Machine Learning Systems Design

Lecture 6: Evaluation

Reply in Zoom chat:

If you could invite anyone to give a guest lecture in this class, who would that be?

CS 329 | Chip Huyen

Logistics

- Notes for lecture 3 & lecture 4 part I out
- Assignment 2 out
  - Join Stanford CS 329S on HuggingFace if you want to use their paid API
  - Should have received OpenAI API invite by now
- Tips for final project
  - Have P0s, P1s, P2s, etc.

Agenda

1. Model development & training
2. AutoML
3. Model calibration
4. Debugging
5. ML system evaluation
6. Model evaluation

Model development & training

Feedback loop

- When model's predictions influence model's performance
  - Model ranks A as most likely to be watched
  - You put A on top
  - More people click on A
  - Model thinks A is good, predicts A even more
- Very common with recommendation systems

Feedback loop: solutions

- Use positional features (hopefully models will give this feature high weight)
  - In training, predicting number of views with 1st_position = True for videos shown on top
  - In production: predicting number of views with 1st_position = False
- Calibrate predictions
  - Mix both most likely + novelty
- Limit number of times a sample is shown
  - Put a threshold on views = 1000, then show it only 50% now

Data parallelism

split the data across devices

each device sees a fraction of the batch

each device replicates the model

each device replicates the optimizer

GPT-3: 3.2M batch size

1M samples

- 1000 samples/batch/machine
- 1 machine: 1000 batches
- 100 machines: **10 batches**

Scaling SGD Batch Size to 32K for ImageNet Training (You et al., 2017)

Data parallelism

split the data across devices

each device sees a fraction of the batch

each device replicates the model

each device replicates the optimizer

GPT-3: 3.2M batch size

Challenge 1: Learning rate

- Too small -> too long to converge
- Too large -> unstable learning

Scaling SGD Batch Size to 32K for ImageNet Training (You et al., 2017)

Data parallelism: LR scaling

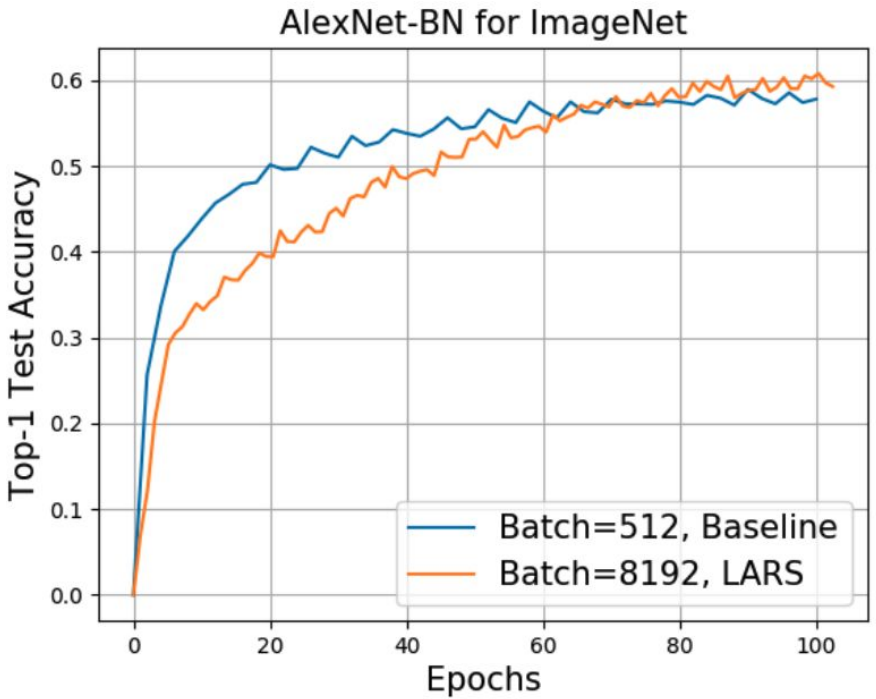
split the data across devices

each device sees a fraction of the batch

each device replicates the model

each device replicates the optimizer

GPT-3: 3.2M batch size



[Scaling SGD Batch Size to 32K for ImageNet Training](#) (You et al., 2017)

Data parallelism: gradient updates

Challenge 2: How to aggregate gradient updates?

- Synchronous: have to wait for stragglers
- Asynch: gradients become stale

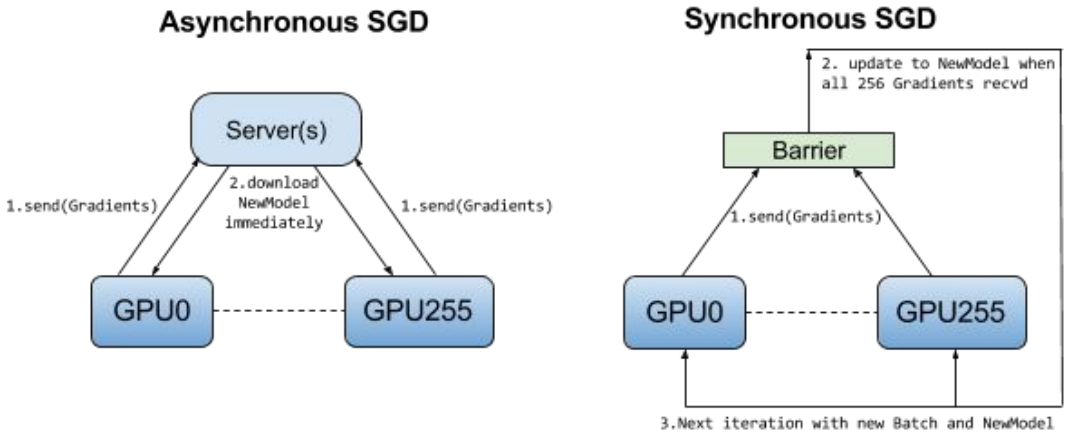


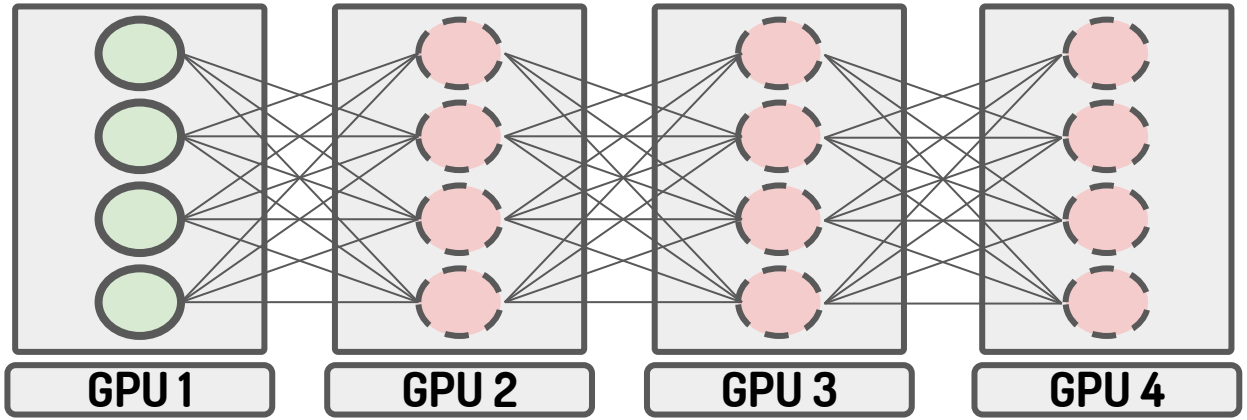
Image from Distributed TensorFlow (Jim Dowling, O'Reilly 2017)

Model parallelism

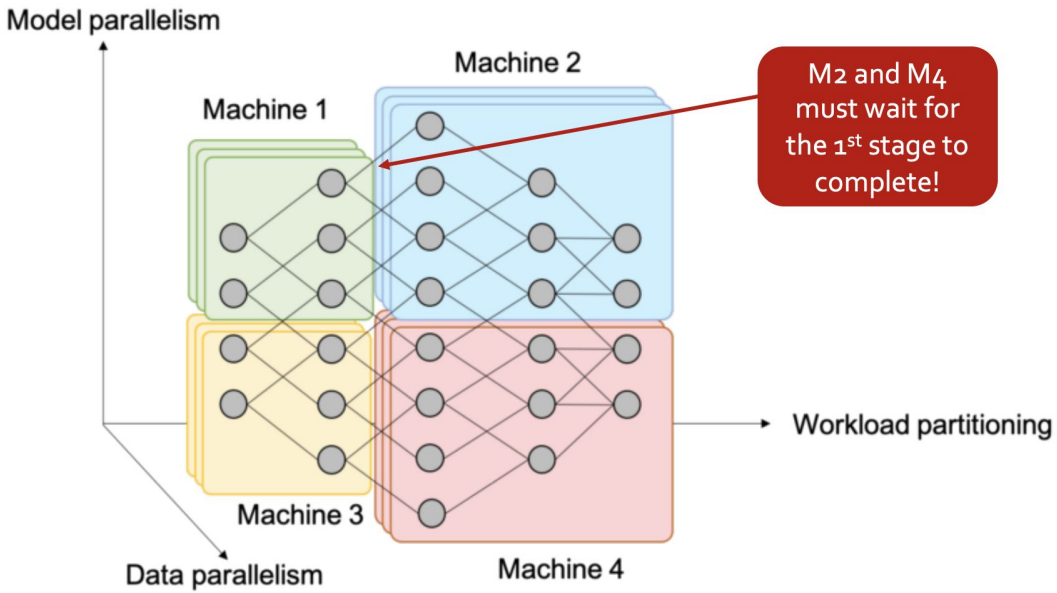
split the model across devices

each device runs a fragment of the model

- Split by:
  - layers
  - forward-backward passes
  - etc.
- Challenge: load balancing?



Model parallelism and data parallelism

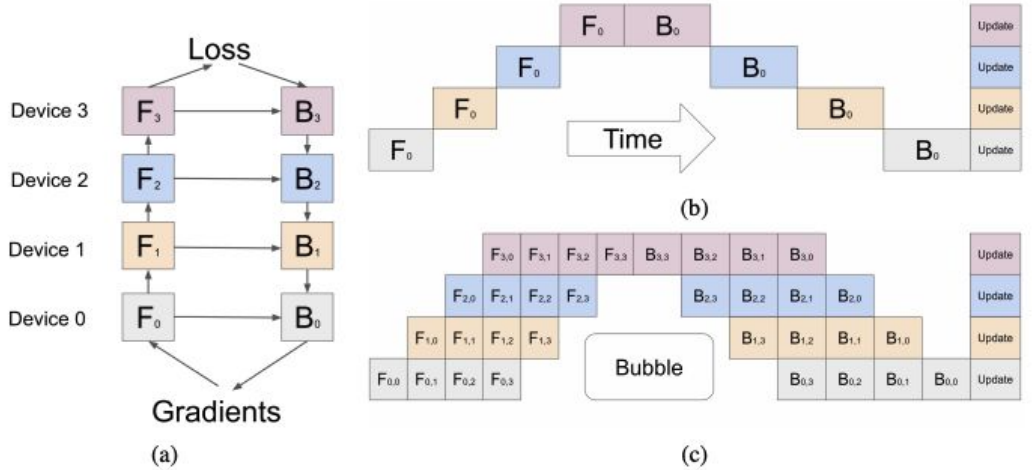


Mining Massive Datasets course,Stanford, lecture 13. Jure Leskovec 2020.

Pipeline parallelism: model + data parallelism

- Divide
  - layers into stages
  - mini-batches into micro-batches
- Spread them across devices

Figure 2: (a) An example neural network with sequential layers is partitioned across four accelerators. F_k is the composite forward computation function of the k-th cell. B_k is the back-propagation function, which depends on both B_{k+1} from the upper layer and F_k. (b) The naive model parallelism strategy leads to severe under-utilization due to the sequential dependency of the network. (c) Pipeline parallelism divides the input mini-batch into smaller micro-batches, enabling different accelerators to work on different micro-batches simultaneously. Gradients are applied synchronously at the end.

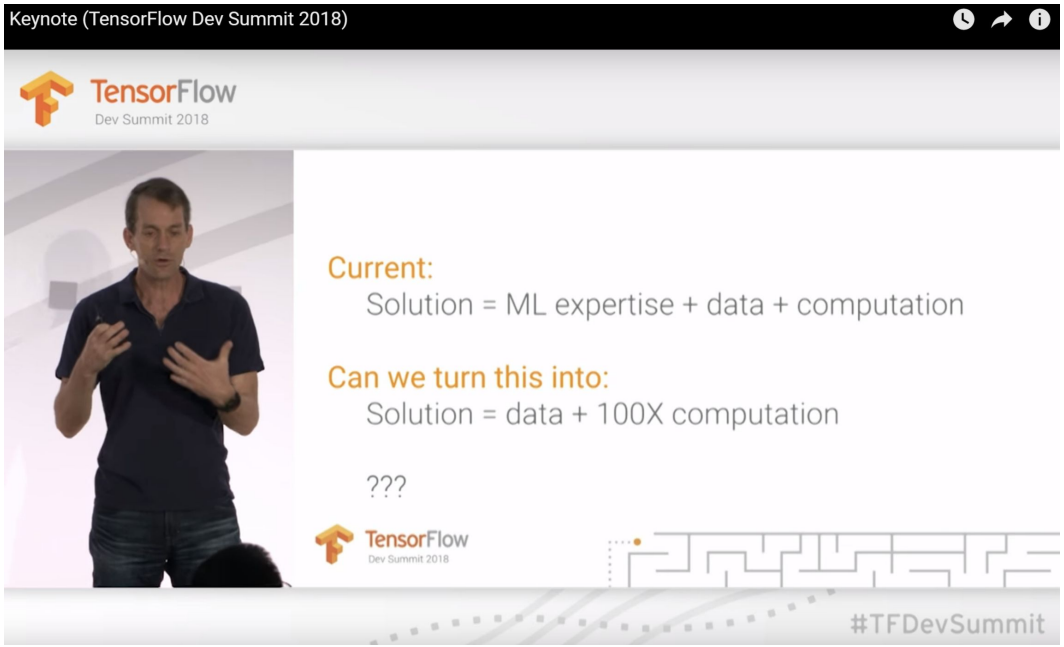


GPipe: Easy Scaling with Micro-Batch Pipeline Parallelism (Huang et al., 2018)

AutoML

AutoML

- A good ML researcher is someone who will automate themselves out of job
- Google: what if we replace ML experts with 100x compute?



AutoML

- Soft AutoML:
  - hyperparameter tuning
- Hard AutoML
  - neural architecture search
  - learned optimizer

More computationally expensive

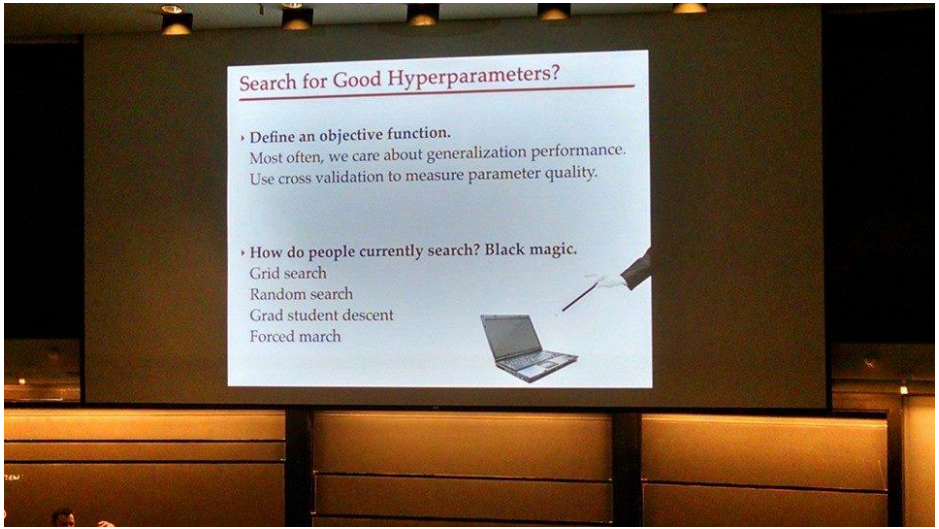
Soft AutoML: Hyperparameter tuning

- Weaker models with well-tuned hyperparameters can outperform stronger, fancier models
  - On the State of the Art of Evaluation in Neural Language Models (Melis et al. 2018)
- Many different hyperparameters:
  - batch size, learning rate, quantization, number of hidden layers, number of hidden units, dropout probability, etc.

https://twitter.com/GuyZys/status/592847074170896384/photo/1

Soft AutoML: Hyperparameter tuning

- Graduate Student Descent (GSD):
  - A graduate student fiddles around with the hyperparameters until the model works



https://twitter.com/GuyZys/status/592847074170896384/photo/1

NAS: Neural architecture search

- **Search space**
  - Set of operations
    - e.g. convolution, fully-connected, pooling
  - How operations can be connected

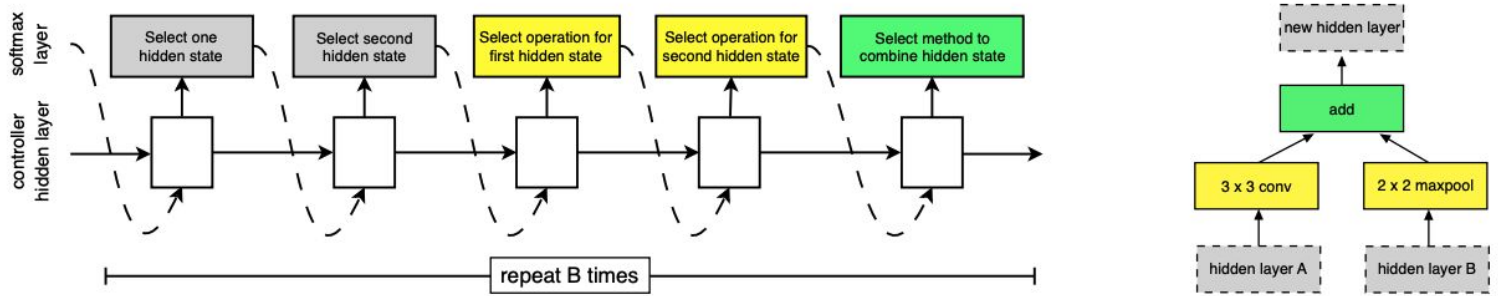


Figure 3. Controller model architecture for recursively constructing one block of a convolutional cell. Each block requires selecting 5 discrete parameters, each of which corresponds to the output of a softmax layer. Example constructed block shown on right. A convolutional cell contains B blocks, hence the controller contains $5B$ softmax layers for predicting the architecture of a convolutional cell. In our experiments, the number of blocks B is 5.

Learning Transferable Architectures for Scalable Image Recognition (Zoph et al., 2017)

NAS: Neural architecture search

- Search space
- **Performance estimation strategy**
  - How to evaluate **many** candidate architectures?
  - Ideal: should be done without having to re-construct or re-train them from scratch.

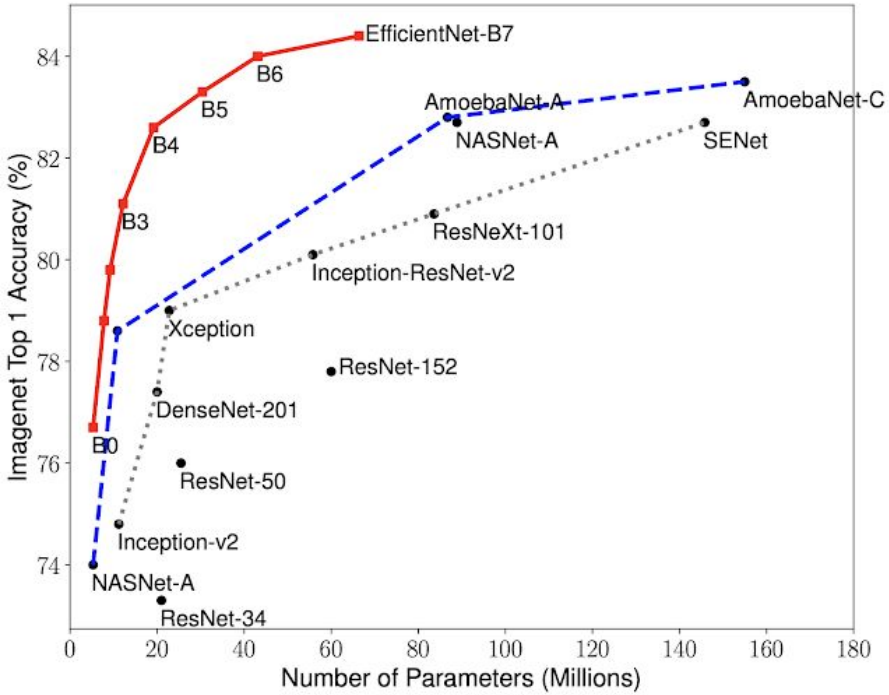
NAS: Neural architecture search

- Search space
- Performance estimation strategy
- **Search strategy**
  - Random
  - Reinforcement learning
    - reward the choices that improve performance estimation
  - Evolution
    - mutate an architecture
    - choose the best-performing offsprings
    - so on

NAS: Neural architecture search

- **Search space**
- **Performance estimation strategy**
- **Search strategy**

Very successful



EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (Tan et Le., 2019)

Learning: architecture + learning algorithm

- Learning algorithm:
  - A set of functions that specifies how to update the weights.
  - Also called **optimizers**
    - Adam, Momentum, SGD

Learned optimizer

Deep learning

engineering features → learning features

SIFT (Lowe et. al. 1999)
HOG (Dalal et. al. 2005)

LeNet (LeCun et. al. 1998)
AlexNet (Krizhevsky et. al. 2012)

Meta learning

engineering to learn → learning to learn

SGD (Robbins et. al. 1951, Bottou 2010)
Autoencoders (Hinton et. al. 2006)

Learning To Learn (Hochreiter et. al. 2001)
Learned Optimizers (Andrychowicz et. al. 2016, Li et. al. 2016, Wichrowska et. al. 2017, Metz et. al. 2018, 2019)

Slide courtesy of Luke Metz

Learned optimizer

- Learn how to learn on a set of tasks
- Generalize to new tasks

Using a thousand optimization tasks to learn hyperparameter search strategies

Luke Metz [1] Niru Maheswaranathan [1] Ruoxi Sun [1] C. Daniel Freeman [1] Ben Poole [1] Jascha Sohl-Dickstein [1]

Slide courtesy of Luke Metz

Learned optimizer

- Learn how to learn on a set of tasks
- Generalize to new tasks
- The learned optimizer can then be used to train a better version of itself!

