A dynamic naive Bayes classifier (DNBC) is an extension of the popular probabilistic graphical model called hidden Markov model. Output variables are assumed to be statistically independent, which helps us in case of the curse of dimensionality occurring in high-dimensional space. Moreover, output variables that come from different probability distributions can be learned easier.

This thesis aims to create a parallel implementation of DNBC that can be easily executed on a computational cluster. For that reason, the algorithm will be implemented in the Scala language on top of Apache Spark.

1. Study DNBC and its possibilities of parallelism.
2. Implement DNBC that is applicable for both discrete and continuous output variables.
3. Parallelize the implementation on top of Apache Spark.
4. Evaluate the parallel implementation and compare it with the sequential implementation in terms of parallel scalability.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Paralelní implementace dynamického naivního Bayesovského klasifikátoru

Pavel Lučivňák

Katedra teoretické informatiky
Supervisor: Ing. Tomáš Šabata

March 15, 2018

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In V Praze on March 15, 2018 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Lučivňák, Pavel. *Paralelní implementace dynamického naivního Bayesovského klasifikátoru*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by se čtenář měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova Nahraďte seznamem klíčových slov v češtině oddělených čárkou.

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords Nahraďte seznamem klíčových slov v angličtině oddělených čárkou.

Contents

List of Figures

Introduction

Chapter 1

Cíl práce

Chapter 2

Analysis

2.1 Hidden Markov Model

2.1.1 Description

A Hidden Markov Model (HMM) is defined by

1. Initial transition function I: $S \mapsto \mathbf{R}$
2. Transition function T: $S \times S \mapsto \mathbf{R}$
3. Emission function E: $S \times X \mapsto \mathbf{R}$
4. Set of hidden states S
5. Set of observed states X

The aforementioned functions return a probability. This means the resulting number lies in interval $[0, 1]$ and the sum of returned numbers for all function parameters adds to 1.

The set of observed states can be either discrete or continuous. The set of hidden states is assumed to be discrete in the standard definition of HMM.

Given the following:

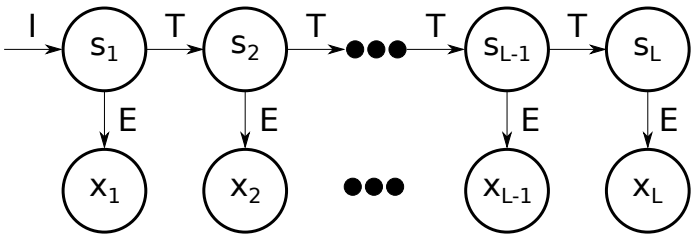

Figure 2.1: Visualization of a Hidden Markov Model. At each time point, there is a hidden state s_i and an observed state x_i.

1. A number $L \in [1, \text{inf})$
2. A sequence of hidden states of length L
3. A sequence of observed states of length L
4. An index $i \in [1, L]$: one can think of this as a discrete point in time

At time point i, there is a hidden state s_i and an observed state x_i. Value of the observed state depends on value of the hidden state. The dependency is described by emission function E.

If $i \neq 1$ and $L > 1$, there is a hidden state s_i that depends on value of previous hidden state s_{i-1}. The dependency is described by transition function T.

If $i = 1$, there is a hidden state s_1 that depends on initial transition function I.

2.1.2 Operations

There are two most important operations that can be performed on HMM. Learning and inference.

2.1.2.1 Learning

Learning is used to calculate the model parameters $\{I, E, T\} = \lambda$. It is desired to estimate the parameters such that $\prod_{j=1}^{M} P(\mathbf{x}_j|\lambda)$ is maximized. Where $M \in \mathbf{N}$ is a number of sequences to be learned. $\mathbf{x}_j$ is a j-th sequence of observed states. In another words, it is desired to maximize the product of probabilities that given sequence of observed states was generated by given model.

2.1.2.2 Inference

Inference returns the most likely sequence of hidden states, given a sequence of observed states.

2.1.2.3 Scoring?

2.1.3 Learning

2.1.3.1 Maximum likelihood estimation

This method uses maximum likelihood estimation (MLE) to compute model parameters. MLE is a technique for finding parameters of a probabilistic model that best describe behavior of a random variable. The method aims to maximize the likelihood of all the learning data.

Formally $\text{argmax}_{\theta \in \Theta} \prod_{j=1}^{M} L(x_j, \theta)$, where:

- M is number of data points
- x_j is a j-th data point
- L is a likelihood function (defined further)
- θ describes model parameters
- Θ is a set of all model parameters

Let's examine how one can view parameters of HMM as probabilistic functions of random variables.

Initial transition function Function I: $S \mapsto \mathbf{R}$ is in fact a probability function corresponding to random variable S. To represent it, let's use a discrete probabilistic model with parameter ϵ.

Transition function Since the function is $S \times S \mapsto \mathbf{R}$, one can think about it the following way. For each hidden state $s \in S$, there is a probability function $S \mapsto \mathbf{R}$. We can represent such a function in the same way as the initial transition function. Therefore for each hidden state $s \in S$, there is a discrete probabilistic model with parameter ϵ.

Emission function Since E: $S \times X \mapsto \mathbf{R}$, one can think about it this way. For each hidden state $s \in S$, there is a probability function $X \mapsto \mathbf{R}$. This function can be represented by a continuous probabilistic model. For purposed of this thesis, it is assumed that continuous random variables have Gaussian or Gaussian mixture distribution.

Discrete random variables Here I present a straightforward way of defining the L likelihood function in case of discrete random variables. $L(x, \theta) = x_{cnt}/M$, where x_{cnt} is the number of times x occurs in learning data. The number of data points is M. Since the likelihood function does not depend on parameter θ, there is no expression to optimize.

There is a reason I did not choose any standard discrete probability distribution to define L. The random variable does not have to be $\mathbf{Z}$, nor $\mathbf{N}$. In fact it can be any abstract object, such as an animal. A type, where comparison between two objects doesn't make sense. Therefore, it wouldn't make sense to assign non zero probability to values that are not specified in learning phase.

As an example, consider the following data: {dog, bird, dog, cat, bird, dog}. Figure 2.3 shows a likelihood function L associated with the data.

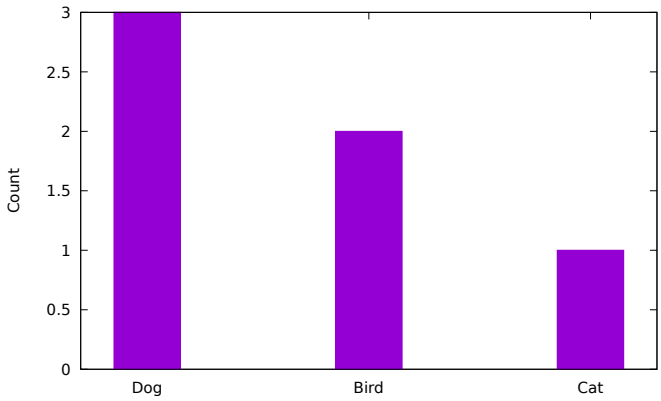

Figure 2.2: Frequency of data in discrete data set.

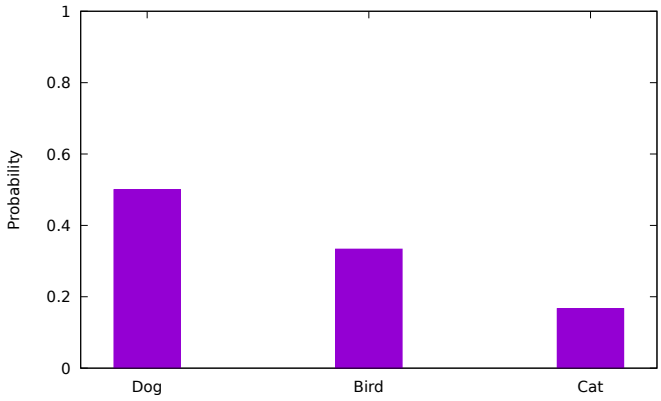

Figure 2.3: Likelihood function L of data in discrete data set. Probability is zero at undefined states.

Continuous random variables Let's consider Gaussian (normal) distribution, defined by $\theta = \{\mu, \sigma^2\}$. The goal is to find parameter $\theta \in \Theta$, such that the product $\prod_{j=1}^{M} L(x_j, \theta)$ is maximized. In case of Gaussian distribution, the likelihood function L is defined by $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$.

Taking derivative with respect to μ equal to 0 yields $\sum_{j=1}^{M} x_j - M\mu = 0$. Maximum likelihood estimate of μ is therefore equal to $\overline{X}_M$.

Derivative with respect to σ^2 equal to zero results in MLE of σ^2 to be equal to $\frac{1}{M} \sum_{j=1}^{M} (X_j - \overline{X}_M)^2$.

Figure 2.4 shows an example of MLE on normally distributed random variable.

TODO: Gaussian mixtures

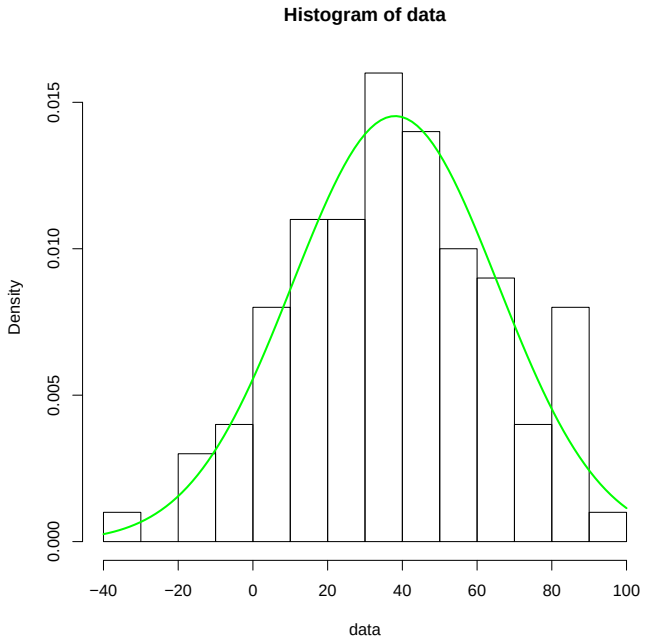

Figure 2.4: Histogram of randomly generated data from normal distribution $\mathcal{N}(40, 32^2)$. The green curve is a plot of normal distribution with the maximum likelihood estimate of θ parameters.

2.1.4 Inference

Given a sequence of observed states, what is the most likely associated sequence of hidden states? Viterbi algorithm answers this question. Let's define $a_t(i)$ to be the maximum probability of sequence of hidden and observed states up to time point t:

$$a_t(i) = max_{s_1,s_2,\ldots,s_{t-1}} P(s_1, s_2, \ldots, s_{t-1}, s_t = S_i, x_1, x_2, \ldots, x_t|\theta) \tag{2.1}$$

Where S_i is a function $\mathbf{N} \mapsto S$, assigning a unique index to each hidden state in S. θ are parameters of HMM model.

Initialization At time point $t = 1$, the probability of being at hidden state S_i is simply the initial probability of being at that state and a probability of being at observed state x_1 given hidden state S_i. In another words:

$$a_1(i) = I(S_i)E(S_i, x_1) \tag{2.2}$$

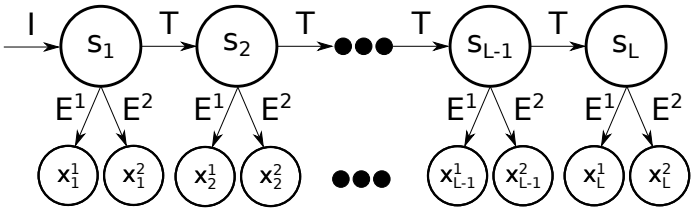

Figure 2.5: Visualization of a Dynamic naive Bayesian classifier with two observed variables. At each time point, there is a hidden state s_i and two observed states: x_i^1 and x_i^2.

Recursion Taking into account the structure of HMM, the equation 2.1 can be rewritten recursively as

$$a_t(j) = [\max_i a_{t-1}(i)T(S_i, S_j)]E(S_j, x_t) \tag{2.3}$$

Hidden state at particular time point t is determined by

$$s_t = S_{\text{argmax}_i\, a_t(i)} \tag{2.4}$$

Complexity At each time point t and hidden state index j, the equation 2.3 loops through every hidden state index i. There are $|S|$ hidden states. In addition, the recursion continues for every time point t. There are L time points. Thus the overall complexity is $|S|^2 L$.

2.2 Dynamic naive Bayesian classifier

2.2.1 Description

Dynamic naive Bayesian classifier (DNBC) is an extension of Hidden Markov Model (HMM). The difference is that there may be a number of observed variables. In contrast, HMM is defined for only one observed variable. Figure 2.5 demonstrates model structure with 2 observed variables. DNBC is defined by

1. Number of observed variables N
2. Initial transition function $I : S \mapsto \mathbf{R}$
3. Transition function $T : S \times S \mapsto \mathbf{R}$
4. Emission functions $E^j : S \times X^j \mapsto \mathbf{R}, j \in [1, N]$
5. Set of hidden states S
6. Sets of observed states $X^j, j \in [1, N]$

Each set of observed variables contains a set of observed states. This set can be either discrete or continuous. The set of hidden states is assumed to be discrete.

