Lecture 6: Text Classification (II)

Instructor: Stephanie Schoch

CSCI 680: Natural Language Processing
Department of Computer Science, College of William & Mary
September 9, 2026

WILLIAM & MARY

Many slides are adapted from Jurafsky & Martin (3rd ed.)

Lecture Outline

- Logistic Regression
  - Overview
  - Cross-Entropy
  - Gradient Descent
  - Evaluating Text Classification (Friday)
  - Harms of Text Classification (Friday)

Logistic Regression Classification

Components of Supervised Machine Learning

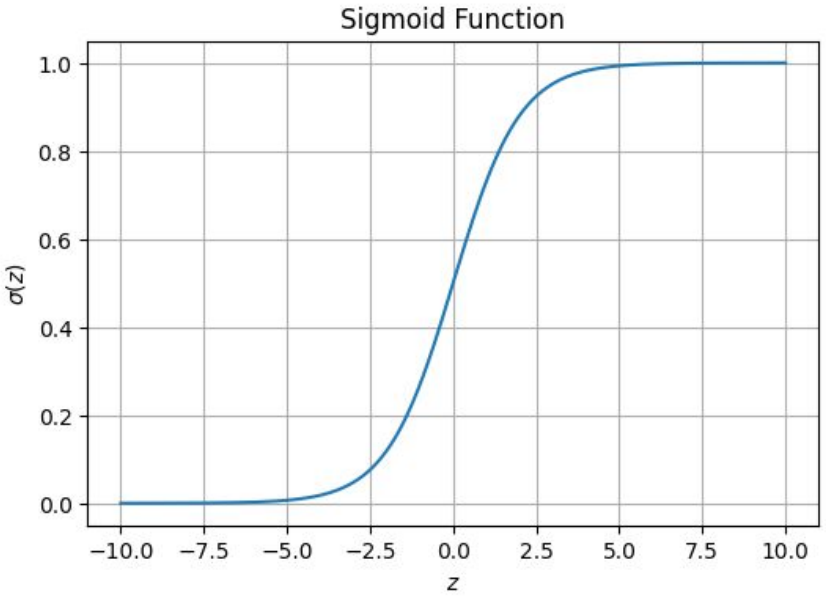
Training / Learning

- **Data** as N input/output pairs $(x^{(i)}, y^{(i)})$, $i \in \{ 1, \ldots, N \}$
  a. A **feature representation** of the input. Typically represented by a feature vector
     $\mathbf{x}^{(i)} = [x_1, x_2, \ldots, x_n]$
     i. e.g. word embeddings
- **Model**
  a. A **classification function** that computes $\hat{y}$, the estimated class, via $p(y|x)$
     e.g. logistic regression using **sigmoid** or **softmax** functions
- **Loss**
  - An objective function for learning
    - e.g. **cross-entropy loss**, L_{CE}
- **Optimization**
  - An algorithm for optimizing the objective function
    - e.g. **stochastic gradient descent**

→ **Inference** / Evaluation

Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks
- Discriminative classifier
  - Learn a model that can (given the input) discriminate between two classes (i.e. learn a decision boundary)



The Two Phases of Logistic Regression

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

Test (also called **inference**): 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) has higher probability

Binary Classification in Logistic Regression

Given a series of input/output pairs: $(x^{(i)}, y^{(i)})$

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

- We represent 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’": $w_,$ = -2

Logistic Regression for One Observation x

- Input observation: vector $x = [x_1, x_2, \ldots, x_n]$
- Weights: one per feature: $W = [w_1, w_2, \ldots, w_n]$
  - Sometimes we call the weights $\theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output: a predicted class $\hat{y}^{(i)} \in \{0, 1\}$

- (multinomial logistic regression: $\hat{y}^{(i)} \in \{0,1,2,3,4\}$)

How to do Classification

- For each feature x_i , weight w_i tells us the importance of x_i
  - We also have a bias term b , which is a weight not associated with any feature
  - Parameter vector: $\theta = [\mathbf{w}; b]$
- We use the sum of weighted features and the bias

$$z = \sum_{i=1}^{n} w_i x_i + b$$

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

If this sum is high, we say $\hat{y}=1$; if low, then $\hat{y}=0$

$$z \in \mathbb{R}$$

How to determine a threshold?

