Naive Bayes

Jeffrey Arnold

4/3/2018

Naive Bayes

Bayes' theorem can almost immediately be supervised classification algorithms. The Naive Bayes classifiers are a family of classifiers which apply Bayes' Rule to classify a discrete response y using observed features $(x_1, \dots, x_K)$, with a simplifying assumption of independence.

Suppose that y is the class of an observation; i.e., it is a discrete variable taking values $j \in 1, \dots, J$. Suppose that $(x_1, \dots, x_k)$ is a vector of K observed features (predictors) for an observation. These can be discrete or continuous. We are interested in the probabilities of each class after having observed its features, which we can reformulate in Bayes' rule.

$$p(y|x_1, \dots, x_k) = \frac{p(x_1, \dots, x_k|y)p(y)}{\sum_{j=1}^J p(x_1, \dots, x_k|y = j)p(y = j)}$$

The "naive" modifier in "naive Bayes" comes from an additional asumption that distributions of features are independent conditional on the class,

$$p(x_k|y, x_1, \dots, x_K) = p(x_k|y)$$

for all $k \in 1, \dots, K$. This independence is a strong one, but will make this problem much more tractable. It is much easier to model and estimate the univariate $p(x_k|y)$ probabilities, but much harder to model and estimate a K-variate distribution, $p(x_1, \dots, x_K|y)$.

Using independence, we can rewrite the posterior distribution as,

$$p(y|x_1, \dots, x_k) = \frac{p(y)p(x_1|y) \cdots p(x_K|y)}{\sum_{j=1}^J p(y)p(x_1|y) \cdots p(x_K|y)} = \frac{p(y) \prod_{k=1}^K p(x_k|y)}{\sum_{j=1}^J p(y) \prod_{k=1}^K p(x_k|y)}.$$

Moreover, often interested in the most likely class, where we can ignore the marginal likelihood,

$$\begin{aligned} \arg \max_{j \in 1, \dots, J} p(y = j|x_1, \dots, x_k) &= \frac{p(y = j) \prod_{k=1}^K p(x_k|y = j)}{\sum_{j=1}^J p(y = j) \prod_{k=1}^K p(x_k|y = j)} \\ &\propto p(y = j) \prod_{k=1}^K p(x_k|y = j). \end{aligned}$$

In applying naive Bayes, there are two choices that need to be made:

1. probability distributions for likelihood of each feature, $p(x_i|y)$, and
2. prior distribution $p(y)$.

After choosing the distributional forms of $p(y)$, $p(x_1|y)$, …, $p(x_K|y)$ appropriate for your model, the workflow is,

1. Train your model on data $(x_1, \dots, x_k,\ y)$ to estimate the distributions $\hat{p}(y)$, $\hat{p}(x_1|y)$, …, $\hat{p}(x_K|y)$.
2. For new observations, calculate $p(y|x)$ using the learned parameters of the distribution.
3. Evaluate the model using predictive criteria

In our cases we will be using *maximum a posteriori* estimators to find the parameters of $\hat{p}(y)$, $\hat{p}(x_1|y)$, …, $\hat{p}(x_K|y)$. What makes this convenient is that the MAP estimator can be estimated separately for each term.

Federalist Papers

The Federalist Papers comprise 85 articles published under the pseudonym "Publius" in New York newspapers between 1787 and 1788. It was written by Alexander Hamilton, James Madison, and John Jay to persuade the public to ratify the Constitution. John Jay wrote five papers, and Alexander Hamilton wrote 51, and James Madison 14. The authorship of the remaining 15 papers is (was) disputed between Hamilton and Madison, though lar

In an early example of empirical Bayesian statistics and computational NLP, F. Mosteller and D. L. Wallace used naive Bayes to classify the disputed articles and conclude that there is strong evidence to suggest that Madison wrote all the disputed articles.

This example will use the following libraries.

```r
library("corpus")
library("tidytext")
library("tidyverse")
```

Load the text of Federalist papers from the **corpus package**.

```r
data("federalist", package = "corpus")

federalist <- federalist %>%
  # add a document number
  mutate(number = row_number())
```

We will often only be referring to Hamilton and Madison and ignore John Jay, so assign their names to a variable that we will use several times.

```r
AUTHORS <- c("Hamilton", "Madison")
```

Create a data frame of document-term counts,

```r
federalist_wc <- term_counts(federalist) %>%
  mutate(text = as.integer(text),
         term = as.character(term)) %>%
  left_join(select(federalist, text = number, author),
            by = "text")
```

The features we will use for paper is the

```r
federalist_wc <- federalist_wc %>%
  filter(term %in% corpus::stopwords_en)
```

Having seen the data, let's turn to our analysis.

