Coresets for Data-efficient Training of Machine Learning Models

Baharan Mirzasoleiman*, Jeff Bilmes, Jure Leskovec***

UCLA *

LELAND STANFORD JUNIOR UNIVERSITY ORGANIZED 1891 *

** UNIVERSITY of WASHINGTON

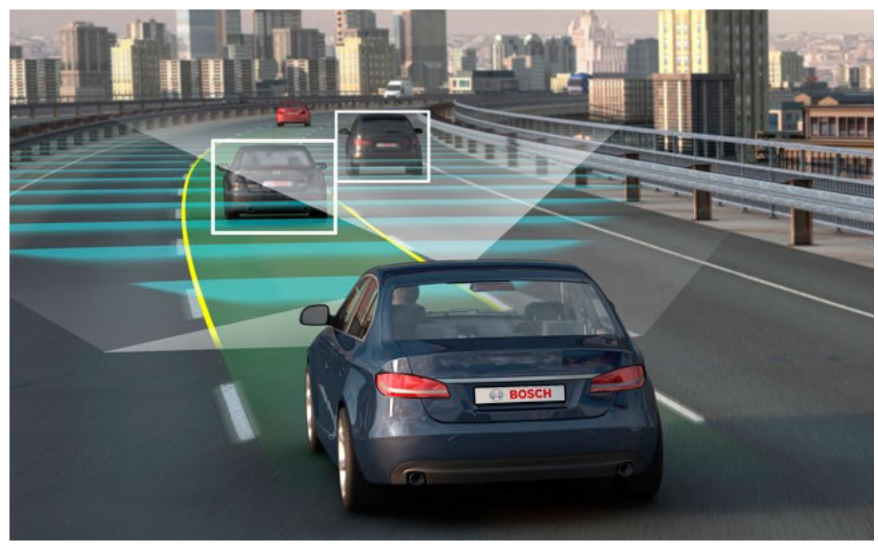
Machine Learning Becomes Mainstream

Personalized medicine

Robotics

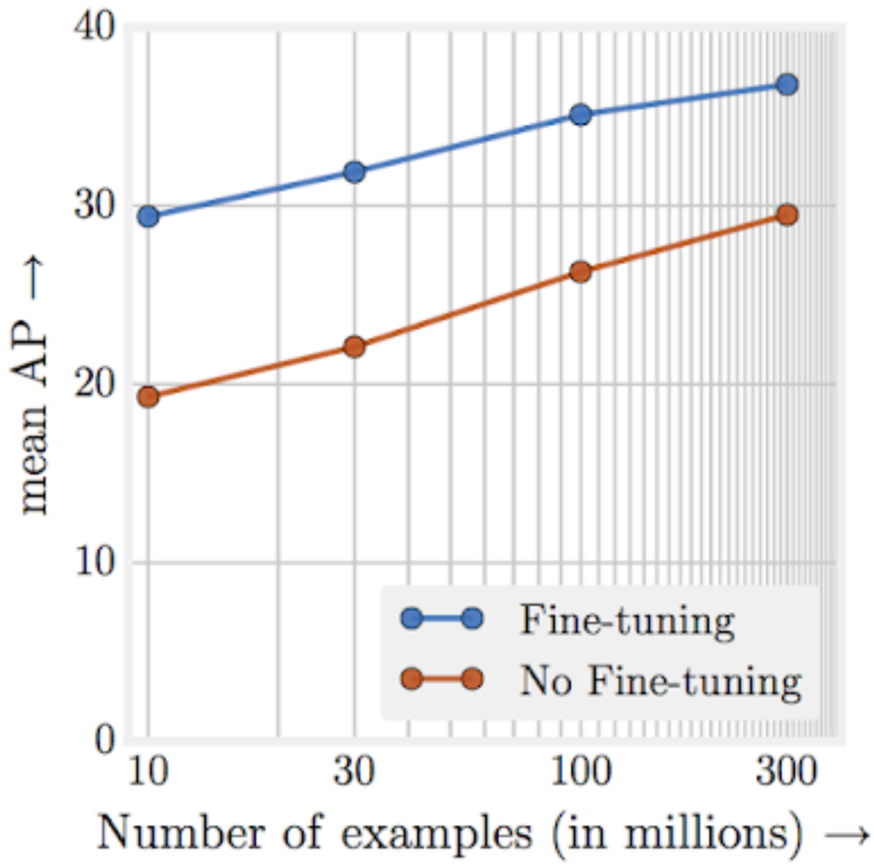
Finance

Autonomous cars



Data is the Fuel for Machine Learning

Example: object detection



Object detection performance in mAP@[.5,.95] on COCO minival [Google AI]

Problem 1: Training on Large Data is Expensive

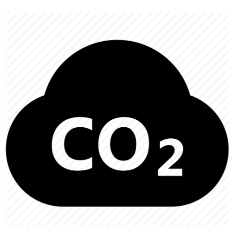
Example: training a **single** deep model for NLP (with NAS)

[SGM'19]

