DATA ANALYTICS USING DEEP LEARNING

GT 8803 // FALL 2018 // JOY ARULRAJ

LECTURE #02: ACCELERATING MACHINE LEARNING INFERENCE WITH PROBABILISTIC PREDICATES

ANNOUNCEMENTS

- Course webpage:
  - https://jarulraj.github.io/data-analytics-course/
- Start thinking about project topics
  - Read assigned papers for inspiration
- No classes next week
- Submit reviews in PDF format
  - GT username as filename

TODAY'S PAPER

- Accelerating Machine Learning Inference with Probabilistic Predicates
  - Query optimization
  - ML inference queries
- Slides based on a presentation by Yao Lu @ SIGMOD 2018

TODAY'S PAPER

MACHINE TRANSLATION

STORAGE MANAGEMENT

QUERY PROCESSING

HARDWARE ACCELERATION

LAYERS OF A DATA ANALYTICS SYSTEM

TODAY'S AGENDA

- Problem Overview
- Key Idea
- Technical Details
- Experiments
- Discussion

ML INFERENCE ON BIG-DATA PLATFORMS

- SQL + user-defined functions
  - On unstructured data blobs
  - Videos, Images, and Unstructured text

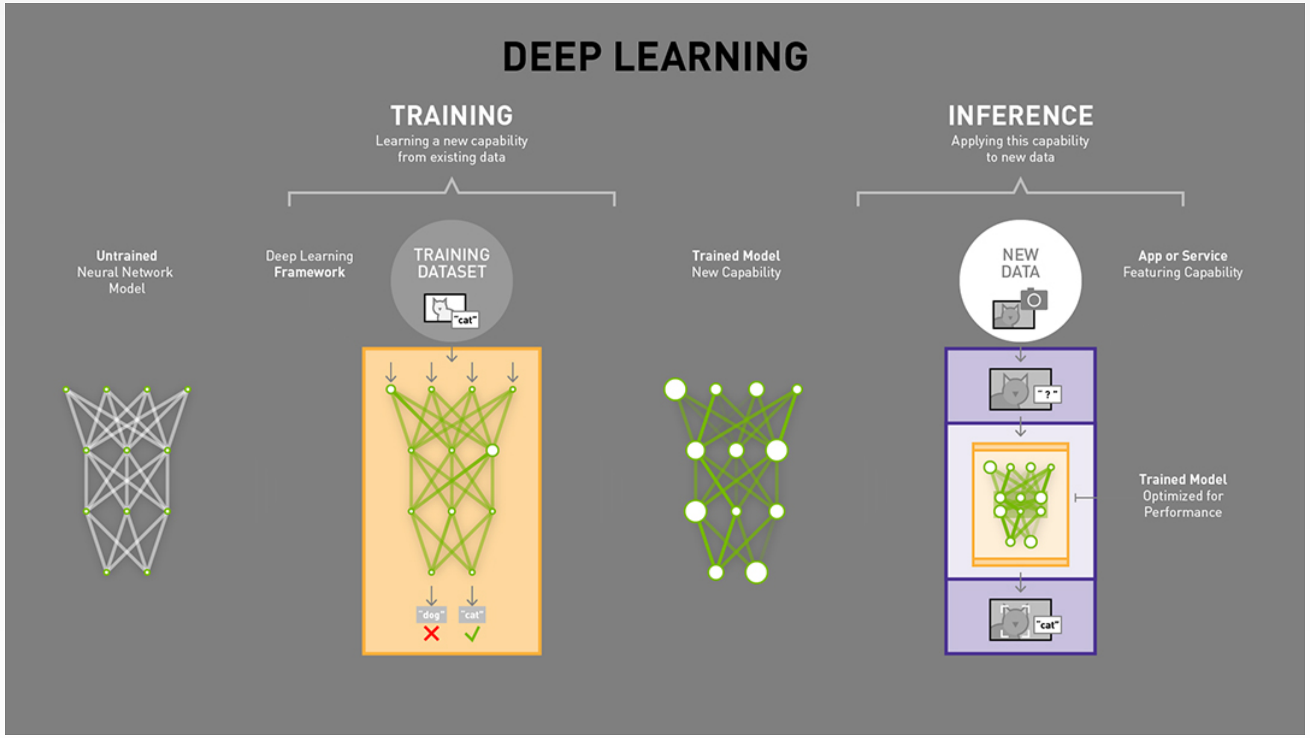
APACHE PIG

Spark SQL

HIVE

ML INFERENCE

Untrained neural network → Training → Inference on new data



Source: What's the Difference Between Deep Learning Training and Inference?, Michael Copeland, August 2016, NVIDIA Blog

ML INFERENCE QUERY EXAMPLE

- Find images of oranges

Images → **UDF_YOLOv2** → σ_{orange} → Result

→ **UDF_YOLOv2** → has person?

→ has bear?

→ has orange?

→ …

ML INFERENCE QUERY EXAMPLE

- Inference takes time
  - Even when the predicate has low selectivity
  - Perhaps only 1-in-100 images have oranges
- Reason
  - Every image has to be processed by all the UDFs

$$\text{Images} \rightarrow \textbf{UDF\_YOLOv2} \rightarrow \sigma_{\text{orange}} \rightarrow \text{Result}$$

PROBLEM OVERVIEW

- How can we accelerate such inference queries?

$$\text{Images} \rightarrow \textbf{UDF\_YOLOv2} \rightarrow \sigma_{\text{orange}} \rightarrow \text{Result}$$

SOLUTION #1: PREDICATE PUSHDOWN

- Traditional query optimization technique
  - Move filtering of data as close to the source as possible to avoid loading unnecessary data into higher-level operators

    Join Tables A, B → **Predicates on A, B** → Result
- Cannot push predicates below the UDF
  - No "contains orange" column exists
  - Need to construct it using UDF

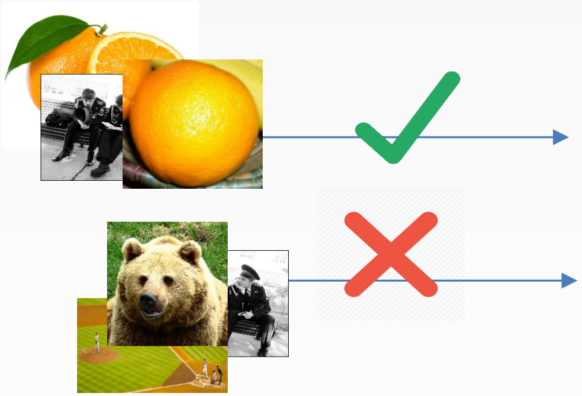
SOLUTION #2: PRE-COMPUTING

- Pre-computing all possible columns
  - High cost since too many UDFs & query predicates
- Not a good fit for <u>ad-hoc queries</u>
  - Since only certain columns corresponding to certain images will be required
- Not a good fit for <u>online queries</u>
  - Need to do inference on live data

KEY IDEA

- Accelerate queries by <u>early filtering</u>

Images $\rightarrow \boxed{\text{Filter}} \rightarrow$ **UDF_YOLOv2** $\rightarrow \sigma_{\text{orange}} \rightarrow$ Result

