CSE 234
Data Systems for Machine Learning

Arun Kumar

Topic 3: Feature Engineering and Model Selection Systems

DL book; Chapters 8.2 and 8.3 of MLSys book
Model Selection in the Lifecycle

Data Scientist/ML Engineer

Source → Build → Deploy

ML/AI + Data Systems Infrastructure

Data acquisition
Data preparation

Feature Engineering
Training & Inference
Model Selection

Serving
Monitoring
Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
   ❖ Feature Engineering
   ❖ Hyperparameter Tuning
   ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
   ❖ Feature Engineering Systems
❖ Advanced Model Selection Systems Issues
Bias-Variance-Noise Decomposition

ML (Test) Error = Bias + Variance + Bayes Noise

Complexity of model/hypothesis space

Discriminability of examples

\[ x = (a,b,c); y = +1 \]
\[ x = (a,b,c); y = -1 \]
Hypothesis Space of Functions

❖ A trained ML model is a parametric prediction function:

\[ f : \mathcal{D}_W \times \mathcal{D}_X \rightarrow \mathcal{D}_Y \]

❖ Hypothesis Space: The set \( \mathcal{H} \) of all possible functions \( f \) that can be represented by a model

❖ Training: Picks one \( f \) from hypo. space; needs estimation procedure (e.g. optimization, greedy, etc.)

❖ Factors that determine hypo. space:
  ❖ Feature representation
  ❖ Inductive bias of model
  ❖ Regularization
Another View of Bias-Variance

- Bias arise because hypo. space does not hold “truth”
  - Shrinking hypo. space raises bias
- Variance arises due to finite training sample
  - Estimation approximately nears truth
  - Shrinking hypo. space lowers variance
3 Ways to Control Learning/Accuracy

- Reduce Bayes Noise:
  - Augment with *new useful* features from appl.
- Reduce Bias:
  - Enhance hypo. space: derive different features; more complex model
  - Reduce shrinkage (less regularization)
- Reduce Variance:
  - Shrink hypo. space: derive different features; drop features; less complex model
  - Enhance shrinkage (more regularization)
The Double Descent Phenomenon

- DL and some other ML families can get arbitrarily complex
  - Can “memorize” entire training set
  - Curiously, variance can drop after rising; bias goes to 0!
  - Interpolation Regime is open question in ML theory

Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
  ❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
  ❖ Feature Engineering Systems
❖ Advanced Model Selection Systems Issues
Unpredictability of Model Selection

- Recall 3 ways to control ML accuracy: reduce bias, reduce variance, reduce Bayes noise
- Alas, the exact raises/drops in errors on *given* training task and sample are *not predictable*
- Need *empirical comparisons* of configurations on data
- Train-validation-test splits; cross-validation procedures
The data scientist/AutoML procedure must steer 3 key activities to alter the Model Selection Triple (MST):

1. **Feature Engineering (FE):** What is/are the domain(s) of the hypo. space(s) to consider?

2. **Algorithm/Architecture Selection (AS):** What exact hypo. space to use (model type/ANN architecture)?

3. **Hyper-parameter Tuning (HT):** How to configure hypo. space shrinkage and estimation procedure approx.?
The data scientist/AutoML procedure must steer 3 key activities to explore the Model Selection Triple (MST)

- FE1
- FE2
- ...

- AS1
- AS2
- ...

- HT1
- HT2
- ...

Train and test model config(s) on ML system

Post-process and consume results

Next iteration

Stopping criterion is application-specific / user-specific on Pareto surface: time, cost, accuracy, tiredness (!), etc.

Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
❖ Feature Engineering
❖ Hyperparameter Tuning
❖ Algorithm/Architecture Selection
❖ Model Selection Systems
❖ Feature Engineering Systems
❖ Advanced Model Selection Systems Issues
Feature Engineering

❖ Process of converting prepared data into a feature vector representation for ML training/inference
  ❖ Aka feature extraction, representation extraction, etc.
  ❖ Activities vary based on data type:

Join and Group Bys
Feature interactions
Feature selection

Temporal feature extraction

Value recoding
Dimensionality reduction
Feature Engineering

❖ Process of converting prepared data into a feature vector representation for ML training/inference
❖ Aka feature extraction, representation extraction, etc.
❖ Activities vary based on data type:

Bag of words
N-grams
Parsing-based features

Signal processing-based features

Deep learning
Transfer learning
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
  ❖ Feature Engineering Systems
❖ Advanced Model Selection Systems Issues
Hyperparameter Tuning

- Most ML models have hyper-parameter knobs

- Learning rate
- Regularization

- Complexity

- Learning rate
- Regularization
- Dropout prob.

