Machine Learning

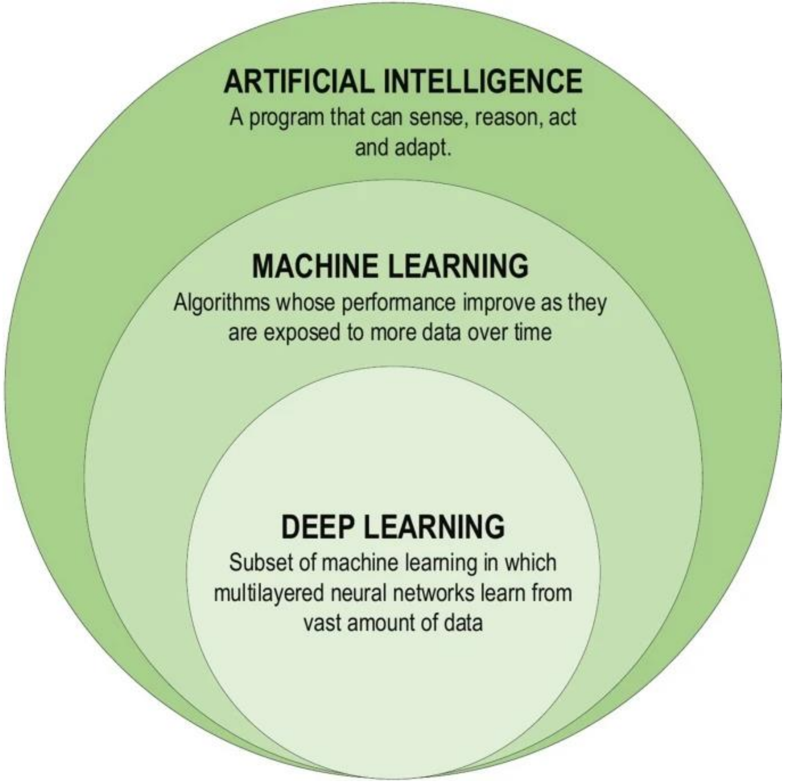
Compute Ontario Summer School

Weiguang Guan, guanw@sharcnet.ca
SHARCNet/Digital Research Alliance of Canada

AI-related courses in COSS 2024

- Data preparation
- Introduction to Scalable and Accelerated Data Analytics
- Text mining
- Machine learning (https://training.computeontario.ca/courses/course/view.php?id=94)
- Artificial Neural Networks (Deep learning)
- AI Showcases (new course)

AI, machine learning, and deep learning



Source: https://doi.org/10.1186/s40537-021-00444-8

Contents

- Morning
  - Introduction to machine learning
  - Logistic regression
  - Polynomial regression
- Afternoon
  - A case study
    - The whole cycle of developing machine learning models
    - Several machine learning methods are used
    - Evaluate and compare different models

Goals:

- What is model?
- How machine learns a model?

Where we run the demos

Any clusters (Graham, Cedar, Beluga, Narval)

- Python 3.11.5
- Jupyterhub (https://jupyterhub.sharcnet.ca)
- Tensorflow 2.15.1, scikit-learn 1.3.1, numpy, matplotlib, etc

NOTE: Most of other versions of the above tools would work as well.

Reference

- *Machine learning course*, by Andrew Ng on Coursera
- *Machine Learning Crash Course*, by Google (https://developers.google.com/machine-learning/crash-course)
- *Overfitting vs. Underfitting*, by Scikit-learn (https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html)

What's machine learning

x	-10	0	10	20	50	...
y	14	32	50	68	122	...

What is y if x=100?

Fahrenheit (y) = Celsius (x)*1.8+32

y = x*1.8+32 = 100*1.8+32 = 212 when x=100.

What's machine learning (cont.)

"**The field of study that gives computers the ability to learn without explicitly being programmed**." by AI pioneer Arthur Samuel in 1959.

Tasks that can be explicitly programmed:

- Solve quadratic equation $ax^2 + bx + c = 0$
- Sort a list of numbers

Tasks that can't be explicitly programmed:

- Face detection and recognition
- Speech recognition

Machine learning (cont.)

- Supervised learning
  - To learn a mapping $x \rightarrow y$ from samples $(x_i, y_i), i = 1,2, \dots, n$

    x_i is the input (a vector of features) of *i-th* sample,

    y_i is the corresponding target (integer representing the class, or float-point values)
- Unsupervised learning
  - To learn structures (or patterns) existing in unlabeled data
- Reinforcement learning

Supervised learning examples

x (input)	y (output or label)	Application
Email	Spam or not (1 or 0)	Spam filter *
House info	Market value	House valuation ⟊
Image	Contain dog or not (1 or 0)	Image classification *
Text in English	Text in French	Language translation ï
Stock info	Going up or down (1, 0)	Stock prediction *
Stock info	[-100%, -5%], [-5%, 0%], [0%, 5%], [5%, ∞],	Stock prediction *
Stock info	Change in percentage	Stock prediction ⟊

* Classification

⟊ Regression

ï Sequence to sequence

Applications of machine learning

- Image recognition
- Speech recognition
- Natural language processing
- Object detection and tracking in videos
- Image/video/voice synthesis/enhancement
- Medical diagnosis
- Generative AI
- ...

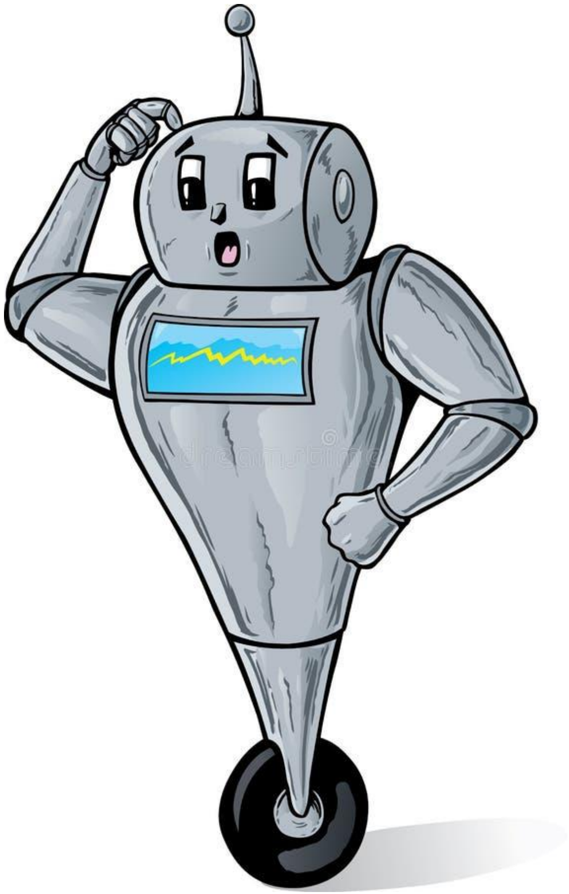
Training data

	Feature #1	Feature #2	Feature #3		Feature #n	Target
	Id	**Size (in sq feet)**	**# of bedrooms**	**…**	**Type of dwelling**	**Sold price (in thousands)**
Sample #1	2339560	3500	5	…	detached	1200
Sample #2	2356346	1600	2	…	townhouse	700
Sample #3	2356345	2600	4	…	detached	900
	…	…	…	…	…	…
Sample #m	2367758	800	1	…	condo	500

Supervised learning methods

- KNN
- Linear/logistic models
- Decision tree
- Ensemble learning (AdaBoost, Random Forest, etc)
- SVMs
- Gradient Boosting
- Neural networks
- ...

How does machine learn?



How does machine learn? (cont.)

