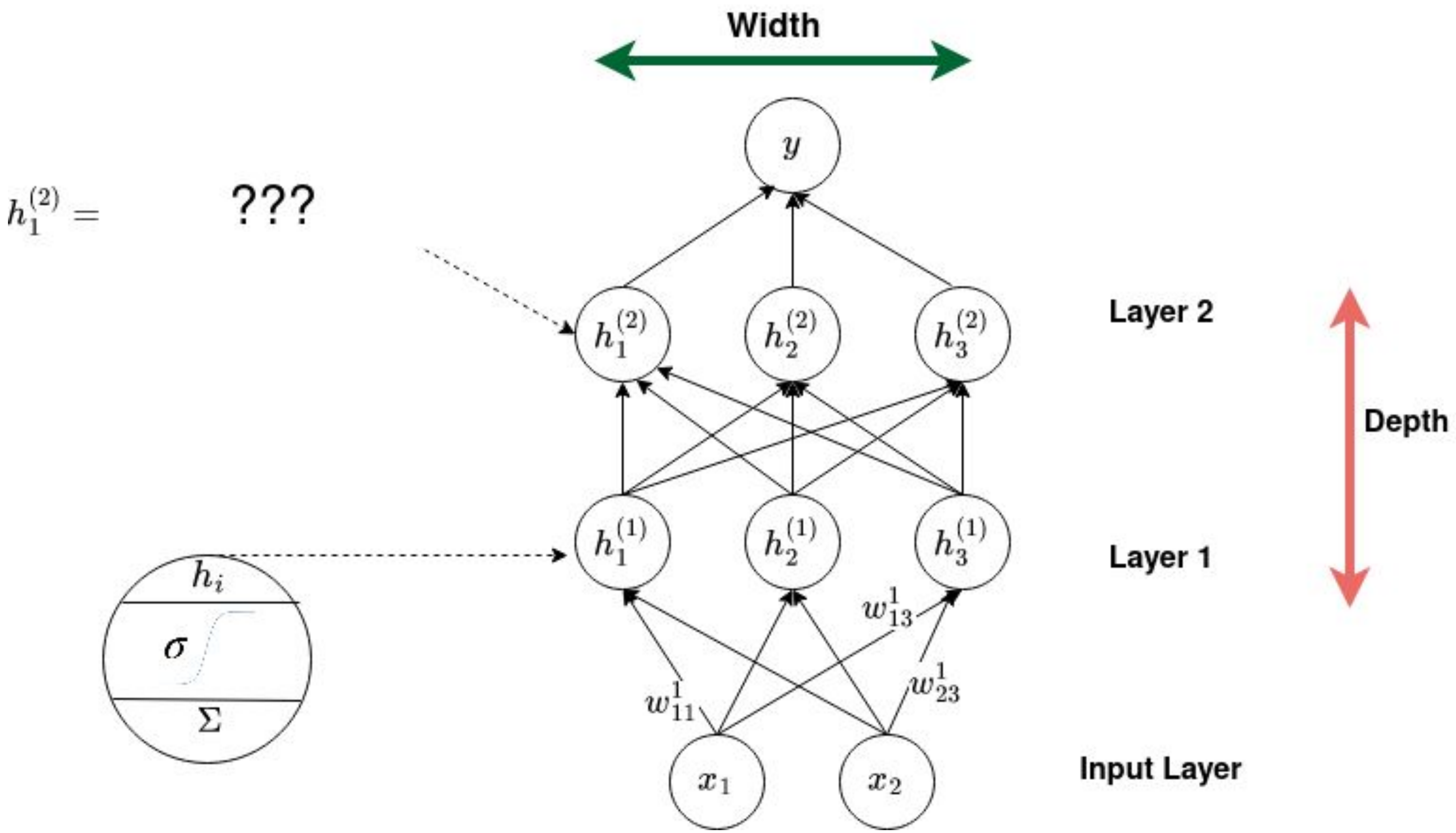
x_1, x_2	**inputs**
w_1, w_2	**weights**
b	$bias$
σ	**activation function**
h_1	**neuron activation**



$$h(x_1, x_2) = \sigma\left(w_1 x_1 + w_2 x_2 + b\right)$$

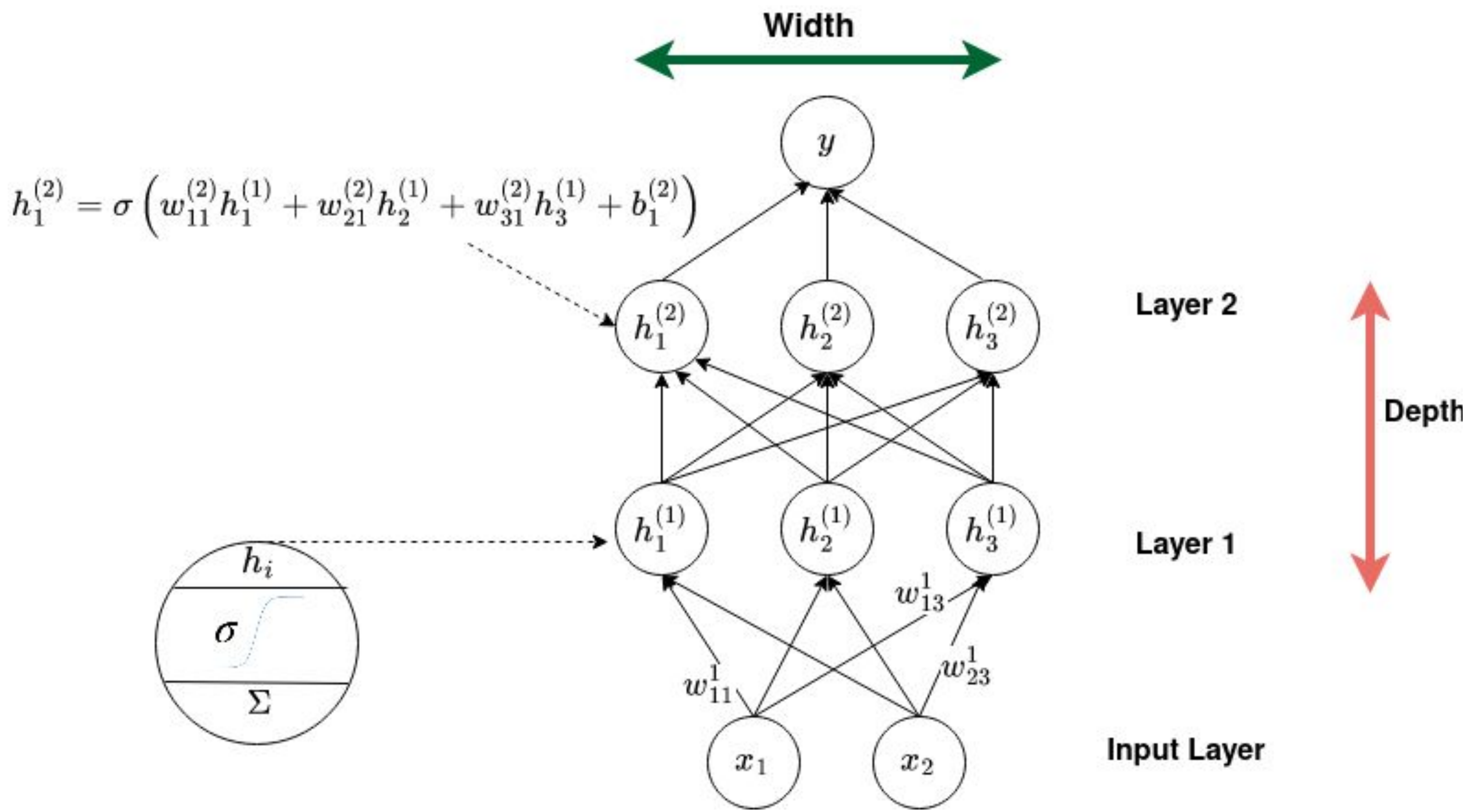
Neural Networks

- Simple computation blocks that work together



Neural Networks

- Simple computation blocks that work together



Review of Linear Algebra

- Scalar
- Vector
- Matrices
- Tensors

Representation in numerical software packages

Numpy

```python
import numpy as np

# scalar
a =5

# vector
b = np.array([1,1])
c = np.array([2,2])
np.dot(b,c)
```

```
4
```

```python
A = np.array([[1,2],[3,4]])
B = np.array([[2,3],[4,1]])
print(A)
```

```
[[1 2]
 [3 4]]
```

