Optimization in the "Big Data" Regime 3: Tradeoffs in Large Scale Learning.

Sham M. Kakade

Machine Learning for Big Data
CSE547/STAT548

University of Washington

Announcements...

- Project milestones due Mon.
  - read/related work summary
  - some empirical work
- HW3 posted shortly.

Today:

- Review: SVRG
- New: Tradeoffs in large scale learning
  How do we optimize in the "big data" regime?

Review

Machine Learning and the Big Data Regime...

goal: find a d-dim parameter vector which minimizes the loss on n training examples.

- have n training examples $(x_1, y_1), \ldots (x_n, y_n)$
- have parametric a classifier $h(x, w)$, where w is a d dimensional vector.

$$\min_w L(w) \text{ where } L(w) = \sum_i \text{loss}(h(x_i, w), y_i)$$

- "Big Data Regime": How do you optimize this when n and d are large? memory? parallelization?

Can we obtain linear time algorithms to find an ϵ-accurate solution? i.e. find $\hat{w}$ so that

$$L(\hat{w}) - \min_w L(w) \leq \epsilon$$

Review: Stochastic Gradient Descent

- Suppose $L(w)$ is μ strongly convex.
- Suppose each loss $\mathrm{loss}(\cdot)$ is L-smooth
- To get ϵ accuracy:
  - # iterations to get ϵ-accuracy:

    $$\frac{L}{\mu\epsilon}$$

    (see related work for precise problem dependent parameters)
  - Computation time to get ϵ-accuracy:

    $$\frac{L}{\mu\epsilon}d$$

    (assuming $O(d)$ cost pre gradient evaluation.)

(another idea) Stochastic Variance Reduced Gradient (SVRG)

1. exact gradient computation: at stage s, using $\widetilde{w}_s$, compute:

$$\nabla L(\widetilde{w}_s) = \frac{1}{n} \sum_{i=1}^{n} \nabla \text{loss}(h(x_i, \widetilde{w}_s), y_i)$$

2. variance reduction + SGD: initialize $w \leftarrow \widetilde{w}_s$. for m steps,

   $m = 2/\mu$

   sample a point (x, y)

   $$w \leftarrow w - \eta \left(\nabla \text{loss}(h(x, w), y) - \nabla \text{loss}(h(x, \widetilde{w}_s), y) + \nabla L(\widetilde{w}_s) \right)$$

3. update and repeat: $\widetilde{w}_{s+1} \leftarrow w$.

Properties of SVRG

- unbiased updates: What is the mean of the blue term?

$$\mathbb{E}[\nabla \text{loss}(h(x, \widetilde{w}_s), y) - \nabla L(\widetilde{w}_s)] = ?$$

  where the expectation is for a random sample (x, y).
- If $\widetilde{w} = w_*$, then no update.
- Memory is $O(d)$.
- No “dual” variables.
  Applicable to non-convex optimization.

Guarantees of SVRG

- set $m = L/\mu$.
- # of gradient computations to get ϵ accuracy:

$$\left(n + \frac{L}{\mu}\right) \log 1/\epsilon$$

Comparisons

- a gradient evaluation is at a point (x, y).
  - SVRG: # of gradient computations to get ϵ accuracy:

SPCA

$$\left(n + \frac{L}{\mu}\right) \log 1/\epsilon$$

  - # of gradient evaluations for batch gradient descent:

$$n\frac{\tilde{L}}{\mu} \log 1/\epsilon$$

    where $\tilde{L}$ is the smoothness of $L(w)$.
  - # of gradient computations for SGD:

$$\frac{L}{\mu\epsilon}$$

Non-convex comparisons

- How many gradient evaluations does it take to find w so that:

$$\|\nabla L(w)\|^2 \leq \epsilon^2$$

  (i.e. ”close” to a stationary point)
- Rates: the number of gradient evaluations, at a point (x, y), is:
  - GD: $O(n/\epsilon)$
  - SGD: $O(1/\epsilon^2)$
  - SVRG: $O(n + n^{2/3}/\epsilon)$

Does SVRG work well in practice?

