Adapting kernel representations online using submodular maximization

Martha White
Assistant Professor
Department of Computer Science
Indiana University
(…soon to be at the University of Alberta)

Motivation

Motivation

- **Goal**: predict target **y** given observations **x**

Motivation

- **Goal**: predict target **y** given observations **x**
- Target is a nonlinear function of observations

Motivation

- **Goal**: predict target **y** given observations **x**
- Target is a nonlinear function of observations
- **Strategy**: Obtain transformation (representation) of observation to learn nonlinear functions

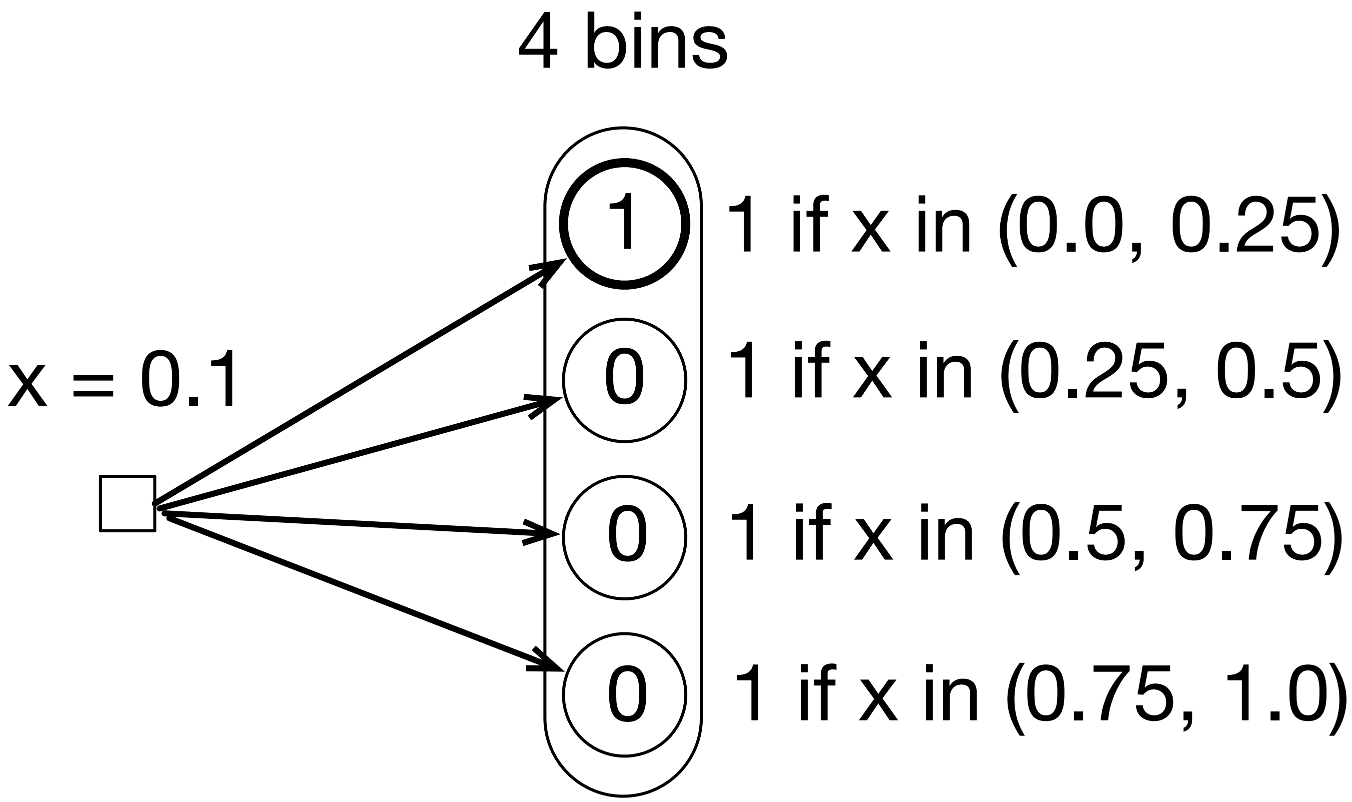
Motivation

- **Goal**: predict target **y** given observations **x**
- Target is a nonlinear function of observations
- **Strategy**: Obtain transformation (representation) of observation to learn nonlinear functions

$$f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w} = \sum_{i=1}^{b} \boldsymbol{\phi}(\mathbf{x})_i \mathbf{w}_i, \qquad f(\mathbf{x}) \approx y$$

$$\mathbf{x} \in \mathbb{R}^d, \quad \boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^b, \quad \mathbf{w} \in \mathbb{R}^b$$

Kernel representation

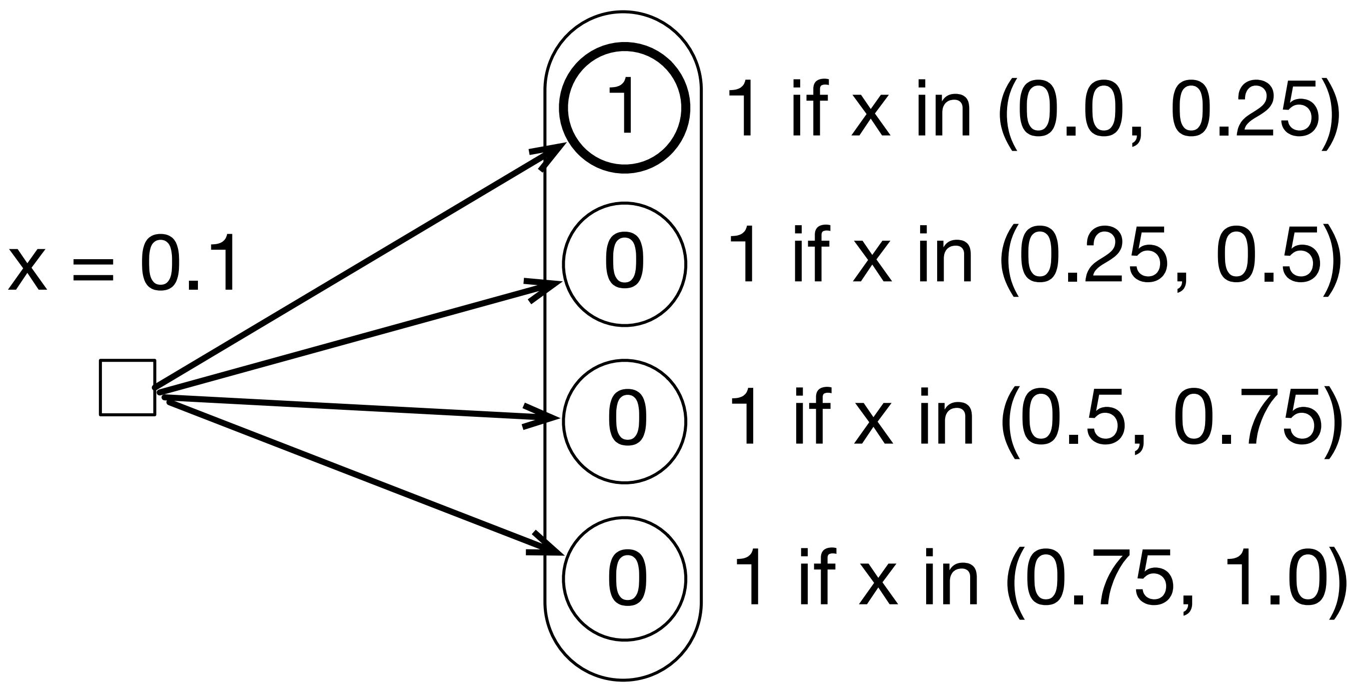


$$f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}$$
$$= \sum_{i=1}^{b} \boldsymbol{\phi}(\mathbf{x})_i \mathbf{w}_i$$

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{z}_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{z}_b) \end{bmatrix} \in \mathbb{R}^b$$

Kernel representation

4 bins

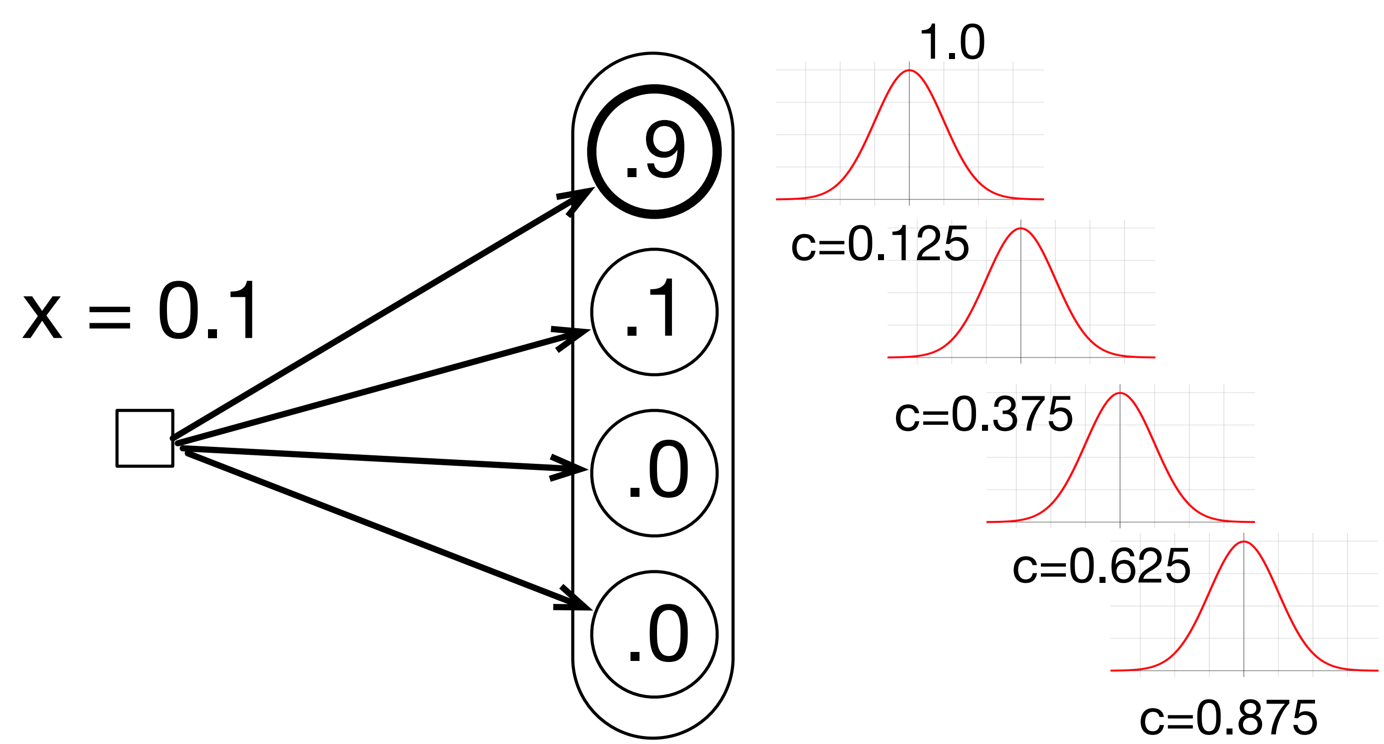


$$f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}$$

$$= \sum_{i=1}^{b} \phi(\mathbf{x})_i \mathbf{w}_i$$

$$\phi(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{z}_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{z}_b) \end{bmatrix} \in \mathbb{R}^b$$

