BAYESIAN GLOBAL OPTIMIZATION

Using Optimal Learning to Tune Deep Learning Pipelines

Scott Clark

scott@sigopt.com

OUTLINE

1. Why is Tuning AI Models Hard?
2. Comparison of Tuning Methods
3. Bayesian Global Optimization
4. Deep Learning Examples
5. Evaluating Optimization Strategies

Deep Learning / AI is
extremely powerful

Tuning these systems is
extremely non-intuitive

What is the most important unresolved problem in machine learning?

"...we still don't really know why some configurations of deep neural networks work in some case and not others, let alone having a more or less **automatic approach to determining the architectures and the hyperparameters**."

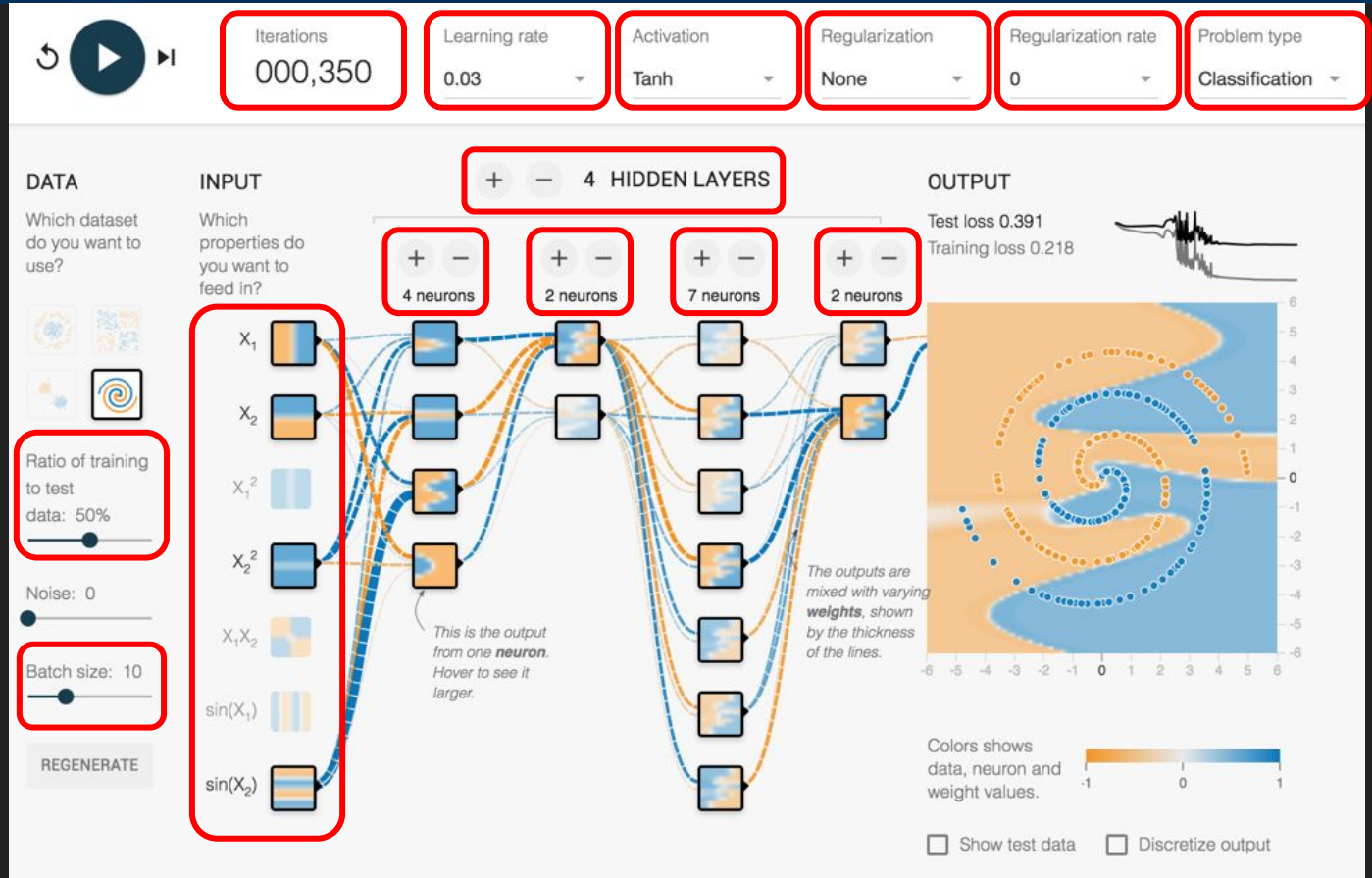
Xavier Amatriain, *VP Engineering at Quora*

(former Director of Research at Netflix)

https://www.quora.com/What-is-the-most-important-unresolved-problem-in-machine-learning-3

Photo: Joe Ross

TUNABLE PARAMETERS IN DEEP LEARNING



TUNABLE PARAMETERS IN DEEP LEARNING

mxnet

class `mxnet.optimizer.` **`RMSProp`** ***(learning_rate=0.001, gamma1=0.9, gamma2=0.9, epsilon=1e-08, centered=False, clip_weights=None, **kwargs)***

The RMSProp optimizer.

Two versions of RMSProp are implemented:

If `centered=False`, we follow http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf by Tieleman & Hinton, 2012.

If `centered=True`, we follow http://arxiv.org/pdf/1308.0850v5.pdf (38)-(45) by Alex Graves, 2013.

This optimizer accepts the following parameters in addition to those accepted by **`Optimizer`**:

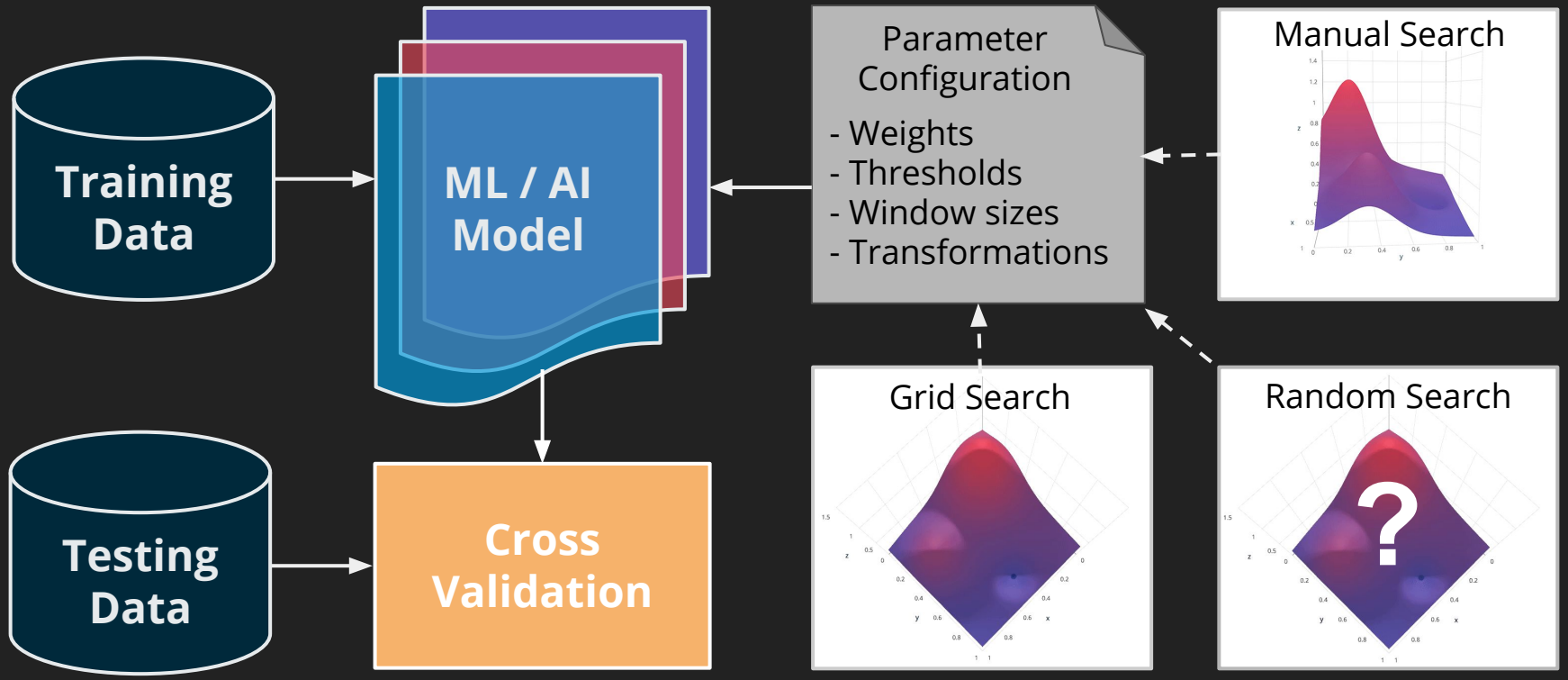
Parameters:

- **gamma1** (*float, optional*) – Decay factor of moving average for `gradient^2`.
- **gamma2** (*float, optional*) – A "momentum" factor. Only used if `centered=True`.
- **epsilon** (*float, optional*) – Small value to avoid division by 0.
- **centered** (*bool, optional*) – Use Graves' or Tieleman & Hinton's version of RMSProp.
- **clip_weights** (*float, optional*) – clip weights into range `[-clip_weights, clip_weights]`

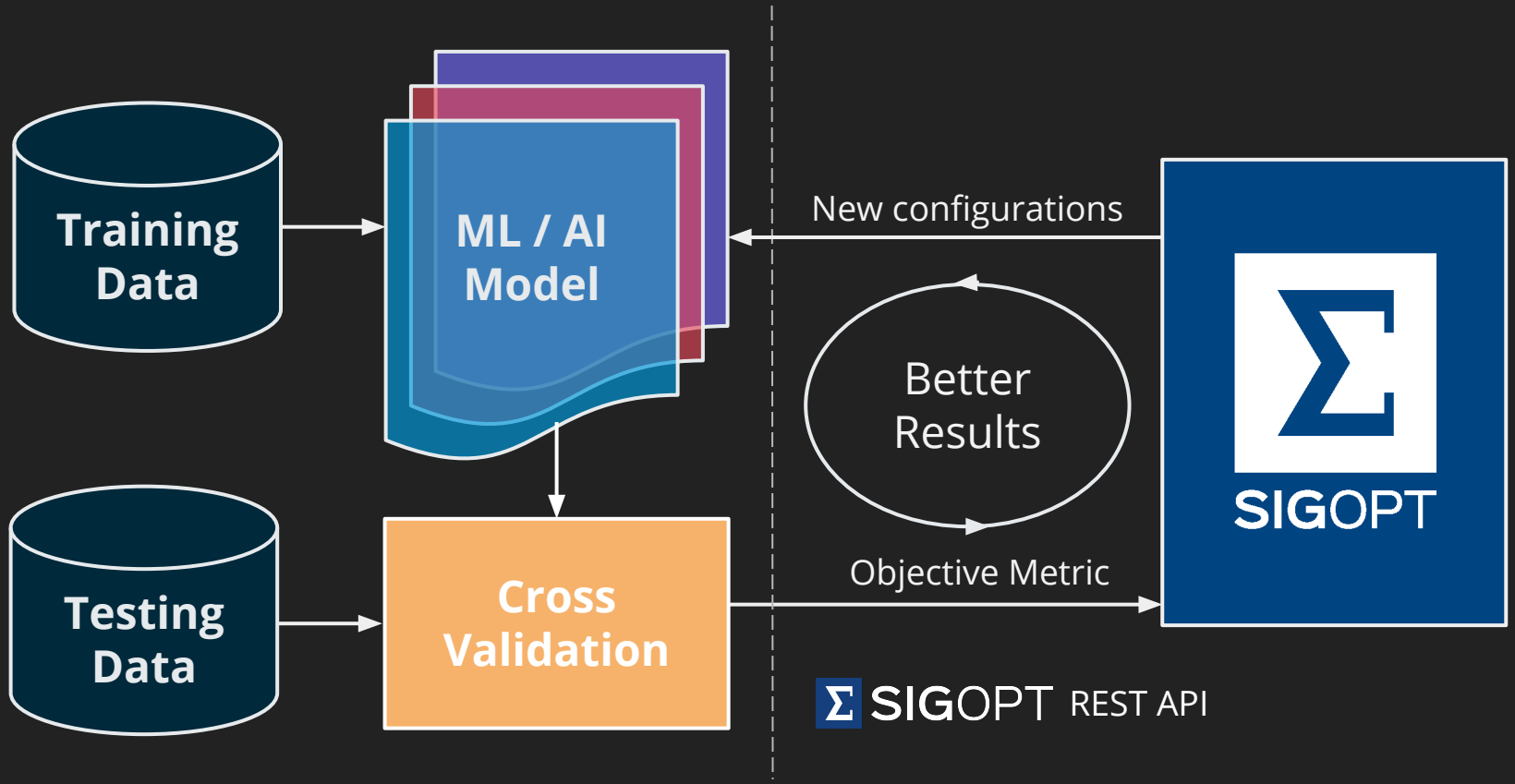
Photo: Tammy Strobel

STANDARD METHODS FOR HYPERPARAMETER SEARCH

STANDARD TUNING METHODS



OPTIMIZATION FEEDBACK LOOP



BAYESIAN GLOBAL OPTIMIZATION

OPTIMAL LEARNING

… the challenge of how to **collect information as efficiently as possible**, primarily for settings where collecting information is **time consuming and expensive**.

Prof. Warren Powell - Princeton

What is the **most efficient way to collect information**?

Prof. Peter Frazier - Cornell

How do we **make the most money**, as fast as possible?

Scott Clark - CEO, SigOpt

BAYESIAN GLOBAL OPTIMIZATION

- Optimize objective function
  - Loss, Accuracy, Likelihood
- Given parameters
  - Hyperparameters, feature/architecture params
- Find the best hyperparameters
  - Sample function as few times as possible
  - Training on big data is expensive

HOW DOES IT WORK?

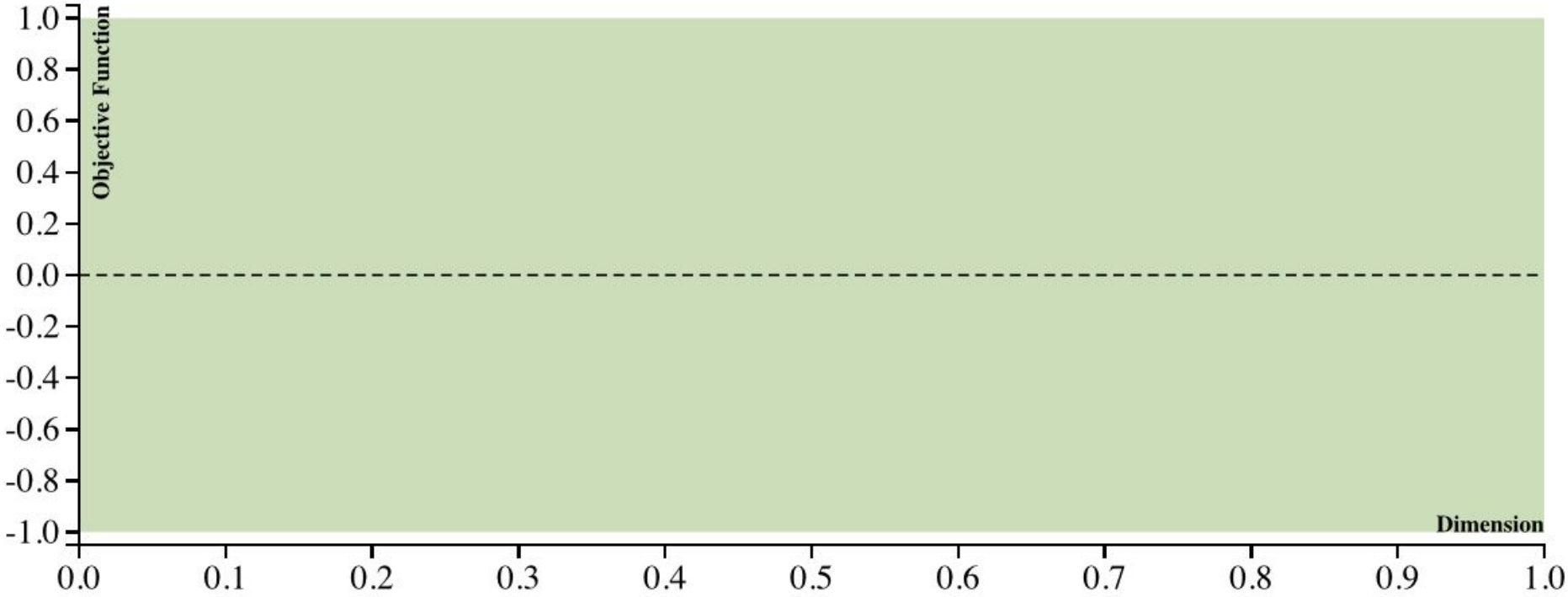
SMBO

Sequential **M**odel-**B**ased **O**ptimization

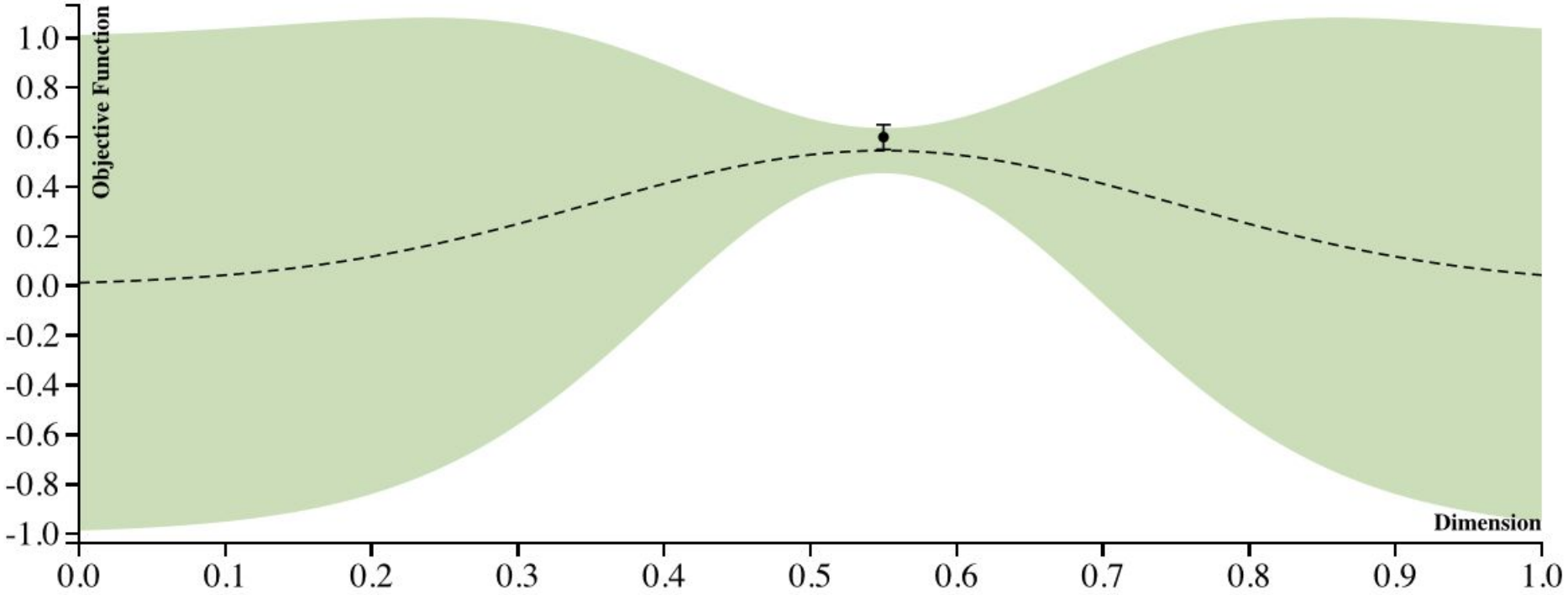
GP/EI SMBO

1. Build Gaussian Process (GP) with points sampled so far
2. Optimize the fit of the GP (covariance hyperparameters)
3. Find the point(s) of highest Expected Improvement within parameter domain
4. Return optimal next best point(s) to sample

