Master
Reinforcement Learning 2022
Lecture B:
A Summary of Machine Learning

Aske Plaat

What is Intelligence?

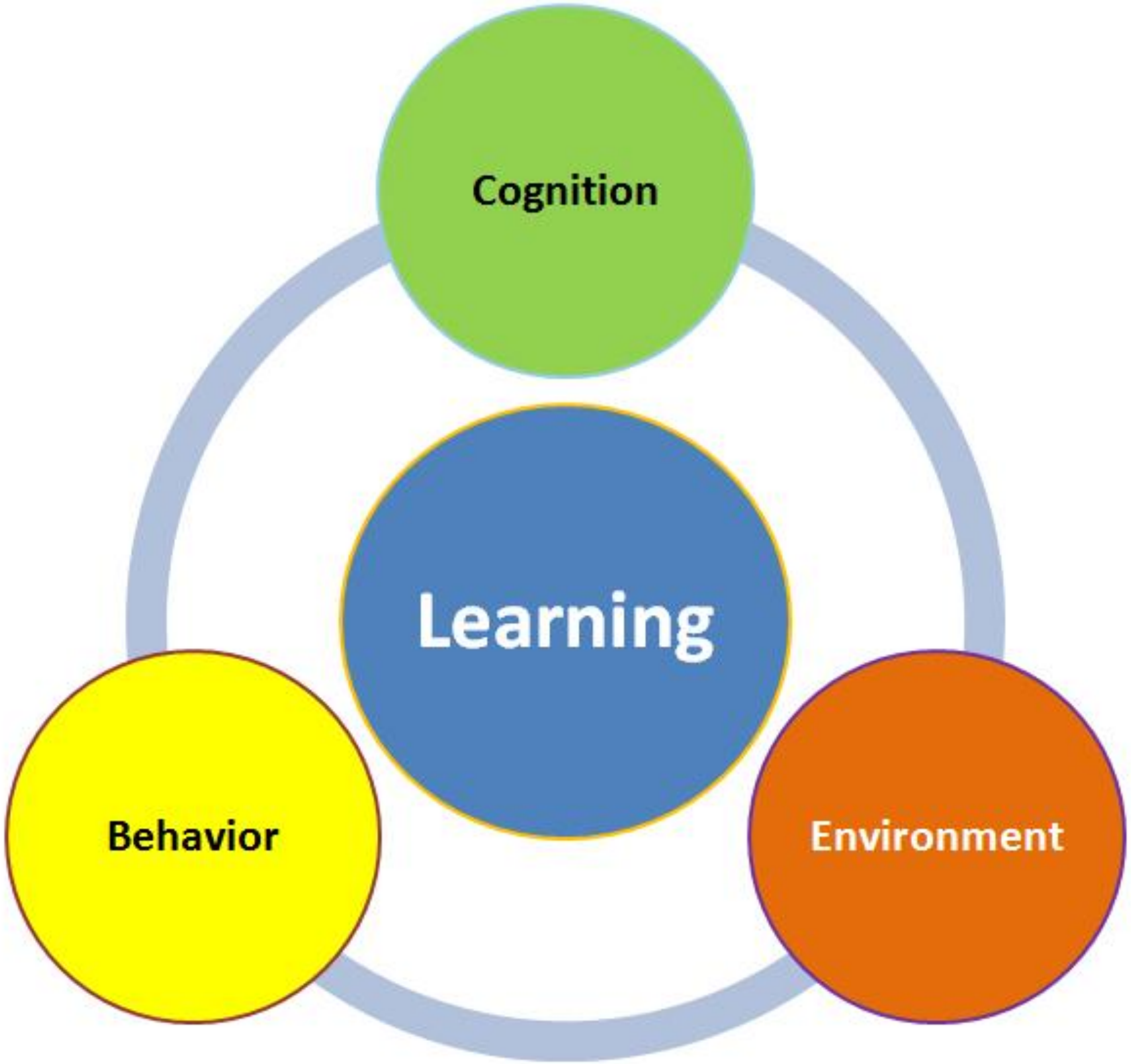
Cognition

Behavior

- Recogniti
- Memory
- Logical Reasoning
- Learning
- Adaptive Behavior

- Creativity
- Intuition
- Free Will
- Self Awareness
- Consciousness

What is Learning?



A very short summary of Machine Learning and Deep Learning

Traditional Programming

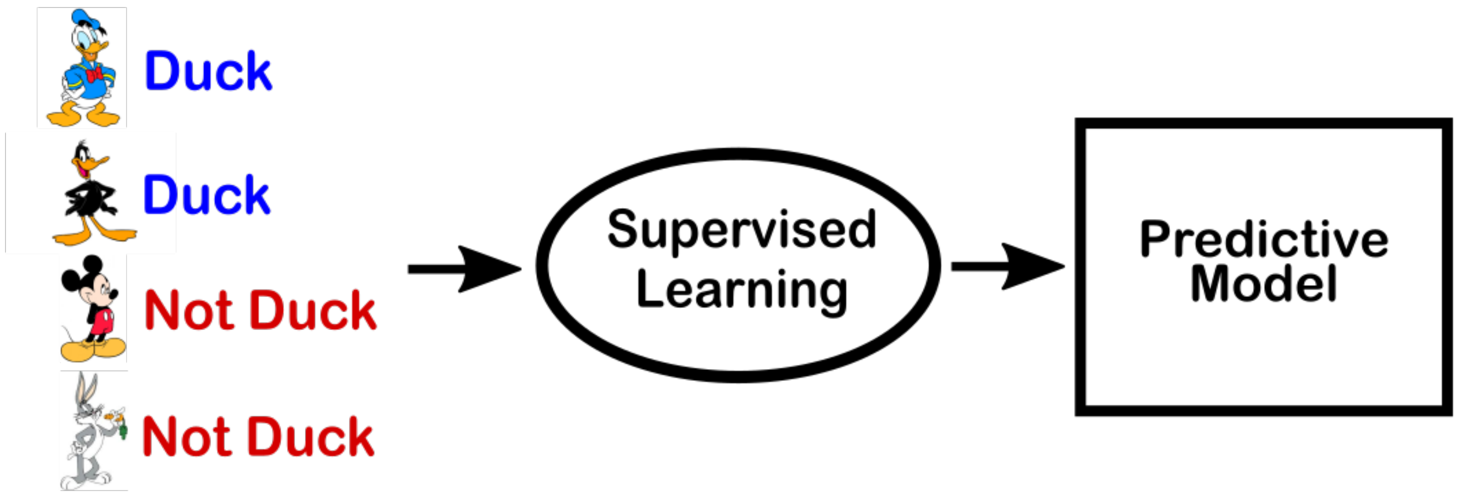
Machine Learning

Machine Learning

- Supervised Learning - learning from example & label
- Reinforcement Learning - learning from interaction number
- Unsupervised Learning - learning inherent relations

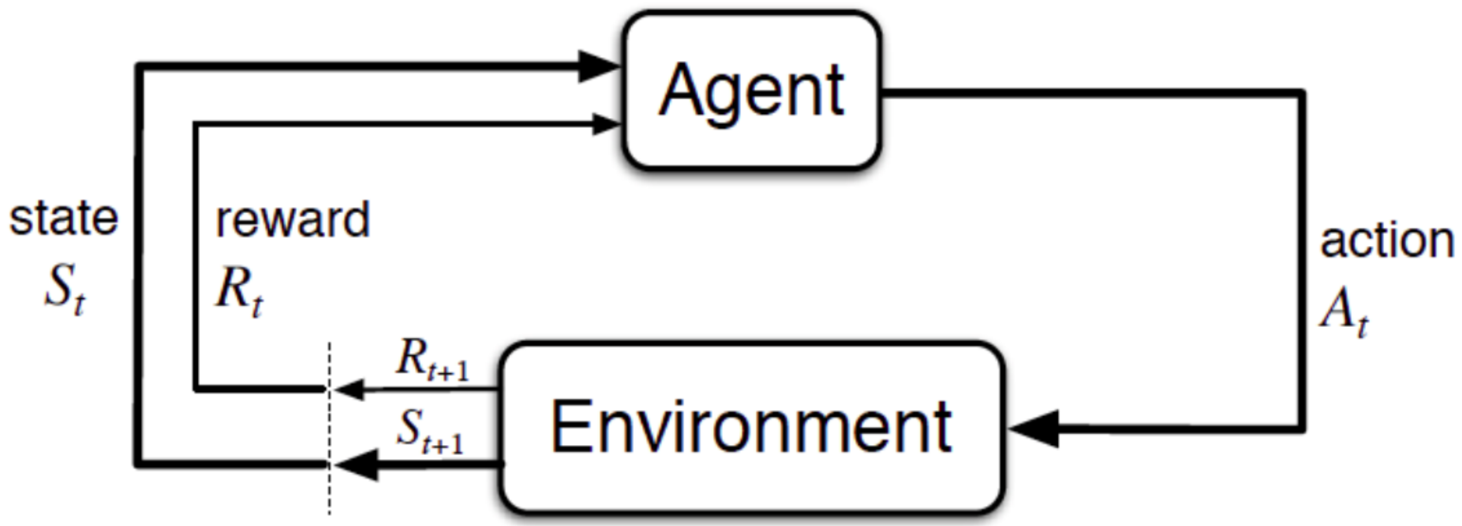
Supervised Learning

- Is learning by example
- Database learning
- (Image, label) -> correct?



Reinforcement Learning

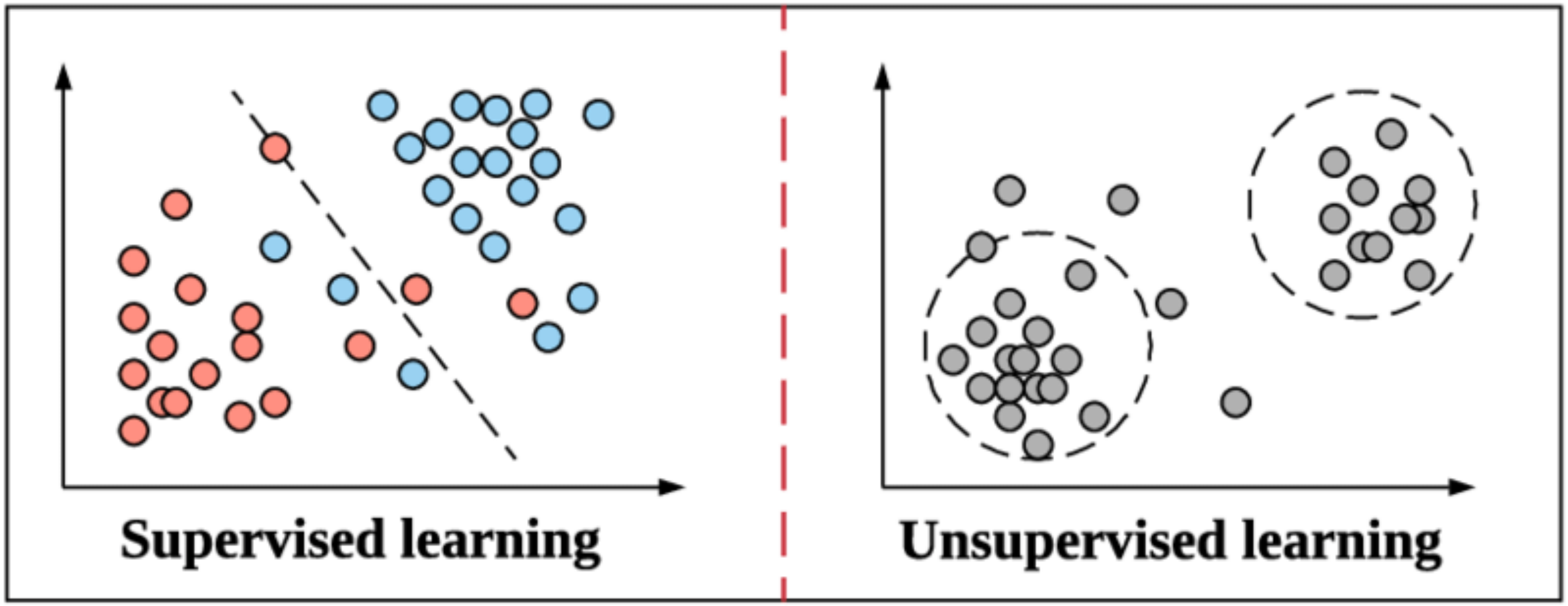
- is learning by interaction
- (state, action) -> reward number

