DEV DAY

DEV DAY

MLS2

Building Models Automatically with Amazon SageMaker

Julien Simon
Global Evangelist, AI & Machine Learning
Amazon Web Services

@julsimon

Amazon SageMaker helps you build, train, and deploy models

Prepar e

Build

Train & Tune

Deploy & Manage

Web-based IDE for machine learning

Automatically build and train m

Fully managed data processing jobs and data labeling workflows

101011010
010101010
000011110

Collect and prepare training data

One-click collaborative notebooks and built-in, high performance algorithms and models

Choose or build an ML algorithm

One-click training

Set up and manage environments for training

Debugging and optimization

Train, debug, and tune models

Visually track and compare experiments

Manage training runs

One-click deployment and autoscaling

Deploy model in production

Automatically spot concept drift

Monitor models

Add human review of predictions

Fully managed with auto-scaling for 75% less

Modular service and APIs, from experimentation to production

DEV DAY

Automatic Model Tuning with Amazon SageMaker

aws

Algorithms require many hyperparameters

XGBoost

Tree depth
Max leaf nodes
Gamma
Eta
Lambda
Alpha
…

Which ones are the most influential?

Which values should I pick?

How many combinations should I try?

Neural Networks

Number of layers
Hidden layer width
Learning rate
Embedding dimensions
Dropout
…

Setting hyperparameters in Amazon SageMaker

- Built-in algorithms
  - Python parameters set on the relevant estimator (*KMeans*, *LinearLearner*, etc.)
    `xgb.set_hyperparameters(max_depth=5, eta=0.2, gamma=4)`

- Built-in frameworks
  - *hyperparameters* parameter passed to the relevant estimator (*TensorFlow*, *MXNet*, etc.)
  - This must be a Python dictionary
    `tf_estimator = TensorFlow(…, hyperparameters={'epochs’: 1, ‘lr’: ‘0.01’})`
  - Your code must be able to accept them as command-line arguments (script mode)

- Bring your own container
  - *hyperparameters* parameter passed to the *Estimator*
  - This must be Python dictionary
  - It’s automatically copied inside the container: `/opt/ml/input/config/hyperparameters.json`

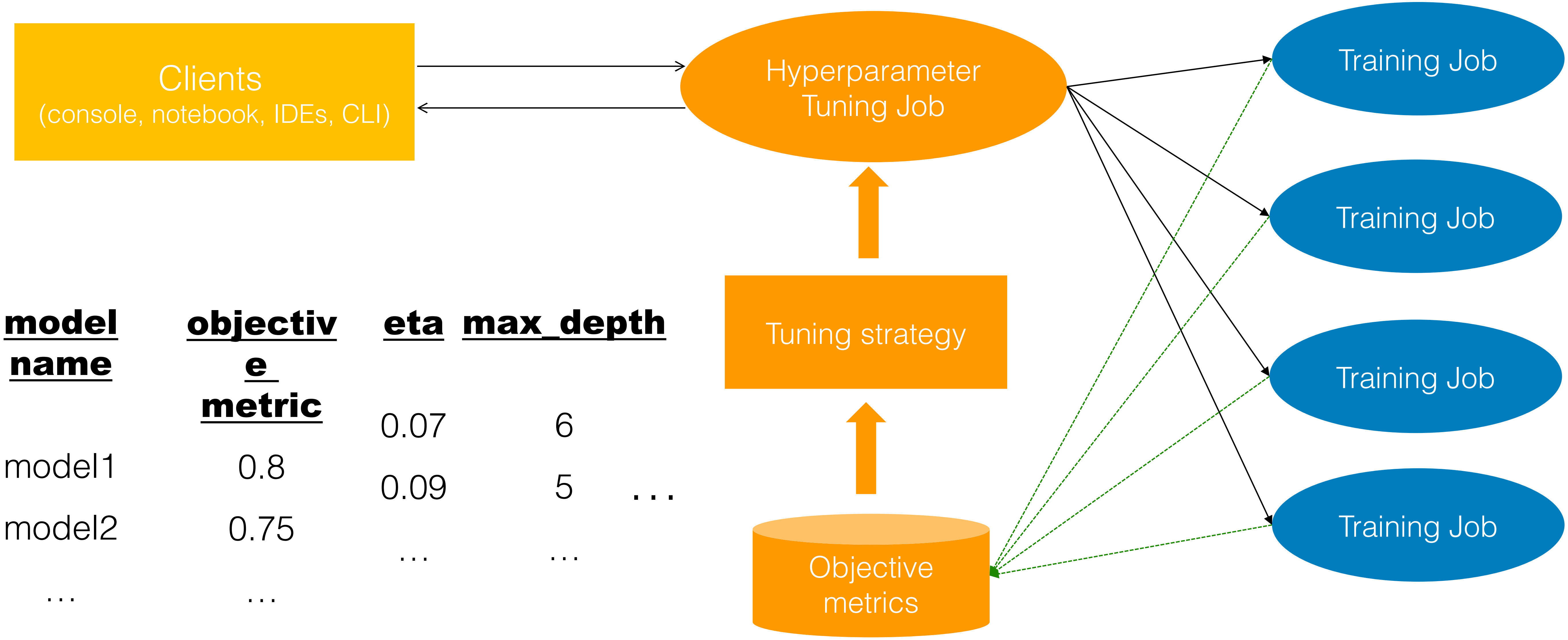
Tactics to find the optimal set of hyperparameters

