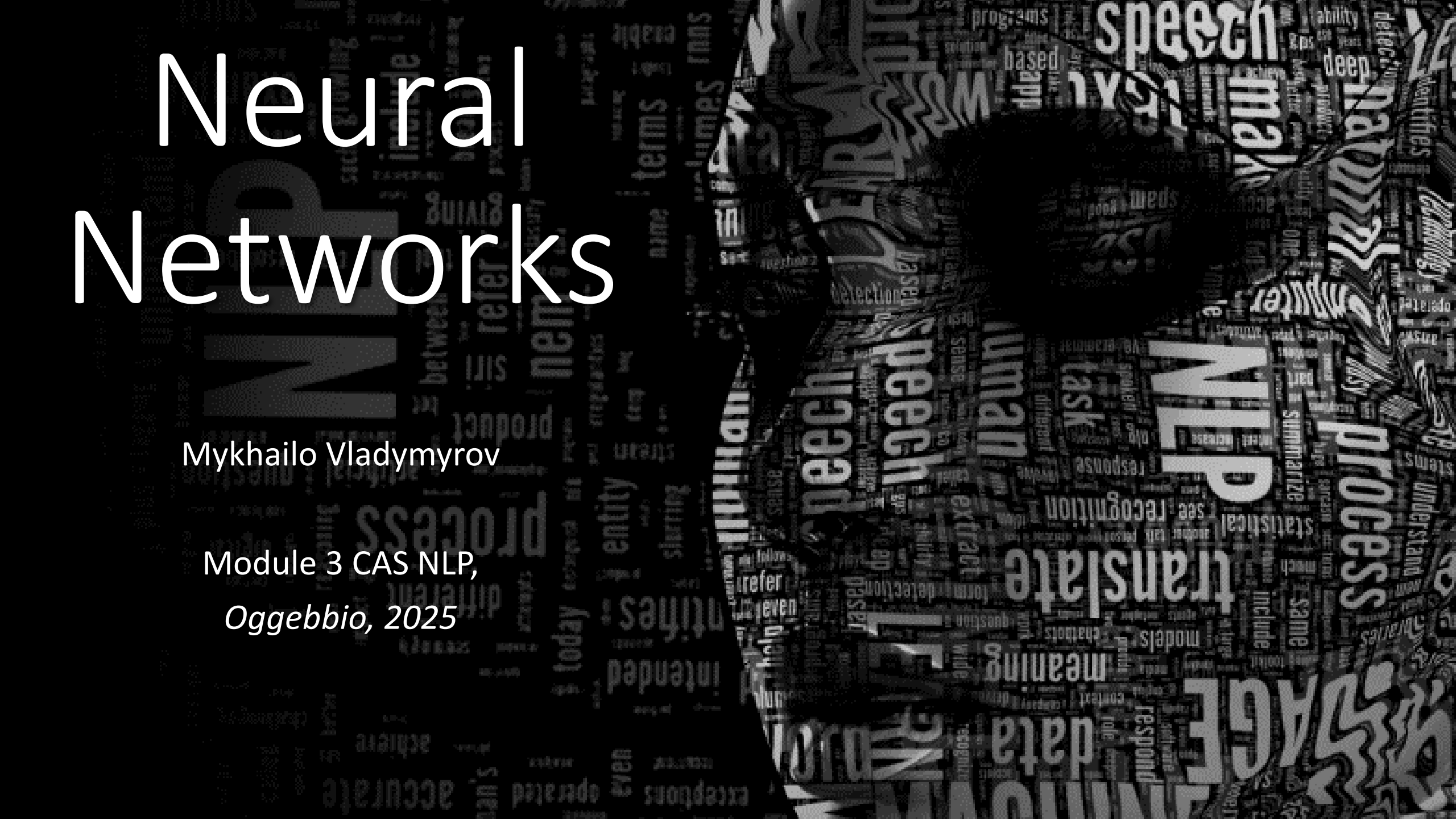
Neural Networks

Mykhailo Vladymyrov

Module 3 CAS NLP,

Oggebbio, 2025

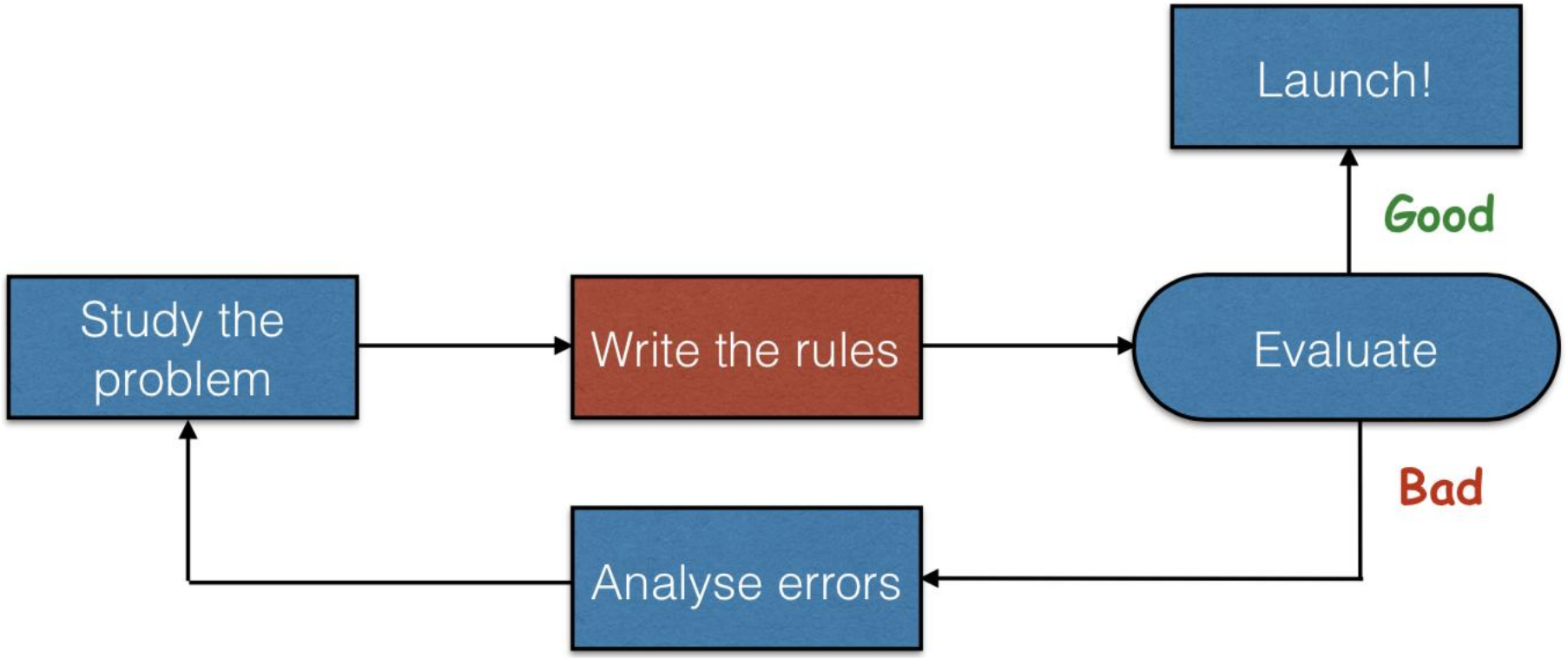


Introduction

What is the aim of Machine Learning?

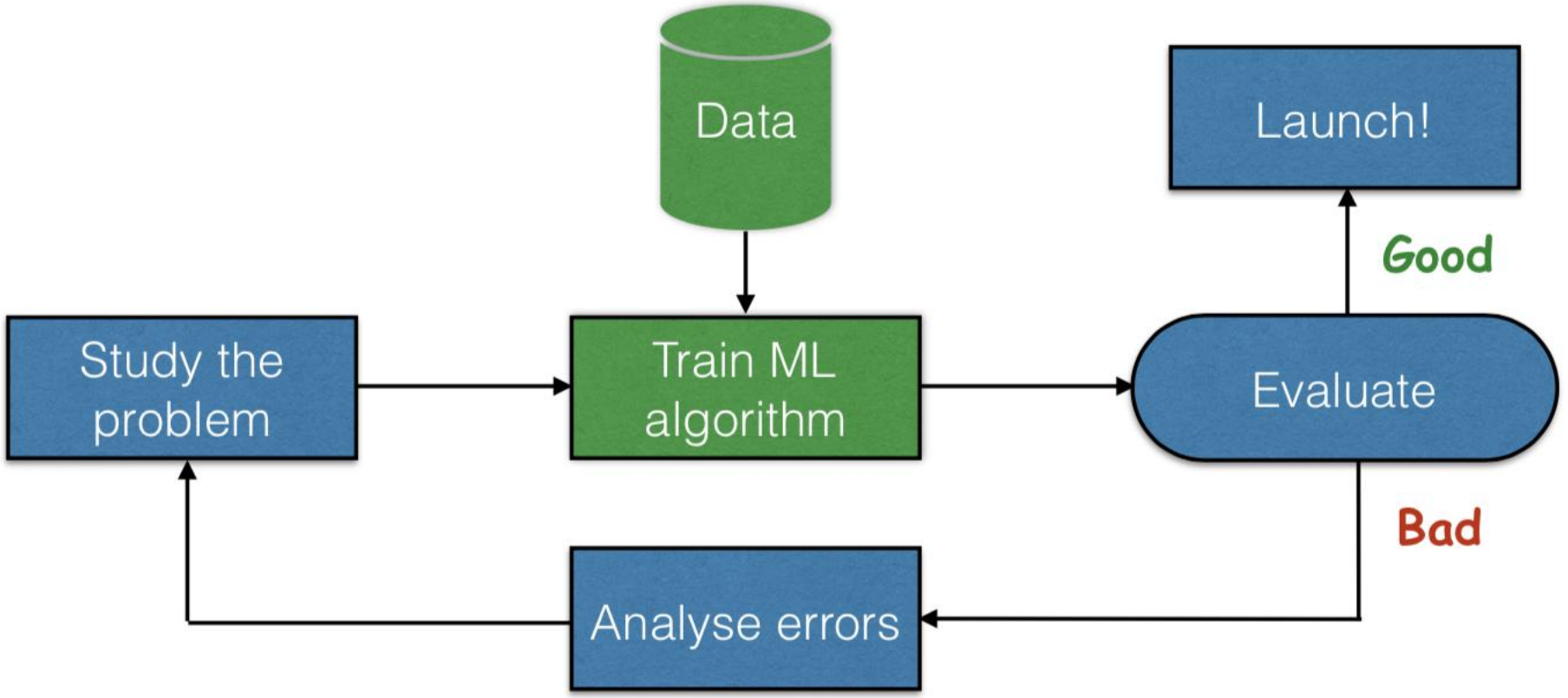
- The aim of Machine Learning is to enable computers to **learn** patterns from **data** and make **predictions** accordingly.

Traditional approach (Software 1.0)



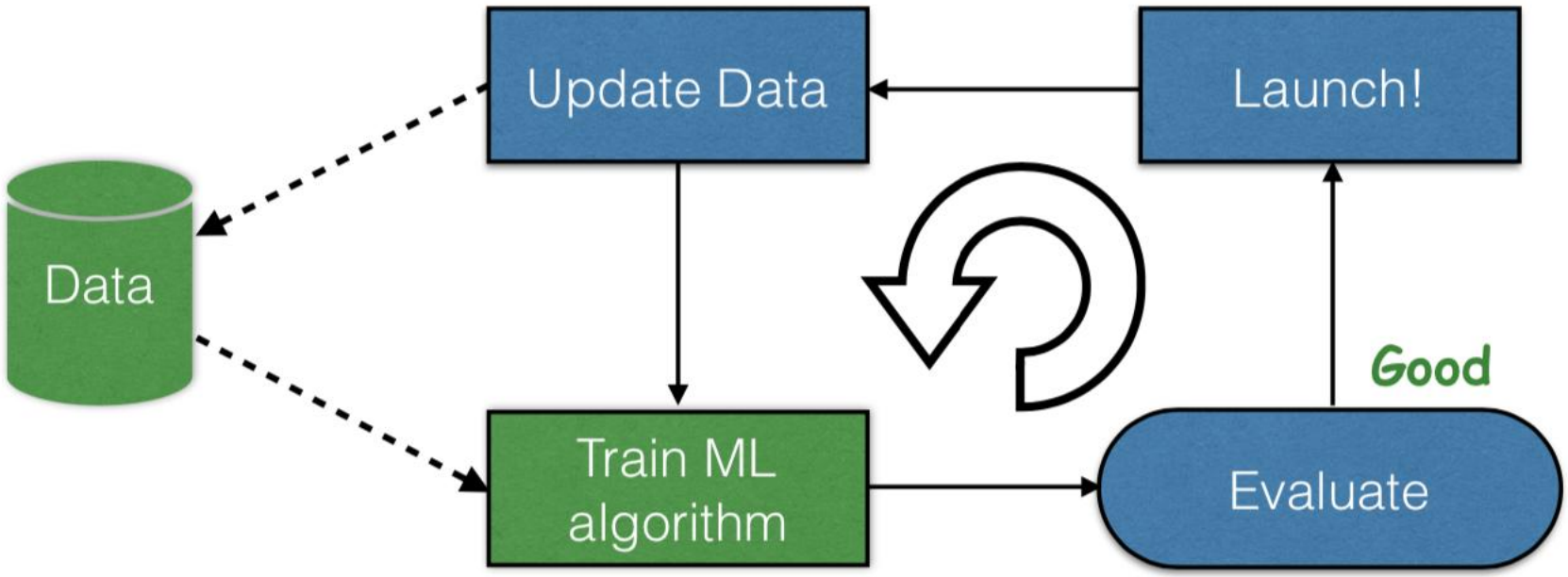
List of all the knowledge and formal rules

Machine Learning approach (Software 2.0)



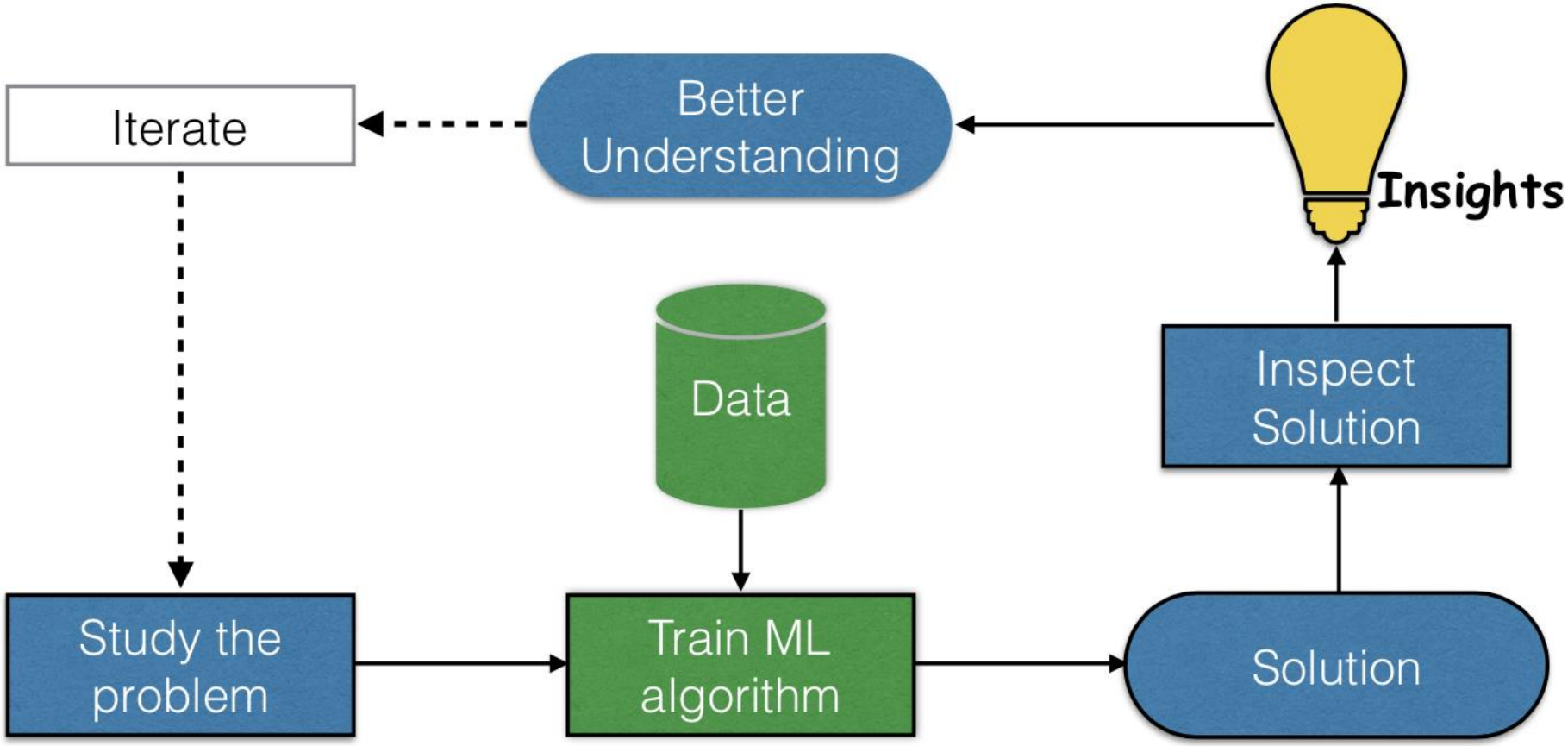
Learning from examples

Machine Learning approach (Software 2.0)



Adapting to change

Machine Learning approach (Software 2.0)