We have D documents. Let $y \in Y = \{\text{Hamilton}, \text{Madison}\}$ be the author of each document. We want to infer the probability of an author given that

$$p(y|d) = \frac{p(d|y)p(y)}{\sum_{y \in Y} p(d|y)p(y)}.$$

The next question is how to represent and model documents, meaning which features we should use and the distribution we should use for those.

Suppose that each document can be represented as a vector of word counts. Suppose that each document is represented as a vector of word counts. Suppose that the vocabularly in the corpus consists of K distinct words, e.g. "a", "an", ... Each document can be represented as $(w_1, \dots, w_K)$, where w_k is the counts of word k in the document. Let $N_d = \sum_{k=1}^{K} w_k$ be the total number of words in document d. Then conditional probability of a document given its class can be modeled as a multinomial distribution,

$$p(d|y) = \text{Multinomial}(y; \theta_1, \dots, \theta_K, N_d) = \frac{N_d!}{w_1! \cdots w_K!} \theta_1^{w_1} \cdots \theta_K^{w_K}$$

where $\theta_k \in 1, \dots, K$ is the probability that word k occurs. The values that we will be interested in calculating are $\hat{\theta}_1, \dots, \hat{\theta}_K$.

For a corpus, we can calculate these parameters with the maximum a posteriori estimator,

$$\hat{\theta}_{k,y=j} = \frac{\sum_{d \in D_j} w_{k,d} + 1}{\sum_{d \in D_j} N_d + V} =$$

The estimator $\hat{\theta}_{k,y}$ is the fraction of times that word k appears among all words in all documents in a category.

For the priors $p(y)$ we will use the proportion of documents in each class,

$$\hat{\gamma}_j = p(y = j) = \frac{D_j}{D}$$

The value of $\hat{\gamma}$ is the fraction of documents in category j to all documents.

```r
p_words_author <- federalist_wc %>%
  # keep only Hamilton and Madison
  filter(author %in% AUTHORS) %>%
  # count terms used by each author
  group_by(author, term) %>%
  summarise(count = sum(count)) %>%
  ungroup() %>%
  # ensure all (author, term) combinations appear
  # fill in missing combinations with count = 0
  complete(author, term, fill = list(count = 0)) %>%
  # calculate p(w | c) for each author
  group_by(author) %>%
  mutate(p = (count + 1) / sum(count + 1))
```

For the priors on categories we can use the prior generated from the proportion of documents in each class,

$$\hat{p}(c) = \frac{D_c}{D},$$

where D_c is the number of documents in class c, and D is the total number of documents.

```r
p_author <- federalist %>%
  filter(author %in% AUTHORS) %>%
  count(author) %>%
  mutate(p = n / sum(n),
         # log-probabilites are easier to work with
         logp = log(p))
p_author
```

```
## # A tibble: 2 x 4
##   author       n     p   logp
##   <chr>    <int> <dbl>  <dbl>
## 1 Hamilton    51 0.785 -0.243
## 2 Madison     14 0.215 -1.54
```

Use Bayes' rule to find the posterior probability of each document:

```r
post <- p_words_author %>%
  select(author, term, p) %>%
  # start with p_word_cat so that all (term, author)
  # combinations are represented
  left_join(select(federalist_wc, text, term, count),
            by = c("term")) %>%
  mutate(count = if_else(is.na(count), 0, count)) %>%
  # calculate density of multinomial distribution for each probability distribution
  group_by(author, text) %>%
  arrange(author, text, term) %>%
  summarise(logp_words_author = dmultinom(count, prob = p, log = TRUE))
```

