RICE SOCIAL SCIENCES
Department of Linguistics

logistic regression

LING 430 Computational Linguistics Fall 2026
lecture 8b
Oct 15 2026

Siyu Liang
siyu.liang@rice.edu

last time

- text classification
- Naive Bayes
  - bag of words
  - prior × likelihood
  - add-one smoothing

agenda

- logistic regression
  - features & weights
  - the sigmoid
  - loss function
  - gradient descent

classification (revisit)

classification

- **classification**: assigning each input one label from a fixed set
  - spam detection
  - sentiment analysis
  - authorship identification
  - language identification
- even language modeling is classification
  - the classes are the words of the vocabulary
  - predicting the next word is classifying the context so far into a class for each possible next word

classification

supervised learning

- **training**
  - input: a fixed set of classes, and m hand-labeled examples $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$
  - output: a learned classifier
- **testing**, also called **inference**
  - input: a document d
  - output: a predicted class $\hat{y}$
- the hat on $\hat{y}$ marks a predicted value, as opposed to the true y

classification

many kinds of classifiers

- Naive Bayes: last week
- logistic regression: today
- neural networks: week 12
- k-nearest neighbors, decision trees, …
- large language models, fine-tuned as classifiers or prompted to give a classification

logistic regression

logistic regression

- an important analytic tool in the natural and social sciences
- the baseline supervised machine learning tool for classification
- the foundation of neural networks

logistic regression

components

- four components of a probabilistic classifier
  - a **feature representation** of the input: a vector x of numbers
  - a **classification function** that computes $\hat{y}$ from x: the sigmoid
  - a **loss function**: how far $\hat{y}$ is from the true y
  - an algorithm for **learning the weights**: gradient descent

logistic regression

two phases

- **training**
  - learn the weights w and the bias b from labeled examples with a loss and gradient descent
- **testing**, or **inference**
  - given an input x, compute $P(y|x)$ with weights that are already learned, and return the more probable label

features & weights

features

- each feature x_i is a number computed from the document
- each feature has a **weight** w_i that says how much it matters, and in which direction
- e.g. to classify positive movie reviews:
  - x_i = the review contains *awesome*: $w_i = +10$
  - x_j = the review contains *abysmal*: $w_j = -10$
  - x_k = the review contains *mediocre*: $w_k = -2$
- Naive Bayes gave every word a vote sized by its counts
  - here, the weight of a vote matters for logistic regression

features

where to get them

- we could manually make features
  - period disambiguation: is this period the end of a sentence?
    - *This ends in a period. vs. The house at 465 Main St. is new.*
    - we can manually make features
      - the next word starts with a lower-case letter
      - the word before the period is a known abbreviation
      - the word before the period is one letter long
- while they're helpful, they don't scale
  - **representation learning**: learning features automatically



ICLR
International Conference On
Learning Representations

features

how to classify

- multiply each feature by its weight, add them up, add a bias b

$$z = \sum_i w_i x_i + b = w \cdot x + b$$

  - if the sum z is high $\rightarrow$ output $y = 1$
  - if it is low $\rightarrow$ output $y = 0$
- this is the classifier!
  - the rest of class is to learn how to turn z into a probability for the output and where w comes from

the sigmoid

the sigmoid

- $z = w \cdot x + b$ can be any number from $-\infty$ to $+\infty$
- it is a score, not a probability
- we want $P(y = 1|x)$, a number between 0 and 1, as Naive Bayes gave us
- solution: pass z through a function that goes from 0 to 1
  - the sigmoid function, also logistic function