3.2M

11.4 days

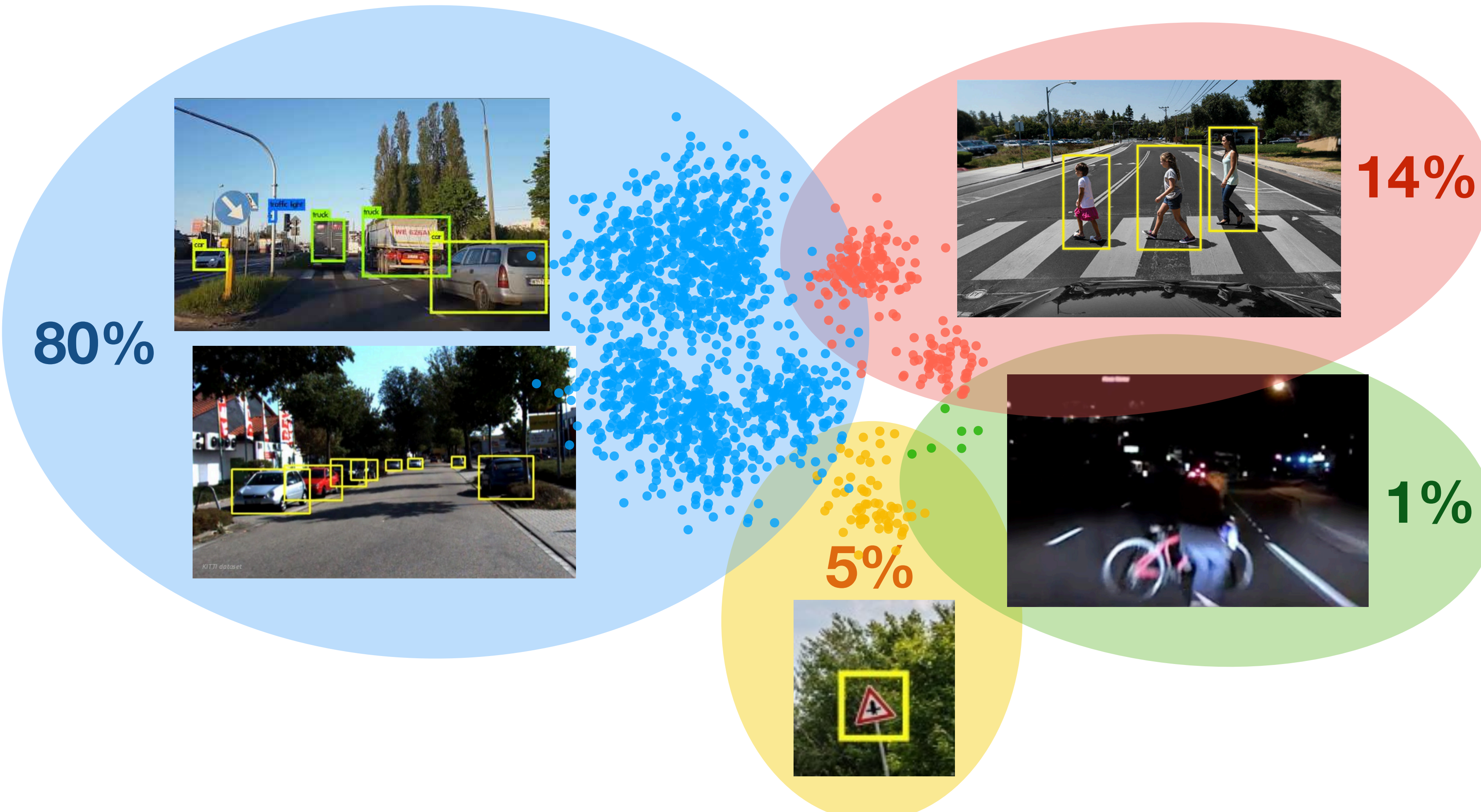
5.3X the yearly **energy** consumption of the average American



5x a lifetime of a car **CO2**

Problem 2: What We Care About is Underrepresented

Example: self driving data



How can we find the “right” data for efficient machine learning?

Setting: Training Machine Learning Models

Often reduces to minimizing a regularized empirical risk function

Feature **Label**

Training data volume: $\{(x_i, y_i), i \in V\}$

Regularizer

$$w_* \in \arg\min_{w \in \mathcal{W}} f(w), \quad f(w) = \sum_{i \in V} f_i(w) + r(w), \quad f_i(w) = l(w, (x_i, y_i))$$

Loss function associated with training example $i \in V$

Examples:

- **Convex $f(w)$**: Linear regression, logistic regression, ridge regression, regularized support vector machines (SVM)
- **Non-convex $f(w)$**: Neural networks

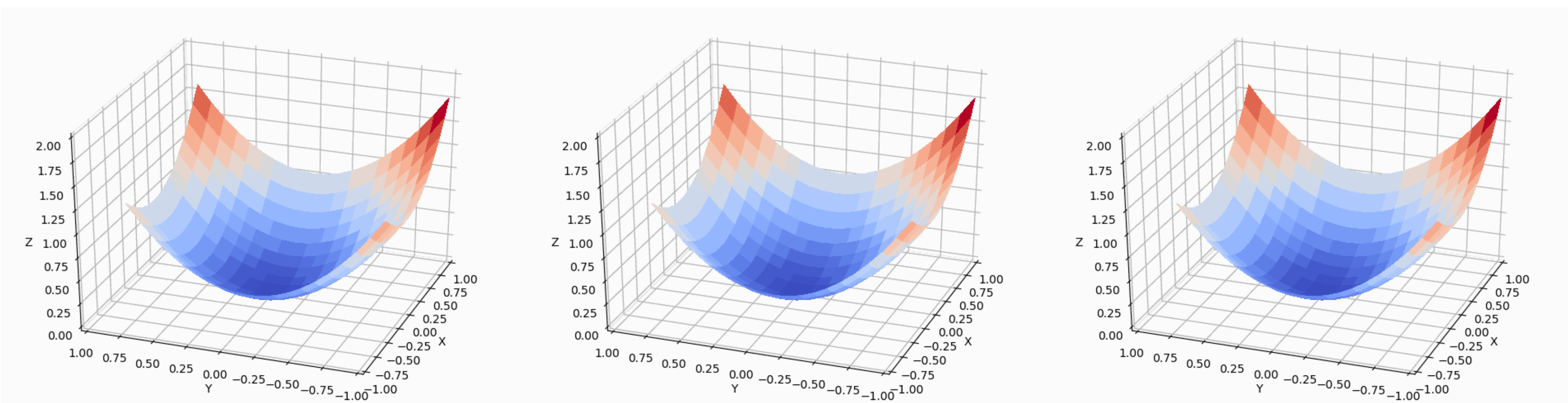
Setting: Training Machine Learning Models

Incremental gradient methods are used to train on large data

- Sequentially step along the gradient of functions f_i

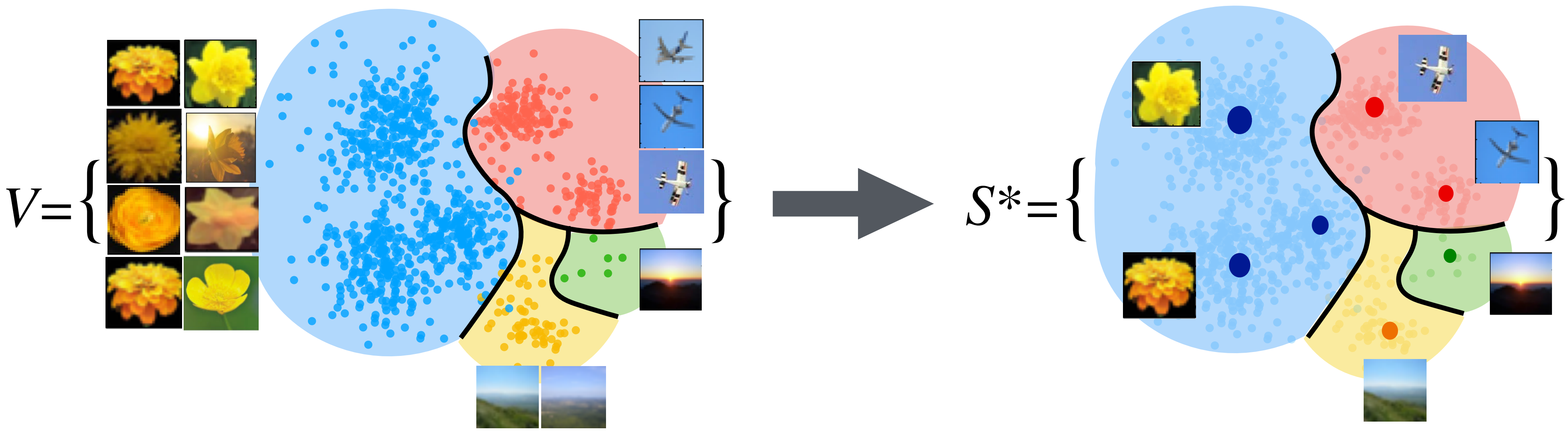
$$w_i^k = w_{i-1}^k - \alpha_k \nabla f_i(w_{i-1})$$

- Consider every $\nabla f_i(\,.\,)$ as an unbiased estimate of $\nabla f(\,.\,) = \sum_{i \in V} \nabla f_i(\,.\,)$
- Therefore, they are **slow** to converge



Problem: How to Find the "Right" Data for Machine Learning?

