THKO01

# 加速器運転における機械学習の応用

## Machine Learning Applications to Accelerator Operation

前坂 比呂和
理化学研究所 放射光科学研究センター
令和７年８月７日
第 22 回 日本加速器学会 年会

# もくじ

- はじめに
  - 近年の高性能・複雑な加速器の運転に関する諸課題
  - 物理モデルベース型 vs. データ駆動型 (機械学習)
  - ツール紹介 (Python, Jupyter notebook, etc.)
- 加速器性能の最適化
  - ベイズ最適化
  - 複数の目的関数の同時最適化
  - 強化学習
- 高効率・高次元ビーム診断
  - ベイズ型測定アルゴリズム
  - 生成系 AI による仮想ビーム診断
- 異常検知・故障予知
  - サイラトロンの異常検知の例
- まとめ
- コミュニティ・ワークショップの宣伝

# はじめに

- 加速器が大規模・高性能になるにつれ、制御点数や調整精度が年々高まっている。
- 常に高い性能を発揮するのが当たり前になってきた。
- 極めて高い稼働率も求められるようになっている。
- これまでの古典的な調整手法・運転手法では性能を維持するのが困難になりつつある。
- そこで、機械学習を応用した新しい手法が模索されている。

Generated by Gemini 2.5 Flash

Generated by Gemini 2.5 Flash

# 高性能・複雑な加速器の運転に関する諸課題

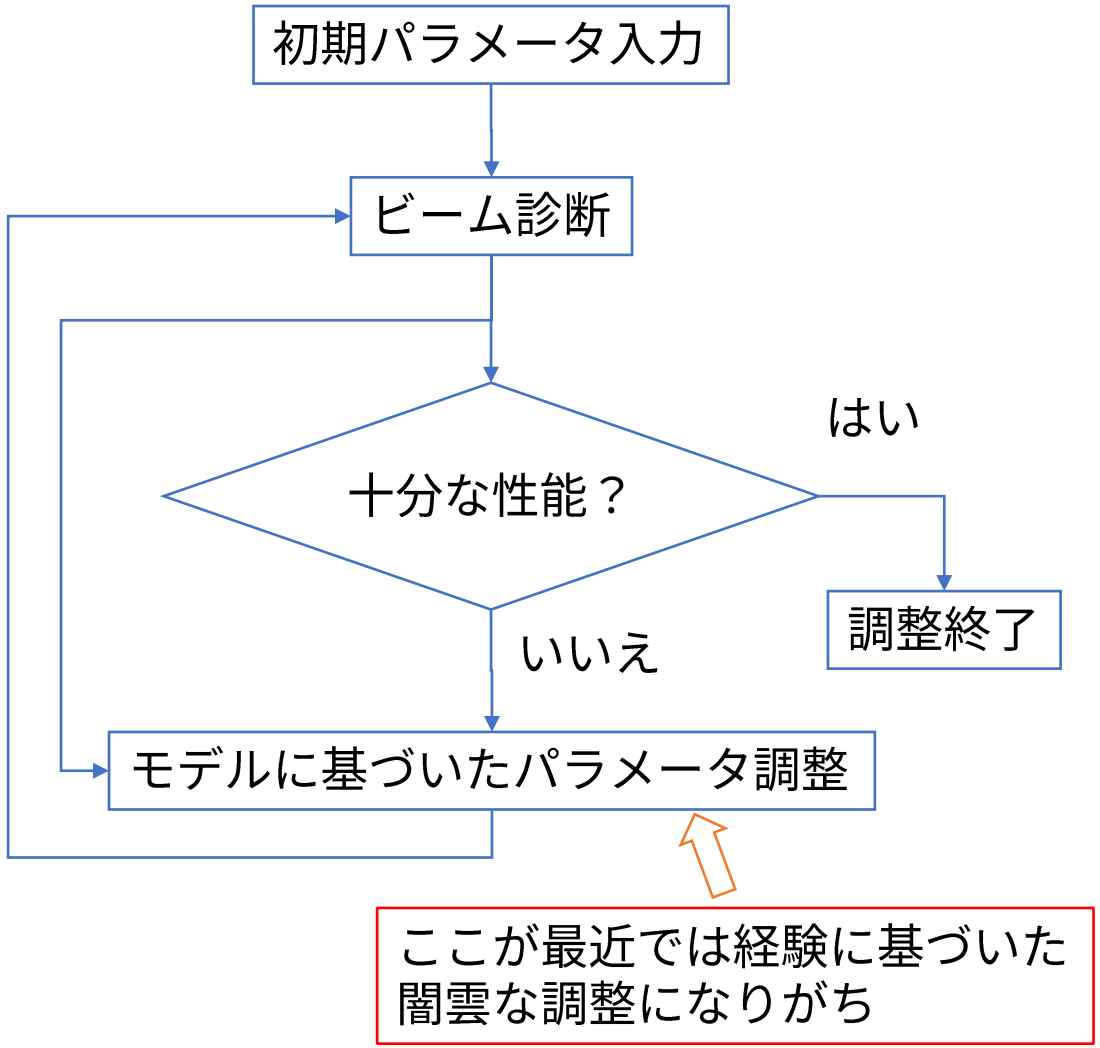
- 過去の加速器は物理的にも比較的単純なモデルで設計されており、ビームに対する応答がわかりやすかった。
- 物理モデルや加速器設計に基づいた調整方法でおおむね設計性能がえられることが多かった。
- システムが多少不安定でも許容できる余地があった。
- 制御やデータ収集においても、現場でしかできないことも多かった。

- 90年代以降の加速器では性能向上に伴い、制御項目が増え、遠隔制御・遠隔監視の必要性が高まった。
- 物理モデルも複雑となり、解析的な手法に加え、トラッキングシミュレーション等が必須となった。
- 構成要素数も増大し、機器の故障による運転停止や性能低下のリスクが高まった。
- それでも、制御室からの遠隔制御や遠隔監視を強化する程度でもなんとか成り立っていた。

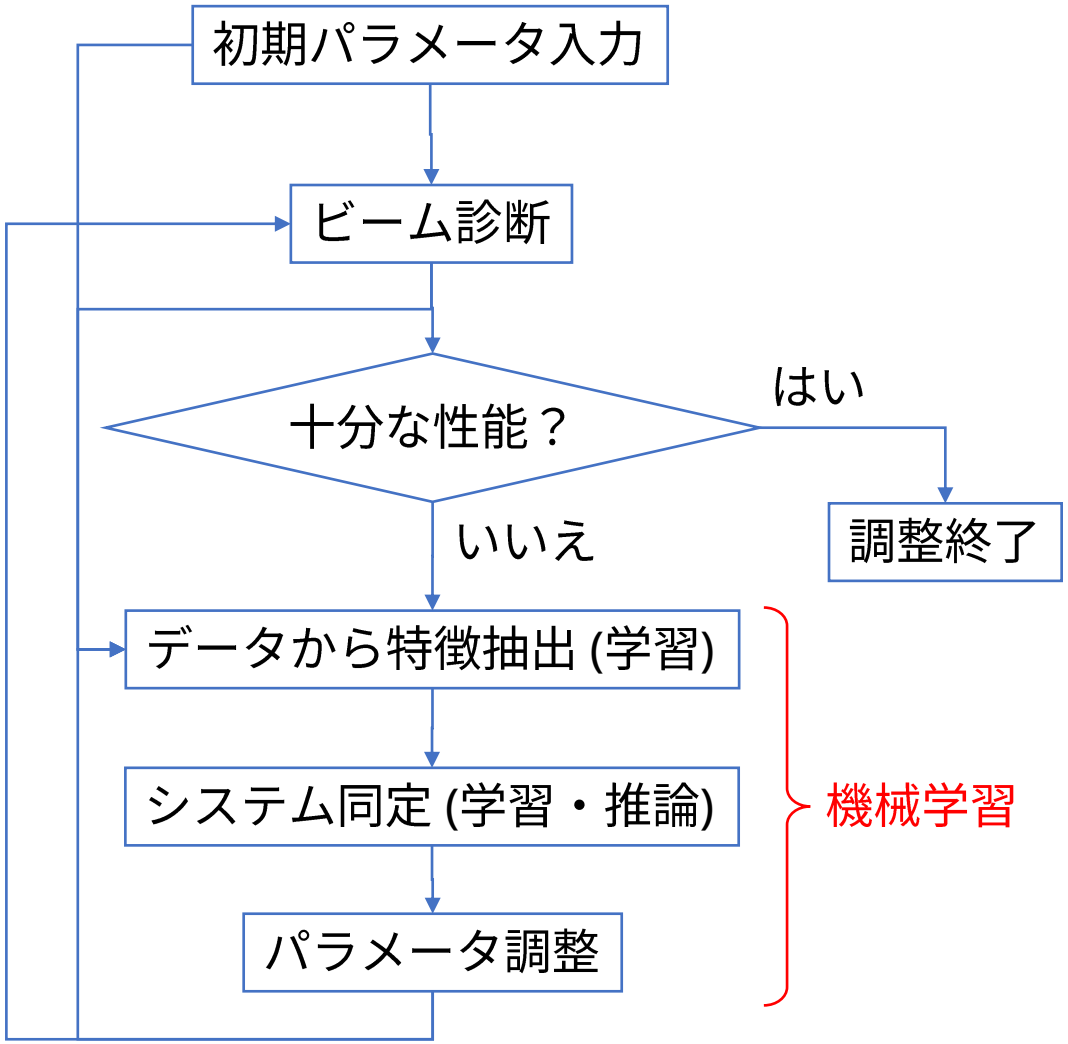
- 近年ではさらなる性能向上や技術的な成熟に伴い、ユーザーの満足度をさらに高める運転方法が求められるようになった。
- 物理モデルやシミュレーションに合わせるだけでは設計性能が得られにくくなっている。
- ある一定の性能から先は運転関係者による地道な調整等で少しずつ性能を上げていくことが常態化している。
- にもかかわらず、建設後のコミッショニングでは早く設計性能を達成しなければならない。
- 日々の運転でも瞬間最大に近い性能を常に出せることへの要求が高まっている。
- 機器の不具合等による加速器の停止や性能低下対する風当たりも厳しさを増している。
- 運転関係者による手動調整や定期監視等だけでは成り立たなくなりつつある。

# 物理モデルベース型 vs. データ駆動型 (機械学習)

## 物理モデルベース型の調整




## データ駆動型の調整 (機械学習)



# モデルベースか機械学習か (個人的見解)

- **問題が多少複雑だからと言って安易に機械学習に手を出すことには賛成しかねる。**
  - 加速器の挙動や荷電粒子ビームの運動は比較的物理モデルに乗りやすい。
  - 何か困難に直面したときは新しい物理が裏に隠れていることが多いため、その新しい物理モデルを探索する方がよっぽどまし。
  - フィードバック制御等はまずちゃんと制御理論を勉強すべし。
  - 人工知能 (AI)・機械学習 (ML) に染まってしまうとブラックボックス化して背景にある物理がおろそかになる恐れがある。
  - 実際、物理モデルを考えるより AI に放り込んだ方が楽なのに、という風な態度をとる人がまれにみられ、わたしはそういう考えを AI 原理主義と考えて注意するように心がけている。
- **モデルベースでできることをやりつくした者だけが機械学習の助けを借りられると心せよ。**
  - 機械学習がモデルベースより性能が劣ることは許されない。
  - 機械学習がポンコツな場合は使ってもらえるわけがない。
  - 機械学習を応用するならモデルベースを超える性能をたたき出し、それが日々の運転に使えるように仕立て上げる覚悟を持つべし。
- **深層学習 (Deep Neural Network) 系は最終手段と思え。**
  - 加速器のような物理モデルに乗りやすいシステムの場合は他の浅い部類の機械学習アルゴリズムが適していることが多々ある。

# 長期的な展望

- 加速器の運転において機械学習や人工知能に期待されている当面の課題の解決にめどがついてくれば、もう少し明るい未来を描いてもいいだろう。
- その展望のひとつに自動運転がまず考えられる。
- 熟練運転員が頭で考え、手を動かしてやってきた運転に関する多くの操作を人工知能ベースのアルゴリズムに任せられるシステム。
- どうしても人の操作が必要なときだけ運転員が行う。
- 機器の異常の兆候を的確に検知し、運転計画に穴をあけないように予防的メンテナンスの指針を示してくれるシステム。
- このレベルになってくると深層学習系のアルゴリズムが常に必要な状況になるだろう。

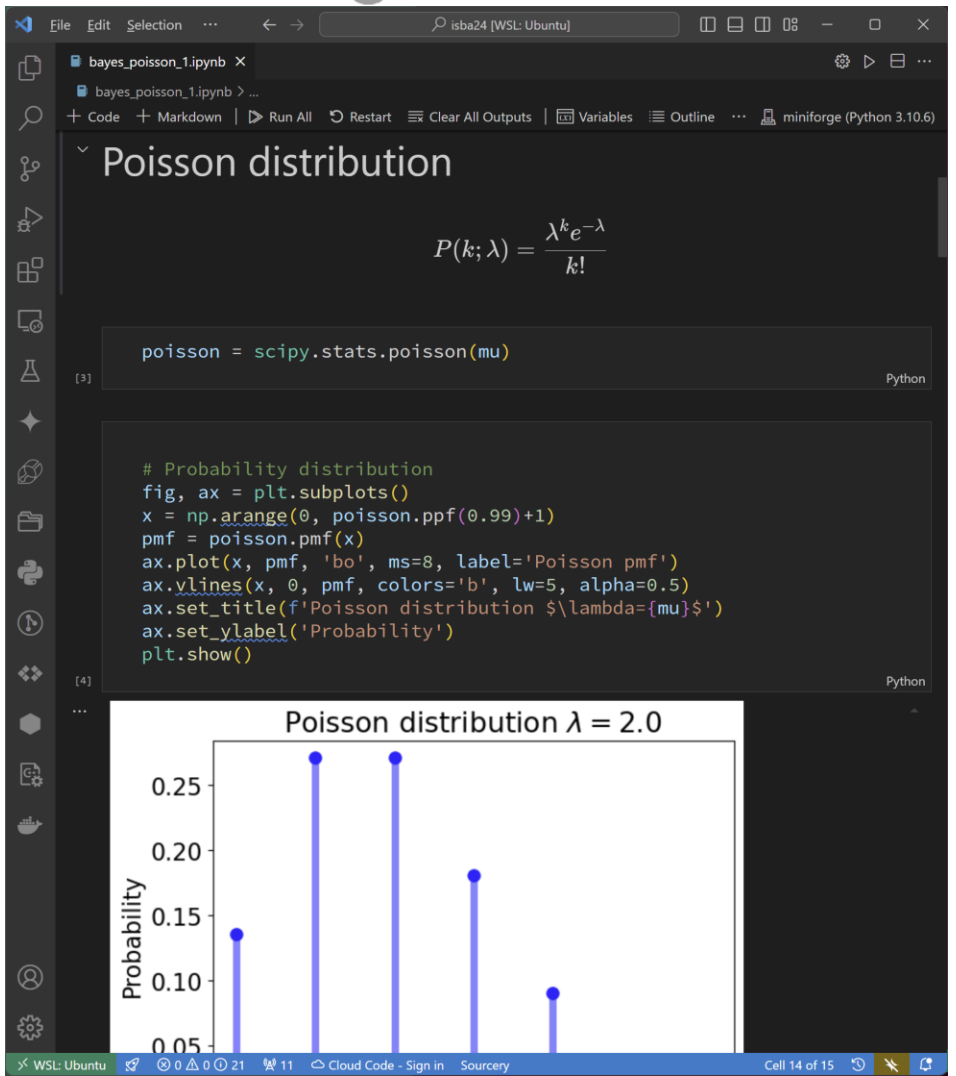
# ツール紹介 (Python)

- Python: インタプリタ言語のひとつ
- https://www.python.org/
- 科学計算ライブラリが充実


- **NumPy**: 基礎的な科学計算モジュール
  https://numpy.org/
- **SciPy**: 科学計算に有用な多くのアルゴリズム
  https://scipy.org/
- **Pandas**: データ処理や解析のためのモジュール
  https://pandas.pydata.org/
- **Matplotlib**: データ可視化ようモジュール（グラフ作成）
  https://matplotlib.org/
- **Scikit-learn**: 多彩な機械学習モジュール
  https://scikit-learn.org/
- **SymPy**: 記号ベースの数式処理モジュール (Mathematica 的)
  https://www.sympy.org/
- 他、多数

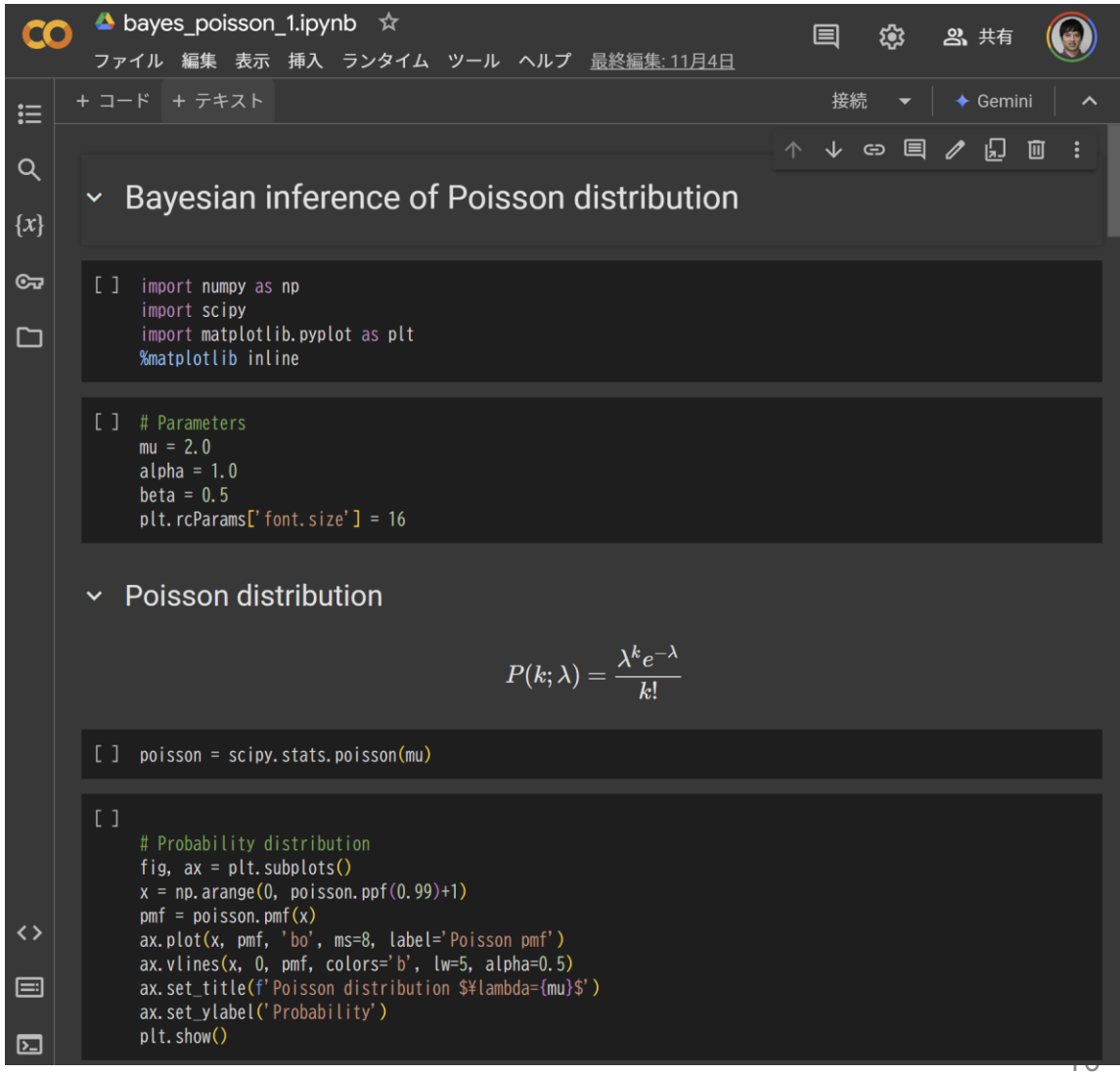
# Jupyter notebook

- セルにコードの断片を入力し、セルごとに実行することができる環境。
  - Python, Java, R, Julia, …
- ウェブブラウザ、または、Visual Studio Code 上で動作可能。
- Markdown テキストも書き込める。
  - LaTeX 数式も使える。
- 作成したノートブックは HTML や PDF に変換可能。
- トライ＆エラーの多いデータ解析などに大変有用。
- Jupyter 環境が構築できるパッケージ例
  - Anaconda (アカデミックを除き有料): https://www.anaconda.com/
  - Miniforge (無料): https://github.com/conda-forge/miniforge



# Google Colaboratory

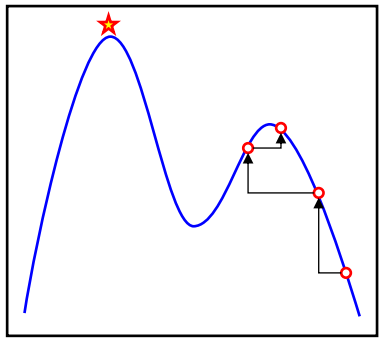
- クラウド上の Jupyter サービスのひとつ
  https://colab.research.google.com/
- ある程度無料で使用可能。
  - Google アカウントは必要。
  - 課金すれば多くの CPU, GPU, メモリが使用可。
- 環境を作らなくてもウェブブラウザがあれば使用できる。
- わたしの過去の例題が以下の講演スライドにあるので是非試してみてください。
  - 加速器・ビーム物理のワークショップ 2024
    http://xfel.riken.jp/workshop2024/program.html
  - International School on Beam Dynamics and Accelerator Technology 2024
    https://conference-indico.kek.jp/event/275/timetable/#20241107.detailed



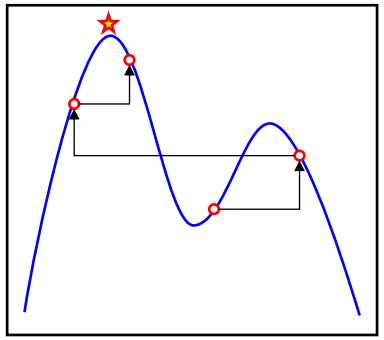
# 加速器性能の最適化

- データ駆動型の最適化手法は古典的な手法と機械学習的な手法に分けられる。
- 古典的な手法には勾配降下法やネルダー・ミード法などが挙げられる。
- これらの古典的な手法は局所最適値に陥りやすかったり、データの誤差やドリフトの影響を受けやすかったりするため、うまくいかないことがある。
- そこで、機械学習的な手法が模索されるようになった。
- 機械学習的な最適化では、設定パラメータに対する評価値の応答から特徴を抽出し、システムの応答関数を同定する (学習)。
- その応答関数をもとに、設定パラメータに対する応答を推論する。
- 応答がよさそうなパラメータに設定してデータを取得する。
- これを繰り返して最適値を得る。
- 機械学習的な手法にはベイズ最適化や強化学習などがある。
- ここではガウス過程最適化を中心に話を進める。

古典的手法

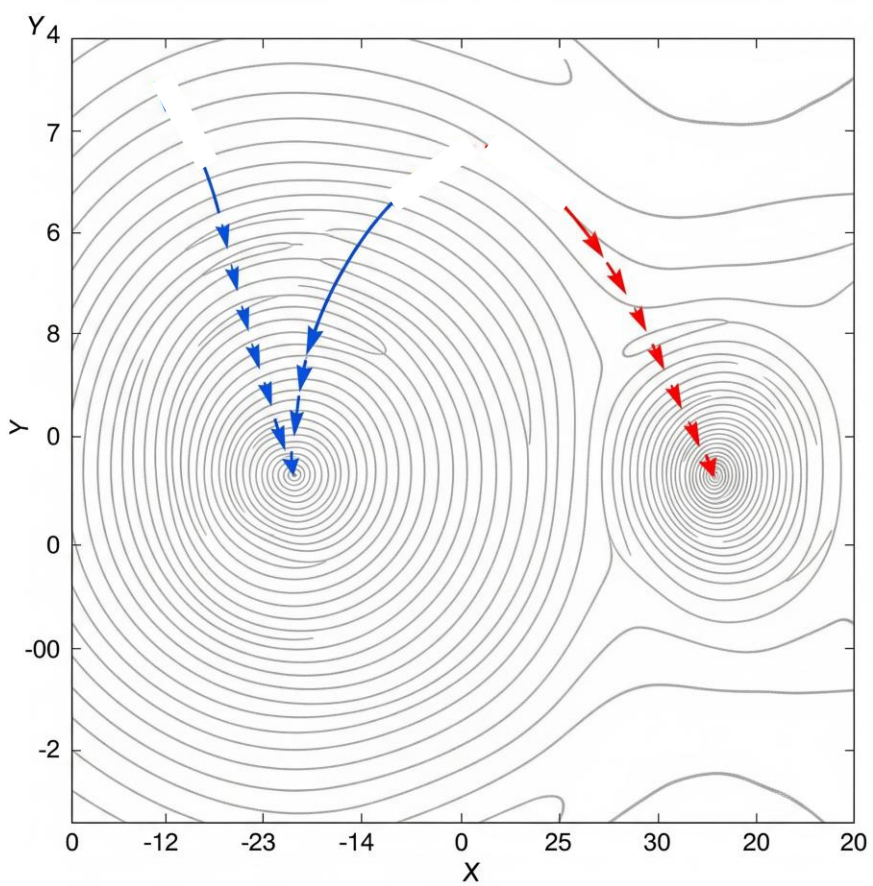


機械学習的手法



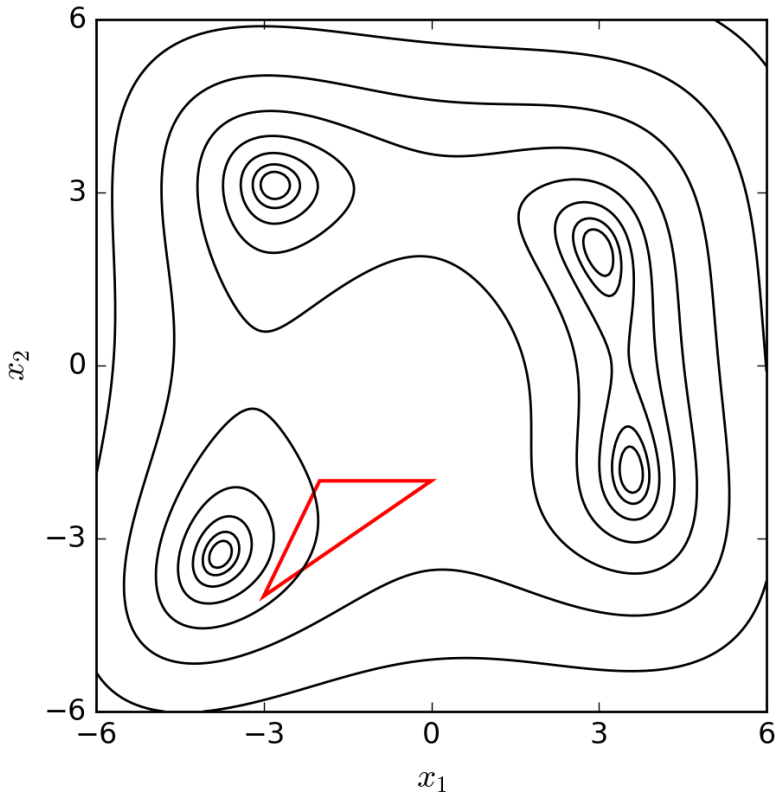
# 古典的な最適化例

**勾配降下法**
評価関数の勾配をもとに極値を
求めるアルゴリズム




Generated by Gemini 2.5 Pro + manual modification

**Nelder-Mead 法**
N+1 個の頂点からなる N 次元単体をアメーバの
ように動かしながら極値を探索するアルゴリズム
反射・膨張・収縮の３種類の更新方法を使い分ける。




Nicoguaro, CC BY 4.0 , via Wikimedia Commons
https://commons.wikimedia.org/wiki/File:Nelder-Mead_Himmelblau.gif

# ベイズ推定 (機械学習の第一歩)

- 加速器の運転において、評価値 $y$ は入力パラメータ $\boldsymbol{x}$ に対する目的関数 (Objective functoin) $f$ を用いて以下のように表すことができる。

$$y = f(\boldsymbol{x}; \boldsymbol{\theta})$$

$\boldsymbol{\theta}$ は目的関数を特徴づけるパラメータ。

- ベイズ推定の目標は、ベイズの定理を用いて $\boldsymbol{\theta}$ の確率分布を求めること、すなわち目的関数を同定することである。
- ベイズの定理：

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

$p(A)$ は事象 $A$ が起こる確率。
$p(A|B)$ は事象 $B$ のもとで事象 $A$ が起こる条件付確率。

- ベイズ推定では、得られた評価値データ $\boldsymbol{y}$ とそのときの入力パラメータ群 $X$ からベイズの定理を用いて $\boldsymbol{\theta}$ の確率分布を以下のように算出する。

$$p(\boldsymbol{\theta}|X, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}, X)\,p(\boldsymbol{\theta})}{p(\boldsymbol{y}|X)}$$

$X$ と $\boldsymbol{y}$ はそれぞれ $n$ 点のデータを取ってまとめたもの。
$X = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n),\ \boldsymbol{y} = (y_1, \cdots, y_n)$

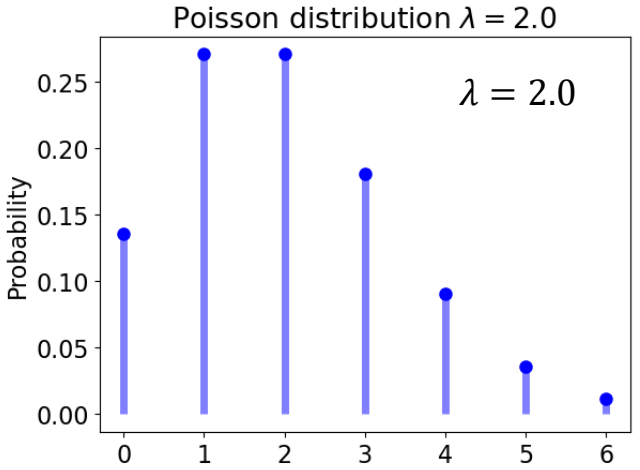
  - $p(\boldsymbol{\theta})$ は事前分布 (Prior distribution) で、 $\boldsymbol{\theta}$ のとりうる値をある程度反映した適当な確率分布である。(物理モデルに忠実である必要はない。)

- 目的関数の関数形 $f(\boldsymbol{x}; \boldsymbol{\theta})$ は $p(\boldsymbol{y}|\boldsymbol{\theta}, X)$ を計算するうえで事前にある程度決めておく必要がある。
- 分母の $p(\boldsymbol{y}|X)$ は周辺尤度 (Marginal likelihood) と呼ばれ、入力 $X$ のもとで $\boldsymbol{y}$ が得られる確率について、とりうるすべての $\boldsymbol{\theta}$ で積分したものである。

$$p(\boldsymbol{y}|X) = \iint p(\boldsymbol{y}|\boldsymbol{\theta}, X)\,p(\boldsymbol{\theta})\,d\boldsymbol{\theta}$$

- このようにして、データからベイズ推定を用いて事後分布 (Posterior distribution) $p(\boldsymbol{\theta}|X, \boldsymbol{y})$ を得ることができる。⟸ 学習
- この事後分布を使って任意の入力パラメータ $\hat{\boldsymbol{x}}$ に対する評価値の確率分布 $\hat{y} = f(\hat{\boldsymbol{x}}; \boldsymbol{\theta})$ を推定することができる。⟸ 推論
  - 通常この計算は解析的に行うことが難しいのがほとんどで、数値的に求めるのが普通である。

# ベイズ推定の例 (ポアソン分布)




- 目的関数をポアソン分布とし、そのパラメータ $\lambda$ をベイズ推定で求めてみよう。

$$P(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- パラメータ $\lambda$ の事前分布としてガンマ分布を使うのが便利であることが知られている。

$$f(\lambda;\alpha,\beta) = \frac{\beta^\alpha \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)}$$

- ひとつ目のデータ $k_1$ が得られた後の事後分布はベイズの定理を用いて次のように求められる。

$$p(\lambda|k_1) = \frac{P(k_1;\lambda)\, f(\lambda;\alpha,\beta)}{p(k_1)} = \frac{\beta^\alpha \lambda^{\alpha-1+k_1} e^{-(\beta+1)\lambda}}{k_1!\,\Gamma(\alpha)\, p(k_1)}$$

- 周辺尤度 $p(k_1)$ は $k_1$ の確率分布を $\lambda$ で積分して次のように得られる。

$$p(k_1) = \int_0^\infty P(k_1;\lambda)\, f(\lambda;\alpha,\beta)\, d\lambda = \frac{\beta^\alpha}{k_1!\,\Gamma(\alpha)} \int_0^\infty \lambda^{\alpha-1+k_1} e^{-(\beta+1)\lambda} d\lambda = \frac{\beta^\alpha \Gamma(\alpha+k_1)}{k_1!\,(\beta+1)^{\alpha+k_1}\Gamma(\alpha)}$$

- よって、事後分布 $p(\lambda|k_1)$ は以下のように求まる。

$$P(\lambda|k_1) = \frac{(\beta+1)^{\alpha+k_1} \lambda^{\alpha-1+k_1} e^{-(\beta+1)\lambda}}{\Gamma(\alpha+k_1)} = f(\lambda;\alpha+k_1,\beta+1)$$

- 事後分布もガンマ分布で、データによりパラメータ $\alpha,\beta$ が更新されていることがわかる。
- 事前分布にガンマ分布を選んだ理由は、ポアソン分布モデルの場合、事後分布もガンマ分布になるためで、このような分布を共役事前分布と呼ぶ。

ガンマ関数

$$\Gamma(x) = \int_0^\infty t^{x-1} e^t dt\,, \Gamma(n) = (n-1)!$$

# ベイズ推定の例 (ポアソン分布) つづき

- そして、$n$ 個のデータ $\boldsymbol{k} = (k_1, \cdots, k_n)$ が得られると、$\lambda$ の事後分布 $p(\lambda|\boldsymbol{k})$ は以下のようになる。

$$p(\lambda|\boldsymbol{k}) = \frac{P(\boldsymbol{k};\lambda)\, f(\lambda;\alpha,\beta)}{p(\boldsymbol{k})} = \left(\prod_{m=1}^{n} \frac{\lambda^{k_m} e^{-\lambda}}{k_m!}\right) \frac{\beta^{\alpha}\lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)\, p(\boldsymbol{k})} = \frac{\beta^{\alpha}\ \lambda^{\alpha-1+\sum_{m=1}^{n} k_m}\ e^{-(\beta+n)\lambda}}{\Gamma(\alpha)\, p(\boldsymbol{k}) \prod_{m=1}^{n} k_m!}$$

- 周辺尤度 $p(\boldsymbol{k})$ は次のように計算できる。

$$p(\boldsymbol{k}) = \int_0^{\infty} P(\boldsymbol{k};\lambda)\, f(\lambda;\alpha,\beta)\, d\lambda = \frac{\beta^{\alpha}}{\Gamma(\alpha)} \int_0^{\infty} \left(\prod_{m=1}^{n} \frac{\lambda^{k_m} e^{-\lambda}}{k_m!}\right) \lambda^{\alpha-1} e^{-\beta\lambda} d\lambda$$

$$= \frac{\beta^{\alpha}}{\Gamma(\alpha) \prod_{m=1}^{n} k_m!} \int_0^{\infty} \lambda^{\alpha-1+\sum_{m=1}^{n} k_m} e^{-(\beta+n)\lambda} d\lambda = \frac{\beta^{\alpha}\Gamma(\alpha + \sum_{m=1}^{n} k_m)}{(\beta+n)^{\alpha+\sum_{m=1}^{n} k_m}\Gamma(\alpha) \prod_{m=1}^{n} k_m!}$$

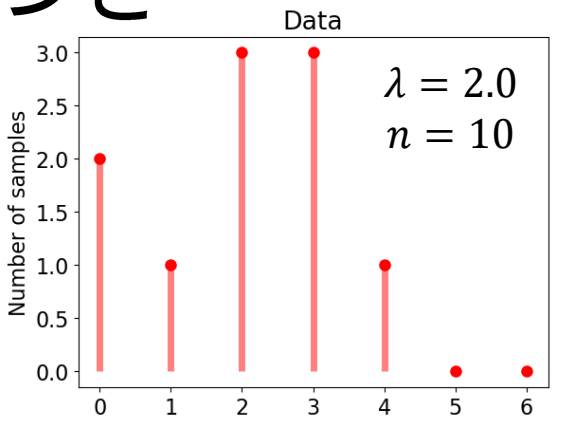
- よって、事後分布 $p(\lambda|\boldsymbol{k})$ は以下のようなガンマ分布となる。

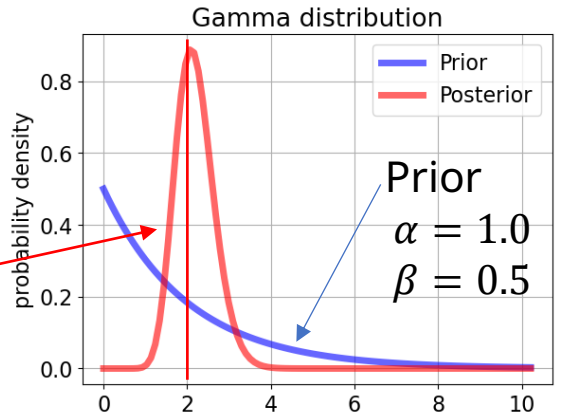
$$p(\lambda|\boldsymbol{k}) = \frac{(\beta+n)^{\alpha+\sum_{m=1}^{n} k_m} \lambda^{\alpha-1+\sum_{m=1}^{n} k_m} e^{-(\beta+n)\lambda}}{\Gamma(\alpha + \sum_{m=1}^{n} k_m)} = f\left(\lambda; \alpha + \sum_{m=1}^{n} k_m, \beta + n\right)$$

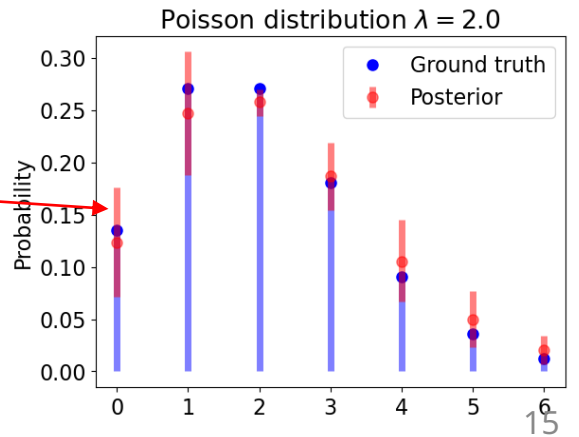
- このように、**事後分布はデータにより更新 (学習) され、真値の周りの細い分布となる。**
- 目的関数 (ポアソン分布) の振る舞いは以下の積分で求められる。(推論)

$$p(\hat{k}) = \int_0^{\infty} P(\hat{k};\lambda)\, p(\lambda|\boldsymbol{k})\, d\lambda = \int_0^{\infty} \frac{\lambda^{\hat{k}} e^{-\lambda}}{\hat{k}!} f\left(\lambda; \alpha + \sum_{m=1}^{n} k_m, \beta + n\right) d\lambda$$

  - この積分は解析的には解きづらいので数値計算で求める。
- ここまでできれば、単なるベイズ推定から、ベイズ学習と呼んでもいいレベルであろう。



# ガウス過程

Roman Garnett, “Bayesian Optimization” (2023). https://bayesoptbook.com/
Ryan Roussel *et al.*, Phys. Rev. Accel. Beams **27**, 084801 (2024).

- ガウス過程 (Gaussian process) とは、すべての確率変数が多変量ガウス分布に従う確率過程のことである。
- 最適化問題では目的関数 $f\colon \mathcal{X} \to \mathbb{R}$ として以下のようなものを考えるのが一般的である。

$$\phi = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathcal{X}, \qquad \boldsymbol{x} = (x_1, \cdots, x_n)$$

- 関数 $f$ のガウス過程を次のように表記する。

$$p(f) = \mathcal{GP}(f; \mu, K)$$

  - $\mu\colon \mathcal{X} \to \mathbb{R}$ : 期待値関数。 $\mu(\boldsymbol{x}) = \mathbb{E}(\phi|\boldsymbol{x})$
  - $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ : 共分散関数 (カーネル)。 $K(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{cov}(\phi|\boldsymbol{x}, \phi'|\boldsymbol{x}')$

- 関数 $f$ がガウス過程に従うとき、$(\phi, \boldsymbol{x})$ の有限サンプルは多変量ガウス分布に従う。

$$\boldsymbol{\phi} = (\phi_1, \cdots, \phi_N) = f(\mathrm{X}) = \big(f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_N)\big), \qquad \mathrm{X} \subset \mathcal{X}, \qquad \boldsymbol{x} \in \mathrm{X}$$

$$p(\boldsymbol{\phi}|\mathrm{X}) = \mathcal{N}(\boldsymbol{\phi}; \boldsymbol{\mu}, \Sigma), \qquad \boldsymbol{\mu} = \mathbb{E}(\boldsymbol{\phi}|\mathrm{X}), \qquad \Sigma = \mathrm{cov}(\boldsymbol{\phi}|\boldsymbol{x}) = K(\mathrm{X}, \mathrm{X})$$

$$\mathcal{N}(\boldsymbol{\phi}; \boldsymbol{\mu}, \Sigma) \equiv \frac{\exp\left[-\frac{1}{2}(\boldsymbol{\phi} - \boldsymbol{\mu})^{\mathsf{T}} \Sigma^{-1} (\boldsymbol{\phi} - \boldsymbol{\mu})\right]}{\sqrt{(2\pi)^N |\Sigma|}}$$

  - ここに、$K(\mathrm{X}, \mathrm{X})$ はある 2 点の共分散関数を行列にまとめたものである。

$$K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

  - これは X のグラム行列 (Gram matrix) とか、カーネルとかと呼ばれる。

- この強力な条件は学習や推論の際の計算を大いに簡単にしてくれる。
- ガウス過程最適化の目的は、$f$ の確率分布を推定 (学習) し、大域的な極値（最大値または最小値）を目指して繰り返し探索し、それを素早く見つけることである。

# ガウス過程回帰 (Gaussian process regression)

- ガウス過程回帰は、すべての点がガウス過程に従うと仮定して、有限のデータ点から関数形を推定するアルゴリズムである。
- いくつかのデータ点 $\mathcal{D} = (\boldsymbol{y}_\mathcal{D}, \mathrm{X}_\mathcal{D})$ が得られたとき、そのデータ点と事前分布の同時分布もガウス過程となる。

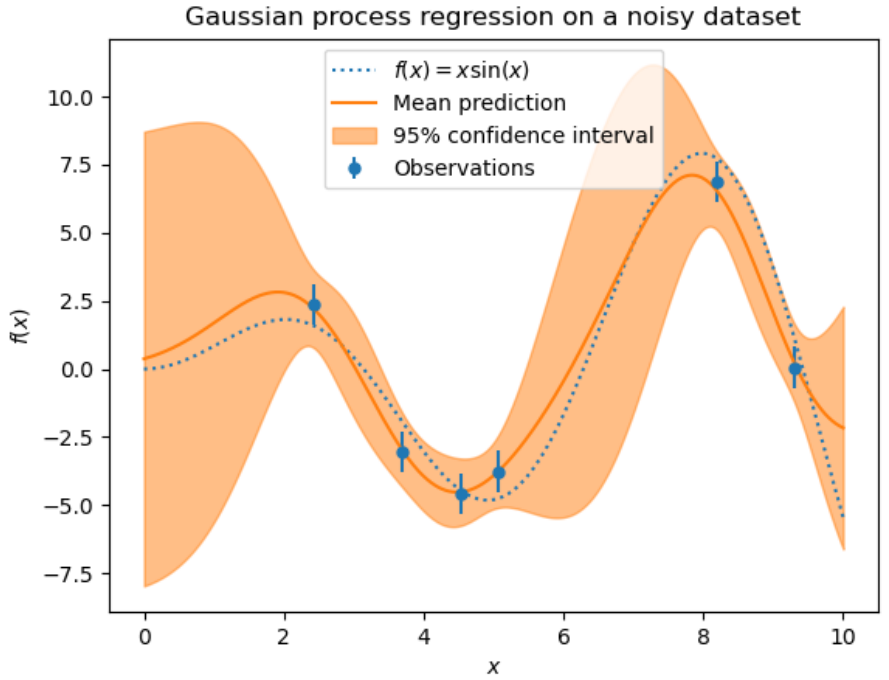
$$p(f, \mathcal{D}) = \mathcal{GP}\left(\begin{bmatrix} f \\ \mathcal{D} \end{bmatrix}; \begin{bmatrix} \mu_0 \\ \boldsymbol{y}_\mathcal{D} \end{bmatrix}, \begin{bmatrix} K_0 & \boldsymbol{\kappa}^\mathrm{T} \\ \boldsymbol{\kappa} & \mathbf{C} \end{bmatrix}\right)$$

  - ここに、$(\mu_0, K_0)$ は事前分布の期待値関数と共分散関数である。
$$p(f) = \mathcal{GP}(f; \mu_0, K_0)$$
  - データ点自体も何らかのガウス過程に従うものと仮定する。
$$p(\mathcal{D}) = \mathcal{GP}(\mathcal{D}; \boldsymbol{y}_\mathcal{D}, \mathbf{C})$$
  - また、$\boldsymbol{\kappa}$ は $\mathcal{D}$ と $f$ の相互共分散関数 (cross-covariance function) である。
$$\boldsymbol{\kappa} = \mathrm{cov}(\boldsymbol{y}_\mathcal{D} | \mathrm{X}_\mathcal{D}, \phi | \boldsymbol{x}) = K(\mathrm{X}_\mathcal{D}, \boldsymbol{x}), \qquad \phi = f(\boldsymbol{x})$$
  - 関数 $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ はガウス過程のカーネルと呼ばれる。
  - 各データ点はガウス過程に従うので $\mathbf{C}$ も同様のカーネルとなる。
$$\phi_1 = f(\boldsymbol{x}_1), \qquad \phi_2 = f(\boldsymbol{x}_2) \quad \Rightarrow \quad p(\phi_1, \phi_2) = \mathcal{N}\left(\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}; \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) \end{bmatrix}\right)$$

- 事後分布はベイズの定理から以下のように求められる。

$$p(f|\mathcal{D}) = \mathcal{GP}(f; \mu_\mathcal{D}, K_\mathcal{D}) = \frac{p(\mathcal{D}|f)\, p(f)}{p(\mathcal{D})} = \frac{p(f, \mathcal{D})}{p(\mathcal{D})} = \frac{\mathcal{GP}\left(\begin{bmatrix} f \\ \mathcal{D} \end{bmatrix}; \begin{bmatrix} \mu_0 \\ \boldsymbol{y}_\mathcal{D} \end{bmatrix}, \begin{bmatrix} K_0 & \boldsymbol{\kappa}^\mathrm{T} \\ \boldsymbol{\kappa} & \mathbf{C} \end{bmatrix}\right)}{\mathcal{GP}(\mathcal{D}; \boldsymbol{y}_\mathcal{D}, \mathbf{C})}$$

- 事後分布 $p(f|\mathcal{D})$ の $f(\boldsymbol{x})$ の入力 $\boldsymbol{x}$ は任意に選ぶことが可能。
- したがって、ガウス過程回帰は未知の関数形を推定するのに非常に便利である。
- 期待値だけでなくその分散・共分散が得られることも大きな特長。




https://scikit-learn.org/1.5/modules/gaussian_process.html

# ガウス過程回帰の導出 (1)

- 一次元関数の簡単な例。
- 事前分布: 期待値 $\mu_0 = 0$, 分散は $\sigma_0^2$

$$p(\phi) = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left(-\frac{\phi^2}{2\sigma_0^2}\right)$$

- まず、１点だけのデータ $\mathcal{D} = (y_\mathcal{D}, \boldsymbol{x}_\mathcal{D})$ が誤差の分散 $\sigma_D^2$ で取得されたとする。
- $\boldsymbol{x} = \boldsymbol{x}_\mathcal{D}$ での事後分布の期待値と分散は事前分布とデータの重み付き平均と一致すべきなので、

$$\mathbb{E}(\phi|\mathcal{D}, \boldsymbol{x}_\mathcal{D}) = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_\mathcal{D}^2} y_\mathcal{D}, \qquad \mathrm{var}(\phi|\mathcal{D}, \boldsymbol{x}_\mathcal{D}) = \frac{\sigma_0^2 \sigma_\mathcal{D}^2}{\sigma_0^2 + \sigma_\mathcal{D}^2}$$

- カーネル関数としてよく採用されるのが動径基底関数 (radial basis function) で、
  遠い点同士の相関は小さくなるべき、という仮定に基づいたものである。

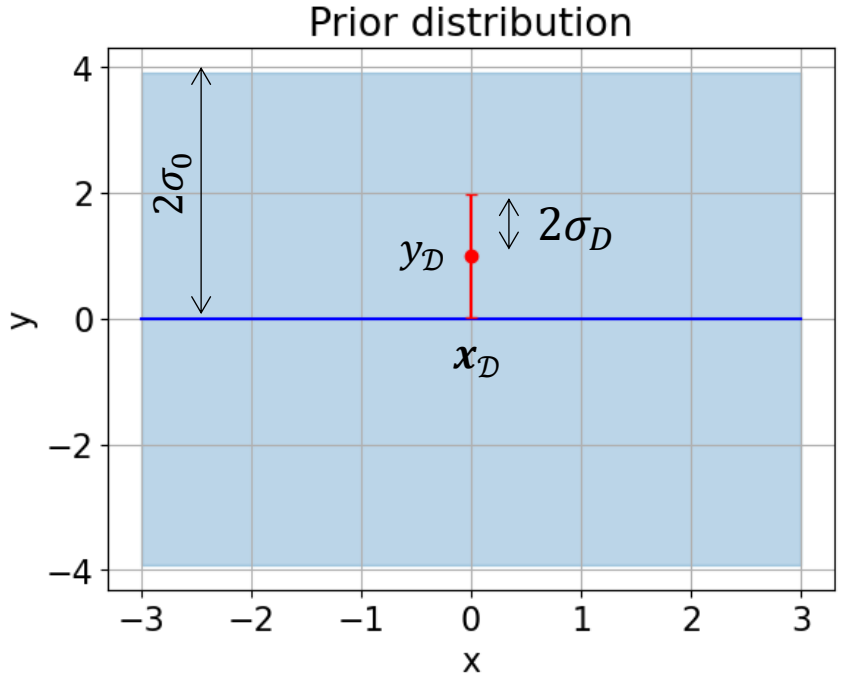
$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) \propto \exp\left(-\frac{1}{2}|\boldsymbol{x}_1 - \boldsymbol{x}_2|^2\right)$$

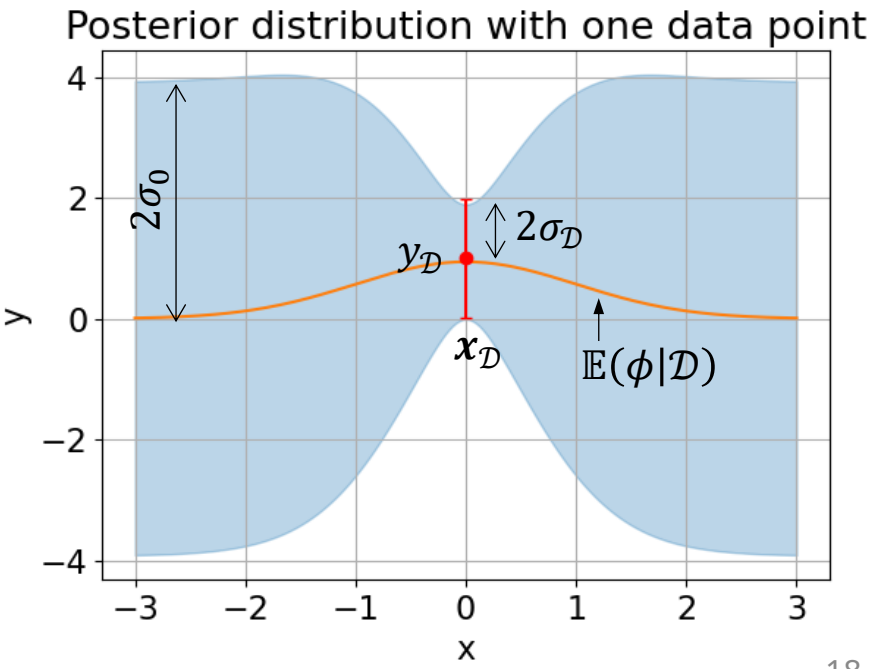
- このことから、事後分布の期待値と分散は以下のようになるべきことが想像できる。

$$\mathbb{E}(\phi|\mathcal{D}) = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_\mathcal{D}^2} y_\mathcal{D} \exp\left(-\frac{1}{2}|\boldsymbol{x} - \boldsymbol{x}_\mathcal{D}|^2\right), \qquad \mathrm{var}(\phi|\mathcal{D}) = \sigma_0^2 \left[1 - \frac{\sigma_0^2}{\sigma_0^2 + \sigma_\mathcal{D}^2} \exp(-|\boldsymbol{x} - \boldsymbol{x}_\mathcal{D}|^2)\right]$$

- こうなるような事後分布は以下の形でなければならない。

$$p(\phi|\mathcal{D}) = \frac{1}{\sqrt{2\pi\sigma_0^2\left[1 - \frac{\sigma_0^2}{\sigma_0^2 + \sigma_\mathcal{D}^2} \exp(-|\boldsymbol{x} - \boldsymbol{x}_\mathcal{D}|^2)\right]}} \exp\left(-\frac{\left[\phi - \frac{\sigma_0^2}{\sigma_0^2 + \sigma_\mathcal{D}^2} y_\mathcal{D} \exp\left(-\frac{1}{2}|\boldsymbol{x} - \boldsymbol{x}_\mathcal{D}|^2\right)\right]^2}{2\sigma_0^2\left[1 - \frac{\sigma_0^2}{\sigma_0^2 + \sigma_\mathcal{D}^2} \exp(-|\boldsymbol{x} - \boldsymbol{x}_\mathcal{D}|^2)\right]}\right)$$



# ガウス過程回帰の導出 (2)

- さらにデータ点を増やしてみよう。

$$\mathcal{D} = (\boldsymbol{y}_\mathcal{D}, \mathrm{X}_\mathcal{D}), \qquad \mathrm{X}_\mathcal{D} = \left(\boldsymbol{x}_{\mathcal{D},1}, \cdots, \boldsymbol{x}_{\mathcal{D},N_\mathcal{D}}\right), \qquad \boldsymbol{y}_\mathcal{D} = \left(y_{\mathcal{D},1}, \cdots, y_{\mathcal{D},N_\mathcal{D}}\right)$$

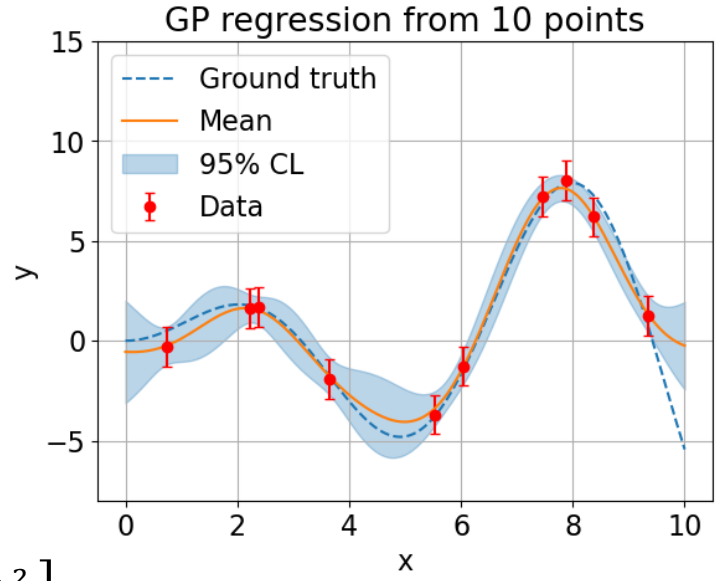
- 事前分布とデータ点の同時分布は以下のように書ける。

$$p(\phi, \mathcal{D}) = \mathcal{N}\left(\begin{bmatrix}\phi \\ \boldsymbol{y}_\mathcal{D}\end{bmatrix}; \begin{bmatrix}0 \\ 0\end{bmatrix}, \begin{bmatrix}\sigma_0^2 & \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T} \\ \boldsymbol{\kappa}_\mathcal{D} & \boldsymbol{\Sigma}_\mathcal{D}\end{bmatrix}\right) = \frac{1}{\sqrt{(2\pi)^{N_\mathcal{D}+1}\left|\sigma_0^2\boldsymbol{\Sigma}_\mathcal{D} - \boldsymbol{\kappa}_\mathcal{D}\boldsymbol{\kappa}_\mathcal{D}^\mathrm{T}\right|}} \exp\left[-\frac{1}{2}\begin{pmatrix}\phi & \boldsymbol{y}_\mathcal{D}^\mathrm{T}\end{pmatrix}\begin{pmatrix}\sigma_0^2 & \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T} \\ \boldsymbol{\kappa}_\mathcal{D} & \boldsymbol{\Sigma}_\mathcal{D}\end{pmatrix}^{-1}\begin{pmatrix}\phi \\ \boldsymbol{y}_\mathcal{D}\end{pmatrix}\right]$$

$$\boldsymbol{\kappa}_\mathcal{D} = \left[K\left(\boldsymbol{x}, \boldsymbol{x}_{\mathcal{D},1}\right), \cdots, K\left(\boldsymbol{x}, \boldsymbol{x}_{\mathcal{D},N_\mathcal{D}}\right)\right]$$

$$\boldsymbol{\Sigma}_\mathcal{D} = \begin{pmatrix} K\left(\boldsymbol{x}_{\mathcal{D},1}, \boldsymbol{x}_{\mathcal{D},1}\right) + \sigma_\mathcal{D}^2 & K\left(\boldsymbol{x}_{\mathcal{D},1}, \boldsymbol{x}_{\mathcal{D},2}\right) & \cdots & K\left(\boldsymbol{x}_{\mathcal{D},1}, \boldsymbol{x}_{\mathcal{D},N_\mathcal{D}}\right) \\ K\left(\boldsymbol{x}_{\mathcal{D},2}, \boldsymbol{x}_{\mathcal{D},1}\right) & K\left(\boldsymbol{x}_{\mathcal{D},2}, \boldsymbol{x}_{\mathcal{D},2}\right) + \sigma_\mathcal{D}^2 & \cdots & K\left(\boldsymbol{x}_{\mathcal{D},2}, \boldsymbol{x}_{\mathcal{D},N_\mathcal{D}}\right) \\ \vdots & \vdots & \ddots & \vdots \\ K\left(\boldsymbol{x}_{\mathcal{D},N_\mathcal{D}}, \boldsymbol{x}_{\mathcal{D},1}\right) & K\left(\boldsymbol{x}_{\mathcal{D},N_\mathcal{D}}, \boldsymbol{x}_{\mathcal{D},2}\right) & \cdots & K\left(\boldsymbol{x}_{\mathcal{D},N_\mathcal{D}}, \boldsymbol{x}_{\mathcal{D},N_\mathcal{D}}\right) + \sigma_\mathcal{D}^2 \end{pmatrix}$$

$$K = (\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma_0^2 \exp\left(-\frac{1}{2}|\boldsymbol{x}_1 - \boldsymbol{x}_2|^2\right)$$




- データ点同士も同じカーネルによる相関を持つものとする。
- 各データ点の誤差は独立と考えて対角成分に加えてある。
- この事後分布は以下のようになる。

$$p(\phi|\mathcal{D}) = \mathcal{N}\left(\phi; \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T}\boldsymbol{\Sigma}_\mathcal{D}^{-1}\boldsymbol{y}_\mathcal{D}, \sigma_0^2 - \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T}\boldsymbol{\Sigma}_\mathcal{D}^{-1}\boldsymbol{\kappa}_\mathcal{D}\right) = \frac{1}{\sqrt{2\pi\left|\sigma_0^2 - \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T}\boldsymbol{\Sigma}_\mathcal{D}^{-1}\boldsymbol{\kappa}_\mathcal{D}\right|}} \exp\left[-\frac{\left(\phi - \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T}\boldsymbol{\Sigma}_\mathcal{D}^{-1}\boldsymbol{y}_\mathcal{D}\right)^2}{2\left(\sigma_0^2 - \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T}\boldsymbol{\Sigma}_\mathcal{D}^{-1}\boldsymbol{\kappa}_\mathcal{D}\right)}\right]$$

- この期待値と分散は以下のとおりである。

$$\mathbb{E}(\phi|\mathcal{D}) = \mu_f(\boldsymbol{x}) = \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T}\boldsymbol{\Sigma}_\mathcal{D}^{-1}\boldsymbol{y}_\mathcal{D}, \qquad \mathrm{var}(\phi|\mathcal{D}) = V_f(\boldsymbol{x}) = \sigma_0^2 - \boldsymbol{\kappa}_\mathcal{D}^\mathrm{T}\boldsymbol{\Sigma}_\mathcal{D}^{-1}\boldsymbol{\kappa}_\mathcal{D}$$

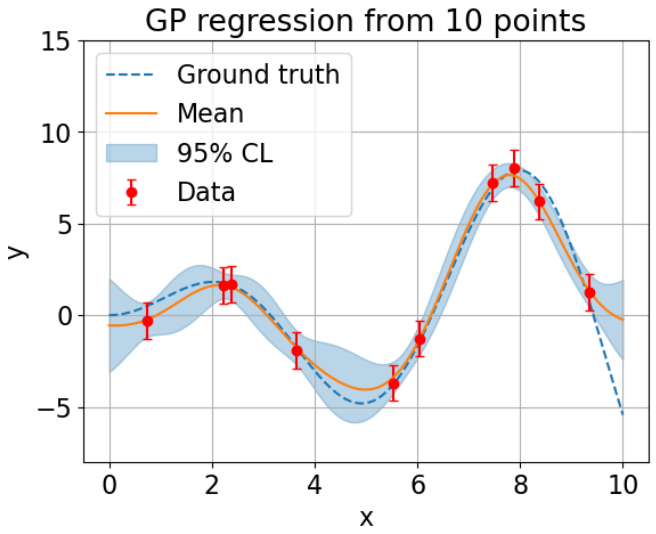
- このように、ガウス過程の仮定のもと、カーネル関数を決めるだけで、得られたデータ点から元の関数形を忠実に、分散込みで推定できることがわかる。

# ガウス過程回帰の計算

- 期待値や分散を多数の点で計算する場合はカーネル値の行列を作成して計算するのが効率的。
- ここでは事前分布も $\mathbf{K}(\mathrm{X},\mathrm{X})$ と表記する。

$$\mathbf{K}(\mathrm{X},\mathrm{X}) = \begin{pmatrix} K(\boldsymbol{x}_1,\boldsymbol{x}_1) & K(\boldsymbol{x}_1,\boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ K(\boldsymbol{x}_2,\boldsymbol{x}_1) & K(\boldsymbol{x}_2,\boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2,\boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_N,\boldsymbol{x}_1) & K(\boldsymbol{x}_N,\boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{pmatrix}, \qquad \mathbf{K}_{\mathcal{D}}(\mathrm{X}_{\mathcal{D}},\mathrm{X}) = \begin{pmatrix} K(\boldsymbol{x}_{\mathcal{D},1},\boldsymbol{x}_1) & K(\boldsymbol{x}_{\mathcal{D},1},\boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_{\mathcal{D},1},\boldsymbol{x}_N) \\ K(\boldsymbol{x}_{\mathcal{D},2},\boldsymbol{x}_1) & K(\boldsymbol{x}_{\mathcal{D},2},\boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_{\mathcal{D},2},\boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_{\mathcal{D},N_{\mathcal{D}}},\boldsymbol{x}_1) & K(\boldsymbol{x}_{\mathcal{D},N_{\mathcal{D}}},\boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_{\mathcal{D},N_{\mathcal{D}}},\boldsymbol{x}_N) \end{pmatrix},$$

$$\mathrm{X} = (\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N), \qquad \mathrm{X}_{\mathcal{D}} = (\boldsymbol{x}_{\mathcal{D},1},\boldsymbol{x}_{\mathcal{D},2},\cdots,\boldsymbol{x}_{\mathcal{D},N_{\mathcal{D}}}), \qquad K(x_a,x_b) \propto \exp\left(-\frac{1}{2}|\boldsymbol{x}_a-\boldsymbol{x}_b|^2\right)$$



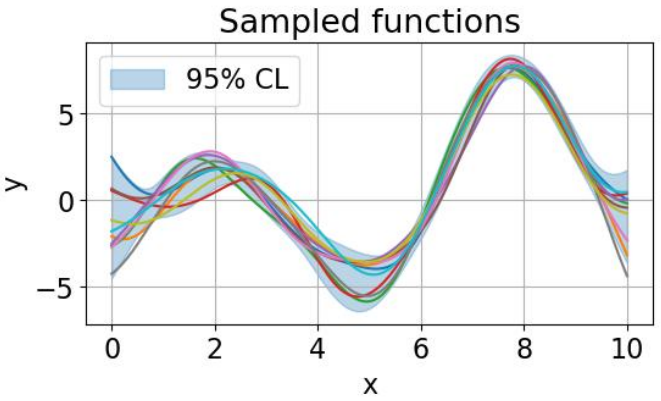

- 事前分布とデータ点の同時分布は以下のように書ける。

$$p(\boldsymbol{\phi},\mathcal{D}|\mathrm{X},\mathrm{X}_{\mathcal{D}}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\phi}\\ \boldsymbol{y}_{\mathcal{D}}\end{bmatrix};\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}\mathbf{K} & \mathbf{K}_{\mathcal{D}}^{\top}\\ \mathbf{K}_{\mathcal{D}} & \boldsymbol{\Sigma}_{\mathcal{D}}\end{bmatrix}\right) = \frac{\exp\left[-\frac{1}{2}\begin{pmatrix}\boldsymbol{\phi}^{\top} & \boldsymbol{y}_{\mathcal{D}}^{\top}\end{pmatrix}\begin{pmatrix}\mathbf{K} & \mathbf{K}_{\mathcal{D}}^{\top}\\ \mathbf{K}_{\mathcal{D}} & \boldsymbol{\Sigma}_{\mathcal{D}}\end{pmatrix}^{-1}\begin{pmatrix}\boldsymbol{\phi}\\ \boldsymbol{y}_{\mathcal{D}}\end{pmatrix}\right]}{\sqrt{(2\pi)^{N_{\mathcal{D}}+N}|\boldsymbol{\Sigma}_{\mathcal{D}}|\left|\mathbf{K}-\mathbf{K}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\mathbf{K}_{\mathcal{D}}\right|}}$$

- 事後分布は以下のように得られる。

$$p(\boldsymbol{\phi}|\mathcal{D},\mathrm{X},\mathrm{X}_{\mathcal{D}}) = \frac{p(\boldsymbol{\phi},\mathcal{D}|\mathrm{X},\mathrm{X}_{\mathcal{D}})}{p(\mathcal{D}|\mathrm{X}_{\mathcal{D}})} = \frac{\mathcal{N}\left(\begin{bmatrix}\boldsymbol{\phi}\\ \boldsymbol{y}_{\mathcal{D}}\end{bmatrix};\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}\mathbf{K} & \mathbf{K}_{\mathcal{D}}^{\top}\\ \mathbf{K}_{\mathcal{D}} & \boldsymbol{\Sigma}_{\mathcal{D}}\end{bmatrix}\right)}{\mathcal{N}(\boldsymbol{y}_{\mathcal{D}};0,\boldsymbol{\Sigma}_{\mathcal{D}})}$$

$$= \frac{\exp\left[-\frac{1}{2}\left(\boldsymbol{\phi}^{\top}-\boldsymbol{y}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\mathbf{K}_{\mathcal{D}}\right)\left(\mathbf{K}-\mathbf{K}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\mathbf{K}_{\mathcal{D}}\right)^{-1}\left(\boldsymbol{\phi}-\mathbf{K}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\boldsymbol{y}_{\mathcal{D}}\right)\right]}{\sqrt{(2\pi)^{N}\left|\mathbf{K}-\mathbf{K}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\mathbf{K}_{\mathcal{D}}\right|}}$$




- この期待値と分散共分散行列は次のとおり。

$$\boldsymbol{\mu}_f(\mathrm{X}) = \mathbf{K}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\boldsymbol{y}_{\mathcal{D}}, \qquad \boldsymbol{V}_f(\mathrm{X}) = \mathbf{K}-\mathbf{K}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\mathbf{K}_{\mathcal{D}}$$

# ２つの多変量ガウス分布の同時分布

- 次の２つの多変量ガウス分布 $\boldsymbol{x}_1, \boldsymbol{x}_2$ の同時分布について考える。

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right), \qquad \boldsymbol{\Sigma} \equiv \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}$$

- この分散共分散行列の逆行列は、次のようになる。

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{bmatrix}\boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12}\\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22}\end{bmatrix} = \begin{bmatrix}(\boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21})^{-1} & -(\boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21})^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\\ -(\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12})^{-1}\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1} & (\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12})^{-1}\end{bmatrix}$$

- したがって、同時分布は以下のとおりとなる。

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2)$$

$$= \frac{1}{\sqrt{(2\pi)^{k_1+k_2}|\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}([\boldsymbol{x}_1 - \boldsymbol{\mu}_1]^{\mathsf{T}} \quad [\boldsymbol{x}_2 - \boldsymbol{\mu}_2]^{\mathsf{T}}) \begin{pmatrix}\boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12}\\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22}\end{pmatrix} \begin{pmatrix}\boldsymbol{x}_1 - \boldsymbol{\mu}_1\\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2\end{pmatrix}\right]$$

$$= \frac{1}{\sqrt{(2\pi)^{k_1+k_2}|\boldsymbol{\Sigma}|}} \exp\left[-\frac{(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^{\mathsf{T}}\boldsymbol{\Lambda}_{11}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1) + (\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^{\mathsf{T}}\boldsymbol{\Lambda}_{12}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) + (\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^{\mathsf{T}}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1) + (\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^{\mathsf{T}}\boldsymbol{\Lambda}_{22}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2)}{2}\right]$$

$$= \frac{1}{\sqrt{(2\pi)^{k_1+k_2}|\boldsymbol{\Sigma}|}} \exp\left[-\frac{(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^{\mathsf{T}}\boldsymbol{\Lambda}_{11}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1) + 2(\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^{\mathsf{T}}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1) + (\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^{\mathsf{T}}\boldsymbol{\Lambda}_{22}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2)}{2}\right]$$

$$= \frac{1}{\sqrt{(2\pi)^{k_1+k_2}|\boldsymbol{\Sigma}|}} \exp\left[-\frac{[\boldsymbol{x}_2 - \boldsymbol{\mu}_2 + \boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)]^{\mathsf{T}}\boldsymbol{\Lambda}_{22}[\boldsymbol{x}_2 - \boldsymbol{\mu}_2 + \boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)] + (\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^{\mathsf{T}}(\boldsymbol{\Lambda}_{11} - \boldsymbol{\Lambda}_{12}\boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21})(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)}{2}\right]$$

  - ここで、$\boldsymbol{\Lambda}$ は対称なので、次の関係を使った。

$$(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^{\mathsf{T}}\boldsymbol{\Lambda}_{12}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) = (\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^{\mathsf{T}}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)$$

# 多変量ガウス分布の周辺化と条件付き確率

- 2つの多変量ガウス分布の同時確率分布から、周辺分布 (Marginal distribution) と条件付き確率 (Conditional probability) を導くことができる。
- 周辺分布 (Marginal distribution)

$$p(\boldsymbol{x}_1) = \int p(\boldsymbol{x}_1, \boldsymbol{x}_2) d\boldsymbol{x}_2$$

$$= \frac{\exp\left[-\frac{(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^\top (\boldsymbol{\Lambda}_{11} - \boldsymbol{\Lambda}_{21}^\top \boldsymbol{\Lambda}_{22}^{-1} \boldsymbol{\Lambda}_{21})(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)}{2}\right]}{\sqrt{(2\pi)^{k_1+k_2} |\boldsymbol{\Sigma}_{11}| |\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}|}} \int \exp\left[-\frac{[\boldsymbol{x}_2 - \boldsymbol{\mu}_2 + \boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)]^\top \boldsymbol{\Lambda}_{22} [\boldsymbol{x}_2 - \boldsymbol{\mu}_2 + \boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)]}{2}\right] d\boldsymbol{x}_2$$

$$= \frac{\exp\left[-\frac{(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_{11}^{-1} (\boldsymbol{x}_1 - \boldsymbol{\mu}_1)}{2}\right]}{\sqrt{(2\pi)^{k_1+k_2} |\boldsymbol{\Sigma}_{11}| |\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}|}} \sqrt{(2\pi)^{k_2} |\boldsymbol{\Lambda}_{22}^{-1}|} = \frac{\exp\left[-\frac{(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_{11}^{-1} (\boldsymbol{x}_1 - \boldsymbol{\mu}_1)}{2}\right]}{\sqrt{(2\pi)^{k_1} |\boldsymbol{\Sigma}_{11}|}} = \mathcal{N}(\boldsymbol{x}_1; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$$

- 条件付き確率 (Conditional probability)

$$p(\boldsymbol{x}_2 | \boldsymbol{x}_1) = \frac{p(\boldsymbol{x}_1, \boldsymbol{x}_2)}{p(\boldsymbol{x}_1)} = \frac{1}{\sqrt{(2\pi)^{k_2} |\boldsymbol{\Lambda}_{22}^{-1}|}} \exp\left[-\frac{[\boldsymbol{x}_2 - \boldsymbol{\mu}_2 + \boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)]^\top \boldsymbol{\Lambda}_{22} [\boldsymbol{x}_2 - \boldsymbol{\mu}_2 + \boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)]}{2}\right]$$

$$= \mathcal{N}(\boldsymbol{x}_2; \boldsymbol{\mu}_2 - \boldsymbol{\Lambda}_{22}^{-1}\boldsymbol{\Lambda}_{21}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Lambda}_{22}^{-1}) = \mathcal{N}(\boldsymbol{x}_2; \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12})$$

# ブロック行列の行列式と逆行列

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

- ここに、$A$ と $D$ は逆行列をもつ正方行列とする。

- 行列式:

$$|M| = |A||S| = |D||T|$$
$$S = D - CA^{-1}B$$
$$T = A - BD^{-1}C$$

- 逆行列:

$$M^{-1} = \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}$$
$$= \begin{pmatrix} T^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}$$
$$= \begin{pmatrix} T^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & -BD^{-1} \\ -CA^{-1} & I \end{pmatrix}$$
$$= \begin{pmatrix} T^{-1} & -T^{-1}BD^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}$$

- $S$ と $T$ の逆行列

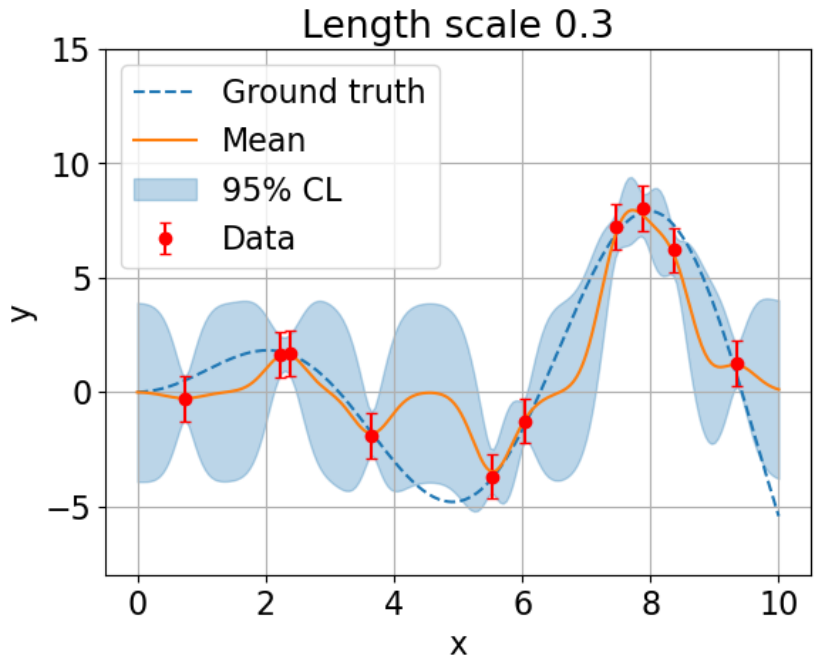
$$S^{-1} = (D - CA^{-1}B)^{-1} = D^{-1} + D^{-1}CT^{-1}BD^{-1} = D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1}$$
$$T^{-1} = (A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}BS^{-1}CA^{-1} = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}$$

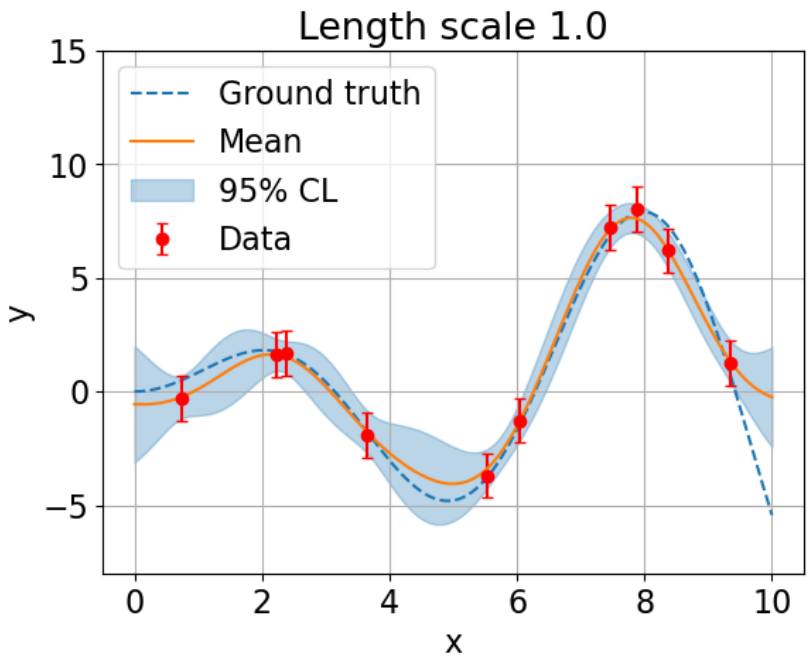
# ガウス過程回帰のハイパーパラメータ

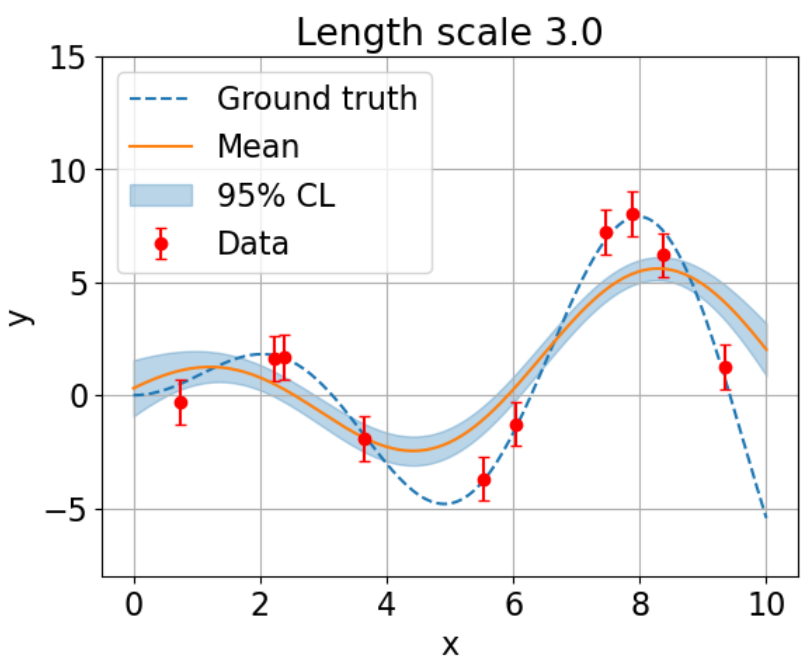
- ここまではカーネル関数として動径基底関数 $K(\boldsymbol{x}_a, \boldsymbol{x}_b) \propto \exp\left(-\frac{1}{2}|\boldsymbol{x}_a - \boldsymbol{x}_b|^2\right)$ を使用してきた。
- 動径基底関数のパラメータとして距離尺度 (length scale) なども導入できる。
- このようなパラメータをハイパーパラメータ (Hyper parameter) と呼ぶ。
- 距離尺度を入れた動径基底関数は以下のように定義できる。

$$K(\boldsymbol{x}_a, \boldsymbol{x}_b) \propto \exp\left(-\frac{1}{2}(\boldsymbol{x}_a - \boldsymbol{x}_b)^{\mathrm{T}}\Lambda^{-1}(\boldsymbol{x}_a - \boldsymbol{x}_b)\right), \qquad \Lambda^{-1} = [\mathrm{diag}\,(\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\lambda})]^{-1} = \begin{pmatrix} \lambda_1^{-2} & & O \\ & \ddots & \\ O & & \lambda_n^{-2} \end{pmatrix}, \qquad \boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_n)$$

- ベクトル $\boldsymbol{\lambda}$ が距離尺度にあたる。
- 回帰結果はハイパーパラメータに大きく依存する。



# ハイパーパラメータの最適化

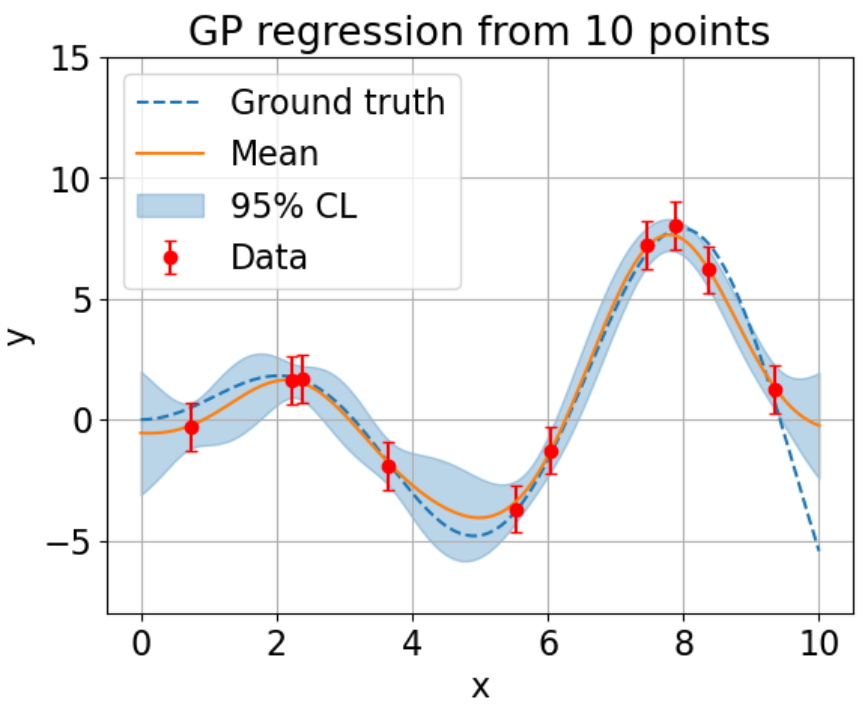
- ベイズ学習においては、ハイパーパラメータは周辺尤度で評価することが一般的である。

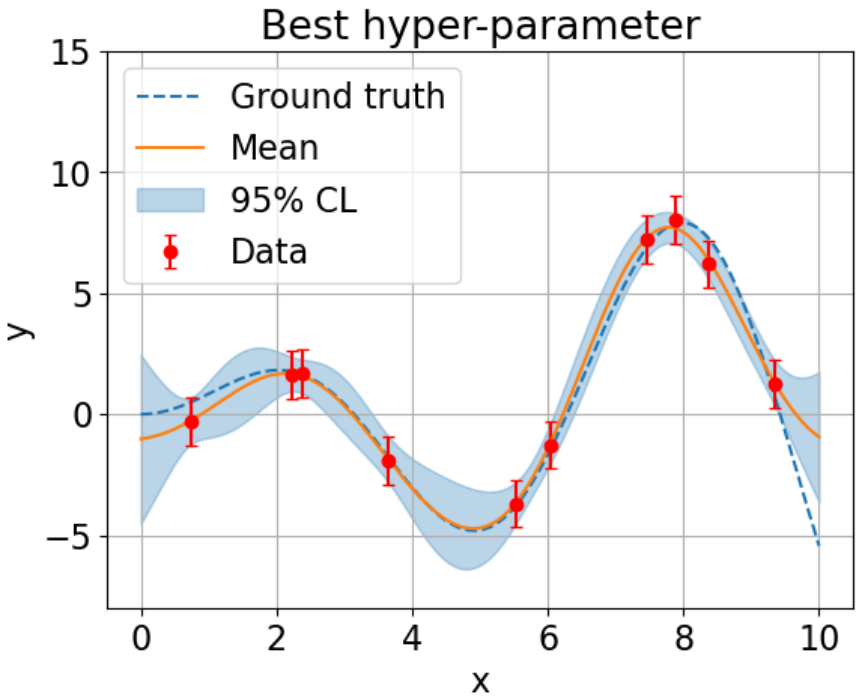
$$p(\mathcal{D}|\mathrm{X}_{\mathcal{D}}) = \mathcal{N}(\boldsymbol{y}_{\mathcal{D}}; 0, \boldsymbol{\Sigma}_{\mathcal{D}}) = \frac{\exp\left(-\frac{1}{2}\boldsymbol{y}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\boldsymbol{y}_{\mathcal{D}}\right)}{\sqrt{(2\pi)^{N_{\mathcal{D}}}|\boldsymbol{\Sigma}_{\mathcal{D}}|}}$$

- これは対数を取った方が計算しやすい。

$$\ln p(\mathcal{D}|\mathrm{X}_{\mathcal{D}}) = -\frac{1}{2}\boldsymbol{y}_{\mathcal{D}}^{\top}\boldsymbol{\Sigma}_{\mathcal{D}}^{-1}\boldsymbol{y}_{\mathcal{D}} - \frac{1}{2}\ln|\boldsymbol{\Sigma}_{\mathcal{D}}| - \frac{N_{\mathcal{D}}}{2}\ln 2\pi$$

- もし、モデルとそのハイパーパラメータがデータをよく再現している場合、周辺尤度が高くなる。
- 通常、ハイパーパラメータに対する周辺尤度の最大値は解析的には計算しづらいため、数値的に求めることが多い。



# ベイズ最適化

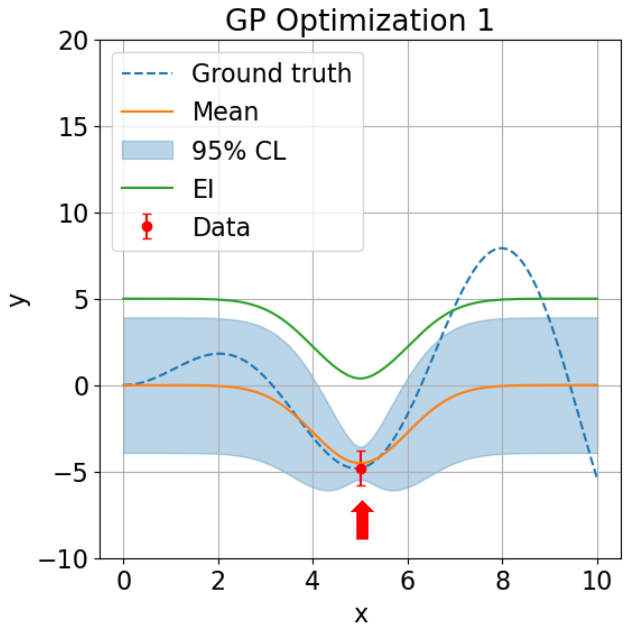
- ガウス過程回帰等のベイズ学習により、目的関数の形状を学習・推論するところまでできた。
- 最適化をするためには、入力パラメータに対する評価値を探索し、評価値の最適値を求めなければならない。
- 次の入力パラメータをどう決める？
- 目的関数の確率分布から次の入力パラメータを決めるための関数を獲得関数 (Acquisition function) と呼ぶ。
- ここでは、ガウス過程回帰を行い、獲得関数として期待改善度 (Expected improvement, EI) と信頼上限 (Upper confidence bound, UCB) を例として最適化を行ってみる。
- ガウス過程最適化の特徴
  - 試行回数が比較的少なく、高速である。
  - 局所最適値があっても大域最適値に到達しやすい。
  - 初期値やハイパーパラメータが比較的少ない。
  - モデルに誤差を入れられるので、測定誤差に基づいて最適化できる。

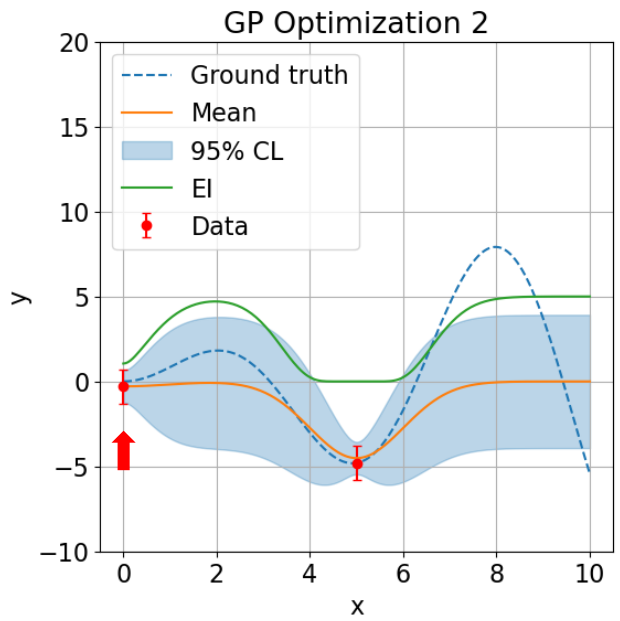
# 期待改善度 (Expected improvement, EI)

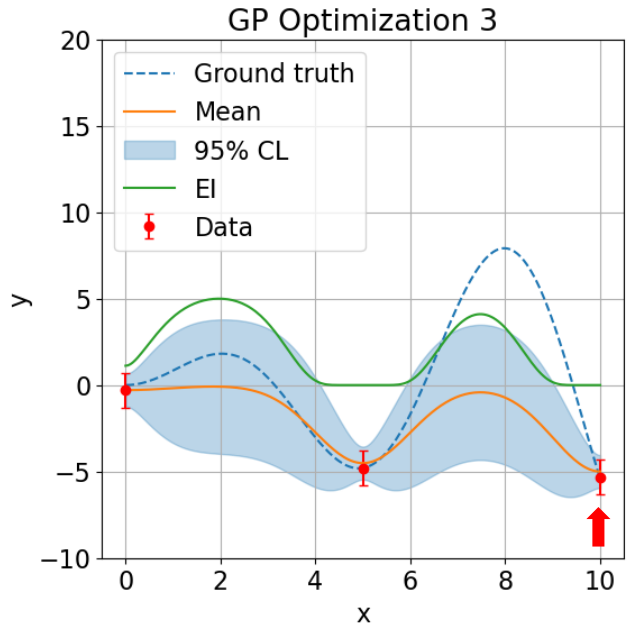
- **期待改善度** (EI) とは、これまでに得られた最適値 $(y_{\max})$ よりよい値が得られる確率のことで、次のように定義される。

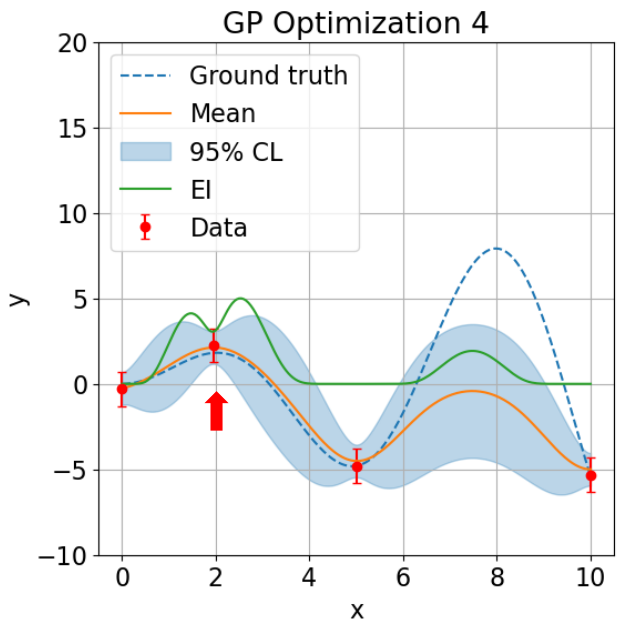
$$\alpha_{\mathrm{EI}}(\boldsymbol{x}) = \int_{y_{\max}}^{\infty} (y - y_{\max})\, p(y|\boldsymbol{y}_{\mathcal{D}}, \mathrm{X}_{\mathcal{D}}, \boldsymbol{x})\, dy$$

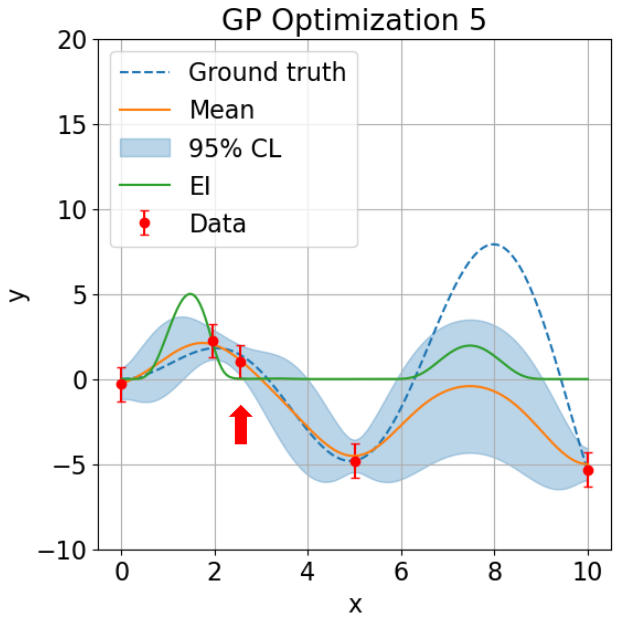
- 次の試行点は EI が最大値をとる $\boldsymbol{x}$ を選ぶ。

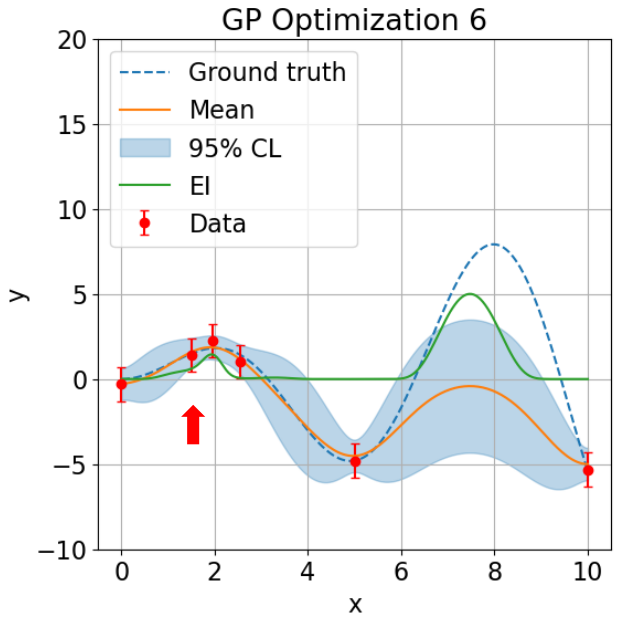


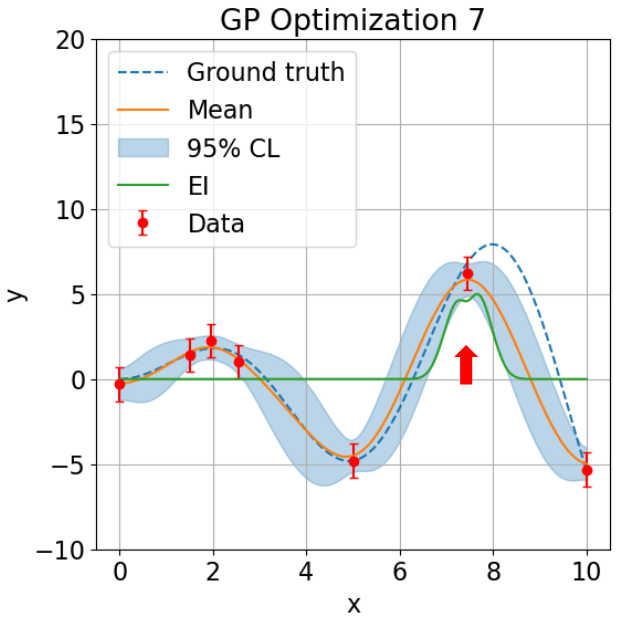



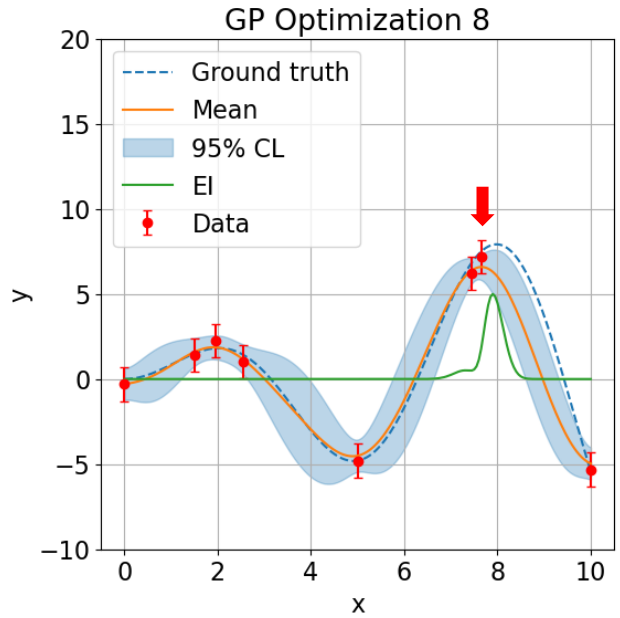



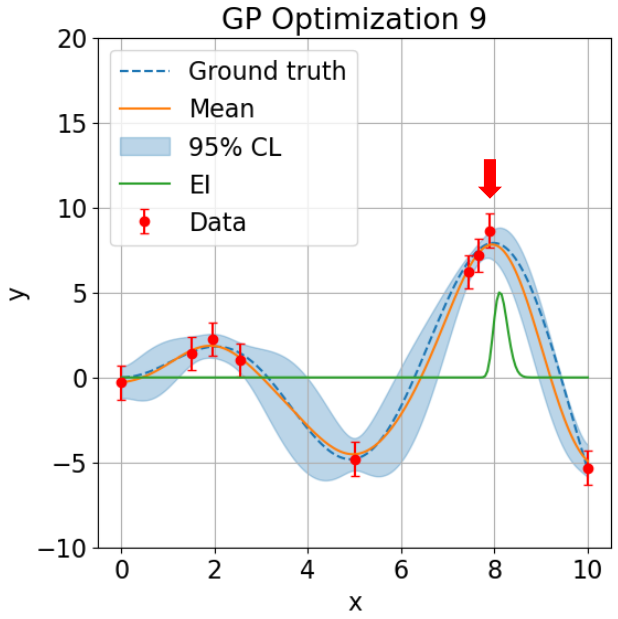



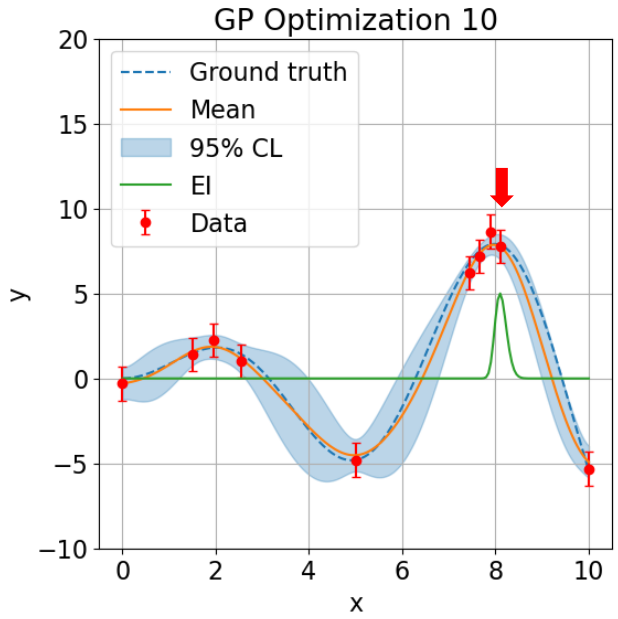


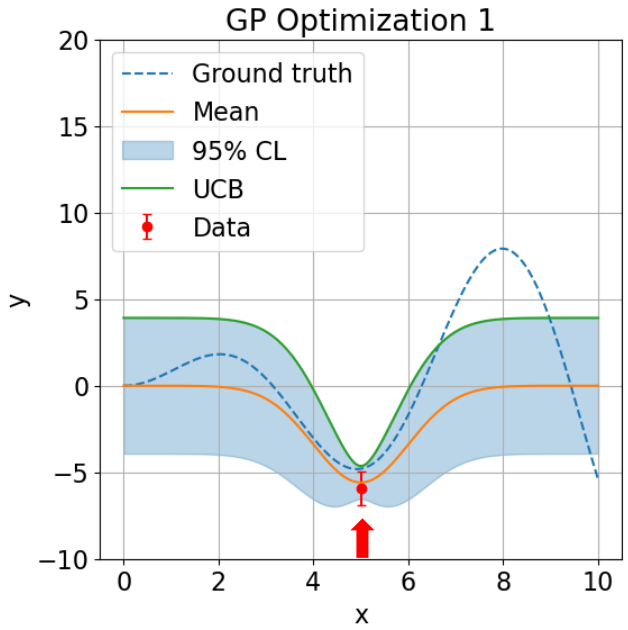

大域最適値がわずか 10 回程度の試行で得られた。

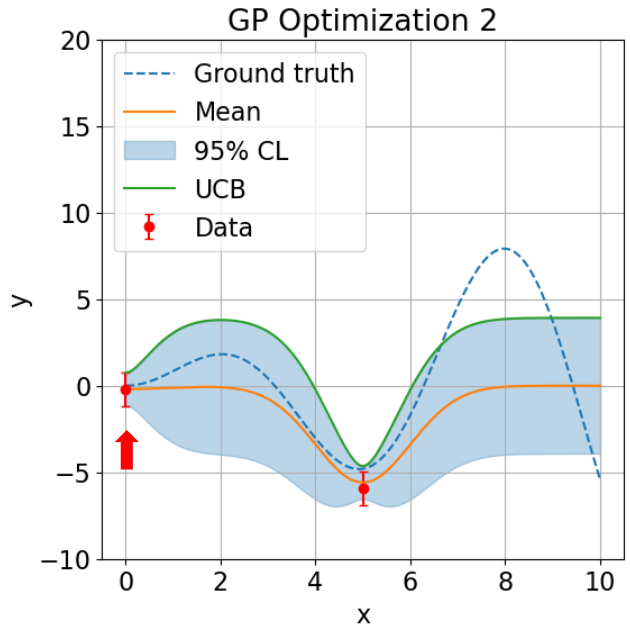
# 信頼上限 (Upper confidence bound, UCB)

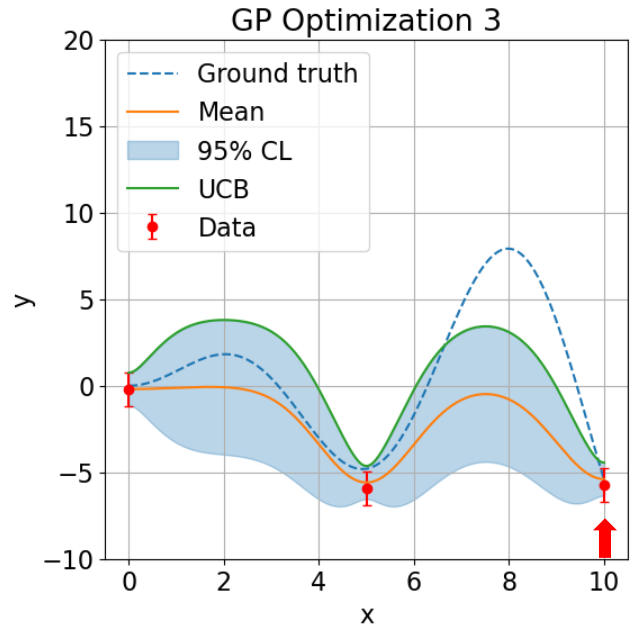
- 信頼上限 (UCB) とは、信頼区間の上限の関数のことで、次のように適宜される。

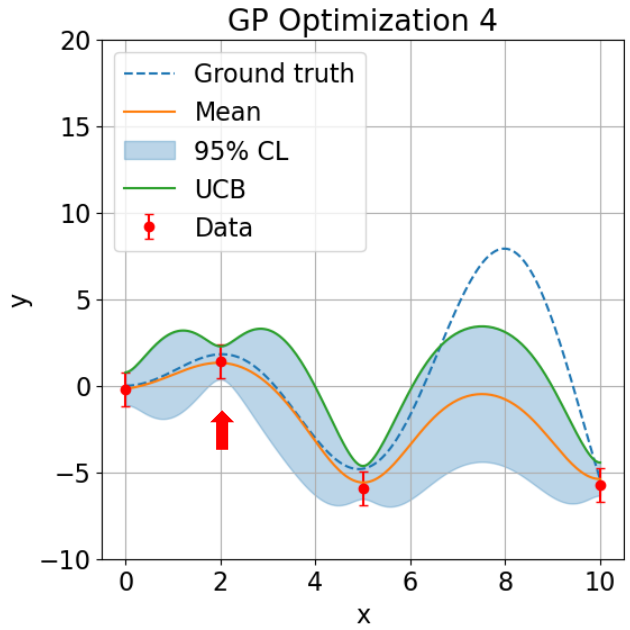
$$\alpha_{\mathrm{UCB}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \lambda\sigma(\boldsymbol{x})$$

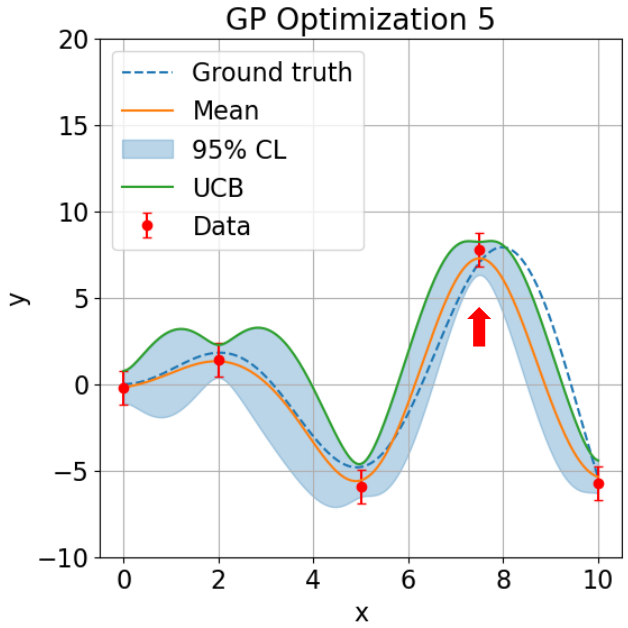
  - ここで $\mu$ は期待値、$\sigma$ は標準偏差である。
  - 係数 $\lambda$ は適当に決める。
- 次の試行点は UCB が最大値をとる $\boldsymbol{x}$ を選ぶ。

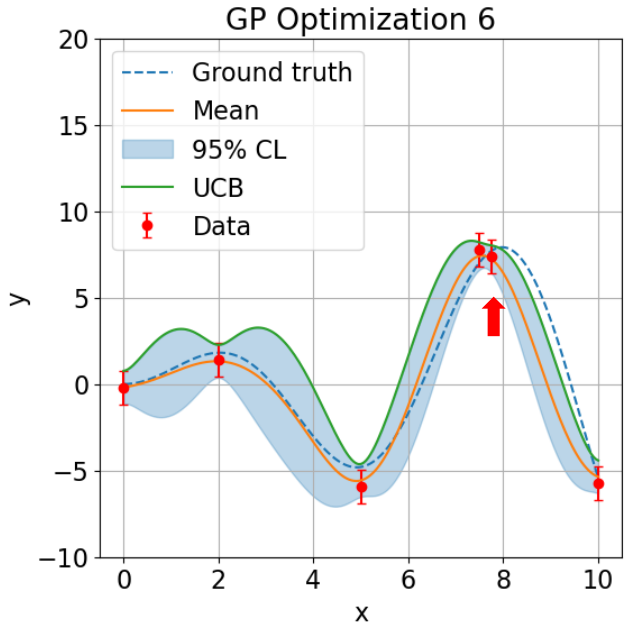


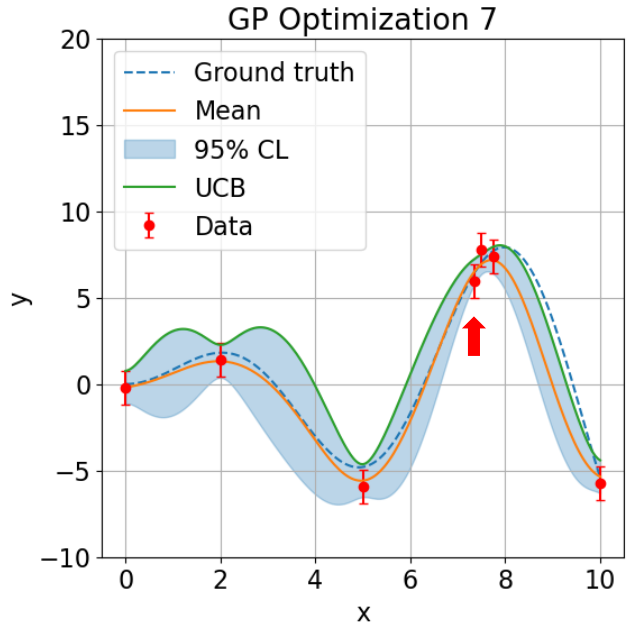



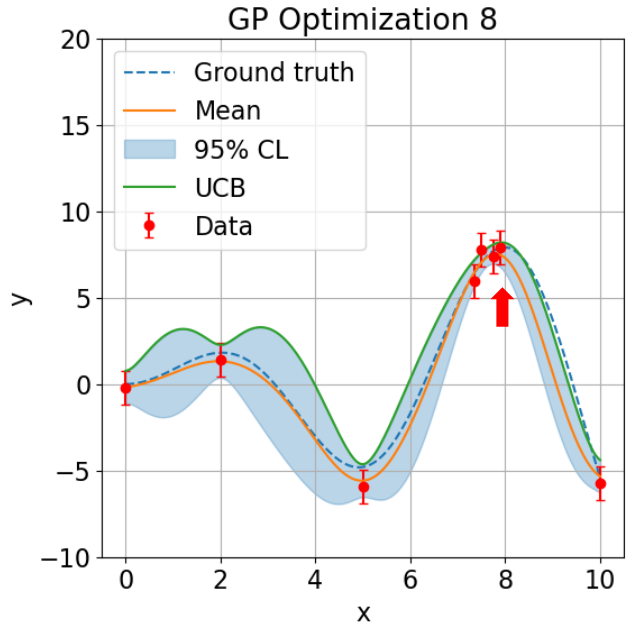



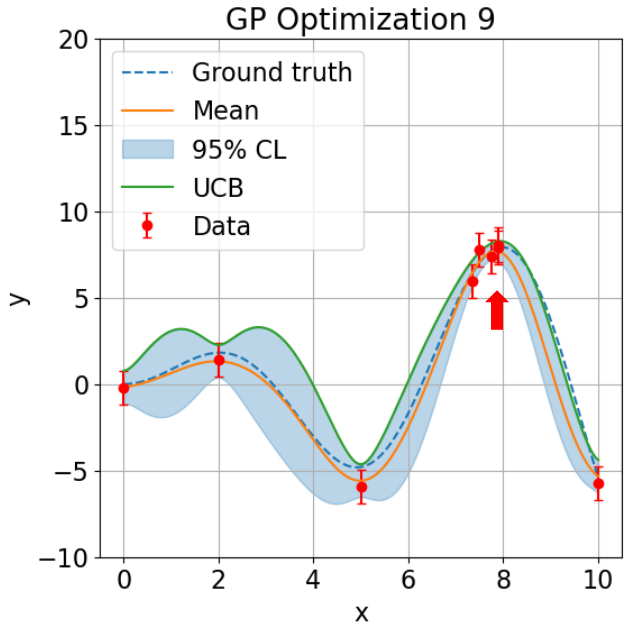



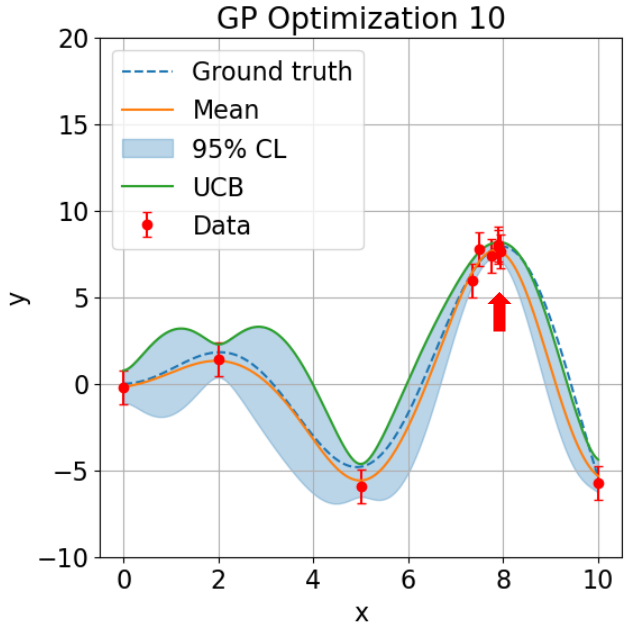


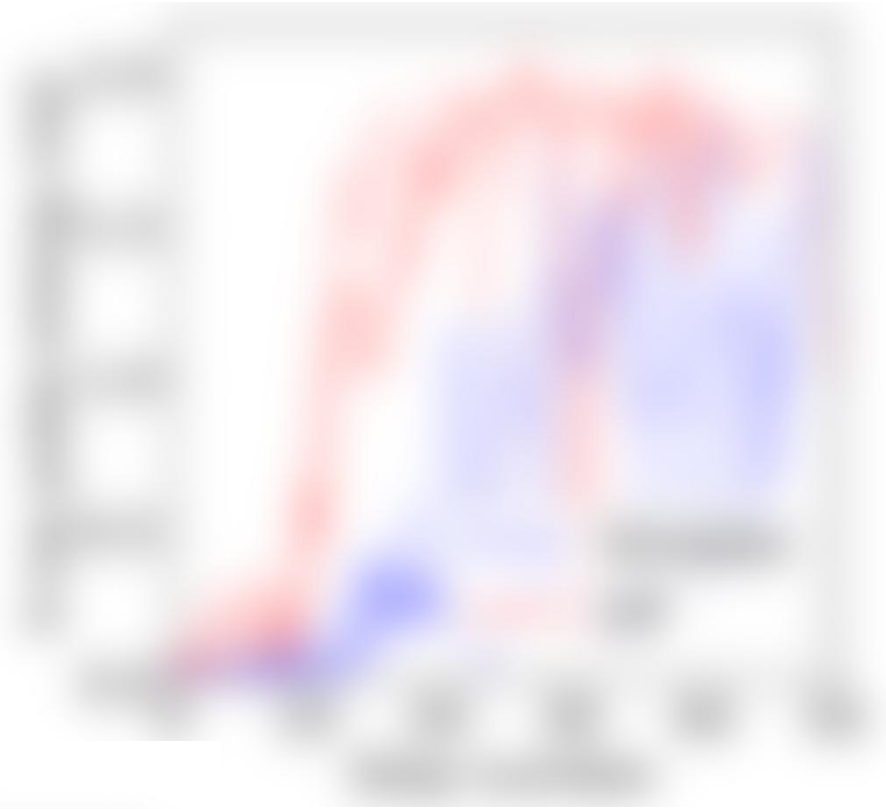




- 大域最適値がわずか 10 回程度の試行で得られた。
- 局所最適値での試行は 2 回で済んだ。
  - EI は 3 回だった。
- EI と UCB のどちらを使うかは実際のシステムの応答に対する最適値への到達速度や未探索点への行きやすさなどを見ながら選ぶことになるだろう。

# LCLS (SLAC) における XFEL のパルス強度最適化

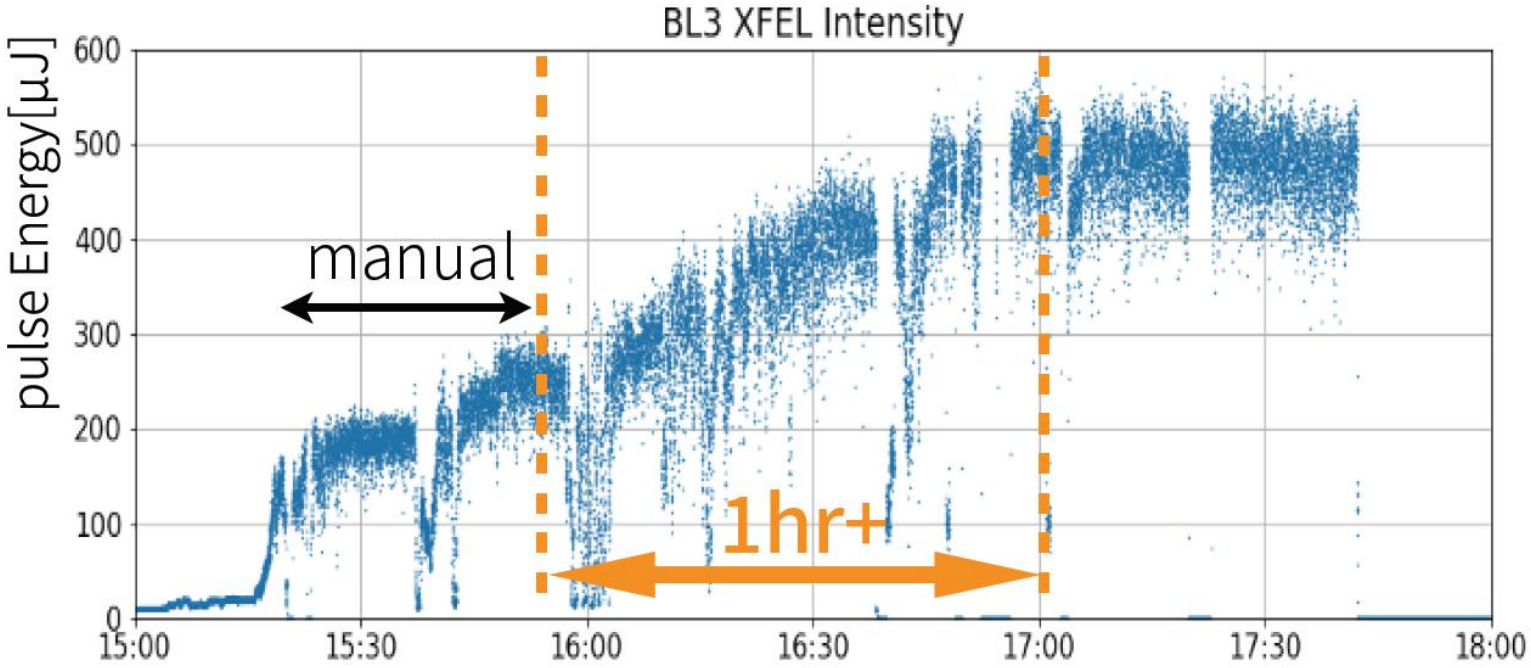
J. Duris *et al*.,. Phys. Rev. Lett. **124**, 124801 (2020)

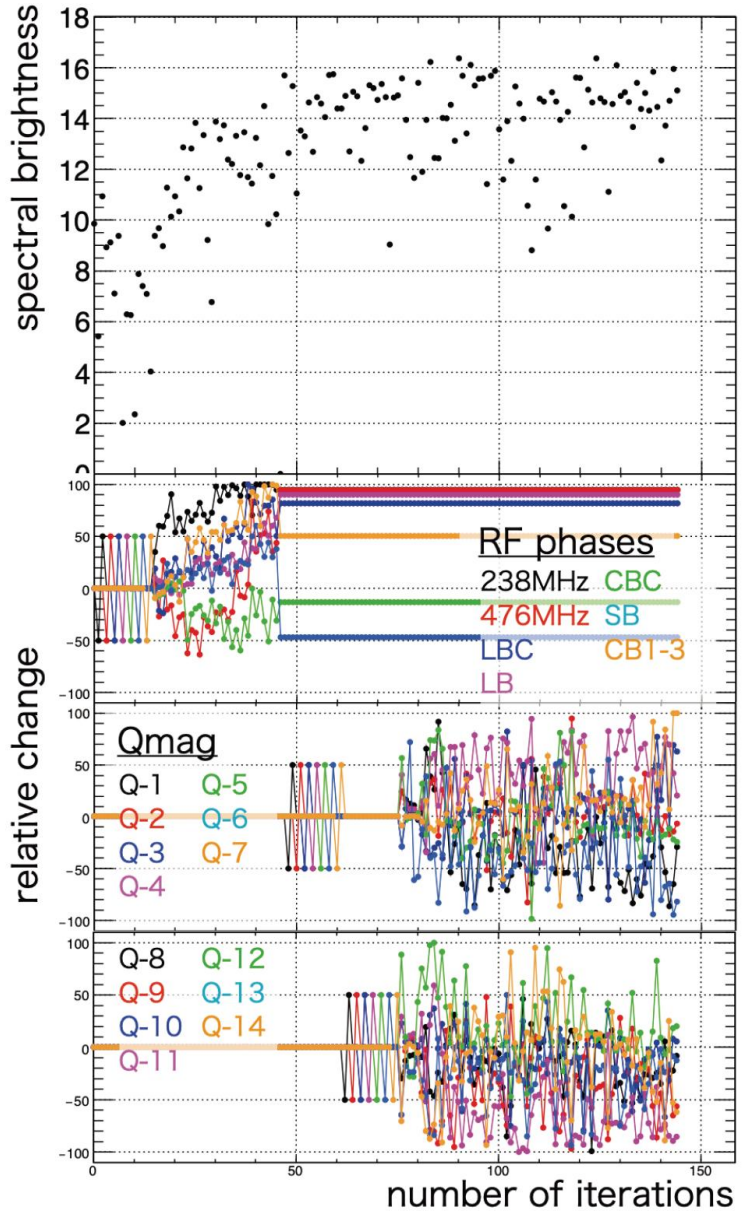
- ガウス過程最適化が XFEL に応用された最初の例。
- ガウス過程最適化が Nelder-Mead 法のような古典的な手法より優れていることが示された。
- この例では 12 台の四極磁石を同時に変えて探索した例である。



# SACLA での XFEL のパルス強度最適化

- 上流部の加速高周波位相やマッチング部の四極磁石強度をガウス過程最適化で調整した。
- 10 以上のパラメータを同時に調整。
- XFEL 強度が 1 時間ほどで最適化され、手動調整より高速であった。

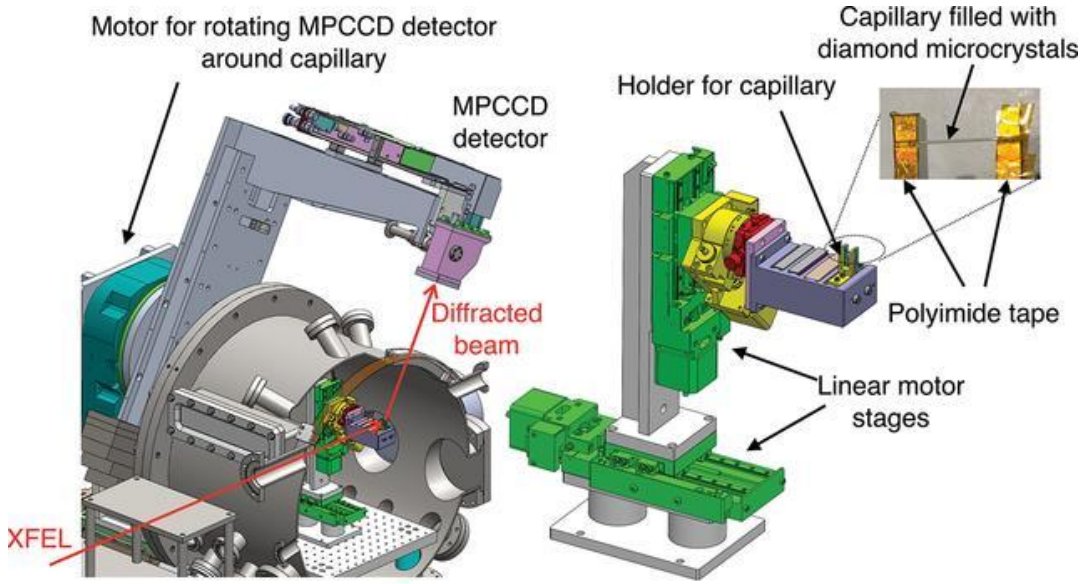


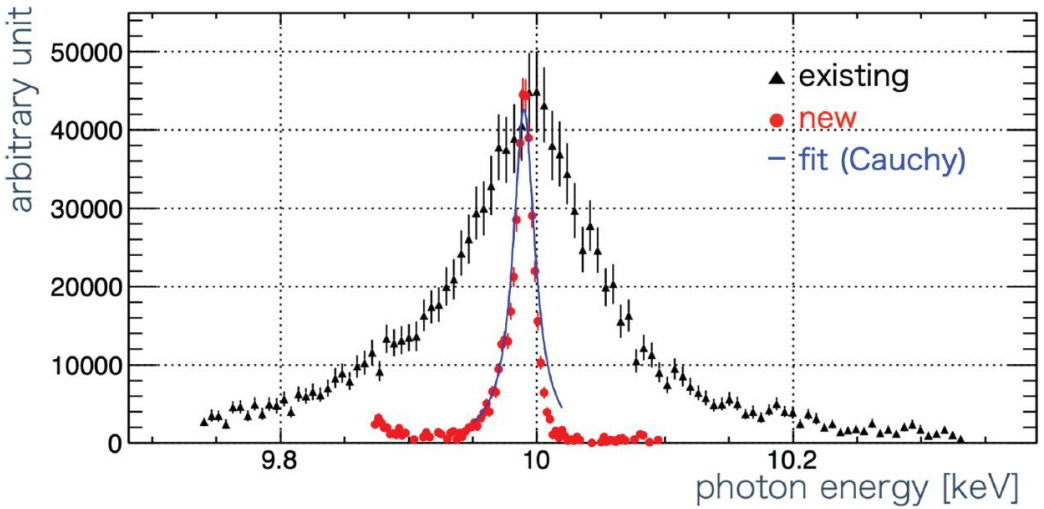



# SACLA における XFEL のスペクトル輝度最適化

E. Iwai *et al*., J. Synchrotron Rad. **30**, 1048–1053 (2023).
I. Inoue *et al*., J. Synchrotron Rad. **29**, 862–865 (2022).

- XFEL ユーザーにとっては、強度とともにスペクトル輝度も重要である。
- 常時測定可能なシングルショット分光器は存在したが分解能が足りなかった。
- そこで、常時測定可能な高分解能シングルショット分光器を新たに開発。
- この新しい分光器のデータを使ってスペクトル輝度を最適化した。



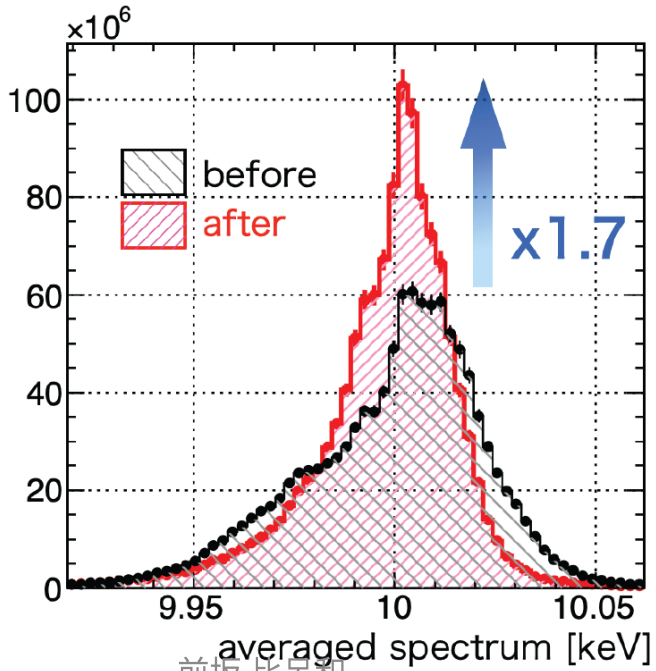



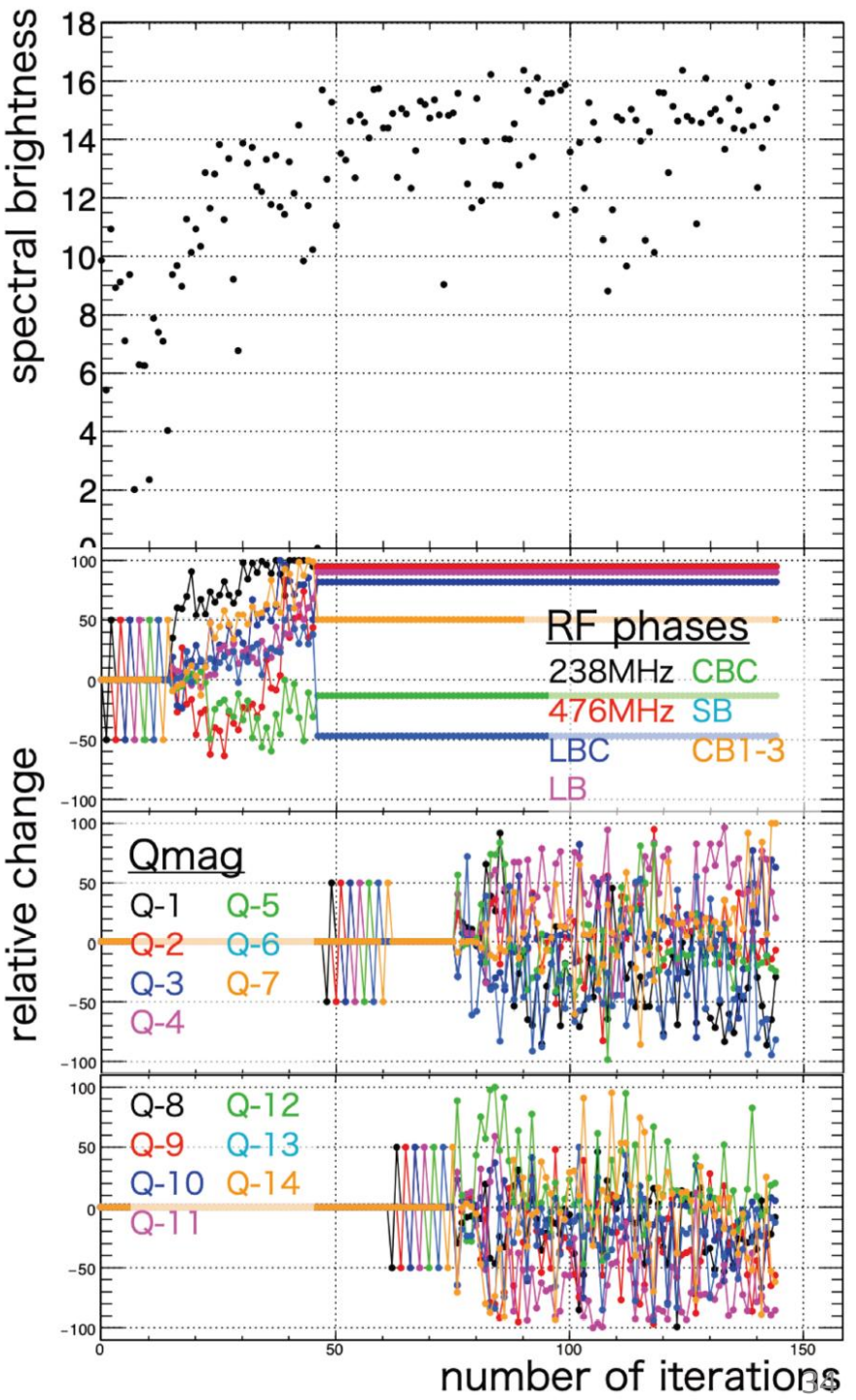
# スペクトル輝度の最適化

- スペクトル輝度の定義

$$B_{\mathrm{avg}} = \frac{P}{\sigma_{\mathrm{avg}}}, \qquad \sigma_{\mathrm{avg}} = \sqrt{\sigma_{\mathrm{width}}^2 + \sigma_{\mathrm{med}}^2}$$

  - $P$: XFEL パルスエネルギー
  - $\sigma_{\mathrm{width}}$: XFEL のスペクトル幅
  - $\sigma_{\mathrm{med}}$: スペクトルの中央値のふらつき
- 調整ノブ
  - 上流部の各加速ユニットの高周波位相 (7 ケ)
  - 低エネルギー部の磁気レンズの強さ (9 ケ)
  - 四極磁石の強さ (62 ケ)
  - ステアリング磁石の強さ (8 ケ)
  - アンジュレータ K 値のテーパ (7 ケ)
- 10 ケ 以上のパラメータを同時に最適化。
- スペクトル輝度を約 1.7 倍に増やすことに成功。
- 機械学習ベースの最適化には良い目的関数 (ビームモニタ) が重要。



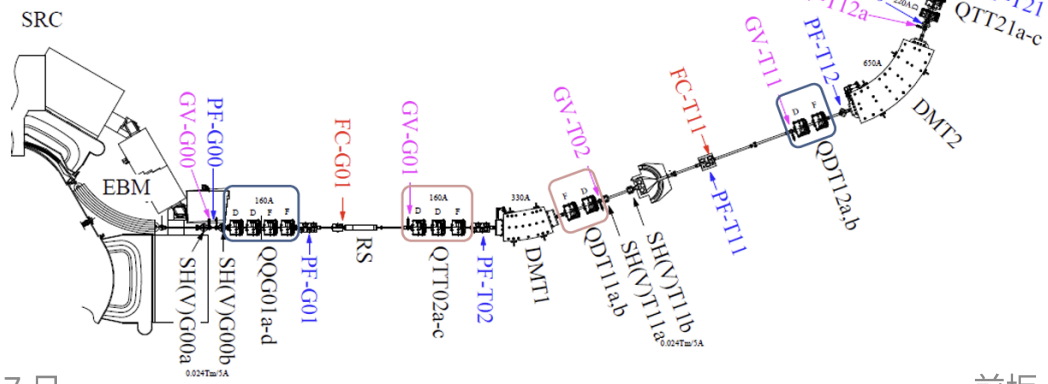



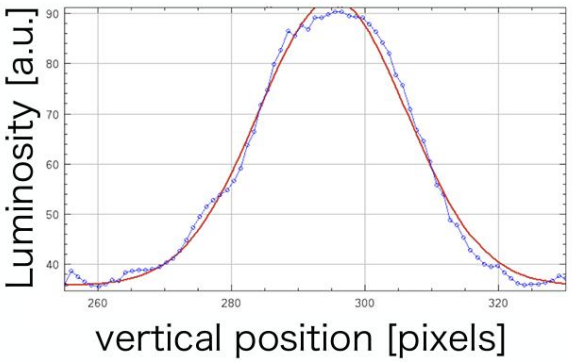
# RIBF での重イオンビームライン光学系の最適化

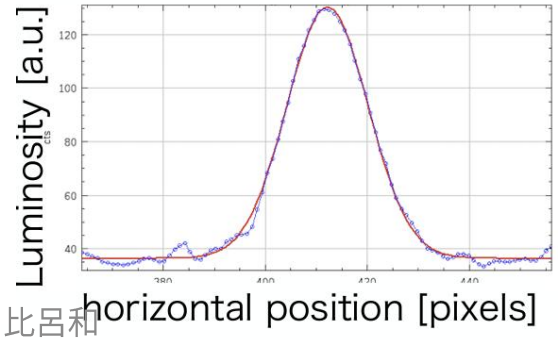
T. Nishi *et al.*, Proceedings of the 18th annual meeting of particle accelerator society of Japan (PASJ2021), TUOA03.
T. Nishi, Lecture on ISBA'22.

- 重イオンビームの強度を上げるには、ビーム損失を抑制することが極めて重要。
- ビームプロファイルと輸送効率をガウス過程で最適化。

超伝導リングサイクロトロン (SRC) から標的までのビームライン



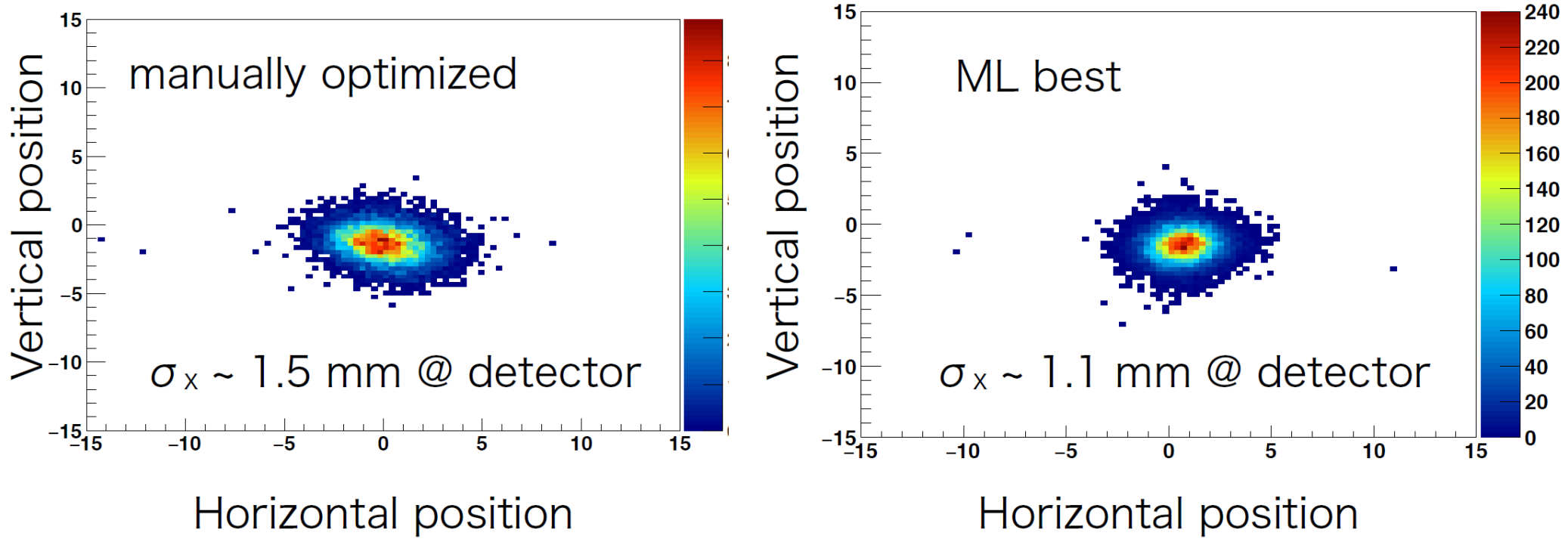




典型的なビームプロファイル

# 高強度ビームでの最適化例

- 高強度の一次ビームには蛍光スクリーンが使えないため、二次ビームの位置分布をガス飛跡検出器でとらえてビームプロファイルを測定した。
- ガウス過程最適化の結果、輸送効率を維持したままビームプロファイルを改善することができた。

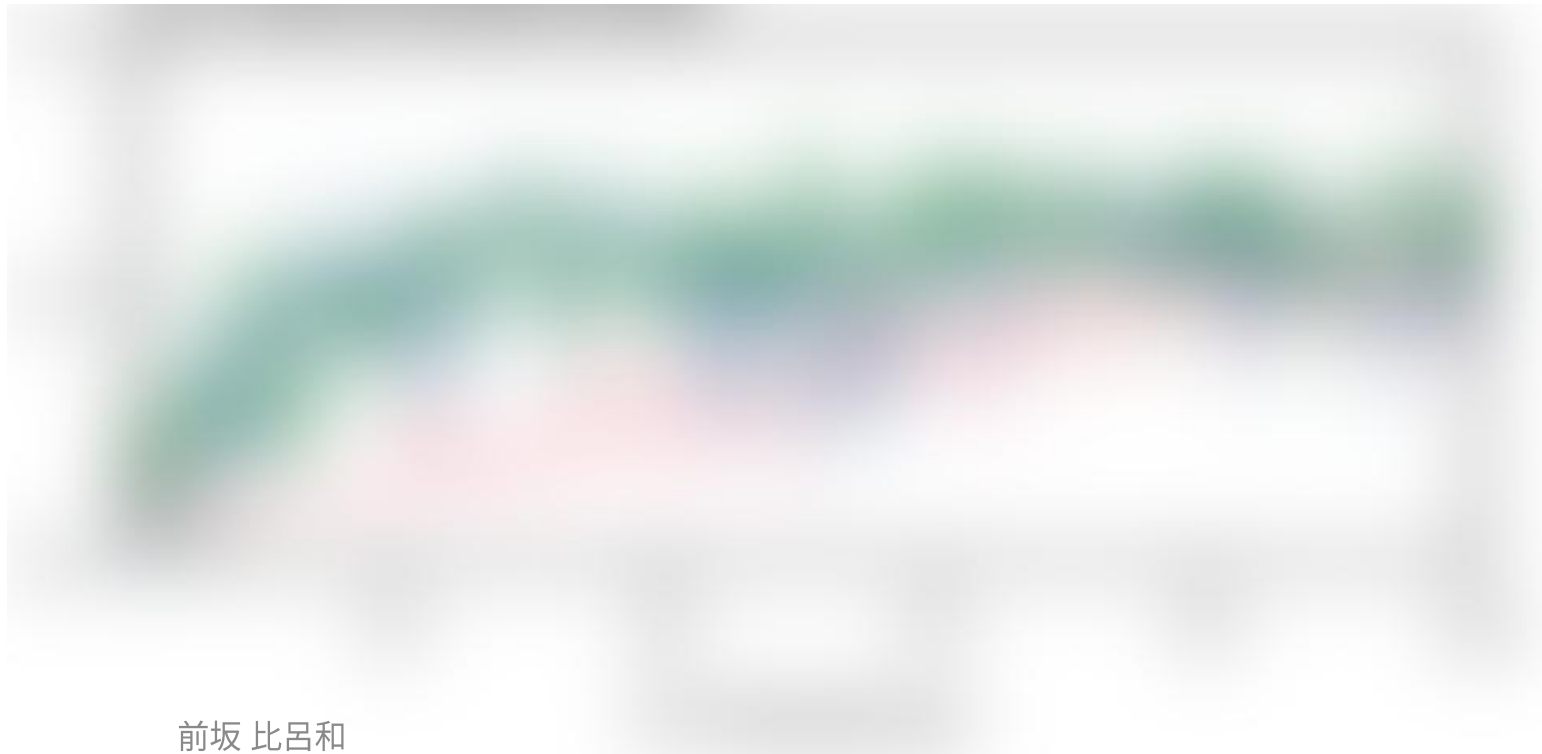
$Kr^{34+}$



# KIT-LS での入射効率最適化

Chenran Xu *et al.*, Phys. Rev. Accel. Beams **26**, 034601 (2023).

- KIT-LS: Karlsruhe Institute of Technology Light Source, Germany
- 入射バンプ軌道の途中に複数の六極磁石があり、その非線形キックで入射効率が低下する問題があった。
- ガウス過程最適化で入射効率を向上させることができた。



# ガウス過程最適化ツールの例

- General Gaussian process tools
  - Scikit Learn: https://scikit-learn.org/
    - Plenty of machine learning tools
  - BoTorch: https://botorch.org/
    - Bayesian optimization tool built on PyTorch
  - GPyTorch: https://gpytorch.ai/
    - Gaussian process inference tool built on PyTorch
- Gaussian process optimizer for accelerators
  - SACLA original: E. Iwai *et al*., J. Synchrotron Rad. 30, 1048–1053 (2023).
  - Xopt: https://github.com/xopt-org/Xopt
  - Badger: https://xopt-org.github.io/Badger/
  - APSopt: N. Kuklev *et al*., IPAC'24, TUPS50.
  - GeOFF: https://zenodo.org/records/8434513

# 複数の目的関数の同時最適化 (多目的最適化)

Multi-objective optimization

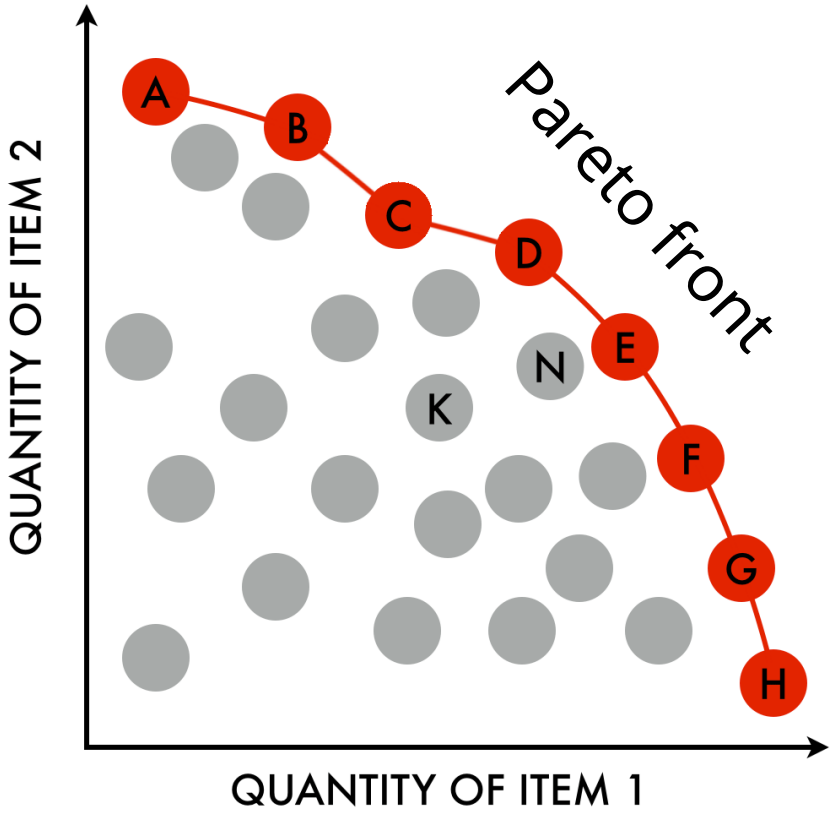
- **最適化の目的関数はひとつとは限らない。**
  - ビーム強度を上げたいが、ビームロスはしたくない。
  - XFEL のバンド幅を狭くしたいが、強度もほしい。
  - などなど。
- 複数の目的関数を組み合わせてひとつの目的関数にまとめることができれば問題ない。

$$y = a_1 y_1 + a_2 y_2$$

  などとできるなら $y$ を最適化すればよい。
- **ふつうのユーザーは複数の目的関数の重みを定量的に示せない。**
  - こっちは最低これくらいほしいし、あっちも最低これくらいはほしいのだけれど、そこから先はどっちもなるべく上げてほしい、といったあいまいな要求が来ることが世の常。
- このように複数の目的関数を同時最適化したいときはどのような考え方をすればよいのだろうか。

# パレートフロント (Pareto Front)

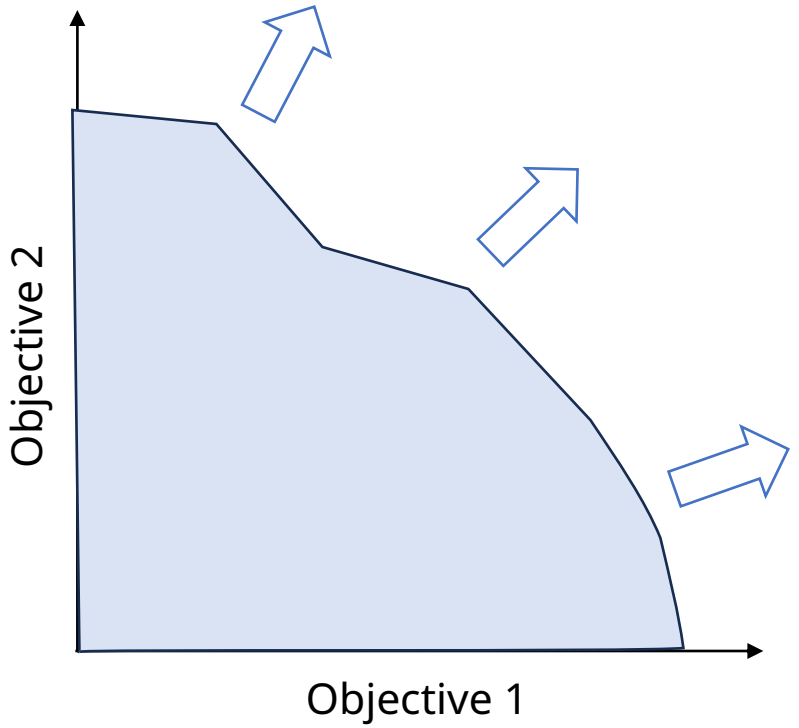
- トレードオフの関係にある複数の目的関数を最適化しようとするとそれ以上進めない壁にぶつかる。
- これをパレートフロント (Pareto front) と呼ぶ。
- 多目的最適化の最初の目標はこのパレートフロントの位置・形状を見出すことになる。
- これをユーザーに提示してどこがいいかを選んでもらう。
  - 何も情報がないよりはパレートフロントの情報を見てその中から選ぶ方が楽。




Author: Njr00 - Own work, CC BY-SA 3.0
https://commons.wikimedia.org/wiki/File:Pareto_Efficient_Frontier_1024x1024.png

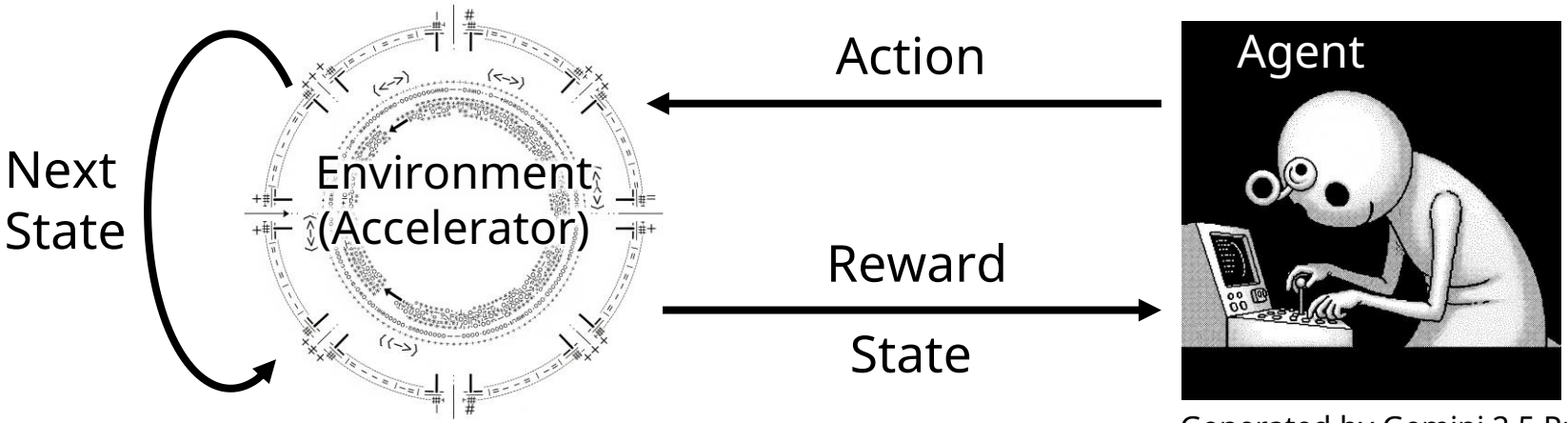
# ハイパーボリューム (Hyper Volume) (超体積)

- パレートフロントがある程度定まったとしても、それが限界なのか、まだ伸びしろがあるのかの指標がほしい。
- そういうときのためにハイパーボリューム (Hyper volume, 超体積) という考え方がある。
- ハイパーボリュームとはパレートフロントで囲まれた部分の体積のことで、パレートフロントという多次元空間からスカラーに落とし込むひとつの方法。
- パレートフロントをただ広げたい場合はハイパーボリュームを目的関数として最適化アルゴリズムを走らせるという選択肢がある。
- ハイパーボリュームの伸びしろがなくなったら最適化の限界と判断できる。
  - ユーザーにも努力の限界についての理解を得やすいだろう。



# 強化学習による最適化

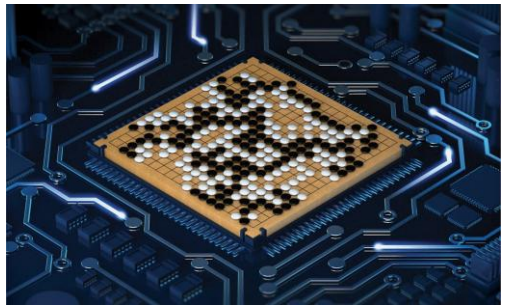
- ガウス過程最適化のようなベイズ最適化は十分な能力を持ってはいるものの、非効率な点がいくつかある。
  - 物理モデルを知らないため物理的にあり得ないところも探索しにいってしまう。
  - 毎回何も知らないところから始まるため効率が良くない。
    - 事前分布をあらかじめ与えておくことも可能だが、加速器の状態が変わっているとかえって効率が悪くなる恐れもある。
  - 全パラメータを平等に扱うため、次元やデータが増えると計算量が単純に増えてしまう。
- ベイズ最適化の次のステップとして強化学習が候補に挙げられる。
  - 強化学習には制御対象となる環境 (Environment) とそこで行動するエージェント (Agent) が存在する。
  - エージェントは、環境の状態 (State) を観察し、それをもとに行動・操作 (Action) を行い、状態の変化に応じて報酬 (Reward) を受け取る。
  - 収益 (累積報酬) が最大となるように方策 (Policy) を最適化 (学習) する。

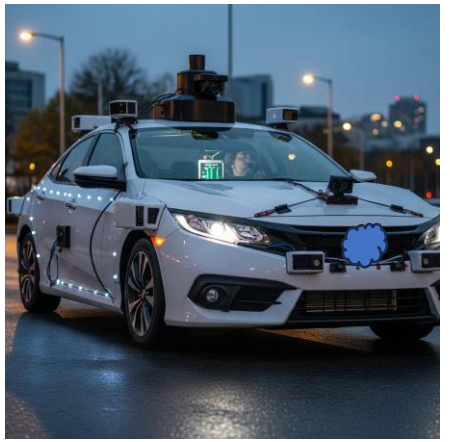



Generated by Gemini 2.5 Pro

# 強化学習の有名な応用例

- AlphaGo (DeepMind 社)
  - 2016 年に韓国の最強棋士 イ・セドル 氏を破ったことで有名。
- Agent57 (DeepMind 社)
  - Atari2600 の多くのレトロゲームでピクセル情報を入力として人間を上回るスコアをたたき出す。
- 二足歩行ロボット
  - 二足歩行だけでなく、宙返りができるようになった例も。
- 自動運転車
  - 実用レベルに近い自動運転車が強化学習で開発されている。

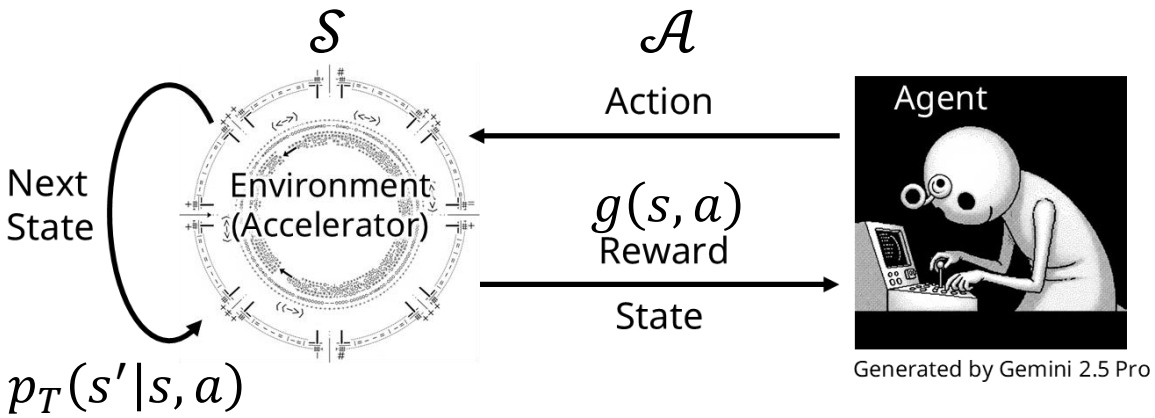




Generated by Gemini 2.5 Pro

# マルコフ決定過程

- マルコフ性: 事象の確率が１ステップ前の状態のみに依存し、
  それ以前の状態とは独立であること。
- 状態集合: $\mathcal{S}$
- 行動集合: $\mathcal{A}$
- 状態遷移確率: $p_T(s'|s,a)$
  - 状態 $s$ に行動 $a$ を起こした際に
    状態が $s'$ に遷移する確率。
- 報酬関数: $g(s,a)$
  - 状態 $s$ に行動 $a$ を起こした際に得られる報酬。
- 初期状態確率: $p_{s_0}(s)$




- この過程がうまく働くにはこれらがあらかじめ決まっていなければならない。
- 未知の場合はシステム同定やシミュレーションをするなどして決めてやらないといけない。
- これを学習しながら決めていくのが強化学習。
- なお、行動に対する状態変化が決まっているということはそこに物理法則が埋め込まれているということでもある。

# 方策 (policy) と収益 (累積報酬, return)

- 方策: $\pi(a|s) \in [0,1]$
  - 状態 $s$ のときに行動 $a$ を取る確率。
- 方策集合: $\Pi$
- 方策に関するある目的関数があれば、

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} f(\pi)$$

  となるような方策を選ぶことになる。
- 収益: $C_t$

$$C_t = \lim_{k \to \infty} \sum_{k=0}^{K} \gamma^k R_{t+k}$$

  - すなわち、ある時点から先の割引累積報酬のこと。
  - 割引率: $\gamma \in [0,1)$, 時点 $t$ での報酬: $R_t$

収益を計算するにしても結局未来がどうなるかが見通せてないといけないので、システム同定がちゃんとできていることがキモとなる。
そこが強化学習の難しいところ。

- 価値関数 (value function): $V^\pi(s) = \mathbb{E}^\pi(C_0|S_0 = s)$
  - 初期状態が $s$ のときに方策 $\pi$ に従って行動した場合に得られる収益の期待値。
- 方策の目的関数としてこの価値関数 $V^\pi(s)$ がよく用いられる。

# ベルマン方程式 (Bellman equation)

- ある方策をとったときの状態変化の確率 $p_{MC}^{\pi}(s'|s)$ は取りうる行動に対する和から以下のように書ける。

$$p_{MC}^{\pi}(s'|s) = \sum_{a \in \mathcal{A}} \pi(a|s)\, p_T(s'|s,a)$$

- この過程にエルゴード性があればある定常確率分布 $p_{\infty}^{\pi}(s)$ に落ち着くことが知られている。
- このとき、方策の目的関数も定常的となる。

$$f_{\infty}^{\pi}(s) = \sum_{s \in \mathcal{S}} p_{\infty}^{\pi}(s)\, V^{\pi}(s)$$

