Lecture 11: Naïve Bayes Classifier

Ling 1330/2330 Intro to Computational Linguistics

Na-Rae Han, 10/5/2021

Overview

- Text classification; Naïve Bayes classifier
  - *Language and Computers*: Ch.5 Classifying documents
  - NLTK book: Ch.6 Learning to classify text

  - Exercise 6 review

Automatic classification

- A ***classifier*** is an algorithm that processes a linguistic input and assigns it a **class** from a user-defined set.
  - It usually denotes a statistical model induced through machine learning.
  - The algorithm works off a set of weighted contextual features.

Example: movie reviews

all of this , of course , leads up to the predictable climax . but as foreseeable as the ending is , it's so damn cute and well-done that i doubt any movie in the entire year contains a scene the evokes as much pure joy as this part does . when ryan discovers the true identity of her online love , i was filled with such , for lack of a better word , happiness that for the first time all year , i actually left the theater smiling .

the acting is below average , even from the likes of curtis .. sutherland is wasted and baldwin , well , he's acting like a baldwin , of course . the real star here are stan winston's robot design , some schnazzy cgi , and the occasional good gore shot , like picking into someone's brain . so , if robots and body parts really turn you on , here's your movie . otherwise , it's pretty much a sunken ship of a movie .

- Classify each document as "positive" or "negative"

  ← A type of **sentiment analysis**

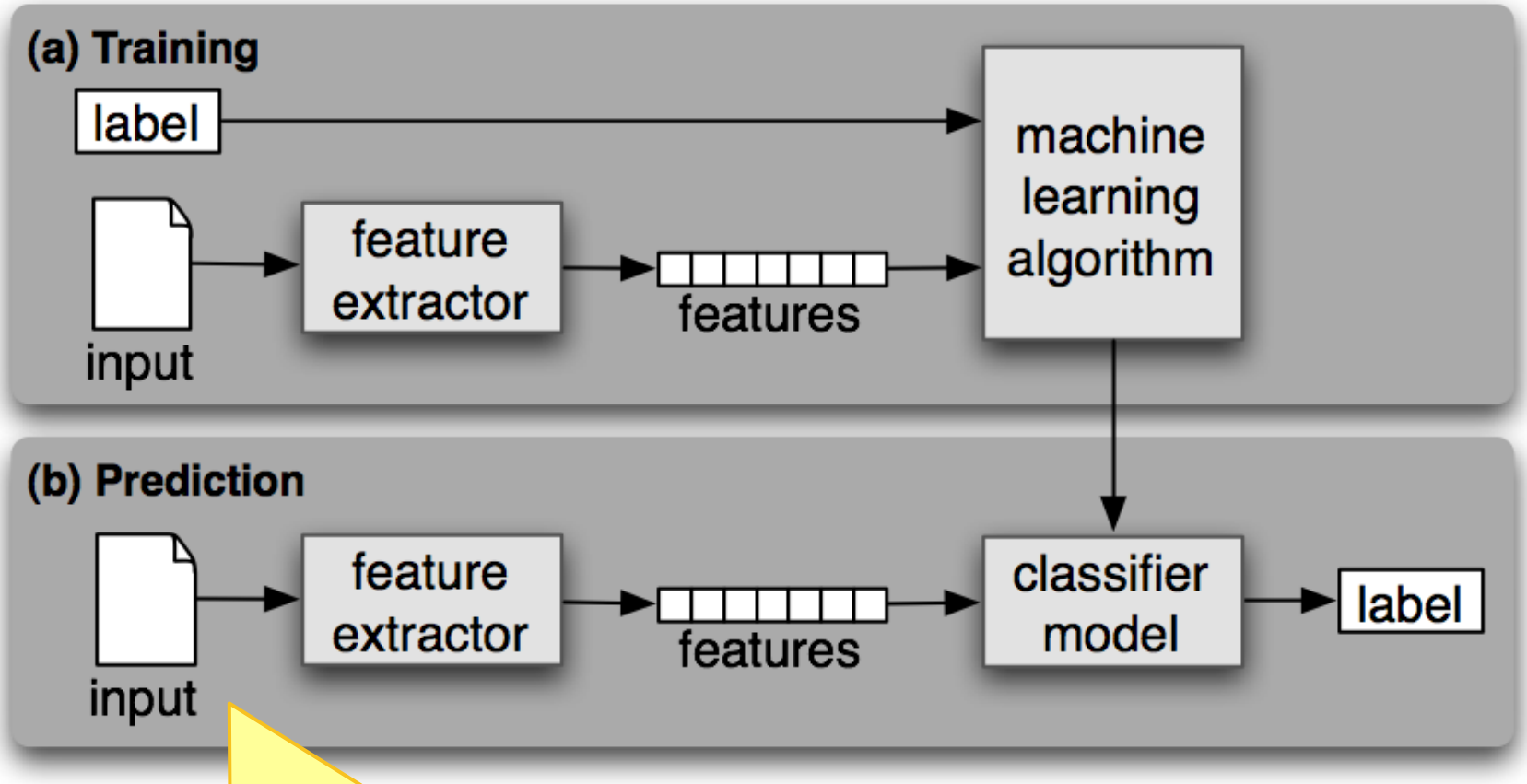
- What "features" can we use?
  - Words themselves → Exercise 6
  - N-grams, length, ...