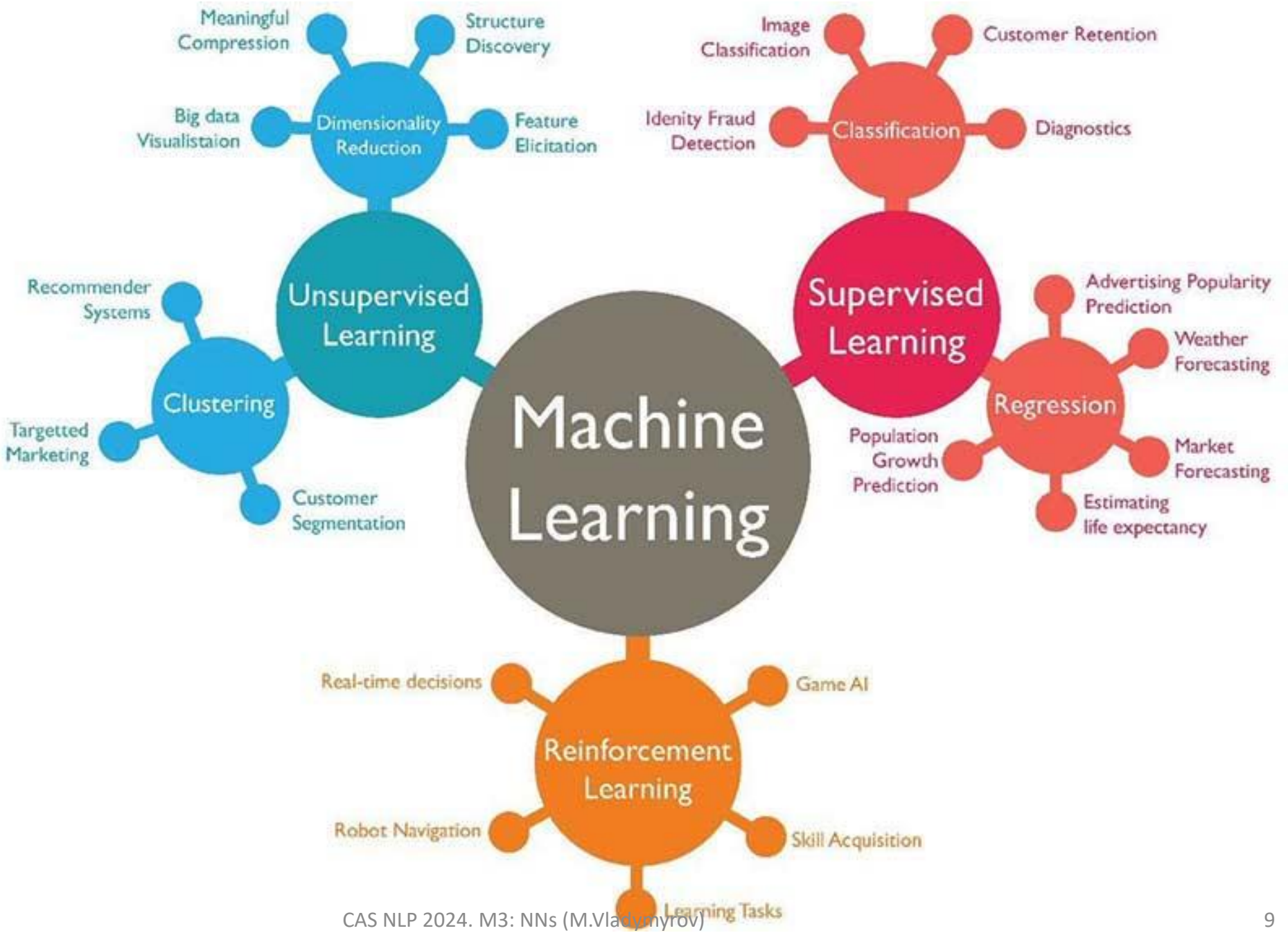


Help Humans learn

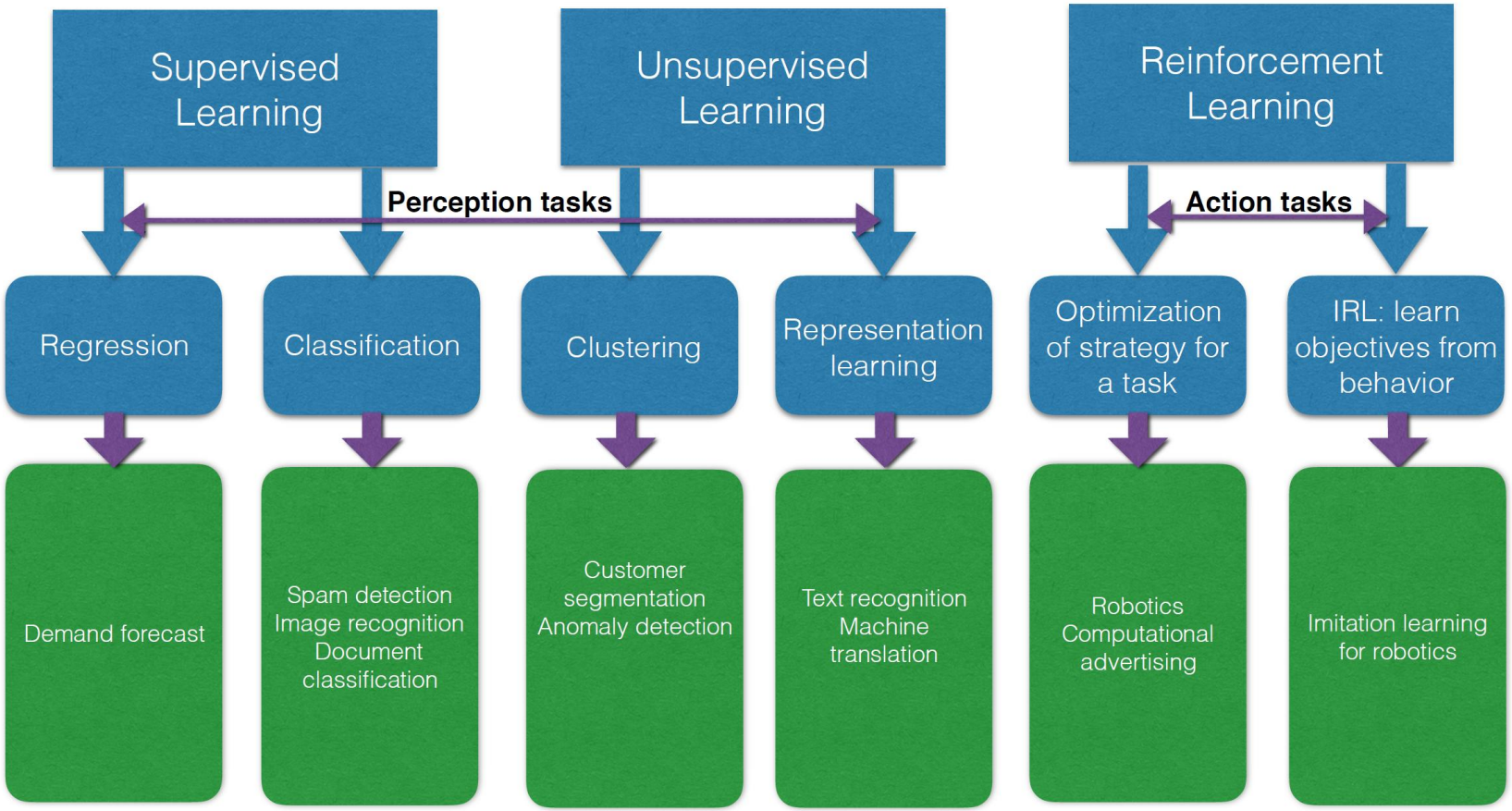
Tasks

Find the function f that satisfies f(x) = y using the *data*

Tasks

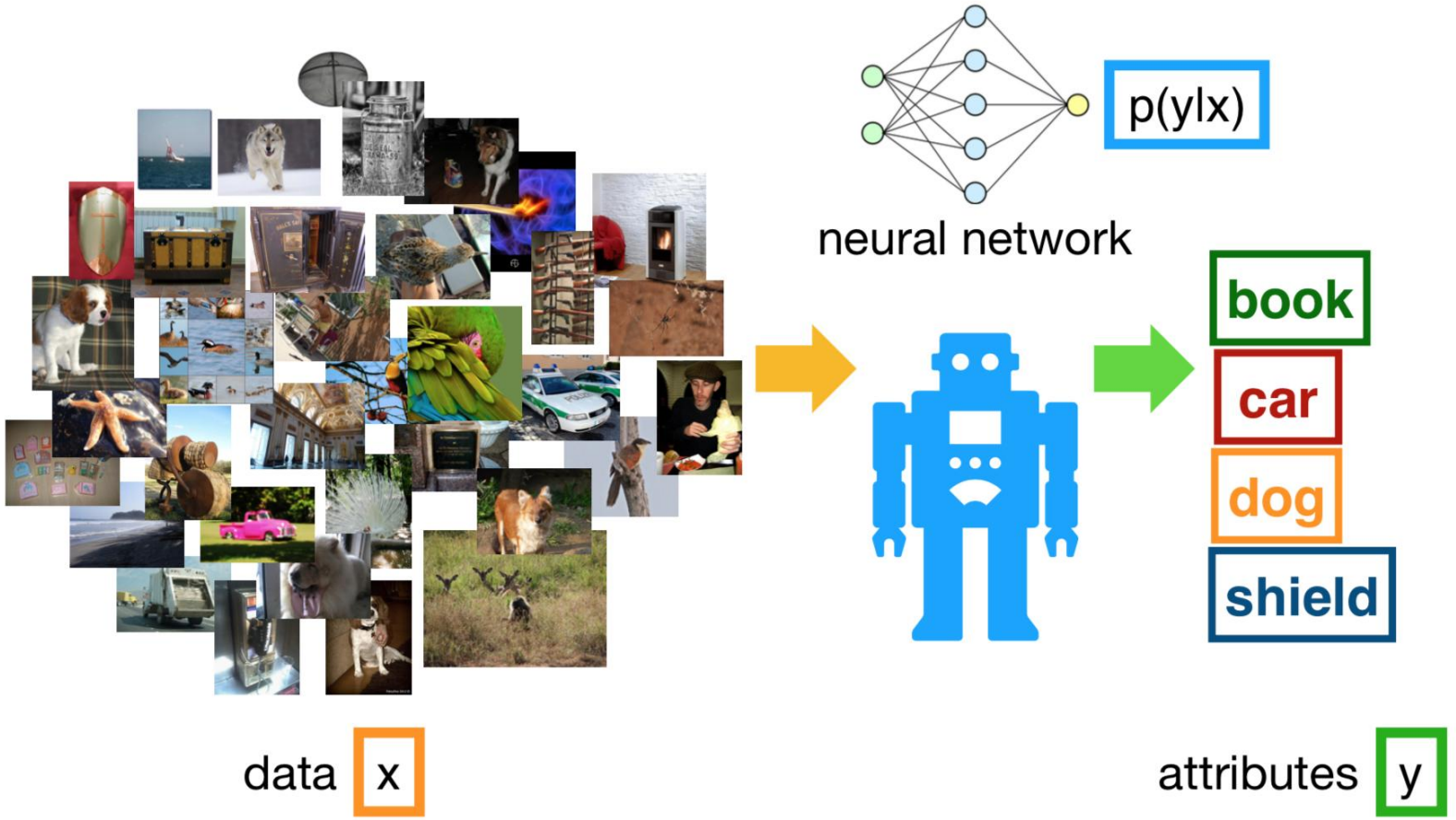

Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Problem Types



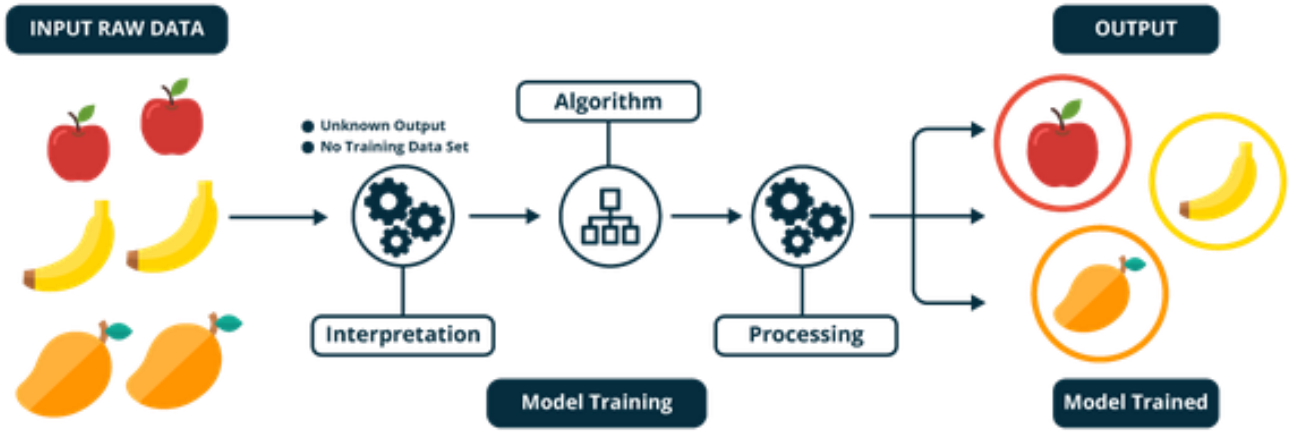
Supervised Learning

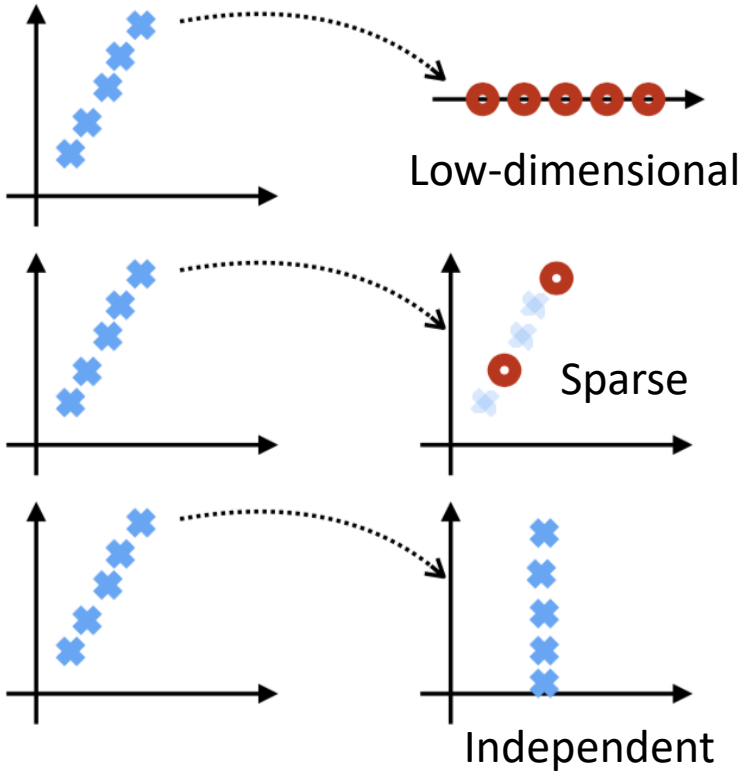
- Prediction of an output **y** given an input **x**



Unsupervised Learning

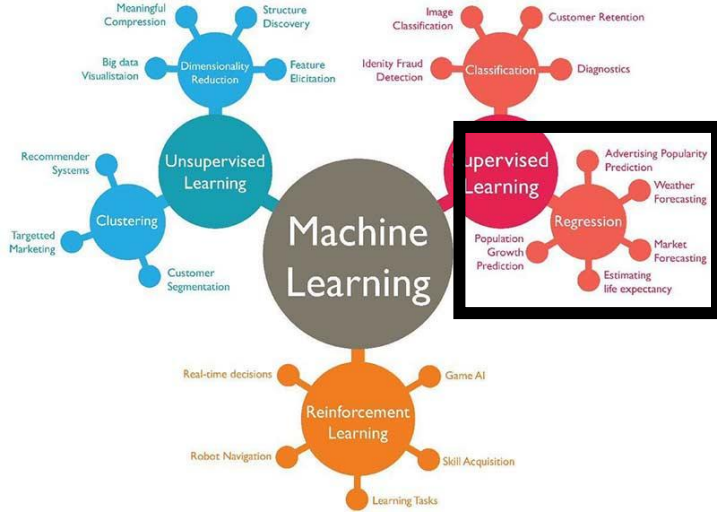
- Find a *suitable data representation*
  - Preserving all task-relevant information
  - Simpler than the original data and easier to use





Regression

Predict results within *a continuous output*



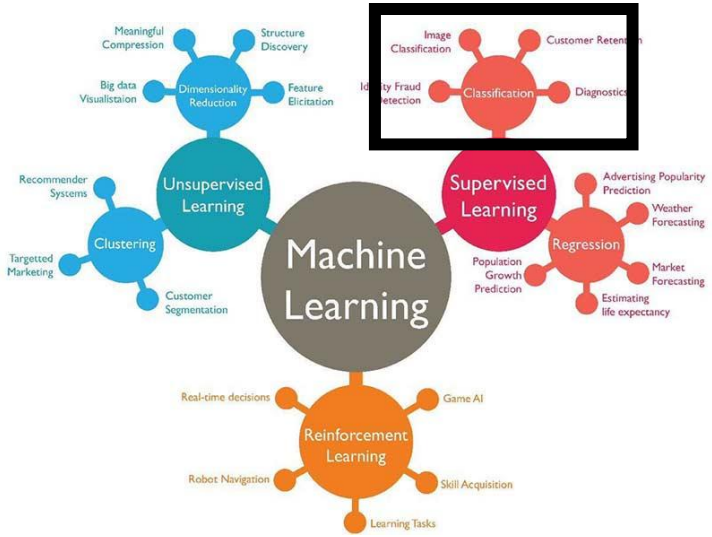
$82000

$55500

???

Classification

- categorize new inputs as belonging to one of a set of categories → Predict results within *a discrete output (categories)*

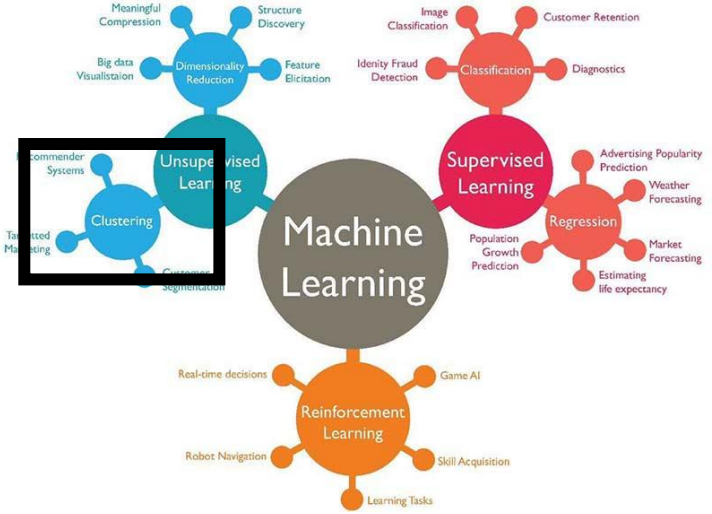


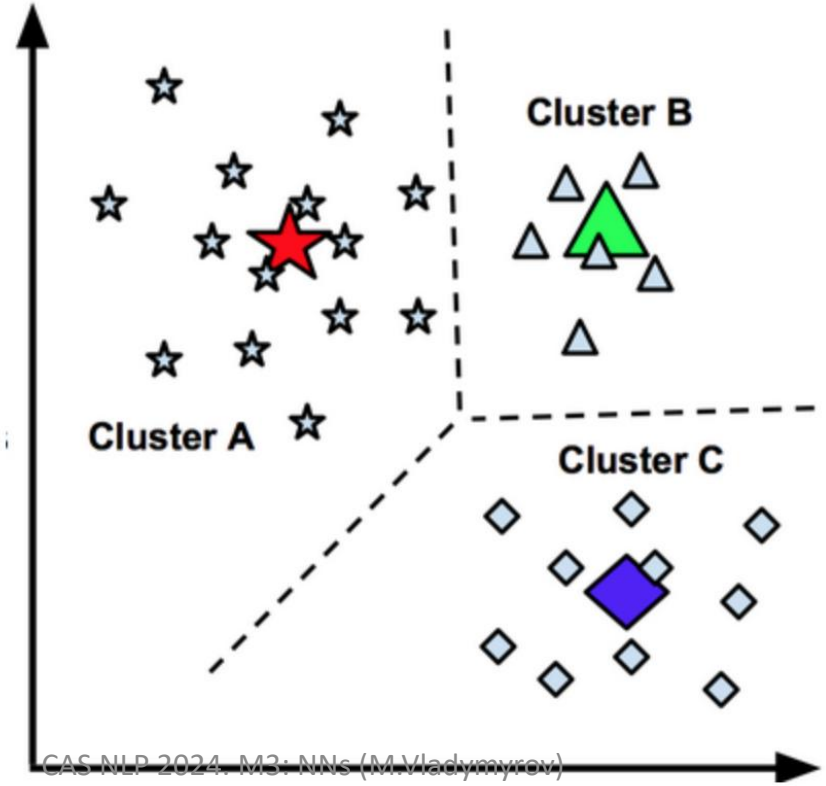
Dog: 96%
Cat: 29%
Duck: 2%
Bird: 0%

Dog: 36%
Cat: 94%
Duck: 2%
Bird: 1%

Clustering

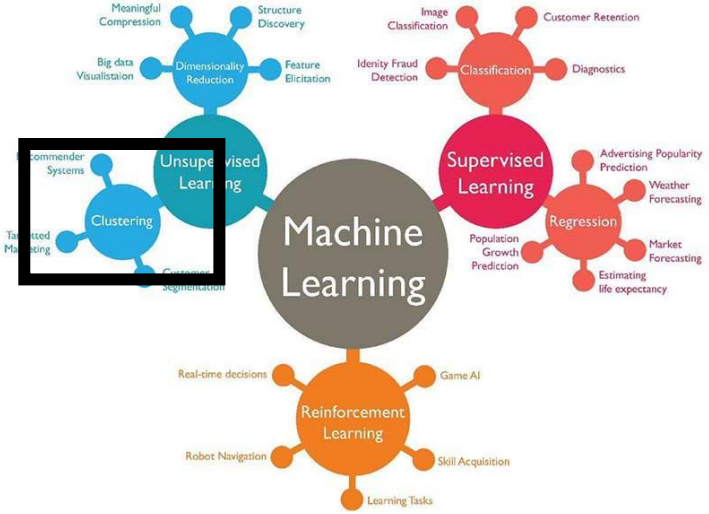
- create a set of categories, for which individual data instances have a set of common or similar characteristics.

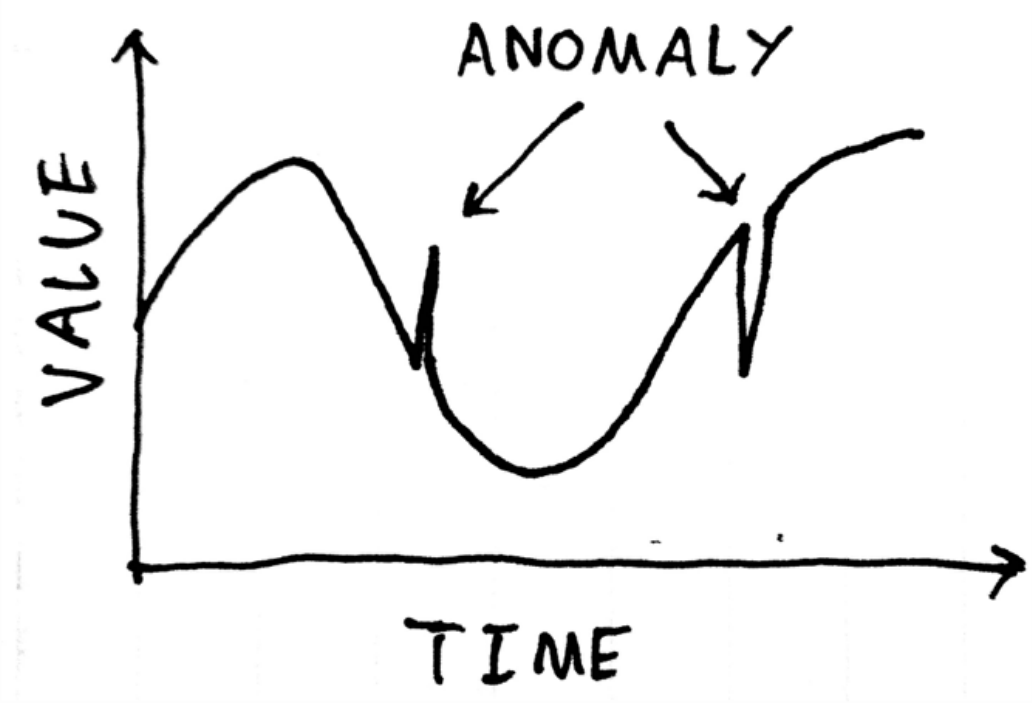


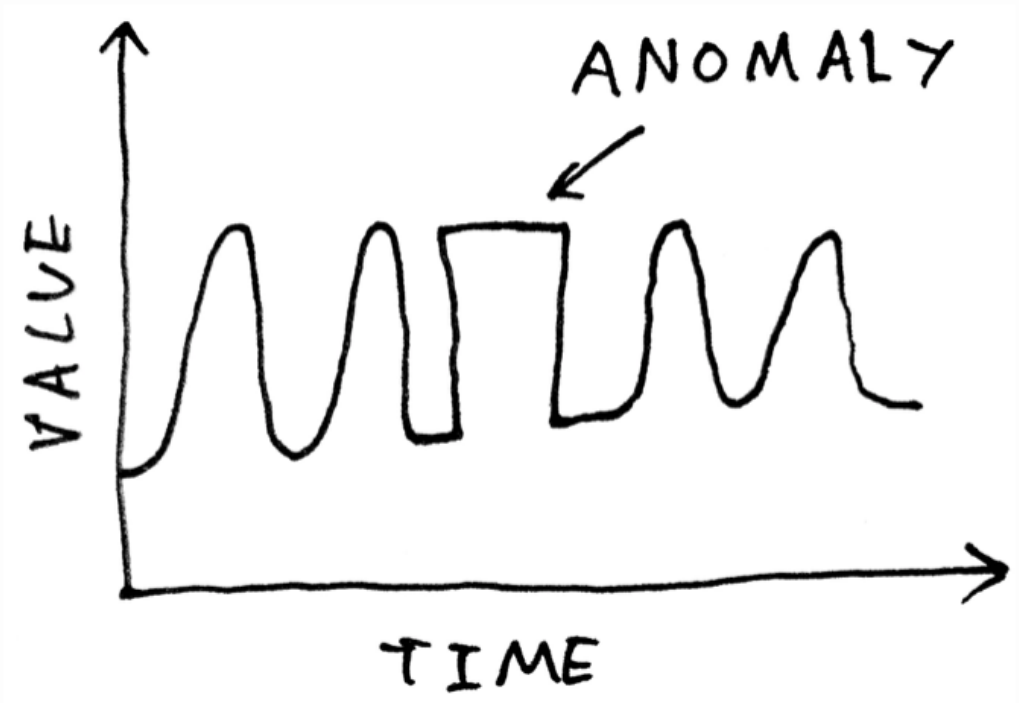


Anomaly Detection

- determine whether specific inputs are out of the ordinary.

