Lecture 6
Logistic Regression for Text Classification

$\boxed{9/21/21 \quad \text{Slides}}$

CS 490A, Fall 2021

https://people.cs.umass.edu/~brenocon/cs490a_f21/

Laure Thompson and Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

[Many slides from Ari Kobren

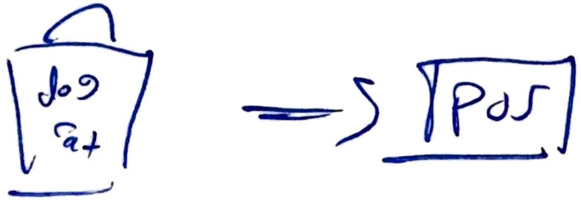
- HW1!
- Zoom OH demand? Piazza Poll!
- Accept exercises late? Yes, for check-minus: see the "Grading and Policies" page. Please let us know if you can't upload to Gradescope.

Linear classification models

- The foundational model for machine learning-based NLP!

- Examples
  - The humble "keyword count" classifier (no ML)
  - Naive Bayes ("generative" ML)

- Today: **Logistic Regression**
  - "discriminative" ML
  - allows for features
  - used within more complex models (neural networks)



bag of words

BOW linear model for text classif.

- Problem: classify doc d into one of $k=1..K$ classes

- Parameters: For each class $k \in 1..K$ and word type $w \in V$, there is a *word weight*

$$\beta_{w,k} \in \mathbb{R}$$

"association b/w w and k"

$$\begin{array}{cc} & K \\ V & \begin{bmatrix} \quad \end{bmatrix} \end{array}$$

$\hookrightarrow \vec{\beta}_{k,\cdot} \in \mathbb{R}^{V}$

- Representation: bag-of-words vector of doc d's word counts

$x \in \mathbb{R}^{|V|}$

$x_w = \#(\text{word } w \text{ in doc})$

- Prediction rule: choose class y with highest score

$$\forall k: \quad \text{score}_k = \vec{\beta}_k^{T} \vec{x} = \sum_{w=1}^{V} \beta_{k,w} x_w$$

$$\hat{y} = \arg\max_{k \in 1..K} \{ \text{score}_k \}$$

Have we seen a text classification model that can be seen as an instance of this??

Lexicon Counting

$$n_{pos} = \#(\text{words in POS dict})$$

$$n_{neg} = \#(\text{words in NEG dict})$$

$$\hat{y} = \begin{cases} \text{pos} & \text{if } n_{pos} > n_{neg} \\ \text{neg} & \text{if } n_{pos} \leq n_{neg} \end{cases}$$

---

as Linear Model

$K = 2$ classes

$$\beta_{pos,w} = \begin{cases} 1 & \text{if } w \in \text{POS dict} \\ 0 & \text{otherwise} \end{cases}$$

$$\beta_{neg,w} = \begin{cases} 1 & \text{if } w \in \text{NEG dict} \\ 0 & \text{o/w} \end{cases}$$

score_{pos} = count of pos words

Naive Bayes

$$\log p(y|x) \propto \left[\sum_{i=1}^{N_{tok}} \log p(w_i|y)\right] + \log p(y) \quad \left[\text{Token form}\right]$$

$$\propto \left[\sum_{w=1}^{V} \underbrace{x_w}_{\vec{x}} \underbrace{\log p(w|y)}_{\beta_{w,y}}\right] + \underbrace{\log p(y)}_{bias_y}$$

Features! Features! Features!

- Input document **d** (a string...)
- Engineer a <u>feature function</u>, f(d), to generate <u>feature vector</u> ***x***

$$f(d) \longrightarrow x$$

f(d) = (
Count of "happy",
(Count of "happy") / (Length of doc),
log(1 + count of "happy"),
Count of "not happy",
Count of words in my pre-specified word list, "positive words according to my favorite psychological theory",
Count of several different "happy" emoticons
Count of "of the",
Length of document,
...
)

Typically these use <u>feature templates</u>:
Generate many features at once

for each word w:
- ${w}_count
- ${w}_is_count_nonzero
- NOT_${w}_bigram_count

- Not just word counts. Anything that might be useful!
- <u>Feature engineering</u>: when you spend a lot of trying and testing new features. Very important!! This is a place to put linguistics in, or just common sense about your data.

