Machine Learning II
Week 4 Lecture – 30th October 2019
Conditional independence and the naive Bayes Classifier

Contents

► Naive Bayes classifier
- Conditional independence

► Brief Overview of Bayesian Machine Learning
- The general Bayesian approach
- Bayesian Network
- Bayesian Neural Network

Classification as conditional probability

A supervised machine learning method is called *regression* if the outcome variable is numeric, usually continuous. If the outcome is a nominal variable then the mothod is called *classification*, which is what we are considering today.

The outcome Y variable belongs to one of K classes: $Y \in \{1, 2, \ldots K\}$
We also have predictor variables $x_1, \ldots x_p$ which are known.

Suppose we know the probability that $Y{=}k$ given the values $x_1, \ldots x_p$. This is written:

$$P(Y{=}k|x_1, \ldots, x_p) \tag{1}$$

A sensible classifier is one that chooses the value k which maximises this conditional probability.

This is called a **Bayes classifier** (ML1 Lecture 8) and in the language of Bayesian statistics Equation (1) is the posterior probability of Y given $x_1, \dots, x_p$

The Bayes classifier is a theoretical best case classifier, not an algorithm.

In practice we can rarely obtain this probability exactly, we need a method to approximate these probabilities, and there are many different methods to do this.

Lets take a step back and simplify the expression in (1) ...

Assume that there is only one predictor X and this is a discrete random variable.

The probability in Equation (1) becomes $P(Y{=}k|X{=}x)$

Bayes' Theorem "reverses" the direction of the conditioning, so the expression is in terms of the probability X given Y...

$$P(Y{=}k|X{=}x) = \frac{P(X{=}x|Y{=}k)p(Y{=}k)}{P(X{=}x)}$$

The denominator does not involve Y at all, so the value of k which maximises the left hand side also maximises $P(X{=}x|Y{=}k)P(Y{=}k)$

In other words if we compute

$$f(k) = P(X{=}x|Y{=}k)P(Y{=}k)$$

for every class k, then we can choose $\arg\max f(k)$ as the best class and

$$P(Y{=}k|X{=}x) = \frac{f(k)}{\sum_j f(j)}$$

In Bayesian statistics we often express (1) as

$$P(Y{=}k|X{=}x) \propto P(X{=}x|Y{=}k)p(Y{=}k)$$

$P(X{=}x|Y{=}k)$ is called the likelihood function and $P(Y{=}k)$ the prior probability.

Both the likelihood and prior are usually feasible to compute.

Now we consider when we have two discrete predictor variables X_1 and X_2.

Starting with (1) and applying Bayes' Theorem twice, we get

$$P(Y{=}k|X_1{=}x_1, X_2{=}x_2) \propto P(X_1{=}x_1|Y{=}k, X_2{=}x_2)P(X_2{=}x_2|Y{=}k)P(Y{=}k). \tag{2}$$

Independence and conditional independence

Revision:
X_1 and X_2 are independent when

$$P(X_1|X_2) = P(X_1)$$

"The probability of X_1 doesn't change if we know X_2"

Equivalently, X_1 and X_2 are independent when

$$P(X_1 \cap X_2) = P(X_1)P(X_2)$$

and

$$P(X_2|X_1) = P(X_2)$$

Example:
The probability of rolling a six and passing the course Machine Learning II are independent.

Conditional independence

X_1 and X_2 are **conditionally independent given** Y when

$$P(X_1|Y, X_2) = P(X_1|Y)$$

"If we know Y then it's irrelevant what X_2 is!"

Equivalently, X_1 and X_2 are conditionally independent given Y when

$$P(X_1 \cap X_2|Y) = P(X_1|Y)P(X_2|Y)$$

and

$$P(X_2|Y, X_1) = P(X_2|Y)$$

Examples of conditional independence in practice

The event that a Beuth student this semester ...

studies *Statistical Computing*	X_1
studies *Computer Science for Big Data*	X_2
is a 1st semester Data Science Masters Student	Y

X_1 and X_2 are not independent. $P(X_1|X_2) > P(X_1)$.

They (probably) are independent once we know whether the student is a 1st Semester Data Science student:

$$P(X_1|Y, X_2) = P(X_1|Y)$$

(Y “causes” X_1 and Y “causes” X_2)

The event that a person ...

is a regular smoker	X_1
has high blood pressure	Y
suffers a heart attack	X_2

X_1 and X_2 are not independent but they are independent once we know that the person has high blood pressure or does not have high blood pressure, (X_1 “causes” Y “causes” X_2)

In the *naive Bayes* (NB) model we assume that all of the predictor variables are conditionally independent given which class Y belongs to.

Returning to Equation 2: if X_1 and X_2 are conditionally independent given Y

$$\begin{aligned} P(Y{=}k|X_1{=}x_1, X_2{=}x_2) &\propto P(X_1{=}x_1|Y{=}k, X_2{=}x_2)P(X_2{=}x_2|Y{=}k)P(Y{=}k) \\ &= P(X_1{=}x_1|Y{=}k)P(X_2{=}x_2|Y{=}k)P(Y{=}k) \end{aligned}$$

So the classification based on p variables, applying Bayes' Theorem p times:

$$\begin{aligned} P(Y{=}k|x_1, \ldots, x_p) \propto\ & P(x_1|Y{=}k, x_2, \ldots, x_p) \\ & \times P(x_2|Y{=}k, x_3, \ldots, x_p) \\ & \cdots \times P(x_p|Y{=}k)P(Y{=}k) \end{aligned}$$

and **if** all of the predictor variables are conditionally independent, once the class Y is known then each of the factors $P(x_j|Y{=}k, x_{j+1}, \ldots, x_p)$ reduces to $P(x_j|Y{=}k)$; knowing the class tells us all we need to know about the distribution of x_j.

Under the Naive Bayes assumption

$$P(Y{=}k|x_1, \ldots, x_p) \propto P(Y{=}k) \prod_{j=1}^{p} P(x_j|Y{=}k)$$

Usually $P(x_j|Y{=}k)$ is much easier to compute than $P(x_j|Y{=}k, x_{j+1}, \ldots, x_p)$.

The conditional independence assumption of the Naive Bayes model is a strong assumption.

Is this important?

On a coarse level we can say “as long as the results are good, who cares if the predictor variables are conditionally independent on the class?”

A more mathematical justification is: “the assumption may well be wrong, but the resulting difference in $P(Y{=}k|x_1, \ldots, x_p)$ is small enough that the k which maximises the posterior probability is the same, whether or not the assumption is genuinely true.”

Fitting a naive Bayes classifier

$$P(Y{=}k|x_1, \ldots, x_p) \propto P(Y{=}k) \prod_{j=1}^{p} P(x_j|Y{=}k)$$

$P(Y{=}k)$ the prior probability is easy to estimate. Just obtain the relative frequencies for Y in the data, ignoring the values of the predictor variables.

$P(x_j|Y{=}k)$ is the likelihood of x_j given the class k. This factor will usually be fitted by assuming a distribution and estimating the parameters.

Nominal predictor variables Suppose Variable X_j takes values, which we can just relabel (for convenience) as “A”, “B”,“C” ,...
$P(X_j = A|Y{=}k)$ is estimated directly from the relative frequency of $Y{=}k$ from those observations with a $X_j = A$, ignoring all the other $X_\bullet$ variables.

If the cross-tabulation of Y and X_j is

X_j	A	B	C	D	Total
Y					
Yes	26	27	7	4	64
No	5	5	2	0	12
Total	31	32	9	4	76

The likelihood terms are found by taking the row relative frequencies:

X_j	A	B	C	D
Y				
Yes	0.406	0.422	0.109	0.062
No	0.417	0.417	0.167	0.000

So $P(X_j{=}B|Y{=}\text{Yes}) = 0.422$

Continuous predictor variables

A very common assumption is to use the normal distribution density for the likelihood

$$(x_j|Y{=}k) \sim N(\mu_{jk}, \sigma_{jk}^2)$$

The estimation is then very easy:

Estimate μ_{jk} by taking the mean of x_j for all observations with training class k.

