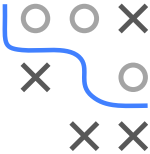
Resources

Frankenstein Monster is created from

1. Mostly taken from LMU Munich's SLDS I2ML Website
2. Some things are also taken from FAST Foundation's slides. YouTube

Overview

Introduction to Machine Learning

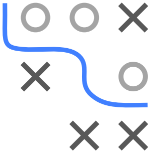


ML-Basics
What is Machine Learning?

Learning goals

- Understand basic terminology of and connections between ML, AI, DL and statistics
- Know the main directions of ML: Supervised, Unsupervised and Reinforcement Learning

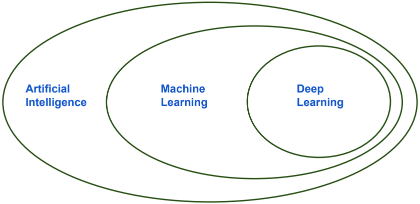
MACHINE LEARNING IS CHANGING OUR WORLD

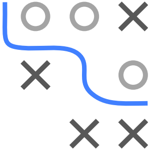


- Search engines learn what you want
- Recommender systems learn your taste in books, music, movies,...
- Algorithms do automatic stock trading
- Google Translate learns how to translate text
- Siri learns to understand speech
- DeepMind beats humans at Go
- Cars drive themselves
- Smart-watches monitor your health
- Election campaigns use algorithmically targeted ads to influence voters
- Data-driven discoveries are made in physics, biology, genetics, astronomy, chemistry, neurology,...
- ...

THE WORLD OF ARTIFICIAL INTELLIGENCE

... and the connections to Machine Learning and Deep Learning

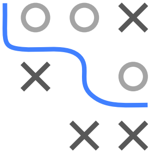




Many people are confused what these terms actually mean.

And what does all this have to do with statistics?

ARTIFICIAL INTELLIGENCE



- AI is a general term for a very large and rapidly developing field.
- There is no strict definition of AI, but it's often used when machines are trained to perform on tasks which until that time could only be solved by humans or are very difficult and assumed to require "intelligence".
- AI started in the 1940s - when the computer was invented. Scientists like Turing and John von Neumann immediately asked the question: If we can formalize computation, can we use computation to formalize "thinking"?
- AI includes machine learning, natural language processing, computer vision, robotics, planning, search, game playing, intelligent agents, and much more.
- Nowadays, AI is a "hype" term that many people use when they should probably say: ML or ... basic data analysis.

MACHINE LEARNING

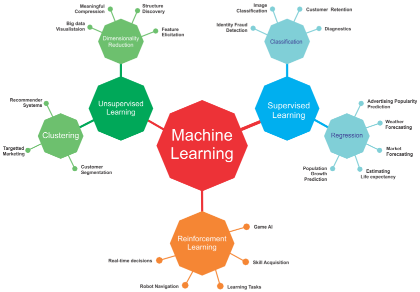
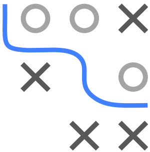
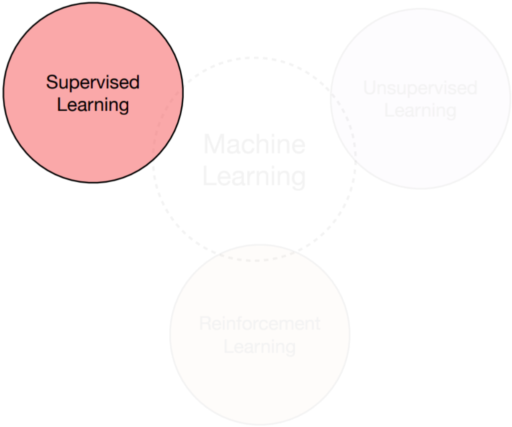
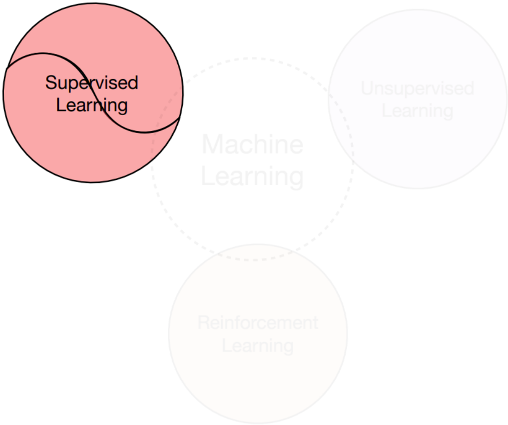


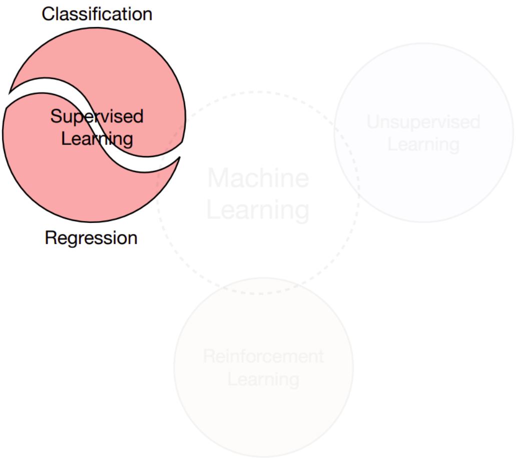
Image via `https://www.oreilly.com/library/view/java-deep-learning/9781788997454/assets/899ceaf3-c710-4675-ae99-33c76cd6ac2f.png`

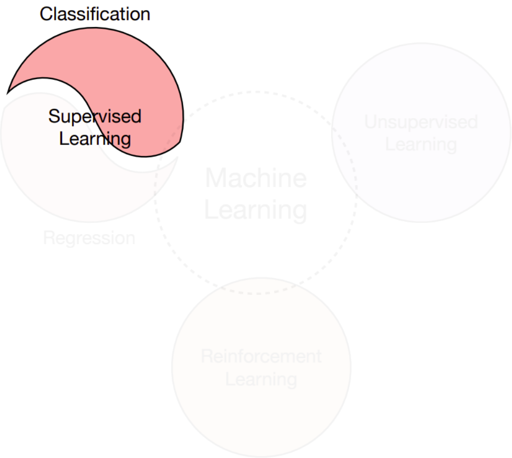


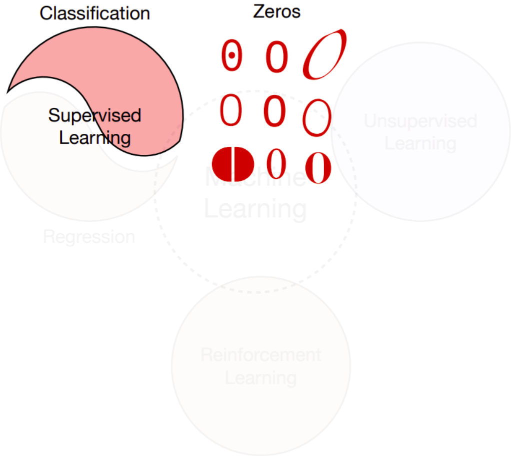
- Mathematically well-defined and solves reasonably narrow tasks.
- ML algorithms usually construct predictive/decision models from data, instead of explicitly programming them.
- A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
  Tom Mitchell, Carnegie Mellon University, 1998

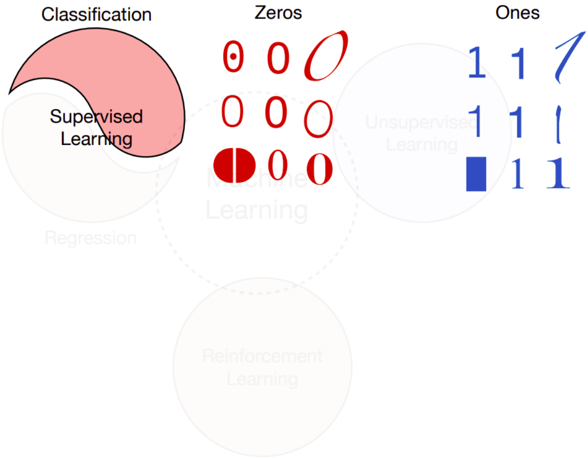

Supervised Learning
Machine Learning
Unsupervised Learning
Reinforcement Learning

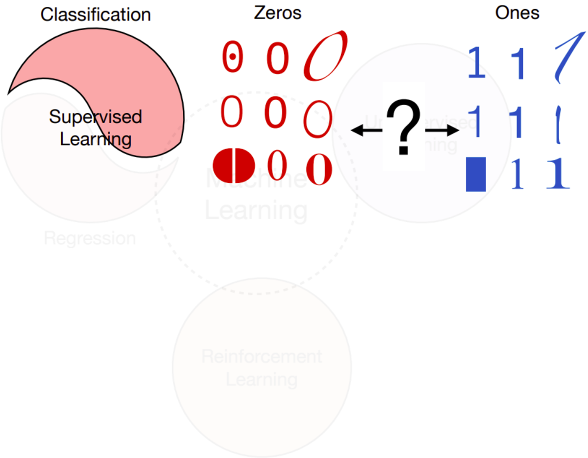

Supervised Learning
Machine Learning
Unsupervised Learning
Reinforcement Learning


Classification
Supervised Learning
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning


Classification
Supervised
Learning
Machine
Learning
Unsupervised
Learning
Regression
Reinforcement
Learning

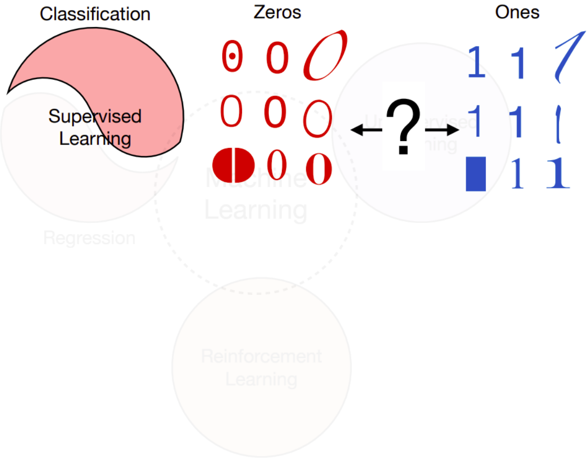

Classification
Zeros
Supervised Learning
Unsupervised Learning
Machine Learning
Regression
Reinforcement Learning

