PES UNIVERSITY ONLINE

Machine Intelligence

Preet Kanwal

Associate Professor

Department of Computer Science & Engineering

Acknowledgement : Dr. Rajanikanth K (Former Principal MSRIT, PhD(IISc), Board of Studies Member, PESU)

Machine Intelligence

Unit 3 -- Bayesian Learning

Preet Kanwal

Department of Computer Science & Engineering

Machine Intelligence

Unit 3 - Outline (Bayesian Learning)

Class 1 :

- Bayesian Learning -- Idea
- Basics of Probability and Bayes theorem

Class 2 :

- Maximum A posteriori Hypothesis, Maximum Likelihood Hypothesis
- Applying Bayes theorem to Concept Learning

  --- Brute force Bayes Concept learning

Class 3 :

- Naive Bayes Classifier

Class 4 :

- Bayes Optimal Classifier and Gibbs Algorithm
- Maximum likelihood hypothesis and Least squared error

Bayesian reasoning provides a probabilistic approach to inference.

(Learning is viewed as a generalization/inference problem from usually small sets of high dimensional, noisy data.)

It is based on the assumption that the quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data.

Bayesian reasoning provides the basis for learning algorithms that directly manipulate probabilities : Naive Bayes, HMM

as well as it provides a framework for analyzing the operation of other algorithms that do not explicitly manipulate probabilities. We can have a Bayesian perspective to these algorithms which will kind of justify the working/approach of these algorithms (Find-S, Candidate Elimination, Neural networks, decision trees).

Example : In the neural network, the weights can be initialized from a probability distribution. The output vector from a softmax (activation function) in neural network is also probability

MACHINE INTELLIGENCE

Probability Basics & Bayes theorem

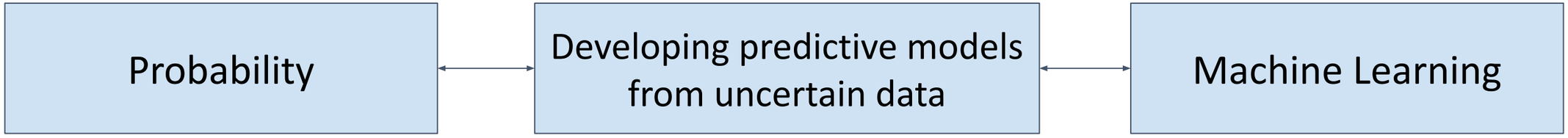
Preet Kanwal

Department of Computer Science and Engineering

Machine Intelligence

Importance of Probability

Probability is the Bedrock of Machine Learning



Why Uncertainty?

- Imperfect or Incomplete information - we are trying to learn based on certain instances.
- Noisy data, erroneous data -- no perfect learning

Basics - Event

Let us revise our concepts before moving ahead.

Event : A set of outcomes from a random experiment.

Example: tossing a coin, we can get head or a tail -- event

Event 1 = set consisting of heads or tails.
All possible subsets of set would also be called as events.

Event 2 = set of only heads
Event 3 = set of only tails
Event 4 = empty set
So there will be 2^n events (Power set)

Each event would have a probability.
P(empty set) = 0, P(Heads) = ½, P(Tails) = ½, P(Heads or tails) = 1

Basics - Random Variable

A random variable assigns a numerical value to each outcome in a sample space.
(real -- discrete random variable / integer-- continuous random variable)

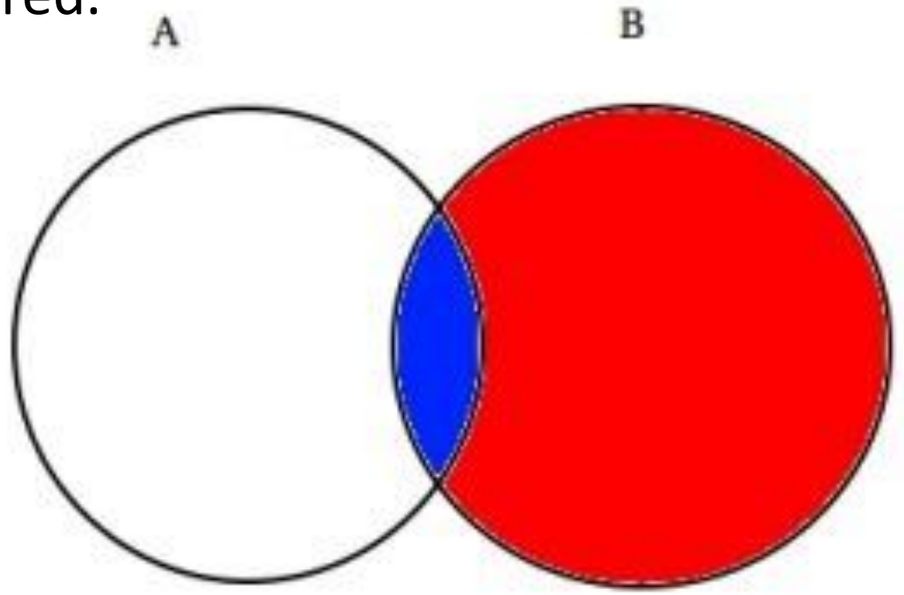
It is customary to denote random variables with uppercase letters.
The letters X, Y , and Z are most often used.

Machine Intelligence

Basics - Conditional Probability

You are given that an event has occurred, given that information what can you say about probability of other events occurring.

Conditional probability of an event A is the probability that the event will occur given the knowledge that an event B has already occurred.



P(A|B) - Probability of A given B

Now since B has happened, the part which now matters for A is the part shaded in blue which is interestingly:
A ∩ B

So, the probability of A given B turns out to be:

P(A|B) = $\frac{BlueArea}{RedArea + BlueArea}$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Machine Intelligence

Basics - Conditional Probability Example

When we throw a die, we know the probability of getting any one face (1, 2, 3, 4, 5, 6) is ⅙.

If you throw the die only once, then what is the probability of getting a 2?
Answer : ⅙

Now, if you are given the information that the roll of a die resulted in an even face. Then what is the probability of getting a 2?
P(getting a 2|even face) = ⅓

When you say an event has occurred, it reduces the sample space to only those events which make that B possible (as it is P(A|B).

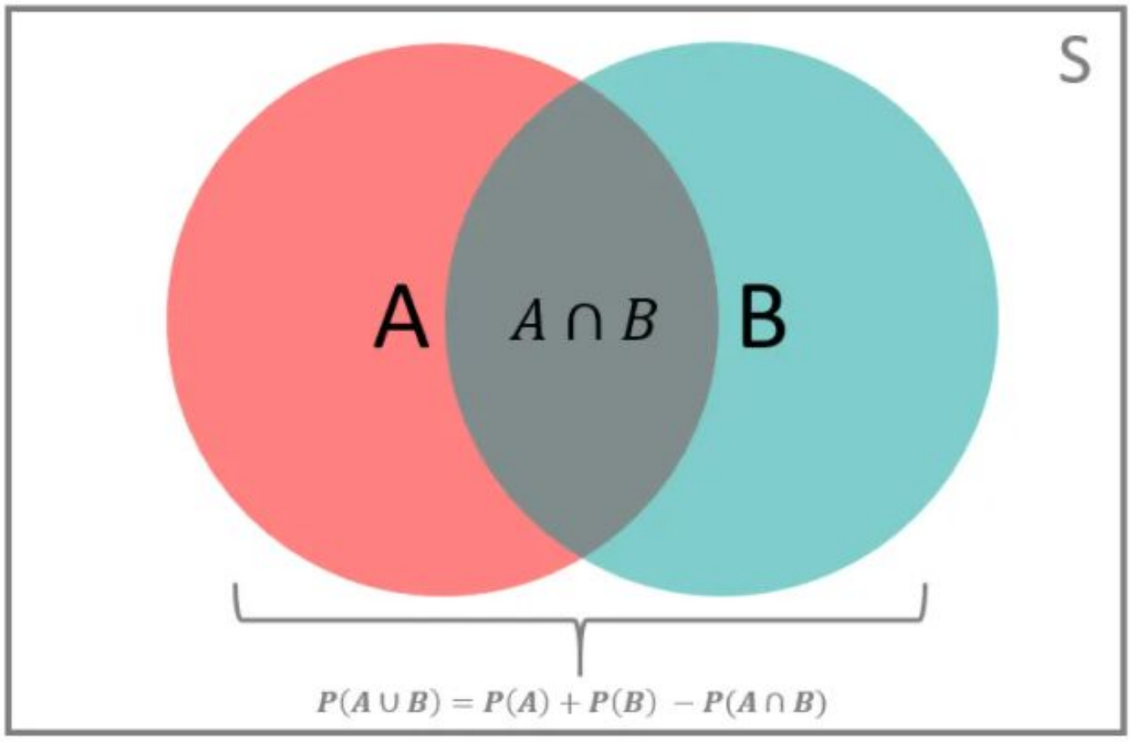
Within the reduced sample space, we must then calculate the probability of getting an A.

Machine Intelligence

Basics - Addition Theorem

If A and B are two events in a probability experiment, then the probability that either one of the events will occur is:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$



to get the correct count for (A U B) we subtract

$P(A \cap B)$

from P(A) + P(B) as it occurred twice (first when we counted A, second when we counted B).

Machine Intelligence

Basics - Addition Theorem Example

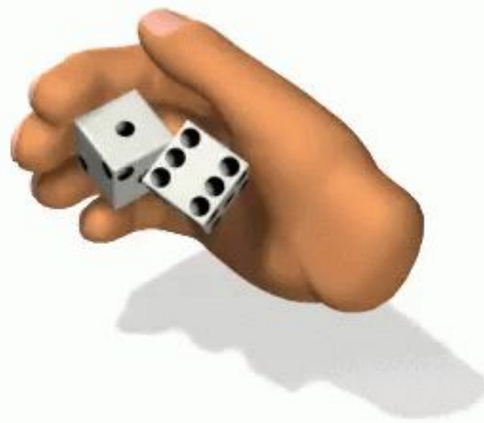
Let us say for example, Two dice are rolled together. Find the probability of getting a doublet (A) or sum of faces as 6 (B).

Sample space of getting a doublet (A)= 6
A={ (1,1), (2,2), (3,3), (4,4), (5,5), (6,6)}

Sample space of getting sum of faces as 6 (B)= 5
B= { (1,5),(5,1),(2,4),(4,2), (3,3)}

Sample space of getting a doublet (A) and sum of faces as 6 (B)
$A \cap B = \{(3,3)\}$ ---- Its a set intersection. Could be an empty set.

