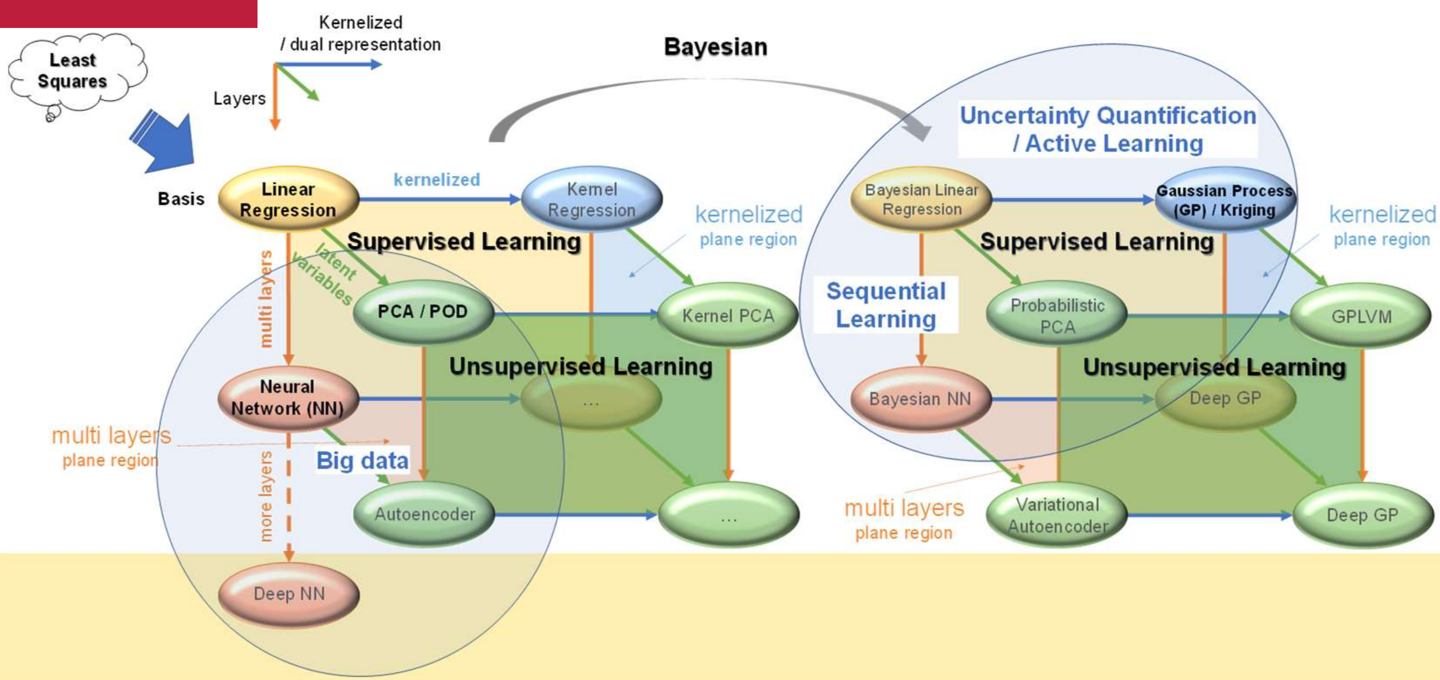


Scientific Machine Learning

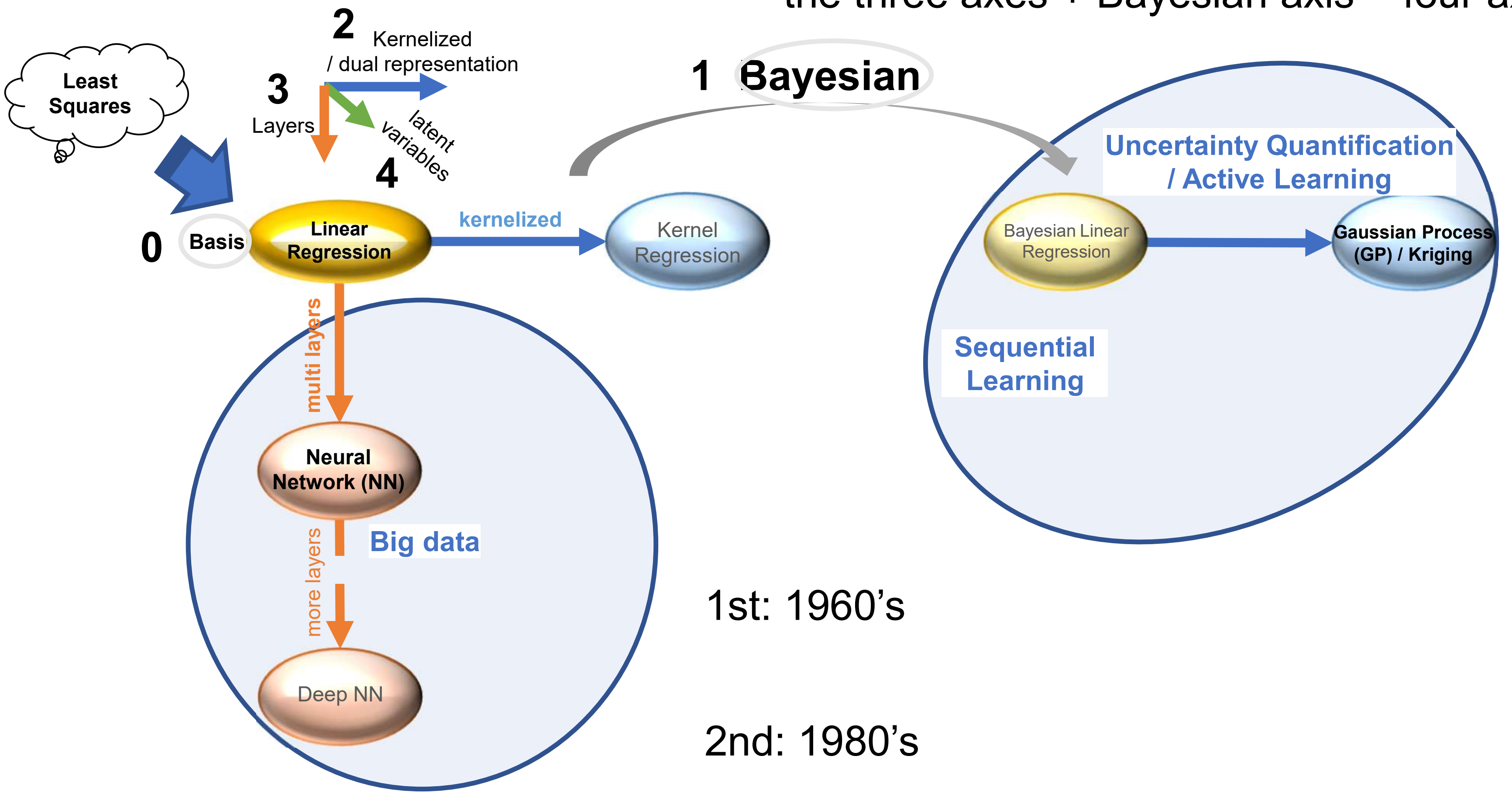
Lecture 10: Generalized Linear Model (GLM), Neural Network

Dr. Daigo Maruyama

Prof. Dr. Ali Elham

Key Components

the three axes + Bayesian axis = four axes



1st: 1960's

2nd: 1980's

3rd: 2010's - now

Lecture content

- From linear regression to generalized linear model (GLM)
  - Classification (opposite: regression) in applications
- To neural networks (NNs)
- Technical Issues in neural networks

GLM $\Rightarrow$ NNs

mind

The lecture of this time partially follows the Chapters 4 and 5 of the book:
Christopher M. Bishop "Pattern Recognition And Machine Learning" Springer-Verlag (2006)
The name of this book is shown as “PRML” when it is referred in the slides.

The lecture slides contains a few recent topics and evidence in neural networks.

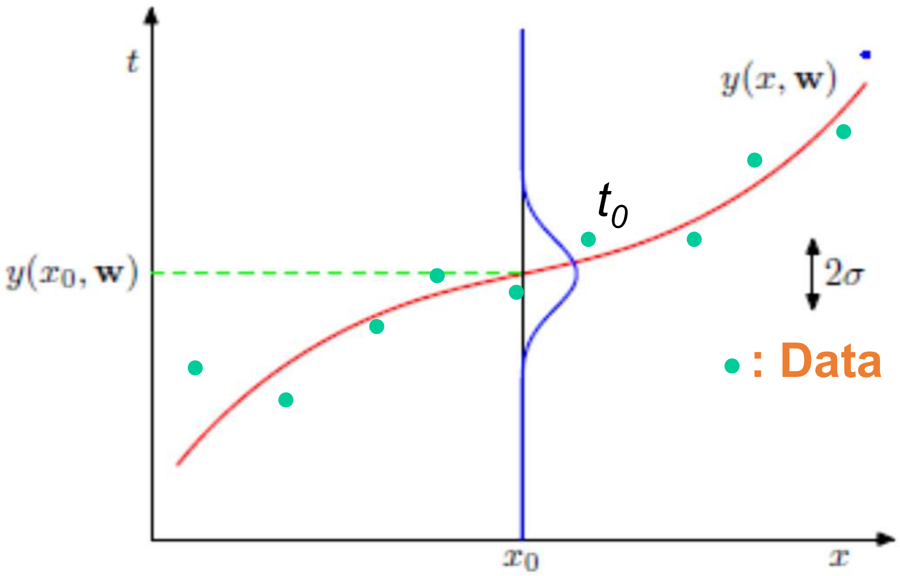
Lecture content

- From linear regression to generalized linear model (GLM)
    - Classification (opposite: regression) in applications

