Parameters estimation of the Viscous Blast-Wave model using Machine learning techniques

Nachiketa Sarkar (NISER)

ICPQGP-2023, PURI

Why This Approach?

We want to determine a set of model parameters that characterize the system formed in the ultra-relativistic heavy ions by comparing model outputs with the experimental results.

But, It's Not So Simple

- From collision to particle dedication different stages are described by different physics/models.
- Need huge CPU time, sometimes beyond our affordable range.

  - *Calculation the centrality dependence of bulk observables requires $O(10^4)$ minimum-bias events @ $O(10^{-1})$/ hours.*
  - *$O(10^3)$ hours per parameter sample.*
  - *Statistically significant sample size > $O(10^6)$.*
  - *Total computation time >$O(10^9)$ hours ~$O(10^5)$ years.*

- Experimental measurements always contain some noise.
- A model often has multiple inputs-outputs and usually carries complex interrelationships.

We Want To Achieve

- **Minimisation of the CPU time.**
- **Taking care of the correlations between parameters.**
- **Taking care of the different uncertainties in an efficient way.**
- **Finally, want to build a data-driven technic that is model-independent.**

Viscous Blast-Wave Model

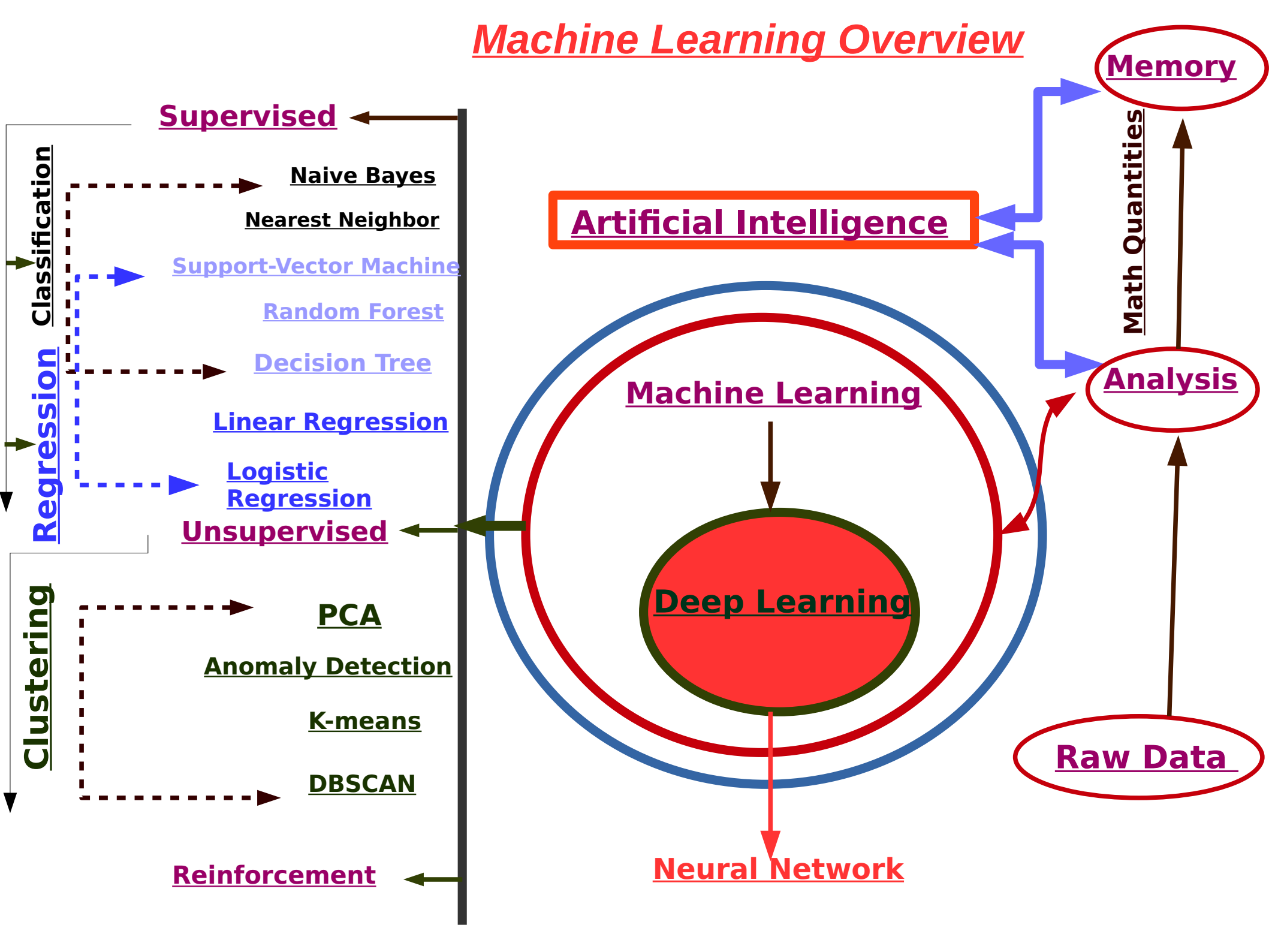
A Generalized blast-wave model that includes viscous effects by employing a viscosity-based survival scale for geometrical anisotropies formed in the early stages of relativistic heavy-ion collisions was developed by Amaresh Jaiswal and Volker Koch.

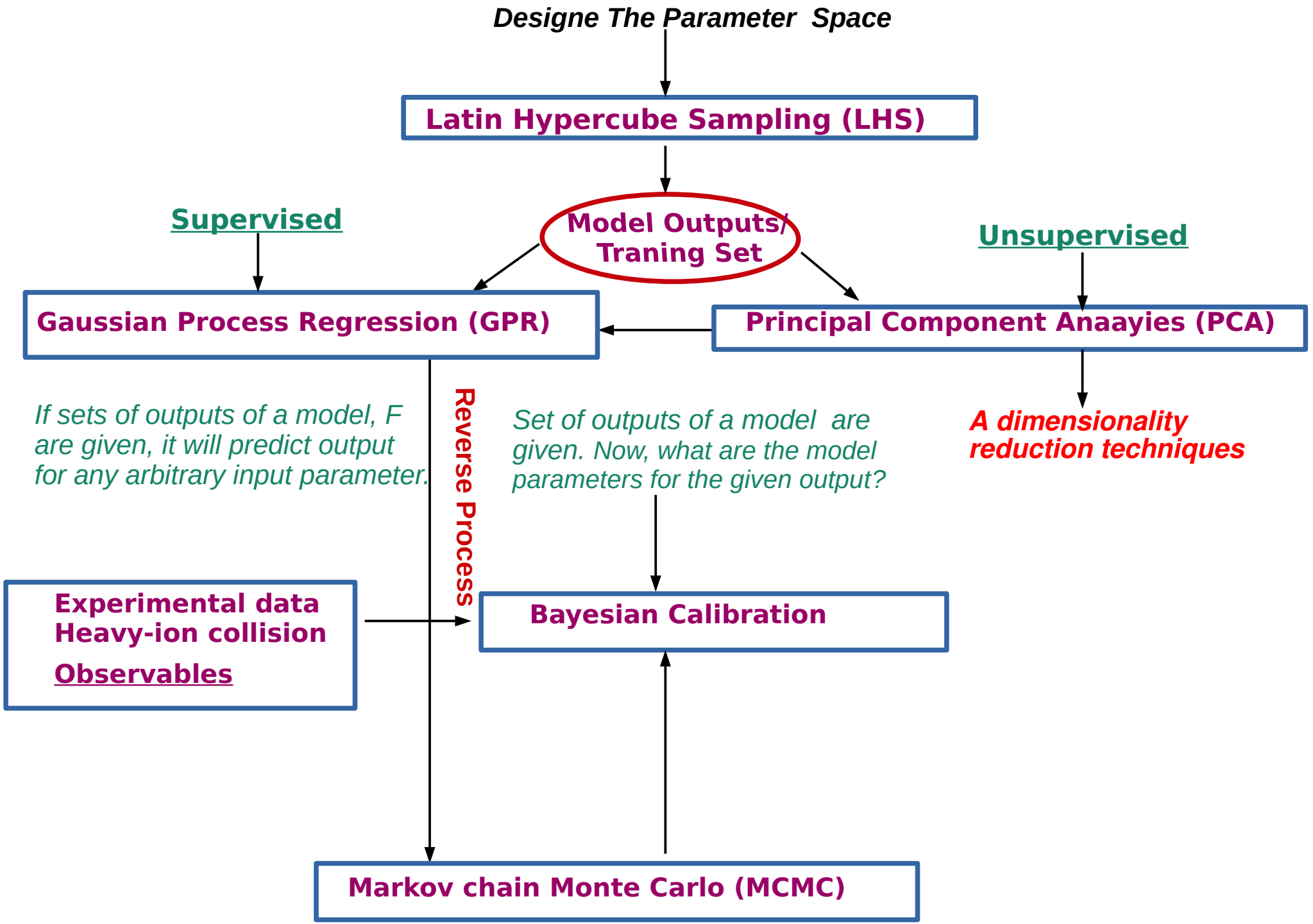
$$\frac{d^2N}{d^2p_T\,dy} = \frac{1}{(2\pi)^3}\int_0^R r\,dr\int_0^{2\pi} d\varphi\int_{-\infty}^{\infty}\tau\,d\eta_s\,m_T\cosh(y-\eta_s)f.$$

$$v_n(p_T) \equiv \frac{\int_{-\pi}^{\pi} d\varphi_p\,\cos[n(\varphi_p-\Psi_n)]\,\frac{dN}{dy\,p_T\,dp_T\,d\varphi_p}}{\int_{-\pi}^{\pi} d\varphi_p\,\frac{dN}{dy\,p_T\,dp_T\,d\varphi_p}},$$

The model has six parameters

- *Freeze-out temperature,* T_f
- *Radial flow velocity,* β_0
- η/s
- *Three other constants : m,* α_0 *and* κ


Machine Learning Overview
Supervised
Classification
Naive Bayes
Nearest Neighbor
Support-Vector Machine
Random Forest
Decision Tree
Regression
Linear Regression
Logistic Regression
Unsupervised
Clustering
PCA
Anomaly Detection
K-means
DBSCAN
Reinforcement
Artificial Intelligence
Machine Learning
Deep Learning
Neural Network
Memory
Math Quantities
Analysis
Raw Data


Designe The Parameter Space
Latin Hypercube Sampling (LHS)
Supervised
Model Outputs/
Traning Set
Unsupervised
Gaussian Process Regression (GPR)
Principal Component Anaayies (PCA)
If sets of outputs of a model, F are given, it will predict output for any arbitrary input parameter.
Reverse Process
Set of outputs of a model are given. Now, what are the model parameters for the given output?
A dimensionality reduction techniques
Experimental data
Heavy-ion collision
Observables
Bayesian Calibration
Markov chain Monte Carlo (MCMC)

Construction of Design Points / Latin Hypercube Sampling (LHS)

Let, the model is to be evaluated on a set of m training points $X(x_1.....X_m)$, in n-dimensional parameter space.

Now, how to generate efficient design points? , i.e., simultaneously optimize emulator accuracy and computation time.

Possible Strategies.

***i*) Factorial Design: k points in each of n dimensions; k^n total points.**

For k = 20, n = 5 No. Design Points = 3200000; ***Fails in high dimensions.***

***ii*) Random Points : A Monte Carlo approach, is good when the design points are large.**

With low design points, often leave large regions with no points.

iii) Latin Hypercube Sampling: The most commonly used sampling algorithm.

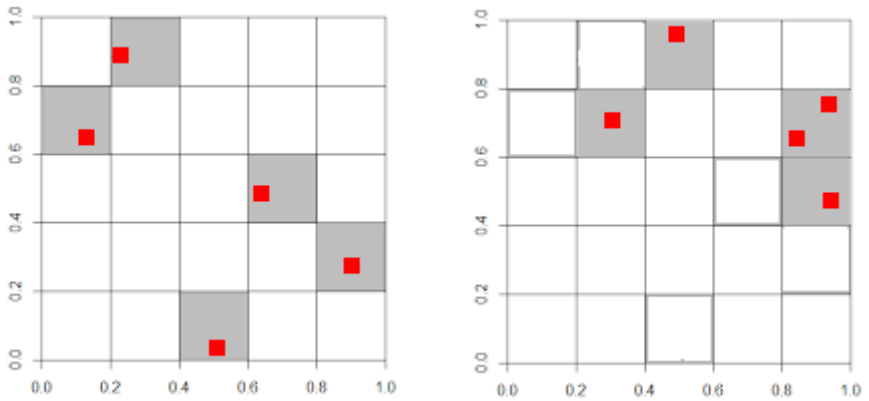
A semi-random, space-filling design algorithm.

Ref: Latin Hypercube Sampling By Jaroslav

Latin Hypercube Sampling (LHS)

Two Dimension

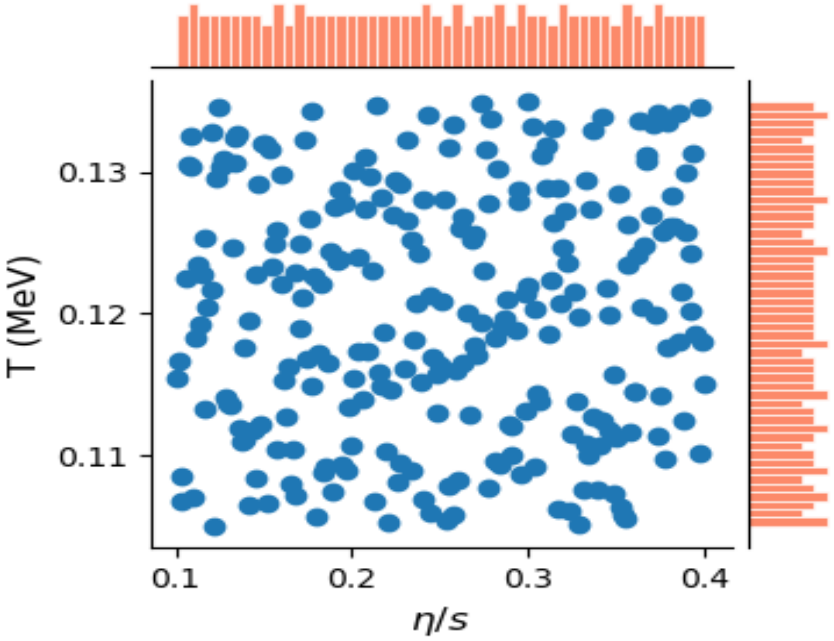
In a statical sampling, a square grid containing sample position is a Latin Square if and only if there is only one sample in each row and column.