How computers "learn"

- Document classification is an example of computer science engineering called **machine learning**
  - Just like humans learn from "experience", a computer algorithm *learns* from data
    - Learns what exactly? → Statistical patterns
  - Machine learning is not limited to linguistic data
    - Example?
- Machine learning requires:
  - *Training* data, often lots of them
  - *Testing* data for evaluation

    (Also: sometimes *development test data* for error analysis)

Machine learning (supervised)



If testing, correct labels are known → can calculate model's **accuracy**

Source: NLTK book

Features and evidence

- A classification decision must rely on some observable evidence → **features**
  - Female or male names?
    - The last letter of the name: 'a', 'k', etc.
  - What POS is *park*? What about *carbingly*?
    - Is *park* preceded by *the*? *to*?
    - Does *carbingly* end with 'ly'? 'ness'?
  - Is this document SPAM or HAM?
    - Does it contain the word *enlargement*?
    - Does it contain *linguistics*?

Feature engineering

- Deciding what features are relevant. Two types:
- **Kitchen sink** strategy
  - Throw a set of features to the machine learning algorithm, see what features are given greater weight and what gets ignored
  - Example: using each word in a document as a feature:
    - 'has-*cash*': True, 'has-*the*': True, 'has-*linguistics*': False, ..
- **Hand-crafted** strategy
  - Utilizing expert knowledge, determine a small set of features that are likely to be relevant
  - Example: grammatical error detection
    - For each sentence, determine grammatical/ungrammatical
    - Hand-coded features:
      - Subject-verb agreement, fragment or not, etc.

Weighting the evidence

- A classification decision involves reconciling multiple features with different levels of predictive power.

  ⇐Different types of classifiers use different algorithms for:

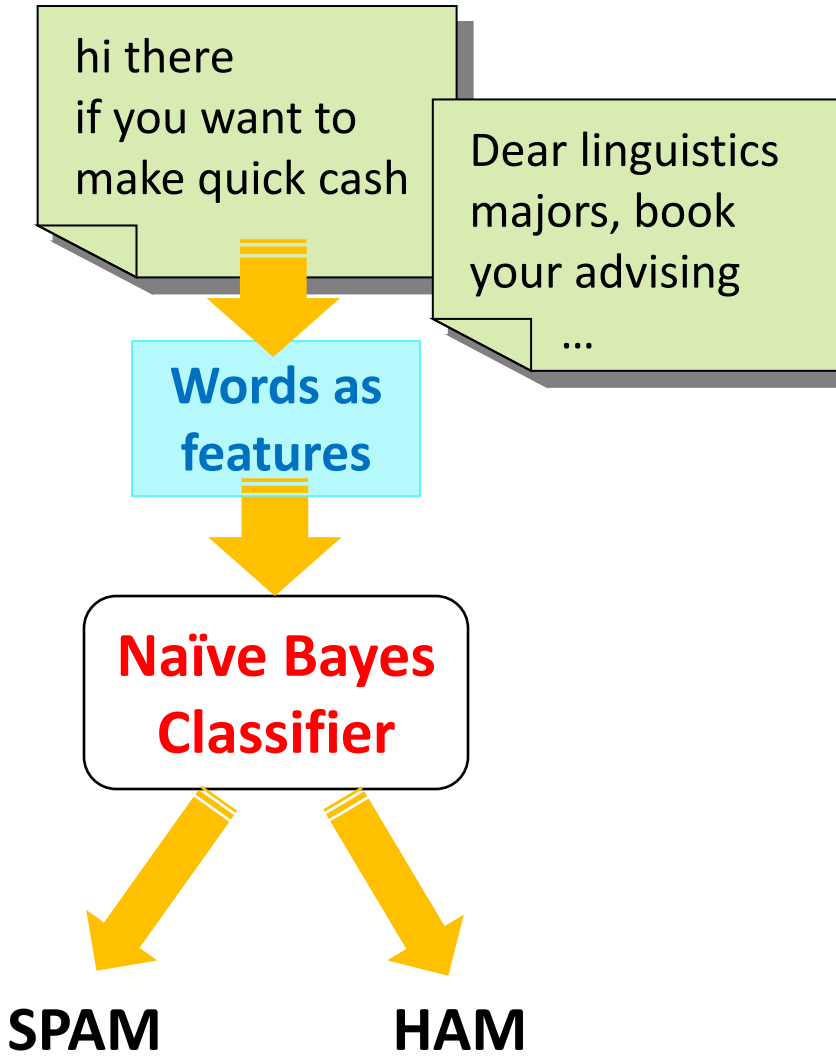
  1. Determining the **weights of individual features** in order to maximize its labeling success in the training data
  2. When given an input, using the feature weights to **compute the likelihood of a label**

- Popular machine learning methods:
  - **Naïve Bayes**
  - Decision tree
  - Maximum entropy (ME)
  - Hidden Markov model (HMM)
  - Neural network → Deep learning (!!)
  - Support vector machine (SVM)

A spam filter as a Naïve Bayes model

- Unit of linguistic input:
  - A document (email text)
  - Features: Words in document (kitchen sink strategy)
  - A document is reduced to **the set of words it contains**

    ← "Bag of words" assumption
- Classifier type:
  - Naïve Bayes
- Class labels:
  - SPAM/HAM



Bag of words + feature independence

P(Document)

= P('a', 'boring', 'movie', 'really', 'terrible', 'this', 'was', 'what', ',', '.') ❶

= P('a') * P('boring') * P('movie') * P('really') * P('terrible') * P('this') * P('was') * P('what') * P(',') * P('.') ❷

This was a really boring movie. What a terrible, terrible movie.

❶ **"Bag of words" assumption**: a document is reduced to the set of words it contains.

❷ **Feature independence assumption**: features are independent of each other (obviously false)

- The overall probability of this particular document P(Document) is then calculated as the product of the probabilities of each word occurring.
- How to calculate P('boring')?
  - 2,000 movie review documents, 'boring' found in 217 of them
    - ➔ 217/2000 = 0.1085
  - cf. P('really') = 867/2000 = 0.4335
  - cf. P('a') = 1996/2000 = 0.998

The Naïve Bayes Classifier

- A rough sketch of a Naïve Bayes algorithm:
  - Given an email document (D), process each piece of evidence (f_1, f_2, f_3... f_n) to support the two hypothesis:

    H_1: D is a SPAM.

    H_2: D is a HAM.

1. It starts with the **base probabilities**, also known as **priors**.

Suppose 70% of all email in the training data is SPAM:

H_1: D is a SPAM: 70%

H_2: D is a HAM: 30%.

2. Each piece of evidence (=feature) will strengthen one hypothesis and weaken the other.
   - `'contains-cash:YES'` → H_1 is now 85%, H_2 15%.
3. Repeat 2. for all features.
4. When done, rule for hypothesis with higher probability.

Inducing feature weights

- But how is the feature weight determined?

  `f1: 'contains-cash:YES'`

  ←Weight for SPAM hypothesis: P(f1|SPAM) ❶

  = the probability of 'cash' occurring, given a spam document

  ←Weight for HAM hypotheses: P(f1|HAM) ❷

  = the probability of 'cash' occurring, given a ham document

- In the training data:

	SPAM (n=140)	HAM (n=60)
day	20	20
linguistics	1	15
cash	90	3
Viagra	20	0
the	138	60
from	70	29

❶ = 90/140 = 0.64

❷ = 3/60 = 0.05

SPAM-to-HAM **odds ratio** of f1:

❶/❷ = 12.8 ← Feature strength

How about *linguistics* (f2)?

❶ P(f2|SPAM) = 1/140 = 0.007

❷ P(f2|HAM) = 15/60 = 0.25

HAM-to-SPAM odds ratio: **35.7**

Inducing feature weights

- But how is the feature weight determined?

