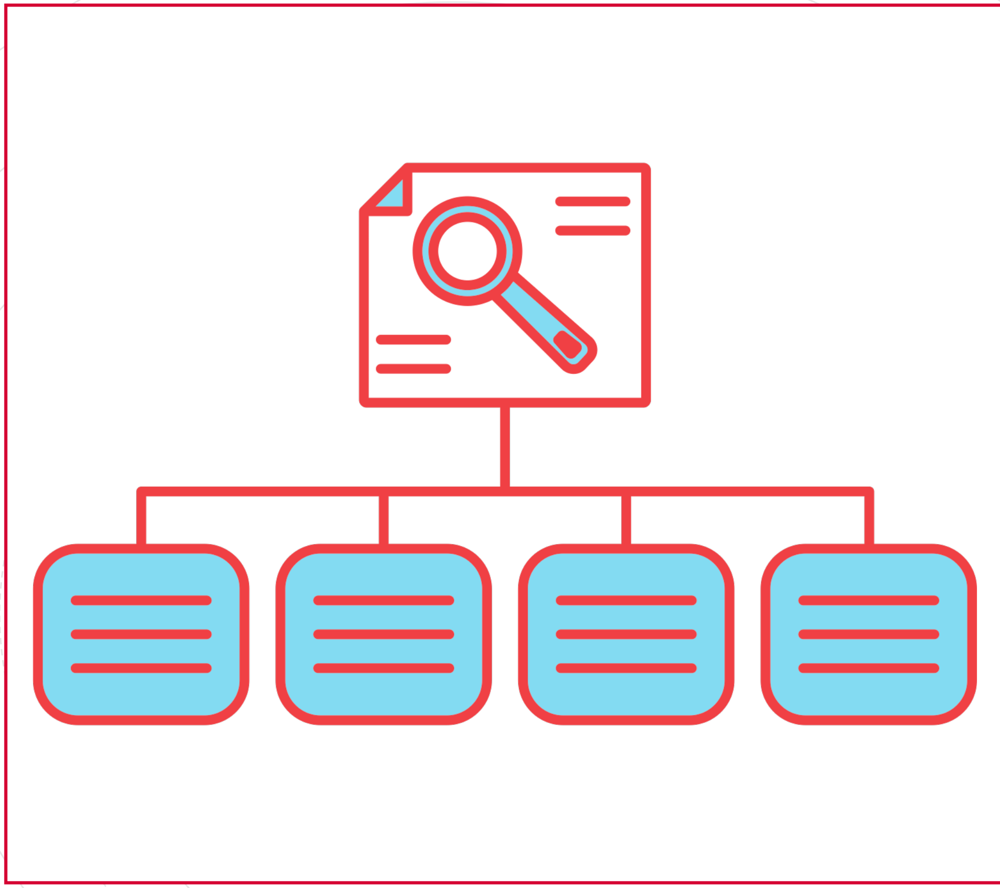


Naïve Bayes and Evaluating Text Classifiers

Natalie Parde

UIC CS 421

This Week's Topics

Text classification
Naïve Bayes
Evaluating text classifiers

Thursday

Tuesday

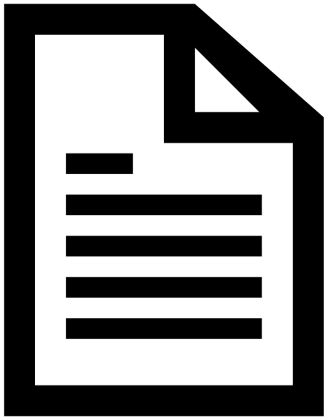
Logistic regression
Cross-entropy loss function
Gradient descent optimization
Other classification details

What is text classification?

The process of deciding the **category** in which an **instance** belongs

- **Instance:** A document, sentence, word, transcript, or other individual language sample

Fundamental to many NLP tasks



Common Applications of Text Categorization

- Spam detection

Dear Dr. Parde Natalie,

Journals of [redacted] are devoted to the principles and core ethics of Open Access. Our goal is to create an egalitarian platform to enable unrestricted knowledge exchange among researchers, experts, and curious minds alike. We are breaking away from old traditions to open the doors wide open to people from all corners of the world. First and foremost, we respect the author's right of ownership to the articles they create.

Our publications provides a global platform and a targeted source for publishing original research. Do you have an article/ebook ready for submission? We are accepting submissions ***with 139 USD as pocessing charges for the Open Access Week 2019***. Please feel free to revert with any further questions about the special themes.

Looking forward!

[redacted]
Editorial Coordinator
[redacted]

Spam

Not Spam

Common Applications of Text Categorization

- Spam detection
- Authorship attribution

"What can be the meaning of that emphatic exclamation?" cried he. "Do you consider the forms of introduction, and the stress that is laid on them, as nonsense? I cannot quite agree with you _there_. What say you, Mary? For you are a young lady of deep reflection, I know, and read great books and make extracts."

Mary wished to say something sensible, but knew not how.

"While Mary is adjusting her ideas," he continued, "let us return to Mr. Bingley."

"I am sick of Mr. Bingley," cried his wife.

"The world is full of obvious things which nobody by any chance ever observes. Where do you think that I have been?"

"A fixture also."

"On the contrary, I have been to Devonshire."

"In spirit?"

Voltaire

Sir Arthur Conan Doyle

Jane Austen

Common Applications of Text Categorization

- Spam detection
- Authorship attribution
- Sentiment analysis

Natalie's poem about Halloween was really dreadful. The word "Halloween" doesn't even rhyme with "trick or treat!" She should stick to writing NLP programs.

Natalie's poem about Halloween was a true delight! The way she rhymed "Halloween" with "trick or treat" was artful and unexpected. I can't wait to read what she writes next!

Natalie wrote a poem about Halloween. She wrote it as if the words "Halloween" and "trick or treat" rhyme with one another. It was her first poem.

Positive

Negative

Neutral

Common Applications of Text Categorization

- Spam detection
- Authorship attribution
- Sentiment analysis
- Domain identification

"What can be the meaning of that emphatic exclamation?" cried he. "Do you consider the forms of introduction, and the stress that is laid on them, as nonsense? I cannot quite agree with you _there_. What say you, Mary? For you are a young lady of deep reflection, I know, and read great books and make extracts."

Mary wished to say something sensible, but knew not how.

"While Mary is adjusting her ideas," he continued, "let us retu

Mr. Bingley."

"I am sick of Mr. Bingley," cried his wif

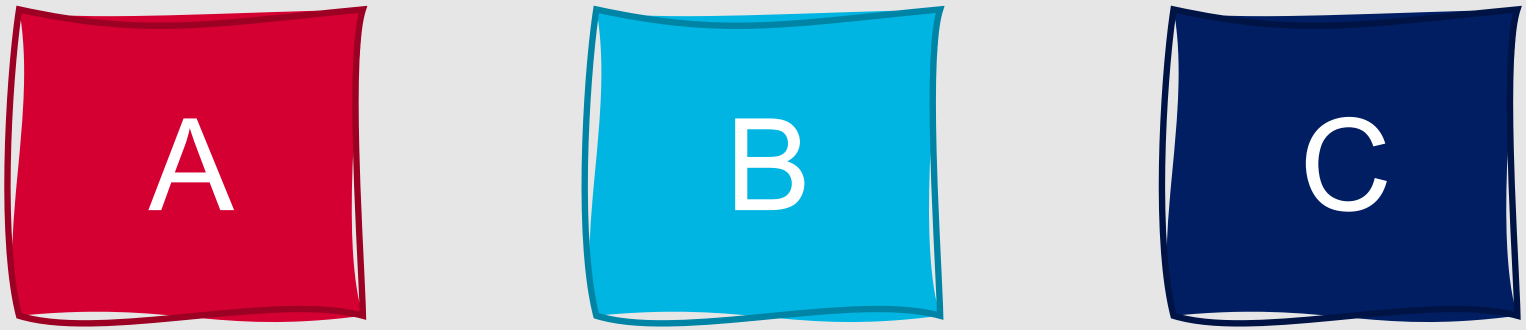
The model takes two inputs: the tokenized interview, and the corresponding POS tag list. Word embeddings for the interview text tokens are computed using pre-trained 300 dimensional GloVe embeddings trained on the Wikipedia 2014 and Gigaword 5 dataset (Pennington et al., 2014). The POS tag for each word is represented as a one-hot encoded vector. The word embeddings and POS vectors are input to two different CNNs utilizing the same architecture, and the output of the two CNNs is then flattened and given as input to a bidirectional LSTM with an attention mechanism.

