Bayesian Machine Learning

Proseminar Data Mining

Severin Bals
Fakultät für Informatik
Technische Universität München
Email: ga58fok@mytum.de

***Abstract*— The purpose of this paper is to give a generally intelligible introduction to Bayesian machine learning. Target audience of this paper is everyone, who has little to no experience in machine learning, but the active interest to gather some. The goal is to give a quick portion of Bayesian statistics, based on which the Bayesian way of learning is presented. In this process, a few different machine learning models are broached. Also bayesian networks, a type of graphical models is introduced to. Upon the end of this paper, the reader will have a proper entrance point to other more extensive papers or books about Bayesian machine learning.**

***Index Terms*—**

I. Introduction

Since many years the idea of artificial intelligence is widespread as for example in form of hyperintelligent robots in sci-fi movies. But on the current state of technology, these milestones are more and more science yet less fiction. Even though we are obviously not able to build such a robot yet, this scientific branch generated two big research topics: artificial intelligence and machine learning.
In modern Computer Science, machine learning forms a constantly growing field of research. It already finds usage in different areas like IT, traffic, health service or marketing. Due to the seemingly infinite amount of application areas, a lot of different approaches emerged, that consider various models and other factors. One of these factors is the uncertainty of predictions, which mainly statistical methods measure and deal with. The majority of these statistical approaches base on the Bayesian probability theory and are therefore collectively denoted as Bayesian machine learning, which is the topic of this paper. Individual members of it can be found in various fields, such as graphical, parametric or non parametric models. A few of these will be introduced in this paper. Hereby strategies for both classification and regression exist, as shown throughout the later chapters.

II. Mathematical derivation of Bayes' rule of probabilistic inference

To explain Bayesian Machine Learning, it is important to first understand the basics of Bayesian statistics. The core of these is Bayes' rule of probabilistic inference. The general idea behind this rule is according to Wooldridge [1]:"*Given our previous beliefs about an event, how should we revise the probability assigned to the event in light of the new evidence at hand?*".
Because of the importance of the rule for the following chapters, this section will derivate it mathematically.
Bayes' rule derives from two basic rules of statistics - the rule of **conditional probability** and the **product rule**.

Suppose the two events A and B. The rule of conditional probability defines the probability of event A, given that B is already true, in the following equation:

$$p(A|B) = \frac{p(A,B)}{p(B)}, p(B) > 0$$

,where p(A,B) equals the joint probability of two events. This on the other hand can be calculated using the product rule.
Suppose again the two events A and B, then the joint probability of the two equals after the product rule:

$$p(A,B) = p(A \cap B) = p(A|B)p(B) = p(B|A)p(A)$$

Applying the product rule to more than two random variables leads straight to the chain rule of probability. This is applied in the following

$$p(X_1, ..., X_n) = p(X_1)p(X_2|X_1)...p(X_n|X_1, ..., X_{n-1})$$

Combining the two rules from above directly defines Bayes' rule, which sets the foundation to the rest of this paper:

$$\mathbf{p(X = x|Y = y)} = \frac{p(X = x, Y = y)}{p(Y = y)} \quad (1)$$

$$= \frac{\mathbf{p(X = x)p(Y = y|X = x)}}{\mathbf{p(Y = y)}}$$

,where p(X = x) corresponds to the probability of the random variable X attaining the value x.
In Bayesian approaches of learning, computing conditional probabilities, as with Bayes' rule, is directly interpreted as inference, which will be shown in later chapters.

III. Bayesian statistics for machine learning

Bayesian probability interprets probability as the measure of confidence or belief, that a specific event will occur. In that respect traditional *frequentist probability* differs, as it describes probability as its relative frequency of occurence in a large amount of test repetitions.
This chapter explains the statistical basics for Bayesian machine learning.

A. Prior, likelihood and posterior

The three terms **prior, likelihood** and **posterior** are the most important concept in Bayesian machine learning. The following paragraphs are based on chapter 3 of Kevin P. Murphy's book [2], who uses the so called *number game* to show the correlation at an example. As it is a good explanatory instance of Bayesian machine learning problems, the next sections only have slight changes in values of the game. For the rest of this chapter the game works as follows. Given the concept 'Natural numbers between 1 and 100' and the data set $\mathcal{D}$, which contains arbitrary postive example in the field of the concept. What other values would be positive examples of the concept?
However in most cases $\mathcal{D}$ is given and you first have to predict, which concept this data originates from. Therefore you suppose a **hypothesis space** $\mathcal{H}$, containing several different concepts h. Those concepts,that are consistent with $\mathcal{D}$, form the **version space**. To find the fitting concept, the following parameters are needed.

