

Gaussian Process

Useful reference:

Gaussian Processes for Machine Learning by Rasmussen and Williams @ www.GaussianProcess.org/gpml

Essential Definitions: Vectors, Matrices, Tensors

\mathbf{x} Vector: (denoted in ***bold lowercase***) elements of a **Vector Space** V

V Vector Space: a set of elements \mathcal{V} with the operations of addition and scalar multiplication, such that

$\mathcal{V} + \mathcal{V} \in \mathcal{V}$ a vector in the space plus another in the space yields another in the space

$\mathbb{R} \times \mathcal{V} \in \mathcal{V}$ a vector times a real number in the space yields a vector in the space

You are very familiar with
Euclidean vectors in \mathbb{R}^3

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

or perhaps even \mathbb{R}^n
(*tuples of n real numbers*)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Gaussian Distribution

Gaussian distribution is the most well-studied distribution in science, engineering, and ML

- *ubiquity (often stemming from **central limit theorem**)*
- *computationally convenient*

univariate

$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

multivariate

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{m/2} |\boldsymbol{\Sigma}|^{-1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

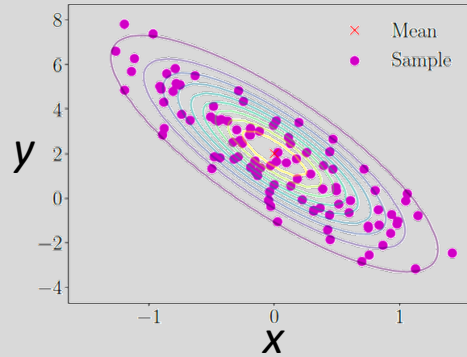
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \text{for short}$$

standard normal distribution $\rightarrow \mathbf{0}$ mean and \mathbf{I}_n as the covariance matrix

Basic tenant of Gaussian distributions: *doing things with Gaussians results in Gaussians*

Gaussian Distribution

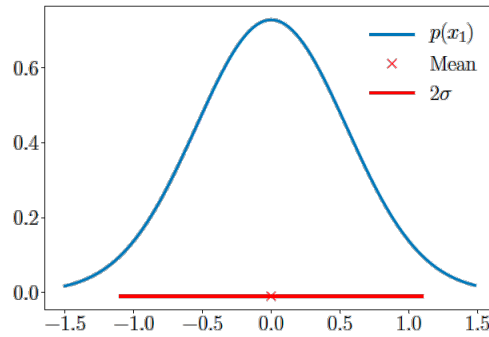
Consider the joint distribution over X and Y :



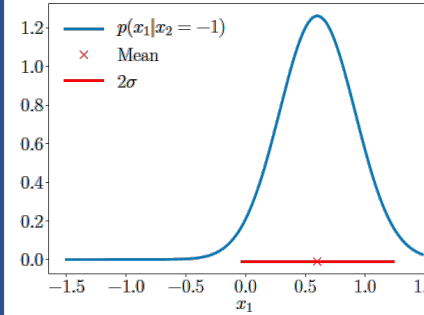
$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N} \left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right)$$

- *Marginals over Gaussians are Gaussians*

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) d\mathbf{y} = \mathcal{N}(\mathbf{x} | \mu_x, \Sigma_{xx})$$



- *Conditionals of Gaussians are Gaussians*



$$p(\mathbf{x} | \mathbf{y}) = \mathcal{N}(\mu_{x|y}, \Sigma_{x|y})$$

$$\mu_{x|y} = \mu_x + \Sigma_{xy} \Sigma_{yy}^{-1} (\mathbf{y} - \mu_y)$$

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx}$$

- *The product of Gaussians is Gaussian*

$$\mathcal{N}(\mathbf{x} | \mathbf{a}, \mathbf{A}) \mathcal{N}(\mathbf{x} | \mathbf{b}, \mathbf{B}) \rightarrow c \mathcal{N}(\mathbf{x} | \mathbf{c}, \mathbf{C})$$

$$\mathbf{C} = (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1}$$

$$\mathbf{c} = \mathbf{C}(\mathbf{A}^{-1} \mathbf{a} + \mathbf{B}^{-1} \mathbf{b}) \quad c = \mathcal{N}(\mathbf{a} | \mathbf{b}, \mathbf{A} + \mathbf{B})$$

- *The sum over independent Gaussian variables is... Gaussian*

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}) p(\mathbf{y})$$

$$p(\mathbf{x} + \mathbf{y}) = \mathcal{N}(\mu_x + \mu_y, \Sigma_x + \Sigma_y)$$

- *Any linear transformation of Gaussians is Gaussian* $p(a\mathbf{x} + b\mathbf{y}) = \mathcal{N}(a\mu_x + b\mu_y, a^2\Sigma_x + b^2\Sigma_y)$

What are Gaussian Processes?

A **Gaussian process (GP)** is a probability distribution over possible functions; any combination of these functions jointly Gaussian distributed


Key Points:

- GPs can be used as models for both regression and classification
- Since the model is a probability distribution (over a function space that is Gaussian distributed), we can calculate means AND variances
- The **mean function** is calculated from the posterior (recall Baye's) distribution of possible functions → regression predictions
- The **variances** provide native uncertainty measures on these predictions, which are typically absent in other ML models (e.g., ANNs)
- The model, as a posterior, can be updated based on new observations

In “math” speak:

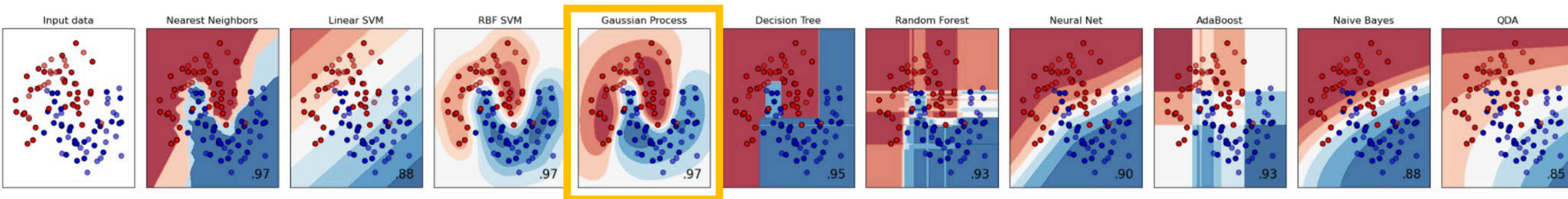
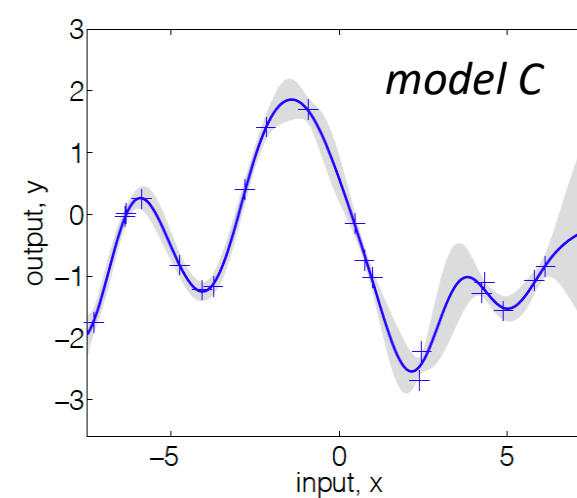
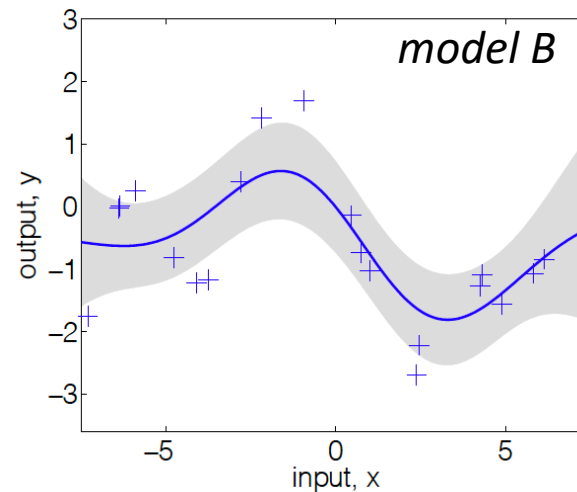
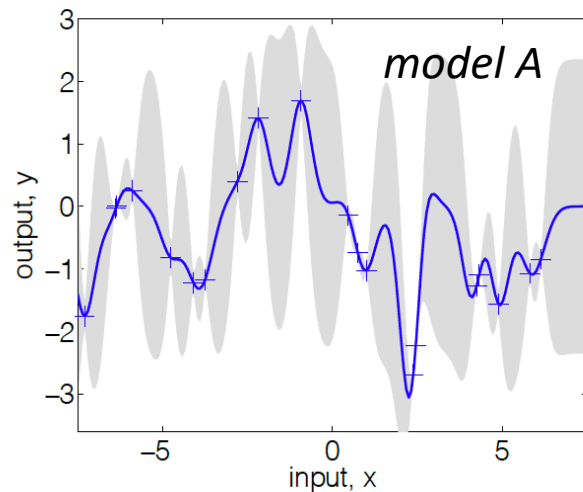
$$f(x) \sim \mathcal{N}(\mu(x), \mathbf{K}(x, x'))$$

covariance/Kernel
function that dictates
model “smoothness”



Why should we use them?

- Our observations may be “**noisy**” and it is important to capture that behavior
- They allow you to incorporate **prior knowledge**
- They provide a reasonable framework to **regulate model complexity** and more reliably **indicate model uncertainty**

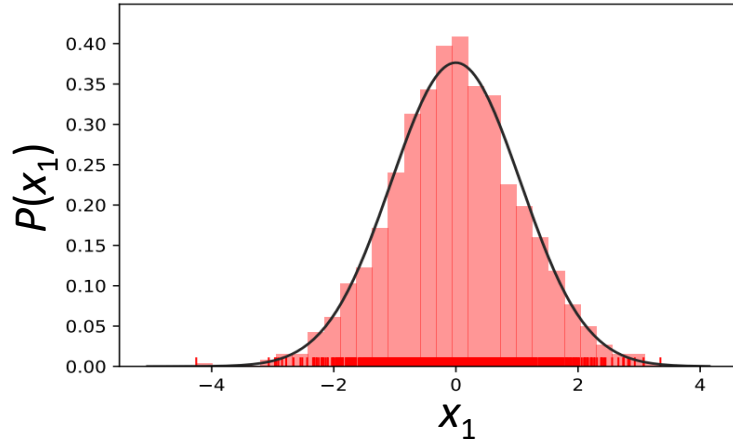


Some basic intuition underlying Gaussian Processes

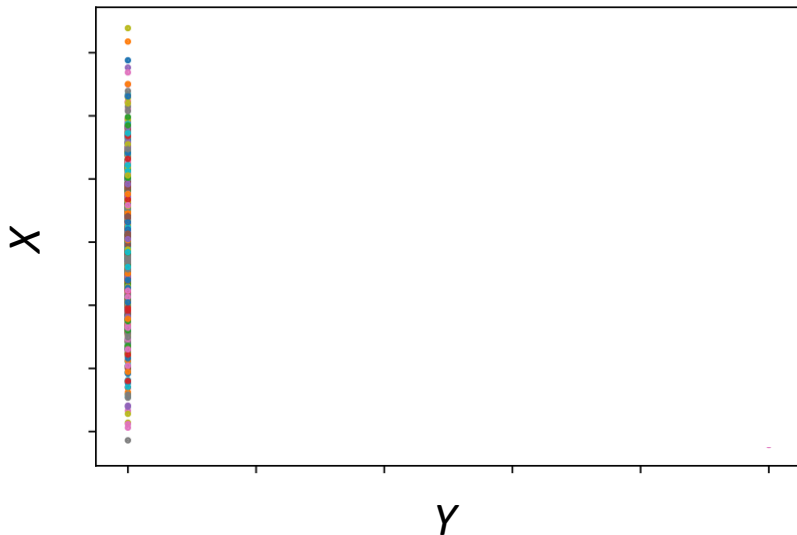
Suppose we have a random variable X_1 with distribution

$$P_{X_1}(x_1) \sim \mathcal{N}(\mu, \sigma^2)$$

We can generate samples from this distribution



and then project them onto a new axis

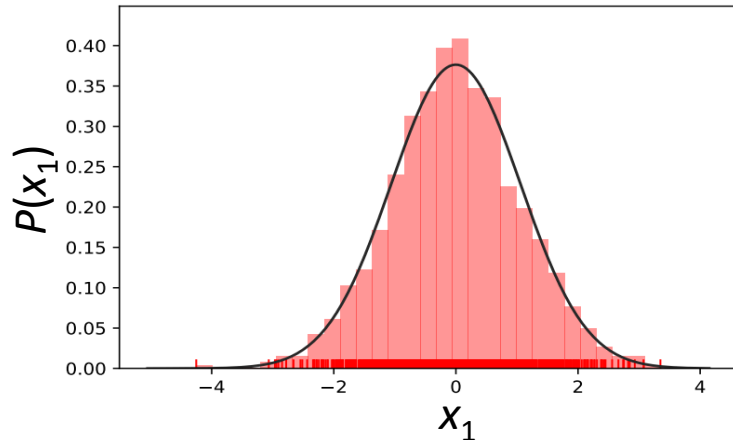


Some basic intuition underlying Gaussian Processes

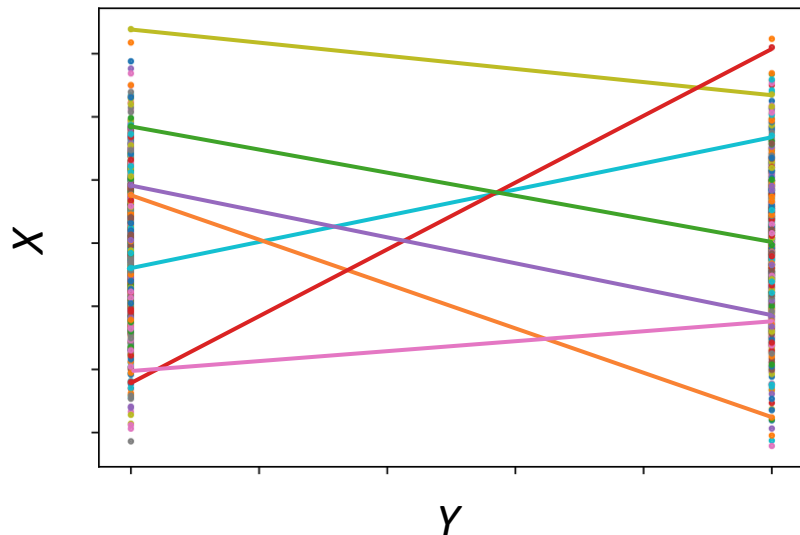
Suppose we have a random variable X_1 with distribution

$$P_{X_1}(x_1) \sim \mathcal{N}(\mu, \sigma^2)$$

We can generate samples from this distribution



and then project them onto a new axis



We can do this for another set of samples to obtain two independent Gaussian vectors

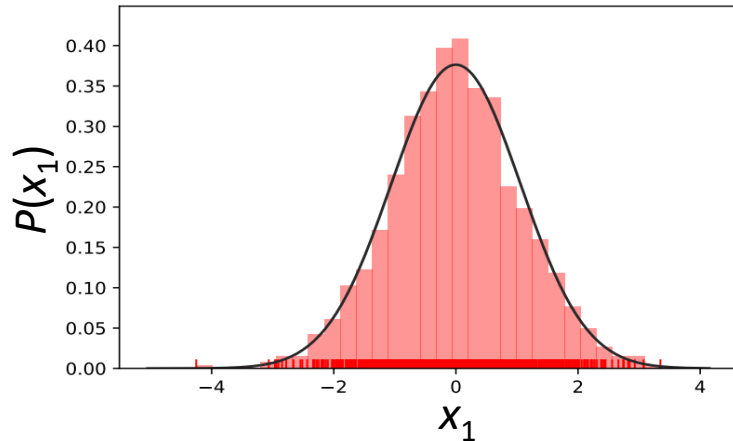
Then by connecting random points between the two vectors, we would get a set of linear functions

Some basic intuition underlying Gaussian Processes

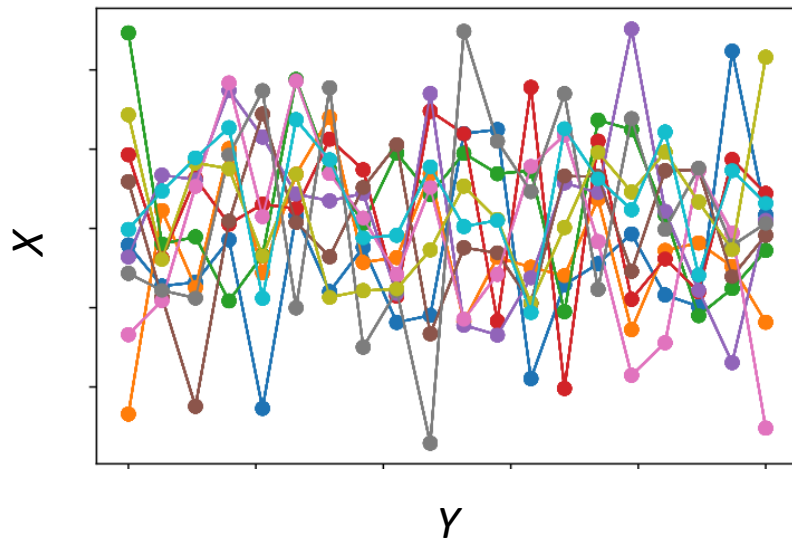
Suppose we have a random variable X_1 with distribution

$$P_{X_1}(x_1) \sim \mathcal{N}(\mu, \sigma^2)$$

We can generate samples from this distribution



and then project them onto a new axis



*using more and more
Gaussian vectors, we could
represent more complex
functions (showing only ten),
but the resulting functions are
inherently noisy!*

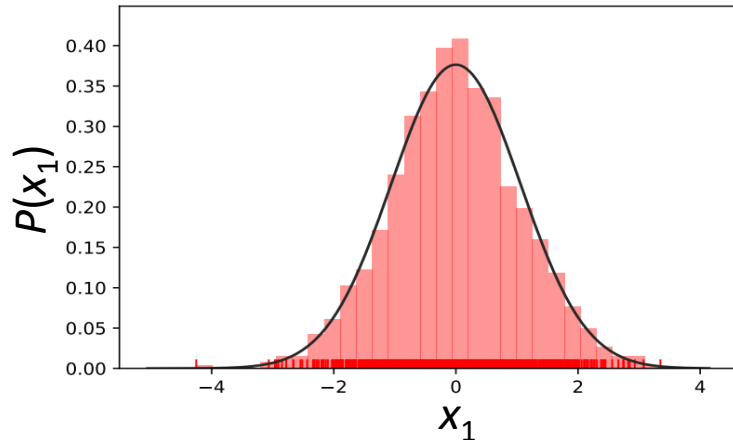
Why?

