Basics of Machine Learning

[CSED490X] Recent Trends in ML: A Large-Scale Perspective

Jungtaek Kim

jtkim@postech.ac.kr

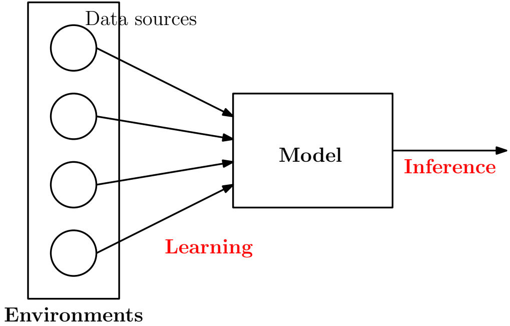
POSTECH
Pohang 37673, Republic of Korea
https://jungtaek.github.io

March 2, 2022

Table of Contents

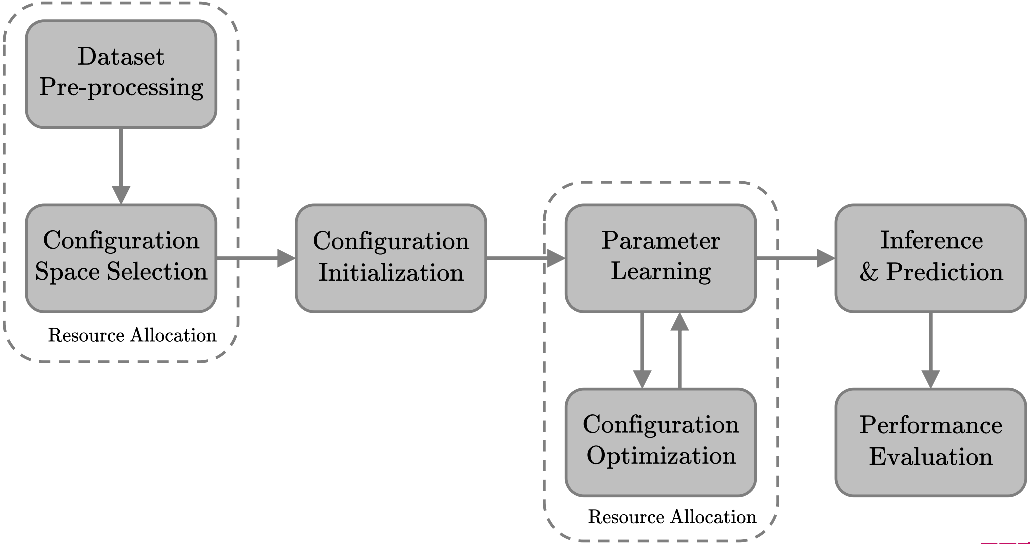
Machine Learning

What Is Machine Learning?



- Machine learning is a data-driven method for artificial intelligence.
- Three key ingredients in machine learning
  1. Data;
  2. A machine learning model;
  3. A learning algorithm.

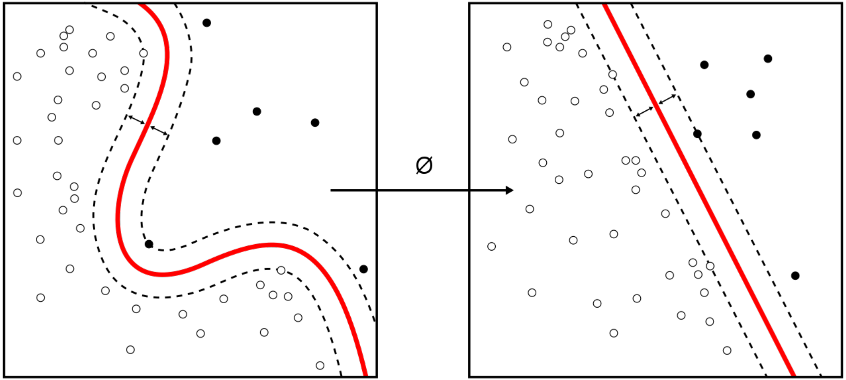
Machine Learning Pipeline



Machine Learning Taxonomy

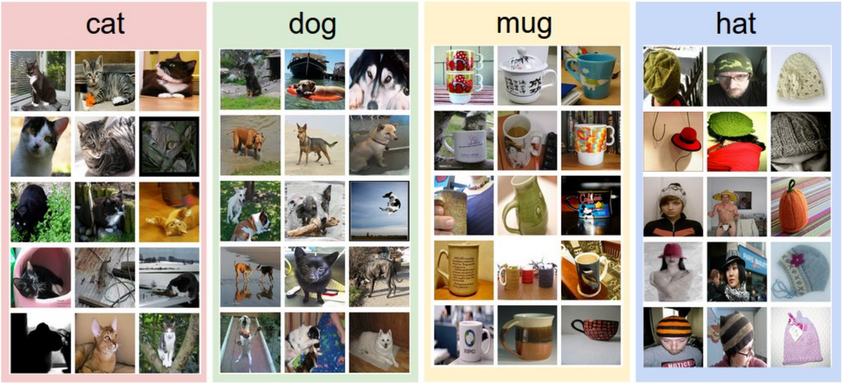
	Feedback	Goal
Supervised learning	Instructive feedback	Regression & classification
Unsupervised learning	No feedback	Representation learning & clustering
Reinforcement learning	Evaluative feedback	Sequential decision making

