Fast large scale Gaussian process regression using approximate matrix-vector products

Vikas Chandrakant Raykar and Ramani Duraiswami {vikas,ramani}@umiacs.umd.edu

> Department of Computer Science University of Maryland, CollegePark

> > March 21, 2007

Regression



- Training data $\mathfrak{D} = \{x_i \in \mathbf{R}^d, y_i \in \mathbf{R}\}_{i=1}^N$.
- Predict y_* for a new x_* .
- Learn $f \in \mathcal{F} : \mathbf{R}^d \to \mathbf{R}$ such that $y_* = f(x_*)$.

Regression



- Training data $\mathfrak{D} = \{x_i \in \mathbf{R}^d, y_i \in \mathbf{R}\}_{i=1}^N$.
- Predict y_* for a new x_* .
- Learn $f \in \mathcal{F} : \mathbf{R}^d \to \mathbf{R}$ such that $y_* = f(x_*)$.
- Non-parametric No parametric assumptions on f.
- Bayesian Predictive distribution for y_{*}

Gaussian process regression¹

- A popular Bayesian non-linear non-parametric approach.
- The regression function is represented by an ensemble of functions, on which we place a Gaussian prior.
- This prior is updated in the light of the training data.
- As a result we obtain predictive distributions.

¹C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. The MIT Press, 2006 🕨 💈 🔊 🔍

 $\frac{\mathsf{Model}}{y = f(x) + \varepsilon}$

3

イロト イポト イヨト イヨト

Model $y = f(x) + \varepsilon$ • ε is $\mathcal{N}(0, \sigma^2)$.

3

Model

- $y = f(x) + \varepsilon$
 - ε is $\mathcal{N}(0, \sigma^2)$.
 - f is a zero-mean Gaussian process with covariance function K(x, x').
 - Most common covariance function is the Gaussian.

Model

- $y = f(x) + \varepsilon$
 - ε is $\mathcal{N}(0, \sigma^2)$.
 - f is a zero-mean Gaussian process with covariance function K(x, x').
 - Most common covariance function is the Gaussian.

Training data

$$\mathfrak{D} = \{x_i \in \mathbf{R}^d, y_i \in \mathbf{R}\}_{i=1}^N$$

• • = • •

Model

- $y = f(x) + \varepsilon$
 - ε is $\mathcal{N}(0, \sigma^2)$.
 - f is a zero-mean Gaussian process with covariance function K(x, x').
 - Most common covariance function is the Gaussian.

Training data

$$\mathfrak{D} = \{x_i \in \mathbf{R}^d, y_i \in \mathbf{R}\}_{i=1}^N$$

Infer the posterior

Given the training data \mathfrak{D} and a new input x_* our task is to compute the posterior $p(f_*|x_*,\mathfrak{D})$.

< 回 ト < 三 ト < 三 ト

Solution

- The posterior is a Gaussian.
- The mean is used as the prediction.
- The variance is the uncertainty associated with the prediction.

Solution

- The posterior is a Gaussian.
- The mean is used as the prediction.
- The variance is the uncertainty associated with the prediction.



Gaussian Process regression

Notation

- Training data $\mathfrak{D} = \{x_i \in \mathbf{R}^d, y_i \in \mathbf{R}\}_{i=1}^N$
- **K** is $N \times N$ covariance matrix where $[\mathbf{K}]_{ij} = K(x_i, x_j)$
- $\mathbf{y} = [y_1, \ldots, y_N]^T$
- $\mathbf{k}(x_*) = [K(x_*, x_1), \dots, K(x_*, x_N)]^T$

A B A A B

Gaussian Process regression

Notation

- Training data $\mathfrak{D} = \{x_i \in \mathbf{R}^d, y_i \in \mathbf{R}\}_{i=1}^N$
- **K** is $N \times N$ covariance matrix where $[\mathbf{K}]_{ij} = K(x_i, x_j)$
- $\mathbf{y} = [y_1, \ldots, y_N]^T$

•
$$\mathbf{k}(x_*) = [K(x_*, x_1), \dots, K(x_*, x_N)]^T$$

The posterior $p(f_*|x_*, \mathfrak{D})$ is a Gaussian

- Mean $\mathbf{k}(x_*)^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$
- Variance $K(x_*, x_*) \mathbf{k}(x_*)^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}(x_*)$

A D A D A D A

Gaussian Process regression

Notation

- Training data $\mathfrak{D} = \{x_i \in \mathbf{R}^d, y_i \in \mathbf{R}\}_{i=1}^N$
- **K** is $N \times N$ covariance matrix where $[\mathbf{K}]_{ij} = K(x_i, x_j)$
- $\mathbf{y} = [y_1, \ldots, y_N]^T$

•
$$\mathbf{k}(x_*) = [K(x_*, x_1), \dots, K(x_*, x_N)]^T$$

The posterior $p(f_*|x_*,\mathfrak{D})$ is a Gaussian

- Mean $\mathbf{k}(x_*)^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$
- Variance $K(x_*, x_*) \mathbf{k}(x_*)^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}(x_*)$

• Training
$$\xi = (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

• Prediction $\mathbf{k}(x_*)^T \xi = \sum_{i=1}^N \xi_i K(x_*, x_i)$

3

(本部) (本語) (本語)

Computational cost

Prediction

$$f(x_*) = \sum_{i=1}^N \xi_i K(x_*, x_i)$$

- $\mathcal{O}(N)$ cost to predict at a new point x_* .
- $\mathcal{O}(MN)$ cost to predict at M such points.

-∢ ∃ ▶

Computational cost

$\xi = (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$ or solve for $(\mathbf{K} + \sigma^2 \mathbf{I}) \xi = \mathbf{y}$

- Direct computation of the linear system solution requires $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage.
- Impractical even for problems of moderate size (typically a few thousands).

• • = • • = •

Computational cost

$$\xi = (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$
 or solve for $(\mathbf{K} + \sigma^2 \mathbf{I}) \xi = \mathbf{y}$

- Direct computation of the linear system solution requires $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage.
- Impractical even for problems of moderate size (typically a few thousands).

Example

1D regression with N = 25,600 using Cholesky decomposition Takes around 10 hours, assuming you have enough RAM.

A D A D A D A

A First Step to Speed Up: Iterative methods ³ Conjugate gradient (CG)

 $(\mathbf{K} + \lambda \mathbf{I})\xi = \mathbf{y}.$

- The iterative method generates a sequence of approximate solutions ξ_k at each step which converge to the true solution ξ.
- Can use the conjugate-gradient ² method since (K + λI) is symmetric and positive definite.

²C. T. Kelley. Iterative Methods for Linear and Non-linear Equations. SIAM, 1995.

³D. MacKay and M. N. Gibbs. Efficient implementation of Gaussian processes. unpublished, 1997: 🕨 🗧 🔊 🤉 🔿

A First Step to Speed Up: Iterative methods ³ Conjugate gradient (CG)

 $(\mathbf{K} + \lambda \mathbf{I})\xi = \mathbf{y}.$

- The iterative method generates a sequence of approximate solutions ξ_k at each step which converge to the true solution ξ.
- Can use the conjugate-gradient ² method since (K + λI) is symmetric and positive definite.
- $\bullet\,$ Given a tolerance 0 $<\eta<1$ a practical CG scheme iterates till

$$\|\mathbf{y} - (\mathbf{K} + \lambda \mathbf{I})\xi_k\|_2 \le \eta \|\mathbf{y} - (\mathbf{K} + \lambda \mathbf{I})\xi_0\|_2.$$

Vikas C. Raykar (Univ. of Maryland)

²C. T. Kelley. Iterative Methods for Linear and Non-linear Equations. SIAM, 1995.

³D. MacKay and M. N. Gibbs. Efficient implementation of Gaussian processes. unpublished, 1997: 🕨 🗧 🔊 🤉 🔅

A First Step to Speed Up: Iterative methods ³ Conjugate gradient (CG)

 $(\mathbf{K} + \lambda \mathbf{I})\xi = \mathbf{y}.$

- The iterative method generates a sequence of approximate solutions ξ_k at each step which converge to the true solution ξ.
- Can use the conjugate-gradient ² method since (K + λI) is symmetric and positive definite.
- $\bullet\,$ Given a tolerance 0 $<\eta<1$ a practical CG scheme iterates till

$$\|\mathbf{y} - (\mathbf{K} + \lambda \mathbf{I})\xi_k\|_2 \le \eta \|\mathbf{y} - (\mathbf{K} + \lambda \mathbf{I})\xi_0\|_2.$$

• An estimate for the number of iterations required is

$$k \geq \ln\left[rac{2\sqrt{\kappa}}{\eta}
ight]/2\ln\left[rac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}
ight].$$

where $\kappa = \lambda_{max}/\lambda_{min}$ is the spectral condition number.

 2 C. T. Kelley. Iterative Methods for Linear and Non-linear Equations. SIAM, 1995.

³D. MacKay and M. N. Gibbs. Efficient implementation of Gaussian processes. unpublished, 1997: 🕨 🤙 🔊 🔍

Computational cost of conjugate-gradient

- Requires one matrix-vector multiplication and 5N flops per iteration.
- Four vectors of length N are required for storage.
- Hence computational cost now reduces to $\mathcal{O}(kN^2)$.

Computational cost of conjugate-gradient

- Requires one matrix-vector multiplication and 5N flops per iteration.
- Four vectors of length N are required for storage.
- Hence computational cost now reduces to $\mathcal{O}(kN^2)$.

Example

1D regression with N = 25,600 with $CG(\eta = 10^{-3})$ Takes around 17 minutes (compare to 10 hours).

• The core computational step in each conjugate-gradient iteration is the multiplication of the matrix **K** with a vector, say **q**.

 $(\mathbf{Kq})_j = \sum_{i=1}^N q_i k(x_i, x_j) - \mathcal{O}(N^2)$ cost.

• The core computational step in each conjugate-gradient iteration is the multiplication of the matrix **K** with a vector, say **q**.

$$(\mathbf{Kq})_j = \sum_{i=1}^N q_i k(x_i, x_j) - \mathcal{O}(N^2)$$
 cost.



- 本間 と えき と えき とうき

- Fast approximate algorithms have been proposed which can compute the same in $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ time.
 - Fast Gauss Transform (FGT)
 - Improved Fast Gauss Transform (IFGT)
 - Dual-tree methods

- Fast approximate algorithms have been proposed which can compute the same in $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ time.
 - Fast Gauss Transform (FGT)
 - Improved Fast Gauss Transform (IFGT)
 - Dual-tree methods
- These algorithms compute the sum to a user specified ε precision−|computed value - actual value| ≤ ε.

- Fast approximate algorithms have been proposed which can compute the same in $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ time.
 - Fast Gauss Transform (FGT)
 - Improved Fast Gauss Transform (IFGT)
 - Dual-tree methods
- These algorithms compute the sum to a user specified ε precision–|computed value - actual value| ≤ ε.

Example

1D regression with N = 25,600 with $CG(\eta = 10^{-3})$ and $IFGT(\epsilon = 1e^{-6})$ Takes around 3 secs. (compare to 10 hours[direct] or 17 minutes[CG]).

< 回 ト < 三 ト < 三 ト

Fast matrix vector product algorithms

Fast Gauss Transform(FGT)

- Gaussian kernel.
- Based on Hermite and Taylor expansion of the Gaussian kernel.
- Special case of the fast multipole methods used in computational physics.
- Suitable for $d \leq 3$.
- L. Greengard and J. Strain. The fast Gauss transform. SIAM Journal of Scientific and Statistical Computing, 12(1):79–94, 1991.
- L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. J. of Comp. Physics, 73(2):325–348, 1987.
- Fortran code for the FGT is available at http://math.berkeley.edu/~strain/Codes/index.html.

A D A D A D A

Fast matrix vector product algorithms

Improved Fast Gauss Transform

Improved Fast Gauss Transform(IFGT)

- Gaussian kernel.
- Based on a single Taylor series expansion.
- Scales well with dimensions $(d \le 10)$.

Fast Improved Gauss Transform with *kd*-tree(FIGTree)

- New version which uses kd-trees for neighbor searching.
- Works well for small and large bandwidths.
- The Improved Fast Gauss Transform with applications to machine learning Vikas C. Raykar and Ramani Duraiswami, To appear in Large Scale Kernel Machines L. Bottou, O. Chapelle, D. Decoste, and J. Weston (Eds), MIT Press 2006.
- Fast computation of sums of Gaussians in high dimensions. Vikas C. Raykar, C. Yang, R. Duraiswami, and N. Gumerov, CS-TR-4767, Department of computer science, University of Maryland, CollegePark.
- Efficient Kernel Machines Using the Improved Fast Gauss Transform. Changjiang Yang, Ramani Duraiswami, and Larry Davis, In Advances in Neural Information Processing Systems, Volume 17, pages 1561-1568, 2005.
- C++ source code with MATLAB bindings available under LGPL at http://www.umiacs.umd.edu/~vikas/Software/software.html.

Fast matrix vector product algorithms

Dual tree methods

Dual tree methods

- Works for any kernel.
- Based on kd-trees or ball trees.
- Works well for small bandwidths.
- A. G. Gray and A. W. Moore. Nonparametric density estimation: Toward computational tractability. In SIAM International conference on Data Mining, 2003.
- Y. Shen, A. Ng, and M. Seeger. Fast Gaussian process regression using KD-trees. In Y. Weiss, B. Scholkopf, and J. Platt, editors, Advances in Neural Information Processing Systems 18. MIT Press, Cambridge, MA, 2006
- The C++ code with MATLAB bindings for the dual-tree algorithms can be downloaded from the website http://www.cs.ubc.ca/~awll/nbody_methods.html.

(3)

Comparison of fast methods

Dimension d and bandwidth h

	Small dimensions $d \leq 3$	Moderate dimensions $3 < d < 10$	Large dimensions $d \ge 10$
Small bandwidth $h\lessapprox 0.1$	FIGTree, Dual tree	FIGTree, Dual tree	FIGTree, Dual tree
Moderate bandwidth $0.1 \lessapprox h \lessapprox 0.5 \sqrt{d}$	FIGTree, FGT	FIGTree	Direct
Large bandwidth $h \gtrsim 0.5\sqrt{d}$	FIGTree, FGT	FIGTree	FIGTree

3

< 🗇 🕨 <

-

Two parameters

- For conjugate-gradient we specify the convergence tolerance η (Typically $\eta = 10^{-3}$).
- For the Fast matrix product we specify an accuracy parameter ϵ (Typically $\epsilon = 10^{-6}$).

⁴V. Simoncini and D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. SIAM J. Sci. Comput., 25(2):454–477, 2004. ← □ ▶ ← (□)

Two parameters

- For conjugate-gradient we specify the convergence tolerance η (Typically $\eta = 10^{-3}$).
- For the Fast matrix product we specify an accuracy parameter ϵ (Typically $\epsilon = 10^{-6}$).
- Coarser the accuracy ϵ Faster is the algorithm.

⁴V. Simoncini and D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. SIAM J. Sci. Comput., 25(2):454–477, 2004.

Two parameters

- For conjugate-gradient we specify the convergence tolerance η (Typically $\eta = 10^{-3}$).
- For the Fast matrix product we specify an accuracy parameter ϵ (Typically $\epsilon = 10^{-6}$).
- Coarser the accuracy ϵ Faster is the algorithm.
- How does this affect the convergence of CG ?

- Two parameters
 - For conjugate-gradient we specify the convergence tolerance η (Typically $\eta = 10^{-3}$).
 - For the Fast matrix product we specify an accuracy parameter ϵ (Typically $\epsilon = 10^{-6}$).
- Coarser the accuracy ϵ Faster is the algorithm.
- How does this affect the convergence of CG ?
- Can ϵ change at every iteration ?

⁴V. Simoncini and D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. SIAM J. Sci. Comput., 25(2):454–477, 2004. ← □ → ← (□) → (□) →

- Two parameters
 - For conjugate-gradient we specify the convergence tolerance η (Typically $\eta = 10^{-3}$).
 - For the Fast matrix product we specify an accuracy parameter ϵ (Typically $\epsilon = 10^{-6}$).
- Coarser the accuracy ϵ Faster is the algorithm.
- How does this affect the convergence of CG ?
- Can ϵ change at every iteration ?
- Use some recent results in the theory of inexact Krylov subspace methods ⁴.

- At the kth iteration of the CG the approximate MVP can be written as (A + E_k)v_k instead of Av_k.
- E_k is an error matrix.
- Question: How large can $||E_k||$ be to guarantee convergence?

- At the k^{th} iteration of the CG the approximate MVP can be written as $(A + E_k)v_k$ instead of Av_k .
- E_k is an error matrix.
- Question: How large can $||E_k||$ be to guarantee convergence?
- Let $r_k = ||Ax_k b||$ be the residual at the end of the k^{th} iteration.
- Let r
 _k be the corresponding residual when an approximate matrix-vector product is used.

- At the k^{th} iteration of the CG the approximate MVP can be written as $(A + E_k)v_k$ instead of Av_k .
- E_k is an error matrix.
- Question: How large can $||E_k||$ be to guarantee convergence?
- Let $r_k = ||Ax_k b||$ be the residual at the end of the k^{th} iteration.
- If at every iteration

$$\|E_k\| \leq I_m \frac{1}{\|\tilde{r}_{k-1}\|} \delta,$$

then after k iterations $\|\tilde{r}_k - r_k\| \leq \delta$.

- At the k^{th} iteration of the CG the approximate MVP can be written as $(A + E_k)v_k$ instead of Av_k .
- E_k is an error matrix.
- Question: How large can $||E_k||$ be to guarantee convergence?
- Let $r_k = ||Ax_k b||$ be the residual at the end of the k^{th} iteration.
- If at every iteration

$$\|E_k\| \leq I_m \frac{1}{\|\tilde{r}_{k-1}\|} \delta,$$

then after k iterations $\|\tilde{r}_k - r_k\| \leq \delta$.

Matrix-vector product may be performed in an increasingly inexact manner as the iteration progresses and still allow convergence to the solution.

Strategy to choose ϵ ?

Given η and δ

$$\epsilon_k \leq rac{\delta}{N} rac{\|\mathbf{y} - \widetilde{\mathbf{K}}\xi_0\|}{\|\widetilde{r}_{k-1}\|}.$$

This guarantees that

$$\frac{\|\mathbf{y} - \widetilde{\mathbf{K}}\xi_k\|_2}{\|\mathbf{y} - \widetilde{\mathbf{K}}\xi_0\|_2} \le \eta + \delta.$$



Vikas C. Raykar (Univ. of Maryland)

March 21, 2007 19 / 39

3

Experiments

Datasets

- **robotarm** N = 10,000 and d = 2.
- **abalone** N = 4,177 and d = 7.

Evaluation procedure

- 10-fold cross validation.
- standardized mean squared error (SMSE).
- Squared exponential covariance function.

$$\mathcal{K}(x, x') = \sigma_f^2 \exp\left(-\sum_{k=1}^d \frac{(x_k - x'_k)^2}{h_k^2}\right)$$

- Hyperparameters ([h₁,..., h_d, σ_f, σ]) were selected by optimizing the marginal likelihood on the subset using the direct method.
- Same hyperparmeters used for all methods.

Experiments

Previous approaches consider a subset of size m chosen from N training examples.

Methods compared-Training time

- Subset of datapoints (SD)– $\mathcal{O}(m^3)$.
- Subset of regressors/projected process (SR and PP)– $\mathcal{O}(m^2N)$.
- Conjugate gradient + IFGT algorithm– $\mathcal{O}(km)$.
- Conjugate gradient + dual-tree algorithm-O(km).
- See Chapter 8 of C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. The MIT Press, 2006.
- Code can be downloaded from http://www.gaussianprocess.org/gpml/code/matlab/doc/.

(日) (同) (日) (日)

Prediction time

robotarm N = 10,000 and d = 2.



Vikas C. Raykar (Univ. of Maryland)

March 21, 2007 22 / 39

Prediction time

abalone N = 4,177 and d = 7.



Vikas C. Raykar (Univ. of Maryland)

Training time robotarm N = 10,000 and d = 2.



Vikas C. Raykar (Univ. of Maryland)

March 21, 2007 24 / 39

Training time abalone N = 4,177 and d = 7.

abalone 10^{2} Training time (secs) 1 0 10⁰ -O-SD 10⁻¹ SR and PP ---CG+IFGT -x- CG+kd-tree 256 512 1024 2048 3759 m

э

Image: A math a math

SMSE

robotarm N = 10,000 and d = 2.



Vikas C. Raykar (Univ. of Maryland)

March 21, 2007 26 / 39

SMSE

abalone N = 4,177 and d = 7.



Vikas C. Raykar (Univ. of Maryland)

March 21, 2007 27 / 39

Application: Implicit surface fitting

Given a point cloud data find a implicit surface representation.



Implicit surface fitting

As a regression problem

- Point cloud data: A set of N points $\{x_i \in \mathbb{R}^d\}_{i=1}^N (d = 2, 3)$.
- Find $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(x_i) = 0$, for $i = 1, \dots, N$.
- Avoid the trivial solution f(x) = 0 add additional constraints *i.e.*, points where the function f is not zero.
- Off-surface points $f(x_i) = d_i \neq 0$, for i = N + 1, ...

2D Example Off-surface points



э.

▲ 同 ▶ ▲ 目

2D Example

Fitted function via Gaussian Process Rregression



A 🖓

2D Example

Zero level set of the implicit surface



2D Example

two sigma uncertainty surfaces



Implicit surface fitting

As Gaussian process regression

- One of the major bottlenecks for most implicit surface methods is their prohibitive computational complexity.
- Most approaches scales as $\mathcal{O}(N^3)$ during training.
- Using the proposed approach we can handle point clouds containing millions of points.
- Our training times are comparable with the fastest method ^a.

A D A D A D A

^a J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In ACM SIGGRAPH 2001, pages 67–76, Los Angeles, CA, 2001.



3D example



3

<ロ> (日) (日) (日) (日) (日)

Variance at the mesh

Variance is more where there are few data points



• Fast GPR using the conjugate-gradient method coupled with IFGT.

- Fast GPR using the conjugate-gradient method coupled with IFGT.
- Accuracy parameter ϵ can increase as iteration progresses.

- Fast GPR using the conjugate-gradient method coupled with IFGT.
- Accuracy parameter ϵ can increase as iteration progresses.
- Practical for small dimensional datasets ($d \le 10$).

- Fast GPR using the conjugate-gradient method coupled with IFGT.
- Accuracy parameter ϵ can increase as iteration progresses.
- Practical for small dimensional datasets ($d \le 10$).
- Unlike methods which rely on choosing a subset of the dataset we use all the available points and still achieve $\mathcal{O}(N)$ complexity.
 - Other approaches Exact inference in an approximate model.
 - Our approach Approximate inference in an exact model.

- Fast GPR using the conjugate-gradient method coupled with IFGT.
- Accuracy parameter ϵ can increase as iteration progresses.
- Practical for small dimensional datasets ($d \le 10$).
- Unlike methods which rely on choosing a subset of the dataset we use all the available points and still achieve $\mathcal{O}(N)$ complexity.
 - Other approaches Exact inference in an approximate model.
 - Our approach Approximate inference in an exact model.
- Applied to implicit surface fitting.

- Fast GPR using the conjugate-gradient method coupled with IFGT.
- Accuracy parameter ϵ can increase as iteration progresses.
- Practical for small dimensional datasets ($d \le 10$).
- Unlike methods which rely on choosing a subset of the dataset we use all the available points and still achieve $\mathcal{O}(N)$ complexity.
 - Other approaches Exact inference in an approximate model.
 - Our approach Approximate inference in an exact model.
- Applied to implicit surface fitting.

Source code for the IFGT available under LGPL. http://www.umiacs.umd.edu/~vikas/Software/software.html

Check out our other paper at AISTATS. A fast algorithm for learning large scale preference relations.

Graduating this semester and am looking for a job.