Online vs. Offline Learning, Variance Reduction, and SVRG

CS6787 Lecture 5 — Fall 2017

Recall from Lecture 2

- Gradient descent
  - Computationally slow to run
  - Statistically converges at a linear rate

$$\mathbf{E}\left[\|x_t - x^*\|^2\right] = O(\gamma^t)$$

- Stochastic gradient descent (SGD)
  - Computationally fast iterations, **no dependence on dataset size**
  - Statistically **converges at a slower rate** — or to a noise ball

$$\mathbf{E}\left[\|x_t - x^*\|^2\right] = O(1/t)$$

Can We Do Better?

- Is there an algorithm that has the **computational structure of SGD**, but still gets the **fast linear rates of gradient descent**?

- Intermediate question: can we find problems for which vanilla SGD already converges at a linear rate, rather than converging to a noise ball?
  - If we find such a problem, we can **understand why it happens**.

Matrix Completion

- Suppose you have some **rank-1 matrix** $A = xx^T$

- Carelessly, **you lost most of the entries** of **A**
  - You only have access to a sparse, randomly-chosen subset of the entries

- Goal: **recover the original matrix** **A** from the sparse samples.
  - Applications include recommender systems, principle component analysis, etc.

Matrix Completion as Optimization

- Simplest thing: minimize squared error between model and samples.

$$\text{minimize}_x \sum_{(i,j)\in\text{samples}} (e_i^T x x^T e_j - e_i^T A e_j)^2$$

- Is this convex?

- We can try to solve this with SGD: randomly choose **(i, j)** and run

$$x_{t+1} = x_t - 2\alpha(e_i^T x x^T e_j - e_i^T A e_j)(e_i e_j^T x + e_j e_i^T x)$$

Aside: What is the cost of SGD here?

- Update rule is

$$x_{t+1} = x_t - 2\alpha(e_i^T x x^T e_j - e_i^T A e_j)(e_i e_j^T x + e_j e_i^T x)$$

- Suppose we have **K** samples and $x \in \mathbb{R}^n$.

- What is the **time complexity** of computing an iteration of SGD?

- It's really fast: **O(1)** — this makes SGD very attractive here

Demo

A Linear Rate for SGD? Why?

- Variance of the gradient estimator **goes to zero over time**.

- What is the variance at a particular point **x**?

$$\mathbf{E}\left[\left\|\tilde{\nabla} f(x)\right\|^2\right] = \frac{4}{K} \sum_{(i,j)\in\text{samples}} \left\|(e_i^T x x^T e_j - e_i^T A e_j)(e_i e_j^T x + e_j e_i^T x)\right\|^2$$

$$= \frac{4}{K} \sum_{(i,j)\in\text{samples}} (e_i^T x x^T e_j - e_i^T A e_j)^2((e_j^T x)^2 + (e_i^T x)^2)$$

- At an optimal point, $\mathbf{xx^T = A}$, the **variance is zero**!

The Role of Variance

- Hypothesis: if the variance becomes small when we get close to the optimum, we converge at a linear rate.

- In fact, we can prove that we get a linear rate if for some **C**

$$\mathbf{Var}\left(\nabla \tilde{f}(x)\right) \leq C \left\| x - x^* \right\|^2$$

  - Or more generally

$$\mathbf{E}\left[\left\| \nabla \tilde{f}(x) \right\|^2\right] \leq C \left\| \mathbf{E}\left[\nabla \tilde{f}(x)\right] \right\|^2 = C \left\| \nabla f(x) \right\|^2$$

Can we make this happen for any objective?

- One way to do it:

$$\nabla \tilde{g}(x) = \nabla \tilde{f}(x) - \nabla \tilde{f}(x^*)$$

- In expectation, this is the same since

$$\mathbf{E}\left[\nabla \tilde{g}(x)\right] = \nabla f(x) - \nabla f(x^*) = \nabla f(x) - 0$$

- And if the samples are Lipschitz continuous with parameter L,

$$\left\| \nabla \tilde{g}(x) \right\|^2 = \left\| \nabla f(x) - \nabla f(x^*) \right\|^2 \le L^2 \left\| x - x^* \right\|^2$$

Does this mean we can always get a linear rate?

- **Yes!** ...for any problem for which we already know the solution.

- **Doesn’t seem very useful**.