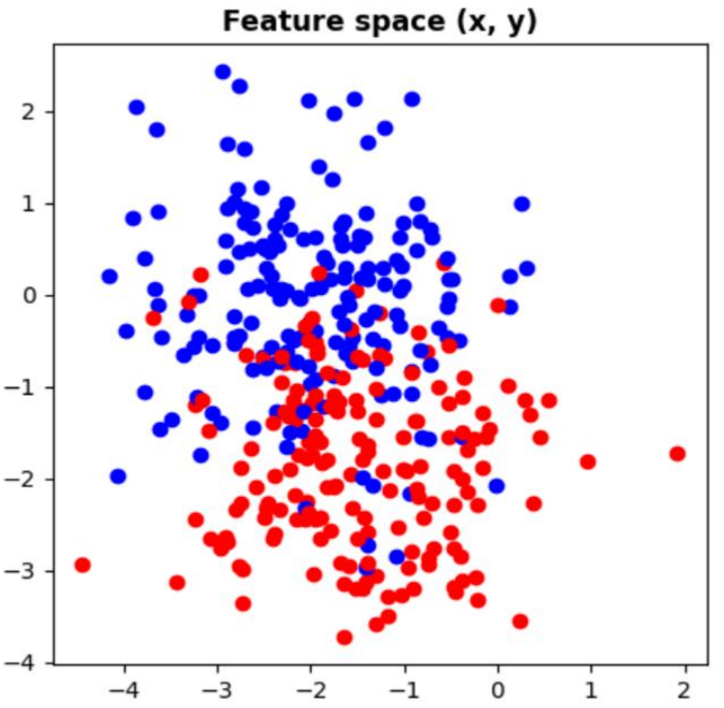
Machine learning is formulated as an optimization problem.

- Choose a model architecture
- Define a loss function of set of (trainable) parameters
- Train the model: determining the values of (trainable) parameters of the model that minimize the loss function
  - Analytical solutions (or closed-form solutions)
  - Numerical solutions (gradient-descent)

Example 1: Logistic regression

- **Task**: Classification based on two features (x, y).
- **Training data**: Two clusters of dots in 2D space, which are drawn from two normal distributions with different centers.

- **Labels**: 1 (red), 0 (blue)