Classification



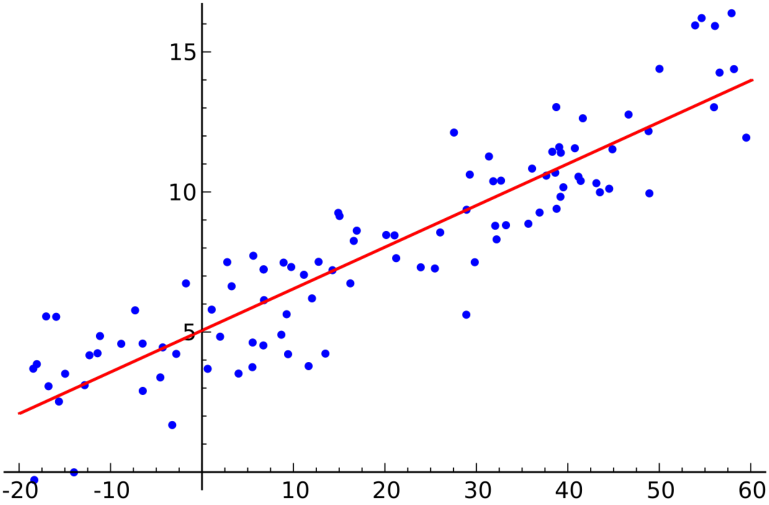
Taken from Wikipedia.

Image Classification



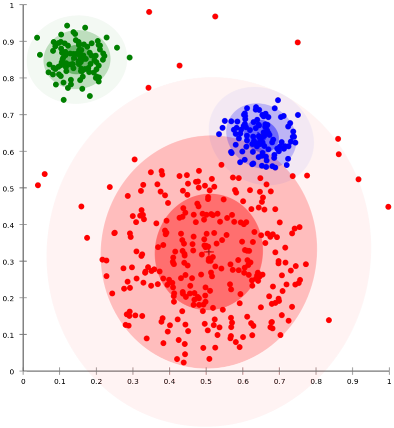
Taken from `https://cs231n.github.io`.

Regression



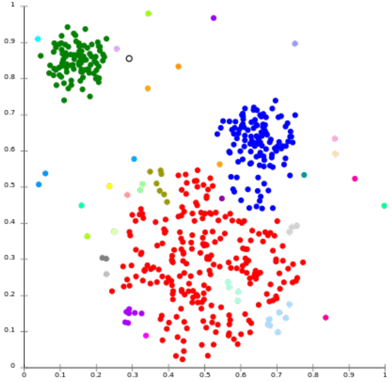
Taken from Wikipedia.

Clustering



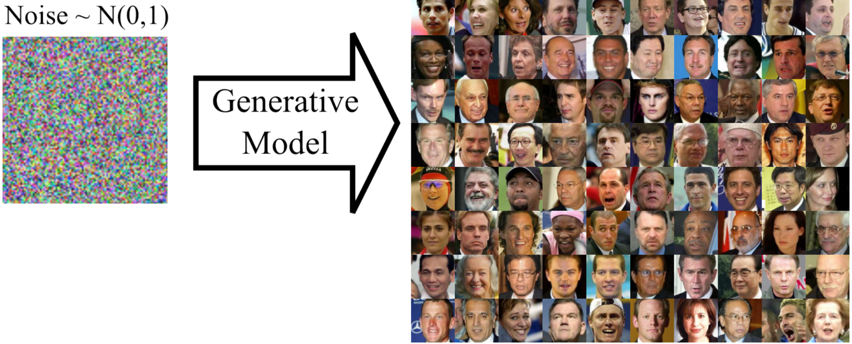
Taken from Wikipedia.

Anomaly Detection



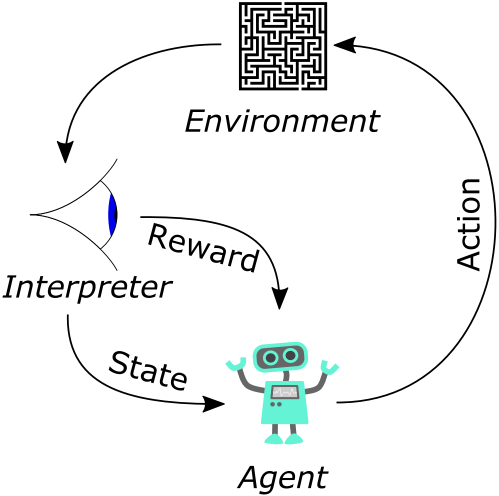
Taken from `https://iwringer.wordpress.com/2015/11/17/anomaly-detection-concepts-and-techniques/`.

Generative Models



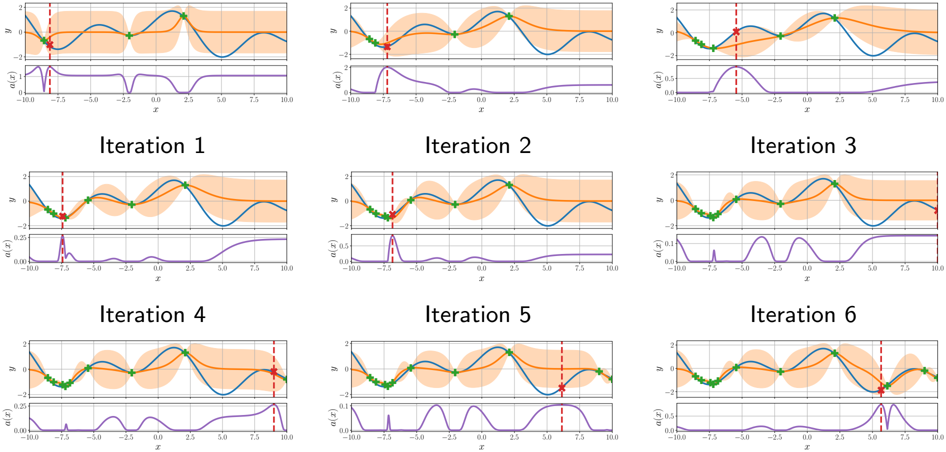
Taken from http://torch.ch/blog/2015/11/13/gan.html.

