Text Classification

He He

New York University

September 15, 2020

Text classification

- Application
  - Spam filter, sentiment classification, document classification, ...
  - Textual entailment, paraphrase detection, ...

Table of Contents

Intuition

Example: sentiment classification for movie reviews

> *Bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life, such as "Teachers". My 35 years in the teaching profession lead me to believe that Bromwell High's satire is much closer to reality than is "Teachers". The scramble to survive financially, the insightful students who can see right through their pathetic teachers' pomp, the pettiness of the whole situation, all remind me of the schools I knew and their students...*
>
> Label: positive

Intuition

Example: sentiment classification for movie reviews

> *Bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life, such as "Teachers". My 35 years in the teaching profession lead me to believe that Bromwell High's satire is much closer to reality than is "Teachers". The scramble to survive financially, the insightful students who can see right through their pathetic teachers' pomp, the pettiness of the whole situation, all remind me of the schools I knew and their students...*
>
> Label: positive

Idea: assign a score of positivity/negativity for each word

[demo]

Side note: tokenization

Splitting a string of text s to a sequence of **tokens** $[x_1, \ldots x_n]$. didn 't

did not

Ph.D.

Language-specific solutions

- Regular expression: "I didn't watch the movie". $\rightarrow$ ["I", "did", "n't", "watch", "the", "movie", "."]
- Dictionary / sequence labeler: "我没有去看电影。" $\rightarrow$ ["我", "没有", "去", "看", "电影", "。"]

Side note: tokenization

Splitting a string of text s to a sequence of **tokens** $[x_1, \ldots x_n]$.

Language-specific solutions

- Regular expression: "I didn't watch the movie". $\rightarrow$ ["I", "did", "n't", "watch", "the", "movie", "."]
- Dictionary / sequence labeler: "我没有去看电影。" $\rightarrow$ ["我", "没有", "去", "看", "电影", "。"]

General solution: don't split by words

- Characters: ["u", "n", "a", "f", "f", "a", "b", "l", "e"]
- Subword (e.g. byte pair encoding): ["un", "aff", "able#"]

$$\begin{tikzcd}
a \arrow[dr, dash] & & b \arrow[dl, dash] & c & d \arrow[ddll, dash, bend left] & \rightarrow \\
 & ab \arrow[dr, dash, bend right] & & & & \\
 & & & abd & \leq k &
\end{tikzcd}$$

Problem formulation

- Input: a sequence of tokens $X = (X_1, \ldots X_n)$ where $X_i \in \mathcal{V}$.
- Output: label $Y \in \{0, 1\}$.
- Probabilistic model:

$$f(x) = \begin{cases} 1 & \text{if } p_\theta(y \mid x) > 0.5 \\ 0 & \text{otherwise} \end{cases},$$

  where p_θ is a distribution parametrized by $\theta \in \Theta$.
- Question: how to choose p_θ? (**inductive bias**)

Model $p(y \mid x)$

How to write a review:

1. Decide the sentiment by flipping a coin $p(y)$
2. Generate word ~~sequentially~~ independently conditioned on the sentiment $p(x \mid y)$

Bayes rule

$$\underline{p(y \mid x)} = \frac{p(x \mid y)p(y)}{p(x)} = \frac{p(x \mid y)p(y)}{\sum_{y \in \mathcal{Y}} p(x \mid y)p(y)}$$

- $p(y) = \mathrm{Ber}(\alpha)$
- $p(x \mid y) = \prod_{i=1}^{n} \underline{p(x_i \mid y)}$

$|V|$ categorical$(\theta_1, \ldots, \theta_{|V|})$ $\sum_{i=1}^{|V|} \theta_i = 1$

Naive Bayes models

Naive Bayes assumption

The input features are **conditionally independent** given the label:

$$p(x \mid y) = \prod_{i=1}^{n} p(x_i \mid y) .$$

A strong assumption, but works surprisingly well in practice.

Note: $p(x_i \mid y)$ doesn't have to be a categorical distribution.

Inference:

$$y = \arg\max_{y \in Y} P(y \mid x)$$

$$= \arg\max_{y \in Y} \frac{P(x \mid y) \cancel{P(y)}}{\cancel{P(x)}}$$

$$= \arg\max_{y \in Y} \prod_{i=1}^{n} P(x_i \mid y) = \arg\max_{y \in Y} \sum_{i=1}^{n} \log P(x_i \mid y)$$

Maximum likelihood estimation

Task: estimate parameters θ of a distribution $p(y; \underline{\theta})$ given i.i.d. samples $D = (y_1, \ldots, y_N)$ from the distribution.