Some basic intuition underlying Gaussian Processes

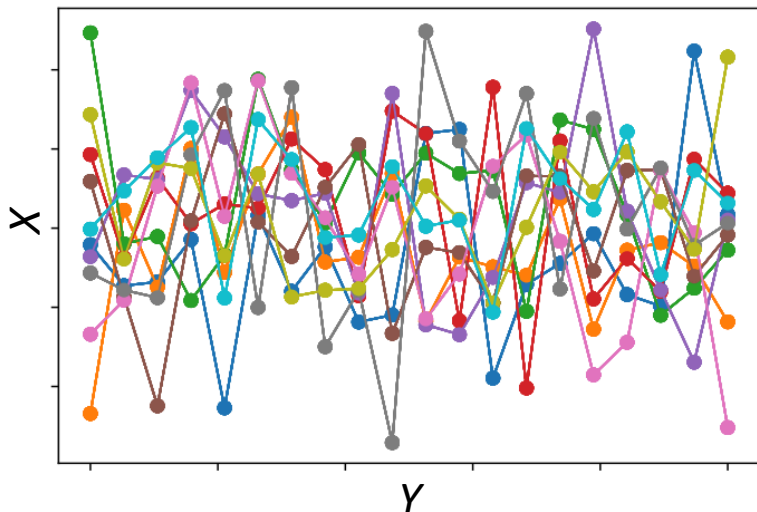
Suppose we have a random variable X_1 with distribution

$$P_{X_1}(x_1) \sim \mathcal{N}(\mu, \sigma^2)$$

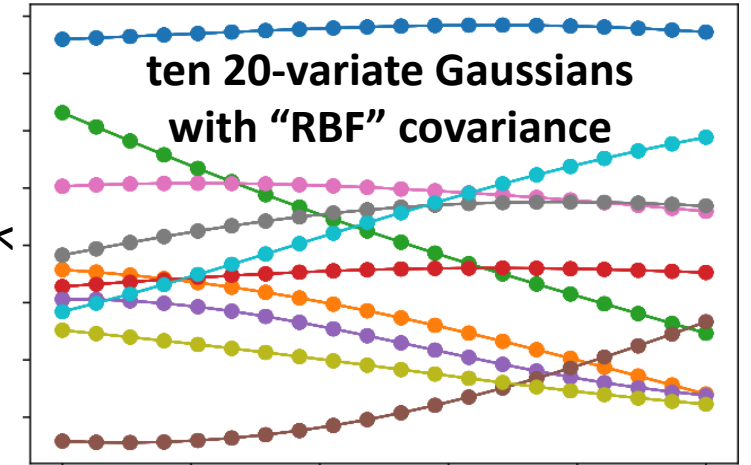
We can generate samples from this distribution



and then project them onto a new axis



*using multivariate
Gaussians with true
covariance enables
more smoothly
varying functions*

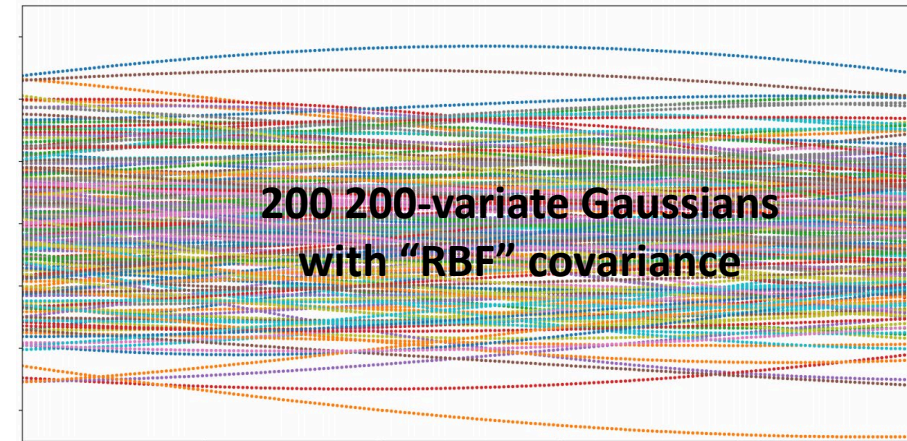


e.g.,

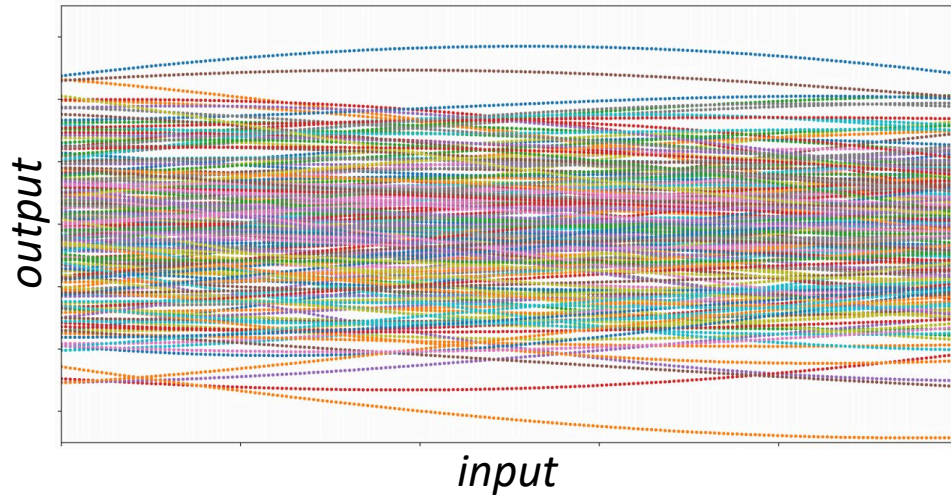
$$K(x_i, x_j) = \exp \left[-\frac{(x_i - x_j)^2}{2} \right]$$

*using more and more
Gaussian vectors, we could
represent more complex
functions (showing only ten),
but the resulting functions are
inherently noisy!*

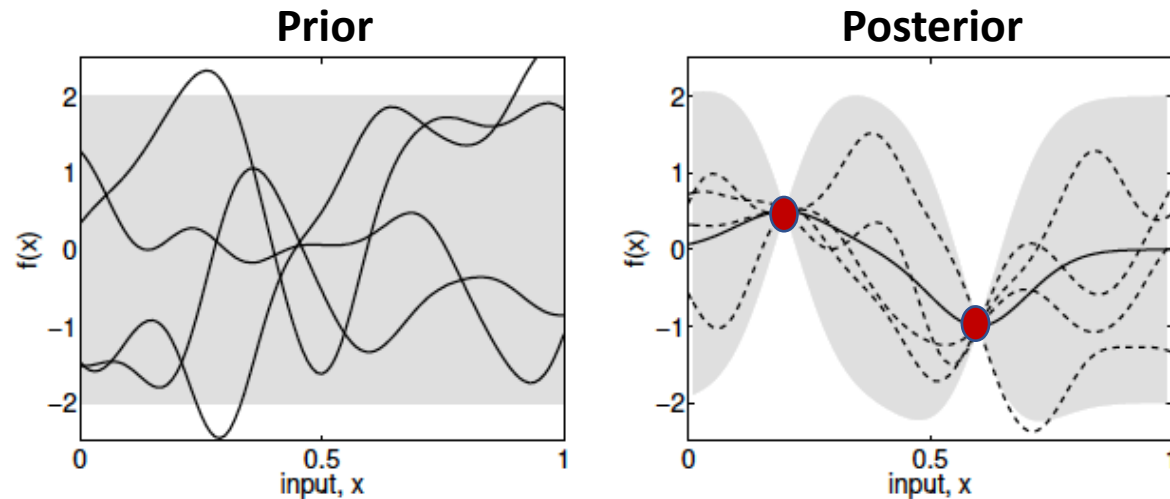
**...because the Gaussian
vectors are independent**