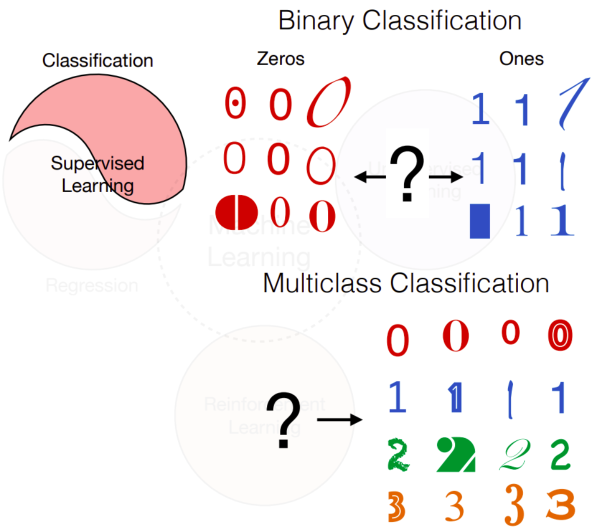

Classification
Supervised Learning
Zeros
Ones
Unsupervised Learning
Machine Learning
Regression
Reinforcement Learning

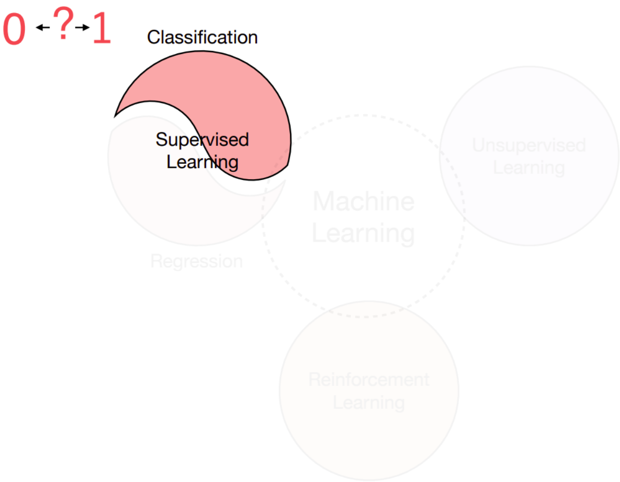


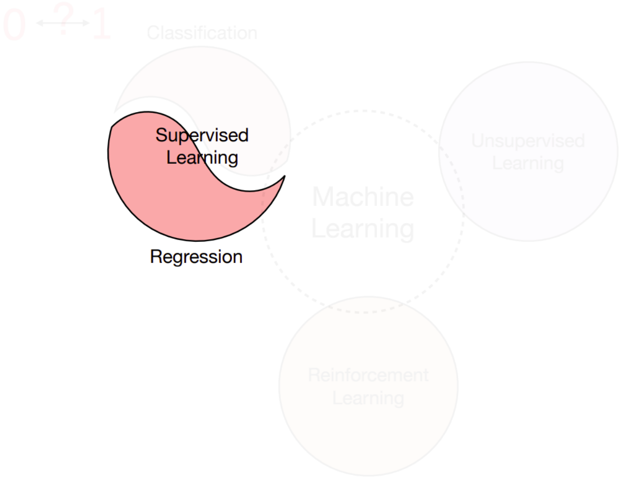
Classification
Supervised
Learning
Zeros
Ones
?
Machine
Learning
Regression
Reinforcement
Learning

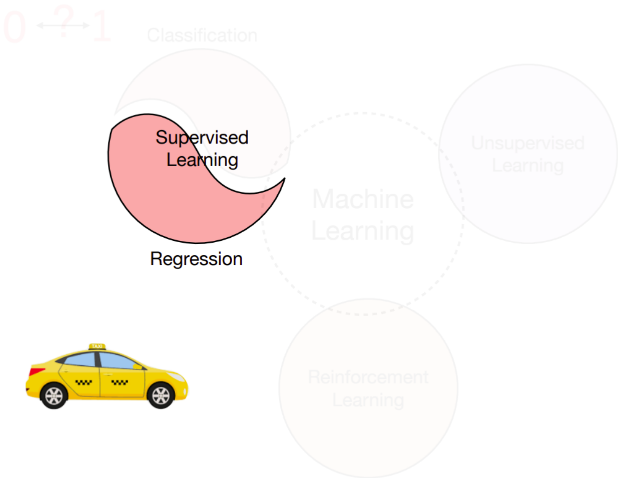
Binary Classification

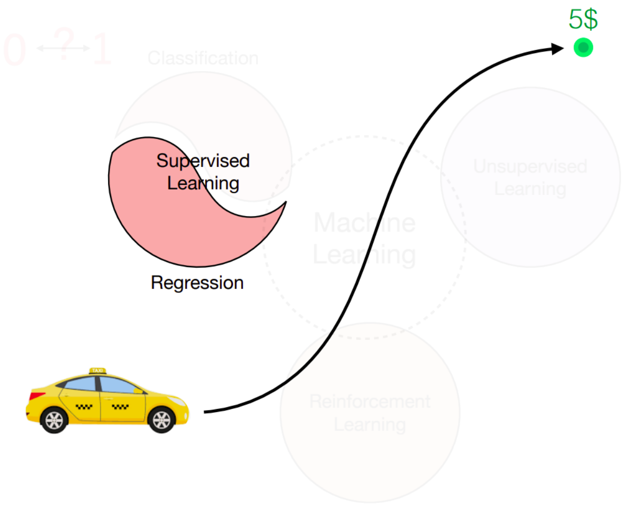


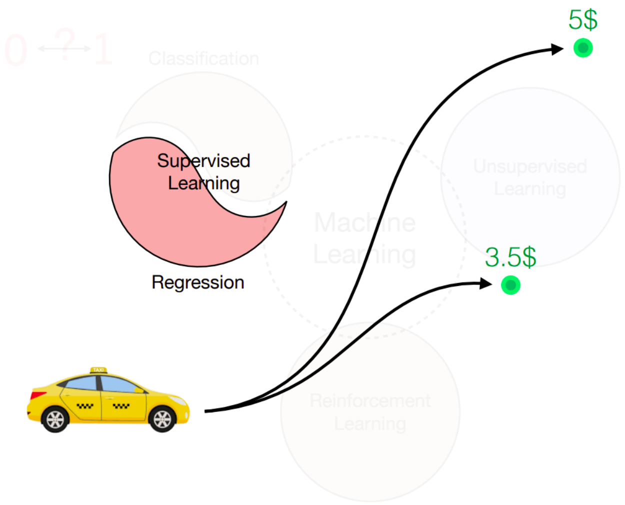

Classification
Supervised
Learning
Regression
Machine
Learning
Reinforcement
Learning
Binary Classification
Zeros
Ones
?
Multiclass Classification
?

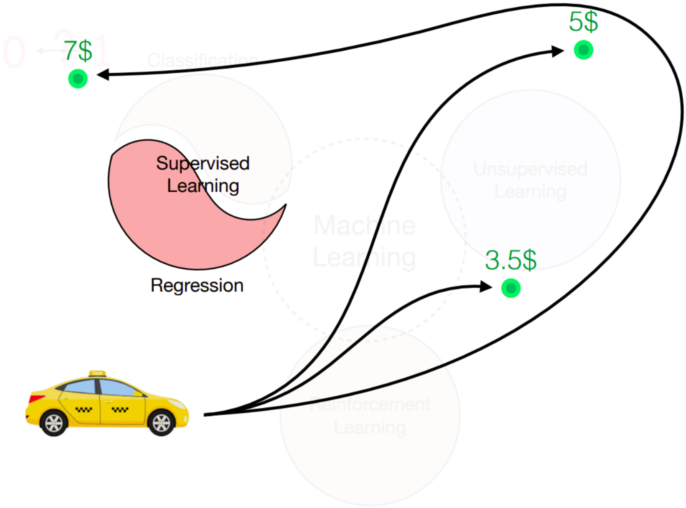

0 ? 1
Classification
Supervised Learning
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning

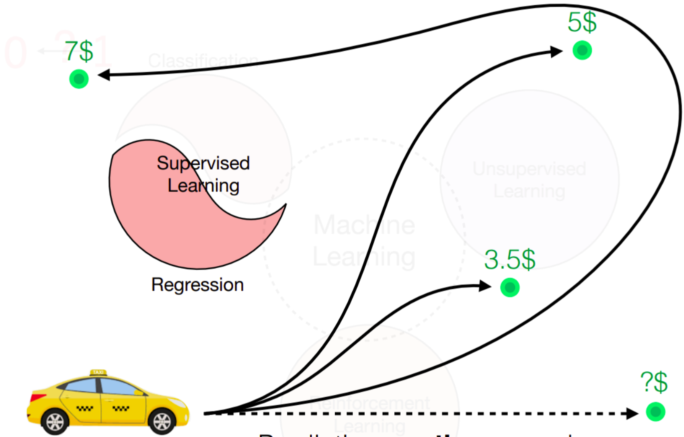

0 ? 1
Classification
Supervised Learning
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning


0 ? 1
Classification
Supervised Learning
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning
TAXI

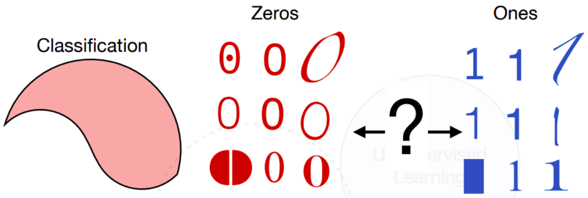

5$
0 ? 1
Classification
Supervised Learning
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning


5$
Supervised
Learning
Regression
3.5$

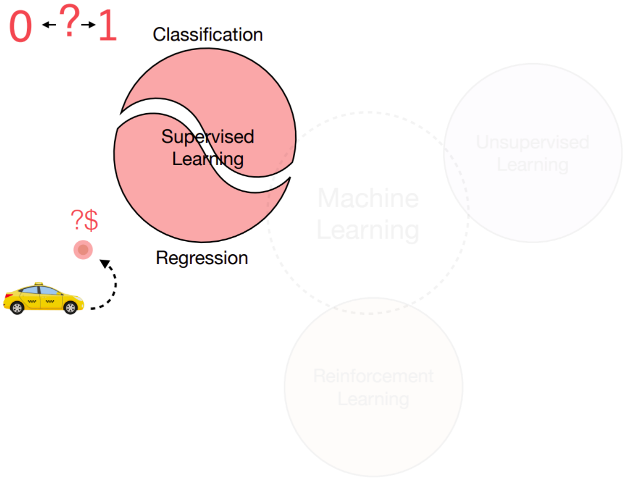

7$
5$
3.5$
Classification
Supervised
Learning
Regression
Unsupervised
Learning
Machine
Learning
Reinforcement
Learning

