第 7 章

メタ的なデータサイエンス:ベイズ的最適化による実験計画法を例に

これまでは基本的にデータを直接的に扱い分析することを指してデータサイエンスとよんできた傾向にあります。その一方で、広義のデータサイエンス的素養は、研究やフィールドワークそれ自体の計画を考えるのに使うなど、ある種のメタ的とも言うべき用途も考えられます。いわゆる実験計画法だったりワークフローとでもいうものでしょうか。ここでは、実験計画法の中でもとくにベイズ的最適化について少し述べたいと思います。テクニカルな部分もありますが、背景にある考え方はかなり一般的な内容ですので、研究に使ってみようかな... といった場合は以降を読んで試してみるか、私にお声がけください。

ベイズ的最適化の定式化の多くはガウス過程 (Gaussian Process; GP) を用いて行われます[*1]。ガウス過程に関しては PRML [6] や GPML [11] といった機械学習分野におけるバイブルの他に、比較的新しい日本語の文献 [12] もありますので是非チェックしてみてください。ちなみに PRML も GPML も著者のページで無料で pdf がダウンロードできます。オープンリソースの時代、素晴らしいですね。また [ガウス過程と機械学習 [12]] は、非常に丁寧に説明されている素晴らしい本です。ガウス過程を GPML で勉強してきた人にとっても待望の日本語の書籍であったかと推測しますが、平易に書かれている文章の節々に「あっここの言い回し (行間) は GPML とかで書いてたアレが考慮されてるな!?」という気づきが散りばめられていて含蓄に富んだ素晴らしい本です[*2]。

7.1 ベイズ的最適化による実験計画法

研究や教育の現場、あるいは日常生活において、実験条件や計算のパラメータ (一般に多次元) を変えながら興味のある量 (測定量や計算結果)、とくにその最適解 (最大値・最小値を与える条件・パラメータ) を調べたい、こうしたシチュエーションはめちゃくちゃたくさんあるかと思います。

[*1] 原理的には他の確率過程でも良いはずですが性能やとっつきやすさの点で GP が選ばれるのだと想像します。

[*2] 読む人の知識量に応じて、印象や情報量が変わるようなある種の [文章の多層性] みたいなのって素敵ですよね (伝われ...)。これを端的に言い表す言葉があれば教えてください。

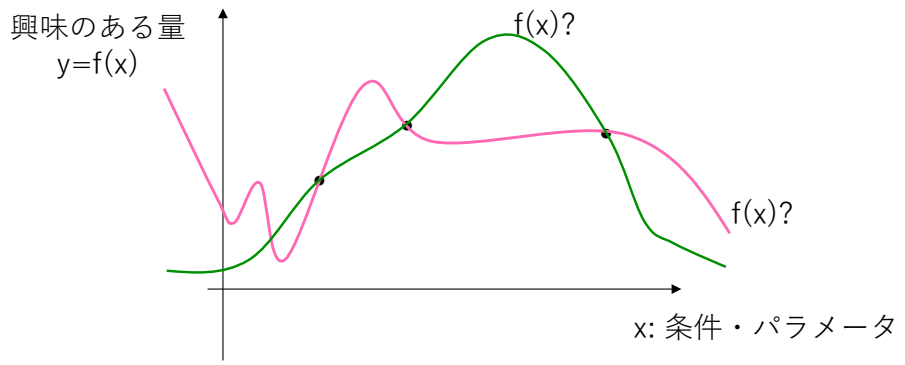

図 7.1: 研究や教育で目にするブラックボックス関数のポンチ絵。黒点はすでに測定 (計算) されたものを表す。

こうした興味のある量を $f(x)$ として、それを特徴づける実験条件や計算のパラメータなどを x と書くことにすると、我々の目標は

> すべての点 x では $f(x)$ を調べられないけれど、なんとか一番良い値を与える x をみつけたい

ということになろうかと思います。ポンチ絵を図 7.1 に示しました。もちろん、条件を変えながら網羅的に調べ尽くせば関数 $f(x)$ の形は予測できますが、その一方で、教育・研究に避けるリソース (お金・マシンタイム・時間など) は限られています。

興味があるのが $f(x)$ の最大値でも最小値でもどっちでもよいのですが[*3]、簡単のため最大値が知りたいとしましょう。また、一般に $f(x)$ はブラックボックス関数になっていて、勾配が解析的に計算できません[*4]。つまり、ある点 x_1 で $f(x_1)$ の値がわかっても、勾配が解析的に計算できないので[*5]、最適解をうまく求められないのです。最適解をうまく探すための方法は最適化手法と呼ばれますが、その多くが勾配 (一階微分や二階微分など) の情報を使いますので、勾配が計算できないあるいは勾配の計算自体が大変すぎるというのはたいへん困るわけです。x_1 から微小距離離れた点での値を計算して微分を数値的に計算する方法もありますが、説明変数の次元が上がると計算が大変になってしまうので、計算が生きてる間に終わらない、なんてことになってしまいます。

そんなとき、どうすればいいでしょうか？「私費や学生のマンパワーを駆使して強行!!」や「おとなしく科研費等の書類を書きまくる」というのは、とても現代の価値観にはふさわしくありません[*6]。以下では、ベイズ的最適化 (Bayesian optimization) という概念を用いて、少ない探索回数で最適解の近似値を推定する方法のひとつをご紹介します。

と、その前に、どうして条件を変えながら網羅的に探索する方法 [グリッドサーチ] がうまく行かないかもう少し具体的に説明しておきましょう。たとえばパラメータが N 次元だったとして、各

[*3] なにか $f(x)$ の最小値を求めたいというとき、$f(x)$ に-1 をかけて定義してしまえば、最大値を求める問題に置き換えられます。
[*4] 最大値・最小値になるところでは微分 (勾配) がゼロになるということは中学校や高校の数学で習いました。習いましたよね？
[*5] 進んだ注:Hellmann-Feynman の定理が使える系では例外です
[*6] 当然ながら、これまでがそうだったと言っているわけではありません。

成分を-1.0 から 1.0 まで 0.1 刻みで変えながら探索したいとします。$N = 1$ の場合なら、21 通り調べるだけです。1 個の値あたりにかかるコストをたとえば時間で表すとして...1 秒かかるとしましょう。21 秒、なんのことはないですね。では $N = 2$ の場合はどうかというと 21^2 通り調べないといけなくなるので、440 秒くらいです。8 分もあればできそう、といったところでしょうか。では思い切って $N = 10$ にしてみましょう。21^{10} がいくらかと言うと... だいたい 2.0×10^{13}、20 兆秒かかります。20 兆秒というのはだいたい 63 万年です。年になおしてもまだピンと来ないくらい膨大な時間がかかってしまいます。論文やレポートを書くのに 63 万年もかかっていては大変 (?) です。このように、次元が増えるにつれて探索すべき組み合わせが爆発的に増大してしまい (次元の呪いといったりします)、とてもじゃないけれども人間が生きている時間スケールで探索が終わらなくなってしまいます。

ここでひとつ断っておくと、なにかパラメータが 10 個あるというのはなにも極端な例ではありません。計算機科学と呼ばれる分野で行われているシミュレーションではウン百ものパラメータがあるというのは日常茶飯事ですし、機械学習モデルのニューラルネットワークでの各ノードでの重みやバイアスをパラメータと呼ぶことにすると、10 個なんてすぐ超えます。ただ、今の例でも、実際の研究や産業利用なんかでも、もちろん 20 兆もの組み合わせを闇雲に探すことはしません (できません)。

いくつか探索 (実験・計算) をしていくうち、このあたりのパラメータは有望そう (=最大値・最小値を与えそう) だな、あのあたりのパラメータは望みがないな、といった [経験] が得られます。ベイズ的最適化ではこうした経験を積極的に活用して、人間の勘ではなく数学的な定式化によって提案される探索を行うことで、最適化問題を効率よく解くことを目指します。

機械学習などの分野ではよく [探索と活用のトレードオフ] という概念が登場します。[活用] というのは「ふむふむ今いるあたりは良さそうだ」と今探している近くを探すこと、[探索] というのは「井の中の蛙状態は怖いし、もう少し別の可能性も探ってみようか...」と遠くの可能性も探ってみることです[*7]。ここでいう近いとか遠いとかっていうのは、パラメータ同士の距離 (たとえばユークリッド距離) です。ベイズ的最適化はこの探索と活用のトレードオフを定式化します。

簡単のため一次元の説明変数 x で考えてみましょう。図 7.2 に、模式図を示しました。まず、$x = 1$ と $x = 5$ の 2 点で既に y の値が観測されていて、$-2 \leq x \leq 12$ の領域で最大の y を与える x を知ることが目的だとします。ただ 1 つの x の値に対して対応する y の値を求めるコストが大きくて、x を変えながらしらみつぶしに探すのは厳しいとしましょう。たとえば実験に使う薬品が高価だとか、高性能なパソコンでも計算時間がめちゃくちゃかかるといった状況がこれに当てはまります。

また、図 7.3 に、このデータを生成している背後にある真の関数を描きました。今は思考実験のた

[*7] 以後は観測 (データを実験なり計算なりで得ること) のことを探索と言ったりもするので、この文脈で用いるときは [活用] とセットで明示的に使います。

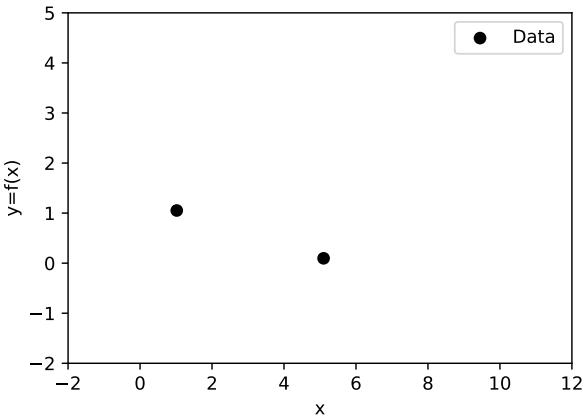

図 7.2: 2 個データが観測されているとする。

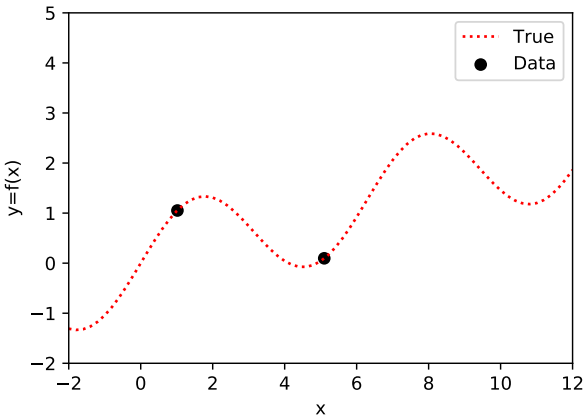

図 7.3: 真の関数 (赤線) を加えたもの

め、真の関数は $f_{\mathrm{true}} = \sin x + 0.2x$ で与えられると仮定しておいて、観測されるデータには、この値 (赤線) に分散が 0.01 のランダムノイズが加えられているとしましょう。$x = 8$ あたりで最大値を与えるなというのが視覚的に分かりますし、真の関数の形がわかっている場合は、微分すれば $x = \arccos(-0.2) + 2n\pi$ で極大値をもつことがわかり、$n = 1$ がこの場合の最適解だということが求められます。現実の問題の場合にはそもそもこの赤色の線がわからなくて苦労してるわけです。

さて、データが 2 点 $(x = 1, 5)$ 観測されています、と。次に探索すべき点はどこでしょうか？ 多くの人は (赤線を見なかったとしても),[なんとなく]$x = 1$ や $x = 5$ のすぐ近くではなく、左肩上がりに上がっていそうだと期待して $x = -2$ を探したり、まだ全然探していない $x = 9$ あたりを探してみたくなるのではないでしょうか？

以下ではこのなんとなくをもう少しちゃんと定式化することにしましょう。詳細はあとに譲ります (→ 7.1.1 節) が、ガウス過程 (GP) と呼ばれる確率過程を導入することで、様々な可能な関数の重ね合わせとして、各点 x で関数の確率分布が得られます。図 7.4 にその様子を示しました。うっすら描かれている曲線が、ガウス過程から生成されたサンプル関数です。

図をみやすくするために数本しか描いていませんが、こうした関数を無数に集めてきた際の [平均的な関数] が青い破線で示されていて、青い領域はその不確実性 (1σ) に対応します。

ガウス過程を用いたベイズ的最適化では、[獲得関数] と呼ばれる量を最大化する点を次の探索点として提案します。かなり大雑把に言い換えると、未知のブラックボックス関数がより不確実なところ、つまり次の探索で関数の不定性がぐっと小さくなりそうなところで次の測定をしましょうというのが基本的なアイデアです。

ここでは最もポピュラーな獲得関数の一つである Probability of Improvement (PI) と呼ばれるも

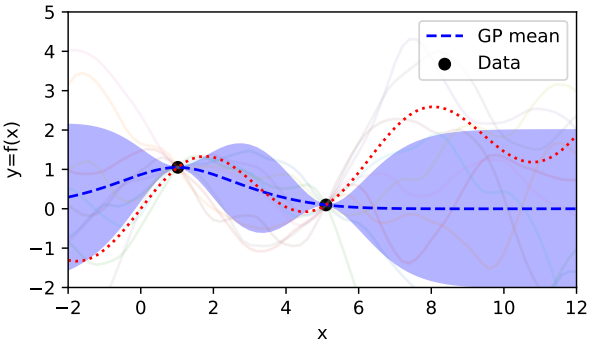

図 7.4: 2 個データが観測されている場合のガウス過程 (GP) による予測の模式図。GP の予測の平均が青の破線、1 シグマの誤差が青いバンドで示されています。うっすらとした線はガウス過程から生成された [関数のサンプル] です。

のを紹介します:

$$a_{PI}(x_c) = \Phi(Z), \tag{7.1}$$

$$Z \equiv \frac{\mu(x_c) - f_+}{\sigma(x_c)}, \tag{7.2}$$

というものです[*8]。Φ は正規分布の累積分布 (cdf) で, f_+ はこれまでの測定で得られた y の最大値, $\mu(x_c)$ と $\sigma(x_c)$ は値が未知の各点 x_c(次の探索候補 (candidate) を表す c をつけました。) でのガウス過程による予測の平均と分散です。$\Phi(Z)$ は Z の値が正に大きくなればなるほど 1 に単調に近づく関数ですので、上の式で言っているのは, Z が大きくなるところ、つまり、GP による関数分布予測の平均 $\mu(x_c)$ が今までの最大値よりも大きくなるところで獲得関数の値が大きくなることを意味しています。次に調べる点 x_{new} はこの $a_{PI}(x_c)$ が最大になるところを選べばよい、というわけです。

PI は、ガウス過程の予言の平均が一番デカくなるところを探せ、と直観的に分かりやすいのですが、実際にはすぐ局所的な探索になってしまう場合が多いことが知られているので、PI を拡張した Expected Improvement (EI):

$$f_{EI}(x_c) = (\mu(x_c) - f_+)\Phi(Z) + \sigma(x_c)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(\mu(x_c) - f_+)^2}{2\sigma^2(x_c)}\right) \tag{7.3}$$

が使われることも多いです。他にもありますが、以降はこの EI を使います。

EI の定義で獲得関数を計算したものが図 7.5 になります。これを見ると、我々が感覚的に探索したくなる $x = 9$ あたりと定義域の端っこの方 $x = -2$ で、獲得関数の値が大きくなっています。それでは、$x = -2$ で観測をしてみましょう。図 7.6 のようになりました。

新しく測定されたデータが黒点で示されています (図の上半分)。データ (経験) が一つ増えたので、それをもとにしながらまた GP と獲得関数を計算してやると (図の下半分)、次に探索するの

[*8] f_+ に正の微小量を加える流儀もありますが省略します

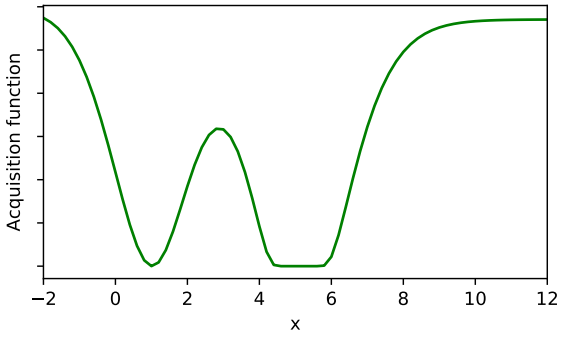

図 7.5: N_d(観測されたデータの数)$= 2$ での獲得関数。獲得関数の値のスケールには意味はないので y 軸に値は書いていません。

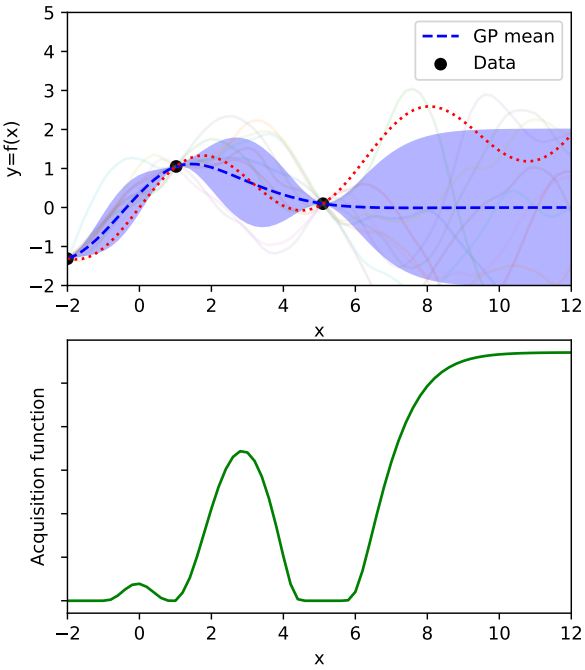

図 7.6: N_d(観測されたデータの数)$= 3$。獲得関数の値のスケールには意味はないので y 軸に値は書いていません。

は右端の方が良いようです。といったように、逐次観測と GP および獲得関数の計算を繰り返すことで、最適解の探索を行います。

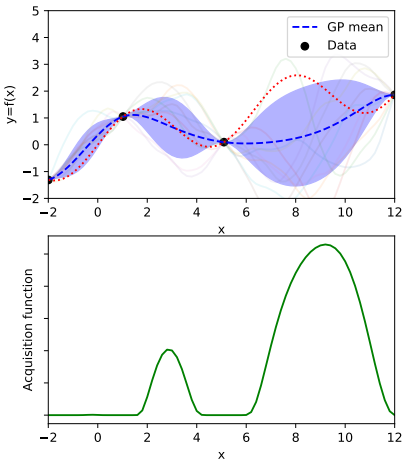

図 7.7: $N_d = 4$

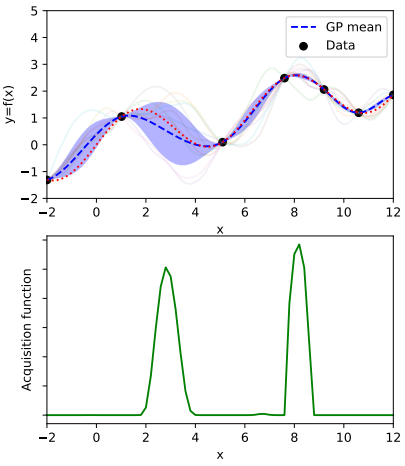

図 7.8: $N_d = 7$

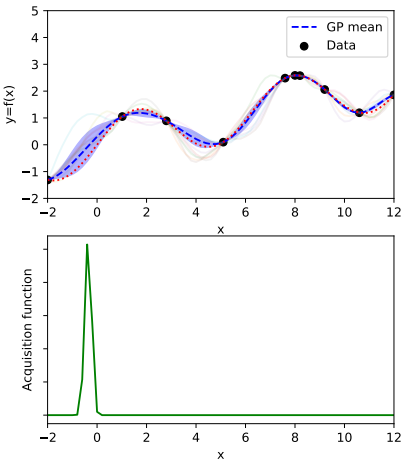

図 7.9: $N_d = 10$

さて、この作業を図 7.7–7.9 のように繰り返していくと... 10 個も観測すれば最適解をかなりの精度で見つけることができました。さらに真の関数 (赤線) の関数形がわからなくとも、GP による予測は、観測を繰り返すことで尤もらしい関数がどんどん狭まってきて、予測 (青の破線とバンド) が真の関数を内包しながら漸近していくことが見て取れます[*9]。

これまで示した例は 1 次元でしたので、あまりベイズ的最適化の [ありがたみ] は感じられなかったかもしれません。ただ、次元が 3 次元以上になると、人間の頭では視覚的にとらえることができませんから、[次にこのあたりを探せばよさそうだ] とアタリをつける事自体容易ではありません。こうした観点からも、大域的最適解を効率よく求める方法としてベイズ的最適化を用いるというのは一つの有効な手段になりそうです。

ここでは、1 回の探索にかかるコストに比べて、GP や獲得関数の計算が劇的に低コストだと仮定していることに注意してください[*10]。

適切な [距離] を定義してやれば、x 軸は実数値でなくても定式化できますので、少し想像力を働かせると、様々なことに応用ができます。

なお、これまでの絵を描いたりガウス過程や獲得関数の計算をするコード (JupyterNotebook) を GitHub の実践データサイエンス用のレポジトリで公開しています。実は、こうしたベイズ的最適化の過程というのはライブラリ化されていて、たとえば Python なら pip などで [Gpy] と [GpyOpt] というライブラリをインストールすればすぐに (もっと複雑な機能を) 使うことが出来ます[*11]。

---

*9 この計算結果が、真の関数への漸近を証明したわけではないことに注意

*10 多くの場合これは正しいですが、ガウス過程は多次元の正規分布によって定式化されていますので、パラメータの次元が上がったり、探索が密になったりすると共分散行列が満たすべき半正定値性などが数値的に損なわれてしまう恐れがあるので、適切なカーネル関数の設計や行列のスケール変換をするなどの工夫 (とその正当性の検証) が必要になり計算コストも増大します。

*11 ただし、GPy というライブラリは予測分散がデフォルトで大きめに設定されているため、使用する際は注意が必要です参考となるノートブック

車輪の再発明

プログラミングの世界でよく用いられる表現に [車輪の再発明] という言葉があります。すでにあるものをイチから作るのではなく既存のよく出来たものを使いましょうというような意味です。ただし、自分でイチから [車輪] を作ってその仕組みや動作を体験しておくことは、背景にある理屈をきちっと理解したり、世界で限られた人にしか必要がないマニアックな計算 (=つまり研究) をするためには、かなり重要だと個人的には考えています。いきなり洗練されたライブラリを眺めても長過ぎたり依存関係が複雑で中で何をやっているのか分からない、ということは往々にしてありますし、汎用ライブラリではすぐにできない [かゆいところに手が届くコード] を作成するには、こうしたイチから作るといった経験が必要になるかと思います。また、誰かが作った有名なコードだからバグが無いという保証も有りません。

7.1.1　ガウス過程 (Gaussian Process; GP)

この節では、もう少し具体的に GP の定式化を説明しておきます。この節は読み飛ばしても構いませんが、実際に応用する際には重要になります。

ガウス過程の説明はどういう文脈から導入するかに左右されますが[*12]、ここでは、回帰モデル、とくに回帰を行う関数の生成器としての立場から簡略な説明をします。より進んだ内容としては Gaussian Processes for Machine Learning (通称:GPML) [11] というバイブルがあるので、是非御覧ください。著者のページから無料で読めます。

まずいまの図 7.2 のように、データがあったとして、これを表現する関数は無数に存在します。ガウス過程は、そのような無限の関数が存在する関数空間の中で、ある仮定のもとで尤もらしいものに高い重みを与える確率過程とみなすことができます。定式化のために以下の仮定を置きます。

- 仮定１:任意の近傍二点 (今の場合一次元の x_i, x_j) でとる値 (y_i, y_j) は [似ている]
- 仮定２:[似ている] 度合いは非負の値を持つカーネル関数で規定される
- 仮定３:既知のデータと未知の点での値が多次元正規分布に従って生成される

仮定１は有る種の連続性の仮定のようなもので、考えたい関数が大きな不連続変化をしないという (弱い) 仮定です[*13]。

仮定２に関してですが、カーネル関数およびカーネル法の詳細に関しては文献に譲るとして、ここでは少々天下り的に、Mátern Kernel と呼ばれるものを導入します。x_i, x_j の 2 点に対して、距

[*12] 初学者の方に向けた Tips: 特徴空間という言葉が最初にでてくる書籍は、とりあえず使ってみたいという場合に選ぶのはあまりおすすめしません。一方で、きちんと理解したいという目的には素晴らしい本である可能性が高いです。

[*13] 進んだ注:関数が不連続に変化するというと、物理系では相転移が存在するような場合に相当しますが、実際に相転移がある場合にも適切にカーネルの設計を行えば GP で扱える例があることが論文で示されています [13]。

離 $r \equiv |x_i - x_j|$ の関数

$$k_M(r = |x_i - x_j|; \nu) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{\ell} \right)^{\nu} K_\nu(\frac{\sqrt{2\nu} r}{\ell}). \tag{7.4}$$

で、$\Gamma(\cdot)$ はガンマ関数, $K_\nu(\cdot)$ は第 ν 種の変形ベッセル関数で、ℓ は相関長 (correlation length) と呼んだりします[*14]。この式が言っていることは、距離が近い 2 点では相関が強い (k_M の値が正に大きい) よね、とただそれだけです。ガウス過程の応用では通常 $\nu = 3/2$ や $\nu = 5/2$ が用いられ[*15]、この場合はベッセル関数とかガンマ関数とかいうものはすっかり消え失せて指数関数だけでもう少し簡単な形に書き表すことができます。

以降では $\nu = 5/2$ の場合の Mátern Kernel を使うことにして、具体的な表式を書き下しておきましょう。

$$k_M(r; \nu = 5/2) = \left(1 + \frac{\sqrt{5} r}{\ell} + \frac{5r^2}{3\ell^2} \right) \exp\left(-\frac{\sqrt{5} r}{\ell} \right). \tag{7.5}$$

さっきの表式よりは、[距離が近い 2 点では相関が強い] が分かりやすい式になっているかと思います[*16]。

仮定３について、多次元正規分布という言葉が出てきましたので、少し整理しておきます。変数 $\boldsymbol{\theta}$ についての規格化された N 次元正規分布は,N 次元の平均ベクトル $\boldsymbol{\mu}$ と,$N \times N$ の共分散行列 $\boldsymbol{\Sigma}$ で以下のように書けます:

$$\mathcal{N}(\theta|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \equiv \frac{1}{\sqrt{(2\pi)^N \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}) \right). \tag{7.6}$$

変数 $\boldsymbol{\theta}$ が多次元正規分布に従うことを $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ と書いたりします。

仮定 3 が言っていることは、興味のある量の観測値と予測値を $\boldsymbol{y}$ と $\boldsymbol{y}'$ というベクトルで書くと、$\boldsymbol{y}$ と $\boldsymbol{y}$ の同時分布が以下の多次元正規分布で与えられるということです。

$$P(\boldsymbol{y}, \boldsymbol{y}') = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}' \end{bmatrix} \middle| \middle| \begin{bmatrix} \boldsymbol{\mu}_y \\ \boldsymbol{\mu}'_y \end{bmatrix}, \boldsymbol{\Sigma} \right), \tag{7.7}$$

ここで、観測値 (D 個) と予測値 (P 個) の平均ベクトル $\boldsymbol{\mu}$,$\boldsymbol{\mu}'$ は通常データを平均 0, 標準偏差 1 に標準化 (白色化) してしまうので、ゼロベクトルに取ることが多いです。$D + P$ 次元の共分散行

[*14] この相関長は、ハイパーパラメータと呼ばれるものの一種で、大雑把にはどれくらいの離れたデータが相関し合うかするかを特徴づける量です。簡単のため今は適当な値にきめてしまいましょう。

[*15] ガウス過程からサンプルされる平均関数が $\nu - 1/2$ 階微分可能になります。詳しくは GPML の 4 章を参照。

[*16] 個人的にはカーネルの頭にもうひとつのハイパーパラメータ τ (global strength) を入れる流儀が好みですが、簡単のため $\tau = 1.0$ にしました。

列 $\mathbf{\Sigma}$ の各成分を、上で定義したカーネル関数を用いて

$$\mathbf{\Sigma} = \begin{pmatrix} k_M(|x_1 - x_1|) & \cdots & k_M(|x_1 - x_D|) & k_M(|x_1 - x'_1|) & \cdots k_M(|x_1 - x'_P|) \\ k_M(|x_2 - x_1|) & \cdots & k_M(|x_2 - x_D|) & k_M(|x_2 - x'_1|) & \cdots k_M(|x_2 - x'_P|) \\ \vdots & & & & \vdots \\ k_M(|x'_P - x_1|) & \cdots & k_M(|x'_P - x_D|) & k_M(|x'_P - x'_1|) & \cdots k_M(|x'_P - x'_P|) \end{pmatrix}, \tag{7.8}$$

で与えます。ここでは r や $\nu = 5/2$ は明示的には書きませんでした。共分散行列は x(観測点) と x'(予測点) のあわせて $D + P$ 次元正方行列ですが、

$$\mathbf{\Sigma} = \begin{pmatrix} K_{XX} & K_{XX'} \\ K_{XX'}^T & K_{X'X'} \end{pmatrix}, \tag{7.9}$$

と 4 つのブロックに分割することが出来ます。K_{XX} は観測点間の $D \times D$ 相関行列、$K_{X'X'}$ は予測点同士の $P \times P$ 相関行列、$K_{XX'}$ は観測点と予測点の相関を表す $D \times P$ 行列です。

数学的には、共分散行列は半正定値性という性質を持ちます。実対称性行列とすると、固有値がすべて非負になるという性質です。$\mathbf{\Sigma}$, K_{XX}, $K_{X'X'}$, $K_{XX'}^T K_{XX}^{-1} K_{XX'}$ はいずれも半正定値性を満たします。これは多次元正規分布の条件付き確率などを考える際にも大事な性質になってきます。このあたりの詳細は PRML [6] の 2 章などを参照してください[*17]。

あとは興味のある値 $\boldsymbol{y}'$ の条件付き確率 $P(\boldsymbol{y}'|\boldsymbol{y})$ を計算して (これも P 次元正規分布になります)、そこからランダムに値をサンプルしてやれば、$\boldsymbol{y}'$ の分布が得られるというわけです。

[*17] 備忘録を兼ねてそのうちまとめるかもしれません

カーネルの選択に関する小話

多くの教科書では RBF カーネル (Mátern カーネルの $\nu = \infty$ に相当) が標準的とされる一方で、実際の応用では Mátern カーネル ($\nu = 3/2, 5/2$) が好まれる傾向があります。その理由として述べられるのは「RBF カーネルを採用することはガウス過程からサンプルされる関数の連続性として無限回微分可能を仮定することと等しいが、これは強すぎる制約である」というもので、色んな論文にも書いてあります。もちろんそうなのですが「RBF カーネルだと、考えたい点が密になると共分散行列が解析的には満たすはずの半正定値性が数値的に不安定になりやすい」という、より実用的な問題が実のところではないか、と個人的には思っています。手で書いたガウス過程の計算コードで RBF カーネルを使って値にノイズがない場合を計算してみるといくつかの場合で数値エラーを起こすはずです。もちろん x 軸のスケールを変えたり対角要素に微小な値を加えるなどの対処法がありますが、個人的には後者は後述の理由でまずやりませんし、前者をやる場合には対数で距離を定義してスケール依存性をハイパーパラメータに組み込むのが好みです。後者のいわゆる ϵ 処方の妥当性などの小話はこちらのページや GitHub を御覧ください。大雑把にまとめると、ある半正定値行列 A の逆行列 A^{-1} と、A に微小な対角項を足した $A + \epsilon I$ の逆行列 $(A + \epsilon I)^{-1}$ は、A が悪条件のとき、数値計算上で一桁も合っていない場合があるので危険、というお話です。このあたりのことは、精度保証付きの数値計算の専門家の先生にとっては当たり前の内容ですが、私のような GP のいわゆる practitioner や、データサイエンティスト・エンジニアには余り知られていないようですし、プログラミング言語の各種ライブラリの中でもかなり naive に扱われています。研究に応用した際に得たかなりテクニカルな知見ですが、誰かのためになればいいと思って残しておきます。

また、ガウス過程から生成されるサンプル関数の連続性や微分可能性に関して説明された日本語の文献というのがすぐに見つかりませんでした。個人的な興味からそのうちまとめるかもしれません。