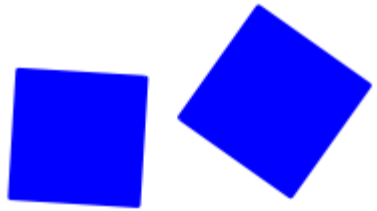




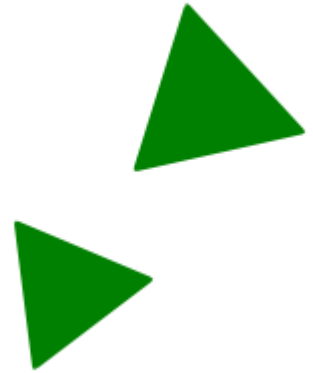


Classification example

Which class does the object belong to?



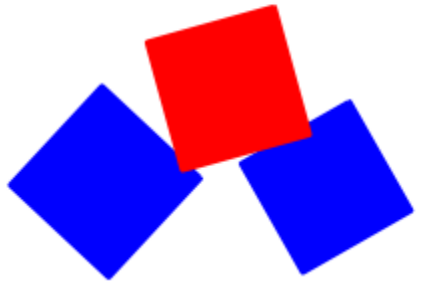
Class 1



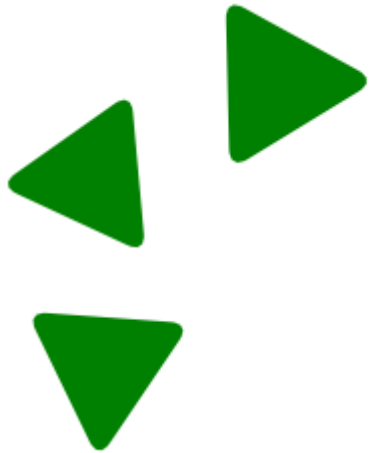
Class 2

Task complexity

Which class does the object belong to?



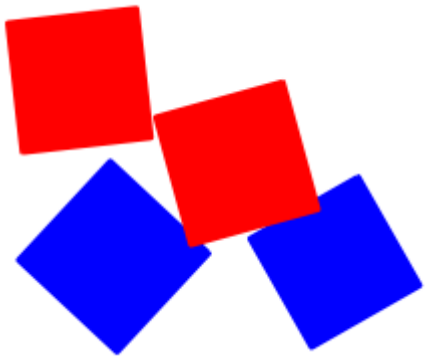
Class 1



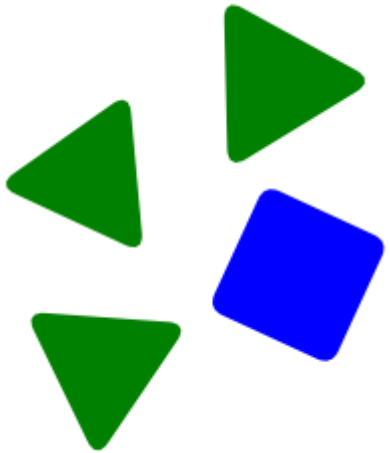
Class 2

Task complexity

Which class does the object belong to?



Class 1



Class 2

Data

Data assumption

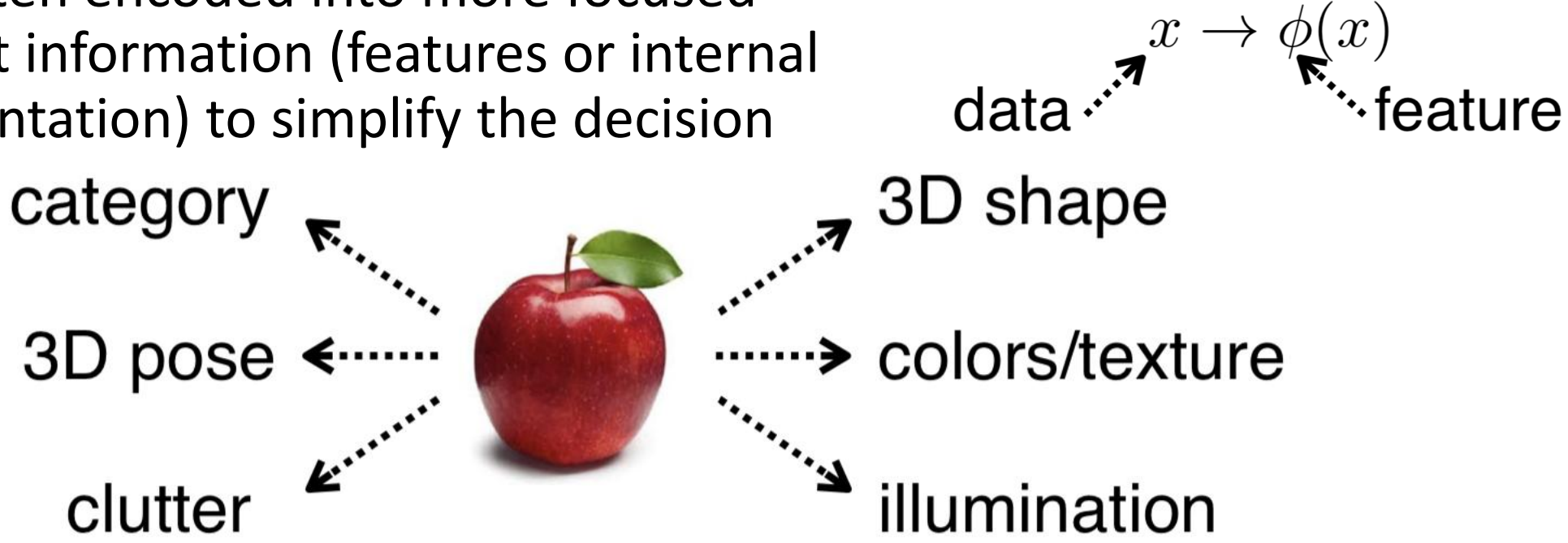
- independent and identically distributed

    1) Come from the *same distribution*

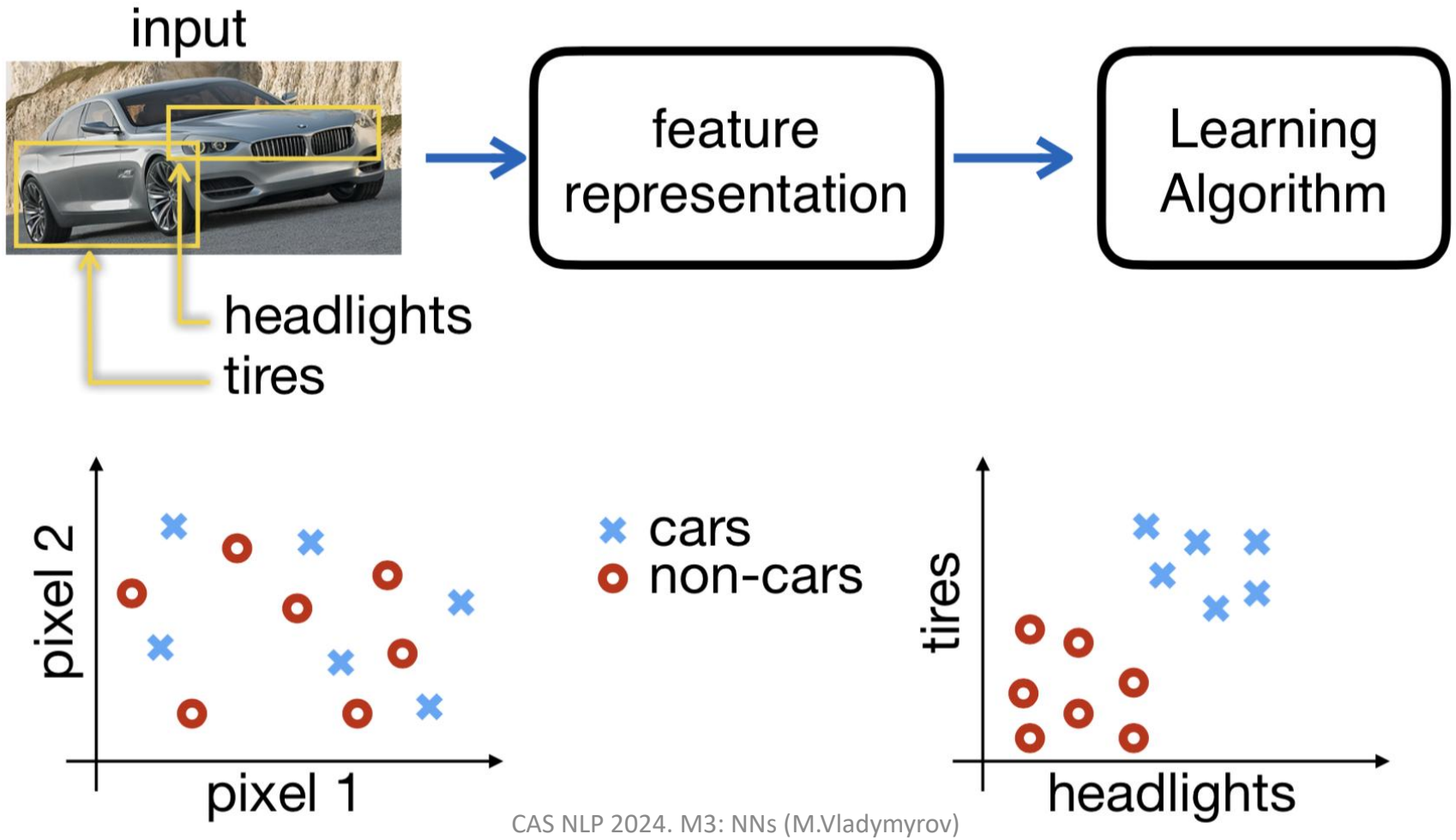
    2) Are *independent*

Features

- Data often encoded into more focused relevant information (features or internal representation) to simplify the decision

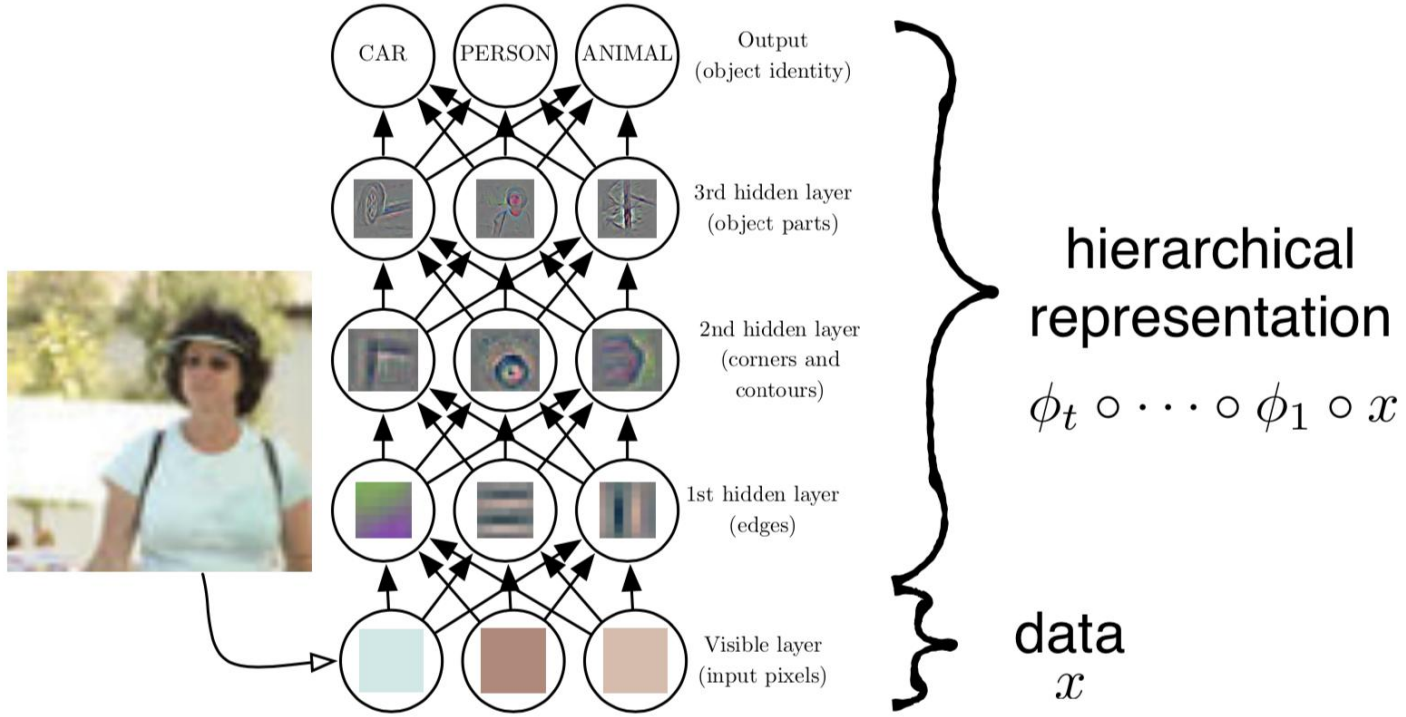


Features Example: Image classification



Deep Learning

« Build a machine that can learn from experience and understand the world as a hierarchy of concepts »

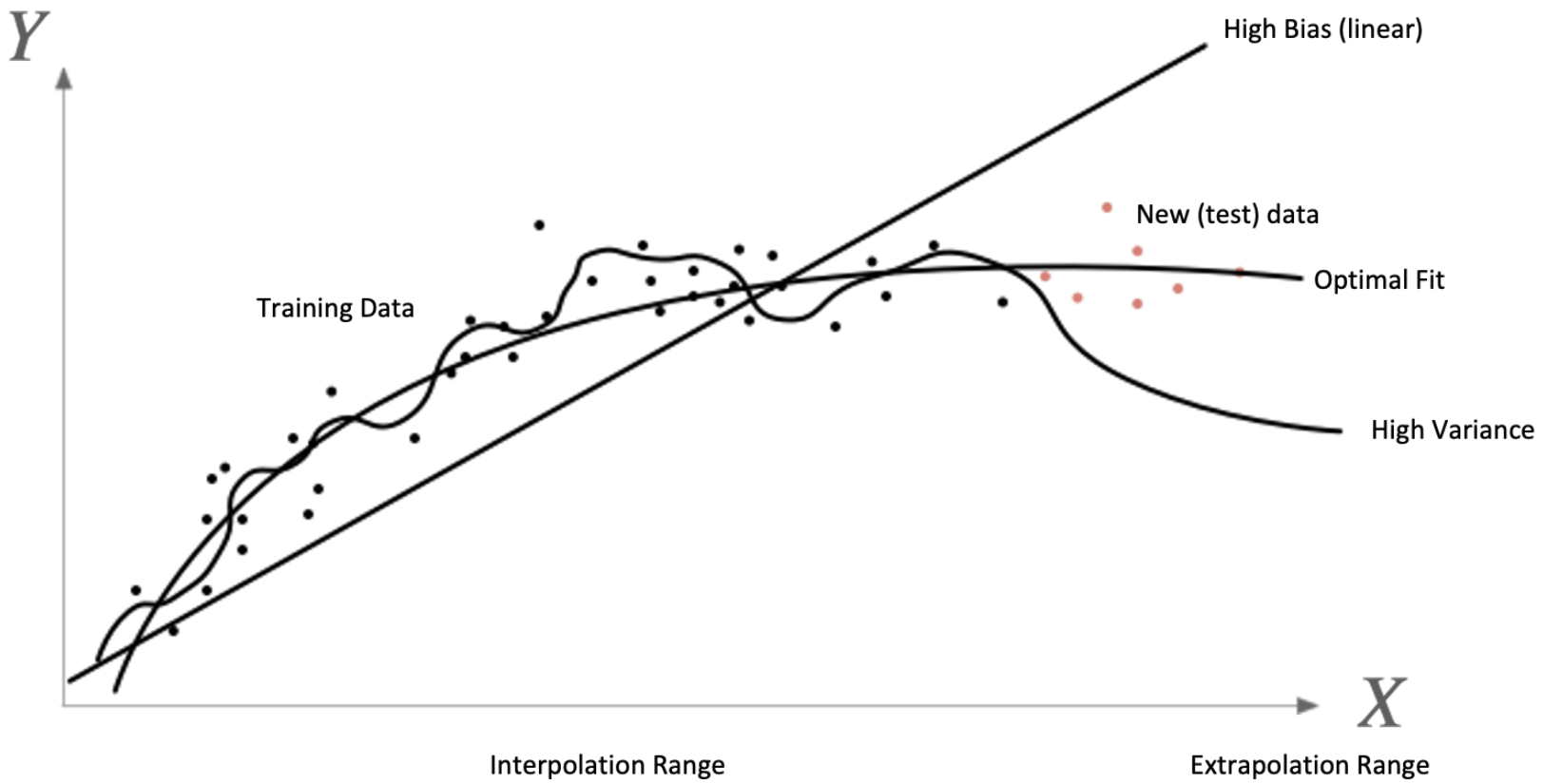


Training/Validation/Test sets

- Separate the data into 2(3) sets
  - Training set for training
  - Validation set to find the best parameters
  - (Test set to estimate the performance)
- Separation depends on the size of the dataset

- Make sure no algorithmic decisions are being made using data which are also being used to test the algorithm

Training/Validation/Test sets



First of all, understand your data !

- Carry out manual error analysis
  - Look at *mislabeled development set* examples (*do not look at test set*)
  - For example : check by hand 500 pictures (incorrect labels ? Foggy pictures ? Other causes ?)
- Clean up incorrectly labeled data
  - Apply same process to your dev and test sets !

how well the algorithm performs on the “walking” task

Performance Measure P

Estimate the ML algorithm performance on task T using the validation set

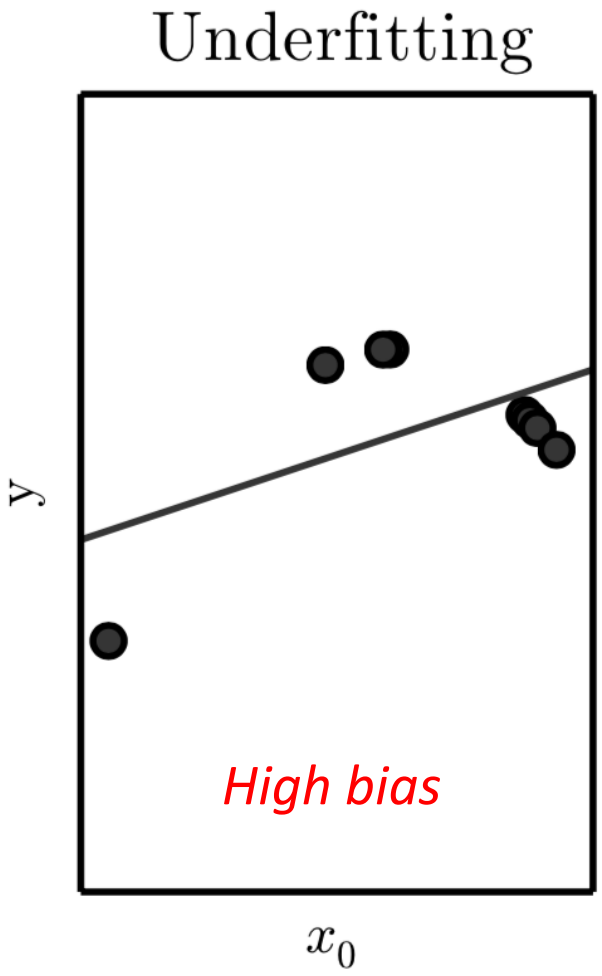
Introduction

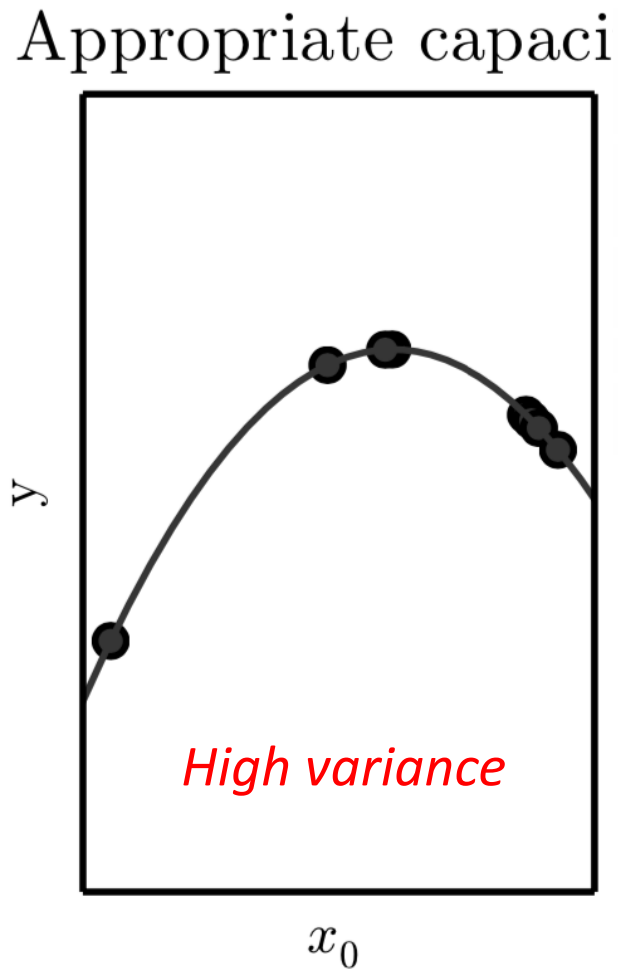
- To evaluate a ML algorithm, we need a way to measure **how well it performs on the task**

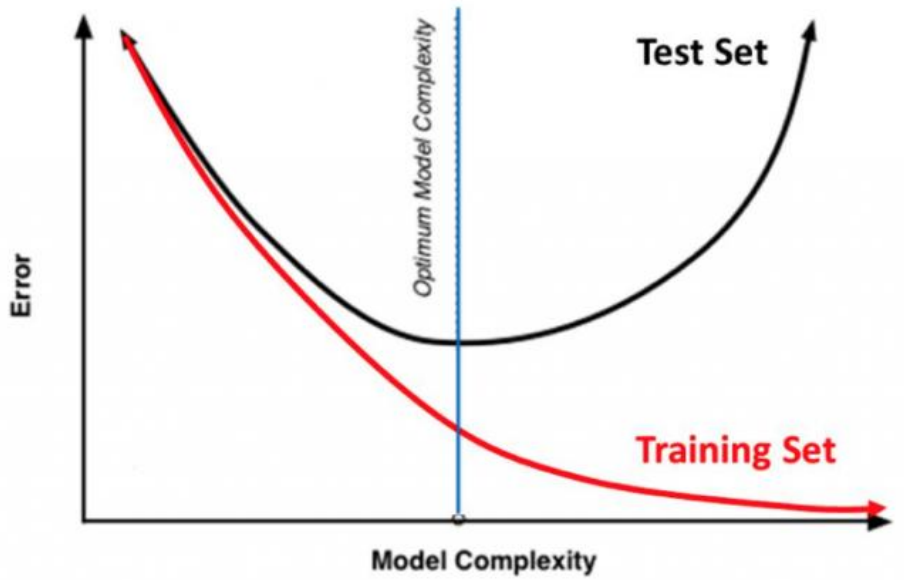
- It is measured **on a separate set** (test set) from what we use to build the function f (training set)

- **Examples** :
  - Classification accuracy (portion of correct answers)
  - Error rate (portion of incorrect answers)
  - Regression accuracy (e.g. least squares errors)

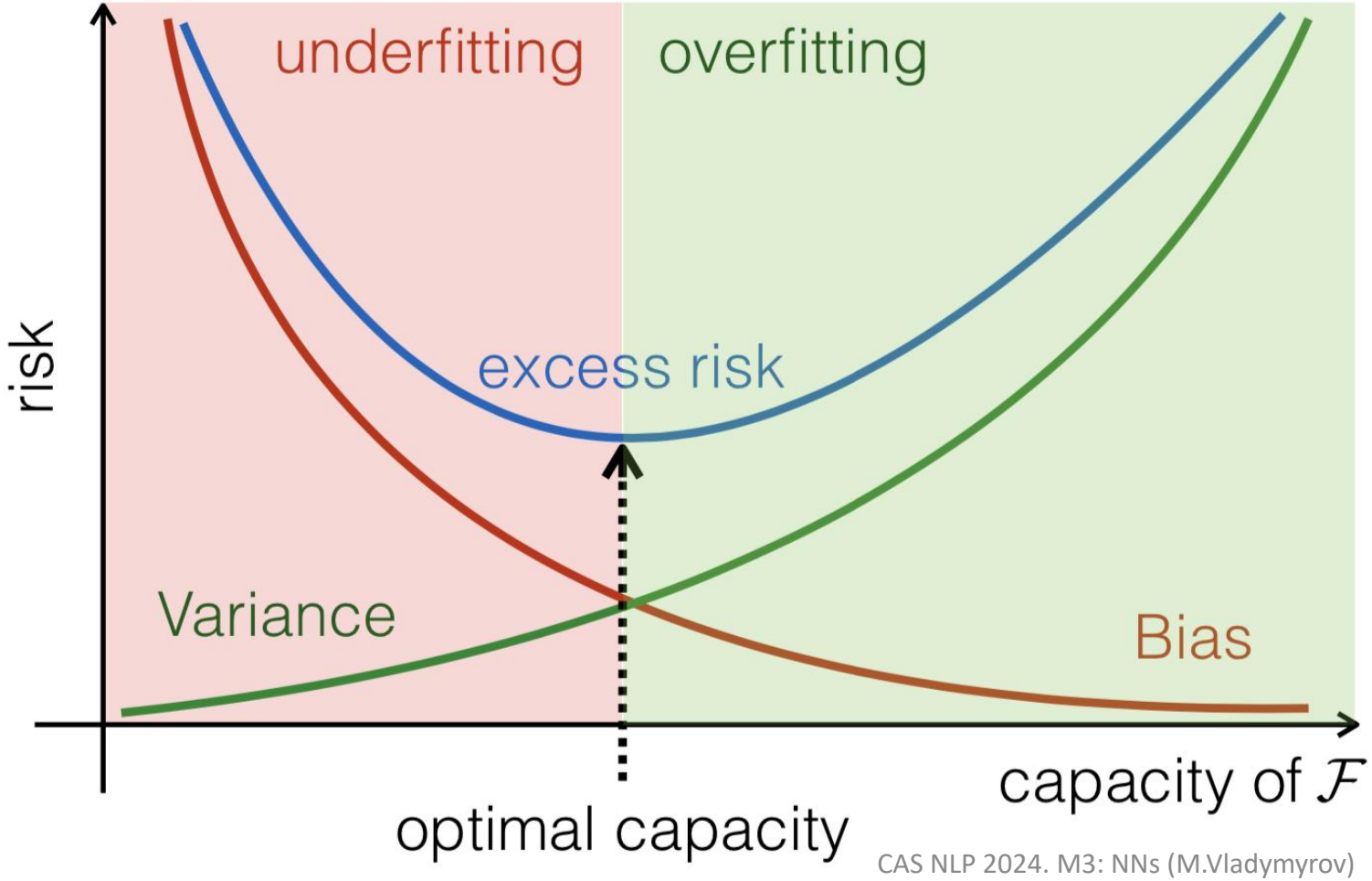
Overfitting and Underfitting

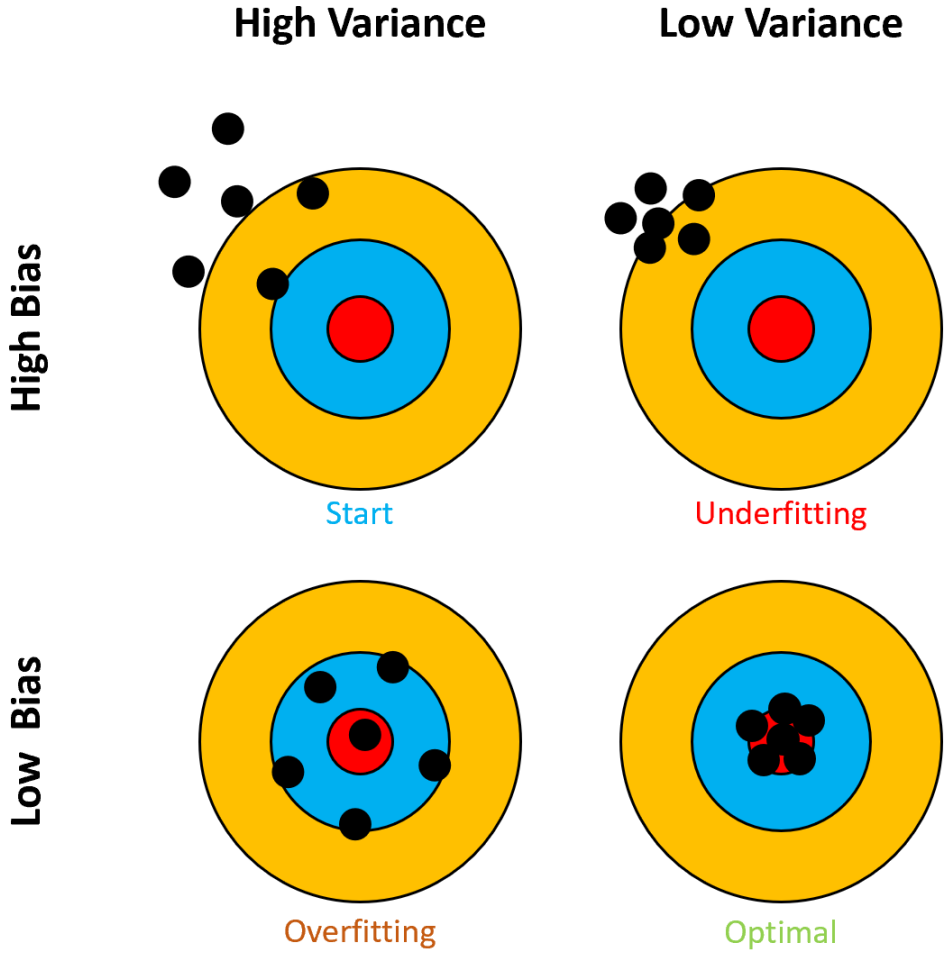






Bias and Variance - Overfitting and Underfitting





Classification Metrics

		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** **hit**	**False negative (FN),** **type II error, miss,** **underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** **type I error, false alarm,** **overestimation**	**True negative (TN),** **correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR–) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

Sources: [6][7][8][9][10][11][12][13][14] view · talk · edit

Source: wiki (Precision and Recall)

F1-score

- F1-score is a harmonic mean combining p and r

$$\text{F1-Score} = \frac{2}{\frac{1}{p}+\frac{1}{r}}$$

	Precision	Recall	F1 Score
Algo 1 →	0.5	0.4	0.444 ✓
Algo 2 →	0.7	0.1	0.175
Algo 3 →	0.02	1.0	0.0392

- See also balanced accuracy (average recall)

Summary

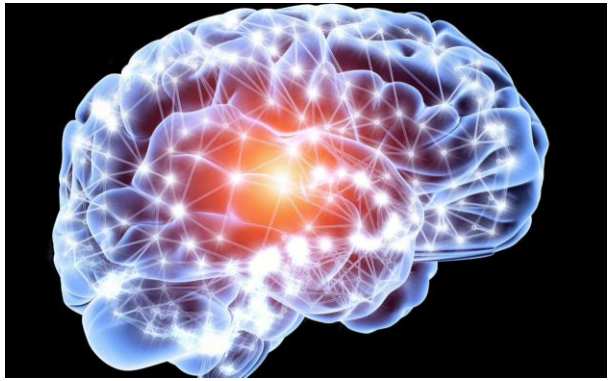
In ML the algorithm is derived directly form data

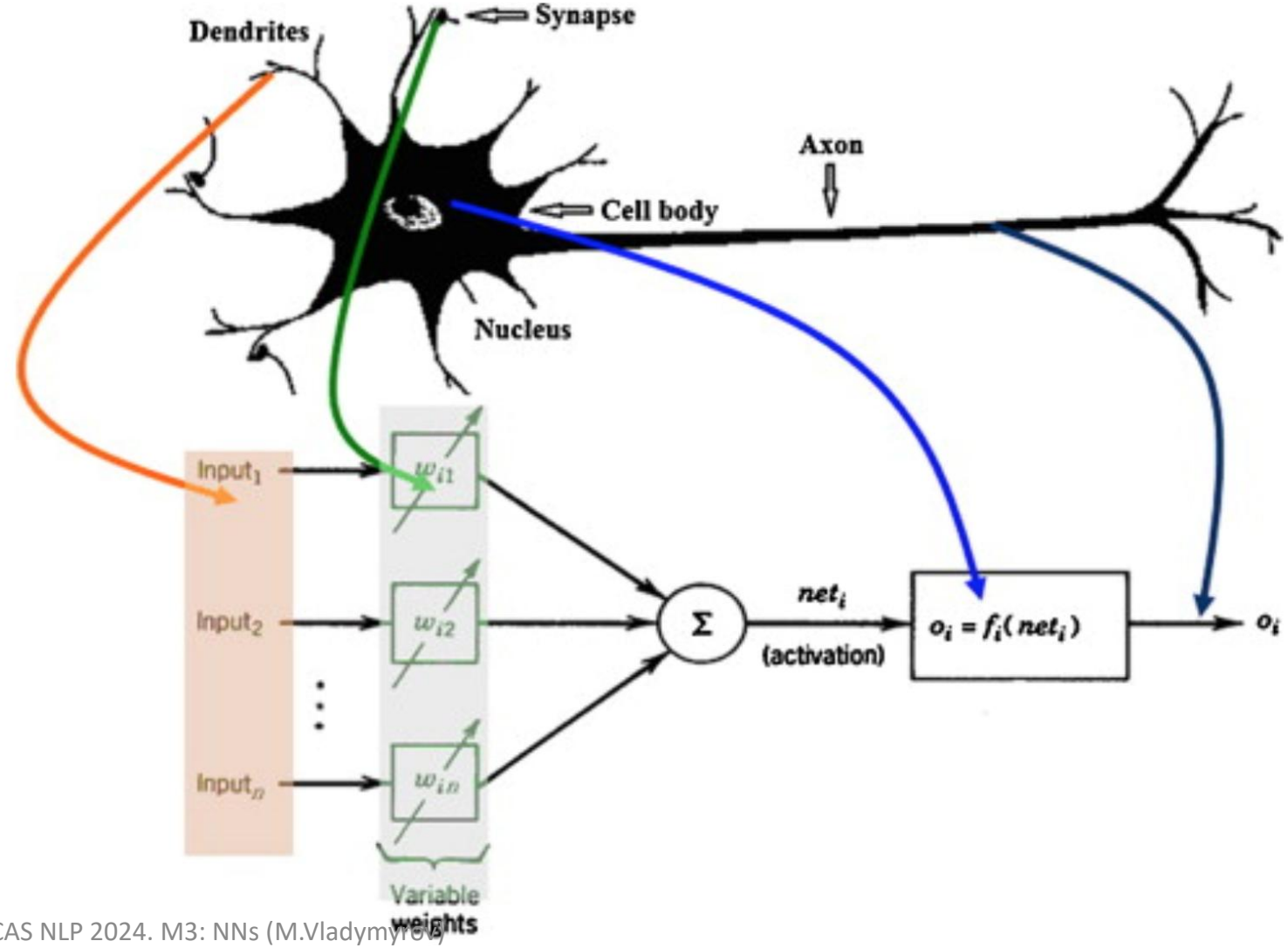
- Any problem can be formulated as one or combination of “standard” tasks
- Quality and quantity of data directly impacts the what model learns
- To ensure the model generalizes well:
  - Use independent data set for testing
  - Use suitable metric

Neural Networks

Neural Network (NN)

Learning algorithm inspired
by *how the brain works*

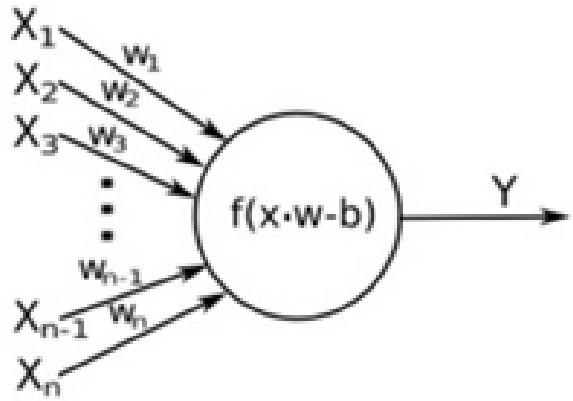




(Deep) Feedforward NN (DFF)

- the simplest type of neural network
- All units are fully connected (between layers)
- information flows from **input** to **output** layer without back loops
- The first single-neuron network was proposed already in 1958 by AI pioneer Frank Rosenblatt

Perceptron



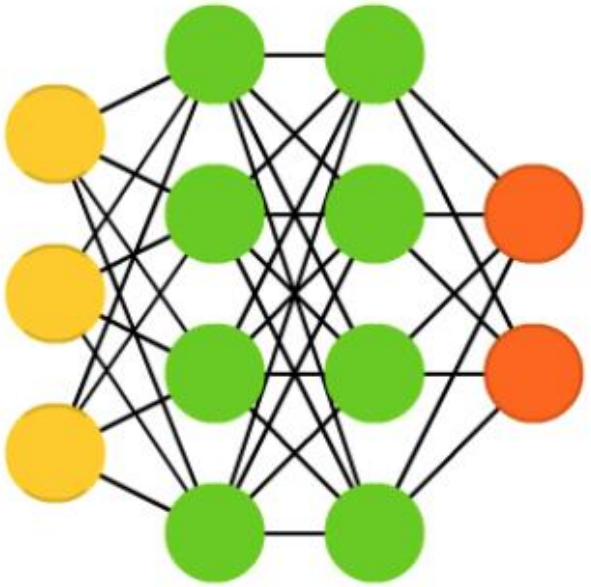
Feed Forward (FF)

(Deep) Feedforward NN (DFF)

- the simplest type of neural network
- All units are fully connected (between layers)
- information flows from **input** to **output** layer without back loops
- The first single-neuron network was proposed already in 1958 by AI pioneer Frank Rosenblatt
- Deep for “more than 1 **hidden layer**”

Feed Forward (FF)

Deep Feed Forward (DFF)



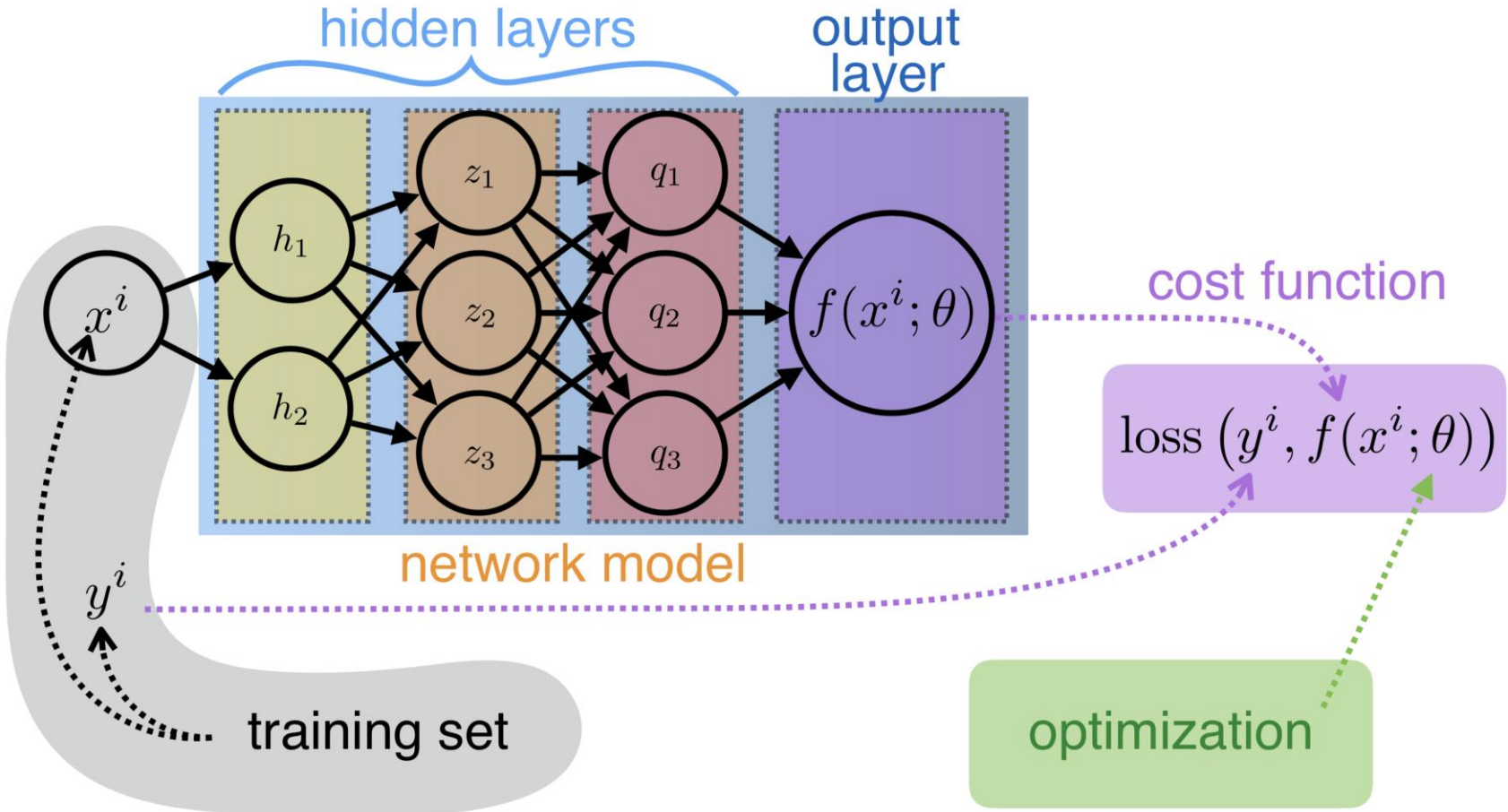
Components of a Neural Network

Given a task (in terms of **I/O mappings**), we need :

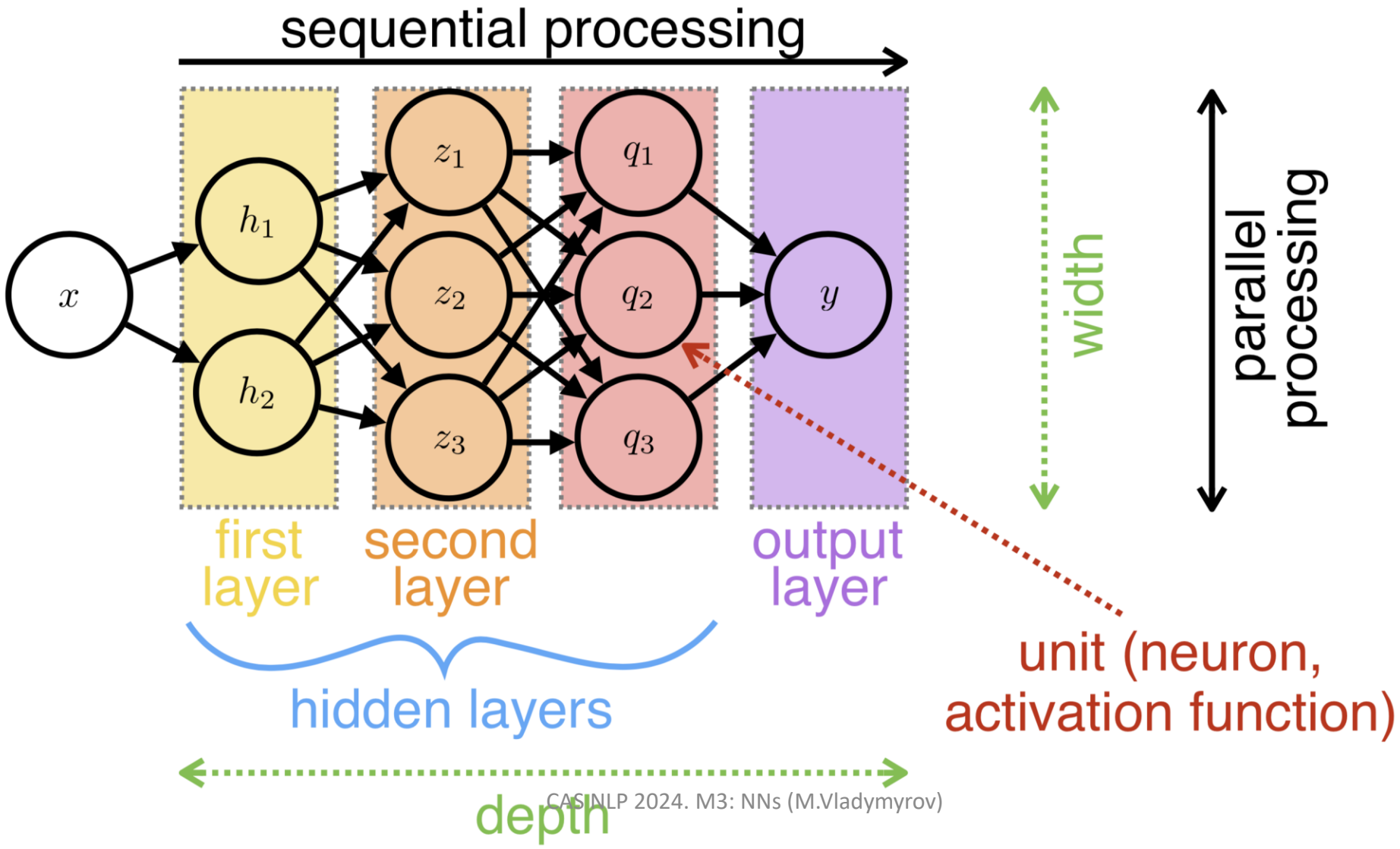
1) **Network model**

2) **Cost function (/objective/loss function)**

3) **Optimization**

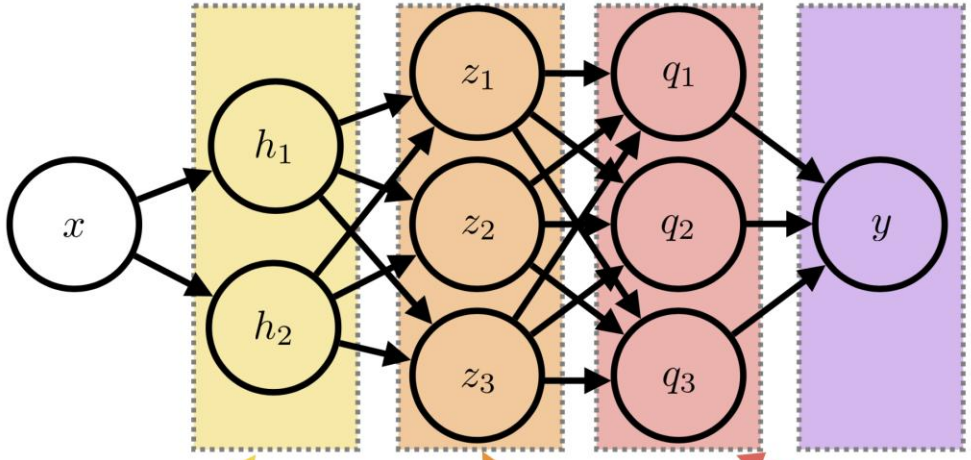


Network model



1) Activation functions

Different types of activation functions for the hidden layers and the output layer



$$y = f_4(q_1, q_2, q_3)$$

$$y = f_4(f_{3,1}(f_{2,1}(f_{1,1}(x), f_{1,2}(x)), \dots), \dots)$$

Hierarchical representation

$$h_1 = f_{1,1}(x)$$
$$h_2 = f_{1,2}(x)$$

$$z_1 = f_{2,1}(h_1, h_2)$$
$$z_2 = f_{2,2}(h_1, h_2)$$
$$z_3 = f_{2,3}(h_1, h_2)$$

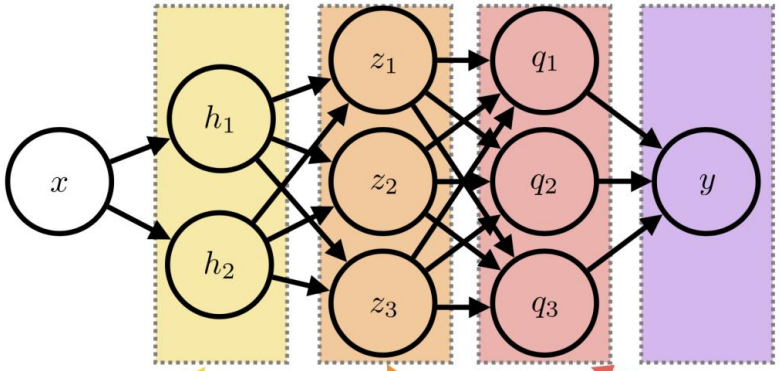
$$q_1 = f_{3,1}(z_1, z_2, z_3)$$
$$q_2 = f_{3,2}(z_1, z_2, z_3)$$
$$q_3 = f_{3,3}(z_1, z_2, z_3)$$

Fully connected $f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2 + b_{2,2}$

Weights w and bias b parameters to optimize

Activation functions

Different types of activation functions for the hidden layers and the output layer



$$y = f_4(q_1, q_2, q_3)$$

$$h_1 = f_{1,1}(x)$$
$$h_2 = f_{1,2}(x)$$

$$z_1 = f_{2,1}(h_1, h_2)$$
$$z_2 = f_{2,2}(h_1, h_2)$$
$$z_3 = f_{2,3}(h_1, h_2)$$

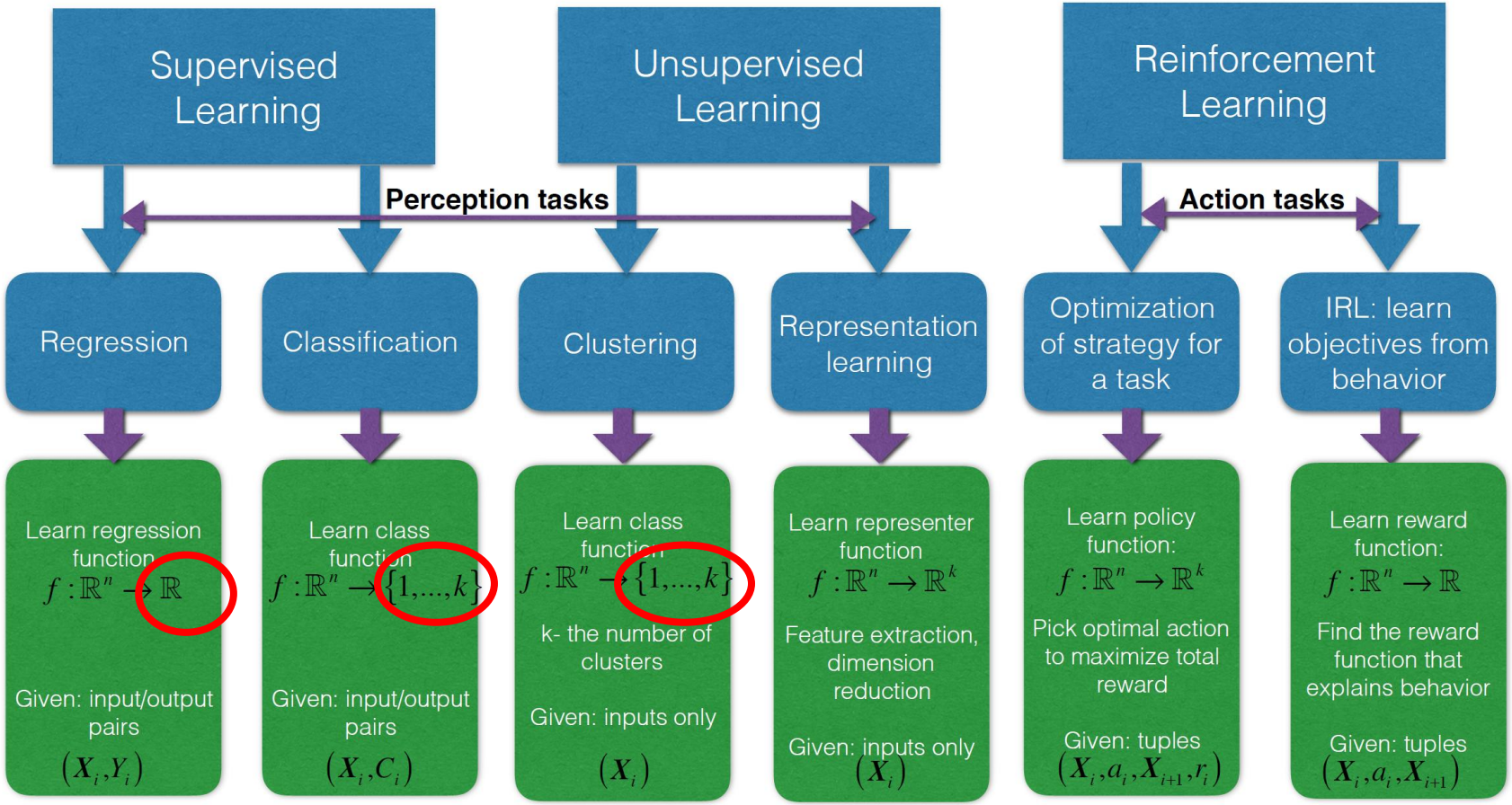
$$q_1 = f_{3,1}(z_1, z_2, z_3)$$
$$q_2 = f_{3,2}(z_1, z_2, z_3)$$
$$q_3 = f_{3,3}(z_1, z_2, z_3)$$

(3,1) **(3,2)** **(2,1)**

$$Z = W_z H$$

Fully connected $f_{2,2}(h_1, h_2) = w_1 h_1 + w_2 h_2 + b_{2,2}$

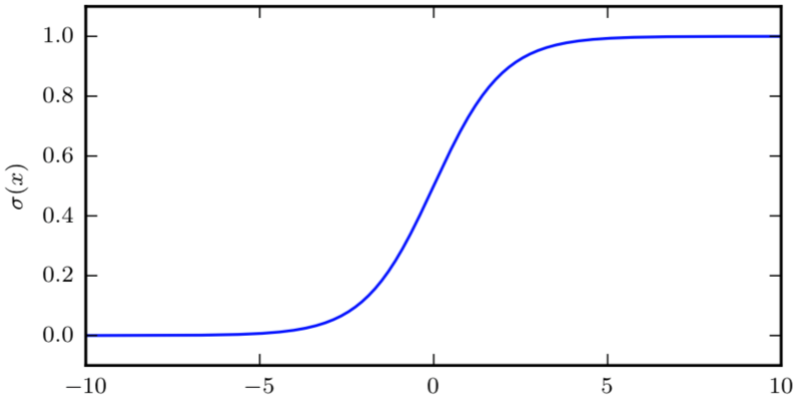
Neural Network Outputs



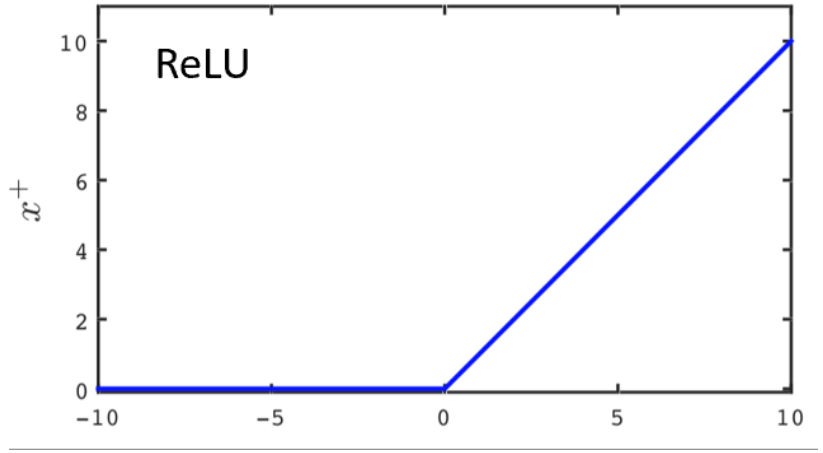
Output layer : activation functions

1) **Classification** : probability vector
   - Sigmoid (binary class)
   - Softmax (multiple class)

$$Z = \sigma(W_z H)$$



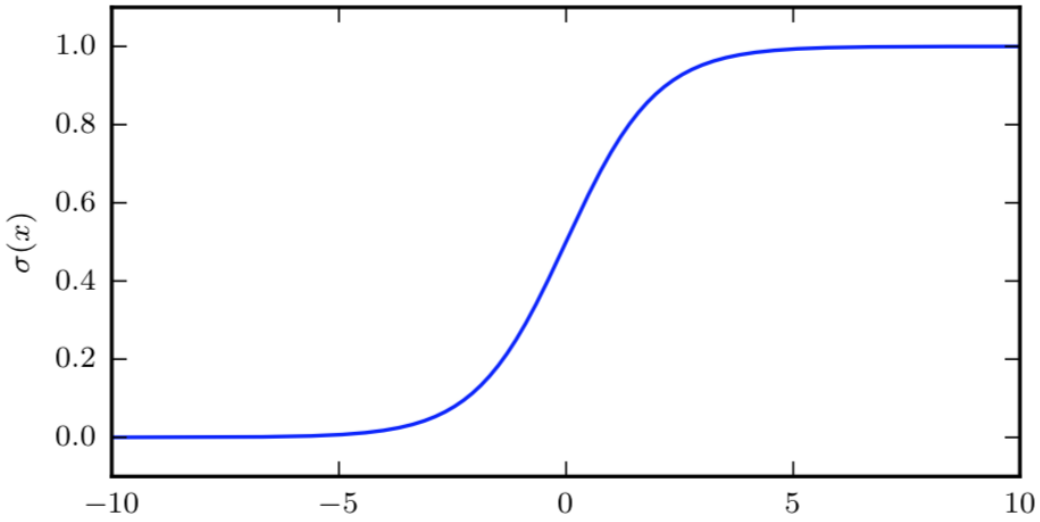
2) **Regression** : mean estimate
   - ReLU
   - Softplus
   - Smoothed max
   - Generalization of ReLU (leaky ReLU,…)



Sigmoid and softmax

Sigmoid (*two-class* classifier) :

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



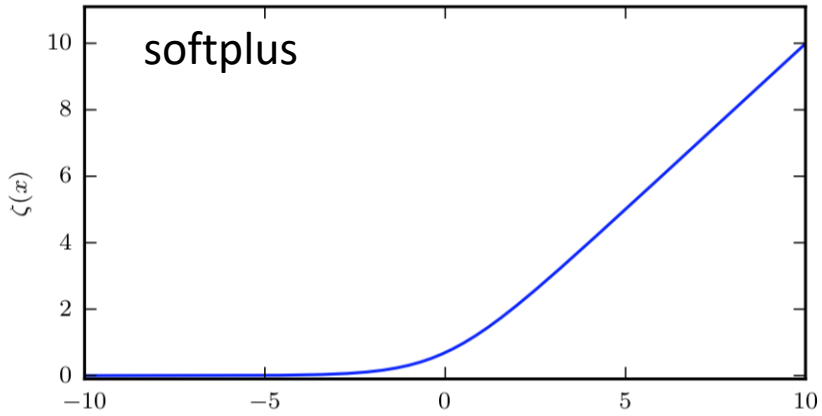
Softmax (*multi-class* classifier) :

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

ReLU, softplus and smoothed max

Softplus (smooth approx. of ReLU) :

$$\zeta(x) = \log(1 + \exp(x))$$

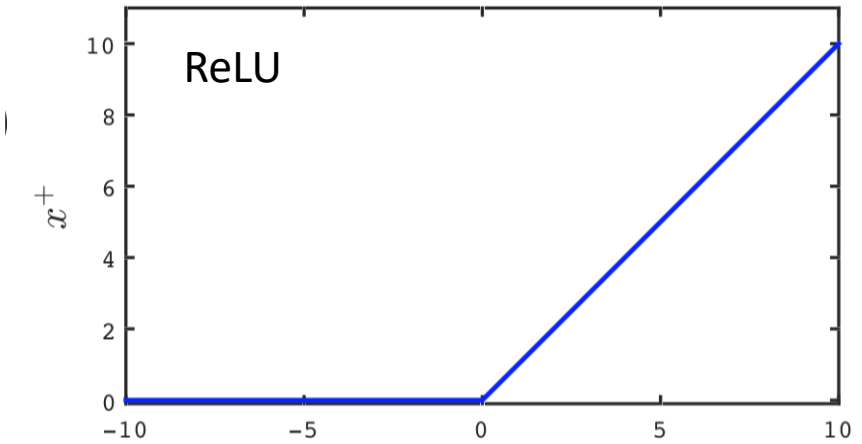


Smoothed max (*extension* of softplus) :

$$\zeta(x) = \log \sum_j \exp(x_i)$$

ReLU (Rectified Linear Unit) :

$$x^+ = \max(0, x)$$



2) Loss and Cost functions

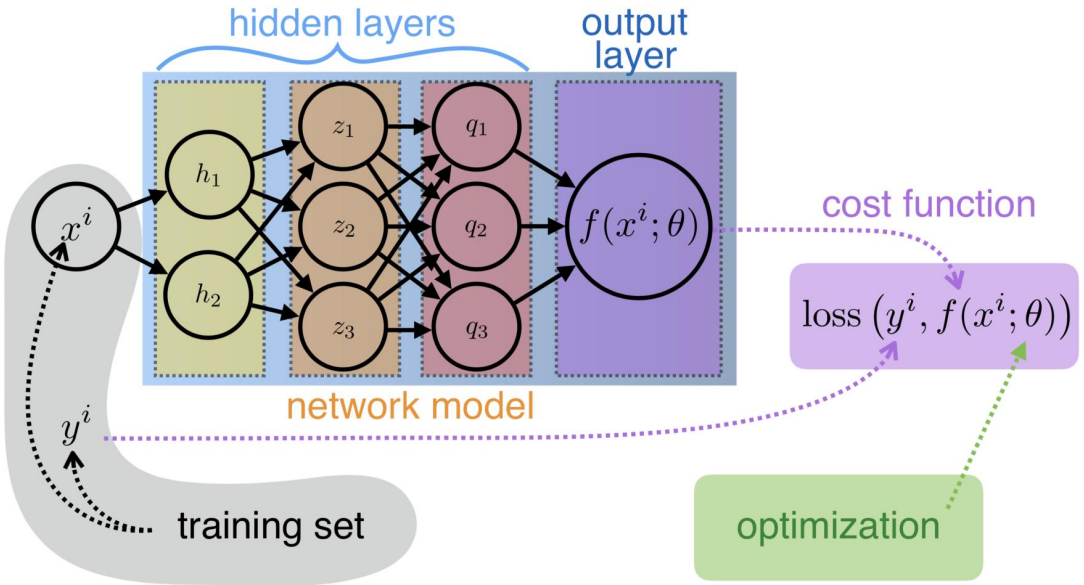
- Loss function $L(\hat{y}^{(i)}, y^{(i)})$, also called error function, measures how different the prediction $\hat{y} = f(x)$ and the desired output y are

- Cost function $J(w, b)$ is the average of the loss function on the *entire training set*

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

- Goal of the optimization is to find the *parameters* $\theta = (w, b)$ that minimize the cost function