$\Rightarrow P(A \cup B) = 6/36 + 5/36 - 1/36$

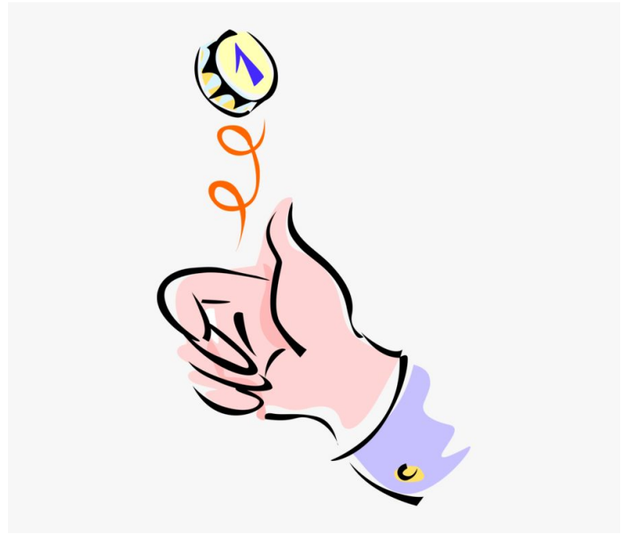


Machine Intelligence

Basics - Independent Events

Let A and B be two events associated with a sample space S.

If the probability of occurrence of one of them is not affected by the occurrence of the other, then we say that the two events are independent.



Example: If you throw two coins, then probability of getting head or tail in second coin is independent of probability of getting head or tail in first coin.

$$P(B \mid A) = P(B), \text{ provided } P(A) \neq 0$$
$$P(A \mid B) = P(A), \text{ provided } P(B) \neq 0$$

Machine Intelligence

Basics - Multiplication theorem

Multiplication rule of probability helps one calculate the probability of occurrence of both the events A and B.

If A and B are two **independent** events in an experiment, then the probability of both events occurring simultaneously is given by:

$P(A \cap B) = P(A).\ P(B)$

If A and B are **dependent** events, then the probability of both events occurring simultaneously is given by:

$P(A \cap B) = P(B)\ .\ P(A|B)$

$P(A \cap B) = P(A)\ .\ P(B|A)$

Machine Intelligence

Basics - Multiplication theorem

The theorem can he extended easily as:

For independent events
$P(A \cap B \cap C)=P(A).P(B).P(C)$

For dependent events

$$P(A \cap B \cap C) = P(A) \cdot P(B|A) \cdot P(C|(A \cap B))$$
$$= P(A) \cdot P(B|A) \cdot P(C|AB)$$

The rule can be extended to any number of events.

$$P(A_1 \cap A_2 \cap A_{3\ldots\ldots} \cap A_n) = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_1 \cap A_2) \cdot P(A_4|A_1 \cap A_2 \cap A_3)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots P(A_n | A_1 \cap A_2 \cap A_{3\ldots\ldots} \cap A_{n-1})$$

Machine Intelligence

Basics - Independent Events and Joint Probability

Using the multiplication theorem on probability, we have

Joint Probability= $P(A \cap B) = P(A) \cdot P(B)$

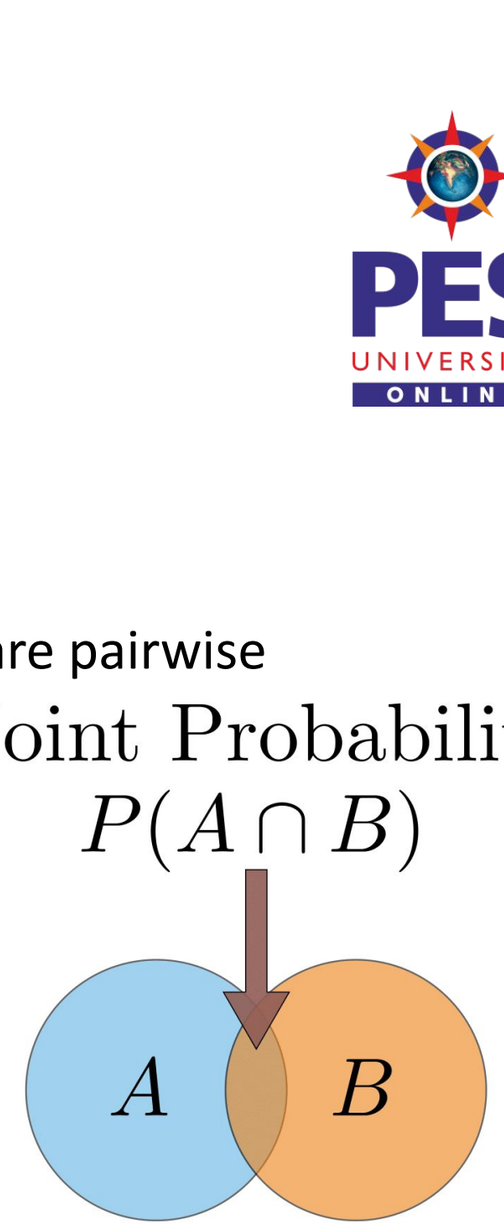
Multiple events such as A, B and C are said to be mutually independent if they are pairwise independent :

$P(A \cap B) = P(A)\ P(B)$

$P(A \cap C) = P(A)\ P(C)$

$P(B \cap C) = P(B)\ P(C)$

and $P(A \cap B \cap C) = P(A)\ P(B)\ P(C)$



if the events are not independent, we will have a difficulties calculating joint probabilities (since we will have to take conditional probabilities into account) --- This assumption is made for simplicity so that the problem becomes tractable, computations become manageable if there are large number of events.

Machine Intelligence

Mutually Exclusive Events, Collectively Exhaustive Events

Mutually exclusive events :

If one event has occurred, that automatically means the other event has not occurred. That means the probability of their jointly occurring is zero. P (A ⋂ B) = 0

For example, in throwing a die, if an even number shows up, it automatically excludes the possibility that odd number has shown up.

Set intersection of mutually exclusive events is an empty set.

Collectively Exhaustive events: The events are called collectively exhaustive events when their union is the original sample space. P(A U B) = P(A) + P(B) = 1

if events are mutually exclusive and collectively exhaustive, that basically means the sample space has been partitioned.

Machine Intelligence

Independent, identically distributed random variable (IID)

For mathematical tractability, we use this concept.

A collection of random variables is independent and identically distributed if each random variable has the same probability distribution as the others and all are mutually independent.

Example : Outcomes of flipping of a coin are IID.
Independent - one outcome does not depend on the other outcome
Identical - because every sample comes from the same distribution (there is no change in the distribution when we flip a coin).

Most classifier-learning algorithms assume that the training data is independently and identically distributed. For example, support vector machine (SVM), back-propagation for Neural Networks, and many other common algorithms implicitly make this assumption as part of their derivation. Nevertheless, this assumption is commonly violated in many real life problems where sub-groups of samples exhibit a high degree of correlation amongst both features and labels.

Machine Intelligence

Total Probability Theorem

Sometimes an event can occur via different paths. Through several distinct events, it expresses the total probability of an event. To find the probability of such an event we need to add the probabilities of all the paths. This is leads to the theorem of total probability.

Let A1,...,An be mutually exclusive and exhaustive events,
i.e. P (A1 ∩ A2 ∩ A3……. ∩ An) = 0
P (A1 U A2 U A3……. U An) = 1
where ∀ i P(Ai)>0. Let B be any event. Then

$$P(B) = \sum_{1}^{n} P(A_i)P(B|A_i).$$

Machine Intelligence

Total Probability Theorem

$$P(B) = \sum_{1}^{n} P(A_i)P(B|A_i).$$

$$= \sum_{i=1}^{n} P(A_i) \quad \frac{P(B \cap A_i)}{P(A_i)}$$

$$= \sum_{i=1}^{n} P(B \cap A_i)$$

$$= P(B \cap A_1) + P(B \cap A_2) + P(B \cap A_3) \ldots + P(B \cap A_n)$$

$$= P(B \cap A_1 \cup A_2 + A_3 + \ldots A_n)$$

$$= P(B \cap S)$$

$$= P(B)$$

Machine Intelligence

Total Probability Theorem

You received an assignment to be completed on MI. The probabilities of completion of the assignment on time with and without internet are 0.90 and 0.42 respectively. If the probability that internet is available is 0.45, then determine the probability that the assignment will be completed on time.

Let B be the event that the assignment will be completed on time and A be the event that the internet is available.

$P(A) = 0.45$

$P(!A) = 0.55$

$P(B|A) = 0.90$

$P(B|\ !A) = 0.45$

$P(B) = P(A) * P(B|A) + P(!A) * P(B|\ !A)$

$=0.45 \times 0.9 + 0.55 \times 0.45 = 0.6525$

Machine Intelligence

Total Probability Theorem

In a certain county

- 60% of registered voters are Republicans
- 30% are Democrats
- 10% are Independents.

- When those voters were asked about increasing military spending
  - 40% of Republicans opposed it
  - 65% of the Democrats opposed it
  - 55% of the Independents opposed it.
- What is the probability that a randomly selected voter in this county opposes increased military spending?

Let B be the event that a voter opposes the increase in military spending.

P(opposes) = P(republican) . P(opposes | republican)
+ P(democrat) . P(opposes | democrat)
+ P(independent) . P(opposes | independent)
= 0.6 * 0.4 + 0.30 * 0.65 + 0.1 * 0.55

Machine Intelligence

Bayes Theorem

If A and B are two random variables

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

In the context of classifiers, hypothesis **h** and training data **D** are related as

- $P(h)$–prior probability of hypothesis h
- $P(D)$–prior probability of training data D
- $P(h|D)$ – posterior probability of h given D
- $P(D|h)$ – Likelihood: probability of D given h

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

Machine Intelligence

Bayes Theorem

Bayes' Theorem: Time for some formal definition

If E1, E2,..., En are mutually exclusive and exhaustive events associated with a sample space, and A is any event of non zero probability, then

$$P(E_i|A) = \frac{P(E_i)P(A|E_i)}{\sum P(E_i)P(A|E_i)}$$

Machine Intelligence

Bayes Theorem

Bayes Theorem says, what is the probability of a hypothesis given the data or evidence i.e what is P(h|D) --- Posterior probability?

P(h|D) can be calculated as P(D|H) (which is the probability of getting that particular kind of data evidence given that the hypothesis holds) multiplied by prior probability P(h) and divided by the prior probability P(D).

P(h) - before looking at any evidence/data what is the probability that the hypothesis is true.

P(D) - the probability of that kind of Data appearing without referring to any hypothesis.

Bayes theorem is the foundation of Bayesian learning.

Question --- How do we use Bayes theorem for classification??

Machine Intelligence

Bayes Theorem

Question --- How do we use Bayes theorem for classification??

a) Using Bayes theorem, one could compute the posterior probability of different hypothesis.

One approach would be to choose the hypothesis which has the maximum value for posterior probability, and return the verdict given by that hypothesis.

It may or may not be best in reality. But, from probability perspective it seems to be the most logical.

So we can say best hypothesis is the one which is the most probable hypothesis.

b) Or one could find out the most probable classification given the data.

