慶應義塾大学理工学部

機械学習基礎

第３回　基礎的概念

情報工学科　教授　杉浦孔明
komei.sugiura@keio.jp

連絡

次回はオンデマンド（K-LMSからアクセスしてください）

本講義の到達目標と今回の授業の狙い

本講義の到達目標

- DNNの基礎理論と実装の関係を理解する
- 種々のDNNをコーディングできる

今回の授業の狙い

- 関連分野の歴史
- 具体例を用いた基礎的概念の習得

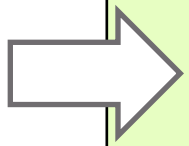
- 出席確認： K-LMS上の機械学習基礎のMainページへアクセス

機械学習関連分野の歴史

本講義の資料は電気情報工学科　村田先生と共同作成しています

分野の関係（再）

- 線形代数
- 微分積分学
- 確率論
- その他の理論



簡単な歴史①

1800年頃	ガウスやルジャンドルらが最小二乗法を発明 「なぜ４乗や６乗でなく２乗なのか」をガウス分布で説明
1958年	**パーセプトロン発表** [Rosenblatt 1958]
	第１次AIブーム

簡単な歴史②

1800年頃	ガウスやルジャンドルらが最小二乗法を発明 「なぜ４乗や６乗でなく２乗なのか」をガウス分布で説明
1958年	**パーセプトロン発表** [Rosenblatt 1958]
	第１次AIブーム
1960年代後半	☃**冬の時代（パーセプトロンで解けない問題が示された）** 人工ニューロン（神経細胞のモデル）
1986年	**誤差逆伝播法** [Rumelhart+ 1986] ※実際には60年代から存在
1989年頃	**普遍近似定理** [Cybenko 1989] 十分なノード数があれば３層のニューラルネットで任意の連続関数を近似可能
1990年代後半	☃**冬の時代（応用へのつながりが弱かった）**

ニューラルネットの低迷

2003年に発表された論文 [Simard+ ICDAR2003] のイントロ

「1990年代前半まではニューラルネットは広く研究されていたが、最近5年間、見向きされなくなった。 2000年には、Neural Information Processing Systems（NIPS）のオーガナイザが、論文タイトルに**「ニューラルネットワーク」という語を含む場合、採択率と負の相関**があることを指摘した。」

Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis

Patrice Y. Simard, Dave Steinkraus, John C. Platt
Microsoft Research, One Microsoft Way, Redmond WA 98052
{patrice,v-davste,jplatt}@microsoft.com

Abstract

Neural networks are a powerful technology for classification of visual inputs arising from documents. However, there is a confusing plethora of different neural network methods that are used in the literature and in industry. This paper describes a set of concrete best practices that document analysis researchers can use to get good results with neural networks. The most important practice is getting a training set as large as possible: we expand the training set by adding a new form of distorted data. The next most important practice is that convolutional neural networks are better suited for visual document tasks than fully connected networks. We propose that a simple "do-it-yourself" implementation of convolution with a flexible architecture is suitable for many visual document problems. This simple convolutional neural network does not require complex methods, such as momentum, weight decay, structure-dependent learning rates, averaging layers, tangent prop, or even finely-tuning the architecture. The end result is a very simple yet general architecture which can yield state-of-the-art performance for document analysis. We illustrate our claims on the MNIST set of English digit images.

1. Introduction

After being extremely popular in the early 1990s, neural networks have fallen out of favor in research in the last 5 years. In 2000, it was even pointed out by the organizers of the Neural Information Processing System (NIPS) conference that the term "neural networks" in the submission title was negatively correlated with acceptance. In contrast, positive correlations were made with support vector machines (SVMs), Bayesian networks, and variational methods.

In this paper, we show that neural networks achieve the best performance on a handwriting recognition task (MNIST). MNIST [7] is a benchmark dataset of images of segmented handwritten digits, each with 28x28 pixels. There are 60,000 training examples and 10,000 testing examples.

Our best performance on MNIST with neural networks is in agreement with other researchers, who have found that neural networks continue to yield state-of-the-art performance on visual document analysis tasks [1][2].

The optimal performance on MNIST was achieved using two essential practices. First, we created a new, general set of elastic distortions that vastly expanded the size of the training set. Second, we used convolutional neural networks. The elastic distortions are described in detail in Section 2. Sections 3 and 4 then describe a generic convolutional neural network architecture that is simple to implement.

We believe that these two practices are applicable beyond MNIST, to general visual tasks in document analysis. Applications range from FAX recognition, to analysis of scanned documents and cursive recognition in the upcoming Tablet PC.

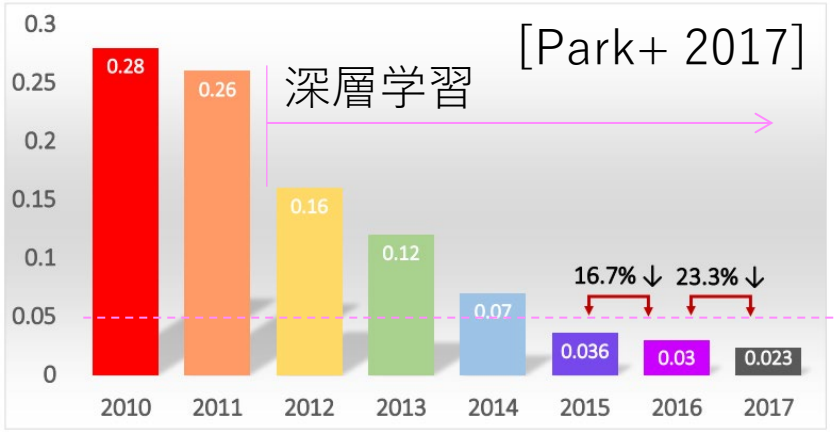
2. Expanding Data Sets through Elastic Distortions

Synthesizing plausible transformations of data is simple, but the "inverse" problem – transformation invariance - can be arbitrarily complicated. Fortunately, learning algorithms are very good at learning inverse problems. Given a classification task, one may apply transformations to generate additional data and let the learning algorithm infer the transformation invariance. This invariance is embedded in the parameters, so it is in some sense free, since the computation at recognition time is unchanged. If the data is scarce and if the distribution to be learned has transformation-invariance properties, generating additional data using transformations may even improve performance [6]. In the case of handwriting recognition, we postulate that the distribution has some invariance with respect to not only affine transformations, but also elastic deformations corresponding to uncontrolled oscillations of the hand muscles, dampened by inertia.

Simple distortions such as translations, rotations, and skewing can be generated by applying affine displacement fields to images. This is done by computing for every pixel a new target location with respect to the original location. The new target location, at position (x,y) is given with respect to the previous position. For instance if $\Delta x(x,y)=1$, and $\Delta y(x,y)=0$, this means that the new location of every pixel is shifted by 1 to the right. If

簡単な歴史③

2006年	Hintonらが**深層学習**を提唱 [Hinton+ 2006]
	学術会議等で徐々に浸透するものの、広まりは緩やか 「３層ニューラルネットで十分では？」と思われていた
2012年	深層ニューラルネットが画像認識コンペILSVRCにおいて優勝し注目される（**ImageNet**データセット[Deng+ 2009]）
	第３次AIブーム 一般応用が爆発的に増加



人間による誤り率
=0.051

ImageNet の画像例

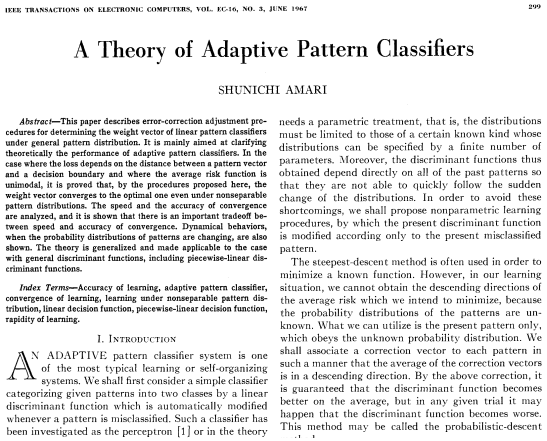
簡単な歴史③

2006年	Hintonらが**深層学習**を提唱 [Hinton+ 2006]
	学術会議等で徐々に浸透するものの、広まりは緩やか 「３層ニューラルネットで十分では？」と思われていた
2012年	深層ニューラルネットが画像認識コンペILSVRCにおいて優勝し注目される（**ImageNet**データセット[Deng+ 2009]）
	第３次AIブーム 一般応用が爆発的に増加
2022年	Text-to-imageや大規模言語モデル等の生成AIが一般に普及
2024年	Hopfield, Hintonがノーベル物理学賞受賞 Hassabisらがノーベル化学賞受賞（蛋白質構造解析へのAI応用）

甘利俊一先生と福島邦彦先生

甘利俊一先生

- 多層パーセプトロンの確率的勾配降下法 [Amari 1967]
  - Rumelhartら [Rumelhart+ 1986] が誤差逆伝播法（backpropagation）として再発見



IEEE TRANSACTIONS ON ELECTRONIC COMPUTERS, VOL. EC-16, NO. 3, JUNE 1967 299

A Theory of Adaptive Pattern Classifiers

SHUNICHI AMARI

Abstract—This paper describes error-correction adjustment procedures for determining the weight vector of linear pattern classifiers under general pattern distribution. It is mainly aimed at clarifying theoretically the performance of adaptive pattern classifiers. In the case where the loss depends on the distance between a pattern vector and a decision boundary and where the average risk function is unimodal, it is proved that, by the procedures proposed here, the weight vector converges to the optimal one even under nonseparable pattern distributions. The speed and the accuracy of convergence are analyzed, and it is shown that there is an important tradeoff between speed and accuracy of convergence. Dynamical behaviors, when the probability distributions of patterns are changing, are also shown. The theory is generalized and made applicable to the case with general discriminant functions, including piecewise-linear discriminant functions.

Index Terms—Accuracy of learning, adaptive pattern classifier, convergence of learning, learning under nonseparable pattern distribution, linear decision function, piecewise-linear decision function, rapidity of learning.

I. Introduction

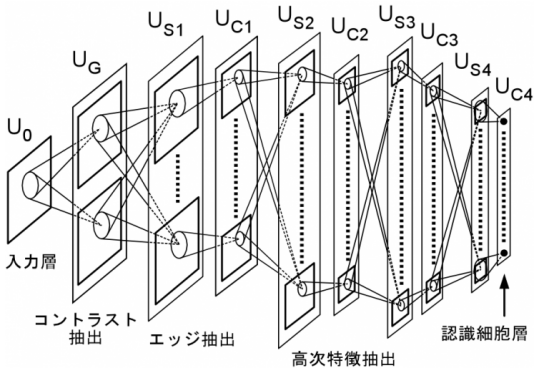
An adaptive pattern classifier system is one of the most typical learning or self-organizing systems. We shall first consider a simple classifier categorizing given patterns into two classes by a linear discriminant function which is automatically modified whenever a pattern is misclassified. Such a classifier has been investigated as the perceptron [1] or in the theory needs a parametric treatment, that is, the distributions must be limited to those of a certain known kind whose distributions can be specified by a finite number of parameters. Moreover, the discriminant functions thus obtained depend directly on all of the past patterns so that they are not able to quickly follow the sudden change of the distributions. In order to avoid these shortcomings, we shall propose nonparametric learning procedures, by which the present discriminant function is modified according only to the present misclassified pattern.

The steepest-descent method is often used in order to minimize a known function. However, in our learning situation, we cannot obtain the descending directions of the average risk which we intend to minimize, because the probability distributions of the patterns are unknown. What we can utilize is the present pattern only, which obeys the unknown probability distribution. We shall associate a correction vector to each pattern in such a manner that the average of the correction vectors is in a descending direction. By the above correction, it is guaranteed that the discriminant function becomes better on the average, but in any given trial it may happen that the discriminant function becomes worse. This method may be called the probabilistic-descent

福島邦彦先生

- ネオコグニトロン [福島 1979]
- 2024年ノーベル物理学賞の背景説明「畳み込みニューラルネットワーク構造はネオコグニトロンに起源を持つ」

https://www.nobelprize.org/prizes/physics/2024/advanced-information/



https://dbnst.nii.ac.jp/upload/images/research/498/pro/e71b520310ea4230ee0028e3058e9a5d/fig0825std.png

チューリング賞（2018）を受賞した深層学習研究におけるキーパーソン３名

Geoffrey Hinton @ Toronto Univ.

ノーベル物理学賞 (2024)

