innovate achieve lead

M.Tech DSE
Machine Learning
(DSECL ZG565)

Dr. Monali Mavani

BITS Pilani
Pilani Campus

Agenda

- Bayes optimal classifier (T1 book by Tom Mitchell - 6.7)
- Gibbs Algorithm (T1 book by Tom Mitchell - 6.8)
- Naïve Bayes Classifier (T1 book by Tom Mitchell - 6.9)
- Gaussian Naïve Bayes Classifier
- Text classification model (T1 book by Tom Mitchell - 6.9)

Two Principles for Estimating Parameters

- Maximum Likelihood Estimate (MLE): choose θ that maximizes probability of observed data $\mathcal{D}$

$$\widehat{\theta} = \arg\max_{\theta} \; P(\mathcal{D} \mid \theta)$$

- Maximum a Posteriori (MAP) estimate: choose θ that is most probable given prior probability and the data

$$\widehat{\theta} = \arg\max_{\theta} \; P(\theta \mid \mathcal{D})$$

$$= \arg\max_{\theta} = \frac{P(\mathcal{D} \mid \theta) P(\theta)}{P(\mathcal{D})}$$

Most Probable Classification of New Instances

- So far we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP})

- Given new instance x, what is its most probable *classification*?
  - $h_{MAP}(x)$ is not the most probable classification!

- Consider:
  - Three possible hypotheses:

    $$P(h_1|D) = .4,\ P(h_2|D) = .3,\ P(h_3|D) = .3$$

  - Given new instance x, classification given by above 3 hypotheses is

    $$h_1(x) = +,\ h_2(x) = -,\ h_3(x) = -$$

$P(\oplus|h_1) = 1$ and $P(\ominus|h_1) = 0$

  - What's most probable classification of x?

Bayes' Optimal Classifier

- The most probable classification of the new instance (or the label produced by the most probable classifier) is obtained by combining **the predictions of all hypotheses, weighted by their posterior probabilities.**
- v_j from some set V, then the probability $P(v_j \mid D)$ that the correct classification for the new instance is v_j is:

$$P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i) P(h_i|D)$$

- The optimal classification of the new instance is the value vj for which $P(v_j \mid D)$ is maximum

BITS Pilani, Pilani Campus

Bayes Optimal Classifier

- **Bayes optimal classification:**

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- Example:

$$P(h_1|D) = .4,\ P(-|h_1) = 0,\ \ P(+|h_1) = 1$$

$$P(h_2|D) = .3,\ P(-|h_2) = 1,\ \ P(+|h_2) = 0$$

$$P(h_3|D) = .3,\ P(-|h_3) = 1,\ \ P(+|h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$
$$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

and

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = -$$

Gibbs Classifier

- Bayes optimal classifier provides best result, but can be expensive if many hypotheses.
- Gibbs algorithm:
  1. Choose one hypothesis at random, according to posterior prob. Distribution over h , $P(h|D)$
  2. Use this h to classify new instance

Surprising fact: under certain conditions, the expected misclassification error for the Gibbs algorithm is at most twice the expected error of the Bayes optimal classifier

$$E[error_{Gibbs}] \leq 2E\,[error_{BayesOptional}]$$

- Suppose correct, uniform prior distribution over *H*, then
  - Pick any hypothesis from *Version space*, with uniform probability
  - Its expected error no worse than twice Bayes optimal

Naïve Bayes

Given a data point x, what is the probability of x belonging to some class c?

Recall….

Independent Events A and B

Joint Probability distribution of Independent RVs - X and Y

$$P(X,Y) = P(X)\,P(Y)$$

Conditional Probability distribution of Independent RVs - X and Y

$$P(X|Y) = P(X)$$

Conditional independence

- **Definition**: X is conditionally independent of Y given Z, if the probability distribution governing X is independent of the value of Y, given the value of Z

$$(\forall i, j, k)\ P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z_k)$$

$$P(X|Y, Z) = P(X|Z)$$

Example:

$$P(\text{Thunder}|\text{Rain}, \text{Lightning}) = P(\text{Thunder}|\text{Lightning})$$

Slide credit: Tom Mitchell

Applying conditional independence

- Naïve Bayes assumes X_i are conditionally independent given Y
  e.g., $P(X_1|X_2,Y) = P(X_1|Y)$

- $P(X_1,X_2|Y) = P(X_1|X_2,Y)P(X_2|Y)$ [By general property of probabilities]
  $= P(X_1|Y)P(X_2|Y)$

- **General form: $\boldsymbol{P(X_1,\cdots,X_n|Y) = \prod_{j=1}^{n} P(X_j|Y)}$**

Naïve Bayes Independence assumption

- Assumption:

$$P(X_1, \cdots, X_n | Y) = \prod_{j=1}^{n} P(X_j | Y)$$

- i.e., X_i and X_j are <u>conditionally independent</u> given Y for $i \neq j$

Naïve Bayes classifier

Goal of learning P(Y|X) where X = <X_1,…, X_n>

- Bayes rule:

$$P(Y = y_k | X_1, \cdots, X_n) = \frac{P(Y = y_k) P(X_1, \cdots, X_n | Y = y_k)}{\sum_j P(Y = y_j) P(X_1, \cdots, X_n | Y = y_j)}$$

- Assume conditional independence among X_i's:

$$P(Y = y_k | X_1, \cdots, X_n) = \frac{P(Y = y_k) \Pi_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \Pi_i P(X_i | Y = y_j)}$$

- Pick the most probable (MAP) Y for $X_{new} = < X_1, \ldots, X_n >$

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}}\, \underbrace{P(Y = y_k)}_{\text{Prior Probability}} \Pi_i \underbrace{P(X_i | Y = y_k)}_{\text{MLE}}$$

Slide credit: Tom Mitchell

Naive Bayes Algorithm – Discrete valued inputs X_i

- For each target value Y_k *(MLE estimate)*

$P(Y = y_k) \leftarrow$ No. of instances with Y_k *class/No. of Total instances*

- For each attribute value a_i

$P\ (X_i|Y = y_k) \leftarrow$ *No. of instances with* X_i *within* Y_k *class / No. of instances with* Y_k *class*

- Classify New Instance(*x*)

Pick the most probable (MAP) Y

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}}\, P(Y = y_k)\Pi_i P(X_i|Y = y_k)$$

Naive Bayes: Example

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

TABLE 3.2
Training examples for the target concept *PlayTennis*.

BITS Pilani

Naive Bayes: Example

- Consider *PlayTennis*, and new instance

  <*Outlk = sun, Temp = cool, Humid = high, Wind = strong*>
- Want to compute:

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}} P(Y = y_k)\Pi_i P(X_i|Y = y_k)$$

$$P(y)\, P(sun|y)\, P(cool|y)\, P(high|y)\, P(strong|y) = .005$$

$$P(n)\, P(sun|n)\, P(cool|n)\, P(high|n)\, P(strong|n) = .021$$

BITS Pilani

Naive Bayes Algorithm – Continuous valued inputs X_i

Gaussian Naïve Bayes

- Assume that for each possible discrete value y_k of Y, the distribution of each continuous X_i is Gaussian, and is defined by a mean and standard deviation specific to X_i and y_k

$$\boxed{p(X_i = x | Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} \, e^{-\frac{1}{2}\left(\frac{x-\mu_{ik}}{\sigma_{ik}}\right)^2}}$$

$$\boxed{\begin{aligned} \mu_{ik} &= E[X_i | Y = y_k] \\ \sigma_{ik}^2 &= E[(X_i - \mu_{ik})^2 | Y = y_k] \end{aligned}}$$

mean and standard deviation of each of these Gaussians

Maximum likelihood estimates:

jth training example

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k)$$

ith feature

kth class

δ()=1 if (Y^j=y_k) else 0

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$

Naive Bayes Algorithm – Continuous valued inputs X_i

- For each target value Y_k (MLE estimate)

$P(Y = y_k) \leftarrow$ No. of instances with Y_k *class/No. of Total instances*

- For each attribute value X_i estimate $P\,(X_i|Y = y_k)$
  - **class conditional mean , variance**

- Classify New Instance(*x*)

