Course ID 80250993 Dates: 9/16-12/23/2021 @ I-205
Tencent/VooV Meeting 521 4678 6257 (passwd 1205)

Part I
Deterministic Machines for Supervised Learning

Xuegong Zhang
September 23, 2021

Course ID 80250993 Dates: 9/16-12/23/2021 @ I-205
Tencent/VooV Meeting 521 4678 6257 (passwd 1205)

Chapter 3
Linear Learning Machines

Xuegong Zhang
September 23, 2021

3.1
Fisher's Linear Discriminant

Sir Ronald Aylmer Fisher (R.A. Fisher)

(17 February 1890 – 29 July 1962)

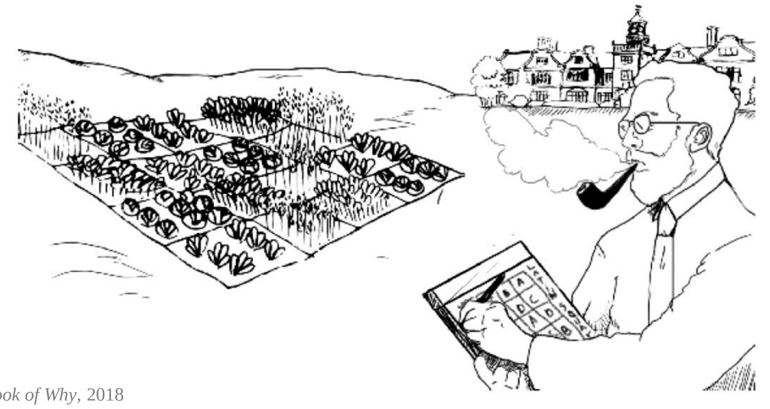
- British statistician and geneticist
  - "a genius who almost single-handedly created the foundations for modern statistical science"
  - "the single most important figure in 20th century statistics"
  - "the greatest of Darwin's successors".
- Some of the stuff he invited or popularized
  - ANOVA (analysis of variance)
  - Maximum likelihood
  - Fisher's z-distribution (F distribution)
  - Fisher's method for data fusion (meta-analysis)
  - The 0.05 cutoff of p-value, the notion of hull hypothesis
  - Fisher's exact test
  - Fisher's Discriminant Analysis (in 1936)
  - ……
  - *The Genetical Theory of Natural Selection* (1930)
  - *The Design of Experiments* (1935)

W

From Wikipedia

"Natural selection is a mechanism for generating an exceedingly high degree of improbability."

The controversial side of R.A. Fisher



Judea Pearl, *The Book of Why*, 2018

FIGURE 4.3. R. A. Fisher with one of his many innovations: a Latin square experimental design, intended to ensure that one plot of each plant type appears in each row (fertilizer type) and column (soil type). Such designs are still used in practice, but Fisher would later argue convincingly that a randomized design is even more effective. (*Source:* Drawing by Dakota Harr.)

Homework (optional): Why proving smoking harmful to human health was so difficult?

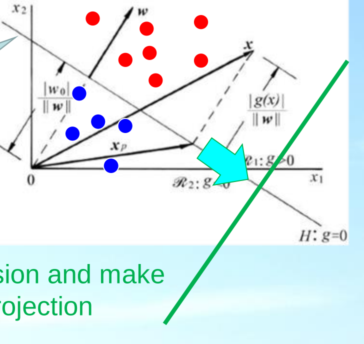
Linear Classifier / Linear Discriminant

$$g(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0$$

$$y = \operatorname{sgn}\left(\sum_{i=1}^{d} w_i x_i + w_0\right)$$

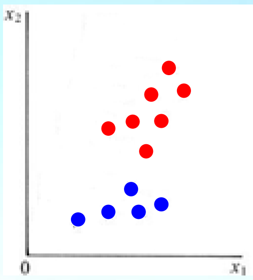
$$y = \begin{cases} +1 & \Rightarrow \text{class } A \text{ } or \text{ } \mathbf{x} \in \omega_1 \\ -1 & \Rightarrow \text{class } B \text{ } or \text{ } \mathbf{x} \in \omega_2 \end{cases}$$

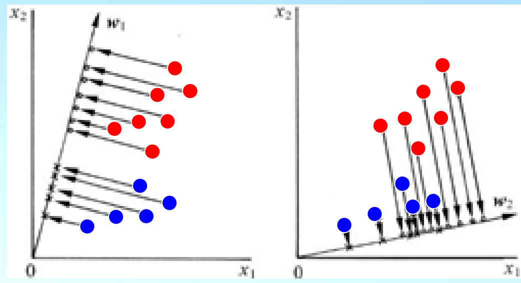
Decision line
Classification boundary
Discriminant



Projecting to one dimension and make decision based on the projection

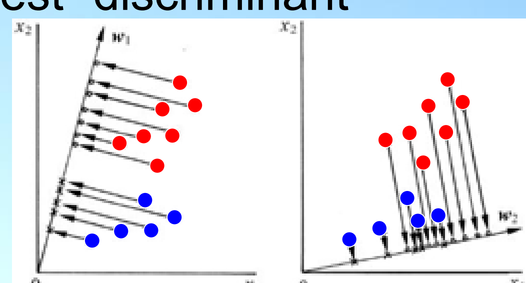
Fisher's criterion for "the best" discriminant





- Which projection is better? Why and How?
- Fisher: The best projection: samples of the two classes *"best separated"*
  - Within-class: as close as possible
  - Between-classes: as far as possible

Fisher's criterion for "the best" discriminant



- Sample set: $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$,

  where, Class 1 (ω_1): $\mathcal{X}_1 = \{\boldsymbol{x}_1^1, \cdots, \boldsymbol{x}_{N_1}^1\}$,

  Class 2 (ω_2): $\mathcal{X}_2 = \{\boldsymbol{x}_1^2, \cdots, \boldsymbol{x}_{N_2}^2\}$
- Projection:

  $\mathcal{X} \rightarrow \mathcal{Y}$: $y_i = \boldsymbol{w}^T \boldsymbol{x}_i, \; i = 1, \cdots, N$
- Task: find the $\boldsymbol{w}$ that best separates the two classes

The “closeness” of samples before projection

- In $\mathcal{X}$ space:

Class mean $\boldsymbol{m}_i = \frac{1}{N_i} \sum_{x_j \in \mathcal{X}_i} \boldsymbol{x}_j, \ i = 1,2$

Within-class scatter matrix

$$\boldsymbol{S}_i = \sum_{x_j \in \mathcal{X}_i} (\boldsymbol{x}_j - \boldsymbol{m}_i)(\boldsymbol{x}_j - \boldsymbol{m}_i)^T, \qquad i = 1,2$$

Total within-class scatter matrix $\boldsymbol{S}_w = \boldsymbol{S}_1 + \boldsymbol{S}_2$

Between-class scatter matrix

$$\boldsymbol{S}_b = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T$$

The “closeness” of samples after projection

- In $\mathcal{Y}$ space:

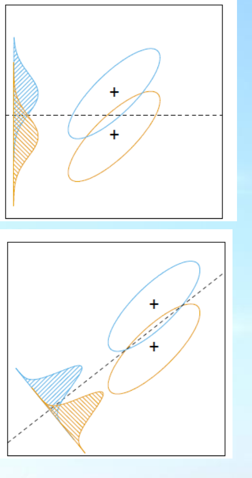
Class mean $\tilde{m}_i = \frac{1}{N_i} \sum_{y_j \in \mathcal{Y}_i} y_j, \ i = 1,2$

Within-class scatter

$$\tilde{S}_i = \sum_{y_j \in \mathcal{Y}_i} (y_j - \tilde{m}_i)(y_j - \tilde{m}_i)^T, \qquad i = 1,2$$

Total within-class scatter $\tilde{S}_w = \tilde{S}_1 + \tilde{S}_2$

Between-class scatter $\tilde{S}_b = (\tilde{m}_1 - \tilde{m}_2)^2$

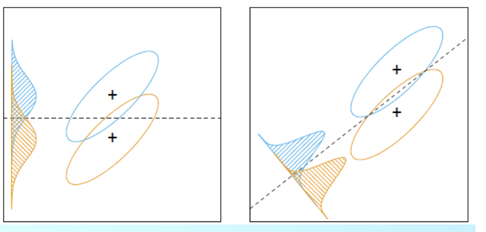


Fisher's Criterion

- Fisher's Criterion (of best separation):

$$\max_{\boldsymbol{w}} J_F(\boldsymbol{w}) = \frac{(\tilde{m}_1 - \tilde{m}_2)^2}{\tilde{S}_1 + \tilde{S}_2}$$

$$y_i = \boldsymbol{w}^T \boldsymbol{x}_i$$



- i.e.,

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}} \; J_F(\boldsymbol{w})$$

$$J_F(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_b \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_w \boldsymbol{w}}$$

Solution for $\max J_F(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_b \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_w \boldsymbol{w}}$?

