Deep Learning: Part 1

東京海洋大学 TUMSAT

竹縄知之 Tomoyuki Takenawa

目次

1 Introduction to deep learning 3

1.1 In a nutshell 4

1.2 Three basic principles of neural network learning 5

1.3 Towards deeper networks 14

1.4 Demonstration movie 17

1.5 Exercises using Scikit-learn and Keras 19

2 What is machine larning? 20

2.1 What is a learning algorithm? 21

2.2 Calculating the regression line by stochastic fradient descent method 25

2.3 Exercises on stochastic gradient descent with NumPy 27

3 Information theory 28

3.1 Introduction to probability theory and statistics 29

3.2 Basics of information theory 36

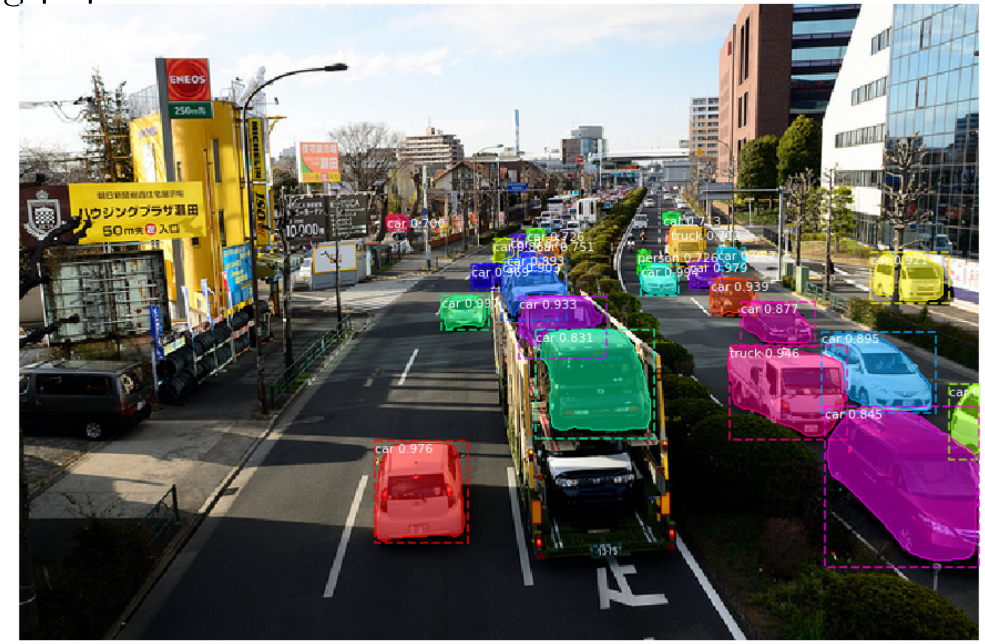
1 Introduction to deep learning

In this course, we will study the techniques used in deep learning in detail, but since each topic is quite heavy, there is a concern that we may lose sight of what we are learning. So, let's start with an overview of deep learning to get an overall picture.

In the first half of this course, we will look at the contents of this section in more detail. In the following, some new terms may appear without explanation, but we will explain them again later, so please put them aside for now and move on.

1.1 In a nutshell

- Machine learning with complex neural networks with many layers
- In recent years, it has become possible to analyze images (analyzing the contents of images and extracting objects from images) and analyze voice and text (analyzing the contents and translating) with the same or higher level of accuracy than humans, and this technology is rapidly becoming popular.

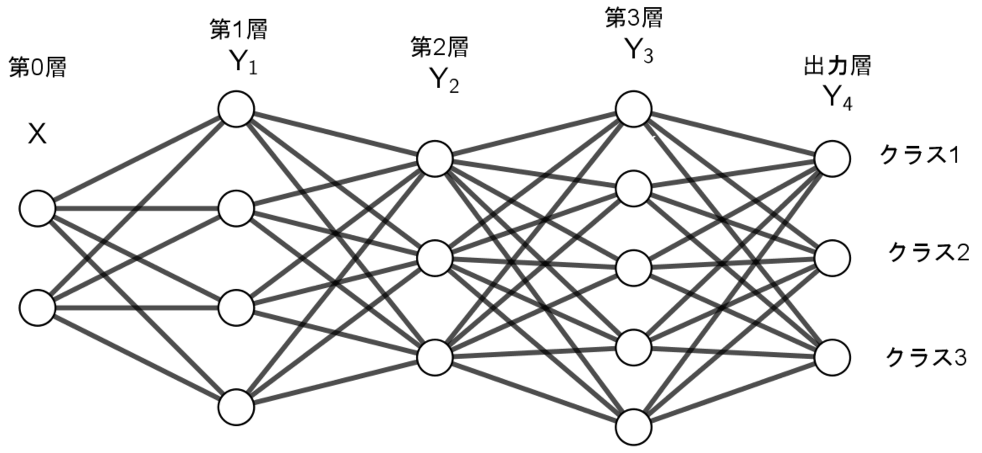


Mask R-CNN

1.2 Three basic principles of neural network learning

1. Computation of predictions from input data by iterative application of linear transformations and activation functions (Required knowledge: linear algebra)
2. Differentiation of the error between the prediction and the correct answer with respect to the parameters of the network (Required knowledge: probability, statistics, information theory, differentiation of multiple variables)
3. Update parameters using stochastic gradient descent method (Required knowledge: probability, optimization)

