Gaussian Model Trees for Traffic Imputation

Sebastian Buschjäger, Thomas Liebig and Katharina Morik

TU Dortmund University - Artifical Intelligence Group

February 20, 2019

Motivation: Smart Cities

Motivation: Smart Cities

Idea Distribute small devices across the entire city to monitor specific locations

Motivation: Smart Cities

Idea Distribute small devices across the entire city to monitor specific locations

Design requirements

1. Sensing devices should be as small and as energy efficient as possible to minimize costs
2. Sensing devices should be low-priced to minimize initial investment costs
3. Data should not be processed globally to minimize communication and maximize privacy
4. Prediction models should be small, but accurate enough to be used on the sensing devices
5. The system should report possible sensor locations with respect to its accuracy.

Traffic Imputation

Our focus here Count the number of vehicles at a given coordinate (latitude / longitude)
Formally Imputation problem, where we impute missing sensor values

Traffic Imputation

Our focus here Count the number of vehicles at a given coordinate (latitude / longitude)
Formally Imputation problem, where we impute missing sensor values
Popular method Gaussian Processes

$$p(y|\mathcal{D}, \vec{x}) \sim \mathcal{N}(f(\vec{x}), \cdot)$$

with

$$f(\vec{x}) = \langle K(\vec{x}, \mathcal{D})K(\mathcal{D})^{-1}, \vec{y}\rangle$$

Traffic Imputation

Our focus here Count the number of vehicles at a given coordinate (latitude / longitude)
Formally Imputation problem, where we impute missing sensor values
Popular method Gaussian Processes

$$p(y|\mathcal{D},\vec{x}) \sim \mathcal{N}(f(\vec{x}),\cdot)$$

with

Kernel vector $[k(x,x_1),\ldots,k(x,x_N)]^T$

Target vector $[y_1,\ldots,y_N]^N$

$$f(\vec{x}) = \langle \boxed{K(\vec{x},\mathcal{D})}\boxed{K(\mathcal{D})}^{-1}, \boxed{\vec{y}}\rangle$$

Kernel matrix including noise $[k(x_i,x_j)]_{i,j} + \sigma_n I$

Traffic Imputation

Our focus here Count the number of vehicles at a given coordinate (latitude / longitude)
Formally Imputation problem, where we impute missing sensor values
Popular method Gaussian Processes

$$p(y|\mathcal{D}, \vec{x}) \sim \mathcal{N}(f(\vec{x}), \cdot)$$

with

Kernel vector $[k(x, x_1), \ldots, k(x, x_N)]^T$

Target vector $[y_1, \ldots, y_N]^N$

$$f(\vec{x}) = \langle K(\vec{x}, \mathcal{D}) K(\mathcal{D})^{-1}, \vec{y} \rangle$$

Kernel matrix including noise $[k(x_i, x_j)]_{i,j} + \sigma_n I$

Challenges

- GPs do not scale well, due to matrix inversion (runtime $\mathcal{O}(N^3)$)
- GPs do not have a traffic-flow model, e.g. by using map data

State of the art GPs

Scaleable GPs Well-studied problem with solutions utilizing subset of data points, sparse kernels, sparse approximation, implicit and explicit block structures, . . .

Important for us Each local sensing device should execute one small expert model

Deisenroth 2015 Distributed Gaussian Processes (DGP)
Idea Factorize global likelihood into product of m individual likelihoods

$$p(y|\mathcal{D}) \approx \prod_{k=1}^{m} \beta_k p_k(y|\mathcal{D}_k)$$

State of the art GPs

Scaleable GPs Well-studied problem with solutions utilizing subset of data points, sparse kernels, sparse approximation, implicit and explicit block structures, ...

Important for us Each local sensing device should execute one small expert model

Deisenroth 2015 Distributed Gaussian Processes (DGP)
Idea Factorize global likelihood into product of m individual likelihoods

$$p(y|\mathcal{D}) \approx \prod_{k=1}^{m} \underbrace{\boxed{\beta_k}}_{\text{Expert weight}} \underbrace{\boxed{p_k(y|\mathcal{D}_k)}}_{\text{Small GP with samples } \mathcal{D}_k \subset \mathcal{D}}$$

State of the art GPs

Scaleable GPs Well-studied problem with solutions utilizing subset of data points, sparse kernels, sparse approximation, implicit and explicit block structures, . . .