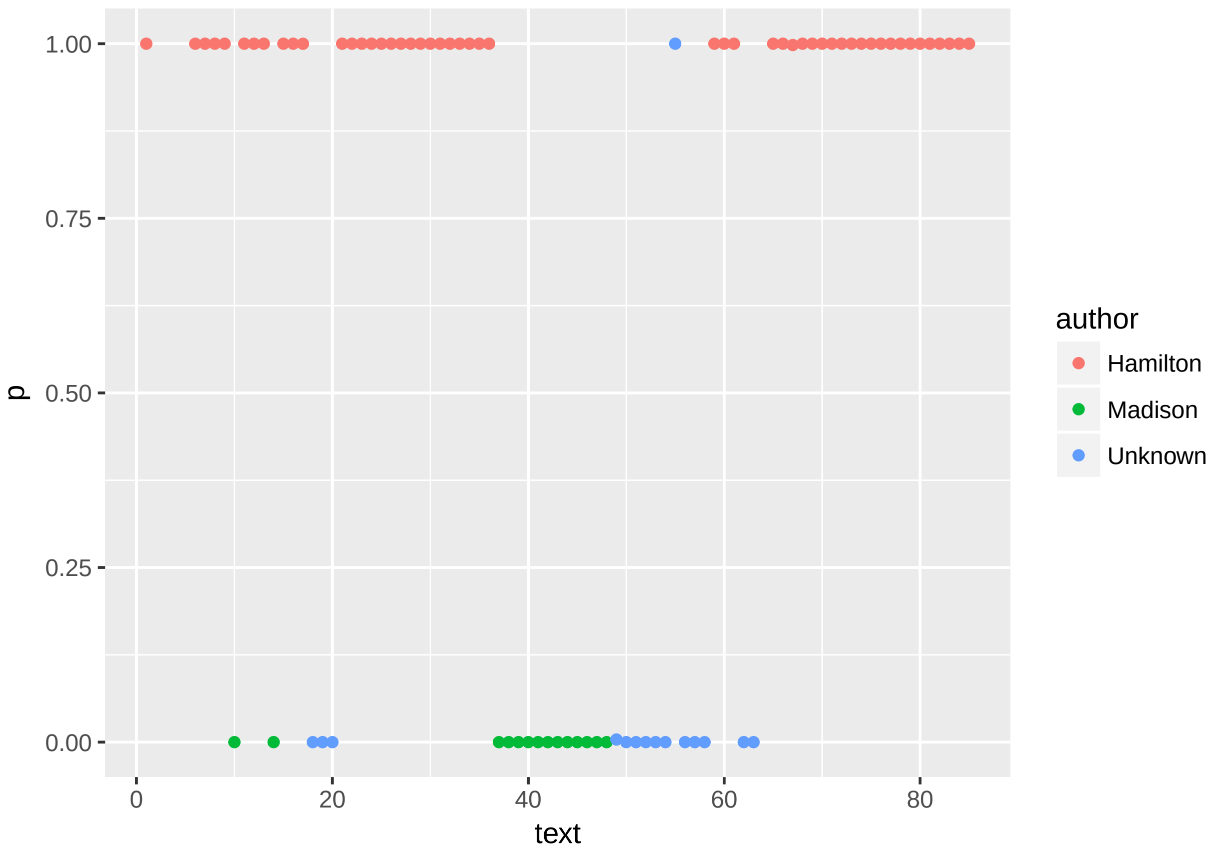
Add a column with priors,

```r
post <- left_join(post,
          select(p_author, author, logp_author = logp), by = "author") %>%
  # log(p(c)) + log(p(w | c))
  mutate(p_author_words = exp(logp_words_author + logp_author)) %>%
  # normalize to probabilities
  group_by(text) %>%
  mutate(p_author_words = p_author_words / sum(p_author_words))
```

Find the most probable document:

```r
predictions <- post %>%
  select(author, text, p_author_words) %>%
  # For each document, choose the author with the highest probability:
  group_by(text) %>%
  arrange(text, desc(p_author_words)) %>%
  slice(1) %>%
  rename(pred = author) %>%
  # add labels
  left_join(select(federalist, text = number, author),
            by = "text") %>%
  # using p(author = Hamilton) will be easier to interpret later
  mutate(p = p_author_words,
         p = if_else(pred == "Madison", 1 - p, p),
         author = if_else(is.na(author), "Unknown", author)) %>%
  filter(author != "Jay")
```

```r
ggplot(predictions, aes(x = text, colour = author, y = p)) +
  geom_point()
```



References

- Scikit-Learn, Naive Bayes
- Harry Zhang (2004) The Optimality of Naive Bayes.
- https://web.stanford.edu/class/cs124/lec/naivebayes.pdf