Introduction to Machine Learning

Computational Linguistics: Jordan Boyd-Graber
University of Maryland
NAÏVE BAYES AND LOGISTIC REGRESSION

Slides adapted from Hinrich Schütze and Lauren Hannah

By the end of today ...

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

Probabilistic Classification

Given:

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

Probabilistic 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.

Probabilistic 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.
- A fixed set of labels $y \in \mathbb{C} = \{c_1, c_2, \dots, c_J\}$

Probabilistic 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.
- A fixed set of labels $y \in \mathbb{C} = \{c_1, c_2, \ldots, c_J\}$
  - The classes are human-defined for the needs of an application (e.g., spam vs. ham).

Probabilistic 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.
- A fixed set of labels $y \in \mathbb{C} = \{c_1, c_2, \ldots, c_J\}$
  - 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 $\{(x_1, y_1) \ldots (x_N, y_N)\}$

Probabilistic 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.
- A fixed set of labels $y \in \mathbb{C} = \{c_1, c_2, \dots, c_J\}$
  - 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 $\{(x_1, y_1) \dots (x_N, y_N)\}$

We learn a classifier γ that maps documents to class probabilities:

$$\gamma : (x, y) \rightarrow [0, 1]$$

such that $\sum_y \gamma(x, y) = 1$

Generative vs. Discriminative Models

Generative

Model joint probability $p(x,y)$ including the data x.

Naïve Bayes

- Uses Bayes rule to reverse conditioning $p(x|y) \rightarrow p(y|x)$
- Naïve because it ignores joint probabilities within the data distribution

Discriminative

Model only conditional probability $p(y|x)$, excluding the data x.

Logistic regression

- Logistic: A special mathematical function it uses
- Regression: Combines a weight vector with observations to create an answer
- General cookbook for building conditional probability distributions

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
```

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,

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

- 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 $P(X_i = 1 \mid C_1)$ 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 $P(X_i = 1 \mid C_1) = 12/16$ and $P(X_i = 1 \mid C_2) = 6/18$
- 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

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,

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

How do we estimate a probability?

- Suppose we want to estimate $P(w_n = \text{``}buy\text{''} | y = \texttt{SPAM})$.

How do we estimate a probability?

- Suppose we want to estimate $P(w_n = \text{“}buy\text{''} | y = \texttt{SPAM})$.

buy	**buy**	nigeria	opportunity	viagra
nigeria	opportunity	viagra	fly	money
fly	**buy**	nigeria	fly	**buy**
money	**buy**	fly	nigeria	viagra

How do we estimate a probability?

- Suppose we want to estimate $P(w_n = \text{``}buy\text{''} | y = \texttt{SPAM})$.

buy	**buy**	nigeria	opportunity	viagra
nigeria	opportunity	viagra	fly	money
fly	**buy**	nigeria	fly	**buy**
money	**buy**	fly	nigeria	viagra

- Maximum likelihood (ML) estimate of the probability is:

$$\hat{\beta}_i = \frac{n_i}{\sum_k n_k} \tag{1}$$

How do we estimate a probability?

- Suppose we want to estimate $P(w_n = \text{“}buy\text{”} | y = \texttt{SPAM})$.

buy	**buy**	nigeria	opportunity	viagra
nigeria	opportunity	viagra	fly	money
fly	**buy**	nigeria	fly	**buy**
money	**buy**	fly	nigeria	viagra

- Maximum likelihood (ML) estimate of the probability is:

$$\hat{\beta}_i = \frac{n_i}{\sum_k n_k} \tag{1}$$

- Is this reasonable?

The problem with maximum likelihood estimates: Zeros (cont)

- If there were no occurrences of "bagel" in documents in class SPAM, we'd get a zero estimate:

$$\hat{P}(\text{"bagel"} | \texttt{SPAM}) = \frac{T_{\texttt{SPAM}, \text{"bagel"}}}{\sum_{w' \in V} T_{\texttt{SPAM}, w'}} = 0$$

- $\rightarrow$ We will get $P(\texttt{SPAM}|d) = 0$ for any document that contains bagel!
- Zero probabilities cannot be conditioned away.

How do we estimate a probability?

- For many applications, we often have a prior notion of what our probability distributions are going to look like (for example, non-zero, sparse, uniform, etc.).
- This estimate of a probability distribution is called the maximum a posteriori (MAP) estimate:

$$\beta_{\text{MAP}} = \text{argmax}_{\beta} f(x|\beta)g(\beta) \tag{2}$$

How do we estimate a probability?

- For a multinomial distribution (i.e. a discrete distribution, like over words):

$$\beta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$

- α_i is called a smoothing factor, a pseudocount, etc.

How do we estimate a probability?

- For a multinomial distribution (i.e. a discrete distribution, like over words):

$$\beta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$

- α_i is called a smoothing factor, a pseudocount, etc.
- When $\alpha_i = 1$ for all i, it's called "Laplace smoothing" and corresponds to a uniform prior over all multinomial distributions (just do this).

How do we estimate a probability?

- For a multinomial distribution (i.e. a discrete distribution, like over words):

$$\beta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$

- α_i is called a smoothing factor, a pseudocount, etc.
- When $\alpha_i = 1$ for all i, it's called "Laplace smoothing" and corresponds to a uniform prior over all multinomial distributions (just do this).
- To geek out, the set $\{\alpha_1, \ldots, \alpha_N\}$ parameterizes a Dirichlet distribution, which is itself a distribution over distributions and is the conjugate prior of the Multinomial (don't need to know this).

The Naïve Bayes classifier

- The Naïve 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 Naïve Bayes classifier

- The Naïve 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 Naïve Bayes classifier

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

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

- 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 Naïve Bayes classification is the most likely or maximum a posteriori (MAP) class c_{map} :

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

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

Naïve Bayes conditional independence assumption

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

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

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.

$$\lg x = a \Leftrightarrow 2^a = x \qquad \ln x = a \Leftrightarrow e^a = x \tag{4}$$

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

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

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

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.

$$\lg x = a \Leftrightarrow 2^a = x \qquad \ln x = a \Leftrightarrow e^a = x \tag{4}$$

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

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

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

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.

$$\lg x = a \Leftrightarrow 2^a = x \qquad \ln x = a \Leftrightarrow e^a = x \tag{4}$$

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

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

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