# Logistic Regression

**Natalie Parde, Ph.D.**

Department of Computer Science

University of Illinois at Chicago

CS 521: Statistical Natural Language Processing

Spring 2020

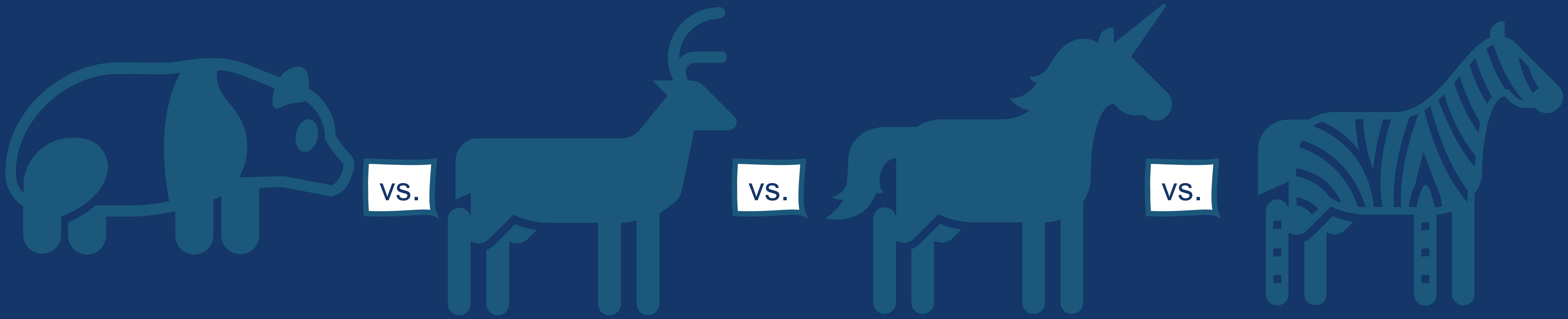
Many slides adapted from Jurafsky and Martin (https://web.stanford.edu/~jurafsky/slp3/).

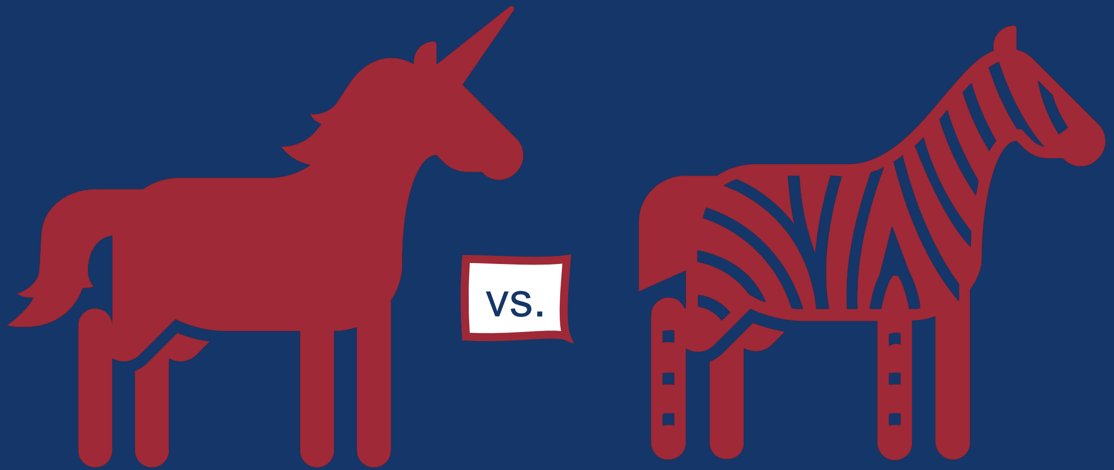
# What is logistic regression?

- Fundamental **supervised machine learning algorithm**
- Used for **text classification**
- Very close relationship with **neural networks!**

# Did someone say “neural networks”? 🤓🤩

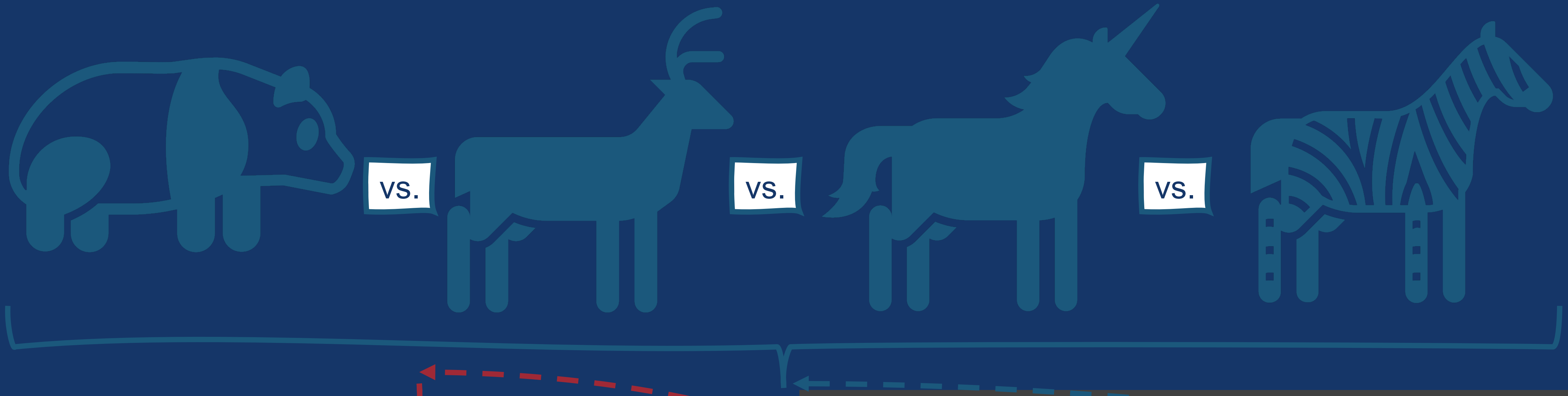
- Coming up in a couple weeks! 😉
- One way to view feedforward neural networks is as a **series of logistic regression classifiers** stacked on top of one another