Reinforcement Learning



Taken from Wikipedia.

Bayesian Optimization



Iteration 1 Iteration 2 Iteration 3

Iteration 4 Iteration 5 Iteration 6

Iteration 7 Iteration 8 Iteration 9

Figure 1: Bayesian optimization results with the EI criterion.

First Ingredient: Data

First Ingredient: Data

- For a supervised learning case, a set of data is

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is a d-dimensional input, $y \in \mathbb{R}$ is a scalar output, and n is the number of data.

- For a unsupervised learning case, a set of data is

$$\{\mathbf{x}_i\}_{i=1}^n, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^d$ is a d-dimensional input and n is the number of data.

- For an (offline) reinforcement learning case, a set of data, i.e., a set of episodes, is

$$\{\{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}_{j+1}, r_j)\}_{j=0}^{t_i-1}\}_{i=1}^n, \tag{3}$$

where $\mathbf{s}_j$, $\mathbf{a}_j$, and r_j are state, action, and reward at iteration j, respectively.

Boston House-Prices Dataset

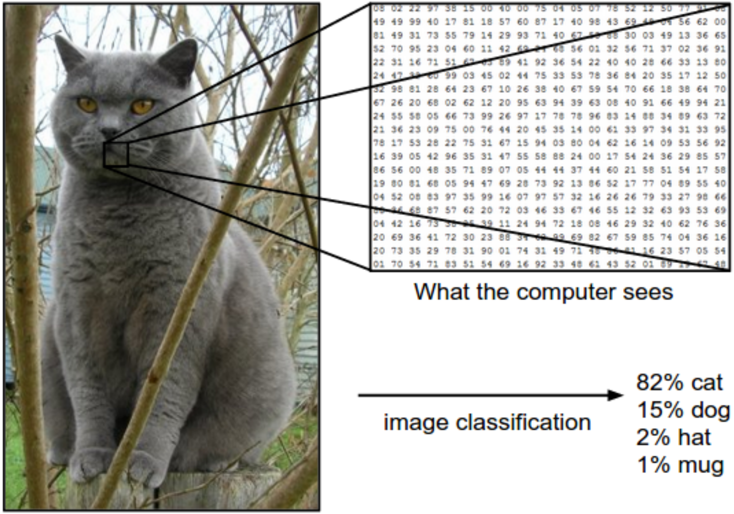
- A regression task
- Number of samples: 506
- Dimensionality: 13
- Target: medv - median value of owner-occupied homes in $1000s (target), 5.0 - 50.0
- Features:

1. crim - per capita crime rate by town
2. zn - proportion of residential land zoned for lots over 25,000 sq.ft
3. indus - proportion of non-retail business acres per town
4. chas - Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. nox - nitrogen oxides concentration (parts per 10 million)
6. rm - average number of rooms per dwelling
7. age - proportion of owner-occupied units built prior to 1940
8. dis - weighted mean of distances to five Boston employment centres
9. rad - index of accessibility to radial highways
10. tax - full-value property-tax rate per $10,000
11. ptratio - pupil-teacher ratio by town
12. black - $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
13. lstat - lower status of the population (percent)

Breast Cancer Wisconsin Dataset

- A classification task
- Number of samples: 569
- Dimensionality: 30
- Target: malignant or benign
- Features:
  - mean radius, mean texture, mean perimeter, mean area, mean smoothness, mean compactness, mean concavity, mean concave points, mean symmetry, mean fractal dimension, radius error, texture error, perimeter error, area error, smoothness error, compactness error, concavity error, concave points error, symmetry error, fractal dimension error, worst radius, worst texture, worst perimeter, worst area, worst smoothness, worst compactness, worst concavity, worst concave points, worst symmetry, worst fractal dimension

Images



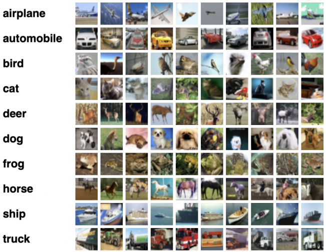
Taken from `https://cs231n.github.io`.

MNIST

- A classification task
- Number of samples: training 60,000 / test 10,000
- Dimensionality: 28×28 $(= 784)$
- Target: digits, 0 – 9

http://yann.lecun.com/exdb/mnist/

CIFAR-10 / CIFAR-100



(a) CIFAR-10

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

(b) CIFAR-100

Figure 2: CIFAR-10 / CIFAR-100

- A classification task
- Number of samples: training 50,000 / test 10,000
- Dimensionality: $32 \times 32 \times 3$ $(= 3{,}072)$
- Target: category, 10 (CIFAR-10) / 100 (CIFAR-100)

