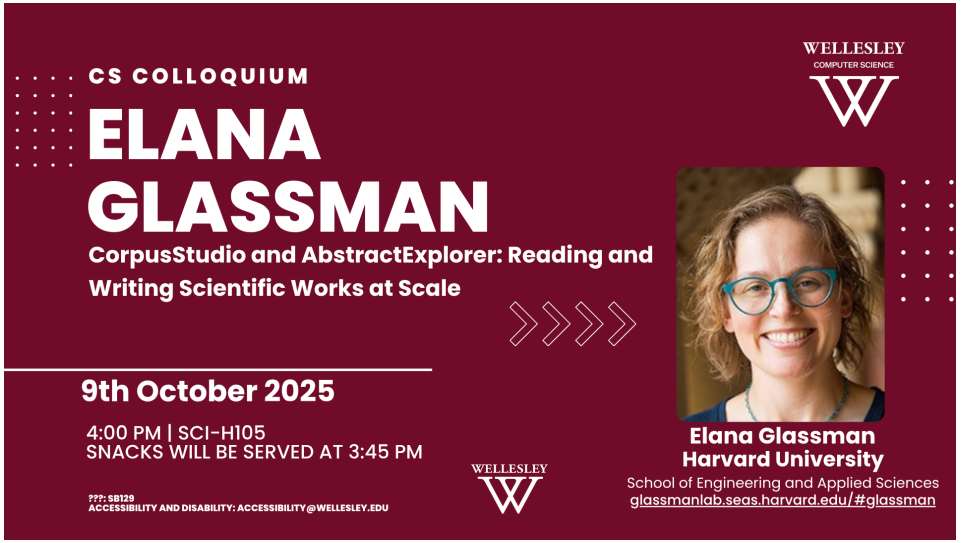
Classifying with Regression

Reminders

- Midterm 1: this Friday
- My next help hours: Thursday **3-4pm**
- CS Colloquium from 4-5pm: Elena Glassman
- Social Choice conference Oct. 14th-15th
- **Over break**: try to install PyTorch in your CS 333 Python environment



