Logistic Regression

Erhard Hinrichs

Seminar für Sprachwissenschaft
Eberhard-Karls Universität Tübingen

Comparing Linear Regression to Logistic Regression

	Linear regression	Logistic regression
Activation function:	-	non-linear sigmoid function
Decision Boundary	-	yes
Predictions:	continuous	discrete based on probability scores
Cost function:	mean squared error	cross-entropy
Type of Output:	numeric values	classification labels

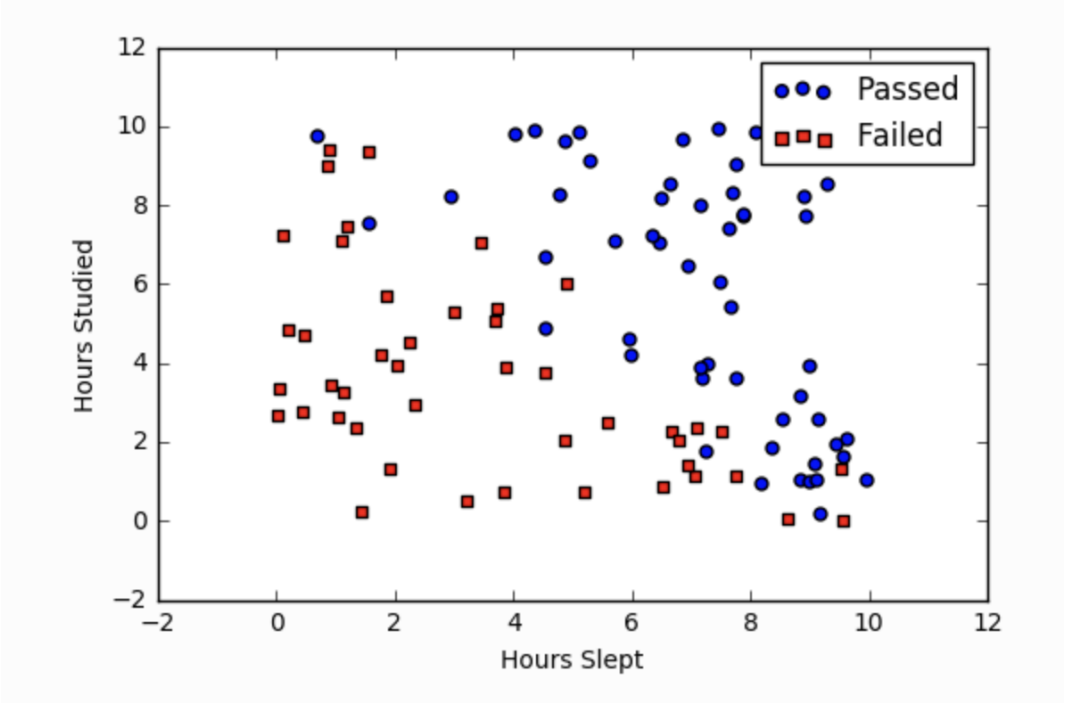
Binary Logistic Regression

Sample Data Set: Student Exam Results

Studied	Slept	Passed
4.85	9.63	1
8.62	3.23	0
5.43	8.23	1
9.21	6.34	0

Example taken from:
ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

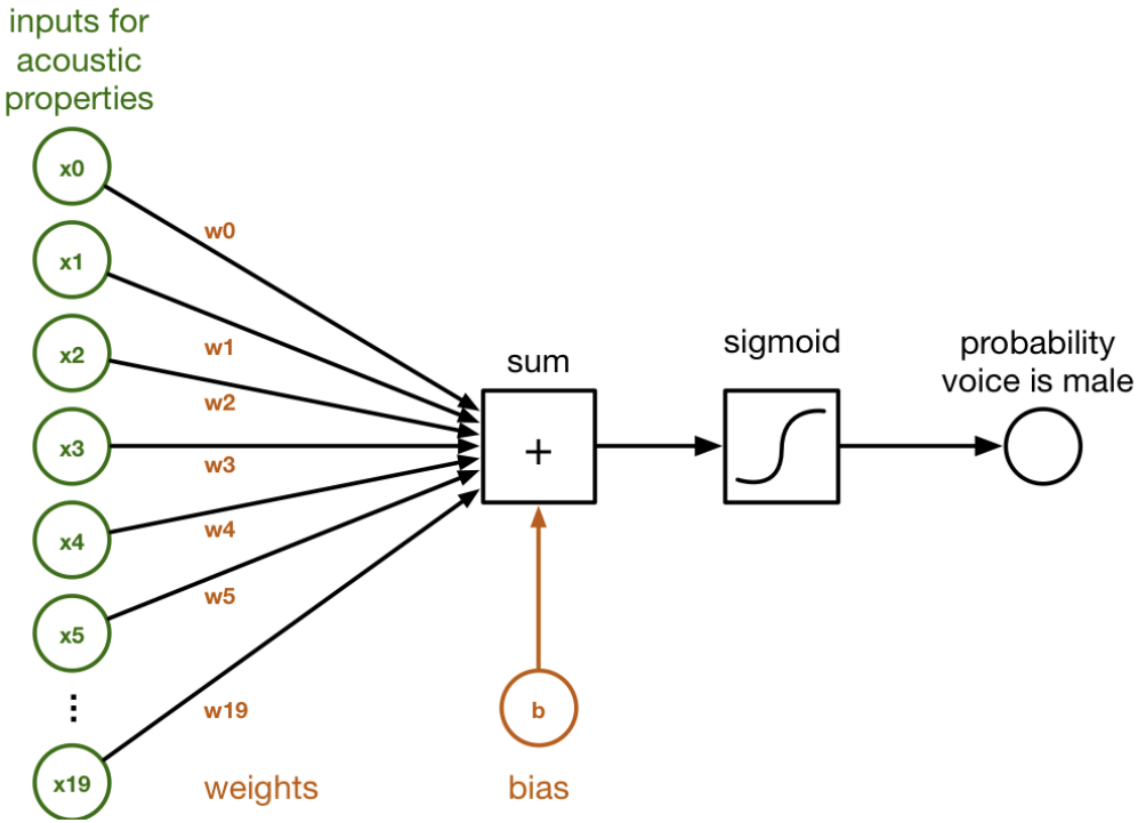
Scatter plot for the data set Student Exam Results



Graphics taken from:

ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

Example: Voice Detection



Graphics taken from: https://pin.it/336eRFM

Sigmoid Activation Function

Logistic regression uses the sigmoid function as its activation function, with the predicted value $mx + b$ serving as input x of the sigmoid function.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

- where e is a constant, the base of the natural logarithm, invented by Leonard Euler in the 18th century.
- $$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} \cdots$$
- e = 2,71828 18284 59045 33536 02874 71352 66249 77572 47093 69995 ...

Important Properties of the Sigmoid Activation Function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

- $0 \leq \sigma(x) \leq 1$.
  - $\lim_{x \to \infty} e^{-x} \to 0$; $\lim_{x \to \infty} \sigma(x) \to 1$
  - $\lim_{x \to -\infty} e^{-x} \to \infty$; $\lim_{x \to -\infty} \sigma(x) \to 0$
- $1 - \sigma(x) = \sigma(-x)$
- $\sigma(x) + \sigma(-x) = 1$
  - For a proof see Slides 9 and 10.

