A Probability-Based Approach to Soft Discretization for Bayesian Networks

Research Report GT-ME-2009-002

Imme Ebert-Uphoff*
Adjunct Associate Professor
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0405
ebert@me.gatech.edu

September 22, 2009

Key Words: Soft Discretization, Bayesian Network, Probability, Fuzzy Set Theory, Membership Function, Convolution.

Abstract

This report discusses how soft discretization can be implemented to train a discrete Bayesian Network directly from continuous data. The method consists of a soft discretization step that converts the continuous variables of the training cases into soft evidence, followed by a suitable parameter learning algorithm for the Bayesian Network. The learning algorithm is a modification of the Maximum Likelihood Estimation algorithm which is modified to accept soft evidence as input. We also discuss how to use soft discretization for inference and how to convert the inference results from the discrete network to meaningful continuous output values.

Most literature on the use of soft discretization for Bayesian Networks proposes to use fuzzy set theory which is based on membership functions. Our approach goes back one step further and starts out with a probability density function that spreads the influence of a continuous variable to its neighbors, followed by a discretization step. Thus our approach to soft discretization is based on probability theory, rather than fuzzy set theory. We then show an interesting connection between these approaches. Namely, a membership function can be generated from the probability density function through convolution, yielding a set of probability-based membership functions.

Prime applications of this method include any system with limited training data whose underlying mechanism is continuous in nature. These types of applications are common in the natural sciences and medicine. Using the continuity of the system, i.e. the fact that neighboring states in a continuous system are related to each other, we hope that soft discretization can yield more robust and more accurate models from small sample sizes. This report describes the method in enough detail to allow anyone to implement it themselves. Preliminary tests indicate increased robustness, but extensive tests of the performance of the new models in comparison to traditional models have yet to be performed.

1 Introduction

As the name indicates the main idea of soft discretization is to use soft boundaries. For example, if a continuous variable, X, is to be discretized, hard discretization maps it to a single discrete state, while soft discretization maps it to several discrete states with different weights.

*Joint Appointment with the Robotics and Intelligent Machines Center, School of Interactive Computing, College of Computing, Atlanta, GA 30308.

Let us consider a specific example. Our research is motivated by the application of precipitation forecasting (see Section 1.3 for details), and we use precipitation rates simulated by an atmospheric general circulation model as example to illustrate the concept of soft discretization. Let us consider a variable X that represents the amount of rainfall in a certain area per day in mm. We only have limited data, so we use only three states for the variables, dry (0-1mm), medium rain (1-20mm) and heavy rain (20mm-∞). As rain is exponentially distributed the first interval, despite its small size, contains more than half of all cases. Considering a sample case for which X falls into the medium rain interval it makes a big difference in practice whether the value of X is $X = 2mm$ or $X = 18mm$, but using hard discretization each of those values would simply be mapped to the medium rain state. Soft discretization would map $X = 2mm$ to a large weight for the medium rain state and a smaller weight for the dry state. Similarly, $X = 18mm$ would be mapped to the medium rain state with a large weight and to the heavy rain state with a small weight.

As illustrated in this example the largest weight is generally assigned to the discrete case corresponding to hard discretization, but smaller weights are assigned to neighboring states. That way the boundaries become more fluent and the weights better represent the specific location of the continuous value X *within* the medium rain interval.

1.1 Why This Report?

This report grew out of my frustration while trying to find details on how to best implement soft discretization in the context of Bayesian Networks. There are a few papers using fuzzy set theory for soft discretization, but as this is a fairly new topic most of them focus on developing the theoretical framework, and have little space to deal with implementation details, such as how to best choose a membership function for soft discretization.

Furthermore, in an effort to start at the very basics of how soft discretization should be defined, our approach does *not* follow the usual route of fuzzy set theory. Rather than starting with a membership function (as fuzzy approaches do), we start with a probability-based spread model, namely a probability density function. The probability density function (PDF) defines explicitly 'how far and in what shape' a continuous value should be spread to neighboring values before discretization. Once a continuous value is spread to its neighbors, the resulting model can be discretized, resulting in soft discretization. In this fashion the probability density function defines a soft discretization procedure that converts *continuous hard evidence* into *discrete soft evidence*. The key point is that our approach to soft discretization is thus based on probability theory, while fuzzy approaches are based on fuzzy set theory. (An interesting connection between both approaches is established in Section 5.)

As many Bayesian Network toolboxes do not provide a simple means to use soft evidence for the training of Bayesian Networks, we also provide the details on how to modify the standard learning algorithm, Maximum Likelihood Estimation, to use soft evidence for training. The two steps combined, soft discretization and modified Maximum Likelihood Estimation, provide a self-contained learning algorithm that allows one to learn the parameters of a discrete Bayesian Network directly from continuous data (or mixed discrete-continuous data), while having full control over the spread model for each continuous variable.

The purpose of this report it to provide enough details for anyone to easily implement both parts (soft discretization and training the network from training cases) of this algorithm in any toolbox. It also provides the details for implementing inference with soft discretization, as well as suggests strategies for the inverse problem, namely converting the soft evidence that results as output from the inference algorithm into continuous values, which is known as defuzzification in fuzzy set theory. We hope that this document provides a basis for more researchers in the Bayesian Network community to use soft discretization in their work.

1.2 Literature

Many applications generate data with continuous values, but the current algorithms for Bayesian Networks only allow one to model them as continuous nodes in the network if they have a Gaussian distribution. Many applications include continuous valued nodes that do not have Gaussian distribution (in our application all continuous nodes follow an exponential distribution), so these nodes have to be modeled as discrete nodes. For theses applications one must find a way to bridge the gap between continuous data and discrete network nodes, which requires some type of discretization.

There is as of yet no concensus on how to best handle this discretization. Fu [6] reviewed the state-of-the-art for learning the *structure* of (discrete) Bayesian Networks from continuous data. A short version is provided in Fu and Tsamardinos [7]. While *structure* learning is even more complex than just parameter learning, the discretization plays an important role and Fu and Tsamardinos state in the motivation of their research that *[...] biomedical and biological data are routinely continuous. By neglecting to adequately address the ramifications of discretization, researchers unknowingly may lose information such as interactions and dependencies between variables and impact the learned structure. Unfortunately, there is no concensus on a standard procedure for discretization. Consequently, it is still an unresolved research question as how to best handle continuous data* (Fu and Tsamardinos [7]).

Jensen and Nielsen [8] provide an excellent summary of the most common parameter learning approaches. Extending those approaches Kampa [9] suggests an additional approach, parameter tying. Rather than learning the values in the conditional probability table of a discretized variable separately, he suggests to model its distribution as a certain (known) distribution with few unknown parameters and learning those parameters. This method achieves part of our goal, namely linking conditional probability entries of neighboring states to each other and thus making sure all entries are well defined, even for small amounts of training data. This approach requires advance knowledge of the distribution shape of the variables, which is not always available. Nevertheless, this may be a viable option for many applications that deserves further investigation. A preliminary literature search did not turn up any published work on the topic, although it is likely that such research exists. Thus a more thorough literature search is in order.

In contrast, fuzzy approaches to soft discretization do not require any advance knowledge of the distributions and are discussed next.

1.2.1 Fuzzy Bayesian Networks

Several research groups recently proposed to use fuzzy set theory in connection with Bayesian networks to create different types of *Fuzzy Bayesian Networks*, see Baldwin and Di Tomaso [1], Tang and Liu [15], Pan and Liu [13] and Fogelberg et al. [5].

The approach taken by Baldwin and Di Tomaso [1] is most relevant for the work discussed here for two reasons. First of all, soft discretization of continuous variables is their primary reason for using fuzzy set theory. Secondly, the structure of their approach is very similar to our training approach. Both of them perform preprocessing of the training cases to convert them to soft evidence, then train the Bayesian Network from the soft evidence. As such the fuzziness is implemented explicitly *outside* of the Bayesian Network.

In contrast, Fogelberg et al. [5] seek to generalize Bayesian Networks to Fuzzy Bayesian Networks that replace discrete states with fuzzy states *throughout* the network. Thus in their work the fuzziness is integrated completely *inside* the Bayesian Network. They start out with already discrete states (thus *not* implementing explicit soft discretization), which are represented by fuzzy states to increase robustness of the system. The emphasis of their work is on generalizing belief propagation in the network to apply to those fuzzy states. Pan and Liu [13] and Tang and Liu [15] also focus on developing a framework that integrates fuzzy components *inside* the Bayesian Network and are their work is thus also less closely related to our approach than the work by Baldwin and Di Tomaso.

All fuzzy approaches start with a *membership function*. As most of the fuzzy approaches listed above focus on the theory required to extend Bayesian Networks to fuzzy Bayesian Networks, most of them spend little time on discussing how to choose membership functions for this purpose. For example, Baldwin and Di Tomaso [1] give the example of triangular membership functions and only mention that fuzzy sets with a different shape than a triangular one, could be used. Tang and Liu [15] use trapezoidal membership functions in their example. Fogelberg et al. [5] discuss constraints for membership functions in more detail, but do not make specific recommendations.

Of course, more research on the choice of membership functions for other applications - *not* related to Bayesian Network - can be found in the general fuzzy set theory literature, see for example Boston [2] and Koehl and Zeng [10].

Our soft discretization approach starts out with a probability density function (spread function) to model how much a continuous variable should be spread to neighboring regions. Section 5 shows that the probability density function, when convolved with a rectangular function representing the discretization interval, results in a membership function. This generates a probability-based set of membership functions that may be

particularly suitable for the purpose of soft discretization - for Bayesian Network and maybe even beyond.

1.2.2 Other Soft Discretization Approaches for BNs

While most soft discretization approaches for Bayesian Networks in literature are based on fuzzy approaches (and thus are based on membership functions), there are a few other approaches. Lin et al. [12] adds a Gaussian Mixture model (GMM) node for every continuous node in a Bayesian Network. The GMM node converts the continuous value to soft discrete evidence. The parameters of the GMM node are obtained through training of the resulting Bayesian Network from training data. Details of the approach (e.g. number of terms of each GMM node) are not provided, but it appears to outperform hard discretization models in experiments.

Yu at al. [17] mention the use of soft discretization to preprocess their data, but unfortunately the method used for soft discretization is not provided. In their experiments hard discretization actually outperformed soft discretization.

1.3 Target Applications

The research reported here was motivated by the down-scaling problem for precipitation forecast in monsoon areas, brought to our attention by Annalisa Bracco (Assistant Professor, Earth and Atmospheric Science, Georgia Tech). In this problem we have 40 years of *observed* historical precipitation data available on a *high resolution* grid, as well as a current *forecast* on a *low resolution* grid resulting from an atmospheric general circulation model (see Bracco et al. [3] and Kucharski et al. [11] for the atmospheric general circulation model). The goal is to train a Bayesian Network using the historical data and to use the trained model to convert the low resolution forecast from the atmospheric model into a high resolution forecast. A decent high resolution forecast would be very useful for farmers in India and neighboring regions to help them plan when to best plant their crops.