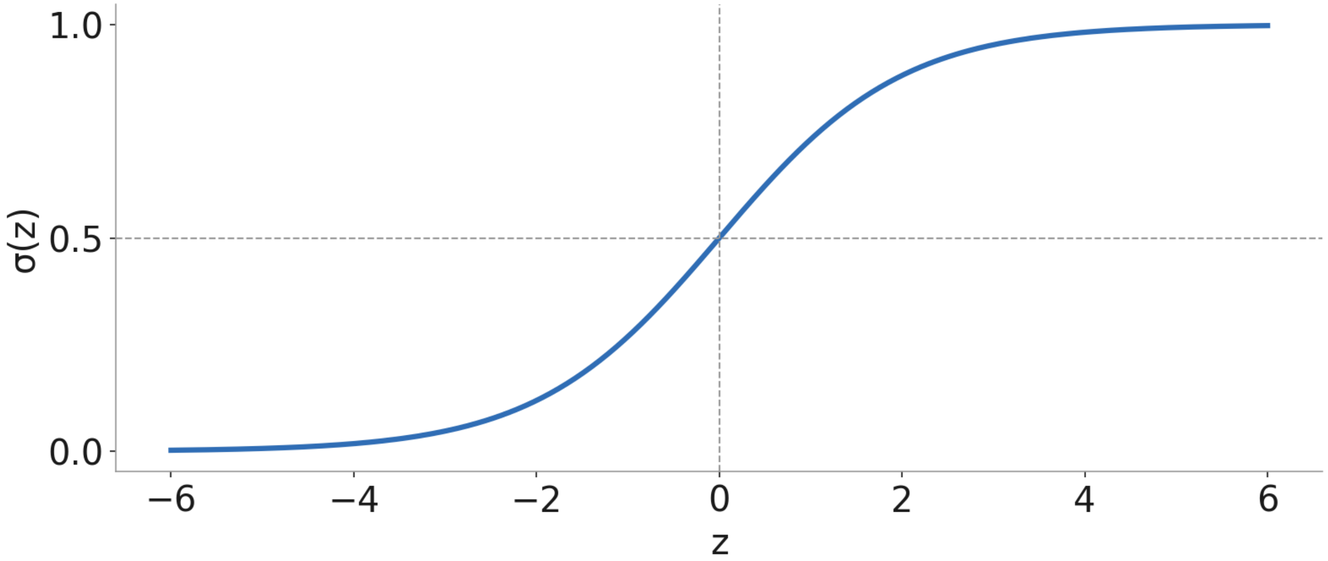
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

the sigmoid

characteristics of the sigmoid

- steep near 0, flat at the extremes
- $\sigma(0) = 0.5$
- a large score saturates near certainty

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



the sigmoid

- with the sigmoid, we can get the probability of the positive class
  - and with that, the negative class, too
  - the two add up to 1, as probabilities must

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

the sigmoid

the decision

- with the probabilities, we can then turn the them into output labels

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \quad \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \quad \text{if } w \cdot x + b \leq 0 \end{cases}$$

- 0.5 is the **decision boundary**

an example

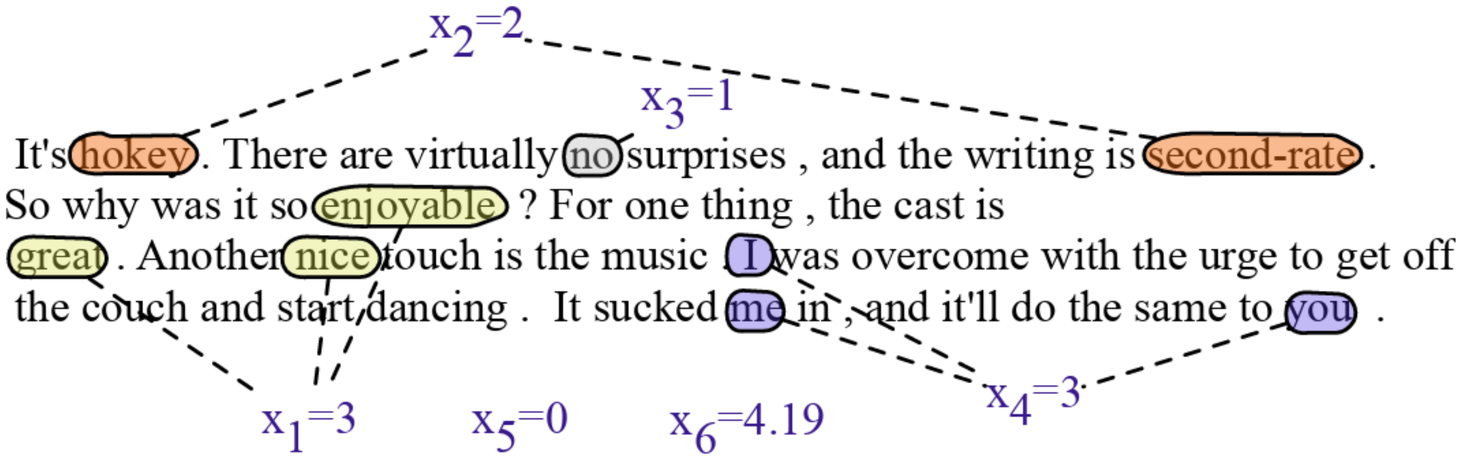
positive or negative?

- *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.*

an example

1. the features

- we start with the features (hand-picked for now)



feature	what it counts	value
x_1	positive lexicon words: *enjoyable, great, nice*	3
x_2	negative lexicon words: *hokey, second-rate*	2
x_3	*no* occurs in the review	1
x_4	first and second person pronouns	3
x_5	*!* occurs in the review	0
x_6	log of the word count, ln(66)	4.19

an example

2. the weights

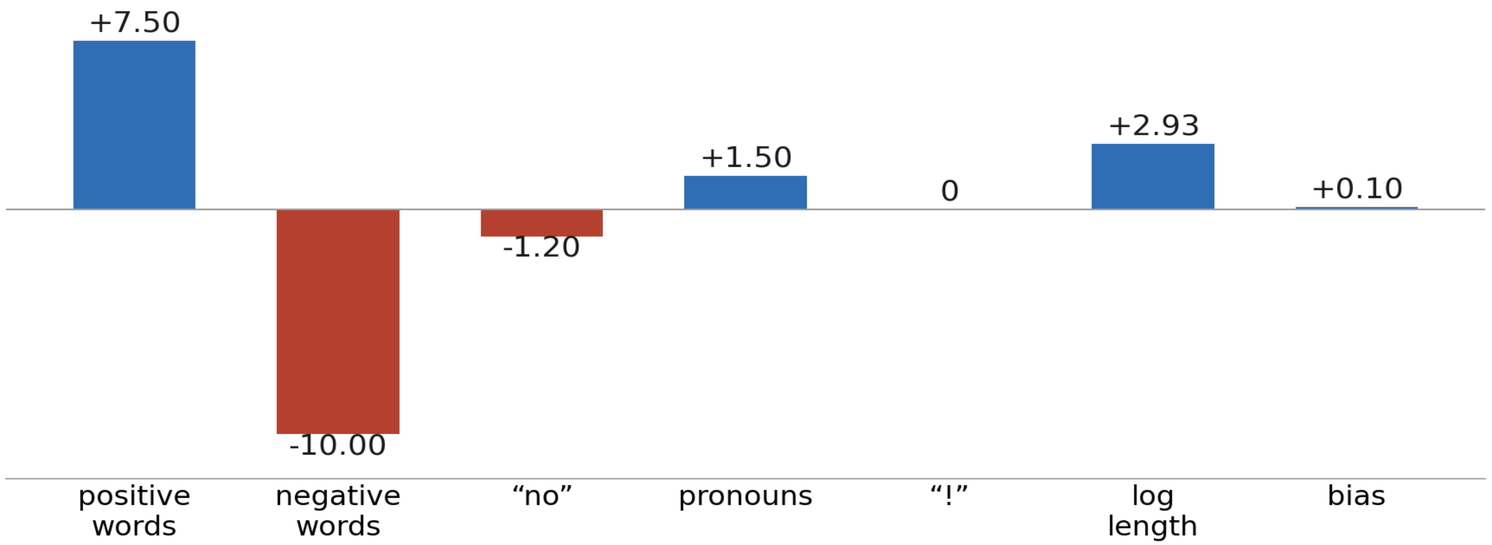
- suppose we have the weights w and bias b learned from labeled reviews
  - calculate each feature's contribution $w \cdot x$

feature	what it counts	x	w	$w \cdot x$	b
x_1	positive lexicon words	3	2.5	7.5	
x_2	negative lexicon words	2	–5.0	–10.0	
x_3	*no* occurs	1	–1.2	–1.2	
x_4	1st and 2nd person pronouns	3	0.5	1.5	
x_5	*!* occurs	0	2.0	0	
x_6	log of the word count	4.19	0.7	2.93	
bias					0.1

an example

3. the score

- add up the contributions and the bias
  $z = 7.5 - 10.0 - 1.2 + 1.5 + 0 + 2.93 + 0.1 = 0.833$



feature	x	w	$w \cdot x$	b
x_1	3	2.5	7.5	
x_2	2	–5.0	–10.0	
x_3	1	–1.2	–1.2	
x_4	3	0.5	1.5	
x_5	0	2.0	0	
x_6	4.19	0.7	2.93	
bias				0.1

