Bayesian Machine Learning

Seminar Data Mining

Niclas Mettenleiter
Department of Informatics
TU Munich
Email: niclas.mettenleiter@tum.de

***Abstract*— Bayesian Machine learning (BML) is a branch of Artificial Intelligence that gives models a measure of uncertainty while making them robust to over-fitting and trainable with small amounts of data. BML methods are applied in critical classification models, where miss-classification leads to major problems as for example object-detection in autonomous driving. Besides that they are used when one only has a small data set, since they also work well with less amounts of data. This paper introduces the necessary concepts to understand BML, shows the applications of Bayesian methods to Linear Regression, Neural Networks and Stochastic processes and discusses how they compare to their non-Bayesian counterparts.**

***Keywords*— Artificial Intelligence, Bayesian Machine Learning, Bayesian Neural Networks, Data Mining, Linear Regression, Gaussian Processes,**

I. Introduction

There are two branches of Machine Learning: frequentist Machine Learning, where the model does a point-wise Estimation of the weights, and Bayesian Machine Learning, where the model learns probability distributions as weights. When massive amounts of data are available, the uncertainty of the model is not important and no prior assumptions about the data exist one chooses the frequentist approach. In other areas like in medical applications it is important to know whether the model is 2 or 99 percent sure if the patient has cancer. Besides that, Bayesian methods are robust to over-fitting and allow to incorporate our prior knowledge or belief into the models. This allows to guide the model into a certain direction to check assumptions or help in finding the optimal solution. When doing a full Bayesian approach we can incorporate prior beliefs about the data in a prior distribution. Then the model observes data and calculates the posterior from the data and the prior. The posterior then becomes the new prior, the model observes new data, and the process starts again. We end up with the true posterior distribution of the parameters (Fig. 1. last row, second column).
Given that, the question arises why to use Frequentist methods at all and the reason for that is, the bigger the number of parameters is the costlier the computation of the posterior gets and since we are living in the age of big data, where models that have millions of parameters are not seldom, exact computations with this approach are impossible. In this paper, we will look at Bayesian methods in linear regression, followed by applications to neural networks and lastly Gaussian processes as non-parametric estimators.

A. Probability review

First, we inspect Bayesian and Frequentists, then we introduce Bayes' Theorem, the foundation of Bayesian Machine Learning, followed by a short paragraph on the Gaussian distribution.

1) Bayesian vs. Frequentist: Frequentists determine uncertainty through repetition of a well-defined statistical procedure, like throwing a coin and taking the long-run average afterwards. Bayesians specify a distribution (prior) that fits the data first and then use data to update the prior to find an adjusted parameter distribution (posterior).
Frequentist methods only work when the number of data points is much bigger than the number of parameters, while the Bayesian approach works for all sizes of data [1]. If a coin flip results in 53 times heads in 100 repetitions the probability of heads for a Frequentist is 53 percent. Bayesians start by choosing a proper prior distribution, mostly a Gaussian with a mean of 50 given this problem statement, and update this prior after each coin toss to end up with the true posterior after 100 throws [2], [3]. To summarize, the Frequentist believes the parameters W are fixed and the data X is random and therefore tries to find the W that explains the data best through a Maximum Likelihood Estimation (MLE), while the Bayesian thinks the data X is fixed and the parameters W are random and tries to find the parameter through inferring a posterior using Bayes' Theorem [4].

2) Gaussian Distribution: While one big part of Bayesian Machine Learning (BML) is Bayes' Theorem, the other one is distributions, which allow to encode prior beliefs into models and deliver a measure of uncertainty. In this paper, the focus is on the Gaussian distribution (GD, often also referred to as the normal distribution) because it is easy to understand, general and often used; furthermore, the concept for inferring is similar for other distributions. Through choosing a GD to model the distribution of a random variable one implicitly assumes that small deviations are common while big deviations are seldom. The GD $\mathcal{N}$ has two parameters: the mean μ and the variance σ^2 or in the multivariate case the co-variance matrix Σ. The plot of the univariate probability density function has a bell-like shape with the mean determining where the mode is and the variance determining how slim the plot is. We define it as:

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\Big(\frac{(x-\mu)^2}{2\sigma^2}\Big).$$

Important to note is that the fraction in front of the exponential function only has the purpose of normalizing the term such that all probabilities sum up to one, so we omit that in some calculations [5]. We now know the properties of the GD that are important in this paper. Next, we look at Bayesian and Frequentist ways of inferring a distribution.

3) Bayes' Theorem: The basis of BML is Bayes' Theorem which builds upon the concept of conditional probability. $Pr[X|Y]$ reads as "the probability of X given Y" and describes the probability of X under condition Y. As an example there are two events A = "it is raining", B = "the street is wet" and we want to know the probability of rain given the event that the street is wet. To get the conditional probability $Pr[A|B]$ we take the number of events when it is raining and the street is wet and divide it by the number of events when the street is wet. We define conditional probability as

$$Pr[A|B] = \frac{Pr[A \cap B]}{Pr[B]}.$$

To calculate $Pr[A \cap B]$ we multiply the equation by $Pr[B]$ and to get $Pr[B]$ we use the law of total probability that states $Pr[B] = \sum_A Pr[B|A]Pr[A]$. We sum up overall occurrences of B given every element of the probability space. Combining those three building blocks we end up with Bayes' Theorem:

$$Pr[A|B] = \frac{Pr[B|A]Pr[A]}{\sum_A Pr[B|A]Pr[A]}.$$

In BML the goal is to find the posterior parameter distribution $Pr[W|X]$, given the prior distribution of parameters $Pr[W]$, the evidence $Pr[X]$ that is equal to $\sum_A Pr[X|W]Pr[W]$ and the likelihood $\mathcal{L}[X|W]$(we substituted Pr with $\mathcal{L}$ to clarify that this is a likelihood) [4], [6], [7].
We now have the foundation to do BML, in the upcoming chapter we explain the likelihood term and how Frequentists approach the parameter estimation problem.

4) Likelihood: Instead of using Bayes' Theorem, Frequentists use the term likelihood to infer parameters. The likelihood term $P(W|X)$ is read as "how likely is the parameter W given the data X". The Frequentist assumes there is a fixed parameter that explains the data best or "a parameter that is most likely given the data". Therefore the approach is to maximize this quantity to find the parameter that fits the data best. This process is also known as Maximum Likelihood Estimation (MLE).
We now have the basic building blocks required to understand Bayesian Machine Learning.

II. Bayesian Linear Regression

Linear Regression (LR) is the most basic and intuitive Machine Learning algorithm, but there are different approaches with their own advantages and disadvantages. We will compare Bayesian LR and Frequentist LR and discuss the problem of finding a proper regularization term for Frequentist LR and why a regularization term is obsolete for Bayesian LR. This chapter lays the foundation for Bayesian Neural Networks.

A. *Basic Linear Regression*

First, we look at the easiest way of doing LR. Given some data X with n features we fit a line to that data that predicts a numeric value for new X' :

$$f : \mathbb{R}^n - > \mathbb{R}.$$

An application is the prediction of the purchase prices of a flat given attributes like the location, the number of rooms and overall size. The most basic model looks like this: we minimize the loss function L given as

$$L = \frac{1}{2}||\hat{y} - y||_2^2$$

with $\hat{y}$ denoting the actual value and y denoting the prediction function $w^T\phi(X)$ with w denoting our weights and $\phi(X)$ denoting basis functions. The extension with basis functions allows us to incorporate non-linear terms but for formulas in this paper, we assume $\phi(X) = X$ for simplification. Solving this minimization problem incorporates gradient descent methods [8].
Minimizing that function is also called the Least Squares problem, that often leads to over-fitting especially in the case of polynomial basis functions. Over-fitting denotes the phenomenon of high accuracy on the training set and low accuracy on the test set.
As we want a model that makes good predictions for new data, we extend our model to

$$L = \frac{1}{2}||\hat{y} - y||_2^2 + \frac{\lambda}{2}||W||_2^2$$

with a regularization term, controlled by the new hyper-parameter λ, that penalizes high values of W that lead to over-fitting. Hyper-parameter fitting is difficult and there are limitations as we will see in the next sections [9], [10].