- Special chars? Caps?
- Punct! . . .

Negation

Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA).
Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79—86.

Add NOT_ to every word between negation and following punctuation:

```
didn't like this movie , but I
```

```
didn't NOT_like NOT_this NOT_movie but I
```

[Slide: SLP3]

Classification: LogReg (I)

Binary

- compute **features** (xs)

$$x_i \quad = (\text{count “nigerian”, count “prince”, count “nigerian prince”})$$

- given **weights** (betas)

$$\beta \quad = (-1.0, \quad -1.0, \quad 4.0)$$

$N_{feat} = 3$

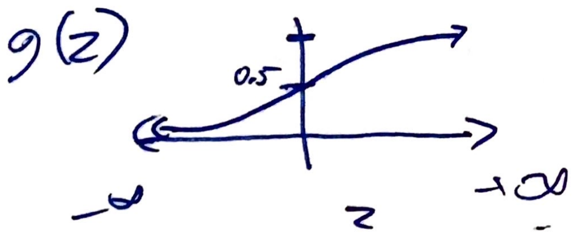
Classification: LogReg (II)

- Compute the **dot product**

$$z = \beta^T x$$

- Compute the **logistic function** for the label probability

$$P(y=1|x) = \underbrace{g(z) = \frac{1}{1+e^{-z}}}_{\text{Logistic function}} = \frac{e^{z}}{1+e^{z}}$$



LogReg Exercise

features: (count “nigerian”, count “prince”, count “nigerian prince”)

x = (1, 1, 1)

β = (-1.0, -1.0, 4.0)

$z = \beta^T x = -1 - 1 + 4 = 2$

$P(y=1 \mid x) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-2}} = .88$

Learning Weights

So let's try to maximize probability of the entire dataset - **maximum likelihood estimation**

$$\beta^{MLE} = \arg\max_{\beta} \log P(y_0, \ldots, y_n | \mathbf{x_0}, \ldots, \mathbf{x_n}; \beta)$$

$$= \arg\max_{\beta} \sum_{d=1}^{N_{doc}} \log \underbrace{P(y_d | x_d ; \beta)}_{\text{logReg } g(\beta^T x_d)}$$

Learning Weights

We know:

$$g(z) = \frac{1}{1 + e^{-z}} \qquad P(y = 1 \mid x) = g\left(\sum_{j=1}^{\text{Nfeat}} \beta_j x_{ij}\right)$$

So let's try to maximize probability of the entire dataset - **maximum likelihood estimation**

Learning Weights

$$
\begin{array}{ccccc}
x_{00} & x_{01} & \cdots & x_{0m} & y_0 \\
x_{10} & x_{11} & \cdots & x_{1m} & y_1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x_{n0} & x_{n1} & \cdots & x_{nm} & y_n
\end{array}
$$

n examples; xs - features; ys - class

Multiclass Logistic Regression

- Binary logreg: let x be a feature vector for the doc, and y either 0 or 1

$$p(y=1|x,\beta) = \frac{\exp(\beta^\mathsf{T} x)}{1+\exp(\beta^\mathsf{T} x)}$$

β is a weight vector across the *x* features.

- Multiclass logistic regression: weight vector for each class

$$\vec{\beta}_k = [\beta_{k,1}, \beta_{k,2}, \dots \beta_{k,V}]$$

- Prediction: dot product for each class

$$z_k = \vec{\beta}_k^\mathsf{T} \vec{x} \quad \in \mathbb{R} \qquad [z_1, z_2, \dots z_K]$$

- Predicted probabilities: apply the *softmax function* to normalize

$$P(y=k|x) = P_k = \frac{e^{z_k}}{\boxed{\sum_{j=1}^{K} e^{z_k}}} \qquad [e^{z_1}, e^{z_2}, \dots e^{z_K}]$$