- Number of trees
- Max height/min split
- Learning rate?

- Most of them raise bias slightly but reduce variance more

- No hyp.par. settings universally best for all tasks/data
Hyperparameter Tuning

- Common methods to tune hyp. par. configs:
  - Grid search
  - "Random" search

- Manual Tuning
  - NeurIPS: 357, Grid Search: 150, Random Search: 28, Other: 24
  - ICLR: 309, Grid Search: 150, Random Search: 21, Other: 21

References:
Hyperband

- An automated ML (AutoML) procedure for tuning hyp.par.

**Basic Idea:** For iterative procedures (e.g., SGD), stop non-promising hyp.par. configs at earlier epochs
  - Based on multi-armed bandit idea from gambling/RL

**Benefits:**
- Reapportioning resources with *early stopping* may help reach better overall accuracy sooner
- Total resource use may be lower vs grid/random search

- 2 knobs as input:
  - \( R \): Max budget per config (e.g., # SGD epochs)
  - \( \eta \): Stop rate for configs

Hyperband

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

```
input: R, \eta (default \eta = 3)
initialization: s_{\text{max}} = \lceil \log_\eta(R) \rceil, B = (s_{\text{max}} + 1)R

1. for s ∈ \{s_{\text{max}}, s_{\text{max}} - 1, \ldots, 0\} do
2.     n = \left\lfloor \frac{B \eta^s}{R^{s+1}} \right\rfloor, \quad r = R\eta^{-s}
3.     // begin SUCCESSIVEHALVING with (n, r) inner loop
4.     T = \text{get_hyperparameter_configuration}(n)
5.     for i ∈ \{0, \ldots, s\} do
6.         n_i = \left\lfloor n\eta^{-i} \right\rfloor
7.         r_i = r\eta^i
8.         L = \{run\_then\_return\_val\_loss(t, r_i) : t ∈ T\}
9.         T = \text{top_k}(T, L, \lfloor n_i/\eta \rfloor)
10.    end
11. return Configuration with the smallest intermediate loss seen so far.
```

Brackets: independent trials
Akin to random search
Survival of the fittest!

Hyperband

<table>
<thead>
<tr>
<th>i</th>
<th>$n_i$</th>
<th>$r_i$</th>
<th>$n_i$</th>
<th>$r_i$</th>
<th>$n_i$</th>
<th>$r_i$</th>
<th>$n_i$</th>
<th>$r_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>81</td>
<td>1</td>
<td>27</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>6</td>
<td>27</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>27</td>
<td>2</td>
<td>81</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>27</td>
<td>1</td>
<td>81</td>
<td>5</td>
<td>81</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>27</td>
<td>1</td>
<td>81</td>
<td>4</td>
<td>1</td>
<td>81</td>
<td>4</td>
</tr>
</tbody>
</table>

$R = 81$; $\eta = 3$

$n_i$: # hyp.par.configs run
$r_i$: # epochs per config

- Still not as popular as grid/random search; latter is simpler and easier to use (e.g., how to set $R$ and $\eta$?)

Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
  ❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
  ❖ Feature Engineering Systems
❖ Advanced Model Selection Systems Issues
Algorithm Selection

❖ **Basic Goal:** AutoML procedure to pick among a set of interchangeable models (hyp.par. tuning included)

❖ Automate a data scientist’s intuition on feature preprocessing, missing values, hyp.par. tuning, etc.

❖ Many heuristics: AutoWeka, AutoSKLearn, DataRobot, etc.

---

Algorithm Selection

AutoScikitLearn uses a more sequential Bayesian optimization approach

NAS and AutoKeras

- DL NCG arch. akin to model family in classical ML
- Some AutoML tools aim to automate NCG design too

Google’ NAS uses RL to construct and evaluate NCGs

AutoKeras uses Bayesian optimization and has optimized impl.