1) ***Prior****:* Suppose $\mathcal{D}$ is given. Some concepts of the version space fit the data better than others. However you also have to observe, how likely it is, that such a concept occurs. For example h_1:= 'odd numbers that have less than three syllables' is less natural than h_2:= 'odd numbers'. This unfortunatily subjective ratio is called the **prior**. It enables putting our personal belief into our mathematical model. With that, h_1 will get a much smaller prior value assigned than h_2. The subjectivity causes prior distributions and results to differ from observer to observer.

2) ***likelihood****:* Suppose $\mathcal{D} = \{5, 15, 53, 57\}$, h_{five}:= 'all numbers containing 5' is obviously more likely to be the concept we search for than h_{odd}:= 'all odd numbers'. The probability of the dataset $\mathcal{D}$ occuring under the assumption of concept h is called the **likelihood** (:= p($\mathcal{D}|h$)). Unlike the prior, the likelihood can be deterministically computed, in this case using the extension of a concept, which is defined as all elements, that belong to it.
Suppose the concept extension h, then the likelihood in the case of the number game, after picking N samples can be calculated with the following equation.

$$p(\mathcal{D}|h) = \left[\frac{1}{|h|}\right]^N$$

3) ***posterior****:* The posterior, defined as the probability of the concept h given the data $\mathcal{D}$, represents the sought value distribution. It is calculated by the normalized product of likelihood and prior.

$$\textbf{posterior} = p(h|\mathcal{D}) = \frac{p(\mathcal{D}|h)p(h)}{p(\mathcal{D})} = \frac{\textbf{likelihood} \cdot \textbf{prior}}{\sum_{h' \in \mathcal{H}} p(\mathcal{D}, h')}$$

The denominator here is often referred to as *marginal likelihood*. You can easily recognize, that this equation resembles to Bayes' rule of probabilistic inference. Assuming we have a sufficient amount of data, the posterior forms a spike on a specific concept called the MAP estimate (Maximum a posteriori), since the prior is constant and only the likelihood changes. For an universal definition of the posterior (and prior, likelihood), one only has to change the variable h of concepts to a general probabilisty parameter θ. Assuming an uniformally distributed prior, the posterior only depends on the likelihood, so the MAP estimate is the concept with the highest likelihood. This is also called the MLE (Maximum likelihood estimate). Since the prior is a constant, the MAP estimate converges to the MLE.

B. MLE and MAP

The main goal of Bayesian machine learning is to get the best possible estimation of the probability parameter θ of the underlying mathematical model, even though sometimes an approximation is satisfying. The two most common strategies to find this optimal θ, MLE and MAP, are shortly explained in this section.

As already mentioned maximum likelihood estimation, or short MLE aims to calculate the parameter θ, under which the training data set $\mathcal{D}$ is most probable. The mathematical definition of this term reads:

$$\theta^{MLE} = \underset{\theta}{\operatorname{argmin}}\, p(\mathcal{D}|\theta)$$

As Mitchell describes in the second chapter of his unfinished book [3], the basic idea of MLE is, that the data $\mathcal{D}$ is more likely to be obtained, in a world (θ), where this Data is the most probable to appear. For most mathematical or machine learning models, a formula for calculating this maximum likelihood estimate θ^{MLE} can be derived by finding an alternative term of $p(\mathcal{D}|\theta)$. Optimizing the first derivation for a maximum will compute the one or many searched θ^{MLE}.
However in many cases, it is easier not to compute $\operatorname{argmin}_\theta p(\mathcal{D}|\theta)$, but $\operatorname{argmin}_\theta ln(p(\mathcal{D}|\theta))$, expecially in terms of derivating. This is an allowed alternative, since *ln(x)* increases monotonically with *x*. Because this method is quite broadly used, it is also called the log likelihood estimation.
Mitchell shows this procedure at the example of a coin getting tossed n times. The probability parameter θ describes the probability of the coin landing on heads, while $\mathcal{D}$ consists of the already seen coinflips. With α observed heads and β observed tails, we can form the equation:

$$p(\mathcal{D}|\theta) = \theta^{\alpha}(1-\theta)^{\beta} \quad 2.2.1$$

with $(1-\theta)$ as the probability of the coin landing on tails. Deriving the *ln* of the right side of this equation, produces a concave function. Equating this function to zero, yields θ^{MLE} = $\frac{\alpha}{\alpha+\beta}$. This result also matches with the naive approach to take the ratio of heads to total coin tosses as probability of heads.