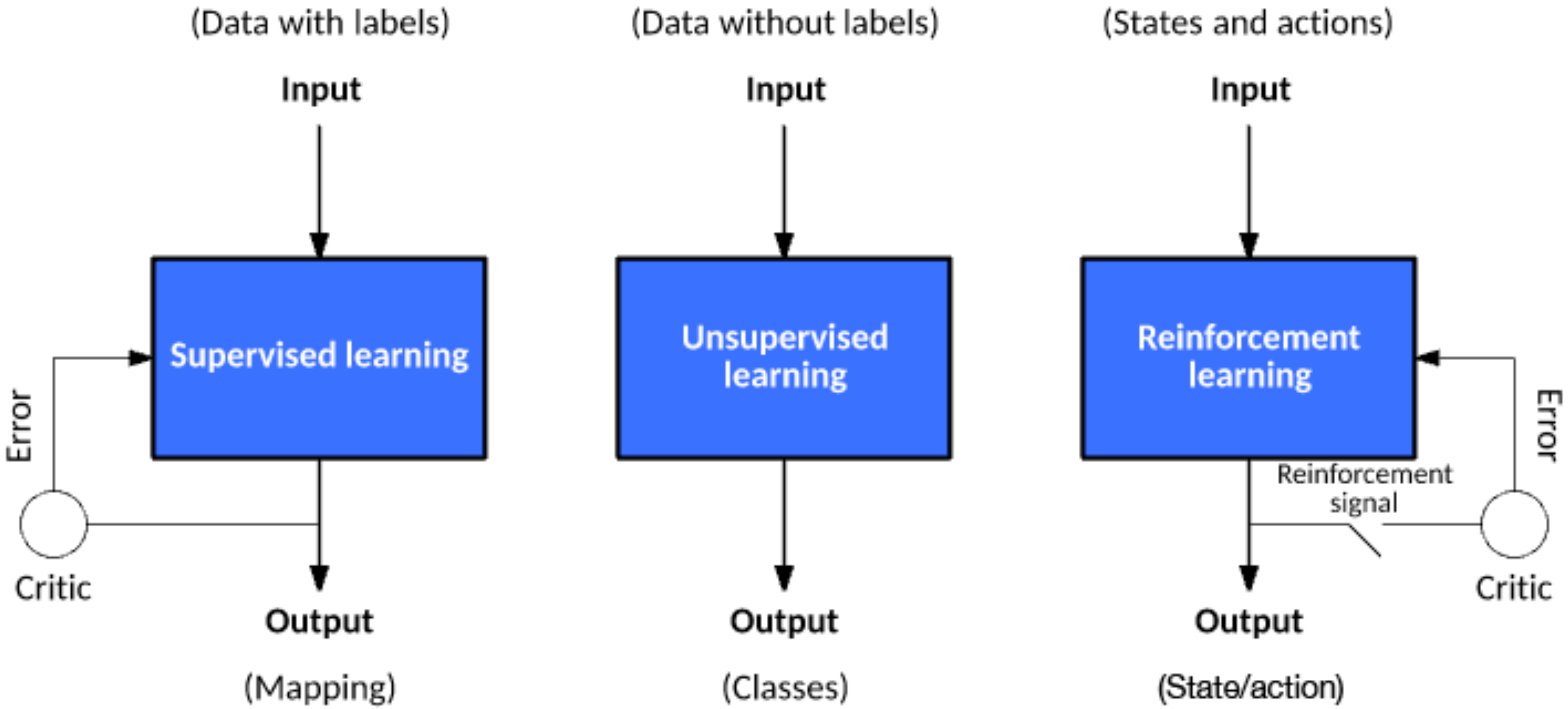


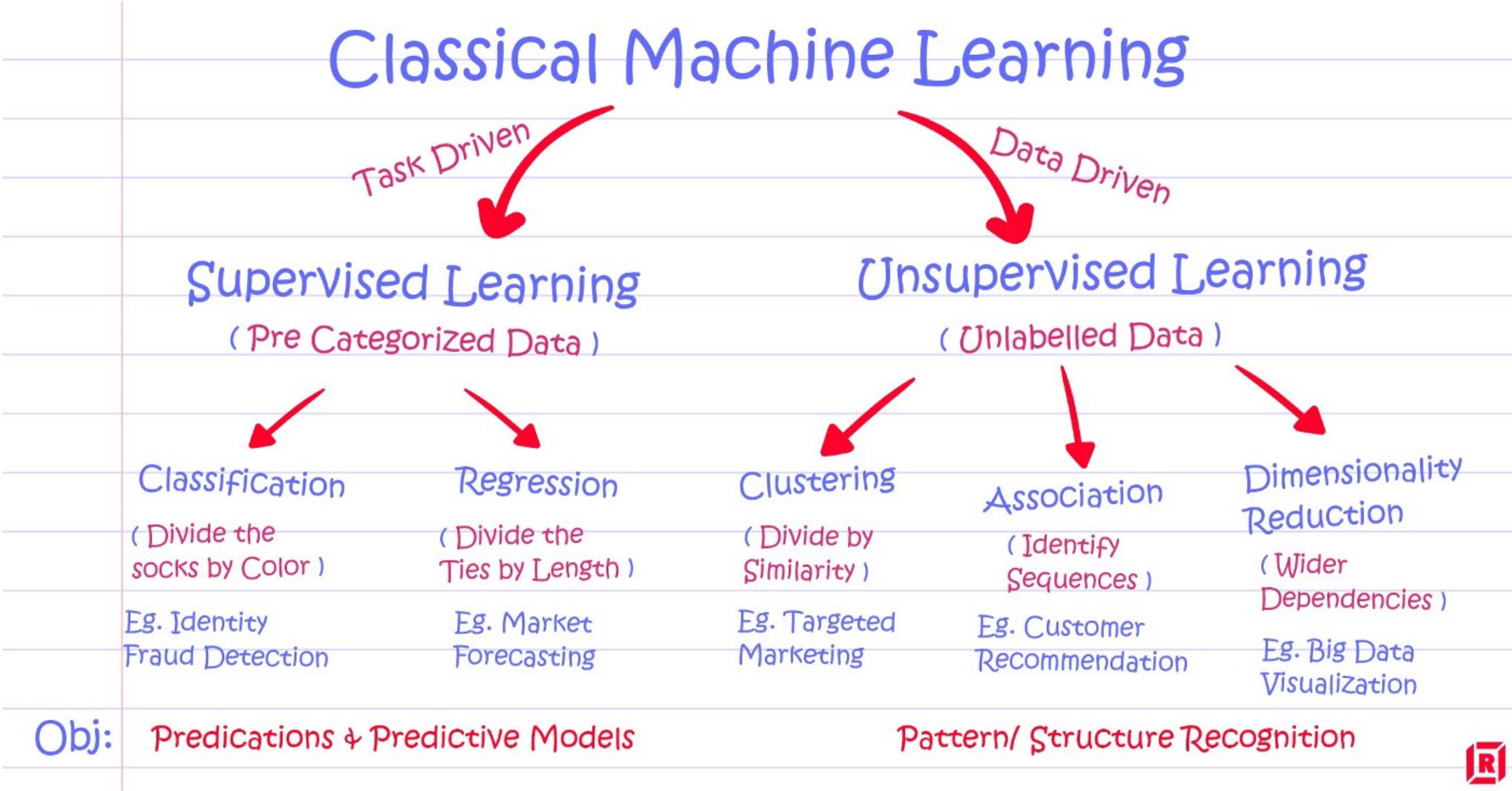
The agent-environment interaction in reinforcement learning. (Source: Sutton and Barto, 2017)

Unsupervised Learning

- Is learning without examples, from inherent measures
- Database learning
- Clustering, data compression, dimensionality reduction




(Data with labels)
Input
Supervised learning
Error
Critic
Output
(Mapping)
(Data without labels)
Input
Unsupervised learning
Output
(Classes)
(States and actions)
Input
Reinforcement learning
Error
Reinforcement signal
Critic
Output
(State/action)



Classical Machine Learning
Task Driven
Data Driven
Supervised Learning
(Pre Categorized Data)
Unsupervised Learning
(Unlabelled Data)
Classification
(Divide the socks by Color)
Eg. Identity Fraud Detection
Regression
(Divide the Ties by Length)
Eg. Market Forecasting
Clustering
(Divide by Similarity)
Eg. Targeted Marketing
Association
(Identify Sequences)
Eg. Customer Recommendation
Dimensionality Reduction
(Wider Dependencies)
Eg. Big Data Visualization
Obj:
Predications & Predictive Models
Pattern/ Structure Recognition

Machine learning models cheat sheet

Supervised learning

Data scientists provide input, output and feedback to build model (as the definition)

EXAMPLE ALGORITHMS:

Linear regressions
- sales forecasting
- risk assessment

Support vector machines
- image classification
- financial performance comparison

Decision tree
- predictive analytics
- pricing

Unsupervised learning

Use deep learning to arrive at conclusions and patterns through unlabeled training data.

EXAMPLE ALGORITHMS:

Apriori
- sales functions
- word associations
- searcher

K-means clustering
- performance monitoring
- searcher intent

Semi-supervised learning

Builds a model through a mix of labeled and unlabeled data, a set of categories, suggestions and exampled labels.

EXAMPLE ALGORITHMS:

Generative adversarial networks
- audio and video manipulation
- data creation

Self-trained Naïve Bayes classifier
- natural language processing

Reinforcement learning

Self-interpreting but based on a system of rewards and punishments learned through trial and error, seeking maximum reward.

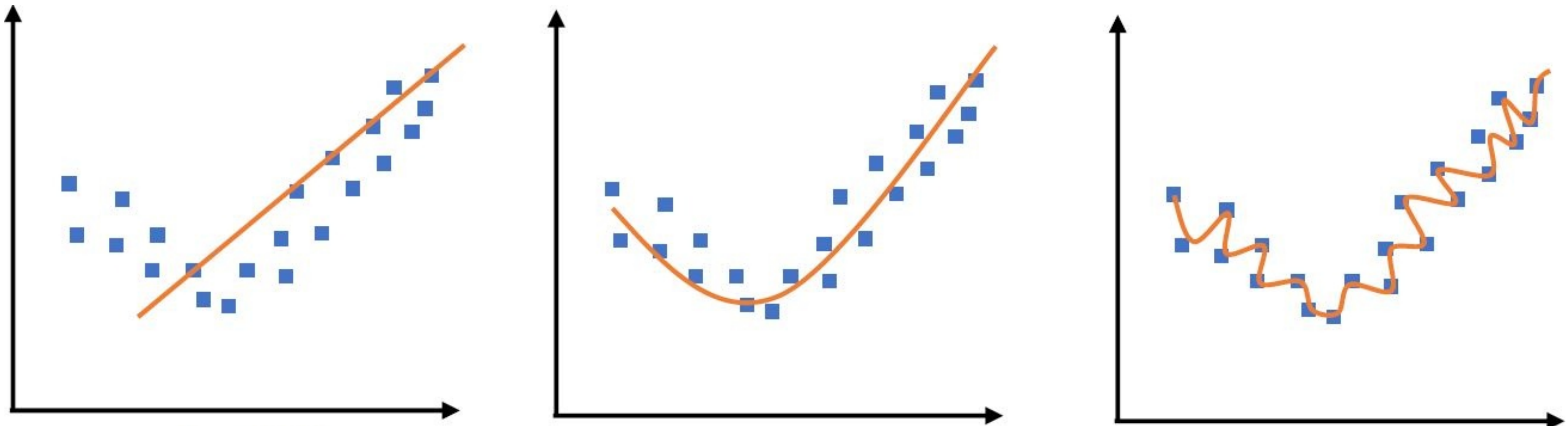
EXAMPLE ALGORITHMS:

Q-learning
- policy creation
- consumption reduction

Model-based value estimation
- linear tasks
- estimating parameters

Machine Learning
Learning a Function

- Fit a function to data (x, y) items
- Iterative minimization process of error between learned function f and data labels
- regression loss: L=avg(y-f)^2
- categorization loss: L=-sum (y log f)



Accuracy

- True Positives = # classified correctly as in class
- False Positives = # classified incorrectly as in class
- True Negatives = # classified correctly as not in class
- False Negatives = # classified incorrectly as not in class
- Accuracy = Trues/All = (TP+TN) / (TP+TN+FP+FN)
  How well correct elements are predicted

Confusion Matrix

		Ground truth		
		+	-	
Predicted	+	True positive (TP)	False positive (FP)	Precision = TP / (TP + FP)
	-	False negative (FN)	True negative (TN)	
		Recall = TP / (TP + FN)		Accuracy = (TP + TN) / (TP + FP + TN + FN)

Function Approximation

- Small or Large data set?
- Exact Memoization: Small
- Feature Learning/Generalization: Large

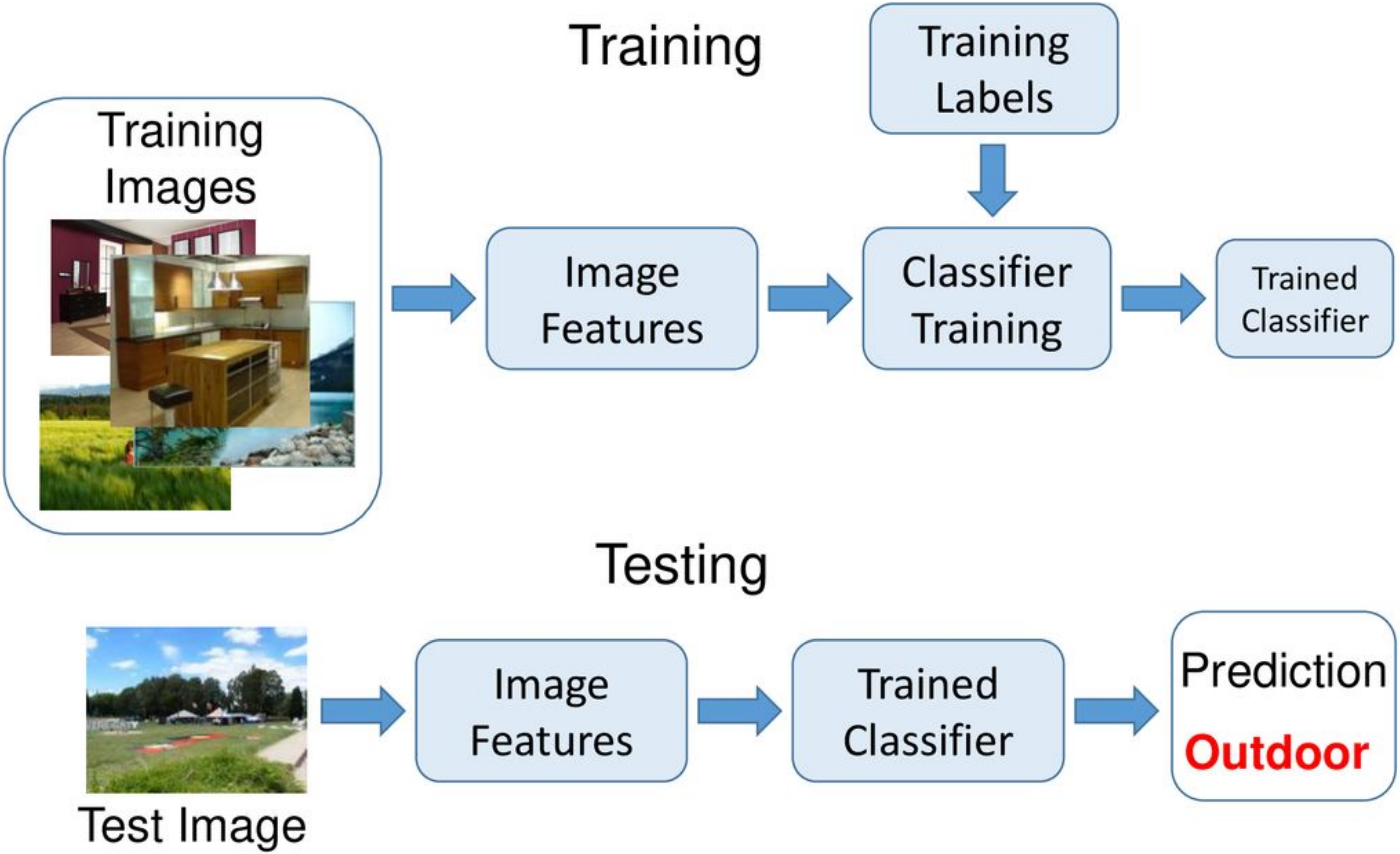
What is a <u>generalization</u>?

- A generalization is a broad statement about a group of people, things, or ideas.
- It states something they have in common.

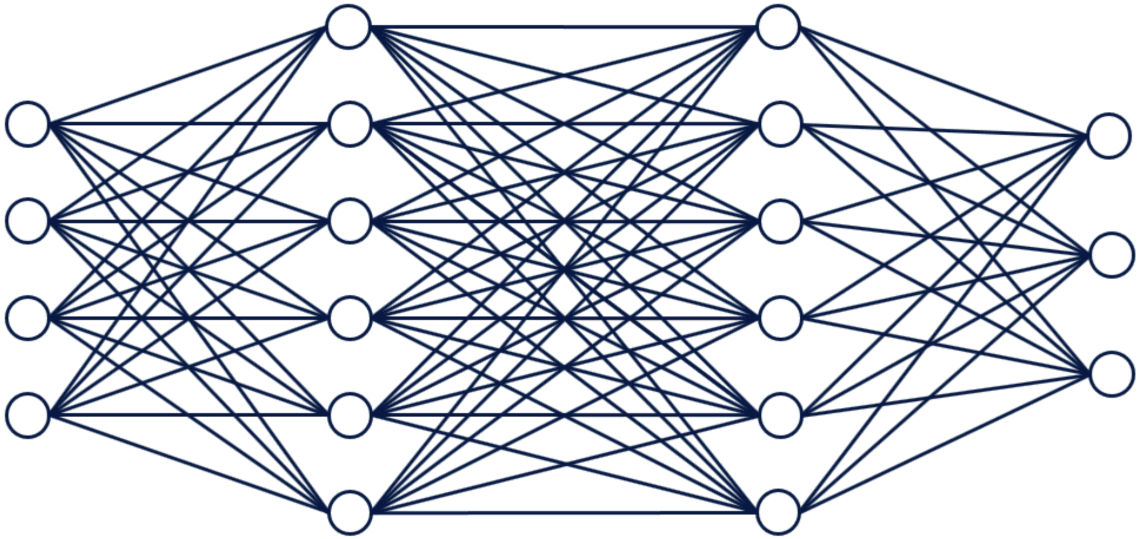
Generalization

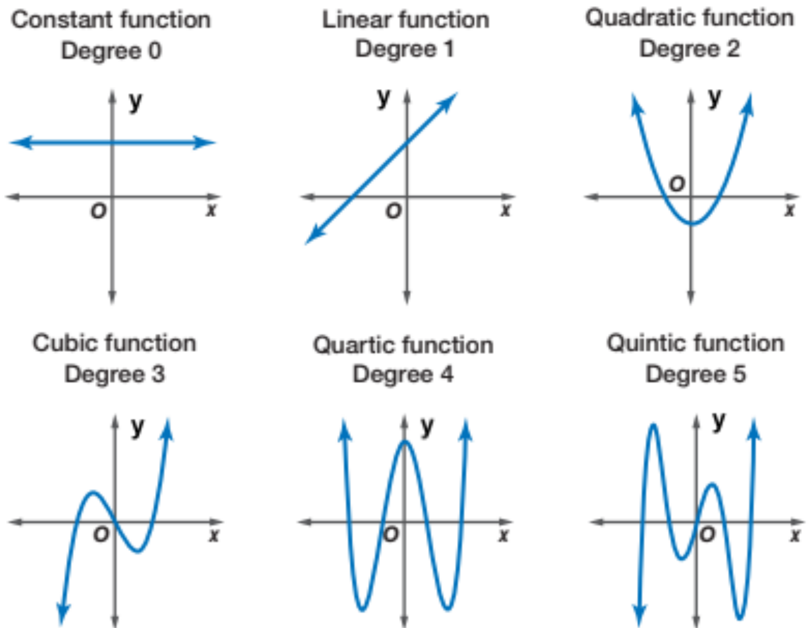
- Split data set D into Training set and Test set
- Train on Training set
- Test on Test set
- An algorithm generalizes well from training to test if the accuracy at test time is about as good as at training time

Image Categorization

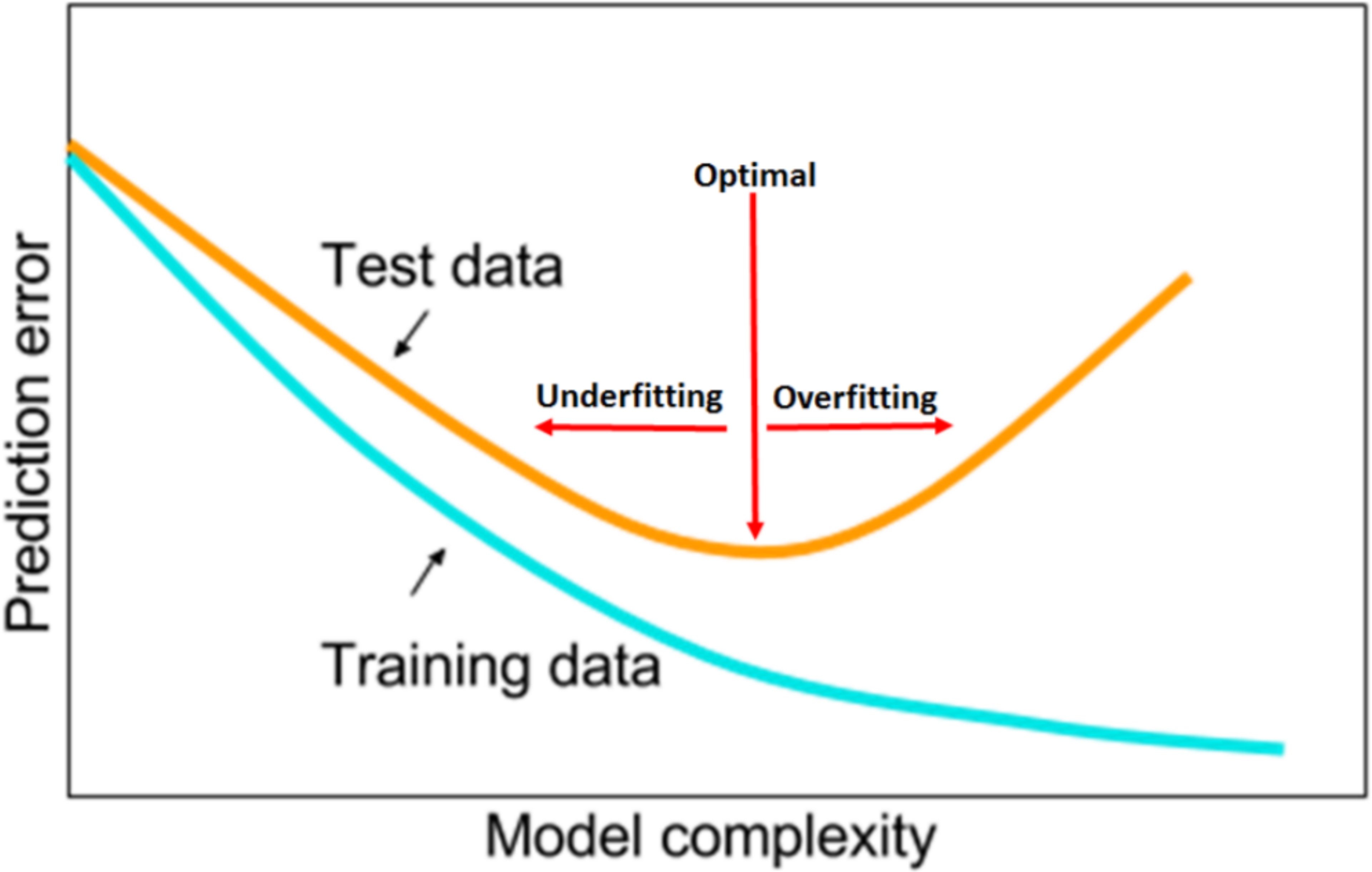


Model Complexity

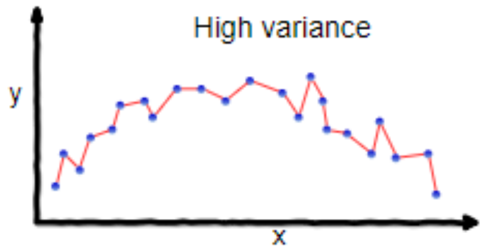




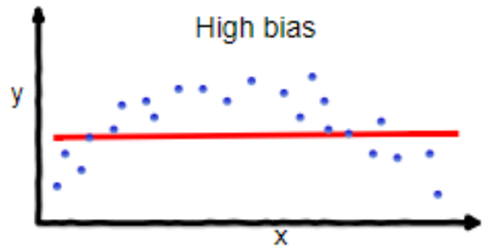
Capacity


Optimal
Test data
Underfitting
Overfitting
Training data
Prediction error
Model complexity