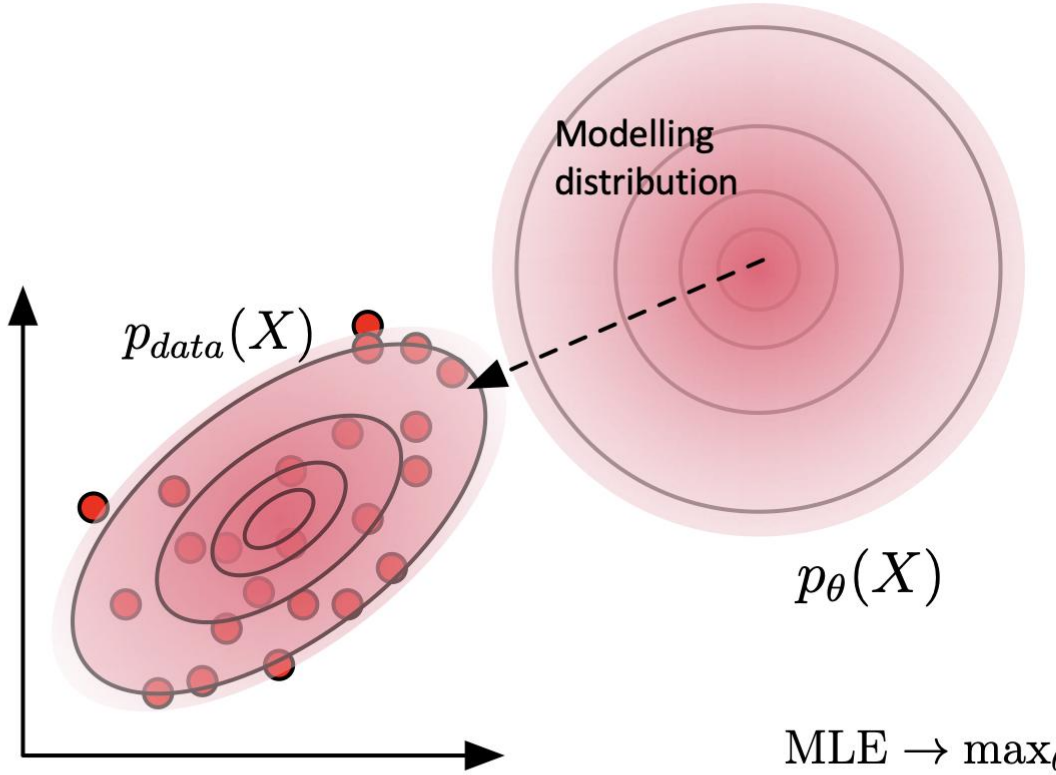
3) Optimization



- Given a task we define
  - Training data $\{x^i, y^i\}_{i=1,\ldots,m}$
  - Network $f(x;\theta)$
  - Cost function $J(\theta) = \sum_{i=1}^{m} \text{loss}\left(y^i, f(x^i;\theta)\right)$
  - Parameter initialization (weights, biases)
    - *random weights*, *biases initialized to small values (0 or 0.1)*

- Next, we *optimize the network parameters* θ (training)
- In addition, we have to set values for hyperparameters

Maximum Likelihood



$$\text{MLE} \rightarrow \max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x_i)$$

Fisher 1922

$$\min_{\theta \in \mathcal{M}} KL\left(P_{\text{data}}, P_\theta\right) = \mathbb{E}_{\mathbf{x} \sim P_{\text{data}}} \left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_\theta(\mathbf{x})}\right]$$

Loss function choice

- Choice determined by the output representation
  - Probability vector (**classification**): Cross-entropy

$$\hat{y} = \sigma(w^\top h + b) \qquad p(y|\hat{y}) = \hat{y}^y(1-\hat{y})^{(1-y)}$$

$$L(\hat{y}, y) = -\log p(y|\hat{y}) = -\big(y\ \log(\hat{y}) + (1-y)\log(1-\hat{y})\big)$$

(binary classification)

  - Mean estimate (**regression**): Mean Squared Error, L2 loss

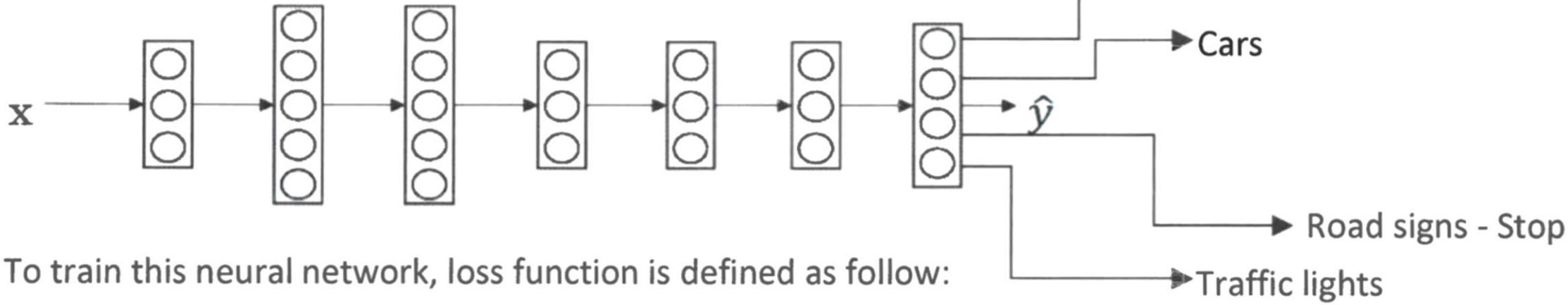
$$\hat{y} = W^\top h + b \qquad p(y|\hat{y}) = \mathrm{N}(y; \hat{y})$$

$$L_2(\hat{y}, y) = -\log p(y|\hat{y}) = \sum_{i=0}^{m} \big(y^i - \hat{y}^i\big)^2$$

Loss function example

- NN does simultaneously several tasks (multi-task)

Neural Network architecture

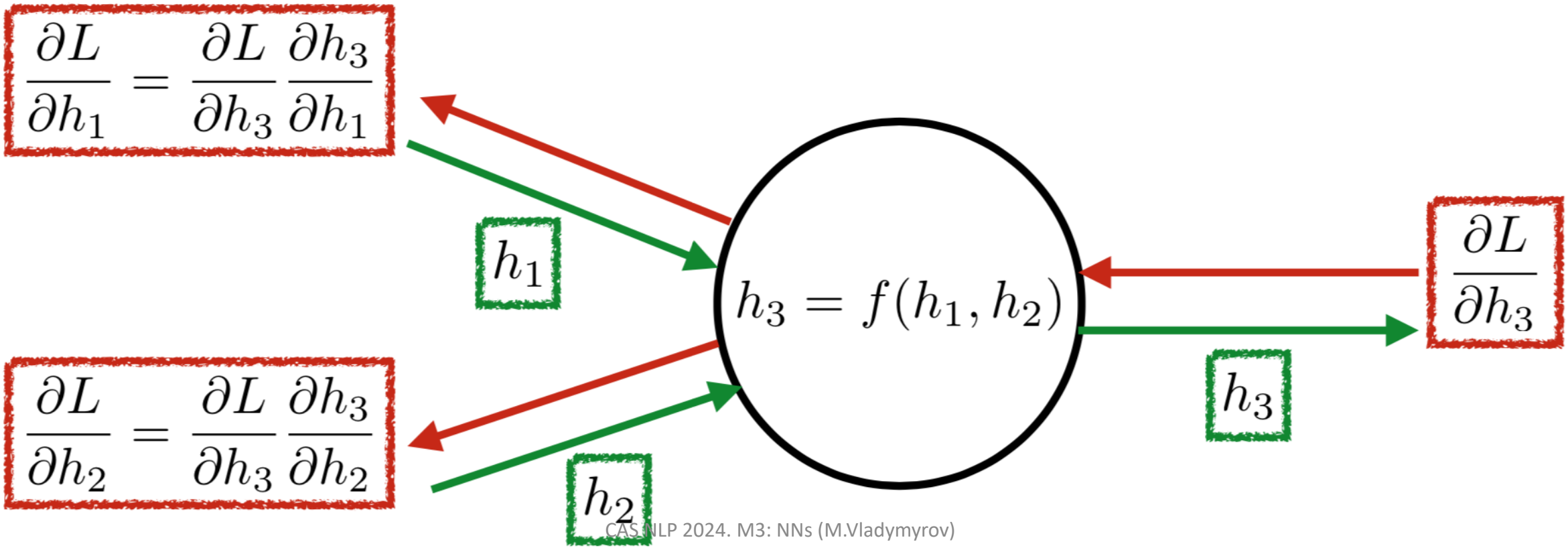


To train this neural network, loss function is defined as follow:

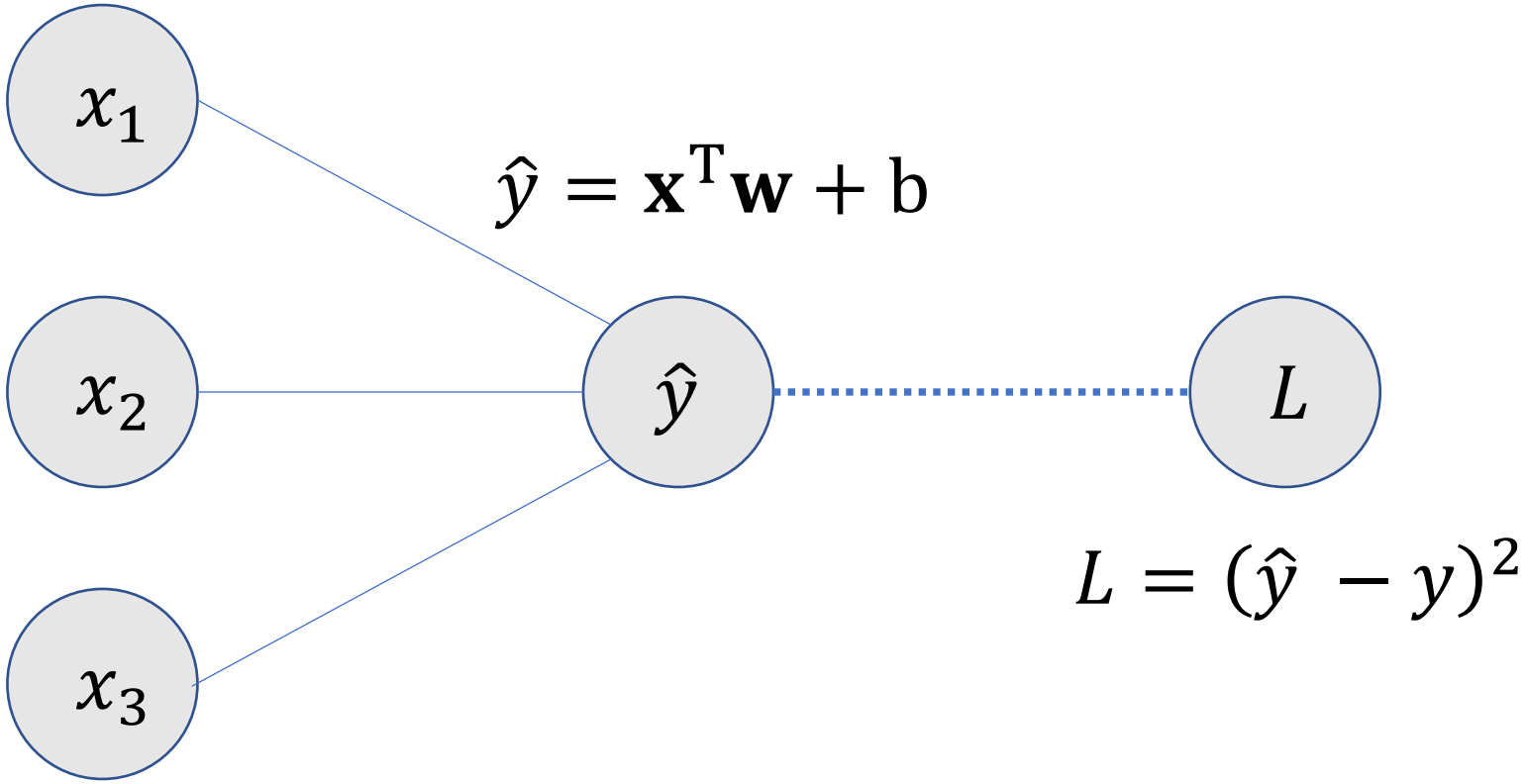
$$-\frac{1}{m}\sum_{i=1}^{m}\boxed{\sum_{j=1}^{4}}\left(y_j^{(i)}\log\left(\hat{y}_j^{(i)}\right)+\left(1-y_j^{(i)}\right)\log\left(1-\hat{y}_j^{(i)}\right)\right)$$

Backpropagation

- Efficient implementation of the chain-rule to compute derivatives with respect to network weights



Example


x_1
x_2
x_3
$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$
$\hat{y}$
L
$L = (\hat{y} - y)^2$

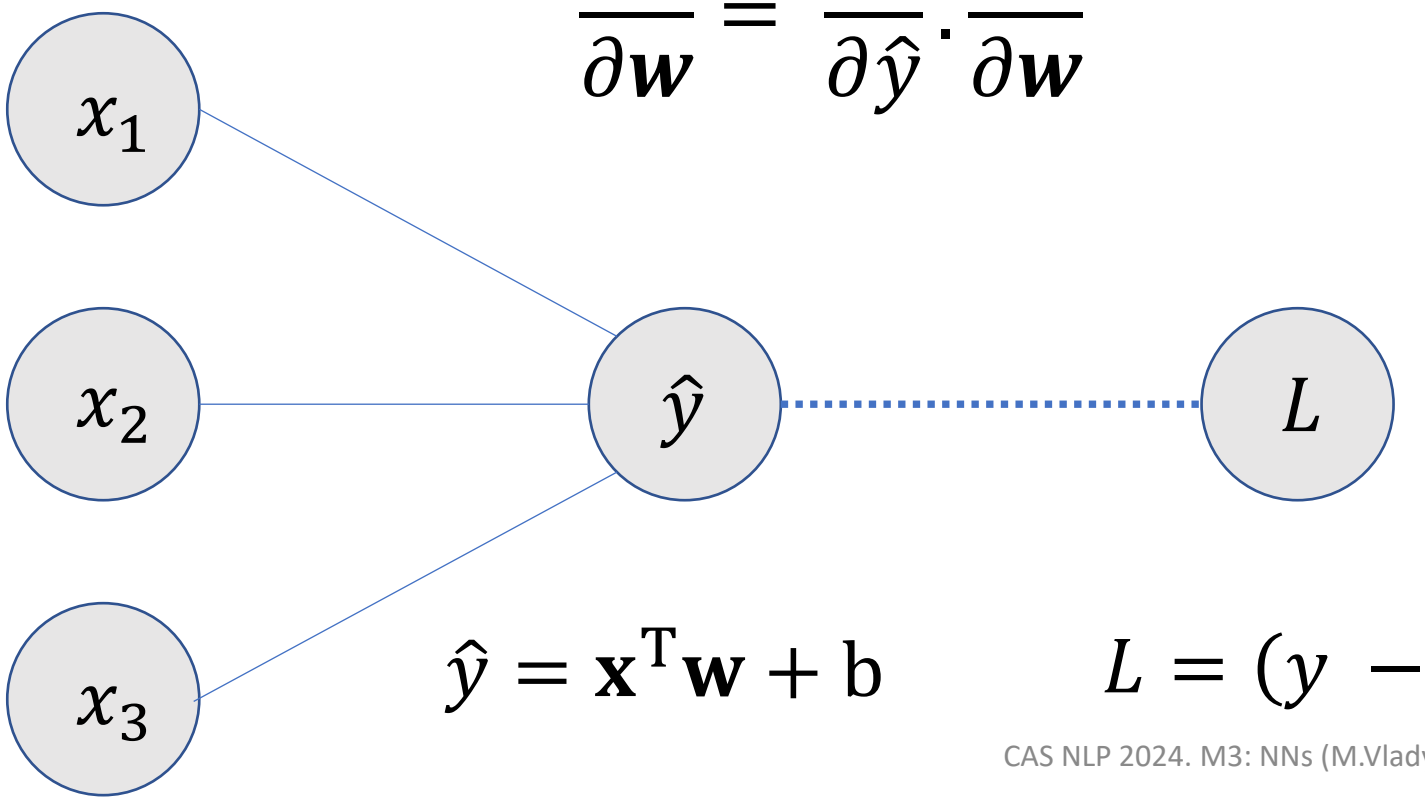
Example

We need to calculate the gradients:

Let's start with this part!

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b}$$



$$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$$

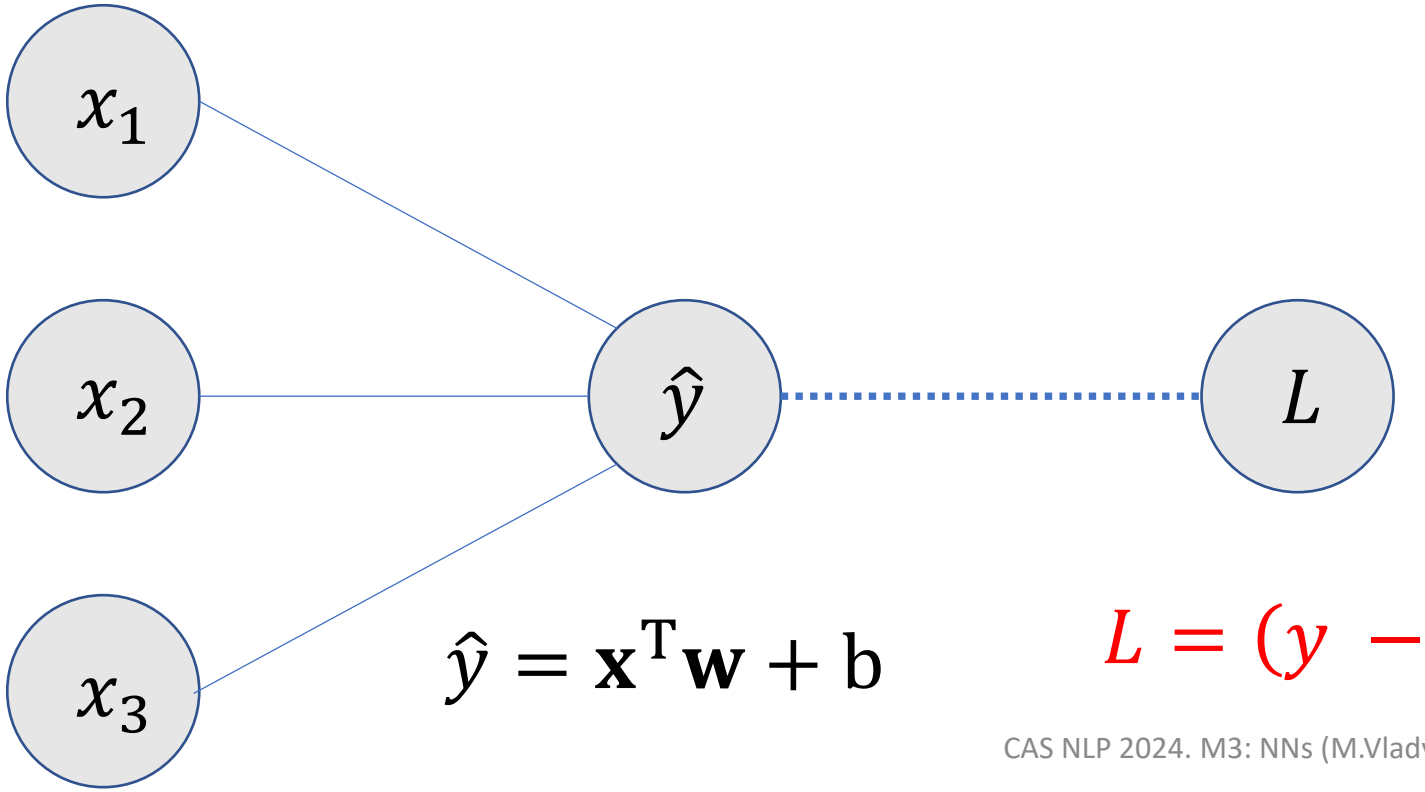
$$L = (y - \hat{y})^2$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

Example

First:

$$\frac{\partial L}{\partial \hat{y}} = -2y - 2\hat{y} = 2(\hat{y} - \mathrm{y})$$



$$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$$

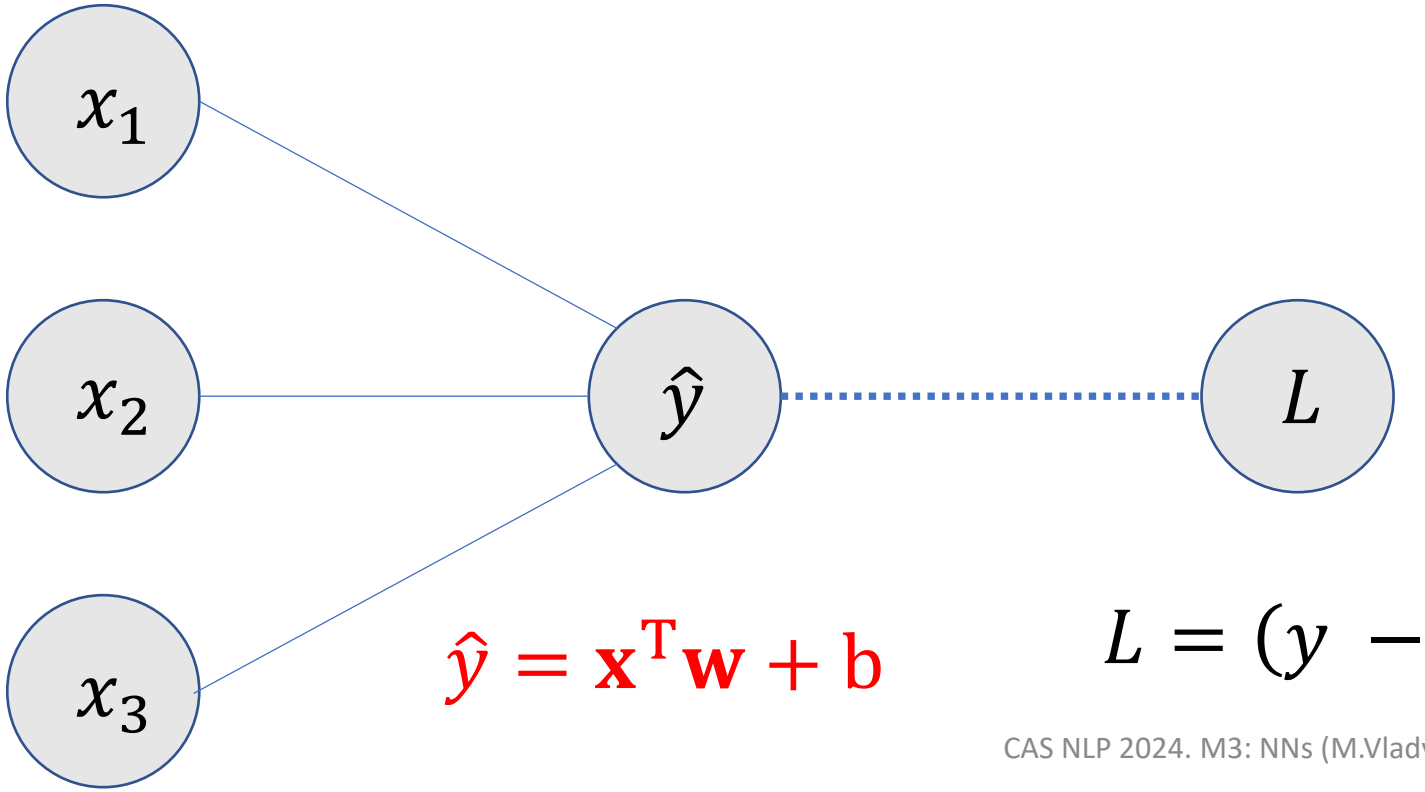
$$L = (y - \hat{y})^2 = \mathrm{y}^2 - 2\mathrm{y}\hat{y} + \hat{y}^2$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

$$\frac{\partial L}{\partial \hat{y}} = -2y\ - 2\hat{y} = 2(\hat{y} - \mathrm{y})$$

Example

Second: $$\frac{\partial}{\partial \boldsymbol{w}}(\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = \boldsymbol{x}^{T} . \frac{\partial}{\partial \boldsymbol{w}}(\mathbf{w}) = \boldsymbol{x}^{T}$$



$$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$$

$$L = (y\ -\ \hat{y})^2$$

Example

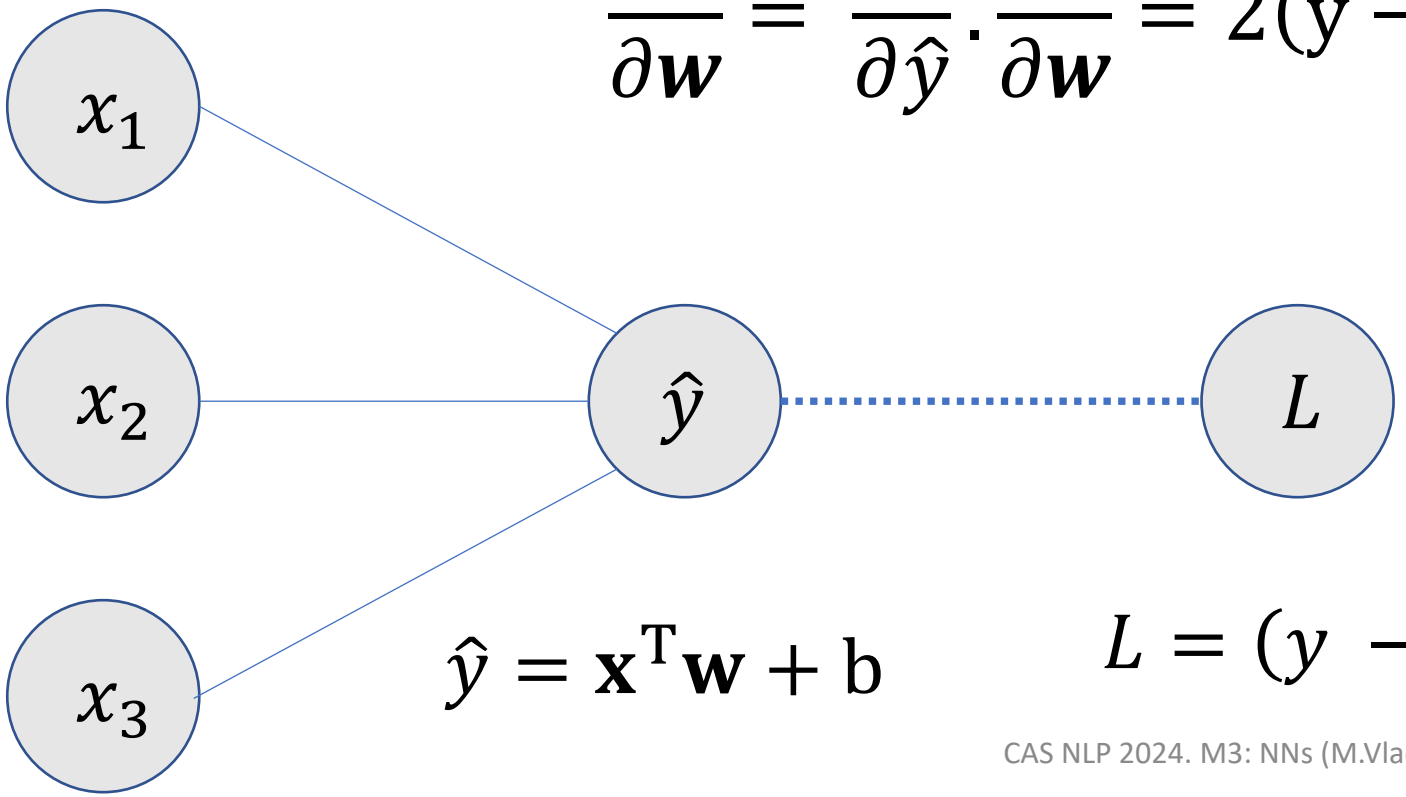
$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

$$\frac{\partial L}{\partial \hat{y}} = -2y \ - 2\hat{y} = 2(\hat{y} - \mathrm{y})$$

$$\frac{\partial}{\partial \boldsymbol{w}} (\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = \boldsymbol{x}^{T}$$

Putting these together:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}} = 2(\mathrm{y} - \hat{y}) . \boldsymbol{x}^{T}$$



$$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$$

$$L = (y \ - \ \hat{y})^2$$

Example

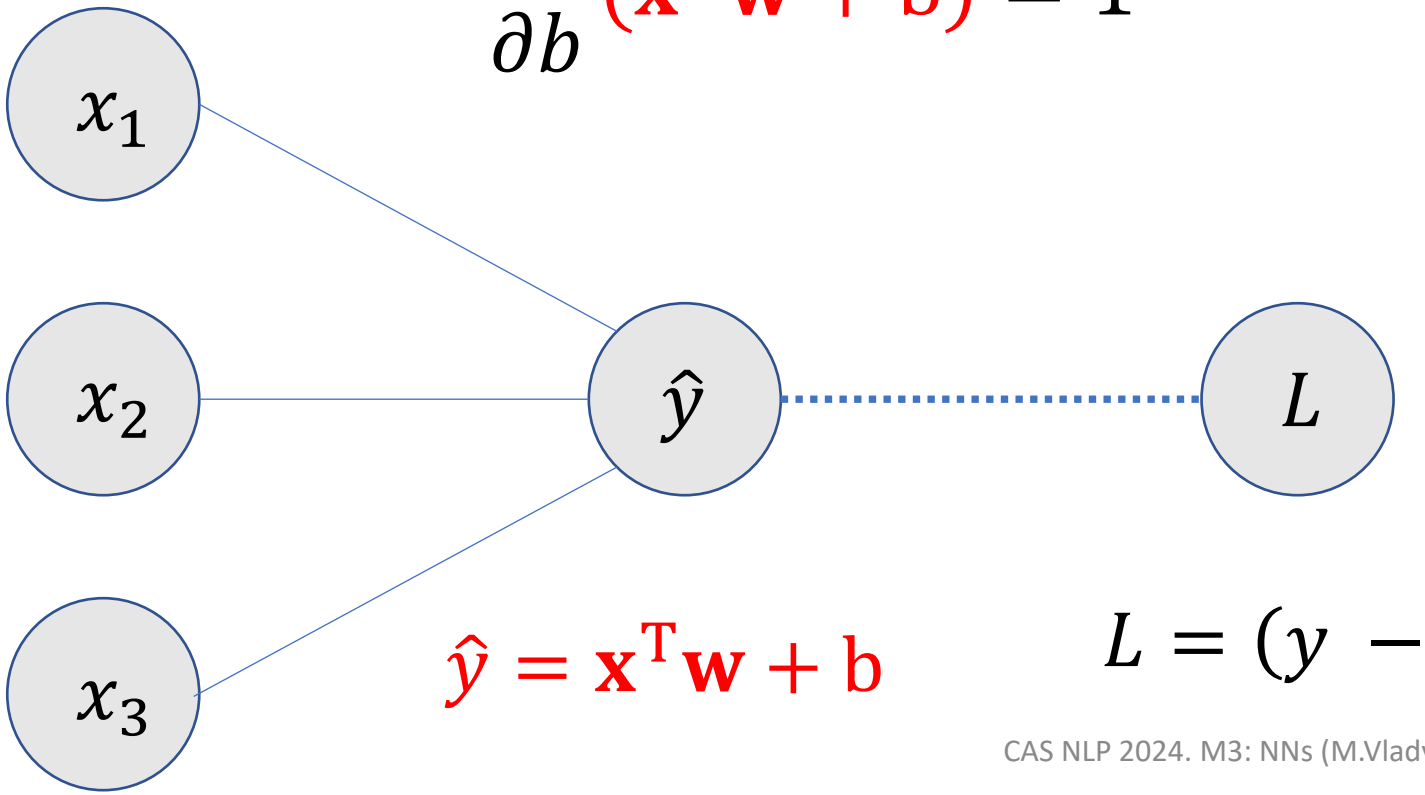
$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

$$\frac{\partial L}{\partial \hat{y}} = -2y\ - 2\hat{y} = 2(\hat{y} - \mathrm{y})$$

$$\frac{\partial}{\partial \boldsymbol{w}}(\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = \boldsymbol{x}^{T}$$

Now for the bias…

$$\frac{\partial}{\partial b}(\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = 1$$



$$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$$

$$L = (y\ -\ \hat{y})^2$$

Example

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

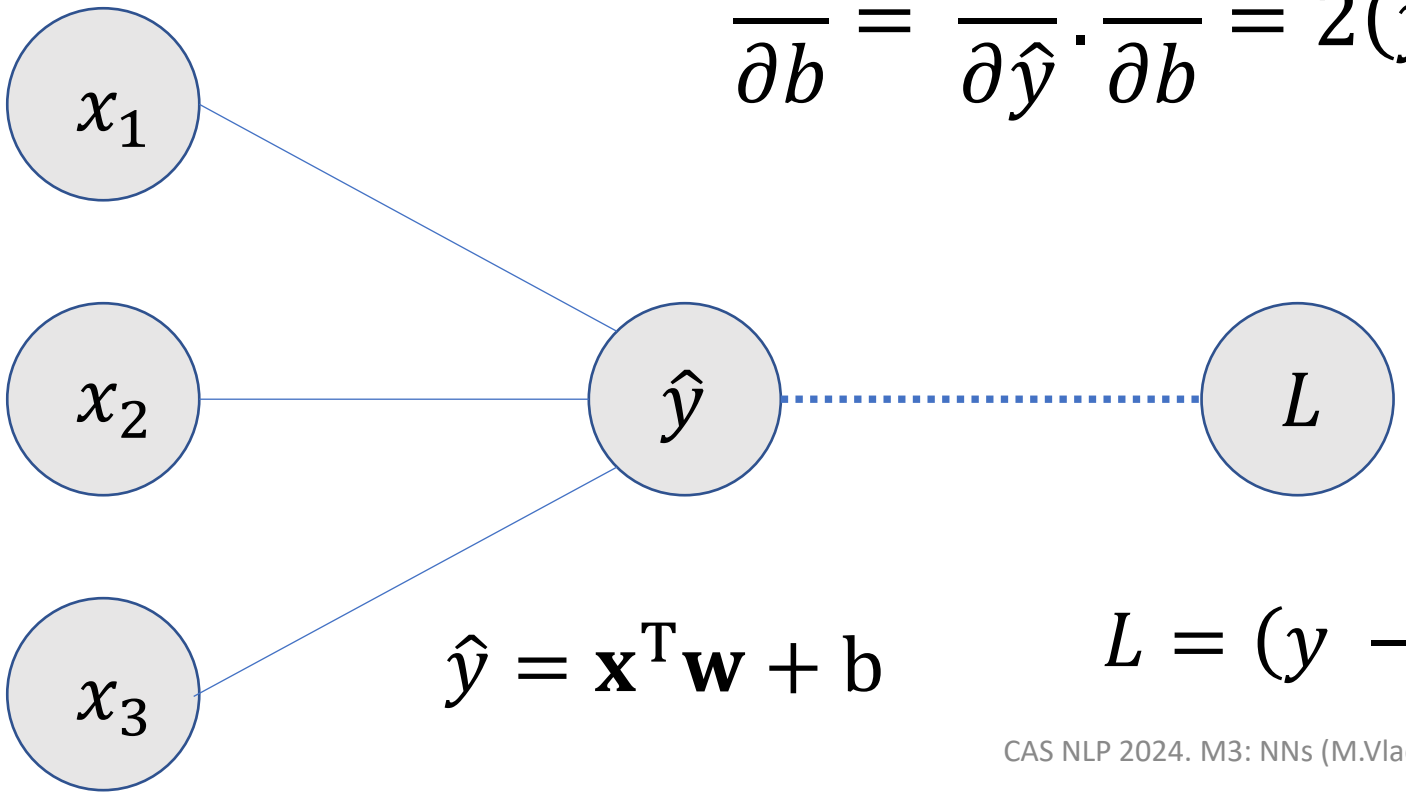
$$\frac{\partial L}{\partial \hat{y}} = -2y \ - 2\hat{y} = 2(\hat{y} - \mathrm{y})$$

$$\frac{\partial}{\partial \boldsymbol{w}} (\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = \boldsymbol{x}^{T}$$

$$\frac{\partial}{\partial b} (\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = 1$$

Putting these together:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial b} = 2(\hat{y} - y).1$$



$$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$$

$$L = (y \ - \ \hat{y})^2$$

Example

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial L}{\partial \hat{y}} . \frac{\partial \hat{y}}{\partial \boldsymbol{w}}$$

$$\frac{\partial L}{\partial \hat{y}} = -2y - 2\hat{y} = 2(\hat{y} - \mathrm{y})$$

$$\frac{\partial}{\partial \boldsymbol{w}}(\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = \boldsymbol{x}^{T}$$

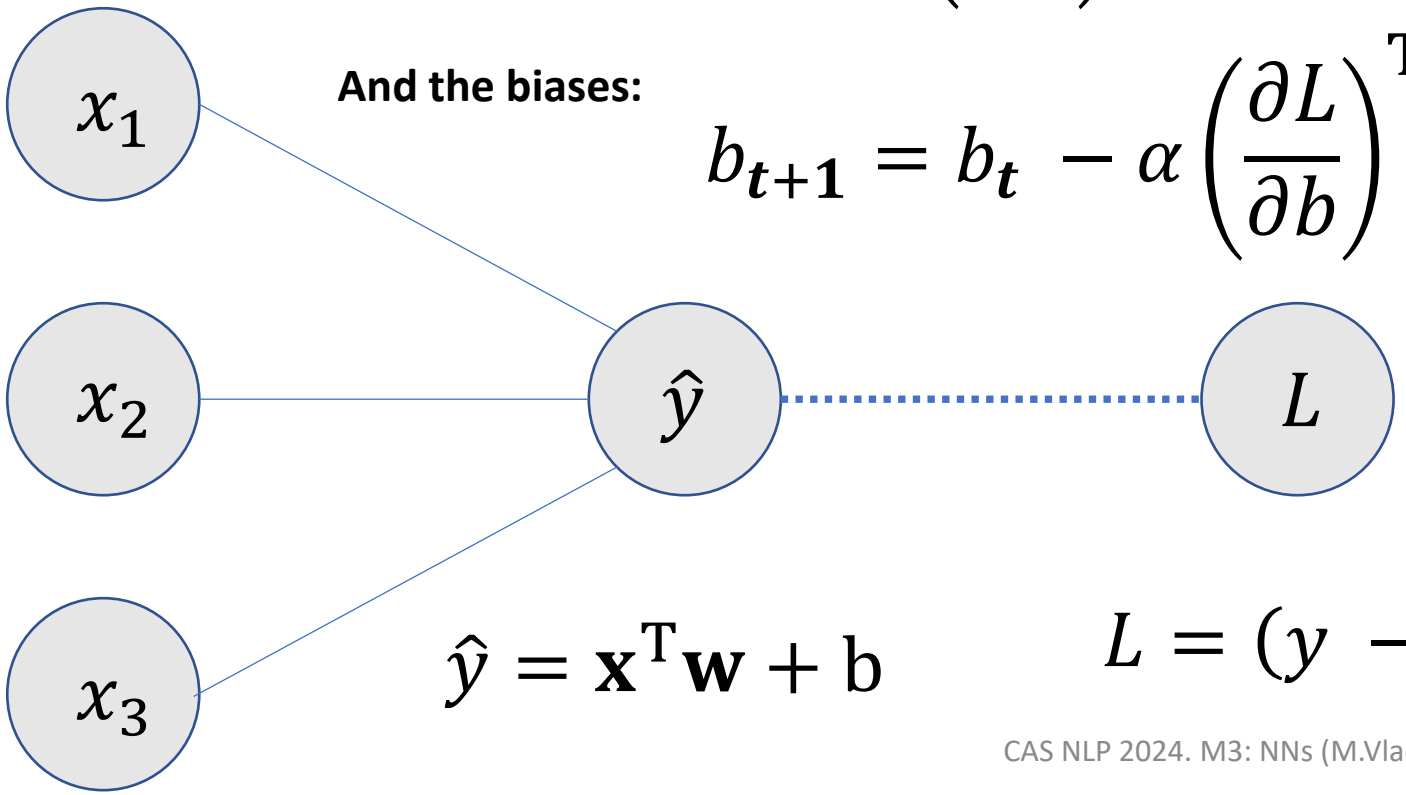
$$\frac{\partial}{\partial b}(\mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}) = 1$$

Finally the updates for the weights:

$$\boldsymbol{w_{t+1}} = \boldsymbol{w_t} - \alpha \left(\frac{\partial L}{\partial \boldsymbol{w}}\right)^{\mathrm{T}} = \boldsymbol{w_t} - 2\alpha(\hat{y} - y)\boldsymbol{x}$$

And the biases:

$$b_{\boldsymbol{t+1}} = b_{\boldsymbol{t}} - \alpha \left(\frac{\partial L}{\partial b}\right)^{\mathrm{T}} = b_{\boldsymbol{t}} - 2\alpha(\hat{y} - y)$$



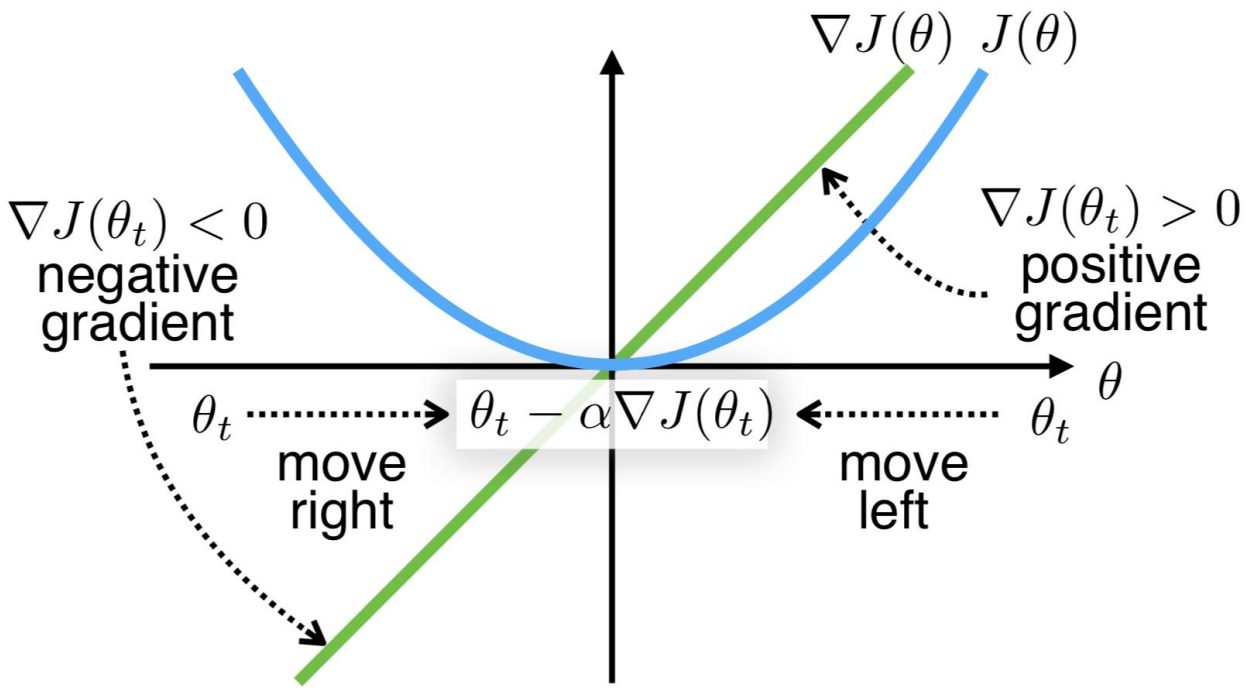
$$\hat{y} = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \mathrm{b}$$