PyTorch

```python
import torch

b = torch.from_numpy(b)
c = torch.from_numpy(c)
print(b)

```

```
tensor([1, 1])
```

```python
torch.dot(b,c)
```

```
tensor(4)
```

Matrix & Elementwise Multiplications

Numpy

```python
import numpy as np
A = np.array([[1,2],[3,4]])
B = np.array([[1,1],[1,1]])

#Matrix multiply AB
print('Matrix Multiply:')
print(np.matmul(A,B))

print('ElementWise Multiply:')
#Elementwise Multiply
print(A*B)
```

```
Matrix Multiply:
[[3 3]
 [7 7]]
ElementWise Multiply:
[[1 2]
 [3 4]]
```

PyTorch

```python
import torch
A = torch.tensor([[1,2],[3,4]])
B = torch.tensor([[1,1],[1,1]])

#Matrix multiply AB
print('Matrix Multiply:')
print(torch.matmul(A,B))

print('ElementWise Multiply:')
#Elementwise Multiply
print(A*B)
```

```
Matrix Multiply:
tensor([[3, 3],
        [7, 7]])
ElementWise Multiply:
tensor([[1, 2],
        [3, 4]])
```

Tensors

[36]

```
c = torch.tensor([[1,2],[3,4]])
print(c)
print(c.shape)
```

```
tensor([[1, 2],
        [3, 4]])
torch.Size([2, 2])
```

[37]

```
d = torch.tensor([[[1,2],[3,4]],
                  [[3,2],[6,7]]])
print(d)
print(d.shape)
```

```
tensor([[[1, 2],
         [3, 4]],

        [[3, 2],
         [6, 7]]])
torch.Size([2, 2, 2])
```

Indexing

[16]

```python
d = torch.tensor([[[1,2],[3,4]],
                  [[3,2],[6,7]]])
print(d)
print(d.shape)
```

```
tensor([[[1, 2],
         [3, 4]],

        [[3, 2],
         [6, 7]]])
torch.Size([2, 2, 2])
```

[17]

```python
print(d[0,:,:])
```

```
tensor([[1, 2],
        [3, 4]])
```

```python
print(d[0,0,:])
```

```
tensor([1, 2])
```

Matrix Inverse

Identity Matrix

$$\boldsymbol{I}_n \longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inverse

