Logistic Regression

(Classification algorithm)
(Bag of words model)

Imran Khan, PhD

Note: These slides heavily borrow from the book “Speech and language processing” authored byDaniel Jurafsky and James Martin.

Logistic Regression

- Logistic regression can be used to classify an observation into one of two classes (like 'positive sentiment' and 'negative sentiment'), or into one of many classes.

- Logistic regression is the baseline supervised machine learning algorithm for classification tasks.

- Logistic regression is the probabilistic classifier requires a training corpus of input output pairs $(x^{(i)}, y^{(i)})$

How does supervised text dataset look like?

Texts	Label
"I absolutely loved the new restaurant! The food was delicious and the service was excellent."	Positive
"The movie was fantastic! The plot was engaging and the acting was top-notch."	Positive
The movie was a waste of time. The plot was predictable and the acting was terrible."	Negative
"The weather today is neither too hot nor too cold. It's just an average day."	Neutral
"The concert was a disaster. The sound was terrible and the band seemed unprepared."	Negative
"I was very disappointed with the service at the restaurant. The food was cold and the staff were rude."	Negative

How many examples/instances/observations do we have in the above dataset?
Is this a binary class classification problem or multiclass classification problem?
What do we mean by features? Where are the features in the above examples?

Four components of ML classification

- A feature representation of the input.
- A classification function (sigmoid or softmax) that computes the estimated class via *P(y|x)*.
- An objective function (loss function) for learning, usually involving minimizing error (cross-entropy loss) on training example.
- An algorithm for optimizing (stochastic gradient descent) the objective function.

Generative versus Discriminative

Generative versus Discriminative

Discriminative model:

In discriminative model, we directly compute probability of class c given document d, *P(c|d)*

How does the discriminative model learn?
Model puts higher weight on features that directly improves its ability to discriminate between possible classes.

Generative model

- How does naïve Bayes build the ML model? It assigns a class **c** to document **d** *not* by directly computing *P(c|d)* but by computing **likelihood** and **prior.**

$$\hat{c} = \operatorname*{argmax}_{c \in C} \overbrace{P(d|c)}^{\text{likelihood}} \; \overbrace{P(c)}^{\text{prior}}$$

Generative model
How to generate features of a document if we know if it was of class c?

Sigmoid function

- The logistic regression (using sigmoid function) helps us to classify each example in one of two classes. We want to know the probability P(y = 1/x) and P(y = 0/x) (say) representing positive sentiment and negative sentiment.

- How does the logistic regression solve this task of learning? It learns weights and a bias. Each weight (parameter) is a real number that represents the importance of input feature to the classification decision.

Example

We might assign high positive weight to word “awesome” because we tend to associate this word with positive sentiment. Similarly, we might assign very negative weight to word “abysmal” because we tend to associate this word with negative sentiment.

What do we mean by “learning” in Machine Learning?

Sigmoid function

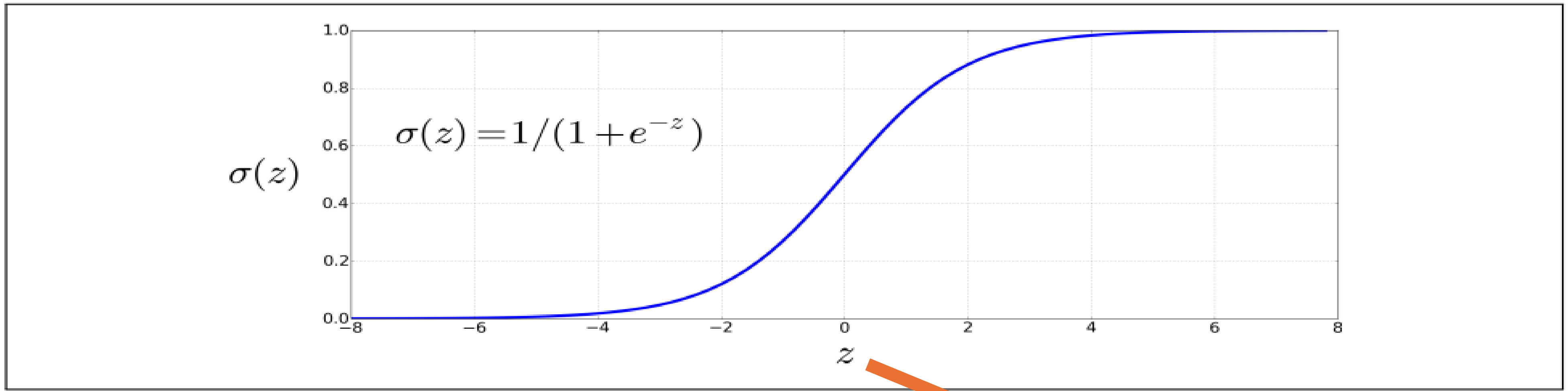


Figure 5.1 The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $(0,1)$. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

