Escuela Superior de Informática

Universidad de Castilla-La Mancha

MAT

Nyström-based approximations for Kernel Logistic Regression: Application to transport choice modelling

José Ángel Martín-Baos

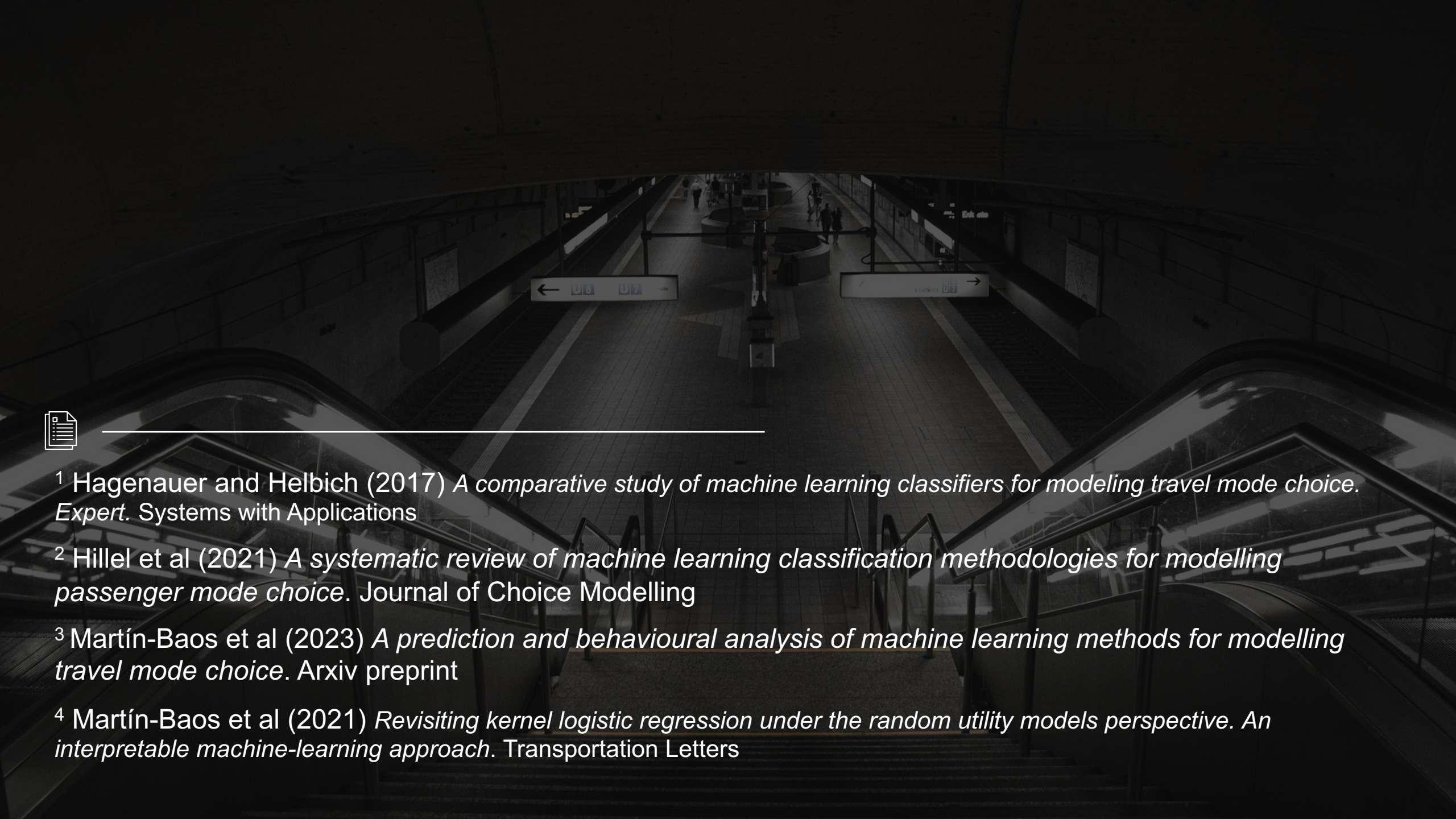
Ricardo García-Ródenas

Luis Rodriguez-Benitez

Michel Bierlaire

17-21 July 2023

World Conference on Transport Research



[1] Hagenauer and Helbich (2017) *A comparative study of machine learning classifiers for modeling travel mode choice. Expert.* Systems with Applications

[2] Hillel et al (2021) *A systematic review of machine learning classification methodologies for modelling passenger mode choice*. Journal of Choice Modelling

[3] Martín-Baos et al (2023) *A prediction and behavioural analysis of machine learning methods for modelling travel mode choice*. Arxiv preprint

[4] Martín-Baos et al (2021) *Revisiting kernel logistic regression under the random utility models perspective. An interpretable machine-learning approach*. Transportation Letters

$$U_{in} = V_{in} + \epsilon_{in}$$

$$U_{in} = \boxed{\text{KLR}} + \epsilon_{in}$$

KLR

$$V_i(\mathbf{x} \mid \boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_{in} k(\mathbf{x}_{in}, \mathbf{x})$$

$$[\mathbf{K}_i]_{n,n'} = k(\mathbf{x}_{in}, \mathbf{x}_{in'}) \quad \text{for } n, n' = 1, \ldots, N$$

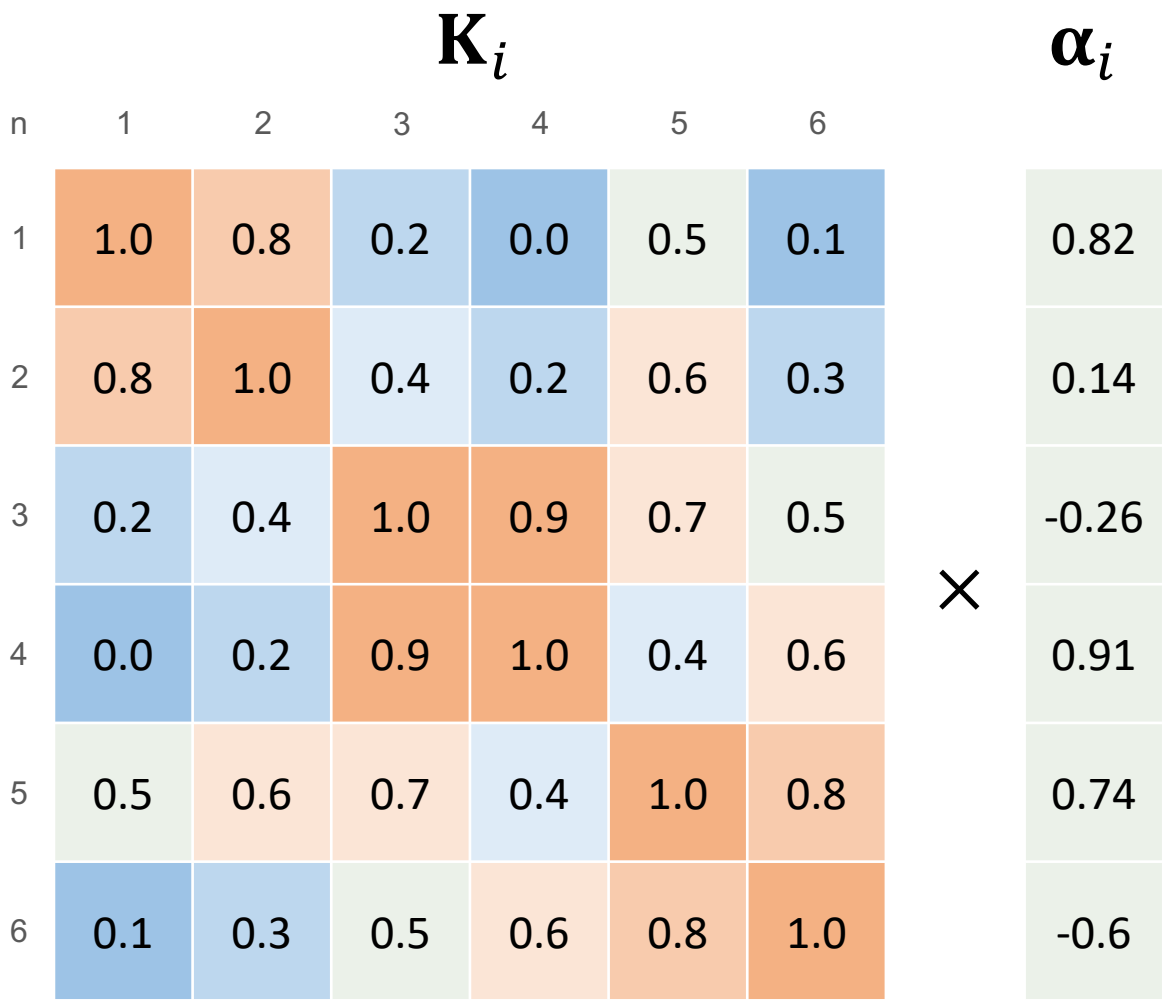
$$V_i(\mathbf{x} \mid \boldsymbol{\alpha}) = \mathbf{K}_i^{(n)\top} \boldsymbol{\alpha}_i$$

$$\mathbf{K}_i$$

n	1	2	3	4	5	6
1	1.0	0.8	0.2	0.0	0.5	0.1
2	0.8	1.0	0.4	0.2	0.6	0.3
3	0.2	0.4	1.0	0.9	0.7	0.5
4	0.0	0.2	0.9	1.0	0.4	0.6
5	0.5	0.6	0.7	0.4	1.0	0.8
6	0.1	0.3	0.5	0.6	0.8	1.0

$$\mathbf{K}_i$$

n	1	2	3	4	5	6
1	1.0	0.8	0.2	0.0	0.5	0.1
2	0.8	1.0	0.4	0.2	0.6	0.3
3	0.2	0.4	1.0	0.9	0.7	0.5
4	0.0	0.2	0.9	1.0	0.4	0.6
5	0.5	0.6	0.7	0.4	1.0	0.8
6	0.1	0.3	0.5	0.6	0.8	1.0


$\mathbf{K}_i$
$\boldsymbol{\alpha}_i$
n
1
2
3
4
5
6
1.0 0.8 0.2 0.0 0.5 0.1
0.8 1.0 0.4 0.2 0.6 0.3
0.2 0.4 1.0 0.9 0.7 0.5
0.0 0.2 0.9 1.0 0.4 0.6
0.5 0.6 0.7 0.4 1.0 0.8
0.1 0.3 0.5 0.6 0.8 1.0
×
0.82
0.14
-0.26
0.91
0.74
-0.6

$$\mathbb{P}(\, i \mid \mathbf{K}_n, \boldsymbol{\alpha}\,) = \frac{e^{V_i}}{\sum_{j=1}^{I} e^{V_j}} = \frac{e^{\mathbf{K}_i^{(n)\top}\boldsymbol{\alpha}_i}}{\sum_{j=1}^{I} e^{\mathbf{K}_j^{(n)\top}\boldsymbol{\alpha}_j}}$$

Numerical results

GKLR Python package

https://github.com/JoseAngelMartinB/gklr

- Ubuntu 20.04 LTS
- 3.8 GHz 12 core AMD Ryzen
- 32 GB of RAM

LPMC

- Single day travel diary data from 2012 to 2015.
- 81,096 surveys with 31 variables.
- After pre-processing, 20 variables selected.

 **35%**  **44%**  **3%**  **18%**

NTS

- ML focused dataset:
  - Data from a Dutch transport survey from 2010 to 2012.
  - Environmental data.
- 230,608 surveys with 16 variables.

 **4%**  **55%**  **24%**  **17%**

KLR estimation problem

Spatial complexity to store the Gram matrix

$$\mathcal{O}(N^2)$$

Computational cost of V

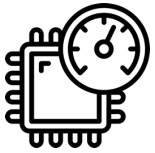
$$\mathcal{O}(N^2)$$

Nyström method

$$V_i = \mathbf{K}_i \boldsymbol{\alpha}_i$$

$$\mathbf{K} \approx \mathbf{K}_{N,L} \cdot \mathbf{K}_{L,L}^{\dagger} \cdot \mathbf{K}_{N,L}^{\top}, \quad \text{with } L \ll N$$

$$V_i = \mathbf{K}_{N,L} \cdot \left(\mathbf{K}_{L,L}^{\dagger} \cdot \left(\mathbf{K}_{N,L}^{\top} \cdot \boldsymbol{\alpha} \right) \right)$$

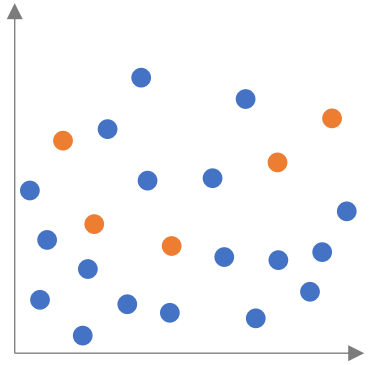


Complexity $\mathcal{O}(N \cdot L)$

Nyström method

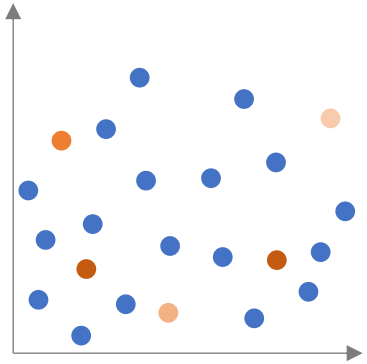
Random strategy

Nyström KLR



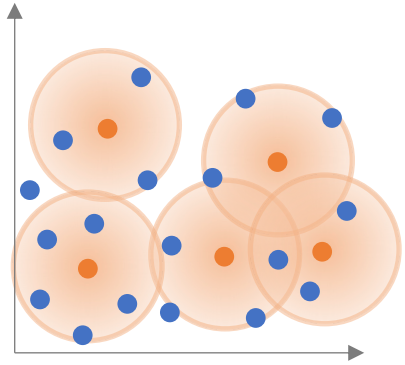
Divide-and-conquer ridge-leverage strategy

