



## ΣΥΛΟΓΗ ΔΕΛΤΙΩΝ

1107  
0101  
1110



# Logistic Regression

we finished this lecture

- Logistic regression is a **probabilistic** classifier that makes use of **supervised machine learning**.
- It is also a **discriminative** classifier: distinguish between the classes by learning features.
- In natural language processing, **logistic regression** is the **baseline supervised machine learning algorithm for classification**, and also has a very close relationship with **neural networks**.



# Binary Logistic Regression

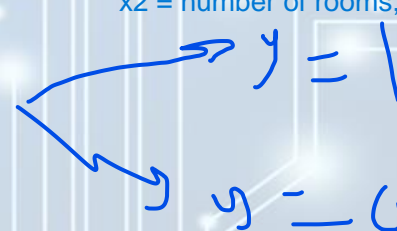
- Train a classifier that can make a **binary decision** about the class of a new input observation → **using sigmoid classifier**.
- A single input observation  $x$  represents a vector of features  $[x_1, x_2, \dots, x_n]$ .
- The classifier output  $y$  can be **1** (meaning the observation **is a** member of the class) or **0** (the observation is **not** a member of the class).
- We want to know the probability  $P(y = 1/x)$ ,  
suppose the decision is either “*positive sentiment*” or “*negative sentiment*”:
  - $P(y = 1/x)$  is the probability that the document has **positive sentiment**.
  - $P(y = 0/x)$  is the probability that the document has **negative sentiment**.
- Logistic regression solves this task by learning, from a training set, **a vector of weights** and **a bias term**.

# Binary Logistic Regression

- Each **weight  $w_i$**  is a **real number** that is **associated** with **one** of the **input features  $x_i$** .
  - The weight  $w_i$  **represents** how **important** that input feature is to the classification decision.
  - It can be **positive** (providing evidence that the instance being classified belongs in the positive class) or **negative** (providing evidence that the instance being classified belongs in the negative class).
  - Example: in a sentiment task the word **awesome** is expected to have a **high positive weight**, while the word **bad** is expected to have a high negative weight.
- The **bias term**, also called the **intercept**, is another **real number** that's added to the **weighted inputs**.



$x_1$  = area  
 $x_2$  = number of rooms,





# Binary Logistic Regression

- To make a decision on a test instance:

$$\underline{z} = \left( \sum_{i=1}^n w_i x_i \right) + \underline{b}$$

The resulting single number  $z$  expresses the weighted sum of the evidence for the class.

- Using linear algebra, the above  $z$  equation can be re-written as:

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

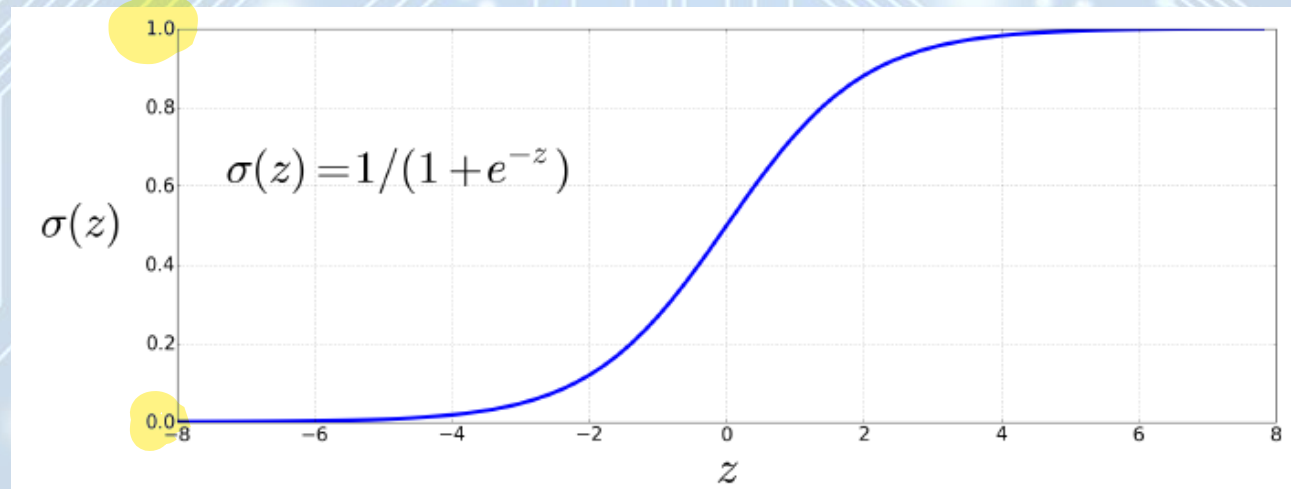
Where:  $w$  and  $x$  are vectors  
 $\cdot$  is the dot product between two vectors

- Nothing in the equation forces  $z$  to be a legal probability, (lie between 0 and 1).
  - In fact, since weights are real-valued, the output might even be negative.
  - $z$  ranges from  $-\infty$  to  $\infty$ .

# Binary Logistic Regression

- To create a probability, we'll pass  $z$  through the **sigmoid function/logistic function**,  $\sigma(z)$ :

$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-z)}$$



The sigmoid function takes a **real value** and maps it to the range **[0,1]**.  
It is nearly linear around 0 but flattens toward the ends  
→ it tends to **squash outlier values toward 0 or 1**.

# Binary Logistic Regression

- To make it a probability, we just need to make sure that the two cases,  $p(y = 1)$  and  $p(y = 0)$ , sum to 1.

$$\begin{aligned}P(y = 1) &= \frac{\sigma(\mathbf{w} \cdot \mathbf{x} + b)}{1} \\&= \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))} \\P(y = 0) &= \frac{1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)}{1} \\&= 1 - \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))} \\&= \frac{\exp(-(\mathbf{w} \cdot \mathbf{x} + b))}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x} + b))}\end{aligned}$$

Important property of sigmoid function:

$$1 - \sigma(x) = \sigma(-x)$$

$$P(y = 0) \text{ as } \sigma(-(\mathbf{w} \cdot \mathbf{x} + b))$$

- How do we make a *decision* about the class of a test instance  $x$ ?

→ Using the decision boundary:

$$\text{decision}(x) = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

# Binary Logistic Regression

## Sentiment Classification:

- Features:

Var	Definition	Value in Fig. 1
$x_1$	count(positive lexicon words $\in$ doc)	3
$x_2$	count(negative lexicon words $\in$ doc)	2
$x_3$	$\begin{cases} 1 & \text{if "no" } \in \text{ doc} \\ 0 & \text{otherwise} \end{cases}$	1
$x_4$	count(1st and 2nd pronouns $\in$ doc)	3
$x_5$	$\begin{cases} 1 & \text{if "!" } \in \text{ doc} \\ 0 & \text{otherwise} \end{cases}$	0
$x_6$	log(word count of doc)	$\ln(66) = 4.19$

- Sample test document:

It's **hokey**. There are virtually **no** surprises, and the writing is **second-rate**. So why was it so **enjoyable**? For one thing, the cast is **great**. Another **nice** touch is the music. **I** was overcome with the urge to get off the couch and start dancing. It sucked **me** in, and it'll do the same to **you**.

Feature values for the sample document:

- $x_1 = 3$  (positive words: hokey, enjoyable, great, nice)
- $x_2 = 2$  (negative words: no, second-rate, sucked)
- $x_3 = 1$  (word "no" present)
- $x_4 = 3$  (1st and 2nd pronouns: I, me, you)
- $x_5 = 0$  (word "!" not present)
- $x_6 = 4.19$  (log(word count) = log(66))



# Binary Logistic Regression

## Sentiment Classification:

- Let's assume that we've already learned  $w$  and  $b$ :

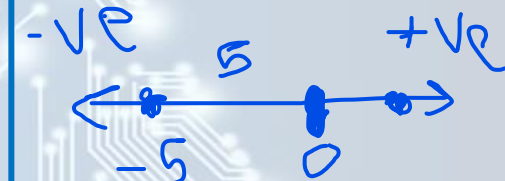
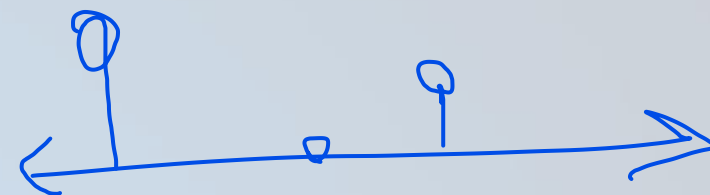
- $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$  and  $b = 0.1$

$w_1$ : tells us the importance of positive lexicon words.

$w_2$ : tells us the importance of negative lexicon words.

$w_1 = 2.5$  is positive, while  $w_2 = -5.0$  is negative:

meaning that negative words are **negatively** associated with a positive sentiment decision and are about **twice** as important as positive words.



$$\begin{aligned} p(+|x) &= P(y = 1|x) = \sigma(\mathbf{w} \cdot \mathbf{x} + b) \\ &= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\ &= \sigma(.833) \\ &= 0.70 \\ p(-|x) &= P(y = 0|x) = 1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b) \\ &= 0.30 \end{aligned}$$

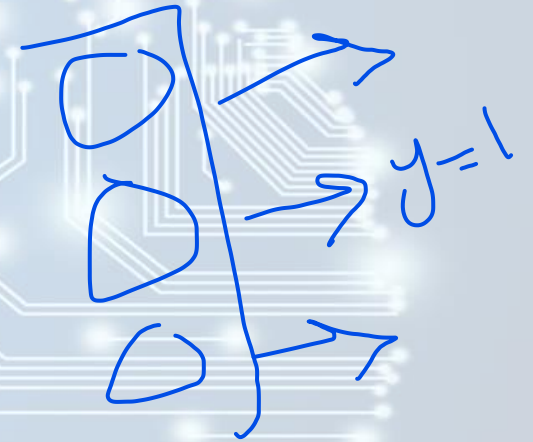
# Multinomial Logistic Regression

- Also called **softmax regression** (in older NLP literature sometimes referred to as **maxent classifier**).
- For handling ***more than two*** classes.
- We want to label each observation with a class  $k$  from a set of  $K$  classes (called **hard classification**: an observation cannot be in multiple classes).
- Let's use the following representation: the output  $y$  for each input  $x$  will be a vector of length  $K$ . If class  $c$  is the correct class, we'll set  $y_c = 1$ , and set all the other elements of  $y$  to be 0, i.e.,  $y_c = 1$  and  $y_j = 0 \forall j \neq c$  ( $y$  is a one-hot vector).
- The job of the classifier is produce an estimate vector  $\hat{y}$ . For each class  $k$ , the value  $\hat{y}_k$  will be the classifier's estimate of the probability  $p(y_k = 1/x)$ .

# Multinomial Logistic Regression

- Uses a generalization of the **sigmoid**, called the **softmax function**
- The softmax function takes a vector  $z = [z_1, z_2, \dots, z_K]$  of  $K$  arbitrary values and maps them to a **probability distribution**, with each value in the **range (0,1)**, and all the values **summing to 1**.

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^K \exp(z_j)} \quad 1 \leq i \leq K$$




- Like the **sigmoid**,
  - it is an **exponential** function.
  - the softmax has the property of **squashing** values toward 0 or 1. Thus if one of the inputs is **larger than the others**, it will tend to push **its probability toward 1**, and suppress the probabilities of the smaller inputs.

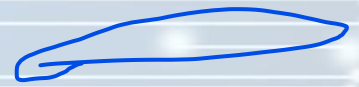


# Multinomial Logistic Regression

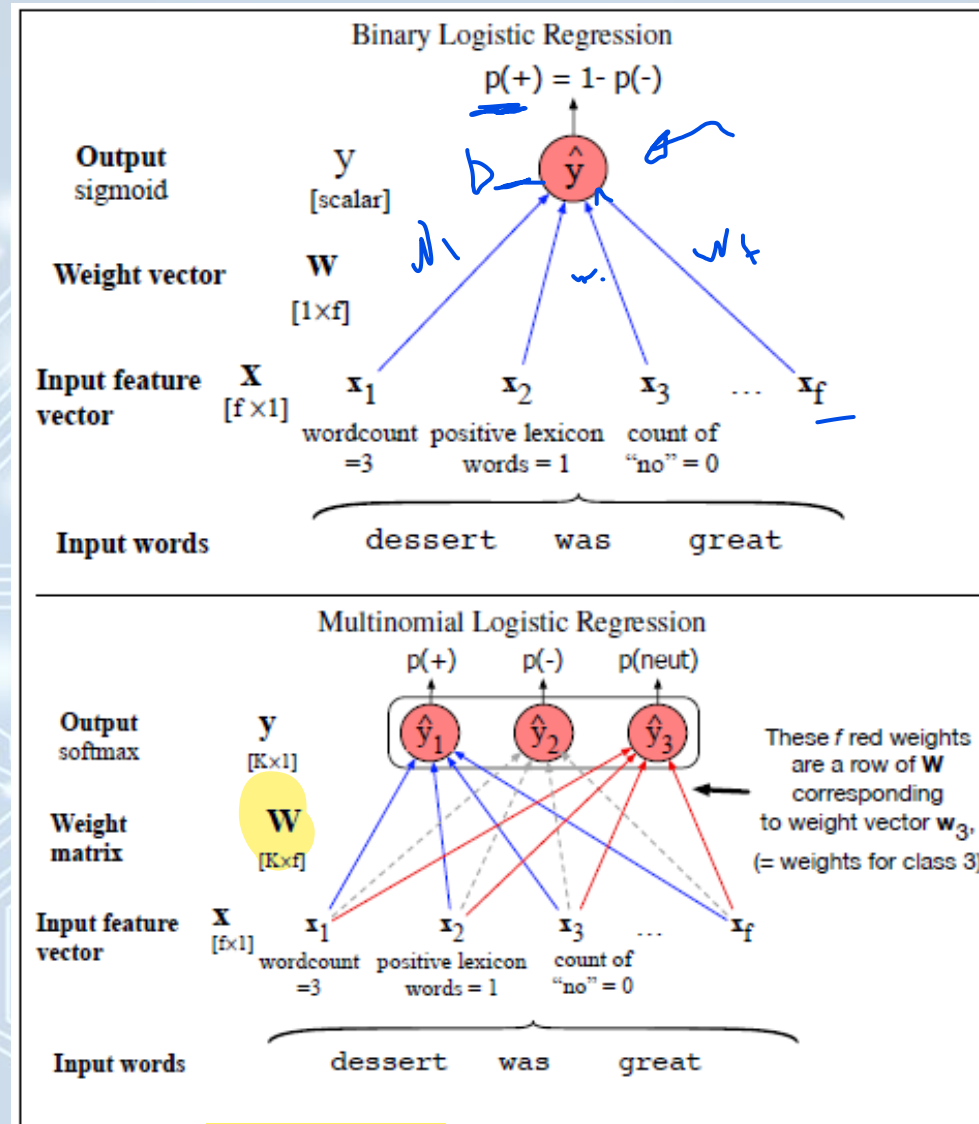
- Now we'll need separate weight vectors  $w_k$  and bias  $b_k$  for each of the  $K$  classes.


$$p(\mathbf{y}_k = 1 | \mathbf{x}) = \frac{\exp(\mathbf{w}_k \cdot \mathbf{x} + b_k)}{\sum_{j=1}^K \exp(\mathbf{w}_j \cdot \mathbf{x} + b_j)}$$

- Using linear algebra:
  - $\mathbf{W}$  is a weight matrix, each row  $k$  of  $\mathbf{W}$  corresponds to the vector of weights  $w_k$ ,  $\mathbf{W}$  has shape  $[K \times f]$  where  $f$  is the number of input features.
  - $\mathbf{b}$  is a bias vector.


$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

# Logistic Regression



- We want to learn parameters ( $w$  and  $b$ )
  - Need *loss/cost function*, commonly used: **cross-entropy loss**.
  - Need an *optimization algorithm* for iteratively updating the parameters so as to minimize the loss function, the standard algorithm is **gradient descent**.

# Logistic Regression vs Naïve Bayes

- Logistic regression is much **more robust to correlated features** while Naïve Bayes has **overly strong conditional independence** assumptions.
  - Consider two features  $f_1$  and  $f_2$  which are strongly correlated; imagine that we just add the same feature  $f_1$  twice:
    - Naïve Bayes will treat both copies of  $f_1$  as if they were separate, multiplying them both, overestimating the evidence.
    - By contrast in logistic regression, if two features  $f_1$  and  $f_2$  are perfectly correlated, then regression will simply assign part of the weight to  $w_1$  and part to  $w_2$ .
- **When there are many correlated features, logistic regression will assign a more accurate probability than naive Bayes.**
- Despite the **less accurate probabilities**, naive Bayes still often **makes the correct classification decision**.
- Logistic regression is also one of the **most useful analytic tools**, because of its ability to transparently **study the importance of individual features**.
- Naïve Bayes is **easy to implement and very fast to train** (there's **no optimization step**).





# Thank You

0100

0100

0110

0110

1001

1001

0100

0100

1101

0100

0110

0100

0100

0101

0100

0110

0100

0100

0101

1010

0110

1001

0101

1101

1010

0110

1001

0101

0110

0100

1101

1010

0110

1001