Using a thousand optimization tasks to learn hyperparameter search strategies

Luke Metz [1] Niru Maheswaranathan [1] Ruoxi Sun [1] C. Daniel Freeman [1] Ben Poole [1] Jascha Sohl-Dickstein [1]

Slide courtesy of Luke Metz

Model calibration

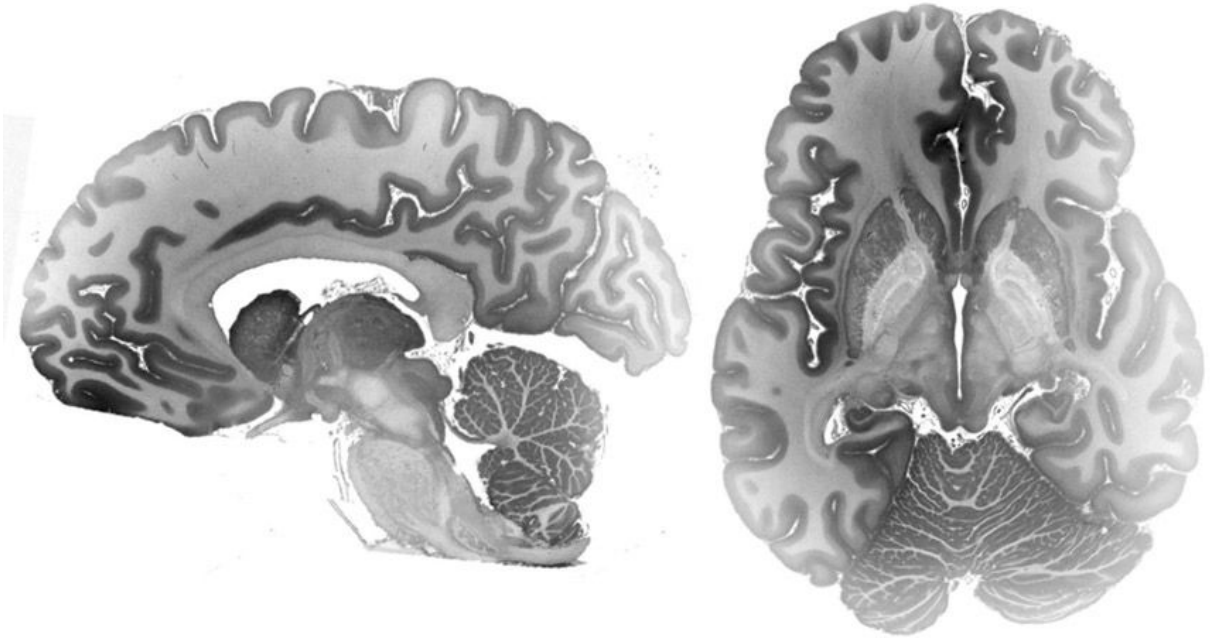
Anastasios Angelopoulos
UC Berkeley

System-level vs. instance-level

- System-level metrics: measuring system's performance **on average**
  - Good when we want a lot of approximate predictions
- Instance-level metrics: measuring system's performance **on each sample**
  - Required when a single wrong prediction can be catastrophic
  - Accuracy is not enough when class frequencies are uneven!
    - What if 80% of patients have class `normal` and 20% have class `cancer`?

A model could have 80% accuracy by always outputting `normal`!

Decision making requires uncertainty



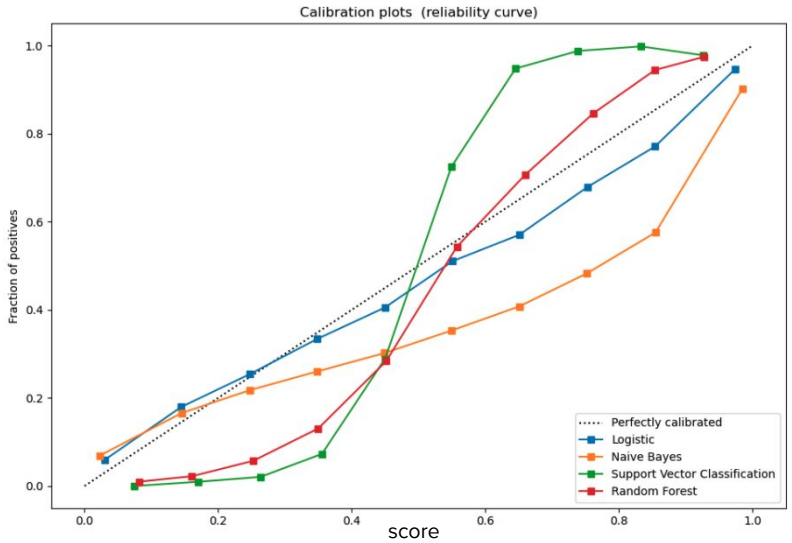
Decision-making algorithms require uncertainty to “rule-in” or “rule-out” risky classes.
The patient may be normal, but if there’s a 5% chance of deadly cancer, the doctor should know.

https://www.biorxiv.org/content/10.1101/649822v2.full.pdf

The binary case: calibration

$$\mathbb{P}[\text{The top class is correct} \mid \text{its score is } p] = p$$

If proba for top class is 0.8, then it's correct 80% of the time



Need to ensure the **top class is correct on average**

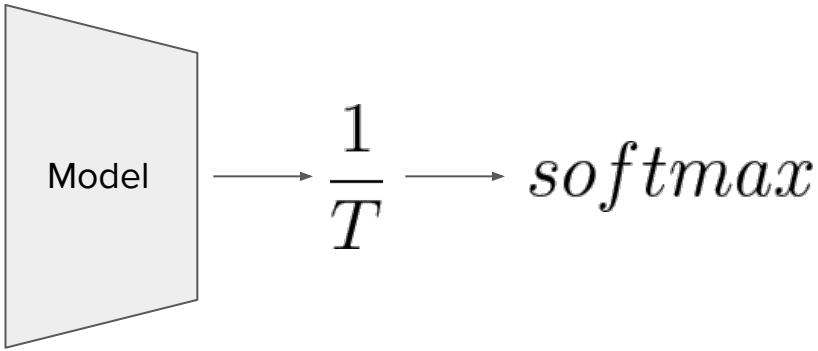
Image from Probability calibration (sklearn)

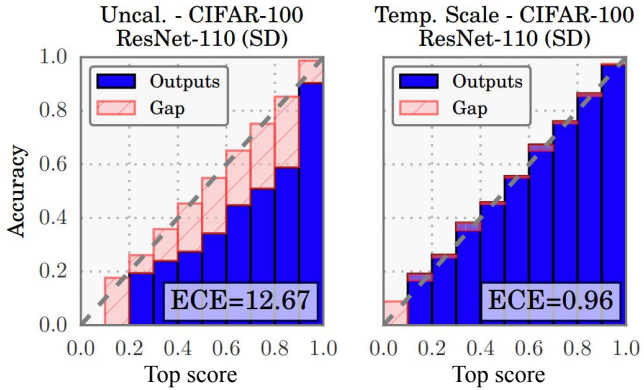
The binary case: calibration

$$\mathbb{P}[\text{The top class is correct} \mid \text{its score is } p] = p$$

- Need to ensure the **top class is correct on average**.
- The score of the top class corresponds to the confidence of your model.
  - E.g. if the calibrated score < 90%, say **“I don’t know!”** (We get a guarantee here!)

The most common way is **platt scaling/temperature scaling** (Guo et al.) .





Multiclass problems can be reduced to pseudo-binary by asking, “is my top class correct?”

The n-ary case: prediction sets

{ fox squirrel 0.99 }

{ fox squirrel 0.82, gray fox, 0.03, bucket, 0.02, rain barrel 0.02 }

{ marmot 0.30, fox squirrel 0.22, mink 0.18, weasel 0.16, beaver 0.03, polecat 0.01 }

Rather than predict one output, we **predict a set** to aid the decision-maker.

$$P(Y \in \mathcal{C}(X)) \geq 1 - \alpha$$

These sets should:

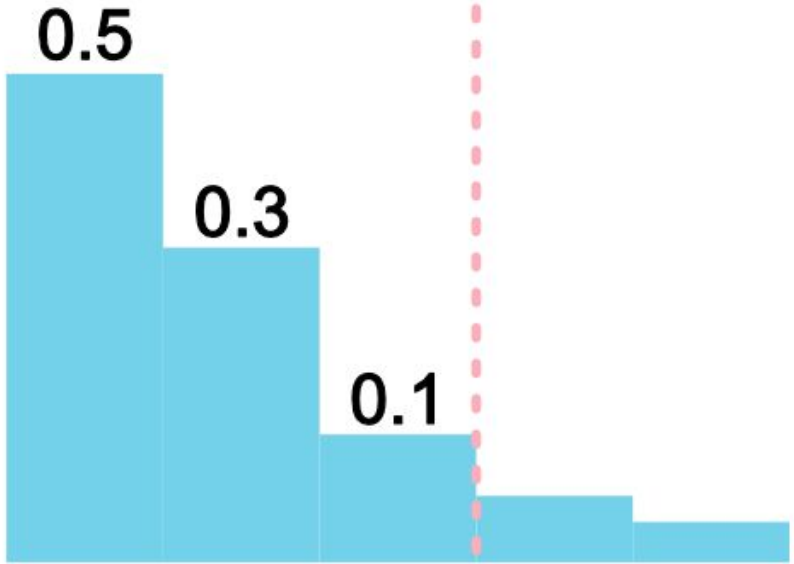
- Have **correct coverage** (i.e. contain the correct class with high probability)
- Be as **small** as possible
- Be **adaptive**: the sets should be smaller for ‘easy’ than ‘hard’ examples

We only assume the data are i.i.d. (nothing about the true distribution, algorithm, etc)

$$(x_i, y_i) \sim \mathbb{P}$$

The n-ary case: a naive strategy

Simple strategy: sort classes from highest-to-lowest probability. Stop at 90%.



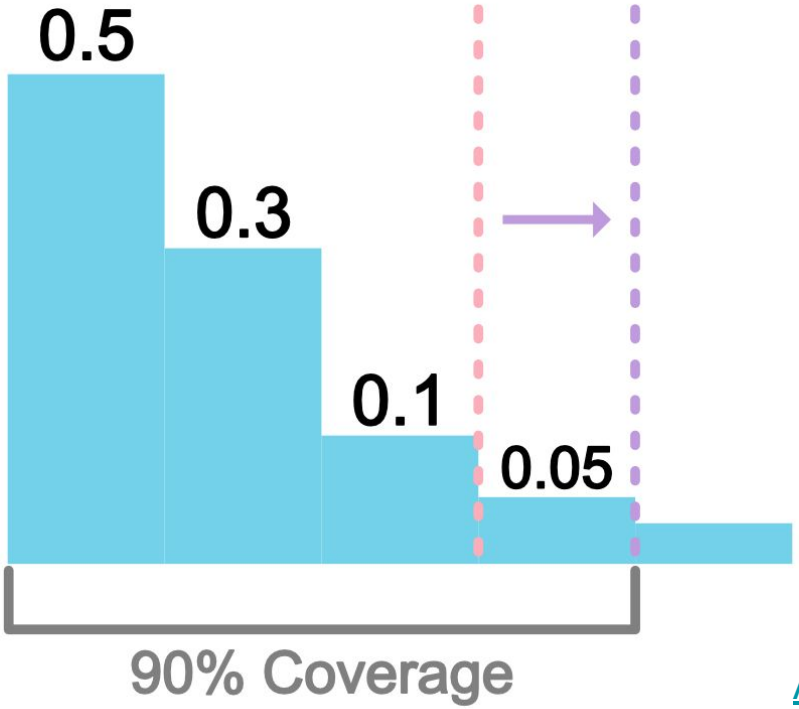
Problems

(1) **Incorrect coverage** due to estimation error

(2) Small probability estimates are unreliable -> **large & unstable sets**

The n-ary case: conformal prediction

Idea: choose threshold based on a holdout set



E.g., choose 95% nominal coverage to get 90% coverage on test data

- Benefits:
  - **Exact coverage** in finite samples
  - Sets are **adaptive** to difficulty of an example
  - Computationally **cheap** (with sample splitting)

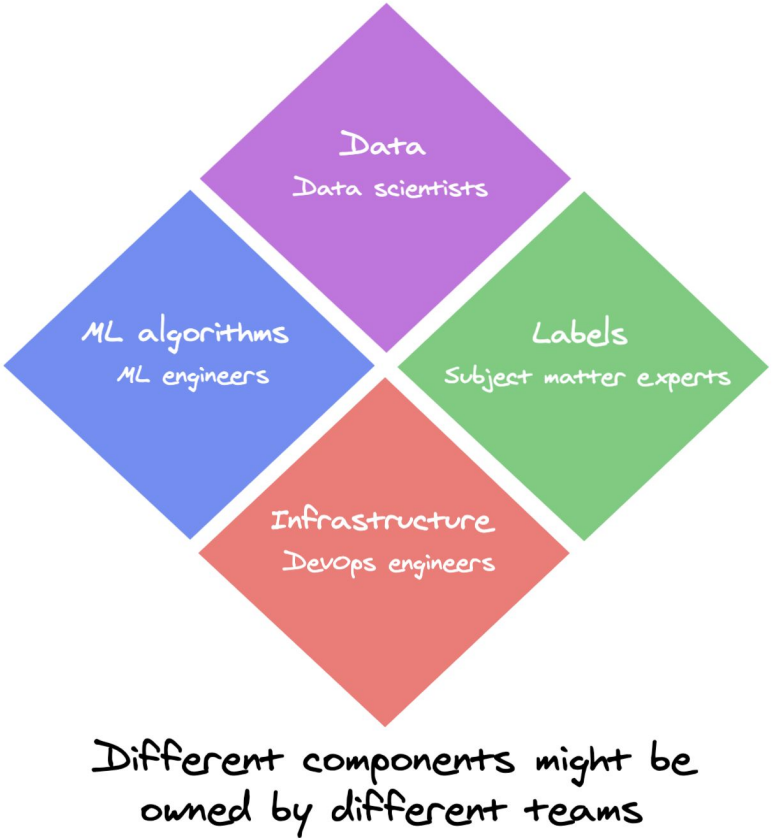
$$P(Y \in \mathcal{C}(X)) \geq 1 - \alpha$$

Angelopoulos, Bates, Malik, and Jordan (arxiv:2009.14193)
Romano, Sesia, and Candes (*arXiv:2006.02544*)
Vladimir Vovk (*Algorithmic Learning in a Random World*)

Debugging

Why is debugging ML systems hard?

1. Cross-functional complexity
    a. Components owned by different teams
    b. Errors can be because of any or a combination



Why is debugging ML systems hard?

1. Cross-functional complexity
2. Silent errors
   a. Developers don't notice
   b. Users don't notice

Facebook translates 'good morning' into 'attack them', leading to arrest

Palestinian man questioned by Israeli police after embarrassing mistranslation of caption under photo of him leaning against bulldozer

Why is debugging ML systems hard?

1. Cross-functional complexity
2. Silent errors
3. Slow updating cycle
    a. Might have to wait a long time to see whether a bug is fixed

https://xkcd.com/303/

Sources of errors

- **Theoretical constraints**
  - wrong assumptions, poor model/data fit, problems impossible to solve with ML, etc.
- **Poor model implementation**
  - layers not connected, call model.train() instead of model.eval()
- **Poor choice of hyperparameters**
- **Data problems**
  - mismatched inputs/labels, over/under-preprocessed data, noisy labels, bad splits
- **Feature engineering problems**
  - Features lacking predictive power
  - Data leakage
- **Pipeline problems**
  - changes in training pipeline not replicated in inference pipeline
- **etc.**

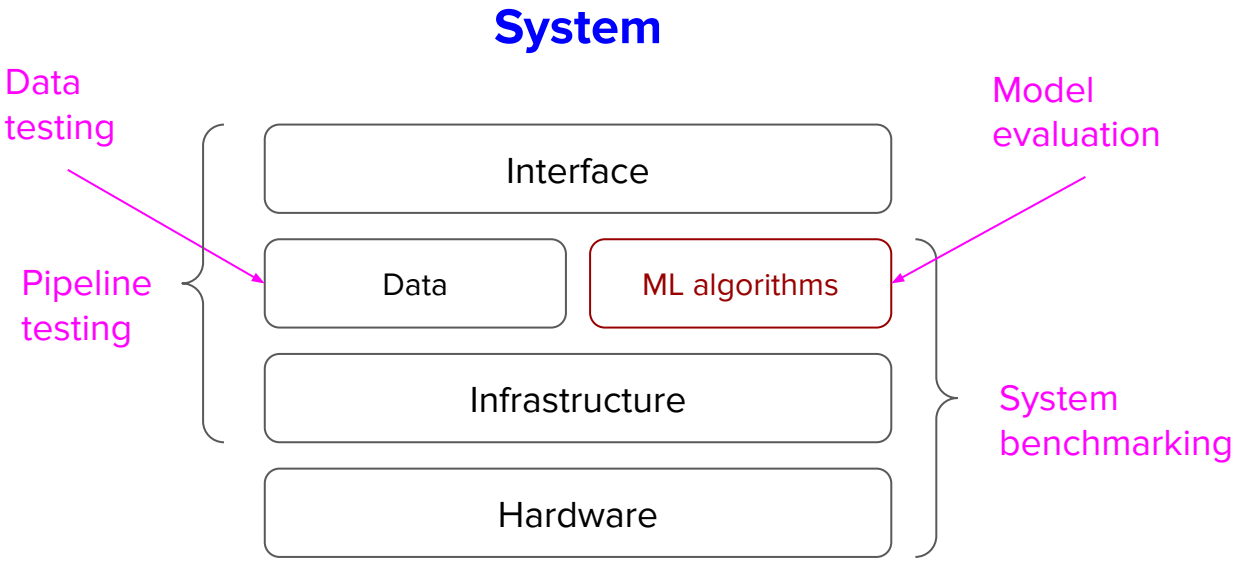
Breakout exercise (5 minutes, group of 5)

How might you address the following sources of errors?

- **Poor implementation**
  - layers not connected, call model.train() instead of model.eval()
- **Poor choice of hyperparameters**
- **Data problems**
  - mismatched inputs/labels, over/under-preprocessed data, noisy labels, bad splits
- **Feature engineering problems**
  - Features lacking predictive power
  - Data leakage
- **Pipeline problems**
  - changes in training pipeline not replicated in inference pipeline

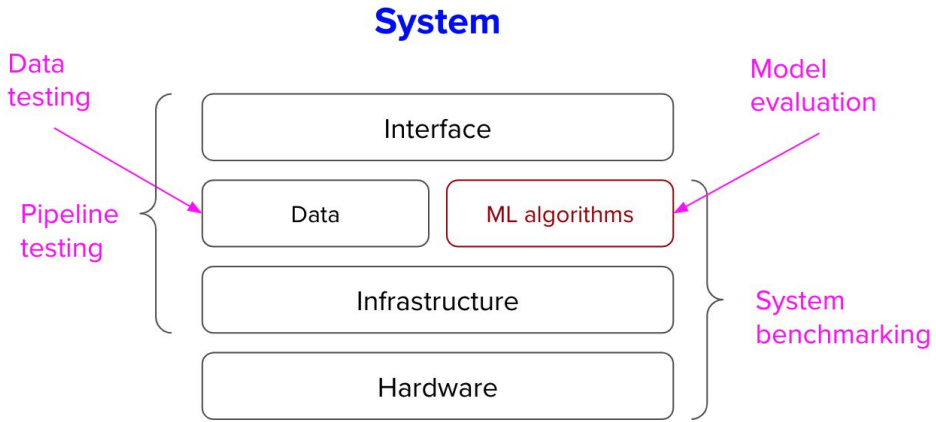
ML system evaluation

ML system evaluation & testing



ML system evaluation

- **Data testing**: ensuring new data satisfies your assumptions
  - **NOT covered here**: test to see how good a new sample is for your model
- **Pipeline testing**: ensuring your pipeline is set up correctly
- **〜System benchmarking〜**: reporting your system's overall performance on well-defined tasks, metrics, and rules, usually for the purpose of comparison
- **Model evaluation**: evaluating how good your ML model is



Pipeline testing

- Consistency
  - Does your system make the same prediction on the same input in during training & inference?
  - Does your system make the same prediction on the same input on different runs?
- Integration between components
  - Can the entire pipeline run end-to-end?
    - From data loading to feature engineering to training to serving?
- [Hard] Visibility
  - Can you map a sample's transformation through your entire pipeline?
- Edge cases
  - What happens if inputs violate assumptions?

Tips for ensuring your pipeline correctness

- Start simple and gradually add more components
  - Validate prediction assumption
  - Validate pipeline
  - Understand the impact of each added component
- Overfit a single batch
  - If a model can’t memorize a small amount of data perfectly, something is wrong
- Make sure an input has same output in both training and inference
- Write an end-to-end test
- Set random seeds
  - For reproducibility
- Change random seeds
  - The more sensitive your model is to inconsequential things, the harder it’ll be retrain/maintain it

Data testing: label quality & correlation

- Manually check quality of your labels
- If labels come from multiple sources, check correlation/conflicts among them

Data testing: feature engineering

- Do features make sense?
- When new features are added, are old features still necessary?
- Are missing values replaced with mean/default values?
- Sanity-check after each transformation
- ...

Data testing: assumptions

- Models embed assumptions about data
  - Both **inputs and outputs**
  - Example assumptions:
    - geotags are available -> model performs weird when geotags are null
    - answers' lengths are within 1 minute
- How to know if these assumptions hold in production?

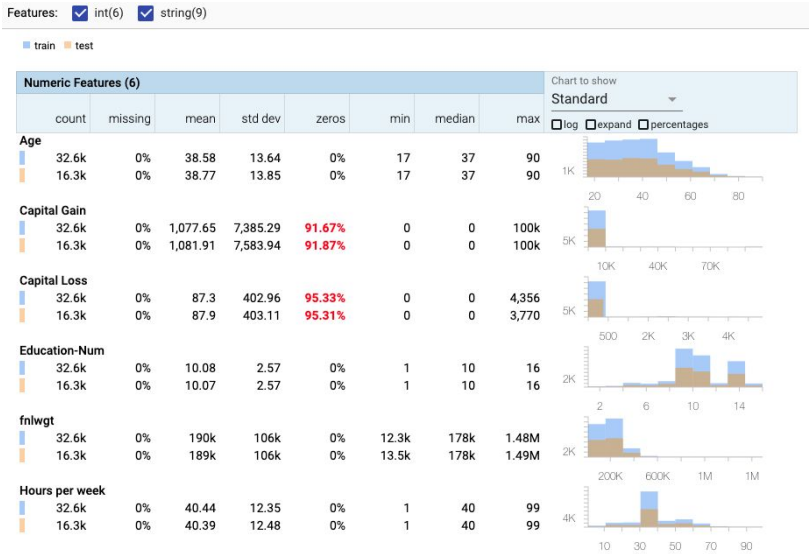
Data testing: assumptions

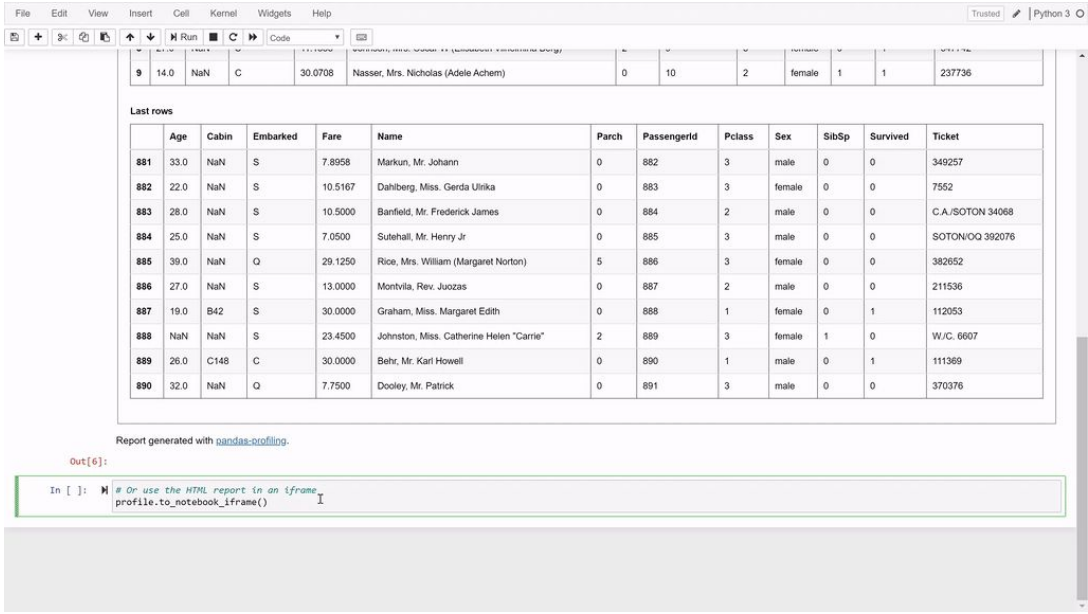
- Pre-train
  - Data exploration
    - Some data might make no sense to non-experts
  - **Data profiling & visualization**
  - **Create assertions**
- Post-train
  - Make sure each new data sample passes assertions

```
StartTime,Dur,Proto,SrcAddr,Sport,Dir,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes,Label
2011/08/18 10:21:46.633335,1.060248,tcp,93.45.239.29,1611,   ->,147.32.84.118,6881,S_RA,0,0,4,252,132,flow=Background-TCP-Attempt
2011/08/18 10:19:49.027650,279.349152,tcp,62.240.166.118,1031,  <?>,147.32.84.229,13363,SRPA_PA,0,0,15,1318,955,flow=Background-TCP-Attempt
2011/08/18 10:22:07.160628,166.390015,tcp,147.32.86.148,58067,   ->,66.235.132.232,80,SR_SA,0,0,3,212,134,flow=Background-TCP-Established
2011/08/18 10:26:02.052163,1.187083,tcp,147.32.3.51,3130,   ->,147.32.84.46,10010,S_RA,0,0,4,244,124,flow=Background-TCP-Attempt
2011/08/18 10:26:52.226748,0.980571,tcp,88.212.37.169,3134,   ->,147.32.84.118,6881,S_RA,0,0,4,244,124,flow=Background-TCP-Attempt
2011/08/18 10:27:57.611601,1.179357,tcp,94.44.60.103,49905,   ->,147.32.84.118,6881,S_RA,0,0,4,268,148,flow=Background-TCP-Attempt
2011/08/18 10:28:15.463038,1.140237,tcp,2.159.127.100,1378,   ->,147.32.84.118,6881,S_RA,0,0,4,252,132,flow=Background-TCP-Attempt
2011/08/18 10:28:37.132447,12.058948,tcp,213.233.154.219,36381,   ->,147.32.84.229,13363,SRA_SA,0,0,7,508,208,flow=Background-TCP-Established
2011/08/18 10:29:03.150806,0.942243,tcp,88.212.37.169,62055,   ->,147.32.84.118,6881,S_RA,0,0,4,244,124,flow=Background-TCP-Attempt
2011/08/18 10:29:43.750355,3.547213,tcp,95.210.161.212,58571,   ->,147.32.84.118,6881,S_RA,0,0,8,488,248,flow=Background-TCP-Attempt
```

Data profiling & visualization

- Examining data & collecting:
  - statistics
  - informative summaries
- pandas-profiling
- facets





Data assertions

- Common sense assertions
  - “the” is most common word in English
- Categorical data belongs to a predefined set
- Continuous data is within a range
- Outputs should follow a distribution
- Inputs should follow a distribution

Data testing: specifying schema in your code

```python
from pydantic import BaseModel, ValidationError, validator


class UserModel(BaseModel):
    name: str
    username: str
    password1: str
    password2: str

    @validator('name')
    def name_must_contain_space(cls, v):
        if ' ' not in v:
            raise ValueError('must contain a space')
        return v.title()

    @validator('password2')
    def passwords_match(cls, v, values, **kwargs):
        if 'password1' in values and v != values['password1']:
            raise ValueError('passwords do not match')
        return v

    @validator('username')
    def username_alphanumeric(cls, v):
        assert v.isalnum(), 'must be alphanumeric'
        return v
```

```python
user = UserModel(
    name='samuel colvin',
    username='scolvin',
    password1='zxcvbn',
    password2='zxcvbn',
)
print(user)
#> name='Samuel Colvin' username='scolvin' password1='zxcvbn' password2='zxcvbn'

try:
    UserModel(
        name='samuel',
        username='scolvin',
        password1='zxcvbn',
        password2='zxcvbn2',
    )
except ValidationError as e:
    print(e)
    """
    2 validation errors for UserModel
    name
      must contain a space (type=value_error)
    password2
      passwords do not match (type=value_error)
    """
```

https://pydantic-docs.helpmanual.io/usage/validators/

Data testing: creating data schema

Table shape

- `expect_column_to_exist`
- `expect_table_columns_to_match_ordered_list`
- `expect_table_columns_to_match_set`
- `expect_table_row_count_to_be_between`
- `expect_table_row_count_to_equal`
- `expect_table_row_count_to_equal_other_table`

Missing values, unique values, and types

- `expect_column_values_to_be_unique`
- `expect_column_values_to_not_be_null`
- `expect_column_values_to_be_null`
- `expect_column_values_to_be_of_type`
- `expect_column_values_to_be_in_type_list`

```
expect_column_values_to_be_between(
    column="room_temp",
    min_value=60,
    max_value=75,
    mostly=.95
)
```


"Values in this column should be between 60 and 75, at least 95% of the time."

"Warning: more than 5% of values fell outside the specified range of 60 to 75."

https://github.com/great-expectations/great_expectations

Benchmarking

- Reproducible experiments with well-defined tasks, metrics, and rules
- Guide development
  - E.g. shared tasks on machine translation from 2006
- Measure progress
  - E.g. benchmark datasets/tasks for research
- Customers might require it (possibly to compare with other solutions)
  - E.g. latency, memory usage, prediction cost, accuracy of your model on their provided data
- Compete with other systems
  - E.g. MLPerf

⚠️ Designing benchmarks can be very, very difficult ⚠️

One benchmark, two headlines

cloud.google.com › products › ai-machine-learning › g...

Google wins MLPerf benchmark contest with fastest ML ...

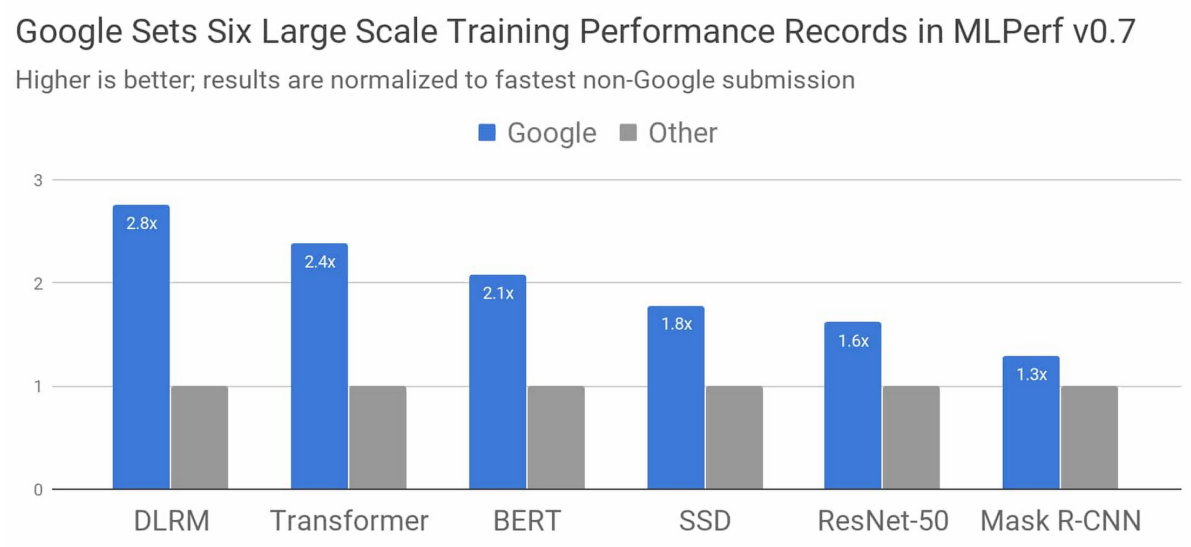
Jul 29, 2020 — **Google** breaks AI performance records in **MLPerf** with world's fastest training supercomputer. gcp **ml perf**.jpg. Naveen Kumar.

blogs.nvidia.com › mlperf-training-benchmark-records

NVIDIA Breaks 16 Records in MLPerf AI Benchmarks | NVIDIA ...

Jul 29, 2020 — **NVIDIA** Breaks 16 AI Performance Records in Latest **MLPerf** Benchmarks. **NVIDIA** A100 GPUs, DGX SuperPOD systems declared world's fastest ...

One benchmark, two results



Model evaluation

Model evaluation

(Evaluating against baselines already covered in a previous lecture)

1. Slice-based evaluation
2. Perturbation evaluation
    a. Invariance tests
    b. Directional expectation tests
3. Ablation study

Slice-based evaluation

Models perform differently on different slices

- Classes
  - Might perform worse on minority classes
- Subgroups
  - Gender
  - Location
  - Time of using the app
  - etc.

Models perform the same on different slices with different cost

- User churn prediction
  - Paying users are more critical
- Predicting adverse drug reactions
  - Patients with underlying conditions are more critical

⚠️ Focusing on improving only overall metrics might hurt performance on subgroups ⚠️

Simpson’s paradox

- Models A and B to predict whether a customer will buy your product
- A performs better than B overall
- B performs better than A on both female & male customers

Simpson's paradox

	Treatment 1	Treatment 2
Group A	**93% (81/87)**	87% (234/270)
Group B	**73% (192/263)**	69% (55/80)
Overall	78% (273/350)	**83% (289/350)**

Numbers from a kidney stone treatment study. (Charig et al., 1986)

Simpson's paradox: Berkeley graduate admission '73

	All		Men		Women	
	Applicants	Admitted	Applicants	Admitted	Applicants	Admitted
Total	12,763	41%	8442	**44%**	4321	35%

Bias against women in the process, or is there?

Sex Bias in Graduate Admissions: Data from Berkeley (Bickel et al., 1975)

Simpson's paradox: Berkeley graduate admission '73

Department	All		Men		Women	
	Applicants	Admitted	Applicants	Admitted	Applicants	Admitted
A	933	64%	***825***	62%	108	**82%**
B	585	63%	***560***	63%	25	**68%**
C	918	35%	325	**37%**	***593***	34%
D	792	34%	417	33%	375	**35%**
E	584	25%	191	**28%**	***393***	24%
F	714	6%	373	6%	341	**7%**

⚠️ **Aggregation can conceal and contradict actual situation** ⚠️

Sex Bias in Graduate Admissions: Data from Berkeley (Bickel et al., 1975)

Slice-based evaluation

- Evaluate your model on different slices
  - E.g. when working with website traffic data, slice data among:
    - gender
    - mobile vs. desktop
    - browser
    - location
- Check for consistency over time
  - E.g. evaluate your model on data slices from each day

Slice-based evaluation

- Improve model's performance both overall and on critical data
- Help avoid biases
- Even when you don't think slices matter, slicing can:
  - give you confidence on your model (to convince your boss)
  - might reveal non-ML problems

How to identify slices?

- Heuristics
  - Might require subject matter expertise
- Error analysis
  - Patterns among misclassified samples
- Slice finder
  - Exhaustive/beam search
  - Clustering
  - Decision tree

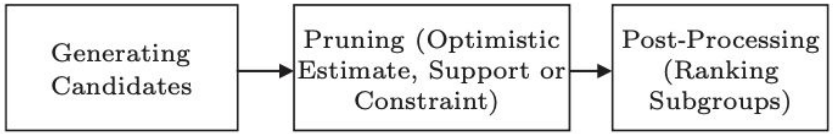


Fig.1. Methodology for subgroup discovery.

Slice finder: Automated data slicing for model validation (Chung et al., 2019)

Subgroup Discovery Algorithms: A Survey and Empirical Evaluation (Sumyea Helal, 2016)

How to identify slices?

⚠️ **Still more of an art than a science** ⚠️

- Heuristics
  - Might require subject matter expertise
- Error analysis
  - Patterns among misclassified samples
- Slice finder
  - Exhaustive/beam search
  - Clustering
  - Decision tree

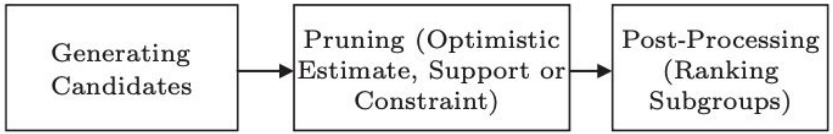


Fig.1. Methodology for subgroup discovery.

Slice finder: Automated data slicing for model validation (Chung et al., 2019)

Subgroup Discovery Algorithms: A Survey and Empirical Evaluation (Sumyea Helal, 2016)

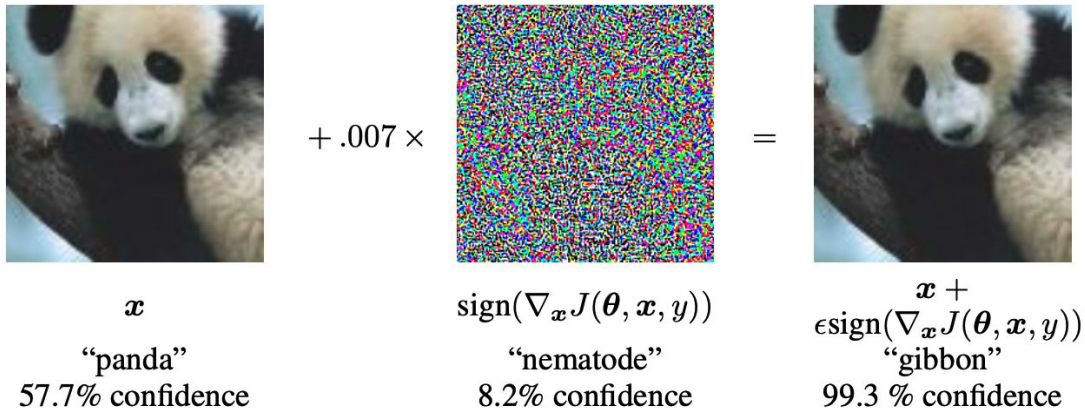
Perturbation evaluation

Perturbation evaluation

- Randomly add small noise to data to see how much outputs change
  - User inputs might contain noise
    - Speech recognition: background noise
    - Object detection: different lighting
    - Text inputs: typos, intentional changes (e.g. looooooooong)
  - Do small changes in inputs lead to big changes in outputs?

Perturbation evaluation

- Randomly add noise to data to see how much outputs change
- The more sensitive the model is to noise:
  - The harder it is to maintain
  - The more vulnerable the model is to adversarial attacks



EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES (Goodfellow et al., 2015)

Invariance tests

- Some input changes shouldn’t lead to changes in outputs
  - Changing race/gender info shouldn’t change predicted approval outcome
  - Changing name shouldn’t affect resume screening results

> The Berkeley study found that both face-to-face and online lenders rejected a total of 1.3 million creditworthy black and Latino applicants between 2008 and 2015. Researchers said they believe the applicants "would have been accepted had the applicant not been in these minority groups." That's because when they used the income and credit scores of the rejected applications but deleted the race identifiers, the mortgage application was accepted.

Disparity in home lending costs minorities millions, researchers find (CBS News, 2019)

Directional expectation tests

- Some changes to inputs should cause predictable changes in outputs
  - E.g. when predicting housing prices:
    - Increasing lot size shouldn’t decrease the predicted price
    - Decreasing square footage shouldn’t increase the predicted price

Ablation study

- Systematically removing parts of your model to see which ones are relevant to the model's performance
  - Features
  - Model components: e.g. regularizations

⚠️ Short-term results ≠ long-term results ⚠️

- Identical results short-term don’t mean identical results long-term
- Superior models now don’t mean superior models a week from now on

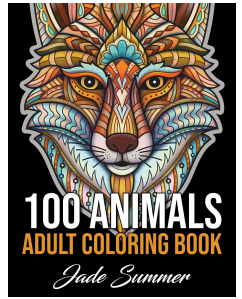
9

6

3

Ishan got a head scratcher!

2



7

5

1

8

10

4

Machine Learning Systems Design

Next class: Versioning and experiment tracking