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n.$$

Norms

L_p norm

$$||\boldsymbol{x}||_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

Shorthand for p=2, length of a vector

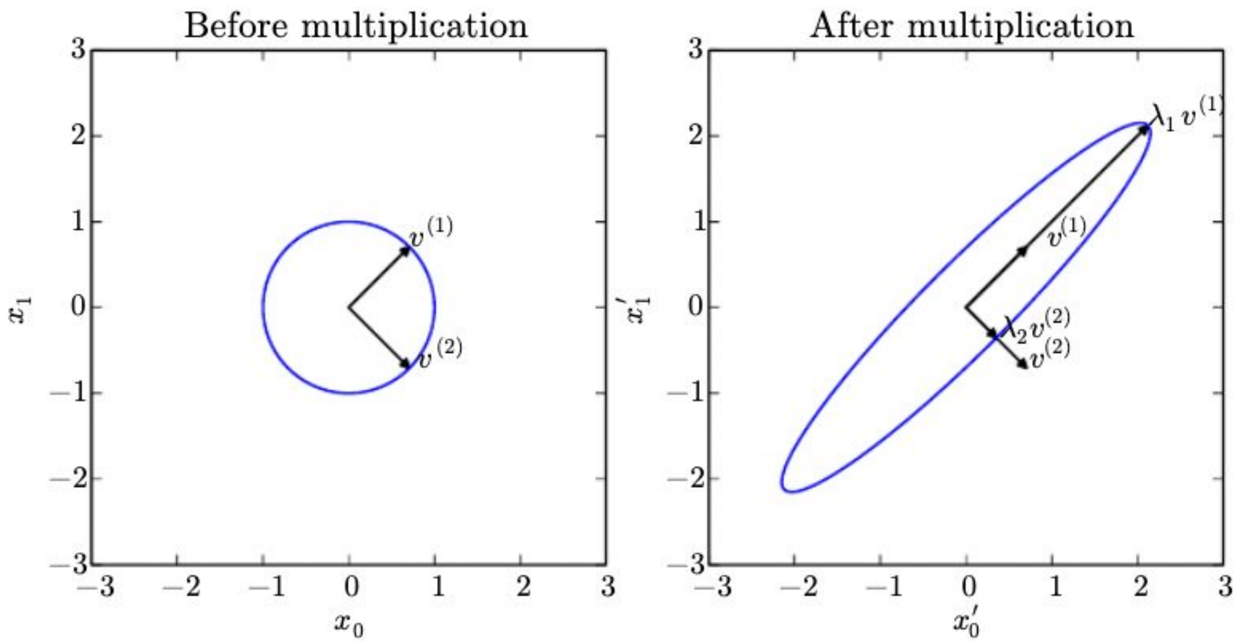
$$||\boldsymbol{x}||$$

Eigenvalues/vectors

- Eigenvalues are a property of a square matrix

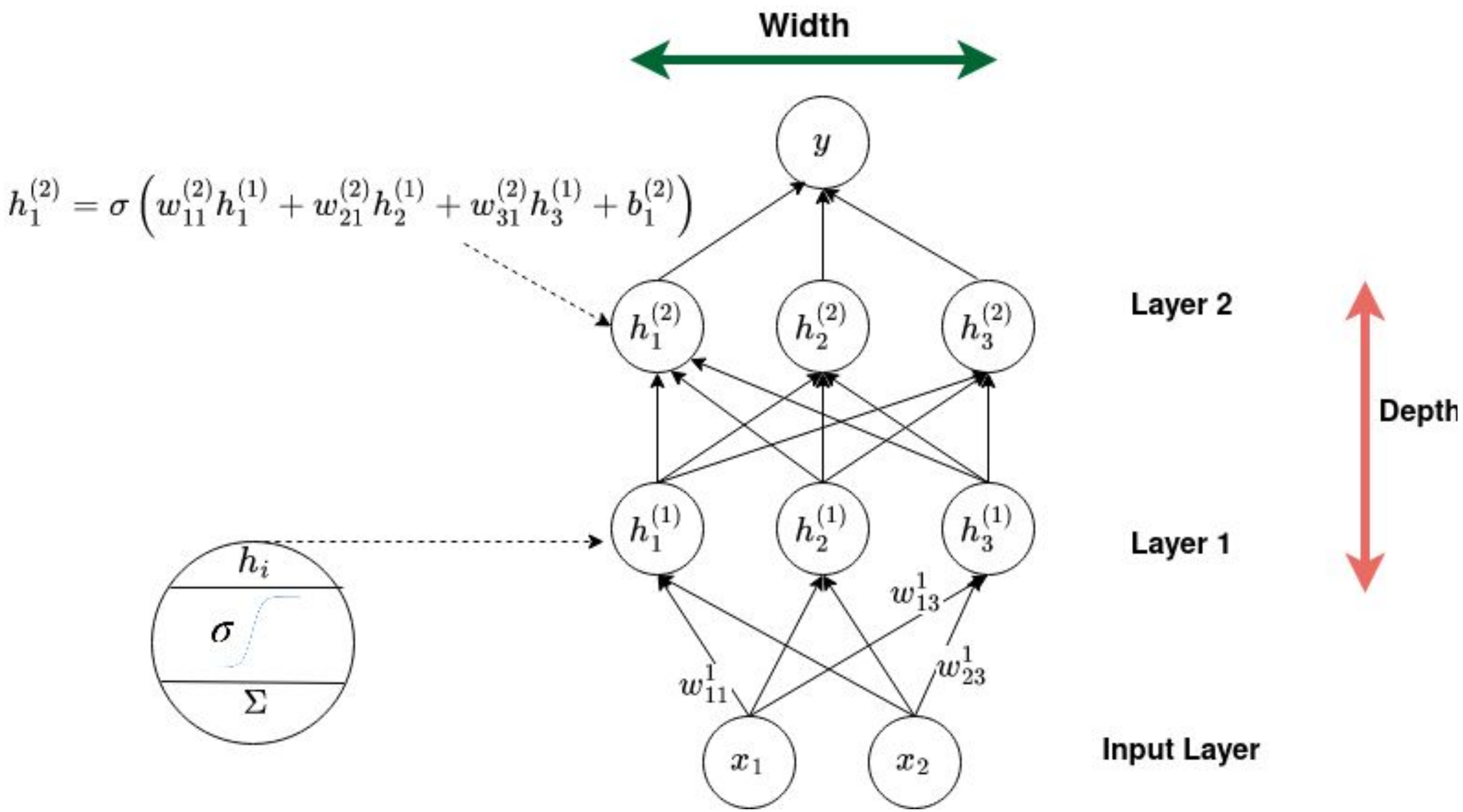
Eigenvector is a vector such that applying A is a rescaling

$$Av = \lambda v.$$



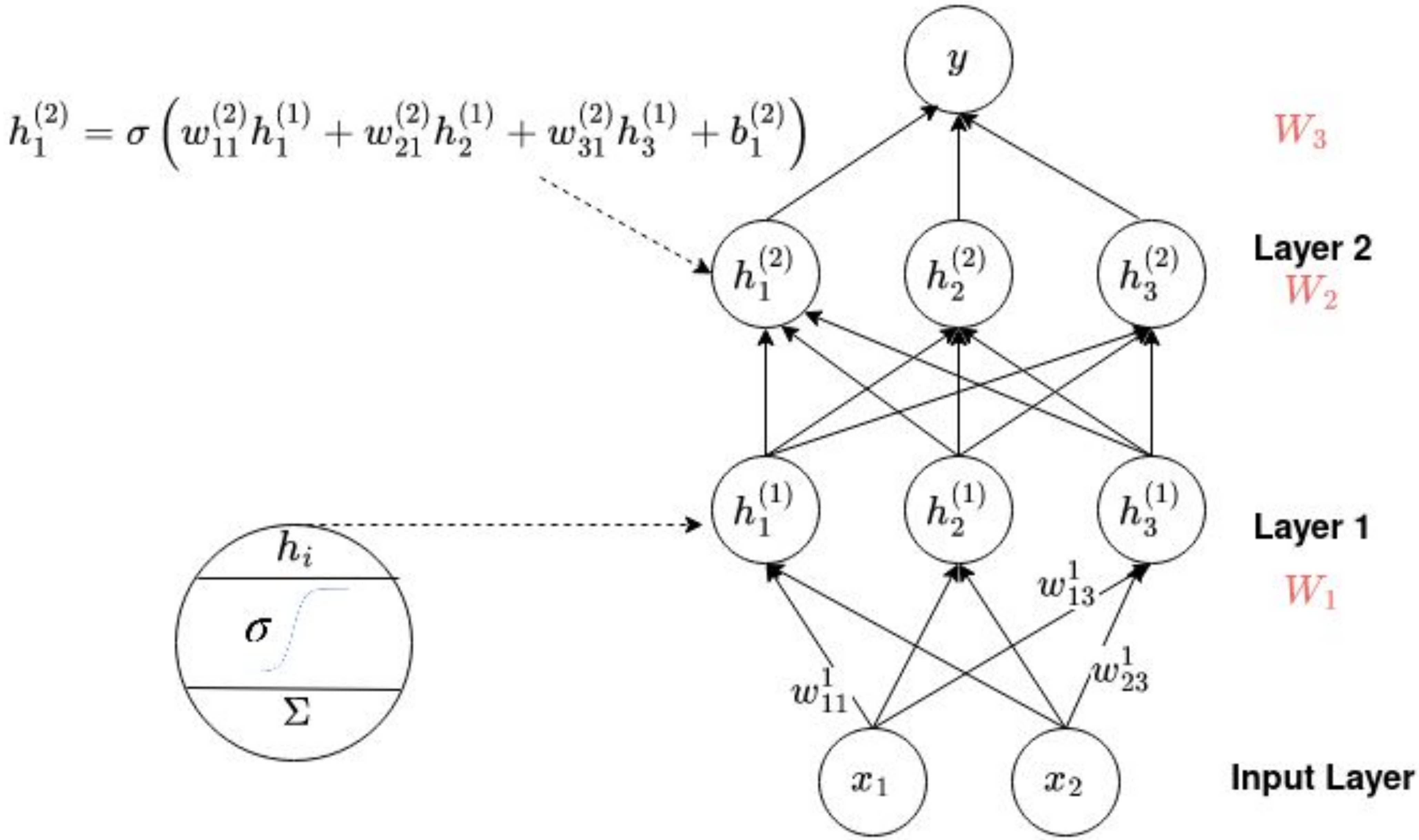
Neural Networks

- Simple computation blocks that work together



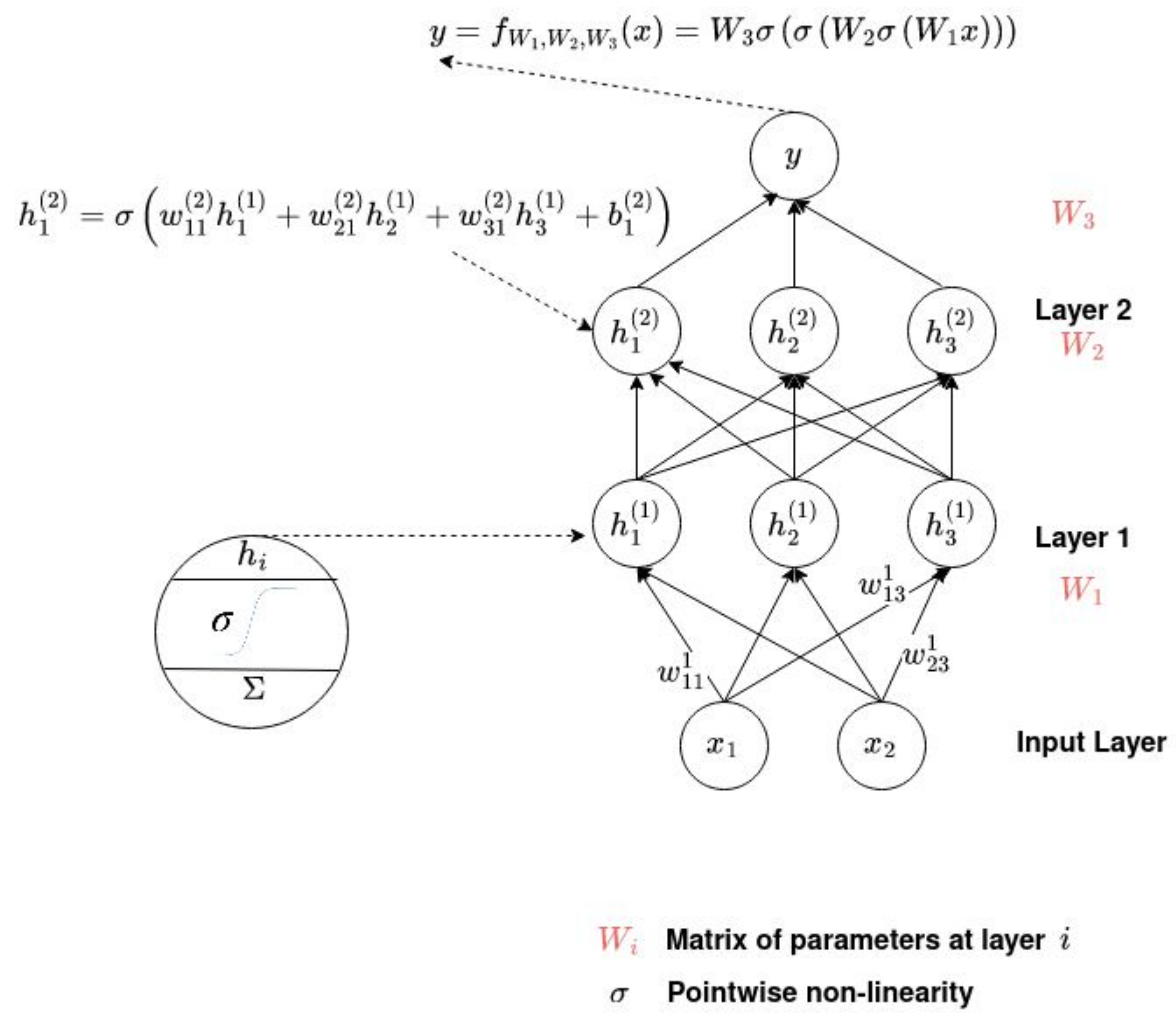
Neural Networks

- Can be written as composition of simple linear algebra operations (matrix multiplies and elementwise non-linearities)



Neural Networks

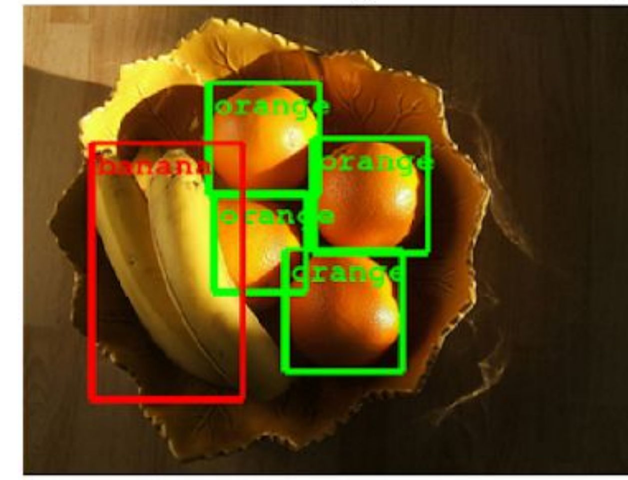
- Can be written as composition of simple linear algebra operations (matrix multiplies and elementwise non-linearities)
- Bias term can be included in W to simplify notations