Fiction

Academic

Classification

- Goal:
  - Take a single **observation**
  - Extract some useful **features**
  - Classify the observation into one of a set of discrete classes based on those features

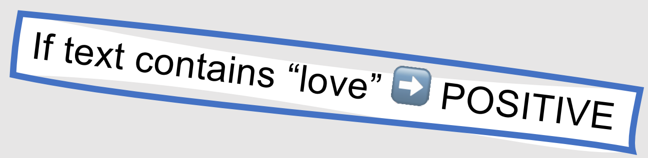


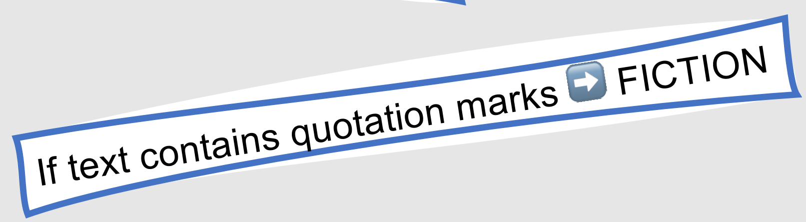
How is classification performed?

- Rule-based methods
- Statistical methods
  - Including feature-based methods and deep learning methods

Rule-Based Classification Methods

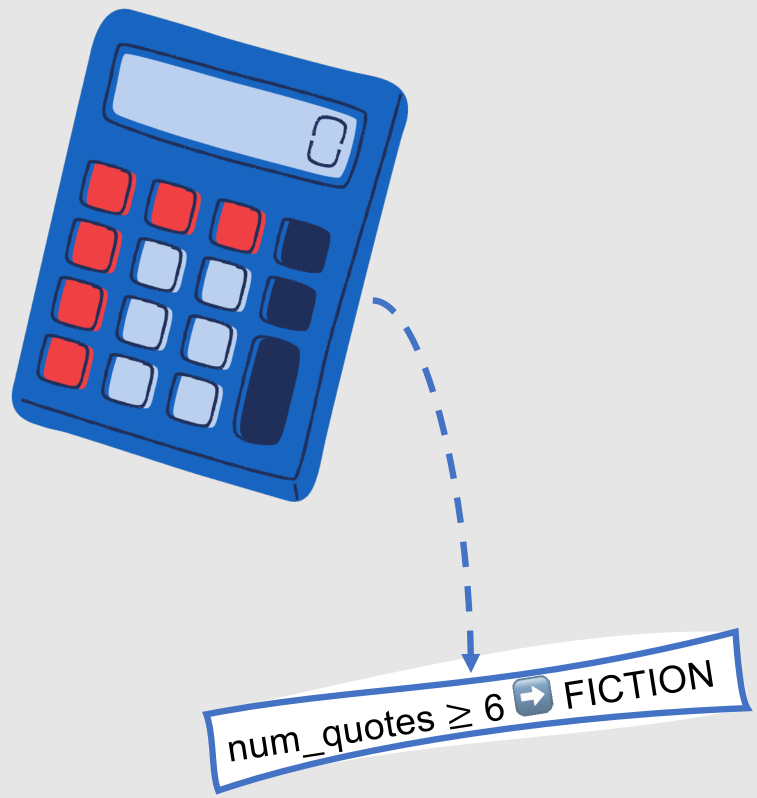
- **Manually create a set of rules** based on expected differences among features from different classes
- Use that information to classify test data





Statistical Classification Methods

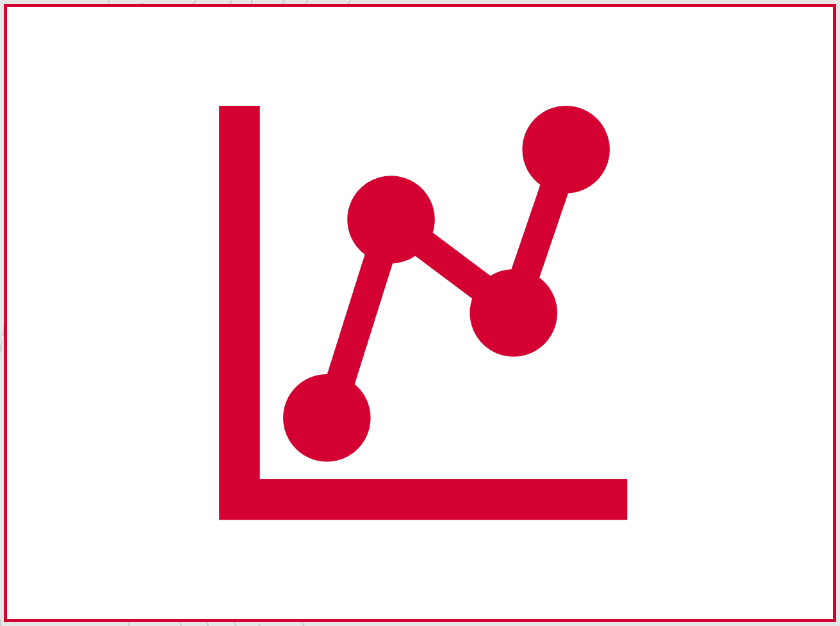
- Automatically **learn which characteristics best distinguish different classes** from one another based on a collection of training data
- Use that information to classify test data



Is rule-based or statistical classification better?

- In modern computing environments (i.e., scenarios with plentiful data), **statistical classification is generally a better choice**
- If data is really limited, rule-based methods may work better

Language is dynamic.



- Word uses can change over time, and so can data
  - No *cap*
  - *Covid-19*
- With rule-based methods, we have to write new rules to accommodate changes in language
  - We also might miss some changes!
- Statistical methods can be automatically retrained when new data is available

Types of Statistical Classification Techniques

Supervised learning: Statistical classification *with* a labeled training set

Unsupervised learning: Statistical classification *without* a labeled training set

Formal Definition: Supervised Learning

- Take an input *x* from a set of inputs $x \in X$
- Consider a fixed set of output classes $y \in Y$, where $Y = \{y_1, y_2, \ldots, y_M\}$
  - In text classification, we may refer to *x* as *d* (for "document") and *y* as *c* (for "class")
- We have a training set of *N* documents, each of which have been manually labeled with a class: $\{(d_1, c_1), \ldots, (d_N, c_N)\}$
- Goal: Learn a classifier that is capable of mapping from a new document *d* to its correct class $c \in C$ (equivalently, learning to predict the correct class $y \in Y$ for an input $x \in X$)

Types of Supervised Probabilistic Classification Models

- Naïve Bayes
- Logistic regression
- Support vector machine
- K-nearest neighbors
- Multilayer perceptrons (neural networks)
- …and many more!

These classification models can be further subdivided into groups.

- **Generative classifiers** build models of how classes could generate input data
  - Given an observation, they return the class most likely to have generated it
- **Discriminative classifiers** learn which features from the input are most useful to discriminate between different possible classes
  - Given an observation, they return the best match based on these weighted features

This Week's Topics

Tuesday

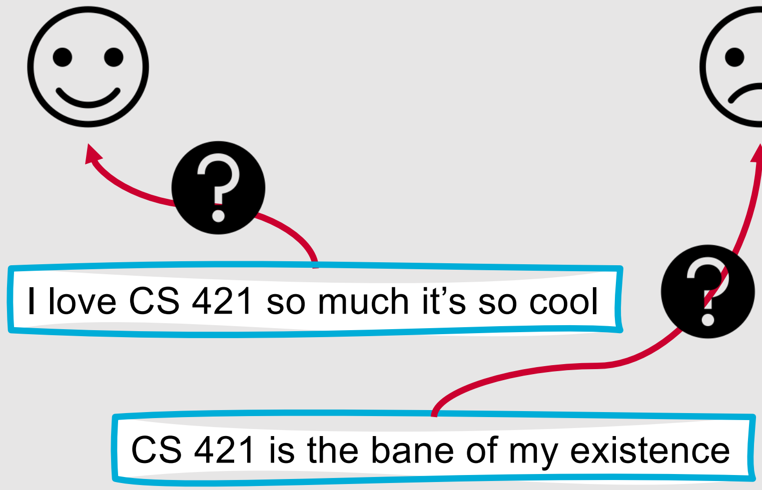
- Text classification
- Naïve Bayes
- Evaluating text classifiers

Thursday

- Logistic regression
- Cross-entropy loss function
- Gradient descent optimization
- Other classification details

One probabilistic, generative classifier?

- Naïve Bayes: A **probabilistic classifier** that learns to **predict labels** for new documents



Why is it “Naïve” Bayes?

- Naïve Bayes classifiers make a naïve assumption that features don’t impact each other and instead **are all independent from one another**
- Is this really the case?
  - No! As we’ve seen with language models and HMMs, words are dependent on their contexts
  - However, Naïve Bayes classifiers still perform reasonably well despite this assumption

Types of Naïve Bayes Classifiers

Gaussian Naïve Bayes: Assumes the outcomes for the input data are normally distributed along a continuum

Multinomial Naïve Bayes: Assumes the outcomes for the input data follow a multinomial distribution (there is a discrete set of possible outcomes)

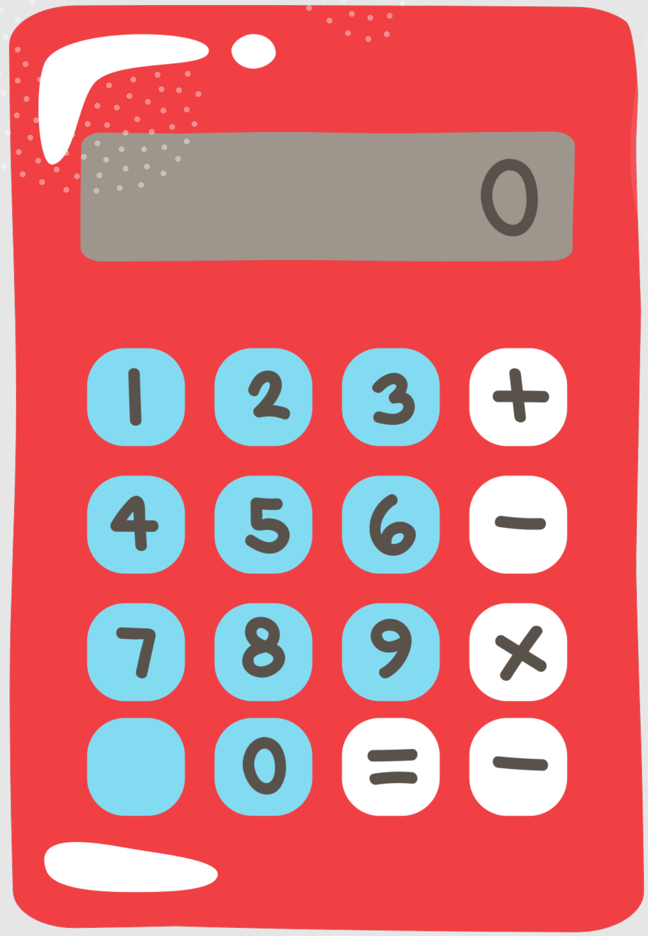
Binomial Naïve Bayes: Assumes the outcomes for the input data follow a binomial distribution (there are two possible outcomes)

How does naïve Bayes work?

- For a document *d*, out of all classes $c \in C$ the classifier returns the class *c*' which has the maximum **posterior probability**, given the document
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c|d)$

Posterior probabilities are computed using Bayesian inference.

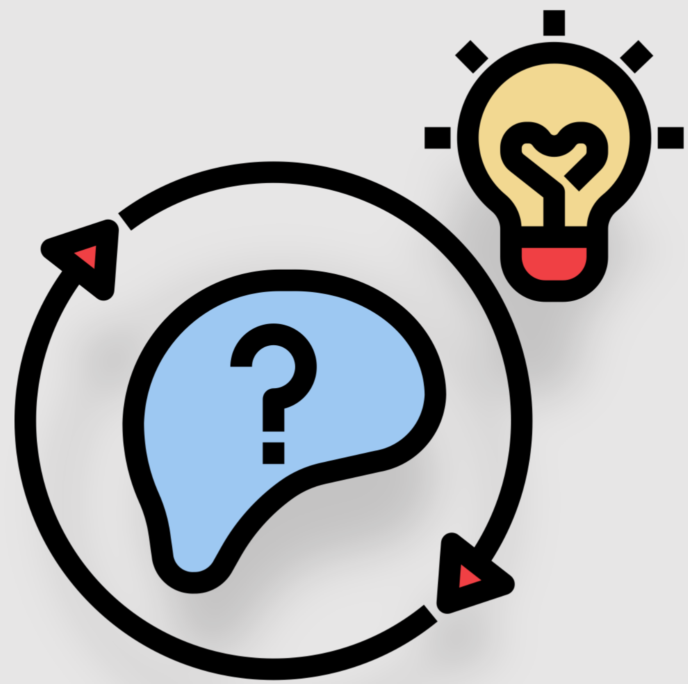
- Bayesian inference uses **Bayes' rule** to transform probabilities like those shown previously into other probabilities that are easier or more convenient to calculate
- Bayes' rule:
  - $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$



Applying Bayesian inference in Naïve Bayes

- If we take Bayes' rule:
  - $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$
- And substitute it into our previous equation:
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c|d)$
- We get the following:
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c|d)$

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



We can simplify this….

- Drop the denominator P(d)
  - We'll be computing $\frac{P(d|c)P(c)}{P(d)}$ for each class, but P(d) doesn't change for each class
    - We're always asking about the most likely class for the same document d
- Thus:
  - $c' = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) = \underset{c \in C}{\operatorname{argmax}}\, P(d|c)P(c)$

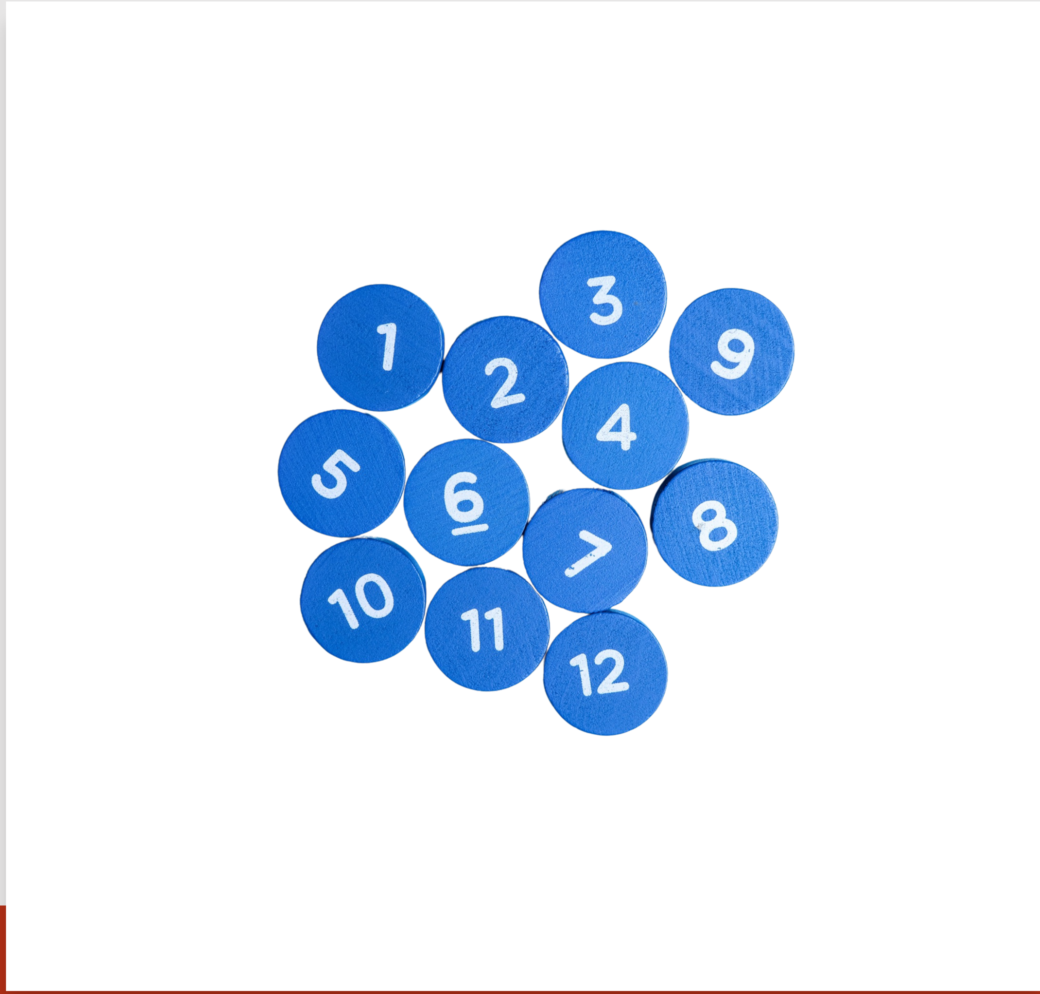
So, the most probable class *c*' given some document *d* is the class that has the highest product of two probabilities.

- **Prior probability** of the class P(*c*)
- **Likelihood** of the document P(*d*|*c*)

$$\boxed{c' = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(d|c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}}}$$

To find these probabilities….

- We need to represent our text sample using one or more numbers
- These numbers can represent different **features** of the data



Example Feature Representation

- Represent each document as a **bag of words**
  - Unordered set of words and their frequencies
- Decide how likely it is that a document belongs to a class based on its distribution of **word frequencies**

Ankit's poem about Thanksgiving rivaled Natalie's notorious Halloween poem.

's	2
poem	2
Ankit	1
about	1
Thanksgiving	1
rivaled	1
Natalie	1
notorious	1
Halloween	1

More formally, this means that….

- Bags of words are sets of features {f_1, f_2, …, f_n}, where each feature f corresponds to the frequency of one of the words in the vocabulary
- Therefore:
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c) = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(f_1, f_2, \ldots, f_n|c)}_{\text{likelihood}} \underbrace{P(c)}_{\text{prior}}$
- The Naïve Bayes assumption means that we can "naïvely" multiply our probabilities for each feature together, since they're assumed to be independent
- Therefore:
  - $P(f_1, f_2, \ldots, f_n|c) = P(f_1|c) * P(f_2|c) * \cdots * P(f_n|c)$

Putting this all together….

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

$$= \underset{c \in C}{\operatorname{argmax}} P(f_1, f_2, \ldots, f_n|c)\, P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

- To avoid underflow (generating numbers that are too tiny to be adequately represented) and increase speed, we can also do these computations in log space:
  - $c' = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in N} \log P(f_i|c)$

When viewed in log space, we can see that this is a linear classifier.

- A linear classifier predicts classes as a **linear function** of the input features
  - $c' = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in T} \log P(w_i|c)$
- Some linear classifiers:
  - Naïve Bayes
  - Logistic Regression

How do we train a Naïve Bayes classifier?

- We need to learn P(c) and P($f_i|c$) based on available data
- To compute P(c), we figure out what percentage of the instances in our training set are in class c
  - Let N_c be the number of instances in our training data with class c
  - Let N_{doc} be the total number of instances, or documents
  - $P(c)' = \frac{N_c}{N_{doc}}$
- To compute P($f_i|c$)….
  - **Maximum likelihood estimates!**

Maximum Likelihood for Naïve Bayes

- To compute $P(f_i|c)$, find the fraction of times f_i appears among all documents of class *c*
  - $P(f_i|c)' = \frac{count(f_i,c)}{\sum_{f \in V} count(f,c)}$
    - Note: Since we're assuming features are words in a bag-of-words model, *V* is the set of all included vocabulary words across all classes (not just the word types in class *c*)

To avoid having a single zero probability “zero out” the entire product, we can apply smoothing techniques.

- Simple, common solution: Laplace (add-one) smoothing
  - $P(f_i|c)'$

$$= \frac{count(f_i,c)+1}{\sum_{f\in V}(count(f,c)+1)}$$

$$= \frac{\boldsymbol{count(f_i,c)+1}}{\boldsymbol{\sum_{f\in V}(count(f,c))+|V|}}$$

Other scenarios to address:

- **Unknown words**
  - Solution: Ignore words that didn't exist in the training data (remove from test document + do not compute any probabilities for them)
- **Stopwords**
  - Ignore very frequent words like *a* and *the* in many cases using an automatically or manually defined stopword list

Example: Naïve Bayes

Natalie was soooo thrilled that Ankit had a famous new poem.

She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.

Ankit was happy that his poem about Thanksgiving was so successful.

He congratulated Natalie for getting #2 on the bestseller list.

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Natalie was soooo thrilled that Ankit had a famous new poem. **Sarcastic**

She was totally 100% not annoyed that it had surpassed her poem on the bestseller list. **Sarcastic**

Ankit was happy that his poem about Thanksgiving was so successful. **Not Sarcastic**

He congratulated Natalie for getting #2 on the bestseller list. **Not Sarcastic**

Natalie told Ankit she was soooo totally happy for him. **?**

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What is the prior probability for each class?
  - $P(c)' = \frac{N_c}{N_{doc}}$

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What is the prior probability for each class?
  - $P(c)' = \frac{N_c}{N_{doc}}$
- P(Sarcastic) = 2/4 = 0.5
- P(Not Sarcastic) = 2/4 = 0.5

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What is the prior probability for each class?
  - $P(c)' = \frac{N_c}{N_{doc}}$
- P(Sarcastic) = 2/4 = 0.5
- P(Not Sarcastic) = 2/4 = 0.5
- Note: This means we have a **balanced training set**
  - Balanced: An equal number of samples for each class

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- Taking a closer look at our data, let's remove:
  - Stop words
  - Unknown words

Natalie told Ankit she was soooo totally happy for him.

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

- Taking a closer look at our data, let's remove:
  - **Stop words**
  - Unknown words

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- Taking a closer look at our test instance, let's also remove:
  - Stop words
  - **Unknown words**

Natalie told Ankit she was soooo totally happy for him.

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)}{\sum_{w \in V} count(w,c)}$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$

Make sure to use smoothing!

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("Ankit"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Ankit"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("Ankit"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Ankit"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("soooo"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("Ankit"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Ankit"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("soooo"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$
  - P("totally"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("totally"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- What are the likelihoods from the training set for the remaining words in the test instance?
  - $P(w_i|c)' = \frac{count(w_i,c)+1}{(\sum_{w \in V} count(w,c))+|V|}$
  - P("Natalie"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Natalie"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("Ankit"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("Ankit"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$
  - P("soooo"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("soooo"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$
  - P("totally"|Sarcastic) = $\frac{1+1}{15+21} = 0.056$
  - P("totally"|Not Sarcastic) = $\frac{0+1}{12+21} = 0.030$
  - P("happy"|Sarcastic) = $\frac{0+1}{15+21} = 0.028$
  - P("happy"|Not Sarcastic) = $\frac{1+1}{12+21} = 0.061$

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- Given all of this information, how should we classify the test sentence?
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.056	0.061
Ankit	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- Given all of this information, how should we classify the test sentence *s*?
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$
  - P(Sarcastic)*P(*s*|Sarcastic) = 0.5 * 0.056 * 0.056 * 0.056 * 0.056 * 0.028 = 1.377 * 10^{-7}

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.056	0.061
Ankit	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- Given all of this information, how should we classify the test sentence *s*?
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$
  - P(Sarcastic)*P(*s*|Sarcastic) = 0.5 * 0.056 * 0.056 * 0.056 * 0.056 * 0.028 = 1.377 * 10^{-7}
  - P(Not Sarcastic)*P(*s*|Not Sarcastic) = 0.5 * 0.061 * 0.061 * 0.030 * 0.030 * 0.061 = 1.021 * 10^{-7}

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.056	0.061
Ankit	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Example: Naïve Bayes

Training	
Document	**Class**
Natalie was soooo thrilled that Ankit had a famous new poem.	Sarcastic
She was totally 100% not annoyed that it had surpassed her poem on the bestseller list.	Sarcastic
Ankit was happy that his poem about Thanksgiving was so successful.	Not Sarcastic
He congratulated Natalie for getting #2 on the bestseller list.	Not Sarcastic
Test	
Document	**Class**
Natalie told Ankit she was soooo totally happy for him.	?

- Given all of this information, how should we classify the test sentence *s*?
  - $c' = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in T} P(w_i|c)$
  - **P(Sarcastic)*P(*s*|Sarcastic) = 0.5 * 0.056 * 0.056 * 0.056 * 0.056 * 0.028 = 1.377 * 10^{-7}**
  - P(Not Sarcastic)*P(*s*|Not Sarcastic) = 0.5 * 0.061 * 0.061 * 0.030 * 0.030 * 0.061 = 1.021 * 10^{-7}

Word	P(Word|Sarcastic)	P(Word|Not Sarcastic)
Natalie	0.056	0.061
Ankit	0.056	0.061
soooo	0.056	0.030
totally	0.056	0.030
happy	0.028	0.061

P(Sarcastic) = 0.5
P(Not Sarcastic) = 0.5

Natalie told Ankit she was soooo totally happy for him.

Sarcastic

Optimizing for Specific Tasks

- There are a variety of task-specific ways to improve performance with this model
- You may want to specifically encode:
  - Whether a feature exists in the data (rather than how many times)
  - Whether specific types of words (e.g., **negation**) are present

We can derive alternate or additional features (not word counts) from external **lexicons**.

- For example, add a feature that is counted whenever a word from a specific lexicon occurs
- **Lexicons** generally contain annotated characteristics (e.g., sentiment labels) for a list of words
- For sentiment analysis:
  - Linguistic Inquiry and Word Count (http://liwc.wpengine.com/)
  - Opinion Lexicon (https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon)
  - MPQA Subjectivity Lexicon (https://mpqa.cs.pitt.edu/lexicons/subj_lexicon/)

Whether this works well may depend on data sparsity.

Large dataset:

- Using many features will work better than just using a few binary features (allows for the classifier to learn more complex ways to discriminate between classes)

Small dataset:

- Using a smaller number of more general features may work better (allows for the classifier to learn meaningful differences, rather than making predictions based on one or two occurrences of a given feature)

This Week's Topics

Tuesday

- Text classification
- Naïve Bayes
- Evaluating text classifiers

Thursday

- Logistic regression
- Cross-entropy loss function
- Gradient descent optimization
- Other classification details

We've learned a bit about text classification now….

How can we measure the performance of our models?

Gold Labels

- Before determining anything, we need some sort of basis upon which to make our comparisons
  - *Is "Sarcastic" the correct label for "Natalie told Ankit she was soooo totally happy for him." ?*
- We can acquire **gold standard labels** from human annotators

Does it matter who our annotators are?

- Depends on the task
- For complex tasks, you may want to recruit experts
  - Rating translation quality
  - Labeling pedagogical strategies in teacher-student interactions
- For simpler tasks, you can probably recruit non-experts
  - Deciding whether text is sarcastic or non-sarcastic
  - Deciding whether a specified event takes place before or after a second event

Confusion Matrices

- Once we have our gold standard labels (either from an existing dataset, or after collecting our own), we can begin comparing **predicted** and **actual** labels
- To do this, we can create a **contingency table** or **confusion matrix**

Confusion Matrices

- In a confusion matrix, each cell indicates a set of possible outcomes
- These outcomes are generally referred to as:
  - **True positives**
    - Predicted true and actually true
  - **False positives**
    - Predicted true and actually false
  - **True negatives**
    - Predicted false and actually false
  - **False negatives**
    - Predicted false and actually true

	Actual	
Predicted	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

We can compute a variety of metrics using confusion matrices.

- Precision
- Recall
- F-Measure
- Accuracy

	Actual	
Predicted	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

Accuracy

- **Accuracy:** The percentage of all observations that the system labels correctly
- Accuracy = $\frac{\mathrm{tp}+\mathrm{tn}}{\mathrm{tp}+\mathrm{fp}+\mathrm{tn}+\mathrm{fn}}$

Why not just use accuracy and be done with it?

- This metric can be unreliable when dealing with unbalanced datasets!
    - Imagine that we have 999,900 non-sarcastic sentences, and 100 sarcastic sentences
    - Our classifier might decide to just predict “non-sarcastic” every time to maximize its expected accuracy
        - 999900/1000000 = 99.99% accuracy
    - However, such a classifier would be useless …it would never tell us when a sentence *is* sarcastic

What are some more useful alternative metrics?

- Precision
- Recall
- F-Measure

Actual

Predicted		
	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

Precision

- **Precision:** Of the instances that the system predicted to be positive, what percentage actually are?
- Precision = $\frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}}$

Actual

Predicted

True Positive (TP)	False Positive (FP)
False Negative (FN)	True Negative (TN)

Recall

- **Recall:** Of the instances that actually are positive, what percentage did the system predict to be?
- Recall = $\frac{tp}{tp+fn}$

Precision and recall both emphasize a specific class of interest.

- For example:
  - **Sarcastic** or Non-Sarcastic
  - Positive or **Negative**
- In our problematic example case, precision and recall for the positive (sarcastic) case would both be 0
  - Precision = 0/(0+0) = 0
  - Recall = 0/(0+100) = 0

Predicted \ Actual		
	TP: 0	FP: 0
	FN: 100	TN: 999,900

Which is more useful: Precision or recall?

- Depends on the task!
- If it's more important to maximize the chances that all predicted true values really are true, at the expense of predicting some of the true values as false, focus on precision
- If it's more important to maximize the chances that all true values are predicted to be true, at the expense of predicting some false values to be true as well, focus on recall

What if both are important?

- **F-measure** combines aspects of both **precision** and **recall** by computing their weighted harmonic mean
  - $F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$
- The β parameter weights the importance of precision and recall, depending on the needs of the application
  - $\beta > 1$ means that recall is more important
  - $\beta < 1$ means that precision is more important
  - $\beta = 1$ means that the two are equally important

F-Measure

- Most commonly, researchers set β = 1 to weight precision and recall equally
- In this case, the metric is generally referred to as F_1
  - $F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R}$
  - Note: With this equation, the lower of the two numbers will factor slightly more heavily into the final score

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	
Oh yay more things to grade!!!	Sarcastic	
Oh yay my new subscription box arrived!!!	Not Sarcastic	
Where is the closest coffee shop?	Not Sarcastic	
I just love large committee meetings.	Sarcastic	

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted		
	TP: ?	FP: ?
	FN: ?	TN: ?

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted

TP: 1	FP: ?
FN: ?	TN: ?

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted

TP: 1	FP: 1
FN: ?	TN: ?

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted

TP: 1	FP: 1
FN: 3	TN: ?

Positive Class: Sarcastic

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Positive Class: Sarcastic

Actual

Predicted

TP: 1	FP: 1
FN: 3	TN: 2

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted	TP: 1	FP: 1
	FN: 3	TN: 2

Positive Class: Sarcastic

Precision = $\frac{\text{tp}}{\text{tp+fp}} = \frac{1}{1+1}$ = 0.5

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted

TP: 1	FP: 1
FN: 3	TN: 2

Positive Class: Sarcastic

Precision = $\frac{\text{tp}}{\text{tp+fp}} = \frac{1}{1+1}$ = 0.5

Recall = $\frac{\text{tp}}{\text{tp+fn}} = \frac{1}{1+3}$ = 0.25

Example: Precision, Recall, and F_1

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted		
	TP: 1	FP: 1
	FN: 3	TN: 2

Positive Class: Sarcastic

Precision = 0.5

Recall = 0.25

$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = \frac{2*0.5*0.25}{0.5+0.25} = 0.333$

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted		
	TP: ?	FP: ?
	FN: ?	TN: ?

Positive Class: Not Sarcastic

Precision = ?

Recall = ?

$F_1 = \frac{(1^2+1)PR}{1^2 P+R} = \frac{2PR}{P+R}$ = ?

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted

TP: 2	FP: 3
FN: 1	TN: 1

Positive Class: Not Sarcastic

Precision = ?

Recall = ?

$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R}$ = ?

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted

TP: 2	FP: 3
FN: 1	TN: 1

Positive Class: Not Sarcastic

Precision = 0.4

Recall = ?

$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R}$ = ?

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

Actual

Predicted		
	TP: 2	FP: 3
	FN: 1	TN: 1

Positive Class: Not Sarcastic

Precision = 0.4

Recall = 0.667

$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R}$ = ?

Example: Same, but what if the positive class is Not Sarcastic?

Instance	Actual Label	Predicted Label
I was absolutely thrilled that my smoke alarm broke.	Sarcastic	Not Sarcastic
I was absolutely thrilled that my paper was accepted!	Not Sarcastic	Not Sarcastic
I am soooo sad that tomorrow's 8 a.m. meeting is cancelled.	Sarcastic	Sarcastic
Oh yay more things to grade!!!	Sarcastic	Not Sarcastic
Oh yay my new subscription box arrived!!!	Not Sarcastic	Sarcastic
Where is the closest coffee shop?	Not Sarcastic	Not Sarcastic
I just love large committee meetings.	Sarcastic	Not Sarcastic

	Actual	
Predicted	TP: 2	FP: 3
	FN: 1	TN: 1

Positive Class: Not Sarcastic

Precision = 0.4

Recall = 0.667

$$F_1 = \frac{(1^2+1)PR}{1^2P+R} = \frac{2PR}{P+R} = \frac{2*0.4*0.667}{0.4+0.667} = 0.50009$$

Summary: Naïve Bayes Essentials

- **Naïve Bayes** is a **probabilistic, supervised classification algorithm**
- When making predictions, a classifier takes a test observation, extracts a set of features from it, and assigns a label to the observation based on similarities between its feature values and those of observations in the training dataset
- **Multinomial Naïve Bayes** assumes that there is a discrete set of possible classes for the data
- Naïve Bayes is "naïve" because it makes the simplifying assumption that **all features are independent of one another**
- Naïve Bayes classifiers generally use **bag of words** features, but may use other features (e.g., those from external **lexicons**) depending on the task
- Classification model performance is determined by comparing the model's predictions to a set of **gold standard labels**
- The similarities and differences between predicted and actual labels can be summarized in a **contingency table** containing **true positives**, **false positives**, **true negatives**, and **false negatives**
- Four common metrics can be computed from values in this table
  - **Precision:** Of the observations predicted to be true, how many actually are?
  - **Recall:** Of the observations that are true, how many were predicted to be?
  - **F-Measure:** What is the harmonic mean between precision and recall?
  - **Accuracy:** What percentage of observations did the model label correctly?

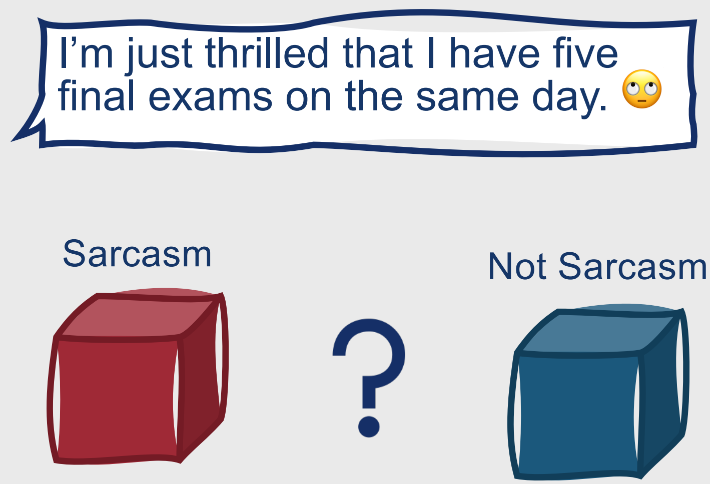
Logistic Regression

Natalie Parde

UIC CS 421

Looking back at 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



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….

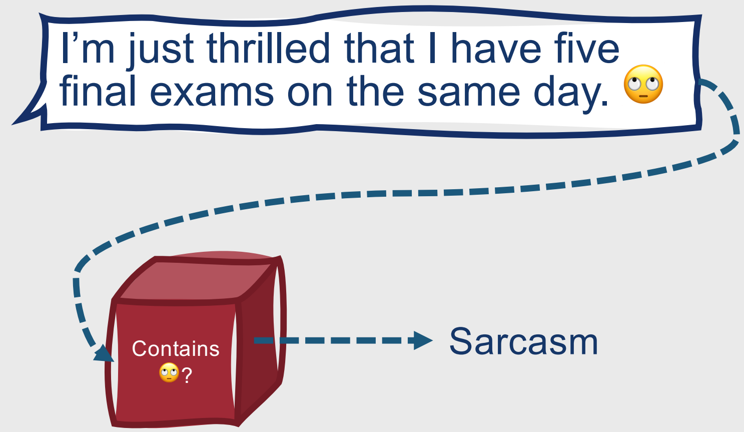
- Recall the definition of naïve Bayes:
  - $\hat{c} = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(d|c)}_{\text{Likelihood}} \underbrace{P(c)}_{\text{Prior}}$

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

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

Discriminative Classifiers

- Goal: Learn to distinguish between classes
  - No need to learn that much about them individually
- Bases the classification decision on the distinguishing feature(s) between classes



More formally….

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

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

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

This Week's Topics

Tuesday

- Text classification
- Naïve Bayes
- Evaluating text classifiers

Thursday

- Logistic regression
- Cross-entropy loss function
- Gradient descent optimization
- Other classification details

This Week's Topics

Tuesday

- Text classification
- Naïve Bayes
- Evaluating text classifiers

Thursday

- Logistic regression
- Cross-entropy loss function
- Gradient descent optimization
- Other classification details

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 $[x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)}]$ 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

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:
  - $z = \boldsymbol{w} \cdot \boldsymbol{x} + b$

Vector of all weights

Bias term

Vector of all features for *x*

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

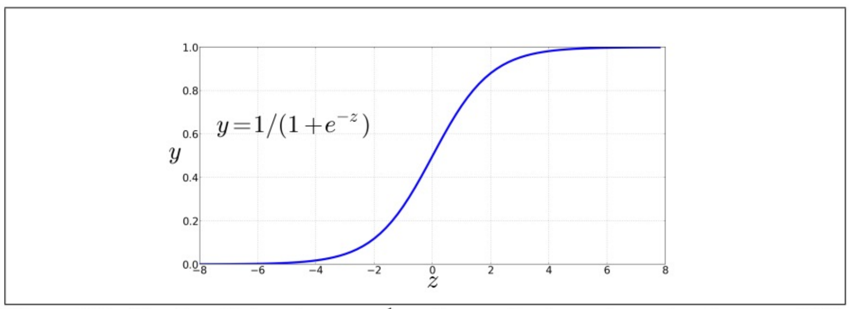


Figure 5.1 The sigmoid function $y = \frac{1}{1+e^{-z}}$ takes a real value and maps it to the range [0, 1]. It is nearly l

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!

- To simplify classification in true binary logistic regression, you can just assume:
  - $P(y=1)= \sigma(z)$
  - $P(y=0)=1-\sigma(z)$
- More generally though, you'll need to compute separate probabilities for each class based on feature weights associated with that 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
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

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

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

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

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

Tuesday

- Text classification
- Naïve Bayes
- Evaluating text classifiers

Thursday

- Logistic regression
- Cross-entropy loss function
- Gradient descent optimization
- Other classification details

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
  - $loss(y_i, \hat{y_i}) = -\sum_{c=1}^{|C|} p_{i,c} \log \widehat{p_{i,c}}$
    - *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*
- 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

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

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. 🙄	0.96	0.04	1	0

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

$$loss(y_i, y_i') = -1 * \log 0.96 - 0 * \log 0.04$$

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

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

$$loss(y_i, y_i') = -1 * \log 0.96 - 0 * \log 0.04 = -\log 0.96 = 0.02$$

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

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

$$loss(y_i, y_i') = -1 * \log 0.04 - 0 * \log 0.96 = -\log 0.04 = 1.40$$

Greater loss value!

This Week's Topics

Tuesday

- Text classification
- Naïve Bayes
- Evaluating text classifiers

Thursday

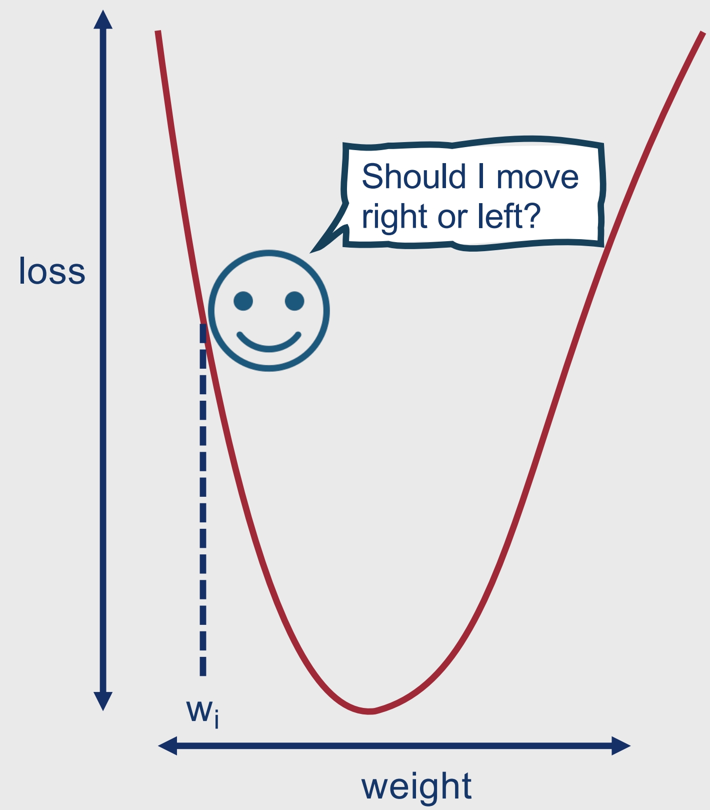
- Logistic regression
- Cross-entropy loss function
- Gradient descent optimization
- Other classification details

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**

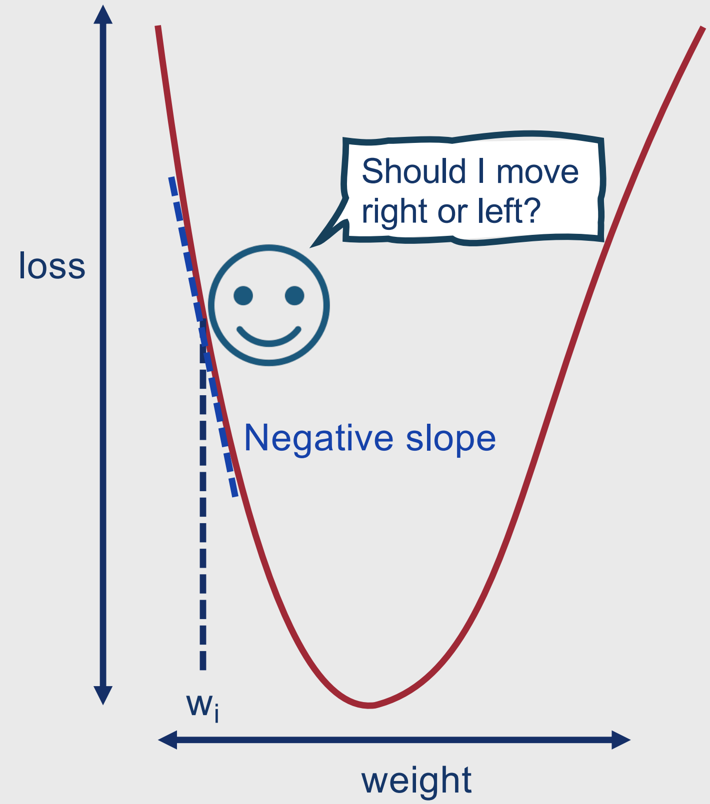
Gradient Descent

- Finds the minimum of a function by updating weights based on the 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



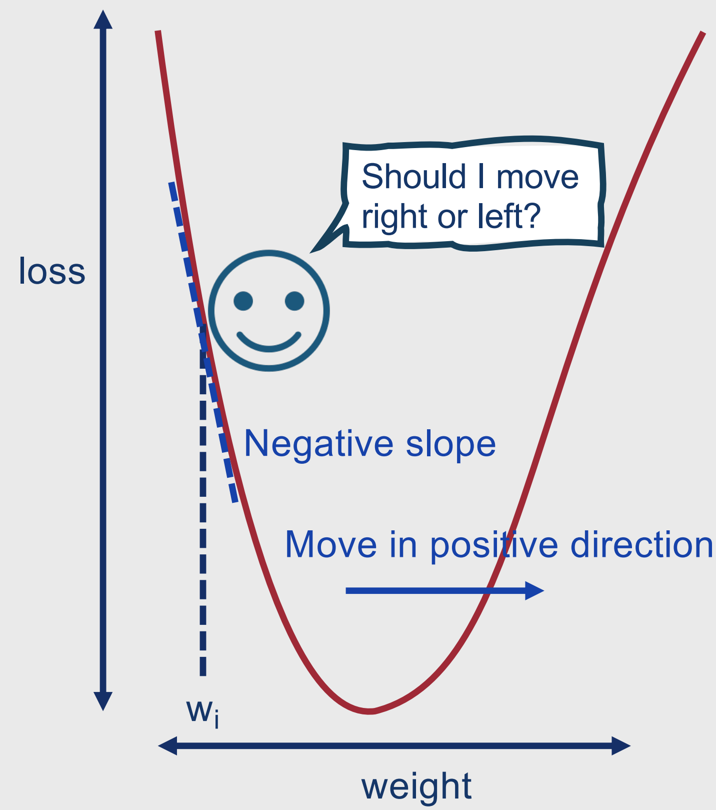
Gradient Descent

- Finds the minimum of a function by updating weights based on the 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



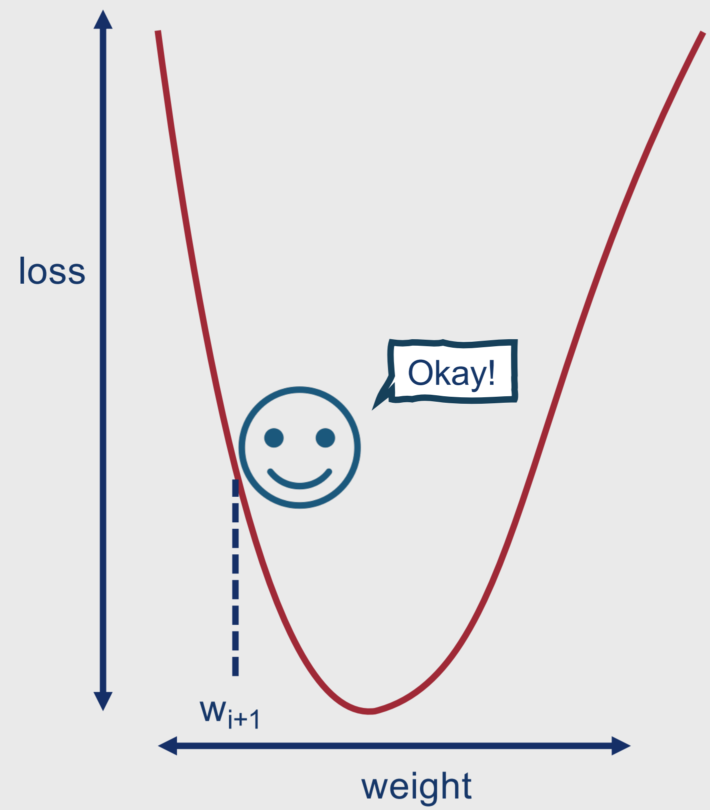
Gradient Descent

- Finds the minimum of a function by updating weights based on the 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

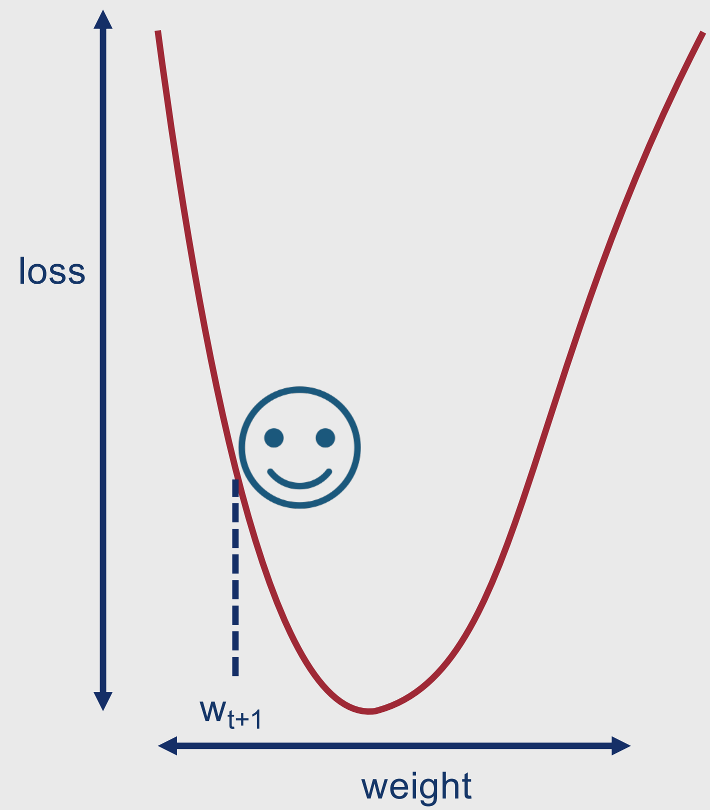
- Finds the minimum of a function by updating weights based on the direction of the function's slope
- For logistic regression, loss functions are **convex**
  - Only one minimum
  - Gradient descent starting at any point is guaranteed to find it



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 → move *w* more on each step
- So, the change to a weight is actually:
  - $w^{t+1} = w^t - \eta \frac{d}{dw} f(x; w)$

Derivative of loss function curve with respect to a given weight



Remember, 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 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:
  - $loss(y_i, \hat{y_i}) = -\sum_{c=1}^{|C|} y \log \hat{y} = -\sum_{c=1}^{|C|} y \log \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
- The derivative for this function is:
  - $\frac{dL_{CE}(w,b)}{dw_j} = \underbrace{[\sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b) - y]}_{\text{Difference between true and estimated } y} x_j$

Difference between true and estimated *y*

Corresponding input observation

Stochastic Gradient Descent Algorithm

```
θ←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 ← ∇_θ L(f(x^(i); θ), y^(i))

            θ ← θ − ηg  # What are our updated parameters?
      return θ
```

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

Bias (b) = 0
Learning rate (η) = 0.1

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

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

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

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

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

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

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

Bias (b) = 0

Learning rate (η) = 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 (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

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

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

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

Example: Gradient Descent (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

Bias (b) = 0.05

Learning rate (η) = 0.1

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

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

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

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

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

Bias (b) = 0.05

Learning rate (η) = 0.1

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

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

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

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

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

Interpreting Logistic Regression Weights

Feature	Weight	Value
Contains 🙄	0.096	1
Contains 😊	0	0
Contains “I’m”	0.096	1

Less informative

More informative

How do we evaluate logistic regression classifiers?

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

This Week's Topics

Tuesday

- Text classification
- Naïve Bayes
- Evaluating text classifiers

Thursday

- Logistic regression
- Cross-entropy loss function
- Gradient descent optimization
- Other classification details

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

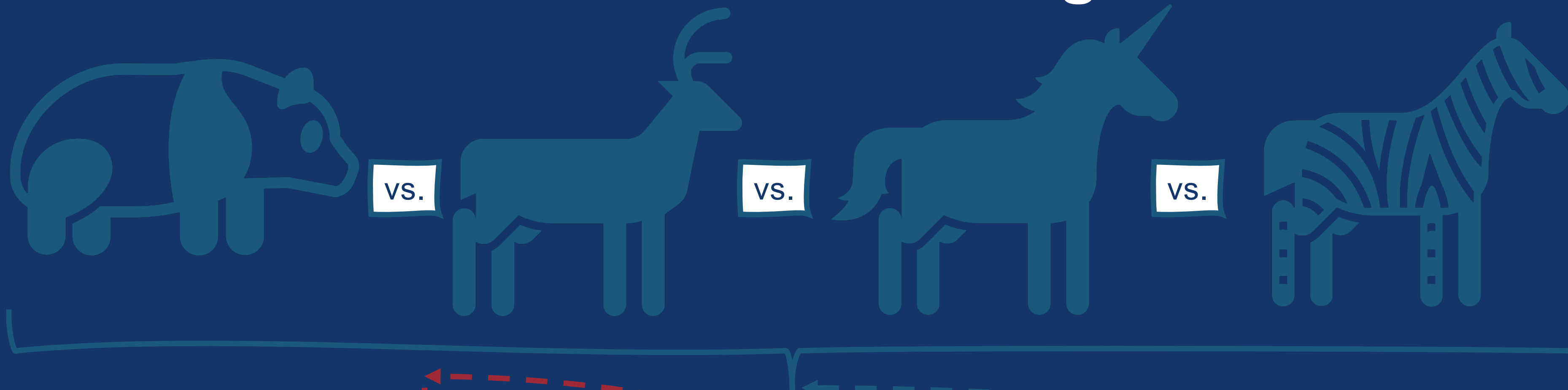
What if the entire dataset is pretty small?

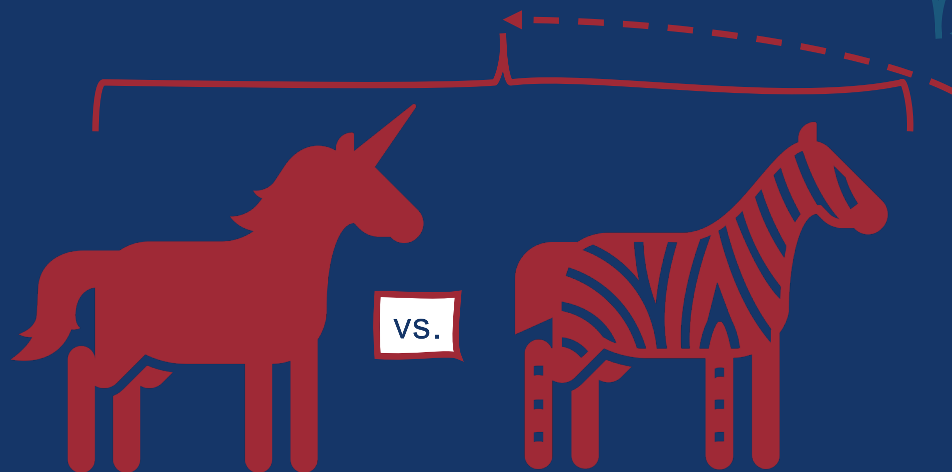
- In cases where the entire dataset is small, it may be undesirable to reserve an entire fold of data for validation
  - Smaller training set (less data from which to learn)
  - Smaller test set (less data on which to evaluate)
- In these cases, a reasonable alternative is cross-validation
  - Randomly 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

- 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?





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

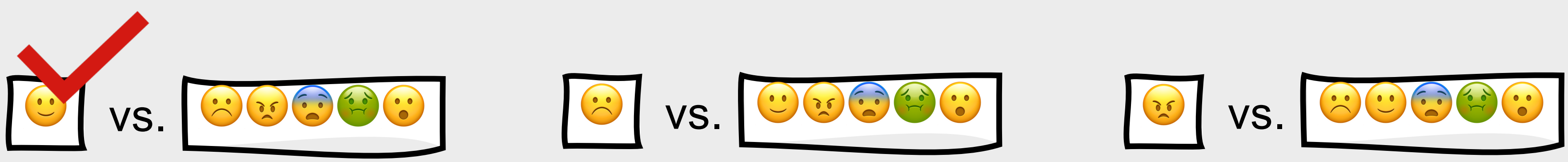
- **Binary**
  - Class A vs. Class B
- **Multinomial**
  - Class A vs. Class B vs. Class C vs. Class D….

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 **z** of arbitrary values (same as the sigmoid function) and maps them to a probability distribution summing to 1
  - $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{|\mathbf{Z}|} e^{z_j}}$

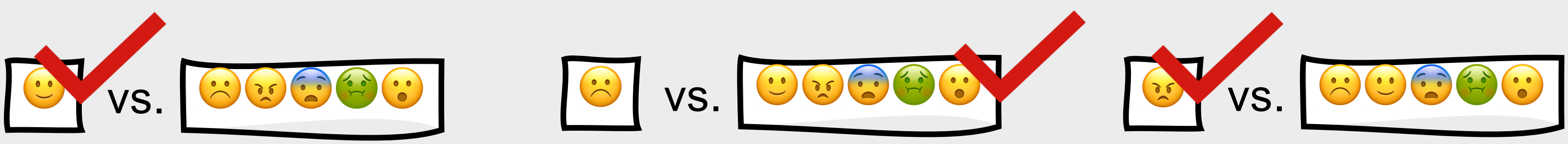
One vs. All Classification

- 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

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

Multi-Class Precision

Actual

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

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

Multi-Class Recall

Actual

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

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

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

Macroaveraging

- We can check the system's **overall performance** in multi-class classification settings by combining all of the calculated performance values
- **Macroaveraging:** Compute the performance for each class, and then average over all classes
  - Evenly distributes performance estimates across classes, so less-frequent (and/or lower-performing) classes factor in equally as heavily

Macroaveraging

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

class 1

TP: a	FP: b+c
FN: d+g	

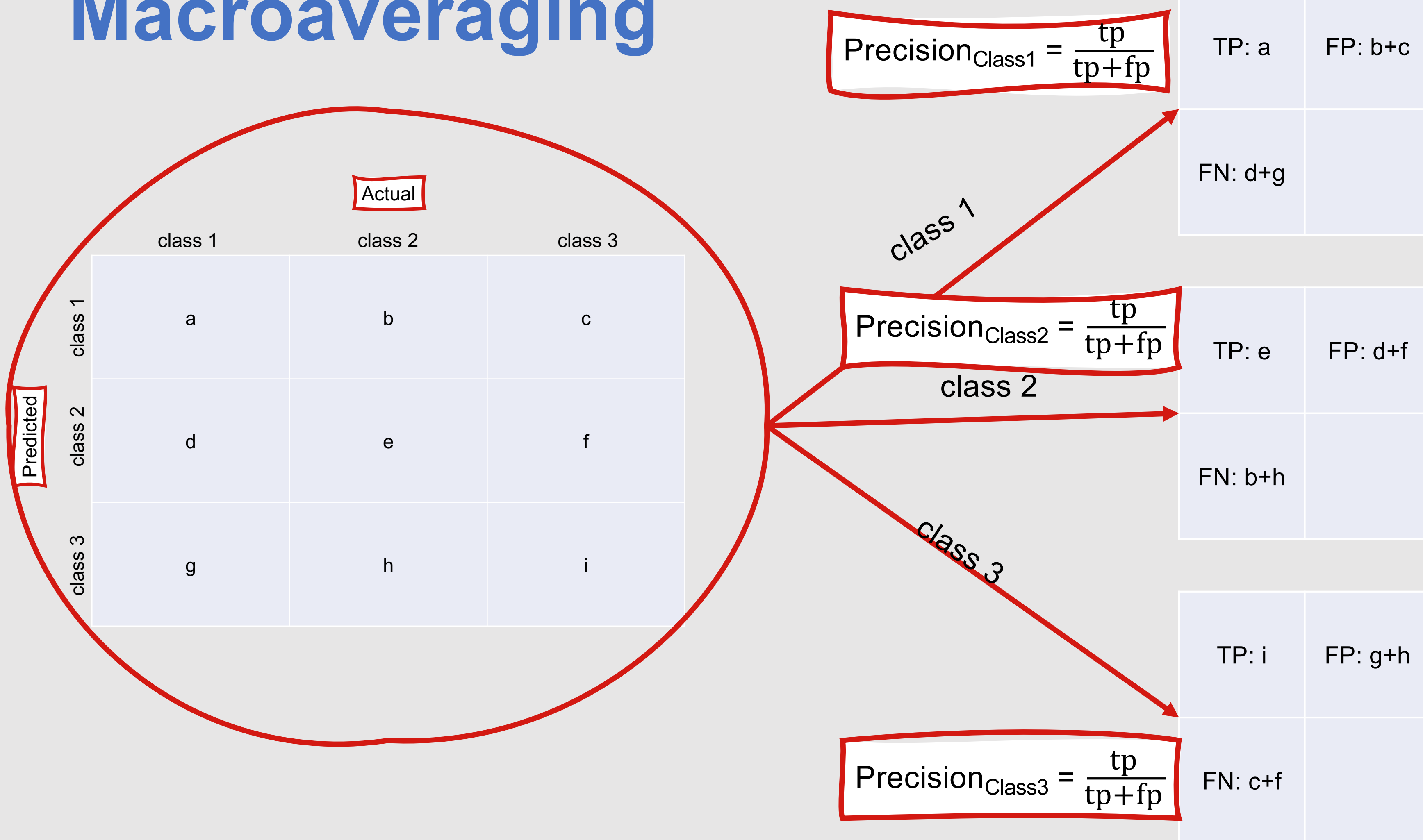
class 2

TP: e	FP: d+f
FN: b+h	

class 3

TP: i	FP: g+h
FN: c+f	

Macroaveraging



Macroaveraging

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

class 1

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

TP: a	FP: b+c
FN: d+g	

class 2

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

TP: e	FP: d+f
FN: b+h	

class 3

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

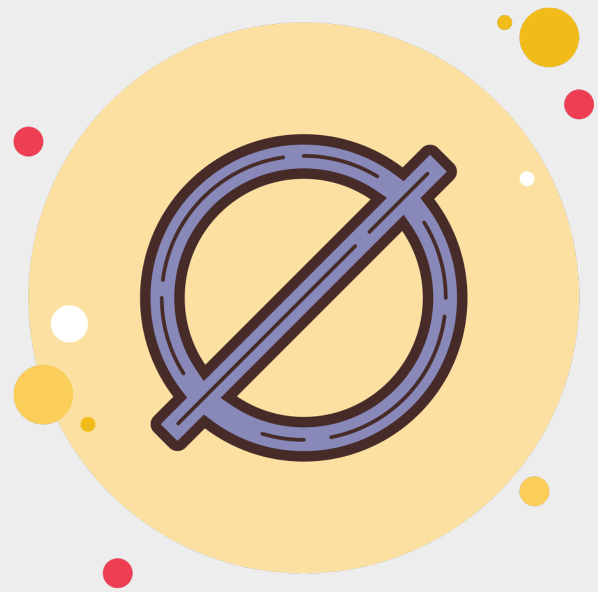
TP: i	FP: g+h
FN: c+f	

Macroaveraged Precision

$$= \frac{Precision_{Class1} + Precision_{Class2} + Precision_{Class3}}{3}$$

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:** This is due to chance, rather than some meaningful reason



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 δ(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 δ(x*(i))  # Performance difference between Models A
                          # and B for the bootstrap sample x*(i)
for each x*(i):
      s = s+1 if δ(x*(i)) > 2δ(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

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**
- We can evaluate the significance of our evaluation results using **statistical significance testing**