Threshold Definition of Early Warning Systems to Natural Hazards

by

Jonathan Feinberg

Thesis
for the Degree of
Master of Science

(Master i Modellering og dataanalyse)

Faculty of Mathematics and Natural Sciences
University of Oslo

January 2009

Det matematisk- naturvitenskapelige fakultet
Universitet i Oslo
# Contents

1 Introduction 3  
1.1 A Simplified Tsunamigenic Rockslide Model 4  
1.2 Objective 5  
1.3 Chapters 5  
1.4 Resources Used 5  

2 Theoretical Basis 7  
2.1 Influence Diagram 7  
2.1.1 Directed Acyclic Graph 7  
2.1.2 Influence Diagram 9  
2.2 Bayesian Network 10  
2.2.1 Bayesian Paradigm 10  
2.2.2 Bayesian Updating 11  
2.2.3 D-separation 13  

3 Interpretation of an Expert Opinion 18  
3.1 Simplified Information 18  
3.2 Converting to Continuous 19  
3.3 Fit of Dependent Variables 23  

4 Model Construction 27  
4.1 Base Model 27  
4.1.1 Rockslide 29  
4.1.2 Tsunami 30  
4.1.3 Season 31  
4.1.4 Consequence 31  
4.2 Early Warning System 34  
4.2.1 Sensor 34  
4.2.2 Early Warning System 35  
4.2.3 Threshold 35  
4.2.4 Loss 38
## 5 Monte Carlo Simulation

5.1 Introduction to sampling .................................. 40
   5.1.1 Base Model ............................................ 43
   5.1.2 Logic Sampling ........................................ 43
   5.1.3 Sensor Information ................................... 47

5.2 Setting the Loss and Threshold Functions ................. 49
   5.2.1 Searching for Optimum ............................... 51
   5.2.2 Inference on Thresholds .............................. 53

## 6 Conclusion ............................................. 57

## A Python Scripts .......................................... 59
   A.1 Particle Swarm Optimizer ............................... 59
   A.2 Distribution Fitting .................................... 61
   A.3 Parameter Retrieval .................................... 62
   A.4 Node Sampler .......................................... 64
   A.5 Threshold Tester ....................................... 66

## B Acknowledgement ......................................... 69
   References ................................................ 69
Abstract

Information collected from warning systems monitoring natural threats can be synthesized into a risk measure to determine the state of nature, and for defining the threshold for issuing (or not) of an early warning. This means that a change in the risk measure can trigger the implementation of countermeasures for reducing the hazard, or for reducing the vulnerability and the consequences. Associated costs in the implementation of either should be added to the risk measure, so that the updated risk can be used as the index defining the corresponding warning level.

This work introduces the case of a pre-reliability analysis of the tsunami-genic rockslide at Åknes, Norway. For this purpose, information gathered from engineers, geologists and other stakeholders are incorporated into a probability template based on inference diagrams, which allow for representing causal dependence between events (represented by nodes), taking place from the threat triggering factors up to the definition of the risk measure. Some of these include key events such as the threats triggering factors, the threats themselves (rockslide and tsunami) and the effect of the Early Warning System. In addition, it is necessary to define how these events interact, which should be stated through the definition of probability distributions.

Once the information from the key events is gathered, it can be resolved using a directed acyclic graph or network, making a model which is graphically more intuitive to understand. In this way, it is possible to trace the passing of information through the network by making use of the dependencies defined for each probability distribution. This is possible either through probability transformation or by letting a set of random variables be a part of a second level of parameters (hyper-parameters). This means that it is possible to create complex information structures using simple causal representations, which help to enhance inferences about the risk measures based on associations.

After constructing the network, the major challenge is to find optimal thresholds for assigning the warning levels. This thesis introduces Monte Carlo simulation techniques to propagate probability states, through variation of threshold levels, so that optimal risk measures with the lowest expected consequence can be found.

Information from the Åknes site is collected and processed in real time. Bayesian principles will be introduced creating a Bayesian Network which will allow for updating information in any node in the network, and propagate it back and forth on different directions according to the presence of new evidence at any time. This exercise is a pre-reliability
analysis, which can help to build the decision-making process associated to the implementation of an Early Warning System.
Chapter 1

Introduction

The Åknes project aims at implementing a monitoring and warning system on a large unstable rockslope in Storfjord in western Norway. The goal is to minimize the risk associated to a potential rockslope failing and plunging into the fjord, creating a large tsunami that would affect the surrounding communities. One of the most effective counter measures for reducing the risk is to develop a well-defined emergency plan including evacuation, road closings and other active measures, plus the implementation of an Early Warning System that determines the level of emergency. Currently Early Warning System is based on the interpretations of expert opinion into an active monitoring system. The system monitors physical evidence from the rockslide including crack extensions and local displacements through reflectors, extensometers and crackometers, and pore water pressure and water content through DMS and climate changes as temperature, rain precipitation and snow precipitation. The experts also take into account information gathered in a regional hazard analysis[15].

Experts from multiple disciplines need to combine their expertise given the data from the monitoring system to determine the true state of nature in an adequate way. Since the classification is time sensitive, gathering the experts opinion can be costly at best and impossible at worst. Given the probable state of nature generated, threshold levels has to be chosen to determine the level of emergency.

This thesis addresses this problem by introducing a tool to estimate the state of nature and to make a framework for the corresponding decision making process. This work presents a probabilistic tool as applied to a real case study based on information gathered from experts involved in the Åknes project.
1.1 A Simplified Tsunamigenic Rockslide Model

Figure 1.1 illustrates the sequence of events associated to a tsunamigenic rockslide at the Åknes site. A series of potential triggering factors, such as seasonal weather conditions, rain precipitation, earthquake, snow melt, geometric and kinematic constraints and state of material composition may initiate a rockslide event. If this happens with high enough velocity and volume into the fjord, it would cause a tsunami that could potentially bring disastrous consequences to its surrounding. The Early Warning System, or EWS for abbreviation, is implemented as a mitigating measure. It’s function is to minimize the potential impact of the tsunami by issuing a warning before the tsunami develops. Looking at both the consequences and the effect of the evacuation we can associate a risk measure using economical, casualties and/or social impact as scale.

The states in each node can be associated with a probability, converting the flowchart into an influence diagram. An influence diagram is a graphical tool that defines the structure of a flowchart as a probability distribution. Each node in the network will assume a probability distribution, which immediately will affect other nodes. Information inserted into the diagram
can propagate making the nodes dependent. How the information propagates and what properties the diagram has, depends on which underlying techniques for information propagation are implemented. This work will discuss two of these: Forward propagation and Bayesian Network.

1.2 Objective

The aim of the thesis is to demonstrate the advantages and disadvantages of an Influence Diagram and a Bayesian Network when modeling a hazard's problem. The answers given in the thesis are dependent on assumptions, so the estimates presented are not intended for policy making. The intention herein is to give an introduction to a framework that can be useful for future applications and research.

It is relevant to mention that in the case of the Åknes project, there is not enough data available to use statistical analysis alone, and some expert assumptions were considered.

1.3 Chapters

Chapter 2 will introduce theoretical tools needed for the thesis. These are mainly tools taken from probability theory, statistics and computer science. It will in detail describe how a Bayesian Network works as a tool. This is not limited to how evidence can be inserted into the network.

Chapter 3 will discuss how information can be collected from experts, and how it is then fitted into distributions. A case study will be introduced as a way to illustrate the probabilistic methods applicability.

In chapter 4, the implementation of the tools defined in chapter 2 will be used on the model defined in chapter 3. The model will be extended to include an Early Warning System. Some inferences will be used to describe the contents of the model. The chapter will not introduce solutions to the problems in the thesis, but will give an overview of the parameters that need to be calibrated or chosen.

Chapter 5 will use simulation techniques to calibrate the Early Warning System. It will demonstrate how optimal thresholds in the case study can be found for the model. It will also demonstrate how evidence or assumptions can be inserted to do inference.

1.4 Resources Used

Many types of software are available to implement an Influence Diagram and/or Bayesian Network to the expert data. Two software tools
encountered during the work of the thesis that are worth mentioning are Riscue[12] and GeNlе[2]. Risque is a good tool for simulating under an Influence Diagram. It is easy to use and is excellent for doing Monte Carlo simulations. GeNlе is an implementation-tool for Bayesian Network. It has a graphical interface and API that makes it easy to work with.

These two tools are excellent in their respective areas. This thesis needs the functionality of both softwares and a little extra that falls between them. Therefore an new tool had to be developed.

All software used in this thesis have been created in Python. The package scipy[5] has been used for numerical calculations including statistical functions. For plotting figures matplotlib[4] is used. The rest is written as part of the thesis. The code used for doing the simulation can be found in the appendix.
Chapter 2

Theoretical Basis

2.1 Influence Diagram

2.1.1 Directed Acyclic Graph

A graph is a way of structuring information[18]. This can be used when information is categorized into distinct elements of contents, and when these relate to each other exclusively in a pairwise manner. A graph consists of two components: nodes and edges. A node is a container where information is stored, an edge is a relation between a pair of nodes. Two nodes can at most have one edge between them. Because of this, graphs have a visual representation which is easy to interpret. As in figure 2.1, nodes are represented by ellipses with either its name or a property symbolized inside the figures. Edges are represented as lines between the nodes which they are connecting.

In general the relationship represented by an edge is symmetrical. If node $A$ is connected to node $B$, then node $B$ is connected to node $A$. But the relationship does not need to be symmetrical. An unsymmetrical relationship can be represented by letting edges be visually represented by arrows instead of lines. If there is an arrow starting in $A$ ending in $B$, this means that $A$ is connected to $B$, but $B$ is not connected to $A$. This can also be written as $A \to B$. An edge that represents an unsymmetrical relationship is also called an arc. An example of an undirected and a directed graph is represented in figure 2.2.

A graph that only consists of unsymmetrical relationships, is called a directed graph. Equivalently, a graph that is only consisting of symmetrical relationships, is called an undirected graph. Let $A$ and $B$ be two nodes in the same directed graph. If by following the direction of the arcs it is possible to travel from $A$ to $B$, it is said that there is a directed path from $A$ to $B$. This can also be written as $A \rightsquigarrow B[6]$. If $A \rightsquigarrow B$ and $B \rightsquigarrow A$, it is said that the graph contains a cycle. A graph that contains a cycle is called a cyclic graph. If it does not contain a cycle it is called an acyclic graph. Graphs
that will be discussed in this paper are all directed and acyclic. A directed unyclic graph is called a DAG for abbreviation.

There are several ways to semantically refer to a DAG. The relationship $A \rightarrow B$ can be referred to as $A$ being $B$’s parent and $B$ is $A$’s child[18]. Following this metaphor a DAG can be referred to as a family. For example, $A$ and $B$ are siblings if they share at least one parent. All parents, all grandparents, all grandparents parents and so on of $A$ are referred to as $A$’s ancestors. Equivalently $A$’s children, children’s children and so on are referred to as $A$’s descendants. Let $\text{chi}(A)$, $\text{par}(A)$, $\text{dec}(A)$ and $\text{anc}(A)$ respectively be the sets of children, parents, descendants and ancestors of $A$. Another often used metaphor is the tree metaphor. A node without parents is referred to as a root node. A node without children is referred to as a leaf node. A branch is defined as a node and all of its descendants. The tree metaphor is usually used when there is at most one path between each pair of nodes. For general graphs, root nodes and leaf nodes can be used, but branches are usually substituted with descendants. The path metaphor
is already referred to indirectly. Formalizing it, there is a directed path from \( A \) to \( B \) if \( B \) is a descendant of \( A \). An undirected path is a path without cycles that ignores the direction of the arcs.

### 2.1.2 Influence Diagram

An influence diagram is the representation of a probability model in a directed acyclic graph. It consist mainly of nodes with random properties. These are referred to as chance nodes or uncertainty nodes.

**Definition 1** (Chance Node). Let \( A \) be a node in the graph \( G \). Let all nodes in \( G \) have a known or unknown value. A is a chance node if it can be identified as a random variable with probability distribution \( P(A|\text{par}(A)) \).

Let the nodes \( X_1, \ldots, X_n \) be elements defining the influence diagram \( D \). If \( X_i \) is in \( \text{anc}(X_j) \), but not in \( \text{par}(X_i) \) and \( i \neq j \), \( X_i \) and \( X_j \) are indirectly dependent. Since \( X_j \sim X_i, X_j \)'s sample value can in turn affect \( X_i \)'s value through the links of dependencies. But if for each path from \( X_j \) to \( X_i \) exists a node where the sample value is known, i.e. a random variable \( X_l = x_l \), the dependency link is broken. By knowing the sample value of a random variable, it is said to block the path from its parents to itself. If all possible paths \( X_j \sim X_i \) contain a blocking node, \( X_i \) is independent of \( X_j \). The relationship \( X_j \) independent of \( X_i \) given a set \( S \subseteq \{X_1, \ldots, X_n\} \setminus X_i, X_j \) can be written as \( X_i \perp X_j|S \).

