

BUS 243

Lecture 5: Discriminative classifiers

PROBLEM SET REVIEW

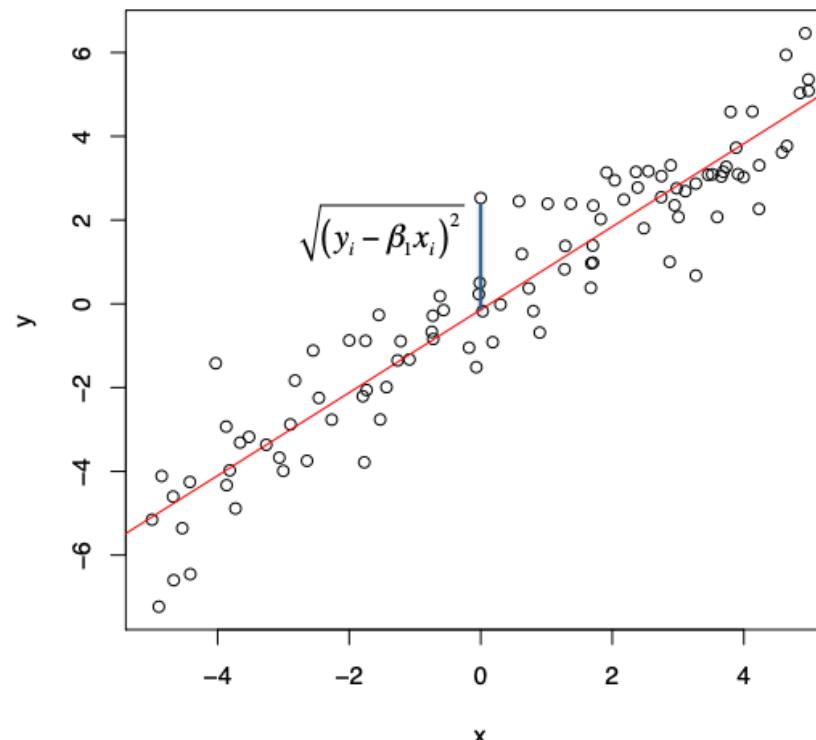
- Purpose of Tokenization?
- One hot vector vs. BOW vs. Incidence Matrix
- Homework 3 would be more difficult