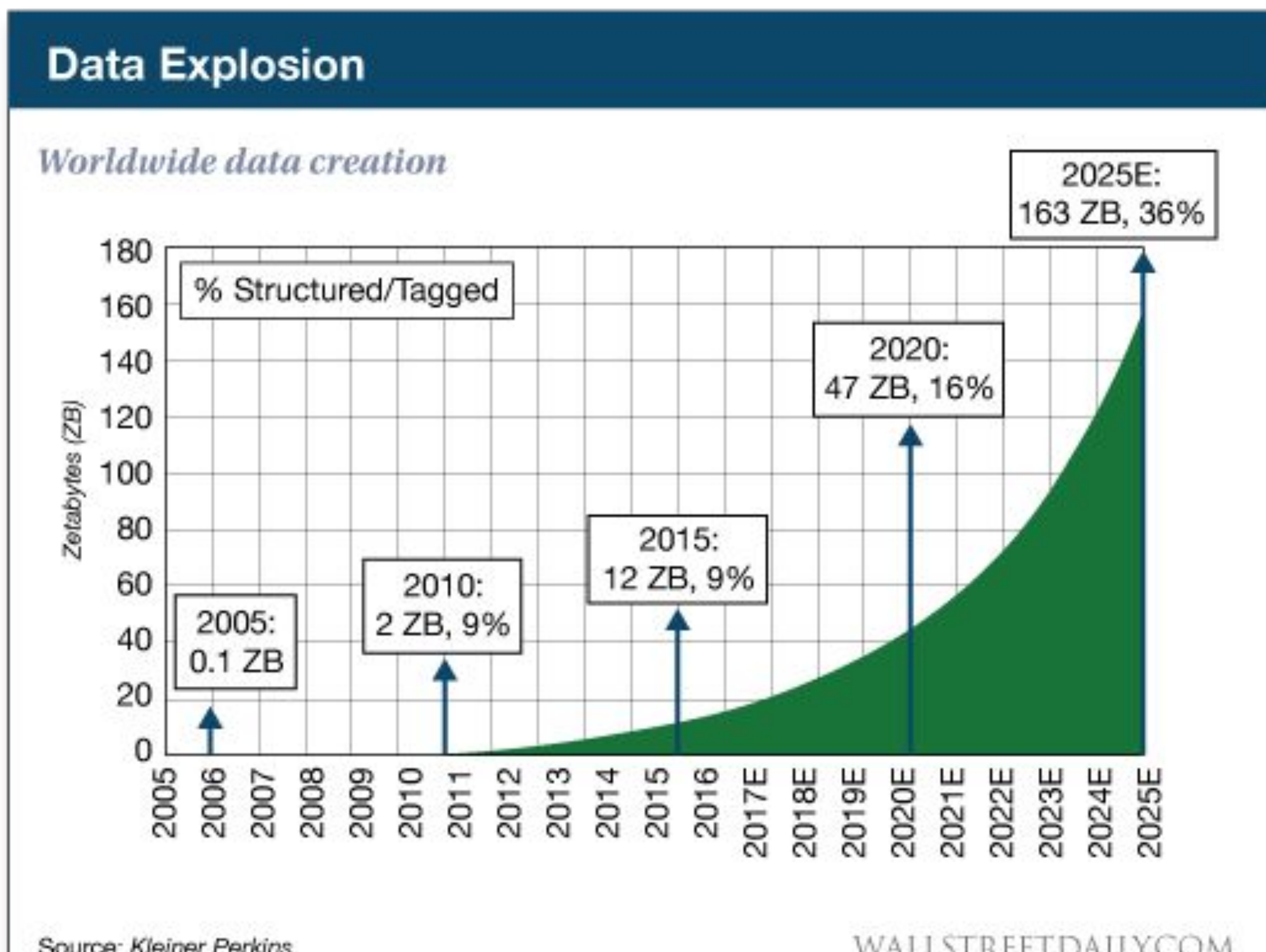


Intro to Machine Learning

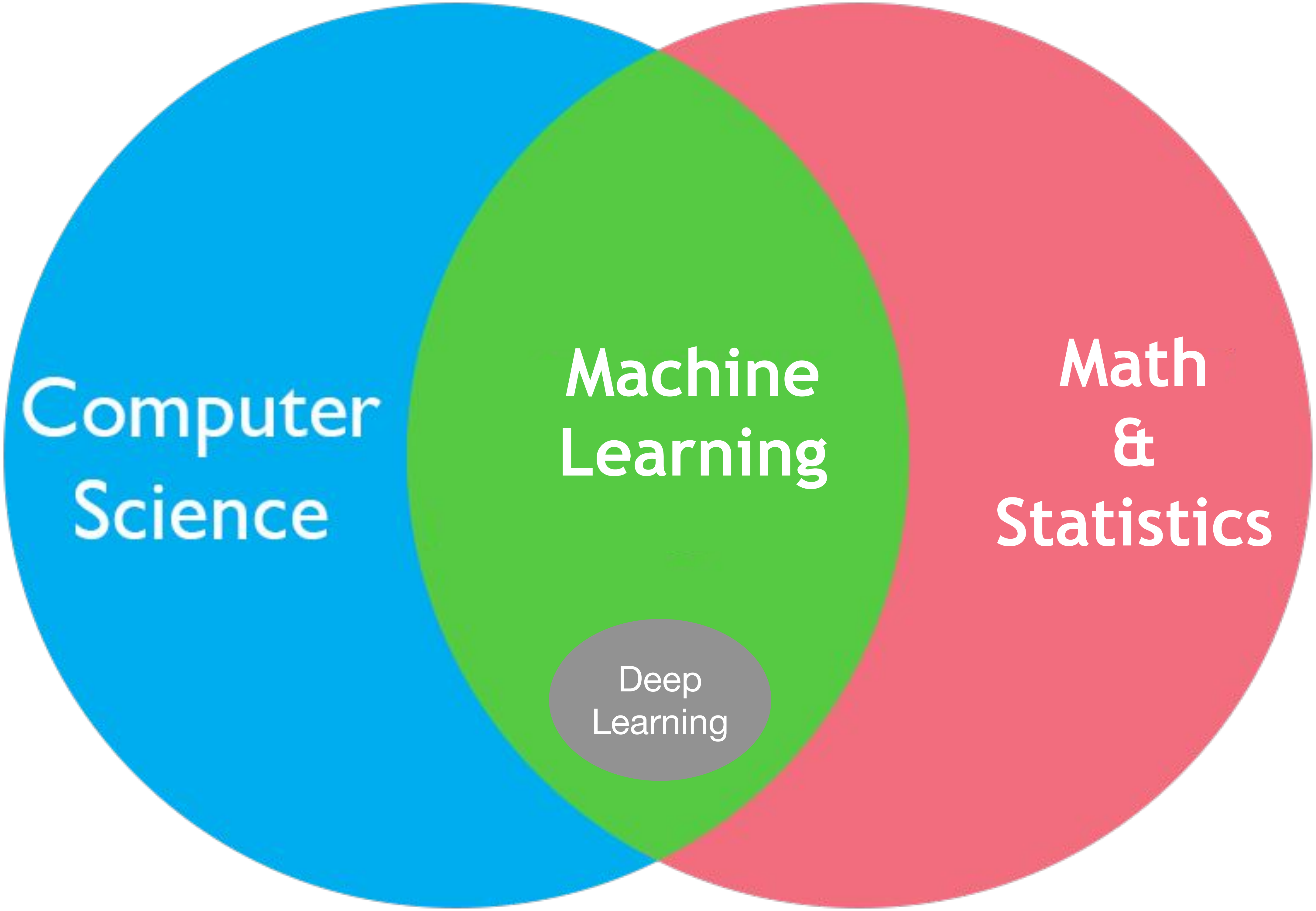
Data Explosion

- Millions and billions of measurements
- Millions and Billions of examples



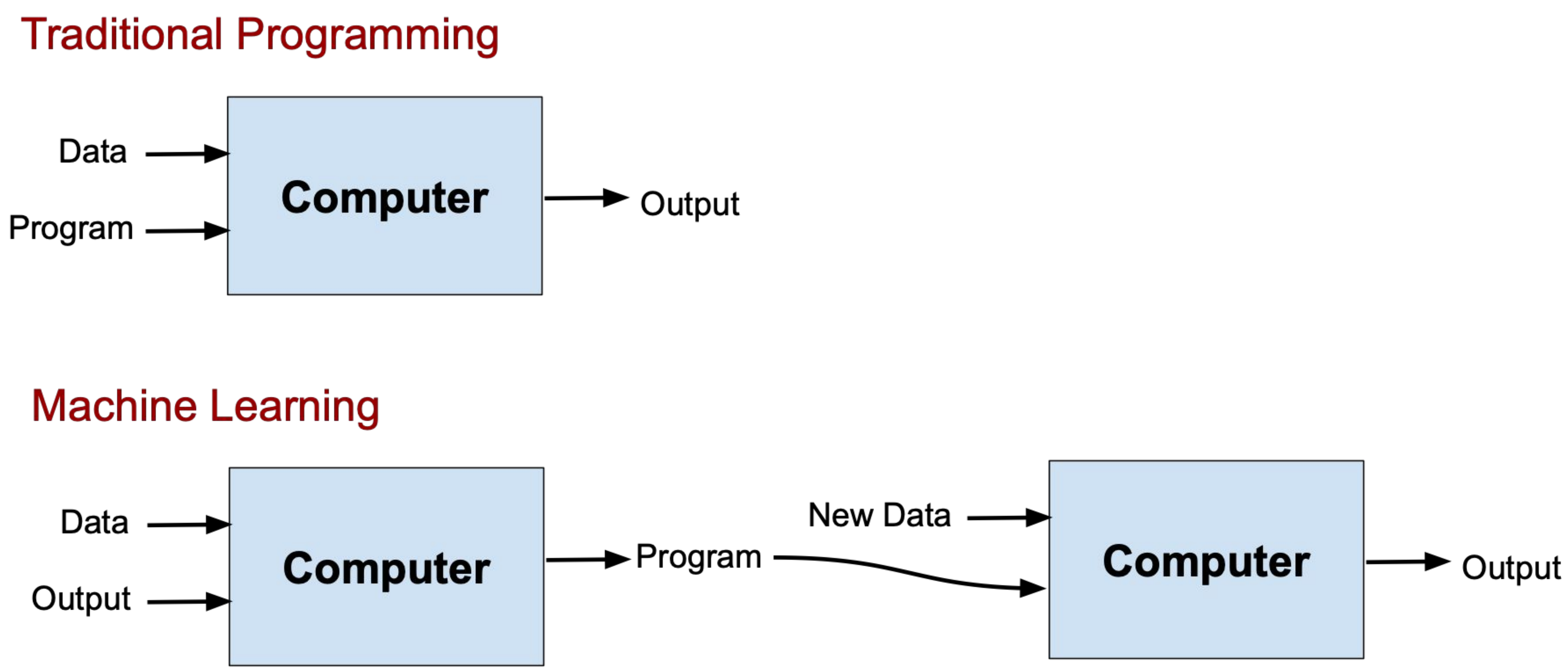


What is Machine Learning



What is Machine Learning

- "Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed." -Arthur Samuel (1959)
- "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T, as measured by P, improves with experience E." Michel (1997)

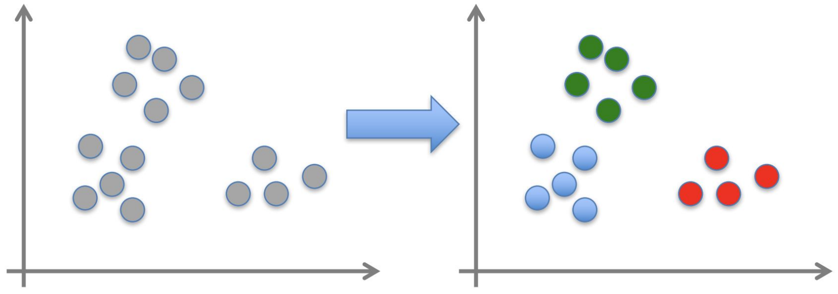


Types of Machine Learning

- ***Supervised Learning*** (Given: Training Data and Output examples)

