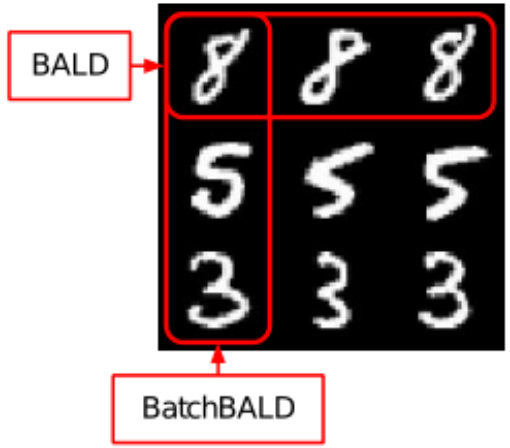
BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning

(NIPS 2019)

Motivation of Batch Active Learning

- Reduce iteration times to improve data efficiency and reduce total run time
  - Re-training a deep model is computationally expensive
  - Mobilising an expert is time consuming
- Considering the joint and complementary relationship between sample points, diverse.
  - Naively acquiring the top k scoring points can lead to redundant acquisitions.



BALD & Dropout variation inference

- Likelihod: $p(y \mid \theta, x)$
- Bayesian: prior: $p(\theta)$, posterior: $p(\theta \mid \mathcal{D})$

BALD: "Bayesian Active Learning by Disagreement"

$$\mathrm{I}[\boldsymbol{\theta}, y|\boldsymbol{x}, \mathcal{D}] = \arg\max_{\boldsymbol{x}} \mathrm{H}[y|\boldsymbol{x}, \mathcal{D}] - \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathcal{D})}\left[\mathrm{H}[y|\boldsymbol{x}, \boldsymbol{\theta}]\right]$$

- First term captures general uncertainty of model
- Second term captures the uncertainty of a given draw of the model parameters

$$\text{where}, H(y|x, D) = \sum_{C} p(y = c \mid x, D)\ \log(\ p(y = c|x, D))$$

$$p(y = c \mid x, D) = \int_{\theta\sim p(\theta|D)} p(y = c \mid x, D)\ \boldsymbol{p}(\boldsymbol{\theta} \mid \boldsymbol{D})d\theta$$

→

- Dropout as variation opproximation
- Kullback-Leibler (KL) divergence
- MCMC

BatchBALD I - Introduction

→ Score batches of points **jointly**

Joint entropy: Hard to compute

$$a_{BatchBALD}\left(\{x_1, \dots, x_n\}, p(\omega|\mathcal{D}_{train})\right) = \mathbb{H}(y_1, \dots, y_n|x_1, \dots, x_b, \mathcal{D}_{train})$$
$$- \mathbb{E}_{p(\omega|\mathcal{D}_{train})}[\mathbb{H}(y_1, \dots, y_n|x_1, \dots, x_b, \omega)].$$

Expectation over entropy: easy to compute

BatchBALD II - MC estimator

$$
\begin{aligned}
\mathbb{H}(y_1, \ldots, y_n) &= \mathbb{E}_{\mathrm{p}(y_1,\ldots,y_n)}\left[-\log \mathrm{p}(y_1, \ldots, y_n)\right] \\
&= \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}\, \mathbb{E}_{\mathrm{p}(y_1,\ldots,y_n|\boldsymbol{\omega})}\left[-\log \mathbb{E}_{\mathrm{p}(\boldsymbol{\omega})}\left[\mathrm{p}(y_1, \ldots, y_n \mid \boldsymbol{\omega})\right]\right] \\
&\approx -\sum_{\hat{y}_{1:n}} \left(\frac{1}{k}\sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{\omega}}_j)\right) \log\left(\frac{1}{k}\sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{\omega}}_j)\right).
\end{aligned}
$$

- k - the number of samples of the parameter distribution
- $\hat{y}$ - a configuration of a batch of labels
- Size of $\hat{y}$ grows exponentially with $|\mathcal{D}_{pool}|^b$

BatchBALD III - Greedy Approximation

To avoid combinatorial explosion: we grow the acquisition batch one by one.

$$\frac{1}{k}\sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n} \mid \hat{\boldsymbol{\omega}}_j) = \frac{1}{k}\sum_{j=1}^{k} \mathrm{p}(\hat{y}_{1:n-1} \mid \hat{\boldsymbol{\omega}}_j)\, \mathrm{p}(\hat{y}_n \mid \hat{\boldsymbol{\omega}}_j) = \left(\frac{1}{k}\hat{P}_{1:n-1}\hat{P}_n^T\right)_{\hat{y}_{1:n-1},\hat{y}_n} .$$

Algorithm 1: Greedy BatchBALD $1 - 1/e$-approximate algorithm

Input: acquisition size b, unlabelled dataset $\mathcal{D}_{\mathbf{pool}}$, model parameters $\mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}_{\mathbf{train}})$

$A_0 \leftarrow \emptyset$

for $n \leftarrow 1$ **to** b **do**

  foreach $\boldsymbol{x} \in \mathcal{D}_{pool} \setminus A_{n-1}$ **do** $s_{\boldsymbol{x}} \leftarrow a_{\texttt{BatchBALD}}\left(A_{n-1} \cup \{\boldsymbol{x}\}, \mathrm{p}(\boldsymbol{\omega} \mid \mathcal{D}_{\mathbf{train}})\right)$

  $\boldsymbol{x}_n \leftarrow \underset{\boldsymbol{x} \in \mathcal{D}_{\mathbf{pool}} \setminus A_{n-1}}{\arg\max}\ s_{\boldsymbol{x}}$