- Not that popular in practice; compute-intensive; hard to debug

Review Questions

❖ How does regularization affect the B-V-N tradeoff?
❖ How does a key-key join affect the B-V-N tradeoffs?
❖ Name 2 ways to reduce bias in DL model selection.
❖ Why do you think grid/random searches remain such popular forms of hyper-parameter tuning?
❖ Why does Hyperband kill some configs?
❖ Explain 1 pro and 1 con of automated vs manual neural architecture tuning/search.
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
  ❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
  ❖ Feature Engineering Systems
❖ Advanced Model Selection Systems Issues
ML/data mining folks have studied model selection from an algorithmic automation/accuracy standpoint.

But its **resource efficiency** is a pressing ML systems issue:
- Long running times; need lots of CPUs/GPUs
- Cost and energy footprints non-trivial
- If user is in the loop, latency matters too

Need to raise **throughput** of exploring training configs with minimal resource expenses.
Asynchronous Successive Halving (ASHA)

❖ Successor to Hyperband that uses resource more fully

❖ Issues -> New Ideas:
  ❖ Top-k evals in Hyperband are sync. point bottleneck when configs are diverse -> *Asynchronous top-k check*; better for diverse configs
  ❖ Fewer and fewer configs towards bracket end (lower deg. of par.) -> *Add new hyp.par. configs* on the fly; keep all workers busy
  ❖ ASHA adapts AutoML procedure to cluster setting for massive parallel hyp.par tuning

https://blog.ml.cmu.edu/2018/12/12/massively-parallel-hyperparameter-optimization/
Asynchronous Successive Halving (ASHA)

CIFAR10 Using Small Cuda Convnet Model

- Test Error vs. Duration (Minutes)
- 25 workers

LSTM on PTB

- Perplexity vs. Time
- 500 workers
- Total time of weeks!
Introducing Cerebro

❖ **Key Observation:** False dichotomy of 2 main parallelism paradigms in ML for scalable training / model selection

**Task Parallelism**
(Dask, Hyperband, ASHA, Vizier, etc.)

- Config 1
- Config 2
- Config 3

1 worker per config

Worker 1

Worker 2

Worker 3

**Data Parallelism**
(RDBMS, Spark, PS, Horovod, etc.)

- Servers
- 1 config at a time

Worker 1

Worker 2

Worker 3

D_1

D_2

D_3

... + High throughput model selection

+ Best accuracy from Sequential SGD

— Low data scalability; wastes space (copy) or network (remote read)

+ High data scalability via sharding

— BSP does not converge; mini-batch level has high communication costs

— Low throughput overall
Q: Can we get the best of both worlds?
Cerebro’s Model Hopper Parallelism

❖ A new hybrid of task- and data-parallelism for SGD

Epoch 1.2 starts in parallel
Key Insight: SGD is robust to randomness of data ordering

Properties of Model Hopper Parallelism (MOP):

- All configs visit dataset in some sequential order; ensures similar accuracy as task parallelism
- Scheduler keeps all workers busy on shard; just like data parallelism
- No sync. point within an epoch of training all configs; very little idling of workers due to 1 comm. step per epoch
Positioning MOP in context

- MOP is the first known form of Bulk *Asynchronous* Parallelism

**Task-Parallel Systems**

- Dask, Celery, Vizier, Spark-HyperOpt

- No Partitioning (Full replication)

**Data-Parallel Systems**

- MOP/CEREBRO *(This Work)*

- Spark or TF Model Averaging

- Async. Param. Server

- Sync. Param. Server, Horovod

- Bulk (Partitions)

- Fine-grained (Mini-batches)
**Communication Cost Analysis of MOP**

- $p$ workers; $|S|$ configs; $k$ epochs; $b$ batch size; $m$ model size

**Table 2: Communication cost analysis of MOP and other approaches.**

- *Full replication.*
- †Remote reads.
- ‡Parameters for the example: $k = 20$, $|S| = 20$, $p = 10$, $m = 1$GB, $\langle D \rangle = 1$TB, and $|D|/b = 100$K.