Input $\Rightarrow$

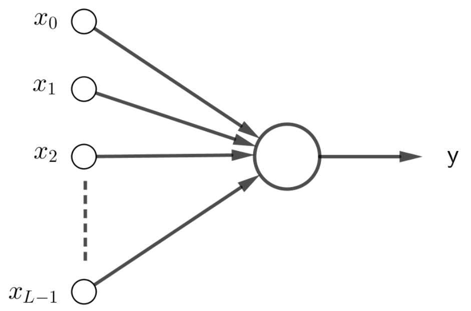


$\Rightarrow$Output

1. Computation of predictions using linear transformations and activation functions

Perceptron

The perceptron is an algorithm that takes multiple signals as input and outputs a single signal.



From the vertices (neurons) lined up vertically on the left side of the diagram above, signals are sent to the neurons on the right side following the arrows, and are combined into a single signal for output.

The perceptron has "weights" $w_0, w_1, \ldots, w_{L-1}$ and "bias" b as parameters, and for input $x_0, x_1, \ldots, x_{L-1}$ from the neuron on the left side, outputs

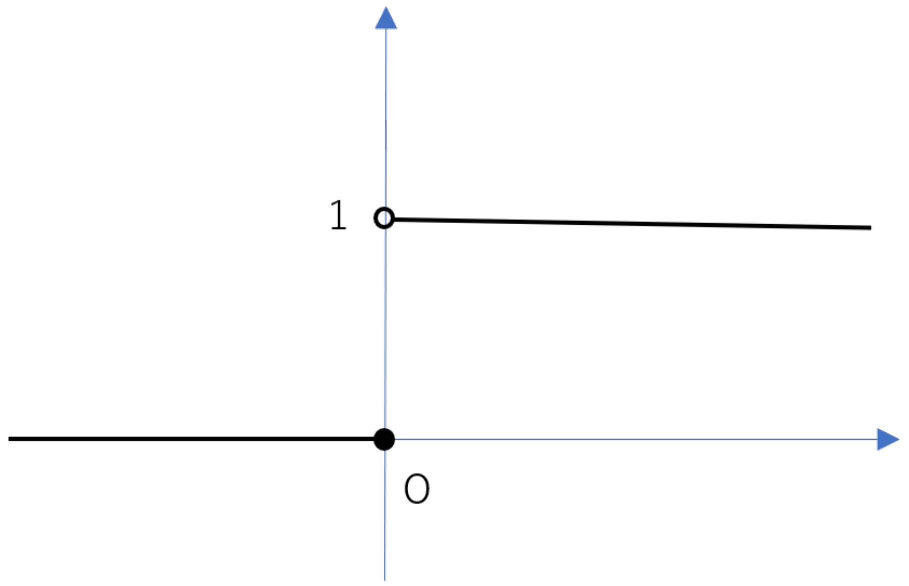
$$a = x_0 w_0 + x_1 w_1 + \cdots + x_{L-1} w_{L-1} + b$$

$$y = h(a) = \begin{cases} 1 & (a > 0) \\ 0 & (a \leq 0) \end{cases}$$

from the neuron on the right side. The $h(x)$ is a function called the activation function, which in this case is a step function.

Differentiation and learning possibilities

A perceptron is also called a neuron, and a neural network is essentially a combination of perceptrons. However, the step function is not suitable for learning because its derivative is zero except at the origin, so we need to change the activation function to one that can be differentiated.



The step function cannot be differentiated at $x = 0$, and the derivative is zero at $x \neq 0$. (Both of these hinder learning.)

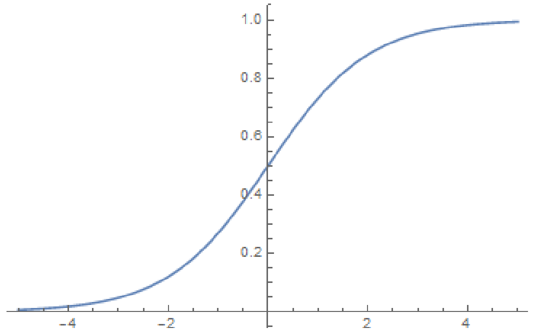
Neural networks

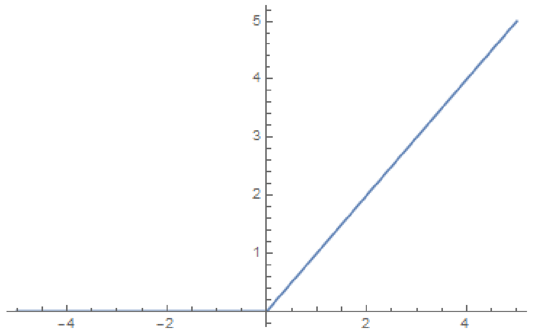
A neural network is a network consisting of multiple layers consisting of neurons (vertices of a graph) aligned in parallel. The operation in the mth neuron of layer i is a linear transformation followed by a nonlinear transformation, as in the perceptron:

$$\begin{cases} a_m^{(i)} &= x_0^{(i-1)} w_{0m}^{(i)} + x_1^{(i-1)} w_{1m}^{(i)} + \cdots + x_{L-1}^{(i-1)} w_{L-1,m}^{(i)} + b_m^{(i)} \\ x_m^{(i)} &= f\left(a_m^{(i)}\right) \end{cases}$$

where $(x_0^{(i-1)}, \cdots, x_{L-1}^{(i-1)})$ is the output value of the previous layer, $w_{lm}^{(i)}$ and $b_m^{(i)}$ are values called model weight parameters. Also, $f(a)$ is a nonlinear function called the activation function with a non-zero derivative as in the following two examples.

Sigmoid function $f(x) = \dfrac{1}{1+e^{-x}}$ (left) ReLU (Rectified Linear Unit) function $f(x) = \max\{0, x\}$ (right)





Universal approximation theorems

There is a theorem shown by [Hornik et al., 1989; Cybenko, 1989; Leshno et al., 1993]:
“An arbitrary Borel measurable function from $\mathbb{R}^d$ to $\mathbb{R}^e$ for natural numbers d and e, can be approximated with arbitrary accuracy by a feedforward neural network of d-dimensional input and e-dimensional outputs with only one hidden layer with some activation functions.’’

Here, the condition that a function is Borel measurable is satisfied if for example it is piecewise continuous from $\mathbb{R}^d$ to $\mathbb{R}^e$, so it is correct to say that all realistic functions can be approximated.

Although the universal approximation theorem guarantees that a neural network (with large hidden layers) can approximate any function, whether or not this can be achieved by learning is another matter.

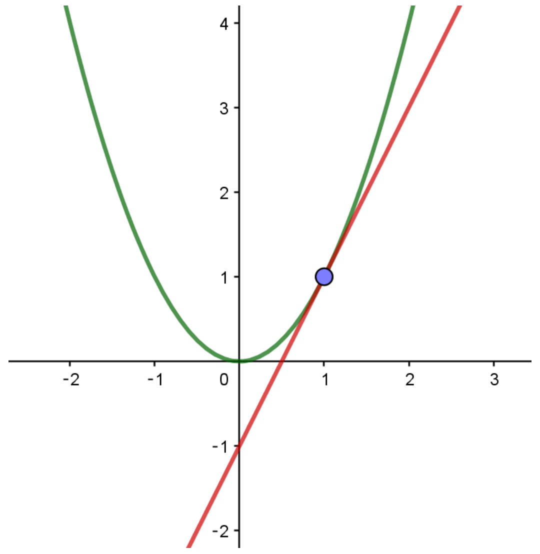
In fact, there is a theorem called the No-Free Lunch Theorem[*1] [Wolpert, D.H., Macready, W.G., 1995], which states that there is no learning method that has better generalization performance than any other model on average for all the possible problems.

[*1] Incidentally, the No Free Lunch Theorem is an unusual name, but it comes from a saying ”There ain’t no such thing as a free lunch” in R. A. Heinlein’s science-fiction novel The Moon Is a Harsh Mistress, which means ”it is impossible to get something for nothing.”

2. Derivative of the error function with respect to parameters

Derivative

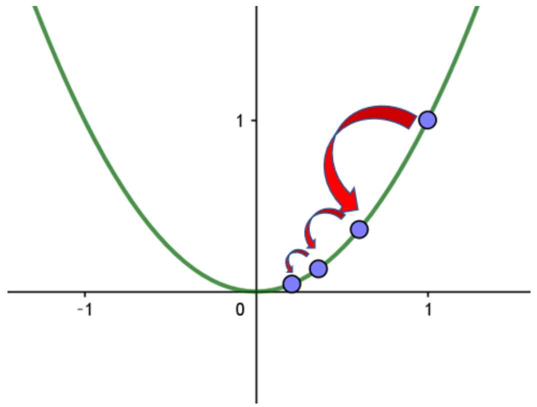
The derivative of a function is a measure of how much the output of a function changes when the input of the function is slightly changed. In the case of neural networks, it is necessary to calculate the derivatives when the weight parameter is the input and the error determined by the difference between the prediction and the correct answer is the output. Since there are many weight parameters, they are multivariable differentiations. There are also many layers, so they are derivatives of composite functions. However, deep learning libraries (such as Tnsorflow, Chainer, Pytorch, etc.) will automatically calculate these derivatives.



3. Update parameters by stochastic gradient descent method

Usual gradient descent method: find the minimum value of the function $f(x)$

1. Take a some value for x.
2. Calculate $f'(x)$(note that the derivative represents the slope of the graph)
3. For a real constant $\lambda > 0$, update x as $x \leftarrow x - \lambda f'(x)$ and return to 2



For $f(x) = x^2$, $\lambda = 0.2$, and $x = 1$, $x - \lambda f'(x) = 1 - 0.2 \times 2 = 0.6$

The point goes down the slope of the graph as the value of the function becomes smaller.