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

z ranges from $-\infty$ to ∞.

dot product between weights and features
z is known as logit

Sigmoid function

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

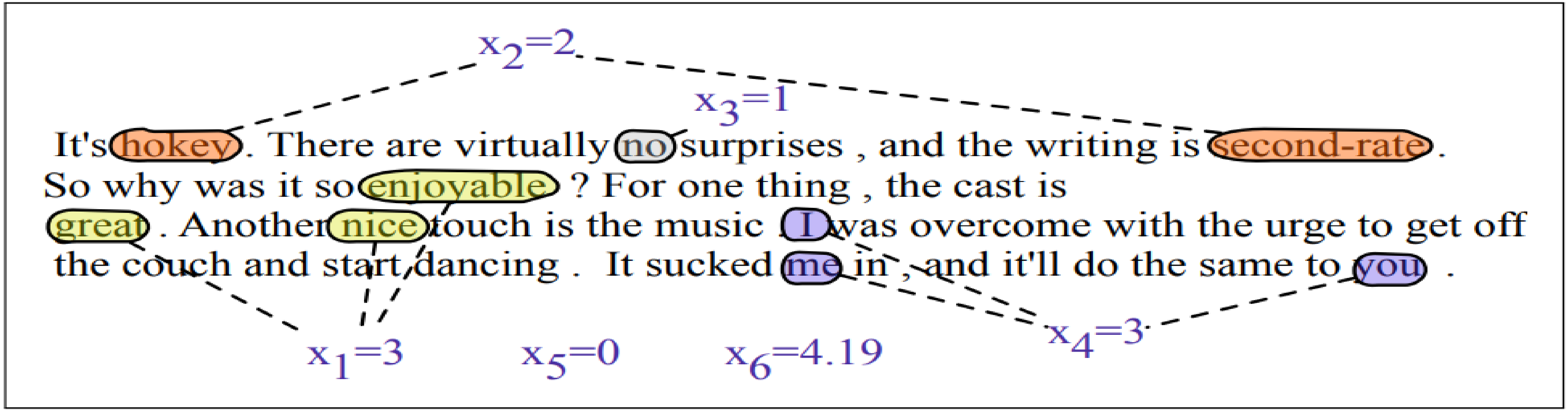
$$\begin{aligned}
P(y=0) &= 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b) \\
&= 1-\frac{1}{1+\exp\left(-(\mathbf{w}\cdot\mathbf{x}+b)\right)} \\
&= \frac{\exp\left(-(\mathbf{w}\cdot\mathbf{x}+b)\right)}{1+\exp\left(-(\mathbf{w}\cdot\mathbf{x}+b)\right)}
\end{aligned}$$

Classification with Logistic Regression

- Sigmoid function helps us compute probability P(y =1|x). Now the question is how to classify the given example in one of two classes.
- **Decision boundary**

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

Simple sentiment analysis example using logistic regression



x2=2
x3=1
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 .
x1=3
x5=0
x6=4.19
x4=3

Figure 5.2 A sample mini test document showing the extracted features in the vector x.

Var	Definition	Value in Fig. 5.2
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	ln(word count of doc)	$\ln(66) = 4.19$

Features

How many weights we are going to have?

Let us suppose six weights corresponding to six features are [2.5, -5.0, -1.2, 0.50, 2.0, 0.70] and a bias b = 0.1

$$\begin{aligned}
p(+|x) = P(y=1|x) &= \sigma(\mathbf{w}\cdot\mathbf{x}+b) \\
&= \sigma([2.5,-5.0,-1.2,0.5,2.0,0.7]\cdot[3,2,1,3,0,4.19]+0.1) \\
&= \sigma(.833) \\
&= 0.70 \qquad (5.8) \\
p(-|x) = P(y=0|x) &= 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b) \\
&= 0.30
\end{aligned}$$

Processing many examples at once

$$\textbf{foreach} \quad x^{(i)} \quad \text{in input } [x^{(1)}, x^{(2)}, \ldots, x^{(m)}]$$
$$y^{(i)} = \sigma(\mathbf{w} \cdot \mathbf{x^{(i)}} + b)$$