While developing different Bayesian Network (BN) models for this problem we noticed several properties that are shared by many other real-world problems in the natural sciences and medicine. Namely, a system with the following properties is most likely to benefit from soft discretization:

1. The underlying mechanism of the system to be modeled is continuous. That means that not only are the values of the variables continuous, but the relationships *between* the variables are also continuous, i.e. the mappings from parent states to child states are continuous with the exception of some inherent system uncertainty.

2. The variable interaction *cannot* be modeled by a Gaussian distribution, which is the only continuous node type currently available in BN modeling tools. Thus using a BN with continuous nodes is not an option and a BN with discrete nodes must be used.

3. The variables are highly connected, i.e. many variable have at least three or four parents, requiring a large number of entries in the conditional probability tables (CPTs).

4. The number of data sets for training, though apparently large (e.g. daily data for 40 years in our case), is not sufficient to fill all entries of the conditional probability tables, unless an extremely coarse resolution is used for each variable. For example, there are usually some very rare state combinations that may not show up in the training cases, but may occasionally show up in test cases.

For a system with the properties listed above, the *standard* approach would be the following. One would define a BN with discrete nodes, using a very coarse discretization for each variable (because of the lack of sufficient training data) and discretize all training data accordingly before training the discrete BN.

However, due to Property 1 above, not only are the values of the variables continuous, but the relationships *between* the variables are also continuous, i.e. the mappings from parent states to child states are continuous with the exception of some inherent system uncertainty. We are seeking here to leverage this continuity by deriving a BN model directly from the continuous data sets using soft discretization. The idea is that the *location* of a continuous value *within* a discrete interval should matter during training. Soft discretization

does take this location into account. Furthermore, the fact that neighboring states in a continuous system are related to each other allows us to fill otherwise empty CPT entries naturally through soft discretization.

Other potential benefits of using soft discretization include increased robustness of the resulting model, especially reduced sensitivity on how interval boundaries are chosen.

2 Notation

2.1 Basics

Throughout this report we use the following notation for any scalar variable, X:

- X^{cont} is a continuous variable with values x.
- X^{discr} is the discrete counterpart of X^{cont} with values $x_i, i = 1, \ldots, N^X$.
- N^X denotes the number of discrete states of X^{discr}.
- Whenever it is clear from the context, we drop the superscripts $cont$ and $discr$, to cut down on indices. For example $X = x_i$ refers to a discrete, $X = x$ to a continuous variable.
- $\{B_1^X, \ldots, B_{N^X+1}^X\}$ denotes the set of $(N^X + 1)$ boundaries of discrete variable X, where we require that the first and last boundary represent minus infinity and infinity, respectively:

$$B_1^X = -\infty \quad \text{and} \quad B_{N^X+1}^X = \infty. \tag{1}$$

  (More about this requirement in Subsection 7.3.)
  The ith state of X, $X = x_i$, therefore represents the interval

$$[B_{i,low}^X, B_{i,up}^X) = [B_i^X, B_{i+1}^X)$$

  of continuous variable X.

Throughout this report there are many more functions or variables that are associated with a variable, say X. This association is generally indicated by the variable name as *superscript*, for example N^X or B^X above.

2.2 Parent Vector Notation

Let us first consider a node Y with only one parent X, i.e. $X \longrightarrow Y$.

- $P(x_i, y_j)$ denotes the joint probability of X and Y.
- $P(y_j|x_i)$ denotes the Conditional Probability Distribution (CPD) of Y given X.

For the multi-parent case we use the vector notation, $\mathbf{X}$, whenever possible to denote a *set* of parents of node Y. Likewise vector $\mathbf{x}_i$ then denotes the state *combination* of the parents, i.e. one state for each parent, where the single index i is used to enumerate all different state combinations. Thus i goes from 1 to the *total number of different state combinations* of the parents, $i = 1, \ldots,$ # state combinations. Thus the joint and conditional probabilities can be expressed in a very similar way:

- $P(\mathbf{x}_i, y_j)$ denotes the joint probability of Y and parent set $\mathbf{X}$.
- $P(y_j|\mathbf{x}_i)$ denotes the Conditional Probability Distribution (CPD) of Y given parent set $\mathbf{X}$.

In some instances the parent variables must be listed individually. In those cases M denotes the number of parents, and Roman numerals are used as superscripts to indicate each parent variable. Thus we have the following correspondences to the vector notation:

- Set of parents: $\mathbf{X} = \{X^I, X^{II}, \ldots, X^M\}$
- Index of a parent combination: $i = \{i^I, i^{II}, \ldots, i^M\}$

- Set of states for a parent: $\mathbf{x}_i = \{x_{i^I}^I, x_{i^{II}}^{II}, \ldots, x_{i^M}^M\}$.

Since these individual indices become quite messy very quickly, the vector notation is used wherever possible.

Note that subscripts always denote states, while superscripts denote which variable is meant. For example, x_i denotes the ith discrete state of discrete variable X. x^m denotes a continuous value of the mth parent, X^m. x_i^m denote the ith discrete state for the mth parent, X^m.

2.3 Acronyms

We use just a few acronyms:

- PDF = Probability Density Function
- MF = Membership Function (as used in fuzzy set theory)
- CPD = Conditional Probability Distribution
- CPT = Conditional Probability Table (which represents a discrete CPD).
- MLE = Maximum Likelihood Estimation

3 Standard Approach For Training Discrete BNs From Continuous Data – Hard Discretization and Maximum Likelihood Estimation

This section first summarizes the standard method of using hard discretization combined with Maximum Likelihood Estimation (MLE) of the conditional probabilities. Then it describes how the soft discretization method with modified Maximum Likelihood Estimation can be derived from it.

The standard approach to train discrete networks from continuous data is the following two-step approach:

1. Use hard discretization to convert all continuous test cases to (hard) discrete cases.
2. Use Maximum Likelihood Estimation to calculate the discrete conditional probability tables from the discrete cases.

Given discrete and disjoint intervals for each variable, hard discretization maps each continuous value of a variable, x^{cont}, to exactly one discrete state. Once all continuous variables are mapped to discrete states Maximum Likelihood Estimation, which is the primary method for training Bayesian Networks with discrete nodes from complete data (no missing data), can be applied. Although a description of MLE can be found in any standard textbook we briefly summarize this method here using a formulation that helps to understand the connection to the soft discretization method.

Maximum Likelihood Estimation is based on two assumptions (Jensen and Nielsen [8]):

- *Global Independence* The parameters for the various variables are independent. This means that we can modify the tables for the variables independently.
- *Local Independence* The uncertainty of the parameters for different parent configurations are independent. For example, for a child node A with two parents B, C, and $b, b^{'}$ and $c, c^{'}$ denoting different states of B and C, respectively, the uncertainty on $P(A|b, c)$ is then independent of the uncertainty on $P(A|b^{'}, c^{'})$, and the parameters for the two distributions can be modified independently.

Based on these two assumptions the joint probability $P_{MLE}(y_j, \mathbf{x}_i)$ can simply be estimated as the frequency of simultaneous occurrence of states y_j and $\mathbf{x}_i$ in the training data:

$$P_{MLE}(y_j, \mathbf{x}_i) = \frac{\#\text{ cases with } Y = y_j \text{ and } \mathbf{X} = \mathbf{x}_i}{\#\text{ cases}} \quad (2)$$

Equation (2) is the central equation for Maximum Likelihood Estimation. From it we get

$$P_{MLE}(\mathbf{x}_i) = \sum_j P_{MLE}(y_j, \mathbf{x}_i) \;=\; \frac{\#\text{ cases with } \mathbf{X} = \mathbf{x}_i}{\#\text{ cases}}$$

Using Bayes' theorem we can then estimate the Conditional Probability Distribution (CPD) for each node by the following well known formula:

$$P_{MLE}(y_j|\mathbf{x}_i) \quad = \quad \frac{P_{MLE}(y_j, \mathbf{x}_i)}{P_{MLE}(\mathbf{x}_i)} = \frac{\#\text{ cases with } Y = y_j \text{ and } \mathbf{X} = \mathbf{x}_i}{\#\text{ cases with } \mathbf{X} = \mathbf{x}_i}.$$

The central concept to take from this section is that the key term to calculate is the joint distribution, $P_{MLE}(y_j, \mathbf{x}_i)$, in Equation (2). The estimates of the desired CPD entries follow directly as shown above. Likewise for the soft discretization approach the central question is also the calculation of the joint distribution, $P_{soft}(y_j, \mathbf{x}_i)$, and the CPD values follow from that in the same manner as above.

4 Soft Discretization Approach to Training

The soft discretization approach is based on the same assumptions of Global Independence and Local Independence as the Maximum Likelihood Estimation and follows the same order, namely estimating the joint distribution first, $P(y_j, \mathbf{x}_i)$, and then calculating the CPDs for each node from that. The difference is that the training data is interpreted as soft evidence, rather than hard evidence.

This means that we have to

1. Develop a soft discretization procedure, which converts each training case into soft evidence
2. Modify the Maximum Likelihood Estimation to accept soft evidence.

Our approach for the soft discretization is outlined in Figure 1, where we consider the discretization of a single case (e.g. a single training case) in order to obtain the CPD of node Y with parent set $\mathbf{X}$. We denote this case as *Case* k. It represents in this context one continuous instantiation, $(\hat{\mathbf{x}}, \hat{y})$, of the variables, $\mathbf{X}$ and Y. Note that for continuous functions we refer to $\hat{\mathbf{x}}, \hat{y}$ explicitly, while for discrete functions we refer to it by index k (for Case k). Let us go through Figure 1 layer by layer.

Layer 1: The conditional probability $P_k(y_j|\mathbf{x}_i)$ listed in Layer 1 is the target quantity that we want to calculate. It represents the conditional probability table for node Y.

Layer 2: From Section 3 we know that it is sufficient to find the discrete joint distribution, $P_k(\mathbf{x}_i, y_j)$, shown in Layer 2, since one can calculate $P_k(y_j|\mathbf{x}_i)$ from it using Bayes' theorem.

Layer 3: Layer 3 lists the *continuous* joint probability density function, $p_{(\hat{\mathbf{x}},\hat{y})}(\mathbf{x}, y)$, that represents the *spread model* for variables Y and $\mathbf{X}$. A typical shape of a spread model is shown in Figure 2 which displays a two-dimensional spread model, i.e. a model for the case where Y has only one parent, X. Spread models are discussed in detail in Subsection 4.1. The key facts to note for now are that the spread model is represented by a *continuous probability density function* and that by integrating that function over the discrete intervals we obtain the *discrete joint probability distribution* in Layer 2.

Layer 4: To make life easier we decouple the problem of finding a good spread model for variables Y and $\mathbf{X}$ by choosing an individual spread model for each variable, i.e. for Y, X^I, X^{II}, ..., X^M. The spread model for Y, for example, is denoted as $p_{\hat{y}}(y)$, which means it depends only on continuous value $\hat{y}$. Each function $p_{\hat{y}}(y)$, $p_{\hat{x}^I}(x^I)$, etc., is chosen as probability density function and suitable candidates are discussed in Subsection 4.1.1. The combined spread model for $\mathbf{X}$ and Y is obtained as product of the individual spread models (see Section 4.1.2 for details). Since all one-dimensional spread models are chosen as probability density functions, their product is automatically a probability density function for the joint probability, i.e. it is a proper spread model as desired in Layer 2.