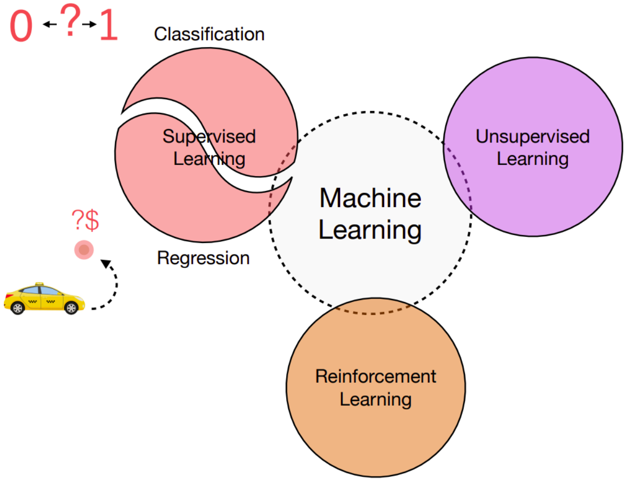


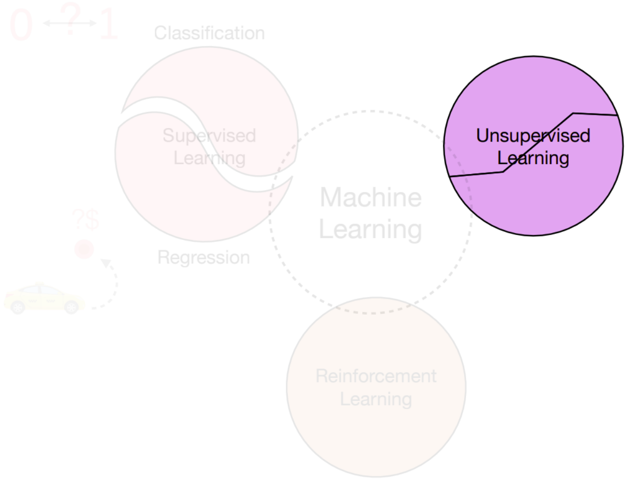
Predicting **continuous** values

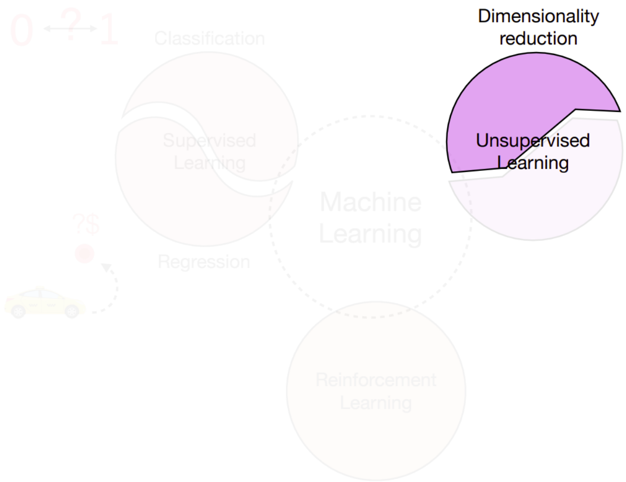


What is the main similarity/difference between these two classes of **supervised learning**?

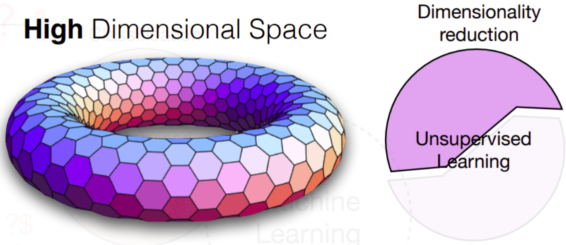

0 ? 1
Classification
Supervised Learning
?$
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning

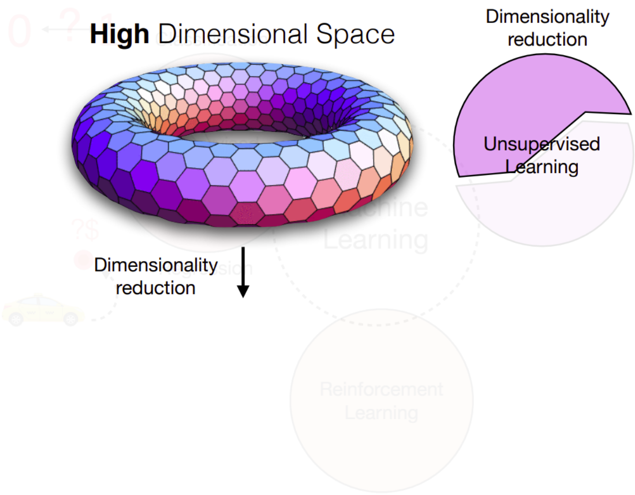

0 ← ? → 1
Classification
Supervised Learning
?$
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning

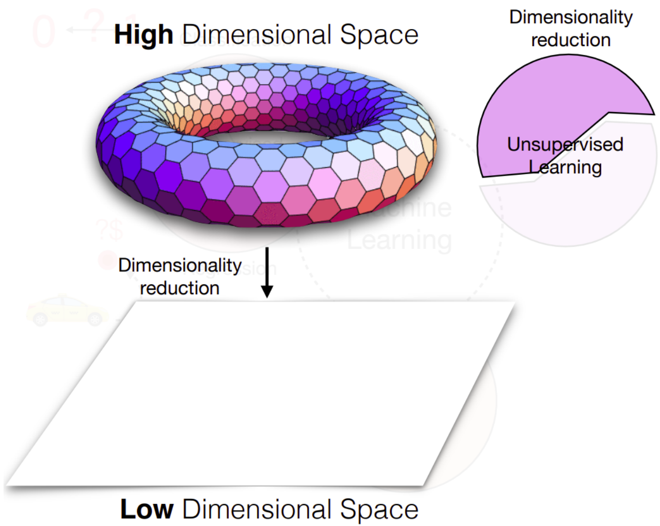

0 ? 1
Classification
Supervised Learning
?$
Regression
Machine Learning
Unsupervised Learning
Reinforcement Learning

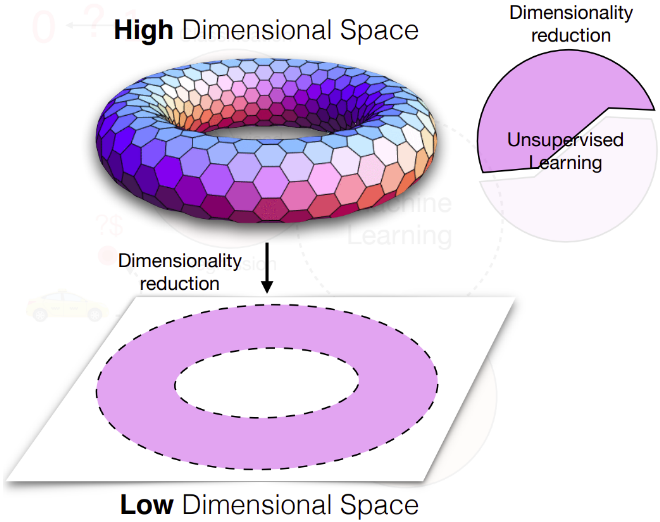

Dimensionality
reduction
Unsupervised
Learning
Classification
Supervised
Learning
Regression
Machine
Learning
Reinforcement
Learning
0 ? 1
?$

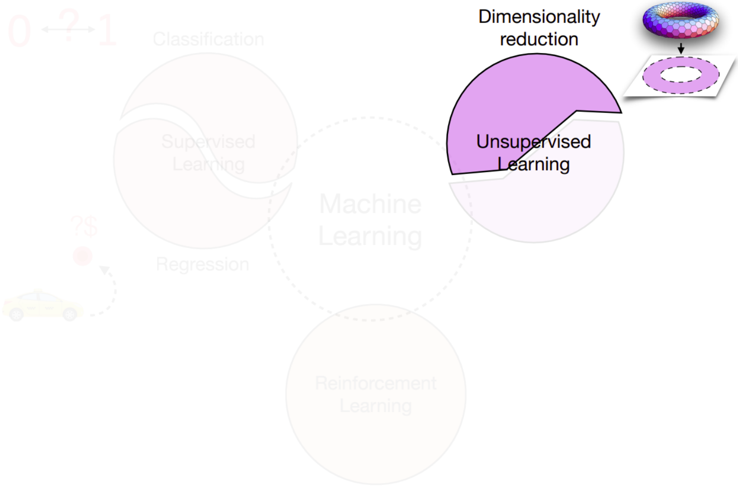
High Dimensional Space

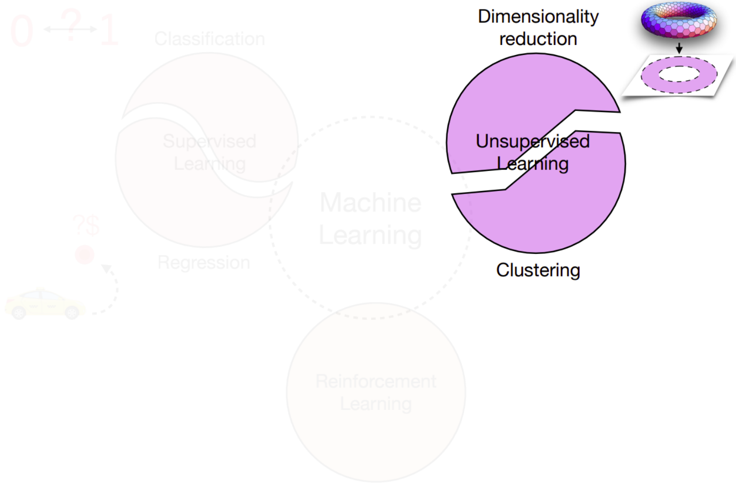


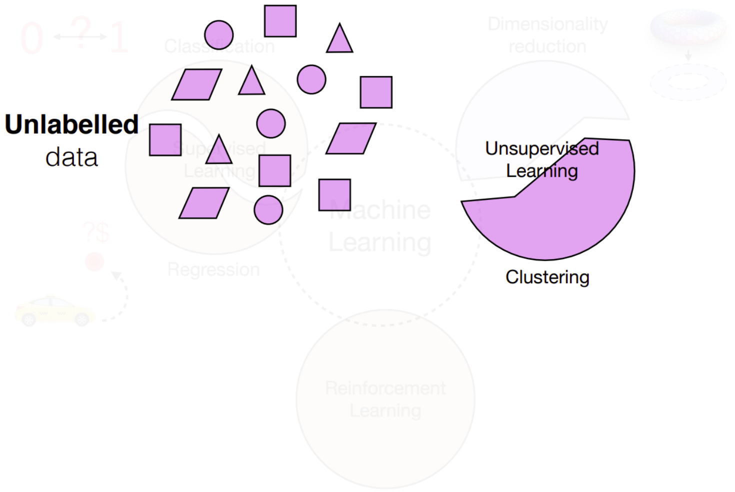

High Dimensional Space
Dimensionality reduction
Dimensionality reduction
Unsupervised Learning