MACHINE INTELLIGENCE

MAP and ML hypothesis

Preet Kanwal

Department of Computer Science and Engineering

Machine Intelligence

Maximum A Posteriori (MAP) Hypothesis

- The learner considers some set of candidate hypotheses H and it is interested in finding the most probable hypothesis h H given the observed data D
- Any such maximally probable hypothesis is called a maximum a posteriori (MAP) hypothesis h_{MAP}.
- Maximum a Posteriori or MAP for short is a Bayesian-based approach to estimating a distribution and model parameters that best explain an observed dataset.
- MAP involves calculating a conditional probability of observing the data given a model weighted by a prior probability or belief about the model.

Machine Intelligence

Maximum A Posteriori (MAP) Hypothesis

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

- By using Bayes Theorem we can find the most probable hypothesis that is called the MAP Hypothesis
  (A posteriori, Latin for "from the latter", is a term from logic, which usually refers to reasoning that works backward from an effect to its causes)

- Some terms:
  - h: a specific Hypothesis
  - H: Hypothesis space - set of hypothesis
- We need to maximise the probability of hypothesis given the data, let's write that mathematically

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

Tom Mitchell- Textbook

Machine Intelligence

Maximum A Posteriori (MAP) Hypothesis

Goal of learning: Find maximum a posteriori hypothesis h_{MAP}:

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

$$= \underset{h \in H}{\operatorname{argmax}} \frac{P(D \mid h)P(h)}{P(D)}$$

$$= \underset{h \in H}{\operatorname{argmax}} P(D \mid h)P(h)$$

Bayes theorem:

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

This is the optimal hypothesis in the sense that no other hypothesis is more likely.

* note that $P(D)$ can be dropped, because it is a constant independent of h

Machine Intelligence

Maximum Likelihood Hypothesis

In some cases, if every hypothesis in H is equally probable a priori i.e., ***$P(h_i) = P(h_j)$*** *for all h_i and h_j in H*

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

h_{ML} is called the “maximum likelihood hypothesis”.

P(D|h) – the likelihood of the data D given h

Machine Intelligence

Summary - MAP and ML hypothesis

1. MAP Hypothesis

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

2. ML Hypothesis

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

Bayesian Recipe for learning

When is $h_{MAP} = h_{ML}$?

If $P(h_i) = P(h_j)$ $\forall$ i,j then, $h_{MAP} = h_{ML}$

Does patient have cancer or not?

A patient takes a test for cancer. The test has two outcomes: **positive** and **negative**.

It is known that if the patient has cancer, the test is positive **98%** of the time. If the patient does not have cancer, the test is negative **97%** of the time. It is also known that **0.008** of the population has cancer.

The patient's test is positive.

Which is more likely: Should we diagnose a patient whose lab result is positive as having cancer?

Machine Intelligence

Example to see how Bayes Idea work

- **Hypothesis space:** **Outcome**

  h_1 = Patient has cancer D ={+,-}

  h_2 = Patient does not have cancer

- **Prior Probability:** **0.008** of the population has cancer. Thus

  P(**cancer**) = P(h_1) = 0.008

  P(**¬cancer**) =P(h_2) = 0.992

- **Conditional probability:**

  P(**+** | h_1) = 0.98, P(**—** | h_1) = 0.02

  P(**+** | h_2) = 0.03, P(**—** | h_2) = 0.97

- **Posterior knowledge:**

  Blood test is **+** for this patient.

Machine Intelligence

Example to see how Bayes Idea work

Initial belief :

Probability that population has cancer.

P(**cancer**) = P(**h1**) = **0.008**

Probability that population doesn't have cancer

P(**not cancer**) = P(**h2**) =**0.992**

		Test Prediction	
		+	-
Reality	+	0.98	0.02
	-	0.03	0.97

From Bayes theorem we arrive at this proportionality

$$P(\text{cancer} \mid +) \;\alpha\; P(+ \mid \text{cancer}) \cdot P(\text{cancer}) = 0.98 \times 0.008 = 0.00784$$

$$P(\text{not cancer} \mid +) \;\alpha\; P(+ \mid \text{not cancer}) \cdot P(\text{not cancer}) = 0.03 \times 0.992 = \mathbf{0.02976}$$

0.02976 > 00784

So we can say that P(not cancer| +) is more probable

Machine Intelligence

Example to see how Bayes Idea work

$$h_{MAP} \equiv \underset{h \epsilon H}{\operatorname{argmax}} P(h|D)$$

$$\equiv \underset{h \epsilon H}{\operatorname{argmax}} \{ P(h_1|+), \{ P(h_2|+) \}$$

$$\equiv \underset{h \epsilon H}{\operatorname{argmax}} \{ P(+|h_1)P(h_1), P(+|h_2)P(h_2) \}$$

$$\equiv \underset{h \epsilon H}{\operatorname{argmax}} \{ 0.98 * 0.08, \quad 0.03 * 0.992 \}$$

$$\equiv \underset{h \epsilon H}{\operatorname{argmax}} \{ 0.0078, 0.0298 \}$$

$h_{MAP} \equiv h_2$ (¬ cancer)

The most probable hypothesis is h2(The patient does not have cancer)

Example to see how Bayes Idea work

Normalization of values - to get probabilities

The exact **posterior probabilities** can be determined by normalizing the above properties to 1

$$P(h_1|+) = \frac{0.0078}{0.0078 + 0.0298} = 0.21$$

$$P(h_2|+) = \frac{0.0298}{0.0078 + 0.0298} = 0.79$$

$\Rightarrow$ the result of Bayesian inference depends strongly on the prior probabilities, which must be available in order to apply the method directly

MACHINE INTELLIGENCE

Applying Bayes theorem to Concept Learning / Brute force MAP Learning

Preet Kanwal

Department of Computer Science and Engineering

Machine Intelligence

Applying Bayes Theorem to Concept learning

Let us look at one use of Bayes theorem in concept learning.

To apply Bayes theorem to concept learning, **we make some simplifying assumptions:**

1) The hypothesis space is conjunctive (same as before)
2) Training data is noise free (no error)
   ⇒ Any hypothesis which is consistent with data is okay with us.
3) We always assume that the target concept is in the Hypothesis space.
   if it's not in the hypothesis space, we are souped anyway because we will not be able to learn the concept at all.
4) **Crucial Idea :** No reason to believe that one hypothesis is more probable than other. (As we do not have prior probabilities data P(h), we assume all the hypothesis are equally likely.)
   if you do not want to make this assumption then you should specify what are the prior probabilities.

Machine Intelligence

Brute-force MAP learning

Let us calculate the posterior probability P(h|D) of each hypothesis h given in observed training data D Let's compute using the famous Bayes Theorem.

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

Question :

Given these assumptions

a) what values will P(h) get?
b) What values will P(D|h) get?
c) What will be the value of P(D)? (although we ignored this previously)

Applying Bayes Theorem to Concept learning

Given these assumptions what values will P(h) get?

- As we stated earlier, we have no priori knowledge to believe that any hypothesis is more probable than the other, so it is reasonable to assign the **same probability** to every hypothesis h in H (which is the hypothesis space)

- Furthermore, because we assume the target concept is contained in H we require that all these prior probabilities **sum upto 1**

When we combine these two constraint we get:

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

Machine Intelligence

Applying Bayes Theorem to Concept learning

What values will P(D|h) be given now?

P(D|h) is the probability of observing the target values D=<d1….dm> for the fixed set of instances <X1, X2…. Xm>
Again due to the assumptions that we made before of the data being noise free and probability of observing di given h is given by:

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

In other words, the probability of getting data D given hypothesis h is
1 if h is consistent with D
0 otherwise

⇒ For any consistent hypothesis, P(D|h) = 1 and for any inconsistent hypothesis, P(D|h) = 0

What is V $S_{H;D}$

The version space, V S(H,D), with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D

Set of all consistent hypothesis

Machine Intelligence

Brute-force MAP learning

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

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|}$$

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|}$$

$$= \frac{|VS_{H,D}|}{|H|}$$

Brute-force MAP learning

Now, First consider the case where **h is inconsistent** with training data D:
So that means P(D|h) =0

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

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

Brute-force MAP learning

If **h is consistent** with training data D, then P(D|h) is 1

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

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

Number of consistent hypothesis over all the hypothesis cardinality(version space) / cardinality (hypothesis space)

$$= \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$$

Brute-force MAP learning

So to summarise,

If h is inconsistent with training data D $P(h|D)=0$

If h is consistent hypothesis has posterior probability $1/|VS_{H,D}|$ and is a MAP hypothesis.

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

Bayes Theorem

So, Bayes theorem tells us that any consistent hypothesis has the same A posteriori probability.

if you are using MAP, according to the Bayes theorem any consistent hypothesis is fine.

Bayes theorem gives us a justification why Find-S, Candidate Elimination algorithms are okay. Both are justifiable from Bayes perspective.

Machine Intelligence

Brute-force MAP learning

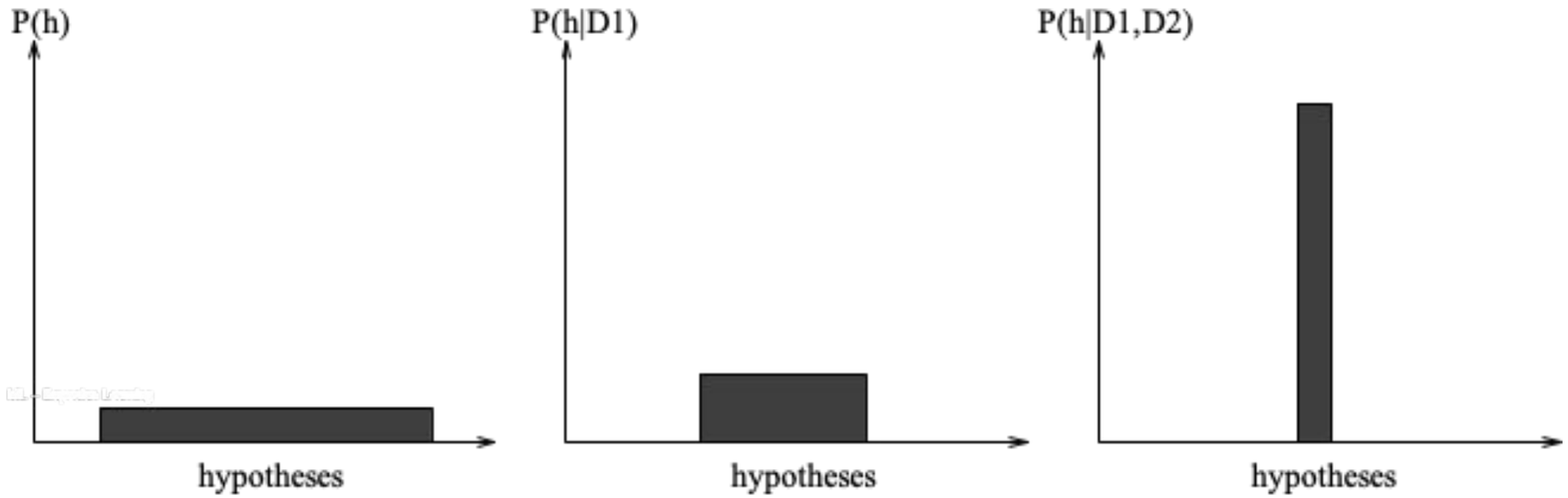


Figure (a) Figure (b) Figure (c)

As data is added, certainty of hypotheses increases.

Interpretations and Conclusions from the graph above:

(a) Uniform priors assign **equal probability** to each hypothesis.

(b) As training data increases first to Dl (b), then to Dl ^D2

(c), the posterior probability of inconsistent hypotheses becomes zero, while posterior probabilities increase for hypotheses remaining in the version space.

while the total probability summing to 1 is shared equally among the remaining consistent hypotheses

Credits: "Machine Learning", Tom Mitchell, McGraw Hill Education (India), 2013.

Brute-force MAP learning

Training:

Choose the hypothesis with the highest posterior probability

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

Testing:

Given x, compute $h_{MAP}(x)$,where x is a new data point to be classified.

Drawback:

Requires to compute all probabilities $P(D|h)$ and $P(h)$.

This brute-force MAP learning algorithm may be computationally infeasible, as it requires applying the Bayes theorem to all $h \in$ H. However, this is still useful as a standard against which other concept learning approaches may be judged.

MACHINE INTELLIGENCE

Naïve Bayes Classifier

Preet Kanwal

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Naive Bayes Classifier

Naive Bayes Classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and the data instance could belong to one of the categories.

The label of the data instance could be binary(v1, v2) or can have multiple classes(v1, v2, v3 ….).

In some domains, the performance of Naive Bayes has shown to be comparable to that of Neural Networks and Decision Tree Learning.

In this section, we will look at the Bayesian approach to classify a new instance X_{new}.

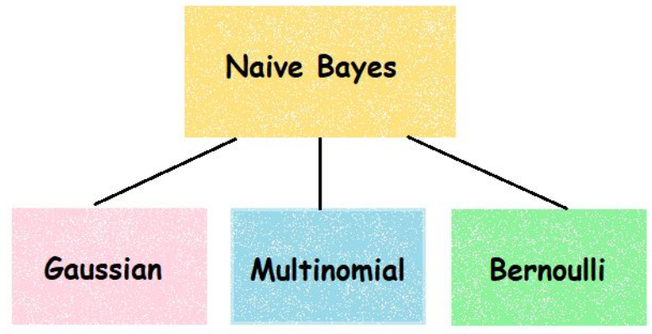
MACHINE INTELLIGENCE

Naive Bayes Classifier -- Types

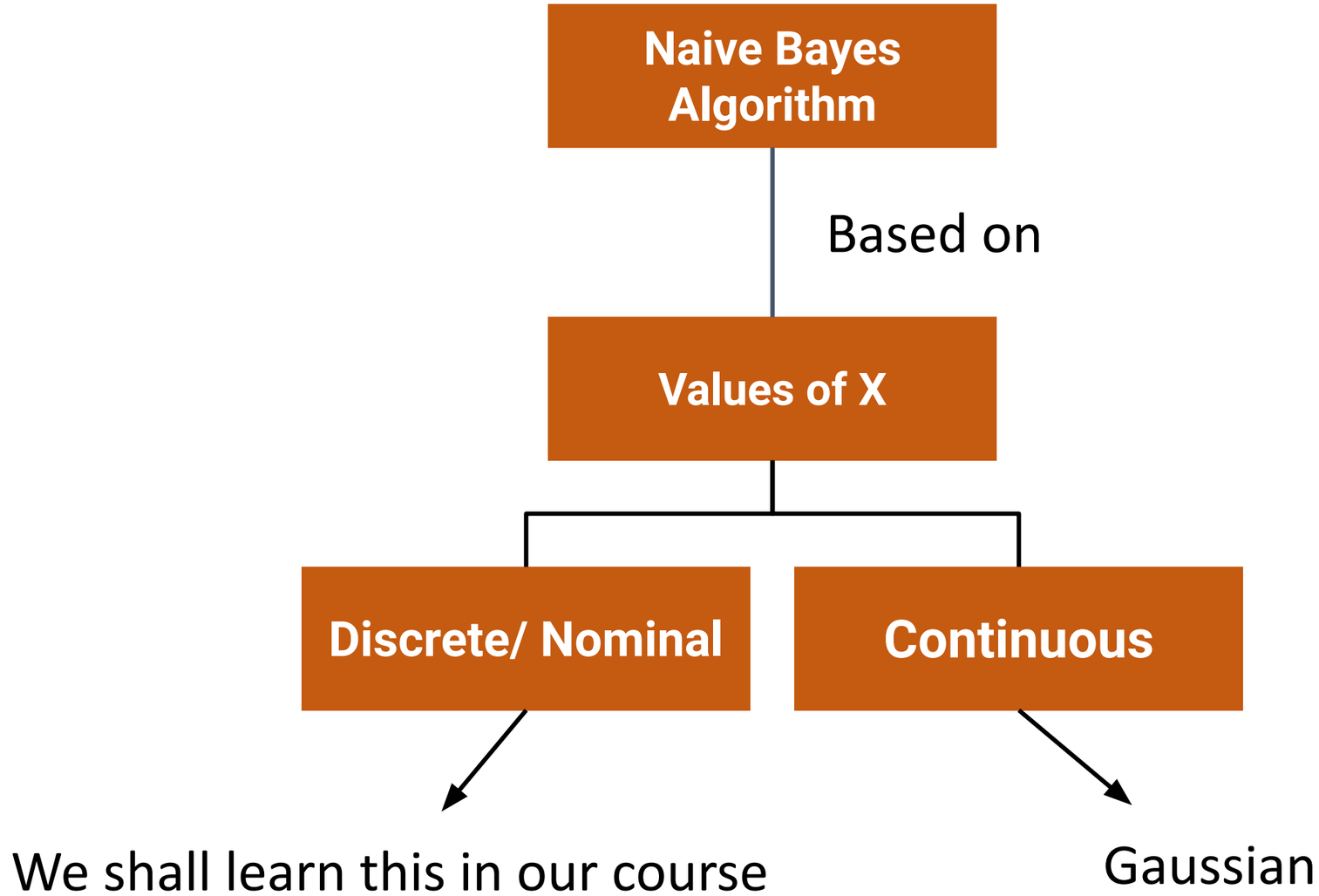
- **Gaussian**: It is used for continuous data. It is used in classification and it assumes that features follow a normal distribution.

- **Multinomial**: It is used for discrete counts. For example, let's say, we have a text classification problem.
  - Example "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.



Naive Bayes Classifier



MACHINE INTELLIGENCE

Naive Bayes Classifier

Most probable classification of a new instance X_{new}.

X_{new} is defined by d attribute values

$$X_{new} \in \mathbb{R}^d \qquad X_{new} = \begin{bmatrix} X_1 = a_1 \\ X_2 = a_2 \\ \vdots \\ X_d = a_d \end{bmatrix}$$

Naive Bayes Classifier

we must predict the class/label of this new instance.

Assume the target function $f(x)$ can take on any value from some finite set V.

↳ set of possible classes

$V = \{ v_1, v_2 \ldots v_c \}$ ← The various categories.

for example,

→ In the classification of an insurance application, we could have 4 classes in our dataset. So,

$V = \{ v_1 = \text{accept with standard premium},$
$v_2 = \text{accept with double premium},$
$v_3 = \text{accept with Super premium},$
$v_4 = \text{Reject the application} \}$

→ In the play tennis example, the label was binary,

$V = \{ v_1 = \text{yes}, v_2 = \text{no} \}$

MACHINE INTELLIGENCE

Naive Bayes Classifier

We can use Bayesian approach to classify the new instance.
The most probable classification / target value (V_{MAP}) would be the one for which the Probability of observing that particular class will be the maximum given the data (X_{new})

$$V_{map} = \underset{v_j^o \in V}{\operatorname{argmax}} \; P(v_j^o \mid X_{new})$$

Using Bayes theorem,

$$V_{map} = \underset{v_j^o \in V}{\operatorname{argmax}} \; \frac{P(X_{new} \mid v_j^o) * P(v_j^o)}{P(X_{new})}$$

← same for all (hence ignored)

MACHINE INTELLIGENCE

Naive Bayes Classifier

Given the data, $P(V_j)$ can be easily computed by looking at the last column where we have our label.

$P(V_1) = P(\text{Can we Pet} = \text{yes}) = \frac{8}{14}$

similarly

$P(V_2) = P(\text{Can we Pet} = \text{no}) = \frac{6}{14}$

	Animals	Size of Animal	Body Color	Can we Pet them
0	Dog	Medium	Black	Yes
1	Dog	Big	White	No
2	Rat	Small	White	Yes
3	Cow	Big	White	Yes
4	Cow	Small	Brown	No
5	Cow	Big	Black	Yes
6	Rat	Big	Brown	No
7	Dog	Small	Brown	Yes
8	Dog	Medium	Brown	Yes
9	Cow	Medium	White	No
10	Dog	Small	Black	Yes
11	Rat	Medium	Black	No
12	Rat	Small	Brown	No
13	Cow	Big	White	Yes

MACHINE INTELLIGENCE

Naive Bayes Classifier

Given $V_{map} = \underset{v_j \in V}{\text{argmax}} \; P(X_{new} \mid v_j) * P(v_j)$

↓ easy to compute

Let us see how to compute $P(X_{new} \mid v_j)$:

We know X_{new} is described by tuple of attribute values $\langle a_1, a_2 \ldots a_d \rangle$

So, we need to find

$$P(x_1 = a_1, x_2 = a_2 \ldots x_d = a_d \mid v_j)$$

This is the probability of $x_1 = a_1$ and $x_2 = a_2 \ldots$ and $x_d = a_d$ occurring together given the class/label v_j.

— gts a Joint probability calculation.

MACHINE INTELLIGENCE

Naive Bayes Classifier

We know, how to compute joint probability:

$$P(X_1, X_2 \ldots X_d \mid V_j) = P(X_1 \mid V_j) * P(X_2 \mid X_1 \cap V_j) * P(X_3 \mid X_1 \cap X_2 \cap V_j) * P(X_4 \mid X_1 \cap X_2 \cap X_3 \cap V_j) \ldots \ldots * P(X_d \mid X_1 \cap X_2 \ldots \cap X_{d-1} \cap V_j)$$

Used a shorthand notation instead of writing

$P(X_1 = a_1, X_2 = a_2 \ldots X_d = a_d \mid V_j)$

Hence, we will have to compute multiple conditional probabilities, and this number increases exponentially with increase in # of attributes.

MACHINE INTELLIGENCE

Naive Bayes Classifier

for example,

Consider only 2 attributes X_1, X_2 both of which are binary.

Assume the label is also boolean.

Then we need to compute the following joint probabilities

$X_1\ X_2$

$P(00|V_1)$ $P(00|V_2)$

$P(01|V_1)$ $P(01|V_2)$

$P(10|V_1)$ $P(10|V_2)$

$P(11|V_1)$ $P(11|V_2)$

feature space (# combinations)

$= 2 \times 2 = 4$

joint probabilities to compute

= feature space × #labels

$= 4 \times 2 = 8$

conditional probabilities to compute

$= (\#\text{labels})^{\text{feature\_space}} = 2^4 = 16$

MACHINE INTELLIGENCE

Naive Bayes Classifier

Instead of computing so many probabilities, we make an assumption that all features are independent

— This is the naiveness

So now, the joint probability can be considered as simply product of individual probabilities.

This simplifies the computations enormously and is reasonable to apply. This method is better if we have more number of training instances.

Naive Bayes Classifier

Now, for the same example with 2 binary attributes X_1 and X_2 & a boolean label, we will only have to compute the following probabilities :-

for X_1 :- $P(X_1=0 \mid V_1=0)$, $P(X_1=0 \mid V_1=1)$
$P(X_1=1 \mid V_1=0)$, $P(X_1=1 \mid V_1=1)$ } 4

for X_2 :- $P(X_2=0 \mid V_1=0)$, $P(X_2=0 \mid V_1=1)$
$P(X_2=1 \mid V_1=0)$, $P(X_2=1 \mid V_1=1)$ } 4

Prob's to compute = # classes * $\sum_{\text{All features}}$ # values a feature can take

$= 2 * (2+2)$

$= 8$

MACHINE INTELLIGENCE

Naive Bayes Classifier

Let us look at a case, where we have 3 features (each of which are binary) and a binary label.

The joint probabilities to compute:

$P(000|v_1)$ $P(000|v_2)$

$P(001|v_1)$ $P(001|v_2)$

$P(010|v_1)$ $P(010|v_2)$

$P(011|v_1)$ $P(011|v_2)$

$P(100|v_1)$ $P(100|v_2)$

$P(101|v_1)$ $P(101|v_2)$

$P(110|v_1)$ $P(110|v_2)$

$P(111|v_1)$ $P(111|v_2)$

feature space * #labels

$= 2 \times 2 \times 2 * 2$

$= 16$

conditional probs to compute $= 2^8$

Assuming independence of features

probs to compute

$= \#\text{classes} * \sum_{\text{All attributes}} \#\text{values each attr can take}$

$= 2 * (2 + 2 + 2) = 12$

Naive Bayes Classifier

Assuming features are independent, one must only compute the following probabilities

for X_1 : $P(X_1 = 0 \mid V_1)$, $P(X_1 = 1 \mid V_1)$
$P(X_1 = 0 \mid V_2)$, $P(X_1 = 1 \mid V_2)$ } 4 +

for X_2 : $P(X_2 = 0 \mid V_1)$, $P(X_2 = 1 \mid V_1)$
$P(X_2 = 0 \mid V_2)$, $P(X_2 = 1 \mid V_2)$ } 4 +

for X_3 : $P(X_3 = 0 \mid V_1)$, $P(X_3 = 1 \mid V_1)$
$P(X_3 = 0 \mid V_2)$, $P(X_3 = 1 \mid V_2)$ } 4

Naive Bayes Classifier

In general, Assume the label has c classes i.e. $|V| = c$

if we have 5 features

Each can take (2 values) (3 values) (5 values) (4 values) (2 values)

Probs to compute assuming Independence $= \sum_{\text{All features}}$ # classes * # values a feature can take

$= \#\text{classes} * \sum_{\text{All features}} \#\text{values a feature can take}$

$= c * (2+3+5+4+2)$

$= c * 16$ [instead of $C^{2*3*5*4*2}$]

MACHINE INTELLIGENCE

Naive Bayes Classifier

→ As the number of attributes increase, the difference in the # of probabilities to compute assuming independence assumption and without this assumption becomes dramatic.

→ Because one is an exponential growth, the other one is a linear one.

→ If the problem size is small, we can work out the joint probability.

→ However, in real life we never get such simple problems and hence we end up making this assumption.

Note :
Estimating priors etc is "easy" only in the toy examples that we use! In real life, we need very large data sets to make reliable probability estimates and then we will have all the data preprocessing issues associated with large data sets!

Naive Bayes Classifier

Coming back to the original problem,

$$V_{map} = \underset{v_j \in V}{\operatorname{argmax}} \; P(v_j | X_{new})$$

$$= \underset{v_j \in V}{\operatorname{argmax}} \; \frac{P(X_{new} | v_j) * P(v_j)}{P(X_{new})} \leftarrow \text{ignored}$$

$$= \underset{v_j \in V}{\operatorname{argmax}} \; P(X_1 = a_1, X_2 = a_2 \ldots X_d = a_d | v_j) * P(v_j)$$

$$= \underset{v_j \in V}{\operatorname{argmax}} \; P(X_1 = a_1 | v_j) * P(X_2 = a_2 | v_j) \ldots * P(X_d = a_d | v_j) * P(v_j)$$

$$= \underset{v_j \in V}{\operatorname{argmax}} \; P(v_j) * \prod_{i=1}^{d} P(X_i = a_i | v_j)$$

To summarize,

The Naive Bayes learning method involves a learning step in which various $P(V_j)$ and $P(X_i = a_i \mid V_j)$ terms are estimated based on their frequencies over training data.

Set of these estimates corresponds to the learned hypothesis.

In this case, the space of possible hypothesis is the space of possible values that can be assigned to various $P(V_j)$ and $P(X_i = a_i / V_j)$ terms.

So, Hypothesis is formed without searching, simply by counting the frequency of various data combinations within training examples.

Naive Bayes Classifier
Example 1

MACHINE INTELLIGENCE

Naive Bayes Classifier

I.T Industry	Chemicals	Crops	Exports	Economy?
Good	Avg	Avg	Good	Great
Avg	Avg	Good	Good	Ok
Poor	Good	Good	Avg	Ok
Good	Avg	Good	Good	Great
Avg	Good	Good	Avg	Great
Good	Avg	Avg	Avg	Bad
Poor	Avg	Good	Avg	Bad
Avg	Avg	Avg	Avg	Bad
Good	Good	Avg	Avg	Ok

Consider the following dataset, apply Naive Bayes Classification technique to classify the following instance

Xnew =
<IT=good, Chemicals = Avg,
Crops = good, Exports = Avg>

MACHINE INTELLIGENCE

Naive Bayes Classifier

We need to find the class for

$X_{new} = \langle IT = good, Chemicals = avg, Crops = good, Exports = avg \rangle$

We need to determine, which of the following is the largest:

$P(Great \mid X_{new})$, $P(OK \mid X_{new})$, $P(Bad \mid X_{new})$

V_1 V_2 V_3

$$V_{map} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) * \prod_{i=1}^{4} P(X_i = a_i \mid v_j)$$

$P(Great) = \frac{3}{9}$ $P(OK) = \frac{3}{9}$ $P(Bad) = \frac{3}{9}$

∴ All classes are equally likely

$$V_{map} = \underset{v_j \in V}{\operatorname{argmax}} \prod_{i=1}^{4} P(X_i = a_i \mid v_j)$$

MACHINE INTELLIGENCE

Naive Bayes Classifier

$X_{new} = \langle IT = good, Chemicals = avg, Crops = good, Exports = Avg \rangle$

$$\prod_{i=1}^{4} P(X_i = a_i \mid Great) = P(IT = good \mid Great) * P(Chemicals = avg \mid Great) * P(crops = good \mid Great) * P(Exports = Avg \mid Great)$$

$$= \frac{2}{3} * \frac{2}{3} * \frac{2}{3} * \frac{1}{3} = \frac{8}{34}$$

$$\prod_{i=1}^{4} P(X_i = a_i \mid OK) = P(IT = good \mid OK) * P(chemicals = avg \mid OK) * P(Crops = good \mid OK) * P(Exports = Avg \mid OK)$$

$$= \frac{1}{3} * \frac{1}{3} * \frac{2}{3} * \frac{2}{3} = \frac{4}{34}$$

$$\Rightarrow \prod_{i=1}^{4} P(X_i = a_i \mid bad) = P(IT = good \mid bad) * P(chemicals = avg \mid bad) * P(crops = good \mid bad) * P(Exports = avg \mid bad)$$

$$= \frac{1}{3} * 1 * \frac{1}{3} * 1 = \frac{1}{9}$$

$$V_{map} = \underset{v_j \in V}{\operatorname{argmax}} \left(\prod_{i=1}^{4} P(X_i = a_i \mid Great), \prod_{i=1}^{4} P(X_i = a_i \mid OK), \prod_{i=1}^{4} P(X_i = a_i \mid bad) \right)$$

⇒ this given instance is classified as **Bad**.

Naive Bayes Classifier -- Zero frequency problem

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.

Naive Bayes Classifier

I.T Industry	Chemicals	Crops	Exports	Economy?
Good	Avg	Avg	Good	Great
Avg	Avg	Good	Good	Ok
Poor	Good	Good	Avg	Ok
Good	Avg	Good	Good	Great
Avg	Good	Good	Good	Great
Good	Avg	Avg	Avg	Bad
Poor	Avg	Good	Avg	Bad
Avg	Avg	Avg	Avg	Bad
Good	Good	Avg	Avg	Ok

To classify X_{new}= <IT=good, Chemicals = Avg, Crops = good, Exports = Avg>

Making this change in the dataset we get
P(Exports = Avg | Great) = 0
because of which the entire product of prob's become 0.

That means whenever

$$\hat{P}(a_i|v_j) = 0, \text{ and...}$$

$$\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$$

In order to avoid making the entire product of probabilities as zero, introduce **m virtual rows into the dataset with target value = vj** according to prior probability p(probability that the attribute takes a specific value ai when target class = vj).

What should be the value of m??

--- Designers choice

--- Depends on the problem

Naive Bayes Classifier
Avoiding Zero Frequency Problem

Naive Bayes Classifier (m-estimate of probability)

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

where

- n is number of training examples for which $v = v_j$,
- n_c number of examples for which $v = v_j$ and $a = a_i$
- p is prior estimate for $\hat{P}(a_i|v_j)$
- m is weight given to prior (i.e. number of "virtual" examples)

Naive Bayes Classifier

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

In the additional virtual rows added for target class= vj, m*p rows have a specific value ai for an attribute.

p - probability that the attribute takes a specific value ai when target class = vj.

