S21CSE471: Statistical Methods in AI **Spring 2021**

Naive Bayes

Prepared by: Rani Tresa, Rishi Tripathi, Ronit Ray

1 Bayes' Theorem

1. It finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)} \tag{1}$$

$P(c \mid x)$ is the posterior probability of target(c) given predictor (x).

P(c) is the prior probability of target.

$P(x \mid c)$ is the likelihood which is the probability of predictor given class.

P(x) is the prior probability of predictor.

2 The Gaussian Discriminant Analysis model

1. When we have a classification problem in which the input features x are continuous-valued random variables, we can then use the Gaussian Discriminant Analysis (GDA) model, which models $p(x|y)$ using a multivariate normal distribution.

The model is

$$\begin{aligned} y &\sim \text{Bernoulli}(\phi) \\ x \mid y = 0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x \mid y = 1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

Writing out the distributions, this is:

$$\begin{aligned} p(y) &= \phi^y (1-\phi)^{1-y} \\ p(x \mid y = 0) &= \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T \Sigma^{-1}(x-\mu_0)\right) \\ p(x \mid y = 1) &= \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T \Sigma^{-1}(x-\mu_1)\right) \end{aligned}$$

Here, the parameters of our model are $\phi, \sigma, \mu_0 \text{and} \mu_1$.

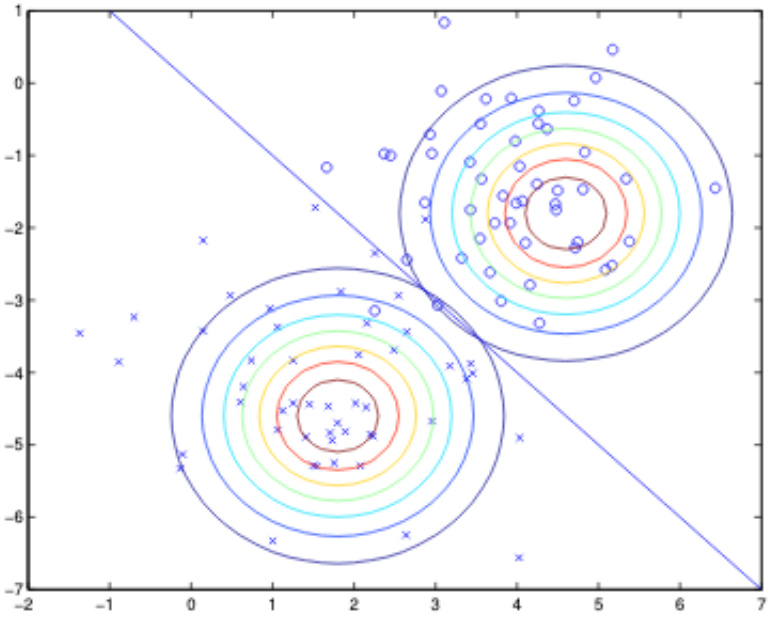
The log-likelihood of the data is given by

$$\begin{aligned} \ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{n} p\left(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma\right) \\ &= \log \prod_{i=1}^{n} p\left(x^{(i)} \mid y^{(i)}; \mu_0, \mu_1, \Sigma\right) p\left(y^{(i)}; \phi\right) \end{aligned}$$

By maximizing l with respect to the parameters, we find the maximum likelihood estimate of the parameters to be:

$$\phi = \frac{1}{n}\sum_{i=1}^{n} 1\left\{y^{(i)} = 1\right\}$$
$$\mu_0 = \frac{\sum_{i=1}^{n} 1\left\{y^{(i)} = 0\right\} x^{(i)}}{\sum_{i=1}^{n} 1\left\{y^{(i)} = 0\right\}}$$
$$\mu_1 = \frac{\sum_{i=1}^{n} 1\left\{y^{(i)} = 1\right\} x^{(i)}}{\sum_{i=1}^{n} 1\left\{y^{(i)} = 1\right\}}$$
$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}\left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T.$$

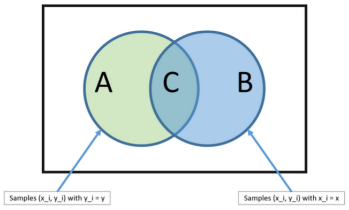
Pictorially, what the algorithm is doing can be seen in as follows:



Shown in the figure are the training set, as well as the contours of the two Gaussian distributions that have been fit to the data in each of the two classes. Note that the two Gaussians have contours that are the same shape and orientation, since they share a covariance matrix σ, but they have different means $\mu_0$andμ_1. Also shown in the figure is the straight line giving the decision boundary at which $p(y = 1|x) = 0.5$. On one side of the boundary, we'll predict $y = 1$ to be the most likely outcome, and on the other side, we'll predict $y = 0$.