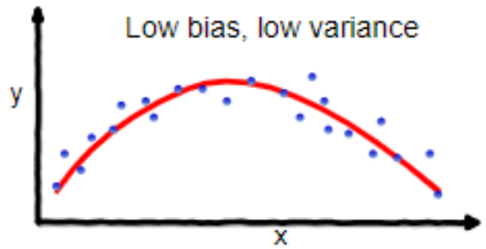
Bias-Variance



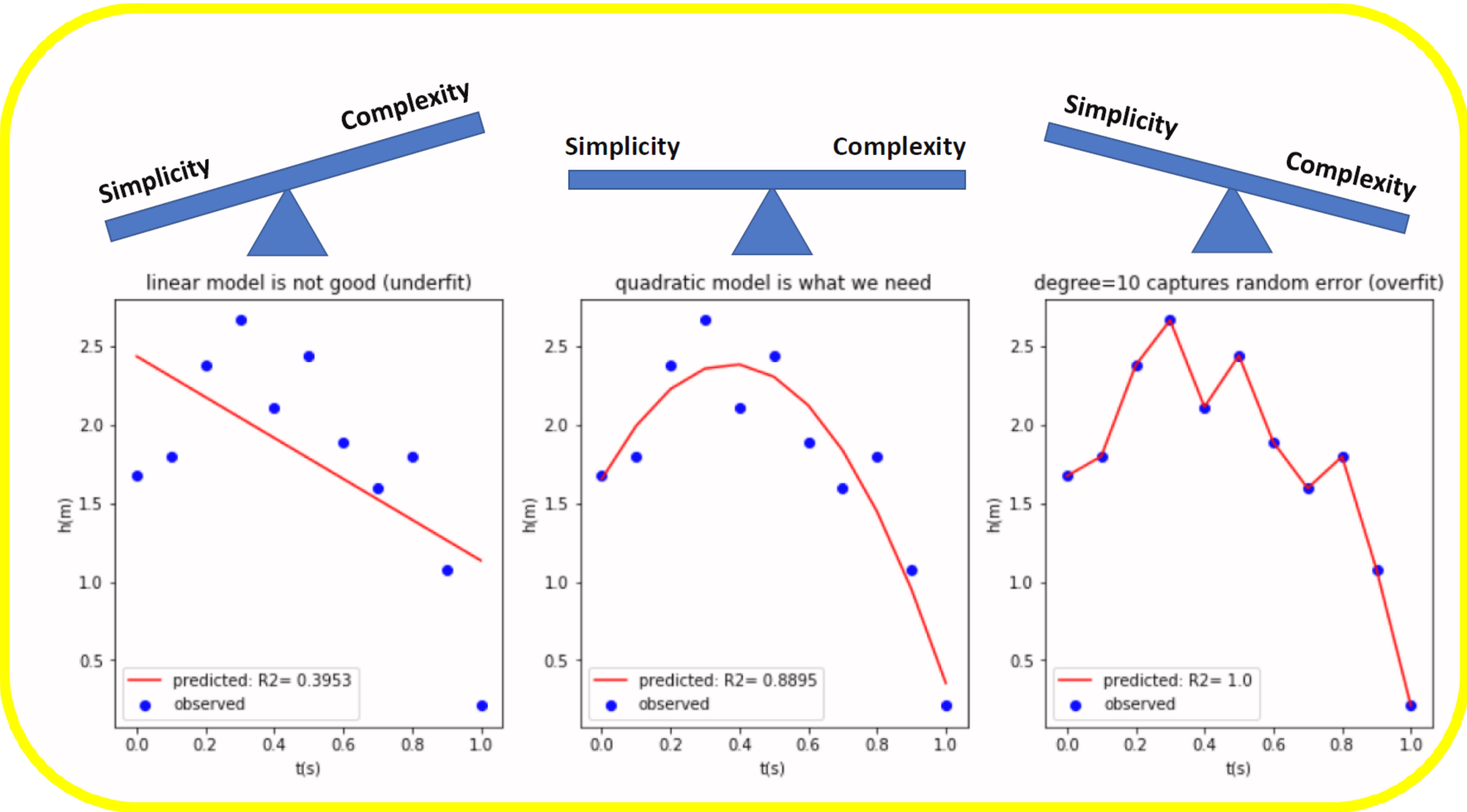
overfitting



underfitting



Good balance



Simplicity
Complexity
Simplicity
Complexity
Simplicity
Complexity
linear model is not good (underfit)
quadratic model is what we need
degree=10 captures random error (overfit)
h(m)
t(s)
predicted: R2= 0.3953
observed
predicted: R2= 0.8895
observed
predicted: R2= 1.0
observed

Regularization

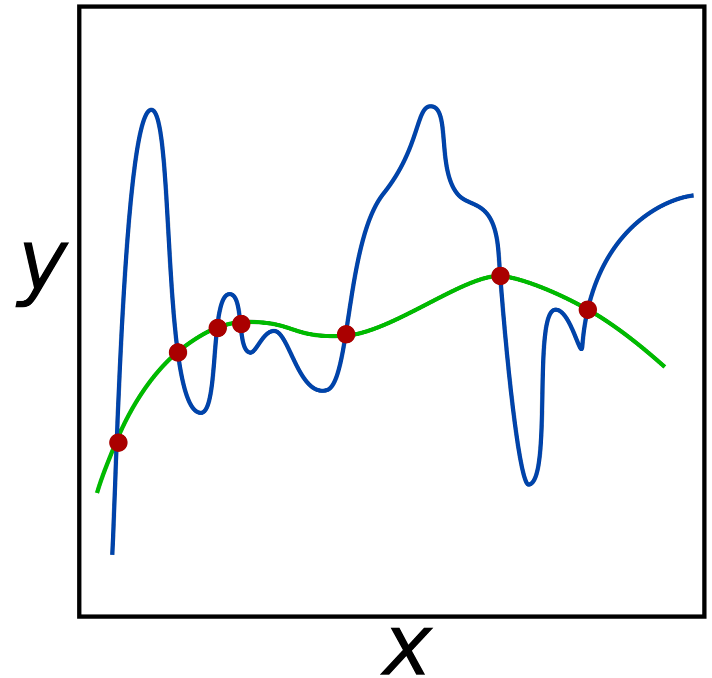
- The world is smooth
- Penalize large weights

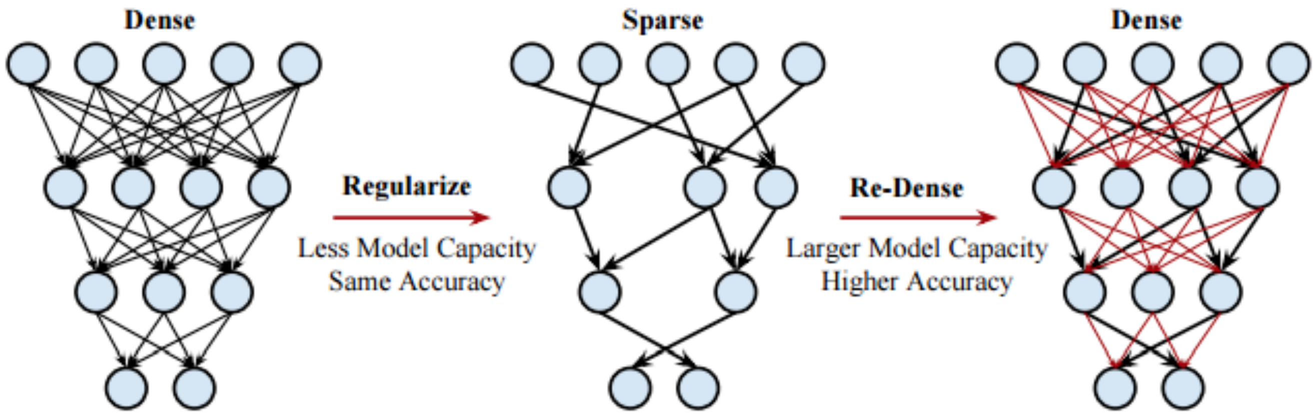
L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

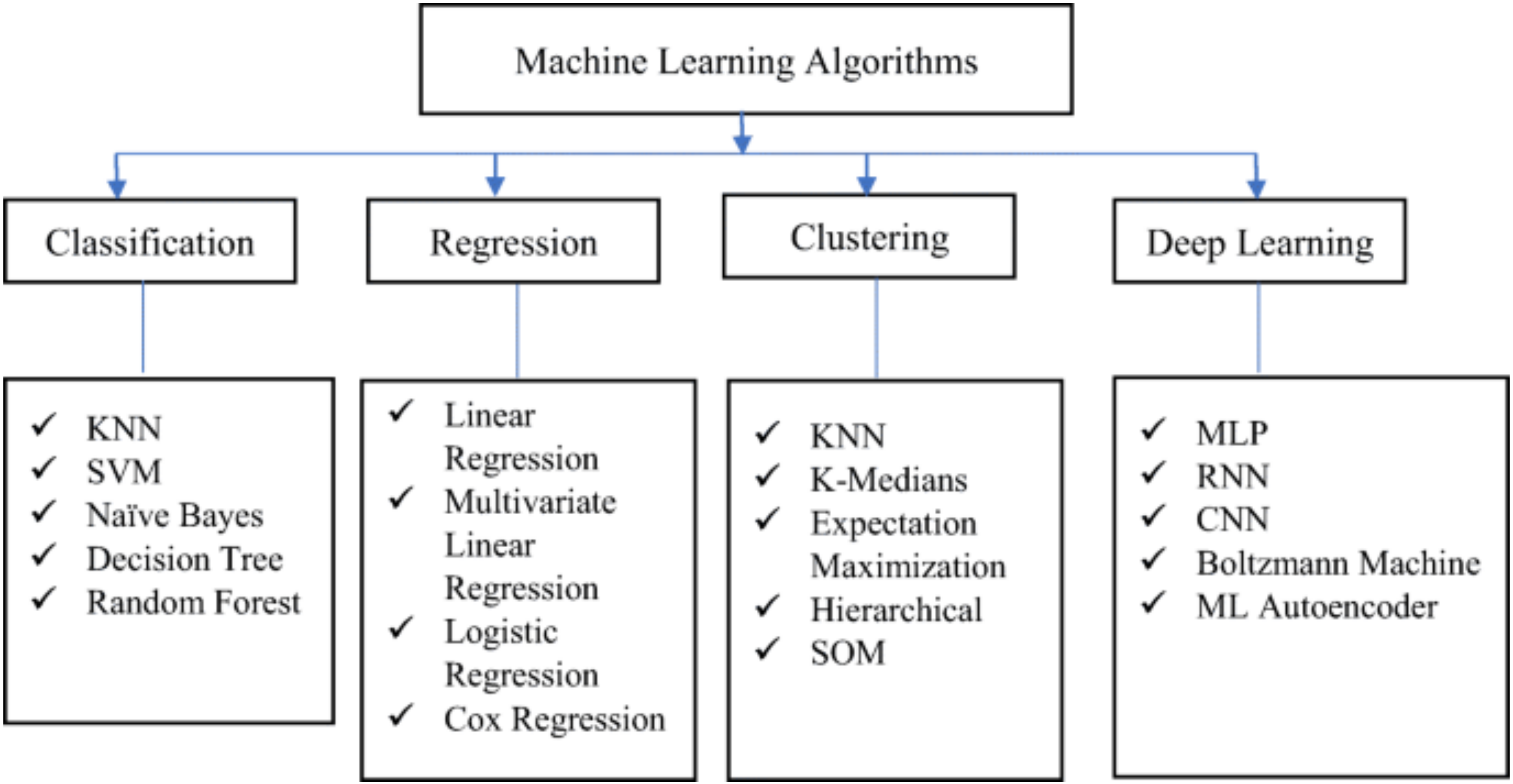
L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$