エルゴード性とは時間平均とアンサンブル平均が等しいこと。

- 収益の再帰構造から価値関数は以下のようにあらわすことができる。

$$V^{\pi}(s) = \mathbb{E}^{\pi}(C_0|S_0 = s) = \mathbb{E}^{\pi}(R_0 + C_1|S_0 = s)$$
$$= \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ g(s,a) + \gamma \sum_{s \in \mathcal{S}} p_T(s'|s,a)\, \mathbb{E}^{\pi}(C_1|S_1 = s') \right\}$$

- よって、価値関数 $V^{\pi}$ に関する再帰式であるベルマン方程式が以下のように得られる。

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ g(s,a) + \gamma \sum_{s' \in \mathcal{S}} p_T(s'|s,a)\, V^{\pi}(s') \right\}$$

- **強化学習はこの方程式を解いて最適な方策を決定することといえる。**

# 強化学習の学習方法

- マルコフ決定過程の各種情報は最初はかなり乏しい状態からスタートすることが多いだろう。
- そこから状態遷移確率や価値関数などを推定してアップデートしながら学習を進めていく。
- 探索 (exploration) と活用 (exploitation) をバランスよく組み合わせて推定精度を高める学習アルゴリズムがいろいろと存在している。
  - 探索: 方策の精度を高めるためのデータ収集。(違うやり方を試す)
  - 活用: 収益の最大化。 (今の方策でなるべく収益を増やす)
- 学習にはかなりの試行回数と入力データが必要となるため、加速器の実機だけで行うのは非現実的で、シミュレーションやデジタルツインを活用することが不可欠と考えられる。

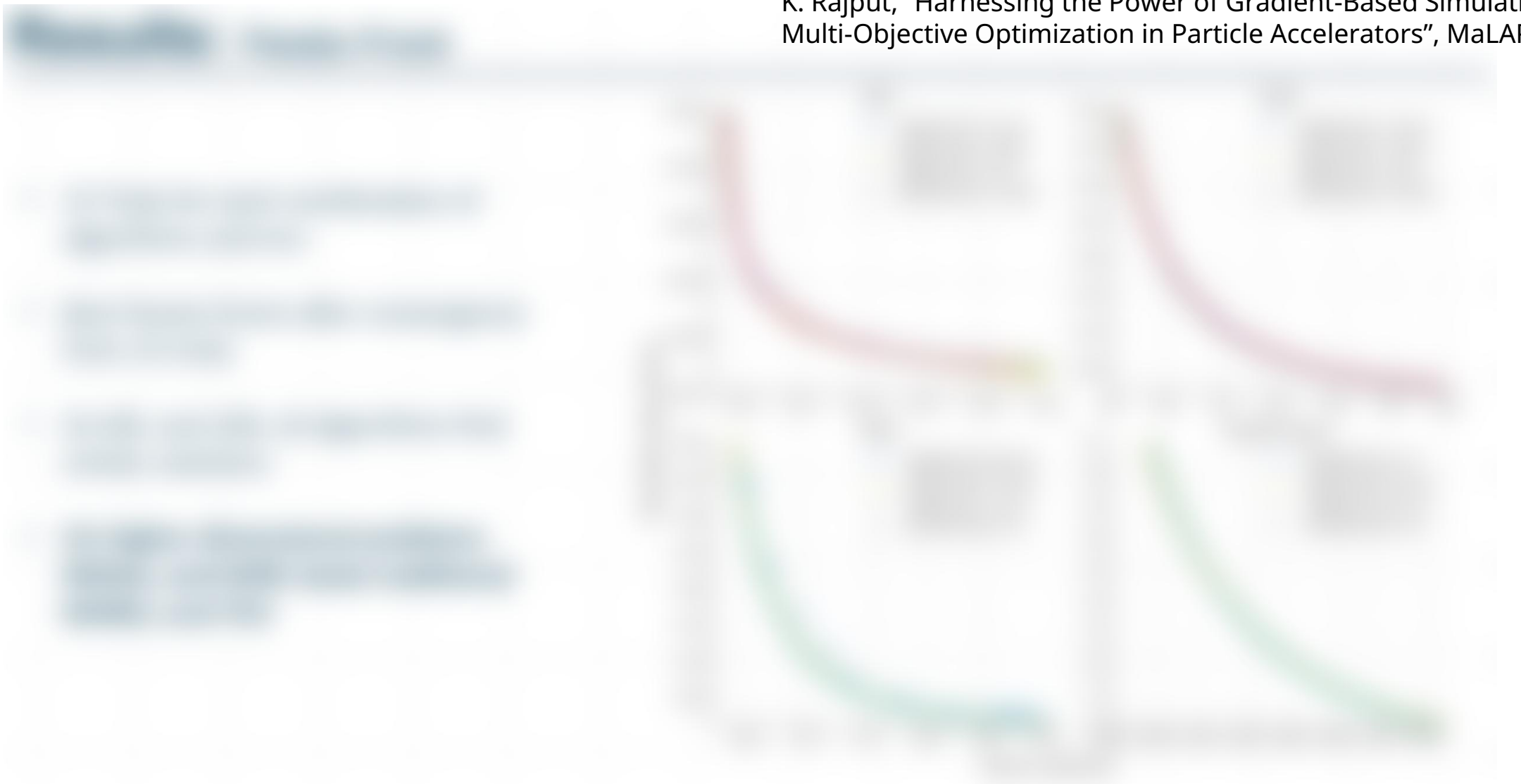
# 強化学習の応用例

K. Rajput, “Harnessing the Power of Gradient-Based Simulations for Multi-Objective Optimization in Particle Accelerators”, MaLAPA 2025

Jlab CEBAF にて熱負荷やトリップレートを抑えつつビームエネルギーを最大化する試み。

# 各アルゴリズムのパレートフロントの比較

K. Rajput, “Harnessing the Power of Gradient-Based Simulations for Multi-Objective Optimization in Particle Accelerators”, MaLAPA 2025



# 収束時間の比較

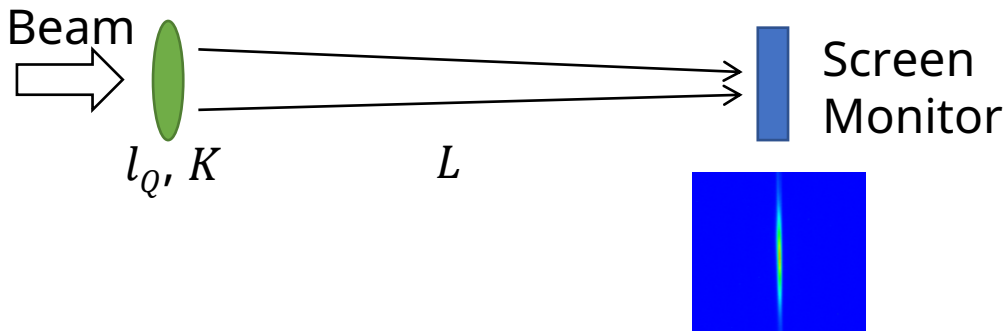
# 高効率・高次元ビーム診断

- 加速器調整においてビームの状態を正しく知ることは極めて重要。
- 各ビーム診断機器はビームの一部の特徴量に特化したものにならざるを得ない。
  - ビーム電流・電荷、ビームサイズ・プロファイル、バンチ長、など。
- 一方で、エミッタンスやベータ関数等、直接測定しづらい物理量が調整に不可欠である。
- 究極的には６次元位相空間分布がわかればトラッキングシミュレーションで追跡可能。
- 機械学習を応用することで、特に間接的にしか得られないビームの特徴量を高効率・高次元に捉えることができるようになりつつある。

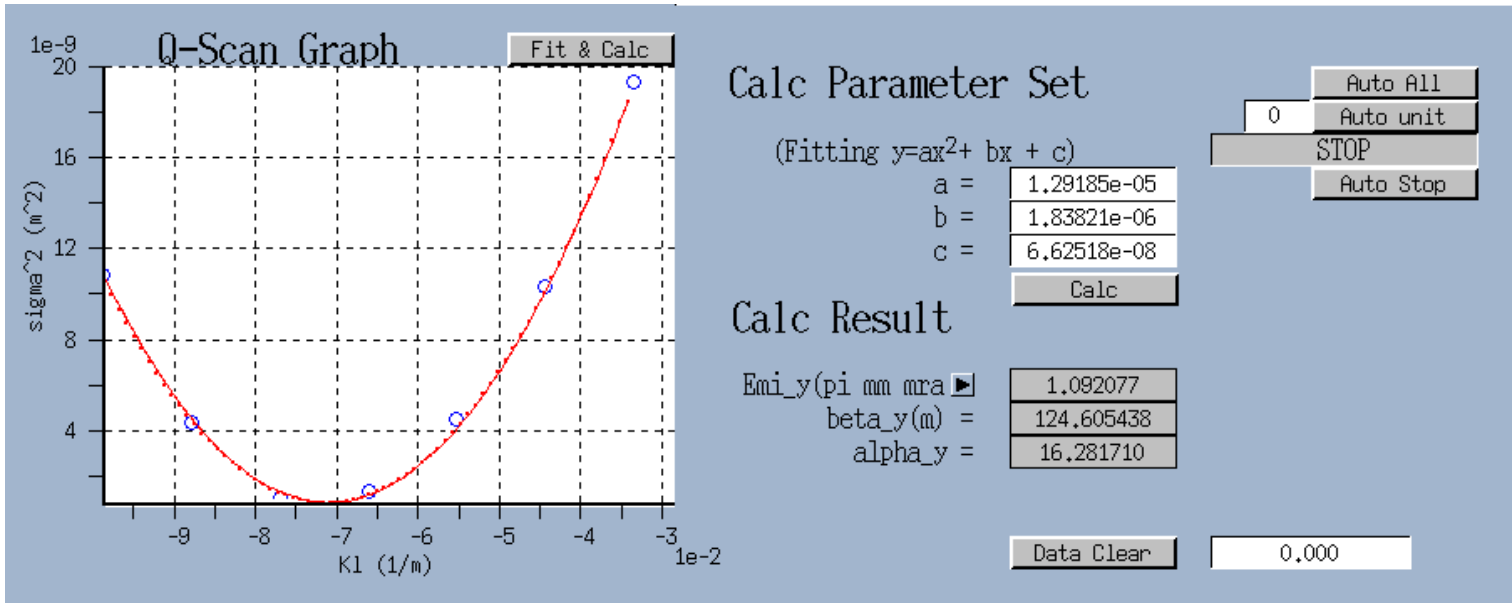
# 高効率ビーム診断・制御

- 機械学習を応用して限られたデータから効率よく物理量を抽出する。
- それをもとにビーム性能を最適化したり、ビーム制御したりする。
- 例１: Bayesian Algorithm eXecution (BAX) によるエミッタンス測定と最適化 (SLAC)
  - S.A. Miskovich *et al*., Mach. Learn. Sci. Technol. **5**, 015004 (2024). Multipoint-BAX: a new approach for efficiently tuning particle accelerator emittance via virtual objectives
  - Also presented at 3rd ICFA Beam Dynamics Mini-Workshop on Machine Learning Applications for Particle Accelerators https://www.bnl.gov/mlaworkshop2022/
- 例２: 蓄積リングの Neural Network による軌道補正
  - D. Schirmer, “First experiences with machine learning techniques at the 1.5 GeV synchrotron light source DELTA”, 2nd ICFA Mini-Workshop on Machine Learning for Charged Particle Accelerators
  - E. Meier et al., WEPPP057, IPAC’12.
- 例３: 蓄積リングの Nerural Network によるオプティクス測定
  - R. Li *et al*., Applied Sciences **13**, 8034, (2023)

# 四極スキャンによるエミッタンス・ベータ関数測定




$$\sigma_x^2 = \beta\varepsilon\left(LKl_Q + \frac{\alpha L}{\beta} - 1\right)^2 + \frac{\varepsilon L^2}{\beta}$$




- 四極スキャン (Q-scan) 法は、ガウシアンビームを仮定してエミッタンスとベータ関数 (Twiss パラメータ) を測定する一般的な方法。
- 四極磁石の強さをスキャンして下流でビームサイズを測定すると、ビームサイズの２乗が四極強さの二次関数にしたがう。
- ひととおりスキャンしないと求められないので一定の時間がかかる。
- ガウシアンではないビームの場合は誤差が出る。

# LCLS 電子銃のエミッタンス測定と最適化

S.A. Miskovich et al., Mach. Learn. Sci. Technol. 5, 015004 (2024).

四極スキャンでエミッタンスを測定しながら電子銃のソレノイド磁石等を調整してエミッタンスを最適化してきた。

時間がかかるので機械学習で効率化したい。

ガウス過程回帰でエミッタンスを推定すれば毎回細かくスキャンする必要はない。

調整ノブとエミッタンス測定用の四極を効率よく変えながら最適化すれば、測定と探索を同時に行える。

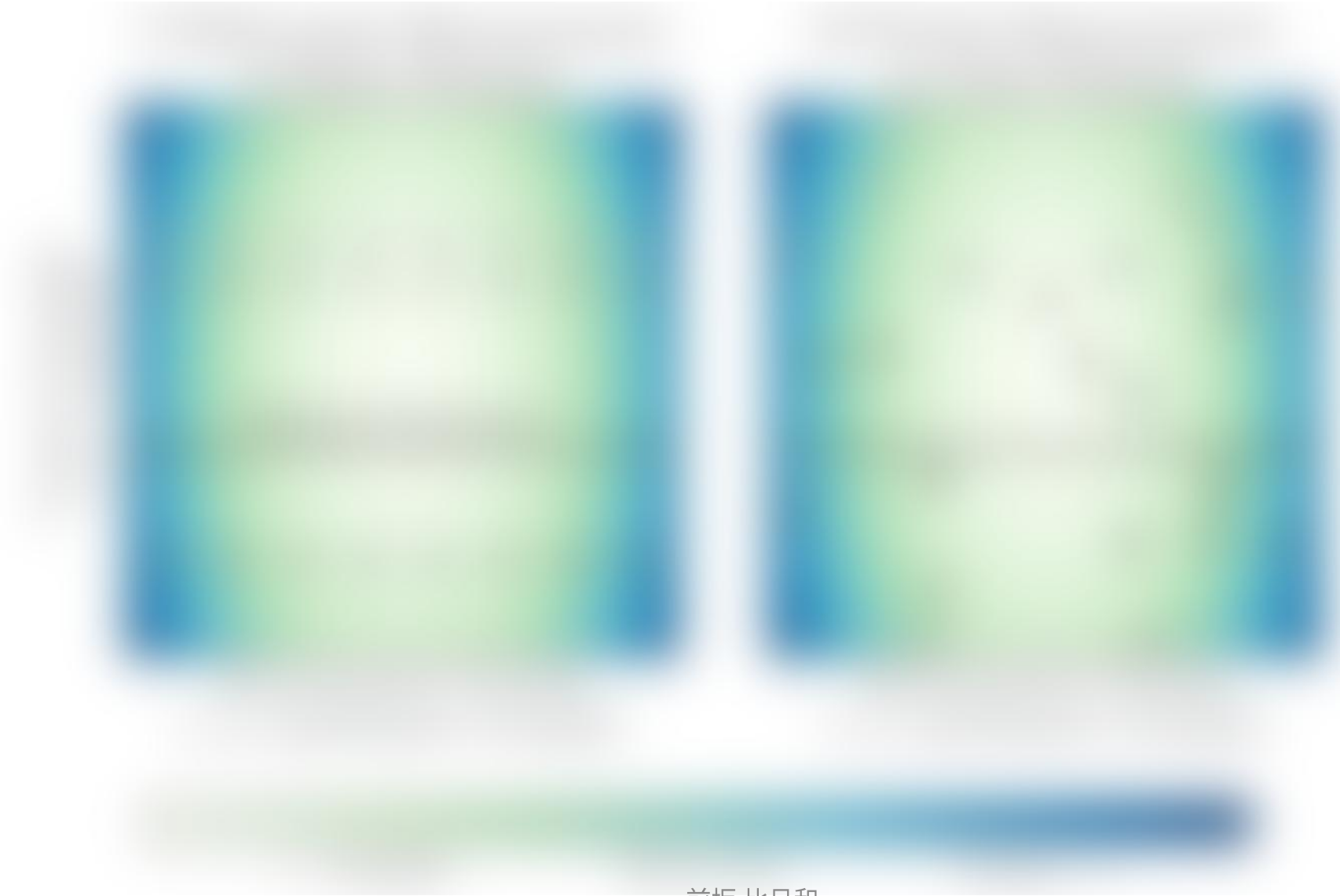
BAX: Bayesian Algorithm eXecution

BAX website
https://willieneis.github.io/bax-website/

# エミッタンスの測定と最適化を同時実行

S.A. Miskovich et al., Mach. Learn. Sci. Technol. 5, 015004 (2024).

BAX: Bayesian Algorithm eXecution



# エミッタンス最適化結果

S.A. Miskovich et al., Mach. Learn. Sci. Technol. 5, 015004 (2024).

24 分間、90 回程度の試行で人による調整より優れたエミッタンスに最適化。

複数のデータによる間接測定でしか得られないビームパラメータの測定や最適化に有効。

# 蓄積リングのビーム軌道補正

いまいちな例

- 蓄積リングの閉軌道歪み (Closed Orbit Distortion, COD) の応答は線形ビーム光学系から以下のように得られる。

$$\Delta x_i = \frac{\sqrt{\beta_i \beta_j}}{2 \sin \pi\nu} \Delta\theta_j \cos\left(\pi\nu - \left|\psi_i - \psi_j\right|\right) \equiv R_{ij} \Delta\theta_j$$

$\Delta x_i$: beam displacement at $i$
$\Delta\theta_j$: error kick at $j$
$\beta_{i,j}$: beta function at $i, j$
$\nu$: betatron tune
$\psi_{i,j}$: betatron phase at $i, j$

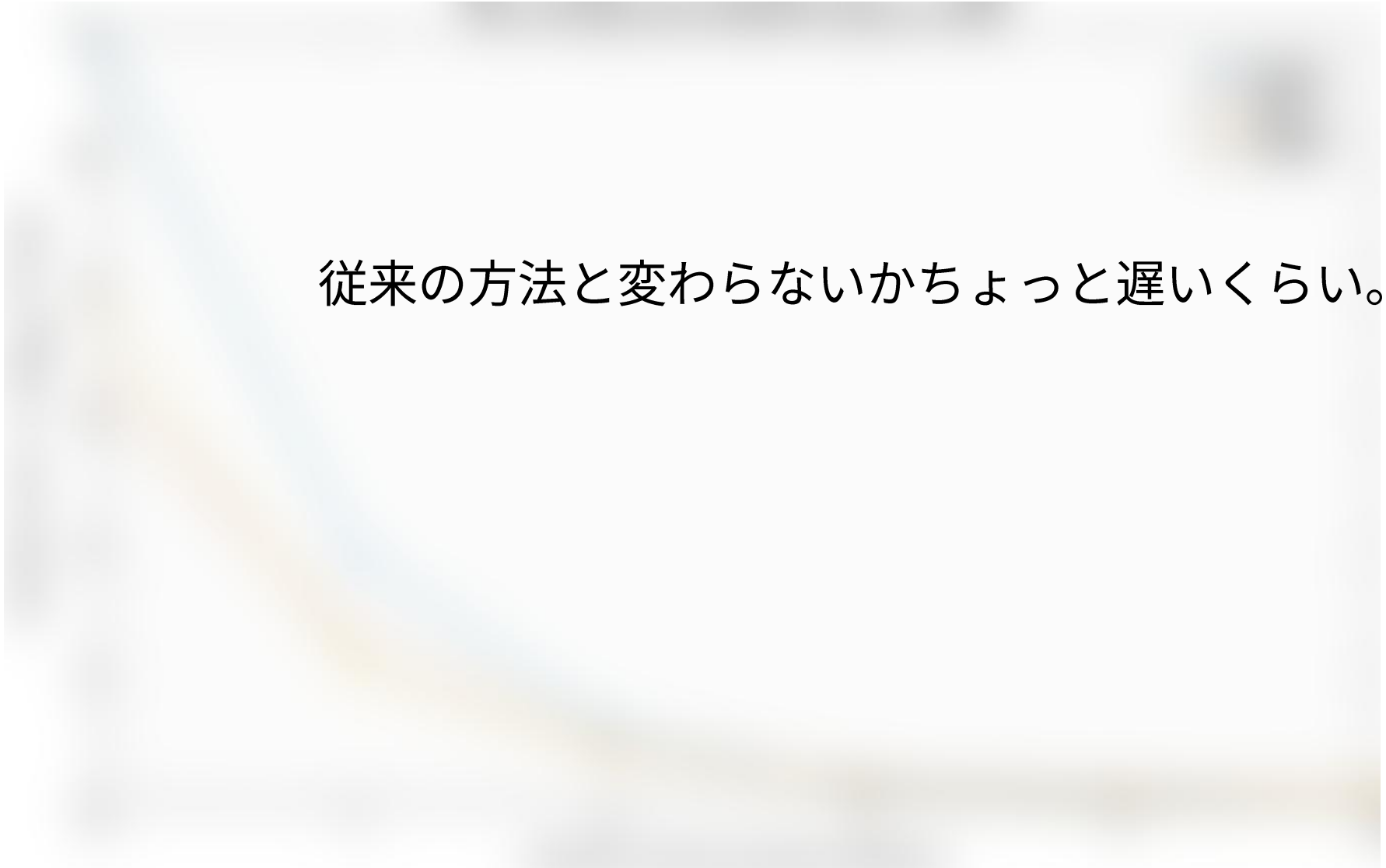
- COD 補正用の磁石の強さは以下のような応答行列 $R_{ij}^{\dagger}$ の擬逆行列 $R_{ij}^{\dagger}$ を使った古典的方法などで容易に補正できる。
  - $\Delta\theta_j = R_{ij}^{\dagger} \Delta x_i$
  - ある行列 $A$ の擬逆行列 $A^{\dagger}$： $AA^{\dagger}A = A$
- 機械学習は基本的に必要ない。
- しかし、非線形効果やビーム光学系の誤差に強くしたいとか何かの理由で機械学習を使いたがる人が出てくる。

# COD 補正にニューラルネットを適用した例

D. Schirmer, “First experiences with machine learning techniques at the 1.5 GeV synchrotron light source DELTA”, 2nd ICFA Mini-Workshop on Machine Learning for Charged Particle Accelerators

# COD 補正結果

D. Schirmer, “First experiences with machine learning techniques at the 1.5 GeV synchrotron light source DELTA”, 2nd ICFA Mini-Workshop on Machine Learning for Charged Particle Accelerators



従来の方法と変わらないかちょっと遅いくらい。

# ニューラルネットワークだが、応答を更新しながら COD 補正

E. Meier et al., WEPPP057, IPAC'12.



- Actor-Critic 制御方式による適応学習
  - Action ニューラルネットワークが COD 補正。
  - Critic ニューラルネットワークが COD から損失関数 $J_k$ を求める。
- どちらのネットワークも $J_k$ を減らすように毎回更新。
- このように、随時追加学習するならまだまし。
- ニューラルネットワークでないといけないかは疑問。ベイズの方がわかりやすいかも。

# シミュレーション結果

シミュレーションのみで実験はしていない

E. Meier et al., WEPPP057, IPAC’12.

ニューラルネットワークは最初は未学習。

数百回の試行後に COD 補正がうまく働くようになっている。

損失関数も 0 に収束。

このような方法なら応答関数自体も補正してくれるので機械学習らしいと言えばそのとおり。

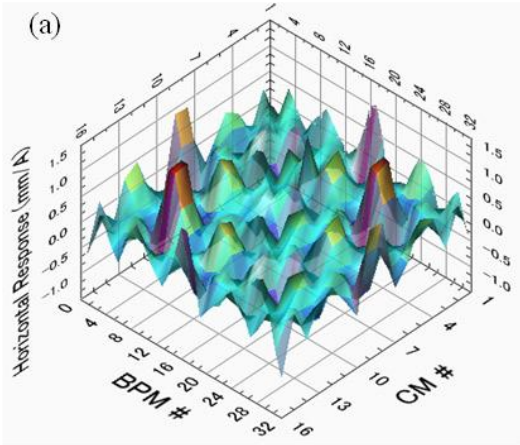
応答行列をベイズ推定で更新する方がわかりやすそう。
誰かもうやっていそうだがわたしはまだ知らない…

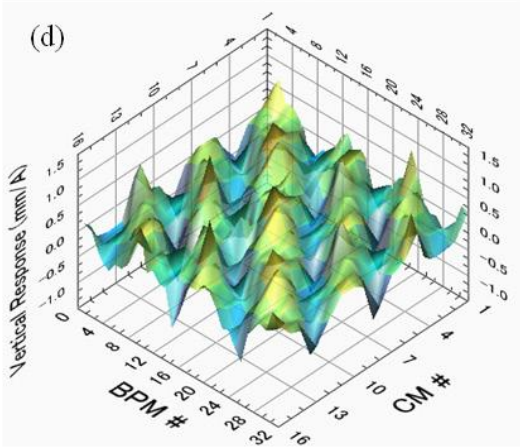
# 蓄積リングの線形光学系の測定

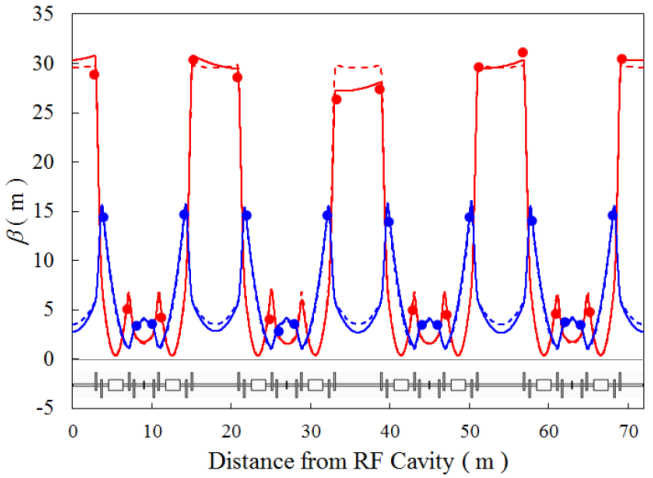
- 蓄積リングの線形光学系パラメータ (ベータ関数) は閉軌道応答 (COD response) で測定・解析することが多い。

$$\Delta x_i = \frac{\sqrt{\beta_i \beta_j}}{2 \sin \pi\nu} \Delta\theta_j \cos\left(\pi\nu - \left|\psi_i - \psi_j\right|\right)$$

- LOCO (Linear Optics from Closed Orbit) など。
- この方法は補正磁石の応答を多数取得する必要があり、時間がかかる。
- 機械学習を応用することで効率化が図れるかも。
  - 少ないデータセットで十分な精度を得るとか。
  - 効率的に測定できる補正磁石の組み合わせとか。
  - 光学系補正と測定を同時並行的に行うアルゴリズムとか。
- この辺はまだいろいろと宝が埋まっている可能性。



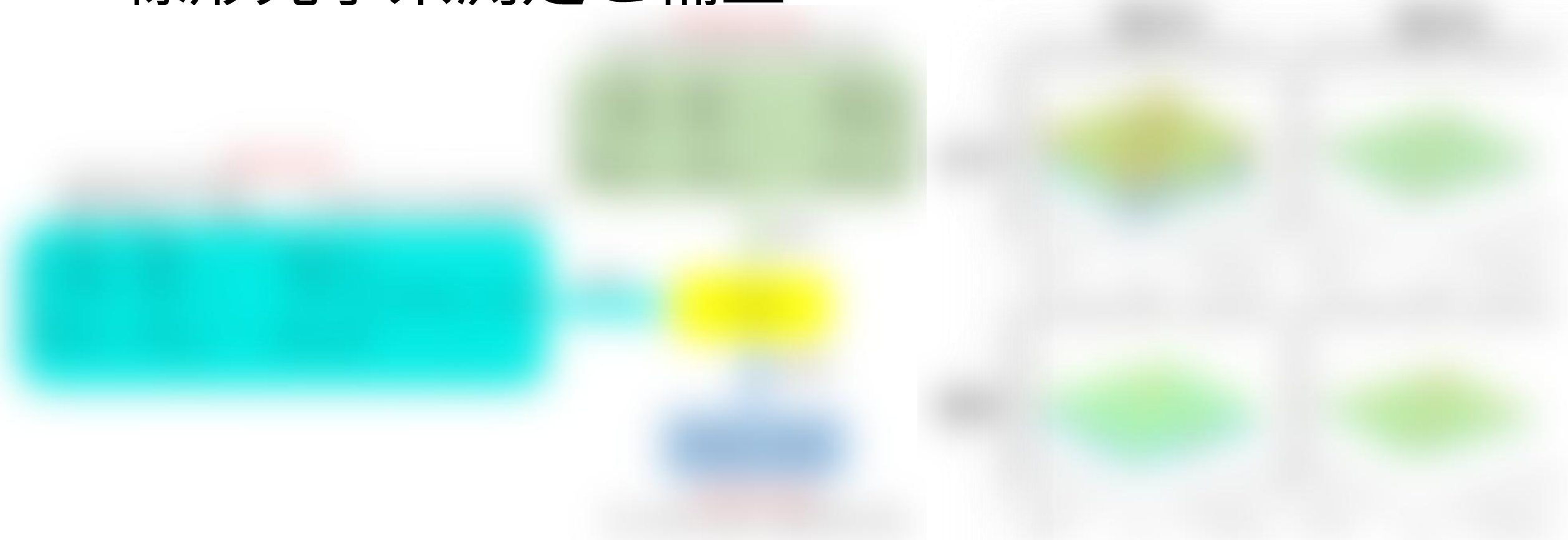




T. Takano et al., 11th PASJ Meeting, SAP017. (2014)

# 畳み込みニューラルネット (CNN) を使った線形光学系測定と補正

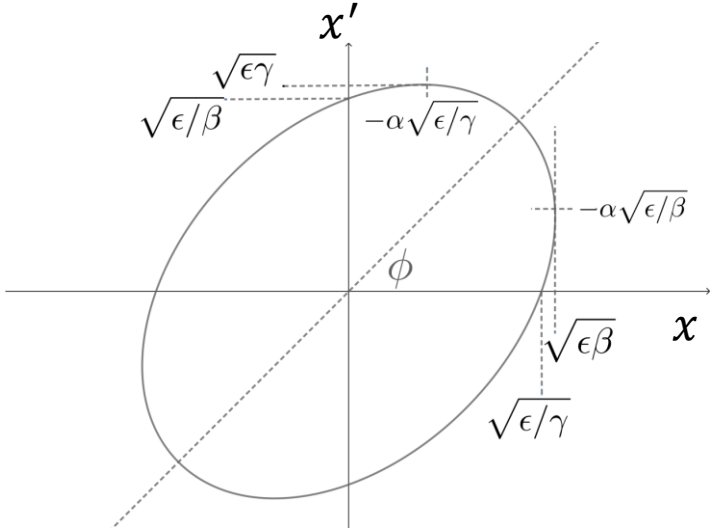
R. Li *et al*., Applied Sciences **13**, 8034, (2023)



- 入力データは (BPM x 補正磁石) の２次元のため、Convolutional Neural Network (CNN) を使ったようだ。
- 確かに補正はできるようだが、速くなるとかそういうメリットは感じられない。
- 機械学習を使うなら効率化とか高速化とかのメリットがほしい。

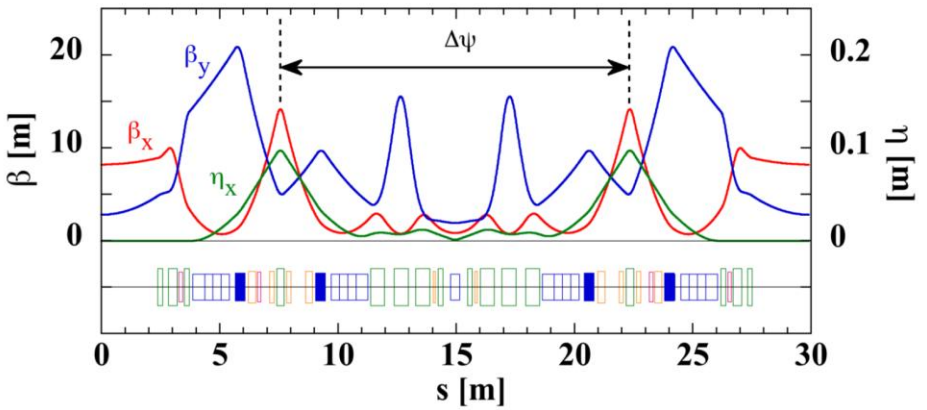
# 多次元位相空間分布の測定

- 多次元の位相空間分布を測定・再構成するのは簡単なことではない。
- ただ、ガウシアンビームを仮定すれば比較的容易に位相空間分布を推定することができる。
  - ガウシアンビームであれば位相空間分布はエミッタンス $\epsilon$ と Twiss パラメータ $(\alpha, \beta, \gamma)$ で表現できる。
  - ガウシアンビームの時間発展や輸送は転送行列等を用いて簡単に予測できる。
- この近似は蓄積リングではよく成り立つが、XFEL のような高性能な線型加速器では成り立たないことがある。
  - 蓄積リングは周期境界条件により定常状態に収束するというありがたい性質がある。




By PianoDan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=114242040

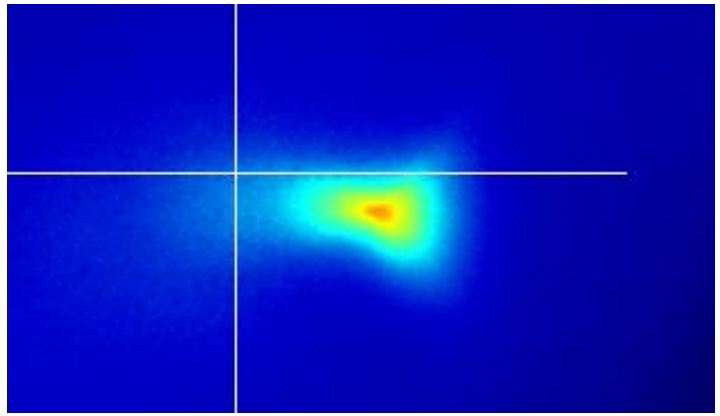
SPring-8-II



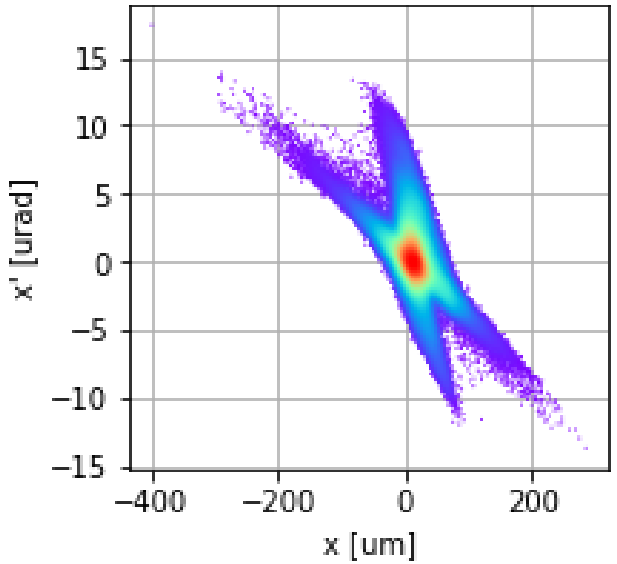
# 複雑な位相空間分布の測定

- 線型加速器ではガウシアンビームによる近似はしばしば破綻する。
  - 位相空間分布は初期状態に依存するうえ、さまざまな加速器コンポーネントの影響を受けて歪んでいく。
- XFEL のような近年の高精度な線型加速器では正確な位相空間分布の情報が性能向上に直結する。
- 加速器の振る舞いやビームの挙動を理解するために、シミュレーションで得られる位相空間分布を測定結果と比較したくなる。
- ある点での位相空間分布があれば、運動方程式に基づいて上下流に転送することも可能。
- 複雑な位相空間分布を推定・再構成する手法として機械学習を応用する研究が盛んに行われている。

Beam profile at SACLA

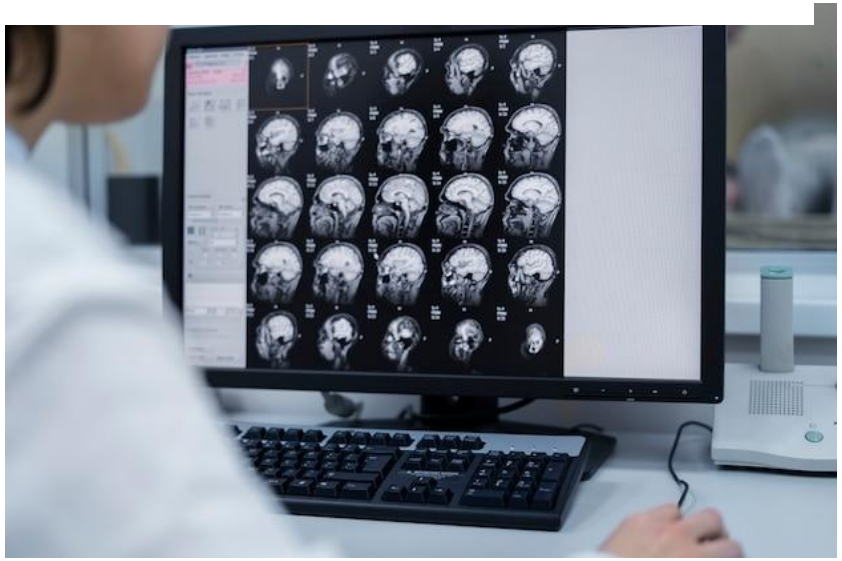


Simulated phase-space distribut



# 位相空間トモグラフィー

- 従来型の位相空間分布の測定方法に位相空間トモグラフィがある。
  - 医療用 CT と同じ考え方。
- 複数の四極磁石をうまくスキャンして位相空間内のビームを回転させてプロファイルを取得する。
- それを解析して位相空間分布を再構成する。
  - M. Röhrs *et al*., Phys. Rev. ST Accel. Beams **12**, 050704 (2009).
  - M. Scholz and B. Beutner (DESY), Proc. IPAC'17, MOPAB047.
  - G. Ricci *et al*., Phys. Rev. Accel. Beams **27**, 093001 (2024).
- 測定が複雑で時間がかかるとともに、解析も大変なので日々の運転に使うのは難しいのが現実。

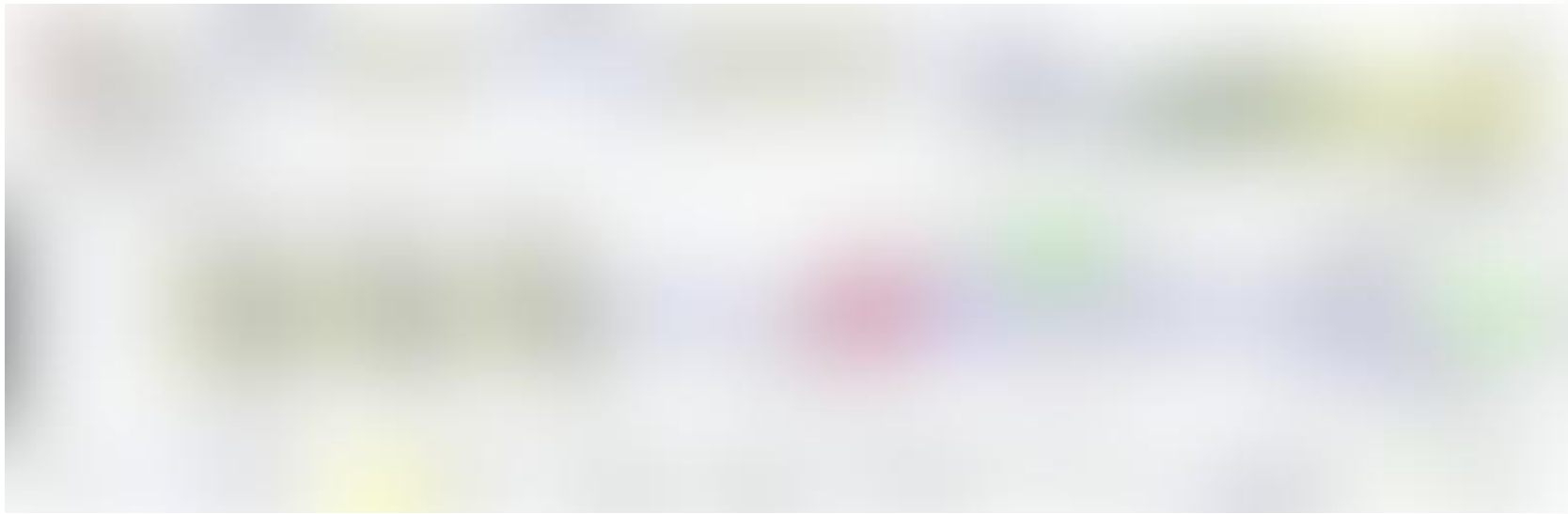


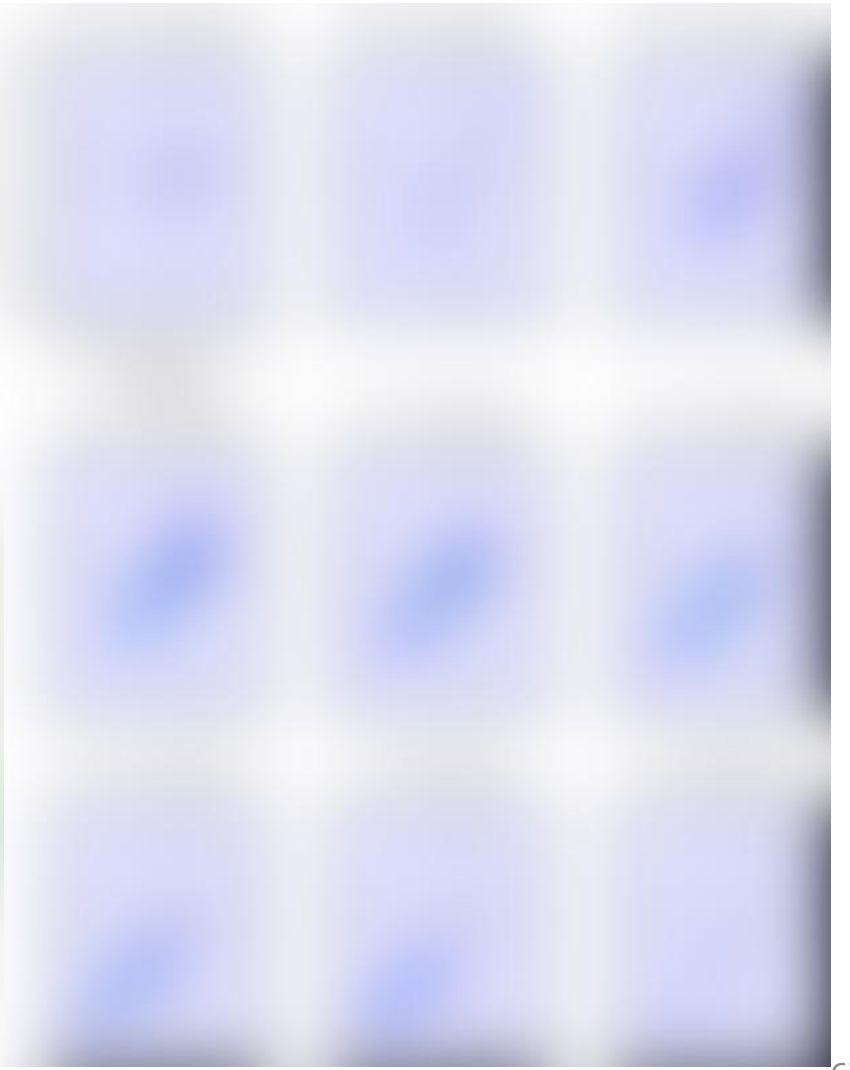
# 時間スライスごとの位相空間トモグラフィの例

M. Röhrs *et al*., Phys. Rev. ST Accel. Beams **12**, 050704 (2009).

- DESY の FLASH での測定例。
- ベータトロン位相を複数の四極磁石でスキャンし、かつ、RF ディフレクタで時間方向にも引き延ばす。
- そのビームプロファイルをスクリーンモニタで取得する。
- 得られたデータを解析して位相空間分布を再構成する。

再構成されたスライスごとの位相空間分布





# 機械学習を用いた例

- Ryan Rousell *et al*., Phys. Rev. Lett. **130**, 145001 (2023).
  Phase Space Reconstruction from Accelerator Beam Measurements Using Neural Networks and Differentiable Simulations

- Alexander Scheinker *et al*., IPAC'22 TUOXGD3.
  6D Phase Space Diagnostics Based on Adaptively Tuned Physics Physics-Informed Generative Convolutional Neural Networks

- Owen Convery *et al*., Phys. Rev. Accel. Beams **24**, 074602 (2021).
- Adi Hanuka *et al*., Scientific Reports **11**, 2945 (2021).
  Accurate & Confident Prediction of Electron Beam Longitudinal Properties using Spectral Virtual Diagnostics

# 位相空間再構成パイプライン

画像生成系 AI とよく似たアルゴリズム構造

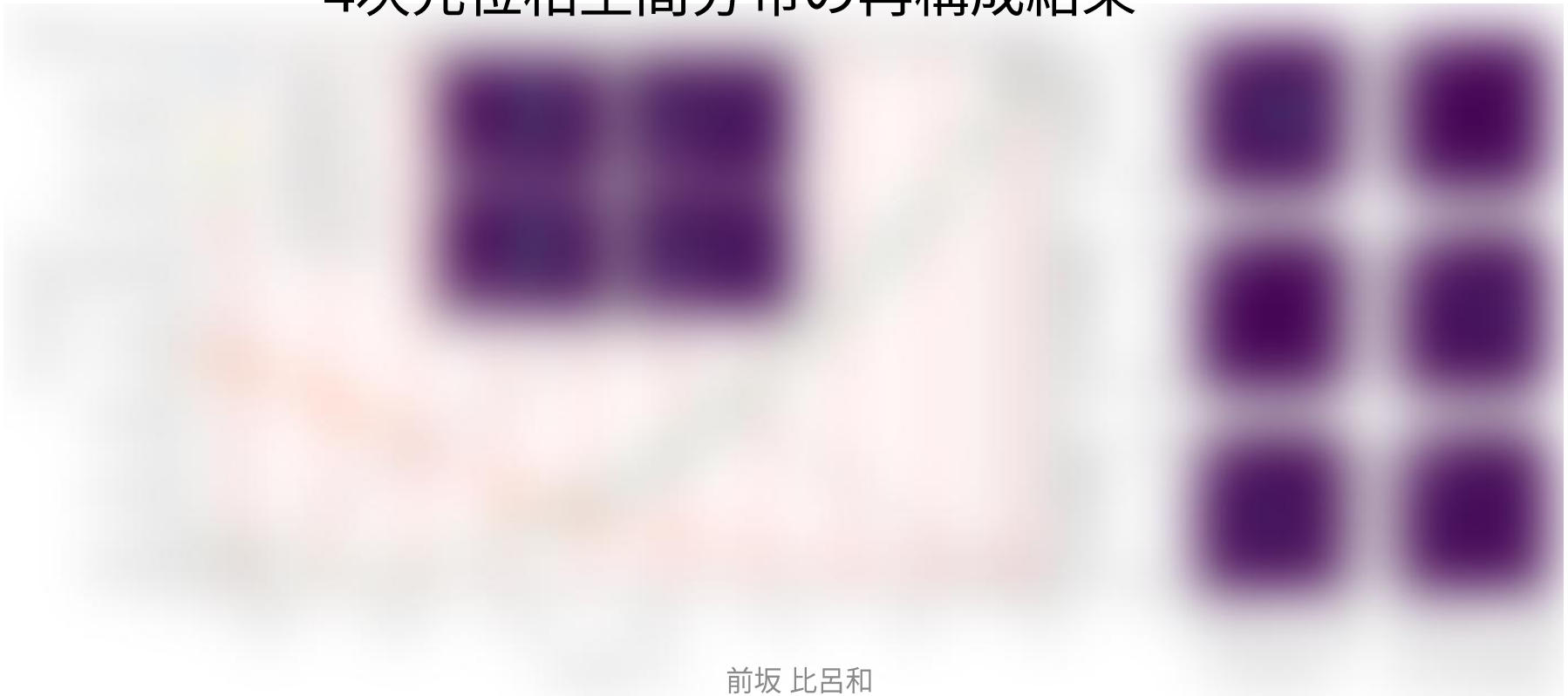
Ryan Rousell et al., Phys. Rev. Lett. **130**, 145001 (2023).

# AWA での実験結果

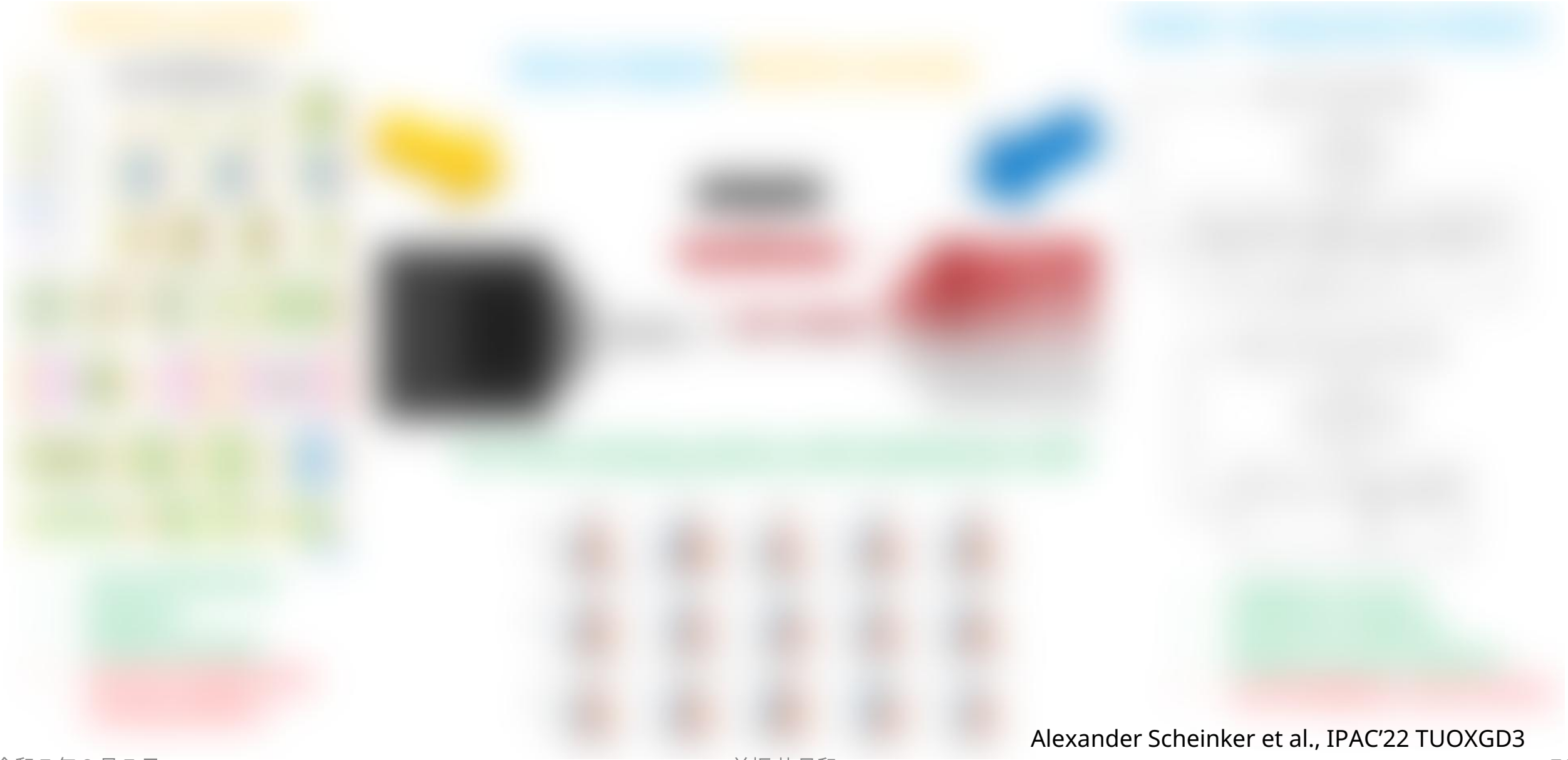
AWA: Argonne Wakefield Accelerator

Ryan Rousell et al., Phys. Rev. Lett. **130**, 145001 (2023).

4次元位相空間分布の再構成結果



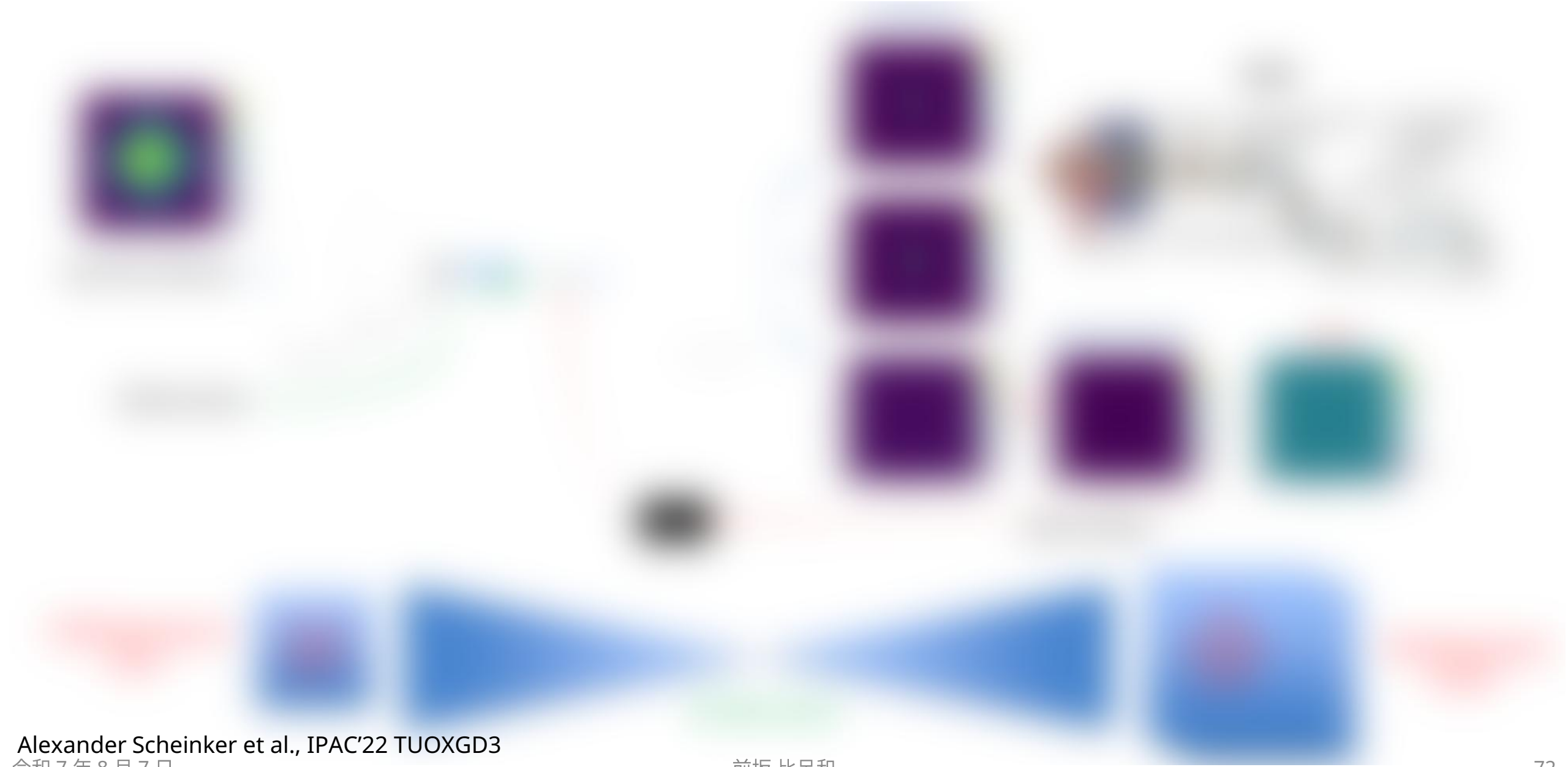
# 物理モデルを学習した CNN とモデルに依存しない適応機械学習による位相空間分布の測定



Alexander Scheinker et al., IPAC’22 TUOXGD3

# HiRES での試験結果

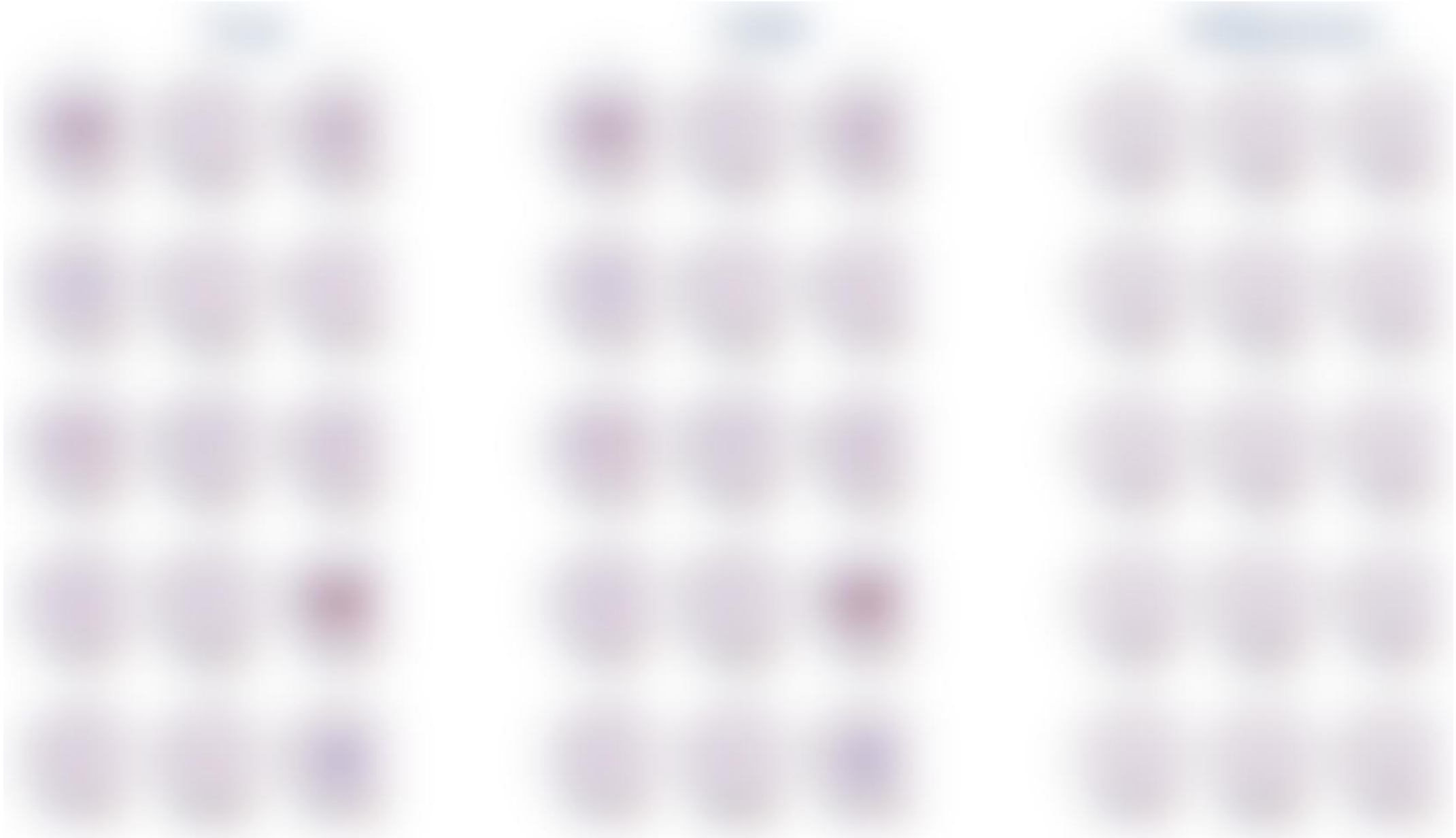
Compact Ultra-fast Electron Diffraction Facility



Alexander Scheinker et al., IPAC’22 TUOXGD3

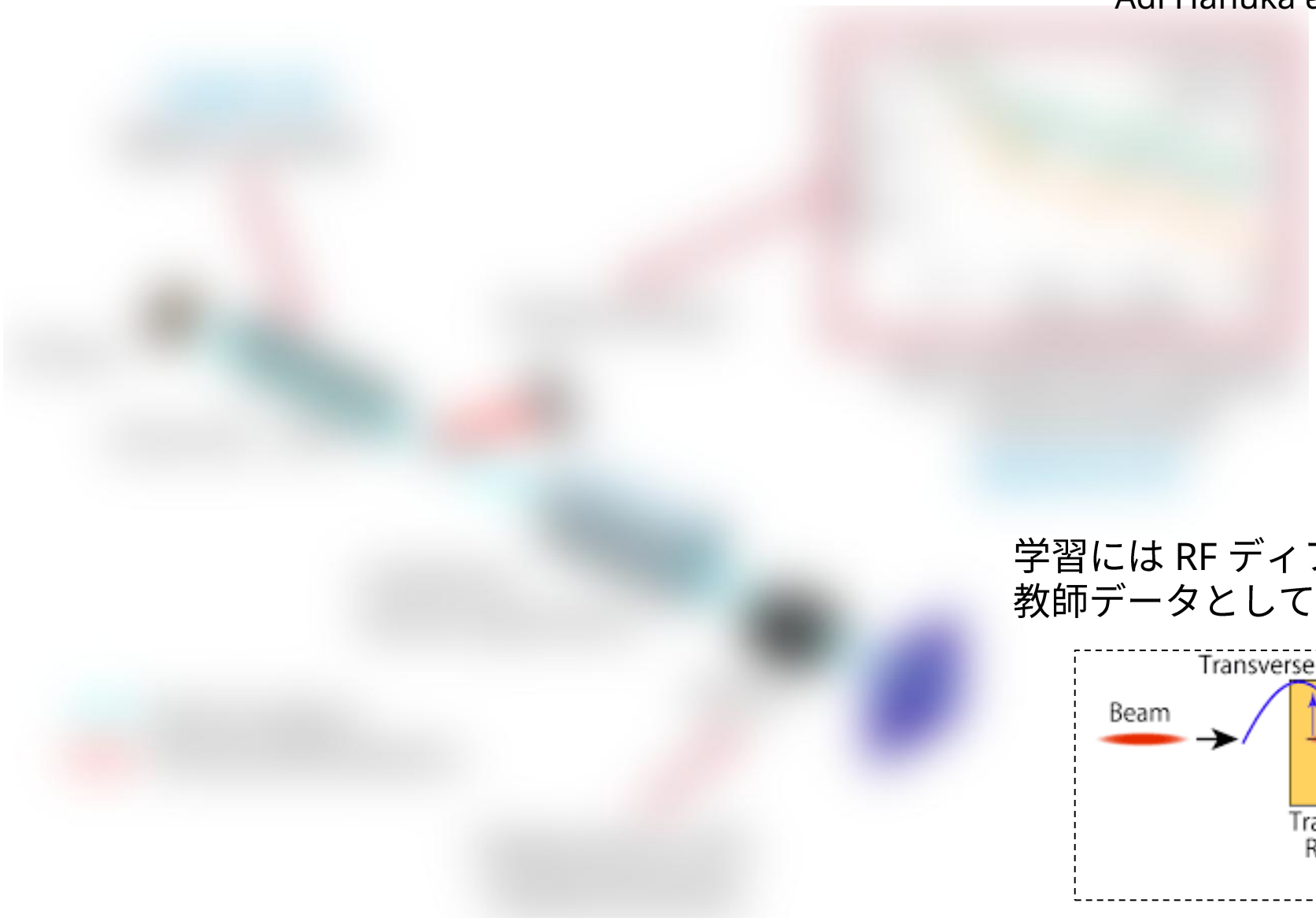
# ６次元位相空間分布の２次元への射影

Alexander Scheinker et al., IPAC’22 TUOXGD3



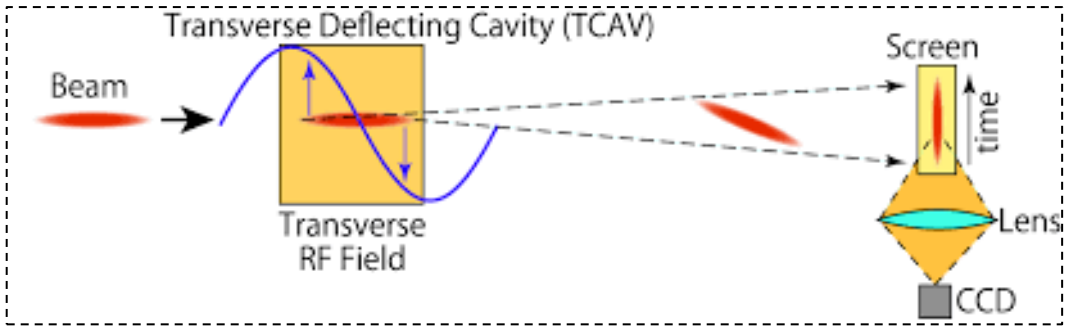
# 縦方向位相空間の機械学習による仮想診断

Adi Hanuka *et al*., Scientific Reports **11**, 2945 (2021).



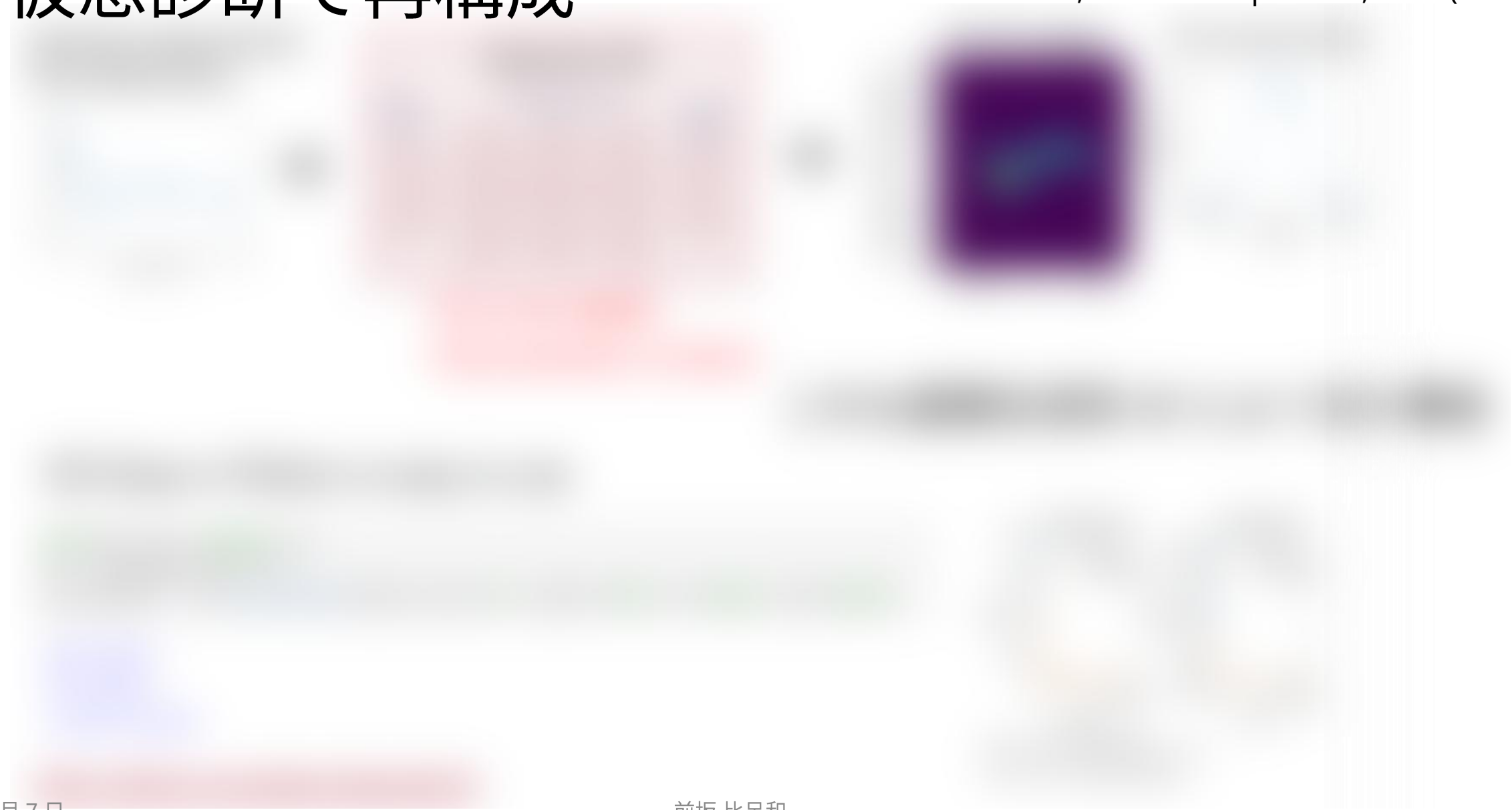
コヒーレント放射のスペクトルから
縦方向位相空間分布を再構成

学習には RF ディフレクタ (TCAV) の測定結果を
教師データとして使用



# 1次元スペクトルから2次元縦方向位相空間を仮想診断で再構成

Adi Hanuka *et al*., Scientific Reports **11**, 2945 (2021).



# LCLS での測定結果

Adi Hanuka *et al*., Scientific Reports **11**, 2945 (2021).

電流分布

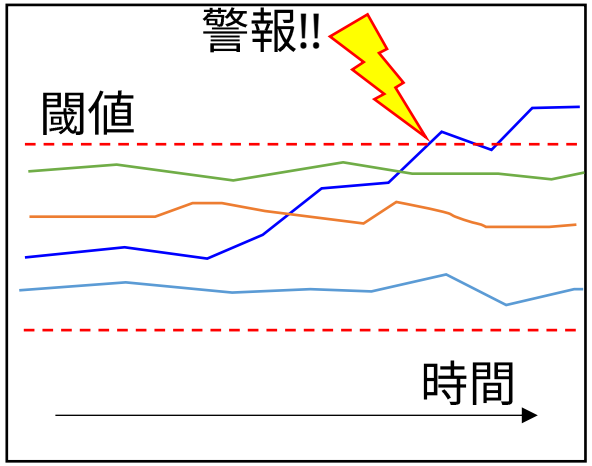
縦方向位相空間分布

このように、位相空間分布の再構成のような複雑なタスクにはニューラルネットワークが有効のようである。

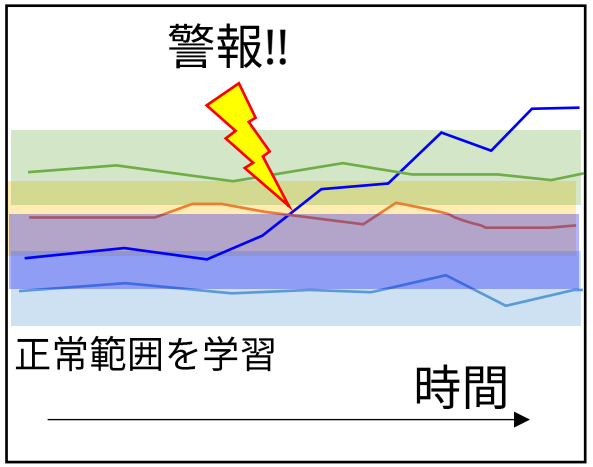
# 異常検知・故障予知

- 加速器の安定運転や稼働率の最大化に対する期待は年々高まっている。
- 安定稼働のためには機器の故障を未然に防ぐことが重要。
- 旧来の機器保護手法では、各機器が異常を示した時点で動作を停止し、加速器全体を停止することが多かった。
- 動作停止を伴う異常に至る前に兆候を捉えるため、計器の値を収集し、あらかじめ決められた閾値を超えた際などに警告を出すシステムが使われるようになった。
- これだけでもかなり安定な運転に貢献があった。
  - SPring-8 の稼働率 > 99.5 %
- さらなる稼働率の向上に向けた対策が求められるようになりつつある。
- まだまだ不具合の原因となる機器の状態監視ができていない項目が多くあるため、監視対象を広げることも重要である。
- 一方で、監視対象が増えるにつれてすべてに細かく警告の条件を付けていくことが困難になってきている。
- また、同じ機器でも個体差があるものも存在する。
- 監視データに対して正常範囲を学習し、それを逸脱した際に警告を出すようにすればいいのではないか。
- 状態変化を早く検知することで予防的、計画的な対策がとれる。
- このような異常検知・故障予知に対する機械学習の応用について考える。

古典的な異常検知




機械学習による異常検知

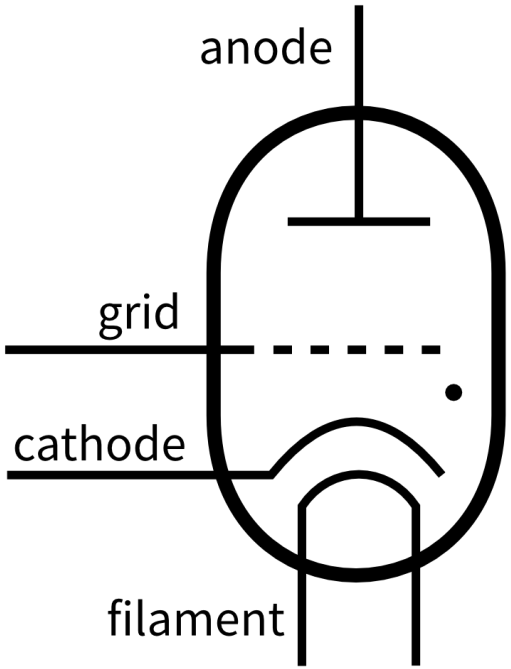


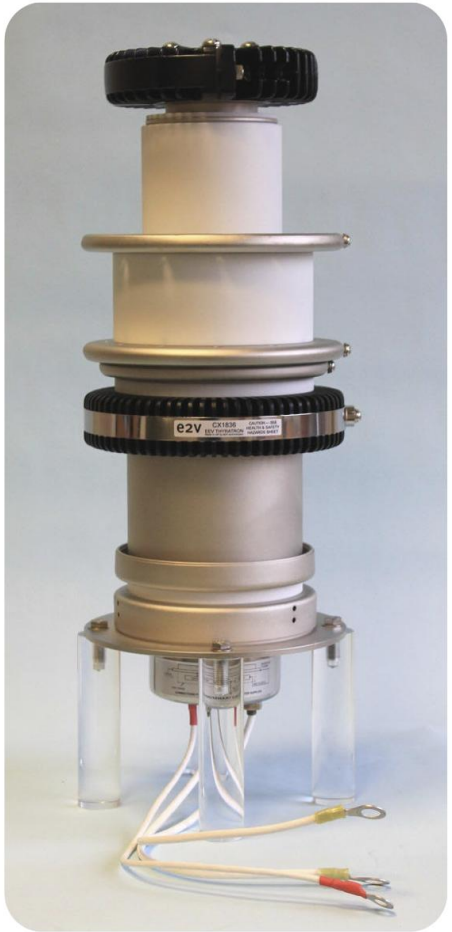
# 異常検知のいろいろ

- 機械学習による異常検知だけでも多くのアルゴリズムがある。
  - クラスタリングを行い、多数を占めるものを正常、少数を異常とする。
  - 正常データの確率分布を推定し、その分布から有意に外れたデータを異常とする。
  - 次元削減と再構成を行い、うまく再構成できないものを異常とする。
  - などなど。
- 加速器には数多くの構成機器があり、異常の兆候もさまざま。
  - 時系列データに兆候が出ることがほとんど。
    - 製造現場の検査時などでは個々の製品の状態を見る必要があるが、加速器は一度組みあがったシステムなので個々の状態の差は気にしないことが多い。
  - 正常値が運転条件により異なることがある。
  - 同じ機器でも正常範囲が異なることがある。
  - 閾値をゆっくり超えただけなら問題ないが急激に超えると異常。
  - などなど。
- 異常検知には異常の兆候に合わせたデザインが重要。
- ここでは線型加速器のパルス高周波源の大電力スイッチに使用するサイラトロンの異常検知の事例を紹介する。

# サイラトロン (Thyratron)

- サイラトロンはガス封入型の熱陰極管で、高電圧・大電力スイッチとして用いられる。
- アノード・カソード間に高電圧がかかった状態でグリッドにトリガパルスが印加されると放電によりガスがプラズマ状態になり管が導通する。
- SACLA では約 80 本のサイラトロンが使用されている。
- サイラトロンの平均寿命は約 20,000 時間ほどで、他の大電力機器と比べて短い。
- サイラトロンの寿命による不具合により SACLA の運転がしばしば中断する。
- 不具合の兆候が見られたサイラトロンを計画的にメンテナンスすることが安定運転に重要。
- しかしながら、高価な機器のため、寿命ぎりぎりまで使い切りたい。
- 主なサイラトロンの故障モード
  - 自己放電の多発
  - グリッドからの逆流電圧過大。

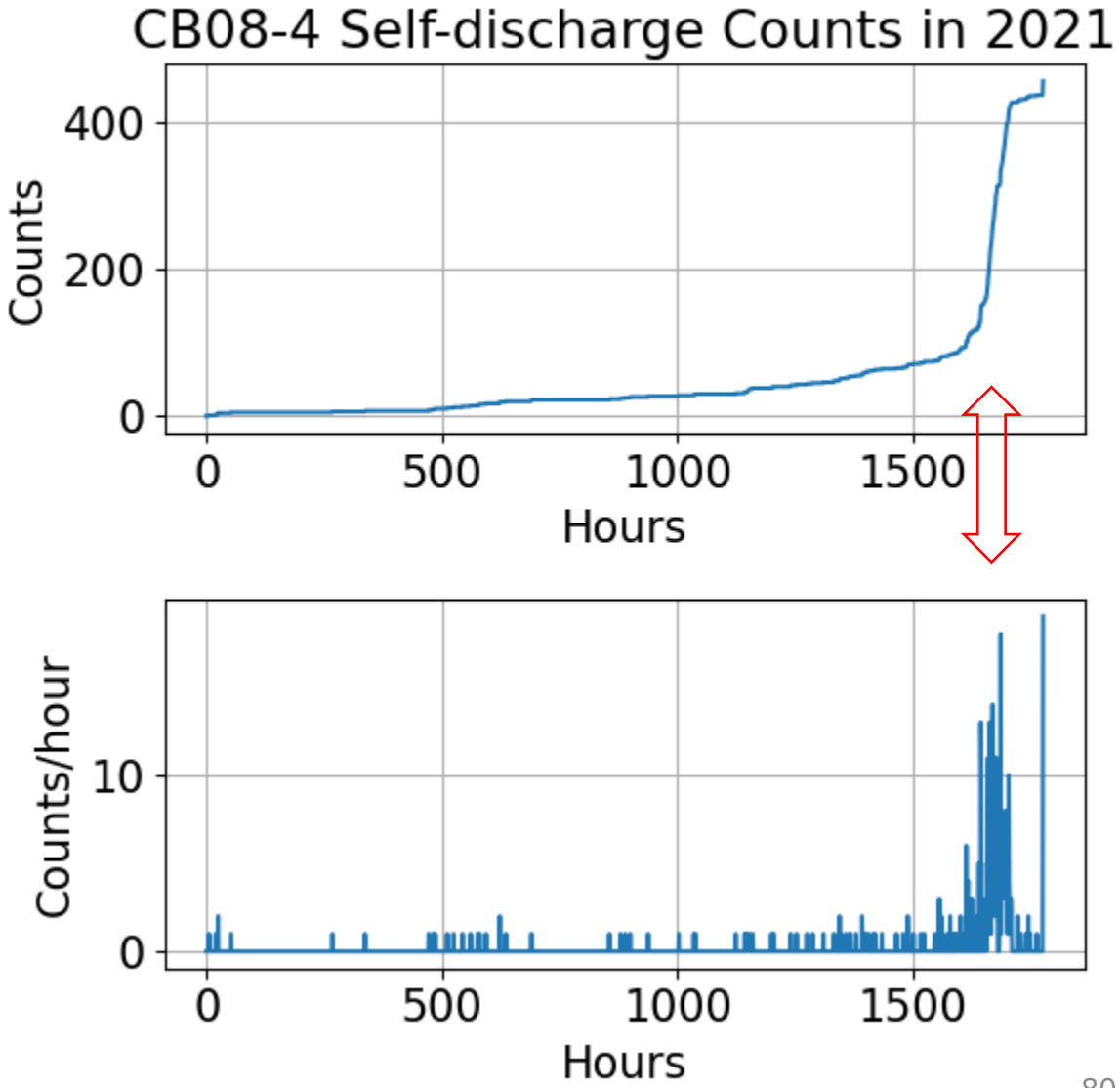



Teledyne e2v CX1836
https://www.teledyne-e2v.com/en-us/solutions/rf-power/rf-devices/thyratrons

# サイラトロンの自己放電回数増加の検知

- サイラトロンはトリガが来る前に自己放電してしまうことがたまに起こる。
- サイラトロンが寿命を迎えた際に、自己放電が多発することがある。
- 自己放電の確率には個性があり、一律に警告の閾値を設けることが難しい。
- 右図の例では正常時で１時間当たり 0.03 回程度。
- 横軸 1700 時間当たりで自己放電回数が急激に増えて寿命と判断された。



# 自己放電確率のベイズ学習

- 自己放電確率はポアソン分布に従うと仮定する。

$$P(n;\lambda) = \frac{\lambda^n e^{-\lambda}}{n!}$$

- パラメータ $\lambda$ の事前分布は、ポアソン分布の共役事前分布であるガンマ分布とする。

$$f(\lambda;\alpha,\beta) = \frac{\beta^\alpha \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)}$$

- ある程度のデータ $N = (n_1, \cdots, n_T)$を取得した際の事後分布は以下のようになる。

$$p(\lambda|N) = \frac{[\prod_{t=1}^T P(n_t;\lambda)]\, f(\lambda;\alpha,\beta)}{p(N)} = \frac{\beta^\alpha \lambda^{\alpha-1+\sum_{t=1}^T n_t} e^{-(\beta+T)\lambda}}{[\prod_{t=1}^T n_t!]\Gamma(\alpha)\, p(k_1)}$$

- 分母の周辺尤度 $p(k_1)$ は以下のように計算できる。

$$p(k_1) = \int_0^\infty \left[\prod_{t=1}^T P(n_t;\lambda)\right] f(\lambda;\alpha,\beta)\, d\lambda = \frac{\beta^\alpha}{[\prod_{t=1}^T n_t!]\Gamma(\alpha)} \int_0^\infty \lambda^{\alpha-1+\sum_{t=1}^T n_t} e^{-(\beta+T)\lambda} d\lambda$$

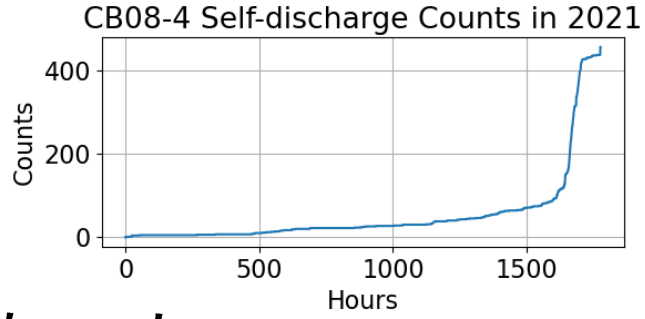
$$= \frac{\beta^\alpha \Gamma(\alpha + \sum_{t=1}^T n_t)}{[\prod_{t=1}^T n_t!](\beta+T)^{\alpha+\sum_{t=1}^T n_t}\Gamma(\alpha)}$$

- したがって、$\lambda$ の事後分布は以下のように求まる。

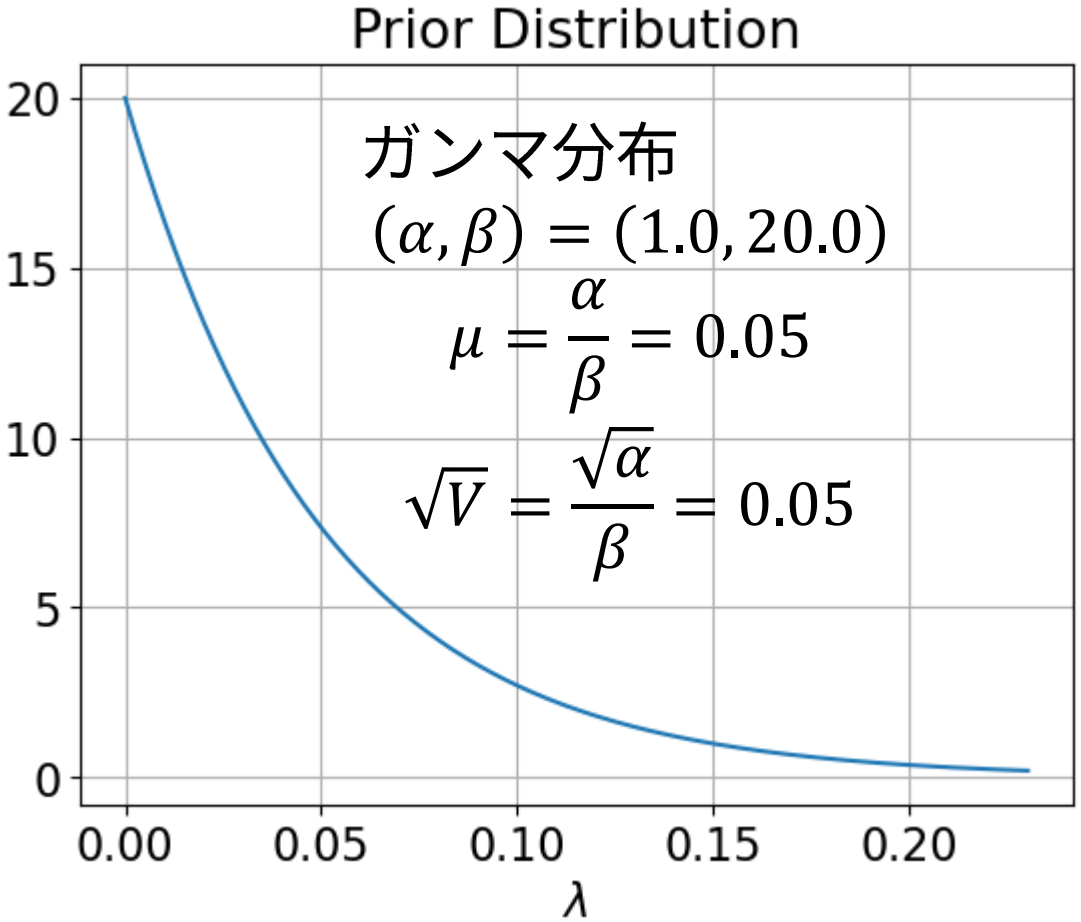
$$p(\lambda|N) = \frac{(\beta+T)^{\alpha+\sum_{t=1}^T n_t} \lambda^{\alpha-1+\sum_{t=1}^T n_t} e^{-(\beta+T)\lambda}}{\Gamma(\alpha+\sum_{t=1}^T n_t)} = f(\lambda;\hat{\alpha},\hat{\beta}), \qquad \hat{\alpha} = \alpha + \sum_{t=1}^T n_t, \qquad \hat{\beta} = \beta + T$$

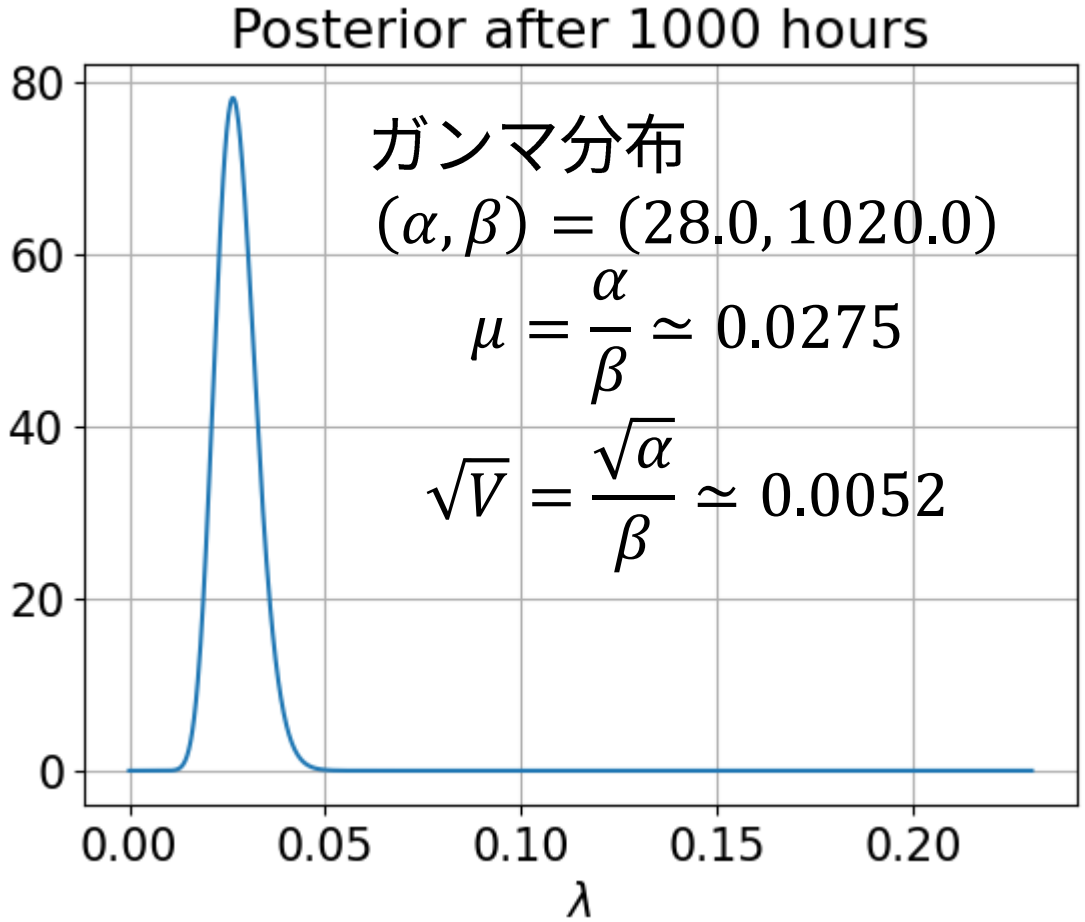
- このように、$\lambda$ の事後分布はパラメータが更新されたガンマ分布となる。

# 事前分布と事後分布




- 事前分布のパラメータを $(\alpha, \beta) = (1.0, 20.0)$ とした。
- 1000 時間後の事後分布は $(\alpha, \beta) = (28.0, 1020.0)$ となった。



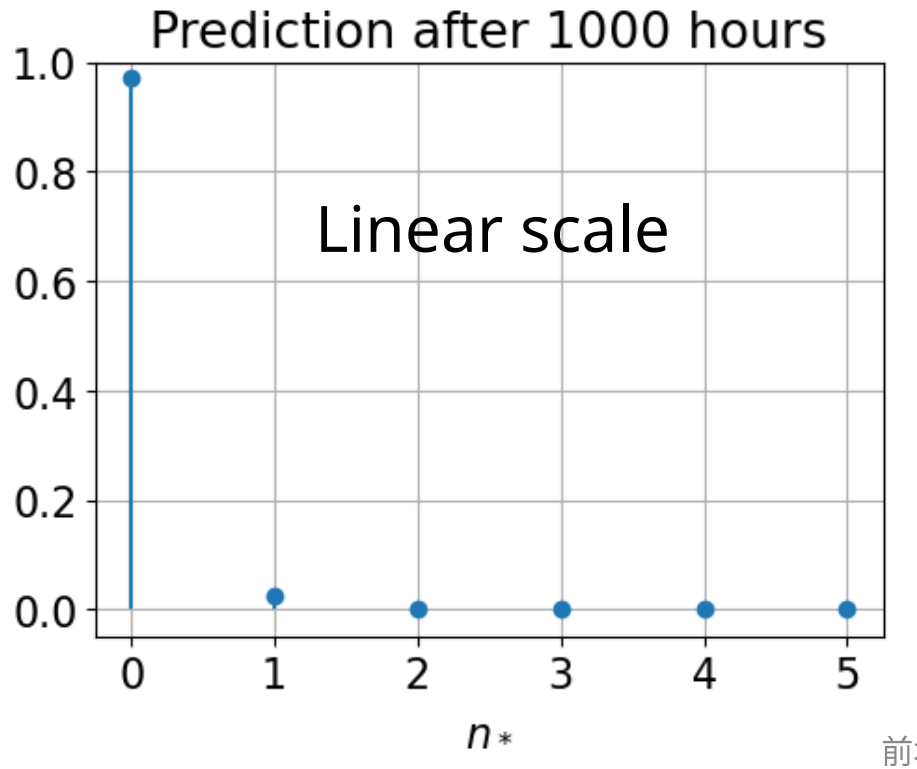
# 予測される自己放電確率

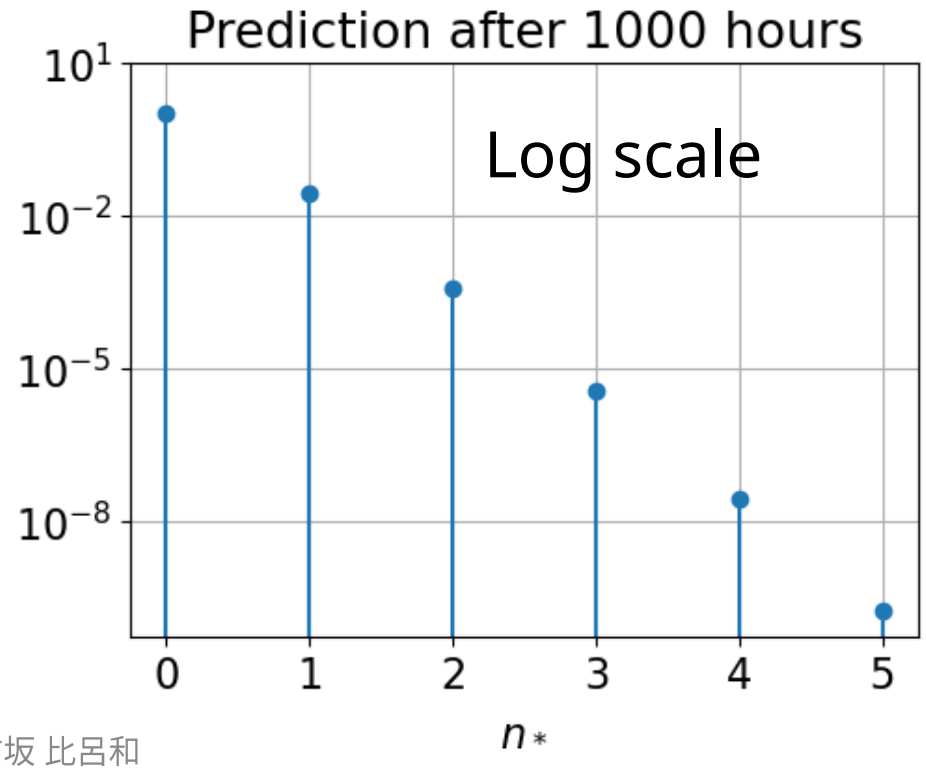
- 自己放電回数 $n_*$ の確率分布は以下の式のようになる。

$$p(n_*) = \int_0^\infty p(n_*|\lambda)\, p(\lambda) d\lambda = \frac{\hat{\beta}^{\hat{\alpha}}}{n_*!\,\Gamma(\hat{\alpha})} \int_0^\infty \lambda^{\hat{\alpha}-1+n_*} e^{-(\hat{\beta}+1)\lambda} d\lambda$$

$$= \frac{\hat{\beta}^{\hat{\alpha}}\Gamma(\hat{\alpha}+n_*)}{n_*!\left(\hat{\beta}+1\right)^{\hat{\alpha}+n_*}\Gamma(\hat{\alpha})} = {}_{n_*}C_{\hat{\alpha}-1}\left(\frac{\hat{\beta}}{\hat{\beta}+1}\right)^{\hat{\alpha}}\left(\frac{1}{\hat{\beta}+1}\right)^{n_*} = {}_{n_*}C_{r-1}(1-p)^r p^{n_*}$$

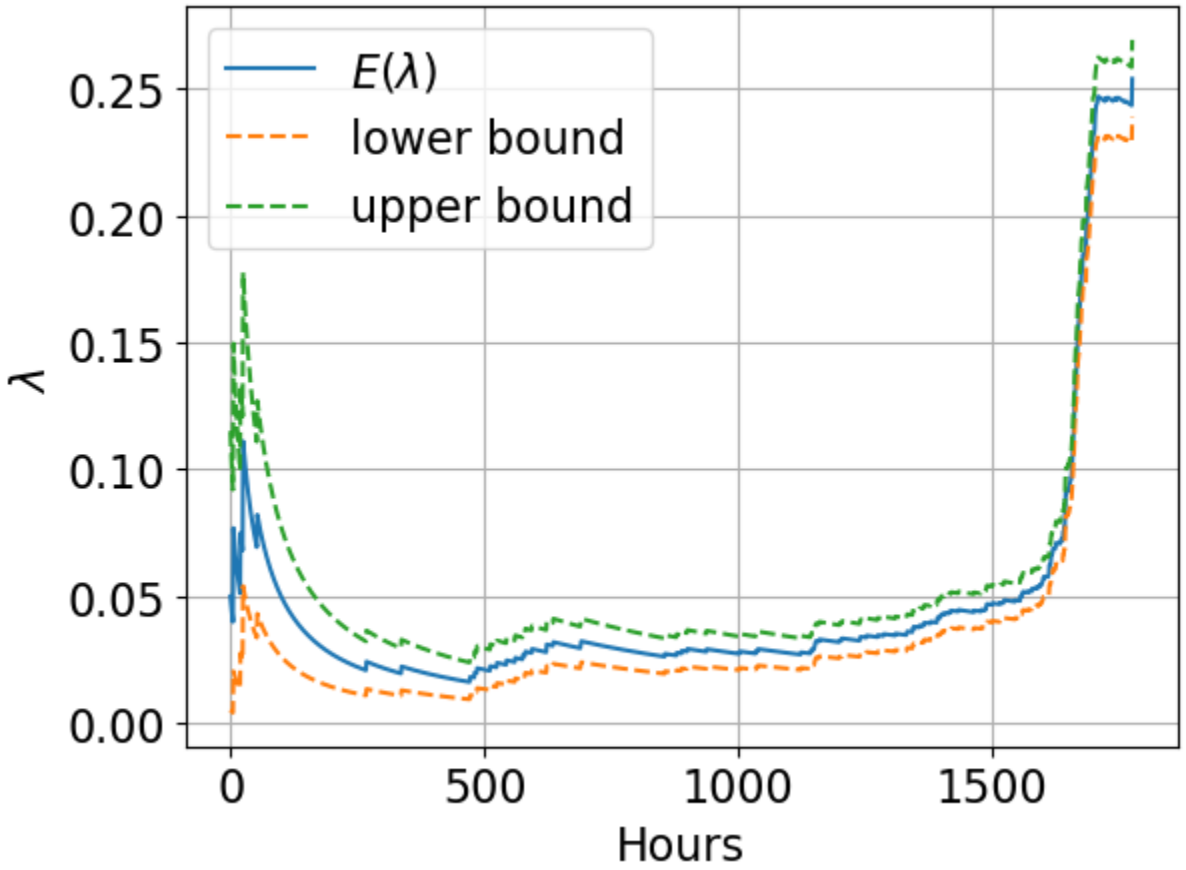
- ここで、$r = \hat{\alpha},\ p = 1/\left(\hat{\beta}+1\right)$を定義した。
- この分布は負の二項分布 (Negative binomial distribution) と呼ばれる。



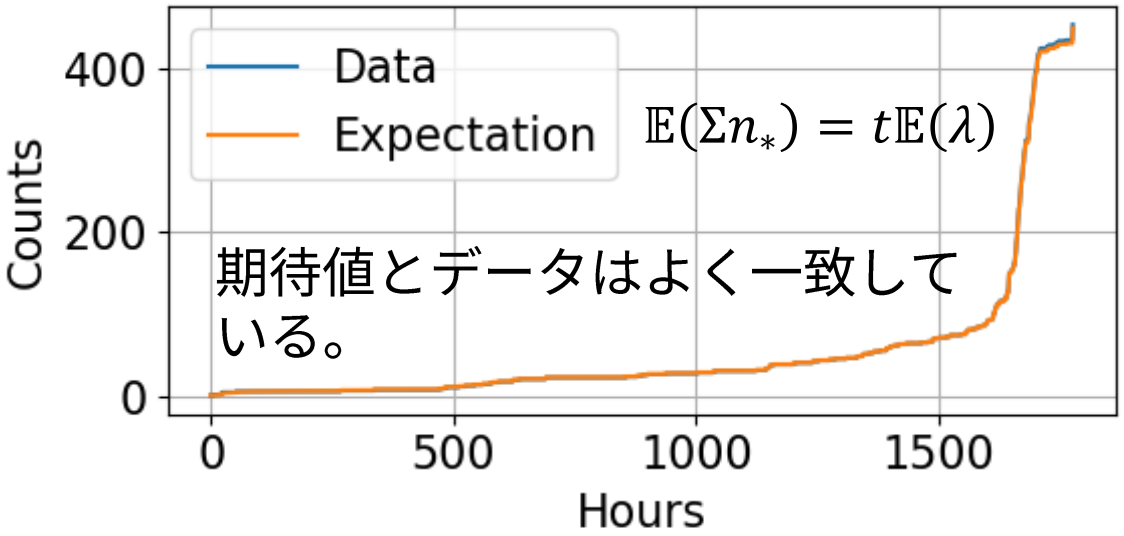



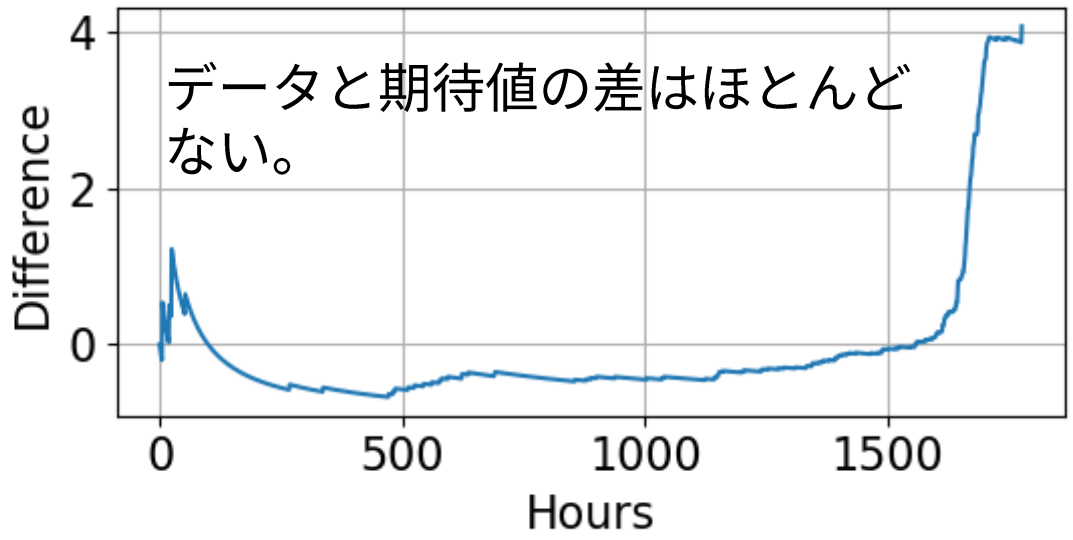
# パラメータ λ の確率分布のトレンドと累積自己放電回数

- 事後分布が適切に計算され、累積自己放電回数と矛盾しないことがわかる。




累積自己放電回数



# 累積自己放電回数の分布の推定

- 自己放電回数の確率変数 $n^*$ の和の確率分布をまず求める。
- これはモーメント母関数 (Moment-generating function, MGF) を使って算出することができる。

$$\mathrm{MGF}_X(t) = \mathbb{E}(e^{tX}) = \int e^{tX} p(X) dX\,, \qquad t \in \mathbb{R}$$

  - ここに、$X$ はある確率変数を示す。
- モーメント母関数は確率変数のラプラス変換のようなものである。
- 確率変数の和の分布は各項の畳み込み計算で求めることになるが、ラプラス変換すると畳み込みが積に変換されて容易に計算できる。

$$\mathrm{MGF}_{X+Y}(t) = \mathrm{MGF}_X(t)\,\mathrm{MGF}_Y(t)$$

- 負の二項分布のモーメント母関数は以下のとおり。

$$\mathrm{MGF}_{n_*}(t) = \left(\frac{1-p}{1-pe^t}\right)^r = \left(\frac{\hat{\beta}}{\hat{\beta}+1-e^t}\right)^{\hat{\alpha}}$$

- したがって、負の二項分布に従う確率変数の和のモーメント母関数は以下のようになる。

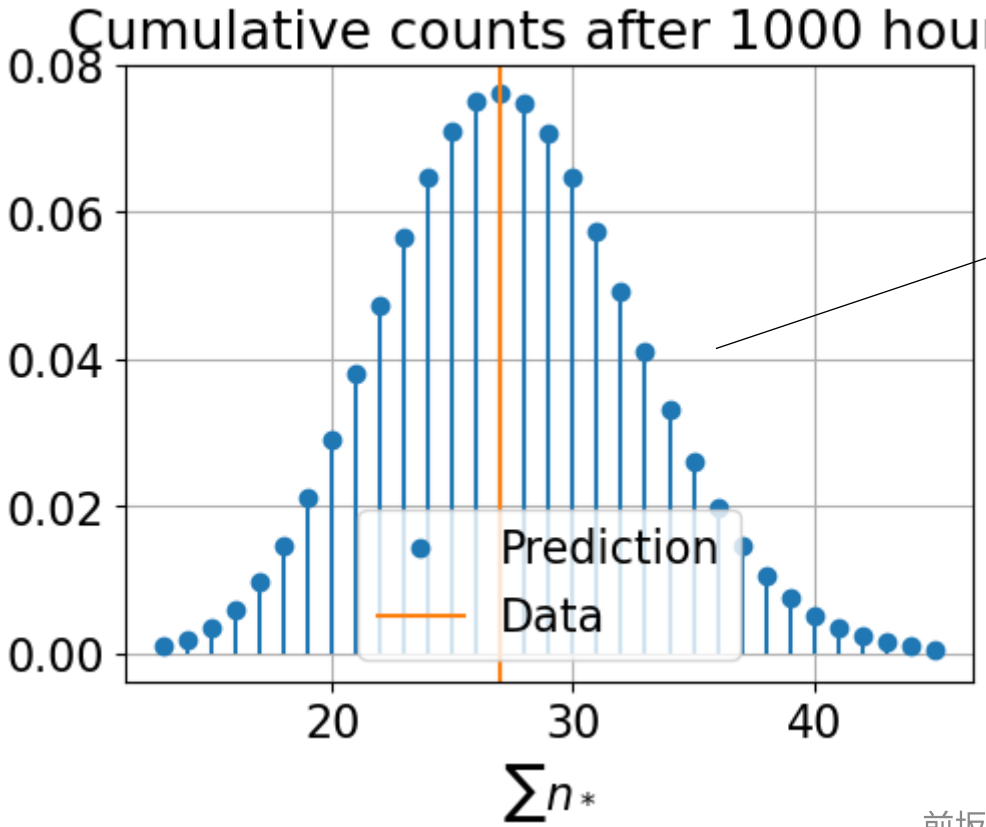
$$\mathrm{MGF}_{Tn_*}(t) = \left(\frac{1-p}{1-pe^t}\right)^{Tr} = \left(\frac{\hat{\beta}}{\hat{\beta}+1-e^t}\right)^{T\hat{\alpha}}$$

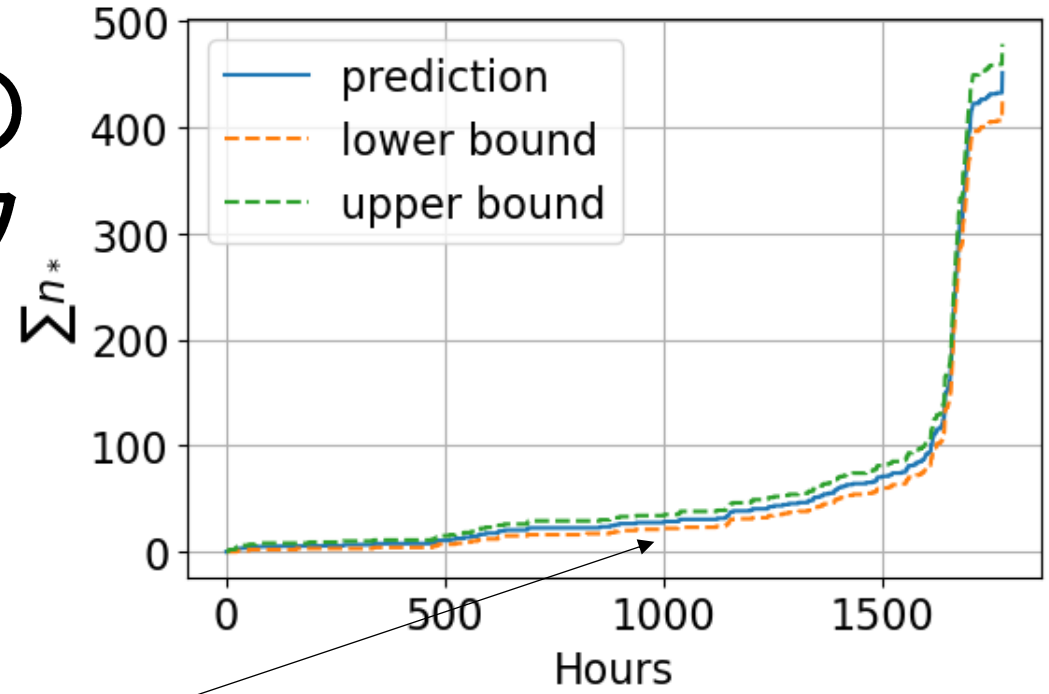
- このように、累積自己放電回数の分布も以下のような負の二項分布となる。

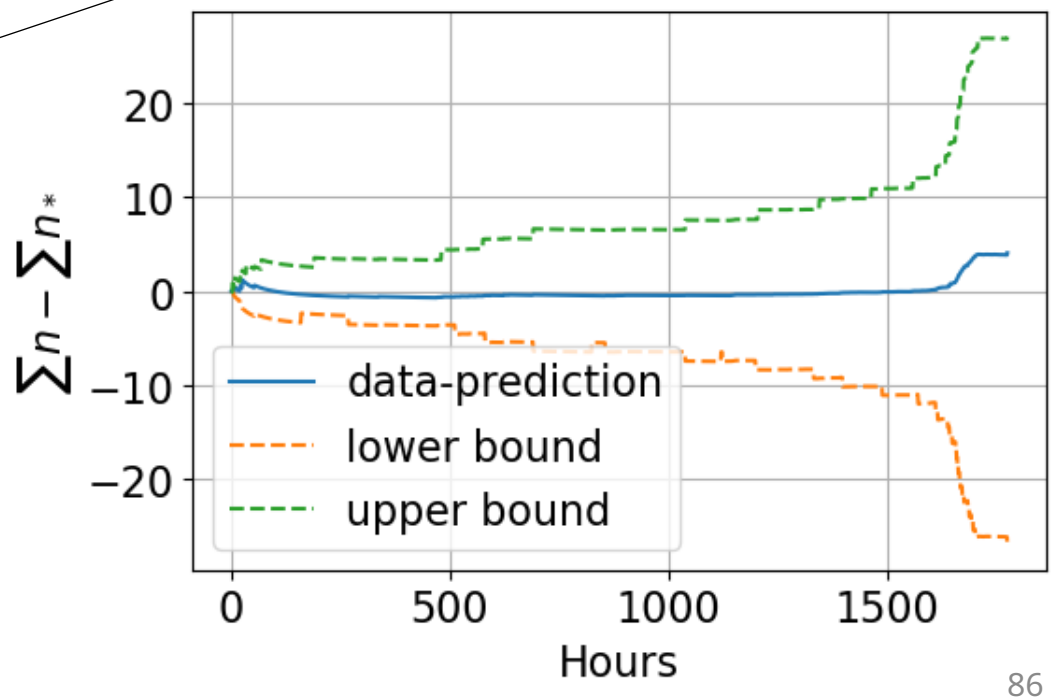
$$p(\Sigma\, n_*) = p(Tn_*) = {}_{Tn_*}C_{T\hat{\alpha}-1} \left(\frac{\hat{\beta}}{\hat{\beta}+1}\right)^{T\hat{\alpha}} \left(\frac{1}{\hat{\beta}+1}\right)^{Tn_*}$$

# 推定された自己放電回数の分布とそのトレンドグラフ

- 自己放電回数の分布が正しく計算されている。



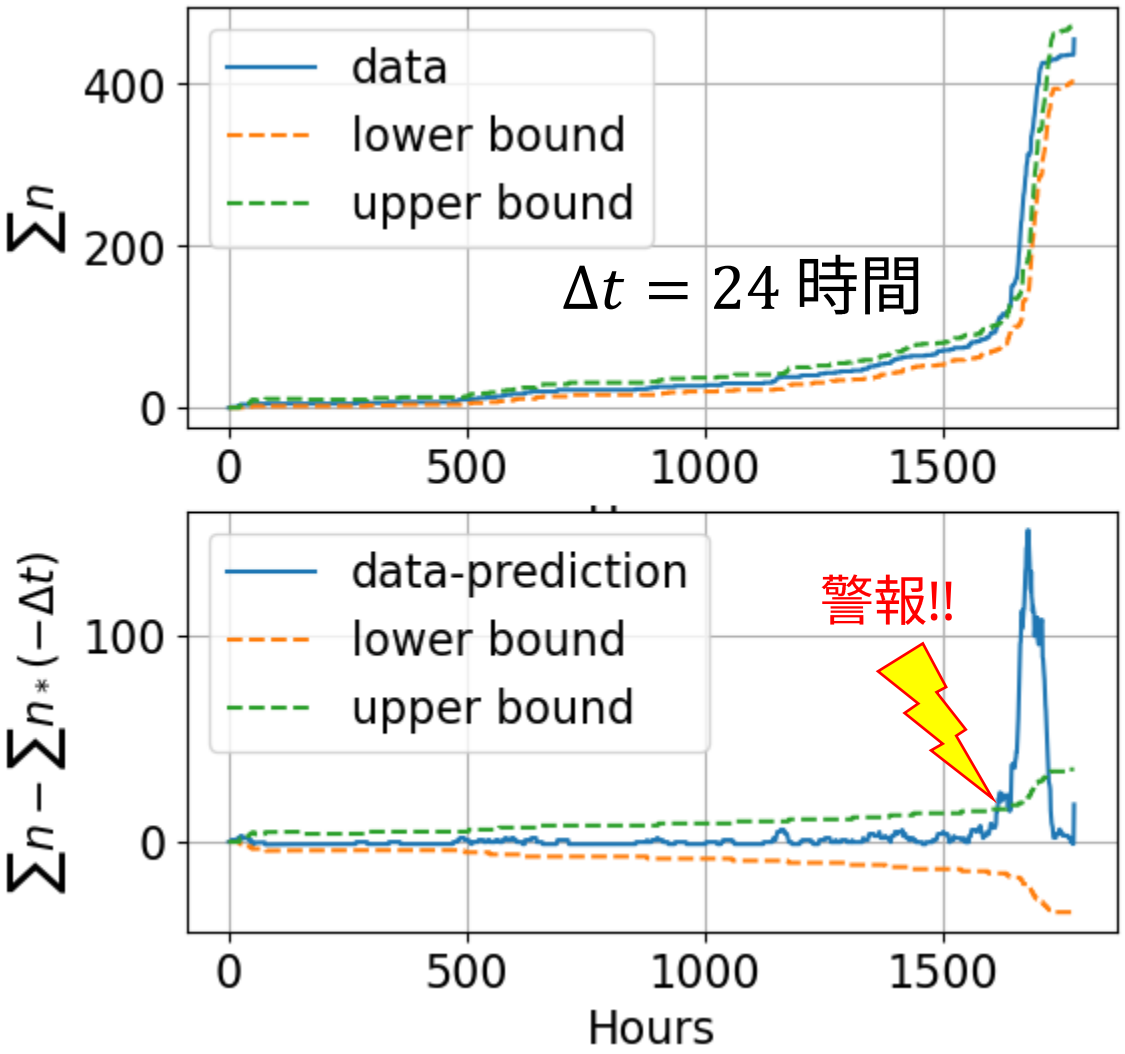
# 累積自己放電回数の異常検知

- 異常検知をするためには、一定時間前の正常時の推定結果を時間発展させて将来予測し、現時点でのデータと比較する必要がある。
- 一定の時間 $\Delta t$ 前の推定結果 $\hat{\alpha}(-\Delta t), \hat{\beta}(-\Delta t)$ をもとに $\Delta t$ だけ時間発展させた分布を求めると以下のようになる。

$$p\left(\Sigma\, n_*(-\Delta t)\right) = p\left(Tn^*(-\Delta t)\right)$$

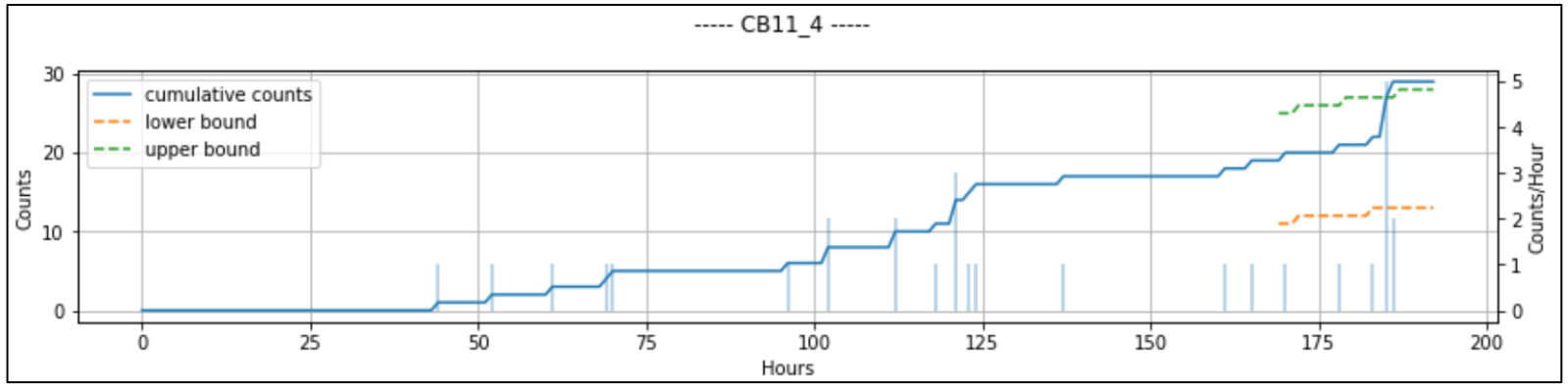
$$= {}_{Tn_*(-\Delta t)}C_{T\hat{\alpha}(-\Delta t)-1} \left(\frac{\hat{\beta}(-\Delta t)}{\hat{\beta}(-\Delta t)+1}\right)^{T\hat{\alpha}(-\Delta t)} \left(\frac{1}{\hat{\beta}(-\Delta t)+1}\right)^{Tn_*(-\Delta t)}$$

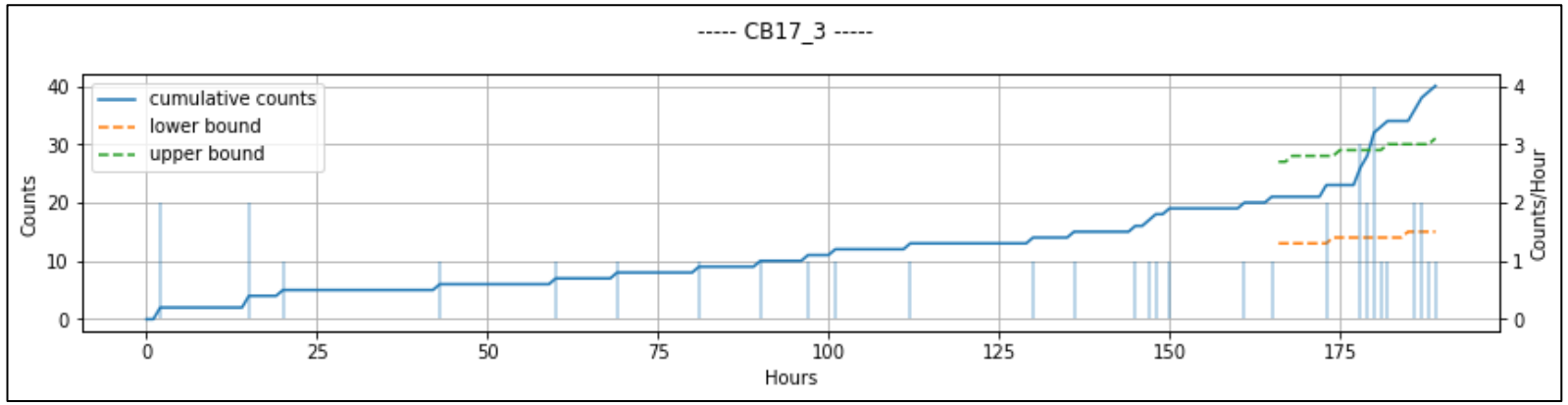
- このようにして、右図のように自己放電回数の急激な増加を検知することができた。

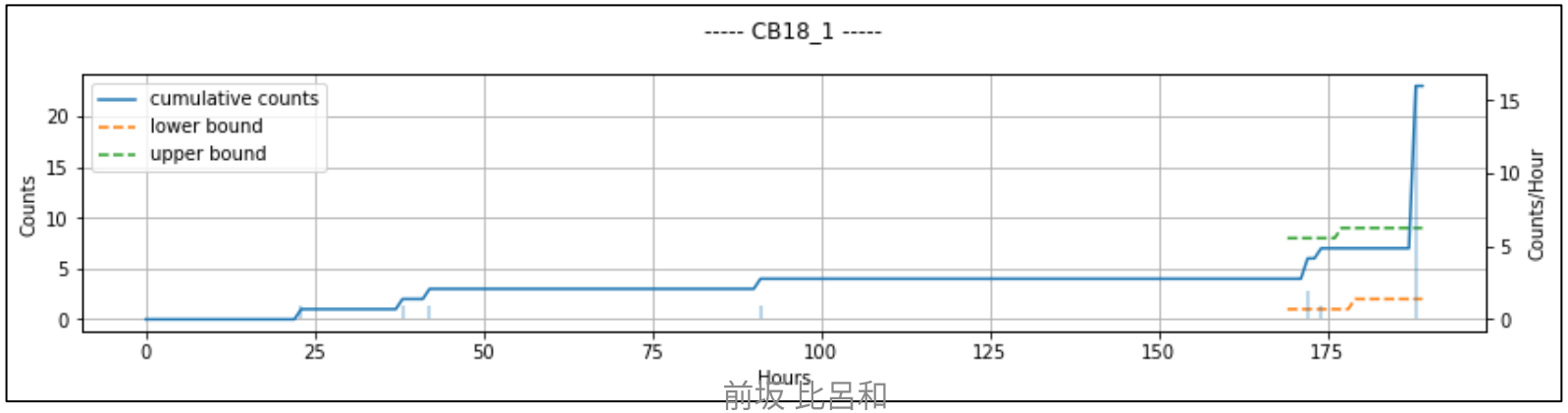


# SACLA で使用中のウェブアプリのグラフ

このアルゴリズムで自己放電回数の増加が検知された例。



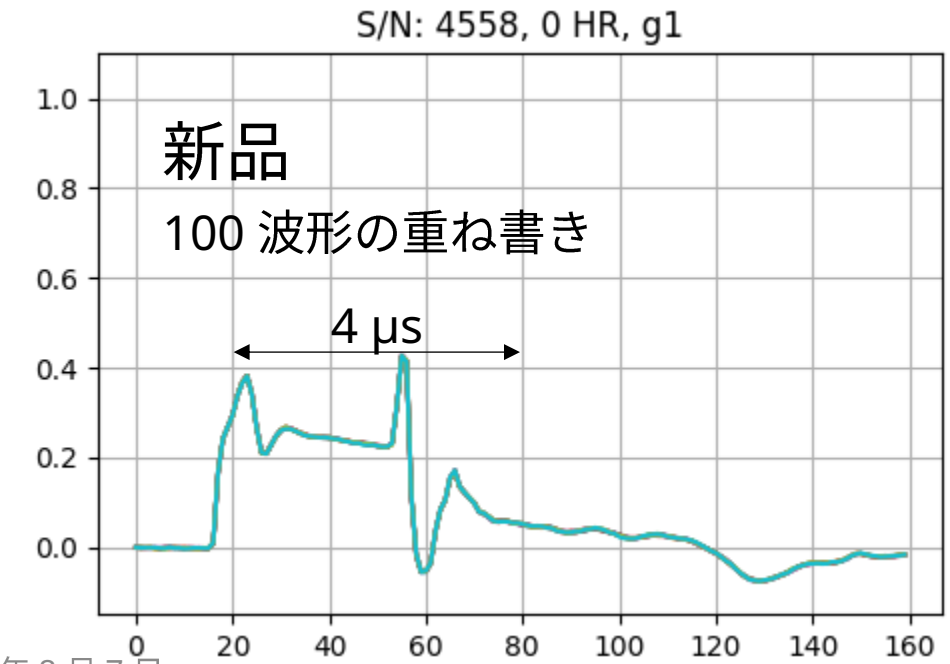



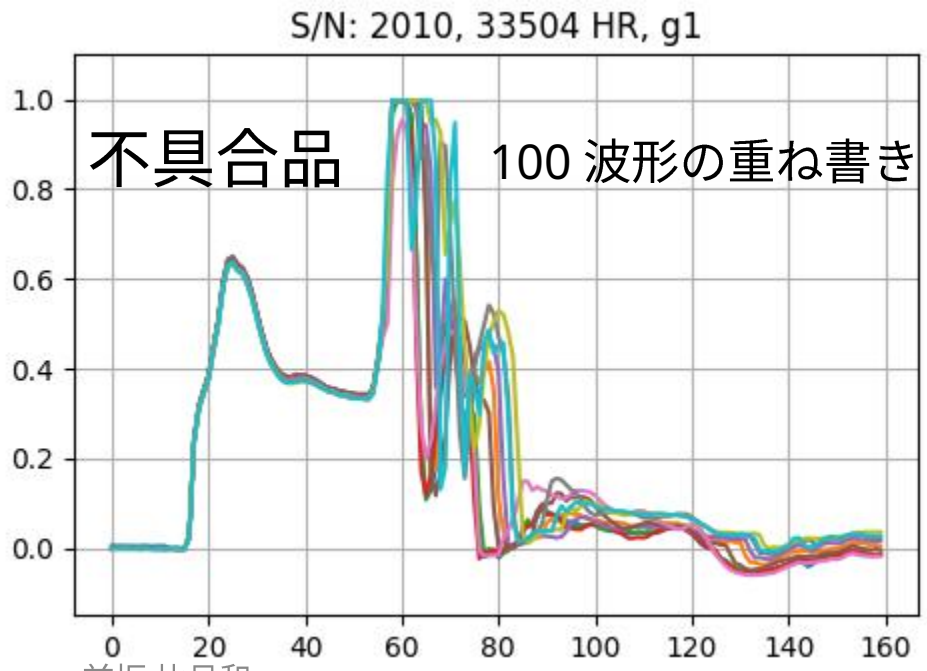



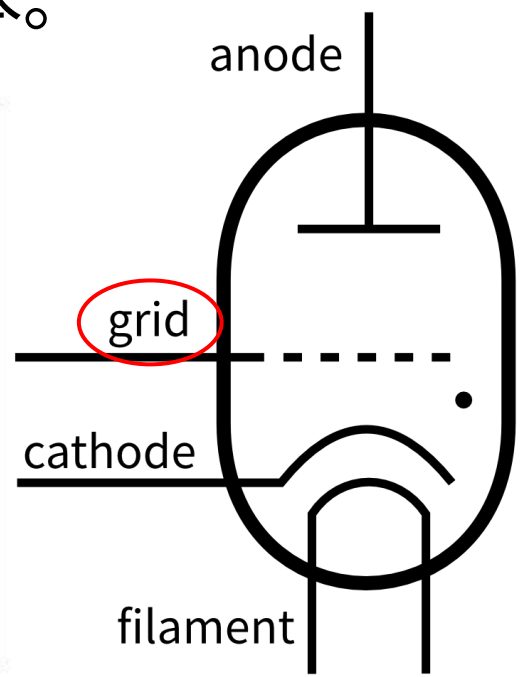

中央電子(株)との
共同制作

# サイラトロンのグリッド波形の監視

- サイラトロンのグリッド波形には継時変化があることが知られている。
- 古くなるとスパイク上のサージ電圧が大きくなる。
- サージ電圧が大きくなるとトリガ回路に悪影響を及ぼすなどの問題があるため、大きくなりすぎると寿命と判断される。
- このグリッド波形を使って異常検知ができるのではないかと考えた。
- データセット: 新品、中古品、不具合品の３種類、全 46 本。



# 主成分分析 (PCA) による次元削減

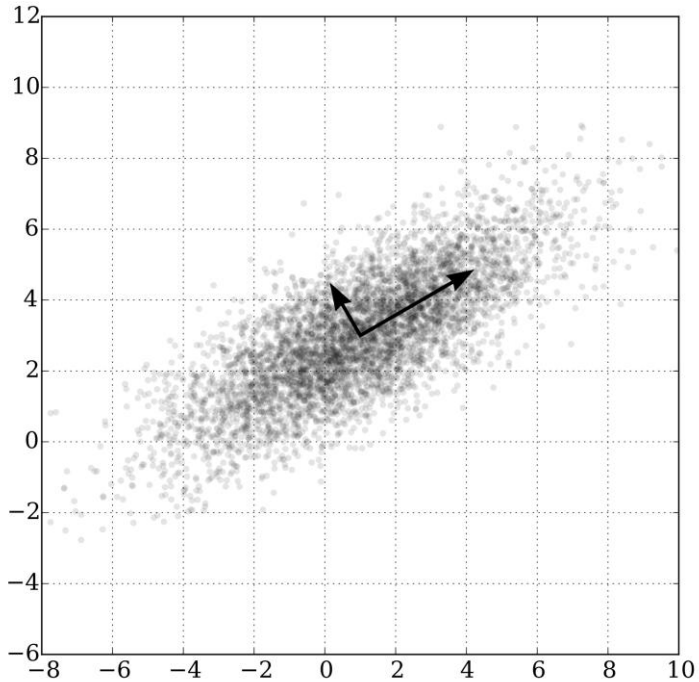
- 波形データは量が多く扱いづらいので次元削減して特徴抽出するのが良い。
- ここでは主成分分析 (Principal Component Analysis, PCA) による次元削減と特徴抽出を行う。
- 主成分分析では行列の特異値分解 (Singular Value Decomposition, SVD) を用いる。

$$M = U\Sigma V^{\mathrm{T}}$$

  - $M$: $m \times n$ 行列
  - $U$: $m \times m$ ユニタリ行列
  - $V$: $n \times n$ ユニタリ行列
  - $\Sigma$: $m \times n$ 非負要素からなる長方対角行列
- 波形データの主成分分析では以下のように行列を構築する。

$$M = \begin{pmatrix} x_1(t_1) & \cdots & x_1(t_n) \\ \vdots & \ddots & \vdots \\ x_m(t_1) & \cdots & x_m(t_n) \end{pmatrix}$$

  - $n$: 各波形のサンプル数。
  - $m$: 波形データ数。
- 行列 $\Sigma$ の対角成分は特異値 (Singular value) と呼ばれる。
- 特異値が大きいほど、その成分の寄与が大きいことを意味する。
- 大きな特異値に対応する $V$ の列ベクトルが特徴的な基底波形となる。
- いくつかの大きな特異値に対する基底波形でデータを再現することで次元削減と特徴抽出ができる。

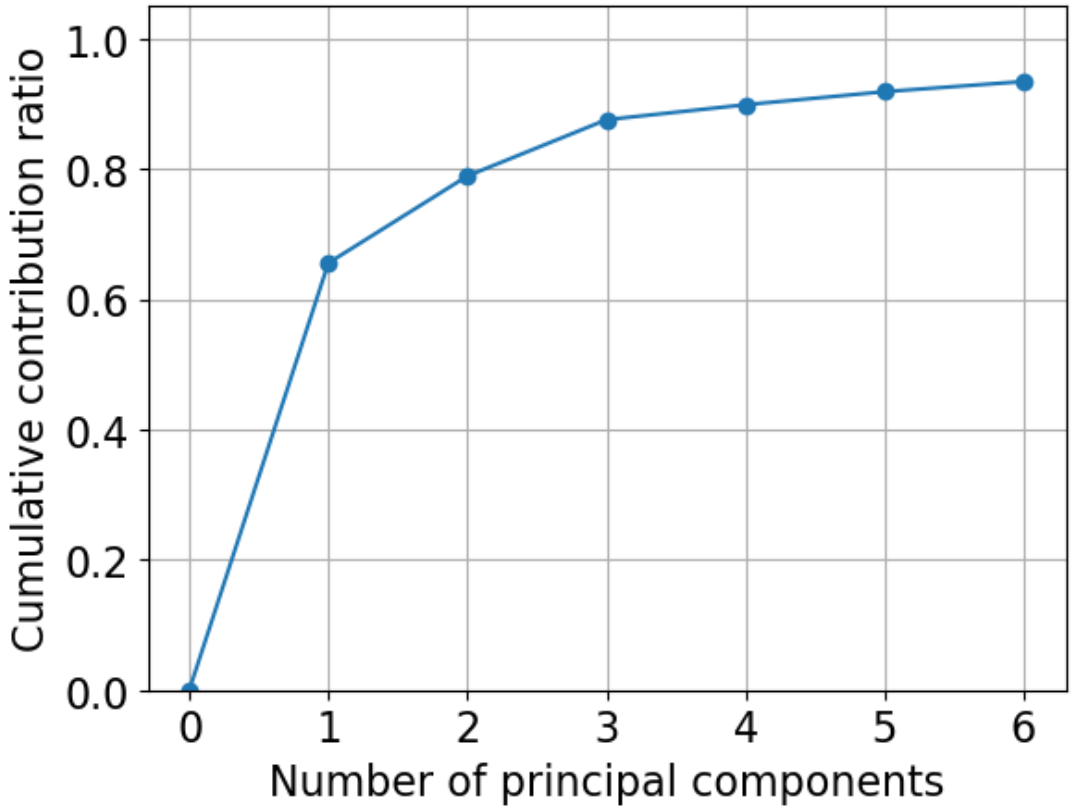


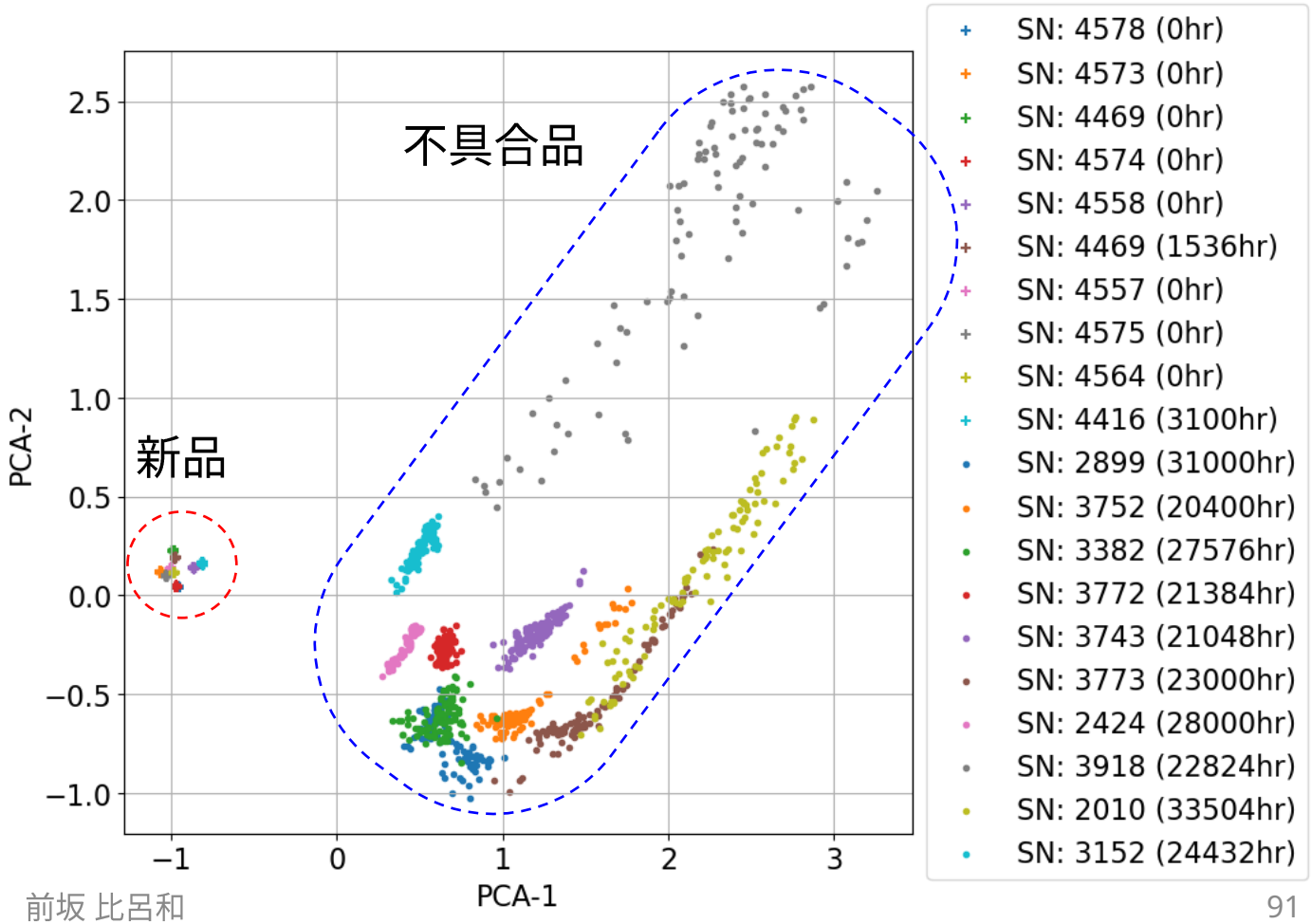

Decomposition of two-dimensional normal distribution

Author: Nicoguaro, CC BY 4.0, https://en.wikipedia.org/wiki/Principal_component_analysis

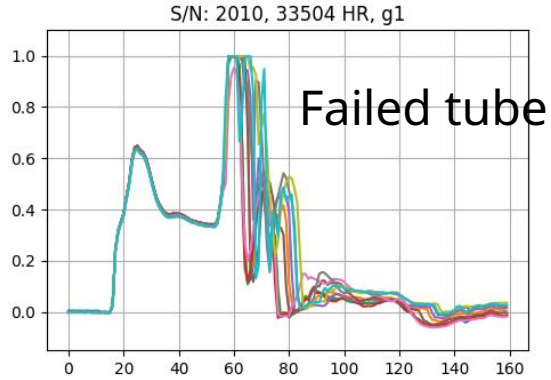
# 主成分分析結果

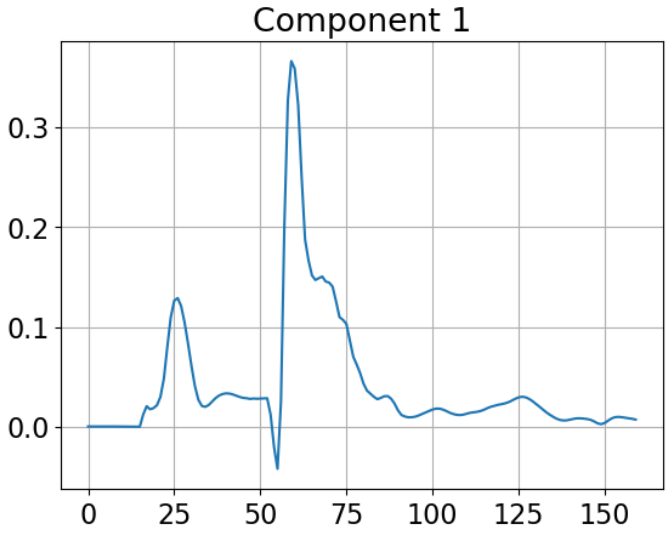
- 累積寄与率は 6 個の特徴量だけで 0.9 を超えた。
- 新品と不具合品が 2 個の特徴量だけではっきり分かれた。

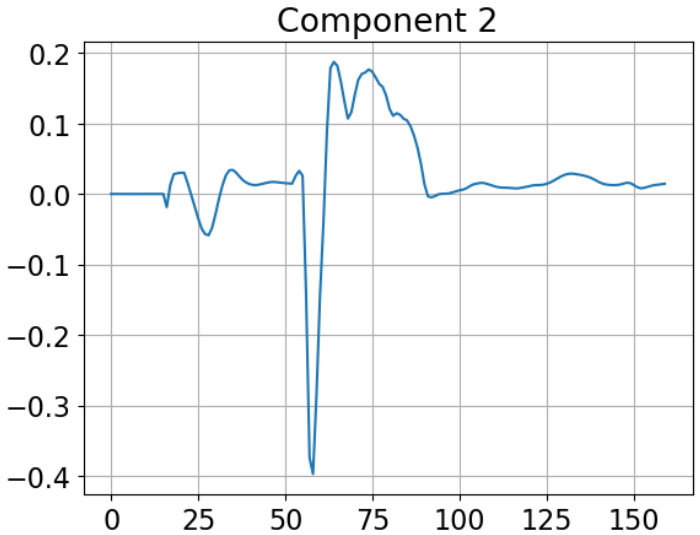


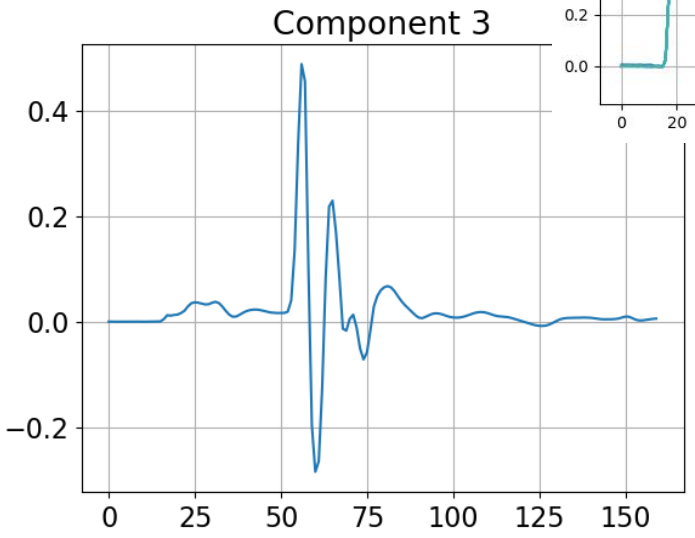



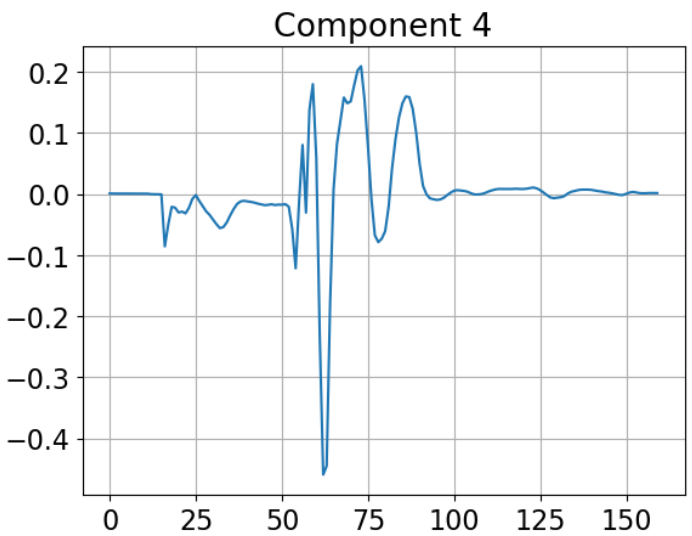
# 特異値の大きい基底波形

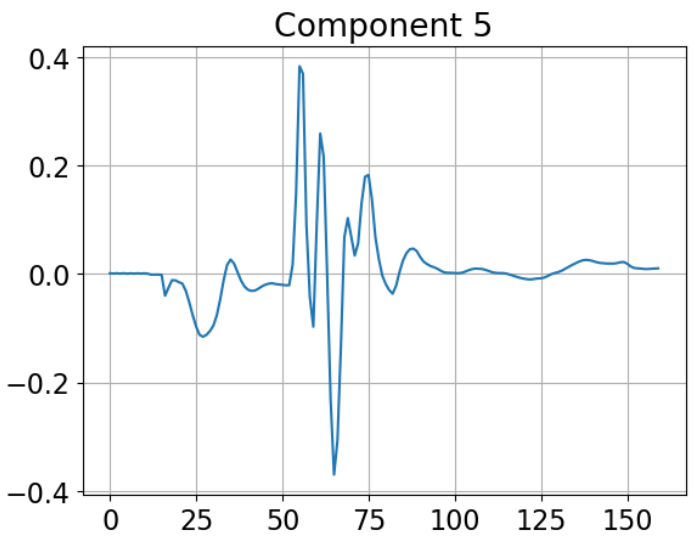


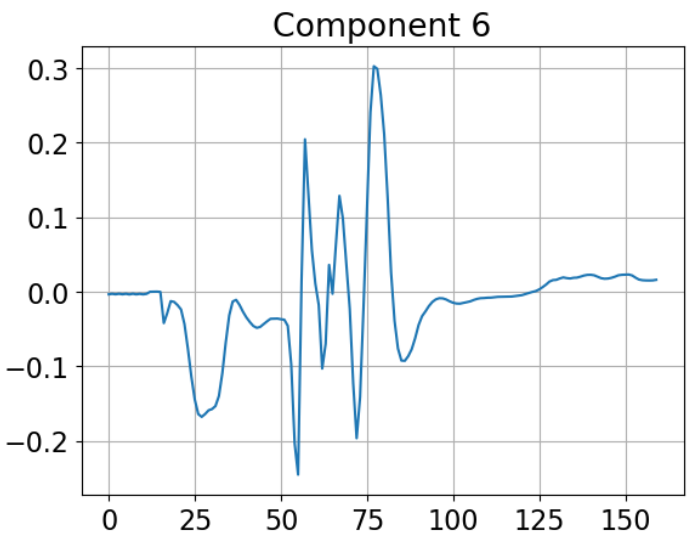



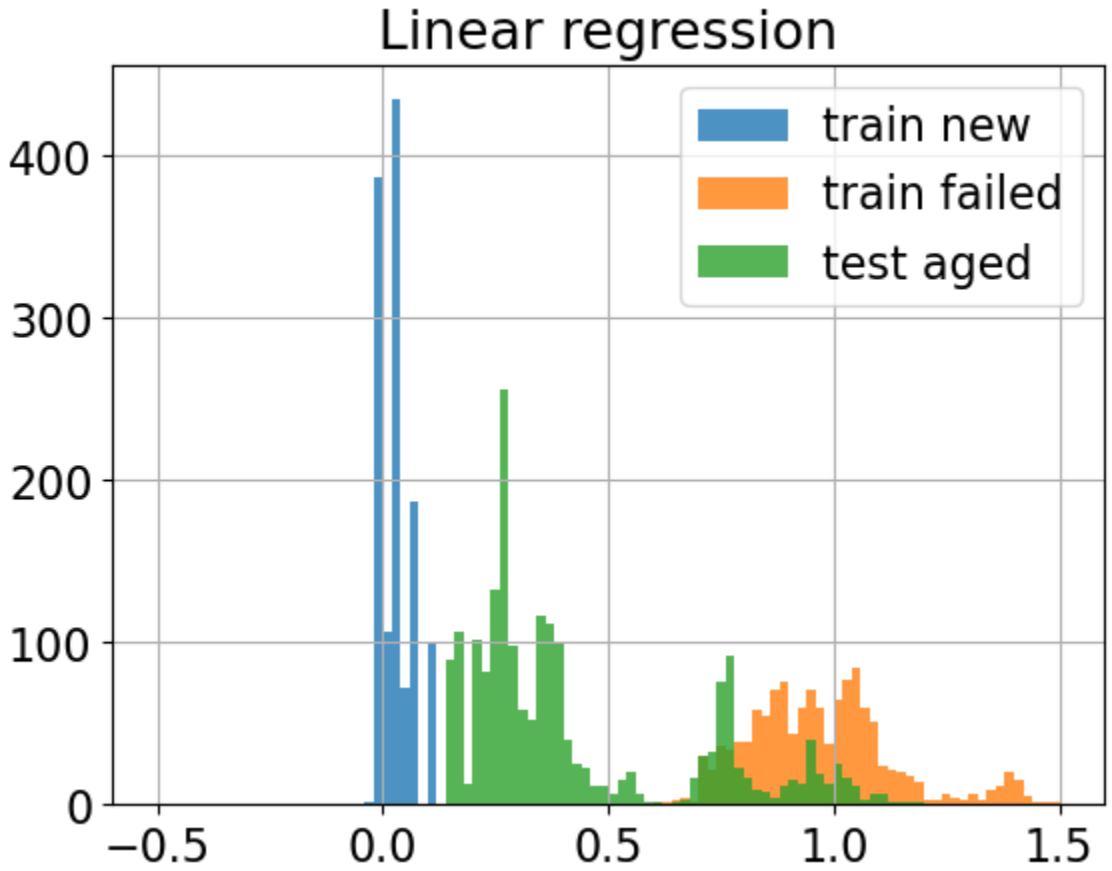



# 線形回帰 (Linear regression) による劣化度推定

- 主成分分析結果を線形回帰で解析。

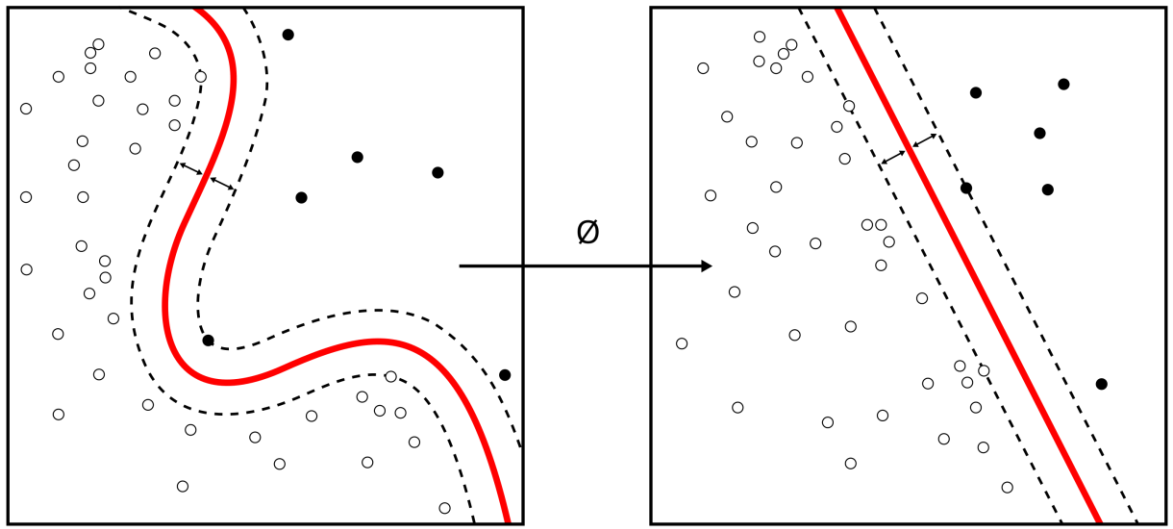
$$y = \sum_{i=1}^{N} \boldsymbol{w}_i^{\top} \boldsymbol{x}_i$$

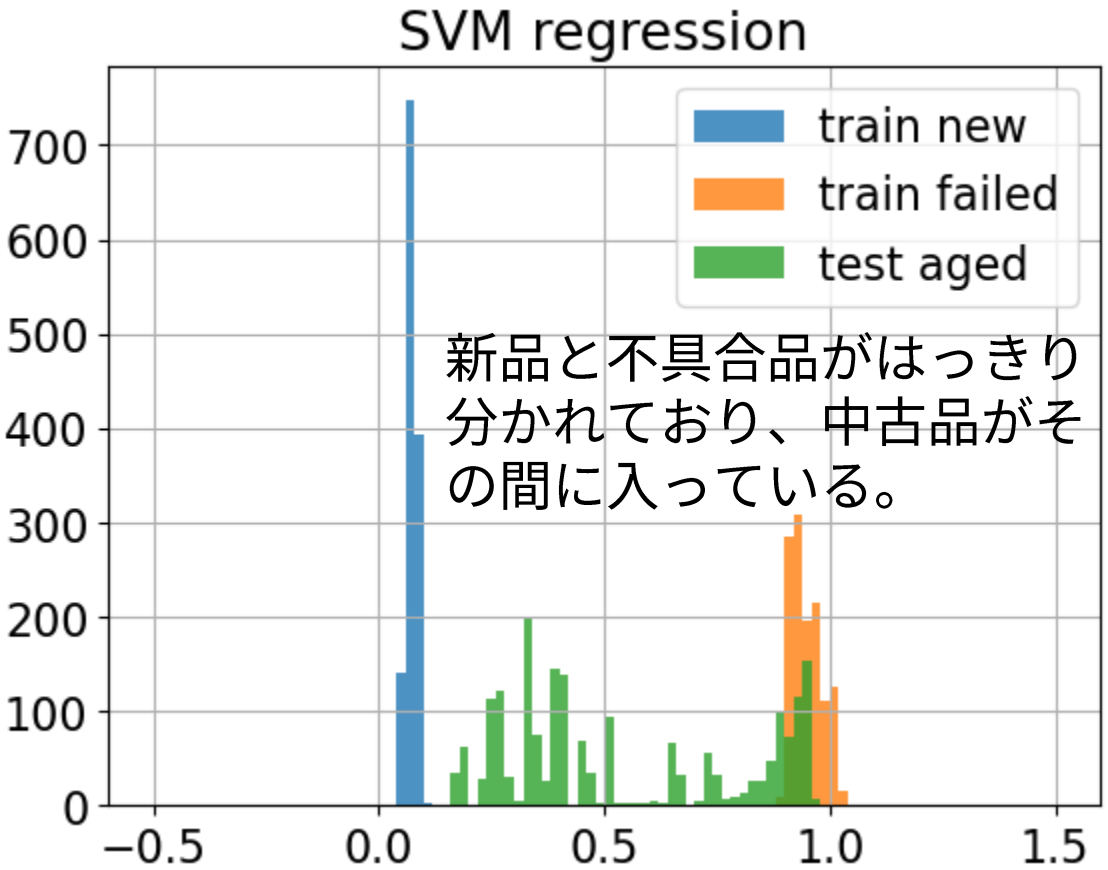
  - 今回は $N = 6$ 。
- 学習データには新品と不具合品だけを使用した。
  - 新品: $y = 0$
  - 不具合品: $y = 1$
- 中古品を評価用のテストデータとして使用した。
- 新品と不具合品ははっきりと分かれた。
- 中古品がその間に入っている。



# サポートベクターマシンによる劣化度推定

- サポートベクターマシン (Support Vector Machine, SVM) は便利な教師あり機械学習アルゴリズムのひとつ。
- ガウス過程と類似のカーネル関数を使って SVM モデルをうまくデータとその教師ラベルにフィットさせることができる。
- SVM は教師ラベル値の異なるデータをより遠くに引き離すような関数にうまくフィットさせる。




新品と不具合品がはっきり分かれており、中古品がその間に入っている。

線形回帰より鋭い分布になっており、精度が高そうに思われる。

By Original: Alisneaky  Vector: Zirguezi - Own work based on: Kernel Machine.png, CC BY-SA 4.0, https://en.wikipedia.org/wiki/Support_vector_machine

# Eu-XFEL の超伝導高周波空洞のクエンチ検出

A. Eichler et al., Phys. Rev. Accel. Beams **26**, 012801 (2023)

- 超伝導空洞の応答はほぼモデルどおりにふるまうため、モデルベースのクエンチ検出を行っている。
  - 通常は高周波パルスの立ち下がり時の空洞内のパワーの減少率から負荷 Q 値を算出し、その Q 値が空洞ごとに設定した正常範囲を逸脱したらクエンチとみなす、とのこと。
- しかし、現行のクエンチ検出方法だと、意図的に空洞を少しデチューンした場合や、運転を止めるほどではない電解放出による波形変動で、望まれないクエンチ検出がされて運転が止まってよくないらしい。
- 入力 RF、反射 RF、空洞内 RF の波形の関係をモデルでより丁寧に計算し、計算からの残差が大きい際にクエンチとみなすようにした。
- これで望まれないクエンチ検出が減らせるとのこと。
  - たぶん実機への実装はまだ。

残差の評価には Generalized Likelihood Ratio (GLR) を使用。

$$\lambda_{\mathrm{GLR}}(k) = \frac{K}{2}\left(\frac{1}{K}\sum_{i=k-K+1}^{k} r(i)^{\top}\right)\Sigma^{-1}\left(\frac{1}{K}\sum_{i=k-K+1}^{k} r(i)\right)$$

GLR: 残差の重み付き 2 乗和の移動平均のようなもの。

この続編が MaLAPA’25 でも報告あり。

L. Boukela et al., “Enhancing Quench Detection in SRF Cavities at the European XFEL”, MaLAPA’25.

# まとめ

- 近年の加速器の高性能化・複雑化に伴い、加速器の運転にさまざまな課題が生じてきた。
- 原則として加速器は物理モデルに準拠した運転・調整を行うべきであるが、模型で表現しきれない部分はデータ駆動型、即ち、機械学習的な手法で対処する必要が出てきている。
  - 安易に機械学習に頼るのは個人的には賛成しない。
- 加速器性能の最適化として、ベイズ最適化 (ガウス過程) や強化学習が盛んに研究・応用されている。
- 高効率・高次元ビーム診断も高度な加速器調整に必要とされている。
- 機械学習を応用した異常検知・故障予知で運転を極力止めずに計画的にメンテナンスするすることで安定運転への貢献が期待される。

# 最後に宣伝

- 加速器機械学習フォーラム（日本）
  - お気軽にご参加ください。
    https://www.rcnp.osaka-u.ac.jp/Divisions/acc/accml/
- 次回ワークショップは J-PARC 主催。
  - 2025 年 12 月 8 日、9 日、JAEA Tokai Mirai Base にて。

- MaLAPA Workshops
  - Machine Learning Applications for Particle Accelerators
    https://malapa.org/workshops
  - Discord invitation (Chat app)
    https://discord.gg/7kJTCNqT9t
  - GitHub repository
    https://github.com/MALAPA-Collab
- 次回ワークショップは SPring-8/SACLA 主催！
  - 2026 年 4 月 21 日 ～ 24 日、アクリエひめじ にて。