- What if we can **approximate the solution**? For $\hat{x} \approx x^*$

$$\nabla \tilde{g}(x) = \nabla \tilde{f}(x) - \nabla \tilde{f}(\hat{x})$$

  - But now our **gradients are biased** — SGD converges to $\hat{x}$ not x^*

Unbiased gradients with approximate solutions

- We can force the gradient to be unbiased by letting

$$\nabla \tilde{g}(x) = \nabla \tilde{f}(x) - \nabla \tilde{f}(\hat{x}) + \mathbf{E}\left[\nabla \tilde{f}(\hat{x})\right]$$

- Using a **full gradient as an anchor** to lower the variance

- But what is the **computational cost** of doing this?
  - Is it feasible to compute the full gradient in every setting?
  - Is it worth it to get a linear rate?

Online and Offline Learning

Two Types of Settings for ML Problems

- **Online learning**
  - The training examples arrive one-at-a-time as we are learning
  - We don't have access to all the training examples
  - Not even necessarily a finite training set — new training examples may be generated in real time in response to e.g. changes in the environment

- **Offline learning**
  - We have access to all the training examples upfront
  - The objective is a finite sum over the given training set

Online Learning

- Have some **distribution of training examples**, and goal is to

$$\text{minimize}_w \ \mathbf{E}_{\tilde{x} \sim \text{distribution}} \left[\text{loss}(w; \tilde{x}) \right]$$

- But we **don't actually have an expression** for the distribution

- All we can do is **draw samples from it**

$$\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \ldots$$

Advantages of Online Learning

- Online learning **generally doesn’t overfit**
  - Why? The training distribution is the same as the test distribution.

- Online learning **easily handles new data** from the environment

- Systems benefit: we don’t need to materialize the entire training set
  - Great for **scaling up** to problems that don’t fit in memory

Disadvantages of Online Learning

- **Can't compute exact/full objectives and gradients**
  - Because we don't even know distribution

- **Difficult to evaluate** convergence

- Generally **don't reuse training examples** multiple times
  - So don't make efficient use of the cache for the training set

- Neural networks sometimes **catastrophically forget older examples.**

Limitations on Online Learning

- 1-D least squares regression: for some distribution μ over **R**,

$$\text{minimize}_x \mathbf{E}_{u \sim \mu}\left[\frac{1}{2}(x-u)^2\right]$$

- Optimal solution is just the **mean**, regardless of what μ is

$$x^* = \mathbf{E}_{u \sim \mu}\left[u\right]$$

Limitations on Online Learning (continued)

- Suppose there were an online learning algorithm that converged at a linear rate for this 1-D least squares problem. Using **t** samples:

$$\mathbf{E}\left[(x_t - x^*)^2\right] = O(\gamma^t)$$

- But we know (from statistics) the lowest-variance estimator for the mean of a distribution, given **t** samples, is just the **sample mean**

$$\bar{u} = \frac{1}{t}\sum_{i=1}^{t} u_t \Rightarrow \mathbf{Var}\left(\bar{u}\right) = \frac{1}{t}\mathbf{Var}\left(u_t\right)$$

- **Contradiction.** No online algorithm can be this good!

Limitations on Online Learning (continued)

- Conclusion: there's **no online learning algorithm that converges at a linear rate** for general convex problems.

- This doesn't mean that online SGD never converges at a linear rate
    - We saw that the matrix completion example did

- But it does suggest that if we want to make SGD converge at a linear rate, **we need more information than what we have in the online setting**.

Aside: Online Learning in Research

- Online learning is an **active area of research**.

- Just from a search of the titles, there were **18 papers** mentioning online learning in this year's ICML and **28 papers** in this year's NIPS.
  - And many more if we look at the abstracts.

- Particularly interesting to us because of the **computational benefits** of being able to run online.

Offline Learning

- **Offline or batch learning** is the more traditional setting of minimizing a finite sum of training losses

$$\text{minimize}_w \ \frac{1}{n}\sum_{i=1}^{n} l(w; x_i, y_i)$$

  - Offline learning is often just defined as "not online learning"

- We have access to everything:
  - The loss function **l**
  - The training examples **x**
  - The training labels **y**

Benefits of Offline Learning

- **Can compute exact/full objectives and gradients**

- Consequence: it's trivially possible to **converge at a linear rate**
  - Just use gradient descent

- **Can we leverage this to make an SGD-like algorithm fast?**

Stochastic Variance-Reduced Gradient (SVRG)

Recall: Unbiased low-variance samples

- From a few slides ago, we were looking at using samples of the form

$$\nabla \tilde{g}(x) = \nabla \tilde{f}(x) - \nabla \tilde{f}(\hat{x}) + \mathbf{E}\left[\nabla \tilde{f}(\hat{x})\right]$$

- These samples have **reduced variance** when $\hat{x}$ is close to x^*

- We asked when we could do this, and now we have an answer:
  - **Only in the offline setting**!