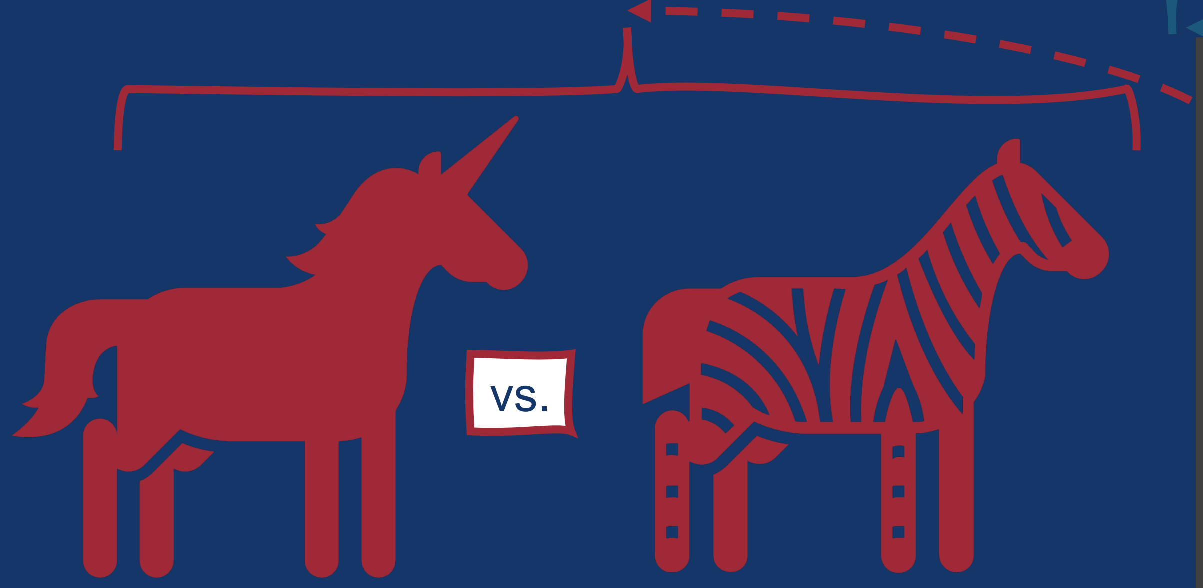



Logistic regression can be used for binary classification or multinomial classification.

- Binary
    - Class A vs. Class B
- Multinomial
    - Class A vs. Class B vs. Class C vs. Class D….




Logistic regression can be used for binary classification or multinomial classification.

- Binary
  - Class A vs. Class B
- Multinomial
  - Class A vs. Class B vs. Class C vs. Class D….

# How does logistic regression differ from naïve Bayes?

## Naïve Bayes

- Generative classifier

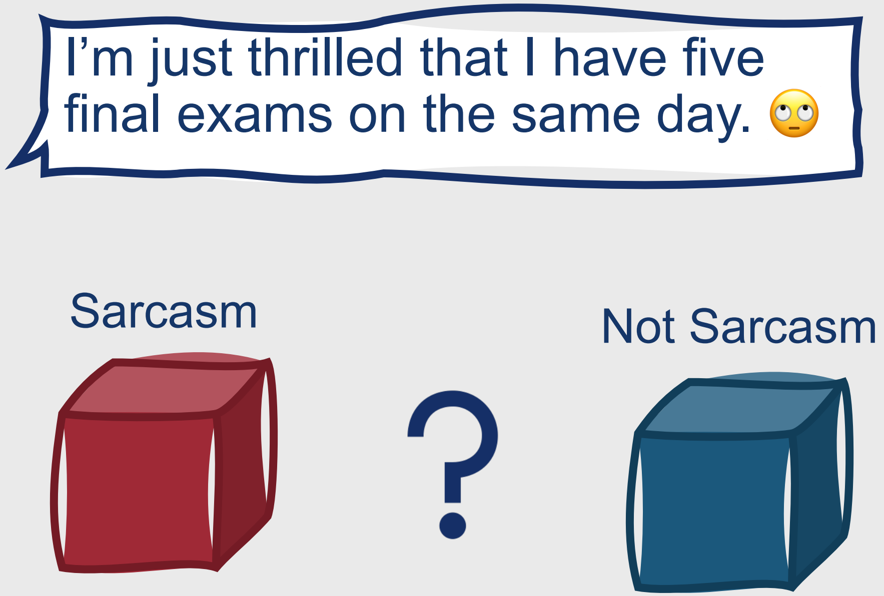
## Logistic Regression

- Discriminative classifier

Not sure what naïve Bayes is? Check out the course slides from CS 421: http://www.natalieparde.com/teaching/cs_421_fall2019/Naive%20Bayes,%20Text%20Classification,%20and%20Evaluation%20Metrics.pdf

# Generative Classifiers

- Goal: Understand what each class looks like
    - Should be able to “generate” an instance from each class
- To classify an instance, determines which class model better fits the instance, and chooses that as the label



# Discriminative Classifiers

- Goal: Learn to distinguish between two classes
  - No need to learn that much about them individually
- To classify an instance, determines whether the distinguishing feature(s) between classes is present

# More formally….

- Recall the definition of naïve Bayes:
  - $\hat{c} = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(d|c)}_{\text{Likelihood}} \underbrace{P(c)}_{\text{Prior}}$

# More formally….

- Recall the definition of naïve Bayes:
  - $\hat{c} = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(d|c)}_{\text{Likelihood}} \underbrace{P(c)}_{\text{Prior}}$

A generative model like naïve Bayes makes use of the **likelihood** term

- **Likelihood:** Expresses how to generate an instance *if it knows it is of class c*

# More formally….

- Recall the definition of naïve Bayes:
  - $\hat{c} = \underset{c\in C}{\operatorname{argmax}} \cancel{\underbrace{P(d|c)}_{\text{Likelihood}}\underbrace{P(c)}_{\text{Prior}}}$

$$\hat{c} = \underset{c\in C}{\operatorname{argmax}}\, P(c|d)$$

A discriminative model instead tries to compute P(*c*|*d*) directly!

## However, naïve Bayes and logistic regression also have some similarities.

Both are **probabilistic classifiers**

Both perform **supervised machine learning**

- **Supervised machine learning:** Machine learning with labeled training and test data
  - Generally formalized as *x*s (instances) and *y*s (labels), where an individual instance is an $x^{(i)}$, $y^{(i)}$ pair

# Which is better …naïve Bayes or logistic regression?

- Depends on the task and the dataset
- For larger datasets, logistic regression is usually better
- For smaller datasets, naïve Bayes is sometimes better
- Naïve Bayes is easy to implement and faster to train
- **Best to experiment with multiple classification models to determine which is best for your needs**

# In general, supervised machine learning systems for text classification have four main components.

- **Feature representation** of the input
  - Typically, a **vector** of features $[x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)}]$ for a given instance $x^{(j)}$
- **Classification function** that computes the estimated class, $\hat{y}$
  - Sigmoid
  - Softmax
  - Etc.
- **Objective function** or **loss function** that computes error values on training instances
  - Cross-entropy loss function
- **Optimization function** that seeks to minimize the loss function
  - Stochastic gradient descent