Tradeoffs in Large Scale Learning.

Tradeoffs in Large Scale Learning.

- Many issues sources of “error”
- approximation error: our choice of a hypothesis class
- estimation error: we only have n samples
- optimization error: computing exact (or near-exact) minimizers can be costly.
- How do we think about these issues?

The true objective

- hypothesis map $x \in \mathcal{X}$ to $y \in \mathcal{Y}$.
- have n training examples $(x_1, y_1), \ldots (x_n, y_n)$ sampled i.i.d. from $\mathcal{D}$.
- Training objective: have a set of parametric predictors $\{h(x, w) : w \in \mathcal{W}\}$,

$$\min_{w \in \mathcal{W}} \hat{L}_n(w) \text{ where } \hat{L}_n(w) = \frac{1}{n} \sum_{i=1}^{n} \text{loss}(h(x_i, w), y_i)$$

- True objective: to generalize to $\mathcal{D}$,

$$\min_{w \in \mathcal{W}} L(w) \text{ where } L(w) = \mathbb{E}_{(X,Y)\sim\mathcal{D}} \text{loss}(h(X, w), Y)$$

Optimization: Can we obtain linear time algorithms to find an ϵ-accurate solution? i.e. find $\hat{h}$ so that

$$L(\hat{w}) - \min_{w \in \mathcal{W}} L(w) \leq \epsilon$$

Definitions

- Let h^* is the *Bayes optimal hypothesis*, over all functions from $\mathcal{X} \to \mathcal{Y}$.
$$h^* \in \mathrm{argmin}_h L(h)$$
- Let w^* is the *best in class hypothesis*
$$w^* \in \mathrm{argmin}_{w \in \mathcal{W}} L(w)$$
- Let w_n be the *empirical risk minimizer:*
$$w_n \in \mathrm{argmin}_{w \in \mathcal{W}} \hat{L}_n(w)$$
- Let $\tilde{w}_n$ be what our algorithm returns.

Loss decomposition

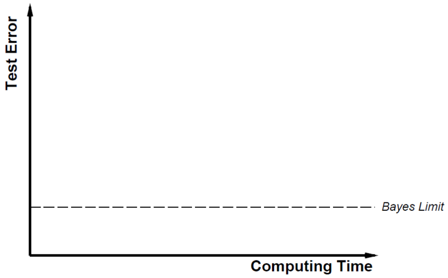
- Observe:

$$
\begin{aligned}
L(\tilde{w}_n) - L(h^*) = \quad & L(w^*) - L(h^*) && \text{Approximation error} \\
& + L(w_n) - L(w^*) && \text{Estimation error} \\
& + L(\tilde{w}_n) - L(w_n) && \text{Optimization error}
\end{aligned}
$$

- Three parts which determine our performance.
- Optimization algorithms with “best” accuracy dependencies on $\hat{L}_n$ may not be best.
  Forcing one error to decrease much faster may be wasteful.