The second main strategy of estimating θ is the maximum a posteriori estimation, or short MAP. On the contrary to MLE, the MAP estimate is the value of θ, that is most probable

with regard to the data $\mathcal{D}$. This condition can be expressed mathematically as follows:

$$\theta^{MAP} = \underset{\theta}{\operatorname{argmin}}\, p(\theta|\mathcal{D}) = \underset{\theta}{\operatorname{argmin}}\, \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \\ (= \underset{\theta}{\operatorname{argmin}}\, p(\mathcal{D}|\theta)p(\theta)) \tag{2}$$

The term on the right side of this equation derives from the definition of the posterior $p(\theta|\mathcal{D})$ from Bayes rule of inference as derived in the previous chapter.
Further it can be written as the term in the brackets, if $p(\mathcal{D})$ does not depend on θ. As this expression only differs from equation 2.2.1 by the extra $p(\theta)$, the result of the MLE can be reused to compute the MAP. Thus only the prior distribution $p(\theta)$ is missing.
According to Mitchell, the prior distribution can be derived from the underlying probabilistic model. For example, if a Bernoulli distributed random variable describes the data $\mathcal{D}$, a common form of the prior distribution is a Beta distribution. If the prior distribution is furthermore of the same form as the posterior distribution, it is commonly called the *conjugate prior*.
Using the same process as described for the MLE, one can now optimize the equation of θ^{MAP} and thereby find the sought MAP esimate.

IV. Bayesian networks

A Bayesian Network or belief network is a probabilistic graphical type of model, that is helpful for representing the conditional dependencies between a set of variables. They build a bridge between Bayesian statistics and Bayesian machine learning. There are even learning algorithms, that base on learning and inferencing from a bayesian network. A big area of application for bayesian networks is for example in the medical field.

A. Structure of a Bayesian network

Given a joint distribution over variables $\{x_1, ...x_n\}$, the chain rule of probability allows to write the joint distribution as:

$$p(x_1, x_2, ..., x_n) = p(x_1)p(x_2|x_1)...p(x_n|x_1, ..., x_{n-1})$$

As Kevin P. Murphy [2] states, the table representing such a set of variables, each having *K* possible states, would be of size $\mathcal{O}(K^n)$. This results, as the table then contains the cross product of each variable. This amount of data is too large to work with in an acceptable time complexity. Therefore the amount of table entries has to be minimized. According to Murphy [2] an efficient method for this is to take conditional independence into account. Conditional independence, noted as $X \perp Y|Z$ is defined as $X \perp Y|Z \iff p(X, Y|Z) = p(X|Z)p(Y|Z)$. If we now use the *first order Markov assumption*, that $x_{t+1} \perp x_1, ..., x_{t-1}|x_t$, the joint distribution is now calculated as:

$$p(x_1, ..., x_n) = p(x_1) \prod_{t=1}^{n} p(x_t|x_{t-1})$$

Thus each variable only depends on its direct predecessor and only a fraction of the original table entries are needed.
For distributions on higher orders or arbitrary variable collections, graphical models, such as Bayesian networks can be useful. Bayesian networks are graphical models, that consist of a directed acyclic graph or short DAG. Thus the vertices can be arranged in a topological ordering, so that each parent node is located before its children. The variables of a Bayesian network can be divided into two groups. Wooldridge [1] names these *hypothesis* variables and *information* variables. In the example of two students *A* and *B* being late, given the knowledge if the train is late, the variables, that represent if *A*/*B* is late, are hypothesis variables. On the contrary the variable, that defines if the train is late, is an information variable and thus the hypothesis variables have some level of dependency on it. Further, according to Murphy [2], the *ordered Markov property* is defined for these networks as $x_s \perp x_{pred(s)\setminus pa(s)}|x_{pa(s)}$, where pa() yields the parents of a node and pred() yields its predecessors. This definition expresses, that a node is only dependent on its immidiate parents.
How this affects the calculation of the joint distribution is shown here for the examplary DAG in Figure 1:

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_1, \cancel{x_2})p(x_4|x_1, x_2, \cancel{x_3})$$

Murphy [2] shows, that, with each node having $\mathcal{O}(F)$ parents, the complexity of $\mathcal{O}(K^n)$ from before decreases significantly to $\mathcal{O}(nK^F)$.
The most important benefit of Bayesian networks can be found in revising probabilities as a consequence of new observation, according to Wooldridge [1]. Assuming the model of the two students again, as shown as belief network and conditional probability table in Figure 2, we will now revise the probabilitiy of a train strike, given that student *A* is late. This is calculated with Bayes rule:

$$p(t = 1|A = 1) = \frac{p(A = 1|t = 1)p(t = 1)}{p(A = 1)} \approx 0.47$$

Furthermore the probabilities of *B* can be updated based on the new probability of the train. This process of updating the values upon a new observation is called *propagation*. Wooldridge also states, that in larger networks the propagation problem corresponds to a NP-hard problem.

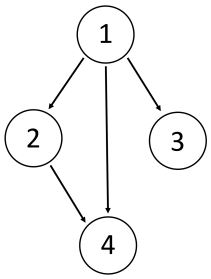

Fig. 1. Bayesian Network

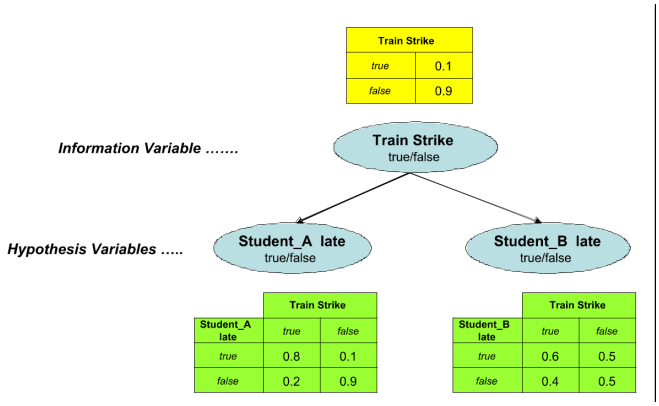

Fig. 2. An examplary Bayesian Network - Source: Scott Wooldridge [1]

B. Inference strategies

The following section is based on the fifth chapter of Bouckaerts [4] thesis.
Remco Ronaldus Bouckaert defines the calculation of conditional dependencies in Bayesian Networks as *inference*. The NP-hard problem of propagation, which he calls *exact inference*, led to another approach of infering on Bayesian networks, called *approximate inference*. Several algorithms for the latter task base on the principle of *simulation*. The main idea behind this principle is to produce a multi-set of configurations of the distribution, that is constituted by the network. The relative frequency of a specific variable to appear in this multi-set determines its approximated probability. Even though approximate inference is also NP-hard, Bouckaert states that the execution-time complexity is linear in $\mathcal{O}(\#\text{nodes} \cdot \#\text{samples})$ and the topology has less influence on it, than on exact inference.[1]

As already mentioned, simulation is an approach of approximate inference algorithms. Suppose the *sample space* $\mathcal{D}$, a function $f : \mathcal{D} \to \mathbb{R}$ and the equation:

$$\phi = \sum_{x \in \mathcal{D}} \frac{f(x)}{|\mathcal{D}|}$$

Simulation algorithms approximate this ϕ, using a *simulation scheme*. This scheme determines, how the elements, also called *sample trials*, of a sample are selected from $\mathcal{D}$. Suppose the multi-set $\{x_1, ...x_n\}$ is a sample of size n, then $\{f(x_1), ..., f(x_n)\}$ is the corresponding simulation of f. An approximation of ϕ would now be the mean. Yet, if some values f(x) are exceptionally bigger than most, the mean is biased toward those x. To find a reasonable ϕ, simulation schemes for Bayesian networks consist of three components. *Sample distributions* are the first auxiliary tool and define a joint probability distribution over the sampled nodes. A common choice for the sample distribution is a product of distributions, that belong to elements of the power set of all sampled nodes. The second component is called the *sample trial generator*. A value on the basis of the sample distribution, is now assigned to every sample trial. At last, the *scoring method* counts the sample scores of each variable and assigns weights for each configuration, by which the approximation values are calculated.
Even though approximation inference algorithms are not exact, the results are sufficient as we often set a threshold value instead of exact accuracy.