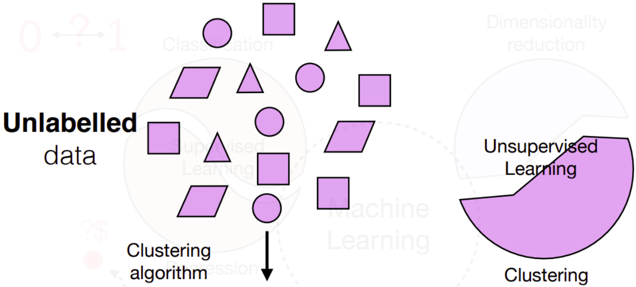

High Dimensional Space
Dimensionality reduction
Unsupervised Learning
Dimensionality reduction
Low Dimensional Space

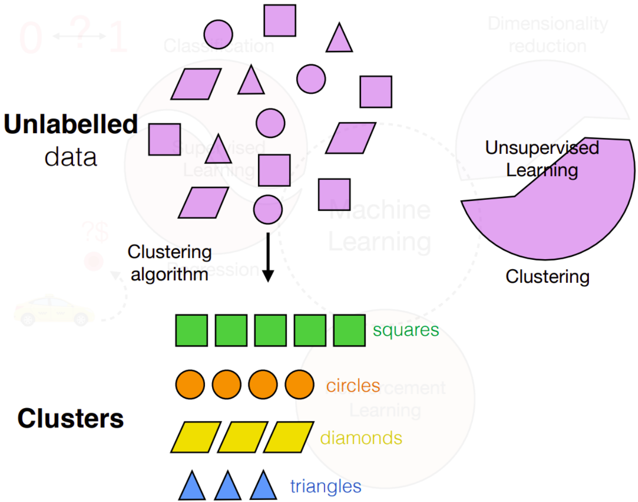

High Dimensional Space
Dimensionality reduction
Unsupervised Learning
Dimensionality reduction
Low Dimensional Space

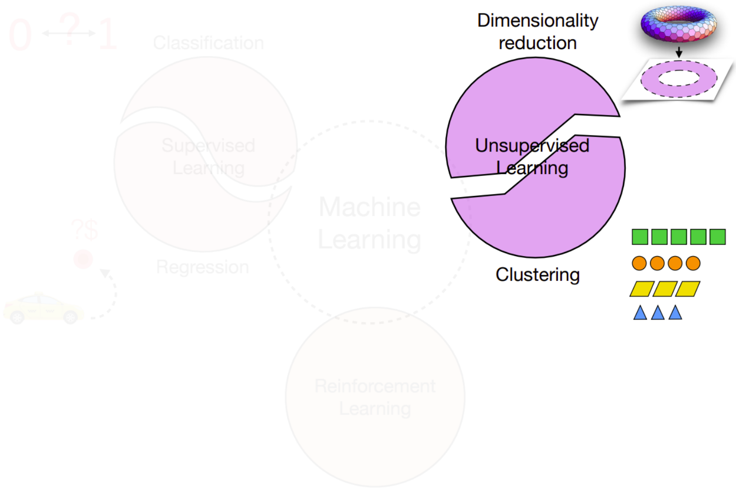

0 ? 1
Classification
Supervised Learning
Regression
?$
Machine Learning
Dimensionality reduction
Unsupervised Learning
Reinforcement Learning


Dimensionality reduction
Unsupervised Learning
Clustering
Classification
Supervised Learning
Regression
Machine Learning
Reinforcement Learning
0
1
?$

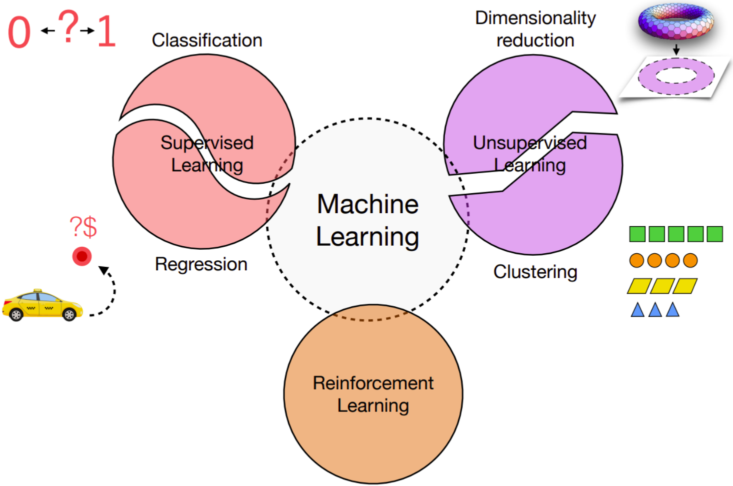

Unlabelled
data
Unsupervised
Learning
Clustering

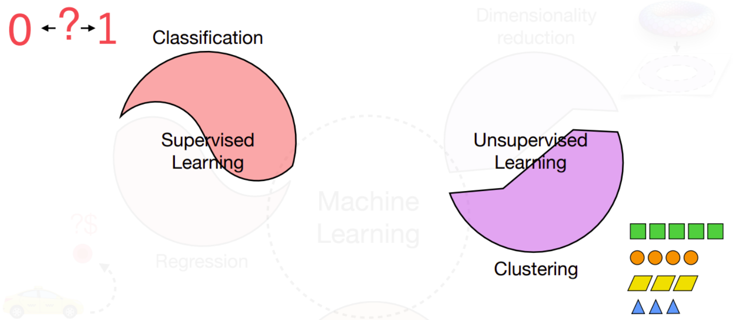

Unlabelled
data
Clustering
algorithm
Unsupervised
Learning
Clustering


Unlabelled
data
Clustering
algorithm
Clusters
squares
circles
diamonds
triangles
Unsupervised
Learning
Clustering

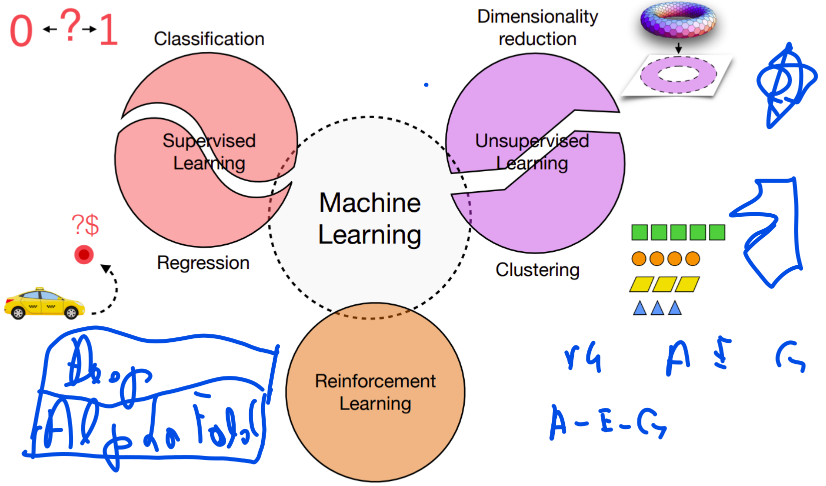


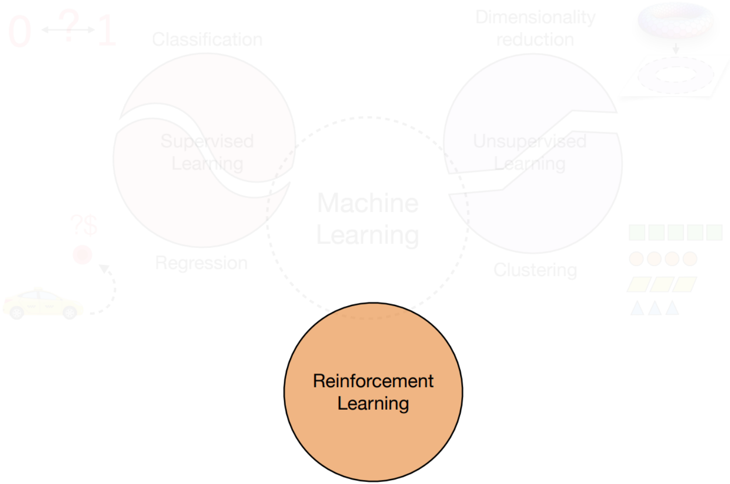
0 ?-? 1
Classification
Supervised Learning
?$
Regression
Machine Learning
Reinforcement Learning
Dimensionality reduction
Unsupervised Learning
Clustering

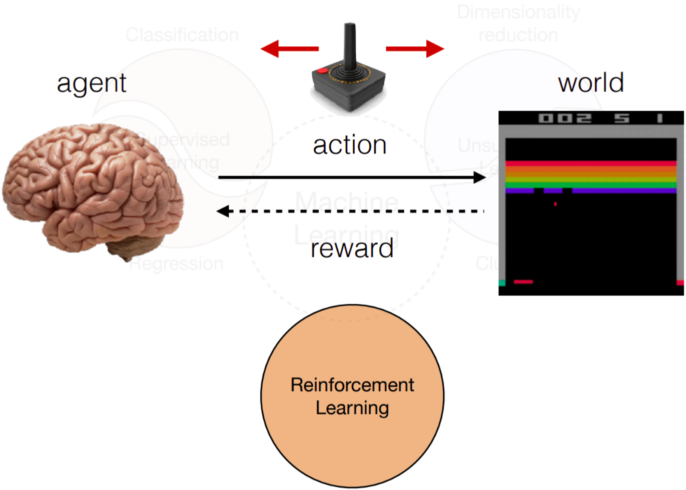

0 ← ? → 1
Classification
Supervised Learning
Regression
?$
Dimensionality reduction
Unsupervised Learning
Clustering
Machine Learning
Reinforcement Learning

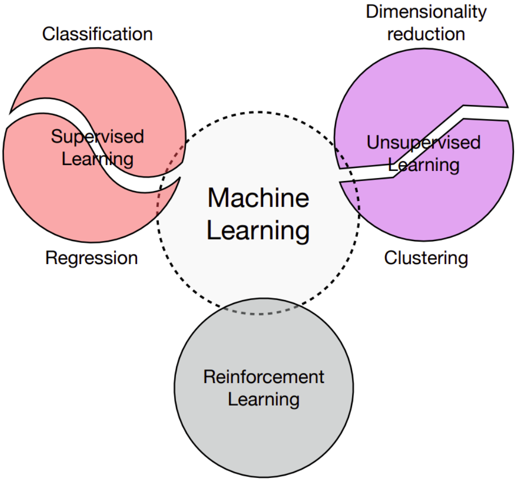


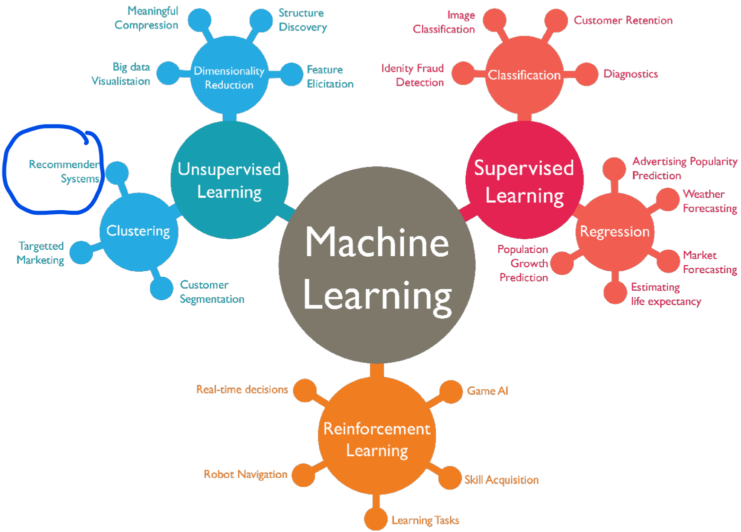
What is the main similarity/difference between **classification** and **clustering**?


0 ← ? → 1
Classification
Supervised Learning
Regression
?$
Machine Learning
Dimensionality reduction
Unsupervised Learning
Clustering
Reinforcement Learning
A - E - G


0 ? 1
Classification
Supervised Learning
Regression
?$
Machine Learning
Dimensionality reduction
Unsupervised Learning
Clustering
Reinforcement Learning


agent
action
reward
world
Reinforcement Learning

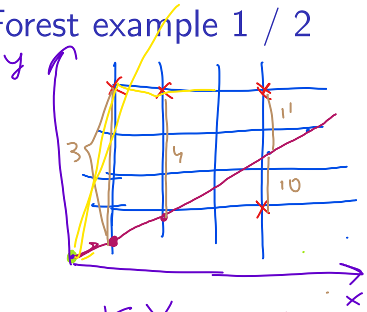

Classification
Supervised Learning
Regression
Dimensionality reduction
Unsupervised Learning
Clustering
Machine Learning
Reinforcement Learning


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

More examples

- Classification - Given first few words predict the ending of a se
- Regression - Based on apartment location, size, floor ... predict the price
- Clustering - People donate blood and we want to group them by risk-levels (how likely they are to experience and adverse event) based on their age, weight, gender etc.
- Reinforcement Learning - AI playing hide and seek (https://www.youtube.com/watch?v=kopoLzvh5jY)

Forest example 1 / 2



1000

0.1

$y = \underline{k} x$

$k = \underline{1.5} \rightarrow 100$

1.51

$k = 20 \rightarrow \underline{0.05}$

$k = 20.2$

Forest example 2 / 2

Formalizing

Now we need to formalize the following concepts.

1. Path (line (model))

$f(x_1, x_2, x_3) \to y$

$x = 4$

$1.5x \to 6$

Formalizing

Now we need to formalize the following concepts.

1. Path (line (model))
2. Hungriness (Risk, Cost, Error)

Formalizing

Now we need to formalize the following concepts.

1. Path (line (model))
2. Hungriness (Risk, Cost, Error)
3. Updating path (Optimization, Gradient Descent)

Formalizing

Now we need to formalize the following concepts.

1. Path (line (model))
2. Hungriness (Risk, Cost, Error)
3. Updating path (Optimization, Gradient Descent)

Model

WHAT IS A MODEL?

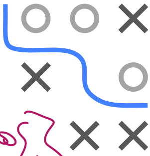
$f: \mathcal{X} \to \mathbb{R}^g$

- A **model** (or **hypothesis**)

$$f : \mathcal{X} \to \mathbb{R}^g$$

is a function that maps feature vectors to predicted target values.

- In conventional regression: $g = 1$; for classification g is the number of classes, and output vectors are scores or class probabilities (details later).



inp

y

X, y

$\hat{y}$ y hat

New Features

x_1	x_2
6	3300
5	3100

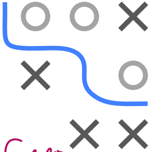
Model f

Prediction of Target Variable

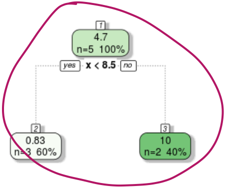
$\hat{y}$
2050
2200

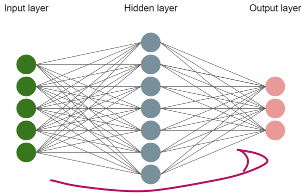
WHAT IS A MODEL?

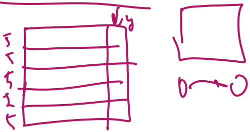
- f is meant to capture intrinsic patterns of the data, the underlying assumption being that these hold true for *all* data drawn from $\mathbb{P}_{xy}$.
- It is easily conceivable how models can range from super simple (e.g., linear, tree stumps) to very complex (e.g., deep neural networks) and there are infinitely many choices how we can construct such functions.



Data Generation





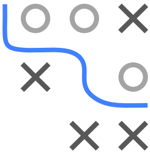


$y_{n+1} > 3 \rightarrow 0$

$< 3 \rightarrow 100$

- In fact, ML requires **constraining** f to a certain type of functions.

HYPOTHESIS SPACES



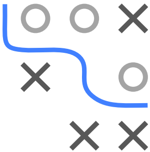
- Without restrictions on the functional family, the task of finding a "good" model among all the available ones is impossible to solve.
- This means: we have to determine the class of our model *a priori*, thereby narrowing down our options considerably. We could call that a **structural prior**.
- The set of functions defining a specific model class is called a **hypothesis space** $\mathcal{H}$:

$$\mathcal{H} = \{f : f \text{ belongs to a certain functional family}\}$$

$\mathcal{H}$

$$\mathcal{H} = \{c_0 + c_1 x + c_2 x^2 \mid c_0, c_1, c_2 \in \mathbb{R}\}$$

PARAMETRIZATION



- All models within one hypothesis space share a common functional structure. We usually construct the space as **parametrized family of curves**.
- We collect all parameters in a **parameter vector** $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_d)$ from **parameter space** Θ.
- They are our means of fixing a specific function from the family. Once set, our model is fully determined.
- Therefore, we can re-write $\mathcal{H}$ as:

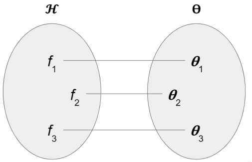
$$\mathcal{H} = \{f_{\boldsymbol{\theta}} : \ \underline{f_{\boldsymbol{\theta}} \text{ belongs to a certain functional family}} \ \text{parameterized by } \boldsymbol{\theta}\}$$

θ

$f_{\{3,5,5\}}$ $\quad 3+5x+5x^2$

PARAMETRIZATION

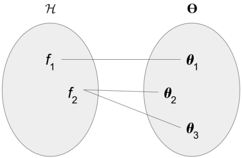
- This means: finding the optimal model is perfectly equivalent to finding the optimal set of parameter values.
- The relation between optimization over $f \in \mathcal{H}$ and optimization over $\boldsymbol{\theta} \in \Theta$ allows us to operationalize our search for the best model via the search for the optimal value on a d-dimensional parameter surface.



- $\boldsymbol{\theta}$ might be scalar or comprise thousands of parameters, depending on the complexity of our model.

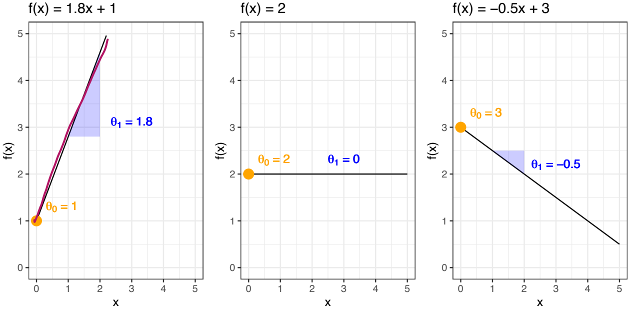
PARAMETRIZATION

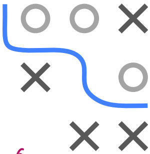
- Short remark: In fact, some parameter vectors, for some model classes, might encode the same function. So the parameter-to-model mapping could be non-injective.
- We call this then a non-identifiable model.
- But this shall not concern us here.



EXAMPLE: UNIVARIATE LINEAR FUNCTIONS

$$\mathcal{H} = \{f : f(\mathbf{x}) = \theta_0 + \theta_1 x, \boldsymbol{\theta} \in \mathbb{R}^2\}$$

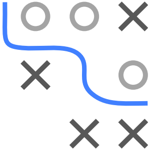




$\theta_0, \theta_1 \in R$

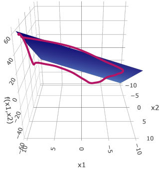
$\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \in R^2$

EXAMPLE: BIVARIATE QUADRATIC FUNCTIONS

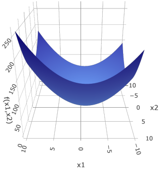


$$\mathcal{H} = \{f : f(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2, \boldsymbol{\theta} \in \mathbb{R}^6\},$$

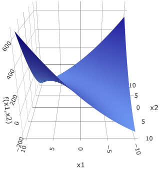
$f(x) = 3 + 2x_1 + 4x_2$



$f(x) = 3 + 2x_1 + 4x_2 +$
$+ 1x_1^2 + 1x_2^2$

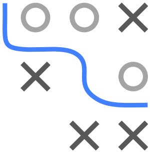


$f(x) = 3 + 2x_1 + 4x_2 +$
$+ 1x_1^2 + 1x_2^2 + 4x_1 x_2$



x_1 x_2

SUPERVISED LEARNING EXAMPLE



Imagine we want to investigate how working conditions affect productivity of employees.