B. *MLE Regression*

Least Squares Problem and Ridge Regression are linear models with no probabilities involved. In this section, we will examine how the Frequentist approaches LR. We define the model as

$$\hat{y} = y + \epsilon \ \Leftrightarrow \hat{y} = w^T X + \epsilon, \text{ with } \epsilon \text{ denoting the error.}$$

We now assume that y is a Gaussian distributed random variable $y \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu = y = w^T X$ and some variance σ^2. Now the error ϵ is part of the GD. So we rewrite the Likelihood $\mathcal{L}(\hat{y}|w, X) = \mathcal{N}(\hat{y}|w^T X, \sigma^2)$. As the Frequentist thinks the parameter W is fixed, we do a MLE to find the parameter W_{MLE} that is most likely given the data and end up with the problem

$$W_{MLE} = argmax_W \mathcal{N}(\hat{y}|w^T X, \sigma^2).$$

Since this is an optimization problem, we can leave out the constant normalization term $\frac{1}{\sqrt{2\sigma^2\pi}}$ to make the model simpler. We apply the logarithm to the maximization problem to make it a minimization problem that is easier to compute. We end up with

$$W = argmin_W \frac{1}{2\sigma^2}(\hat{y} - y)^2.$$

Setting $\sigma = 1$ gives us the least squares problem. We showed how the frequentists approach parameter-estimation at the example of LR and that FLR is the same as doing the Least-Squares-Problem, which is surprising since the approaches are very different. [5], [10].
In the next section, we show an advanced way of doing parameter estimation through Bayesian methods.

C. MAP Regression

By looking at LR having a Bayesian perspective, we distinguish between the partial Bayesian (PB) approach, that uses a Maximum a Posteriori (MAP) Estimate and the full Bayesian (FB) approach. The MAP method is also Frequentist LR but has the advantage of incorporating prior knowledge into the model. While PB is applicable in many areas FB has some restrictions especially when we deal with a huge number of parameters.
We start with PB. To estimate the posterior $P(W|\hat{y}, X)$, we multiply the Likelihood $\mathcal{L}$ we derived above with a prior distribution of the parameters $P(W|\mu, \sigma^2)$:

$$P(W|\hat{y}, X) = P(\hat{y}|X, W)P(W|\mu_0, \sigma_0^2).$$

We define our prior as GD $P(W|\mu_0, \sigma^2) = \mathcal{N}(\mu_0, \sigma^2)$ with the parameters μ and σ^2 that allow to specify the prior the way we think it fits the data best [7]. When there is no reason not to so we assume $\mu = 0$ and again ignore the normalization term to simplify to

$$P(W|\mu_0, \sigma_0^2) \propto exp\Big(\frac{W^2}{2\sigma_0^2}\Big).$$

Setting $\lambda = \frac{1}{\sigma_0^2}$ and taking the logarithm leads us to our final derivation:

$$W = argmin_W \; -\frac{1}{2}||\hat{y} - W^T X||_2^2 - \frac{\lambda}{2}||W||_2^2$$

that should look familiar [5], [10]. We showed that the partial Bayesian approach is equal to the Frequentist ridged regression with the advantage of specifying a prior to encode our prior knowledge into the model. Next, we will see how the FB approach looks like.

D. Full Bayesian Approach

What we did in the last section was a pseudo-Bayesian way of doing linear regression. The attentive reader realized that we left out the evidence term of the posterior in the last section since it is very hard to compute. This is allowed since constant terms do not change the optimization problem. The key idea of the FB approach is to step by step add new data points (x', y') and for each new point calculating a new posterior which then is used as the prior for the next posterior estimation. So when we have k data points, we have to do k-times the calculation of the posterior. which is computationally expensive since now the evidence term has to be computed. [5].