When λ is a small positive number, substituting $f(x + \Delta x) \fallingdotseq f(x) + f'(x)\Delta x$ for $\Delta x = -\lambda f'(x)$, we get $f\bigl(x - \lambda f'(x)\bigr) \fallingdotseq f(x) - \lambda\bigl(f'(x)\bigr)^2 \leq f(x)$.

Stochastic gradient decent method

Algorithm for the SGD:

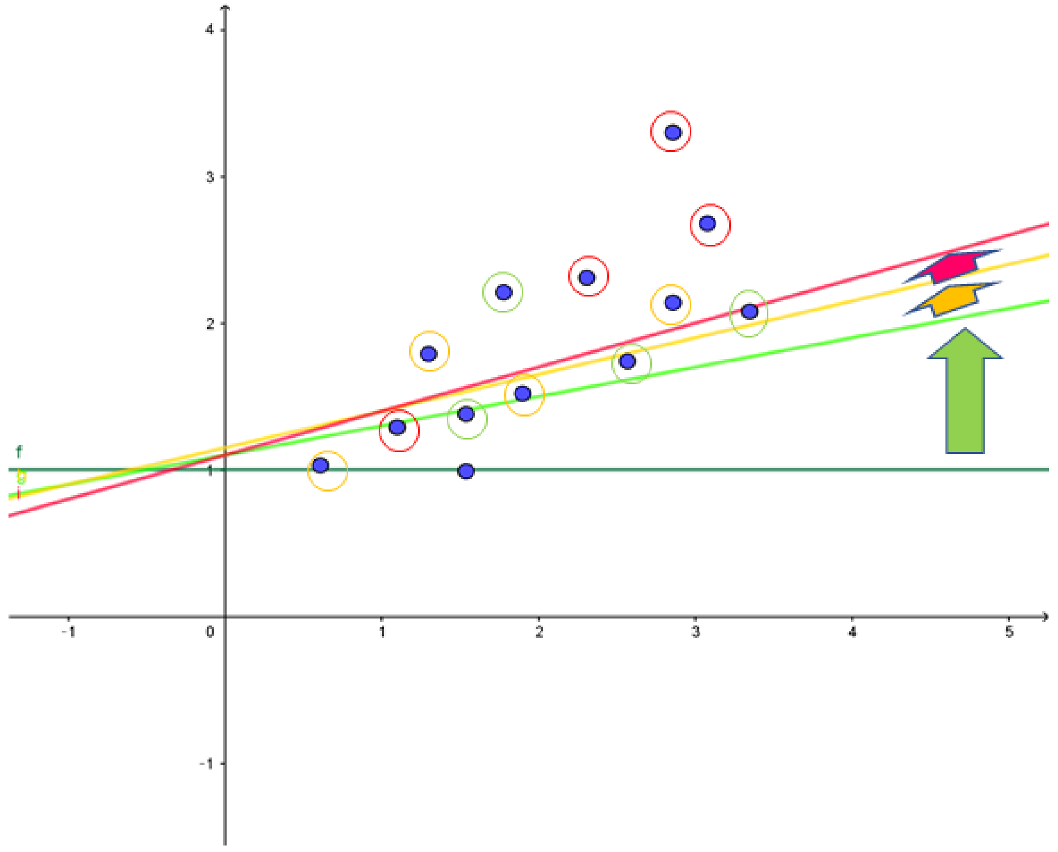
Assume that we have a set of input data with correct answers.

1. Take arbitrary value for parameter W.
2. Select a few (hundreds) samples randomly (mini-batch) from the set of input data with correct answers
3. Compute a prediction for each of the input data.
4. Calculate the derivative $\frac{\partial E}{\partial W}$, where E is the mean of the error from the correct answer in the mini-batch and W is a parameter
5. For a real constant $\lambda > 0$, update W as

   $$W \leftarrow W - \lambda \frac{\partial E}{\partial W}$$

   and return to 2

A learning method that randomly selects a set of data like this is generally called mini-batch learning. On the other hand, learning that uses all data at once is called batch learning, and learning that randomly selects one data at a time is called online learning.



Using SGD for a linear regression problem:

Some data (circled in color in the figure) are selected and the regression line is updated.

1.3 Towards deeper networks

Characteristics and problems of neural networks

- It has rich expressive power with many parameters.
- The goal of learning is not to reduce the error on the training data itself, but to correctly predict the unknown input data.
- The above two things are generally in opposition to each other, and overtraining a richly expressive predictor will often result in a loss of predictive ability (generalization performance) for unknown data. This condition is called overfitting.
- SGD converges slowly, but is advantageous in preventing overfitting. Methods such as randomly transforming the input data (data augmentation) are also effective.
- As the number of layers increases (5, 6, or more), learning does not progress well, and there are several ways to deal with this, such as improving the optimization method SGD, dropout, batch normalization, and changing the initial values of weight parameters, but changing the network structure itself is more effective. (However, these are only experimental results, and there seems to be little theoretical explanation.)