H

- Not unique: If we change the norm of $\boldsymbol{w}$, $J_F(\boldsymbol{w})$ won't change.
- Fix the denominator $\boldsymbol{w}^T \boldsymbol{S}_w \boldsymbol{w} = c \neq 0$ and maximize the numerator $\boldsymbol{w}^T \boldsymbol{S}_b \boldsymbol{w}$, i.e.

$$\max \; \boldsymbol{w}^T \boldsymbol{S}_b \boldsymbol{w}$$
$$\text{s.t.} \; \boldsymbol{w}^T \boldsymbol{S}_w \boldsymbol{w} = c$$

- Define the Lagrange function (*Lagrangian*)

$$L(\boldsymbol{w}, \lambda) = \boldsymbol{w}^T \boldsymbol{S}_b \boldsymbol{w} - \lambda(\boldsymbol{w}^T \boldsymbol{S}_w \boldsymbol{w} - c)$$

Let $\frac{\partial L}{\partial w} = 0$, we get

$$S_w^{-1} S_b w^* = \lambda w^*$$

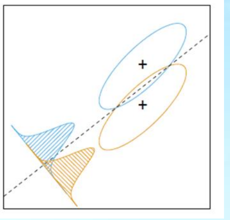
本征向量（特征向量）

i.e., w^* is the *eigenvector* of matrix $S_w^{-1} S_b$

Substitute $S_b = (m_1 - m_2)(m_1 - m_2)^T$ in, we have

$$\lambda \mathbf{w}^* = \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{w}^* \stackrel{\Delta}{=} \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2) R$$

$$R \stackrel{\Delta}{=} (\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{w}^*$$



We only care about the direction, so

$$w^* = S_w^{-1}(m_1 - m_2)$$

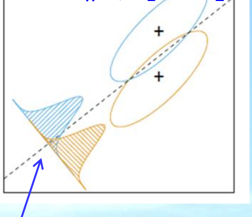
- Are we done for the solution?
- What was our original goal?
  - a "best" linear classifier, not a direction of projection

$$y = \operatorname{sgn}\left(\sum_{i=1}^{n} w_i x_i + w_0\right) = \operatorname{sgn}(\mathbf{w}^T \mathbf{x} + w_0),$$

$$y = \begin{cases} +1 & \Rightarrow \mathbf{x} \in \omega_1 \\ -1 & \Rightarrow \mathbf{x} \in \omega_2 \end{cases}$$

- How shall we choose w_0?

$w^* = S_w^{-1}(m_1 - m_2)$



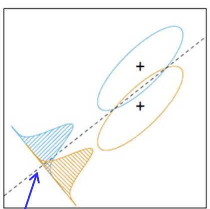
- Commonly used thresholds:

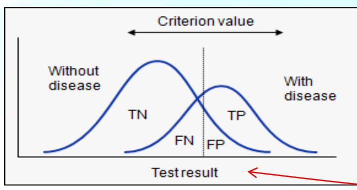
$$w_0 = -\frac{1}{2}(\tilde{m}_1 + \tilde{m}_2)$$

$$w_0 = -\tilde{m}$$

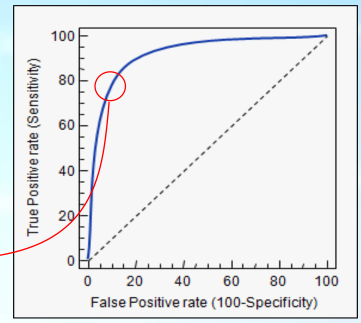
$$w_0 = -\frac{1}{2}(\tilde{m}_1 + \tilde{m}_2) + \frac{1}{N_1 + N_2 - 2}\ln\frac{P(\omega_1)}{P(\omega_2)}$$

- **Choose the threshold with ROC curve**

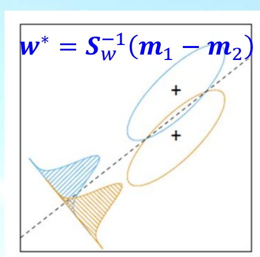


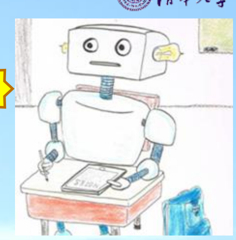


adjusting the threshold



- *“Wait, was that Machine Learning, not mathematics?”*



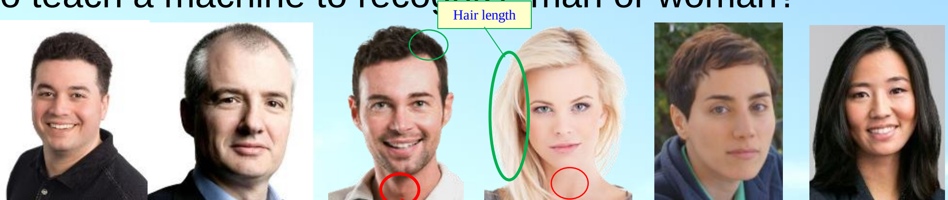


- Learning is the process of acquiring new, or modifying existing, knowledge, behaviors, **skills**, values, or preferences.
- Many popular learning algorithms are iterative, but ML does not have to be iterative.

1-minute stretch break

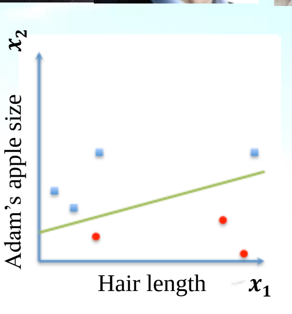
A toy example

- How to teach a machine to recognize man or woman?



The **Adam's apple**, or laryngeal prominence, is a feature of the human neck, and is the lump or protrusion that is formed by the angle of the thyroid cartilage surrounding the larynx seen especially in males.

An iterative learning procedure



For the green line:
$$w_1x_1 + w_2x_2 + w_0 = 0$$
For each blue dots:
$$w_1x_1 + w_2x_2 + w_0 > 0$$
For each red dots:
$$w_1x_1 + w_2x_2 + w_0 < 0$$

3.2
Perceptron:
"the first machine that learns to classify"

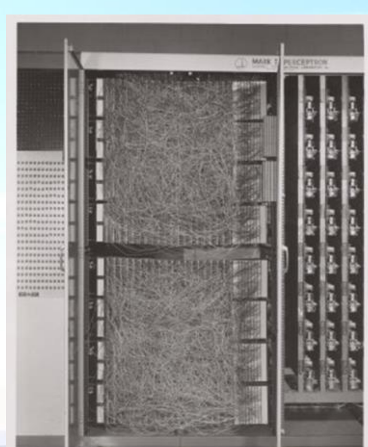
Perceptron （感知器）

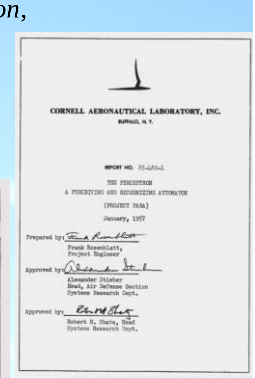
Frank Rosenblatt, *The Perceptron – a perceiving and recognizing automaton*, Report 85-460-1, Cornell Aeronautical Laboratory, Jan. 1957

- The first learning machine: perceptron

$$y = \mathrm{sgn}(\sum_{i=1}^{d} w_i x_i + w_0)$$

- Why was it called a "learning machine"?
  - ① **It was a machine!**




CORNELL AERONAUTICAL LABORATORY, INC.
BUFFALO, N. Y.

REPORT NO. 85-460-1

THE PERCEPTRON
A PERCEIVING AND RECOGNIZING AUTOMATON
(PROJECT PARA)
January, 1957

Prepared by: Frank Rosenblatt,
Project Engineer

Approved by: Alexander Stieber
Head, Air Defense Section
Systems Research Dept.

Approved by: Robert M. Shatz, Head
Systems Research Dept.

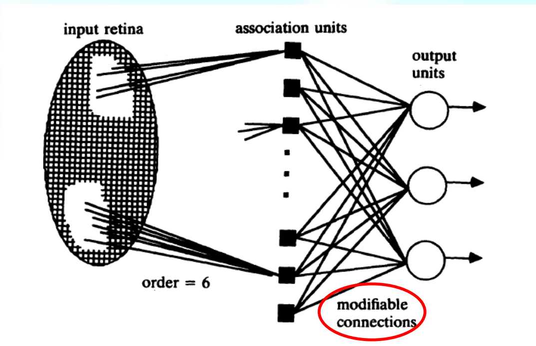
https://en.wikipedia.org/wiki/Perceptron

Perceptron

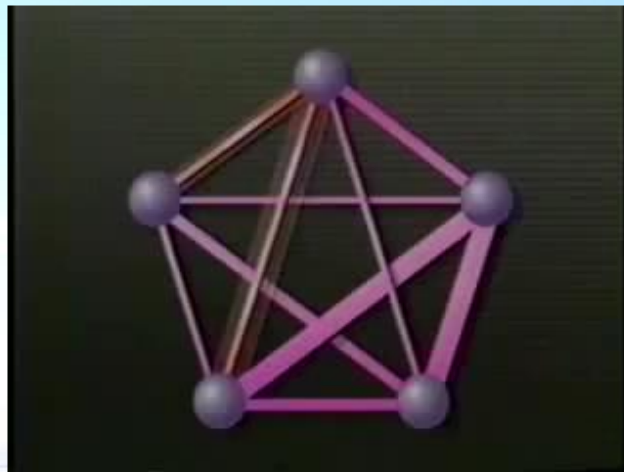
- Why is it a "learning machine"?

$$y = \mathrm{sgn}(\sum_{i=1}^{n} w_i x_i + w_0)$$

  ② **Because it can learn!**
  – It is not hard-coded in a Von-Neumann computing program, but rather, it is modifiable based on training data.



M. Olazaran, A sociological study of the official history of the perceptrons controversy, *Social Studies of Science*, 1996

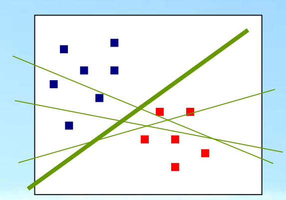


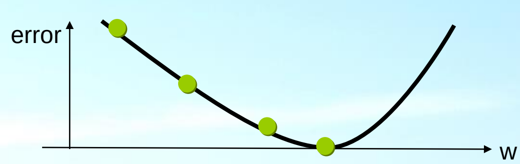
Xuegong Zhang

How Perceptron learns?