GLM $\Rightarrow$ NNs

mind

Linear Regression (REVIEW)



based on PRML, p. 29

$$\mathbf{x} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^{\mathrm{T}}$$

$$\mathbf{t} = \left(t^{(1)}, t^{(2)}, \cdots, t^{(N)}\right)^{\mathrm{T}}$$

N: sample size

Define a **regression model**

$$y(\boldsymbol{x}, \boldsymbol{w}) = \underbrace{\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})}_{\text{Linear regression model}}$$

Define a **Probabilistic model**

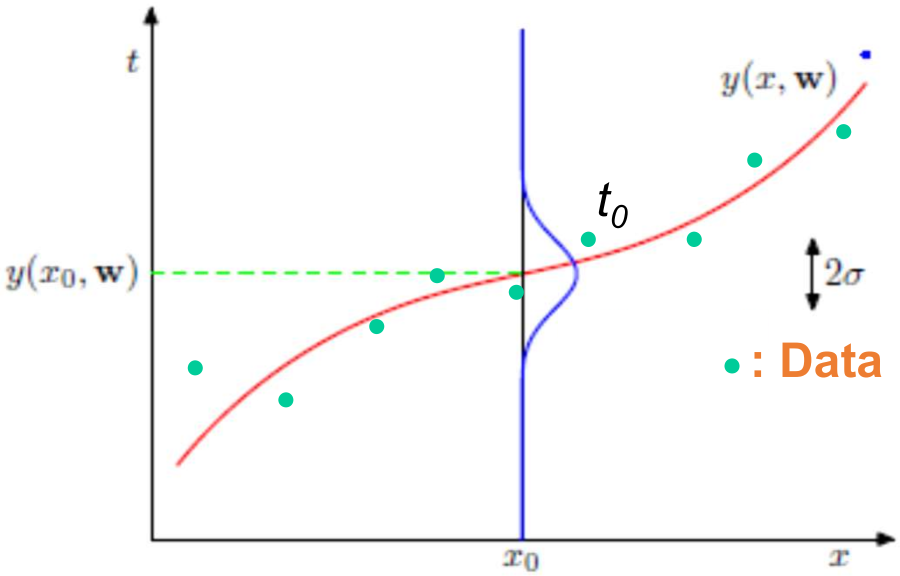
$$p(t|\boldsymbol{x}, \mu, \sigma) = \mathcal{N}(t|\mu(\boldsymbol{x}), \sigma^2)$$

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w})$$

$\mu(\boldsymbol{x})$: the regression model

$$\begin{aligned} p(t|\boldsymbol{x}, \boldsymbol{w}, \sigma) &= \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}), \sigma^2) \\ &= \mathcal{N}\left(t\middle|\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}), \sigma^2\right) \end{aligned}$$

Linear Regression (REVIEW)



based on PRML, p. 29

$$\mathbf{x} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^{\mathrm{T}}$$
$$\mathbf{t} = \left(t^{(1)}, t^{(2)}, \cdots, t^{(N)}\right)^{\mathrm{T}}$$

N: sample size

Define a **regression model**

$$y(\boldsymbol{x}, \boldsymbol{w}) = \underbrace{\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})}_{\text{Linear regression model}}$$

Define a **Probabilistic model**

$$p(t|\boldsymbol{x}, \mu, \sigma) = \mathcal{N}(t|\mu(\boldsymbol{x}), \sigma^2)$$

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w})$$

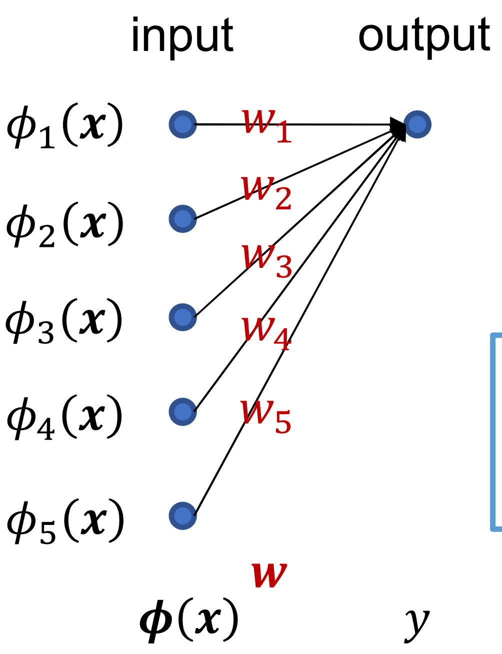
$\mu(\boldsymbol{x})$: the regression model

After learning $\boldsymbol{w}$ by data: **(point estimate)**

$$\hat{\mu}(\boldsymbol{x}_{new}) = \widehat{\boldsymbol{w}}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_{new})$$

Linear Regression (extension to multiple output)

$$y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = w_1\phi_1(\boldsymbol{x}) + w_2\phi_2(\boldsymbol{x}) + \cdots + w_M\phi_M(\boldsymbol{x})$$

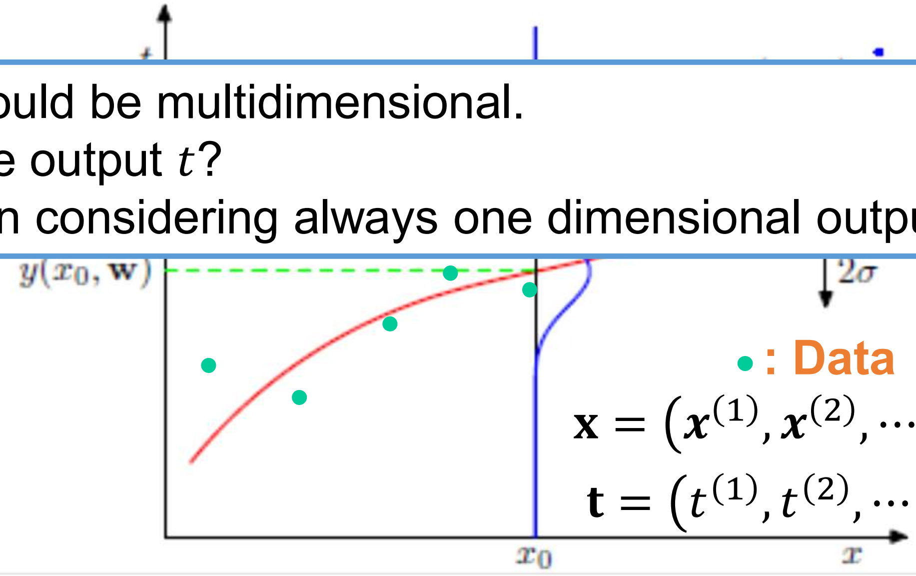


$M = 5$

$$\hat{\boldsymbol{w}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

$$= \begin{pmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix}\begin{pmatrix} t^{(1)} \\ \vdots \\ t^{(N)} \end{pmatrix}$$

The input $\boldsymbol{x}$ could be multidimensional.
How about the output t?
(we have been considering always one dimensional output)



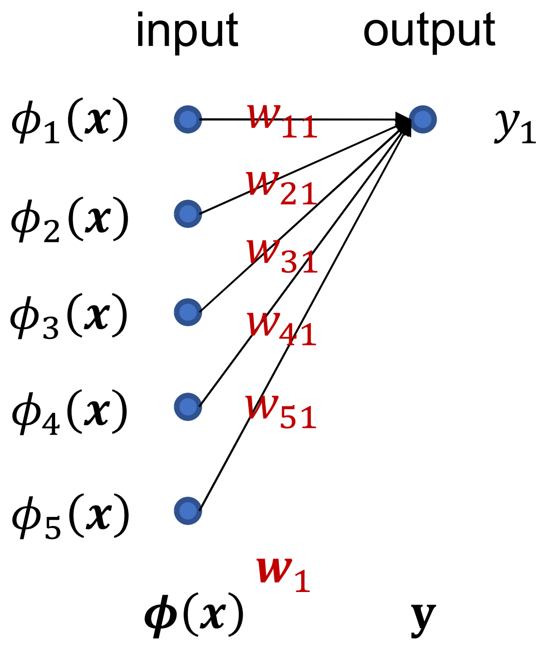
● : Data

$$\mathbf{x} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^{\mathrm{T}}$$

$$\mathbf{t} = \left(t^{(1)}, t^{(2)}, \cdots, t^{(N)}\right)^{\mathrm{T}}$$

Linear Regression (multiple output)

$$y(\boldsymbol{x}, \boldsymbol{w}_1) = \boldsymbol{w}_1{}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) = w_{11}\phi_1(\boldsymbol{x}) + w_{21}\phi_2(\boldsymbol{x}) + \cdots + w_{M1}\phi_M(\boldsymbol{x})$$



$$\widehat{\boldsymbol{w}}_1 = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}_1$$

$$= \begin{pmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix} \begin{pmatrix} t_1{}^{(1)} \\ \vdots \\ t_1{}^{(N)} \end{pmatrix} = \begin{pmatrix} \widehat{w}_{11} \\ \vdots \\ \widehat{w}_{51} \end{pmatrix}$$

Consider a multiple output case:

$$\mathbf{T} = \left(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots, \boldsymbol{t}^{(N)}\right)^{\mathrm{T}}$$