Layer	Probability Type	Description
1	$P_k(y_j \mid \mathbf{x}_i)$ for all i, j	Conditional probability distribution of Node Y for case k
	⇑ Bayes' Theorem	
2	$P_k(\mathbf{x}_i, y_j)$ for all i, j	Discrete joint probability distribution of Y and $\mathbf{X}$ for case k
	⇑ Integration over discrete intervals	
3	$p_{(\hat{\mathbf{x}},\hat{y})}(\mathbf{x}, y)$	Continuous joint probability density for case k = Spread Model for Y and $\mathbf{X}$ evaluated at case k
	⇑ Product – see Equation (6)	
4	$p_{\hat{y}}(y)$ $p_{\hat{x}^I}(x^I)$ $p_{\hat{x}^{II}}(x^{II})$ ⋮ $p_{\hat{x}^M}(x^M)$	Continuous one-dimensional probability density function = Individual Spread Model for each variable at case k

Figure 1: Overview of soft discretization approach for variable Y with parent set $\mathbf{X}$ for a single case k. Case k represents a set of continuous values, $(\hat{\mathbf{x}}, \hat{y})$, for $\mathbf{X}$ and Y.

While our method was explained above following the layers in Figure 1 from top to bottom, the calculations proceed in the opposite direction, from Layer 4 up to Layer 1. This means that, given Case k with continuous values, $(\hat{\mathbf{x}}, \hat{y})$, we first evaluate the individual spread models in Layer 4, then multiply the results to gain the joint spread model in Layer 3, then integrate that function over discrete intervals to get the discrete joint probability distribution in Layer 2, to finally obtain the discrete conditional probability distribution in Layer 1.

Furthermore, it will be seen that the integration of function $p_{(\hat{\mathbf{x}},y)}(\mathbf{x}, y)$ actually decouples into integrating the individual functions $p_{\hat{y}}(y)$ directly. This process results in a set of weights that can then be multiplied to yield the discrete joint probability distribution in Layer 2. Finally, we ultimately want to train the network parameters not from a single case, but a large number of cases. The place to combine the information from the different cases is Layer 2, where we take the average over the joint probabilities from all the different cases.

Taking these considerations into account, the soft discretization process is described in the following subsections as follows:

1. Define a one-dimensional spread model (Probability Density Function), that models how far and in what shape a continuous value should be spread to its neighbors (Subsection 4.1.1).

2. Discretize the one-dimensional spread model through integration over the discrete intervals. This results in a set of weights (Subsection 4.2).

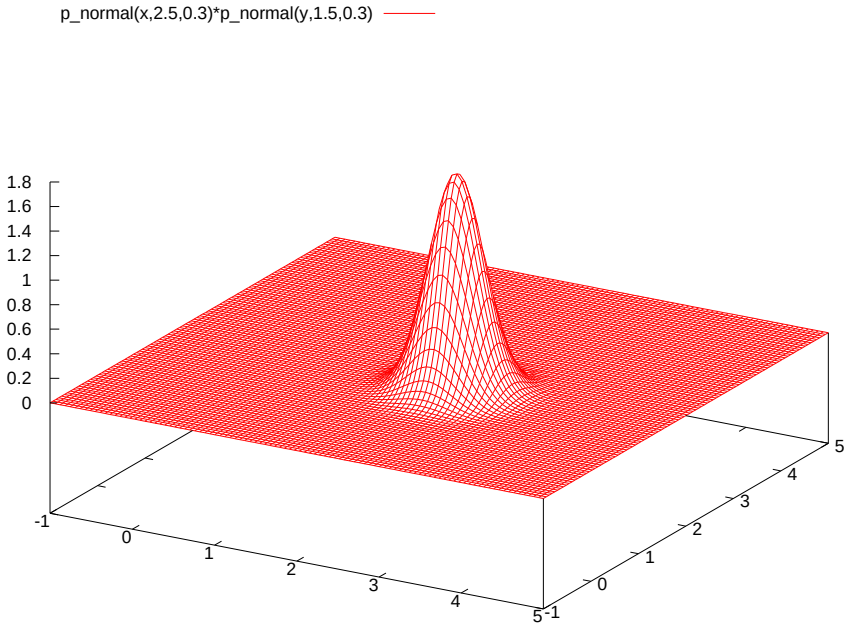

Figure 2: Example of a two-dimensional spread model for a node Y with only one parent X for case of $(\hat{x}, \hat{y}) = (2.5, 1.5)$. This model is the product of two Gaussian density functions in x and y with means 2.5 and 1.5, respectively, and with deviation $\sigma = 0.3$.

3. Calculate the joint probability approximation $P_k(\mathbf{x}_i, y_j)$ for case k by multiplying the appropriate weights (Subsection 4.2.2).

4. Combine the discrete joint distributions $P_k(\mathbf{x}_i, y_j)$ from all cases to calculate the overall conditional probability distribution $P_{soft}(y_j|\mathbf{x}_i)$ of discrete variables X and Y (Subsection 4.3).

The final values of $P_{soft}(y_j|\mathbf{x}_i)$ are the desired CPD values and constitute the output of the modified MLE algorithm.

4.1 Defining a Spread Model (Probability Density Function)

4.1.1 Spread Function for a Single Variable

This subsection discusses how to choose the spread model for a single variable, X, with values x. *Defining the spread model for each variable separately assumes that how the value of one variable is spread is independent of the values of the other variables.* This appears to be a very reasonable assumption and strongly simplifies our approach.

Given one continuous value, $\hat{x}$, to be discretized, we want to *spread* its effect in some way to its neighboring areas. The spread function, $p_{\hat{x}}(x)$, formalizes how far and in what shape the spreading occurs in a very intuitive and graphical way. p is a function in x that depends on $\hat{x}$ as a parameter, which we denote as $p_{\hat{x}}(x)$.

We already know that p should be a probability density function. What other properties do we want for the spread function?

1. The influence should be largest right at the continuous value, so $p_{\hat{x}}(x)$ should peak at $\hat{x}$.

2. $p_{\hat{x}}(x)$ should decline monotonously while moving away from $\hat{x}$ in either direction. (Rectangular functions are allowed, since p does *not* have to decline *strictly* monotonously.)

3. We do not want negative effects, so we want $p_{\hat{x}}(x) \geq 0$ for all x.

4. Since p is used to calculate weights, it is convenient to choose it such that its integral equals 1,

$$\int_{-\infty}^{\infty} p_{\hat{x}}(x)dx = 1.$$

That way the weights are automatically normalized to 1 later on.

Since p is chosen as probability density function, Properties 3 and 4 above are automatically satisfied.

To achieve the peak at $\hat{x}$, we can choose a function $p(x)$ with peak at 0 and shift it by $\hat{x}$:

$$p_{\hat{x}}(x) = p(x - \hat{x}).$$

This way we obtain a family of spread functions with identical shape for all $\hat{x}$, i.e. all continuous values are spread in the same way. (We will see later ways to make the spread function dependent on $\hat{x}$ if so desired, for example by using a free variance parameter that depends on $\hat{x}$.)

Candidates of suitable one-dimensional probability density functions include:

1. **Normal (Gaussian) Density Function**
The probability density function (Figure 3) of the normal – also known as Gaussian – distribution is defined as

$$p_{Gauss}(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}}\, exp^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3}$$

where μ denotes the mean, which is also the peak, and σ the standard deviation. The normal distribution is a natural choice, since uncertainty in a variable can often be modeled fairly well by the Bell-shaped density function of the Gaussian distribution.

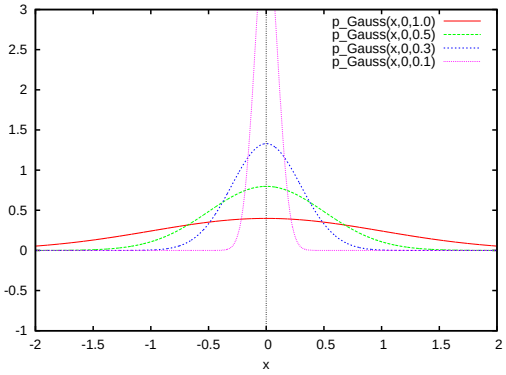

Figure 3: Gaussian PDF for $\mu = 0$ and $\sigma = 1.0, 0.5, 0.3$ and 0.1

We can thus choose as spread function

$$p(x) = p_{Gauss}(x; 0, \sigma),$$

which corresponds to

$$p(x - \hat{x}) = p_{Gauss}(x - \hat{x}; 0, \sigma) = p_{Gauss}(x; \hat{x}, \sigma),$$

where the variance $\sigma = \sigma(\hat{x})$ is a free parameter that can be chosen dependent on $\hat{x}$ if desired. Choosing values for σ is a trade-off. If σ is chosen very small, then there is little difference to hard discretization. If σ is chosen very large, then the soft discretization washes out more and more information in the training and test data. In fact, for $\sigma \to \infty$, *any* continuous value is mapped *equally* to all discrete states, thus no information is left. More discussion on choosing σ can be found in Section 7.2.

2. **Piece-Wise Straight Functions**
   Piece-wise straight functions can be used to emulate the rough shape of the normal density function or to implement a different shape. Piece-wise straight functions are easier to integrate, thus simplifying the calculation of the weights later on and may thus be a preferred option if computational complexity is an issue.

   Choices include rectangular and triangular functions (Figure 4) and trapezoidal functions.

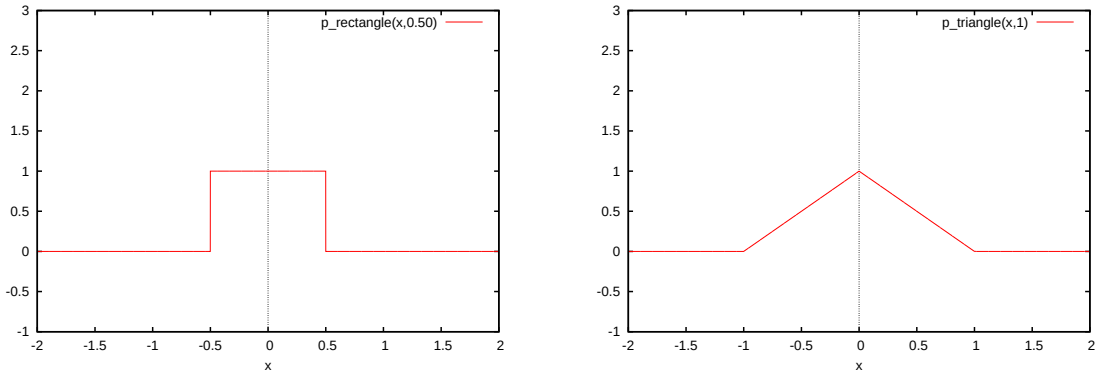

Figure 4: Rectangular (left) and triangular (right) probability density functions.