Need a probabilistic model to tell us:

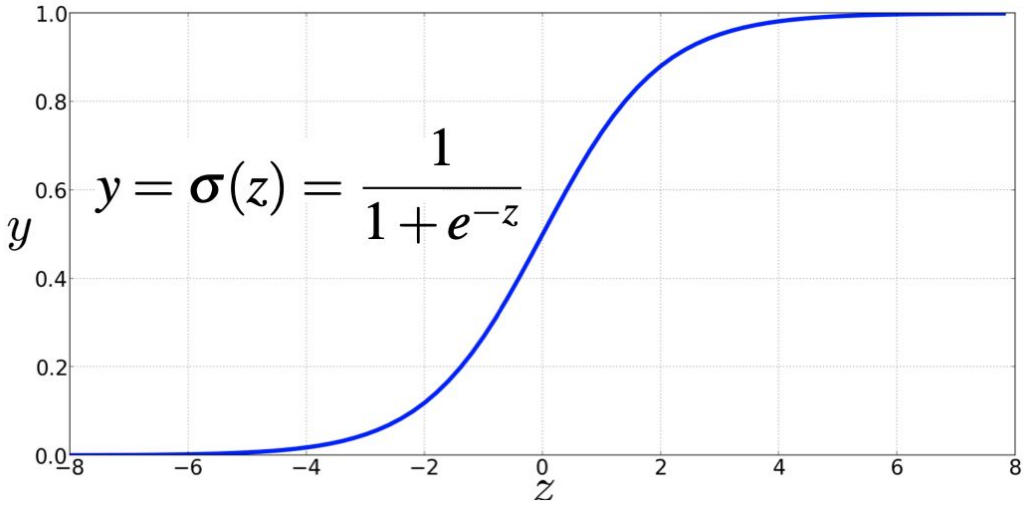
$$P(y = 1 \mid \mathbf{x}; \theta)$$

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

z is a Number (Not a Probability): Map it into the Range 0-1

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

- We'll compute z = *w·x+b*
- And then we'll pass it through the **sigmoid function**: *σ(w·x+b)*
- We'll just treat it as a probability
- **Note:** sigmoid function is differentiable (good candidate for optimization)