Goal: find the parameters that make the observed data most <u>probable.</u>

Likelihood function of θ given D:

$$L(\theta; D) \overset{\text{def}}{=} p(D; \theta) = \prod_{i=1}^{N} p(y_i; \theta) .$$

Maximum likelihood estimator:

$$\hat{\theta} = \underset{\theta \in \Theta}{\arg\max}\ L(\theta; D)$$

$$= \underset{\theta \in \Theta}{\arg\max} \sum_{i=1}^{N} \log p(y_i; \theta)$$

optimization

MLE and ERM

ERM:

$$\min \sum_{i=1}^{N} L(x^{(i)}, y^{(i)}, \theta)$$

MLE:

$$\max \sum_{i=1}^{N} \log p(y^{(i)} \mid x^{(i)}; \theta)$$

Loss func in MLE:

$$L_{NLL}(x^{(i)}, y^{(i)}, \theta) \overset{def}{=} -\log p(y^{(i)} \mid x^{(i)}; \theta)$$

(NLL)

MLE for our Naive Bayes model

$$\ell(\theta) = \sum_{i=1}^{N} \log p(x^{(i)}, y^{(i)}; \theta) \rightarrow \underline{\theta_{w,1}} \quad |V| \times 2$$

$$= \sum_{i=1}^{N} \log p(x^{(i)} \mid y^{(i)}; \theta)\, p(y^{(i)}; \underline{\alpha})$$

$$p(y^{(i)}; \alpha) = \begin{cases} \alpha & \text{if } y^{(i)} = 1 \\ 1-\alpha & \text{if } y^{(i)} = 0 \end{cases}$$

$$= \mathbb{1}(y^{(i)}=1)\alpha + \mathbb{1}(y^{(i)}=0)(1-\alpha)$$

$$\frac{\partial \ell}{\partial \alpha} = \sum_{i=1}^{N} \log \left[\mathbb{1}(y^{(i)}=1)\alpha + \mathbb{1}(y^{(i)}=0)(1-\alpha)\right]$$

$$= \sum_{i=1}^{N} \mathbb{1}(y^{(i)}=1)\log\alpha + \mathbb{1}(y^{(i)}=0)\overset{\log}{}(1-\alpha)$$

$$= \sum_{i=1}^{N} \frac{1}{\alpha}\mathbb{1}(y^{(i)}=1) + \frac{1}{1-\alpha}\mathbb{1}(y^{(i)}=0) = 0$$

$$\alpha_{MLE} = \sum_{i=1}^{N} \mathbb{1}(y^{(i)}=1) / N$$ prop of pos. ex.

MLE for our Naive Bayes model

MLE solution:

$$\begin{aligned}
\text{count}(w, y) &\overset{\text{def}}{=} \text{frequency of } w \text{ in documents with label } y \\
p(w \mid y) &= \frac{\text{count}(w, y)}{\sum_{w \in \mathcal{V}} \text{count}(w, y)} \\
p(y = k) &= \frac{\sum_{i=1}^{N} \mathbb{I}\left(y^{(i)} = k\right)}{N}
\end{aligned}$$

MLE for our Naive Bayes model

MLE solution:

$$\text{count}(w, y) \overset{\text{def}}{=} \text{frequency of } w \text{ in documents with label } y$$

$$p(w \mid y) = \frac{\text{count}(w, y)}{\sum_{w \in \mathcal{V}} \text{count}(w, y)}$$

$$p(y = k) = \frac{\sum_{i=1}^{N} \mathbb{I}\left(y^{(i)} = k\right)}{N}$$

Smoothing: reserve probability mass for unseen words

$$p(w \mid y) = \frac{\alpha + \text{count}(w, y)}{\sum_{w \in \mathcal{V}} \text{count}(w, y) + \alpha|\mathcal{V}|}$$

Laplace smoothing: $\alpha = 1$

Feature design

Naive Bayes doesn't have to use single words as features

- Lexicons, e.g. LIWC.
- Task-specific features, e.g. is the subject all caps.
- Bytes and characters, e.g. used in language ID detection.

Table of Contents

Discriminative models

Key idea: directly model the conditional distribution $p(y \mid x)$

generative models	discriminative models
$p(x, y)$	$p(y \mid x)$
more assumption	-
generative story	feature extractor

Model $p(y \mid x)$

How to model $p(y \mid x)$?

