5. Bayes Learning

The next step is how to apply the probabilistic concepts to ML.

The main transformation that we will use is the Bayes theorem:

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

What we want to compute is the probability distribution for each hypothesis, how likely is a particular hypothesis to generate the dataset. One hypothesis can be more likely than another.

Bayesian learning is a very practical way to apply and it is very easy, it is considered as a baseline for many ML problems

$P(D)$ = probability of extracting D from the entire distribution

$P(h)$ = prior probability of a single hypothesis, before we get only dataset

$P(h|D)$ = prob. that h has generated D

$P(D|h)$ = prob. that given h then I generate the dataset D.
(MUCH EASIER TO COMPUTE)

Generally we want the hypothesis that maximizes the posterior probability. THIS IS CALLED MAXIMUM A POSTERIORI HYPOTH., h_{MAP}

$$h_{MAP} = \underset{h \in H}{\arg\max} \frac{P(D|h)\,P(h)}{P(D)}$$

this can be removed since it doesn't depend on h

$$= \underset{h \in H}{\arg\max}\ P(D|h)\,P(h)$$

The argmax is an invariant wrt scale and monotonic transformation, namely if we scale a function by a constant factor the argmax does not change

In some cases you cannot define the prior probability of h, if $P(h)$ is constant, is a uniform distribution, that means $P(h_i) = P(h_j)$ we can simplify the formula by REMOVING $P(h)$

Maximum Likelihood $\quad h_{ML} = \underset{h \in H}{\arg\max}\ P(D|h)$

Let's consider the algorithm for computing MAP.

1. $\forall h \in H$ calculate $P(h|D)$
2. Take $\underset{h \in H}{\arg\max}\ P(h|D)$

This algorithm is impossible to compute, since H can be infinite, but the main point is that h_{MAP} is not enough, because h_{MAP} is just one possible hypothesis in the space.

The use of only h_{MAP} over a new instance is not enough.

Given a new instance x' and three hypotesis, such that:

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

so h_1 is a MAP. Then $h_1(x') = +$, $h_2(x') = -$
$h_3(x') = -$

If you consider only $h_{MAP} = h_1$, you classify x' as $+$, that is not the best, because the evidence / contribution for the $-$ class comes from two hypotheses.

We should consider the contribution of all possible hypothesis, by making a WEIGHTED AVG:

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

$f : X \to V$, $V = \{v_1, \dots, v_k\}$, $x \notin D$

Once you have h you don't need D anymore:

- $P(v_j \mid x, h_i, D) \Rightarrow P(v_j \mid x, h_i)$
- $P(h_i \mid x, D) \Rightarrow P(h_i \mid D)$ h_i does not depend on x

This is the best we can do, and it is called

$$\boxed{\text{Bayes Opt. Classifier} \quad \underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} \underbrace{P(v_j \mid x, h_i)}_{\text{the vote that } h_i \text{ gives to a class}} \underbrace{P(h_i \mid D)}_{\text{weight}}}$$

There are proofs which say that no other ML can give better result (*). Again this is not a practical method because we still have to repeat for all the hypothesis.

(*) Optimal Learner Concept (on the slides), no other classification method using the same H and same prior knowledge can outperform this method on average

The argmax is very powerful, because labelling new instances x with the argmax can correspond to none of the hypothesis in H.

Note: the logarithm is monotonic, the argmax is the same, because it is a monotonic transformation.

Sometimes it is useful to recognize that random phenomena that we study belong to a family of distributions. The problem of extracting the candy from a bag, tossing a coin, is an example of a Bernoully distribution.

BERNOULLY DISTRIBUTION models any phenomenon that wants to asses prob. distribution over random and boolean variable:

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

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

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

Given a dataset $D = \{x_i\}$, maximum likelihood estimation:

$$\theta_{ML} = |\{x_i = 1\}|$$

We have multivariate Bernoulli distribution when we repeat the experiment with different random variables (extracting a lime candy and observing head of a coin).

Joint probability distribution of a set of binary random variables $X_1, \dots, X_m$, each random variable following Bernoulli distribution.

$$P(X_1 = k_1, \dots, X_m = k_m; \theta_1, \dots, \theta_m)$$

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

UNDER THE ASSUMPTION THAT X_i (Random var.) ARE MUTUALLY INDEPENDENT, the multivariate distribution is the product of m Bernoulli distribution:

$$\boxed{\prod_{i=1}^{m} P(X_i = k_i; \theta_i)}$$

BINOMIAL DISTRIBUTION: Probability of k outcomes from m Bernoulli trials (flipping a coin m times and observing k heads, extracting k times lime candies after m extractions, ...)

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

MULTINOMIAL: Generalization of binomial distribution for discrete valued random variables with d possible outcomes. We have a set of discrete random variables and we want to compute the joint distribution of all these random variables.

Ex:

Rolling a d-sided dice m times and ~~[illegible]~~ observing k times a particular value and at same time extracting k times candies after m extractions ...

$$P(X_1 = k_1, \dots, X_d = k_d; m, \theta_1 \dots \theta_d) = \frac{m!}{k_1! \dots k_d!} \theta_1^{k_1} \dots \theta_d^{k_d}$$

NAIVE BAYES CLASSIFIER

In many cases the concept of assumption is important, because:

- SIMPLIFY the PROBLEM
- ALLOW TO FIND A SOLUTION

But ANY ASSUMPTION may be not always true and if applied the FINAL SOLUTION IS AN APPROXIMATION of the optimal one, this is the price to pay.

Naive Bayes classifier uses conditional independence to approximate the solution:

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

WHAT IS DIFFICULT IS TO COMPUTE THE JOINT PROBABILITIES.
Assume a target function $f: X \to V$, where each instance x is described by attributes $\langle a_1, \dots, a_M \rangle$, the goal is:

> Compute $\underset{v_J \in V}{\arg\max}\ P(v_J | x, D) = \underset{v_J \in V}{\arg\max}\ P(v_J | a_1, \dots, a_M, D)$
> without explicit representation of hypothesis.

Given D and a new instance $x = \langle a_1, \dots, a_M \rangle$, the most prob. value of $f(x)$ is:

$$\boxed{v_{MAP} = \underset{v_J \in V}{\arg\max}\ P(a_1, \dots, a_M | v_J, D) P(v_J | D)}$$

Bayes rule, discarding the denominator.

NAIVE BAYES ASSUMPTION: all attributes are independent each other

THIS IS A VERY STRONG ASSUMPTION THAT IS NEVER TRUE IN MANY CASES! With NB assumption:

$$\boxed{P(a_1, \dots, a_M | v_J, D) = \prod_i P(a_i | v_J, D)}$$

Naive Bayes Classifier!

$$\boxed{v_{NB} = \underset{v_J \in V}{\arg\max}\ P(v_J \mid D) \prod_i P(a_i \mid v_J, D)}$$

This can be easily computed, we need to estimate small numbers of probabilities. The number of parameters is limited!

With this formulation we can apply an algorithm:

$f: X \to V, \quad V = \{v_1, \dots, v_k\}$
$X = A_1 \times A_2 \dots$

FIRST PHASE of ESTIMATION

NBLearner (A, V, D)
- for each $v_J \in V$
  - $\hat{P}(v_J \mid D) \leftarrow$ estimate $P(v_J \mid D)$
  - for each attribute A_k
    - for each value $a_i \in A_k$
      - $\hat{P}(a_i \mid v_J, D) \leftarrow$ estimate $P(a_i \mid v_J, D)$

SECOND PHASE Classification

Classify_New_Instance (x)

$$v_{NB} = \underset{v_J \in V}{\arg\max}\ \hat{P}(v_J \mid D) \prod \hat{P}(a_i \mid v_J, D)$$

If the attributes are conditional independent, then the v_{NB} is the OPT, otherwise v_{NB} is NOT OPT in general but it is an approxim.

How we make the estimation? We may just consider the number of times the event happens divided the total number of times

$$\hat{P}(v_J \mid D) = \frac{|\{\langle \dots v_J \rangle\}| \to \text{\# samples with } v_J}{|D| \to \text{\# samples.}}$$

$$\hat{P}(a_i \mid v_J, D) = \frac{|\{\langle \dots a_i \dots v_J \rangle\}| \leftarrow \text{\# samples with } v_J \text{ and attribute } a_i}{|\{\langle \dots v_J \rangle\}|}$$

$\uparrow$ # samples with v_J.

Two problems!

- IF YOU CHANGE D, the probabilities change
- IF YOUR DATASET NOT CONTAINS SOME COMBINATION of a_i and v_j, $\hat{P}(a_i | v_j, D) = 0$

To solve the second issue, we add some virtual example, we sum a proportion mp of prediction that the particular sample will be in D.

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

- p is prior estimate for $P(a_i | v_j, D)$
- m is a weight given to prior.

THE IMPORTANT IS THAT $\hat{P}$ IS NOT ZERO.

$\boxed{\text{Learning to classify text}}$

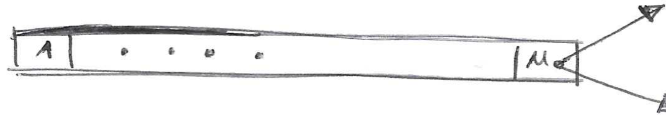
The input is a set of documents, where each document can be seen as a sequence of words, we have a variable lenght input.

I want to learn a target function $f: Docs \mapsto \{c_1, \ldots, c_k\}$

In order to have a good representation of the input we introduce the bag of words representation.

V = vocabulary = set of all words in the dataset;
m is the size of the vocabulary.

Each document in the bag of word representation is a vector with m-components, so it has the size of the vocabulary.

d: 

0, 1 $\begin{cases} 0 & \text{the } m\text{-th word is not present} \\ 1 & \text{otherwise} \end{cases}$

occurrencies of the m-th word.

WE LOSE THE INFORMATION ABOUT THE ORDER OF WORDS. Two different documents can be represent with the very same vector.
VERY STRONG APPROXIMATION!

BINARY FEATURE VECTOR $\longrightarrow$ generate a (Multivariate) Bernoulli distribution

ORDINAL FEATURE VECTOR $\longrightarrow$ generate a Multinomial distribution.

1. NB assumption all the words are independent each other given the class:

$$P(d_i \mid c_J, D) = \prod_{i=1}^{|d_i|} \underbrace{P(a_i = w_k \mid c_J, D)}_{\text{Prob. of having word } w_k \text{ in position } i \text{ given } c_J.}$$

2. The second assumption is that for each position the probability of a particular word to appear is the same:

$$P(a_i = w_k \mid v_J, D) = P(a_m = w_k \mid v_J, D)$$

$\forall\, i, m$, thus we consider only $P(w_k \mid v_J, D)$.

MULTIVARIATE - BERNOULLI DISTRIBUTION.

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

$$P(d \mid c_j, D) = \prod_{i=1}^{m} P(w_i \mid c_j, D)^{I(w_i \in d)} \left(1 - P(w_i \mid c_j, D)\right)^{1 - I(w_i \in d)}$$

$$I(w_i \in d) = \begin{cases} 1 & \text{if } w_i \in d \\ 0 & \text{otherwise} \end{cases}$$

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

LAPLACE SMOOTHING

- $t_{i,J}$ = # docs in class c_J containing w_i
- t_J = # docs in class c_J.

MULTINOMIAL NB DISTRIBUTION

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

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

$tf_{i,j}$ = term frequency of word w_i in the d document of class c_j

tf_j = $\sum$ all term frequencies of document d of class c_j.

α = smoothing parameter.

ALGORITHM FOR NB

$V \leftarrow$ distinct words in the set of docs. D

$\forall$ each $c_j \in C$

$docs_j \leftarrow \{ d \in D \mid \text{class of } d \text{ is } c_j \}$

$t_j \leftarrow |docs_j|$

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

MULTINOMIAL. ($TF_j \leftarrow$ total number of words in doc_j)

$\forall$ w_i in V do:

MULTINOMIAL NB

$TF_{i,j} \leftarrow$ total number of words w_i occurring in doc_j

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

BERNOULLI NB

$t_{i,j} \leftarrow$ # docs in c_j containing w_i

$$\hat{P}(w_i \mid c_j) \leftarrow \frac{t_{i,j} + 1}{t_j + 2}$$

CLASSIFY_NAIVE_BAYES_TEXT(d)

remove from d all words not in V

return

$$V_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \; \hat{P}(c_j) \prod_{i=1}^{|d|} \hat{P}(w_i | c_j)$$