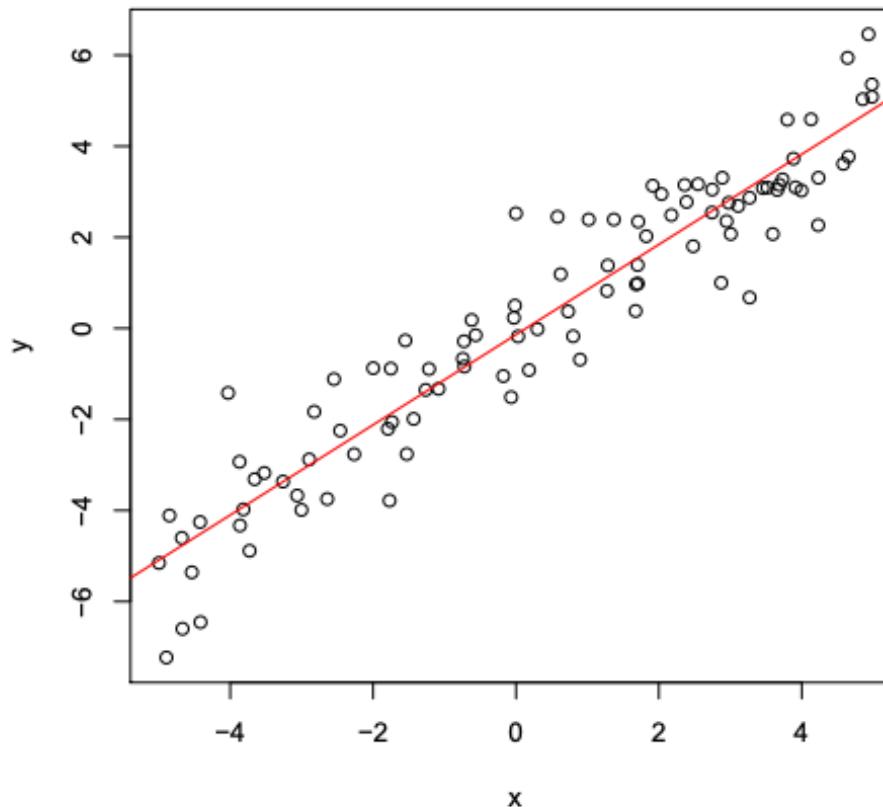
FITTING A LINEAR REGRESSION

- Idea: minimize the Euclidean distance between data and fitted line

- $$RSS(\beta) = \frac{1}{n} \sum_i (y_i - \beta_1 x_i)^2$$

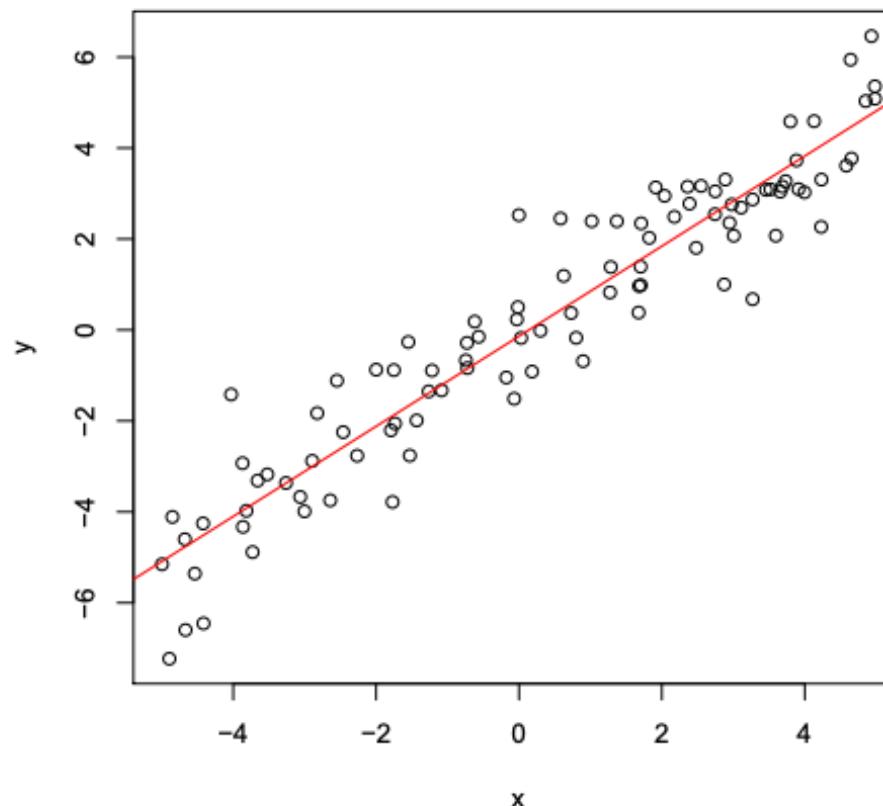


- How to find β
 - Use calculus to find the value of β that minimizes the RSS

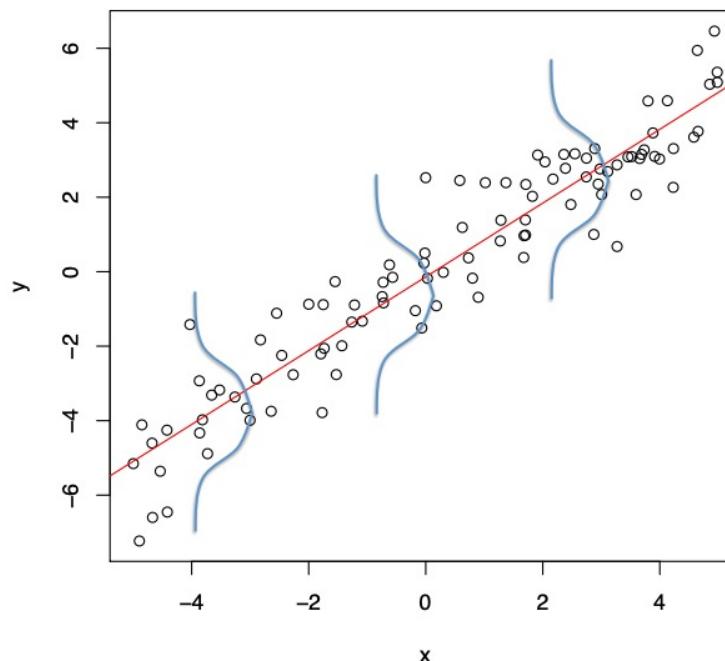


- Probabilistic Interpretation

- Our analysis so far has not included any probabilities
- Linear regression does have a probabilistic (probability model-based) interpretation



- Linear regression assumes that response values have a Gaussian distribution around the linear mean function
 - $Y_i | \mathbf{x}_i, \beta \sim N(\mathbf{x}_i \beta, \sigma^2)$
- This is discriminative model where inputs \mathbf{x} are not modeled



- Minimizing RSS is equivalent to maximizing conditional likelihood



CLASSIFICATION: LOGISTIC REGRESSION

- What are we talking about?
 - Statistical classification: $P(y|x)$
 - Examples?
 - Discriminative classifiers vs. generative classifier
 - Building block of other machine learning methods



GENERATIVE AND DISCRIMINATIVE CLASSIFIERS

- Suppose we're distinguishing cat from dog images



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



- Oh, look. Dogs have collars! Let's ignore everything else



LOGISTIC REGRESSION

- Discriminative classifiers learn what features from the input are most useful to discriminate between the different possible classes!
- Train a classifier that can make a binary decision about the class of a new input
- To create a probability, we need to transform!



- Train a classifier that can make a binary decision about the class of a new input

- Weight vector β_i with bias (intercept) β_0

- Observations X_i

- $$P(Y = 0|X) = \frac{1}{1 + \exp(\beta_0 + \sum_i \beta_i X_i)}, P(Y = 1|X) = \frac{\exp(\beta_0 + \sum_i \beta_i X_i)}{1 + \exp(\beta_0 + \sum_i \beta_i X_i)}$$

- For shorthand, we'll say that

- $$P(Y = 0|X) = \sigma(-(\beta_0 + \sum_i \beta_i X_i))$$

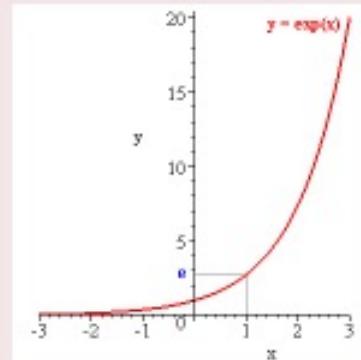
- $$P(Y = 1|X) = 1 - \sigma(-(\beta_0 + \sum_i \beta_i X_i))$$

- $$\text{Where } \sigma(z) = \frac{1}{1 + \exp(-z)}$$

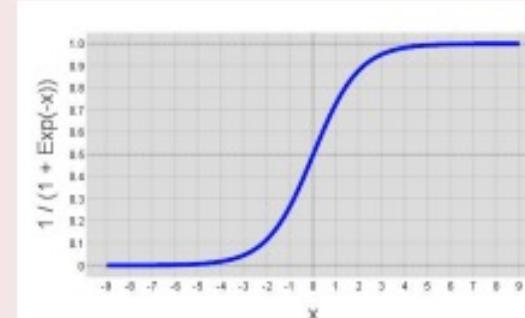


Exponential

- What is this “exp” doing?
 - e is a special number, 2.71...
 - e^x is the limit of compound interest formula as compounds become infinitely small
 - It's the function whose derivative is itself
 - The logistic function is $\sigma(z) = \frac{1}{1+e^{-z}}$
 - S-shape and between 0 and 1
 - Model probabilities



Logistic



SENTIMENT EXAMPLE: Y=1 OR Y=0?