4 radial basis functions



Example: Matching similarity for categorical data

$$\mathbf{x} = \begin{array}{l} \text{age} \\ \text{gender} \\ \text{income} \\ \text{education} \end{array}$$

age	{15-24, 25-34, …, 65+}
gender	{F, M}
income	{Low, Medium, High}
education	{Bachelors, Trade-Sch, High-Sch, …}

Census dataset: Predict hours worked per week

Example: Matching similarity for categorical data

$$\mathbf{x} = \begin{bmatrix} \text{age} \\ \text{gender} \\ \text{income} \\ \text{education} \end{bmatrix}$$

Example: Matching similarity for categorical data

$$\mathbf{x} = \begin{bmatrix} \text{age} \\ \text{gender} \\ \text{income} \\ \text{education} \end{bmatrix}$$

Example: Matching similarity for categorical data

$$\mathbf{x} = \begin{bmatrix} \text{age} \\ \text{gender} \\ \text{income} \\ \text{education} \end{bmatrix}$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = k\left(\begin{bmatrix} \text{24-34} \\ \text{F} \\ \text{Medium} \\ \text{Trade-Sch} \end{bmatrix}, \begin{bmatrix} \text{35-44} \\ \text{F} \\ \text{Medium} \\ \text{Bachelors} \end{bmatrix} \right) = 0.5$$

Why kernel representations?

- Many specialized kernels (similarity measures)
  - convolutional kernels for images
  - string kernel for text and gene analysis
- Universal function approximation capabilities
  - but simple linear estimation techniques, given prototypes
- Intuitive and interpretable solution

Improving optimization for kernels is key

- Widespread use seems limited
    - unlike (for example) neural networks
- Need to investigate effective optimization principles and heuristics to make kernels easy-to-use
    - Automatically and efficiently selecting prototypes
    - Automatically selecting kernels and kernel parameters

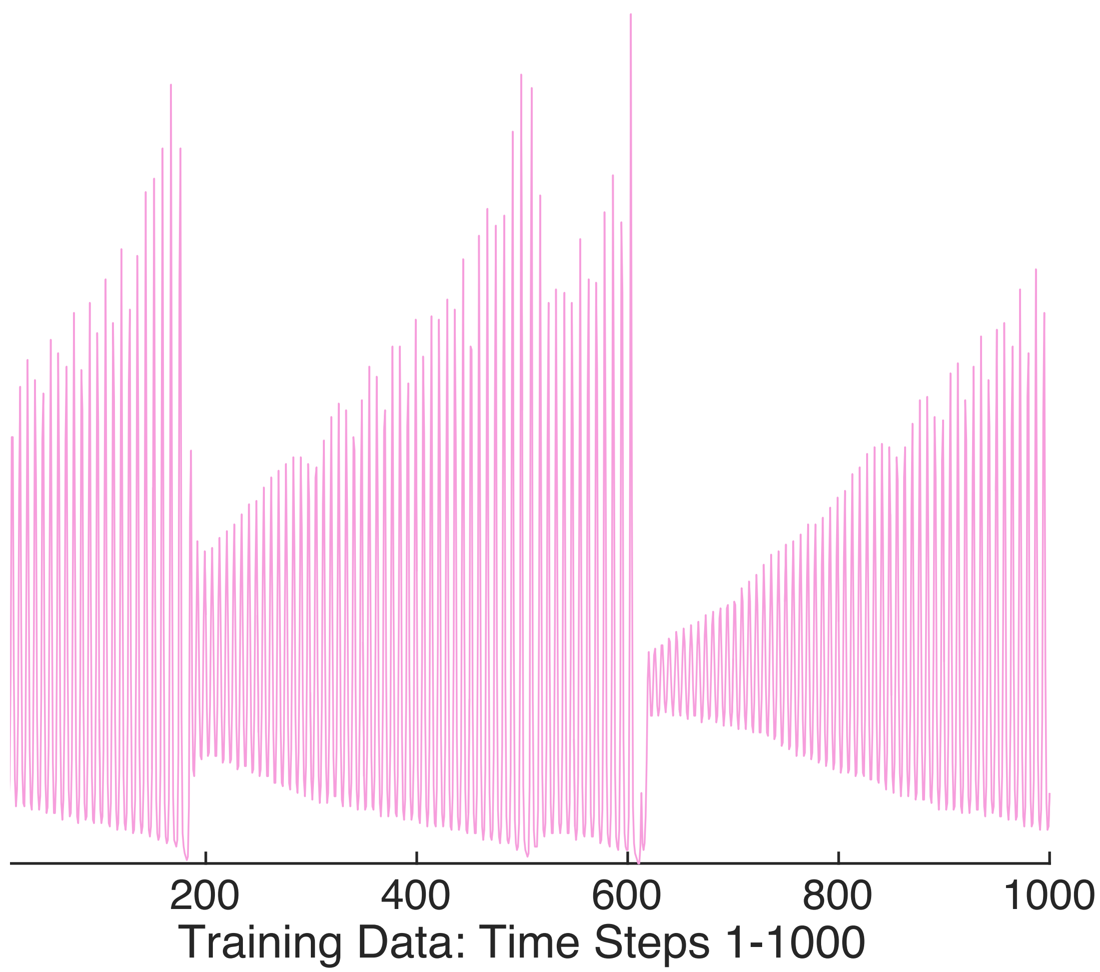
Continual learning setting

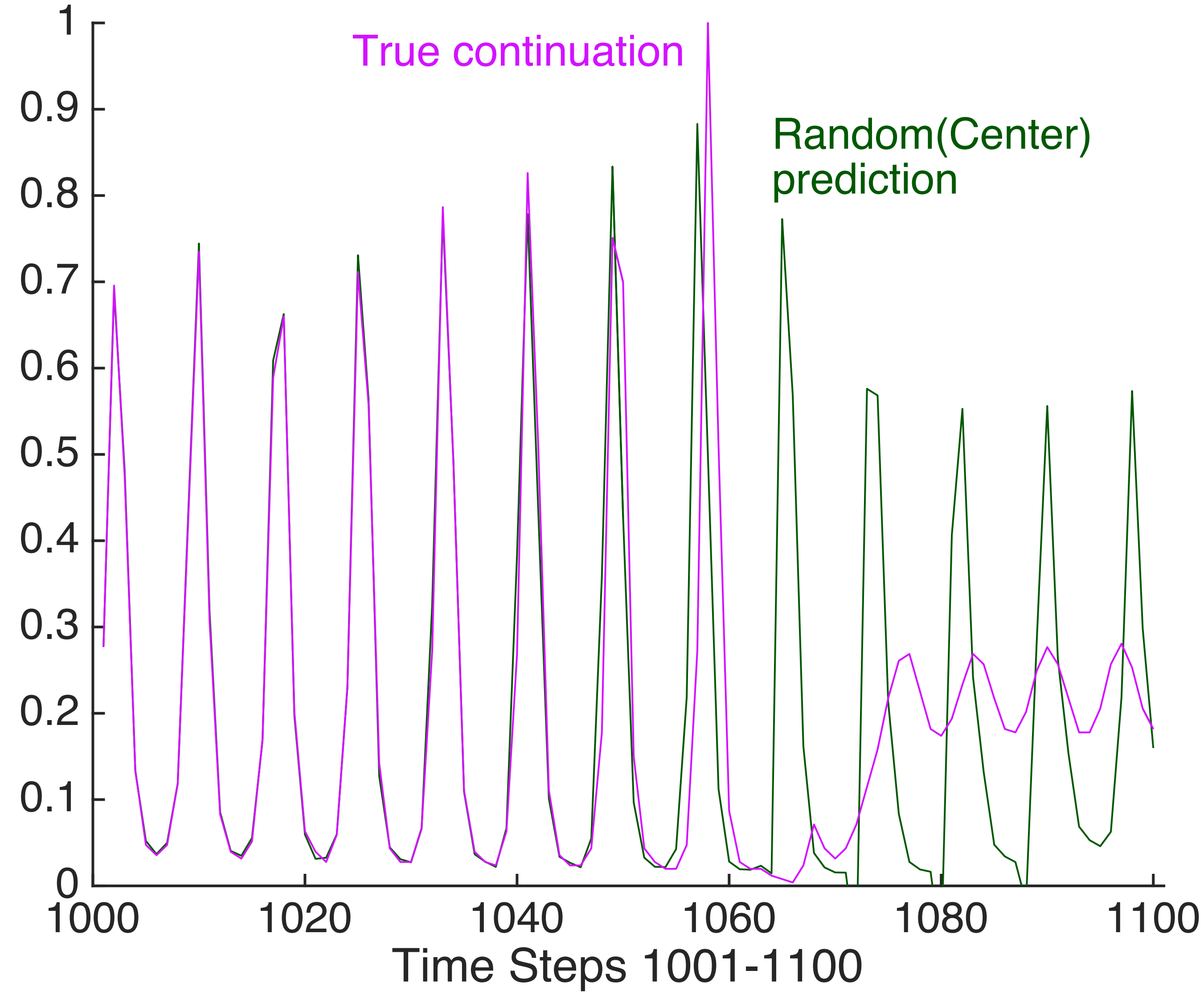
- Modern setting
  - Constant streams of data collected by companies
  - Agent interacting with environment in reinforcement learning or online learning
- Requires efficient per-step updating for real-time computation — linear in the number of prototypes

Why linear in the number of prototypes?