Estimate σ_{jk}^2 similarly using the observed variances. To avoid some combinations of j and k having very small observed variances, the assumption is often made that the variance $\sigma_{jk}^2 = \sigma_j^2$ is independent of the class.

Comments

- Naive Bayes classifiers have been successful in text classification and spam filtering.
- It can be directly used for adaptive learning, eg. as new emails come in.
- It is a fast algorithm.
- There is no means of directly testing whether a variable improves the model or not. Splitting the data into training and test data sets is recommended to assess the accuracy of the classifier. If a model with potentially many variables is to be fitted, it is best to test each variable using cross validation methods (ML 1).

Simple Example: Titanic data

Classifying which of the crew and passengers on the Titanic survive based on: Class (or Crew), Sex and Age.

```
> NB_Titanic=naiveBayes(Survived ~., data=Titanic_df)
> NB_Titanic

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
      No      Yes
0.676965 0.323035

Conditional probabilities:
     Class
Y            1st        2nd        3rd       Crew
  No  0.08187919 0.11208054 0.35436242 0.45167785
  Yes 0.28551336 0.16596343 0.25035162 0.29817159
```

```
     Sex
Y            Male     Female
  No   0.91543624 0.08456376
  Yes 0.51617440 0.48382560

     Age
Y           Child      Adult
  No   0.03489933 0.96510067
  Yes 0.08016878 0.91983122

> NB_Preds=predict(NB_Titanic,Titanic_df)
> table(NB_Preds,Titanic_df$Survived)

NB_Preds   No  Yes
     No  1364  362
     Yes  126  349
```

Connection between NB and linear discriminant analysis

The above naive Bayes model is very similar to the linear discriminant analysis (LDA) model from ML1 Lecture 8.

LDA always uses a multivariate normal distribution for the likelihood $P(x_1, \ldots, x_p | Y{=}k)$:

$$(x_1, \ldots, x_p | Y{=}k) \sim N(\boldsymbol{\mu}_k, \Sigma) \qquad \text{(LDA)}$$

Here Σ is the Variance-Covariance, which allows the predictor variables to be correlated. There is no assumption of conditional independence.

In LDA the prior term $P(Y{=}k)$ is often essentially ignored and is implicitly chosen as $P(Y{=}k) = \frac{1}{K}$

Naive Bayes (NB) can use any distribution for the likelihood, but the correlation between x-variables must be zero.

LDA is can be thought of as a non-naive Bayes with a normal likelihood.

In both LDA and NB the common Variance-Covariance assumption between classes k can be relaxed, in discriminant analysis this results in the QDA model.

$$(x_1, \ldots, x_p | Y{=}k) \sim N(\boldsymbol{\mu}_k, \Sigma_k) \tag{QDA}$$

Bayesian Methods in machine learning

The general Bayesian approach Returning to Equation 1, the prior probability is

$$P(Y{=}k) = p_k$$

and the prosterior probability is

$$P(Y{=}k | x_1, \ldots, x_s) = p_k | x_1, \ldots, x_s \qquad \text{for } k = 1, \ldots K$$

So far we have considered these probabilities as being fixed, where p_k and $p_k | x_1, \ldots, x_p$ are *fixed* parameters.

The full Bayesian method says there is uncertainty associated with these parameters. There is some kind of *distribution associated with our parameters*.

Even if we start with a poor prior distribution, a good model will lead to an accurate posterior distribution once we have collected enough data.

The standard notation is that the full collection of parameters in a model is a vector $\boldsymbol{\theta}$. In the above example $\boldsymbol{\theta}$ is the vector $(p_1, p_2, \dots p_s)^\top$

In supervised learning once the values of $\boldsymbol{\theta}$ and the predictor variables are known, then we can predict the outcome variable Y.

The general Bayesian approach is

- to **consider the parameters as random variables**.
- A prior distribution is given to each parameter $\boldsymbol{\theta}$, and
- a likelihood distribution is given to $x_1, \dots, x_p | Y$.
- The posterior distribution of the parameters $\boldsymbol{\theta}|x_1, \dots, x_p$ tells us everything about the outcome variable Y.

The conceptual idea can be expressed as

$$P(\boldsymbol{\theta}|\text{Data}) = \frac{P(\text{Data}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\text{Data})}$$

Can a model parameter be a random variable? This has been a heavily debated question.

In practice Bayesian methods work well but there is a large trade-off between computational feasibility and placing assumptions on the structure of the model.

The benefits of Bayesian learning are

- You not only get “best predictions” (point estimates) for the model , but a distribution for the predictions. Obtaining a confidence interval for a parameter or estimating quantiles is straight forward. It is also straightforward to assess how sensitive/robust the model is to specific parameters.
- The parameters are often correlated, which the posterior distribution incorporates.
- It is possible to simulate outcome variables from the posterior distribution, so a whole range of possible outcomes can be analysed.

An example of the 3rd point:
Time series data for a stock market index is modelled using a RNN.
At the end of the trading day we have an estimate of the price at the end of tomorrow +5 cents per share.

One stockbroker wants to know: “if I buy 1000 shares tomorrow morning what is the risk that I lose money by the end of the day?”

Another stockbroker wants to know: “if I buy 1000 shares tomorrow morning what are the chances the shares rise by 10 cents?”

If the posterior distribution of tomorrows share price is obtained then these questions (and many others) can be easily answered.

A **Bayesian Network (Bayes Net)** specifies which model parameters are conditionally independent. This is done using a graph which connects all the parameters.

Each parameter is represented by a node. If there is a conditional dependency between two parameters, the two nodes are connected by an edge.

Two parameters which do not have a direct connection are conditionally independent if the other parameter values are known.

We have seen in the Naive Bayes Classifier, that the computation in a Bayesian model can be greatly simplified by using this conditional independence structure.

An artificial neural network (NN) can easily be set in a Bayesian framework. A neural network is often represented using input nodes, hidden layer nodes and output nodes. Many but not all of the nodes are connected by lines (the network) which have a corresponding weight attached.

When specifying neural net as a Bayes net, we already know the conditional dependencies! They are defined by the connections in the Neural net.

Two parameters are conditionally independent if the other (neighbouring) parameters are known.

In a NN two quite different combination of parameters can give a similarly good point estimate. With standard NN methods this codependency is not easy to investigate, but if the posterior distribution is simulated, then we can capture this.

Some researchers recommend using an ensemble of NNs, running similar models and averaging the output/take a majority vote. Estimating the posterior distribution is a better approach.

Disadvantages of the Bayesian approach

$$P(\boldsymbol{\theta}|\text{Data}) = \frac{P(Data|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\text{Data})}$$

The numerator is usually straightforward to calculate.
The problem comes in calculating the denominator.

The law of total probability gives us:

$$P(\text{Data}) = \int P(Data|\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}$$

But this is usually very difficult to approximate, as it is a many dimensional integral.

In Naive Bayes we do not have a fully Bayesian model. We assume fixed values for the prior $P(Y{=}k)$ which we estimate, rather than a distribution. As long as we have a sensible number of classes we can calculate the posterior by using the raw relative frequencies of $Y{=}k$.

In a neural network and most Bayes nets we have many continuous parameters and it is not feasible to obtain $P(\text{Data})$.

The usual approach not to obtain the distribution of $\boldsymbol{\theta}|\text{Data}$ explicitly, but to programme a method to simulate from this distribution.

This is done using so called MCMC (Monte Carlo Markov Chain) simulation. In ML2 Week 1 you very briefly met Gibbs Sampling which is s specific type of MCMC simulation.

We will not look further into fitting a Bayesian neural networks.... A whole Data Science course could be on MCMC methods!