- To train the machine $y = \mathrm{sgn}(\sum_{i=1}^{d} w_i x_i + w_0)$ with data $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}, \boldsymbol{x}_i \epsilon R^d$
  – Goal: to optimize an objective function $\boldsymbol{J}(\boldsymbol{w})$
    - The training error (*i.e.*, # of mistakes)
  – Learning algorithm
    - do $\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla J$
    - until $(\nabla J < threshold)$

    —— *Basic Gradient Descent*

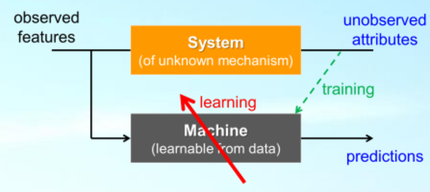




Xuegong Zhang 22

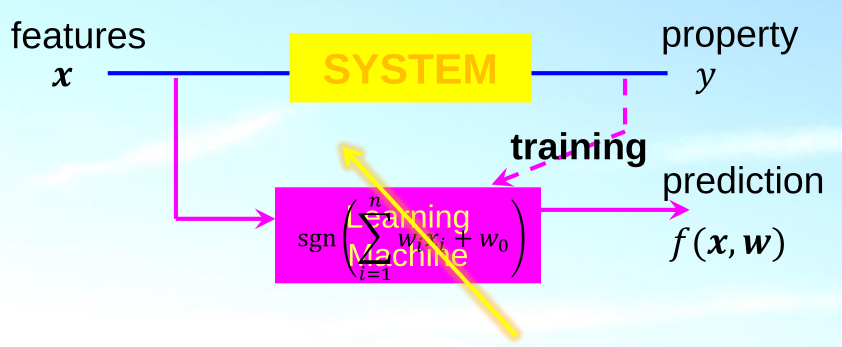
Basic concepts of (supervised) Machine Learning

- How can we make a learning machine?
  - It needs a teacher.
    - → We (people) *design* it (features, the model) and *train* it.
  - We need materials to train it. / It needs materials to learn from.
    - → Training data
  - We need to tell what is the goal of the learning.
    - → Objective function/criterion
  - We need to tell it how to learn.
    - → Learning/training algorithm



---- The basic elements of *Supervised Learning*

- Linear Machines



Linear discriminant function

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \boldsymbol{\alpha}^T\mathbf{y}$$

$\boldsymbol{y} = \begin{bmatrix}1\\ \mathbf{x}\end{bmatrix}$ augmented feature vector

$\boldsymbol{\alpha} = \begin{bmatrix}w_0\\ \boldsymbol{w}\end{bmatrix}$ augmented weight vector

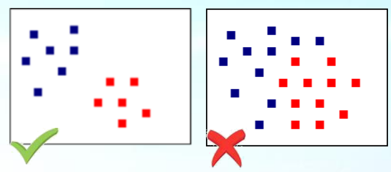
To simplify the treatment, we normalize the function by the following replacement

$$\mathbf{y}_i' = \begin{cases}\mathbf{y}_i, & \text{if } \mathbf{y}_i \in \omega_1, \\ -\mathbf{y}_i, & \text{if } \mathbf{y}_i \in \omega_2,\end{cases} \quad i = 1, \cdots, N$$

then we have $\boldsymbol{\alpha}^T\mathbf{y}_i' > 0, i = 1, \cdots, N$ if all samples are correctly classified. We still use $\boldsymbol{y}_i$ to represent a "*normalized augmented feature vector*" for convenience.

- Linearly separable:

  $\exists\boldsymbol{\alpha}, \quad \boldsymbol{\alpha}^T\mathbf{y}_i > 0, i = 1, \cdots, N$

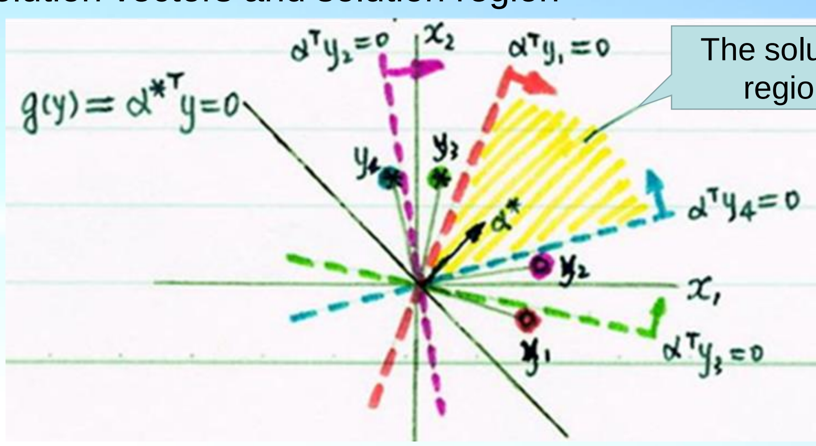


Solution vectors and solution region

- A solution $\boldsymbol{\alpha}^*$: a weight vector that satisfies

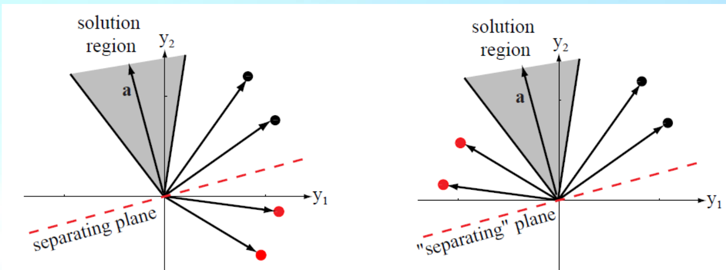
  $\boldsymbol{\alpha}^T\boldsymbol{y}_i > 0,$
  $i = 1, \cdots, N$

- Solution region: the region of all solution vectors in the weight space



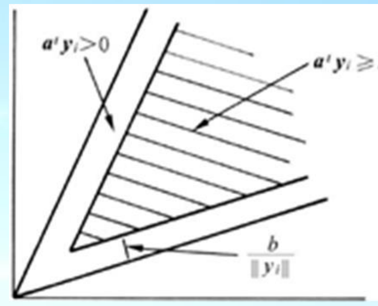
The solution region

Solution vectors and solution region



Duda et al, *Pattern Classification*

- Introducing margins
  $$\boldsymbol{\alpha}^T \boldsymbol{y}_i \geq b > 0$$

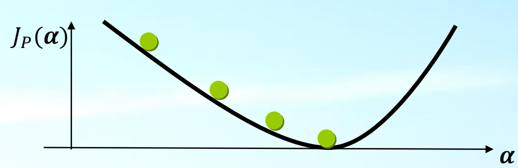


The Perceptron Criterion

$$J_P(\boldsymbol{\alpha}) = \sum_{\mathbf{y}_j \in \mathcal{Y}^k} (-\boldsymbol{\alpha}^T \mathbf{y}_j)$$

$\mathcal{Y}^k$: set of samples misclassified by $\boldsymbol{\alpha}$ at step k

- The Perceptron solution for linearly separable cases
  $$J_P(\boldsymbol{\alpha}^*) = \min J_P(\boldsymbol{\alpha}) = 0$$
- **Gradient descent**:
  $$\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) - \rho_k \nabla J$$
  $$\nabla J = \frac{\partial J_P(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \sum_{\mathbf{y}_j \in \mathrm{Y}^k} (-\mathbf{y}_j)$$
  $$\therefore \quad \boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \rho_k \sum_{\mathbf{y}_j \in \mathrm{Y}^k} \mathbf{y}_j$$



The Perceptron Algorithm

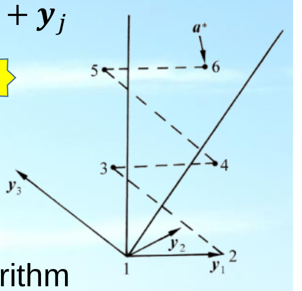
- Fixed increment rule

$$\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \rho_k \sum_{\mathbf{y}_j \in \mathrm{Y}^k} \mathbf{y}_j$$

  ① Initialization (set initial weights to anything, say, zeros)

  ② For sample $\boldsymbol{y}_j$, if $\boldsymbol{\alpha}(k)^T \boldsymbol{y}_j \le 0$ (or b), then $\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \boldsymbol{y}_j$

  ③ Repeat (2) for all samples till $J_P = 0$



- Variable increment rule $\rho_k = \dfrac{|\boldsymbol{\alpha}(k)^T \mathbf{y}_j|}{\|\mathbf{y}_j\|^2}$

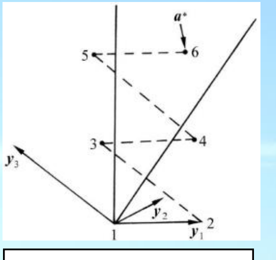
- Perceptron Convergence Theorem:

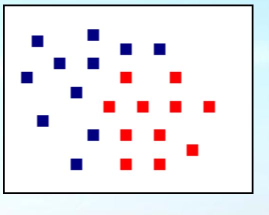
If training samples are linearly separable, the Perceptron Algorithm will **converge** to a solution vector **in a finite number of updates.**

Discussion

$$\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \rho_k \sum_{\mathbf{y}_j \in \mathrm{Y}^k} \mathbf{y}_j$$

- Questions:
  - What if the data are not linearly separable?
  - How can we know the data are linearly separable?
  - Can the machine still find a “useful” solution?
  - Explore the answer with your homework.
- Solutions:
  - Force the learning to stop, e.g., by shrinking the learning rate (step size)
  - Use other linear methods that allow for errors
  - Turn to nonlinear methods





单选题 1分

设置

4-min break

A Yes, I’m back to my computer.

B Sorry, not yet.

提交

清华大学

3.3
Linear Regression

Machines to predict a real value?

e.g., Can it predict the score a student will get based on some observations?

$$\boldsymbol{y} = f(\boldsymbol{x})$$

$\boldsymbol{y}$ → prediction

f → Real function

$\boldsymbol{x}$ → sample

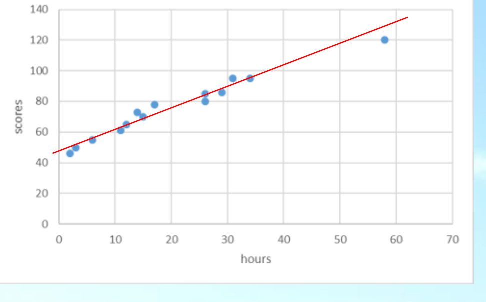
- How shall we make such a machine?
  - What features may predict the score?
    - Study hours, background, IQ, EQ/looking/shape/size, …?
  - What do we need for making such a machine?
    - A function set (the machine)
    - Training data
    - Objective function
    - Algorithm

- How can we make a learning machine?
  - It needs a teacher.
    - → We (people) *design* it (features, the model) and *train* it.
  - We need materials to train it. / It needs materials to learn from.
    - → Training data
  - We need to tell what is the goal of the learning.
    - → Objective function/criterion
  - We need to tell it how to learn.
    - → Learning/training algorithm

---- The basic elements of *Supervised Learning*

Toy data example

Student id	Final score	Study Hours per Week
1	50	3
2	95	34
3	78	17
4	55	6
5	65	12
6	70	15
7	80	26
8	86	29
9	73	14
10	120	58
11	46	2
12	95	31
13	85	26
14	61	11



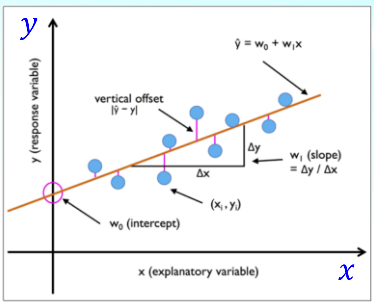
Simple Linear Regression

$$y = w_0 + w_1 x$$

Linear Regression

y
dependent variable
response variable
regressand
output variable
target variable
predication
…

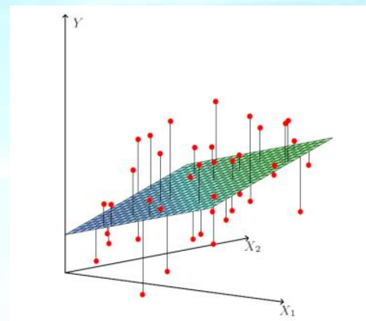
- Simple Linear Regression
  $$y = w_0 + w_1 x$$



Raschka & Mirjalili, *Python Machine Learning*

x
independent variables
explanatory variables
predictor variables
regressors
input variable
features
…

- Multiple Linear Regression
  $$y = w_0 + w_1 x + \cdots + w_d x_d$$
  $$= \sum_{i=0}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$$



Hastie, Tibshirani, Friedman, *Elements of Statistical Learning*

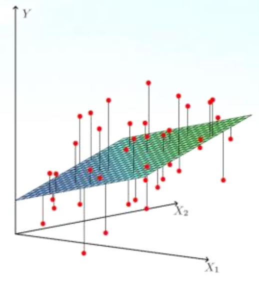
Linear Regression as a Learning Machine

- How can it *learn*?
  - It needs a teacher.
    → We *design* it: $f(\boldsymbol{x}) = \sum_{i=0}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$
  - We need materials to train it. / It needs materials to learn from.
    → Training data: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in R$
  - We need to tell what is the goal of the learning.
    → Objective function: ?
  - We need to tell it how to learn.
    → Learning algorithm: ?

Linear Regression as a Learning Machine

- How can it *learn*?
  - It needs a teacher.
    - → We *design* it: $f(\boldsymbol{x}) = \sum_{i=0}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_j \in R^{d+1}, y_j \in R$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E = \frac{1}{N}\sum_{j=1}^{N}\left(f(\boldsymbol{x}_j) - y_j\right)^2$

      E: mean squared error, sum of squares, in-sample error, …
  - We need to tell it how to learn.
    - → Learning algorithm: ?



Linear Regression Algorithm

$$\min_{\boldsymbol{w}} E(\boldsymbol{w}) = \frac{1}{N}\sum_{j=1}^{N}\left(f(\boldsymbol{x}_j) - y_j\right)^2 = \frac{1}{N}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

where $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$.

Solution:

Let $\nabla E(\boldsymbol{w}) = \frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{2}{N}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0$,

we have $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$.

Therefore, $\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, if $(\boldsymbol{X}^T\boldsymbol{X})$ is invertible.

R

Pseudo-inverse: $\boldsymbol{X}^+ = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$

$$\Big[\,[\quad]\,[\quad]\,\Big]^{-1}[\quad]$$

dim: $(d+1) \times N$ $\quad N \times (d+1)$ $\quad (d+1) \times N$

Linear regression algorithm:

1: Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$, where each $\mathbf{x}$ includes the $x_0 = 1$ bias coordinate, as follows

$$\mathrm{X} = \underbrace{\begin{bmatrix} \text{—}\mathbf{x}_1^{\mathrm{T}}\text{—} \\ \text{—}\mathbf{x}_2^{\mathrm{T}}\text{—} \\ \vdots \\ \text{—}\mathbf{x}_N^{\mathrm{T}}\text{—} \end{bmatrix}}_{\text{input data matrix}}, \qquad \mathbf{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

2: Compute the pseudo-inverse $\mathrm{X}^\dagger$ of the matrix X. If $\mathrm{X}^{\mathrm{T}}\mathrm{X}$ is invertible,

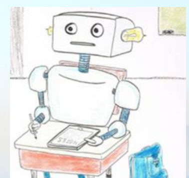
$$\mathrm{X}^\dagger = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}.$$

3: Return $\mathbf{w}_{\text{lin}} = \mathrm{X}^\dagger \mathbf{y}$.

Abu-Mostafa, Magdon-Ismail, Lin, *Learning from Data*

---- Ordinary least squares (OLS) algorithm

- Is this “learning” or just an analytic solution?
  - Well, LS methods was published by Legendre in 1805 and by Gauss in 1809, long before the idea of “machine learning” was born.
  - As long as it gets a model from the data, we are happy to call it “learning”.

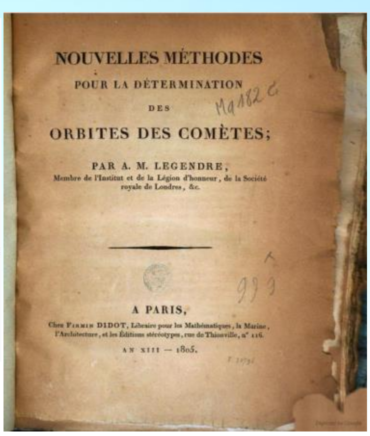


A.M. Legendre. Nouvelles méthodes pour la détermination des orbites des comètes, Firmin Didot, Paris, 1805. “Sur la Méthode des moindres quarrés” appears as an appendix.

Adrien-Marie Legendre

1820 watercolor caricature of Adrien-Marie Legendre by French artist Julien-Leopold Boilly (see portrait debacle), the only existing portrait known[1]

Born	18 September 1752 Paris, France
Died	10 January 1833 (aged 80) Paris, France



NOUVELLES MÉTHODES
POUR LA DÉTERMINATION
DES
ORBITES DES COMÈTES;
PAR A. M. LEGENDRE,
Membre de l'Institut et de la Légion d'honneur, de la Société royale de Londres, &c.

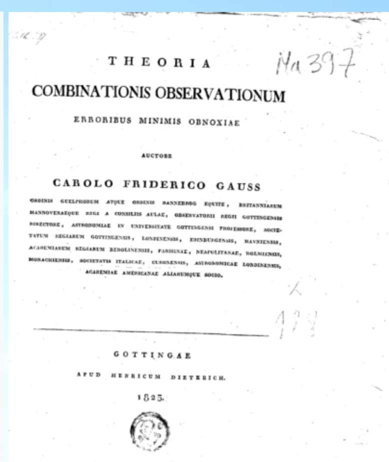
A PARIS,
Chez Firmin Didot, Libraire pour les Mathématiques, la Marine, l'Architecture, et les Éditions stéréotypes, rue de Thionville, n° 116.
AN XIII — 1805.

C.F. Gauss. Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientum. (1809)

Carl Friedrich Gauss

Carl Friedrich Gauß (1777–1855), painted by Christian Albrecht Jensen

Born	Johann Carl Friedrich Gauss 30 April 1777 Brunswick, Principality of Brunswick-Wolfenbüttel
Died	23 February 1855 (aged 77) Göttingen, Kingdom of Hanover, German Confederation



THEORIA
COMBINATIONIS OBSERVATIONUM
ERRORIBUS MINIMIS OBNOXIAE
AUCTORE
CAROLO FRIDERICO GAUSS

GOTTINGAE
APUD HENRICUM DIETERICH.
1823.

- Wikipedia

Any question?

“ $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, if $(\boldsymbol{X}^T\boldsymbol{X})$ is invertible.”

- What is “invertible”? What happens if it is not?
  - Invertible (nonsingular, nondegenerate, full rank)

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}_{N\times(d+1)}$$

  - $(\boldsymbol{X}^T\boldsymbol{X})$ invertible: $\boldsymbol{X}$ full column rank : features linearly independent
  - Usually true when $N \gg d+1$
  - When features not linearly independent, pseudo-inverse can still be defined, but solution is not unique.
  - Solutions:
    - Remove redundancy by feature selection or transformation
    - Introduce other criteria (e.g., SVD or regularization) to restrict the solution

Recall: Similar assumption in FLD

- Fisher’s criterion: $\max \quad J_F(\boldsymbol{w}) = \frac{\boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w}}$

Solution:

$$\max \boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w}$$
$$s.t.\ \boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w} = c$$

Lagrange function

$$L(\boldsymbol{w},\lambda) = \boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w} - \lambda(\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w} - c)$$

Let $\frac{\partial L}{\partial w} = 0$, we get $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b\boldsymbol{w}^* = \lambda\boldsymbol{w}^*$

and

$$\boldsymbol{w}^* = \boldsymbol{S}_w^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

How to evaluate the regression?

Evaluation of regression models

$\boldsymbol{R^2}$: the ***goodness-of-fit***, the ***coefficient of determination***, …

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

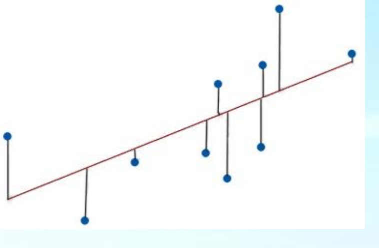
Unexplained variation

Total variation

For OLS,

$$R^2 = \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

and $0 \leq R^2 \leq 1$.



Evaluation of regression models

$\boldsymbol{R^2}$: the ***goodness-of-fit***, the ***coefficient of determination***, …

R

- For OLS, $0 \le R^2 \le 1$.
  - $R^2 = 1$: Perfect regression.
  - $R^2 = 0$: Baseline model. Predictions are the average.
- For other types of regression as a general measure of goodness-of-fit

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \in [-\infty, 1]$$

and should no longer be called R^2.

Xuegong Zhang 45

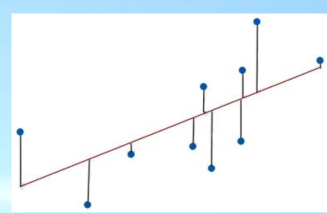
R^2 is not enough for evaluating regression

$$y_i = w_0 + \boldsymbol{w}^T\boldsymbol{x}_i + \epsilon_i, \qquad i = 1, \cdots, N$$

Dependent variable

Fitted values (deterministic)

Error, residual, noise (stochastic)



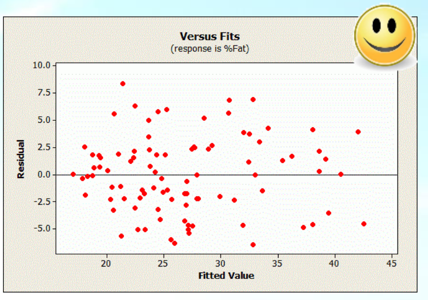
- R^2: percentage of dependent variable variations that the linear model explains
- R^2 does not indicate if the regression model provides an adequate fit to the data

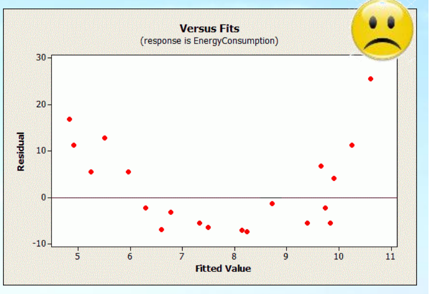
Xuegong Zhang 46

R^2 is not enough for evaluating regression

$$y_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}_i + \epsilon_i, \qquad i = 1, \cdots, N$$

- R^2: percentage of dependent variable variations that the linear model explains
- R^2 does not indicate if the regression model provides an adequate fit to the data
- ***Residual plots***
  - To check if model is adequate
  - Poor fitting if error/ residual not random





Evaluating each coefficient

$$y_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}_i + \epsilon_i = w_0 + \sum_{j=1}^{d} w_j x_{ij} + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2), i = 1, \cdots, N,$$

Does each w_j contribute?

Test for statistical significance of regression coefficients

$$\frac{\hat{w}_j - w_j}{s_{\hat{w}_j}} \sim t_{N-d-1}, \qquad j = 0, 1, \cdots, d.$$

1-minute break

3.4
Minimum Squared Error (MSE) Method

Recap on Perceptron

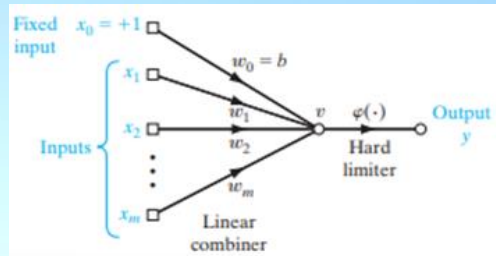
- Objective function: classification error

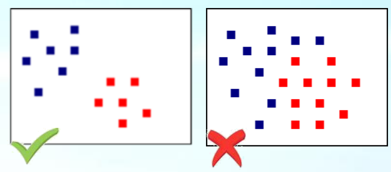
$$J_P(\boldsymbol{\alpha}) = \sum_{\mathbf{y}_j \in Y^k} (-\boldsymbol{\alpha}^T \mathbf{y}_j)$$

- The Perceptron Criterion:

$$J_P(\boldsymbol{\alpha}^*) = \min J_P(\boldsymbol{\alpha}) = 0$$

- The learning algorithm:
  - Gradient descent: $\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) - \rho_k \nabla J$
- Major limitation:
  - Designed for linearly separable cases

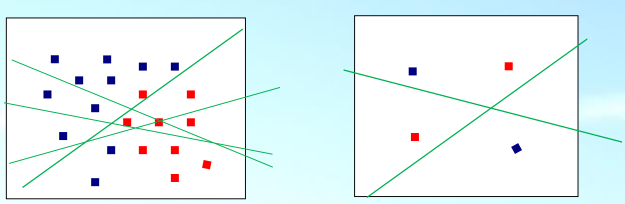




Linearly non-separable cases

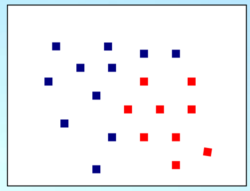
- How about the following cases?

Not perfectly separable, but can achieve reasonable separation with a linear classifier.

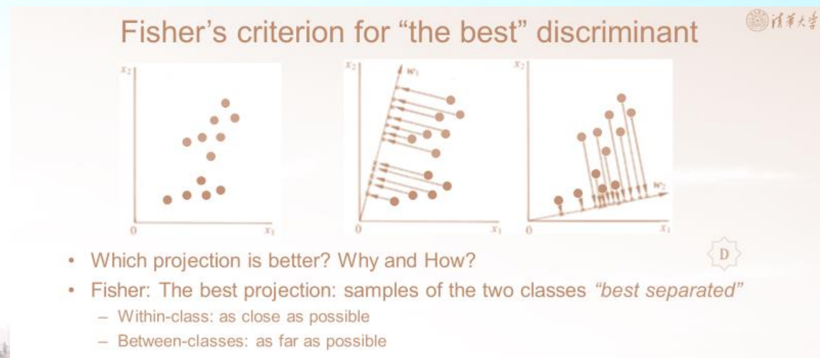


No reasonable linear solution.

- Can we get a "best" linear classifier when there is no perfect one?



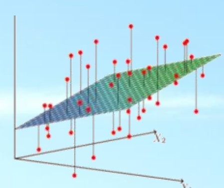
- What is "the best"?
  - A basic principle that differentiates many different methods.



Minimum Squared Error (MSE) method

- How do we calculate and minimize "errors"?
- We knew how to minimize mean squared error for linear regression

$$\min E = \frac{1}{N}\sum_{j=1}^{N}\left(f(\boldsymbol{x}_j) - y_j\right)^2$$



- Our goal was to find the $\boldsymbol{w}$ that $y = \boldsymbol{w}^T\boldsymbol{x}$ for all samples
- Now our goal is to find the $\boldsymbol{w}$ that $\boldsymbol{\alpha}^T\boldsymbol{y} > 0$ for all samples

---- well, for as many samples as possible.

- How about to define a $b_j > 0$ for each sample so that

$$\boldsymbol{\alpha}^T\boldsymbol{y}_j = b_j, \quad j = 1, \cdots, N$$

for as many sample as possible?

Resuming the setting of perceptron

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \boldsymbol{\alpha}^T\mathbf{y}$$

$\boldsymbol{y} = \begin{bmatrix}1\\ \mathbf{x}\end{bmatrix}$ augmented feature vector

$\boldsymbol{\alpha} = \begin{bmatrix}w_0\\ \boldsymbol{w}\end{bmatrix}$ augmented weight vector

To simplify the treatment, we normalize the function by the following replacement

$$\mathbf{y}_i' = \begin{cases}\mathbf{y}_i, & \text{if } \mathbf{y}_i \in \omega_1, \\ -\mathbf{y}_i, & \text{if } \mathbf{y}_i \in \omega_2,\end{cases} \quad i = 1, \cdots, N$$

then we have $\boldsymbol{\alpha}^T\mathbf{y}_i' > 0, i = 1, \cdots, N$ if all samples are correctly classified. We still use $\boldsymbol{y}_i$ to represent a "*normalized augmented feature vector*" for convenience.

The MSE Criterion

$$\boldsymbol{\alpha}^T\mathbf{y}_i > 0 \quad \Longleftrightarrow \quad \boldsymbol{\alpha}^T\mathbf{y}_i = b_i > 0, i = 1, \cdots, N$$

- From inequation to equation

$$\mathbf{Y}\boldsymbol{\alpha} = \boldsymbol{b}, \qquad \boldsymbol{b} = [b_1, b_2, \cdots, b_N]^T$$

- The MSE Criterion

$$\alpha^*: \quad \min_{\boldsymbol{\alpha}} J_S(\boldsymbol{\alpha})$$

$$J_S(\boldsymbol{\alpha}) = \|\mathbf{Y}\boldsymbol{\alpha} - \mathbf{b}\|^2 = \sum_{i=1}^{N}(\boldsymbol{\alpha}^T\mathbf{y}_i - b_i)^2$$

The MSE Criterion

$$\min \quad J_S(\boldsymbol{\alpha}) = \|\mathbf{Y}\boldsymbol{\alpha} - \mathbf{b}\|^2 = \sum_{i=1}^{N} (\boldsymbol{\alpha}^T \mathbf{y}_i - b_i)^2$$

- Solutions
  - Pseudo-Inverse

$$\boldsymbol{\alpha}^* = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{b} = \mathbf{Y}^+\mathbf{b},$$
$$\mathbf{Y}^+ = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T$$

  - Gradient Descent

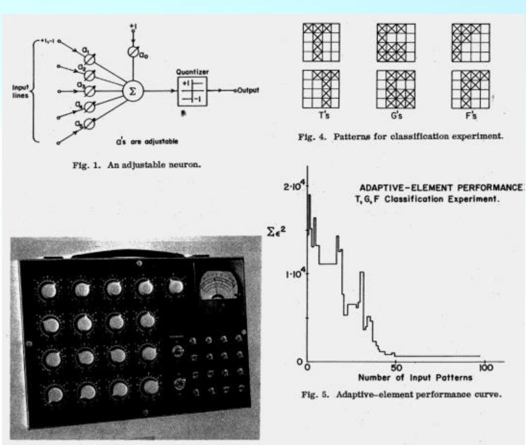
$$\nabla J_s(\boldsymbol{\alpha}) = 2\mathbf{Y}^T(\mathbf{Y}\boldsymbol{\alpha} - \mathbf{b})$$
$$\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \rho_k(b_k - \boldsymbol{\alpha}(k)^T\mathbf{y}^k)\mathbf{y}^k$$

--- Widrow-Hoff Algorithm
or the ADALINE

R

ADALINE

Widrow & Hoff, Adaptive switching circuits, *1960 IRE Western Electric Show and Convention Record, Part 4*, pp.96-104, Aug, 1960



Widrow & Hoff, Adaptive switching circuits, *1960 IRE Western Electric Show and Convention Record, Part 4*, pp.96-104, Aug, 1960

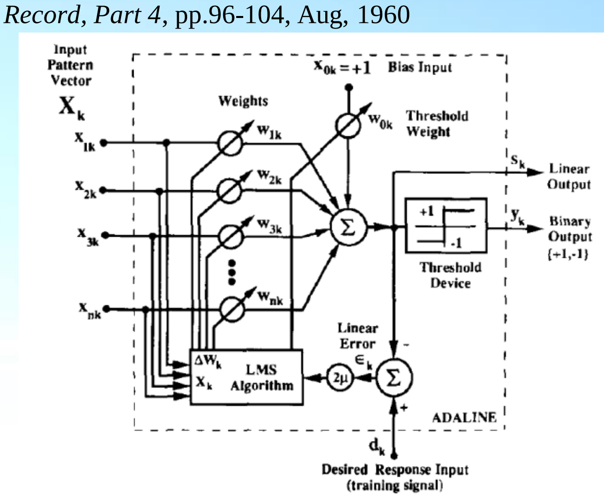


Fig. 2. Adaptive linear element (Adaline).

Widrow & Lehr, 30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation, *Proceedings of the IEEE*, 78(9): 1415-1442, 1990

ADALINE / Widrow-Hoff Algorithm / LMS Algorithm

Algorithm: Widrow-Hoff (ADALINE)

① Normalize the augmented feature vectors $\boldsymbol{z}_j$ of all training samples by

$$\boldsymbol{z}_i' = \begin{cases} \boldsymbol{z}_i, & \text{if } \boldsymbol{z}_i \in \omega_1, \\ -\boldsymbol{z}_i, & \text{if } \boldsymbol{z}_i \in \omega_2, \end{cases} \qquad i = 1, \cdots, N$$

② Initialization: Set $k = 0$; Set initial weights to all zeros $\boldsymbol{\alpha}(0) = \mathbf{0}$; Set proper target values b_i for all samples;

③ Pick up a sample $\boldsymbol{z}_j$ from the training set, compute the gradient and update the weight

$$\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \rho_k\left(b_j - \boldsymbol{\alpha}(k)^T \boldsymbol{z}_j\right)\boldsymbol{z}_j .$$

④ Let $k = k + 1$. Repeat ② for all samples till the stopping criterion is met.

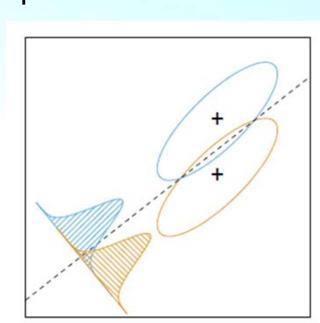
Any questions?

How do we assign the $\boldsymbol{b}$?

- It can be proven that if we assign the $\boldsymbol{b}$ as

$$b_i = \begin{cases} N/N_1, & if\ \boldsymbol{y}_i \epsilon \omega_1 \\ N/N_2, & if\ \boldsymbol{y}_i \epsilon \omega_2 \end{cases},$$

then the MSE solution is equal to the FLD solution with $w_0 = \hat{m}$.



The "best"?

- It can also be proven that if we assign the $\boldsymbol{b}$ as

$$b_i = 1,\ i = 1, \cdots, N\ ,$$

then when $N \to \infty$, the MSE solution best approximates the *Bayesian Discriminant*

$$g_0(\boldsymbol{x}) = P(\omega_1|\boldsymbol{x}) - P(\omega_2|\boldsymbol{x})$$

with minimal square error $e^2 = \int [\boldsymbol{a}^T\boldsymbol{y} - g_0(\boldsymbol{x})]^2 p(\boldsymbol{x}) d(\boldsymbol{x})$.

the optimal classifier when class distribution densities are known

Let's leave it for future classes.

单选题 1分 设置

4-min break

A Yes, I'm back to my computer.

B Sorry, not yet.

提交

3.5
Logistic Regression

Regression for binary outcome?

$$\boldsymbol{y} = f(\boldsymbol{x})$$

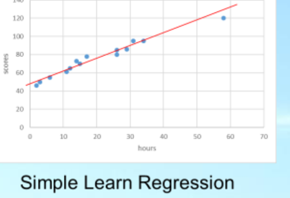
prediction ↑ real function ↑ sample ↑

$$X = [\mathbf{x}_1, \cdots, \mathbf{x}_m] = \begin{bmatrix} x_{11} & \cdots & x_{m1} \\ \vdots & \ddots & \vdots \\ x_{1n} & \cdots & x_{mn} \end{bmatrix}$$

← features

A sample ↑

Student id	Final score	Study Hours per Week
1	50	3
2	95	34
3	78	17
4	55	6
5	65	12
6	70	15
7	80	26
8	86	29
9	73	14
10	120	58
11	46	2
12	95	31
13	85	26
14	61	11



Simple Learn Regression
$y = w_0 + w_1 x$

Regression for binary outcome?

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$$

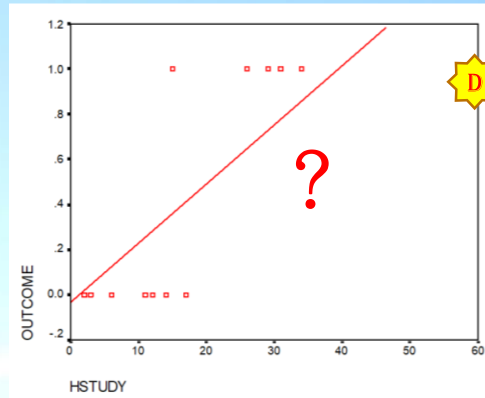
prediction → binary function → sample →

$$X = [\mathbf{x}_1, \cdots, \mathbf{x}_m] = \begin{bmatrix} x_{11} & \cdots & x_{m1} \\ \vdots & \ddots & \vdots \\ x_{1n} & \cdots & x_{mn} \end{bmatrix}$$

← features

A sample →

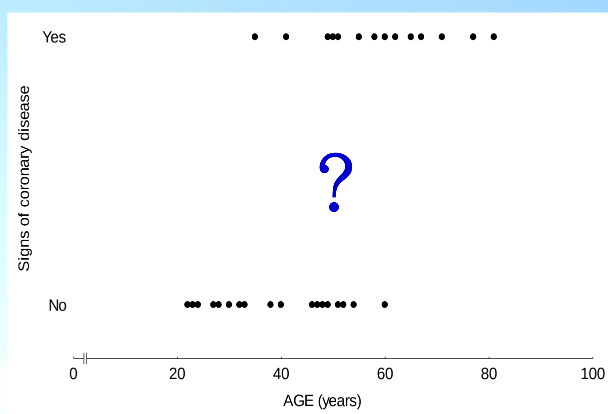
Student id	Final score	Study Hours per Week
1	0	3
2	1	34
3	0	17
4	0	6
5	0	12
6	1	15
7	1	26
8	1	29
9	0	14
10	1	58
11	0	2
12	1	31
13	1	26
14	0	11



Another example

Age and signs of coronary heart disease (CD)

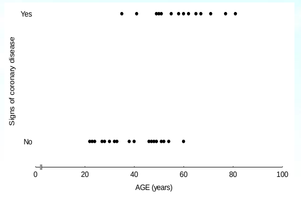
Age	CD	Age	CD	Age	CD
22	0	40	0	54	0
23	0	41	1	55	1
24	0	46	0	58	1
27	0	47	0	60	1
28	0	48	0	60	0
30	0	49	1	62	1
30	0	49	0	65	1
32	0	50	1	67	1
33	0	51	0	71	1
35	1	51	1	77	1
38	0	52	0	81	1



Let’s count the percentage

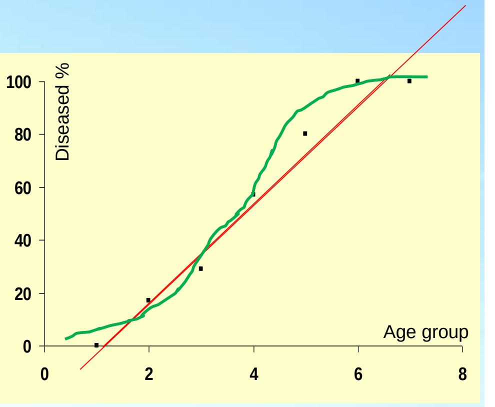
Age and signs of coronary heart disease (CD)