[1] # means "number of"

V. Bayesian machine learning models

Bayesian machine learning has asserted itself as a term for many probabilistic machine learning strategies, which are not necessarily based on bayesian statistics. In addition, Bayesian approaches find more and more application possibilities in new and already established machine learning methods. Other than most methods, Bayesian approaches deal with the big issue, that parameter estimations are inherently uncertain due to limited available training data. They often deal with measurement noise and model uncertainty too. Based on these core thoughts, this chapter presents a few different Bayesian machine learning models. Up to now this paper only had examples, consisting of discrete data values. The designation of such mathematical models in machine learning is *classification*. Alongside with these, there is a second group of machine learning methods, called *regression*. Regression is used for models made on data over a continuos value range. This chapter demonstrates both classification and regression methods for assorted Bayesian machine learning strategies.

A. Linear models

To present the applicability of Bayesian machine learning approaches, this section will first demonstrate standard linear modelling and subsequently show Bayesian linear models. Inspiration and information of this chapter orientate on David Barber [5]

*1) **General approach**:* Suppose the training data set $\mathcal{D} \coloneqq \{(x^n, y^n)|n \in \{1, ..., N\}\}$ and $N = |\mathcal{D}|$, with the scalar input x^n and scalar observed output y^n. A linear regression fit for this model is:

$$y(x) = a + bx$$

The goal now is to find values for a and b, so that the difference between observed output y^n and linear regression fit $y(x)$ is minimal. The procedure that calculates this discrepancy is called *ordinary least squares*, and computes through the following equation:

$$E(a, b) = \sum_{n=1}^{N} [y^n - y(x^n)]^2 = \sum_{n=1}^{N} (y^n - a - bx^n)^2$$

Differentiating this equation with respect to a and b and equating to zero, yields a system of two linear equations, from which can determine the values for a and b, that optimize our discrepancy between y^n and linear regression fit. This can be graphically represented in a 2-dimensional plane, in

which the values of y^n are drawn, as the line, that has the least possible distance to the x^n as seen in figure 3. Another way would be to minimize the orthogonal difference between the points and line.

This approach can be generalised for vector inputs $\mathbf{x}$. Hence, $\mathcal{D} = \{(\mathbf{x}^n, y^n)|n \in \{1, ..., N\}\}$ and the *linear parameter regression model* (LPM) is defined by $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$, with the vector function ϕ and parameter vector $\mathbf{w}$. Thus the linear regression fit case can be reconstructed, using $\phi(x) = (1, x)^T$ and $\mathbf{w} = (a, b)^T$.
Again we compute the sum of squared differences between the observed output and the predictions of our model and define it as *train error* as follows.

$$E(\mathbf{w}) = \sum_{n=1}^{N}(y^n - \mathbf{w}^T\phi(\mathbf{x}^n))^2$$

Differentiating and equating to zero, now yields a linear equation system, which can be solved to get the optimization of each of our parameters.
This can further be extended to vector outputs. Therefor we use a seperate weight vector $\mathbf{w}_i$ for each output. This way we can apply the last equation on every specific output, to find its' own optimal weight parameters.
Another possible extension of our derived model is the attempt to control the complexity of finding the linear function, rather than it fitting the training data as good as possible. For that a regularising term with strength scalar can be added to the train error:

$$E'(\mathbf{w}) = E + \lambda R(\mathbf{w})$$

As Barber [5] shows, a common choice for *R* is the Identity matrix $\mathbf{I}$. Using a scalar multiples of the Identity matrix to control ill-posed problems is called *ridge regression* and is wide spread in the area of machine learning.

This regression method can also be used to derive a classification problem, by using a function as for example the *logit*. This function maps the training data onto their probability with respect on the weights. Then a so called *decision boundary* seperates the training data into two different classes (for simplicity 1 and 0) based on the respective probability. The principle behind this is very similar to *the perceptron*, a very important early model of Artificial Intelligence.