Some basic intuition underlying Gaussian Processes



- If the dimensionality of the ***multivariate Gaussians*** (MVG) becomes infinite \rightarrow a continuous function
- With an infinite number of such functions, we could make predictions at any point
- Functions as MVGs is our initial assumption (***prior***)
- The functions shown provide expected outputs as a function of inputs without having observed any data



- As data is collected, we can downselect from infinite functions to only the functions that describe the data/observations \rightarrow ***posterior*** (kind of like wavefunctions in quantum mechanics, if that helps)
- As we add more data, the current posterior can be used as the prior to obtain a new posterior

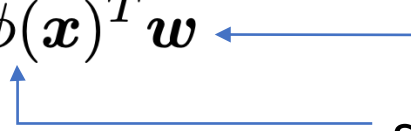
Theoretical Development of GPs for Regression

A **Gaussian process (GP)** is a probability distribution over possible functions; any combination of these functions jointly Gaussian distributed

$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ this is a stochastic function, i.e., a collection of random variables

$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ its distribution is characterized by its mean and covariance function

$$k(\mathbf{x}, \mathbf{x}') = \text{Cov}[f(\mathbf{x}), f(\mathbf{x}')] = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$$

A simple example $f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}$  weighting coefficients BUT $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$
some vector field, e.g., $\phi(\mathbf{x})^T = (\sum_i x_i, \sum_i x_i^2, \dots, \sum_i x_i^m)$

for such a process, it is easy to show that $\mu(\mathbf{x}) = 0; k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \Sigma_p \phi(\mathbf{x}')$

that is just to say that $f(\mathbf{x})$ and $f(\mathbf{x}')$ are indeed jointly Gaussian with the above mean and covariance \rightarrow any number of input points will be jointly Gaussian

Theoretical Development of GPs for Regression

Let's suppose we have a dataset now $\mathcal{D} := \{(\mathbf{x}_i, f_i = f(\mathbf{x}_i)) \text{ for } i = 1, \dots, n\}$

\mathbf{f}, \mathbf{x}

training

$\mathbf{f}_*, \mathbf{x}_*$

test

These two sets should be jointly distributed according to our Gaussian prior as

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} k(\mathbf{x}, \mathbf{x}) & k(\mathbf{x}, \mathbf{x}_*) \\ k(\mathbf{x}_*, \mathbf{x}) & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right) \quad \text{prior distribution}$$

Now, to get the posterior distribution, we must restrict our prior distribution to only feature the functions that agree with our observations/training data (i.e., we should make our prior distribution conditional on our observations)

from math review

$$\mathbf{f}_* | \mathbf{f}, \mathbf{x}, \mathbf{x}_* \sim \mathcal{N}(k(\mathbf{x}_*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1} \mathbf{f},$$

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N} \left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right)$$

$$k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}k(\mathbf{x}, \mathbf{x}_*))$$

$$p(\mathbf{x} | \mathbf{y}) = \mathcal{N}(\mu_{x|y}, \Sigma_{x|y}) \quad \Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}$$

$$\mu_{x|y} = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(\mathbf{y} - \mu_y)$$

easy to just sample function values from the posterior by computing the mean and covariance in the above!

Theoretical Development of GPs for Regression

Let's suppose we have a dataset now $\mathcal{D} := \{(\mathbf{x}_i, f_i = f(\mathbf{x}_i)) \text{ for } i = 1, \dots, n\}$

\mathbf{f}, \mathbf{x}
training

$\mathbf{f}_*, \mathbf{x}_*$
test

In the prior example, our observations were treated as Noise-free; in many scenarios, we would instead encounter

$$y = f(\mathbf{x}) + \varepsilon \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \quad \text{cov}(\mathbf{y}, \mathbf{y}) = k(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}$$

This would slightly modify our joint distribution as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} k(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I} & k(\mathbf{x}, \mathbf{x}_*) \\ k(\mathbf{x}_*, \mathbf{x}) & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right)$$

Then, the posterior is obtained by again conditioning on our observations
these are governing equations

$$\mathbf{f}_* | \mathbf{y}, \mathbf{x}, \mathbf{x}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*, \mathbf{f}_*))$$

$$\bar{\mathbf{f}}_* := \mathbb{E}[\mathbf{f}_* | \mathbf{y}, \mathbf{x}, \mathbf{x}_*] = k(\mathbf{x}_*, \mathbf{x}) [k(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}$$

$$\mathbb{V}[\mathbf{f}_*] = \text{cov}(\mathbf{f}_*, \mathbf{f}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{x}) [k(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}]^{-1} k(\mathbf{x}, \mathbf{x}_*)$$

Log Marginal Likelihood

“marginal likelihood” or “evidence”

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{x})p(\mathbf{f}|\mathbf{x})d\mathbf{f} \quad \rightarrow \text{how likely would we observe the data that we have given the data originates from } \mathbf{f}(\mathbf{x})$$

Assuming a Gaussian process, we can show that

$$\log p(\mathbf{y}|\mathbf{x}) = -\frac{1}{2}\mathbf{y}^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K} + \sigma_n^2 \mathbf{I}| - \frac{n}{2} \log 2\pi$$

“log marginal likelihood”

- All the terms necessary to evaluate this are also used to obtain the mean function
- Useful metric for evaluating different models
- Hyperparameter optimization based on minimizing negative log likelihood

Basic Algorithm Outline

1. Define data/models: $\mathbf{x}, \mathbf{y}, k, \sigma_n^2, \mathbf{x}_*$

2. Perform Cholesky decomposition

$$\mathbf{L} := \text{cholesky}(k(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I})$$

3. Compute predictive mean function

$$\boldsymbol{\alpha} := \mathbf{L}^T \setminus (\mathbf{L} \setminus \mathbf{y})$$

$$\bar{f}_* := k(\mathbf{x}, \mathbf{x}_*)^T \boldsymbol{\alpha}$$

4. Compute predictive variance

$$\mathbf{v} := \mathbf{L} \setminus k(\mathbf{x}, \mathbf{x}_*)$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^T \mathbf{v}$$

