Text Classification

He He

NEW YORK UNIVERSITY

January 24, 2023

Text classification

- Input: text (sentence, paragraph, document)
- Predict the category or property of the input text
  - Sentiment classification: Is the review positive or negative?
  - Spam detection: Is the email/message spam or not?
  - Hate speech detection: Is the tweet/post toxic or not?
  - Stance classification: Is the opinion liberal or conservative?

Text classification

- Input: text (sentence, paragraph, document)
- Predict the category or property of the input text
  - Sentiment classification: Is the review positive or negative?
  - Spam detection: Is the email/message spam or not?
  - Hate speech detection: Is the tweet/post toxic or not?
  - Stance classification: Is the opinion liberal or conservative?
- Predict the relation of two pieces of text
  - Textual entailment (HW1): does the premise entail the hypothesis?
    Premise: The dogs are running in the park.
    Hypothesis: There are dogs in the park.
  - Paraphrase detection: are the two sentences paraphrases?
    Sentence 1: The dogs are in the park.
    Sentence 2: There are dogs in the park.

Table of Contents

Intuition

- **Example**: sentiment classification for movie reviews

  Action. Comedy. Suspense. This movie has it all. The Plot goes that 4 would be professional thieves are invited to take part in a heist in a small town in Montana. every type of crime movie archetype character is here. Frank, the master mind. Carlos, the weapons expert. Max, the explosives expert. Nick, the safe cracker and Ray, the car man. Our 4 characters meet up at the train station and from the beginning none of them like or trust one another. Added to the mix is the fact that Frank is gone and they are not sure why they have called together. Now Frank is being taken back to New Jersey by the 2 detectives but soon escapes on foot and tries to make his way back to the guys who are having all sorts of problems of their own. Truly a great film loaded with laughs and great acting. Just an overall good movie for anyone looking for a laugh or something a little different

Intuition

- **Example**: sentiment classification for movie reviews

  Action. Comedy. Suspense. This movie has it all. The Plot goes that 4 would be professional thieves are invited to take part in a heist in a small town in Montana. every type of crime movie archetype character is here. Frank, the master mind. Carlos, the weapons expert. Max, the explosives expert. Nick, the safe cracker and Ray, the car man. Our 4 characters meet up at the train station and from the beginning none of them like or trust one another. Added to the mix is the fact that Frank is gone and they are not sure why they have called together. Now Frank is being taken back to New Jersey by the 2 detectives but soon escapes on foot and tries to make his way back to the guys who are having all sorts of problems of their own. Truly a great film loaded with laughs and great acting. Just an overall good movie for anyone looking for a laugh or something a little different

- **Idea**: count the number of positive/negative words
  - What is a “word”?
  - How do we know which are positive/negative?

Preprocessing: tokenization

Goal: Splitting a string of characters 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”, “.”]
  - Special cases: U.S., Ph.D. etc.
- Dictionary / sequence labeler: “我没有去看电影。” $\rightarrow$ [“我”, “没有”, “去”, “看”, “电影”, “。”]

Preprocessing: tokenization

Goal: Splitting a string of characters to a sequence of **tokens** $[x_1, \ldots, x_n]$.

Language-specific solutions

- Regular expression: “I didn’t watch the movie”. → [“I”, “did”, “n’t”, “watch”, “the”, “movie”, “.”]
  - Special cases: U.S., Ph.D. etc.
- Dictionary / sequence labeler: “我没有去看电影。” → [“我”, “没有”, “去”, “看”, “电影”, “。”]

General solutions: don’t split by words

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

Classification: problem formulation

- **Input**: a sequence of tokens $x = (x_1, \ldots x_n)$ where $x_i \in \mathcal{V}$.
- **Output**: binary label $y \in \{0, 1\}$.
- **Probabilistic model**:

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

  where p_θ is a distribution parametrized by $\theta \in \Theta$.
- Modeling question: what's the parametric form of p_θ?

Modeling $p(y \mid x)$

How to write a review:

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

Modeling $p(y \mid x)$

How to write a review:

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

$$p(y) = \tag{1}$$

$$p(x \mid y) = \tag{2}$$

$$= \tag{3}$$

Modeling $p(y \mid x)$

How to write a review:

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

$$p(y) = \text{Bernoulli}(\alpha) \tag{1}$$

$$p(x \mid y) = \tag{2}$$

$$= \tag{3}$$

Modeling $p(y \mid x)$

How to write a review:

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

$$p(y) = \text{Bernoulli}(\alpha) \tag{1}$$

$$p(x \mid y) = \prod_{i=1}^{n} p(x_i \mid y) \qquad \text{(independent assumption)} \tag{2}$$

$$= \tag{3}$$

Modeling $p(y \mid x)$

How to write a review:

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

$$p(y) = \text{Bernoulli}(\alpha) \tag{1}$$

$$p(x \mid y) = \prod_{i=1}^{n} p(x_i \mid y) \qquad \text{(independent assumption)} \tag{2}$$

$$= \prod_{i=1}^{n} \text{Categorical}(\underbrace{\theta_{1,y}, \ldots, \theta_{|\mathcal{V}|,y}}_{\text{sum to 1}}) \tag{3}$$

Modeling $p(y \mid x)$

How to write a review:

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

$$p(y) = \text{Bernoulli}(\alpha) \tag{1}$$

$$p(x \mid y) = \prod_{i=1}^{n} p(x_i \mid y) \qquad \text{(independent assumption)} \tag{2}$$

$$= \prod_{i=1}^{n} \text{Categorical}(\underbrace{\theta_{1,y}, \ldots, \theta_{|\mathcal{V}|,y}}_{\text{sum to 1}}) \tag{3}$$

Bayes rule:

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

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 (e.g., Gaussian distribution)

Learning: maximum likelihood estimation

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

Goal: find the parameters that make the observed data most probable.

Learning: maximum likelihood estimation

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

Goal: find the parameters that make the observed data most probable.

Likelihood function of θ given D:

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

Maximum (log-)likelihood estimator:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} L(\theta; D) = \arg\max_{\theta \in \Theta} \sum_{i=1}^{N} \log p(y_i; \theta) \tag{4}$$

MLE and ERM

ERM:

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

MLE and ERM

ERM:

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

MLE:

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

MLE and ERM

ERM:

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

MLE:

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

What's the connection between MLE and ERM?

MLE is equivalent to ERM with the **negative log-likelihood** (NLL) loss function:

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

MLE solution for our Naive Bayes model

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

$$\begin{aligned} p_{\text{MLE}}(w \mid y) &= \frac{\text{count}(w, y)}{\sum_{w \in \mathcal{V}} \text{count}(w, y)} \\ &= \text{how often the word occur in positive/negative documents} \\ &= \text{“positive/negative score of the word”} \end{aligned}$$

$$\begin{aligned} p_{\text{MLE}}(y = k) &= \frac{\sum_{i=1}^{N} \mathbb{I}\left(y^{(i)} = k\right)}{N} \\ &= \text{fraction of positive/negative documents} \end{aligned}$$

Inference: make predictions using the model

Inference: $y = \arg\max_{y \in \mathcal{Y}} p_\theta(y \mid x)$

Inference: make predictions using the model

Inference: $y = \arg\max_{y \in \mathcal{Y}} p_\theta(y \mid x)$

Compare $p_\theta(y = 1 \mid x)$ and $p_\theta(y = 0 \mid x)$:

$$\frac{p_\theta(y = 1 \mid x)}{p_\theta(y = 0 \mid x)} = \frac{p_\theta(x \mid y = 1)p_\theta(y = 1)}{p_\theta(x \mid y = 0)p_\theta(y = 0)}$$

Inference: make predictions using the model

Inference: $y = \arg\max_{y \in \mathcal{Y}} p_\theta(y \mid x)$

Compare $p_\theta(y = 1 \mid x)$ and $p_\theta(y = 0 \mid x)$:

$$\frac{p_\theta(y = 1 \mid x)}{p_\theta(y = 0 \mid x)} = \frac{p_\theta(x \mid y = 1) p_\theta(y = 1)}{p_\theta(x \mid y = 0) p_\theta(y = 0)}$$

Assuming $p_\theta(y = 1) = p_\theta(y = 0)$, we only need to compare $p_\theta(x \mid y = 1)$ and $p_\theta(x \mid y = 0)$.

$$\text{score of class } k = \log p_\theta(x \mid y = k) = \sum_{i=1}^{n} \log p_\theta(x_i \mid y = k)$$

(Adding up positive/negative scores of each word)

Feature design

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

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

Summary of Naive Bayes models

- Modeling: the conditional indepedence assumption simplifies the problem
- Learning: MLE (or ERM with negative log-likelihood loss)
- Inference: very fast (adding up scores of each word)

Table of Contents

Discriminative models

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

Discriminative models

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

	generative models	discriminative models
modeling	joint: $p(x, y)$	conditional: $p(y \mid x)$
assumption on y	yes	yes
assumption on x	yes	no
development	generative story	feature extractor

Model $p(y \mid x)$

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

y is a Bernoulli variable:

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

Model $p(y \mid x)$

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

y is a Bernoulli variable:

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

Bring in x:

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

Model $p(y \mid x)$

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

y is a Bernoulli variable:

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

Bring in x:

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

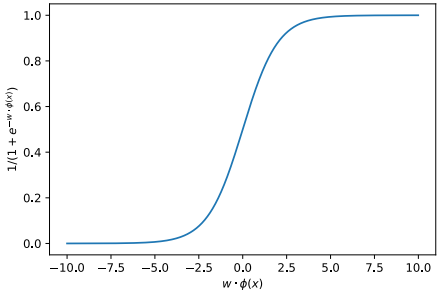
Parametrize $h(x)$ using a linear function:

$$h(x) = w \cdot \phi(x) + b \qquad \phi \colon \mathcal{X} \to \mathbb{R}^d$$

Problem: $h(x) \in \mathbb{R}$ (score)

Logistic regression

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



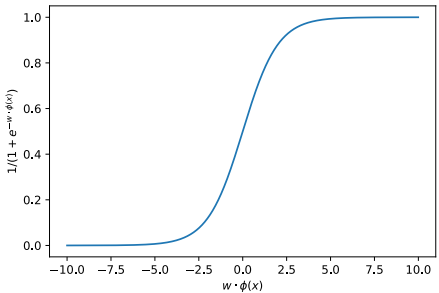
$$p(y = 1 \mid x; w) = \frac{1}{1 + e^{-w \cdot \phi(x)}} \quad (y \in \{0, 1\})$$

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

"softmax"

Logistic regression

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



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

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

Inference

$$\hat{y} = \arg\max_{k \in \mathcal{Y}} p(y = k \mid x; w) \tag{5}$$

$$= \arg\max_{k \in \mathcal{Y}} \frac{e^{w_k \cdot \phi(x)}}{\sum_{i \in \mathcal{Y}} e^{w_i \cdot \phi(x)}} \tag{6}$$

$$= \arg\max_{k \in \mathcal{Y}} e^{w_k \cdot \phi(x)} \tag{7}$$

$$= \arg\max_{k \in \mathcal{Y}} \underbrace{w_k \cdot \phi(x)}_{\text{score for class } k} \tag{8}$$

MLE for logistic regression

- Likelihood function is concave / NLL is convex
- No closed-form solution
- Use stochastic gradient ascent

BoW representation

Example:

$$\mathcal{V} = \{\textit{the}, \textit{a}, \textit{an}, \textit{in}, \textit{for}, \textit{penny}, \textit{pound}\}$$
$$\text{sentence} = \textit{in for a penny, in for a pound}$$
$$x = (\textit{in}, \textit{for}, \textit{a}, \textit{penny}, \textit{in}, \textit{for}, \textit{a}, \textit{pound})$$

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

BoW representation

Example:

$$\begin{aligned} \mathcal{V} &= \{\textit{the}, \textit{a}, \textit{an}, \textit{in}, \textit{for}, \textit{penny}, \textit{pound}\} \\ \text{sentence} &= \textit{in for a penny, in for a pound} \\ x &= (\textit{in}, \textit{for}, \textit{a}, \textit{penny}, \textit{in}, \textit{for}, \textit{a}, \textit{pound}) \end{aligned}$$

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

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

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

$$\phi_{\text{one-hot}}(x_1) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{the sentence contains the word "in"}$$

$$\phi_{\text{BoW}}(x) = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 1 & 1 \end{bmatrix} \quad \text{the sentence contains 2 occurrences of "in"}$$

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, a, ...
- Bigram: in/for, for/a, a/penny, ...
- Trigram: in/for/a, for/a/penny, ...

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

Feature extractor

Logistic regression allows for richer features (what's the limitation of NB?)

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

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

Table of Contents

Error decomposition

Let's ignore the optimization error, assuming we always find the optimum

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

- Approximation error: $\text{risk}(\text{best hypo in } \mathcal{H}) - \text{risk}(h^*)$
  Does my hypothesis space contain the true hypothesis?
- Estimation error: $\text{risk}(\hat{h}) - \text{risk}(\text{best hypo in } \mathcal{H})$
  Can I find the best hypothesis given limited data?

Error decomposition

Let's ignore the optimization error, assuming we always find the optimum

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

- Approximation error: $\text{risk}(\text{best hypo in } \mathcal{H}) - \text{risk}(h^*)$
  Does my hypothesis space contain the true hypothesis?
- Estimation error: $\text{risk}(\hat{h}) - \text{risk}(\text{best hypo in } \mathcal{H})$
  Can I find the best hypothesis given limited data?

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

Reduce the number of features. (Ideas for text classification?)

Reduce the dimensionality

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

Reduce the number of features. (Ideas for text classification?)

For other predictors:

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

Regularization

Reduce the "size" of w:

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

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

Gradient descent with ℓ_2 regularization

Run SGD on

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

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

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

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

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**
[board]

Validation

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

***K*-fold cross validation**
[board]

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