Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

5. Bayesian Learning

L. Iocchi, F. Patrizi

Bayesian learning methods are relevant to our study of machine learning for two different reasons. First, Bayesian learning algorithms that calculate explicit probabilities for hypotheses, such as the naive Bayes classifier, are among the most practical approaches to certain types of learning problems. The second reason that Bayesian methods are important to our study of machine learning is that they provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities.

Outline

- Bayes Theorem
- MAP, ML hypotheses
- MAP learners
- Bayes optimal classifier
- Naive Bayes learner
- Example: Learning over text data

References

T. Mitchell. Machine Learning. Chapter 6

summary

Two Roles for Bayesian Methods

Bayesian methods can accommodate hypotheses that make probabilistic predictions. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities

Provides practical learning algorithms:

- Naive Bayes learning (examples affect prob. that a hypothesis is correct)
  Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.
- Combine prior knowledge (prior probabilities) with observed data
  Prior knowledge can be combined with observed data to determine the final probability of a hypothesis
- Make probabilistic predictions (new instances classified by weighted combination of multiple hypotheses)
- Requires prior probabilities (often estimated from available data)

Provides useful conceptual framework

Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

- Provides "gold standard" for evaluating other learning algorithms

One practical difficulty in applying Bayesian methods is that they typically require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions. A second practical difficulty is the significant computational cost required to determine the Bayes optimal hypothesis in the general case summary

Basic Formulas for Probabilities

- *Product Rule*: probability of conjunction of A and B:

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

- *Sum Rule*: probability of disjunction of A and B:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- *Theorem of total probability*: if events $A_1, \ldots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

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

In machine learning we are often interested in determining the best hypothesis from some space H, given the observed training data D. One way to specify what we mean by the best hypothesis is to say that we demand the most probable hypothesis, given the data D plus any initial knowledge about the prior probabilities of the various hypotheses in H. Bayes theorem provides a direct method for calculating such probabilities.

- *Bayes theorem*:

SEE ALSO BLOCK 4

Bayes theorem is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability P(h|D), from the prior probability P(h), together with P(D) and P(D|h)

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

Classification as Probabilistic estimation

usually we want to learn a function with a dataset with the goal of classifing a new instance

Given target function $f : X \to V$, dataset D and a new instance x', best prediction $\hat{f}(x') = v^*$

optimal prediction

$$v^* = \operatorname*{argmax}_{v \in V} P(v|x', D)$$

More general formulation: given D and x', compute the probability distribution over V

given the dtaset D and a new sample x' (condition properties->we know D and x'), we want to compute the probability of V (it is the prediction of the model on x')

$$\boxed{P(V|x', D)}$$

probability distribution that models our problem. we want to maximize this prob

Learning as Probabilistic estimation

Given dataset D and hypothesis space H, compute a probability distribution over H given D.

$$P(H|D)$$

D is known while H must be computed

Bayes rule

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

- $P(h)$ = prior probability of hypothesis h
- $P(D)$ = prior probability of training data D
- $P(h|D)$ = probability of h given D
- $P(D|h)$ = probability of D given h

MAP Hypotheses

In many learning scenarios, the learner considers some set of candidate hypotheses H and is interested in finding the most probable hypothesis h ∈ H given the observed data D

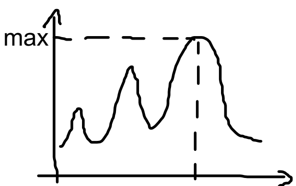
Any such maximally probable hypothesis is called a maximum a posteriori (MAP) hypothesis. We can determine the MAP hypotheses by using Bayes theorem to calculate the posterior probability of each candidate hypothesis.

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

Generally we want the most probable hypothesis h given D

Maximum a posteriori hypothesis h_{MAP}:

max over the all possible hypothesis



$$h_{MAP} \overset{def}{\equiv} \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} \frac{P(D|h)P(h)}{\cancel{P(D)}}$$

$$= \arg\max_{h \in H} \cancel{P(D|h)}P(h)$$

operator that is invariant with respect to a scaling of a constant and 1/P(D) is constant with respect to h, so I can remove P(D)

ML Hypotheses

MAXIMUM LIKELIHOOD

P(D|h) is often called the likelihood of the data D given h, and any hypothesis that maximizes P(D|h) is called a maximum likelihood (ML) hypothesis, h_ML

We don't have info about priors of hp. So we assume that the prior prob of the all hp is the same. so the hp space is uniformly distributed

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

If assume $P(h_i) = P(h_j)$, we can further simplify, and choose the *Maximum likelihood* (ML) hypothesis

$$h_{ML} = \arg\max_{h \in H} P(D|h)$$

Brute Force MAP Hypothesis Learner

We can design a straightforward concept learning algorithm to output the maximum a posteriori hypothesis, based on Bayes theorem

(H as the space of candidate target functions)

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

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

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

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

we can implement this algo when is feasible to enumerate all the hps

This algorithmmay require significant computation, because it applies Bayes theorem to each hypothesisin H to calculate P(h|D) .While this may prove impractical for large hypothesis spaces, the algorithm is still of interest because it provides a standard against which we may judge the performance of other concept Learning algorithms.