# Likewise, supervised machine learning systems generally have two phases.

- **Training**
  - For logistic regression, you train **weights** *w* and a **bias** *b* using **stochastic gradient descent** and **cross-entropy loss**
- **Test**
  - Using your trained model, you compute P($y$|$x$) and return the highest probability label

# Classifier Building Blocks: The Sigmoid

- Goal of binary logistic regression:
  - Train a classifier that can decide whether a new input observation belongs to class *a* or class *b*
- To do this, the classifier learns a **vector of weights** (one associated with each input feature) and a **bias term**
- A given **weight indicates how important its corresponding feature is** to the overall classification decision
  - Can be positive or negative
- The **bias term is a real number** that is added to the weighted inputs

# Classifier Building Blocks: The Sigmoid

- To make a classification decision, the classifier:
  - Multiplies each feature for an input instance *x* by its corresponding weight (learned from the training data)
  - Sums the weighted features
  - Adds the bias term *b*
- This results in a weighted sum of evidence for the class:
  - $z = b + \sum_i w_i x_i$

Bias term

Weight for feature *i*

Feature *i* for instance *x*

# Vector Notation

- Letting $w$ be the weight vector and $x$ be the input feature vector, we can also represent the weighted sum $z$ using vector notation:
  - $z = \boldsymbol{w} \cdot \boldsymbol{x} + b$

Vector of all weights

Bias term

Vector of all features for $x$

# However, this still computes a linear function of *x*!

- What we really want is a **probability** ranging from 0 to 1
- To do this, we pass *z* through the sigmoid function, $\sigma(z)$
  - Also called the **logistic function**, hence the name **logistic regression**

# Sigmoid Function

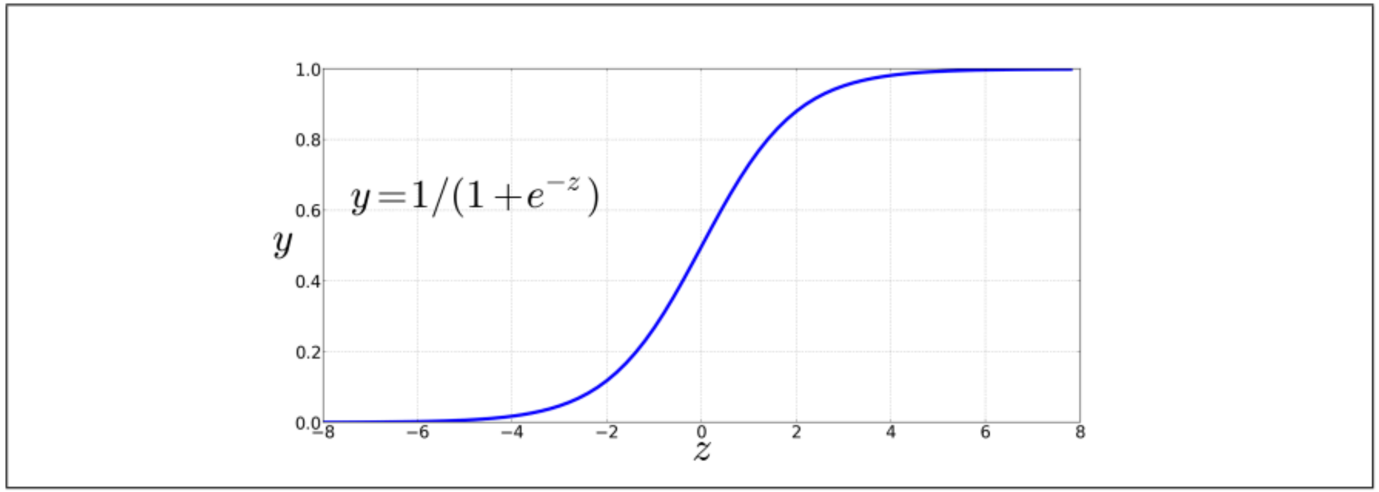



**Figure 5.1** The sigmoid function $y=\frac{1}{1+e^{-z}}$ takes a real value and maps it to the range [0, 1]. It is nearly linear around 0 but outlier values get squashed toward 0 or 1.

- Sigmoid Function:
  - $\sigma(x) = \frac{1}{1+e^{-x}}$
- Given its name because when plotted, it looks like an *s*
- Results in a value *y* ranging from 0 to 1
  - $y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-w \cdot x+b}}$

# There are many useful properties of the sigmoid function!

- Maps a real-valued number to a 0 to 1 range
  - Just what we need for a probability….
- Squashes outlier values towards 0 or 1
- Differentiable
  - Necessary for learning….

# How do we convert the sigmoid output to a real probability?

- Just make all the cases sum to 1
  - $P(y = 1) = \sigma(z)$
  - $P(y = 0) = 1 - \sigma(z)$

# How do we make a classification decision?

- Choose a **decision boundary**
  - For binary classification, often 0.5
- For a test instance *x*, assign a label *c* if $P(y = c|x)$ is greater than the decision boundary
  - If performing binary classification, assign the other label if $P(y = c|x)$ is lower than or equal to the decision boundary

# Example: Sigmoid Classification

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

# Example: Sigmoid Classification

I’m just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature |
|---|
| Contains 🙄 |
| Contains 😊 |
| Contains “I’m” |

# Example: Sigmoid Classification

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature | Weight |
|---|---|
| Contains 🙄 | 2.5 |
| Contains 😊 | -3.0 |
| Contains "I'm" | 0.5 |

# Example: Sigmoid Classification

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature | Weight |
|---|---|
| Contains 🙄 | 2.5 |
| Contains 😊 | -3.0 |
| Contains "I'm" | 0.5 |

Positively associated with sarcasm

Negatively associated with sarcasm

# Example: Sigmoid Classification

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 2.5 | 1 |
| Contains 😊 | -3.0 | 0 |
| Contains "I'm" | 0.5 | 1 |

# Example: Sigmoid Classification

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 2.5 | 1 |
| Contains 😊 | -3.0 | 0 |
| Contains "I'm" | 0.5 | 1 |

Bias = 0.1

# Example: Sigmoid Classification

I’m just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 2.5 | 1 |
| Contains 😊 | -3.0 | 0 |
| Contains “I’m” | 0.5 | 1 |

Bias = 0.1

$$z = b + \sum_i w_i x_i$$

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

# Example: Sigmoid Classification

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 2.5 | 1 |
| Contains 😊 | -3.0 | 0 |
| Contains "I'm" | 0.5 | 1 |

Bias = 0.1

$$z = b + \sum_i w_i x_i$$

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

$$P(\text{sarcasm}|x) = \sigma\big(0.1 + (2.5 * 1 + (-3.0) * 0 + 0.5 * 1)\big) = \sigma(0.1 + 3.0) = \sigma(3.1) = \frac{1}{1 + e^{-3.1}} = 0.96$$

# Example: Sigmoid Classification

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 2.5 | 1 |
| Contains 😊 | -3.0 | 0 |
| Contains "I'm" | 0.5 | 1 |

Bias = 0.1

$$z = b + \sum_i w_i x_i$$

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

$$P(\text{sarcasm}|x) = \sigma\big(0.1 + (2.5 * 1 + (-3.0) * 0 + 0.5 * 1)\big) = \sigma(0.1 + 3.0) = \sigma(3.1) = \frac{1}{1+e^{-3.1}} = 0.96$$

$$P(\text{not sarcasm}|x) = 1 - \sigma\big(0.1 + (2.5 * 1 + (-3.0) * 0 + 0.5 * 1)\big) = 1 - \sigma(0.1 + 3.0) = 1 - \sigma(3.1) = 1 - \frac{1}{1+e^{-3.1}} = 1 - 0.96 = 0.04$$

# Example: Sigmoid Classification

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic or not sarcastic?

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 2.5 | 1 |
| Contains 😊 | -3.0 | 0 |
| Contains "I'm" | 0.5 | 1 |

Bias = 0.1

$$z = b + \sum_i w_i x_i$$

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

$$P(\text{sarcasm}|x) = \sigma\big(0.1 + (2.5 * 1 + (-3.0) * 0 + 0.5 * 1)\big) = \sigma(0.1 + 3.0) = \sigma(3.1) = \frac{1}{1+e^{-3.1}} = 0.96$$

$$P(\text{not sarcasm}|x) = 1 - \sigma\big(0.1 + (2.5 * 1 + (-3.0) * 0 + 0.5 * 1)\big) = 1 - \sigma(0.1 + 3.0) = 1 - \sigma(3.1) = 1 - \frac{1}{1+e^{-3.1}} = 1 - 0.96 = 0.04$$

# A little bit about features….

- Anything can be a feature!
  - Specific words or n-grams
  - Information from external lexicons
  - Grammatical elements
  - Part-of-speech tags
- In neural classification models, the feature vector often includes word embeddings
  - More about these next week!

# Learning in Logistic Regression

- How are the parameters of a logistic regression model, $w$ and $b$, learned?
  - **Loss function**
  - **Optimization function**
- Goal: Learn parameters that make $\hat{y}$ for each training observation as close as possible to the true $y$

# Loss Function

- We need to determine the distance between the predicted and true output value
  - How much does $\hat{y}$ differ from $y$?
- We do this using a **conditional maximum likelihood estimation**
  - Select *w* and *b* such that they maximize the log probability of the true *y* values in the training data, given their observations *x*
- This results in a **negative log likelihood loss**
  - More commonly referred to as **cross-entropy loss**

# Cross-Entropy Loss

- Most common loss function for many classification tasks
- Measures the distance between the probability distributions of predicted and actual values
  - $loss(y_i, \widehat{y_i}) = -\sum_{c=1}^{|C|} p_{i,c} \log \widehat{p_{i,c}}$
    - *C* is the set of all possible classes
    - $p_{i,c}$ is the actual probability that instance *i* should be labeled with class *c*
    - $\widehat{p_{i,c}}$ is the predicted probability that instance *i* should be labeled with class *c*
- Ranges from 0 (best) to 1 (worst)
- Observations with a big distance between the predicted and actual values have much higher cross-entropy loss than observations with only a small distance between the two values

# Example: Cross-Entropy Loss

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic

Not Sarcastic

# Example: Cross-Entropy Loss

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic

Not Sarcastic

| Instance | Predicted Probability: Sarcastic | Predicted Probability: Not Sarcastic | Actual Probability: Sarcastic | Actual Probability: Not Sarcastic |
|---|---|---|---|---|
| I'm just thrilled that I have five final exams on the same day. 🙄 | | | 1 | 0 |

# Example: Cross-Entropy Loss

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic

Not Sarcastic

| Instance | Predicted Probability: Sarcastic | Predicted Probability: Not Sarcastic | Actual Probability: Sarcastic | Actual Probability: Not Sarcastic |
|---|---|---|---|---|
| I'm just thrilled that I have five final exams on the same day. 🙄 | 0.7 | 0.3 | 1 | 0 |

# Example: Cross-Entropy Loss

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic

Not Sarcastic

| Instance | Predicted Probability: Sarcastic | Predicted Probability: Not Sarcastic | Actual Probability: Sarcastic | Actual Probability: Not Sarcastic |
|---|---|---|---|---|
| I'm just thrilled that I have five final exams on the same day. 🙄 | 0.7 | 0.3 | 1 | 0 |

$$loss(y_i, y_i') = -\sum_{c=1}^{|C|} p_{i,c} \log \widehat{p_{i,c}} = -p_{i,sarcastic} \log \widehat{p_{i,sarcastic}} - p_{i,not\ sarcastic} \log \widehat{p_{i,not\ sarcastic}}$$

# Example: Cross-Entropy Loss

I’m just thrilled that I have five final exams on the same day. 🙄

Sarcastic

Not Sarcastic

| Instance | Predicted Probability: Sarcastic | Predicted Probability: Not Sarcastic | Actual Probability: Sarcastic | Actual Probability: Not Sarcastic |
|---|---|---|---|---|
| I’m just thrilled that I have five final exams on the same day. 🙄 | 0.7 | 0.3 | 1 | 0 |

$$loss(y_i, y_i') = -\sum_{c=1}^{|C|} p_{i,c} \log \widehat{p_{i,c}} = -p_{i,sarcastic} \log \widehat{p_{i,sarcastic}} - p_{i,not\ sarcastic} \log \widehat{p_{i,not\ sarcastic}}$$

$$loss(y_i, y_i') = -1 * \log 0.7 - 0 * \log 0.3$$

# Example: Cross-Entropy Loss

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic

Not Sarcastic

| Instance | Predicted Probability: Sarcastic | Predicted Probability: Not Sarcastic | Actual Probability: Sarcastic | Actual Probability: Not Sarcastic |
|---|---|---|---|---|
| I'm just thrilled that I have five final exams on the same day. 🙄 | 0.7 | 0.3 | 1 | 0 |

$$loss(y_i, y_i') = -\sum_{c=1}^{|C|} p_{i,c} \log \widehat{p_{i,c}} = -p_{i,sarcastic} \log \widehat{p_{i,sarcastic}} - p_{i,not\ sarcastic} \log \widehat{p_{i,not\ sarcastic}}$$

$$loss(y_i, y_i') = -1 * \log 0.7 - 0 * \log 0.3 = -\log 0.7 = 0.15$$

# Example: Cross-Entropy Loss

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic

Not Sarcastic

| Instance | Predicted Probability: Sarcastic | Predicted Probability: Not Sarcastic | Actual Probability: Sarcastic | Actual Probability: Not Sarcastic |
|---|---|---|---|---|
| I'm just thrilled that I have five final exams on the same day. 🙄 | | | 1 | 0 |

What if our predicted values were switched?

# Example: Cross-Entropy Loss

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic

Not Sarcastic

| Instance | Predicted Probability: Sarcastic | Predicted Probability: Not Sarcastic | Actual Probability: Sarcastic | Actual Probability: Not Sarcastic |
|---|---|---|---|---|
| I'm just thrilled that I have five final exams on the same day. 🙄 | 0.3 | 0.7 | 1 | 0 |

$$loss(y_i, y_i') = -\sum_{c=1}^{|C|} p_{i,c} \log \widehat{p_{i,c}} = -p_{i,sarcastic} \log \widehat{p_{i,sarcastic}} - p_{i,not\ sarcastic} \log \widehat{p_{i,not\ sarcastic}}$$

$$loss(y_i, y_i') = -1 * \log 0.3 - 0 * \log 0.7 = -\log 0.3 = 0.52$$

Greater loss value!

# Why does minimizing the negative log probability work?

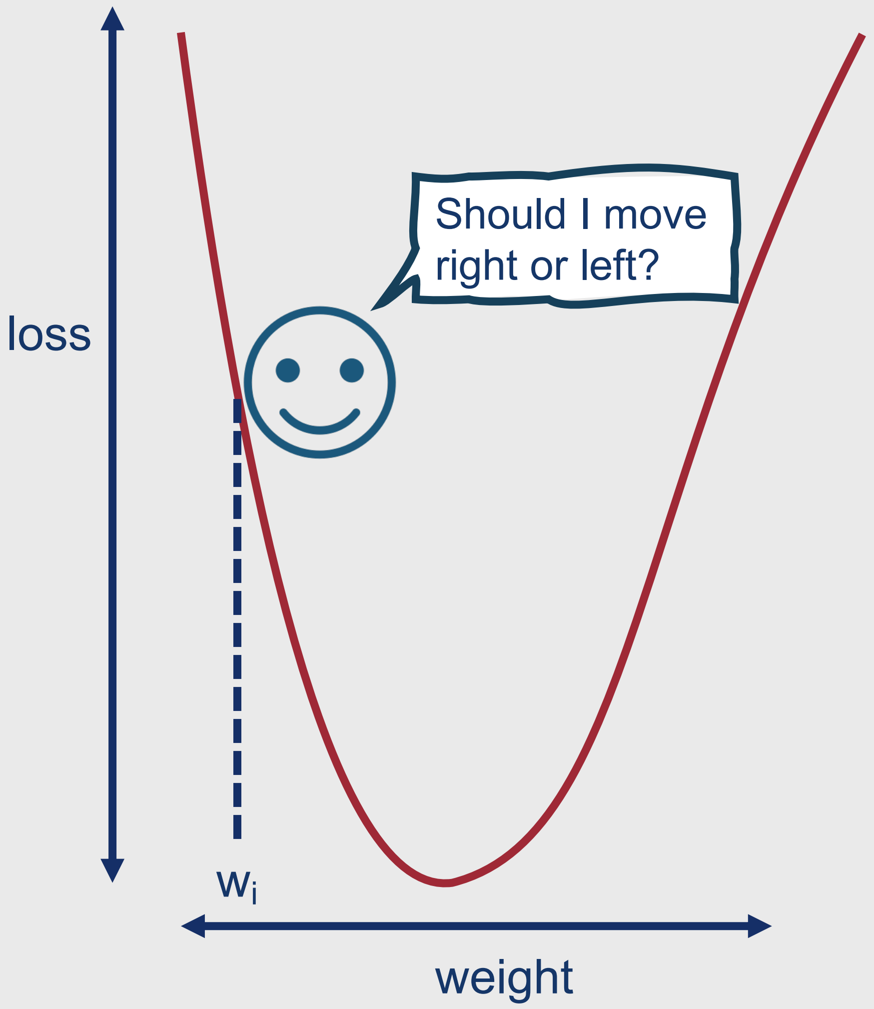
- Perfect binary classifier:
  - Assign probability of 1.0 to the correct class
  - Assign probability of 0.0 to the incorrect class
- Thus, higher $\hat{y}$ is better
- Correspondingly, negative log of 1.0 $\rightarrow$ no loss ($-\log 1.0 = 0$); negative log of 0.0 $\rightarrow$ infinite loss ($-\log 0.0 = \infty$)

# Finding Optimal Weights

- Goal: Minimize the loss function defined for the model
  - $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(y^{(i)}, x^{(i)}; \theta)$
- For logistic regression, $\theta = w, b$
- One way to do this is by using **gradient descent**