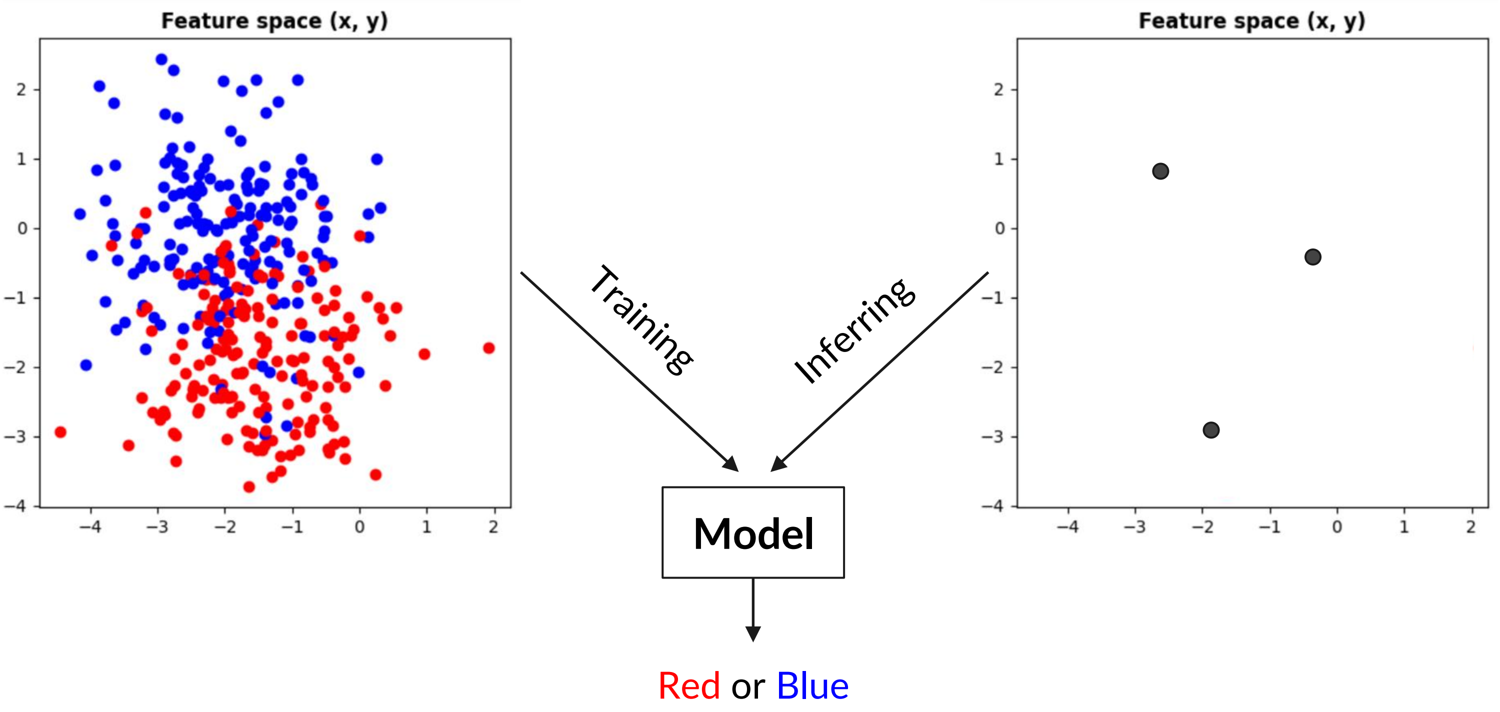


Logistic regression (cont.)

Possible use cases:

- Classifying dogs from cats based (x=weight, y=height).
- Classifying if a person has diabetes based on (x=hours of exercise, y=calorie intake)
- ...

Case study #1 (cont.)

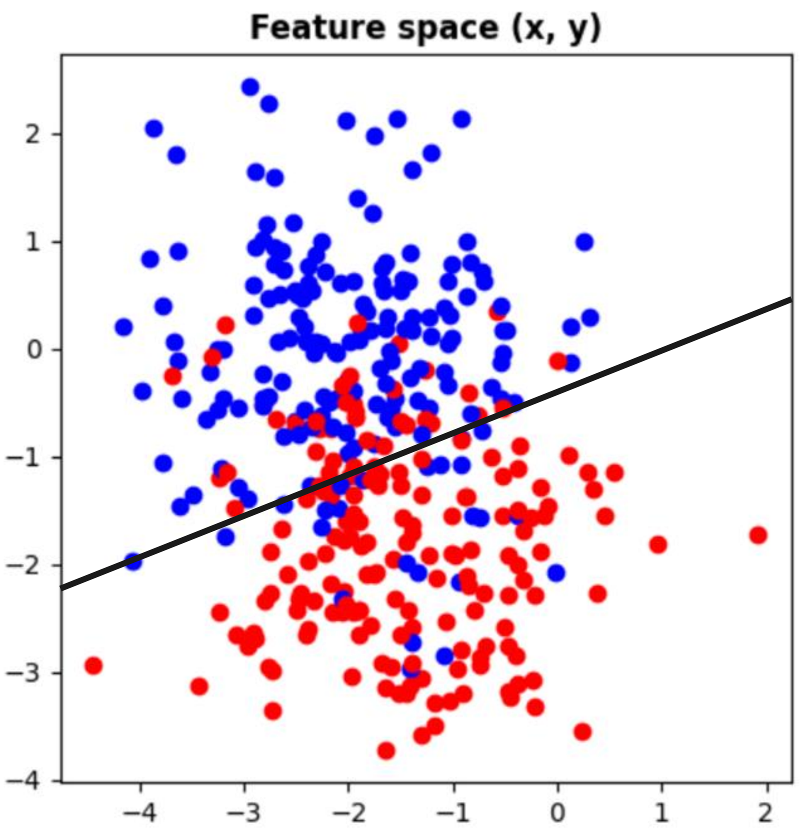


Logistic regression (cont.)

