

Logistic Regression and Introduction to Vector Semantics

Natalie Parde

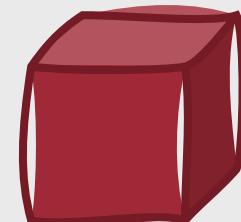
UIC CS 421

Generative Classifiers

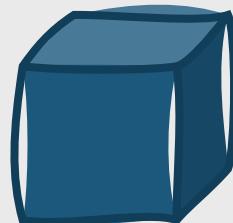
- Goal: Understand what each class looks like
 - Should be able to “generate” an instance from each class
 - To classify an instance, determines which class model better fits the instance, and chooses that as the label

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

Sarcasm



Not Sarcasm

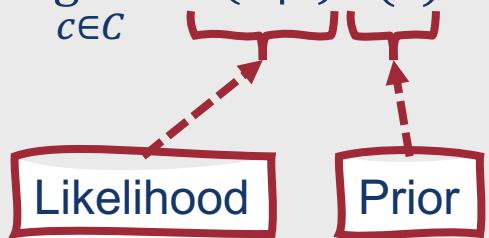


?

More formally....

- Recall the definition of naïve Bayes:

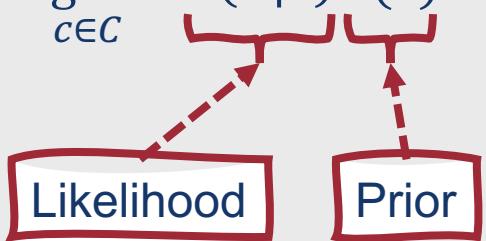
$$\hat{c} = \operatorname{argmax}_{c \in C} P(d|c)P(c)$$



More formally....

- Recall the definition of naïve Bayes:

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



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*

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:

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



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

This Week's Topics

Logistic regression
Cross-entropy loss function
Gradient descent optimization

Tuesday

Thursday

Advanced classification details
Vector semantics
TF-IDF

This Week's Topics



Logistic regression
Cross-entropy loss function
Gradient descent optimization

Tuesday

Thursday

Advanced classification details
Vector semantics
TF-IDF

Logistic Regression

- **Discriminative** supervised machine learning algorithm
- Very close relationship with **neural networks!**
- How does it compare with naïve Bayes?
 - Often performs a bit better
 - May be more complex to implement
 - May take longer to train

Logistic regression follows a standard setup that is reflected in most discriminative learning algorithms.

- **Feature representation** of the input
 - Typically, a **vector** of features $[x_1^{(j)}, x_2^{(j)}, \dots, x_n^{(j)}]$ 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

Binary Logistic Regression

- Goal:
 - 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**
 - Generalized (multinomial) case → vector of weights associated with each class
 - In true binary logistic regression we only need to learn one set of weights to discriminate between classes
- 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

Binary Logistic Regression

- 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

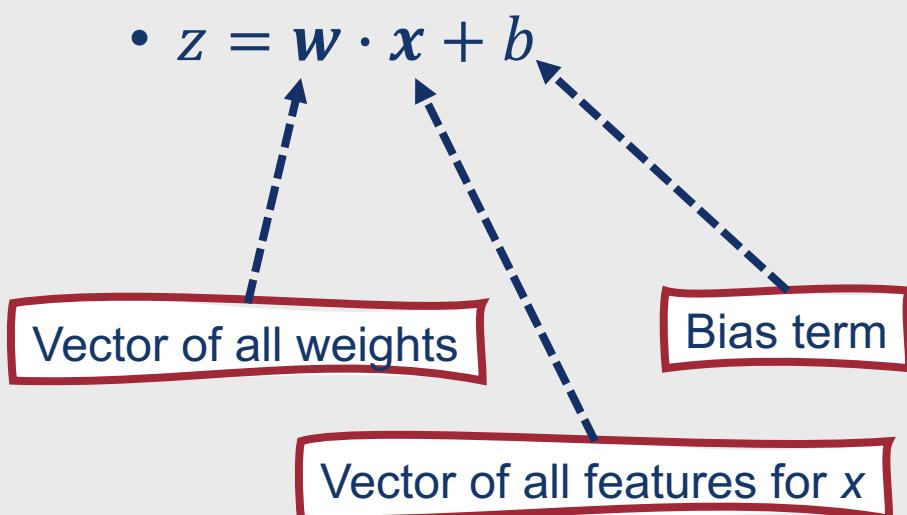
Weight for feature i

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:

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



How do we map from a linear weighted sum (z) to a probability ranging from 0-1?

- Pass z through the sigmoid function, $\sigma(z)$
 - Also called the **logistic function**, hence the name **logistic regression**

Sigmoid Function

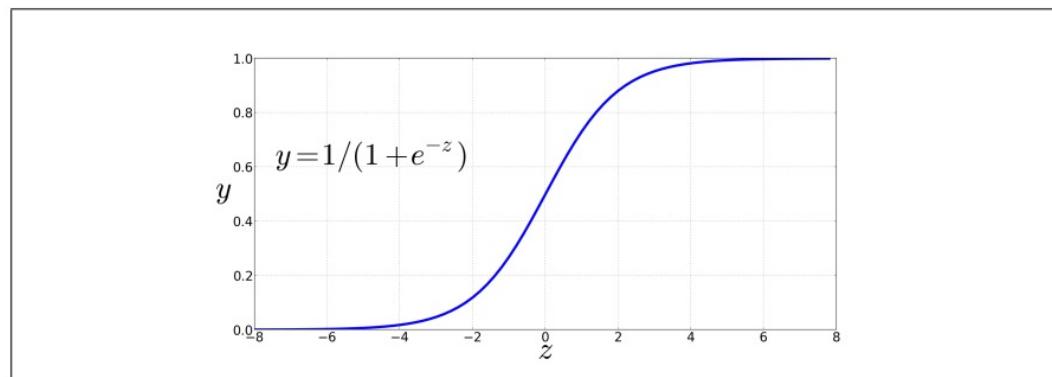


Figure 5.1 The sigmoid function $y = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range $[0, 1]$. It is nearly linear near zero.

Source: <https://web.stanford.edu/~jurafsky/slp3/5.pdf>

- 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
 - $y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-w \cdot x + b}}$
- This function has many useful properties:
 - Squashes outliers towards 0 or 1
 - Differentiable



Probabilities for all classes must sum to 1.0!

- In true binary logistic regression, you can just assume:
 - $P(y = 1) = \sigma(z)$
 - $P(y = 0) = 1 - \sigma(z)$
- If you have separate feature weights associated with different classes, you'll need to compute separate probabilities for each class

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

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	Positively associated with sarcasm
Contains 😊	-3.0	Negatively associated with sarcasm
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	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

$$\text{Bias} = 0.1$$

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

$$\text{Bias} = 0.1$$

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

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

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

$$\text{Bias} = 0.1$$

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

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

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

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

$$\text{Bias} = 0.1$$

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

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

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





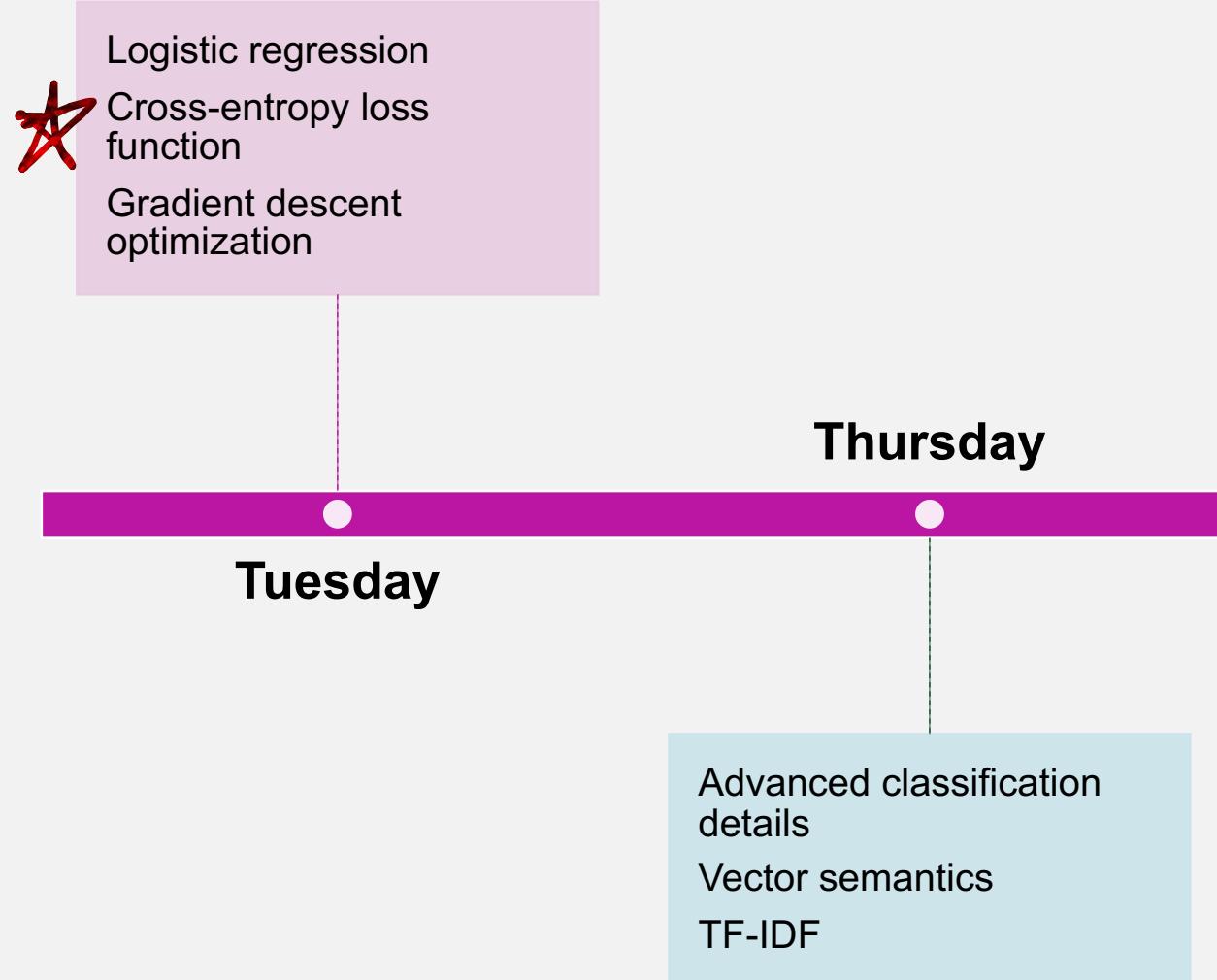
Any useful (or not useful) property of the language sample can be a feature!

- For example....
 - Specific words or n-grams
 - Information from external lexicons
 - Grammatical elements
 - Part-of-speech tags

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

This Week's Topics



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

- Measures the distance between the probability distributions of predicted and actual values
 - $loss(y_i, \hat{y}_i) = -\sum_{c=1}^{|C|} p_{i,c} \log \hat{p}_{i,c}$
 - C is the set of all possible classes
 - $p_{i,c}$ is the actual probability that instance i should be labeled with class c
 - $\hat{p}_{i,c}$ is the predicted probability that instance i should be labeled with class c
 - 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. 😳	0.96	0.04	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. 😳	0.96	0.04	1	0

$$\text{loss}(y_i, \hat{y}_i) = - \sum_{c=1}^{|C|} p_{i,c} \log \hat{p}_{i,c} = -p_{i,\text{sarcastic}} \log \hat{p}_{i,\text{sarcastic}} - p_{i,\text{not sarcastic}} \log \hat{p}_{i,\text{not sarcastic}}$$

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. 😳	0.96	0.04	1	0

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

$$\text{loss}(y_i, y'_i) = -1 * \log 0.96 - 0 * \log 0.04$$

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. 😳	0.96	0.04	1	0

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

$$\text{loss}(y_i, y'_i) = -1 * \log 0.96 - 0 * \log 0.04 = -\log 0.96 = 0.02$$

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

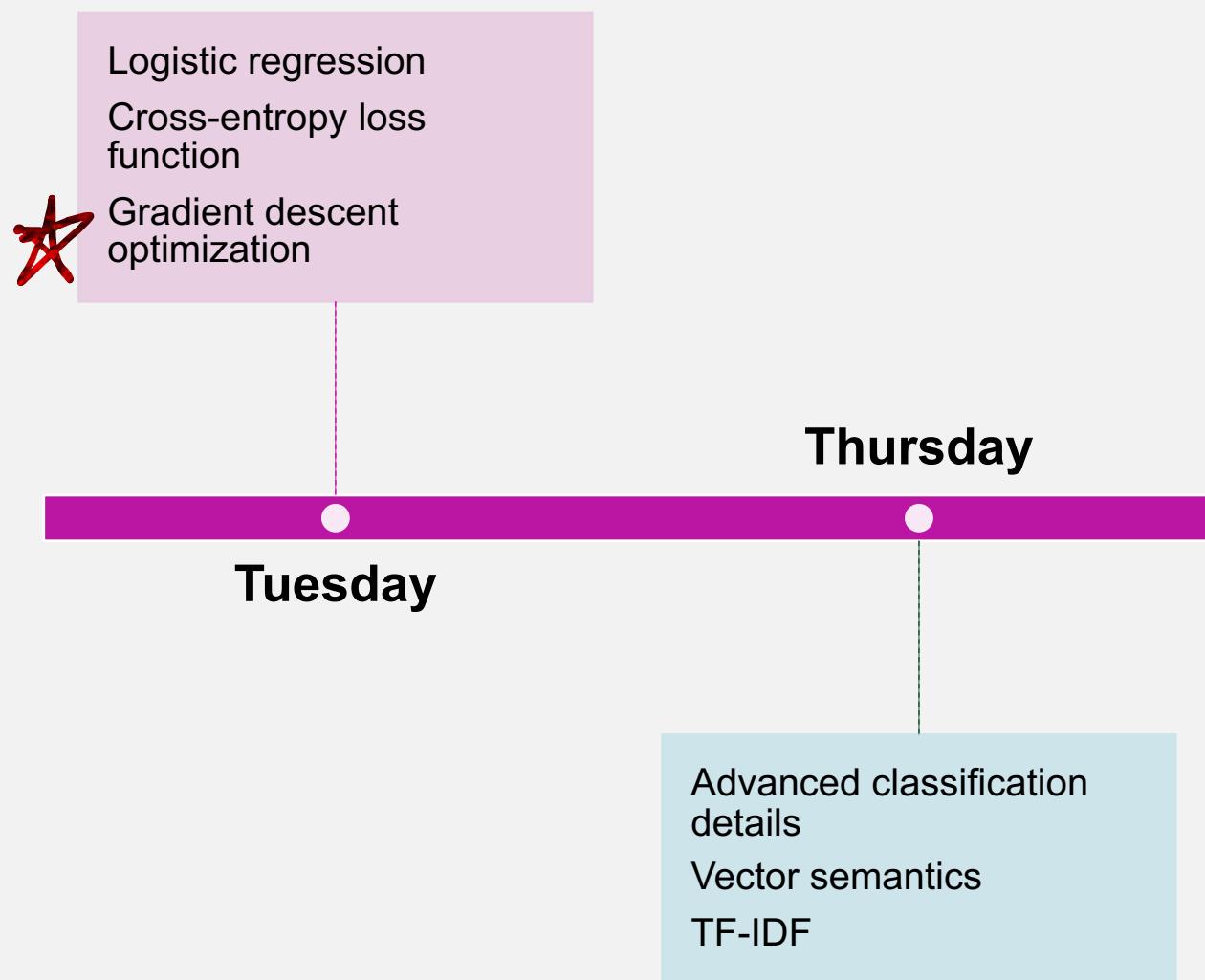
Instance	Predicted Probability: Sarcastic	Predicted Probability: Not Sarcastic	Actual Probability: Sarcastic	Actual Probability: Not Sarcastic
I'm just thrilled that I have five final exams on the same day. 😳	0.04	0.96	1	0

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

$$\text{loss}(y_i, y'_i) = -1 * \log 0.04 - 0 * \log 0.96 = -\log 0.04 = 1.40$$

Greater loss value!

This Week's Topics

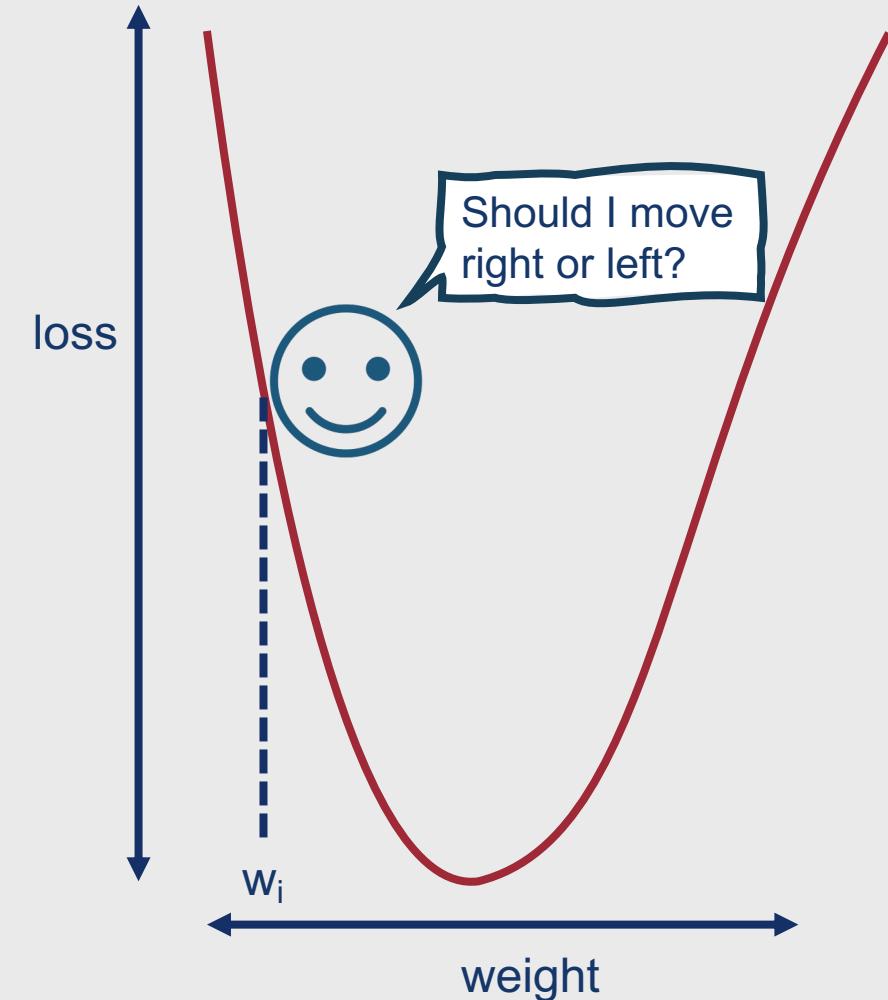


Finding Optimal Weights

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

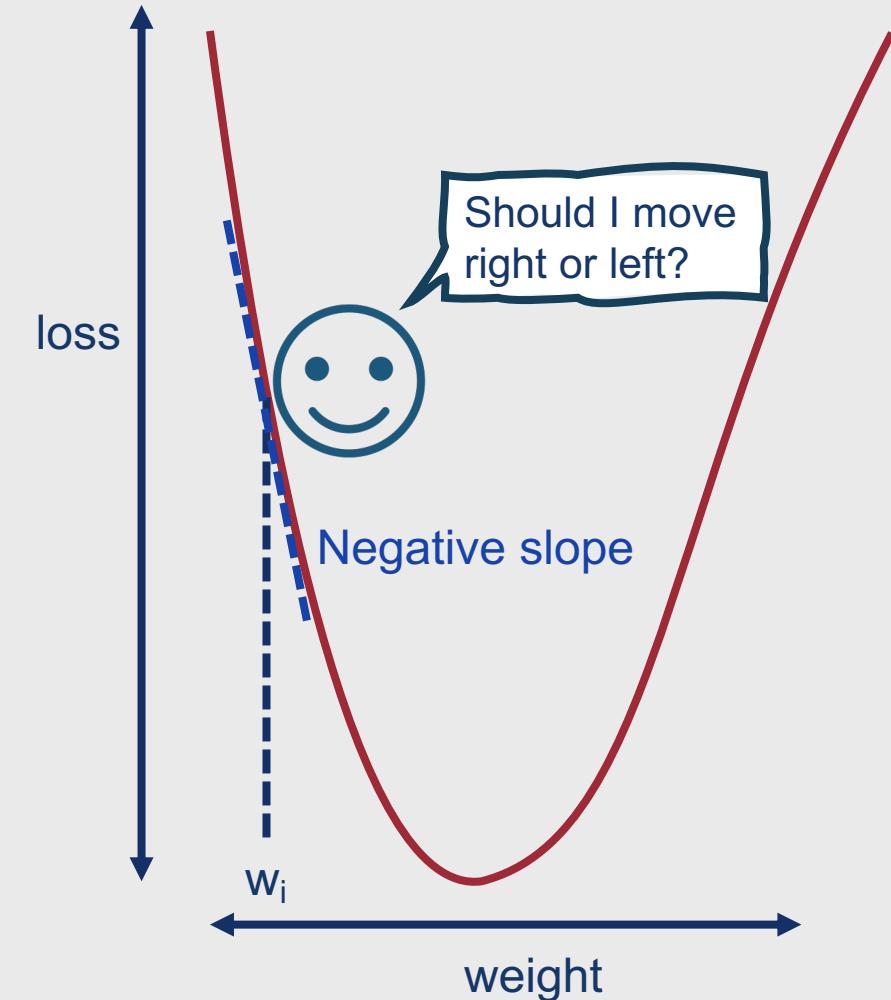
Gradient Descent

- Finds the minimum of a function by moving in the opposite direction of the function's slope
- For logistic regression, loss functions are **convex**
 - Only one minimum
 - Gradient descent starting at any point is guaranteed to find it



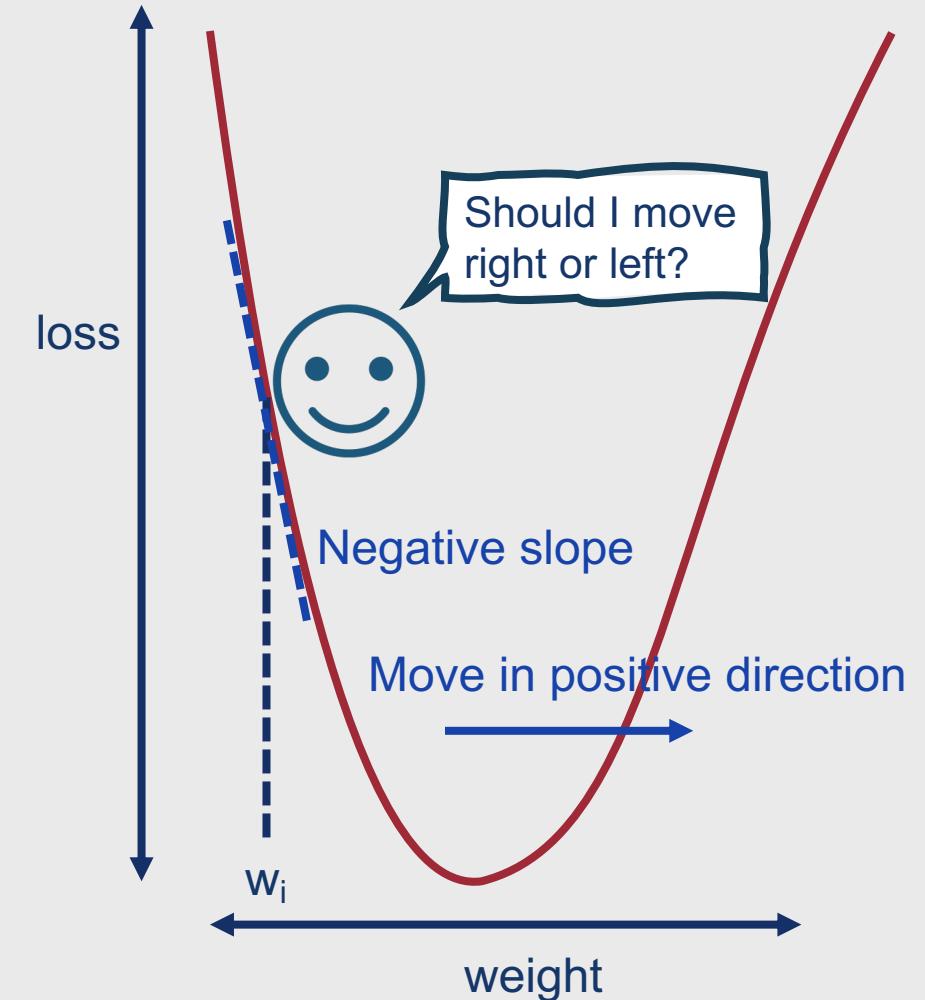
Gradient Descent

- Finds the minimum of a function by moving in the opposite direction of the function's slope
- For logistic regression, loss functions are **convex**
 - Only one minimum
 - Gradient descent starting at any point is guaranteed to find it



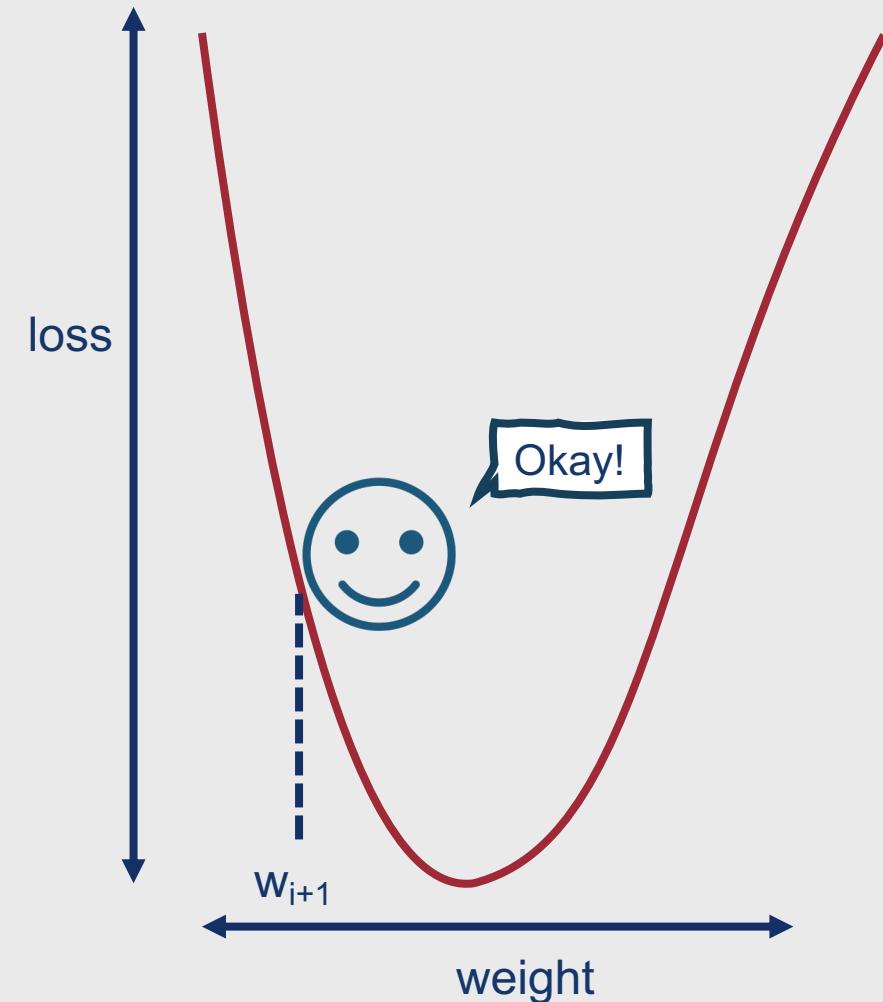
Gradient Descent

- Finds the minimum of a function by moving in the opposite direction of the function's slope
- For logistic regression, loss functions are **convex**
 - Only one minimum
 - Gradient descent starting at any point is guaranteed to find it



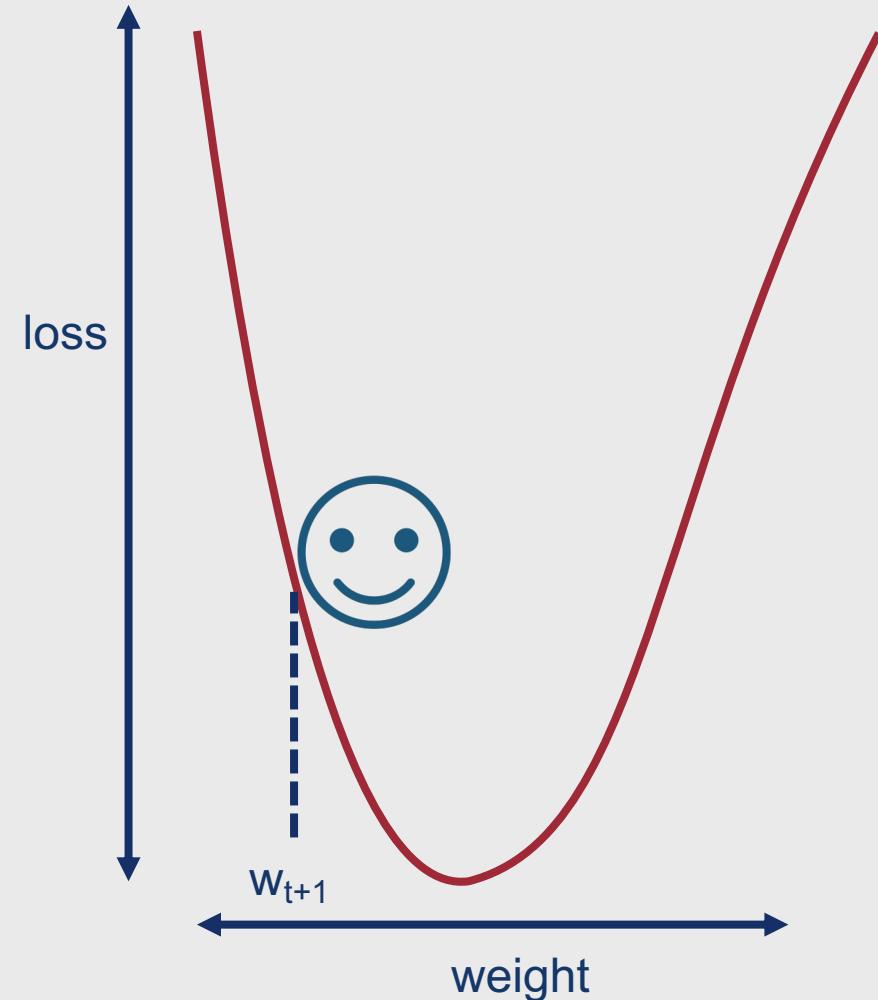
Gradient Descent

- Finds the minimum of a function by moving in the opposite direction of the function's slope
- 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 η
- Faster learning rate \rightarrow move w more on each step
- So, the change to a weight is actually:
 - $w^{t+1} = w^t - \eta \frac{d}{dw} f(x; w)$





Remember, there are weights for each feature.

- The gradient is then a vector of the slopes of each dimension:

$$\bullet \nabla_{\theta} L(f(x; \theta), y) = \begin{bmatrix} \frac{d}{dw_1} L(f(x; \theta), y) \\ \vdots \\ \frac{d}{dw_n} L(f(x; \theta), y) \end{bmatrix}$$

- This in turn means that the final equation for updating θ is:
 - $\theta_{t+1} = \theta_t - \eta \nabla L(f(x; \theta), y)$

The Gradient for Logistic Regression

- Recall our cross-entropy loss function:
 - $loss(y_i, \hat{y}_i) = -\sum_{c=1}^{|C|} y \log \hat{y} = -\sum_{c=1}^{|C|} y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b)$

- The derivative for this function is:

- $\frac{dL_{CE}(w,b)}{dw_j} = [\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y]x_j$

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:
```

```
        # What is our gradient, given our current parameters?
```

```
         $g \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$ 
```

```
         $\theta \leftarrow \theta - \eta g$  # What are our updated parameters?
```

```
return  $\theta$ 
```

Example: Gradient Descent (First 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 (First 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

$$\text{Bias } (b) = 0$$

$$\text{Learning rate } (\eta) = 0.1$$

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

Example: Gradient Descent (First 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

$$\text{Bias } (b) = 0$$

$$\text{Learning rate } (\eta) = 0.1$$

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

$$\text{Recall: } \frac{dL_{CE}(w, b)}{dw_j} = [\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y]x_j$$

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

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

Example: Gradient Descent (First 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

$$\text{Bias } (b) = 0$$

$$\text{Learning rate } (\eta) = 0.1$$

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

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

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

Example: Gradient Descent (First 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 (η) = 0.1

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

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



Example: Gradient Descent (Second Step)

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

Sarcastic

Feature	Weight	Value
Contains 😳	0.05	1
Contains 😊	0	0
Contains "I'm"	0.05	1

$$\text{Bias } (b) = 0.05$$

$$\text{Learning rate } (\eta) = 0.1$$

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

$$\text{Recall: } \frac{dL_{CE}(w, b)}{dw_j} = [\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y]x_j$$

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

$$\nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)}) = \begin{bmatrix} \frac{dL_{CE}(w, b)}{dw_1} \\ \frac{dL_{CE}(w, b)}{dw_2} \\ \frac{dL_{CE}(w, b)}{dw_3} \\ \frac{dL_{CE}(w, b)}{db} \end{bmatrix} = \begin{bmatrix} (\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y)x_1 \\ (\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y)x_2 \\ (\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y)x_3 \\ \sigma(\mathbf{w} \cdot \mathbf{x} + b) - y \end{bmatrix} = \begin{bmatrix} (\sigma(0.05 * 1 + 0 * 0 + 0.05 * 1 + .05) - 1)x_1 \\ (\sigma(0.05 * 1 + 0 * 0 + 0.05 * 1 + .05) - 1)x_2 \\ (\sigma(0.05 * 1 + 0 * 0 + 0.05 * 1 + .05) - 1)x_3 \\ \sigma(0.05 * 1 + 0 * 0 + 0.05 * 1 + .05) - 1 \end{bmatrix} = \begin{bmatrix} (\sigma(0.15) - 1)x_1 \\ (\sigma(0.15) - 1)x_2 \\ (\sigma(0.15) - 1)x_3 \\ \sigma(0.15) - 1 \end{bmatrix} = \begin{bmatrix} (0.54 - 1)x_1 \\ (0.54 - 1)x_2 \\ (0.54 - 1)x_3 \\ (0.54 - 1) \end{bmatrix} = \begin{bmatrix} -0.46 * 1 \\ -0.46 * 0 \\ -0.46 * 1 \\ -0.46 \end{bmatrix} = \begin{bmatrix} -0.46 \\ 0 \\ -0.46 \\ -0.46 \end{bmatrix}$$