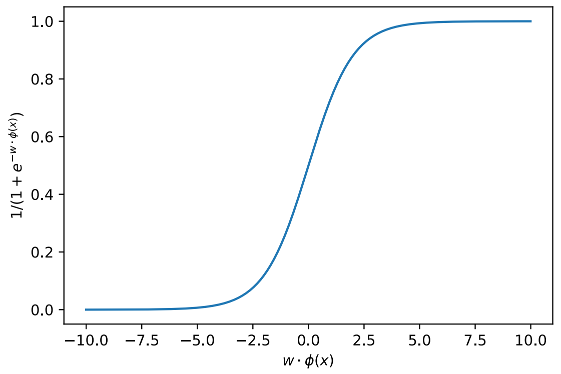
Bernoulli: $p(y \mid x) = \alpha^{y}(1-\alpha)^{1-y}$

Bring in x: $p(y \mid x) = \underline{h(x)^{y}(1-h(x))^{1-y}}$ $h(x) \in [0,1]$

Linear predictor: $f(x) = \underbrace{w \cdot x + b}_{\text{score}}$ $w \in \mathbb{R}^{d}$ $f(x) \in \mathbb{R}$

Logistic regression

Map $w \cdot \phi(x) \in \mathbb{R}$ to a probability by the **logistic function**



$- w \cdot \phi(x)$

$$p(y = 1 \mid x; w) = \frac{1}{1 + e^{-\underbrace{w \cdot \phi(x)}_{\text{score}}}} \quad (y \in \{0, 1\})$$

$$p(y = k \mid x; w) = \frac{e^{\underline{w_k \cdot \phi(x)}}}{\sum_{i \in \mathcal{Y}} e^{w_i \cdot \phi(x)}} \quad (y \in \{1, \ldots, K\})$$

$\underline{\text{soft argmax}}$

MLE for logistic regression

$$\ell(\theta) = \sum_{i=1}^{N} \log \frac{1}{1+e^{-w\cdot\phi(x)}}$$

$$\min \sum_{i=1}^{N} \log\left(1+e^{-\underline{w\cdot\phi(x)}}\right)$$

$$\log\left(1+e^{-z}\right) \qquad (\text{plot} \,/\, f''(z) \geq 0)$$

BoW representation

Feature extractor: $\phi\colon \mathcal{V}^n \to \mathbb{R}^d$.

Idea: a sentence is the "sum" of words.

Example:

$$\mathcal{V} = \{\textit{the, a, an, in, for, penny, pound}\} \quad |V|=7$$

$$\text{sentence} = \textit{in for a penny, in for a pound}$$

$$x = (\textit{in, for, a, penny, in, for, a, pound})$$

$$in = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{matrix} \mathbb{1}(\text{"in"} = \text{"the"}) \\ \vdots \end{matrix} \qquad for = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \qquad \phi_{BoW}(x) = \begin{array}{c} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \\ 2 \\ 1 \\ 1 \end{bmatrix} \end{array} \begin{array}{l} \text{the} \\ \text{a} \\ \text{an} \\ \text{in} \\ \text{for} \\ \text{pen} \\ \text{pou} \end{array}$$

one-hot encoding

$$\phi_{BoW}(x) = \sum_{i=1}^{n} \phi_{\text{one-hot}}(x_i)$$

Compare with naive Bayes

- Our naive Bayes model ($\underline{x_i} \in \{1, \ldots, |\mathcal{V}|\}$):

$$X_i \mid Y = y \sim \text{Categorical}(\theta_{1,y}, \ldots, \theta_{|\mathcal{V}|,y}) \ .$$

- The naive Bayes generative story corresponds to a multinomial distribution of the BoW count vector:

$$\phi_{\text{BoW}}(X) \mid Y = y \sim \text{Multinomial}(\theta_{1,y}, \ldots, \theta_{|\mathcal{V}|,y}, n) \ .$$

- Both multinomial naive Bayes and logistic regression learn a linear separator $w \cdot \phi_{\text{BoW}}(x) + b = 0$.

Question: what's the advantage of using logistic regression?

Feature extractor

Define each feature as a function $\phi_i \colon \mathcal{X} \to \mathbb{R}$.

$$\phi_1(x) = \begin{cases} 1 & x \text{ contains ``happy''} \\ 0 & \text{otherwise} \end{cases},$$

$$\phi_2(x) = \begin{cases} 1 & x \text{ contains words with suffix ``yyyy''} \\ 0 & \text{otherwise} \end{cases}.$$

In practice, use a dictionary

$$\texttt{feature\_vector}[\texttt{"prefix=un+suffix=ing"}] = 1$$

Feature vectors for multiclass classification

Multinomial logistic regression

$$p(y=k \mid x; w) = \frac{e^{w_k \cdot \phi(x)}}{\sum_{i \in \mathcal{Y}} e^{w_i \cdot \phi(x)}} \quad (y \in \{1, \dots, K\})$$

$$\big\downarrow \text{ fl}$$

compatible features

$$p(y=k \mid x; w) = \frac{e^{w \cdot \underline{\Psi(x,k)}}}{\sum_{i \in \mathcal{Y}} e^{w \cdot \Psi(x,i)}} \quad (y \in \{1, \dots, K\})$$

Multivector construction of $\Psi(x, y)$:

$$\Psi(x,1) \overset{\text{def}}{=} [\underbrace{\vec{0}}_{y=0} \quad \underbrace{\phi(x)}_{y=1} \quad \cdots \quad \underbrace{\vec{0}}_{y=K}] \qquad K \times d$$

N-gram features

Potential problems with the the BoW representation?

N-gram features

Potential problems with the the BoW representation?

N-gram features:

in for a penny , in for a pound

- Unigram in, for ...
- Bigram in_for, for_a ...
- Trigram in_for_a_ ...

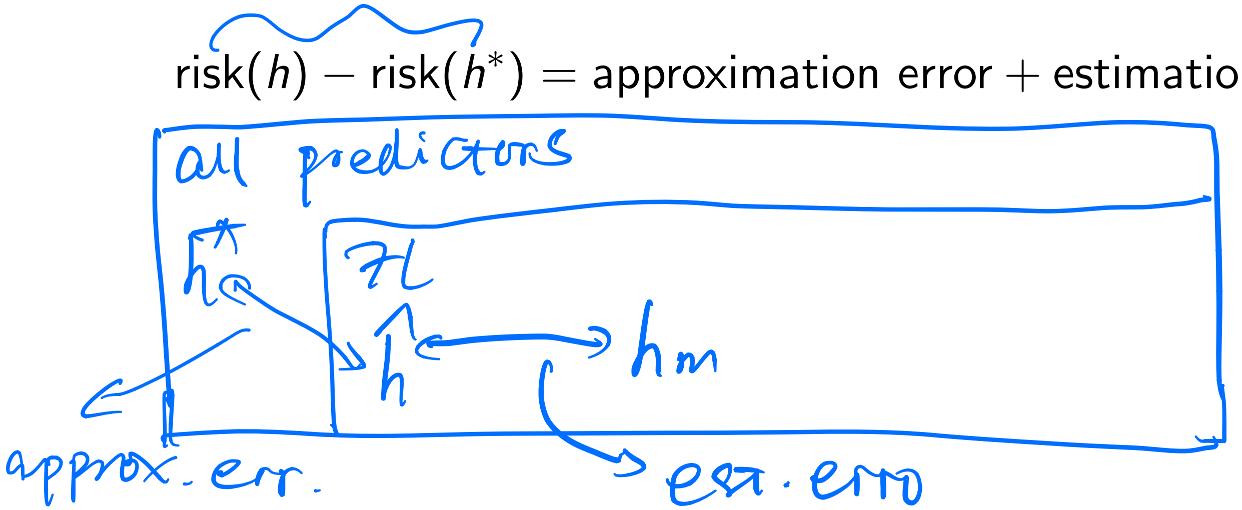
What's the pros/cons of using higher order n-grams?

Table of Contents

Bias-variance trade-off

Error decomposition:

$$\text{risk}(h) - \text{risk}(h^*) = \text{approximation error} + \text{estimation error}$$



Larger hypothesis class: approximation error $\downarrow$, estimation error $\uparrow$

Smaller hypothesis class: approximation error $\uparrow$, estimation error $\downarrow$

How to control the size of the hypothesis class?

Reduce the dimensionality

Linear predictors: $\mathcal{H} = \{w : w \in \mathbb{R}^d\}$

Reduce the number of features:

- reduce n-gram order
- filter by frequency
- stemming.

  playing, played $\longrightarrow$ play
- stopwords, and, the, they (task-spec)

Feature selection

- Forward/Backward selection
- ℓ_1 regularization
- boosting.

