COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Supervised Classification with Logistic Regression

CMSC 470

Marine Carpuat

The Perceptron
What you should know

- What is the underlying function used to make predictions
- Perceptron test algorithm
- Perceptron training algorithm
- How to improve perceptron training with the averaged perceptron
- Fundamental Machine Learning Concepts:
    - train vs. test data; parameter; hyperparameter; generalization; overfitting; underfitting.
- How to define features

Logistic Regression for **Binary** Classification

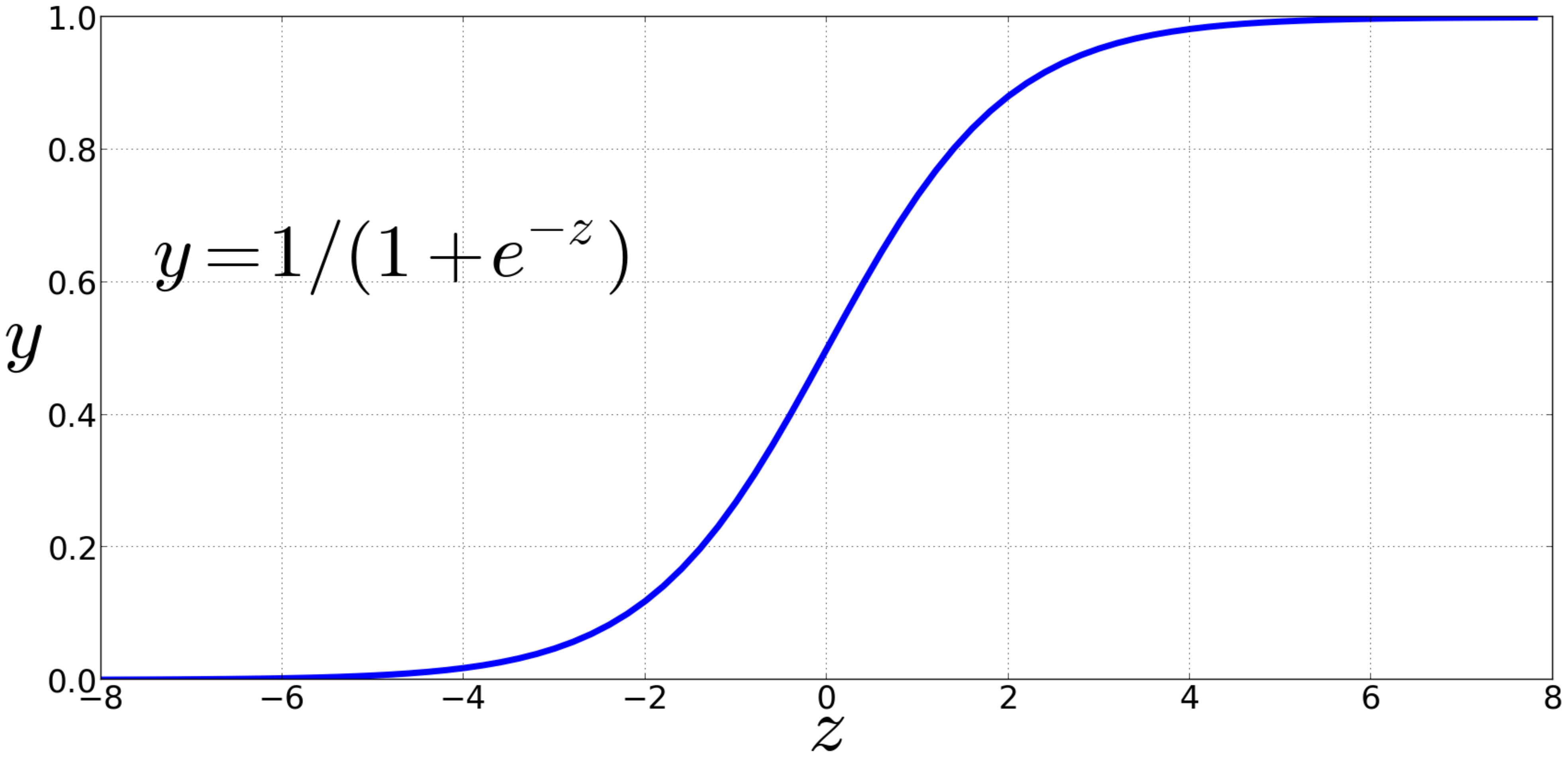
Images and examples: Jurafsky & Martin, SLP 3 Chapter 5

From Perceptron to Probabilities: the Logistic Regression classifier

- The perceptron gives us a prediction y, and the activation can take any real value

- What if we want a probability $p(y|x)$ instead?

