Intelligent Experimentation In Artificial Intelligence

Tobias Andreasen – Machine Learning Engineer

intel | SIGOPT

SigOpt empowers teams to build the best models

OpenAI

TWO SIGMA

UNI FR — UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG

Design Experiments

"Integrating SigOpt into our modeling platform empowers our team to more efficiently experiment, optimize and, ultimately, model at scale."

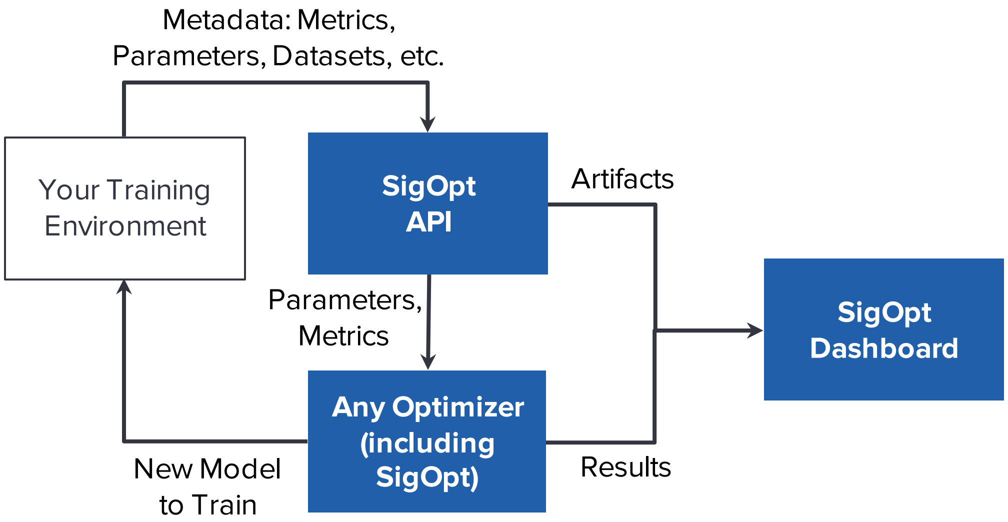
Explore Modeling Problems

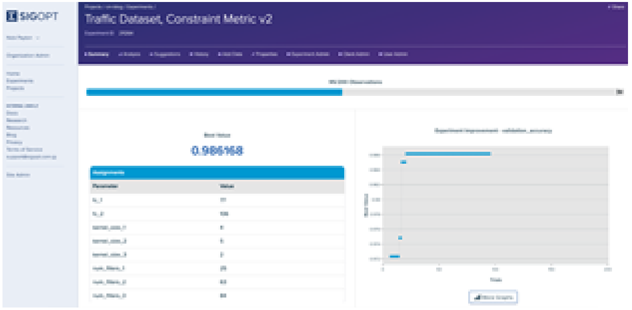
"We've integrated SigOpt's optimization service and are now able to get better results faster and cheaper than any solution we've seen before."

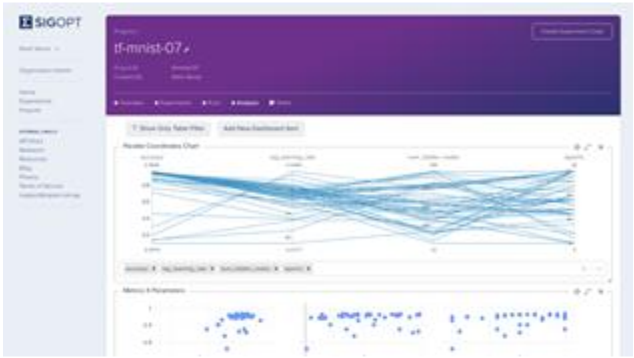
Optimize Models

"SigOpt is the most advanced and complete solution...we have encountered, and it enables us to produce robust and reproducible research with reliable results."

How SigOpt empowers AI developers







SigOpt works with any stack in any domain

OpenAI

TWO SIGMA

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Numenta

PayPal

Massachusetts Institute of Technology

Mila

iqt IN-Q-TEL

TECHNISCHE UNIVERSITÄT DARMSTADT

Stanford University

anastasia

University of Pittsburgh

intel

What is an intelligent approach to experimentation?

Define metrics

Compare runs

Warm start development

Track work

Understand model behavior

Analyze parameter space

Generate plots

Experimentation

Debug code

Optimize hyperparameters

Stitch together tooling

Utilize compute

Select architecture

Report on metrics

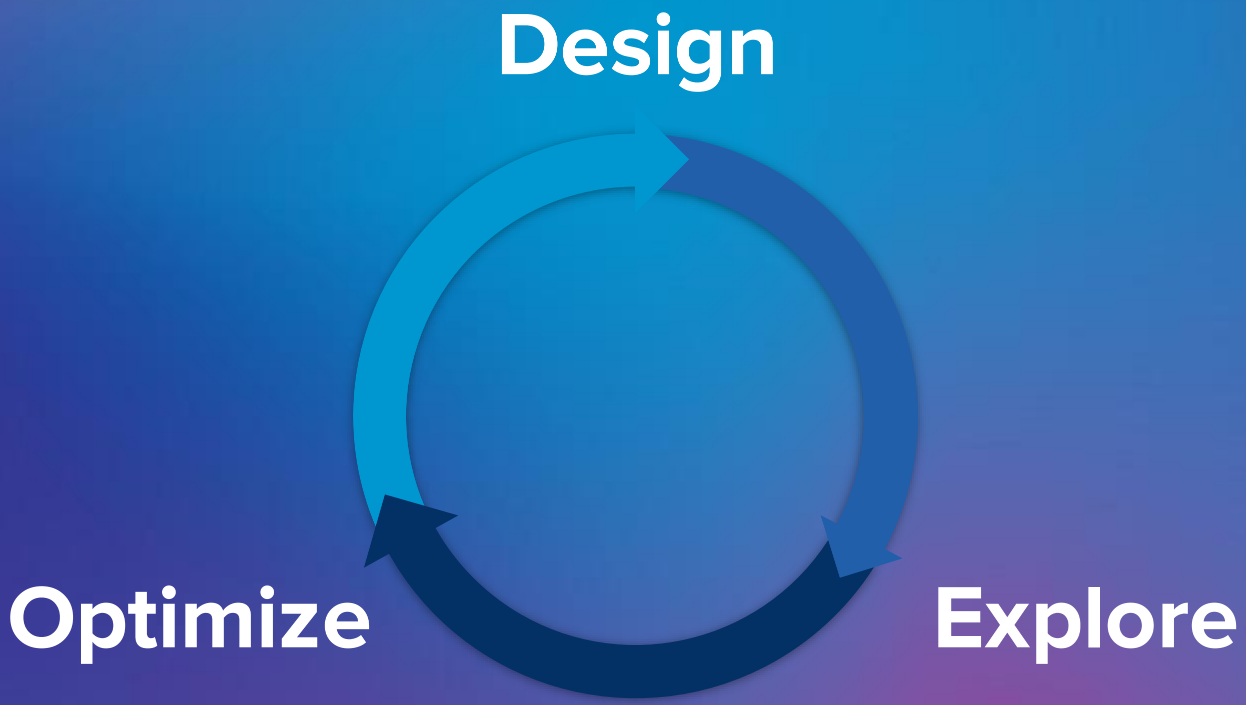
Collaborate on projects

Experimentation

“…provides insight into cause-and-effect by demonstrating what outcome occurs when a particular component is manipulated…”

Intelligent Experimentation

“…makes **recommendations** based on **design decisions** on how to **explore** modeling problems in order to find the **optimal solution**(s)…”


Design
Explore
Optimize

Design Explore Optimize

Design is a series of decisions

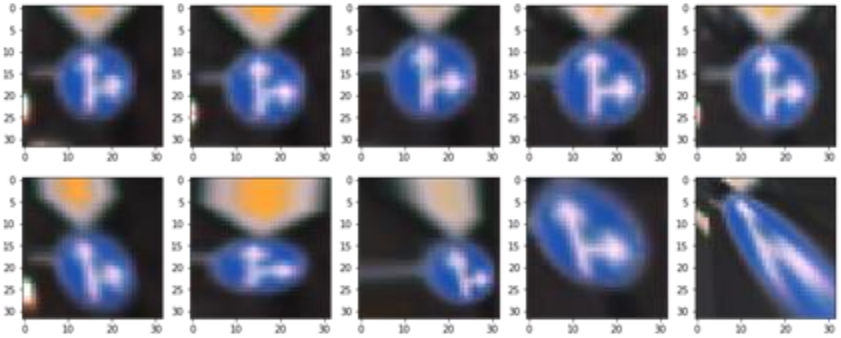
Choose the data

Choose the model

Choose the loss function

Data decisions to be made:

- Data sources/versions
- Cleaning
- Feature engineering
- Augmentation
- more…



Design Explore Optimize

Design is a series of decisions

- Choose the data
- **Choose the model**
- Choose the loss function

H2O	Keras	LightGBM
MXNet	PyTorch	Sagemaker
scikit-learn	TensorFlow	XGBoost

Design Explore Optimize

Design is a series of decisions

- Choose the data
- Choose the model
- **Choose the loss function**

- regression application
  - `regression` , L2 loss, aliases: `regression_l2` , `l2` , `mean_squared_error` , `mse` , `l2_root` , `root_mean_squared_error` , `rmse`
  - `regression_l1` , L1 loss, aliases: `l1` , `mean_absolute_error` , `mae`
  - `huber` , Huber loss
  - `fair` , Fair loss
  - `poisson` , Poisson regression
  - `quantile` , Quantile regression
  - `mape` , MAPE loss, aliases: `mean_absolute_percentage_error`
  - `gamma` , Gamma regression with log-link. It might be useful, e.g., for modeling insurance claims severity, or for any target that might be gamma-distributed
  - `tweedie` , Tweedie regression with log-link. It might be useful, e.g., for modeling total loss in insurance, or for any target that might be tweedie-distributed
- binary classification application
  - `binary` , binary log loss classification (or logistic regression)
  - requires labels in {0, 1}; see `cross-entropy` application for general probability labels in [0, 1]
- multi-class classification application
  - `multiclass` , softmax objective function, aliases: `softmax`
  - `multiclassova` , One-vs-All binary objective function, aliases: `multiclass_ova` , `ova` , `ovr`
  - `num_class` should be set as well

Design Explore Optimize

Design is a series of decisions

Choose the data

Choose the model

Choose the loss function

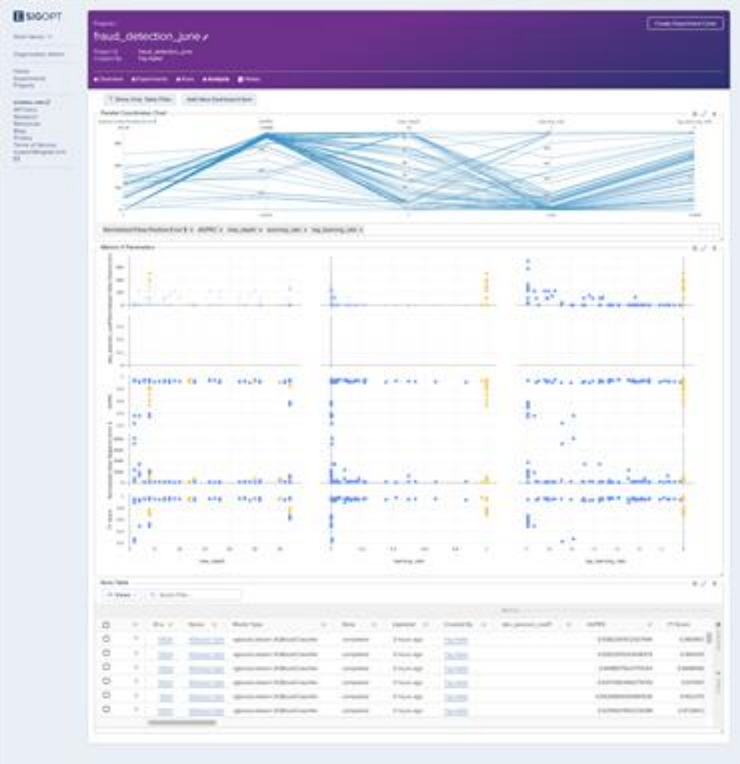
How do you get all of these decisions right?