- It is a **regression** task since the target *productivity* is continuous.
- We collect data about worked minutes per week (*productivity*), how many people work in the same office as the employee in question, and the employee's salary.

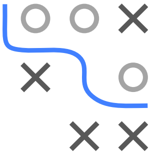
Features x		Target y
People in Office (Feature 1) x_1	Salary (Feature 2) x_2	Worked Minutes Week (Target Variable)
4	4300 €	2220
12	2700 €	1800
5	3100 €	1920

$n = 3$

$p = 2$

$x_1^{(2)}$ $x_2^{(1)}$ $y^{(3)}$

SUPERVISED LEARNING EXAMPLE

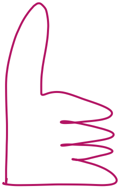


How could we construct a model from these data?

We could investigate the data manually and come up with a simple, hand-crafted rule such as:

- The baseline productivity of an employee with salary 3000 and 7 people in the office is 1850 minutes
- A decrease of 1 person in the office increases productivity by 30
- An increase of the salary by 100 increases productivity by 10

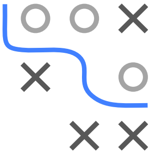
=> Obviously, this is neither feasible nor leads to a good model



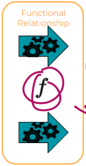
IDEA OF SUPERVISED LEARNING

Goal: Automatically identify the fundamental functional relation in the data that maps an object's features to the target.

- **Supervised** learning means we make use of *labeled* data for which we observed the outcome.
- We use the labeled data to learn a model f.
- Ultimately, we use our model to compute predictions for **new** data whose target values are unknown.



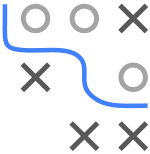
	x_1	x_2	y
Already seen Data	4	4300 €	2200
	12	2700 €	1800
	15	3100 €	1920
New Data	6	3300 €	???
	5	3100 €	???



→ da

→ predict

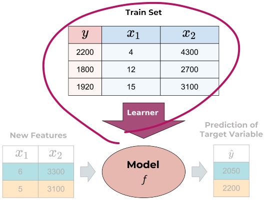
LEARNER DEFINITION



- The algorithm for finding our f is called **learner**. It is also called **learning algorithm** or **inducer**.
- We prescribe a certain hypothesis space, the learner is our means of picking the best element from that space for our data set.
- Formally, it maps training data $\mathcal{D} \in \mathbb{D}$ (plus a vector of **hyperparameter** control settings $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$) to a model:

$$\mathcal{I} : \mathbb{D} \times \boldsymbol{\Lambda} \to \mathcal{H}$$

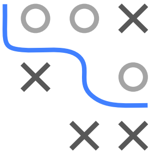
$\mathcal{I} : \mathbb{D} \times \Lambda \to \mathcal{H}$



LEARNER DEFINITION

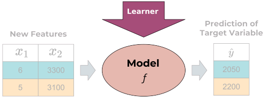
As pseudo-code template it would work like this:

- Learner has a defined model space of parametrized functions $\mathcal{H}$.
- User passes data set $\mathcal{D}_{\text{train}}$ and control settings $\boldsymbol{\lambda}$.
- Learner sets parameters so that model matches data best.
- Optimal parameters $\hat{\theta}$ or function $\hat{f}$ is returned for later usage.



Train Set

y	x_1	x_2
2200	4	4300
1800	12	2700
1920	15	3100



f $\hat{y}$ y

1100 1020 1050

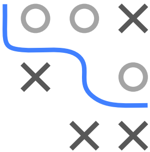
250 260 → 310

~~Model~~

Hungriness

(evaluation)

HOW TO EVALUATE MODELS

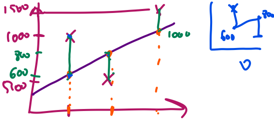


- When training a learner, we optimize over our hypothesis space, to find the function which matches our training data best.
- This means, we are looking for a function, where the predicted output per training point is as close as possible to the observed label.

Features x		Target y		Prediction $\hat{y}$
People in Office (Feature 1) x_1	Salary (Feature 2) x_2	Worked Minutes Week (Target Variable)		Worked Minutes Week (Target Variable)
4	4300 €	2220	$\overset{?}{\approx}$	2588
12	2700 €	1800		1644
5	3100 €	1920		1870

$\mathcal{D}_{\text{train}}$

1, 3, $\boxed{4}$, 5, 20



- To make this precise, we need to define now how we measure the difference between a prediction and a ground truth label pointwise.

$|1000-600| = 400$

$|500-800| = -300$

$1500-1000 = 500$

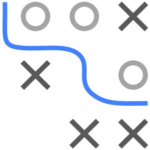
$\frac{1}{3}(400-300+500) = 200$

$y - \hat{y}$

$\boxed{y_n, \hat{y}_n}$

$L : \mathcal{Y}^1 \times \mathbb{R}^1 \to \mathbb{R}$

LOSS

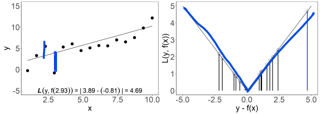


The **loss function** $L(y, f(\mathbf{x}))$ quantifies the "quality" of the prediction $f(\mathbf{x})$ of a single observation $\mathbf{x}$:

$$L : \mathcal{Y} \times \mathbb{R}^g \to \mathbb{R}.$$

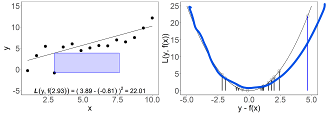
In regression, we could use the absolute loss $L(y, f(\mathbf{x})) = |f(\mathbf{x}) - y|$;

L1



or the L2-loss $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$.

L2



RISK OF A MODEL

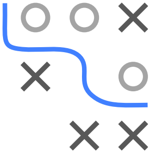
- The (theoretical) **risk** associated with a certain hypothesis $f(\mathbf{x})$ measured by a loss function $L(y, f(\mathbf{x}))$ is the **expected loss**

$$\mathcal{R}(f) := \mathbb{E}_{xy}[L(y, f(\mathbf{x}))] = \int L(y, f(\mathbf{x}))\, d\mathbb{P}_{xy}.$$

- This is the average error we incur when we use f on data from $\mathbb{P}_{xy}$.
- Goal in ML: Find a hypothesis $f(\mathbf{x}) \in \mathcal{H}$ that **minimizes** risk.

Empri: Ris M

RISK OF A MODEL



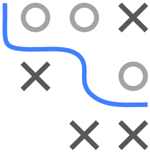
Problem: Minimizing $\mathcal{R}(f)$ over f is not feasible:

- $\mathbb{P}_{xy}$ is unknown (otherwise we could use it to construct optimal predictions).
- We could estimate $\mathbb{P}_{xy}$ in non-parametric fashion from the data $\mathcal{D}$, e.g., by kernel density estimation, but this really does not scale to higher dimensions (see “curse of dimensionality”).
- We can efficiently estimate $\mathbb{P}_{xy}$, if we place rigorous assumptions on its distributional form, and methods like discriminant analysis work exactly this way.

But as we have n i.i.d. data points from $\mathbb{P}_{xy}$ available we can simply approximate the expected risk by computing it on $\mathcal{D}$.

EMPIRICAL RISK

To evaluate, how well a given function f matches our training data, we now simply sum-up all f's pointwise losses.



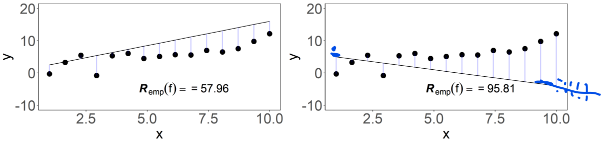
$$\mathcal{R}_{\text{emp}}(f) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right) \qquad \frac{1}{n}$$

This gives rise to the **empirical risk function** which allows us to associate one quality score with each of our models, which encodes how well our model fits our training data.

$3 \cdot \theta_1 + 6\theta_2 = 0$

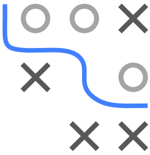
?

$$\mathcal{R}_{\text{emp}} : \mathcal{H} \rightarrow \mathbb{R}$$



:30

EMPIRICAL RISK



- The risk can also be defined as an average loss

$$\bar{\mathcal{R}}_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right).$$

The factor $\frac{1}{n}$ does not make a difference in optimization, so we will consider $\mathcal{R}_{\text{emp}}(f)$ most of the time.

- Since f is usually defined by **parameters** $\boldsymbol{\theta}$, this becomes:

$$\mathcal{R} : \mathbb{R}^d \to \mathbb{R}$$

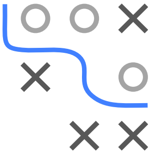
$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)\right)$$

EMPIRICAL RISK MINIMIZATION

V porth

v hug

The best model is the model with the smallest risk.



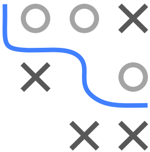
If we have a finite number of models f, we could simply tabulate them and select the best.

Model	$\theta_{intercept}$	θ_{slope}	$\mathcal{R}_{\text{emp}}(\theta)$
f_1	2	3	194.62
f_2	3	2	127.12
f_3	6	-1	95.81
f_4	1	1.5	57.96

$\{\theta_1 Kx \mid K \in \mathbb{R}\}$

EMPIRICAL RISK MINIMIZATION

But usually $\mathcal{H}$ is infinitely large.

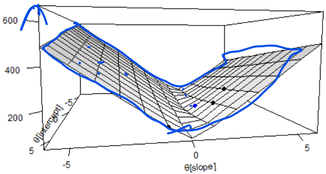


Instead we can consider the risk surface w.r.t. the parameters $\boldsymbol{\theta}$.
(By this I simply mean the visualization of $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$)

$K = 3$
$d = 4$

$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}.$$

Model	$\boldsymbol{\theta}_{intercept}$	$\boldsymbol{\theta}_{slope}$	$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$
f_1	2	3	194.62
f_2	3	2	127.12
f_3	6	-1	95.81
f_4	1	1.5	57.96



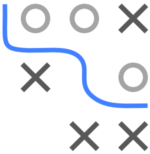
EMPIRICAL RISK MINIMIZATION

Minimizing this surface is called **empirical risk minimization** (ERM).

$$\hat{\theta} = \underset{\boldsymbol{\theta} \in \Theta}{\arg\min}\, \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}).$$

$\hat{\theta}$

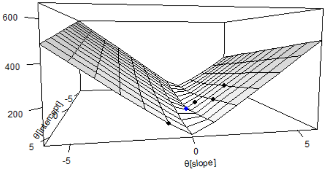
$\Theta = \mathbb{R}^2$



Usually we do this by numerical optimization.

$$\mathcal{R} : \mathbb{R}^d \to \mathbb{R}.$$

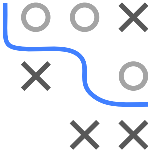
Model	$\boldsymbol{\theta}_{intercept}$	$\boldsymbol{\theta}_{slope}$	$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$
f_1	2	3	194.62
f_2	3	2	127.12
f_3	6	-1	95.81
f_4	1	1.5	57.96
f_5	1.25	0.90	23.40



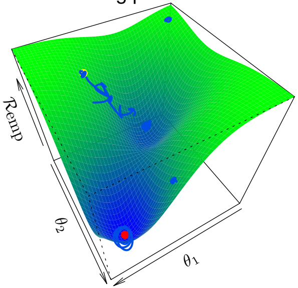
In a certain sense, we have now reduced the problem of learning to **numerical parameter optimization**.

Optimization

LEARNING AS PARAMETER OPTIMIZATION

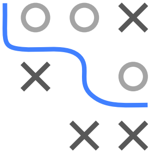


- We have seen, we can operationalize the search for a model f that matches training data best, by looking for its parametrization $\boldsymbol{\theta} \in \Theta$ with lowest empirical risk $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$.
- Therefore, we usually traverse the error surface downwards; often by local search from a starting point to its minimum.



initi

LEARNING AS PARAMETER OPTIMIZATION



The ERM optimization problem is:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\arg\min}\, \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}).$$

For a **(global) minimum** $\hat{\boldsymbol{\theta}}$ it obviously holds that

$$\forall \boldsymbol{\theta} \in \Theta : \quad \mathcal{R}_{\text{emp}}(\hat{\boldsymbol{\theta}}) \leq \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}).$$

This does not imply that $\hat{\boldsymbol{\theta}}$ is unique.

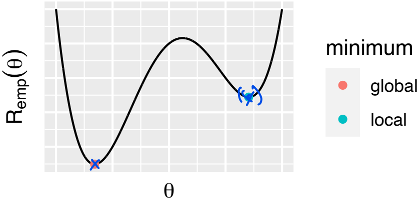
Which kind of numerical technique is reasonable for this problem strongly depends on model and parameter structure (continuous params? uni-modal $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$?). Here, we will only discuss very simple scenarios.

LOCAL MINIMA

If $\mathcal{R}_{\text{emp}}$ is continuous in $\boldsymbol{\theta}$ we can define a **local minimum** $\hat{\boldsymbol{\theta}}$:

$$\exists \epsilon > 0 \; \forall \boldsymbol{\theta} \text{ with } \left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\| < \epsilon : \quad \mathcal{R}_{\text{emp}}(\hat{\boldsymbol{\theta}}) \leq \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}).$$

Clearly every global minimum is also a local minimum. Finding a local minimum is easier than finding a global minimum.

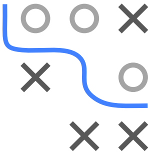


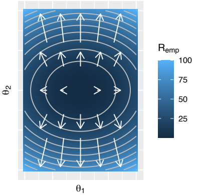
LOCAL MINIMA AND STATIONARY POINTS

If $\mathcal{R}_{\text{emp}}$ is continuously differentiable in $\boldsymbol{\theta}$ then a **sufficient condition** for a local minimum is that $\hat{\boldsymbol{\theta}}$ is **stationary** with 0 gradient, so no local improvement is possible:

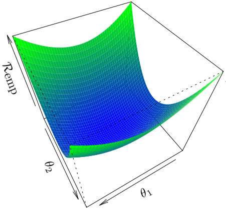
$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{R}_{\text{emp}}(\hat{\boldsymbol{\theta}}) = 0$$

and the Hessian $\frac{\partial^2}{\partial \boldsymbol{\theta}^2} \mathcal{R}_{\text{emp}}(\hat{\boldsymbol{\theta}})$ is positive definite. While the neg. gradient points into the direction of fastest local decrease, the Hessian measures local curvature of $\mathcal{R}_{\text{emp}}$.

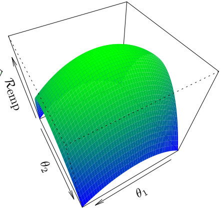




$\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$



const. pos. def. Hessian



const. neg. def. Hessian

$f_1, f_2, \dots$ L2 $(f(x)-y)^2 = \sum_{i=1} (\theta^T x^{(i)} - y^{(i)})^2$ MSE

$\theta = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $y = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ $\theta^T X$

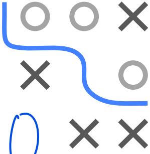
LEAST SQUARES ESTIMATOR

Now, for given features $\mathbf{X} \in \mathbb{R}^{n\times p}$ and target $\mathbf{y} \in \mathbb{R}^n$, we want to find the best linear model regarding the squared error loss, i.e.,

$$\mathcal{R}_{\text{emp}}(\theta) = \|\mathbf{X}\theta - \mathbf{y}\|_2^2 = \sum_{i=1}^{n} (\theta^\top \mathbf{x}^{(i)} - y^{(i)})^2.$$

L2

L1

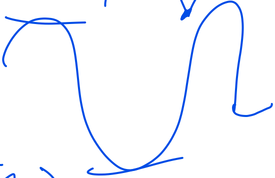


R = 0

With the sufficient condition for continously differentiable functions it can be shown that the **least squares estimator**

$$\hat{\theta} = \boxed{(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}}.$$

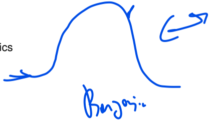
closed



is a local minimum of $\mathcal{R}_{\text{emp}}$. If $\mathbf{X}$ is full-rank, $\mathcal{R}_{\text{emp}}$ is strictly convex and there is only one local minimum - which is also global.

Note: Often such analytical solutions in ML are not possible, and we rather have to use iterative numerical optimization.

$\begin{bmatrix} 7.5 \\ 3.6 \end{bmatrix} \rightarrow 7.5x_1 + 3.6x_2$



CLT

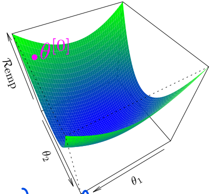
$X_1 + X_2 + X_3 + X \dots$

MLE

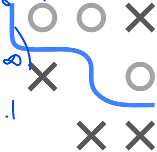
GRADIENT DESCENT

The simple idea of GD is to iteratively go from the current candidate $\boldsymbol{\theta}^{[t]}$ in the direction of the negative gradient, i.e., the direction of the steepest descent, with learning rate α to the next $\boldsymbol{\theta}^{[t+1]}$:

$$\boldsymbol{\theta}^{[t+1]} = \boldsymbol{\theta}^{[t]} - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}^{[t]}).$$



We choose a random start $\boldsymbol{\theta}^{[0]}$ with risk $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}^{[0]}) = 76.25$.



$f(x,y)$ $\begin{bmatrix} f'_x \\ f'_y \end{bmatrix}$

$\alpha \in (0, \infty)$

$\alpha: 0.1$

$10 \to 10 - 3 \cdot 0.1 \to 9.7 \to 9.2$

x_1 x_2 y

x_1	x_2	y
3	4	5
6	9	8

$R = \sum (\ldots)^2$

$\theta = \begin{bmatrix} 8 \\ 45 \end{bmatrix}$

$\theta_0 3 + 4\theta_1 = 5$

$6 \quad 9 \to 8$

x^2 Polynom.

x_1 x_2 x_1^2 $\sin x_1$

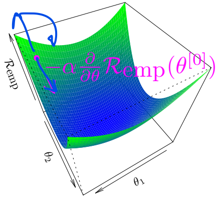
$\theta_0 x + \theta_1 x_1 + \theta_2 x_1^2$

$\theta_3 x_1 + \theta_1 x_2$

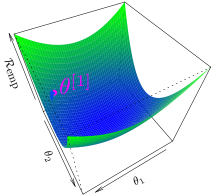
$x_1^2 + \sin x_1 \to y$

Linear Regression

GRADIENT DESCENT - EXAMPLE

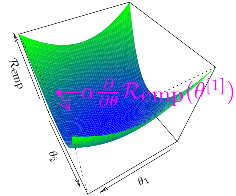


Now we follow in the direction of the negative gradient at $\boldsymbol{\theta}^{[0]}$.

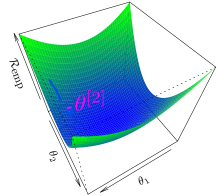


We arrive at $\boldsymbol{\theta}^{[1]}$ with risk $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}^{[1]}) \approx 42.73$.
We improved: $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}^{[1]}) < \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}^{[0]})$.

GRADIENT DESCENT - EXAMPLE

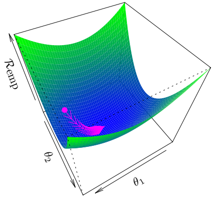


Again we follow in the direction of the negative gradient, but now at $\boldsymbol{\theta}^{[1]}$.

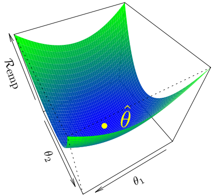


Now $\boldsymbol{\theta}^{[2]}$ has risk $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}^{[2]}) \approx 25.08$.

GRADIENT DESCENT - EXAMPLE



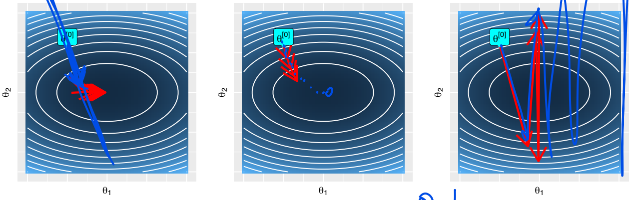
We iterate this until some form of convergence or termination.



We arrive close to a stationary $\hat{\boldsymbol{\theta}}$ which is hopefully at least a local minimum.

GRADIENT DESCENT - LEARNING RATE

- The negative gradient is a direction that looks locally promising to reduce $\mathcal{R}_{\text{emp}}$.
- Hence it weights components higher in which $\mathcal{R}_{\text{emp}}$ decreases more.
- However, the length of $-\frac{\partial}{\partial \boldsymbol{\theta}}\mathcal{R}_{\text{emp}}$ measures only the local decrease rate, i.e., there are no guarantees that we will not go "too far".
- We use a learning rate α to scale the step length in each iteration. Too much can lead to overstepping and no converge, too low leads to slow convergence.
- Usually, a simple constant rate or rate-decrease mechanisms to enforce local convergence are used



good convergence for α_1 | poor convergence for $\alpha_2 (< \alpha_1)$ | no convergence for $\alpha_3 (> \alpha_1)$

scheduling

L.R. decay

0.1

$\alpha := \alpha \cdot 0.99$

Hypers

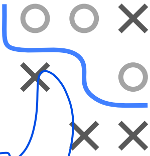
$I: \mathcal{D} \times \Lambda \to \mathcal{H}$

0.1

0-1000

FURTHER TOPICS

- GD is a so-called first-order method. Second-order methods use the Hessian to refine the search direction for faster convergence.
- There exist many improvements of GD, e.g., to smartly control the learn rate, to escape saddle points, to mimic second order behavior without computing the expensive Hessian.
- If the gradient of GD is not derived from the empirical risk of the whole data set, but instead from a randomly selected subset, we call this **stochastic gradient descent** (SGD). For large-scale problems this can lead to higher computational efficiency.



Regression with L2 Loss

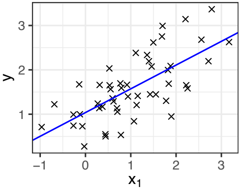
LINEAR REGRESSION

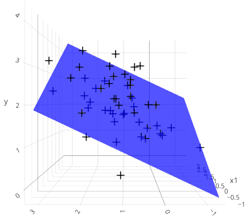
- Idea: predict $y \in \mathbb{R}$ as **linear** combination of features[1]:

$$\hat{y} = f(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} = \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p$$

  $\rightsquigarrow$ find loss-optimal params to describe relation $y|\mathbf{x}$

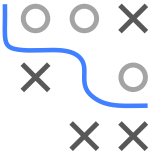
- Hypothesis space: $\mathcal{H} = \{f(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} \mid \boldsymbol{\theta} \in \mathbb{R}^{p+1}\}$





[1] Actually, special case of linear model, which is linear combo of *basis functions* of features $\rightsquigarrow$ Polynomial Regression Models

DESIGN MATRIX



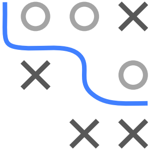
- Mismatch: $\boldsymbol{\theta} \in \mathbb{R}^{p+1}$ vs $\mathbf{x} \in \mathbb{R}^{p}$ due to intercept term
- Trick: pad feature vectors with leading 1, s.t.
  - $\mathbf{x} \mapsto \mathbf{x} = (1, x_1, \ldots, x_p)^\top$, and
  - $\boldsymbol{\theta}^\top \mathbf{x} = \theta_0 \cdot 1 + \theta_1 x_1 + \cdots + \theta_p x_p$
- Collect all observations in **design matrix** $\mathbf{X} \in \mathbb{R}^{n \times (p+1)}$
  $\rightsquigarrow$ more compact: single param vector incl. intercept
- Resulting linear model:

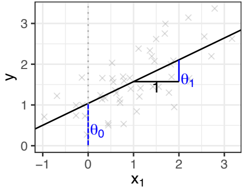
$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} = \begin{pmatrix} 1 & x_1^{(1)} & \ldots & x_p^{(1)} \\ 1 & x_1^{(2)} & \ldots & x_p^{(2)} \\ \vdots & \vdots & & \vdots \\ 1 & x_1^{(n)} & \ldots & x_p^{(n)} \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_p \end{pmatrix} = \begin{pmatrix} \theta_0 + \theta_1 x_1^{(1)} + \cdots + \theta_p x_p^{(1)} \\ \theta_0 + \theta_1 x_1^{(2)} + \cdots + \theta_p x_p^{(2)} \\ \vdots \\ \theta_0 + \theta_1 x_1^{(n)} + \cdots + \theta_p x_p^{(n)} \end{pmatrix}$$

- We will make use of this notation in other contexts

EFFECT INTERPRETATION

- Big plus of LM: immediately **interpretable** feature effects
- "Marginally increasing x_j by 1 unit increases y by θ_j units"
  $\rightsquigarrow$ *ceteris paribus* assumption: $x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_p$ fixed



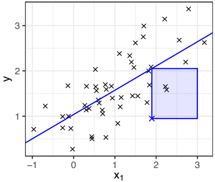



```
Call:
lm(formula = y ~ x_1, data = dt_univ)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.10346 -0.34727 -0.00766  0.31500  1.04284 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.03727    0.11360   9.131 4.55e-12 ***
x_1          0.53521    0.08219   6.512 4.13e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5327 on 48 degrees of freedom
Multiple R-squared:  0.469,	Adjusted R-squared:  0.458 
F-statistic:  42.4 on 1 and 48 DF,  p-value: 4.129e-08
```

MODEL FIT

- How to determine LM fit? $\rightsquigarrow$ define risk & optimize
- Popular: *L*2 **loss** / **quadratic loss** / **squared error**

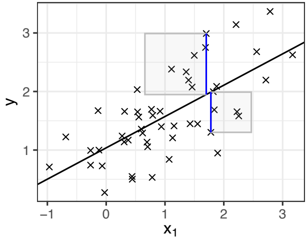
$$L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2 \text{ or } L(y, f(\mathbf{x})) = 0.5 \cdot (y - f(\mathbf{x}))^2$$

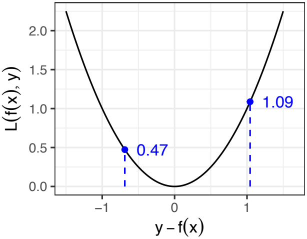


- Why penalize **residuals** $r = y - f(\mathbf{x})$ quadratically?
  - Easy to optimize (convex, differentiable)
  - Theoretically appealing (connection to classical stats LM)

LOSS PLOTS

We will often visualize loss effects like this:





- Data as $y \sim x_1$
- Prediction hypersurface
  $\rightsquigarrow$ here: line
- Residuals $r = y - f(\mathbf{x})$
  $\rightsquigarrow$ squares to illustrate loss

- Loss as function of residuals
  $\rightsquigarrow$ strength of penalty?
  $\rightsquigarrow$ symmetric?
- Highlighted: loss for residuals shown on LHS

OPTIMIZATION

- Resulting risk equivalent to **sum of squared errors (SSE)**:

$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\theta}^{\top} \mathbf{x}^{(i)} \right)^2$$

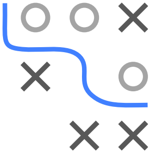
- Consider example with $n = 5$ $\rightsquigarrow$ different models with varying SSE

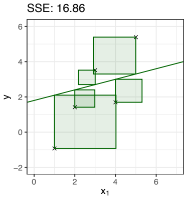
OPTIMIZATION

- Resulting risk equivalent to **sum of squared errors (SSE)**:

$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)} \right)^2$$

- Consider example with $n = 5 \rightsquigarrow$ different models with varying SSE



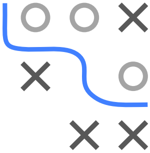


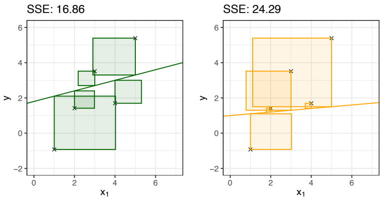
OPTIMIZATION

- Resulting risk equivalent to **sum of squared errors (SSE)**:

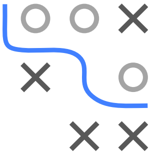
$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)} \right)^2$$

- Consider example with $n = 5 \rightsquigarrow$ different models with varying SSE





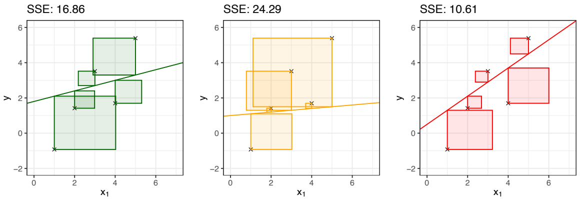
OPTIMIZATION



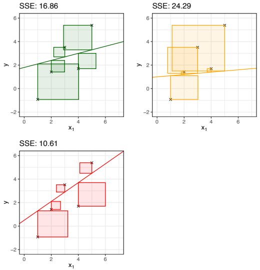
- Resulting risk equivalent to **sum of squared errors (SSE)**:

$$\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)} \right)^2$$

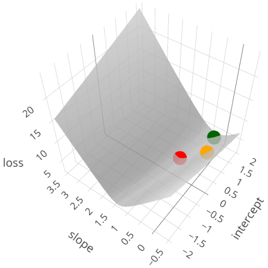
- Consider example with $n = 5 \rightsquigarrow$ different models with varying SSE



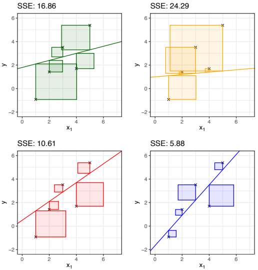
OPTIMIZATION

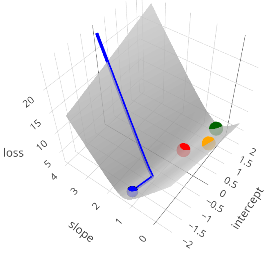


Intercept θ_0	Slope θ_1	SSE
1.80	0.30	16.86
1.00	0.10	24.29
0.50	0.80	10.61



OPTIMIZATION





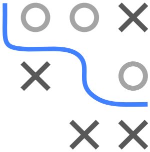
Intercept θ_0	Slope θ_1	SSE
1.80	0.30	16.86
1.00	0.10	24.29
0.50	0.80	10.61
-1.65	1.29	5.88

Instead of guessing, of course, use **optimization**!

ANALYTICAL OPTIMIZATION

- Special property of LM with *L2* loss: **analytical solution** available

$$\hat{\boldsymbol{\theta}} \in \underset{\boldsymbol{\theta}}{\arg\min}\, \mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\arg\min} \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)} \right)^2$$
$$= \underset{\boldsymbol{\theta}}{\arg\min}\, \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2$$



- Find via **normal equations**

$$\frac{\partial \mathcal{R}_{\text{emp}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0$$

- Solution: **ordinary-least-squares (OLS)** estimator

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$