One method is to use for-loop to compute each test example one at a time

$$\begin{aligned} P(y^{(1)} = 1 | x^{(1)}) &= \sigma(\mathbf{w} \cdot \mathbf{x}^{(1)} + b) \\ P(y^{(2)} = 1 | x^{(2)}) &= \sigma(\mathbf{w} \cdot \mathbf{x}^{(2)} + b) \\ P(y^{(3)} = 1 | x^{(3)}) &= \sigma(\mathbf{w} \cdot \mathbf{x}^{(3)} + b) \end{aligned}$$

The computation for the first three test examples

Naïve bayes versus Logistic regression

Loss function

- What do we mean by learning in Machine Learning? How does the learning in logistic regression happen? In other words, how do we learn weights and biases that help us make better predictions (or that minimizes prediction error).

$$L(\hat{y}, y) = \text{How much } \hat{y} \text{ differs from the true } y$$

True label (y)	Predicted label (y-hat)	Should Loss be "high" or "low"?
Positive	Positive	Low
Positive	Negative	High
Negative	Negative	Low
Negative	Positive	High

Cross entropy loss function

When to use cross entropy loss?
When the model outputs predicted probabilities over multiple classes

The binary cross-entropy loss for a single instance is given by:

$$L = -[y\log(\hat{y}) + (1 - y)\log(1 - \hat{y})]$$

Where:

- y is the true label, which can be either 0 or 1.
- $\hat{y}$ is the predicted probability of the positive class (class 1).

For a dataset with multiple instances, the binary cross-entropy loss is averaged over all instances. If we have N instances, the average loss is given by:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i\log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Where:

- N is the number of instances.

Binary cross entropy loss for N instances is the simple average of N individual cross entropy losses for each instance.

Intuitive example
(Cross entropy loss)

- Let us suppose the correct class label is 1 (i.e., y = 1) and the predicted probability that y = 1 is 0.70.

$$\text{Loss} = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$
$$= -1.\log(0.70)$$
$$= 0.36$$

- Let us suppose the correct class label is 0 (i.e., y = 1) and the predicted probability that y = 1 is still 0.70.

$$\text{Loss} = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$
$$= 0.\log(0.70) + 1.\log(1-0.70)$$
$$= 1.\log(0.30)$$
$$= 1.2$$

These calculations show that the binary cross entropy loss for any example is low when the model accurately predicts the class label. And the loss will be high when the model inaccurately predicts the class label.