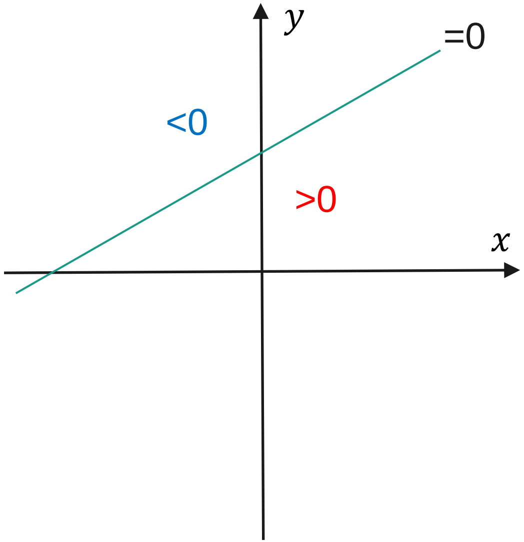
Classification model: Linear model (line that divides the 2D space into halves)

$$y = kx + b$$

$$ax + by + c = 0$$



Logistic regression (cont.)



$$ax + by + c \begin{cases} > 0, & if\ (x, y)\ on\ one\ side \\ = 0, & if\ (x, y)\ on\ the\ line \\ < 0, & if\ (x, y)\ on\ the\ other\ side \end{cases}$$

$|ax + by + c|$ becomes bigger when (x, y) is farther away from the line

Logistic regression (cont.)