   Any such function, $p(x)$, must be shifted to have its highest value at 0 and scaled to have an area of one. Its shifted version, $p(x - \hat{x})$, can then be used as spread function.

3. **Dirac Delta Function**
   To emulate hard discretization we use the normal density function and let the variance converge toward zero, i.e. we completely eliminate any spread in the model. The result is the well known Dirac delta function, $\delta(x)$:

$$\delta(x) = \lim_{\sigma \to 0} \; p_{Gauss}(x; 0, \sigma). \tag{4}$$

   The Dirac delta function is a generalized function with an area of 1. Its shifted version, $\delta(x - \hat{x})$, is a generalized function with the values

$$p(x - \hat{x}) = \delta(x - \hat{x}) = \begin{cases} \infty & \text{for } x = \hat{x} \\ 0 & \text{otherwise} \end{cases} . \tag{5}$$

Other types of spread functions are possible and the precise effect of the shape of the spread function on the CPD entries remains to be investigated.

4.1.2 Joint Probability as Product of One-Dimensional Functions

As mentioned in Section 4.1.1, we assume that how the value of one variable is spread is independent on the values of the other variables. This allows us to define the combined spread model, $p_{\hat{\mathbf{x}},\hat{y}}(\mathbf{x}, y)$ simply as product of the individual spread models, namely

$$p_{\hat{\mathbf{x}},\hat{y}}(\mathbf{x}, y) \quad = \quad p_{\hat{x}^I,\ldots,\hat{x}^M,\hat{y}}(x^I, \ldots, x^M, y) \quad = \quad p_{\hat{y}}(y) \cdot \prod_{m=I}^{M} p_{\hat{x}^m}(x^m) \tag{6}$$

Figure 2 shows a two-dimensional spread model, obtained by multiplying two Gaussian density functions. It shows clearly for the continuous value pair, here $(\hat{x}, \hat{y}) = (2.5, 1.5)$, the level of influence in its surrounding areas.

4.2 Discretizing the Spread Model (Obtaining the Weights)

Given the continuous joint probability density function $p_{(\hat{\mathbf{x}},\hat{y})}(\mathbf{x}, y)$ we can now calculate the probability function $P_k(\mathbf{x}_i, y_j)$ for the discrete states (i, j) by integration.

Recall that i here denotes a state combination of the parent variables, i.e. it stands for an index set $\{i^I, i^{II}, \ldots, i^M\}$ for parents $\{X^I, X^{II}, \ldots, X^M\}$. To avoid dealing with all those messy indices, we first look at the special case of only one parent and then generalize the equations afterward.

4.2.1 Single Parent Case

For the single parent case X is a scalar variable and i indicates its scalar state, x_i. $P_k(X = x_i, Y = y_j)$ is obtained from p through integration over the boundaries of states x_i and y_j, namely

$$P_k(X = x_i, Y = y_j) = \int_{B^X_{i,low}}^{B^X_{i,up}} \int_{B^Y_{j,low}}^{B^Y_{j,up}} p_{(\hat{x},\hat{y})}(x, y)\; dy\; dx. \tag{7}$$

Since $p_{(\hat{\mathbf{x}},\hat{y})}(x, y)$ was chosen as product of independent functions in x and y the integral decouples as follows:

$$\begin{aligned} P(X = x_i, Y = y_j) &= \int_{B^X_{i,low}}^{B^X_{i,up}} \int_{B^Y_{j,low}}^{B^Y_{j,up}} p_1(x - \hat{x})\; p_2(y - \hat{y})\; dy\; dx, \\ &= \underbrace{\left(\int_{B^X_{i,low}}^{B^X_{i,up}} p_1(x - \hat{x})\; dx \right)}_{w_i^X} \cdot \underbrace{\left(\int_{B^Y_{j,low}}^{B^Y_{j,up}} p_2(y - \hat{y})\; dy\; dx \right)}_{w_j^Y} \\ &= w_i^X \cdot w_j^Y, \end{aligned}$$

where p_1 denotes the one-dimensional spread model function chosen for X and p_2 the spread model function chosen for Y. w_i^X is the weight assigned for $\hat{x}$ to the ith discrete state of X,

$$w_i^X = \int_{B^X_{i,low}}^{B^X_{i,up}} p_1(x - \hat{x})\; dx. \tag{8}$$

Likewise w_j^X is the weight assigned for $\hat{y}$ to the jth state of Y,

$$w_j^Y = \int_{B^Y_{j,low}}^{B^Y_{j,up}} p_2(y - \hat{y})\; dy. \tag{9}$$

In summary, for each set of continuous values, $(\hat{x}, \hat{y})$, we can calculate one set of weights, w_i^X, for X from $\hat{x}$, and another set, w_j^Y, for Y from $\hat{y}$. The joint distribution results as product of the weights,

$$P(x_i, y_j) = w_i^X\; w_j^Y.$$

4.2.2 General Case

Just as the weights in X and Y can be calculated independently in the single-parent case, the same procedure can be applied in the case of several parents. Let us denote the $M \geq 1$ parents of Y as $X^1, \ldots, X^M$, and consider the soft discretization of one set of continuous values, $(\hat{x}^1, \ldots, \hat{x}^M, \hat{y})$. We obtain a set of weights, w_j^Y, for the child based on $\hat{y}$, and one set of weights, $w_{i^m}^{X^m}$, for *each* parent, X^m, based on $\hat{x}^m$. Namely, the weights for parent variable, X^m, are defined as

$$w_{i^m}^{X^m} = \int_{B^{X^m}_{i^m,low}}^{B^{X^m}_{i^m,up}} p_m(x^m - \hat{x}^m)\; dx^m, \tag{10}$$

where p_m denotes the spread model chosen for X^m. The weights for Y are defined as

$$w_j^Y = \int_{B_{j,low}^Y}^{B_{j,up}^Y} p_{M+1}(y - \hat{y})\, dy, \tag{11}$$

where p_{M+1} denotes the spread model chosen for Y.

The joint distribution results as the product of the weights,

$$P_k(x_{i^1}^I, \ldots, x_{i^M}^M, y_j) = w_j^Y \cdot \prod_{m=1}^{M} w_{i^m}^{X^m}. \tag{12}$$

Equation (12) also applies for the case of no parents, $M = 0$, in which case it degenerates to

$$P_k(y_j) = w_j^Y.$$

Interpretation of the weights: For a single variable, say X, its set of weights, w_j^X, is the soft evidence representation of its continuous value $\hat{x}$. In fact the weights can be interpreted as the probability of the discrete variable X^{discr} taking state x_i , if we know that the continuous variable takes value $\hat{x}$,

$$w_j^X \;=\; P(X^{discr} = x_j | X^{cont} = \hat{x}), \tag{13}$$

given the uncertainty model defined by the one-dimensional spread model $p_{spread}(x)$.

Since the weights for each variable are calculated separately, different functions, $p_i(x)$, can be used for each variable. Furthermore, it is very easy to deal with networks that have mixed nodes – some discrete by nature, some continuous. In fact one can freely choose for each node in the network one of these three discretization types:

1. Node is already discrete;
2. Node is continuous - want hard discretization;
3. Node is continuous - want soft discretization.

The first two types simply yield weights that take values of only 0 or 1, while for the third type the weights are calculated through soft discretization. Thus all three types of nodes can be combined freely in the network and the algorithm automatically treats them accordingly.

4.2.3 Examples for Weight Calculation

Example 1: Weights for Normal Distribution
For a variable, X, that uses the normal density functions as its one-dimensional spread model the weights are defined as

$$w_i^X \quad = \quad \int_{B_{i,low}^X}^{B_{i,up}^X} p_{normal}(x - \hat{x}; 0, \sigma) \quad dx,$$

where $\sigma = \sigma^X(\hat{x})$ can be defined as function of parameter $\hat{x}$. Denoting as $P_{normal}^{CDF}(x; \mu, \sigma)$ the cumulative normal distribution, i.e.

$$P_{normal}^{CDF}(x; \mu, \sigma) = \int_{-\infty}^{x} p_{normal}(\xi; \mu, \sigma)\, d\xi,$$

we get

$$w_i^X = P_{normal}^{CDF}(B_{i,up}^X; \hat{x}, \sigma) - P_{normal}^{CDF}(B_{i,low}^X; \hat{x}, \sigma),$$

where $P_{normal}^{CDF}(x; \mu, \sigma)$ can be calculated from the well-known error function, $erf(x)$, using the formula

$$P_{normal}^{CDF}(x; \mu, \sigma) = \frac{1}{2}\left(1 + erf\left(\frac{x - \mu}{\sigma * \sqrt{2}}\right)\right). \tag{14}$$

The error function, $erf(x)$, is available in most standard languages from C to Matlab.

Example 2: Weights for Dirac delta function
Using the shifted Dirac delta function as spread model for variable X leads to weights with values of either 0 or 1. In fact it can be shown that using the Lebesque integral we get

$$w_i^X = \int_{B_{i,low}^X}^{B_{i,up}^X} \delta(x - \hat{x})\, dx \;=\; \begin{cases} 1 & \text{if } \hat{x} \in \left[B_{i,low}^X, B_{i,high}^X\right] \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

which is simply the weight assignment for hard discretization. Of course one would *never* integrate the Dirac delta function in practice to calculate hard discretization, but instead simply use the right most expression of Equation (15) to calculate those weights. Nevertheless, using the Dirac function in this derivation shows how hard discretization can be described by the same formalism and that it can simply be interpreted as the limiting case of soft discretization.

4.3 Combining the joint distributions from all cases to calculate CPTs for all nodes

Given N training cases, we have N sets of continuous variables, $(\hat{x}_k^1, \ldots, \hat{x}_k^M, \hat{y}_k)$. Each continuous variable yields one set of weights, $w_{j,k}^Y$ and $w_{i^m,k}^{X^m}$, which combined provide the joint distribution

$$P_k(\mathbf{x}_i, y_j) = P_k(x_{i^1}, \ldots, x_{i^m}, y_j) = w_{j,k}^Y \cdot \prod_{m=1}^{M} w_{i^m,k}^{X^m},$$

where we use the vector notation, $\mathbf{x}_i$, defined in Section 2.2 to denote a state combination for the parents.

Now we can finally combine the estimates obtained from all the different cases by simply taking the average of their joint probabilities. Namely, the combined joint probability for soft discretization, $P_{soft}(\mathbf{x}_i, y_j)$, is defined as:

$$P_{soft}(\mathbf{x}_i, y_j) = \frac{1}{N}\sum_{k=1}^{N} P_k(\mathbf{x}_i, y_j) \;=\; \frac{1}{N}\sum_{k=1}^{N} w_{j,k}^Y \cdot \prod_{m=1}^{M} w_{i^m,k}^{X^m}. \tag{16}$$

The definition of $P_{soft}(y_j, \mathbf{x}_i)$ in Equation (16) is the central definition for the soft discretization. $P_{soft}(y_j, \mathbf{x}_i)$ takes the role here that $P_{MLE}(y_j, \mathbf{x}_i)$ (defined in Equation (2) of Section 3) plays for Maximum Likelihood Estimation.

