Gaussian Processes for classification

Vamshi Kumar Kurva

Indian Institute of Science
Bangalore

Outline

- Gaussian distribution
- Introduction to GPs
- GP Regression
- GP Classification
- Challenges in the implementation
- How to scale GPs for large Datasets?
- Relation of GPs to Deep NNs
- Deep Gaussian Processes

Gaussian Distribution

- Normal distribution has nice analytical properties
- Let X, Y be jointly Gaussian

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX}, & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} \right)$$

- Closed under marginalisation.

$$X \sim \mathcal{N}(\mu_X, \Sigma_{XX})$$

- Closed under Conditioning

$$X/Y \sim \mathcal{N}(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{XY})$$

Gaussian Processes

- Generalisation of a Gaussian distribution
- Defines distribution over finite random variables which are jointly Gaussian
- In general, random variables in the stochastic process are indexed over time, but here the random variables corresponds to function values over the set of possible input values
- GPs defines distribution over functions.
- GPs are non-parametric in the sense that parameters depends on the number of samples.
- GPs are discriminative, i.e. models the output directly in terms of input $p(y/x)$.

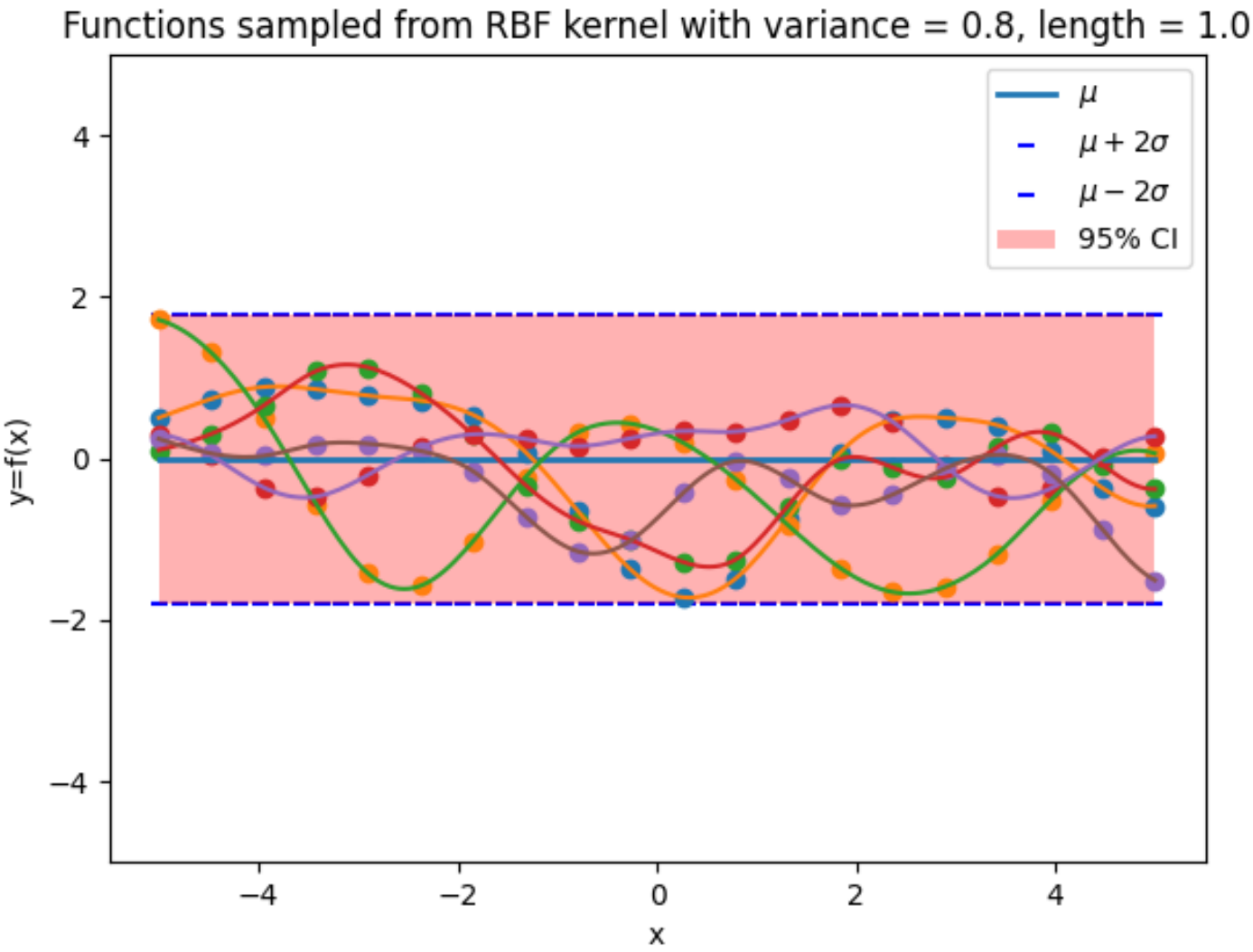
GP Regression

- Assumes a prior over functions
- Set of hypothesis functions can be controlled by the kernel which gives the covariance matrix.
- A commonly used kernel is Squared exponential or Radial Basis Kernel

$$k(x, x') = \sigma^2 \exp\left(- \frac{\|x - x'\|^2}{2l^2} \right)$$

- RBF is a stationary kernel, invariant to translation,i.e depends only on the relative position between the data points.
- As l increases, the correlation between farther points in space increases and the functions become smoother.
- Any other kernel function which is symmetric and positive definite can also be used.