Or

- ***Unsupervised Learning*** (Given: Training Data and no output)
  - Data exploration and visualization



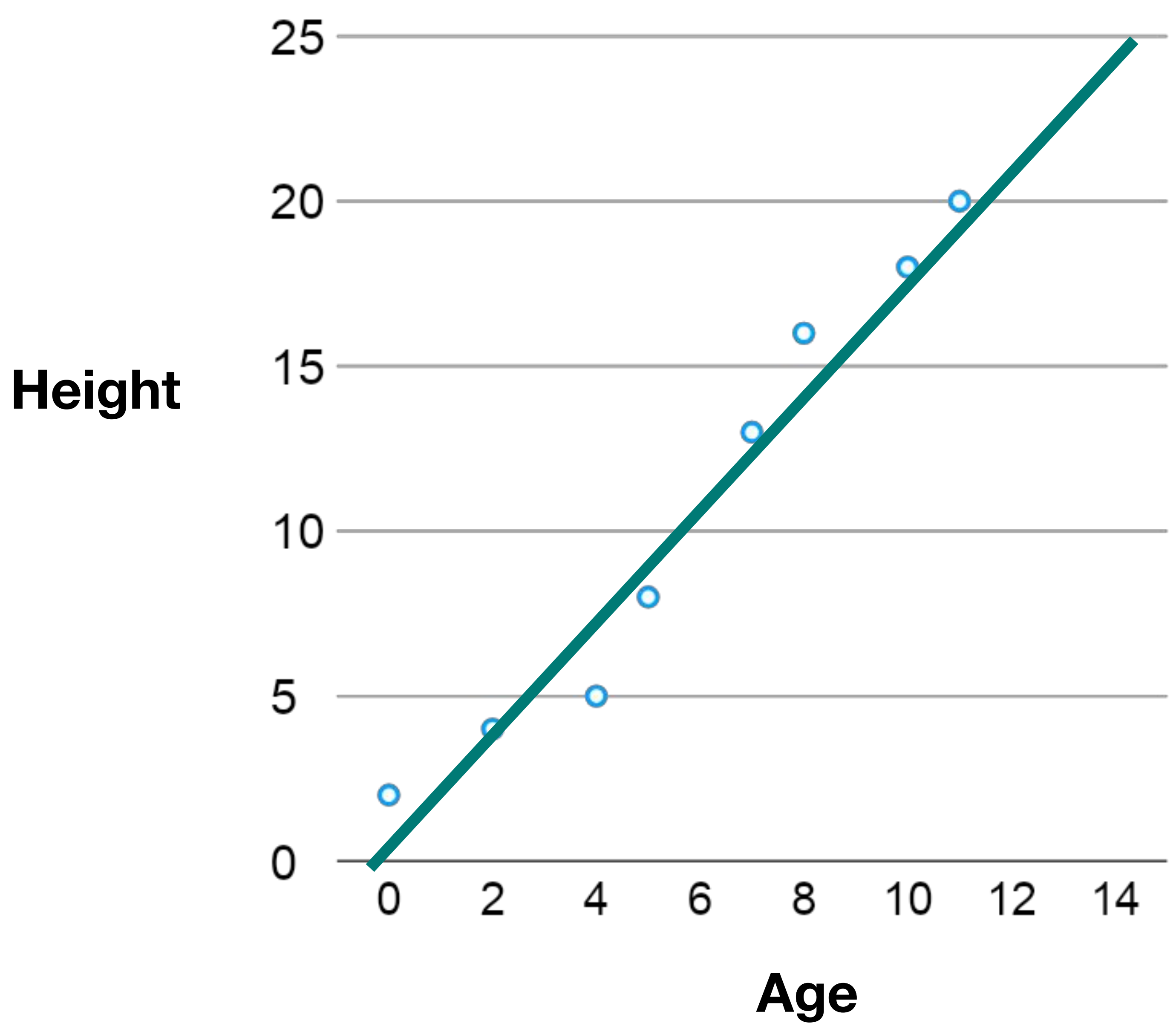
  - Generating data

- ***Reinforcement Learning***
  - Rewards from sequences of actions

Supervised Learning: Regression

How tall is my dog? (depending on age)

	Age	Height
Dog 1	1	3
Dog 2	3	4.5
Dog 3	1	1
Dog 4	5	5.5
Dog 5	9	16
Dog 6	4	5.2
Dog 7	6	7.5
Dog 8	12	20



Supervised Learning: Regression

Training Data

	Age	Height
Dog 1	1	3
Dog 2	3	4.5
Dog 3	1	1
Dog 4	5	5.5
Dog 5	9	16
Dog 6	4	5.2
Dog 7	6	7.5
Dog 8	12	20

Testing Data

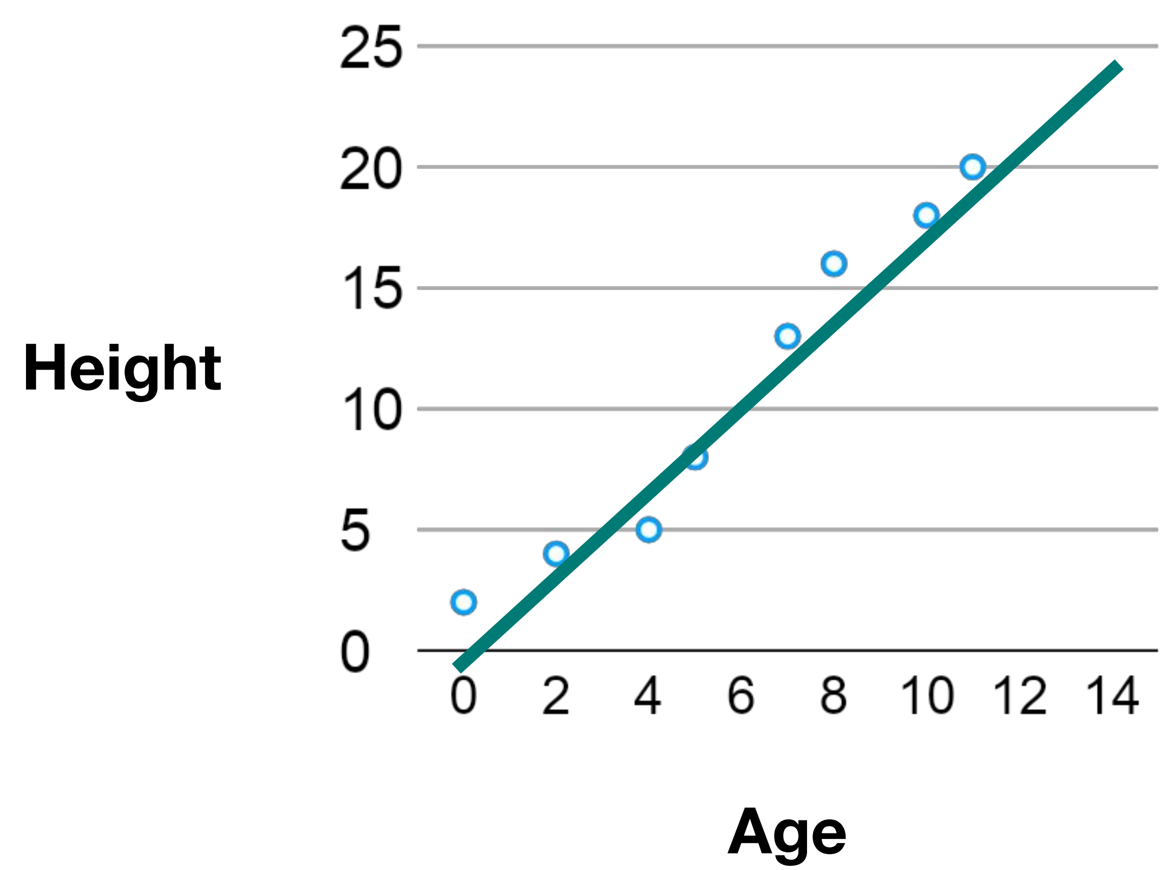
	Age	Height
New Dog 1	1.2	?
New Dog 2	3.1	?

