4 Regression and maximum likelihood

We have so far given an overview of **supervised learning** where we propose a model in form of a parameterized function and where we then learned the model parameters from example data. We have discussed this with a simple linear model in the introduction section. We will now revise this method to include stochastic data. This will show how modern probabilistic machine learning can formulated. We will follow a simple stochastic generalization of the linear regression example to introduce the formalism. Later chapters will be generalizing these ideas to non-linear problems in higher dimensions.

4.1 Trends in stochastic data

In supervised learning, examples of input-output relations are given and our goal is to make a model that can make **predictions** of previously unseen data. Let's consider here an example from robotics where we want to model how a far a terrestrial robot is moving when the wheels are turning for a given number of seconds after activating the corresponding motors with a certain power.

More formally, we denote again the training data as pairs $(\mathbf{x}, y)$, where the feature inputs $\mathbf{x}$ are the time in seconds we let the motor run, and the outputs or labels y is the distance that the robot traveled. This distance true distance traveled has to be provided by the teacher, likely in form of measurements such as from using a ruler or to use another distance sensor such as a laser range finder or and ultrasonic sensor. In the following we consider m training examples, the pairs of values

$$\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1...m\}. \tag{4.1}$$

The above notation describes mapping examples from an n-dimensional feature space to a 1-dimensional output space, hence the vector notation for $\mathbf{x}$ and the scalar notation for y. We consider here a 1-dimensional output space for our discussions mainly for convenience. Generalization to higher-dimensional are straight forward. Note that an index with brackets, (i), is used to label different training examples.

Figure 4.1 shows data from a Lego Mindstorm robot that has two motorized wheels and and a ultrasonic distance motor attached to it. Let us consider how such a terrestrial robot moves when both motors are driven for a certain amount of time. To automate the collection of data we use an ultrasonic sensor to measure the distance to a wall while driving the robot for different amount of time forward and backward. In Figure 4.1B we show several measurements of the distance traveled for different times the motors are activated.

The data clearly reveal some systematic relation between the time of running the motor and the distance traveled, the general trend being that the traveled distance

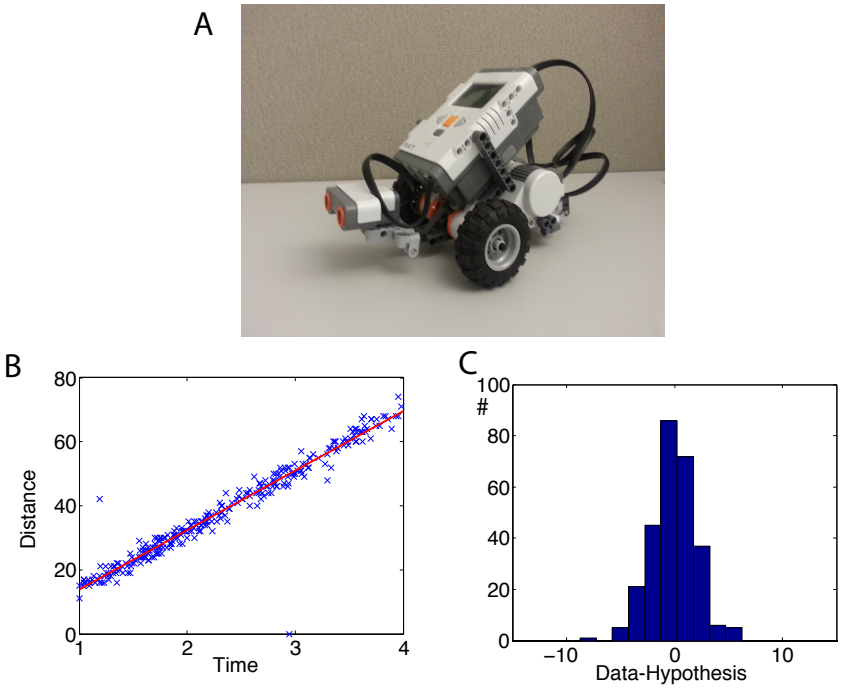

Fig. 4.1 (A) A terrestrial robot build with Lego Mindstorm with an ultrasonic sensor at the front. (B) Measurements of distance travelled by a robot when running the motor for different number of milliseconds with a given power. (C) Corresponding histogram of differences between data and a line that is fitted by minimizing the the mean square error between the data points and the line.

increases with increasing running time of the motors. While there seems to be some noise in the data, the outliers and the noise can not hide a linear trend for most of the data. This hypothesis can be quantified as a parameterized function,

$$\hat{y}(x; \mathbf{w}) = w_0 + w_1 x. \tag{4.2}$$

This notation means that the hypothesis $\hat{y}$ is a function of the quantity x, and the hypothesis includes all possible straight lines, where each line can have a different offset w_0 (intercept with the y-axis), and slope w_1.