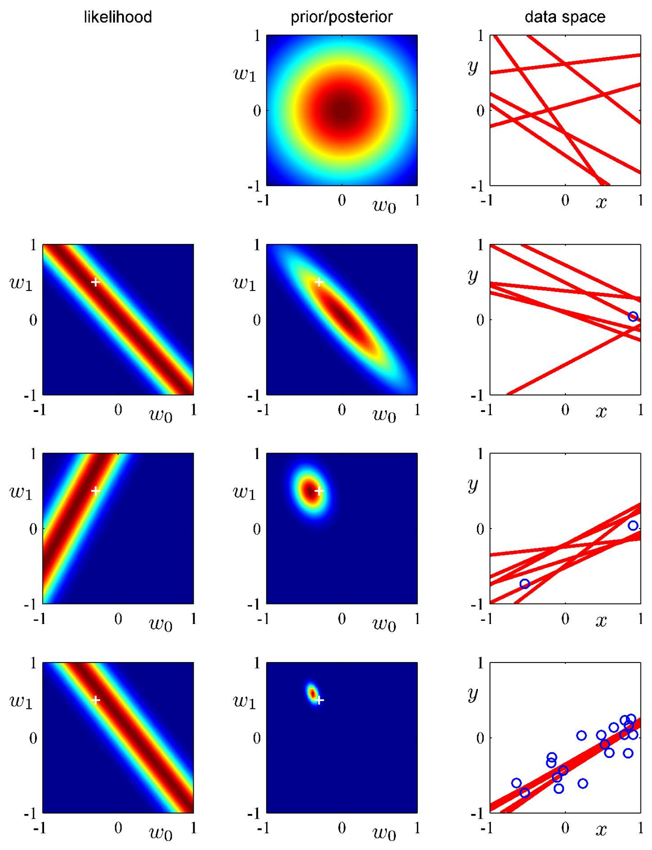

Fig. 1. Iterations of a full Bayesian approach on LR [6]

Fig. 1 shows some iterations of this sequential process. The left-hand column contains the plots of the likelihood function $\mathcal{L}(y'|X, W)$ that states the probability of predicting y' given the data. So it is a function of our weights W. The prior/posterior column shows how the distributions change over time while the data space shows six samples from our model $y(x, W) = w_0 + w_1 x$ with weights drawn from our prior distribution. The first row shows the situation before we add any points.
In the second row, you can see what happens if we add the first data point (x', y') (blue point in data space). In non-mathematical terms, the likelihood function has the constraint that the red line has to be close to the data point, represented with a white cross, while close depends on the variance σ^2 of the distribution. To get the posterior, we multiply our prior $P(w|\hat{y}, x)$ with the likelihood function. This leads to our new distribution in the middle column. We draw six random pairs of weights from our adjusted distribution and get a data space where each line passes the neighborhood of our inserted data-point, so our prediction improved.
The third row shows what happens if we add a second data point. The process is the same: we insert a new point but this time we use the distribution we got in the last step and multiply it with the likelihood function of the newly inserted point to update our distribution. This leads to the middle plot

in row three, which already is a big improvement compared to our distribution in the beginning, since two points are enough to define a line. A key takeaway is that the distribution we got in row three could also be obtained by applying the likelihood functions of the two inserted points to our first iteration. Row four shows us the result after 20 iterations. If the number of iterations approaches infinity, the distribution converges to the true values of the parameters. Predicting requires a function which gives us the following estimation for a new value x:

$$P(y'|\hat{y}, x) = \int_w P(y'|x', w)P(w|\hat{y}, x) = \mathbb{E}_W\left[P(y'|x', w)\right]$$

[6]. Now we saw how to do a FB Inference and what posterior estimation looks like. This chapter showed different approaches of inference for the weights of a linear model.
To summarize: Frequentist Regression is equal to the Least Squares Problem and tends to over-fit. To avoid over-fitting, a regularization term is introduced and we have a Ridge Regression model that is mathematically equal to the Bayesian MAP estimate with the advantage of adding knowledge through a prior distribution into the model. We also saw FB inference which leads to the exact posterior that allows to make the best estimations, but is not scalable and becomes impossible to do with a large number of parameters. The following chapter shows another way of approaching LR through parameter-free models.

III. Gaussian Processes

Gaussian Processes (GP) are a subset of Stochastic Processes (SP) where we assume every point of our training data as random variable. They are used to model systems that randomly change over time [11]. To give an introduction to that topic we go back to our Bayesian linear regression model by looking at our assumption $\hat{y} = f(X) + \epsilon$ with $f(X) = W^T X$. In case of a linear model we have

$$W = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad X = \begin{bmatrix} 1 \\ x \end{bmatrix} \text{ s.t. } y = W^T X \Leftrightarrow y = \theta_0 + \theta_1 x$$

We there try to fit the weights W such that our error function is minimal and generalizes well to new datapoints. Therefore, we assume a prior distribution that is on the weights W and update it step by step for each new point to get increasingly better posterior distributions. In GP we assume a prior distribution over all functions $f(x)$ that are consistent given our data $\mathcal{X}$. To give an intuition on what this means, think of data-points that follow the parabola $y = x^2$. No matter what weights you choose, $y = \theta_0 + \theta_1 x$ will not be able to make a good fit, but when we incorporate another parameter θ_2 we will get the model $y = \theta_0 + \theta_1 x + \theta_2 x^2$, which will give a good prediction for new data. In GP's Instead of defining how many parameters we want in our regression model, we look at all functions that fit our finite set of data points, no matter how many parameters they have. Since the number of functions is infinite, we have to make some constraints to find a reasonable prior.

We get our constraints intuitively through thinking about what we can alter. There is the domain of values we are interested in, the mean of all values in that domain and how smooth the plot is. While domain and mean are straightforward to define, we use the co-variance matrix Σ to define the smoothness of the function and to ensure that inputs which are close together produce outputs that also are close together [10], [12]. In math terms: given some finite set of N points $\mathcal{X} = x_1, x_2, ..., x_N$ a GP assumes that $Pr[f(x_1), f(x_2), ..., f(x_N)]$ is jointly Gaussian distributed with mean $\mu(x)$ and co-variance matrix $\Sigma(\mathcal{X})$ with $\Sigma_{ij}(\mathcal{X}) = \mathcal{K}(\mathcal{X}_i, \mathcal{X}_j)$ and $\mathcal{K}$ denoting a positive definite kernel function [10].

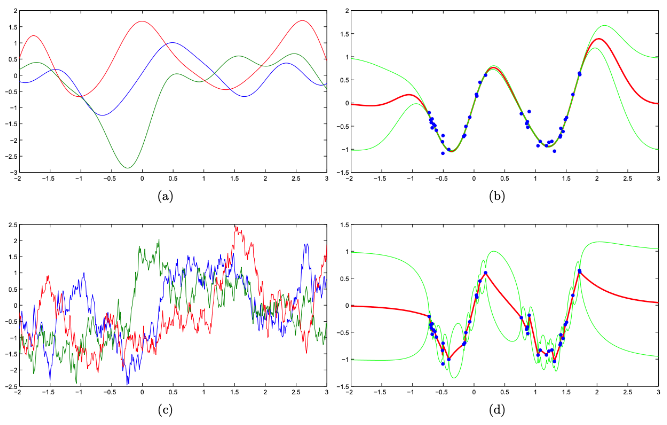

Fig. 2. a) shows three samples from a smooth GP prior, c) shows 3 samples from a fuzzy GP prior b) and d) show the corresponding posteriors after adding points. The red lines in b) and d) represent the interpolation of the given points while the green lines represent $\mathbb{E}[f(x)] \pm 2std(f(x))$ [9].

A. GPs for regression

We define the prior of the Gaussian process as

$$f(x) \sim GP(m(x), \mathcal{K}(x, x'))$$

with the mean function

$$m(x) = \mathbb{E}[f(x)]$$

and our squared exponential kernel function

$$\mathcal{K}(x, x') = \sigma_f^2 exp(-\frac{1}{2l^2}(x - x')^2).$$

σ controls the vertical while l controls the horizontal deviation. We use our kernel function to define the matrix K with the property $K_{ij} = \mathcal{K}(x_i, x_j)$ which gives us the similarity of x_i and x_j. Furthermore, K_* denotes the similarity between the test set and the trainingset and K_{**} the similarity between the test set data points. Bear in mind that we want to get the function values f_* given new points x_*, the function values f and datapoints x. For known x the GP interpolates the corresponding function values f as you see in Fig. 2, represented by the red line. The joint distribution of the GP has the following form

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left[\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} K & K_* \\ K_*^T & K_{**} \end{bmatrix}\right]$$

Through some mathematical derivations we get the posterior

$$p(f_*|X_*, X, f) = \mathcal{N}(f_*|\mu_*, \Sigma_*)$$

where μ_* and Σ_* are calculated in terms of K and μ which you can look up in [10]. As stated in Fig 5 the area between the green lines in the posteriors b) and d) represent a 95% confidence interval, so in comparison to a normal linear regression we gained a couple of things: first we do not have to specify the number of parameters θ, second we have a model robust to over-fitting, that gives us confidence intervals and converges quickly. On the other hand, GP's are only applicable when dealing with small numbers of parameters and become intractable to compute when that, number increases. Besides that there is the critical step of choosing a good kernel function that fits the data, as a wrong function will lead to wrong results [6], [10]. The last chapter is about Bayesian methods in Neural Networks to trace problems of computer vision.

IV. Bayesian Neural Networks

Since the Deep Learning breakthrough in 2012, Deep neural networks (DNN) are mainstream and applied to all kinds of different problems. DNNs without Bayesian components give us no measure of their uncertainty when they predict something. We can think of many areas where not knowing the uncertainty of a model is problematic when critical decisions have to be made like the prediction of tumors. Another big problem is adversarial attacks on computer vision applications that add some noise to images to make the models predict other things. [13].
Bayesian neural networks (BNN) on the other hand give a measure of uncertainty, allow parameter pruning, are more robust to over-fitting and furthermore deliver solid results when learning from small data sets. The main difference of BNNs is instead of estimating points as weights of the model they infer distributions over the weights and use them (See Fig. 4). We start with a small introduction to neural networks and backpropagation, followed by the main part on Bayesian Inference in neural networks.

A. *Neural Networks*

The basic purpose behind neural networks (NN) is to approximate complex functions to find high-level features in raw data. NNs have 3 kinds of layers: one input, one output and a certain number of hidden layers(Fig. 3). You feed data to the input layer, the hidden layers process the data and propagate them to the output layer that delivers a prediction. This prediction then is used to calculate the prediction error, which is used to update the weights of the model through backpropagation. As an intuition: the hidden layers after the input layer learn small features like textures or edges, while the ones at the end put those patterns together to learn much more complex features. For example, in face recognition, the first layers detect some small parts of the face like edges of the nose while the hidden layers at the end combine those to an image [10].

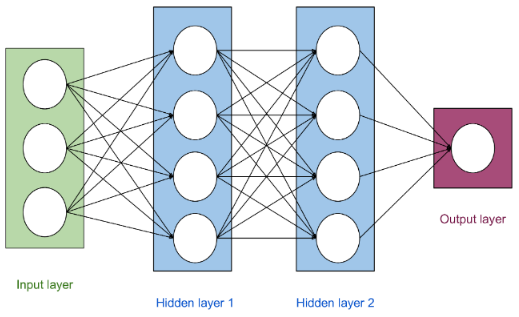

Fig. 3. basic DNN with one input, one output and two hidden layers [14]

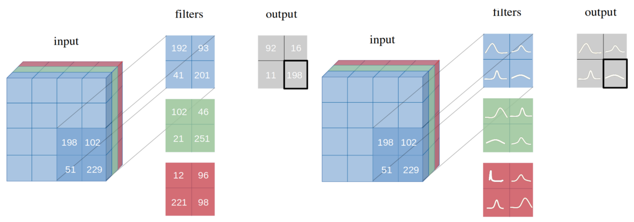

Fig. 4. point-estimates as weights on the left-hand side and inferred distributions as weights on the right-hand side [15].

B. *Backpropagation*

The most common way of training NNs is using the back-propagation algorithm. It comprises two phases. In the first phase, we propagate our inputs X forward through the network given the initial weights W and compute the prediction error through a loss function.
In the second phase, we compute the gradient of the loss function with respect to the weights and propagate that loss backward to update our weights. To minimize that loss function, gradient descent methods are used as in MLE and MAP estimates. DNNs use a frequentist approach, therefore the regression problem could also be stated as an NN problem with 1 hidden layer [1]. Through using the frequentist approach like in MLE and MAP the predictions of non Bayesian NNs have no measure of uncertainty.

C. *Bayesian Neural Networks*

There are a couple of ways of incorporating the uncertainty of predictions into NNs. This paper shows a special form of backpropagation: the Bayes by Backprop (BBBP) algorithm [16]. As calculating the true posterior is very expensive, BBBP utilizes variational inference, a concept from information theory, to approximate a posterior distribution. The idea is to find the parameters θ of a distribution $q_\theta(w|X)$ that is close to the original distribution $p(w|X)$. To achieve this we use the Kullback-Leibler-divergence ($\mathcal{KL}$) which measures the similarity of probability distributions. By minimizing

$\mathcal{KL}[q(x)||p(x)]$ (Fig. 5.) we approximate a distribution q out of a family of easy to compute distributions Q that is closest to the true distribution $p(x)$. In this paper our family Q will consist of GD with mean μ and the diagonal co-variance matrix Σ. By minimizing $\mathcal{KL}$ we get

$$\theta_{opt} = argmin_\theta \, \mathcal{KL}[q_\theta(w|X)||p(w|X)]$$

where $\mathcal{KL}$ is defined as

$$\mathcal{KL}[q_\theta(w|X)||p(w|X)] = \int q(w|X) log \frac{q(w|X)}{p(w|X)} dw.$$

To solve this problem, we use a stochastic variational method that is beyond the scope of this paper. Applying this method gives us the loss function

$$\mathcal{F}(X, \theta) \approx \sum_{i=1}^{n} \log q_\theta(w_i|X) - \log p(w_i) - \log p(X|w_i)$$

with the weight samples w_i drawn from the variational distribution $q_\theta(w, X)$ [14], [16]. This step makes computation possible but slow.

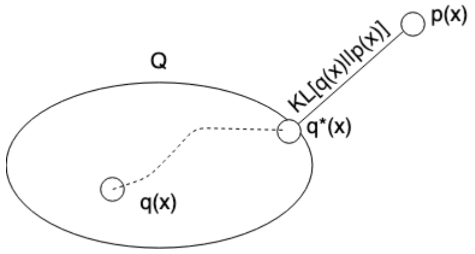

Fig. 5. Process of minimizing $\mathcal{KL}[q(x)||p(x)]$, where $q(x)$ denotes the initial starting point, $p(x)$ the distribution we want to approximate, Q the family of distributions and $q_*(x)$ the distribution which minimizes the KL divergence [4].

This chapter showed what BNNs are, why we use them and how Bayesian backpropagation works. Important to note is that BNNs main applications are the critical ones that require a measure of uncertainty, like autonomous driving. Other applications are problems that only have a small data-set available since BNNs don't need as much data as other DNNs. As a consequence of cheaper computational costs, frequentist DNNs are chosen whenever the drawbacks of using them are smaller than the gains of using a BNNs.

V. Summary and Outlook

This paper showed how Bayesian methods are applied to machine learning. We saw how to do basic LR, how to incorporate prior knowledge into LR models and how to do a full Bayesian linear regression. Then we looked at the non-parametric model of GP's and how to inference with them. At the end we showed how to use Bayesian methods in Neural Networks.

To conclude, the advantages of Bayesian methods are the encoding of prior beliefs into a model and the uncertainty measure that they add. Besides that Bayesian methods add robustness to over-fitting to the model. The main disadvantage is that those models scale bad and become expensive to intractable to compute when the number of parameters increases. Therefore, Bayesian models are primarily used for problems with a small number of parameters or when an uncertainty measure is necessary.

Frequentist methods on the other hand scale comparatively well with the downsides of estimating point-wise only and the problem of over-fitting. If one has to deal with huge data sets where uncertainty is not important, he should choose frequentist ML methods and otherwise when computationally possible he should choose Bayesian ML methods.

It is difficult to make predictions regarding the future of Bayesian ML methods. As computing power increases year by year and better algorithms are developed Bayesian methods become more interesting to apply to big problems over time. On the other hand the data sets become bigger and bigger and so one has to see if the computing power raises faster then the size of data sets.

References

[1] M. J. Beal, "Variational algorithms for approximate bayesian inference," Ph.D. dissertation, Gatsby Computational Neuroscience Unit, University College London, 2003. [Online]. Available: http://www.cse.buffalo.edu/faculty/mbeal/thesis/index.html

[2] M. J. Bayarri and J. O. Berger, "The interplay of bayesian and frequentist analysis," *Statistical Science*, vol. 19, no. 1, pp. 58–80, 2004. [Online]. Available: http://www.jstor.org/stable/4144373

[3] Z. Z., "Bayesian machine learning," http://fastml.com/bayesian-machine-learning/, [Online; 28. May 2019].

[4] D. Polykovskiy, "Bayesian approach to statistics," https://www.coursera.org/learn/bayesian-methods-in-machine-learning/lecture/wTqJf/bayesian-approach-to-statistics, [Online; 28. May 2019].

[5] A. Kristiadi, "Linear regression: A bayesian point of view," https://wiseodd.github.io/techblog/2017/01/05/bayesian-regression/.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[7] , *SAS/STAT 9.2 User's Guide, Chapter 13 - Introduction to Survival Analysis Procedures*. Cary: SAS Institute.

[8] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 116–. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015332

[9] D. Barber, *Bayesian Reasoning and Machine Learning -*. Cambridge: Cambridge University Press, 2012.

[10] K. P. Murphy, *Machine Learning - A Probabilistic Perspective*. Cambridge: MIT Press, 2012.

[11] , *Elementare Wahrscheinlichkeitstheorie und stochastische Prozesse -*. Berlin Heidelberg New York: Springer-Verlag, 2013.

[12] K. Bailey, "Bayesian processes," https://katbailey.github.io/post/gaussian-processes-for-dummies/, note =.

[13] R. Romijnders, "Bayesian deep learning with 10 percent of the weights - rob romijnders," https://www.youtube.com/watch?v=Z7VN7oRA6TY, [Online; 30. May 2019].

[14] F. Laumann, K. Shridhar, and A. L. Maurin, "Bayesian convolutional neural networks," *CoRR*, vol. abs/1806.05978, 2018. [Online]. Available: http://arxiv.org/abs/1806.05978

[15] ——, "Bayesian convolutional neural networks," *CoRR*, vol. abs/1806.05978, 2018. [Online]. Available: http://arxiv.org/abs/1806.05978

[16] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," *arXiv e-prints*, p. arXiv:1505.05424, May 2015.