Text classification

He He

January 22, 2024

Table of Contents

Course overview
- Logistics
- A brief history of NLP/AI
- Challenges in NLP

Supervised learning basics
- Generalization
- Loss functions
- Optimization

Text classification
- Generative models: naive Bayes
- Discriminative models: logistic regression

Basic information

- Instructor: He He
- TAs: Xiang Pan, Haitian Jiang, Arun Purohit
- You can find all information about this course on the website: https://nyu-cs2590.github.io/spring2025/.
- Best way to communicate with us is through **Campuswire** (https://campuswire.com/p/GDE4D5420).
- Let us know if you have accessibility needs.
- Pdf slides will be uploaded before the lecture.

What we expect you to know

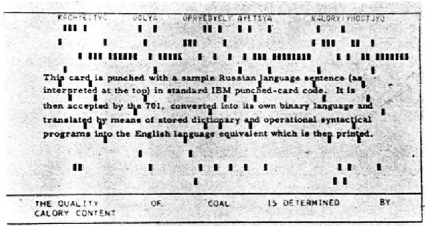
- **Linear algebra**: vector space, vector norm, dot product, gradient etc.
- **Probability and statistics**: conditional probability, expectation, Bayes rule etc.
- **Basic machine learning**: loss function, gradient descent, logistic regression, neural networks etc.
- **Programming**: Python, Numpy, HPC, and deep learning libraries (Pytorch, Huggingface etc.)

Grading

- **Assignment (60%)**: 4 assignments (written + coding questions), each counting 15%
- **Quizzes (15%)**: to encourage attendence to guest lectures, we will have an online quiz about each lecture.
- **Final exam (25%)**: there will be an online exam through Gradescope.

Early rule-based systems: the Georgetown-IBM experiment

- The Russian-English machine translation program:



- A vocabulary of 250 words
- Using 6 grammar rules, e.g.,

  `If first code is 110, is third code associated with preceding complete word equal to 21? If so, reverse order of appearance of words in output (i.e., word carrying 21 should follow that carrying 110)---otherwise, retain order.`

Limitations of early systems

- Optimism in the 50's and 60's: working on tasks that were too complex at that time

  "Within the very near future—much less than twenty-five years—we shall have the technical capability of substituting machines for any and all human functions in organizations."

- Disappointing results due to
  - **Limited computation**: hardware has limited speed and memory
  - **Combinatorial explosion**: algorithms are intractable in realistic settings
  - **Underestimated complexity**: ambiguity, commonsense knowledge, multimodality, etc.

The rise of statistical learning in the 80's

- Notable progress in MT from IBM (neglected knowlege of linguistics).
- HMMs widely used for speech recognition.

  "Every time I fire a linguist, the performance of the speech recognizer goes up."—Frederick Jelinek.
- The paradigm shift: expert knowledge + rules $\rightarrow$ data + features
- Statistical learning became the main driving force of NLP.

The deep learning tsunami

- Before deep learning (circa 2015), NLP is mostly about structured prediction and feature engineering.
- Neural networks can automatically learn good features/representations for a task
- The paradigm shift: features $\rightarrow$ network architectures + embeddings
- All NLP models are neural networks nowadays.

Models and data getting larger

- Since around 2018, Transformer-based pretrained models have become the foundation models.
- Pretraining on large data provides useful representations for many downstream tasks.
- The paradigm shift: architecture design $\rightarrow$ transfer learning (fine-tuning)
- More recently, large-scale language modeling enables models with general capabilities (e.g., ChatGPT by OpenAI).
- The paradigm shift: transfer learning $\rightarrow$ "elicitation"

Structure of the course

- **Module 1: supervised learning**
  How to formalize NLP tasks?
  - Word vectors, RNNs, Transformers, encoders and decoders
- **Module 2: representation learning**
  How to learn general representations of text without annotation?
  - Pretraining and finetuning (BERT, GPT, T5)
- **Module 3: large language models**
  Can a single model solve all tasks?
  - Scaling, alignment (SFT, RLHF, DPO), evaluation
- **Guest lectures on advanced topics**
  - RAG, agents, open-source LLMs

Why is natural language hard?

Why is natural language hard?

- **Discrete**
  - How to define metrics?

