Naive Bayes

Astrini Sie

asie@seattleu.edu

Week 6b

ECEGR4750 - Introduction to Machine Learning
Seattle University

October 26, 2023

Recap and Updates

- Paper Presentation this Tuesday and Thursday
- Office Hours
  - T, Th 12-1p at Bannan 224
  - W 7-9p via Zoom
  - F 9-9.45a via Zoom
- Zoom Link: `https://seattleu.zoom.us/j/7519782079?pwd=cnhCM2tPcHJKVWwxZVArS2VHSUNJZz09`
  - Meeting ID: 751 978 2079
  - Passcode: 22498122

Overview

Generative vs. Discriminative Learning Algorithms

Consider a classification problem where we want to distinguish between cats ($y = 0$) and dogs ($y = 1$).

- **Discriminative Learning Algorithms** (like Logistic Regression):
  - Training: Find a straight line (decision boundary) that separates the data of cats and dogs.
  - Inference: Checks which side of the decision boundary a new data point falls.

  Tries to *discriminate* between features of different classes.
- **Generalized Learning Algorithms**:
  - Training: Build a model of what cats look like, and another model of what dogs look like.
  - Inference: Match the new data against the cat model, and then against the dog model, to see whether the new data looks more like the cats or the dogs from the training set.

  Tries to *generalizes* the representation of each class instead of finding the discriminating fatures between them.

Generative vs. Discriminative Learning Algorithms

Consider a classification problem where we want to distinguish between cats ($y = 0$) and dogs ($y = 1$).

- **Discriminative Learning Algorithms** (like Logistic Regression): Tries to learn probability of y given x $p(y|x)$ directly, or tries to learn mappings from the input space X to the label space $0, 1$.
- **Generative Learning Algorithms**: Tries to model $p(x|y)$ and $p(y)$. For example, if $y = 0$ is cats and $y = 1$ is dogs, $p(x|y = 0)$ models the distribution of cats, and $p(x|y = 1)$ models the distribution of dogs.

Bayes Rule

After modeling $p(x|y)$ and $p(y)$, we can use Bayes rule to derive the posterior distribution on y given x:

Bayes Rule

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

where:

- $p(x) \neq 0$, and $p(x) = p(x|y=1)p(y=1) + p(x|y=0)p(y=0)$
- $p(y)$ is the *priori* (probability of event y before the evidence x is seen).
- $p(x|y)$ is the *likelihood* (likelihood of evidence x given envent y).
- $p(y|x)$ is the *posteriori* (probability of event y after the evidence x is seen).

Bayes Rule

If we are calculating $p(y|x)$ to make a prediction, $p(x)$ is a constant with respect to the variable y, so we could say that $p(y|x)$ is proportional to $p(x|y)p(y)$ with respect to the variable y:

$$
\begin{aligned}
\arg\max_y p(y|x) &= \arg\max_y \frac{p(x|y)p(y)}{p(x)} \\
&\propto \arg\max_y p(x|y)p(y)
\end{aligned}
$$

Maximum Likelihood Estimate (MLE)

From Week 4 Logistic Regression, recall, Likelihood $L(\theta)$:

$$\begin{aligned} L(\theta) &= P(y|X;\theta) \\ &= \prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta) \\ &= \prod_{i=1}^{n} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

Note that:

$$p(y|x;\theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

because when $y = 1$, $p(y|x;\theta) = h_\theta(x)$ and
when $y = 0$, $p(y|x;\theta) = (1 - h_\theta(x))$

Maximum Likelihood Estimate (MLE)

And we solve for θ that maximizes the likelihood function:

MLE

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} p\left(y^{(i)} | x^{(i)}; \theta\right)$$

θ is viewed as a *constant valued, but unknown parameter*. This is the **frequentist** view of the world – θ is not random, but unknown.

Maximum A Posteriori Estimate (MAP)

An alternative take is the **Bayesian** view of the world – θ is a *random variable whose value is unknown*. In this approach, we specify a prior distribution $p(\theta)$ on θ that expresses our prior beliefs about the parameter.

