Bayesian Batch Active Learning as Sparse Subset Approximation

Robert Pinsler
Jonathan Gordon
Eric Nalisnick
José Miguel Hernández-Lobato
• Acquiring labels for supervised learning can be costly and time-consuming
• In such settings, active learning (AL) enables data-efficient model training by intelligently selecting points for which labels should be requested
Introduction

Train model

Labeled training set

Unlabeled pool set

Oracle

Model

Pool-based active learning (AL)

Select queries
Sequential AL loop

Train model → Query single data point → Train model
Introduction

Batch AL approaches:
- scale to large datasets and models
- enable parallel data acquisition
- (ideally) trade off diversity and representativeness

How to construct such a batch?
Bayesian Batch Active Learning

**Bayesian approach:** Choose set of points that maximally reduces uncertainty over parameter posterior

- NP-hard, but greedy approximations exist: *MaxEnt, BALD*
- Naïve batch strategy: Select $b$ best points according to acquisition function
Bayesian Batch Active Learning

**Bayesian approach:** Choose set of points that maximally reduces uncertainty over parameter posterior

- NP-hard, but greedy approximations exist: *MaxEnt, BALD*
- Naïve batch strategy: Select $b$ best points according to acquisition function

Budget is **wasted** on selecting nearby points
Bayesian Batch Active Learning

**Bayesian approach:** Choose set of points that maximally reduces uncertainty over parameter posterior

- NP-hard, but greedy approximations exist: *MaxEnt, BALD*
- Naïve batch strategy: Select $b$ best points according to acquisition function

**Idea:** Re-cast batch construction as optimizing a sparse subset approximation to complete data posterior
Bayesian Batch Active Learning

**Bayesian approach:** Choose set of points that maximally reduces uncertainty over parameter posterior
- NP-hard, but greedy approximations exist: *MaxEnt, BALD*
- Naïve batch strategy: Select $b$ best points according to acquisition function

---

**Idea:** Re-cast batch construction as optimizing a sparse subset approximation to complete data posterior
Related Work
Bayesian coresets

Idea: Re-cast batch construction as optimizing a sparse subset approximation to complete data posterior

We take inspiration from **Bayesian coresets**
- **Coreset**: Summarize data by sparse, weighted subset
- **Bayesian coreset**: Approximate posterior by sparse, weighted subset
**Related Work**

**Bayesian coresets**

**Idea:** Re-cast batch construction as optimizing a sparse subset approximation to complete data posterior

We take inspiration from **Bayesian coresets**

- **Coreset:** Summarize data by sparse, weighted subset
- **Bayesian coreset:** Approximate posterior by sparse, weighted subset

- **Batch AL with Bayesian coresets:** Batch = Bayesian coreset
Choose batch $\mathcal{D}' = (\tilde{X}', \tilde{Y}') \subseteq \mathcal{D}_p$ such that $\log p(\theta|\mathcal{D}_0 \cup \mathcal{D}')$ best approximates $\log p(\theta|\mathcal{D}_0 \cup \mathcal{D}_p)$.
Choose batch $\mathcal{D}' = (\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}) \subseteq \mathcal{D}_p$ such that
$log p(\theta|\mathcal{D}_0 \cup \mathcal{D}')$ best approximates $log p(\theta|\mathcal{D}_0 \cup \mathcal{D}_p)$

We don't know the labels of the points in the pool set before querying them
Choose batch $\mathcal{D}' = (\mathcal{X}', \mathcal{Y}') \subseteq \mathcal{D}_p$ such that 

$$\log p(\theta | \mathcal{D}_0 \cup \mathcal{D}')$$

best approximates $\log p(\theta | \mathcal{D}_0 \cup \mathcal{D}_p)$

We don't know the labels of the points in the pool set before querying them.

