A Survey on Machine Learning and Learning Representations of Distributions

Chen Zhang
NetID : czhang49
czhang49@illinois.edu

1 Report Objective and Organization

This report is a summary of CS446 course project. The initial objective of the project is to have an in-depth understanding of representation of multi-linear representations by studying the paper ”Learning multi-linear representations of distributions for efficient inference”[1]. While doing the project, another two paper titled ”Learning with mixtures of trees”[2] and ”Learning probabilistic decision graphs”[3] is found to have a very close relation with what is discussed in the course. These two papers make use of decision trees, Bayesian networks, probability distribution and EM algorithm. Thus a broader view and survey regarding these three papers is conducted while an inclination to the first one is still maintained. The first part of the report is a general description of machine learning. Following this, some of the needed theories in understanding the multi-linear representation described in [1]is introduced. Then a study of the three papers [1] are presented in the author's own words, although some of the equations may be the same as in above cited paper.

2 Machine Learning

Machine Learning is a science discipline that focus on how to use computers to simulate human learning process, so that it could obtain, identify and even create new knowledge. In machine learning, the patterns of how human being learn are analyzed, and algorithms are developed so that computers could be able to make predictions and decisions, rather than follow an specifically programmed instructions. Tom.M.Michell put it as ”A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”[4]. There exist a great number of reviews and explanations of machine learning [5][6] and people from different perspective have different understanding, vision, and make use of different terminologies. A history and general overview of machine learning can be found in the above cited papers. There are three general categories in inductive learning: supervised learning, unsupervised learning and reinforcement learning[7]. In supervised learning, the learner is provided with examples as inputs and their outputs and expected to learn the rules that map the inputs into outputs. In unsupervised learning, outputs of examples are not given, and the learner is expected to learn the structure of the input. In reinforcement learning, the learner is given a set of environments and actions, and expected to learn the rule in order to perform a certain task.[8]

2.1 Supervised Learning algorithms

In this section, a comparison of the supervised learning algorithms, particularly those covered in class, is summarized and compared based on some of their properties. Theses supervised learning algorithms include : decision trees, perceptron based methods, support vector machines, naive Bayes classifiers, and Bayesian networks.Many good review and survey papers can be found on this subject. [8][5][6][9].

Bias and variance are two important quantity that measures the error of the classifier when trained on different data, and measure the error of deviation, respectively. Naive Bayes and the Bayesian network are found to have a high bias, due to the fact that the assumption of NB is not satisfied by most of the dataset. On the other hand, decision trees and support vector machines have high variance, meaning that they may tend to overfit the dataset as the size of the dataset grows.

In terms of the storage, Naive Bayes and Bayesian Networks are reported to behave very well, since they only store the prior and conditional probabilities, and in the case of BN, only the conditional probabilities of each node. This is in contrast to algorithms like Perceptron, decision trees, and support vector machines, where large space of storage is required, which leads to many modifications on the algorithm such as the kernel methods to reduce the storage.

Another one important consideration is the accuracy. Unfortunately there is no universal rule saying which algorithm is better than which. A more practical method is simply to apply each of the candidate algorithm to the problem and compare the accuracy.[10][11] Many people looked at combining different classification methods. But still there is no universal rule of differentiating one combined classifier over another.

3 Symbolic Propagation in MLR

Now we will start to look at the Multi-linear representations of distributions. We will start by looking at the symbolic representation in Bayesian networks. Castillo introduced a set of symbolic representation of distributions in Bayesian Networks[12]. The main idea is as follows. Let $X = X_1, X_2, \dots, X_n$ be a set of n discrete variables, each can take values in a set $0, 1, \dots, r_i$, which is the possible states of X_i. Using the Bayesian networks, the joint probability density of $\vec{X}$ can be written as:

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^{\pi} P_i(x_i|\pi_i).$$

Introduce another notation for the parameters

$$\theta_{ij\pi} = P_i(X_i = j|\pi_i), j \in (0, \dots, r_i).$$

The first subscript in $\theta_{ij\pi}$ refers to the node number, the second subscript refers to the label of the node and the last subscript refers to the label of its parent. An example would be

$$\theta_{200} = P(X_2 = 0|X_1 = 0)$$
$$\theta_{301} = P(X_3 = 0|X_1 = 0)$$
$$\theta_{4001} = P(X_4 = 0|X_2 = 0, X_3 = 1)$$

With this notation, the conditional probability in the Bayesian network can be expressed as

$$P(X_i = j) = \sum c_{jr} m_r = p_j(\theta) \tag{1}$$

This propagation notation offers a new set of representation of distributions. The θ_{ijk} represents the conditional probability (relations between nodes), and the c_i is the undetermined coefficients that need to be learned. A set of algorithm to identify the necessary nodes, parameters, monomials, and calculate the coefficients are employed to calculate the propagation. Note that the equation is linear in the sense of the monomials. With this representations, the author shows that the probability distribution over Bayesian Networks can be written in the multi-linear polynomials.