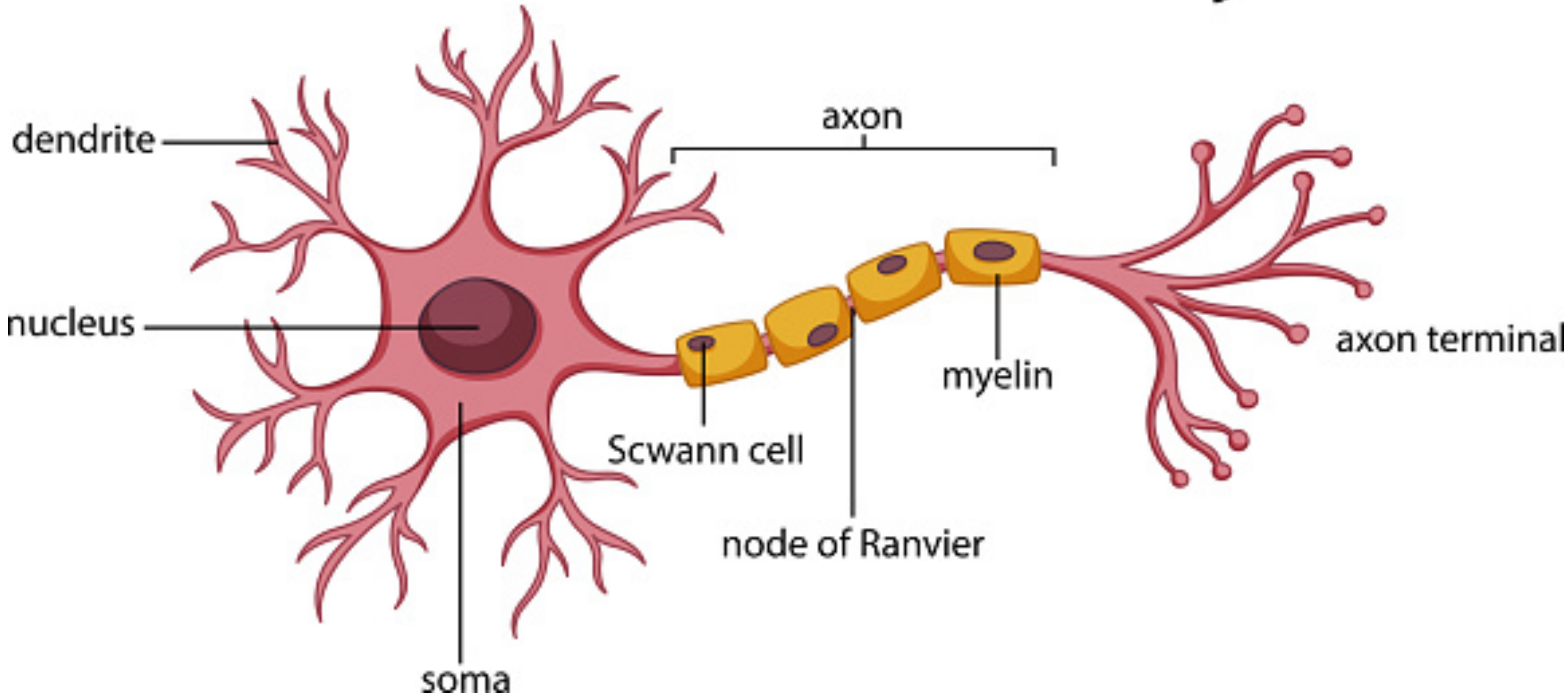


Neural Networks

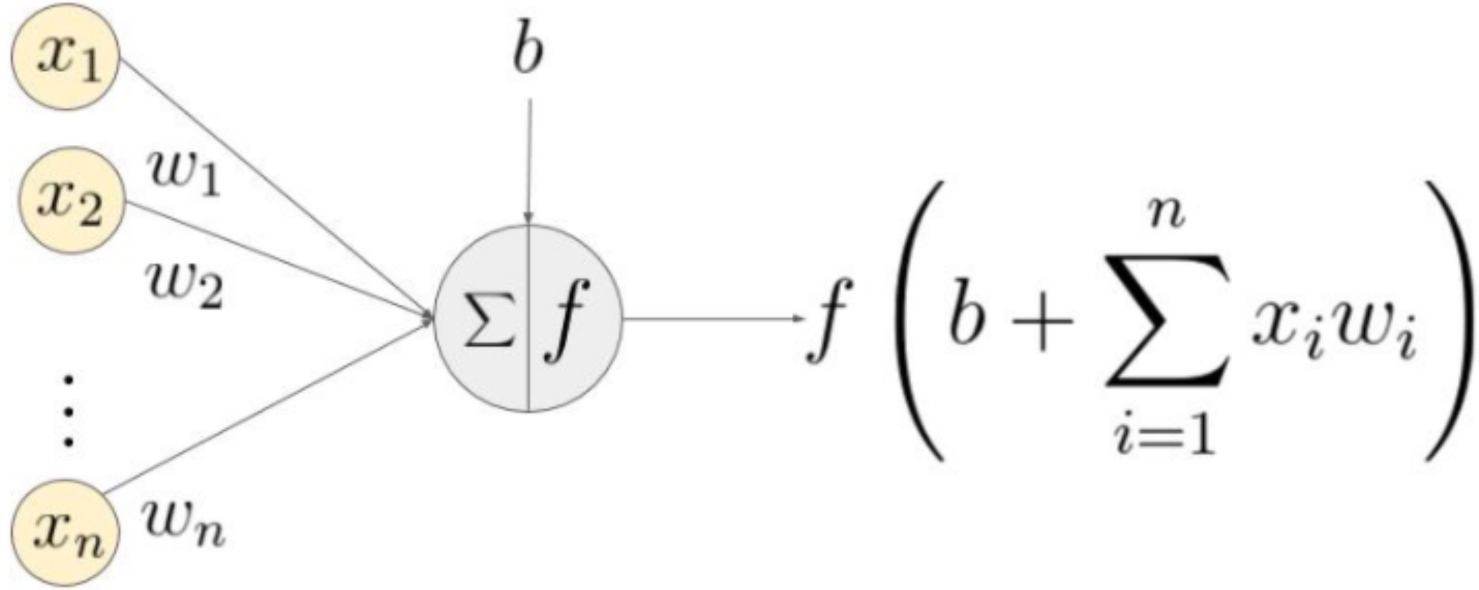


Machine Learning Algorithms
Classification
Regression
Clustering
Deep Learning
✓ KNN
✓ SVM
✓ Naïve Bayes
✓ Decision Tree
✓ Random Forest
✓ Linear Regression
✓ Multivariate Linear Regression
✓ Logistic Regression
✓ Cox Regression
✓ KNN
✓ K-Medians
✓ Expectation Maximization
✓ Hierarchical
✓ SOM
✓ MLP
✓ RNN
✓ CNN
✓ Boltzmann Machine
✓ ML Autoencoder