- For sufficient complexity, need many prototypes
  - similar to enabling large hidden layers
- Consider differences between b and b^2
  - b = 1k —> b^2 = 1 million
  - b = 10k —> b^2 = 100 million

Why do we need careful selection of prototypes?

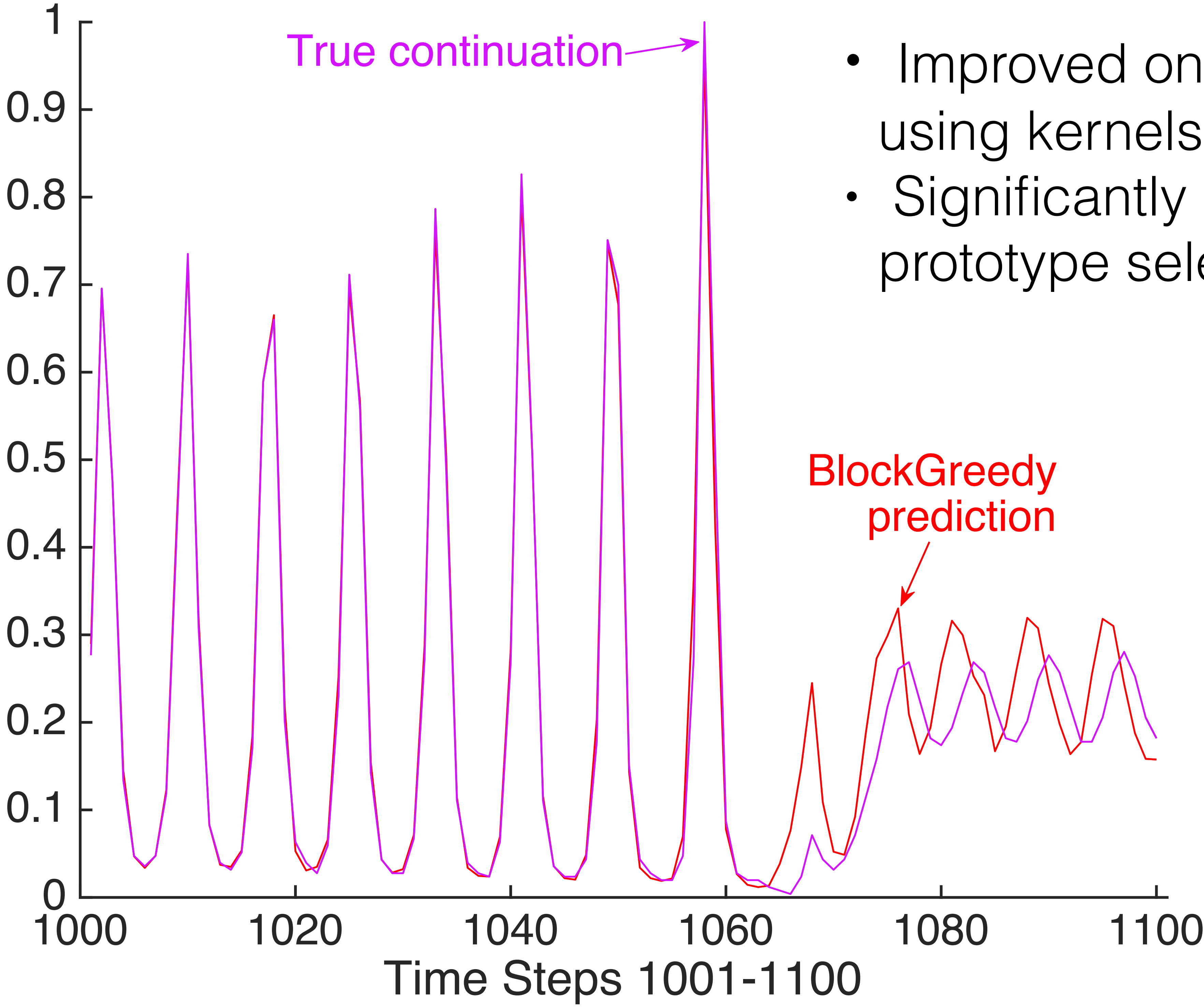




Cannot predict intensity collapse event

b = 300

Example setting: Time series



- Improved on previous results using kernels
- Significantly more efficient prototype selection

Talk outline

- Problem formulation for selecting prototypes
- Using submodular maximization to solve this problem for continual setting
    - prove that simple, easy-to-use algorithm is effective
- Experiments demonstrating
    - approximation quality of our algorithm
    - efficacy of selected prototypes for prediction

Our focus

- Select prototypes $\mathbf{z}_1, \ldots, \mathbf{z}_b \in \mathbb{R}^d$

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{z}_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{z}_b) \end{bmatrix} \in \mathbb{R}^b$$

Goal

- efficient, easy-to-use algorithm

How do we pick prototypes?

- This topic has been widely explored
  - unsupervised: active-set selection, facility location, k-medians, k-mediods, k-means
  - supervised: sparse GPs, specialized methods for classification
- We revisit the criteria for continual learning

How do we pick prototypes?

- This topic has been widely explored
  - unsupervised: active-set selection, facility location, k-medians, k-mediods, k-means
  - supervised: sparse GPs, specialized methods for classification
- We revisit the criteria for continual learning

Minimize distance to the function that uses all the instances as prototypes

Criteria

$$\min_{\substack{S \subset \mathcal{X} \\ |S| = b}} \min_{\mathbf{w} \in \mathbb{R}^b} \|f - f_{S,\mathbf{w}}\|^2$$

Finite set $\mathcal{X}$: $\quad f(\mathbf{x}) = \sum_{\mathbf{z}_i \in \mathcal{X}} \alpha_i k(\mathbf{x}, \mathbf{z}_i)$

$$f_{S,\mathbf{w}}(\mathbf{x}) = \sum_{\mathbf{z}_i \in S} \mathbf{w}_i k(\mathbf{x}, \mathbf{z}_i)$$

Criteria

$$\min_{\substack{S\subset\mathcal{X}\\|S|=b}} \min_{\mathbf{w}\in\mathbb{R}^b} \|f - f_{S,\mathbf{w}}\|^2$$

Finite set $\mathcal{X}$: $$f(\mathbf{x}) = \sum_{\mathbf{z}_i\in\mathcal{X}} \alpha_i k(\mathbf{x},\mathbf{z}_i)$$

$$f_{S,\mathbf{w}}(\mathbf{x}) = \sum_{\mathbf{z}_i\in S} \mathbf{w}_i k(\mathbf{x},\mathbf{z}_i)$$

Obtain a generalized coherence criterion that is an upper bound on this objective

An instance of this criterion

Unsupervised measure preferring diverse prototypes

$$g(S) = \log\det(\mathbf{K}_S + \mathbf{I}) \qquad \mathbf{K}_S(i,j) = k(\mathbf{z}_i, \mathbf{z}_j)$$

$$= \sum_{i=1}^{b} \log(1 + \lambda_i) \qquad \mathbf{K}_S = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top$$

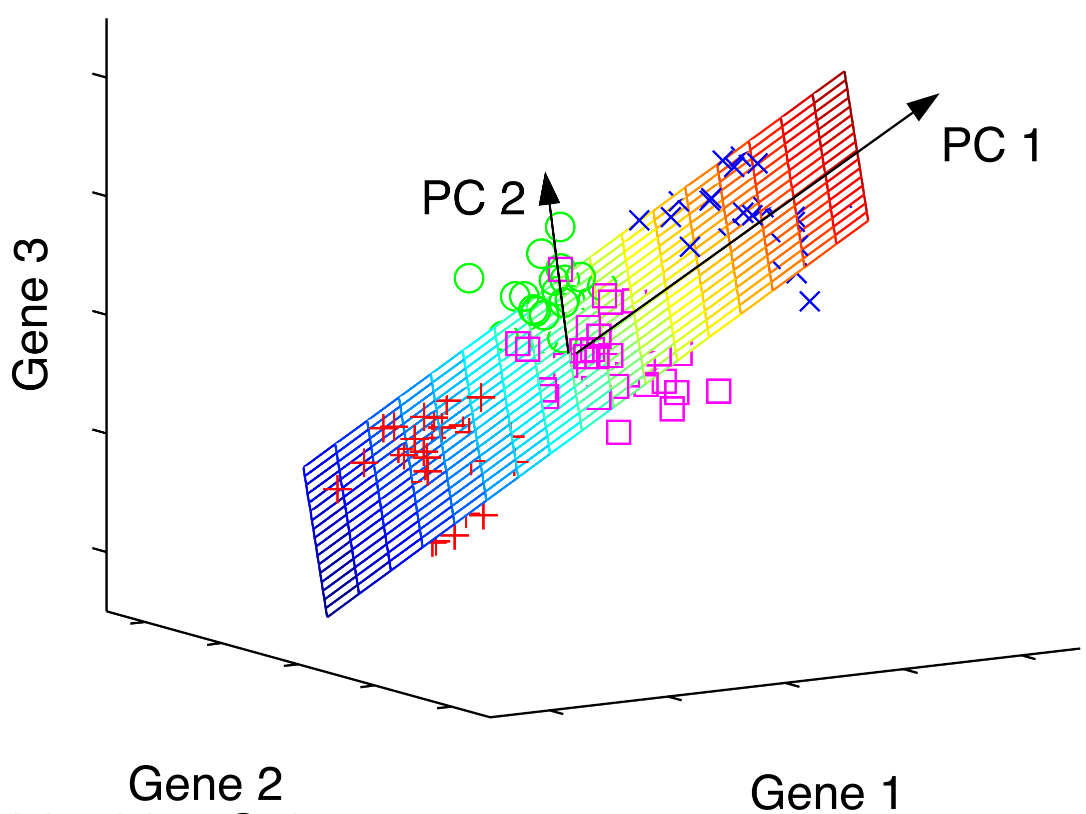
An instance of this criterion

Unsupervised measure preferring diverse prototypes

$$g(S) = \log\det(\mathbf{K}_S + \mathbf{I}) \qquad \mathbf{K}_S(i,j) = k(\mathbf{z}_i, \mathbf{z}_j)$$

$$= \sum_{i=1}^{b} \log(1 + \lambda_i) \qquad \mathbf{K}_S = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$$