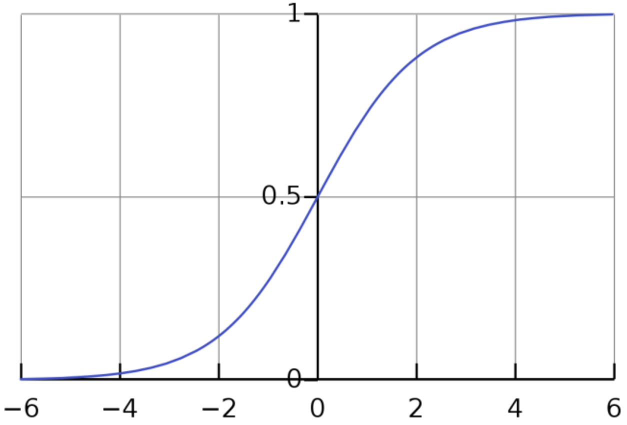
- Logistic model

$$\omega = ax + by + c$$

$$z = \frac{1}{1+e^{-\omega}}$$

- Decision boundary

$$z = \frac{1}{1+e^{-\omega}} = 0.5 \text{ if } \omega = ax + by + c = 0$$

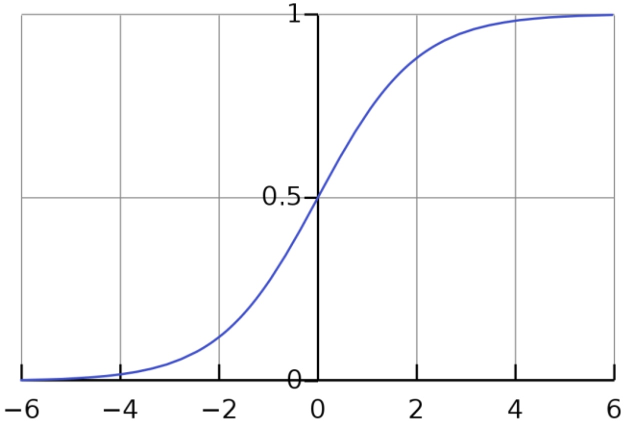


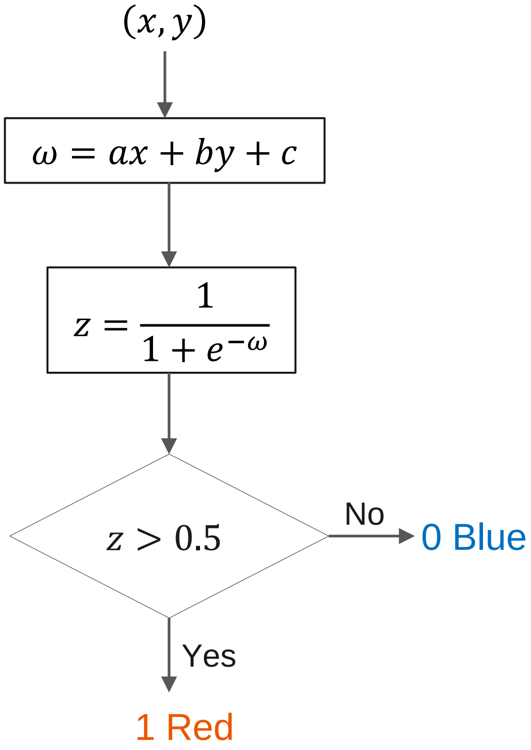
Logistic regression is a combination of a linear model and a sigmoid function

Logistic regression (cont.)

Why sigmoid function?

- It outputs values in the range of (0, 1), which can be interpreted as probability of being class "red"





Logistic regression (cont.)

We have chosen a logistic model to classify dots

$$z = \frac{1}{1 + e^{-(ax+by+c)}}$$

How to determine the proper values of those parameters a, b, c of the model?

Logistic regression (cont.)

Loss function:

Entropy: $L(a,b,c) = \frac{1}{m}\sum_{i=1}^{m} -(l_i \, log \, z_i + (1-l_i) \, log(1-z_i))$ or

MSE: $L(a,b,c) = \frac{1}{m}\sum_{i=1}^{m}(l_i - z_i)^2$

defined over dataset $\{x_i, y_i, l_i\}_{i=1}^{m}$, where

$z_i = z(x_i, y_i)$ is the output of the model on the *i-th* sample

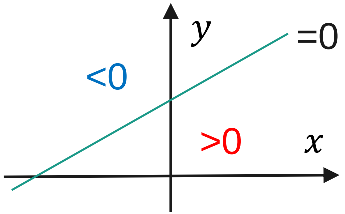
l_i is the true label of the *i-th* sample

Logistic regression (cont.)

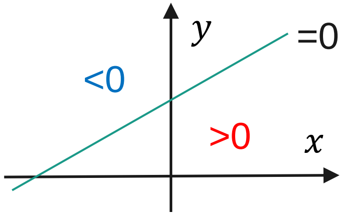
Scaling the parameters (a, b, c) by k will not change the line because they represent the same line

- $ax + by + c = 0$
- $kax + kby + kc = 0$

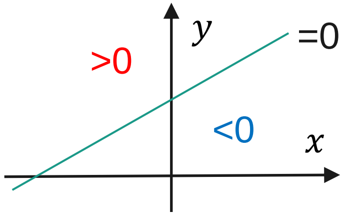
However, it will flip the signs if $k < 0$.



$z = ax + by + c$



$z = kax + kby + kc, k > 0$



$z = kax + kby + kc, k < 0$

Logistic regression (cont.)

A line in 2D space has 2 degrees of freedom.

But $ax + by + c = 0$

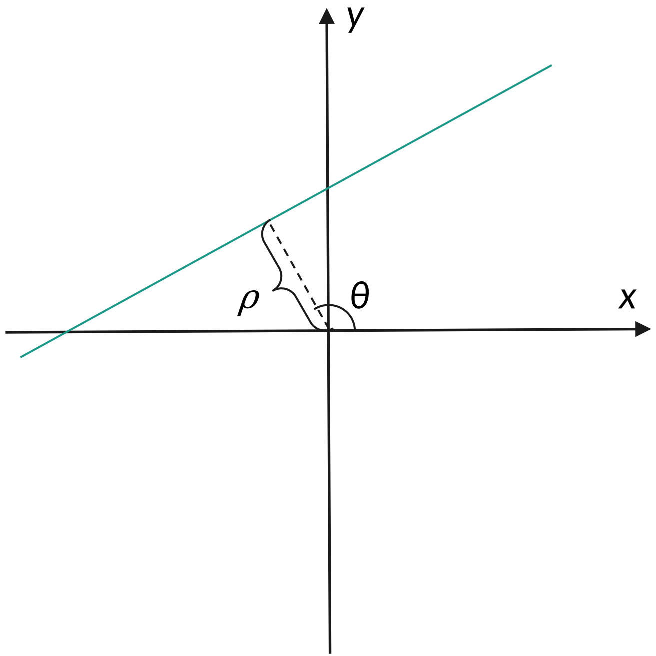
👎 Has three parameters (a, b, c)

👎 (a, b, c) is not unique for the same line

Logistic regression (cont.)