Since the dependencies only travel in the direction of the arcs, all root nodes are independent. This implies that the joint probability over the rootnodes can be formalized as the product over the marginals. Since the root nodes do not have any parents, the associated random variable is a marginal. In general the probability calculus gives us that \( P(A, B) = P(A)P(B|A) \). This rule for dependent structures can be used to create a joint probability between a rootnode \( A \) and \( \text{child}(A) = B \). Using this merging of nodes as a repeating principal, many nodes can be collected into a single joint distribution. Since the influence diagram has no cycles, the joint probability over all random variables in a network can be retrieved\[13\].

\[
P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{par}(X_i))
\]  

(2.1)

In addition to the chance node, there are two other node types that can be used in an influence diagram: decision nodes and utility nodes. These are tools for helping with decision making and are defined as follows:

**Definition 2** (Decision Node). Let the node \( D \) be a root node in the influence diagram \( G \) with the states \( \textbf{d} = \{d_1, d_2, \ldots, d_n\} \).
2.2 Bayesian Network

2.2.1 Bayesian Paradigm

The fundamental principle of the Bayesian paradigm is based on Bayes’ theorem.[17]

**Theorem 1** (Bayes’ Theorem). Let the vector of parameters $\theta$ be an element in the parameter space $\Theta$ with $m$ elements. The goal is to estimate $\theta$ based on available evidence.

Let $x$ be a vector of data or observations that is used to estimate $\theta$’s value.

$$
P(\theta|x) = \frac{P(\theta)P(x|\theta)}{P(x)} = \frac{P(\theta)P(x|\theta)}{\int_\Theta P(\theta)P(x|\theta) \, d\theta} = \alpha \times P(\theta) \times L(\theta)
$$
The marginal distribution $P(\theta)$ and the conditional distribution $P(\theta|x)$ are respectively referred to as the prior and the posteriori distribution. Since $x$ is defined as data, $P(x|\theta)$ is the likelihood function $L(\theta)$. The denominator is independent of $\theta$ and can be looked upon as a normalizing constant $a$.

When using regular probability theory in an influence diagram, information travels in one direction. It starts at the root nodes and ends at the leaf nodes, following the direction of the arcs. This is called forward propagation. But by using the Bayesian approach, information can do what is called backward propagation and travel both in and against the direction of the arcs.

### 2.2.2 Bayesian Updating

The definition of a Bayesian Network is an influence diagram where information is allowed to propagate both forward and backward with the help of Bayes’ theorem. For instance in figure 2.4 the three chance nodes $A$, $B$ and $C$ define the influence diagram $\mathcal{D}$ with the structure $A \rightarrow B \rightarrow C$ and the distributions $P(A)$, $P(B|A)$ and $P(C|B)$ respectively. Because of the structure, observing $B = b$ will affect $C$ directly. Its distribution becomes $P(C|B = b)$. Because of backward propagation $A$ is also affected. It has the following new distribution.

$$P(A|B) = \alpha \times P(A) \times L(A)$$  \hspace{1cm} (2.2)

**Example 1** (Diagnosing Phil). Let the variables in figure 2.4 represent a diagnostic problem.[13] The patient, Phil is coughing a lot and the doctor is suspecting cancer. Let $A \in \{\text{yes}, \text{no}\}$ represent the possibility that Phil smokes cigarettes, let $B \in \{\text{yes}, \text{no}\}$ represent the possibility that Phil has cancer and let $C \in \{\text{yes}, \text{no}\}$ represent the possibility of cancer being confirmed by an x-ray machine. The model assumes that smoking can cause cancer and in turn cancer can be confirmed by an x-ray. The implications are not deterministic, each state occurs with a given probability distribution. Studies show that in the neighborhood that Phil lives in, 30% of the people of his gender and age smokes. Without asking Phil if he smokes, the model can have the prior probabilities 0.3 and 0.7 for Phil smoking cigarettes.
Table 2.1: The conditional probability table for smoking, cancer and x-ray.

and not smoking respectively. Through experience the doctor expects to find cancer in 4 in every 100 smoking patients he suspects has cancer. For non-smokers the numbers are 1 in every 100. At the same time the accuracy of the X-ray test is not perfect. 10% of positive resulted tests turns out to be wrong. On the other hand, 5% of negative resulted test also turns out to be wrong. The conditional probabilities that define the network can be formalized in a table. The conditional probability table, abbreviated CPT, can for this example be seen in table 2.1.

To update information from one node to another with Bayes' theorem both a prior and a likelihood is needed. The likelihood is given in the conditional probability table. The priors can be calculated from the same table. The prior for having cancer is as follows.

\[
P(B = \text{yes}) = \sum_{a \in \{\text{yes}, \text{no}\}} P(B = \text{yes} | A = a) P(A = a) = 0.019
\]

Looking at this result as a weighted mean between having cancer while respectively being a smoker and not being a smoker, the probability falls intuitively between the two conditional probabilities, 0.01 and 0.04. Continuing calculating the marginals for all variables and all states results in the following. For simplifications \( P(A = k) \) is denoted \( P(A_k) \).

\[
\begin{align*}
P(A_{\text{yes}}) &= 0.3 & P(A_{\text{no}}) &= 0.7 \\
P(B_{\text{yes}}) &= 0.019 & P(B_{\text{no}}) &= 0.981 \\
P(C_{\text{yes}}) &= 0.06615 & P(C_{\text{no}}) &= 0.93385
\end{align*}
\]

The probability of getting a positive result on an x-ray machine is low. Since the probability for having cancer is low, the relative frequency of positive results should be low as well.

Wondering if Phil has cancer, the doctor takes an x-ray of Phil’s chest. The result is cancer. A common error amongst doctors is to assume that since both the conditional and the marginal probability of the x-ray machine diagnosing correctly
is high, the probability of having cancer given positive results must be equally high. Using equation (2.2) to calculate the latter probability shows that this is not always the case.

\[
P(B_{yes}|C_{yes}) = \alpha \ P(B_{yes}) P(C_{yes}|B_{yes}) = \alpha \ 0.0171
\]

\[
P(B_{no}|C_{yes}) = \alpha \ P(B_{no}) P(C = yes|B_{no}) = \alpha \ 0.04905
\]

\[
\alpha = \frac{1}{0.0171-0.01495} = 15.11715
\]

\[
P(B_{yes}|C_{yes}) = 0.25850 \quad P(B_{no}|C_{yes}) = 0.74149
\]

The probability for having cancer given a positive x-ray is low. The reason is that the cancer is uncommon. A positive result is more likely to be false positive than a true positive. The doctor should therefore continue the diagnosing. The doctor asks Phil if he smokes and he gets a confirmation in return.

\[
P(B_{yes}|A_{yes}, C_{yes}) = \alpha \ P(B_{yes}|A_{yes}) P(C_{yes}|A_{yes}, B_{yes}) =
\]

\[
\alpha \ P(B_{yes}|A_{yes}) P(C_{yes}|B_{yes}) = \alpha \ 0.036
\]

\[
P(B_{no}|A_{yes}, C_{yes}) = \alpha \ P(B_{no}|A = yes) P(C_{yes}|A_{yes}, B_{no}) =
\]

\[
\alpha \ P(B_{no}|A_{yes}) P(C_{yes}|B_{no}) = \alpha \ 0.048
\]

\[
\Rightarrow \alpha = \frac{1}{0.036+0.048} = 11.90476
\]

\[
\Rightarrow P(B_{yes}|C_{yes}) = 0.42857
\]

\[
P(B_{no}|C_{yes}) = 0.57142
\]

The fact that Phil smokes helps the doctor diagnose Phil. When making models it is important to account for as many variables as possible. This example demonstrates that unaccounted variables that seemingly have little effect on the model, can have a great impact under some circumstances.

In the calculations the probability \( P(C_{yes}|A_{yes}, B_{yes}) \) was reduced to \( P(C_{yes}|B_{yes}) \). This is a consequence of the conditional independence \( A \perp \!\!\!\perp C|B \). The model is constructed such that there are no direct dependencies between smoking and x-ray, i.e. if the doctor knows that Phil has cancer, smoking will not affect the x-ray test. This might not be the case in reality. If there is a direct dependence between smoking and x-ray it implies that there should be an arc between the variables. This possible discrepancy needs to be addressed when constructing diagrams.

### 2.2.3 D-separation

In figure 2.4 it was assumed that \( P(C_{yes}|A_{yes}, B_{yes}) = P(C_{yes}|B_{yes}) \). The following calculations, using equation (2.1), confirm this assumption:

\[
P(C|A \cap B) = \frac{P(A \cap B \cap C)}{P(A \cap B)} = \frac{P(A)P(B|A)P(C|B)}{\sum_{C} P(A)P(B|A)P(C|B)} = \frac{P(C|B)}{\sum_{C} P(C|B)} = P(C|B)
\]

(2.3)
This gives a rule for dependency between ancestors and descendants. This section will investigate the rules that make different variables dependent or independent. A model illustrating the sibling relationship is illustrated in figure 2.5. D is parent to E and F with no direct dependency between E and F. A simple example can demonstrate that E and F are prior dependent (see example 2). But given the observation of D, the variables become independent. This is shown in the following calculation:

\[
P(E|D \cap F) = \frac{P(D \cap E \cap F)}{P(D \cap F)} = \frac{P(D)P(E|D)P(F|D)}{\sum_E P(E)P(E|D)P(F|D)} = \frac{P(E|D)}{\sum_E P(E|D)} = P(E|D)
\]

(2.4)

Figure 2.6 illustrates the opposite structure with two nodes G and H with a common child I. The following calculation confirms that there is a prior independence between G and H. It is based on equation (2.1):

\[
P(G \cap H \cap I) = P(G)P(H)P(I|G \cap H)
\]

\[
P(G) = \frac{P(G \cap H \cap I)}{P(H)P(I|G \cap H)} = \frac{P(G \cap H \cap I)}{P(G \cap H \cap I)} = P(G|H)
\]

(2.5)

(2.6)

Example 2 (Diagnosing Phil part II). Continuing example 1, the doctor takes into account two extra variables to determine if Phil has cancer. If the patient
Figure 2.7: A model with five variables: smoker (S), pollution (P), cancer (C), dyspnea (D) and x-ray (X). The structure is $S \rightarrow C \leftarrow P$ and $D \leftarrow C \rightarrow X$.

<table>
<thead>
<tr>
<th>P: Pollution</th>
<th>$P = high$</th>
<th>$P = low$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S: Smoker</th>
<th>$S = yes$</th>
<th>$S = no$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C: Cancer</th>
<th>$C = yes$</th>
<th>$C = no$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P = high, S = yes$</td>
<td>0.05</td>
<td>0.95</td>
</tr>
<tr>
<td>$P = high, S = no$</td>
<td>0.02</td>
<td>0.98</td>
</tr>
<tr>
<td>$P = low, S = yes$</td>
<td>0.03</td>
<td>0.97</td>
</tr>
<tr>
<td>$P = low, S = no$</td>
<td>0.001</td>
<td>0.999</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X: X-ray</th>
<th>$X = yes$</th>
<th>$X = no$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C = yes$</td>
<td>0.9</td>
<td>0.1</td>
</tr>
<tr>
<td>$C = no$</td>
<td>0.05</td>
<td>0.95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D: Dyspnea</th>
<th>$D = yes$</th>
<th>$D = no$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C = yes$</td>
<td>0.65</td>
<td>0.35</td>
</tr>
<tr>
<td>$C = no$</td>
<td>0.3</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 2.2: The conditional probability table for pollution, smoker, cancer, x-ray and dyspnea.

...
CPT is given in table 2.2. This gives the following prior probabilities:

\[
P(P_{\text{high}}) = 0.1 \quad P(P_{\text{low}}) = 0.9 \\
P(S_{\text{yes}}) = 0.3 \quad P(S_{\text{no}}) = 0.7 \\
P(C_{\text{yes}}) = 0.01163 \quad P(C_{\text{no}}) = 0.98837 \\
P(X_{\text{yes}}) = 0.20814 \quad P(X_{\text{no}}) = 0.79189 \\
P(D_{\text{yes}}) = 0.30407 \quad P(D_{\text{no}}) = 0.69592
\]

The doctor observes that Phil has dyspnea. This affects the network in the following way.

\[
P(C_{\text{yes}} | D_{\text{yes}}) = \frac{P(C_{\text{yes}})P(D_{\text{yes}} | C_{\text{yes}})}{P(D_{\text{yes}})} = 0.02486
\]
\[
P(S_{\text{yes}} | D_{\text{yes}}) = \sum_{C} P(S_{\text{yes}} | C)P(C | D_{\text{yes}}) = \sum_{C} \frac{P(S_{\text{yes}})P(C | S_{\text{yes}})}{P(C)}P(C | D_{\text{yes}})
\]
\[
= \sum_{C} \frac{P(S_{\text{yes}})\sum_{P} P(C | S_{\text{yes}}, P)P(P)}{P(C)}P(C | D_{\text{yes}}) = 0.28651
\]
\[
P(P_{\text{high}} | D_{\text{yes}}) = \sum_{C} P(P_{\text{high}} | C)P(C | D_{\text{yes}}) = \sum_{C} \frac{P(P_{\text{high}})P(C | P_{\text{high}})}{P(C)}P(C | D_{\text{yes}})
\]
\[
= \sum_{C} \frac{P(P_{\text{high}})\sum_{S} P(C | P_{\text{high}}, S)P(S)}{P(C)}P(C | D_{\text{yes}}) = 0.10199
\]
\[
P(X_{\text{yes}} | D_{\text{yes}}) = \sum_{C} P(X_{\text{yes}} | C)P(C | D_{\text{yes}}) = 0.21740
\]

After asking Phil, the doctor finds out that Phil does not smoke. This can be updated into the network.

\[
P(C_{\text{yes}} | D_{\text{yes}}, S_{\text{no}}) = \frac{P(C_{\text{yes}} | S_{\text{no}})P(D_{\text{yes}} | C_{\text{yes}}, S_{\text{no}})}{P(D_{\text{yes}} | S_{\text{no}})}
\]
\[
= \frac{P(C_{\text{yes}} | S_{\text{no}})P(D_{\text{yes}} | C_{\text{yes}})}{\sum_{C} P(D_{\text{yes}} | C)P(C | S_{\text{no}})}
\]
\[
= \frac{P(C_{\text{yes}} | S_{\text{no}})P(D_{\text{yes}} | C_{\text{yes}})}{\sum_{C} \left( P(D_{\text{yes}} | C) \sum_{P} \left( P(C | S_{\text{no}}, P)P(P) \right) \right)} = 0.00626
\]
\[
P(P_{\text{high}} | D_{\text{yes}}, S_{\text{no}}) = \sum_{C} P(P_{\text{high}} | C)P(C | D_{\text{yes}}, S_{\text{no}})
\]
\[
= \sum_{C} \frac{P(P_{\text{high}})P(C | P_{\text{high}})}{P(C)}P(C | D_{\text{yes}}, S_{\text{no}}) = 0.99188
\]
\[
P(X_{\text{yes}} | D_{\text{yes}}, S_{\text{no}}) = \sum_{C} P(X_{\text{yes}} | C)P(C | D_{\text{yes}}, S_{\text{no}}) = 0.10500
\]
Equation (2.4) implies that x-ray should not be affected by the first update. But the independence in the equation depends on the observation of a common parent. Since cancer is not observed, equation (2.4) does not hold.