---

https://www.cs.toronto.edu/~kriz/cifar.html

Large-Scale CelebFaces Attributes (CelebA) Dataset

- A classification task
- Number of samples: 202,599
- Dimensionality: $178 \times 218 \times 3$ $(= 116{,}412)$
- Target: person identification, 10,177
- Note: 5 landmark locations, 40 binary attributes annotations per image

https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

Tensors

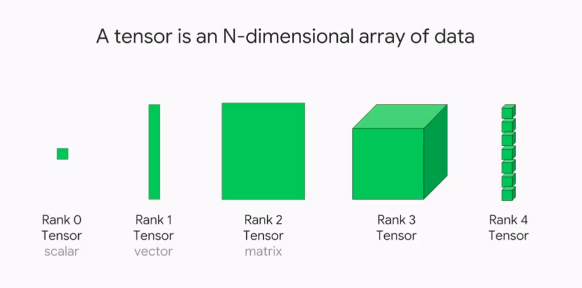


Figure 3: Illustration of tensors

- A k-dimensional array of data is defined as a tensor of order (or rank) k $\in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_k}$.

Figure 3 is taken from `https://medium.com/mlait/tensors-representation-of-data-in-neural-networks-bbe8a711b93b`.

Second Ingredient: A Machine Learning Model

Second Ingredient: A Machine Learning Model

- A supervised learning model considers the dependence of a scalar response $\mathbf{y} \in \mathbb{R}^n$ on a covariate $\mathbf{X} \in \mathbb{R}^{n \times d}$:

$$y = f(\mathbf{x}) + \epsilon, \tag{4}$$

  where ϵ is an observation noise.

- A multi-output (or multi-class) expansion is

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}, \tag{5}$$

  where $\mathbf{y} \in \mathbb{R}^k$ a k-dimensional output and $\boldsymbol{\epsilon} \in \mathbb{R}^k$ is an observation noise vector.

Linear Regression

- Linear regression is a linear model over basis functions $\boldsymbol{\phi}(\mathbf{x})$:

$$f(\mathbf{x}) = \sum_{j=0}^{M} w_j \phi_j(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \tag{6}$$

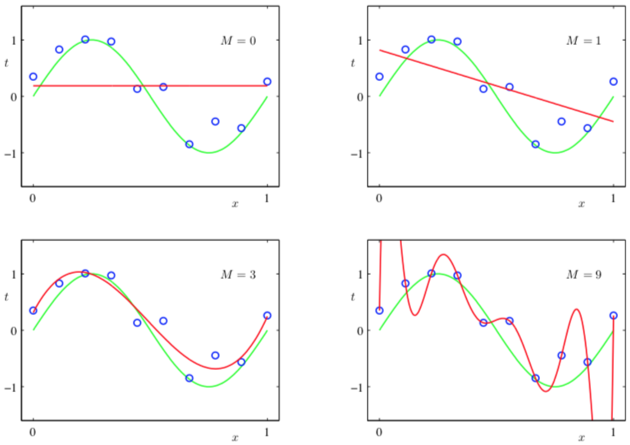
where $\phi_j(\cdot)$ is a basis function and

$$\begin{aligned} \mathbf{w} &= \left[w_0, w_1, \ldots, w_M\right], \\ \boldsymbol{\phi}(\cdot) &= \left[\phi_0(\cdot), \phi_1(\cdot), \ldots, \phi_M(\cdot)\right]. \end{aligned}$$

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Polynomial Regression

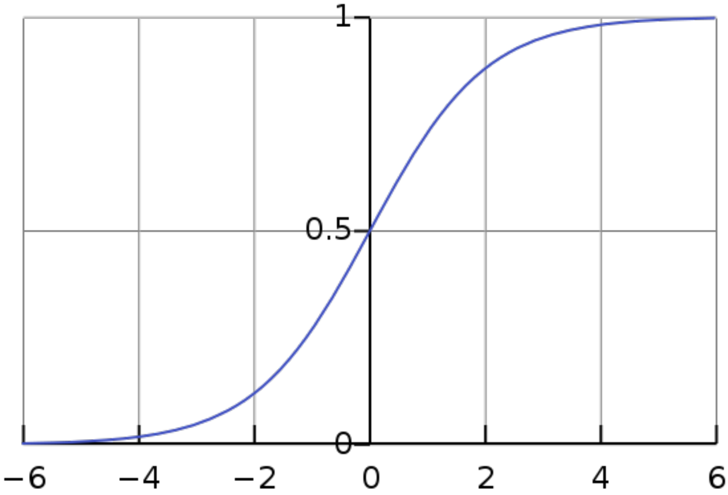
- $y = \sum_{j=0}^{M} w_j \phi_j(x) = \sum_{j=0}^{M} w_j x^j.$



Logistic Regression

- Logistic regression is a regression task, of which the output is bounded in $[0, 1]$.
- It is used for a binary classification task.
- It is defined as
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}, \tag{7}$$
where $\mathbf{w} \in \mathbb{R}^d$ is a learnable parameter and $\mathbf{w} \in \mathbb{R}^d$ is a data point.
- A class probability is computed with a logistic function.
- A classifier predicts a class label by
$$t(\mathbf{x}) = \begin{cases} 0 & \text{if } \sigma(f(\mathbf{x})) \leq 0.5, \\ 1 & \text{otherwise}, \end{cases} \tag{8}$$
where σ is a logistic function.

Logistic Function



Taken from Wikipedia.

Expansion to Multi-Class Classification

- Multi-class classification is defined with a multinomial distribution.
- It is defined as
$$f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}, \tag{9}$$
where $\mathbf{W} \in \mathbb{R}^{d \times k}$ is a learnable parameter and $\mathbf{w} \in \mathbb{R}^d$ is a data point. Note that k is the number of classes.
- A class probability is computed with a softmax function:
$$\sigma(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}, \tag{10}$$
where $\mathbf{z} = [z_1, \ldots, z_k]$.
- By the definition of the softmax function, $\sum_{i=1}^{k} \sigma(z_i) = 1$.
- Finally, a classifier predicts a class label by $t(\mathbf{x}) = \arg\max_{i \in [k]} \sigma(z_i)$.

Multilayer Perceptron

- Each layer is a fully-connected layer (a.k.a. a dense layer).
- It usually contains non-linear transformation.
- Given $\mathbf{x} = [x_1, x_2, \ldots, x_d] \in \mathbb{R}^d$, three-layer multilayer perceptron is defined as

$$f(\mathbf{x}) = \sigma\left(\mathbf{W}_3^\top \sigma\left(\mathbf{W}_2^\top \sigma\left(\mathbf{W}_1^\top \mathbf{x}\right)\right)\right) \in \mathbb{R}^k, \tag{11}$$

where σ is an activation function, $\mathbf{W}_1 \in \mathbb{R}^{d \times d_1}$, $\mathbf{W}_2 \in \mathbb{R}^{d_1 \times d_2}$, and $\mathbf{W}_3 \in \mathbb{R}^{d_2 \times k}$.

Multilayer Perceptron

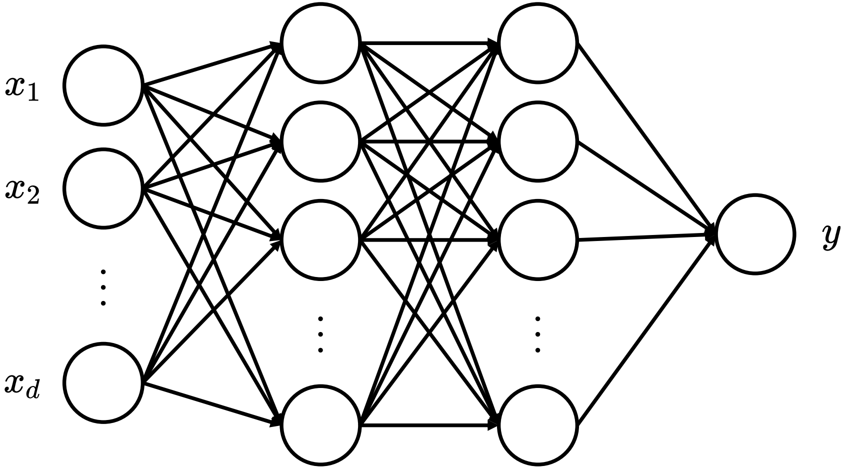
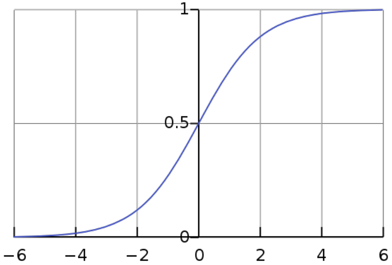


Figure 4: Case with $k = 1$.

Activation Functions

- It is a function to express the switch which has two outputs, ON and OFF.
- In a neural network field, it is non-linear and its shape is usually sigmoid.
- There are several activations such as logistic function, hyperbolic tangent function, and rectified linear unit (ReLU).



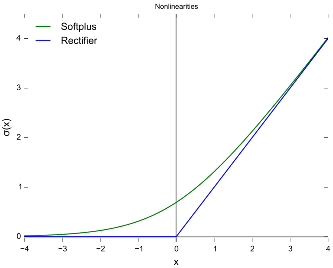


Figure 5: Activation functions: *(left)* logistic function and *(right)* ReLU.

Figure 5 is taken from Wikipedia.

Third Ingredient: A Learning Algorithm

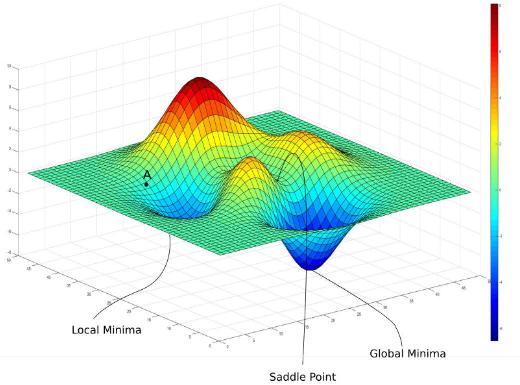
Third Ingredient: A Learning Algorithm

- To learn an optimal model, we need to optimize the following:

$$\mathbf{w}^\star = \arg\min \sum_{i=1}^{n} \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}_i), \tag{12}$$

  where $\mathcal{L}$ is a loss function (or an objective), given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$.
- We will have two questions:
  - How do we define a loss function $\mathcal{L}$?
  - How do we solve the problem (12)?