$x \cos\theta + y \sin\theta = \rho$ has two parameters (ρ, θ).

- Train with $ax + by + c$
- Convert (a, b, c) *to* (ρ, θ) and plot loss map in the parameter space of (ρ, θ) .



Logistic regression (cont.)

Conversion between $ax + by + c = 0$ and $x\cos\theta + y\sin\theta = \rho$

$$\theta = atan2(b, a)$$

$$\rho = -\frac{c}{\sqrt{a^2+b^2}}$$

$$a = \cos\theta$$

$$b = \sin\theta$$

$$c = -\rho$$

Summary of logistic regression

Training data: A set of dots in 2D space that are labelled either “0” or “1”

Data sample: a dot (x_i, y_i, $label_i$)

Features: x and y coordinates of dots

Model: $output = \frac{1}{1+e^{-(ax+by+c)}}$

Parameters of model: *a*, *b*, *c* in training (or θ, ρ in plotting)

Training: an optimization process of determining a point (*a*, *b*, *c*) in the parameter space that minimizes a pre-defined loss function

Example 2: Polynomial regression

- **Task**: Predict a target value y based on a single feature x
- **Assumption**: n-degree polynomial model between x and y:

  $y = w_1 x + w_2 x^2 + \ldots + w_n x^n + b$
- **Training data**: a set of (x_i, y_i) value pairs.

Polynomial regression (cont.)

- Use synthetic data instead of real data $y = cos(1.5\pi x) + \sigma$, where σ is a white noise
  - We know the true function behind the noisy data
  - We can easily control the shape of the true function
- Use sklearn to solve the polynomial regression instead of writing our own

$$y = w_1 \boxed{x} + w_2 \boxed{x^2} + \ldots + w_n \boxed{x^n} + b$$

$$y = w_1 \boxed{x_1} + w_2 \boxed{x_2} + \ldots + w_n \boxed{x_n} + b$$

Polynomial regression (cont.)

$$x$$

$$\downarrow$$

PolynomialFeatures(degree=n)

$$\downarrow$$

$$x, x^2, \ldots, x^n$$

$$\downarrow$$

LinearRegression()

$$\downarrow$$

$$w_1 x + w_2 x^2 + \cdots + w_n x^n + b$$

Polynomial regression (cont.)

Loss function of polynomial regression

$$J(w_1, w_2, \dots, w_n, b) = \sum_{i=1}^{m} \left(w_1 x_i + w_2 x_i^2 + \cdots + w_n x_i^n + b - y_i\right)^2$$

where (x_i, y_i) is the *i-th* sample.

Add a regularization term to the loss function in Ridge

$$J(w_1, w_2, \dots, w_n, b) = \sum_{i=1}^{m} \left(w_1 x_i + w_2 x_i^2 + \cdots + w_n x_i^n + b - y_i\right)^2 + \alpha \sum_{i=1}^{n} w_i^2$$

Exercise

- Play with “*polynomial-regression.py*” code
  - For LinearRegression() model
    - Increase number of samples (n_samples)
    - Choose different degrees (for example, 1, 2, 7, 25)
  - For Ridge(alpha=?) model
    - Change the alpha value

Summary of polynomial regression

- The degree n of a polynomial model controls the complexity (or flexibility) of the model
  - If model is too simple for the problem, then under-fitting occurs
  - If model is too complex for the problem, the over-fitting occurs
- More training samples can help to reduce the over-fitting problem.