- The most informative subset $S^* = \arg\max_{S \subseteq V} F(S), \quad \text{s.t.} \quad |S| \leq k$
- **What is a good choice for $F(S)$?**

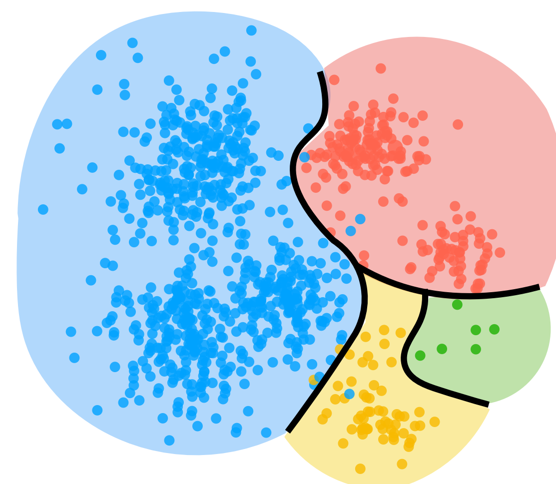


- If we can find S^*, we get a $|V|/|S^*|$ speedup by only training on S^*

Finding S^* is Challenging

1. How to chose an informative subset for training?
   - Points close to decision boundary vs. a diverse subset?

2. Finding S^* must be fast
   - Otherwise we don't get any speedup



3. We also need to decide on the step-sizes

4. We need theoretical guarantees
   - For the quality of the trained model
   - For convergence of incremental gradient methods on the subset

Our Approach: Learning from Coresets

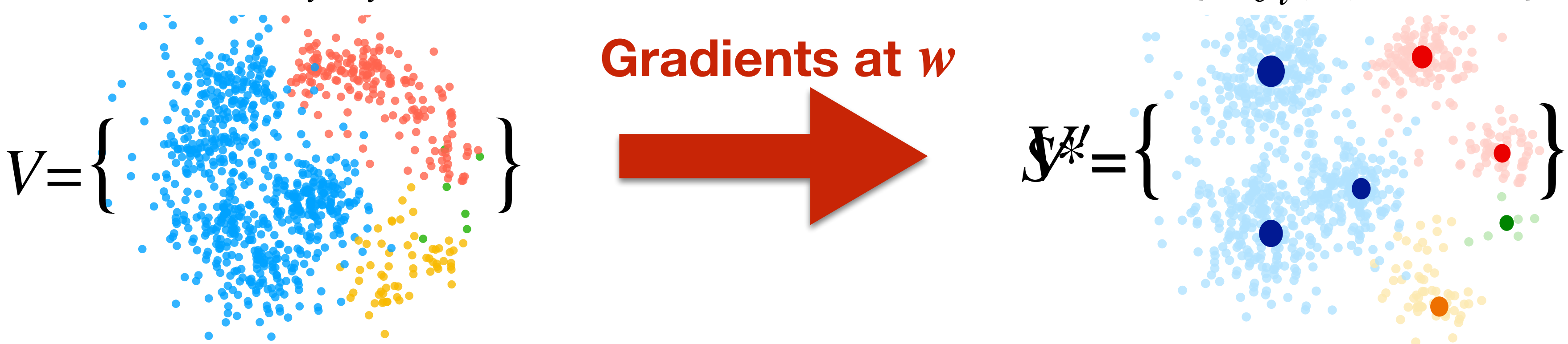
Idea: select the smallest subset S^* and weights γ that closely estimates the full gradient

$$S^* = \arg\min_{S \subseteq V, \gamma_j \geq 0\ \forall j} |S|, \quad \text{s.t.} \quad \max_{w \in \mathcal{W}} \| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \| \leq \epsilon\,.$$

Full gradient **Gradient of S**

Solution: for every $w \in \mathcal{W}$, S^* is the set of **exemplars** of all the data points in the **gradient space**

Training Data: $\{(x_i, y_i), i \in V\}$ $V' = \{\nabla f_i(w), i \in V\}$



Our Approach: Learning from Coresets

How can we find exemplars in big datasets?

- **Exemplar clustering is submodular!**

$$F(S^*) = \sum_{i \in V} \min_{j \in S^*} \|\nabla f_i(w) - \nabla f_j(w)\| \leq \epsilon$$

Submodularity is a natural diminishing returns property

$\forall\ A \subseteq B$ and $B \not\ni x:\qquad F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B)$

A simple greedy algorithm can find exemplars S^* in large datasets

However, S^* depends on w!

- We have to update S^* after every SGD update

Slow! :(

Our approach: Learning from Coresets

Can we find a subset S^* that bounds the estimation error for all $w \in \mathscr{W}$?

$$F(S^*) = \sum_{i \in V} \min_{j \in S^*} \|\nabla f_i(w) - \nabla f_j(w)\| \leq \epsilon$$

Idea: consider worst-case approximation of the estimation error over the entire parameter space $\mathscr{W}$

$$F(S^*) = \sum_{i \in V} \min_{j \in S^*} \|\nabla f_i(w) - \nabla f_j(w)\| \leq \sum_{i \in V} \min_{j \in S^*} \underbrace{\max_{w \in \mathscr{W}} \|\nabla f_i(w) - \nabla f_j(w)\|}_{d_{ij}} \leq \epsilon$$

d_{ij}: upper-bound on the gradient difference over the entire parameter space $\mathscr{W}$

Our approach: Learning from Coresets

How can we efficiently find upper-bounds d_{ij}?