Neuron Anatomy

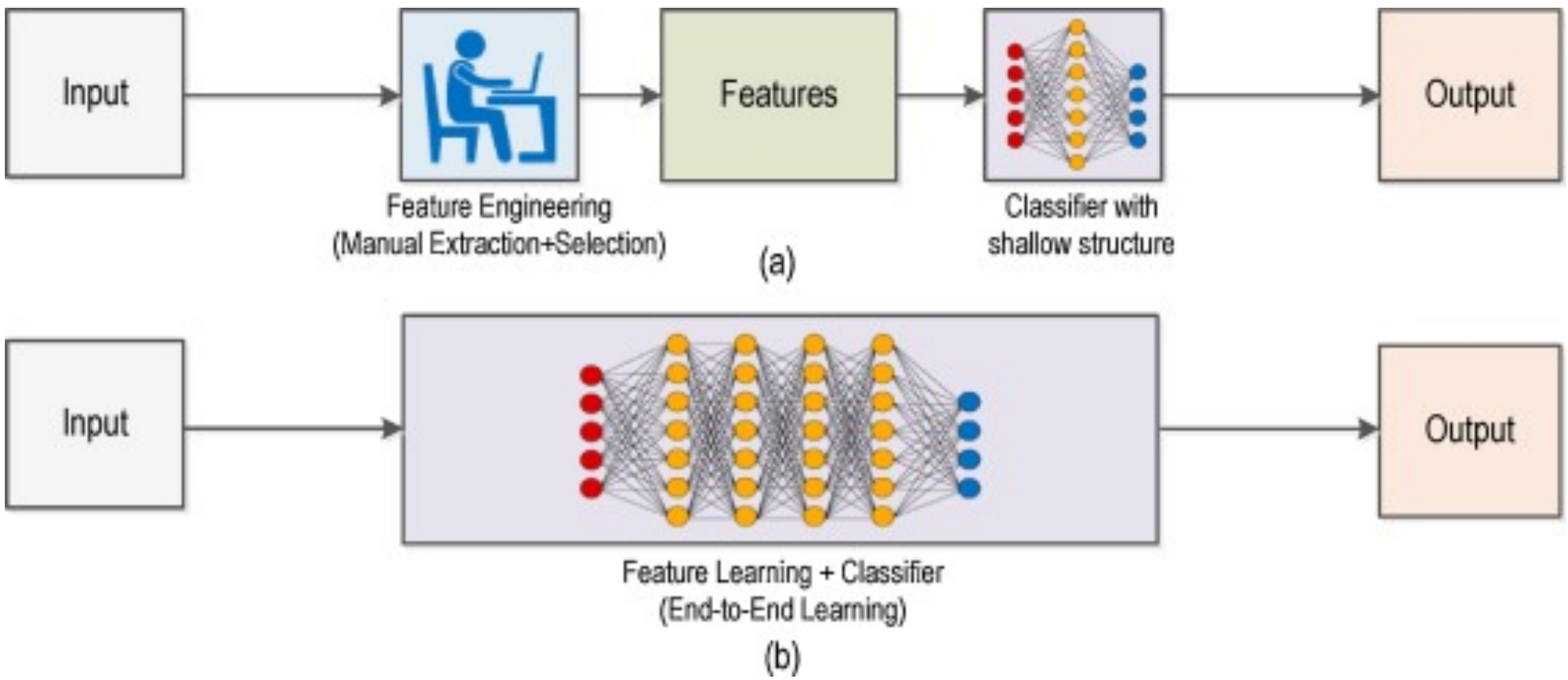


Artificial Neuron

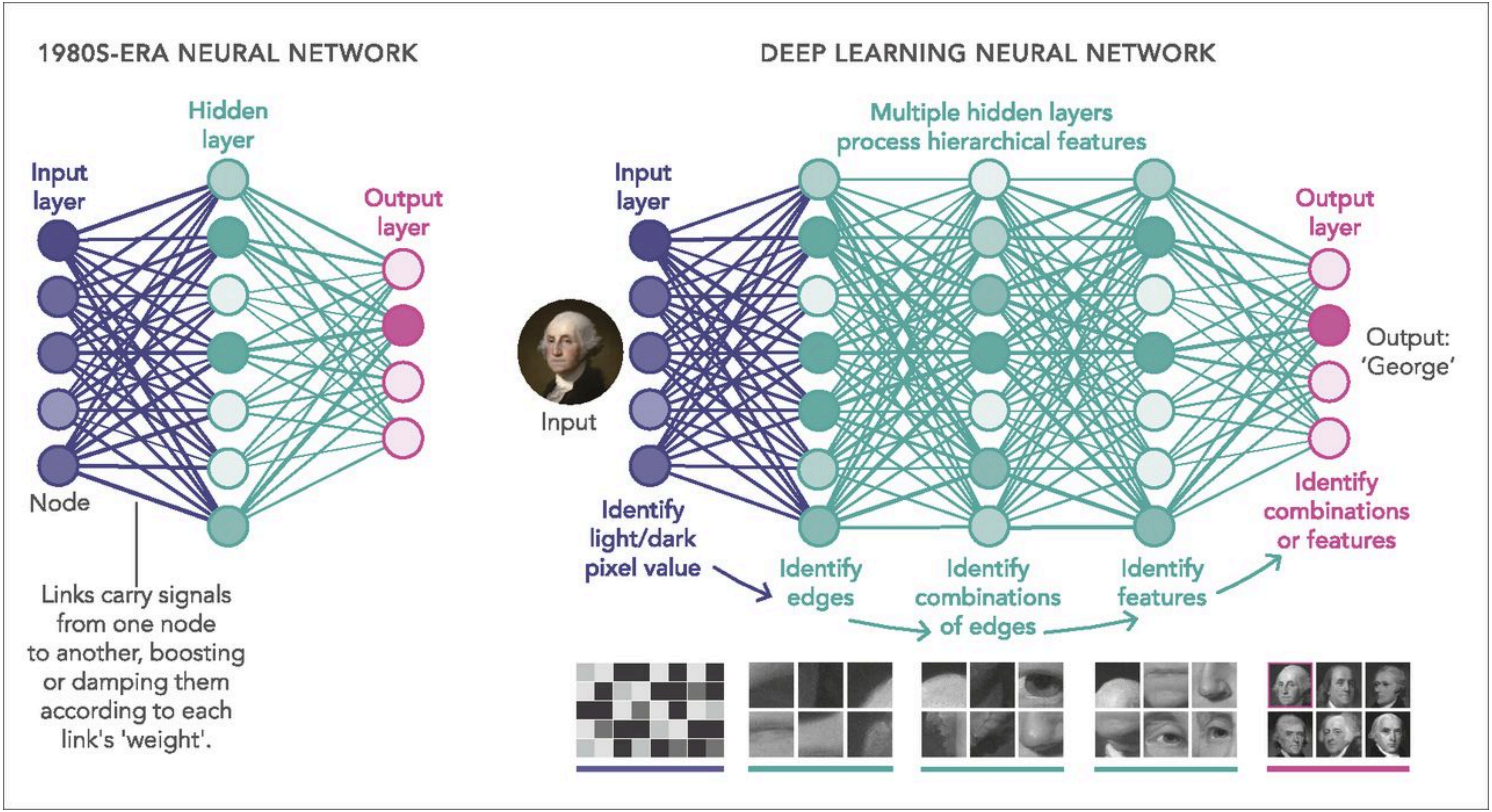


An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

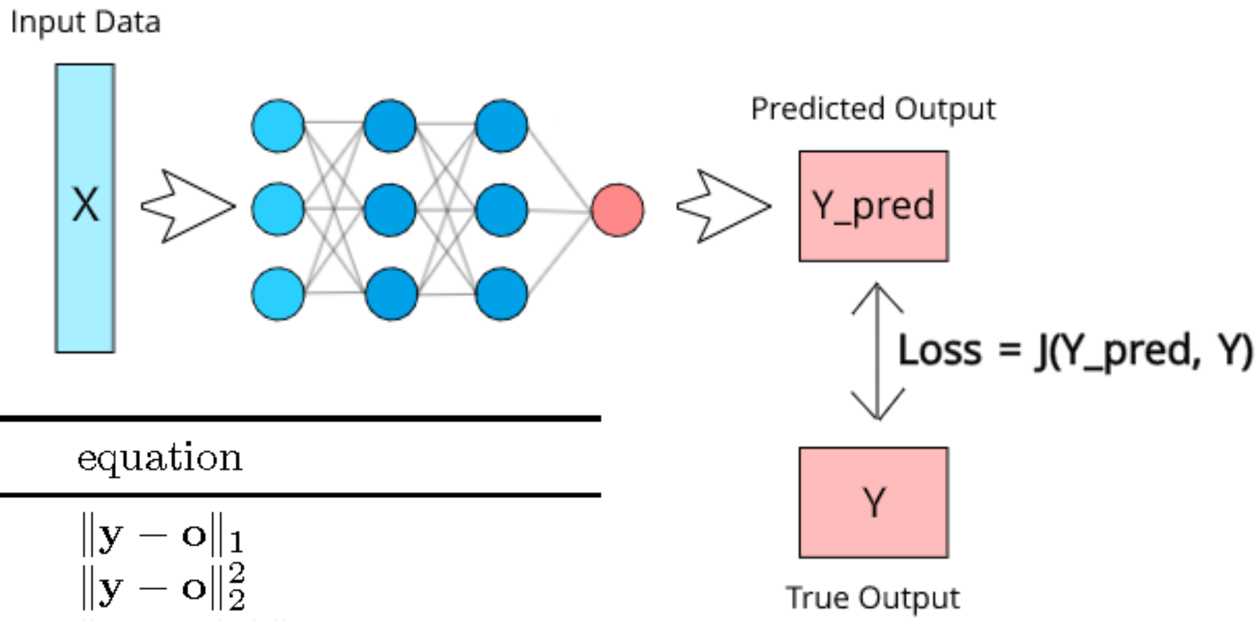
End-to-end Learning



Deep Learning



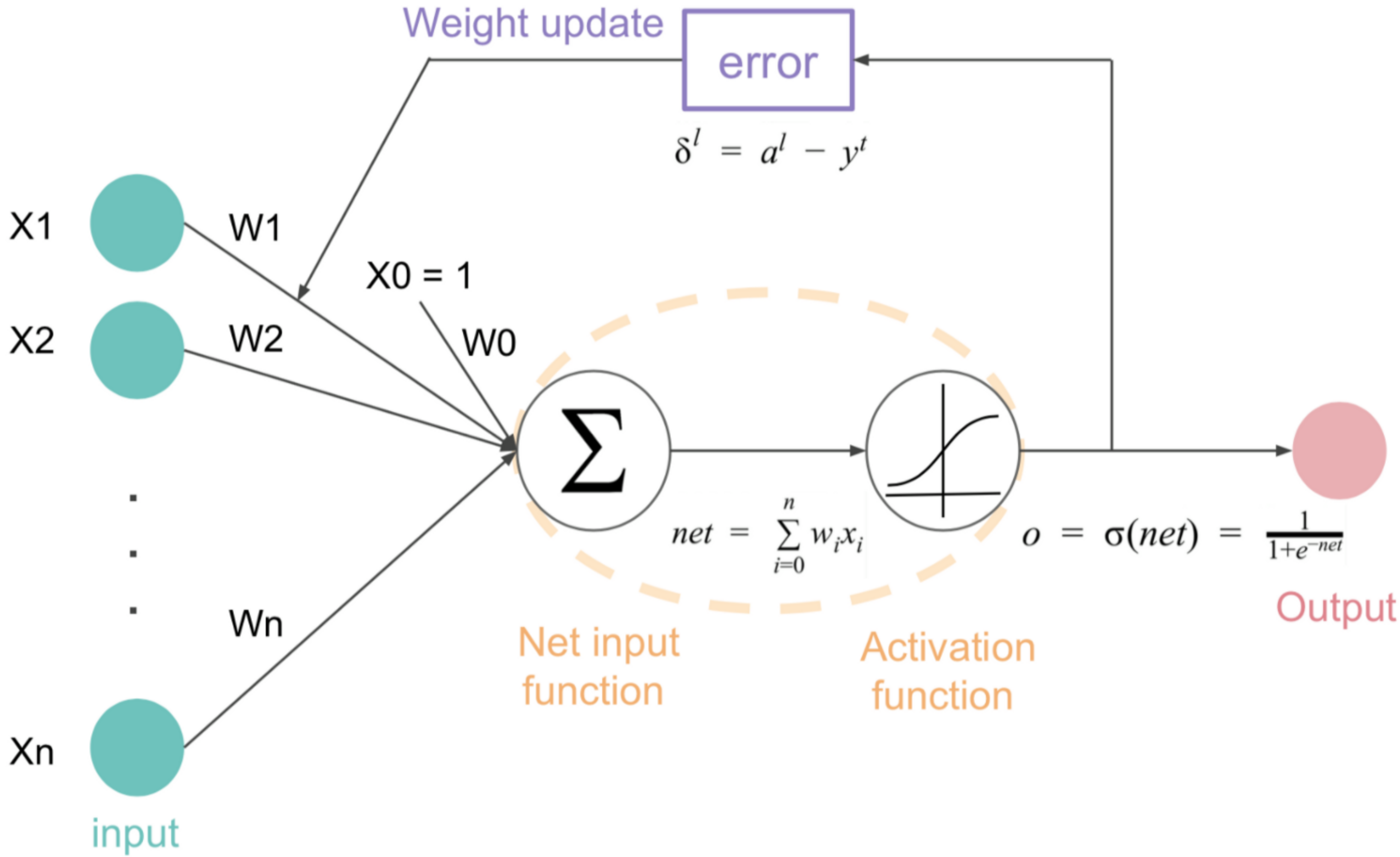
Loss



imate.

symbol	name	equation
$\mathcal{L}_1$	L_1 loss	$\|\mathbf{y}-\mathbf{o}\|_1$
$\mathcal{L}_2$	L_2 loss	$\|\mathbf{y}-\mathbf{o}\|_2^2$
$\mathcal{L}_1 \circ \sigma$	expectation loss	$\|\mathbf{y}-\sigma(\mathbf{o})\|_1$
$\mathcal{L}_2 \circ \sigma$	regularised expectation loss[1]	$\|\mathbf{y}-\sigma(\mathbf{o})\|_2^2$
$\mathcal{L}_\infty \circ \sigma$	Chebyshev loss	$\max_j \lvert\sigma(\mathbf{o})^{(j)}-\mathbf{y}^{(j)}\rvert$
hinge	hinge [13] (margin) loss	$\sum_j \max(0, \frac{1}{2}-\hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})$
hinge^2	squared hinge (margin) loss	$\sum_j \max(0, \frac{1}{2}-\hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})^2$
hinge^3	cubed hinge (margin) loss	$\sum_j \max(0, \frac{1}{2}-\hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})^3$
log	log (cross entropy) loss	$-\sum_j \mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}$
$\log^2$	squared log loss	$-\sum_j [\mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}]^2$
tan	Tanimoto loss	$\frac{-\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}{\|\sigma(\mathbf{o})\|_2^2+\|\mathbf{y}\|_2^2-\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}$
D_{CS}	Cauchy-Schwarz Divergence [3]	$-\log \frac{\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}{\|\sigma(\mathbf{o})\|_2\|\mathbf{y}\|_2}$

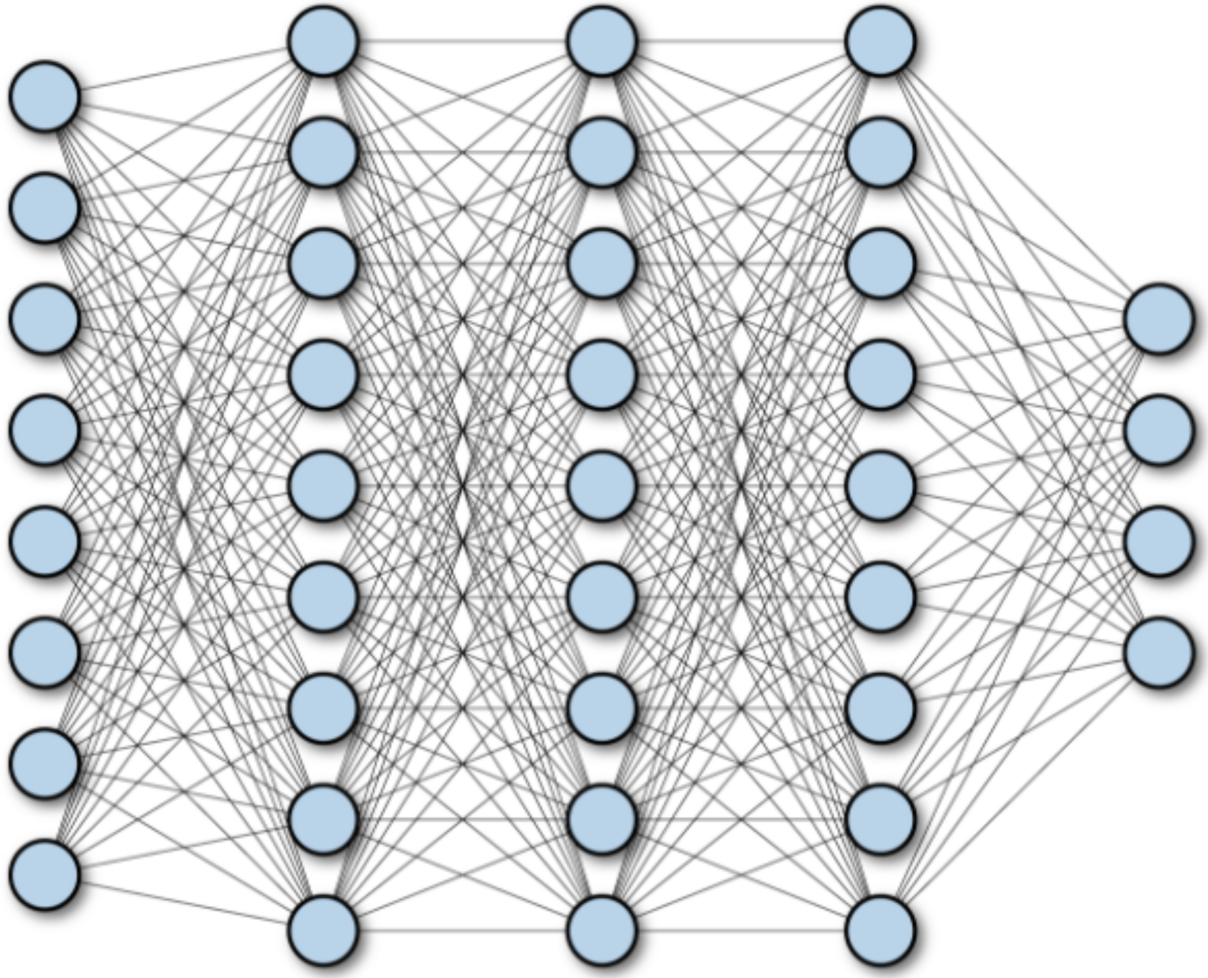
Backpropagation



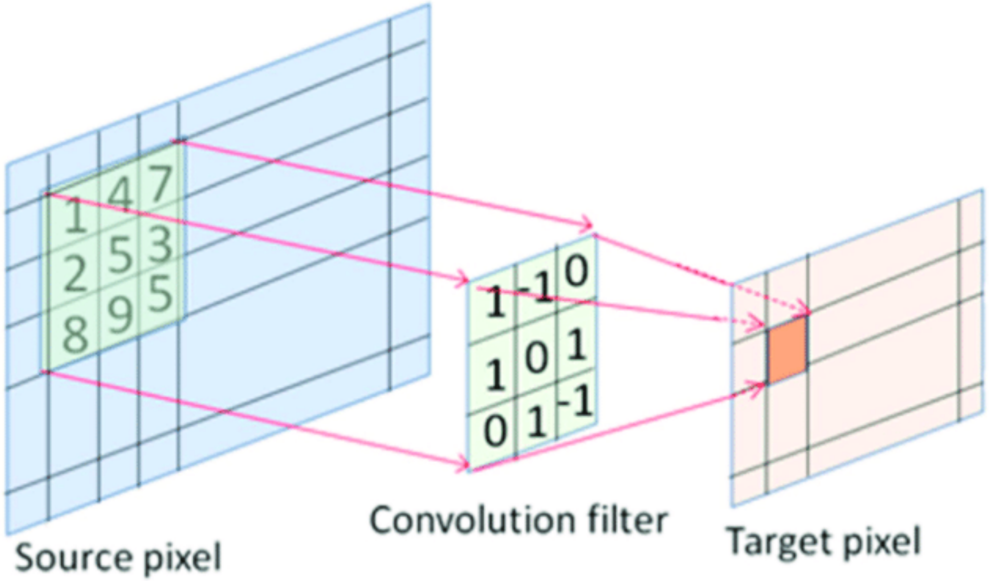
Network Architectures

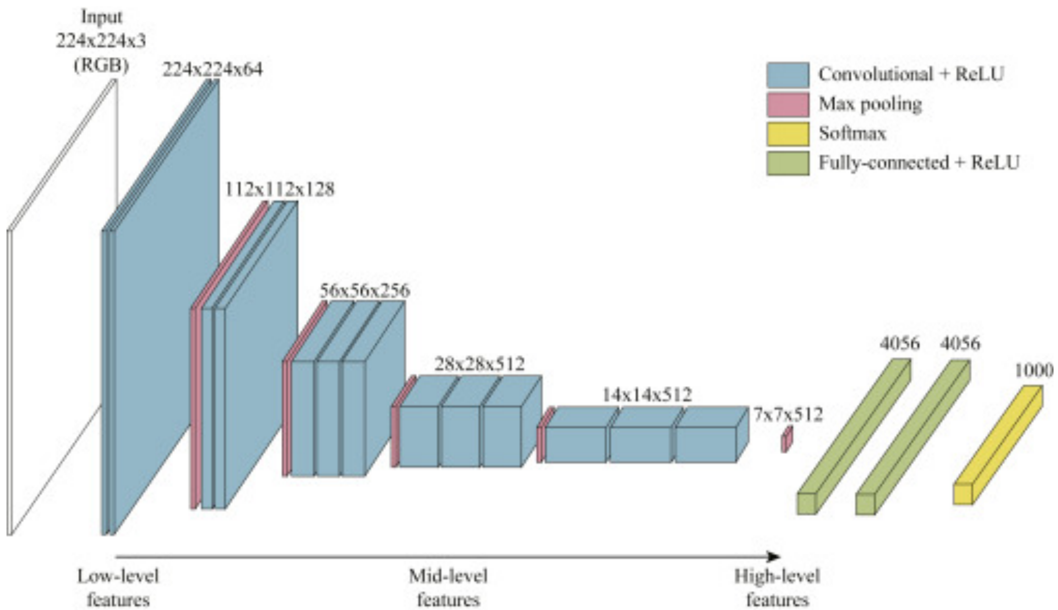
FCN

- Computational Complexity
- Overfitting



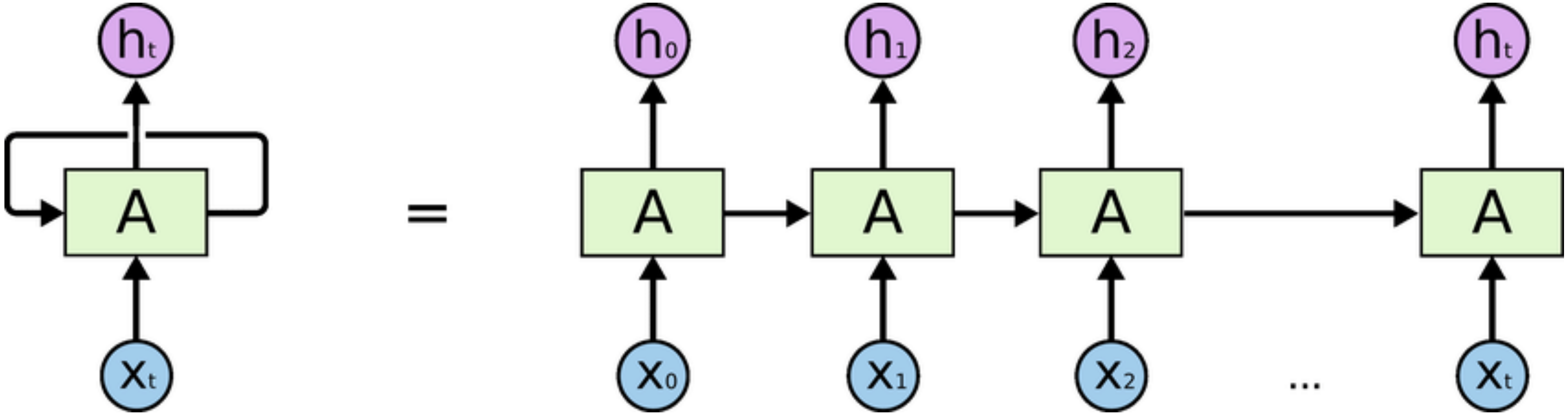
CNN - spatial





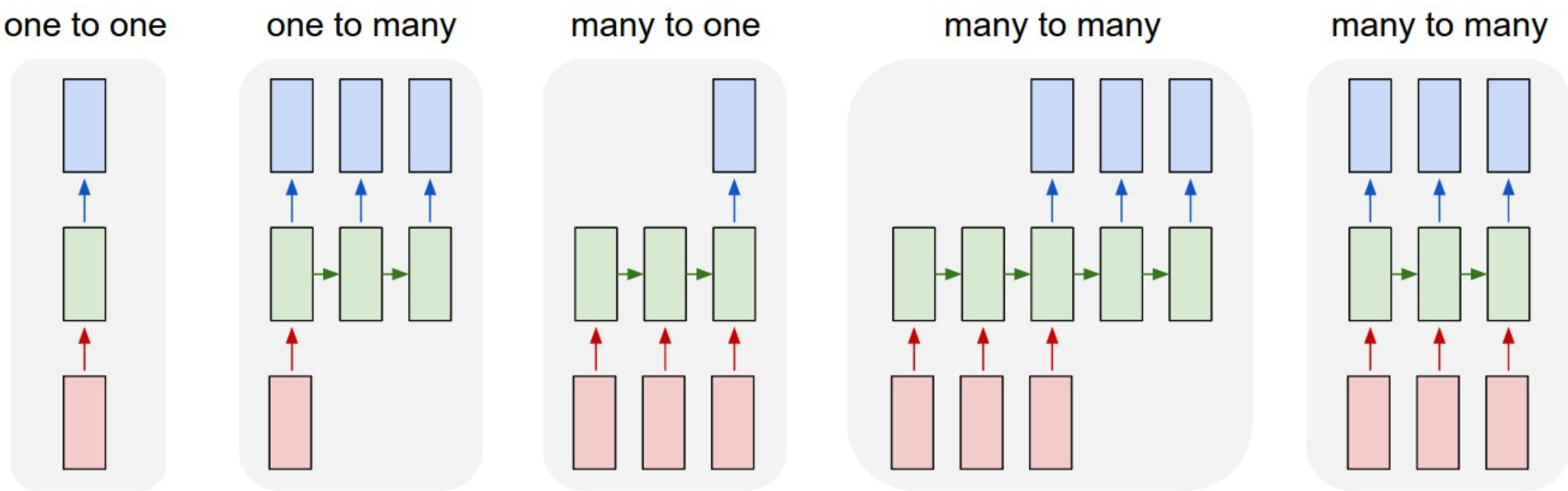
- spatial hierarchy
- sparsely connected layers (filters share weights)
- pooling layers reduce dimensionality even more
- reduce overfitting

RNN - sequential



- state
- unfolding in deep layered network
- vanishing gradients - ReLU

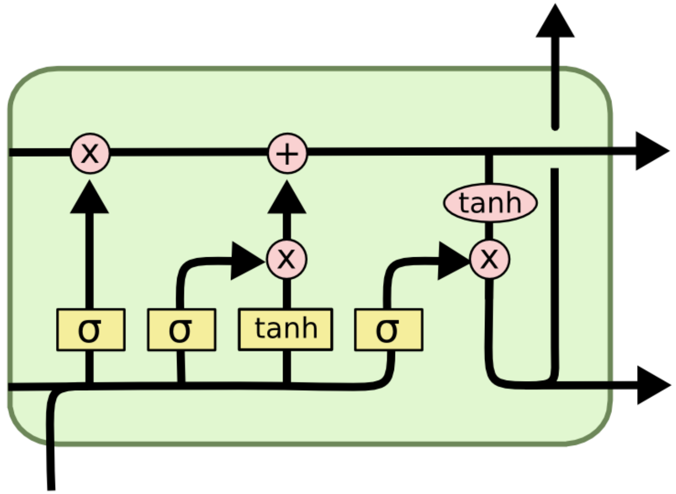
RNN - sequential



- 1-1: fixed to fixed: image classification
- 1-n. fixed to sequence. image to sentence of words
- n-1. sequence to fixed. sentiment analysis (sentence to class)
- n-n. sequence to sequence. machine translation (sentence to sentence)
- n-n synchronized. video classification

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

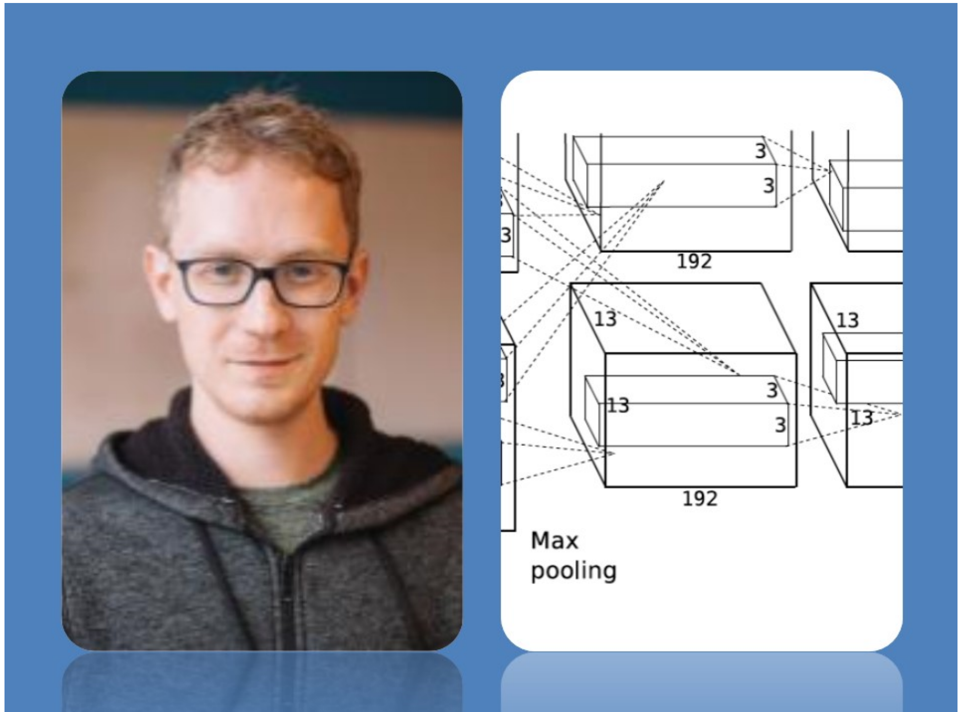
LSTM - state



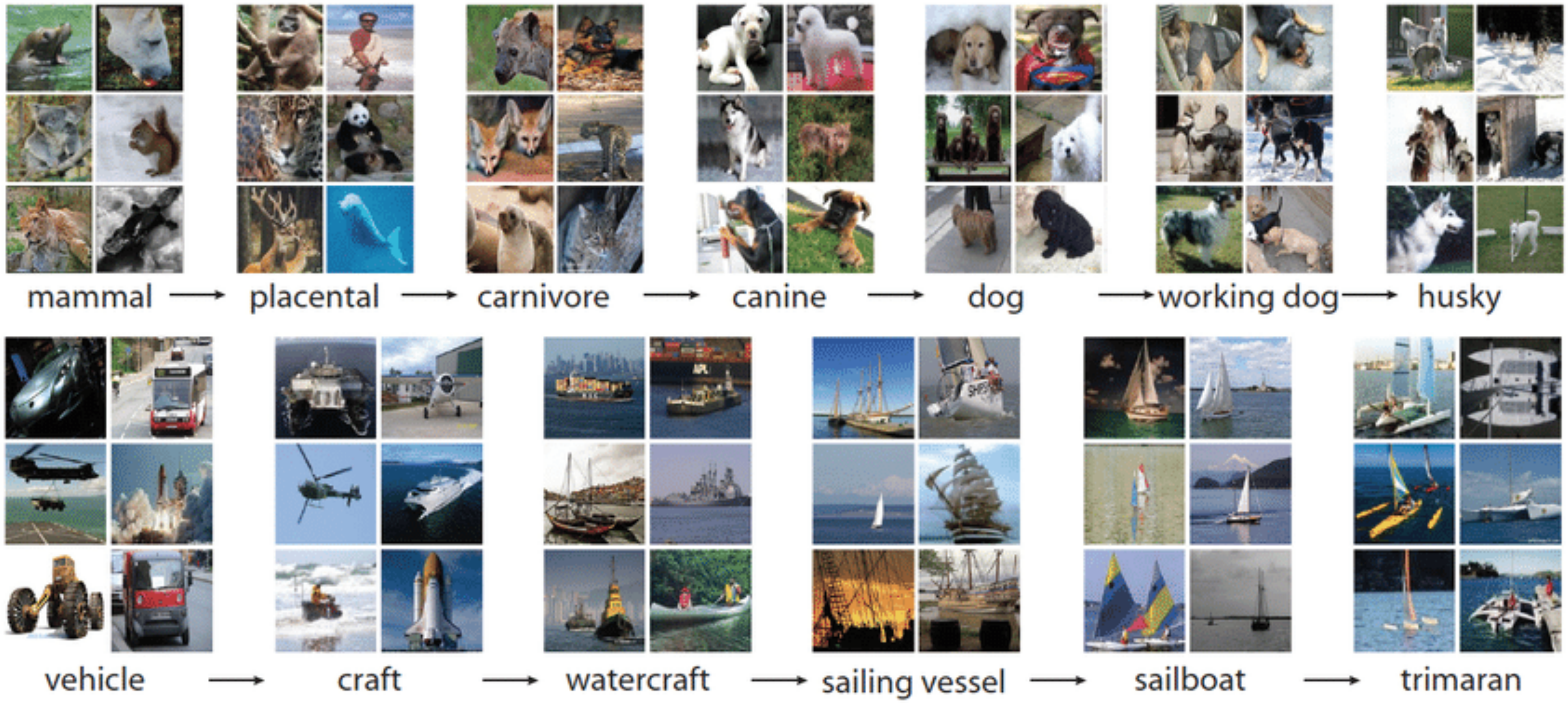
- Long Short Term Memory
- Explicit state, suffers less from vanishing gradient
- Good building block for large recurrent networks