<table>
<thead>
<tr>
<th>Method</th>
<th>Comm. Cost</th>
<th>Example‡</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Hopper Parallelism</td>
<td>$kmp</td>
<td>S</td>
</tr>
<tr>
<td>Task Parallelism (FR*)</td>
<td>$p\langle D \rangle + m</td>
<td>S</td>
</tr>
<tr>
<td>Task Parallelism (RR‡)</td>
<td>$k</td>
<td>S</td>
</tr>
<tr>
<td>Bulk Synchronous Parallelism</td>
<td>$2kmp</td>
<td>S</td>
</tr>
<tr>
<td>Centralized Fine-grained</td>
<td>$2kmp</td>
<td>S</td>
</tr>
<tr>
<td>Decentralized Fine-grained</td>
<td>$kmp</td>
<td>S</td>
</tr>
</tbody>
</table>

\[2km(p - 1)|S|\left\lceil \frac{|D|}{(bp)} \right\rceil \quad 72 \text{ PB}\]
Empirical Results

- Cerebro/MOP is near Pareto-optimal on completion time, memory/space efficiency, and network cost.

<table>
<thead>
<tr>
<th>System</th>
<th>Runtime (hrs)</th>
<th>GPU Utili. (%)</th>
<th>Storage Footprint (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF PS - Async</td>
<td>19.00</td>
<td>8.6</td>
<td>250</td>
</tr>
<tr>
<td>Horovod</td>
<td>5.42</td>
<td>92.1</td>
<td>250</td>
</tr>
<tr>
<td>TF Model Averaging</td>
<td>1.97</td>
<td>72.1</td>
<td>250</td>
</tr>
<tr>
<td>Celery</td>
<td>1.72</td>
<td>82.4</td>
<td>2000</td>
</tr>
<tr>
<td>Cerebro</td>
<td>1.77</td>
<td>79.8</td>
<td>250</td>
</tr>
</tbody>
</table>

Figure 10: Reading data from remote storage.
Vision of Cerebro Platform

High-level Model Building APIs

- Transfer Learning
- Ablation Analysis
- Sequence Analysis
- Hyperparameter Tuning
- Architecture Search
- Feature Transfer
- Grouped Learning
- Multi-task Batching

Optimization and Scheduling Layer

Execution and Storage Layer

CLIs

GUIs

TensorBoard

mlflow

https://adalabucsd.github.io/cerebro.html
Human-in-the-loop GUI for Cerebro

- Enables *intermittent* human-in-the-loop execution of configs
- Bridges gap between fully automated heuristics and interactive manual search

New GUI for Cerebro

- Enables intermittent human-in-the-loop execution of configs

### Intermittent Human-in-the-Loop Cerebro

**Upload a model script file OR choose a popular model**

- Drag and Drop or Select a File
- Select a popular model: ResNet-50
- Select this model

### Setup experiment

- Name of experiment (string type)
- Description (optional)
- Name of features (string type, comma separated)
- Name of label (string type)
- Maximum training epochs (integer type)
- Data store prefix path (string type)
- Estimator function name (`<module_name>;<function_name>`)  
- Hyperparameters search strategy (please select)

### Hyperparameter grid table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Hyperparameter Type</th>
<th>Choices</th>
<th>Min Value</th>
<th>Max Value</th>
<th>Quantum</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning_rate</td>
<td>quantum log uniform</td>
<td>-6</td>
<td>-4</td>
<td>2</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>batch_size</td>
<td>categorical</td>
<td>32,256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12_reg</td>
<td>quantum log uniform</td>
<td>-6</td>
<td>-4</td>
<td>2</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>categorical</td>
<td>ResNet50, VGG16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Models

- Input model name and operate on the
- Model Name
- Resume | Delete | Stop | Clone
- Michael | ResNet-Example
- Christopher | ResNet-Example

### Clone Model

- Base Model Name
- Warm Start on Base Model | Start From Scratch

### Hyperparameter grid table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Hyperparameter Type</th>
<th>Choices</th>
<th>Min Value</th>
<th>Max Value</th>
<th>Quantum</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Rate</td>
<td>log uniform</td>
<td>None</td>
<td>-3</td>
<td>-1</td>
<td>0.5</td>
<td>float</td>
</tr>
<tr>
<td>Batch Size</td>
<td>quantum uniform</td>
<td>None</td>
<td>64</td>
<td>64</td>
<td>0</td>
<td>integer</td>
</tr>
<tr>
<td>Drop Out</td>
<td>uniform</td>
<td>None</td>
<td>0.1</td>
<td>0.1</td>
<td>None</td>
<td>float</td>
</tr>
</tbody>
</table>