Most Probable Classification of New Instances

h_{MAP}: most probable *hypothesis* given data D.

Given a new instance x', what is its most probable *classification* of x'?

$h_{MAP}(x')$ may not be the most probable classification !!!

Most Probable Classification of New Instances

Consider: lets make this example

- Three possible hypotheses h_1, h_2, h_3:

  $P(h_1|D) = 0.4, \; P(h_2|D) = 0.3, \; P(h_3|D) = 0.3$

- Given a new instance x, h1 predicts + as value of x, h2 and h3 predict - as value for x

  $h_1(x) = \oplus, \; h_2(x) = \ominus, \; h_3(x) = \ominus$

  +: 0.4 -: 0.3+0.3=0.6 - is higher than +

- What is the most probable classification of x ?

Bayes Optimal Classifier

Consider target function $f : X \mapsto V$, $V = \{\overset{\text{set of classes}}{v_1, \ldots, v_k}\}$, data set D and a new instance $x \notin D$:

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

total probability over H

$P(v_j|x, h_i)$: probability that $h_i(x) = v_j$ is independent from D given h_i

$\Rightarrow P(v_j|x, h_i) = P(v_j|x, h_i, D)$

h_i does not depend on $x \notin D \Rightarrow P(h_i|x, D) = P(h_i|D)$

Bayes Optimal Classifier

this method is good but it is not applicable because we can't iterate over the all hps, so we need other methods

in general the space of hps is huge (so the method is not applicable) and continuous

Bayes Optimal Classifier

Class of a new instance x:

$$v_{OB} = \arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|x, h_i)P(h_i|D)$$

"what is the most probable classification of the new instance given the training data?"

Bayes Optimal Classifier

it means that optimally solves the problem that we have stated at the beginning

Example:
consider a hypothesis space containing three hypotheses, h1, h2, and h3.

posterior probability

because for h1, x is positive

$$P(h_1|D) = 0.4, \quad P(\ominus|x, h_1) = 0, \quad P(\oplus|x, h_1) = 1$$
$$P(h_2|D) = 0.3, \quad P(\ominus|x, h_2) = 1, \quad P(\oplus|x, h_2) = 0$$
$$P(h_3|D) = 0.3, \quad P(\ominus|x, h_3) = 1, \quad P(\oplus|x, h_3) = 0$$

h1 is the MAP hypothesis.

therefore

Suppose a new instance x is encountered, which is classified positive by h1, but negative by h2 and h3.

$$\sum_{h_i \in H} P(\oplus|x, h_i)P(h_i|D) = 0.4$$
$$\sum_{h_i \in H} P(\ominus|x, h_i)P(h_i|D) = 0.6$$

The most probable classification (negative) in this case is different from the classification generated by the MAP hypothesis.

and the most probable classification of the new instance is obtained by combining the predictions of all hypotheses, weighted by their posterior probabilities

$$v_{OB} = \arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|x, h_i)P(h_i|D) = \ominus$$

Bayes Optimal Classifier

Optimal learner: no other classification method using the same hypothesis space and same prior knowledge can outperform this method on average.

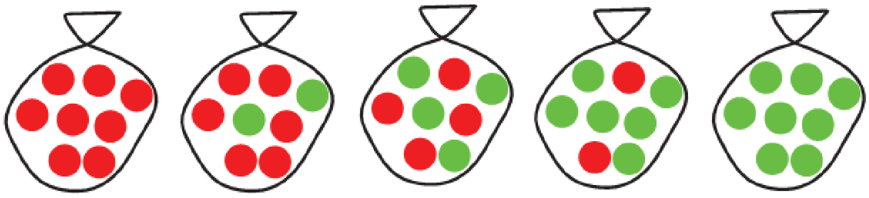
It maximizes the probability that the new instance x is classified correctly, i.e., $\mathrm{argmax}_{v_j \in V} P(v_j|x, D)$.

Very powerful: labelling new instances x with $\mathrm{argmax}_{v_j \in V} P(v_j|x, D)$ can correspond to none of the hypotheses in H.

Bayesian Learning Example

Five kinds of bags of candiers:

1. 10% are h_1: 100% cherry
2. 20% are h_2: 75% cherry, 25% lime
3. 40% are h_3: 50% cherry, 50% lime
4. 20% are h_4: 25% cherry, 75% lime
5. 10% are h_5: 100% lime



Bayesian Learning Example

We choose a random bag (not knowing which type it is) and extract some candies from it.

What kind of bag is it? What is the probability of extracting a candy of a specific flavor next?

Bayesian Learning Example

Prior probability distribution:

$$\boxed{\mathrm{P}(H) = < 0.1, 0.2, 0.4, 0.2, 0.1 >}$$

Likelihood for lime candy:

lime

$$\mathrm{P}(l|H) = < 0, 0.25, 0.5, 0.75, 1 >$$

Probability of extracting a lime candy (without data set):

P(l)

$$\sum_{h_i} \boxed{P(l|h_i)P(h_i)} = 0 \cdot 0.1 + 0.25 \cdot 0.2 + 0.5 \cdot 0.4 + 0.75 \cdot 0.2 + 1 \cdot 0.1 = 0.5$$

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Bayesian Learning Example

1. First candy is lime: $D_1 = \{l\}$

$P(h_i|\{d_1\}) = \alpha P(\{d_1\}|h_i)P(h_i)$ (Bayes rule)

$$\begin{aligned}
\mathrm{P}(H|D_1) &= \alpha < 0, 0.25, 0.5, 0.75, 1 > \cdot < 0.1, 0.2, 0.4, 0.2, 0.1 > \\
&= \alpha < 0, 0.05, 0.2, 0.15, 0.1 > \\
&= < 0, 0.1, 0.4, 0.3, 0.2 >
\end{aligned}$$

Bayesian Learning Example

2. Second candy is lime: $D_2 = \{l, l\}$

$P(h_i|\{d_1, d_2\}) = \alpha P(\{d_1, d_2\}|h_i)P(h_i)$ (Bayes rule)

after the bayes rule we can apply the independence

$= \alpha P(\{d_2\}|h_i) \underbrace{P(\{d_1\}|h_i)P(h_i)}_{\text{equal to the previous step}}$ (independent data samples)

$$
\begin{aligned}
\mathrm{P}(H|D_2) &= \alpha < 0, 0.25, 0.5, 0.75, 1 > \cdot \underbrace{\boxed{< 0, 0.1, 0.4, 0.3, 0.2 >}}_{}^{P(H|D_1)} \\
&= \alpha < 0, 0.025, 0.2, \underline{0.225}, 0.2 > \\
&= < 0, 0.038, 0.308, 0.346, 0.308 >
\end{aligned}
$$

argmax is the same because it doesn't depends on alpha

Bayesian Learning Example

3. Third candy is lime: $D_3 = \{l, l, l\}$

$P(h_i|\{d_1, d_2, d_3\}) = \alpha P(\{d_1, d_2, d_3\}|h_i)P(h_i)$ (Bayes rule)

$= \alpha \underline{P(\{d_3\}|h_i)}\ \underline{P(\{d_2\}|h_i)\ P(\{d_1\}|h_i)P(h_i)}$ (independent data samples)

$$
\begin{aligned}
\mathrm{P}(H|D_3) &= \alpha < \underline{0, 0.25, 0.5, 0.75, 1} > \cdot < \underline{0, 0.038, 0.308, 0.346, 0.308} > \\
&= \alpha < 0, 0.01, 0.154, 0.260, 0.308 > \\
&= < 0, 0.013, 0.211, 0.355, 0.421 >
\end{aligned}
$$

Bayesian Learning Example

multiset is a set with repetition

What is probability of having another lime candy after $\boxed{D_3 = \{l, l, l\}}$?