an example

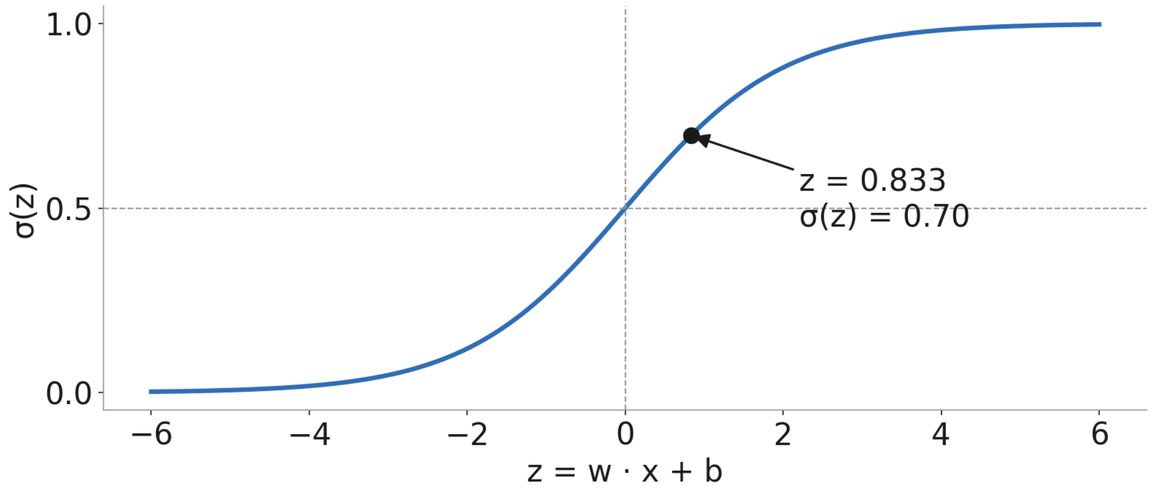
4. the probability

- pass the score through the sigmoid

$$P(+) = \sigma(0.833) = \frac{1}{1 + e^{-0.833}} = 0.70$$

$$P(-) = 1 - 0.70 = 0.30$$

- 0.70 is above 0.5
  - predict + positive



activity 1

- use the same six features and weights to predict a new review
  - *Nice cast and great music, but a dull plot and a boring ending. We laughed anyway, and you might too!*
  - 1. the six feature values
    - the review has 20 words, ln(20) = 3
  - 2. the score z
  - 3. $\sigma(z)$
- which label, and how sure is the model?

feature	w	b
x_1 positive words	2.5	
x_2 negative words	–5.0	
x_3 *no* occurs	–1.2	
x_4 pronouns	0.5	
x_5 *!* occurs	2.0	
x_6 log word count	0.7	
bias		0.1

logistic regression

learning

- so far the weights were handed to us
- real weights are **learned** from labeled examples
- two components
  - a **loss function**: how bad the current prediction is
  - an **optimization algorithm**: how to change the weights to make it less bad

loss function

- we need a number that says how bad a prediction is
- what should it depend on?
  - the probability the model gave to the true label
  - if $y = 1$, that probability is $\hat{y}$
  - if $y = 0$, it is $1 - \hat{y}$
- we want the loss small when that probability is high, and large when it is low
- a negative log can do this

$$L = -\log P(y|x)$$

loss function

- how does –log behave?
  - a confident and right prediction costs almost nothing
  - a confident and wrong one costs a lot

$$L = -\log P(y|x)$$

***P*(y|x)**	**loss**
0.9	0.11
0.7	0.36
0.5	0.69
0.1	2.30
0.01	4.61

loss function

- the model said $P(+) = 0.70$, what was the loss?
  if $y = 1$: $L = -\log 0.70 = 0.36$
  if $y = 0$: $L = -\log 0.30 = 1.20$
- loss is larger when the model was wrong
- and it would be larger still had the model been both confident and wrong

loss function

- we can write the two cases into one formula

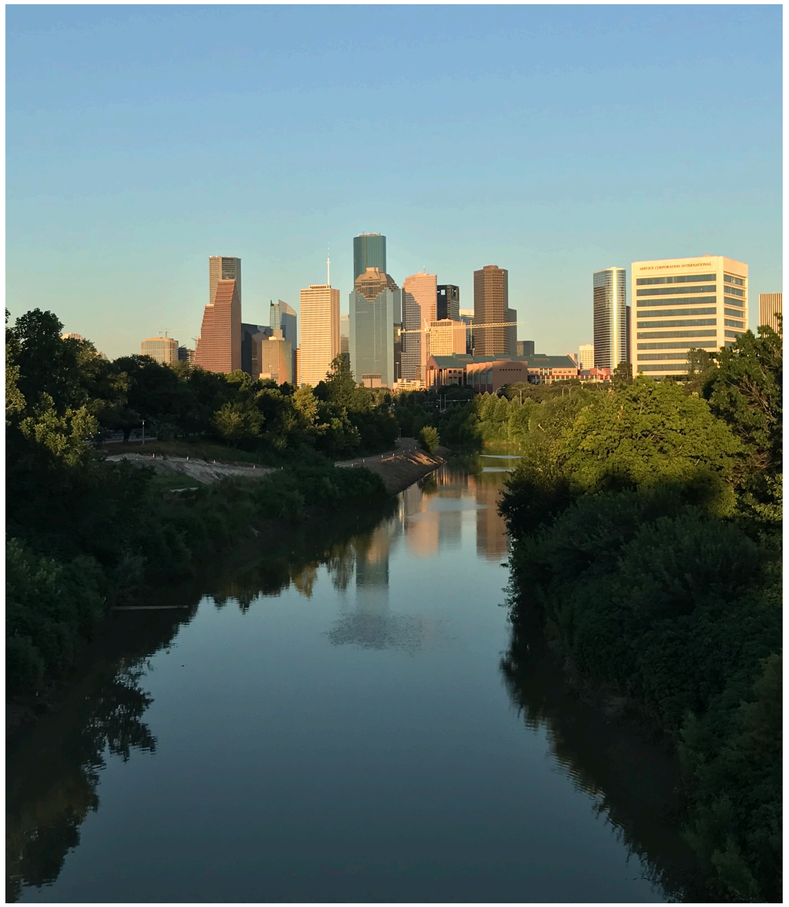
$$L = -[y \log\hat{y} + (1 - y)\log(1 - \hat{y})]$$

- when $y = 1$ only the first term is left, when $y = 0$ only the second
- this is the **cross-entropy loss**
- our next step: how to choose the weights that make the loss smallest

gradient descent

bayou descent

- how do we get to the bottom of a bayou?
  - look around 360°
  - find the direction of steepest slope down
  - go that way, a little



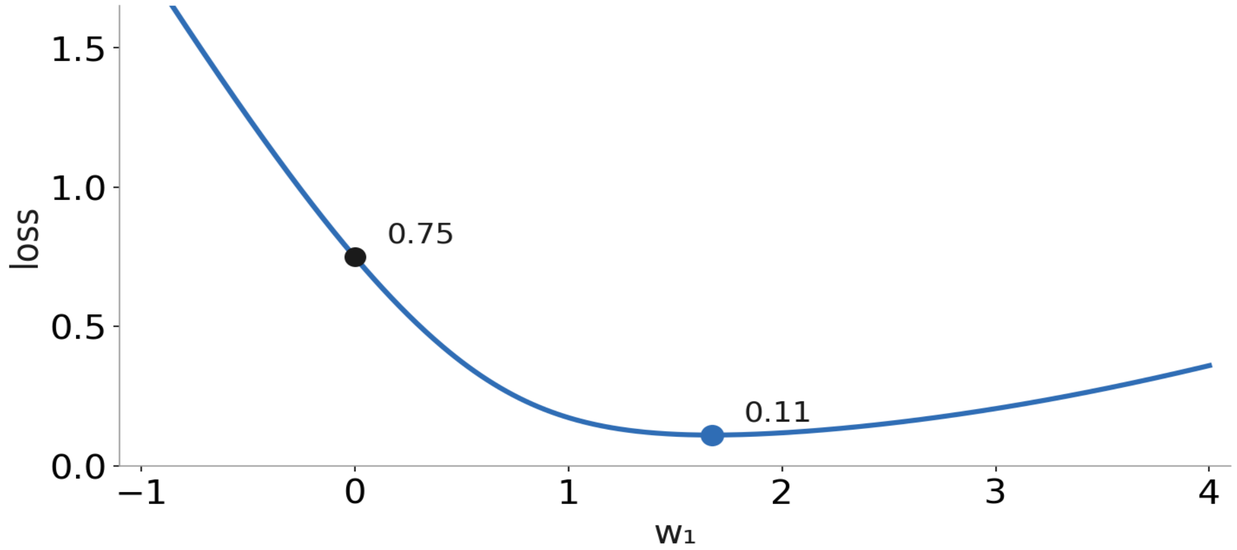
gradient descent

- the loss is a function of the weights
  - change the weights, and the loss and prediction also change
- e.g. four short reviews, two features
  - suppose we hold $w_2 = -1$ and $b = 0$ fixed, and try to only determine w_1

review	x_1 **positive words**	w_1	x_2 **negative words**	w_2	y
hokey … enjoyable, great, nice	3	-1	2	?	1
great cast, nice music	2	-1	0	?	1
the worst, hated it	0	-1	2	?	0
fine cast, dull, boring, long	1	-1	3	?	0

gradient descent

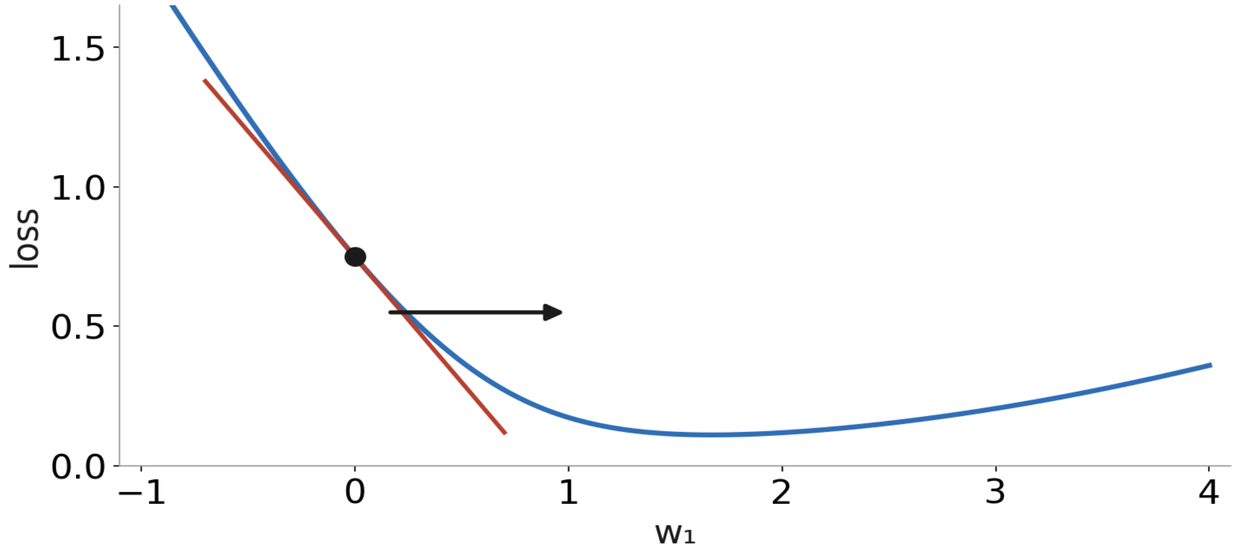
- the loss of logistic regression is **convex**
  - one bottom and no other dips
- the bottom is at $w_1 = 1.67$, loss 0.11, but we don't know this yet!
- suppose we start at $w_1 = 0$, loss = 0.75



gradient descent

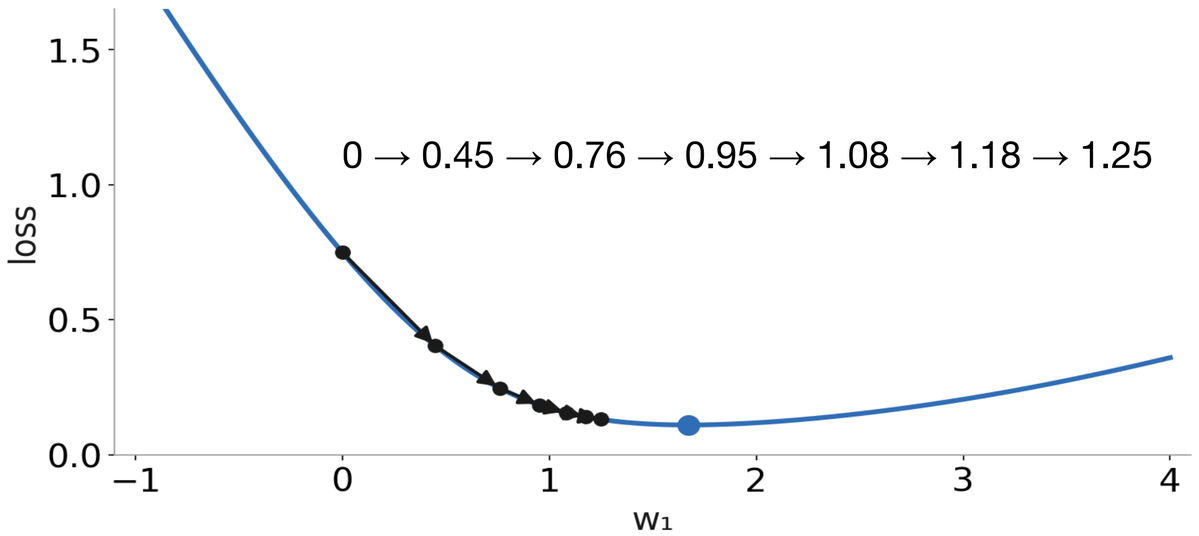
- standing at $w_1 = 0$, should w_1 get bigger or smaller?
- we can check the **slope** of the curve where we stand
  - –0.90, the loss falls as the weight grows
- so move down the slope, increase w_1