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 .



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** .

$x_1=3$ $x_5=0$ $x_6=4.19$ $x_2=2$ $x_3=1$ $x_4=3$

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	$\log(\text{word count of doc})$	$\ln(66) = 4.19$



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

$$b = 0.1$$



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

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



■ How is Logistic Regression Used?

- Given a set of weights $\vec{\beta}$, we know how to compute the conditional likelihood $P(y|\beta, x)$
- Find the set of weights $\vec{\beta}$ that maximize the conditional likelihood on training data (soon!)
- Intuition: higher weights mean that this feature...
 - this is the class you want for this observation
- Naïve Bayes is a special case of logistic regression that uses Bayes rule and conditional probabilities to set these weights
 - $\text{Argmax}_{c_j} [\ln \hat{P}(c_j) + \sum_i \ln \hat{P}(w_i|c_j)]$



▪ Contrasting Naïve Bayes and Logistic Regression

- Naïve Bayes easier
- Naïve Bayes better on smaller datasets
- Logistic regression better on medium-sized datasets
- On huge datasets, it does not really matter (data always win)
- Logistic regression allows arbitrary features (biggest difference!)
- Do not memorize any formula...



- Wait, where did the β 's come from?
 - Supervised classification: correct label y (either 0 or 1) for each x
 - But what the system produces is an estimate \hat{y}
 - We need
 - a loss function or a cost function
 - An optimization algorithm to update β s to minimize the loss



- A loss function
- Cross-entropy loss
- An optimization algorithm
- Stochastic gradient descent
- Don't worry: see the code



LOSS FUNCTION

- We want to know how far is the classifier output from the true output
 - $L(\hat{y}, y)$: how much they are different
- Since there are only two discrete outcomes, this is a Bernoulli distribution
 - $p(y|x) = \hat{y}^y(1 - \hat{y})^{1-y}$
 - Take the log of both sides, flip the sign, and plug in the definition of \hat{y}
 - $L_{CE} = -[y \log \sigma(wx + b) + (1 - y) \log(1 - \sigma(wx + b))]$
- Want to compare the distribution, not the single values



- Let's see if this works for our sentiment example
- 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 .



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

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

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

What's the loss?

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



- Suppose true value instead was $y=0$

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

What's the loss?

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

Sure enough, loss was bigger when model was wrong!



- Goal: minimize the loss
 - consider weights $\theta = (w, b)$
 - We'll represent \hat{y} as $f(x; \theta)$ to make the dependence on θ more obvious
 - We want the weights that minimize the loss, averaged over all examples

- $\hat{\theta} = \operatorname{argmin}_{\theta} \frac{1}{m} \sum_i L_{CE}(f(x^{(i)}; \theta), y^{(i)})$