Given a training set $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, when we are asked to make a prediction on a new value x, we can compute the posterior distribution on the parameters

$$\begin{aligned} p(\theta|S) &= \frac{p(S|\theta)p(\theta)}{p(S)} \\ &= \frac{\left(\prod_{i=1}^{n} p(y^{(i)}|x^{(i)}, \theta)\right) p(\theta)}{\int_{\theta} \left(\prod_{i=1}^{n} p(y^{(i)}|x^{(i)}, \theta) p(\theta)\right) d\theta} \end{aligned}$$

Maximum A Posteriori Estimate (MAP)

And then we can compute the posterior distribution on the class label using the posterior distribution on θ:

$$p(y|x, S) = \int_\theta p(y|x, \theta)p(\theta|S)d\theta$$

It is computationally very difficult to compute the posterior distribution $p(\theta|S)$ over θ, hence in practice we will approximate the posterior distribution for θ with a single point estimate. The Maximum A Posteriori (MAP) estimate for θ is given by:

MAP

$$\theta_{MAP} = \arg\max_\theta \prod_{i=1}^{n} p(y^{(i)}|x^{(i)}, \theta)p(\theta)$$

MLE and MAP Similarities

Notice that θ_{MLE} is the same as θ_{MAP} except for the prior term $p(\theta)$ at the end.

In practical applications, a common choice for $p(\theta)$ is to assume that $\theta \sim \mathcal{N}(0, \tau^2 I)$ (uniform distribution), thus making $p(\theta)$ a constant.

MLE is a special case of MAP, when the prior is a uniform distribution.

The Bayesian MAP estimate is less susceptible to overfitting than the MLE, especially when the dataset is small.

Naive Bayes

Let's build an email spam filter.

Training set: a set of emails with features x_j labeled as spam $y = 1$ and not-spam $y = 0$.

Naive Bayes

Let's build an email spam filter.

0. A training sample $x^{(i)}$ (an email) is a feature vector with length d equal to the number of words in the dictionary. If the email consists of the j-th word in the dictionary, $x_j = 1$ otherwise $x_j = 0$. For example if the email contains the words: "buy a watch", the input vector will be:

$$x^{(i)} = \begin{array}{c c} \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} & \begin{array}{c} a \\ aardvark \\ aardwolf \\ \vdots \\ buy \\ \vdots \\ watch \\ \vdots \\ zygmurgy \end{array} \end{array}$$

Naive Bayes

Let's build an email spam filter.

1. Model $p(x|y)$. **Assumption:** We will assume that x_js are conditionally independent given y.

   This is called a **Naive Bayes assumption** and the resulting algorithm is called the Naive Bayes classifier.

   For example:

   - Given spam email $y = 1$ and "buy" is word 2087 and "watch" is word 49831
   - If I tell you that $y = 1$, then the knowledge of x_{2078} (whether "buy" appears in the email) will have no effect on your beliefs about the value of x_{49831} (knowledge if "watch" appears in the email).
   - $p(x_{2087}|y) = p(x_{2087}|y, x_{49831})$.
   - This is not the same as saying that x_{2087} and x_{49831} are independent!

Naive Bayes

Let's build an email spam filter.

1. Model $p(x|y)$. **Assumption:** We will assume that x_js are conditionally independent given y.

$$\begin{aligned} p(x_1, \ldots, x_d|y) &= p(x_1|y)p(x_2|y, x_1)\ldots p(x_d|y, x_1, x_2, \ldots, x_{d-1}) \\ &= p(x_1|y)p(x_2|y)\ldots p(x_d|y) \\ &= \prod_{j=1}^{d} p(x_j|y) \end{aligned}$$

Naive Bayes

Let's build an email spam filter.

2. Prediction $p(y|x)$.
   As the denominator remain constant for all entries in the dataset it can be removed.

$$\begin{aligned} p(y|x) &= \frac{p(x|y)p(y)}{p(x)} \\ &= \frac{\left(\prod_{j=1}^{d} p(x_j|y)\right) p(y)}{\prod_{j=1}^{d} p(x_j)} \\ &\propto p(y) \prod_{j=1}^{d} p(x_j|y) \end{aligned}$$

Naive Bayes

Let's build an email spam filter.

2. Prediction $p(y|x)$.
   Pick the class with a higher posterior probability.

$$y = \arg\max_y p(y) \prod_{j=1}^{d} p(x_j|y)$$

Laplace Smoothing

What if one of the classes is never seen before?
$p(x) = 0 \rightarrow p(y|x) = 0$
This is called a “zero-frequency problem”. To avoid this, we can use **Laplace Smoothing**.

Laplace Smoothing

$$p(z_j|y = 1) = \frac{\sum_{j=1}^{n} 1\{z_j = j\} + \alpha}{n_{y=1} + \alpha * j}$$

where:

- α is the smoothing parameter
- z is the number of samples where $z = j$ and $y = 1$
- $n_{y=1}$ is the number of counts where $y = 1$

Pros and Cons

Pros

- Works very fast and simple
- Performs better than other models when training data is small

Cons

- Only works when the assumption of *independence features given output* is true
- Zero-frequency problem
- Not great for imbalanced dataset

References

Chris Re, Andrew Ng, and Tengyu Ma (2023)
CSE229 Machine Learning
Stanford University