You don't. Instead, **try** and **track** everything.

Design Explore Optimize

Try and track everything in SigOpt with a few lines of code

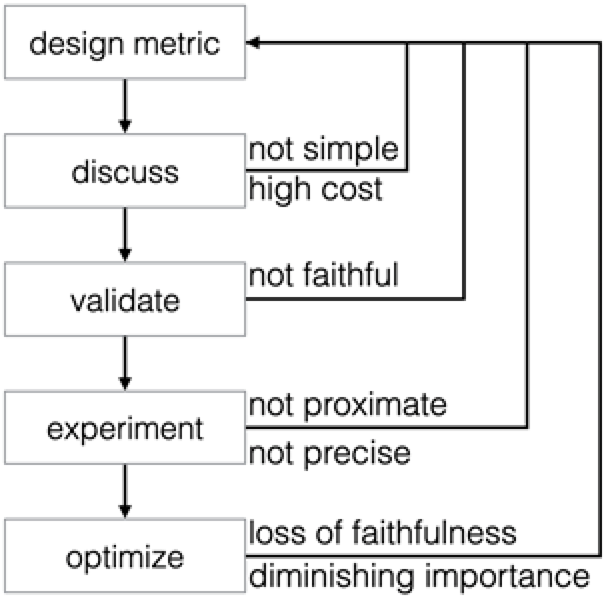
```python
def run_and_track_in_sigopt():

    (features, labels) = get_data()

    sigopt.log_dataset(DATASET_NAME)
    sigopt.log_metadata(key="Dataset Source", value=DATASET_SRC)
    sigopt.log_metadata(key="Feature Eng Pipeline Name", value=FEATURE_ENG_PIPELINE_NAME)
    sigopt.log_metadata(key="Dataset Rows", value=features.shape[0]) # assumes features X are like a numpy array w
    sigopt.log_metadata(key="Dataset Columns", value=features.shape[1])
    sigopt.log_metadata(key="Execution Environment", value="Colab Notebook")
    sigopt.log_model(MODEL_NAME)
    sigopt.params.max_depth = numpy.random.randint(low=3, high=15, dtype=int)
    sigopt.params.learning_rate = numpy.random.random(size=1)[0]
    sigopt.params.min_split_loss = numpy.random.random(size=1)[0]*10

    args = dict(X=features,
                y=labels,
                max_depth=sigopt.params.max_depth,
                learning_rate=sigopt.params.learning_rate,
                min_split_loss=sigopt.params.min_split_loss)

    mean_accuracy, training_and_validation_time = evaluate_xgboost_model(**args)

    sigopt.log_metric(name='accuracy', value=mean_accuracy)
    sigopt.log_metric(name='training and validation time (s)', value=training_and_validation_time)
```



More than just model.fit()

- Training Metrics
- Validation Metrics
- Guardrail Metrics
- Production Metrics

Lifecycle of a metric



Design **Explore** Optimize

More than just model.fit()

- **Training Metrics**
- Validation Metrics
- Guardrail Metrics
- Production Metrics

Training metrics and how to optimize them

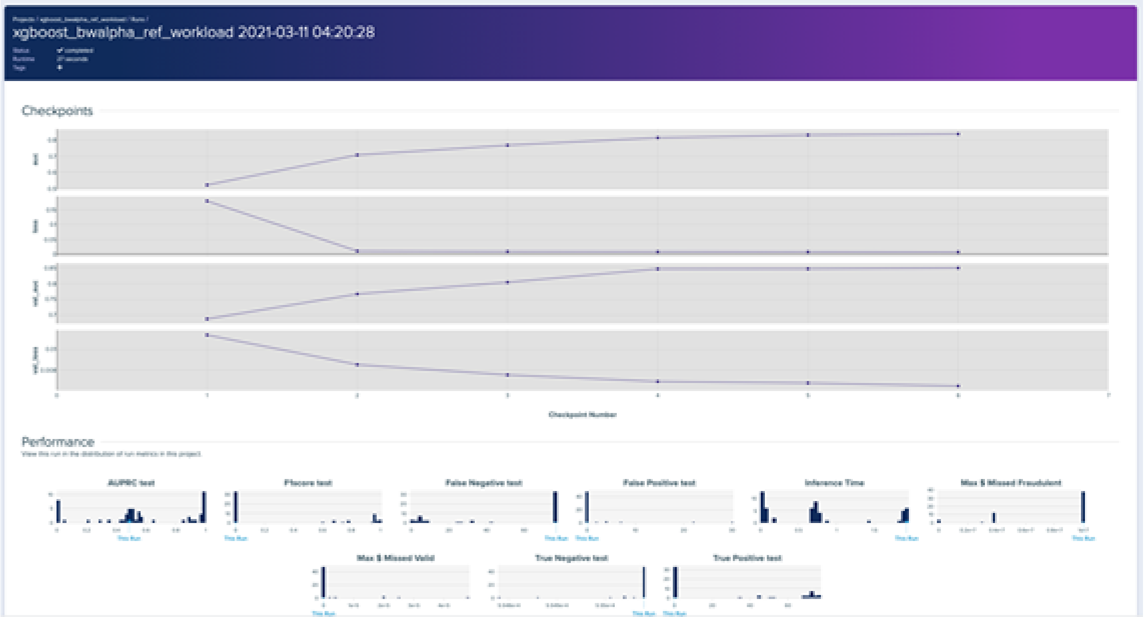
- regression application
  - `regression`, L2 loss, aliases: `regression_l2`, `l2`, `mean_squared_error`, `mse`, `l2_root`, `root_mean_squared_error`, `rmse`
  - `regression_l1`, L1 loss, aliases: `l1`, `mean_absolute_error`, `mae`
  - `huber`, Huber loss
- binary classification application
  - `binary`, binary log loss classification (or logistic regression)
  - requires labels in {0, 1}; see `cross-entropy` application for general probability labels in [0, 1]
- multi-class classification application
  - `multiclass`, softmax objective function, aliases: `softmax`
  - `multiclassova`, One-vs-All binary objective function, aliases: `multiclass_ova`, `ova`, `ovr`
  - `num_class` should be set as well

Image source: Ruder, Sebastian. An overview of gradient descent optimization algorithms.

More than just model.fit()

- **Training Metrics**
- Validation Metrics
- Guardrail Metrics
- Production Metrics

Convergence Matters, Track Training Metrics

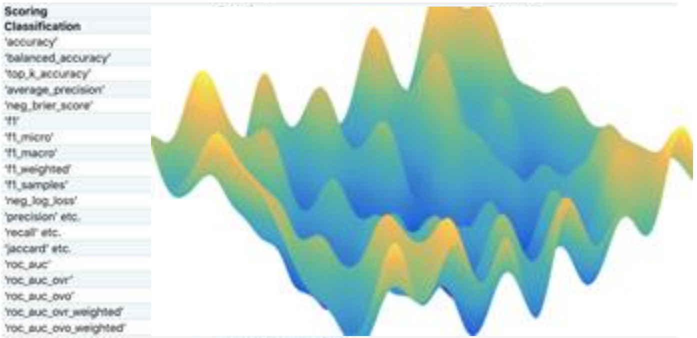


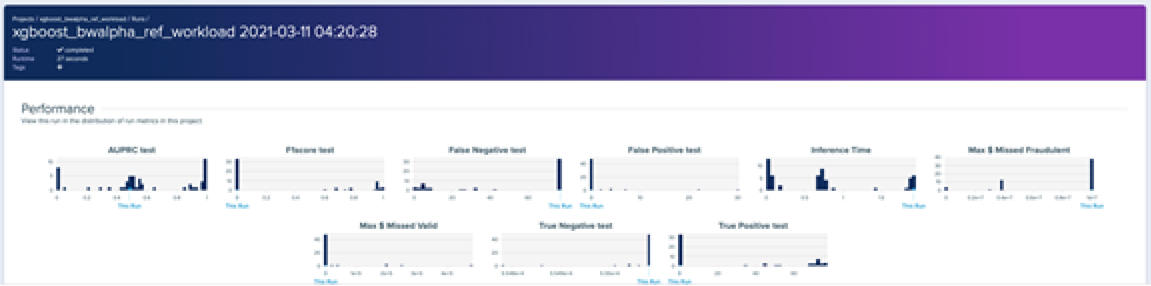
 I Image source: [Ruder, Sebastian. An overview of gradient descent optimization algorithms.]()

More than just model.fit()

- Training Metrics
- **Validation Metrics**
- Guardrail Metrics
- Production Metrics

Track Validation Metric Tradeoffs





More than just model.fit()

- Training Metrics
- Validation Metrics
- **Guardrail Metrics**
- Production Metrics

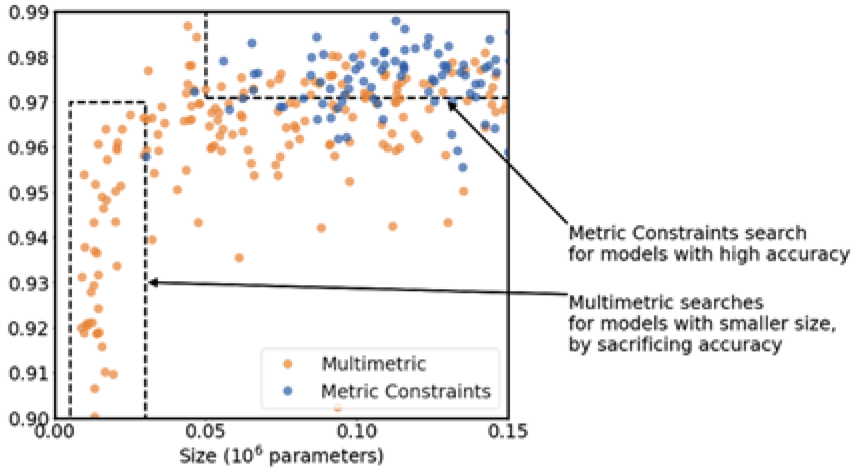
Guardrail metrics represent limitations enforced on the models so as to be viable in production.

- Maximum inference time
- Minimum throughput
- Maximum model size
- Maximum power
- Model interpretability
- Application-specific needs

More than just model.fit()

- Training Metrics
- Validation Metrics
- **Guardrail Metrics**
- Production Metrics

Another view at optimizing multiple metrics



Size ≤ 0.15 * 10^6 parameters

More than just model.fit()

- Training Metrics
- Validation Metrics
- Guardrail Metrics
- **Production Metrics**

True measure of modeling success

The metrics of most interest to us are often unavailable during model development.

- Click-through-rate tomorrow
- Profit over the next month
- Failure probability in a new market

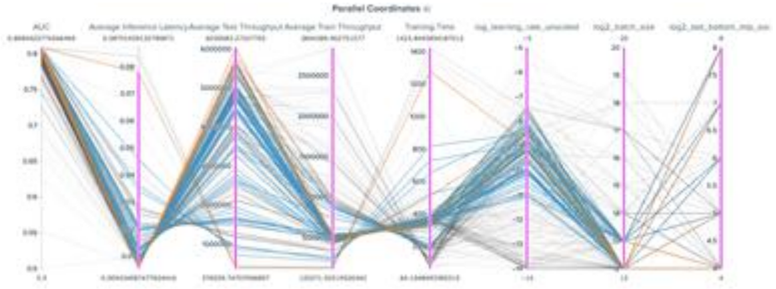
Making final decisions about deployment may involve multiple stakeholders and experts in a project.

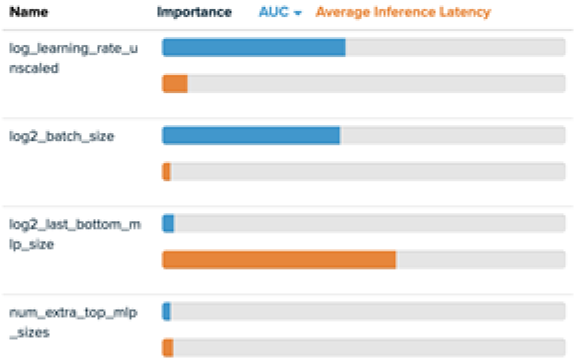
Design Explore Optimize

More than just model.fit()

- Training Metrics
- Validation Metrics
- Guardrail Metrics
- **Production Metrics**

Can our metrics help us prepare for production?





Design Explore **Optimize**

Boost model performance

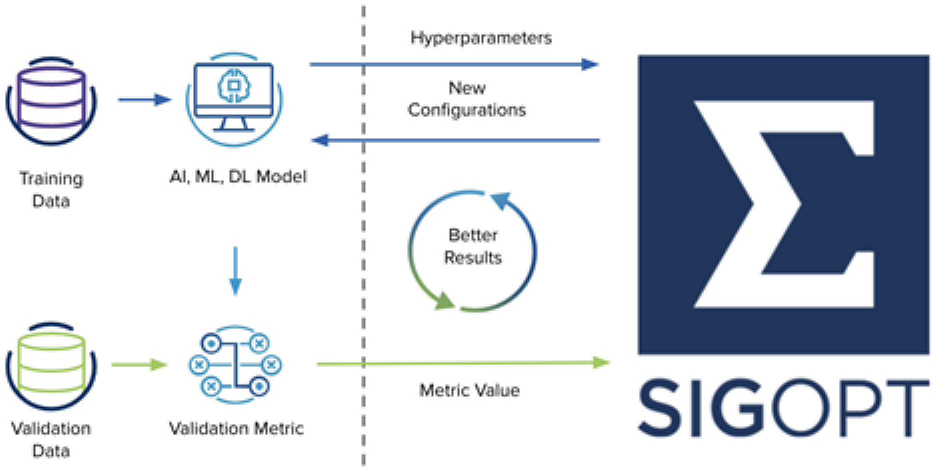
Bayesian Optimization

Optimizing Many Metrics

Bring Your Own Optimizer

And More…

Use SigOpt to optimize validation metrics

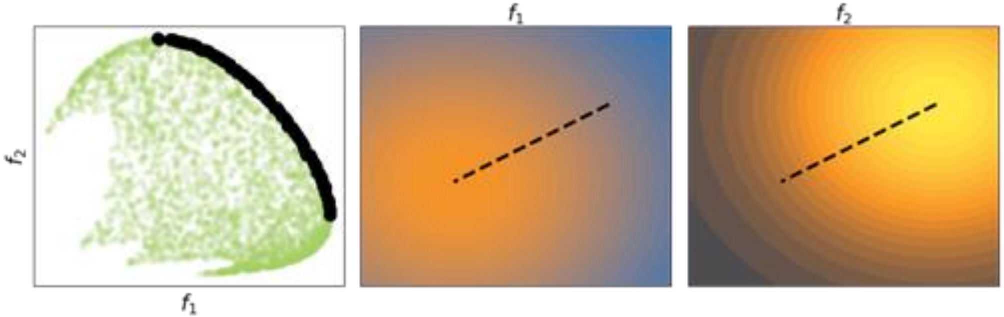


Design Explore **Optimize**

Boost model performance

- Bayesian Optimization
- **Optimizing Many Metrics**
- Bring Your Own Optimizer
- And More...

Use SigOpt to optimize multiple validation metrics



Design Explore **Optimize**

Boost model performance

- Bayesian Optimization
- Optimizing Many Metrics
- **Bring Your Own Optimizer**
- And More…

Use SigOpt to log runs from any optimizer

Random search | Grid search | Hyperopt | Optuna

Optuna was introduced in 2019 by Takuya et al. at Preferred Networks. Using a code like the one below you can use Optuna as an optimizer while leveraging the logging and visualization functionality of SigOpt.

Optuna

```
def optuna_objective_function(trial):
  args = dict(
    hidden_layer_size=trial.suggest_int("hidden_layer_size", 32, 512, 1),
    activation_function=trial.suggest_categorical("activation_function", ["tanh", "r
  )
  optuna_run = Run(number_of_epochs=NUMBER_OF_EPOCHS, run_type="optuna search")
  metric_value = optuna_run.execute(args)
  return metric_value


study = optuna.create_study(direction="maximize")
study.optimize(optuna_objective_function, n_trials=BUDGET, show_progress_bar=False)
```

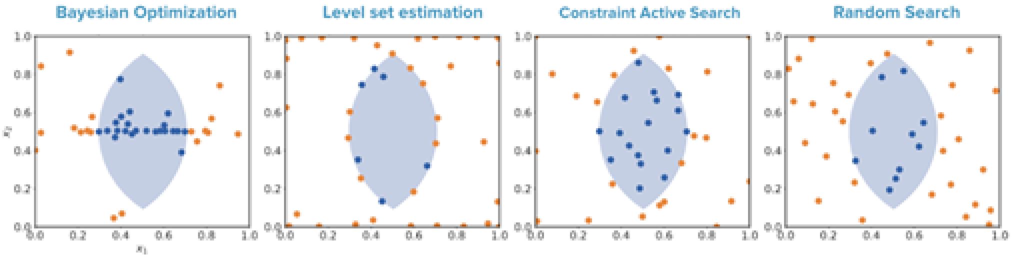
Design Explore **Optimize**

Boost model performance

- Bayesian Optimization
- Optimizing Many Metrics
- Bring Your Own Optimizer
- **And More...**

Examples of advanced features in SigOpt

Metric Strategy	Parameter Constraints	Conditional Parameters	Black-Box Constraints
Failure Regions	Multitask Optimization	Convergence Monitoring	Automated Early Stopping
Experiment Transfer	Multimetric Optimization	Multisolution Optimization	**Constraint Active Search**



Design

Choose the data
Choose the model
Choose the loss function

Optimize

Bayesian Optimization
Optimizing Many Metrics
Bring Your Own Optimizer
And More…

Explore

Training Metrics
Validation Metrics
Guardrail Metrics
Production Metrics

Putting It All Together

Follow Along On Your Own:

1) Sign Up For A SigOpt Account:

sigopt.com/signup

1) Navigate To The Course Material:

tinyurl.com/SigOptTraining

Thank You

Tobias Andreasen – Machine Learning Engineer, tobias.andreasen@intel.com