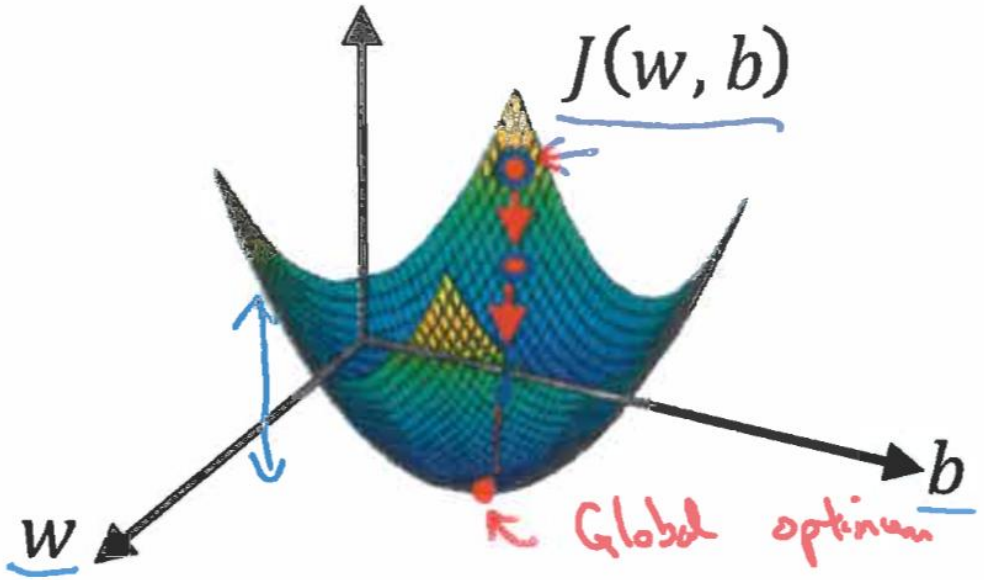
$$L = (y - \hat{y})^2$$

Gradient Descent

- Iterative method to find the parameters $\theta = (w, b)$ that minimize $J(\theta)$

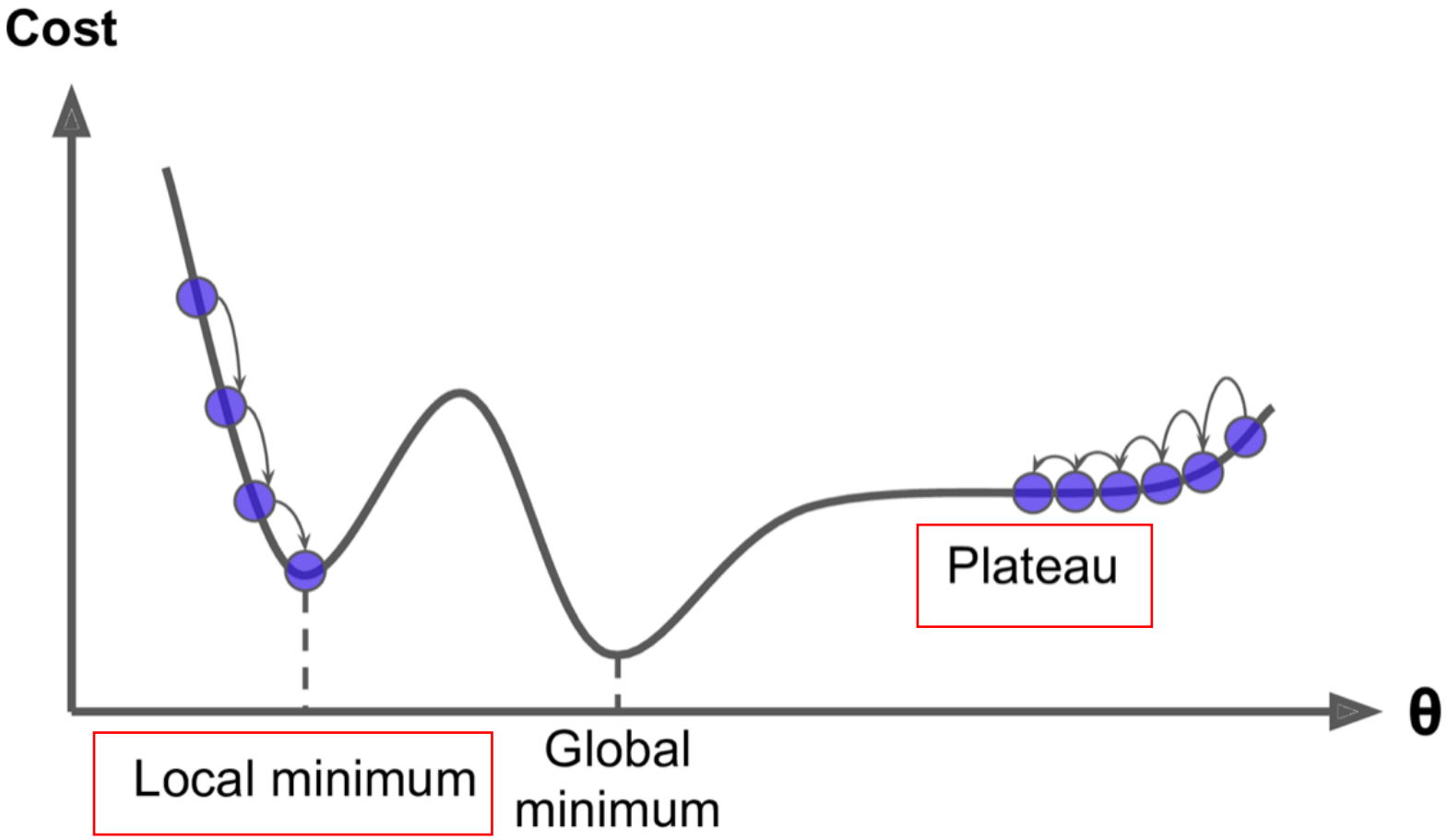
gradient descent $\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$



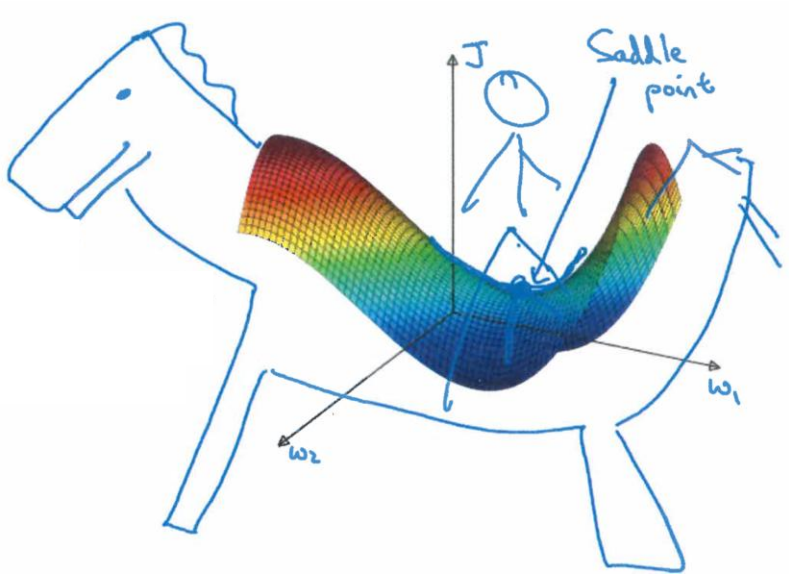


$$\nabla J(w) = \frac{dJ(w,b)}{dw} \qquad \nabla J(b) = \frac{dJ(w,b)}{db}$$

Optimization pitfalls



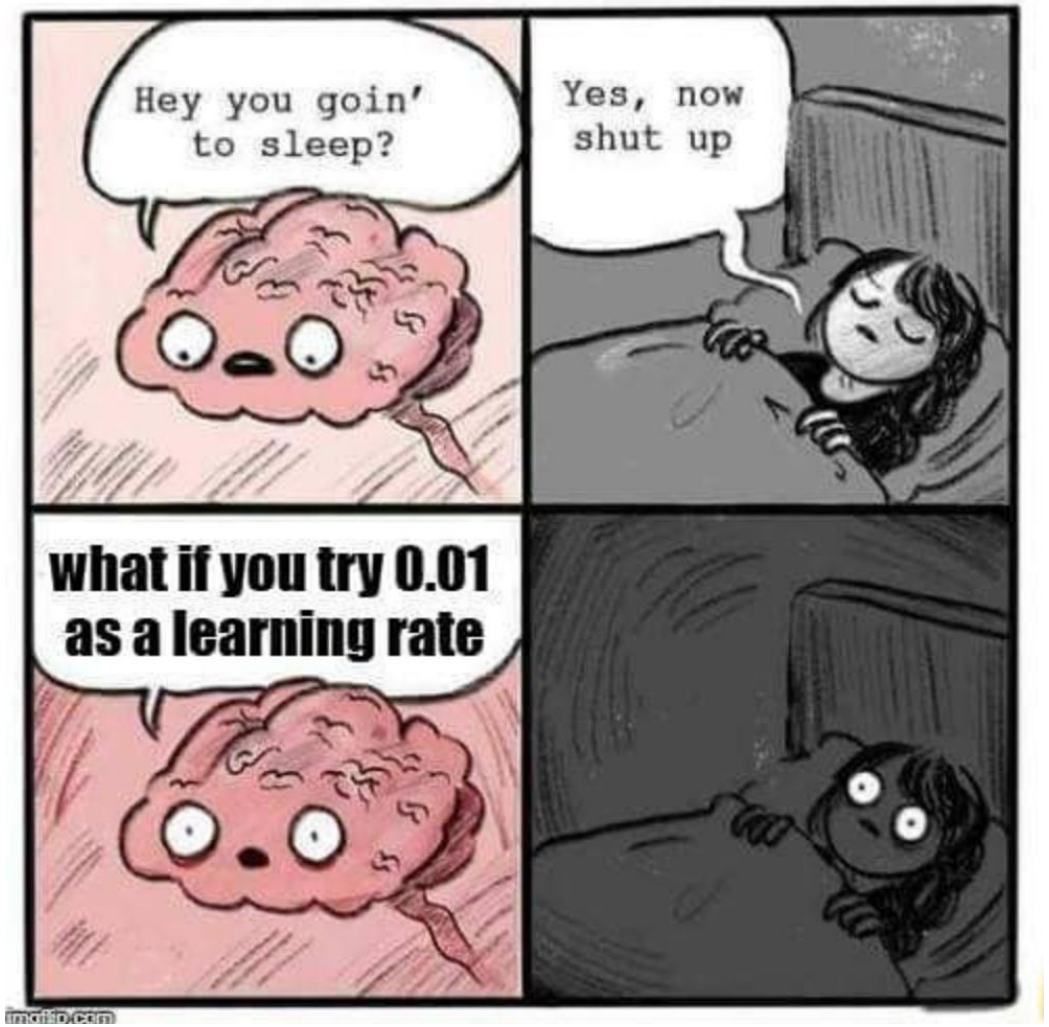
Saddle point



Hyperparameters

- Parameters that cannot be learnt directly from training data

- A long list…
  - Learning rate α
  - Number of iterations (epochs)
  - Number of hidden layers
  - Number of hidden units
  - Choice of activation function



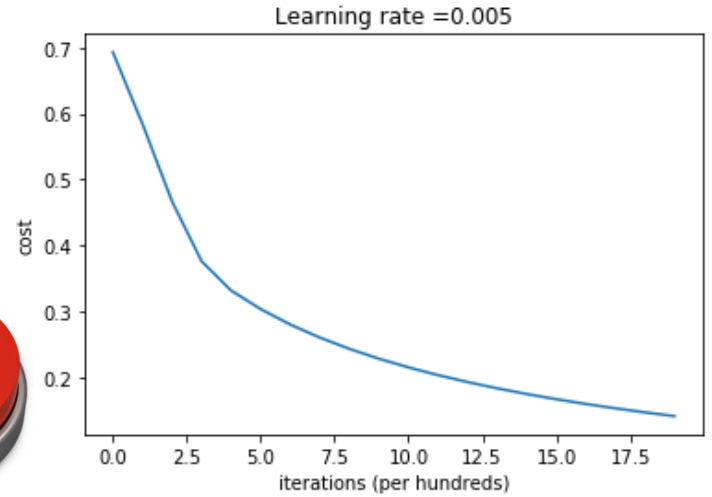
Training

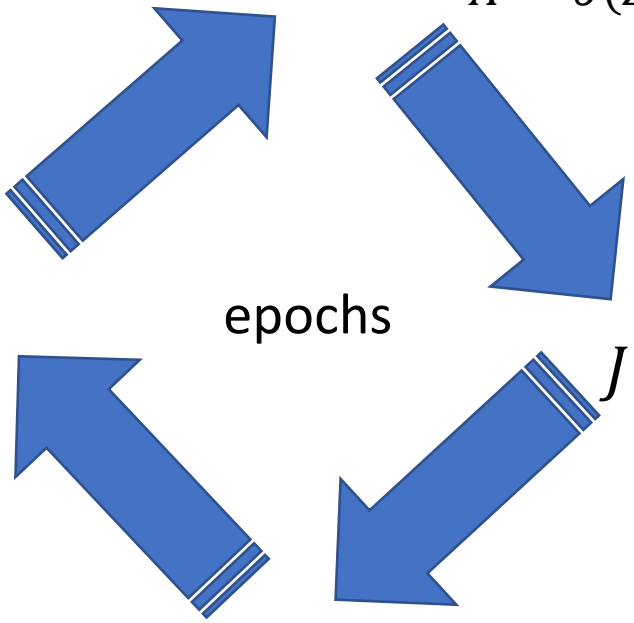
- *Iterative* process

Forward propagation $Z = w^T x + b$

$A = \sigma(Z)$

Learning curve





Parameter update (gradient descent)

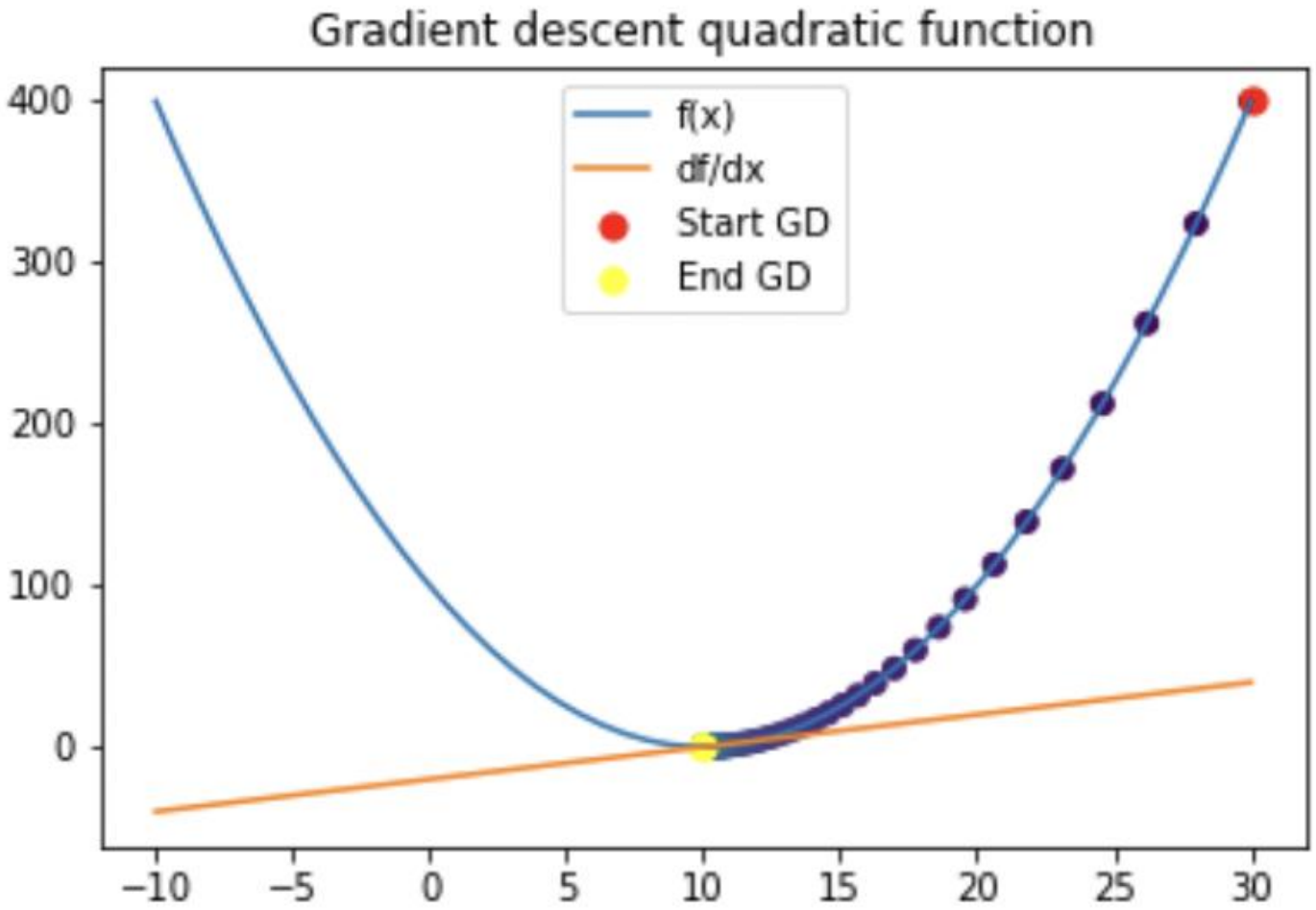
epochs

Cost function $J(w, b) = J(\theta)$

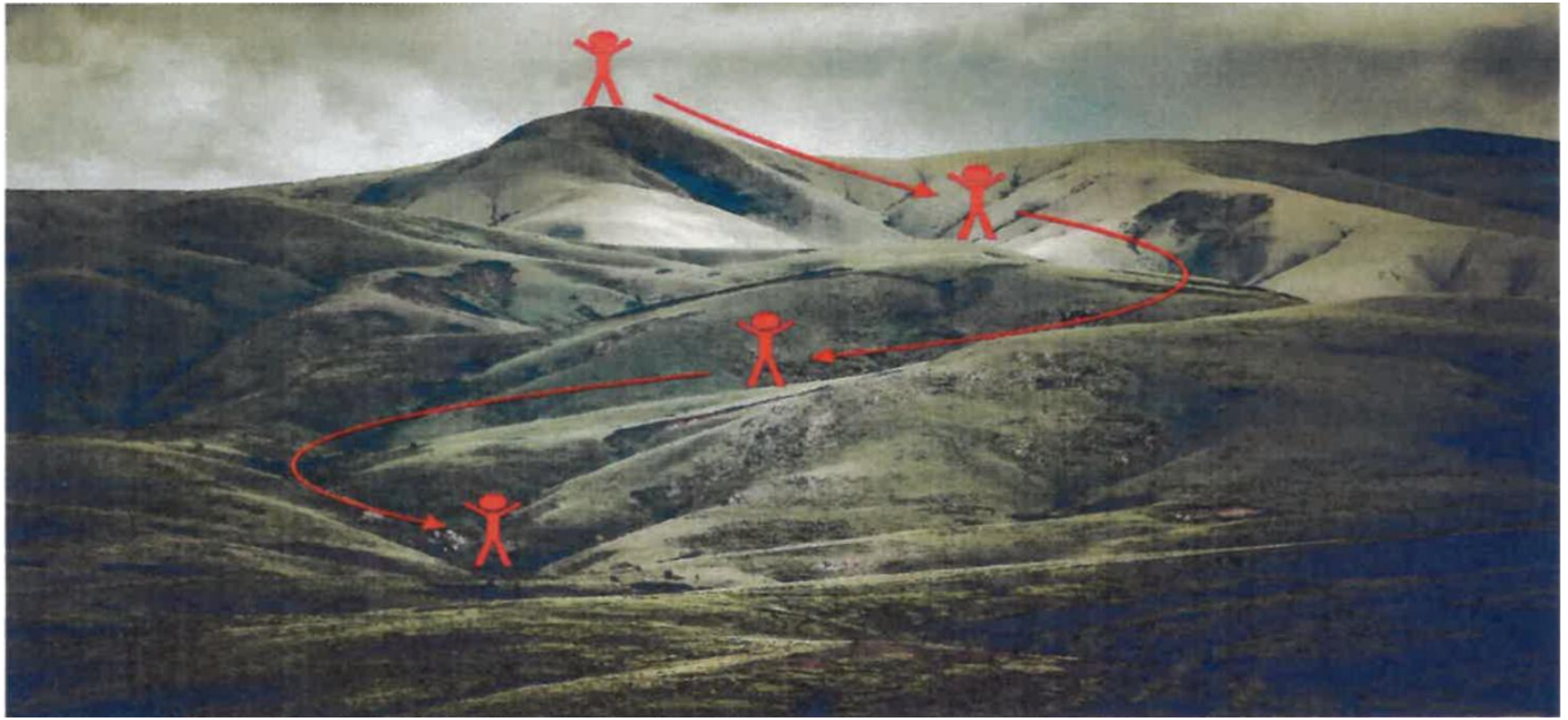
learning rate α

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$$

Backward propagation (dJ/dw, dJ/db)


Gradient descent quadratic function
f(x)
df/dx
Start GD
End GD
400
300
200
100
0
−10
−5
0
5
10
15
20
25
30

Gradient Descent Illustration



Summary

- Neural Networks are an efficient generic fitter
- Training is performed iteratively by adjusting trainable parameters
- The model's architecture is built to
  - Match model input to the data shape
  - Output to labels shape and range
  - The model in between is shaped to optimize performance