Dependency between two parents through a common child is opposite from the dependency between two siblings through a common parent. Equation (2.6) says that the parents can be dependent only if their child is observed. In the second update, smoking and pollution is independent given that all their common descendants are not updated. This gives the following theorem:

**Theorem 2** (Rule of D-separation). [13] Let $X_i$ and $X_j$ be two arbitrary nodes in a Bayesian Network $\mathcal{G}$.

$X_i$ and $X_j$ are conditionally independent, or d-separated, if the following holds:

For each undirected path between $X_i$ and $X_j$, there exist at least one node $V$ where one of the following three criteria holds.

- $V$ is in a chain structure in the path and $V$ is already observed.
- $V$ is a common parent in the path and $V$ is already observed.
- $V$ is a common child in the path and neither $V$ nor any of $V$’s descendants are observed.
Chapter 3

Interpretation of an Expert Opinion

3.1 Simplified Information

Because of lack of statistical data in the Åknes project, expert opinion is one of the main sources of information available to construct the tsunamigenic rockslide EWS model. The lack of statistical data reduces the possibility of statistical inference. Using expert opinion as a substitute can be problematic. In this case information is subjective and can sometimes be strongly influenced by how one asks the questions to the expert. Even when the information gathered can be trusted, the problem of interpreting still needs to be addressed. The information needed to construct a model is often not formulated in a way that can be easily interpreted.

As mentioned in the last section, a Bayesian Network is constructed from a graph and a CPT. Getting information to well-define a graph should be straightforward. Look at the following sentence: “Rockslides is the only relevant factor to cause tsunamis.” This sentence can intuitively be interpreted as a graph with two nodes: rockslide and tsunami. An arc can be placed from the former to the latter. Continuing in this fashion the whole network can easily be constructed.

It is possible to construct parts of the network or all of it from statistical data. Some of the theory discussed in this chapter will use techniques that are typically used in classical statistics. However, limited study on this topic is included in this work. For further information about the use of statistical techniques the reader is referred to Korb[13].

When constructing the graph of a Bayesian Network, it is important to understand that the absence of an arc implies that there is no direct dependency between the two nodes.[13] Having an arc on the other hand, even if placed in the wrong direction of the true state of nature, implies almost nothing. An arc does not exclude the possibility of independence.
3.2 Converting to Continuous

The implication of the direction of the arc can be observed in theorem 2. But too many arcs work against the strengths of a Bayesian Network. The graphical representation becomes hard to follow and the network becomes computationally demanding. Thus the goal in the construction process should then be to find as many independencies as possible from a full network, rather than finding as many dependencies as possible from an empty one.

Creating the CPT for the network may not be a simple task. A node with an underlying continuous distribution needs to be defined by a person experienced in probability theory. And even if the expert has such experience in addition to his or her own field of study, there still might not be any way to estimate the distribution parameters.

To generate information that can be actually useful, this should be gathered from experts in a systematic and simple manner. This can be done by requesting the information in a discrete finite set distribution[15]. For instance, look at the following question: “What is the probability for respectively no, small, medium and large tsunami, given a large earthquake?” It is not hard to see it would be easier to answer this question than to describe the difference between a Gaussian and a Gamma distribution, and fitting parameters. The precision of the estimates depends on the number of categories presented to the expert. If more categories are presented, the precision becomes better.

3.2 Converting to Continuous

A good way of understanding the scaled data is to compare it to a histogram. A traditional histogram is generated from samples by using the relative frequencies to get estimates on a set of subintervals. The scaled data is not generated from samples, but from experts, i.e. they both have a source from where they can be defined. The number and length of subintervals are user defined. But having too few, too many, too small or too large intervals gives bad results. The more samples added to a classical histogram, the better defined it is. This can be used to refine the histogram by allowing for including more subintervals. Analogously, the more certain an expert is, the more categories would be natural.

The result of a discrete finite set distribution can be used to guess how a continuous equivalent would look like. This is done by fitting the parameters of an appropriate distribution to the experts’ discrete data. This is much like traditional parameter fitting. The end result can be presented back to the experts for revision. In this way, by having an initial suggestion on a continuous model, better models can be constructed. This process can be repeated until there is some kind of consensus. Generating an expert probability distribution in this way creates a prior network. The network
can be updated using the principles described in section 2.2. How this is done will be discussed in chapter 5.

Fitting continuous probability distributions to categorical relative frequency histograms obtained from experts, is similar to fitting a distribution to a set of samples. There are some distinct differences. The categorical data do not necessarily have a scale. A set of disjoint sorted intervals \( c = \{c_1, \ldots, c_n\} \), have to be created by the experts. The space covered by \( c \) spans over the event space. In this case, let \( l_i \) and \( u_i \) be the infimum and supremum of the interval \( i \). Let also \( I = \{l_1, \ldots, l_n\} \) and \( U = \{u_1, \ldots, u_n\} \). The implementation of the scale is usually not complicated, e.g. the categories no, low, medium and high for the tsunami intensity \( X \) could be expressed as the number of meters above sea level. An expert could respectively divide these into e.g. \( X = 0 \), \( 0 < X \leq 5 \), \( 5 < X \leq 15 \) and \( 15 < X \) meters.

Histograms are often used to help guess what kind of distribution one should try to fit. This can also be applied to expert data. On the other hand, histograms are not designed for estimating parameters. The histogram does not contain as much information as the samples. The analogue to parameter estimation from samples would be the experts giving their assessment on parameter values. In those cases, most parameters are not easily calculated which makes this difficult to apply. Other approaches need to be used.

As a consequence of the central limit theorem, the relative frequency in each bin in a histogram converges towards the probability of the bin. This should be the same for the experts’ opinion. The probability for being in each bin according to the continuous distribution should be the same as the probability of the bin given the expert statement. This can be formalized as a set of equations. \( F_X \) is the continuous distribution’s cumulative function and \( \theta \) is the set of parameters.

\[
P(X \leq u_1|\theta) = F_X(u_1|\theta) \\
P(X \leq u_2|\theta) = F_X(u_2|\theta) \\
\vdots \\
P(X \leq u_n|\theta) = F_X(u_n|\theta) 
\] (3.1)

It is natural to assume that the number of intervals \( n \) is larger than the number of parameters \( m \). This implies that in most cases there are no solutions that satisfy equations (3.1). The closest that can be constructed is a solution that minimizes the difference between the probability, \( P(X \leq u_i|\theta) \) and \( F_X(u_i|\theta) \) for all subintervals \( i \). The minimum sum of squares from regression theory\[17\] is introduced to find an optimum.

\[
Q(\theta) = \sum_{i}^{n} \left( P(X \leq u_i|\theta) - F_X(u_i|\theta) \right)^2 
\] (3.2)

\[
\theta^{opt} = \arg\min_{\theta} \left( Q(\theta) \right)
\]
3.2 Converting to Continuous

Figure 3.1: The probability of going at a certain speed over the speed limit given that the car is going to crash.

**Example 3 (Car accident).** Most cars that accidently crash are driving above the speed limit. Considering only cars that crash while speeding, let \( X \) represent the speed. \( X \) is measured in kilometers per hour above the speed limit when the crash occurs. Since legal speeds are not considered, \( X > 0 \). Let it be assumed that there are no statistics on the speed for crashes, the information has to be gathered from experts, e.g. the police. They conclude that the probabilities for the intervals \( 0 \leq X < 5 \), \( 5 \leq X < 15 \), \( 15 \leq X < 25 \) and \( X \geq 25 \), are respectively 0.3, 0.4, 0.2, 0.1. Filling this into equations (3.1) gives the following.

\[
\begin{align*}
F(5|\theta) &= 0.3 \\
F(15|\theta) &= 0.7 \\
F(25|\theta) &= 0.9 \\
F(\infty|\theta) &= 1
\end{align*}
\]

The last equation is trivially true from the definition of cumulative distributions and can be neglected. The histogram that can be seen in figure 3.1 shows that the distribution start high around 0 and drops after that. This gives reasons to try the exponential and the gamma distribution. Using the exponential distribution gives
Figure 3.2: The sum of squares for the (a) exponential distribution and (b) gamma distribution. (b) is in log-scale where the contours are $e^{-8}$ (inner) until $e^{-1}$.

the following equations.

$$1 - e^{-5\lambda} = 0.3$$
$$1 - e^{-15\lambda} = 0.7$$
$$1 - e^{-25\lambda} = 0.9$$

The sum of squares is as follows (residuals):

$$Q(\lambda) = \left(0.3 - \left(1 - e^{-5\lambda}\right)\right)^2 + \left(0.7 - \left(1 - e^{-15\lambda}\right)\right)^2 + \left(0.9 - \left(1 - e^{-25\lambda}\right)\right)^2$$

In figure 3.2 (a) the sum of squares is plotted as a function of lambda. A minimum can be observed at $\lambda = 0.08056$ with a residual value $Q = 0.00211$.

Since the Exponential distribution is a special case of the Gamma distribution, the Gamma distribution can potentially get better results. Using Gamma instead of Exponential gives the following calculations.

$$F(5|\alpha, \lambda) = 0.3$$
$$F(15|\alpha, \lambda) = 0.7$$
$$F(25|\alpha, \lambda) = 0.9$$

$$Q(\alpha, \lambda) = \left(F(5|\alpha, \lambda) - 0.3\right)^2 + \left(F(15|\alpha, \lambda) - 0.7\right)^2 + \left(F(25|\alpha, \lambda) - 0.9\right)^2$$

In figure 3.2 (b) the sum of squares is contour plotted as a function of $\lambda$ and $\alpha$. To be able to observe the minimum, the scale is logarithmic. The minimum $Q = 0.00027$ which is as expected lower than with the exponential fit. The minimum of the residual can be found at $\alpha = 1.24683$ and $\lambda = 0.10391$. 

22
The residual for the Exponential distribution is about ten times larger than of the Gamma distribution. The choice of distribution is important for a good fit. Therefore, testing available (default) options of probability distributions is always a possibility.

3.3 Fit of Dependent Variables

The method described before converts an independent probability distribution from discrete to continuous. In an influence diagram most nodes are not independent. Since the discrete structure is finite spaced, constructing distinct probability functions one at a time for each dependent distribution is possible. Extending the model in example 3, if one node represents the speed, a child could represent the possibility of fatality connected to the crash. One distribution for each interval in the speed node has to be constructed for the fatality node. Each distribution requires its own set of parameters. Each dependent distribution summarizes the fatality distribution over it’s subinterval. Similar to the discrete case, the state of the parents is required to determine the behavior of a node.

Because the parent also is continuous, more information is passed then in the discrete case, e.g. with discrete nodes, on what interval the state of a node is on is passed on to the children. With continuous nodes, the actual state is passed on. But since one distribution is constructed for each interval, the accurate state is discarded and the interval is used instead. In addition to this loss of information, the discontinuities between the intervals are a problem. E.g. the difference between the speed 14.99 and the 15.01 is insignificantly small, but the impact since they are part of separate intervals can be very large. When converting all variables, a smooth transition is expedient. This can be done by inserting the state value of parents as part of the parameters, \( \theta_X = \theta_X(par(X)) \), i.e. instead of creating four different fatality distributions, one with a set of flexible parameters is enough.

Equations (3.1) are the same for independent and dependent variables. The difference is that the parameters for the dependent nodes are dependent on the state of their parents. These take the following form:

\[
F_X(u_i|\theta(par(X))) = P(X \leq u_i|par(X) \in v_j) \quad (3.3)
\]

Here \( u_i \) is the supremum of the interval \( i \) for node \( X \), and \( v_j \) is the interval \( j \) in the node \( par(X) \). If \( par(X) \) is more than one node, \( v_j \) is a vector of intervals and \( j \) traverses through all combinations.

The sum of squares \( Q \) from equations (3.2) will produce a vector when used on equation (3.3). It can not be used to determine minimum. To
Table 3.1: The conditional probability table for fatality in car crashes given
the speed of the car.

<table>
<thead>
<tr>
<th>A</th>
<th>[0,5]</th>
<th>[5,15]</th>
<th>[15,25]</th>
<th>[25,∞]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.8</td>
<td>0.7</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>[0,2]</td>
<td>0.1</td>
<td>0.2</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>[2,4]</td>
<td>0.05</td>
<td>0.05</td>
<td>0.15</td>
<td>0.2</td>
</tr>
<tr>
<td>[4,∞]</td>
<td>0.05</td>
<td>0.05</td>
<td>0.1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

determine fit on all distributions, a sum over subintervals can be added:

\[
Q(\theta) = \sum_{i}^{n} \sum_{j}^{m} \left( P(X \leq u_i | \text{par}(X) \in v_j) - F_X(u_i | \theta(\text{par}(X))) \right)^2 \tag{3.4}
\]

\[
\theta^{opt} = \arg\min_{\theta} \left( Q(\theta) \right)
\]

Finding good guesses for the form of \( \theta(\text{par}(X)) \) can be done in more than
one way. If no changes are to be applied to the parameter, the function
would be piecewise constant on each subinterval. The function will for
all intents and purposes resemble a piecewise constant interpolation. If
the behavior of a parameter over the set of intervals looks familiar, an
interpolation function can be guessed and tried. One of the simplest ways to
interpolate is to create a linear spline over the center value on each interval.
This can create large discontinuities in the first derivative at each center
value. It is not investigated in this work if the linear spline interpolation
will be improved by using a quadratic or cubic spline. This is though
beyond the scope of this thesis. For further information about interpolation
the reader is referred to Denison[7].

As a side note it is worth mentioning that parameter functions are not
the only way for letting the information about evidence travel through the
network. It is also possible to let information travel through a function
transformation of the parent.[16], i.e. if \( X \) is the only parent of \( Y \), the value
of \( Y \) can be defined as the value of a function \( g(X, U) \), where \( U \) is a random
variable that explains \( Y \)'s random property that is not explained by \( X \). This
text will only focus on dependency traversing through parameters.

Example 4 (Car accident II). Continuing example 3, let the speed of the driver
be formulated as node \( A \). Let the number of casualties in the accident be defined
as node \( B \), child of node \( A \). From fitting, \( A \) is gamma distributed with parameters
\( \alpha = 1.24683 \) and \( \lambda = 0.10391 \). Let the conditional probabilities generated from
experts for \( B \) be given in table 3.1.

One of the conditions not fulfilled with table 3.1 is that there is a non-zero
probability that the number of fatalities is exactly 0, even though it is still non-
negative. The definition of a continuous distribution requires that the probability
of any single value is 0. There are different ways of getting around this. One solution is to make the start location for the distributions a parameter that also has to be estimated. By moving the start location over on the negative axis, the cumulative value of 0 can be anything. To avoid going outside the intervals of definition, left censoring in 0 is used, i.e. all negative values are interpreted as 0. A consequence of adding a parameter, is that the distributions become more flexible, which potentially can reduce the sum of squares in equation (3.2).

The shape of the distribution still looks like an Exponential or gamma distribution. The equations that need optimizing under the exponential distribution are defined as follows.

\[
F_B(5|\lambda(v_1), \text{loc}(v_1)) = 0.8 \quad F_B(15|\lambda(v_1), \text{loc}(v_1)) = 0.9 \\
F_B(25|\lambda(v_1), \text{loc}(v_1)) = 0.95 \quad F_B(\infty|\lambda(v_1), \text{loc}(v_1)) = 1 \\
F_B(5|\lambda(v_2), \text{loc}(v_2)) = 0.7 \quad F_B(15|\lambda(v_2), \text{loc}(v_2)) = 0.9 \\
F_B(25|\lambda(v_2), \text{loc}(v_2)) = 0.95 \quad F_B(\infty|\lambda(v_2), \text{loc}(v_2)) = 1 \\
F_B(5|\lambda(v_3), \text{loc}(v_3)) = 0.5 \quad F_B(15|\lambda(v_3), \text{loc}(v_3)) = 0.75 \\
F_B(25|\lambda(v_3), \text{loc}(v_3)) = 0.9 \quad F_B(\infty|\lambda(v_3), \text{loc}(v_3)) = 1 \\
F_B(5|\lambda(v_4), \text{loc}(v_4)) = 0.3 \quad F_B(15|\lambda(v_4), \text{loc}(v_4)) = 0.55 \\
F_B(25|\lambda(v_4), \text{loc}(v_4)) = 0.75 \quad F_B(\infty|\lambda(v_4), \text{loc}(v_4)) = 1 \quad (3.5)
\]

Here \( \text{loc} \) is the location where the distribution starts and \( \{v_j\}_{j=1}^4 = \{[0,5], [5,15], [15,25], [25,\infty]\} \). Since \( F_B(\infty) = 1 \) is trivially true due to probability calculus\[17\], these equations can be neglected. The solution is as follows:

\[
\begin{align*}
\lambda &= 2.87115 & \text{loc} &= -4.61866 & \forall \; B \in v_1 \\
\lambda &= 1.96541 & \text{loc} &= -2.37567 & \forall \; B \in v_2 \\
\lambda &= 2.66681 & \text{loc} &= -1.83229 & \forall \; B \in v_3 \\
\lambda &= 3.92571 & \text{loc} &= -1.34369 & \forall \; B \in v_4
\end{align*}
\]

By defining one distribution for each subinterval, \( \lambda \) and \( \text{loc} \) are piecewise constant on the set of intervals. \( \lambda \) and \( \text{loc} \) as a piecewise constant function of the speed can be observed in figure 3.3. To avoid discontinuities, the piecewise constant function can be replaced. A linear spline interpolated through the middle value of each interval can be observed in the figure as a dashed line. Infinity is replaced with a suitable finite number.
Figure 3.3: (a) $\lambda$ and (b) $\text{loc}$ illustrated as a piecewise constant function and a linear spline.
Chapter 4

Model Construction

4.1 Base Model

As part of the Åknes project, a conditional probability table was generated from the experts involved for a simplified model.[9] The suggested structure can be observed in figure 4.1. It consists of four nodes: Rockslide, Tsunami, Season and Consequence. The Rockslide node defines the probability of the rockslide failure within the next year. It is measured in $10^6$ cubic meter volume. The model only considers a rockslide that is large in size. The Tsunami node is dependent on the Rockslide and is measured as run up distance in meters above sea level. The Season node is binary and adds or removes the effect of tourist season. The last node, Consequence, measures the loss of life given a Tsunami and Season.

The probability tables vary in different locations because of topography and population. The conditional probabilities in table 4.1 are based on

![Figure 4.1: A simplified graph modeling a tsunami threat[9].](image-url)
Table 4.1: Expert generated conditional probability tables for figure 4.1 for the Hellesylt area.

numbers from Hellesylt. The calculation structure is however applicable on all locations.

All nodes are by construction positively correlated. It implies that high values in one node, leads to high values in other nodes, e.g. if Rockslide is high, Tsunami, Consequence, Sensor and EWS are all expected to be high as well.

The sum of squares, $Q$, defined in equations (3.2) and equations (3.4) are used to find a suitable distribution. This thesis will focus on piecewise constancy on the subintervals of the parents.

Four probability distributions are used to fit the marginal relative fre-

<table>
<thead>
<tr>
<th>Consequence (C)</th>
<th>T</th>
<th>S</th>
<th>0</th>
<th>(1,3]</th>
<th>(3,10]</th>
<th>(10,30]</th>
<th>(30,60]</th>
<th>(60,100]</th>
<th>(100,300]</th>
<th>(300,∞)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Y</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(0,1]</td>
<td>Y</td>
<td>0.99</td>
<td>0.01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(1,5]</td>
<td>Y</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
<td>0.15</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.25</td>
<td>0.4</td>
<td>0.25</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(5,10]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.15</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.15</td>
<td>0.05</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(10,15]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>(15,20]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.3</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>(20,25]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>(25,30]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(30,∞]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tsunami Run Up (T)</th>
<th>R</th>
<th>No</th>
<th>(0,0.5]</th>
<th>(0.5,2]</th>
<th>(2,4]</th>
<th>(4,7]</th>
<th>(7,12]</th>
<th>(12,20]</th>
<th>(20,35]</th>
<th>(35,∞)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td></td>
<td>(0,0.5]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(0.5,2]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(2,4]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(4,7]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(7,12]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(12,20]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(20,35]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(35,∞]</td>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0.999</td>
<td>0.001</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rockslide (R)</th>
<th>0</th>
<th>(0,0.5]</th>
<th>(0.5,2]</th>
<th>(2,4]</th>
<th>(4,7]</th>
<th>(7,12]</th>
<th>(12,20]</th>
<th>(20,35]</th>
<th>(35,∞)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
<td>0.0004</td>
<td>0.0007</td>
<td>0.0007</td>
</tr>
<tr>
<td>0.05</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td>0.0158</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td>0.0032</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td>0.0015</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td>0.0009</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td>0.0006</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td>0.0004</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td>0.0007</td>
<td>0.25</td>
<td>0.75</td>
<td>0.9269</td>
<td>0.05</td>
<td>0.0158</td>
<td>0.0032</td>
<td>0.0015</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
</tbody>
</table>

Chapter 4. Model Construction
frequency histograms shown on table 4.1: Gamma, Gaussian, Inverse-gamma and Inverse-gaussian. The parameters of the respective distributions are ($\alpha$, $\lambda$, loc), ($\sigma^2$, loc), ($\alpha$, $\beta$, loc), ($\mu$, $\lambda$, loc). To increase robustness into the modeling, the loc-parameter is added to shift the distribution to the left or to the right. (For the Gaussian distribution loc equals $\mu$.)

4.1.1 Rockslide

The Rockslide event is defined in figure 4.1 as a root node, which makes the process of fitting a distribution simple. This has a (fairly large) positive probability of being 0 for a non-negative distribution. Sensoring is used to avoid getting samples smaller than 0. Since $Q$ is defined by the cumulative probability in a set of reference points where all are non-negative, the censoring will not affect $Q$. The integral from $-\infty$ to 0 for a continuous distribution is the same as a single point probability in 0, if the probability value for them both are the same.

The relevant probabilities from table 4.1 are inserted into equations (3.2). Minimizing the sum of squares over the four distributions gives the following parameters and $Q$.

**Gamma distribution**
\[ \alpha = 0.01297, \lambda = 0.21084, \text{loc} = -0.00776 \]
\[ Q = 7.19344\times10^{-6} \]

**Gaussian distribution**
\[ \sigma^2 = 0.00973, \text{loc} = -0.01411 \]
\[ Q = 0.00061 \]

**Inverse Gamma distribution**
\[ \alpha = 0.82923, \beta = 0.00492, \text{loc} = 0.00372 \]
\[ Q = 0.85914 \]

**Inverse-gaussian distribution**
\[ \mu = 0.88752, \lambda = 43.16288, \text{loc} = 0.00144 \]
\[ Q = 0.85922 \]

The Gamma distribution has the lowest $Q$ and will be used. In figure 4.2 the probabilities and the fitted continuous distribution are plotted together. The latter follows the jagged contour of the former, illustrating a good fit.

Even though there are distributions where loc is positive, it is expected that loc is smaller than 0 for good solutions. The reference requires that the cumulative distribution is non-zero in 0. To get the lowest reference to fit, the distribution can not be 0 for non-positive values.
4.1.2 Tsunami

Tsunami, like Rockslide, fulfills the requirement for using left censoring at 0. Tsunami is dependent on Rockslide and will generate a vector of parameters for each distribution. Equation (3.4) is used as a measure to determine the fit. The results are as follows:

\[
\begin{align*}
\text{Gamma:} & \quad Q = 0.00518 \\
\text{Gaussian:} & \quad Q = 0.00741 \\
\text{Inverse-gamma:} & \quad Q = 1.01036 \\
\text{Inverse-gaussian:} & \quad Q = 0.43292
\end{align*}
\]

Similarly to example 3 discussed in section 3.2 the Gamma distribution has a lower sum of square than the other distributions. The probability of having exactly 0 meters of run up in Tsunami given presence of a Rockslide is 0. This implies that the need for estimates for the distribution for \(T|R = 0\) is not required. If \(R = 0\), then \(T = 0\). In figure 4.3 the different parameters from the gamma distribution are plotted against the value of \(R\). \(R = 0\) is not plotted.
As mentioned earlier Rockslide and Tsunami are by definition positively correlated. Large Rockslide usually gives large Tsunamis. This is reflected in the $\text{loc}$ parameter. $\text{loc}$ defines where the distribution starts by shifting the whole distribution to the left or the right. In figure 4.3 it starts low and grows quickly implying that the distribution more and more is shifted to the right.

Gamma distribution’s expected value is $\frac{\alpha}{\lambda}$. This implies that to maintain the same expected value, the ratio between $\alpha$ and $\lambda$ must be constant. This requirement explains the high top on both $\alpha$ and $\lambda$ for low values of Rockslide in figure 4.3.

### 4.1.3 Season

Season is a root node with only two states. This node is kept discrete since its sample space is binary. Making a continuous estimate of the node will not improve the model, but make it more complex.

Except for Season, all other nodes have an intuitive definition of high and low. By observing the relationship between Season and Consequence, it is logical to define Tourist Season as high and Non-tourist Season as low. This because Tourists drive the number lives lost up in case of a large Tsunami.

### 4.1.4 Consequence

The last node is Consequence. It is the only node with two parents. Since Season only has two discrete states, constructing the Consequence node can be done simpler by doing it in two rounds: One for Tourist Season and one for Non-tourist Season. Using equation (3.4) again, the four distributions are tested for different fitting distributions, as before. The result for within Tourist Season is as follows:

- Gamma: $Q = 0.00500$
- Gaussian: $Q = 0.01199$
- Inverse-gamma: $Q = 1.98369$
- Inverse-gaussian: $Q = 0.13093$

And for outside Tourist Season.

- Gamma: $Q = 0.00193$
- Gaussian: $Q = 0.00352$
- Inverse-gamma: $Q = 2.00423$
- Inverse-gaussian: $Q = 0.06937$

The Gamma distribution has lower sum of squares for both season conditions, and will be used for further analysis in the following sections. Gamma is the best distribution in the four fittings done so far. Figure 4.4 plots Consequence against Tsunami in and outside Tourist Season.
Figure 4.3: In Tsunami, the parameters (a) $\alpha$, (b) $\lambda$ and (c) $loc$ plotted against the state of Rockslide.
Figure 4.4: Consequence respectively inside and outside Tourist Season, the parameters (a & b) $\alpha$, (c & d) $\lambda$ and (e & f) $loc$ plotted against the state of Tsunami.
4.2 Early Warning System

The four nodes Rockslide, Tsunami, Season and Consequence models together represent the Åknes threat. But it does not take into account the effect of an Early Warning System. The goal is to look at the effect of intervening for reducing the expected risk. For the EWS influence to be included, the model needs to be expanded. In figure 4.5 the Early Warning System and the original model are brought together into a new extended model. The stippled line separates the new nodes from the old. The new nodes are: Sensor (S), Threshold (θ), Early Warning System (EWS) and Loss (L). Rockslide is parent to Sensor. Sensor, Season and Threshold are parents to Early Warning System. Early Warning System and Consequence are parents to Loss.

4.2.1 Sensor

The Sensor node (S) is dependent on Rockslide and is measured in the same scale as Rockslide: $10^6 \text{ m}^3$. It measures potential threat factors that can trigger a rockslide. If the potential threat factors, abbreviated PTF could predict a rockslide perfectly, Sensor would be a copy of Rockslide. But since there are amounts of uncertainty associated with the PTF, there is discrepancy between Rockslide and Sensor. Sensor is therefore a
distribution dependent on Rockslide.

This work will not discuss how well information about a rockslide can be observed by sensors, or what distributions should be used. Gaussian distribution with \( \mu = R \) is used to model Sensor. It is used since the expected value matches, but the content may deviate dependent on \( \sigma^2 \).

The variance \( \sigma^2 \) has to be adjusted for practice and is in this work treated as a constant. Since Rockslide is non-negative, Sensor is censored at 0. This implies that there is a positive probability that the Sensor picks up a threat, even though there are none. If this feature is unwanted, it can be removed, by making 0 an absorbent state, this means that if Rockslide is 0, Sensor will also be 0. Sensor’s behavior will be discussed in more detail in section 5.1.3.

The choice of distribution needs not be Gaussian. Since the values are non-negative, it may be just as intuitive to use for example a log-normal distribution. The parameter \( \sigma^2 \) in the Gaussian distribution may also be dependent upon Rockslide, e.g. \( \sigma^2 = \sqrt{R} \). Gaussian distribution with constant variance is only chosen for reason of simplicity.

In reality, Sensor does not measure Rockslide, it only measures potential triggering factors. The way the model is constructed, Rockslide sends information to Sensor. To take account for this an additional node can be inserted as a common parent of Rockslide and Sensor and the arc between the two can be removed.

### 4.2.2 Early Warning System

The Early Warning System (EWS) is defined as the node that decides what state of emergency the system is in. It contains five discrete states. They are from low to high state of emergency: Green (g), Blue (b), Yellow (y), Orange (o) and Red (r). The criteria for deciding a state of emergency is based on observable information. In the model this is Sensor and Season. There is therefore arcs from these two nodes to EWS.

### 4.2.3 Threshold

The EWS state depends on how sensitive the system is. The thresholds that decide the sensitivity of the EWS is for structural reasons separated out and added to the model as a unique node: Threshold. Threshold (\( \theta \)) is a decision node and is defined as follows.

\[
\theta : \text{Sensor and Season} \mapsto \{0, 1, 2, 3, 4\}
\]

\( \theta \) is a deterministic function that maps observable information and into five categories. 0 through 4 respectively represent green, blue, yellow, orange and red state of emergency. In general \( \theta \) is a function from the definition values of EWS’s parents. In practice this is Sensor and Season.
Threshold is a function to be decided, while EWS is the implementation of that function. This means that EWS gives a state of emergency dependent on what rules are given from Threshold. One of the main questions the model will try to answer is what the Thresholds function should look like, i.e. how much threat should be observed by sensors before the system should act and trigger a different state of emergency?

To maintain positive correlations between EWS and the rest of the network, it is natural to assume that $\theta$ is a monotonically increasing function, i.e. let $X = \{X_1, \ldots, X_p\}$ and $Y = \{Y_1, \ldots, X_p\}$ be elements in Sensor and Season and $X_i > Y_i$ for all $i$. Monotonicity then implies that $\theta(X) \geq \theta(Y)$.

$\theta$ can also under this assumption be interpreted as a multistate system. Let then $\theta$ consist of the following.

$$\theta = \sum_{i=1}^{4} \theta_i \quad \theta_i : \text{Sensor and Season} \mapsto \{0, 1\}$$

$\theta_i$ is a monotonically increasing binary function. It is the function that defines the individual thresholds, i.e. $\theta_1$ define where the border between green and blue state of emergency lies. Likewise $\theta_2$, $\theta_3$ and $\theta_4$ respectively defines where the border between blue and yellow, yellow and orange and orange and red lies. To ensure that this stays consistent, $\theta_i$ is a (weakly) decreasing sequence.

Given that Sensor and Season consist of a single node, $\theta_i$ can uniquely be defined by a constant. Because of the monotonicity there exists a $c_i$ such that $\theta_i$ is 1 for values larger than $c_i$ and 0 for smaller values than $c_i$. Let $c = \{c_1, c_2, c_3, c_4\}$ be the set of constants uniquely defining $\theta$. To retain $\theta$'s restrictions, $c_1 \leq c_2 \leq c_3 \leq c_4$. The problem is in other words reduced to finding a sequence of increasing constants. The constants are the borders between the different states of emergency.

The connection between EWS, Sensor, $\theta$ and $c$ can be seen in figure 4.6. For each value in Sensor, the function $\theta$ maps a value into EWS. $\theta$ is piecewise constant between $c_i$ for $i = 1, 2, 3, 4$.

Expanding the construction of Threshold to be able to handle more complex situations are beyond the scope of this thesis. In general $\theta_i$ is a hyper surface in the span of Sensor and Season and bounded by the rules of $\theta$. A suggestion to circumvent this problem is to expand the EWS to consist of more than one node. The number of parents can be reduced through a system of nodes ending in a single leafnode: Threat. By letting this Threat represent the probability of a threat (e.g. Rockslide) or believed results (e.g. Consequence) and letting it be EWS' only parent beyond Threshold, the search for $\theta$ will in general be reduced to a search for four constants. But
4.2 Early Warning System

![Graph showing \( \theta \) as a function from Sensor to EWS.]

Figure 4.6: \( \theta \) as a function from Sensor to EWS.

Doing such a simplification will potentially discard relevant information (like Season). Finding a general solution for hyper surfaces is therefore favorable, but rather out of the scope of this work.

EWS has three parents: Sensor, Season, and Threshold. This implies that \( \theta \) must take account of two nodes, which in general implies that \( \theta \) is a function of two variables. But since Season is binary, the problem can be reduced to finding two sets of increasing constants: One set of thresholds for Sensor in Tourist Season and one set of thresholds for Sensor in Non-tourist Season.

Based on the scale of Rockslide, \( c \) is will most likely be spread out on the interval \([0, 40]\). Let \( c = \{2, 8, 16, 20\} \) be the threshold decided for \( \theta \) in Tourist Season. If now the sensor receives the value 10 from Sensor during Tourist Season, \( c \) defines EWS as 2 i.e. yellow, since 10 lies between threshold two and three. Likewise if the value is 1 in Tourist Season, EWS is 0 or green. This will uniquely define the sensitivity of the system. Consider the two following extreme examples. If \( c \) is low, e.g. \( \{0, 1, 2, 3\} \) the system is oversensitive and will move up to red state with samples as low as 3. This will make red state very likely. On the other hand if \( c \) is high, for instance \( \{30, 35, 40, 45\} \) the system is undersensitive and will not reach blue state before Sensor reaches 30. This will give the opposite effect. Red state
becomes highly unlikely and green becomes (even more) dominant.

- c placement is essential for creating a good EWS. A function to decide a good placement must be introduced. This will be addressed in the next subsection.

4.2.4 Loss

The last node added to the model is the Loss node ($L$). It is a utility node dependent on Consequence and EWS, and measures how good the early warning given Threshold is. The threshold level is good if the early warning matches well with consequences. This is, if low Consequence is accompanied with low level of emergency, or high Consequence is accompanied with a high level of emergency. If this is the case, the system is working well and $L$ should be low, close to the minimum cost imposed by the implementation of the EWS. If on the other hand the match is bad, i.e. low Consequence with high emergency level or vice versa, $L$ must be high. The type of discrepancy can also influence the size of $L$. For example, high Consequence and low emergency could be penalized harder than low Consequence and high emergency. In the Åknes case the expected Loss function should be defined based on the stakeholders demands. The following analyses a systematic approach they can follow.

<table>
<thead>
<tr>
<th>Color</th>
<th>Lower</th>
<th>Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Blue</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Yellow</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>Orange</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>Red</td>
<td>45</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

Table 4.2: Acceptable upper and lower bounds for $C$ in different states of EWS.

- Given that the EWS attempts to capture the Åknes rockslide state of nature, $L$ is expected to be 0. To define the EWS’ state of probability, it has to have a set of intervals in $C$ where $L$ is 0. This is where the system works perfectly. Table 4.2 displays such a suggestion that will be used. The 99, the 99.9 and the 99.99 percentile of C is respectively 13.32721, 159.42296 and 402.20302. The scale is adjusted to these numbers.

- In figure 4.7 a general loss function is plotted. Outside the area where $L = 0$, $L$ should penalize progressively. It is 0 from the bounds and grow monotonically. The function defining how it grows can have any form but should start at 0 and grow monotonically.

- Let $\overline{C}$ and $\underline{C}$ be respectively upper and lower bounds defined in table 4.2. To separate overestimations from underestimations, $L$ is defined as two
functions: \( L_{\text{lower}}(C - C) \) and \( L_{\text{upper}}(C - \bar{C}) \). \( L_{\text{lower}} \) is the function used when \( C \) is lower than \( \bar{C} \) and \( L_{\text{upper}} \) is used when \( C \) is larger than \( \bar{C} \).

Since the argument of the functions are respectively \( C - C \) and \( C - \bar{C} \) the requirement of \( L \)'s shape is reduced to the following. \( L_{\text{lower}} \) and \( L_{\text{upper}} \) are monotonically increasing functions and \( L_{\text{lower}}(0) = L_{\text{upper}}(0) = 0 \). Monotonicity implies that the further away from optimal placement \( C \) is, the higher \( L \) is. That the function starts in 0 implies that the loss function is continuous. If for example \( L_{\text{lower}}(0) \neq 0 \), then \( L \) will be discontinuous at \( \bar{C} \).

Examples of loss functions are as follows, sorted from non-aggressive to aggressive: constant \((a \ a > 0)\), logarithmic \((a \log(bx + 1) \ a, b > 0)\), linear \((ax \ a > 0)\), squared \((ax^2 + bx \ a, b > 0)\) and exponential \((a(e^{bx} - 1) \ a, b > 0)\). When using an exponential function, the value of loss is expected to be much higher then for a constant one. How the different components affect optimal loss will be discussed in the next chapter.
Chapter 5

Monte Carlo Simulation

5.1 Introduction to sampling

The main reason for constructing a model is to do inference. Doing analytical inference in a continuous influence diagram is possible, but for the most part unfeasible. An example will be used to demonstrate the difficulties. Consider the following problem. Let node $X$ be parent to node $Y$ in a two node network. $X$ and $Y$ are respectively uniformly distributed on the interval $[10, 20]$ and Exponential with parameter $\lambda = X$.

\[
\begin{align*}
    f_X &= \frac{1}{10} \quad \forall \ x \in [10, 20] \\
    f_{Y|X} &= xe^{-xy} \quad \forall \ x, y > 0
\end{align*}
\]

The following calculations finds $Y$’s marginal distribution.

\[
\begin{align*}
    f_Y &= \int_{10}^{20} \frac{x}{10}e^{-xy} \, dx \\
    &= \left[-\frac{x}{10y}e^{-xy}\right]_{10}^{20} - \int_{10}^{20} \frac{1}{10y}e^{-xy} \, dx \\
    &= \left[-\frac{x}{10y}e^{-xy} - \frac{1}{10y}e^{-xy}\right]_{10}^{20} \\
    &= e^{-10y}\left(\frac{1}{y} + \frac{1}{10y}\right) - e^{-20y}\left(\frac{2}{y} + \frac{1}{10y}\right) \\
    &= e^{-10y}\left(\frac{1}{y} + \frac{1}{10y}\right) - 2e^{-20y}\left(\frac{1}{y} + \frac{1}{10y}\right) \\
    &= e^{-10y}\left(\frac{1}{y} + \frac{1}{10y}\right) - 2e^{-20y}\left(\frac{1}{y} + \frac{1}{10y}\right)
\end{align*}
\] (5.1)

Even though $X$ and $Y|X$ consist of simple distributions, $Y$’s marginal distribution is complex. Complexity can be driven up quickly in a number of ways. If one of the CPT distributions were more complex, e.g. $X$ were Gamma distributed, the above calculations could not have been written except on integral form. The calculations would be more complex if different distributions were used for different subintervals or more nodes were introduced. Structures more complex than chains, will also contribute.
Let the Åknes project be used as example: For a network with eight nodes with many Gamma distributed nodes with dependent distributions split into subintervals, analytical results are not within grasp. This section will address the use of an approximative method to do inference. There are more than one way to e.g. calculate the marginal of the distributions in a network. This thesis will take a look at the Monte Carlo simulation technique and how it can be used to do complex inference.