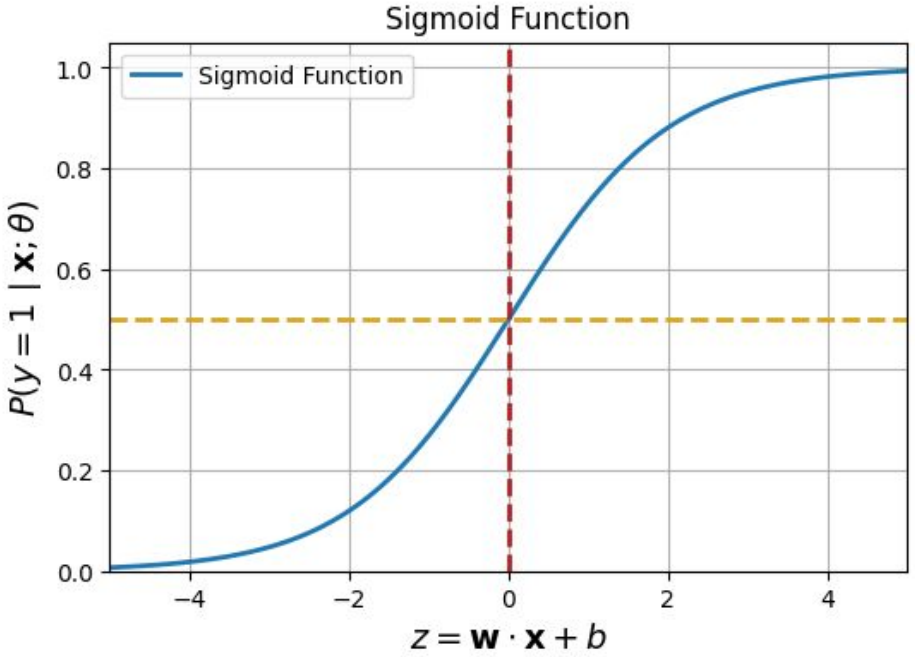
Turning a Probability into a Classifier

0.5 : **decision boundary**

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

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

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

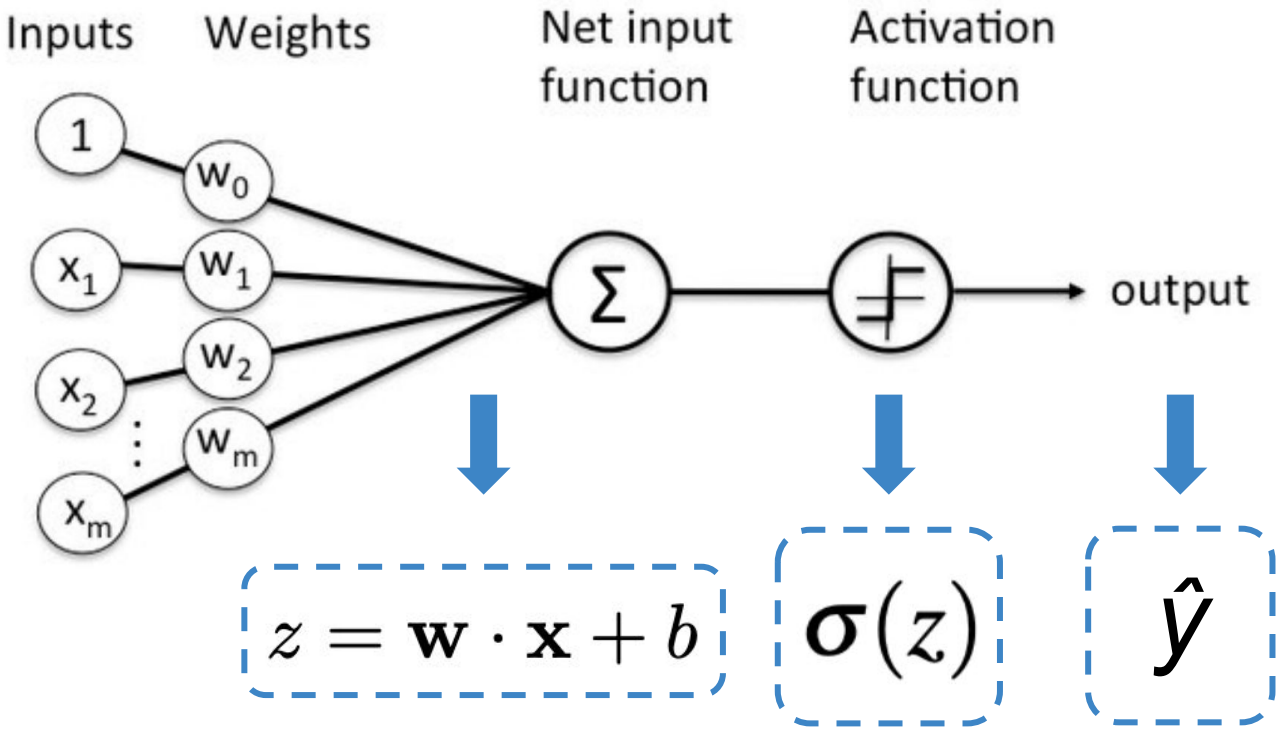


Making Probabilities with Sigmoids

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

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

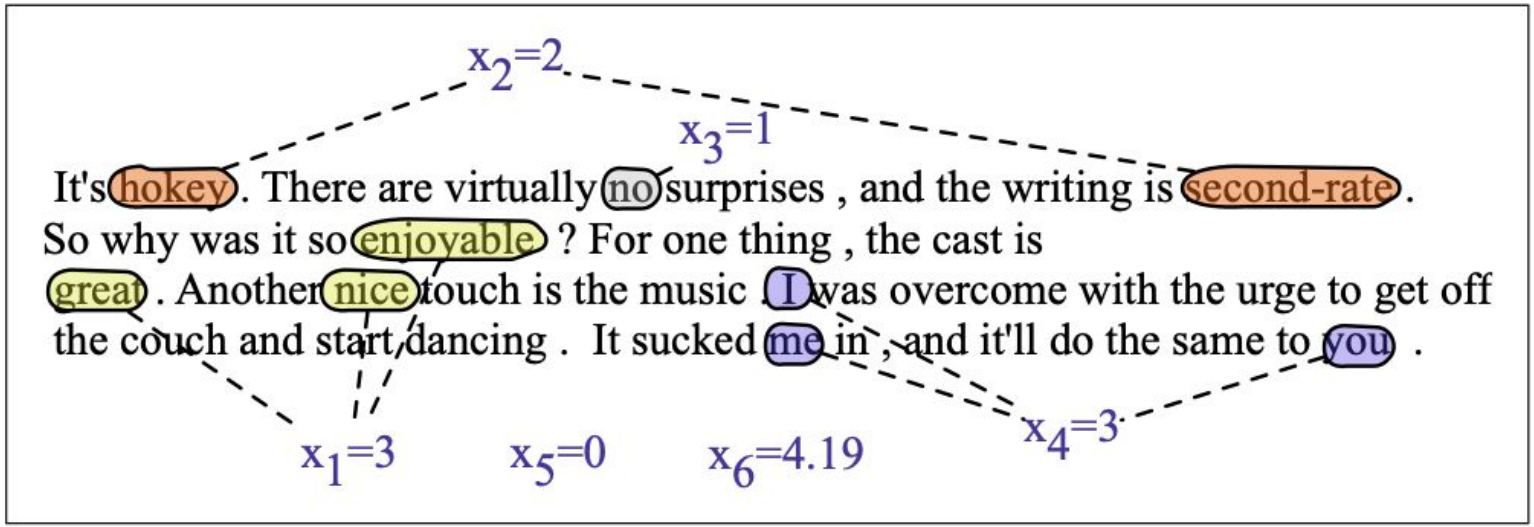
Logistic Regression



1. Compute $w \cdot x + b$
2. Pass it through the sigmoid function: $\sigma(w \cdot x + b)$
3. Treat it as a probability

Logistic Regression: A Text Example on 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



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)	$\ln(66) = 4.19$

Classifying sentiment for input x

Var	Definition	Va . 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 = $[2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$

$b = 0.1$

Classifying sentiment for input x

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

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

But where do the w's and *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, $\hat{y}$
- We want to set **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**
    - e.g. cross-entropy loss
  - We need an **optimization algorithm** to update **w** and *b* to minimize the loss.
    - e.g. stochastic gradient descent

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 output:
  - $y \in \{0,1\}$
- We'll call this difference:
  - $L(\hat{y}, y)$ = how much $\hat{y}$ differs from the true y

Intuition of Negative Log Likelihood Loss (Cross-Entropy Loss)

- A case of conditional maximum likelihood estimation
- We choose the parameters **w**, *b* that maximize the log probability of the true *y* labels in the training data given the observations *x*

Deriving Cross-Entropy Loss for a Single Observation x

