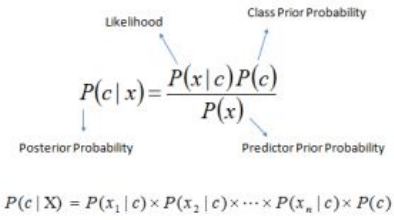
ANALYTIXLABS

Machine Learning: Bayesian Classification

Bayesian Classification

Bayesian Theorem

- Given training data D, posterior probability of a hypothesis c, P(c|x), follows the Bayes theorem



- The goal is to identify the hypothesis with the maximum posterior probability
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Bayesian Classification

We want to determine P(C|X), that is, the probability that the record $X=<x_1,\ldots,x_k>$ is of class C.

e.g. X = <rain, hot, high, light> Play Tennis?

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

Bayesian Classification

- The classification problem may be formalized using a-posteriori probabilities
- P(C|X) = probability that the record X=<x_1,…,x_k> is of class C.

  e.g. P(class=N | outlook=sunny, windy=light,…)
- Thus: assign to sample X the class label C such that P(C|X) is maximal

Estimating a-posteriori probabilities

- Bayes theorem:

$$P(C|X) = P(X|C)\cdot P(C) / P(X)$$

- **P(X) is constant for all classes**
- P(C) = relative frequency of class C samples
- C such that P(C|X) is maximum =
  C such that P(X|C)·P(C) is maximum
- Problem: computing P(X|C) = P(x_1,…,x_k|C) is infeasible!

Naive Bayesian Classification

Naïve assumption: attribute independence

$P(x_1,\ldots,x_k|C) = P(x_1|C)\cdot P(x_2|C)\cdot\ldots\cdot P(x_k|C)$

- If i attribute is categorical:
  - $P(x_i|C)$ is estimated as the relative freq of samples having value x_i as i-th attribute in class C
- If i-th attribute is continuous:
  - $P(x_i|C)$ is estimated through a Gaussian density function
- Computationally easy in both cases

Naive Bayesian Classifier

- The assumption that attributes are conditionally independent greatly reduces the computational cost.
- Given a training set, we can compute the following probabilities

Example: P(sunny|Play)

Outlook	P	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5

Play-tennis example: estimating P(C)

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis*?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

P(P) = 9/14
P(N) = 5/14

Play-tennis example: estimating P(xi|C)

Outlook	Temp.	Humidity	Wind	*Play*?
Sunny	Hot	High	Light	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Light	Yes
Rain	Mild	High	Light	Yes
Rain	Cool	Normal	Light	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Light	No
Sunny	Cool	Normal	Light	Yes
Rain	Mild	Normal	Light	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Light	Yes
Rain	Mild	High	Strong	No

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(strong|p) = 3/9	P(strong|n) = 3/5
P(light|p) = 6/9	P(light|n) = 2/5

Play-tennis example: classifying X

- **An Example**

X = <rain, hot, high, light>

P(play | X) ⇨ P(X | play) · P(play) =

=P(rain | play) · P(hot | play) ·

· P(high | play) · P(light | play) · P(p) =

= 3/9 ·2/9 · 3/9 · 6/9 · 9/14 =

= 0.010582

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(strong|p) = 3/9	P(strong|n) = 3/5
P(light|p) = 6/9	P(light|n) = 2/5

ANALYTIXLABS

Play-tennis example: classifying X

P(don't play | X) ⇨ P(X | don't play) · P(don't play) =

= P(rain | don't play) · P(hot | don't play) ·

· P(high | don't play) · P(light | don't play) ·

· P(don't play) =

= 2/5 · 2/5 · 4/5 · 2/5 · 5/14 = 0.018286 > 0.010582

Sample X is classified in class N (don't play)

ANALYTIXLABS

What are the Pros and Cons of Naive Bayes?

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

4 Applications of Naive Bayes Algorithms?

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

ANALYTIXLABS

The "zero-frequency problem"

- What if an attribute value doesn't occur with every class value? (e.g. "Outlook = Overcast" for class "No")
  - Probability will be zero! $Pr[Outlook = Overcast \mid No] = 0$
- *A posteriori* probability will also be zero! (No matter how likely the other values are!)

$$\Pr[No \mid <..., Outlook = Overcast>] = 0$$

- Remedy: add 1 to the count for every attribute value-class combination (*Laplace estimator)*
- Result: probabilities will never be zero! (also: stabilizes Probability estimates)

ANALYTIXLABS

Modified probability estimates

- In some cases adding a constant different from 1 might be more appropriate
- Example: attribute *outlook* for class *yes*

$$\frac{2+\mu/3}{9+\mu} \quad \frac{4+\mu/3}{9+\mu} \quad \frac{3+\mu/3}{9+\mu}$$

Sunny ***Overcast*** ***Rainy***

- Weights don't need to be equal (but they must sum to 1)

$$\frac{2+\mu p_1}{9+\mu} \quad \frac{4+\mu p_2}{9+\mu} \quad \frac{3+\mu p_3}{9+\mu}$$

ANALYTIXLABS

Missing values

- Training: instance is not included in frequency count for attribute value-class combination
- Classification: attribute will be omitted from calculation

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(strong|p) = 3/9	P(strong|n) = 3/5
P(light|p) = 6/9	P(light|n) = 2/5

Example:

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	Strong	?

Likelihood of "yes" = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$

Likelihood of "no" = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

P("yes") = 0.0238 / (0.0238 + 0.0343) = 41%

P("no") = 0.0343 / (0.0238 + 0.0343) = 59%

ANALYTIXLABS

Numeric attributes

- Usual assumption: attributes have a *normal* or *Gaussian* probability distribution (given the class)
- The *probability density function* for the normal distribution is defined by two parameters:
  - *Sample mean* μ

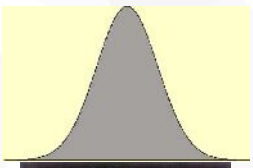
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

  - *Standard deviation* σ

$$\sigma = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu)^2$$

  - Then the density function *f(x) is*

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Karl Gauss, 1777-1855
great German mathematician

ANALYTIXLABS

Probability densities

- Relationship between probability and density:

$$\Pr[c - \frac{\varepsilon}{2} < x < c + \frac{\varepsilon}{2}] \approx \varepsilon * f(c)$$

- But: this doesn't change calculation of *a posteriori* probabilities because ε cancels out
- Exact relationship:

$$\Pr[a \le x \le b] = \int_a^b f(t)dt$$

Statistics for weather data

Outlook			Temperature		Humidity		Windy			Play	
	Yes	No	Yes	No	Yes	No		Yes	No	Yes	No
Sunny	2	3	64, 68,	65, 71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72, 80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, ...	85, ...	80, ...	95, ...					
Sunny	2/9	3/5	μ =73	μ =75	μ =79	μ =86	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	σ =6.2	σ =7.9	σ =10.2	σ =9.7	True	3/9	3/5		
Rainy	3/9	2/5									

- Example density value:

$$f(temperature = 66 \mid yes) = \frac{1}{\sqrt{2\pi}\,6.2} e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.0340$$

Classifying a new day

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Likelihood of "yes" = $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

Likelihood of "no" = $3/5 \times 0.0291 \times 0.0380 \times 3/5 \times 5/14 = 0.000136$

P("yes") = 0.000036 / (0.000036 + 0. 000136) = 20.9%

P("no") = 0.000136 / (0.000036 + 0. 000136) = 79.1%

- Missing values during training are not included in calculation of mean and standard deviation

Naïve Bayes: discussion

- Naïve Bayes works surprisingly well (even if independence assumption is clearly violated)
- Why? Because classification doesn't require accurate probability estimates *as long as maximum probability is assigned to correct class*
- However: adding too many redundant attributes will cause problems (e.g. identical attributes)
- Note also: many numeric attributes are not normally distributed

Naïve Bayes Extensions

- Improvements:
  - select best attributes (e.g. with greedy search)
  - often works as well or better with just a fraction of all attributes
- Bayesian Networks
- combine Bayesian reasoning with causal relationships between attributes
- is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG)

An Example with Python >>

ANALYTIXLABS

How to build a basic model using Naive Bayes in Python?

scikit learn (python library) will help here to build a Naive Bayes model in Python. There are three types of Naive Bayes model under scikit learn library:

Gaussian: It is used in classification and it assumes that features follow a normal distribution.

Multinomial: It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".

Bernoulli: The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

Tips to improve NB model?

Here are some tips for improving power of Naive Bayes Model:
✓If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.

✓If test data set has zero frequency issue, apply smoothing techniques "Laplace Correction" to predict the class of test data set.

✓Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.

✓Naive Bayes classifiers has limited options for parameter tuning like alpha=1 for smoothing, fit_prior=[True|False] to learn class prior probabilities or not and some other options (look at detail here). I would recommend to focus on your pre-processing of data and the feature selection.

✓You might think to apply some *classifier combination technique like* ensembling, bagging and boosting but these methods would not help. Actually, "ensembling, boosting, bagging" won't help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.

.

Contact Us

Visit us on: http://www.analytixlabs.in/

For more information, please contact us: http://www.analytixlabs.co.in/contact-us/
Or email: info@analytixlabs.co.in

Call us we would love to speak with you: (+91) 88021-73069

Join us on:
Twitter - http://twitter.com/#!/AnalytixLabs
Facebook - http://www.facebook.com/analytixlabs
LinkedIn - http://www.linkedin.com/in/analytixlabs
Blog - http://www.analytixlabs.co.in/category/blog/