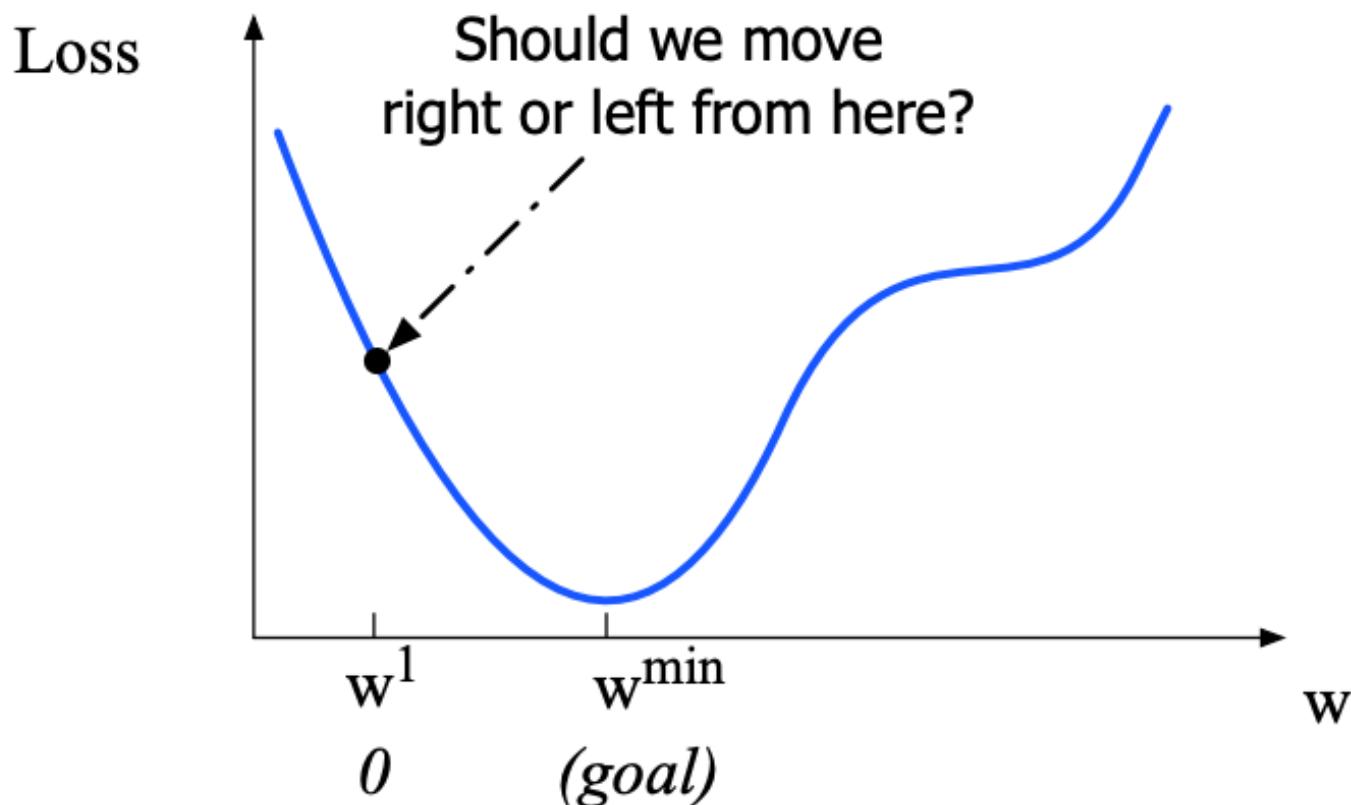


- For logistic regression, loss function is convex
 - A convex function has just one minimum
 - Gradient descent starting from any point is guaranteed to find the minimum