Reduce the dimensionality

Linear predictors: $\mathcal{H} = \left\{w : w \in \mathbb{R}^d\right\}$

Reduce the number of features:

Other predictors:

- Depth of decision trees
- Degree of polynomials
- Number of decision stumps in boosting

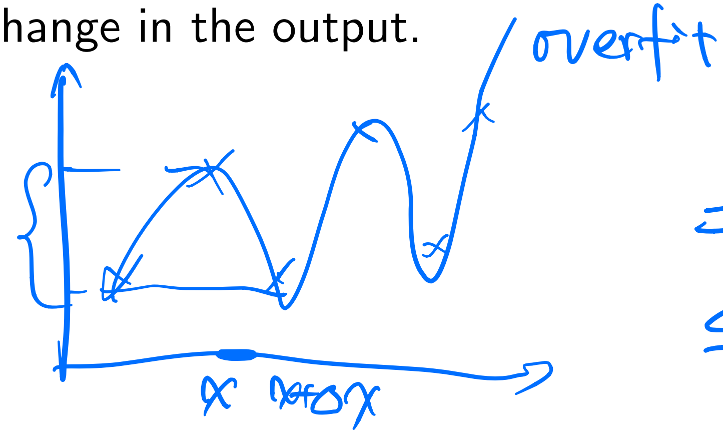
Regularization

Reduce the norm of w:

$$\underbrace{\min_w \frac{1}{N}\sum_{i=1}^{N} L(x^{(i)}, y^{(i)}, w)}_{\text{average loss}} + \underbrace{\frac{\lambda}{2}\|w\|_2^2}_{\ell_2 \text{ norm}}$$

ERM

Why is small norm good? Small change in the input doesn't cause large change in the output.



$$|w\cdot x - w\cdot(x+\Delta x)|$$
$$= |w\cdot \Delta x|$$
$$\le \underline{\|w\|}\,\|\Delta x\|$$

Gradient descent with ℓ_2 regularization

Run SGD on

$$\min_w \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(x^{(i)}, y^{(i)}, w)}_{\text{average loss}} + \underbrace{\frac{\lambda}{2}\|w\|_2^2}_{\underline{\ell_2 \text{ norm}}}$$

Also called **weight decay** in the deep learning literature:

$$w \leftarrow w - \eta(\nabla_w L(x, y, w) + \underline{\lambda w})$$

Shrink w in each update.

Hyperparameter tuning

Hyperparameters: parameters of the learning algorithm (not the model)

Example: use MLE to learn a logistic regression model using BoW features

Hyperparameter tuning

Hyperparameters: parameters of the learning algorithm (not the model)

Example: use MLE to learn a logistic regression model using BoW features

How do we select hyperparameters?

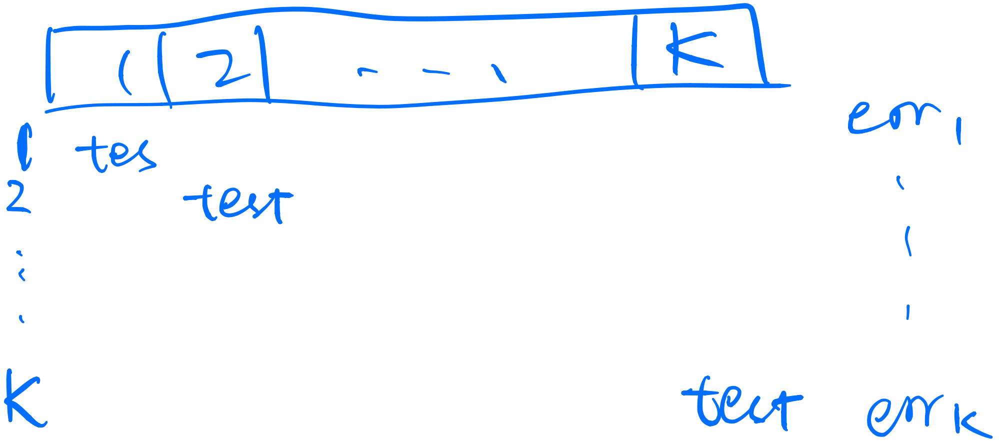
  Pick those minimizing the training error

  Pick those minimizing the test error

Validation

Validation set: a subset of the training data reserved for tuning the learning algorithm (also called the **development set**).

K-**fold cross validation**



It's important to look at the data and errors during development, but not the test set.

Evaluation

- Accuracy
- Precision
- Recall
- F1
- Macro vs micro average