GAUSSIAN MODEL TREES FOR TRAFFIC IMPUTATION

Sebastian Buschjäger, Thomas Liebig and Katharina Morik

TU Dortmund University - Artificial 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

\[
p(y|\mathcal{D}, x) \sim N(f(\tilde{x}), \cdot)
\]

with

\[
f(\tilde{x}) = \langle K(\tilde{x}, \mathcal{D})K(\mathcal{D})^{-1}, \tilde{y} \rangle
\]
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

\[
p(y|D, \tilde{x}) \sim N(f(\tilde{x}), \cdot)
\]

with

- Kernel vector \([k(x, x_1), \ldots, k(x, x_N)]^T\)
- Target vector \([y_1, \ldots, y_N]^N\)
- 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

\[
p(y|\mathcal{D}, \tilde{x}) \sim \mathcal{N}(f(\tilde{x}), \cdot)
\]

with

- Kernel vector \([k(x, x_1), \ldots, k(x, x_N)]^T\)
- Target vector \([y_1, \ldots, y_N]^N\)
- \(f(\tilde{x}) = \langle K(\tilde{x}, \mathcal{D})K(\mathcal{D})^{-1}, \tilde{y} \rangle\)
- 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 \(O(N^3)\))
- 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

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

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

- Small GP with samples $\mathcal{D}_k \subset \mathcal{D}$
- Expert weight
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

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

- **Nice**
  - $p_k(y|D_k)$ are independent from each other
  - $D_k$ can potentially be small

- **Problematic**
  - All experts need to be evaluated to compute $p(y|D)$
  - $D_k$ 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 $D_k$?

But 2 Can we get away without any communication at all?
GP induction as loss minimization problem

\[
\arg\min_{f \in \mathcal{H}} \frac{1}{2} \|f\|_{\mathcal{H}}^2 + \frac{1}{2\sigma^2} \sum_{(\tilde{x}, y) \in \mathcal{D}} (y_i - f(\tilde{x}))^2
\]
GP induction as loss minimization problem

Noise assumption from GP

\[
\arg\min_{f \in \mathcal{H}} \left( \frac{1}{2} \|f\|^2_{\mathcal{H}} + \frac{1}{2\sigma^2_n} \sum_{(\tilde{x}, y) \in \mathcal{D}} (y_i - f(\tilde{x}))^2 \right)
\]

Regularization: Norm of \( f \) in RKHS \( \mathcal{H} \)  \quad \text{MSE of GP model}
GP induction as loss minimization problem

Noise assumption from GP

\[
\arg\min_{f \in \mathcal{H}} \frac{1}{2} \|f\|^2_{\mathcal{H}} + \frac{1}{2\sigma^2_n} \sum_{(\tilde{x}, y) \in D} (y_i - f(\tilde{x}))^2
\]

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

MSE of GP model

Goal

Decompose optimization problem into two independent problems.

- Let \( \mathcal{A} \subseteq D \) denote a set of \( c \) inducing points. Let \( \mathcal{B} = D \setminus \mathcal{A} \)
- Assume \( k(\tilde{x}_i, \tilde{x}_j) \approx 0 \) for \( \tilde{x}_i \in \mathcal{A} \) and \( \tilde{x}_j \in \mathcal{B} \)
GP induction as loss minimization problem

Noise assumption from GP

\[
\arg \min_{f \in \mathcal{H}} \left( \frac{1}{2} ||f||_{\mathcal{H}}^2 + \frac{1}{2\sigma_n^2} \sum_{(\tilde{x}, y) \in \mathcal{D}} (y - f(\tilde{x}))^2 \right)
\]

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 \( k(\tilde{x}_i, \tilde{x}_j) \approx 0 \) for \( \tilde{x}_i \in \mathcal{A} \) and \( \tilde{x}_j \in \mathcal{B} \)

Then we can split the optimization problem into two problems

\[
\arg \min_{f_{\mathcal{A}} \in \mathcal{H}, f_{\mathcal{B}} \in \mathcal{H}} \frac{1}{2} ||f_{\mathcal{A}}||_{\mathcal{H}}^2 + \frac{1}{2\sigma_n^2} \sum_{(\tilde{x}, y) \in \mathcal{A}} (y - f_{\mathcal{A}}(\tilde{x}))^2 + \quad \frac{1}{2} ||f_{\mathcal{B}}||_{\mathcal{H}}^2 + \frac{1}{2\sigma_n^2} \sum_{(\tilde{x}, y) \in \mathcal{B}} (y - f_{\mathcal{B}}(\tilde{x}))^2
\]
GP induction as loss minimization problem

Noise assumption from GP

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

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

**Goal** Decompose optimization problem into two independent problems.

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

Then we can split the optimization problem into two problems

\[
\begin{align*}
\arg \min_{f_\mathcal{A} \in \mathcal{H}, f_\mathcal{B} \in \mathcal{H}} & \quad \frac{1}{2} \| f_\mathcal{A} \|_{\mathcal{H}}^2 + \frac{1}{2\sigma_n^2} \sum_{(\tilde{x}, y) \in \mathcal{A}} (y - f_\mathcal{A}(\tilde{x}))^2 + \\
& \quad + \frac{1}{2} \| f_\mathcal{B} \|_{\mathcal{H}}^2 + \frac{1}{2\sigma_n^2} \sum_{(\tilde{x}, y) \in \mathcal{B}} (y - f_\mathcal{B}(\tilde{x}))^2 \\
\end{align*}
\]

\[
f(\tilde{x}) = \langle K(\tilde{x}, \mathcal{A})K(\mathcal{A})^{-1}, \tilde{y} \rangle
\]

\[
f(\tilde{x}) = \langle K(\tilde{x}, \mathcal{B})K(\mathcal{B})^{-1}, \tilde{y} \rangle
\]
Subset selection (1)

**Question** How to find sets $A$ and $B$?
**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

$$k(\mathbf{x}_i, \mathbf{x}_j) \approx 0 \Rightarrow k(\mathbf{x}_i, \mathbf{x}_k) \approx 0 \text{ for } k(\mathbf{x}_j, \mathbf{x}_k) \approx 1.$$
**Subset selection (1)**

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

![Subset selection diagram with points $x_i$, $x_j$, and $x_k$.]

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

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

8 / 17
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

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

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

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

Idea Formulate another maximization problem

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

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

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

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** $\frac{1}{2} \log \det(I + aK(\mathcal{A}))$ 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

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

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

Why submodularity? It offers a simple algorithm with guaranteed performance

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

Train full GP on these data-sets
Putting it all together (2)

Algorithm 2 Gaussian Model Tree (GMT).

1: function TRAINGMT(\(\mathcal{D}, c, \tau\))
2:  if |\(\mathcal{D}\)| ≥ \(\tau\) then
3:      \(\mathcal{A}\) = SimpleGreedy(\(\mathcal{D}, c\))
4:      for \((x, y)\) ∈ \(\mathcal{D}\) do
5:         \(r = \arg \max\{k(x, e)| e \in \mathcal{A}\}\)
6:         \(\mathcal{D}_r = \mathcal{D}_r \cup \{x\}\)
7:      for \(i = 1, \ldots, c\) do
8:         trainGMT(\(\mathcal{D}_i, c, \tau\))
9:  else
10:     trainFullGP(\(\mathcal{D}\))

Parameters

- \(\mathcal{D}\): Training data
- \(c\): Number of regions
  (∆ Number of children per inner node)
- \(\tau\): Minimum number of data points
  (∆ 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

$$SMSE = \frac{1}{\text{var}(D_{\text{Test}}) |D_{\text{Test}}|} \sum_{(\tilde{x},y) \in D_{\text{Test}}} (f(\tilde{x}) - y)^2$$

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

Experiments

<table>
<thead>
<tr>
<th>Method and Parameters</th>
<th>Kernel</th>
<th>SMSE Avg. Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full GP, $c = 1000$</td>
<td>0.5/0.5</td>
<td>0.767 1000</td>
</tr>
<tr>
<td>Informative Vector Machine, $c = 500$</td>
<td>2/2</td>
<td>0.866 500</td>
</tr>
<tr>
<td>Distributed GPs, $c = 2800$, $m = 50$</td>
<td>0.5/0.5</td>
<td>0.733 2800</td>
</tr>
<tr>
<td>Gaussian Model Trees, $c = 50$, $\tau = 1000$</td>
<td>1/2</td>
<td>0.583 56.80</td>
</tr>
</tbody>
</table>

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

Error measure Standardized mean-squared error

\[
SMSE = \frac{1}{\text{var}(D_{\text{Test}}) |D_{\text{Test}}|} \sum_{(\tilde{x}, y) \in D_{\text{Test}}} (f(\tilde{x}) - y)^2
\]

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

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

<table>
<thead>
<tr>
<th>Method and Parameters</th>
<th>Kernel</th>
<th>SMSE</th>
<th>Avg. Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full GP, ( c = 1000 )</td>
<td>0.5/0.5</td>
<td>0.767</td>
<td>1000</td>
</tr>
<tr>
<td>Informative Vector Machine, ( c = 500 )</td>
<td>2.0/2.0</td>
<td>0.866</td>
<td>500</td>
</tr>
<tr>
<td>Distributed GPs, ( c = 2800, m = 50 )</td>
<td>0.5/0.5</td>
<td>0.733</td>
<td>2800</td>
</tr>
<tr>
<td>Gaussian Model Trees, ( c = 50, \tau = 1000 )</td>
<td>1.0/2.0</td>
<td>0.583</td>
<td>56.80</td>
</tr>
</tbody>
</table>

Table: Parameter configuration with smallest SMSE per algorithm.
Results on Luxembourg data set

**Error measure** Standardized mean-squared error

\[
SMSE = \frac{1}{\text{var}(D_{\text{Test}})|D_{\text{Test}}|} \sum_{(\tilde{x}, y) \in D_{\text{Test}}} (f(\tilde{x}) - y)^2
\]

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

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

<table>
<thead>
<tr>
<th>Method and Parameters</th>
<th>Kernel</th>
<th>SMSE</th>
<th>Avg. Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full GP, ( c = 1000 )</td>
<td>0.5/0.5</td>
<td>0.767</td>
<td>1000</td>
</tr>
<tr>
<td>Informative Vector Machine, ( c = 500 )</td>
<td>2.0/2.0</td>
<td>0.866</td>
<td>500</td>
</tr>
<tr>
<td>Distributed GPs, ( c = 2800, m = 50 )</td>
<td>0.5/0.5</td>
<td>0.733</td>
<td>2800</td>
</tr>
<tr>
<td>Gaussian Model Trees, ( c = 50, \tau = 1000 )</td>
<td>1.0/2.0</td>
<td>0.583</td>
<td>56.80</td>
</tr>
</tbody>
</table>

**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 → Use other prediction methods in leaf-node.
- Borrow ideas from Decision Trees for post- and pre-pruning

[https://bitbucket.org/sbuschjaeger/ensembles/src](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.

<table>
<thead>
<tr>
<th>Method and Parameters</th>
<th>Kernel</th>
<th>SMSE</th>
<th>Avg. Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGP, ( c = 500 )</td>
<td>0.5/2.0</td>
<td>0.967</td>
<td>500</td>
</tr>
<tr>
<td>IVM, ( c = 300 )</td>
<td>2.0/5.0</td>
<td>0.972</td>
<td>300</td>
</tr>
<tr>
<td>DGP, ( c = 1000, m = 100 )</td>
<td>0.5/0.5</td>
<td>0.951</td>
<td>1000</td>
</tr>
<tr>
<td>GMT, ( c = 300, \tau = 500 )</td>
<td>2.0/5.0</td>
<td>0.865</td>
<td>49.69</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method and Parameters</th>
<th>Kernel</th>
<th>SMSE</th>
<th>Avg. Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGP, ( c = 500 )</td>
<td>1.0/0.5</td>
<td>0.934</td>
<td>500</td>
</tr>
<tr>
<td>IVM, ( c = 300 )</td>
<td>0.5/2.0</td>
<td>0.947</td>
<td>300</td>
</tr>
<tr>
<td>DGP, ( c = 500, m = 200 )</td>
<td>1.0/0.5</td>
<td>0.92</td>
<td>500</td>
</tr>
<tr>
<td>GMT, ( c = 100, \tau = 500 )</td>
<td>0.5/1.0</td>
<td>0.553</td>
<td>177.317</td>
</tr>
</tbody>
</table>

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