Take expectation w.r.t. current predictive posterior distribution:

$$\mathbb{E}_{\mathcal{Y}_p} \left[ \log p(\theta | \mathcal{D}_0 \cup (\mathcal{X}_p, \mathcal{Y}_p)) \right] = \log p(\theta | \mathcal{D}_0) + \mathbb{E}_{\mathcal{Y}_p} \left[ \log p(\mathcal{Y}_p | \mathcal{X}_p, \theta) \right] + \mathbb{H}[\mathcal{Y}_p | \mathcal{X}_p, \mathcal{D}_0]$$

$$= \log p(\theta | \mathcal{D}_0) + \sum_{m=1}^{M} \left( \mathbb{E}_{y_m} \left[ \log p(y_m | x_m, \theta) \right] + \mathbb{H}[y_m | x_m, \mathcal{D}_0] \right),$$

$$\mathcal{L}_m(\theta)$$
Batch Construction as Sparse Subset Approximation

Choose batch $\mathcal{D}' = (\mathcal{X}', \mathcal{Y}') \subseteq \mathcal{D}_p$ such that
$log p(\theta | \mathcal{D}_0 \cup \mathcal{D}')$ best approximates $log p(\theta | \mathcal{D}_0 \cup \mathcal{D}_p)$

⚠️ We don’t know the labels of the points in the pool set before querying them

Take expectation w.r.t. current predictive posterior distribution:

$$
\mathbb{E}_{\mathcal{Y}_p} [ \log p(\theta | \mathcal{D}_0 \cup (\mathcal{X}_p, \mathcal{Y}_p)) ] = \log p(\theta | \mathcal{D}_0) + \mathbb{E}_{\mathcal{Y}_p} [ \log p(\mathcal{Y}_p | \mathcal{X}_p, \theta) ] + \mathbb{H}[\mathcal{Y}_p | \mathcal{X}_p, \mathcal{D}_0]
$$

$$
= \log p(\theta | \mathcal{D}_0) + \sum_{m=1}^{M} \left( \mathbb{E}_{\mathcal{y}_m} [ \log p(\mathcal{y}_m | \mathcal{x}_m, \theta) ] + \mathbb{H}[\mathcal{y}_m | \mathcal{x}_m, \mathcal{D}_0] \right)
$$

$$
\mathcal{L}_m(\theta)
$$
Batch Construction as Sparse Subset Approximation
Hilbert coresets

\[ \mathcal{L} = \sum_{m} \mathcal{L}_m \]

\[ \mathcal{L}_m : \Theta \mapsto \mathbb{R} \]
Batch Construction as Sparse Subset Approximation
Hilbert coresets

\[ \mathcal{L} = \sum_m \mathcal{L}_m \]

\[ \mathcal{L}(w) = \sum_m w_m \mathcal{L}_m \]

\[ \mathcal{L}_m : \Theta \mapsto \mathbb{R} \]

\( w^* = \min_w \| \mathcal{L} - \mathcal{L}(w) \|^2 \) subject to \( w_m \in \{0, 1\} \) \( \forall m, \sum_m 1_m \leq b. \)
Batch Construction as Sparse Subset Approximation

Hilbert coresets

\[ \mathcal{L} = \sum_m \mathcal{L}_m \]
\[ \mathcal{L}(w) = \sum_m w_m \mathcal{L}_m \]

\[ \mathcal{L}_m : \Theta \mapsto \mathbb{R} \]

\[ w^* = \min_w \| \mathcal{L} - \mathcal{L}(w) \|^2 \text{ subject to } w_m \in \{0, 1\} \quad \forall m, \sum_m 1_m \leq b. \]

- Considers **directionality** of residual error → adaptively construct batch while accounting for similarity between data points (induced by norm)
- Still intractable!
Batch Construction as Sparse Subset Approximation
Frank-Wolfe optimization

\[ w^* = \minimize_w \| \mathcal{L} - \mathcal{L}(w) \|^2 \quad \text{subject to} \quad w_m \in \{0, 1\} \quad \forall m, \quad \sum_m 1_m \leq b. \]
Batch Construction as Sparse Subset Approximation
Frank-Wolfe optimization

\[ w^* = \min_w \| \mathcal{L} - \mathcal{L}(w) \|^2 \quad \text{subject to} \quad w_m \in \{0, 1\} \quad \forall m, \quad \sum_m 1_m \leq b. \]

\[ w_m \geq 0 \quad \forall m, \quad \sum_m w_m \sigma_m = \sigma. \]

\[ \sigma_m = \| \mathcal{L}_m \|, \quad \sigma = \sum_m \sigma_m \]

1 Relax constraints
Batch Construction as Sparse Subset Approximation
Frank-Wolfe optimization

\[ w^* = \min_w \| \mathcal{L} - \mathcal{L}(w) \|^2 \quad \text{subject to} \quad w_m \in \{0, 1\} \quad \forall m, \quad \sum \mathbb{1}_m \leq b. \]