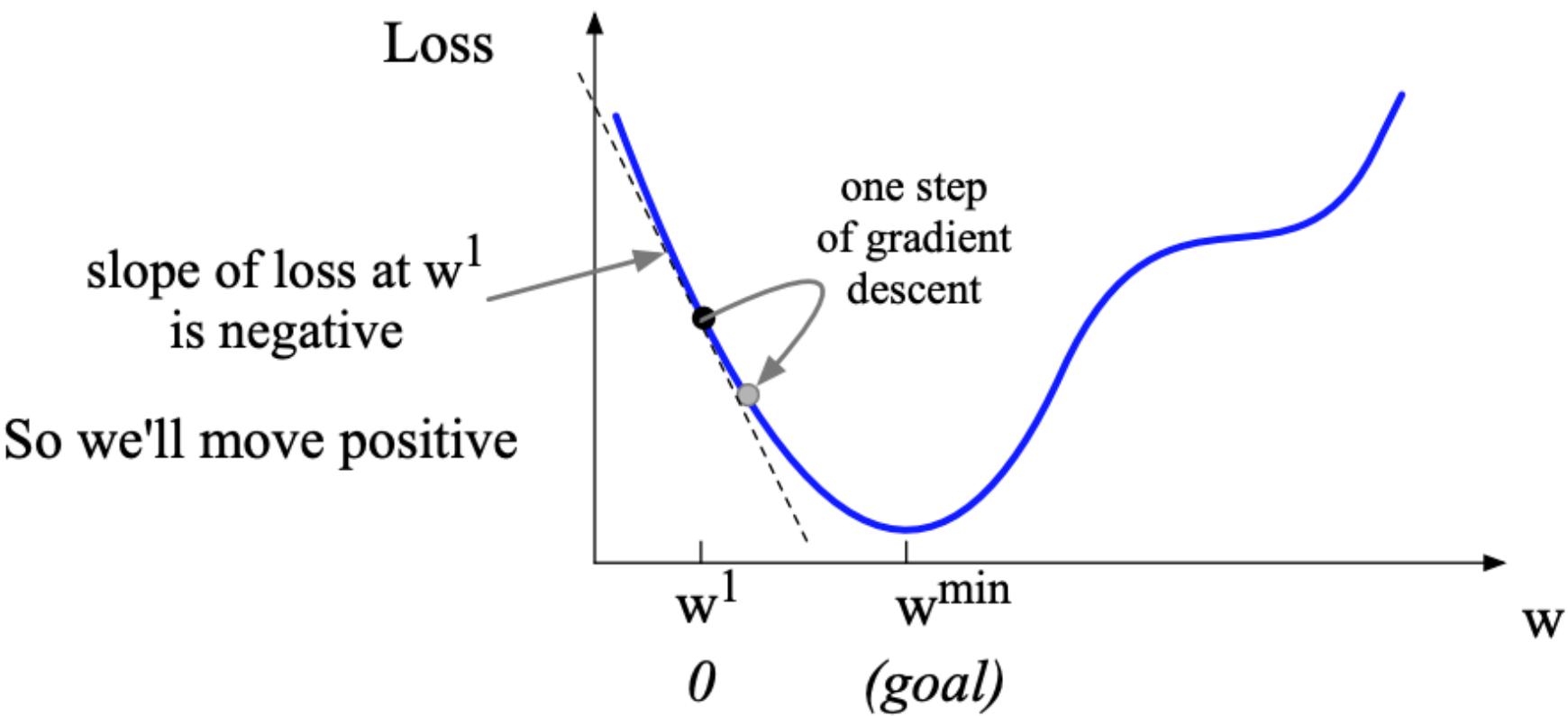
Q: Given current w , should we make it bigger or smaller?

A: Move w in the reverse direction from the slope of the function



Q: Given current w , should we make it bigger or smaller?

A: Move w in the reverse direction from the slope of the function



- The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function

- **Gradient Descent**

- Find the gradient of the loss function at the current point and move in the **opposite** direction



■ How much do we move in that direction?

- The value of the gradient (slope in our example) $\frac{d}{dw} L(f(x; w), y)$

weighted by a learning rate η

- Higher learning rate means move w faster

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



- Now let's consider N dimensions
 - We want to know where in the N-dimensional space (of the N parameters that make up θ) we should move
 - The gradient is just such a vector; it expresses the directional components of the sharpest slope along each of the N dimensions

