Bayesian Learning - Naive Bayes and EM

Machine Learning, A.Y. 2022/23

Fabio Aiolli

December 5th, 2022

Naive Bayes classifier

One of the simplest and most popular techniques

When to use it:

- large enough data sets
- attributes describing the instances are conditionally independent given the classification

Applications on which it has been successful:

- Diagnosis
- Classification of textual documents

Naive Bayes classifier

Target function $f : X \to V$, with instances x described by attributes $\langle a_1, a_2, \ldots, a_n \rangle$.

The most probable classification of a new instance $f(x)$ is:

$$\begin{aligned} v_{MAP} &= \arg\max_{v_j \in V} P(v_j | a_1, a_2, \ldots, a_n) \\ &= \arg\max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n | v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)} \\ &= \arg\max_{v_j \in V} P(a_1, a_2, \ldots, a_n | v_j) P(v_j) \end{aligned}$$

It is impossible to correctly estimate all probabilities $P(a_1, a_2, \ldots, a_n | v_j)$!
Naive Bayes assumption:

$$P(a_1, a_2, \ldots, a_n | v_j) = \prod_i P(a_i | v_j)$$

Naive Bayes classifier

Putting it all together:

$$v_{MAP} = \arg\max_{v_j \in V} P(a_1, a_2, \ldots, a_n | v_j) P(v_j)$$

$$P(a_1, a_2, \ldots, a_n | v_j) = \prod_i P(a_i | v_j)$$

Naive Bayes classifier:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

How many parameters (probabilities) do we need to estimate? We need to estimate all $P(a_i|v_j)$. So, number of distinct attribute values multiplied by the number of classes.

Naive Bayes algorithm

Naive_Bayes_Learn(`Examples`)

- **For each** target value v_j
  - $\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$ on `Examples`
  - **For each** possible value of each a_i,
    $\hat{P}(a_i|v_j) \leftarrow$ estimate $P(a_i|v_j)$ on `Examples`
- **return** $\hat{P}(v_j), \hat{P}(a_i|v_j)\ \forall i, j$

Classify_New_Instance(x)

- $v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i(x)|v_j)$
- **return** v_{NB}

Shall we play tennis again?

Is it a good day for a tennis match?

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

Shall we play tennis again?

Let us consider again the "Playing Tennis" problem we encountered earlier

Consider a new instance:
<O = sunny, T = cool, H = high, W = strong>

We want to compute:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

$$\begin{aligned} P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes) &\approx 0.005 \\ P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no) &\approx 0.021 \\ &\rightarrow v_{NB} = no \end{aligned}$$

Naive Bayes: additional considerations

The assumption of conditional independence is often violated

$$P(a_1, a_2, \ldots, a_n | v_j) = \prod_i P(a_i | v_j)$$

- ... but it seems to work anyway. Why? Note that it is not necessary to correctly estimate the posterior probability $\hat{P}(v_j|x)$; it is sufficient that:

$$\arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

- the posterior probability calculated by Naive Bayes is often close to 1 or 0 even though it should not (bad estimate, but the order among probabilities is important here).

Naive Bayes: additional considerations

What if no learning examples with target value v_j has the $a_i = k$ attribute value? In that case:

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

A typical solution is the Bayesian *m-estimate* for $\hat{P}(a_i|v_j)$:

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

where

- n is the number of training examples where $v = v_j$
- n_c is the number of training examples where $v = v_j$ and $a = a_i$
- p is our prior estimate of $\hat{P}(a_i|v_j)$, usually $1/|a_i|$
- m is a constant (a.k.a. equivalent sample size) which determines how heavily to weight p (that is the number of "virtual" examples distributed according to p)

An example: learning to classify text

- learn which documents are of interest
- learn to classify web pages by topic
- spam / no spam
- ...

The Naive Bayes classifier had been for a long time one of the most used techniques in these contexts

What attributes to use to represent textual documents?

An example: learning to classify text

Target concept: *Interesting*? : *Document* $\rightarrow \{\oplus, \ominus\}$

- Represent each document by a vector of words. An attribute for each word position in the document
- Learning: use examples to estimate:
  $P(\oplus), P(\ominus), P(doc|\oplus), P(doc|\ominus)$

Assumption of conditional independence of Naive Bayes:

$$P(doc|v_j) = \prod_{i}^{length(doc)} P(a_i = w_k|v_j)$$

where $P(a_i = w_k)$ is the probability the word in position i is w_k, given v_j.

An additional assumption: $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, j, k, m$

An example: learning to classify text

It is therefore necessary to estimate "only" the $P(v_j)\ \forall j$ and the $P(w_k|v_j)\ \forall k, j$

We can use a m-estimate with uniform priors and m equal to the size of the vocabulary.

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

where

- n is the total number of word positions in all training documents having class v_j
- n_k is the number of times the word w_k is in these positions
- $|Vocabulary|$ is the total number of distinct words found in the training set

An example: learning to classify text

LEARN_NAIVE_BAYES_TEXT($Examples$, V)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. *collect all words, punctuation, and other tokens that occur in Examples*
   - $Vocabulary \leftarrow$ the set of all distinct words and other tokens occurring in any text document from $Examples$
2. *calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms*
   - For each target value v_j in V do
     - $docs_j \leftarrow$ the subset of documents from $Examples$ for which the target value is v_j
     - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
     - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
     - $n \leftarrow$ total number of distinct word positions in $Text_j$
     - for each word w_k in $Vocabulary$
       - $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
       - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT(Doc)

Return the estimated target value for the document Doc. a_i denotes the word found in the ith position within Doc.

- $positions \leftarrow$ all word positions in Doc that contain tokens found in $Vocabulary$
- Return v_{NB}, where

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

Expectation Maximization (EM) algorithm

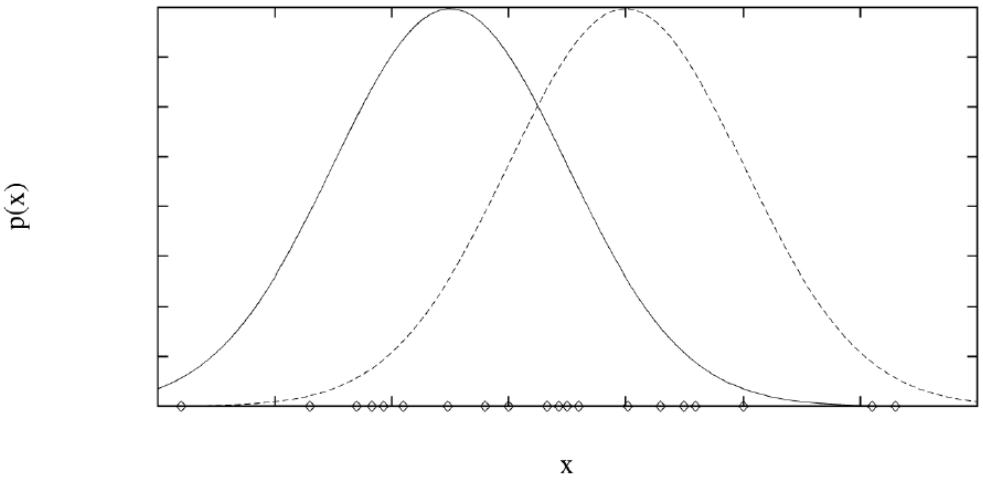
When to use it:

- data only partially observable
- unsupervised clustering (target value not observable)
- supervised learning (some attributes with missing values)

Some examples:

- learning Bayesian networks
- learning of Hidden Markov Models

EM example



Let us assume that each instance x is generated:

- choosing one of the Gaussians with uniform probability
- then, generating a "random" instance according to the chosen Gaussian

EM to estimate k-means

Given:

- X instances generated by a mixture of k Gaussians
- Unknown Gaussian means $\langle \mu_1, \ldots, \mu_k \rangle$ (we assume σ^2 known and the same for all the Gaussians).
- it is not known which instance x_i was generated by which Gaussian

Determine:

- maximum-likelihood extimates of $\langle \mu_1, \ldots, \mu_k \rangle$

In the case $k = 2$, each instance can be seen in the form $y_i = \langle x, z_{i1}, z_{i2} \rangle$ (easy to generalize to the case $k > 2$), where:

- z_{ij} is 1 if the example i was generated by the Gaussian j, 0 otherwise
- x_i observable
- z_{ij} not observable

EM to estimate k-means ($k = 2$)

EM algorithm:

Randomly choose the initial hypothesis $h = \langle \mu_1, \mu_2 \rangle$. Then repeat:

- **E step**: Calculate the expected value $E[z_{ij}]$ of each unobservable variable z_{ij}, assuming the current hypothesis h holds

$$\begin{aligned} E[z_{ij}] &= \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i | \mu = \mu_n)} \\ &= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}} \end{aligned}$$

- **M step**: Compute the new maximum-likelihood hypothesis h, assuming that the value taken from each unobservable variable z_{ij} is its expected value (calculated above)

$$\pi_{ij} = \frac{E[z_{ij}]}{\sum_{i=1}^{m} E[z_{ij}]} \quad \text{and} \quad \mu_j \leftarrow \sum_{i=1}^{m} \pi_{ij} x_i$$

Recap

Topics:

- Bayes theorem
- MAP and ML hypotheses
- Most probable classification
- Intro Naive Bayes
- Naive Bayes for textual documents
- EM algorithm

Exercises:

- Implement NB on textual documents (for example "Twenty User Newsgroups", `http://scikit-learn.org/stable/datasets/twenty_newsgroups.html`)
- Compare it with other classifiers