The sigmoid function
(aka the logistic function)



From Perceptron to Probabilities for Binary Classification

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

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

Making Predictions with the Logistic Regression Classifier

- Given a test instance x, predict class 1 if P(y=1|x) > 0.5, and 0 otherwise

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

- Inputs x for which P(y=1|x) = 0.5 constitute the **decision boundary**

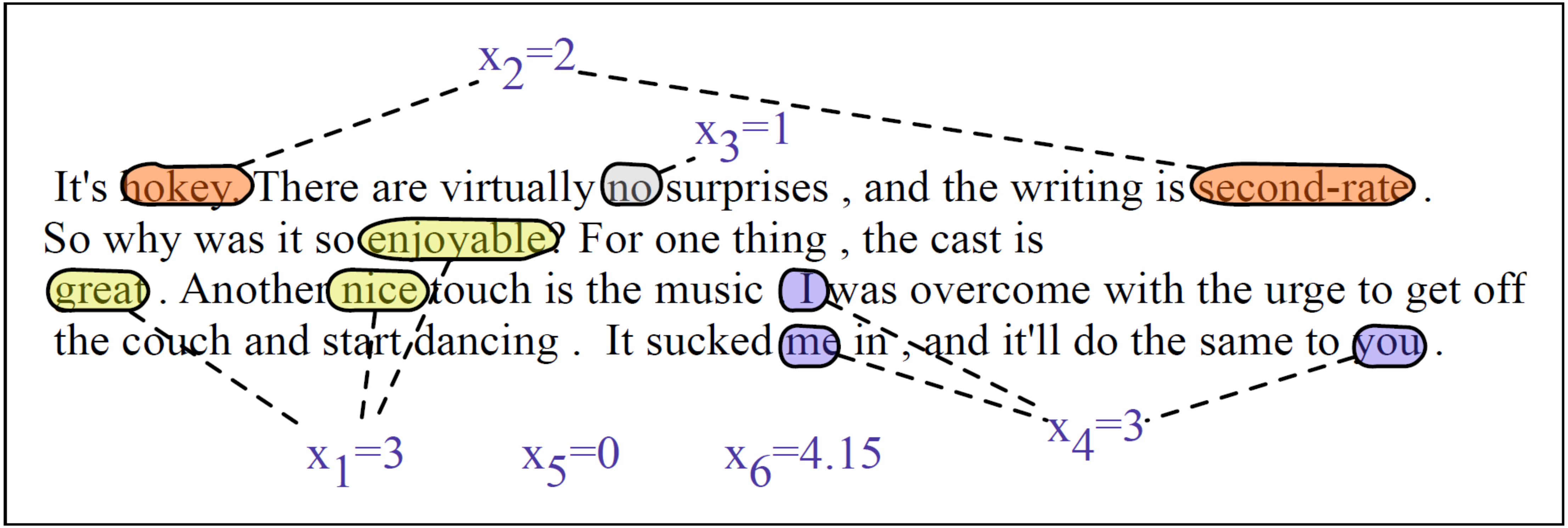
Example: Sentiment Classification with Logistic Regression

- 2 classes: 1 (positive sentiment) or 0 (negative sentiment)
- Examples are movie reviews
- Features:

Var	Definition	Value in Fig. 5.2
x_1	$\text{count(positive lexicon)} \in \text{doc)}$	3
x_2	$\text{count(negative lexicon)} \in \text{doc)}$	2
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	$\text{count}(\text{1st and 2nd pronouns} \in \text{doc})$	3
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	$\log(\text{word count of doc})$	$\ln(64) = 4.15$

Var	Definition	Value in Fig. 5.2
x_1	$\text{count(positive lexicon)} \in \text{doc)}$	3
x_2	$\text{count(negative lexicon)} \in \text{doc)}$	2
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	$\text{count(1st and 2nd pronouns} \in \text{doc)}$	3
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	$\log(\text{word count of doc})$	$\ln(64) = 4.15$