Convolutional neural networks

- A convolutional neural network is a neural network that uses convolutional operations as described in the next page.
- In a normal neural network, the output values of all the neurons in the previous layer are used to calculate the output value of a neuron. Convolutional operations use only the values of the neurons ”near” a neuron in the previous layer to help extract the ”nearness” information.
- This is the most basic method in deep learning, and many of the networks used in deep learning are convolutional neural networks and their extensions.

Convolutional operations

For example, if the input X to a layer and the weight parameter F, called the filter, is given by

$X =$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$F =$

1	1	1
2	0	1
−1	−1	0

(the gray area is the area used to calculate y_{00} below), then the output of the convolution operation becomes

$$Y = \begin{array}{|c|c|}\hline y_{00} & y_{01} \\ \hline y_{10} & y_{11} \\ \hline \end{array} = \begin{array}{|c|c|}\hline 4 & 8 \\ \hline 20 & 24 \\ \hline \end{array}$$

Here, where y_{00} is the inner product of the filter and the part of X with the same shape as that in the upper left corner (each considered as a vector)

$$\begin{aligned} y_{00} =& (1, 2, 3, 5, 6, 7, 9, 10, 11) \cdot (1, 1, 1, 2, 0, 1, -1, -1, 0) \\ =& \, 1 + 2 + 3 + 10 + 7 - 9 - 10 = 4, \end{aligned}$$

y_{01} is the part taken from X shifted by 1 to the right, y_{10} is the part shifted by 1 to the bottom, and y_{11} is the part shifted by 1 both to the right and to the bottom.

1.4 Demonstration movie

Let's take a look at a video that was created using a deep learning model called Mask R-CNN.

Follow the link from index.html.

The above is a general overview of deep learning.

Next, we will try to create and train a neural network and deep learning model using the Python library. You may have done this before, but it is surprisingly easy to do (at least with short code).

In this lecture, we will spend some time learning about the principles of these libraries.

1.5 Exercises using Scikit-learn and Keras

Notebooks:

- `1-1_Keras（と XGBoost）で回帰分析`
- `1-2_Keras で手書き数字の分類問題`

Scikit-learn is a Python library for machine learning other than deep learning, while Keras is a higher-level library mainly for making deep learning libraries such as Tensorflow easier to use.

Please keep the following points in mind when doing the exercises.

- Difference between regression and classification problems
- Difference between training data and test data
- What data are loaded and how are they loaded?
- Which method is more accurate for predicting Boston home prices: linear regression, neural networks, or decision trees?
- Which method is more accurate for handwriting recognition (MNIST): k-nearest neighbor method or convolutional neural network? Which method requires more training time?

2 What is machine larning?

small Deep learning is a type of neural network. And neural networks are a type of (statistical) machine learning. The term ”(statistical)” comes from the fact that data is viewed as a sample from a population.

In this section, we will explain what machine learning is in the first place, and discuss related terms.

2.1 What is a learning algorithm?

Definition by T. M. Mitchell [Machine Learning, 1997]:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Task T

Classification, regression (prediction of real values), transcription (extraction of discrete data such as text data from less structured data, e.g., character recognition and speech recognition), machine translation, structural output (finding the tree structure of data features and identifying the location of objects from images), anomaly detection (e.g., credit card fraud detection), synthesis and sampling (creating new samples, e.g., speech synthesis and image generation), missing value completion, noise reduction, probability density estimation, and so on. For each of the above, the task also includes the case where the input has missing values.

Performance measure P

To evaluate the capability of a machine learning algorithm, a quantitative measure of its performance must be designed. For example, model accuracy and cross-entropy error are often used for classification problems, and squared error is often used for regression problems. In addition, since it is important in machine learning whether a task is accomplished on unknown data that has not been experienced in training, the evaluation of performance should be done on data that has not been used in training (called test data). In contrast to test data, the data used for training is called training data.

Experience E

Machine learning algorithms can be broadly divided into supervised learning and unsupervised learning. Tasks that often use supervised learning include classification problems, regression problems, and machine translation. Common unsupervised learning tasks include classification problems, probability density estimation, and denoising.

Capacity of a model, overfiting, underfitting

In general, machine learning involves (1) train using the training data, (2) calculate the error on the training data (training error), and (3) Calculate the error on the test data (generalization error).

The goal of ordinary optimization methods is to reduce the training error, while the goal of machine learning is to reduce the generalization error. This means that we want to reduce the prediction error for unknown data. In order to reduce the generalization error, we need to reduce the training error as well. As a result, it can be said that the goal of machine learning is to reduce both.

After training, underfitting is a state in which the training error is not very small, and overfitting is a state in which the training error is small but the test error is not.

The number of data types that a model can identify is called its capacity. If the capacity of the model is too small, training will not be very effective and underfitting will occur, while if the capacity is too large, errors will increase and overfitting will occur for unknown data.

Hyperparameters and validation data

Most machine learning algorithms have parameters that control the behavior of the algorithm. These parameters are not changed by the learning itself. These parameters are called hyperparameters.

When a machine learning model has parameters that determine its capacity, they must not be optimized by training, but must be hyperparameters. This is because if these parameters are trained, the model will learn to increase its capacity in order to reduce the training error, which will degrade the generalization performance.