    I work at NYU. vs I work for NYU.
    This is good. vs This is actually good.

  - How to define transformations?

    The food is okay. → The food is awesome!
    They made a brief return to Cambridge. → They returned.

  - In general, it's hard to represent text as mathematical objects.

Why is natural language hard?

- **Compositional**
  - The whole is built from parts (chars, words, sentences, paragraphs, documents...)
  - How to generalize when we don't see all possible combinations?
  - Can't brute force! E.g., Lake et al., 2018

    Vocabulary:

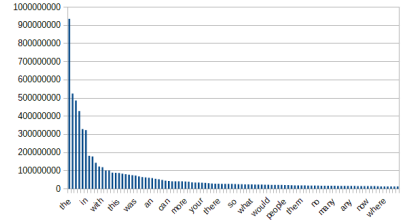
    {jump, walk, turn, once, twice, left, right, before, after, and}

    Sentences:

    jump
    jump left
    jump left and walk right
    jump left after walk right once before turn left twice
    ...

Why is natural language hard?

- **Sparse**
  - How to handle the long tail?
  - Zipf's law: word frequency $\propto \frac{1}{\text{rank}}$



  - Many linguistic phenomena follow Zipf's law, e.g.,
    BoA's financial assistant Erica: *The bank "learned [that] there are over 2,000 different ways to ask us to move money."*[1]

[1] https://www.aiqudo.com/2019/06/28/voice-success-story-erica-bank-america/

Why is natural language hard?

- **Ambiguous**
  - How to interpret meaning in context?

    Bass: fish? guitar? frequency? (word sense disambiguiation)

    I shot an elephant in my pajamas: who is in the pajamas? (PP attachment)

    The spirit is willing, but the flesh is weak.
    $\rightarrow$ The vodka is strong but the meat is rotten. (machine translation)

Table of Contents

Course overview
- Logistics
- A brief history of NLP/AI
- Challenges in NLP

Supervised learning basics
- Generalization
- Loss functions
- Optimization

Text classification
- Generative models: naive Bayes
- Discriminative models: logistic regression

Example: spam filter

- Writing **rules**

  Contains “Viagra”
  Contains “Rolex”
  Subject line is all caps
  ...

- Learning from **data**
  1. Collect emails labeled as spam or non-spam
  2. Design features, e.g., first word of the subject, nouns in the main text
  3. Learn a binary classifier

Example: spam filter

- Writing **rules**

  Contains “Viagra”
  Contains “Rolex”
  Subject line is all caps
  …

- Learning from **data**
  1. Collect emails labeled as spam or non-spam
  2. Design features, e.g., first word of the subject, nouns in the main text
  3. Learn a binary classifier

Pros and cons of each approach?

Key challenges in machine learning

- Availability of large amounts of (annotated) **data**
  - Data collection: scraping, crowdsourcing, expert annotation
  - Quality control: data quality can have large impact on the final model (garbage in garbage out)
  - Don’t take it for granted: always check the data source!

Key challenges in machine learning

- Availability of large amounts of (annotated) **data**
  - Data collection: scraping, crowdsourcing, expert annotation
  - Quality control: data quality can have large impact on the final model (garbage in garbage out)
  - Don't take it for granted: always check the data source!

How would you collect a dataset for the spam filtering task?

Key challenges in machine learning

- **Generalize** to unseen samples
  - We want to build a model: $h\colon \mathcal{X}$ (input space) $\to \mathcal{Y}$ (output space)
  - It is easy to achieve high accuracy on the training set.
  - But we want the model to perform well on unseen data, too.
  - What should be the learning objective?

Risk minimization

- Assume a data generating distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ (e.g., spam writers and non-spam writers)

Risk minimization

- Assume a data generating distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ (e.g., spam writers and non-spam writers)
- We have access to a training set: m samples from $\mathcal{D}$, $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$

Risk minimization

- Assume a data generating distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ (e.g., spam writers and non-spam writers)
- We have access to a training set: m samples from $\mathcal{D}$, $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$
- We can measure the goodness of a prediction $h(x)$ by comparing it against the groundtruth y using some **loss function** L

Risk minimization

- Assume a data generating distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ (e.g., spam writers and non-spam writers)
- We have access to a training set: m samples from $\mathcal{D}$, $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$
- We can measure the goodness of a prediction $h(x)$ by comparing it against the groundtruth y using some **loss function** L
- Our goal is to minimize the expected loss over $\mathcal{D}$ (**risk**):

$$\text{minimize} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[L(h, x, y)\right] \ ,$$

but it cannot be computed (why?).