to compute this we can use H map approach

vote of each hp for lime

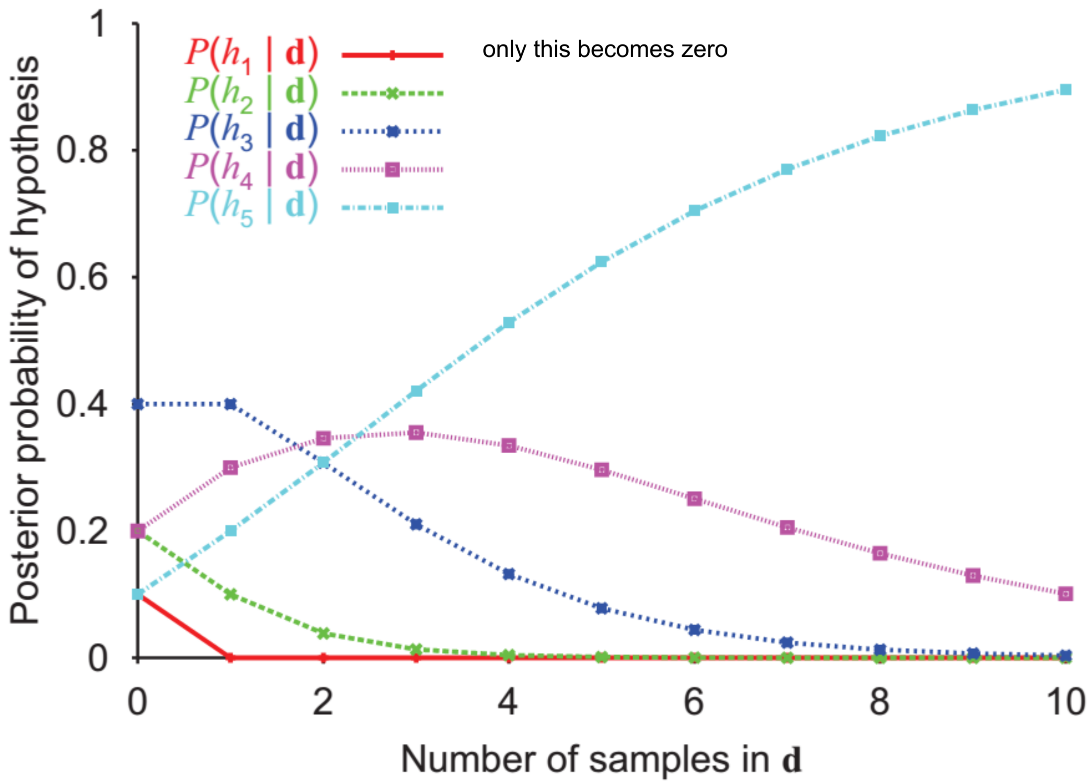
weights of each hp (how likelihood is this hp)

$$
\begin{aligned}
P(l|D_3) &= \sum_{h_i} \boxed{P(l|h_i)} \boxed{P(hi|D_3)} \\
&= 0 \cdot 0 + 0.25 \cdot 0.013 + 0.5 \cdot 0.211 + 0.75 \cdot 0.355 + 1 \cdot 0.421 \\
&= 0.8
\end{aligned}
$$

optimal prediction because we are considering the contribution of all phs

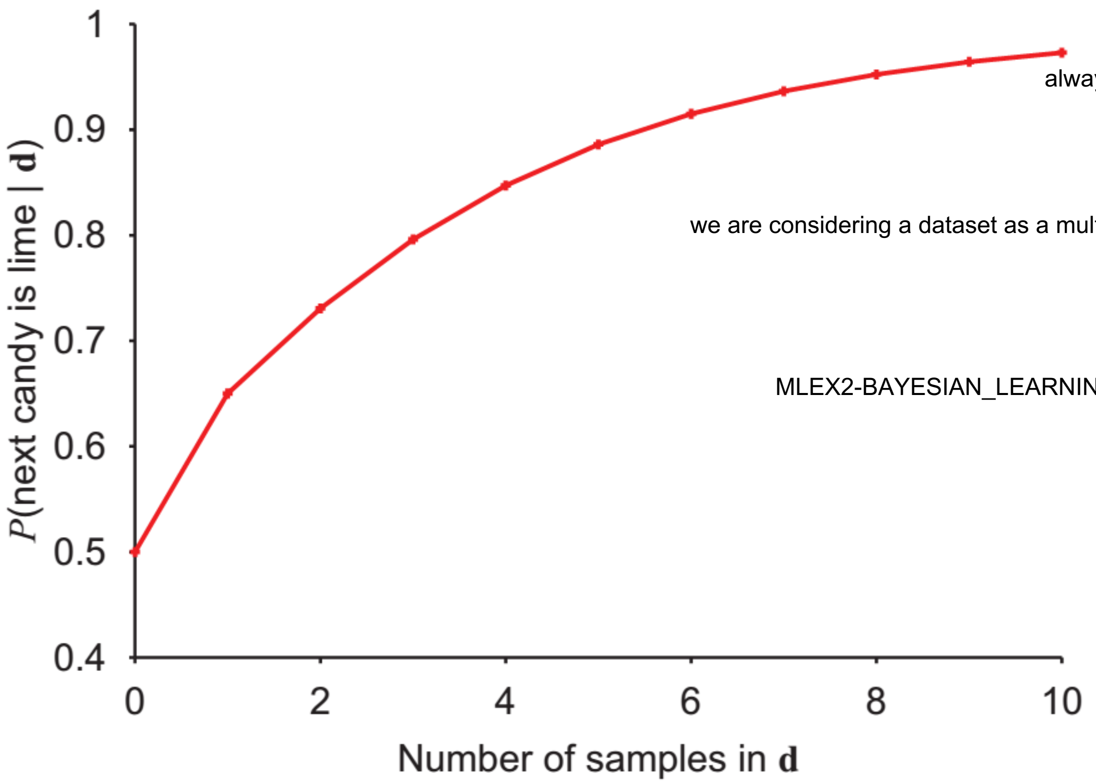
Bayesian Learning Example

plot the result of evolution



Bayesian Learning Example

prob of extracting a lime candy after different numbers of extraction



always less than 1

we are considering a dataset as a multiset

MLEX2-BAYESIAN_LEARNING

Bayesian Learning Example 2

this ex is an extension

Consider a new manufacturer producing bags with an arbitrary choice of cherry/lime candies. $\boxed{\theta \equiv \frac{nr.\,of\;cherry\;candies}{N} \in [0,1]}$.
Continuous space for hypotheses: h_θ hp with a particular value of theta. if theta is between 0 and 1 we have a continuous space for hp

Data set: $D = \{c \text{ cherries}, l \text{ lime}\}$, $N = c + l$

size of dataset (tot num of candies)

$$\boxed{\begin{array}{l} P(c|h_\theta) = \theta \\ P(l|h_\theta) = 1 - \theta \end{array}}$$

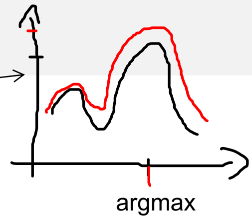
we assume that we not have info about the prior of hp

- What is the ML hypothesis?

maximum likelihood

Bayesian Learning Example 2

when the function is monotonic the max changes but the argmax remain the same



$$h_{ML} = \underset{h_\theta}{\operatorname{argmax}} P(D|h_\theta) = \underset{h_\theta}{\operatorname{argmax}} L(D|h_\theta)$$

with $L(D|h_\theta) = \log P(D|h_\theta)$ log func is monotonic

$\in D$

$$P(D|h_\theta) = \prod_{j=1\ldots N} P(d_j|h_\theta) = \theta^c \cdot (1-\theta)^l$$

we apply the log function to this term

the fact that we can transform in sum the product makes the computation simpler

$$L(D|h_\theta) = c \log \theta + l \log(1-\theta)$$

$$\frac{dL(D|h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0 \Rightarrow \theta_{ML} = \frac{c}{c+l} = \frac{c}{N}$$

candies

dim of dataset

maximum likelihood hp

General approach

Given dataset $D = \{d_i\}$ with $d_i \in \{0, 1\}$,
assuming a probability distribution $P(d_i; \Theta)$ ← parametric prob distribution (parametrized by theta)

Maximum likelihood estimation

$$\Theta_{ML} = \underset{\Theta}{\operatorname{argmax}} \log P(d_i|\Theta)$$

particular parametric distribution

Example: for Bernoulli distribution $\boxed{P(X = k; \theta) = \theta^k \, (1-\theta)^{1-k}}$

$$\theta_{ML} = \ldots = \frac{|\{d_i = 1\}|}{|D|}$$

we can apply this method also to other distributions

Bernoulli distribution

Probability distribution of a binary random variable $X \in \{0,1\}$

$$P(X=1)=\theta \quad P(X=0)=1-\theta$$

(e.g., observing head after flipping a coin, extracting a lime candy, ...).

$$P(X=k;\theta)=\theta^k\,(1-\theta)^{1-k}$$

Multi-variate Bernoulli distribution

it's just a combination of a set of binary random variable

Joint probability distribution of a set of binary random variables $X_1, \ldots X_n$, each random variable following Bernoulli distribution

$$P(X_1=k_1,\ldots,X_n=k_n\,;\,\theta_1,\ldots,\theta_n)$$

$k_i \in \{0,1\}$

(e.g., observing head after flipping a coin **and** extracting a lime candy, ...).

Under the assumption that random variables X_i are mutually independent, Multi-variate Bernoulli distribution is the product of n Bernoulli distributions

we assume that the random variables are mutually independent each other, and so

$$P(X_1=k_1,\ldots;\theta_1,\ldots,\theta_n)=\prod_{i=1}^{n}P(X_i=k_i;\theta_i)=\prod_{i=1}^{n}\theta_i^{k_i}(1-\theta_i)^{1-k_i}$$

Binomial distribution

case with repetitions

Probability distribution of k outcomes from n Bernoulli trials

$$P(X = k; n, \theta) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

(e.g., flipping a coin n times and observing k heads, extracting k lime candies after n extractions, ...).

Multinomial distribution

Generalization of binomial distribution for discrete valued random variables with d possible outcomes.
Probability distribution of k_1 outcomes for X_1, ..., k_d outcomes for X_d, after n trials (with $\sum_{i=1\ldots d} k_i = n$)

each var is modeled as binomial distribution

$$P(X_1 = k_1, \ldots, X_d = k_d; n, \theta_1, \ldots, \theta_d) = \frac{n!}{k_1! \ldots k_n!} \theta_1^{k_1} \cdot \ldots \cdot \theta_d^{k_d}$$

(e.g., rolling a d-sided die n times and observing k times a particular value, extracting k lime candies after n extractions form a bag containing d different flavors, ...).

Remarks

Probabilistic classification

$$\operatorname*{argmax}_{v_j \in V} P(v_j | x, D)$$

- Bayes Optimal Classifier
  provides best result, not practical when hypothesis space is large

Continuous model

- Maximum likelihood estimation
  efficiently solved when analytical solutions are available

num of hp is small or there is a simple analytical solution

What are more practical and general solutions?

Naive Bayes Classifier

approximation of OBC using the conditional independence assumption

One highly practical Bayesian learning method is the naive Bayes learner, often called the naive Bayes classijier. In some domains its performance has been shown to be comparableto that of neural network and decision tree learning.

Naive Bayes Classifier uses conditional independence to approximate the solution.

X is conditionally independent of Y given Z

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

this is the same without Y

Naive Bayes Classifier

Assume target function $f : X \to V$, where each instance x is described by attributes $\langle a_1, a_2 \dots a_n \rangle$.

Compute

$$\boxed{\operatorname*{argmax}_{v_j \in V} P(v_j | x, D) = \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \dots a_n, D)}$$

without explicit representation of hypotheses.

Naive Bayes Classifier

we will manipulate the previous formula (SLIDE 35) to make it feasible

Given a data set D and a new instance $x = \langle a_1, a_2 \dots a_n \rangle$, most probable value of $f(x)$ is:

$$\begin{aligned} v_{MAP} &= \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \dots a_n, D) \\ &= \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j, D) P(v_j | D)}{\cancel{P(a_1, a_2 \dots a_n | D)}} \\ &= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \dots a_n | v_j, D) P(v_j | D) \end{aligned}$$

we apply Bayes rule

we remove this term that cannot depends on v. Since we are interested in argmax, this wll be just a normalization factor that we can ignore

(Bayes rule)

Naive Bayes Classifier

this is not real in general. We are approximating the solution and so it's not guaran teed anymore to get the optimal value

Naive Bayes assumption: all the attributes are mutually conditionally independent each other given the dataset and the classification value

$$\boxed{P(a_1, a_2, \ldots, a_n | v_j, D) = \prod_i P(a_i | v_j, D)}$$

Naive Bayes classifier

Class of new instance x:

$$\boxed{v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j | D) \prod_i P(a_i | v_j, D)}$$

target value output by the naive Bayes classifier

Naive Bayes Algorithm

Target function $f : X \mapsto V$, $X = A_1 \times \ldots \times A_n$, $V = \{v_1, ..., v_k\}$, data set D, new instance $x = \langle a_1, a_2 \ldots a_n \rangle$.

Naive_Bayes_Learn(A, V, D)

for each target value $v_j \in V$

$\boxed{\hat{P}(v_j | D)} \leftarrow$ estimate $P(v_j | D)$

for each attribute A_k

for each attribute value $a_i \in A_k$

$\boxed{\hat{P}(a_i | v_j, D)} \leftarrow$ estimate $P(a_i | v_j, D)$

Classify_New_Instance(x)

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} \hat{P}(v_j | D) \prod_{a_i \in x} \hat{P}(a_i | v_j, D)$$

Naive Bayes estimation

$$\hat{P}(v_j|D) = \frac{|\{< \ldots, v_j >\}|}{|D|}$$

$$\hat{P}(a_i|v_j, D) = \frac{|\{< \ldots, a_i, \ldots, v_j >\}|}{|\{< \ldots, v_j >\}|}$$

Note: if none of the training instances with target value v_j have attribute value a_i, then $\hat{P}(a_i|v_j, D) = 0$ and thus $\hat{P}(v_j|D) \prod_i \hat{P}(a_i|v_j, D) = 0$

there are methods to avoid that all the prediction are equal to zero, and this method makes all the prediction strictly > 0
(for ex. VIRTUAL EXAMPLES METHOD -> SLIDE 40)

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Naive Bayes estimation

***m*-estimate of probability:**

$$\frac{n_c + mp}{n + m}$$

Typical solution is Bayesian estimate with prior estimates

$$\hat{P}(a_i|v_j, D) = \frac{|\{< \ldots, a_i, \ldots, v_j >\}| + mp}{|\{< \ldots, v_j >\}| + m}$$

where

- p is a prior estimate for $P(a_i|v_j, D)$
- m is a weight given to prior (i.e. number of "virtual" examples)

Naive Bayes: Example

Consider *PlayTennis* again, and new instance

$$\langle Outlook = sun, Temp = cool, Humid = high, Wind = strong \rangle$$

We want to compute: we have a binary classification problem (v = Yes or No)

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j|D) \prod_i P(a_i|v_j, D)$$

without making any hypothesis space explicit.

$$v_{NB} = \operatorname*{argmax}_{v_j \in \{yes,no\}} P(v_j) \prod_i P(a_i|v_j)$$

$$= \operatorname*{argmax}_{v_j \in \{yes,no\}} P(v_j) \quad P(Outlook = sunny|v_j) P(Temperature = cool|v_j)$$

$$\cdot P(Humidity = high|v_j) P(Wind = strong|v_j) \qquad (6.21)$$

Naive Bayes: Example

Note: easy notation with conditioning on D omitted.

$$P(PlayTennis = yes) = P(y) = 9/14 = 0.64$$
$$P(PlayTennis = no) = P(n) = 5/14 = 0.36$$

$$P(Wind = strong|y) = 3/9 = 0.33$$
$$P(Wind = strong|n) = 3/5 = 0.60$$

$$\ldots$$

$$P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ P(strong|y) = .005$$
$$P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ P(strong|n) = .021$$

Thus, the naive Bayes classifier assigns the target value PlayTennis = no to this new instance, based on the probability estimates learned from the training data

$$\rightarrow v_{NB} = n$$

we can solve this by using a prediction tree

Naive Bayes Remarks

Conditional independence assumption is often violated

$$P(a_1, a_2 \ldots a_n | v_j, D) \cancel{\approx} \prod_i P(a_i | v_j, D)$$

...but it works surprisingly well anyway.

Note: don't need estimated posteriors $\hat{P}(v_j | x, D)$ to be correct; need only that

it is true even if this term is very different from this other

$$\boxed{\underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j | D) \prod_i \boxed{\hat{P}(a_i | v_j, D)} \cancel{\approx} \underset{v_j \in V}{\operatorname{argmax}} P(v_j | D) \boxed{P(a_1 \ldots, a_n | v_j, D)}}$$

Issue: Naive Bayes posteriors often unrealistically close to 1 or 0

Learning to classify text

for ex spam classification

Input: set of documents (sequences of words)

Learn target function $f : Docs \mapsto \{c1, \ldots, c_k\}$

Examples:

- spam classification (e-mail, SMS, ...)
- sentiment analysis (facebook/twitter posts, web reviews, ...)
- ...

Bag of words representation

Vocabulary $V = \{w_k\}$: set of all the words appearing in any document of the data set.
$n = |V|$: size of the vocabulary

VECTORIZATION

Bag of words representation of a text: n-dimensional feature vector

Note: BoW representation looses information (order of words in a text is important!)

Bag of words representation

Two options for representing each feature:

1. boolean features: 1 if word appears in the text, 0 otherwise (multivariate Bernoulli distribution)
2. ordinal features: number of occurrences of the words in the text (multinomial distribution)

Learning to Classify Text: Naive Bayes approach

Classification of documents $Docs$ in classes C.

Target function $f : Docs \mapsto C$, $C = \{c_1, \dots, c_k\}$

Data set $D = \{< d_i, c_i >\}$

Given a new document d_i, compute

$$c_{NB} = \operatorname*{argmax}_{c_j \in C} P(c_j|D) \prod_i P(d_i|c_j, D)$$

Learning to Classify Text: Naive Bayes approach

we introduce other assumptions

Naive Bayes conditional independence assumption

we are interesting in a word appears not where

$$P(d_i|c_j, D) = \prod_{i=1}^{length(d_i)} P(a_i = w_k|c_j, D)$$

where $P(a_i = w_k|c_j)$ is probability that word in position i is w_k, given c_j

one more assumption: $P(a_i = w_k|v_j, D) = P(a_m = w_k|v_j, D), \forall i, m,$
thus consider only $P(w_k|c_j, D)$.