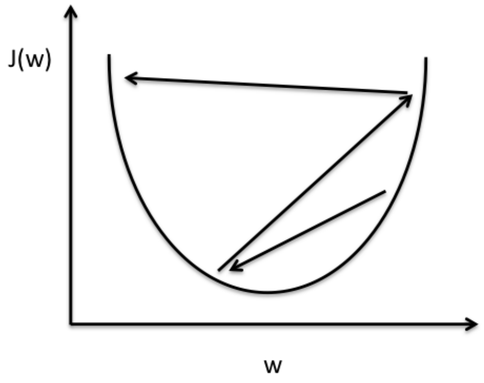
gradient descent

- we move by a step size $\eta = 0.5$
  - new $w_1 = w_1 - \eta \times$ slope
  - repeat
- the steps get shorter as the slope flattens near the bottom
- **gradient descent**: walk downhill on the loss, one step at a time, until the slope approximates zero

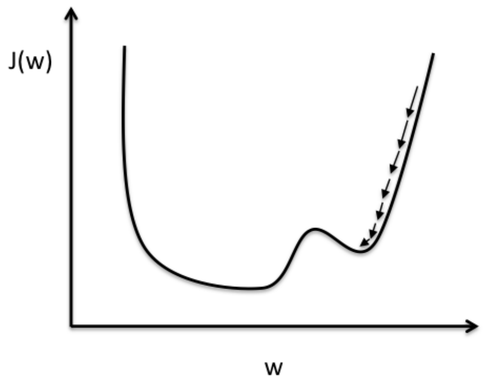


gradient descent

- the **gradient** is the slope of the loss
- the **descent** is the steps we take to approximate the lowest loss
- which way should a weight move?
  - against the slope
- how far? a small step η, the **learning rate**
  - too high overshoots, too low takes forever



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

https://medium.com/codex/gradient-descent-cb0f02dc6eab

gradient descent

- for logistic regression the slope has a simple form
$$\frac{\partial L}{\partial w_j} = (\hat{y} - y)x_j$$
  - the **slope** equals the **error** $\hat{y} - y$, times the **feature** x_j
- so the update for each weight is $w_j \leftarrow w_j - \eta(\hat{y} - y)x_j$
- a feature that is 0 in this example gets no update, whereas a large error moves the weights a lot

gradient descent example

- the same *hokey* review, with only two of its features
  - the true label is $y = 1$
- start with no opinions: every weight 0, and the learning rate $\eta = 0.1$

feature	what it counts	x	w at the start
x_1	positive lexicon words	3	0
x_2	negative lexicon words	2	0
bias	always 1	1	0

gradient descent example

1. predict

- with every weight at 0, the score is 0
  $\hat{y} = \sigma(0 \cdot 3 + 0 \cdot 2 + 0) = \sigma(0) = 0.5$
- a coin flip; the true label is 1, so the **error** is
  $\hat{y} - y = 0.5 - 1 = -0.5$

feature	what it counts	x	w at the start
x_1	positive lexicon words	3	0
x_2	negative lexicon words	2	0
bias	always 1	1	0

gradient descent example

2. the slopes

- the slope of the loss for each weight is the error times that weight's feature

feature	x	error $\hat{y} - y$	slope = error × x
x_1	3	–0.5	–1.5
x_2	2	–0.5	–1.0
bias	1	–0.5	–0.5

gradient descent example

3. update

- move each weight against its slope, by $\eta = 0.1$ times the slope

weight	old	slope	$\eta \times$ slope	new = old – $\eta \times$ slope
w_1	0	–1.5	–0.15	0.15
w_2	0	–1.0	–0.10	0.10
b	0	–0.5	–0.05	0.05

gradient descent example

4. check

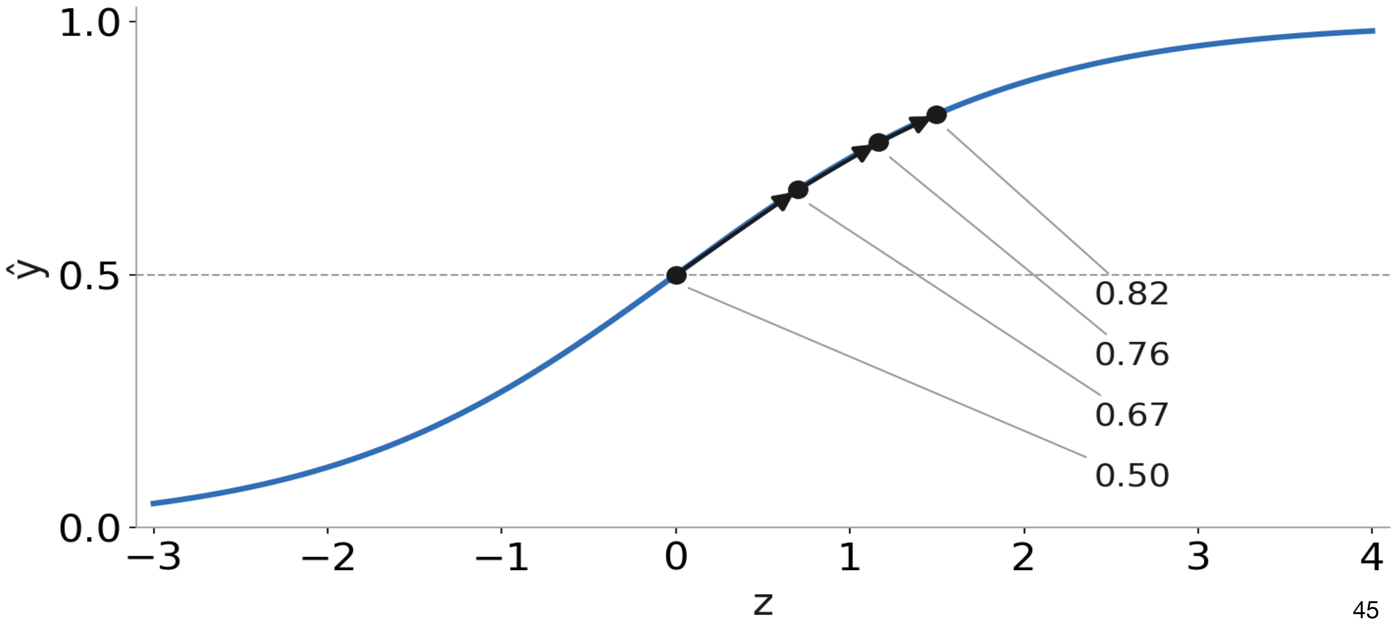
- predict again with the new weights
  $z = 0.15 \cdot 3 + 0.10 \cdot 2 + 0.05 = 0.70$
  $\hat{y} = \sigma(0.70) = 0.67$
- from 0.50 to 0.67: the prediction moved toward the label!

gradient descent example

5. more steps

- the training continues by repeating this over every example
- each step uses the new weights and the new, (hopefully) smaller error
- the prediction climbs the sigmoid toward the label, the steps shrink as the error shrinks

step	w_1	w_2	b	z	$\hat{y}$	error
0	0	0	0	0	0.50	–0.50
1	0.15	0.10	0.05	0.70	0.67	–0.33
2	0.25	0.17	0.08	1.17	0.76	–0.24
3	0.32	0.21	0.11	1.50	0.82	–0.18



gradient descent

in practice

- **stochastic gradient descent**: updating weights and biases in a semi-random manner
- **mini-batch** training: the gradient over a batch of examples
- this loop of prediction, updates, and repetition is the same loop that trains models such as large language models with billions of weights

large language models

- What's the main difference between these two groups?







VS.







large language models

- the key component of a trained model is just a file of numbers for its weights and biases
  - a logistic regression model for movie review classification can have tens of thousands of weights and one bias
  - deepseek-v3: ~ 671 billion
  - llama-3: up to 405 billion
- **open-weight** models publish that file: LLaMA (Meta), Gemma (Google), Mistral, DeepSeek
  - you can run these locally
- **closed-weight** models do not: GPT-4 and later, Claude, Gemini
  - you send something to a server and get something back

next time

- next week
  - hidden Markov models
  - Viterbi
  - part-of-speech tagging
- **Take the anonymous midterm course survey**
  - helps me prepare the course to help you learn better for the second half!
- **HW 4 due Tuesday, Oct 20, 11:59 pm**