Goal: maximize probability of the correct label *p(y|x)*

Since there are only 2 discrete outcomes (0 or 1) we can express the probability *p(y|x)* from our classifier (the thing we want to maximize) as

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

noting:

if *y*=1, this simplifies to $\hat{y}$

if *y*=0, this simplifies to 1- $\hat{y}$

Deriving Cross-Entropy Loss for a Single Observation x

Goal: maximize probability of the correct label *p(y|x)*

Maximize:

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

Take the log of both sides:

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

Deriving Cross-Entropy Loss for a Single Observation x

Goal: maximize probability of the correct label *p(y|x)*

Maximize:

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

Flip the sign to turn it into a loss (something to minimize):

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

Plug in for $\hat{y}$:

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

Let's Revisit Our Sentiment Example

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

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

What's the loss?

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

What if *y=0* ?

Let's Revisit Our Sentiment Example

What if the true value 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?

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

Let's Revisit Our Sentiment Example

The loss when the model was right (if true *y*=1):

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

Is lower than the loss when the model was wrong (if true *y*=0):

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

Stochastic Gradient Descent

Our Goal: Minimize the Loss

Let's make explicit that the loss function is parameterized by weights θ=[**w**;*b*]

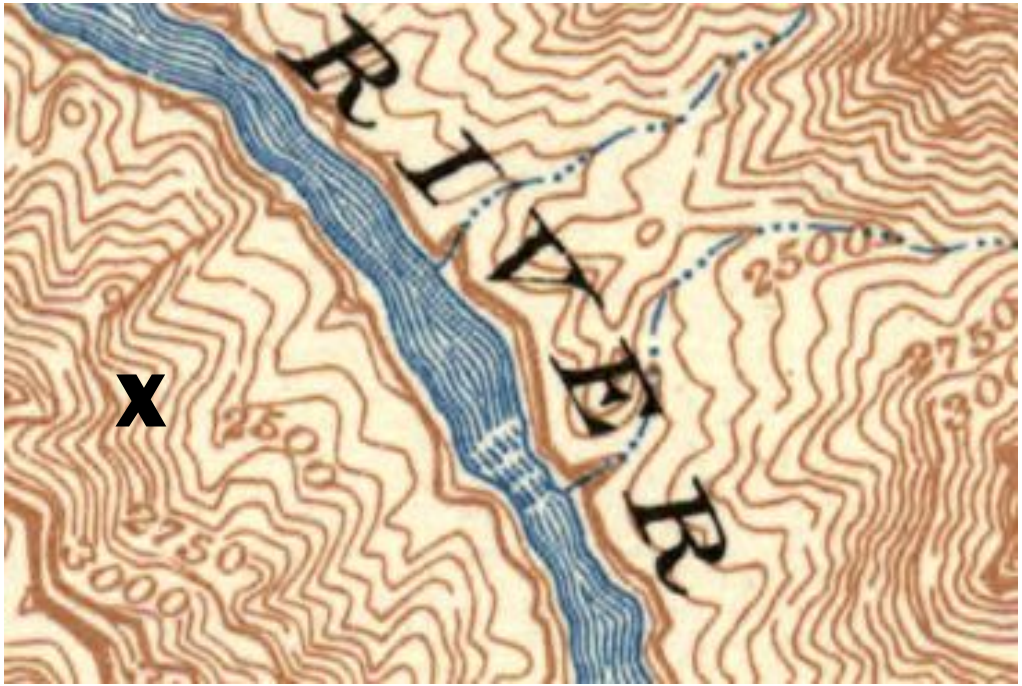
- We'll represent $\hat{y}$ as $f(x;\theta)$ to make the dependence on θ more obvious.

We want the weights that minimize the loss, averaged over all examples

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)};\theta), y^{(i)})$$

Intuition of Gradient Descent

How do I get to the bottom of this river canyon?

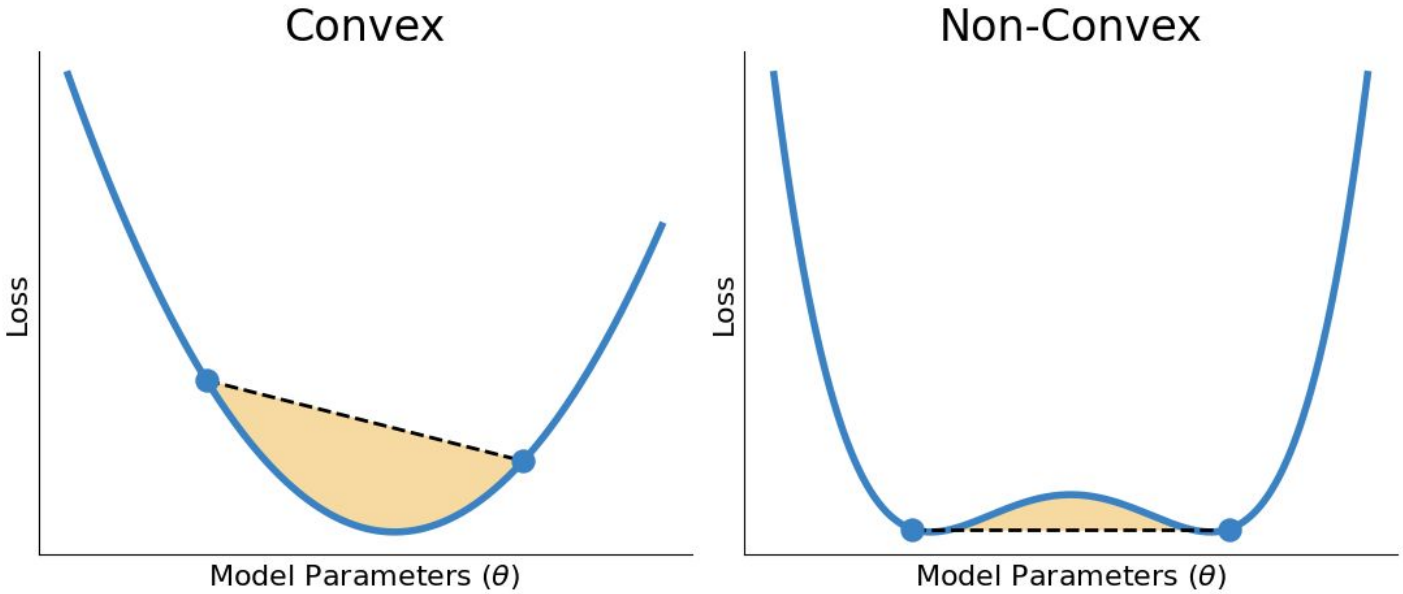


Look around me 360°

Find the direction of the steepest slope down

Go that way

Our Goal: Minimize the Loss



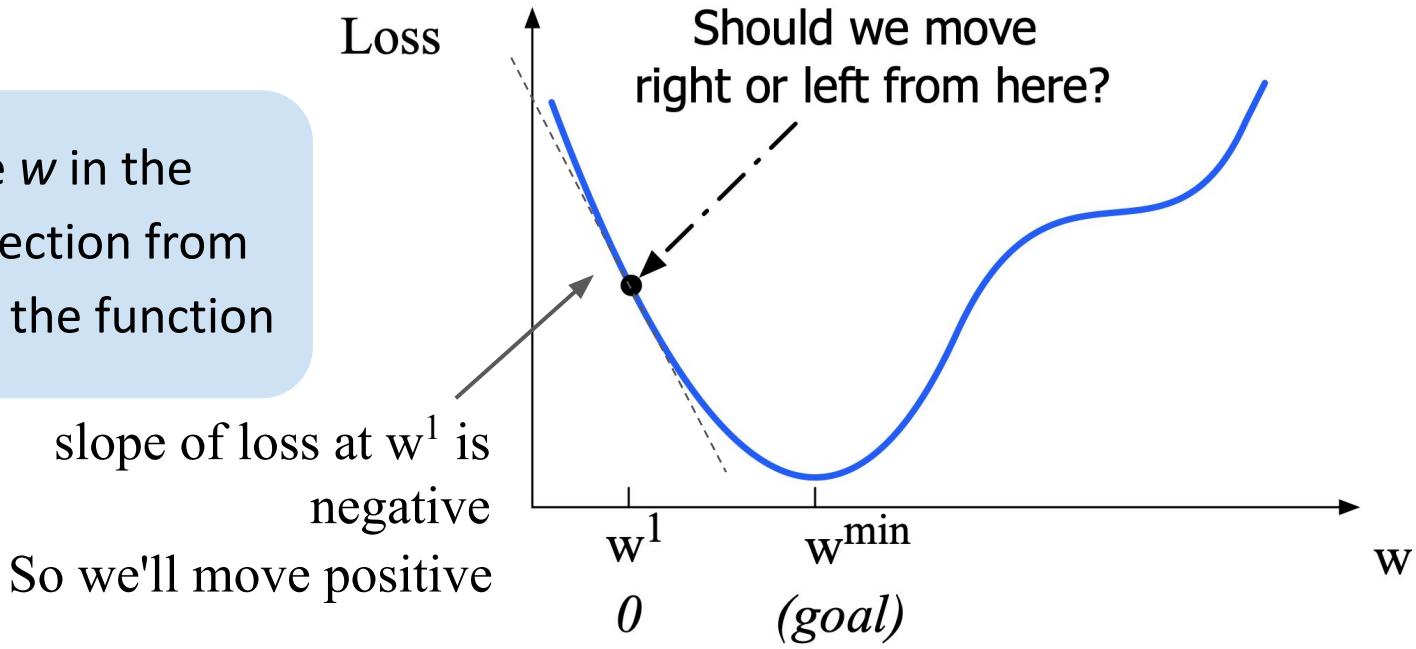
For logistic regression, loss function is **convex**

- A convex function has just one minimum
- Gradient descent starting from any point is guaranteed to find the minimum
- (Loss for neural networks is non-convex)

Visualize: A Single Scalar *w*

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

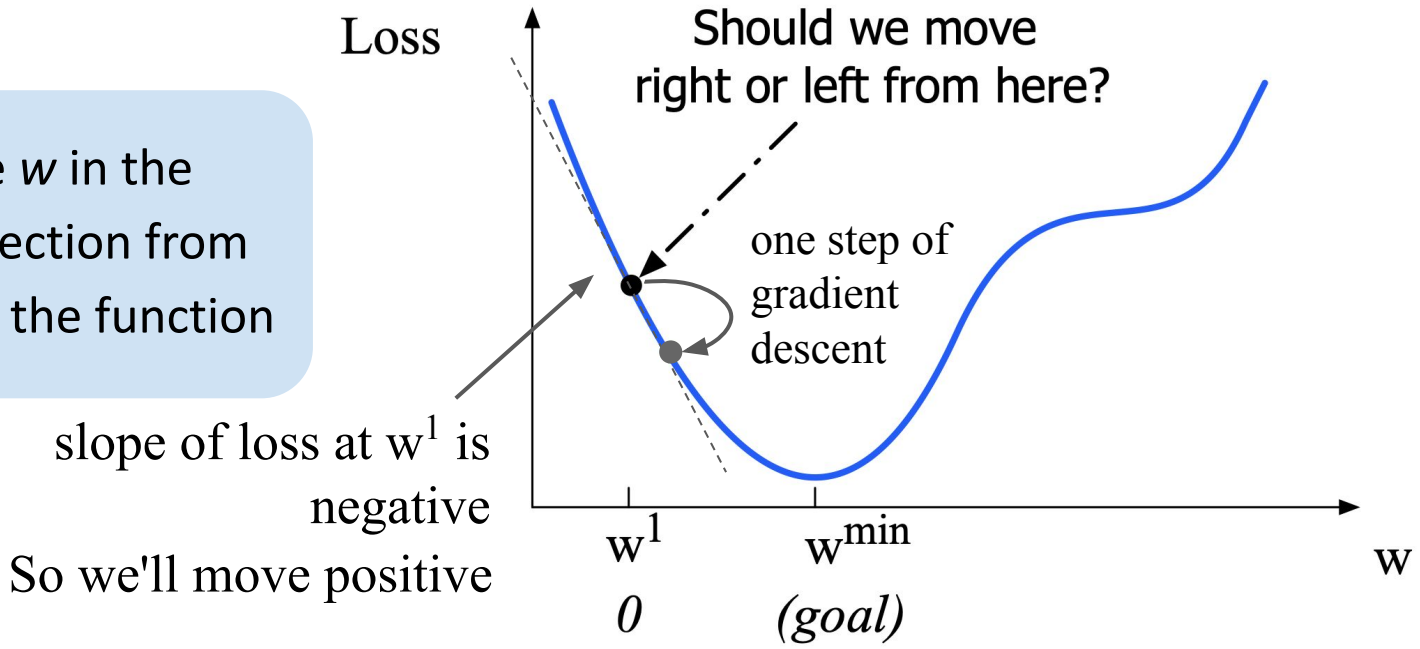
A: Move *w* in the reverse direction from the slope of the function