- **Convex** $f(w)$: Linear/logistic/ridge regression, regularized SVM

Feature vector

$$d_{ij} \leq \text{const. } \|x_i - x_j\|$$

☑ S^* **can be found as a preprocessing step**

- **Non-convex** $f(w)$: Neural networks

Input to the last layer

$$d_{ij} \leq \text{const. } (\|\nabla_{z_i^{(L)}} f_i(w) - \nabla_{z_j^{(L)}} f_j(w)\|)$$ [KF'19]

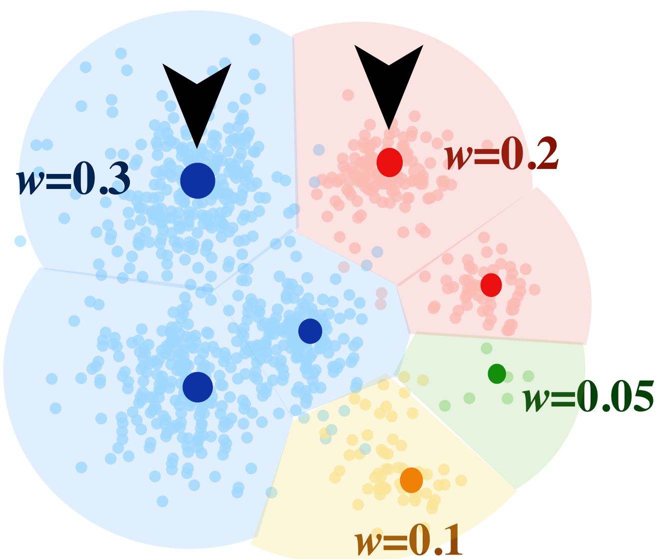
☑ d_{ij} **is cheap to compute, but we have to update** S^*

Our Approach: CRAIG

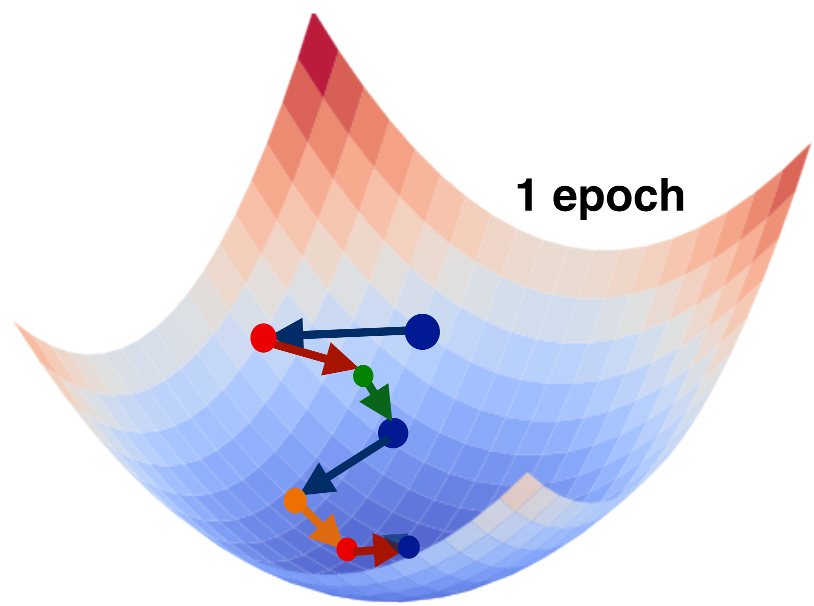
Idea: select a weighted subset that closely estimates the full gradient

Algorithm:

- **(1)** use **greedy** to find the set of exemplars S^* from dataset V
- **(2)** **weight** every elements of S^* by the size of the corresponding cluster
- **(3)** apply weighted incremental gradient descent on S^*



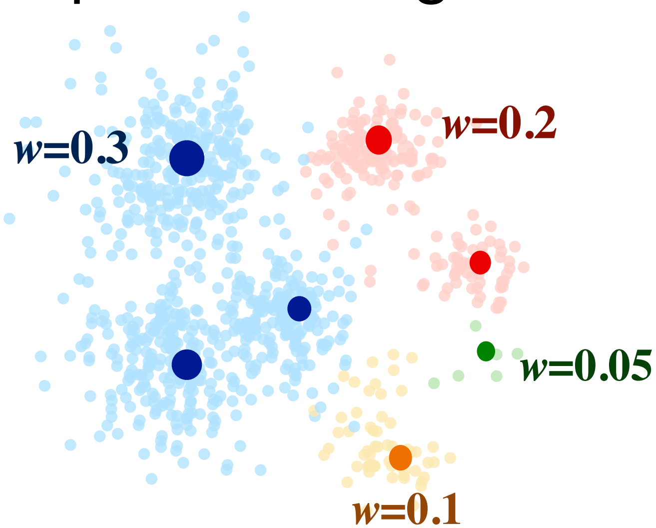
Gradients of data points $i \in V$

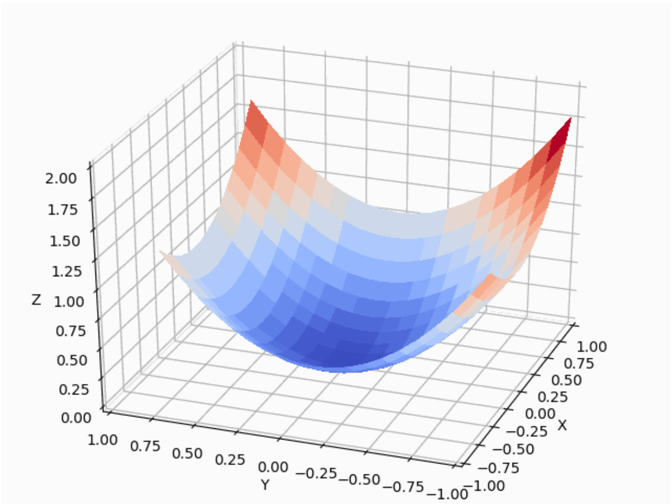


Loss function

Our approach: CRAIG

Weighted incremental gradient descent on the subset $S \subseteq V$ of exemplars in the gradient space





Theorem: For a μ-strongly convex loss function, CRAIG with decaying step-size $\Theta(1/k^\tau), \tau < 1$ converges to a $2\epsilon/\mu$ neighborhood of the optimal solution, with a rate of $\mathcal{O}(1/k^\tau)$

We get up to $|V|/|S|$ speedup!

Existing Techniques

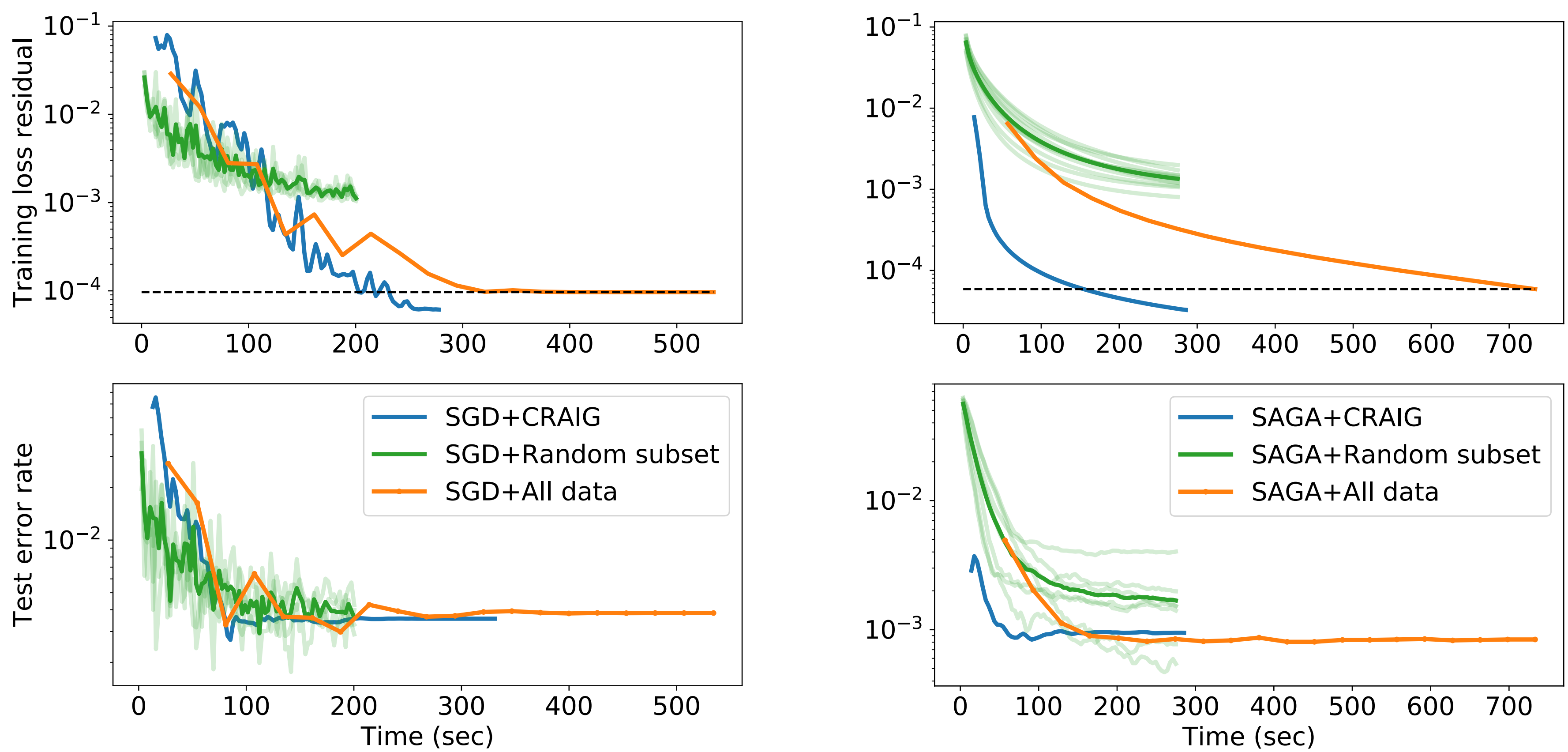
Speeding up stochastic gradient methods

- Variance reduction techniques [JZ’13, DB’14, A’18]
- Choosing better step sizes [KB’14, DHS’11, Z’12]
- Importance sampling [NSW’13, ZZ'14, KF’18]

CRAIG is complementary to all the above methods

Application of CRAIG to Logistic Regression

Training on subsets of size **10%** of Covtype with **581K points**

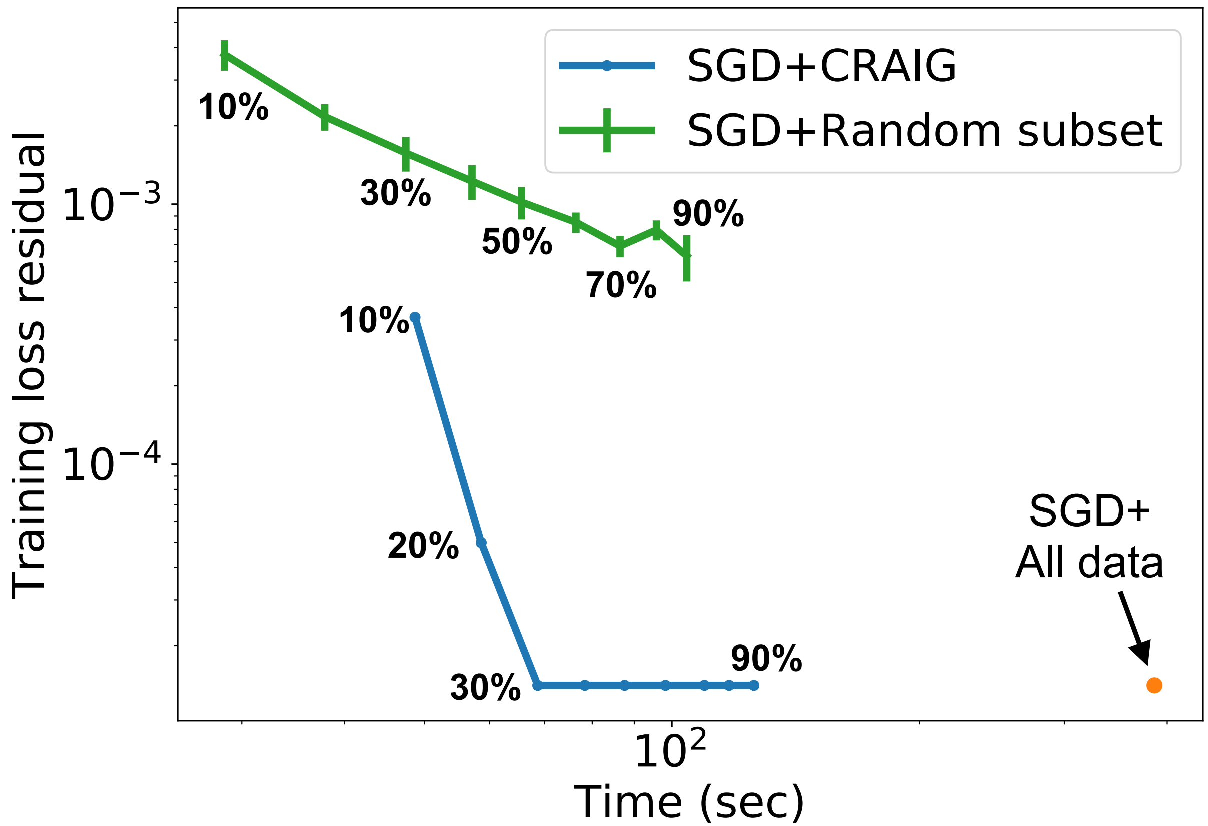


Up to 6x faster than training on the full data, with the same accuracy

Application of CRAIG to Logistic Regression

Training on subsets of various size of ljcnn1 with **50K points**

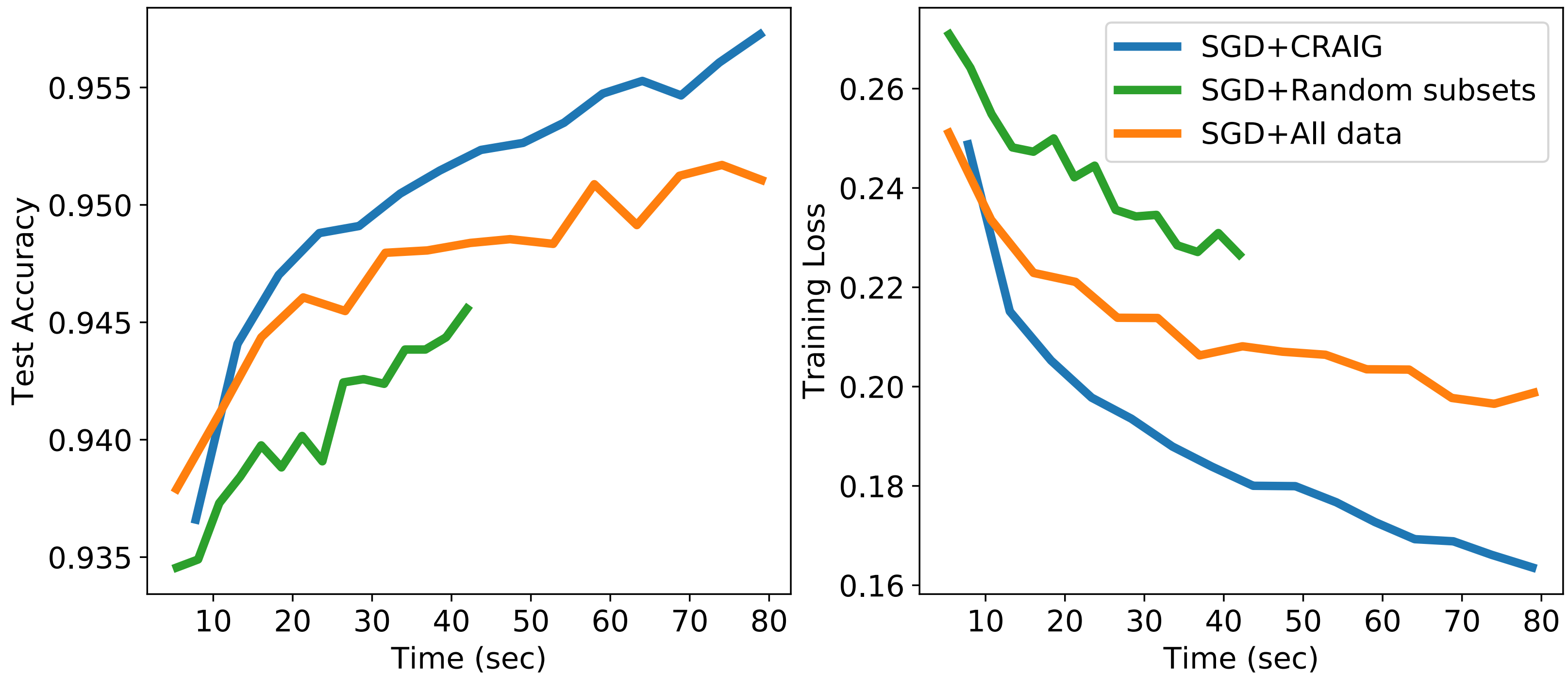
(Imbalanced)



Up to 7x faster than training on the full data, with the same accuracy

Application of CRAIG to Neural Networks

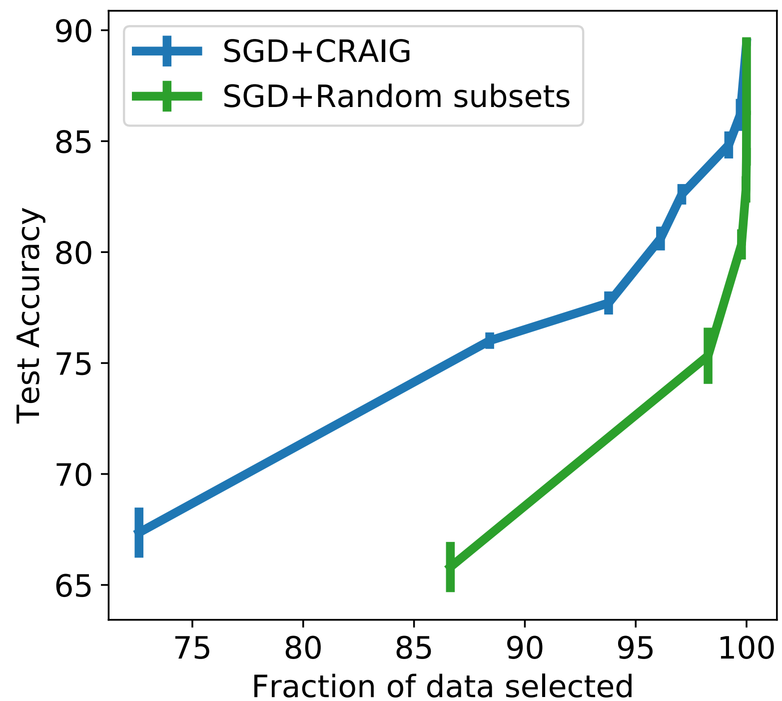
Training on MNIST with a 2-layer neural network with 50K points



2x-3x faster than training on the full data, with **better generalization**

Application of CRAIG to Deep Networks

Training ResNet20 on subsets of various size from CIFAR10 with 50K points



CRAIG is data-efficient

Summary

- We developed the first rigorous method for data-efficient training of general machine learning models
  - Converges to the near optimal solution
  - Similar convergence rate as Incremental gradient methods
  - Speeds up training by up to 7x for logistic regression and 3x for deep neural networks

Come to our poster for more details!