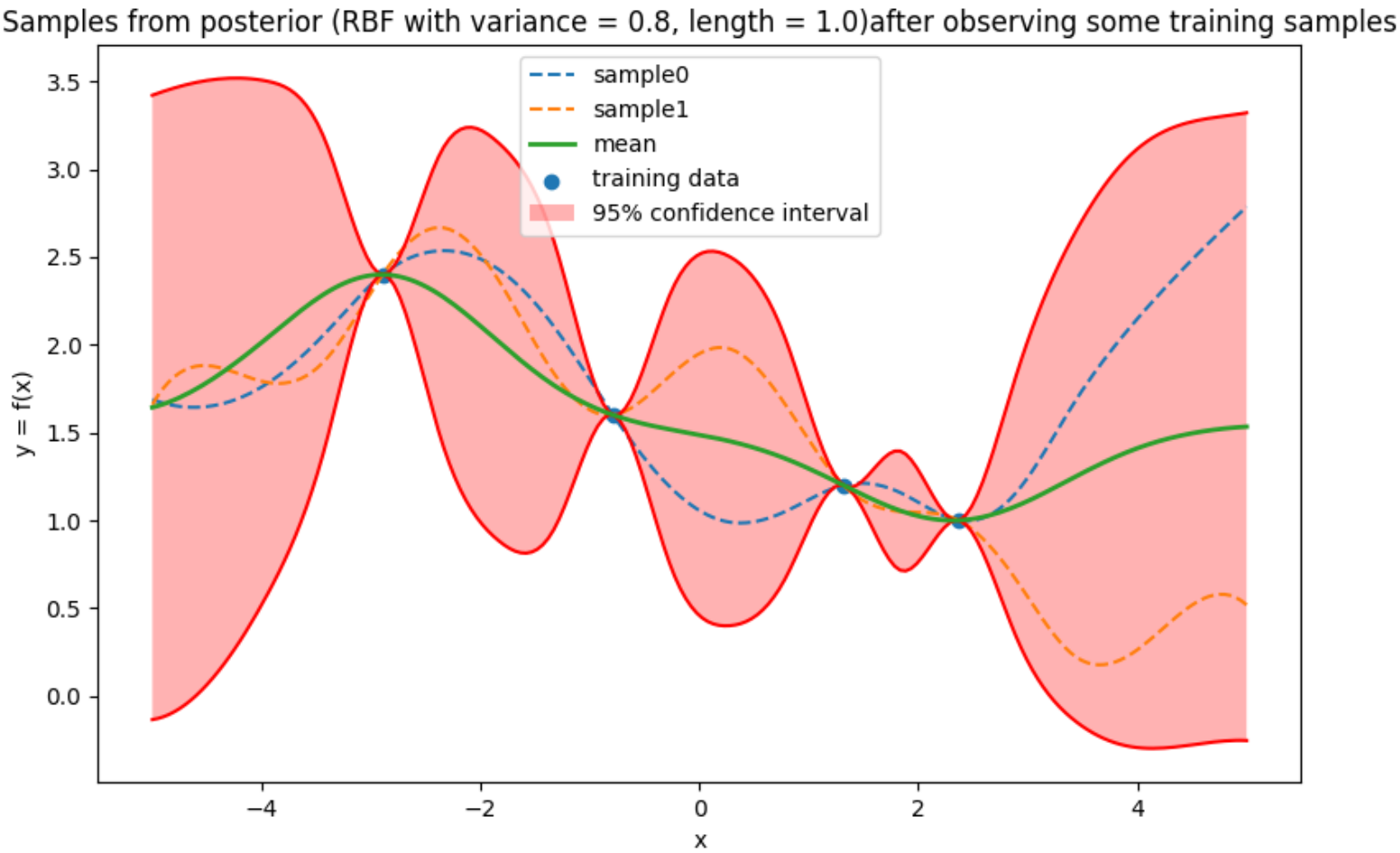
GP prior


Functions sampled from RBF kernel with variance = 0.8, length = 1.0
μ
μ + 2σ
μ − 2σ
95% CI
y=f(x)
x
4
2
0
−2
−4
−4
−2
0
2
4

GP Regression...continued

- Posterior over functions after observing some samples is given by the conditional which is a gaussian.
- This can be thought of as sampling functions from the prior and rejecting the ones that doesn't agree with the observations.
- Data points near the observed data will have less uncertainity in their predictions compared to the ones that are farther away.
- Non parametric i.e. when the test samples are available, we directly make predictions based on the already observed training data. GPs don't learn from the training data itself.

GP posterior



Complexity of GPR

- The posterior involves the calculation of inverse of covariance matrix of training samples Σ_{YY}^{-1}
- The inversion of a matrix is of $\mathcal{O}(N^3)$ complexity
- Doesn't scale well with the number of observations, so we need some approximation methods.

GPs for Classification

- Idea is to generate the latent function values f by GPR, and then apply squashing function (e.g. sigmoid) to limit the values between 0 and 1 to represent the probability.

$$\pi(x) := p(y = +1/x) = \sigma(f(x))$$

- GP prior is placed over latent function values f, and sigmoid is applied to get a prior over $\pi(x)$, i.e. probabilities over probabilities.
- Natural generalisation of linear logistic regression
- Intuitively latent function values capture the logits (just like in logistic regression case).

GPs for classification...continued

- We don't observe the values of f itself (we only observe inputs X and class labels y) and are not interested in f, but π
- Inference is done in two stages. First is to compute the distribution of latent function corresponding to the test samples

$$p(f_*/X, y, x*) = \int \underbrace{p(f_*/X, y, f)}_{\substack{\text{conditional gaussian} \\ \text{from GPR}}} \underbrace{p(f/X, y)}_{\substack{\text{posterior over} \\ \text{latent variable f}}} df \qquad (1)$$

- Next is to produce a probability prediction using f_*

$$\bar{\pi}_* = p(y_* = 1/X, y.x_*) = \int \sigma(f_*) p(f_*/X, y, x*) df_* \qquad (2)$$

...continued

- Posterior over latent function

$$p(f/X,y) = \frac{p(y/f)p(f/X)}{p(y/X)} \tag{3}$$

- $p(f/X)$ - prior over latent function $f \sim \mathcal{N}(0, K_{XX})$
- Likelihood function

$$p(y/f) = \left\{ \begin{array}{ll} \sigma(f) & y = +1 \\ 1 - \sigma(f) = \sigma(-f) & y = -1 \end{array} \right\} = \sigma(yf)$$

- Even though prior is gaussian, since the likelihood is non-gaussian, (3) is non-gaussian and this makes the integral in (1) analytically intractable.
- Similarly (2) is also analytically intractable

Laplace approximation

Is there a way we can approximate the posterior to make the integral in (1) analytically tractable?

- A method to approximate a probability distribution whose normalisation constant is difficult to evaluate.
- In (3) it's difficult to compute the denominator
- The idea is to find a normal distribution with mode at the same point as that of the original distribution

Algorithm 1: Laplace approximation

Result: $q(f/X, y)$

1 $\hat{f} = \arg\max_f p(f/X, y)$

2 $A = -\nabla^2 p(\hat{f}/X, y)$

3 Approximate $p(f/X, y)$ with $q(f/X, y) \sim \mathcal{N}(f; \hat{f}, A^{-1})$

Laplace approximation : In action

- Since log is an increasing function

$$\log p(f/X,y) = \log p(y/f) + \log p(f/X) - \underbrace{\log p(y/x)}_{\text{independent of f}}$$

$$= \log p(y/f) - \frac{1}{2} f^T K^{-1} f - \frac{1}{2} \log \left|K\right| - \frac{n}{2} \log 2\pi$$

$$\nabla \log p(f/X,y) = \nabla \log p(y/f) - K^{-1} f$$

$$\nabla^2 \log p(f/X,y) = \nabla^2 \log p(y/f) - K^{-1} = -(W + K^{-1})$$

- $\nabla \log p(f/X,y) = 0 \implies \hat{f} = K(\nabla \log p(y/\hat{f}))$
- The above equation can't be solved directly, since it is a self-consistent equation, i.e. $\hat{f}$ is expressed in terms of $\hat{f}$ itself.

... continued

- $\nabla^2 \log p(y/f)$ - diagonal, since y_i for each x_i depends only on f_i given latent function f_i
- Also positive definite since $p(y/f) = \sigma(yf)$ is a strictly increasing function
- This means $\nabla^2 \log p(f/X, y)$ is negative definite, which means $\log p(f/X, y)$ is strictly concave and has a unique maxima.
- Hence, the finding the mode can be posed as an optimisation problem and the update equation using Newton's method is given by

$$\begin{aligned} f_{k+1} &= f_k + (K^{-1} + W)^{-1}(\nabla \log p(y/f_k) - K^{-1} f_k) \\ &= (K^{-1} + W)^{-1}(W f_k + \nabla \log p(y/f_k)) \end{aligned}$$

... continued

- The updates corresponding to the well explained data points under the current f_k will be close to zero since W_{ii} and $\partial \log p(y_i/f_{ki})/\partial f_{ki}$ are close to zero.
- Laplace approximation is simple, but may not a be a good approximation to the true shape of the original distribution if the original distribution is multi-modal or has a skewed peak.
- Since the log posterior in our case is strictly concave and has a unique maxima, Laplace approximation may not be so bad.
- Now, the posterior over the latent values corresponding to the test samples x_* is given by

$$p(f_*/X, y, x_*) = \int p(f_*/X, y, f)q(f/X, y)df$$
$$\sim \mathcal{N}(f_*/\mu_*, \Sigma_*)$$

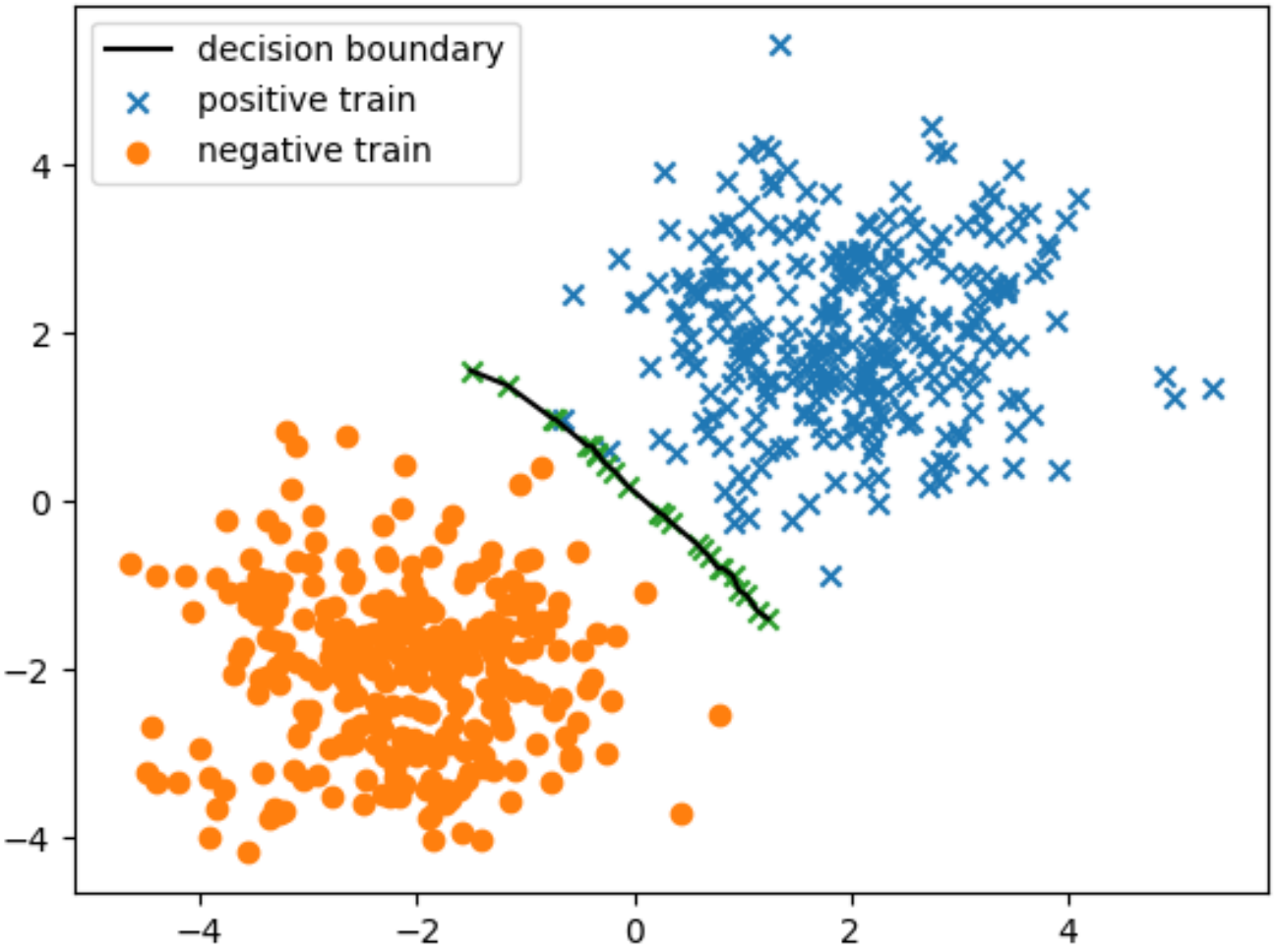
... continued

- $\mu_* = k_*^T K^{-1} \hat{f} = k_*^T \nabla \log p(y/\hat{f})$
- $\Sigma_* = k_{**} - k_*^T K^{-1} \hat{f} + k_*^T K^{-1} (K^{-1} + W)^{-1} K k_*$
- Predictive mean is given by

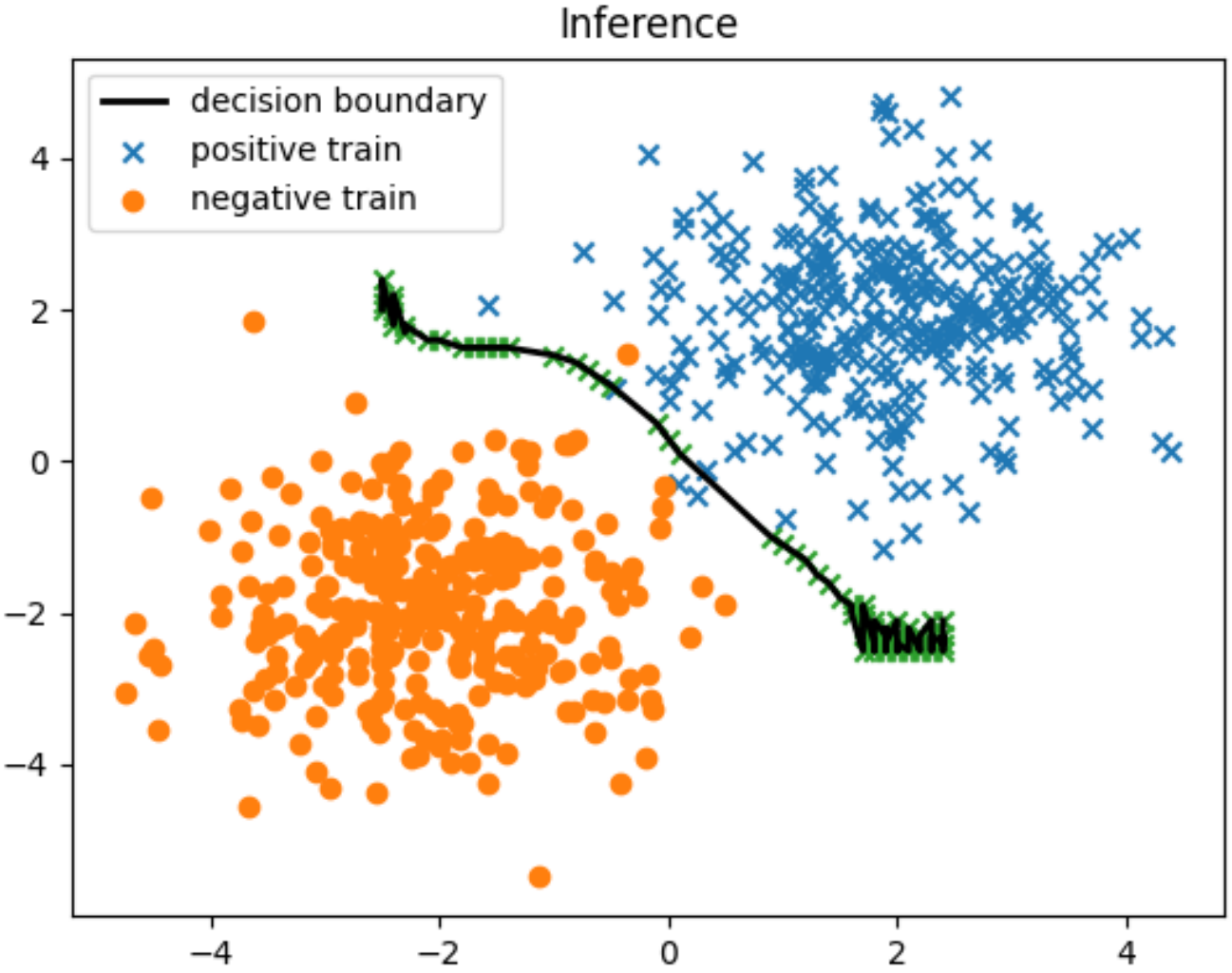
$$\bar{\pi}_* = p(y_* = 1/X, y.x_*) = \int \sigma(f_*) \mathcal{N}(f_*/\mu_*, \Sigma_*)$$
$$= \mathbb{E}_{\hat{f} \sim \mathcal{N}(\mu_*, \Sigma_*)}(\sigma(f_*))$$

- Can be approximated by drawing the samples from the distribution and finding the sample average.

GPC in Action



.. continued


Inference
decision boundary
positive train
negative train
4
2
0
−2
−4
−4
−2
0
2
4

Computational tricks

- Kernel matrix K might have eigen values close to zero and calculating the inverse could be unstable. One idea is to calculate the Cholesky decomposition $K = LL^T$ where L is a lower triangular matrix.

$$\begin{aligned} K^{-1} &= (LL^T)^{-1} \\ &= L^{T^{-1}} L^{-1} \\ &= L^{-1^T} L^{-1} \end{aligned}$$

- Cholesky decomposition is considered to be numerically stable.
- Calculation of posterior over latent values for a given test sample also involve the calculation of $(K^{-1} + W)^{-1}$.

..continued

- Using the matrix inversion lemma, the inverse can be expressed as

$$(K^{-1} + W)^{-1} = K - KW^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} K$$

  where $B = \mathrm{I} + W^{\frac{1}{2}} K W^{\frac{1}{2}}$

- B is well-conditioned since it's eigen values are bounded below by 1. Hence Calculating it's inverse is stable.
- Matrix Inversion Lemma (assuming all the relevant inverses exists)

$$(Z + UWV^T)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^T Z^{-1} U)^{-1} V^T Z^{-1}$$

Scaling methods for Huge Datasets

- Low rank Approximation methods (Nystrom method)
- Sparse Gaussian Processes
- Subset of Regressors
- Bayesian Committee members
- Iterative solution of Linear system

Nystrom Approximation

- Nystrom method approximates the PSD matrix with a low rank matrix.
- We randomly sample $m << n$ points from original data points and assuming that (without loss of generality) the dataset is arranged such that the selected m points come first, the kernel matrix can be written as

$$K = \begin{pmatrix} K_{mm} & K_{m(n-m)} \\ K_{(n-m)m} & K_{(n-m)(n-m)} \end{pmatrix}$$

- Nystrom approximation for K is given by

$$\tilde{K} = K_{nm} K_{mm}^{-1} K_{mn}$$

..continued

- K_{mm} is symmetric and positive definite and hence can be eigen decomposed as $K_{mm} = PDP^T$ where P is orthonormal matrix.
- Nystrom approximation can be written as

$$\begin{aligned}\tilde{K} &= K_{nm}K_{mm}^{-1}K_{mn} \\ &= K_{nm}(PD^{-1}P^T)K_{mn}^T \\ &= (K_{nm}PD^{-\frac{1}{2}})(K_{nm}PD^{-\frac{1}{2}})^T \\ &= Q_{nm}Q_{nm}^T\end{aligned}$$

- Now the inverse can be calculated using the matrix inversion lemma (only involves inverse of an $m \times m$ matrix)

$$(QQ^T + \sigma_n^2 \mathrm{I}_n)^{-1} = \sigma_n^{-2}\mathrm{I}_n - \sigma_n^{-2}Q(\sigma_n^2\mathrm{I}_m + Q^TQ)^{-1}Q^T$$

- Complexity $\mathcal{O}(m^2 n)$

Bayesian Committee Machines

- Introduced to faster GP Regression
- Splits the data into partitions and models each partition using a GP
- Makes a prediction on a test sample using a Bayesian combination of GPs.
- Let $\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_p$ be the p partitions of the dataset $\mathcal{D}$ and f_* be the latent value corresponding to the test sample x_*.

$$p(f_*/\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_p) \propto p(\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_p/f_*)p(f_*)$$

  where $p(f_*)$ is the prior over latent function.
- BCM makes the conditional independent assumption that $\mathcal{D}_i \perp\!\!\!\perp \mathcal{D}_j/f_*$

BCM...continued

- Using conditional independent assumption

$$
\begin{aligned}
p(f_*/\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_p) &\propto p(\mathcal{D}_1/f_*)p(\mathcal{D}_2/f_*)....p(\mathcal{D}_p/f_*)p(f_*) \\
&\propto \frac{p(\mathcal{D}_1, f_*)}{p(f_*)}\frac{p(\mathcal{D}_2, f_*)}{p(f_*)}....\frac{p(\mathcal{D}_p, f_*)}{p(f_*)}p(f_*) \\
&\propto \frac{p(f_*/\mathcal{D}_1)p(f_*/\mathcal{D}_2)....p(f_*/\mathcal{D}_p)}{p^{p-1}(f_*)} \\
&= c\frac{\prod_{i=1}^{p} p(f_*/\mathcal{D}_i)}{p^{p-1}(f_*)}
\end{aligned}
$$

  where c is a normalisation constant.

- All the terms in the above equation are Gaussian distributions and hence it is easy to find the predictive mean and variance of f_*

BCM...continued

- Predictive mean

$$\mathbb{E}(f_*/D) = [cov(f_*/\mathcal{D})] \sum_{i=1}^{p} [cov(f_*/\mathcal{D}_i)]^{-1} \mathbb{E}(f_*/\mathcal{D}_i)$$

- Predictive variance

$$[cov(f_*/\mathcal{D})]^{-1} = -(p-1)K_{**}^{-1} + \sum_{i=1}^{p} [cov(f_*/\mathcal{D}_i)]^{-1}$$

K_{**} is the covariance matrix evaluated at test points.

- Dataset $\mathcal{D}$ can be partitioned in any ways. If m is the partition size we want, we will have $p = n/m$ partitions. We can use the $p-$mean clustering and use the clusters as partitions to get improved performance.
- Complexity $\mathcal{O}(pm^3) = \mathcal{O}(m^2 n)$

Iterative solution of linear system

- Idea is to find the solution to the system $(K + \sigma_n^2 I)v = y$ iteratively using Conugate Gradient method
- To get the correct solution it will take atmost n steps. The idea is to stop the process after certain iterations $k < n$ to get an approximated solution.
- Time complexity $\mathcal{O}(n^2 k)$

Deep Kernel GPs

- GPs are powerful non-parametric probabilistic methods with the kernel function at it's core.
- Model performance depends on the selection of the kernel function and standard kernels are limited by their design choices.
- More expressive kernels which can find the hidden structure in data automatically can be developed by using deep learning.
- GPs with deep kernels are referred to as Deep Kernel GPs.
- Deep Kernel GPs have the flexibility of non-parametric kernel methods combined with the structural properties of Deep learning architectures.

Deep Kernel Learning

- Let x_i, x_j are inputs, k is a base kernel function parametrized by θ. DKL involves transforming the inputs through non-linearity before applying the kernel.

$$k(x_i, x_j/\theta) \rightarrow k(g(x_i, w), g(x_j, w)|\theta, w)$$

  where $g(x, w)$ is the non-linear transformation given a deep network architecture.

- Deep Kernel Hyper parameters $\{w, \theta\}$ i.e. network parameters w and base kernel hyper parameters θ can be jointly learnt by maximizing the log likelihood $\mathcal{L}$ of the training data under GP.

DKL...continued

- $f/X \sim \mathcal{N}(0, K)$ and $y/f \sim \mathcal{N}(0, \sigma_n^2 \mathrm{I})$
- $p(y/X) = \int p(y/f)p(f/X)df$
- Log likelihood $\log p(y/X)$ is given by

$$\mathcal{L} = -\frac{1}{2} y^T (K + \sigma_n^2 \mathrm{I})^{-1} y - \frac{1}{2} \log(K + \sigma_n^2 \mathrm{I}) - \frac{n}{2} \log 2\pi$$

- Partial derivatives w.r.to θ

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial K} \frac{\partial K}{\partial \theta}$$

- Partial derivatives w.r.to w

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial K} \frac{\partial K}{\partial g(x, w)} \frac{\partial g(x, w)}{\partial w}$$

...continued

- Absorbing the noise variance $\sigma_n^2 \mathbf{I}$ into the co variance matrix and treating it as a kernel hyper parameter θ

$$\frac{\partial L}{\partial K_\gamma} = \frac{1}{2}(K_\gamma^{-1} y y^T K_\gamma^{-1} - K_\gamma^{-1})$$

- Time complexity to invert the covariance matrix is still $\mathcal{O}(n^3)$
- However, it can be used when the no of samples are very low, especially in low data regime.
- GPs predictive capabilities combined with the Deep kernels were proven effective in Few Shot Learning.

Questions?