7 Bayes Nets: Multivariate causal modelling

7.1 Causal models

In the previously discussed regression example we mainly considered a model for one random variable or at most two. We now consider more complex models with many more factors described by random variables. Probability theory nicely generalizes to multiple random variables, and such multivariate cases are described by a joint probability as outlined in the review of probability theory. An example from Sebastian Thrun of a model to diagnose when a car is not starting is shown in Figure 7.1.

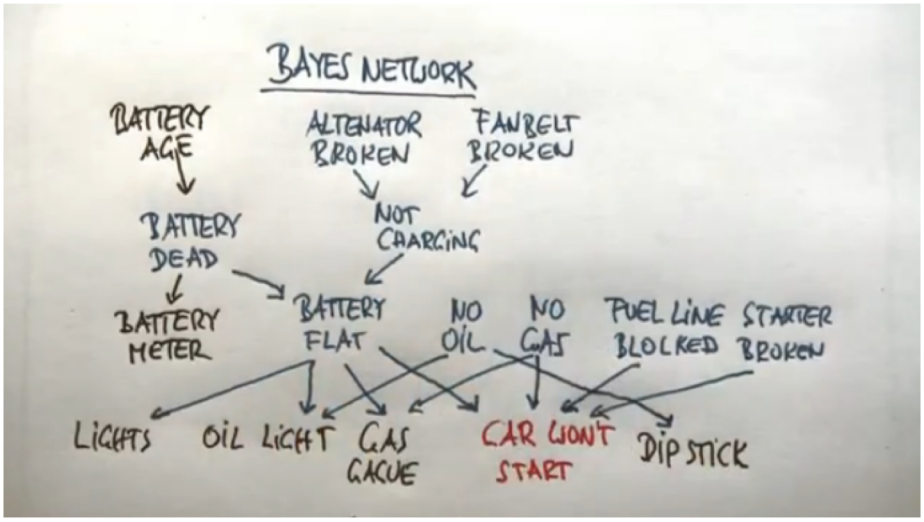

Fig. 7.1 Example of causal model.

This example is reasonably sized although real world problems would be likely be even larger than this. This model considered 16 random numbers to determine possible causes if the car does not start (the variable "car does not start is not a random number as we are using this model when we know already that the car is not starting). The random numbers themselves could have two possible outcomes (like if there is gas in the tank) or even multiple possible values (like the age of the battery). At this time let us simplify the model with only considering binary values. That is, the age of the battery would only be specified as new or old. The joined probability table for the 16 variables would then have $2^{16} - 1 = 65535$ entries. These parameters have to be

estimated (learned) from examples using MAP or MLE.

In addition to the shear explosion of parameters with increasing model complexity, there is another reason why the joined probability function is also not exactly what we need to know. The joint density functions of multiple variables describe the co-occurrence of specific values of the random variables. Indeed, the joint probability function $p(X, Y)$ is symmetric in its arguments,

$$p(X, Y) = p(Y, X), \tag{7.1}$$

What we really want to do is to a model to reason about the world, or specifically, to reason about possible events. For this we want to add knowledge or hypotheses about **causal relations**. For example, a fire alarm should be triggered by a fire, although there is some small chance that the alarm will not sound when the unit is defect. However, it is (hopefully) unlikely that the sound of a fire alarm will trigger a fire. It is useful to illustrate such casual relations with graphs such as

In such **graphical models**, the nodes represent random variables, and the links between them represent causal relations with conditional probabilities, $p(A|F)$. Since we use arrows on the links we are discussing here **directed graphs**, and we are also restricting our discussions here to graphs that have no loops, so called **acyclic graphs**. **Directed acyclic graphs** are also called **DAG**s.

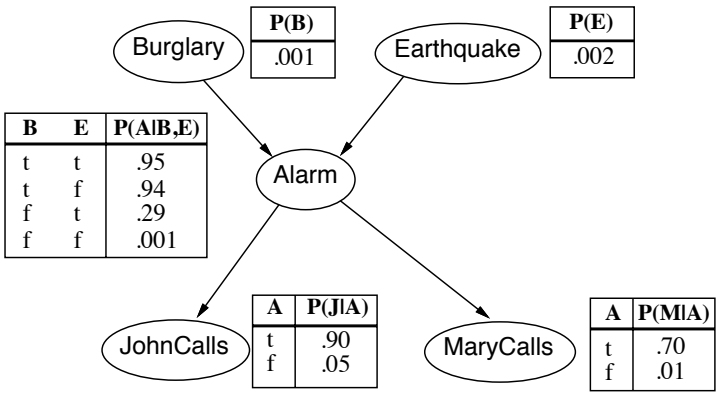

Fig. 7.2 Example of causal model.

Graphical causal models have been advanced largely by Judea Pearl, and the following example is taken from his book[2]. The model is shown in Figure 7.2. Each of the five nodes stands for a random binary variable (Burglary B={yes,no}, Earthquake E={yes,no}, Alarm A={yes,no}, JohnCalls J={yes,no}, MaryCalls M={yes,no}) The figure also include **conditional probability tables (CPTs)** that specify the conditional probabilities represented by the links between the nodes.

[2] Judea Pearl, 'Causality: Models, Reasoning and Inference', Cambridge University Press 2000, 2009'.

The joined distribution of the five variables can be factories in various ways following the chain rule mentioned before (equations 4.37), for example as

$$p(B, E, A, J, M) = P(B|E, A, J, M)P(E|A, J, M)P(A|J, M)P(J|M)P(M) \tag{7.2}$$

However, the the causal model represents a specific factorization of the joint probability functions, namely

$$p(B, E, A, J, M) = P(B)P(E)P(A|B, E)P(J|A)P(M|A), \tag{7.3}$$

which is much easier to handle. For example, if we do not know the conditional probability functions, we need to run many more experiments to estimate the various conditions ($2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 31$) instead of the reduced conditions in the causal model ($1 + 1 + 2^2 + 2 + 2 = 10$). It is also easy to use the casual model to do inference (drawing conclusions), for specific questions. For example, say we want to know the probability that there was no earthquake or burglary when the alarm rings and both John and Mary call. This is given by

$$\begin{aligned}
&P(B = f, E = f, A = t, J = t, M = t) = \\
&= P(B = f)P(E = f,)P(A = t|B = f, E = f)P(J = t|A = t)P(M = t|A = t) \\
&= 0.998 * 0.999 * 0.001 * 0.7 * 0.9 \\
&\approx 0.00062
\end{aligned}$$

Although we have a casual model where parents variables influence the outcome of child variables, we can also use a child evidence to infer some possible values of parent variables. For example, let us calculate the probability that the alarm rings given that John calls, $P(A = t|J = t)$. For this we should first calculate the probability that the alarm rings as we need this later. This is given by

$$\begin{aligned}
P(A = t) = &P(A = t|B = t, E = t)P(B = t)P(E = t) + ... \\
&P(A = t|B = t, E = f)P(B = t)P(E = f) + ... \\
&P(A = t|B = f, E = t)P(B = f)P(E = t) + ... \\
&P(A = t|B = f, E = f)P(B = f)P(E = f) \\
= &0.95 * 0.001 * 0.002 + 0.94 * 0.001 * 0.998 + ... \\
&0.29 * 0.999 * 0.002 + 0.001 * 0.999 * 0.998 \\
= &0.002516442
\end{aligned}$$

We can then use Bayes' rule to calculate the required probability,

$$\begin{aligned}
P(A = t|J = t) &= \frac{P(J = t|A = t)P(A = t)}{P(J = t|A = t)P(A = t) + P(J = t|A = f)P(A = f)} \\
&\approx \frac{0.9 * 0.0025}{0.9 * 0.0025 + 0.05 * 0.9975} \\
&\approx 0.0434
\end{aligned}$$

We can similarly apply the rules of probability theory to calculate other quantities, but these calculations can get cumbersome with larger graphs. It is therefore useful to use

numerical tools to perform such inference. A Python toolbox for Bayesian networks is introduced in the next section.

While inference is an important application of causal models, inferring causality from data is another area where causal models revolutionize scientific investigations. Many traditional methods evaluate co-occurrences of events to determine dependencies, such as a correlation analysis. However, such a correlation analysis is usually not a good indication of causality. Consider the example above. When the alarm rings it is likely that John and Mary call, but the event that John calls is mutually independent of the event that Mary calls. Yet, when John calls it is also statistically more likely to observe the event that Mary calls. Sometimes we might just be interested in knowing about the likelihood of co-occurrence, for which a correlation analysis can be a good start, but if we are interested in describing the causes of the observations, then we need another approach. Some algorithms have been proposed for **structural learning**, such as an algorithm called **inferred causation (IC)**, which deduces the most likely causal structure behind given data is.

7.2 Discrete probabilistic modeling in Python using LEA

An Python toolbox for working with discrete probabilities is give by LEA. The package and tutorials are provided at https://bitbucket.org/piedenis/lea. We will only briefly demonstrate it on the burglary/earthquake example above.

The first step is to create the conditional probability tables, which can be done like

```
burglary    = Lea.boolProb(1,1000)
earthquake = Lea.boolProb(2,1000)

alarm = Lea.buildCPT(
        (  burglary &  earthquake , Lea.boolProb(950,1000)),
        (  burglary & ~earthquake , Lea.boolProb(940,1000)),
        ( ~burglary &  earthquake , Lea.boolProb(290,1000)),
        ( ~burglary & ~earthquake , Lea.boolProb(  1,1000)))

johnCalls = Lea.buildCPT(
            (  alarm , Lea.boolProb(90,100)),
            ( ~alarm , Lea.boolProb( 5,100)))
maryCalls = Lea.buildCPT(
            (  alarm , Lea.boolProb(70,100)),
            ( ~alarm , Lea.boolProb( 1,100)))
```

In this notation we already define what the probabilities depend, and this hence also defines the graph structure for the DAG.

We are now ready to calculate some inference. As an example of an inference we recalculate the example above:

```
print(Pf(~burglary & ~earthquake & alarm & johnCalls & maryCalls))
print(Pf(alarm))
print(alarm.given(johnCalls))
```

7.3 Naive Bayes

This is a simple example with the following graph:

In the previous example we used two dimensional feature vectors to illustrate the classification problems with two dimensional plots. However, most machine learning applications work with high dimensional feature vectors. We will now discuss an important method with generative models which is called **Naive Bayes** that is often used with high-dimensional data. We will discuss this method with an example of text processing, specifically the example by Andrew Ng of making a spam filter that classifies email messages as either spam ($y = 1$) or non-spam ($y = 0$) emails. To do this we need first a method to represent the problem in a suitable way. We chose here to represent a text (email in this situation) as **vocabulary** vector. A vocabulary is simply the list of all possible words that we consider, and the text is represented by this vector with entries 1 if the word can be found in the list or an entry 0 if not, e.g.

$$\mathbf{x} = \begin{pmatrix} 1 & \text{a} \\ 0 & \text{aardvark} \\ 0 & \text{aardwolf} \\ . & . \\ . & . \\ . & . \\ 1 & \text{buy} \\ . & . \\ . & . \\ 0 & \text{zygmurgy} \end{pmatrix} \tag{7.4}$$

We are here only considering values 0 and 1 instead of, for example, counting how often the corresponding word appears. The later is usually called a 'bag of words'. The difference is that each entry is a binomial random variable and would be a multinomial in the other example, though the methods generalize directly to the other case. Note that this feature vector is typically very high dimensional. Let us consider here that our vocabulary has 50.000 word, which is a typical size of common languages.

We now want to build a discriminative model from some training examples. That is, we want to model

$$p(\mathbf{x}|y) = p(x_1, x_2, ..., x_{50000}|y). \tag{7.5}$$

This is a very high dimensional density function which has $2^{50.000} - 1$ parameters (the -1 comes from the normalization condition). We can factorize this conditional density function with the chain rule

$$p(x_1, x_2, ..., x_{50000}|y) = p(x_1|y)p(x_2|y, x_1)...p(x_{50000}|y, x_1, ..., x_{49999}). \tag{7.6}$$

While the right hand side has only 50.000 factors, there are still $2^{50.000} - 1$ parameters we have to learn. We now make a strong assumption namely that all the words are conditionally independent in each text, that is,

$$p(x_1|y)p(x_2|y, x_1)...p(x_{50000}|y, x_1, ..., x_{49999}) = p(x_1|y)p(x_2|y)...p(x_{50000}|y). \tag{7.7}$$

This is called the **Naive Bayes (NB) assumption**. Hence, we can write the conditional probability as a factor of terms with 50.000 parameters

$$p(\mathbf{x}|y) = \prod_{j=1}^{50000} p(x_j|y). \tag{7.8}$$

To estimate these parameters we can apply maximum likelihood estimation, which gives

$$\phi_{j,y=1} = \frac{1}{|\{y=1\}|} \sum_{i \in \{y=1\}} x_j^{(i)} \tag{7.9}$$

$$\phi_{j,y=0} = \frac{1}{|\{y=0\}|} \sum_{i \in \{y=0\}} x_j^{(i)} \tag{7.10}$$

$$\phi_y = \frac{|\{y=1\}|}{m}. \tag{7.11}$$

The first equation is the probability that the word j appears in a spam text, the second equation is that the word j appears in a non-spam text, and the third equation specifies the frequency of spam examples in the data set.

With these parameters we can now calculate the probability that email $\mathbf{x}$ is spam as

$$p(y=1|\mathbf{x}) = \frac{\prod_{j=1}^{50.000} \phi_{j,y=1}\phi_{y=1}}{\prod_{j=1}^{50.000} \phi_{j,y=1}\phi_{y=1} + \prod_{j=1}^{50.000} \phi_{j,y=0}\phi_{y=0}}. \tag{7.12}$$

In practice this often works to some extent, at least when the Naive Bayes assumption is appropriate. Of course, words in a text should be highly correlated, but the gist here is that the pure frequency of words has some correlates with the type of text.

Finally, a note that there is a slight problem if some of the words, say x_{100}, are not part of the training set. In this case we get an estimate that the probability of this word every occurring is zero, $\phi_{100,y=1} = 0$ and $\phi_{100,y=0} = 0$, and hence $p(y=1|x) = \frac{0}{0}$. A common trick, called **Laplace smoothing** is to add one occurrence of this word in every case, which will insert a small probability proportional to the number of training examples to the estimates,

$$\phi_{j,y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2} \tag{7.13}$$

$$\phi_{j,y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2}. \tag{7.14}$$

It is interesting to compare the Naive Bayes classification with other classification methods.