Use Model:

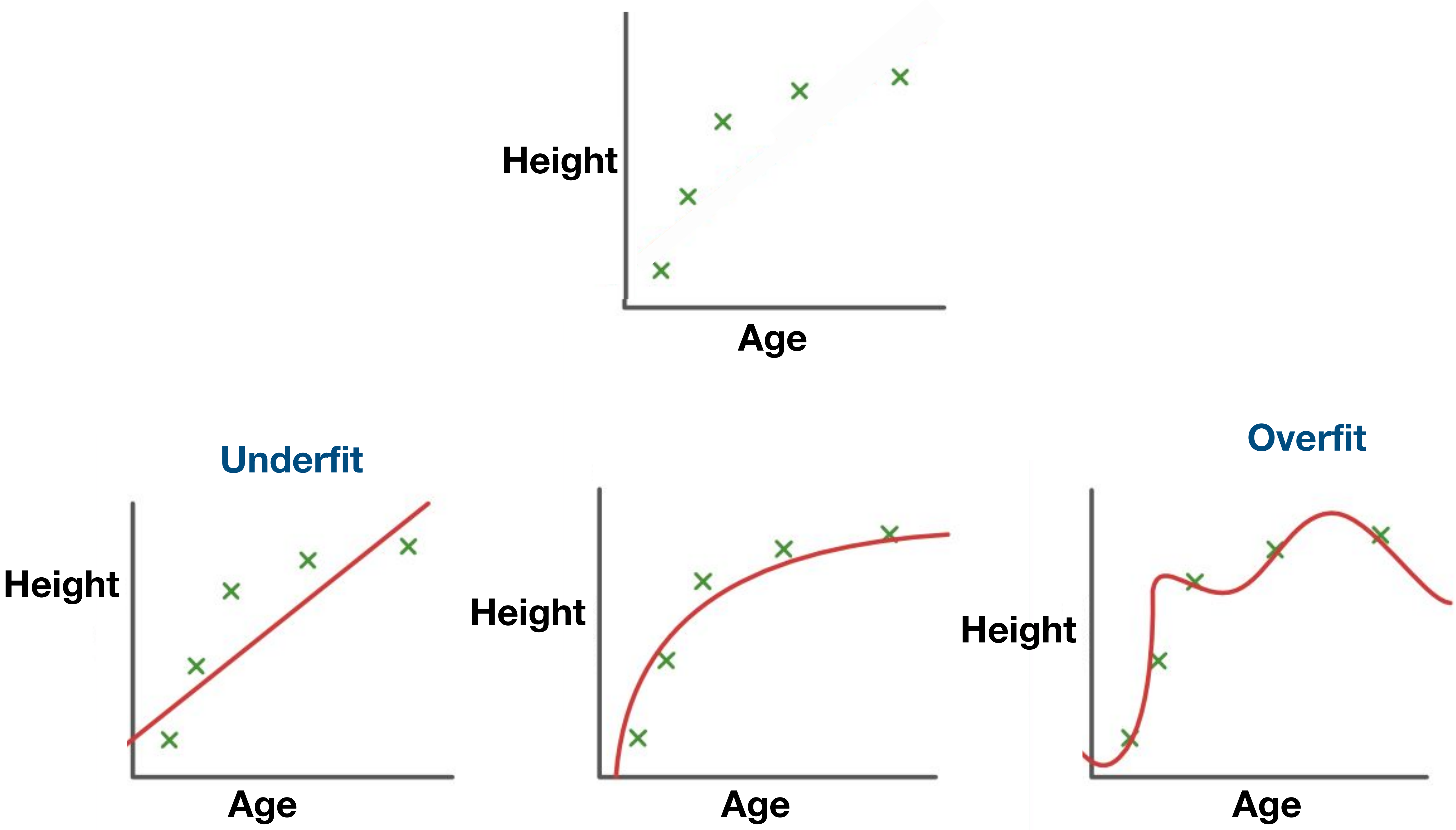
$$y = a^{*}x + b$$

Optimize Objective:

$$\min_{a,b} \sum_i (\text{height}_i - (a \, {}^{*}\text{age}_i + b))^2$$

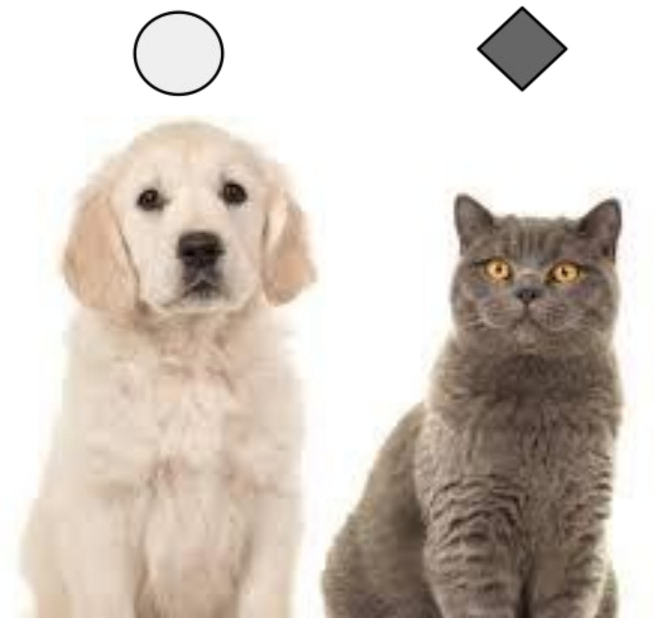


Overfitting and Underfitting



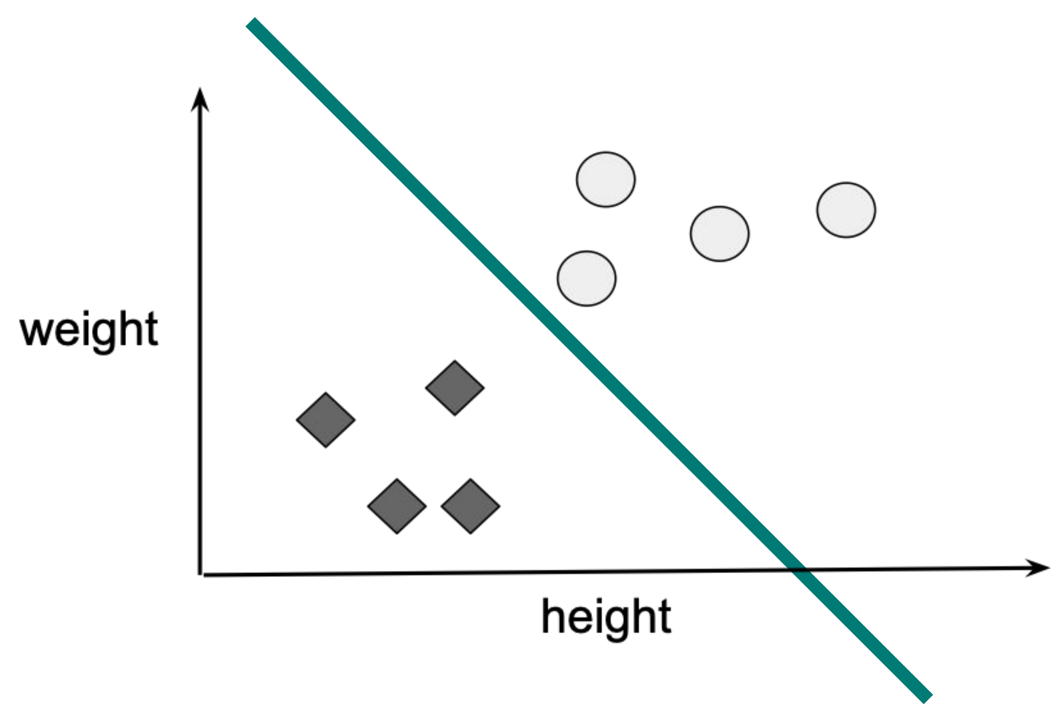
Supervised Learning: Classification

Classify cats and dog using height and weight



	Height	Weight
Cat 1	110 cm	20 kg
Cat 2	108 cm	15kg
Cat 3	109 cm	21kg
Cat 4	112.5 cm	23 kg
Dog 1	130 cm	28kg
Dog 2	125 cm	29kg
Dog 3	109.5 cm	30 kg
Dog 4	130 cm	22kg

Decision Boundary



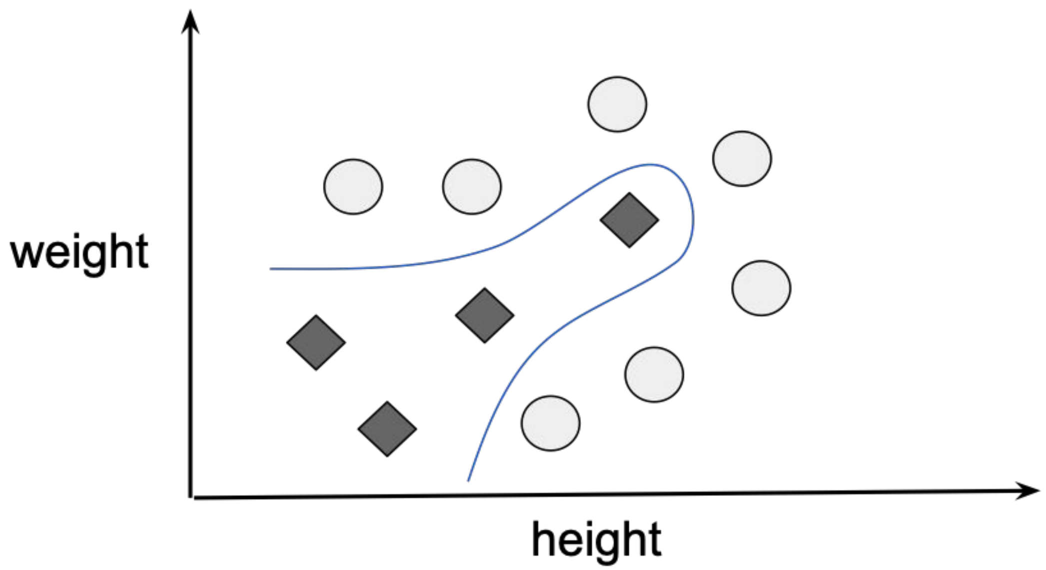
Learning in High Dimensions Data is Harder

Classify cats and dog from raw pixels

VS

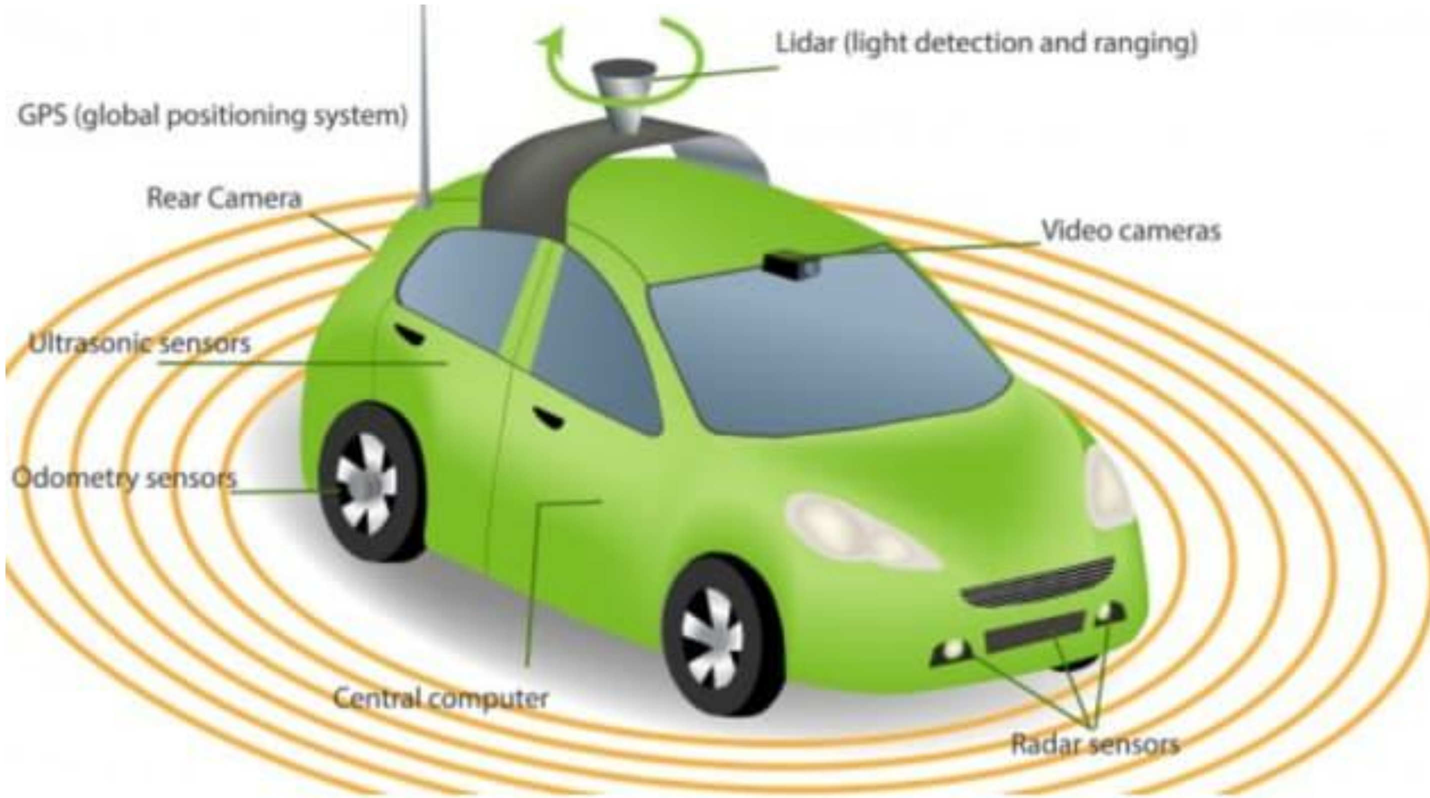
High Dimensional

Images are 3x512x512 ~ 1 million dimensions



Non-Linear Boundary

Autonomous Driving Example



- Supervised Learning
  - Identify objects
  - Uncertainty
  - Driving decisions from example
- Unsupervised Learning
  - Which data to label?
- Reinforcement Learning
  - Improve driving decision
  - Simulations