Pick the most probable (MAP) Y

$$\hat{Y} \leftarrow \underset{y_k}{\operatorname{argmax}}\, P(Y = y_k)\Pi_i P(X_i|Y = y_k)$$

Example - 2

		Humidity	*Mean*	*StDev*
Play Golf	yes	86 96 80 65 70 80 70 90 75	79.1	10.2
	no	85 90 70 95 91	86.2	9.7

Laplace Smoothing

Issues with Naïve Bayes Classifier

Consider the table with Tid = 7 deleted

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naïve Bayes Classifier:

P(Refund = Yes | No) = 2/6
P(Refund = No | No) = 4/6
P(Refund = Yes | Yes) = 0
P(Refund = No | Yes) = 1
P(Marital Status = Single | No) = 2/6
P(Marital Status = Divorced | No) = 0
P(Marital Status = Married | No) = 4/6
P(Marital Status = Single | Yes) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
P(Marital Status = Married | Yes) = 0/3
For Taxable Income:
If class = No: sample mean = 91
sample variance = 685
If class = No: sample mean = 90
sample variance = 25

Given X = (Refund = Yes, Divorced, 120K)

P(X | No) = 2/6 X 0 X 0.0083 = 0

P(X | Yes) = 0 X 1/3 X 1.2 X 10^{-9} = 0

Naïve Bayes will not be able to classify X as Yes or No!

Smoothing

If one of the conditional probabilities is zero, then the entire expression becomes zero

- Technique for smoothing categorical data.
- A small-sample correction, or **pseudo-count**, will be incorporated in every probability estimate.
- No probability will be zero.

Smoothing

Probability estimation:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

N_c: number of instances in the class

N_{ic}: number of instances having attribute value A_i in class c

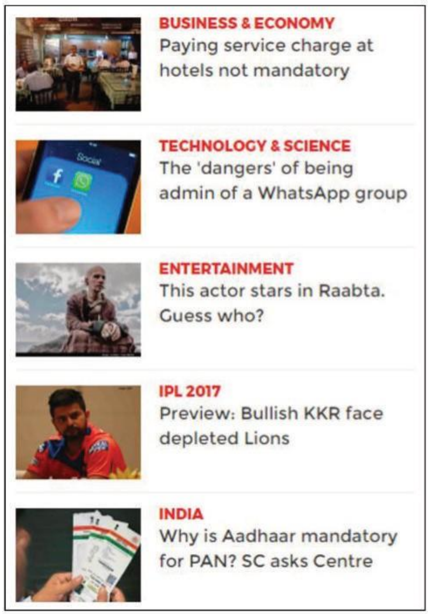
p: prior probability of the class

m: constant called the **equivalent sample size**, which determines how heavily to weight p relative to the observed data

Applications

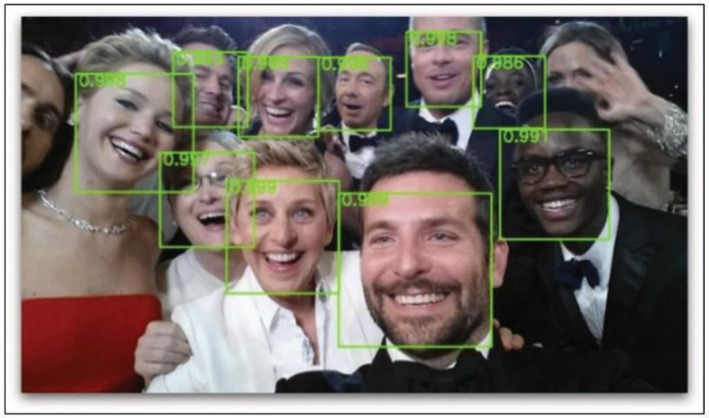
Naïve Bayes Classifier Applications

Categorizing News

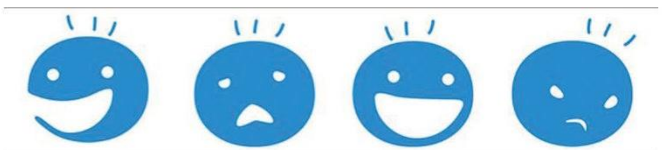


Email Spam Detection

Face Recognition



Sentiment Analysis



Naive Bayes Classifier

- Along with decision trees, neural networks, one of the most practical learning methods.
- When to use
  - category labels
  - category features (easier to convert continuous features into categorical features)
  - Many features (e.g text)
  - Features are conditionally independent given labels
  - Need to have simple code
- Successful applications:
  - Diagnosis
  - Classifying text documents

Text Classification using Naive Bayes Classifier

Example 1

Which Tag sentence " A very close game" belong to?