Multi-variate Bernoulli Naive Bayes distribution

Feature vector for document d: n-dimensional vector 1 if word w_k appears in document d, 0 otherwise

$$P(d|c_j, D) = \prod_{i=1}^{n} P(w_i|c_j, D)^{I(w_i \in d)} \cdot (1 - P(w_i|c_j, D))^{1 - I(w_i \in d)}$$

$I(w_i \in d) = 1$ if $w_i \in d$, 0 otherwise

$$\hat{P}(w_i|c_j, D) = \frac{t_{i,j} + 1}{t_j + 2}$$

$t_{i,j}$: number of documents in D of class c_j containing word w_i
t_j: number of documents in D of class c_j
1, 2: parameters for Laplace smoothing

Multinomial Naive Bayes distribution

Feature vector for document d: n-dimensional vector with number of occurrences of word w_i in document d

$$P(d|c_j, D) = Mu(d; n, \theta) = \dots$$

$$\hat{P}(w_i|c_j, D) = \frac{\sum_{d \in D} tf_{i,j} + \alpha}{\sum_{d \in D} tf_j + \alpha \cdot |V|}$$

$tf_{i,j}$: term frequency (number of occurrences) of word w_i in document d of class c_j
tf_j: all term frequencies of document d of class c_j
α: smoothing parameter ($\alpha = 1$ for Laplace smoothing)

Naive Bayes Text Classification algorithm

During learning, the procedure LEARN_NAIVE_BAYES_TEXT examines all training documents to extract the vocabulary of all words and tokens that appear in the text, then counts their frequencies among the different target classes to obtain the necessary probability estimates.

Estimate $\hat{P}(c_j)$ and $\hat{P}(w_i|c_j)$ using *Bernoulli distribution*.

LEARN_NAIVE_BAYES_TEXT_$\boxed{\text{BE}}$$(D, C)$

$\boxed{V} \leftarrow$ all distinct words in D we first build the vocabulaty

for each target value $c_j \in C$ do

$docs_j \leftarrow$ subset of D for which the target value is c_j

$t_j \leftarrow |docs_j|$: total number of documents in c_j

$\boxed{\hat{P}(c_j) \leftarrow \frac{t_j}{|D|}}$ we estimate the prob of each class

for each word w_i in V do

$t_{i,j} \leftarrow$ number of documents in c_j containing word w_i

$\boxed{\hat{P}(w_i|c_j) \leftarrow \frac{t_{i,j}+1}{t_j+2}}$ then we estimate the prob of each word given the class

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Naive Bayes Text Classification algorithm

Estimate $\hat{P}(c_j)$ and $\hat{P}(w_i|c_j)$ using *multinomial distribution*.

LEARN_NAIVE_BAYES_TEXT_$\boxed{\text{MU}}$$(D, C)$

$V \leftarrow$ all distinct words in D

for each target value $c_j \in C$ do

$docs_j \leftarrow$ subset of D for which the target value is c_j

$t_j \leftarrow |docs_j|$: total number of documents in c_j

$\boxed{\hat{P}(c_j) \leftarrow \frac{t_j}{|D|}}$

$TF_j \leftarrow$ total number of words in $docs_j$ (counting duplicates)

for each word w_i in V do

$TF_{i,j} \leftarrow$ total number of times word w_i occurs in $docs_j$

$\boxed{\hat{P}(w_i|c_j) \leftarrow \frac{TF_{i,j}+1}{TF_j+|V|}}$

Naive Bayes Text Classification algorithm

given a new document to be classified, the procedure CLASSIFY_NAIVE_BAYES_TEXT uses these probability estimates to calculate V_NB according to Equation:

$$\boxed{v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j|D) \prod_i P(a_i|v_j, D)}$$

Use estimated $\hat{P}(c_j)$ and $\hat{P}(w_i|c_j)$ to classify a new document.

$\textsc{Classify\_Naive\_Bayes\_text}(d)$ classification of a new document

remove from d all words not included in vocabulary V

return

$$\boxed{v_{NB} = \operatorname*{argmax}_{c_j \in C} \hat{P}(c_j) \prod_{i=1}^{length(d)} \hat{P}(w_i|c_j)}$$

N.B: Note that any words appearing in the new document that were not observed in the training set are simply ignored by CLASSIFY_NAIVE_BAYES_TEXT

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Text Classification improvements

NATURA LANGUAGE PROCESSING

- Stop words: remove from all the documents common words ("the", "a", etc.)
- Stemming: replace words with basic forms ("likes" $\rightarrow$ "like", "liking" $\rightarrow$ "like", etc.)
- Bi-gram, n-gram: token is a sequence of words
- ...

MLEX3-SPAM_CLASSIFICATION