Now that we have the combined joint probability distribution, $P_{soft}(\mathbf{x}_i, y_j)$, we can follow the simple steps for the MLE calculation in Section 3. Namely, we first calculate $P_{soft}(\mathbf{x}_i)$ by summation,

$$P_{soft}(\mathbf{x}_i) = \sum_j P_{soft}(\mathbf{x}_i, y_j)$$

and finally obtain the desired conditional probability values of each node by Bayes' theorem:

$$P_{soft}(y_j|\mathbf{x}_i) = \frac{P_{soft}(y_j, \mathbf{x}_i)}{P_{soft}(\mathbf{x}_i)}.$$

5 Connection to Fuzzy Set Theory - Spread Function vs. Membership Function

Let us consider the soft discretization of a single variable, X, and let $p(x)$ denote the probability density function (spread model) chosen for its soft discretization. Let us define the *weight function*, $F_i(\hat{x})$, for the ith interval as follows

$$F_i(\hat{x}) = \int_{B^X_{i,low}}^{B^X_{i,up}} p(x - \hat{x})dx.$$

We call $F_i(\hat{x})$ a weight function, because it represents how the weight for the ith interval changes with the value of $\hat{x}$. $F_i(\hat{x})$ may not always be easy to calculate in closed form, but if we can express it in closed form the weights simply follow by evaluation of F_i at the continuous value, $w_i^X = F_i(\hat{x})$.

Comparison shows that the weight function $F_i(\hat{x})$ takes exactly the role of the *membership function* defined in fuzzy set theory. However, **the difference is that in our soft discretization method we choose the shape of the spread model (density function) and $F_i(\hat{x})$ follows through integration. In contrast, in fuzzy set theory one chooses the shape of $F_i(\hat{x})$ directly for each discrete state i.**

The membership function $F_i(\hat{x})$ can actually be described as the *convolution* of $f(x, \hat{x})$ with a rectangular clip function representing the ith interval. Let us denote such a rectangular clip function as $1_{[a,b]}(x)$ for interval $[a, b]$:

$$1_{[a,b]}(x) = \begin{cases} 1 & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}.$$

Then we can rewrite $F_i(\hat{x})$ as follows:

$$\begin{aligned} F_i(\hat{x}) &= \int_{-\infty}^{\infty} p(x - \hat{x}) \cdot \underbrace{1_{[B^X_{i,low}, B^X_{i,up}]}(x)}_{g(x)} \; dx \\ &= \int_{-\infty}^{\infty} p(x - \hat{x}) \cdot g(x) \; dx \;=\; (p * g)(\hat{x}). \end{aligned}$$

In summary the membership function $F_i(\hat{x})$ used in fuzzy set theory for soft discretization is the convolution $(p * g)(\hat{x})$ of the probability density function, $p(x)$, from the soft discretization spread model with the rectangular function representing the discretization interval, $g(x) = 1_{[B^X_{i,low}, B^X_{i,up}]}(x)$.

5.1 Sample Correspondences

To get an intuitive feeling for the relationship between the two types of functions, Figures 5, 6 and 7 show several probability density functions with their corresponding membership functions for the sample interval $[a, b] = [1, 2]$.

Figure 5 shows rectangular probability density functions and the corresponding member functions. Note that the rectangular spread function shown in Figure 5(a) corresponds to a trapezoidal membership function, which is a very common shape for a membership function. Let us consider the width, W, of the peak of the trapezoidal membership function. Denoting by $[-A, A]$ the interval for which the rectangular density function is one, and by $[a, b]$ the discretization interval, then the peak of the trapezoidal membership function has a width of

$$W = \|2 * A - (b - a)\|.$$

Thus for the case shown in Figure 5(a) it is $W = 1$. Figure 5(b) shows the special case where $W = 0$ and thus the membership function becomes triangular. Figure 5(c) shows the transition from $A = 1$ down to $A = 0.25$. For $A \to 0$ the PDF converges to the Dirac delta function and the membership function converges to a rectangle, which corresponds to hard discretization. Note how the non-zero part of the trapezoidal membership function on the right of Figure 5 decreases continuously for decreasing A, approaching the discretization interval $[a, b] = [1, 2]$. The peak width, W, decreases to a single point and then increases

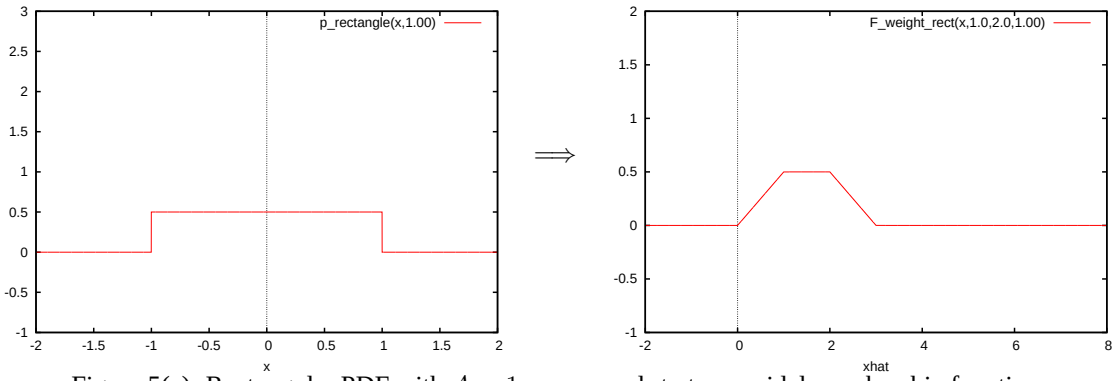

Figure 5(a): Rectangular PDF with $A = 1$ corresponds to trapezoidal membership function

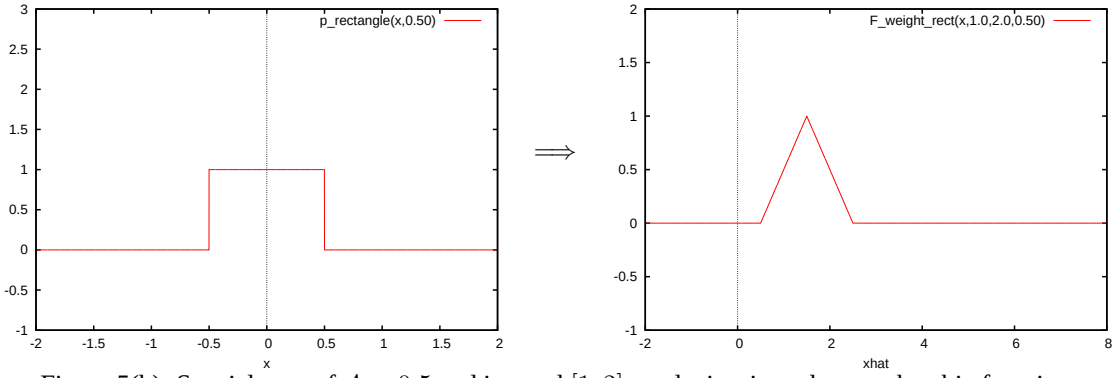

Figure 5(b): Special case of $A = 0.5$ and interval $[1, 2]$ results in triangular membership function

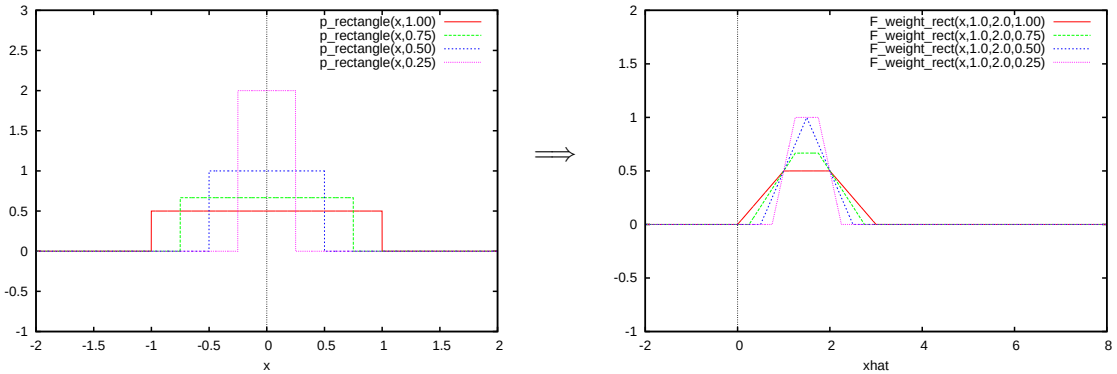

Figure 5(c): Rectangular PDF for $A = 1,\ 0.75,\ 0.5$ and 0.25, and corresponding membership functions

Figure 5: Rectangular probability density functions and corresponding membership functions for interval $[1, 2]$.

again to also approach $[a, b] = [1, 2]$. **The key fact to take away from Figure 5 is that the trapezoidal membership function - which is probably the most commonly used type of membership function in fuzzy set theory - corresponds to a box-shaped spread model that spreads the continuous value $\hat{x}$ with non-decreasing weight to the right and left of its value by up to $\pm A$. Thus $\hat{x}$ is simply enlarged to an interval $[\hat{x} - A, \hat{x} + A]$ with sharp cut-off on both sides.**

Figure 6 shows a triangular probability density function. Its corresponding member function looks a little like a Bell curve, but is really a concatenation of a flat portion and of four quadratic functions in $\hat{x}$ of the form $c_1 + c_2\hat{x} + c_3\hat{x}^2$.

Figure 7 finally shows the Gaussian probability density functions and their corresponding membership functions. Figure 7(a) shows the correspondence for $\sigma = 0.3$. Figure 7(b) uses a very small value, $\sigma = 0.01$,

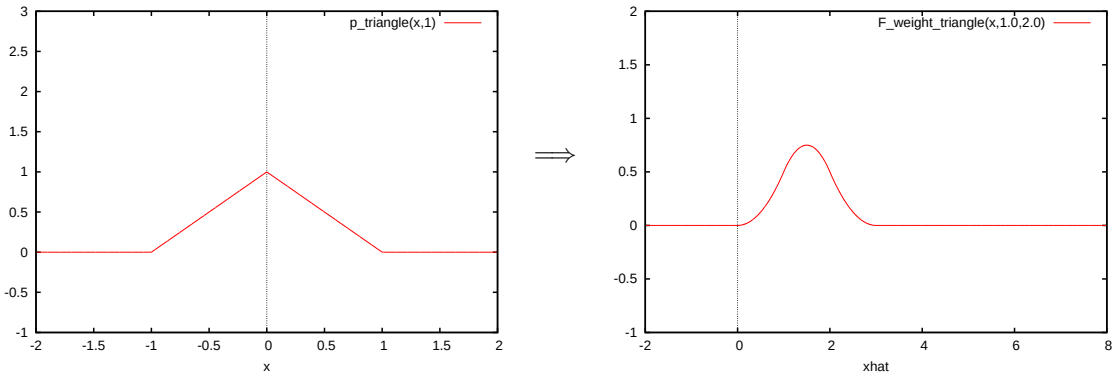

Figure 6: Triangular probability density function and corresponding membership function for interval $[1, 2]$.

to show that for $\sigma \to 0$ the PDF converges again toward the Dirac delta function and the membership function converges toward a rectangle with peak at the discretization interval, $[1, 2]$, which represents hard discretization. Figure 7(c) shows how the shapes of both functions change as σ decreases ($\sigma = 1.0, 0.5, 0.3, 0.1$).

5.2 Discussion

Studying these figures reinforces our view that it is much more intuitive to choose a spread model by choosing a probability density function, rather than defining an empirically chosen membership function as is usually done in fuzzy set theory. For example, to gain an intuitive feeling of how different a model is from hard discretization, we can simply look at the probability density function and see how much it differs from the Dirac delta function (shown approximately on the left of Figure 7(b)). For example, to analyze a triangular spread model one would look at the difference between triangular spread on the left side of Figure 6 and the narrow peak on the left side of Figure 7(b). An intuitive measure for that difference is the width of the peak of the probability density function, which also tells us exactly how far the continuous value is spread to the right and left. In contrast, looking at the membership function directly does *not* yield such easy answers. To answer the question of how different a model is from hard discretization we need to analyze the difference between its membership function and the rectangular function shown on the right side of Figure 7(b). For example, to analyze a triangular spread model one would look at the difference between the bumpy function on the right side of Figure 6 and the rectangular function on the right side of Figure 7(b). Thus it is much harder to estimate this difference by visual inspection of the membership function than it is using the probability density function. Thus for our type of application it seems more meaningful to define a probability density function, rather than starting with a "regular" membership function that is chosen regardless of probability theory.

When Zadeh founded fuzzy set theory in his seminal 1965 paper [18] he emphasized that membership functions are *not* to be designed using probability theory: *It should be noted that, although the membership function of a fuzzy set has some resemblance to a probability function when X is a countable set [...], there are essential differences between these concepts [...]. In fact, the notion of a fuzzy set is completely nonstatistical in nature.*

However, this view is slowly changing and the relationships between probability theory and fuzzy logic have been topic of much controversy recently, see Dubois and Prade [4]. The emerging view is that probability theory and fuzzy set theory are complementary, Zadeh [19], and that they can be useful in combination, Ross et al. [14]. We agree with this view and believe that our soft discretization approach is yet another example where the framework of fuzzy set theory and probability theory can be combined.

5.3 Summary

Because of the convolution connection between probability density functions and membership functions established in this section, it does not really matter in which framework soft discretization is implemented,

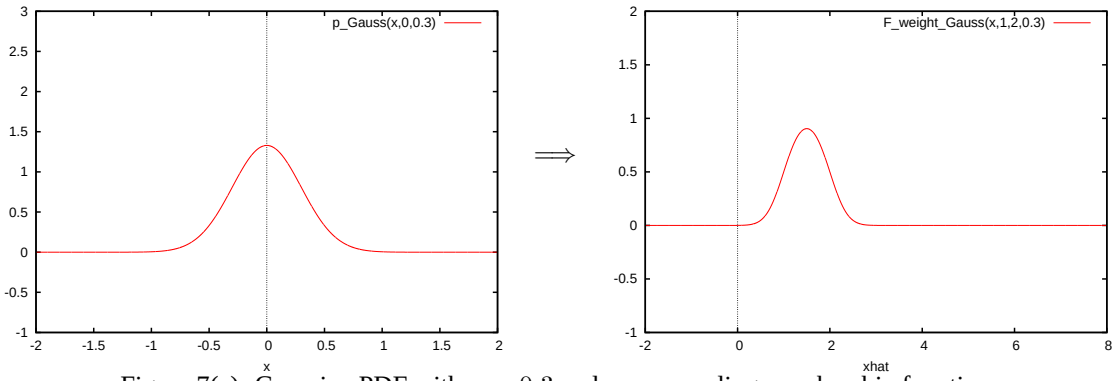

Figure 7(a): Gaussian PDF with $\sigma = 0.3$ and corresponding membership function

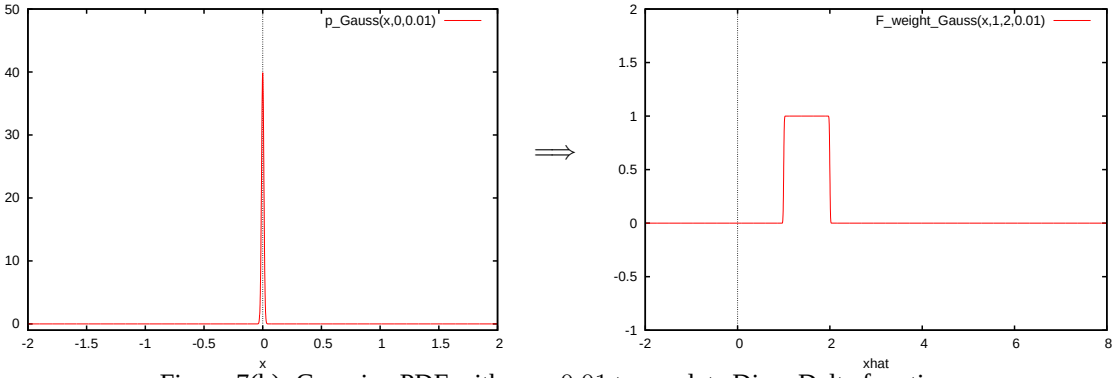

Figure 7(b): Gaussian PDF with $\sigma = 0.01$ to emulate Dirac Delta function

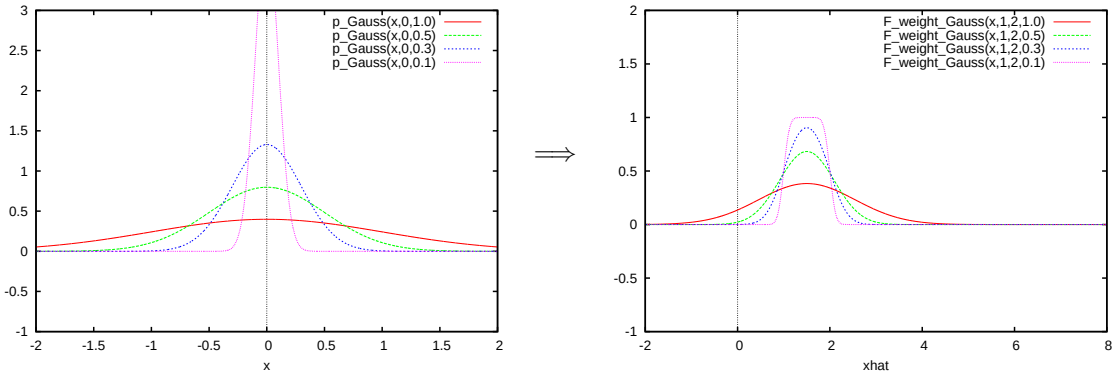

Figure 7(c): Gaussian PDF with $\sigma = 1.0,\ 0.5,\ 0.3$ and 0.1 and corresponding membership function

Figure 7: Gaussian probability density functions and corresponding membership functions for interval $[1, 2]$.

because both formulations can be made equivalent. However, it *does* matter how the basic function (Probability Density Function or Membership Function) is chosen. Since Bayesian Networks are fundamentally based on probability theory, it seems more appropriate to combine them with a Soft Discretization approach also based on probability theory. Analyzing the probability density function visually also provides more intuitive insight than the membership function. Thus if fuzzy set theory is to be used, then we believe the membership function should be derived from probability theory in the matter described above. It still remains, however, to show through experiments how much the choice of the basic functions actually affects the results in practice. No such experiments have yet been conducted.

6 Soft Discretization and Inference

So far we discussed how to *train* a discrete Bayesian Network from continuous data. Once the network is trained one usually wants to use the model for inference.

Subsection 6.1 discusses how to generate the right input for the inference algorithm. That is a very simple problem because we can just use the soft discretization method developed in Subsection 4.2 to convert continuous evidence into soft evidence. Soft evidence is accepted by most Bayesian Network toolboxes. Thus the soft evidence can be entered and then an inference algorithm applied.

Subsection 6.2 proposes a method for converting the *output* of the inference into continuous output values. This is the reverse problem of soft discretization, as we try to convert soft evidence of a variable into a meaningful continuous value.

6.1 Generating the Input for Inference Algorithm

Before performing inference we usually need to enter evidence into the discrete network and some or all evidence variables may have continuous values. For any variable X with continuous value $\hat{x}$ we simply apply Equation (8) to calculate the set of weights corresponding to its discrete states. For consistency, we use the same probability density functions for each variable as was used during training. The resulting weight for any discrete state can be interpreted as the probability with which X takes the corresponding state, $P(X = x_i) = w_i$. Thus the set of weights can be interpreted as soft evidence. Most Bayesian Network software packages allow the user to enter soft evidence, so we can simply enter the weights as soft evidence and then perform inference.

6.2 Generating a Continuous Output from the Discrete Model

Let us say we trained the discrete Bayesian Network from continuous data, converted evidence to soft evidence, entered the soft evidence in the network and applied the inference algorithm, we can now look at the *output of the inference algorithm*.

Let us consider the problem of inferring the states of a single variable, Z, that is continuous in nature, i.e. for which continuous values were given in the training data. The result of the inference is a set of probabilities for all the discrete states of the variable,

$$p_i = P(Z = z_i|e), \tag{17}$$

where e represents the evidence. In many cases it is best to leave the output in this form to provide the user with the maximum amount of information. For example, in a weather forecast it may be most meaningful for a user to know that there is a 45% chance of no rain, 50% chance of light rain and 5% chance of heavy rain.

However, in other cases it may be necessary to convert the inference output to a single value. This constitutes the inverse problem of the soft discretization method, which is also known as *defuzzification* in fuzzy set theory, does not have a simple solution. Wikipedia [16] lists twenty different approaches to solve the defuzzification problem and it is still topic of active research. The remainder of this section considers solving this problem not from a fuzzy set theory viewpoint, but from a Bayesian Network viewpoint.

The simplest - and most common - approach for interpreting the output of discrete Bayesian Networks is to use the *most likely state* of the inferred variable as output. However, there is a tremendous amount of information lost in this approach. For example, one would not be able to distinguish between the following two scenarios: (45% no rain, 50% light rain, 5% heavy rain) and (5% no rain, 50% light rain, 45% heavy rain). Both scenarios would simply say that "light rain" is the most likely outcome. Furthermore, this "most likely state" approach is very susceptible to how the boundaries between the states are chosen. For example, if we were to slightly increase the cut-off value between the "no rain" and "low rain" states, we could easily change the first scenario to (48% no rain, 47% light rain, 5% heavy rain), which would yield "no rain" as the most likely outcome.

It thus makes more sense to use some kind of weighted sum that takes the probabilities of the states into

account. The most natural way is to define an *expected value* for Z as follows:

$$E(Z) \;=\; \sum_{j=1}^{N^Z} P(Z = z_j|e) \;\; R(z_j) \quad = \sum_{j=1}^{N^Z} p_j \;\; R(z_j), \tag{18}$$

where p_j is the output of the inference and $R(z_j)$ denotes some representative value of the jth state of Z.

We first tried using the center of each interval as representative values, but that approach has at least two problems. First of all, the first and last interval include the values of $-\infty$ or ∞ as one of their boundaries, and it is thus unclear how to define their centers. Secondly, even for the other intervals the center quite often does not represent the interval very well. For example, the distribution of precipitation is exponential, thus values in each interval tend to cluster on the side closer to 0.

This illustrates the need to choose the representative value of each interval depending on the actual distribution of each considered variable. To solve this issue, and also to make the approach adjustable to a wide host of applications, we decided to use the training data to find good values for $R(z_j)$. Namely, *for each interval* we calculate its mean value based on the training data. In practice this means that for the jth interval of variable Z we first find all the data sets in the training data for which Z's value falls into the jth interval. Using *only those data sets* we calculate the mean of Z, and that mean is used as the value of $R(z_j)$. Using this method the set of values

$$\{R(z_j), j = 1, ..., \#\text{ states of } Z\}$$

can be calculated separately for each variable Z in the network using the training data. This preprocessing step does not require much time and only has to be done once. After this preprocessing step, $R(z_j)$, is readily available at the time of inference and $E(Z)$ can then be evaluated using Formula (18), yielding a continuous output.

This heuristic has provided decent results for our application in preliminary tests, but has not yet been tested for other applications. However, we believe its flexibility to adjust to other applications makes it a very promising candidate for further experimentation.

7 Implementation Details and First Results

This method was implemented using the Bayes Net Toolbox (BNT) for Matlab. We used the formulas above to calculate the probabilities, $P_{soft}(y_j|\mathbf{x}_i)$, for all nodes and assigned them to the CPT tables of the discrete BN, $P(y_j|\mathbf{x}_i)$. Inference was also implemented in the way described above.

7.1 Planned Release of Matlab Code

I am *planning* to eventually put the Matlab files on the web (at www.dataonstage.com) to make them available to anyone for further testing, but the code requires some clean-up and additional documentation before it can be released. Whether I go through with this additional work depends on the level of interest, so please contact me if you would like to use the code. You may also be able to get a copy of the files before the release that way. Please check the web page, www.DataOnStage.com, for updates or contact me directly (ebert@me.gatech.edu or ebert@stups.com).

7.2 Normal Density Function as Basic Building Block

We chose the probability density function of the normal distribution as the one-dimensional spread model, because it provides a smooth spreading of the continuous value and provides control over the amount of spreading through the intuitive parameter σ.

In our application all variables represent rainfall. Thus each variable only takes values between 0 and ∞ and measurement uncertainty of the amount of rainfall is expected to increase about proportionally with the amount of rainfall. For this type of variable a good model is to choose the deviation to be proportional to the value to be discretized. We can use $\sigma^x(\hat{x}) = \alpha \cdot \hat{x}$, where $\alpha \in (0, 1)$ is a constant parameter. Values of α larger than one are possible, but for our application choosing $\alpha = 0.3$ was more than sufficient.

Other choices for σ include an absolute deviation value, or deviation chosen proportional to the interval size that $\hat{x}$ corresponds to. Since the best way to choose the deviation is still topic of future research, in our implementation the user can choose for each variable to use either a fixed amount of deviation or a percentage of $\hat{x}$. Those expressions can also easily be replaced in the code by customized functions by the user.

Choosing values for σ is a trade-off. If σ is chosen very small, then there is little difference to hard discretization. If σ is chosen very large, then the soft discretization washes out more and more information in the training and test data. (For $\sigma \to \infty$, any continuous value is mapped *equally* to all discrete states.) As mentioned above, it remains to develop guidelines for how to choose the deviation function $\sigma^X(x)$ based on applications and to explore alternative spread functions.

7.3 Adjusting Interval Boundaries for Soft Discretization

As stated in Subsection 2.1 we require for each variable, X, to be discretized that the lower boundary of its left-most interval is $-\infty$ and the upper boundary of its right-most interval is ∞:

$$B^X_{1,low} = -\infty, \qquad B^X_{N^X,up} = \infty.$$

When using existing discretization intervals, for example boundaries defined previously for hard discretization, usually the intervals combined only span a smaller range, say $[A, B]$, rather than $[-\infty, \infty]$. A smaller range is always based on the implicit assumption that no values can ever fall outside the interval. For example the amount of rainfall can never be negative. However, when we *spread* values through soft discretization we may generate values outside of this range in the process. For example, the amount of rainfall can then become negative. To avoid having to test for these cases and to map for example all negative rainfall to zero rainfall, we simply override the outmost interval boundaries with the values $-\infty$ and ∞. This is permissible, because it does not change to which interval a continuous value is mapped, and automatically takes care of all the values falling outside of the normal range.

7.4 Check Sum Test

The first check for our method is to trace whether it is indeed possible to make all entries of the discrete conditional probability table well defined. To track that we defined the following check_sum function, that sums the CPT for each parent state combination, $\mathbf{x}_i$, over all child states,

$$\text{check\_sum}(\mathbf{x}_i) = \sum_j P(y_j|\mathbf{x}_i).$$

If all CPT entries, $P(y_j|\mathbf{x}_i)$, are well defined and perfectly accurate, we should get a value of 1 for the check sum:

$$\sum_j P(y_j|\mathbf{x}_i) = \sum_j \frac{P(y_j, \mathbf{x}_i)}{P(\mathbf{x}_i)} = \frac{1}{P(\mathbf{x}_i)} \sum_j P(y_j, \mathbf{x}_i) = \frac{P(\mathbf{x}_i)}{P(\mathbf{x}_i)} = 1.$$

If there is not enough data for training with the standard approach (hard discretization + MLE), then there are undefined CPT entries. For example, if one parent combination, say $\mathbf{x}_{i^*}$, never appears in the training data, the following CPT entries are all undefined:

$$P(y_j|\mathbf{x}_{i^*}), \text{for all } j.$$

Undefined CPT entries are set to the default value of zero in BNT. The check sum in this specific case should be zero for the i^* parent configuration, $P(\mathbf{x}_{i^*}) = 0$. Thus calculating the check sum for each parent configuration and checking how close it is to to the value one gives a good indication on how well defined the CPTs really are.

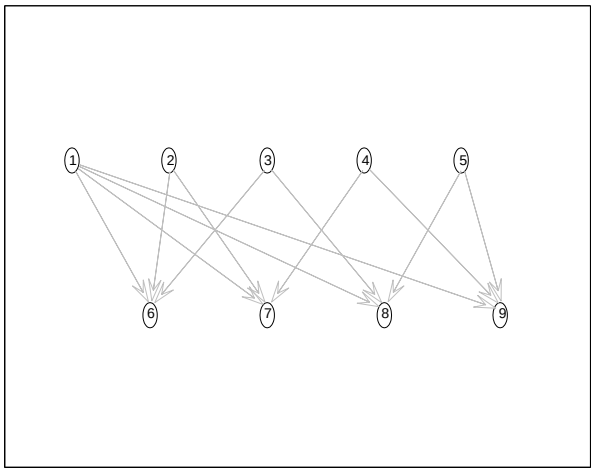

Figure 8: Graph of sample BN model for Precipitation Downscaling for a single Low Resolution Area

Discretization	# of check_sum entries = 1	(perc)	# of check_sum entries = 0	(perc)
Hard Discr. ($\sigma = 0$)	**192**	**(74%)**	**69**	**(26%)**
$\sigma = 5\%$	213	(82%)	48	(18%)
$\sigma = 10\%$	232	(89%)	29	(11%)
$\sigma = 15\%$	261	(100%)	0	(0%)
$\sigma = 30\%$	261	(100%)	0	(0%)

Table 1: Check Sum results for a sample model

7.5 Sample Model

We use the application of the downscaling problem for monsoon precipitation forecast as example. Figure 8 shows the graph for the sample Bayesian Network model used here, which represents a model centering around rainfall in a single region, which we call Region R. Node 1 represents the central low resolution node representing rainfall in Region R. Nodes 2 to 5 are also low resolution nodes representing the rainfall in areas to the north, west, east and south of Region R (Node 1). Region R is divided into four smaller regions and those are represented by Nodes 6-9 (high resolution nodes). The goal is to see how the rainfall in Region R and its neighbors (Nodes 1-5) maps to the rainfall in the high discretization areas of Region R (Nodes 6-9). In this model we adopt the view that the large weather pattern causes the small weather pattern, thus the low resolution nodes (1-5) serve as parents of the high resolution nodes in Figure 8. (Node 1 is parent of all high resolution nodes, while Nodes 2-5 are only parents of the high resolution nodes they are geographically closest to.) Each node has four states to represent the amount of rainfall. High resolution nodes use the intervals, ($(-\infty, 0.1)$, $[0.1, 5)$, $[5, 50)$, $[50, \infty)$, while low resolution nodes use $(-\infty, 0.1)$, $[0.1, 10)$, $[10, 100)$, $[100, \infty)$, all measured in mm.

7.6 Numerical Results

Throughout this section we use $\sigma = 0$ to indicate hard discretization. While hard discretization is *in theory* identical to soft discretization with $\sigma = 0$, in practice we always use regular hard discretization to calculate those values, since it is the easiest way.

For our application of precipitation downscaling we obtained the following results. Table 1 lists the number of check_sum entries that are one, and zero, respectively, as σ is increased from 0 to 30%. For hard

$P(N_1 = i)$	i=1	i=2	i=3	i=4
Hard Discr. ($\sigma = 0$)	**0.0862**	**0.2283**	**0.3539**	**0.3316**
$\sigma = 5\%$	0.0872	0.2278	0.3552	0.3298
$\sigma = 10\%$	0.0872	0.2280	0.3574	0.3273
$\sigma = 15\%$	0.0872	0.2286	0.3595	0.3247
$\sigma = 30\%$	0.0883	0.2334	0.3626	0.3157

Table 2: CPT entries for Node 1 for different values of σ.

$P(N_6 = 1 \mid N_1 = i, N_2 = j, N_3 = 1)$	j=1	j=2	j=3	j=4
i=1	1.0000	1.0000	1.0000	0
i=2	0.7403	0.6031	0.4274	0
i=3	0.6364	0.2830	0.1854	0.2414
i=4	0	0	0	0

Table 3: Subset of CPT table for Node 6 using hard discretization.

discretization ($\sigma = 0$) only 74% of the check_sum entries are one. This percentage increases with increasing σ. Already for $\sigma = 15\%$ all check_sum entries are one, so we can assume that all CPT entries are now well defined. Thus for this case the soft discretization is successful to fill undefined CPT entries.

