Naïve Bayes

Data Science: Jordan Boyd-Graber
University of Maryland
MARCH 1, 2018

By the end of today . . .

- You'll be able to frame many standard nlp tasks as classification problems
- Apply logistic regression (given weights) to classify data
- Learn naïve bayes from data

Formal definition of Classification

Given:

- A universe $\mathbb{X}$ our examples can come from (e.g., English documents with a predefined vocabulary)

Formal definition of Classification

Given:

- A universe $\mathbb{X}$ our examples can come from (e.g., English documents with a predefined vocabulary)
  - Examples are represented in this space. (e.g., each document has some subset of the vocabulary; more in a second)

Formal definition of Classification

Given:

- A universe $\mathbb{X}$ our examples can come from (e.g., English documents with a predefined vocabulary)
    - Examples are represented in this space. (e.g., each document has some subset of the vocabulary; more in a second)
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$

Formal definition of Classification

Given:

- A universe $\mathbb{X}$ our examples can come from (e.g., English documents with a predefined vocabulary)
  - Examples are represented in this space. (e.g., each document has some subset of the vocabulary; more in a second)
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \dots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., spam vs. ham).

Formal definition of Classification

Given:

- A universe $\mathbb{X}$ our examples can come from (e.g., English documents with a predefined vocabulary)
  - Examples are represented in this space. (e.g., each document has some subset of the vocabulary; more in a second)
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., spam vs. ham).
- A training set D of labeled documents with each labeled document $d \in \mathbb{X} \times \mathbb{C}$

Formal definition of Classification

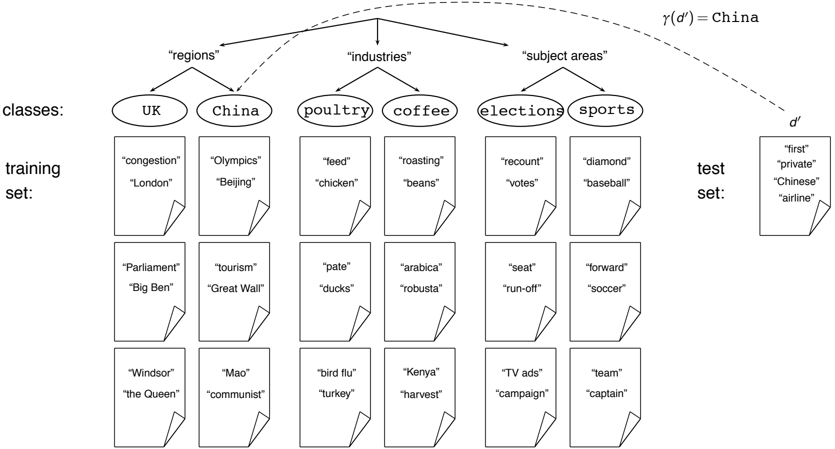
Given:

- A universe $\mathbb{X}$ our examples can come from (e.g., English documents with a predefined vocabulary)
  - Examples are represented in this space. (e.g., each document has some subset of the vocabulary; more in a second)
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \dots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., spam vs. ham).
- A training set D of labeled documents with each labeled document $d \in \mathbb{X} \times \mathbb{C}$

Using a learning method or learning algorithm, we then wish to learn a classifier γ that maps documents to classes:

$$\gamma : \mathbb{X} \rightarrow \mathbb{C}$$

Topic classification



Examples of how search engines use classification

- Standing queries (e.g., Google Alerts)
- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. nonspam)
- The automatic detection of sexually explicit content (sexually explicit vs. not)
- Sentiment detection: is a movie or product review positive or negative (positive vs. negative)
- Topic-specific or vertical search – restrict search to a "vertical" like "related to health" (relevant to vertical vs. not)

Classification methods: 1. Manual

- Manual classification was used by Yahoo in the beginning of the web. Also: ODP, PubMed
- Very accurate if job is done by experts
- Consistent when the problem size and team is small
- Scaling manual classification is difficult and expensive.
- $\longrightarrow$ We need automatic methods for classification.

Classification methods: 2. Rule-based

- There are “IDE” type development enviroments for writing very complex rules efficiently. (e.g., Verity)
- Often: Boolean combinations (as in Google Alerts)
- Accuracy is very high if a rule has been carefully refined over time by a subject expert.
- Building and maintaining rule-based classification systems is expensive.

Classification methods: 3. Statistical/Probabilistic

- As per our definition of the classification problem – text classification as a learning problem
- Supervised learning of a the classification function γ and its application to classifying new documents
- We will look at a couple of methods for doing this: Naive Bayes, Logistic Regression, SVM, Decision Trees
- No free lunch: requires hand-classified training data
- But this manual classification can be done by non-experts.

Generative vs. Discriminative Models

- Goal, given observation x, compute probability of label y, $p(y|x)$
- Naïve Bayes (later) uses Bayes rule to reverse conditioning
- What if we care about $p(y|x)$? We need a more general framework ...

Generative vs. Discriminative Models

- Goal, given observation x, compute probability of label y, $p(y \mid x)$
- Naïve Bayes (later) uses Bayes rule to reverse conditioning
- What if we care about $p(y \mid x)$? We need a more general framework ...
- That framework is called logistic regression (later)
- Naïve Bayes is a special case of logistic regression

A Classification Problem

- Suppose that I have two coins, C_1 and C_2
- Now suppose I pull a coin out of my pocket, flip it a bunch of times, record the coin and outcomes, and repeat many times:

```
C1: 0 1 1 1 1
C1: 1 1 0
C2: 1 0 0 0 0 0 0 1
C1: 0 1
C1: 1 1 0 1 1 1
C2: 0 0 1 1 0 1
C2: 1 0 0 0
```

- Now suppose I am given a new sequence, `0 0 1`; which coin is it from?

A Classification Problem

This problem has particular challenges:

- different numbers of covariates for each observation
- number of covariates can be large

However, there is some structure:

- Easy to get $P(C_1)$, $P(C_2)$
- Also easy to get $P(X_i = 1 \mid C_1)$ and $P(X_i = 1 \mid C_2)$
- By conditional independence,

$$P(X = 0\,1\,0 \mid C_1) = P(X_1 = 0 \mid C_1)P(X_2 = 1 \mid C_1)P(X_2 = 0 \mid C_1)$$

- Can we use these to get $P(C_1 \mid X = 0\,0\,1)$?

A Classification Problem

This problem has particular challenges:

- different numbers of covariates for each observation
- number of covariates can be large

However, there is some structure:

- Easy to get $P(C_1) = 4/7$, $P(C_2) = 3/7$
- Also easy to get $P(X_i = 1 \mid C_1)$ and $P(X_i = 1 \mid C_2)$
- By conditional independence,

$$P(X = 010 \mid C_1) = P(X_1 = 0 \mid C_1) P(X_2 = 1 \mid C_1) P(X_2 = 0 \mid C_1)$$

- Can we use these to get $P(C_1 \mid X = 001)$?

A Classification Problem

This problem has particular challenges:

- different numbers of covariates for each observation
- number of covariates can be large

However, there is some structure:

- Easy to get $P(C_1) = 4/7$, $P(C_2) = 3/7$
- Also easy to get $P(X_i = 1 \,|\, C_1) = 12/16$ and $P(X_i = 1 \,|\, C_2) = 6/18$
- By conditional independence,

$$P(X = 0\,1\,0 \,|\, C_1) = P(X_1 = 0 \,|\, C_1) P(X_2 = 1 \,|\, C_1) P(X_2 = 0 \,|\, C_1)$$

- Can we use these to get $P(C_1 \,|\, X = 0\,0\,1)$?

A Classification Problem

Summary: have $P(data|class)$, want $P(class|data)$

Solution: Bayes' rule!

$$
\begin{aligned}
P(class|data) &= \frac{P(data|class)P(class)}{P(data)} \\
&= \frac{P(data|class)P(class)}{\sum_{class=1}^{C} P(data|class)P(class)}
\end{aligned}
$$

To compute, we need to estimate $P(data|class)$, $P(class)$ for all classes

Naive Bayes Classifier

This works because the coin flips are independent given the coin parameter. What about this case:

- want to identify the type of fruit given a set of features: color, shape and size
- color: red, green, yellow or orange (discrete)
- shape: round, oval or long+skinny (discrete)
- size: diameter in inches (continuous)

Naive Bayes Classifier

Conditioned on type of fruit, these features are not necessarily independent:

Given category “apple,” the color “green” has a higher probability given “size < 2”:

$$P(green \mid size < 2, apple) > P(green \mid apple)$$

Naive Bayes Classifier

Using chain rule,

$$
\begin{aligned}
&P(apple \mid green, round, size = 2) \\
&\qquad = \frac{P(green, round, size = 2 \mid apple) P(apple)}{\sum_{fruits} P(green, round, size = 2 \mid fruit\, j) P(fruit\, j)} \\
&\qquad \propto P(green \mid round, size = 2, apple) P(round \mid size = 2, apple) \\
&\qquad\qquad \times P(size = 2 \mid apple) P(apple)
\end{aligned}
$$

But computing conditional probabilities is hard! There are many combinations of $(color, shape, size)$ for each fruit.