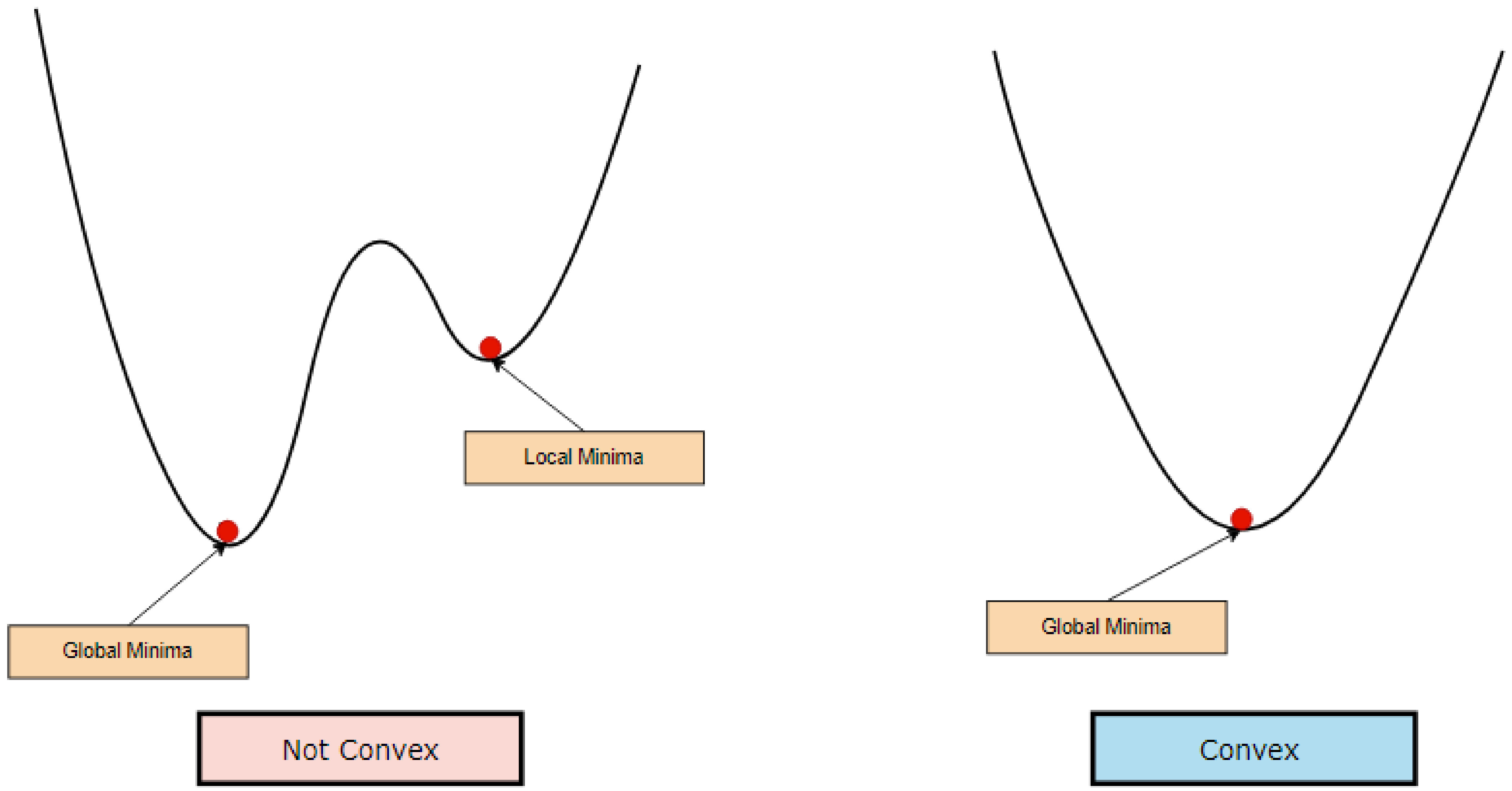
How to minimize the loss function?

- Now we know minimizing the loss function is what we want to do for better classification. The question is how do we minimize the loss function? Gradient descent. Let us suppose we have m training examples.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)};\theta), y^{(i)})$$

$argmin\ \theta$ means that we need set of weights and biases that minimizes the loss function.

- For logistic regression, this loss function is conveniently convex.
- A convex function has at most one minimum; there are no local minima to get stuck in, so gradient descent starting from any point is guaranteed to find the minimum


Local Minima
Global Minima
Not Convex
Global Minima
Convex

Multinomial logistic regression

Multinomial logistic regression vs binary logistic regression

- In binary logistic regression, we are trying to classify an example into one of the two classes. On the other hand, in multinomial logistic regression, we are trying to classify an example into one of the many classes (more than two classes).

- Congratulations! You win a million-dollar lottery.

Spam
Vs
Non-spam

- The movie was fantastic! The plot was engaging, and the acting was top-notch.

Positive
Negative
Neutral

One-hot encoding

Each example will be labeled one category from the set of **[Positive, Negative, Neutral]** categories.

Texts	**Label**	**One-hot representation of label**
"I absolutely loved the new restaurant! The food was delicious and the service was excellent."	Positive	**[1 0 0]**
"The movie was fantastic! The plot was engaging and the acting was top-notch."	Positive	**[1 0 0]**
The movie was a waste of time. The plot was predictable and the acting was terrible."	Negative	**[0 1 0]**
"The weather today is neither too hot nor too cold. It's just an average day."	Neutral	**[0 0 1]**
"The concert was a disaster. The sound was terrible and the band seemed unprepared."	Negative	**[0 1 0]**
"I was very disappointed with the service at the restaurant. The food was cold and the staff were rude."	Negative	**[0 1 0]**

We have one for correct class and zeros otherwise in one-hot representation

Generalization of sigmoid for multinomial logistic regression

- Softmax (which is the generalization of sigmoid) is used to compute vector **y-hat (vector of output probabilities for each of the K classes).** The vector y-hat for each example would contain the probabilities that the example belongs to one of many K classes (for example, the probability that the example belongs to either positive, negative or neutral category in case of three classes.)

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

In multiclass classification problem, we are learning separate weight vectors and bias for each of the K classes.

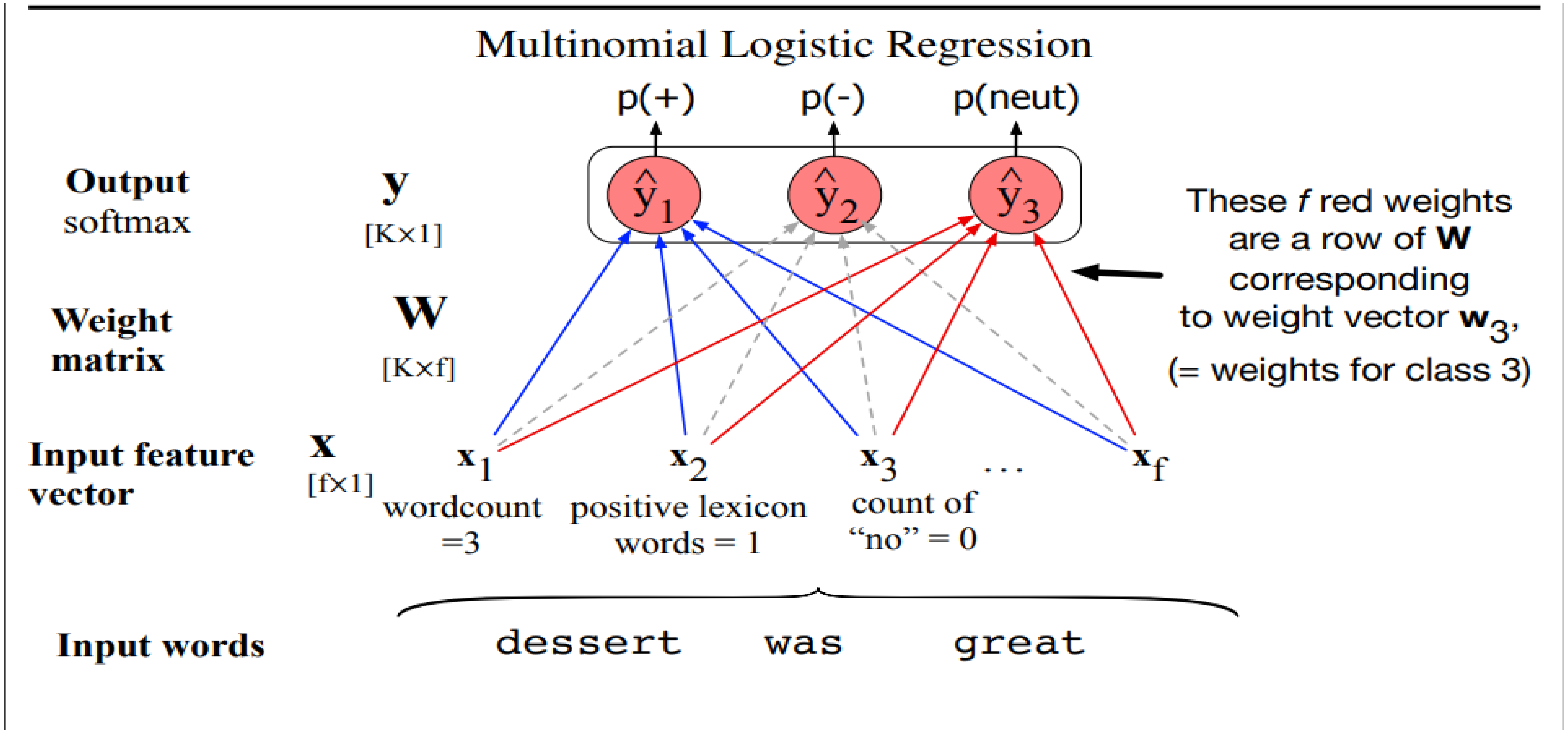
K is the total number of classes out of which we have to assign a class label to each example

Vector of output probabilities (y-hat) for a single example

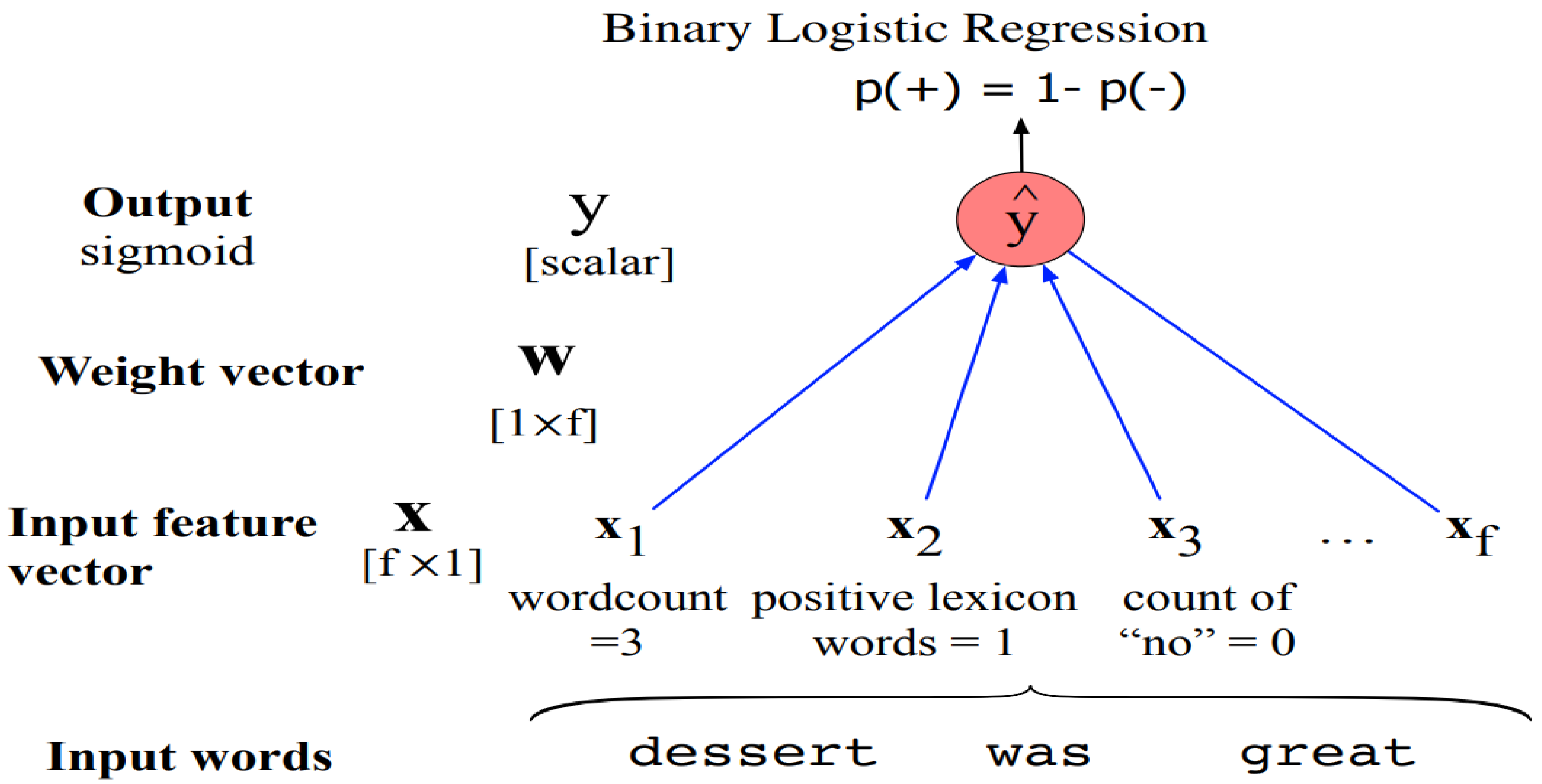
$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

W thus has shape **[K × f]**, for **K** the number of output classes and **f** the number of input features. The bias vector **b** has one value for each of the **K** output classes.

Multinomial logistic regression



Binary Logistic regression



Loss function for multinomial logistic regression

- The loss function for multinomial logistic regression generalizes the loss function for binary logistic regression from 2 to K classes.

$$\begin{aligned}
L_{\mathrm{CE}}(\hat{\mathbf{y}},\mathbf{y}) &= -\sum_{k=1}^{K} y_k \log \hat{y}_k \\
&= -\log \hat{y}_c, \quad \text{(where } c \text{ is the correct class)} \\
&= -\log \hat{p}(y_c = 1|\mathbf{x}) \quad \text{(where } c \text{ is the correct class)} \\
&= -\log \frac{\exp(\mathbf{w_c}\cdot\mathbf{x}+b_c)}{\sum_{j=1}^{K}\exp(\mathbf{w_j}\cdot\mathbf{x}+b_j)} \quad (c \text{ is the correct class})
\end{aligned}$$