BC COMS 2710: Computational Text Analysis

Lecture 17 – Machine Learning: Text Classification (Logistic Regression)

Announcements – Assignments

- Readings 05:
  - link posted to course site
  - due Sunday
- HW 03:
  - Released last Friday
  - Optional – has anyone looked at it?
- HW04/Tutorial 5.1
  - Releasing tomorrow or Friday

HW04/Tutorial 5.1

- Given Tweet IDs and labels
- Task:
  - retrieve the tweets using Twitter API
  - Build machine learning classifiers to predict labels on held-out examples
- Data comes from a real 2020 shared task:
  - *For this task, participants are asked to develop systems that automatically identify whether an English Tweet related to the novel coronavirus (COVID-19) is informative or not. Such informative Tweets provide information about recovered, suspected, confirmed and death cases as well as location or travel history of the cases*.

Final Project – Deliverables

- Project ideation – Friday May 28st
  - 5 points
- Project proposal – Sunday June 6th
  - 9 points
- Project presentations – Monday June 14th
  - 6 points
- Project submissions – Friday June 18th
  - 15 points
- http://coms2710.barnard.edu/final_project

Final Project – Proposal due Sunday June 6th

Beefed up version of project ideation

1. Research Question
2. Detailed source of data:
   1. List of twitter user's, subreddits, etc
3. Detailed methods you plan on applying for exploratory data analysis
   1. Tf-idf, topic modeling, …
4. Prediction

Final Project Submission – Friday June 18th

- Paper/write up:
  - 3-5 page double spaces, including a few figures and tables
- Notebook:
  - With code for data collection, data analysis, prediction
- Data

Naive Bayes

Naive Bayes Classifier

Given **X**, what is the most probable **Y**? P(Y | X)

How do we determine most probable **Y**?

By Bayes Rule!

"Prior"

"Likelihood"

$$Y \leftarrow \operatorname{argmax}_{y_i} P(Y = y_i) P(X \mid Y = y_i)$$

Naive Bayes Conditional Independence Assumption

$$Y \leftarrow \operatorname{argmax}_{y_i} P(Y = y_i)\, P(x_1 \mid Y = y_i) P(x_2 \mid Y = y_i) * \cdots * P(x_n \mid Y = y_i)$$

$$Y \leftarrow \operatorname{argmax}_{y_i} P(Y = y_i) \prod_n P(x_n \mid Y = y_i)$$

Issue with probabilities?

$$Y \leftarrow \operatorname{argmax}_{y_i} P(Y = y_i) \prod_n P(x_n \mid Y = y_i)$$

Hint: Multiplying probabilities leads to …

even smaller numbers and eventual floating-point underflow

Any solutions?

logs to the rescue

$$\log(x * y) = \log(x) + \log(y)$$

$$Y \leftarrow \operatorname{argmax}_{y_i} P(Y = y_i) \prod_n P(x_n \mid Y = y_i)$$

$$Y \leftarrow \operatorname{argmax}_{y_i} \log P(y_i) + \sum_n \log P(x_n \mid y_i)$$

Class with highest log probability is still most probably label ***Y*** for example ***X***.

Features

- Inputs to classifiers are *features*
- We counted words, so think of each word as a feature

- Define a *feature function* over document **x**:

$$f_i(x)$$

- Each unique word has a feature index ***i***
- The function returns the count of word ***i***

Slide from Nate Chambers

Naive Bayes Summary

- Fast algorithm:
  - Only requires going through the data once
- Works well with small amounts of training data
- Robust to irrelevant features
- Optimal if independence assumption holds
- Interpretable
- A good dependable baseline for text classification

Slide from Dan Jurafsky

Logistic Regression

Alternative to Naive Bayes

- Don't want independence assumption
- Goal: *weigh* a feature that helps improve accuracy, but give less *weight* to the feature if other features overlap with the same correct prediction
- Solution: Logistic Regression
  - Maximum Entropy (MaxtEnt)
  - Multinomial logistic regression
  - Log-linear model
  - Neural network (single layer)

Slide from Nate Chambers

Feature Example

Document	Text	Author
X_1	the lady doth protest too much methinks	Shakespeare
X_2	it was the best of times it was the worst of times	Dickens

$f_7(x)$ is "the"

$f_7(x_1) = 1$

$f_7(x_2) = 2$

$f_{72}(x)$ is "the best"

$f_{72}(x_1) = 0$

$f_{72}(x_2) = 1$

Slide from Nate Chambers

Weights

Assume we have the a document with the following features

$f_1(x)$ = 1

$f_2(x)$ = 2

$f_3(x)$ = 1

Slide from Nate Chambers

Weights

Assume we have the a document with the following features. Goal is to classify the document as being written by Shakespeare or Dickens

$f_1(x) = 1$

$f_2(x) = 2$

$f_3(x) = 1$

Let's add weights to the features

Slide from Nate Chambers

Weights

- Now let's add *weights* to the features

	Shakespeare	Dickens
$f_1(x) = 1$	1.31	-0.23
$f_2(x) = 2$	0.49	0.72
$f_3(x) = 1$	-0.82	0.1

Slide from Nate Chambers

Weights

- Now let's add *weights* to the features
- We want a *score* for each class label

	Shakespeare	Dickens
$f_1(x) = 1$	1.31	-0.23
$f_2(x) = 2$	0.49	0.72
$f_3(x) = 1$	-0.82	0.1

Slide from Nate Chambers

Weights

- Now let's add *weights* to the features
- We want a *score* for each class label

	Shakespeare	Dickens
$f_1(x)$ = 1	1.31	-0.23
$f_2(x)$ = 2	0.49	0.72
$f_3(x)$ = 1	-0.82	0.1
	1.47	1.31

$$\text{score}(x, c) = \sum_i w_{i,c} f_i(x)$$

Slide from Nate Chambers

Weights

$$\text{score}(x, c) = \sum_i w_{i,c} f_i(x)$$

Shakespeare	Dickens
1.47	1.31

But we want probabilities:

$$P(c \mid x) = \frac{\sum_i w_{i,c} f_i(x)}{Z} \qquad Z = \sum_c \sum_i w_{i,c} f_i(x)$$

Math Trick to guarantee values between [0, 1]

$$P(c \mid x) = \frac{\exp(\sum_i w_{i,c} f_i(x))}{Z} \qquad Z = \sum_c \sum_i \exp(w_{i,c} f_i(x))$$

Slide from Nate Chambers

Logistic Regression

- Logistic Regression is a vector of weights multiplied by a vector of features

$$P(c \mid x) = \frac{1}{Z} exp(\sum_i w_{i,c} f_i(x))$$

Slide from Nate Chambers

Logistic Regression

- Logistic Regression is a vector of weights multiplied by a vector of features

$$P(c \mid x) = \frac{1}{Z} exp(\sum_i w_{i,c} f_i(x))$$

Slide from Nate Chambers

Logistic Regression

- Logistic Regression is a vector of weights multiplied by a vector of features

$$P(c \mid x) = \frac{1}{Z} exp(\sum_i w_{i,c} f_i(x))$$

Slide from Nate Chambers

Logistic Regression

- Logistic Regression is a vector of weights multiplied by a vector of features

$$P(c \mid x) = \frac{1}{Z} exp(\sum_i w_{i,c} f_i(x))$$

- Normalized to get probabilities

Slide from Nate Chambers

Logistic Regression

"it was the best of times it was the worst of times"

	it	was	the	best	of	apple	pizza	worst	ok
$f(x)$	2	1	2	1	2	0	0	1	0

Dickens **w**

0.2	-0.4	0.32	-0.43	0.3	0.01	0.29	-0.31	0.02

Shakespeare **w**

-0.02	0.5	0.2	0.11	0.22	0.32	0.12	-0.3	0

Where do the weights come from?

Slide from Nate Chambers

Learning in Logistic Regression

- We need to learn the weights
- Goal: choose weights the give the “best results”
  - or the weights the give the “least error”

- Loss function: measures how *wrong* our predictions are

$$Loss(y) = -\sum_{n=1}^{N} 1\{y = n\} \log p(y \mid x_n)$$

Example!

$$Loss(dickens) = -\log p(dickens \mid x_n)$$

when p(y|x) = 1.0, the loss will be 0

Slide from Nate Chambers

Learning in Logistic Regression

- **Goal:** choose weights the give the "least error"

$$Loss(y) = -\sum_{n=1}^{N} 1\{y = n\} \log p(y \mid x_n)$$

$$P(y \mid x) = \frac{1}{Z} exp(\sum_i w_{i,y} f_i(x))$$

- Choose weights the give probabilities close to 1.0 to each of the correct labels

Slide from Nate Chambers

Logistic Regression in a picture

Choose weights the give probabilities close to 1.0 to each of the correct labels

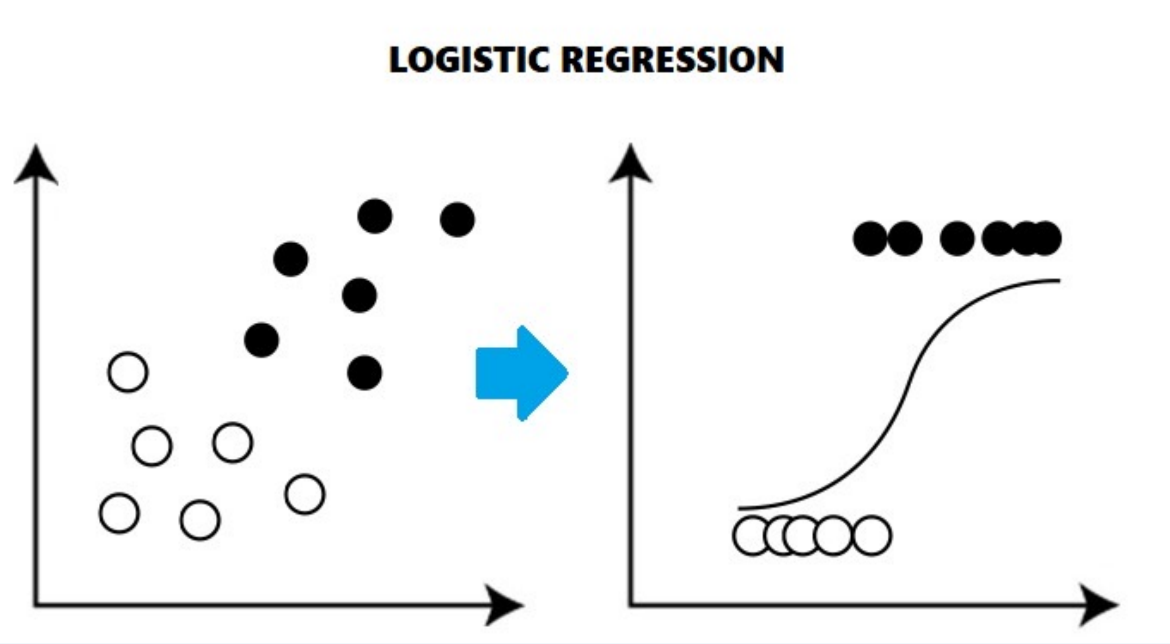

Image from https://www.analyticsvidhya.com/blog/2021/04/beginners-guide-to-logistic-regression-using-python/

Learning in Logistic Regression

- **Goal:** choose weights the give the “least error”

$$Loss(y) = -\sum_{n=1}^{N} 1\{y = n\} \log p(y = n \mid x_n)$$

- Choose weights the give probabilities close to 1.0 to each of the correct labels
  - But how?!?
    - By using Calculus

Slide from Nate Chambers

Learning in Logistic Regression

- **Gradient descent**: how we update the weights

1. Find the slope of each weight w_i
   - By taking the partial derivative (Calculus III)
2. Move in the direction of the slope
3. Update all the weights
4. Recalculate the loss function based on new weights
5. Repeat

Slide from Nate Chambers

Learning in Logistic Regression

- **Gradient descent**: how we update the weights

Hand waving descriptions:

1. Randomly initialize weights
2. Compute probabilities for all training examples
3. Jiggle the weights up and down based on mistakes
4. Repeat

Slide from Nate Chambers

Summary of Logistic Regression

- Optimizes $\mathrm{P}(Y \mid X)$ directly
- Define the features
- Learn a vector of weights for each label $y \epsilon Y$
  - Gradient descent, update weights based on error
- Multiple feature vector by weight vector
- Output is $\mathrm{P}(Y = y \mid X)$ after normalizing
- Choose the most probable Y

Slide from Nate Chambers