Batch-Expansion Training: An Efficient Optimization Framework

UNIVERSITY OF CALIFORNIA SANTA CRUZ

Microsoft

Michał Dereziński Dhruv Mahajan S. Sathiya Keerthi S. V. N. Vishwanathan Markus Weimer

Large-scale convex optimization

Goal: minimize $f(\mathbf{w}) \triangleq \frac{1}{N}\sum_{i=1}^{N} \ell_i(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$

Challenges of large-scale optimization:

- Data does not fit on a single machine
- Disk-access is slow
- Not all machines are available immediately

Batch vs stochastic optimization

Batch methods:

- Slow iterations
- Need all data from the start
- Easy to parallelize
- No tuning needed

Stochastic methods:

- Quick iterations
- Adapt to gradual data availability
- Tricky to parallelize
- Require tuning

Batch-Expansion Training (BET)

Goal: Batch optimization without the tradeoffs
Idea: We do not need all data to start converging

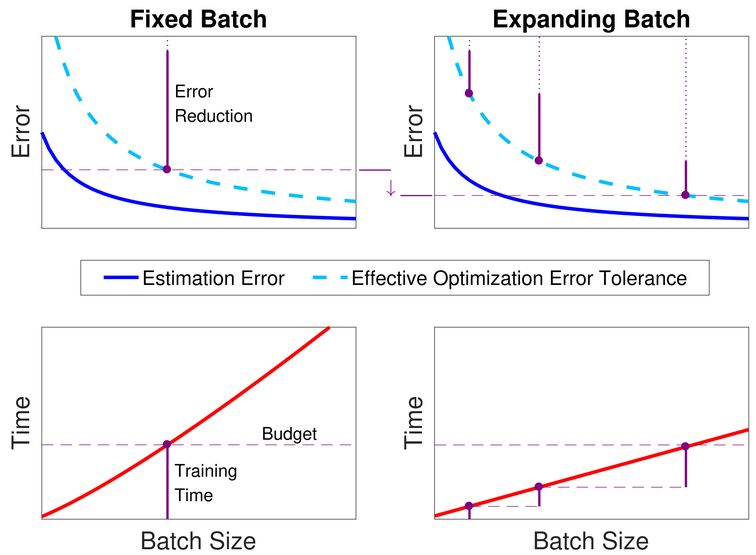
Pick initial model $\mathbf{w}_0$
Load first n_1 data points
for $t = 1..T$ **do**
  $\mathbf{w}_t \leftarrow$ run κ_t steps on n_t data points
  $n_{t+1} \leftarrow b_t n_t$ (increase data size)
  Load $n_{t+1} - n_t$ data points
end for
return $\mathbf{w}_T$

- Adapts to gradual data availability
- Easy to parallelize
- No tuning needed
- Works with any optimizer

1. How and when to increase the data size?
2. What are the theoretical guarantees?
3. Is BET effective in parallel and distributed settings?

Estimation error of expanding batches

Estimation Error: $\|\hat{f} - f\|$ ($\hat{f}$ is computed from a batch)



Improved data-access complexity

Linear optimizer: (for any $\mathbf{w}$ and $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$)

$$f(\mathrm{Update}(\mathbf{w})) - f(\mathbf{w}^*) \le (1 - (1/\kappa))(f(\mathbf{w}) - f(\mathbf{w}^*))$$

What is the cost of computing $\widehat{\mathbf{w}}$ such that $f(\widehat{\mathbf{w}}) - f(\mathbf{w}^*) \le \epsilon$?

Cost of batch optimization:

$$\overbrace{\text{data size}}^{N} \times \overbrace{\text{convergence rate}}^{\kappa} \times O(\log(1/\epsilon)) = O(N\,\kappa\,\log(1/\epsilon))$$

Optimal BET: Double the batch every $O(\kappa)$ iterations

Cost of optimal BET:

$$\underbrace{\overbrace{\text{1st batch}}^{n_1} \times \overbrace{\left(1 + 2 + 4 + \cdots + 2^{\log(N/n_1)}\right)}^{T \text{ stages}}}_{O(N)} \times \overbrace{\text{conv. rate}}^{O(\kappa)} = O(N\,\kappa)$$

Conclusion: BET saves a factor of $\log(1/\epsilon)$ over Batch!

Two-track algorithm

Question: When should we double the data size in practice?
Idea: Run two parallel optimization tracks with different data sizes

Stage t	Data	Loss	Model
Fast track	■■	$\hat{f}_{t-1}$	$\mathbf{w}_{\text{fast}}$
Slow track	■■■■	$\hat{f}_t$	$\mathbf{w}_{\text{slow}}$

When $\hat{f}_t(\mathbf{w}_{\text{slow}}) < \hat{f}_t(\mathbf{w}_{\text{fast}})$, double the data:

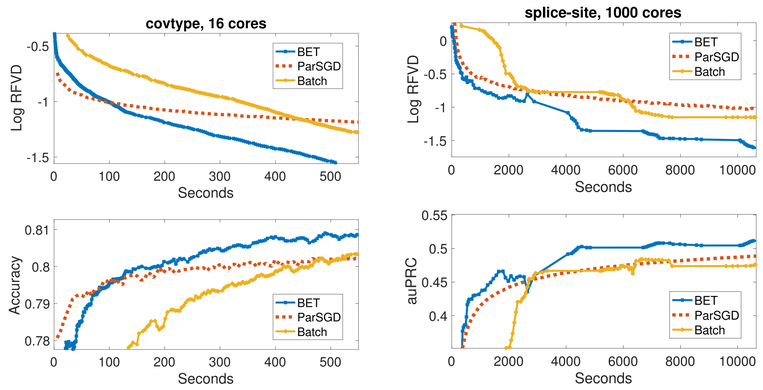
Stage $t+1$	Data	Loss	Model
Fast track	■■■■	$\hat{f}_t$	$\mathbf{w}_{\text{fast}}$
Slow track	■■■■■■■■	$\hat{f}_{t+1}$	$\mathbf{w}_{\text{slow}}$

...

until full dataset reached

BET in parallel and distributed settings

1. **BET** with Limited-memory BFGS as optimizer
2. **Batch** with Limited-memory BFGS as optimizer
3. **ParSGD** (Parallelized Stochastic Gradient Descent)



log RFVD: $\log\left((f(\mathbf{w}) - f(\mathbf{w}^*))/f(\mathbf{w}^*)\right)$

Dataset, size	Train/Test	Dim.	λ
covtype, 19GB	0.5M/69k	170k	1e-6
splice-site, 1.5TB	25M/4.6M	11.7M	2e-10