4 Multi-Linear Representation

Let $X = X_1, X_2, \dots, X_n$ be a set of n discrete variables, and each variable have label from a finite set $S_i = [\kappa_i]$. Any distribution $P(\chi)$ can be written as

$$P(\chi) = \sum_{s_1 \in S_1, s_2 \in S_2, \dots, s_n \in S_n} P_{s_1, s_2, \dots, s_n} \prod_{j=1}^{n} I_{X_j = s_j}$$

where $p(s_1, s_2, \dots, s_n) = Pr(X_1 = s_1, X2 = s_2, \dots, X_n = s_n)$ and I is the indicator function. We can see that in this expression, for each term in the monomial, there is one indicator function. The Multi-Linear Representation is defined in a similar which says that each term in the multinomial has at most one indicator variable corresponding to any variable $X_i \in \chi$. We can see that in terms of the indicator variables, the representation is linear. Thus it is called multi-linear representation. Writing this equation in a form similar to the propagation introduced in the previous section, it could be expressed as

$$P_D(\chi) = \sum_{i=1}^{t} c_i r_i = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} I_{X_j} \tag{2}$$

In the case of boolean variables, (2) can be expressed as

$$P_D(\chi) = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} X_j^{s_{ij}} (1 - X_j)^{s'_{ij}} \qquad (3)$$

In (3), X_j is the variable value, and c_i is the undetermined coefficients. The goal then is to learn a set of c_i that could best describe the model.

4.1 Validity of ML Representation

To prove the validity of the representation, we need to show two things: the integral of distribution is 1 and the distribution always give positive probabilities. Here we will look at the first condition.
To prove the first requirement, we first write out the multi-linear representation

$$P_D(\chi) = \sum_{i=1}^{t} c_i r_i = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} I_{X_j}$$

$$P_D(\chi) = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} X_j^{s_{ij}} (1 - X_j)^{s'_{ij}}$$

From these two equations, we can see that to satisfy requirement 1, we need to impose a normalization factor to the coefficients so that the following equation is satisfied

$$\sum_{x \in S_1 \times S_2 \times \ldots \times S_n} P(x) = 1$$

By doing integration over all the variables, this constraint lead to the equation

$$\sum_{i=1}^{t} c_i 2^{n - \sum_{j=1}^{n} (b_{ij} + b'_{ij})} = 1$$

which can be used as the normalization factor to impose. Thus the equation becomes nonlinear in the sense that the normalization factor also contains the unknown variables.

4.2 Exact Learning of Coefficients

Now that we know the MLR is a valid representation for distributions, and a normalization constant needs to be applied, we will look at how to learn the coefficients and then the representation. The exact learning of the coefficients involves calculating the maximum likelihood of the unknown variable c_i. To do this, we need to write the problem in an optimization formulation. Given a set of examples $\chi = X^1, X^2, \ldots, X^m$, the joint probability could be written as

$$\begin{aligned} P(Y|C) &= \prod_{l=1}^{m} P(X = Y^l | C) \\ &= \prod_{l=1}^{m} (\sum_{i=1}^{t} c_i \prod_{j=1}^{n} (Y_j^l)^{s_{ij}} (1 - Y_j^l)^{s'_{ij}}) \end{aligned}$$

Thus using maximum likelihood principle, an optimization procedure can be formulated as(using the log likelihood):

$$\begin{aligned} \max_{c_i} \quad & \sum_{l=1}^{m} \log(\sum_{i=1}^{t} q_{i,l} c_i) \\ \text{subject to} \quad & c_i \geq 0, \qquad \forall i \\ & \sum_{x \in (0,1)^n} \sum_{i=1}^{t} c_i \prod_{j=1}^{n} X_j^{s_{ij}} (1 - X_j)^{s'_{ij}} = 1 \end{aligned}$$

It can be seen that in this case the objective function is concave and the constraints are linear. Thus the local minimum is guaranteed to be a global minimum. Now if we optimize this problem, and substitute the resulting c_i into (2), we will have the resulting representation of distributions.

4.3 Approximate Algorithm for Learning

It is shown that the exact learning procedure can be very slow. Thus a trick is introduced to make the algorithm faster. We know that the log likelihood function can be written as

$$LL = \sum_{l=1}^{m} log(\sum_{i=1}^{t} q_{i,l} c_i)$$

A new quantity $Q(l) = \sum_{i}^{t} q_{i,l}$ is defined. This denotes the number of terms satisfied by the lth data of instance Y_l. Then the log-likelihood function can be written , using the property of log function, as

$$LL = \sum_{l=1}^{m} log(\sum_{i=1}^{t} \frac{q_{i,l}}{Q(l)} c_i) + \sum_{l=1}^{m} log Q(l) \qquad (4)$$

Since log is a concave function, moving the log operator inside the summation will give a smaller value than outside the summation, thus we will have

$$LL \geq \sum_{l=1}^{m}\sum_{i=1}^{t}\frac{q_{i,l}}{Q(l)}\log c_i + \sum_{l=1}^{m} logQ(l)$$

If we ignore the last term in the equation above, the '$\geq$' still holds. Thus a different optimization problem can be formulated based on the new equation.

$$\max_{c_i} \quad \sum_{l=1}^{m}\sum_{i=1}^{t}\frac{q_{i,l}}{Q(l)}\log c_i \qquad (5)$$

$$\text{subject to} \quad c_i \geq 0, \qquad \forall i \qquad (6)$$

$$\sum_{i=1}^{t} c_i \prod_{j=1}^{n} X_j^{s_{ij}}(1-X_j)^{s'_{ij}} = 1 (7)$$

5 Learning with mixture of trees

The basics of Bayesian Networks and learning algorithm that we discussed in the course is presented in the paper by Heckerman Et al.[13]. Yet the resulting representation is hard in the sense of computation to make inference with. In 2001, Meila[2] used a mixture of trees in an attempt to reduce the complexity and improve inference efficiency. The main points of the mixture of trees is summarized here. The mixture of trees can be defined as:

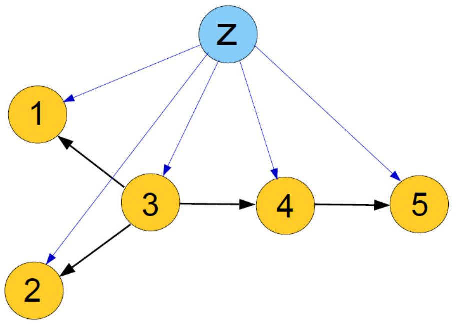

Figure 1: A Mixture Of Trees. Courtesy:Heckerman, Et al.

$$Q(x) = \sum_{k=1}^{m} \lambda_k T^k(x) \qquad (8)$$

with the constraint that

$$\lambda_k \geq 0, k = 1, \ldots, m \qquad \sum_{k=1}^{m}\lambda_k = 1$$

This basically means that more than one trees are proposed in the model, each one of the tree is denoted by $T^k(x)$, with the probability of λ_k. To learn this mixture of trees, suppose we are given a set of observations $\vec{X} = (X_1, X_2, \ldots, X_n)$, then the mixture of trees need to satisfy

$$Q = argmax \prod_{i=1}^{N} Q(x_i)$$

Using the log-likelyhood, the equation becomes

$$Q = argmax \sum_{i=1}^{N} \log Q(x_i).$$

Now that we don't know the distribution of the mixture of trees, we will use the EM algorithm to calculate the likelihood. The conditional distribution of the mixture of trees can be calculated as:

$$\gamma_k(i) = \frac{\lambda_k T^k(x^i)}{\sum_{k'} \lambda_{k'} T^{k'}(x^i)}$$

Substituting this into the expectation of log-likelihood, we obtain

$$E[(x^{1,\ldots,N}, Q^{1,\ldots,m})] = \sum_{i=1}^{N}\sum_{k=1}^{m} \gamma_k(i)(\log\lambda_k + logT^k(x^i))$$

With this formulation, we can take the maximum likelihood of the undetermined parameters by using the gradient methods.
Using the mixture of trees, experiments are carried out, and it is observed that the mixture of trees has a better accuracy over methods like linear discrimination, Naive Bayes and Bayesian networks. In terms of computational cost, there is a slight increase on the efficiency using traditional algorithms. By taking advantage of the sparseness of the tree structures, a 5-7 times improvement on the efficiency is reported in the experiments.

6 Learning Probabilistic Decision Graphs

In 2006, Jaeger[3] proposed a method to learn a probabilistic decision graph. A probabilistic decision graph is rooted on tree representations of distributions. Eachnode X_i in the forest is expanded into a set V_i of nodes. v_i in the probabilistic decision graph is connected in the following manner: for each successor X_j of X_i in the tree, and each possible value of X_i there is one outgoing edge connecting these two states. Basically each V_i represent a node and each v_j represent a state of the node, the line style (solid or dotted) connecting two v_j represents value 1 or 0, respectively. Then the values in the node denotes the probability distribution over the possible states of X_i. The probabilistic decision

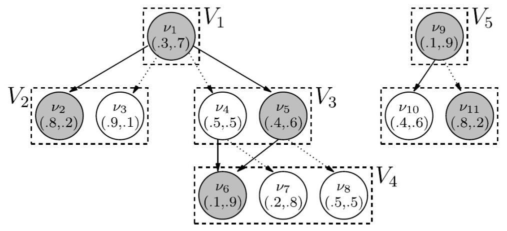

Figure 2: A probabilistic distribution graph. Courtesy:Jaeger, Et al.

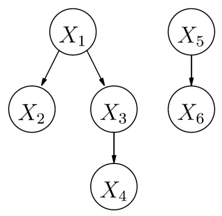

Figure 3: A probabilistic distribution graph. Courtesy:Jaeger, Et al.

graph in Fig. 2 is based on the tree structure shown in Fig. 3.

Similar to the Bayesian networks, the efficiency of learning a probabilistic decision graph can be measured by the size of the PDG. It is reported that the results from applying the PDG method is very similar to the Bayesian networks. Some improvement of accuracy is observed with PDG over Bayesian networks, but the overall computational cost and complexity is very close.

7 Summary

This report summarize the CS446 project of the author. In this project, the author conducted a survey of the machine learning history and its development. A short literature review for the comparison of some of the algorithms that are discussed in the course is also presented. Then author studied in-depth the multi-linear representations of distributions. Although some of the concepts is not well understood, the main idea and logic of the paper is studied and presented. The author also found two paper interesting and very well related to the concepts and algorithms discussed in class. These two papers focused on learning distributions with mixture of trees and learning probabilistic decision graph. The author went into details for the parts which has a connection with the algorithms discussed in the course.

The author felt learning quite a lot in both the course and the project. The project helps the author to have the feeling that what is discussed in the course is indeed the state-of-the-art of machine learning. Now that we are equipped with the essentials to enter this field, either as a personal interest or as a specialist.

References

[1] Dan Roth and Rajhans Samdani. Learning multi-linear representations of distributions for efficient inference. *Machine Learning*, 76(2-3):195–209, 2009.

[2] Marina Meila and Michael I Jordan. Learning with mixtures of trees. *The Journal of Machine Learning Research*, 1:1–48, 2001.

[3] Manfred Jaeger, Jens D Nielsen, and Tomi Silander. Learning probabilistic decision graphs. *International Journal of Approximate Reasoning*, 42(1):84–100, 2006.

[4] John Robert Anderson, Ryszard Stanisław Michalski, Jaime Guillermo Carbonell, and Tom Michael Mitchell. *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986.

[5] David M Dutton and Gerard V Conroy. A review of machine learning. *The Knowledge Engineering Review*, 12(04):341–367, 1997.

[6] Ramon Lopez de Mantaras and Eva Armengol. Machine learning from examples: Inductive and lazy methods. *Data & Knowledge Engineering*, 25(1):99–123, 1998.

[7] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intel-*

ligence. Prentice-Hall, Egnlewood Cliffs, 25, 1995.

[8] Sotiris B Kotsiantis, ID Zaharakis, and PE Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[9] Donald Michie, David J Spiegelhalter, and Charles C Taylor. Machine learning, neural and statistical classification. 1994.

[10] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[11] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

[12] Enrique Castillo, José Manuel Gutiérrez, and Ali S Hadi. Goal oriented symbolic propagation in bayesian networks. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 1263–1268. Citeseer, 1996.

[13] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.