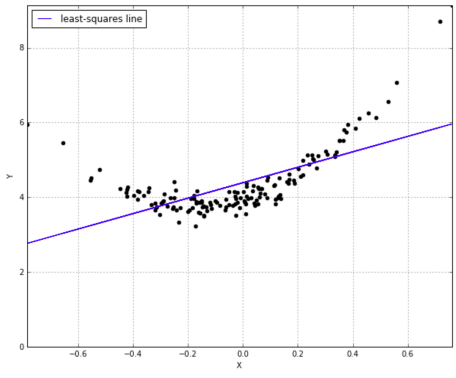

Fig. 3. Ordinary least-squares line for a set of data points. It is noticeable, that a line is not necessarily the best 'interpolation'. Source: Katherine Bailey [6]

*2) **Bayesian approach**:* The models above all computed the best possible fit and thus they were trained according to maximum likelihood. They also dont deal with the uncertainty of the parameters, that comes with the limited amount of training data.
Suppose again a set of observed training data $\mathcal{D} = \{(\mathbf{x}^n, y^n)|n \in \{1, ...N\}\}$, then the observed output y results from adding a noise η to the model output. If the output of our model is describable as a function $f(\mathbf{x};\mathbf{w})$ the following equation ensues:

$$y = f(\mathbf{x}; \mathbf{w}) + \eta$$

This noise η can be for instance Gaussian distributed. If each data tuple is generated identically and independently, then the likelihood can be computed as:

$$p(\mathcal{D}|\mathbf{w}) = \prod_{n=1}^{N} p(y^n|\mathbf{w}, \mathbf{x}^n)p(\mathbf{x}^\mathbf{n})$$

One can now involve his initial belief in the adequacy of the parameter configuration as a prior distribution p($\mathbf{w}$). Thereby Bayes' rule of inference deduces the proportionality of our posterior weight distribution and the product of likelihood and prior ($p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$).
Since the input-output mapping of the non-Bayesian LPMs can be non-linear, we can prohibit extreme output values by punishing big weight values through an adjusted prior. Barber [5] refers to an example for a natural weight prior as the following Gaussian distributed prior

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

The *precision* α here is the inverse of the variance. Thus the matching posterior distribution is also Gaussian distributed, conditional on certain *hyperparameters*. Hyperparameters in Bayesian statistics are parameters, that originate from the prior or posterior distribution in contrast to the parameters, that originate from the underlying mathematical model. There are various methods to determine these hyperparameters, like the marginal likelihood (ML-II) or through the gradient, which Barber [5] lists in his book.

As before in the standard linear model, a classification method can be directly derived. The difference here is, that the uncertainty of estimations of the weights will now be attended to by setting a prior distribution (for example Gaussian). This is a reasonable improvement, as estimation errors in this model can lead to assigning data to false classes.

B. Gaussian Processes

As the previous section demonstrated, linear model regression is a parametric type of prediction. That is because two parameters were searched, so that the associated linear function fits the data points in the best possible way. Thus the output of additional data input can be calculated with the newly aquired model. Suppose a set of data points, which can not be sufficiently represented by a linear, but with a quadratic or cubic polynomial as shown in Figure C. This requires to compute more than two parameters. In conclusion if the the data points best fitting function is unknown, so is the amount of parameters, that have to be computed. To evade this problem, this section introduces to *Gaussian processes* as an example for non-parametric prediction models.

As Kathrine Bailey [6] phrases, "Gaussian Processes (GPs) are the natural next step in the journey" of linear models. To develop Gaussian Processes from the Bayesian linear models introduced in the last chapter, once again suppose $\mathcal{D} = \{(\mathbf{x}^n, y^n)|n \in \{1, ..., N\}\}$, with the observed inputs $\mathbf{x}^n$ and outputs y^n. On this basis, the following section is oriented towards Murphy's [2] chapter on Gaussian processes. Thus we make the assumption $\exists f : y_i = f(\mathbf{x}_i)$, where our observation of y_i could be corrupted by noise. The obvious approach is to find a distribution over a set of functions $p(f|\mathbf{X}, \mathbf{y})$, which all fit the current model and can be used to predict future outputs. Therefor GPs follow the bayesian procedure and define a prior over this set of functions, that is used to calculate the posterior of the functions. For terms of simplicity, the distribution of each specific function is defined on a finite set of points. Thereby as Rasmussen & Williams [7] state in definition 2.1 of their book, "a Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution".
Further GPs have a mean μ and a covariance $\mathbf{\Sigma}$, where $\Sigma_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, that smoothes the function with the approach that for similar input, the output is also similar. $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is a positive definite kernel function. These two factors generate the set of functions, over which the sought distribution is. An example of a kernel function is $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-||x_i - x_j||^2}$.