Visualize: A Single Scalar *w*

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

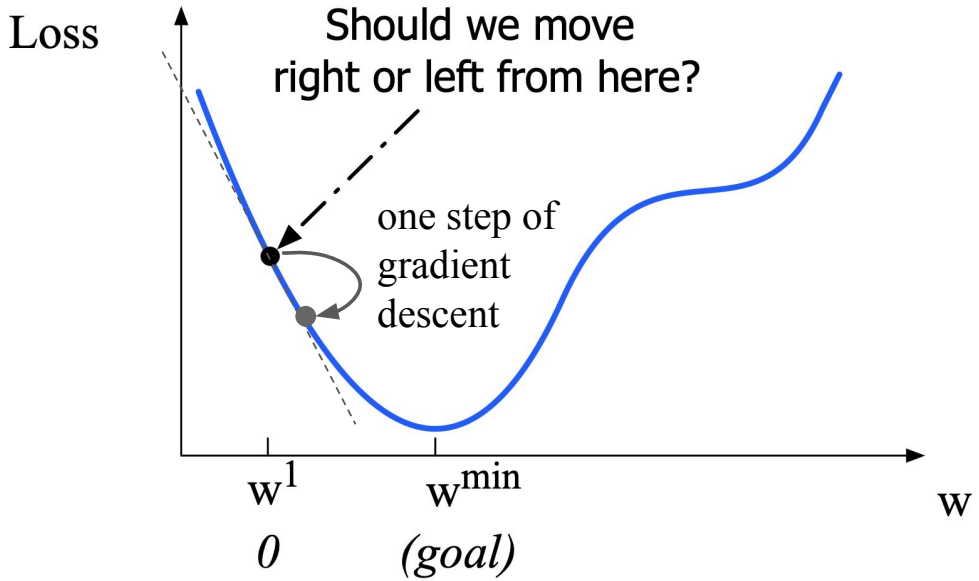
A: Move *w* in the reverse direction from the slope of the function



Gradients

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? Gradient Updates

- Move *w* by the value of the gradient $\frac{\partial}{\partial w} L(f(x;w),y)$, weighted by a learning rate η
- Higher learning rate means we move *w* faster

$$w_{t+1} = w_t - \eta \frac{\partial}{\partial w} L(f(x;w),y)$$

Learning rate η is called a **hyperparameter**

Too high: learner will take big steps and overshoot.
Too low: learner will take too long.