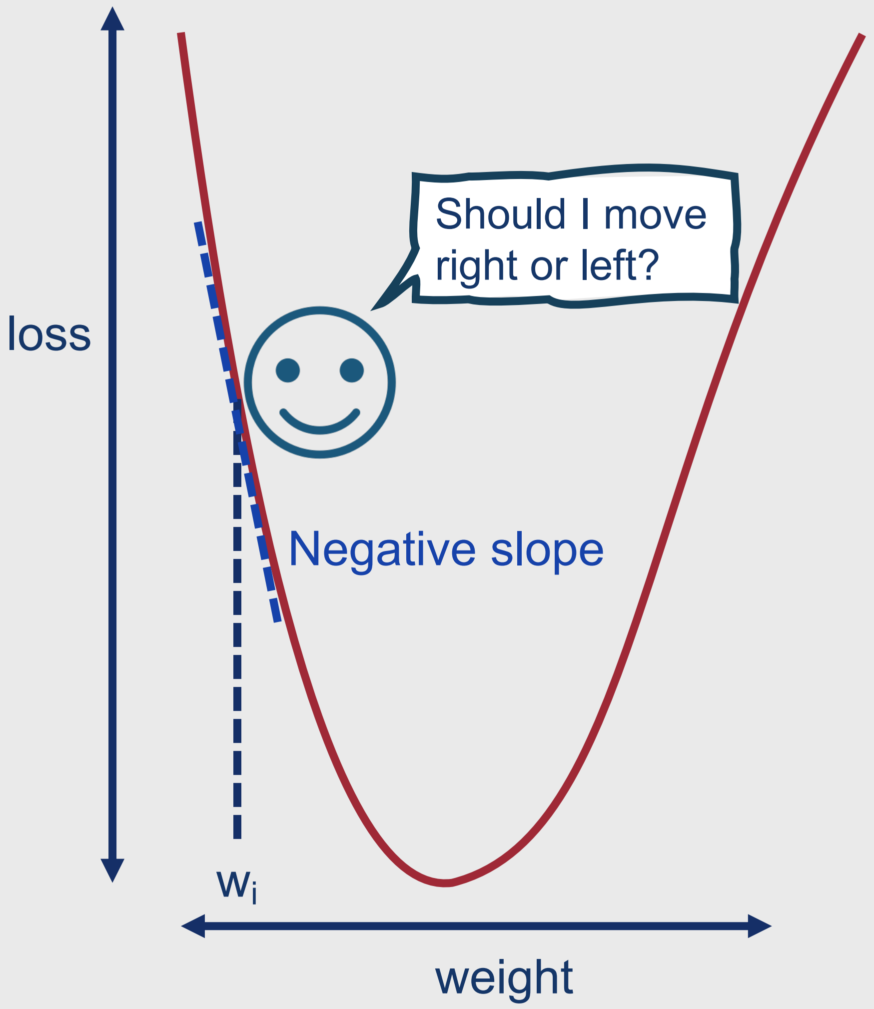
# Gradient Descent

- Finds the minimum of a function by:
  - Figuring out the direction (in the space of $\theta$) the function's slope
  - Moving in the opposite direction
- For logistic regression, loss functions are **convex**
  - Only one minimum
  - Gradient descent starting at any point is guaranteed to find it



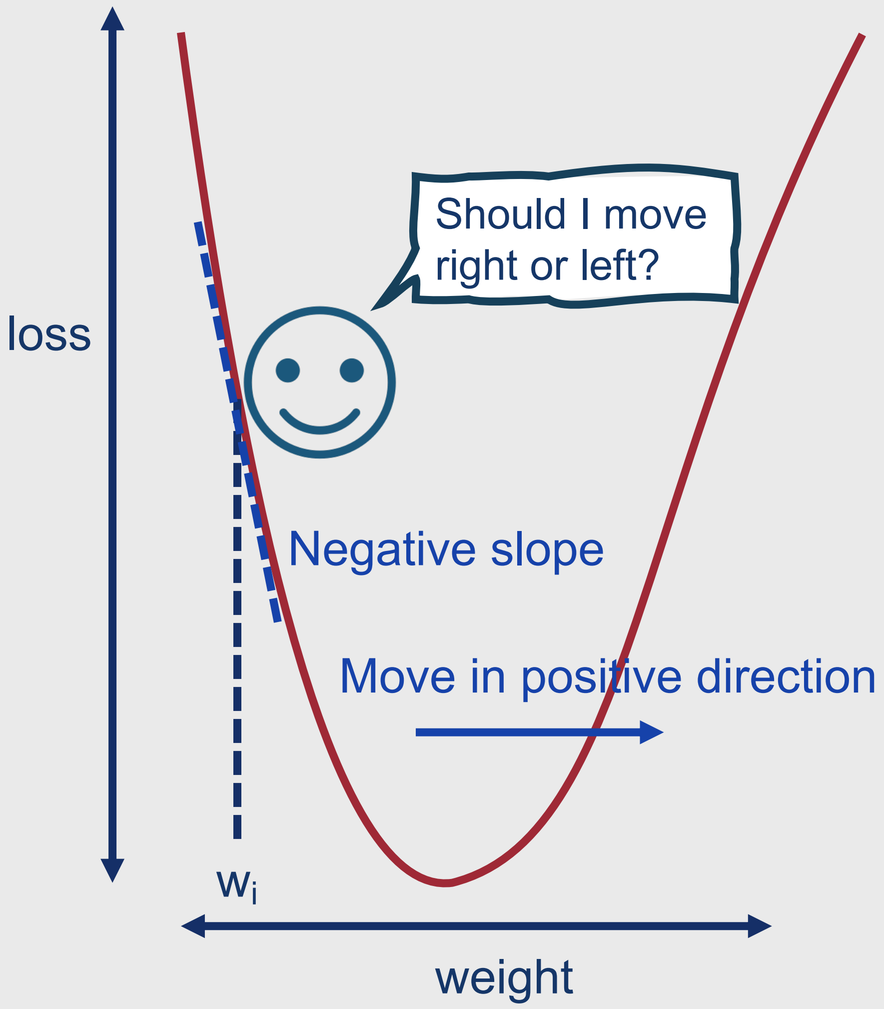
# Gradient Descent

- Finds the minimum of a function by:
  - Figuring out the direction (in the space of $\theta$) the function's slope
  - Moving in the opposite direction
- For logistic regression, loss functions are **convex**
  - Only one minimum
  - Gradient descent starting at any point is guaranteed to find it



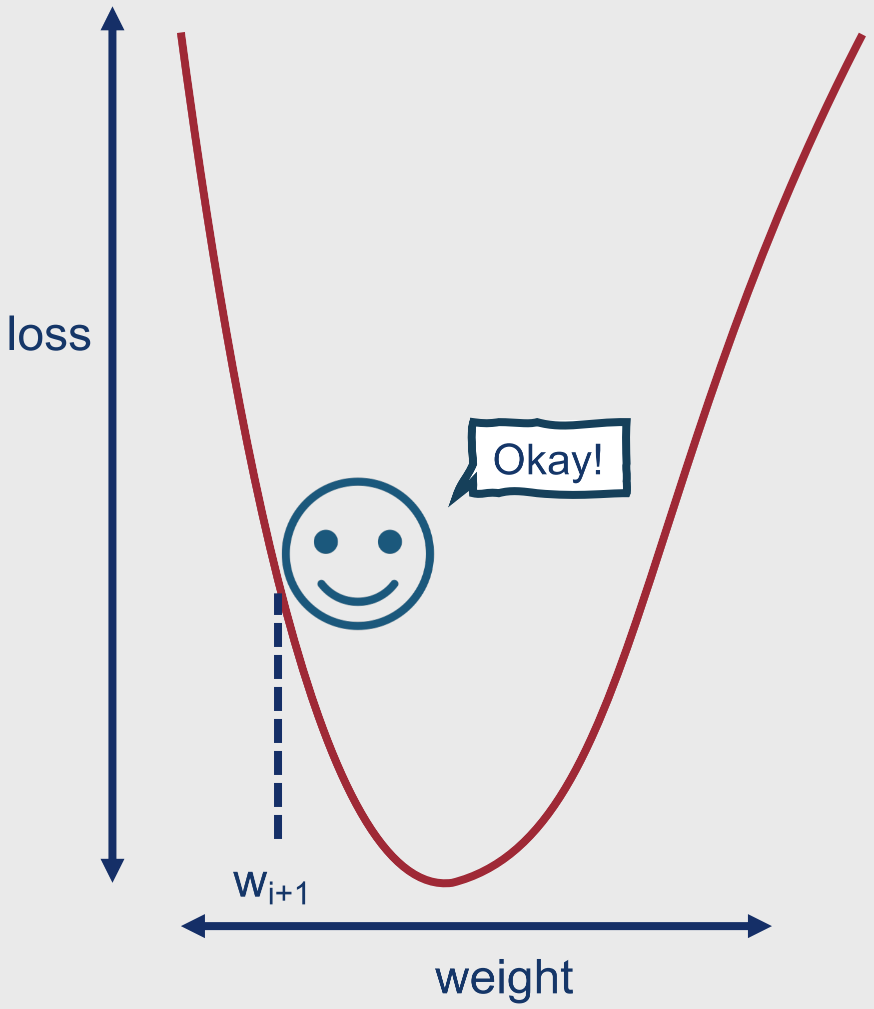
# Gradient Descent

- Finds the minimum of a function by:
  - Figuring out the direction (in the space of $\theta$) the function's slope
  - Moving in the opposite direction
- For logistic regression, loss functions are **convex**
  - Only one minimum
  - Gradient descent starting at any point is guaranteed to find it



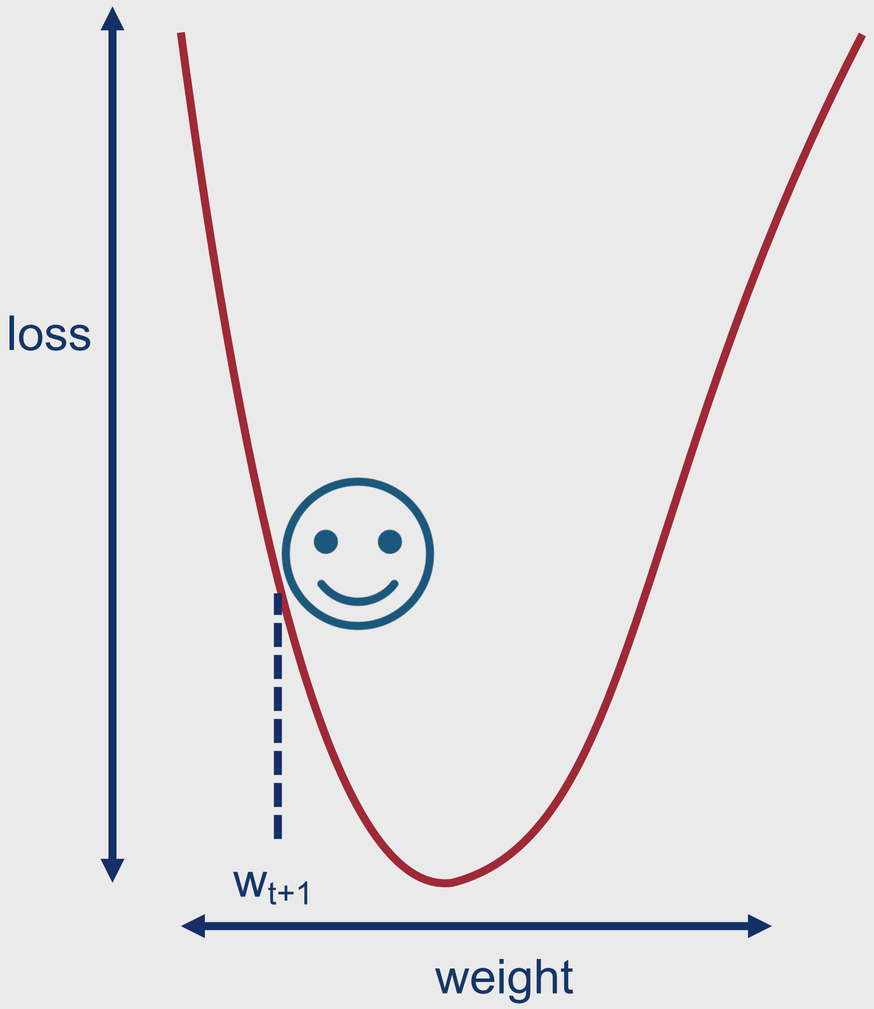
# Gradient Descent

- Finds the minimum of a function by:
  - Figuring out the direction (in the space of $\theta$) the function's slope
  - Moving in the opposite direction
- For logistic regression, loss functions are **convex**
  - Only one minimum
  - Gradient descent starting at any point is guaranteed to find it



# Gradient Descent

- How much do we move?
  - Value of the slope
    - $\frac{d}{dw} f(x; w)$
  - Weighted by a learning rate $\eta$
- Faster learning rate → move *w* more on each step
- So, the change to a weight at time t is actually:
  - $w^{t+1} = w^{t} - \eta \frac{d}{dw} f(x; w)$



# Remember, in actual logistic regression, there are weights for each feature.

- The gradient is then a vector of the slopes of each dimension:
  - $\nabla_\theta L(f(x;\theta),y) = \begin{bmatrix} \frac{d}{dw_1} L(f(x;\theta),y) \\ \cdots \\ \frac{d}{dw_n} L(f(x;\theta),y) \end{bmatrix}$
- This in turn means that the final equation for updating $\theta$ is:
  - $\theta_{t+1} = \theta_t - \eta \nabla L(f(x;\theta),y)$

# The Gradient for Logistic Regression

- Recall our cross-entropy loss function:
  - $loss(y_i, \hat{y}_i) = -\sum_{c=1}^{|C|} y \log \hat{y} = -\sum_{c=1}^{|C|} y \log \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
- The derivative for this function is:
  - $\frac{dL_{CE}(w,b)}{dw_j} = [\sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b) - y]x_j$

Difference between true and estimated *y*

Corresponding input observation

# Stochastic Gradient Descent Algorithm

$\theta \leftarrow 0$ # initialize weights to 0

repeat until convergence:

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

$g \leftarrow \nabla_\theta L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right)$ # change to $\theta$ to maximize loss

$\theta \leftarrow \theta - \eta g$ # go the other way instead

return $\theta$

# Example: Gradient Descent (Single Step)

I'm just thrilled that I have five final exams on the same day. 🙄

Sarcastic

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 0 | 1 |
| Contains 😊 | 0 | 0 |
| Contains "I'm" | 0 | 1 |

# Example: Gradient Descent (Single Step)

I'm just thrilled that I have five final exams on the same day. 🙄

← Sarcastic

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 0 | 1 |
| Contains 😊 | 0 | 0 |
| Contains "I'm" | 0 | 1 |