Sigmoid Function: Some Mathematical Background

Let's denote the sigmoid function as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

The derivative of the sigmoid is

$$\frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x)) \tag{4}$$

Sigmoid Function as a Probability Distribution (1)

$$
\begin{aligned}
P(y=1) &= \sigma(\mathbf{w}\cdot\mathbf{x}+b) \\
&= \frac{1}{1+exp(-\mathbf{w}\cdot\mathbf{x}+b)}
\end{aligned} \tag{5}
$$

$$
\begin{aligned}
P(y=0) &= 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b) \\
&= 1-\frac{1}{1+exp(-\mathbf{w}\cdot\mathbf{x}+b)} \\
&= \frac{exp(-\mathbf{w}\cdot\mathbf{x}+b)}{1+exp(-\mathbf{w}\cdot\mathbf{x}+b)}
\end{aligned} \tag{6}
$$

Sigmoid Function as a Probability Distribution (2)

$$
\begin{aligned}
P(y=1)+P(y=0) &= \frac{1}{1+exp(-\mathbf{w}\cdot\mathbf{x}+b)} + \frac{exp(-\mathbf{w}\cdot\mathbf{x}+b)}{1+exp(-\mathbf{w}\cdot\mathbf{x}+b)} \\
&= \frac{1+exp(-\mathbf{w}\cdot\mathbf{x}+b)}{1+exp(-\mathbf{w}\cdot\mathbf{x}+b)} \\
&= 1
\end{aligned}
\tag{7}
$$

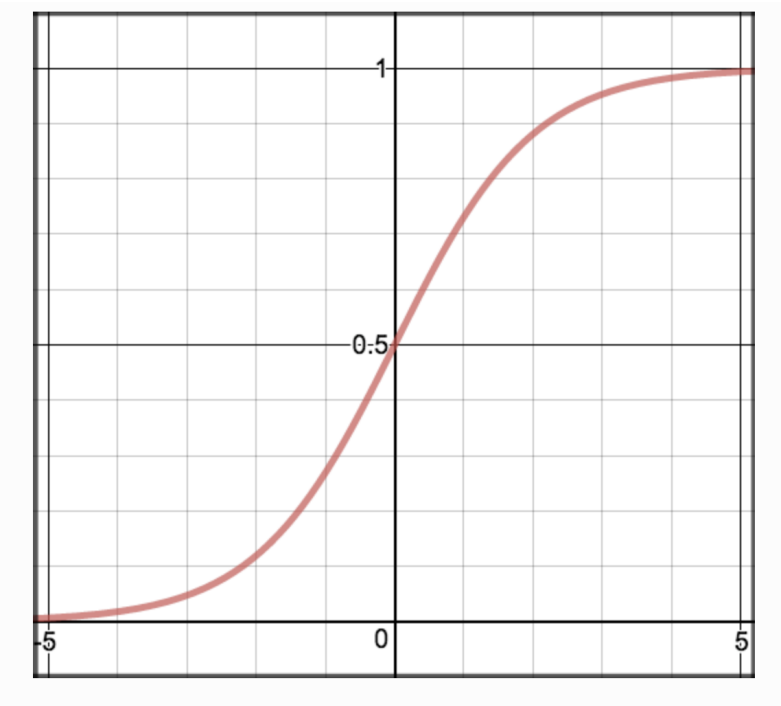
Calculating the derivative of the Sigmoid (1)

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}(1+e^{-x})^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right)
\end{aligned}
\tag{8}
$$

Calculating the derivative of the Sigmoid (2)

$$
\begin{aligned}
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}} \right) \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}} \right) \\
&= \sigma(x) \cdot (1 - \sigma(x))
\end{aligned}
\tag{9}
$$

The Sigmoid Function

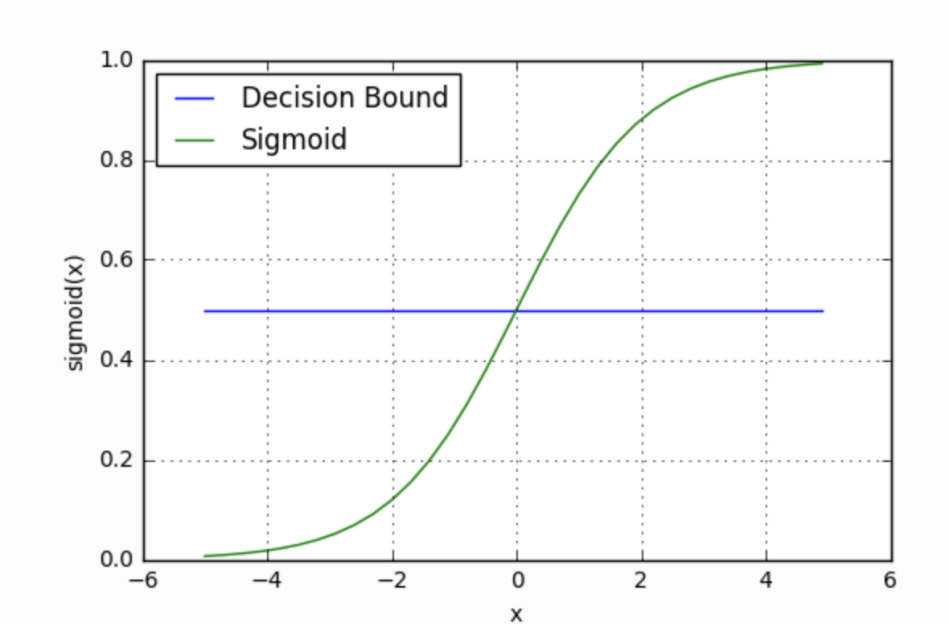


Graphics taken from:

ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

The Decision Boundary

$p \geq .5$, class 1
$p < .5$, class 0



Graphics taken from:
ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

Two Phases of Logistic Regression Modeling

- Training: Optimizing weights w and b using gradient descent and cross-entropy loss.
- Testing: Given a test example x, the conditional probability $p(y|x)$ is computed, using learned the weights w and b, and the label (y = 1 or y = 0) with the higher probability is used for classification.

Example

Extracting Features

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 .

Word count = 64, ln(64) = 4.15

Var	Definition	Value
x_1	Count of positive lexicon words	3
x_2	Count of negative lexicon words	2
x_3	Does no appear? (binary feature)	1
x_4	Number of 1st and 2nd person pronouns	3
x_5	Does ! appear? (binary feature)	0
x_6	Log of the word count for the document	4.15

Slide due to: https://computational-linguistics-class.org/slides/03-logistic-regression.pdf

Example

Var	Definition	Value	Weight	Product
x_1	Count of positive lexicon words	3	2.5	
x_2	Count of negative lexicon words	2	-5.0	
x_3	Does no appear? (binary feature)	1	-1.2	
x_4	Num 1st and 2nd person pronouns	3	0.5	
x_5	Does ! appear? (binary feature)	0	2.0	
x_6	Log of the word count for the doc	4.15	0.7	
b	bias	1	0.1	

$$z = \sum_i w_i x_i + b$$

Slide due to: https://computational-linguistics-class.org/slides/03-logistic-regression.pdf

Example

Computing Z

Var	Definition	Value	Weight	Product
x_1	Count of positive lexicon words	3	2.5	7.5
x_2	Count of negative lexicon words	2	-5.0	-10
x_3	Does no appear? (binary feature)	1	-1.2	-1.2
x_4	Num 1st and 2nd person pronouns	3	0.5	1.5
x_5	Does ! appear? (binary feature)	0	2.0	0
x_6	Log of the word count for the doc	4.15	0.7	2.905
b	bias	1	0.1	.1

$$z = \sum_i w_i x_i + b$$

z=0.805

Slide due to: https://computational-linguistics-class.org/slides/03-logistic-regression.pdf

Example

Sigmoid(Z)

Var	Definition	Value	Weight	Product
x_1	Count of positive lexicon words	3	2.5	7.5
x_2	Count of negative lexicon words	2	-5.0	-10
x_3	Does no appear? (binary feature)	1	-1.2	-1.2
x_4	Num 1st and 2nd person pronouns	3	0.5	1.5
x_5	Does ! appear? (binary feature)	0	2.0	0
x_6	Log of the		0.7	2.905
b	bias		0.1	.1

$\sigma(0.805)$
$= 0.69$

Slide due to: https://computational-linguistics-class.org/slides/03-logistic-regression.pdf

Calculating $p(+|x)$ and $p(-|x)$

$$\begin{aligned} p(+|x) &= P(y = 1 \mid x) = \sigma(w \cdot x + b) \\ &= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.15] + 0.1) \\ &= \sigma(0.805) = .69 \end{aligned} \tag{10}$$

$$\begin{aligned} p(-|x) &= P(y = 0 \mid x) = 1 - \sigma(w \cdot x + b) \\ &= .31 \end{aligned} \tag{11}$$

Processing many examples at once

Instead of computing test examples one at a time:

$$\begin{gathered}\textbf{foreach } x^{(i)} \text{ in input } [x^{(1)}, x^{(2)}, \ldots, x^{(m)}] \\ y^{(i)} = \sigma(\mathbf{w} \cdot \mathbf{x^{(i)}} + b)\end{gathered} \tag{12}$$

define a single input matrix of shape $[m \times f]$ in combination with a bias vector $\mathbf{b}= [b, b, \ldots, b]$ of size $1 \times m$ and a vector $\hat{\mathbf{y}} = [\hat{y}^1, \hat{y}^2, \ldots, \hat{y}^m]$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^{(1)} & \mathbf{x}_2^{(1)} & \ldots & \mathbf{x}_f^{(1)} \\ \mathbf{x}_1^{(2)} & \mathbf{x}_2^{(2)} & \ldots & \mathbf{x}_f^{(2)} \\ \mathbf{x}_1^{(3)} & \mathbf{x}_2^{(3)} & \ldots & \mathbf{x}_f^{(3)} \\ \ldots & & & \end{bmatrix} \tag{13}$$

... represent where w is as a column vector and replace (12) by:

$$y = \mathbf{X}\mathbf{w} + \mathbf{b} \tag{14}$$

Another Example: Period Disambiguation

- Binary classification task: Classifying each period in a text into one of two classes: EOS (end-of-sentence) and not-EOS
- Examples:
  - *I swear.* . $\rightarrow$ EOS
  - *Is Main St. east of the train station?* . $\rightarrow$ not-EOS
- Important pre-processing step for sentence splitting and for named-entity recognition

Period Disambiguation: Designing Discriminative Features

$$x_1 = \begin{cases} 1 & \text{if } \text{“}Case(w_i) = Lower\text{”}, \\ 0, & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1, & \text{if } \text{“}w_i \in AcronymDict\text{”}, \\ 0, & \text{otherwise} \end{cases}$$

$$x_3 = \begin{cases} 1, & \text{if } \text{“}w_i = St. \ \& \ Case(w_{i-1}) = Cap\text{”} \\ 0, & \text{otherwise} \end{cases}$$

Cross-Entropy

- The **entropy** of a random variable X measures the uncertainty inherent in the possible outcomes of the variable X

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot log\ P(x_i) \tag{15}$$

- The **cross-entropy** between two probability distributions p and q is a measure of the difference between two probability distributions for a given random variable or set of events.

The cross-entropy

$$H(p, q) = -\sum_{x \in \chi} p(x) \cdot log\ q(x) \tag{16}$$

Conditional Maximum Likelihood Estimation for the Loss Function

Goal: For an observation x, define a loss function $L(\hat{y}, y)$ that measures the difference between the classifier output $\hat{y}$ and the correct output y and that maximizes the log probability of the true labels in the training data.

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

Strategy: Given an observation x, choose the parameters w, b that maximize the probability of the correct label $p(y|x)$. Since there are only two possible outcomes in a binary classification task, these conditional probabilities have a Bernoulli distribution.

$$p(y|x) = \hat{y}^y(1-\hat{y})^{1-y} \tag{17}$$

If $y = 1$, Eq. 17 simplifies to $\hat{y}$. If $y = 0$, Eq. 17 simplifies to $1 - \hat{y}$.

Deriving the Cross-Entropy Loss Function

Taking the log of both sides of the equation in Eq. 17 yields:

$$\begin{aligned} \log p(y|x) &= \log[\hat{y}^y(1-\hat{y})^{1-y}] \\ &= y \log \hat{y} + (1-y)\log(1-\hat{y}) \end{aligned} \tag{18}$$

In order to turn Eq. 18 into the cross-entropy loss function L_{CE}, the sign needs to be changed to minus:

$$\begin{aligned} L_{CE}(\hat{y}, y) &= -\log p(y|x) \\ &= -[y \log \hat{y} + (1-y)\log(1-\hat{y})] \end{aligned} \tag{19}$$

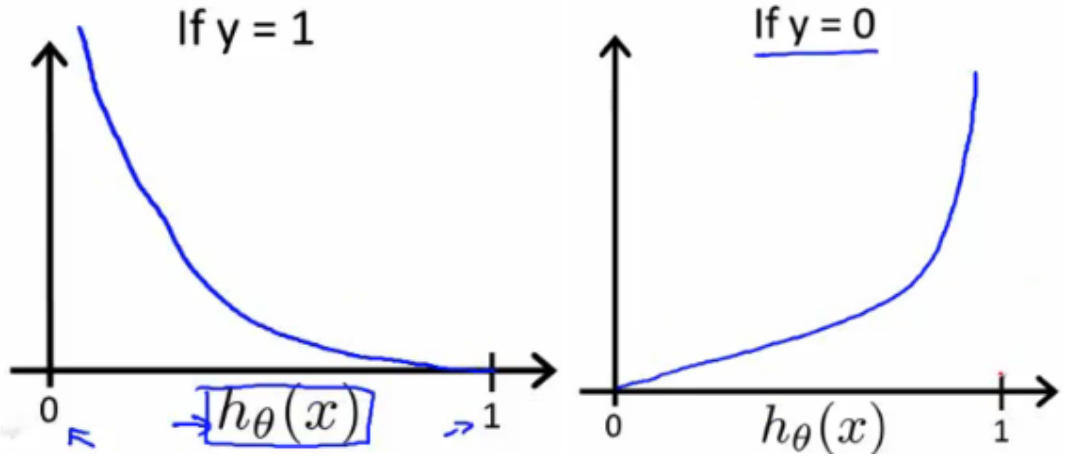
$$L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1-y)\log(1-\sigma(w \cdot x + b))] \tag{20}$$

The cross-entropy loss function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \tag{21}$$

$$Cost(h_\theta(x), y) = -log(h_\theta(x)) \qquad \text{if y} = 1$$

$$Cost(h_\theta(x), y) = -log(1 - h_\theta(x)) \qquad \text{if y} = 0$$



Cross-Entropy Loss – Example

Cross-entropy loss

Why does minimizing this negative log probability do what we want? We want the **loss** to be **smaller** if the model's estimate is **close to correct**, and we want the **loss** to be **bigger if it is confused.**

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 .

P(sentiment=1 | It's hokey...) = 0.69. Let's say y=1.

$$L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))]$$

$$= -[\log \sigma(w \cdot x + b)]$$

$$= -\log(0.69) = \mathbf{0.37}$$

Slide due to: https://computational-linguistics-class.org/slides/03-logistic-regression.pdf

Cross-Entropy Loss – Example

Cross-entropy loss

Why does minimizing this negative log probability do what we want? We want the **loss** to be **smaller** if the model's estimate is **close to correct**, and we want the **loss** to be **bigger if it is confused.**

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 .

P(sentiment=1 | It's hokey...) = 0.69. Let's **pretend** y=0.

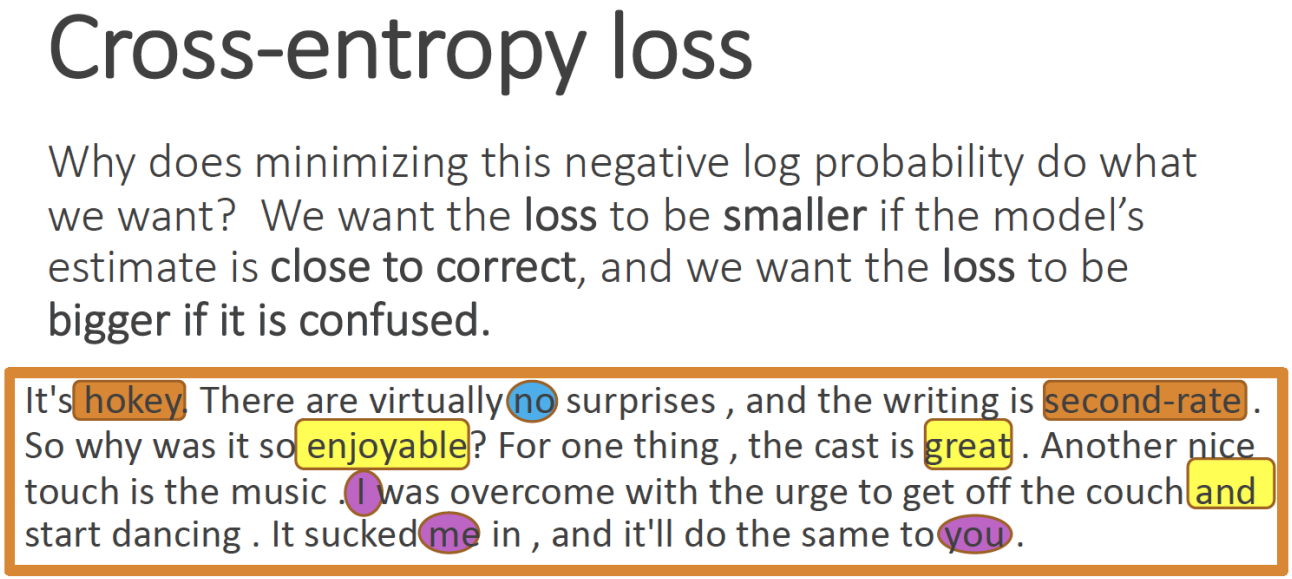
$$L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))]$$

$$= -[\log(1 - \sigma(w \cdot x + b))]$$

$$= -\log(0.31) = \mathbf{1.17}$$

Slide due to: https://computational-linguistics-class.org/slides/03-logistic-regression.pdf

Cross-Entropy Loss – Example



If our prediction is **correct**, then our CE loss is **lower**

$$= -\log(0.69) = \mathbf{0.37}$$

If our prediction is **incorrect**, then our CE loss is **higher**

$$-\log(0.31) = \mathbf{1.17}$$

Slide due to: https://computational-linguistics-class.org/slides/03-logistic-regression.pdf

Applying the cross-entropy loss function to one training data instance: an example

$$\begin{aligned} L_{CE}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))] \\ &= -[\log \sigma(w \cdot x + b)] \\ &= -\log(.69) \\ &= .37 \end{aligned} \tag{22}$$

$$\begin{aligned} L_{CE}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log(1 - \sigma(w \cdot x + b))] \\ &= -[\log(1 - \sigma(w \cdot x + b))] \\ &= -\log(.31) \\ &= 1.17 \end{aligned} \tag{23}$$

Multinomial logistic regression

- multi-nominal regression, also called *softmax regression* or *maxent classifier*, is used when more than two labels are needed for classification, that is:
  - when we want to label the target variable **y** of each data instance x^i with a unique label **k**, taken from a set of **K** classes.
  - We can represent the correct target value by a **one-hot vector** that assigns the value 1 to the correct class and the value 0 to all other classes.
  - For each prediction $\hat{y}$, we can claculate the probability for each class $\mathbf{k} \in \mathbf{K}$ classes, using the **softmax** function.

$$softmax(z_i) = \frac{exp(z_i)}{\sum_{j=1}^{k} exp(z_j)} \quad 1 \leq i \leq k \tag{24}$$

Softmax

- The softmax function is a generalization of the sigmoid.
- The softmax of an input vector $z = [z_1, z_2, ..., z_k]$ is a vector itself:

$$softmax(z) = \left[\frac{exp(z_1)}{\sum_{i=1}^{k} exp(z_i)}, \frac{exp(z_2)}{\sum_{i=1}^{k} exp(z_i)}, \dots \frac{exp(z_k)}{\sum_{i=1}^{k} exp(z_i)} \right] \tag{25}$$

- The denominator $\sum_{i=1}^{k} exp(z_i)$ normalizes all the values into probabilities.

Softmax: An Example

Given a vector $\mathbf{z}$

$$\mathbf{z} = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

the rounded softmax$(\mathbf{z})$ is:

$$\mathbf{z} = [0.055, 0.090, 0.006, 0.099, 0.74, 0.010]$$

with:

$exp(.6) = 1.822119, exp(1.1) = 3.004166, exp(-1.5) = 0.2231302$
$exp(1.2) = 3.320117, exp(3.2) = 24.53253, exp(-1.1) = 0.3328711$

$denom = exp(0.6) + exp(1.1) + exp(-1.5) + exp(1.2) + exp(3.2) +$
$exp(-1.1) = 33.23493$

$\frac{exp(.6)}{33.23493} = 0.055, \frac{exp(1.1)}{33.23493} = 0.090, \frac{exp(-1.5)}{33.23493} = 0.006, \ldots$

Applying Softmax in Logistic Regression

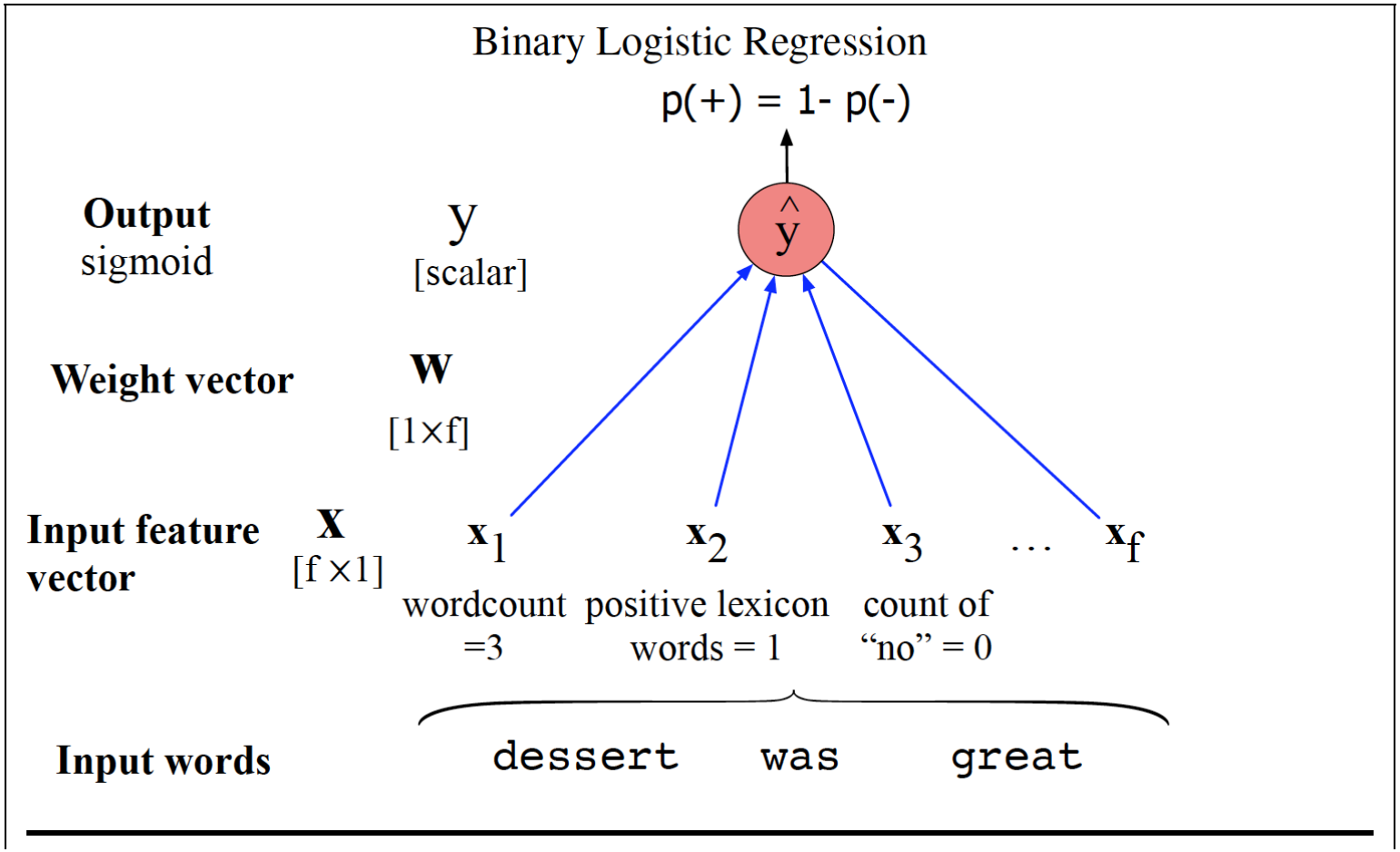
The probability of each output class $\hat{y}_k$ can be computed as:

$$p(\mathbf{y}_k = 1|x) = \frac{exp(\mathbf{w}_k \cdot \mathbf{x} + \mathbf{b}_k)}{\sum_{j=1}^{k} exp(\mathbf{w}_j \cdot \mathbf{x} + b_j)} \tag{26}$$

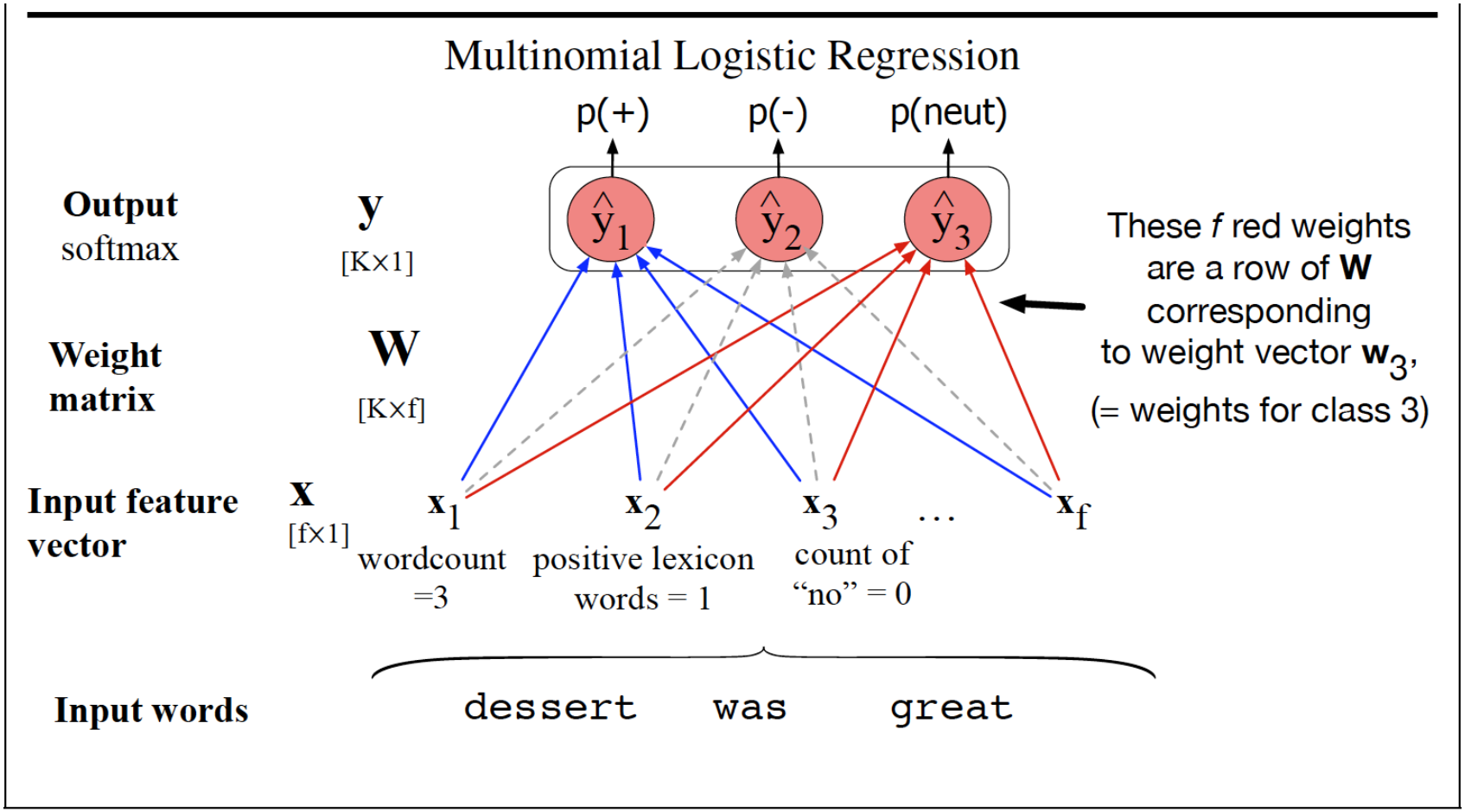
The vector $\hat{\mathbf{y}}$ of output probabilities for each of the K classes can be computed by:

$$\hat{\mathbf{y}} = softmax(\mathbf{X}\mathbf{w} + \mathbf{b}) \tag{27}$$

Binary Logistic Regression



Multinomial Logistic Regression



Learning in Logistic Regression

- How are the parameters $\theta = [\mathbf{w}, b]$ learnt (i.e. optimized)?
- This requires two components:
  - the **cross-entropy loss function** (see above), which provides a distance measure between the system's output (aka: predictions, hypotheses) $\hat{y}$ and the gold output y.
  - an **optimization algorithm (gradient descent)** for updating the weight and bias parameters

Gradient Descent for Binary Logistic Regr.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(f(x^{(i)}; \theta), y^{(i)}) \tag{28}$$

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x; w), y) \tag{29}$$

$$\nabla_\theta L(f(x;\theta), y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta), y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta), y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta), y) \\ \frac{\partial}{\partial b} L(f(x;\theta), y) \end{bmatrix}$$

The final equation for updating θ based on the gradient is thus

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x;\theta), y) \tag{30}$$

The Gradient for Binary Logistic Regression and Updating θ

The final equation for updating θ based on the gradient is thus

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x;\theta), y) \tag{31}$$

One Step of Gradient Descent

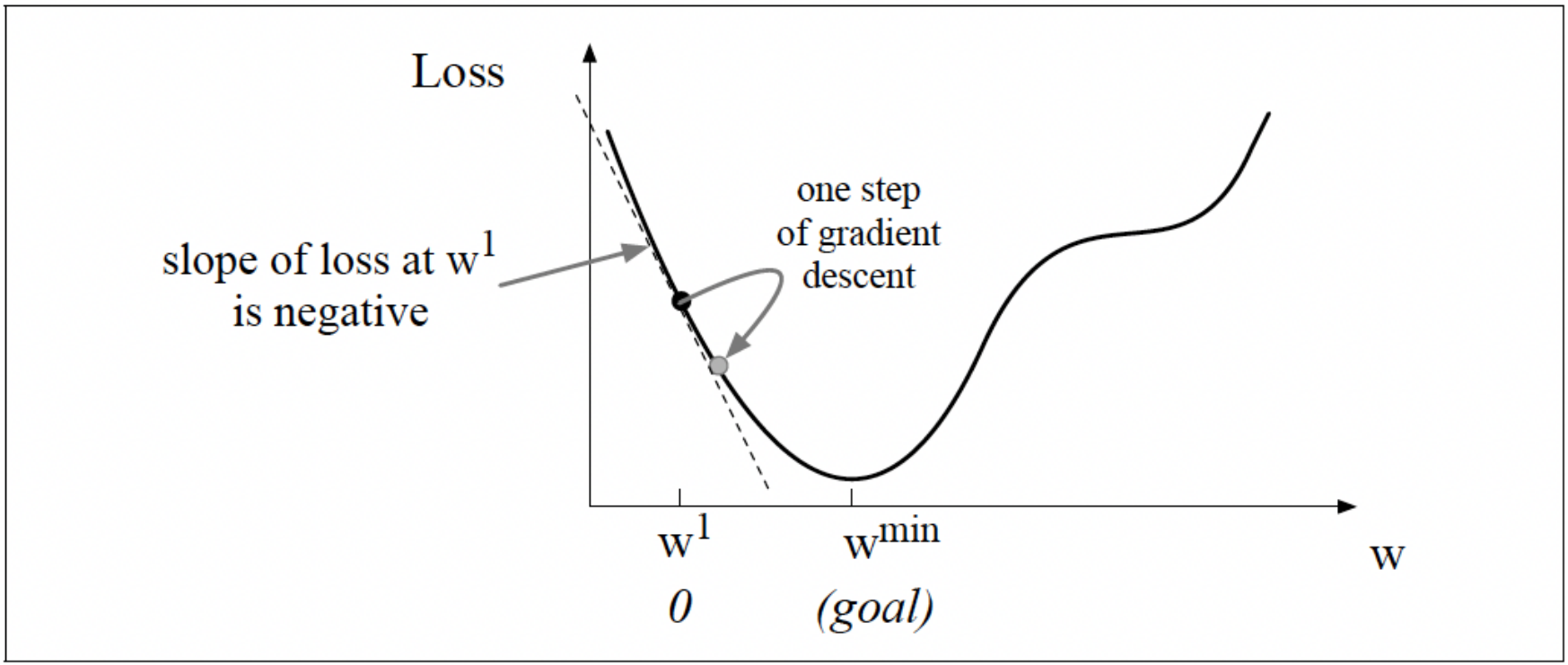


Figure 5.4 The first step in iteratively finding the minimum of this loss function, by moving w in the reverse direction from the slope of the function. Since the slope is negative, we need to move w in a positive direction, to the right. Here superscripts are used for learning steps, so w^1 means the initial value of w (which is 0), w^2 the value at the second step, and so on.

Visualization of Gradient Vector

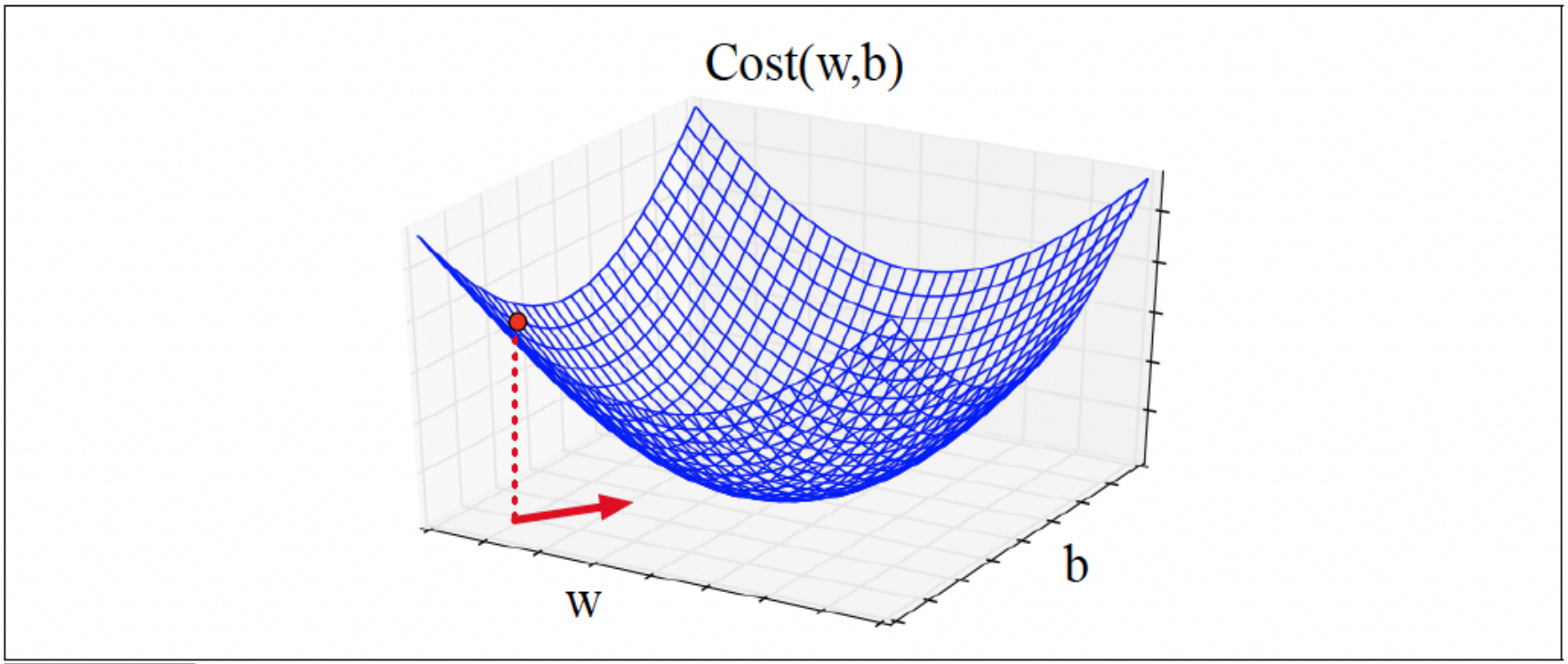


Figure 5.5 Visualization of the gradient vector at the red point in two dimensions w and b, showing a red arrow in the x-y plane pointing in the direction we will go to look for the minimum: the opposite direction of the gradient (recall that the gradient points in the direction of increase not decrease).

Computing the Gradient of the Cross-Entropy Loss Function (1)

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

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

$$z = w_0 + x_1 * w_1 + x_2 * w_2 + \ldots x_n * w_n = W^T X \tag{3}$$

$$ChainRule: \quad \frac{\partial L(W)}{\partial W} = \frac{\partial L(W)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial W} \tag{4}$$

$$\frac{\partial L(W)}{\partial \hat{y}} = -(\frac{y}{\hat{y}} - \frac{1 - y}{1 - \hat{y}}) = \frac{-y(1 - \hat{y}) + (1 - y)\hat{y}}{\hat{y}(1 - \hat{y})} = \frac{\hat{y} - y}{\hat{y}(1 - \hat{y})} \tag{5}$$

Computing the Gradient of the Cross-Entropy Loss Function (2)

$$
\begin{aligned}
\frac{\partial \hat{y}}{\partial z} &= \frac{\partial}{\partial z}[\frac{1}{1+e^{z}}] = \frac{\partial}{\partial z}(1+e^{-z})^{-1} \\
&= -(1+e^{-z})^{-2}(-e^{-z}) = \frac{e^{-z}}{(1+e^{-z})^{2}} \\
&= \frac{1}{(1+e^{-z})}\frac{e^{-z}}{(1+e^{-z})} = \frac{1}{1+e^{-z}}\frac{(1+e^{-z})-1}{(1+e^{-z})} \qquad (6) \\
&= \frac{1}{1+e^{-z}}(\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}) \\
&= \frac{1}{1+e^{-z}}(1 - \frac{1}{1+e^{-z}}) = \hat{y}(1-\hat{y})
\end{aligned}
$$

Computing the Gradient of the Cross-Entropy Loss Function (3)

Chain Rule (4) repeated:

$$\frac{\partial L(W)}{\partial W} = \frac{\partial L(W)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial W} \tag{4}$$

$$\frac{\partial z}{\partial W} = X \tag{7}$$

$$\frac{\partial L(W)}{\partial W} = \frac{\hat{y} - y}{\hat{y}(1-\hat{y})} \hat{y}(1-\hat{y})X = (\hat{y} - y)X \tag{8}$$

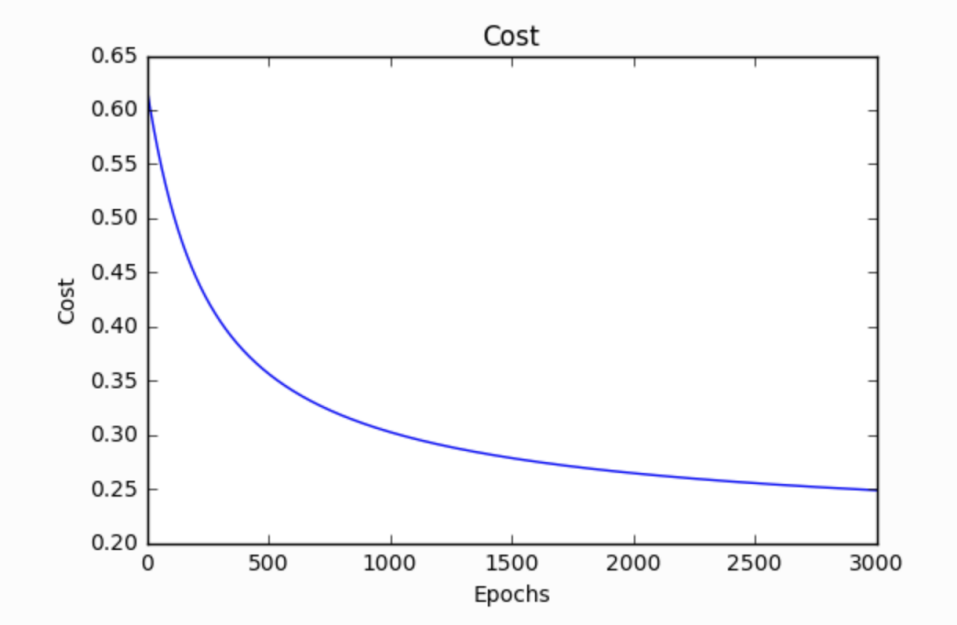
Model Evaluation

If the model is working properly, the cost should go down after every iteration.

For example:
iter: 0 cost: 0.635
iter: 1000 cost: 0.302
iter: 2000 cost: 0.264
Final cost: 0.2487. Final weights: [-8.197, .921, .738]

Example taken from:
ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

Cost History



Working through an example

- ... of a single step θ_{t+1} of the gradient descent algorithm, using a simplified version of the sentimental analysis example (see slides 16 ff. above)

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x;\theta), y) \tag{32}$$

- for a single observation x with value $y = 1$ and a feature vector $x = [x_1, x_2]$ for two features:

$$\begin{aligned} x_1 &= 3 && \text{(count of positive lexicon words)} \\ x_2 &= 2 && \text{(count of negative lexicon words)} \end{aligned}$$

- with initial weights $\mathbf{w_1} = \mathbf{w_2} = b = 0$ and learning rate $\eta = 0.1$.

