# APPM 7400 (special topics) “Theory of Machine Learning” Prof. Becker, spring 2020

# Student projects

https://github.com/stephenbeckr/ML-theory-class

## Data-Driven Model Predictive Control

(Felipe Galarza-Jimenez)

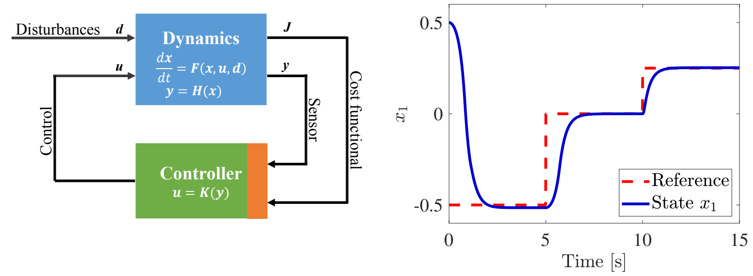



Fig. 3: Inverted pendulum without friction and MPC feedback controller.

## PAC-Bayesian Framework

(Ayoub Ghriss)

$\Delta : [0,1] \times [0,1] \mapsto \mathbb{R}$ is a convex function

**General theorem (Bégin et al, Germain (2014,2015))**

For any prior P, any $\delta$, and any $\Delta$, we have w.p $\geq 1 - \delta$:

$$\forall Q \text{ on } \mathcal{H} : \Delta(R_{out}(Q), R_{in}(Q)) \leq \frac{1}{m}\Big[D_{\mathrm{KL}}(Q \parallel P) + \log \frac{\mathcal{J}_\Delta(m)}{\delta}\Big],$$

where

$$\mathcal{J}_\Delta(m) = \sup_{r \in [0,1]} \Big[\sum_{k=0}^{m} \binom{m}{k} r^k (1-r)^{m-k} \exp m\Delta(k/m, r)\Big]$$

## Multiclass Problems and Linear Multiclass Predictors

(Stevan Maksimovic)

In the one-vs-all method, we train $k$ binary classifiers, each of which discriminates between one class and the rest of the classes. Given a training set $S$, we construct $k$ binary training sets, $S_1, \ldots, S_k$ where $S_i = ((x_j, (-1)^{1_{y_j \neq i}}))_{j=1}^{m}$). In other words, $S_i$ is the set of instances labeled 1 if their label in $S$ is $i$, and $-1$ otherwise. For each $i \in [k]$ we train a binary predictor $h_i : X \to \{\pm 1\}$ based on $S_i$. Then, given $h_1, \ldots, h_k$, we construct a multiclass predictor using the rule

$$h(\mathbf{x}) \in \operatorname{argmax}_{i \in Y} h_i(\mathbf{x}) \qquad (1)$$

## Distributed and Inexact Proximal Gradient Method for Online Convex Optimization

(Amirhossein Ajalloeian)

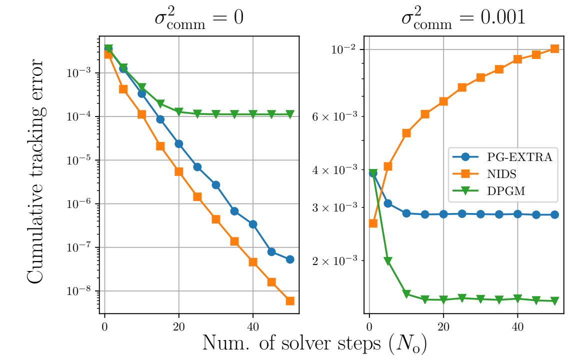
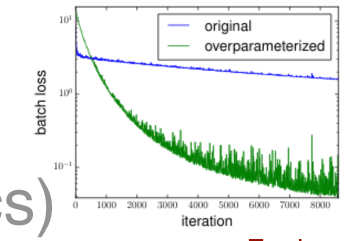



Fig. 1. Comparison in terms of cumulative tracking error of DPGM (proposed in this paper), PG-EXTRA [20], and NIDS [21] for a time-varying sparse linear regression problem, without and with communication errors. The step-size of each algorithm is chosen as $0.5\bar{\alpha}$.

## Implicit Acceleration by Overparameterization (Arora, Cohen, Hazan 2018)

(Mike McCabe)




## A Review of “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks” by Finn et. al.

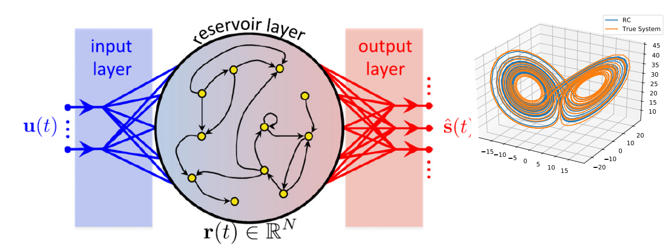
(C. Alexander Hirst)

“Learning to learn”: How can we make machine learning algorithms flexible, able to adapt to new tasks quickly (with little training data)?

- A hallmark of human intelligence, we are able to leverage relevant past experiences to quickly learn (i.e. driving in town, with small car -> driving in city, with SUV)
- For AI agents deployed to the real-world, they will have to adapt to unseen data quickly and safely
- “Life-long Learning”
- Few-shot Meta-Learning: Train a model that can quickly adapt to a new task using only a few training points and training iterations.

## Exploration of the Bias-Variance Trade-off with Respect to the Size and Training Regularization of a Reservoir Computer

(Nicholas Landry)



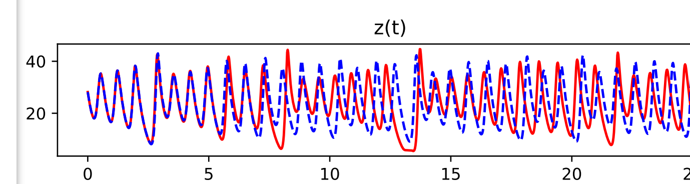




Figure 4: Reservoir computer prediction (red) and the true dynamical system (blue

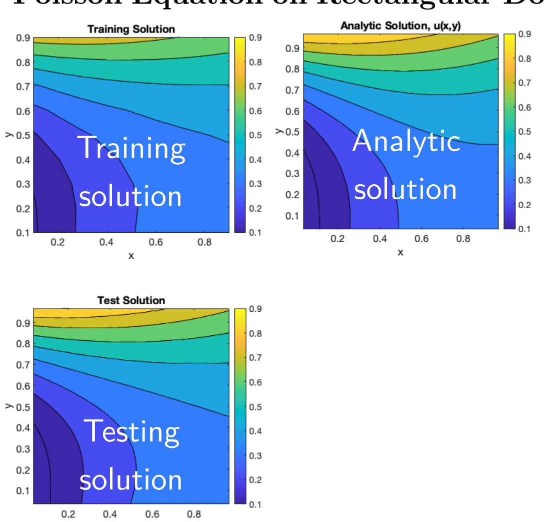
## Student backgrounds:

- Applied Math (BS/MS/PhD)
- Math (PhD)
- Engineering Physics (BS)
- Computer Science (PhD)
- Electrical & Comp. Eng. (BS/PhD)
- Aerospace (PhD)
- Mechanical Eng. (BS)

## Analysis of Least Squares Support Vector Machines for Learning Solutions to Partial Differential Equations

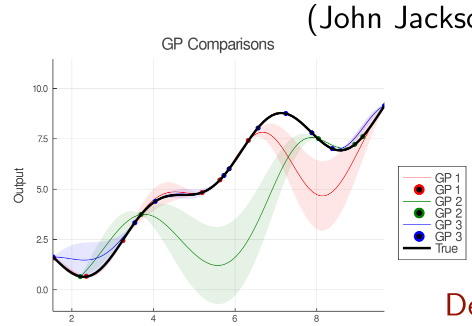
(DeAnna Gilchrist)

**Poisson Equation on Rectangular Domain.**




## Review of Error Bounds for Gaussian Process Regression

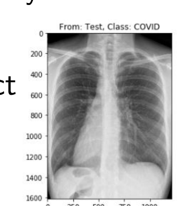
(John Jackson)

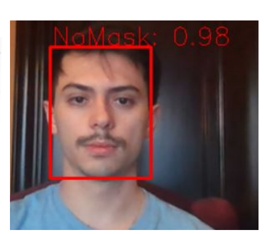


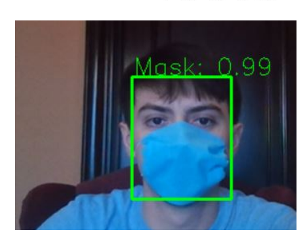

## Deep Learning against COVID-19

(Arturo Freydig Avila)

- classification of infected lungs via x-ray data using convolutional neural nets
- Recurrent Neural Networks to predict spread of COVID-19 in Mexico
- Mask detection in images

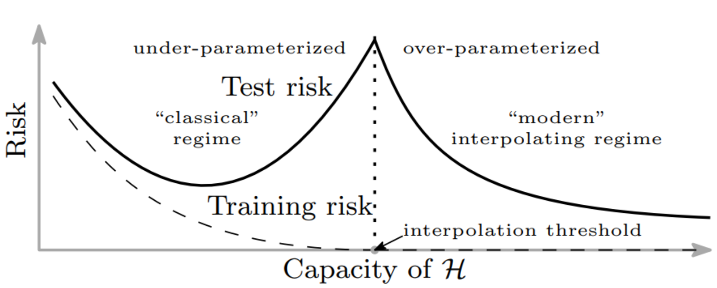


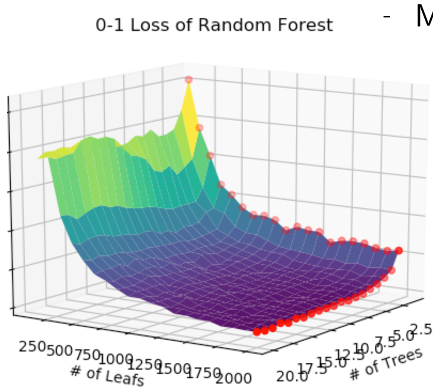




## Empirical Evidence for the Double Descent Curve

(Aidan Bohenick and Evan Gorman)




0-1 Loss of Random Forest




## Generating an Operation on Embeddings Using Neural Networks

(Jordan DuBeau and Albany Thompson)

The goal of our project was to design a neural network that could help create an **embedding algebra**. An embedding algebra consists of a set $E$ of strictly increasing functions $\mathbb{N} \to \mathbb{N}$ (we call the members of $E$ **embeddings**), together with an operation $*$ such that the following conditions hold for all embeddings $a, b, c$:

1. $a * b$ is an embedding,
2. if $b$ is not the identity function, then $\operatorname{crit}(a * b) = a(\operatorname{crit}(b))$ (see below), and
3. $a * (b * c) = (a * b) * (a * c)$.

In item (2), we understand $\operatorname{crit}(f)$ for a strictly increasing function $f : \mathbb{N} \to \mathbb{N}$ to be the smallest $n$ for which $f(n) > n$, called the **critical point** of $f$. Every strictly increasing function $\mathbb{N} \to \mathbb{N}$ has a critical point, with the exception of the identity function.

| # of Samples | Num epochs | HL activation | Dropout | Metric | Training Error | Testing error |
|---|---|---|---|---|---|---|
| 9.00E+04 | 12 | tanh | 0.2 | accuracy | 13061.05583 | 13253.61704 |
| 9.00E+04 | 12 | sigmoid | 0.2 | accuracy | 12771.9213 | 12864.02219 |