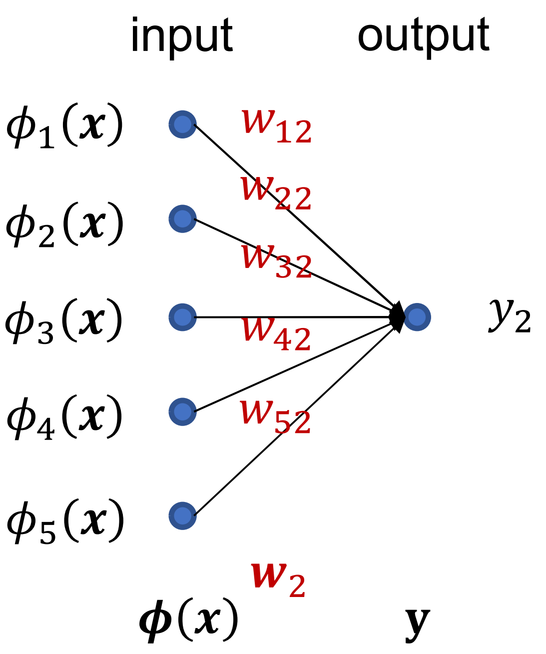
by focusing on the i th component

$$\mathbf{t}_i = \left(t_i{}^{(1)}, t_i{}^{(2)}, \cdots, t_i{}^{(N)}\right)^{\mathrm{T}}$$

At first, focusing on the first component y_1 in $\mathbf{y}$

Linear Regression (multiple output)

$$y(\boldsymbol{x}, \boldsymbol{w}_2) = {\boldsymbol{w}_2}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) = w_{12}\phi_1(\boldsymbol{x}) + w_{22}\phi_2(\boldsymbol{x}) + \cdots + w_{M2}\phi_M(\boldsymbol{x})$$



$$\hat{\boldsymbol{w}}_2 = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}_2$$

$$= \begin{pmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix} \begin{pmatrix} {t_2}^{(1)} \\ \vdots \\ {t_2}^{(N)} \end{pmatrix} = \begin{pmatrix} \hat{w}_{12} \\ \vdots \\ \hat{w}_{52} \end{pmatrix}$$

Consider a multiple output case:

$$\mathbf{T} = \left(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots, \boldsymbol{t}^{(N)}\right)^{\mathrm{T}}$$

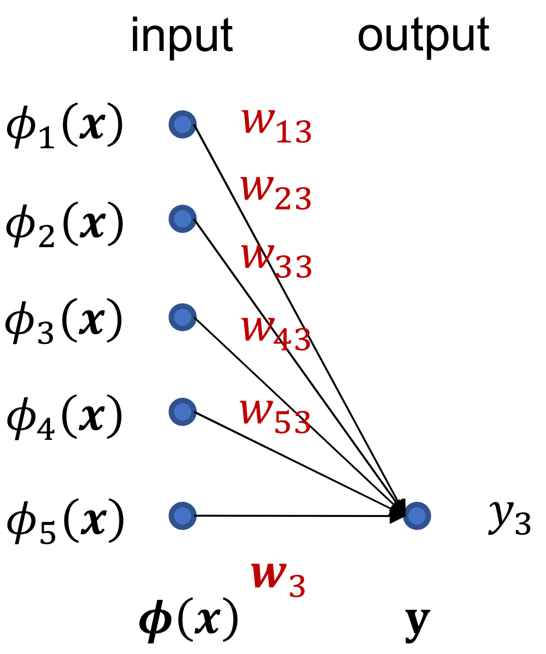
by focusing on the i th component

$$\mathbf{t}_i = \left({t_i}^{(1)}, {t_i}^{(2)}, \cdots, {t_i}^{(N)}\right)^{\mathrm{T}}$$

Then, focusing on the second component y_2 in $\mathbf{y}$

Linear Regression (multiple output)

$$y(\boldsymbol{x}, \boldsymbol{w}_3) = {\boldsymbol{w}_3}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) = w_{13}\phi_1(\boldsymbol{x}) + w_{23}\phi_2(\boldsymbol{x}) + \cdots + w_{M3}\phi_M(\boldsymbol{x})$$



$$\widehat{\boldsymbol{w}}_3 = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}_3$$
$$= \begin{pmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix} \begin{pmatrix} {t_3}^{(1)} \\ \vdots \\ {t_3}^{(N)} \end{pmatrix} = \begin{pmatrix} \widehat{w}_{13} \\ \vdots \\ \widehat{w}_{53} \end{pmatrix}$$

Consider a multiple output case:

$$\mathbf{T} = \left(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots, \boldsymbol{t}^{(N)}\right)^{\mathrm{T}}$$

by focusing on the i th component

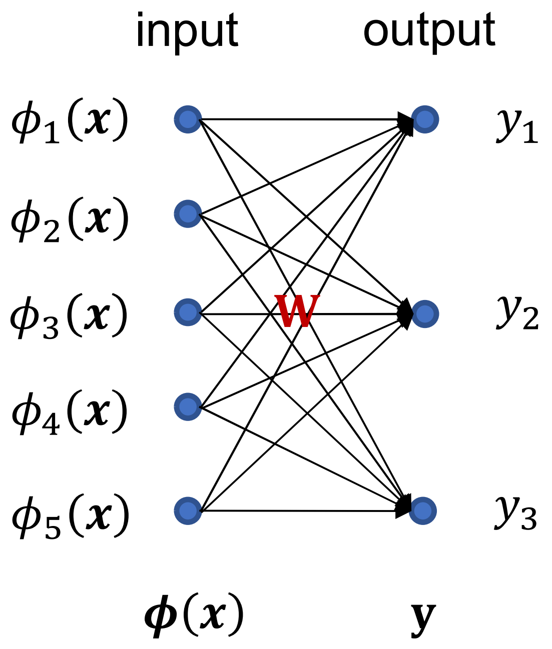
$$\mathbf{t}_i = \left({t_i}^{(1)}, {t_i}^{(2)}, \cdots, {t_i}^{(N)}\right)^{\mathrm{T}}$$

<u>The output $\mathbf{y}$ is three dimensional in this example.</u>

Linear Regression (multiple output)

$$\mathbf{y}(\boldsymbol{x},\mathbf{W}) = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$$

The tuned parameter $\hat{\boldsymbol{w}}_i$ (i=1,2,3) can be obtained at the same time as $\hat{\mathbf{W}}$.



Looks like a neural network?

$$\hat{\mathbf{W}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{T}$$

$$= \begin{pmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix}\begin{pmatrix} {t_1}^{(1)} & \cdots & {t_3}^{(1)} \\ \vdots & \ddots & \vdots \\ {t_1}^{(N)} & \cdots & {t_3}^{(N)} \end{pmatrix} = \begin{pmatrix} \hat{w}_{11} & \cdots & \hat{w}_{13} \\ \vdots & \ddots & \vdots \\ \hat{w}_{51} & \cdots & \hat{w}_{53} \end{pmatrix}$$

$$\hat{\mathbf{W}} = (\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, \hat{\boldsymbol{w}}_3)$$

Consider a multiple output case:

$$\mathbf{T} = \left(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots, \boldsymbol{t}^{(N)}\right)^{\mathrm{T}}$$

This is still the **linear regression model**.

The objective is to obtain $\hat{\mathbf{W}}$ or $p(\mathbf{W})$ (by using data).

Generalized Linear Model (GLM)

Linear regression

$$\boxed{y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})}$$

the single output form

$$\widehat{\boldsymbol{w}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

Analytical solution:
The important property of linear regression

Generalized linear model (GLM)

$$\boxed{y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)}$$

f: **nonlinear function**

To obtain $\widehat{\boldsymbol{w}}$, we need
numerical optimization tools.

Even if we lose this nice property (analytical solution),
we will obtain further benefit!

Generalized Linear Model (GLM)

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)$$

General ideas of using this form:

1. Transformation to probability output (0-1)

   Classification

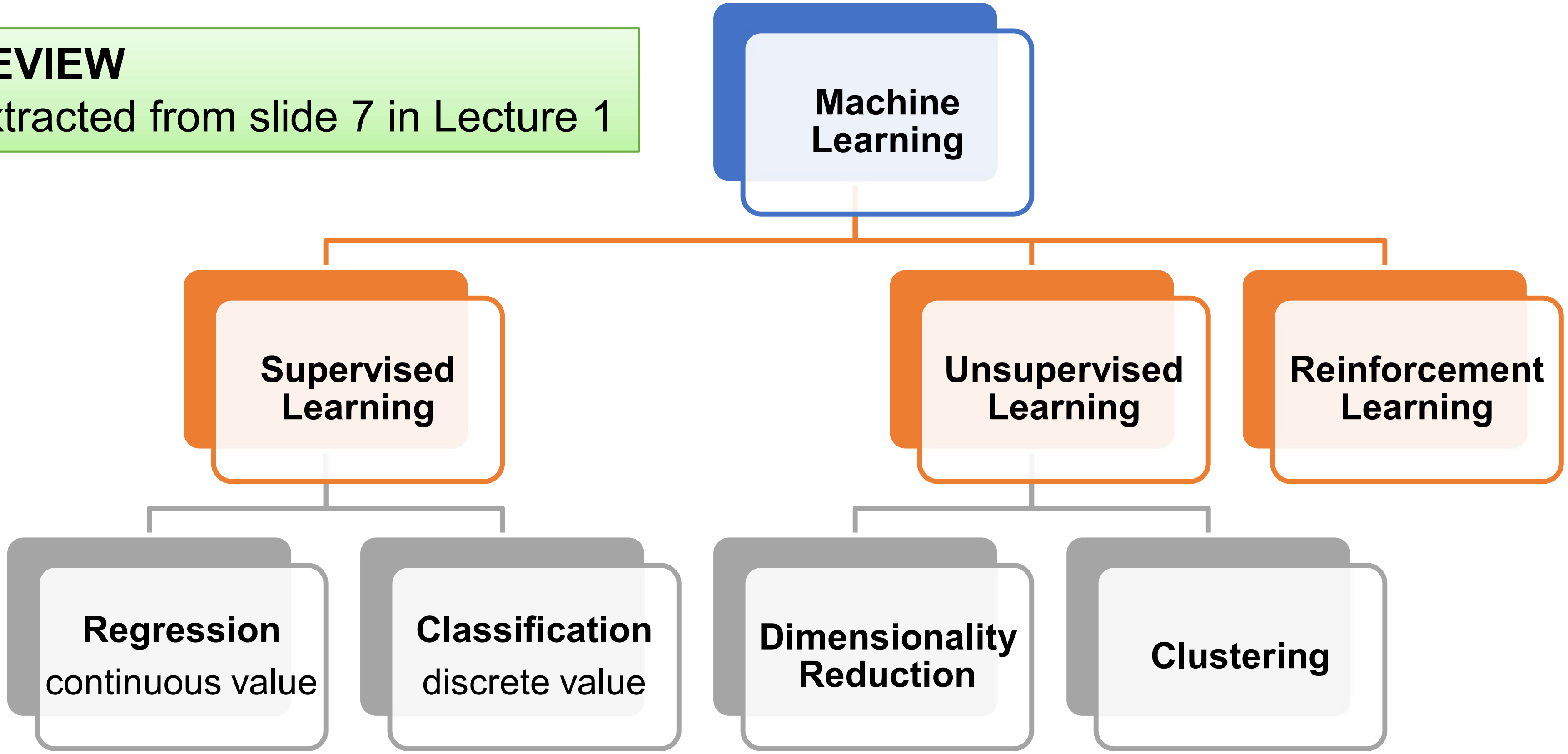
2. Efficient construction of a complex function