Working through an example

$$\nabla_{(w,b)} = \begin{bmatrix} \frac{\partial L_{CE}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{CE}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{CE}(\hat{y},y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix} \tag{33}$$

$$= \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

$$\theta^{t+1} = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} .15 \\ .1 \\ .05 \end{bmatrix} \tag{34}$$

Cost Function for Multinomial Logistic Regression

$$L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1-y)\log(1-\hat{y})] \quad (35)$$

$$\begin{aligned} L_{CE}(\hat{y}, y) &= -\sum_{k=1}^{K} y_k \log \hat{y}_k \\ &= -\log \hat{y}_c, \quad \text{(where c is the correct class)} \\ &= -\log \hat{p}(y_c = 1|x), \quad \text{(where c is the correct class)} \\ &= -\log \frac{exp(w_c \cdot x + b_c)}{\sum_{j=1}^{K} exp(w_j \cdot x + b_j)}, \quad \text{(where c is the correct class)} \end{aligned} \quad (36)$$

Gradient in Multinomial Logistic Regression

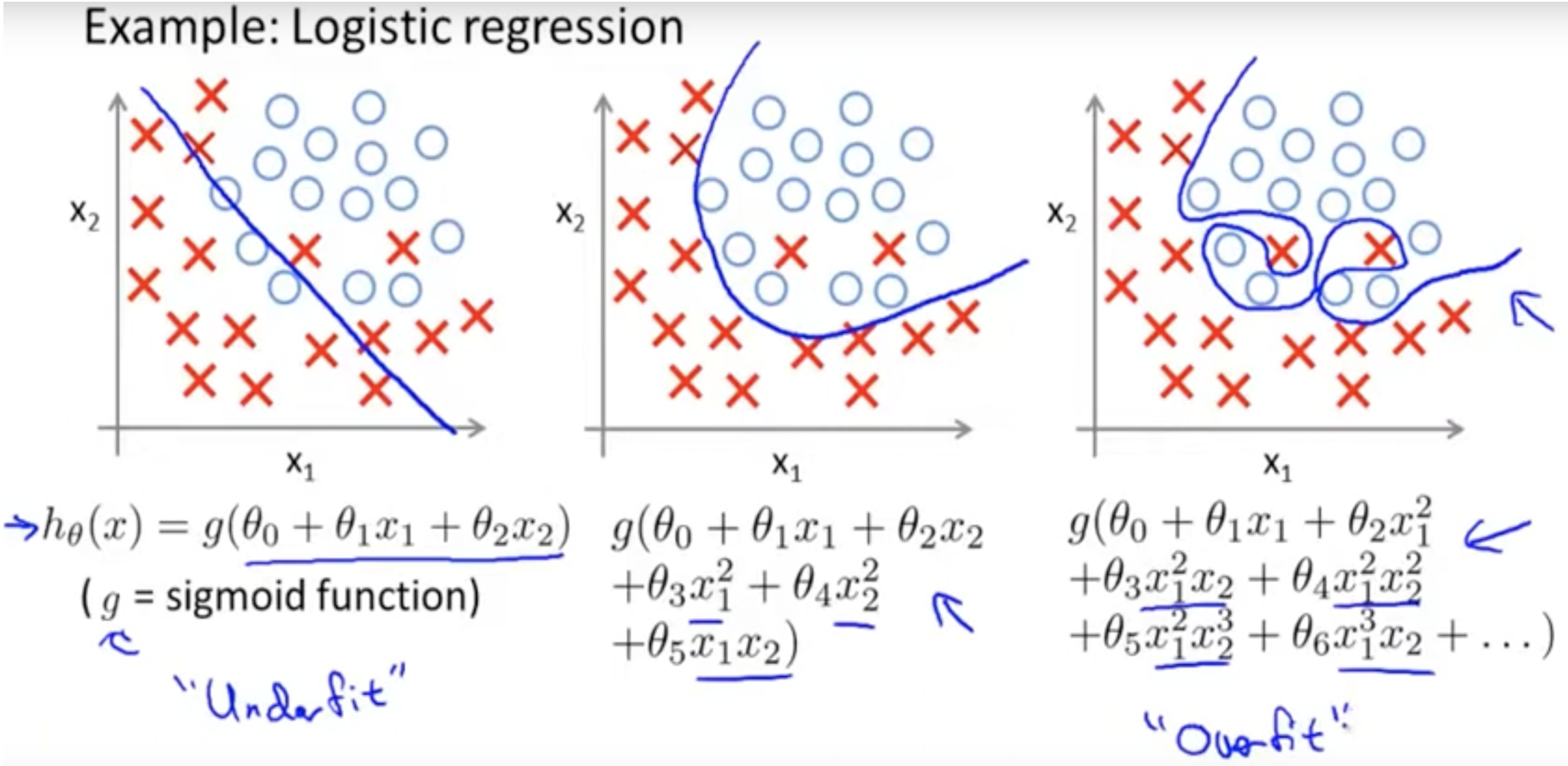
$$
\begin{aligned}
\frac{\partial L_{CE}}{\partial w_{k,i}} &= -(y_k - \hat{y}_k)x_i \\
&= -(y_k - p(y_k = 1|x))x_i \\
&= -\left(y_k - \frac{exp(w_k \cdot x + b_k)}{\sum_{j=1}^{K} exp(w_j \cdot x + b_j)}\right) x_i
\end{aligned}
\tag{37}
$$

Overfitting and how to Deal with it

- **Overfitting** means that the parameters of a learning model are too closely optimized for a particular set of training data.
  - For example: due to the particular distribution of labels in the training data, features that are highly predictive of a particular class label will receive high feature weights.
  - However, such skewed distributions may not generalize to other data sets!
- Overfitting is a general issue for machine learning models – by no means restricted to logistic regression models
- For logistic regression models overfitting can be alleviated by **regularization** of the model parameters

Overfitting: an Illustration

Source: Andrew Ng;
https://www.youtube.com/watch?v=QjOILAQ0EFg



L2 Regularization (also called: Ridge Regularization)

- Adds a penalty term to the objective function equal to the square of the feature weights (coefficients).
- The penalty term uses
  - a regularization hyperparameter α, sometimes also identified by the symbol λ
  - the squared L2 norm: $||\theta||_2^2$
- Leads to a model where all coefficients are close to zero since the derivative of the squared L2 norm subtracts 2θ as a penalty.

Regularization

L2 Regularization:

$$\hat{\theta} = \underset{\theta}{argmax} \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) - \alpha R(\theta) \tag{38}$$

$$R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{n} \theta_j^2 \tag{39}$$

The L2 regularized objective function becomes:

$$\hat{\theta} = \underset{\theta}{argmax} \left[\sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} \theta_j^2 \tag{40}$$

Regularization

L1 Regularization:

$$R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} |\theta_i| \tag{41}$$

The L1 regularized objective function becomes:

$$\hat{\theta} = \underset{\theta}{argmax} \left[\sum_{1=i}^{m} \log P(y^{(i)}|x^{(i)}) \right] - \alpha \sum_{j=1}^{n} |\theta_j| \tag{42}$$