

LOGISTIC REGRESSION

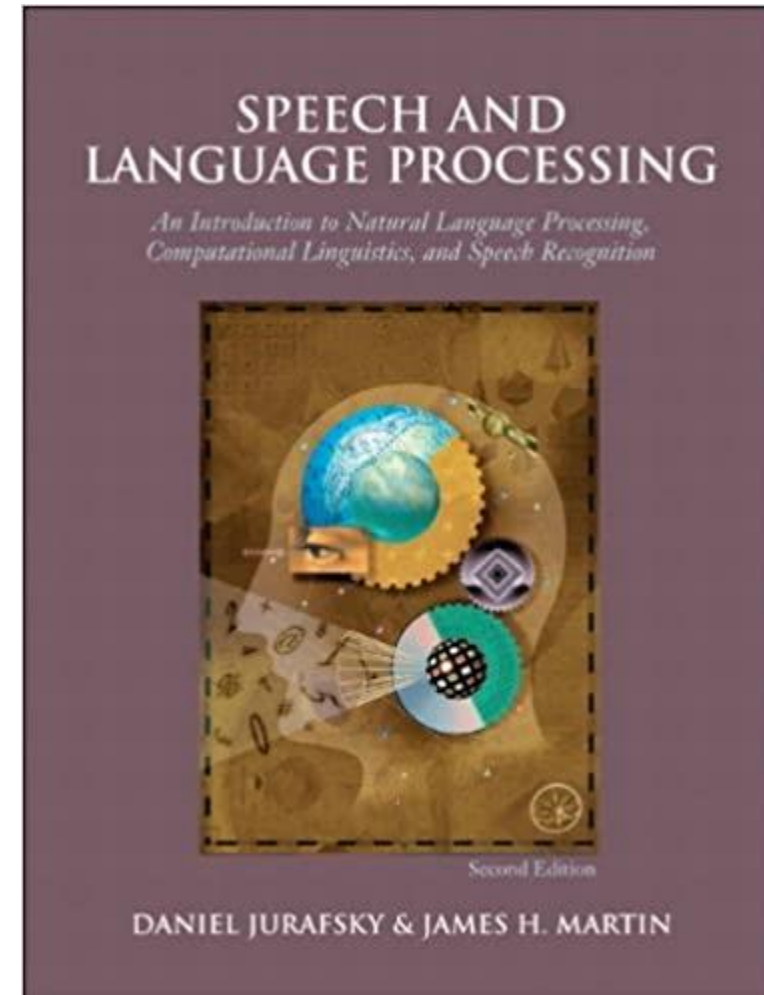
Spring 2023

CS6431 Natural Language Processing

B1:

*Speech and Language Processing (Third Edition draft
– Jan2022)*

Daniel Jurafsky, James H. Martin



Credits

1. B1

Assignment

Read:

B1: Chapter 5

Problems:

Generative and Discriminative Classifiers

- Generative: knows how to generate features if it belonged to a particular class

- E.g. Naïve Bayes

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

- Discriminative model: directly compute $P(c|d)$ by giving more importance to features that are better at discriminating output classes

- Logistic Regression



Logistic Regressor

- A single layer neural network with
 - ▣ Sigmoid/SoftMax as the activation function
 - ▣ Cross-entropy loss function
 - ▣ Uses stochastic gradient descent for optimization