   Neural Networks

1+2 is possible.

Machine Learning Classification by Use/Application

REVIEW
Extracted from slide 7 in Lecture 1



In this course, machine learning classification is done by **methods and their concepts**.

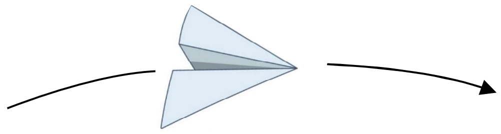
➡ Then the use/application is naturally derived/understood.

Classification (Example: Binomial case)

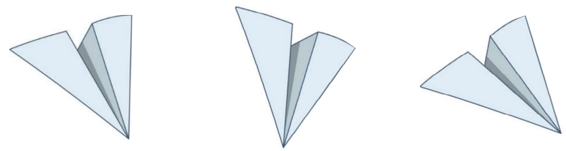
The example in slide 7 in Lecture 5

Example:

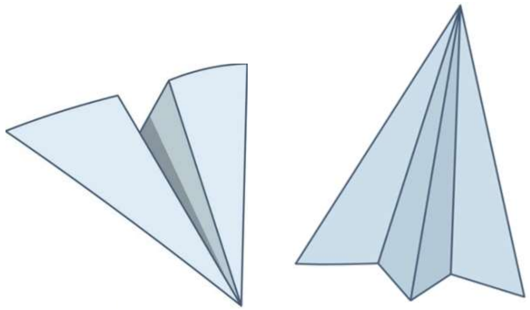
You threw a paper plane three times.



Observed Data: all the three times they were front.



Front **Back**



probability μ

probability $1-\mu$

1 (front) or 0 (back):
Discrete output value

$0 \le \mu \le 1$

What is the probability μ (front)?

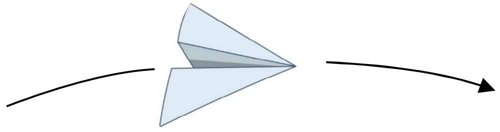
We want to predict $p(\mu)$.

and $1 - p(\mu)$

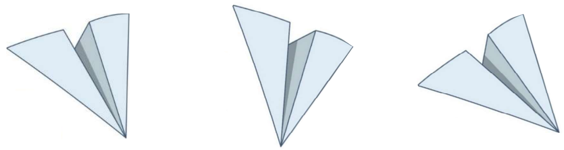
Classification (Example: Binomial case)

Another example (the problem slightly change):

You threw three different paper planes.

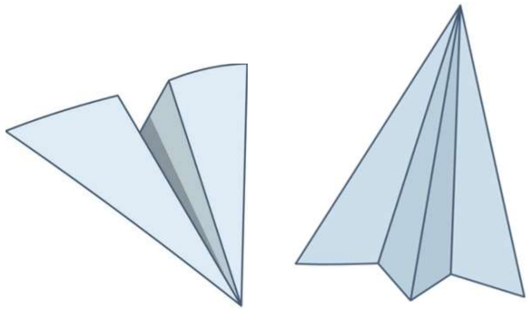


Observed Data: all the three planes were front.



Shape1 $\boldsymbol{x}^{(1)}$ Shape2 $\boldsymbol{x}^{(2)}$ Shape3 $\boldsymbol{x}^{(3)}$

Front **Back**



probability $\mu(\boldsymbol{x})$

probability $1-\mu(\boldsymbol{x})$

$0 \leq \mu(\boldsymbol{x}) \leq 1$

What is the probability $\mu(\boldsymbol{x})$?

We want to predict $p\big(\mu(\boldsymbol{x})\big)$.

in the regression (curve fitting):

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w}) \ = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$$

Classification (Example: Binomial case)

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right)$$

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) \in [-\infty, \infty]$$

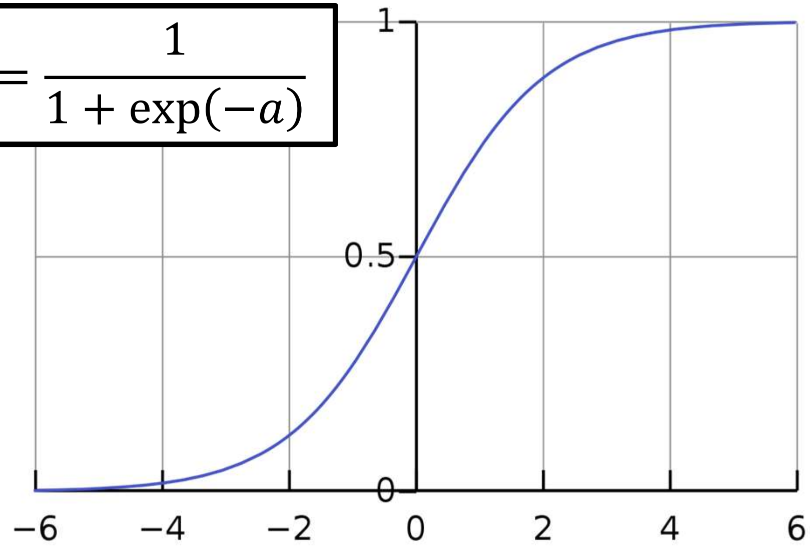
nonlinear transformation

$$f\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right) \in [0,1]$$

to a probability like $p\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right) \in [0,1]$

f: sigmoid function

$$f(a) = \frac{1}{1 + \exp(-a)}$$



https://en.wikipedia.org/wiki/Sigmoid_function

$\boldsymbol{\phi}(\boldsymbol{x})$: **Input** - shape of the paper plane

$\boldsymbol{w}$: unknown (we want to determine by data)

f: sigmoid function

y: **Output** - front or back (1 or 0)

$$\hat{\mu}(\boldsymbol{x}) = f\left(\widehat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right)$$

The output $\hat{\mu}(\boldsymbol{x}_{new})$ is interpreted as a probability when front.

Classification (Example: Binomial case)

Another example:

$\boldsymbol{\phi}(\boldsymbol{x})$: **Input** - weight, lung capacity, etc.

$\boldsymbol{w}$: unknown (we want to determine by data)

f: sigmoid function

y: **Output** - disease or not (1 or 0)

Classification (Example: Multiple case)

A new picture was given.

You want to classify it from the following three possibilities:

$$p(\text{"}dog\text{"}) = A?$$

$$p(\text{"}cat\text{"}) = B?$$

$$p(\text{"}fox\text{"}) = C?$$

$A + B + C$ has to be 1.

Classification (Example: Multiple case)

f: softmax function

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)$$

$$f(\boldsymbol{a})_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

$$\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) \in [-\infty, \infty]$$

nonlinear transformation

$$f\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right) \in [0,1]$$

to a probability like
$p\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right) \in [0,1]$

$\boldsymbol{\phi}(\boldsymbol{x})$: **Input** – RGB numbers at each pixel

$\boldsymbol{w}$: unknown (we want to determine by data)

f: softmax function

y: **Output** – “dog” or “cat” or “fox” (e.g.: 1 or 2 or 3)

Classification

Let's summarize the processes by following **the process as usual**.

1. **Define a regression model**

2. **Define probabilistic model**

$\Downarrow$

A likelihood function
(or posterior) $\Leftarrow$ Data (and prior)

$\Downarrow$

The point estimate (by MLE) of $\boldsymbol{w}$

optimization

We do not consider the Bayesian approach as *the probability distribution* of $\boldsymbol{w}$
(Because analytical solutions are not expected).

Classification (2 classes: binary)

The regression model

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w}) = sigmoid\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right)$$

called logistic regression
(even though this is classification…)

The probabilistic model

$$p(t|\mu) = \mathrm{Bern}(t|\mu) = \mu^{t}(1-\mu)^{1-t}$$

$$= p(t|\boldsymbol{w})$$

Bernoulli distribution
(see Lecture 3, slide 19)

because <u>the output is discrete</u>
(see Lecture 3, slide 18).