Important for us Each local sensing device should execute one small expert model

Deisenroth 2015 Distributed Gaussian Processes (DGP)
Idea Factorize global likelihood into product of m individual likelihoods

$$p(y|\mathcal{D}) \approx \prod_{k=1}^{m} \underbrace{\boxed{\beta_k}}_{\text{Expert weight}} \underbrace{\boxed{p_k(y|\mathcal{D}_k)}}_{\text{Small GP with samples } \mathcal{D}_k \subset \mathcal{D}}$$

Nice

- $+$ $p_k(y|\mathcal{D}_k)$ are independent from each other
- $+$ $\mathcal{D}_k$ can potentially be small

Problematic

- – All experts need to be evaluated to compute $p(y|\mathcal{D})$
- – $\mathcal{D}_k$ is randomly sampled

Gaussian Model Trees: Key questions

So far DGPs offer small expert models, which only require communication of local predictions

But 1 Is there a better way to sample $\mathcal{D}_k$?

But 2 Can we get away without any communication at all?

GP induction as loss minimization problem

$$\arg\min_{f \in \mathcal{H}} \frac{1}{2} ||f||^2_{\mathcal{H}} + \frac{1}{2\sigma_n^2} \sum_{(\vec{x},y) \in \mathcal{D}} (y_i - f(\vec{x}))^2$$

GP induction as loss minimization problem

Noise assumption from GP

$$\arg\min_{f \in \mathcal{H}} \boxed{\frac{1}{2}||f||^2_{\mathcal{H}}} + \boxed{\frac{1}{2\sigma_n^2}} \boxed{\sum_{(\vec{x},y)\in\mathcal{D}} (y_i - f(\vec{x}))^2}$$

Regularization: Norm of f in RKHS $\mathcal{H}$

MSE of GP model

GP induction as loss minimization problem

Noise assumption from GP

$$\arg\min_{f \in \mathcal{H}} \underbrace{\frac{1}{2}||f||^2_{\mathcal{H}}}_{\text{Regularization: Norm of } f \text{ in RKHS } \mathcal{H}} + \overbrace{\frac{1}{2\sigma_n^2}}^{\text{Noise assumption from GP}} \underbrace{\sum_{(\vec{x},y)\in\mathcal{D}} (y_i - f(\vec{x}))^2}_{\text{MSE of GP model}}$$

Regularization: Norm of f in RKHS $\mathcal{H}$

MSE of GP model

Goal Decompose optimization problem into two independent problems.

- Let $\mathcal{A} \subseteq \mathcal{D}$ denote a set of c inducing points. Let $\mathcal{B} = \mathcal{D} \setminus \mathcal{A}$
- Assume $k(\vec{x}_i, \vec{x}_j) \approx 0$ for $\vec{x}_i \in \mathcal{A}$ and $\vec{x}_j \in \mathcal{B}$

GP induction as loss minimization problem

Noise assumption from GP

$$\arg\min_{f \in \mathcal{H}} \boxed{\frac{1}{2}||f||^2_{\mathcal{H}}} + \boxed{\frac{1}{2\sigma_n^2}} \boxed{\sum_{(\vec{x},y)\in\mathcal{D}} (y_i - f(\vec{x}))^2}$$

Regularization: Norm of f in RKHS $\mathcal{H}$ MSE of GP model

Goal Decompose optimization problem into two independent problems.

- Let $\mathcal{A} \subseteq \mathcal{D}$ denote a set of c inducing points. Let $\mathcal{B} = \mathcal{D} \setminus \mathcal{A}$
- Assume $k(\vec{x}_i, \vec{x}_j) \approx 0$ for $\vec{x}_i \in \mathcal{A}$ and $\vec{x}_j \in \mathcal{B}$

Then we can split the optimization problem into two problems

$$\arg\min_{f_{\mathcal{A}} \in \mathcal{H}, f_{\mathcal{B}} \in \mathcal{H}} \frac{1}{2}||f_{\mathcal{A}}||^2_{\mathcal{H}} + \frac{1}{2\sigma_n^2} \sum_{(\vec{x},y)\in\mathcal{A}} (y - f_{\mathcal{A}}(\vec{x}))^2 + \frac{1}{2}||f_{\mathcal{B}}||^2_{\mathcal{H}} + \frac{1}{2\sigma_n^2} \sum_{(\vec{x},y)\in\mathcal{B}} (y - f_{\mathcal{B}}(\vec{x}))^2$$

GP induction as loss minimization problem

Noise assumption from GP

$$\arg\min_{f \in \mathcal{H}} \boxed{\frac{1}{2}||f||^2_{\mathcal{H}}} + \boxed{\frac{1}{2\sigma_n^2}} \boxed{\sum_{(\vec{x},y)\in\mathcal{D}} (y_i - f(\vec{x}))^2}$$

Regularization: Norm of f in RKHS $\mathcal{H}$

MSE of GP model

Goal Decompose optimization problem into two independent problems.

- Let $\mathcal{A} \subseteq \mathcal{D}$ denote a set of c inducing points. Let $\mathcal{B} = \mathcal{D} \setminus \mathcal{A}$
- Assume $k(\vec{x}_i, \vec{x}_j) \approx 0$ for $\vec{x}_i \in \mathcal{A}$ and $\vec{x}_j \in \mathcal{B}$

Then we can split the optimization problem into two problems

$$\arg\min_{f_{\mathcal{A}} \in \mathcal{H}, f_{\mathcal{B}} \in \mathcal{H}} \boxed{\frac{1}{2}||f_{\mathcal{A}}||^2_{\mathcal{H}} + \frac{1}{2\sigma_n^2} \sum_{(\vec{x},y)\in\mathcal{A}} (y - f_{\mathcal{A}}(\vec{x}))^2} + \boxed{\frac{1}{2}||f_{\mathcal{B}}||^2_{\mathcal{H}} + \frac{1}{2\sigma_n^2} \sum_{(\vec{x},y)\in\mathcal{B}} (y - f_{\mathcal{B}}(\vec{x}))^2}$$

$\boxed{f(\vec{x}) = \langle K(\vec{x}, \mathcal{A}) K(\mathcal{A})^{-1}, \vec{y} \rangle}$

$\boxed{f(\vec{x}) = \langle K(\vec{x}, \mathcal{B}) K(\mathcal{B})^{-1}, \vec{y} \rangle}$

Subset selection (1)

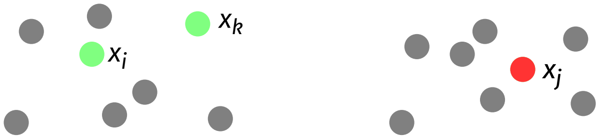
Question How to find sets $\mathcal{A}$ and $\mathcal{B}$?

Subset selection (1)

Question How to find sets $\mathcal{A}$ and $\mathcal{B}$?

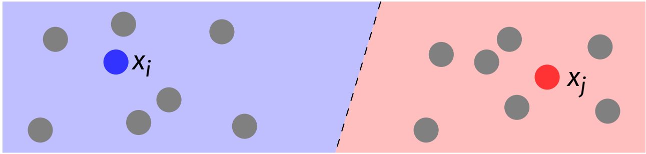
Subset selection (1)

Question How to find sets $\mathcal{A}$ and $\mathcal{B}$?



Subset selection (1)

Question How to find sets $\mathcal{A}$ and $\mathcal{B}$?



Observation If kernel is stationary, then $k(\vec{x}_i, \vec{x}_j) \approx 0 \Rightarrow k(\vec{x}_i, \vec{x}_k) \approx 0$ for $k(\vec{x}_j, \vec{x}_k) \approx 1$.

Thus Points $\vec{x}_j$ and $\vec{x}_k$ that are similar to each other, will have similar dissimilarity with $\vec{x}_i$

Subset selection (2)

Thus It is enough to store a reference point for each set $\mathcal{A}$ and $\mathcal{B}$.

Conclusion We need to find reference points which are maximally dissimilar to each other

Subset selection (2)

Thus It is enough to store a reference point for each set $\mathcal{A}$ and $\mathcal{B}$.

Conclusion We need to find reference points which are maximally dissimilar to each other

Idea Formulate another maximization problem

$$\frac{1}{2}\log\det\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} = \frac{1}{2}\log\left(k_{11}\cdot k_{22} - k_{12}\cdot k_{21}\right) \rightarrow \max \text{ if } k_{12} = k_{21} \approx 0$$

Subset selection (2)

Thus It is enough to store a reference point for each set $\mathcal{A}$ and $\mathcal{B}$.

Conclusion We need to find reference points which are maximally dissimilar to each other

Idea Formulate another maximization problem

$$\frac{1}{2}\log\det\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} = \frac{1}{2}\log\left(k_{11}\cdot k_{22} - k_{12}\cdot k_{21}\right) \rightarrow \max \text{ if } k_{12} = k_{21} \approx 0$$

More formally

$$\arg\max_{\mathcal{A}\subset\mathcal{D}, |\mathcal{A}|=c} \frac{1}{2}\log\det(I + aK(\mathcal{A}))$$

Subset selection (2)

Thus It is enough to store a reference point for each set $\mathcal{A}$ and $\mathcal{B}$.

Conclusion We need to find reference points which are maximally dissimilar to each other

Idea Formulate another maximization problem

$$\frac{1}{2}\log\det\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} = \frac{1}{2}\log\left(k_{11}\cdot k_{22} - k_{12}\cdot k_{21}\right) \rightarrow \max \text{ if } k_{12} = k_{21} \approx 0$$

More formally

$$\arg\max_{\mathcal{A}\subset\mathcal{D}, |\mathcal{A}|=c} \frac{1}{2}\log\det(I + aK(\mathcal{A}))$$

Still This is a very difficult problem, since we need to check all possible subsets of $\mathcal{A}\subset\mathcal{D}$

Lawrence 2003 $\frac{1}{2}\log\det(\mathcal{I} + aK(\mathcal{A}))$ is sub-modular

Subset selection (2)

Thus It is enough to store a reference point for each set $\mathcal{A}$ and $\mathcal{B}$.

Conclusion We need to find reference points which are maximally dissimilar to each other

Idea Formulate another maximization problem

$$\frac{1}{2}\log\det\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} = \frac{1}{2}\log\left(k_{11}\cdot k_{22} - k_{12}\cdot k_{21}\right) \rightarrow \max \text{ if } k_{12} = k_{21} \approx 0$$

More formally

$$\arg\max_{\mathcal{A}\subset\mathcal{D},|\mathcal{A}|=c} \frac{1}{2}\log\det(I + aK(\mathcal{A}))$$

Still This is a very difficult problem, since we need to check all possible subsets of $\mathcal{A}\subset\mathcal{D}$

Lawrence 2003 $\frac{1}{2}\log\det(\mathcal{I} + aK(\mathcal{A}))$ is sub-modular

Why submodularity? It offers a simple algorithm with guaranteed performance

Nemhaus 1978 SimpleGreedy has a guaranteed performance of $\geq 1 - (1/e) \approx 63\%$

Putting it all together (1)

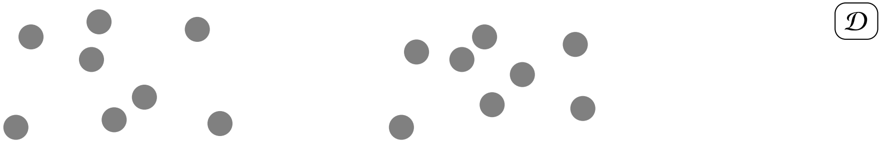
Overall approach Greedy Top-Down algorithm

- Select c 'most dissimilar' samples
- View each sample as 'region'
- Repeat until only M points or less are present in a region. Train a full GP on those regions.

Putting it all together (1)

Overall approach Greedy Top-Down algorithm

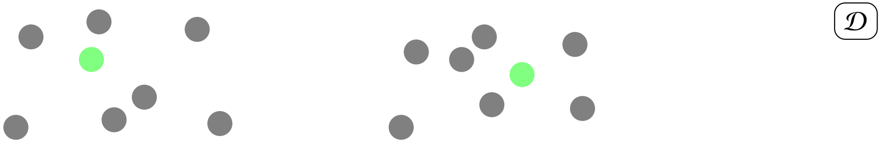
- Select c ‘most dissimilar’ samples
- View each sample as ‘region’
- Repeat until only M points or less are present in a region. Train a full GP on those regions.



Putting it all together (1)

Overall approach Greedy Top-Down algorithm

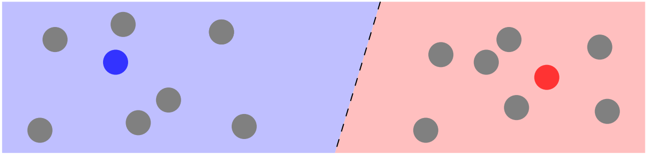
- Select c 'most dissimilar' samples
- View each sample as 'region'
- Repeat until only M points or less are present in a region. Train a full GP on those regions.

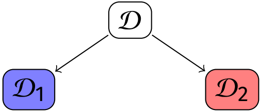


Putting it all together (1)

Overall approach Greedy Top-Down algorithm

- Select c ‘most dissimilar’ samples
- View each sample as ‘region’
- Repeat until only M points or less are present in a region. Train a full GP on those regions.

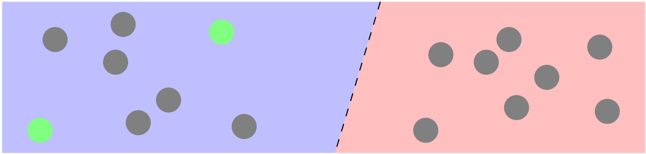


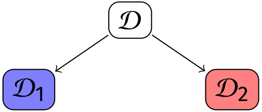


Putting it all together (1)

Overall approach Greedy Top-Down algorithm

- Select c 'most dissimilar' samples
- View each sample as 'region'
- Repeat until only M points or less are present in a region. Train a full GP on those regions.

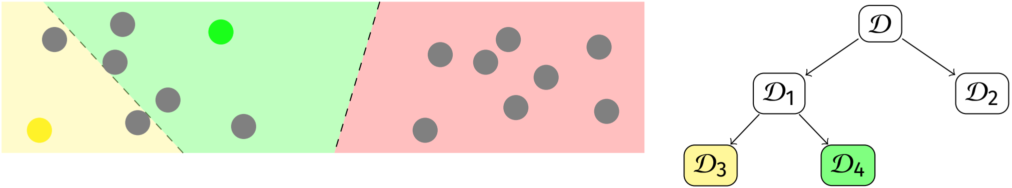




Putting it all together (1)

Overall approach Greedy Top-Down algorithm

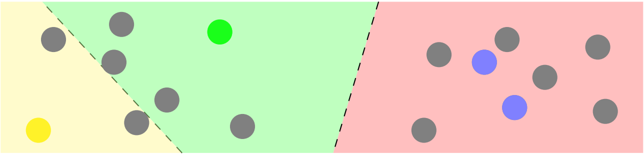
- Select c 'most dissimilar' samples
- View each sample as 'region'
- Repeat until only M points or less are present in a region. Train a full GP on those regions.

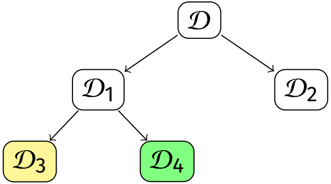


Putting it all together (1)

Overall approach Greedy Top-Down algorithm

- Select c ‘most dissimilar’ samples
- View each sample as ‘region’
- Repeat until only M points or less are present in a region. Train a full GP on those regions.

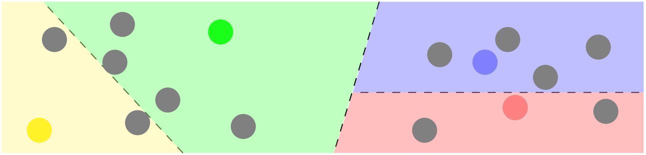


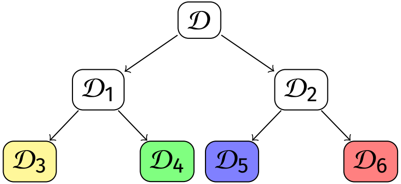


Putting it all together (1)

Overall approach Greedy Top-Down algorithm

- Select c 'most dissimilar' samples
- View each sample as 'region'
- Repeat until only M points or less are present in a region. Train a full GP on those regions.

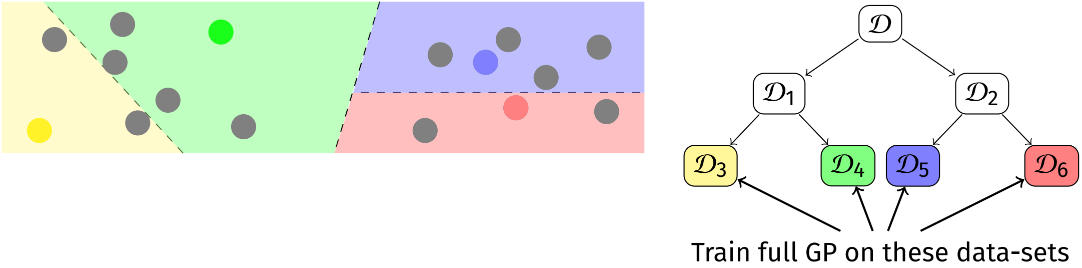




Putting it all together (1)

Overall approach Greedy Top-Down algorithm

- Select c 'most dissimilar' samples
- View each sample as 'region'
- Repeat until only M points or less are present in a region. Train a full GP on those regions.



Putting it all together (2)

Algorithm 2 Gaussian Model Tree (GMT).

```
function TRAINGMT(𝒟, c, τ)
    if |𝒟| ≥ τ then
        𝒜 = SimpleGreedy(𝒟, c)
        for (x, y) ∈ 𝒟 do
            r = arg max{k(x, e)|e ∈ 𝒜}
            𝒟_r = 𝒟_r ∪ {x}
        for i = 1, ..., c do
            trainGMT(𝒟_i, c, τ)
    else
        trainFullGP(𝒟)
```

Parameters

- $\mathcal{D}$: Training data
- c: Number of regions
  ($\rightarrow$ Number of children per inner node)
- τ: Minimum number of data points
  ($\rightarrow$ size of experts in the end)

Note We can parallelise over c. The expected runtime is $O(log_c(n) \cdot n \cdot c^2 + n \cdot \tau^3)$

Experimental setup

Question 1 What is the accuracy of the proposed method?

Question 2 How much memory is required per node?

Experimental setup

Question 1 What is the accuracy of the proposed method?

Question 2 How much memory is required per node?

Approach Use traffic simulator SUMO to generate data with sufficient ground truth

- 24h simulation for the City of Luxembourg
- 3523 simulated sensor available
- We simulated 131357 vehicle counts per sensor from 7:00 till 11:00

Goal predict average number of vehicles per sensor node (given as its coordinates)

Results on Luxembourg data set

Error measure Standardized mean-squared error

$$SMSE = \frac{1}{var\left(\mathcal{D}_{Test}\right)\left|\mathcal{D}_{Test}\right|} \sum_{(\vec{x},y) \in \mathcal{D}_{Test}} (f(\vec{x}) - y)^2$$

Observation The average prediction $f(\vec{x}) = 1/N \sum_i y_i$ has a SMSE of roughly 1

Results on Luxembourg data set

Error measure Standardized mean-squared error

$$SMSE = \frac{1}{var\left(\mathcal{D}_{Test}\right)\left|\mathcal{D}_{Test}\right|} \sum_{(\vec{x},y)\in\mathcal{D}_{Test}} (f(\vec{x}) - y)^2$$

Observation The average prediction $f(\vec{x}) = 1/N \sum_i y_i$ has a SMSE of roughly 1

Experiments Compare 576 different hyperparameter combinations with a 5-fold cross validation.

Method and Parameters	Kernel	SMSE	Avg. Size
Full GP, $c = 1000$	0.5/0.5	0.767	1000
Informative Vector Machine, $c = 500$	2.0/2.0	0.866	500
Distributed GPs, $c = 2800$, $m = 50$	0.5/0.5	0.733	2800
Gaussian Model Trees, $c = 50$, $\tau = 1000$	1.0/2.0	0.583	56.80

Table: Parameter configuration with smallest SMSE per algorithm.

Results on Luxembourg data set

Error measure Standardized mean-squared error

$$SMSE = \frac{1}{var\left(\mathcal{D}_{Test}\right)\left|\mathcal{D}_{Test}\right|} \sum_{(\vec{x},y)\in\mathcal{D}_{Test}} (f(\vec{x}) - y)^2$$

Observation The average prediction $f(\vec{x}) = 1/N \sum_i y_i$ has a SMSE of roughly 1

Experiments Compare 576 different hyperparameter combinations with a 5-fold cross validation.

Method and Parameters	Kernel	SMSE	Avg. Size
Full GP, $c = 1000$	0.5/0.5	0.767	1000
Informative Vector Machine, $c = 500$	2.0/2.0	0.866	500
Distributed GPs, $c = 2800$, $m = 50$	0.5/0.5	0.733	2800
Gaussian Model Trees, $c = 50$, $\tau = 1000$	1.0/2.0	0.583	56.80

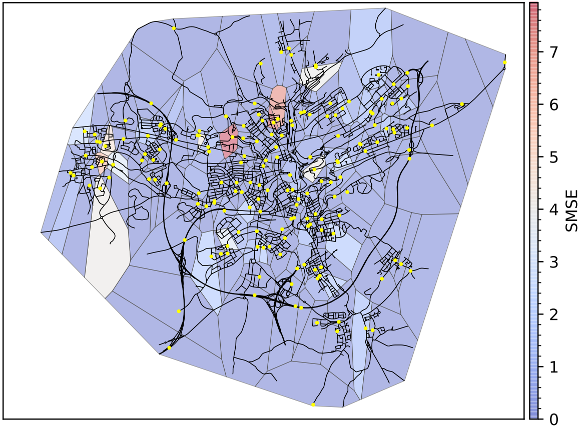
Table: Parameter configuration with smallest SMSE per algorithm.

Observation 1 GMT compares favorably to FPG and DGP.

Observation 2 GMT requires 17 – 58 times fewer resources per node than FGP and DGP!

Results on Luxembourg data set (2)

Nice bonus We can visualize the regions where GMT fails



Recap: Gaussian Model Trees

Goal Distribute small sensor devices in the city each with a small, locale ML model

- View GP induction as optimization problem
- Decompose optimization problem into independent sub-problems
- View decomposition as sample selection with guaranteed performance by submodularity
- Built a tree-structured classifier by recursively partition data into smaller sub-problems

Recap: Gaussian Model Trees

Goal Distribute small sensor devices in the city each with a small, locale ML model

- View GP induction as optimization problem
- Decompose optimization problem into independent sub-problems
- View decomposition as sample selection with guaranteed performance by submodularity
- Built a tree-structured classifier by recursively partition data into smaller sub-problems

So far Very promising results on data in the context of Smart Cities

Recap: Gaussian Model Trees

Goal Distribute small sensor devices in the city each with a small, locale ML model

- View GP induction as optimization problem
- Decompose optimization problem into independent sub-problems
- View decomposition as sample selection with guaranteed performance by submodularity
- Built a tree-structured classifier by recursively partition data into smaller sub-problems

So far Very promising results on data in the context of Smart Cities

Outlook

- Use different kernel hyperparameters per node
- Gaussian assumption often violated $\rightarrow$ Use other prediction methods in leaf-node.
- Borrow ideas from Decision Trees for post- and pre-pruning

`https://bitbucket.org/sbuschjaeger/ensembles/src`

More experiments

Note Full GP is still manageable with $N = 3523$. What about bigger data-sets?

More experiments

Note Full GP is still manageable with $N = 3523$. What about bigger data-sets?

First follow-up experiment UK-traffic imputation data from 2017

- Same as Luxembourg task, but in the UK with $N = 18149$ sensors

Second follow-up experiment 'Rate' an area in the city, e.g. by quality of life.

Problem No good data available. Thus we used a (arguably bad) proxy data set

- Predict the apartment price given its coordinates in the UK from 2015
- In total $N = 64431$
- No further information given on the apartments

Results on UK data sets

Again Compare 576 different hyperparameter configurations with a 5-fold cross validation.

Method and Parameters	Kernel	SMSE	Avg. Size
FGP, $c = 500$	0.5/2.0	0.967	500
IVM, $c = 300$	2.0/5.0	0.972	300
DGP, $c = 1000, m = 100$	0.5/0.5	0.951	1000
GMT, $c = 300, \tau = 500$	2.0/5.0	0.865	49.69

Table: Parameter configuration with smallest SMSE per algorithm on UK traffic data.

Method and Parameters	Kernel	SMSE	Avg. Size
FGP, $c = 500$	1.0/0.5	0.934	500
IVM, $c = 300$	0.5/2.0	0.947	300
DGP, $c = 500, m = 200$	1.0/0.5	0.92	500
GMT, $c = 100, \tau = 500$	0.5/1.0	0.553	177.317

Table: Parameter configuration with smallest SMSE per algorithm on UK apartment-price data.