While it is encouraging to see that many CPT entries went from undefined to well-defined, the process raises two questions:

1. What does the transition look like - from undefined to well defined?

2. How are the other CPT entries affected - the ones that were well defined from the beginning? How much do they change?

The remaining tables show the typical change of CPT entries for both of those classes, originally well defined and originally undefined CPT entries, as observed for our application. Table 2 shows the CPT entries for Node 1. As Node 1 is a root of the network it only has four CPT entries for its four different states. Clearly, these CPT entries are well populated, so as one would expect, the soft discretization (increasing σ) changes the values only very little.

Table 3 displays a subset for the CPT entries of Node 6 obtained using hard discretization. Node 6 has three parents, so its CPT is a four-dimensional array. Table 3 displays the CPT entries for State i of Node 1, State j of Node 2, State 1 of Node 3 and State 1 of Node 6, i.e. $P(N_6 = 1|N_1 = i, N_2 = j, N_3 = 1)$. (Recall that State 1 represents the dry state, while the other states indicate increasing amounts of rainfall.) We know that all entries that differ from 0 in Table 3 are well-defined. However, any entry that equals zero is a suspect for an undefined entry, since BNT sets undefined entries to default value 0. Table 4 shows how each entry of the array in Table 3 changes with increasing σ. Some observations from Table 4 are:

1. Entries that are well defined for $\sigma = 0$ (hard discretization) tend to change very little.
   Examples: (i, j) = (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3), (3,4), possibly (4,1).

2. Table 3 has six entries that are exactly zero. Looking at Table 4 shows that at least five of them are in undefined, i.e. they are zero only because the parent state combination did not occur in the (limited) training data.
   In contrast it seems that the entry for $(i, j) = (4, 1)$ may truly be zero for $\sigma = 0$, i.e. even for infinite amount of training data its value would be close to 0.

3. Some entries that are undefined for $\sigma = 0$ have slow transitions.
   Examples: (i,j)=(2,4),(4,2)

$P(N_6 = 1 \mid N_1 = i, N_2 = j, N_3 = 1)$	j=1	j=2	j=3	j=4
i=1				
$\sigma = 0\%$	**1.0000**	**1.0000**	**1.0000**	**0**
$\sigma = 5\%$	0.9982	1.0000	0.9926	0
$\sigma = 10\%$	0.9982	1.0000	0.9927	1.0000
$\sigma = 15\%$	0.9982	1.0000	0.9927	1.0000
$\sigma = 30\%$	0.9967	0.9956	0.9842	0.8601
i=2				
$\sigma = 0\%$	**0.7403**	**0.6031**	**0.4274**	**0**
$\sigma = 5\%$	0.7969	0.6148	0.4415	0.0226
$\sigma = 10\%$	0.7949	0.6153	0.4432	0.1953
$\sigma = 15\%$	0.7935	0.6155	0.4440	0.2759
$\sigma = 30\%$	0.7943	0.6160	0.4405	0.2974
i=3				
$\sigma = 0\%$	**0.6364**	**0.2830**	**0.1854**	**0.2414**
$\sigma = 5\%$	0.6676	0.2804	0.2131	0.2247
$\sigma = 10\%$	0.6872	0.2856	0.2072	0.2266
$\sigma = 15\%$	0.6977	0.2849	0.2042	0.2209
$\sigma = 30\%$	0.7104	0.2843	0.2037	0.2010
i=4				
$\sigma = 0\%$	**0**	**0**	**0**	**0**
$\sigma = 5\%$	0	0.1554	0.0519	0.0314
$\sigma = 10\%$	0	0.4412	0.0481	0.0347
$\sigma = 15\%$	0.0000	0.5369	0.0448	0.0383
$\sigma = 30\%$	0.0267	0.5630	0.0685	0.0508

Table 4: Subset of CPT table for Node 6 for different values of σ.

4. Other undefined entries have very sudden transitions, for example jumping from 0.0 to 1.0. This behavior was observed for several entries for other nodes as well.
   Example: (i,j)=(1,4).

5. Changes do not have to be monotonous.
   Example: (i,j)=(1,4)

8 Computational Complexity

Soft discretization is of course of much higher computational complexity than hard discretization. Let us consider a node Y with a set of parents, $\mathbf{X}$, and compare the complexity of calculating the whole CPT table with hard discretization and with soft discretization.

Let N denote the number of samples and M the number of parents. Let us assume for simplicity that each variable has the same number of discrete states, D. Then the number of CPT entries of node Y is

$$\#\text{ CPT entries of } Y = D^{M+1}.$$

We assume in the following that the number of samples is much larger than the number of CPT entries,

$$N \gg \#\text{ CPT entries of } Y,$$

which is a necessary condition to get a decent model.

Using hard discretization the calculation of the joint distribution $P_{MLE}(y_j, \mathbf{x}_i)$ in Equation (2) only requires a *single* run through the N data samples to generate *all* CPT entries. Thus the complexity to calculate all CPT entries of Y using hard discretization is $O(N + D^{M+1}) = O(N)$.

For soft discretization we first have to calculate all weights. Since this is done independently for each variable, we have D integration operations for each of the $(M + 1)$ variables. However, this step has to be performed for all samples, thus the overall complexity for this first step is $O(N \cdot (M + 1) \cdot D)$. The second contribution to computational cost of soft discretization is the calculation of the joint distribution $P_{soft}(y_j, \mathbf{x}_i)$ using Equation (16). Equation (16) requires N summations and $(N \cdot M)$ multiplications for *each* CPT entry, thus this step is of complexity $O(N \cdot M \cdot D^{M+1})$. Since the second step is of higher complexity than the first, the overall complexity to calculate all CPT entries using soft discretization is $O(N \cdot M \cdot D^{M+1})$.

These results match the observations from our experiments, where the second step of soft discretization took longer than the first step, even for only $M = 3$ parents with $D = 4$ states each, when using a Gaussian distribution as basic building block. Thus the choice of the specific spread model does not seem to greatly impact the computational cost. In contrast, the number of CPT entries (i.e. the number of parents and number of states per variable) and the number of samples are the dominating factor for computational cost. As number of parents and number of samples are usually given by the application, the number of states per variable is the only free parameter that can be adjusted if computational cost becomes an issue.

9 Conclusions and Future Work

For our particular application the soft discretization proved to be successful in filling all the undefined CPT entries. Furthermore, manual inspection of all CPT entries of all nodes of our model showed that the CPT entries already well defined for $\sigma = 0$ only changed very little with increasing σ. Thus for this example we can achieve well defined CPTs without washing out the other CPTs too much. More rigorous testing, especially with many other models, is yet to be conducted.

We are still at the beginning of research on probability-based - and other - soft discretization approaches for Bayesian Networks. It remains to study the effect of the shape of probability density functions on the CPT values, as well as developing guidelines on how to choose σ if a Gaussian distribution is used. Furthermore, not only the CPT values, but the quality of inference results needs to be studied and tested for different models. The method for the inverse problem in Section 6.2 needs to be rigorously tested and alternatives should be developed and compared. In short, we hope that this report will stimulate much more research on this topic that will lead to a rigorous framework for the use of soft discretization in Bayesian Networks.

10 Acknowledgments

My thanks go to Annalisa Bracco at Georgia Tech for being a terrific collaborator and for introducing me to the application of monsoon precipitation modeling.

References

[1] Jim F. Baldwin and E. Di Tomaso. Inference and learning in fuzzy bayesian networks. In *IEEE International Conference on Fuzzy Systems*, volume 1, pages 630–635, May 2003.

[2] J.R. Boston. Effects of the shape of fuzzy membership functions on fuzzy inference. In *Proceedings of the 3rd International Symposium on Uncertainty Modelling and Analysis*, volume 1, page 32, 1995.

[3] Annalisa Bracco, Fred Kucharski, Franco Molteni, Wilco Hazeleger, and Camiel Severijns. A recipe for simulating the interannual variability of the Asian summer monsoon and its relation with ENSO. *Clim Dyn*, 28(5):441–460, September 2006.

[4] Didier Dubois and Henri Prade. Fuzzy sets and probability : Misunderstandings, bridges and gaps. In *In Proceedings of the Second IEEE Conference on Fuzzy Systems*, pages 1059–1068. IEEE, 1993.

[5] Christopher Fogelberg, Vasile Palade, and Phil Assheton. Belief propagation in fuzzy bayesian networks. In *1st International Workshop on Combinations of Intelligent Methods and Applications(CIMA) at ECAI'08*, pages 19–24, University of Patras, Greece, 21–22 July 2008.

[6] Lawrence D. Fu. A comparison of state-of-the-art algorithms for learning bayesian network structure from continuous data. Master's thesis, Biomedical Informatics, Vanderbilt University, December 2005.

[7] Lawrence D. Fu and Ioannis Tsamardinos. A comparison of bayesian network learning algorithms from continuous data. In *AMIA Annual Symposium Proceedings*, page 960, 2005.

[8] Finn V. Jensen and Thomas D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, 2nd edition, 2007.

[9] Kittipat Kampa. Personal communication, 2009. Ph.D. Student, Deparment of Electrical and Computer Engineering, University of Florida.

[10] Ludovic Koehl and Xianyi Zeng. A method for optimizing fuzzy membership functions. *Studies in Information and Control*, 7(2), 1998.

[11] F. Kucharski, A. Bracco, J.H. Yoo, and F. Molteni. Low-frequency variability of the Indian monsoon - ENSO relation and the Tropical Atlantic: The 'weakening' of the '80s and '90s. *J. Climate*, 20(16):4255–4266, 2007.

[12] Chun-Yi Lin, Jun-Xun Yin, Li-Hong Ma, and Jian-Yu Chen. An intelligent model based on fuzzy bayesian networks to predict astrocytoma malignant degree. In *IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–5, June 2006.

[13] Heping Pan and Lin Liu. Fuzzy bayesian networks - a general formalism for representation, inference and learning with hybrid bayesian networks. In *6th International Conference on Neural Information Processing (ICONIP'99)*, Perth, WA, Australia, November 1999.

[14] Timothy J. Ross, Jane M. Booker, and W. Jerry Parkinson. *Fuzzy Logic and Probability Applications: Bridging the Gap*. ASA-SIAM Series on Statistics and Applied Probability. Society for Industrial Mathematics, 2002.

[15] Hao Tang and Shi Liu. Basic theory of fuzzy bayesian networks and its applications in machinery fault diagnosis. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, 2007.

[16] Wikipedia. Topic: Defuzzification, September 2009. http://en.wikipedia.org/wiki/Defuzzification.

[17] Jing Yu, V. Anne Smith, Paul P. Wang, Alexander J. Hartemink, and Erich D. Jarvis. Using bayesian network inference algorithms to recover molecular genetic regulatory networks. In *International Conference on Systems Biology 2002 (ICSB02)*, December 2002.

[18] L.A. Zadeh. Fuzzy sets. *Information And Control*, 8:338 – 353, 1965.

[19] Lofti A. Zadeh. Discussion: Probability theory and fuzzy logic are complementary rather than competitive. *Technometrics*, 37(3):271–276, 1995.