DAC ridge-leverage Nyström KLR



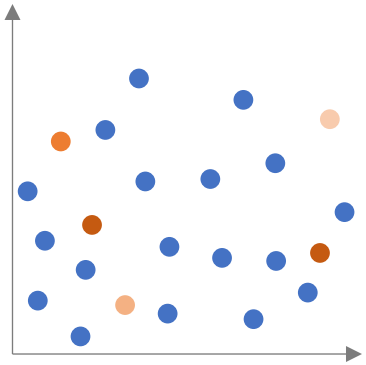
K-means strategy

K-means Nyström KLR

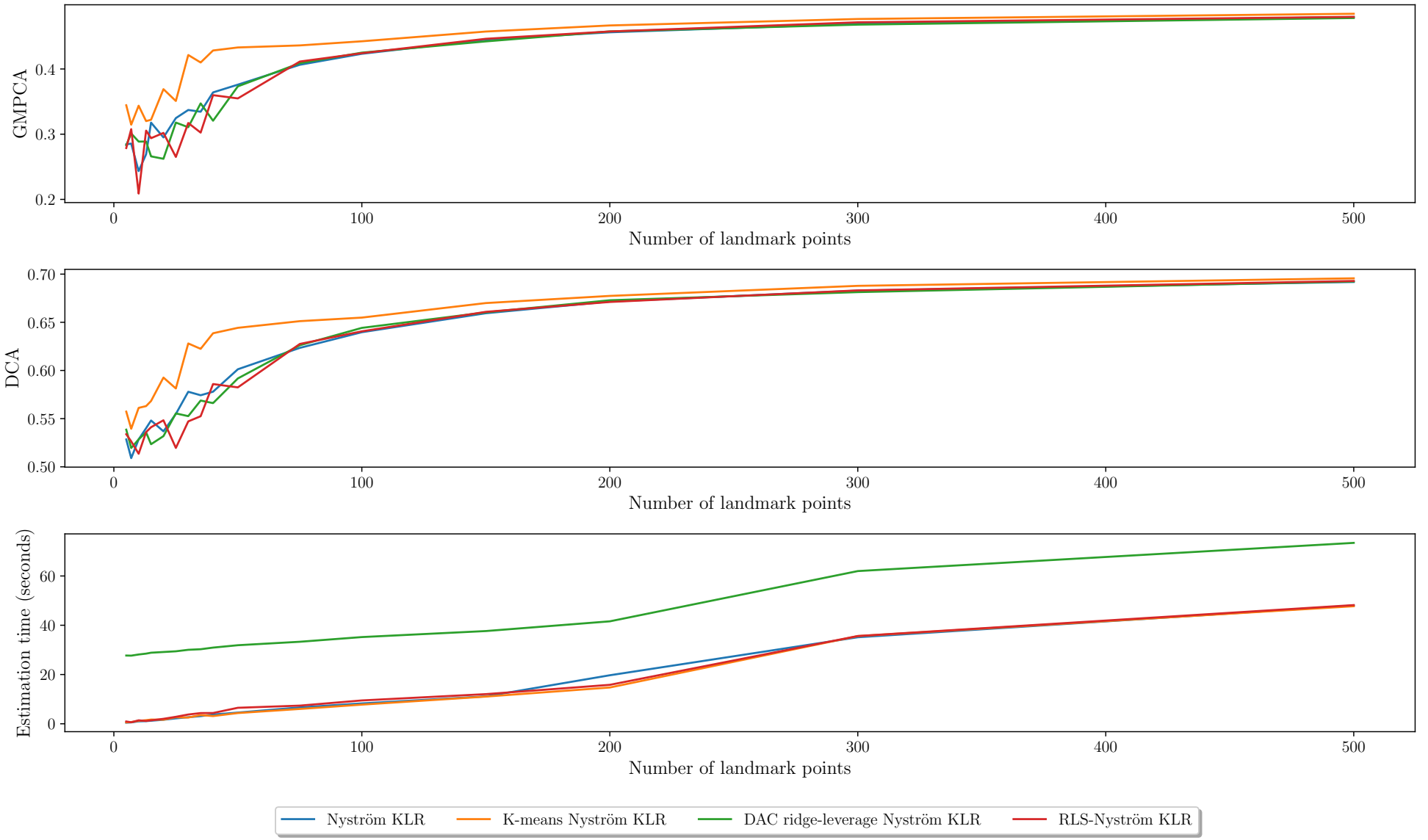


Recursive ridge-leverage strategy

RLS-Nyström KLR



Experiment 1: Comparison Nyström KLR methods



Experiment 2: Comparison Nyström KLR and ML

	LPMC			NTS		
	DCA	GMPCA	Estimation time (s)	DCA	GMPCA	Estimation time (s)
MNL	72.54	48.85	623.43	65.42	43.83	855.61
LinearSVM	72.13	48.92	691.21	64.64	43.72	3,963.52
RF	73.58	50.14	2.67	68.19	46.84	1.87
XGBoost	**74.71**	**51.85**	82.04	**68.72**	**48.05**	138.72
NN	73.87	50.72	5.25	68.40	47.12	7.51
Nyström KLR	73.45	50.41*	303.39	64.98	44.53*	776.46
k-means Nyström KLR	73.46	50.35	309.40	65.09*	44.50	719.25
DAC ridge-leverage Nyström KLR	73.49	50.33	507.37	64.91	44.41	1,010.26
RLS-Nyström KLR	73.62*	50.43	324.85	64.81	44.52	727.24