If λ_3 small



*from Matthias Scholz

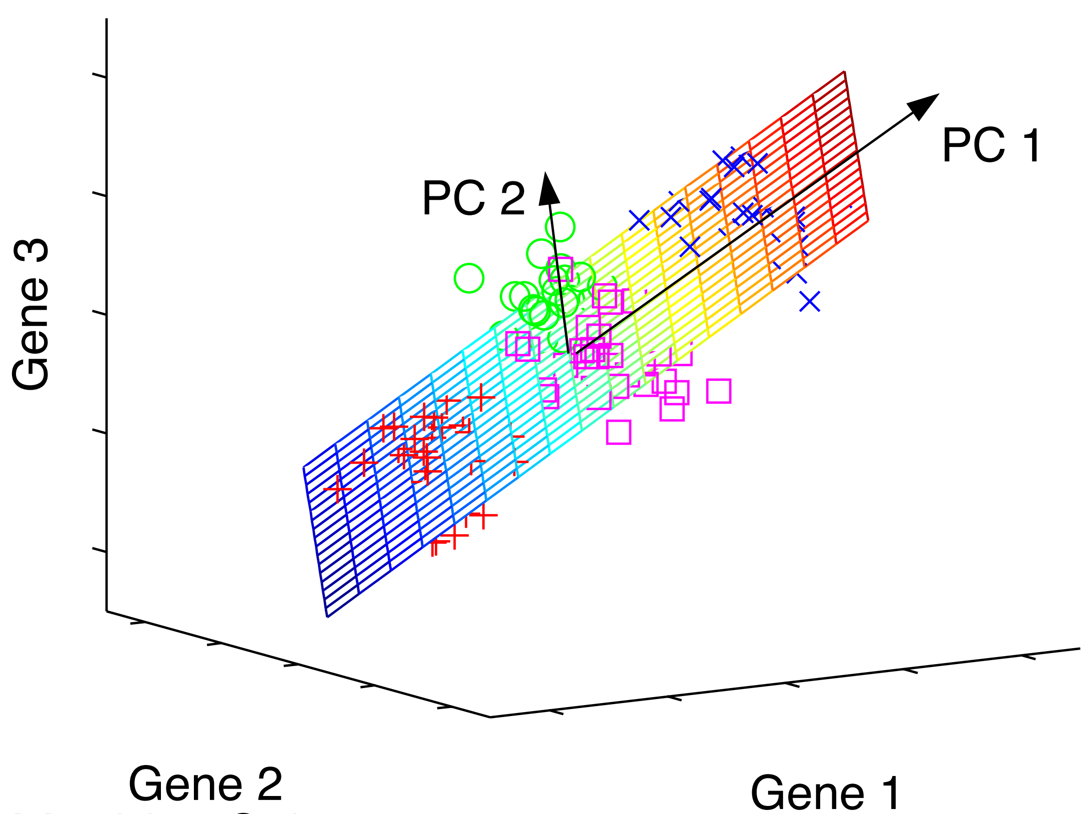
An instance of this criterion

Unsupervised measure preferring diverse prototypes

$$g(S) = \log\det(\mathbf{K}_S + \mathbf{I}) \qquad \mathbf{K}_S(i,j) = k(\mathbf{z}_i, \mathbf{z}_j)$$

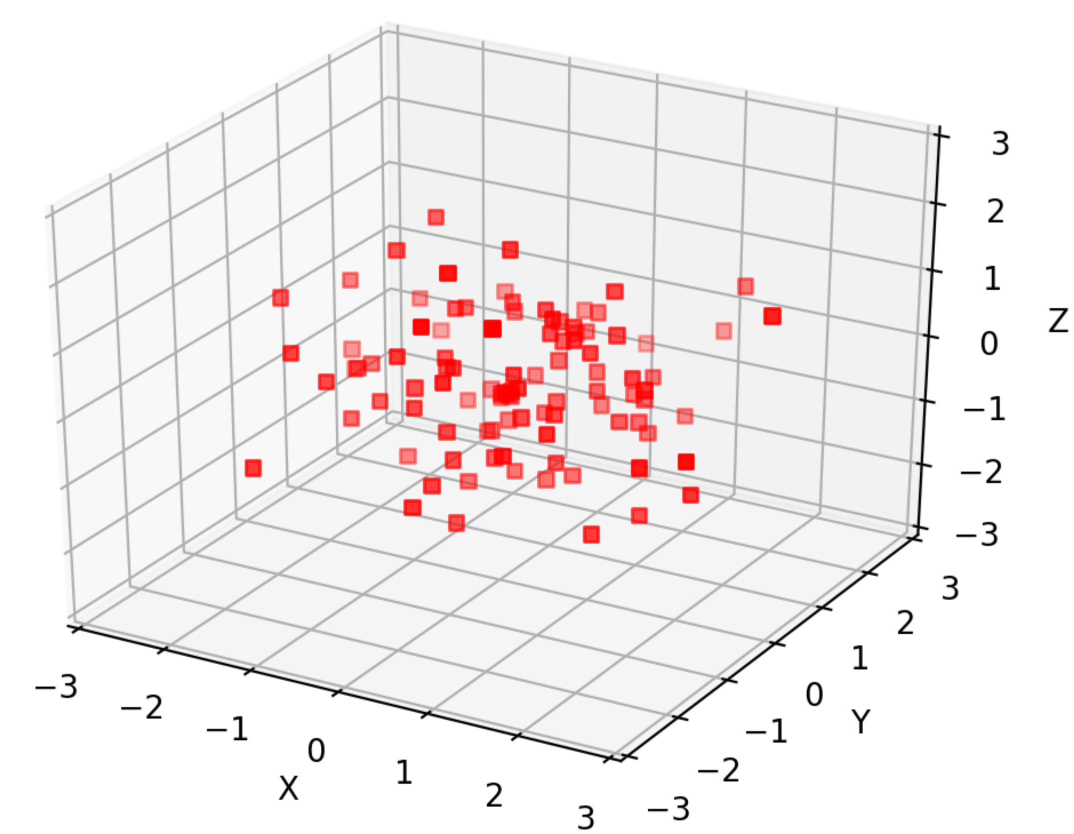
$$= \sum_{i=1}^{b} \log(1 + \lambda_i) \qquad \mathbf{K}_S = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$$

If λ_3 small



*from Matthias Scholz

Larger if all λ_i larger



Our focus for experiments

Unsupervised measure preferring diverse prototypes

$$g(S) = \log\det(\mathbf{K}_S + \mathbf{I})$$
$$= \sum_{i=1}^{b} \log(1 + \lambda_i)$$

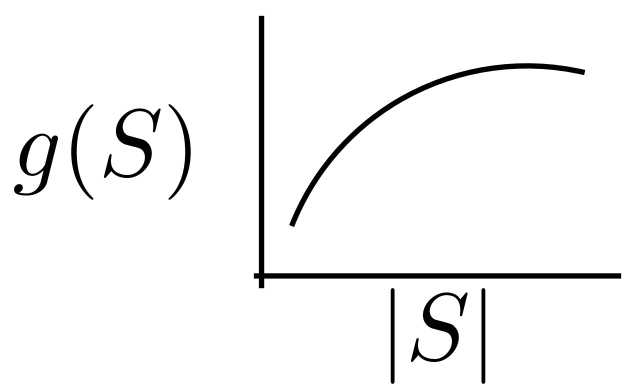
$$\mathbf{K}_S = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top$$

Goal: $$\max_{\substack{S \subset \mathcal{X} \\ |S| = b}} g(S)$$

How do we solve this optimization problem?

- Submodular-set functions g(S) have diminishing returns

$$T \subset S \implies g(S \cup \{\mathbf{z}\}) - g(S) \leq g(T \cup \{\mathbf{z}\}) - g(T)$$



- Greedy maximization algorithms effective for submodular functions $\max_{\substack{S \subset \mathcal{X} \\ |S| = b}} g(S)$

Greedy algorithm

- For a finite set, greedily select the best point, add to set S until reach budget size *b*

$$\arg\max_{\mathbf{z}\in\mathcal{X}\setminus S} g(S \cup \{\mathbf{z}\})$$

- Good approximation ratio for simple greedy algorithm
    - ratio to optimal solution is 1 - 1/e = 0.6321
- Incremental (streaming) versions of this algorithm
    - but requires multiple passes of the dataset