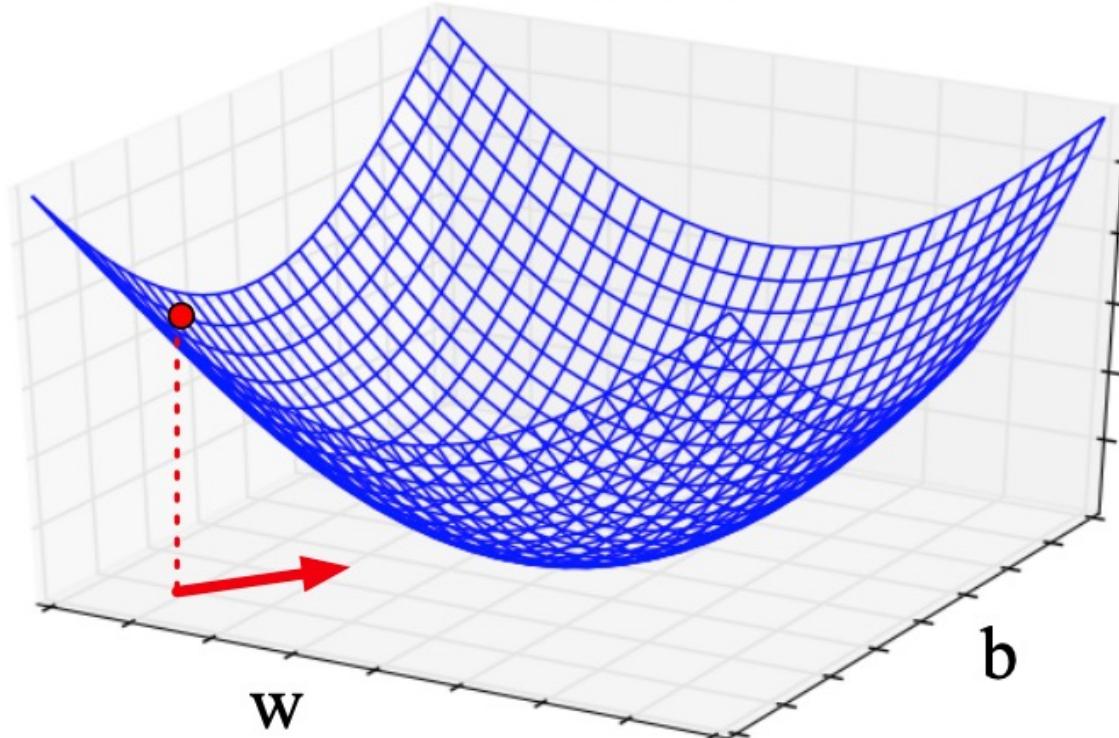


Imagine 2 dimensions, w and b

Visualizing the
gradient vector at
the red point

It has two
dimensions shown
in the x-y plane

$\text{Cost}(w,b)$



We'll represent \hat{y} as $f(x; \theta)$ to make the dependence on θ more obvious:

$$\nabla_{\theta} L(f(x; \theta), y) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x; \theta), y) \\ \frac{\partial}{\partial w_2} L(f(x; \theta), y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x; \theta), y) \end{bmatrix}$$

The final equation for updating θ based on the gradient is thus

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x; \theta), y)$$



The loss function

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

The elegant derivative of this function (see textbook 5.8 for derivation)

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



EXAMPLE

- One step of gradient descent
- A mini-sentiment example, where the true $y=1$ (positive)
- Two features
 - $x_1 = 3$ (count of positive lexicon words)
 - $x_2 = 2$ (count of negative lexicon words)
- Assume 3 parameters (2 weights and 1 bias) in Θ^0 are zero:
 - $w_1 = w_2 = b = 0$
 - $\eta = 0.1$



- Update step for update θ is:

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x; \theta), y)$$

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

Gradient vector has 3 dimensions:

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial b} \end{bmatrix}$$



- Update step for update θ is:

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x; \theta), y)$$

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

Gradient vector has 3 dimensions:

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix}$$



- Update step for update θ is:

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x; \theta), y)$$

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

Gradient vector has 3 dimensions:

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix}$$



- Update step for update θ is:

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x; \theta), y)$$

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

Gradient vector has 3 dimensions:

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y}, y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$



$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_1} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial w_2} \\ \frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial b} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ \sigma(w \cdot x + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} -0.5x_1 \\ -0.5x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

Now that we have a gradient, we compute the new parameter vector θ^1 by moving θ^0 in the opposite direction from the gradient:

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x; \theta), y) \quad \eta = 0.1;$$

$$\theta^1 = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} .15 \\ .1 \\ .05 \end{bmatrix}$$

Note that enough negative examples would eventually make w_2 negative

