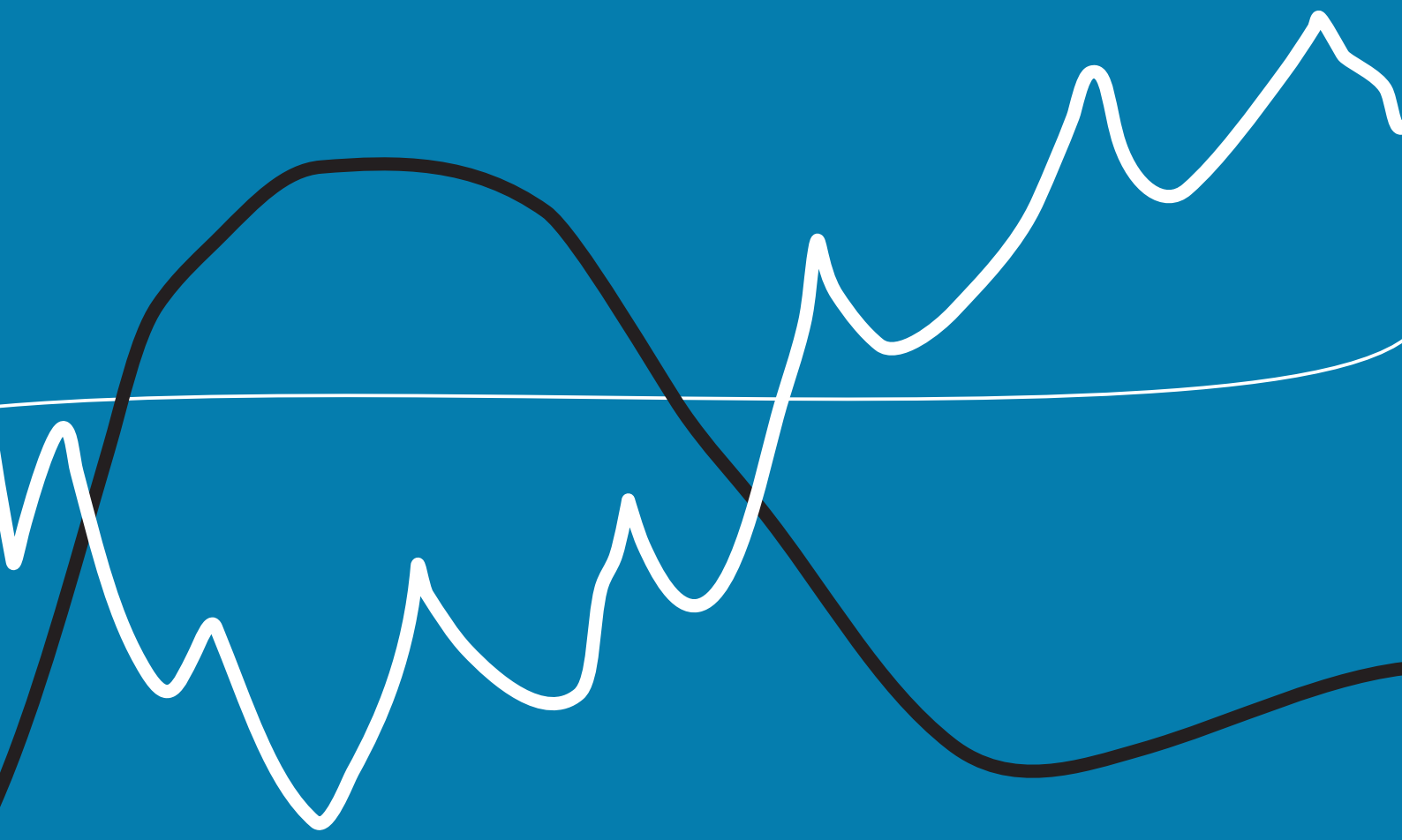
DTU Compute
Section of Cognitive Systems

Investigation of Deep Probibalistic Surrogate-models in Bayesian Optimization

Master Thesis



Investigation of Deep Probibalistic Surrogate-models in Bayesian Optimization

Master Thesis
June, 2022

By
Simon Saugmandsgaard Kruse

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Section of Cognitive Systems, Building 321, 2800 Kgs. Lyngby Denmark
www.github.com/SimonKruse0/master-thesis
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Contents

1 Introduction

Optimization plays an important part in our everyday life, science development, and product design. What transportation should you choose to get fast from A to B? What songs should land on your playlist? and what is the optimal bridge construction? In general, optimization is the art of choosing the best decision. Often we can quantify how good a decision is: A bus takes 20 min, a car takes 15 min to go from A to B, How would you rate the song? This bridge construction costs 10 million. If it is possible to come up with a quantification of how good a decision is in terms of a real number, then we can define is a *mathematical optimization problem*:

$$\min_{x \in \mathcal{X}} f(x)$$

where the functional $f : \mathcal{X} \to \mathbb{R}$ is called the objective function and $\mathcal{X}$ is the set of possible decisions (or decisions you consider). The optimization problem is now in the domain of cold numbers and here many algorithms have been developed to find the minimum of the function $f(\cdot)$. Note that we throughout this thesis refer to optimization as minimizing a cost function - this is equivalent to maximizing the negated function $\max_{x \in \mathcal{X}} f(x) = \min_{x \in \mathcal{X}} -f(x)$.

Evaluation of the objective function can be cheap e.g. if it just requires summing and multiplying numbers or highly expensive if it involves human rating or large simulation and physical experiments. Bayesian optimization is a preferred <ref> framework for optimization of the expensive objective functions. And is also referred to as *sample efficient* optimization.

Bayesian optimization is a probabilistic surrogate-based optimization methodology: Assuming some samples from a highly expensive objective a cheap (surrogate) function is used to fit the samples. The next sample is found by minimizing the surrogate and the process is repeated. Bayesian optimization seeks to enhance this procedure with probability theory, where the surrogate function becomes a probabilistic regression model. The most common choice is a Gaussian Process, as it encapsulates the uncertainty very well, but also because its inference procedure (computing answers to probability queries like $p(y|x)$) is exact.

Even though GP has proven good for many cases, there will be problems where the assumptions do not hold. The assumptions of a GP are essential that the objective function can be described as a GP. The nature of a GP is highly dependent on the choice of its kernel and the parameters chosen in that kernel. Another reason for the popularity of the GP is the closed-form inference and giving a closed-form of the expected improvement acquisition function.

$$p(y|x, \mathcal{D}) = GP(f(x)|\mu(x), \kappa(x, x)) \implies x_{next} = \arg\max_x \sigma_x[u\Phi(u) + \phi(u)]$$

1.1 No free lunch and surrogate reseach

<No free lunch theorem for optimization>, establish some questions on why there is no best method. And a burning research problem is if it is possible to use a different surrogate model than the GP. And whether or not they in general performs better? What assumptions underlie the problem. The assumption for GP regression is that the objective function follows a Gaussian process, whereas we assume it follows a random forest or neural network. These are assumptions that yields, consideration: Is this a good assumption? Bayesian NNs are very expressive.

surrogates:

- GP

- BNN
- Random forest
- SPN
- Kernel estimator

Within Bayesian surrogate model, there has been research on using random forest or Bayesian neural networks. The PhD thesis ”Sample-efficient Optimization Using Neural Networks” from 2020 [1] showcases empirically that using Bayesian neural networks as surrogate models performed better, or at least comparable to GPs on a wide number of problems. The performance difference was more clear for high-dimensional problems.

While most alternative method are investigated due to the expensive inference time for GPs, this thesis investigates alternative surrogates as a method of

Another regression model not seen much in the literature is Gaussian mixture regression, which is modeling the joint distribution between x and y, and using the conditional distribution $p(y|x)$ as the regression model, which is provided exactly.

Research problem/question:

- Which type of optimization problem is a GP not a good choice?

Research aims: As a new model Sum product networks has become a promissing method. Shown to be the most general probabilistic model that allows for exact inference! <ref>. It has relation to neural networks (but with the limit of not being too deep) and Bayesian networks, and provides a framework for learning and inference in deep probabilistic models. As nice as it sounds the project will investigate what SPNs are and how it can be used as a surrogate models.

As we will see later SPNs are just exponentially large mixture models, and therefore it is interesting to investigate Kernel estimators.

state the hypotheses

- Will an SPN work as a surrogate model?
- We can find problems where GPs fails to establish good predictions.
- BNN might work as well as GPs

1.2 Old intro

Optimization plays an important part in our everyday life, science development, and product design. What different transportation should you choose to get fast from A to B, what songs should land on your playlist, and what is the optimal bridge construction. Mathematical optimization problems are all problems in the form,

$$\min_{x \in \mathcal{X}} f(x)$$

where $f : \mathcal{X} \to \mathcal{R}$ is a functional. I.e if it is possible to set up an objective function. e.g. what is the cost of the bridge given a specific design, or how pleasant you think some music, and some constraints such that you keep in the domain of interest. Evaluation of the objective function can be cheap e.g. if it just requires summing and multiplying numbers or highly expensive if it involves human rating or large simulation and physical experiments. Bayesian optimization is a preferred <ref> framework for optimization of the expensive objective functions. And is also referred to as *sample efficient* optimization.

Bayesian optimization is a probabilistic surrogate-based optimization: Assuming some samples from a highly expensive objective a cheap (surrogate) function is used to fit the samples. The next sample is found by minimizing the surrogate and the process is repeated. Bayesian optimization seeks to enhance this procedure with probability theory, where the surrogate function becomes a probabilistic regression model. The most common choice is a Gaussian Process, as it encapsulates the uncertainty very well, but also because its inference procedure (computing answers to probability queries like $p(y|x)$) is exact.

Even though GP has proven good for many cases, there will be problems where the assumptions do not hold. The assumptions of a GP are essential that the objective function can be described as a GP. The nature of a GP is highly dependent on the choice of its kernel and the parameters chosen in that kernel. Another reason for the popularity of the GP is the closed-form inference and giving a closed-form of the expected improvement aqurisition function.

The PhD thesis "Sample-efficient Optimization Using Neural Networks" from 2020 [1] showcases empirically that using Bayesian neural networks as surrogate models performed better, or at least comparable to GPs on a wide number of problems. The performance difference was more clear for high-dimensional problems.

This master thesis project will investigate surrogate models alternative to Gaussian processes in Bayesian optimization. Firstly by examining what types of problems a GP surrogate is not a good choice for and where Bayesian neural nets (BNN) surrogates can have an advantage (inspiration found in this 2020 thesis [1]). Secondly by looking at sum-product networks (SPN) as novel surrogate models. An SPN is - similarly to a BNN - a deep probabilistic model and still expressive but with tractable inference, which potentially could lead to advantages over BNNs.

<Make a figure of how the parts are all connected>

1.3 notation

Throughout this thesis we will be using Bayesian notation, i.e. $p(x) := P(X = x)$ is the probability density function of the random variable X evaluated in x. and $p(y|x) := P(Y = y|X = x)$ or $p(y|x) := P(Y|X = x)$.

And writing $p(y^2|x)$ means $P(Y^2 = y^2|X = x)$ and **not** $P(Y = y^2|X = x)$

1.4 related work

This thesis is <BAHAMIANN>

<DNGO> Focus on the inference time of GP scales cubic, which is not appropriate for parallel BayesOpt. Experiments on 6dim Hartmann function.

<Arayns paper.> Conclusions..!?

Already developed alternative surrogate models has been found in the litterateur. The last presented surrogate model, SPN, is to our knowledge not in any published work. More models might still be added to the list.

DNGO
Deep Networks for Global Optimization (DNGO) is presented in the paper 2015 paper "Scalable Bayesian Optimization Using Deep Neural Networks"[2]. The surrogate model is a neural network, where only the last layer is probabilistic, this leads to Bayesian regression and very fast inference.

BOHAMIANN
Bayesian **O**ptimization with **Hami**ltonian Monte Carlo **A**rtificial **N**eural **N**etworks (BOHAMIANN) is presented in the 2016 paper "Bayesian Optimization with Robust Bayesian Neural Networks"[3]. This is a fully Bayesian Neural Network trained using adaptive Hamiltonian MCMC.

2 Optimization methodology

Given a cost/objective function $f : \mathcal{X} \to \mathbb{R}$, where the domain $\mathcal{X}$ could be a subset of $\mathbb{R}^n$, optimization is a methodology which seeks to find an optimal point, x^*, and value $f^* = f(x)$, given as

$$x^* \in \arg\min_{x \in \mathcal{X}} f(x) \qquad f^* = \min_{x \in \mathcal{X}} f(x) = f(x^*). \tag{2.1}$$

Note that the above formulation is a minimization problem, which is equivalent to a maximization problem maximizing $-f(\cdot)$. Throughout this thesis, we choose to only work with a minimization problem. Solving this problem is intractable except for rare cases e.g. if f is convex and analytically directly solvable or the domain of f is very limited. Hm

Example: Direct solution method
The unconstrained linear least squares,

$$\min_{x \in \mathbb{R}^n} f(x) := ||Ax - b||_2^2$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, is a convex problem, i.e. finding x^* such that $\nabla f(x^*) = 0$ is equivalent to finding the solution to the problem. Assuming $A^T A$ is invertable, linear least squares can be solved directly by the normal equations,

$$\nabla f(x) = 2A^T Ax + 2b^T A = 0 \quad \Leftrightarrow \quad x^* = (A^T A)^{-1} A^T b$$

Most optimization problems are non-convex with multiple local minima. And even if the gradient is given analytically the solution is found among a potentially infinitely large set of stationary points ($\nabla f(x) = 0$) and boundary points - this might be tedious or impossible. When the problem is not directly solvable, mathematical optimization takes an indirect methodology: Design a sequence of experiments that reveal information of the objective function. This information can hopefully lead us to the solution of (2.1). This general way of sequentially solving is presented in the book Bayesian Optimization by Roman Garnett [4] and presented here as Algorithm 1. W

Algorithm 1 Sequencial Optimization [4]

Input: Initial dataset $\mathcal{D}$ ▷ can be empty
while Temination is not reached **do**
  $x \leftarrow$ policy($\mathcal{D}$) ▷ select next observation location
  $y \leftarrow$ observe(x) ▷ observe objective function at chosen location
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x, y)\}$ ▷ update dataset
return: $\mathcal{D}$

Given data points in the optimization landscape[1] a policy selects a point $x \in \mathcal{X}$ where we make our next observation. Policies can be deterministic or probibalitic, e.g. grid search and random search are examples of policies. The next observation provides us a y value, which combined with x is included to the available data $\mathcal{D}$. Finaly, a stopping criterion decides whether to continue or terminate.

[1] "Optimization landscape" defined as the joint set of points in the domain and the objective function evaluated in the points, i.e. $\{(x, f(x)) \in \mathcal{X} \times \mathbb{R} | x \in \mathcal{X}\}$

<example of random search is half as efficient as BO>

Example: Grid search
In grid search values along each dimention in $\mathcal{X}$ is seleced and combined with each other, which thereby defines a grid in the parameter space. All points are ordered and systematicly selected. In the content of algorhtim 1 we define the grid search policy as

$$\text{policy}_{GS}(\mathcal{D}) = x_{|\mathcal{D}|+1}$$

assuming $x_1, x_2, \ldots, x_n$ are the ordered grid points and the size of the obtained data is $|\mathcal{D}|$.

Example: Random search
In random search a uniform distribtuion is layed over the domain space $\mathcal{X}$ and a random point is selected from the distribtuion.

$$\text{policy}_{RS}(\mathcal{D}) = x, \qquad x \sim p(\mathcal{X})$$

Note that grid search and random search are policies which completely ignores the available data. This is a shame and we can do better.

Example: Gradient descent
Gradient descent is the most simple gradient-based optimization approach. The gradient of a continuous function points in the most ascending direction from the location where it is evaluated. In the minimization task, (2.1), we can iteratively use the opposite gradient direction, i.e. the most descenting directiong. This yields the policy:

$$\text{policy}_{GD}(\mathcal{D}) = x_n - \eta \nabla f(x_n)$$

where we for a brief moment modify y to be a vector, since the observation model is given as:

$$\text{observe}_{GD}(x) = [f(x), \nabla f(x)]$$

Example: Surrogate-based optimization
In surrogate-based optimization all available data is fitted by a cheap-to-evaluate approximation to the objective function - this approximation is called a *surrogate* model. Examples of surrogate models could be a neural network or a random forest. The next point is chosen as the point where the surrogate model is minimized.

$$\text{policy}_{sur}(\mathcal{D}) = \min_x \hat{f}(x)$$

where $\hat{f}(x) \approx f(x)$ for x close to the data $\mathcal{D}$. And we hope the approximation holds for x far away from the the data.

2.1 When to use Bayesian optmization

What if $f(x)$ took serval days to evaluate. What if $f(x)$ is noisy? what if discrete points? <more here>

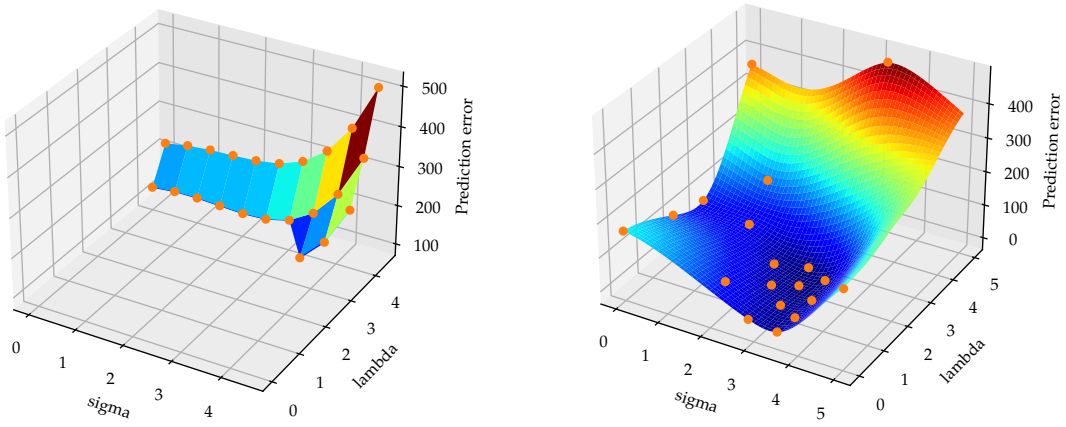


Figure 2.1: Hyper parameter tuning of a model $M(\lambda, \sigma)$, 23 evaluation in grid search vs 23 evaluations using Bayesian optimization

additionally, Bayesian Optimization allow for observation noise,

2.2 Observation model

Many optimization algorithms assume *exact* evaluations of the objective function. However, this assumption is often wrong especially for objective functions with real-life experiments, imperfect simulations, human interaction where measurement noise is a well known. The observation model is typically noisy and described as

$$y = f(x) + \epsilon$$

where ϵ is the measurement error, this is typically assumed to be Gaussian with zero mean and a variance σ^2 (which could depend on x in a heterostodatic setting) and implies a Gaussian observation model,

$$p(y|x, f(x), \sigma) = \mathcal{N}(y; f(x), \sigma^2)$$

we can extend this model to deal with noiseless observations as well, simply by setting $\sigma = 0$ and let the model colaps into a Direct delta distribution,

$$p(y|x, f(x)) = \delta(y - f(x))$$

i.e. all probability mass for y is on the value $f(x)$ giving the observation sample $y = f(x)$

3 Bayesian Optimization

Whereas traditional regression workflow is the following: From data, fit model parameters, make predictions using the parameters. The Bayesian framework allows us to skip the dependency of a single set of parameters and instead use all sets of parameters by treating the set of parameters as a random quantity, θ. What is of interest is the predictive posterior distribution,

$$p(y|x, \mathcal{D}) \tag{3.1}$$

Bayesian optimization is essentially two steps: First, a probabilistic surrogate-model is fitted to the available data $\mathcal{D}$ giving the predictive distribution (3.1). Second, the next sample point is chosen according to a so-called acquisition function, which in a sense balance out the well-known concept, exploitation and exploration. Where exploitation will be chooing the next point according to its expected improvement and exploration will be choosing the next point in a region of high uncertainty and thereby help lower the overall uncertainty. We will first look at the acquisition function used in this thesis, which is also the most commonly used: E xpected improvement.

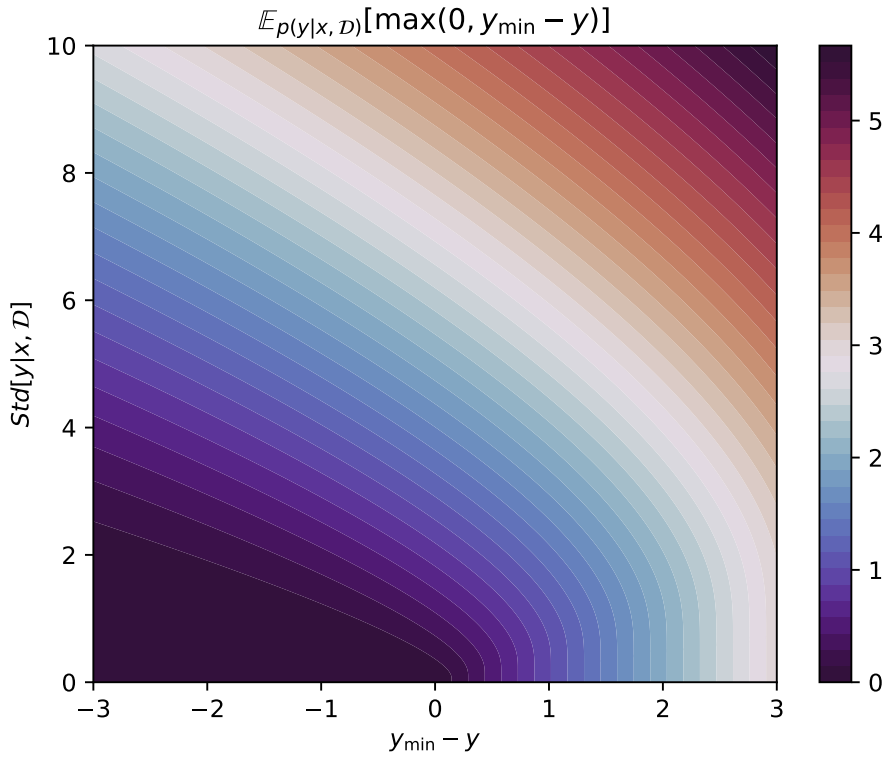
3.1 Acquisition function

3.1.1 Expected improvement

A popular choice of acquisition function is expected improvement,

$$EI(x) = \mathbb{E}_{p(y|x,\mathcal{D})}[\max(0, y_{\min} - y)]$$

this will only look at the expectation of the values y, which improves the current best value.



In the following deveration we assume the predictive distribution can be approxiamted by a normal distribution dependent on the point of interest x and the data $\mathcal{D}$ (note for the GP it is in fact not an approximation),

$$p(y|x,\mathcal{D}) \approx \mathcal{N}(y|\mu(x,\mathcal{D}),\sigma^2(x,\mathcal{D}))$$

where we will change to a less clumpsy notation $\mathcal{N}(y|\mu_x,\sigma_x^2) := \mathcal{N}(y|\mu(x,\mathcal{D}),\sigma^2(x,\mathcal{D}))$. This is completely fine since we x is fixed (and $\mathcal{D}$ is fixed) when evaluating the expected improvement in a point x. Furthermore, the density of a standard normal distribution is denoted $\phi(\cdot) := \mathcal{N}(\cdot|0,1)$, and the cumlative density function (CDF) of a standard normal distribution is denoted, $\Phi(\cdot) := \int_{-\infty}^{\cdot}\phi(\epsilon)d\epsilon$. We will now see that the normal approximation of the predictive distribution yiels closed form solution to the expected improvement function,

$$\begin{aligned}
E_{p(y|x,\mathcal{D})}[\max(0, y_{\min} - y)] &= \int \max(0, y_{\min} - y)p(y|x,\mathcal{D})dy \\
&\approx \int \max(0, y_{\min} - y)\mathcal{N}(y|\mu_x,\sigma_x^2)dy \\
&= \int_{-\infty}^{y_{\min}} (y_{\min} - y)\frac{1}{\sigma_x}\phi\left(\frac{y-\mu_x}{\sigma_x}\right)dy \\
&= \int_{-\infty}^{\frac{y_{\min}-\mu_x}{\sigma_x}} (y_{\min} - \mu_x - \sigma_x\epsilon)\frac{1}{\sigma_x}\phi\left(\epsilon\right)\sigma_x d\epsilon \\
&= \int_{-\infty}^{u} \sigma_x \cdot (u-\epsilon)\phi(\epsilon)d\epsilon \\
&= \sigma_x \cdot \left(u \cdot \int_{-\infty}^{u}\phi(\epsilon)d\epsilon + \int_{-\infty}^{u}(-\epsilon)\phi(\epsilon)d\epsilon\right) \\
&= \sigma_x[u\Phi(u) + \phi(u)]
\end{aligned}$$

where $u := \frac{y_{\min}-\mu_x}{\sigma_x}$. To understand the identity $\phi(u) = \int_{-\infty}^{u}(-\epsilon)\phi(\epsilon)d\epsilon$ used in the last equality, we first see that the antiderivative is $\phi(\epsilon) = \frac{1}{\sqrt{2\pi}}\exp(\frac{-\epsilon^2}{2})$,

$$\begin{aligned}
\frac{d}{d\epsilon}\phi(\epsilon) &= \frac{1}{\sqrt{2\pi}}\frac{d}{d\epsilon}\exp(\frac{-\epsilon^2}{2}) \\
&= \frac{1}{\sqrt{2\pi}}\exp(\frac{-\epsilon^2}{2})(-\epsilon) \\
&= -\epsilon\phi(\epsilon)
\end{aligned}$$

and evaluating the rieman integral is equivalent to evaluate the antiderivative in its boundaries, giving the solution,

$$\int_{-\infty}^{u}(-\epsilon)\phi(\epsilon)d\epsilon = [\phi(\epsilon)]_{-\infty}^{u} = \phi(u) - 0 = \phi(u)$$

We can also explicily write the expected improvement as,

$$EI(x) = (y_{\min} - \mu_x)\Phi\left(\frac{y_{\min}-\mu_x}{\sigma_x}\right) + \sigma_x\phi\left(\frac{y_{\min}-\mu_x}{\sigma_x}\right)]$$

where the first part can be interpretted as exploitation (favouring points with a large improvement $I(x) := (y_{\min} - \mu_x)$) and the second part can be seen a exploitation (favouring points with high

uncertainty.) taking the derivative with respect to $I(x) := (y_{\text{min}} - \mu_x)$ and σ_x, we see that expected improvement is is increasing if the improvement grows or if the variance σ_x grows, i.e

$$\frac{\partial EI(x)}{\partial I(x)} = \Phi\left(\frac{y_{\text{min}} - \mu_x}{\sigma_x}\right) > 0, \qquad \frac{\partial EI(x)}{\partial \sigma_x} = \phi\left(\frac{y_{\text{min}} - \mu_x}{\sigma_x}\right) > 0$$

<obs mistake in the book!!!?>

$$\begin{aligned}
\mathbb{E}_{y_*|\mathbf{x}_*, D_n}[\max(0, y_{\text{min}} - y_*)] &= ?? \\
\mathbb{E}[\min(0, y_{\text{min}} - y_*)|\mathbf{x}_*, D_n] &= \int_{-\infty}^{\infty} \min(0, y_{\text{min}} - y_*) p(y_*|\mathbf{x}_*, D_n) dy_* \\
&= \int_{-\infty}^{y_{\text{min}}} (y_{\text{min}} - y_*) p(y_*|\mathbf{x}_*, D_n) dy_* \\
&\approx \frac{1}{N} \sum_{\theta \in \Omega} [y_{\text{min}} - f_\theta(x)]
\end{aligned}$$

where $\Omega = \{\theta | f_\theta(x) < y_{\text{min}}\}$

3.1.2 Entropy search

We then discuss the knowledge gradient (Section 4.2), entropy search and predictive entropy search (Section 4.3) acquisition functions. These alternate acquisition functions are most useful in exotic problems where an assumption made by expected improvement, that the primary benefit of sampling occurs through an improvement at the point sampled, is no longer true.

The entropy search (ES) (Hennig and Schuler, 2012) acquisition function values the information we have about the location of the global maximum according to its differential entropy

ES seeks the point to evaluate that causes the largest decrease in differential entropy

(Recall from, e.g., Cover and Thomas (2012), that the differential entropy of a continuous probability distribution p(x) is R p(x) log(p(x)) dx, and that smaller differential entropy indicates less uncertainty.)

Predictive entropy search (PES) (Hern´andezLobato et al., 2014) seeks the same point, but uses a reformulation of the entropy reduction objective based on mutual information. Exact calculations of PES and ES would give equivalent acquisition functions, but exact calculation is not typically possible, and so the difference in computational techniques used to approximate the PES and ES acquisition functions creates practical differences in the sampling decisions that result from the two approaches. We first discuss ES and then PES.

Let x be the global optimum of f. The posterior distribution on f at time n induces a probability distribution for x . Indeed, if the domain A were finite, then we could represent f over its domain by a vector (f(x) : x □ A), and x would correspond to the largest element in this vector. The distribution of this vector under the time-n posterior distribution would be multivariate normal, and this multivariate normal distribution would imply the distribution of x

. When A is continuous, the same ideas apply, where x

is a random variable whose distribution is implied by the Gaussian process posterior on f

3.2 surrogate model

As mentioned in the previous sections, the first of two repeated steps in Bayesian optimization is to create a good Bayesian regression model. i.e. finding the probility of prediction for a arbitrary point x given datapoints $\mathcal{D} = \{x_1, y_1, \ldots, x_n, y_n\}$,

$$p(y|x, \mathcal{D})$$

The surrogate model of choise in Bayesian optimization is a Gaussian Process, and Bayesian Neural Network. These are discriminative models, however, another approach, which we focus on in this project, is to model y and x jointly in a so-called generative model.

3.2.1 Discriminative model as surrogate model

When talking about a probabilistic surrogate model we are always implicit talking about a discriminative model: A model of the conditional distribution of the observation, y, conditional on x, i.e.,

$$p(y|x)$$

we also refer to this as the predictive distribution. Gaussian processes and Bayesian neural networks are both discriminative models. There is no distribution over the input x, it is just given. This is indeed sufficient for a surrogate model, where we are interested in the predictive distribution.

3.2.2 Using a generative model as surrogate model

Given a generative model over x and y paramitised with θ, we are dealing with the joint distribution

$$p(x, y|\theta)$$

and we are interested in the condtional distribution of y given x,

$$p(y|x, \theta_{y|x})$$

where we have put subscript on θ in order to jump up a level of abstraction since, in fact there is just a mapping between them $\theta_{y|x} := g(\theta, y, x)$

$$\begin{aligned} p(y|x, \mathcal{D}) &= \int p(y|x, \theta_{y|x}) p(\theta_{y|x}|\mathcal{D}) d\theta_{y|x} \\ &= p(y|x, \hat{\theta}_{y|x}) \end{aligned}$$

Where the last equation holds as we assume that $p(\theta_{y|x}|\mathcal{D})$ is a delta function i.e. a point estimate with value $\hat{\theta}_{y|x}$. In the case of our Gaussian mixture model, we obtain a point estiamte from the EM algorithm for the variance $\Sigma_{y|k}$, mean value $\mu_{y|k}$ and proportion $\pi_{y|k}$ for each component $k = 1, 2, \ldots, K$

$$\hat{\theta}_{y|x} = (\hat{\Sigma}_{y|k}, \hat{\mu}_{y|k}, \hat{\pi}_{y|k})_{k=1}^{K}$$

However, we are not satisfied with the variance estimate for the regression, as it is way too small for areas with no observed data. It is therefore necessary to manipulate the variance estimate accoring to that observation. We multiply the variance obtained using expectation-maximization on the joint distribution with the inverse of the probability of the data x, and control that the scaling factor is not going wild!

$$\hat{\Sigma}_{y|k} = \Sigma_{y|k}^{GMM} \frac{1}{\max(p(x), 0.01)}$$

In a way, this is manipulation in a Bayesian spirit, as we let prior and subjective knowledge influence the variance prediction.