  $A_n \leftarrow A_{n-1} \cup \{\boldsymbol{x}_n\}$

end

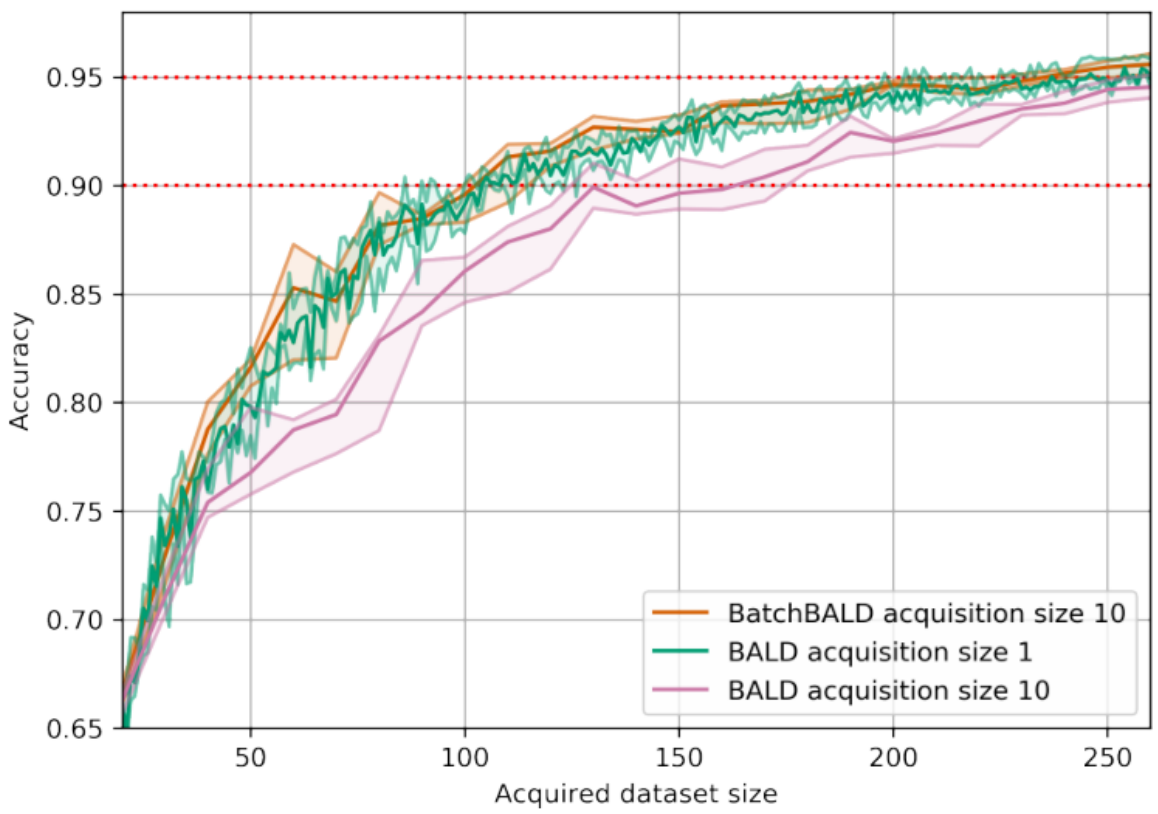
Output: acquisition batch $A_n = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_b\}$

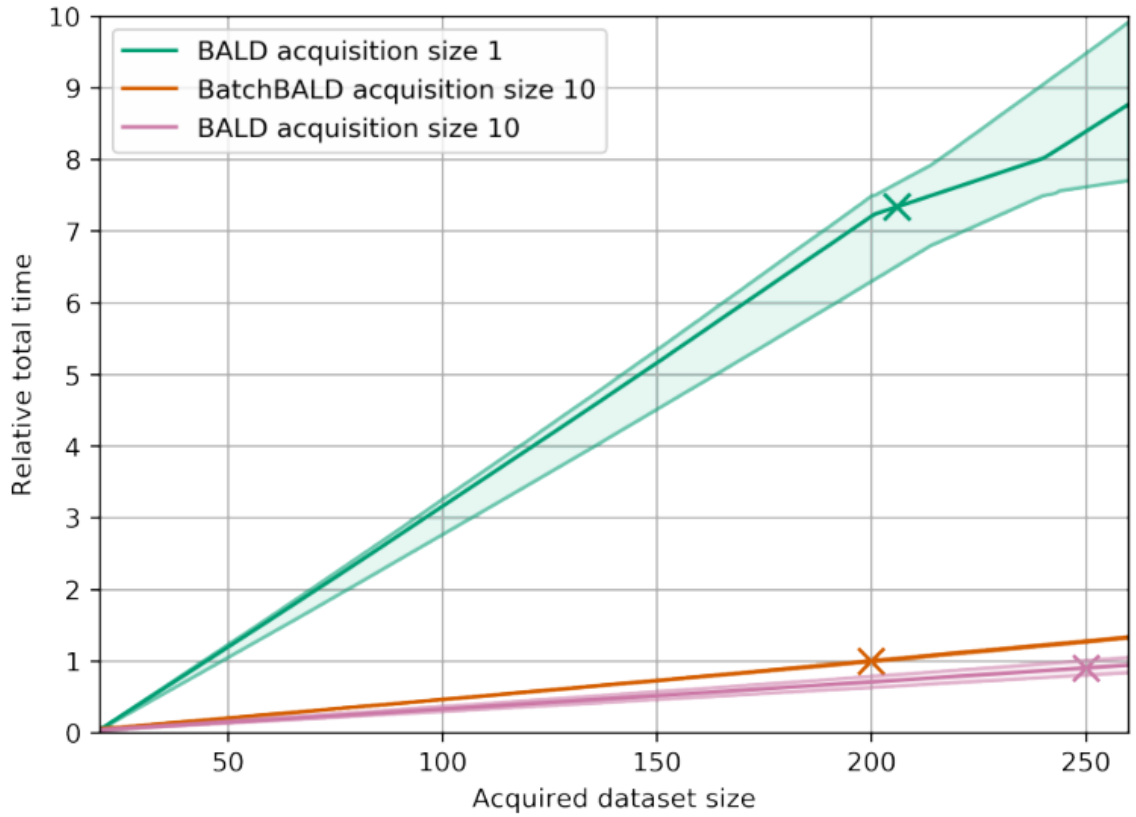
Greedy Approximation and Submodularity

- We can compute the greedy joint entropy exact for the first 4 points, afterwards we use the Monte Carlo estimator

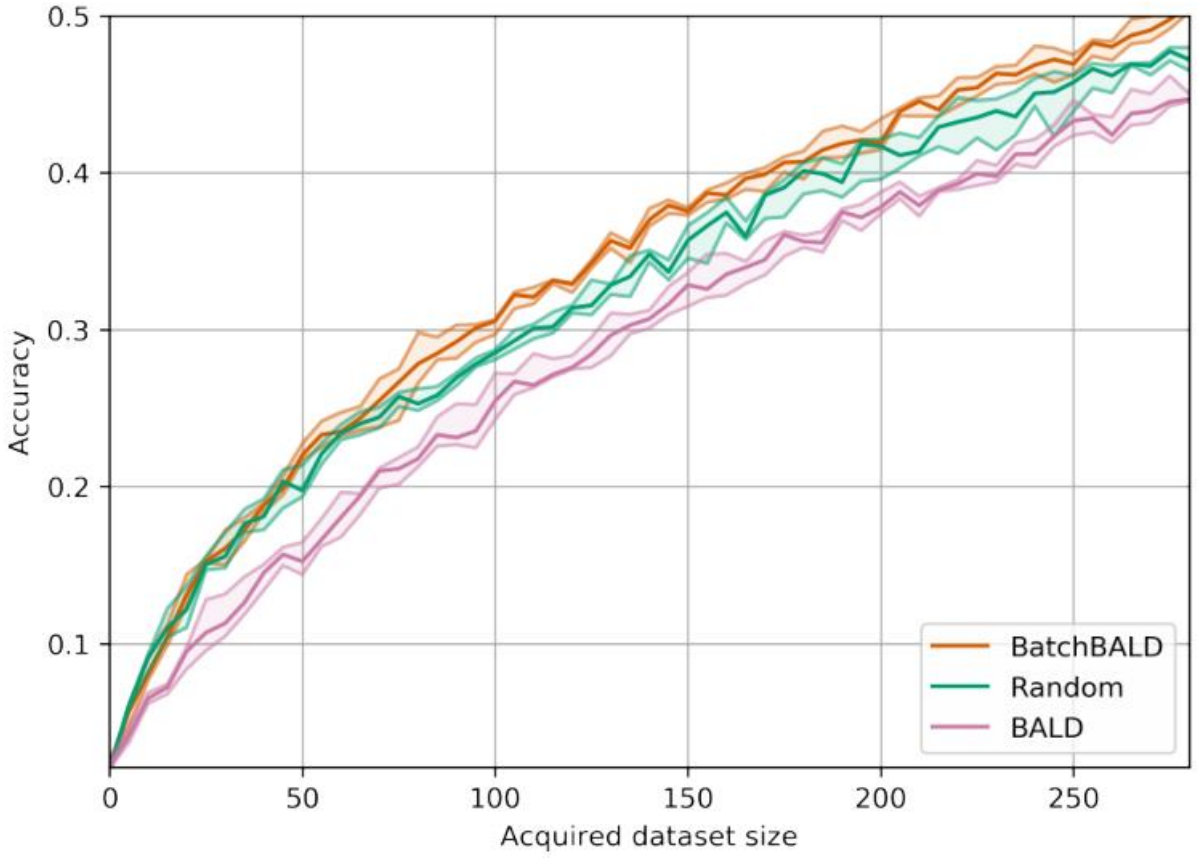
- We prove in the paper that the greedy approximation is a submodular function and therefore its error is bounded by $1 - 1/e$

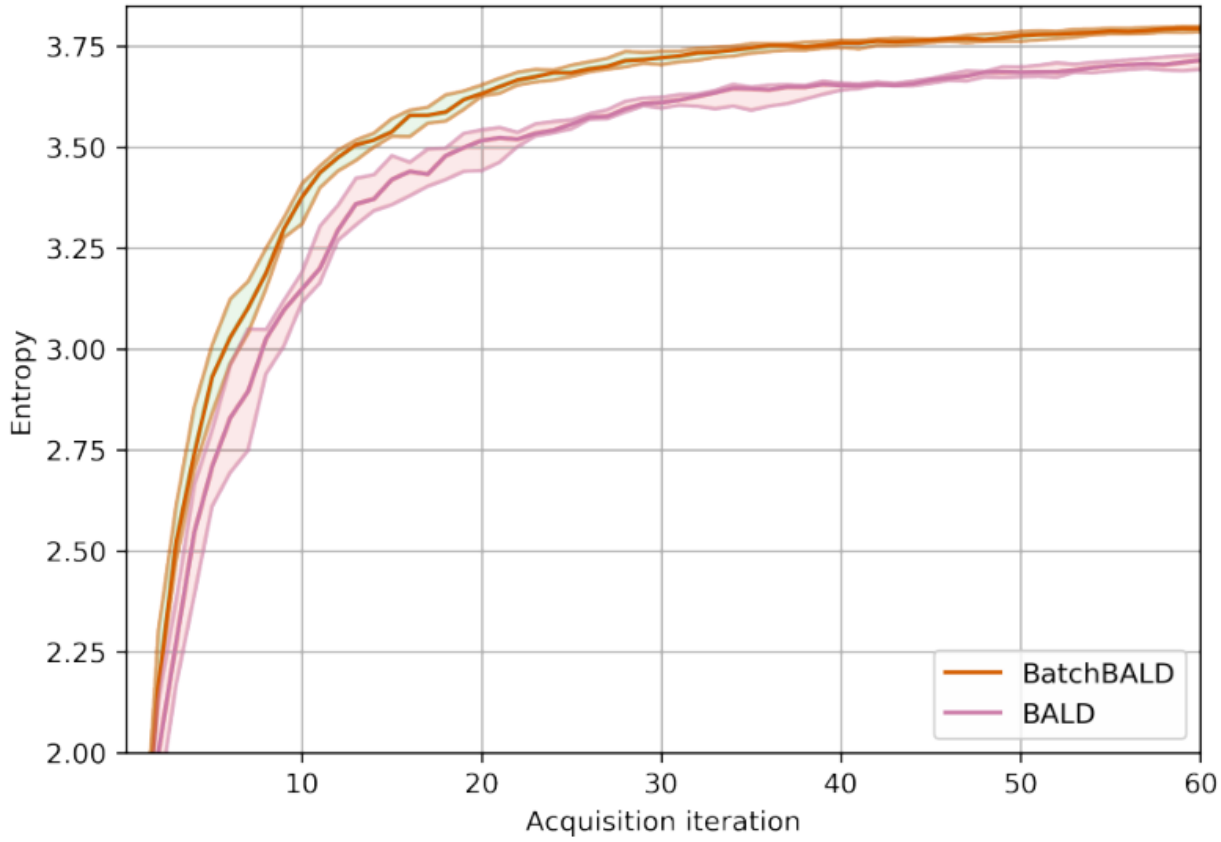
Results MNIST



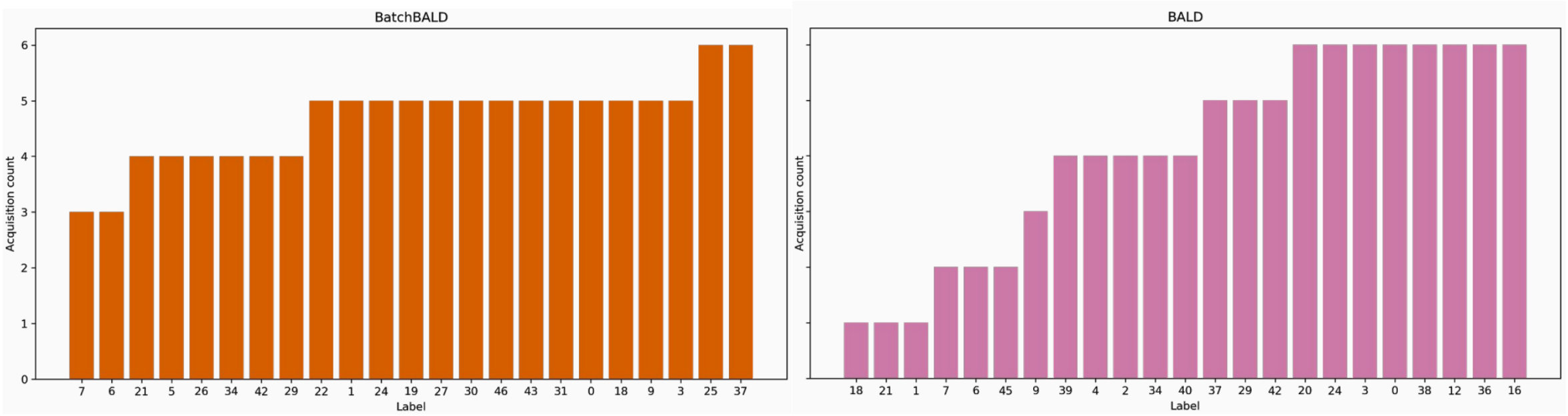


Results EMNIST

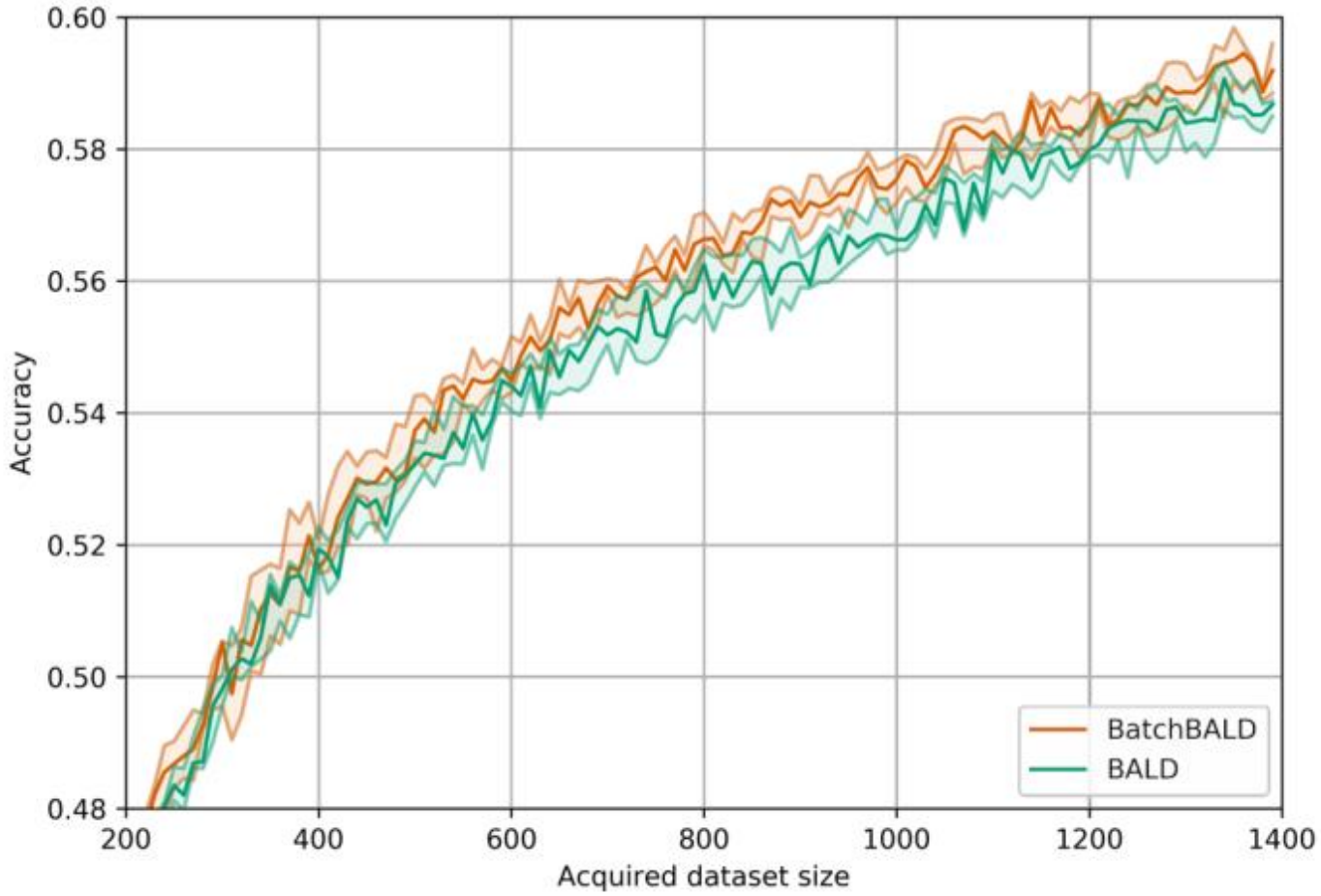




Diversity


BatchBALD
Acquisition count
0
1
2
3
4
5
6
7 6 21 5 26 34 42 29 22 1 24 19 27 30 46 43 31 0 18 9 3 25 37
Label
BALD
Acquisition count
18 21 1 7 6 45 9 39 4 2 34 40 37 29 42 20 24 3 0 38 12 36 16
Label

Results CINIC-10 (CIFAR+ImageNet)



Bayesian Batch Active Learning as Sparse Subset Approximation

(NIPS 2019)

Related paper

- Active Learning For Convolutional Neural Networks: A Core-Set Approch, ICLR, 2018
- Automated Scalable Bayesian Inference via Hilbert Coresets, ICLR, 2019

A Core-Set Approach

$$E_{\mathbf{x},y\sim p_{\mathcal{Z}}}[l(\mathbf{x},y;A_{\mathbf{s}})] \leq \underbrace{\left|E_{\mathbf{x},y\sim p_{\mathcal{Z}}}[l(\mathbf{x},y;A_{\mathbf{s}})] - \frac{1}{n}\sum_{i\in[n]} l(\mathbf{x}_i,y_i;A_{\mathbf{s}})\right|}_{\text{Generalization Error}} + \underbrace{\frac{1}{|\mathbf{s}|}\sum_{j\in\mathbf{s}} l(\mathbf{x}_j,y_j;A_{\mathbf{s}})}_{\text{Training Error}}$$

$$+ \underbrace{\left|\frac{1}{n}\sum_{i\in[n]} l(\mathbf{x}_i,y_i;A_{\mathbf{s}}) - \frac{1}{|\mathbf{s}|}\sum_{j\in\mathbf{s}} l(\mathbf{x}_j,y_j;A_{\mathbf{s}}),\right|}_{\text{Core-Set Loss}}$$

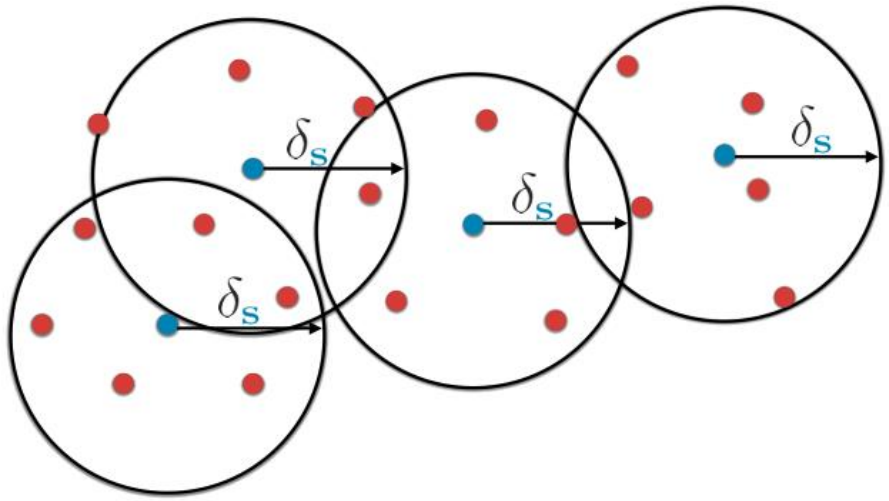


Figure 1: **Visualization of the Theorem 1.** Consider the set of selected points $\mathbf{s}$ and the points in the remainder of the dataset $[n] \setminus \mathbf{s}$, our results shows that if $\mathbf{s}$ is the $\delta_{\mathbf{s}}$ cover of the dataset, $\left|\frac{1}{n}\sum_{i\in[n]} l(\mathbf{x}_i,y_i,A_{\mathbf{s}}) - \frac{1}{|\mathbf{s}|}\sum_{j\in\mathbf{s}} l(\mathbf{x}_j,y_j;A_{\mathbf{s}})\right| \leq \mathcal{O}(\delta_{\mathbf{s}}) + \mathcal{O}\left(\sqrt{\frac{1}{n}}\right)$