- Question: **how do we use this in an algorithm?**

How much did we reduce the variance?

- If the gradient samples are **L**-Lipschitz continuous

$$
\begin{aligned}
\mathbf{Var}\left(\nabla \tilde{g}(x)\right) &= \mathbf{Var}\left(\nabla \tilde{f}(x) - \nabla \tilde{f}(\hat{x}) + \mathbf{E}\left[\nabla \tilde{f}(\hat{x})\right]\right) \\
&= \mathbf{Var}\left(\nabla \tilde{f}(x) - \nabla \tilde{f}(\hat{x})\right) \\
&\leq \mathbf{E}\left[\left\|\nabla \tilde{f}(x) - \nabla \tilde{f}(\hat{x})\right\|^2\right] \\
&\leq L^2 \left\|x - \hat{x}\right\|^2 .
\end{aligned}
$$

Is this enough for a linear rate for SGD?

- **No**, variance at the optimum is reduced, but still not zero!

$$\mathbf{Var}\left(\nabla \tilde{g}(x^*)\right) \leq L^2 \left\| x^* - \hat{x} \right\|^2.$$

- Idea: what if we used a sequence of $\hat{x}$ that approaches the optimum?

- Then **the variance would go to zero over time**!
  - Intuition: if the variance goes to zero at a linear rate, then SGD should also converge at a linear rate.

Is this enough? (continued)

- If we have a sequence of $\hat{x}$ that converges to the optimum at a linear rate, then we can use it to reduce the variance of SGD so that it converges to the optimum at a linear rate.

- **This also doesn't seem useful**.

- Critical insight: **use the iterates of SGD** as $\hat{x}$
  - So, if SGD converges at a linear rate, then SGD will converge at a linear rate
  - Seems circular — but we can make it rigorous

How often to use full gradient samples?

- **Can we use every iteration of SGD as an anchor point $\hat{x}$?**

- We could…but this would **just be gradient descent**.

$$\begin{aligned} \nabla \tilde{g}(x) &= \nabla \tilde{f}(x) - \nabla \tilde{f}(x) + \mathbf{E}\left[\nabla \tilde{f}(x)\right] \\ &= \nabla f(x). \end{aligned}$$

- Instead, use a full gradient sample every **K** iterations of SGD.
  - Called an **epoch**.

Stochastic Variance-Reduced Gradient (SVRG)

- Initialize $\mathbf{x_{0,T}}$ arbitrarily
- Outer loop: **for k = 1 to K**

$$\hat{x}_k \leftarrow x_{k-1,T}$$

$$\hat{g}_k \leftarrow \nabla f(\hat{x}_k) = \mathbf{E}\left[\nabla \tilde{f}(\hat{x}_k)\right]$$

$$x_{k,0} \leftarrow \hat{x}_k$$

  - Inner loop: **for t = 1 to T**
    - Sample $\mathbf{f_{k,t}}$ at random from training set losses

$$x_{k,t} \leftarrow x_{k,t-1} - \alpha\left(\nabla \tilde{f}_{k,t}(x_{k,t-1}) - \nabla \tilde{f}_{k,t}(\hat{x}_k) + \hat{g}_k\right)$$

Computational Cost of SVRG

- Each inner loop runs for **T** iterations
    - Has a computational cost of **O(T)**

- If we have **n** examples, the outer loop gradient computation has a computational cost of **O(n)**

- Over **K** total outer loop iterations, total time is **O(Kn + KT)**

Memory Burden of SVRG

- In addition to the copy of the model that needs to be stored for vanilla SGD, we also need to store
  - An additional copy of the model vector for the anchor point $\hat{x}$
  - An additional vector to store its exact/full gradient

- If the model is of size **d**, we will need to store a total of **3d** numbers
  - Plus the training set, which is usually much larger

- Takeaway: **no significant memory cost to run SVRG**

Linear Rates for SVRG

Very Simple Proof that SVRG Converges

- Strategy: run the inner loop of SVRG long enough that for some $\boldsymbol{\gamma} < 1$

$$\mathbf{E}\left[\left\|x_{k,T} - x^*\right\|^2 \middle| x_{k,0}\right] \le \gamma \left\|x_{k,0} - x^*\right\|^2.$$

- Show that a fixed **T** suffices for every epoch **k**
  - This is enough to show convergence at a linear rate. **Why?**

- You'll see a tighter version of this proof in **this week's paper**.