To select good hyperparameters, we need to select them in such a way that the generalization error (rather than the training error) is small, which requires a separate validation data set from the test data. This is because if we use the test data to adjust the hyperparameters, then the test data is no longer randomly selected data. As an extreme example, if you have many hyperparameters, using test data to tune them means that you are training a model that contains hyperparameters on test data.

2.2 Calculating the regression line by stochastic fradient descent method

As a simple example of a learning algorithm, let's compute the regression line using stochastic gradient descent.

The algorithm to calculate the regression line $y = ax + b$ using SGD for the dataset $(x, y) = (x_0, y_0), (x_1, y_1), \ldots, (x_{N-1}, y_{N-1})$ is as follows.

1. Take arbitrary values a, b.
2. Randomly select K samples from the data set and denote them as $(x_0, t_0), (x_1, t_1), \ldots, (x_{K-1}, t_{K-1})$.
3. Calculate the predicted value y_k by $y_k = ax_k + b$.

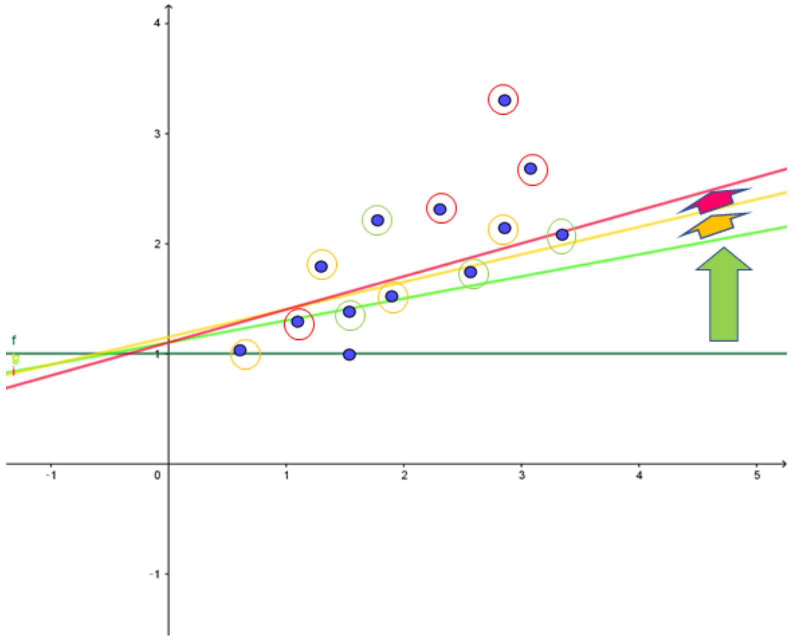
4. Calculate the derivative with respect to the parameters a and b of the squared error $E = \sum_{k=0}^{K-1} (y_k - t_k)^2$ from the correct answer by the formula

$$\frac{\partial E}{\partial a} = \frac{\partial}{\partial a} \sum_{k=0}^{K-1} (y_k - t_k)^2 = \sum_{k=0}^{K-1} 2(y_k - t_k)\frac{\partial y_k}{\partial a} = 2\sum_{k=0}^{K-1} (y_k - t_k)x_k$$

$$\frac{\partial E}{\partial b} = \frac{\partial}{\partial b} \sum_{k=0}^{K-1} (y_k - t_k)^2 = \sum_{k=0}^{K-1} 2(y_k - t_k)\frac{\partial y_k}{\partial b} = 2\sum_{k=0}^{K-1} (y_k - t_k)$$

5. For the real constant $\lambda > 0$, update the parameters a and b as follows and return to 2.

$$a \leftarrow a - \lambda\frac{\partial E}{\partial a}, \quad b \leftarrow b - \lambda\frac{\partial E}{\partial b}$$



The line changes as the algorithm progresses.

Note: The SGD is used to calculate the regression line as an example only. The regression line is usually obtained from the conditions $\dfrac{\partial E}{\partial a} = 0$ and $\dfrac{\partial E}{\partial b} = 0$.

2.3 Exercises on stochastic gradient descent with NumPy

Notebooks: `1-3_確率的勾配降下法およびその解答`

NumPy is a numerical library for Python that allows you to easily perform array (vector and matrix) calculations.

Please keep the following points in mind when performing exercises on gradient descent and stochastic gradient descent for linear regression problems using NumPy.

- Difference between gradient descent and stochastic gradient descent
- Mini-batch learning
- How to handle arrays in Numpy
- How to handle random numbers (samples following a given probability distribution) in Numpy
- Is it appropriate to use stochastic gradient descent for linear regression problems in the first place?

3 Information theory

Statistical machine learning, including neural networks, is designed based on probability theory and information theory. For example, the initial values of the parameters to be trained are taken as random numbers, which requires knowledge of probability distributions, and the design of the error function (i.e., what quantity should be reduced to ensure correct training) requires an understanding of information theory.

In this section, we will study

- Basics of probability theory
- Basics of statistics
- Information content
- Information entropy
- Cross-entropy

3.1 Introduction to probability theory and statistics

Probability distribution (discrete case)

When elementary events (an event that cannot be further divided, whole events consist of them, exclusive with each other) are $A_1, A_2, \cdots, A_K$, the value $P(A_k)$ that represents the probability of each A_k is called the probability. However, the probability of whole events is assumed to be 1.