Empirical risk minimization (ERM)

- Instead, we minimize the average loss on the training set (**empirical risk**)

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} L(h, x^{(i)}, y^{(i)})$$

Empirical risk minimization (ERM)

- Instead, we minimize the average loss on the training set (**empirical risk**)

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} L(h, x^{(i)}, y^{(i)})$$

  - In the limit of infinite samples, empirical risk converges to risk (LLN).

Empirical risk minimization (ERM)

- Instead, we minimize the average loss on the training set (**empirical risk**)

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} L(h, x^{(i)}, y^{(i)})$$

  - In the limit of infinite samples, empirical risk converges to risk (LLN).
- **Key question**: does small empirical risk imply small risk?

Empirical risk minimization (ERM)

- Instead, we minimize the average loss on the training set (**empirical risk**)

$$\text{minimize} \quad \frac{1}{m} \sum_{i=1}^{m} L(h, x^{(i)}, y^{(i)})$$

  - In the limit of infinite samples, empirical risk converges to risk (LLN).
- **Key question**: does small empirical risk imply small risk?
- Trivial solution to (unconstrained) ERM: memorize the data points

Overfitting vs underfitting

- Problem: extrapolate information from one part of the input space to unobserved parts!
  - training set $\rightarrow$ test set
- Solution: constrain the prediction function to a subset, i.e. a **hypothesis space** $h \in \mathcal{H}$.

Overfitting vs underfitting

- Problem: extrapolate information from one part of the input space to unobserved parts!
  - training set $\rightarrow$ test set
- Solution: constrain the prediction function to a subset, i.e. a **hypothesis space** $h \in \mathcal{H}$.
- Trade-off between complexity of $\mathcal{H}$ and generalization
- Question for us: how to choose a good $\mathcal{H}$ for certain domains/tasks

Summary

1. Obtain training data $D_{\text{train}} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$.
2. Choose a loss function L and a hypothesis class $\mathcal{H}$ (domain knowledge).
3. Learn a predictor by minimizing the empirical risk (optimization).

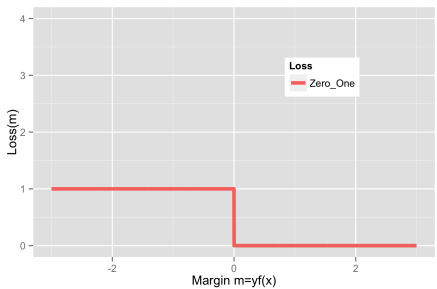
Formalization

- Task: binary classification $x \in \mathcal{X}, y \in \{+1, -1\}$
- Model: $f_w : \mathcal{X} \to \mathbf{R}$ parametrized by $w \in \mathbf{R}^d$
  - Output a score for each example
- Prediction: $\text{sign}(f_w(x))$
  - Positive scores are mapped to the positive class
- Loss functions: quantify the goodness of the model output $f_w(x)$ given y

Zero-one loss

First idea: check if the prediction is the same as the label

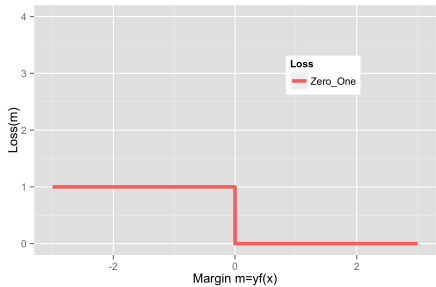
$$L(x, y, f_w) = \mathbb{I}\left[\text{sign}(f_w(x)) \neq y\right] = \mathbb{I}\left[\underbrace{y f_w(x)}_{\text{margin}} \leq 0\right] \tag{1}$$