1. Relax constraints

\[ w_m \geq 0 \quad \forall m, \quad \sum w_m \sigma_m = \sigma. \]

\[ \sigma_m = \| \mathcal{L}_m \|, \quad \sigma = \sum \sigma_m \]

2. Apply Frank-Wolfe algorithm
   - Geometrically motivated convex optimization algorithm
   - Iteratively selects vector most aligned with residual error
   - Corresponds to adding at most one data point to batch in every iteration
**Batch Construction as Sparse Subset Approximation**

Frank-Wolfe optimization

\[ w^* = \min_w \| \mathcal{L} - \mathcal{L}(w) \|^2 \quad \text{subject to} \quad w_m \in \{0, 1\} \quad \forall m, \quad \sum_m 1_m \leq b. \]

1. **Relax constraints**

\[
\begin{align*}
  w_m &\geq 0 \quad \forall m, \quad \sum_m w_m \sigma_m = \sigma. \\
  \sigma_m &= \| \mathcal{L}_m \|, \quad \sigma = \sum_m \sigma_m
\end{align*}
\]

2. **Apply Frank-Wolfe algorithm**
   - Geometrically motivated convex optimization algorithm
   - Iteratively selects vector most aligned with residual error
   - Corresponds to adding at most one data point to batch in every iteration

3. **Project** continuous weights back to feasible space (i.e. binarize them)
Batch Construction as Sparse Subset Approximation
Frank-Wolfe optimization

\[ w^* = \min_w \| \mathcal{L} - \mathcal{L}(w) \|^2 \] subject to \[ w_m \in \{0, 1\} \quad \forall m, \sum m \leq b. \]
\[ w_m \geq 0 \quad \forall m, \sum w_m \sigma_m = \sigma. \]
\[ \sigma_m = \| \mathcal{L}_m \| \quad \sigma = \sum \sigma_m \]

1. Relax constraints

2. Apply Frank-Wolfe algorithm
   - Geometrically motivated convex optimization algorithm
   - Iteratively selects vector most aligned with residual error
   - Corresponds to adding at most one data point to batch in every iteration

3. Project continuous weights back to feasible space (i.e. binarize them)

Which norm is appropriate?
Norm is induced by inner product, e.g.

1. **Weighted Fisher inner product**

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

- Leads to simple, interpretable expressions for linear models
- Requires taking gradients w.r.t. parameters
- Scales quadratically with pool set size
Batch Construction as Sparse Subset Approximation
Choice of Inner Products

Norm is induced by inner product, e.g.

1. Weighted Fisher inner product

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

- Leads to simple, interpretable expressions for linear models
- Requires taking gradients w.r.t. parameters
- Scales quadratically with pool set size

**Example:** Linear regression

\[ y_n = \theta^T x_n + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(0, \sigma_0^2), \quad \theta \sim p(\theta) \]

\[ \langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\hat{\pi}, \mathcal{F}} = \frac{x_n^T x_m}{\sigma_0^4} x_n^T \Sigma_{\theta} x_m \]
Batch Construction as Sparse Subset Approximation

Choice of Inner Products

Norm is induced by inner product, e.g.

1. **Weighted Fisher inner product**

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

- Leads to simple, interpretable expressions for linear models
- Requires taking gradients w.r.t. parameters
- Scales quadratically with pool set size

**Example:** Linear regression

\[
y_n = \theta^T x_n + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(0, \sigma_0^2), \quad \theta \sim p(\theta)
\]

\[
\langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\hat{\pi}, \mathcal{F}} = \frac{x_n^T x_m}{\sigma_0^4} x_n^T \Sigma_{\theta} x_m
\]

\[
\alpha_{\text{ACS}}(x_n; \mathcal{D}_0) = \frac{x_n^T x_n}{\sigma_0^4} x_n^T \Sigma_{\theta} x_n, \quad \alpha_{\text{BALD}}(x_n; \mathcal{D}_0) = \frac{1}{2} \log \left( 1 + \frac{x_n^T \Sigma_{\theta} x_n}{\sigma_0^2} \right)
\]

- Connections to BALD, leverage scores and influence functions
- Probit regression also yields interpretable closed-form solution
Norm is induced by inner product, e.g.

1. **Weighted Fisher inner product**
   \[
   \langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\mathcal{F},\pi} = \mathbb{E}_{\hat{\pi}} \left[ \nabla_{\theta} \mathcal{L}_n(\theta)^T \nabla_{\theta} \mathcal{L}_m(\theta) \right]
   \]
   
   + Leads to simple, interpretable expressions for linear models
   -- Requires taking gradients w.r.t. parameters
   -- Scales quadratically with pool set size

2. **Weighted Euclidean inner product**
   \[
   \langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\pi,2} = \mathbb{E}_{\hat{\pi}} [\mathcal{L}_n(\theta) \mathcal{L}_m(\theta)]
   \]
   
   + Only requires tractable likelihood computations
   + Scalable to large pool set sizes (linearly) and complex, non-linear models through random projections
   -- No gradient information utilized
Batch Construction as Sparse Subset Approximation
Choice of Inner Products

Norm is induced by inner product, e.g.

1. **Weighted Fisher inner product** \( \langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\hat{\pi}, \mathcal{F}} = \mathbb{E}_{\hat{\pi}} \left[ \nabla_{\theta} \mathcal{L}_n(\theta)^T \nabla_{\theta} \mathcal{L}_m(\theta) \right] \)
   
   + Leads to simple, interpretable expressions for linear models
   -- Requires taking gradients w.r.t. parameters
   -- Scales quadratically with pool set size

2. **Weighted Euclidean inner product** \( \langle \mathcal{L}_n, \mathcal{L}_m \rangle_{\hat{\pi}, 2} = \mathbb{E}_{\hat{\pi}} \left[ \mathcal{L}_n(\theta) \mathcal{L}_m(\theta) \right] \)
   
   + Only requires tractable likelihood computations
   + Scalable to large pool set sizes (linearly) and complex, non-linear models through random projections
   -- No gradient information utilized

**J-dimensional random projection** in Euclidean space

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

\[ \hat{\mathcal{L}}_n = \frac{1}{\sqrt{J}} \left[ \mathcal{L}_n(\theta_1), \cdots, \mathcal{L}_n(\theta_J) \right]^T, \quad \theta_j \sim \hat{\pi} \]
Experiments

(i) Does our approach avoid correlated queries?  
   closed form

(ii) Is our method competitive in the small-data regime?  
   closed form

(iii) Does our method scale to large datasets and models?  
   projections
Experimental Setup

Experiments

(i) Does our approach avoid correlated queries? closed form

(ii) Is our method competitive in the small-data regime? closed form

(iii) Does our method scale to large datasets and models? projections

Model: Neural Linear

- Deterministic feature extractor (e.g. ConvNet)
- Stochastic fully connected layer

Exact inference (regression)
Mean-field VI (classification)
Experiments: Probit Regression
Does our approach avoid correlated queries?

BALD
(a) $t = 1$

ACS-FW
(e) $t = 1$
Experiments: Probit Regression
Does our approach avoid correlated queries?

**BALD**

(a) $t = 1$

(b) $t = 2$

**ACS-FW**

(e) $t = 1$

No change
Experiments: Probit Regression
Does our approach avoid correlated queries?

**BALD**
- (a) $t = 1$
- (b) $t = 2$

**ACS-FW**
- (e) $t = 1$
- (f) $t = 2$

No change

Rotates in data space
Experiments: Probit Regression
Does our approach avoid correlated queries?

BALD
(a) $t = 1$
(b) $t = 2$
(c) $t = 3$

ACS-FW
(e) $t = 1$
(f) $t = 2$
(g) $t = 3$

And again...
Experiments: Probit Regression
Does our approach avoid correlated queries?

**BALD**

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

**ACS-FW**

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

ACS-FW queries diverse batch of points
Experiments: Regression
Is our method competitive in the small-data regime?