The likelihood function

$$p(\mathbf{t}|\boldsymbol{w}) = \prod_{n=1}^{N} y(\boldsymbol{x}, \boldsymbol{w})^{t^{(n)}}\left(1 - y(\boldsymbol{x}, \boldsymbol{w})\right)^{1-t^{(n)}}$$

Data
$$\mathbf{x} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^{\mathrm{T}}$$
$$\mathbf{t} = \left(t^{(1)}, t^{(2)}, \cdots, t^{(N)}\right)^{\mathrm{T}}$$

$$E(\boldsymbol{w}) = -\ln p(\mathbf{t}|\boldsymbol{w})$$

Take neg. log as usual
$$\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\mathrm{argmin}}\, E(\boldsymbol{w})$$

Data: generated "<u>i.i.d.</u>"
(see Lecture 2, slide 27)

Classification (multiple classes)

The regression model

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w}) = softmax\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)$$

When “dog”, “cat”, “fox”, $K = 3$.

The probabilistic model

$$p(\boldsymbol{t}|\boldsymbol{\mu}) = \mathrm{Cat}(\boldsymbol{t}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{\,x_k}$$

Categorical distribution
(see Lecture 3, slide 21)

because <u>the output is discrete and multiple</u>
(see Lecture 3, slide 18).

The likelihood function

$$p(\mathbf{t}|\boldsymbol{w}) = \prod_{n=1}^{N} y(\boldsymbol{x}, \boldsymbol{w})^{t^{(n)}} \left(1 - y(\boldsymbol{x}, \boldsymbol{w})\right)^{1-t^{(n)}}$$

Data
$$\mathbf{x} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^{\mathrm{T}}$$
$$\mathbf{t} = \left(t^{(1)}, t^{(2)}, \cdots, t^{(N)}\right)^{\mathrm{T}}$$

$$\Rightarrow \quad E(\boldsymbol{w}) = -\ln p(\mathbf{t}|\boldsymbol{w})$$

Take neg. log as usual
$$\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, E(\boldsymbol{w})$$

Data: generated “<u>i.i.d.</u>”
(see Lecture 2, slide 27)

Neural Networks (for regression – e.g. curve fitting)

The regression model

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)$$

The probabilistic model

$$p(t|\boldsymbol{x}, \mu, \sigma) = \mathcal{N}(t|\mu(\boldsymbol{x}), \sigma^2)$$

(Isotropic) Gaussian distribution
(used often since previous lectures)

because <u>the output is continuous</u>
(see Lecture 3, slide 18).

The likelihood function

$$p(\mathbf{t}|\mathbf{X}, \boldsymbol{w}, \sigma) = \prod_{n=1}^{N} \mathcal{N}\left(t^{(n)} \middle| y\left(\boldsymbol{x}^{(n)}, \boldsymbol{w}\right), \sigma^2\right)$$

Data $\quad \mathbf{X} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^{\mathrm{T}}$
$\quad \mathbf{t} = \left(t^{(1)}, t^{(2)}, \cdots, t^{(N)}\right)^{\mathrm{T}}$

$\Rightarrow$ $E(\boldsymbol{w}) = -\ln p(\mathbf{t}|\boldsymbol{w})$ **Least squares**

Take neg. log as usual $\quad \hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, E(\boldsymbol{w})$

Data: generated “<u>i.i.d.</u>”
(see Lecture 2, slide 27)

Neural Networks (for regression: multiple output)

The regression model

$$\boldsymbol{\mu}(\boldsymbol{x}) = \mathbf{y}(\boldsymbol{x}, \mathbf{W}) = f.\left(\mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right)$$

$f. : f$ is applied to each component

The probabilistic model

(Isotropic) Gaussian distribution
(used often since previous lectures)

$$p(\boldsymbol{t}|\boldsymbol{x}, \mu, \sigma) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{\mu}(\boldsymbol{x}), \sigma^2\mathbf{I})$$

because the output is continuous
(see Lecture 3, slide 18).

The likelihood function

$$p(\mathbf{T}|\mathbf{X}, \mathbf{W}, \sigma) = \prod_{n=1}^{N} \mathcal{N}\left(\boldsymbol{t}^{(n)}|\mathbf{y}\left(\boldsymbol{x}^{(n)}, \mathbf{W}\right), \sigma^2\mathbf{I}\right)$$

Data
$$\mathbf{X} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^{\mathrm{T}}$$
$$\mathbf{T} = \left(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots, \boldsymbol{t}^{(N)}\right)^{\mathrm{T}}$$

⇨ $E(\mathbf{W}) = -\ln p(\mathbf{T}|\mathbf{W})$ **Least squares**

Take neg. log as usual

$\widehat{\mathbf{W}}$ is obtained.

Data: generated "i.i.d."
(see Lecture 2, slide 27)

$\mathbf{W}$ can be expressed by a vector $\boldsymbol{w}$ not by a matrix.

Lecture content

- To neural networks (NNs)

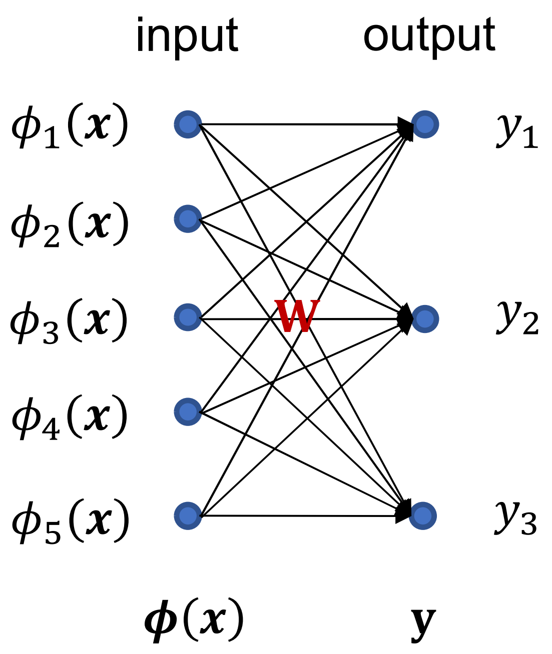
GLM $\Rightarrow$ NNs

mind

Linear Regression (multiple output) - Repost

$$\mathbf{y}(\boldsymbol{x}, \mathbf{W}) = \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})$$

The tuned parameter $\hat{\boldsymbol{w}}_i$ (i=1,2,3) can be obtained at the same time as $\widehat{\mathbf{W}}$.



$$\widehat{\mathbf{W}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{T}$$

$$= \begin{pmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix} \begin{pmatrix} t_1^{(1)} & \cdots & t_3^{(1)} \\ \vdots & \ddots & \vdots \\ t_1^{(N)} & \cdots & t_3^{(N)} \end{pmatrix} = \begin{pmatrix} \hat{w}_{11} & \cdots & \hat{w}_{13} \\ \vdots & \ddots & \vdots \\ \hat{w}_{51} & \cdots & \hat{w}_{53} \end{pmatrix}$$

$$\widehat{\mathbf{W}} = (\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, \hat{\boldsymbol{w}}_3)$$

Consider a multiple output case:

$$\mathbf{T} = \left(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots, \boldsymbol{t}^{(N)}\right)^{\mathrm{T}}$$

This is still the **linear regression model**.

Looks like a neural network?

The objective is to obtain $\widehat{\mathbf{W}}$ or $p(\mathbf{W})$ (by using data).

From Linear Regression to Neural Networks

A Linear regression (in general)

$$\mathbf{y}(\boldsymbol{x}, \mathbf{W}) = \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})$$

Then, nonlinear transformation

$$\mathbf{y}(\boldsymbol{x}, \mathbf{W}) = f.\left(\mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)$$

$f.$:   a common nonlinear function f is applied to each component

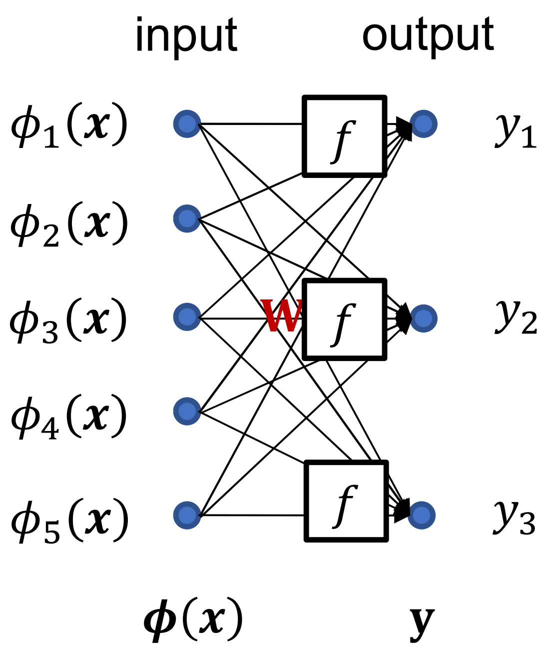
f is a function of a **scalar** input.

e.g. $y_n(\boldsymbol{x}, \boldsymbol{w}_n) = f\left({\boldsymbol{w}_n}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)$

$n = 1,2,3$

$$\mathbf{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3)$$

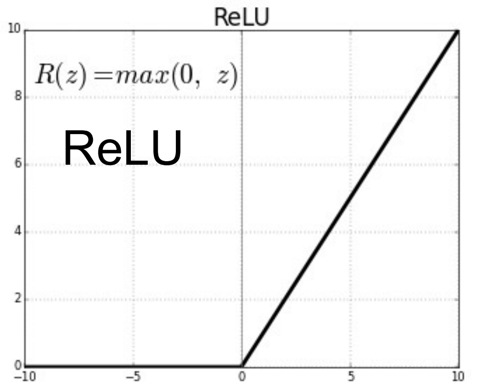
Simple extension of the linear regression



The nonlinear function f

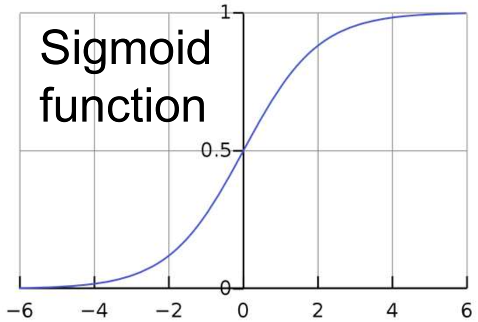
If the application is **regression** (the outputs are <u>continuous</u> values)

- Identity (no mapping)
- ReLU
- Sigmoid function
- …



If the application is **classification** (the outputs are <u>discrete</u> values)

- Sigmoid function (for 2 classes)
- Softmax function (for multi classes)



Important Properties in GLM in General

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)$$

disadvantage

- $\widehat{\boldsymbol{w}}$ cannot be analytically obtained anymore.

$$\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, E(\boldsymbol{w})$$

by optimizer

⇒ Optimizer is required.