- Manual Search: ”I know what I’m doing”
- Grid Search: “X marks the spot”
  - Typically training hundreds of models
  - Slow and expensive
- Random Search: “Spray and pray”
  - *« Random Search for Hyper-Parameter Optimization », Bergstra & Bengio, 2012*
  - Works better and faster than Grid Search
  - But… but… but… it’s random!
- Hyperparameter Optimization: use ML to predict hyperparameters
  - Training fewer models
  - Gaussian Process Regression and Bayesian Optimization
  - https://docs.aws.amazon.com/en_pv/sagemaker/latest/dg/automatic-model-tuning-how-it-works.html

Automatic Model Tuning in Amazon SageMaker

1. Define an *Estimator* the normal way
2. Define the metric to tune on
   - Pre-defined metrics for built-in algorithms and frameworks
   - Or anything present in the training log, provided that you pass a regular expression for it
3. Define parameter ranges to explore
   - Type: categorical (avoid if possible), integer, continuous (aka floating point)
   - Range of values
   - Scaling: linear, logarithmic, reverse logarithmic
4. Create an *HyperparameterTuner*
   - *Estimator*, metric, parameters, total number of jobs, number of jobs in parallel
   - Strategy: bayesian (default), or random search
5. Launch the tuning job with *fit()*

Workflow



model name	objective metric	eta	max_depth	
		0.07	6	
model1	0.8	0.09	5	…
model2	0.75	…	…	
…	…			

Automatic Model Tuning in Amazon SageMaker

- You can view ongoing tuning jobs in the AWS console
  - List of training jobs
  - Best training job

- You can also query their status with the SageMaker SDK

- Calling *deploy()* on the *HyperparameterTuner* deploys the best job
  - The best job so far if the tuning job has not yet completed

DEV DAY

Demo

Tips

- Use the bayesian strategy for better, faster, cheaper results
  - Most customers use random search as a baseline, to check that bayesian performs better
- Don’t run too many jobs in parallel
  - This gives the bayesian strategy fewer opportunities to predict
  - Instance limits!
- Don’t run too many jobs
  - Bayesian typically requires 10x fewer jobs than random
  - Cost vs business benefits (beware of diminishing returns)

Resources on Automatic Model Tuning

Documentation

https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning.html
https://sagemaker.readthedocs.io/en/stable/tuner.html

Notebooks

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/hyperparameter_tuning

Blog posts

https://aws.amazon.com/blogs/aws/sagemaker-automatic-model-tuning/
https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-automatic-model-tuning-produces-better-models-faster/
https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-automatic-model-tuning-now-supports-early-stopping-of-training-jobs/
https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-automatic-model-tuning-becomes-more-efficient-with-warm-start-of-hyperparameter-tuning-jobs/
https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-automatic-model-tuning-now-supports-random-search-and-hyperparameter-scaling/

DEV DAY

AutoML with Amazon SageMaker Autopilot

AutoML

- AutoML aims at automating the process of building a model
  - Problem identification: looking at the data set, what class of problem are we trying to solve?
  - Algorithm selection: which algorithm is best suited to solve the problem?
  - Data preprocessing: how should data be prepared for best results?
  - Hyperparameter tuning: what is the optimal set of training parameters?

- Black box vs. white box
  - Black box: the best model only
    → Hard to understand the model, impossible to reproduce it manually
  - White box: the best model, other candidates, full source code for preprocessing and training
    → See how the model was built, and keep tweaking for extra performance

AutoML with Amazon SageMaker Autopilot

- SageMaker Autopilot covers all steps
  - Problem identification: looking at the data set, what class of problem are we trying to solve?
  - Algorithm selection: which algorithm is best suited to solve the problem?
  - Data preprocessing: how should data be prepared for best results?
  - Hyperparameter tuning: what is the optimal set of training parameters?

- Autopilot is white box AutoML
  - You can understand how the model was built, and you can keep tweaking

- Supported algorithms at launch: regression and classification
  - Linear Learner
  - Factorization Machines
  - KNN
  - XGBoost

AutoML with Amazon SageMaker Autopilot

1. Upload the unprocessed dataset to S3

2. Configure the AutoML job
   - Location of dataset
   - Completion criteria

3. Launch the job

4. View the list of candidates and the autogenerated notebooks

5. Deploy the best candidate to a real-time endpoint, or use batch transform

DEV DAY

Demo

Resources on Amazon SageMaker AutoPilot

Documentation
https://docs.aws.amazon.com/sagemaker/latest/dg/autopilot-automate-model-development.html
https://sagemaker.readthedocs.io/en/stable/automl.html

Notebooks
https://github.com/awslabs/amazon-sagemaker-examples/tree/master/autopilot

Blog posts
https://aws.amazon.com/blogs/aws/amazon-sagemaker-autopilot-fully-managed-automatic-machine-learning/

For more content:

- AWS blog: https://aws.amazon.com/blogs/aws/author/julsimon
- Medium blog: https://medium.com/@julsimon
- YouTube: https://youtube.com/juliensimonfr
- Podcast: http://julsimon.buzzsprout.com
- Twitter https://twitter.com/@julsimon

DEV DAY

Thank you!

Julien Simon
Global Evangelist, AI & Machine Learning
Amazon Web Services

@julsimon