Example: Gradient Descent (Second Step)

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

← Sarcastic

Feature	Weight	Value
Contains 😳	0.05	1
Contains 😊	0	0
Contains "I'm"	0.05	1

$$\text{Bias } (b) = 0.05$$

$$\text{Learning rate } (\eta) = 0.1$$

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

$$\nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)}) = \begin{bmatrix} -0.46 \\ 0 \\ -0.46 \\ -0.46 \end{bmatrix}$$

$$\theta^{t+1} = \theta^t - \eta \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)}) = \begin{bmatrix} 0.05 \\ 0 \\ 0.05 \\ 0.05 \end{bmatrix} - \eta \begin{bmatrix} -0.46 \\ 0 \\ -0.46 \\ -0.46 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0 \\ 0.05 \\ 0.05 \end{bmatrix} - 0.1 \begin{bmatrix} -0.46 \\ 0 \\ -0.46 \\ -0.46 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0 \\ 0.05 \\ 0.05 \end{bmatrix} - \begin{bmatrix} -0.046 \\ 0 \\ -0.046 \\ -0.046 \end{bmatrix} = \begin{bmatrix} 0.096 \\ 0 \\ 0.096 \\ 0.096 \end{bmatrix}$$





Mini-Batch Training

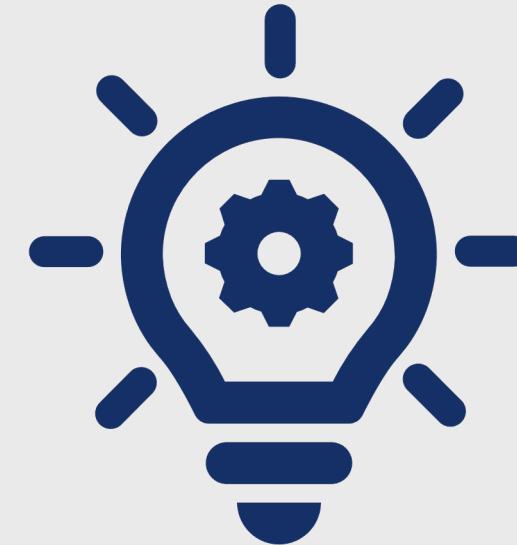
- Stochastic gradient descent chooses a single random example at a time ...this can result in choppy movements!
- Often, the gradient will be computed over batches of training instances rather than a single instance
- **Batch training:** Gradient is computed over the entire dataset
 - Perfect direction, but very computationally expensive
- **Mini-batch training:** Cross-entropy loss and gradient are computed over a group of m examples
 - $L_{CE}(\text{training samples}) = - \sum_{i=1}^m L_{CE}(\hat{y}^{(i)}, y^{(i)})$
 - $\frac{d\theta}{dw_j} = \frac{1}{m} \sum_{i=1}^m [\sigma(w \cdot x^{(i)} + b) - y^{(i)}] x_j^{(i)}$

Regularization

- To avoid **overfitting**, regularization terms ($R(\theta)$) can also be added to the loss function to penalize large weights (which can hinder a model's ability to generalize)
- Two common **regularization terms**:
 - L2 regularization
 - Quadratic function of the weight values
 - Square of the L2 norm (Euclidean distance of θ from the origin)
 - $R(\theta) = \|\theta\|_2^2 = \sum_{j=1}^n \theta_j^2$
 - Easier to optimize
 - Good for weight vectors with many small weights
 - L1 regularization
 - Linear function of the weight values
 - Sum of the absolute values of the weights (Manhattan distance from the origin)
 - $R(\theta) = \|\theta\|_1 = \sum_{i=1}^n |\theta_i|$
 - Good for sparse weight vectors with some larger weights



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

How do we evaluate logistic regression classifiers?

- Same way as naïve Bayes (and all other) text classifiers!
 - Precision
 - Recall
 - F1
 - Accuracy

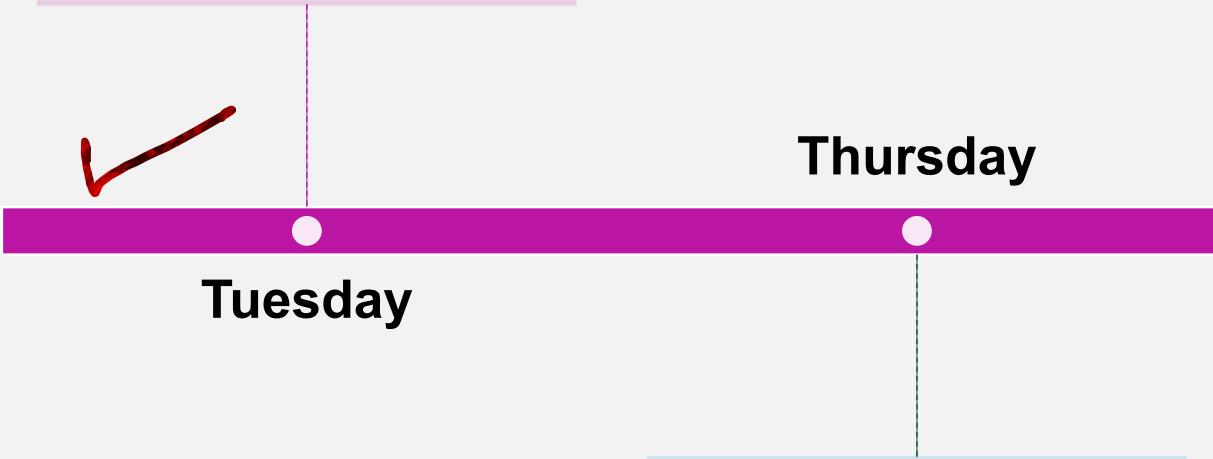


Summary: Logistic Regression

- 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
- 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**

This Week's Topics

Logistic regression
Cross-entropy loss function
Gradient descent optimization



✗ Advanced classification details
Vector semantics
TF-IDF

Training, Validation, and Test Sets

- Text corpora should generally be divided into three separate subsets (sometimes called **splits** or **folds**):
 - **Training:** Used to train the classification model
 - **Validation:** Used to check performance while developing the classification model
 - **Test:** Used to check performance only after model development is finished
- The percentage of data in each fold can vary
 - In many cases, researchers like to reserve 75% or more of their corpus for training, and split the remaining data between validation and test

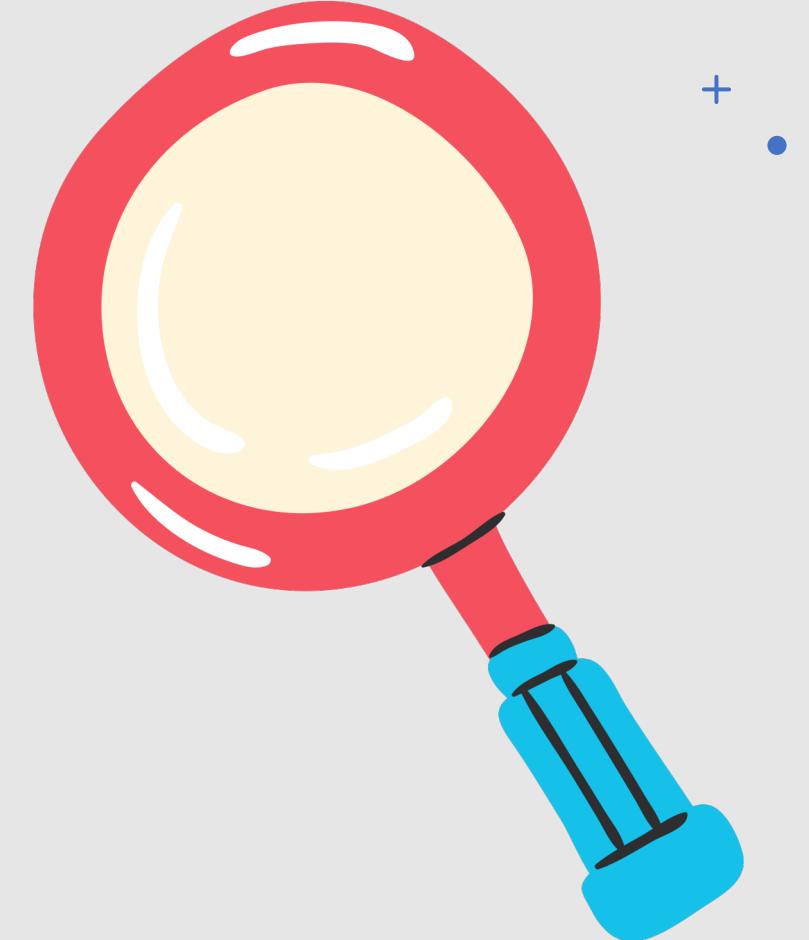


Why is a validation set necessary?

- It helps avoid overfitting
 - **Overfitting:** Artificially boosting performance on the test set by tweaking parameters such that they are particularly well-suited for the test data
- Why is overfitting bad?
 - Models that have been overfit tend to perform poorly on unseen samples in the same domain
 - This means that they cannot generalize easily to real-world scenarios, where the entire test set is not known in advance

What if the entire dataset is pretty small?

- In cases where the entire dataset is small, it may be undesirable to reserve an entire fold of data for validation
 - Smaller training set (less data from which to learn)
 - Smaller test set (less data on which to evaluate)
- In these cases, a reasonable alternative is cross-validation
 - Randomly split the dataset into k folds
 - Train on $k-1$ folds and test on the other fold
 - Repeat with a different combination of $k-1$ folds and other fold
 - Overall, repeat k times
 - Average the performance across all k training/test runs



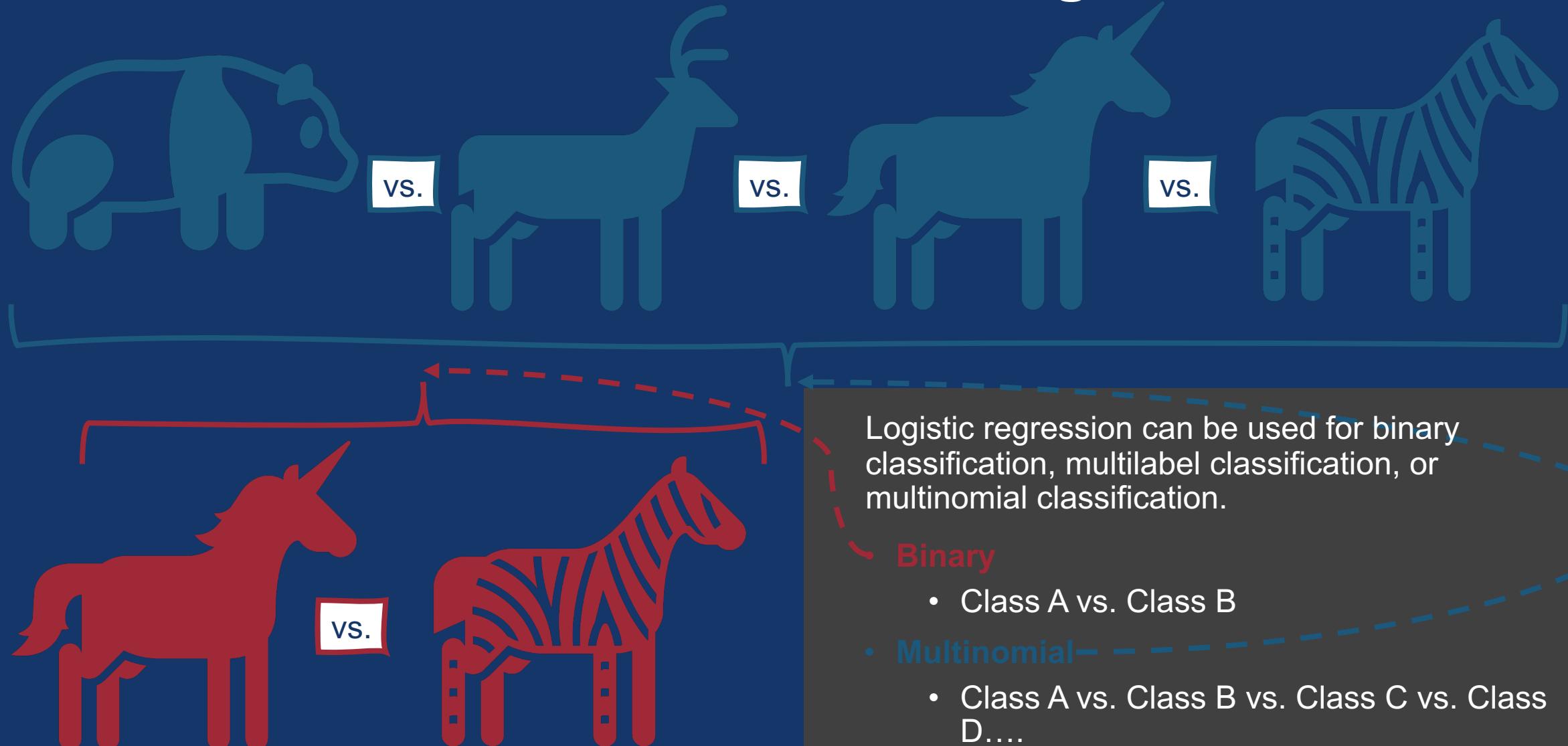


Cross-Validation

66

- Most commonly, $k=10$ in cross-validation
 - Referred to as **10-fold cross-validation**
- One problem with cross-validation?
 - To avoid overfitting, we can't look at any of the data because it's technically *all* test data!
- To avoid this issue, we can:
 - Create a fixed training set and test set
 - Perform k -fold cross-validation on the training set (where it's fine to look at the data) while developing the model
 - Evaluate the model on the test set as usual, training on the entire training set

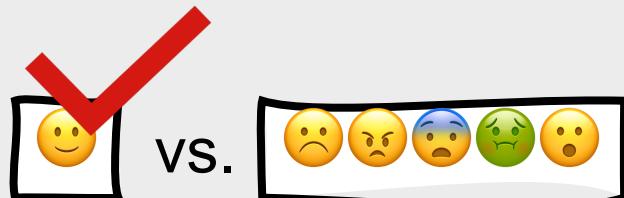
What about more advanced settings?



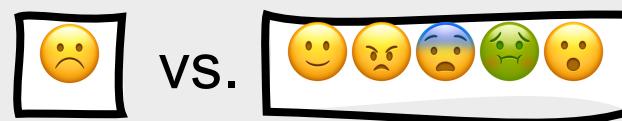
Multinomial Logistic Regression

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

One vs. All Classification



vs.



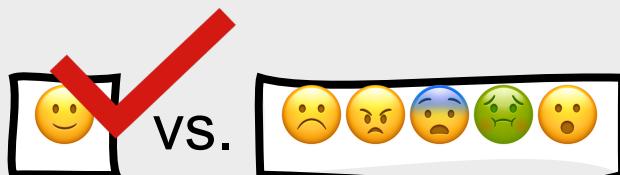
vs.



vs.

- If we don't want to build a true multinomial model, we can also build multiple one vs. all classifiers
- How do we do this?
 - Build separate binary classifiers for each class
 - Run each classifier on the test document
 - Choose the label from the classifier with the highest score

Multi-Label Classification

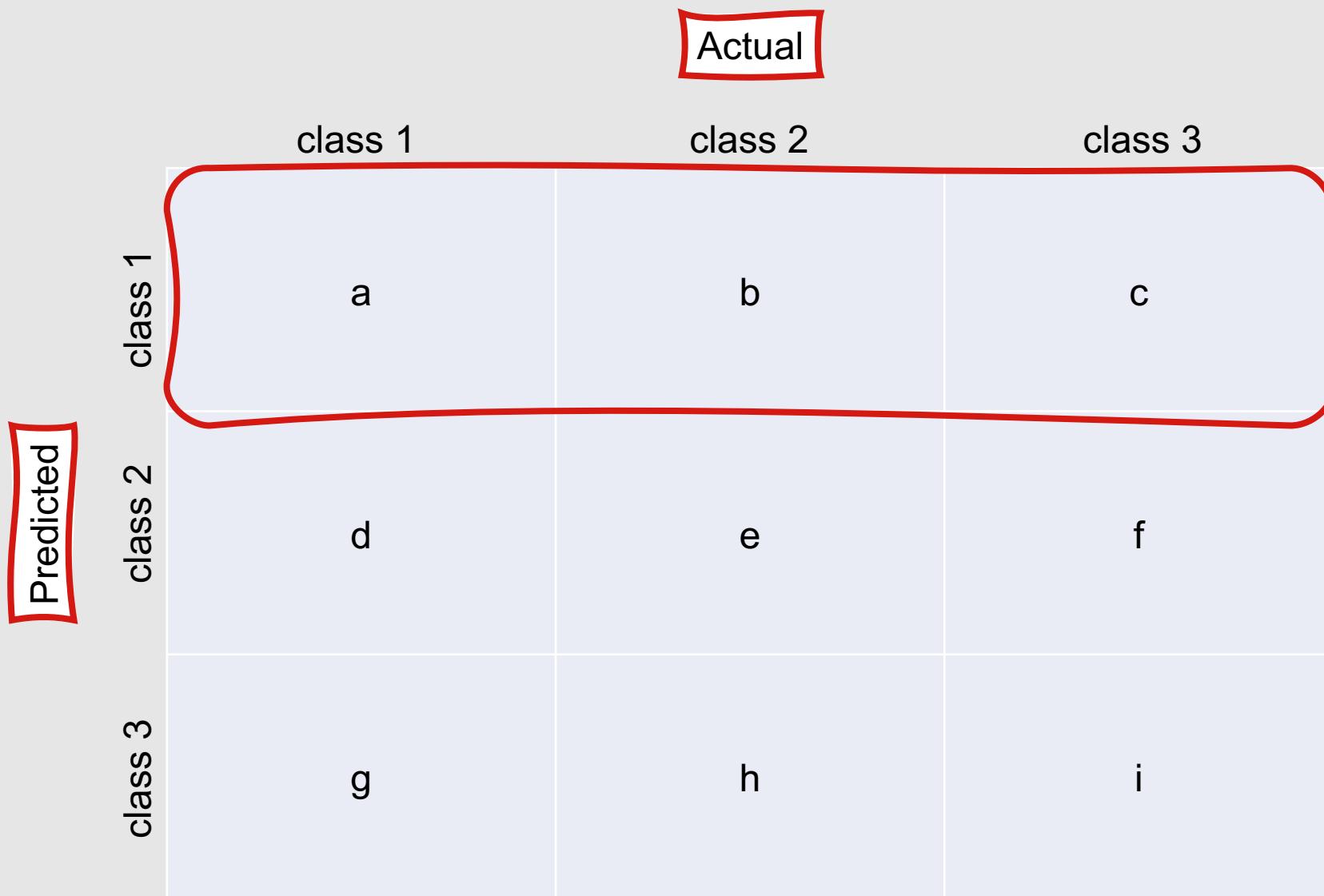


- Each document can be assigned more than one label
- How do we do this?
 - Build separate binary classifiers for each class
 - Positive class vs. every other class
 - Run each classifier on the test document
 - Each classifier makes its decision independently of the other classifiers, therefore allowing multiple labels to be assigned to the document

Multi-Class Contingency Matrix

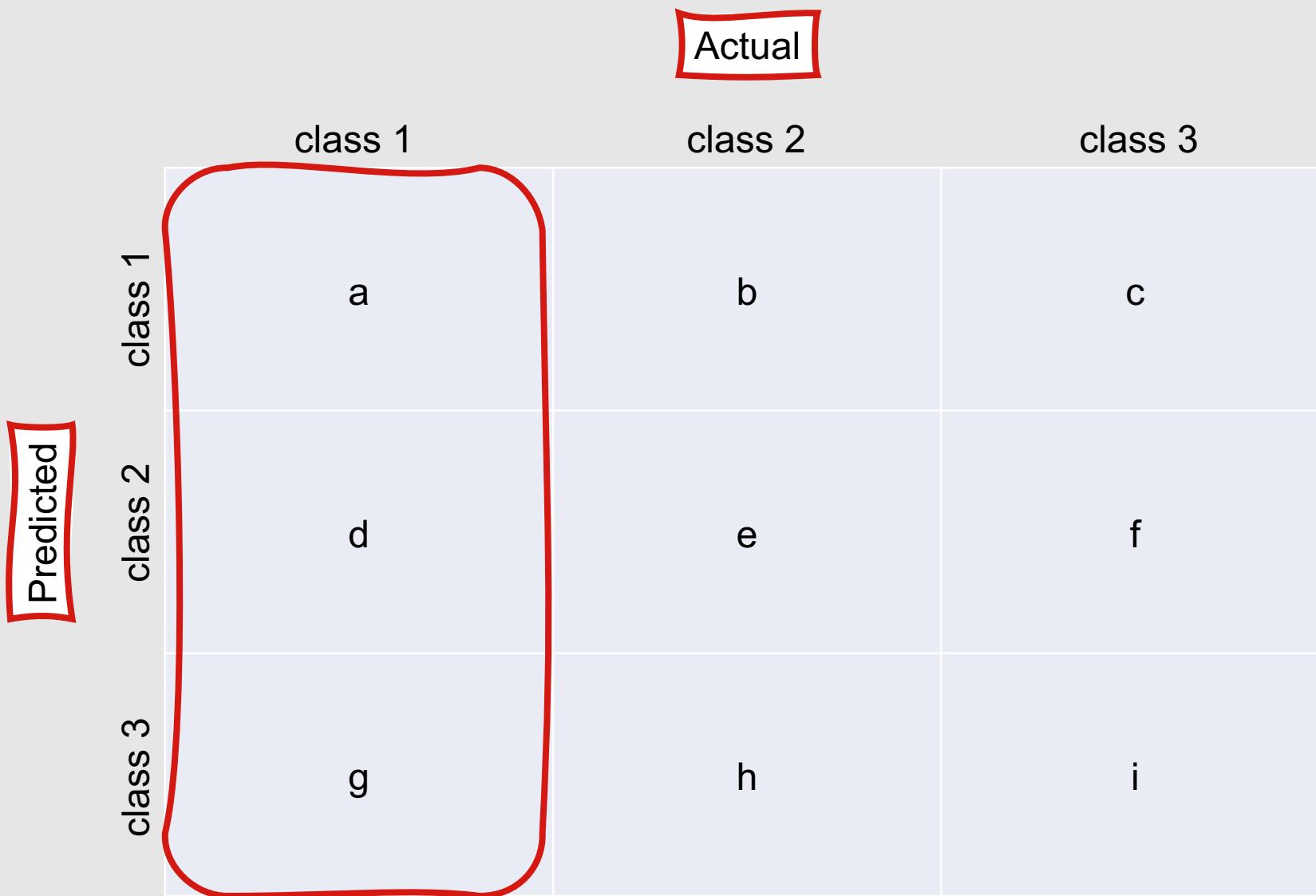
			Actual					
			class 1	class 2	class 3			
			a	b	c			
Predicted	class 1	class 2	a	b	c	class 1	class 2	class 3
class 2	d	e	f	g	h	class 2	class 3	class 3
class 3						i		

Multi-Class Precision



$$\text{Precision} = \frac{a}{a+b+c}$$

Multi-Class Recall



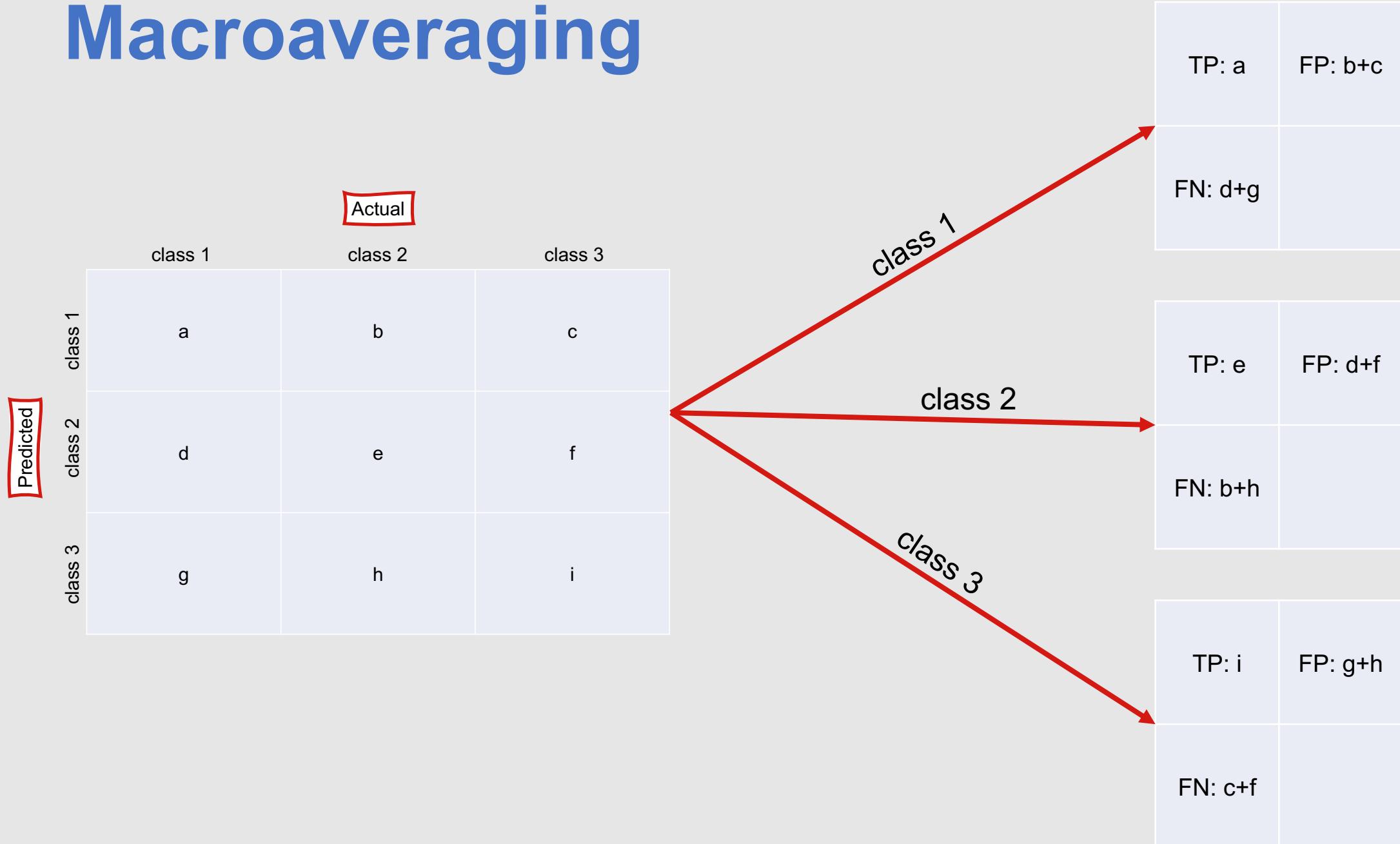
$$\text{Precision} = \frac{a}{a+b+c}$$

$$\text{Recall} = \frac{a}{a+d+g}$$

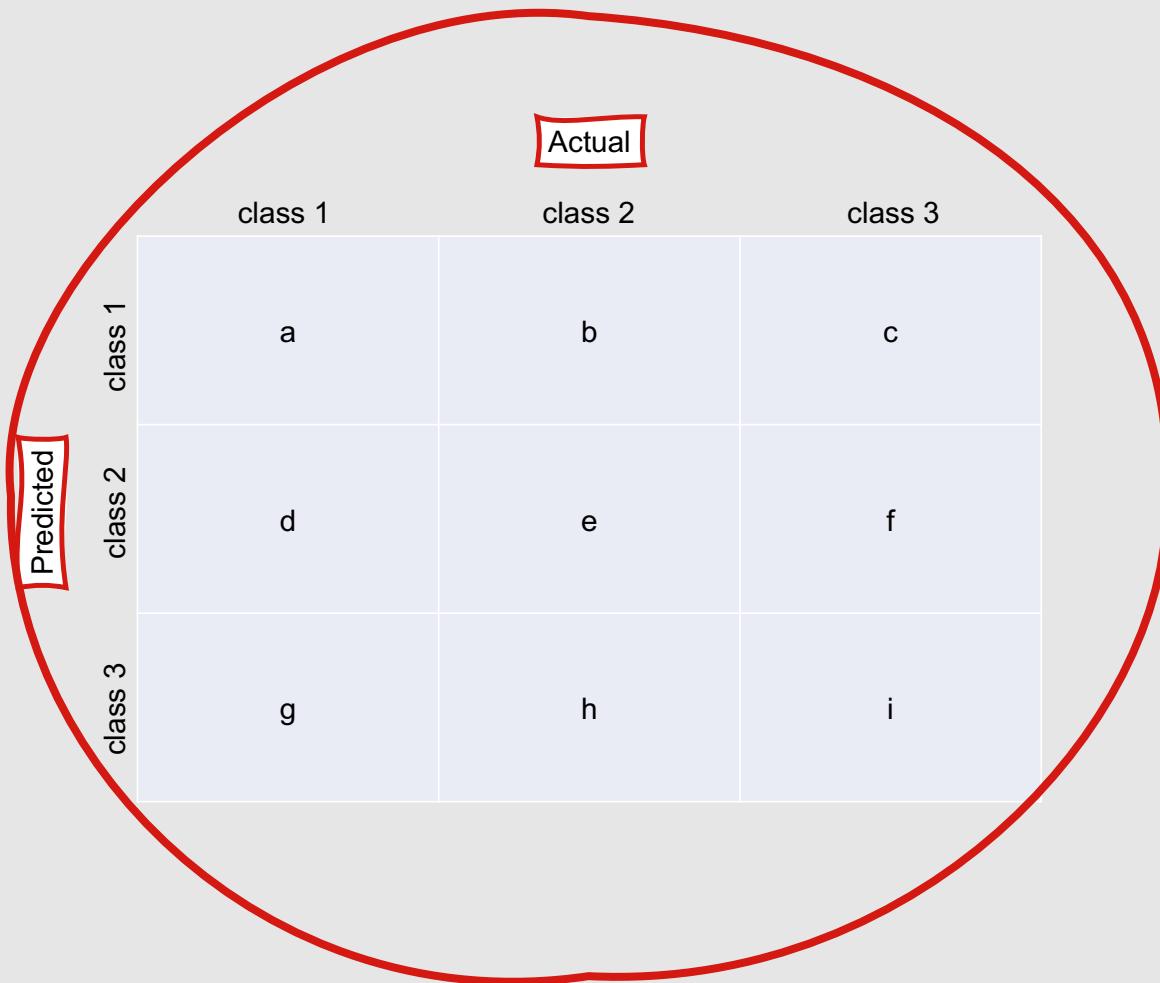
Macroaveraging and Microaveraging

- We can check the system's **overall performance** in multi-class classification settings by combining all of the precision values (or all of the recall values) in two ways:
 - **Macroaveraging**
 - **Microaveraging**
- **Macroaveraging:** Compute the performance for each class, and then average over all classes