Analysis of an Inner Iterate of SVRG

- Starting with the iterate:

$$x_{k,t} = x_{k,t-1} - \alpha \left(\nabla \tilde{f}_{k,t}(x_{k,t-1}) - \nabla \tilde{f}_{k,t}(\hat{x}_k) + \hat{g}_k \right)$$

- Let's simplify it a little by abusing notation to drop the **k** subscripts

$$x_t = x_{t-1} - \alpha \left(\nabla \tilde{f}_t(x_{t-1}) - \nabla \tilde{f}_t(\hat{x}) + \nabla f(\hat{x}) \right)$$

Analysis (continued)

- Expected distance to the optimum:

$$\begin{aligned}\mathbf{E}\left[\|x_t - x^*\|^2 \middle| x_{t-1}\right] &= \mathbf{E}\left[\left\|x_{t-1} - x^* - \alpha\left(\nabla\tilde{f}_t(x_{t-1}) - \nabla\tilde{f}_t(\hat{x}) + \nabla f(\hat{x})\right)\right\|^2 \middle| x_{t-1}\right] \\ &= \|x_{t-1} - x^*\|^2 - 2\alpha(x_{t-1} - x^*)^T \mathbf{E}\left[\nabla\tilde{f}_t(x_{t-1}) - \nabla\tilde{f}_t(\hat{x}) + \nabla f(\hat{x}) \middle| x_{t-1}\right] \\ &\quad + \alpha^2 \mathbf{E}\left[\left\|\nabla\tilde{f}_t(x_{t-1}) - \nabla\tilde{f}_t(\hat{x}) + \nabla f(\hat{x})\right\|^2 \middle| x_{t-1}\right] \\ &= \|x_{t-1} - x^*\|^2 - 2\alpha(x_{t-1} - x^*)^T \nabla f(x_{t-1}) + \alpha^2 \|\nabla f(x_{t-1})\|^2 \\ &\quad + \alpha^2 \mathbf{Var}\left(\nabla\tilde{f}_t(x_{t-1}) - \nabla\tilde{f}_t(\hat{x}) + \nabla f(\hat{x}) \middle| x_{t-1}\right)\end{aligned}$$

- To proceed, need to **bound the second order/variance term**

Analysis (continued)

- Important property: for constant **c**, $\mathbf{Var}\left(X + c\right) = \mathbf{Var}\left(X\right)$

- We can use this to simplify the second order term:

$$\begin{aligned}
&\mathbf{Var}\left(\nabla \tilde{f}_t(x_{t-1}) - \nabla \tilde{f}_t(\hat{x}) + \nabla f(\hat{x}) \middle| x_{t-1}\right) \\
&\quad = \mathbf{Var}\left(\nabla \tilde{f}_t(x_{t-1}) - \nabla \tilde{f}_t(\hat{x}) \middle| x_{t-1}\right) \\
&\quad \leq \mathbf{E}\left[\left\| \nabla \tilde{f}_t(x_{t-1}) - \nabla \tilde{f}_t(\hat{x}) \right\|^2 \middle| x_{t-1}\right] \\
&\quad \leq L^2 \left\| x_{t-1} - \hat{x} \right\|^2 \leq 2L^2 \left\| x_{t-1} - x^* \right\|^2 + 2L^2 \left\| \hat{x} - x^* \right\|^2
\end{aligned}$$

Analysis (continued)

- Substituting this back, we get

$$\mathbf{E}\left[\|x_t - x^*\|^2 \middle| x_{t-1}\right] \leq \|x_{t-1} - x^*\|^2 - 2\alpha(x_{t-1} - x^*)^T \nabla f(x_{t-1}) + \alpha^2 \|\nabla f(x_{t-1})\|^2$$
$$+ \alpha^2 \left(2L^2 \|x_{t-1} - x^*\|^2 + 2L^2 \|\hat{x} - x^*\|^2\right)$$

- Now we can reduce the first part using **strong convexity/Lipschitz**

$$\mathbf{E}\left[\|x_t - x^*\|^2 \middle| x_{t-1}\right] \leq \|x_{t-1} - x^*\|^2 - 2\alpha\mu \|x_{t-1} - x^*\|^2 + \alpha^2 L^2 \|x_{t-1} - x^*\|^2$$
$$+ \alpha^2 \left(2L^2 \|x_{t-1} - x^*\|^2 + 2L^2 \|\hat{x} - x^*\|^2\right)$$

Analysis (continued)

