Exact Nonparametric Decentralized Online Optimization

Hrusikesha Pradhan [†], Amrit Singh Bedi [†], Alec Koppel[§], Ketan Rajawat [†]

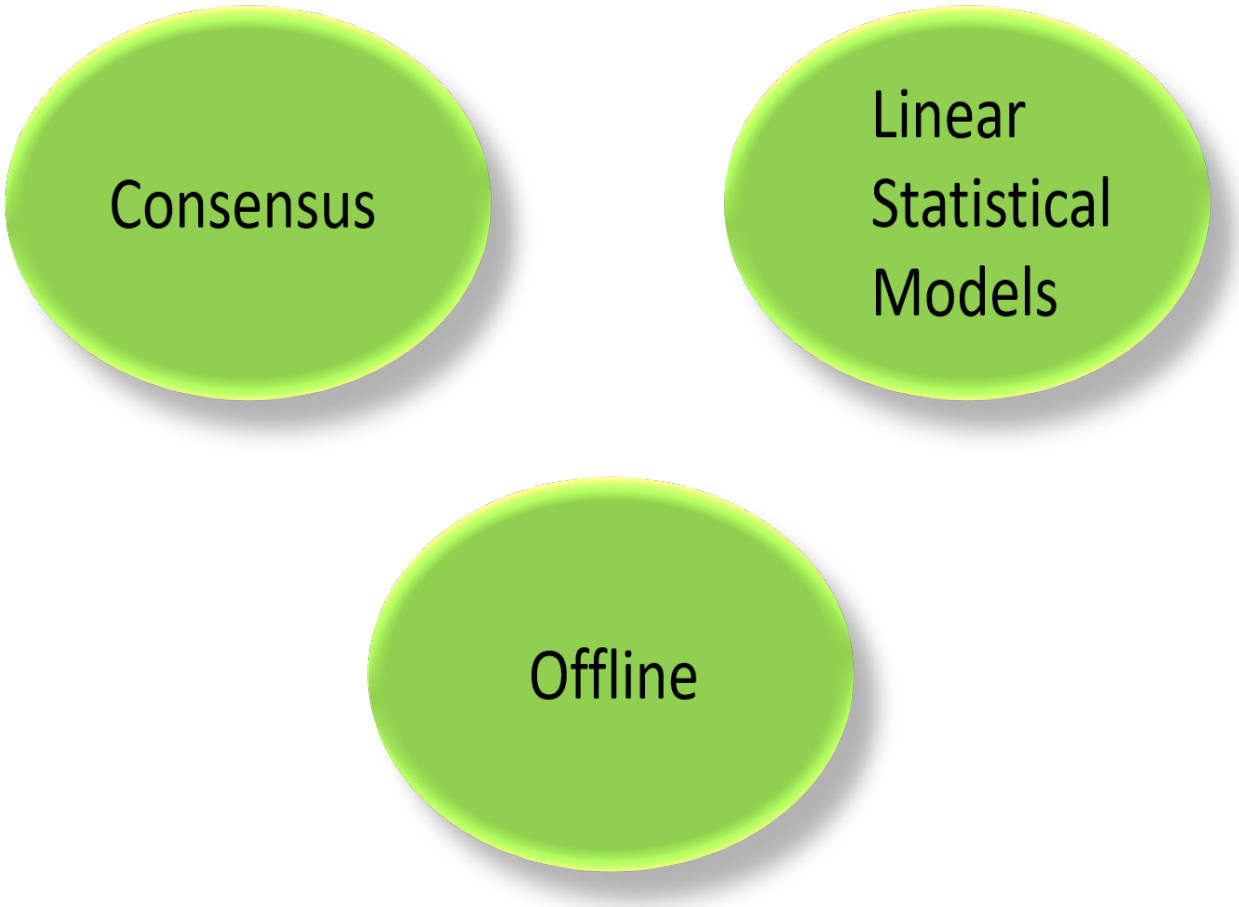
[†] Dept. of EE, India Institute of Technology Kanpur,

[§] CISD, U.S. Army Research Laboratory

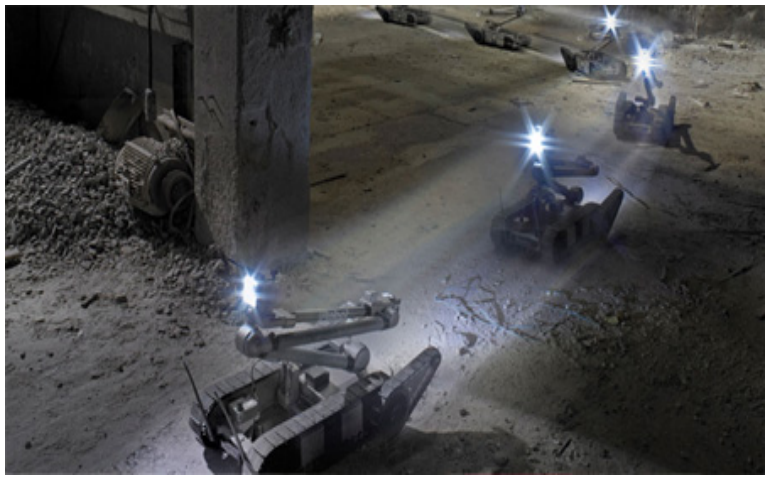
IEEE Global Conference on Signal and Information Processing

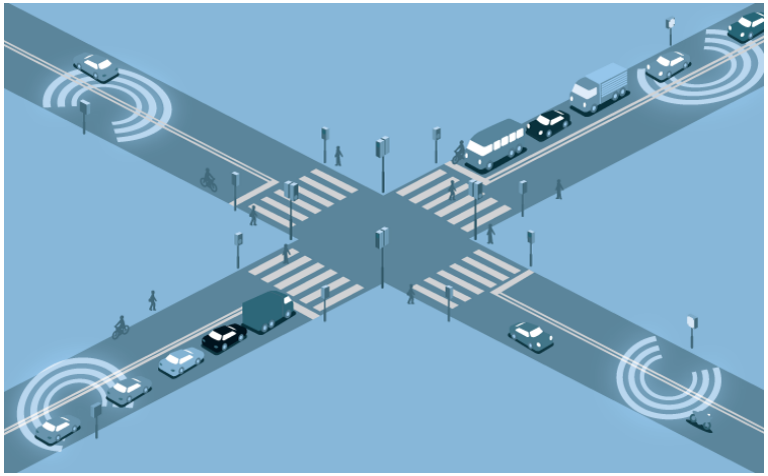
Anaheim, California, Nov. 28, 2018

The State of Distributed Learning

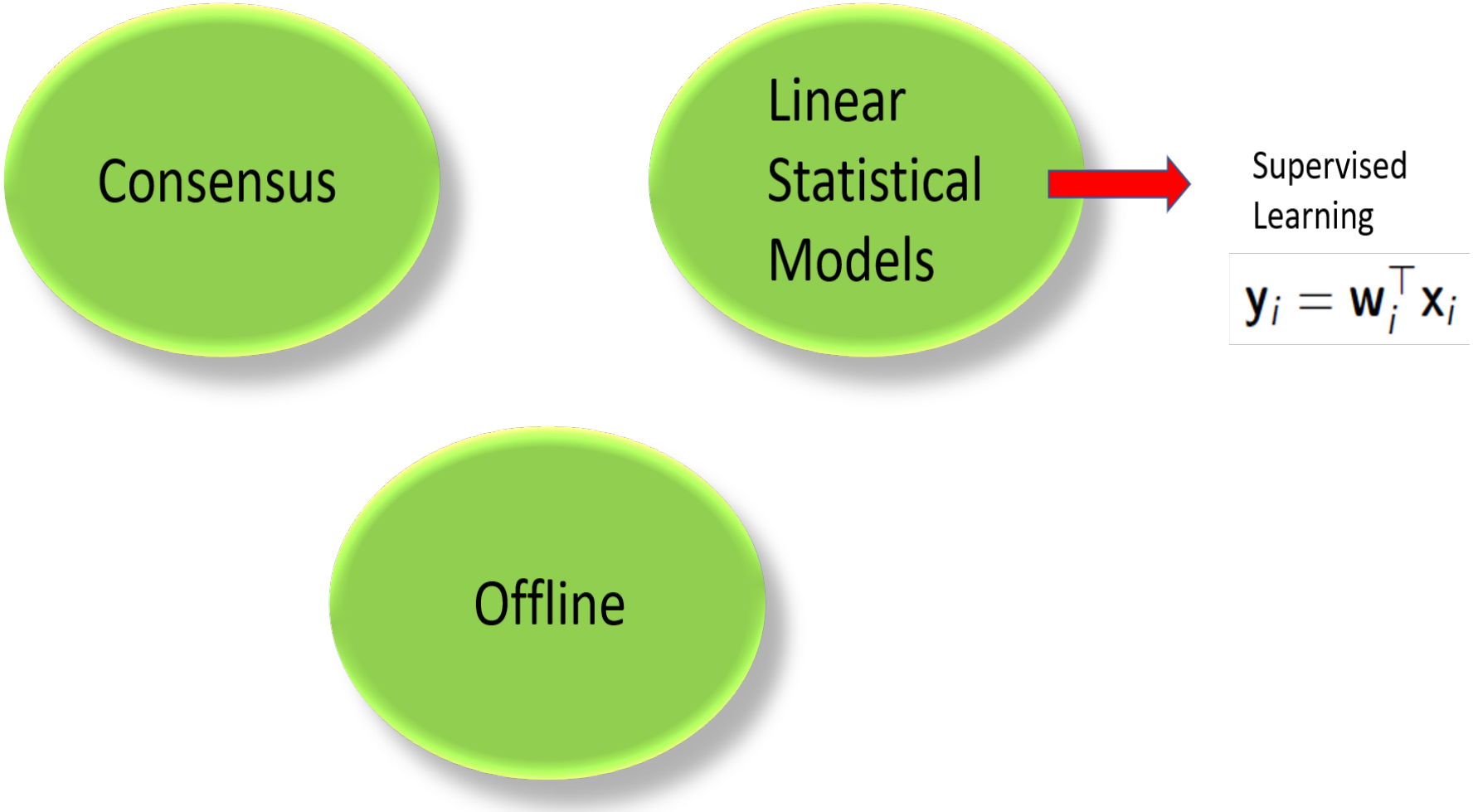


Non-linear Heterogeneous Networked Learning

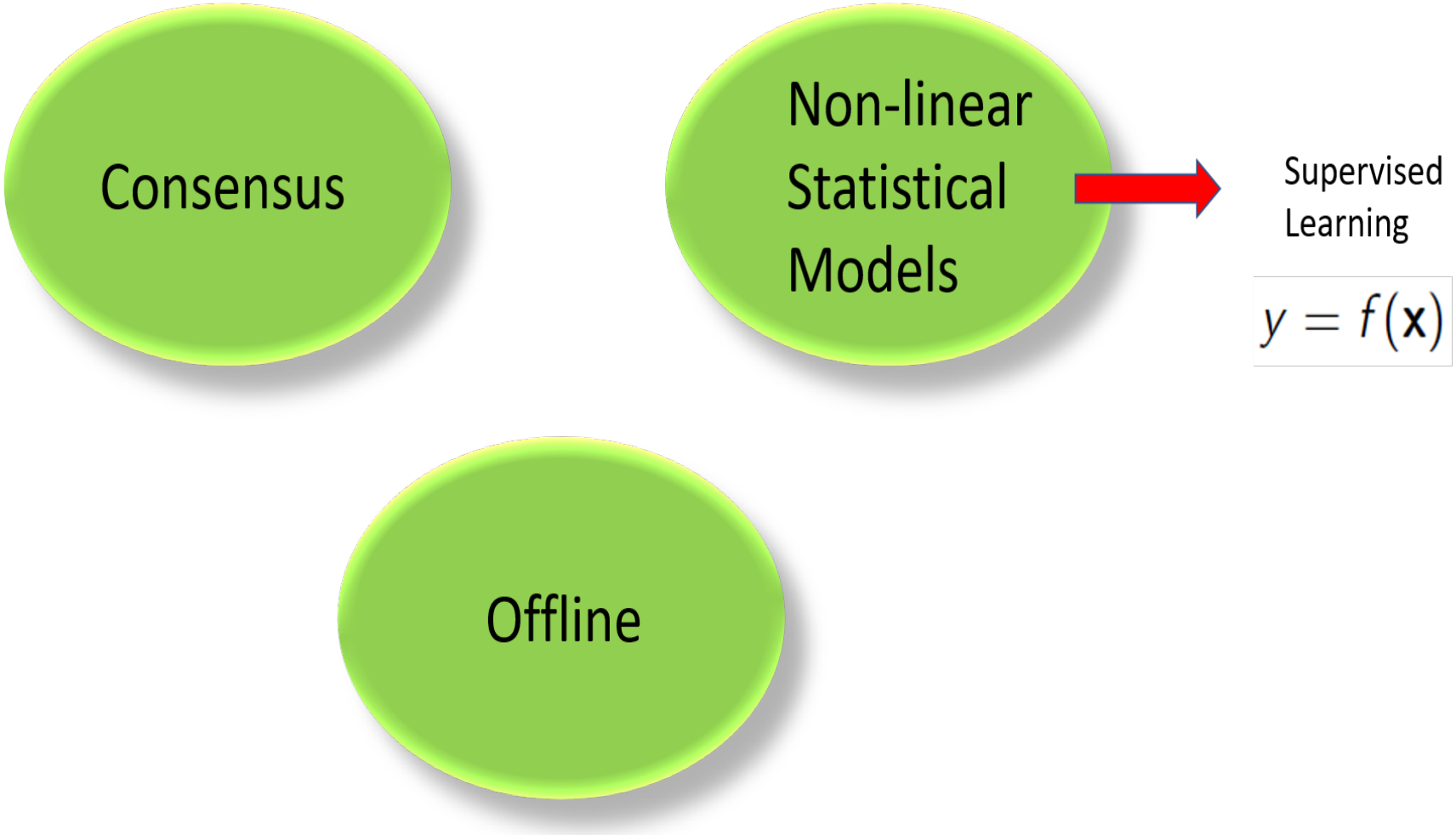




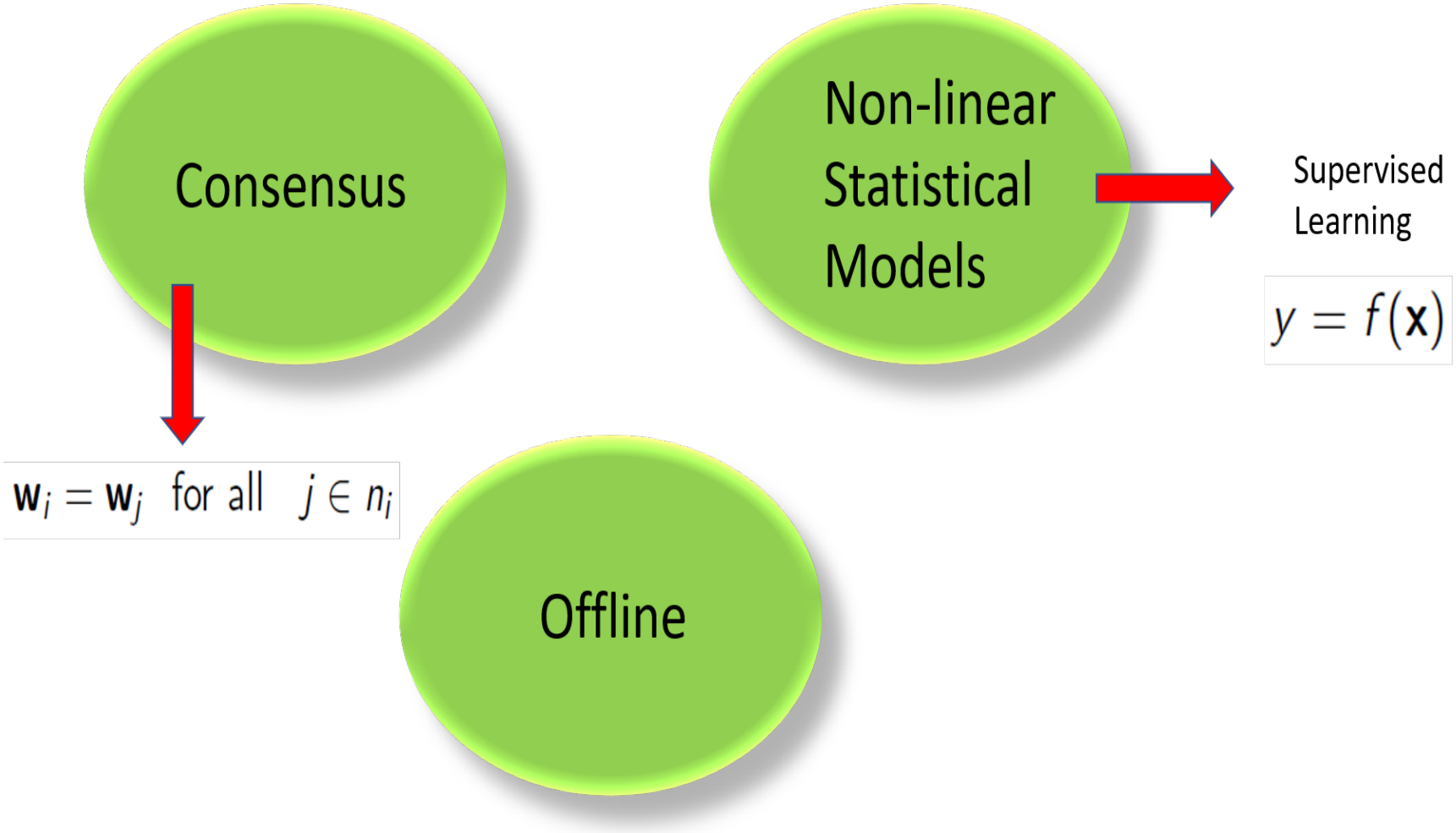
Existing methods don't apply



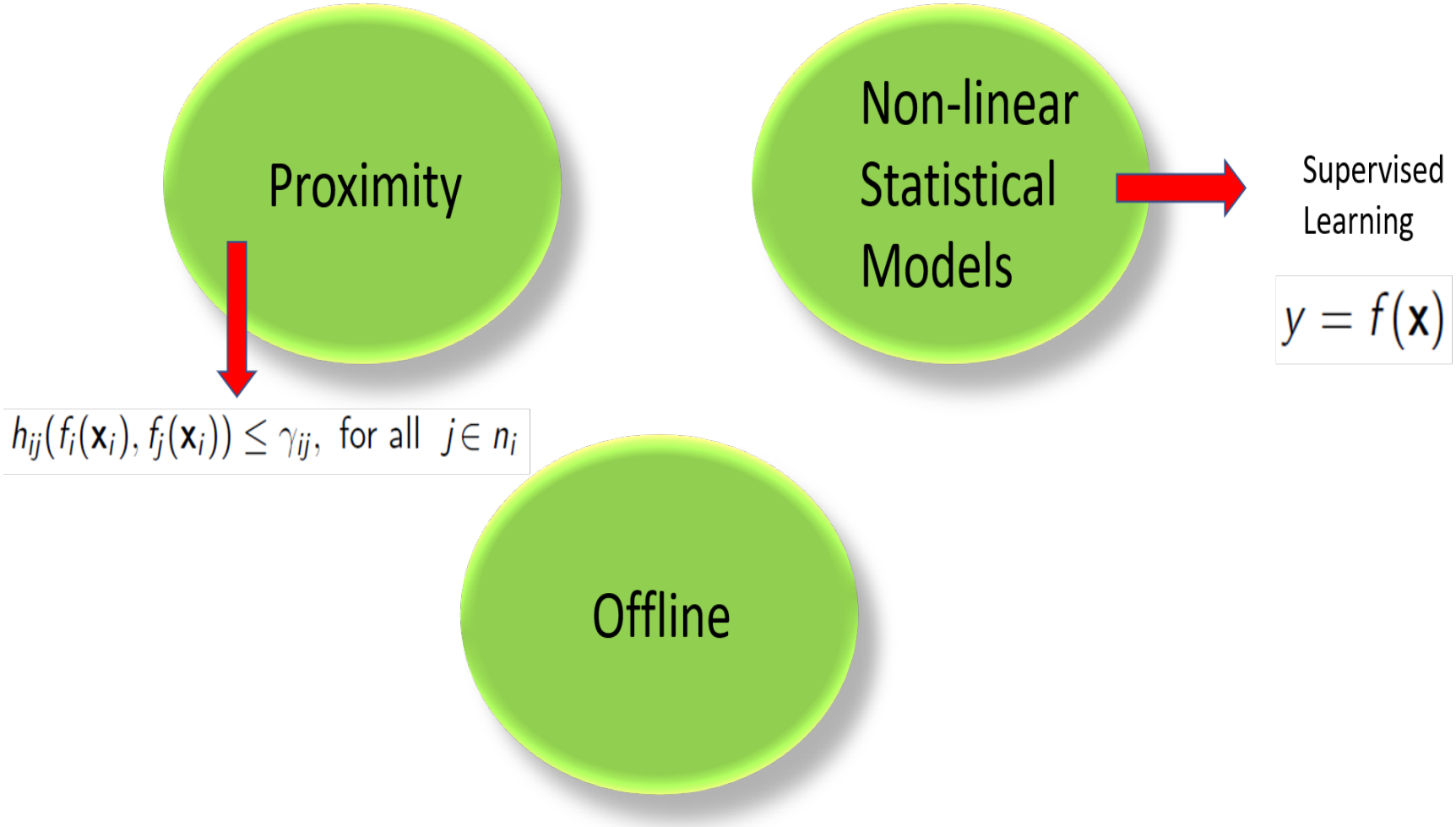
Existing methods don't apply



Existing methods don't apply



Existing methods don't apply



Existing methods don't apply

Proximity

$$h_{ij}(f_i(\mathbf{x}_i), f_j(\mathbf{x}_i)) \leq \gamma_{ij}, \text{ for all } j \in n_i$$

Non-linear Statistical Models → Supervised Learning

$$y = f(\mathbf{x})$$

Offline

$$\mathbf{w}^* := \underset{\mathbf{w} \in \mathbb{R}^{p|\mathcal{V}|}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{|\mathcal{V}|} \sum_{n=1}^{N} \ell_i(\mathbf{w}_i^\top \mathbf{x}_{i,n}, \mathbf{y}_{i,n})$$

Online Non-linear Heterogeneous Learning

Proximity

$$h_{ij}(f_i(\mathbf{x}_i), f_j(\mathbf{x}_i)) \leq \gamma_{ij}, \text{ for all } j \in n_i$$

Non-linear Statistical Models

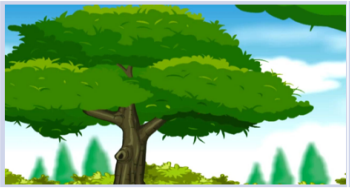
Supervised Learning

$$y = f(\mathbf{x})$$

Online

$$f^\star = \underset{\{f_i\} \in \mathcal{H}}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} \mathbb{E}_{\mathbf{x}_i, y_i} \left[\ell_i(f_i(\mathbf{x}_i), y_i) \right]$$

Contribution

Distributed Online Learning	Linear	Nonlinear
Homogeneous		Koppel ,
Heterogeneous	Sayed, Chen, Nassif, Lee, Sadler	**Our Work**

Haoran Sun, Mingyi Hong, Nikos D. Sidiropoulos

A. Nedic and A. Ozdaglar, Distributed subgradient methods for multiagent optimization, IEEE Transactions on Automatic Control

Mokhtari, Aryan, and Alejandro Ribeiro. "DSA: Decentralized double stochastic averaging gradient algorithm." JMLR

Sirb, B., Ye, X. (2018). Decentralized consensus algorithm with delayed and stochastic gradients. SIAM Journal on Optimization.

Chen, T., Ling, Q., Giannakis, G. B. (2017). An online convex optimization approach to proactive network resource allocation. IEEE Transactions on Signal Processing, 65(24), 6350-6364.

Sun Haoran, and Mingyi Hong. "Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms." arXiv preprint arXiv:1804.02729 (2018)

Sun Haoran, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nikos D. Sidiropoulos. "Learning to optimize: Training deep neural networks for wireless resource management." In SPAWC, 2017 IEEE 18th International Workshop on, pp. 1-6. IEEE, 2017.

Chen, J., Richard, C. and Sayed, A. H. (2014). Multitask diffusion adaptation over networks. IEEE Transactions on Signal Processing

Nassif, R., Richard, C., Ferrari, A. and Sayed, A. H. Distributed learning over multitask networks with linearly related tasks.

Lee, S., Zavlanos, M. M. (2017). On the sublinear regret of distributed primal-dual algorithms for online constrained optimization.

A. Koppel, S. Paternain, C. Richard, and A. Ribeiro, "Decentralized Online Learning with Kernels", in IEEE Trans. Signal Process

A. Koppel, B. Sadler, and A. Ribeiro, "Proximity without Consensus in Online Multi-Agent Optimization," in IEEE Trans. Signal

Outline

- **Approach**:
  - $\Rightarrow$ Hypothesized non-linear function in kernel Hilbert space
  - $\Rightarrow$ Form stochastic lagrangian
  - $\Rightarrow$ Apply stochastic primal dual method
  - $\Rightarrow$ Take subspace projection (to handle memory growth)
- Sublinear convergence
  - $\Rightarrow$ $\mathcal{O}(\sqrt{T})$ for primal optimality
  - $\Rightarrow$ $\mathcal{O}(T^{3/4})$ for constraint violation
- Generalizes existing rate results for primal-dual method
  - $\Rightarrow$ to case of non-linear statistical models
- **Application**: Decentralized correlated random field estimation
  - $\Rightarrow$ online non-linear observation model.

The Problem

- Symmetric, connected and directed network of agents $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- Learning nonlinear statistical models is equivalent to finding

  $\Rightarrow f : \mathcal{X} \to \mathcal{Y}$, such that $y = f(\mathbf{x})$
- Loss $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ penalize deviations between $f(\mathbf{x})$, $\mathbf{y}$
- Encoded by a convex proximity function $h_{ij}(f_i(\mathbf{x}_i), f_j(\mathbf{x}_i))$

  $\Rightarrow$ incentivizes nearby agents to make similar decisions
- Yields the constrained functional stochastic program:

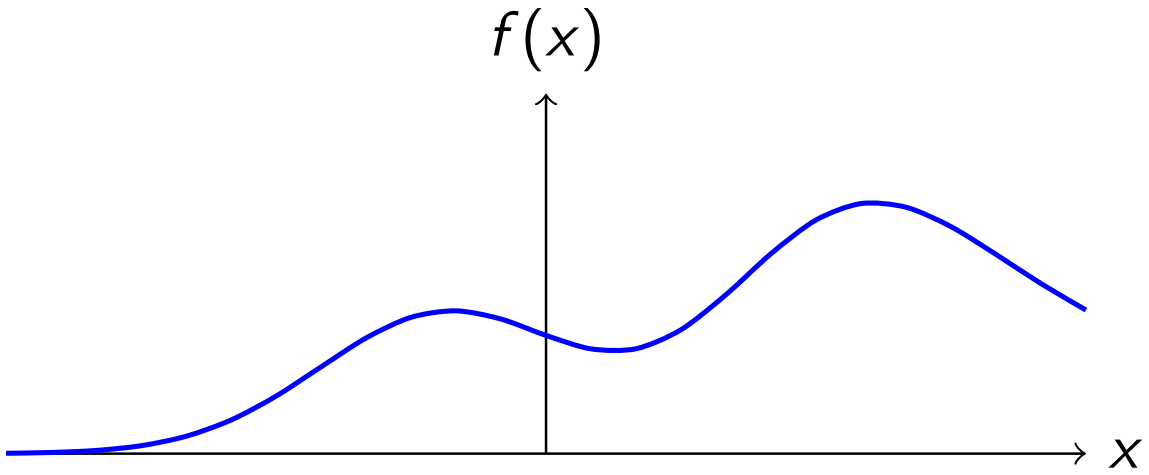
$$f^\star = \underset{\{f_i\} \in \mathcal{H}}{\operatorname{argmin}} \sum_{i \in \mathcal{V}} \left(\mathbb{E}_{\mathbf{x}_i, y_i} \left[\ell_i(f_i(\mathbf{x}_i), y_i) \right] + \frac{\lambda}{2} \|f_i\|^2_{\mathcal{H}} \right)$$

$$\text{s.t.} \quad \mathbb{E}_{\mathbf{x}_i} \left[h_{ij}(f_i(\mathbf{x}_i), f_j(\mathbf{x}_i)) \right] \leq \gamma_{ij}, \ \text{for all} \ \ j \in n_i. \tag{1}$$

Large-Scale Function Estimation

- Equip $\mathcal{H}$ with a unique *kernel function*, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\ ,$$
$$(ii)\ \mathcal{H} = \text{span}\{\kappa(\mathbf{x}, \cdot)\} \quad \text{for all } \mathbf{x} \in \mathcal{X}\ .$$

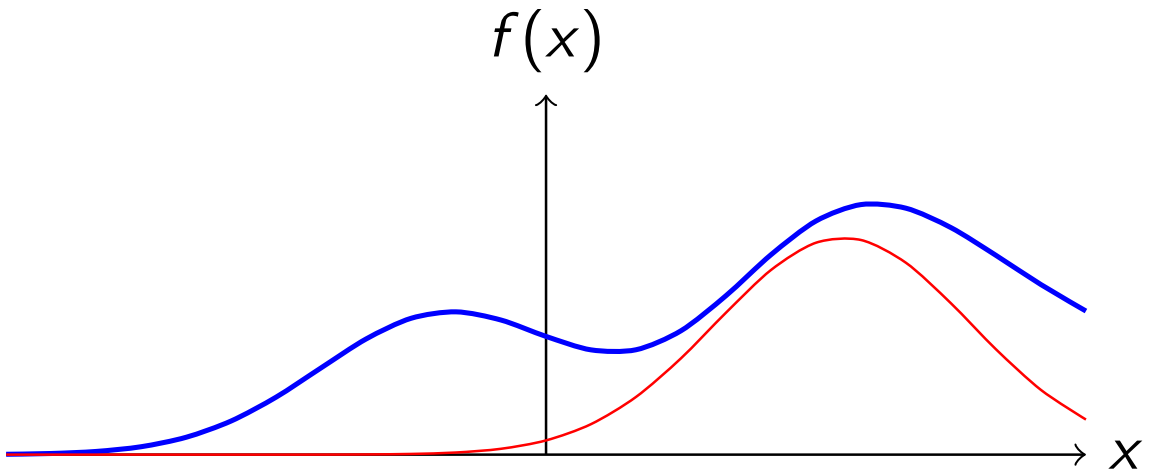


- Property (i) $\Rightarrow$ Will allow us to compute derivatives
- Kernel examples:
  - $\Rightarrow$ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$
  - $\Rightarrow$ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Large-Scale Function Estimation

- Equip $\mathcal{H}$ with a unique *kernel function*, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\ ,$$
$$(ii)\ \mathcal{H} = \text{span}\{\kappa(\mathbf{x}, \cdot)\} \quad \text{for all } \mathbf{x} \in \mathcal{X}\ .$$

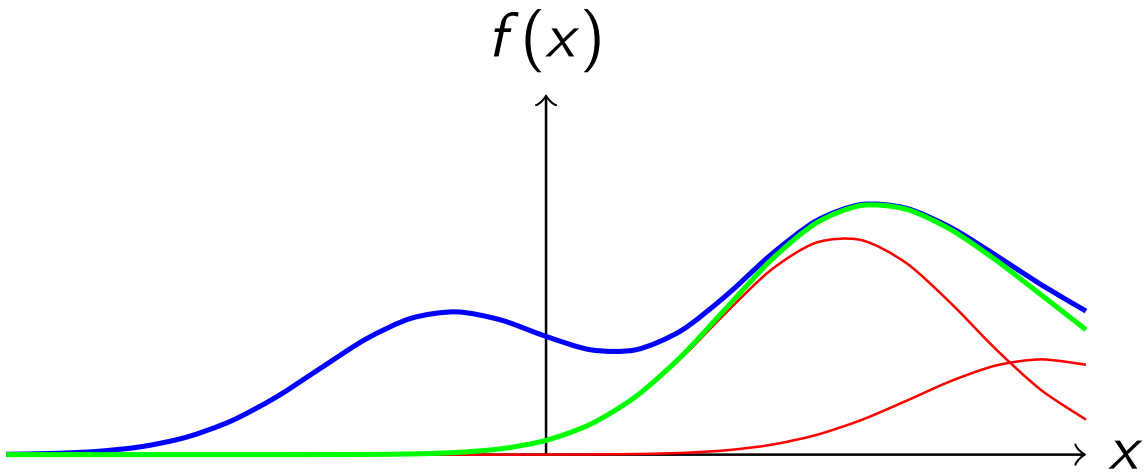


- Property (i) $\Rightarrow$ Will allow us to compute derivatives
- Kernel examples:
  - $\Rightarrow$ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$
  - $\Rightarrow$ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Large-Scale Function Estimation

- Equip $\mathcal{H}$ with a unique *kernel function*, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\ ,$$
$$(ii)\ \mathcal{H} = \text{span}\{\kappa(\mathbf{x}, \cdot)\} \quad \text{for all } \mathbf{x} \in \mathcal{X}\ .$$

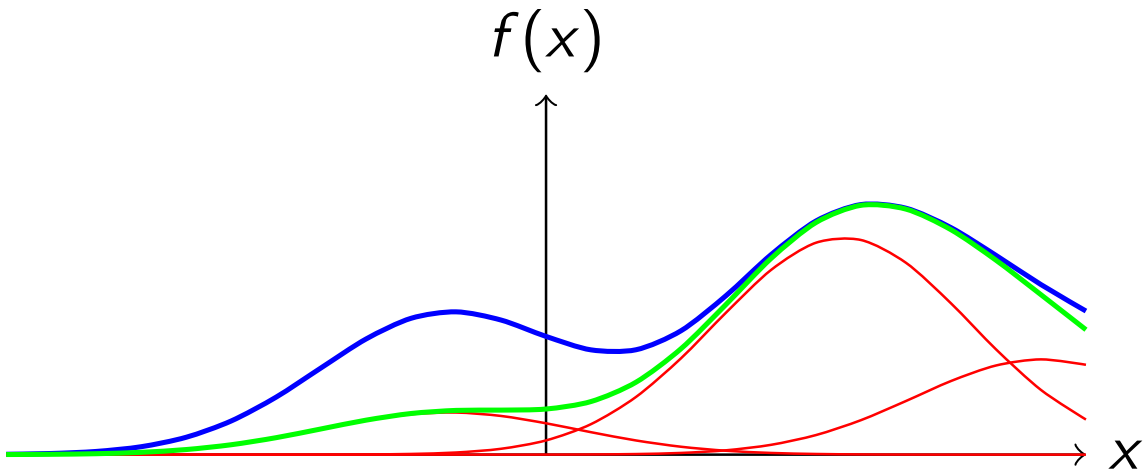


- Property (i) $\Rightarrow$ Will allow us to compute derivatives
- Kernel examples:
  - $\Rightarrow$ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$
  - $\Rightarrow$ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Large-Scale Function Estimation

- Equip $\mathcal{H}$ with a unique *kernel function*, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\ ,$$
$$(ii)\ \mathcal{H} = \text{span}\{\kappa(\mathbf{x}, \cdot)\} \quad \text{for all } \mathbf{x} \in \mathcal{X}\ .$$

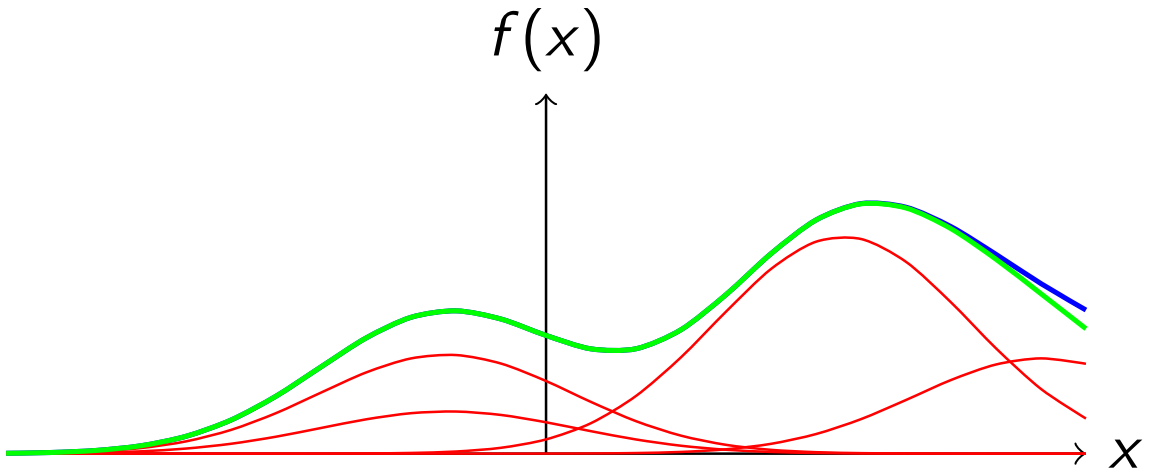


- Property (i) $\Rightarrow$ Will allow us to compute derivatives
- Kernel examples:
  - $\Rightarrow$ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$
  - $\Rightarrow$ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Large-Scale Function Estimation

- Equip $\mathcal{H}$ with a unique *kernel function*, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\ ,$$
$$(ii)\ \mathcal{H} = \text{span}\{\kappa(\mathbf{x}, \cdot)\} \quad \text{for all } \mathbf{x} \in \mathcal{X}\ .$$

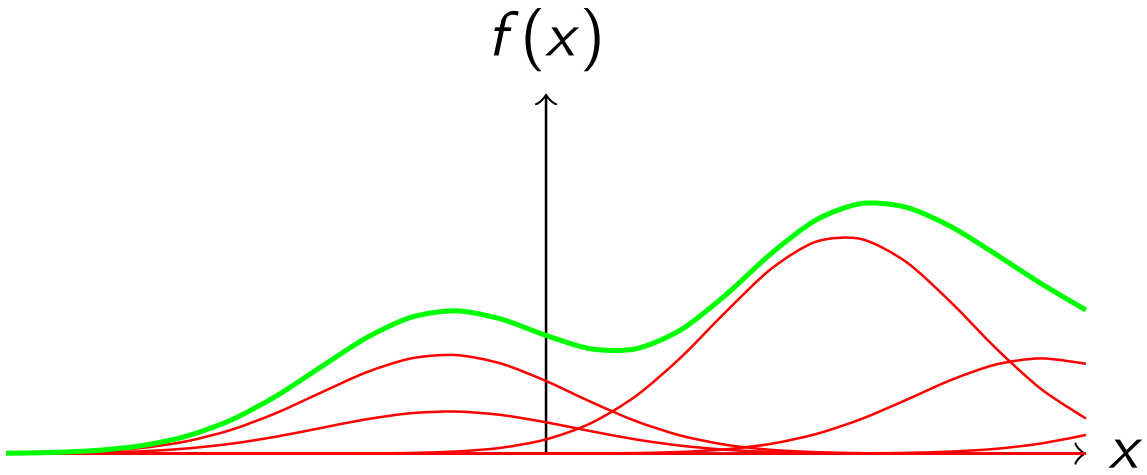


- Property (i) $\Rightarrow$ Will allow us to compute derivatives
- Kernel examples:
  - $\Rightarrow$ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$
  - $\Rightarrow$ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Large-Scale Function Estimation

- Equip $\mathcal{H}$ with a unique *kernel function*, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$(i)\ \langle f, \kappa(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathcal{X}\ ,$$
$$(ii)\ \mathcal{H} = \text{span}\{\kappa(\mathbf{x}, \cdot)\} \quad \text{for all } \mathbf{x} \in \mathcal{X}\ .$$



- Property (i) $\Rightarrow$ Will allow us to compute derivatives
- Kernel examples:
  - $\Rightarrow$ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$
  - $\Rightarrow$ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Learning with Representer Theorem

- Stochastic augmented Lagrangian function of (1) at time t

$$\hat{\mathcal{L}}_t(f,\boldsymbol{\mu}) := \sum_{i\in\mathcal{V}}\Bigg[\ell_i(f_i(\mathbf{x}_{i,t}),y_{i,t}) + \frac{\lambda}{2}\|f_i\|^2_{\mathcal{H}} + \sum_{j\in n_i}\Bigg\{\Big[\mu_{ij}(h_{ij}(f_i(\mathbf{x}_{i,t}),f_j(\mathbf{x}_{i,t})) - \gamma_{ij})\Big] - \frac{\delta\eta}{2}\mu_{ij}^2\Bigg\}\Bigg] \tag{2}$$

where $\boldsymbol{\mu}$ is a lagrange multiplier, with μ_{ij} defined for each $(i,j)\in\mathcal{E}$.

Corollary

Consider the sample average approximation of (1)*, and its associated Lagrangian relaxation. The each ith component of the solution to the resulting saddle-point problem can be expressed as*

$$f_i^* = \sum_{t=1}^{T} w_{i,t}\kappa(\mathbf{x}_{i,t},.) \tag{3}$$

where $w_{i,t}$ are real-valued coefficients.

Functional Derivative

- Functional stochastic gradient of local loss in (2):

$$\ell_i'(f_i(\mathbf{x}_{i,t}), y_{i,t}) := \nabla_{f_i} \ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t})(\cdot) = \frac{\partial \ell_i(f_i(\mathbf{x}_{i,t}), y_{i,t})}{\partial f_i(\mathbf{x}_{i,t})} \frac{\partial f_i(\mathbf{x}_{i,t})}{\partial f_i}(\cdot)$$

- Using the reproducing property of the kernel we obtain

$$\frac{\partial f_i(\mathbf{x}_{i,t})}{\partial f_i} = \frac{\partial \langle f_i, \kappa(\mathbf{x}_{i,t}, \cdot) \rangle_{\mathcal{H}}}{\partial f_i} = \kappa(\mathbf{x}_{i,t}, \cdot) \tag{4}$$

- Now the full gradient result can be written as

$$\begin{aligned} \nabla_{f_i} \hat{\mathcal{L}}_t(f_t, \mu_t) = \ell_i'(f_i(\mathbf{x}_{i,t}), y_{i,t}) \kappa(\mathbf{x}_{i,t}, \cdot) + \lambda f_i \\ + \sum_{j \in n_i} \mu_{ij} h_{ij}'(f_i(\mathbf{x}_{i,t}), f_j(\mathbf{x}_{i,t})) \kappa(\mathbf{x}_{i,t}, \cdot) \end{aligned} \tag{5}$$

Algorithm

- **loop in parallel** for agent $i \in \mathcal{V}$
- Observe local training example realization $(\mathbf{x}_{i,t}, y_{i,t})$
- Send $\mathbf{x}_{i,t}$ to the neighboring nodes, $j \in n_i$ and receive $f_{j,t}(\mathbf{x}_{i,t})$
- Receive $\mathbf{x}_{j,t}$ from the neighbouring nodes, $j \in n_i$ and send $f_{i,t}(\mathbf{x}_{j,t})$
- Stochastic primal descent step on Lagrangian:

$$f_{i,t+1} = f_{i,t}(1-\eta\lambda) - \eta\left[\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) + \sum_{j\in n_i} \mu_{ij} h_{ij}'(f_{i,t}(\mathbf{x}_{i,t}), f_{j,t}(\mathbf{x}_{i,t}))\right] \kappa(\mathbf{x}_{i,t}, \cdot) \quad (6)$$

- Stochastic dual ascent step on Lagrangian:

$$\mu_{ij,t+1} = \left[\mu_{ij,t}(1-\delta\eta^2) + \eta\left(h_{ij}(f_{i,t}(\mathbf{x}_{i,t}), f_{j,t}(\mathbf{x}_{i,t})) - \gamma_{ij}\right)\right]_+ \quad (7)$$

Parametric update of weights and dictionary

- Using $f_{i,t}(\mathbf{x}) = \sum_{n=1}^{t-1} w_{i,n}\kappa(\mathbf{x}_{i,n}, \mathbf{x}) = \mathbf{w}_{i,t}^T \boldsymbol{\kappa}_{\mathbf{X}_{i,t}}(\mathbf{x})$, V parallel parametric updates on both kernel dictionaries $\mathbf{X}_i$ and $\mathbf{w}_i$ are

$$\mathbf{X}_{i,t+1} = [\mathbf{X}_{i,t}, \quad \mathbf{x}_{i,t}] \,, \tag{8}$$

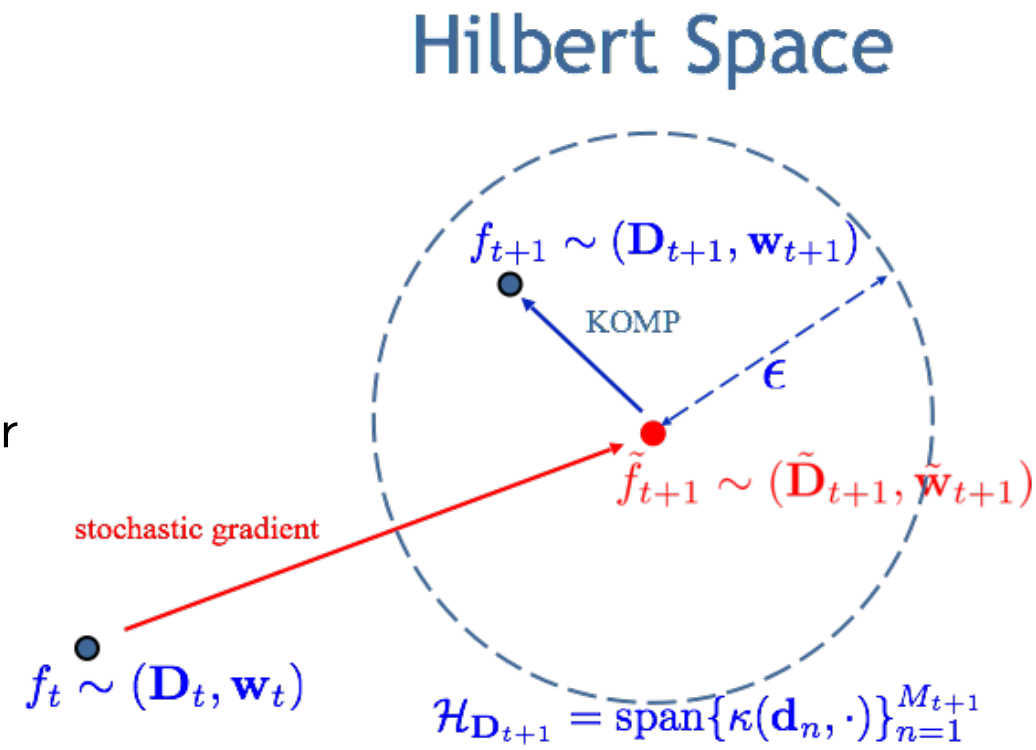
$$[\mathbf{w}_{i,t+1}]_u = \begin{cases} (1-\eta\lambda)[\mathbf{w}_{i,t}]_u, & 0 \le u \le t-1 \\ -\eta\Big(\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) \\ \quad + \sum_{j\in n_i} \mu_{ij} h_{ij}'(f_{i,t}(\mathbf{x}_{i,t}), f_{j,t}(\mathbf{x}_{i,t}))\Big), & u = t \end{cases}$$

- Data points $M_{i,t}$ grows by one each time (curse of kernelization).
- **Proj. Funct. Update**: Onto $\mathcal{H}_{\mathbf{D}_{i,t+1}} = \text{span}\{\kappa(\mathbf{d}_{i,n}, \cdot)\}_{n=1}^{M_{t+1}} \subset \mathcal{H}$

$$f_{i,t+1} := \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{i,t+1}}}\left[f_{i,t} - \eta\nabla_{f_i}\hat{\mathcal{L}}_t(f_t, \mu_t)\right]. \tag{9}$$

Kernel Matching Pursuit

- Fix approximation error ϵ
- $\tilde{f}_{t+1} = f_t - \eta \nabla_{f_i} \hat{\mathcal{L}}_t(f_t, \mu_t)$
- Remove kernel element smallest error
- Project $\tilde{f}_{t+1}$ onto resulting RKHS
- Repeat until error is larger than ε



Convergence Analysis

Theorem

Let $S(f_t) := \sum_{i\in\mathcal{V}} \mathbb{E}\left[\ell_i(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t})\right] + \frac{\lambda}{2}\|f_{i,t}\|^2_{\mathcal{H}}$ *as the objective with* f^* *defined in* (1)*. Then with constant step-size* $\eta = 1/\sqrt{T}$ *and approximation budget* $\epsilon_t = \epsilon = P\eta^2$*, where* $P > 0$ *is termed as the parsimony constant.*

$$\sum_{t=1}^{T} \mathbb{E}[S(f_t) - S(f^*)] \leq \mathcal{O}(\sqrt{T}). \tag{10}$$

Furthermore, the time-aggregation of the expected constraint violation grows sub-linearly with iteration T *as*

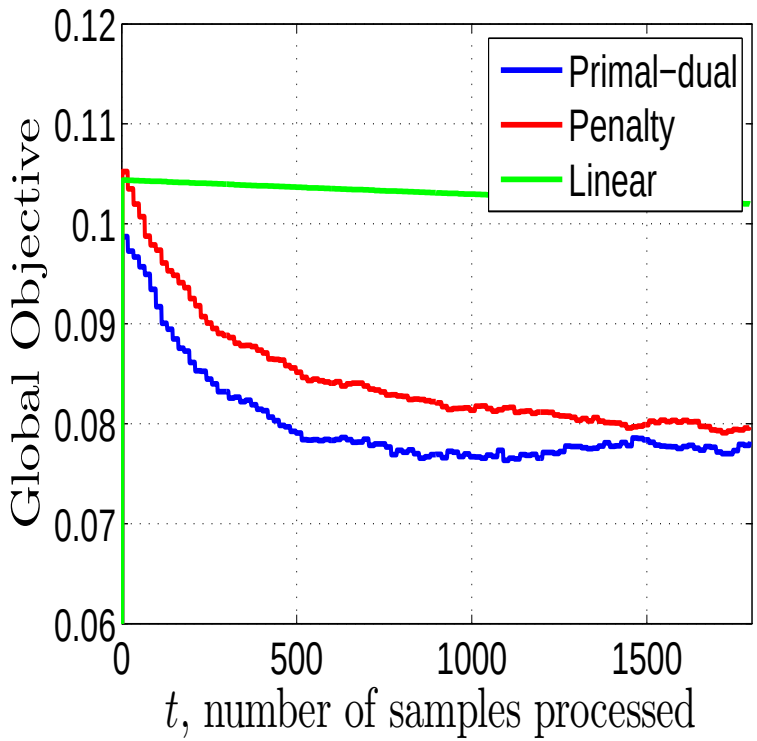
$$\sum_{(i,j)\in\mathcal{E}} \mathbb{E}\left[\sum_{t=1}^{T}(h_{ij}(f_i(\mathbf{x}_{i,t}), f_j(\mathbf{x}_{i,t})) - \gamma_{ij})\right]_+ \leq \mathcal{O}(T^{3/4}). \tag{11}$$

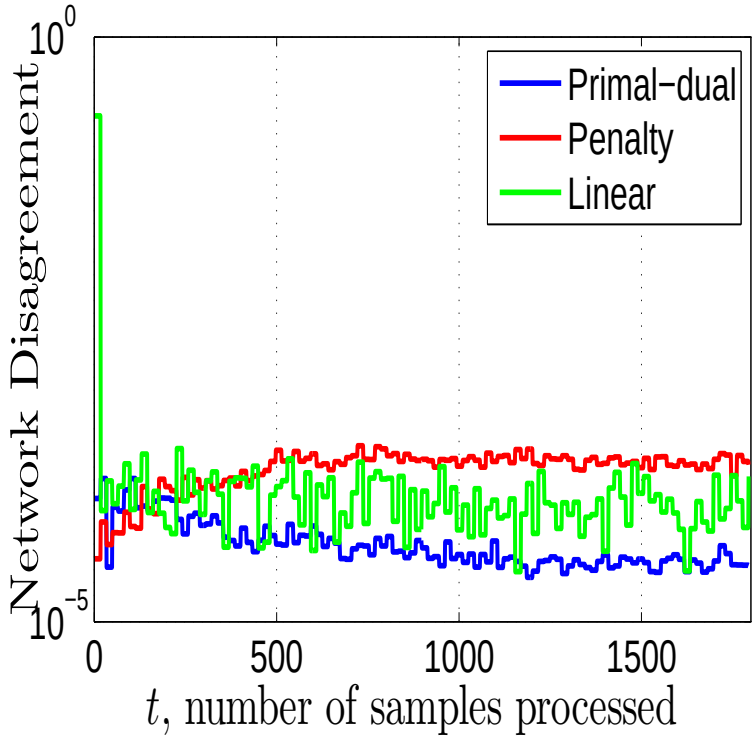
Estimating planar correlated Gaussian random field

- A planar field is a random function of spatial components
- Random field is parameterized by the correlation matrix $\mathbf{R}_x$
  $\Rightarrow [\mathbf{R}_x]_{ij}$ is assumed to have a structure of the form $e^{-\|l_i - l_j\|}$,
- Observation model: $y_{i,t} = h_i \|\mathbf{x}_{i,t}\|^2 + n_{i,t}$, $n_{i,t} \sim \mathcal{N}(0, \sigma^2)$
- 10 nodes distributed in a 10×10 meter square area.
- Instantaneous observation, $\mathbf{x}_t = \pi + \mathbf{C}^T \mathbf{v}_t$
  $\Rightarrow \pi = \{1/V, 2/V, \ldots, 1\}$
  $\Rightarrow \mathbf{C}$: Cholesky factorization of the correlation matrix $\mathbf{R}_x$
  $\Rightarrow \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$

Field Estimation

- Convergence of global objective and reduced network disagreement





Field Estimation

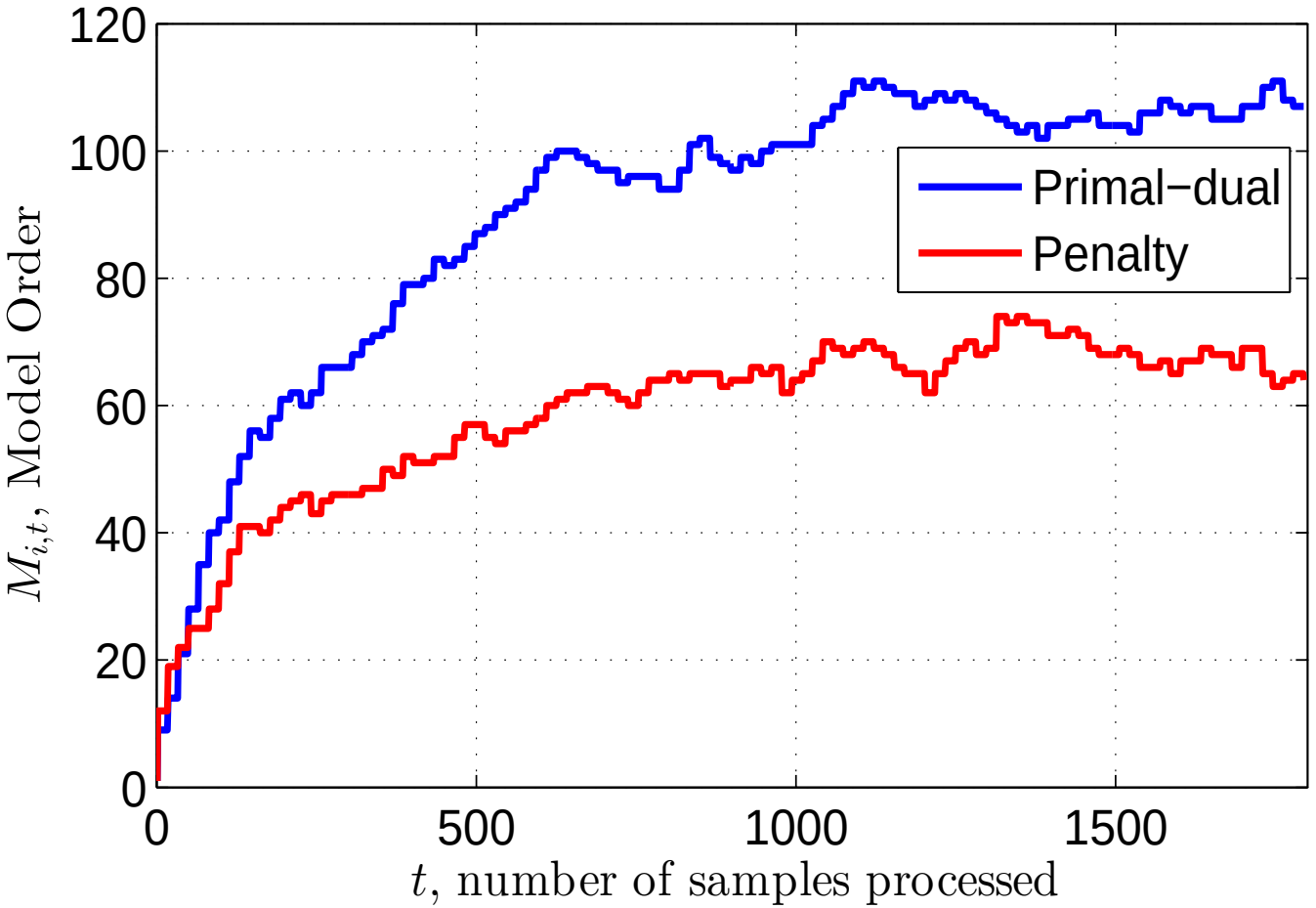


Figure: Model Order $\mathbf{M}_{i,t}$ v.s. Samples

Conclusion and Future Work

- Focused on online learning
  - $\Rightarrow$ Decentralized heterogeneous networks
  - $\Rightarrow$ Non-linear statistical models
- Proposed new variant of projected stochastic primal dual method
  - $\Rightarrow$ Convergence to the optimum
  - $\Rightarrow$ Finite growth of model order
  - $\Rightarrow$ Observed good empirical performance
- **Future Work**:
  - $\Rightarrow$ Asynchrony
  - $\Rightarrow$ Reduce complexity of projections

Appendix: Supervised Learning Background

- $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ is random pair $\Rightarrow$ training examples
- $\ell : \mathcal{W} \to \mathbb{R}$ convex loss ($\mathcal{W} \subset \mathbb{R}^p$), merit of statistical model
- Find parameters $\mathbf{w}^* \in \mathbb{R}^p$ that minimize expected risk $L(\mathbf{w})$

$$\mathbf{w}^* := \underset{\mathbf{w}}{\operatorname{argmin}}\, L(\mathbf{w}) := \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\mathbf{w}^\top \mathbf{x}, \mathbf{y})]$$

- Convex Optimization Problem for *linear statistical models*

  $\Rightarrow$ e.g., $y = \mathbf{w}^T \mathbf{x} \in \mathbb{R}$ or $y = \operatorname{sgn}(\mathbf{w}^T \mathbf{x}) \in \{-1, 1\}$
- Solve with favorite descent method $\Rightarrow$ Good Performance

Appendix: Easy to Implement over Networks

- Symmetric, connected and directed network of agents $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- The nodes aims to make inferences from local data
- $|\mathcal{V}| = V$ nodes, $|\mathcal{E}| = M$ edges, and $n_i := \{j : (i,j) \in \mathcal{E}\}$
- Agent $i \in \mathcal{V}$ has a local copy of the classifier $\mathbf{w}_i$

  $\Rightarrow$ Observes some training examples $\Rightarrow (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X}_i \times \mathcal{Y}_i$

$$\mathbf{w}^* := \underset{\mathbf{w} \in \mathbb{R}^{p|\mathcal{V}|}}{\operatorname{argmin}} \sum_{i=1}^{|\mathcal{V}|} \mathbb{E}_{\mathbf{x}_i, y_i} \left[\ell(\mathbf{w}_i^\top \mathbf{x}_i, \mathbf{y}_i) \right]$$

$$s.t. \quad \mathbf{w}_i = \mathbf{w}_j \quad \text{for all} \quad j \in n_i$$

- Convex Optimization Problem for *linear statistical models*
- Solve with saddle point algorithms or penalty methods

  $\Rightarrow$ Can be implemented in a distributed fashion

Appendix: How to control the Model order?

- Project (6) onto a lower dimensional subspace $\mathcal{H}_{\mathbf{D}} \subseteq \mathcal{H}$
- $\mathcal{H}_{\mathbf{D}}$ is represented by a dictionary $\mathbf{D} = [\mathbf{d}_1, \ \ldots, \ \mathbf{d}_M] \in \mathbb{R}^{p \times M}$.
- $\mathcal{H}_{\mathbf{D}} = \{f \ : \ f(\cdot) = \sum_{n=1}^{M} w_n \kappa(\mathbf{d}_n, \cdot) = \mathbf{w}^T \boldsymbol{\kappa}_{\mathbf{D}}(\cdot)\}$, $\{\mathbf{d}_n\} \subset \{\mathbf{x}_u\}_{u \leq t}$.
- We denote the un-projected functional update as

$$\tilde{f}_{i,t+1} = f_{i,t} - \eta \nabla_{f_i} \hat{\mathcal{L}}_t(f_t, \mu_t) \ . \tag{12}$$

where $\nabla_{f_i} \hat{\mathcal{L}}_t(f_t, \mu_t) :=$
$\lambda f_{i,t} + \Big[\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) + \sum_{j \in n_i} \mu_{ij} h_{ij}'(f_{i,t}(\mathbf{x}_{i,t}), f_{j,t}(\mathbf{x}_{i,t}))\Big] \kappa(\mathbf{x}_{i,t}, \cdot)$.

- $\tilde{f}_{i,t+1}$ in form of dictionary and coefficient vector:

$$\tilde{\mathbf{D}}_{i,t+1} = [\mathbf{D}_{i,t}, \ \ \mathbf{x}_{i,t}] \ ,$$

$$[\tilde{\mathbf{w}}_{i,t+1}]_u = \begin{cases} (1 - \eta\lambda)[\mathbf{w}_{i,t}]_u, & \text{for } 0 \leq u \leq t-1 \\ -\eta\Big(\ell_i'(f_{i,t}(\mathbf{x}_{i,t}), y_{i,t}) + \sum_{j \in n_i} \mu_{ij} h_{ij}'(f_{i,t}(\mathbf{x}_{i,t}), f_{j,t}(\mathbf{x}_{i,t}))\Big), & \text{for } u = t \end{cases}$$