OnlineGreedy

$S_0 \leftarrow \emptyset$

for $t = 1 : b$ **do** $S_t \leftarrow S_{t-1} \cup \{\mathbf{x}_t\}$

while interacting, $t = b + 1, \ldots$ **do**

$$\mathbf{z}' = \operatorname*{argmax}_{\mathbf{z} \in S_{t-1}} g(S_{t-1} \backslash \{\mathbf{z}\} \cup \{\mathbf{x}_t\})$$

$$S_t \leftarrow S_{t-1} \backslash \{\mathbf{z}'\} \cup \{\mathbf{x}_t\}$$

if $g(S_t) - g(S_{t-1}) < \epsilon_t$ **then**

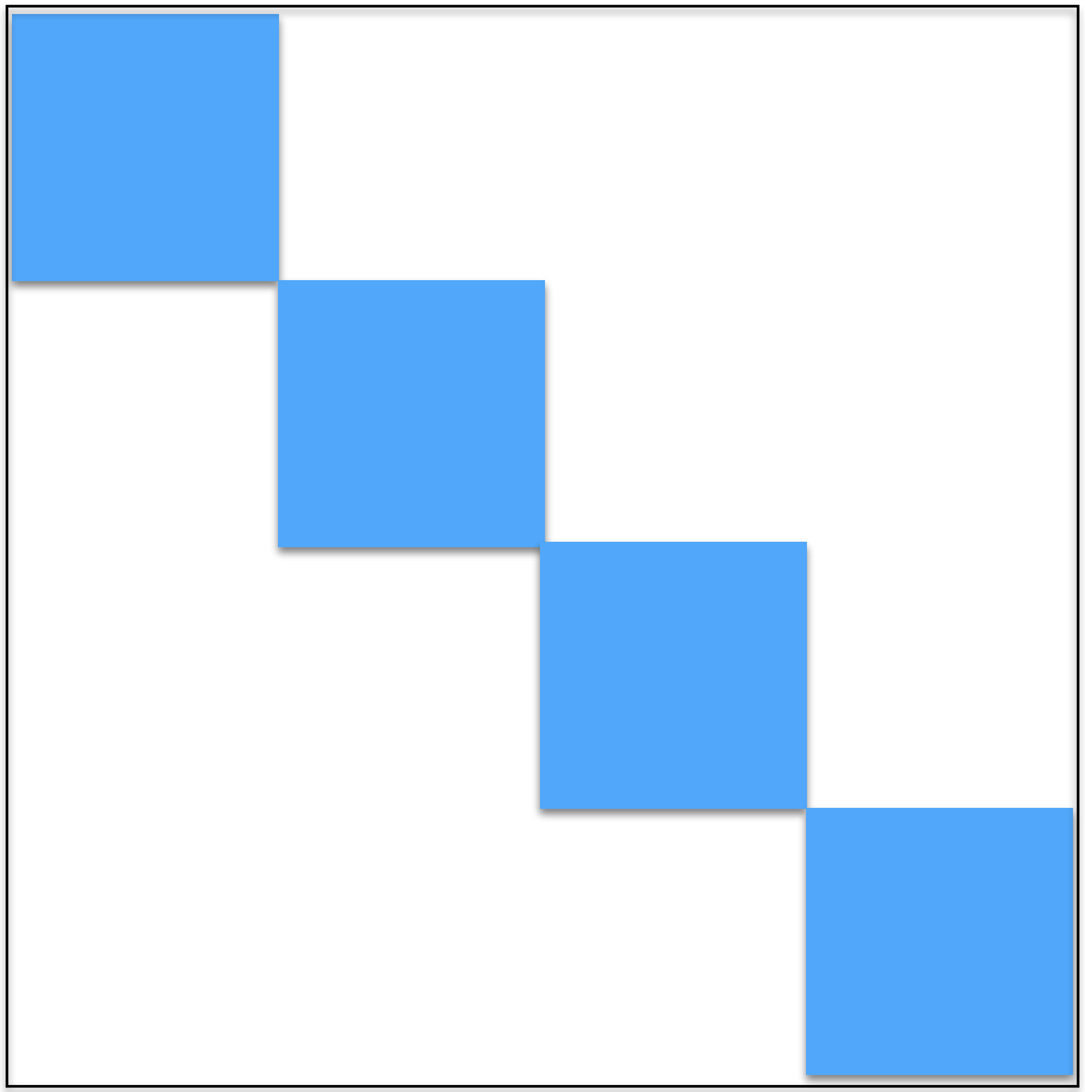
$$S_t \leftarrow S_{t-1}$$

Approximation ratio of about 1/2

Efficient implementation

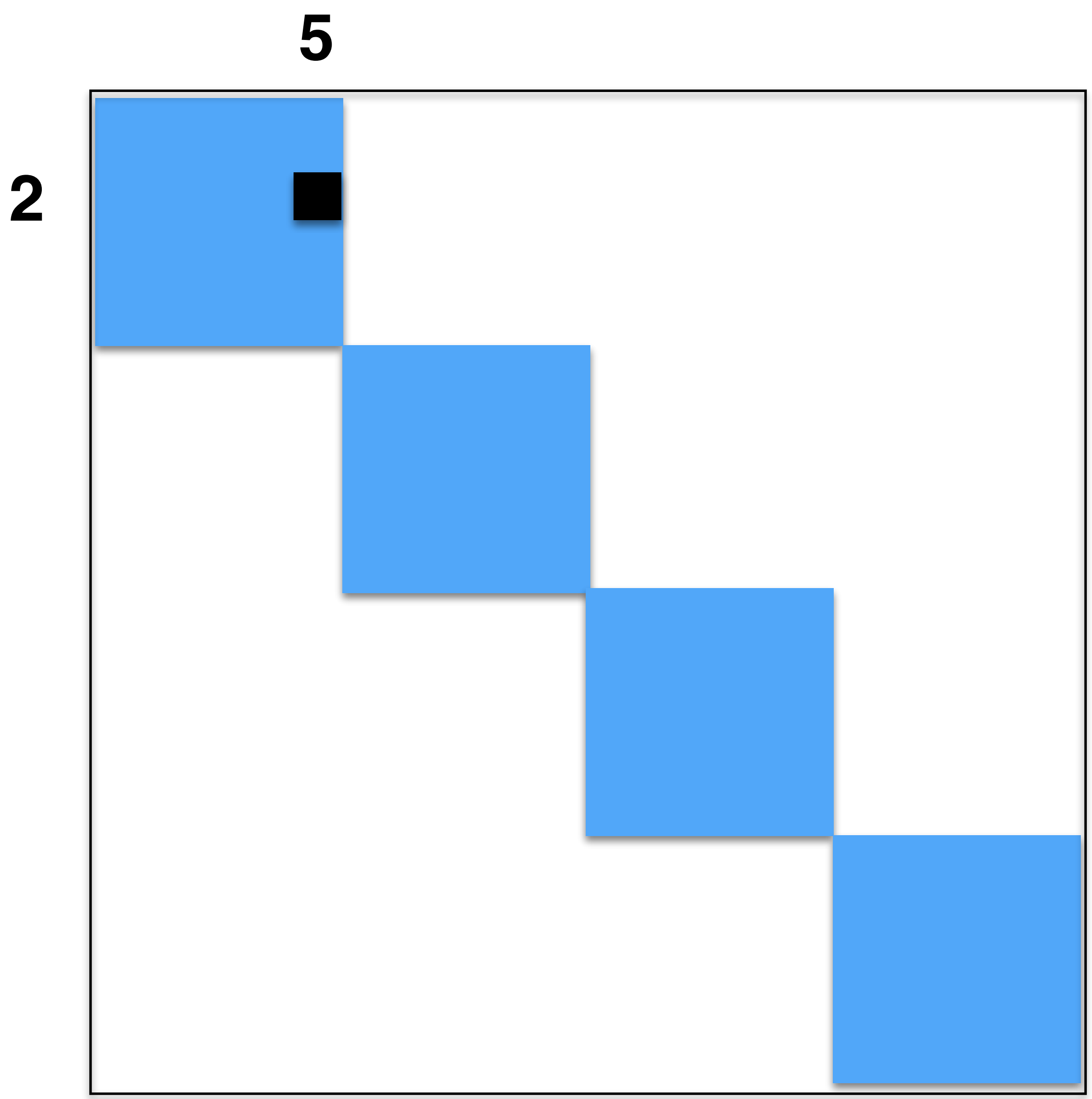
- Computation of g is the bottleneck
  - $O(b^3)$ per step for exact computation!
- Exploit block-diagonal structure of the kernel matrix to get a highly accurate approximation
  - reduce computation to $O(b)$ per step
  - theory allows some inaccuracy in $g(S)$