Memory usage

LPMC

22 GB

$L = 500$

NTS

194 GB

$L = 1000$

Memory usage

LPMC

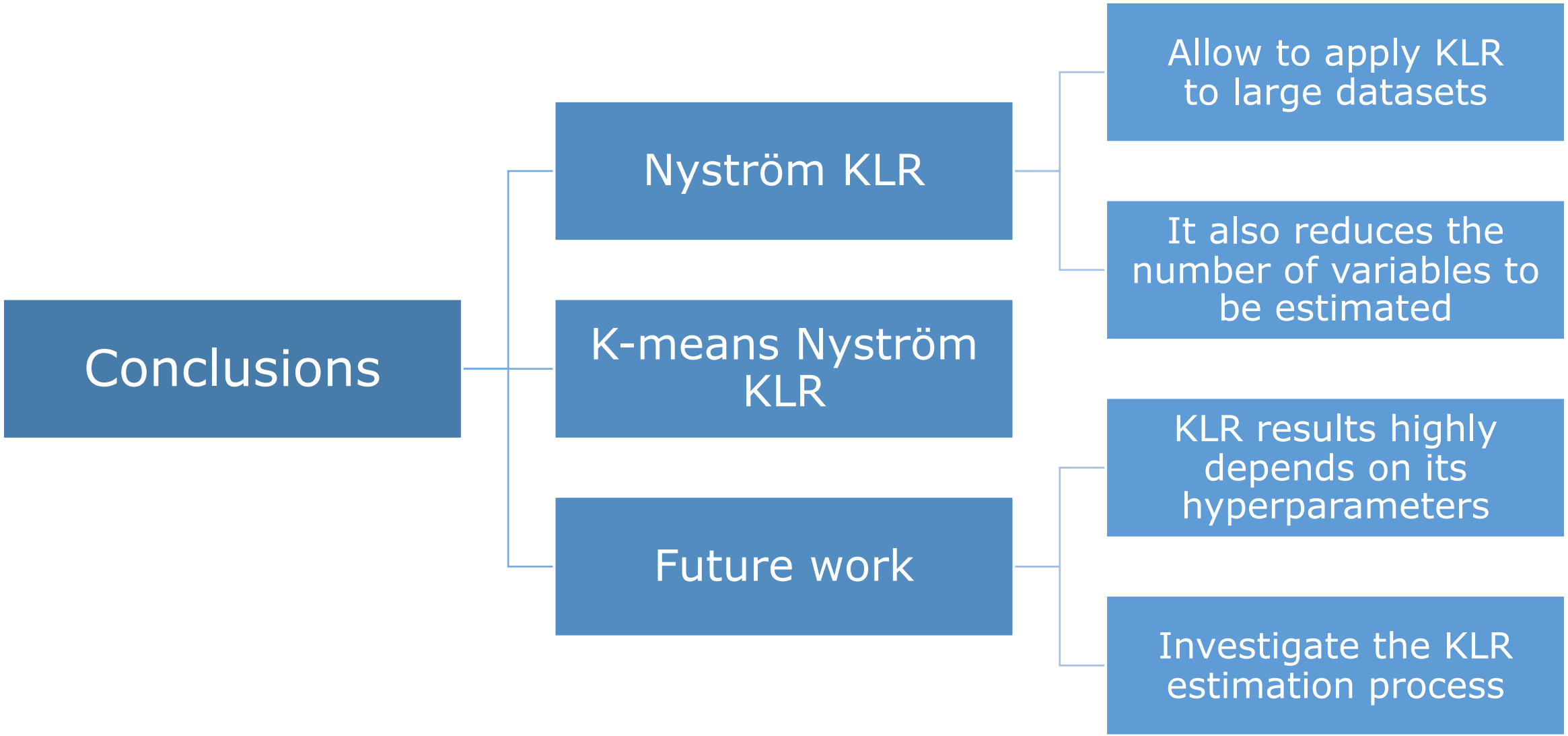
0.2 GB $\times$ **110**

$L = 500$

NTS

1.2 GB $\times$ **160**

$L = 1000$


Conclusions
Nyström KLR
K-means Nyström KLR
Future work
Allow to apply KLR to large datasets
It also reduces the number of variables to be estimated
KLR results highly depends on its hyperparameters
Investigate the KLR estimation process

Thanks for your attention!

For more information you can contact me at:

José Ángel Martín-Baos
Department of Mathematics
University of Castilla-La Mancha

joseangelmartin.com

JoseAngel.Martin@uclm.es

More info of this research:

Universidad de Castilla-La Mancha

World Conference on Transport Research

Escuela Superior de Informática

Universidad de Castilla-La Mancha

Nyström-based approximations for Kernel Logistic Regression: Application to transport choice modelling

José Ángel Martín-Baos

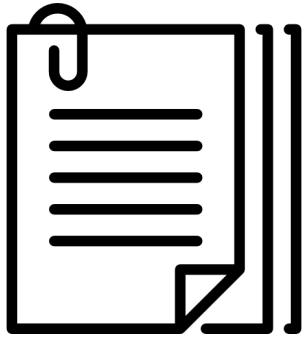
Ricardo García-Ródenas

Luis Rodriguez-Benitez

Michel Bierlaire

17-21 July 2023

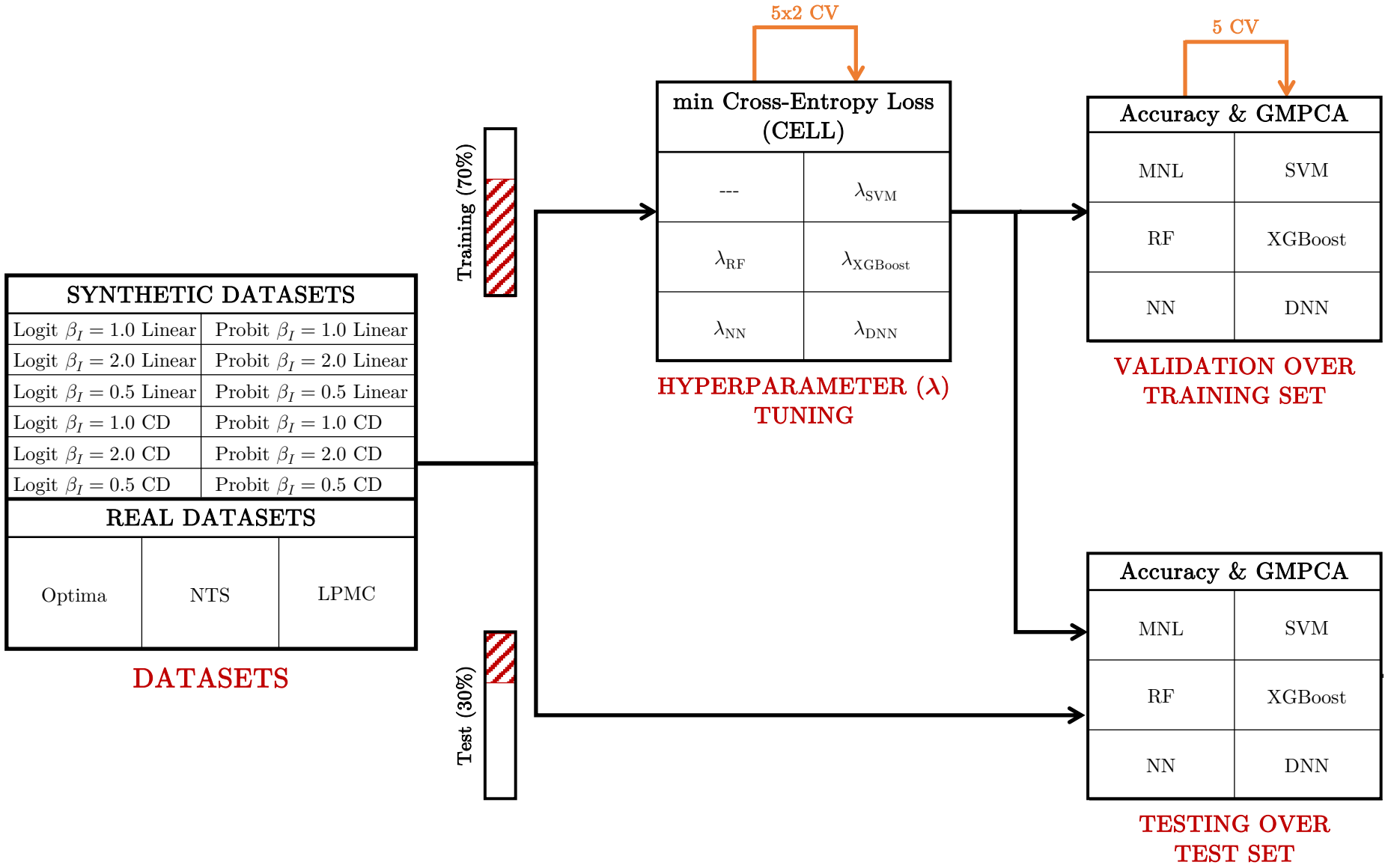
World Conference on Transport Research



Supplementary material

For the curious minds 😉

Hyperparameters tuning of ML models



Hyperparameters space of ML models

Technique $\mathcal{A}$	Name of the hyperparameter	Notation	Type	Search space	NTS	LPMC
LinearSVM	Cost (or soft margin constant)	C	Loguniform distribution	$[0.1, 10]$	2.704	6.380
RF	Number of decision trees	B	Uniform distribution	$[1, 200]$	153	180
	Max features for the best split	m	Uniform distribution	[2, N° features]	8	16
	Max depth of the tree	d	Uniform distribution	$[3, 10]$	10	10
	Min samples to be at a leaf node	l	Uniform distribution	$[1, 20]$	3	11
	Min samples to split an internal node	s	Uniform distribution	$[2, 20]$	15	14
	Goodnes of split metric	c	Choice	[Gini|Entropy]	Entropy	Entropy
XGBoost	Maximum tree depth	d	Uniform distribution	$[1, 14]$	7	7
	Minimum loss for a new split	γ	Loguniform distribution	$[10^{-4}, 5]$	4.970	4.137
	Minimum sum of instance weight needed in a child	w	Uniform distribution	$[1, 100]$	1	32
	Maximum delta step in each tree's weight	δ	Uniform distribution	$[0, 10]$	0	4
	Subsample ratio of the training instance	s	Uniform distribution	$[0.5, 1]$	0.823	0.935
	Subsample ratio of columns when constructing each tree	c_t	Uniform distribution	$[0.5, 1]$	0.553	0.679
	Subsample ratio of columns for each level	c_l	Uniform distribution	$[0.5, 1]$	0.540	0.629
	L1 regularisation term on weights	α	Loguniform distribution	$[10^{-4}, 10]$	0.028	0.003
	L2 regularisation term on weights	λ	Loguniform distribution	$[10^{-4}, 10]$	0.264	$0.5e^{-3}$
	Number of boosting rounds	B	Uniform distribution	$[1, 6000]$	4376	2789
NN	Number of neurons in hidden layer	n_1	Uniform distribution	$[10, 500]$	10	51
	Activation function	f	Choice	[tanh]	tanh	tanh
	Solver for weights optimisation	S	Choice	[LBFGS|SGD|Adam]	LBFGS	SGD
	Initial learning rate	η_0	Uniform distribution	$[10^{-4}, 1]$	0.416	0.041
	Learning rate schedule	η	Choice	[adaptive]	adaptive	adaptive
	Maximum number of iterations	t	Choice	$[10^6]$	10^6	10^6
	Batch Size	BS	Choice	[128|256|512|1024]	512	1024
	Tolerance for optimisation	tol	Choice	$[10^{-3}]$	10^{-3}	10^{-3}
KLR	Kernel function	K	Choice	[RBF]	RBF	RBF
	Parameter of the Gaussian function	γ	Loguniform distribution	$[10^{-3}, 10^{-1}]$	0.037	0.054
	Tikhonov penalization parameter	λ	Fixed	10^{-6}	10^{-6}	10^{-6}

(Penalised) Maximum likelihood estimation

$$\mathcal{L}(\boldsymbol{\alpha}) = \prod_{n=1}^{N} \prod_{i=1}^{I} \mathbb{P}(\, i \mid \mathbf{x}_n, \boldsymbol{\alpha}_i \,)^{y_{in}}$$

$$\log \mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \sum_{i=1}^{I} y_{in} \log \mathbb{P}(\, i \mid \mathbf{x}_n, \boldsymbol{\alpha}_i \,)$$

$$\max_{\boldsymbol{\alpha}} \; \log \mathcal{L}(\boldsymbol{\alpha}) \; - \; \lambda \Omega(\boldsymbol{\alpha})$$

Goodness of fit

Penalisation term

Algorithm 1: Line search method

Input : The total number of iterations T to be performed
The hyperparameters of the optimisation method
Output: The parameter vector $\boldsymbol{\omega}_{T+1}$ of the optimised model

1 Choose an initial guess $\boldsymbol{\omega}_1$
2 **for** $t = 1, 2, \ldots T$ **do**
3 Determine the search direction $g(\boldsymbol{\omega}_t)$
4 Choose a learning rate $\alpha_t > 0$
5 Update the parameter vector as $\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \alpha_t g(\boldsymbol{\omega}_t)$

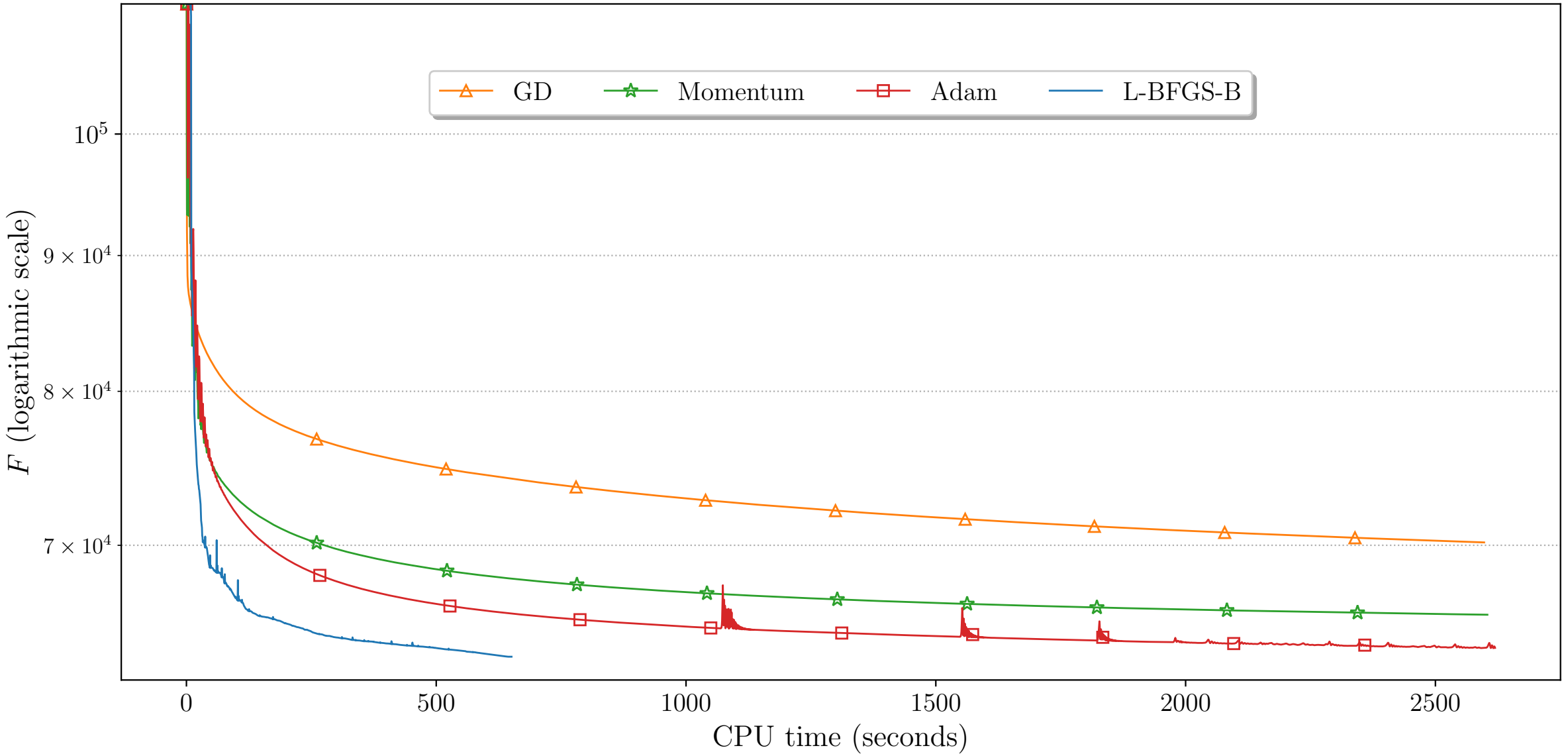
GD:
$g(\boldsymbol{\omega}_t) = -\nabla F(\boldsymbol{w}_t)$

Quasi-Newton:
$g(\boldsymbol{\omega}_t) = -H_t \nabla F(\boldsymbol{w}_t)$

Newton:
$g(\boldsymbol{\omega}_t) = -[\nabla^2 F(\boldsymbol{w}_t)]^{-1} \nabla F(\boldsymbol{w}_t)$

Comparison optimisation techniques

NTS



MNL model considered for the experiments

The MNL is going to be used as the baseline model for this experiment. We have considered linear utility functions for each dataset. For the LPMC dataset we have defined an utility function where all the features were selected as individual specific except for the following features that were selected as alternative specific attributes:

- Walk: *distance* and *dur_walking*.
- Bike: *distance* and *dur_cycling*.
- Public transport: *cost_transit*, *dur_pt_access*, *dur_pt_rail*, *dur_pt_bus*, *dur_pt_int_waiting*, *dur_pt_int_walking*, and *pt_n_interchanges*.
- Car: *cost_driving_total* and *dur_driving*.

For the NTS dataset linear utilities specified over all the attributes have been considered, using different parameters for each alternative.

Results for the LPMC dataset