A typical method for choosing p in the absence of other information is to assume uniform priors; that is, if an attribute has k possible values we set p = i. For example, in estimating P(Exports = AvgIGreat) we note the attribute Exports has two possible values, so uniform priors would correspond to choosing **p = .5**. So, if we are adding **m = 4 rows**, then, 4 * 0.5 = 2 of the rows will have Exports = Avg when target class = Great.

⇒ P(Exports = AvgIGreat) = 2/ (3+4) = 2/7

Add one smoothing -- also called Laplace Smoothing

Standard practice is to choose m to be the number of possible values that particular attribute can take. Here, as Exports can take two values we can choose m=2.

Since Exports = avg never happened before for economy = great. We can have p = 1/m as 1 one of the m instances will now have the value Exports = avg whenever economy = great.

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$$

P(ai|vj) = (0 + m* 1/m) / n + m = 1/(n+m)

Note :

- The m value should not be very high that it dominates whatever values are already present.
- p should not be so far away from actual values.

Naive Bayes Classifier -- Numerical Underflow problem

$P(h|D) \propto P(D|h) * P(h)$

A posterior density is (proportional to) a likelihood function times a prior distribution. The likelihood function, P(D|h) is a product.

The number of data points(in our case the number of features used in defining an instance) is the number of terms in the product.

If these numbers are less than 1, and you multiply enough of them together, the result will be too small to represent in a floating point number and your calculation will underflow to zero. Then subsequent operations with this number, such as dividing it by another number that has also underflowed to zero, may produce an infinite or NaN result.

(Underflow problem -- the number is super small for your computer programming language to represent)

Naive Bayes Classifier -- Numerical Underflow problem

For example, a real dataset might have over 100 features and there will be 101 multiplications and all of these 101 values could be < 1, so the product can diminish to 0.

How do we avoid this?

- Normalization : Scale up the values [multiply by a large number]
- work in log space, and manipulate the log of probabilities instead of probabilities. Products of probabilities will become sums of log probabilities.

Naive Bayes Classifier
Example 2 - Text Classification

MACHINE INTELLIGENCE

Naive Bayes Classifier

Six documents are presented to a Naïve Bayes Text classifier with their labels. The contents are as follows.

D1: An inspiring address (Campaigning)

D2: An aggressive speech (Campaigning)

D3: It was a cowardly act. (Law and Order)

D4: He spoke like a coward. (Campaigning)

D5: A threatening speech (Law and Order)

D6: An act of aggression (Law and Order)

Find the classification for :

D7: An aggressive and threatening address

Note1: Ignore all stop-words (“like” and “He” are also stop-words)

Note2: Lemmatize the words. Example: feel, felt, feeling are all the same word “feel”

Note3: Use Add-one laplace-smoothing if required.

Naive Bayes Classifier -- Text Classification

Use BoW(Bag of Words) approach

--- all words are broken into count and frequency with no preference to a word in particular, all words have same weight.

Vocabulary V

consists of the union of all the word types in all classes, not just the words in one class

https://web.stanford.edu/~jurafsky/slp3/4.pdf

Naive Bayes Classifier -- Text Classification

Unseen Words

What do we do about words that occur in our test data but are not in our vocabulary at all because they did not occur in any training document in any class? The unknown word solution for such unknown words is to **ignore them**—remove them from the test document and not include any probability for them at all.

Stop Words

Finally, some systems choose to completely ignore another class of words: stop words, very frequent words like the and a. This can be done by sorting the vocabulary by frequency in the training set, and defining the top 10–100 vocabulary entries as stop words, or alternatively by using one of the many predefined stop word lists available online. Then each instance of these stop words is simply removed from both training and test documents as if it had never occurred.

In most text classification applications, however, using a stop word list doesn't improve performance, and so it is more common to make use of the entire vocabulary and not use a stop word list.

https://web.stanford.edu/~jurafsky/slp3/4.pdf

Naive Bayes Classifier -- Text Classification

- We assume probability of word occurrence is independent of position within the text.

D1: An inspiring address (Campaigning)
D2: An aggressive speech (Campaigning)
D3: It was a cowardly act. (Law and Order)
D4: He spoke like a coward. (Campaigning)
D5: A threatening speech (Law and Order)
D6: An act of aggression (Law and Order)

Find the classification for :

D7: An aggressive and threatening address

D1 : Inspiring Address
D2 : Aggressive Speech
D3 : Cowardly act
D4 : Speech coward
D5 : threatening speech
D6 : Act aggression

D7 : Aggression threatening address

MACHINE INTELLIGENCE

Naive Bayes Classifier -- Text Classification

D1 : Inspiring Address	Campaigning
D2 : Aggressive Speech	Campaigning
D3 : Cowardly act	Law & Order
D4 : Spoke coward	Campaigning
D5 : threatening speech	Law & Order
D6 : Act aggression	Law & Order

D7 : Aggression threatening address ?

We must find out which of the following probabilities is larger:
P(Campaigning | Aggression threatening address) or **P(Law & Order | Aggression threatening address)?**

		Word Count	
#	Word	Campaigning	Law & Order
1	Inspiring	1	0
2	Address	1	0
3	Aggressive	1	1
4	Speech	1+1	1
5	Cowardly	1	1
6	Act	0	1+1
7	Threatening	0	1
	Total Words	6	6

Naive Bayes Classifier -- Text Classification

		Word Count	
#	Word	Campaigning	Law & Order
1	Inspiring	1	0
2	Address	1	0
3	Aggressive	1	1
4	Speech	1+1	1
5	Cowardly	1	1
6	Act	0	1+1
7	Threatening	0	1
	Total Words	6	6

P(Campaigning | Aggression threatening address) =

P (Aggression | Campaigning) *

P (threatening | Campaigning) *

P (address | Campaigning) *

P (Campaigning)

P(Law & Order | Aggression threatening address) =

P (Aggression | Law & Order) *

P (threatening | Law & Order) *

P (address | Law & Order) *

P (Law & Order)

Naive Bayes Classifier -- Text Classification

		Word Count	
#	Word	Campaigning	Law & Order
1	Inspiring	1	0
2	Address	1	0
3	Aggressive	1	1
4	Speech	1+1	1
5	Cowardly	1	1
6	Act	0	1+1
7	Threatening	0	1
	Total Words	6	6

P (Aggression | Campaigning) = 1/6

P (threatening | Campaigning) = 0/6

P (address | Campaigning) = 1/6

P (Campaigning) = 3/6

⇒ P(Campaigning | Aggression threatening address) =

P (Aggression | Campaigning) *

P (threatening | Campaigning) *

P (address | Campaigning) *

P (Campaigning)

= 0

Naive Bayes Classifier -- Text Classification

		Word Count	
#	Word	Campaigning	Law & Order
1	Inspiring	1 + 1	0
2	**Address**	**1 + 1**	**0**
3	**Aggressive**	**1 + 1**	**1**
4	Speech	1+1 + 1	1
5	Cowardly	1 + 1	1
6	Act	0 + 1	1+1
7	**Threatening**	**0 + 1**	**1**
	Total Words	6 + 7	6

Let us apply Laplace Smoothing:

P (Aggression | Campaigning) = 1/6

P (threatening | Campaigning) = 0/6

P (address | Campaigning) = 1/6

P (Campaigning) = 3/6

P (Aggression | Campaigning) = 2/13

P (threatening | Campaigning) = 1/13

P (address | Campaigning) = 2/13

P (Campaigning) = 3/6

=>P(Campaigning | Aggression threatening address) =

2/13 * 1/13 * 2/13 * 3/6

Naive Bayes Classifier -- Text Classification

		Word Count	
#	Word	Campaigning	Law & Order
1	Inspiring	1	0
2	Address	1	0
3	Aggressive	1	1
4	Speech	1+1	1
5	Cowardly	1	1
6	Act	0	1+1
7	Threatening	0	1
	Total Words	6	6

P (Aggression | Law & Order) = 1/6

P (threatening | Law & Order) = 1/6

P (address | Law & Order) = 0/6

P (Law & Order) = 3/6

P(Law & Order | Aggression threatening address) =

P (Aggression | Law & Order) *

P (threatening | Law & Order) *

P (address | Law & Order) *

P (Law & Order) = **0**

Naive Bayes Classifier -- Text Classification

		Word Count	
#	Word	Campaigning	Law & Order
1	Inspiring	1	0 + 1
2	**Address**	**1**	**0 + 1**
3	**Aggressive**	**1**	**1 + 1**
4	Speech	1+1	1 + 1
5	Cowardly	1	1 + 1
6	Act	0	1+1 + 1
7	**Threatening**	**0**	**1 + 1**
	Total Words	6	6 + 7

Let us apply Laplace Smoothing:

P (Aggression | Law & Order) = 1/6
P (threatening | Law & Order) = 1/6
P (address | Law & Order) =0/6
P (Law & Order) = 3/6

P (Aggression | Law & Order) = 2/13
P (threatening | Law & Order) = 2/13
P (address | Law & Order) =1/13
P (Law & Order) = 3/6

=>P(Law & Order | Aggression threatening address) =
2/13 * 2/13 * 1/13 * 3/6

Naive Bayes Classifier -- Text Classification

P(Campaigning | Aggression threatening address) = 2/13 * 1/13 * 2/13 * 3/6

P(Law & Order | Aggression threatening address) = 2/13 * 2/13 * 1/13 * 3/6

Naive bayes can randomly return a class as the posterior probabilities are same for all the classes.

Naive Bayes Classifier
Example 3 -- For practice

Naive Bayes Classifier

We'll use a sentiment analysis domain with the two classes positive (+) and negative (-), and take the following miniature training and test documents simplified from actual movie reviews. [do not ignore stopwords]

Cat		Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

MACHINE INTELLIGENCE

Naive Bayes Classifier

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

	Word	+	-
1	Just	0	1
2	Plain	0	1
3	Boring	0	1
4	Entirely	0	1
5	Predictable	0	1
6	And	0	1+1
7	lacks	0	1
8	energy	0	1

	Word	+	-
9	no	0	1
10	surprises	0	1
11	very	1	1
12	few	0	1
13	laughs	0	1
14	powerful	1	0
15	the	1+1	0
16	most	1	0
17	fun	1	0
18	film	1	0
19	of	1	0
20	summer	1	0
	Total	9	14

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

The word with doesn't occur in the training set, so we drop it completely (as mentioned above, we don't use unknown word models for naive Bayes). The likelihoods from the training set for the remaining three words "predictable", "no", and "fun", are as follows,

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

For the test sentence S = "predictable with no fun", after removing the word 'with',

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

The model thus predicts the class *negative* for the test sentence.

Naive Bayes Classifier
Example 4 -- For practice

MACHINE INTELLIGENCE

Naive Bayes Classifier

The following dataset contains loan information and can be used to try to predict whether a borrower will default (the last column is the classification). Use the naïve Bayes method to determine whether a loan

X=(Home Owner = No, Marital Status=Married, Income=High)

should be classified as a Defaulted Borrower or not. So, determine which is larger, P(Yes|X) or P(No|X).