Monte Carlo Vs Latin Hypercube

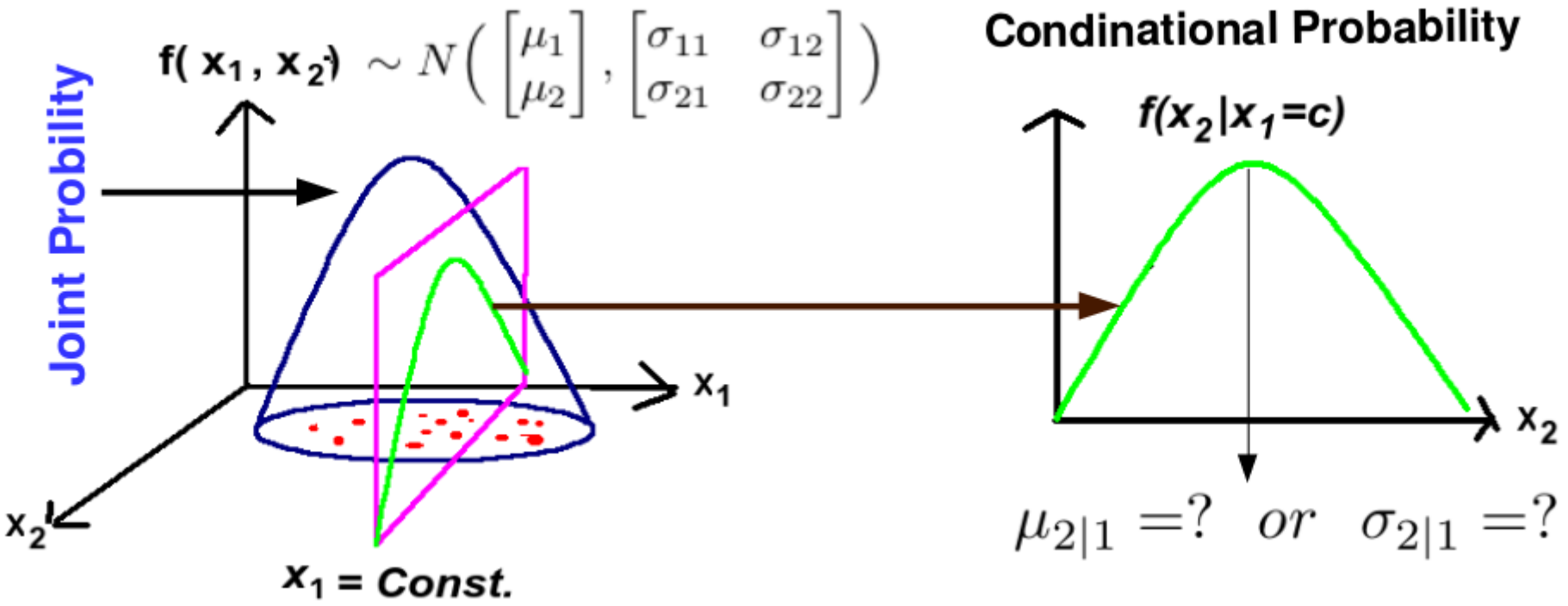
MCS: New sample points are generated without taking into account previously generated sample point.

LHS: Remember which rows and columns have already been occupied by a sample point.



We use 250 Latin hypercube design point across the n = 6 dimensional parameters space.

An Important Property Of MVN



An important property of MVN

If a vector $X(x_1.....X_n)$ has a multivariate normal distribution then any subset of X also has a multivariate normal distribution.

$$\mu_{2|1} = \mu_2 + \sigma_{21}\sigma_{11}^{-1}(X_1 - \mu_1)$$
$$\sigma_{2|1} = \sigma_{22} - \sigma_{21}\sigma_{11}^{-1}\sigma_{12}$$

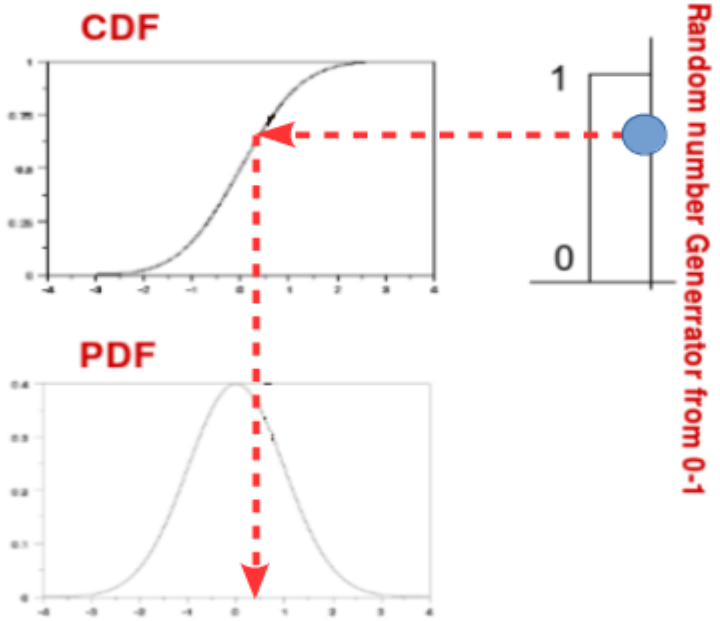
Machine Learning A Probabilistic Perspective by Kevin P. Murphy

Multivarient Normal Distribution

P-dimensional Gaussian PDF for the random vector **X** = [X_1 . . , X_P] has the following form:

$$N_p(\boldsymbol{\mu}, \Sigma) \sim \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} e^{-(\mathbf{x}-\boldsymbol{\mu})'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})/2}$$

Sampling from MVN



We generate a sample from univariate gaussian distribution, *N(0,1) by mapping the distribution of its Cumulative form*, i.e, $X_i \sim N(0,1)$

By extending this idea, for any μ and σ

$$X_i \sim N(\mu, \sigma^2) \sim \mu + \sigma N(0,1)$$

By extending the idea for the multivariant process we can sample a vector **X** from a multivariant gaussian distribution : $N(X; \bar{\mu}, \Sigma_{n\times n})$

$$X_i \begin{bmatrix} x_{1i} \\ . \\ . \\ x_{ni} \end{bmatrix} = \begin{bmatrix} \mu_1 \\ . \\ . \\ \mu_n \end{bmatrix} + LN\left(\begin{bmatrix} 0_1 \\ . \\ . \\ 0_n \end{bmatrix}, I_{n\times n} \right)$$

L is the *Cholesky Decomposition* of $\Sigma_{n\times n}, i.e., \Sigma_{n\times n} = LL^T$

Ref:https://web.mit.edu/urban_or_book/www/book/chapter7/7.1.3.html

Gaussian Process Regression

Extention of MVN in the infinite space brings us to the *Gaussian Process Regresion.*

$$f(x) \sim GP\Big(m(x), k(x,x\prime)\Big)$$

m(x)= mean

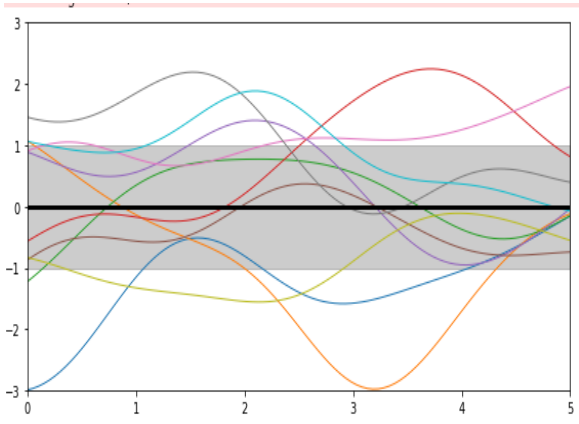
$$\kappa(x,x\prime) = exp[-(x-x\prime)^2/l]$$

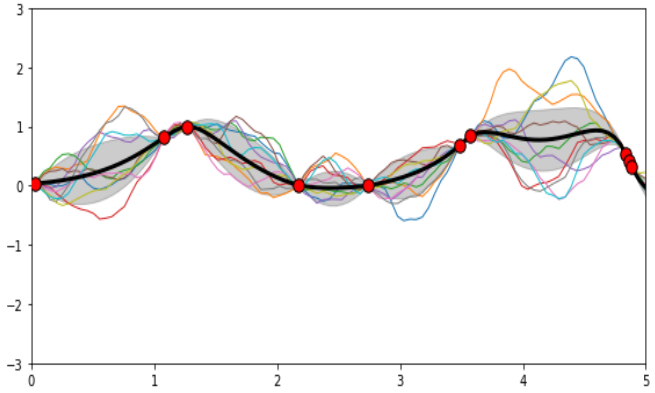
k(x,x`) = Covariance matrix or Karnal

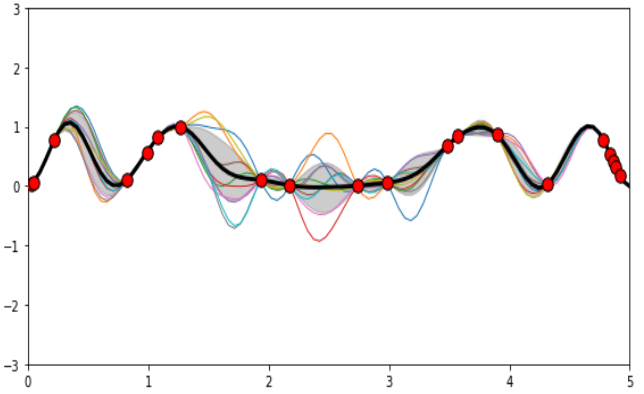
$$k(x,x\prime) = \begin{cases} = 0, & if \quad |x-x\prime| \to \infty \\ = 1, & if \quad x = x\prime \end{cases}$$

$X[x_1....x_n]_{n\to\infty}$, now, look like a function

Now, If we have data set!







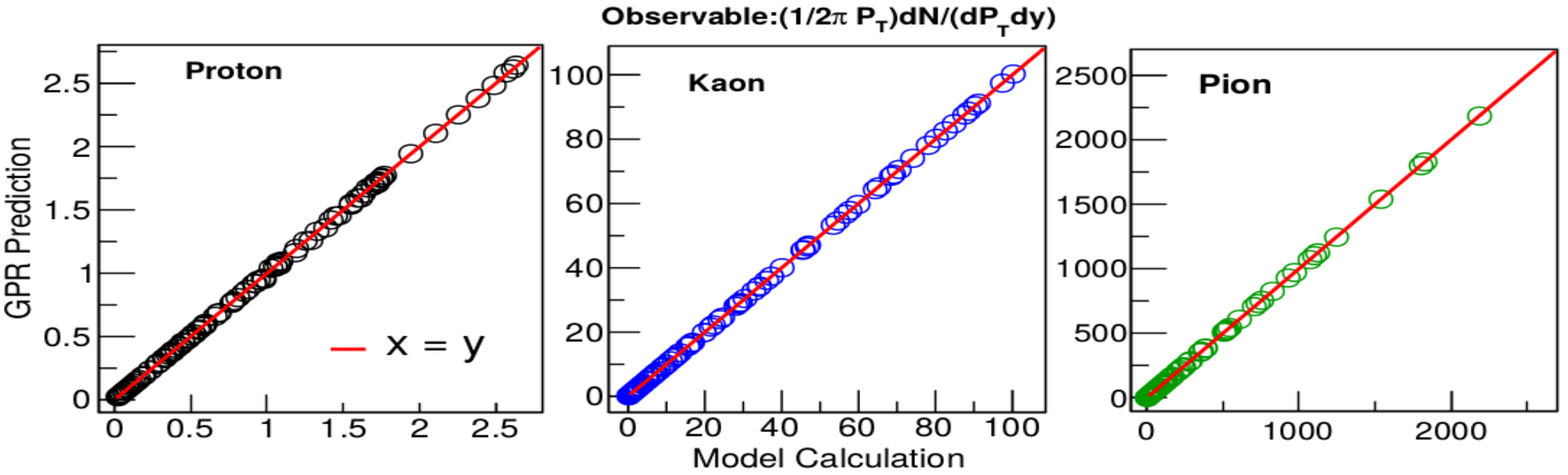
$$\mu_{2|1} = \mu_2 + \sigma_{21}\sigma_{11}^{-1}(X_1 - \mu_1)$$

$$\sigma_{2|1} = \sigma_{22} - \sigma_{21}\sigma_{11}^{-1}\sigma_{12}$$

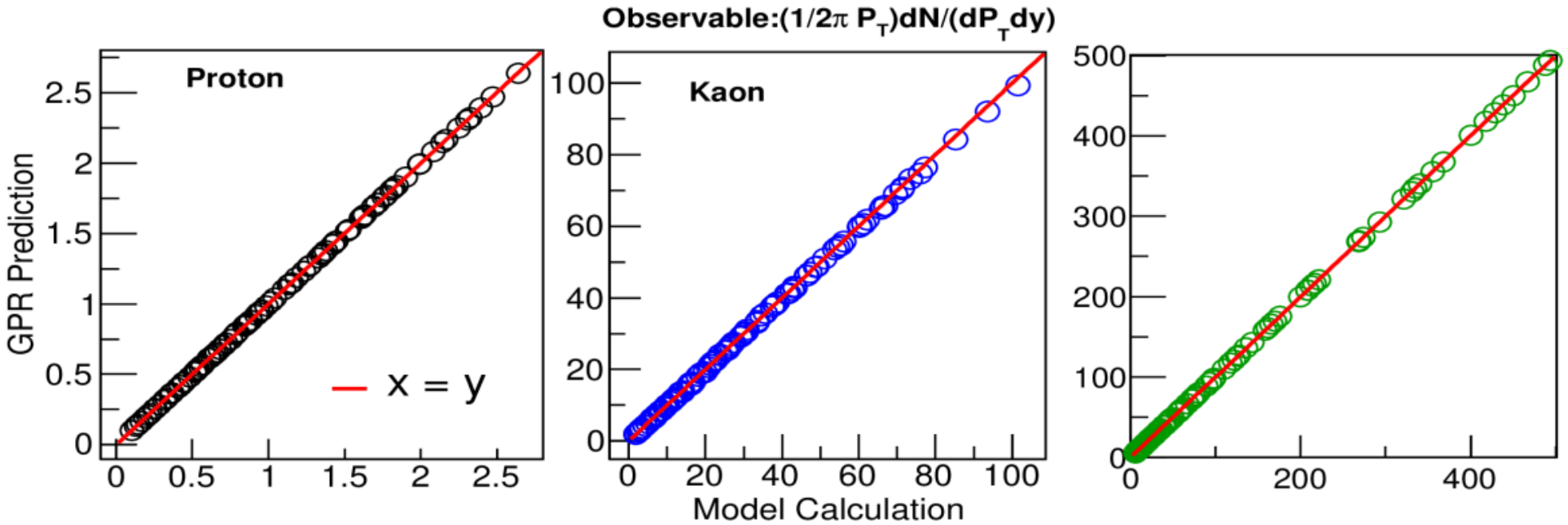
Recal

$$f_\star|f,(X,X_\star) \sim N\Big(\mu + K_\star^T K^{-1}(f-\mu), K_{\star\star} - K_\star^T K^{-1} K^\star\Big)$$

Ref: http://www.gaussianprocess.org



Centality: 0-5%



Centality: 20-30%

Principal Component Analayis (PCA)

Gaussian processes are fundamentally scalar functions, i.e., it take $[X]_{mxn}$ input and gives m output. **M Design Points** **N Input Parameters**

But a model may take $[X]_{mxn}$ input and computes $[Y]_{mxp}$ output.

P are the different observables.

An obvious workaround is to use independent GP for each of the p outputs.
But, we will dissipate the correlated information between different observables.

Way out of this is PCA!!

Principal Component Analysis (PCA): A dimensionality reduction technic (without loss of too much information), achieved by redistributing the data (Linear Transformation) in a set of new orthonormal basis.

New basis are the eigenvector of the covariance matrix of output $Y^T Y$

Ref: M. E. Tipping and C. M. Bishop, "Mixtures of Probabilistic Principal Component Analyzers

Numerical Steps For PCA

Step 1: Standeration of the model output Matrix $[Y]_{mxp,}$ i.e. centering and scaling each of the collum of Y to zero mean and unit variance.

Step 2: Eigen Value decomposition of covariance matrix Y^TY

Using *Singular Value Decomposition (SVD)* of the data matrix **Y**.

$$Y = U\Sigma V^T$$

U and V are the orthogonal matrices (***$U^TU=I$*** & ***$V^TV=I$***) known as left and right *Singular Vectors* respectively and Σ is a diagonal matrix containing the *Singular Values*.

$$Y^TY = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma^2 V^T$$

V is the eigenvectors of $\mathbf{Y^TY}$ and $\mathbf{\Sigma^2}$ contains eigenvalues on the diagonal.

Step 3: PCA transformation of the data, i.e., Z= YV, Z is mxp matrix.

Step 4: Use p independent GP emulators for each of the columns of Z.

Step 5: Transformed back to physical space by Y= VZ.

Dimentional Reduction

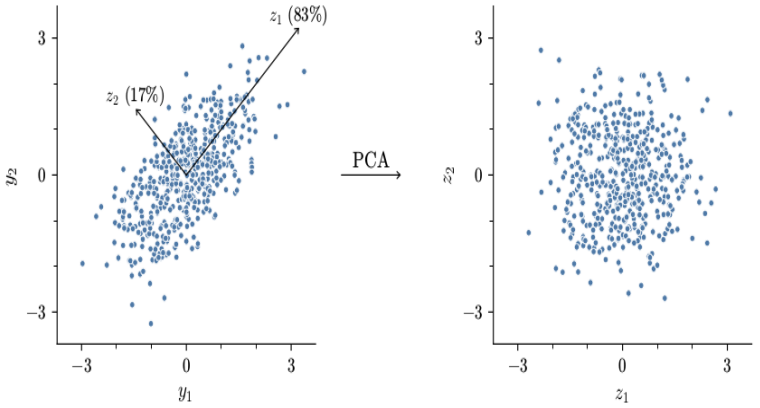
Output Matrxi:

$$Y_{p\times m} = \left[[y_1]_{p\times 1}.....[y_m]_{p\times 1}\right]_{1\times m}$$

Eigenvector Matrix:

$$V_{m\times m} = \left[[e_1]_{m\times 1}.....[e_m]_{m\times 1}\right]_{1\times m}$$

PCA Transformed Output:



$$Z_{p\times m} = YV = \begin{cases} Z_1 = e_{11}X_1 + e_{12}X_2 + ... + e_{1m}X_m; \\ Z_2 = e_{21}X_1 + e_{22}X_2 + ... + e_{2m}X_m; \\ ..... \\ ..... \\ Z_p = e_{p1}X_1 + e_{p2}X_2 + ... + e_{pm}X_p; \end{cases}$$

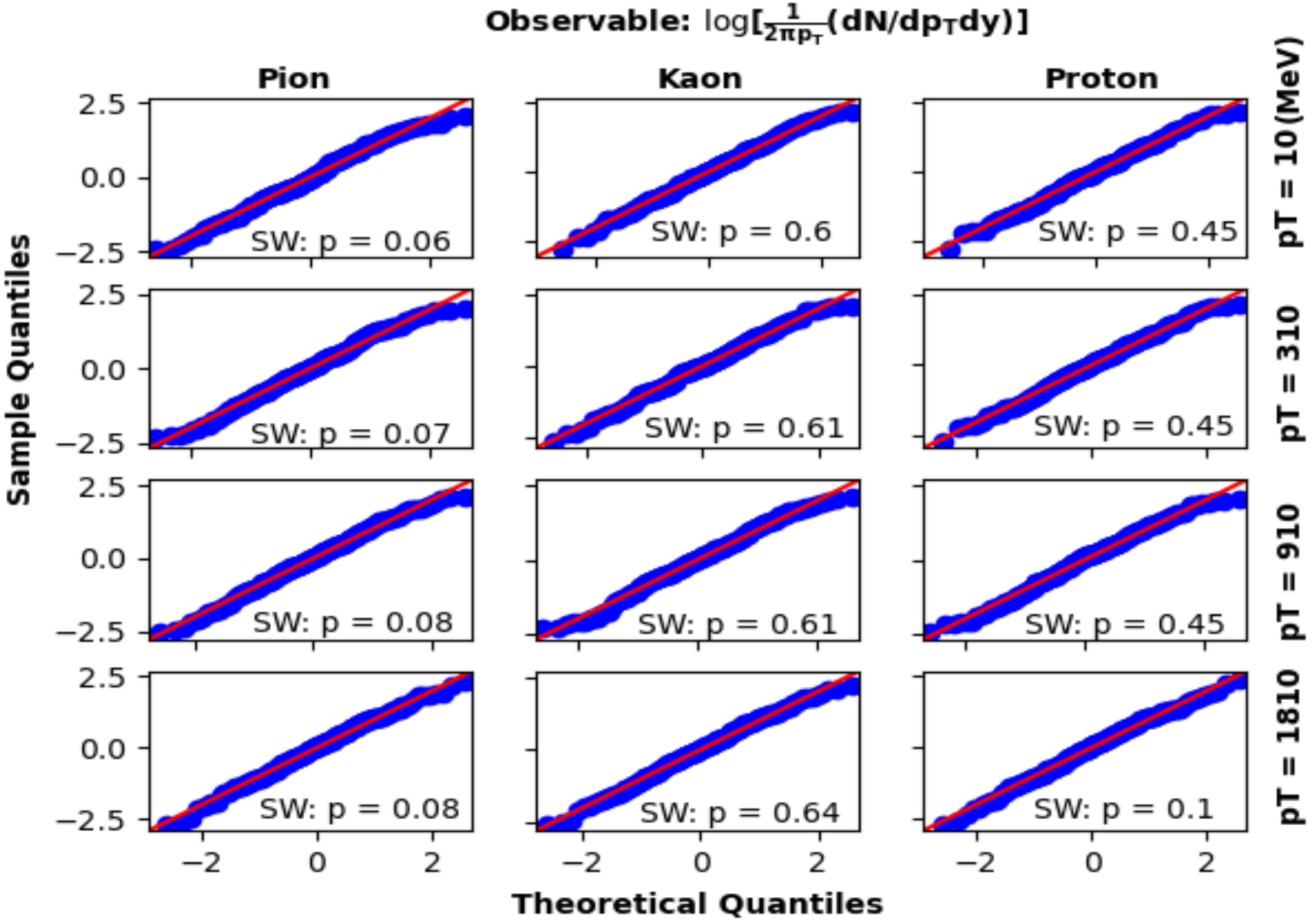
If the eigenvalue and corresponding eigenvector are in *decreasing order.*