3.2.3 Inclusion of prior distribution

The new conditional distribution is essentially a mixture between the old conditional and the prior distribution,

$$\hat{p}(y|x) = \alpha_x p(y|x) + (1 - \alpha_x) p_{prior}(y)$$

where we choose $\alpha_x := \frac{S(x)}{S(x)+1}$. A convex combination of distributions is a distribution, since

$$\begin{aligned}
\int_y \hat{p}(y|x)dy &= \int_y \left[\alpha_x p(y|x) + (1 - \alpha_x) p_{prior}(y)\right] dy \\
&= \alpha_x \int_y p(y|x)dy + (1 - \alpha_x) \int_y p_{prior}(y)dy \\
&= \alpha_x + (1 - \alpha_x) = 1
\end{aligned}$$

3.2.4 Inclusion of prior distribution

Since the prior predictive distribution and the predictive distribution both are assumed to be normal distributions, it is possible to simply let the parameters of the manipulated predictive distributions be a convex combination of the parameters. This will make the manipulation be more natural, and hopefully less coruptive.

$$\begin{aligned}
\hat{p}(y|x) &= \mathcal{N}(y|\hat{\mu}, \hat{\sigma}^2) \\
\hat{\mu} &= \alpha \cdot \mu_{y|x} + (1 - \alpha) \cdot \mu_{prior} \\
\hat{\sigma}^2 &= \alpha \cdot \sigma^2_{y|x} + (1 - \alpha) \cdot \sigma^2_{prior}
\end{aligned}$$

3.2.5 Inclusion of prior distribution

The conditional distribution $p(y|x)$ obtained from using a Gaussian mixture model is without a prior, and the predictive distribution yields a way too small uncertainty estimate in regions where there are no observations. This is because the range of the Gaussian distribution continues throughout the entire space and even though the joint distribution is very small in the region, the conditional distribution is normalized using the (very small) $p(x)$. We, therefore, want to introduce a prior, which can rule over the conditional in regions where the probability is too small. The first approach is simply to define a new conditional distribution,

$$\hat{p}(y|x) \propto S(x) \cdot p(y|x) + p_{prior}(y) \tag{3.2}$$

where the scaling function can be interpreted as the probability of the input x is in the region ball $x \in B_{\frac{1}{2}\Delta}(x)$, and the probability density is not just $p(x)$ but the probability of the average mixture component, which is $K \cdot p(x)$, assuming the mixture is dominated be a single component at the position x. We define the scaling as

$$S(x) := K \cdot p(x) \cdot \Delta$$

Next we need to normalize the manipulated predictive distribution in order to make it a proper probability density, i.e., making sure it integrates to 1,

$$\int S(x) \cdot p(y|x) + p_{prior}(y)dy = S(x) + 1$$

so the we can easily update (3.2) with an equality sign by defining the manipulated predive distribution as,

$$\hat{p}(y|x) = \frac{S(x) \cdot p(y|x) + p_{prior}(y)}{S(x) + 1}.$$

In figure 3.1 the manipulated predictive distribution is illustrated under the influenced of different scalings. Note that if $S(x) = 0$ the manipulated distribution becomes the prior, while in the limit $S(x) \to \infty$ the manipulated distribution become the original predictive distribtuion, $\hat{p}(y|x) = p(y|x)$.

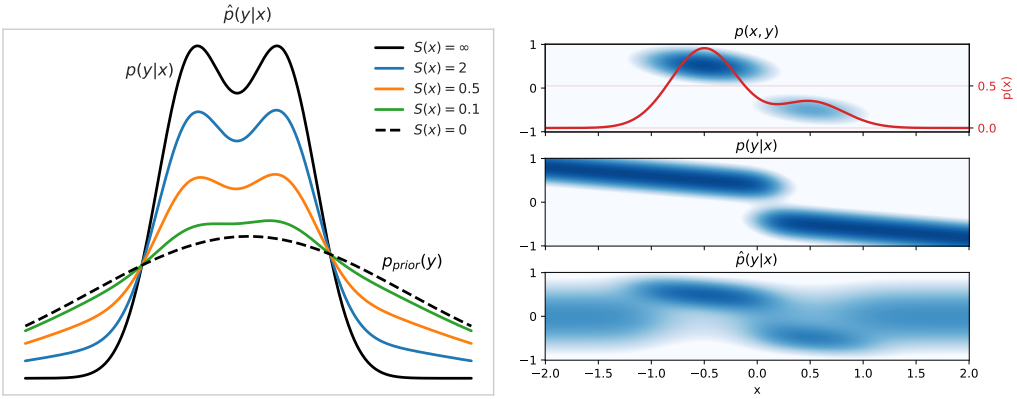


Figure 3.1: Left: Illustration of how the preditive distribution is manipulated according the the scaling function $S(x) := p(x) \cdot N$. Right: Illustration of why it makes sense to manipulate the predictive distribution $p(y|x)$, if there is a small amount of input data at a region, then the predictive distribution should collapse into the uncertain prior

Manipulated predictive mean
What is the mean prediction of the manipulated predictive distribution?

$$\begin{aligned} E_{p_{bayes}(y|x)}[y] &= \int y \cdot \frac{f(n, p(x)) \cdot p(y|x) + \lambda p_{prior}(y)}{Z} dy \\ &= \left(p(x) \cdot N \cdot E_{p(y|x)}[y] + \lambda \cdot E_{p_{prior}(y)}[y] \right) \frac{1}{Z} \\ &= \frac{p(x) \cdot N \cdot E_{p(y|x)}[y]}{p(x) \cdot N + \lambda} \end{aligned}$$

Manipulated predictive variance
What is the variance of the manipulated predictive distribution We first calculate the second moment,

$$\begin{aligned} E_{p_{bayes}(y|x)}[y^2] &= \int y^2 \cdot \frac{p(x) \cdot N \cdot p(y|x) + \lambda p_{prior}(y)}{Z} dy \\ &= (p(x) \cdot N \cdot E_{p(y|x)}[y^2] + \lambda E_{p_{prior}(y)}[y^2]) \frac{1}{Z} \\ &= \frac{p(x) \cdot N \cdot (Var_{p(y|x)}[y] + E_{p(y|x)}[y]^2) + \lambda Var_{p_{prior}(y)}[y]}{p(x) \cdot N + \lambda} \end{aligned}$$

So the variance is calculated as,

$$V_{p_{bayes}(y|x)}[y] = E_{p_{bayes}(y|x)}[y^2] - E_{p(x,y)}[y]^2$$

implementation
It is not necessary to calculate the conditional distribution, since, using Bayes rule

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

so gives

$$p_{bayes}(y|x) = \frac{p(x) \cdot N \cdot p(y|x) + \lambda p_{prior}(y)}{Z} = \frac{N \cdot p(x,y) + \lambda p_{prior}(y)}{Z}$$

$$E_{p(x,y)}[y] = \int_y \int_x y p(x,y) dx dy = \int_y y p(y) dy = E_{p(y)}[y]$$

4 Gaussian process surrogate

4.1 Gaussian Process Regression

We now show how the preditive distribution is calculated exact for Gaussian Processes, i.e.

$$p(y|x,\mathcal{D}) = \int \mathcal{N}(y|f(x),\sigma^2)p(f(x)|\mathcal{D})df(x) \tag{4.1}$$

we will soon see that $p(f(x)|\mathcal{D}) = \mathcal{N}(f(x)|..,...)$ and and thereby that we have a marginal Gaussian distribution for $f(x)$ and a conditional Gaussian distribution of y given $f(x)$, giving us the marginalized distribtuion, $p(y|x,\mathcal{D})$, using formulars (4.2).

Background: Trick with normal distributions [from Bishops book?]

Given a marginal Gaussian distribution of x and a conditional Gaussian distribution of y given x of the form,

$$p(x) = \mathcal{N}(x|\mu,\Lambda^{-1})$$
$$p(y|x) = \mathcal{N}(x|Ax+b,L^{-1})$$

then the marginal distribution of y and the conditional distribution of x given y have the form,

$$p(y) = \mathcal{N}(y|A\mu+b,L^{-1}+A\Lambda^{-1}A^T) \tag{4.2}$$
$$p(x|y) = \mathcal{N}(x|\Gamma\mu+\Gamma[A^TL(y-b)],\Gamma) \tag{4.3}$$
$$\Gamma := (\Lambda+A^TLA)^{-1} \tag{4.4}$$

Posterior function

Recall we assume $\mathbf{f} = (f(\mathbf{x}_1),\dots,f(\mathbf{x}_n))$ is the parameters in our model, therefore we call $p(\mathbf{f}|\mathcal{D})$ the posterior distribution. However, what is of real interest is the function values on unobserved locations, thereby we extend $\mathbf{f}$ to be a function, i.e. an infinitely dimentional vector. We call this quantaty *the posterior function*

$$p(f(\cdot)|\mathcal{D}) = \int p(f(\cdot)|\mathbf{x},\mathbf{f})p(\mathbf{f}|\mathcal{D})d\mathbf{f}. \tag{4.5}$$

Prior we assume that the function takes values accoring to

$$p(\mathbf{f}|\mathbf{x}) = \mathcal{N}(\mathbf{f}|\mathbf{0},c(\mathbf{x},\mathbf{x}))$$

where the covariance is defined at kernel evaluation for each pair of $\mathbf{x}$, where $c(\cdot,\cdot)$ is a covariance function, chosen to be the Matérn covariance function,

$$c(\mathbf{x},\mathbf{x}) = \begin{bmatrix} c(x_1,x_1) & \dots & c(x_1,x_n) \\ \vdots & \ddots & \\ c(x_n,x_1) & \dots & c(x_n,x_n) \end{bmatrix} \qquad c(x,y) := Matern(x,y)...$$

Appendix <...> tries to give the intuition why this makes sense.

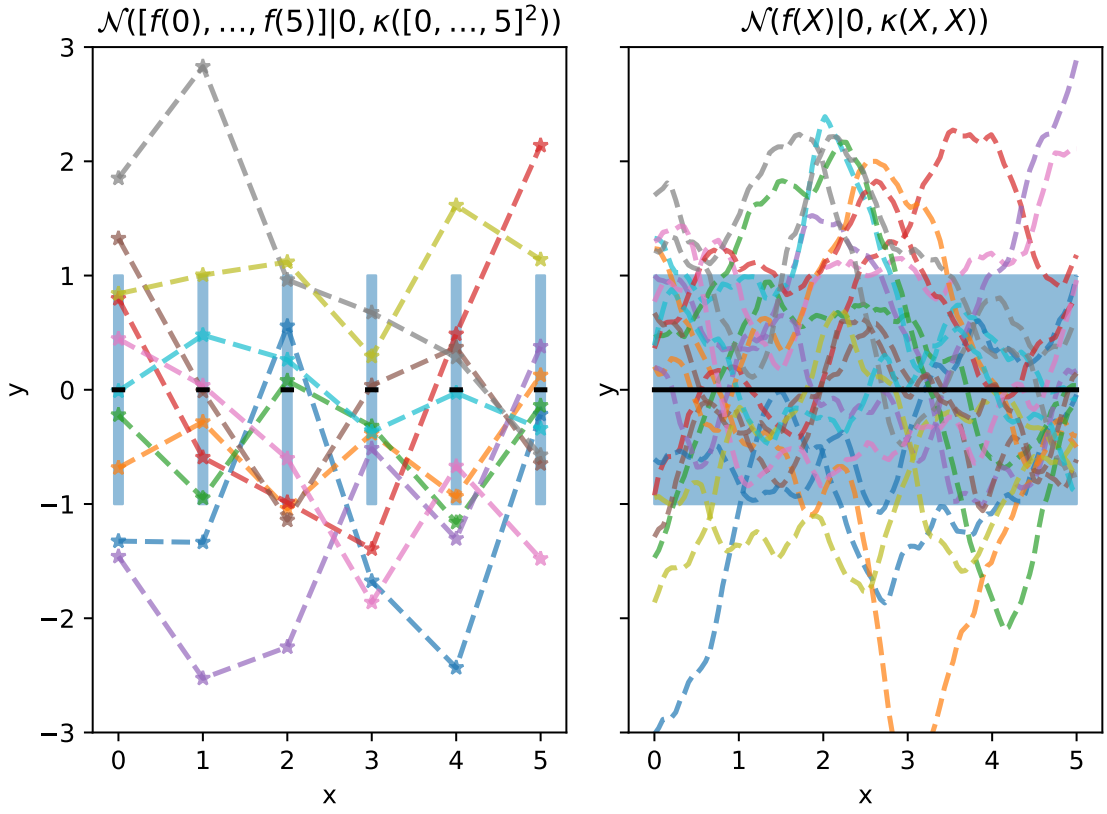


Figure 4.1: Realisations> Illustartaion of GP is just samples from a multivariate normal distribution. In the limit the we have a infinite-variate normal distribution, which we call a GP.

We now calculate the first term in the integral (4.5), $p(f(\cdot)|\mathbf{x}, \mathbf{f})$ using that we have the joint prior distribution,

$$p(f(\cdot), \mathbf{f}|\mathbf{x}) = \mathcal{N}\left(\begin{bmatrix} f(\cdot) \\ \mathbf{f} \end{bmatrix} \middle| \begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} c(\cdot,\cdot) & c(\cdot,\mathbf{x}) \\ c(\mathbf{x},\cdot) & c(\mathbf{x},\mathbf{x}) \end{bmatrix}\right) \tag{4.6}$$

And the conditonal of a joint Gaussian is given using <ref>

$$p(f(\cdot)|\mathbf{x}, \mathbf{f}) = \mathcal{N}(f(\cdot)|c(\cdot,\cdot)^{-1}c(\cdot,\mathbf{x})\mathbf{f}, c(\cdot,\cdot)^{-1})$$

Next we calculate the last term in the integral (4.5), $p(\mathbf{f}|\mathcal{D})$, i.e. the posterior distribution. Assuming iid data, i.e. $p(y_1, \ldots, y_n|x_1, \ldots, x_n, \mathbf{f}) = \prod_{i=1}^{n} p(y_i|x_i, \mathbf{f}_i)$ and that the likelihood is Gaussian with mean $\mathbf{f}$ and variance $\sigma^2 I_n$.

$$\begin{aligned} p(\mathbf{f}|\mathcal{D}) &\propto p(\mathbf{f}|x) \prod_{i=1}^{n} p(y_i|x_i, \mathbf{f}_i) \\ &= \mathcal{N}(\mathbf{f}|\mathbf{0}, c(\mathbf{x},\mathbf{x})) \prod_{i=1}^{n} \mathcal{N}(y|\mathbf{f}_i, \sigma^2) \\ &= \mathcal{N}(\mathbf{f}|\mathbf{0}, c(\mathbf{x},\mathbf{x}))\mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 I_n) \end{aligned}$$

now from <ref> we have that the posterior is the following Gaussian:

$$p(\mathbf{f}|\mathcal{D}) = \mathcal{N}(\mathbf{f}|M^{-1}\sigma^{-2}\mathbf{y}, M^{-1}) \quad M := c(\mathbf{x},\mathbf{x})^{-1} + \sigma^{-2} I_n$$

Now we found that both term in the integral (4.5), and they are related such that it is possible to use (4.2) for arriving at (we define $A := c(\cdot,\cdot)^{-1}c(\cdot,\mathbf{x})$),

$$p(f(\cdot)|\mathcal{D}) = \mathcal{N}(f(\cdot)|AM^{-1}\sigma^{-2}\mathbf{y}, c(\cdot,\cdot)^{-1} + AM^{-1}A^T)$$

Finally we found that both terms in the integral (4.1) also is related in a simlar way, and we use (4.2), again to arrive at the predictive distribtuion,

$$p(y_*|x_*,\mathcal{D}) = \mathcal{N}(y_*|AM^{-1}\sigma^{-2}\mathbf{y}, c(x_*,x_*)^{-1} + AM^{-1}A^T + \sigma^2)$$

Some questions about a naive approach..!

Learning - Emperical bayes inference

Anther inference which is done is then optimizing the hyper parameters using emperical bayes i.e. the variance and length scale for the kernel. Here we optimize the marginalized likelihood function

$$p(y_1,\ldots,y_n|x_1,\ldots,x_n,\theta) = -\frac{1}{2}[(y-\mu)^T(\Sigma+N)^{-1}(y-\mu) + \log|\Sigma+N| + n\log 2\pi]$$

<and how to get to there?>

Model assessment becomes trivial in light of the model posterior if we simply establish preferences over models according to their posterior probability. When using the uniform model prior (4.6) the model posterior is proportional to the marginal likelihood alone, which can be then used directly for model assessment. ??! Youtube - good video! Go through that!

5 Mixture regression surrogate

5.1 Gaussian mixture regression

Taking a convex combination of a set of multivariate Gaussian distributions is a Gaussian mixture model

$$p(z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(z|\mu_k, \Sigma_k)$$

Defining $z := (x, y)$ we can model our data, as a generative model $p(x, y)$, now, since the conditonal of a Gaussain mixture again is Gaussain mixture - i.e. closed form expression, we can exactly calculate $p(y|x) = GMM_{y|x}$

Assuming iid data the likelihood is given as

$$p(\mathcal{D}|\mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \mathcal{N}(z_i|\mu_k, \Sigma_k)$$

And the log likelihood,

$$\log p(\mathcal{D}|..) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(z_i|\mu_k, \Sigma_k)$$

Dealing with wrong variance

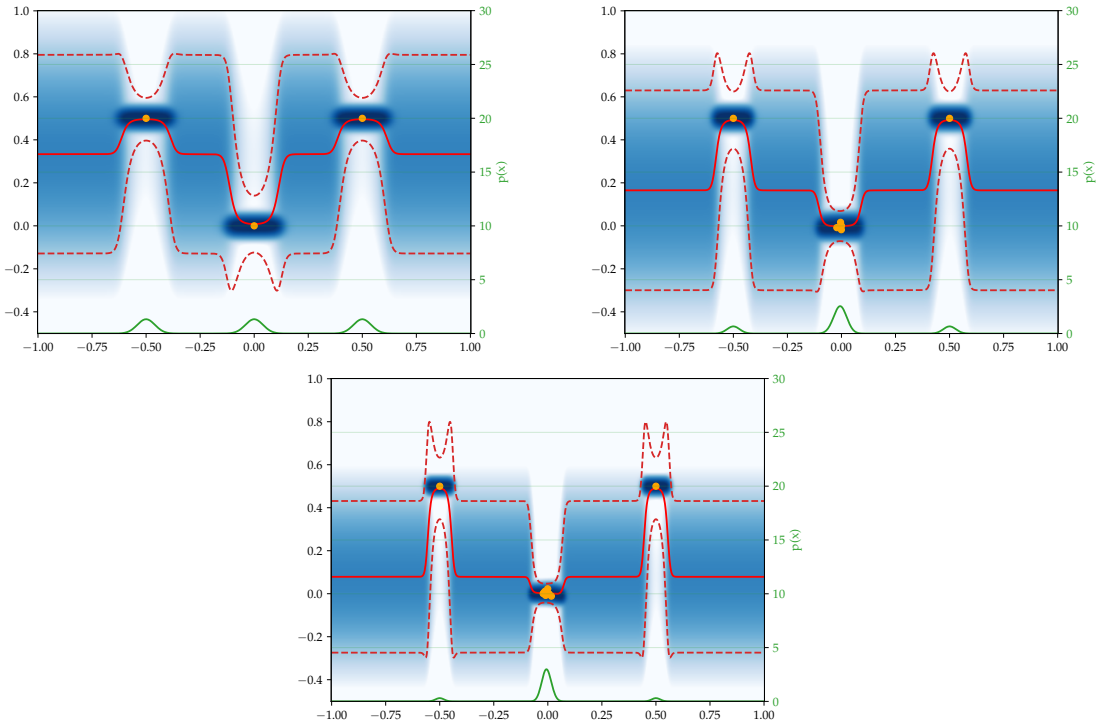


Figure 5.1: EI improvement might in this case be stucked at the point 0, since the conditional distribution is normalized by $p(x)$ and therefore not influenced by the amount of data

<variance manipulation> <Expected improvement manipulation>

5.2 Mixture regression in a Bayesian setting

As seen in examples. The uncertainty of conditional distribution is way too certain in areas with no data points, therefore we need to enhance the model with some bayesian flavour.

$$p(y|x, \mathcal{D}) = p(y|x, Z)p(Z|x)$$

Expetation-maximization algorithm

A way to find local maxima in the likehood function is using the EM algorithm.

If we define a latent/hiddem random variabel $Z_i \in \{1, \dots, K\}$ for each data point, then the likelihood function becomes,

$$L(\theta|\mathcal{D}, Z) = \prod_{i=1}^{n} \sum_{k=1}^{K} 1(Z_i = k)\pi_k \mathcal{N}(z_i|\mu_k, \Sigma_k)$$

Now the expectation wrt. the current value $p(Z|\mathcal{D}, \theta^k)$ is given as

$$Q(\theta|\theta^k) = \mathcal{E}_{p(Z|\mathcal{D},\theta^k)} = L(\theta|\mathcal{D}, Z)$$

And then update the next parameter estimate with

$$\theta^{k+1} = \arg\max_{\theta} Q(\theta|\theta^k)$$

This is repeated untill convergence.

5.3 Sum product networks

We will for a large extend just see SPN as a large mixture model. This is a totally valid observation.

Mixture model

Definition 1 *A sub-network $\bar{S}_z$ of S is an SPN, which includes the root S and then includes nodes according to the following recursive scheme:*

Collection of sub-network S_z of S

function Process(node i, S_z)
  if $i \in \mathcal{L}eaf(S)$ **then**
    return:
  if $i \in \mathcal{S}um(S)$ **then**
    $S_z = S_z.add(j \in ch(i))$ ▷ include one child of node i
    return: Process(j, S_z)
  if $i \in \mathcal{P}rod(S)$ **then**
    $S_z = S_z \cup \{j | j \in ch(i)\}$ ▷ include all childen of node i
    for $j \in ch(i)$ **do**
      return: Process(j, S_z)
  return: S_z
$S_z =$ Process(root,Ø)

So we see that at each sum node the number of different sub-networks multiplies with the number of children for that sum node. And thereby, the total number of sub-networks is

$$Z = \prod_{i \in \mathcal{S}um(S)} |ch(i)|$$

i.e. an exponential large amount of sub-networks. This is the amount of mixture components implicitly defined in an SPN. Denote the set of edges in the sub-network $\mathcal{E}(S_z)$. Now the we define a mixture coeficient, λ_z and component for each S_z as

$$\lambda_z := \prod_{(i,j)\in\mathcal{E}(S_z)} w_{i,j}, \qquad p_z(x,y|\theta) := \prod_{i\in\mathcal{L}(S_z)} p_i(x,y)$$

where $p_i(x,y)$ is the leaf distribution at leaf node i paramitised with θ. It can now be proven that the SPN can be interpreted as the following mixture model,

$$p(x,y|w,\theta) = \sum_{z=1}^{Z} \lambda_z(w) p_z(x,y|\theta)$$

i.e. by the weighted sum of all Z sub-networks. For convinience we define each sum component as $p(z,x,y|w,\theta) := \lambda_z(w)p_z(x,y|\theta)$. Evaluation of $p(x,y|w,\theta)$ will never be done as the sum over Z components, instead there is a proposition.

Proposition 1 *Consider a SPN, S, a sum node $q \in \mathcal{S}um(S)$ and a child $i \in ch(q)$, then the following relation holds,*

$$\sum_{z:(q,i)\in\mathcal{E}(S_z)} \lambda_z(w) p_z(x,y|\theta) = w_{i,q} \frac{\partial S}{\partial v(q)} v(i)$$

Conditional of SPN

We will soon see how it is possible to write the conditional distribution as the mixture,

$$p(y|x) = \sum_{z\in\Sigma(S)} \gamma(x) p_{z_y}(y)$$

where $\Sigma(S)$ is the set of all sub-networks in the SPN, S - IT IS EXPONENTIALLY LARGE. And where $p_{z_y}(y)$ is defined through $p_z(x,y)$,

$$\begin{aligned} p_z(x,y) &= \prod_{l\in\mathcal{L}eaf(z_x)} \phi_l(x) \prod_{l\in\mathcal{L}eaf(z_y)} \phi_l(y) \\ &=: p_{z_x}(x) p_{z_y}(y) \end{aligned}$$

where ϕ_l is the density of the l'th leafs tractable distribution. Recall that we can interpret an SPN as the mixture model,

$$p(x,y) = \sum_{z\in\Sigma(S)} \lambda_z p_z(x,y)$$

where $\lambda_z = \prod_{(q,j)\in\mathcal{E}(z)} w_{q,j}$. First we calculate the marginal density, $p(x)$,

$$\begin{aligned} p(x) &= \int p(x,y)dy \\ &= \int \sum_{z\in\Sigma(S)} \lambda_z p_z(x,y) dy \\ &= \sum_{z\in\Sigma(S)} \lambda_z p_{z_x}(x) \int p_{z_y}(y) dy \\ &= \sum_{z\in\Sigma(S)} \lambda_z p_{z_x}(x) \end{aligned}$$

Now we are ready to calculate the conditional density,

$$\begin{aligned} p(y|x) &= \frac{p(x,y)}{p(x)} \\ &= \frac{\sum_{z\in\Sigma(S)} \lambda_z p_z(x,y)}{p(x)} \\ &= \sum_{z\in\Sigma(S)} \frac{\lambda_z p_{z_x}(x)}{p(x)} p_{z_y}(y) \\ &= \sum_{z\in\Sigma(S)} \frac{\lambda_z p_{z_x}(x)}{\sum_{z\in\Sigma(S)} \lambda_z p_{z_x}(x)} p_{z_y}(y) \\ &= \sum_{z\in\Sigma(S)} \gamma(x) p_{z_y}(y) \end{aligned}$$

So we defined $\gamma(x) = \frac{\lambda_z p_{z_x}(x)}{\sum_{z\in\Sigma(S)} \lambda_z p_{z_x}(x)}$ and this is very convinient, as we will see soon is equavalent to the derivative of the log-likehood of the SPN, which is easily obtained by automatic differentiation.

calculation of $\gamma(x)$
expectation maximization of a mixture model, is given by Bishop.. the responsibility of a datapoint to belong to one mixture component, is given by

$$\gamma(z_{nk}) = \frac{w_k p_j(x_n)}{\sum_i w_i p_i(x_n)}$$

We can prove that the responsibility is equal to the gradient of the log likehood,

$$L := \sum_n \log \sum_j w_j \exp \psi_j(x_n)$$

where we define $\psi_j(x_n) = \log p_j(x_n)$. Take the gradient

$$\frac{\partial L}{\partial \psi_j(x_n)} = \frac{w_k p_j(x_n)}{\sum_i w_i p_i(x_n)}$$

Note that the gradient easily can be found using autograd.

Mean and variance of $p(y|x)$
The mean of the conditional is just

$$\begin{aligned} E_{p(y|x)}[y] &= \sum_{z\in\Sigma(S)} \gamma(x) \int y p_{z_y}(y) dy \\ &= \sum_{z\in\Sigma(S)} \gamma(x) \prod_{l\in\mathcal{L}eaf(z_y)} E_{\phi_l}[y] \end{aligned}$$

and the variance is found using the second moment,

$$\begin{aligned} E_{p(y|x)}[y^2] &= \sum_{z\in\Sigma(S)} \gamma(x) \int y^2 p_{z_y}(y) dy \\ &= \sum_{z\in\Sigma(S)} \gamma(x) \prod_{l\in\mathcal{L}eaf(z_y)} (Var_{\phi_l}[y] + E_{\phi_l}[y]^2) \end{aligned}$$

SPN is an exponential large mixture model, with linear inference - unlike GMM. !? Write naive bayesian mixture model as a Sum Product Network

This thesis will only work with RAT spn. Which takes combinations of each dimension. and...

sum nodes play a role of mixtures over their children distribution, similar to a classic mixture model

Product nodes on the other hand, are equivalent to factorizations over independent distributions as they are combining disjoint RVs.

SPNs can also be interpreted as deep feed forward neural network [@vergari]. Here, imagine the weights of the sum nodes are parameters, leaf distributions are input neurons, root node is output and all other nodes correspond to hidden neurons

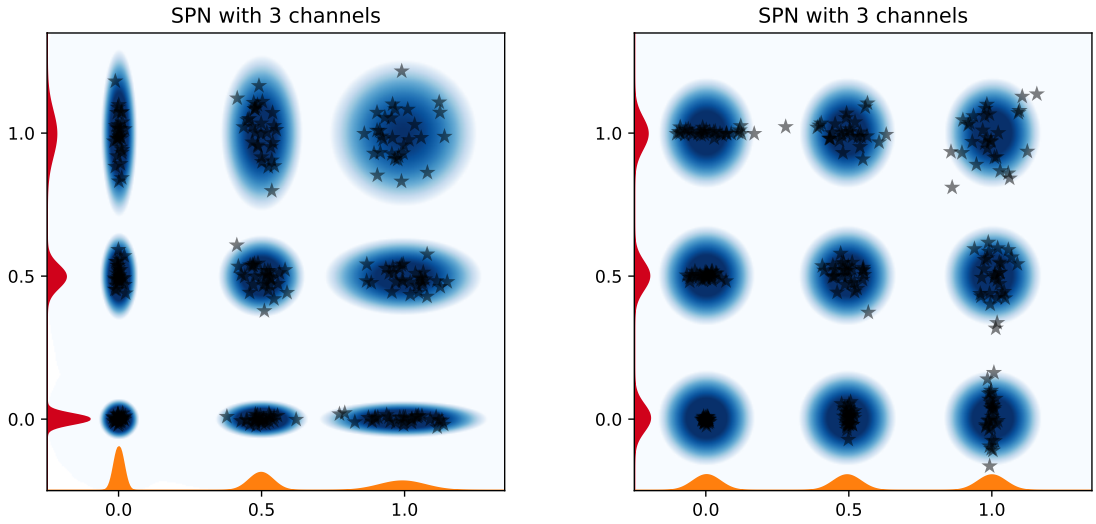


Figure 5.2: SPN

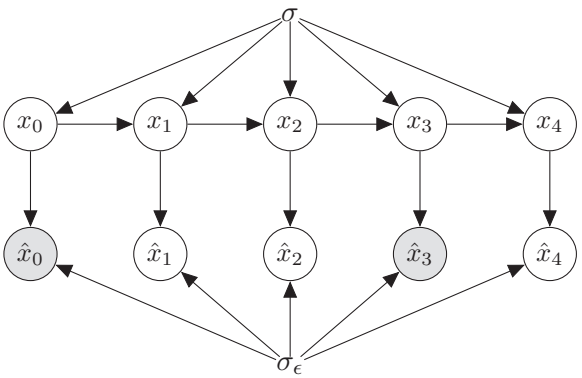
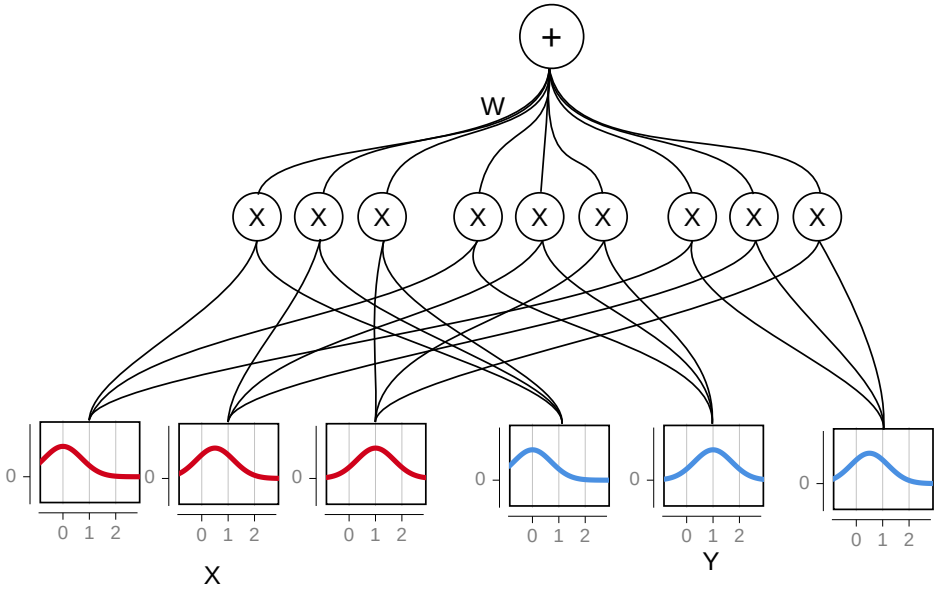


Figure 5.3: Model including wifi information



5.4 SPN

Sum-product networks are generative models, i.e. statistical models of the joint distribution $p(x, y)$. We need, however, a disciminative model for regression, i.e. a model of the conditional distribution $p(y|x)$. SPNs allow for exact inference of the joint distribution and any marginalized distribution. These combined queries is sufficient for the exact predictive posterior.

5.4.1 SPN - prediction

Prior to the inference of the predictive distribution, we assume that the SPN, S, is trained, i.e. trained leaf distributions $p_j(\cdot)$ for all leaf nodes, $j \in \mathcal{L}eaf(S) := \{j \in \mathcal{V}(S)|pa(j) = \emptyset\}$ and weights $w_{i,j}$ for the connections between every sum nodes $i \in \mathcal{S}$ and its children, $j \in ch(i)$.

The joint and the marginal distribution are evaluated in the following recursive way

---

Calculation of $p(x, y)$

Input: Fully trained SPN, with leaf distributions $p_i(\cdot)$ for $i \in \mathcal{L}eaf(S)$ and weigts $w_{i,j}$ for $(i, j) \in \{(i, j)|i \in \mathcal{S}um(S), j \in ch(i)\}$

function Eval(node i)

  if $i \in \mathcal{L}eaf(S)$ **then**

    return: $p_i(x, y)$ ▷ evaluate leaf distributions at point (x, y)

  if $i \in \mathcal{S}um(S)$ **then**

    return: $\sum_{j \in ch(i)} w_{i,j}$Eval$(j)$

  if $i \in \mathcal{P}rod(S)$ **then**

    return: $\prod_{j \in ch(i)}$ Eval(j)

$p(x) =$ Eval(root)

Calculation of $p(x)$

Input: Fully trained SPN, with leaf distributions $p_i(\cdot)$ for all leaves i and weigts $w_{\cdot,\cdot}$.
function Eval(node i)
  if $i \in \mathcal{L}eaf(S)$ **then**
    if node handle x **then**
      return: $p_i(x, y)$ ▷ evaluate leaf distributions at point (x, y)
    else
      return: 1 ▷ set node equal 1 at point (x, y)
  if $i \in \mathcal{S}um(S)$ **then**
    return: $\sum_{j \in ch(i)} w_{i,j}$Eval$(j)$
  if $i \in \mathcal{P}rod(S)$ **then**
    return: $\prod_{j \in ch(i)}$ Eval(j)
$p(x) =$ Eval(root)

So after doing two slightly different forward passes through the SPN, $p(x)$ and $p(x, y)$, using Bayes rule, we can combined the two queries into the conditional distribution:

$$p(y|x) = \frac{p(x, y)}{p(x)}$$

The predictive distribution is found with a cost of just $O(E + E + 1) = O(E)$, where E is number of edges/connections in the SPN.

5.4.2 SPN - learning

It is not enouth to do predictive inference on a SPN, we also need to fit it on the data. It is possible interpret sum-product network as a large mixture model and therefore use expectation-maximization to train the model. We will introduce that idea now. The Paper [**SPN_EM**]... defines SPN as a mixture of all sub-networks of an SPN.

Definition 2 *A sub-network $\bar{S}_z$ of S is an SPN, which includes the root S and then includes nodes according to the following recursive scheme:*

Collection of sub-network S_z of S

function Process(node i, S_z)
  if $i \in \mathcal{L}eaf(S)$ **then**
    return:
  if $i \in \mathcal{S}um(S)$ **then**
    $S_z = S_z.add(j \in ch(i))$ ▷ include one child of node i
    return: Process(j, S_z)
  if $i \in \mathcal{P}rod(S)$ **then**
    $S_z = S_z \cup \{j | j \in ch(i)\}$ ▷ include all childen of node i
    for $j \in ch(i)$ **do**
      return: Process(j, S_z)
  return: S_z
$S_z =$ Process(root,Ø)

So we see that at each sum node the number of different sub-networks multiplies with the number of children for that sum node. And thereby, the total number of sub-networks is

$$Z = \prod_{i \in \mathcal{S}um(S)} |ch(i)|$$

i.e. an exponential large amount of sub-networks. This is the amount of mixture components implicitly defined in an SPN. Denote the set of edges in the sub-network $\mathcal{E}(S_z)$. Now the we define a mixture coeficient, λ_z and component for each S_z as

$$\lambda_z := \prod_{(i,j) \in \mathcal{E}(S_z)} w_{i,j}, \qquad p_z(x, y|\theta) := \prod_{i \in \mathcal{L}(S_z)} p_i(x, y)$$

where $p_i(x, y)$ is the leaf distribution at leaf node i paramitised with θ. It can now be proven that the SPN can be interpreted as the following mixture model,

$$p(x, y|w, \theta) = \sum_{z=1}^{Z} \lambda_z(w) p_z(x, y|\theta)$$

i.e. by the weighted sum of all Z sub-networks. For convinience we define each sum component as $p(z, x, y|w, \theta) := \lambda_z(w) p_z(x, y|\theta)$. Evaluation of $p(x, y|w, \theta)$ will never be done as the sum over Z components, instead there is a proposition.

Proposition 2 *Consider a SPN, S, a sum node $q \in \mathcal{S}um(S)$ and a child $i \in ch(q)$, then the following relation holds,*

$$\sum_{z:(q,i) \in \mathcal{E}(S_z)} \lambda_z(w) p_z(x, y|\theta) = w_{i,q} \frac{\partial S}{\partial v(q)} v(i)$$

5.5 Expectation-maximization for mixture models

Mixture models can be seen as probibalistic graphical models, <fig> there one mixture component is picked according to the realization of a catagorical variable $\mathbf{Z}$ with parameters according the the mixture weights, i.e we can reformelate,

$$p(x) = \sum_{k=1}^{K} w_k p_k(x) \tag{5.1}$$

$$\iff \qquad p(x) = p_z(x), \qquad z \sim Cat(w_i, \dots, w_K). \tag{5.2}$$

In fact $p_z(x)$ is a conditial distribution, $p(x|z)$, and combined with the distribution of Z we can define the joint

$$p(x, z) := p_z(x)p(z)$$

The the case of a statistical model, data $\mathcal{D}$ is fitted by the mixture model by tuning the model parameters $\theta = \{w, \text{paramers for} p_i\}$. Then the joint distribtuion $p(\mathcal{D}, z|\theta)$ is refered as the *complete-data* likehood in the EM algorithm.

$$p(\mathcal{D}, z|\theta) := p(\mathcal{D}|z, \theta)p(z|\theta)$$

When fitting model parameters we essentially want to find the parameters, that maximize the probability of the parameters given the data, $p(\theta|\mathcal{D})$. Assuming an unimformative/flat prior $p(\theta)$,

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

$$\Rightarrow \arg\max_{\theta} p(\theta|\mathcal{D}) = \arg\max_{\theta} p(\mathcal{D}|\theta)$$

we arrive at the maximum likelihood estimate (MLE). The task of finding the MLE is conviniently done using EM algorithm, since we can look at the likehood as the marginalized *complete-data* likehood,

$$p(\mathcal{D}|\theta) = \sum_{z} p(\mathcal{D}, z|\theta)$$

Background: Expectation-maximization EM <based on [5]

Expectation maximization is a convinient method for finding ML (or MAP) estimate of a latent variable model. We consider a probibalistic model paramitised with θ,

$$p(\mathbf{X}, \mathbf{Z}|\theta)$$

where we denote all latent variables $\mathbf{Z}$, and observed variables $\mathbf{X}$. Our goal is to find the maximum of the likehood,

$$p(\mathbf{X}|\theta) = \int p(\mathbf{X}, \mathbf{Z}|\theta)\mu(d\mathbf{Z})$$

maximizating the likehood itself $p(\mathbf{X}|\theta)$ is assumed dificult but maximizating of the *complete-data* likehood $p(\mathbf{X}, \mathbf{Z}|\theta)$ is much easier. The algorithm iterates over two steps: The expecation (E) step and the maximization (M) step, defined in the following way for iteration t,

E-step

Define the functional $Q(\theta, \theta^{(t)})$, to be the expected value of the complete-data log likehood (log likehood function of θ), with respect to the only random quantaty $\mathbf{Z}$, which is assumed to follow a distribtuion with the density $p(\mathbf{Z}|\mathbf{X}, \theta^{(t)})$, i.e. the conditional distribution of $\mathbf{Z}$ given $\mathbf{X}$ and the current parameter point estimate $\theta^{(t)}$:

$$Q(\theta, \theta^{(t)}) := E_{p(\mathbf{Z}|\mathbf{X}, \theta^{(t)})}[\log p(\mathbf{X}, \mathbf{Z}|\theta)]$$

M-step

After the E-step we find the point estimate $\theta^{(t+1)}$ which maximizes $Q(\cdot|\theta^{(t)})$, i.e.

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)})$$

(local) maximization of $p(\mathcal{D}|\theta)$

Input: dataset $\mathcal{D}$, joint model $p(\mathcal{D}, \mathbf{Z}|\theta)$
while Not converged **do**
  $Q(\cdot, \theta^{(t)}) \leftarrow E_{p(\mathbf{Z}|\mathcal{D},\theta^{(t)})}[\log p(\mathcal{D}, \mathbf{Z}|\cdot)]$ ▷ E-step
  $\theta^{(t+1)} \leftarrow \arg\max_\theta Q(\theta|\theta^{(t)})$ ▷ M-step
return: $\theta^{(end)}$

Proof of correctness

We will now give a short proof that maximizing $Q(\cdot|\theta^{(t)})$ maximizes the likelihood $p(\mathbf{X}|\theta)$, where we assume that $\mathbf{Z}$ is a random vector with a discrete distribution. This allow us to use Gibbs inequality:

$$\sum_z p_1(z) \log p_1(z) \geq \sum_z p_1(z) \log p_2(z)$$

where $p_1(\cdot)$ and $p_2(\cdot)$ are densities belonging to two discrete distributions of Z, equality if $p_1(\cdot) = p_2(\cdot)$. From now on we will alter the subscript on the expecations, just have in mind that

$$E_{\theta^{(t)}}[g(Z)] := E_{p(\mathbf{Z}|\mathbf{X},\theta^{(t)})}[g(Z)] = \sum_z g(z) p(\mathbf{z}|\mathbf{X}, \theta^{(t)})$$

Now to the proof: From bayes rule $p(\mathbf{X}|\theta) = \frac{p(\mathbf{X},\mathbf{Z})}{p(\mathbf{X})}$ we can write

$$\log p(\mathbf{X}|\theta) = \log p(\mathbf{X}, \mathbf{Z}) - \log p(\mathbf{Z}|\mathbf{X}, \theta)$$

Now, taking the expecation of the above w.r.t. $p(\mathbf{Z}|\mathbf{X}, \theta^{(t)})$, yields,

$$\begin{aligned} \log p(\mathbf{X}|\theta) &= E_{\theta^{(t)}}[\log p(\mathbf{X}, \mathbf{Z}|\theta)] - E_{\theta^{(t)}}[\log p(\mathbf{Z}|\mathbf{X}, \theta)] \\ &= Q(\theta, \theta^{(t)}) + E_{\theta^{(t)}}[\log p(\mathbf{Z}|\mathbf{X}, \theta)] \end{aligned}$$

Since the above equation holds for any θ, it also holds for $\theta^{(t)}$ now we have,

$$\log p(\mathbf{X}|\theta^{(t)}) = Q(\theta^{(t)}, \theta^{(t)}) + E_{\theta^{(t)}}[\log p(\mathbf{Z}|\mathbf{X}, \theta^{(t)})]$$

Subtracting the two equations, we get,

$$\log p(\mathbf{X}|\theta) - \log p(\mathbf{X}|\theta^{(t)}) = Q(\theta, \theta^{(t)}) - Q(\theta^{(t)}, \theta^{(t)}) + E_{\theta^{(t)}}[\log p(\mathbf{Z}|\mathbf{X}, \theta)] - E_{\theta^{(t)}}[\log p(\mathbf{Z}|\mathbf{X}, \theta^{(t)})]$$

From Gibb's inequality we have that $E_{\theta^{(t)}}[\log p(\mathbf{Z}|\mathbf{X}, \theta^{(t)})] \leq E_{\theta^{(t)}}[\log p(\mathbf{Z}|\mathbf{X}, \theta)]$ where equality only holds for $\theta^{(t)} = \theta$, giving

$$\log p(\mathbf{X}|\theta) - \log p(\mathbf{X}|\theta^{(t)}) \geq Q(\theta, \theta^{(t)}) - Q(\theta^{(t)}, \theta^{(t)})$$

so optimizing $Q(\theta, \theta^{(t)})$ will optimize $\log p(\mathbf{X}|\theta)$ as least as much.

EM for Gaussian mixture

For a Gaussian mixture the p_k distributions in (5.1) is substituted by Gaussain pdfs, i.e. $p_k(x) = \mathcal{N}(x|\mu_k, \Sigma_k)$ and the density of a catagorical distribution is $p(z) = \sum_{k=1}^{K} 1_{z=k} w_k = w_z$, combining the two we get the joint distribution,

$$p(x, z|w, \mu, \Sigma) = w_z \mathcal{N}(x|\mu_z, \Sigma_z)$$

Taking the log and defning $\theta = \{\mu_1, \Sigma_1, w_1, \ldots, \mu_K, \Sigma_K, w_K\}$, and assuming iid data

$$\log p(X, Z|\theta) = \sum_{i}^{n} (\log(w_{z_i}) + \log(\mathcal{N}(x_i|\mu_{z_i}, \sigma_{z_i})))$$

Now we are ready to calculate $Q(\theta, \theta^{(t)})$, by taking the expectation of the complete-data log likehood with respect to the distribution, $p(Z|X, \theta)$,

$$\begin{aligned} E_{p(Z|X,\theta^{(t)})}[\log p(X, Z|\theta)] &= \sum_{i}^{n} E_{p(Z|X,\theta^{(t)})}[p(X_i, Z_i|\theta)] \\ &= \sum_{i}^{n} E_{p(z_i|x_i,\theta^{(t)})}[p(x_i, z_i|\theta)] \end{aligned}$$

> **Expectation with repect to unnecessary variables**
> the last equation holds since taking expecation over a function of a random variable x with respect to a that random variable and more random variabels, x, y, is equivalent to the expecation with respect to just x, i.e.
>
> $$\begin{aligned} E_{x,y}[g(x)] &= \int\int g(x)p(x, y)dydx \\ &= \int g(x) \int p(x, y)dydx \\ &= \int g(x)p(x)dy = E_x[g(x)] \end{aligned}$$

the posterior distribution is calculated the following way,

$$\begin{aligned} p(z|x, \theta^{(t)}) &= \frac{p(x, z|\theta^{(t)})}{p(x|\theta^{(t)})} \\ &= \frac{p(x, z|\theta^{(t)})}{\sum_z p(x, z|\theta^{(t)})} \\ &= \frac{w_z^{(t)} \mathcal{N}(x|\mu_z^{(t)}, \Sigma_z^{(t)})}{\sum_{k=1}^{K} w_k^{(t)} \mathcal{N}(x|\mu_k^{(t)}, \Sigma_k^{(t)})} \end{aligned}$$

For simplificaiton we will denote, $\gamma^{(t)}(z_i) := p(z_i|x_i, \theta^{(t)})$, interpreted as the probability of datapoint x_i to belong to class z_i. Bishop [5] calls this probability function the *responsibility*. We can now conclude the **E-step**.

$$
\begin{aligned}
Q(\cdot, \theta^{(t)}) &= \sum_{i=1}^{n} p(x_i, z_i|\cdot)\gamma^{(t)}(z_i) \\
&= \sum_{i=1}^{n} \left[\gamma^{(t)}(z_i)\log(\cdot_{z_i}) + \gamma^{(t)}(z_i)\log(\mathcal{N}(x_i|\cdot_{z_i}, \cdot_{z_i}))\right]
\end{aligned}
$$

or more concretely $\theta = \{\mu_1, \Sigma_1, w_1, \ldots, \mu_K, \Sigma_K, w_K\}$,

$$
Q(\theta, \theta^{(t)}) = \sum_{i=1}^{n} \gamma^{(t)}(z_i)\log(w_{z_i}) + \gamma^{(t)}(z_i)\log(\mathcal{N}(x_i|\mu_{z_i}, \Sigma_{z_i})).
$$

$Q(\cdot, \theta^{(t)})$ is a concave function - the Gaussian is log-concave and a sum of concave functions is also concave - so it is sufficient and nessesary to find its maxima by the root of its derivative,

$$
\frac{d}{d\theta}Q(\theta^*, \theta^{(t)}) = 0 \iff \theta^* = \arg\max_{\theta} Q(\theta, \theta^{(t)})
$$

Giving the updates...

EM for SPN

5.6 Gaussian Mixture Regression

The Gaussian mixture is a generative model of the joint probability of x and y given as,

$$
p(x, y) = \sum_{k=1}^{K} \pi^{(k)}\mathcal{N}(x, y|\mu^{(k)}, \Sigma^{(k)}), \qquad \mu = \begin{bmatrix}\mu_x \\ \mu_y\end{bmatrix}, \ \Sigma = \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}
$$

This is trained using the EM algorithm. We will now show how the conditial is calculated exact.

5.6.1 GMR - prediction

We need the conditional distribution of the Gaussian mixture in order to get the predictive distribution. We will now formulate the conditial distribution in terms of conditional and marginals of the individual mixture components. First of all the marginal distribution $p(x)$ of the mixture is given as, How??!

$$
p(x) = \sum_{k=1}^{K} \mathcal{N}(x|\mu_x^{(k)}, \Sigma_{xx}^{(k)})
$$

next the joint distribution can be decomposed with the probability chain rule,

$$
\begin{aligned}
& p(x, y) = p(x)p(y|x) \\
\implies \qquad & \mathcal{N}(x, y|\mu, \Sigma) = \mathcal{N}(x|\mu_x, \Sigma_{xx})\mathcal{N}(y|\mu_{y|x}, \Sigma_{y|x})
\end{aligned}
$$

And we can formulate the conditial in terms of individual multivariate Gaussians,

$$
\begin{aligned}
p(y|x) &= \frac{p(y, x)}{p(x)} && (5.3) \\
&= \sum_{k=1}^{K} \frac{\pi^{(k)}}{p(x)}\mathcal{N}(x, y|\mu^{(k)}, \Sigma^{(k)}) && (5.4) \\
&= \sum_{k=1}^{K} \frac{\pi^{(k)}\mathcal{N}(x|\mu_x^{(k)}, \Sigma_{xx}^{(k)})}{p(x)}\mathcal{N}(y|\mu_{y|x}^{(k)}, \Sigma_{y|x}^{(k)}) && (5.5) \\
&= \sum_{k=1}^{K} \pi_{y|x}^{(k)} p(y|x, \mu_{y|x}^{(k)}, \Sigma_{y|x}^{(k)}) && (5.6)
\end{aligned}
$$

where $\pi_{y|x}^{(k)} := \frac{\pi^{(k)}\mathcal{N}(x|\mu_x^{(k)},\Sigma_{xx}^{(k)})}{\sum_{i=1}^{K}\mathcal{N}(x|\mu_x^{(i)},\Sigma_{xx}^{(i)})}$. So we see that the conditonal of a Gaussian mixture model is again a Gaussian mixture model.

Background: Conditional og multivariate Gaussian

The conditional distribution is defined as [5] ..?

$$p(y|x,\mu,\Sigma) = \mathcal{N}(y|\mu_{y|x},\Sigma_{y|x}) \qquad \mu = \begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix},\ \Sigma = \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}$$

where

$$\mu_{y|x} = \mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x-\mu_x) \tag{5.7}$$
$$\Sigma_{y|x} = \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy} \tag{5.8}$$

5.6.2 GMR - Leaning

EM-algorithm, classic example.

Bibliography

[1] Mashall Aryan. “Sample-efficient Optimization Using Neural Networks”. https://researcharchive.vuw.ac.nz/handle/10063/9035. PhD thesis. Victoria University of Wellington, New Zealand, 2020.

[2] Jasper Snoek et al. *Scalable Bayesian Optimization Using Deep Neural Networks*. 2015. arXiv: 1502.05700 [stat.ML].

[3] Jost Tobias Springenberg et al. “Bayesian Optimization with Robust Bayesian Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. url: https://proceedings.neurips.cc/paper/2016/file/a96d3afec184766bfeca7a9f989fc7e7-Paper.pdf.

[4] Roman Garnett. *Bayesian Optimization*. in preparation. Cambridge University Press, 2022.

[5] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

A Implementation

Bayesian optimization is an effective framework for finding optimizers of a highly expensive (in terms of money, time, human attention or computational processing) or noisy objective function. First a prior is defined over possible functions and updated to a posterior according to already obtained observations/samples of the objective function. Next an acquisition function uses this posterior, also called an surrogate model, and is then utilized to find the next location in the optimization landscape to sample from. The far most common surrogate model is Gaussian Process (GP), partially due to its ability to represent its posterior in closed form. However, it also comes with short comings: Its inference, although it is exact, scales cubic with amount of samples and it impose strong assumptions of a well behaved objective function.

This thesis aims to investigate surrogate models different from GPs in order to improve on either the accuracy of the surrogate model or the inference cost of it. Meanwhile Bayesian Neural Networks (BNN) already have proven useful as surrogate models [1][2][3] (with cost of inference, which scales linearly and less strong assumptions) this thesis additionally wants to investigate Sum Product networks (SPN). An SPN is - similarly to a BNN - a deep probabilistic model and still expressive but with tractable inference, which potentially could lead to advantages over BNNs.

Technical
University of
Denmark

Building 321
2800 Kgs. Lyngby
Tlf. 4525 1700

www.github.com/SimonKruse0/master-thesis