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	High	**No**
2	No	Married	High	**No**
3	No	Single	Low	**No**
4	Yes	Married	High	**No**
5	No	Divorced	Low	**Yes**
6	No	Married	Low	**No**
7	Yes	Divorced	High	**No**
8	No	Single	Low	**Yes**
9	No	Married	Low	**No**
10	No	Single	Low	**Yes**

Solution:

P(Yes) = .3 and P(No) = .7

P(X|No) = P(Home Owner=No|No) x P(Status=Married|No) x P(Income=High|No)

= 4/7 x 4/7 x 4/7 = 0.187

P(X|Yes) = P(Home Owner=No|Yes) x P(Status=Married|Yes) x P(Income=High|Yes)

= 3/3 x 0/3 x 0/3 = 0

Now use equation 1, but since both sides of the comparison will have P(X) in the denominator, we can ignore it. Ignoring that, we have:

P(No|X) = 0.187 x 0.7 = .1309

P(Yes|X) = 0 x .3 = 0

So, since .1309 is larger than 0, we will classify the loan as "No".

Note: these are not actually probabilities since we did not divide by P(X).

Naive Bayes Classifier
Example 5 -- For practice

MACHINE INTELLIGENCE

Example- Play Tennis

PlayTennis: training examples

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

MACHINE INTELLIGENCE

Example- Play Tennis

PlayTennis: training examples

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

- Recall the data set that we used to build a Decision Tree
- with 14 samples of target concept Play Tennis with values YES or NO
- Each day is described by attributes, (OUTLOOK,TEMPERATURE,HUMIDITY,WIND)
- use NB classifier and the training data from the table to classify the following novel instance.

PlayTennis = ?
New instance to classify:
(Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong)

Naive Bayes classifier: $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j)$

MACHINE INTELLIGENCE

Example- Play Tennis

Learning Phase

Probabilities of the different target values can easily be estimated based on their frequency over the 14 training examples

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=Yes	Play=No
High	3/9	4/5
Moderate	6/9	1/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

P(Play= Yes) =9/14

P(Play=No) = 5/14

MACHINE INTELLIGENCE

Example- Play Tennis

Test Phase:
Given a instance of

X' = (Outlook=Sunny, Temperature=Cool, Humidity= High, Wind=Strong)

From the information in the table
For P(Yes|X') we consider the values in the yellow boxes
For P(No|X') we consider the light blue boxes

From our previous slides we know that:
P(Play= Yes) =9/14=0.64
P(Play=No) = 5/14=0.36

Outlook	Play=Yes	Play=No
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=Yes	Play=No
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=Yes	Play=No
High	3/9	4/5
Moderate	6/9	1/5

Wind	Play=Yes	Play=No
Strong	3/9	3/5
Weak	6/9	2/5

Decision making with MAP Rule:

$$P(Yes|x') \approx [P(Sunny|Yes)P(Cool|Yes)P(High|Yes)P(Strong|Yes)]P(Play=Yes) = 0.0053$$
$$P(No|x') \approx [P(Sunny|No)P(Cool|No)P(High|No)P(Strong|No)]P(Play=No) = 0.0206$$

Given the fact $P(Yes|x') < P(No|x')$, we label x' to be "No".

Naive Bayes Classifier
Example 6 -- For practice

MACHINE INTELLIGENCE

Naive Bayes Classifier -Text Classification

- Consider the following data set
- The task is to classify the sentence "A very close game" as sports or not sports
- In this dataset we do not have numbers but we have only text
- **We need to convert all this text into numbers that we can use for calculation. HOW??**
- One solution is to use frequency of words
- Ignore word order and sentence construction
- Treat every document as a set of words it contains.
- Now the feature used in this case is the counts of words i.e(words frequency)
- It's a simplistic approach,but works surprisingly well

sentence	class
A great game	sports
election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Naive Bayes Classifier -Text Classification

- Now, we need to transform the probability we want to calculate into something that can be calculated using word frequencies.

- Bayes Theorem for example:

$$P(\text{sports/a very close game}) = \frac{P(\text{a very close game/sports}) x P(\text{sports})}{P(\text{a very close game})}$$

- since in our classifier, we are just trying to find out which category has bigger probability we can discard the divisor
- This is same for both the categories
- we can compare
  P(A very close game/sports) x P(sports)
  with
  P(A very close game/not sports) x P(not sports)

sentence	class
A great game	sports
election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Naive Bayes Classifier -Text Classification

- The probabilities can be calculated:
  1. count how many times the sentence
     'A very close game' appears in sports category
  2. Divide by the total
  3. obtain P(a very close game|sports)

- **PROBLEM:** we do not have the 'sentence' in the training set

=>probability is zero

Unless every sentence appears in the training set,what we want to classify, the model won't classify

sentence	class
A great game	sports
election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Naive Bayes Classifier -Text Classification

So

- We assume that every word in a sentence is independent of the other ones
- no longer we will look for entire sentences,but for only **individual words** i,e for a sentence “This was a funny party” is same as “funny is party was this” is same as “party funny this was a”
- We can write this as:
  P(a very close game)=P(a)xP(very)xP(close)xP(game)
- This enables to make the model work well
- Now let’s apply,

P(a very close game/sports)=
P(a/sports)xP(very/sports)xP(close/sports)xP(game/sports) x P(sports)

- Now as all these individual words actually show up several times in our training set,we can do our calculations

sentence	class
A great game	sports
election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Naive Bayes Classifier -Text Classification

CALCULATING PROBABILITIES

- the final step is just to calculate every probability and see which one turns to be larger
- **First:** calculate a **priori probability** for each category,i.e for the sentence given in the training set
  (In yellow) P(sports)=3/5=0.6
  (In purple) P(not sports)=2/5=0.4
- **calculate P(game/sports)** : counting number of times the word game appears in the sports sample,divided by the total no of words in sports i.e it appears twice for 11 words**. -note given the class is sports**

P(game/sports)=2/11=0.18181

A problem again- the word close does not appear in any sports ,and would lead us 0 when multiplied with other probability

What do we do?

sentence	class
A great **game**	sports
election is over	Not sports
very clean match	sports
a clean but forgettable **game**	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Naive Bayes Classifier -Text Classification

To resolve this we do something called **Laplace smoothing**

- i.e we add 1 (or any constant value) to every count so it's never zero
- to again balance this,we add the no of possible words to divisor,
- in our case the possible words are:
  {a,great,game,the election,is,over,.....election}=14
- Applying smoothing we get

WORD	P(word/sports)	P(word/Not sports)
a	(2+1)/(14+11)=3/25	(1+1)/(9+14)=2/23
very	(1+1)/(14+11)=2/25	(0+1)/(9+14)=1/23
close	(0+1)/(14+11)=1/25	(1+1)/(9+14)=2/23
game	(2+1)/(14+11)=3/25	(0+1)/(9+14)=1/23

sentence	class
A great game	sports
election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

MACHINE INTELLIGENCE

Naive Bayes Classifier -Text Classification

WORD	P(word/sports)	P(word/Not sports)
a	(2+1)/(14+11)=3/25	(1+1)/(9+14)=2/23
very	(1+1)/(14+11)=2/25	(0+1)/(9+14)=1/23
close	(0+1)/(14+11)=1/25	(1+1)/(9+14)=2/23
game	(2+1)/(14+11)=3/25	(0+1)/(9+14)=1/23

sentence	class
A great game	sports
election is over	Not sports
very clean match	sports
a clean but forgettable game	sports
it was a close election	not sports

Now we multiply all the probabilities to see which is bigger

P(a/sports)xP(very/sports)xP(close/sports)xP(game/sports)=0.000027648

P(a/not sports)xP(very/not sports)xP(close/not sports)xP(game/not sports)=$5.717532x10^{-6}$

We classify it as **"sports category"** as **0.000027648 > $5.717532x10^{-6}$**

Advanced Techniques

- **Removing stop words:** Stop words usually refers to the most common words in a language. example: a,able,the

  a very close game =>>>very close game

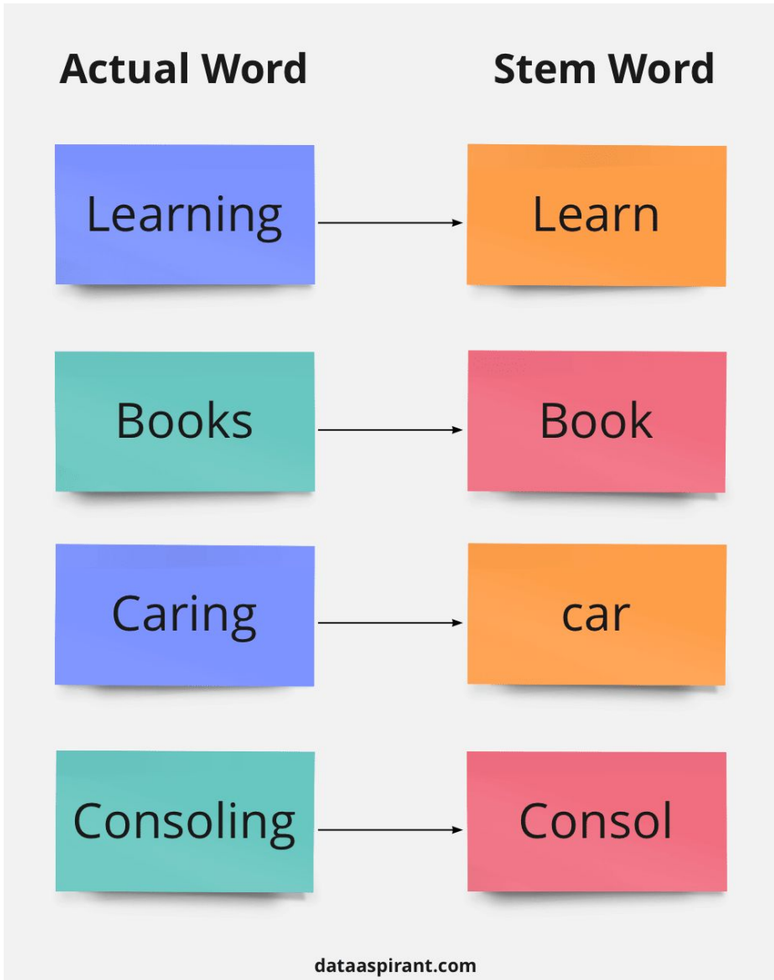
- **Stemming**: Words like election/elected are grouped together and counted as one word

- **Using n-grams:**
  (https://towardsdatascience.com/understanding-word-n-grams-and-n-gram-probability-in-natural-language-processing-9d9eef0fa058))

  Instead of counting individual words we can count sequence of words example:'clean match','close election'

- **TF-IDF (https://www.onely.com/blog/what-is-tf-idf/)**

  Term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.



TF-IDF (https://www.onely.com/blog/what-is-tf-idf/)

it’s a score to highlight each word’s relevance in the entire document. It’s calculated as -

IDF =Log[(# Number of documents) / (Number of documents containing the word)] and

TF = (Number of repetitions of word in a document) / (# of words in a document)

MACHINE INTELLIGENCE

Bayes Optimal Classifier and Gibbs Algorithm

Preet Kanwal

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Bayes Optimal Classifier

You are a team of 3 and you are working on building a model for sentiment classification of instagram comments into two categories - Positive and negative.

Your team decides to use Neural networks for classification and choose to apply softmax activation function at the output layer neurons.

You and your friends decided to implement the model separately and hence each one of you ended up getting a different accuracy on test data. (Your team has 3 models now⇒ 3 hypothesis)

In order to classify a new comment, you decided to use result of all 3 models.

Hypothesis (3 models)	Model Accuracy	Assign probabilities to hyp - using normalization
h1	90%	90/ (90+80+60) = 0.39
h2	80%	80/ (90+80+60) = 0.34
h3	60%	60/ (90+80+60) = 0.26

Bayes Optimal Classifier

Your output layer has 2 neurons.

On a new query instance, Each of the hypothesis(Model) outputs the following values on each neuron :

Hypothesis (3 models)	Assign probabilities to hyp - using normalization P(h | D)	Predicted value at Output layer Neuron 1 P(+|h)	Predicted value at Output layer Neuron 2 P(-|h)
h1	0.39	0.8	0.2
h2	0.34	0.4	0.6
h3	0.26	0.2	0.8

MACHINE INTELLIGENCE

Bayes Optimal Classifier

Using Bayes Optimal Classifier :

The most probable classification of new instance is obtained by combined the prediction of all hypothesis,weighted by their posterior probabilities.

Hypothesis (3 models)	Assign probabilities to hyp - using normalization P(h | D)	Predicted value at Output layer Neuron 1 P(+|h)	Predicted value at Output layer Neuron 2 P(-|h)
h1	0.39	0.8	0.2
h2	0.34	0.4	0.6
h3	0.26	0.2	0.8

MACHINE INTELLIGENCE

Bayes Optimal Classifier

Number of target concepts = 2;

$$P(+|D) = P(+|h1) * P(h1|D) + P(+|h2) * P(h2|D) + P(+|h3) * P(h3|D)$$

$$= 0.8 \times 0.39 + 0.4 \times 0.34 + 0.2 \times 0.26 = 0.5$$

$$P(-|D) = P(-|h1) * P(h1|D) + P(-|h2) * P(h2|D) + P(-|h3) * P(h3|D)$$

$$= 0.2 \times 0.39 + 0.6 \times 0.34 + 0.8 \times 0.26 = 0.49$$

Bayes Optimal Classifier decision: The new instance belongs to the class v2 !

Hypothesis (3 models)	Assign probabilities to hyp - using normalization P(h | D)	Predicted value at Output layer Neuron 1 P(+|h)	Predicted value at Output layer Neuron 2 P(-|h)
h1	0.39	0.8	0.2
h2	0.34	0.4	0.6
h3	0.26	0.2	0.8

Bayes Optimal Classifier --- Example 2

Hypothesis	Probability of hypothesis P(hi|D)	new instance (x) classified by hi as	Probability of (+| hi)	P(-|h1)
h1	0.4	+	1	0
h2	0.3	-	0	1
h3	0.3	-	0	1

$$\sum_{h_i \in H} P(+ve/h_i).P(h_i/D) = 1x0.4 + 0x0.3 + 0x0.3 = 0.4$$

$$\sum_{h_i \in H} P(-ve/h_i).P(h_i/D) = 0x0.4 + 1x0.3 + 1x0.3 = 0.6$$

Bayes Optimal Classifier's decision: The new instance belongs to -ve class

MACHINE INTELLIGENCE

Bayes Optimal Classifier – Example 3

Hypothesis	Probability of hypothesis P(hi|D)	Probability of (v_j| h_i) Of a new instance
h1	0.4	$[0.5.\ 0.3,\ 0.2]^T$
h2	0.3	$[0.1.\ 0.4,\ 0.5]^T$
h3	0.2	$[0.5.\ 0.4,\ 0.1]^T$
h4	0.1	$[0.1.\ 0.3,\ 0.6]^T$

Number of target concepts = 3; v_1, v_2, v_3

$p(v_j)$, j = 1, 2, 3:

0.5 x 0.4 + 0.1 x 0.3 + 0.5 x 0.2 + 0.1 x 0.1 = 0. 34

0.3 x 0.4 + 0.4 x 0.3 + 0.4 x 0.2 + 0.3 x 0.1 = 0. 35

0.2 x 0.4 + 0.5 x 0.3 + 0.1 x 0.2 + 0.6 x 0.1 = 0.31

Bayes Optimal Classifier's decision: The new instance belongs to the class v_2 !

- No hypothesis gives first preference to v_2 ! Yet, that is the class label returned by the Optimal Classifier!
- Suppose the probabilities returned by h_4 are changed to $[0.2.\ 0.3,\ 0.5]^T$. Now there will be a tie between v_1 and v_2 !

MACHINE INTELLIGENCE

Bayes Optimal Classifier

- Bayes Optimal Classification: For a class vj that belongs to set of all classes V
- The algorithm will output the class for which summation over all the hypothesis in hypothesis space, for which product of P(Vj| h). P(hi | D) is maximum.

$$\underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} P(V_j \mid h_j) * P(h_i \mid D)$$

This is called the **Bayes Optimal Classifier**: It outputs that class for a classification problem based on the formula described above

MACHINE INTELLIGENCE

Bayes Optimal Classifier

But Bayes Optimal Classifier is impractical.

Why impractical?

- For a test instance we have to apply all the possible hypothesis on the test instance on the test instance to arrive at the bayes optimal classification.
- As size of hypothesis space is huge, it is not possible to apply Bayes Optimal Classifier

Why is it called Optimal?

No other classifier can outperform using the same hypothesis space and the same prior knowledge can outperform this on an average

To make it possible we use **Gibbs sampling**

In which instead of applying all possible on X we sample the hypothesis space

MACHINE INTELLIGENCE

Gibbs Algorithm

- **Bayes optimal Classifier** obtains the best performance that can be achieved from the training data, it is **quite costly to apply**.
- The expense is due to the fact that it computes **the posterior probability for every hypothesis in H and combines the prediction** of each hypothesis to classify new instance
- An alternative less optimal method is the **"GIBBS ALGORITHM"** defined as follows

1. Choose a hypothesis h from H at **random** according to P(h|D) posterior probability distribution of over H.
    - So for each hypothesis we have a probability associated it with it. So we get a probability distribution over the hypothesis space
2. **use one hypothesis from the distribution** to predict the classification of the new instance x

Note: surprisingly, it can be shown that under certain conditions the expected misclassification error for GIBBS ALGORITHM is **at most (less than or equal to)** twice the expected error of Bayes Optimal Classifier

Error by Gibbs <= 2(Error by Bayes Optimal Classifier)

MACHINE INTELLIGENCE

Maximum Likelihood Hypothesis & Least squared error

Preet Kanwal

Department of Computer Science and Engineering

MACHINE INTELLIGENCE

Maximum Likelihood Hypothesis and Least Squared Error

- Let us consider a continuous valued target function.
- Many learning approaches such as Neural network,Linear Regression, and Polynomial Curve fitting try to learn a continuous valued target function
- A straight forward Bayesian analysis will show that under certain assumptions any learning algorithm that minimizes squared error between output hypothesis prediction and the training data will output **maximum likelihood hypothesis (H_{ML})**

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

- we have already witnessed this???
- In neural network and other curve fitting methods the attempt to minimize the sum of the squared error over the training data- Bayesian justification

MACHINE INTELLIGENCE

Maximum Likelihood and Least Squared Error Hypothesis

In order to find the maximum likelihood in Bayesian learning for continuous valued target function, we start with maximum likelihood hypothesis definition but using p to be the probability density function

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}}\, p(D|h)$$

We can consider the data D to be corresponding sequence of target values D=(d1,....dm)
Here we can write p(D|h) as the product of the various p(di|h)

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} p(d_i|h)$$

We assume that target variable is normal distributed

Maximum Likelihood and Least Squared Error Hypothesis

Given noise e_i obeys Normal Distribution,then d_i obeys Normal Distribution

- According to Normal Distribution
- where $\mu=h(xi)$ = output of hypothesis for ith input in this case
- di= target of ith input

$$= \underset{h\in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\prod\sigma^2}} e^{\frac{-1}{2\sigma^2}(d_i-\mu)^2}$$

$$= \underset{h\in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i-h(x_i)}{\sigma}\right)^2}$$

Instead of maximising this expression lets choose to maximise the logarithm.
Because ln(p) is a monotonic function of p. So by maximising ln p we also maximise p

Maximum Likelihood and Least Squared Error Hypothesis

- we now apply a transformation i.e common in max likelihood calculations
- Rather than maximizing this as complicated expression we shall choose to maximize its (loss complicated) logarithm.

$$= \underset{h \in H}{\operatorname{argmax}} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}$$

- The first term in this expression is a constant ,independent of **h** and can therefore be discarded yielding.

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

Maximum Likelihood and Least Squared Error Hypothesis

- maximizing negative quantity is same as minimizing the positive quantity ,yielding us

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i - h(x_i))^2 \rightarrow h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{1}{2\sigma^2}(d_i - h(x_i))^2$$

- finally we can discard the constants ,independent of h

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

Maximum Likelihood and Least Squared Error Hypothesis

From this final expression that we arrived at in the previous slides the insights that we can arrive at are:

- **Maximum Likelihood Hypothesis is equivalent to the the least squared error hypothesis**

  - di is the target
  - h(xi) is the output of hypothesis h

So to find the error we find the output of hypothesis from the target (di- h(xi))

$$h_{ML} = \underset{h \in H}{\operatorname{argmin}} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

Square of the error

argmin is to pick the LEAST squared error

References

1. “Machine Learning”, Tom Mitchell, McGraw Hill Education (India), 2013.
2. For more examples and further reference on probability concepts: https://www.analyticsvidhya.com/blog/2017/03/conditional-probability-bayes-theorem/
3. Eddie Woo: Probability and Discrete Probability Distributions
4. Probability and likelihood: https://www.youtube.com/watch?v=pYxNSUDSFH4
5. “Machine Learning”, Tom Mitchell, McGraw Hill Education (India), 2013.
6. https://web.cs.hacettepe.edu.tr/~ilyas/Courses/BIL712/lec04-BayesianLearning.pdf
7. https://www.youtube.com/watch?v=O2L2Uv9pdDA&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=39
8. https://www.youtube.com/watch?v=H3EjCKtlVog&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=40
9. https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn

THANK YOU

Preet Kanwal

Department of Computer Science & Engineering

preetkanwal@pes.edu