# Lecture 4:
# Logistic Regression

*Instructor: Swabha Swayamdipta*

*USC CSCI 499 LMs in NLP*

*Jan 29, 2024 Spring*

Slides mostly adapted from Dan Jurafsky, some from Mohit Iyyer

# Logistics / Announcements

- Today: Quiz 1
- HW 1 due on Wednesday, 1/31
- Project Pitches went great, all ideas were very creative and interesting!
  - Votes have been shared
  - If you registered recently, please talk to your classmates and find teammates
  - Teams of 3! We have 27 students who're registered for letter grades
    - Use Piazza / Slack to coordinate remotely or better still, coordinate in person
- Project Proposal is due the Wednesday after next, i.e. 2/7
  - See instructions on the class website

# Lecture Outline

- Announcements
- Recap
  - Generating from a language model
  - Zeros!
  - Smoothing
- Quiz 1
- New topic: Logistic Regression
- Basics of Supervised Machine Learning
  - I. Data
  - II. Model
  - III. Loss
  - IV. Optimization Algorithm
  - V. Inference

# Recap:
# n-gram Language Models
# and Smoothing

# Shakespearean n-grams

| | |
|---|---|
| 1 gram | –To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have<br>–Hill he late speaks; or! a more to leg less first you enter |
| 2 gram | –Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.<br>–What means, sir. I confess she? then all sorts, he is trim, captain. |
| 3 gram | –Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done.<br>–This shall forbid it should be branded, if renown made it empty. |
| 4 gram | –King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in;<br>–It cannot be but so. |

USC Viterbi

# The WSJ is no Shakespeare!

| | |
|---|---|
| 1 gram | Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives |
| 2 gram | Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her |
| 3 gram | They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions |

Shakespearean corpus cannot produce WSJ vocabulary and vice versa

Overfitting!

# Two Types of Overfitting Issues

- At test time:
  - Zero unigram counts
  - Zero n-gram counts
  - May lead to undefined n-gram probabilities and perplexity
  - To be expected, very common!
- Solutions:
  - Zero unigram counts: <UNK> token
    - Closed and Open Vocabularies
  - Zero n-gram counts: Smoothing

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$PPL(\mathbf{w}) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

# N-gram models: Zero Counts

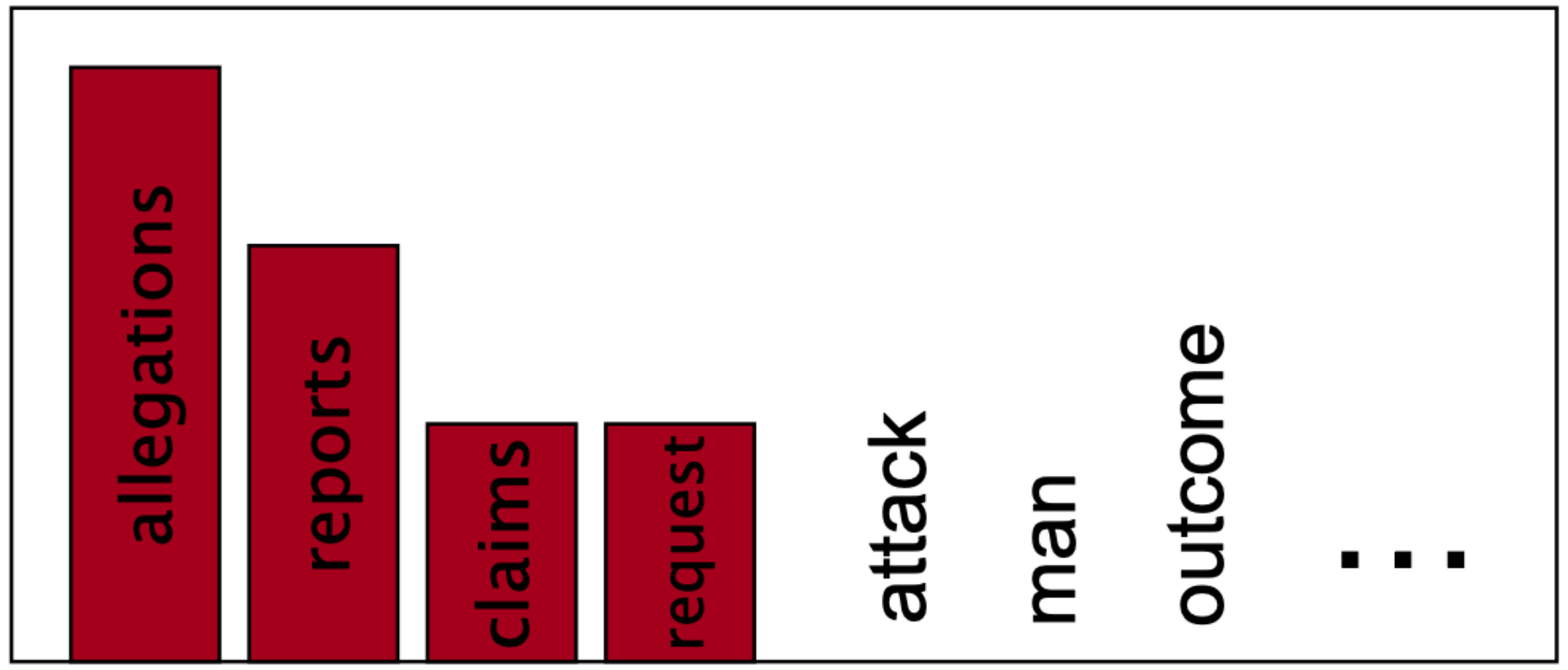
- At test time, we may encounter tokens never seen (unigram with 0 frequency)
  - Very severe yet common problem resulting in undefined probabilities
  - Happens because of new terms, words, different dialects, evolving language
  - These are known as **OOV** for "out of vocabulary", or <UNK> for **unknown tokens**
- **Solution:** Replace all words that occur fewer than $n$ times in the training set, where $n$ is some small number by <UNK> and re-estimate the counts and probabilities
- Design: Open Vocabulary vs. Closed Vocabulary
  - Closed Vocabulary: predetermine the vocabulary
    - Restricted…why?
  - Open Vocabulary: no predetermination but anticipate new tokens

Open vs. Closed Vocabularies

# Smoothing ~ Massaging Probability Masses

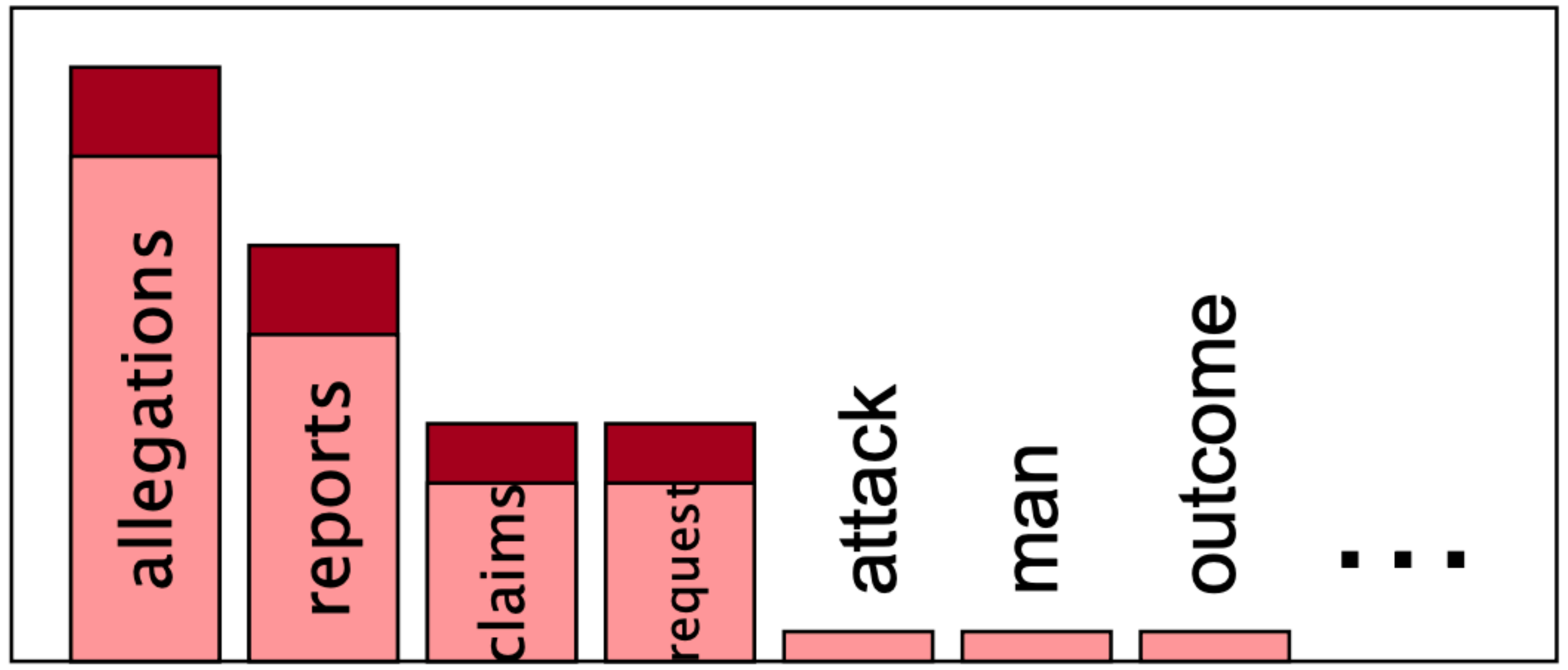
When we have sparse statistics: $Count(w \mid \text{denied the})$

3 allegations
2 reports
1 claims
1 request
**7 total**




Steal probability mass to generalize better: $Count(w \mid \text{denied the})$

2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
**7 total**



USC Viterbi

# Add-One Estimation

MLE estimate

$$P_{MLE}(w_i) = \frac{c(w_i)}{\sum_w c(w)}$$

Laplace smoothing

75 year old method!

1. Pretend we saw each word one more time than we did
2. Just add one to all the counts!
3. All the counts that used to be zero will now have a count of 1…

Add-1 estimate

$$P_{Add-1}(w_i) = \frac{c(w_i) + 1}{\sum_w (c(w) + 1)} = \frac{c(w_i) + 1}{V + \sum_w c(w)}$$

What happens to our $P$ if we don't increase the denominator?

# Add-1 Estimation Bigrams

MLE estimate

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

Pretend we saw each bigram one more time than we did

Add-1 estimate

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i) + 1}{c(w_{i-1}) + V}$$

What does this do to the unigram counts?

Keep the same denominator as before and reconstruct bigram counts

$$= \frac{c^*(w_{i-1}w_i)}{c(w_{i-1})}$$

# Laplace-smoothed bigram counts

Just add one to all the counts!

$w_i$

$w_{i-1}$

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 6 | 828 | 1 | 10 | 1 | 1 | 1 | 3 |
| want | 3 | 1 | 609 | 2 | 7 | 7 | 6 | 2 |
| to | 3 | 1 | 5 | 687 | 3 | 1 | 7 | 212 |
| eat | 1 | 1 | 3 | 1 | 17 | 3 | 43 | 1 |
| chinese | 2 | 1 | 1 | 1 | 1 | 83 | 2 | 1 |
| food | 16 | 1 | 16 | 1 | 2 | 5 | 1 | 1 |
| lunch | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 1 |
| spend | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

USC Viterbi

# Reconstituted Counts

$$c^*(w_{i-1}w_i) = \frac{[c(w_{i-1}w_i) + 1]c(w_{i-1})}{c(w_{i-1}) + V}$$

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 3.8 | 527 | 0.64 | 6.4 | 0.64 | 0.64 | 0.64 | 1.9 |
| want | 1.2 | 0.39 | 238 | 0.78 | 2.7 | 2.7 | 2.3 | 0.78 |
| to | 1.9 | 0.63 | 3.1 | 430 | 1.9 | 0.63 | 4.4 | 133 |
| eat | 0.34 | 0.34 | 1 | 0.34 | 5.8 | 1 | 15 | 0.34 |
| chinese | 0.2 | 0.098 | 0.098 | 0.098 | 0.098 | 8.2 | 0.2 | 0.098 |
| food | 6.9 | 0.43 | 6.9 | 0.43 | 0.86 | 2.2 | 0.43 | 0.43 |
| lunch | 0.57 | 0.19 | 0.19 | 0.19 | 0.19 | 0.38 | 0.19 | 0.19 |
| spend | 0.32 | 0.16 | 0.32 | 0.16 | 0.16 | 0.16 | 0.16 | 0.16 |

USC Viterbi

# Compare with raw bigram counts

**Original, Raw**

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 5 | 827 | 0 | 9 | 0 | 0 | 0 | 2 |
| want | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| to | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| eat | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| chinese | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| food | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| lunch | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| spend | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

**Reconstructed**

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 3.8 | 527 | 0.64 | 6.4 | 0.64 | 0.64 | 0.64 | 1.9 |
| want | 1.2 | 0.39 | 238 | 0.78 | 2.7 | 2.7 | 2.3 | 0.78 |
| to | 1.9 | 0.63 | 3.1 | 430 | 1.9 | 0.63 | 4.4 | 133 |
| eat | 0.34 | 0.34 | 1 | 0.34 | 5.8 | 1 | 15 | 0.34 |
| chinese | 0.2 | 0.098 | 0.098 | 0.098 | 0.098 | 8.2 | 0.2 | 0.098 |
| food | 6.9 | 0.43 | 6.9 | 0.43 | 0.86 | 2.2 | 0.43 | 0.43 |
| lunch | 0.57 | 0.19 | 0.19 | 0.19 | 0.19 | 0.38 | 0.19 | 0.19 |
| spend | 0.32 | 0.16 | 0.32 | 0.16 | 0.16 | 0.16 | 0.16 | 0.16 |

Big change to the counts!

Perhaps 1 is too much, add a fraction?

Add-k smoothing

# Interpolation

Perhaps use some pre-existing evidence

- Condition on less context for contexts you haven't learned much about

**Interpolation**

- mix unigram, bigram, trigram probabilities for a trigram LM
- mix n-gram, (n-1)-gram, … unigram probabilities for an n-gram LM

**Interpolation works better than Add-1 / Laplace**

# Linear Interpolation

**Simple Interpolation**

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_1 P(w_i) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i \mid w_{i-2}w_{i-1})$$

$$\sum_k \boxed{\lambda_k} = 1$$

**Hyperparameters!**

**Context-Conditional Interpolation**

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_3(w_{i-2}^{i-1}) P(w_i \mid w_{i-2}w_{i-1}) + \lambda_2(w_{i-2}^{i-1}) P(w_i \mid w_{i-1}) + \lambda_1(w_{i-2}^{i-1}) P(w_i)$$

**Reconstituted Counts**

Different for every unique context

# How to set the $\lambda$s?

Choose λs to maximize the probability of held-out data:

- Fix the n-gram probabilities (on the training data)
- Then search for λs that give largest probability to held-out set:

$$logP(w_1 \ldots w_n | M(\lambda_1 \ldots \lambda_k)) = \sum_i logP_{M(\lambda_1 \ldots \lambda_k)}(w_i | w_{i-1})$$

# Backoff and Discounting

## Backoff

- use trigram if you have good evidence,
- otherwise bigram, otherwise unigram

Still need a correct probability distribution!

- discount the higher-order n-grams by $d$ to save some probability mass for the lower order n-grams
- need a function $\alpha$ to distribute this probability mass to the lower order n-grams

# Stupid Backoff

- No discounting, just use relative frequencies
- Don't care about a valid language model
- Usually done for extremely large n-gram models

Not a probability distribution (usually denoted as $P$)

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^{i}) > 0 \\ \boxed{0.4}\, S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

**Hyperparameter!**

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

Brants *et al.* 2007

# Lecture Outline

- Announcements
- Recap
  - Generating from a language model
  - Zeros!
  - Smoothing
- Quiz 1
- New topic: Logistic Regression
- Basics of Supervised Machine Learning
  - I. Data
  - II. Model
  - III. Loss
  - IV. Optimization Algorithm
  - V. Inference

# Quiz 1!

# Basics of Supervised Machine Learning

# Ingredients of Supervised Machine Learning

I. **Data** as pairs $(x^{(i)}, y^{(i)})$ s.t $i \in \{1 \ldots N\}$
  - $x^{(i)}$ usually represented by a feature vector $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_d]$,
    - e.g. word embeddings

II. **Model**
  - A classification function that computes $\hat{y}$, the estimated class, via $p(y \mid x)$
    - e.g. sigmoid function: $\sigma(z) = 1/(1 + \exp(-z))$

III. **Loss**
  - An objective function for learning
    - e.g. cross-entropy loss, $L_{CE}$

IV. **Optimization**
  - An algorithm for optimizing the objective function
    - e.g. stochastic gradient descent

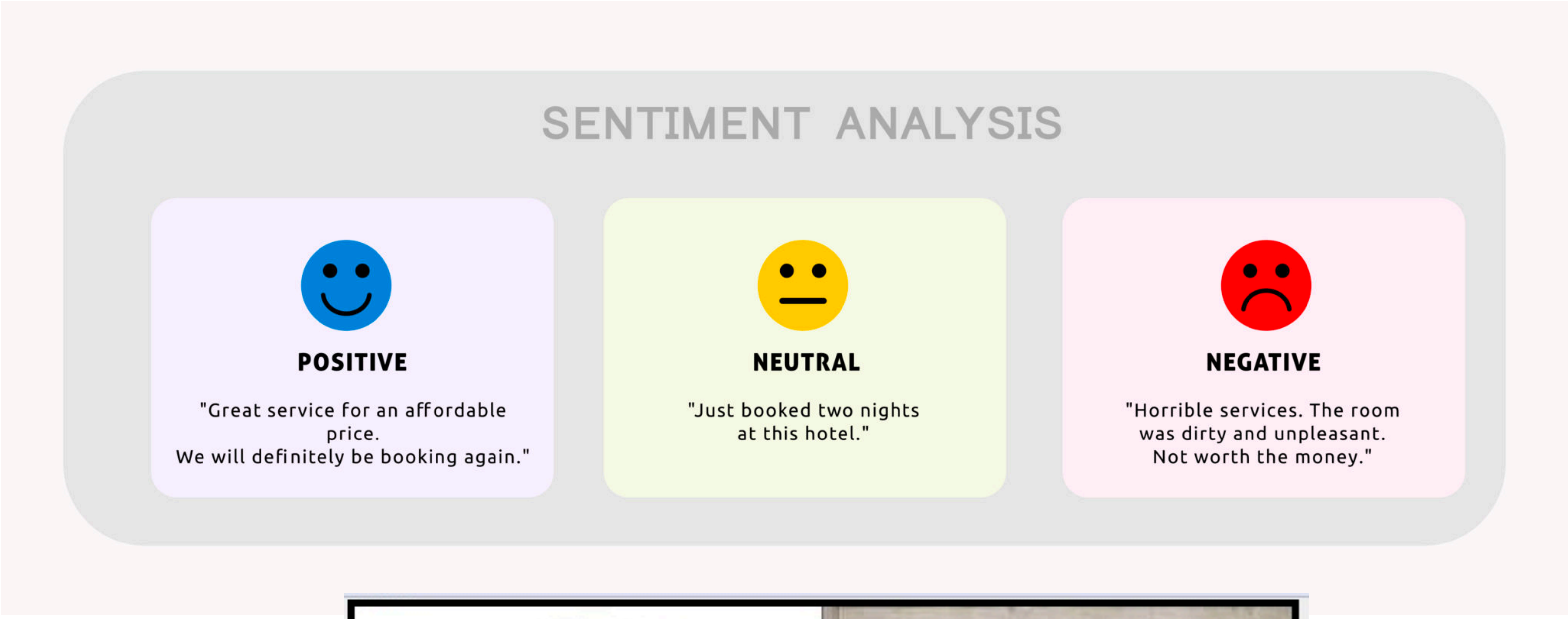
V. **Inference** / Evaluation

Learning Phase

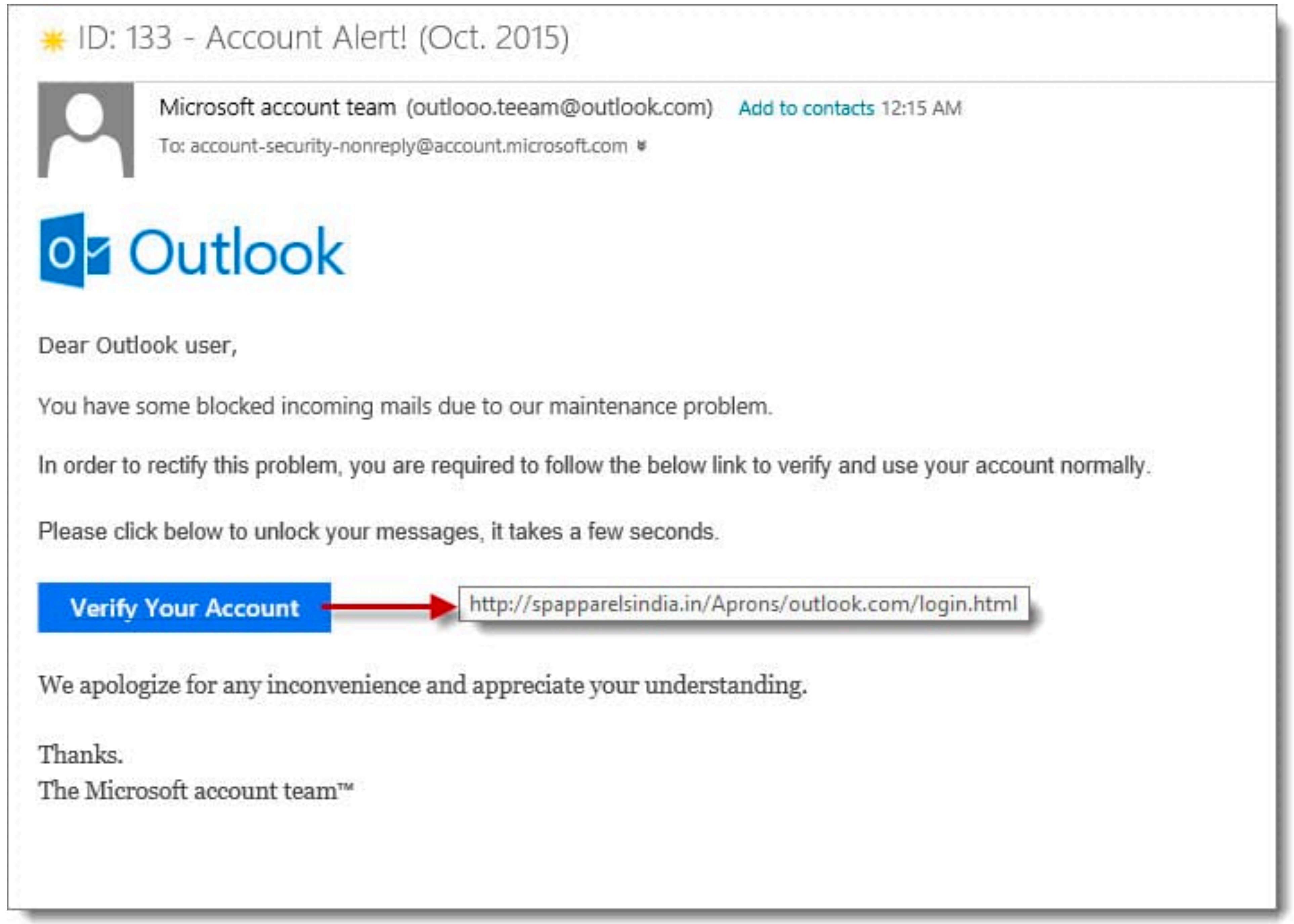
# Learning vs. Inference

- **Learning**: we learn parameter weights by minimizing the loss function using an optimization algorithm
- **Inference**: Given a test example $x_{test}$ we compute $p(y|x)$ using learned weights and return whichever label receives higher probability
- Distinct from **training** and **evaluation**
  - Evaluation only contains inference; no parameters are updated
  - Training contains both learning and inference

# Text Classification Tasks



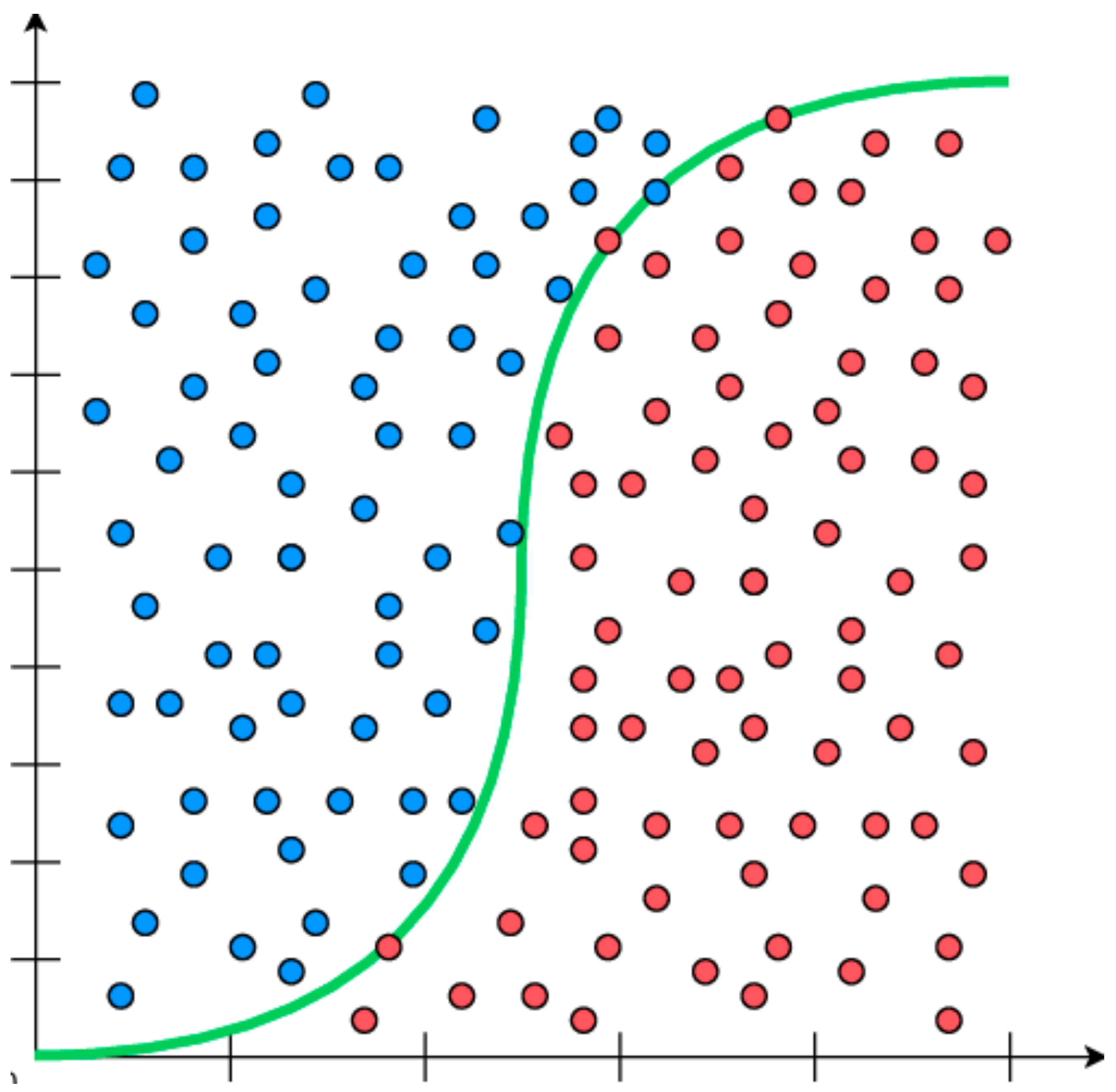





Not just NLP, classification is a general ML technique often applied across a wide variety of prediction tasks!

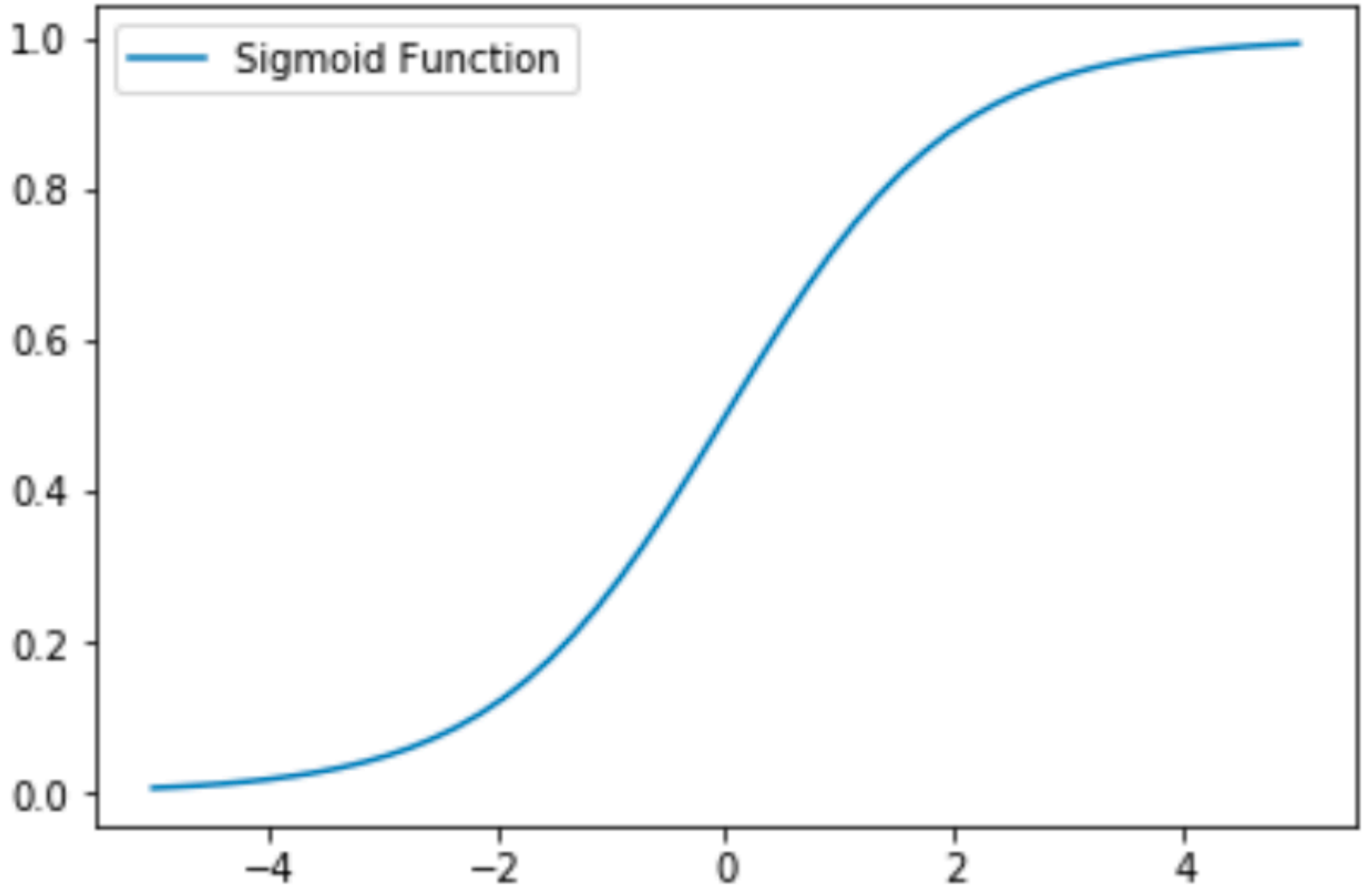
# Text Classification Setup

- Input:
  - a document $x$
    - Each observation $x^{(i)}$ is represented by a feature vector $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$
  - a label $y$ from a fixed set of classes $C = c_1, c_2, \ldots, c_J$
- Output: a predicted class $\hat{y} \in C$
- Setting for Binary Classification: given a series of input / output pairs:
  - $(x^{(i)}, y^{(i)})$ where label $y^{(i)} \in C = \{0,1\}$
- Goal of Binary Classification
  - At test time, for input $x^{test}$, compute an output: a predicted class $\hat{y}^{test} \in \{0,1\}$



# Today: Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks
- Logistic regression is a discriminative classifier
  - Learn a model that can (given the input) distinguish between different classes
- Other classification algorithms: Naïve Bayes, K-Nearest Neighbors, Decision Trees, SVMs




Is language modeling a classification task?

# Classification: Single Observation

- Input observation: vector of features, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $\mathbf{w} = [w_1, w_2, \ldots, w_n]$
  - Sometimes we call the weights $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output: a predicted class
  - Binary logistic regression $\hat{y} \in \{0,1\}$
  - Multinomial logistic regression (e.g. 5 classes): $\hat{y} \in \{0,1,2,3,4\}$

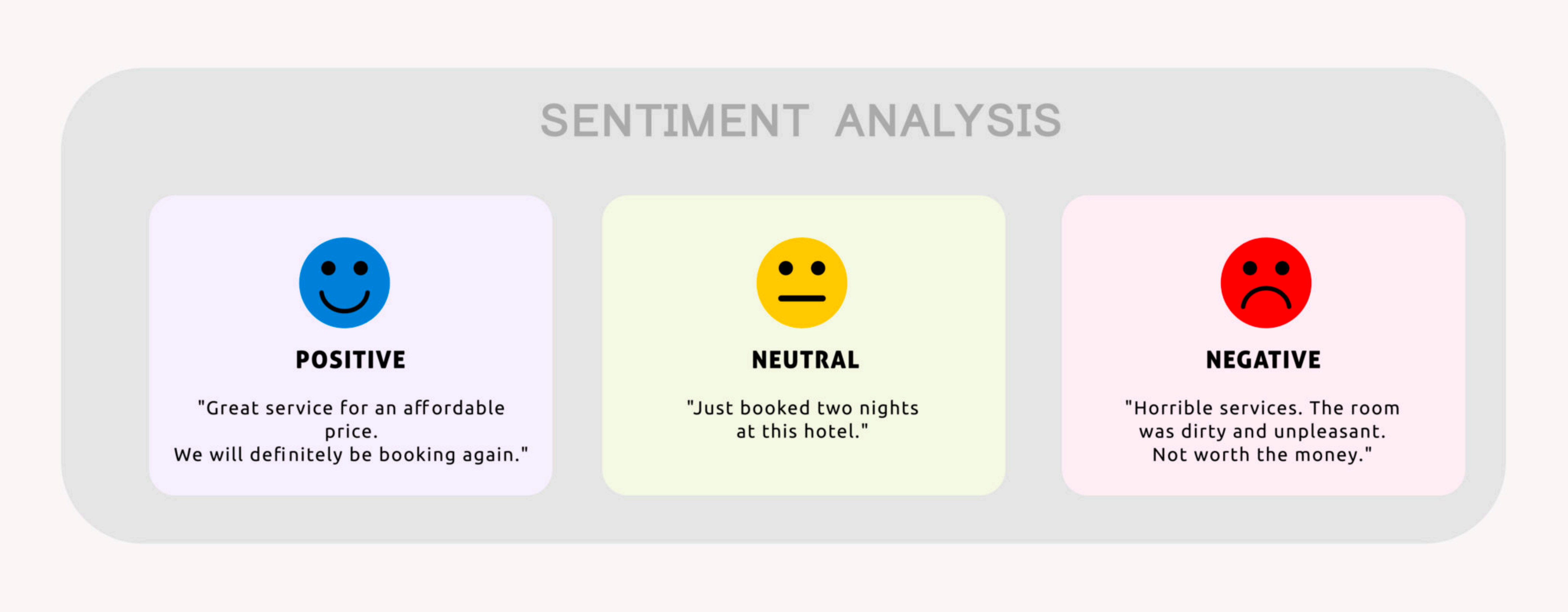
Parametric Model

# Lecture Outline

- Announcements
- Recap
  - Generating from a language model
  - Zeros!
  - Smoothing
- Quiz 1
- New topic: Logistic Regression
- Basics of Supervised Machine Learning
  - I. Data
  - II. Model
  - III. Loss
  - IV. Optimization Algorithm
  - V. Inference

# I. Data

# Features in Classification




- Examples of feature $x_i$
  - $x_i =$"review contains 'awesome'"; $w_i = +10$
  - $x_j =$"review contains 'abysmal'"; $w_j = -10$
  - $x_k =$"review contains 'mediocre'"; $w_k = -2$
- Each $x_i$ is associated with a weight $w_i$ which determines how important $x_i$ is
  - (For predicting the positive class)
- May be
  - manually configured or
  - automatically inferred, as in modern architectures

Can you guess the $w$ for $x_l$ = "review contains 'restaurant'"?

# II. Model: Logistic Regression

# How to get the right $y$?

- For each feature $x_i$, introduce a weight $w_i$, which determines the importance of $x_i$
  - Sometimes we have a bias term, $b$ or $w_0$, which is just another weight not associated to any feature
  - Together, all parameters can be termed as $\theta = [w; b]$
- We consider the weighted sum of all features and the bias

$$z = \left( \sum_d w_d x_d + b \right)$$
$$= \mathbf{w} \cdot \mathbf{x} + b$$

If high, $\hat{y} = 1$

If low, $\hat{y} = 0$

But how to determine the threshold?

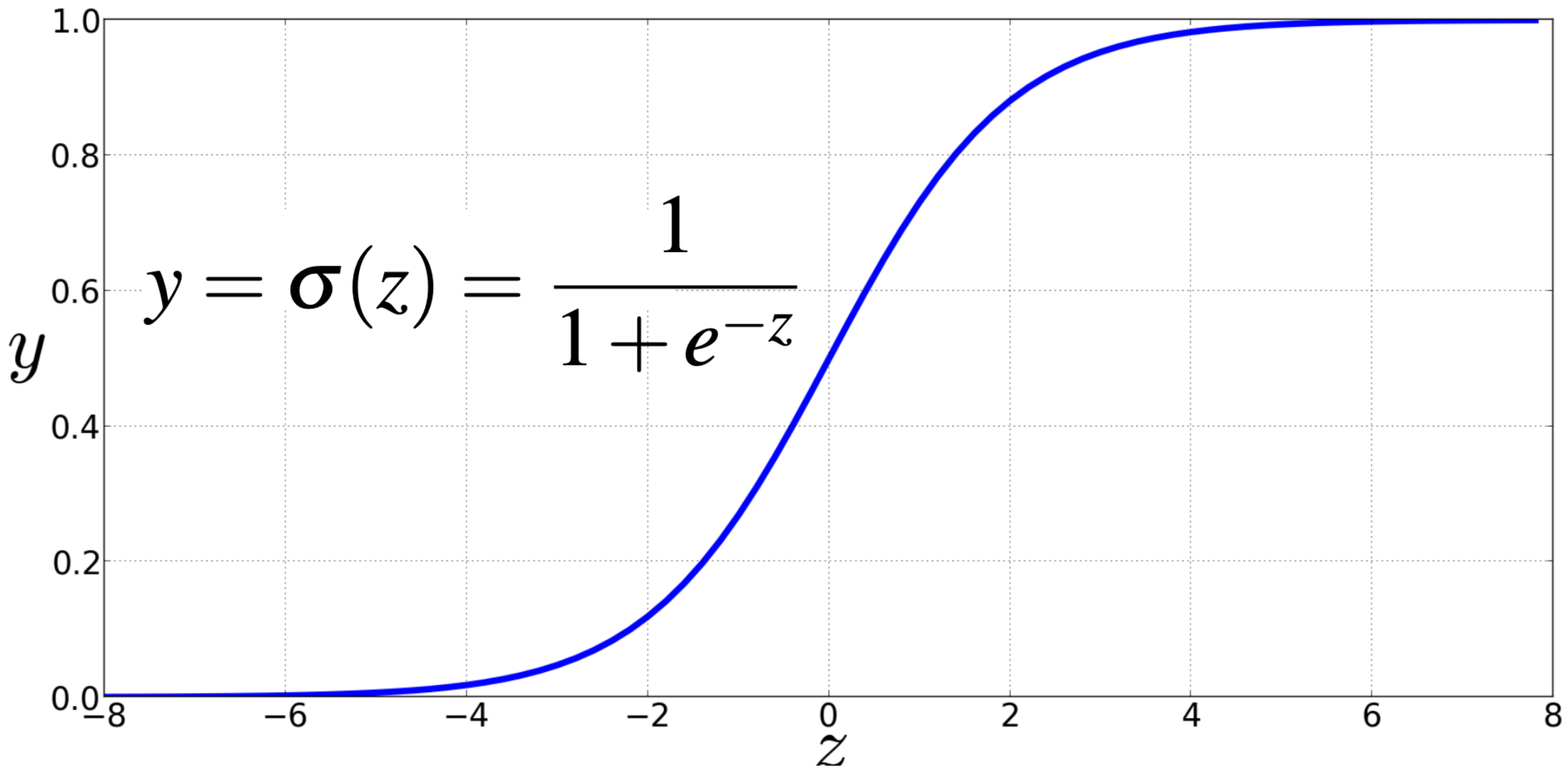
**We need probabilistic models!**

$$P(y = 1 \mid \mathbf{x}; \theta)$$
$$P(y = 0 \mid \mathbf{x}; \theta)$$

# Solution: Squish it into the 0-1 range

$$z = \mathbf{w} \cdot \mathbf{x} + b \qquad z \in \mathbb{R}$$

- Sigmoid Function, $\sigma(\cdot)$
  - Non-linear!
- Compute $z$ and then pass it through the sigmoid function
- Treat it as a probability!
- Also, a differentiable function, which makes it a good candidate for optimization (more on this later!)



# Sigmoids and Probabilities

$$P(y=1\,|\,\mathbf{x};\theta) = \sigma(\mathbf{w}\cdot\mathbf{x}+b)$$
$$= \frac{1}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}$$

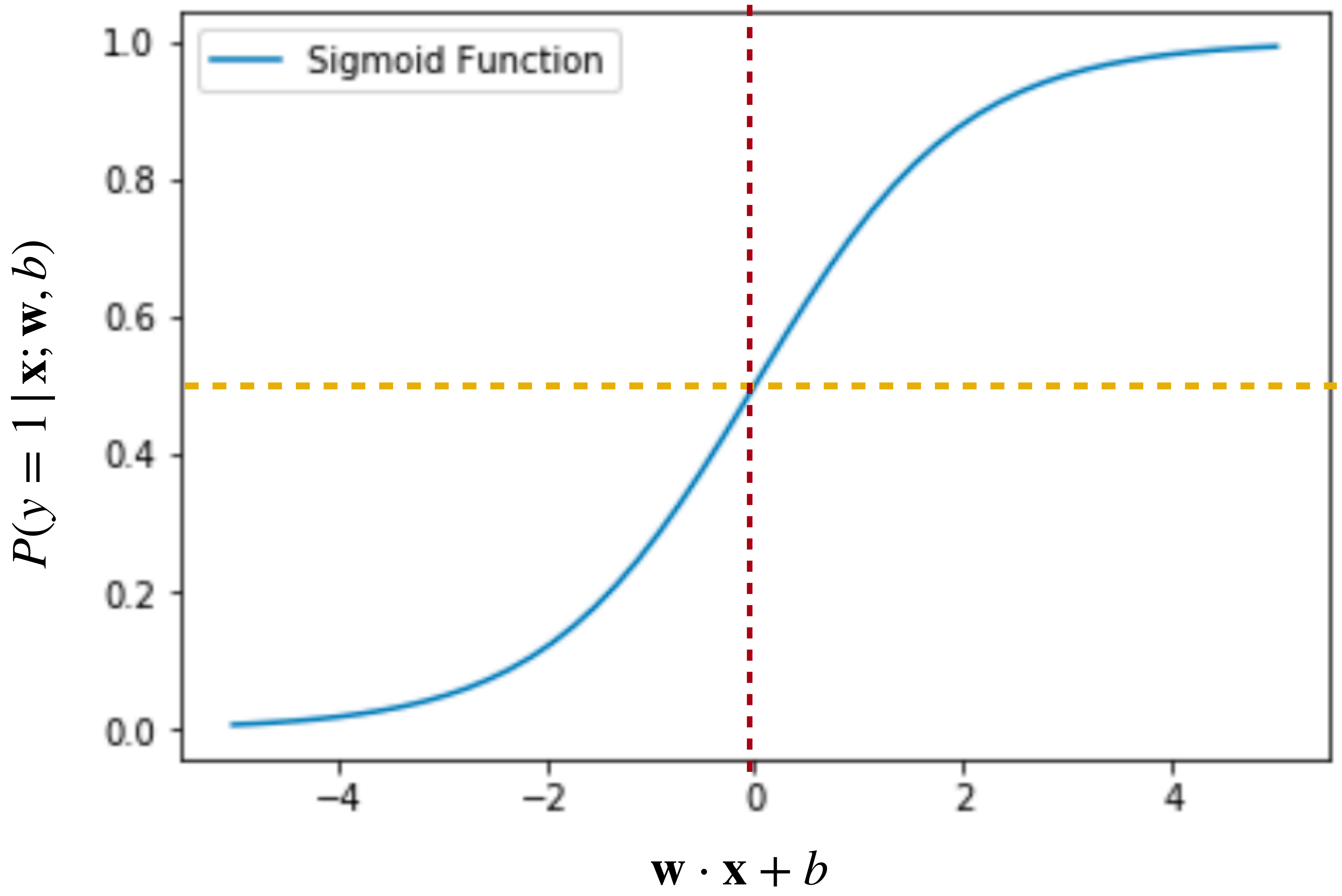
$$P(y=0\,|\,\mathbf{x};\theta) = 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b)$$
$$= 1-\frac{1}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}$$
$$= \frac{\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}$$
$$= \frac{1}{1+\exp(\mathbf{w}\cdot\mathbf{x}+b)}$$
$$= \sigma(-(\mathbf{w}\cdot\mathbf{x}+b))$$

# Classification Decision

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

Decision Boundary

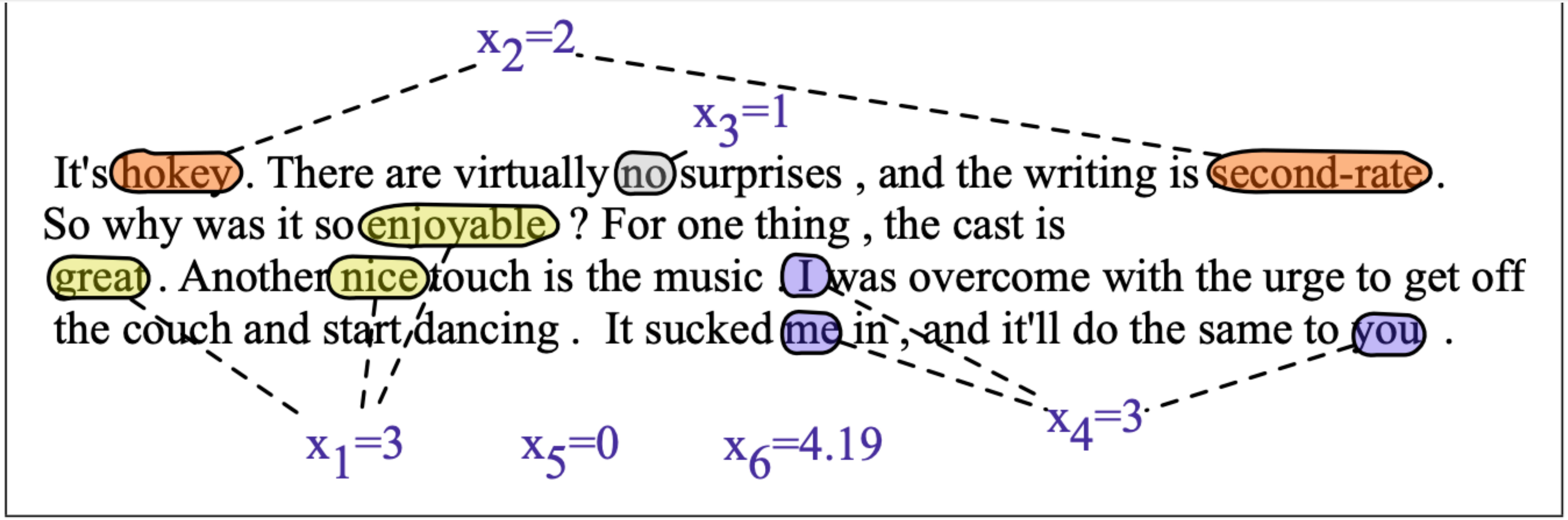
$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \end{cases}$$



# 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.

Is $y = 1$ or $y = 0$?




| Var | Definition | Value in Fig. 5.2 |
|---|---|---|
| $x_1$ | count(positive lexicon) $\in$ doc) | 3 |
| $x_2$ | count(negative lexicon) $\in$ 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) | $\ln(66) = 4.19$ |

# Example: Classifying Sentiment

| Var | Definition | Val | 5.2 |
|---|---|---|---|
| $x_1$ | count(positive lexicon) $\in$ doc) | 3 | |
| $x_2$ | count(negative lexicon) $\in$ 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) | $\ln(66) = 4.19$ | |

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

$b = 0.1$

# Example: Classifying Sentiment

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

$$
\begin{aligned}
p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.30
\end{aligned}
$$

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.

# Applying LR to other tasks

- Example: Period Disambiguation: Does a period correspond to the end of sentence?
  - "We saw many smoothing algorithms in class, e.g. Laplace."

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

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

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

Different tasks need different features; manually designed features must be task specific!

# But where do the $\mathbf{w}$'s and the $b$'s come from?

- Supervised Classification:
  - We know the correct label $y$ (either 0 or 1) for each $x$
  - But what the system produces is an estimate, $\hat{y}$
- Set $\mathbf{w}$ and $b$ to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$
  - We need a distance estimator: a **loss function** or a **cost function**
  - We need an **optimization algorithm** to update $\mathbf{w}$ and $b$ to minimize the loss.

Loss function

Optimization Algorithm

# III. Loss: Cross-Entropy

# The distance between $\hat{y}$ and $y$

- We want to know how far is the classifier output:
  - $\hat{y} = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$
- From the true (ground truth / gold standard) label:
  - $y \in \{0,1\}$
- This difference is called the loss or cost
  - $L(\hat{y}, y)$ = how much $\hat{y}$ differs from $y$
  - In other words, how much would you lose if you mispredicted
  - Or how much would it cost you to mispredict

# Remember maximum likelihood?

- Here: **conditional** maximum likelihood estimation
- We choose the parameters $\mathbf{w}, b$ that maximize
  - the log probability
    - of the true $y$ labels in the training data
    - **given** the observations $x$

$$\max \log p(y \mid x)$$

Suppose we flip the coin four times and see (H, H, H, T).

What is $p$?


$p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

# Maximizing conditional likelihood

For a single observation

**Goal**: maximize probability of the correct label $p(y \mid x)$

Since there are only 2 discrete outcomes (0 or 1) we can express the probability $p(y \mid \mathbf{x})$ from our classifier (the thing we want to maximize) as

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

| | $\hat{y} = 0$ | $\hat{y} = 1$ |
|---|---|---|
| $y = 0$ | 1 | 0 |
| $y = 1$ | 0 | 1 |

# Maximizing conditional likelihood

**Goal**: maximize probability of the correct label $p(y \mid \mathbf{x})$

Maximize: $p(y \mid x) = \hat{y}^y (1 - \hat{y})^{1-y}$

Now take the log of both sides

Why do we need this?

$$\log p(y \mid x) = \log(\hat{y}^y (1 - \hat{y})^{1-y})$$
$$= y \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

Whatever values maximize $\log p(y \mid x)$ will also maximize $p(y \mid x)$

# Minimizing negative log likelihood

**Goal**: maximize probability of the correct label $p(y \mid \mathbf{x})$

Measures how well the training data matches the proposed model distribution and how good the model distribution is

Maximize: $$\log p(y \mid x) = \log(\hat{y}^y (1-\hat{y})^{1-y})$$
$$= y \log \hat{y} + (1-y)\log(1-\hat{y})$$

Now flip the sign for something to minimize (we minimize the loss / cost)

Cross-Entropy Loss

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

# Loss for sentiment classification

We want loss to be:

- smaller if the model estimate is close to correct
- bigger if model is confused

Let's first suppose the true label of this is $y = 1$ (positive)

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.

# Sentiment Example

True value is y=1. How well is our model doing?

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

Pretty well! What's the loss?

$$
\begin{aligned}
L_{\mathrm{CE}}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1-y) \log(1 - \sigma(w \cdot x + b))] \\
&= -[\log \sigma(w \cdot x + b)] \\
&= -\log(.70) \\
&= .36
\end{aligned}
$$

# Sentiment Example: Contd

Now, suppose true value is $y = 0$. How well is our model doing?

$$
\begin{aligned}
p(-|x) = P(Y=0|x) &= 1-\sigma(w \cdot x + b) \\
&= 0.30
\end{aligned}
$$

What's the loss?

$$
\begin{aligned}
L_{\mathrm{CE}}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1-y) \log(1-\sigma(w \cdot x + b))] \\
&= -[\log(1-\sigma(w \cdot x + b))] \\
&= -\log(.30) \\
&= 1.2
\end{aligned}
$$

# Sentiment Example: Summary

The loss when the model is right (if true $y = 1$):

$$
\begin{aligned}
L_{\mathrm{CE}}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))] \\
&= -[\log \sigma(w \cdot x + b)] \\
&= -\log(.70) \\
&= .36
\end{aligned}
$$

Loss is bigger when the model is wrong!

..is lower than the loss when the model was wrong (if true $y = 0$)

$$
\begin{aligned}
L_{\mathrm{CE}}(\hat{y}, y) &= -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))] \\
&= -[\log (1 - \sigma(w \cdot x + b))] \\
&= -\log(.30) \\
&= 1.2
\end{aligned}
$$

Next: an **optimization algorithm** to update $\mathbf{w}$ and $b$ to minimize the loss

# Next Class

- Logistic Regression Contd.
  I. Optimization Algorithm
  II. Inference
- Introduction to Word Embeddings