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>d</th>
<th>RANDOM</th>
<th>MAXENT</th>
<th>ACS-FW</th>
<th>MAXENT (greedy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yacht</td>
<td>308</td>
<td>6</td>
<td>1.342 ± 0.0404</td>
<td>0.991 ± 0.0423</td>
<td>1.033 ± 0.0374</td>
<td>0.940 ± 0.0355</td>
</tr>
<tr>
<td>boston</td>
<td>506</td>
<td>13</td>
<td>4.084 ± 0.0523</td>
<td>3.681 ± 0.0826</td>
<td>3.851 ± 0.0880</td>
<td>3.523 ± 0.0736</td>
</tr>
<tr>
<td>energy</td>
<td>768</td>
<td>8</td>
<td>1.017 ± 0.0421</td>
<td>1.463 ± 0.0863</td>
<td>0.841 ± 0.0348</td>
<td>1.038 ± 0.0596</td>
</tr>
<tr>
<td>power</td>
<td>9568</td>
<td>4</td>
<td>5.135 ± 0.0471</td>
<td>4.977 ± 0.0396</td>
<td>4.978 ± 0.0443</td>
<td>4.865 ± 0.0370</td>
</tr>
</tbody>
</table>
Experiments: Regression
Is our method competitive in the small-data regime?

Table 1: Final test RMSE on UCI regression datasets averaged over 40 (year: 5) seeds.

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>d</th>
<th>RANDOM</th>
<th>MAXENT</th>
<th>ACS-FW</th>
<th>MAXENT (greedy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yacht</td>
<td>308</td>
<td>6</td>
<td>1.342 ± 0.0404</td>
<td><strong>0.991 ± 0.0423</strong></td>
<td>1.033 ± 0.0374</td>
<td>0.940 ± 0.0355</td>
</tr>
<tr>
<td>boston</td>
<td>506</td>
<td>13</td>
<td>4.084 ± 0.0523</td>
<td><strong>3.681 ± 0.0826</strong></td>
<td>3.851 ± 0.0880</td>
<td>3.523 ± 0.0736</td>
</tr>
<tr>
<td>energy</td>
<td>768</td>
<td>8</td>
<td>1.017 ± 0.0421</td>
<td>1.463 ± 0.0863</td>
<td><strong>0.841 ± 0.0348</strong></td>
<td>1.038 ± 0.0596</td>
</tr>
<tr>
<td>power</td>
<td>9568</td>
<td>4</td>
<td>5.135 ± 0.0471</td>
<td><strong>4.977 ± 0.0396</strong></td>
<td><strong>4.978 ± 0.0443</strong></td>
<td>4.865 ± 0.0370</td>
</tr>
</tbody>
</table>

Competitive on small data, even more beneficial for larger N
Experiments: Regression
Does our method scale to large datasets and models?

Table 1: Final test RMSE on UCI regression datasets averaged over 40 (year: 5) seeds.

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>d</th>
<th>RANDOM</th>
<th>MaxEnt</th>
<th>ACS-FW</th>
<th>MaxEnt (greedy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yacht</td>
<td>308</td>
<td>6</td>
<td>1.342 ± 0.0404</td>
<td>0.991 ± 0.0423</td>
<td>1.033 ± 0.0374</td>
<td>0.940 ± 0.0355</td>
</tr>
<tr>
<td>boston</td>
<td>506</td>
<td>13</td>
<td>4.084 ± 0.0523</td>
<td>3.681 ± 0.0826</td>
<td>3.851 ± 0.0880</td>
<td>3.523 ± 0.0736</td>
</tr>
<tr>
<td>energy</td>
<td>768</td>
<td>8</td>
<td>1.017 ± 0.0421</td>
<td>1.463 ± 0.0863</td>
<td>0.841 ± 0.0348</td>
<td>1.038 ± 0.0596</td>
</tr>
<tr>
<td>power</td>
<td>9568</td>
<td>4</td>
<td>5.135 ± 0.0471</td>
<td>4.977 ± 0.0396</td>
<td>4.978 ± 0.0443</td>
<td>4.865 ± 0.0370</td>
</tr>
<tr>
<td>year</td>
<td>515345</td>
<td>90</td>
<td>13.130 ± 0.0463</td>
<td>12.924 ± 0.0521</td>
<td>12.205 ± 0.0286</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Experiments: Classification

Does our method scale to large datasets and models?

Enables efficient AL at scale, without any sacrifice in performance
Conclusion

Introduced **novel Bayesian batch AL approach**
- Based on sparse subset approximations
- Produces diverse batches, enabling efficient AL at scale
- Yields interpretable closed-form solutions
- Generalizes to arbitrary models using random projections

**Future Work**
- Leverage Frank-Wolfe weights in more principled way
- Investigate interactions with other approximate inference methods
- Apply to continual learning