CS 6501 Natural Language Processing

Logistic Regression

Yangfeng Ji

September 12, 2019

Department of Computer Science
University of Virginia

Overview

1. Naive Bayes Classifier, revisited
2. Logistic Regression
3. Classification Evaluation
4. Example
5. Building Better LR Models
6. Discriminative vs. Generative Classifiers

Naive Bayes Classifier, revisited

An Alternative View

- Proir probability: $p(y; \boldsymbol{\theta}) = \theta_y$
- Likelihood: $p(\boldsymbol{x} \mid y; \boldsymbol{\theta}) \propto \prod_{i=1}^{V} {\theta_{i,y}}^{x_i}$

$$p(\boldsymbol{x}, y; \boldsymbol{\theta}) \propto \theta_y \cdot \prod_{i=1}^{V} {\theta_{i,y}}^{x_i} \tag{1}$$

Or

$$\log p(\boldsymbol{x}, y; \boldsymbol{\theta}) \propto \log \theta_y + \sum_{i=1}^{V} (x_i \cdot \log \theta_{i,y}) \tag{2}$$

An Alternative View (II)

$$\log p(\boldsymbol{x}, y; \boldsymbol{\theta}) \propto \log \theta_y + \sum_{i=1}^{V} (x_i \cdot \log \theta_{i,y}) \quad (3)$$

$$= b_y + \boldsymbol{w}_y^\top \boldsymbol{x} \quad (4)$$

where

$$\begin{aligned} b_y &= \log \theta_y \\ \boldsymbol{w}_y^\top &= [\log \theta_{1,y}, \ldots, \log \theta_{V,y}] \\ \boldsymbol{x}^\top &= [x_1, \ldots, x_V] \end{aligned}$$

It's a linear classifier (with respect to $\boldsymbol{x}$)

Logistic Regression

Log-linear Models

Directly modeling a linear classifier as

$$\log p(y \mid \boldsymbol{x}) \propto \boldsymbol{w}_y^\top \boldsymbol{x} + b_y \tag{5}$$

with

- $\boldsymbol{x} \in \mathbb{N}^V$: bag-of-words representation
- $\boldsymbol{w}_y \in \mathbb{R}^V$: classification weights associated with label y
- $b_y \in \mathbb{R}$: classification bias associated with label y

Probabilistic Form

Given

$$\log P(y \mid \boldsymbol{x}) \propto \boldsymbol{w}_y^\top \boldsymbol{x} + b_y, \tag{6}$$

the probabilistic form is

$$P(y \mid \boldsymbol{x}) \propto \exp(\boldsymbol{w}_y^\top \boldsymbol{x} + b_y) \tag{7}$$

or

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x} + b_y)}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x} + b_{y'})} \tag{8}$$

Alternative Form

Rewrite $\boldsymbol{x}$ and $\boldsymbol{w}$ as

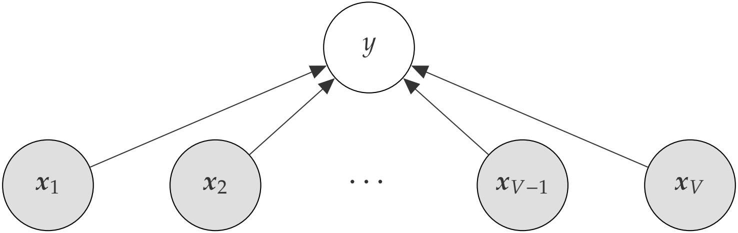
- $\boldsymbol{x}^\top = [x_1, x_2, \cdots, x_V, 1]$
- $\boldsymbol{w}^\top = [w_1, w_2, \cdots, w_V, b_y]$

then,

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x})} \tag{9}$$

Probabilistic Graphical Models

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x})} \tag{10}$$



Binary Classifier

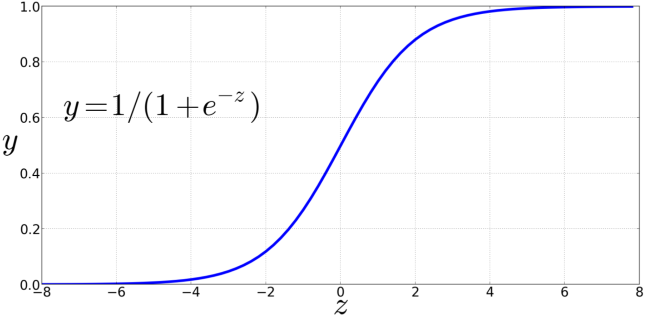
Assume $\mathcal{Y} = \{\textsc{neg}, \textsc{pos}\}$, then the corresponding logistic regression classifier with $Y = \textsc{pos}$ is

$$
\begin{aligned}
P(\textsc{pos} \mid \boldsymbol{x}) &= \frac{\exp(\boldsymbol{w}_{\text{POS}}^{\top}\boldsymbol{x})}{\exp(\boldsymbol{w}_{\text{POS}}^{\top}\boldsymbol{x}) + \exp(\boldsymbol{w}_{\text{NEG}}^{\top}\boldsymbol{x})} && (11)\\
&= \frac{1}{1 + \frac{\exp(\boldsymbol{w}_{\text{NEG}}^{\top}\boldsymbol{x})}{\exp(\boldsymbol{w}_{\text{POS}}^{\top}\boldsymbol{x})}} && (12)\\
&= \frac{1}{1 + \exp(-(\boldsymbol{w}_{\text{POS}} - \boldsymbol{w}_{\text{NEG}})^{\top}\boldsymbol{x})} && (13)\\
&= \frac{1}{1 + \exp(-\boldsymbol{w}^{\top}\boldsymbol{x})} && (14)
\end{aligned}
$$

where $\boldsymbol{w} = \boldsymbol{w}_{\text{POS}} - \boldsymbol{w}_{\text{NEG}}$

Sigmoid Function

$$y = \frac{1}{1 + \exp(-z)} \tag{15}$$



$z \in (-\infty, \infty), y \in (-1, 1)$

Two Questions

... of using a logistic regression classifier

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x})} \tag{16}$$

- How to learn the parameters $\{\boldsymbol{w}_y\}_{y \in \mathcal{Y}}$?
- Can $\boldsymbol{x}$ be better than the bag-of-words representation?

(Log)-likelihood Function

With a collection of training examples $\{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, the likelihood function of $\{\boldsymbol{w}_y\}_{y \in \mathcal{Y}}$ is

$$L(\{\boldsymbol{w}_y\}) = \prod_{n=1}^{N} P(y^{(n)} \mid \boldsymbol{x}^{(n)}) \tag{17}$$

and the log-likelihood function is

$$\ell(\{\boldsymbol{w}_y\}) = \sum_{n=1}^{N} \log P(y^{(n)} \mid \boldsymbol{x}^{(n)}) \tag{18}$$

Log-likelihood Function (II)

With

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x})} \tag{19}$$

the log-likelihood function is

$$\begin{aligned} \ell(\{\boldsymbol{w}_y\}) &= \sum_{n=1}^{N} \log P(y^{(n)} \mid \boldsymbol{x}^{(n)}) && (20) \\ &= \sum_{n=1}^{N} \Big\{ \boldsymbol{w}_{y^{(n)}}^\top \boldsymbol{x}^{(n)} - \log \sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x}^{(n)}) \Big\} && (21) \end{aligned}$$

Given the training examples $\{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, $\ell(\{\boldsymbol{w}_y\})$ is a function of $\{\boldsymbol{w}_y\}$.

Optimization with Gradient

MLE is equivalent to maximize the Negative Log-Likelihood (NLL) as

$$\begin{aligned} \mathrm{NLL}(\{\boldsymbol{w}_y\}) &= -\ell(\{\boldsymbol{w}_y\}) \\ &= \sum_{n=1}^{N} \Big\{ -\boldsymbol{w}_{y^{(n)}}^{\top} \boldsymbol{x}^{(n)} + \log \sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^{\top} \boldsymbol{x}) \Big\} \end{aligned}$$

then

$$\boldsymbol{w}_y \leftarrow \boldsymbol{w}_y - \eta \cdot \frac{\partial \mathrm{NLL}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y}, \quad \forall y \in \mathcal{Y} \tag{22}$$

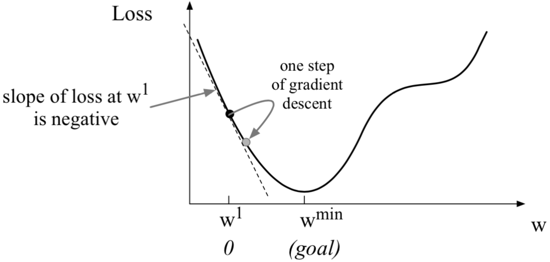
where η is called **learning rate**.

Optimization with Gradient (II)

Two questions answered by the update equation

- which direction?
- how far it should go?

$$\boldsymbol{w}_y \leftarrow \boldsymbol{w}_y - \eta \cdot \frac{\partial \text{NLL}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y} \tag{23}$$



[Jurafsky and Martin, 2019]

Training Procedure

Steps for parameter estimation, given the current parameter $\{\boldsymbol{w}_y\}$

1. Compute the derivative

$$\frac{\partial \text{NLL}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y}, \quad \forall y \in \mathcal{Y}$$

2. Update parameters with

$$\boldsymbol{w}_y \leftarrow \boldsymbol{w}_y - \eta \cdot \frac{\partial \text{NLL}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y}, \quad \forall y \in \mathcal{Y}$$

3. If not done, retrun to step 1

Classification Evaluation

A Development Set

- Training set $\mathcal{T} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{i=1}^{N}$
- Development set $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$
- Test set $\mathcal{U} = \{(\boldsymbol{x}^{(j)}, y^{(j)})\}_{j=1}^{L}$

Cross-validation

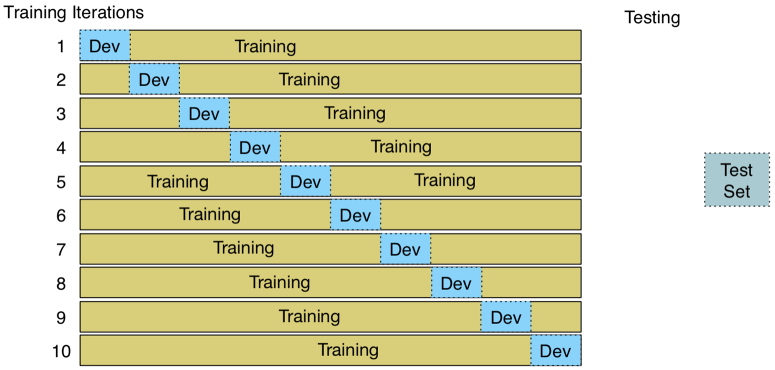


Figure: 10-fold cross validation.

Evaluation Measurements

- Accuracy
- Precision, recall, and F-measure

[Eisenstein, 2018, Sec 4.4]

Accuracy

Given M examples, with $y^{(i)}$ is the ground-truth label of the i-th example, and $\hat{y}^{(i)}$ is the predicted label

$$\textsc{acc}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{M}\delta(y^{(i)}, \hat{y}^{(i)}) \tag{24}$$

δ function is defined as

$$\delta(y, \hat{y}) = \begin{cases} 1 & y = \hat{y} \\ 0 & y \neq \hat{y} \end{cases} \tag{25}$$

Confusion Matrix

		Ground truth	
		POSITIVE	NEGATIVE
Prediction	POSITIVE	True Positive (TP)	False Positive (FP)
	NEGATIVE	False Negative (FN)	True Negative (TN)

Accuracy:

$$\mathrm{acc}(y, \hat{y}) = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{26}$$

Recall, Precision and F Measure

		Ground truth	
		POSITIVE	NEGATIVE
Prediction	POSITIVE	True Positive (TP)	False Positive (FP)
	NEGATIVE	False Negative (FN)	True Negative (TN)

Recall:

$$r(y, \hat{y}) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{27}$$

Precision:

$$p(y, \hat{y}) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{28}$$

F measure:

$$F(y, \hat{y}) = \frac{2 \cdot p \cdot r}{p + r} \tag{29}$$

Example: Balanced case

		Ground truth	
		POSITIVE	NEGATIVE
Prediction	POSITIVE	480	30
	NEGATIVE	20	470

- Accuracy: $\text{acc} = \frac{480+470}{480+20+30+470} = 0.95$
- Precision: $p = \frac{480}{480+30} \approx 0.94$
- Recall: $r = \frac{480}{480+20} \approx 0.96$
- F-measure: $F = \frac{2\times0.94\times0.96}{0.94+0.96} \approx 0.95$

Example: Unbalanced case

		Ground truth	
		POSITIVE	NEGATIVE
Prediction	POSITIVE	80	30
	NEGATIVE	20	870

- Accuracy: $\text{acc} = \frac{80+870}{80+20+30+870} = 0.95$
- Precision: $p = \frac{80}{80+30} \approx 0.73$
- Recall: $r = \frac{80}{80+20} = 0.80$
- F-measure: $F = \frac{2\times 0.94\times 0.96}{0.94+0.96} \approx 0.76$

Example

Example Dataset

A subset of the Yelp Dataset

`https://www.yelp.com/dataset/challenge`

	Training	Development	Test
Documents	40K	5K	5K
Words	4.7M	0.5M	0.6M

- 5 classes (user rating from 1 to 5)
- Code available on
  `https://github.com/jiyfeng/textclassification`

Building BoW Representations

Sklearn function

- `sklearn.feature_extraction.text.CountVectorizer`
- `sklearn.linear_model.LogisticRegression`

- Given a collection of texts, it will build a vocab and also convert all texts into numeric vectors
- Classification accuracy on the development data is 61.4%

How Far We Can Go with BoW?

Tricks to reduce vocab size

- remove punctuation (default)
- lowercase ($\nearrow$)
- remove low-frequency words ($\nearrow$)
- remove high-frequency words ($\searrow$)
- replace numbers with a special token

Comments

- Not always helpful (these are empirical tricks)
- Not always the case (it depends on the data/domain)

Interpretability

Weights learned from training data

$$
\begin{array}{ccc}
\text{Vocab} & w_{\text{rating}=1} & w_{\text{rating}=5} \\
\left\{\begin{array}{c} \textsc{super} \\ \cdots \\ \textsc{quick} \\ \textsc{food} \\ \textsc{friendly} \\ \textsc{eat} \\ \cdots \\ \textsc{delicious} \end{array}\right\}
&
\begin{bmatrix} 0.33 \\ \cdots \\ -1.26 \\ 0.08 \\ -2.57 \\ -0.47 \\ \cdots \\ -3.60 \end{bmatrix}
&
\begin{bmatrix} -0.09 \\ \cdots \\ -0.01 \\ -0.09 \\ 0.16 \\ 0.00 \\ \cdots \\ 0.64 \end{bmatrix}
\end{array}
$$

Interpretability (II)

Top features

rating = 5	rating = 1
exceptional	worst
incredible	joke
phenomenal	disgusted
body	unprofessional
regret	garbage
worried	disgusting
skeptical	*luck*
hesitate	pathetic
happier	apologies
mike	horrible

Building Better LR Models

Ways to Improve LR Models

- A better text representation $\boldsymbol{x}$
- Regularization on $\boldsymbol{w}$

A Generic LR Model

Replace $\boldsymbol{x}$ in the following equation

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{x})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{x})} \tag{30}$$

with a generic feature function $\boldsymbol{f}(\boldsymbol{x})$

$$P(y \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_y^\top \boldsymbol{f}(\boldsymbol{x}))}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}_{y'}^\top \boldsymbol{f}(\boldsymbol{x}))} \tag{31}$$

Rich Features: bi-grams

Combine words together to form high-order features

- Uni-grams:

$$\{\textsc{fast}, \textsc{slow}, \textsc{super}\}$$

- Bi-grams:

$$\{\textsc{super fast}, \textsc{super slow}\}$$

Extended feature set

$$\{\textsc{fast}, \textsc{slow}, \textsc{super}, \textsc{super fast}, \textsc{super slow}\}$$

Other Features

For example: other rich features from $f(x)$ for sentiment analysis

- the length of the text
- the number of sentences
- the number of positive words
- the number of negative words
- the number of pronouns

Combining these features with the uni- and bi-gram features

Performance

Sklearn function

`sklearn.feature_extraction.text.CountVectorizer`

with `ngram_range=(1,2)`

- Even larger vocab size: from 60K to 1M
- Performance change: from 61.4% to 62.4% on the dev data

Problems with Large Feature Set

- Overlap among features

Example

{FAST, SLOW, SUPER, SUPER FAST, SUPER SLOW}

- Learning in high dimensional space — overfitting

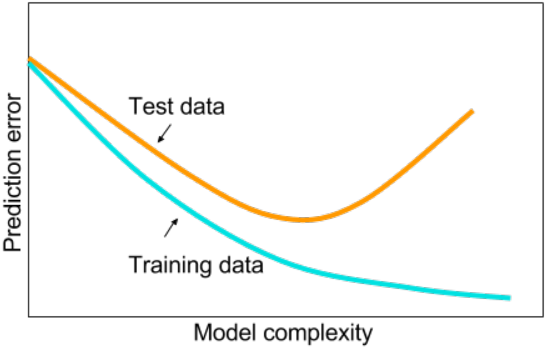
Example

62.4% on dev data

vs.

$\sim$ 100% on training data

Overfitting



[Abu-Mostafa et al., 2012]

Ways to Improve LR Models

- ✓ A better text representation $\boldsymbol{x}$
- ► Regularization on $\boldsymbol{w}$

L_2 Regularization

Add an additional constraint on $\{\boldsymbol{w}_y\}$

$$\ell_{L_2}(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log P(y^{(n)}|\boldsymbol{x}^{(n)}; \boldsymbol{w}) + \frac{\lambda}{2}\sum_{y} \|\boldsymbol{w}_y\|_2^2 \tag{32}$$

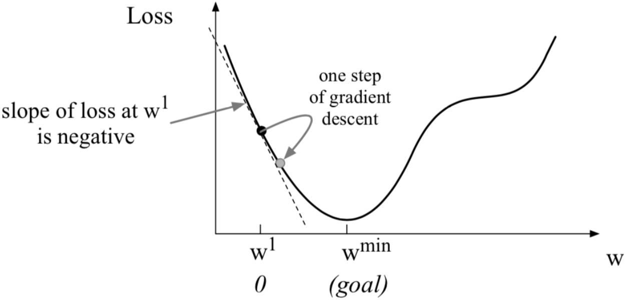
The gradient of the new loss function

$$\frac{\partial \ell_{L_2}(\{\boldsymbol{w}_y\})}{\partial \boldsymbol{w}_y} = -\sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{w}_y} \log P(y^{(n)}|\boldsymbol{x}^{(n)}; \boldsymbol{w}) + \lambda \boldsymbol{w}_y \tag{33}$$

[Eisenstein, 2018, Sec. 2.4.1]

L_2 Regularization (Cont.)

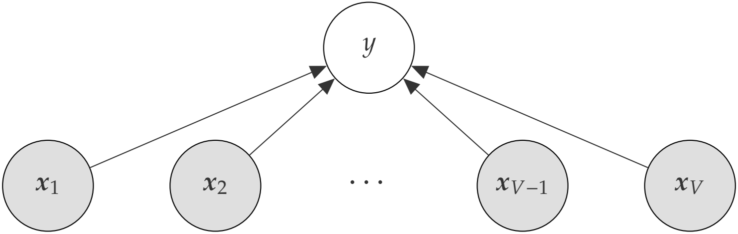
L_2 Regularization introduces a trade-off between the likelihood function and the norm of $\{w_y\}$



Discriminative vs. Generative Classifiers

Discriminative Models

Logistic models

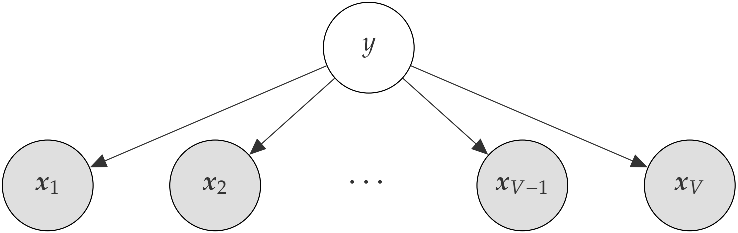


- No conditional independence assumption
  - Rich feature set
- Performance depends the choice of optimization methods

[Jurafsky and Martin, 2019]

Generative Models

Naive Bayes classifiers



- Conditional independence
- Easy to implement
- Works better on *small* training sets or *short* texts

[Jurafsky and Martin, 2019]

Reference

Abu-Mostafa, Y. S., Magdon-Ismail, M., and Lin, H.-T. (2012).
Learning from data, volume 4.
AMLBook New York, NY, USA:.

Eisenstein, J. (2018).
Natural Language Processing.
MIT Press.

Jurafsky, D. and Martin, J. (2019).
Speech and language processing.