- **Microaveraging:** Collect decisions for all classes into a single contingency table, and compute precision and recall from that table

Macroaveraging



Macroaveraging



$$\text{Precision}_{\text{Class1}} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

TP: a FP: b+c

FN: d+g

$$\text{Precision}_{\text{Class2}} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

TP: e FP: d+f

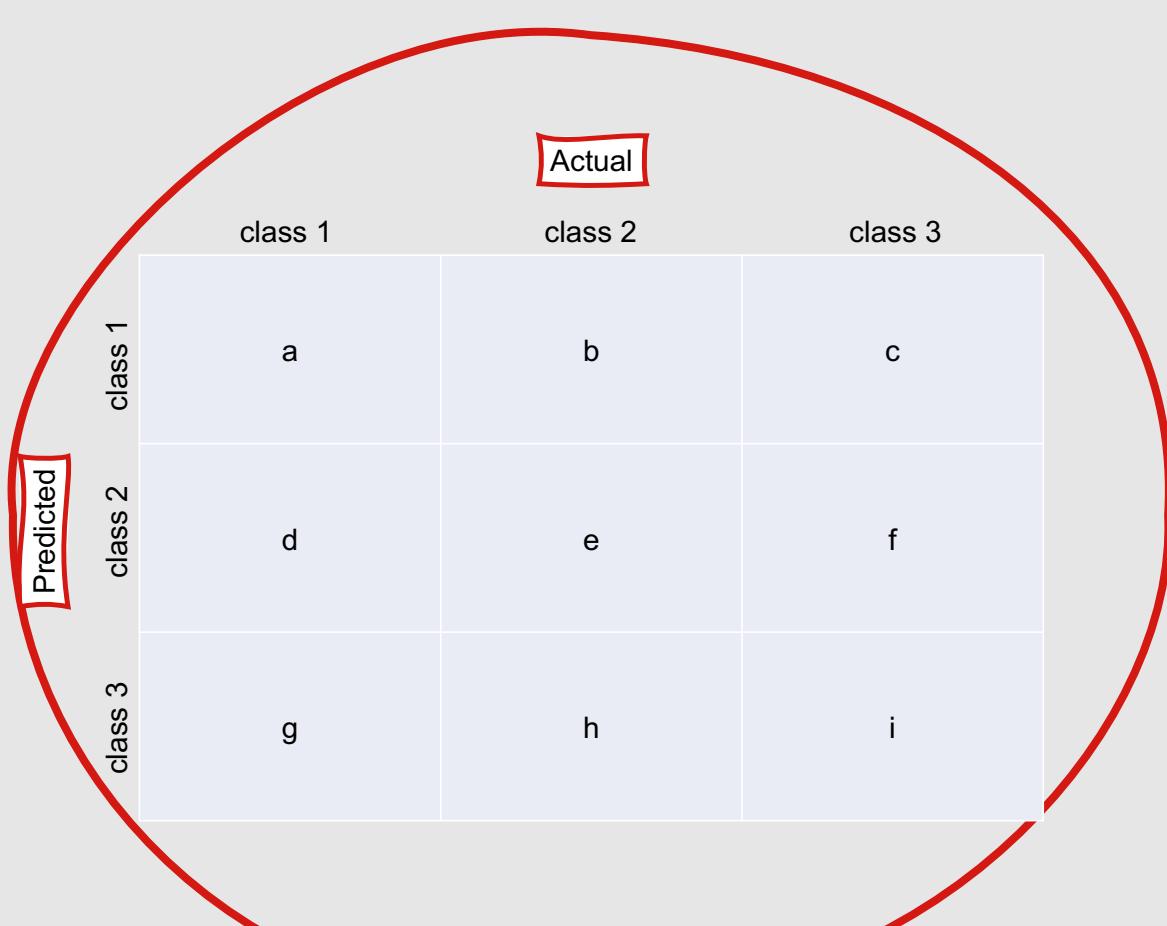
FN: b+h

$$\text{Precision}_{\text{Class3}} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

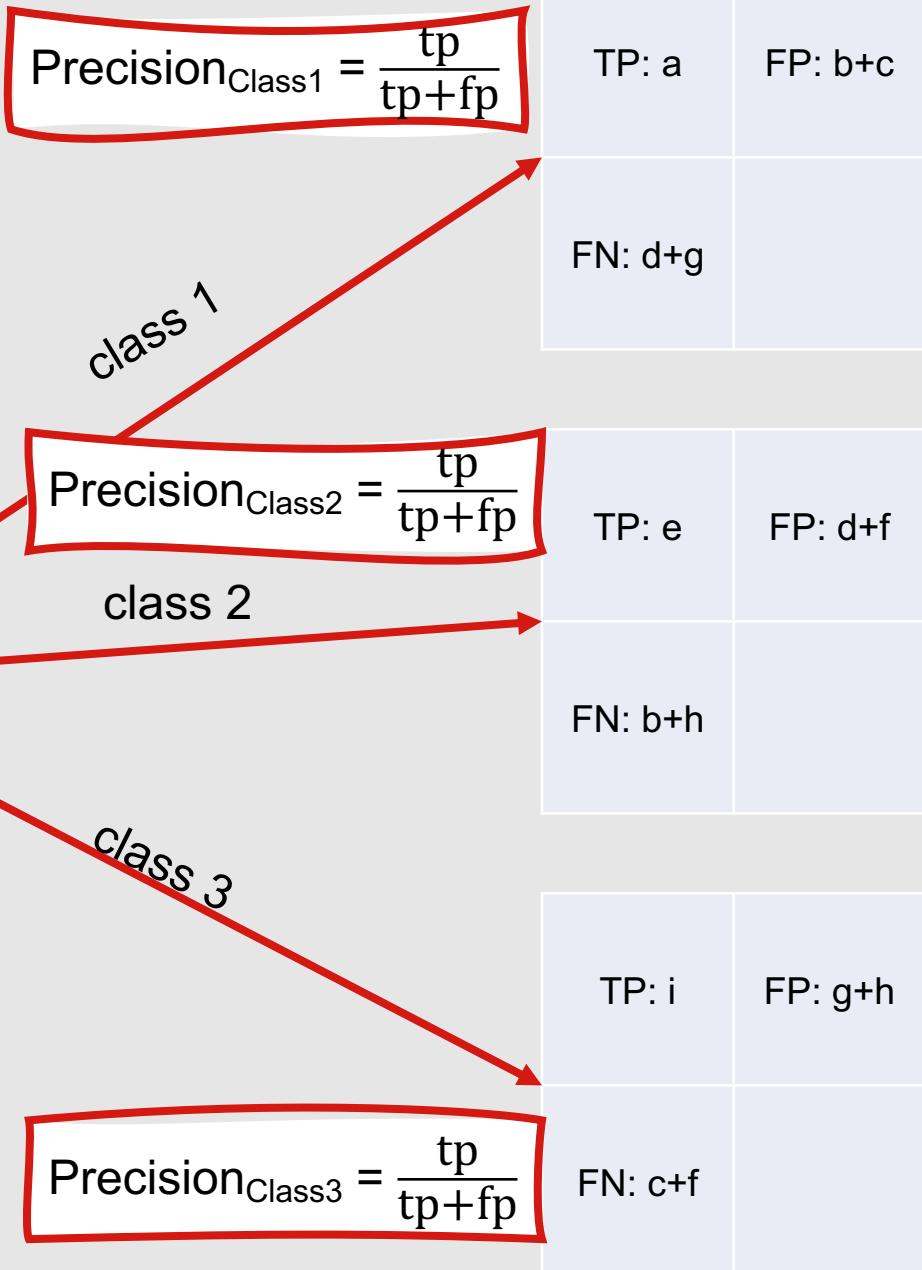
TP: i FP: g+h

FN: c+f

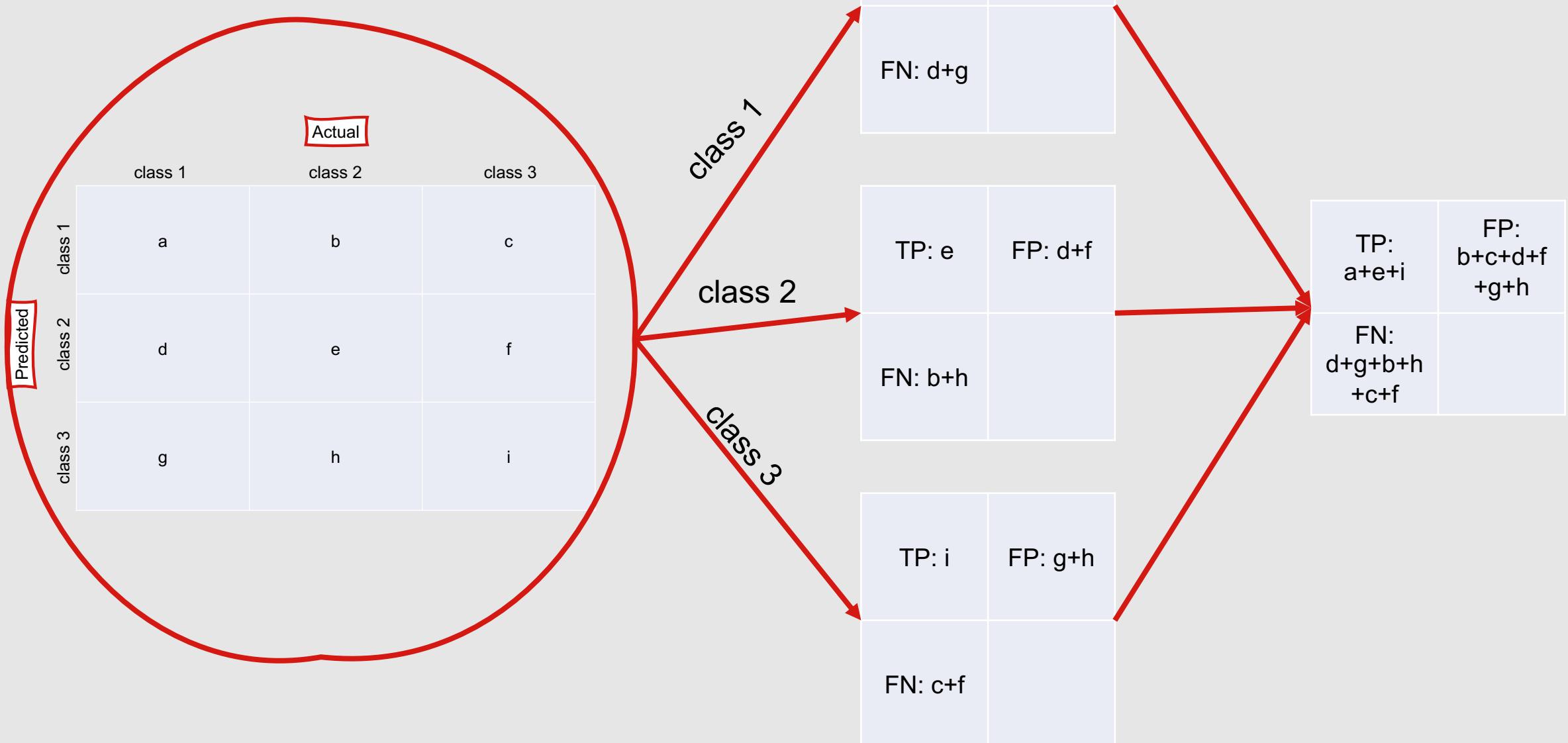
Macroaveraging



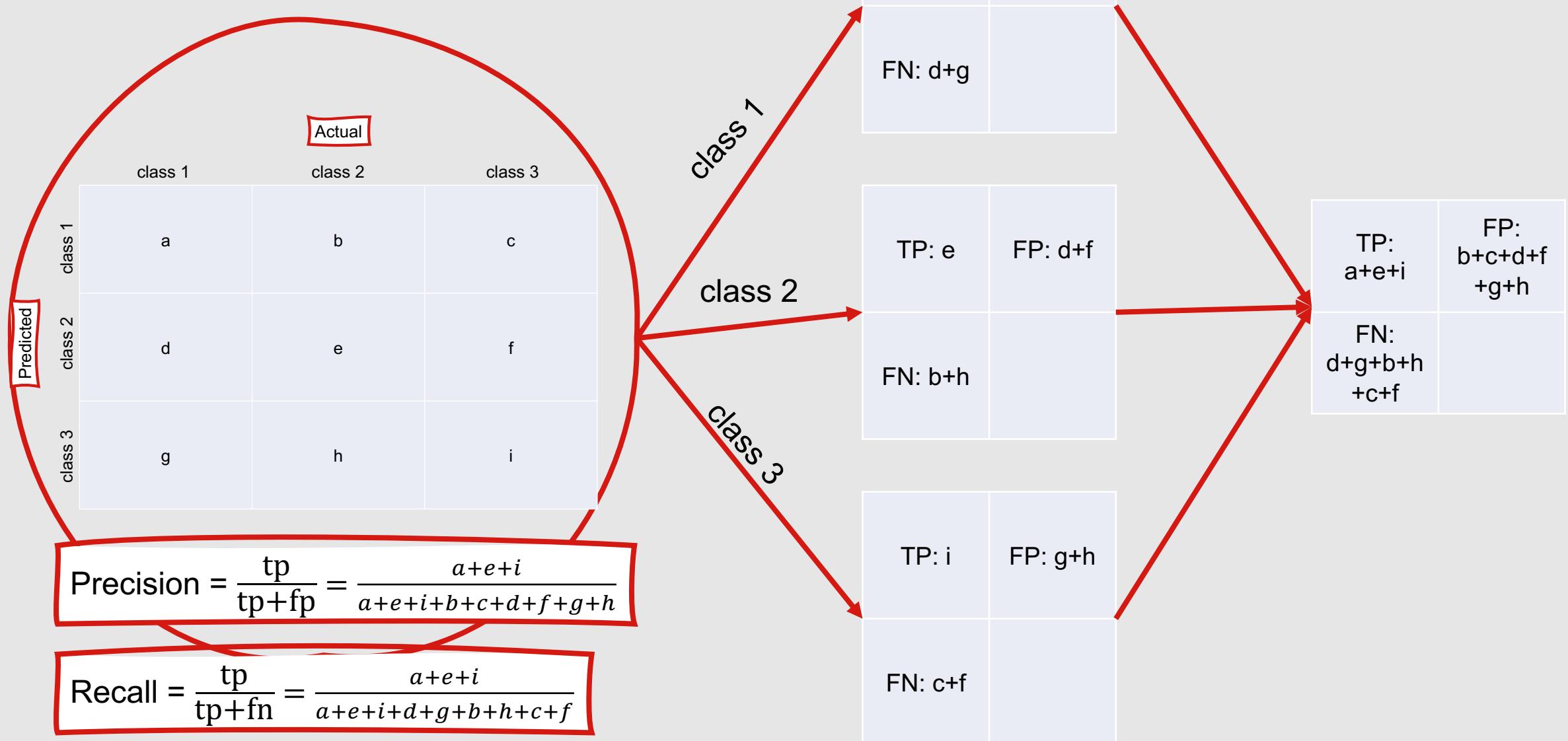
$$\text{Macroaveraged Precision} = \frac{\text{Precision}_{\text{Class1}} + \text{Precision}_{\text{Class2}} + \text{Precision}_{\text{Class3}}}{3}$$



Microaveraging



Microaveraging



What's better: Microaveraging or macroaveraging?

- Depends on the scenario!
- Microaverages tend to be dominated by more frequent classes, since the counts are all pooled together
- Macroaverages are evenly distributed across classes
- Thus, if performance on all classes is equally important, macroaveraging is probably better; if performance on the most frequent class is more important, microaveraging is probably better

Statistical Significance Testing

- We've trained and evaluated our classification model ...how do we know it's better (or worse) than other alternate models?
- We can't necessarily say that Model A is better than Model B purely because its precision/recall/ F_1 /accuracy is higher!
 - Model A might be performing better than Model B just due to chance
- To confirm our suspicions that Model A really is better, we need to perform **statistical significance testing** to reject the **null hypothesis** that Model A is better than Model B just due to chance

Null Hypothesis

- Given observation: Model A performs x% better than Model B
- **Null Hypothesis:** This is due to chance, rather than some meaningful reason
 - If we had many test sets of the same size as ours, and measured Model A's and Model B's performance on all of them, then on average Model A might accidentally perform x% better than Model B



P-Value

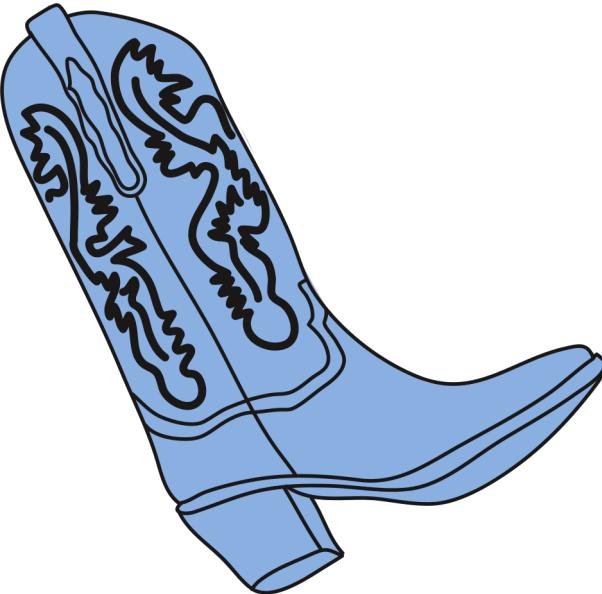
The probability that we'll see equally big performance differences by chance is referred to as the *p-value*

If the *p*-value is sufficiently low (generally 0.05 or 0.01), then we can **reject the null hypothesis**

If we reject the null hypothesis, that means that we have identified a **statistically significant difference** between the performance of Model A and Model B

How do we determine our p -value?

- We can select from among many possible methods based on several factors
 - Distribution of our data
 - Number of samples in our dataset
- Most NLP tasks do not involve data from a known distribution
- Because of this, it's common to use **non-parametric tests** to determine statistical significance:
 - Bootstrap test



Bootstrap Test

- Repeatedly draws many small samples from the test set, with replacement
- Assumes each sample is representative of the overall population
- For each sample, checks to see how well Model A and Model B perform on it
- Keeps a running total of the number of samples for which the difference between Model A's and Model B's performance is more than twice as much as the difference between Model A's and Model B's performance in the overall test set
- Divides the final total by the total number of samples checked to determine the p -value

Formal Algorithm: Bootstrap Test

```
Calculate  $\delta(x)$  # Performance difference between Models A and B
for i = 1 to b do: # b = number of samples
    for j = 1 to n do: # n = size of bootstrap sample
        Randomly select a test instance and add it to the
        bootstrap sample
    Calculate  $\delta(x^{*(i)})$  # Performance difference between Models A
                            # and B for the bootstrap sample  $x^{*(i)}$ 
for each  $x^{*(i)}$ :
    s = s+1 if  $\delta(x^{*(i)}) > 2\delta(x)$ 
p(x) = s/b
```

Interested in learning more about statistical significance testing in NLP?

- Paper: <https://aclanthology.org/P18-1128.pdf>
- Book:
<https://www.morganclaypool.com/doi/10.2200/S00994ED1V01Y202002HLT045>



This Week's Topics

Logistic regression
Cross-entropy loss function
Gradient descent optimization

Tuesday

Thursday



Advanced classification details
Vector semantics
TF-IDF

Vector Semantics

- Facilitates a form of **representation learning** based on the notion that similar words tend to occur in similar environments
 - This notion is known as the distributional hypothesis, which was first formulated by linguists in the 1950s
 - Joos (1950)
 - Harris (1954)
 - Firth (1957)
 - Self-supervised

1. Does language have a distributional structure? For the purposes of the present discussion, the term structure will be used in the following non-rigorous sense: A set of phonemes or a set of data is structured in respect to some feature, to the extent that we can form in terms of that feature some organized system of statements which describes the members of the set and their interrelations (at least up to some limit of complexity). In this sense, language can be structured in respect to various independent features. And whether it is structured (to more than a trivial extent) in respect to, say, regular historical change, social intercourse, meaning, or distribution—or to what extent it is structured in any of these respects—is a matter decidable by investigation. Here we will discuss how each language can be described in terms of a distributional structure, i.e. in terms of the occurrence of parts (ultimately sounds) relative to other parts, and how this description is complete without intrusion of other features such as history or meaning. It goes without saying that other studies of language—historical, psychological, etc.—are also possible, both in relation to distributional structure and independently of it.

The distribution of an element will be understood as the sum of all its environments. An environment of an element A is an existing array of its co-occurrences, i.e. the other elements, each in a particular position, with which A occurs to yield an utterance. A's co-occurrences in a particular position are called its selection for that position.

1.1. Possibilities of structure for the distributional facts.

To see that there can be a distributional structure we note the following: First, the parts of a language do not occur arbitrarily relative to each other: each element occurs in certain positions relative to certain other elements. The perennial man in the street believes that when he speaks he freely puts together whatever elements have the meanings he intends; but he does so only by choosing members of those classes that regularly occur together, and in the order in which these classes occur.

Second, the restricted distribution of classes persists for all their occurrences; the restrictions are not disregarded arbitrarily, e.g. for semantic needs. Some logicians, for example, have considered that an exact distributional description of natural languages is impossible because of their inherent vagueness. This is not quite the case. All elements in a language can be grouped into classes whose relative occurrence can be stated exactly. However, for the occurrence of a particular member of one class relative to a particular member of another class it would be necessary to speak in terms of probability, based on the frequency of that occurrence in a sample.

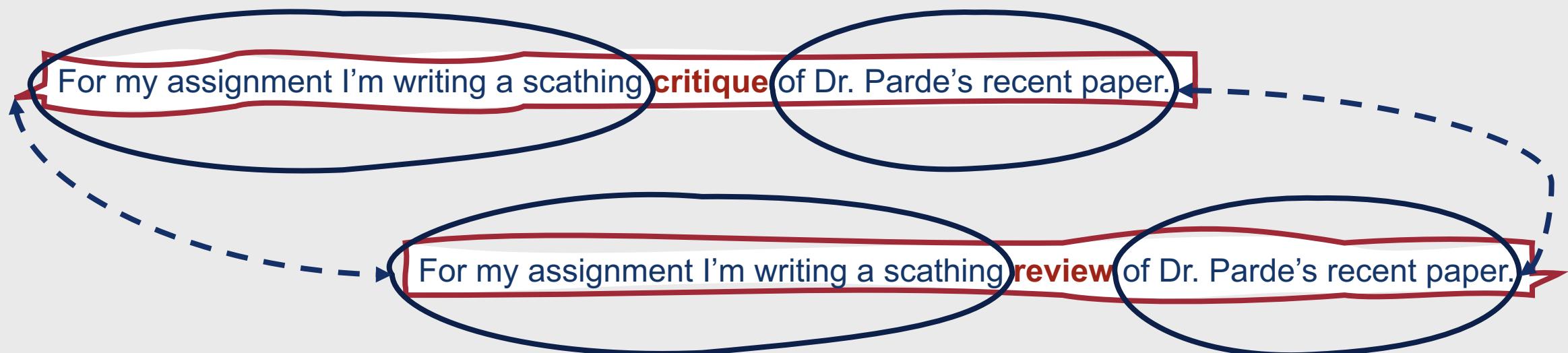
Third, it is possible to state the occurrence of any element relative to any other element, to the degree of exactness indicated above, so that distributional state-

Vector Semantics

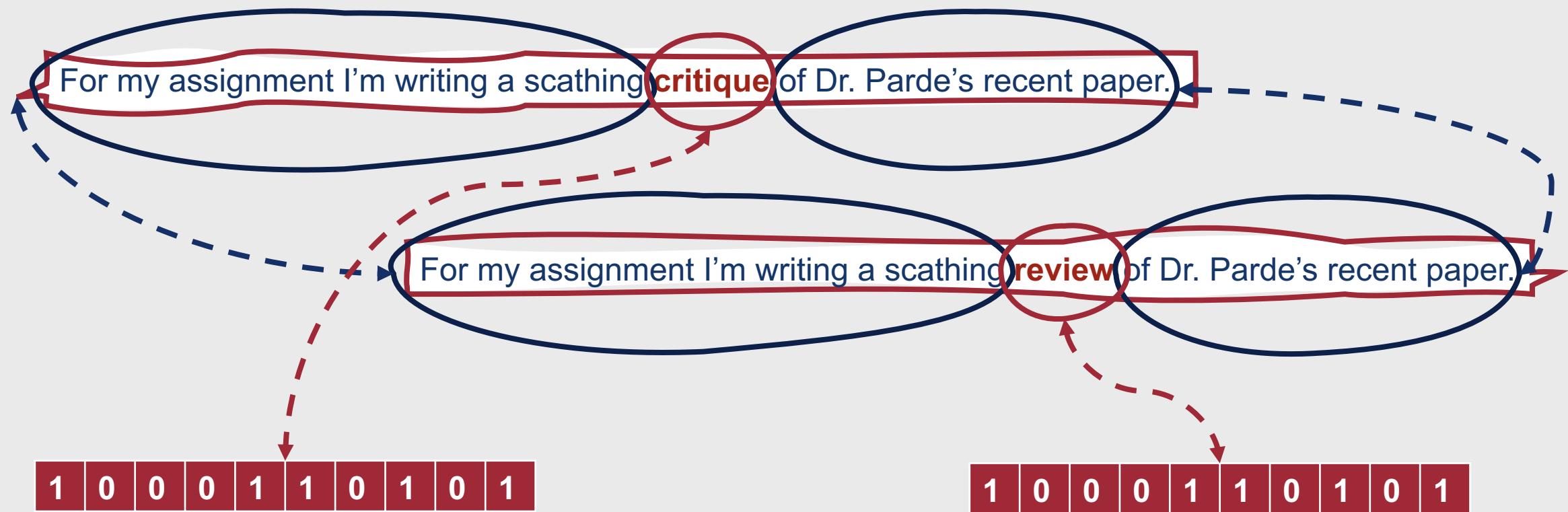
For my assignment I'm writing a scathing **critique** of Dr. Parde's recent paper.

For my assignment I'm writing a scathing **review** of Dr. Parde's recent paper.

Vector Semantics



Vector Semantics

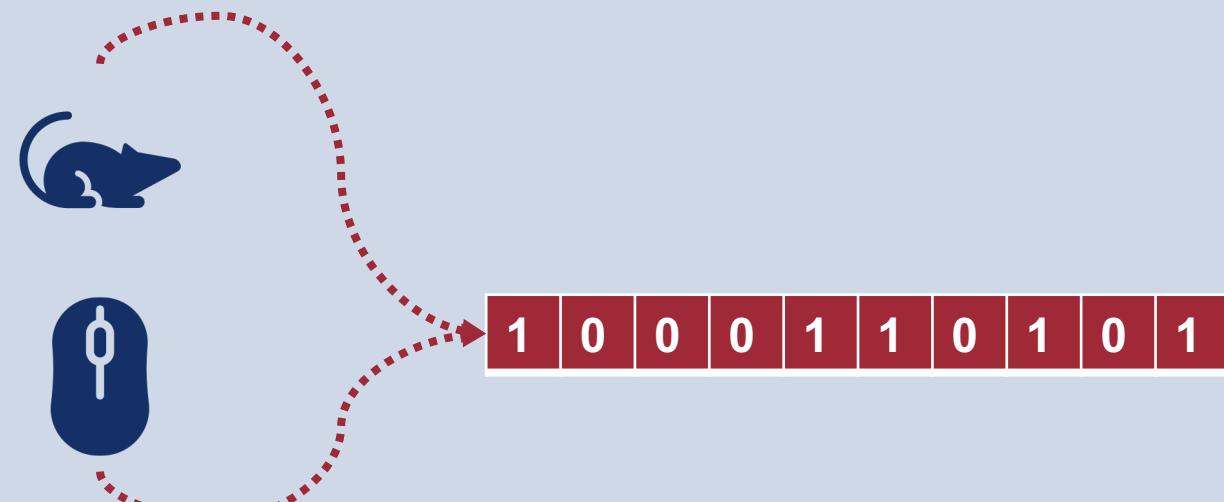


There are many ways to make use of the distributional hypothesis!

- **Classical word vectors**
 - Bag of words representations and their variations
- **Implicitly learned word vectors**
 - Word2Vec
 - GloVe
- All of these approaches seek to encode the same linguistic phenomena observed in studies of lexical semantics

Lemmas and Senses

- **Lemma:** The base form of a word
 - Papers → paper
 - Mice → mouse
- **Word Sense:** Different aspects of meaning for a word
 - Mouse (1): A small rodent
 - Mouse (2): A device to control a computer cursor
- Words with the same lemma should (hopefully!) reside near one another in vector space
- Words with the same sense might also reside near one another in vector space, depending on the representation learning technique



- When a word sense for one word is (nearly) identical to the word sense for another word
- **Synonymy:** Two words are synonymous if they are substitutable for one another in any sentence without changing the situations in which the sentence would be true
 - This means that the words have the same **propositional meaning**

For my assignment I'm writing a scathing **critique** of Dr. Parde's recent paper.

For my assignment I'm writing a scathing **review** of Dr. Parde's recent paper.

Synonymy

Word Similarity and Relatedness

- **Word similarity:** Words are not synonyms, but they can be used in the same contexts as one another
- **Word Relatedness:** Words are associated with one another based on their shared participation in an event

Natalie grabbed the **purple** coffee mug.

Natalie grabbed the **green** coffee mug.



Semantic Frames

- **Semantic Frame:** A set of words that denote perspectives or participants in a particular type of event
 - Commercial Transaction = {buyer, seller, goods, money}
- **Semantic Role:** A participant's underlying role with respect to the main verb in the sentence



Connotation

- Also referred to as **affective meaning**
- The aspects of a word's meaning that are related emotions, sentiment, opinions, or evaluations
 - **Valence:** Positivity
 - High: Happy, satisfied
 - Low: Unhappy, annoyed
 - **Arousal:** Intensity of emotion
 - High: Excited, frenzied
 - Low: Relaxed, calm
 - **Dominance:** Degree of control
 - High: Important, controlling
 - Low: Awed, influenced

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24
life	6.68	5.59	5.89

Word vector! (Osgood et al., 1957)



How, then, should we represent the meaning of a word?

- Many, many approaches!
- Two classic strategies:
 - **Bag of words representations:** A word is a string of letters, or an index in a vocabulary list
 - **Logical representation:** A word defines its own meaning (“dog” = DOG)

How, then, should we represent the meaning of a word?

- Many, many approaches!
- Two classic strategies:
 - **Bag of words representations:** A word is a string of letters, or an index in a vocabulary list
 - **Logical representation:** A word defines its own meaning ("dog" = DOG)

Bag of words features implement a simple form of vector semantics.

- Two words with very similar sets of contexts (i.e., similar distributions of neighboring words) are assumed to have very similar meanings
- We represent this context using vectors
- For bag of words:
 - Define a word as a single vector point in an n -dimensional space, where n = vocabulary size
 - The value stored in a dimension n corresponds to the presence of a context word c in the same sample as the target word w

The goal is for the values in these vector representations to correspond with dimensions of meaning.

- Assuming this is the case, we should be able to:
 - Cluster vectors into semantic groups
 - Perform operations that are semantically intuitive



The goal is for
the values in
these vector
representations
to correspond
with dimensions
of meaning.

- Assuming this is the case,
we should be able to:
 - Cluster vectors into semantic groups
 - Perform operations that are semantically intuitive

summary

+

analysis

=

critique

This Week's Topics

Logistic regression
Cross-entropy loss function
Gradient descent optimization

Tuesday

Thursday

Advanced classification details
Vector semantics
TF-IDF

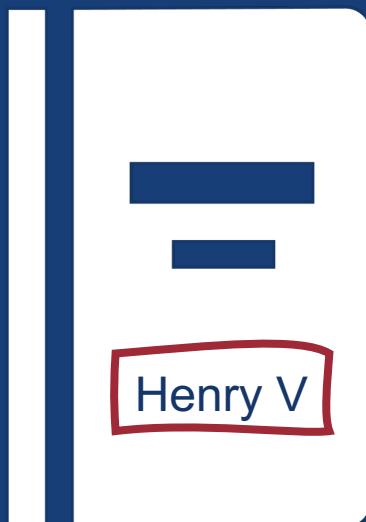
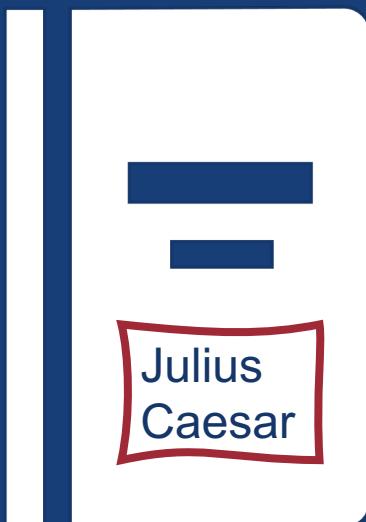


Another approach for learning context using “bag of words” style vectors?

- TF-IDF:
 - Term Frequency * Inverse Document Frequency
 - Meaning of a word is defined by the counts of words in the *same* document, as well as *overall*

TF-IDF originated as a tool for information retrieval.

- Rows: Words in a vocabulary
- Columns: Documents in a corpus

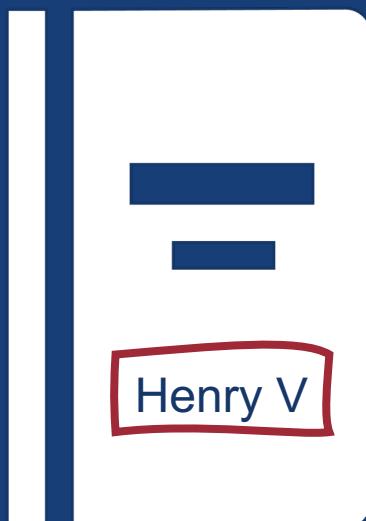
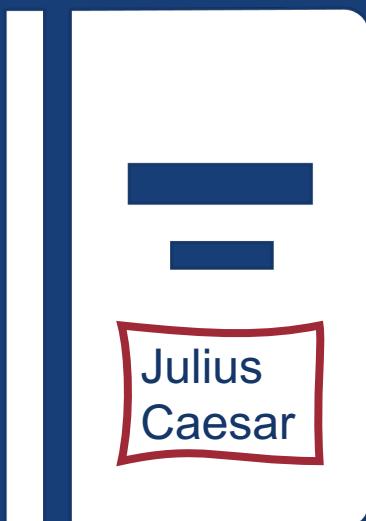


TF-IDF originated as a tool for information retrieval.

- Rows: Words in a vocabulary
- Columns: Documents in a selection

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

“wit” appears 3 times in Henry V

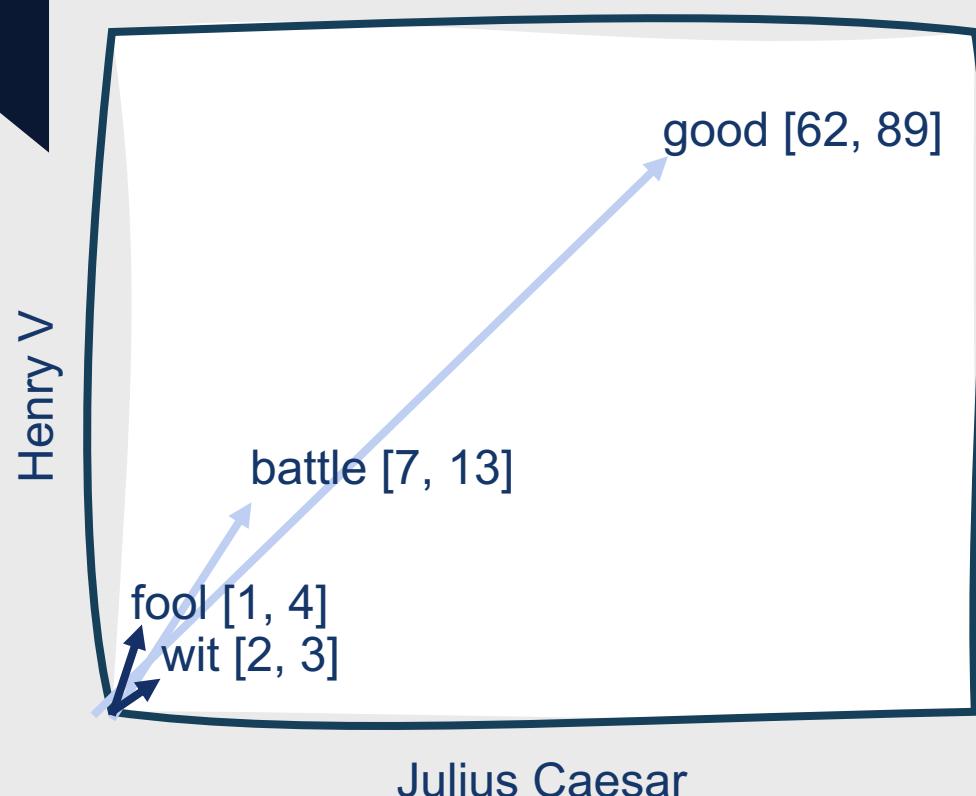


In a term-document matrix, rows can be viewed as word vectors.

- Each dimension corresponds to a document
- Words with **similar vectors** occur in **similar documents**
- This would be one way to define **term frequency** vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

In a term-document matrix, rows can be viewed as word vectors.



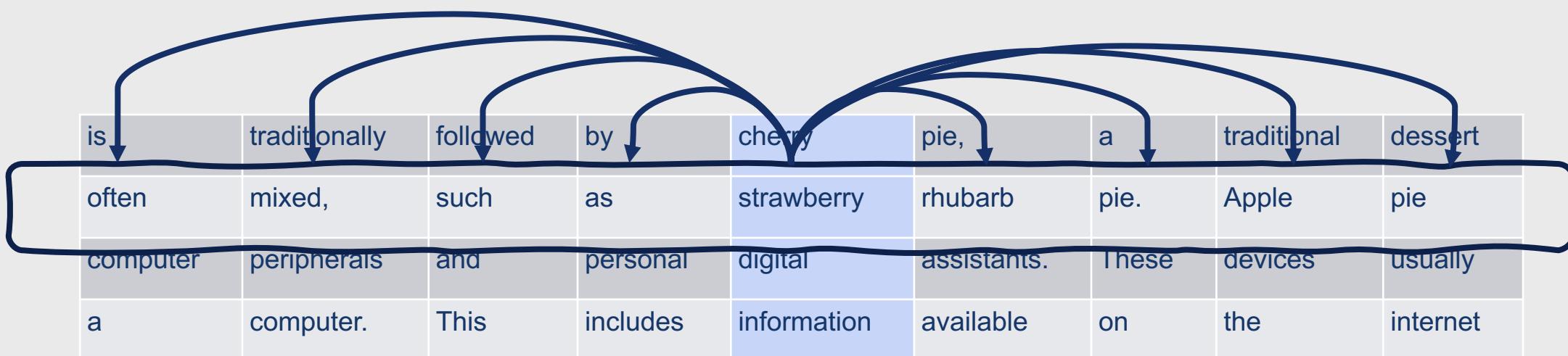
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Different Types of Context

- We can also use **word context** for vector representations
 - Context can be defined however you'd like (e.g., context = entire document, or context = predetermined span length surrounding target word)
- In this type of **word-word matrix**, the columns are also labeled by words
 - Thus, dimensionality is $|V| \times |V|$
 - Each cell records the number of times the row (target) word and the column (context) word co-occur in some context in a training corpus

Example Context Window (Size = 4)

- Take each occurrence of a word (e.g., strawberry)
- Count the context words in the four-word spans before and after it to get a word-word co-occurrence matrix



Sometimes,
raw co-
occurrence
frequency
vectors
don't give
us the most
useful
information.

- Some words co-occur frequently with many words, so won't be very informative
 - *the, it, they*
- We want to know about **words that co-occur frequently with one another, but less frequently across all texts**

TF-IDF is here to save the day!

- **Term Frequency:** The frequency of the word t in the document d
 - $tf_{t,d} = \text{count}(t, d)$
- **Document Frequency:** The number of documents in which the word t occurs

Computing TF-IDF

- **Inverse Document Frequency:** The inverse of document frequency, where N is the total number of documents in the collection
 - $idf_t = \frac{N}{df_t}$
- IDF is higher when the term occurs in fewer documents
 - Document = Whatever is considered an instance or context in your dataset
- It is often useful to perform these computations in log space
 - TF: $\log_{10}(tf_{t,d} + 1) \rightarrow$ Make sure to smooth so you don't try to take the log of 0!
 - IDF: $\log_{10} idf_t$

Computing TF*IDF

- TF-IDF combines TF and IDF
 - $tfidf_{t,d} = tf_{t,d} \times idf_t$



Assume we're looking at a subset of a 37-document corpus of Shakespearean plays....

Example: Computing TF-IDF

- $\text{TF-IDF}(\text{battle}, d_1) = ?$

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Example: Computing TF-IDF

- $\text{TF-IDF}(\text{battle}, d_1) = ?$
- $\text{TF}(\text{battle}, d_1) = 1$

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Example: Computing TF-IDF

- $\text{TF-IDF}(\text{battle}, d_1) = ?$
- $\text{TF}(\text{battle}, d_1) = 1$
- $\text{IDF}(\text{battle}) = N/\text{DF}(\text{battle}) = 37/21 = 1.76$

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58		
wit	20	15		

Overall document frequencies
from our 37 plays

Example: Computing TF-IDF

- $\text{TF-IDF}(\text{battle}, d_1) = ?$
- $\text{TF}(\text{battle}, d_1) = 1$
- $\text{IDF}(\text{battle}) = N/\text{DF}(\text{battle}) = 37/21 = 1.76$
- $\text{TF-IDF}(\text{battle}, d_1) = 1 * 1.76 = 1.76$

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Example: Computing TF-IDF

- $\text{TF-IDF}(\text{battle}, d_1) = ?$
- $\text{TF}(\text{battle}, d_1) = 1$
- $\text{IDF}(\text{battle}) = N/\text{DF}(\text{battle}) = 37/21 = 1.76$
- $\text{TF-IDF}(\text{battle}, d_1) = 1 * 1.76 = 1.76$
- **Alternately, $\text{TF-IDF}(\text{battle}, d_1) = \log_{10}(1 + 1) * \log_{10} 1.76 = 0.074$**

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Example: Computing TF-IDF

- $\text{TF-IDF}(\text{battle}, d_1) = ?$
- $\text{TF}(\text{battle}, d_1) = 1$
- $\text{IDF}(\text{battle}) = \text{N/DF}(\text{battle}) = 37/21 = 1.76$
- $\text{TF-IDF}(\text{battle}, d_1) = 1 * 1.76 = 1.76$
- Alternately, $\text{TF-IDF}(\text{battle}, d_1) = \log_{10}(1 + 1) * \log_{10} 1.76 = 0.074$

	d_1	d_2	d_3	d_4
battle	0.074	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

To convert our entire term frequency matrix to a TF-IDF matrix, we need to repeat this calculation for each element.

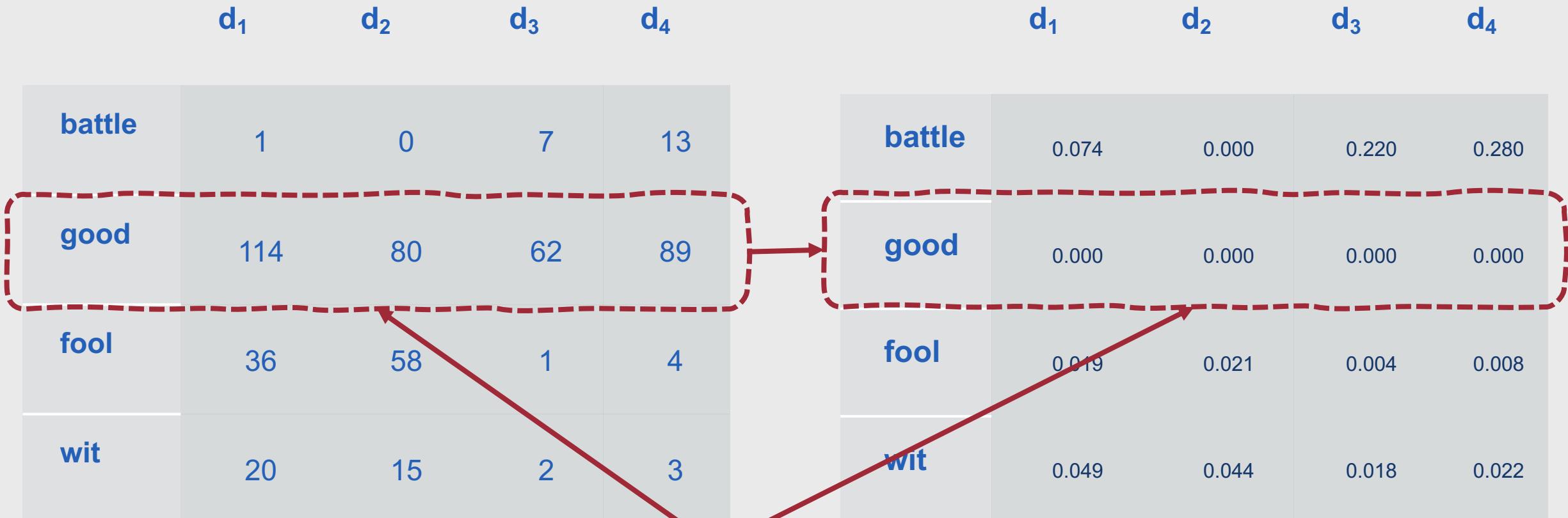
	d ₁	d ₂	d ₃	d ₄
battle	0.074	0.000	0.220	0.280
good	0.000	0.000	0.000	0.000
fool	0.019	0.021	0.004	0.008
wit	0.049	0.044	0.018	0.022

How does the TF-IDF matrix compare to the original term frequency matrix?

	d_1	d_2	d_3	d_4
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

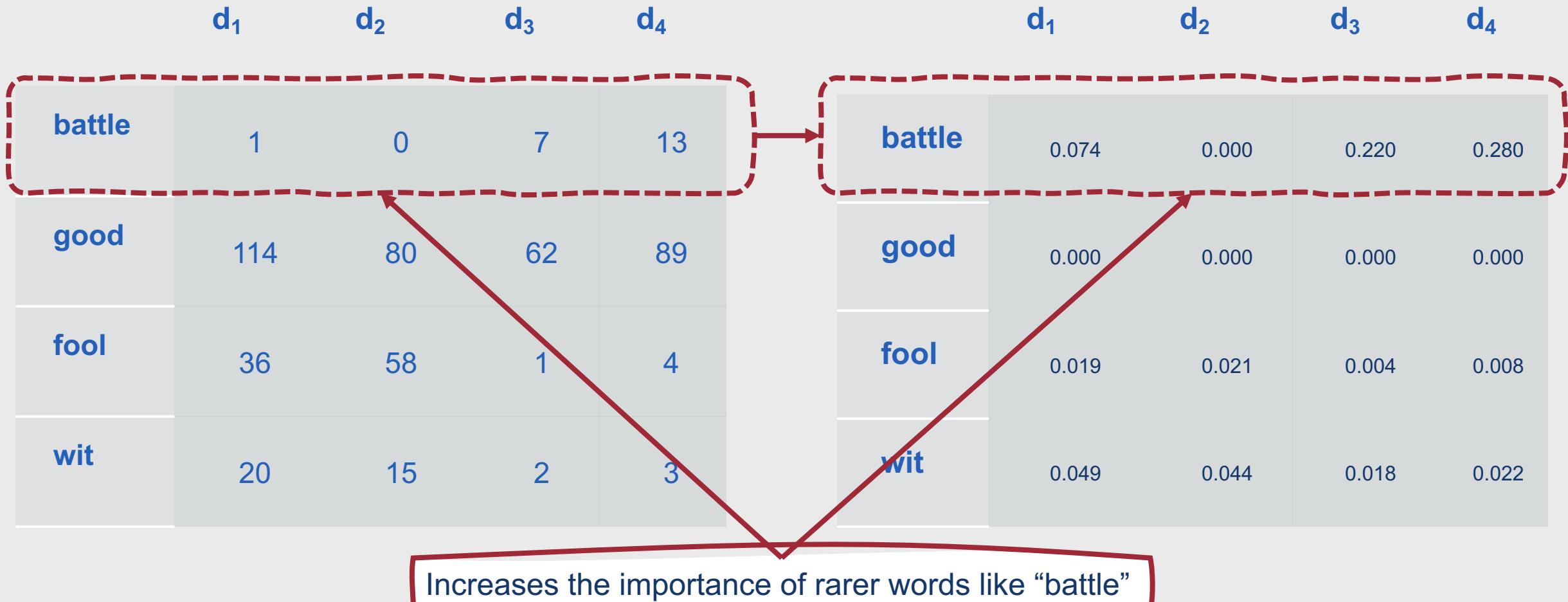
	d_1	d_2	d_3	d_4
battle	0.074	0.000	0.220	0.280
good	0.000	0.000	0.000	0.000
fool	0.019	0.021	0.004	0.008
wit	0.049	0.044	0.018	0.022

How does the TF-IDF matrix compare to the original term frequency matrix?



Occurs in every document ...not important in the overall scheme of things!

How does the TF-IDF matrix compare to the original term frequency matrix?



Note that TF-IDF vectors are sparse.

- Many (usually most) cells have values of 0
- This can make learning difficult

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
battle	0.1	0.0	0.0	0.0	0.2	0.0	0.3
good	0.0	0.0	0.0	0.0	0.0	0.0	0.0
fool	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wit	0.0	0.0	0.0	0.0	0.0	0.0	0.0

We'll learn
about other
word
representation
techniques
soon!

- However, TF-IDF remains a useful starting point for vector space models
- TF-IDF vectors are generally used with feature-based machine learning algorithms
 - Logistic Regression
 - Naïve Bayes

Summary: Introduction to Vector Semantics

- **Representation learning** is the act of building or learning **word vectors** based on **the distributional hypothesis**
- This process seeks to encode the same linguistic phenomena observed in studies of **lexical semantics**
- Bag-of-words representations are one form of word vector, and **TF-IDF representations** are another
- TF-IDF representations combine simple term frequency with **inverse document frequency** to minimize the impact of words that occur more frequently in general