Because of how probabilistic influence diagrams are constructed each node has a uniquely defined probability distribution given the state of their parents. Under this assumption random samples can be generated from a number generator that behaves like the individual nodes. If Rockslide is observed to be $10 \times 10^6 m^3$, Tsunamis distribution is uniquely defined. Making such samples can be referred to as creating realizations of nodes. Because of the node’s random property, realizations will change value from sample to sample, e.g. if Tsunami generates a high wave in one realization, it might not in another.

This technique can be applied to a whole network of nodes. The basics is demonstrated in the following example. Let again $X$ be parent to $Y$ in a two node network. Since $X$ has no parents, samples can be generated from it through a random number generator. $Y$ does not have a known distribution unless $X$ is known. But by making a realization of $X$, $Y$ becomes uniquely defined making it possible to create a realization for it as well. Using realization from the parent to generate samples can also be used on more complex structures. Any child can be realized after a sample is generated for all of its parents. Since all networks are finitely spaced and non-cyclic, it is always possible to propagate through the whole network[13].

Let $X_1, \ldots, X_m$ represent all nodes in a network $G$ with $m$ nodes. Let $X_{i_1, \ldots, i_n}$ be i.i.d. $X_i$ for all $i$.

Furthermore let $Z$ be the joint distribution of all nodes in $G$. $Z$ is referred to as a system sample. Let $Z = \{Z_1, \ldots, Z_n\}$ be i.i.d. system samples. The joint cumulative distribution over all variables can be approximated with the following equation:

$$F_Z(x_1, \ldots, x_m) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_m} f_Z(t) \, dt \approx \frac{1}{n} \sum_{i=1}^{n} I(X_{1,i} \leq x_1, \ldots, X_{m,i} \leq x_m) \quad (5.2)$$

Here $I$ is an indicator function dependent on the validity of its argument. With an estimate on the joint distribution, almost any type of inference can be done. A marginal distribution of any node can be retrieved with the following expression:

$$F_{X_i}(x_j) = \int_{-\infty}^{x_i} \cdots \int_{t_j \leq x_j} f_Z(t) \, dt \approx \frac{1}{n} \sum_{i=1}^{n} I(X_{j,i} \leq x_j) \quad (5.3)$$
This is true due to the fact that any marginal distributions \( F_X(x) \) can directly be gathered from a joint distribution: \( F_{X,Y}(x, \infty) \).

The same logic can be applied to any subset of \( Z \). The following equation demonstrates how this can be done for two variables.

\[
F_{X_j, X_k} = \int \cdots \int_{t_j \leq x_j, t_k \leq x_k} f_Z(t) \, dt \approx \frac{1}{n} \sum_{i=1}^{n} I(X_j \leq x_j, X_k \leq x_k) \quad (5.4)
\]

In figure 5.1 the distribution defined in (5.1) is estimated using equation (5.3). The different subplots illustrate how the accuracy increases with the number of samples.

Algorithm 1 applies Monte Carlo simulation on a general network[13].

**Algorithm 1** (Monte Carlo simulation applied on a probabilistic influence diagram). Let \( G \) represent a non-cyclic probabilistic influence diagram where the CPT is known.

1. Using a Random Number Generator, generate a sample for all rootnodes.
2. For each node that meets the following criteria:

- Node has not been realized.
- All parents of the node are realized.

Generate a sample for the node.

3. Repeat step 2 until all nodes are realized.

The set of all samples in the network is an i.i.d. system sample.
Repeat the algorithm to generate more system samples.

Because the system samples are i.i.d., most of the statistical toolbox is available. First and second order statistics, i.e. mean, variance, covariance and correlation are available from estimates. And again, the accuracy can be arbitrarily high.

5.1.1 Base Model

Applying algorithm 1 and equation (5.2) to the node Rockslide can be observed in figure 5.2.a. The distribution and parameters considered in the model are those given in chapter 4. The Monte Carlo sampling includes $10^7$ samples for each node.

The Rockslide node is censored at 0 and has a probability of 0.9269 for being 0. The cumulative distribution confirms this behavior by starting at 0.

The empirical marginal distribution of Tsunami can be observed in figure 5.2.b. Because of the direct dependence to Rockslide, it also starts at 0.9269. Except for a jump at around 0.05, the slope is smooth. Looking back at figure 4.3, the parameters defining Tsunami, $\alpha$ and $\lambda$ are volatile for low values of Rockslide. The discontinuity in parameters is the reason for the discontinuity in the marginal. By removing the big jumps in the parameters solves the problem. It has not been investigated, but using a linear interpolation instead could help to improve the fit.

Figure 5.3 shows an estimation of Consequence’s distribution dependent on Season, but independent of Tsunami. Non-Tourist and Tourist Seasons are shown in separate subfigures. Both begin at 0 at around 0.985 and grows slowly towards 1. As expected, the slope climbs quicker during Tourist Season. Figure 5.4 shows a marginal estimation of Consequence.

5.1.2 Logic Sampling

As mentioned in section 2.2, a probabilistic influence diagram becomes a Bayesian Network if Bayes’ theorem is applied to let information propagate both with and against the direction of the arc. To sample in Bayesian Network, Logic Sampling is introduced[13].
Figure 5.2: Empirical marginal distributions of (a) Rockslide and (b) Tsunami.
Figure 5.3: Empirical distributions of the node Consequence in (a) Non-Tourist Season and (b) Tourist Season.
Algorithm 2. [Logic Sampling algorithm]
Let $\mathcal{G}$ represent a Bayesian Network where the CPT is known.
Let $X = \{X_1, \ldots, X_n\}$ be the nodes in $\mathcal{G}$.
Let $E = \{E_1, \ldots, E_n\}$ be information inserted into $X$. $E_i$ is an interval for which $X_i$ may be in. If no information is inserted, $E = \mathbb{R}$.

For each round of simulation:

1. For all root nodes, randomly choose a value given evidence $E$.
2. Loop
   Choose values randomly for children, using the conditional probability given the known values of the parents.
   Until all the leaves are reached and a system sample $Z$ is generated.
3. If $Z$ is not an element in $E$, then discard $Z$. Otherwise retain it.

The system samples $Z$ generated from this process are identically distributed conditional on $E$.

The algorithm will require the generation of more samples to converge because of the discarding of samples. If $E$ is very unlikely, the need for extra samples increases. The number of samples will be unfeasibly high.
if $E_i$ is a single point and the corresponding $X_i$ is continuous. As means to go around this problem, a single point can be substituted with an open interval around the point. Algorithm 2 is will give bad results, if the overall evidence is very unlikely to occur.

Evidence added to root nodes are not affected by the discarding. The algorithm generates adjusted samples directly for these nodes.

### 5.1.3 Sensor Information

In subsection 4.2.1 Sensor was defined as a Gaussian variable with $\mu = R$ and $\sigma$ a constant error term. The size of $\sigma$ decides how accurate the measurements in the system are. If $\sigma$ is high, hitting the right emergency state becomes difficult. Optimally $\sigma = 0$ and the information is precise. In practice this may not be the case. This subsection discusses how changes to Sensor will affect the correlation between Sensor and Consequence.

The basis of the simplified model is that by reading information from the network, it is possible to say something about the probable outcome. If there is no connection between the observable information (Sensor) and the outcome (Consequence), the inference becomes useless. The model can not tell anything about how the outcome will be.

In figure 5.5.a the correlation between Sensor and Consequence is plotted with the standard deviation $\sigma$ as variable. It is created from samples generated with equation (5.4). As can be observed, the correlation starts at 0.6 but drops towards 0 as $\sigma$ grows. For example $\sigma = 1$ gives correlation 0.3, half of the starting point.

In practice the only relevant samples are the ones where there is recorded activity. If $R = 0$, then $C = 0$ and $S \approx 0$. By using algorithm 2 it is possible to only observe events where $R > 0$. The evidence $E_R$, specified for $R$, is therefore $(0, \infty)$.

The algorithm assumes a truncated version of $R$’s distribution to generate samples. If $R$ was not a rootnode, samples discarding had to be used. Doing so will yield the same result if the number of samples are high enough. If discarding would be used only 731 600 of 10 000 000 has $R > 0$. 731 000 samples suffice for inference, but it demonstrates that discarding is degenerative for accuracy.

In figure 5.5.b the correlation is plotted again discarding so $R > 0$. The correlation starts about the same place, but the drop is much slower than in figure 5.5.a. This implies that by having less sure information, false positive errors are more quickly to appear than false positives.

Since Sensor now can be smaller then 0, but Rockslide can not, it is natural to assume that negative numbers should be considered as 0. In figure 5.5.c the correlation between Sensor and Consequence is plotted once more. Both the evidence $R > 0$ and left censoring of $S$ in 0. As can
Figure 5.5: Empirical estimated correlation between Sensor and Consequence where (a) Sensor is unchanged, (b) Sensor is larger than 0 and (c) Sensor is larger than 0 and negative samples are censored in 0.
be observed, censoring does not change the plot much from figure 5.5.b. Censoring does not affect this correlation.

In figure 5.6 Sensor is plotted with \( \sigma = 1 \) and censored in 0. As expected, the distribution becomes more spread out than its parent, Rockslide. Otherwise it has about the same shape. The parameter setup in figure 5.6 will be assumed for the network.

Figure 5.7 shows a scatterplot between Sensor and Consequence. It is based on 100 000 samples. The samples are plotted twice: once during and once off Tourist Season. The former is marked in blue, the latter in red. It can be observed that the spread along Consequence is much higher during Tourist Season. There is also a vague visible correlation between Consequence and Sensor.

5.2 Setting the Loss and Threshold Functions

Subsection 4.2.3 defined the shape of the Threshold function, \( \theta \). The goal is to find the \( \theta \) that minimizes Loss. But because of the network's random behavior, a low loss might not say anything about \( \theta \) at all. The following
measure is therefore introduced.

\[ \theta_{\text{opt}} = \arg\min_{\theta} \left( E[L|\theta] \right) \]  

(5.5)

The \( \theta \) that produces the lowest expected loss is favorable.

What needs to be done can be summarized as follows: For each sample in Sensor, given a rule \( \theta \) and a Season, an EWS sample can be generated. By inserting EWS and \( C \) into the loss function, a sample of \( L \) can be produced. By taking the average of all \( L \) samples, an estimation of \( E[L|\theta] \) is acquired. This will be approached in detail in this section.

First of all to generate an expected loss value given a \( \theta \) the loss function has to be defined. Let e.g. \( L \) be linear on both upper and lower side.

\[ L_{\text{lower}} \, \text{=} \, C - \overline{C} \quad L_{\text{upper}} \, \text{=} \, \underline{C} - C \]

Here \( \overline{C} \) and \( \underline{C} \) are respectively suprimum and infimum of acceptable Consequence as defined in table 4.2. \( \overline{C} \) and \( \underline{C} \) are therefore dependent on EWS. Loss is defined as follows.

\[ L(C, EWS) = \begin{cases} 
L_{\text{upper}}(C) & C > \overline{C} \\
L_{\text{lower}}(C) & \underline{C} < C \\
0 & \underline{C} \leq C \leq \overline{C}
\end{cases} \]
The expected loss defining a good threshold in equation (5.5) can be approximated as follows.

\[ E[L|E, \theta] \approx \frac{1}{n} \sum_{i=1}^{n} L((C_i|E), (EWS_i|E, \theta)) \]  

(5.6)

Keep in mind that the evidence \( E \) may if desired be neglected.

Let the evidence \( E \) be set to Non-Tourist Season and threshold \( \theta \) be \( c = \{1, 2, 3, 4\} \). Using equation (5.6) results in 0.64888. If \( c \) is replaced by \( \{2, 4, 8, 16\} \) in \( E \) then the value grows to 0.75652.

Let \( L \) be replaced by the following squared function.

\[ L_{\text{lower}} = (C - \overline{C})^2 \quad L_{\text{upper}} = (C - \overline{C})^2 \]

This will give expected loss 92.47675 for \( c = \{1, 2, 3, 4\} \). Loss is higher with a function that penalize more aggressively. In the same way, using logarithms will yield lower result. Let \( L \) be replaced with the following log and linear function.

\[ L_{\text{lower}} = \log(C - \overline{C} + 1) \quad L_{\text{upper}} = C - \overline{C} \]

1 is added to the argument so that the function starts at 0 at the boundaries and cause error by trying to calculate \( \log(0) \). Expected loss for \( \{1, 2, 3, 4\} \) is 0.04823.

It is important to know that the choice of Loss function will not affect the rest of the network. According to the rule of D-separation in subsection 2.2.3, Loss will not affect anything in the network as long as no evidence is inserted into the node. The point of Loss is to have a tool to calibrate the Early Warning System. Changing \( \theta \), a decision node, will affect other nodes.

### 5.2.1 Searching for Optimum

Finding the minimum expected loss in equation (5.5) is computationally demanding. With \( 10^7 \) samples, generating \( EWS \) and \( L \) requires at least \( 2 \times 10^7 \) operations. With an up to date personal desktop computer, this operation takes over a minute. The definition space of \( c \) is \((\mathbb{R}^+)^4 \). Finding global minima by sampling the whole probability space is a possibility, but it is extremely time consuming. A multi dimensional global function minimizer is a better choice. For this task, it is purposed to use the Particle Swarm Optimizer.\(^1\)

Table 5.1 shows the result of a minimization under different loss functions. Season is inserted as evidence to illustrate the difference in the result pending on Season.