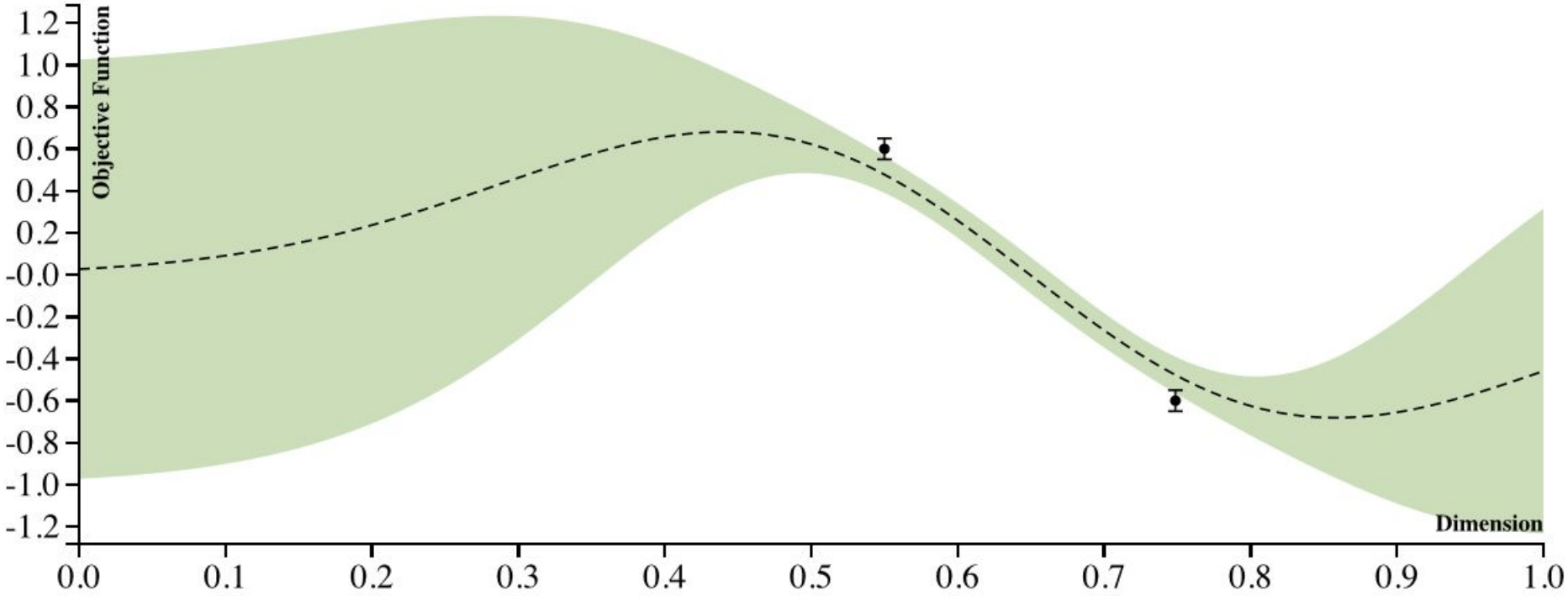
GAUSSIAN PROCESSES



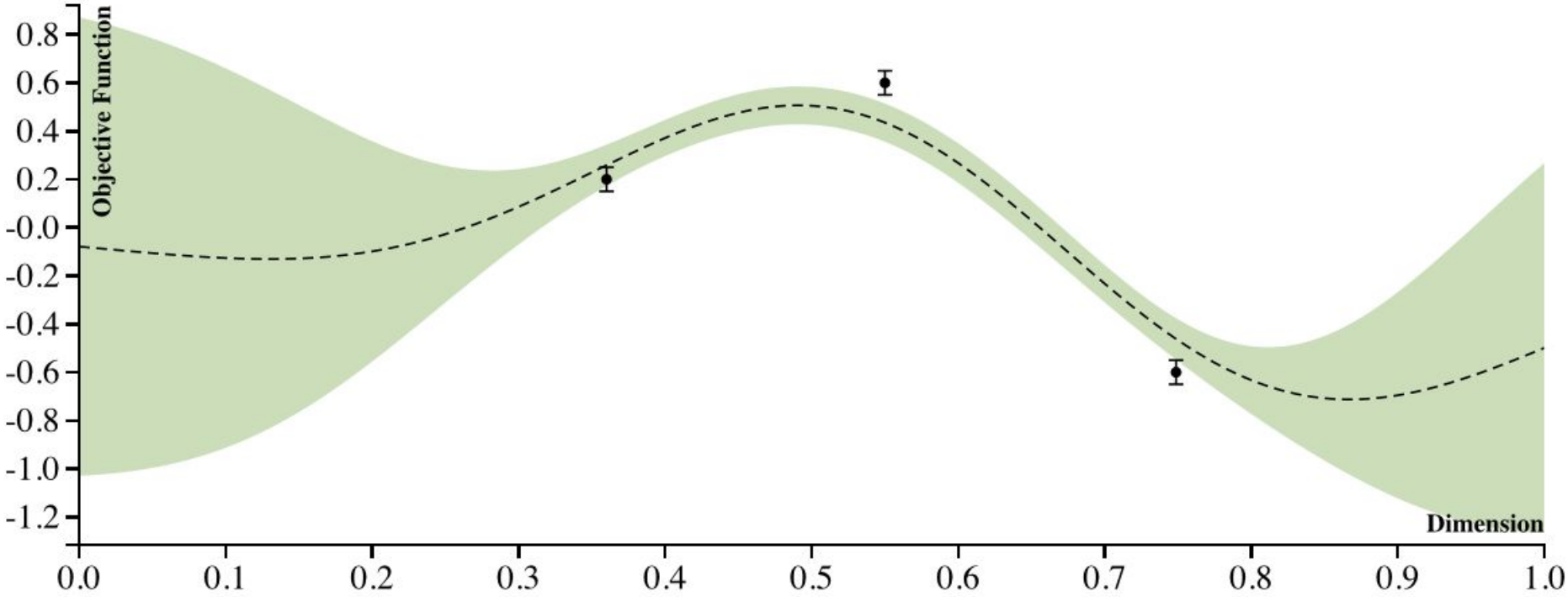
GAUSSIAN PROCESSES



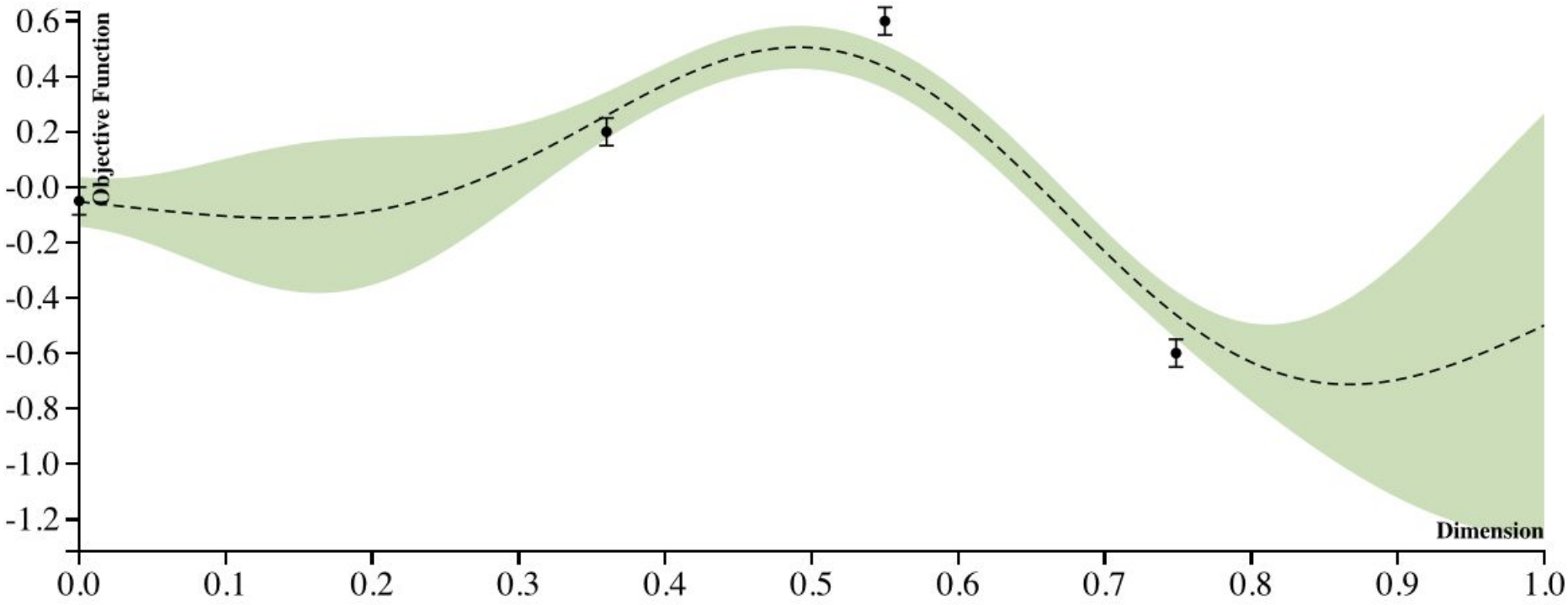
GAUSSIAN PROCESSES



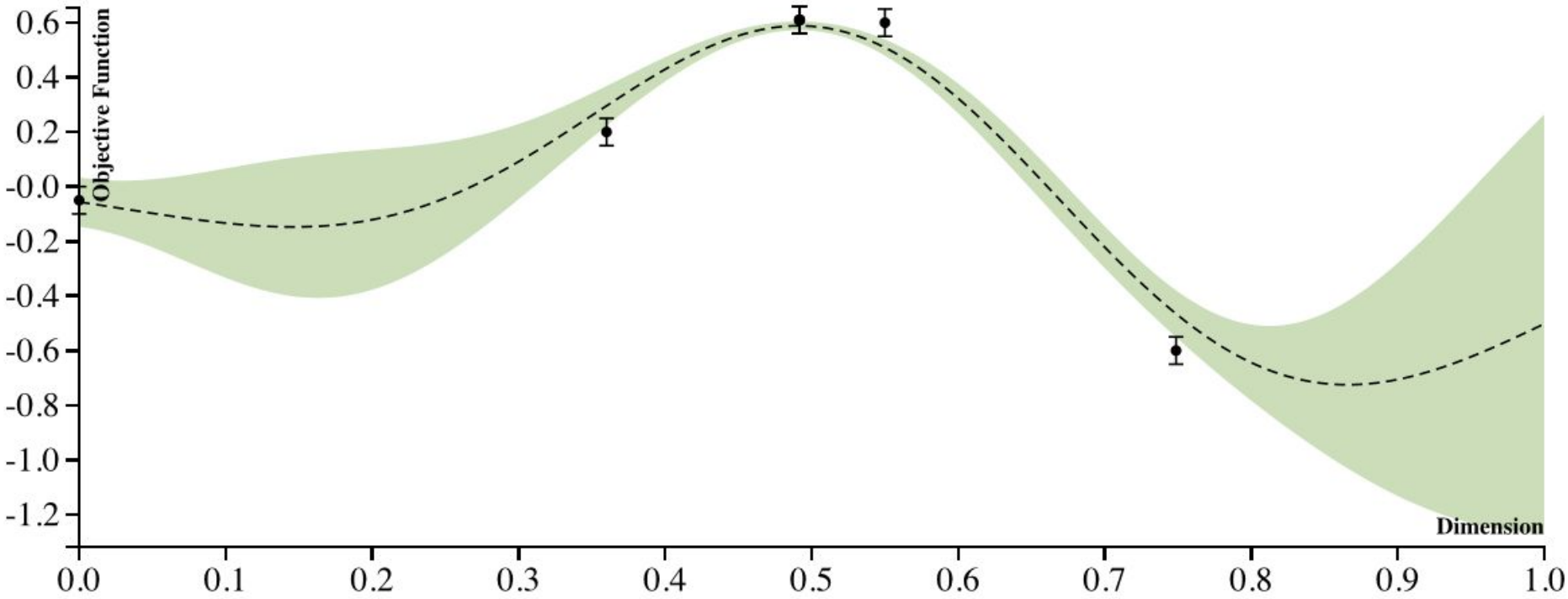
GAUSSIAN PROCESSES



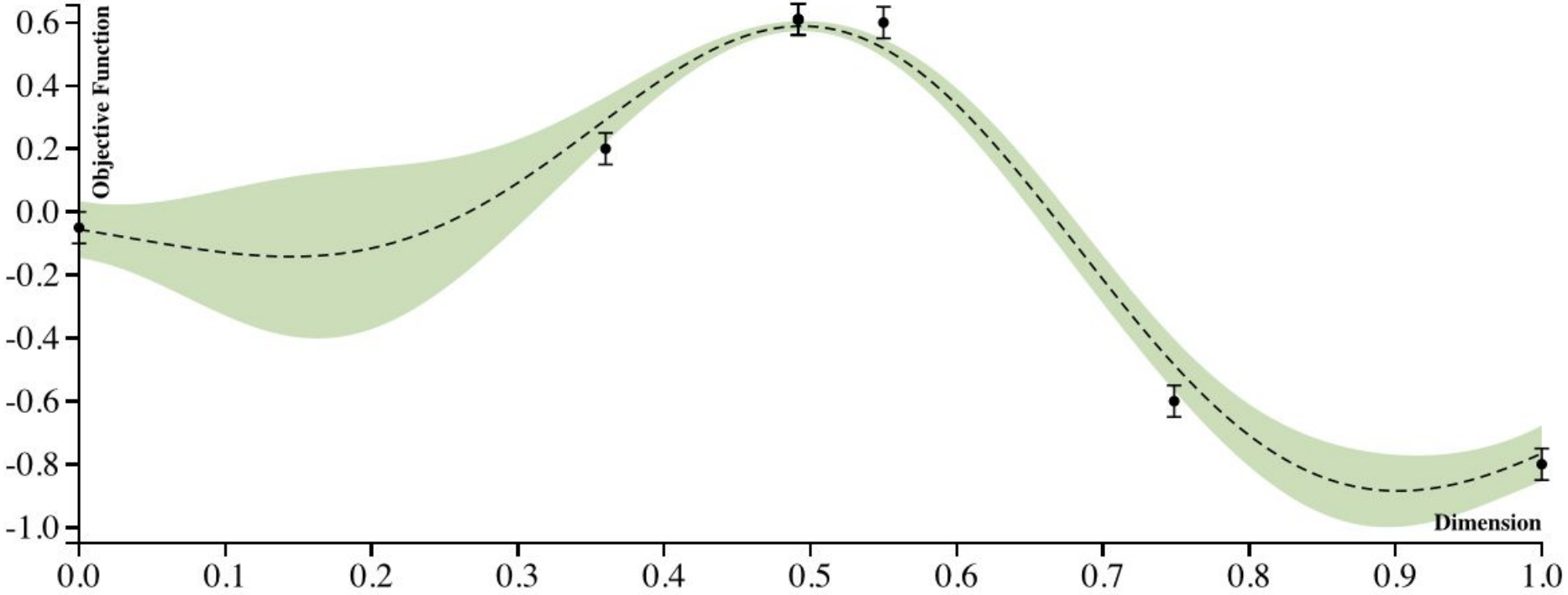
GAUSSIAN PROCESSES



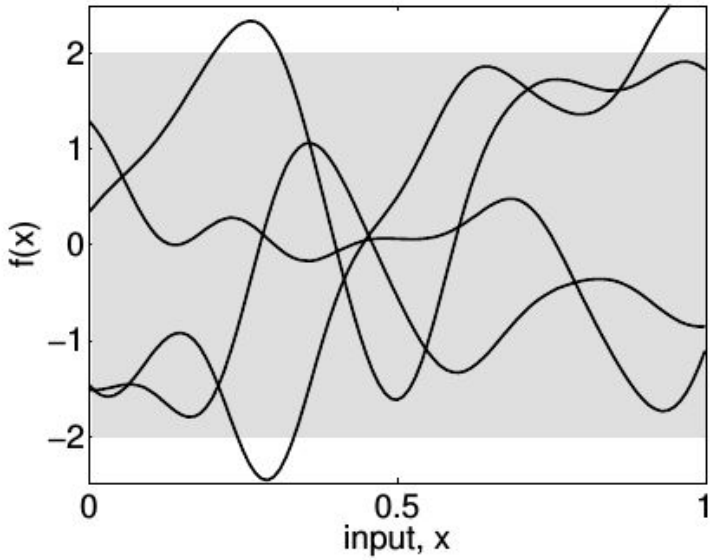
GAUSSIAN PROCESSES



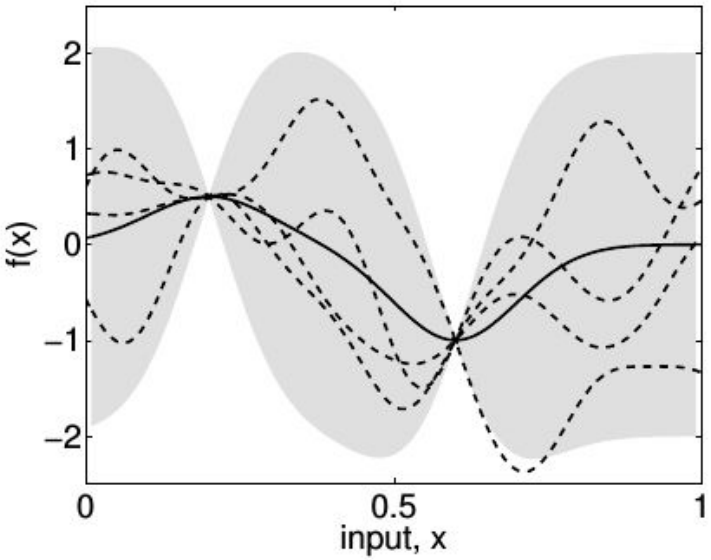
GAUSSIAN PROCESSES



GAUSSIAN PROCESSES

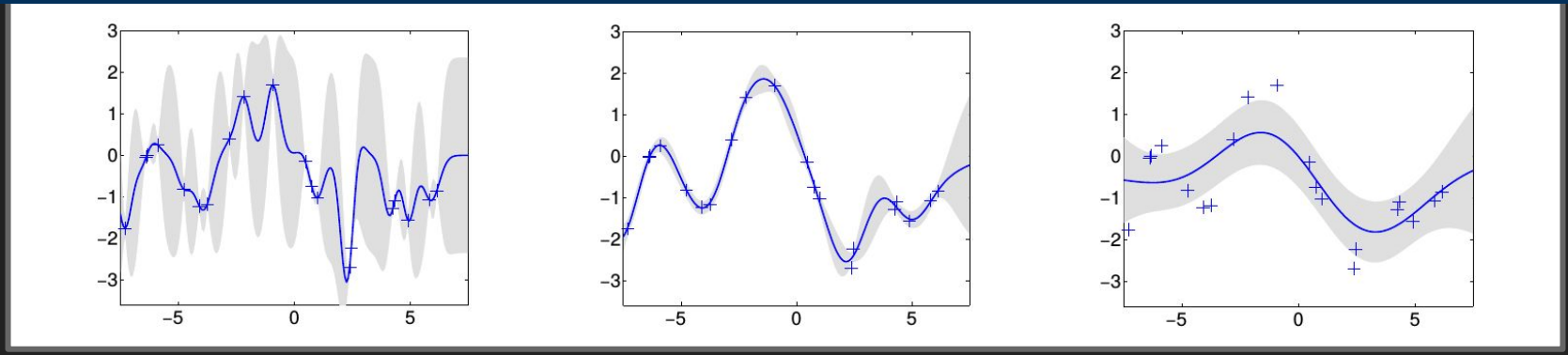


(a), prior

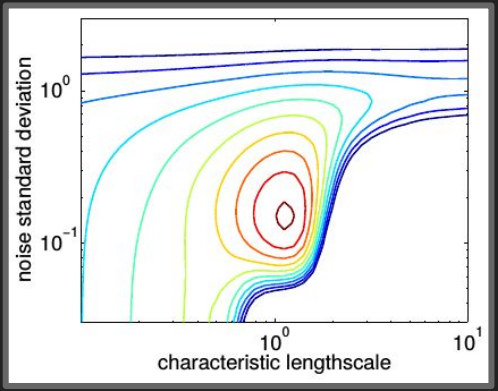


(b), posterior

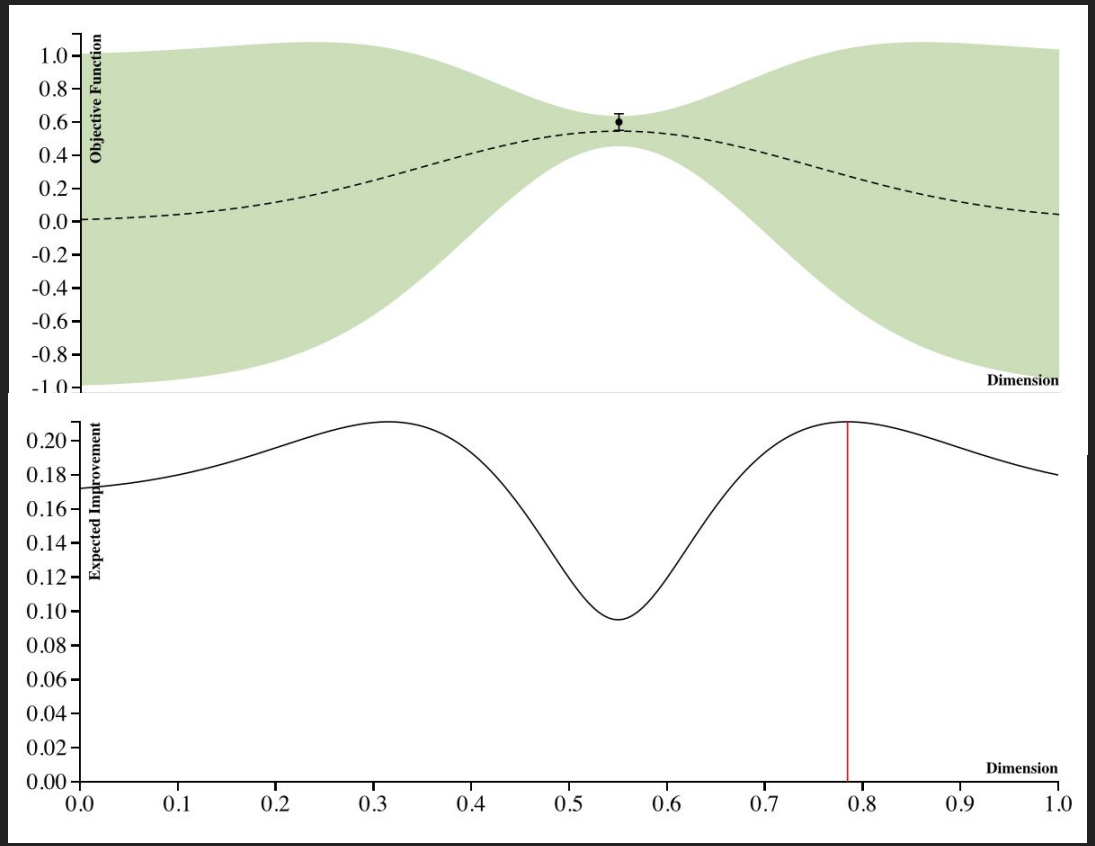
GAUSSIAN PROCESSES



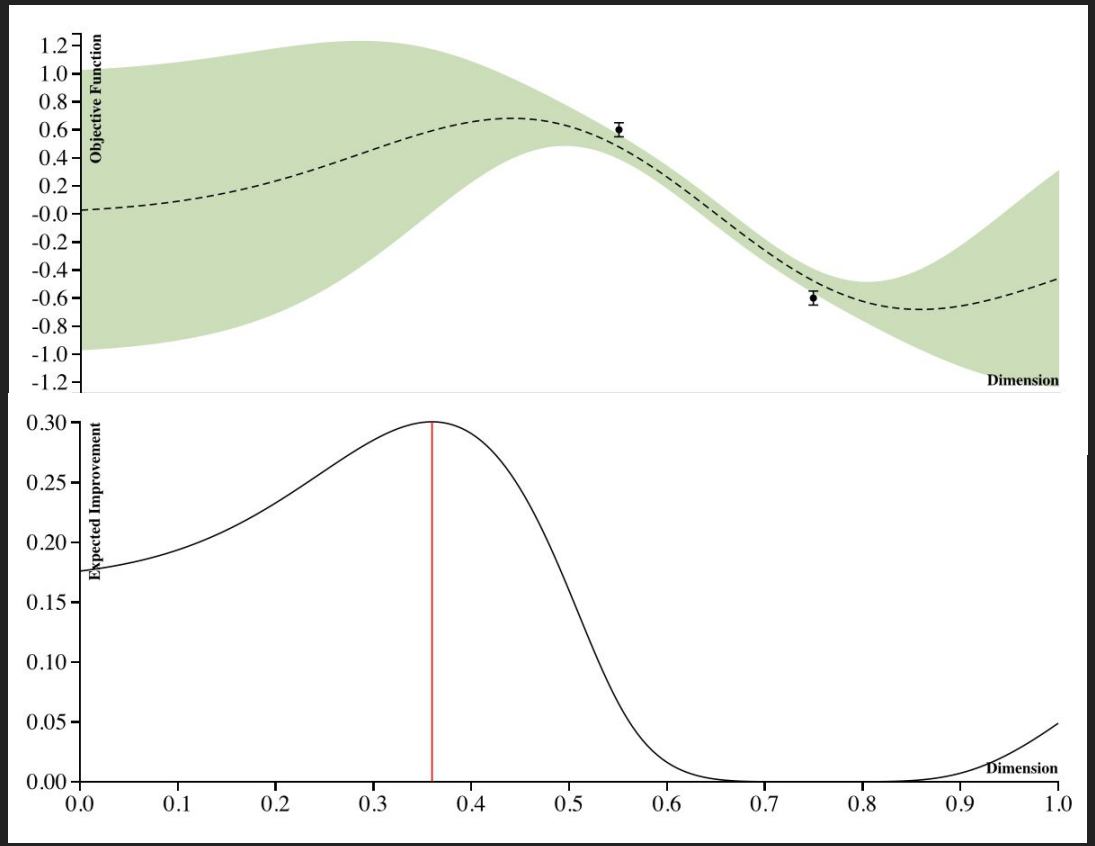
overfit **good fit** **underfit**



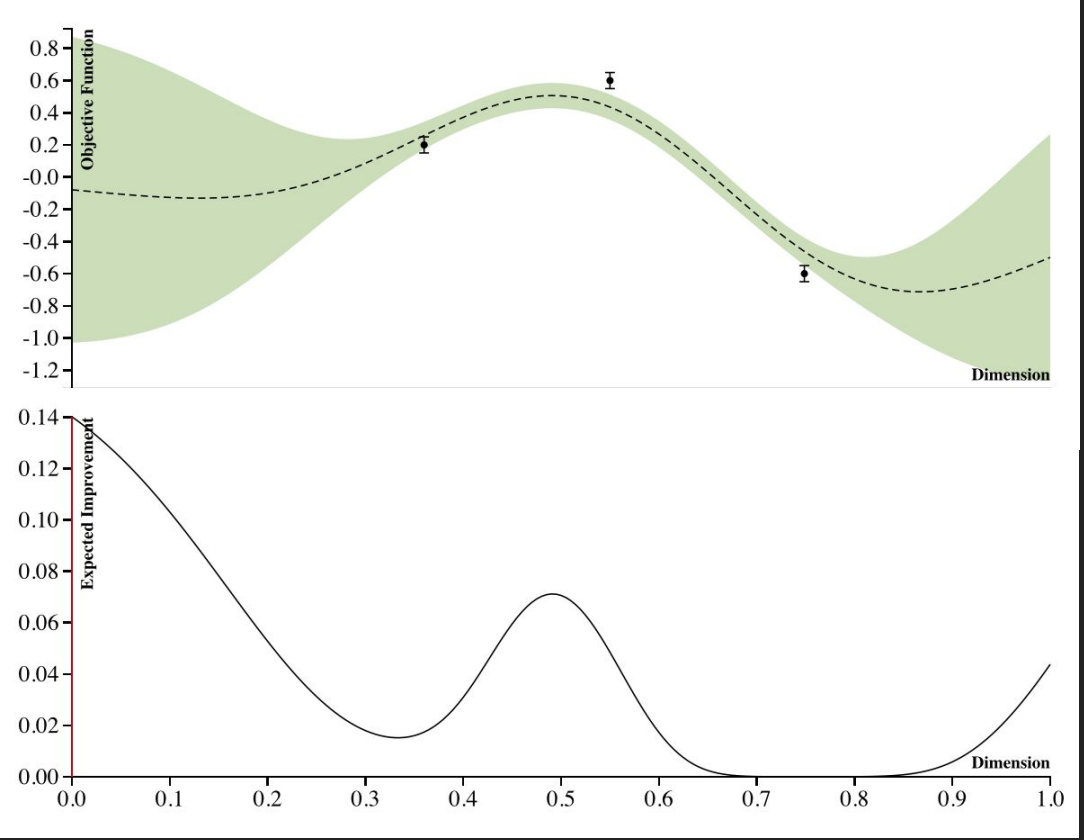
EXPECTED IMPROVEMENT



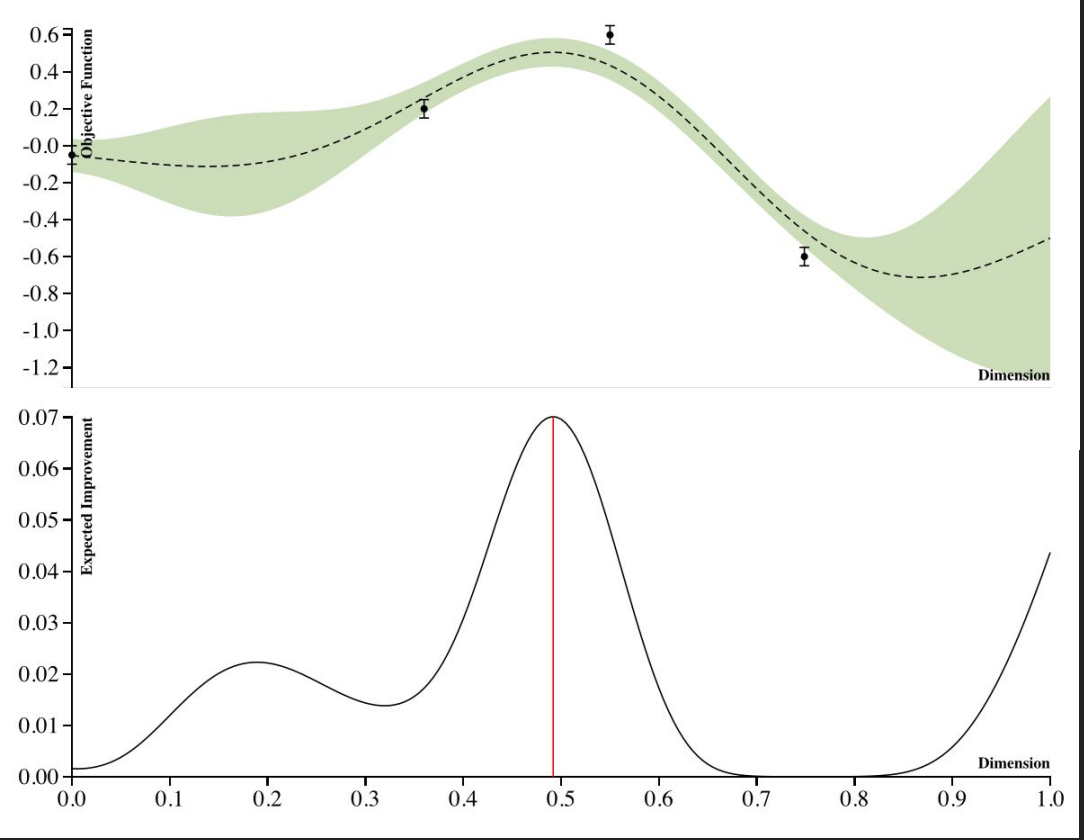
EXPECTED IMPROVEMENT



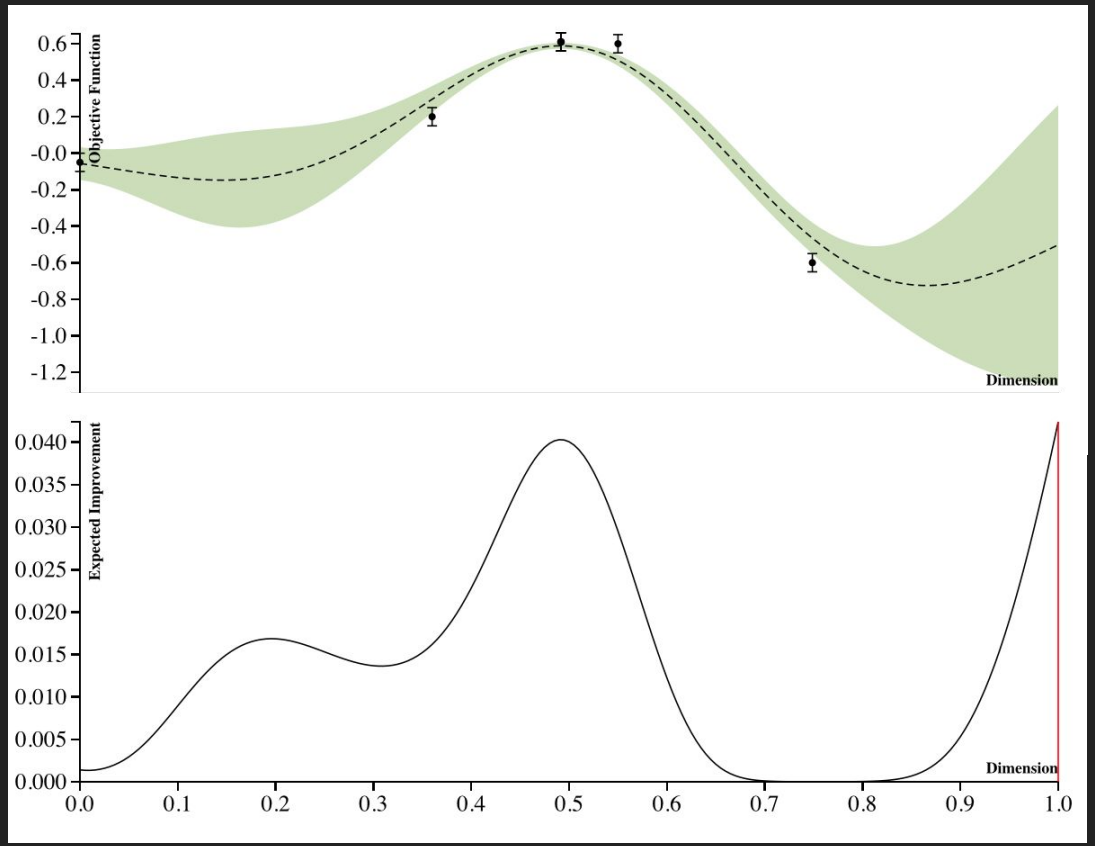
EXPECTED IMPROVEMENT



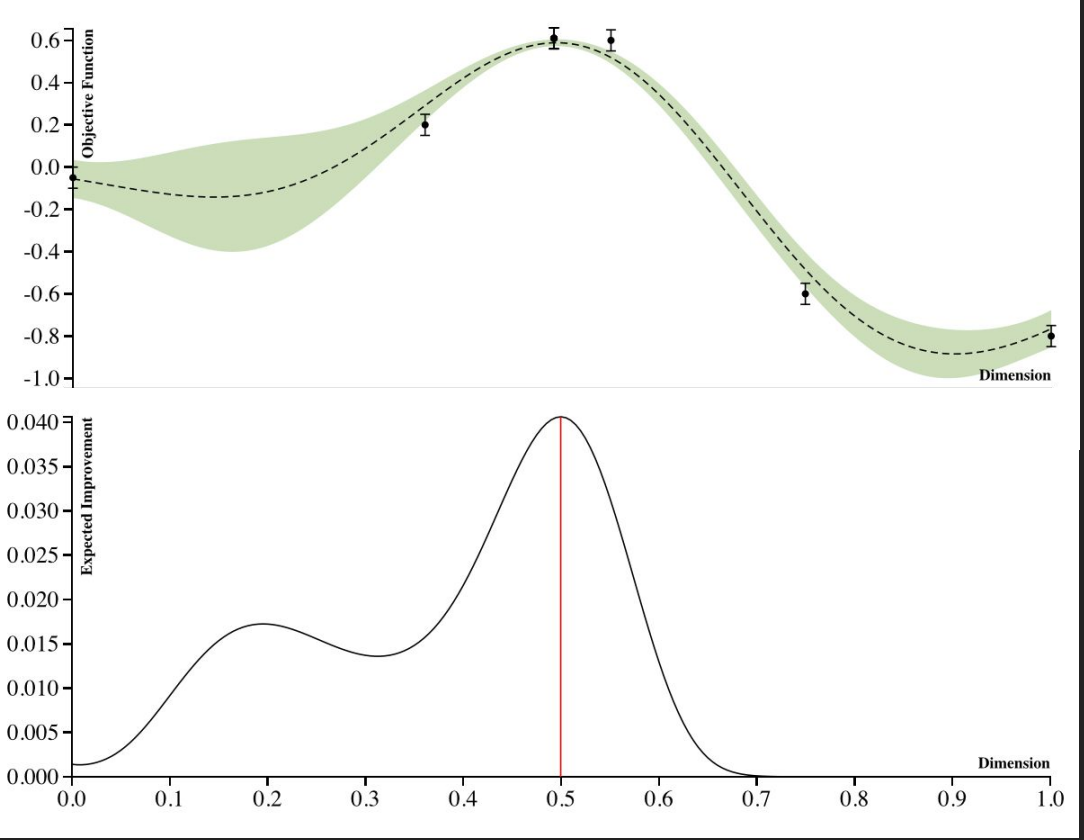
EXPECTED IMPROVEMENT



EXPECTED IMPROVEMENT



EXPECTED IMPROVEMENT



DEEP LEARNING EXAMPLES

SIGOPT + MXNET

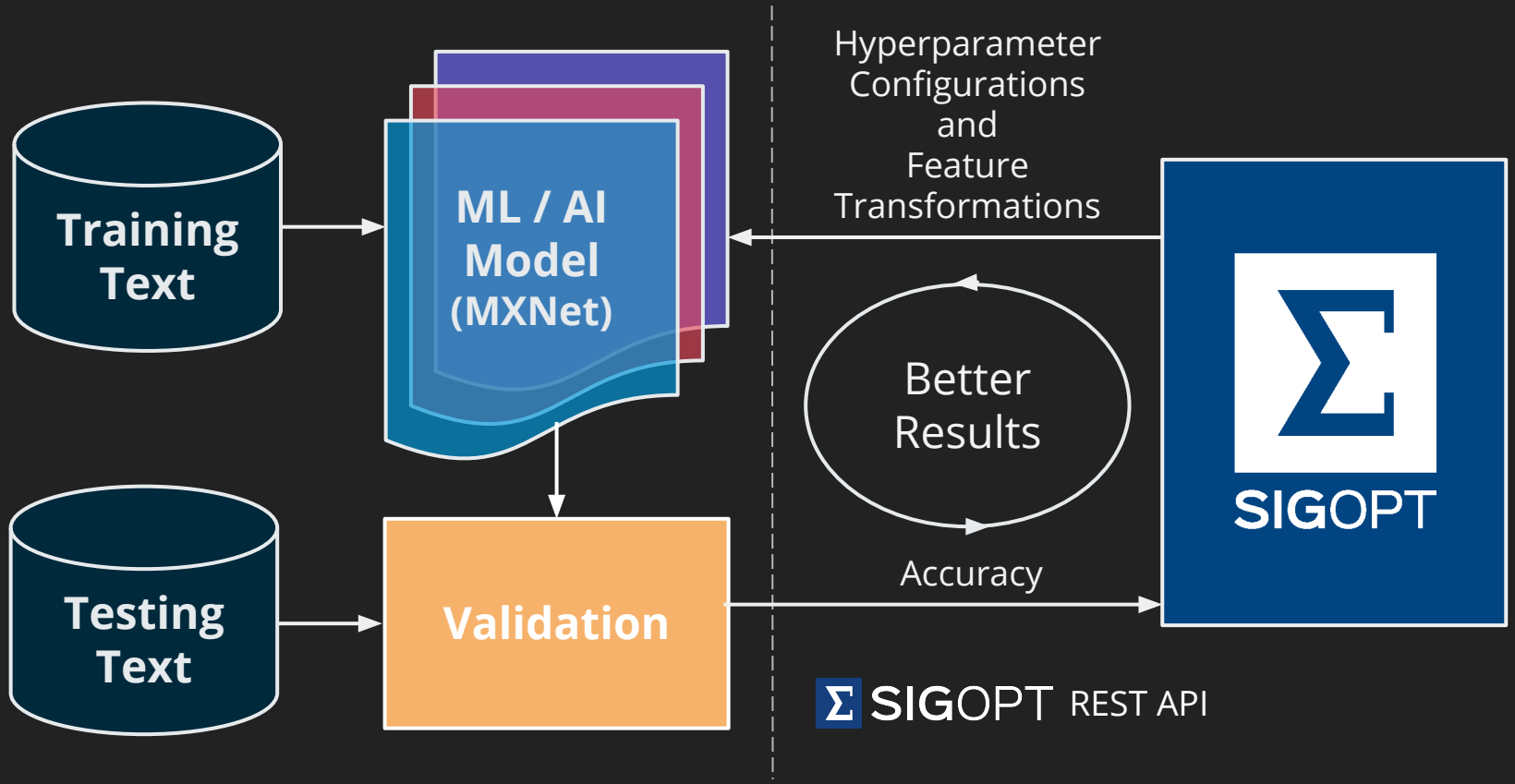
AWS AI Blog

Fast CNN Tuning with AWS GPU Instances and SigOpt

by Steven Tartakovsky | on 01 MAY 2017 | Permalink | Comments

- Classify movie reviews using a CNN in MXNet

TEXT CLASSIFICATION PIPELINE



TUNABLE PARAMETERS IN DEEP LEARNING

mxnet

class `mxnet.optimizer.` **`RMSProp`** ***(learning_rate=0.001, gamma1=0.9, gamma2=0.9, epsilon=1e-08, centered=False, clip_weights=None, **kwargs)***

The RMSProp optimizer.

Two versions of RMSProp are implemented:

If `centered=False`, we follow http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf by Tieleman & Hinton, 2012.

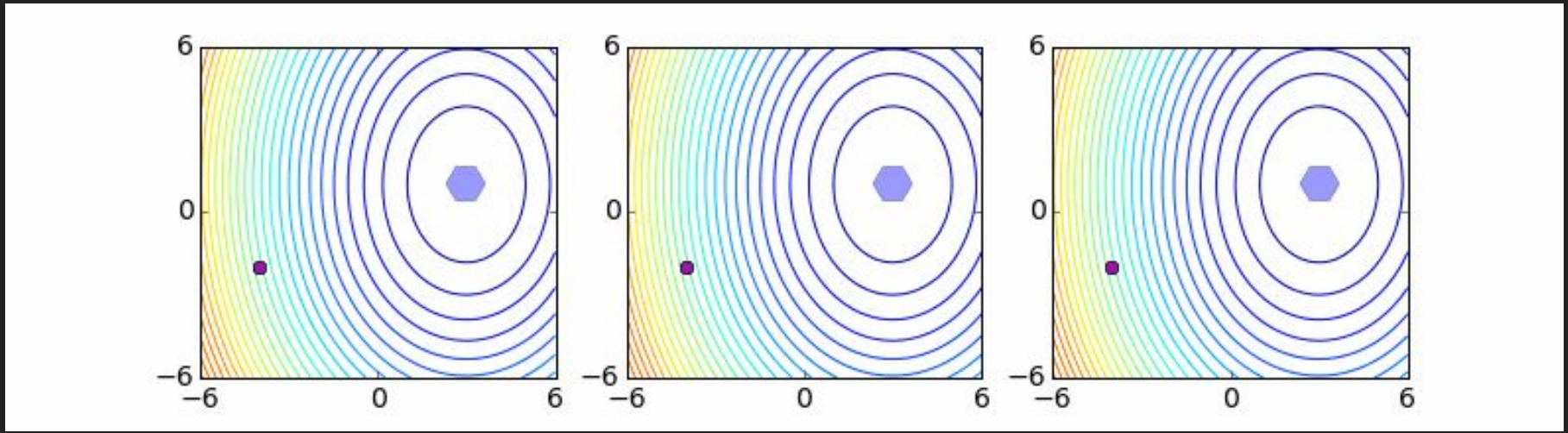
If `centered=True`, we follow http://arxiv.org/pdf/1308.0850v5.pdf (38)-(45) by Alex Graves, 2013.

This optimizer accepts the following parameters in addition to those accepted by **`Optimizer`**:

Parameters:

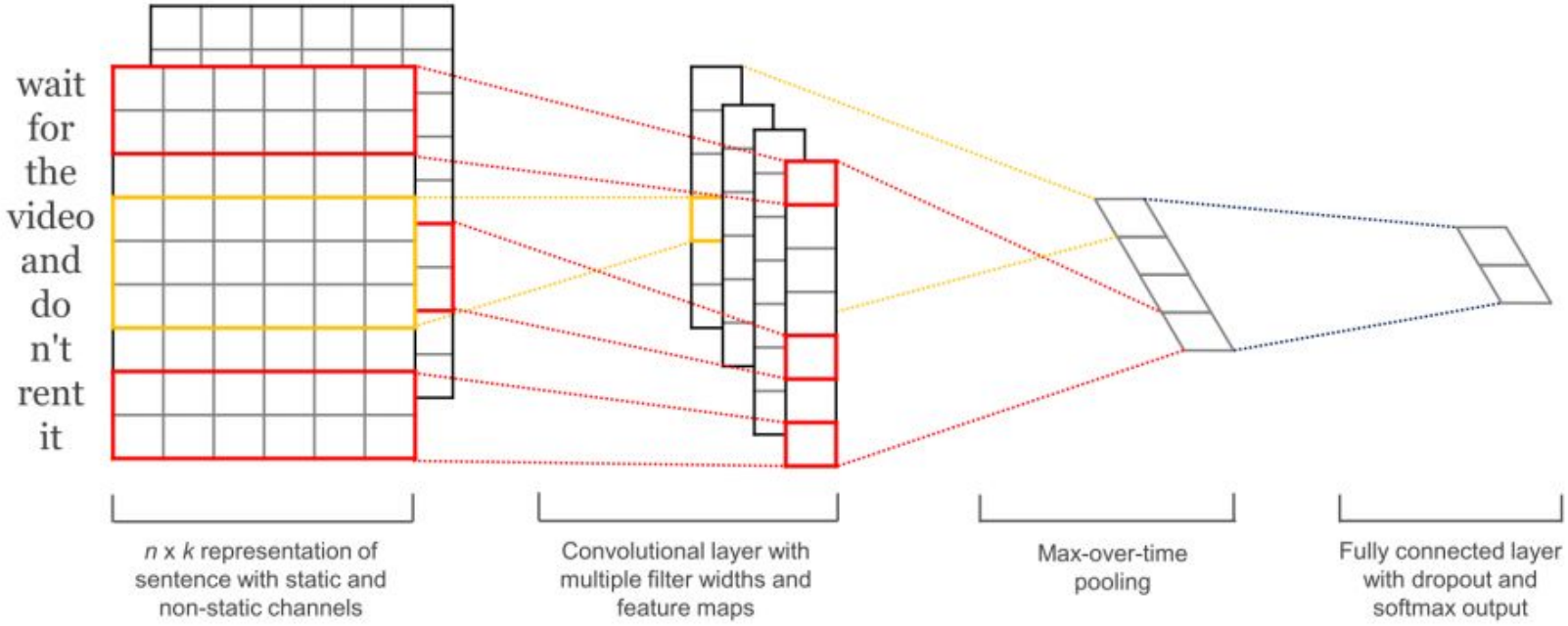
- **gamma1** (*float, optional*) – Decay factor of moving average for `gradient^2`.
- **gamma2** (*float, optional*) – A "momentum" factor. Only used if `centered=True`.
- **epsilon** (*float, optional*) – Small value to avoid division by 0.
- **centered** (*bool, optional*) – Use Graves' or Tieleman & Hinton's version of RMSProp.
- **clip_weights** (*float, optional*) – clip weights into range `[-clip_weights, clip_weights]`

STOCHASTIC GRADIENT DESCENT



- Comparison of several RMSProp SGD parametrizations

ARCHITECTURE PARAMETERS

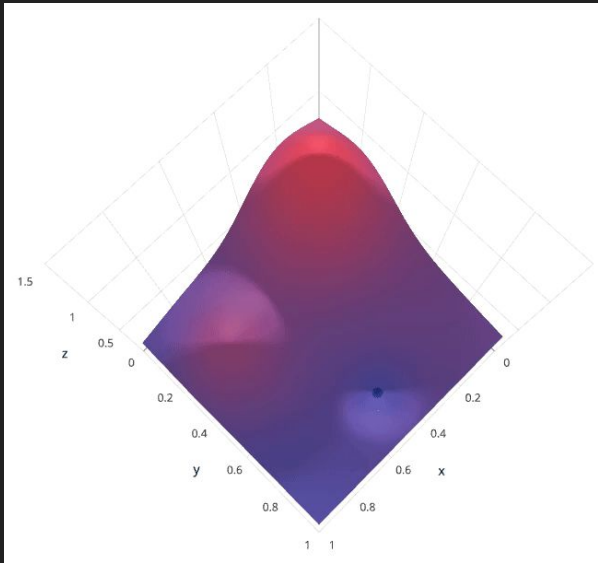


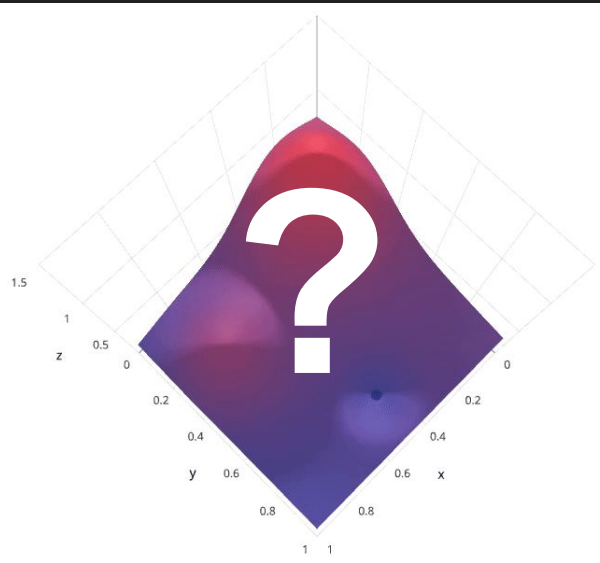
TUNING METHODS

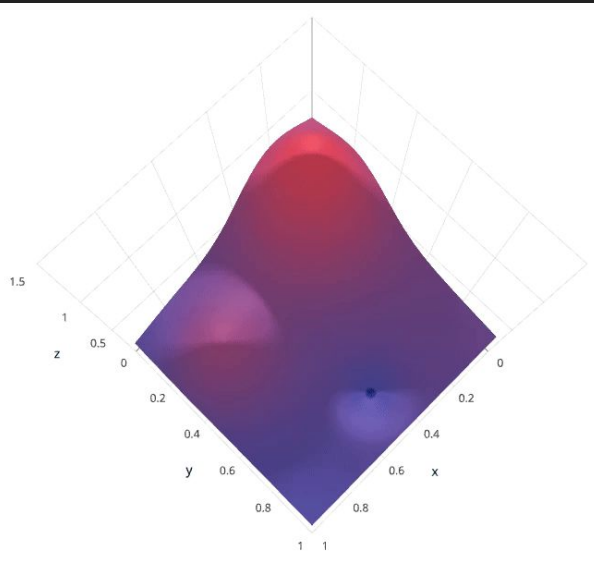
Grid Search

Random Search

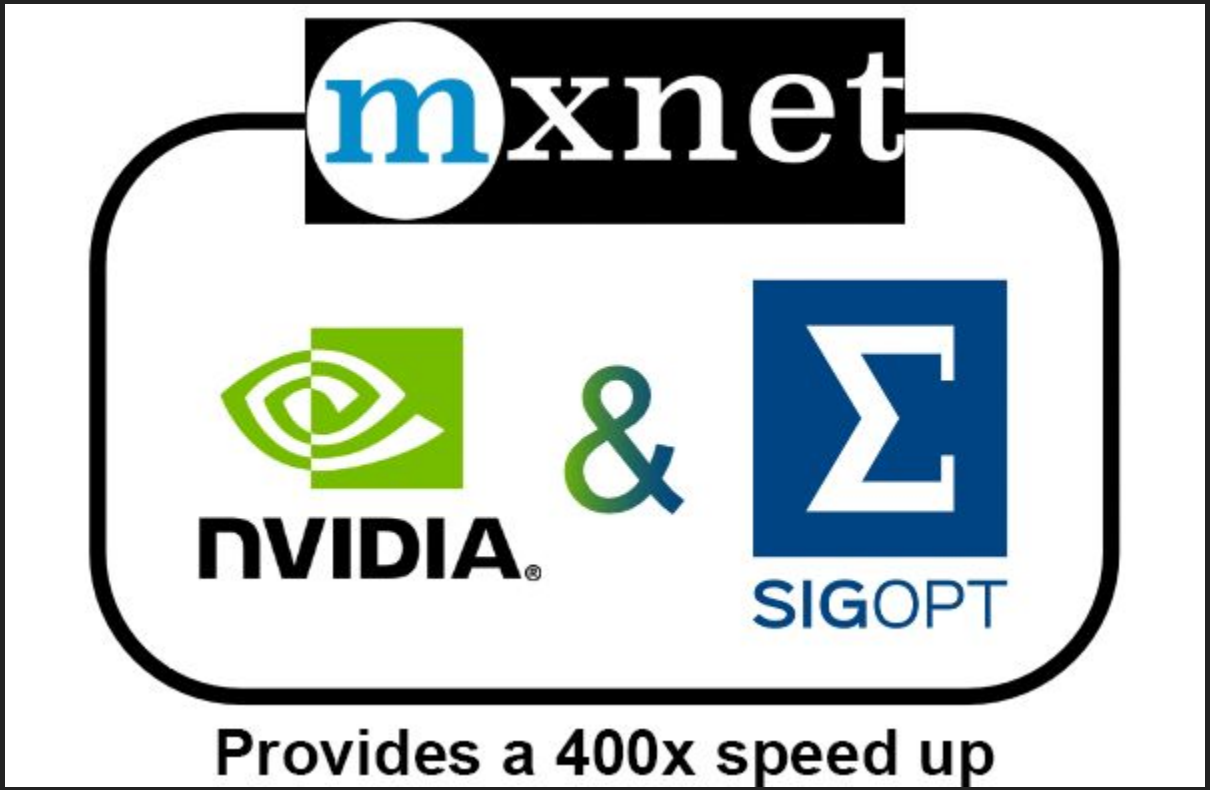
SIGOPT







MULTIPLICATIVE TUNING SPEED UP



SPEED UP #1: CPU -> GPU

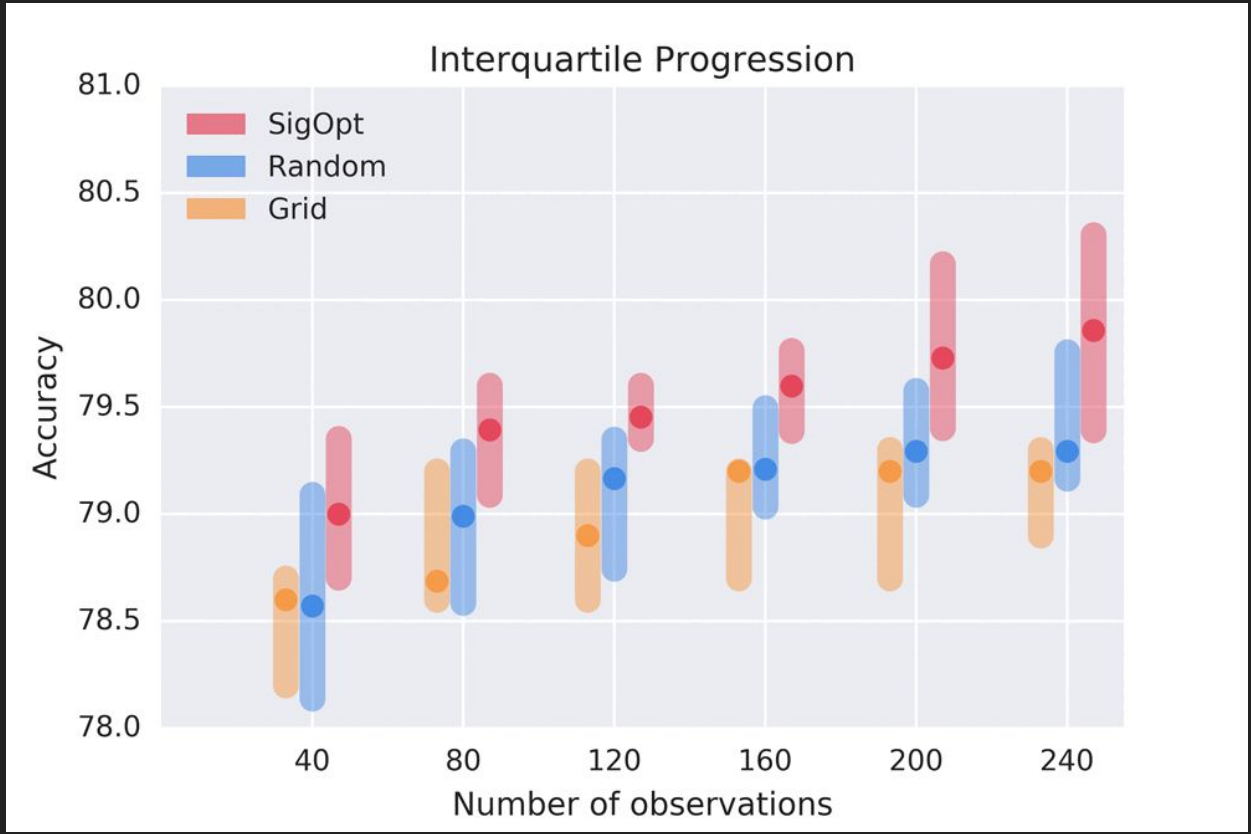
NVIDIA GPU	vCPU
3 seconds per epoch	146 seconds per epoch
optimizer rmsprop maximum gradient 5.0 learning rate (step size) 0.0005 epochs to train for 50 Iter [0] Train: Time: 3.212s Iter [1] Train: Time: 3.047s Iter [2] Train: Time: 3.041s Iter [3] Train: Time: 3.070s Iter [4] Train: Time: 3.051s Iter [5] Train: Time: 3.047s	optimizer rmsprop maximum gradient 5.0 learning rate (step size) 0.0005 epochs to train for 50 Iter [0] Train: Time: 146.908s Iter [1] Train: Time: 146.520s Iter [2] Train: Time: 146.748s Iter [3] Train: Time: 146.313s Iter [4] Train: Time: 145.952s Iter [5] Train: Time: 145.689s

SPEED UP #2: RANDOM/GRID -> SIGOPT

	SigOpt	Random Search	Grid Search
Basic	240 trials	2400 trials	729 trials
Complex	400 trials	4000 trials	59049 trials

Relative Performance	**SigOpt**	**Random Search**	**Grid Search**
Basic Scenario	+6.2%	+5.6%	+4.8%
Complex Scenario	+7.0%	+5.8%	Not feasible

CONSISTENTLY BETTER AND FASTER

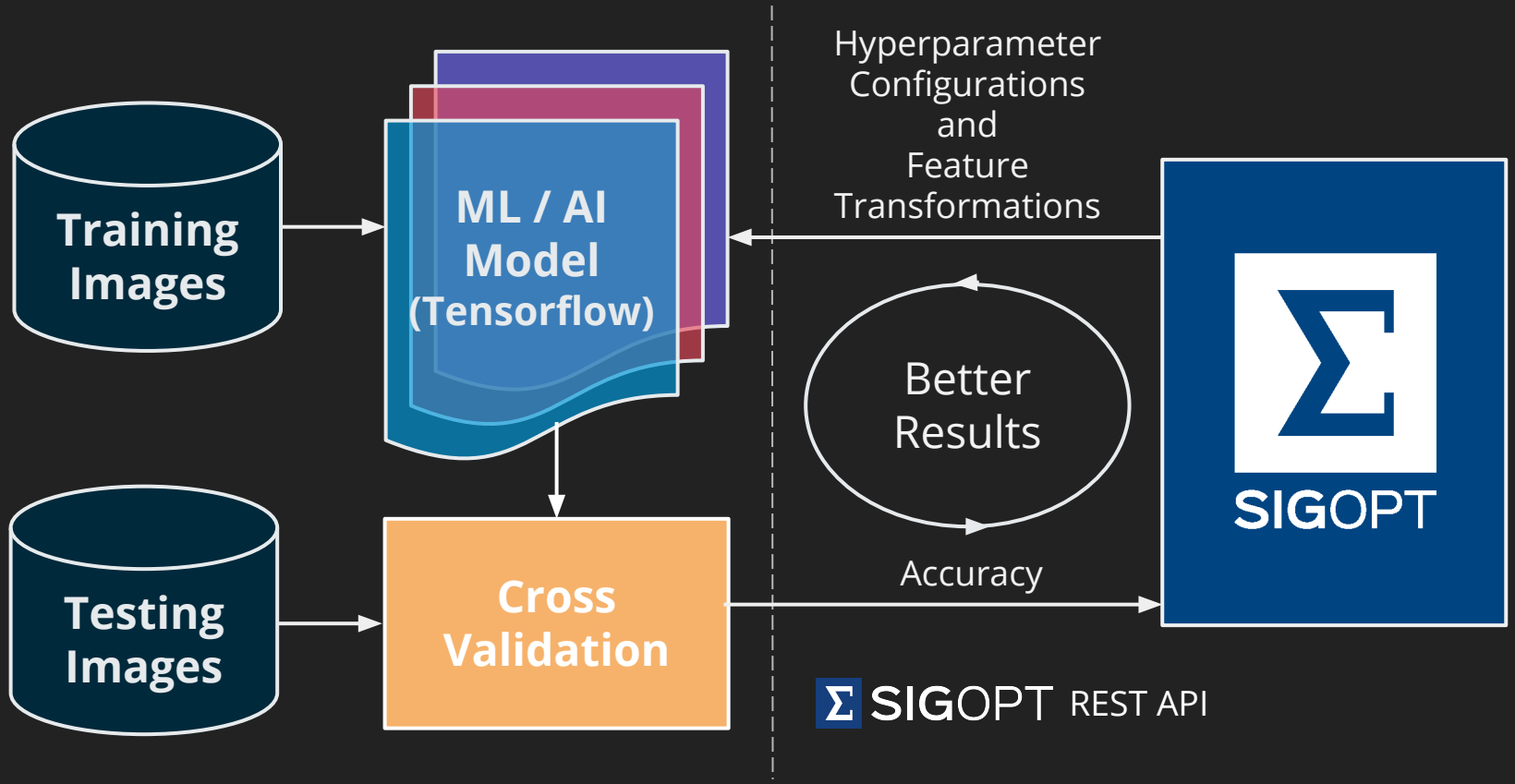


SIGOPT + TENSORFLOW

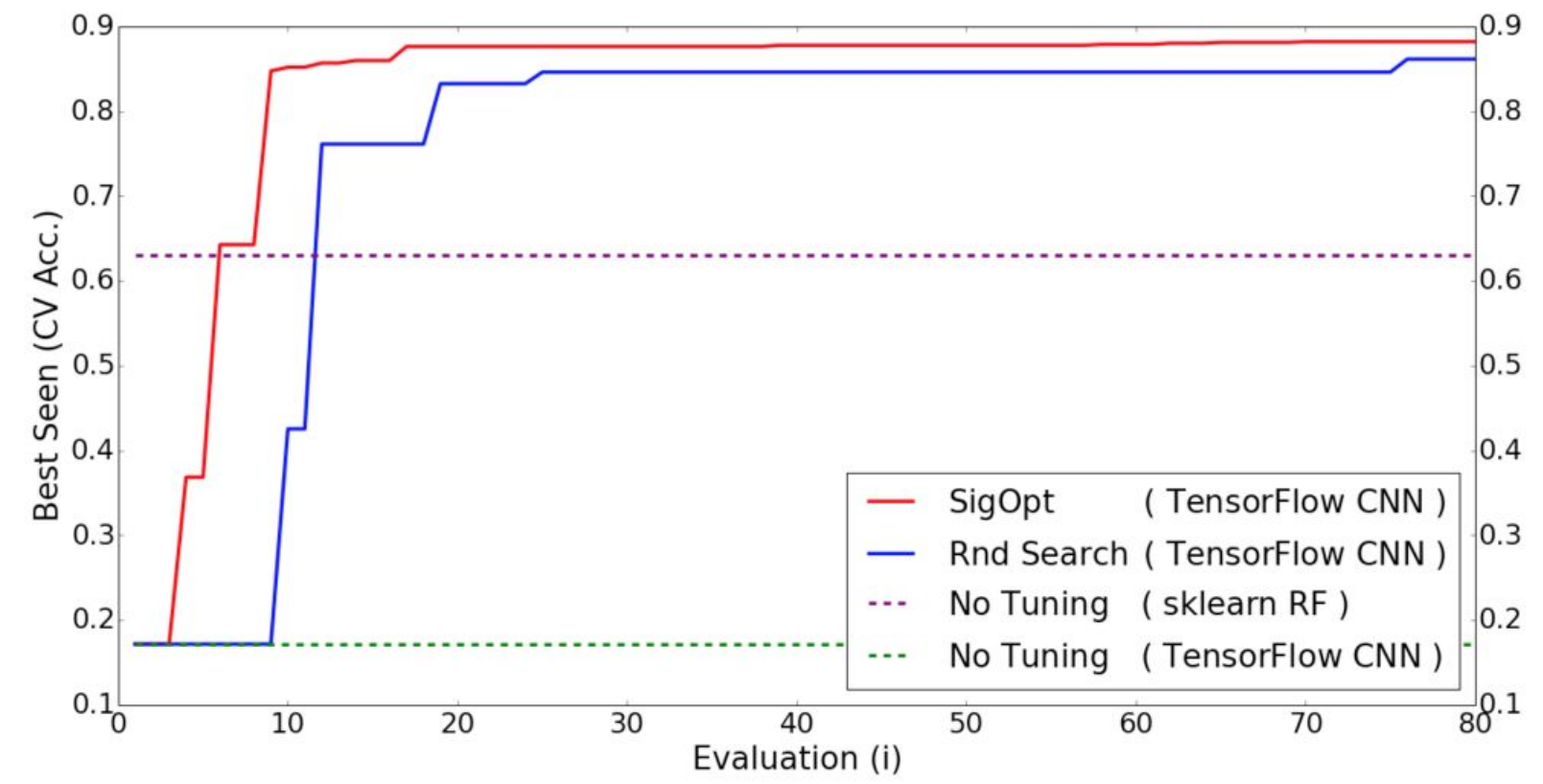
- Classify house numbers in an image dataset (SVHN)

Figure 1. 32x32 cropped samples from the classification task of the SVHN dataset. Each sample is assigned only a single digit label (0 to 9) corresponding to the center digit.

COMPUTER VISION PIPELINE



METRIC OPTIMIZATION

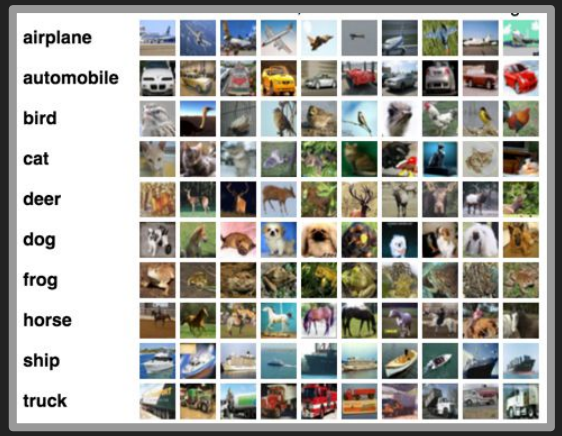


SIGOPT + NEON

STRIVING FOR SIMPLICITY:
THE ALL CONVOLUTIONAL NET

Jost Tobias Springenberg*, Alexey Dosovitskiy*, Thomas Brox, Martin Riedmiller
Department of Computer Science
University of Freiburg
Freiburg, 79110, Germany
{springj, dosovits, brox, riedmiller}@cs.uni-freiburg.de

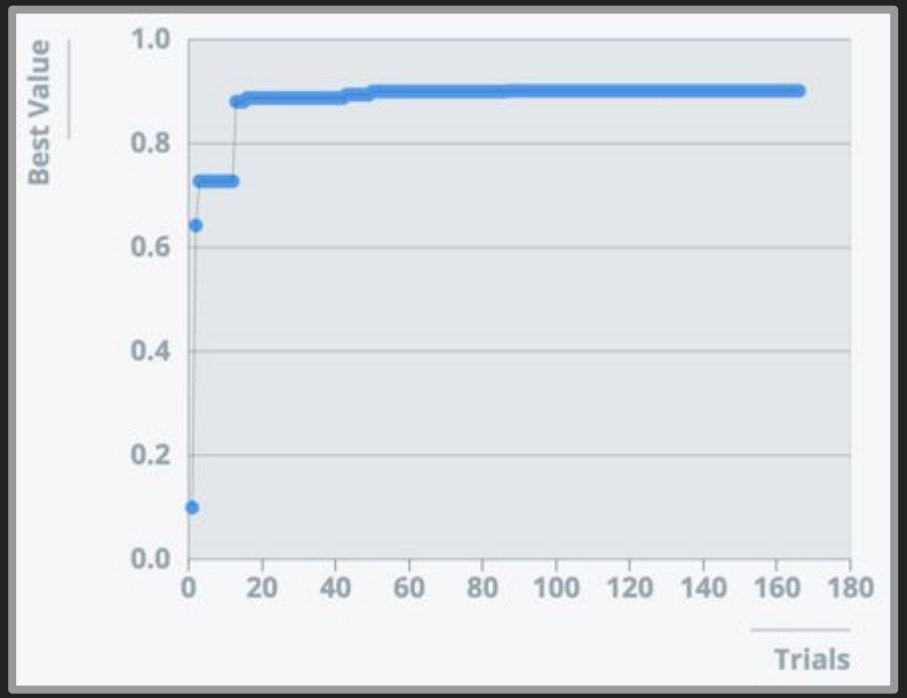
neon



- All convolutional neural network
- Multiple convolutional and dropout layers
- Hyperparameter optimization mixture of domain expertise and grid search (brute force)

http://arxiv.org/pdf/1412.6806.pdf

COMPARATIVE PERFORMANCE



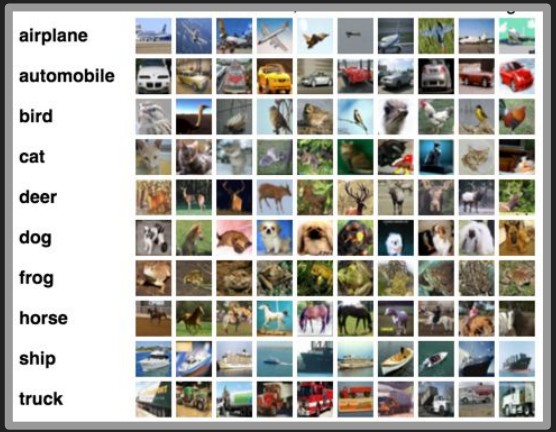
- Expert baseline: 0.8995
  - (using neon)
- SigOpt best: 0.9011
  - 1.6% reduction in error rate
  - No expert time wasted in tuning

SIGOPT + NEON

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

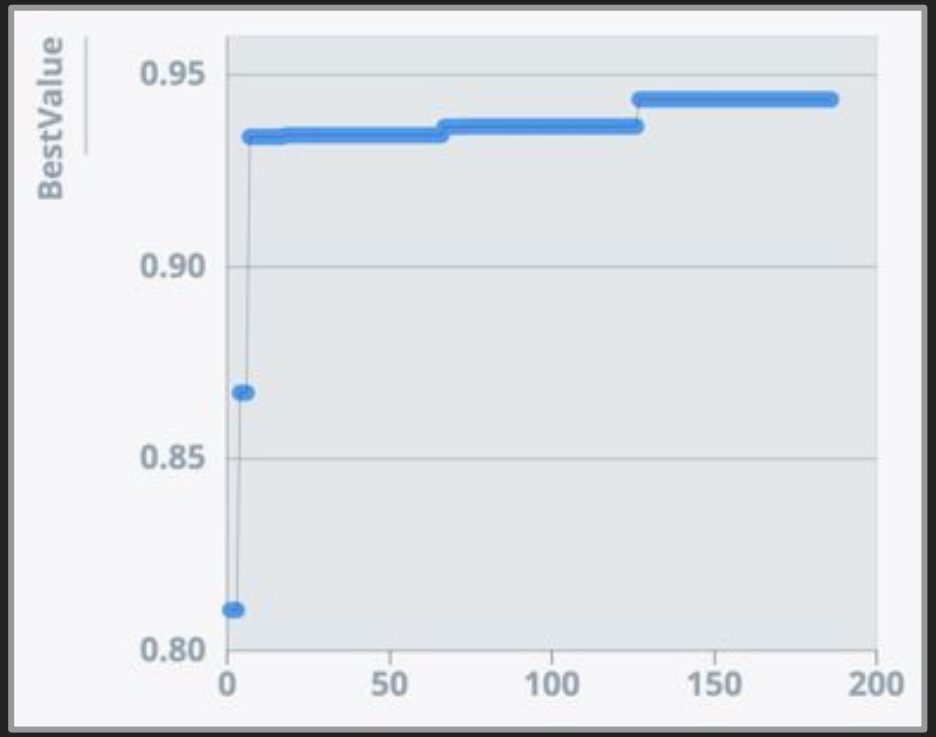
neon



- Explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions
- Variable depth
- Hyperparameter optimization mixture of domain expertise and grid search (brute force)

http://arxiv.org/pdf/1512.03385v1.pdf

COMPARATIVE PERFORMANCE



- Expert baseline: 0.9339
  - (from paper)
- SigOpt best: 0.9436
  - 15% relative error rate reduction
  - No expert time wasted in tuning

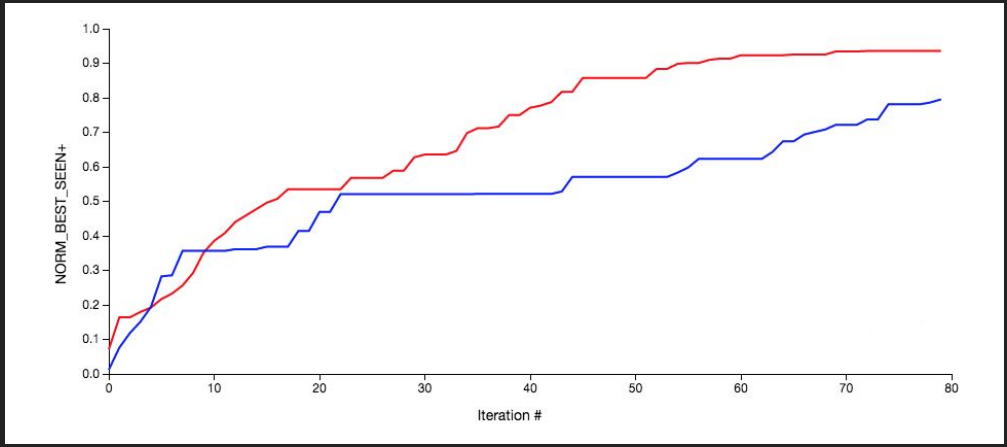
EVALUATING THE OPTIMIZER

OUTLINE

- Metric Definitions
- Benchmark Suite
- Eval Infrastructure
- Visualization Tool
- Baseline Comparisons

METRIC: BEST FOUND

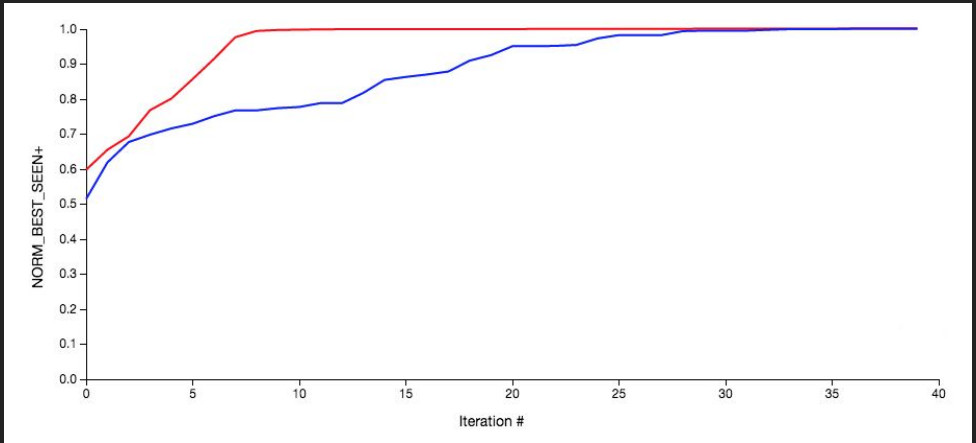
What is the best value found after optimization completes?



	BLUE	RED
BEST_FOUND	0.7225	0.8949

METRIC: AUC

How quickly is optimum found? (area under curve)

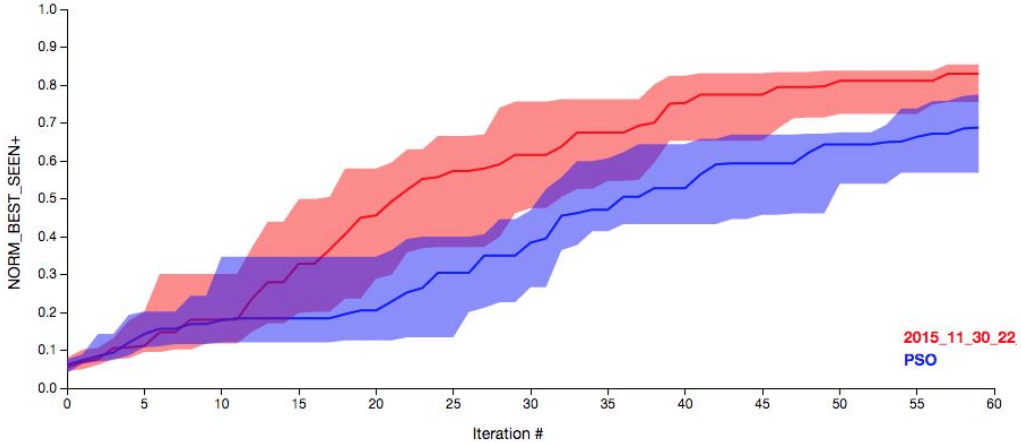


	BLUE	RED
BEST_FOUND	0.9439	0.9435
AUC	0.8299	0.9358

STOCHASTIC OPTIMIZATION

MUL_NOISE_{0.00001}_Ackley(3) with 20 runs

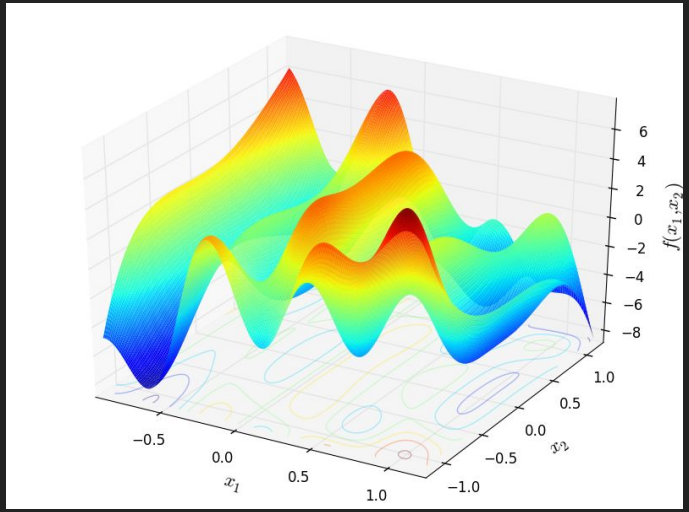
Metric	PSO	2015_11_30_22_43_00.json	p Value
NORM_BEST_FOUND+	0.6284319023543677	0.8008288853693207	0.0004604566971440039
NORM_AUC+	0.37554993997539576	0.5126848207184765	0.001983119320111959



Toggle function image Graph Options

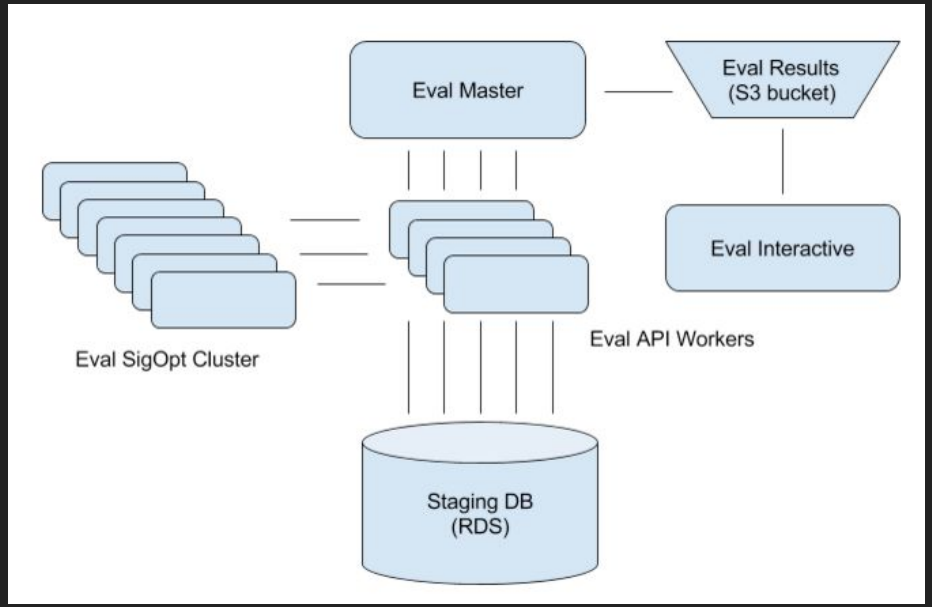
BENCHMARK SUITE

- Optimization functions from literature
- ML datasets: LIBSVM, Deep Learning, etc



TEST FUNCTION TYPE	COUNT
Continuous Params	184
Noisy Observations	188
Parallel Observations	45
Integer Params	34
Categorical Params / ML	47
Failure Observations	30
TOTAL	**489**

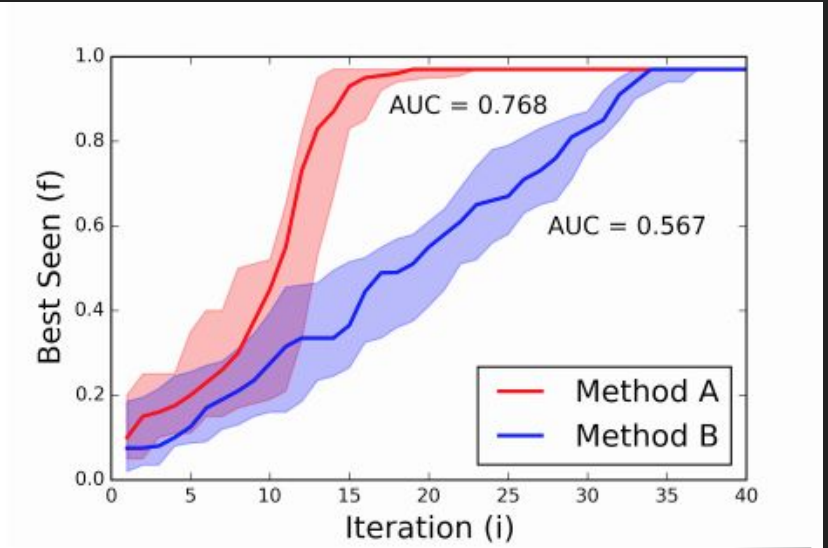
INFRASTRUCTURE

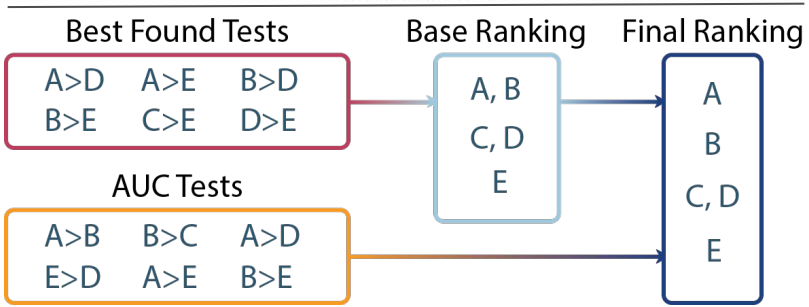


- On-demand **cluster in AWS** for parallel eval function optimization
- Full eval consists of ~20000 optimizations, taking ~30 min

RANKING OPTIMIZERS

1. Mann-Whitney U tests using BEST_FOUND
2. Tied results then partially ranked using AUC
3. Any remaining ties, stay as ties for final ranking





RANKING AGGREGATION

- Aggregate partial rankings across all eval functions using Borda count (sum of methods ranked lower)

$$
\begin{aligned}
f_1 &: \quad A > B > C > D, \\
f_2 &: \quad (A, B) > C, D, \\
f_3 &: \quad C > A > B, D, \\
f_4 &: \quad D > (A, C) > B, \\
f_5 &: \quad (A, B, C, D) > 0, \\
f_6 &: \quad B > (A, C, D),
\end{aligned}
$$

Table 1. Sample aggregation of results

ALGORITHM	BORDA	FIRSTS	TOP THREE
METHOD A	8	3	6
METHOD B	7	3	6
METHOD C	5	2	6
METHOD D	3	2	5

SHORT RESULTS SUMMARY

Table 8. **Noisy functions** (24 total)

Algorithm	Borda	Firsts	Top Three
SigOpt	89	21	24
Spearmint	60	9	16
HyperOpt	47	12	19
SMAC	13	4	6
PSO	64	12	24
Grid	22	8	11
Random	16	7	9

Table 6. **Mostly boring functions** (10 total)

Algorithm	Borda	Firsts	Top Three
Spearmint	46	6	8
SigOpt	45	5	10
HyperOpt	33	2	8
SMAC	11	2	2
PSO	21	1	7
Grid	8	2	2
Random	5	1	1

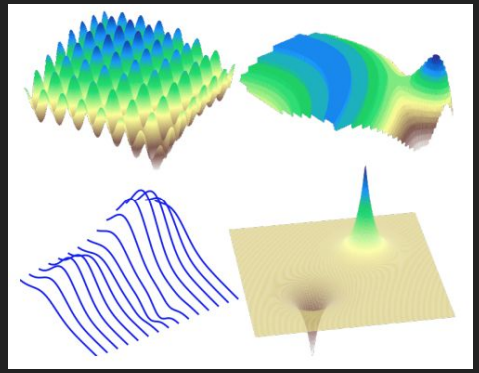
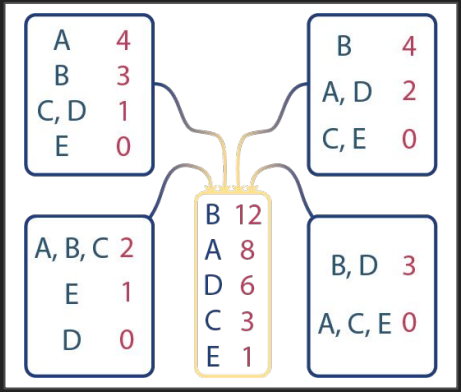


Table 7. **Boundary optimum functions** (13 total)

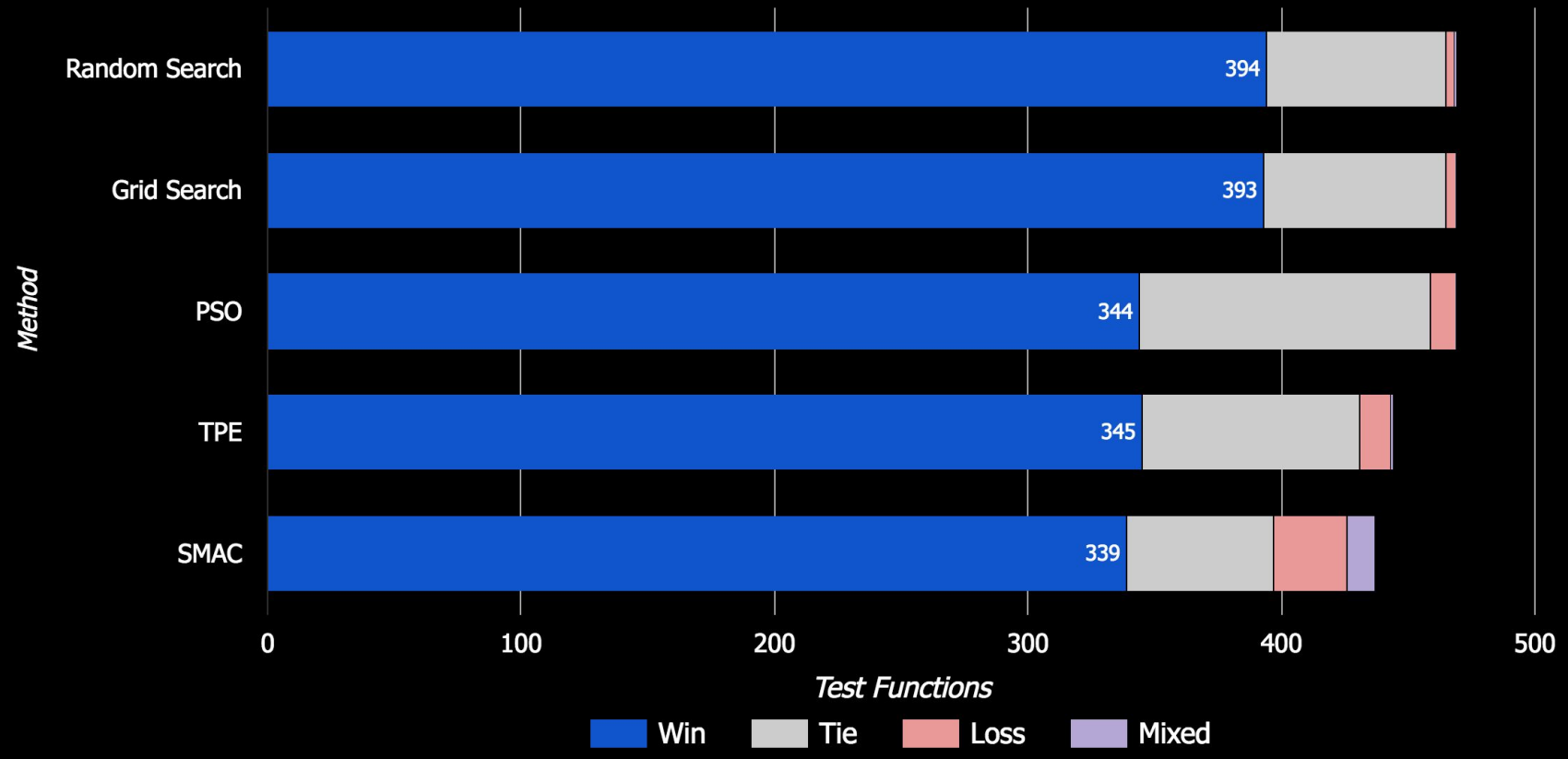
Algorithm	Borda	Firsts	Top Three
Spearmint	74	11	13
SigOpt	67	4	13
HyperOpt	29	–	9
SMAC	7	–	2
PSO	30	–	11
Grid	14	–	5
Random	2	–	1

Table 5. **≥10D functions** (8 total)

Algorithm	Borda	Firsts	Top Three
SigOpt	40	7	8
HyperOpt	35	4	8
Spearmint	23	1	6
SMAC	12	–	3
Random	16	1	3
PSO	12	–	2
Grid	–	–	–

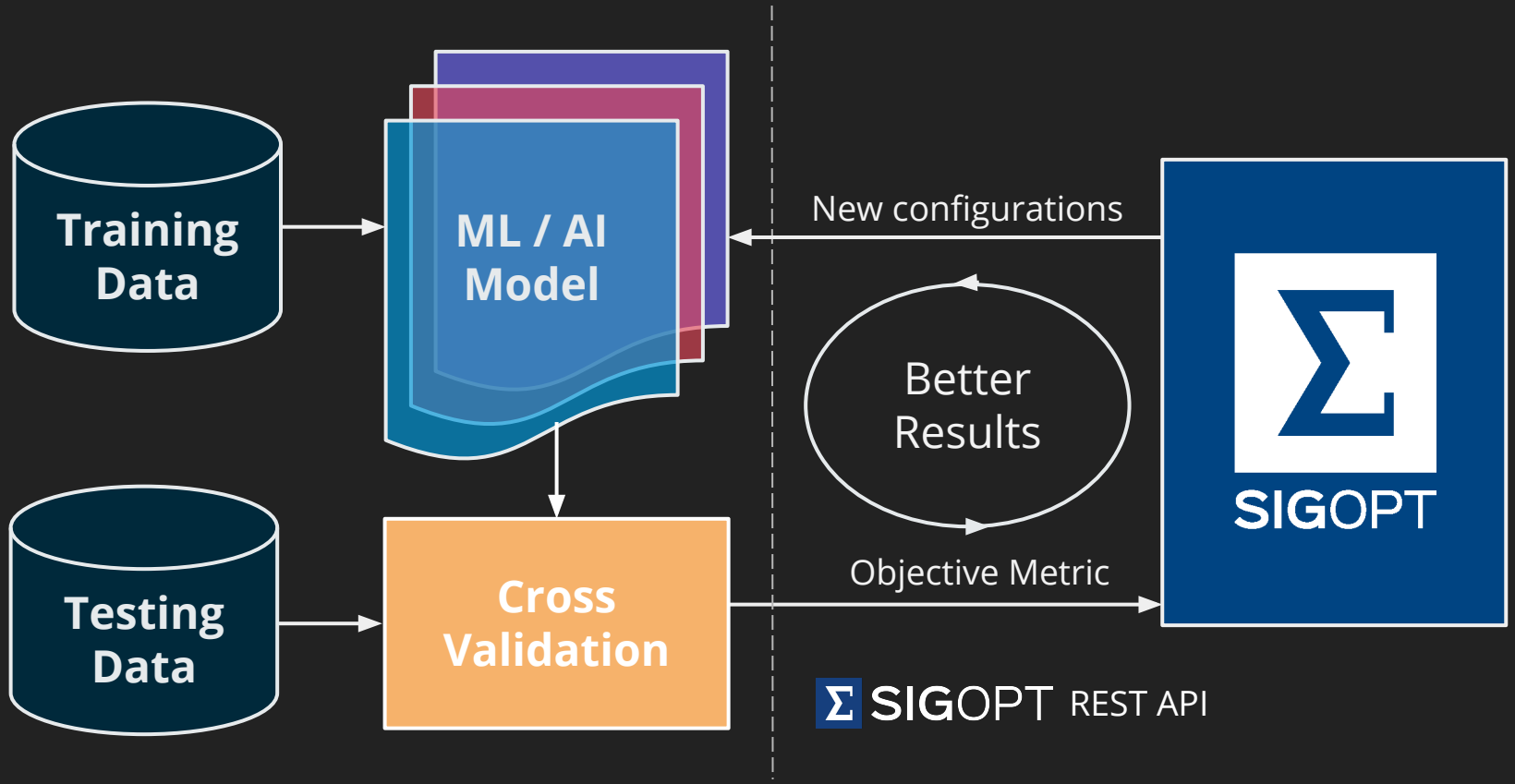


BASELINE COMPARISONS



SIGOPT SERVICE

OPTIMIZATION FEEDBACK LOOP



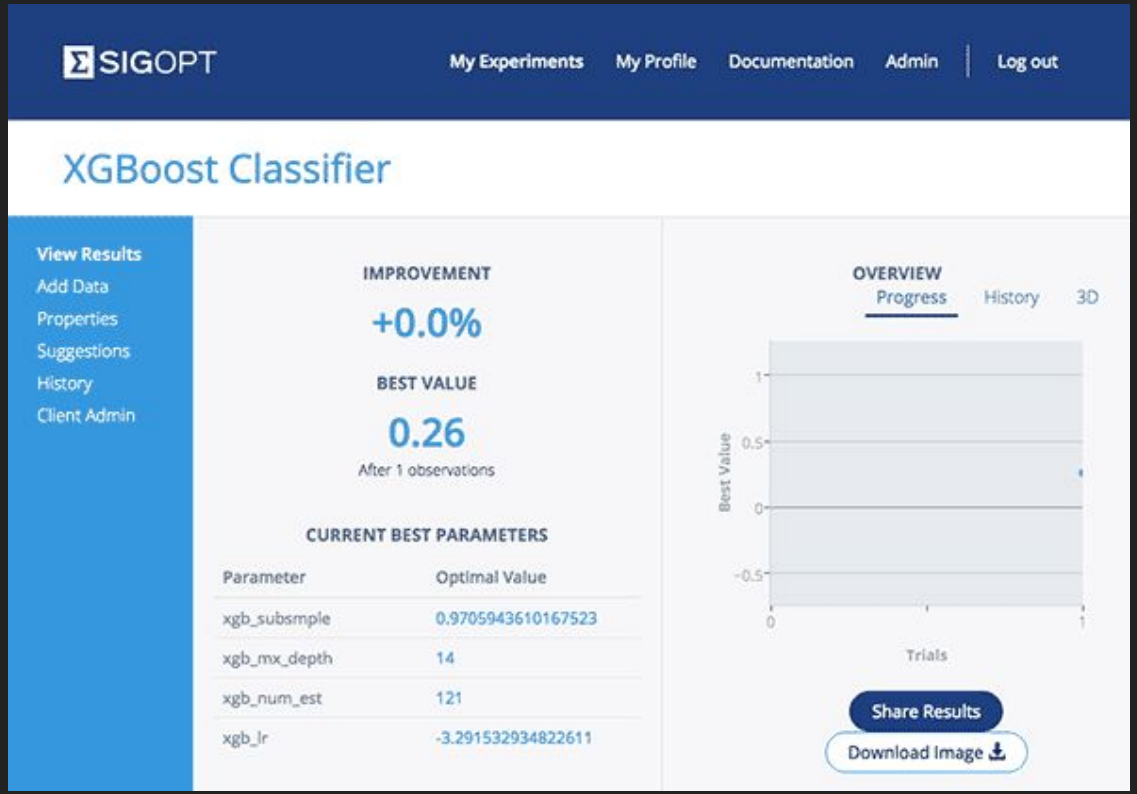
SIMPLIFIED OPTIMIZATION

Client Libraries

- Python
- Java
- R
- Matlab
- And more...

Framework Integrations

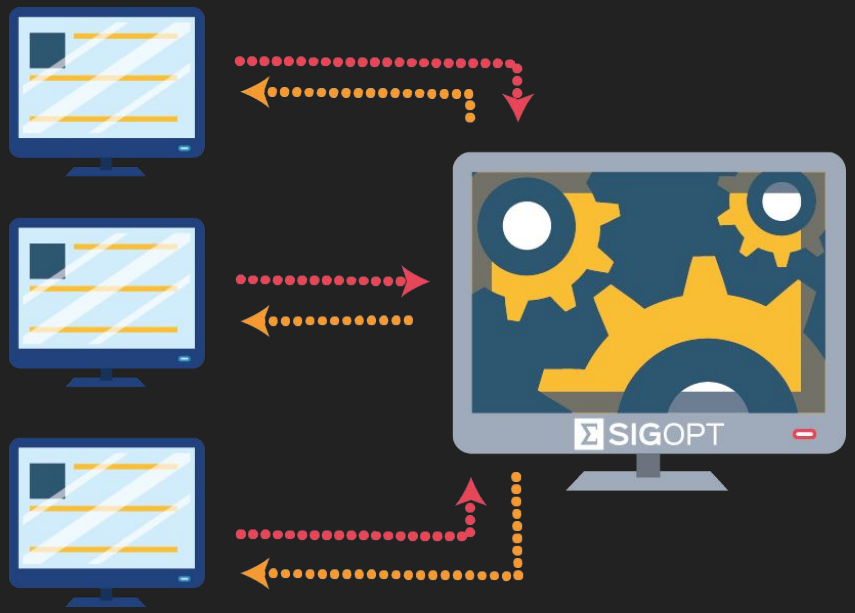
- TensorFlow
- scikit-learn
- xgboost
- Keras
- Neon
- And more...



Live Demo

DISTRIBUTED TRAINING

- SigOpt serves as a distributed scheduler for training models across workers
- Workers access the SigOpt API for the latest parameters to try for each model
- Enables easy distributed training of non-distributed algorithms across any number of models



COMPARATIVE PERFORMANCE

- **Better Results, Faster and Cheaper**
  Quickly get the most out of your models with our proven, peer-reviewed ensemble of Bayesian and Global Optimization Methods
  - A Stratified Analysis of Bayesian Optimization Methods (ICML 2016)
  - Evaluation System for a Bayesian Optimization Service (ICML 2016)
  - Interactive Preference Learning of Utility Functions for Multi-Objective Optimization (NIPS 2016)
  - And more...
- **Fully Featured**
  Tune any model in any pipeline
  - Scales to 100 continuous, integer, and categorical parameters and many thousands of evaluations
  - Parallel tuning support across any number of models
  - Simple integrations with many languages and libraries
  - Powerful dashboards for introspecting your models and optimization
  - Advanced features like multi-objective optimization, failure region support, and more
- **Secure Black Box Optimization**
  Your data and models never leave your system

Try it yourself!

https://sigopt.com/getstarted

Questions?

contact@sigopt.com

SIGOPT

https://sigopt.com
@SigOpt