Assume the prior for a regression method as a GP $f(\mathbf{x}) \sim GP(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$, with mean function $m(\mathbf{x})$ and covariance function $\kappa(\mathbf{x}, \mathbf{x}')$. From the input training data $\mathbf{X}$, the mean and covariance function generate a set of functions.[2] By definition any subset of data points build a joint Gaussian $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mu, \mathbf{\Sigma})$, where $\Sigma_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and $\mu = (m(\mathbf{x}_1), ..., m(\mathbf{x}_N))$. A common choice for the mean is $m(\mathbf{x}) = 0$ and thus $\mu = \mathbf{0}$.
For a model without observation noise, the output directly corresponds to $f(\mathbf{x})$. For our training data $\mathbf{X}$ we want to know what the outputs $\mathbf{f}_*$ for a test set $\mathbf{X}_*$ is. By definition it is already given, that $\mathbf{f} = \mathcal{N}(\mu, \mathbf{\Sigma})$. Therefore the new joint distribution is defined as follows:

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} \Sigma_{1,1} & \cdots & \Sigma_{1,N} & \Sigma_{1,*} \\ \vdots & \ddots & \vdots & \vdots \\ \Sigma_{N,1} & \cdots & \Sigma_{N,N} & \Sigma_{*,*} \end{pmatrix} \right)$$

Standard rules can now be used to calculate μ_*, $\mathbf{\Sigma}_*$ and from these finally the posterior distribution $p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f})$. The result can be seen in Figure 4, which on the left shows a short sample of functions of a GP prior. The right shows the corresponding GP posterior, after five data points were observed. The grayed out region is the 95% confidence area, on which the functions of the posterior distribution are restricted. This demonstrates, that the uncertainty between data points increases with the distance between them.

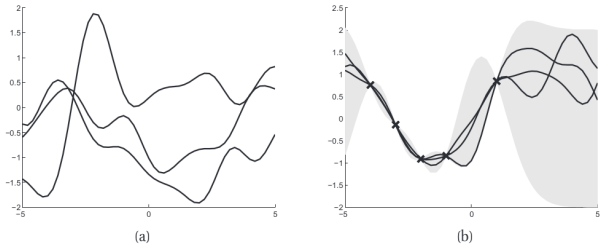

Fig. 4. Plot of a Gaussian process, showing the prior (a), and the posterior after 5 input data points (b). Source: Kevin P. Murphy [2]

VI. Conclusion

As this paper showed, Bayesian machine learning applies to many different problems and models, and is steadily developed. It provides both measurement and solutions to uncertainty and the prior belief in the model. As a result, probability and statistics apparently connect algorithm strategies, that instinctivly seem to be diverse. This makes Bayesian machine learning one of the most important approaches in data mining and machine learning. Thus the earlier one starts to read up on this subject the better. With this fundamental introduction to Bayesian machine learning, the reader is now ready to investigate further and to utilize these methods for own purposes.

References

[1] S. Wooldridge, "Bayesian belief networks," *CSIRO centre for complex systems science*, 2003.
[2] K. P. Murphy, *Machine Learning, A Probabilistic Perspective*. The MIT Press, 2012.
[3] T. M. Mitchell, "Estimating probabilities," http://www.cs.cmu.edu/unfinished book.
[4] R. R. Bouckaert, "Bayesian belief networks: from construction to inference," Ph.D. dissertation, University Utrecht, 1995.
[5] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.
[6] K. Bailey, "Gaussian processes for dummies," http://katbailey.github.io/post/gaussian-processes-for-dummies/, August 2016, accessed: 2018-06-01.

[2]rather a set of values close to each other so they appeal like a function

[7] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes in machine learning*. The MIT Press, 2006.
[8] M. Seeger, "Bayesian modelling in machine learning: A tutorial review," Saarland University, Room 116, Campus E1.4, 66123 Saarbruecken, Tech. Rep., March 2009.
[9] K. B. Korb and A. E. Nicholson, *Bayesian artificial intelligence*, ser. Computer Science and Data Analysis Series. CRC press, 2010.
[10] N. De Freitas, "Machine learning - introduction to gaussian processes," https://www.youtube.com/watch?v=4vGiHC35j9s, February 2013, video recordings of lecture.
[11] S. J. Russell and P. Norvig, Eds., *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.