Loss Function



- Mean squared loss is

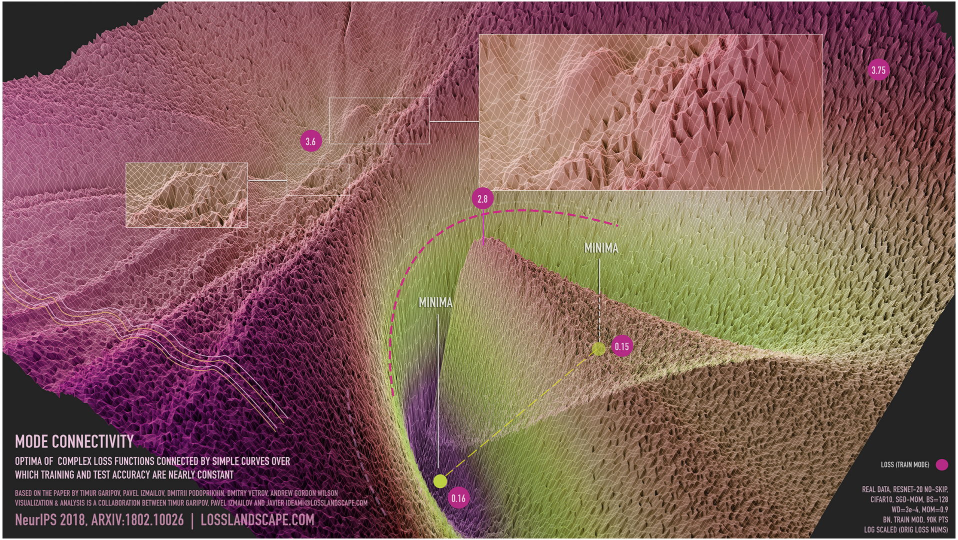
$$\mathcal{L}(\mathbf{x}, y; \mathbf{w}) = (y - f(\mathbf{x}; \mathbf{w}))^2. \tag{13}$$

- Cross-entropy is

$$\mathcal{L}(\mathbf{x}, y; \mathbf{w}) = \text{one-hot}(y)^\top \log p(\mathbf{x}), \tag{14}$$

  where $\text{one-hot}(\cdot)$ converts a category to a one-hot representation and $p(\mathbf{x})$ is a classifier output. Note that $y \in \{1, \ldots, k\}$.

Loss Function



Taken from `https://losslandscape.com`.

Mathematical Optimization

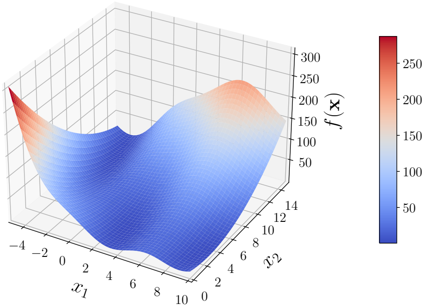


Figure 6: Branin function.

- Given an objective $f : \mathcal{A} \to \mathbb{R}$ where $\mathcal{A}$ is some set, it seeks minimum or maximum of the target function:

$$\mathbf{x}^* = \arg\min f(\mathbf{x}), \tag{15}$$

or

$$\mathbf{x}^* = \arg\max f(\mathbf{x}). \tag{16}$$

Mathematical Optimization

- To optimize an objective, we can select one of such strategies:
  - random searches;
  - gradient-based approaches;
  - convex programming;
  - evolutionary algorithms;
  - simulated annealing;
  - Bayesian optimization.
- Each strategy has the advantage in the corresponding conditions of optimization problem.

Gradients

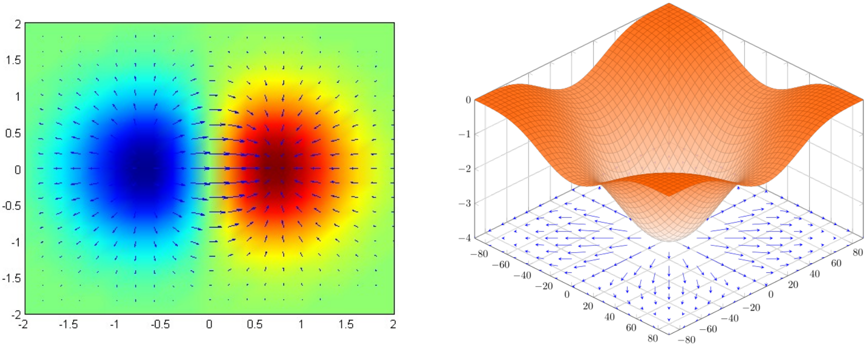


Figure 7: Illustration of gradients.

Taken from Wikipedia.

Gradient Descent

- **Gradient descent** over parameters $\mathbf{w}$ is defined as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \sum_{i=1}^{n} \frac{\partial \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w})}{\partial \mathbf{w}}, \tag{17}$$

where $\mathbf{w}_t$ is a learnable parameter at iteration t, $\mathcal{L}$ is a loss function, and γ is a learning rate.

Simple Variants of Gradient Descent

- **Stochastic gradient descent** over parameters $\mathbf{w}$ is defined as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \frac{\partial \mathcal{L}(\mathbf{x}, y; \mathbf{w})}{\partial \mathbf{w}}, \tag{18}$$

  where $(\mathbf{x}, y)$ is randomly selected from $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$.

- **Mini-batch gradient descent** over parameters $\mathbf{w}$ is defined as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \sum_{i=1}^{b} \frac{\partial \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w})}{\partial \mathbf{w}}, \tag{19}$$

  where b is batch size and each mini-batch is sampled from $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Note that $b \leq n$.