Block-diagonal matrix



$$\mathbf{K}_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$$

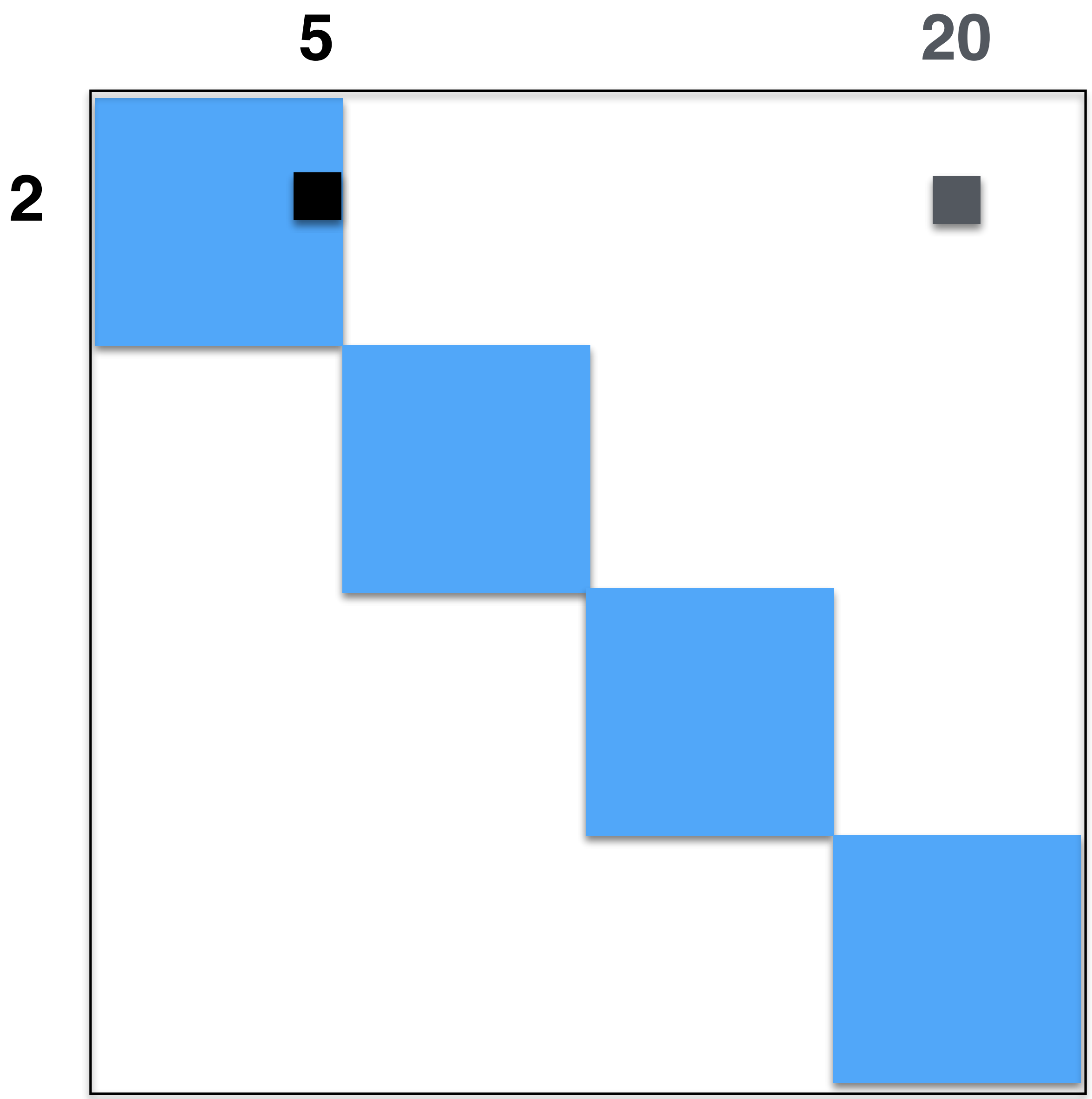
Block-diagonal matrix



k(z_2, z_5) larger

$$\mathbf{K}_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$$

Block-diagonal matrix

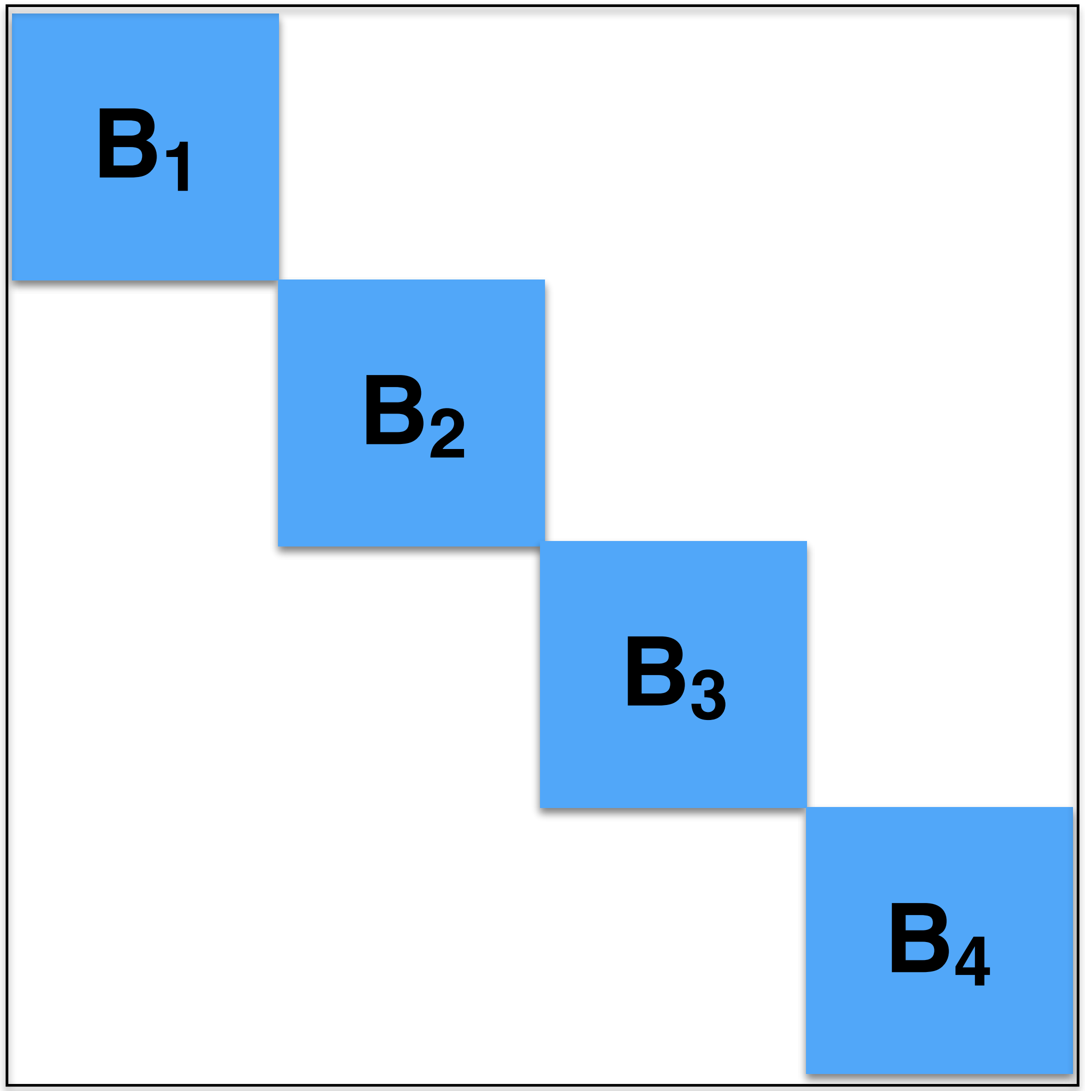


k(z_2, z_5) larger

k(z_2, z_{20}) small

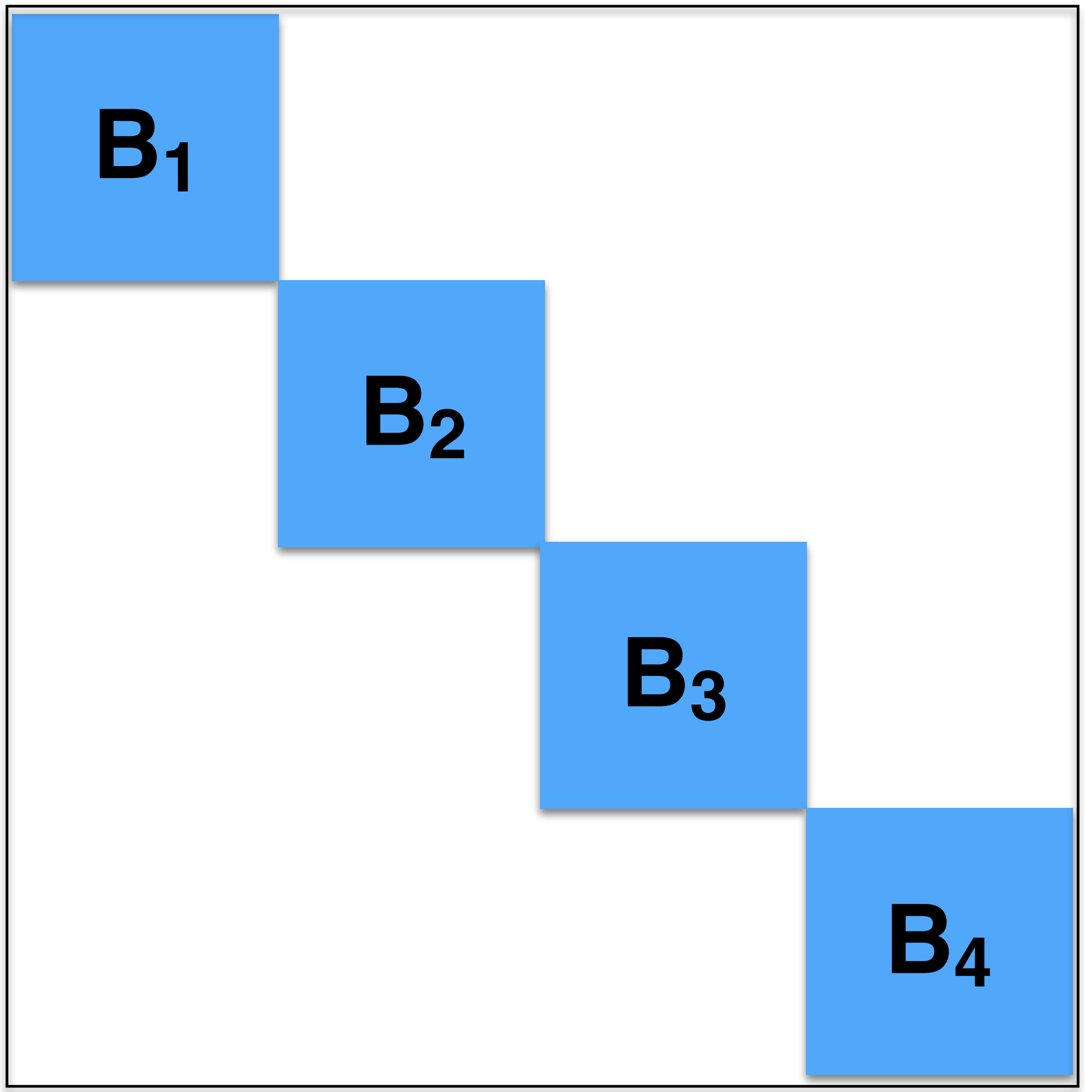
$$\mathbf{K}_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$$