Age	CD	Age	CD	Age	CD
22	0	40	0	54	0
23	0	41	1	55	1
24	0	46	0	58	1
27	0	47	0	60	1
28	0	48	0	60	0
30	0	49	1	62	1
30	0	49	0	65	1
32	0	50	1	67	1
33	0	51	0	71	1
35	1	51	1	77	1
38	0	52	0	81	1

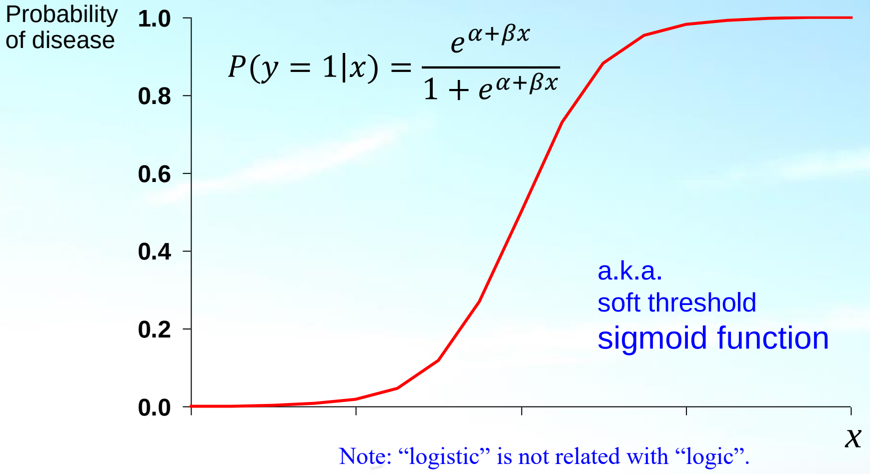


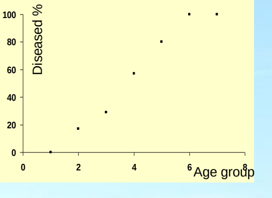
Prevalence (%) of signs of CD according to age group

Age group	# in group	Diseased #	Diseased %
20 - 29	5	0	0
30 - 39	6	1	17
40 - 49	7	2	29
50 - 59	7	4	57
60 - 69	5	4	80
70 - 79	2	2	100
80 - 89	1	1	100



Logistic function

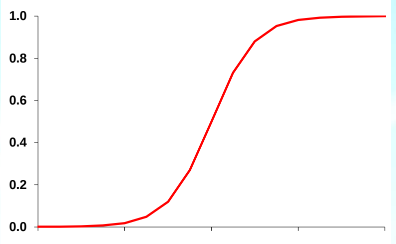




a.k.a.
soft threshold
sigmoid function

Note: “logistic” is not related with “logic”.
So I prefer the translation 罗杰斯特 or 罗杰斯蒂 .

Logistic regression



$$P(y|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

Inverse function

$$\ln\left[\frac{P(y|x)}{1-P(y|x)}\right] = \alpha + \beta x$$

logit of $P(y|x)$

---- *log odds* 对数几率

Multiple logistic regression

$$P = \frac{e^{w_0+w_1x_1+w_2x_2+\cdots+w_px_p}}{1+e^{w_0+w_1x_1+w_2x_2+\cdots+w_px_p}} = \frac{e^{\boldsymbol{w}^T\boldsymbol{x}}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}}} \triangleq \theta(\boldsymbol{w}^T\boldsymbol{x})$$

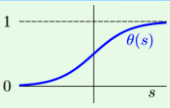
$$odds = \frac{P}{1-P} = e^{w_0+w_1x_1+w_2x_2+\cdots+w_px_p} = e^{\boldsymbol{w}^T\boldsymbol{x}}$$

$$\ln\left(\frac{P}{1-P}\right) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_px_p = \boldsymbol{w}^T\boldsymbol{x}$$

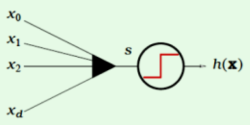
- Interpretation of w_i
  - Increase in log-odds for a one unit increase in x_i with all other x_i's constant
  - Measures association between x_i and log-odds adjusted for all other x_i

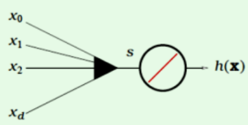
Three types of linear machines

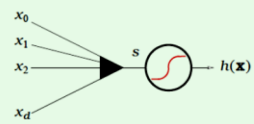
$$s = \sum_{i=0}^{d} w_i x_i \qquad \theta(s) = \frac{e^s}{1+e^s}$$



linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \mathrm{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$



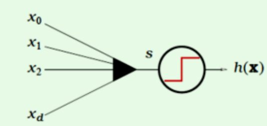




Abu-Mostafa, Magdon-Ismail, Lin, *Learning from Data, Lecture 9*

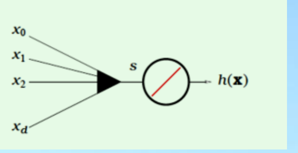
$h(\boldsymbol{x}) = \theta(s)$ is the estimate of the probability of $y = 1$.
$s = \boldsymbol{w}^T\boldsymbol{x}$ is the signal of the event ("risk score")

Basic concepts of ML: Perceptron



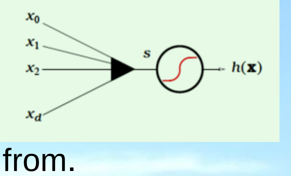
- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $y = \mathrm{sgn}(\sum_{i=1}^{d} w_i x_i + w_0)$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min J_P(\boldsymbol{\alpha}) = \sum_{\mathbf{y}_j \in \mathcal{Y}^k} (-\boldsymbol{\alpha}^T \mathbf{y}_j)$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) - \rho_k \nabla J = \boldsymbol{\alpha}(k) + \rho_k \sum_{\mathbf{y}_j \in \mathrm{Y}^k} \mathbf{y}_j$

Basic concepts of ML: Linear Regression



- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $f(\boldsymbol{x}) = \sum_{i=0}^{d} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in R$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E = \frac{1}{N}\sum_{j=1}^{N}\left(f(\boldsymbol{x}_j) - y_j\right)^2$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$

Basic concepts of ML: Logistic Regression



- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $h(\boldsymbol{x}) = \theta(\boldsymbol{w}^T \boldsymbol{x})$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$
  - We need to tell what is the goal of the learning.
    - → Objective function: ?
  - We need to tell it how to learn.
    - → Learning algorithm: ?

Reasoning behind logistic regression

- Data $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$ were generated i.i.d. by the probability $f(\boldsymbol{x})$:

$$P(y|\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & for\ y = +1 \\ 1 - f(\boldsymbol{x}) & for\ y = -1 \end{cases}$$

A generative model

- We use $h(\boldsymbol{x}) = \theta(\boldsymbol{w}^T\boldsymbol{x})$ to estimate $f(\boldsymbol{x})$.
- How to estimate the parameters of $h(\boldsymbol{x}) = \theta(\boldsymbol{w}^T\boldsymbol{x})$?

Reasoning behind logistic regression

- **Likelihood**（n. 似然度）：
  - For instance $(\boldsymbol{x}_j, y_j)$ of data, if $h = f$, how likely we get y_j from $\boldsymbol{x}_j$?

$$P(y_j|\boldsymbol{x}_j) = \begin{cases} h(\boldsymbol{x}_j) & for\ y_j = +1 \\ 1 - h(\boldsymbol{x}_j) & for\ y_j = -1 \end{cases}$$

  - In other words, now we have this instance of data, how likely h is the "model" that generated the data?
- Noting $\theta(-s) = 1 - \theta(s)$, we have the likelihood on $\boldsymbol{x}_j$:

$$P(y_j|\boldsymbol{x}_j) = \theta(y_j\boldsymbol{w}^T\boldsymbol{x}_j)$$

单选题 10分

设置

The likelihood $P(y_j|\boldsymbol{x}_j) = \theta(y_j \boldsymbol{w}^T \boldsymbol{x}_j)$ is a function of whom?

- A of the output y_i
- B of the weights $\boldsymbol{w}$
- C of the features $\boldsymbol{x}_i$
- D of the sample $(\boldsymbol{x}_j, y_j)$

Xuegong Zhang

提交

77

Maximizing the likelihood

- Likelihood of getting the i.i.d. data $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$, $\boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$ from the model (likelihood of the model on the data):

$P(y_j|\boldsymbol{x}_j) = \theta(y_j \boldsymbol{w}^T \boldsymbol{x}_j)$

$$L(\boldsymbol{w}) = \prod_{j=1}^{N} P(y_j|\boldsymbol{x}_j) = \prod_{j=1}^{N} \theta(y_j \boldsymbol{w}^T \boldsymbol{x}_j)$$

R

- Likelihood Maximization:

$$\min \quad E(\boldsymbol{w}) = -\frac{1}{N} \ln(L(\boldsymbol{w})) = -\frac{1}{N} \ln\left(\prod_{j=1}^{N} \theta(y_j \boldsymbol{w}^T \boldsymbol{x}_j)\right)$$

Xuegong Zhang 78

Maximizing the likelihood

- Likelihood Maximization:

$$\min \quad E(\boldsymbol{w}) = -\frac{1}{N}\ln\big(L(\boldsymbol{w})\big) = -\frac{1}{N}\ln\left(\prod_{j=1}^{N}\theta(y_j\boldsymbol{w}^T\boldsymbol{x}_j)\right)$$

$$= \frac{1}{N}\sum_{j=1}^{N}\ln\left(\frac{1}{\theta(y_j\boldsymbol{w}^T\boldsymbol{x}_j)}\right) = \frac{1}{N}\sum_{j=1}^{N}\ln\left(1+e^{-y_j\boldsymbol{w}^T\boldsymbol{x}_j}\right)$$

R

$$\left[\theta(s) = \frac{1}{1+e^{-s}}\right]$$

- Gradient:

$$\nabla E = -\frac{1}{N}\sum_{j=1}^{N}\frac{y_j\boldsymbol{x}_j}{1+e^{y_j\boldsymbol{w}(k)^T\boldsymbol{x}_j}}$$

Logistic Regression Algorithm

1. Set $k=0$, initialize $\boldsymbol{w}(0)$
2. Do
   - Compute the gradient $\nabla E = -\frac{1}{N}\sum_{j=1}^{N}\frac{y_j\boldsymbol{x}_j}{1+e^{y_j\boldsymbol{w}(k)^T\boldsymbol{x}_j}}$
   - Update the weights $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta\nabla E$, set $k=k+1$

   Until the stopping criterion met
3. Return the final weights $\boldsymbol{w}$

- Initialization:

  All zeros work, but safer to initialize weights randomly, say, normal distribution with 0-mean and small variance
- Termination:

  An upper bound on iterations, or a threshold of the gradient

H R

Abu-Mostafa, Magdon-Ismail, Lin, *Learning from Data, Lecture 9*

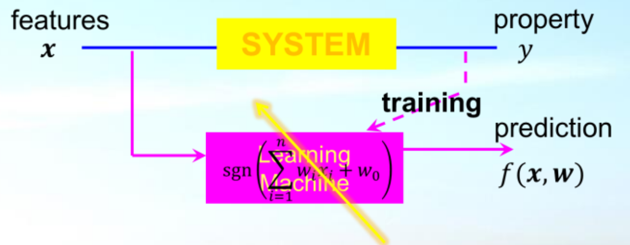
Basic concepts of ML: Logistic Regression

- How can we make a learning machine?
  - It needs a teacher.
    - → The model: $h(\boldsymbol{x}) = \theta(\boldsymbol{w}^T\boldsymbol{x})$
  - We need materials to train it. / It needs materials to learn from.
    - → Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\},\ \boldsymbol{x}_j \in R^{d+1}, y_j \in \{-1,1\}$
  - We need to tell what is the goal of the learning.
    - → Objective function: $\min E(\boldsymbol{w}) = \frac{1}{N}\sum_{j=1}^{N} \ln\left(1 + e^{-y_j \boldsymbol{w}^T \boldsymbol{x}_j}\right)$
  - We need to tell it how to learn.
    - → Learning algorithm: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \rho_k \nabla E$

3.6 Discussion

The basic elements of supervised learning

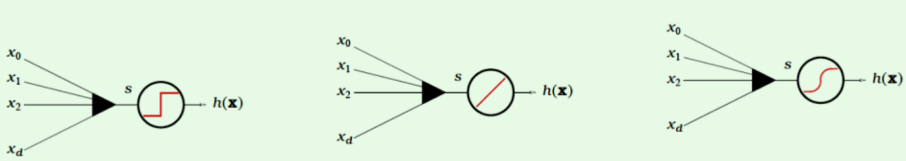
- It needs a teacher.
  - → We (people) *design* it (features, the model) and *train* it.
- We need materials to train it. / It needs materials to learn from.
  - → Training data
- We need to tell what is the goal of the learning.
  - → Objective function/criterion
- We need to tell it how to learn.
  - → Learning/training algorithm



Three major types of linear machines

$$s = \sum_{i=0}^{d} w_i x_i$$

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \mathrm{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$



Abu-Mostafa, Magdon-Ismail, Lin, *Learning from Data, Lecture 9*

Optimization Problems

- For perceptron

$$\min \quad J_P(\boldsymbol{\alpha}) = \sum_{y_j \in \mathrm{Y}^k} (-\boldsymbol{\alpha}^T \boldsymbol{y}_j)$$

Iterative learning

- For linear regression

$$\min \quad E(\boldsymbol{w}) = \frac{1}{N}\sum_{j=1}^{N}\left(\boldsymbol{w}^T \boldsymbol{x}_j - y_j\right)^2$$

Closed-form solution or iterative learning

- For logistic regression

$$\min \quad E(\boldsymbol{w}) = \frac{1}{N}\sum_{j=1}^{N}\ln\left(1 + e^{-y_j \boldsymbol{w}^T \boldsymbol{x}_j}\right)$$

Iterative learning

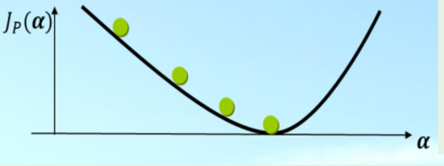
Optimization via Gradient Descent

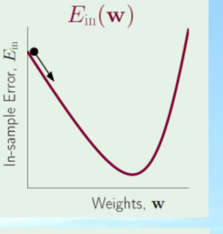
- General method of gradient descent:

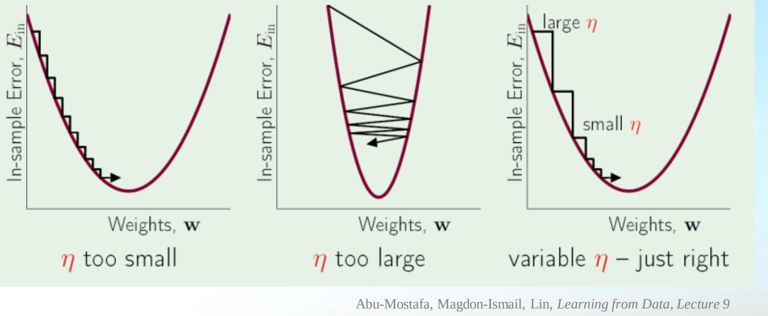
$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta\hat{\boldsymbol{v}}$$

$$\hat{\boldsymbol{v}} = -\nabla E(\boldsymbol{w}(k))$$

- η: learning rate (step size)







Abu-Mostafa, Magdon-Ismail, Lin, *Learning from Data, Lecture 9*

Discussions

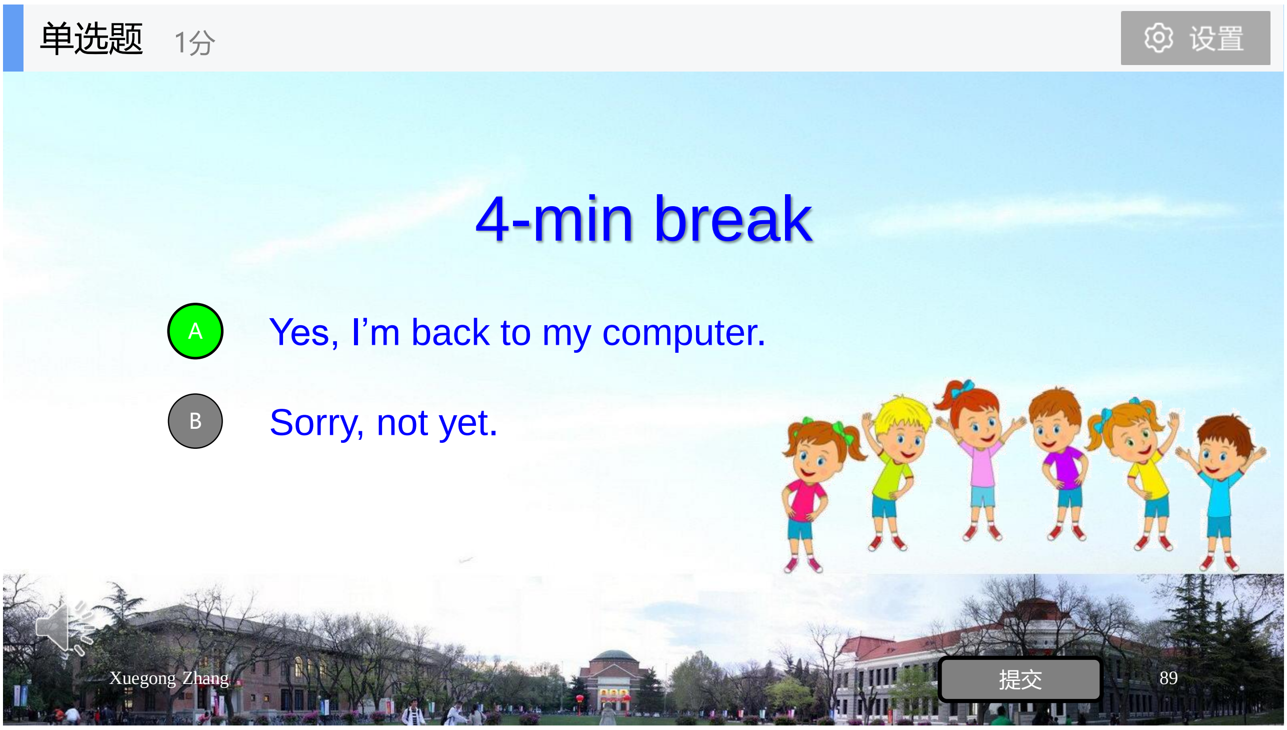
- Machines that can learn
  - Different types of machines for different tasks
  - Different assumptions for the same task
  - Shared general principles and techniques
  - One-size-fit-all solution?
  - Common theory for all machines?

Homework

- Problems (Pr2)
  1. (Op) Inference on smoking-health
  2. FLD and MSE
  3. Perceptron convergence
  4. Logistic function properties
- Deadline:
  - Sept. 29, 23:00 Beijing Time

D

- Computer exercises (Ex1)
  - Coding FLD, Perceptron and Logistic Regression in Python
  - Experimenting on a medical dataset
- Deadline:
  - Oct. 6, 23:00 Beijing Time


单选题 1分
设置
4-min break
A
Yes, I’m back to my computer.
B
Sorry, not yet.
Xuegong Zhang
提交
89