We typically collect parameters in a **parameter vector** $\mathbf{w}$. We only considered one input variable x above, but we can easily generalize this to higher dimensional problems where more input **attributes** are given. For example, we could not only vary the time the motor is running, let us label this attribute now with x_1, but also a certain time we push the robot forward by hand, labeled with x_2. The two methods of moving the robot forward are independent of each other and we can hence independently add the effects of higher dimensions to our hypothesis. To compress our notations further we also introduce here the convention that we consider a constant input, $x_0 = 1$ as the first component of the input vector, so that the corresponding parameter encodes the offset of the function. The state vector can then be written as,

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}. \tag{4.3}$$

With this convention we can write the hypothesis as

$$\hat{y}(\mathbf{x}; \mathbf{w}) = w_0 x_0 + w_1 x_1 + w_2 x_2. \tag{4.4}$$

It is then even easy to write a linear hypothesis with n attributes as

$$\hat{y}(\mathbf{x}; \mathbf{w}) = w_0 x_0 + ... + w_n x_n = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x}, \tag{4.5}$$

where the superscript T indicates the transpose of a vector.

Another factor that influences the distance traveled is the power setting of the motor. Of course, the distance traveled within a certain time does depend on the power and it is not just an independent additive effect on the travelled distance. Results of the experiment for different power settings and different travel times are show in Figure **??**. Figure **??**A also includes a fit to equation 4.4. However, these data are better described by a bilinear hypothesis,

$$\hat{y}(\mathbf{x}; \mathbf{w}) = w_0 x_0 + w_1 x_1 x_2. \tag{4.6}$$

The corresponding fit of the same data is shown in Figure **??**B.How to perform these fits is discussed further below.

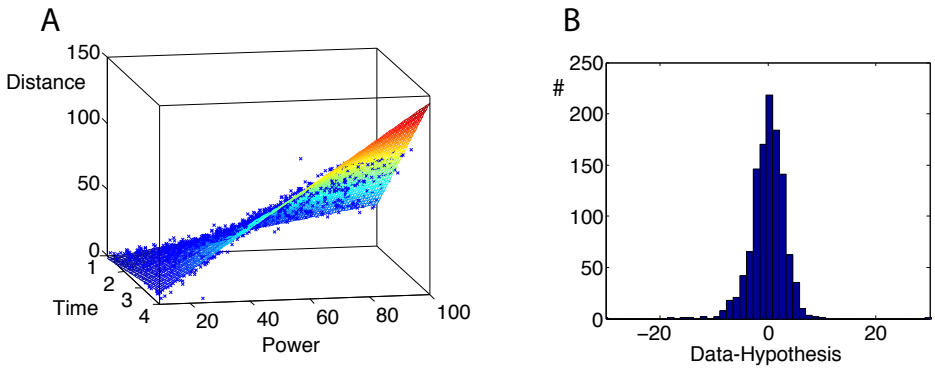

Fig. 4.2 (A) Measurements of distance travelled by the robot when running the motor for different number of milliseconds and various power settings. Fit according to equation 4.4 (B) Histogram of differences between data and hypothesis.

While we had to make a good guess for the functional form of the trend in the data, the actual parameters have so far not been specified. Thus, we made a **hypothesis** in the form of a parameterized function, $h(\mathbf{x}; \mathbf{w})$, and the learning part boils down to determining appropriate values for the parameters from the sample data. After learning these parameters we can then use this function to predict specific reactions of a plant even for motor commands for which no training examples were given. The remaining

question is how we find appropriate values for the parameters. However, before we do this we need to be more faithful to the data and acknowledge fluctuation around our initial hypothesis.

