



Is this spam?

Subject: Important notice!

From: Stanford University <newsforum@stanford.edu>

Date: October 28, 2011 12:34:16 PM PDT

To: undisclosed-recipients;;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

<http://www.123contactform.com/contact-form-StanfordNew1-236335.html>

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

© Stanford University. All Rights Reserved.



Positive or negative movie review?



- unbelievably disappointing



- Full of zany characters and richly applied satire, and some great plot twists



- this is the greatest screwball comedy ever filmed



- It was pathetic. The worst part about it was the boxing scenes.

MEDLINE Article



MeSH Subject Category Hierarchy

- Antagonists and Inhibitors
- Blood Supply
- Chemistry
- Drug Therapy
- Embryology
- Epidemiology
- ...



Text Classification

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
- Sentiment analysis
- ...



Text Classification: definition

- *Input:*
 - a document d
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
- *Output:* a predicted class $c \in C$



Classification Methods:

Hand-coded rules

- Rules based on combinations of words or other features
 - spam: black-list-address OR (“dollars” AND “have been selected”)
- Accuracy can be high
 - If rules carefully refined by expert
- But building and maintaining these rules is expensive



Classification Methods: Supervised Machine Learning

- *Input:*
 - a document d
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
 - A training set of m hand-labeled documents $(d_1, c_1), \dots, (d_m, c_m)$
- *Output:*
 - a learned classifier $\gamma: d \rightarrow c$



Classification Methods: Supervised Machine Learning

- Any kind of classifier
 - Logistic regression
 - Support-vector machines
 - Decision Trees
 - Neural Networks

Basic Text Processing

Words and Corpora

Corpora

Words don't appear out of nowhere!

A text is produced by

- a specific writer(s),
- at a specific time,
- in a specific variety,
- of a specific language,
- for a specific function.

We call a collection text a ***Corpus***

Corpora vary along dimension like

- **Language:** 7097 languages in the world
- **Variety**, like African American Language varieties.
 - AAE Twitter posts might include forms like "*iont*" (*I don't*)
- **Code switching**, e.g., Spanish/English, Hindi/English:
 - S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)
[For the first time I get to see @username actually being hateful! it was beautiful:]
 - H/E: dost tha or ra- hega ... dont worry ... but dherya rakhe
["he was and will remain a friend ... don't worry ... but have faith"]
- **Genre:** newswire, fiction, scientific articles, Wikipedia
- **Author Demographics:** writer's age, gender, ethnicity, SES

Classification Methods: Supervised Machine Learning

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
- A training set of m hand-labeled documents
 $(d_1, c_1), \dots, (d_m, c_m)$

Output:

- a learned classifier $\gamma: d \rightarrow c$

Definition of a Corpus

N = number of **tokens**

V = vocabulary = set of **types**, $|V|$ is size of vocabulary

Type: an element of the vocabulary.

Token: an instance of that type in running text.

Joe took his car to the car mechanic.

- 8 Tokens
- 7 Types (car is repeated)

How many words in a sentence?

"I do uh main- mainly business data processing"

- Fragments, filled pauses

"Seuss's **cat** in the hat is different from other **cats**!"

- **Lemma**: same stem, part of speech, rough word sense
 - **cat** and **cats** = same lemma
- **Wordform**: the full inflected surface form
 - **cat** and **cats** = different wordforms

Other Issues in Tokenization

Clitic: a word that doesn't stand on its own

- "are" in *we're*, French "je" in *j'ai*, "le" in *l'honneur*

When should multiword expressions (MWE) be words?

- *New York, rock 'n' roll*

Lexicons

Sometimes we don't have enough labeled training data

In that case, we can make use of pre-built word lists

Called **lexicons**

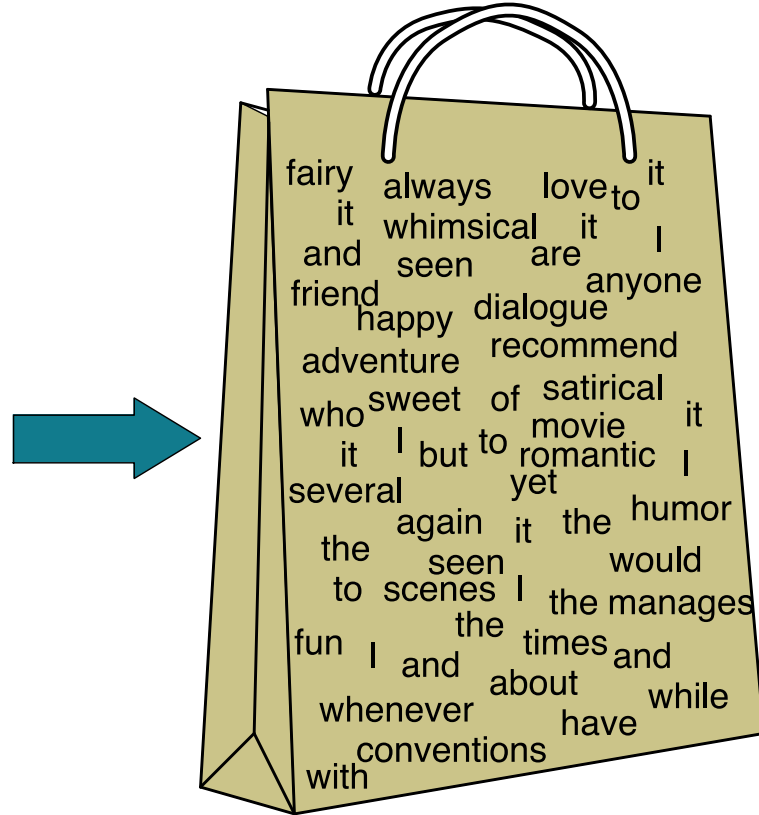
There are various publically available lexicons

Basic Text
Processing

Building a Basic Classifier

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!





it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

The bag of words representation

$Y($

seen	2
sweet	1
whimsical	1
recommend	1
happy	1
...	...

$) = C$

Problems with BoW

I am happy, not sad today.

I am sad, not happy today.

Both result in the same representation!

BoW does not take sequence into account.

This is often why deep learning is used.

Negation

I really like this movie

I really **don't** like this movie

Negation changes the meaning of "like" to negative.

Negation can also change negative to positive-ish

- **Don't** dismiss this film
- **Doesn't** let us get bored

Benefits of BoW

- Can be used to quickly construct a feature set
- Interpretable/ Human-Understandable
- Can be efficient to store.

BoW Representations

Binary: 1 if word appears, zero else

TermFrequency: List number of times a word appears

Term Frequency – inverse Document Frequency:

- **Term-Frequency normalized by Document Frequency**

TF-IDF

$$w_{x,y} = tf_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

TF-IDF

Provides a better representation compared to term frequency

Words mentioned in a lot of documents are weighted less

Rare words are weighted more

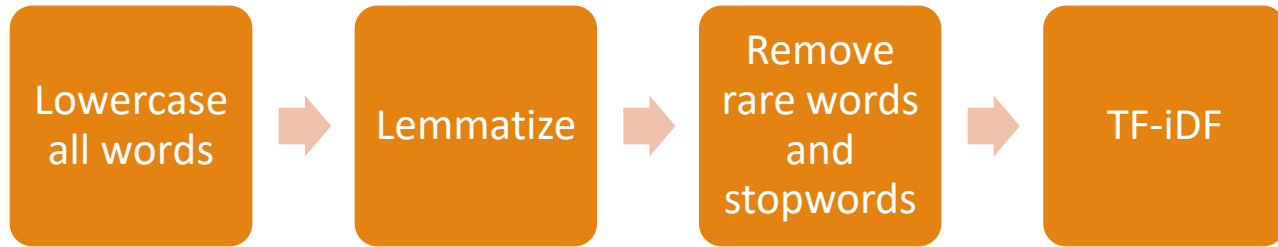
Stopwords and Rarewords

The longest English word is **pneumonoultramicroscopicsilicovolcanoconiosis**, which is forty-five letters long and refers to a type of lung disease.

- **Rare Words:** Not enough frequency to learn its meaning
- **Stopwords:** Too common and don't provide much info

We often remove both

Example Feature Matrix Construction



We can now construct a classifier on the resultant representation.

Example: Logistic Regression

Example: Logistic Regression

Given a series of input/output pairs:

- $(x^{(i)}, y^{(i)})$

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

- We represent $x^{(i)}$ by a **feature vector** $[x_1, x_2, \dots, x_n]$
- We compute an output: a predicted class $\hat{y}^{(i)} \in \{0, 1\}$

Features in logistic regression

- For feature x_i , weight w_i tells is how important is x_i
 - x_i = "review contains 'awesome'": $w_i = +10$
 - x_j = "review contains 'abysmal'": $w_j = -10$
 - x_k = "review contains 'mediocre'": $w_k = -2$

Logistic Regression for one observation x

Input observation: vector $x = [x_1, x_2, \dots, x_n]$

Weights: one per feature: $W = [w_1, w_2, \dots, w_n]$

- Sometimes we call the weights $\theta = [\theta_1, \theta_2, \dots, \theta_n]$

Output: a predicted class $\hat{y} \in \{0, 1\}$

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

How to do classification

For each feature x_i , weight w_i tells us importance of x_i

- (Plus we'll have a bias b)

We'll sum up all the weighted features and the bias

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

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

But we want a probabilistic classifier

We need to formalize “sum is high”.

We’d like a principled classifier that gives us a probability, just like Naive Bayes did

We want a model that can tell us:

$$p(y=1 | x; \theta)$$

$$p(y=0 | x; \theta)$$

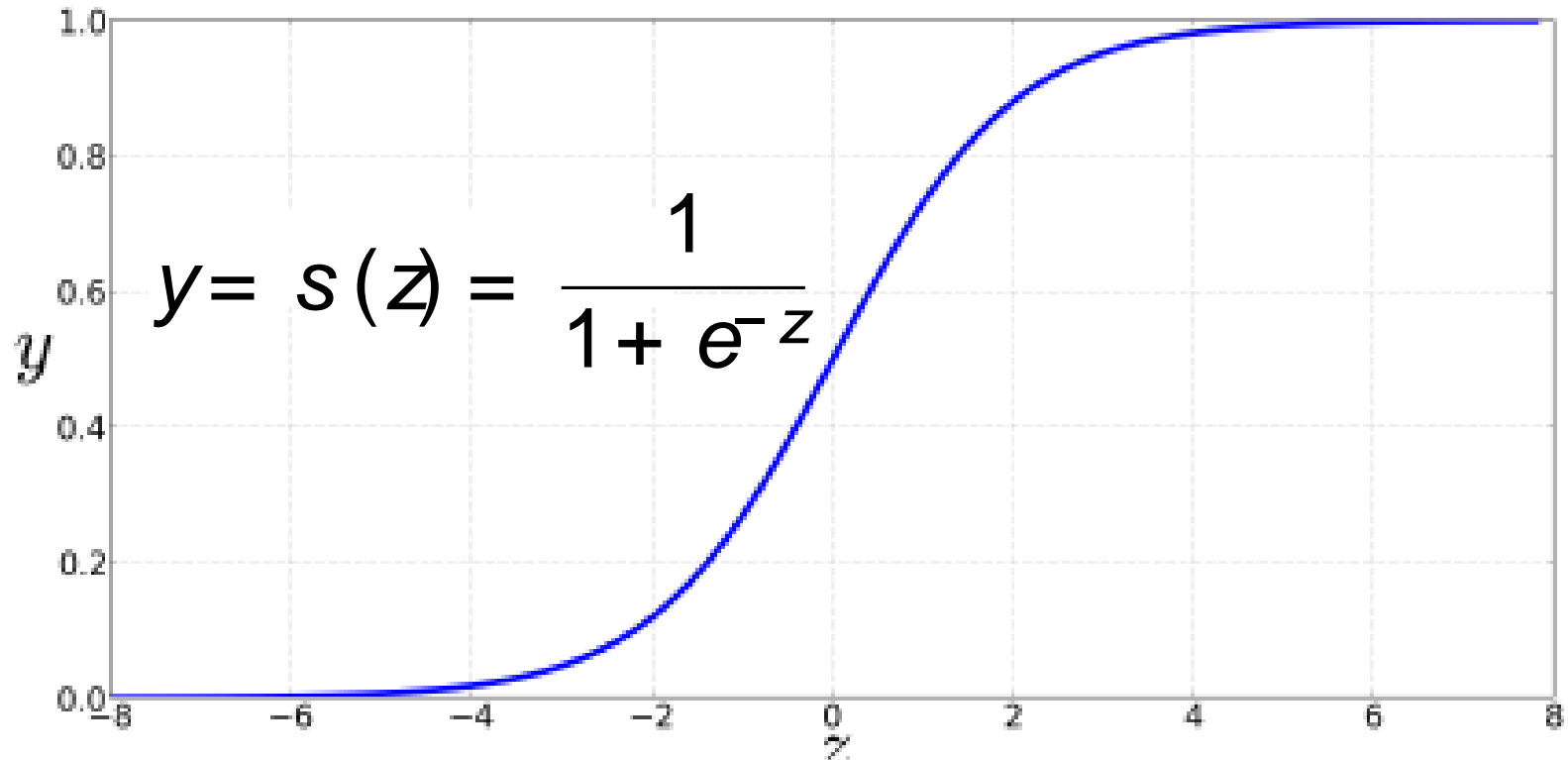
The problem: z isn't a probability, it's just a number!

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

Solution: use a function of z that goes from 0 to 1

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

The very useful sigmoid or logistic function



Idea of logistic regression

We'll compute $w \cdot x + b$

And then we'll pass it through the sigmoid function:

$$\sigma(w \cdot x + b)$$

And we'll just treat it as a probability

Making probabilities with sigmoids

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

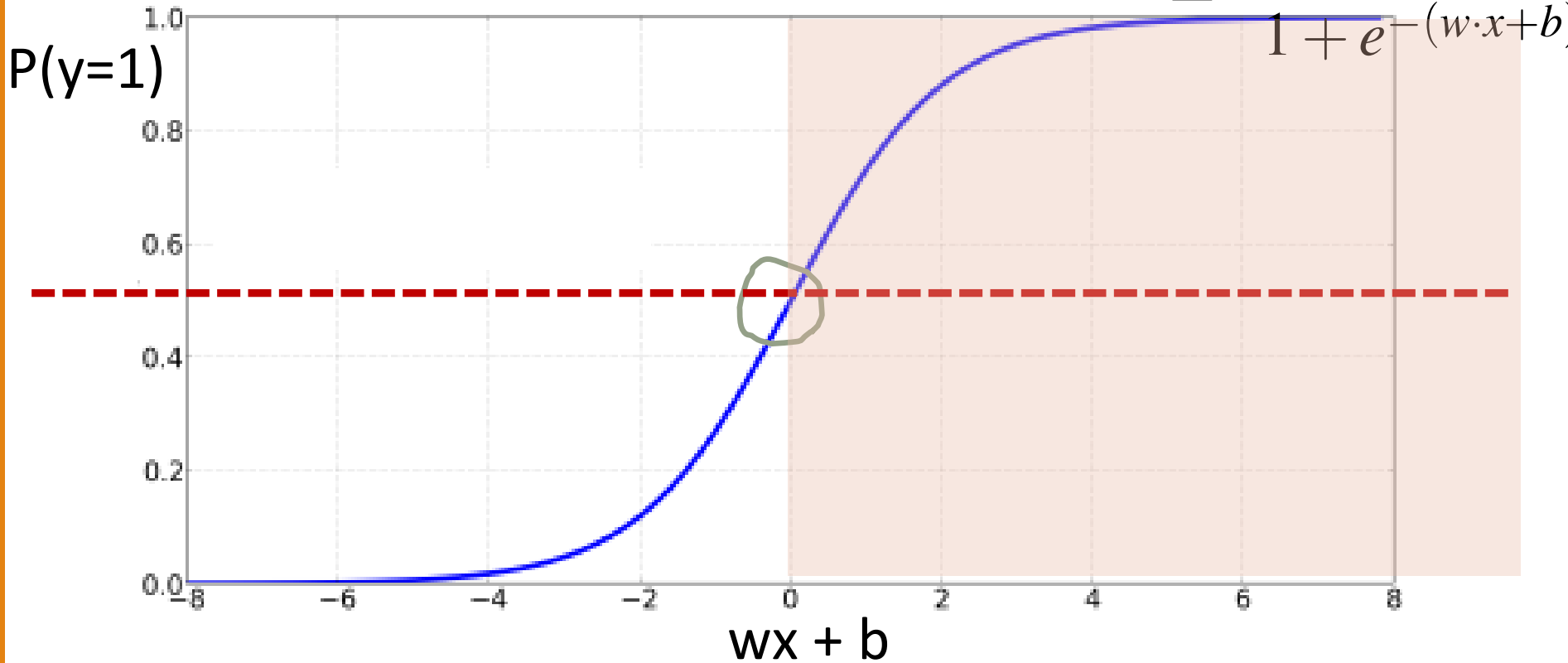
$$\begin{aligned} P(y = 0) &= 1 - \sigma(w \cdot x + b) \\ &= 1 - \frac{1}{1 + \exp(-(w \cdot x + b))} \\ &= \frac{\exp(-(w \cdot x + b))}{1 + \exp(-(w \cdot x + b))} \end{aligned}$$

Turning a probability into a classifier

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

0.5 here is called the **decision boundary**

The probabilistic classifier $P(y = 1) = \sigma(w \cdot x + b)$



Turning a probability into a classifier

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad \begin{array}{l} \text{if } \mathbf{w} \cdot \mathbf{x} + \mathbf{b} > 0 \\ \text{if } \mathbf{w} \cdot \mathbf{x} + \mathbf{b} \leq 0 \end{array}$$

Deriving cross-entropy loss for a single observation x

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

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

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

noting:

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

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

Deriving cross-entropy loss for a single observation x

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

Maximize:

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

Now flip sign to turn this into a loss: something to minimize

Cross-entropy loss (because is formula for cross-entropy(y, \hat{y}))

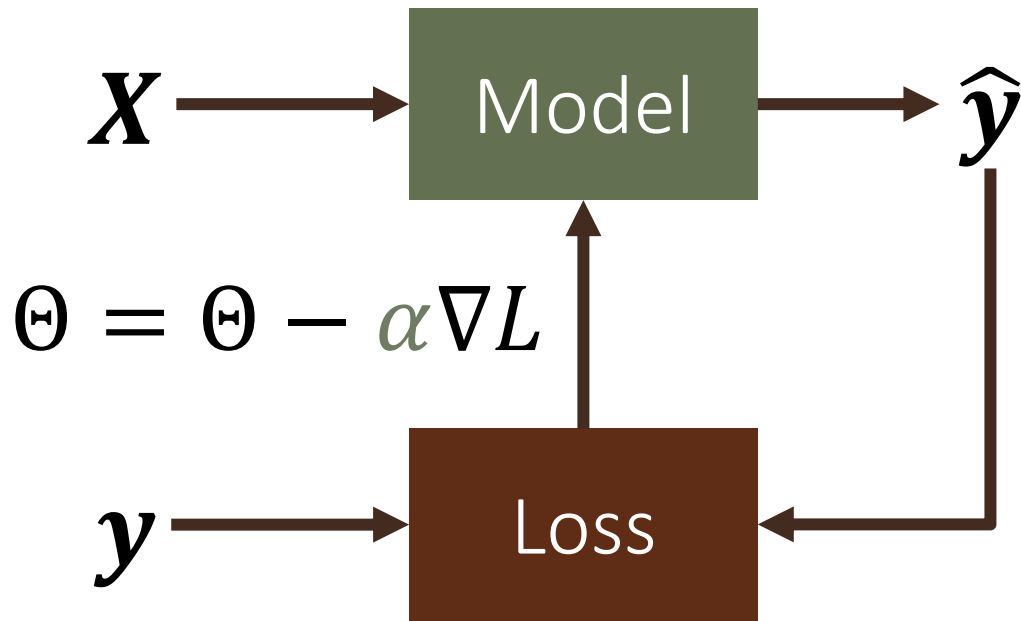
Minimize:

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

Or, plugging in definition of \hat{y} :

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

Gradient Descent



Precision, Recall, and F1

Text
Classification

Evaluating Classifiers: How well does our classifier work?

Let's first address binary classifiers:

- Is this email spam?

spam (+) or not spam (-)

- Is this post about Delicious Pie Company?

about Del. Pie Co (+) or not about Del. Pie Co(-)

We'll need to know

1. What did our classifier say about each email or post?
2. What should our classifier have said, i.e., the correct answer, usually as defined by humans ("gold label")

First step in evaluation: The confusion matrix

		<i>gold standard labels</i>	
		gold positive	gold negative
<i>system output labels</i>	system positive	true positive	false positive
	system negative	false negative	true negative

Accuracy on the confusion matrix

		<i>gold standard labels</i>	
		gold positive	gold negative
<i>system output labels</i>	system positive	true positive	false positive
	system negative	false negative	true negative

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{fp} + \text{tn} + \text{fn}}$$

Why don't we use accuracy?

Accuracy doesn't work well when we're dealing with uncommon or imbalanced classes

Suppose we look at 1,000,000 social media posts to find Delicious Pie-lovers (or haters)

- 100 of them talk about our pie
- 999,900 are posts about something unrelated

Imagine the following simple classifier

Every post is "not about pie"

Accuracy re: pie posts

100 posts are about pie; 999,900 aren't

		<i>gold standard labels</i>	
		gold positive	gold negative
<i>system output labels</i>	system positive	true positive	false positive
	system negative	false negative	true negative

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{fp} + \text{tn} + \text{fn}}$$

Why don't we use accuracy?

Accuracy of our "nothing is pie" classifier

999,900 true negatives and 100 false negatives

Accuracy is $999,900/1,000,000 = 99.99\%$

But useless at finding pie-lovers (or haters)!!

Which was our goal!

Accuracy doesn't work well for unbalanced classes

Most tweets are not about pie!

Instead of accuracy we use precision and recall

		<i>gold standard labels</i>		
		gold positive	gold negative	
<i>system output labels</i>	system positive	true positive	false positive	precision = $\frac{tp}{tp+fp}$
	system negative	false negative	true negative	
		recall = $\frac{tp}{tp+fn}$		accuracy = $\frac{tp+tn}{tp+fp+tn+fn}$

Precision: % of selected items that are correct

Recall: % of correct items that are selected

Precision/Recall aren't fooled by the "just call everything negative" classifier!

Stupid classifier: Just say no: every tweet is "not about pie"

- 100 tweets talk about pie, 999,900 tweets don't
- Accuracy = $999,900/1,000,000 = 99.99\%$

But the Recall and Precision for this classifier are terrible:

$$\mathbf{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\mathbf{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

A combined measure: F1

F1 is a combination of precision and recall.

$$F_1 = \frac{2PR}{P + R}$$

F1 is a special case of the general "F-measure"

F-measure is the (weighted) harmonic mean of precision and recall

$$\text{HarmonicMean}(a_1, a_2, a_3, a_4, \dots, a_n) = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3} + \dots + \frac{1}{a_n}}$$

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} \quad \text{or} \left(\text{with } \beta^2 = \frac{1 - \alpha}{\alpha} \right) \quad F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

F1 is a special case of F-measure with $\beta=1$, $\alpha=\frac{1}{2}$

Suppose we have more than 2 classes?

Lots of text classification tasks have more than two classes.

- Sentiment analysis (positive, negative, neutral) , named entities (person, location, organization)

We can define precision and recall for multiple classes like this 3-way email task:

		<i>gold labels</i>			
		urgent	normal	spam	
<i>system output</i>	urgent	8	10	1	precision_u = $\frac{8}{8+10+1}$
	normal	5	60	50	precision_n = $\frac{60}{5+60+50}$
	spam	3	30	200	precision_s = $\frac{200}{3+30+200}$
		recall_u = $\frac{8}{8+5+3}$	recall_n = $\frac{60}{10+60+30}$	recall_s = $\frac{200}{1+50+200}$	

How to combine P/R values for different classes: Microaveraging vs Macroaveraging

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = .73$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = .60$$

Text Classification and Naive Bayes

Avoiding Harms in Classification

Harms of classification

Classifiers, like any NLP algorithm, can cause harms

This is true for any classifier, whether Naive Bayes or other algorithms

Representational Harms

- Harms caused by a system that demeans a social group
 - Such as by perpetuating negative stereotypes about them.
- Kiritchenko and Mohammad 2018 study
 - Examined 200 **sentiment analysis** systems on pairs of sentences
 - **Identical** except for names:
 - common African American (Shaniqua) or European American (Stephanie).
 - Like "I talked to Shaniqua yesterday" vs "I talked to Stephanie yesterday"
- Result: systems assigned **lower sentiment** and more negative emotion to sentences with **African American names**
- Downstream harm:
 - Perpetuates stereotypes about African Americans
 - African Americans treated differently by NLP tools like sentiment (widely used in marketing research, mental health studies, etc.)

Harms of Censorship

- **Toxicity detection** is the text classification task of detecting hate speech, abuse, harassment, or other kinds of toxic language.
 - Widely used in online content moderation
- Toxicity classifiers incorrectly flag non-toxic sentences that simply mention minority identities (like the words "blind" or "gay")
 - women (Park et al., 2018),
 - disabled people (Hutchinson et al., 2020)
 - gay people (Dixon et al., 2018; Oliva et al., 2021)
- Downstream harms:
 - Censorship of speech by disabled people and other groups
 - Speech by these groups becomes less visible online
 - Writers might be nudged by these algorithms to avoid these words making people less likely to write about themselves or these groups.

Performance Disparities

1. Text classifiers perform worse on many **languages** of the world due to lack of data or labels
2. Text classifiers perform worse on **varieties** of even high-resource languages like English
 - Example task: **language identification**, a first step in NLP pipeline ("Is this post in English or not?")
 - English language detection performance worse for writers who are African American (Blodgett and O'Connor 2017) or from India (Jurgens et al., 2017)

Harms in text classification

- **Causes:**
 - Issues in the data; NLP systems amplify biases in training data
 - Problems in the labels
 - Problems in the algorithms (like what the model is trained to optimize)
- **Prevalence:** The same problems occur throughout NLP (including large language models)
- **Solutions:** There are no general mitigations or solutions
 - But harm mitigation is an active area of research
 - And there are standard benchmarks and tools that we can use for measuring some of the harms