Bias ($b$) = 0
Learning rate ($\eta$) = 0.1

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right)$$

# Example: Gradient Descent (Single Step)

I'm just thrilled that I have five final exams on the same day. 🙄 ← Sarcastic

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 0 | 1 |
| Contains 😊 | 0 | 0 |
| Contains "I'm" | 0 | 1 |

Bias ($b$) = 0
Learning rate ($\eta$) = 0.1

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$$

$$\nabla_\theta L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right) = \begin{bmatrix} \frac{dL_{CE}(w,b)}{dw_1} \\ \frac{dL_{CE}(w,b)}{dw_2} \\ \frac{dL_{CE}(w,b)}{dw_3} \\ \frac{dL_{CE}(w,b)}{db} \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y)x_1 \\ (\sigma(w \cdot x + b) - y)x_2 \\ (\sigma(w \cdot x + b) - y)x_3 \\ \sigma(w \cdot x + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0) - 1)x_1 \\ (\sigma(0) - 1)x_2 \\ (\sigma(0) - 1)x_3 \\ \sigma(0) - 1 \end{bmatrix} = \begin{bmatrix} (0.5 - 1)x_1 \\ (0.5 - 1)x_2 \\ (0.5 - 1)x_3 \\ (0.5 - 1) \end{bmatrix} = \begin{bmatrix} -0.5 * 1 \\ -0.5 * 0 \\ -0.5 * 1 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0 \\ -0.5 \\ -0.5 \end{bmatrix}$$

# Example: Gradient Descent (Single Step)

> I'm just thrilled that I have five final exams on the same day. 🙄

← Sarcastic

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 0 | 1 |
| Contains 😊 | 0 | 0 |
| Contains "I'm" | 0 | 1 |

Bias ($b$) = 0
Learning rate ($\eta$) = 0.1

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right)$$

$$\nabla_\theta L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right) = \begin{bmatrix} -0.5 \\ 0 \\ -0.5 \\ -0.5 \end{bmatrix}$$

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta L\left(f\left(x^{(i)}; \theta\right), y^{(i)}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \eta \begin{bmatrix} -0.5 \\ 0 \\ -0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - 0.1 \begin{bmatrix} -0.5 \\ 0 \\ -0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -0.05 \\ 0 \\ -0.05 \\ -0.05 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0 \\ 0.05 \\ 0.05 \end{bmatrix}$$

# Example: Gradient Descent (Single Step)

I’m just thrilled that I have five final exams on the same day. 🙄

← Sarcastic

| Feature | Weight | Value |
|---|---|---|
| Contains 🙄 | 0 | 1 |
| Contains 😊 | 0 | 0 |
| Contains “I’m” | 0 | 1 |

Bias ($b$) = 0
Learning rate ($\eta$) = 0.1

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$$

$$\nabla_\theta L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right) = \begin{bmatrix} -0.5 \\ 0 \\ -0.5 \\ -0.5 \end{bmatrix}$$

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta L\left(f\left(x^{(i)};\theta\right), y^{(i)}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \eta \begin{bmatrix} -0.5 \\ 0 \\ -0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - 0.1 \begin{bmatrix} -0.5 \\ 0 \\ -0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -0.05 \\ 0 \\ -0.05 \\ -0.05 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0 \\ 0.05 \\ 0.05 \end{bmatrix}$$

# Mini-Batch Training

- Stochastic gradient descent chooses a single random example at a time …this can result in choppy movements!
- Often, the gradient will be computed over batches of training instances rather than a single instance
- **Batch training:** Gradient is computed over the entire dataset
  - Perfect direction, but very computationally expensive
- **Mini-batch training:** Gradient is computed over a group of *m* examples

# Mini-Batch Versions of Cross-Entropy Loss and Gradient

- Cross-Entropy Loss:
  - $L_{CE}(\text{training samples}) = -\sum_{i=1}^{m} L_{CE}(\hat{y}^{(i)}, y^{(i)})$
- Gradient:
  - $\frac{d\theta}{dw_j} = \frac{1}{m}\sum_{i=1}^{m}\left[\sigma\left(w \cdot x^{(i)} + b\right) - y^{(i)}\right]x_j^{(i)}$

# Regularization

- To avoid **overfitting**, regularization terms ($R(\theta)$) are usually added to the loss function
- These terms are used to penalize large weights (which can hinder a model's ability to generalize)
- Two common **regularization terms**:
  - L2 regularization
  - L1 regularization

# L2 Regularization

- Quadratic function of the weight values
- Square of the L2 norm (Euclidean distance of $\theta$ from the origin)
  - $R(\theta) = ||\theta||_2^2 = \sum_{j=1}^{n} \theta_j^2$

# L1 Regularization

- Linear function of the weight values
- Sum of the absolute values of the weights (Manhattan distance from the origin)
  - $R(\theta) = \|\theta\|_1 = \sum_{i=1}^{n} |\theta_i|$

# Which regularization term is better?

- L2 regularization is easier to optimize (simpler derivative)
- L2 regularization → weight vectors with many small weights
- L1 regularization → sparse weight vectors with some larger weights

# Multinomial Logistic Regression

- Other names:
  - Softmax regression
  - Maxent classification
- Uses a **softmax** function rather than a sigmoid function
- Softmax takes a vector **z** of arbitrary values (same as the sigmoid function) and maps them to a probability distribution summing to 1
  - $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{|\mathbf{Z}|} e^{z_j}}$

# Interpreting Models

- What if we want to know more than just the correct classification?
  - Why did the classifier make the decision it made?
- In these cases, we can say we want our model to be **interpretable**
- We can interpret logistic regression models by determining how much weight is associated with a given feature

# This is a key advantage of logistic regression over neural models.

- Manually-defined features facilitate **interpretability**
- **Implicitly-learned** features can be very difficult to interpret!
- Because of this, some researchers may choose to use logistic regression rather than neural models if they are particularly interested in which factors are influencing the model's decisions
  - Common example: Healthcare applications
- This allows logistic regression to function not only as a simple classification model, but as a powerful analytic tool

# Summary: Logistic Regression

- **Logistic regression** is a **discriminative** classification model used for **supervised machine learning**
- It is characterized by four key components:
    - **Feature representation**
    - **Classification function**
    - **Loss function**
    - **Optimization function**
- Classification decisions are made using a **sigmoid** function for binary logistic regression, or a **softmax** function for multinomial logistic regression
- Loss is typically computed using a **cross-entropy** function
- Weights are usually optimized using **stochastic gradient descent**
- A **regularization** term may be added to the loss function to avoid overfitting
- In addition to serving as a simple **classifier** and a useful **foundation for neural networks**, logistic regression can function as a powerful **analytic tool**