- Neither the posterior distributions $p(\boldsymbol{w})$ nor predictive distributions can be analytically obtained $p(t)$.

⇒ No spots of analytical solutions in the Bayesian approach.
The Bayesian approach is very challenging.

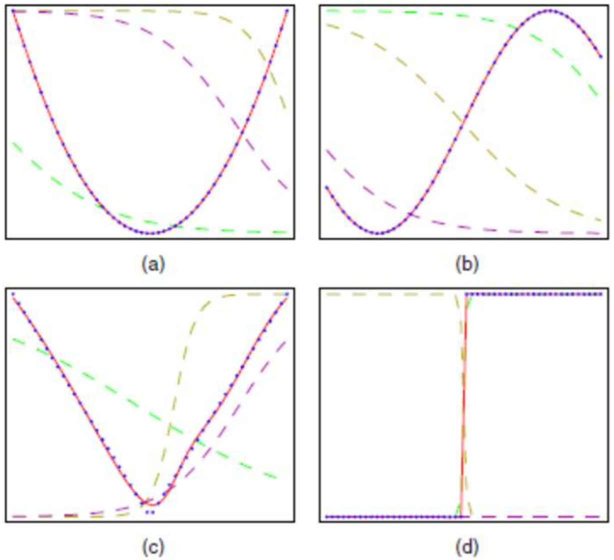
Important Properties in GLM in General

$$\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right)$$

advantage

- A variety of efficient expression of the function (the regression model)

One of the strongest points in neural networks



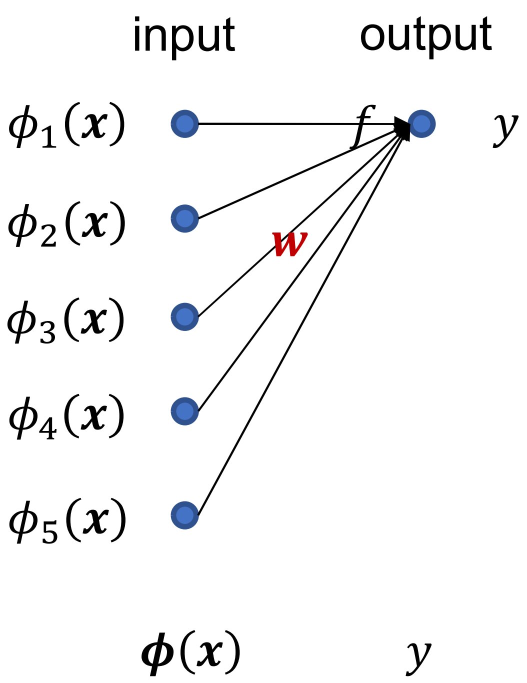
A variety of functions

PRML, Fig. 5.3

Important Properties in GLM in General

Once we lost the nice properties (analytical solutions), we can focus on the trade-off between the complexity of the function (the regression model) and efficiency (computational time in the learning process)

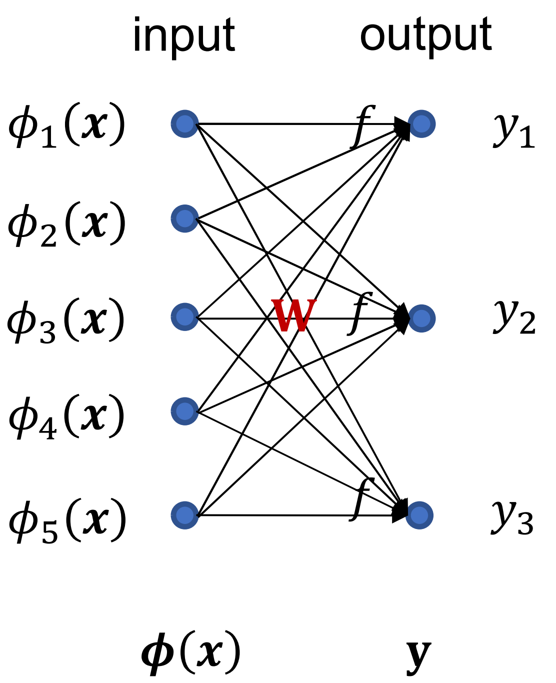
$$y(\boldsymbol{x}, \boldsymbol{w}) = f.\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})\right)$$



Important Properties in GLM in General

Once we lost the nice properties (analytical solutions), we can focus on the trade-off between the complexity of the function (the regression model) and efficiency (computational time in the learning process)

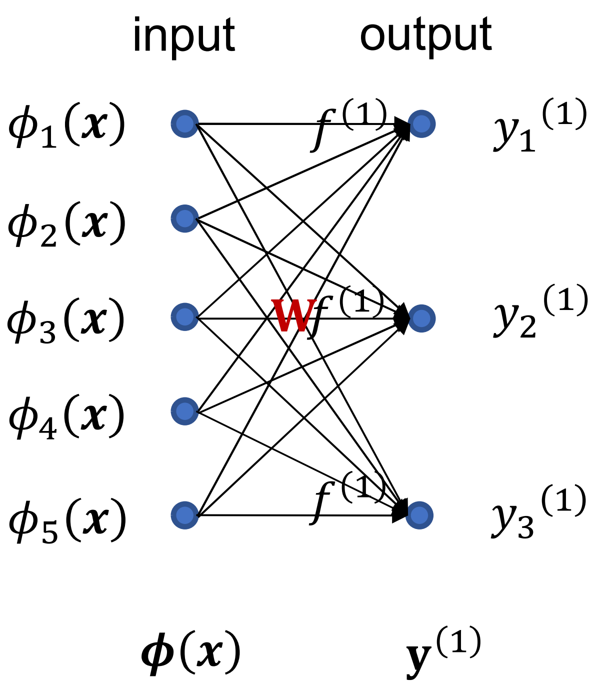
$$\mathbf{y}(\boldsymbol{x}, \mathbf{W}) = f.\left(\mathbf{W}\boldsymbol{\phi}(\boldsymbol{x})\right)$$



Important Properties in GLM in General

Once we lost the nice properties (analytical solutions), we can focus on the trade-off between the complexity of the function (the regression model) and efficiency (computational time in the learning process)

$$\mathbf{y}^{(1)}\left(\boldsymbol{x}, \mathbf{W}^{(1)}\right) = f^{(1)}.\left(\mathbf{W}^{(1)}\boldsymbol{\phi}(\boldsymbol{x})\right)$$



Important Properties in GLM in General

Once we lost the nice properties (analytical solutions), we can focus on the trade-off between the complexity of the function (the regression model) and efficiency (computational time in the learning process)

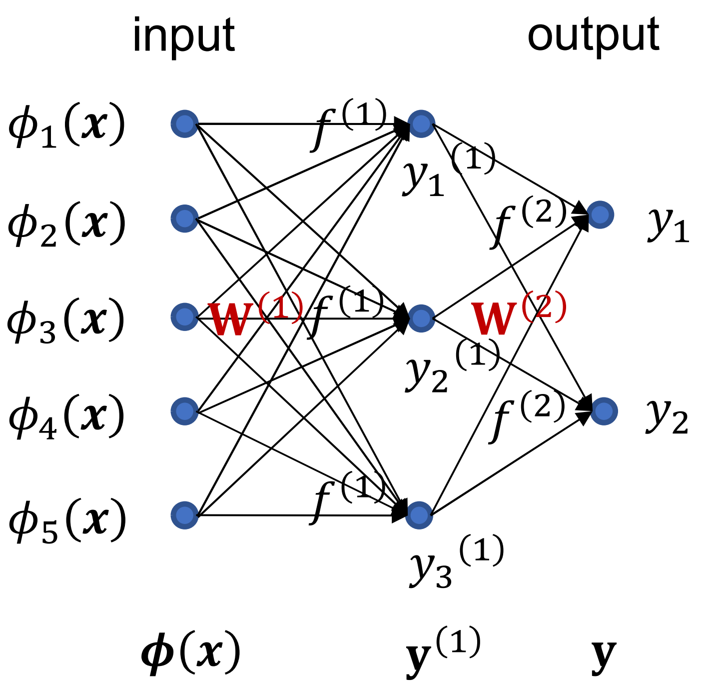
$$\mathbf{y}^{(1)}\left(\boldsymbol{x}, \mathbf{W}^{(1)}\right) = f^{(1)}.\left(\mathbf{W}^{(1)}\boldsymbol{\phi}(\boldsymbol{x})\right)$$

$$\mathbf{y}(\boldsymbol{x}, \mathbf{W}) = \mathbf{W}^{(2)}\underbrace{f^{(1)}.\left(\mathbf{W}^{(1)}\boldsymbol{\phi}(\boldsymbol{x})\right)}_{\mathbf{y}^{(1)}(\boldsymbol{x}, \mathbf{W})}$$

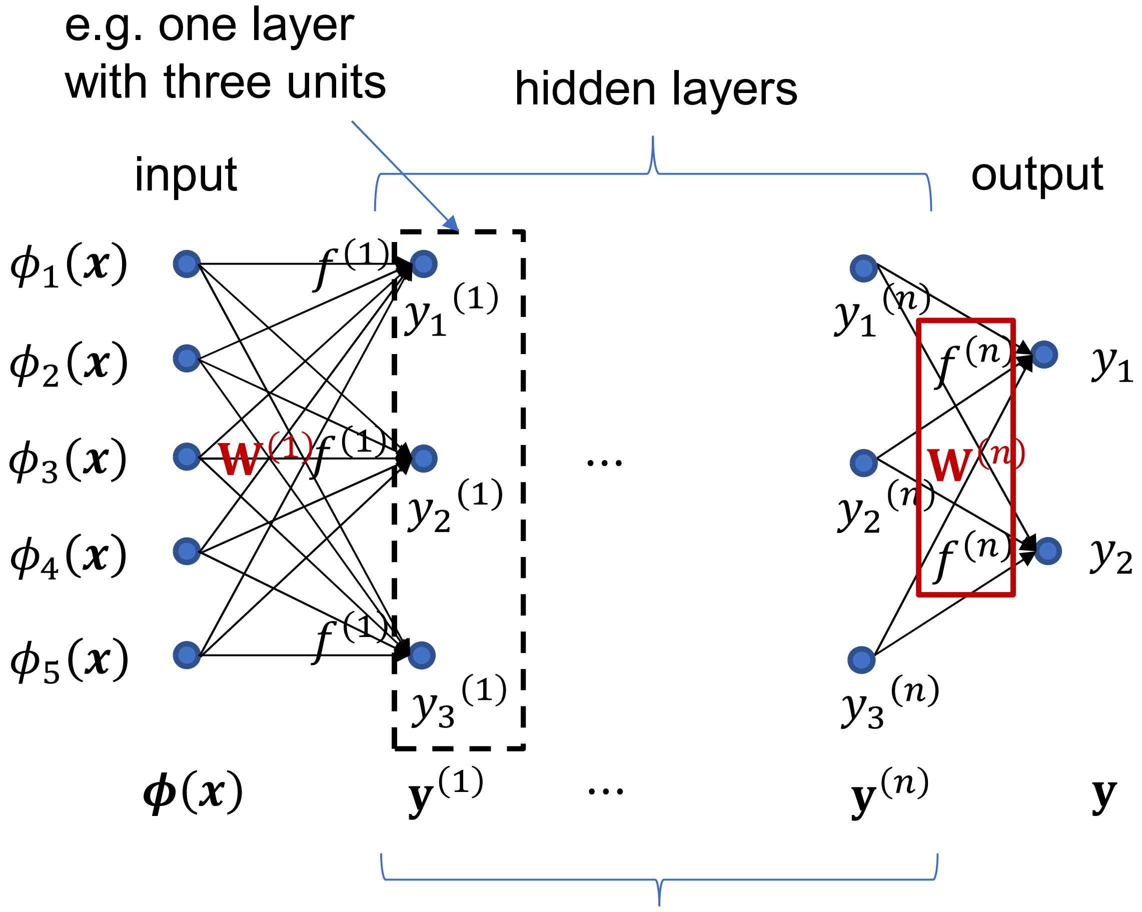
$\boldsymbol{w}$ is composed of $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$.

$$\mathbf{y}(\boldsymbol{x}, \boldsymbol{w}) = f^{(2)}.\left(\mathbf{W}^{(2)}f^{(1)}.\left(\mathbf{W}^{(1)}\boldsymbol{\phi}(\boldsymbol{x})\right)\right)$$
$$= f^{(2)}.\left(\mathbf{W}^{(2)}\mathbf{y}^{(1)}\right)$$

$$\left(\mathbf{y}^{(1)} = \mathbf{y}^{(1)}\left(\boldsymbol{x}, \mathbf{W}^{(1)}\right)\right)$$



Important Properties in GLM in General



If many layers, the model is called **deep** neural network.

If your objective is **classification**, you can set $\boxed{f^{(n)}}$ to the sigmoid/softmax function.

Regression model
$\mathbf{y}(\boldsymbol{x}, \boldsymbol{w})$

The dimensionality of $\boldsymbol{w}$, which is composed of $\mathbf{W}^{(1)}, \cdots, \mathbf{W}^{(n)}$, tends to be **large**.

Example: Gaussian Processes (GPs) for Classification

The regression model

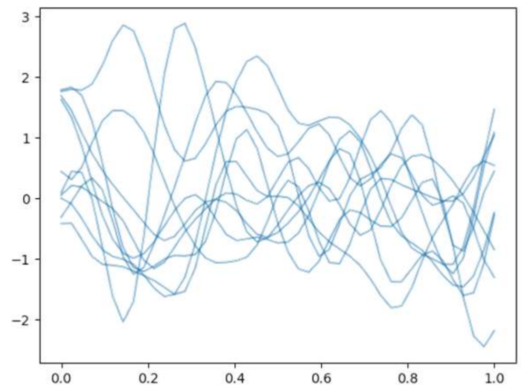
- $\mu(\boldsymbol{x}) = y(\boldsymbol{x}, \boldsymbol{w}) = sigmoid\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})\right)$

The probabilistic model

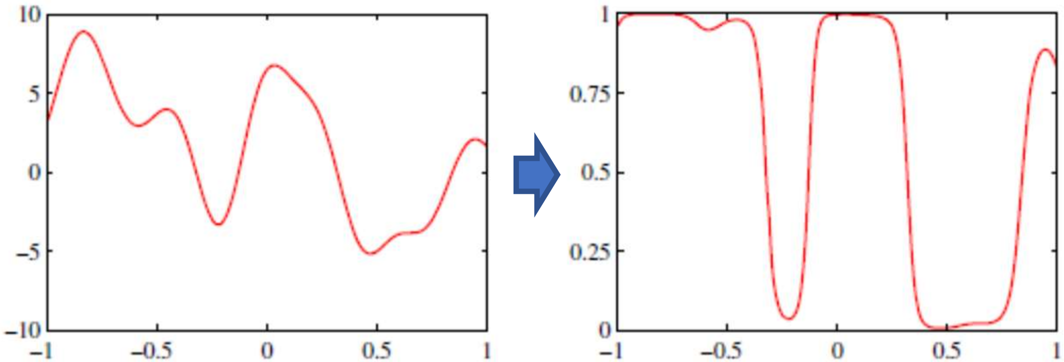
$$p(t|\mu) = \mathrm{Bern}(t|\mu) = \mu^{t}(1-\mu)^{1-t}$$

The same process then… (showing the process skipped)

- $\mu(\boldsymbol{x}) = sigmoid(\mathbf{GP})$



$\mathbf{GP}$ $\quad$ $sigmoid(\mathbf{GP})$



PRML, Fig. 6.11

Strength of Neural Networks

- Why not using complex linear regression models?
- Why not using other nonlinear regression models?
- Why not using Gaussian processes?

Prerequieste:
$\boldsymbol{\phi}$ in linear regression models can be nonlinear functions but need to be determined in advance.

$\boldsymbol{\phi}$: $\boldsymbol{x} \to \boldsymbol{s}$

$$f(s_1, \cdots, s_M)$$

$$f(w_1 s_1 + \cdots + w_M s_M) = f\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{s}\right) \qquad \boldsymbol{w}\text{: } M \text{ dim.}$$

$$w_1 {s_1}^2 + w_2 s_1 s_2 + w_3 {s_2}^2 + \cdots \qquad \boldsymbol{w}\text{: } M^2/2 \text{ dim.}$$

The increase of the number of the parameters from exponential to linear by using the ideas of layers.

Issues in Neural Networks

The difference between **NNs** and **Linear regression models**:

Optimization or **not** (in the **learning process**)

Note:
We saw that optimization was used in Gaussian processes (GPs) to determine the hyperprameters $\boldsymbol{\theta}$ by MLE. The problems in the optimization are different between GPs (see Lecture 9) and NNs.

The error function to be minimize
(when the probabilistic model is the Gaussian)

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} \left\{ \boldsymbol{t}^{(n)} - \mathbf{y}\left(\boldsymbol{x}^{(n)}, \underbrace{\boldsymbol{w}}\right) \right\}^2$$

N: a huge number (**big data**)   millions

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \, E(\boldsymbol{w})$$

The regularization can be naturally used

$$E_{reg}(\boldsymbol{w}) = E(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|^q$$

See Lecture 4, slide 35

Especially when **LASSO as sparse model** is very useful to decrease unnecessary parameters from the plenty of parameters in $\boldsymbol{w}$.

Issues in Neural Networks

How efficiently the learning process (the optimization process) is done

- Optimization techniques
- The properties of the NN model

Human's learning/experimental process from the data

Then, the goal:
As far as the prediction is good, the model and the chosen techniques are ok.

Details of all the theories behind are not clarified yet.

But now you know how to do it (**Bayesian neural network**)!

The Bayesian approach is not used (too expensive),
usually we split data into training data and validation data.

Simply the posterior and the predictive distribution cannot be obtained analytically.

(See Lecture 2, slide 7)

(Lecture 12: numerical approaches)

Lecture content

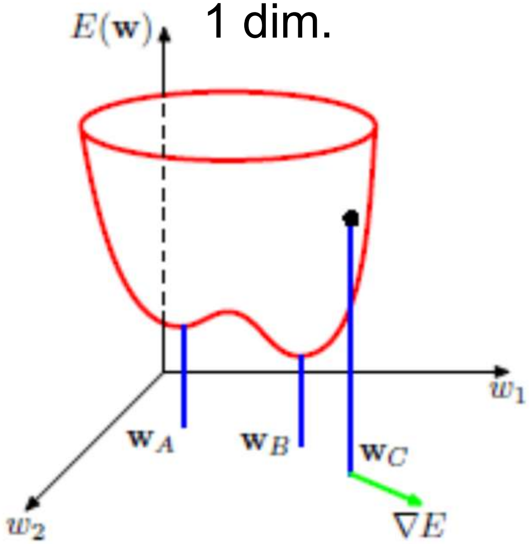
- Technical Issues in neural networks

Technical Issues in Optimization

We need to search for the global minimum of the function $E(\boldsymbol{w})$ in a very high-dimensional space $\boldsymbol{w}$.

PRML, Fig. 5.5

1 dim.



$\boldsymbol{w} = (w_1, w_2, \cdots, w_{10000} \quad)^{\mathrm{T}}$

millions of dim.

Topics of optimization (using some nice properties of the problem setting in NNs):

In such a high-dimensional space,

- **Gradient-based optimization**
  - Techniques to efficiently compute the gradients (**chain rule**) – called back propagation
  - Improved gradient opt. algorithms
    - AdaGrad
    - Adam
    - …many others

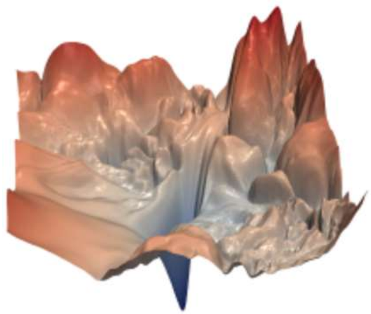
Most of open-source libraries in Python nowadays have these functionalities as standard.

Technical Issues in Optimization

To avoid large computational costs by using all data, a subset of the data is used to compute the gradient at each iteration of the optimization process.

- **Stochastic** gradient descent (SGD) algorithm
  - Drop-out
  - Over-parametrization
  - Parallelization (power of GPU)
  - …many others

As a result, the algorithm also contributes to avoid local minimum!



Li, H., Xu, Z., Taylor, G., Studer C. amd Goldstein, T., "visualizing the loss landscape of neural nets", 2018.

- Convolutional neural network (CNN) – parameters $\boldsymbol{w}$ can be drastically reduced.
- …many others

A variety of methods has been developed for these several years!

Recent Advanced Methods

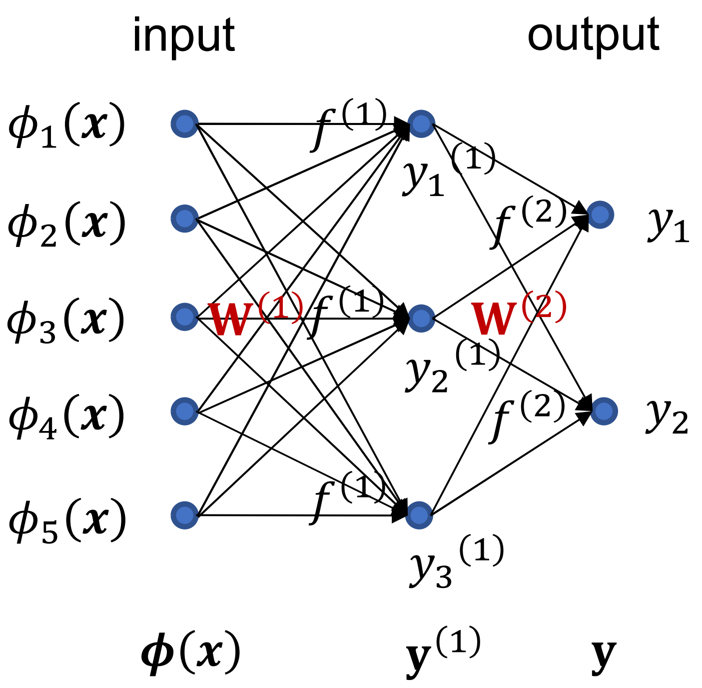
- AlexNet (2012)
  - 8 layers
  - 60 million parameters
- VGG19 Net (2014)
  - 46 layers
  - more than 100 million parameters
- Residual Network (ResNet) (2015)
  - 152 layers
  - 20 million parameters (fewer parameters per layer)

Hyperparameter Tuning

- How many layers do we set?
- How many parameters (the number of units per layer) do we set?
- …

Hyperparameter tuning

By changing these parameters, we try to find a combination of the parameters with the best score (good prediction of a validation data).



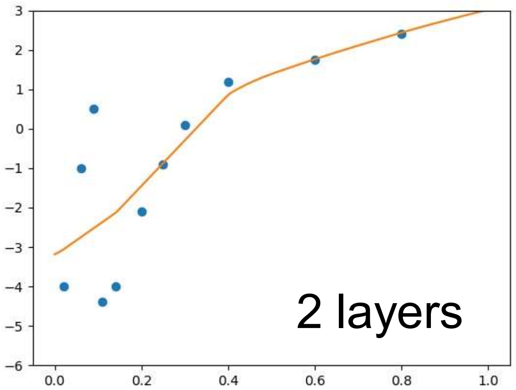
Mathematical Evidence/Proof

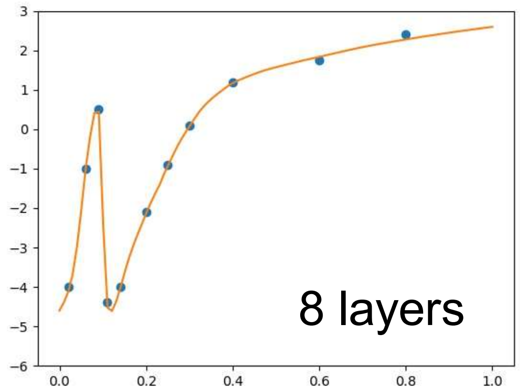
*AER: approximation error rate

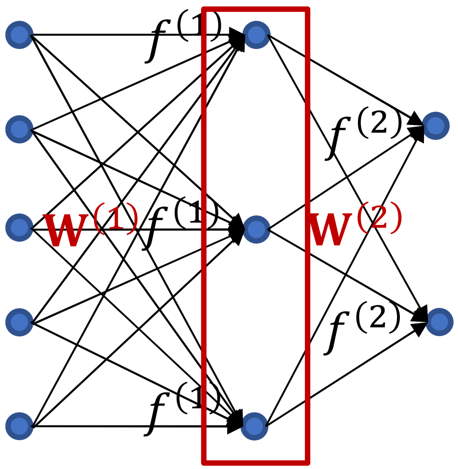
- Any <u>smooth functions</u> can be represented by 2 layers NNs (kernel methods such as GPs as well).
  - No other models can improve AER.

- When the target function is a <u>step</u> / <u>non-uniform</u> function, NNs with 4 layers or more achieve the best AER.
  - No other models can exceed the rate.

Example: A non-uniform function







Note:
When # of the units → ∞,
the model → a GP.

Perspective of Neural Networks

If you are interested…
- The Lottery Ticket Hypothesis
- PAC Bayes
- Double descent

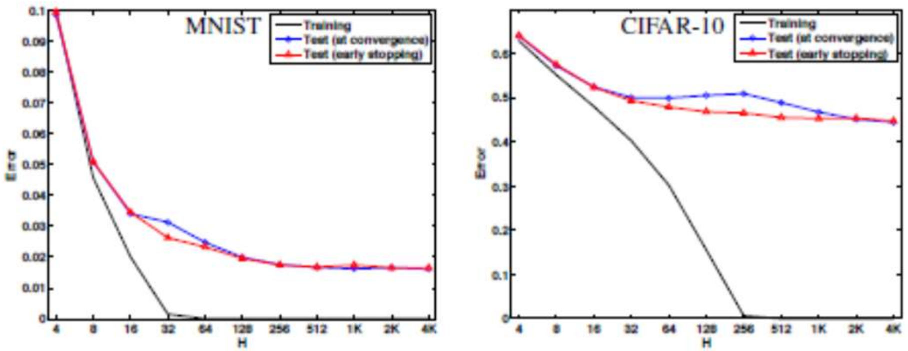
Mystery of Hundred millions of parameters

training data size

When the number of the parameters $\boldsymbol{w} > N$ ⇒ overfitting

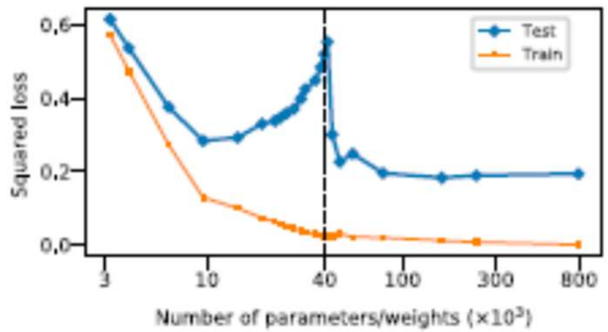
See Lecture 2, slide 6 (Fig. 1.8 in PRML)

More parameters can reduce the error.



Neyshabur, B., Tomioka, R., Salakhutdinov
, R. and Srebro, N., "Geometry of Optimization and Implicit Regularization in Deep Learning", 2017.

Avoiding the overfitting



Belkin, M., Hsu, D., Ma, S. and Mandal, S., "reconciling modern machine learning and the bias-variance trade-off", 2019.

Brief Summary

	Point estimate	*Bayesian*
Linear regression	Analytical solutions	Some useful properties available (Gaussian processes)
Nonlinear regression		
GLM (Classification, NNs)	**A variety of functions**	hard

focusing on this strength

Next Step

Neural networks (not only them but supervised learning techniques in general) are often used with unsupervised learning techniques.

Without these, the state of the art techniques are not discussed.

➡ Unsupervised learning methods in the next lecture