4.2 Probabilistic models and maximum likelihood

So far, we have only modelled the trend of the data, and we should investigate more the fluctuations around this trend. Of course, we expect several sources of noise such as the accuracy of the ultrasonic sensor and the tendency of the robot to sometimes turn due to wheel slippage. We also started new trials when the robot went too much off track. Thus, we already try to minimize error due to a careful setup of the experiment to gather the data. However, these data are still noisy due other sources of uncertainty and instead of trying to avoid uncertainty we are now embracing this and start to model it.

Figures 4.1C and 4.2B are plots of the histogram of the differences between the actual data and the hypothesis regression line. The histograms look a bit Gaussian, which according to the central limit theorem is a likely finding for additive and independent noise sources. In any case, we should revise our hypothesis by acknowledging the stochastic nature of the data and writing a down a specific functional form of a conditional density function for the quantity y given some input values $\mathbf{x}$. Similar to before, we also allow this probabilistic hypothesis to depend on some parameters $\boldsymbol{\theta}$,

$$p(\hat{y}|\mathbf{x};\boldsymbol{\theta}). \tag{4.7}$$

For our specific example of the robot we assume here that the data follow our previous deterministic hypothesis $\hat{y}(\mathbf{x};\mathbf{w})$ with **additive Gaussian noise**, or with other words, that the data in Figure 4.1C are Gaussian distributed with a mean $\mu = \hat{y}(x)$ depends linearly on the value of x,

$$p(\hat{y}|x;\mathbf{w},\sigma) = N(\mu = \mathbf{w}^T\mathbf{x},\sigma) \tag{4.8}$$

$$= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\hat{y}-\mathbf{w}^T\mathbf{x})^2}{2\sigma^2}\right) \tag{4.9}$$

This functions specifies the probability of values for $\hat{y}$, given an input x and the parameters $\mathbf{w}$ and σ. In the following we keep the parameter σ so that we only consider the variables $\mathbf{w}$ as free. This just helps to keep the formulas manageable, though including it should be straight forward.

Specifying a model with a density function is an important step in modern modelling and machine learning. In this type of thinking, we treat data from the outset as fundamentally stochastic, that is, data can be different even in situations that we deem identical. This randomness may come from an **irreducible indeterminacy**, that is, true randomness in the world that can not be penetrated by further knowledge, or this noise might represent **epistemological limitations** such as the lack of knowledge of hidden processes or limitations in observing states directly. The only important fact for us is that we have to live with these limitations. This acknowledgement together with the corresponding language of probability theory has helped to make large progress in the machine learning area.

We will now turn to the important principle that will guide our learning process which corresponds here to estimating the parameters of the model. While the parameterized hypothesis so far describes the form of the data, we need to estimate values for the parameters to make real predictions. We therefore consider now the examples for the input-output pairs, our training set $\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1...m\}$. The important principle that we will now follow is to choose the parameters so that the examples we have are most likely. This is called **maximum likelihood estimation**. To formalize this principle, we need to think about how to combine probabilities for several observations. If the observations are independent, then the joint probability of several observations is the product of the individual probabilities,

$$p(y_1, y_2, ..., y_m|\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m; \boldsymbol{\theta}) = \Pi_i^m p(y_i|\mathbf{x}_i; \boldsymbol{\theta}). \tag{4.10}$$

Note that y_i are still random variables at this point. But we now use our training examples as specific observations for each of these random variables, and introduce the **Likelihood function**

$$L(\boldsymbol{\theta}) = \Pi_i^m h(\boldsymbol{\theta}; y^{(i)}, \mathbf{x}^{(i)}). \tag{4.11}$$

The h on the right hand side is now not a density function, but it is a regular function (with the same form as our parameterized probability density function) of the parameters $\boldsymbol{\theta}$ for the given values $y^{(i)}$ and $\mathbf{x}^{(i)}$. Instead of evaluating this large product, it is common to use the logarithm of the likelihood function, so that we can use the sum over the training examples,

$$l(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = \sum_i^m \log(h(\boldsymbol{\theta}; y^{(i)}, \mathbf{x}^{(i)})). \tag{4.12}$$

Since the log function increases monotonically, the maximum of L is also the maximum of l. The maximum (log-)likelihood can thus be calculated from the examples as

$$\boldsymbol{\theta}^{\mathrm{MLE}} = arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}). \tag{4.13}$$

We might be able to calculate this analytically or use one of the search algorithms to find a maximum from this function.