Text	Tag
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

Apply Naïve Bayes

Laplace Smoothing

- Laplace smoothing: we add 1 or in general constant k to every count so it's never zero.
- To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1
- In our case, the possible words are ['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match'].

- In our example
- we add 1 to every probability, therefore the probability, such as **P(close | sports**), will never be 0.

Apply Laplace Smoothing

Word	P(word | Sports)	P(word | Not Sports)
a	2+1 / 11+14	1+1 / 9+14
very	1+1 / 11+14	0+1 / 9+14
close	0+1 / 11+14	1+1 / 9+14
game	2+1 / 11+14	0+1 / 9+14

$$P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports)$$
$$= 2.76 \times 10^{-5}$$
$$= 0.0000276$$

$$P(a|Not\,Sports) \times P(very|Not\,Sports) \times P(close|Not\,Sports) \times P(game|Not\,Sports) \times P(Not\,Sports)$$
$$= 0.572 \times 10^{-5}$$
$$= 0.00000572$$

Learning to classify document: P(Y|X) the "Bag of Words" model

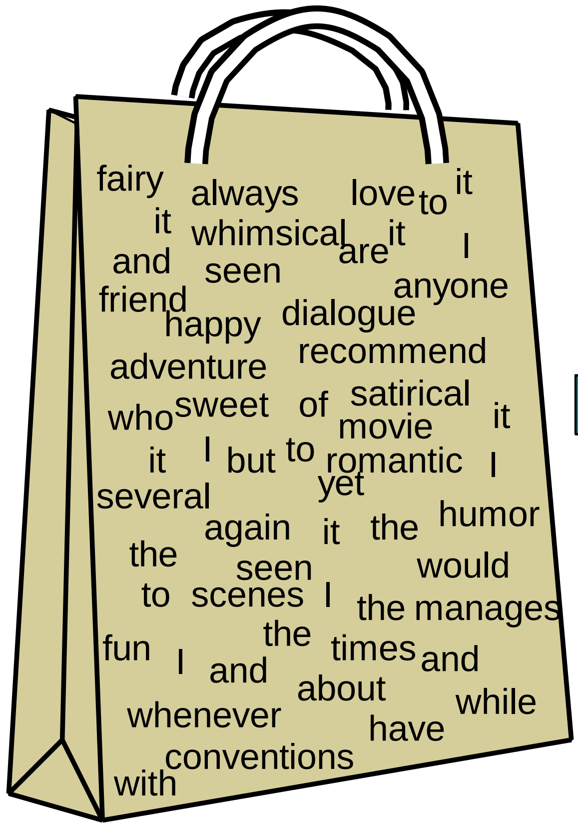
- Y discrete valued. e.g., Spam or not
- $X = \langle X_1, X_2, \ldots X_n \rangle$ = document

- X_i is a random variable describing the word at position i in the document
- possible values for X_i : any word w_k in English

- Document = bag of words: the vector of counts for all w_k's

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

The Bag of Words Representation

- **Bag of Words assumption**: Assume position doesn't matter

A piece of text like "When the lecture is over, remember to take your bag" would look to this algorithm the same as if we just sorted the words alphabetically *"bag is lecture over remember take the to When your"*

Learning to Classify Text

- Why?
  - Learn which news articles are of interest
  - Learn to classify web pages by topic
- **Computer could learn the target concept accurately, it could automatically filter the large volume of online text documents to present only the most relevant documents to the user.**
- Two main design issues:
  - how to represent an arbitrary text document in terms of attribute values
  - to decide how to estimate the probabilities required by the naive Bayes classifier.