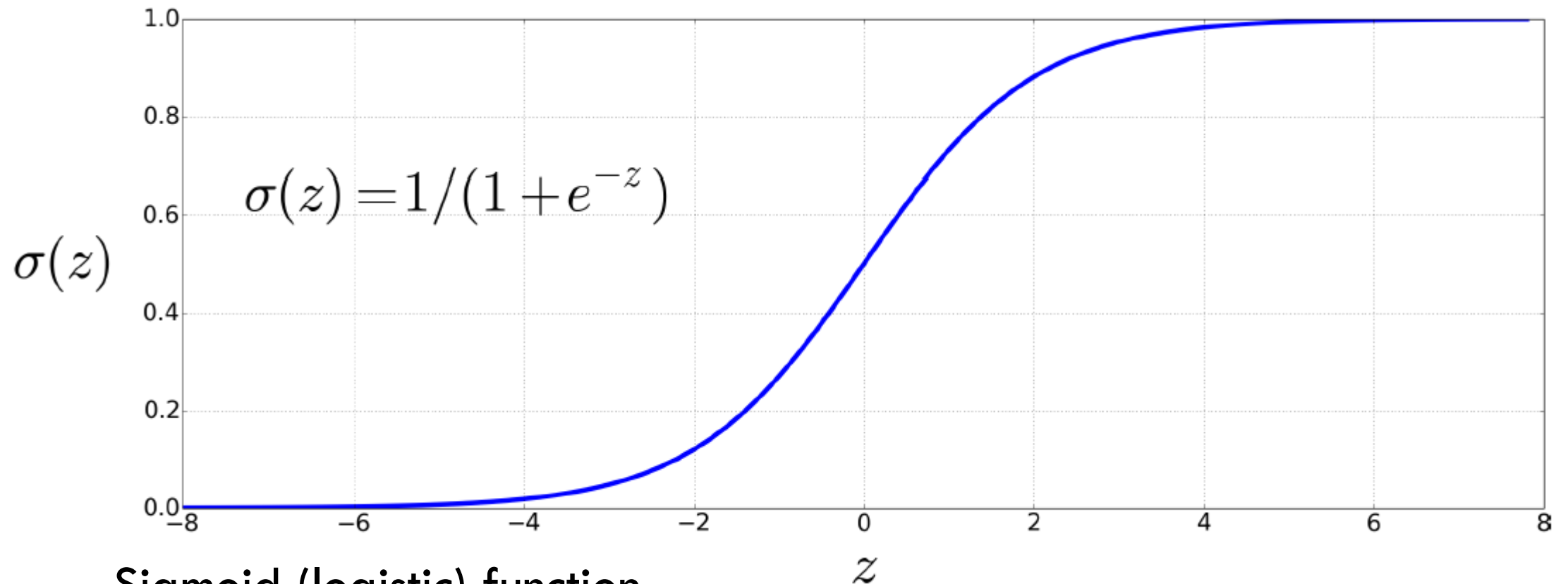
Sigmoid/SoftMax function

□ Inputs, weights, and bias

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

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

if this sum is high , we say $y=1$
if low $y=0$;



Sigmoid (logistic) function

□ Output (in case of binary classification)

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

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

□ For sigmoid

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

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

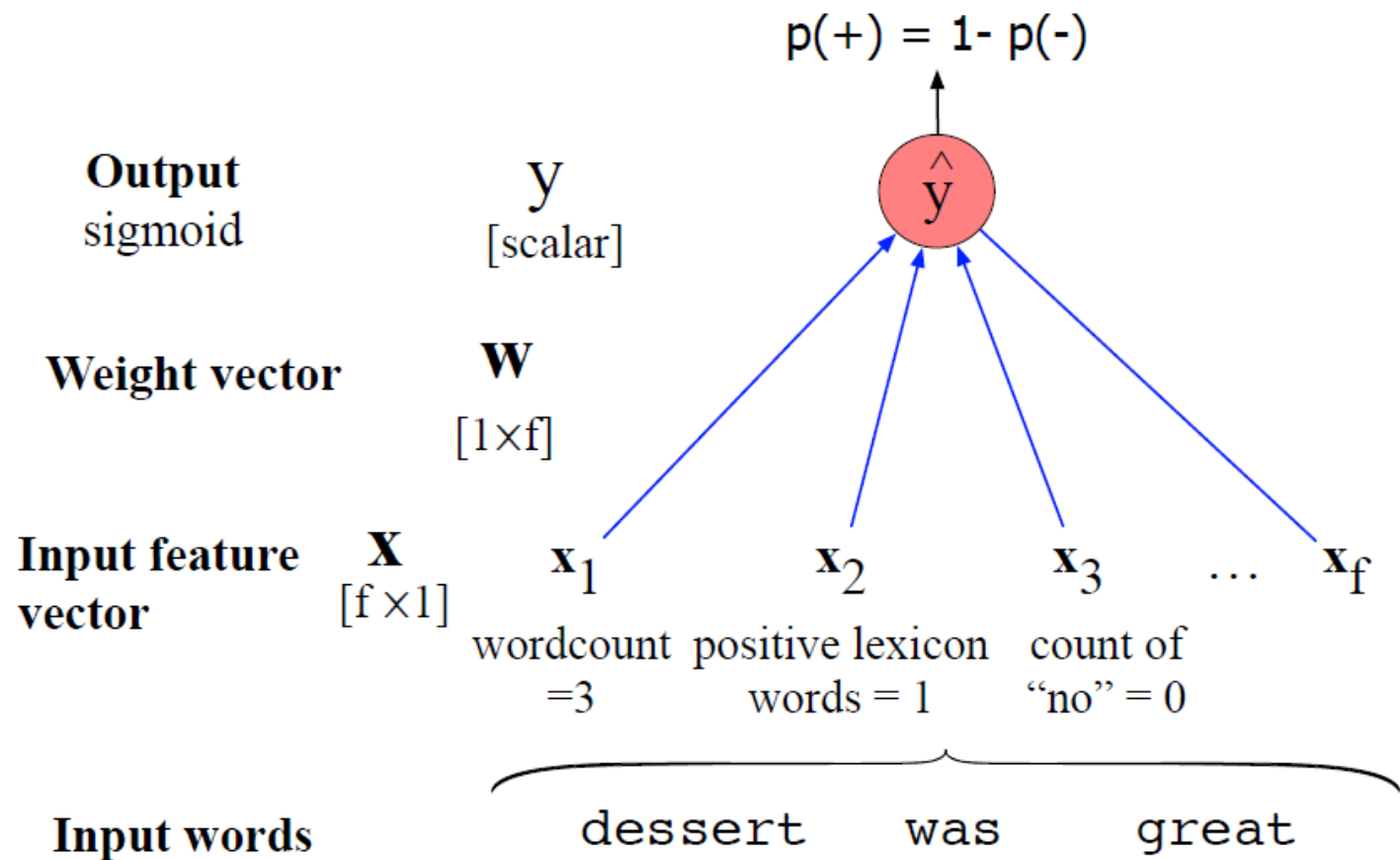
$$\text{decision}(x) = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad \text{decision boundary}$$

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**.

Diagram illustrating feature extraction from the text:

- $x_1 = 3$ (connected to "great")
- $x_2 = 2$ (connected to "no")
- $x_3 = 1$ (connected to "no")
- $x_4 = 3$ (connected to "I" and "you")
- $x_5 = 0$ (connected to "!" in "overcome")
- $x_6 = 4.19$ (connected to "sucked")

Var	Definition	Value in Fig.
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)	$\ln(66) = 4.19$



Var	Definition	Value in Fig.
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)	$\ln(66) = 4.19$

Var	Definition	Value in Fig.
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)	$\ln(66) = 4.19$

Let the corresponding six weights be $[2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and $b = 0.1$

Period disambiguation


- What sort of features would you suggest?
- Hand-crafted features
 - ▣ Feature interactions
 - ▣ Feature templates
- Representation Learning

Scaling Input Features

□ Z-normalization


□ Zero mean

□ Unit variance


$$\mu_i = \frac{1}{m} \sum_{j=1}^m x_i^{(j)}$$

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{j=1}^m \left(\mathbf{x}_i^{(j)} - \mu_i \right)^2}$$

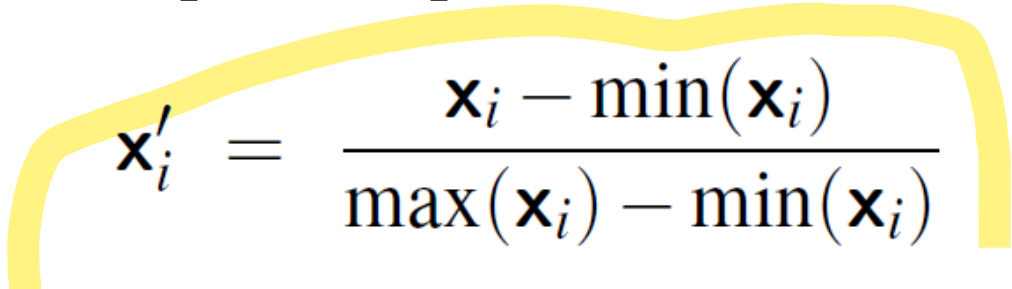
$$\mathbf{x}'_i = \frac{\mathbf{x}_i - \mu_i}{\sigma_i}$$



Handwritten formula for variance:

$$\sigma^2 = \frac{\sum (\mathbf{x}_i - \mu_i)^2}{m}$$

□ Or, simply normalize as ($\in [-1, +1]$)


$$\mathbf{x}'_i = \frac{\mathbf{x}_i - \min(\mathbf{x}_i)}{\max(\mathbf{x}_i) - \min(\mathbf{x}_i)}$$

Logistic Regression vs. Naïve Bayes

- Naïve Bayes has a overly strong conditional independence assumption
 - ▣ Problem with correlated features
- Logistic Regression is much more robust to correlated features
- Naïve Bayes pluses
 - ▣ Works well on small datasets
 - ▣ Easy to implement and fast to train (no optimization step)

Multinomial logistic regression

- Or SoftMax Regression
- Only one among more than two classes can be true
 - ▣ Both predicted output \hat{y} and actual output y are of size k
 - \hat{y}_i estimates $P(y_i = 1|x)$
- Probabilistically normalized version of sigmoid

For a vector \mathbf{z} of dimensionality K , the softmax is defined as:

$$\text{softmax}(\mathbf{z}_i) = \frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^K \exp(\mathbf{z}_j)} \quad 1 \leq i \leq K$$

The softmax of an input vector $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_K]$ is thus a vector itself:

$$\text{softmax}(\mathbf{z}) = \left[\frac{\exp(\mathbf{z}_1)}{\sum_{i=1}^K \exp(\mathbf{z}_i)}, \frac{\exp(\mathbf{z}_2)}{\sum_{i=1}^K \exp(\mathbf{z}_i)}, \dots, \frac{\exp(\mathbf{z}_K)}{\sum_{i=1}^K \exp(\mathbf{z}_i)} \right]$$

□ E.g., input:

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

□ Output:

$$[0.055, 0.090, 0.006, 0.099, 0.74, 0.010]$$

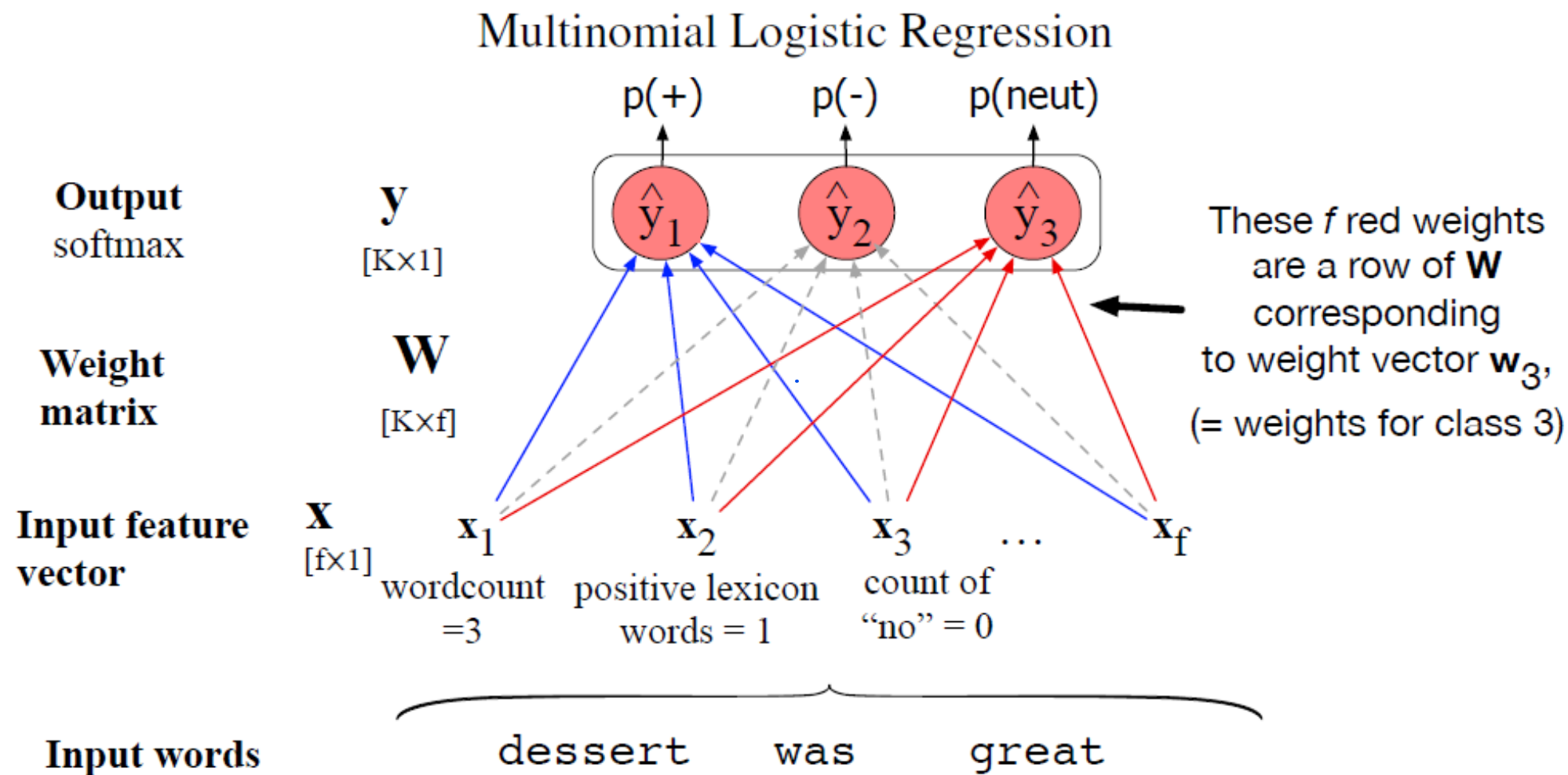
□ For logistic regression

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

Or,

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

$$z = \mathbf{W}\mathbf{x} + \mathbf{b}$$



- In multimodal, a feature can be evidence for or against each individual class.
 - ▣ An exclamation mark ‘!’ may indicate positive or negative emotion, but not neutral

Feature	Definition	$\mathbf{w}_{5,+}$	$\mathbf{w}_{5,-}$	$\mathbf{w}_{5,0}$
$f_5(x)$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	3.5	3.1	-5.3



Cross-entropy loss function

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

- **Conditional maximum likelihood estimation:** Choose w and b that maximize the $\log p(y|x)$ in the training data given the observations x .
- Only two possible outcomes: **Bernoulli distribution**

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

▣ Note: if $y=1$, $p(y|x) = \hat{y}$, else if $y=0$, $p(y|x) = (1 - \hat{y})$

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

- To make it a loss function

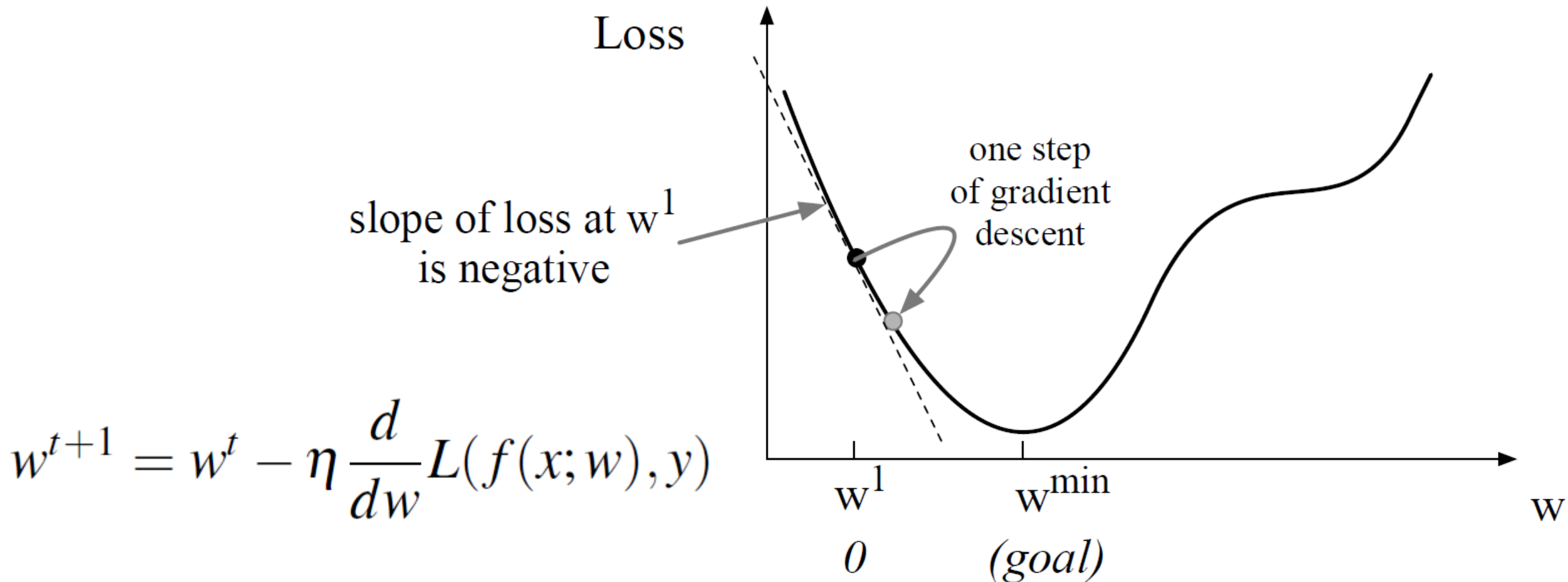
$$L_{\text{CE}}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

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



Stochastic Gradient Descent

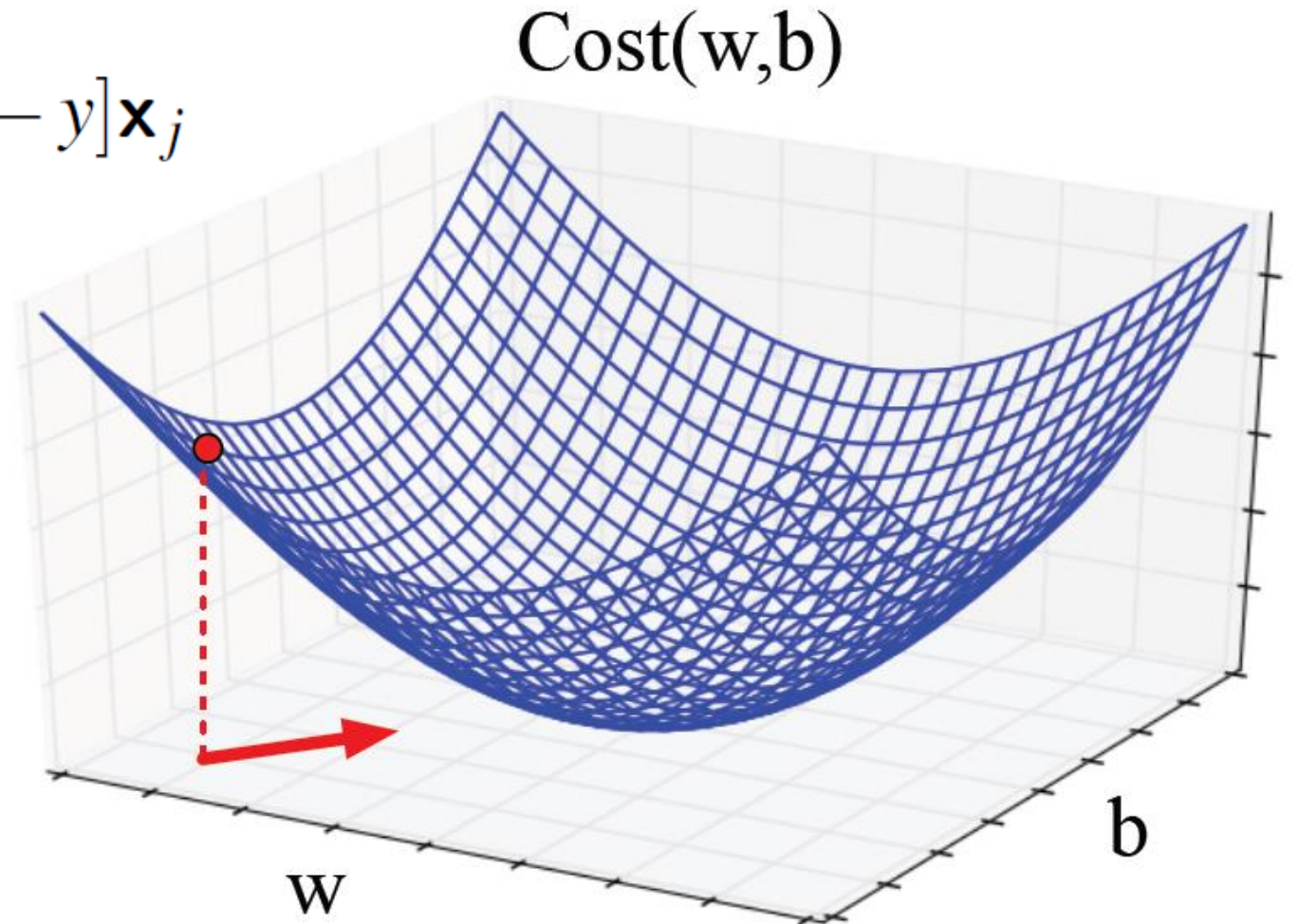
- Figuring out in which direction the function's slope is rising the most steeply, and moving in the opposite direction
- Logistic regression: convex error function
 - ▣ Vs. Neural network: non-convex (multiple local minima)



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

The partial derivative tells the steepest along that dimension

$$\begin{aligned} \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial \mathbf{w}_j} &= [\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y] \mathbf{x}_j \\ &= (\hat{y} - y) \mathbf{x}_j \end{aligned}$$



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 $x^{(1)}, x^{(2)}, \dots, x^{(m)}$

y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \dots, y^{(m)}$

$\theta \leftarrow 0$

repeat til done

For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)

1. Optional (for reporting): # How are we doing on this tuple?

 Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output \hat{y} ?

 Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?

2. $g \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?

3. $\theta \leftarrow \theta - \eta g$ # Go the other way instead

return θ

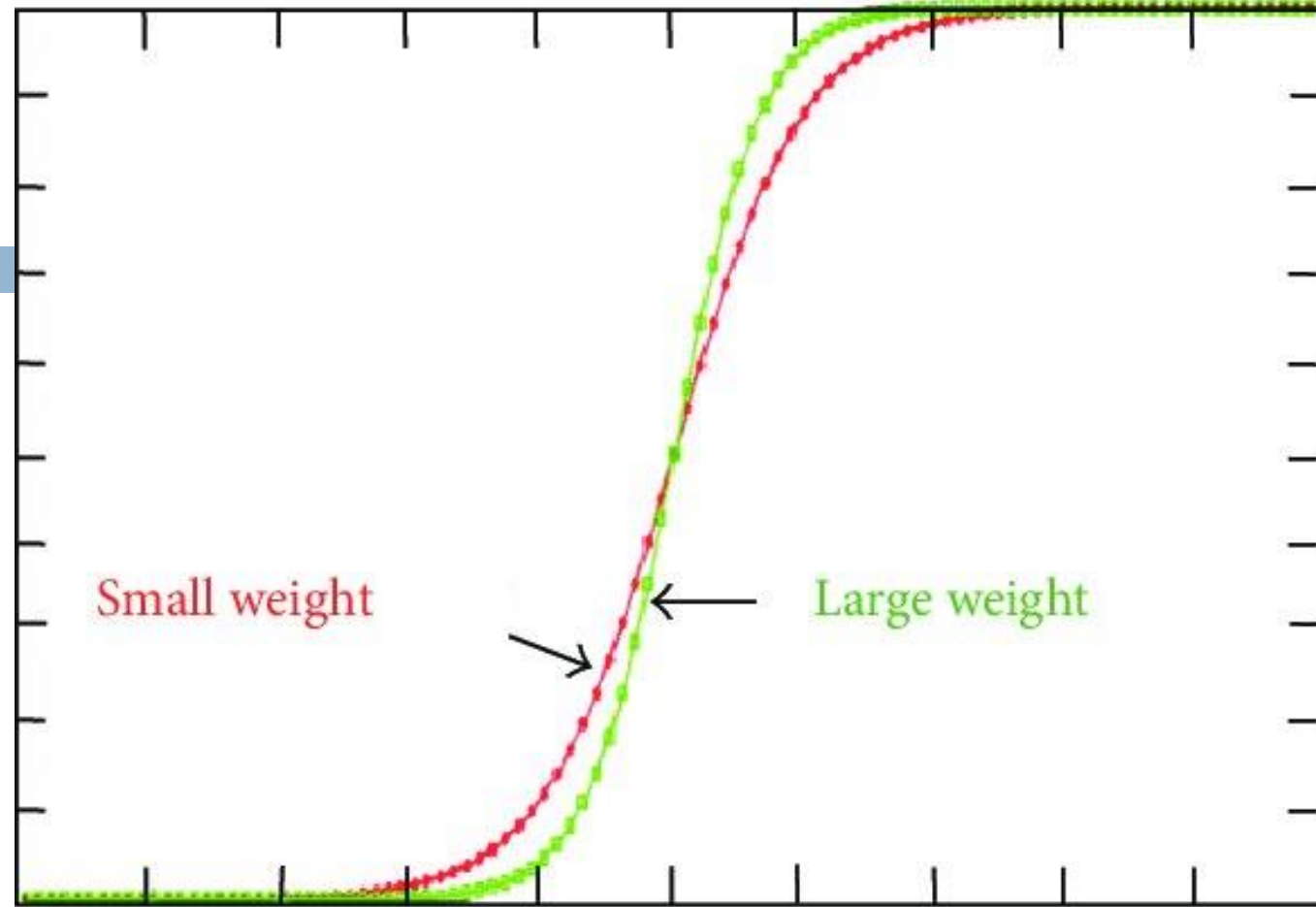
Regularization

- Large weights \Rightarrow over generalization (overfitting)

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

- Add a penalty for large weights

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



- L1 Regularization: Linear function of weights

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

- L2 Regularization: Quadratic function of weight values

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

L1 vs. L2

□ L1

- Linear but non-continuous at 0, complex derivative
- Laplace prior on the weights
- Prefers a sparse weight matrix with a few large weights

□ L2

- Simple derivative
- Gaussian prior with zero mean
- Prefers weight vectors with many small weights