Gradient ascent learning

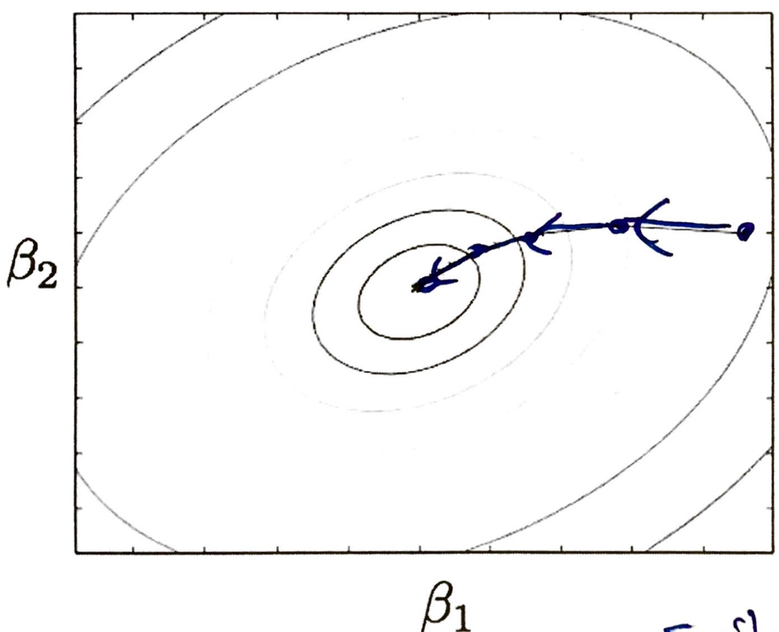
- Follow direction of *steepest ascent.* Iterate

$$\beta^{(new)} = \beta^{(old)} + \eta \frac{\partial \ell}{\partial \beta}$$

$$\ell(\beta) = \sum_{i=0}^{|X|} \log P(y_i | \mathbf{x_i}; \beta)$$

ℓ: Training set log-likelihood

η: Step size (a.k.a. learning rate)



$\left(\frac{\partial \ell}{\partial \beta_1}, \ldots, \frac{\partial \ell}{\partial \beta_J} \right)$: Gradient vector (vector of per-element derivatives)

This is a generic optimization technique. Not specific to logistic regression! Finds the maximizer of any function where you can compute the gradient.

scikit-learn "LogisticRegression"

Name: ____________________________________

Exercise 3, in-class 9/16/21 – UMass CS 490A
Naïve Bayes example (from J&M exercise 4.2)
Turn in via Gradescope after class or by next Monday

We are given the following short movie reviews, each labeled with a genre (comedy or action):

review	label
fun, couple, love, love	comedy
fast, furious, shoot	action
couple, fly, fast, fun, fun	comedy
furious, shoot, shoot, fun	action
fly, fast, shoot, love	action

Now, we are given a new review:
fast, couple, shoot, fly

Compute the most likely class for this review. Assume a naive Bayes classifier and use add-1 smoothing for the likelihoods.

Need V!

V=7

fun
couple
love
fast
furious
shoot
fly

Name: ____

Exercise 3, in-class 9/16/21 – UMass CS 490A
Naïve Bayes example (from J&M exercise 4.2)
Turn in via Gradescope after class or by next Monday

We are given the following short movie reviews, each labeled with a genre (comedy or action):

review	label
fun, couple, love, love	comedy
fast, furious, shoot	action
couple, fly, fast, fun, fun	comedy
furious, shoot, shoot, fun	action
fly, fast, shoot, love	action

Now, we are given a new review:

fast, couple, shoot, fly $= x$

Compute the most likely class for this review. Assume a naive Bayes classifier and use add-1 smoothing for the likelihoods.

$$P(y = \text{"}c\text{"} \mid x) = P(c) \cdot p(\text{fast} \mid c) \cdot p(\text{couple} \mid c)\, p(\text{sh.} \mid c)\, p(\text{fly} \mid c)$$

$$= \frac{2}{5} \cdot \frac{1+1}{9+V} \cdot \frac{2+1}{9+V} \cdot \ldots$$

$$P(\text{action} \mid x) = \frac{3}{5} \cdot \ldots$$