---

Determined AI Training Platform

https://determined.ai/
Kingpin: Learning over Groups

- ML users often build separate models for separate *groups*, e.g., by country, state, and/or age group
- GROUP BY for ML; “learning over groups”
- **Kingpin**: Jointly optimizes data sharding and model selection scheduling AKA “Grouped Learning”
- A new hybrid of task parallelism and data parallelism

Your Reviews on Cerebro

❖ (Walked through in class)
Discussion on Model Selection Systems
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
  ❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
  ❖ Feature Engineering Systems
❖ Advanced Model Selection Systems Issues
Feature Engineering Systems

- Received less attention than model building systems
- **Key issues they address:**
  - **Usability:** Higher level specification of feature eng. ops
  - **Efficiency:** Automated systems-level optimization
- **Challenges:**
  - Feature eng. is very *heterogeneous*; tough for one tool to capture all ops, data types, etc.
  - *Turing-complete code* rampant in feature eng.; tough for automated optimization
Feature Engineering Systems

- Sample of feature engineering systems:
  - Joins
  - Feature interactions
  - Feature selection
  - Textual / signal proc. features
  - Deep transfer learning

- Columbus
- KeystoneML
- Vista & Nautilus
Feature Selection in Columbus

❖ **Setting**: Exploratory feature subset selection for GLMs on tabular data in R (or NumPy/Pandas)

❖ **Goal**: Reduce compute redundancy and data access at scale

❖ **Approach**: An embedded domain-specific language (DSL) with “logical” ops

---

### Example program in Columbus DSL

```
1  e = SetErrorTolerance(0.01)  # Set Error Tolerance
2  d1 = Dataset("USCensus")    # Register the dataset
3  s1 = FeatureSet(“NumHouses”, ...) # Population-related features
4  l1 = CorrelationX(s1, d1)   # Get mutual correlations
5  s1 = Remove(s1, “NumHouses”) # Drop the feature “NumHouses”
6  l2 = CV(lsquares_loss, s1, d1, k=5) # Cross validation (least squares)
7  d2 = Select(d1, “Income >= 10000”) # Focus on high-income areas
8  s2 = FeatureSet(“Income”, ...) # Economic features
9  l3 = CV(logit_loss, s2, d2, k=5) # Cross validation with (logit loss)
10 s3 = Union(s1, s2) # Use both sets of features
11 s4 = StepAdd(logit_loss, s3, d1) # Add in one other feature
12 Final(s4) # Session ends with chosen features
```

[https://adalabucsd.github.io/papers/2014_Columbus_SIGMOD.pdf](https://adalabucsd.github.io/papers/2014_Columbus_SIGMOD.pdf)
Feature Selection in Columbus

- **Optimization techniques:**
  - Some logical ops have alternate physical ops with different runtimes; Columbus picks automatically
  - **Exact:** Batching, Subset materialization, QR decomposition
  - **Approx.:** Coreset sampling, Warm starting

### Materialization Strategies and ROPs Used by Each Strategy

<table>
<thead>
<tr>
<th>Materialization Strategies</th>
<th>Materialization ROPs</th>
<th>Execution ROPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lazy</td>
<td>N/A</td>
<td>&lt;-, %*, solve</td>
</tr>
<tr>
<td>Eager</td>
<td>&lt;-</td>
<td>%*, solve</td>
</tr>
<tr>
<td>Naïve Sampling</td>
<td>&lt;-, sample</td>
<td>%*, solve</td>
</tr>
<tr>
<td>Coreset</td>
<td>&lt;-, %<em>, solve, sample,</em></td>
<td>%*, solve</td>
</tr>
<tr>
<td>QR</td>
<td>&lt;-, qr</td>
<td>backsolve</td>
</tr>
</tbody>
</table>

### Graphs

- **(a) Error Tolerance**
- **(b) # Features**
- **(c) Size of Overlapping Feature Sets**
- **(d) # Iterations**
- **(e) # Threads**
Feature Pipelines in KeystoneML