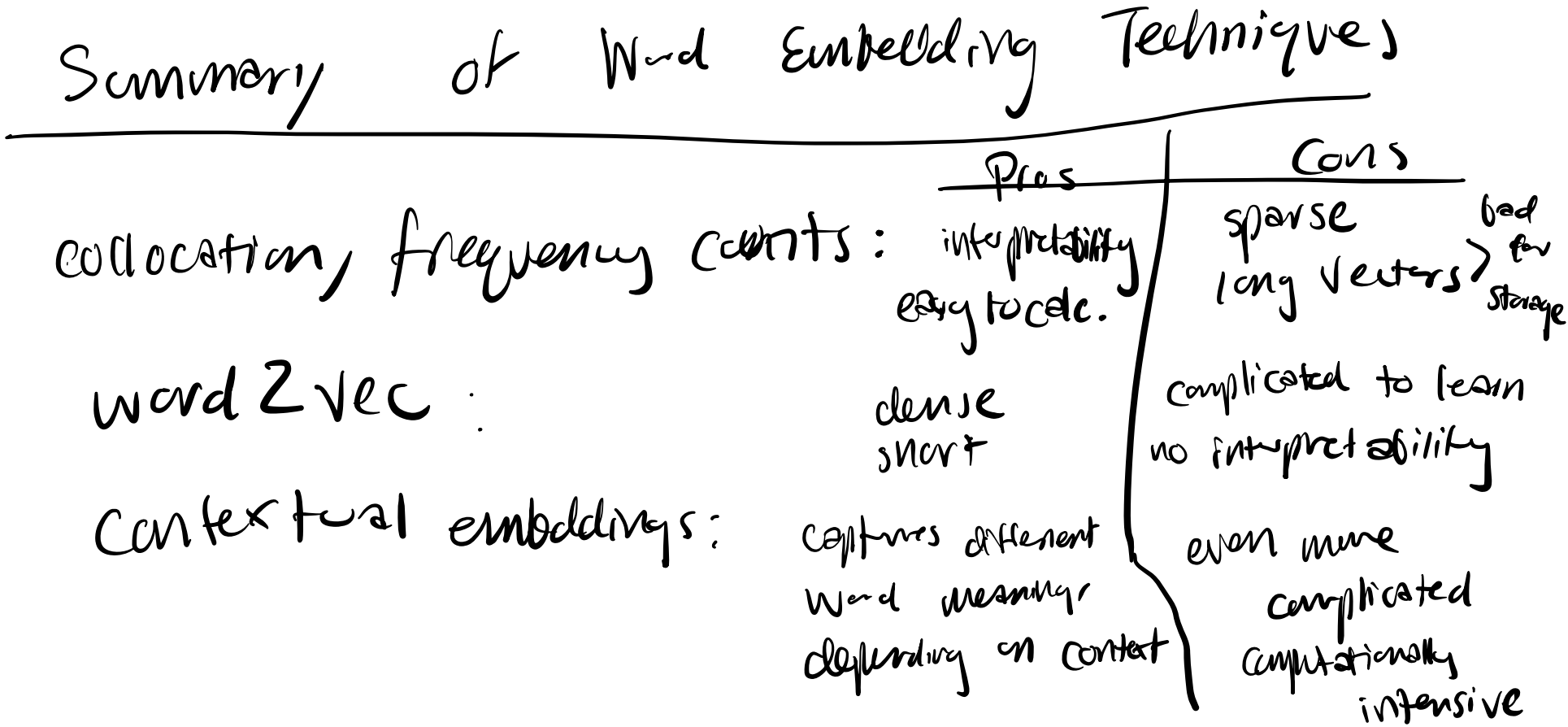
Midterm 1

- In-class programming midterm
- Bring your own laptop to work on
- I will have starter code and documentation for you to download at the beginning, then ask you to turn off wifi.
- At the end, you will submit your code on Gradescope.
- May bring a 4x6in note card

Word Embedding Exploration

Word Embedding Interactive Demo

https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/index.html



Logistic Regression

Generative and Discriminative Classifiers

Naive Bayes is a **generative** classifier.

Logistic regression is a **discriminative** classifier.

Generative and Discriminative Classifiers

Suppose we're distinguishing cat from dog images

me

imagenet

Generative Classifier:

- Build a model of what's in a cat image
  - Knows about whiskers, ears, eyes
  - Assigns a probability to any image:
    - how cat-y is this image?

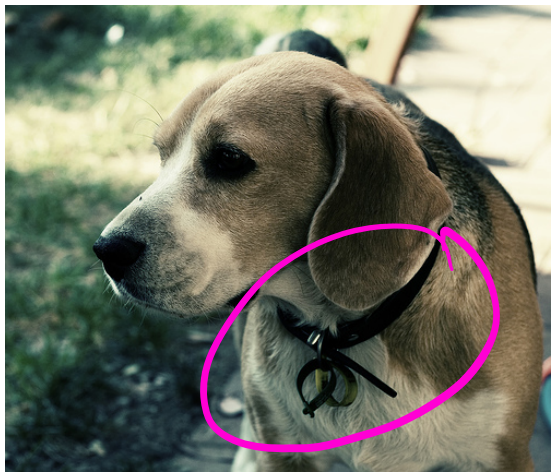
Also build a model for dog images

Now given a new image:

Run both models and see which one fits better

Discriminative Classifier

Just try to distinguish dogs from cats



Dogs have collars! Let's ignore everything else

Not easy to answer questions like: *given that I observe a bone, how likely is it to be a dog?*

But: discriminative classifiers generally perform better.

Finding the correct class c from a document d with Generative vs Discriminative Classifiers

Naive Bayes: $$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \; P(d \mid c) P(c)$$

Logistic Regression: $$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \; P(c \mid d)$$

Logistic Regression Classifiers

Text Classification: definition

Input:

- a document x
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

Output: a predicted class $\hat{y} \in C$

Binary Classification in Logistic Regression

Given a series of input/output pairs:

- $(x^{(i)}, y^{(i)})$

For each observation $x^{(i)}$

- We represent $x^{(i)}$ by a **feature vector** $[x_1, x_2, \ldots, x_n]$
- We compute an output: a predicted class $\hat{y}^{(i)} \in \{0,1\}$

Features in logistic regression

For feature x_i, weight w_i tells is how important is x_i

- x_i ="review contains '`awesome`'": $w_i = +10$
- x_j ="review contains '`abysmal`'": $w_j = -10$
- x_k ="review contains '`mediocre`'": $w_k = -2$

Logistic Regression for one observation x

Input observation: vector $x = [x_1, x_2 \dots x_n]$

Weights: one per feature: $w = [w_1, w_2 \dots w_n]$

Sometimes denoted θ: $\theta = [\theta_1, \theta_2 \dots]$

Output: a predicted class $\hat{y} \in \{0, 1\}$

Multinomial: $\hat{y} \in \{0, 1, \dots n\}$

How to do classification

For each feature x_i, weight w_i tells us importance of x_i

- (Plus we'll have a bias b)

We'll sum up all the weighted features and the bias

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

$$z = w \cdot x + b$$

If z is high, we predict $\hat{y} = 1$

If z is low, we predict $\hat{y} = 0$

But we want a probabilistic classifier

We need to formalize "sum is high".

We'd like a principled classifier that gives us a probability, just like Naive Bayes did

We want a model that can tell us:

$p(y=1 \mid x; W)$

$p(y=0 \mid x; W)$

$p(y=0 \mid x; w) =$

$1 - p(y=1 \mid x; w)$

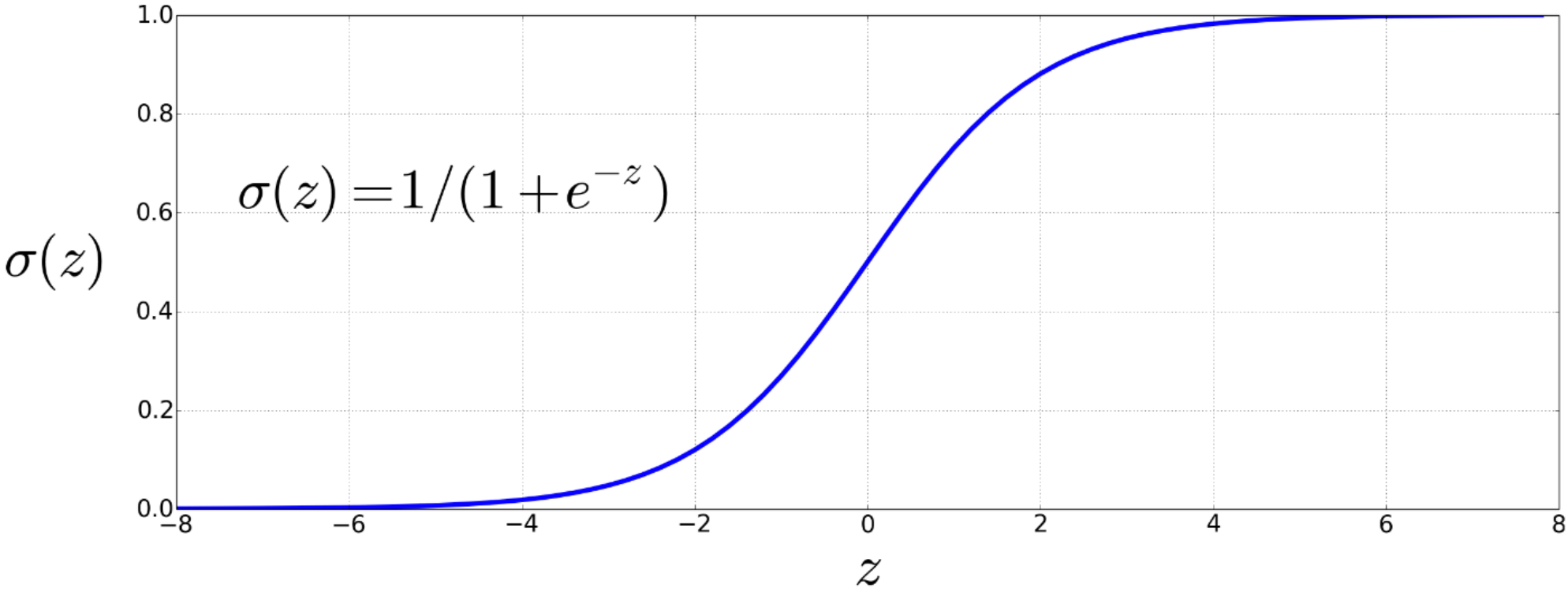
The problem: z isn't a probability, it's just a number!

$$z = w \cdot x + b$$

Solution: use a function of z that goes from 0 to 1

$$\sigma(z) = \frac{1}{1+e^{z}} = \frac{1}{1+\exp(-z)}$$

The very useful sigmoid function



Idea of logistic regression

Compute $w \cdot x + b$, then pass it through the sigmoid function: $\sigma(w \cdot x + b)$.

Treat the result as a probability.

Making probabilities with sigmoids

$$P(y=1) = \sigma(w \cdot x + b)$$

$$= \frac{1}{1 + \exp(-(w \cdot x + b))}$$

$$P(y=0) = 1 - \sigma(w \cdot x + b)$$

$$= 1 - \frac{1}{1 + \exp(-(w \cdot x + b))}$$

$$= \frac{\exp(-(w \cdot x + b))}{1 + \exp(-(w \cdot x + b))}$$

Turning a probability into a classifier

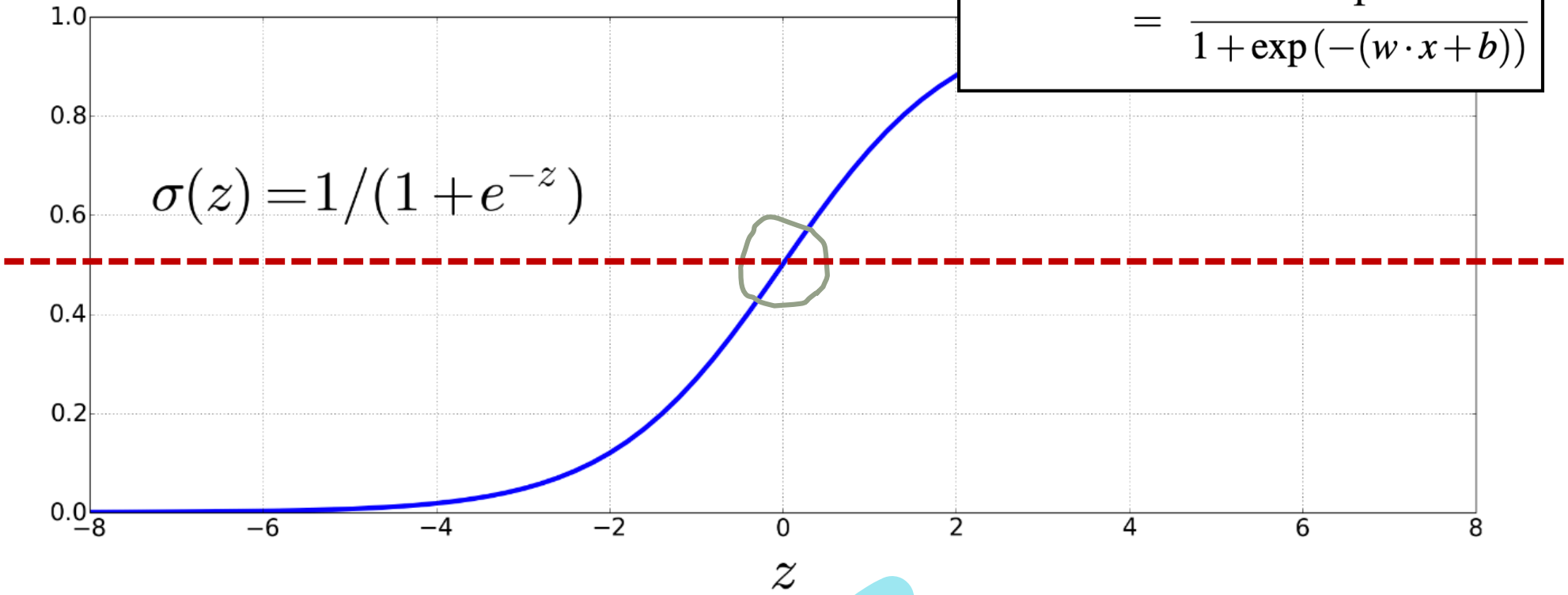
$$\text{decision}(x) = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

0.5 here is called the **decision boundary**

The probabilistic classifier

P(y=1)

$$P(y=1) = \sigma(w \cdot x + b) = \frac{1}{1+\exp(-(w \cdot x + b))}$$



wx + b

-0.1 1 = 0.9

The bias term b can be used to shift the intercept!

Turning a probability into a classifier

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

if $w \cdot x + b > 0$

if $w \cdot x + b \leq 0$

The two phases of logistic regression

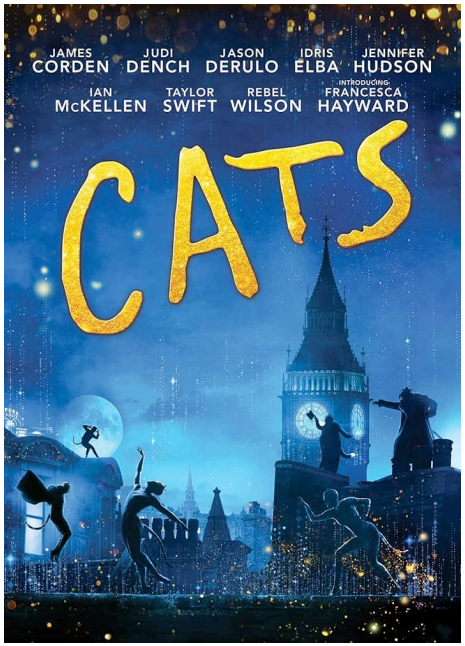
Training: we learn weights *w* and *b* using **stochastic gradient descent** and **cross-entropy loss**.

Test: Given a test example *x* we compute $p(y|x)$ using learned weights *w* and *b*, and return whichever label ($y = 1$ or $y = 0$) is higher probability

Logistic Regression Example: Text Classification

Sentiment example: does y=1 or y=0?

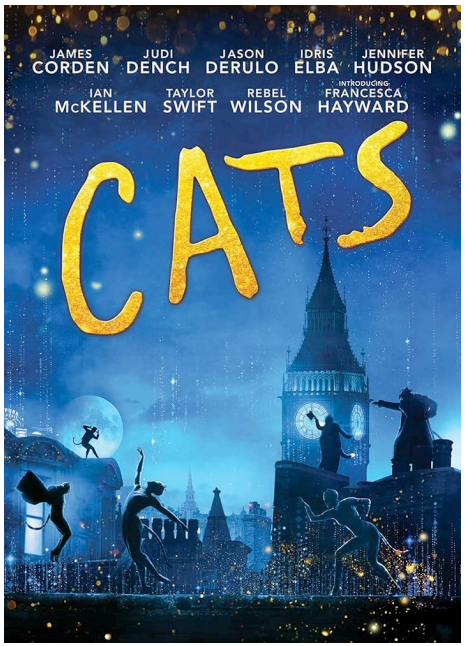
CATS was a marvelous disaster, with witty charm and emotion throughout, cheeky charisma and crying no doubt... I personally went in expecting the worst movie I had ever seen - and it was far more awful and disappointing that I expected.



Sentiment example: does y=1 or y=0?

★½☆☆☆ Verified Jan 12, 2020

CATS was a marvelous disaster, with witty charm and emotion throughout, cheeky charisma and crying no doubt... I personally went in expecting the worst movie I had ever seen - and it was far more awful and disappointing that I expected.



Features

Var	Definition	Value ∈ doc
x_1	count(positive lexicon) ∈ doc	6
x_2	count(negative lexicon) ∈ doc	4
x_3	1 if "no"∈ doc else 0	1
x_4	1 if "!"∈ doc else 0	0
x_5	log(word count of doc)	$\ln(42) = 3.73$

$$X = [6 \quad 4 \quad 1 \quad 0 \quad 3.73]$$

Weights

Var	Definition	Value ∈ doc	W
x_1	count(positive lexicon) ∈ doc	6	2.5
x_2	count(negative lexicon) ∈ doc	4	-5
x_3	1 if "no"∈ doc else 0	1	-1.2
x_4	1 if "!"∈ doc else 0	0	2
x_5	log(word count of doc)	3.73	0.7

Bias b = 0.1

$W = [2.5 \quad -5 \quad -1.2 \quad 2 \quad 0.7]$

Classifying sentiment for input x

$p(+|x) =$ $\sigma(w \cdot x + b)$

$\sigma([2.5 \ -5 \ -1.2 \ 2 \ 0.7] \cdot [6 \ 4 \ 1 \ 0 \ 3.7] + 0.1)$

$\sigma(-1.78) = 0.14$

$p(-|x) =$

$1 - 0.14 = 0.86$

Classification in (binary) logistic regression: summary

Given:

- a set of classes: (+ sentiment,- sentiment)
- a vector **x** of features $[x_1, x_2, \ldots, x_n]$
  - x1= count("awesome")
  - x2 = log(number of words in review)
- A vector **w** of weights $[w_1, w_2, \ldots, w_n]$
  - w_i for each feature f_i

$$P(y=1) = \sigma(w \cdot x + b)$$
$$= \frac{1}{1+\exp(-(w \cdot x + b))}$$

Feature Representations

Example task: period disambiguation

This ends in a period.
The house at 465 Main St. is new.

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

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

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

Multi-class Regression

Multinomial Logistic Regression

Often we need more than 2 classes

- Positive/negative/neutral
- Parts of speech (noun, verb, adjective, adverb, preposition, etc.)
- Classify emergency SMSs into different actionable classes

If >2 classes we use **multinomial logistic regression**

= Softmax regression

= Multinomial logit

= Maximum entropy modeling or MaxEnt

So "logistic regression" means binary (2 classes)

Multinomial Logistic Regression

The probability of everything must still sum to 1

```
P(positive|doc) + P(negative|doc) +
P(neutral|doc) = 1
```

Need a generalization of the sigmoid called **softmax**

- Takes a vector $z = [z_1, z_2, ..., z_k]$ of k arbitrary values
- Outputs a probability distribution

The softmax function

Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of k arbitrary values into probabilities

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \qquad 1 \le i \le k$$

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k} \exp(z_i)} \quad \cdots \quad \frac{\exp(z_k)}{\sum_{i=1}^{k} \exp(z_i)} \right]$$

The softmax function

Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of k arbitrary values into probabilities

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

The denominator $\sum_{i=1}^{K} \exp(\mathbf{z}_i)$ is used to normalize all the values into probabilities.

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

The softmax function

Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of k arbitrary values into probabilities :

$$z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

$$\text{softmax}(z) =$$

The softmax function

Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of k arbitrary values into probabilities :

$$z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

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

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

Softmax in multinomial logistic regression

$$p(y=c|x) = \frac{\exp(w_c \cdot x + b_c)}{\sum_{j=1}^{K} \exp(w_j \cdot x + b_j)}$$

Input is still the dot product between weight vector w and input vector x, but now we need separate weight vectors for each of the K classes.

Features in binary versus multinomial logistic regression

Binary: positive weight

$$x_5 = \begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases} \qquad w_5 = 3.0$$

Multinominal: separate weights for each class:

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