Constructing the feature vector x for one example



Example: Sentiment Classification with Logistic Regression

- Assume we are given the parameters of the classifier

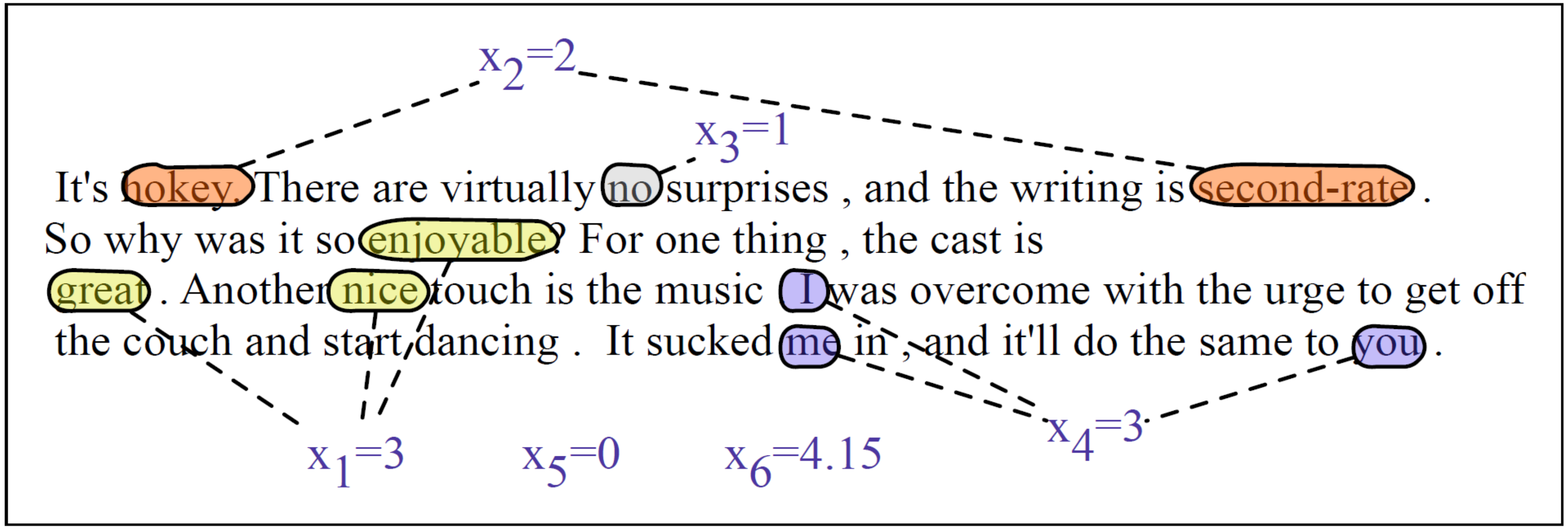
w = $[2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$

b = 0.1

- On this example:

$P(y=1|x) = 0.69$

$P(y=0|x) = 0.31$



Learning in Logistic Regression

- How are parameters of the model (w and b) learned?

- This is an instance of supervised learning
  - We have labeled training examples

- We want model parameters such that
  - For training examples x
  - The prediction of the model $\hat{y}$
  - is as close as possible to the true y

Learning in Logistic Regression

- How are parameters of the model (w and b) learned?

- This is an instance of supervised learning
  - We have labeled training examples

- We want model parameters such that
  - For training examples x, the prediction of the model $\hat{y}$ is as close as possible to the true y
  - Or equivalently so that the distance between $\hat{y}$ and y is small

Ingredients required for training

- Loss function or cost function
  - A measure of distance between classifier prediction and true label for a given set of parameters

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

- An algorithm to minimize this loss
  - Here we'll introduce stochastic gradient descent

The cross-entropy loss function

- Loss function used for logistic regression and often for neural networks

- Defined as follows:

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

Deriving the cross-entropy loss function

- Conditional maximum likelihood
  - Choose parameters that maximize the log probability of true labels y given inputs x