Block-diagonal matrix



$$\mathbf{K}_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$$

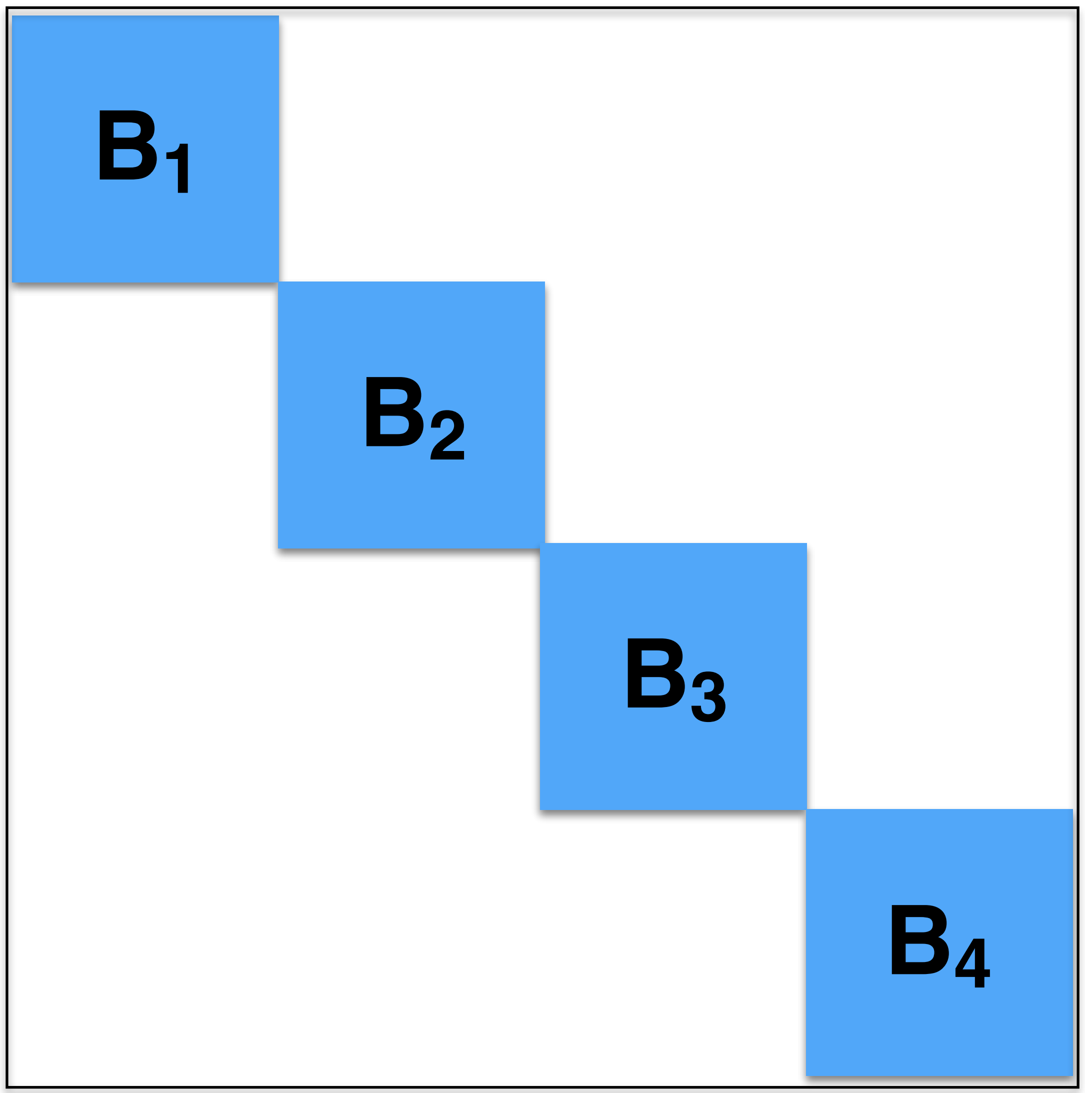
Block-diagonal matrix



$$\log \det(\mathbf{K}) = \sum_{i=1}^{4} \log \det(\mathbf{B}_i)$$

$$\mathbf{K}_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$$

Block-diagonal matrix



$$\mathbf{K}_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$$

$$\log\det(\mathbf{K}) = \sum_{i=1}^{4} \log\det(\mathbf{B}_i)$$

$$\mathrm{tr}(\mathbf{K}^{-1}) = \sum_{i=1}^{4} \mathrm{tr}(\mathbf{B}_i^{-1})$$

Algorithmic take-away

- Principled selection with approximation guarantees
  - OnlineGreedy for submodular maximization
- Efficient — linear in number of prototypes
  - taking advantage of block-diagonal structure of the kernel matrix
- Easy-to-use approach
  - meta-parameters include threshold and block-size

Experiments

- Investigated efficacy of algorithm with log-det
- How effectively are prototypes selected in terms of maximizing the log-det?
- How accurate is the block approximation?
- What are the runtime improvements?
- How accurate is the regression performance?

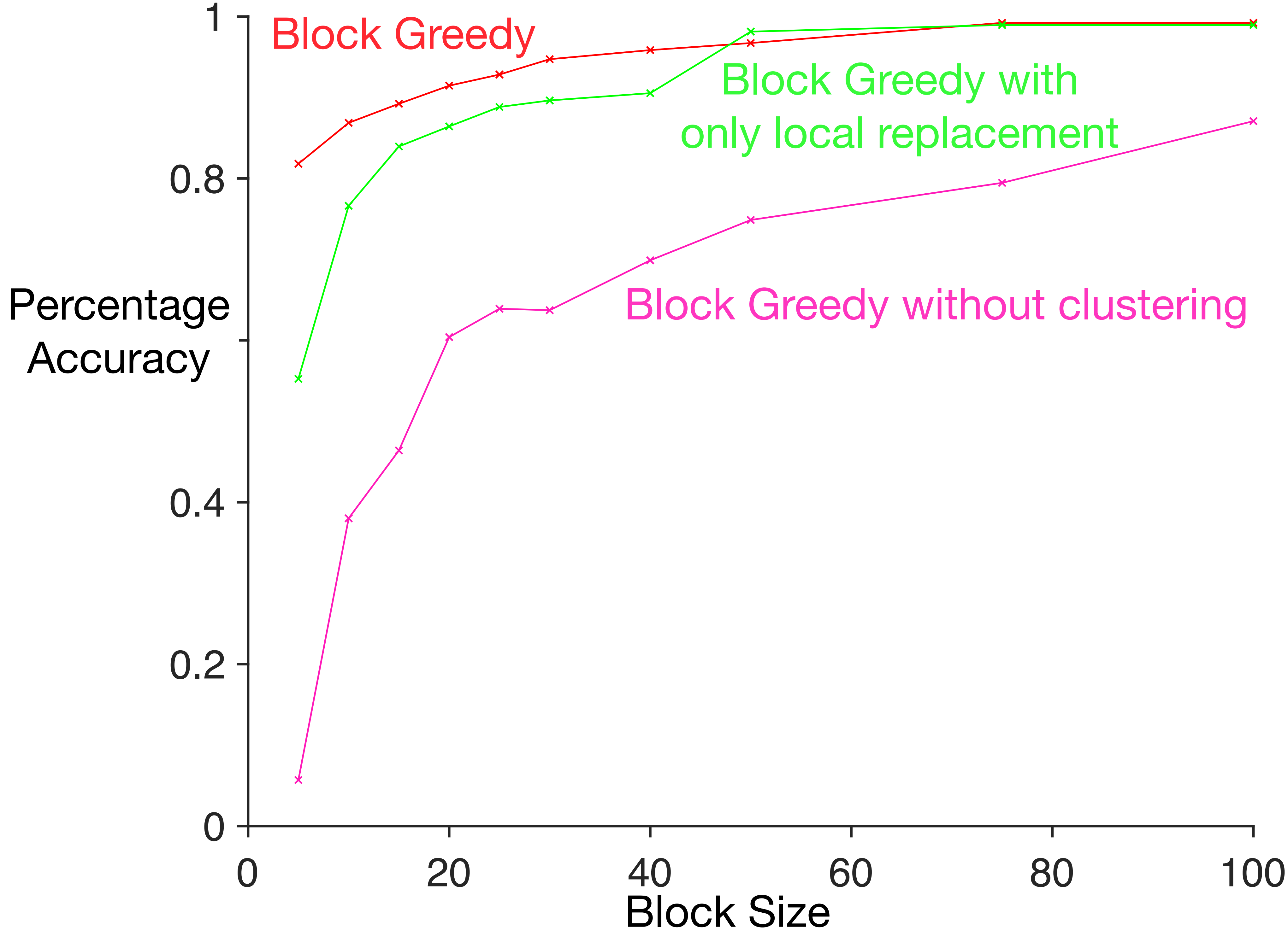
Datasets

- Two simpler datasets used previously for streaming prototype selection
  - Boston housing — 13 features
  - Parkinsons Telemonitoring — 25 features
- Santa Fe A — a benchmark time series dataset
- Census — a large dataset, with categorial features