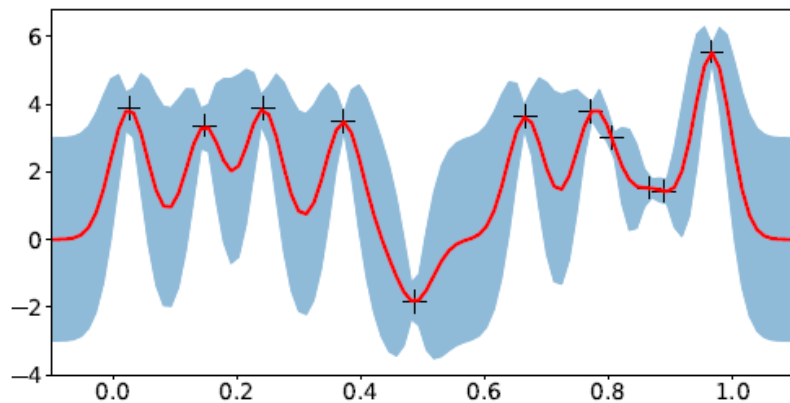
5. Compute log marginal likelihood

$$\log p(\mathbf{y}|\mathbf{x}) = -\frac{1}{2} \mathbf{y}^T \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi$$

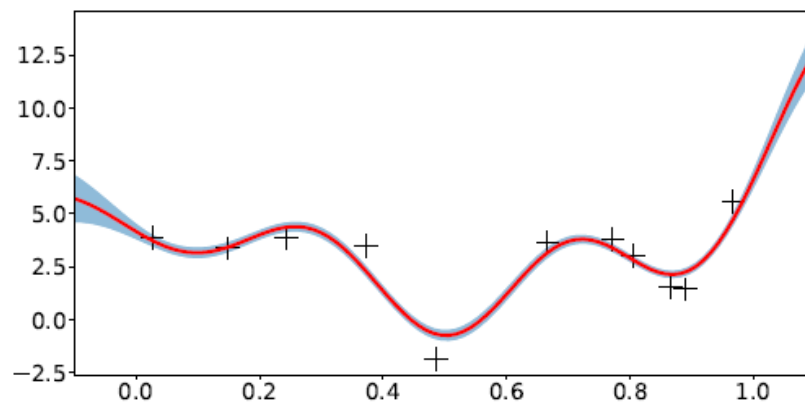
6. Return mean, variance, log marginal likelihood

Non-parametric, but there are still parameters

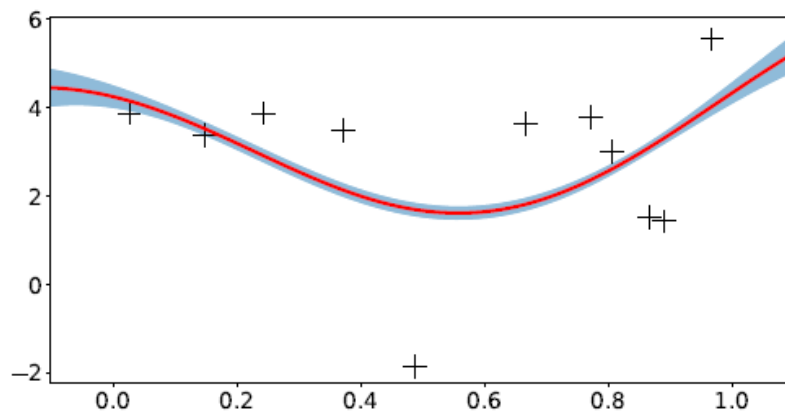
$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp \left(-\frac{1}{2l} (\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j) \right)$$



(a) $l = \text{small}$



(b) $l = \text{medium}$



(c) $l = \text{large}$

$$\Theta^* = \arg \max_{\Theta} \log p(\mathbf{y} | \mathbf{X}, \Theta)$$

$$\bar{\mathbf{f}}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*, \Theta \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*))$$

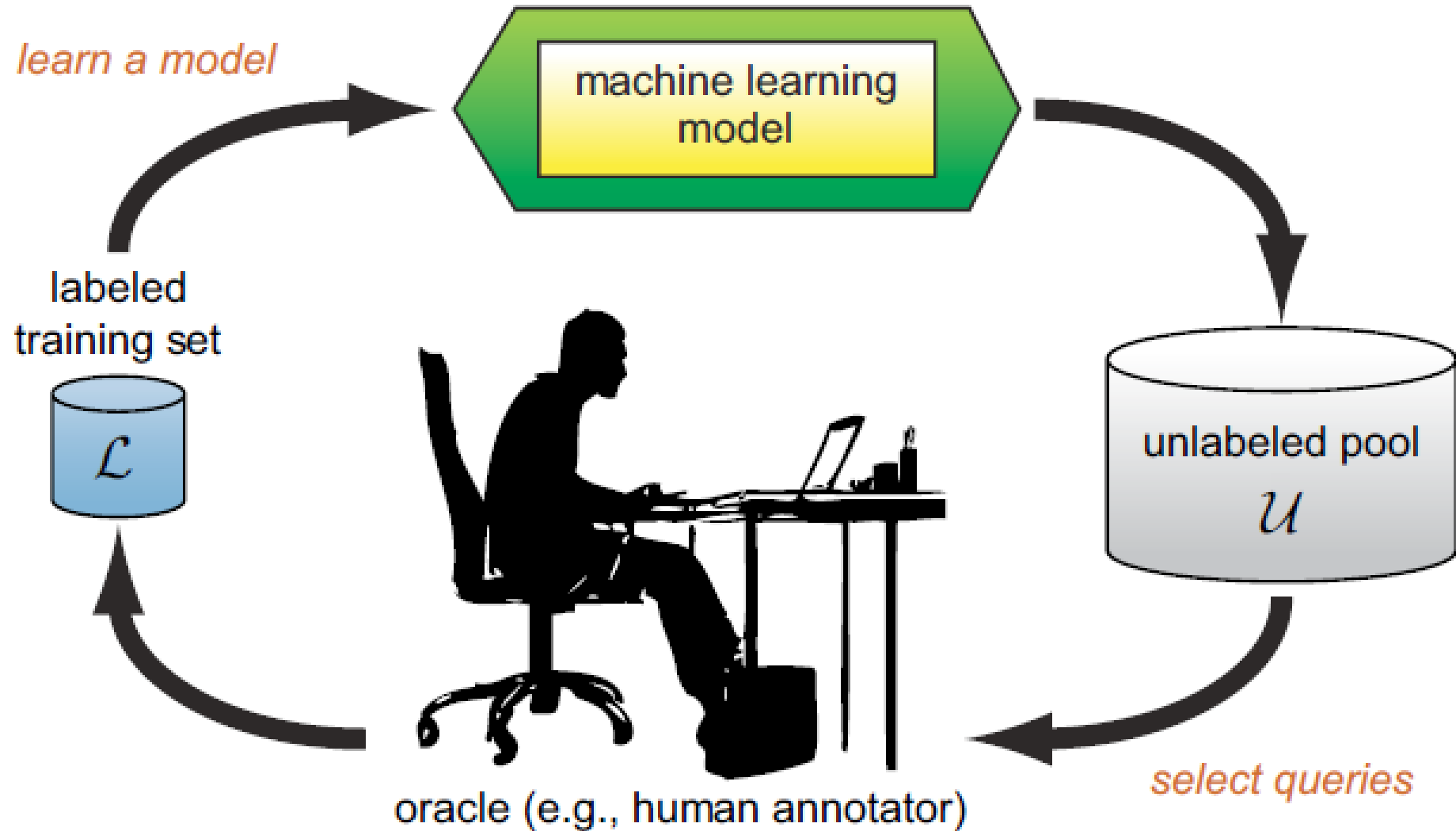
Active Learning

The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabeled data instances to be labeled by an oracle (e.g., a human annotator). Active learning is well-motivated in many modern machine learning problems, where unlabeled data may be abundant or easily obtained, but labels are difficult, time-consuming, or expensive to obtain.