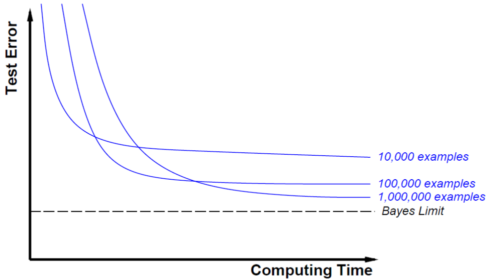
Time to a fixed accuracy

test error versus training time



Comparing sample sizes

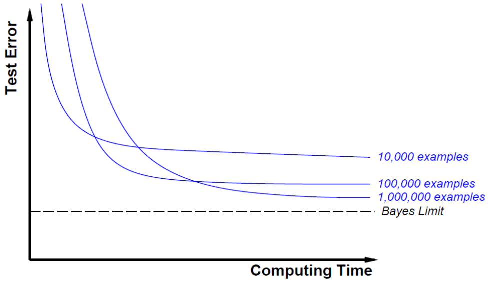
test error versus training time



- Vary the number of examples

Comparing sample sizes and models

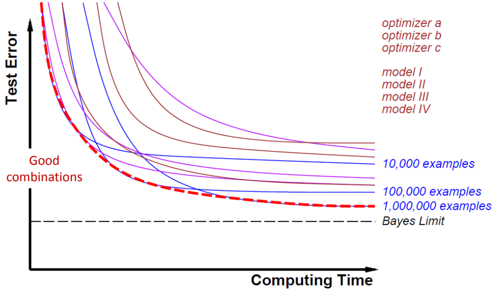
test error versus training time



- Vary the number of examples

Optimal choices

test error versus training time



- Optimal combination depends on training time budget.

Estimation error: simplest case

- Measuring a mean:

$$L(\mu) = \mathbb{E}(\mu - y)^2$$

$y \sim \mathcal{D}$

  The minima is at $\mu = \mathbb{E}[y]$.

- With n samples, the Bayes optimal estimator is the sample mean: $\hat{\mu}_n = \frac{1}{n}\sum_i y_i$.
- The error is:

$$\mathbb{E}[L(\hat{\mu}_n)] - L(\mathbb{E}[y]) = \frac{\sigma^2}{n}$$

  σ^2 is the variance and the expectation is with respect to the n samples. What do you do $n \approx \frac{\sigma^2}{\epsilon}$

- ~~How many samples do we need~~ for ϵ error?

Let's compare:

- SGD: Is $O(1/\epsilon)$ reasonable?
- GD: Is $\log 1/eps$ needed?
- SDCA/SVRG: These are also $\log 1/eps$ but much faster than GD (for large n).

Best in class error

- Fix a class $\mathcal{W}$. What is the best estimator of w^* for this model?
- For a wide class of models (linear regression, logistic regression, etc), the ERM, w_n, is (in the limit) the best estimator:

$$w_n \in \mathrm{argmin}_{w \in \mathcal{W}} \hat{L}_n(w)$$

1. What is the generalization error of best estimator w_n?
2. How well can we do? Note:

$$\begin{aligned} L(\tilde{w}_n) - L(w^*) = \quad & L(w_n) - L(w^*) && \text{Estimation error} \\ & + L(\tilde{w}_n) - L(w_n) && \text{Optimization error} \end{aligned}$$

- Can we generalize as well as the sample minimizer, w_n? (without computing it exactly)

Statistical Optimality

$y = w^* \cdot x + \eta \qquad \eta \sim \mathcal{N}(0, \sigma^2 I)$

- Can generalize as well as the sample minimizer, w_n? (without computing it exactly)
- For a wide class of models (linear regression, logistic regression, etc), we have that the estimation error is: $\sigma^2_{opt} = \text{dim.}\ \sigma^2$

$$\mathbb{E}[L(w_n)] - L(w^*) \overset{n \to \infty}{=} \frac{\sigma^2_{\text{opt}}}{n} = \frac{d\,\sigma^2}{n}$$

for consistent linear regression.

where σ^2_{opt} is an (optimal) problem dependent constant.

- This is the *best* possible statistical rate. (Can quantify the non-asymptotic "burn-in").
- What is the computational cost of achieving exactly this rate? say for large n?

Averaged SGD

- SGD:

$$w_{t+1} \leftarrow w_t - \eta_t \nabla \mathrm{loss}(h(x, w_t), y)$$

- An (asymptotically) optimal algo:
  - Have η_t go to 0 (sufficiently slowly)
  - (iterate averaging) Maintain the a running average:

$$\overline{w_n} = \frac{1}{n} \sum_{t \leq n} w_t$$

  - (Polyak & Juditsky, 1992) for large enough n and with ~~one pass~~ of SGD over the dataset:

$$\mathbb{E}[L(\overline{w_n})] - L(w^*) \overset{n \to \infty}{=} \frac{\sigma^2_{\mathrm{opt}}}{n}$$

for square loss, you can use a constant

Acknowledgements

Some slides from “Large-scale machine learning revisited”, Leon Bottou 2013.