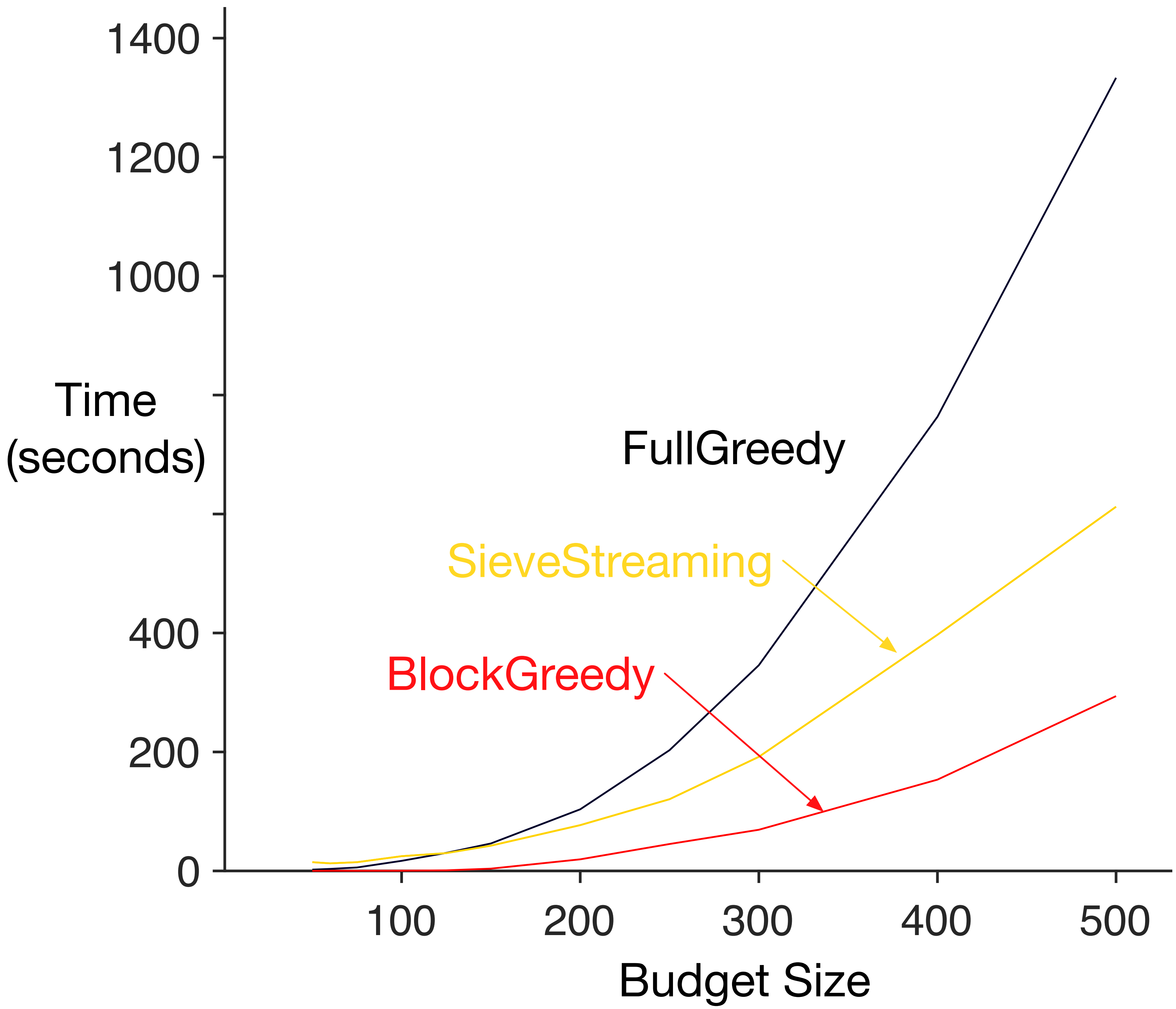
Log-determinant

budget = 200
block size = 5

Impact of block diagonal approximation

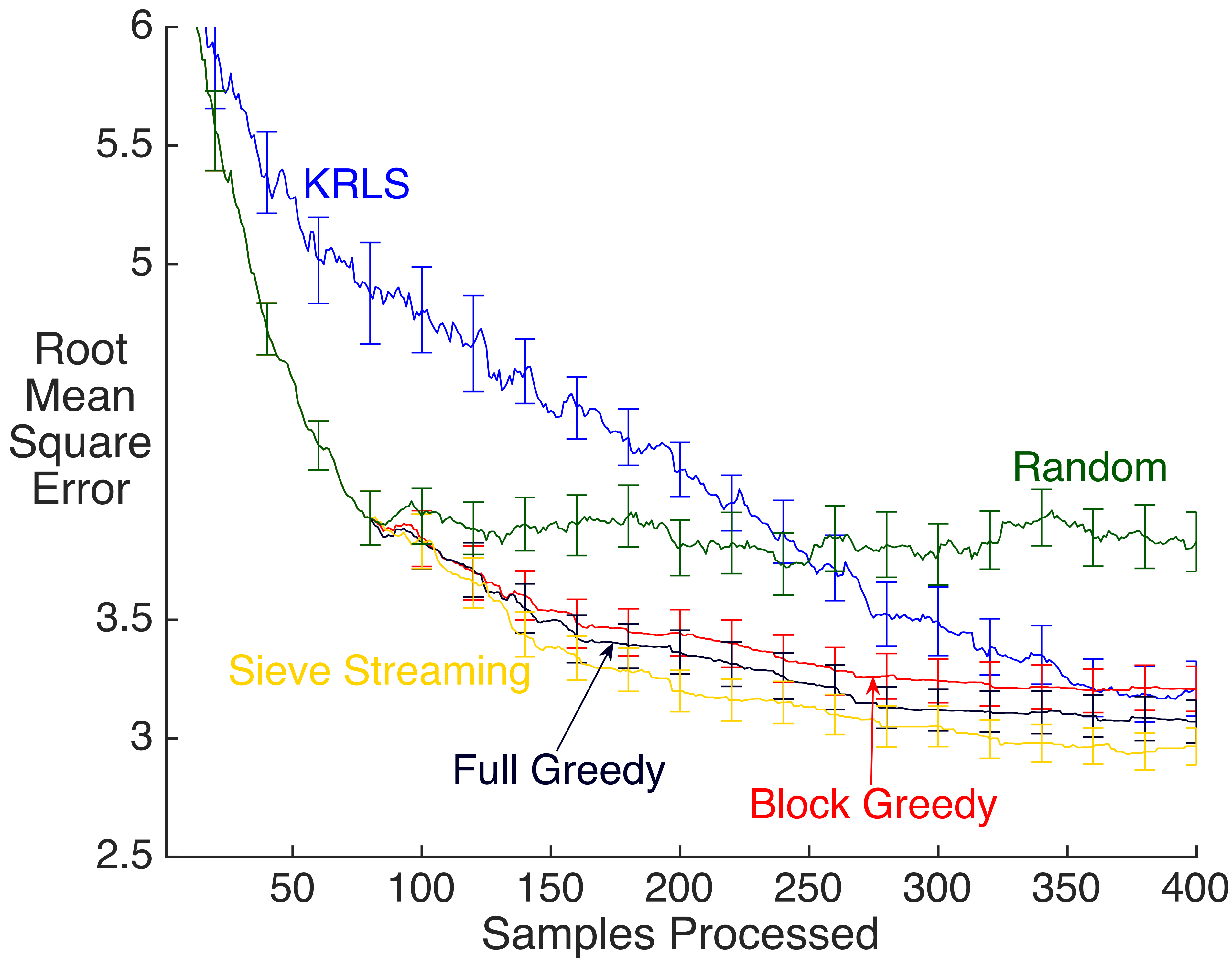


Runtime

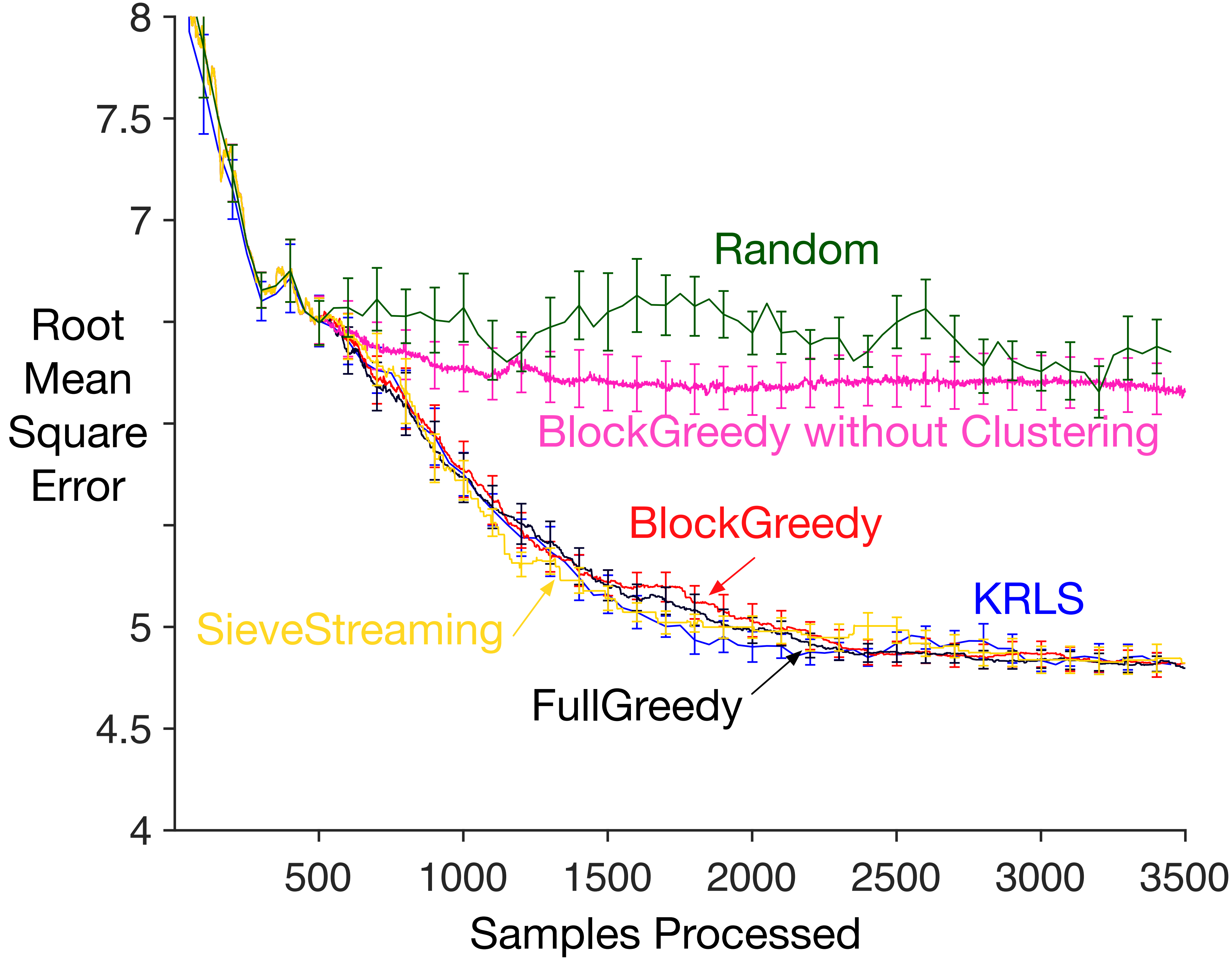


1400
1200
1000
400
200
0
Time
(seconds)
FullGreedy
SieveStreaming
BlockGreedy
100
200
300
400
500
Budget Size

Regression: Boston housing



Regression: Telemonitoring



What is really new?

- BlockGreedy algorithm, which is O(b) per step
- Introduced coverage property to generalize from streaming algorithms to continual learning
- A space of possible supervised and unsupervised criteria to explore under generalized coherence

Next steps

- Incorporate supervised criteria
- Automatically selecting kernels & meta-parameters
- Improve incremental regression algorithm
- More experiments validating practicality of approach

Next steps

- Incorporate supervised criteria
- Automatically selecting kernels & meta-parameters
- Improve incremental regression algorithm
- More experiments validating practicality of approach

Thank you for your attention