\(^1\)An implementation of this optimizer can be seen in appendix A.1
<table>
<thead>
<tr>
<th>$L_{lower}$</th>
<th>$L_{upper}$</th>
<th>Season</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$E[L,θ]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2$</td>
<td>$x^2$</td>
<td>T</td>
<td>0.00000</td>
<td>2.08791</td>
<td>3.41177</td>
<td>3.41177</td>
<td>44.37001</td>
</tr>
<tr>
<td>$x^2$</td>
<td>$x^2$</td>
<td>N</td>
<td>1.16920</td>
<td>5.07736</td>
<td>9.90519</td>
<td>12.5444</td>
<td>5.61565</td>
</tr>
<tr>
<td>$x$</td>
<td>$x^2$</td>
<td>T</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>7.95489</td>
</tr>
<tr>
<td>$x$</td>
<td>$x^2$</td>
<td>N</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>3.56483</td>
</tr>
<tr>
<td>$x^2$</td>
<td>$x$</td>
<td>T</td>
<td>3.47415</td>
<td>18.47447</td>
<td>21.36783</td>
<td>21.36783</td>
<td>0.73367</td>
</tr>
<tr>
<td>$x^2$</td>
<td>$x$</td>
<td>N</td>
<td>6.61968</td>
<td>31.46404</td>
<td>61.44258</td>
<td>79.92014</td>
<td>0.18790</td>
</tr>
</tbody>
</table>

Table 5.1: Proposed minimas for variating loss functions and Season.

Changing the Season will in general give a shift in result. The proportions between the numbers stay about the same. This effect can be explained by looking back at figure 5.3. Consequence goes up quicker towards 1 during Tourist Season than outside. When a Tsunami occurs during Tourist Season, more people die since more people are present. Since the body count goes up quicker, the system has to be more sensitive. This is reflected in the numbers. Expected loss is also much larger during Tourist Season because of this.

Observing squared loss function during Tourist Season, $θ_3 = θ_4$. Having $θ_i = θ_{i+1}$ for one $i = 1, 2, 3$, implies that a state of emergency is not present. Given that EWS moves out of yellow state through the upper bound 3.41177, it changes state directly to red without going through orange. Orange is therefore not present for this choice of thresholds.

Using another loss function will result in other answers. Let $L_{lower} = x$ and $L_{upper} = x^2$. Here all thresholds converge to 0. If $S > 0$, EWS should always be in red state of emergency. Loss of life is penalized so hard that the system implore to evacuate at any site of trouble.

Doing the opposite, letting $L_{lower} = x^2$ and $L_{upper} = x$ results in higher spread. The maximum value amongst Sensor’s samples is 45.39385. This implies that all thresholds above this will have no effect since they would never register anything. The thresholds $c = \{6.61968, 31.46404, 61.44258, 79.92014\}$ will give the same Loss as $c = \{6.61968, 31.46404, 46, 46\}$ since the upper two threshold are both above Sensor’s maximum sample. The penalizing of underestimation is so hard that some of the thresholds are placed outside the bounds of the data.

Having a too asymmetrical functions for Loss can reduce the purpose of the Early Warning System. For $L_{lower}$ much smaller than $L_{upper}$, the thresholds drop to 0 which implies that EWS must be red most of the time. For $L_{lower}$ much larger than $L_{upper}$, some (or all) thresholds surpasses the maximal sample, making one or more state of emergency unreachable. It is therefore important to have this in mind when calibrating the system.

The values from table 5.1 during Non-tourist Season can be seen in figure 5.8. It shows the difference of an over reactive, under reactive and an acceptable set of thresholds. $c = \{6.61968, 31.46404, 61.44258, 79.92014\}$, the optimal when $l_{lower}$ and $l_{upper}$ are respectively squared and linear. Since
this vector is an overestimation, the system is under reactive. This can be observed in the long gaps between each jump. For $c = \{0,0,0,0\}$, the system does not react to the Sensor information at all. It is, as mention earlier, constant red.

When both $l_{lower}$ and $l_{upper}$ are squared, the threshold spread nicely within the range of samples. The gap $c_2$ and $c_3$ tells that the wait from yellow to orange should be longer than e.g. from green to blue. It tells how much threat should be observed before changing the state of emergency.

### 5.2.2 Inference on Thresholds

Given that Loss is defined and an optimum for $\theta$ is found, statistical inference can be done on the Early Warning System. Let $\theta$ be defined by $c = \{0,5,10,20\}$. With algorithm 2 it is possible to find the dependent distribution for the other nodes. A graphical representation of the network with evidence inserted into EWS can be seen in figure 5.9.a. In figure 5.10 the dependent distribution of Tsunami and Consequence is plotted given different states of emergency. For EWS yellow or more severe, the slopes grows quickly towards one. It is still close to the marginal distributions.
Figure 5.9: The simplified tsunamigenic rockslide model where evidence is inserted into (a) EWS and (b) Consequence.
Figure 5.10: Dependent distributions for respectively Tsunami and Consequence when EWS is at least (a & b) yellow, (c & d) orange or (e & f) red.
in figure 5.2.b and 5.4. For EWS at least orange, the slope is lower, but the effect is not very extreme for Tsunami. For Consequence a plateau is formed between 100 and 200. Since high EWS is designed to only pick up threatening events, most of the distribution is placed high. The plateau indicates that false positives, e.g. high EWS with low Consequence, happens in the area between 0 and 100. There are few false positives in the interval from 100 to 200. For EWS red, the plateau in Consequence becomes even more visible and is stretched between 100 and 300. That the plateau is stable on 0.2, tells us that this particular threshold calibration will issue false alarms about 1 in 5 times.

Red EWS is an unlikely event. This is visible in the figure 5.10.e and 5.10.f. The lines are not as smooth as for the other subfigures since it has few samples to work with. For comparison subfigure a and b has 28 343 samples, c and d has 5 862 while e and f has 419. It illustrates that 10^7 samples might not be enough for the model. If more evidence were to be inserted into the model at this time, the distributions might become so inaccurate that a shape could be misleading.

It is also possible to do the inference the other way around. Assume that Consequence is at least 100, the model can tell how likely it is that the EWS is red. Algorithm 2 is used together with equation (5.3). Figure 5.9.b illustrates the model with evidence inserted into Consequence. EWS's distribution with different types of evidence is as follows.

<table>
<thead>
<tr>
<th>Evidence</th>
<th>Green</th>
<th>Blue</th>
<th>Yellow</th>
<th>Orange</th>
<th>Red</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0.95225</td>
<td>0.00004</td>
<td>0.00054</td>
<td>0.00225</td>
<td>0.04490</td>
</tr>
<tr>
<td>C &gt; 10</td>
<td>0.05007</td>
<td>0.01807</td>
<td>0.15489</td>
<td>0.29060</td>
<td>0.48635</td>
</tr>
<tr>
<td>C &gt; 50</td>
<td>0.06127</td>
<td>0.00871</td>
<td>0.09277</td>
<td>0.24901</td>
<td>0.58821</td>
</tr>
<tr>
<td>C &gt; 100</td>
<td>0.08127</td>
<td>0.00370</td>
<td>0.04459</td>
<td>0.15618</td>
<td>0.71424</td>
</tr>
</tbody>
</table>

As Consequence becomes larger, red becomes as expected more and more dominant. It tells how well the systems would do in case of a real threat. E.g. if a Tsunami occurred and over 100 people died of it, the model says that there is over 0.71 chance that the EWS was red. For 50 people, the same number is 0.58. These numbers tell how well the system would do.

The numbers also tell how much the system will guess wrong. E.g. for C > 50 there are 0.06 chance that the EWS would be green and think that nothing happens.
Chapter 6

Conclusion

The work shows how a Bayesian Network can be used to model a hazard problem. It shows how expert data can be quantified and implemented into the model. And it shows how an optimal Threshold can be found.

To construct a model, a CPT and a network structure has to be assumed. A Bayesian Network will only give correct representation of reality if the information inserted into the model is correct. But given that the information inserted is trusted, the model has many advantages.

The model consist of nodes with a flexible construction. To define a node only conditional distribution is required. The distributions can be either discrete, continuous or a combination of the two.

The network can give a broad set of statistical inferences. Any marginal or evidence dependent distribution can be gathered. Correlations, scatter plot, histogram and most other traditional statistical tools are available with arbitrarily high accuracy.

The work presents a way to build the model from expert data. It present a method to convert expert generated histograms into continuous distributions. A tool for generating residuals is used to ensure that the fit is good. This tool is also used on dependent distributions by summing over each dependent distribution. The work include some discussion on alternative fitting techniques.

An Early Warning System is introduced into the network as a tool for decision making. It is incorporated so that the EWS is correlated to the rest of the nodes. This work discusses how it can be calibrated. It also discusses the fallpits one could encounter while calibrating it.

The work ends with inference between the base network and the EWS. It illustrates the effectives of the EWS and gives information about failure type, when predicted was wrong.

Further work can be done on the model. The complexity of $\theta$ goes up with the number of other parents of $EWS$. The thesis only discusses one parent (even though there are two). Multiple parents need to be addressed.
The model is constructed to estimate the state of nature at any given time. It is possible to explore expanding the model to include time dependency, making it a time series model.

Logic sampling which is used for most inferences becomes inaccurate if the ratio between the unlikeliness of the evidence and the number of samples becomes large. An improvement of the algorithm or the use of better algorithms is needed if the model is to be constructed larger.
Appendix A

Python Scripts

A.1 Particle Swarm Optimizer

Particle Swarm Optimizer function is a non-linear function minimizer.[8] It is robust method, meaning that it is fairly immune from ending up in local minima instead of the intended global one. It includes random variables, so there is no guaranty that one gets the same results with the same parameters.

```python
#!/usr/bin/env python
import sys
from numpy import *
from scipy.stats import distributions as d
import time

def fmin(func, xmin, xmax, particles=50, vmax = None, c1 = 1.5, c2 = 1.5, w = 0.73, maxiter=1000, eps = 0.001, disp = 1, full_output = 0):
    """Global minimization method using particle swarm optimization."

    :Parameters:
    func : the Python function or method to be minimized.
    xmin : lower bounds for search space.
    xmax : upper bounds for search space.

    :Returns: xopt, {fopt, iter, flag, time}
    xopt : ndarray
        minimizer of function
    fopt : number
        function evaluated at minimum: fopt = func(xopt)
    iter : number
        number of iterations
    flag : boolean
        indicator if maximum iterations was reached.
    time : number
        time in seconds used searching.

    :OtherParameters:
    c1 : number
```

59
influence factor for personal best
\(c_2\) : number

influence factor for global best
\(c_1\) : number

acceptable error between local minimas and global minimum
\(\epsilon_{ps}\) : number

the maximum iteration to perform
\(\text{maxiter}\) : number

number of search particles in swarm
\(n\) : number

maximum velocity for each particle.
\(v_{\text{max}}\) : ndarray

influence factor for current course
\(w\) : number

non-zero to print convergence message
\(\text{disp}\) : number

non-zero to return all gathered statistics
\(\text{full_output}\) : number

Example:

```python
>>> import swarm
>>> function = lambda (x,y,z): (x-4)**2 + (y+2)**2 + (z)**2 + 1
>>> lower_bound = (-10, -10, -10)
>>> upper_bound = (10, 10, 10)
>>> result = fmin(function, lower_bound, upper_bound)
Minimization terminated successfully.
Function value: 1.000000
Parameter vector: [ 4.00000676e+00 -1.99999019e+00 -5.60480975e-06]
Number of iterations: 89
Time elapsed: 0.228769
```

```python
# initializing values.
t = time.time()
if isinstance(xmax,int):
xmax, xmin, vmax= [xmax], [xmin], [vmax]
m = len(xmax) # dimension of a particle
xmax = array(xmax)
xmin = array(xmin)
if vmax == None or vmax[0] == None:
vmax = (xmax - xmin)/2.
else:
vmax = array(vmax)
n = particles # number of particles
x = (d.uniform.rvs(size=(n,m)))*(xmax - xmin) + xmin # init position
v = (d.uniform.rvs(size=(n,m)))*2*vmax - vmax # init velocity
g_best = x[argmin(map(func,x))] # init global best
p_best = x.copy() # init personal best
converged = 0

# Algorithm run.
for iter in xrange(maxiter):
    # Update velocity
    v = w*v + (c1*d.uniform.rvs(size=n)*(p_best-x).transpose()).transpose()
    + (c2*d.uniform.rvs(size=n)*(g_best-x).transpose()).transpose()
    v += (v>vmax)*(vmax-v) + (v<-vmax)*(-vmax-v)
    # Update position
    x += v
    x += (x>xmax)*(xmax-x) + (x<xmin)*(xmin-x)
```
A.2 Distribution Fitting

Distribution fitting is a method constructed to fit a continuous distribution to a set of samples. It uses Swarm optimizer to do this job.

```python
#!/usr/bin/env python
from scipy.stats import distributions as d
from numpy import *
from swarm import fmin
def distribution_fitting(probability, upper_limits, distribution, xmin, xmax, disp=1):
    '''Optimize the parameters of a distribution to fit a set of probabilities.''
    :Parameters:
    probability : a list of probabilities that sums to 1
    upper_limits : a list of references for where the continous
        and descrete distribution must coenside
    distribution : a Scipy.stats.distribution compatable CDF-method.
    xmin : list of lower limit for the parameters.
    xmax : list of higher limits for the parameters.
    :Returns: (xopt, sum_of_squares)
    xopt : ndarray
        optimal parameters
    sum_of_squares : number
        the sum of square between the distribution given xopt

    f = lambda (x,y,z): (x-4)**2 + (y+2)**2 + (z)**2
    fmin(f,(-10,-10,-10),(10,10,10))
```
A.3 Parameter Retrieval

This method uses the distribution_fitting method to gather a set of parameters. It then saves them to disk using the Pickle-class.

```python
#!/usr/bin/env python
from scipy.stats import distributions as d
from numpy import *
import cPickle

def parameter_retrival(probabilities, upper_limits, distribution, xmin, xmax, name, disp=0):
    dim = len(xmin) # Number of parameters used in distribution.
    if dim==2:
        func = lambda (a,b): distribution.cdf(upper_limits,a,b)
    elif dim==3:
        func = lambda (a,b,c): distribution.cdf(upper_limits,a,b,c)
    elif dim==4:
        func = lambda (a,b,c,d): distribution.cdf(upper_limits,a,b,c,d)
    else:
        func = lambda (a,b,c,d,e): distribution.cdf(upper_limits,a,b,c,d,e)
    cum_prob = cumsum(probability)
    squared_sum = lambda param: sum((cum_prob - func(param))**2)
    optimal_params = fmin(squared_sum, xmin, xmax, particles=10*dim, disp=disp)
    return (optimal_params, squared_sum(optimal_params))
```

```python
if __name__=='__main__':
    prob = array([.0,.3,.4,.2,.1])
    bounds = array([0.,5.,15.,25.,1000.])
    xmin = (-10,0) # (loc, scale=0)
    xmax = (20,20)
    opt_params = distribution_fitting(prob, bounds, d.expon, xmin, xmax)
    print opt_params
```
Optimize the parameters of a distribution to fit multiple sets of probabilities and save to disk.

Parameters:
- **probabilities**: a 2 dimensional list of probabilities where sums along 2nd axis is 1
- **upper_limits**: a list of references for where the continuous and discrete distribution must coalesce
- **distribution**: a Scipy.stats.distribution compatible CDF-method.
- **xmin**: list of lower limit for the parameters.
- **xmax**: list of higher limits for the parameters.
- **name**: name of file for saving.

Returns: **sum_of_squares**

The sum of squared errors over all fits.

**Example:**
```python
>>> prob = zeros((9,9))
>>> prob[0] = 1; prob[1:5,1] = [1,0.4,0.1,0.1]; prob[2:7,2] = [0.45,0.65,0.3,0.2,0.1]
>>> prob[2:8,3] = [0.1,0.2,0.15,0.15]; prob[6:6] = [0.15,0.15,0.62]
>>> prob[6:7] = [0.15,0.2,0.15]; prob[6:8] = [0.15,0.2,0.15,0.2]
>>> print prob
array([[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]])
>>> from scipy.stats import distributions
>>> dist = distributions.gamma
>>> xmin = (0,-50,0) # (alpha>0,loc,scale>0)
>>> xmax = (50,50,50)
>>> name = 'tsunami'
>>> sum_of_squares = parameter_retrival(prob, upper_limits, dist, xmin, xmax, name)
>>> print sum_of_squares
0.0172543626772
>>> f = open('tsunami.data','r')
>>> import cPickle
>>> data = cPickle.load(f); f.close()
>>> alpha, loc, scale = data[0]
>>> print scale
[0.18828219284800851, 0.0053434081161220105, 1.063880597081426, 15.948216291574068,
0.22694252008901022, 0.83362928487652765, 17.9828956738104, 35.661278158964798,
40.174334094602685]
>>> print data[1] == upper_limits
[True, True, True, True, True, True, True, True, True]
```

```python
func = lambda prob: distribution_fitting(prob, upper_limits,
distribution, xmin, xmax, disp)```
opt_params_raw = map(func, probabilities)
if isinstance(probabilities[0], (float, int)): # Root node
    opt_params = list(opt_params_raw[0])
    sum_of_squares = opt_params_raw[1]
else: # General node
    dim = len(xmin)
    opt_params = [[] for i in range(dim)]
    sum_of_squares = 0
    for elem in opt_params_raw:
        for i in range(dim):
            opt_params[i].append(elem[0][i])
        sum_of_squares += elem[1]
data = [opt_params, list(upper_limits), distribution.name]
f = open('%s.data' % name, 'w')
cPickle.dump(data, f); f.close()
return sum_of_squares

if __name__ == '__main__':
    prob = zeros((9,9))
    prob[0,0] = 1; prob[1:5,1] = [1,0.4,0.1,0.1]; prob[2:7,2] = [0.45,0.65,0.3,0.2,0.1]
    prob[2:8,3] = [0.15,0.25,0.4,0.3,0.15,0.1]; prob[4:4] = [0.2,0.4,0.15,0.15,0.05];
    prob[5:5] = [0.1,0.2,0.15,0.15]; prob[6:6] = [0.15,0.15,0.02]
    prob[6:7] = [0.15,0.2,0.02]; prob[6:8] = [0.1,0.25,0.4]
    limits = array([0, 0.5,2,4,7,12,20,35,1900])
    xmin = (0,-50,0) # (alpha>0,loc,scale>0)
    xmax = (50,50,50)
    name = 'tsunami'
    sum_of_squares = parameter_retrival(prob, limits, d.gamma, xmin, xmax, name)
    print sum_of_squares

A.4 Node Sampler

This method gathers parameters and samples from disk and uses it to generate samples for a node. It requires that parameter_retrival has been run on it and its parents and node_sampler has been run on the parents. It is constructed to handle many samples at once.

def node_sampler(name, size = 1e7, parents=[], censor=0):
    '''Gathers data from file generated with parameter_retrival and makes samples using Monte Carlo simulation technique.

    :Parameters:
    name : name of node
    :Returns: None
    :Other Parameters:
    size : number
        the number of samples generated by the simulation
    parents : list
        includes the names of the parents to the node
    censor : number
        non-zero to left censor all samples in origo.
    :Example:
    >>> import os
    >>> os

64
>>> os.system('ls *.data *.samples')
rocksld.data tsunami.data

>>> node_sampler('rocksld',censor=1)

>>> os.system('ls *.data *.samples')
rocksld.data rocksld.samples tsunami.data

>>> node_sampler('tsunami',parents=["rocksld"],censor=1)

>>> os.system('ls *.data *.samples')
rocksld.data rocksld.samples tsunami.data tsunami.samples

# loading node
f = open('%s/%s.data' % (path,name),'r')
params, node_limits, dist_name = cPickle.load(f); f.close()
distribution = eval('d.%s' % dist_name)

# loading parents
category = 0; iter = 1
for parent in parents:
f = open('%s/%s.data' % (path,parent), 'r')
limits = array(cPickle.load(f)[1]); f.close()
f = open('%s/%s.samples' % (path,parent), 'r')
samples = cPickle.load(f); f.close()
cat = sum(map(lambda s: limits>s, samples),axis=0) * iter
iter *= len(parent_limits)
if not category:
category = cat
else:
category += cat

# defining function
if not parents:
dim = len(params)
else:
dim = len(params[0])
if dim==2:
func = lambda x, (a,b): distribution.ppf(x,a,b)
elif dim==3:
func = lambda x, (a,b,c): distribution.ppf(x,a,b,c)
elif dim==4:
func = lambda x, (a,b,c,d): distribution.ppf(x,a,b,c,d)
else:
func = lambda x, (a,b,c,d,e): distribution.ppf(x,a,b,c,d,e)

uni_samples = d.random(size) # random samples
if not parents:
samples = map(lambda x: func(x, params), uni_samples)
else:
params_structured = []
for par in params:
    params_structured.append(map(lambda i: par[i], category))
params_structured.insert(0,uni_samples)
params_compiled = cPickle.dumps((params,i)).tobytes()
samples = map(lambda par: func(par[0],par[1:]), params_compiled)
if censor:
samples = (samples>0)*samples
f = open('%s/%s.samples' % (path,name), 'w')
cPickle.dump(samples,f)
A.5 Threshold Tester

A class to find the expected loss given thresholds. Using automatic optimizers to search for optimal thresholds requires a lot of computer resources.

```python
#!/usr/bin/env python
from numpy import *
from swarm import fmin
import cPickle

class Threshold_tester:
    '''Loads up relevant nodes to find expected loss given thresholds.

    :Parameters:
    name : name of file to save convergence
    lower_function : loss function for under estimation of Consequence
    upper_function : loss function for over estimation of Consequence

    :Returns: None

    :OtherParameters:
    tourist_season : number
        non-zero if consequence should be evaluated during Tourist Season

    :Example:
    >>> lower_loss = lambda a:a**2
    >>> upper_loss = lambda a:a
    >>> s = Threshold_tester('linear_squared', lower_loss, upper_loss)
    >>> lower_bounds = array([0,0,0,0])
    >>> upper_bounds = array([30,30,30,30])
    >>> s.evaluate(lower)
    >>> fmin(s.evaluate, lower, upper)
    >>> f = open('linear_squared.converge','r')
    >>> print f.read(); f.close()
    c = ( 0.000000, 0.000000, 0.000000, 0.000000) -> 1:194.851671266
    c = (12.704600, 30.065687, 58.176529, 65.951621) -> 2:0.757564748552
    c = ( 3.416253, 21.312988, 62.006844) -> 3:0.737373115572
    c = ( 3.409059, 21.928904, 62.829704) -> 1149:0.737371345621
    c = ( 3.583867, 21.985556, 65.478756) -> 2151:0.737343083698
    c = ( 3.844727, 21.111133, 21.584128) -> 3126:0.734589396037
    c = ( 3.579291, 20.251371, 20.460938) -> 18622:0.73412475796
    c = ( 3.638283, 19.052719, 21.555686) -> 11868:0.734178549551
    c = ( 3.654758, 18.935609, 21.412335) -> 12383:0.734106237812
    c = ( 3.654758, 19.404781, 21.369726) -> 12386:0.733968481466
    c = ( 3.590387, 19.229311, 21.293337) -> 14893:0.73394306422
    c = ( 3.676534, 19.272637, 21.378675) -> 15143:0.7338521527
    c = ( 3.493274, 19.078639, 21.329666) -> 15400:0.733874044768
    c = ( 3.582409, 19.176769, 21.317387) -> 16133:0.7338090465
    c = ( 3.495067, 18.778793, 21.388517) -> 16369:0.7338262776
    c = ( 3.554927, 18.541242, 21.293779) -> 17139:0.733818632881
    c = ( 3.496284, 18.772862, 21.294762) -> 17852:0.733762421521
    c = ( 3.478544, 18.762334, 21.368584) -> 19102:0.733709053947
    c = ( 3.474115, 18.681099, 21.374594) -> 24112:0.733687956647
    c = ( 3.474115, 18.474473, 21.367833) -> 24666:0.733675956647

    def __init__(self, name, lower_function, upper_function, tourist_season=1):
        # Getting relevant samples
    '''
```
```
A.5 Threshold Tester

```python
f = open('sensor.samples','r')
self.sensor = cPickle.load(f); f.close()
if tourist_season:
    f = open('consequence_tourist.samples','r')
    self.consequence = cPickle.load(f); f.close()
else:
    f = open('consequence_nontourist.samples','r')
    self.consequence = cPickle.load(f); f.close()

# setting other initial values
self.iterations = 0
self.name = name
self.rules = [(-1.0), (5,10), (40,50), (70,1000)]
self.size = len(self.consequence)
self.best_value = 10**7
self.best_vector = []
self.lower = lower_function
self.upper = upper_function
f = open('%s.converge' % self.name, 'w'); f.close()

def loss_function(self, ews_state):
    '''Evaluates the loss for all samples within a state of emergency.

    :Parameters:
    ews_state : number indicating the state of emergency

    :Returns: expected_loss
    expected_loss : number
        the avarage loss over all samples.'''
    rule = self.rules[i]
    state_of_emerency = (self.ews == ews_state)
    lower_set = (self.consequence < rule[0])*(rule[0]-self.consequence)
    lower_loss = self.lower(lower_set*state_of_emergency)
    upper_set = (self.consequence > rule[1])*(self.consequence-rule[1])
    upper_loss = self.upper(upper_set*state_of_emergency)
    return sum(lower_loss+upper_loss)/float(self.size)

def evaluate(self, increments):
    '''Finds the expected loss of a set of increments and save the loss
to file if the value is lower the previous values.

    :Parameters:
    increments : list of increments between each threshold

    :Returns: expected_loss
    expected_loss : number
        Expected loss given the set of increments'''
    self.iteration += 1
    threshold = cumsum(increments)
    self.ews = array(map(lambda a:a,sum(map(lambda t: (t<self.sensor), threshold),axis=0)))
    average_loss = sum(map(self.loss_function, range(5)))
    if(result < self.best_value):
        f = open('%s.converge' % self.name,'a')
        f.write('c = (%9.6f, %9.6f, %9.6f) -> ' % tuple(threshold))
        f.write('%6d:%s
' % (self.iteration, average_loss)); f.close()
        self.best_value = result
        self.best_vector = increments
```

67
return result
if __name__=='__main__':
    lower_loss = lambda a:a
    upper_loss = lambda a:a**2
    s = Threshold_tester('linear_squared', lower_loss, upper_loss)
    lower_bounds = array([0,0,0,0])
    upper_bounds = array([30,30,30,30])
    s.evaluate(lower)
    fmin(s.evaluate, lower, upper)
Appendix B

Acknowledgement

Huseby, Arne Bang
Medina-Cetina, Zenon
Nadim, Farrokh
Natvig, Bent
Norwegian Geotechnical Institute/International Center for Geohazard
Simula Research Laboratory
Solhjell, Ida
Bibliography


