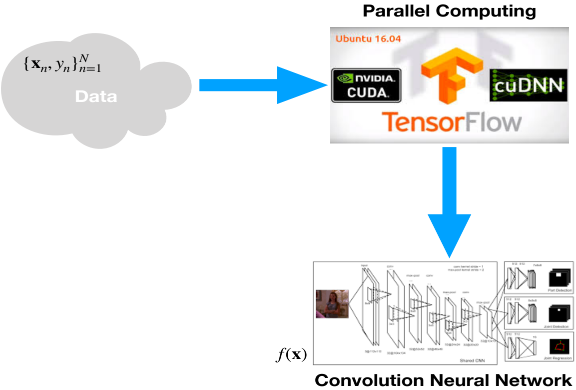
Controlling the Bias-Variance Tradeoff via Coherent Risk for Robust Learning with Kernels

Alec Koppel[⋆] **Amrit Bedi Singh**[⋆] Ketan Rajawat[†]
[⋆]ARL-CISD [†]Dept. of EE, IIT Kanpur

Statistical Learning
IEEE American Control Conference
July 11, 2019

Today's Industrial Machine Learning



Fundamentally requires static big data available in cloud storage

$\Rightarrow$ sample size N large & fixed, $\mathbf{x}_n \in \mathbb{R}^p$, p also large

$\Rightarrow$ $(\mathbf{x}_n, y_n)$ denote training examples

$\Rightarrow$ train model statically deployed in, e.g., Alexa, iPhone

Learning for Autonomy

→ Autonomous systems ⇒ often no big data available

→ Accumulate daily data, send to cloud (Tesla approach)?

⇒ requires standardized platforms

→ Run complex simulations ?

⇒ may be unrepresentative of reality

→ For autonomy, in situ learning & adaptation required

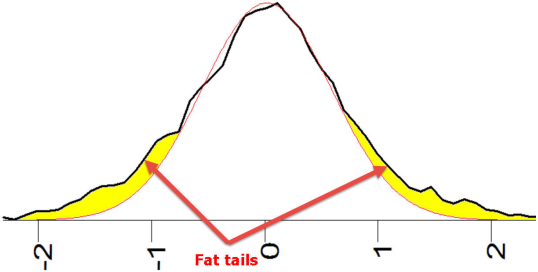
→ Goal: adaptive classification of individuals/vehicles/buildings

⇒ reliable across training, i.e., insensitive to "black swans"

Bias-Variance and Overfitting

$\rightarrow$ If data distribution $\mathbb{P}(\mathbf{x}, \mathbf{y})$ has heavy tails

$\Rightarrow$ then learning $f(\mathbf{x})$ by minimizing **average** loss will "overfit"



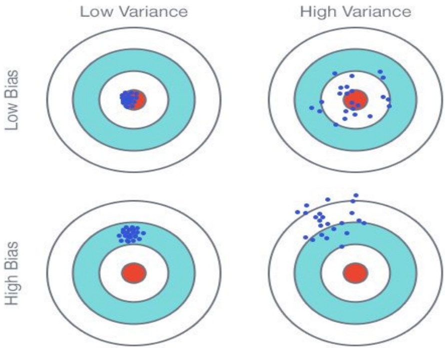
$\Rightarrow$ Overfitting $\Rightarrow$ memorizing the noise

$\rightarrow$ Comms. errors, robot instability, monitor confusion

Bias-Variance and Overfitting

$\rightarrow$ If data distribution $\mathbb{P}(\mathbf{x}, \mathbf{y})$ has heavy tails

$\Rightarrow$ then learning $f(\mathbf{x})$ by minimizing **average** loss will "overfit"



$\Rightarrow$ Overfitting $\Rightarrow$ memorizing the noise

$\rightarrow$ Comms. errors, robot instability, monitor confusion

Estimation & Approximation Error

$\rightarrow$ Supervised learning solves for fixed $f \in \mathscr{F}$

$$f^* = \underset{f \in \mathscr{F}}{\text{argmin}}\, \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})]$$

$\rightarrow$ approximates Bayes optimal $\hat{\mathbf{y}}^\star = \text{argmin}_{\hat{\mathbf{y}} \in \mathcal{Y}^\mathcal{X}}\, \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\hat{\mathbf{y}}(\mathbf{x}), \mathbf{y})]$

$$\mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\hat{f}(\mathbf{x}), \mathbf{y})] - \min_{f \in \mathscr{F}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})] \quad \Rightarrow \text{bias}$$

$$+ \min_{f \in \mathscr{F}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})] - \min_{\hat{\mathbf{y}} \in \mathcal{Y}^\mathcal{X}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(\hat{\mathbf{y}}(\mathbf{x}), \mathbf{y})] \quad \Rightarrow \text{variance}$$

$\Rightarrow$ where $\mathcal{Y}^\mathcal{X}$ denotes the space of all functions from $\mathcal{X} \rightarrow \mathcal{Y}$

Dealing with Variance

Possible approaches

→ Cross validate: run w/ diff. params., remove data subsets

→ Regularization: add a l_1 or l_0 penalty

→ Data augmenting (bootstrap): randomly perturb data & rerun

⇒ all of these are only applicable in offline/batch setting

→ Question: deal with model variance in **online setting**?

Accounting For Approximation

→ Supervised learning solves for fixed $f \in \mathscr{F}$

$$f^* = \underset{f \in \mathscr{F}}{\operatorname{argmin}} \, \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})]$$

→ Due to bias-variance tradeoff, not exactly what we want

⇒ instead, min. *both* avg. loss & surrogate for approx. err.

$$f^* = \underset{f \in \mathscr{F}}{\operatorname{argmin}} \, \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})] + \eta \mathbb{D}[\ell(f(\mathbf{x}), \mathbf{y})]$$

⇒ $\mathbb{D}[\ell(f(\mathbf{x}), \mathbf{y})]$ quantifies dispersion of estimate, e.g, variance

⇒ If dispersion is convex ⇒ coherent risk (term from OR/FE)

⇒ typically, risk is nonlinear function of an expected value

$$\text{Var}[\ell(f(\mathbf{x}), \mathbf{y})] = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left\{ \left(\ell(f(\mathbf{x}), \mathbf{y}) - \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ell(f(\mathbf{x}), \mathbf{y})] \right)_+^2 \right\}$$

⇒ an instance of *compositional stochastic programming*

Compositional Stochastic Optimization

$\rightarrow$ Risk-aware learning $\Rightarrow$ compositional stochastic opt.

$$\min_{f \in \mathscr{F}} \mathbb{E}_{\boldsymbol{\theta},\mathbf{y}^{\theta}}\Big[\ell\big(f(\boldsymbol{\theta}),\mathbf{y}^{\boldsymbol{\theta}},\mathbb{E}_{\boldsymbol{\xi},\mathbf{y}^{\xi}}\big[\mathfrak{h}(f(\boldsymbol{\xi}),\mathbf{y}^{\boldsymbol{\xi}})\big]\big)\Big] + \frac{\lambda}{2}\|f\|_{\mathcal{H}},$$

$\rightarrow$ Nested expectations $\Rightarrow$ func. stochastic *quasi-gradients*

$\rightarrow$ Two time-scale method

$\Rightarrow$ slower time-scale estimates inner-expectation

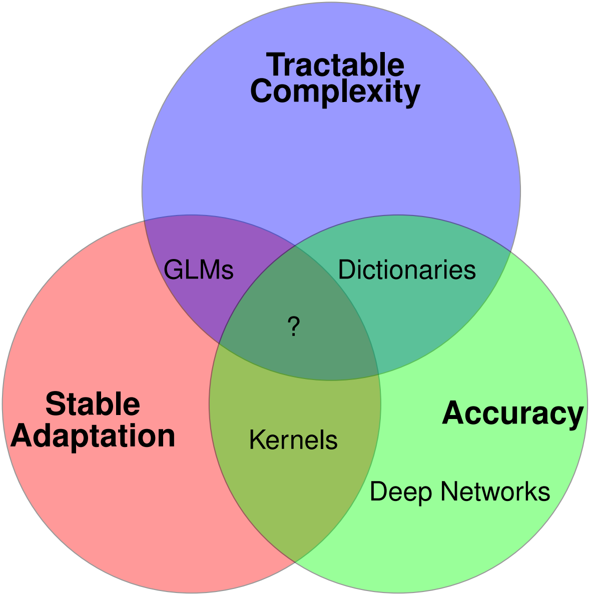
$\Rightarrow$ faster one does stochastic descent

$\rightarrow$ 80s stoch opt. (Korostelev, Ermoliev)

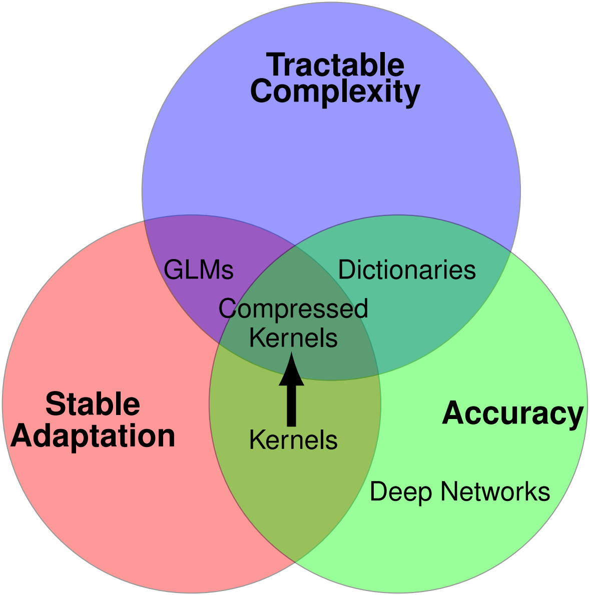
$\Rightarrow$ later heavily studied by Borkar, Tsitsiklis, Konda (97,'01, 04)

$\Rightarrow$ backbone of reinforcement learning (actor-critic, GTD)

On the Choice of $\mathscr{F}$



On the Choice of $\mathscr{F}$

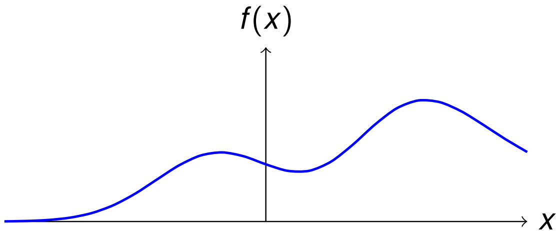


Reproducing Kernel Hilbert Space

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

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

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



→ Property (i) ⇒ Will allow us to compute derivatives

→ Kernel examples:

⇒ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$

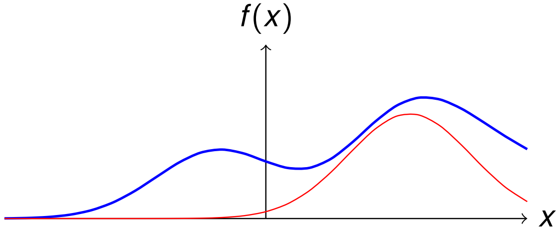
⇒ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Reproducing Kernel Hilbert Space

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

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

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



$\rightarrow$ Property (i) $\Rightarrow$ Will allow us to compute derivatives

$\rightarrow$ Kernel examples:

$\Rightarrow$ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$

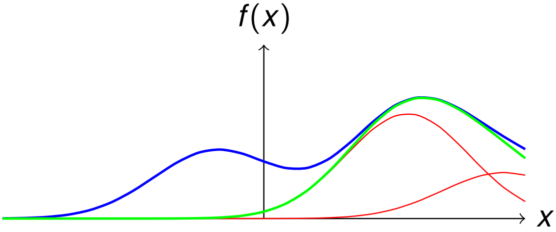
$\Rightarrow$ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Reproducing Kernel Hilbert Space

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

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

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



→ Property (i) ⇒ Will allow us to compute derivatives

→ Kernel examples:

⇒ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$

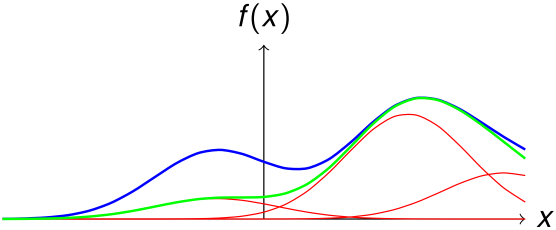
⇒ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Reproducing Kernel Hilbert Space

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

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

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



→ Property (i) ⇒ Will allow us to compute derivatives

→ Kernel examples:

⇒ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$

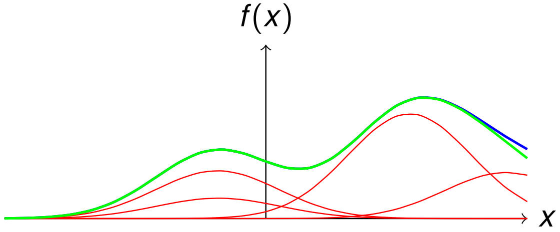
⇒ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Reproducing Kernel Hilbert Space

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

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

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



$\rightarrow$ Property (i) $\Rightarrow$ Will allow us to compute derivatives

$\rightarrow$ Kernel examples:

$\Rightarrow$ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$

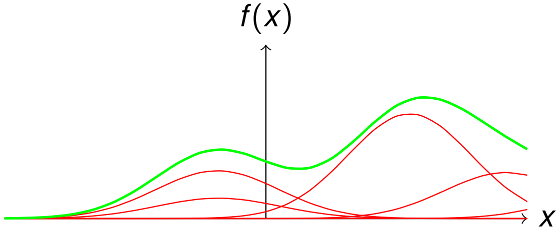
$\Rightarrow$ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Reproducing Kernel Hilbert Space

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

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

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



→ Property (i) ⇒ Will allow us to compute derivatives

→ Kernel examples:

⇒ Gaussian/RBF $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2c^2}\right\}$

⇒ polynomial $\kappa(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}' + b\right)^c$

Stochastic Quasi-Gradient

→ Objective

$$\min_{f \in \mathscr{F}} \mathbb{E}_{\boldsymbol{\theta}, \mathbf{y}^{\boldsymbol{\theta}}}\Big[\ell\big(f(\boldsymbol{\theta}), \mathbf{y}^{\boldsymbol{\theta}}, \mathbb{E}_{\boldsymbol{\xi}, \mathbf{y}^{\boldsymbol{\xi}}}[\mathfrak{h}(f(\boldsymbol{\xi}), \mathbf{y}^{\boldsymbol{\xi}})]\big)\Big] + \frac{\lambda}{2}\|f\|_{\mathcal{H}},$$

→ Apply SGD?

$$f_{t+1} = f_t - \eta_t \nabla_f \ell\big(f(\boldsymbol{\theta}_t), \mathbf{y}_t^{\boldsymbol{\theta}}, \mathbb{E}_{\boldsymbol{\xi}, \mathbf{y}^{\boldsymbol{\xi}}}\left[\mathfrak{h}(f(\boldsymbol{\xi}), \mathbf{y}^{\boldsymbol{\xi}})\right]\big) \nabla_f \mathfrak{h}(f(\boldsymbol{\xi}_t), \mathbf{y}_t^{\boldsymbol{\xi}}) \ .$$

⇒ stoch. grad. depends on $\mathbb{E}_{\boldsymbol{\xi}, \mathbf{y}^{\boldsymbol{\xi}}}\left[\mathfrak{h}(f(\boldsymbol{\xi}), \mathbf{y}^{\boldsymbol{\xi}})\right]$ ⇒ intractable

→ Define scalar estimate sequence g_t to track inner mean:

$$g_{t+1} = (1 - \beta_t) g_t + \beta_t \mathfrak{h}(f(\boldsymbol{\xi}_t), \mathbf{y}_t^{\boldsymbol{\xi}})$$

→ Replace inner mean in above stochastic gradient with g_{t+1}:

$$f_{t+1} = (1 - \lambda\alpha_t) f_t - \alpha_t \nabla_f \ell\big(f(\boldsymbol{\theta}_t), \mathbf{y}_t^{\boldsymbol{\theta}}, g_{t+1}\big) \nabla_f \mathfrak{h}(f(\boldsymbol{\xi}_t), \mathbf{y}_t^{\boldsymbol{\xi}}) \ ,$$

⇒ mitigate nonlinear interaction of inner and outer functions

→ This is stochastic *quasi-gradient* method

Compositional Online Learning with Kernels

→ Learning update rule

⇒ include latest data point

→ Compress w.r.t. metric

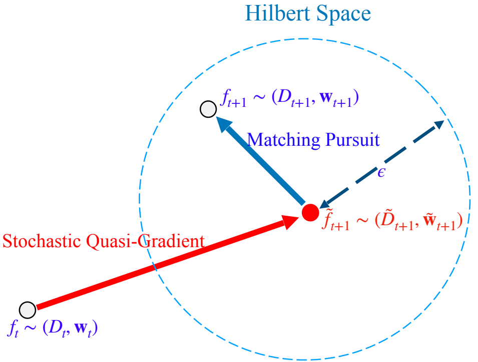
⇒ fix compression error ϵ

⇒ obtain reduced model

→ Similar to POLK

⇒ recursively avg. inner mean

⇒ plug into gradient direction



Convergence Results for COLK

	Diminishing	Constant
Learning rate	$\sum_{t=1}^{\infty} \alpha_t^2 + \beta_t^2 + \frac{\alpha_t^2}{\beta_t} < \infty$	$0 < \alpha < \beta < 1$
Compression	$\epsilon_t = \mathcal{O}(\alpha_t^2)$	$\epsilon = \mathcal{O}(\alpha^2)$
Regularization	$0 < \lambda$	$\lambda = \mathcal{O}(\alpha\beta^{-1} + 1)$
Convergence	$f_t \to f^*$ a.s.	$\inf \mathbb{E}\|f_t - f^*\|_{\mathcal{H}}^2 \to \mathcal{O}(\alpha)$
Model Order	None	Finite

Exact solution requires infinite memory, diminishing step-size

$\Rightarrow$ Approximate, but accurate solution with finite memory

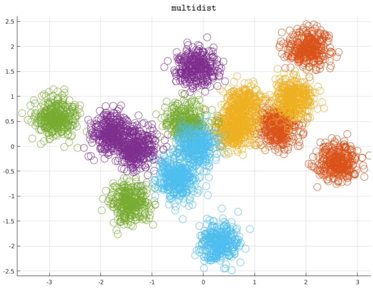
Online Multi-Class Kernel SVM

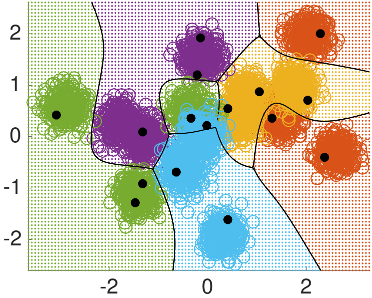
$\rightarrow$ Case where training examples for a fixed class

$\Rightarrow$ drawn from a distinct Gaussian mixture

$\rightarrow$ 3 Gaussians per mixture, $C = 5$ total classes

$\Rightarrow$ 15 total Gaussians generate data





Grid colors $\Rightarrow$ decision, bold black dots $\Rightarrow$ kernel dict. elements

$\rightarrow$ $\sim$ 96% accuracy

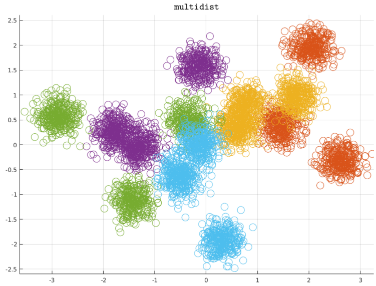
Online Multi-Class Kernel SVM

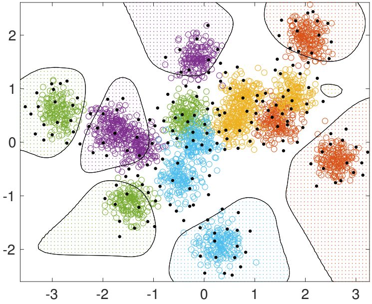
$\rightarrow$ Case where training examples for a fixed class

$\Rightarrow$ drawn from a distinct Gaussian mixture

$\rightarrow$ 3 Gaussians per mixture, $C = 5$ total classes

$\Rightarrow$ 15 total Gaussians generate data

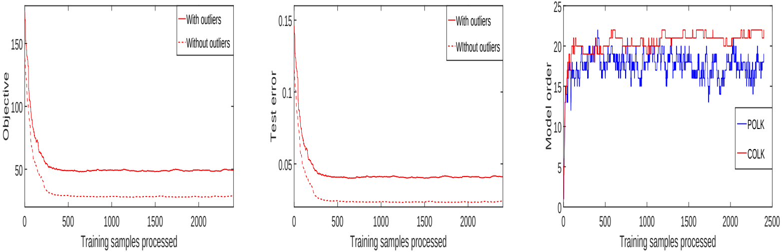




Grid colors $\Rightarrow$ decision, bold black dots $\Rightarrow$ kernel dict. elements

$\rightarrow$ risk constraint prevents confidence in areas of class overlap

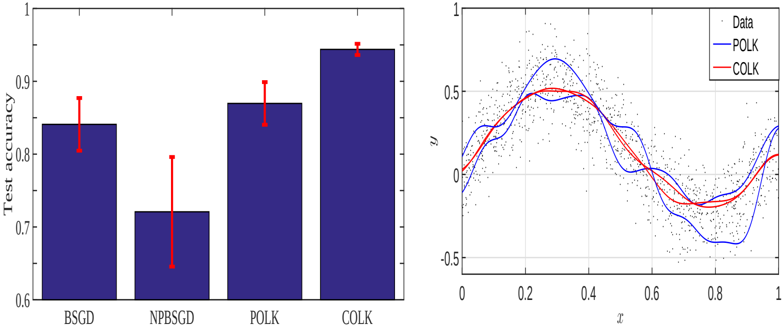
Synthetic Dataset Results



(a) Objective function (b) Test error (c) Model order

Figure: COLK for nonlinear regression without and with training outliers

Synthetic Dataset Results



(a) Statistical Accuracy Comparison (b) Visualization of regression function

Figure: COLK, with $\alpha = 0.02$, $\epsilon = \alpha^2$, $\beta = 0.01$, $K = 5$, $\eta = 0.1$, bandwidth $c = .06$ as compared to other methods.

Real Dataset Results

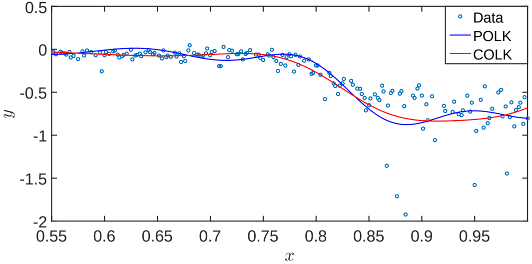
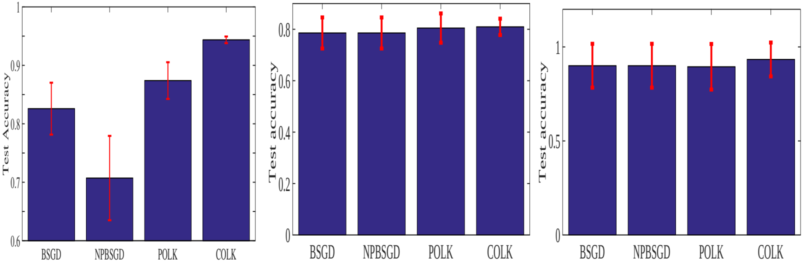


Figure: Interpolation on LIDAR dataset

Experimental Confidence



(a) Synthetic data (b) Sonar Data (c) Wine data

Figure: Online classification performance across training runs

Conclusion & Outlook

$\rightarrow$ Due to lack of big data, learning after deployment required

$\rightarrow$ How to make sure those approaches are reliable?

$\Rightarrow$ risk measures $\Rightarrow$ inoculate against rare events

$\Rightarrow$ but doing so yields compositional opt.

$\rightarrow$ New algorithm for compositional problems

$\Rightarrow$ for specific ML models: nonparametric/kernel method

$\rightarrow$ Stable, reliable, and consistent learning **online**

$\Rightarrow$ globally convergent, good experimental performance

References

→ Conferences

⇒ A. Koppel, A. S. Bedi, K. Rajawat, "Controlling the the Bias-Variance Tradeoff via Coherent Risk for Robust Learning with Kernels," in *IEEE American Control Conference* (to appear), Philadelphia, PA, July 10-12, 2019.

→ Journals

⇒ A. Bedi Singh, A. Koppel, and K. Rajawat. "Nonparametric Compositional Stochastic Optimization: Algorithms for Robust Online Learning with Kernels," in *IEEE Trans. Signal Process* (submitted), Feb. 2019.

Introducing Parsimony

→ Alternatively, function update written as

$$\tilde{f}_{t+1} = \underset{f\in\mathcal{H}}{\text{argmin}} \left\| f - \left[(1-\lambda\alpha_t)f_t - \alpha_t\langle {}'_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \ell'_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle\kappa(\boldsymbol{\xi}_t,\cdot)\right] \right\|_{\mathcal{H}}^2$$

$$= \underset{f\in\mathcal{H}_{\mathbf{U}_{t+1}}}{\text{argmin}} \left\| f - \left[(1-\lambda\alpha_t)f_t - \alpha_t\langle {}'_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \ell'_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle\kappa(\boldsymbol{\xi}_t,\cdot)\right] \right\|_{\mathcal{H}}^2 \quad (1)$$

→ Enforce parsimony by selecting dictionaries **D** such that $M_t \ll t$

⇒ Replace $\mathbf{U}_{t+1}$ by some other dictionary $\mathbf{D}_{t+1}$

$$f_{t+1} = \underset{f\in\mathcal{H}_{\mathbf{D}_{t+1}}}{\text{argmin}} \left\| f - \left((1-\lambda\alpha_t)f_t - \alpha_t\langle {}'_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \ell'_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle\kappa(\boldsymbol{\xi}_t,\cdot)\right) \right\|_{\mathcal{H}}^2$$

$$:= \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}}\left[(1-\lambda\alpha_t)f_t - \alpha_t\langle {}'_{\boldsymbol{\xi}_t}(f(\boldsymbol{\xi}_t)), \ell'_{\boldsymbol{\theta}_t}(\mathbf{g}_{t+1})\rangle\kappa(\boldsymbol{\xi}_t,\cdot)\right] \quad (2)$$