誤差逆伝播法，ボルツマンマシン，
指導学生であるAlex Krizhevsky,
Ilya Sutskever (OpenAI創業者)と
AlexNetを提案

Yoshua Bengio @ Univ. of Montreal

系列データの確率モデリング，
Neural Machine Translation,
敵対的生成ネットワーク 等

Yann LeCun @ New York Univ. & Meta

CNN，誤差逆伝播法の改善，
NN研究の様々な基礎概念の提案 等

ノーベル物理学賞・ノーベル化学賞（2024）

ノーベル物理学賞にAIの中核「機械学習」の基礎に関わった2人

2024年10月8日 23:27更新

https://www3.nhk.or.jp/news/special/nobelprize/2024/physics/article_01.html

2024年のノーベル化学賞に、AIでたんぱく質の構造予測に成功した研究者ら3人

2024年10月9日 20:42更新

https://www3.nhk.or.jp/news/special/nobelprize/2024/chemistry/article_01.html

機械学習の基礎

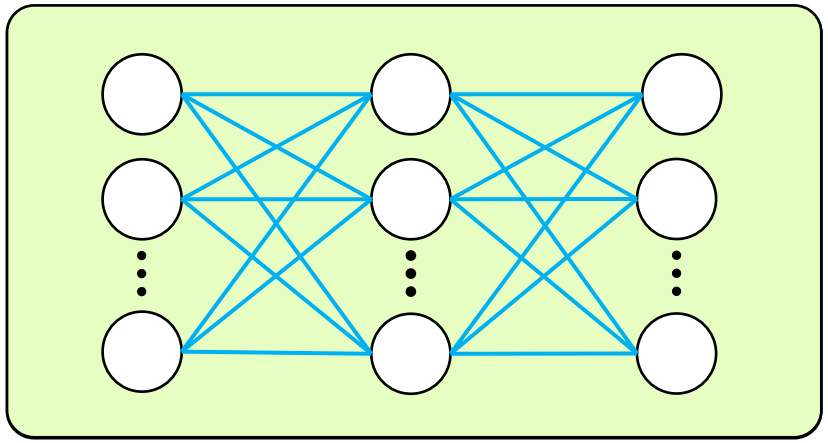
機械学習とは

ニューラルネットワーク等の数理モデルを用いてデータに潜むパターンに基づき予測/分類を行う技術

訓練データ
（既知）

学習

学習済モデル

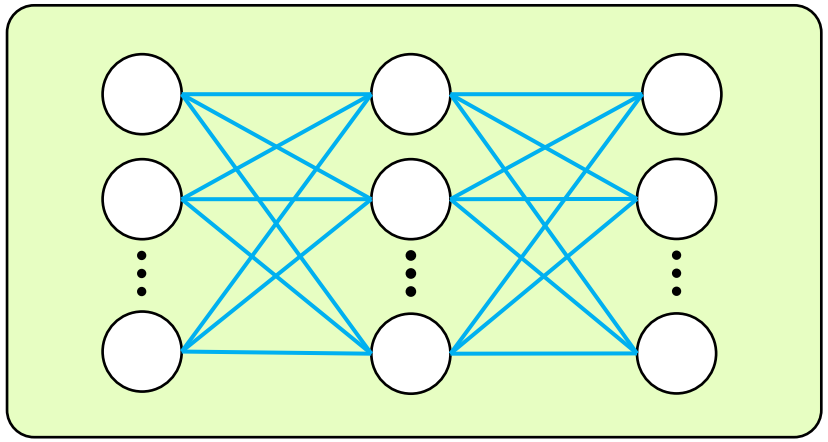


機械学習とは

ニューラルネットワーク等の数理モデルを用いてデータに潜むパターンに基づき予測/分類を行う技術

テストデータ
（未知）

学習済モデル

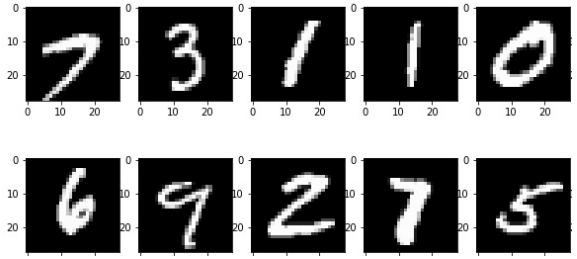


分類

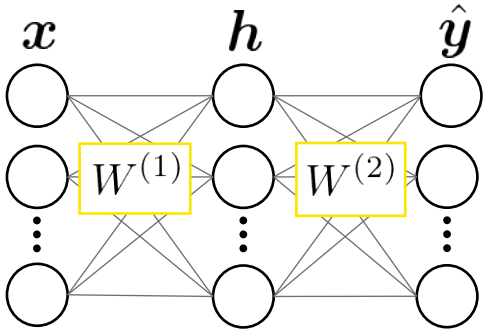
「犬」

機械学習の主要要素：
データ・モデル・目的関数を定めたうえでの最適化問題

学習に使用される**データ**



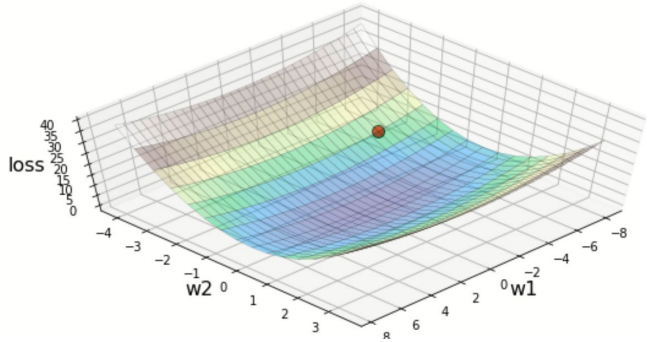
ニューラルネット等の**モデル**



モデルの良さを定量化する**目的関数**

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk} \log P(\hat{y}_{nk})$$

目的関数を最大化/最小化するために、モデルのパラメータを調整する
最適化



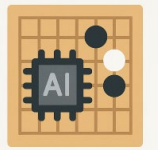
機械学習のサブ分野

教師あり学習（supervised learning）

- **入力ベクトル** x **と目標ベクトル** y のペア (x, y) を用いて学習

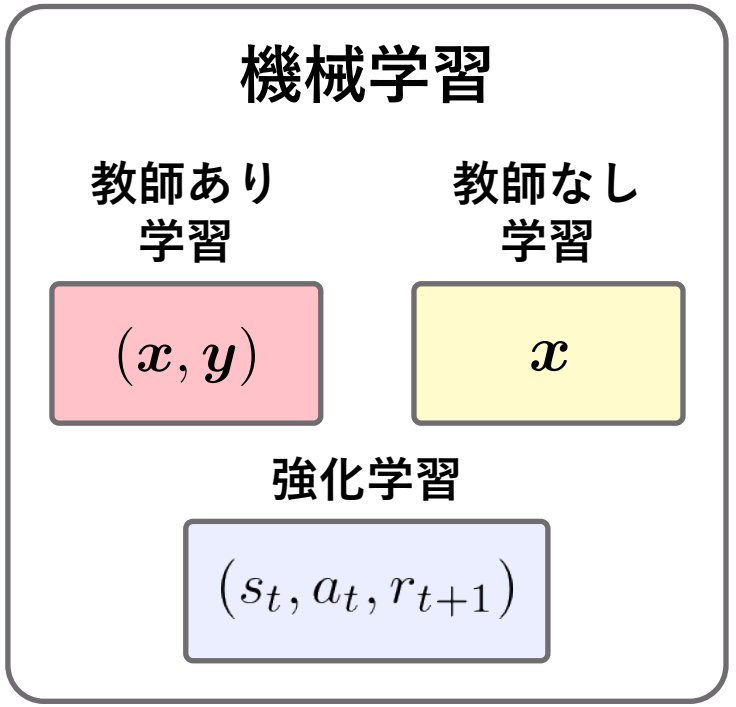
教師なし学習（unsupervised learning）

- 入力ベクトル x を用いて学習



強化学習（reinforcement learning）

- 状態 s_t において、エージェントの行動 a_t によって得られる報酬 r_{t+1} を用いて学習



※この分け方は分野全体をカバーしたものではなく、複合した問題設定もある

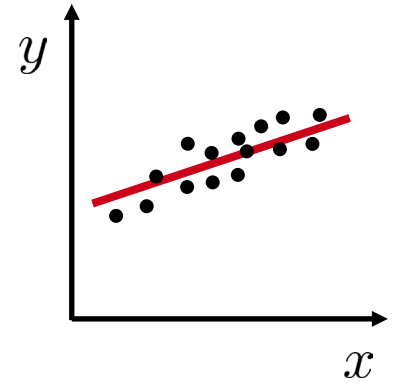
教師あり学習の代表的タスク：回帰・分類

回帰（regression）

- y が**量的変数**の場合　　数や量の大小を表す

  例：気温から来場者数を予測

  - **線形回帰**（右図）/ ニューラルネットワーク 等

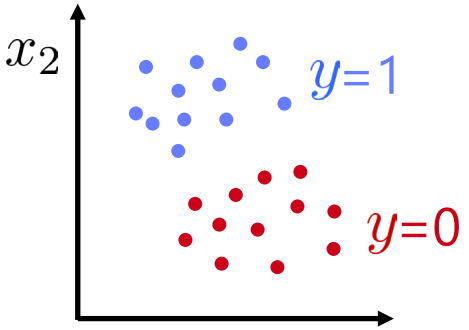


分類（classification）

- y が**質的変数**の場合　　種類や有無を表す

  例：与えられた画像が猫・犬・鳥のどれかを判断する

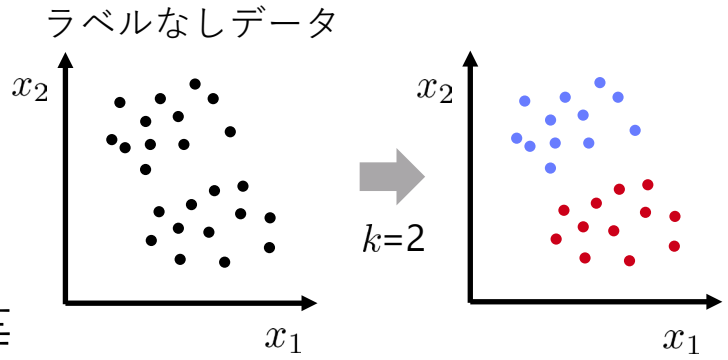
  - **ロジスティック回帰**（右図）/ **ソフトマックス回帰** / ニューラルネットワーク 等
    - 分類にも関わらず「～回帰」と呼ばれることに注意！



教師なし学習の代表的タスク：クラスタリング・次元削減

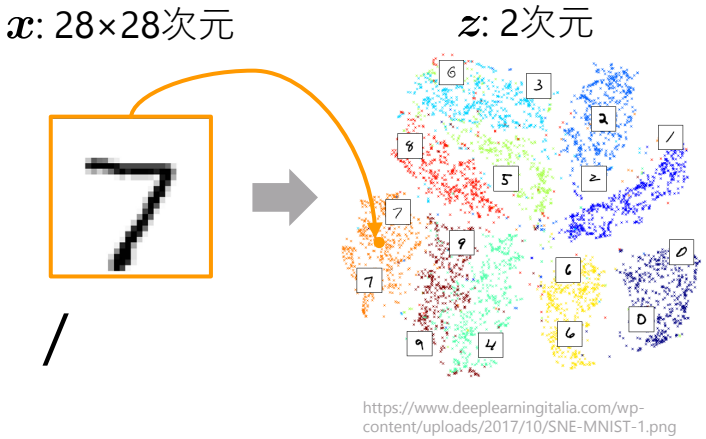
クラスタリング（clustering）

- 入力 $\boldsymbol{x}$ をk個の集合に分ける
  - k平均法（k-means clustering）/ 階層クラスタリング（hierarchical clustering）等



次元削減（dimensionality reduction）

- 高次元の入力 $\boldsymbol{x}$ を低次元のデータ $\boldsymbol{z}$ に次元削減
  - 主成分分析（principal component analysis; PCA）/ t-SNE / 自己符号化器（autoencoder）等



https://www.deeplearningitalia.com/wp-content/uploads/2017/10/SNE-MNIST-1.png

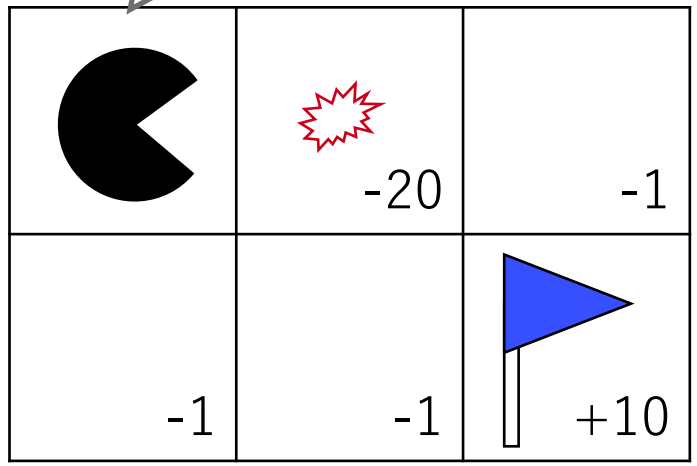
ニューラルネットワークの学習後の内部表現解析等でよく用いられる

強化学習

- 状態 s_t において、エージェントの行動 a_t によって得られる報酬r_{t+1}を用いて学習
  - 応用例：ボードゲーム、ロボットの単純な行動
  - Q学習 / REINFORCE / Deep Q-network等

- 動的計画法、最適制御理論、能動学習との関係が深い

状態 s_t ：左上マス
行動 a_t ：右へ進む
報酬 r_{t+1} ：-20



例題による用語説明

例題：教室のCO_2濃度を予測したい
線形回帰

風速とCO_2濃度の記録

ID	風速 x_n	濃度 y_n
1	2.0	685
2	1.2	790
3	1.8	733
…	…	…
1000	1.6	743
1001	1.8	740
1002	2.6	677

データ

※数値は適当です

- 「風が強ければ濃度が下がる」と考えたので線形**モデル**を使う
  （↑強引）

$$\hat{y}_n = wx_n + b \triangleq f(x_n)$$

パラメータ

- $\hat{y}_n$：**予測値**（prediction）
- y_n：**真値**（ground truth）←目標変数と同義

例題：教室のCO_2濃度を予測したい
最小二乗法

CO_2濃度を予測したい

↓

$\hat{y}_n$がy_nの良い近似になるように、パラメータを設定したい

$$|y_n - \hat{y}_n| \to \min$$

絶対値でも良いが**最小二乗法**（least squares method）を習った

$$\frac{1}{2}\sum_n (y_n - \hat{y}_n)^2 \to \min$$

- 「風が強ければ濃度が下がる」と考えたので線形**モデル**を使う
  （↑強引）

$$\hat{y}_n = wx_n + b \triangleq f(x_n)$$

パラメータ

- $\hat{y}_n$：**予測値**（prediction）
- y_n：**真値**（ground truth）←目標変数と同義

例題：教室のCO_2濃度を予測したい
訓練集合とテスト集合

CO_2濃度を予測したい

↓

$\hat{y}_n$がy_nの良い近似になるように、
パラメータを設定したい

$$|y_n - \hat{y}_n| \to \min$$

絶対値でも良いが**最小二乗法**（least squares method）を習った

$$\frac{1}{2}\sum_n (y_n - \hat{y}_n)^2 \to \min$$

新規サンプルについて予測したい

ID	風速 x_n	濃度 y_n
1	2.0	685
2	1.2	790
3	1.8	733
…	…	…
1000	1.6	743
1001	1.8	740
1002	2.6	677

パラメータ推定に使うサンプル集合

新規サンプルとみなす（＝濃度は知らなかったものとする）

例題：教室のCO_2濃度を予測したい
学習とは

- **訓練集合**（training set）
  $X_{\mathrm{train}} = \{(x_n, y_n) | n = 1, \ldots, 1000\}$
  1個分を訓練サンプルと呼ぶ
- X_{train}から写像$f(x)$を求める
  ＝**訓練**（training）または**学習**（learning）　f(x)=wx+b
- **テスト集合**（test set）
  $X_{\mathrm{test}} = \{(x_n, y_n) | n = 1001, 1002\}$
  を用いて真値と予測値の誤差を評価

新規サンプルについて予測したい

ID	風速 x_n	濃度 y_n
1	2.0	685
2	1.2	790
3	1.8	733
…	…	…
1000	1.6	743
1001	1.8	740
1002	2.6	677

パラメータ推定に使うサンプル集合

新規サンプルとみなす（＝濃度は知らなかったものとする）

例題：教室のCO_2濃度を予測したい
損失関数

- **訓練集合**（training set）
  $X_{\text{train}} = \{(x_n, y_n) | n = 1, \ldots, 1000\}$
  1個分を訓練サンプルと呼ぶ
- X_{train}から写像$f(x)$を求める
  ＝**訓練**（training）または**学習**（learning）
- **テスト集合**（test set）
  $X_{\text{test}} = \{(x_n, y_n) | n = 1001, 1002\}$
  を用いて真値と予測値の誤差を評価

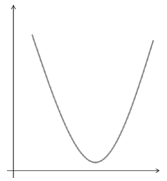
$$E(w,b) = \frac{1}{2}\sum_{n=1}^{1000}(y_n - \hat{y}_n)^2$$
$$= \frac{1}{2}\sum_{n=1}^{1000}(y_n - wx_n - b)^2$$

損失関数（loss function） or
誤差関数（error function） or
コスト関数（cost function） or
目的関数（objective function）
とも呼ばれる

損失関数が小さくなるようにf(x)=y_nを求めたい

例題：教室のCO_2濃度を予測したい
損失関数を最小化するパラメータとは

w, b に関する 2 次式なので最小値のときに偏微分が 0 になる



$$E(w,b) = \frac{1}{2}\sum_{n=1}^{1000}(y_n - \hat{y}_n)^2$$

$$= \frac{1}{2}\sum_{n=1}^{1000}(y_n - wx_n - b)^2$$

$$\frac{\partial E(w,b)}{\partial w} = 0, \quad \frac{\partial E(w,b)}{\partial b} = 0$$

$$\frac{\partial E(w,b)}{\partial w} = \sum_{n=1}^{1000}(y_n - wx_n - b)(-x_n) = w\sum_{n=1}^{1000}x_n^2 + b\sum_{n=1}^{1000}x_n - \sum_{n=1}^{1000}x_n y_n$$

$$\frac{\partial E(w,b)}{\partial b} = \sum_{n=1}^{1000}(y_n - wx_n - b)(-1) = w\sum_{n=1}^{1000}x_n + b\sum_{n=1}^{1000}1 - \sum_{n=1}^{1000}y_n$$

例題：教室のCO_2濃度を予測したい
正規方程式

■ **正規方程式**（normal equation）

$$\begin{pmatrix} \sum_{n=1}^{1000} x_n^2 & \sum_{n=1}^{1000} x_n \\ \sum_{n=1}^{1000} x_n & \sum_{n=1}^{1000} 1 \end{pmatrix} \begin{pmatrix} w \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{1000} x_n y_n \\ \sum_{n=1}^{1000} y_n \end{pmatrix}$$

$$\frac{\partial E(w,b)}{\partial w} = 0, \quad \frac{\partial E(w,b)}{\partial b} = 0$$

$$\frac{\partial E(w,b)}{\partial w} = \sum_{n=1}^{1000} (y_n - wx_n - b)(-x_n) = w \sum_{n=1}^{1000} x_n^2 + b \sum_{n=1}^{1000} x_n - \sum_{n=1}^{1000} x_n y_n$$

$$\frac{\partial E(w,b)}{\partial b} = \sum_{n=1}^{1000} (y_n - wx_n - b)(-1) = w \sum_{n=1}^{1000} x_n + b \sum_{n=1}^{1000} 1 - \sum_{n=1}^{1000} y_n$$

例題：教室のCO_2濃度を予測したい
パラメータの最適化

■ **正規方程式**（normal equation）

$$\begin{pmatrix} \sum_{n=1}^{1000} x_n^2 & \sum_{n=1}^{1000} x_n \\ \sum_{n=1}^{1000} x_n & \sum_{n=1}^{1000} 1 \end{pmatrix} \begin{pmatrix} w \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{1000} x_n y_n \\ \sum_{n=1}^{1000} y_n \end{pmatrix}$$

→最適値 (w^*, b^*) が求まるので、新しい x_n から $w^* x_n + b^*$を予測可能

最適値には ＊（スター）をつける

新規の風速情報から
CO_2濃度が予測可能になった
（実際はもっと複雑）

例題：教室のCO_2濃度を予測したい
テスト集合を用いた性能評価

- 得られた(w^*, b^*)の良さを知るには、テストセット誤差を評価

  複数の機械学習手法を性能比較できる

- **二乗平均平方根誤差**（root-mean-square error; RMSE）

$$E_{\mathrm{RMSE}} = \sqrt{\frac{1}{|X_{\mathrm{test}}|} \sum_{n=1001}^{1002} (y_n - w^* x_n - b^*)^2}$$

他の尺度：**平均二乗誤差**等

全て訓練データに使わず、テストデータをちゃんと用意する

ID	風速 x_n	濃度 y_n
1	2.0	685
2	1.2	790
3	1.8	733
…	…	…
1000	1.6	743
1001	1.8	740
1002	2.6	677