Adam Optimizer

- It uses the estimations of first and second moments of gradient to adapt a learning rate for each weight [Kingma and Ba, 2015].
- Adam = momentum + bias correction + RMSProp.
- It is defined as

$$\mathbf{v}_t = \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1)[\nabla \mathcal{L}(\mathbf{w}_{t-1})], \tag{20}$$

$$\mathbf{r}_t = \beta_2 \mathbf{r}_{t-1} + (1 - \beta_2)[\nabla \mathcal{L}(\mathbf{w}_{t-1})]^2, \quad \text{(element-wise square)} \tag{21}$$

$$\mathbf{v}_t^{\mathrm{bc}} = \frac{\mathbf{v}_t}{1 - \beta_1^t}, \tag{22}$$

$$\mathbf{r}_t^{\mathrm{bc}} = \frac{\mathbf{r}_t}{1 - \beta_2^t}, \tag{23}$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \alpha \mathbf{v}_t^{\mathrm{bc}} / \sqrt{\mathbf{r}_t^{\mathrm{bc}}}, \quad \text{(element-wise division)} \tag{24}$$

where β_1, β_2, and α are hyperparameters.

[Kingma and Ba, 2015] D. P. Kingma and J. L. Ba. ADAM: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, California, USA, 2015.

List of Optimizers

Table 2: List of optimizers considered for our benchmark. This is only a subset of all existing methods for deep learning.

Name	Ref.	Name	Ref.
AcceleGrad	(Levy et al., 2018)	HyperAdam	(Wang et al., 2019b)
ACClip	(Zhang et al., 2020)	K-BFGS/K-BFGS(L)	(Goldfarb et al., 2020)
AdaAlter	(Xie et al., 2019)	KF-QN-CNN	(Ren & Goldfarb, 2021)
AdaBatch	(Devarakonda et al., 2017)	KFAC	(Martens & Grosse, 2015)
AdaBayes/AdaBayes-SS	(Aitchison, 2020)	KFLR/KFRA	(Botev et al., 2017)
AdaBelief	(Zhuang et al., 2020)	L4Adam/L4Momentum	(Rolínek & Martius, 2018)
AdaBlock	(Yun et al., 2019)	LAMB	(You et al., 2020)
AdaBound	(Luo et al., 2019)	LaProp	(Ziyin et al., 2020)
AdaComp	(Chen et al., 2018)	LARS	(You et al., 2017)
Adadelta	(Zeiler, 2012)	LHOPT	(Almeida et al., 2021)
Adafactor	(Shazeer & Stern, 2018)	LookAhead	(Zhang et al., 2019)
AdaFix	(Bae et al., 2019)	M-SVAG	(Balles & Hennig, 2018)
AdaFom	(Chen et al., 2019a)	MADGRAD	(Defazio & Jelassi, 2021)
AdaFTRL	(Orabona & Pál, 2015)	MAS	(Landro et al., 2020)
Adagrad	(Duchi et al., 2011)	MEKA	(Chen et al., 2020b)
ADAHESSIAN	(Yao et al., 2020)	MTAdam	(Malkiel & Wolf, 2020)
Adai	(Xie et al., 2020)	MVRC-1/MVRC-2	(Chen & Zhou, 2020)
AdaLoss	(Teixeira et al., 2019)	Nadam	(Dozat, 2016)
Adam	(Kingma & Ba, 2015)	NAMSB/NAMSG	(Chen et al., 2019b)
Adam^{+}	(Liu et al., 2020b)	ND-Adam	(Zhang et al., 2017a)
AdamAL	(Tao et al., 2019)	Nero	(Liu et al., 2021b)
AdaMax	(Kingma & Ba, 2015)	Nesterov	(Nesterov, 1983)
AdamBS	(Liu et al., 2020c)	Noisy Adam/Noisy K-FAC	(Zhang et al., 2018)
AdamNC	(Reddi et al., 2018)	NosAdam	(Huang et al., 2019)

Taken from [Schmidt et al., 2021].

[Schmidt et al., 2021] R. M. Schmidt, F. Schneider, and P. Hennig. Descending through a crowded valley – benchmarking deep learning optimizers. In Proceedings of the International Conference on Machine Learning (ICML), pages 9367–9376, Virtual, 2021.

List of Optimizers (Cont.)

AdaMod	(Ding et al., 2019)	Novograd	(Ginsburg et al., 2019)
AdamP/SGDP	(Heo et al., 2021)	NT-SGD	(Zhou et al., 2021b)
AdamT	(Zhou et al., 2020)	Padam	(Chen et al., 2020a)
AdamW	(Loshchilov & Hutter, 2019)	PAGE	(Li et al., 2020b)
AdamX	(Tran & Phong, 2019)	PAL	(Mutschler & Zell, 2020)
ADAS	(Eliyahu, 2020)	PolyAdam	(Orvieto et al., 2019)
AdaS	(Hosseini & Plataniotis, 2020)	Polyak	(Polyak, 1964)
AdaScale	(Johnson et al., 2020)	PowerSGD/PowerSGDM	(Vogels et al., 2019)
AdaSGD	(Wang & Wiens, 2020)	Probabilistic Polyak	(de Roos et al., 2021)
AdaShift	(Zhou et al., 2019)	ProbLS	(Mahsereci & Hennig, 2017)
AdaSqrt	(Hu et al., 2019)	PStorm	(Xu, 2020)
Adathm	(Sun et al., 2019)	QHAdam/QHM	(Ma & Yarats, 2019)
AdaX/AdaX-W	(Li et al., 2020a)	RAdam	(Liu et al., 2020a)
AEGD	(Liu & Tian, 2020)	Ranger	(Wright, 2020b)
ALI-G	(Berrada et al., 2020)	RangerLars	(Grankin, 2020)
AMSBound	(Luo et al., 2019)	RMSProp	(Tieleman & Hinton, 2012)
AMSGrad	(Reddi et al., 2018)	RMSterov	(Choi et al., 2019)
AngularGrad	(Roy et al., 2021)	S-SGD	(Sung et al., 2020)
ArmijoLS	(Vaswani et al., 2019)	SAdam	(Wang et al., 2020b)
ARSG	(Chen et al., 2019b)	Sadam/SAMSGrad	(Tong et al., 2019)
ASAM	(Kwon et al., 2021)	SALR	(Yue et al., 2020)
AutoLRS	(Jin et al., 2021)	SAM	(Foret et al., 2021)
AvaGrad	(Savarese et al., 2019)	SC-Adagrad/SC-RMSProp	(Mukkamala & Hein, 2017)
BAdam	(Salas et al., 2018)	SDProp	(Ida et al., 2017)
BGAdam	(Bai & Zhang, 2019)	SGD	(Robbins & Monro, 1951)
BPGrad	(Zhang et al., 2017b)	SGD-BB	(Tan et al., 2016)
BRMSProp	(Aitchison, 2020)	SGD-G2	(Ayadi & Turinici, 2020)
BSGD	(Hu et al., 2020)	SGDEM	(Ramezani-Kebrya et al., 2021)

