Lect4: Numerical Attributes & Linear Regression, etc.

Tokyo Tech.
Intro. to Comp. & Data
Lecture week4

1. How to treat numerical attributes.
2. Basic methods for numerical data prediction.

1. Normal distributions.
2. How to take care of numerical attributes.
3. Linear methods (for regression and classification).

4 On Exercise #4.

* Although the term "numeric attribute" is used in the textbook, I would like to use "numerical attribute" in this course.

* Some of the slide materials (in particular, green ones) are from the slides of the authors of the textbook and their group at the University of Waikato.

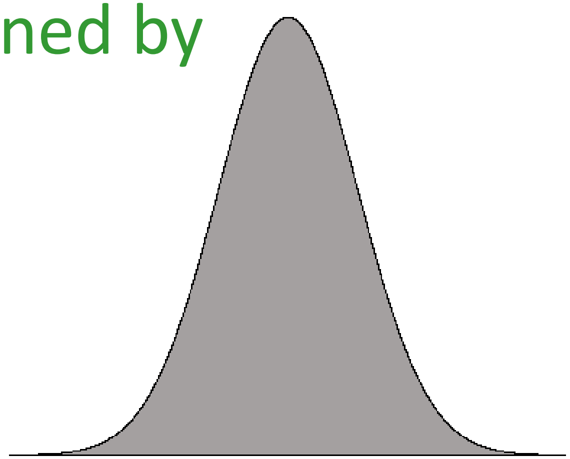
1. Normal distributions

- A normal distribution (sometimes called, a *Gaussian*) is a probability distribution.

- The "*probability density function*" for the normal distribution is defined by two parameters:

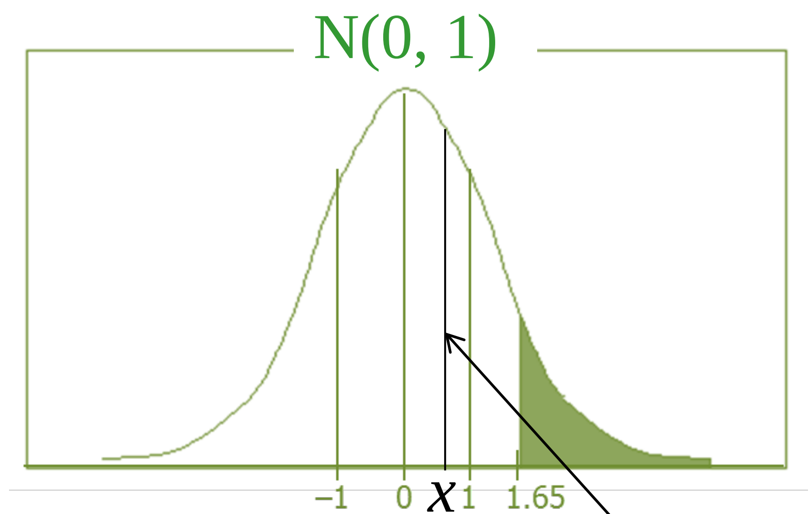
by using expectation μ and standard deviation σ, the density function of $\mathrm{N}(\mu, \sigma)$ is defined by

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



How to use:

Suppose that we obtain n values $x_1, \ldots, x_n$ as the outcomes of n independent random evaluations of X.



What can we say about the probability that $X = x$ for the next evaluation?

$$\boxed{\Pr[\, X = x \,] = 0}$$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

But, we may approximately claim

$$\Pr[\, x - \varepsilon < X < x + \varepsilon \,] \fallingdotseq 2\varepsilon \times f(x)$$

by using

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

Why?

Should be $n-1$ instead of n

Why?