❖ Similar to Columbus but *more general*: larger set of classical ML training and feat. eng. ops on top of Spark
❖ Supports text and signal proc.-based image features

```
val textClassifier = Trim andThen
  LowerCase andThen
  Tokenizer andThen
  NGramsFeaturizer(1 to 2) andThen
  TermFrequency(x ⇒ 1) andThen
  (CommonSparseFeatures(1e5), data) andThen
  (LinearSolver(), data, labels)
val predictions = textClassifier(testData)
```

❖ **Optimizations**: Diff. distributed linear solvers at *op level*; at *full pipeline level*: materializing and caching intermediates, sampling, common sub-expression elimination

Feature Transfer in Vista

**Setting**: Pre-trained CNNs are commonly used to extract image feature repr. for multimodal analytics

**Issue**: No single layer of CNN is universally best for downstream accuracy; need to compare multiple layers
Feature Transfer in Vista

But no single CNN layer is always best for accuracy.

Pre-trained Deep CNN

Downstream ML Model Training
Feature Transfer in Vista

❖ **Approach**: Vista casts feature transfer as a *multi-query optimization* problem and creates *materialized views*

Naive prior approach:

Vista’s multi-query optimization:

❖ **Optimizations**: Staging out layer *materializations* avoids compute redundancy; automated memory management
Nautilus: General Transfer Learning

- Generalization of Vista to arbitrary NCGs
- **Optimizations**: *Materializations* to avoid redundancy across both models and evolving data + *model fusion* to reduce memory stalls from GPU

Tradeoffs of Feature Eng. Systems

❖ Pros:
  ❖ High level ops may help improve ML user productivity
  ❖ Automated resource optimization reduces costs

❖ Cons:
  ❖ Lack of sufficient generality
  ❖ ML user needs to (re)learn new APIs; may be complex
  ❖ Extra dependencies and maintenance issues

❖ Some companies now have in-house custom APIs/tools or general code/notebook orchestration for feat. eng. pipelines (not really optimized). More on “feature stores” in Topic 6.

❖ Later in Topic 6, we will study the rise of “Feature Stores”
Outline

❖ Recap: Bias-Variance-Noise. Decomposition
❖ The Model Selection Triple
  ❖ Feature Engineering
  ❖ Hyperparameter Tuning
  ❖ Algorithm/Architecture Selection
❖ Model Selection Systems
  ❖ Feature Engineering Systems
❖ Advanced Model Selection Systems Issues
End-to-End AutoML

- Some tools claim to automate data preparation, feat. eng., and model building holistically

- Unclear how effective they are; no public benchmarks
- Unclear if they do any holistic optimizations, e.g., caching common intermediates, logical-physical separation
- Open questions on systematizing and optimizing end-to-end AutoML
More Effective Architecture Selection

- Most DL users still hand craft NCG for AS
  - Analogous to manual feat. eng. in classical ML
  - NAS / AutoKeras still have only limited adoption

- Open questions on bridging usability gap
  - Need fast human-in-the-loop tools
  - Domain-specific GUI-based AS tools?

Build custom detectors with the compute power of your own AI chips or the cloud

https://www.youtube.com/watch?v=r5aEkpEkDzl&feature=emb_title
Cloud-Native Model Selection

- ML resource availability is now flexible and heterogenous
  - Local machine -> on-premise cluster -> cloud
- **Cloud-native** offers new opportunities/challenges:
  - **Elasticity**: upscale/downscale compute/RAM as needed
  - **Cheap decoupled storage** (e.g., S3)
  - **Cheap ephemeral compute** (e.g., Spot, Serverless)
- Need to redesign model sel. sys. to be cloud-native:
  - Open questions on optimizing resource efficiency vs runtimes vs total cost
Review Questions

❖ Name 3 model sel. systems/approaches for SGD-based ML discussed in class whose communication complexity is independent of SGD batch size.

❖ Does Cerebro affect SGD convergence rates? If yes, how exactly? If not, why not?

❖ Briefly explain 2 cons of building separate feat. eng. systems.

❖ Briefly explain one common systems-level optimization seen in many feat. eng. systems.

❖ Why bother redesigning model sel. systems for the cloud?