$$\lambda_1 > \lambda_2 > ....\lambda_p.$$

First few principal components contained the maximum Variance or *Information.*

With cause of the minimum loss of information, we can take only the first few principal components and can neglect the rest.

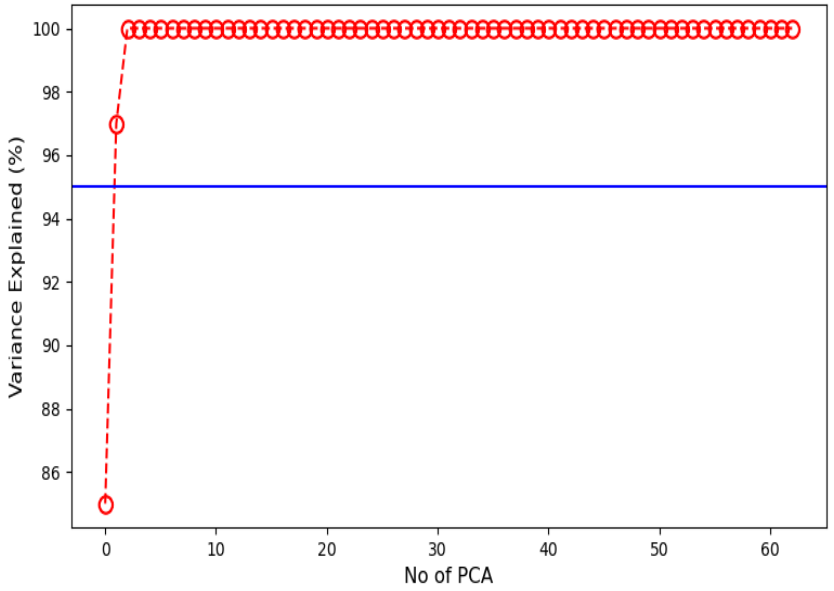
PCA Test



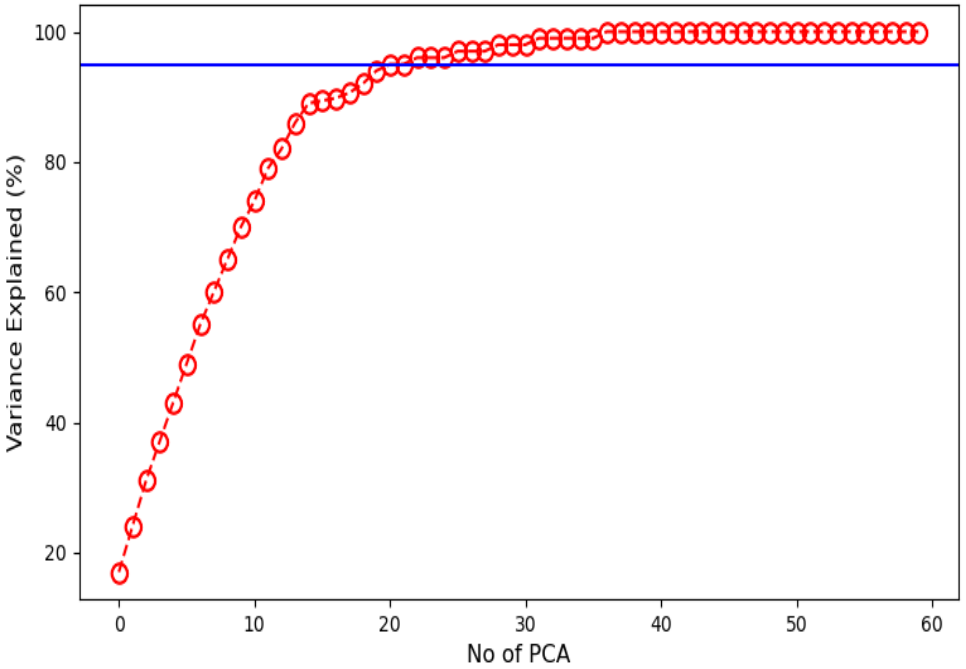
SW=Shapiro–Wilk test , p>0.5 is normal

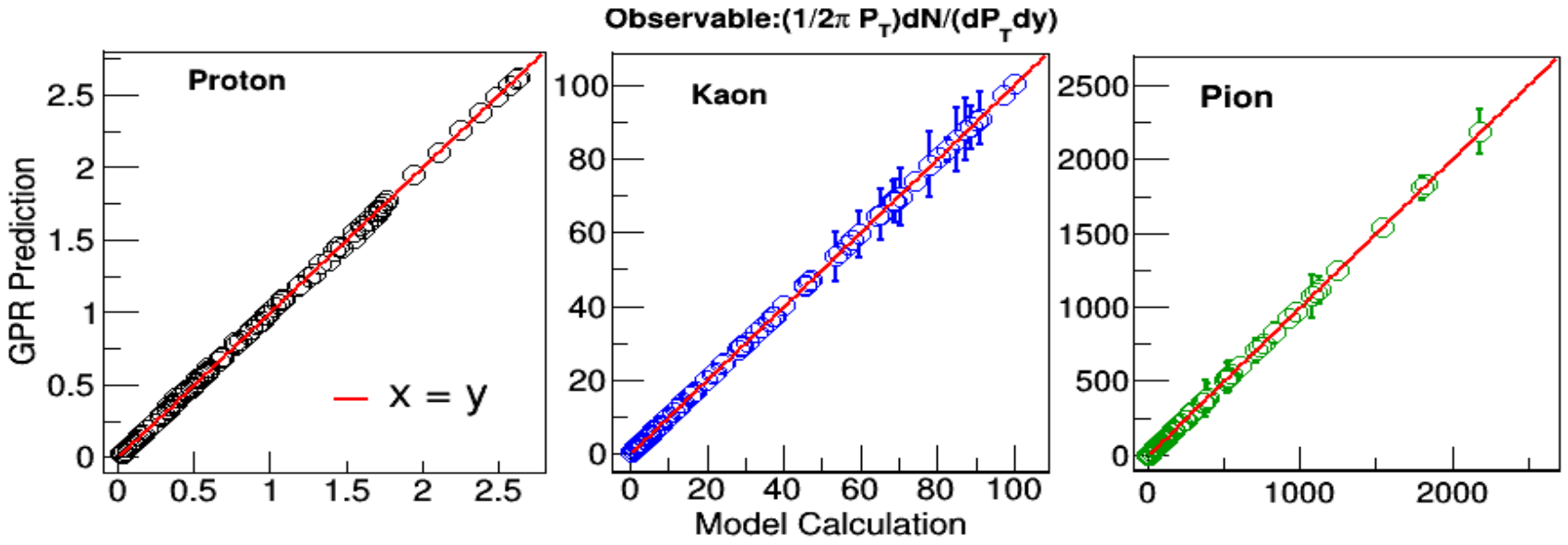
PCA Results

Single Centrality



All Centrality





Calibration

Extraction of model parameters by comparing experimental data

Bayesian inference

The new evidence should not completely determine your belief in a vacuum, it should update prior believe.

Posterior Distribution

$$P(\mathbf{x}|\mathcal{D}) \propto P(\mathcal{D}|\mathbf{x})P(\mathbf{x})$$

Prior Distribution

Likelihood

X : Parameter Vector and D represent all the collected data model + Experiment

Experiment Result

$$\mathbf{y}_e = \mathbf{y}_e^{\text{true}} + \boldsymbol{\epsilon}_e, \quad \boldsymbol{\epsilon}_e \sim \mathcal{N}(\mathbf{0}, \Sigma_e)$$

Model Result

$$\mathbf{y}_m(\mathbf{x}) = \mathbf{y}_m^{\text{ideal}}(\mathbf{x}) + \boldsymbol{\epsilon}_m, \quad \boldsymbol{\epsilon}_m \sim \mathcal{N}(\mathbf{0}, \Sigma_m),$$

$$\mathbf{y}_e = \mathbf{y}_m(\mathbf{x}_\star) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \Sigma), \quad \Sigma = \Sigma_e + \Sigma_m,$$

Likelihood

$$P(\mathcal{D}|\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^m \det \Sigma}} \exp\left\{-\frac{1}{2}[\mathbf{y}_m(\mathbf{x}) - \mathbf{y}_e]^{\mathrm{T}} \Sigma^{-1} [\mathbf{y}_m(\mathbf{x}) - \mathbf{y}_e]\right\}$$