  `f1: 'contains-cash:YES'`

  ← Weight for SPAM hypothesis: P(f1|SPAM) ❶

  = the probability of 'cash' occurring, given a spam document

  ← Weight for HAM hypotheses: P(f1|HAM) ❷

  = the probability of 'cash' occurring, given a ham document

- In the training data:

	SPAM (n=140)	HAM (n=60)
day	20	20
linguistics	1	15
cash	90	3
Viagra	20	0
the	138	60
from	70	29

How about weights of *the*?

❶ = 138/140 = 0.98

❷ = 60/60 = 1.00

SPAM-to-HAM odds ratio: ❶/❷ = 1

How about *from*?

❶ = 70/140 = 0.5

❷ = 29/60 = 0.48

SPAM-to-HAM odds ratio: ❶/❷ = 1

Both are *neutral* features

Smoothing

- We have to account for cases where a feature is never observed with a label.

	SPAM (n=140)	HAM (n=60)
day	20	20
linguistics	1	15
cash	90	3
Viagra	20	0
the	138	60
from	70	29

- 'Viagra' never occurred in a HAM document.
  - ❶ P(f3|SPAM) = 20/140 = 0.14
  - ❷ P(f3|HAM) = 0/60 = 0
- SPAM-HAM odds ratio = 0.14/0 ← !!!
- HAM-SPAM odds ratio = 0/0.14 = 0

  ← Not good, because it single-handedly renders P(HAM|D) to 0, regardless of what other features are present

- Just because we haven't seen a feature/label combination occur in the training set, doesn't mean it's impossible for that combination to occur.
- **Smoothing** is a process through which a very small probability is assigned to such features.

Probabilities of the entire document

H_1 "D is a SPAM" is closely related to P(D, SPAM):

The probability of document D occurring *and* it being a spam

= P(SPAM) * P(D|SPAM)

= P(SPAM) * P(f_1 , f_2 , … , f_n |SPAM) ❶

= P(SPAM) * P(f_1|SPAM) * P(f_2|SPAM) * … * P(f_n|SPAM) ❷

- We have all the pieces to compute this.
- "Bag-of-words" assumption ❶
- "Naïve" Bayes because ❷ assumes **feature independence**.

  ← Why is this assumption naïve/unreasonable?
- If all we're going to do is rule between SPAM and HAM, we can simply compare P(D, SPAM) and P(D, HAM) and choose one with higher probability.
- But we may also be interested in answering:

  "Given D, what are the *chances* of it being a SPAM? 70%? 5%?"

  ← This is P(SPAM|D).

Given D, chance of Spam?

$$P(SPAM \mid D) = \frac{P(SPAM, D)}{P(D)} = \frac{P(SPAM, D)}{P(SPAM, D) + P(HAM, D)}$$

P(SPAM|D): the probability of a given document D being SPAM

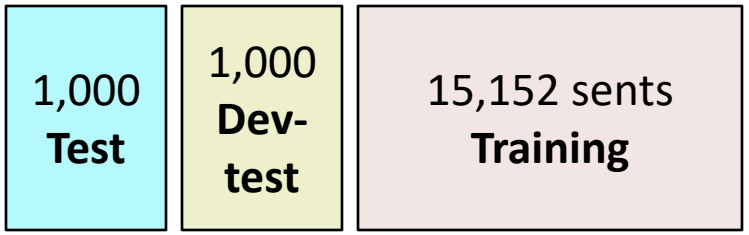
(ex: "This email looks sketchy: 98% chance of spam, 2% OK…")

- Can be calculated from P(SPAM, D) and P(HAM, D)
- More next class… **Bayes Theorem**!

Homework 4: Who Said It?

- Jane Austen or Herman Melville?
  - *I never met with a disposition more truly amiable.*
  - *But Queequeg, do you see, was a creature in the transition stage -- neither caterpillar nor butterfly.*
  - *Oh, my sweet cardinals!*

- Task: build a Naïve Bayes classifier and explore it
- Do three-way partition of data:
  - test data
  - development-test data
  - training data



Wrapping up

- HW 4 out
  - Week-long
    - 10/7 (Thu): [PART A], report on classifier performance
    - 10/12 (Tue): everything due
  - **Don't procrastinate** – start now!
  - This is likely the single most challenging HW!

- Next class (Thu)
  - Bayes Theorem
  - Evaluating performance of a classifier