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

This is because in this way, we have $E[\hat{\sigma}^2] = \sigma^2 := V[X]$. Let X_i denote the i th evaluation of X. Then we have $E[\hat{\mu}] = \mu := E[X]$, and

$$E[\hat{\sigma}^2] = E\left[\frac{1}{n-1}\sum (X_i - \hat{\mu})^2\right] = \frac{1}{n-1}\sum E\left[\left(X_i - \frac{\sum X_j}{n}\right)^2\right] \quad \text{— ①}$$

$$E\left[\left(X_i - \frac{\sum X_j}{n}\right)^2\right] = E[X_i^2] - \frac{2}{n}E\left[X_i \sum X_j\right] + \frac{1}{n^2}E\left[\left(\sum X_j\right)^2\right]$$

$$= E[X_i^2] - \frac{2}{n}E[X_i^2] - \frac{2}{n}E\left[X_i \sum_{i\neq j} X_j\right]$$

$$+ \frac{1}{n^2}\left(E\left[\sum X_j^2\right] + E\left[2\sum_{j<j'} X_j X_{j'}\right]\right)$$

$$= E[X_i^2] - \frac{2}{n}E[X_i^2] + \frac{1}{n^2}E\left[\sum X_j^2\right]$$

$$- \frac{2}{n}E\left[X_i \sum_{i\neq j} X_j\right] + \frac{2}{n^2}E\left[\sum_{j<j'} X_j X_{j'}\right]$$

$E[X_i^2] = E[X_j^2]$

$E[X_i] = E[X_j]$

$$= \left(1 - \frac{1}{n}\right)E[X_i^2] - \frac{2(n-1)}{n}E[X_i]^2 + \frac{2}{n^2}\frac{n(n-1)}{2}E[X_i]^2$$

side note

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

$E[X_i^2] = E[X_j^2]$

$E[X_i] = E[X_j]$

$$= \left(1 - \frac{1}{n}\right)E[X_i^2] - \frac{2(n-1)}{n}E[X_i]^2 + \frac{2}{n^2}\frac{n(n-1)}{2}E[X_i]^2$$

$$= \left(1 - \frac{1}{n}\right)E[X_i^2] - \left(1 - \frac{1}{n}\right)E[X_i]^2$$

$E[X_i^2] - E[X_i]^2$
$= V[X_i]$

$$= \frac{n-1}{n}\left(E[X_i^2] - E[X_i]^2\right) = \frac{n-1}{n}V[X_i]$$

$$\therefore \ ① = \frac{1}{n-1}\sum \frac{n-1}{n}V[X_i] = \frac{1}{n}\sum V[X_i] = \frac{n\sigma^2}{n} = \sigma^2$$

Recall

$$E[\hat{\sigma}^2] = E\left[\frac{1}{n-1}\sum (X_i - \hat{\mu})^2\right] = \frac{1}{n-1}\sum E\left[\left(X_i - \frac{\sum X_j}{n}\right)^2\right] \quad \text{—— } ①$$

Why?

This is because $\hat{\mu}$ is not a real expectation; it is also calculated from data.

The slide p9 of W3Lec. wasn't correct; we should have used $\hat{p}(1-\hat{p})\,/\,(n-1)$.

sorry

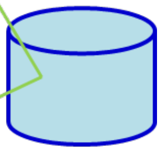
2. Numerical attributes
2.1. In the Naive Bayes

Recall that what we wanted to compute in the Naive Bayes is the following probabilities.

$$\Pr[\, P = \text{Yes} \mid (\mathit{Weather}, T, H, W) = (\text{sunny, 80, 90, windy})\,]$$
$$\Pr[\, P = \text{No} \mid (\mathit{Weather}, T, H, W) = (\text{sunny, 80, 90, windy})\,]$$

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...	...	...	...	...

class value



And we compute these probabilities by

prob. involving numerical value

$$\Pr[\, P = \text{no} \mid (\mathit{Weather}, T, H, W) = (\text{sunny, 80, 90, windy})\,]$$
$$\doteqdot \frac{\Pr[\mathit{Wth}{=}\text{sny} \mid P{=}\text{n}] \cdot \Pr[T{=}\,80 \mid P{=}\text{n}] \cdot \Pr[H{=}90 \mid P{=}\text{n}] \cdot \Pr[W{=}\text{wind} \mid P{=}\text{n}] \cdot \Pr[P{=}\text{n}]}{\Pr[\mathit{Wth}{=}\text{sny} \;\&\; T{=}\,80 \;\&\; H{=}90 \;\&\; W{=}\text{wind}\,]}$$

Outlook			Temperature		Humidity		Windy			Play	
	Yes	*No*	*Yes*	*No*	*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	64, 68,	65, 71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72, 80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, ...	85, ...	80, ...	95, ...					
Sunny	2/9	3/5	μ =73	μ =75	μ =79	μ =86	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	σ =6.2	σ =7.9	σ =10.2	σ =9.7	True	3/9	3/5		
Rainy	3/9	2/5									

Example density value:

$$f(temperature = 66 \mid yes) = \frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.0340$$

A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Likelihood of "yes" = 2/9 × 0.0340 × 0.0221 × 3/9 × 9/14 = 0.000036

Likelihood of "no" = 3/5 × 0.0291 × 0.0380 × 3/5 × 5/14 = 0.000136

P("yes") = 0.000036 / (0.000036 + 0. 000136) = 20.9%

P("no") = 0.000136 / (0.000036 + 0. 000136) = 79.1%

2. Numerical attributes
2.2. Discretization

Discretization

Divide the range of attribute values into a finite number of bins $b_1,\ldots, b_k$ and transform an attribute value v to the index i such that $v \in b_i$.

- Discretize numeric attributes
- Divide each attribute's range into intervals
  - Sort instances according to attribute's values
  - Place breakpoints where the class changes (the majority class)
  - This minimizes the total error
- Example: *temperature* from weather data

supervised

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...	...	...	...	...

```
64    65    68   69   70     71  72  72     75   75     80    81    83    85
Yes | No | Yes Yes Yes | No No Yes | Yes Yes | No | Yes Yes | No
```

More elaborated way:
(in the supervised discretization)

Question: How to choose bin number k?

small ← k → large

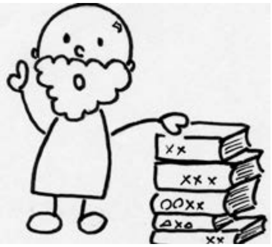
simpler ↔ less error

attr val

There is a way to choose an appropriate k and bins based on *entropy* analysis.

Isn't it a decision stump?

Yes! And it is a key of decision tree learning algorithms ⇒ this feature is included in J4.8



No need to worry!

Yes and No

careful discretization is sometimes better

3. Regression

What if the class value itself is numerical?

Our goal is to create: classifier ⇒ "numerical estimator"

In Weka, both are called a classifier

||
a model (i.e., rule) for computing a class value

regression 回帰分析
= a process for obtaining a numerical estimator.

this may be a recent generalization

Two issues:

1. How to express an estimator?

⇒ <u>linear function</u>, log likelifhood, SVM, perceptron (≦ neural network), decision tree

2. How to compute an estimator?

⇒ <u>least squares method</u>, perceptron learning algo.s

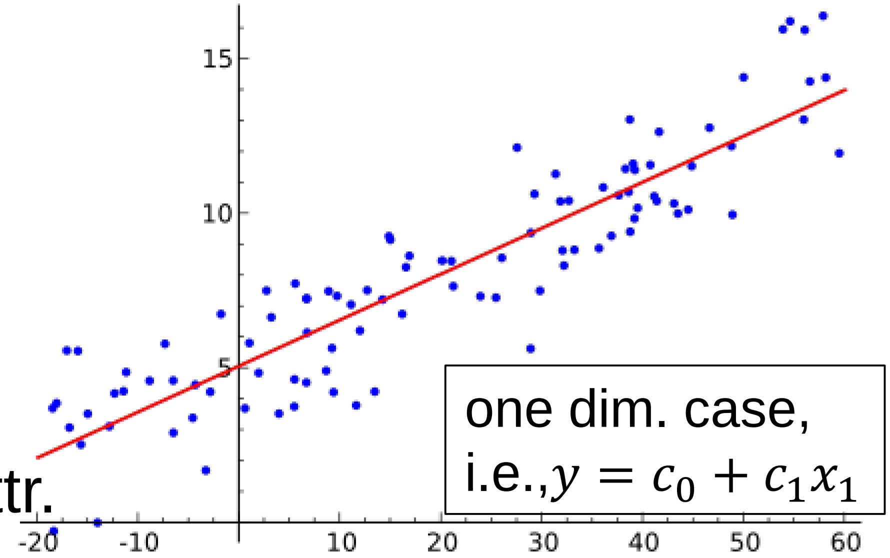
3. Regression
3.1. Linear estimator

Linear function (multi-linear)

$$y = c_0 + c_1 x_1 + \cdots + c_m x_m$$

which can be regarded as
a def. of a hyperplane.

In our situation, y is the class attr.
and $x_1, \ldots, x_m$ are the other attr.s.



one dim. case,
i.e., $y = c_0 + c_1 x_1$

By Sewaqu - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=11967659

Why linear func.?

Simpler better & it indeed works in various situations.

Of course, it should not work always!
For more complicated models: perceptron, SVM, etc.

3. Regression
3.2. Linear regression

||

use linear estimator & obtain it by the least squares method

Let me use the material from the textbook, and for this, change the usage of symbols.

Our goal is to determine the following estimator from the following training data set:

$$x = w_0 + w_1 x_1 + \cdots + w_k x_k$$

class value	attribute values		
$x^{(1)}$	$a_1^{(1)}$	...	$a_k^{(1)}$
	⋮		
$x^{(n)}$	$a_1^{(n)}$	...	$a_k^{(n)}$

- Weights are calculated from the training data
- Predicted value for first training instance $\mathbf{a}^{(1)}$

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$

Least squares method

class value	attribute values		
$x^{(1)}$	$a_1^{(1)}$	…	$a_k^{(1)}$
	⋮		
$x^{(n)}$	$a_1^{(n)}$	…	$a_k^{(n)}$

$$x = w_0 + w_1 x_1 + \cdots + w_k x_k$$

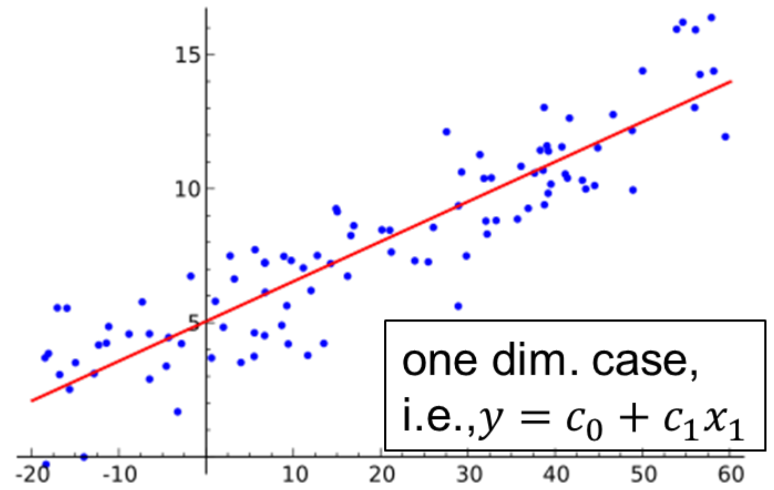
- Choose *k* +1 coefficients to minimize the squared error on the training data
- Squared error: $$\sum_{i=1}^{n}\left(x^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)}\right)^2$$
- Derive coefficients using standard matrix operations
- Can be done if there are more instances than attributes (roughly speaking)
- Minimizing the *absolute error* is more difficult

Some reasoning:

Consider the simplest 1-dim. case.

$$Y = c_0 + c_1 X$$

class value	attr value
b_1	a_1
$\vdots$	$\vdots$
b_n	a_n



Suppose our goal is maximizing

$$\Pr[\ Y_1, \ldots, Y_n = b_1, \ldots, b_n \mid X_1 = a_1, \ldots, X_n = a_n\ ]$$

where X_i and Y_i respectively is the random variable corresponding to the i th instance of the data set.

Suppose further each error (i.e., noise) follows the normal dist. $N(0, \sigma^2)$ independently. That is, $Y_i = c_0 + c_1 X_i + Z_i$ and $Z_i \sim N(0, \sigma^2)$. Then our task is to mimimize

$$\Pr[\ Z_1 = b_1 - (c_0 + c_1 a_1), \ldots\ ] \propto \exp\left(-\frac{1}{2\sigma^2} \|\boldsymbol{b} - (c_0 + c_1 \boldsymbol{a})\|^2\right).$$

3. Regression
3.3. For classifier

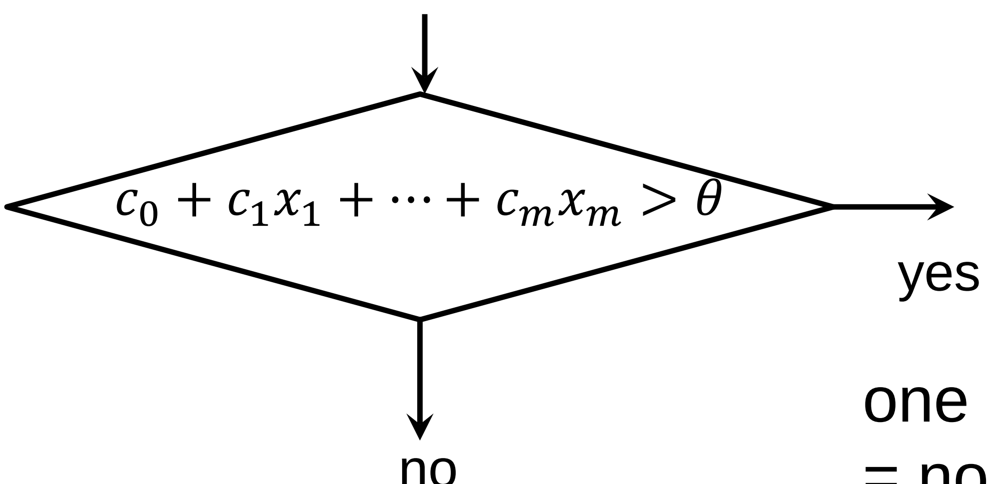
Suppose we get a good linear estimator like this

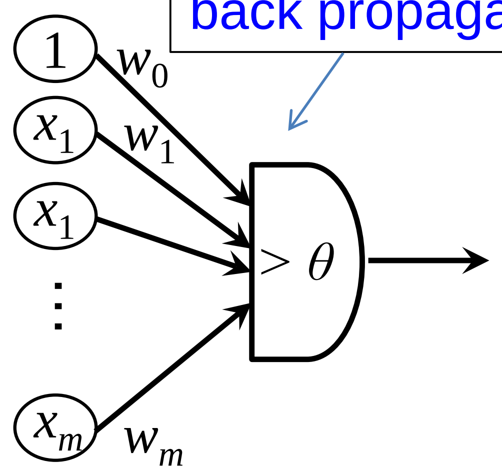
$$f(\boldsymbol{x}) = c_0 + c_1 x_1 + \cdots + c_m x_m$$

for the numerical class value y. Then this can be used for the classification task, most typically, to determine whether $y > \theta$, for a given threshold parameter θ.

different learning algo.
back propagation, etc.

That is, the following classier:





one layer perceptron
= no hidden layer perceptron

Limit of linear estimators as a classifier:

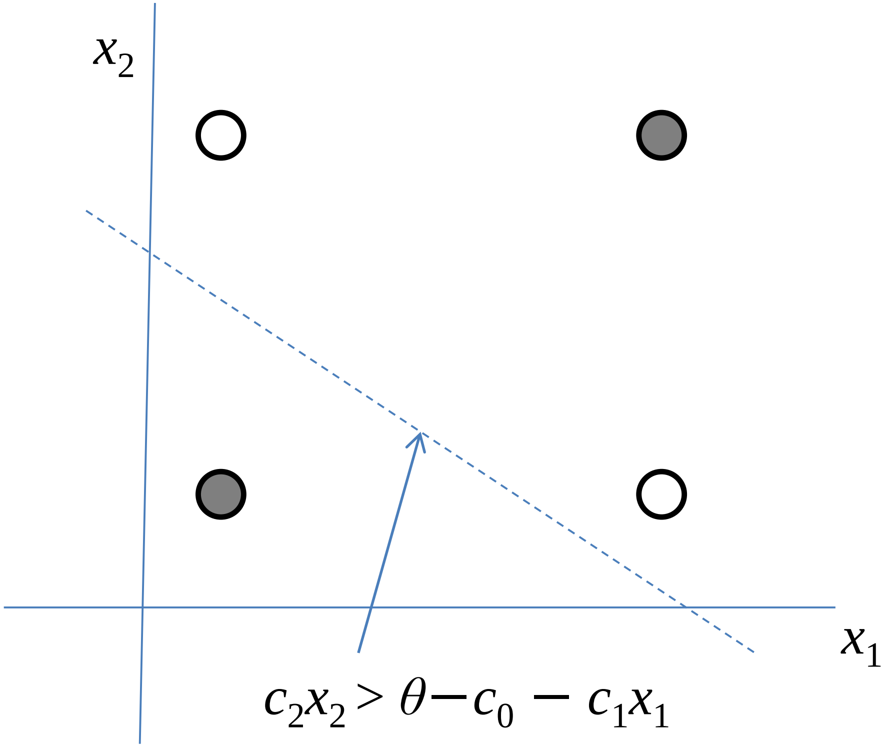
by Marvin Minsky and Seymour Papert

An example case:
The following classification of 2-dim. case.

We need to classify by

$$y = c_0 + c_1 x_1 + c_2 x_2 > \theta$$

but this is impossible! More precisely, the error cannot be reduced less than 1/4 = 25%.



3. Regression
3.4. How to take care of nominal attr.s

What shall we do if the data has some nominal attributes such as

color = red, yellow, green, blue, black
$\Rightarrow$ 0, 1, 2, 3, 4 ×

humidity = low, medium, high
$\Rightarrow$ 0, 1, 2 OK

quality = good, bad $\Rightarrow$ 0, 1 OK

Do not change them to numerical values unless this still make sense!

Since the binary case is always OK, one possibility is to change all nominal attributes to the binary ones.
E.g., color-red = 0 (no)/1 (yes), color-yellow = 0/1, ...

maybe better ways

4. On Exercise #4

- How to take care of numerical attributes, and the mixed case.
- Introduction to linear regression and a NN type model.

Task #1: How to handle numerical attr.s.

(a) By using several example data sets, try various methods*1 to obtain several classification rules and compare these methods from the results.

*1 For example, Weka Preprocess: "Discritize", "NumericToNominal", etc. You might want to use "MultilayerPerceptron" that can be chosen from the choice "function." In this case, use it with the no hidden layer.

Task #2: Use the linear regression, etc.

(b) Use breast-Tumor.arff to experience basic learning algo.s for computing linear estimators.

+ Use linear reguression methods, i.e., "LinearRegression" and "MultilayerPerceptron" (under the single layer option).
+ Try several ways to change nominal attr.s to numerical ones, and compare the obtained estimators.

your original ways are encouraged