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

- Cross-entropy loss is defined as

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

Example: Sentiment Classification with Logistic Regression

- Assume we are given the parameters of the classifier

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

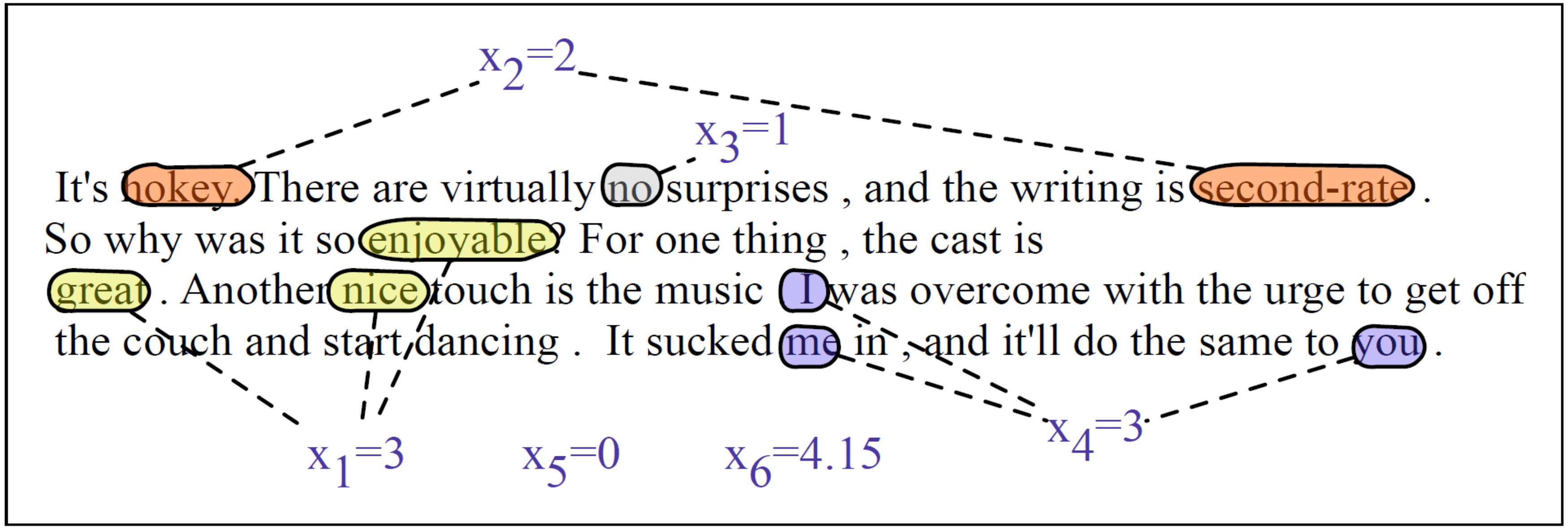
b = 0.1

- On this example:

$$P(y=1|x) = 0.69$$

$$P(y=0|x) = 0.31$$

$$\text{Loss}(w,b) = -\log(0.69) = 0.37$$



Example: Sentiment Classification with Logistic Regression

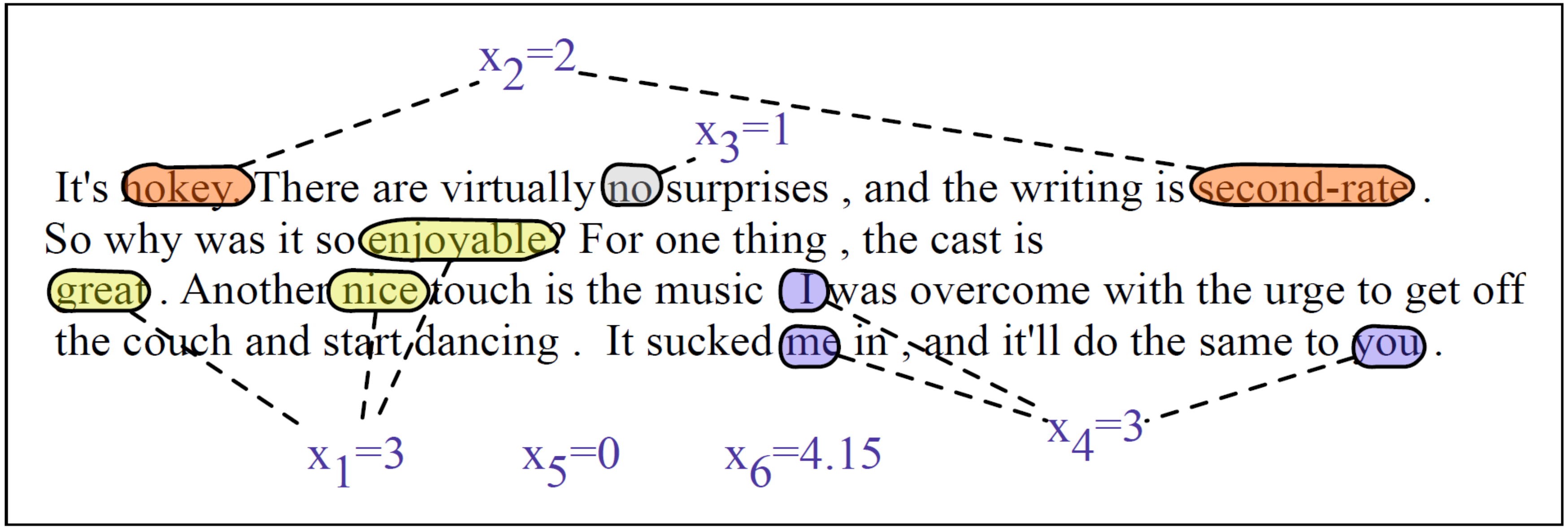
- Assume we are given the parameters of the classifier

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

b = 0.1

- If the example was negative (y=0)

Loss(w,b) = - log(0.31) = 1.17



Gradient Descent

- Goal:
  - find parameters $\theta = w, b$
  - Such that

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

- For logistic regression, the loss is **convex**

Illustrating Gradient Descent

The **gradient** indicates the direction of greatest increase of the cost/loss function.