On the other hand, when a real number $X = x_k$ corresponds to each A_k (apart from the probability), X is called a random variable. For a real number x, there may be multiple elementary events for which $X = x$. The sum of the probabilities of these events is written as $P(X = x)$, and is called the probability distribution of the random variable X.
Example: When choosing one of the four cards A, B, C, and D at random, let the scores of A and B be 1 point, C be 2 points, and D be 5 points, and let the score be the random variable X, the probability distribution of X is as follows

$$P(X = 1) = P(A) + P(B) = \frac{1}{2}$$
$$P(X = 2) = P(C) = \frac{1}{4}$$
$$P(X = 5) = P(D) = \frac{1}{4}$$

Probability distribution (continuous case)

For a real-valued variable X,

$$P(a \leq X \leq b) = \int_a^b p(x)\, dx$$

(except $P(-\infty \leq X \leq \infty) = 1$) is called a continuous probability distribution and $p(x)$ is called the probability density function.

A function $Y = f(X)$ is also a random variable.

Example: The uniform probability distribution on the interval $[-10, 10]$ is given by

$$p(x) = \frac{1}{20} \quad \text{and} \quad P(a \leq X \leq b) = \frac{b-a}{20}$$

for $-10 \leq x \leq 10$ and $-10 \leq a \leq b \leq 10$. When $Y = X^2$, we have

$$\begin{aligned} P_Y(1 \leq Y \leq 25) &= P(-5 \leq X \leq -1) + P(1 \leq X \leq 5) \\ &= \frac{8}{20} = \frac{2}{5}. \end{aligned}$$

Expected value and variance

For a random variable X, the expected value $\mu = E[X]$ and variance $V = V[X]$ are determined as follows.

In the case of a discrete probability distribution: If the possible values of X are $x_1, x_2, \cdots$, then

$$\begin{aligned} \mu = E[X] &= \sum_i x_i P(x_i) \\ V = V[X] &= E[(X-\mu)^2] = \sum_i (x_i - \mu)^2 P(x_i) \\ &= E[X^2] - \mu^2 \end{aligned}$$

In the case of a continuous probability distribution:

$$\begin{aligned} \mu = E[X] &= \int xp(x)dx \\ V = V[f(X)] &= E[(X-\mu)^2] = \int (x-\mu)^2 p(x)\, dx \\ &= E[X^2] - \mu^2 \end{aligned}$$

Also, $\sigma = \sqrt{V}$ is called the standard deviation. The standard deviation is a real number greater than or equal to zero.

Basic properties of expected value and variance

For two random variables X, Y and real numbers a, b, it holds that

$$\begin{aligned} E[X+Y] &= E[X]+E[Y] \\ E[aX+b] &= aE[X]+b \\ V[aX+b] &= a^2V[X] \end{aligned}$$

Application:

We can normalize a random variable X with expectation μ and variance $v = \sigma^2$ to have expectation 0 and variance 1 by transforming

$$Y = \frac{X-\mu}{\sqrt{v}} = \frac{X-\mu}{\sigma}$$

(This is often used, so keep it in mind.)

Q. Show the equation for the above application.

Bernoulli distribution

A discrete probability distribution that takes 1 with probability p and 0 with probability $q = 1 - p$ is called a Bernoulli distribution.

$$\text{Bernoulli distribution: } P(X = 1) = p, \qquad P(X = 0) = q = 1 - p$$

The expected value of the Bernoulli distribution is

$$E[X] = 1 \cdot P(X = 1) + 0 \cdot P(X = 0) = p$$

and the variance is

$$\begin{aligned} V[X] &= (1 - p)^2 \cdot P(X = 1) + (0 - p)^2 \cdot P(X = 0) \\ &= (1 - p)^2 p + p^2 (1 - p) \\ &= pq \end{aligned}$$

- In machine learning, the Bernoulli distribution appears in binary classification problems.

Categorical distribution

For elementary events $A_1, A_2, \cdots, A_K$, let A_k occur with probability p_k, where $p_1 + \cdots + p_K = 1$. Let us define a vector consisting of k random variables

$$X = (X_1, X_2, \cdots, X_k)$$

as $X_k = 1$ when A_k occurs and $X_k = 0$ when it does not. Then it holds that

$$P(X = (0, \cdots, 0, 1, 0, \cdots, 0)) = p_k \qquad \text{(in l.h.s., only the } k\text{-the element is 1)},$$

while it holds that

$$P(X_k = 1) = p_k, \qquad P(X_k = 0) = 1 - p_k$$

for each emement. This is called a categorical distribution (or Martinoulli distribution).

The categorical distribution is represented by K parameters $(p_1, p_2, \cdots, p_K)$, where p_k is the probability of event A_k occurring. Note that in a categorical distribution the value of X_k affects the value of X_l, so X_k and X_l are not independent.

- In machine learning classification problems, a one-hot representation is a vector of K types of labels as shown above, and model prediction is done by estimating the categorical distribution.

Normal distribution (Gaussian dustribution)

When the probability density function is given by

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

the continuous probability distribution X is called the normal distribution (or Gaussian distribution). In this case, it holds that

$$\text{Expected value:} E[X] = \mu \qquad \text{Variance: } V[X] = \sigma^2$$

The normal distribution is often denoted as $N(\mu, \sigma^2)$.

- There is a similarity between normal distributions. When X follows the standard normal distribution $N(0,1)$, the distribution of $\sigma X + \mu$ is $N(\mu, \sigma^2)$, which is used for the initial values of the weight parameters in neural networks.

3.2 Basics of information theory

Information content

The information content (als called Shannon information) when an event A occurs in a trial is defined as

$$I(A) = -\log P(A)$$

using the probability $P(A)$ of the event occurring. The base of the logarithmic function is usually 2 or the base of the natural logarithm e. Since $0 \leq P(A) \leq 1$, the information content is greater than or equal to zero (or infinity).

For example, if event A occurs with probability 2^{-k}, the information content of event A when the base is 2 is

$$-\log_2 P(A) = -\log_2 2^{-k} = k.$$

The reciprocal of the probability is expressed in binary as

$$2^k \text{ (decimal)} = 10^k \text{ (binary)},$$

so the information content is the number of digits -1. Thus, the information content is the length of the symbol sequence needed to represent the unlikelihood of an event.

Information entropy (average information content)

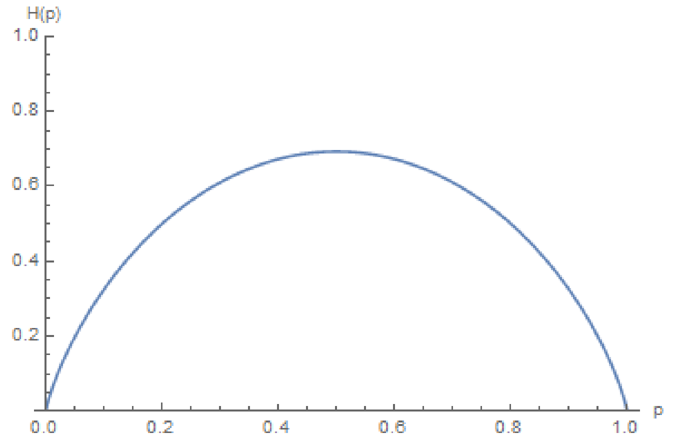
Similar to the categorical distribution, when whole events consist of n events $A_1, A_2, \cdots, A_K$ exclusive of each other, then

$$H(P) = -\sum_{k=1}^{K} P(A_k) \log P(A_k)$$

is called information entropy. Information entropy is the expected value of the information content $I(A_k)$ when it is viewed as a random variable.

For example, the information entropy of Bernoulli distribution is

$$\begin{aligned} H(P) &= -P(X=1)\log P(X=1) - P(X=0)\log P(X=0) \\ &= -p\log p - (1-p)\log(1-p). \end{aligned}$$



Information entropy of Bernoulli distribution

horizontal axis p, vertical axis $H(P)$

The information content is larger for events with smaller probabilities, but its expected value, the information entropy, is larger when the probabilities are equal, i.e. when it is more difficult to predict.

Cross entropy

Suppose that whole events consist of K events $A_1, A_2, \cdots, A_K$ that are exclusive of each other, and two probability distributions $P(A_k)$ and $Q(A_k)$ $(k = 1, 2, \cdots, K)$ are given. Then,

$$H(P, Q) = -\sum_{k=1}^{K} P(A_k) \log Q(A_k)$$

is called the cross-entropy (of Q with respect to P).

The cross-entropy is the expected value of the information content about the probability distribution Q with respect to the probability distribution P. Note that the cross-entropy is not symmetric for P and Q.

In general, minimizing the cross-entropy error for Q with fixed P, we can show that $H(P, Q) = H(P)$ as the minimum value when $Q = P$.

- In machine learning, cross-entropy is very familiar as an error function in the problem of classifying K types. In this case, P is the categorical distribution determined from the one-hot representation of the correct answer value, and Q is the categorical distribution of the prediction.

Meaning of information entropy

When the probability distribution P is fixed and the probability distribution Q is varied, the cross-entropy $H(P,Q)$ is minimized when $P = Q$. In other words, cross-entropy is a measure of how different the probability distributions P and Q are.

For example, the cross-entropy for two Bernoulli distributions P and Q, $P(X = 1) = p$ and $Q(X = 1) = q$, is given by

$$\begin{aligned} H(P,Q) &= -P(X=1)\log Q(X=1) - P(X=0)\log Q(X=0) \\ &= -p\log q - (1-p)\log(1-q). \end{aligned}$$

Assuming $p = \frac{1}{2}$ for simplicity, it becomes

$$H(P,Q) = -\frac{1}{2}\log q(1-q) = -\frac{1}{2}\log\left(-\left(q-\frac{1}{2}\right)^2+\frac{1}{4}\right).$$

Thus, when $q = \frac{1}{2}$, the cross-entropy takes a minimum value of $\frac{1}{2}\log 4$ and increases as q moves away from it.