$\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$

$l(\cdot,\cdot;\mathbf{w}) : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$

$\{x_i, y_i\} \in [n] \sim p(z)$

$s = \{s(j) \in [n]\}_{j\in[m]}$

$\boldsymbol{A_s}$: which outputs a set of parameters $\boldsymbol{w}$ given a labelled set $\boldsymbol{s}$.

Bayesian Coreset

$$p(\boldsymbol{\theta}|\mathcal{D}_0 \cup (\mathcal{X}_p, \mathcal{Y}_p)) = \frac{p(\boldsymbol{\theta}|\mathcal{D}_0)\, p(\mathcal{Y}_p|\mathcal{X}_p, \boldsymbol{\theta})}{p(\mathcal{Y}_p|\mathcal{X}_p, \mathcal{D}_0)} \quad (1)$$

$p(y|x,\theta)$ parameterized by $\theta \in \Theta$

$D_0 = \{x_n, y_n\}_{n=1}^{N}$: labeled dataset

$X_p = \{x_m\}_{m=1}^{M}$: unlabeled pool set

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathcal{Y}_p}[\log p(\boldsymbol{\theta}|\mathcal{D}_0 \cup (\mathcal{X}_p, \mathcal{Y}_p))] &= \mathop{\mathbb{E}}_{\mathcal{Y}_p}[\log p(\boldsymbol{\theta}|\mathcal{D}_0) + \log p(\mathcal{Y}_p|\mathcal{X}_p, \boldsymbol{\theta}) - \log p(\mathcal{Y}_p|\mathcal{X}_p, \mathcal{D}_0)] \\
&= \log p(\boldsymbol{\theta}|\mathcal{D}_0) + \mathop{\mathbb{E}}_{\mathcal{Y}_p}[\log p(\mathcal{Y}_p|\mathcal{X}_p, \boldsymbol{\theta})] + \mathbb{H}[\mathcal{Y}_p|\mathcal{X}_p, \mathcal{D}_0] \\
&= \log p(\boldsymbol{\theta}|\mathcal{D}_0) + \sum_{m=1}^{M} \Bigg(\underbrace{\mathop{\mathbb{E}}_{\boldsymbol{y}_m}[\log p(\boldsymbol{y}_m|\boldsymbol{x}_m, \boldsymbol{\theta})] + \mathbb{H}[\boldsymbol{y}_m|\boldsymbol{x}_m, \mathcal{D}_0]}_{\mathcal{L}_m(\boldsymbol{\theta})} \Bigg)
\end{aligned} \quad (2)$$

$L(\theta) = \sum_{m=1}^{M} L_m(\theta)$

$$L(w,\theta) := \sum_{m=1}^{M} w_m * L_m(\theta) \;\; satisfies \;\; |L(w,\theta) - L(\theta)| < \varepsilon |L(\theta)|, \forall \theta \in \Theta, w_m = 0 \; or \; 1 \quad (3)$$

Uniform Bayesian Coreset

$$L(w,\theta) := \sum_{m=1}^{M} w_m * L_m(\theta) \quad satisfies \quad |L(w,\theta) - L(\theta)| < \epsilon |L(\theta)|, \forall \theta \in \Theta, w_m = 0 \; or \; 1 \qquad (3)$$

The algorithm proposed by Huggins et al.(2016) to construct a Bayesian coreset is as follows. First, compute the ***sensitivity*** σ_m of each data point,

$$\sigma_m := \sup_{\theta \in \Theta} \left| \frac{L_m(\theta)}{L(\theta)} \right|$$

and then subsample the dataset by taking b independent draws with probability proportional to σ_m (resulting in a coreset of $size \leq b$) via

$$\sigma := \sum_{m=1}^{M} \sigma_m \quad (b_1, \ldots, b_M) \sim Multi\left(b, \left(\frac{\sigma_m}{\sigma}\right)_{m=1}^{M}\right) \quad W_m = \frac{\sigma}{\sigma_m}\frac{b_m}{b}$$

Since $E[W_m] = 1$, we have that $E[L(W,\theta)] = L(\theta)$, and we expect that $L(W,\theta) \to L(\theta)$ in some sense as b increases, $\epsilon^2 = O(\frac{1}{b})$

Coreset as sparse vectors

Given a function space $g : \Theta \to \mathbb{R}$ with bounded uniform norm weighted by $\mathcal{L}(\theta)$:

$$\|g\| := \sup_{\theta \in \Theta} \left| \frac{g(\theta)}{\mathcal{L}(\theta)} \right|.$$

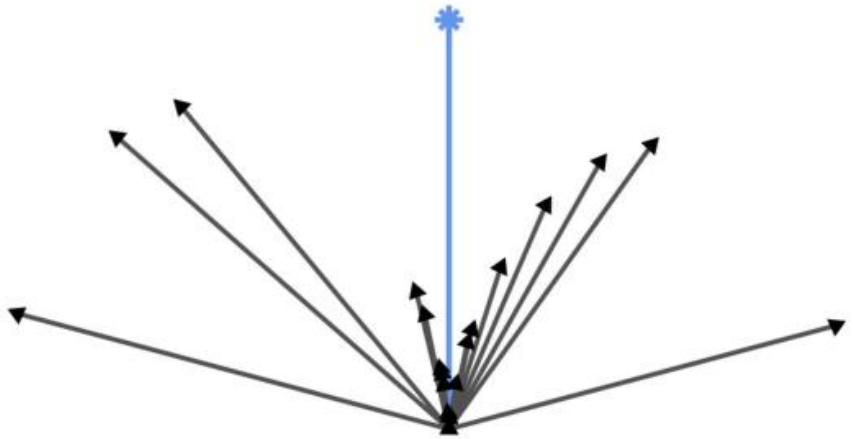
we view $\mathcal{L}_m : \Theta \mapsto \mathbb{R}$ and $\mathcal{L} = \sum_m \mathcal{L}_m$ as vectors in this function space.

we convert the problem of constructing a batch to a sparse subset approximation problem, i.e.

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\text{minimize}} \, \|\mathcal{L} - \mathcal{L}(\boldsymbol{w})\|^2 \quad \text{subject to} \quad w_m \in \{0, 1\} \quad \forall m, \sum_m \mathbb{1}_m \leq b. \qquad (4)$$

Inner product and Hilbert Coreset

- The uniform norm lacks a sense of "directionality" as it does not correspond to an inner-product.



- Constructing Bayesian coresets in a Hilbert space by inner product $< \mathcal{L}_m, \mathcal{L}_n >$

  we relax the binary weight constraint to be non-negative and replace the cardinality constraint with a polytope constraint.

Let $\sigma_m = ||\mathcal{L}_m||$, $\sigma = \sum_{m=1}^{M} \sigma_m$, and $K \in \mathbb{R}^{M \times M}$ be a kernel matrix with $K_{m,n} =< \mathcal{L}_m, \mathcal{L}_n >$. The related optimization problem is:

$$\underset{\boldsymbol{w}}{\text{minimize}} \ (\mathbf{1} - \boldsymbol{w})^T \boldsymbol{K} (\mathbf{1} - \boldsymbol{w}) \quad \text{subject to} \quad w_m \geq 0 \quad \forall m, \sum_m w_m \sigma_m = \sigma, \qquad (5)$$

Inner product and Hilbert Coreset

where we used $\|\mathcal{L} - \mathcal{L}(\boldsymbol{w})\|^2 = (\mathbf{1} - \boldsymbol{w})^T \boldsymbol{K} (\mathbf{1} - \boldsymbol{w})$.

eg. Let $M = 2$,

$L = L_1 + L_2$, $\boldsymbol{w} = (w_1, w_2)^T$, $L(w) = w_1 * L_1 + w_2 * L_2$, $\quad K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}$

$$(1 - \boldsymbol{w})^T K (1 - \boldsymbol{w}) = (1 - w_1, 1 - w_2) \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} (1 - w_1, 1 - w_2)^T$$

$$= (1 - w_1)^2 K_{11} + (1 - w_1)(1 - w_2)(K_{21} + K_{12}) + (1 - w_2)^2 K_{22}$$

$$||L - L(w)||^2 = < L - L(w), L - L(w) >$$

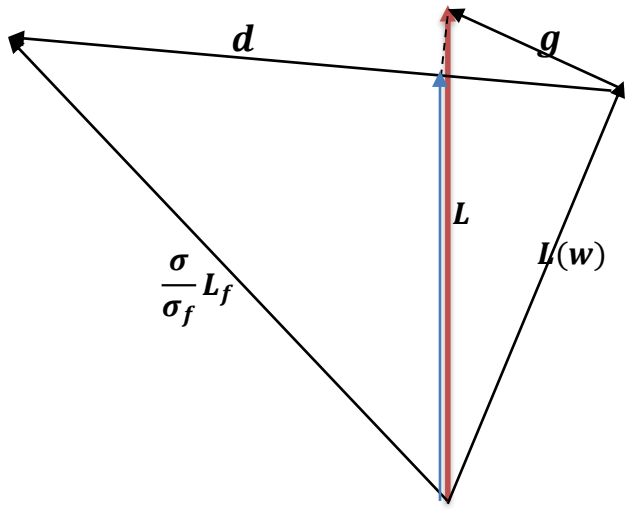
$$= < \mathbf{(1 - w_1)L_1 + (1 - w_2)L_2} \ , \mathbf{(1 - w_1)L_1 + (1 - w_2)L_2} >$$

$$= (1 - w_1)^2 < L_1, L_1 > + (1 - w_1)(1 - w_2)(< L_2, L_1 > + < L_1, L_2 >) + (1 - w_2)^2 < L_2, L_2 >$$

Frank-Wolfe optimization

Algorithm 1 Active Bayesian Coresets with Frank-Wolfe Optimization

procedure ACS-FW($b, \{\mathcal{L}_n\}_{n=1}^N, <\cdot,\cdot>$)
  $\sigma_n \leftarrow \sqrt{\langle \mathcal{L}_n, \mathcal{L}_n \rangle} \quad \forall n$ ▷ Compute norms
  $\sigma \leftarrow \sum_n \sigma_n$
  $\boldsymbol{w} \leftarrow \mathbf{0}$ ▷ Initialize weights to 0
  for $t \in 1, ..., b$ **do**
    $f \leftarrow \arg\max_{n \in N} \left[\left\langle \mathcal{L} - \mathcal{L}(\boldsymbol{w}), \frac{1}{\sigma_n} \mathcal{L}_n \right\rangle \right]$ ▷ Greedily select point f
    $\gamma \leftarrow \frac{\left[\left\langle \frac{\sigma}{\sigma_f} \mathcal{L}_f - \mathcal{L}(\boldsymbol{w}), \mathcal{L} - \mathcal{L}(\boldsymbol{w}) \right\rangle \right]}{\left[\left\langle \frac{\sigma}{\sigma_f} \mathcal{L}_f - \mathcal{L}(\boldsymbol{w}), \frac{\sigma}{\sigma_f} \mathcal{L}_f - \mathcal{L}(\boldsymbol{w}) \right\rangle \right]}$ ▷ Perform line search for step-size γ
    $\boldsymbol{w} \leftarrow (1-\gamma)\boldsymbol{w} + \gamma \frac{\sigma}{\sigma_f} \mathbf{1}_f$ ▷ Update weight for newly selected point
  end for
  return w
end procedure



6: Let $\boldsymbol{g} = L - L(w)$, $\boldsymbol{d} = \frac{\sigma}{\sigma_f} L_f - L(w)$

7: $\gamma = \frac{<\boldsymbol{g},\boldsymbol{d}>}{<\boldsymbol{d},\boldsymbol{d}>} = \frac{||\boldsymbol{g}|| \cos\theta}{||\boldsymbol{d}||}$

8: $(1-\gamma)w + \gamma \frac{\sigma}{\sigma_f} \mathbf{1}_f = w + \gamma \left(w - \frac{\sigma}{\sigma_f} \mathbf{1}_f \right)$

8: $L[w] = \mathrm{L(w)} + \gamma \left(L(w) - \frac{\sigma}{\sigma_f} L_f \right) = L(w) + \gamma \boldsymbol{d}$

$$\overline{\eta}^2 := \max_{n,m \in [N]} \left\| \frac{\mathcal{L}_n}{\sigma_n} - \frac{\mathcal{L}_m}{\sigma_m} \right\|^2,$$

$$\|\mathcal{L}(w) - \mathcal{L}\| \leq \frac{2\sigma\overline{\eta}}{\sqrt{3M+1}}.$$

Choice of inner products

- weighted Fisher inner product:

$$\langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\hat{\pi},\mathcal{F}} = \mathop{\mathbb{E}}_{\hat{\pi}} \left[\nabla_{\boldsymbol{\theta}} \mathcal{L}_n(\boldsymbol{\theta})^T \nabla_{\boldsymbol{\theta}} \mathcal{L}_m(\boldsymbol{\theta}) \right],$$

$$\|\mathcal{L}(w) - \mathcal{L}\|^2_{\hat{\pi},F} := \mathbb{E}_{\hat{\pi}} \left[\|\nabla \mathcal{L}(\theta) - \nabla \mathcal{L}(w,\theta)\|_2^2 \right],$$

where $\hat{\pi}$ to be the current posterior $p(\theta|\mathcal{D}_0)$.

- weighted Euclidean inner product:

$$\langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\hat{\pi},2} = \mathop{\mathbb{E}}_{\hat{\pi}} \left[\mathcal{L}_n(\boldsymbol{\theta}) \mathcal{L}_m(\boldsymbol{\theta}) \right].$$

$$\|\mathcal{L}(w) - \mathcal{L}\|_{\hat{\pi},2} := \mathbb{E}_{\hat{\pi}} \left[\left(\mathcal{L}(\theta) - \mathcal{L}(w,\theta) \right)^2 \right]$$

only requires tractable likelihood computations.

Random projections for non-linear models

- Is limited to models for which the inner product can be evaluated in closed form.
- Requires $\mathcal{O}|\mathcal{P}^2|$ computations
- Given random projections:

$$\hat{\mathcal{L}}_n = \frac{1}{\sqrt{J}} \left[\mathcal{L}_n(\boldsymbol{\theta}_1), \cdots, \mathcal{L}_n(\boldsymbol{\theta}_J)\right]^T, \qquad \boldsymbol{\theta}_j \sim \hat{\pi},$$

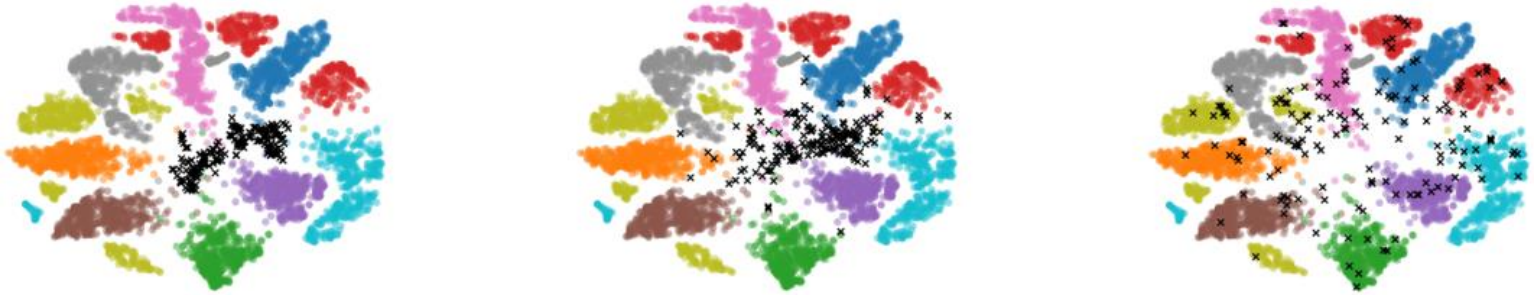
$$\langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\hat{\pi},2} \approx \hat{\mathcal{L}}_n^T \hat{\mathcal{L}}_m,$$

Algorithm 2 ACS-FW with Random Projections (for Weighted Euclidean Inner Product)

procedure ACS-FW(b, J)
  $\boldsymbol{\theta}_j \sim \hat{\pi} \quad j = 1, \ldots, J$ ▷ Sample parameters
  $\hat{\mathcal{L}}_n = \frac{1}{\sqrt{J}} \left[\mathcal{L}_n(\boldsymbol{\theta}_1), \cdots, \mathcal{L}_n(\boldsymbol{\theta}_J)\right]^T \quad \forall n$ ▷ Compute random feature projections
  return ACS-FW($b, \{\hat{\mathcal{L}}_n\}_{n=1}^N, (\cdot)^T(\cdot)$) ▷ Call Algorithm 1 using projections
end procedure

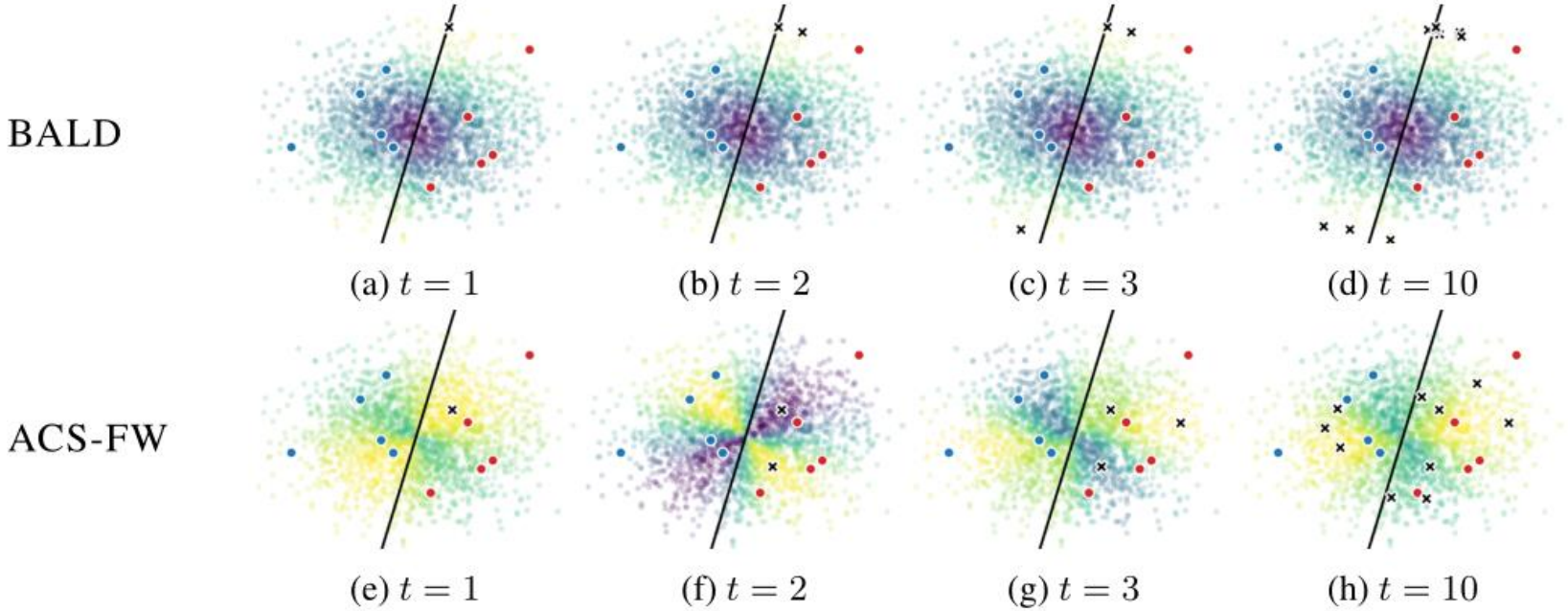
Experiments

Avoid correlated queries:



(a) MaxEnt (b) BALD (c) Ours

Figure 1: Batch construction of different AL methods on *cifar10*, shown as a t-SNE projection [12]. Given 5000 labeled points (colored by class), a batch of 200 points (black crosses) is queried.



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 10$

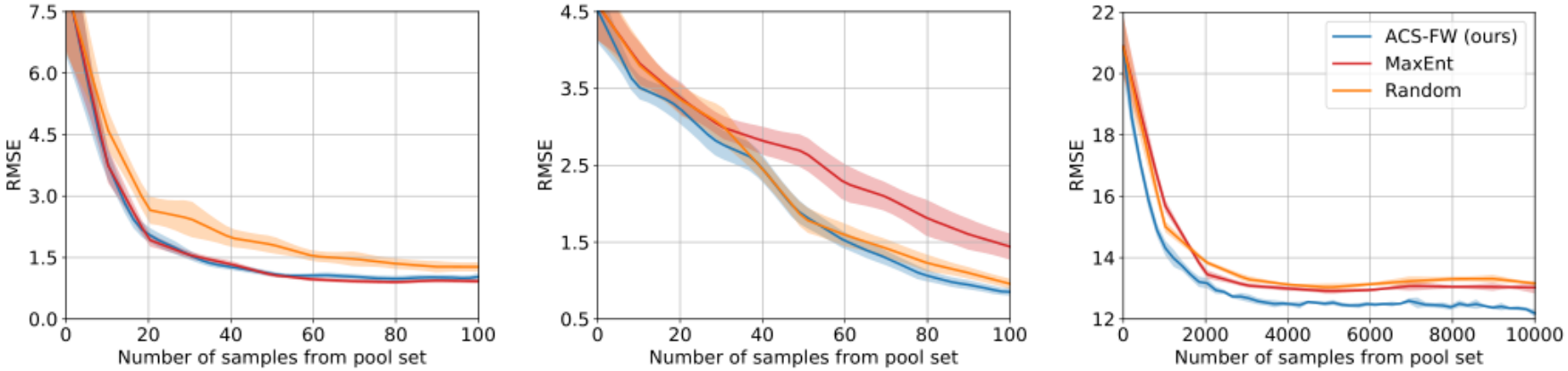
(e) $t = 1$ (f) $t = 2$ (g) $t = 3$ (h) $t = 10$

Experiments

Competitive with greedy methods in the small-data:

Table 1: Final test RMSE on UCI regression datasets averaged over 40 (*year*: 5) seeds. MAXENT-I and MAXENT-SG require order(s) of magnitudes more model updates and are thus not directly comparable.

	N	d	RANDOM	MAXENT	ACS-FW	MAXENT-I	MAXENT-SG
yacht	308	6	1.272±0.0593	**0.923±0.0319**	1.031±0.0438	0.865±0.0276	0.971±0.0350
boston	506	13	4.068±0.0852	**3.640±0.0652**	3.799±0.0858	3.467±0.0676	3.458±0.0682
energy	768	8	0.959±0.0337	1.443±0.0857	**0.855±0.0259**	0.927±0.0461	1.055±0.0740
power	9568	4	5.108±0.0468	**5.022±0.0428**	**4.984±0.0366**	4.834±0.0313	4.855±0.0339
year	515 345	90	13.165±0.0307	13.030±0.0975	**12.194±0.0596**	N/A	N/A

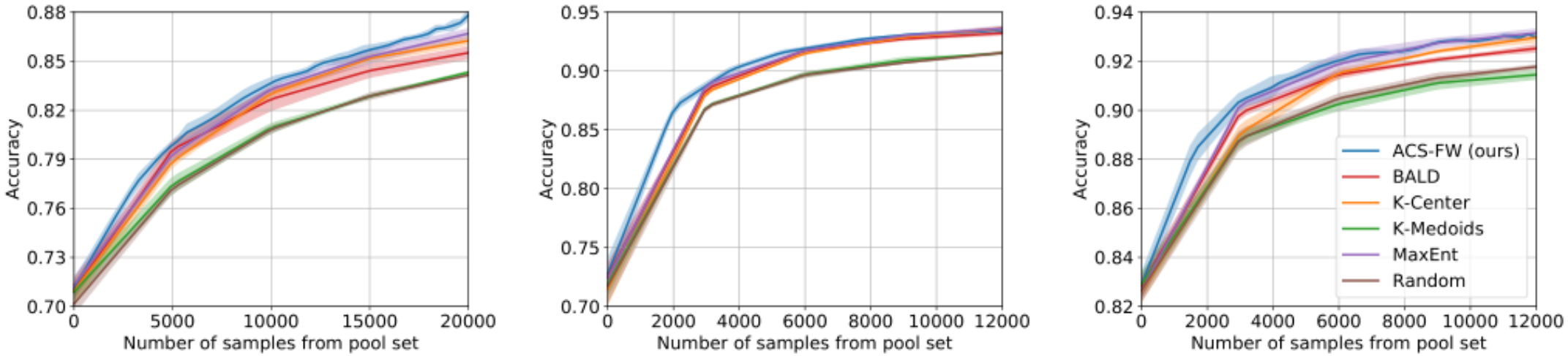


(a) yacht (b) energy (c) year

Figure 3: Test RMSE on UCI regression datasets averaged over 40 (a-b) and 5 (c) seeds during AL. Error bars denote two standard errors.

Experiments

Scale to large datasets and models:



(a) cifar10 (b) SVHN (c) Fashion MNIST

Figure 4: Test accuracy on classification tasks over 5 seeds. Error bars denote two standard errors.

Experiments

Runtime Evaluation:

Table 2: Runtime in seconds on UCI regression datasets averaged over 40 (*year*: 5) seeds. We report mean batch construction time (BT/it.) and total time (TT/it.) per AL iteration, as well as total cumulative time (total). MAXENT-I requires order(s) of magnitudes more model updates and is thus not directly comparable.

	RANDOM			MAXENT			ACS-FW			MAXENT-I		
	BT/it.	TT/it.	total	BT/it.	TT/it.	total	BT/it.	TT/it.	total	BT/it.	TT/it.	total
yacht	0.0	8.9	88.6	1.3	10.2	101.7	0.0	9.1	107.2	12.3	105.7	1057.4
boston	0.0	12.4	123.6	2.4	14.5	144.8	0.1	12.4	132.7	23.5	157.9	1578.6
energy	0.0	12.1	121.4	3.9	16.0	159.6	0.1	12.6	137.8	37.5	170.5	1704.9
power	0.4	9.4	94.0	53.0	61.7	617.0	0.8	10.2	179.8	517.3	609.1	6090.7
year	30.2	381.2	3811.6	3391.5	3746.5	37 464.6	53.0	463.8	28 475.2	N/A	N/A	N/A

Thanks