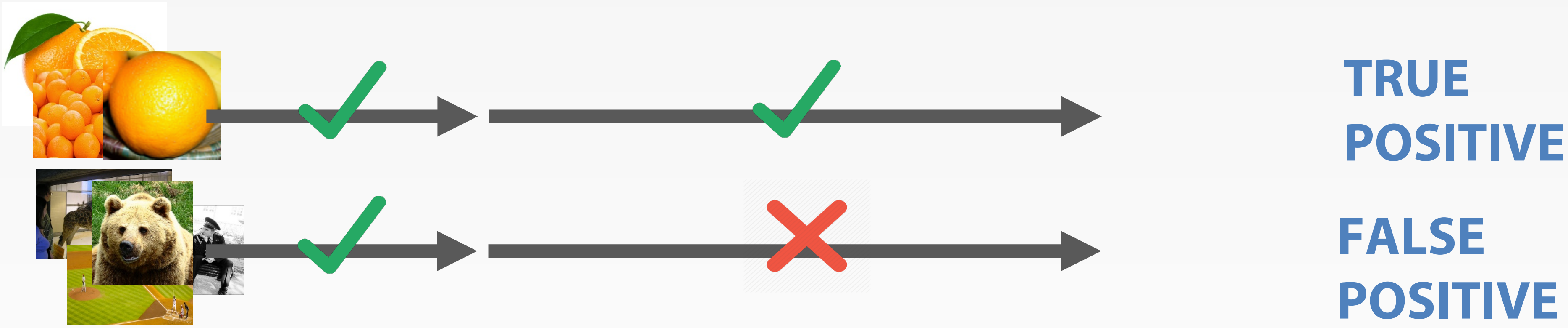


KEY IDEA

- Early filter constraints
- Performance
  - Utility of data reduction >> Execution cost of early filter
- Accuracy
  - Early filtering should not increase false negatives

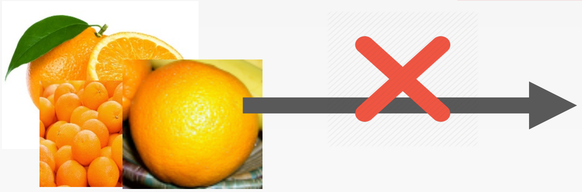
EARLY FILTERING

Images → $\boxed{\text{Filter}}$ → **UDF_YOLOv2** → σ_{orange} → Result



EARLY FILTERING

Images $\rightarrow$ $\boxed{\text{Filter}}$ $\rightarrow$ **UDF_YOLOv2** $\rightarrow$ σ_{orange} $\rightarrow$ Result



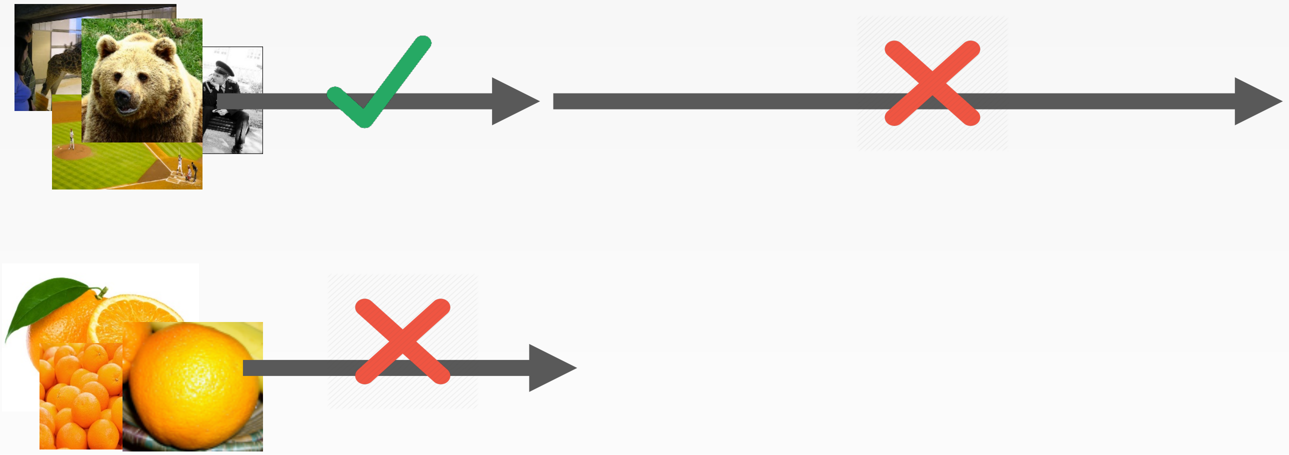
DATA REDUCTION

FALSE NEGATIVE

TRUE NEGATIVE

EARLY FILTERING

Images → $\boxed{\text{Filter}}$ → **UDF_YOLOv2** → σ_{orange} → Result



FALSE POSITIVE –

FALSE NEGATIVE ↑

PROBABILISTIC EARLY FILTERING

- Unlike queries on relational data
  - ML applications have in-built tolerance for errors
  - ML UDFs generate false positives & false negatives
- So, filters can also be probabilistic!
  - Reducing accuracy can increase data reduction rate

PROBABILISTIC PREDICATES

- Goal: query speedup + desired accuracy
  - Train binary classifiers
  - Group input blobs into two categories
    - Blobs that disagree with the query predicate
    - Blobs that may agree with the query predicate
- Classifiers are called probabilistic predicates
  - <Data reduction rate, Execution cost, Accuracy>

PROBABILISTIC PREDICATES (PPs)

$$\text{Images} \rightarrow \underset{\boxed{\text{50-1K fps}}}{PP_{\text{orange}}} \rightarrow \underset{\boxed{\text{10-30 fps}}}{\textbf{UDF\_YOLOv2}} \rightarrow \sigma_{\text{orange}} \rightarrow \text{Result}$$

- Low filter execution cost
  - High data reduction
  - Minimal impact on accuracy

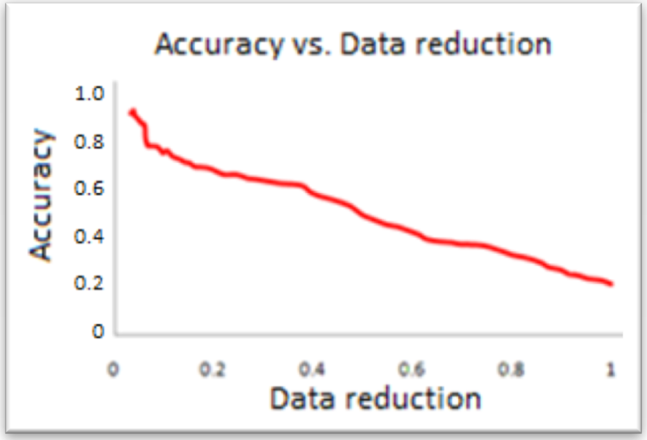
PROBABILISTIC PREDICATES (PPs)

$$\text{Images} \rightarrow PP_{orange} \rightarrow \textbf{UDF\_YOLOv2} \rightarrow \sigma_{orange} \rightarrow \text{Result}$$

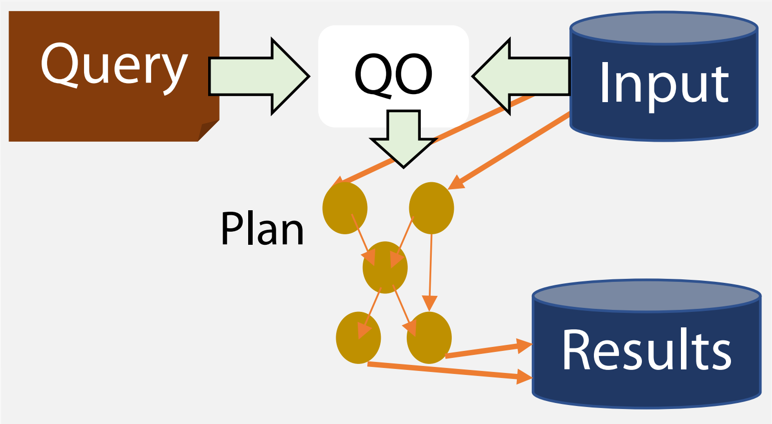
50-1K fps

10-30 fps

- Apply PP directly on raw blob
  - 5-1000x faster than UDF
  - Accuracy vs data-reduction curve

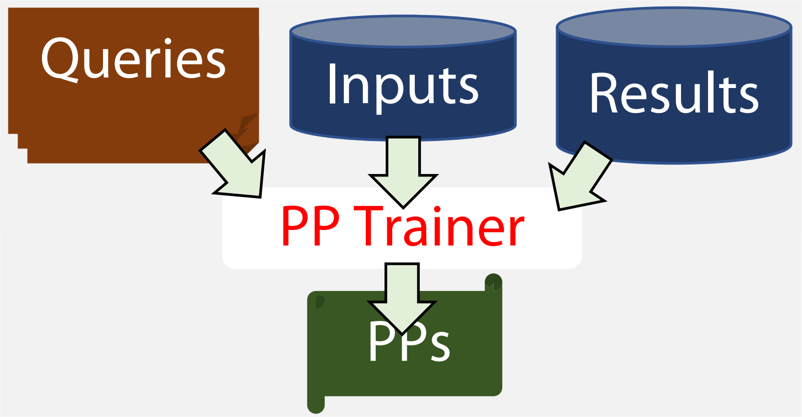


SYSTEM WORKFLOW: BASELINE SYSTEM W/O PPs



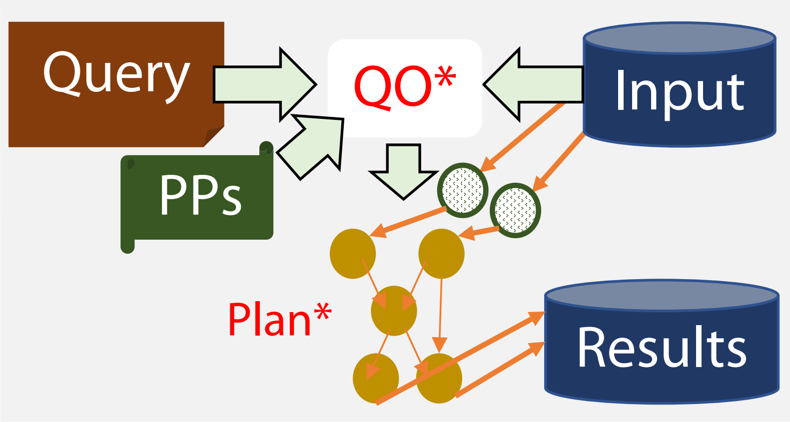
a) Baseline System w/o PPs

SYSTEM WORKFLOW: CONSTRUCTING PPs



b) Constructing PPs

SYSTEM WORKFLOW: FULL SYSTEM W/ PPs



c) Full system w/ PPs

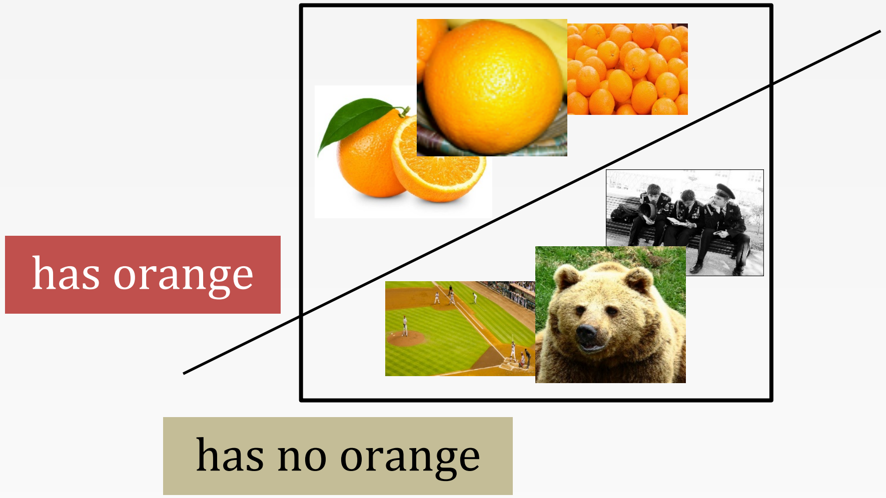
CHALLENGES

- How to build useful PPs
  - Good trade-off between data reduction rate, cost, and accuracy
- Supporting complex query predicates?
  - Using simple PPs for ad-hoc queries

PART-1: BUILDING USEFUL PPs

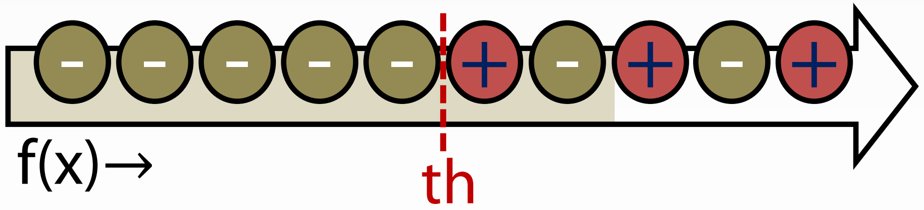
- Probabilistic predicate
  - Can be thought of as a decision boundary separating two classes
  - Any classifier that can identify inputs far away from the decision boundary is an useful PP
- Use different techniques for building PPs
  - Support vector machines (SVMs)
  - Deep neural networks, etc.

SIMPLE PP USING LINEAR CLASSIFIER



$PP_{orange}: f(x) = w \cdot x + b$

Setting accuracy/ reduction tradeoff threshold (th)

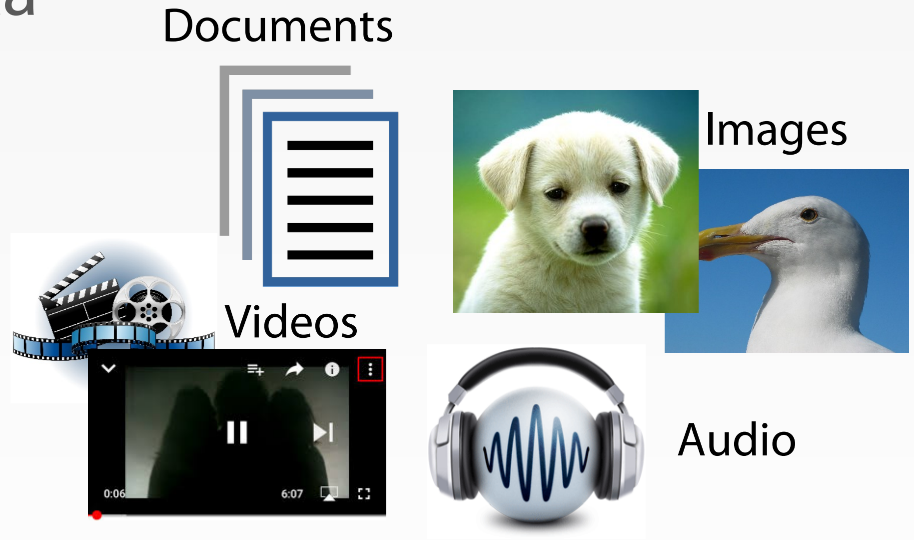


PP discards $f(x) \leq th$

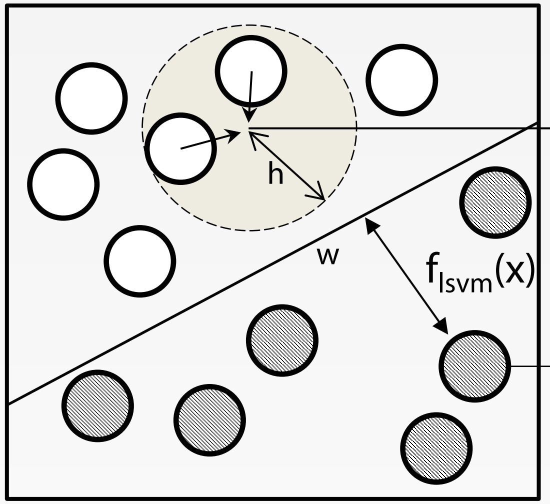
Accuracy = 3/3, Reduction = 5/10
Accuracy = 2/3, Reduction = 7/10

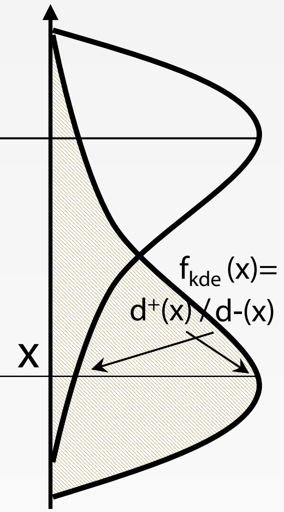
PPs FOR ARBITRARY DATA BLOBS

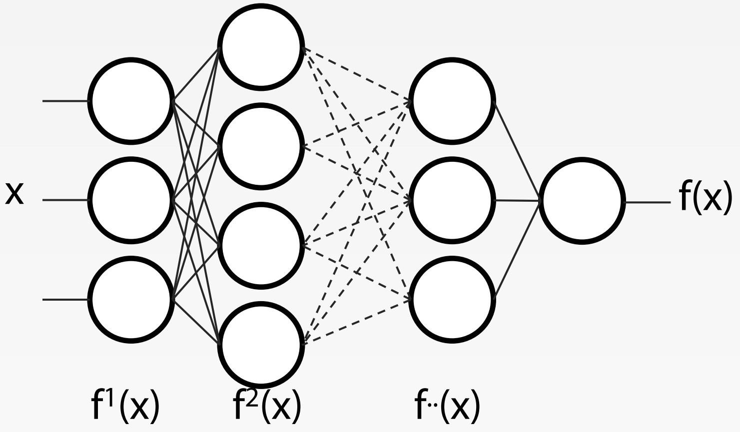
- Input blob characteristics
  - Linearly separable or not
  - High dimensional data
  - Sparse or dense



PPs FOR ARBITRARY DATA BLOBS







More ?

Linear SVM:

$f(x) = w \cdot x + b < th$

Linearly-separable data

Kernel Density Estimator:

$f(x) = kde^+(x) / kde^-(x) < th$

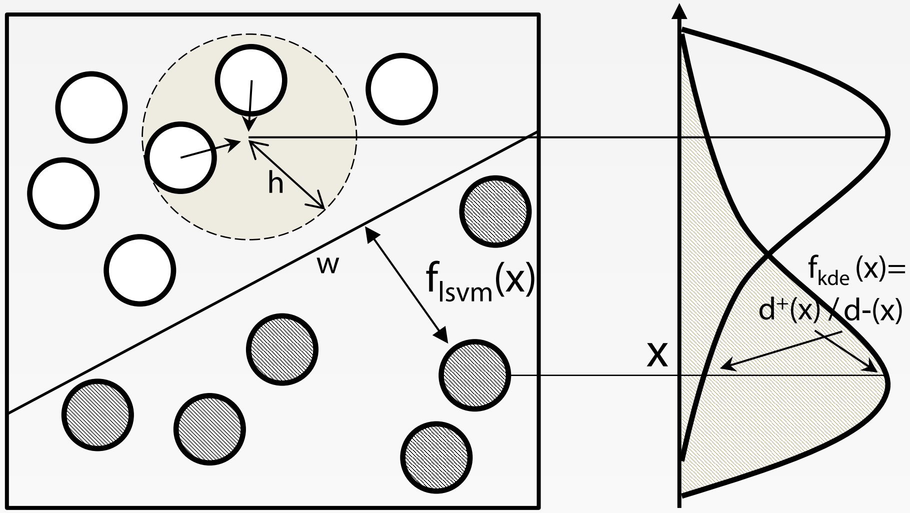
Nonlinearly-separable data

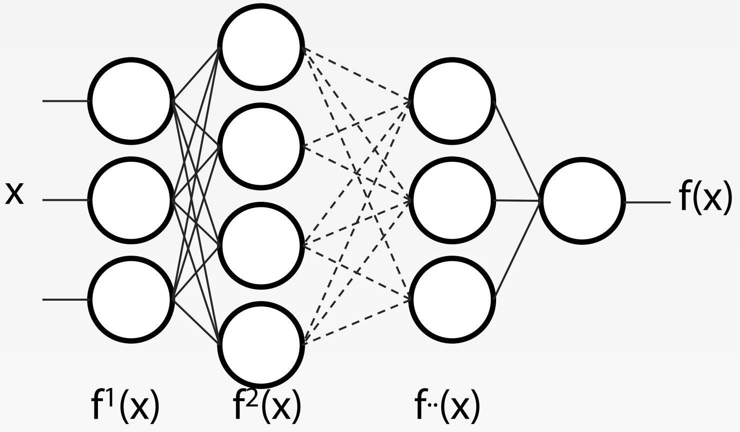
Shallow DNN:

$f^i(x) = g_i(W_i \cdot f^{i-1}(x) + b_i) < th$

Random forest etc.
any function
that fits $f(x) < th$

PPs FOR ARBITRARY DATA BLOBS





More ?

Linear SVM:
$f(\mathrm{x}) = \mathrm{w} \cdot \mathrm{x} + \mathrm{b} < \mathrm{th}$

Kernel Density Estimator:
$f(\mathrm{x}) = \mathrm{kde}^+(\mathrm{x}) \,/\, \mathrm{kde}^-(\mathrm{x}) < th$

Shallow DNN:
$f^i(\mathrm{x}) = g_i(W_i \cdot f^{i-1}(\mathrm{x}) + b_i) < th$

Random forest etc.
any function
that fits $f(\mathrm{x}) < \mathrm{th}$

\+ Dimension Reduction
Example: Feature Hashing,
Principal Component Analysis

\+ Model Selection
Select the best model

MODEL SELECTION

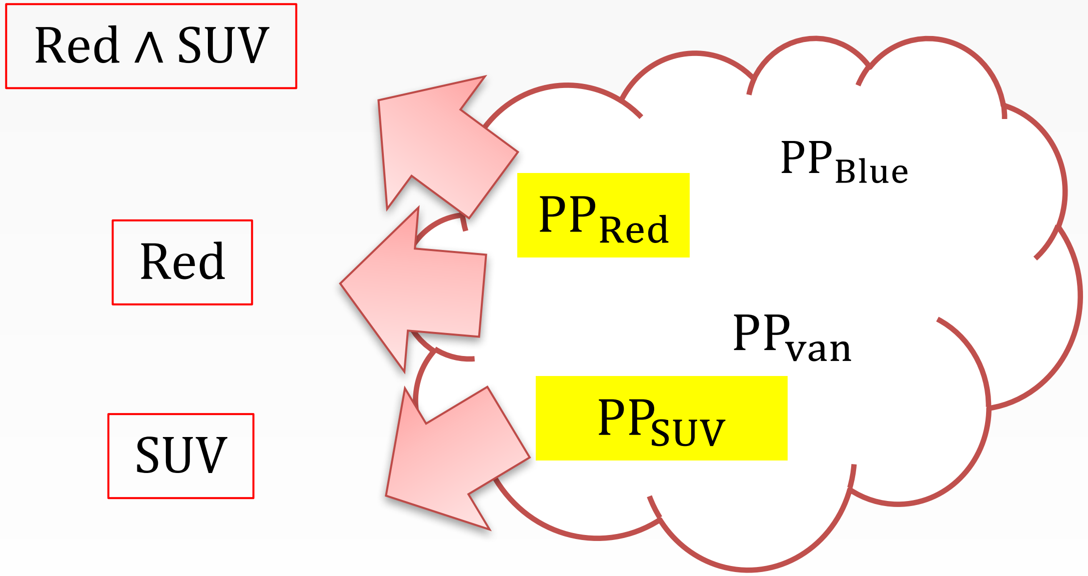
- Given different PP methods, select best PP that maximizes data reduction rate
  - Test PP on a small sample of data
- Model selection insights
  - Input dataset determines PP selection
  - Given a blob type, same PP applies for different predicates & accuracy thresholds

PART-2: SUPPORTING COMPLEX PREDICATES

- Queries with complex or new predicates
  - Large space of possible predicates
  - Costly to train/store a PP for each predicate
  - PPs for complex predicates do not generalize
- Pick best PP combination
  - Query optimization problem
  - Inputs: available PPs, predicate, target accuracy
  - Goal: find PP combination ⇒ max reduction / cost

COMBINING PPs USING QUERY OPTIMIZATION (QO)

- Solution
  - Build PPs for simple predicates
  - Use QO to assemble PP combinations



PPs trained << # predicates

QUERY OPTIMIZATION OVER PPs

- Predicate: $\left(\sigma_{\text{orange}} \vee \sigma_{\text{banana}}\right) \wedge \sigma_{\text{cat}} \wedge \sigma_{\text{dog}}$

- Conventional query optimization technique
  - Ordering predicates by data reduction/cost
  - Do not focus on combining predicates

STEP #1: SELECT CANDIDATE PP EXPRESSIONS

- Explore <u>necessary</u> conditions to satisfy predicate for improving speedup

Necessary conds.

$(\sigma_{\text{orange}} \vee \sigma_{\text{banana}}) \wedge \sigma_{\text{cat}} \wedge \sigma_{\text{dog}}$
$\Rightarrow (\text{PP}_{\text{dog}} \wedge \text{PP}_{\text{cat}}) \wedge (\text{PP}_{\text{orange}} \vee \text{PP}_{\text{banana}})$
$\Rightarrow \text{PP}_{\text{dog}} \wedge \text{PP}_{\text{cat}}$
$\Rightarrow \text{PP}_{\text{orange}} \vee \text{PP}_{\text{banana}}$
$\Rightarrow \text{PP}_{\text{cat}}$
$\Rightarrow \text{PP}_{\text{dog}}$

exponentially many choices

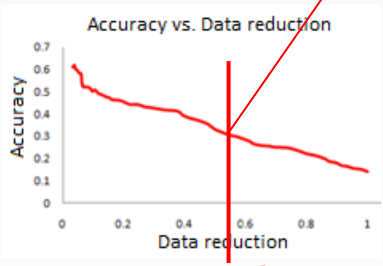
Available: $\text{PP}_{\text{cat}}, \text{PP}_{\text{dog}}, \text{PP}_{\text{orange}}, \text{PP}_{\text{banana}}$

Greedily find a PP combination that has the best reduction / cost

STEP #2: ESTIMATE DATA REDUCTION

- Estimate reduction and cost for every PP combination (trivial for one PP)

$PP_{cat} \wedge PP_{dog}$



th_{cat}

th_{dog}

th $\Leftrightarrow$ a: Lookup table

$$
\begin{aligned}
a &= a_1 * a_2 \\
r[a] &= r_1[a_1] + r_2[a_2] - r_1[a_1] * r_2[a_2] \\
c[a] &= \min(c_1 + (1 - r_1[a_1]) * c_2,\ \ c_2 + (1 - r_2[a_2]) * c_1)
\end{aligned}
$$

Costing rule for $p_1 \wedge p_2$

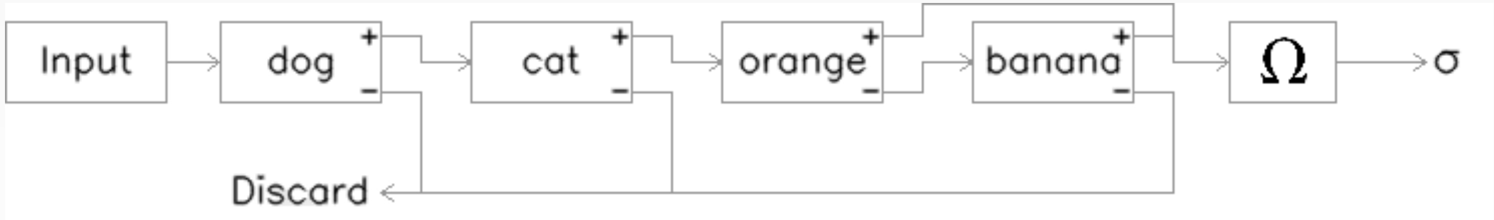
$$\max_{th_{cat}, th_{dog}} \frac{r_{cat \wedge dog}}{c_{cat \wedge dog}}, \ \text{s.t.}\ a_{cat \wedge dog} \geq th$$

Solve using dynamic programming

#3: ADD PPs TO QUERY PLAN

- Adds PPs to the query plan
  - Based on desired accuracy and data reduction constraints

$$(\mathbf{PP_{dog}} \wedge \mathbf{PP_{cat}}) \quad \wedge \quad (\mathbf{PP_{orange}} \vee \mathbf{PP_{banana}})$$



RELATED WORK: MODEL CASCADES

- Cascade of classifiers (Viola et al., 2001)
  - More efficient but inaccurate classifier can be used in front of expensive classifier to lower overall cost
  - Typical cascades use classifiers with equivalent functionality and accept and reject anywhere in the pipeline
  - In contrast, PPs are not equivalent to all UDFs that they bypass and only reject irrelevant blobs

RELATED WORK: EXPLOITING CORRELATIONS

- To accelerate UDFs (Joglekar et al., 2015)
  - Correlations between input columns & UDFs
  - Learns a probabilistic selection method that accepts or rejects inputs without evaluating UDFs
- PPs do not accept blobs early and extend beyond selection queries

RELATED WORK: NOSCOPE

- NoScope (Kang et al., 2018)
  - Uses specialized DNN + video-specific filtering techniques to speed up object detection on videos
  - Requires per query DNN training
- PPs have broader applicability
  - QO explores combinations of simple PPs
  - Avoids per query PP training

EXPERIMENTS

- Two key questions
  - Validating the utility of individual PPs
  - End-to-end system evaluation

- Datasets
  - Document categorization
  - Image labeling
  - Video activity recognition
  - Traffic surveillance video analytics

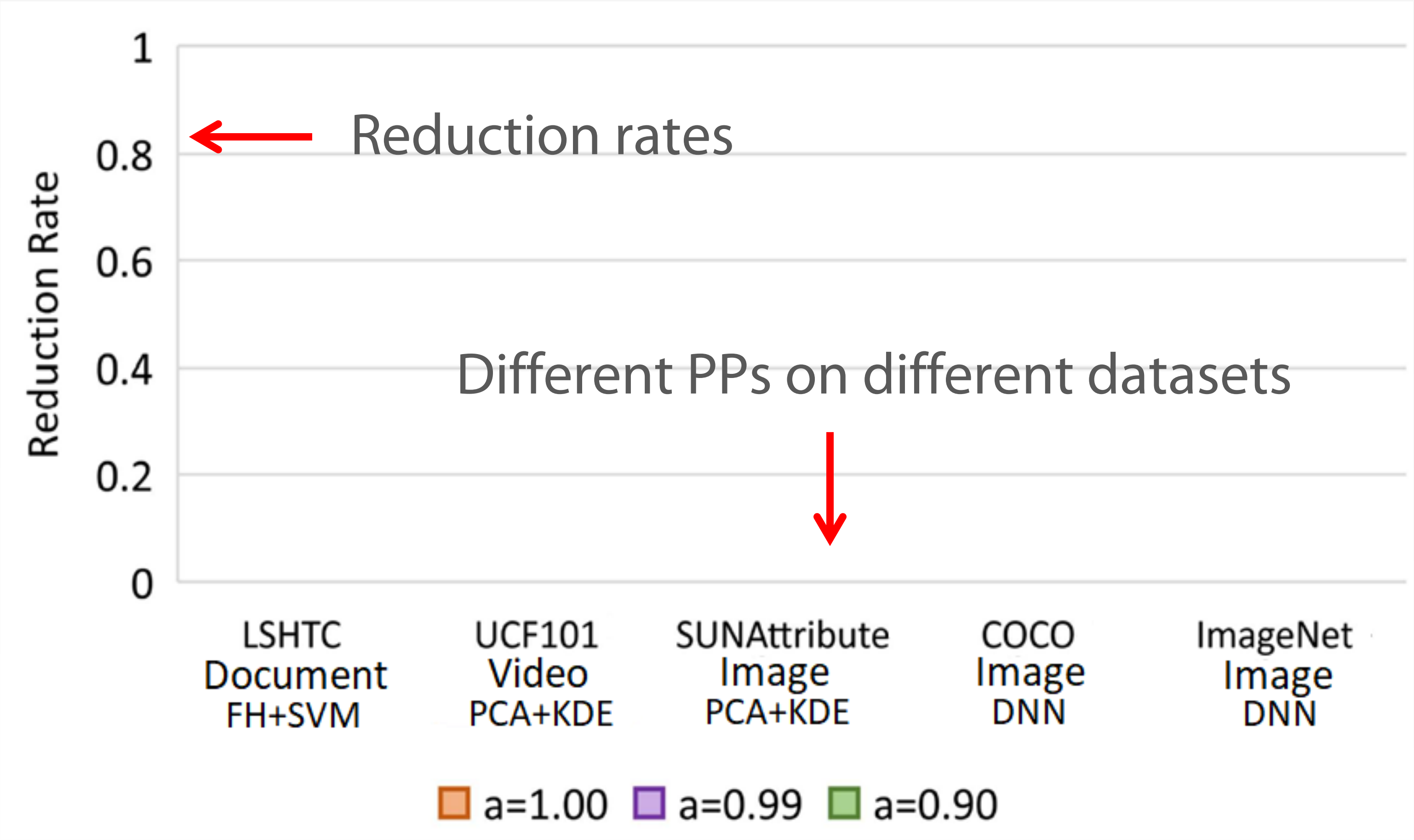
DATASETS

COCO & ImageNet & SUNAttribute
Image Datasets
Predicate: Has “Dog”/”Bicycle”/..
>100 categories

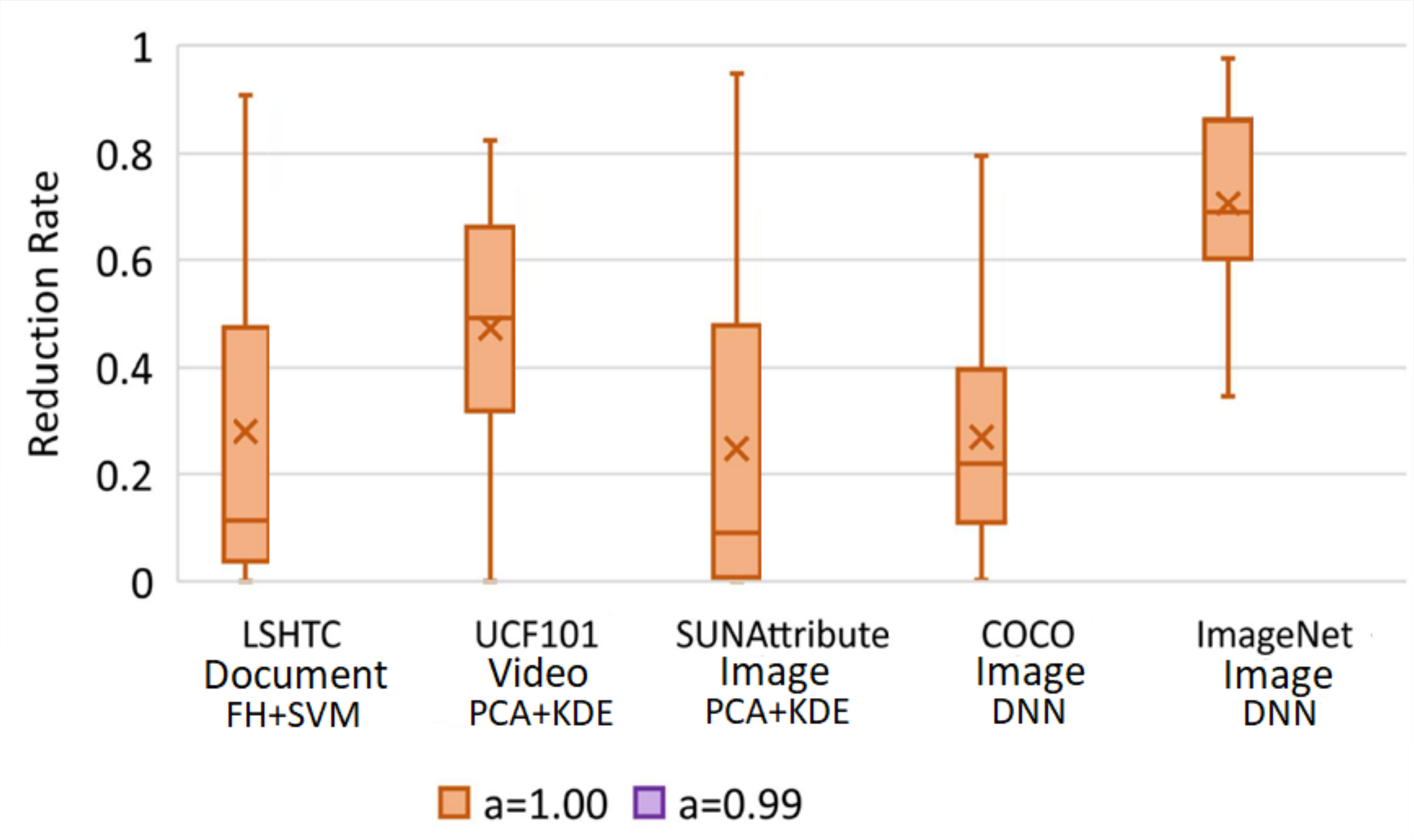
UCF101 Video Activity Recognition Dataset
Predicate: PlayingGuitar / Biking / …
101 video actions

LSHTC Document Classification Dataset
Predicate: 2.4M documents, 400K categories

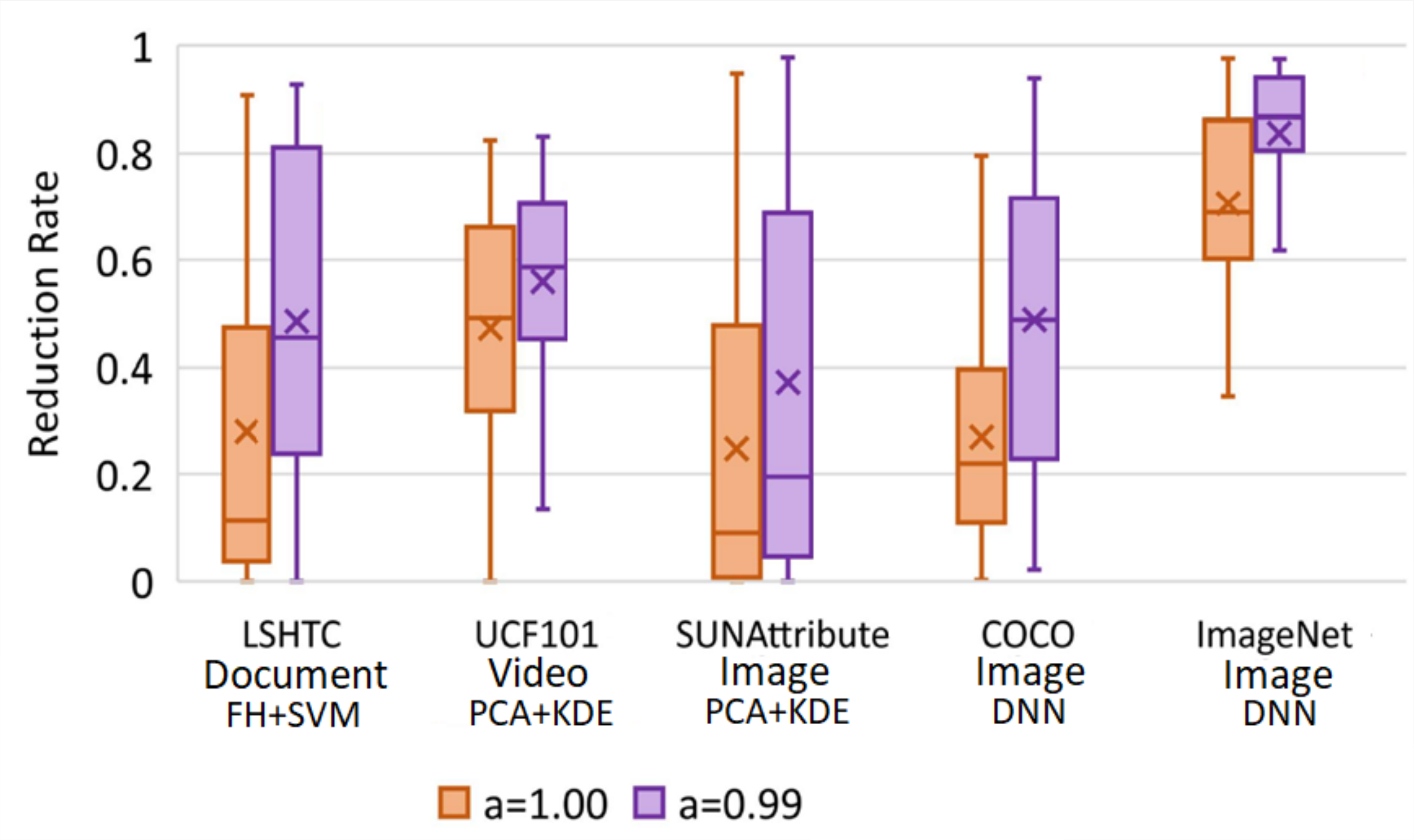
DATA REDUCTION RATES ACHIEVED BY PPs



DATA REDUCTION RATES ACHIEVED BY PPs



DATA REDUCTION RATES ACHIEVED BY PPs



MODEL SELECTION

Dataset	Approach	Avg. data reduction $\overline{r}$ for accuracy a		
		$\overline{r[1]}$	$\overline{r[0.99]}$	$\overline{r[0.9]}$
UCF101	**PCA+KDE**	**0.47**	**0.56**	**0.64**
	PCA + SVM	0.35	0.45	0.54
	Raw + SVM	0.35	0.47	0.59
ImageNet	**DNN**	**0.71**	**0.84**	**0.96**
	SVM		0.39	
	DNN trained on COCO	0.25	0.49	0.82

QUERY OPTIMIZATION OVER PPs

- Does QO choose appropriate PP combination for complex predicates?
- Experiment setup
  - DETRAC Traffic Surveillance Video Dataset
  - Predicate columns:
  - VehicleColor, VehicleType, Speed, Direction
  - Number of possible predicates $>100^5$

QUERY OPTIMIZATION OVER PPs

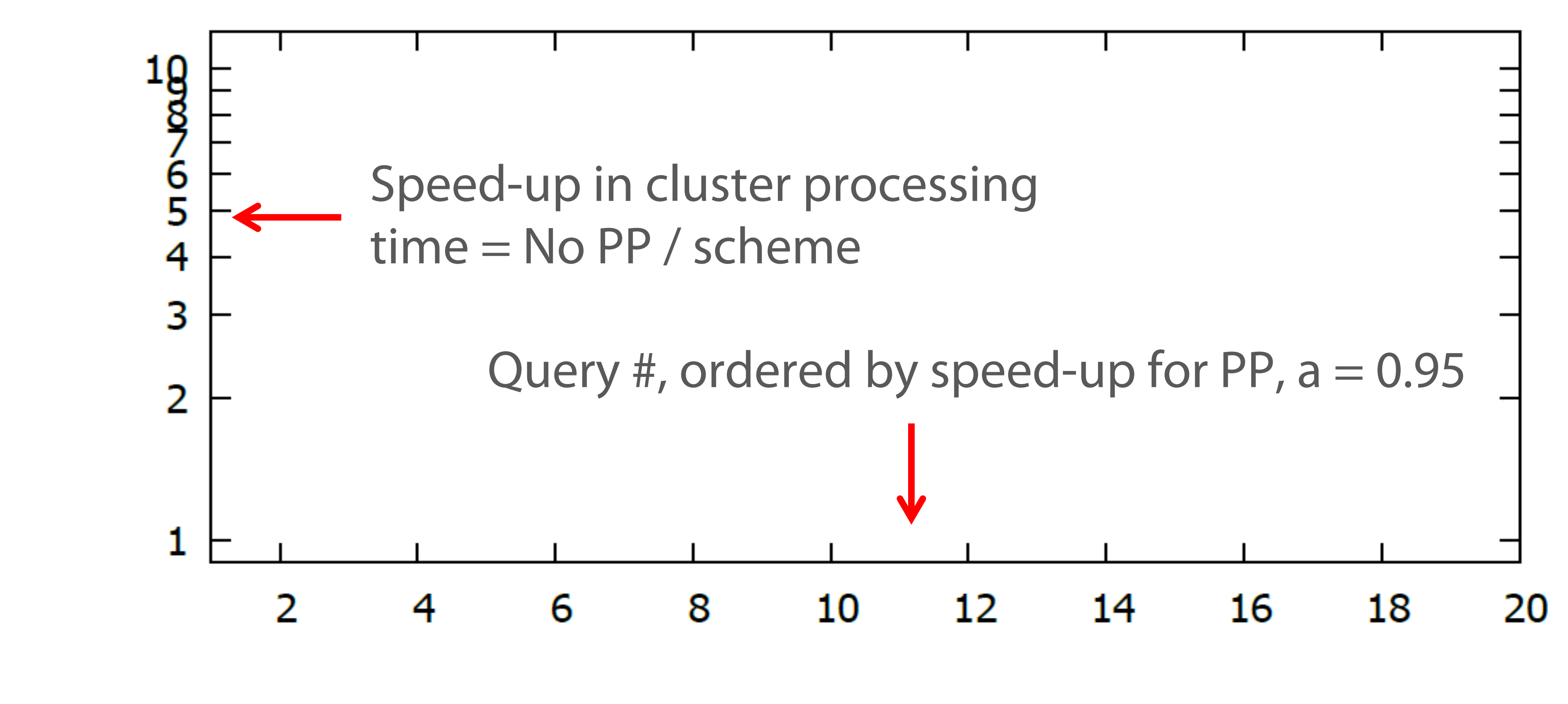
- Experiment setup
  - Number of PPs trained = 32
  - Per categorical column, equality
  (e.g., VehicleColor = Red, VehicleType = SUV)
  - Per range column, multiple inequalities
  (e.g., Speed >65, >75…)

QUERY OPTIMIZATION OVER PPs

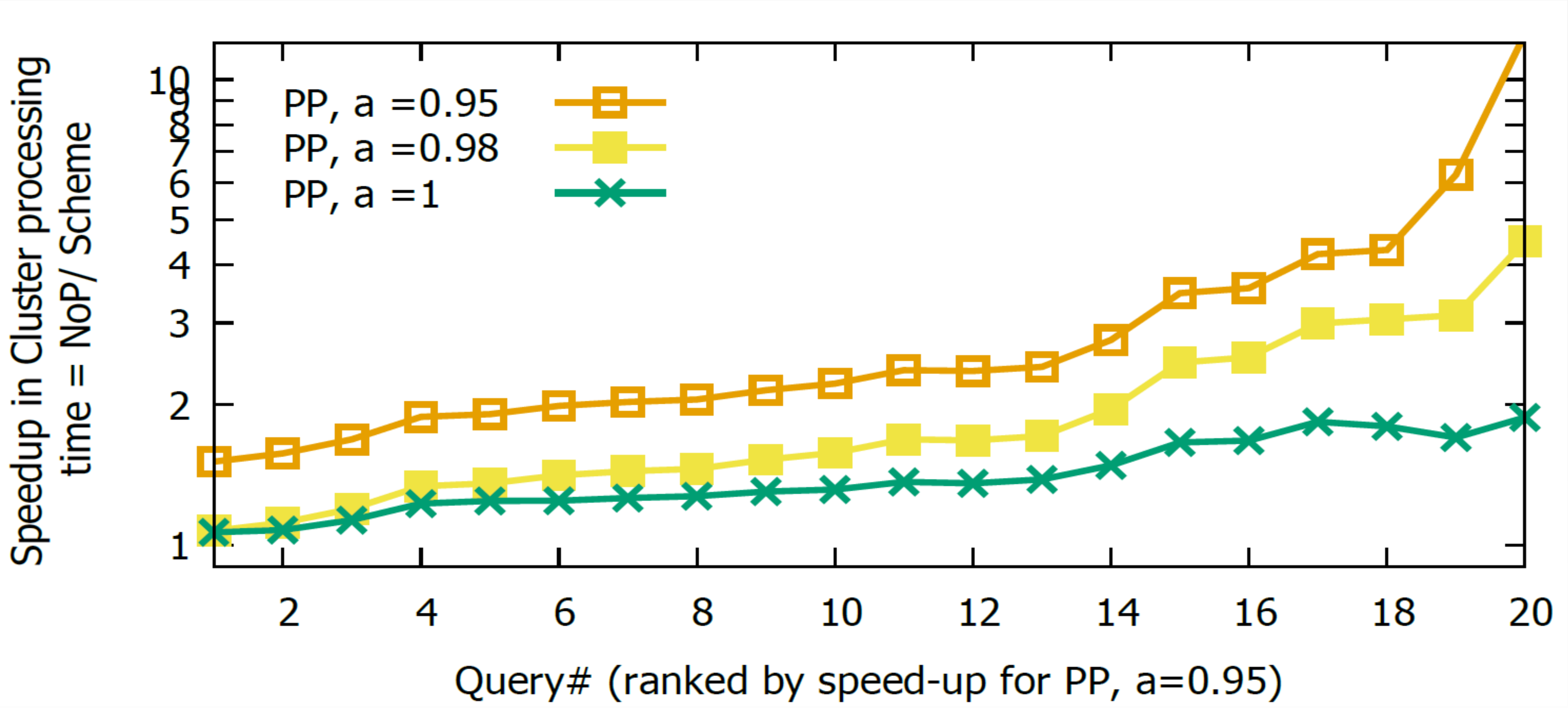
- Complex query predicate example:
  - speed>60 $\wedge$ speed<65 $\wedge$ color=white $\wedge$ type $\in$ {SUV, van}

CANDIDATE PP PLAN	EST. DATA REDUCTION
$PP_{speed>60} \wedge PP_{speed<65} \wedge PP_{\neg sedan} \wedge PP_{\neg truck} \wedge PP_{white}$	0.77 (picked)
$PP_{speed>50} \wedge PP_{speed<70}$	0.43
$PP_{speed>60} \wedge PP_{speed<65} \wedge PP_{\neg sedan}$	0.52
… 216 such expressions	

RESOURCE USAGE IMPROVEMENT



RESOURCE USAGE IMPROVEMENT



CONCLUSION

- Leverage PPs to accelerated ML inference
  - How to construct useful PPs?
  - How to combine PPs to handle complex predicates?
  - Results show utility across varied ML tasks

DISCUSSION

- Domain-agnostic idea
  - Does not focus on a specific blob type
  - Does not focus on a specific ML technique

STRUCTURED + UNSTRUCTURED DATA

- Processing structured + un-structured data
  - Use PPs to accelerate filtering of unstructured data
  - Use the output of UDFs processing filtered unstructured data as structured data
  - Traditional QO techniques for structured data

LEARNING FROM DATA

- Develop algorithms and ML models to learn the patterns from data
  - Data skew
  - Data correlations
  - Use this information during query optimization

QUERY PREDICATE CONSTRUCTION

- Guidance to users for constructing queries around PPs
  - Minor query predicate modifications can have major performance impact
  - Using physical costs during optimization

COMPLEX PREDICATES

- Temporal and causal links in data
  - Nested predicates?
  - More complex predicates?

NATURAL LANGUAGE PROCESSING

- Natural language processing pipelines
  - Leverage classifiers trained on note embeddings, and/or the semantic hierarchies

NEXT CLASS

- Sep 5 (Wed)
  - BlazeIt: Fast Exploratory Video Queries using Neural Networks
  - Video analytics using DNNs