Zero-one loss

First idea: check if the prediction is the same as the label

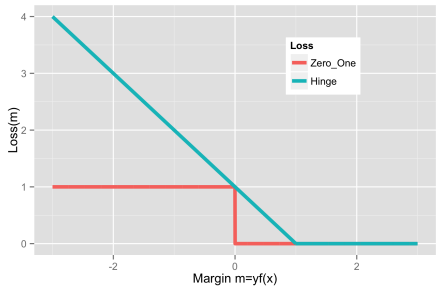
$$L(x, y, f_w) = \mathbb{I}\left[\text{sign}(f_w(x)) \neq y\right] = \mathbb{I}\left[\underbrace{y f_w(x)}_{\text{margin}} \leq 0\right] \tag{1}$$



Problem: not differentiable

Hinge loss

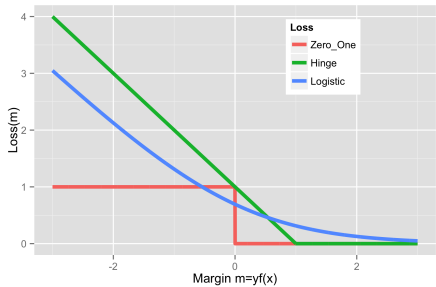
$$L(x, y, f_w) = \max(1 - yf_w(x), 0)$$



- A (sub)differentiable upperbound of the zero-one loss
- Not differentiable at margin $= 1$ (use subgradients)

Logistic loss

$$L(x, y, f_w) = \log(1 + e^{-yf_w(x)})$$



- Differentiable
- Always wants more margin (loss is never 0)

Summary

1. Obtain training data $D_{\text{train}} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$.
2. Choose a loss function L and a hypothesis class $\mathcal{H}$ (domain knowledge).
3. Learn a predictor by minimizing the empirical risk (optimization).

Stochastic gradient descent

- **Gradient descent (GD)** for ERM

$$w \leftarrow w - \eta \nabla_w \underbrace{\sum_{i=1}^{n} L(x^{(i)}, y^{(i)}, w)}_{\text{training set loss}}$$

Stochastic gradient descent

- **Gradient descent (GD)** for ERM

$$w \leftarrow w - \eta \nabla_w \underbrace{\sum_{i=1}^{n} L(x^{(i)}, y^{(i)}, w)}_{\text{training set loss}}$$

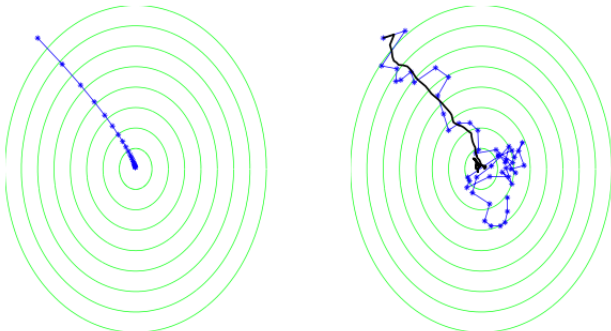
- **Stochastic gradient descent (SGD)**: take noisy but faster updates

For each $(x, y) \in D_{\text{train}}$:

$$w \leftarrow w - \eta \nabla_w \underbrace{L(x, y, f_w)}_{\text{example loss}}$$

GD vs SGD

Figure: Minimize $1.25(x+6)^2 + (y-8)^2$. Example from "Understanding Machine Learning: From Theory to Algorithms"



SGD step is noisier closer to the optimum—need to reduce step size gradually.

SGD summary

- Each update is efficient in both time and space
- Can be slow to converge
- Popular in large-scale ML, including non-convex problems
- In practice,
  - Randomly sample examples.
  - Fixed or diminishing step sizes, e.g. $1/t$, $1/\sqrt{t}$.
  - Stop when objective does not improve.
- Our main optimization techinque

Summary

- Choose hypothesis class based on domain knowledge
- Learning algorithm: empirical risk minimization
- Optimization: stochastic gradient descent

Table of Contents

Course overview
- Logistics
- A brief history of NLP/AI
- Challenges in NLP

Supervised learning basics
- Generalization
- Loss functions
- Optimization

Text classification
- Generative models: naive Bayes
- Discriminative models: logistic regression

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.

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

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 $p(x \mid y)$

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

$$p(x \mid y) = \tag{3}$$

$$\tag{5}$$

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 $p(x \mid y)$

$$p(y) = \text{Bernoulli}(\alpha) \tag{2}$$

$$p(x \mid y) = \tag{3}$$

$$\tag{5}$$

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 $p(x \mid y)$

$$p(y) = \text{Bernoulli}(\alpha) \tag{2}$$

$$p(x \mid y) = \prod_{i=1}^{n} p(x_i \mid y) \qquad \text{(independent assumption)} \tag{3}$$

(5)

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 $p(x \mid y)$

$$p(y) = \text{Bernoulli}(\alpha) \tag{2}$$

$$p(x \mid y) = \prod_{i=1}^{n} p(x_i \mid y) \qquad \text{(independent assumption)} \tag{3}$$

$$p(x_i = w \mid y) = \theta_{w,y} \quad \text{where } w \in \mathcal{V} \tag{4}$$

$$\sum_{w \in \mathcal{V}} \theta_{w,y} = 1 \tag{5}$$

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 $p(x \mid y)$

$$p(y) = \text{Bernoulli}(\alpha) \tag{2}$$

$$p(x \mid y) = \prod_{i=1}^{n} p(x_i \mid y) \qquad \text{(independent assumption)} \tag{3}$$

$$p(x_i = w \mid y) = \theta_{w,y} \quad \text{where } w \in \mathcal{V} \tag{4}$$

$$\sum_{w \in \mathcal{V}} \theta_{w,y} = 1 \tag{5}$$

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:

$$\hat{\theta} = \arg\max_{\theta\in\Theta} L(\theta; D) = \arg\max_{\theta\in\Theta} \sum_{i=1}^{N} \log p(y_i;\theta) \tag{6}$$

Quick remark: MLE and ERM

ERM:

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

Quick remark: MLE and ERM

ERM:

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

MLE:

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

Quick remark: 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 $p_\theta(x \mid y = 0)$.

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

In practice, 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)

Discriminative models: directly model $p(y \mid x)$

- y is a Bernoulli variable:

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

Discriminative models: directly model $p(y \mid x)$

- y is a Bernoulli variable:

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

- Bring in x:

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

Discriminative models: directly 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:

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

Discriminative models: directly 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:

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

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

Discriminative models: directly model $p(y \mid x)$

- y is a Bernoulli variable:

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

- Bring in x:

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

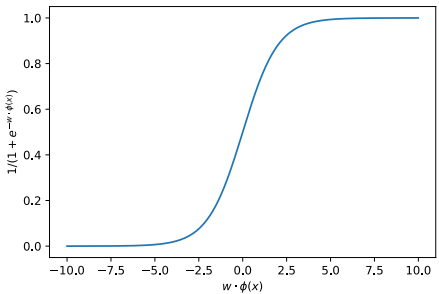
- Parametrize $h(x)$ using a linear function:

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

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

Logistic regression

Map $w \cdot \phi(x) \in \mathbb{R}$ to $[0, 1]$ 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

$$\hat{y} = \arg\max_{k \in \mathcal{Y}} p(y = k \mid x; w) \tag{7}$$

$$= \arg\max_{k \in \mathcal{Y}} \frac{e^{w_k \cdot \phi(x)}}{\sum_{i \in \mathcal{Y}} e^{w_i \cdot \phi(x)}} \tag{8}$$

$$= \arg\max_{k \in \mathcal{Y}} e^{w_k \cdot \phi(x)} \tag{9}$$

$$= \arg\max_{k \in \mathcal{Y}} \underbrace{w_k \cdot \phi(x)}_{\text{score for class } k} \tag{10}$$

MLE for logistic regression

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

- Likelihood function is concave / NLL is convex

MLE for logistic regression

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

- 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: $\phi\colon \mathcal{X} \to \mathbb{R}^d$.

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: $\phi\colon \mathcal{X} \to \mathbb{R}^d$.
BoW Idea: a sentence is the "sum" of words in it.

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

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

$\phi_{\text{BoW}}(x) = \begin{bmatrix}0 & 2 & 0 & 2 & 2 & 1 & 1\end{bmatrix}$ 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 (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

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

Summary

	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