Learning to Classify Text

E.g document : “Our approach to representing arbitrary text documents is disturbingly simple: Given a text document, such as this paragraph, we define **an** attribute for each word position in the document and define the value of that attribute to be the English word found in that position. Thus, the current paragraph would be described by 111 attribute values, corresponding to the 111 word positions. **The value of the first attribute is the word "our," the value of the second attribute is the word "approach**," and so on. Notice that long text documents will require a larger number of attributes than short documents. As we shall see, this will not cause us any trouble.”

- Total Attributes(features) = 111 corresponding to the 111 word positions.
- Two classes v_j => {like, dislike}
- **700 training documents - dislike**
- **300 training documents - like**
- **Thus, class prior probabilities are P(like) =0.3 and P(dislike) = 0.7**

Learning to Classify Text - training

Apply naive Bayes classification and independence assumption

$$P(a_1, \ldots a_{111}|v_j) = \prod_1^{111} P(a_i|v_j)$$

$P(a_i = w_k \mid v_j)$ is probability that word in position i is w_k, given v_j

$$v_{NB} = \operatorname*{argmax}_{v_j \in \{like, dislike\}} P(v_j) \prod_{i=1}^{111} P(a_i|v_j)$$

$$= \operatorname*{argmax}_{v_j \in \{like, dislike\}} P(v_j)\ P(a_1 = \text{“}our\text{”}|v_j) P(a_2 = \text{“}approach\text{”}|v_j)$$

$$\ldots\ P(a_{111} = \text{“}trouble\text{”}|v_j)$$

$P(a_1 = w_k \mid v_j)$, $P(a_2 = w_k \mid v_j)$ . . . by the single position-independent probability $P(w_k \mid v_j)$ e.g P(“our” | like) and P(“our” | dislike)

Learning to Classify Text - training

$$P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$$

- ***m*** equal to the size of the word vocabulary
- Uniform prioir prob. P=1/|vocabulary|

n = total number of word positions in all training examples whose target value is v_j = > (N_c - no of words in like class)

n_k = number of times word w_k is found among these n word positions = > (N_{ic} - no of word "our" in like class)

|Vocabulary| = total number of distinct words (and other tokens) found within the training data.

BITS Pilani

Learning to Classify Text - testing

CLASSIFY_NAIVE_BAYES_TEXT (*Doc*)

- *positions* ← all word positions in test *Doc* that contain tokens found in *Vocabulary*
- *(e.g document contains 111 words)*
- Return v_{NB} where

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

Features of Bayesian learning

- Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.

- Flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.

Practical Issues of Bayesian learning

- Require initial knowledge of many probabilities
  - Often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
- Significant computational cost required to determine the Bayes optimal hypothesis in the general case (linear in the number of candidate hypotheses)

Some references for more examples

- Movie Review: https://www.youtube.com/watch?time_continue=16&v=EGKeC2S44Rs

- Spam mails by Prof. Andrew Ng:

https://www.youtube.com/watch?v=z5UQyCESW64

https://www.youtube.com/watch?v=NFd0ZQk5bR4

- NLP by Prof. Dan Jurafsky, Stanford: https://www.youtube.com/watch?v=Fmu65a0v6Sw

Self Reading

Example 3

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Bayes theorem and Concept Learning

- What is the relationship between Bayes theorem and Concept Learning.
- It can be used for designing a straight forward learning algorithm.
- Brute-Force MAP Learning algorithm

  1. For each hypothesis h ∈ H, calculate the posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

  2. Output hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \underset{h \in H}{argmax}\ P(h|D)$$

Brute-Force Bayes Concept Learning

- **H:-** a finite hypothesis space defined over the instance space ***X***
- **c : X → {0,1} :-** target concept
- some sequence of training examples $\{\langle x_1, d_1\rangle, \ldots, \langle x_m, d_m\rangle\}$
- $d_i = c(x_i)$.
- The sequence of instances $\{x_1, x_2, \ldots, x_m\}$ is held fixed, so that ***D*** can be written simply as the sequence of target values $\{d_1, d_2, \ldots, d_m\}$.

Brute-Force MAP Learning algorithm

- in order to specify a learning problem for the algorithm, values for P(h) and P(D|h) must be specified.
- Assumptions
  - ✓ Training data, D is noise free (i.e. $d_i = c(x_i)$)
  - ✓ Target concept c is contained in H i.e $(\exists h \in H)[(\forall x \in X)[h(x) = c(x)]]$
  - ✓ No reason to believe that any hypothesis is more probable than any other

$$\Rightarrow P(h) = \frac{1}{|H|} \text{ for all } h \in H$$

$$\Rightarrow P(D|h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \in D \\ 0 & \text{otherwise} \end{cases}$$

Brute-Force MAP Learning algorithm

- h is inconsistent with the training data D

$$P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0$$

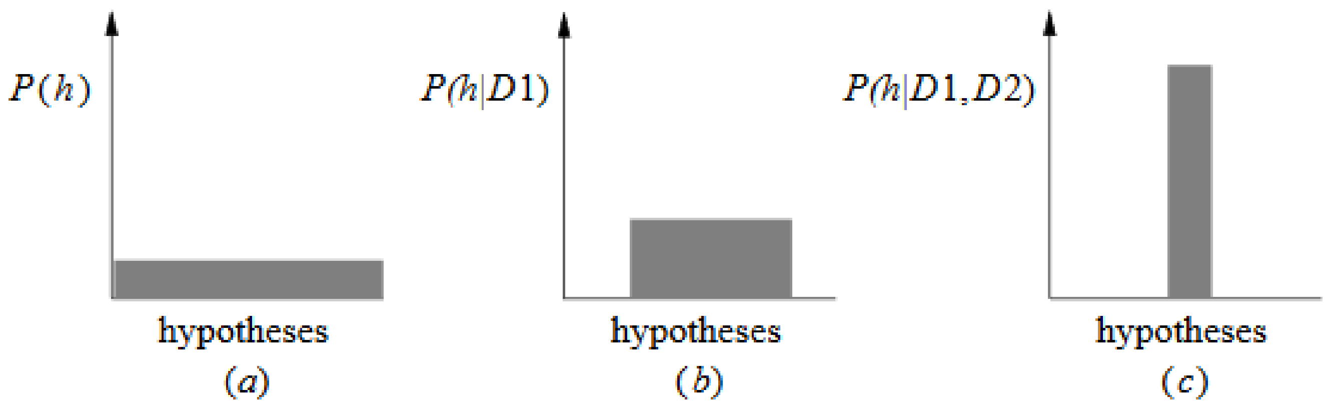
- H is consistent with training data D

$$P(h|D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)} = \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$$

$$P(h|D) = \frac{1}{|VS_{H,D}|}$$

- The above analysis implies that under our choice for *P(h)* and *P(Dlh)*, every *consistent* hypothesis has posterior probability $(1/|VS_{H,D}|)$.
- every inconsistent hypothesis has posterior probability **0.**
- **Every consistent hypothesis is, therefore, a MAP hypothesis.**

MAP Hypotheses and consistent learnrers



Evolution of probabilities:

(a)all hypotheses have the same probability

(b)+ (c) as training data accumulates, the posterior probability of inconsistent hypotheses becomes zero while the total probability summing to 1 is shared equally among the remaining consistent hypotheses

BITS Pilani, Pilani Campus

Naïve Bayes Generative Model

- Naïve Bayes classifier uses likelihood and prior probability to calculate conditional probability of the class
- Likelihood is based on joint probability, which is the core principle of probabilistic generative model
- Naïve Bayes simplifies the calculation of likelihood by the assumption of conditional independence among input parameters
- Each parameter's likelihood is determined using joint probability of the input parameter and the output label

BITS Pilani

Naïve Bayes – Advantages

- Algorithm is simple to implement and fast
- If conditional independence holds, it will converge quickly than other methods
- Even in cases where conditional independence doesn't hold, its results are quite acceptable
- Needs less training data (due to conditional independence assumption)
- Highly scalable, scales linearly with the number of predictors and data points
- Can be used for both binary and multi-class classification problems
- Handles continuous and discrete data
- Not sensitive to irrelevant features
- Doesn't overfit the data due to small model size (compared to other algorithms like Random Forest)
- Handles missing values well

BITS Pilani

Thank You

BITS Pilani