- We can now take the full expectation, given the anchor point

$$\mathbf{E}\left[\left\|x_t - x^*\right\|^2 \middle| \hat{x}\right] \leq (1 - 2\alpha\mu + 3\alpha^2 L^2)\mathbf{E}\left[\left\|x_{t-1} - x^*\right\|^2 \middle| \hat{x}\right] + 2\alpha^2 L^2 \left\|\hat{x} - x^*\right\|^2$$

- Next, for simplicity, let $\rho_t = \mathbf{E}\left[\left\|x_t - x^*\right\|^2 \middle| \hat{x}\right]$

$$\rho_t \leq (1 - 2\alpha\mu + 3\alpha^2 L^2)\rho_{t-1} + 2\alpha^2 L^2 \rho_0$$

- Suppose we want to contract by a factor of **e**. As long as **e p$_{\mathbf{t-1}}$ > p$_{\mathbf{0}}$:**

$$\rho_t \leq (1 - 2\alpha\mu + 3\alpha^2 L^2)\rho_{t-1} + 2\alpha^2 L^2 e \rho_{t-1}$$

Analysis (continued)

- Now we have

$$\begin{aligned}\rho_t &\leq (1 - 2\alpha\mu + 3\alpha^2 L^2)\rho_{t-1} + 2\alpha^2 L^2 e \rho_{t-1} \\ &\leq (1 - 2\alpha\mu + 5\alpha^2 L^2 e)\rho_{t-1}\end{aligned}$$

- Setting the step size such that $\alpha\mu = 5\alpha^2 L^2 e$

$$\rho_t \leq \left(1 - \frac{\mu^2}{5L^2 e}\right)\rho_{t-1} \Rightarrow \rho_t \leq \left(1 - \frac{\mu^2}{5L^2 e}\right)^t \rho_0 \leq \exp\left(-\frac{\mu^2}{5L^2 e} t\right)^t \rho_0$$

Analysis (continued)

- Now, this was all contingent upon **e** $\mathbf{p_{t-1} > p_0}$.

- **How many iterations** do we need to get there?
  - Need **t** such that

$$\exp\left(-\frac{\mu^2}{5L^2e}t\right) \leq \frac{1}{e}$$

- It suffices to pick any

$$t \geq \frac{5L^2e}{\mu^2}$$

Analysis of Inner Loop Is Done!

- We've shown that if we run for $t \geq \frac{5L^2 e}{\mu^2}$ iterations,

$$\mathbf{E}\left[\left\|x_t - x^*\right\|^2 \middle| \hat{x}\right] \leq \frac{1}{e}\left\|\hat{x} - x^*\right\|^2$$

- In particular, this means that across outer loop iterations,

$$\mathbf{E}\left[\left\|\hat{x}_{k+1} - x^*\right\|^2\right] \leq \frac{1}{e}\mathbf{E}\left[\left\|\hat{x}_k - x^*\right\|^2\right]$$

Outer Loop Analysis

- Applying this recursively,

$$\mathbf{E}\left[\left\|\hat{x}_k - x^*\right\|^2\right] \le e^{-k}\left\|\hat{x}_0 - x^*\right\|^2$$

- So, to get down to error ε we need **k** iterations, where

$$k \ge \log\left(\frac{\left\|\hat{x}_0 - x^*\right\|^2}{\epsilon}\right)$$

Bringing it Together

- Total number of stochastic gradient iterations needed is

$$tk \geq \frac{5L^2 e}{\mu^2} \log\left(\frac{\|\hat{x}_0 - x^*\|^2}{\epsilon}\right) = O\left(\log\left(\frac{1}{\epsilon}\right)\right)$$

- **This is a linear rate!**

Demo

Issues with Variance Reduction

- Computational cost

- Overfitting

- Interaction with other techniques

- Choosing parameters
  - **Metaparameter optimization**

Other Methods for Variance Reduction

SAG

- Stochastic average gradient

- At each step, randomly update a single example's gradient estimate using the current iterate, like SGD

- But, use the **sum of all gradient estimates** to perform an update

Systems Comparison: SAG vs SVRG

- SAG requires us to store a gradient sample **for each training example**

- What is the memory cost of doing this, if we have **n** training examples and our model has dimension **d**?

- Answer: it's **O(nd)**

- Compare to SVRG which required **O(3d)**

Many other variance reduction methods

- SAG
- SAGA
- SVRG
- SDCA – stochastic dual coordinate ascent
- Etc.

Questions?

- Upcoming things
  - Paper Presentation #4 **on Wednesday** — read paper before class
  - **Paper Review #3 due Today.**