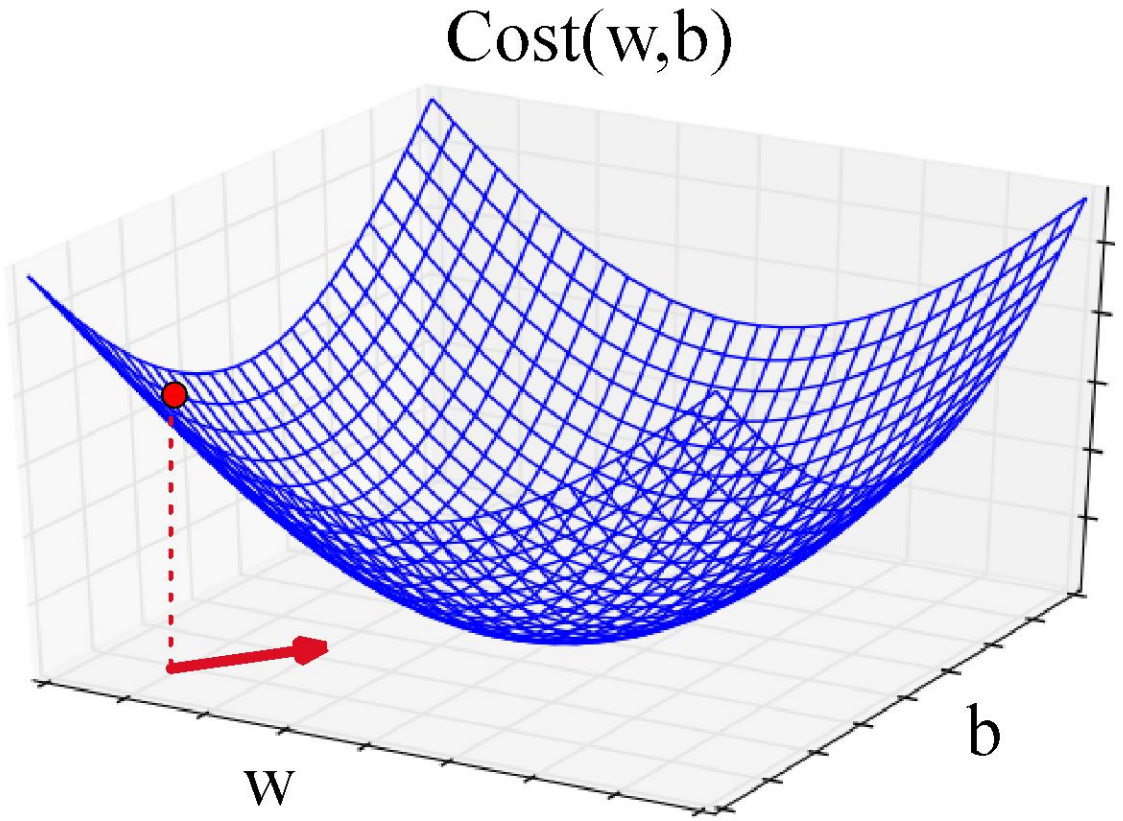
Consider a vector θ of *d* dimensions

The gradient is just such a vector; it expresses it expresses the directional components of the sharpest slope along each of the *d* dimensions.

Consider 2-dimensions: w and *b*

Visualizing the gradient vector at the red point.

It has two dimensions shown in the x-y plane.



Real Gradients

- Are much longer; lots and lots of weights!
- For each dimension w_i the gradient component *i* tells us the slope with respect to that variable.
  - "How much would a small change in w_i influence the total loss function *L*?"
  - We express the slope as a partial derivative $\frac{\partial}{\partial w_i}$ of the loss $\frac{\partial L}{\partial w_i}$
- The gradient is then defined as a vector of these partials.

The Gradient

Again, we'll represent ŷ 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:

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

What are these partial derivatives for logistic regression?

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 derivative of this function (see textbook 5.8 for derivation)

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

Intuitive value: the difference between our estimated $\hat{y}$ and true y, multiplied by the corresponding input value x_j

Working Through an 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
  - $w_1 = w_2 = b = 0$
  - $\eta = 0.1$

Example of Gradient Descent

$w_1 = w_2 = b = 0$;
$x_1 = 3$; $x_2 = 2$

Update step for updating 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:

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

Example of Gradient Descent

$w_1 = w_2 = b = 0$;
$x_1 = 3$; $x_2 = 2$

Update step for updating 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:

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

Example of Gradient Descent

$w_1 = w_2 = b = 0;$
$x_1 = 3; \quad x_2 = 2$

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

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:

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

$\eta = 0.1;$

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

Note: enough negative examples will eventually make w_2 negative.

Stochastic Gradient Descent: Pseudocode

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

$\theta \leftarrow 0$

repeat til done

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

1. Optional (for reporting):
   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

Mini-Batching: Pseudocode

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y, m) **returns** θ

where: L is the loss function
f is a function parameterized by θ
x is the set of training inputs $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$
y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots, y^{(N)}$ and m is the mini-batch size

$\theta \leftarrow 0$ (or randomly initialized)
repeat till done
for each randomly sampled minibatch of size m:
1. for each training tuple $(x^{(i)}, y^{(i)})$ in the minibatch: (in random order)
   i. Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output $\hat{y}^{(i)}$?
   ii. Compute the loss $L_{mini} \leftarrow L_{mini} + L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
2. $g \leftarrow \frac{1}{m} \nabla_\theta L_{mini}$ # How should we move θ to maximize loss?
3. $\theta \leftarrow \theta - \eta g$ # Go the other way instead

return θ

Why is this typically better than SGD in practice?

TODOs:

- **Project Pitch Slide Due: 9/14 (Next Monday)**
- **Homework 1 Released Friday: Due 9/25**

Next Lecture: Evaluating Text Classification; Word Embeddings (I)