ML Revolution

- Why did ML take off in 2012?
- **Algorithms** — deep learning, CNN
- Labeled **Data sets** (ImageNet)
- Compute Power (**GPU**)
- Alex Krizhevsky [2012]



ImageNet

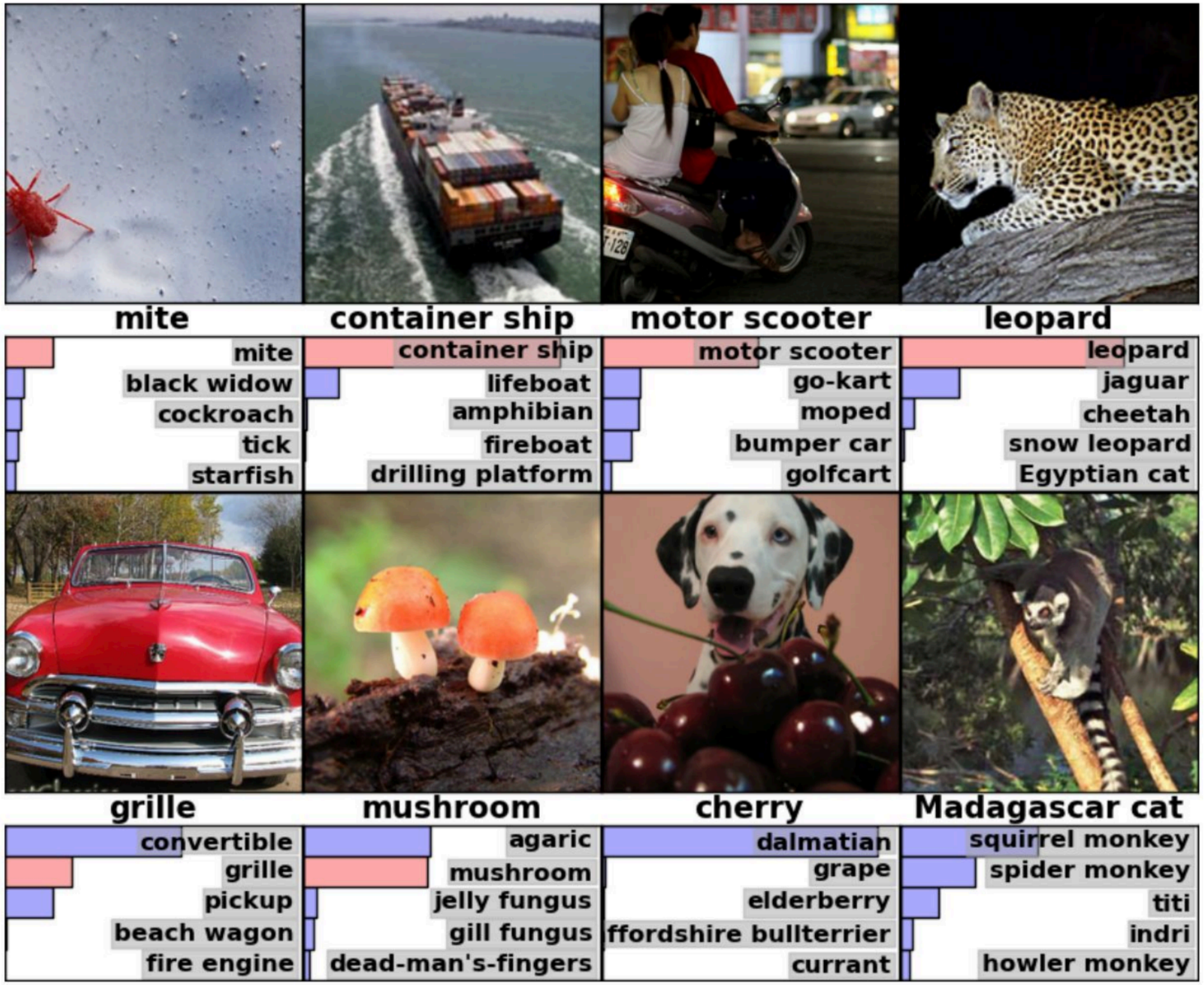


mammal → placental → carnivore → canine → dog → working dog → husky
vehicle → craft → watercraft → sailing vessel → sailboat → trimaran

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.



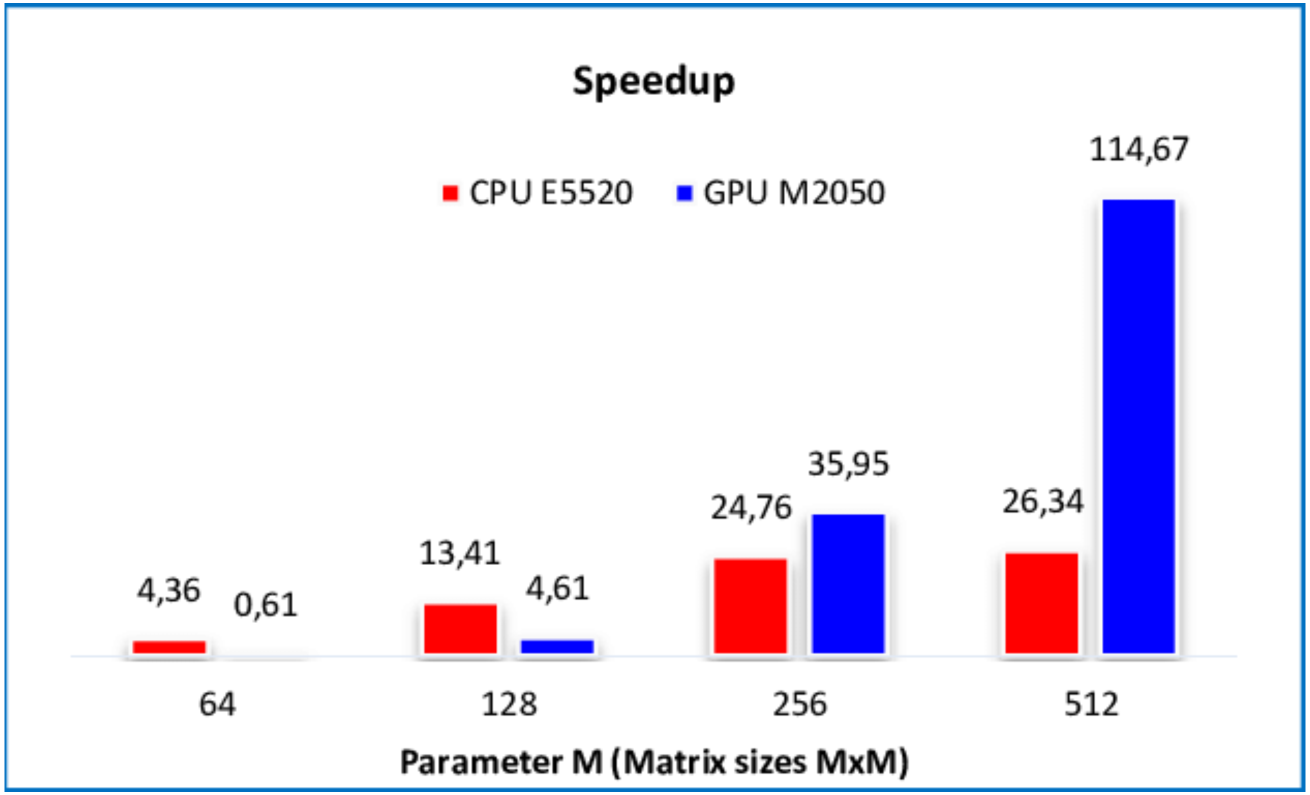
PyTorch & TensorFlow

```python
import torch.nn as nn
import torch.nn.functional as F


class NeuralNet(nn.Module):
    def __init__(self):
        super(NeuralNet, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

```python
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(30, input_dim=8, activation='relu'))
    model.add(Dense(15, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # Compile model
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
```

GPU

- GPU parallel Matrix Multiply large speedup
- Use GPU versions of TensorFlow and PyTorch



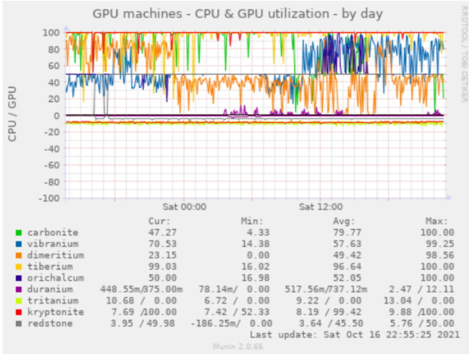
DataScience Lab

- DataScience Lab
- ALICE
- Google Colab

liacs Leiden Institute of Advanced Computer Science

GPU

The following machines contain high-end GPUs and can be used for CUDA-supported computations, such as training neural networks using e.g. TensorFlow and PyTorch.



Hostname	RAM	CPU/GPU	Local Storage	Status	Available to?
`duranium.liacs.nl`	256GB	20 Intel Xeon E5-2650v3 cores @ 2.30GHz (40 threads), 6 NVIDIA GTX 980 Ti (6GB memory each), 2 NVIDIA Titan X (12GB memory each)	3TB under `/local`	OK	Students
`tritanium.liacs.nl`	1TB	20 Intel Xeon E5-2650v3 cores @ 2.30GHz (40 threads), 16 NVIDIA Tesla K80 (11.5GB memory each) These are 8 dual-GPU boards	3TB under `/local`	OK	Staff
`kryptonite.liacs.nl`	64GB	16 Intel Xeon E5-2650 cores @ 2.00GHz (32 threads), 2 NVIDIA GeForce RTX 2080 Ti (11GB memory each)	7.3TB under `/local`	OK	Staff
`phlogiston.liacs.nl`	64GB	16 Intel Xeon E5-2650 cores @ 2.00GHz (32 threads), 2 NVIDIA GeForce RTX 2080 Ti (11GB memory each)	8.6TB under `/local`	OK	Students
`redstone.liacs.nl`	64GB	16 Intel Xeon E5-2650 cores @ 2.00GHz (32 threads), 2 NVIDIA GeForce RTX 2080 Ti (11GB memory each)	8.6TB under `/local`	OK	Staff
`runite.liacs.nl`	64GB	16 Intel Xeon E5-2650 cores @ 2.00GHz (32 threads), 2 NVIDIA GeForce RTX 2080 Ti (11GB memory each)	8.6TB under `/local`	OK	Staff
`carbonite.liacs.nl`	256GB	24 Intel Xeon Silver 4214 cores @ 2.20GHz (48 threads), 2 NVIDIA GeForce RTX 3090 Ti (24GB memory each)	3.4TB under `/local`	OK	Staff
`dimeritium.liacs.nl`	256GB	24 Intel Xeon Silver 4214 cores @ 2.20GHz (48 threads), 2 NVIDIA GeForce RTX 3090 (24GB memory each)	3.4TB under `/local`	OK	Staff
`orichalcum.liacs.nl`	256GB	24 Intel Xeon Silver 4214 cores @ 2.20GHz (48 threads),	3.4TB under `/local`	OK	Staff

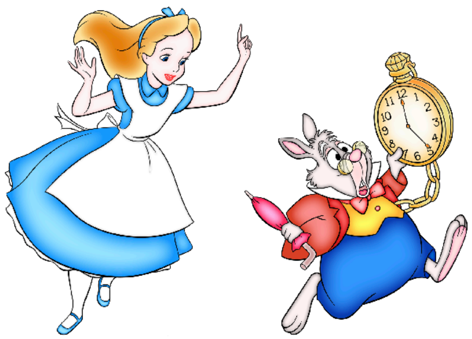
About ALICE

Actions

From ALICE Documentation

ALICE (Academic Leiden Interdisciplinary Cluster Environment) is the high-performance computing (HPC) facility of the partnership between Leiden University and Leiden University Medical Center (LUMC). It is available to any researcher from both partners. Leiden University and LUMC aim to help deliver cutting edge research using innovative technology within the broad area of data-centric HPC. Both partners are responsible for the hosting, system support, scientific support and service delivery of several large super-computing and research data storage resources for the Leiden research community.

This wiki is the main source of documentation about the ALICE cluster.



Off to research computing Wonderland

Contents

[hide]

Questions?