Definition

Active learning (sometimes called “query learning” or “optimal experimental design” in the statistics literature) is a subfield of machine learning and, more generally, artificial intelligence. The key hypothesis is that if the learning algorithm is allowed to choose the data from which it learns—to be “curious,” if you will—it will perform better with less training.

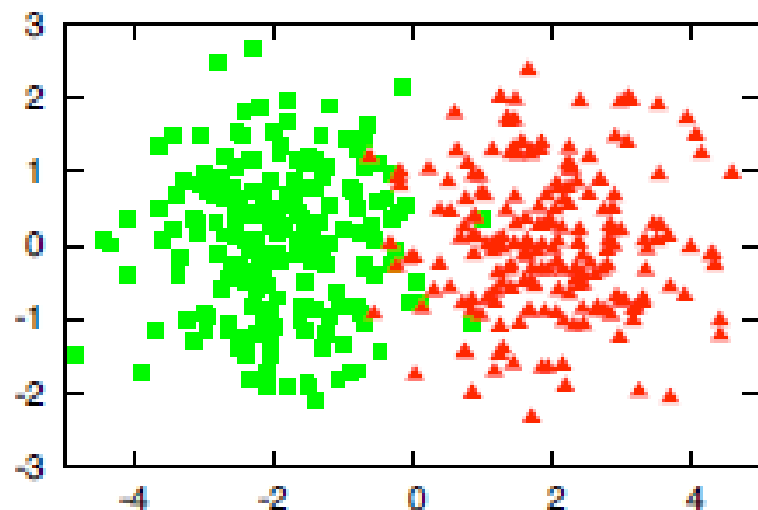
Active Learning



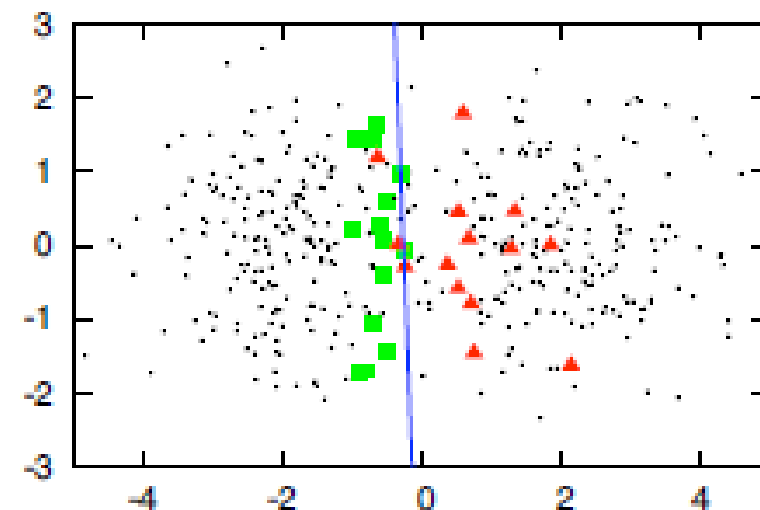
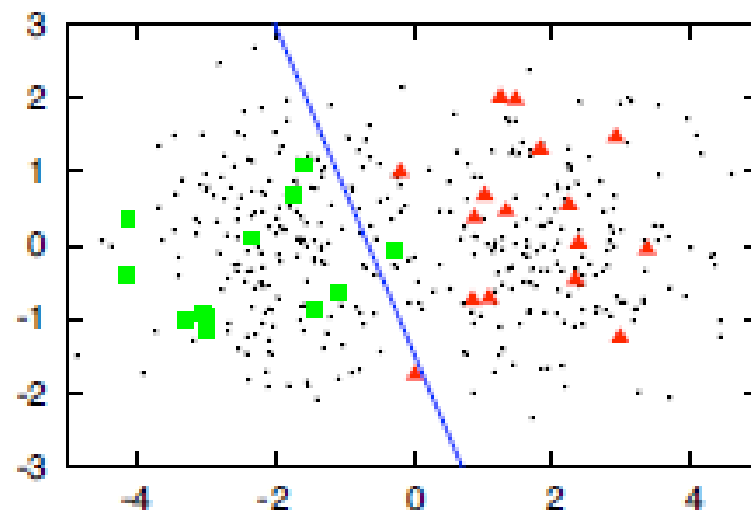
Query Strategies: “Acquisition Functions”

- **Exploration and exploitation:** the choice of examples to label is seen as a dilemma between the exploration and the exploitation over the data space representation.
- **Expected model change:** label points that would most change the current model.
- **Expected error reduction:** label points that would most reduce the model’s generalization error.
- **Uncertainty sampling:** label points for which the current model is least certain as to what the correct output should be.
- **Query by committee:** a variety of models are trained on the current labeled data, and vote on the output for unlabeled data
- **Variance reduction:** label points that would minimize output variance, which is one of the components of error.
- **Conformal predictors:** degree of the similarity within the old examples is used to estimate the confidence in the prediction.

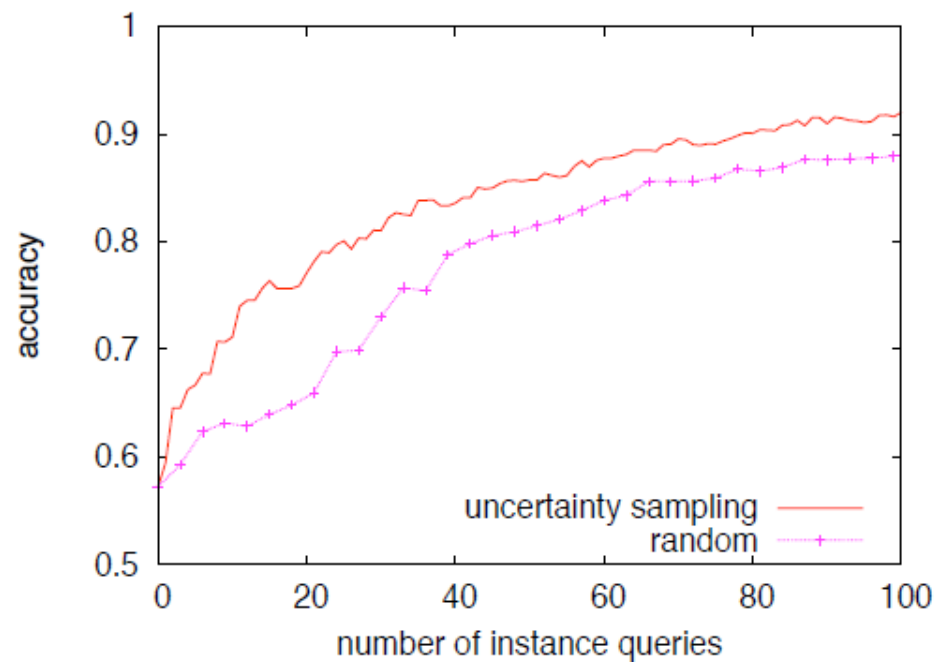
Illustration



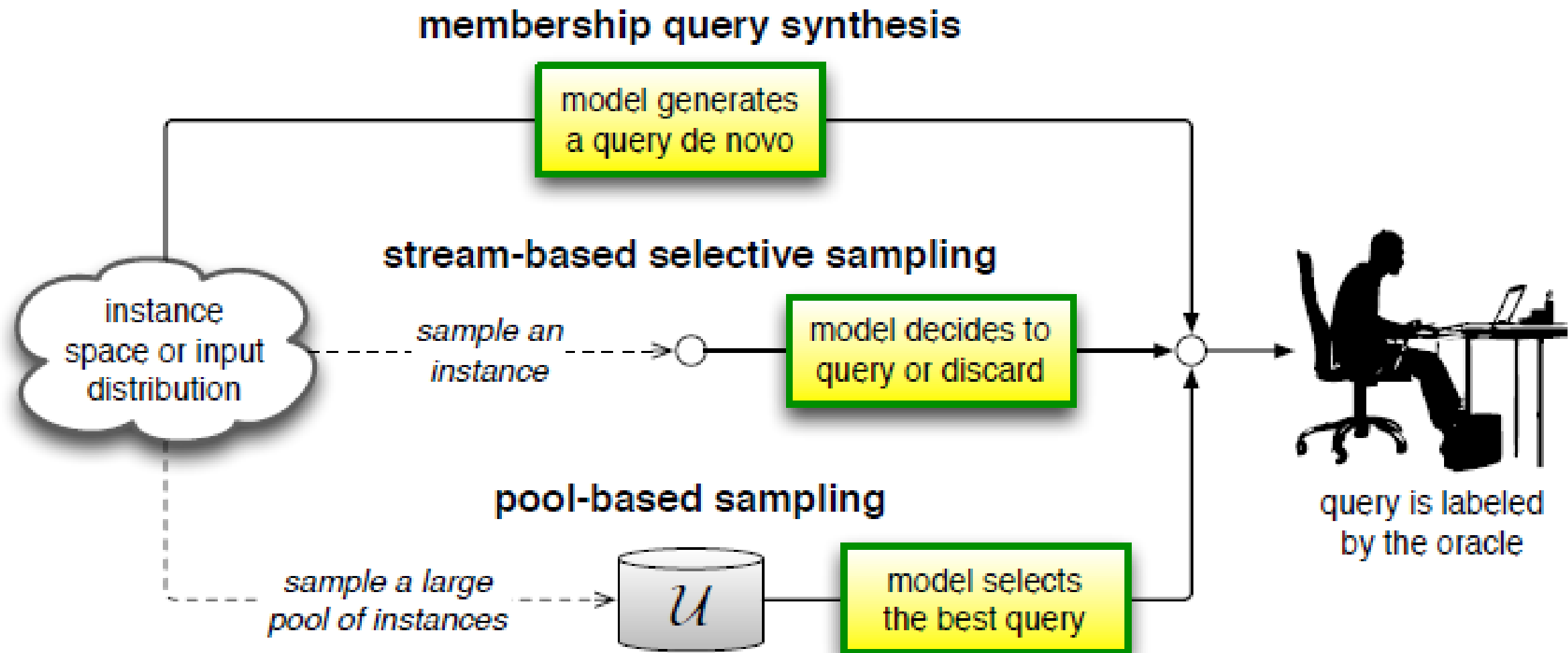
(a)



(c)



Illustration



Membership Query Synthesis

- Learner requests labels for unlabeled instance in input space.
- Query must be synthesized.
- Limitations with human oracles. (digits, text, etc.)
- Impactful for automated scientific discovery where oracle is “robot scientist”.
- Example: Biological experiments. The instance is growth media with a given chemical solution and some mutant. Label is whether mutant thrived in medium.

Stream-Based Selective Sampling

- Unlabeled instances are “free”.
- Sample from the actual distribution (not surrogate) and learner can decide whether to request label.
- Sequential active learning: unlabeled instance is drawn one at a time and learner decides to query or discard.
- Decision to query is based some “informativeness measure” or “query strategy”. There are also “regions of uncertainty”.
- Simply, set a threshold and query everything above it.

Pool-Based Sampling

- Large collections of unlabeled data can be gathered at once.
- Assume a small set of labeled data and a large pool of unlabeled data available.
- Evaluates and ranks the entire collection before selecting the best query.

Query Strategies

- Uncertainty Sampling

$$x_{LC}^* = \operatorname{argmax}_x 1 - P_{\theta}(\hat{y}|x)$$

- Query-By-Committee:

- Committee of models (ensemble learning) with measure of disagreement:

$$x_{VE}^* = \operatorname{argmax}_x - \sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}$$

- Kullback-Leibler divergence

$$x_{KL}^* = \operatorname{argmax}_x \frac{1}{C} \sum_{c=1}^C D(P_{\theta(c)} \| P_C),$$

$$D(P_{\theta(c)} \| P_C) = \sum_i P_{\theta(c)}(y_i|x) \log \frac{P_{\theta(c)}(y_i|x)}{P_C(y_i|x)}$$

Other Query Strategies

- Expected model change: greatest change in current model if we knew its label.
- Expected Error Reduction: how much is generalization error likely to be reduced.
- Variance Reduction: Minimize output variance
 - Key ingredient: Fisher information matrix, the partial derivative of the log-likelihood function w.r.t. model parameters

$$\mathcal{I}(\theta) = N \int_x P(x) \int_y P_\theta(y|x) \frac{\partial^2}{\partial \theta^2} \log P_\theta(y|x),$$

- A-optimality: min trace of inverse IM
- D-optimality: min determinant of inverse IM
- E-optimality: min max eigenvalue of inverse IM

Bayesian Optimization Acquisition Functions

- Probability of Improvement:

- Expected Improvement: