GPUs, linear algebra, and efficient computing for Gaussian process models

Colin Rundel

Duke University

August 6, 2013

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood?

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood? - Invert Σ

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood? - Invert Σ - $\mathcal{O}\left(n^3\right)$!

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood? - Invert Σ - $\mathcal{O}\left(n^3\right)$!
- Drawing a sample?

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood? - Invert Σ - $\mathcal{O}\left(n^3\right)$!

- Drawing a sample? - Choleksy of Σ

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood? - Invert Σ - $\mathcal{O}\left(n^3\right)$!
- Drawing a sample? - Choleksy of Σ - $\mathcal{O}\left(n^3\right)$!

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood? - Invert Σ - $\mathcal{O}\left(n^3\right)$!
- Drawing a sample? - Choleksy of Σ - $\mathcal{O}\left(n^3\right)$!
- Fitting a model?

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood? - Invert Σ - $\mathcal{O}\left(n^3\right)$!
- Drawing a sample? - Choleksy of Σ - $\mathcal{O}\left(n^3\right)$!
- Fitting a model? - Update elements of Σ

The problem with GPs ...

Unless you are lucky (or clever), Gaussian process models are difficult to scale to large problems.

Why?

- MVN Density:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Evaluating a likelihood? - Invert Σ - $\mathcal{O}\left(n^3\right)$!
- Drawing a sample? - Choleksy of Σ - $\mathcal{O}\left(n^3\right)$!
- Fitting a model? - Update elements of Σ - $\mathcal{O}\left(n^2\right)$

What this talk is and isn't ...

I do not have any clever solutions for these problems, but there are some very smart people out there who are working on them.

What this talk is and isn't ...

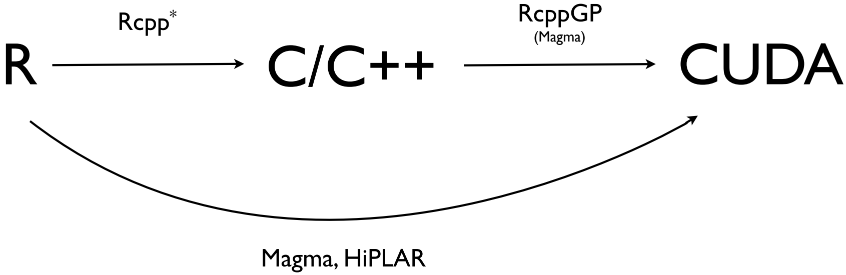
I do not have any clever solutions for these problems, but there are some very smart people out there who are working on them.

What I do have to offer are tools to help make existing models a faster.

R package, RcppGP:

- Low level brute force tools to improve performance (using existing GPU libraries)
- High level tools for unified specification of covariance models
- Painless integration with existing code
- Simplify the development of new models / code

Where does RcppGP fit?



Benchmarking Specs & System

System specs:

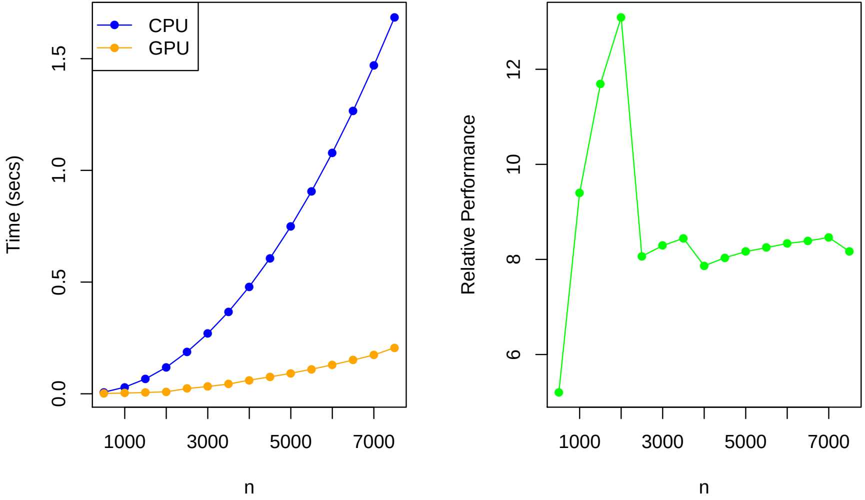
- Intel i5-2500K - 4 cores, 3.3 GHz, 16 GB
- GeForce GTX 460 - 336 cores, 1.44 GHz, 1024 MB

Software specs:

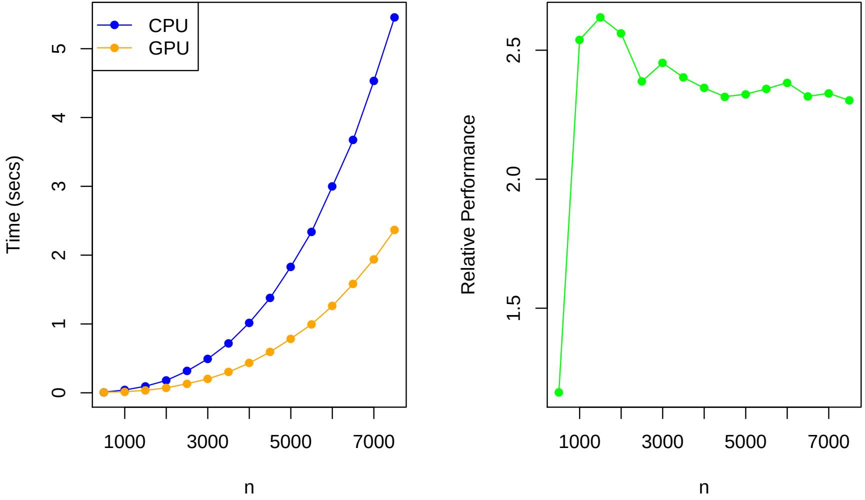
- Ubuntu 13.04
- OpenBlas 0.2.6
- CUDA 5.5 RC1
- Magma 1.4 beta1
- Armadillo 3.900.0

All performance metrics reflect pure C++ implementations (CPU) versus C++ / CUDA / Magma implementations (GPU).

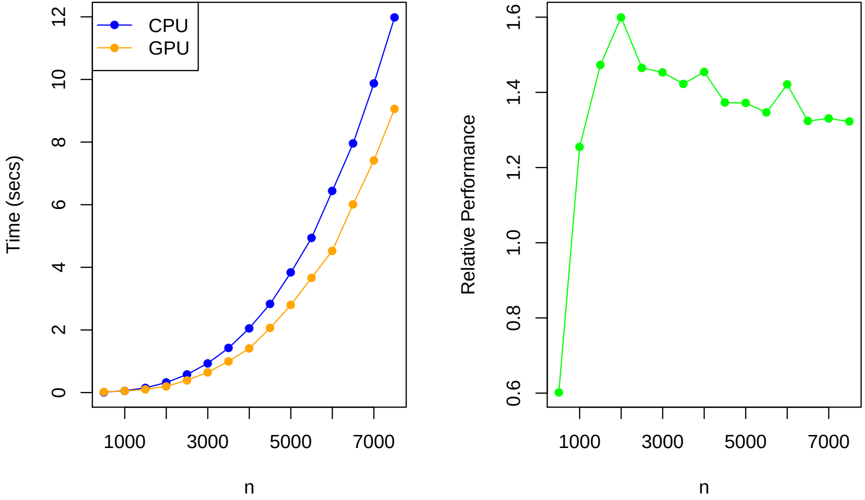
Performance - Calculating covariance matrix



Performance - Cholesky decomposition



Performance - Inverse covariance matrix



Putting RcppGP to use

spBayes is a fantastic package for fitting Bayesian spatial models, but

- wanted expanded functionality, in particular
  - arbitrary coordinate dimensions
  - more flexible covariance models
  - inspired by GPStuff
- improve performance wherever possible.

Putting RcppGP to use

spBayes is a fantastic package for fitting Bayesian spatial models, but

- wanted expanded functionality, in particular
  - arbitrary coordinate dimensions
  - more flexible covariance models
  - inspired by GPStuff
- improve performance wherever possible.

What to do?

Putting RcppGP to use

spBayes is a fantastic package for fitting Bayesian spatial models, but

- wanted expanded functionality, in particular
  - arbitrary coordinate dimensions
  - more flexible covariance models
  - inspired by GPStuff
- improve performance wherever possible.

What to do?

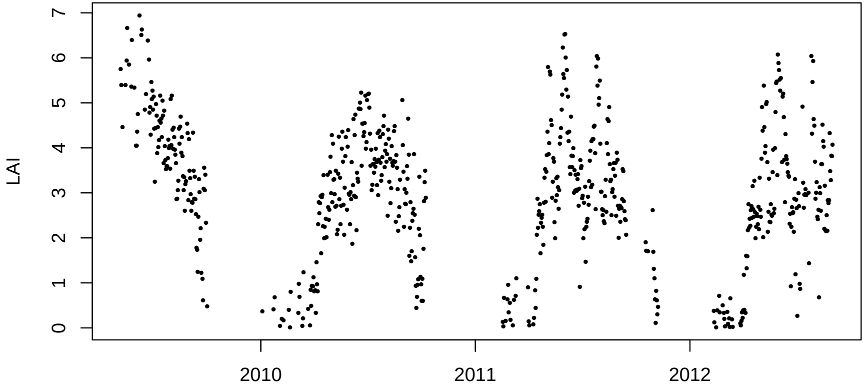
- Rewrote spLM and spPredict using Rcpp and RcppArmadillo.
- Modified the rewrite to use GPU using RcppGP.

CPU vs GPU Code

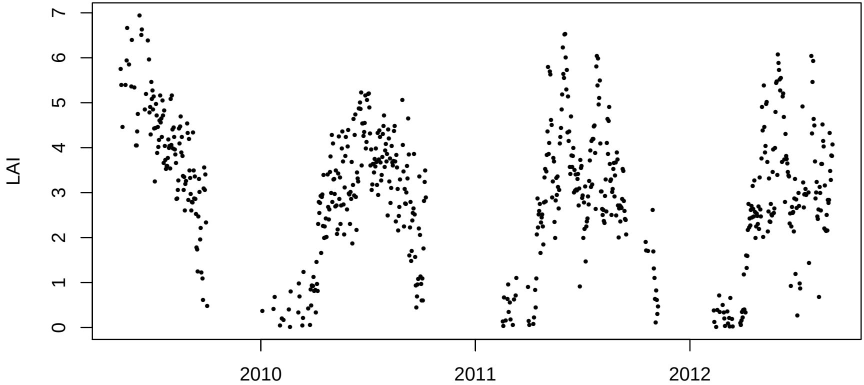
```
void update_covs(arma::mat const& coordsD)
{
    arma::mat C = m->calc_cov(coordsD, theta);

    C_U = arma::chol(C);
    arma::mat C_U_inv = arma::inv( arma::trimatu(C_U) );

    logDet = 2*arma::sum(arma::log(arma::diagvec(C_U)));

    Cinv = C_U_inv * C_U_inv.t();
}
```

```
void update_covs(gpu_mat const& coordsD)
{
    gpu_mat C( m->calc_cov_gpu_ptr(coordsD, theta),
               coordsD.n_rows, coordsD.n_cols );

    chol(C,'U');
    C_U = C.get_mat();

    inv_chol(C,'U');
    Cinv = C.get_mat();

    logDet = 2*arma::sum(arma::log(arma::diagvec(C_U)));
}
```

Example - Leaf Area Index

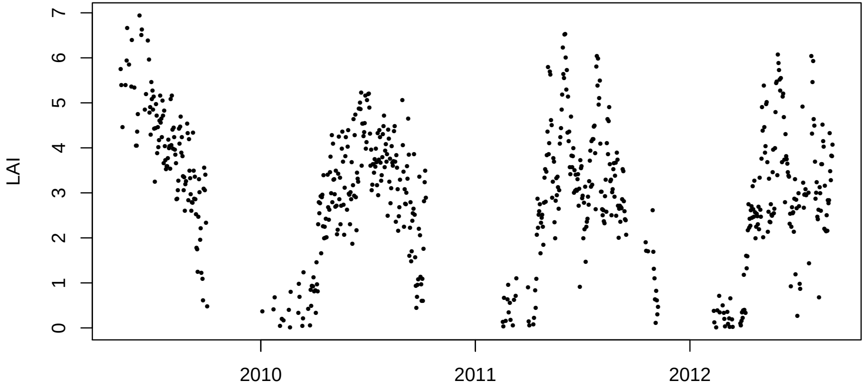


Example - Leaf Area Index



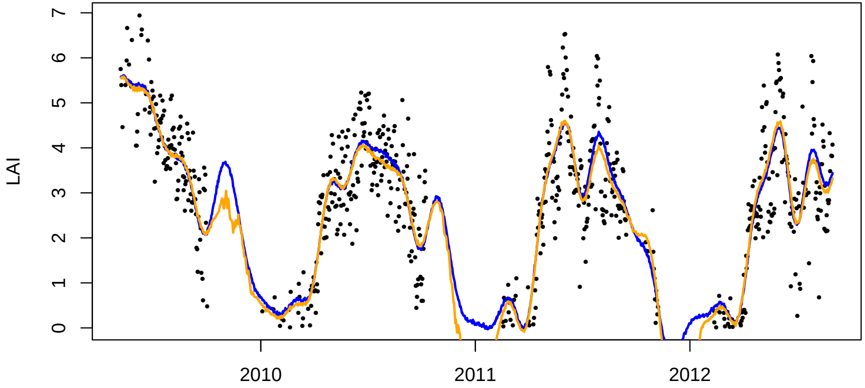
$$C(x,x') = \tau^2 I + \sigma^2 \exp\left(-\frac{1}{2}\left(\frac{\|x-x'\|}{1/\phi}\right)^2 - \frac{2\sin^2\left(\frac{1}{2}\frac{\|x-x'\|}{\gamma}\right)}{1/\lambda^2}\right)$$

Example - Leaf Area Index



$$C(x,x') = \boxed{\tau^2 I} + \sigma^2 \exp\left(\boxed{-\frac{1}{2}\left(\frac{\|x-x'\|}{1/\phi}\right)^2} \boxed{- \frac{2\sin^2\left(\frac{1}{2}\frac{\|x-x'\|}{\gamma}\right)}{1/\lambda^2}} \right)$$

Example - Leaf Area Index



$$C(x,x') = \tau^2 I + \sigma^2 \exp\left(-\frac{1}{2}\left(\frac{\|x-x'\|}{1/\phi}\right)^2 - \frac{2\sin^2\left(\frac{1}{2}\frac{\|x-x'\|}{\gamma}\right)}{1/\lambda^2} \right)$$

Covariance models

```
nugget_cov = list(type = "nugget", params = list(
                  list( name = "tauSq",
                        dist = "ig",
                        trans = "log",
                        start = 2,
                        tuning = 0.1,
                        hyperparams = c(2, 2)
                      )
             ))

perexp_cov = list(type = "periodic_exponential", params = list(
                  list( name = "sigmaSq",
                        dist = "ig",
                        trans = "log",
                        start = 1,
                        tuning = 0.1,
                        hyperparams = c(2, 2)
                      ),
                  list( name = "phi",
                        dist = "unif",
                        trans = "logit",
                        start = 1,
                        tuning = 0.1,
                        hyperparams = c(0, 2)
                      ),
                  list( name = "gamma",
                        dist = "fixed",
                        start = 365.25
                      ),
                  list( name = "decay",
                        dist = "unif",
                        trans = "logit",
                        start = 0.001,
                        tuning = 0.01,
                        hyperparams = c(0, 0.1)
                      )
             ))

cm = cov_model(nugget_cov, perexp_cov)
```

Example - Leaf Area Index - Results

Model Fitting:

$$n = 700,\ \#_{iter} = 5000$$

	Run time (sec)	sec / iter	Rel. Performance
CPU	491.8	0.0983	3.55
GPU	138.5	0.0277	1.00

Model Prediction:

$$n = 1212,\ \#_{pred} = 1000$$

	Run time (sec)	sec / iter	Rel. Performance
CPU	459.5	0.460	5.15
GPU	89.2	0.089	1.00

Summary

1. Goal is to make GPU computing available and painless for common bottlenecks
2. Use of GPU only for covariance calculations and common decompositions can result in a 3-5x speedup
3. High level functionality makes common tasks in GP models easier
4. Core tools are methodology agnostic
5. 12 minutes is not enough time to get into details, look at the code (particularly spPredict)

Summary

1. Goal is to make GPU computing available and painless for common bottlenecks
2. Use of GPU only for covariance calculations and common decompositions can result in a 3-5x speedup
3. High level functionality makes common tasks in GP models easier
4. Core tools are methodology agnostic
5. 12 minutes is not enough time to get into details, look at the code (particularly spPredict)

Coming soon:

1. Stabilization of interface
2. Compatibility with RcppAttributes
3. Full support for GPP and mGPP in spLM example
4. Documentation

Questions, Comments?

Email	:	rundel@gmail.com
RcppGP	:	*http://github.com/rundel/RcppGP*
Presentation	:	*http://github.com/rundel/Presentations/*