Let us apply this to the linear regression discussed above. The log-likelihood function for this example is

$$l(\mathbf{w}, \sigma) = \log \Pi_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma^2}\right) \tag{4.14}$$

$$= \sum_{i=1}^m \left(\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma^2}\right) \tag{4.15}$$

$$= -\frac{m}{2}\log 2\pi\sigma - \sum_{i=1}^m \frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma^2}. \tag{4.16}$$

Thus, the log was chosen so that we can use the sum in the estimate instead of dealing with big numbers based on the product of the examples.

Let us now consider the special case in which the parameter σ is given. We can thus concentrate on the estimation of the other parameters w. Since the first term in the expression 4.16, $-\frac{m}{2}\log 2\pi\sigma$, is independent of $\mathbf{w}$, maximizing the log-likelihood function is equivalent to minimizing a quadratic error term

$$E = \frac{1}{2}(y - h(\mathbf{x};\mathbf{w}))^2 \Longleftrightarrow p(y|\mathbf{x};\mathbf{w}) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{(y - h(\mathbf{x};\mathbf{w}))^2}{2\sigma}) \quad (4.17)$$

This **error function** or **cost function** was a frequently used criteria called **Least Mean Square (LSM)** regression for parameters estimation when considering deterministic hypothesis. I terms of our probabilistic view, the LSM regression is equivalent to MLE for gaussian data with constant variance. When the variance is a free parameter, then we need to minimize equation 4.16. instead.

We have discussed Gaussian distributed data in most of this section, but one can similarly find corresponding error functions for other distributions. For example, a **polynomial error function** correspond more generally to a density model of the form

$$E = \frac{1}{p}||y - h(\mathbf{x};\mathbf{w})||^p \Longleftrightarrow p(y|\mathbf{x};\mathbf{w}) = \frac{1}{2\Gamma(1/p)}\exp(-||y - h(\mathbf{x};\mathbf{w})||^p). \quad (4.18)$$

Later we will specifically discuss and use the **ϵ-insensitive error function**, where errors less than a constant ϵ do not contribute to the error measure, only errors above this value,

$$E = ||y - h(\mathbf{x};\mathbf{w}||_\epsilon \Longleftrightarrow p(y|\mathbf{x};\mathbf{w}) = \frac{p}{2(1-\epsilon)}\exp(-||y - h(\mathbf{x};\mathbf{w})||_\epsilon). \quad (4.19)$$

Since we already acknowledged that we do expect that data are noisy, it is somewhat logical to not count some deviations form the expectation as errors. It also turns out that this error function is much more robust than other error functions.

4.3 Maximum a posteriori estimates

In the maximum likelihood estimation we assumed that we have no prior knowledge of the parameters $\boldsymbol{\theta}$. However, we sometimes might know which values of the parameters are impossible or less likely. This prior knowledge can be summarized in the prior distribution $p(\boldsymbol{\theta})$, and the next question is how to combine this prior knowledge in the maximum likelihood scheme. Combining prior knowledge with some evidence is described by Bayes' theorem. Thus, let us consider again that we have some observations $(\mathbf{x}, y)$ from specific realizations of the parameters, which is given by $p(\mathbf{x}, y|\boldsymbol{\theta})$, and the prior about the possible values of the parameters, given by $p(\boldsymbol{\theta})$. The prior is in this situation sometimes called the **regularizer**, restricting possible values in a specific domain. We want to know the distribution of parameters given the observation, $p(\boldsymbol{\theta}|\mathbf{x}, y)$, which can be calculated from Bayes's theorem,

$$p(\boldsymbol{\theta}|\mathbf{x}, y) = \frac{p(\mathbf{x}, y|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}'\in\Theta} p(\mathbf{x}, y|\boldsymbol{\theta}')p(\boldsymbol{\theta}')\mathrm{d}\boldsymbol{\theta}'}, \quad (4.20)$$

where Θ is the domain of the possible parameter values. We can now use this expression to estimate the most likely values for the parameters. For this we should notice that the

denominator, which is called the **partition function**, does not depend on the parameters $\boldsymbol{\theta}$. The most likely values for the parameters can thus be calculated without this term and is given by the **maximum a posteriori (MAP)** estimate,

$$\boldsymbol{\theta}^{\mathrm{MAP}} = arg\max_{\boldsymbol{\theta}} p(\mathbf{x}, y|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{4.21}$$

This is, in a Bayesian sense, the most likely value for the parameters, where, of course, we now treat the probability function as a function of the parameters (e.g., a likelihood function).

A final caution: ML and MAP estimates give us a **point estimate**, a single answer of the most likely values of the parameters. This is often useful as a first guess and is commonly used to **make decisions** about which actions to take. However, it is possible that other sets of parameters values might have only a little smaller likelihood value, and should therefore also be considered. Thus, one limit of the estimation methods discussed here is that they do not take distribution of answers into account, which is more common in more advanced Bayesian methods.

4.4 Multivariate causal modelling

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 4.3.

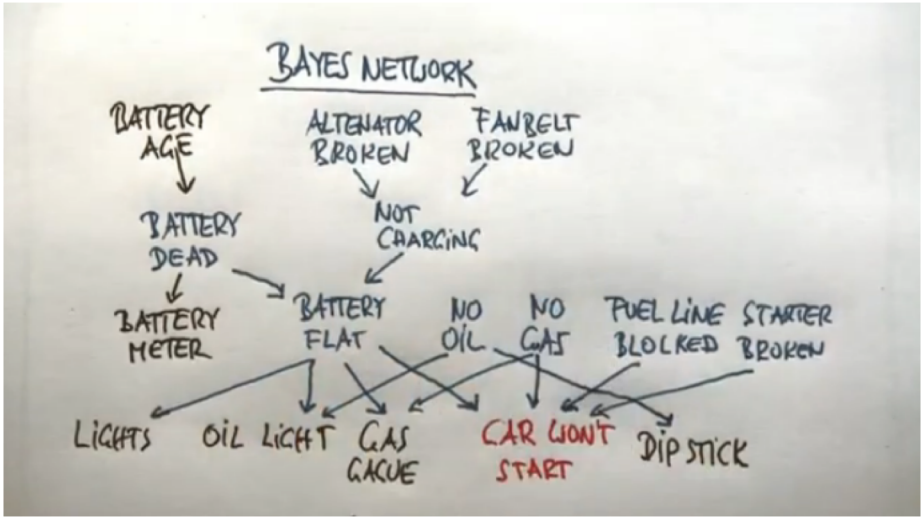

Fig. 4.3 Example of causal model.

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,

$$p(X, Y) = p(Y, X), \tag{4.22}$$

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.

Graphical causal models have been advanced largely by Judea Pearl, and the following example is taken from his book[3]. The model is shown in Figure 4.4. 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.

The joined distribution of the five variables can be factories in various ways following the chain rule mentioned before (equations 3.30), for example as

$$p(B, E, A, J, M) = P(B|E, A, J, M)P(E|A, J, M)P(A|J, M)P(J|M)P(M) \tag{4.23}$$

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

$$p(B, E, A, J, M) = P(B)P(E)P(A|B, E)P(J|A)P(M|A), \tag{4.24}$$

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

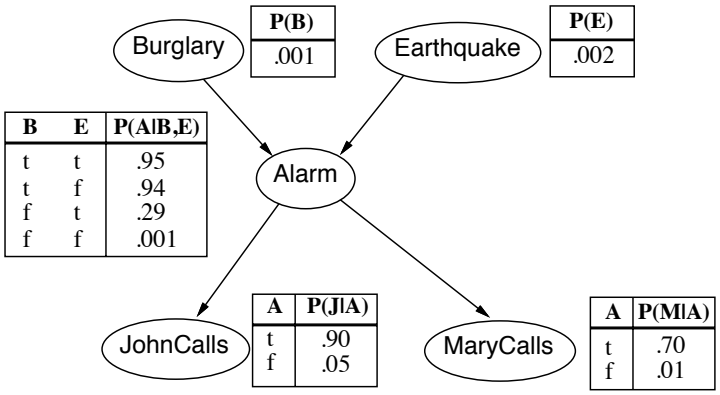

Fig. 4.4 Example of causal model.

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,

$$P(A = t|J = t) = \frac{P(J = t|A = t)P(A = t)}{P(J = t|A = t)P(A = t) + P(J = t|A = f)P(A = f)}$$

$$\approx \frac{0.9 * 0.0025}{0.9 * 0.0025 + 0.05 * 0.9975}$$
$$\approx 0.0434$$

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 Matlab 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.

4.5 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 demonstrate some of its features 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:

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