Given the following:

1. A number $L \in [1, \text{inf})$
2. A sequence of hidden states of length L
3. Sequences of observed states of length L
4. An index $i \in [1, L]$: one can think of this as a discrete point in time

At time point i, there is a hidden state s_i and N observed states x_i^j where $j \in [1, N]$. Value of every observed state depends on value of the hidden state. The dependency is described by emission function E^j. An important property of DNBC is that the observed variables are assumed to be independent. Therefore, DNBC does not define any dependency function between two observed variables.

If $i \neq 1$ and $L > 1$, there is a hidden state s_i that depends on value of previous hidden state s_{i-1}. The dependency is described by transition function T.

If $i = 1$, there is a hidden state s_1 that depends on initial transition function I.

2.2.2 Operations

There are two most important operations that can be performed on DNBC. Learning and inference.

2.2.2.1 Learning

Learning is used to calculate the model parameters $\{I, T, E^1, E^2, \ldots, E^N\} = \lambda$. It is desired to estimate the parameters such that $\prod_{j=1}^{M} P(\mathbf{x}_j|\lambda)$ is maximized. Where $M \in \mathbf{N}$ is a number of sequences to be learned. $\mathbf{x}_j$ is a j-th sequence of observed states. In another words, it is desired to maximize the product of probabilities that given sequence of observed states was generated by given model.

2.2.2.2 Inference

Inference returns the most likely sequence of hidden states, given sequences of observed states. Each sequence of observed states corresponds to particular observed variable. Each sequence thus has the same number of elements.

2.2.3 Learning

2.2.3.1 Maximum likelihood estimation

This learning approach is similar to one described in case of HMM. The only difference is that there are multiple emission functions E^j where $j \in [1, N]$. Every emission function is defined as $E^j : S \times X^j \mapsto \mathbf{R}$. For every j and hidden state $s \in S$, there is a probability function $X^j \mapsto \mathbf{R}$. This function can be represented by continuous probabilistic model. Therefore, there is now $N \times |S|$ continuous distributions to be learned. Each distribution is assumed to be Gaussian or Gaussian mixture.

2.2.4 Inference

Inference algorithm is again Viterbi. It is simply extended to support multiple observed variables. Let x_t^j be a sequence of observed states of variable j up to time point t.

$$x_t^j = x_1^j, x_2^j, \ldots, x_t^j \tag{2.5}$$

Let's define $a_t(i)$ to be the maximum probability of sequence of hidden and sets of observed states up to time point t:

$$a_t(i) = max_{s_1,s_2,\ldots,s_{t-1}} P(s_1, s_2, \ldots, s_{t-1}, s_t = S_i, x_t^1, x_t^2, \ldots, x_t^N|\theta) \tag{2.6}$$

Where S_i is a function $\mathbf{N} \mapsto S$, assigning a unique index to each hidden state in S. θ are parameters of DNBC model.

Initialization At time point $t = 1$, the probability of being at hidden state S_i is equal to the initial probability of being at that state and probabilities of being at observed state x_1^j given hidden state S_i. In another words:

$$a_1(i) = I(S_i) \prod_{j=1}^{N} E^j(S_i, x_1^j) \tag{2.7}$$

Recursion Taking into account the structure of DNBC, the equation 2.6 can be rewritten recursively as

$$a_t(j) = [\max_i a_{t-1}(i) T(S_i, S_j)] \prod_{k=1}^{N} E^k(S_j, x_t^k) \tag{2.8}$$

Hidden state at particular time point t is determined by

$$s_t = S_{\text{argmax}_i \, a_t(i)} \tag{2.9}$$

Complexity At each time point t and hidden state index j, the equation 2.8 loops through every hidden state index i and every observed variable index k. There are $|S|$ hidden states and N observed variables. In addition, the recursion continues for every time point t. There are L time points. Thus the overall time complexity is $|S|(|S| + N)L$.

Chapter 3

Návrh

Chapter 4

Realizace

Conclusion

Appendix A

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

Appendix B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS