# Logistic Regression

T-662-ARTI

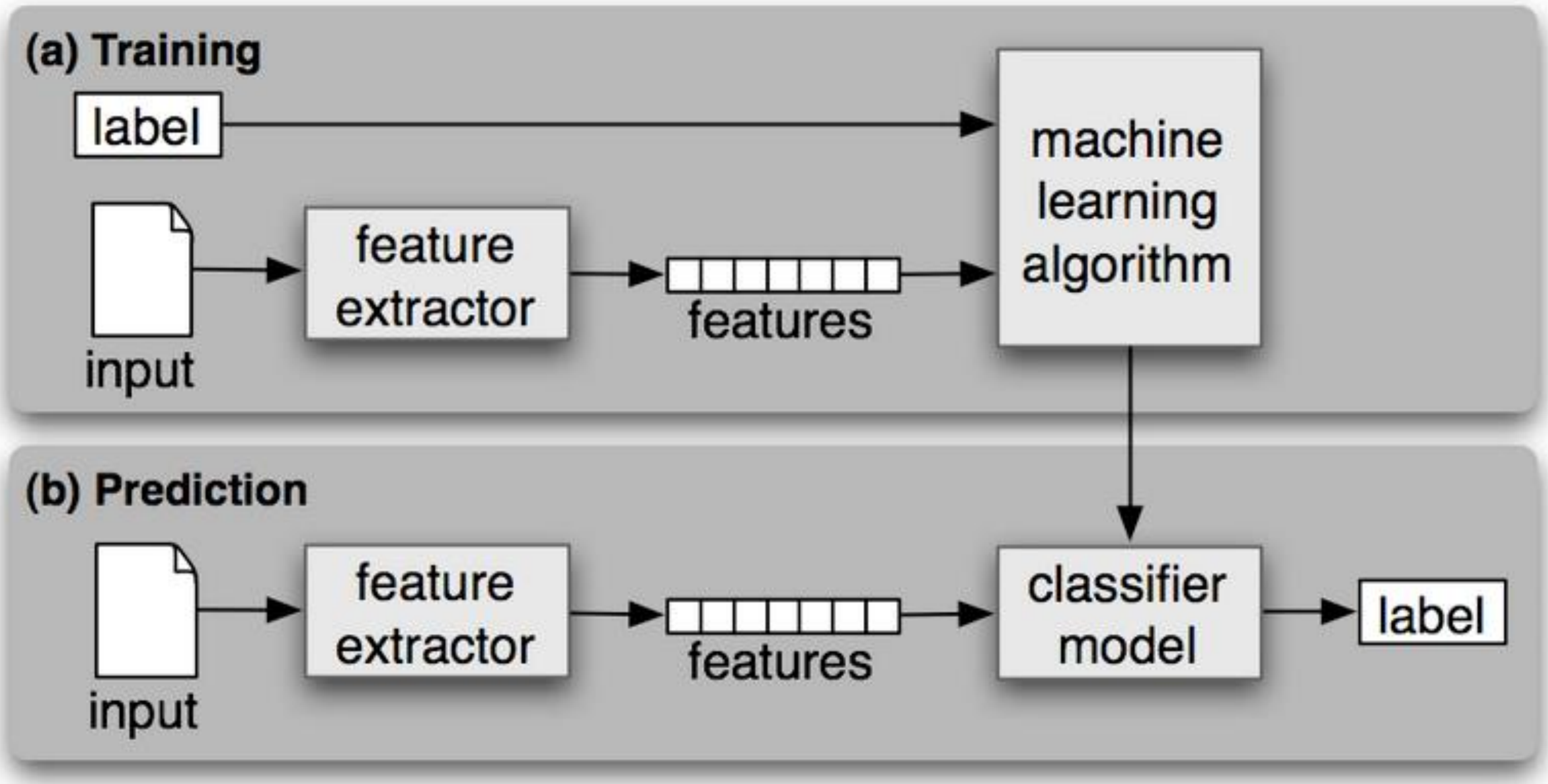
Stefán Ólafsson
Spring 2023

# Classification

- **The task of choosing the correct *class label* for a given input**

- **Examples:**
  - Assigning subject categories, topics, or genres
  - Spam detection
  - Authorship identification
  - Language Identification
  - Sentiment analysis
  - …

# Supervised Classification




https://www.nltk.org/book/ch06.html

# Generative vs. Discriminative Classifier

- **Naïve Bayes is a Generative Classifier**

$$c_{NB} = \underset{c \in C}{argmax}\, P(d|c)P(c)$$

- **P(c|d) *not* computed directly**
- **P(d|c) expresses how to 'generate' the features of d, given class c**

# Generative vs. Discriminative Classifier

- **Logistic Regression is a discriminative classifier**
- **Attempts to directly compute P(c|d)**
- **Learns to assign high weight to features that improve its ability to "discriminate" between possible classes**
- **Cannot generate an example of one of the classes**

# Logistic Regression

- **A training corpus of M observations:**
  - $\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right) \ldots \left(x^{(m)}, y^{(m)}\right)$
- **Four components:**
  - For each input observation $x^{(i)}$:
    - Vector of features $[x_1, x_2, \ldots, x_n]$
  - A classification function that computes the estimated class $\hat{y}$ ⇒ **sigmoid**
  - An objective function for learning, involving minimizing error on training examples ⇒ **cross-entropy loss**
  - An algorithm for optimizing the objective function ⇒ **gradient descent**

# Logistic Regression

- **Estimate $\mathbf{P}(\boldsymbol{y} = \mathbf{1} \mid \boldsymbol{x})$**
- **Learns a vector of weights and a bias term during training**
- **Each weight $\boldsymbol{w_i}$ is a real number associated with feature $\boldsymbol{x_i}$**
  - Represents how important that feature is to the classification decision
  - If positive, then the feature is associated with the class
  - If negative, then the feature is not associated with the class

# Classifying a test instance

- $z = \sum_{i=1}^{n} w_i x_i + b$
- $z = w * x + b$
- z expresses the weighted sum of the evidence for the class
- To create a probability, $z$ is passed through the **sigmoid** function, $\sigma(z)$:
  - $y = \sigma(z) = \frac{1}{1+e^{-z}}$
  - $\hat{y} = \begin{cases} 1 \; if \; p(y = 1|x) > 0.5 \\ 0 \; otherwise \end{cases}$

# The Sigmoid Function

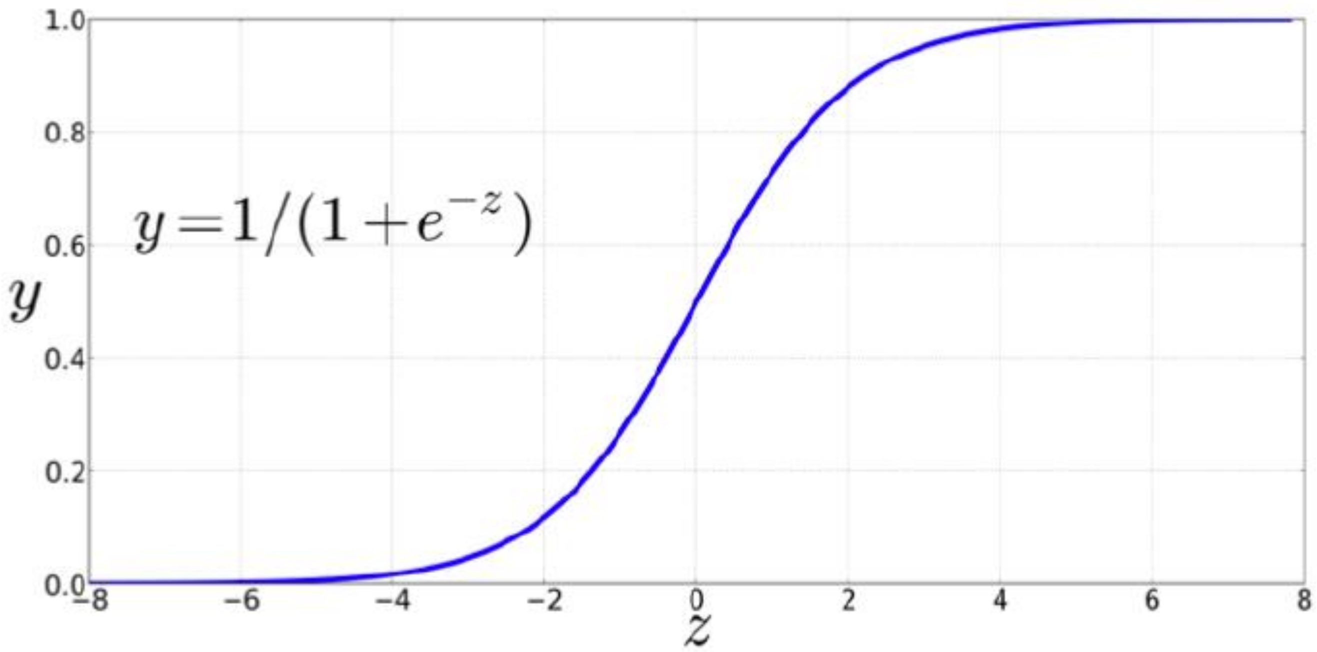



**Figure 5.1** The sigmoid function $y = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $[0, 1]$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

# Example: sentiment classification

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.

- positive = 3
- negative = 2
- “no” = 1
- 1st and 2nd person pronouns = 3
- “!” = 0
- log(word count) = log(64) = 4.15

# Example: sentiment classification

| Var | Definition | Value |
|---|---|---|
| $x_1$ | count(positive lexicon $\in$ doc) | 3 |
| $x_2$ | count(negative lexicon $\in$ doc) | 2 |
| $x_3$ | $x_3 = \begin{cases} 1 \; if \text{ „no“ } \in doc \\ 0 \; otherwise \end{cases}$ | 1 |
| $x_4$ | Count($1^{st}$ and $2^{nd}$ person pronouns $\in$ doc) | 3 |
| $x_5$ | $x_5 = \begin{cases} 1 \; if \text{ „!“ } \in doc \\ 0 \; otherwise \end{cases}$ | 0 |
| $x_6$ | log(word count of doc) | 4.19 |

# Example: sentiment classification

- Let's assume we have already learned the weight vector **w**, and the bias **b**
- w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7], b = 0.1
- p(+|x) = p(Y=1|x) = $\sigma(w * x + b)$

$$= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] * [3, 2, 1, 3, 0, 4.19] + 0.1)$$

$$= \sigma(0.833) = \mathbf{0.70}$$

- p(-|x) = p(Y=0|x) = 1 − $\sigma(w * x + b)$ = **0.30**

# Learning in logistic regression

- Learn weights **w** and bias **b** that make $\hat{y}$ for each training observation as close as possible to the true **y**

1. **loss (cost) function**
2. **Iteratively updating the weights => gradient descent**

# Cross-entropy loss function

- $\boldsymbol{L(\hat{y}, y)}$ **= How much $\boldsymbol{\hat{y}}$ differs from the true $\boldsymbol{y}$**
- **We want to learn weights that maximize the probability of the correct label p(y|x)**
- **Two discrete outcomes (1 or 0)**
- $\mathbf{p}(\boldsymbol{y}|\boldsymbol{x}) = \boldsymbol{\hat{y}}^{\boldsymbol{y}}(\mathbf{1} - \boldsymbol{\hat{y}})^{\mathbf{1}-\boldsymbol{y}}$
  - When y=1: $p(y|x) = \hat{y}$
  - When y=0: $p(y|x) = 1 - \hat{y}$

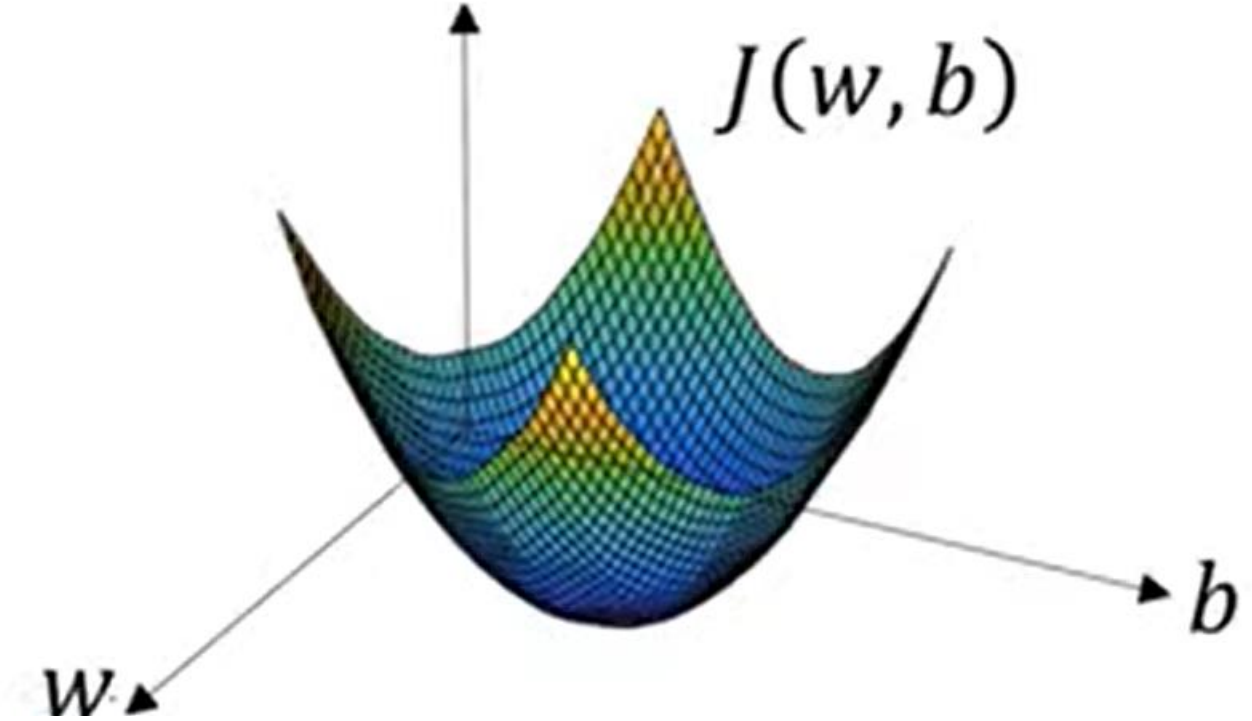
# Cross-entropy loss function

- $\mathbf{\log p(y|x) = y * log\hat{y} + (1 - y) * log(1 - \hat{y})}$
- **To minimize:**
- $\boldsymbol{L_{CE}(\hat{y}, y) = -[y * log\hat{y} + (1 - \hat{y}) * log(1 - \hat{y})]}$
  - When y = 1: this is $-\log\hat{y}$
  - When y = 0: this is $-\log(1 - \hat{y})$

# Gradient Descent

- **Used to find the optimal weights**
- **Minimize the loss function**
  - Cost function:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$




Andrew Ng - https://www.youtube.com/watch?v=uJryes5Vk1o

# Gradient Descent

# Logistic regression derivatives

$$x_1,\ w_1,\ x_2,\ w_2,\ b \rightarrow \boxed{z = w_1 x_1 + w_2 x_2 + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{L(a, y)}$$

$$dz = \frac{dL(a,y)}{dz} = a - y$$

$$da = \frac{dL(a,y)}{da}$$

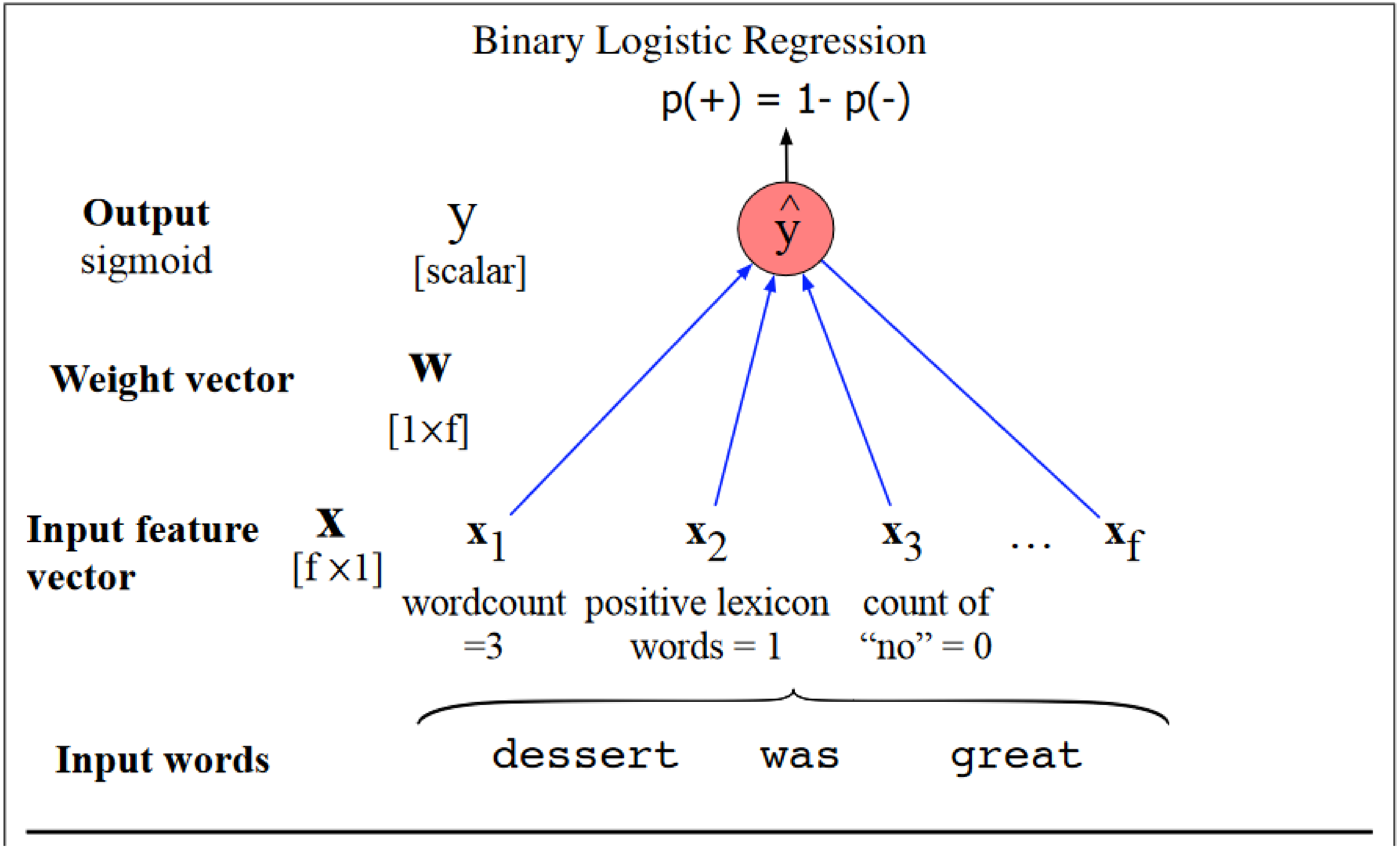
$$dw_1 = \frac{dL}{dw_1} = x_1 * dz$$

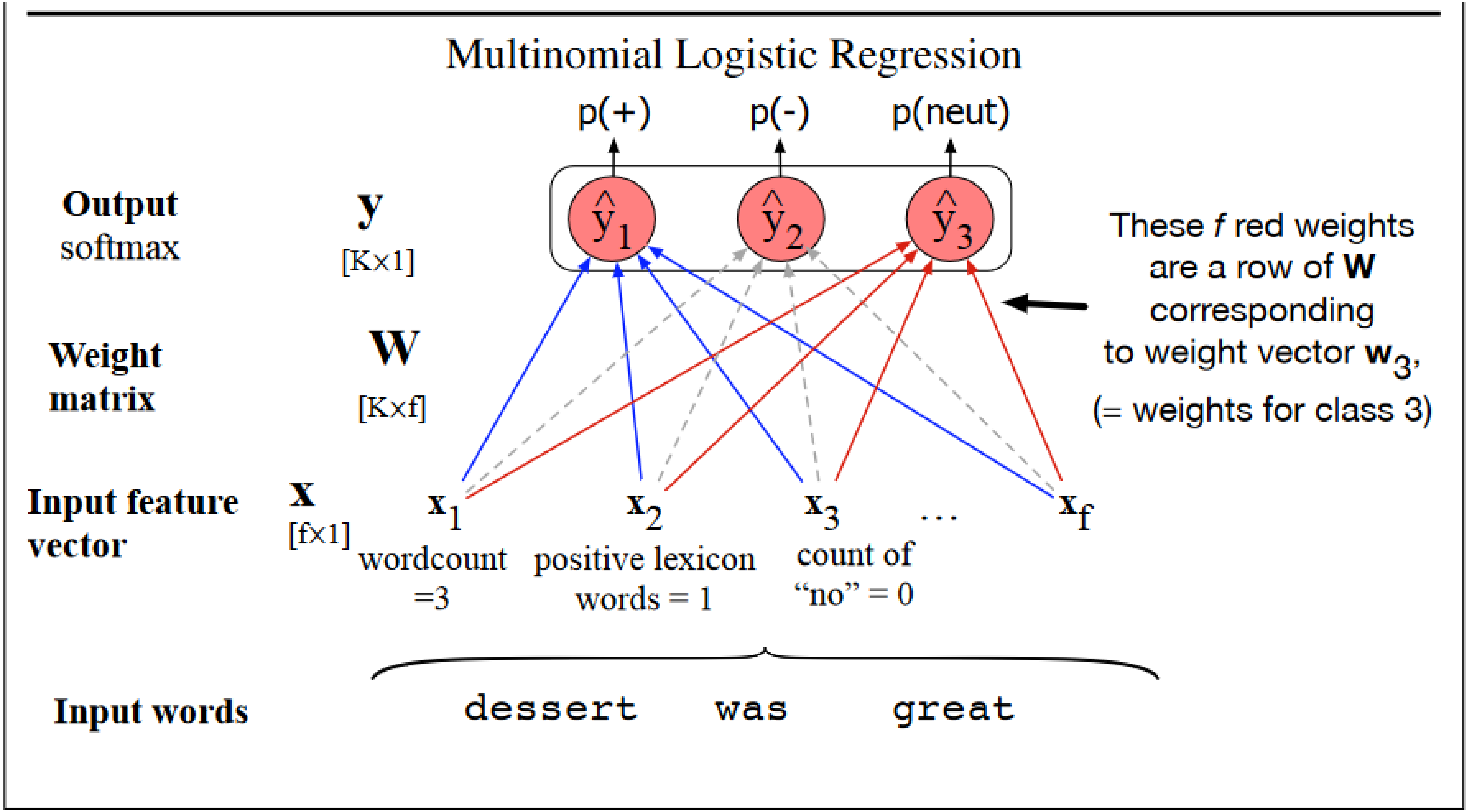
**Parameter updates:**

$$w_1 \leftarrow w_1 - \alpha * dw_1$$

$$w_2 \leftarrow w_2 - \alpha * dw_2$$

$$b \leftarrow b - \alpha * db$$


Binary Logistic Regression
p(+) = 1- p(-)
ŷ
Output
sigmoid
y
[scalar]
Weight vector
w
[1×f]
Input feature
vector
x
[f ×1]
x1
x2
x3
...
xf
wordcount
=3
positive lexicon
words = 1
count of
“no” = 0
Input words
dessert
was
great



Multinomial Logistic Regression
p(+)
p(-)
p(neut)
Output
softmax
y
[K×1]
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
These f red weights are a row of W corresponding to weight vector $w_3$, (= weights for class 3)
Weight matrix
W
[K×f]
Input feature vector
x
[f×1]
$x_1$
$x_2$
$x_3$
...
$x_f$
wordcount =3
positive lexicon words = 1
count of "no" = 0
Input words
dessert was great