Uniform Prior Distribution

$$P(\mathbf{x}) \propto \begin{cases} 1 & \text{if } \min(x_i) \leq x_i \leq \max(x_i) \text{ for all } i \\ 0 & \text{else.} \end{cases}$$

It is better to deal with the log Prior

$$\log P(\mathbf{x}|\mathcal{D}) = \log P(\mathcal{D}|\mathbf{x}) + \log P(\mathbf{x}) + \text{const},$$

$$\log P(\mathcal{D}|\mathbf{x}) = -\frac{1}{2}\mathbf{d}^{\mathrm{T}} \Sigma^{-1} \mathbf{d} - \frac{1}{2} \log(\det \Sigma), \quad \mathbf{d} = \mathbf{y}_m(\mathbf{x}) - \mathbf{y}_e,$$

$$\log P(\mathcal{D}|\mathbf{x}) = -\frac{1}{2}\mathbf{d} \cdot \boldsymbol{\alpha} - \sum_i \log L_{ii}. \qquad \boldsymbol{\alpha} = \Sigma^{-1}\mathbf{d}.$$

$$\Sigma = LL^{\mathrm{T}}, \quad \boldsymbol{\alpha} = \Sigma^{-1}\mathbf{d}.$$

$$LL^{\mathrm{T}}\boldsymbol{\alpha} = \mathbf{d}$$

We Maximize the log probability using MCMC Algorithm

We Run MCMC on

$$P(\mathbf{z}_{\rm exp}|\mathbf{x}_\star) \propto \exp\left\{-\frac{1}{2}(\mathbf{z}_\star - \mathbf{z}_{\rm exp})^\intercal \Sigma_z^{-1}(\mathbf{z}_\star - \mathbf{z}_{\rm exp})\right\}$$

Error Estimation

$$\Sigma_e = \Sigma_e^{\rm stat} + \Sigma_e^{\rm sys}.$$

Statistical uncertainty are uncorrelated

$$\Sigma_e^{\rm stat} = {\rm diag}\left[(\sigma_1^{\rm stat})^2, (\sigma_2^{\rm stat})^2, \ldots, (\sigma_m^{\rm stat})^2\right]$$

Systematic uncertainty usually has a correlation structure

$$\Sigma_{ij} = {\rm cov}(y_i, y_j) = \rho_{ij}\sigma_i\sigma_j \qquad \rho_{ij}^{\rm sys} = \exp\left[-\frac{1}{2}\left(\frac{c_i - c_j}{\ell}\right)^2\right]$$

C is the midpoint of the pT Bin and l=1

$$\Sigma_m = \Sigma_m^{\rm pred} + \Sigma_m^{\rm stat} + \Sigma_m^{\rm sys}.$$

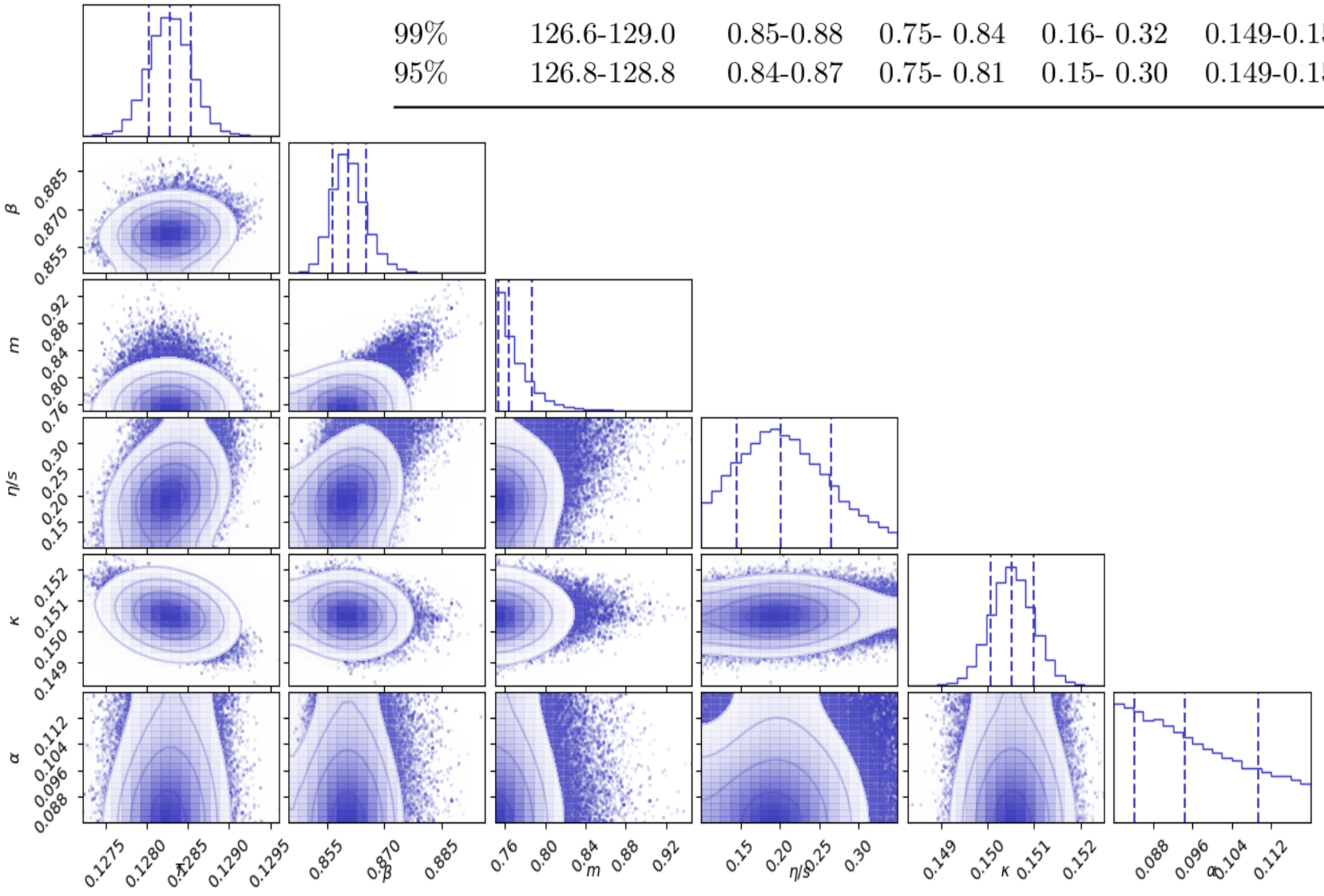
$$\Sigma_m = \Sigma_m^{\rm GP} + \Sigma_m^{\rm sys}.$$

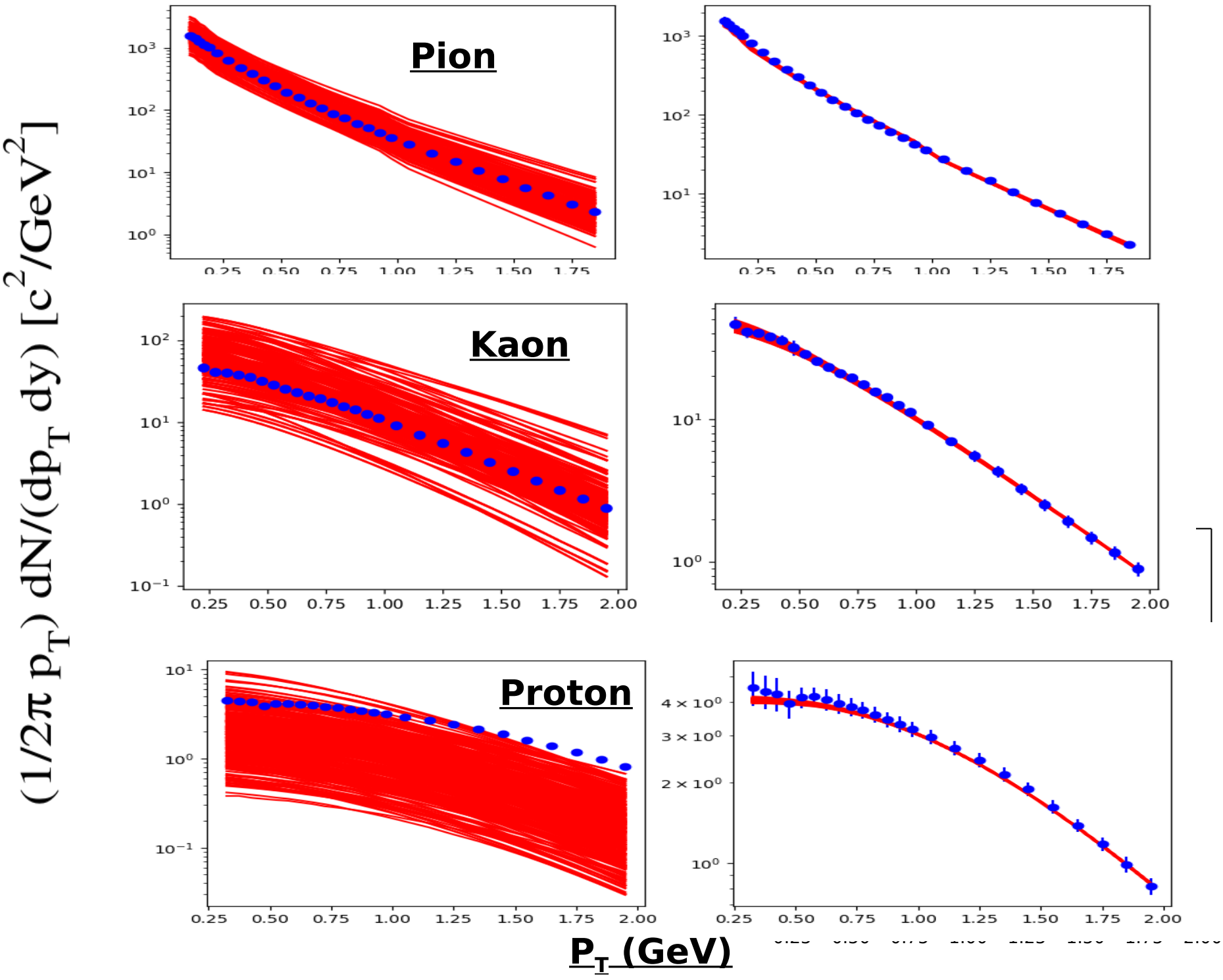
Σ = Predictive Covariance in PC space and V is the PCA transformation matrix

$$\Sigma_m^{\rm GP} = V\Sigma_{m,z}^{\rm GP}V^{\rm T}$$

MCMC Result

C.I.	T_F	β_0	m	η/s	κ	α_0
99%	126.6-129.0	0.85-0.88	0.75- 0.84	0.16- 0.32	0.149-0.152	0.08-0.12
95%	126.8-128.8	0.84-0.87	0.75- 0.81	0.15- 0.30	0.149-0.151	0.08-0.12




Pion
Kaon
Proton
$(1/2\pi\, p_T)\, dN/(dp_T\, dy)\ [c^2/GeV^2]$
P_T (GeV)


Thank
you

Programming libraries and Sources

Open-Sources Python Libraires

- **NumPy**
- **SciPy**
- **scikit-learn**
- **H5py**
- **matplotlib**
- **emcee**

Other Sources: Reading material

- *Bayesian analysis of computer code outputs: A tutorial by A. O'Hagan*
- *Nonparametric Bayesian Methods (Gaussian Processes) Jun Zhu*
- *https://arxiv.org/abs/1902.11082.*
- *An Intuitive Tutorial to Gaussian Processes Regression Jie Wang.*
- *Bayesian estimation of the specific shear and bulk viscosity of quark–gluon plasma/ Nat. Phys. 15, 1113–1117 (2019).*