3 Naive Bayes Classifier

1. Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem.
2. The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the outcome.
3. They are probabilistic classifiers, therefore will calculate the probability of each category using Bayes theorem, and the category with the highest probability will be output
4. Naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector.we are primarily interested in predicting the label y from the features x, we may estimate P(Y—X)



The Venn diagram illustrates that the MLE method estimates

`P(y|x)=|C|/|B|`

$P(y \mid X) = \frac{P(X|y)P(y)}{P(X)}$
where, y is class variable and X is a dependent feature vector (of size n) where:

$X = (x_1, x_2 ... x_n)$

Now, we put the naive assumption to the Bayes' theorem, which is, independence among the features. So now, we split evidence into the independent parts.

As we know, if any two events A and B are independent, then
$P(A, B) = P(A)P(B)$

Thus,
$P(\mathbf{x}|y) = \prod_{\alpha=1}^{d} P(x_\alpha|y)$, where $x_\alpha = [\mathbf{x}]_\alpha$ is the value for feature α

Hence, we reach to the result:
$P\left(y \mid x_1, \ldots, x_n\right) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$

which can be expressed as:
$P\left(y \mid x_1, \ldots, x_n\right) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$

Now, as the denominator remains constant for a given input, we can remove that term:
$P\left(y \mid x_1, \ldots, x_n\right) \propto P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right)$

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability.

This can be expressed mathematically as:
$y = \text{argmax}_y P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right)$

4 Gaussian Naive Bayes classifier

1. In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution

   The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

   $P\left(x_i \mid y\right) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$

5 Multinomial Naive Bayes

1. This is mostly used for document classification problem

   For example,

   $P(Dear|Normal) = 8/17 = 0.47$

   $P(Friend|Normal) = 5/17 = 0.29$

   $P(Lunch|Normal) = 3/17 = 0.18$

   $P(Money|Normal) = 1/17 = 0.06$

   $P(Dear|Spam) = 2/7 = 0.29$

   $P(Friend|Spam) = 1/7 = 0.14$

   $P(Lunch|Spam) = 0/7 = 0.00$

   $P(Money|Spam) = 4/7 = 0.57$

   From our initial assumptions of 8 Normal messages and 4 Spam messages

   $P(Normal) = 8/(8+4) = 0.67$

   $P(Spam) = 1 - P(Normal) = 0.33$

   We have received a normal message as "Dear Friend" and we want to find out if it's a normal message or spam

   P (Spam | Dear Friend) $= P$ (Dear Friend | Spam)*P(Spam)
   $= P$(Dear | Spam)*P(Friend | Spam)*P(Spam)
   $= 0.47^{*}0.29^{*}0.67$
   $= 0.09$

   P (Normal) | Dear Friend) = P (Dear Friend | Normal) *P (Normal)
   = P(Dear | Normal)*P(Friend | Normal)*P(Normal)
   $= 0.29^{*}0.14^{*}0.33$
   $= 0.01$

   $P(Spam|DearFriend) > P(Normal|DearFriend)$

   We can conclude that Dear Friend is a normal message

2. We have received a normal message as "Lunch Money Money Money" and we want to find out if it's a normal message or spam

   $P(Spam \mid LunchMoneyMoneyMoney)$

   = P (Lunch Money Money Money | Spam)*P(Spam)
   = P(Lunch | Spam)*P(Money | Spam)^{3}*P(Spam)
   $= 0.00^{\star}0.57^{3*}0.67$
   $= 0$

   $P(Normal \mid LunchMoneyMoneyMoney)$
   $= P(LunchMoneyMoneyMoney \mid Normal) * P(Normal)$
   $= P(Lunch|Normal) * P(Money|Normal)3 * P(Normal)$

$= 0.18 * 0.063 * 0.33$
$= 0.000013$

$P(Normal|LunchMoneyMoneyMoney) > P(Spam|LunchMoneyMoneyMoney)$

We can intuitively see that it must be spam but it is not so because the $P(Lunch|Spam)$ is 0 and it doesn't matter if a message contains any other words.

If the word Lunch is present, then no matter the message will always be classified as Normal.

To work around this, we use Laplacian smoothing and add minimum value to each probability so that the probability of observing a word we never get 0.

On adding minimum value(0.1) we get,

$P(Spam|LunchMoneyMoneyMoney)$
$= P(LunchMoneyMoneyMoney|Spam) * P(Spam)$
$= P(Lunch|Spam) * P(Money|Spam)3 * P(Spam)$
$= 0.01 * 0.583 * 0.68$
$= 0.0013$

$P(Normal|LunchMoneyMoneyMoney)$
$= P(LunchMoneyMoneyMoney|Normal) * P(Normal)$
$= P(Lunch|Normal) * P(Money|Normal)3 * P(Normal)$
$= 0.19 * 0.073 * 0.34 = 0.000022$

$P(Spam|LunchMoneyMoneyMoney) > P(Normal|LunchMoneyMoneyMoney)$

6 Laplacian smoothing

1. Naïve Bayes works well enough in text classification problems such as spam filtering and the classification of reviews as positive or negative. The algorithm seems perfect at first, but the fundamental representation of Naïve Bayes can create some problems in real-world scenarios.

$$p(y = 1 \mid x) = \frac{\prod_{j=1}^{d} p\left(x_j \mid y = 1\right) p(y = 1)}{\prod_{j=1}^{d} p\left(x_j \mid y = 1\right) p(y = 1) + \prod_{j=1}^{d} p\left(x_j \mid y = 0\right) p(y = 0)}$$
$$= \frac{0}{0}$$

2. Suppose in the real word scenarios, 1 denotes spam class, 0 non span, and an unknown word is present in the mail which is first seen in both the classes. The fraction evaluates to 0/0 form. In the naive implementation, it will give zero by zero error.It is a bad idea to estimate the probability of some event to be zero just because its not present in the data-set.
Laplace smoothing solves this by giving the last word a small non-zero probability for both classes, so that the posterior probabilities don't suddenly drop to zero.

The prior probability of feature in x is modified as given

$\phi_j = \frac{\sum_{i=1}^{n} 1\{z^{(i)}=j\}}{n}$
$\phi_j = \frac{1+\sum_{i=1}^{n} 1\{z^{(i)}=j\}}{k+n}$

And the conditional probability of a feature j in X given class 0 or 1 is modified as

$$\phi_{j|y=1} = \frac{1 + \sum_{i=1}^{n} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\right\}}{2 + \sum_{i=1}^{n} 1\left\{y^{(i)} = 1\right\}}$$
$$\phi_{j|y=0} = \frac{1 + \sum_{i=1}^{n} 1\left\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\right\}}{2 + \sum_{i=1}^{n} 1\left\{y^{(i)} = 0\right\}}$$

Now even if the feature(word) is not present in either class, the probability will not be zero, instead a small value of probability will be assigned

7 Bag of words

1. The bag-of-words model is a way of representing text data when modeling text with machine learning algorithms.The BoW model is simple to understand and implement and has seen great success in problems such as language modeling and document classification.
2. The BoW model represents text as numbers. We can represent a sentence as a BoW vector. BoW is an orderless document representation — only the counts of words matter and not their position.
3. In Bayesian spam filtering, an e-mail message is modeled as an unordered collection of words selected from one of two probability distributions: one representing spam and one representing not spam.

   Example : Let us suppose we have these sentences:

   it was the best of times,
   it was the worst of times,
   it was the age of wisdom,
   it was the age of foolishness,

   We make a list of unique words and keep count vectors for each sentence.

   Unique words: [it, was, the, best, of, times,worst,age,wisdom,foolishness]

   "it was the worst of times"
   = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

   "it was the age of wisdom"
   = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

   "it was the age of foolishness"
   = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

   1=word present, 0=word absent
4. Advantages of BoW model for Bayes classifier
   - Very simple to understand and implement

- Offers a lot of flexibility for customization on specific text data

5. Disadvantages of BoW model for Bayes classifier
   - Vocabulary: Requires careful design in order to manage the size, which impacts the sparsity of the document representations.
   - Sparsity: Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.
   - Meaning: Discarding word order ignores the context, and in turn meaning of words in the document
   - We are retaining no information on the grammar of the sentences nor on the ordering of the words in the text

8 Term Frequency-Inverse Document Frequency (TF-IDF)

1. It is a numerical statistic that is intended to reflect how important a word is to a document in a collection.
   The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

   TF is a measure of how frequently a term, t, appears in a document, d.

   $tf(t,d) = log(1 + freq(t,d))$

   IDF is a measure of how important a term is

   $idf_t = \log \frac{\text{number of documents}}{\text{number of documents with term } 't'}$

   $idf(t,D) = \log\left(\frac{N}{\text{count}(d \in D : t \in d)}\right)$

   We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

   $tf - idf(t,d,D) = tf(t,d).idf(t,D)$

9 word2-vec

Word2vec is a two-layer neural network that processes text by vectorizing words. Its takes text as an input and and outputs features vectors of words in that text. In Word2Vec, algorithms predicts a given word based on its context (CBOW), or predicting a surrounding context based on a given word (Skip-Gram). unigrams , bigrams, or even trigrams , but the storage requirement increases largely in higher dimesnion(brigrams or trigrams) as in bigrams each word has to checked in case of every other word. Sometimes, Bi-grams are not really meaningful (ex, "walked _today") and the algorithms will miss unigrams world like walked and today. training data will also be limited as there are not many instance's of two words combined. like walked today , whereas individually they occur often.

10 Naive Bayes vs logistic regression

- Naive bayes, is a very fast algorithm. It doesn't requires iterative process involving lot of epoch values, instead it calculates the closed form solution.
- Naive bayes makes assumptions on the independence of features whereas logistic regression doesn't

References

[1] http://cs229.stanford.edu/notes2020spring/cs229-notes2.pdf

[2] http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote05.html

[3] https://www.youtube.com/watch?v=02L2Uv9pdDA

[4] https://www.youtube.com/watch?v=pDHEX2usCS0

[5] https://www.geeksforgeeks.org/naive-bayes-classifiers/

[6] https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

[7] https://monkeylearn.com/blog/what-is-tf-idf/

[8] https://web.stanford.edu/~jurafsky/slp3/slides/7_NB.pdf

[9] https://machinelearningmastery.com/gentle-introduction-bag-words-model/