Taken from [Schmidt et al., 2021].

[Schmidt et al., 2021] R. M. Schmidt, F. Schneider, and P. Hennig. Descending through a crowded valley – benchmarking deep learning optimizers. In Proceedings of the International Conference on Machine Learning (ICML), pages 9367–9376, Virtual, 2021.

List of Optimizers (Cont.)

C-ADAM	(Tutunov et al., 2020)	SGDHess	(Tran & Cutkosky, 2021)
CADA	(Chen et al., 2021)	SGDM	(Liu & Luo, 2020)
Cool Momentum	(Borysenko & Byshkin, 2020)	SGDR	(Loshchilov & Hutter, 2017)
CProp	(Preechakul & Kijsirikul, 2019)	SHAdagrad	(Huang et al., 2020)
Curveball	(Henriques et al., 2019)	Shampoo	(Anil et al., 2020; Gupta et al., 2018)
Dadam	(Nazari et al., 2019)	SignAdam++	(Wang et al., 2019a)
DeepMemory	(Wright, 2020a)	SignSGD	(Bernstein et al., 2018)
DGNOpt	(Liu et al., 2021a)	SKQN/S4QN	(Yang et al., 2020)
DiffGrad	(Dubey et al., 2020)	SM3	(Anil et al., 2019)
EAdam	(Yuan & Gao, 2020)	SMG	(Tran et al., 2020)
EKFAC	(George et al., 2018)	SNGM	(Zhao et al., 2020)
Eve	(Hayashi et al., 2018)	SoftAdam	(Fetterman et al., 2019)
Expectigrad	(Daley & Amato, 2020)	SRSGD	(Wang et al., 2020a)
FastAdaBelief	(Zhou et al., 2021a)	Step-Tuned SGD	(Castera et al., 2021)
FRSGD	(Wang & Ye, 2020)	SWATS	(Keskar & Socher, 2017)
G-AdaGrad	(Chakrabarti & Chopra, 2021)	SWNTS	(Chen et al., 2019c)
GADAM	(Zhang & Gouza, 2018)	TAdam	(Ilboudo et al., 2020)
Gadam	(Granziol et al., 2020)	TEKFAC	(Gao et al., 2020)
GOALS	(Chae et al., 2021)	VAdam	(Khan et al., 2018)
GOLS-I	(Kafka & Wilke, 2019)	VR-SGD	(Shang et al., 2020)
Grad-Avg	(Purkayastha & Purkayastha, 2020)	vSGD-b/vSGD-g/vSGD-l	(Schaul et al., 2013)
GRAPES	(Dellaferrera et al., 2021)	vSGD-fd	(Schaul & LeCun, 2013)
Gravilon	(Kelterborn et al., 2020)	WNGrad	(Wu et al., 2018)
Gravity	(Bahrami & Zadeh, 2021)	YellowFin	(Zhang & Mitliagkas, 2019)
HAdam	(Jiang et al., 2019)	Yogi	(Zaheer et al., 2018)

Taken from [Schmidt et al., 2021].

[Schmidt et al., 2021] R. M. Schmidt, F. Schneider, and P. Hennig. Descending through a crowded valley – benchmarking deep learning optimizers. In Proceedings of the International Conference on Machine Learning (ICML), pages 9367–9376, Virtual, 2021.

Integration of All Ingredients

Linear Regression: Least Mean Square

- Least mean square is a gradient descent method which minimizes the instantaneous error $\mathcal{L}_t$, where

$$\mathcal{L}_{\text{LS}} = \sum_{t=1}^{N} \mathcal{L}_t = \frac{1}{2} \sum_{t=1}^{N} \left(y_t - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_t) \right)^2 . \tag{25}$$

- The gradient descent method leads to the updating rule for $\mathbf{w}$ that is of the form

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \eta \nabla \mathcal{L}_t \\ &\leftarrow \mathbf{w} + \eta \left(y_t - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_t) \right) \boldsymbol{\phi}(\mathbf{x}_t), \end{aligned} \tag{26}$$

where $\eta > 0$ is a hyperparameter, referred to as a learning rate.

Any Questions?

References I

D. P. Kingma and J. L. Ba. ADAM: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, California, USA, 2015.

R. M. Schmidt, F. Schneider, and P. Hennig. Descending through a crowded valley – benchmarking deep learning optimizers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 9367–9376, Virtual, 2021.