AWS re:Invent

AIM361R

Optimizing Your Machine Learning Models on Amazon SageMaker

Julien Simon
AI/ML Evangelist
AWS

Dr Steve Turner
Head of Emerging Technologies, UKIR
AWS

Agenda

1. Welcome & housekeeping
2. An introduction to Automatic Model Tuning (AMT) and AutoML
3. Labs
4. Wrap-up and clean-up

What you'll learn today

- How to use AMT to find optimal model hyperparameters
- How to use AMT to explore deep learning architectures
- How to use Amazon SageMaker Autopilot to find the optimal algorithm, data preprocessing steps and hyper parameters

Our team today

- Antje
- Chris
- Srikanth
- Wei
- Marc
- Michael E
- Matt
- Mike
- Guillaume
- Michael M
- Frank
- Shashank
- John
- Abhi
- Navjot
- Bo
- Boaz
- Mohamed

Housekeeping

- Please be a good neighbor ☺
- Turn off network backups and any network-hogging app
- Switch your phones to silent mode
- Help the people around you if you can
- Don’t stay blocked. Ask questions!

Automatic Model Tuning with Amazon SageMaker

AWS re:Invent

aws

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
…

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

Setting hyperparameters in Amazon SageMaker

- Built-in algorithms
  - Python parameters for the relevant estimator (*KMeans*, *LinearLearner*, etc.)

- Built-in frameworks
  - *hyperparameters* parameter for the relevant estimator (*TensorFlow*, *MXNet*, etc.)
  - This must be a Python dictionary

```
tf_estimator = TensorFlow(…, hyperparameters={'epochs': 1, 'lr': '0.01'})
```

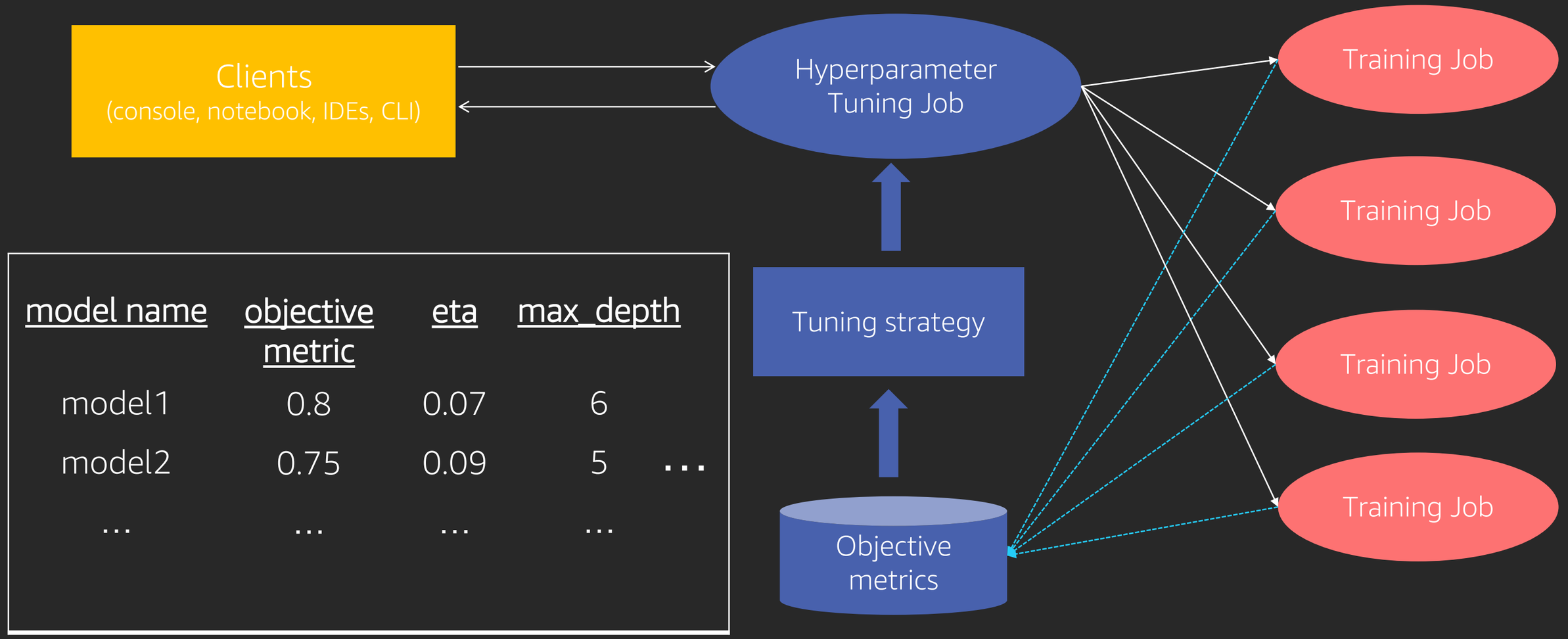
  - Your code must be able to accept them as command-line arguments (script mode)

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

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
   - Scaling: linear (default), 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


Clients
(console, notebook, IDEs, CLI)
Hyperparameter Tuning Job
Training Job
Training Job
Training Job
Training Job
Tuning strategy
Objective metrics
model name
objective metric
eta
max_depth
model1
0.8
0.07
6
model2
0.75
0.09
5
...
...
...
...
...

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

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!

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/

AutoML with Amazon SageMaker Autopilot

AWS re:Invent

aws

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

Labs

Labs

1. Use AMT to find optimal model hyperparameters for XGBoost

2. Use Autopilot to find the optimal algo, preprocessing steps and hyper parameters

3. Use AMT to explore deep learning architectures on Keras

https://gitlab.com/juliensimon/aim361

Thank you!

AWS re:Invent

aws

Please complete the session survey in the mobile app.