Naive Bayes Classifier

Idea: assume conditional independence for all features given class,

$$P(green \mid round, size = 2, apple) = P(green \mid apple)$$
$$P(round \mid green, size = 2, apple) = P(round \mid apple)$$
$$P(size = 2 \mid green, round, apple) = P(size = 2 \mid apple)$$

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq i \leq n_d} P(w_i|c)$$

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq i \leq n_d} P(w_i|c)$$

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \le i \le n_d} P(w_i|c)$$

- n_d is the length of the document. (number of tokens)
- $P(w_i|c)$ is the conditional probability of term w_i occurring in a document of class c
- $P(w_i|c)$ as a measure of how much evidence w_i contributes that c is the correct class.
- $P(c)$ is the prior probability of c.
- If a document's terms do not provide clear evidence for one class vs. another, we choose the c with higher $P(c)$.

Maximum a posteriori class

- Our goal is to find the “best” class.
- The best class in Naive Bayes classification is the most likely or maximum a posteriori (MAP) class c_{map} :

$$c_{\text{map}} = \arg\max_{c_j \in \mathbb{C}} \hat{P}(c_j|d) = \arg\max_{c_j \in \mathbb{C}} \hat{P}(c_j) \prod_{1 \le i \le n_d} \hat{P}(w_i|c_j)$$

- We write $\hat{P}$ for P since these values are estimates from the training set.

Naive Bayes Classifier

Why conditional independence?

- estimating multivariate functions (like $P(X_1, \ldots, X_m \mid Y)$) is mathematically hard, while estimating univariate ones is easier (like $P(X_i \mid Y)$)
- need less data to fit univariate functions well
- univariate estimators differ much less than multivariate estimator (low variance)
- ... but they may end up finding the wrong values (more bias)

Naïve Bayes conditional independence assumption

To reduce the number of parameters to a manageable size, recall the Naive Bayes conditional independence assumption:

$$P(d|c_j) = P(\langle w_1, \ldots, w_{n_d} \rangle | c_j) = \prod_{1 \leq i \leq n_d} P(X_i = w_i | c_j)$$

We assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(X_i = w_i | c_j)$.
Our estimates for these priors and conditional probabilities: $\hat{P}(c_j) = \frac{N_c + 1}{N + |C|}$ and $\hat{P}(w|c) = \frac{T_{cw} + 1}{(\sum_{w' \in V} T_{cw'}) + |V|}$

Implementation Detail: Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- From last time lg is logarithm base 2; ln is logarithm base e.

$$\lg x = a \Leftrightarrow 2^a = x \qquad \ln x = a \Leftrightarrow e^a = x \tag{1}$$

- Since $\ln(xy) = \ln(x) + \ln(y)$, we can sum log probabilities instead of multiplying probabilities.
- Since ln is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \arg\max_{c_j \in \mathbb{C}} [\hat{P}(c_j) \prod_{1 \leq i \leq n_d} \hat{P}(w_i|c_j)]$$

$$\arg\max_{c_j \in \mathbb{C}} [\ln \hat{P}(c_j) + \sum_{1 \leq i \leq n_d} \ln \hat{P}(w_i|c_j)]$$

Implementation Detail: Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- From last time lg is logarithm base 2; ln is logarithm base e.

$$\lg x = a \Leftrightarrow 2^a = x \qquad \ln x = a \Leftrightarrow e^a = x \tag{1}$$

- Since $\ln(xy) = \ln(x) + \ln(y)$, we can sum log probabilities instead of multiplying probabilities.
- Since ln is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \arg\max_{c_j \in \mathbb{C}} [\hat{P}(c_j) \prod_{1 \leq i \leq n_d} \hat{P}(w_i|c_j)]$$

$$\arg\max_{c_j \in \mathbb{C}} [\ln \hat{P}(c_j) + \sum_{1 \leq i \leq n_d} \ln \hat{P}(w_i|c_j)]$$

Implementation Detail: Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- From last time lg is logarithm base 2; ln is logarithm base *e*.

$$\lg x = a \Leftrightarrow 2^a = x \qquad \ln x = a \Leftrightarrow e^a = x \tag{1}$$

- Since $\ln(xy) = \ln(x) + \ln(y)$, we can sum log probabilities instead of multiplying probabilities.
- Since ln is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \arg\max_{c_j \in \mathbb{C}} [\hat{P}(c_j) \prod_{1 \leq i \leq n_d} \hat{P}(w_i | c_j)]$$

$$\arg\max_{c_j \in \mathbb{C}} [\ln \hat{P}(c_j) + \sum_{1 \leq i \leq n_d} \ln \hat{P}(w_i | c_j)]$$

Contrasting Naïve Bayes and Logistic Regression

- Naïve Bayes easier
- Naïve Bayes better on smaller datasets
- Logistic regression better on medium-sized datasets
- On huge datasets, it doesn't really matter (data always win)
  - Optional reading by Ng and Jordan has proofs and experiments
- Logistic regression allows arbitrary features (biggest difference!)