パラメータ推定に使うサンプル集合

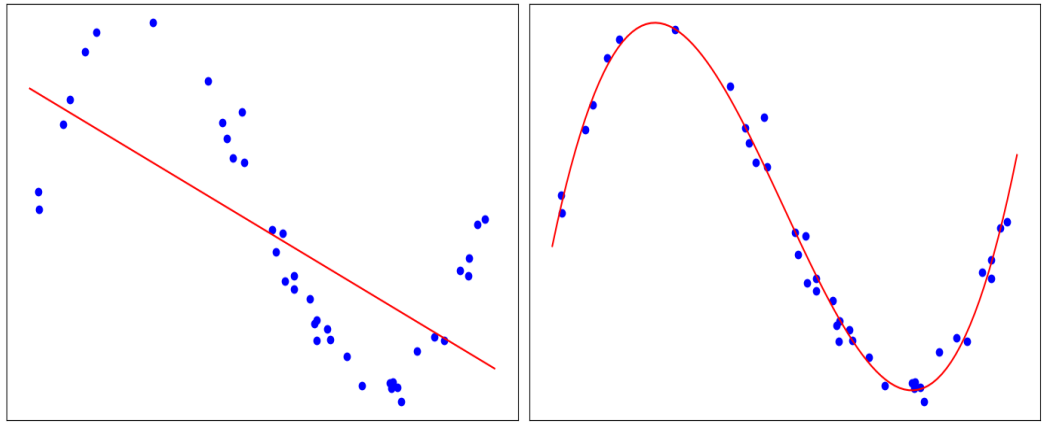
学習に使わなかったことで、性能評価に使えるようになった

単純なモデルの学習からDNNの学習へ

- ここまで紹介した話
  - 単純なモデル
  - データが少ない

→解析的に損失関数を最小化可能

例　$\hat{y} = wx + b$　　$\hat{y} = w_3x^3 + w_2x^2 + w_1x + b$



※以下の多項式曲線フィッティング問題では、損失関数を最小化する係数 $\mathbf{w}^*$ は解析的に求まる

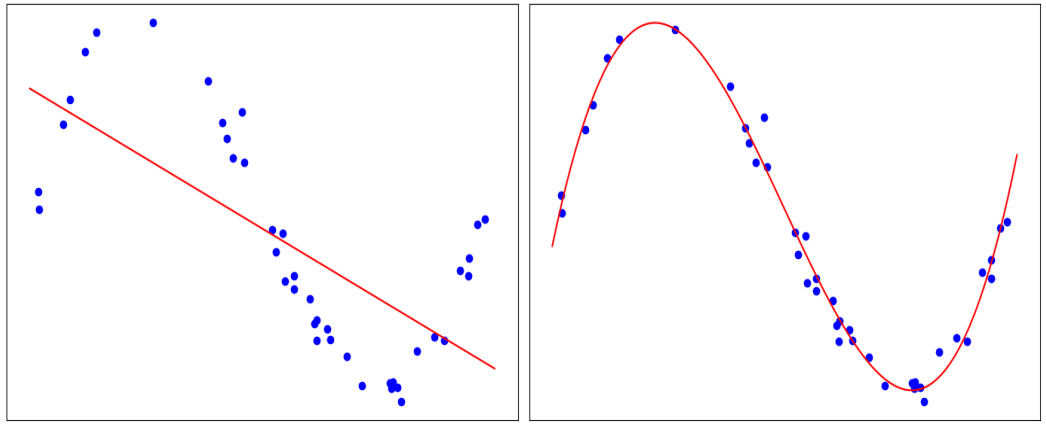
$$y(x; \mathbf{w}) = w_0 + w_1x + w_2x^2 + w_3x^3 + \cdots + w_Mx^M$$

単純なモデルの学習からDNNの学習へ

- ここまで紹介した話
  - 単純なモデル
  - データが少ない

→解析的に損失関数を最小化可能

例　$\hat{y} = wx + b$　　$\hat{y} = w_3x^3 + w_2x^2 + w_1x + b$



※以下の多項式曲線フィッティング問題では、損失関数を最小化する係数 $\mathbf{w}^*$ は解析的に求まる

$$y(x; \mathbf{w}) = w_0 + w_1x + w_2x^2 + w_3x^3 + \cdots + w_Mx^M$$

- 深層学習の実応用
  - 複雑なモデル
  - データが多い

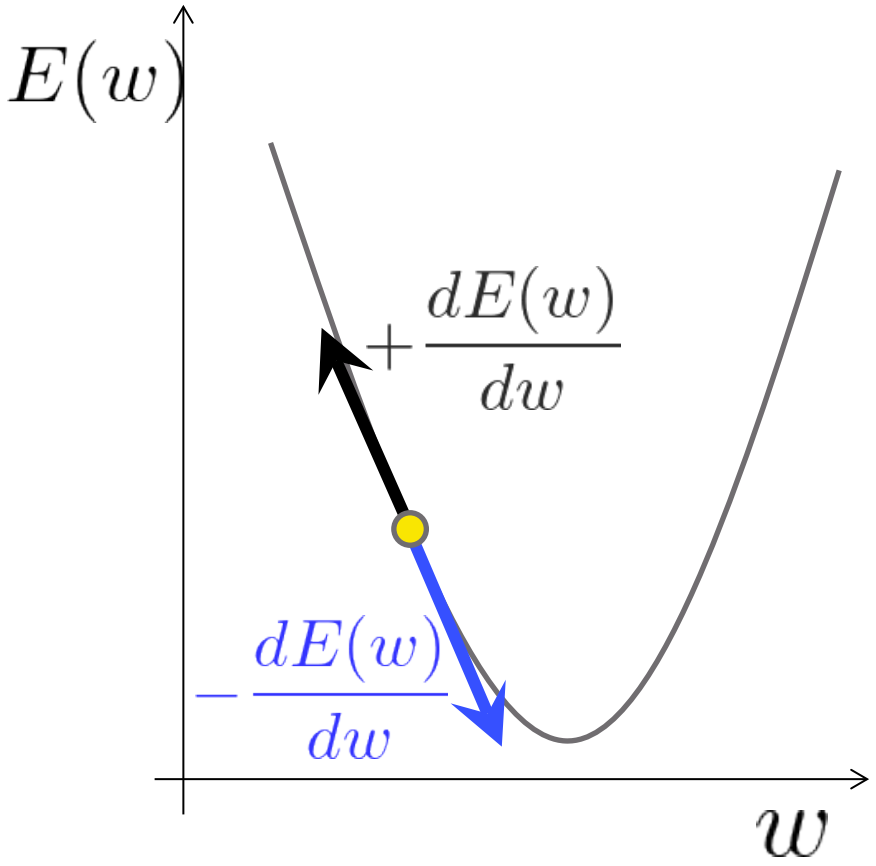
→解析的な損失関数最小化が困難

→**反復的に損失関数を最小化**

今回簡単に紹介し、今後の講義で深堀りする

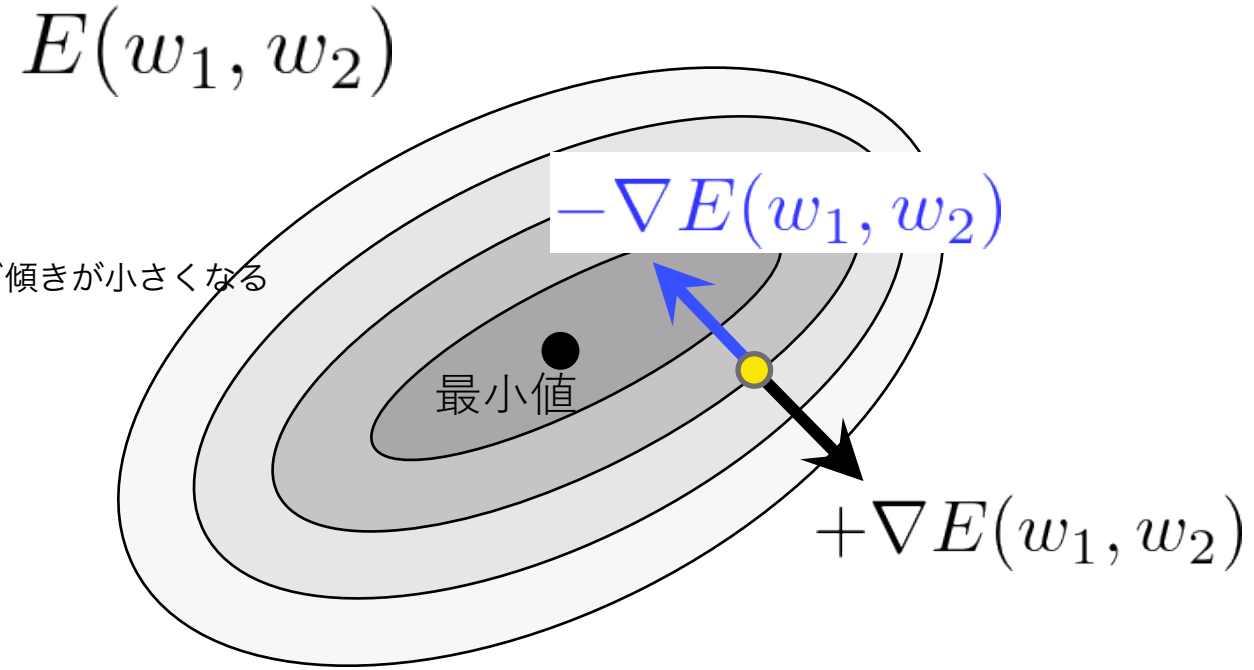
勾配降下法：誤差に応じてパラメータを更新する

w が 1 次元の場合

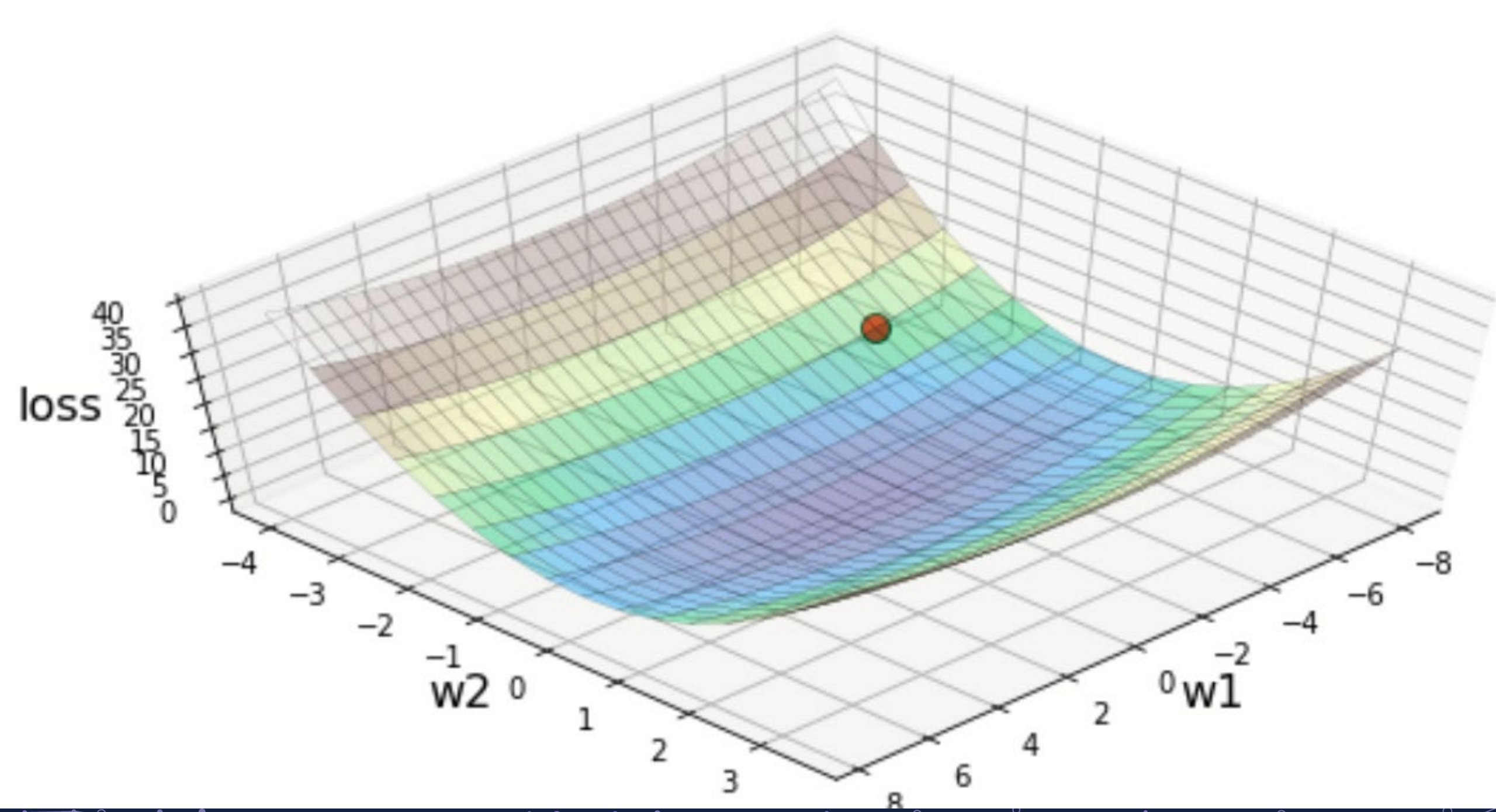


勾配が小さいほど傾きが小さくなる

w が 2 次元の場合



$$\nabla E(w_1, w_2) = \left(\frac{\partial E(w_1, w_2)}{\partial w_1}, \frac{\partial E(w_1, w_2)}{\partial w_2} \right)$$


loss
40
35
30
25
20
15
10
5
0
−4
−3
−2
−1
w2
0
1
2
3
8
6
4
2
0
w1
−2
−4
−6
−8

勾配降下法の数式表現：線形ユニットの場合

1. 初期値を設定

   例：$(w, b) \leftarrow (1, 2)$

2. 現在の(w, b)を用いて勾配の値を計算し、(w, b) を更新

$$w \leftarrow w - \eta \frac{\partial E(w,b)}{\partial w}, \qquad b \leftarrow b - \eta \frac{\partial E(w,b)}{\partial b}$$

3. 2を繰り返す

学習率：一度の更新幅を小さくする（例：0.02）

理工学基礎実験「ニューラルネットワーク」で
実際に手計算してもらいます

バッチ学習とミニバッチ学習

- **バッチ学習**（batch learning）
  - １回の更新に訓練集合全部を使用
- 訓練集合
  $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)|n = 1, ..., N\}$

- **ミニバッチ学習**（mini-batch learning）
  - １回の更新に訓練集合の一部（ミニバッチ $\mathcal{B}^{(t)}$）を使用

2.0	685
1.2	790
1.8	733
…	…
…	…
…	…
…	…
1.6	743

$\mathcal{B}^{(1)}$ → 1回目の更新

$\mathcal{B}^{(2)}$ → 2回目の更新

⋮

$\mathcal{B}^{(K)}$ → K回目の更新

ミニバッチ確率的勾配降下法（ミニバッチSGD）
minibatch stochastic gradient descent

①パラメータに初期値 $(w^{(0)}, b^{(0)})$
を設定：ランダムや固定値等

※口語では「SGD」が「ミニバッチSGD」を指すことが多いので注意。正確には、バッチサイズが１のときを「SGD」と呼ぶ。

ミニバッチ確率的勾配降下法（ミニバッチSGD）
minibatch stochastic gradient descent

①パラメータに初期値 $(w^{(0)}, b^{(0)})$ を設定：ランダムや固定値等

②ランダムに訓練集合の一部を選び、平均の損失の勾配を計算

$$\frac{1}{|\mathcal{B}^{(t)}|}\sum_{x_n\in\mathcal{B}^{(t)}}\left.\frac{\partial E(w,b)}{\partial w}\right|_{w=w^{(t)}}$$

$$\frac{1}{|\mathcal{B}^{(t)}|}\sum_{x_n\in\mathcal{B}^{(t)}}\left.\frac{\partial E(w,b)}{\partial b}\right|_{b=b^{(t)}}$$

右肩のカッコつきの数字は更新回数を表す

※口語では「SGD」が「ミニバッチSGD」を指すことが多いので注意。正確には、バッチサイズが１のときを「SGD」と呼ぶ。

ミニバッチ確率的勾配降下法（ミニバッチSGD）
minibatch stochastic gradient descent

②ランダムに訓練集合の一部を選び、平均の損失の勾配を計算

$$\left.\frac{1}{|\mathcal{B}^{(t)}|}\sum_{x_n\in\mathcal{B}^{(t)}}\frac{\partial E(w,b)}{\partial w}\right|_{w=w^{(t)}}$$

$$\left.\frac{1}{|\mathcal{B}^{(t)}|}\sum_{x_n\in\mathcal{B}^{(t)}}\frac{\partial E(w,b)}{\partial b}\right|_{b=b^{(t)}}$$

③パラメータを更新
（二乗誤差の場合）

学習率（learning rate）

$$w^{(t+1)} = w^{(t)} - \frac{\eta}{|\mathcal{B}^{(t)}|}\sum_{x_n\in\mathcal{B}^{(t)}}(y_n - w^{(t)}x_n - b^{(t)})(-x_n)$$

$$b^{(t+1)} = b^{(t)} - \frac{\eta}{|\mathcal{B}^{(t)}|}\sum_{x_n\in\mathcal{B}^{(t)}}(y_n - w^{(t)}x_n - b^{(t)})(-1)$$

右肩のカッコつきの数字は更新回数を表す

ミニバッチ確率的勾配降下法（ミニバッチSGD）
minibatch stochastic gradient descent

③パラメータを更新
（二乗誤差の場合）

学習率（learning rate）

$$w^{(t+1)} = w^{(t)} - \frac{\eta}{|\mathcal{B}^{(t)}|} \sum_{x_n \in \mathcal{B}^{(t)}} (y_n - w^{(t)} x_n - b^{(t)})(-x_n)$$

$$b^{(t+1)} = b^{(t)} - \frac{\eta}{|\mathcal{B}^{(t)}|} \sum_{x_n \in \mathcal{B}^{(t)}} (y_n - w^{(t)} x_n - b^{(t)})(-1)$$

④手順②③を繰り返す

→実習で試してみよ

本講義全体の参考図書

- ★機械学習スタートアップシリーズ これならわかる深層学習入門　瀧雅人著　講談社（本講義では、異なる表記を用いることがあるので注意）
- ★Dive into Deep Learning (https://d2l.ai/)
- 深層学習 改訂第2版 (機械学習プロフェッショナルシリーズ)　岡谷貴之著　講談社
- ディープラーニングを支える技術　岡野原大輔著　技術評論社
- 画像認識 (機械学習プロフェッショナルシリーズ)　原田達也著　講談社
- 深層学習による自然言語処理 (機械学習プロフェッショナルシリーズ)　坪井祐太、海野裕也、鈴木潤 著、講談社
- 東京大学工学教程 情報工学 機械学習　中川 裕志著、東京大学工学教程編纂委員会編　丸善出版
- パターン認識と機械学習 上・下　C.M. ビショップ著　丸善出版

参考文献

1. 金谷 健一, これなら分かる最適化数学—基礎原理から計算手法まで, 共立出版, 2005.
2. Abdulle, A., & Wanner, G. (2002). 200 years of least squares method. Elemente der Mathematik, 57(2), 45-60.
3. Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review 65.6 (1958): 386.
4. Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." nature 323.6088 (1986): 533-536.
5. Cybenko, George. "Approximation by superpositions of a sigmoidal function." Mathematics of control, signals and systems 2.4 (1989): 303-314.
6. Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.
7. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," CVPR, 2009, pp. 248–255.

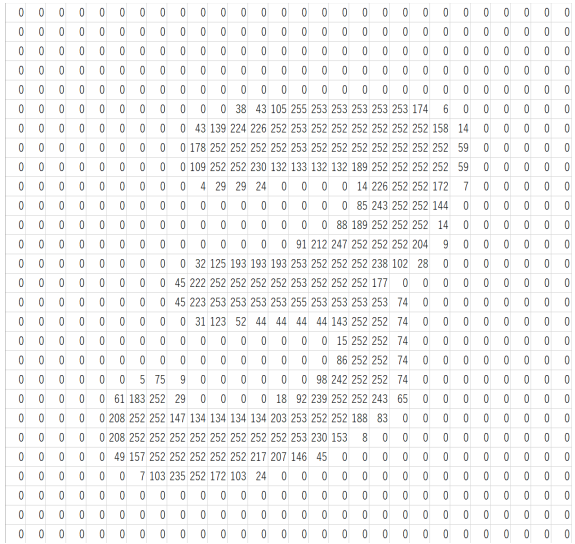
実習

【復習】テンソル：スカラー、ベクトル、行列等を一般化した概念

- 0階のテンソル（スカラー）、1階のテンソル（ベクトル）、2階のテンソル（行列）、3階のテンソル（3次元配列）、…
- プログラミングにおける多次元配列と似た概念
  - X[i, j, k]： (i, j, k)番目の要素にアクセス

- **例１：28x28のグレースケール画像**
  - shape = (28, 28)

- **例２：例１を８枚集めて作ったミニバッチ**
  - shape = (8, 28, 28)　※3階のテンソル
  - X[0, 0, 0]で最初のサンプルの(0, 0)要素にアクセス



LLMへの質問例


ML02_simple_mlp.ipynb
変更を保存できませんでした
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ
共有
コマンド
コード
テキスト
すべてのセルを実行
ドライブにコピー
RAM
ディスク
準備
1. 「ランタイム」→「ランタイムのタイプを変更」→ハードウェア アクセラレータとして「GPU」を選択し、「保存」ボタンを押す
[2]
0秒
import torch.nn.functional as F
from torchvision import datasets, transforms
import torch
import torch.nn as nn
MNISTデータセットのダウンロード
[3]
9秒
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,)
training_set = datasets.MNIST("./data", train=True, download=True, transform=transform)
test_set = datasets.MNIST("./data", train=False, transform=transform)
train_loader = torch.utils.data.DataLoader(training_set, batch_size=64, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=1000)
Gemini
上記について、変えられるとしたらどこを変える？
上記のコードブロックで変更できる箇所はいくつかあります。主に、データセットの読み込み方法や、データの前処理方法を変更することが考えられます。
具体的には以下の部分が変更可能です。
1. transform の内容:
transforms.Normalize の平均と標準偏差の値 (0.1307,), (0.3081,) は MNIST データセット全体の統計量に基づいています。もし別のデータセットを使用する場合や、独自の正規化方法を試したい場合は、これらの値を変更したり、別の正規
Change batch_size to 128
Add random rotation to transform
Show how to instantiate MyMnistNet
何を構築しますか？
Gemini は不正確な情報を表示することがあるため、内容を再確認し、コードは注意して使用してください。 詳細
変数
ターミナル
2:10
T4 (Python 3)

実習

実習の目的

- コーディングと基礎理論の関係を学ぶ

実習課題の場所

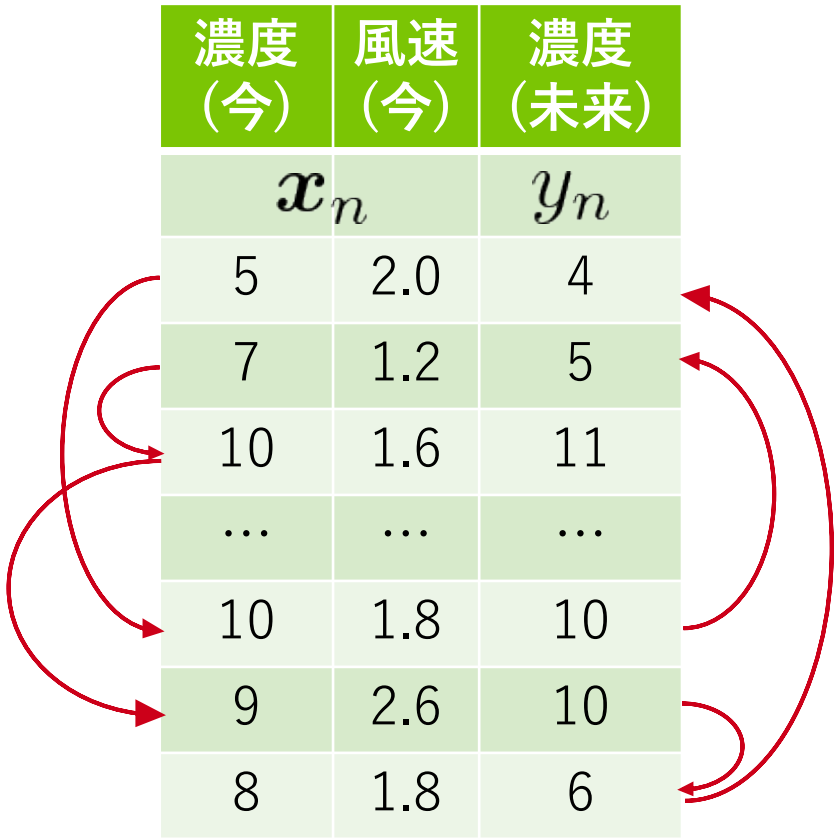
- K-LMSから辿る

実習に関する質問

- LLMに説明させる
- 教科書で調べる・検索・周囲と相談（私語禁止ではありません）
- 上記で解消しなければ挙手

付録

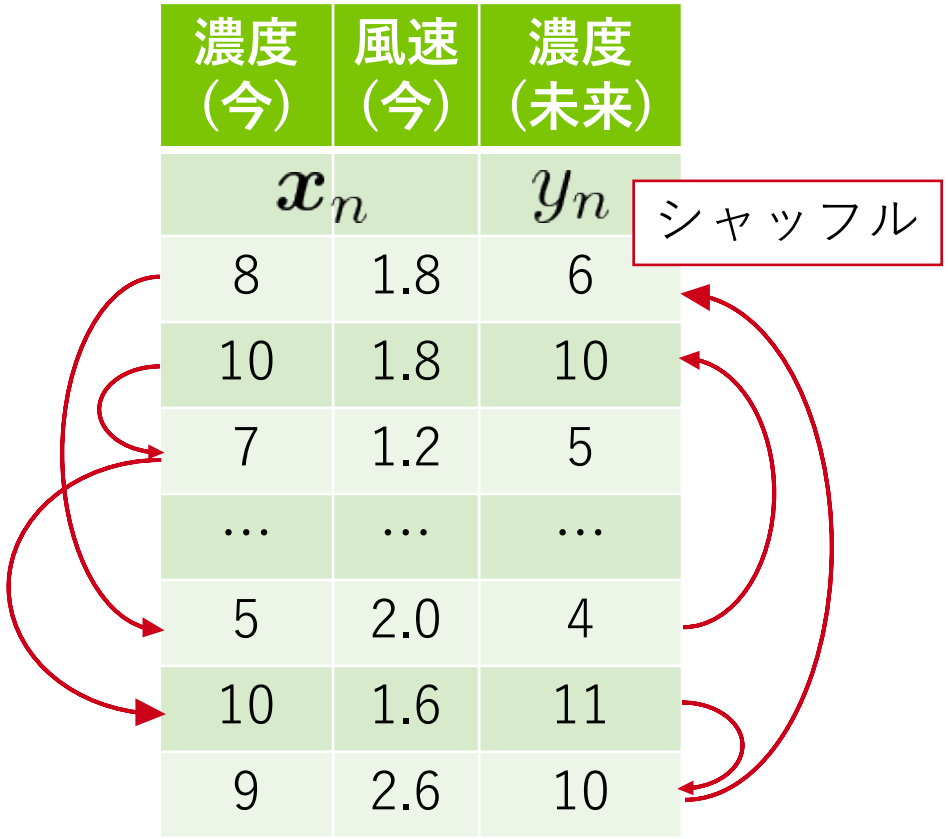
ミニバッチの作成方法の例



濃度（今）	風速（今）	濃度（未来）
$\boldsymbol{x}_n$		y_n
5	2.0	4
7	1.2	5
10	1.6	11
…	…	…
10	1.8	10
9	2.6	10
8	1.8	6

※これ以外の作成方法でも良い

ミニバッチの作成方法の例



※これ以外の作成方法でも良い

ミニバッチの作成方法の例

濃度（今）	風速（今）	濃度（未来）
$\boldsymbol{x}_n$		y_n
8	1.8	6
10	1.8	10
7	1.2	5
…	…	…
5	2.0	4
10	1.6	11
9	2.6	10

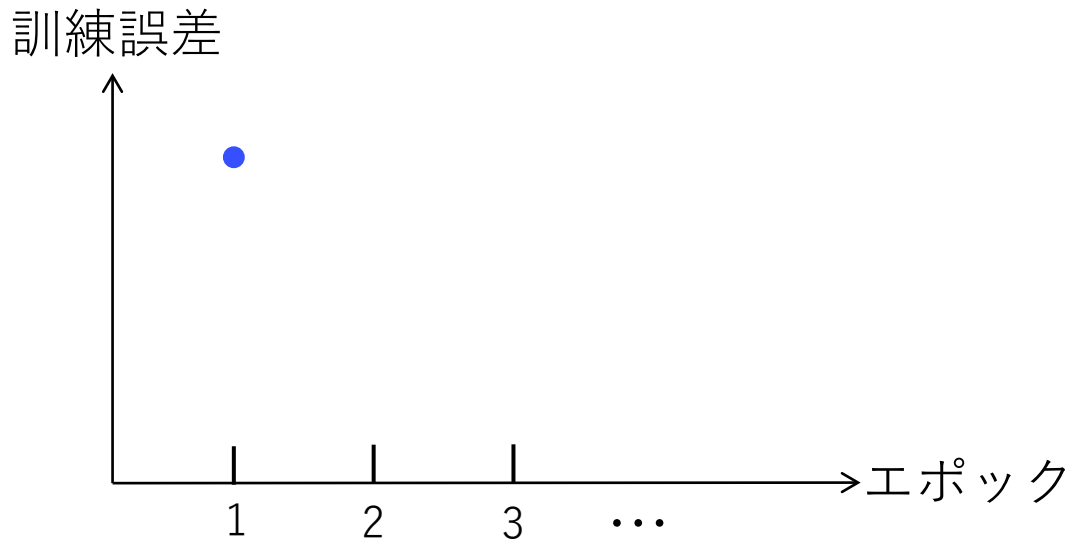
$\mathcal{B}^{(1)}$ → 1回目の更新

$\mathcal{B}^{(2)}$ → 2回目の更新

⋮

$\mathcal{B}^{(K)}$ → K回目の更新

エポック
（epoch）



エポック毎に再シャッフル

※これ以外の作成方法でも良い

ミニバッチの作成方法の例

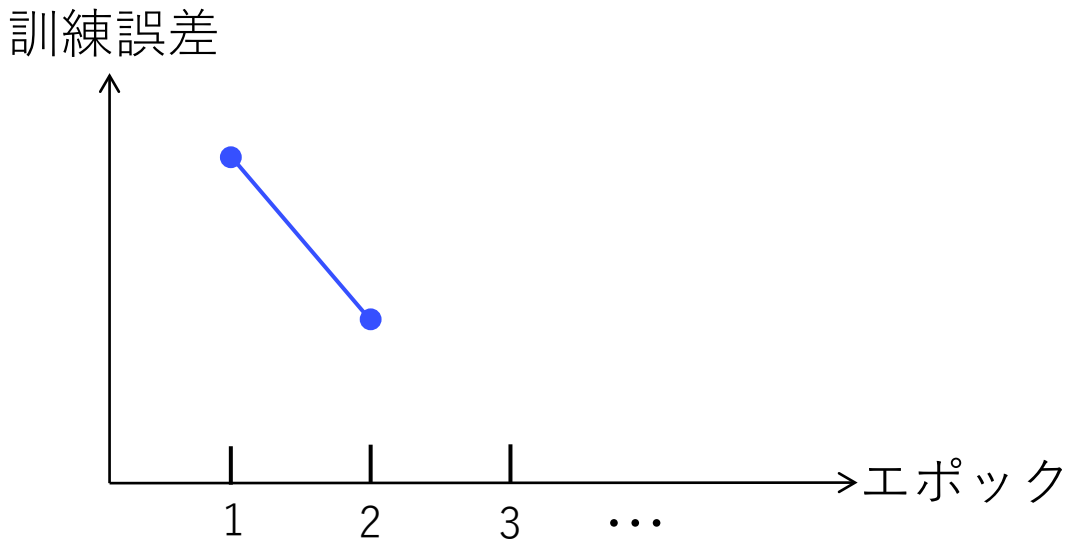
濃度（今）	風速（今）	濃度（未来）
$\boldsymbol{x}_n$		y_n
9	2.6	10
5	2.0	4
10	1.8	10
…	…	…
8	1.8	6
7	1.2	5
10	1.6	11

$\mathcal{B}^{(1)}$ → 1回目の更新

$\mathcal{B}^{(2)}$ → 2回目の更新

⋮

$\mathcal{B}^{(K)}$ → K回目の更新



エポック
（epoch）

エポック毎に再シャッフル

※これ以外の作成方法でも良い

ミニバッチの作成方法の例

濃度（今）	風速（今）	濃度（未来）
$\boldsymbol{x}_n$		y_n
10	1.6	11
5	2.0	4
8	1.8	6
…	…	…
7	1.2	5
10	1.8	10
9	2.6	10

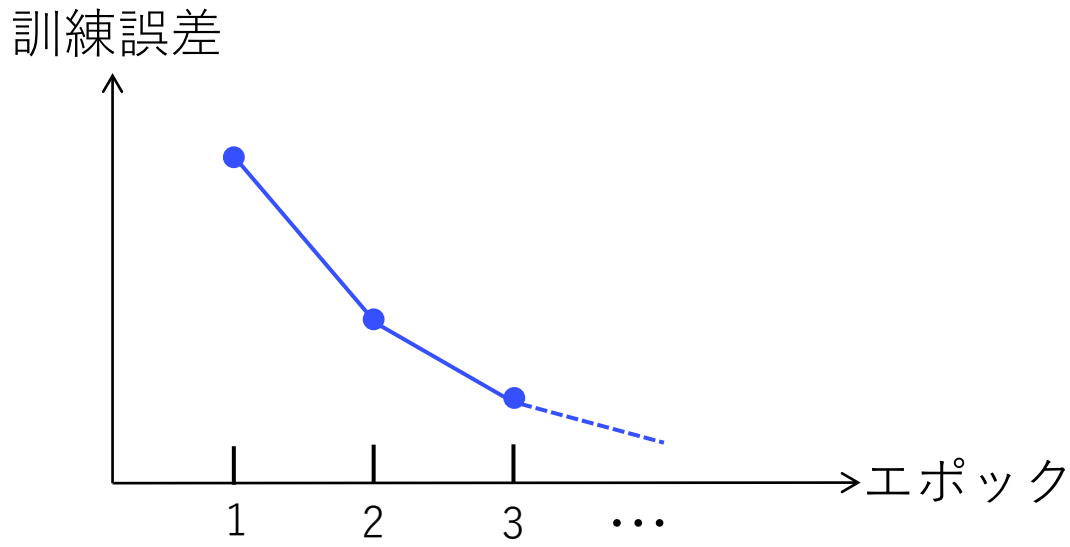
$\mathcal{B}^{(1)}$ → 1回目の更新

$\mathcal{B}^{(2)}$ → 2回目の更新

⋮

$\mathcal{B}^{(K)}$ → K回目の更新

エポック
（epoch）



エポック毎に再シャッフル

※これ以外の作成方法でも良い

Typical metrics for regression

- Mean squared error

$$\mathrm{MSE} = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2$$

- Root mean squared error

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2}$$

- Mean absolute error

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N}|y_n - \hat{y}_n|$$

- Predicted value is with hat /予測値にはハットをつける → $\hat{y}_i$
- Target is without hat / 真値にはハットをつけない → y_i