Gradient descent finds parameters (w,b) that decrease the loss by taking a step in the opposite direction of the gradient.

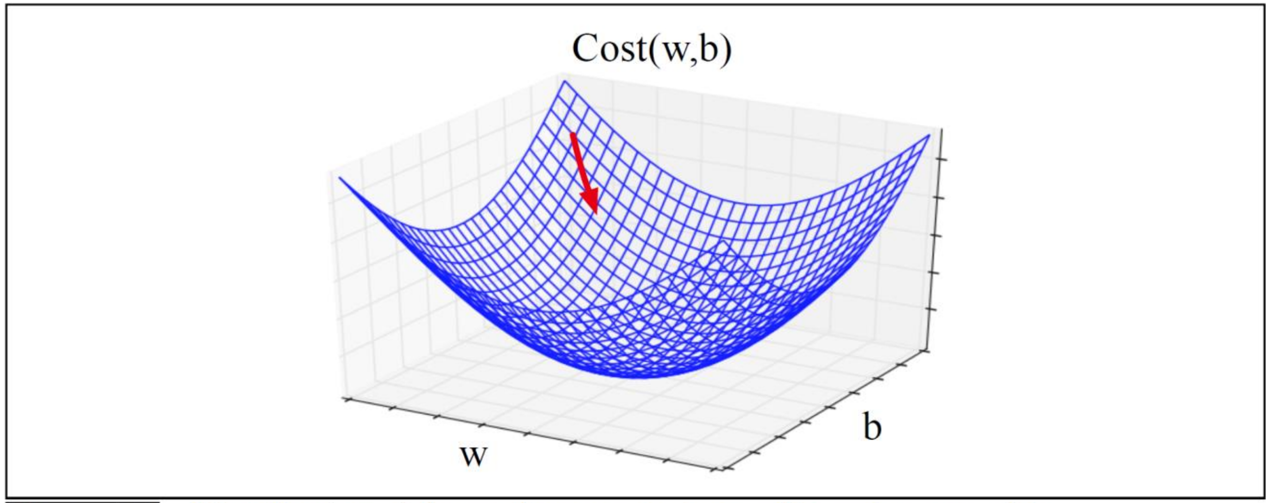


Figure 5.4 Visualization of the gradient vector in two dimensions w and b.

```
function STOCHASTIC GRADIENT DESCENT(L(), f(), x, y) returns θ
    # where: L is the loss function
    #        f is a function parameterized by θ
    #        x is the set of training inputs x^(1), x^(2), ..., x^(n)
    #        y is the set of training outputs (labels) y^(1), y^(2), ..., y^(n)

θ ← 0
repeat til done   # see caption
  For each training tuple (x^(i), y^(i)) (in random order)
     1. Optional (for reporting):          # How are we doing on this tuple?
        Compute ŷ^(i) = f(x^(i); θ)       # What is our estimated output ŷ?
        Compute the loss L(ŷ^(i), y^(i))  # How far off is ŷ^(i)) from the true output y^(i)?
     2. g ← ∇_θ L(f(x^(i); θ), y^(i))     # How should we move θ to maximize loss?
     3. θ ← θ − η g                        # Go the other way instead
return θ
```

Figure 5.5 The stochastic gradient descent algorithm. Step 1 (computing the loss) is used to report how well we are doing on the current tuple. The algorithm can terminate when it converges (or when the gradient $< \varepsilon$), or when progress halts (for example when the loss starts going up on a held-out set).

The gradient for logistic regression

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

$$\frac{\partial L_{CE}(w,b)}{\partial w_j} = [\sigma(w\cdot x+b)-y]x_j$$

Difference between the model prediction and the correct answer y

Feature value for dimension j

Note: the detailed derivation is available in the reading (SLP3 Chapter 5, section 5.8)

Logistic Regression
What you should know

How to make a prediction with logistic regression classifier

How to train a logistic regression classifier

Machine learning concepts:

Loss function

Gradient Descent Algorithm