CS60021: Scalable Data Mining

Large Scale Machine Learning

Sourangshu Bhattacharya

THEORETICAL GUARRANTEES

Convergence Rate and Assumptions

A sequence $\{x^k\}$ is said to converge at the rate γ^k, if:

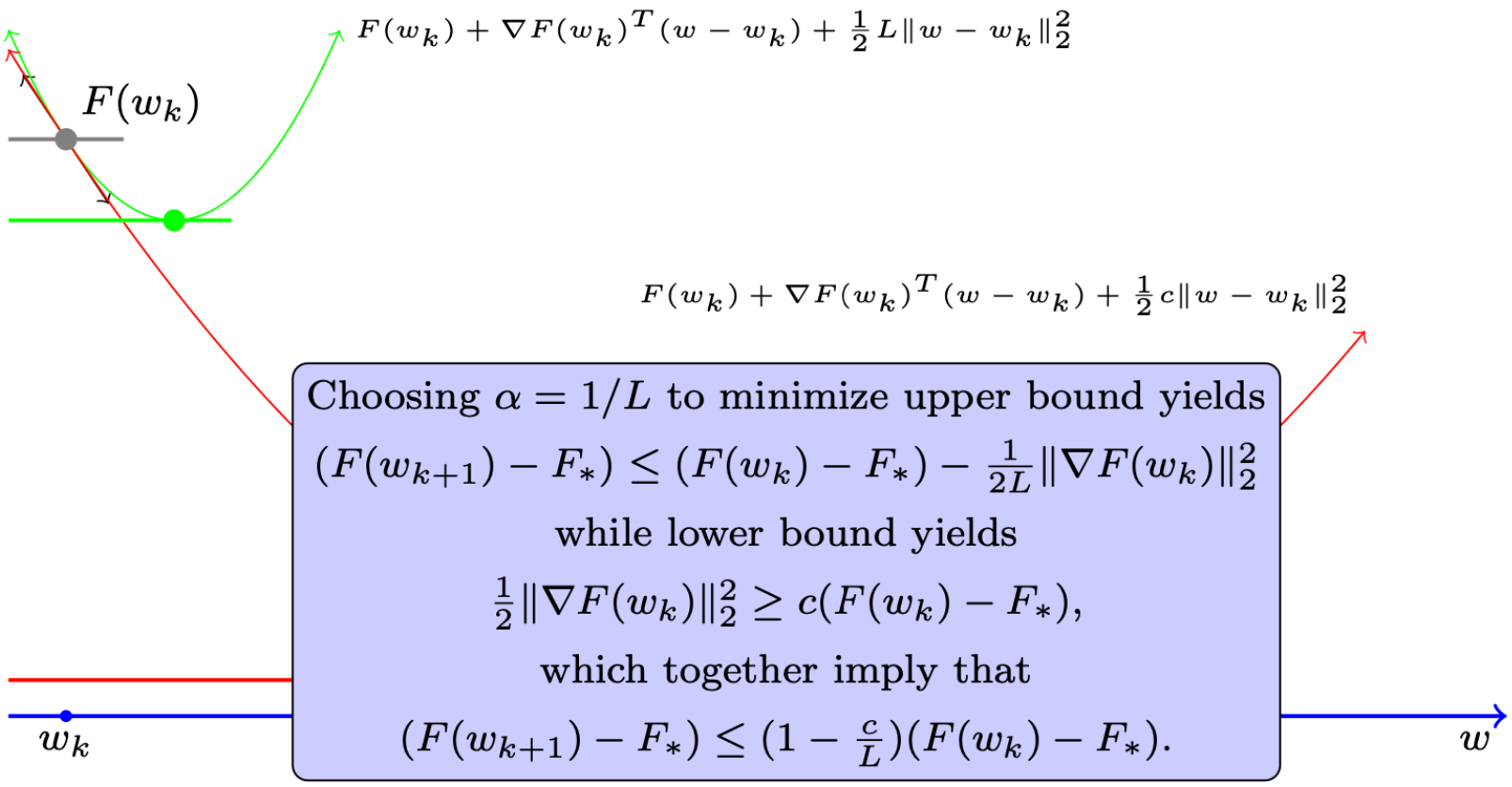
$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\| \leq \gamma \|\mathbf{x}^k - \mathbf{x}^*\| \quad (\Rightarrow \|\mathbf{x}^k - \mathbf{x}^*\| \leq \gamma^k \|\mathbf{x}^0 - \mathbf{x}^*\|),$$

Assumption $\langle L/c \rangle$

The objective function $F : \mathbb{R}^d \to \mathbb{R}$ is

- *c-strongly convex ($\Rightarrow$ unique minimizer) and*
- *L-smooth (i.e., ∇F is Lipschitz continuous with constant L).*

Gradient Descent Convergence



Convergence Rate and Computational Complexity

Overall Complexity (ϵ) = Convergence Rate^{-1}(ϵ) $*$ Complexity of each iteration

	Strongly Convex + Smooth			Convex + Smooth		
	Convergence Rate	Complexity of each iteration	Overall Complexity	Convergence Rate	Complexity of each iteration	Overall Complexity
GD	$O\left(\exp\left(-\frac{t}{Q}\right)\right)$	$O(n \cdot d)$	$O\left(nd \cdot Q \cdot \log\left(\frac{1}{\epsilon}\right)\right)$	$O\left(\frac{\beta}{t}\right)$	$O(n \cdot d)$	$O\left(nd \cdot \beta \cdot \left(\frac{1}{\epsilon}\right)\right)$
SGD	$O\left(\frac{1}{t}\right)$	$O(d)$	$O\left(\frac{d}{\epsilon}\right)$	$O\left(\frac{1}{\sqrt{t}}\right)$	$O(d)$	$O\left(\frac{d}{\epsilon^2}\right)$

SGD Analysis

THEOREM 14.8 *Let $B, \rho > 0$. Let f be a convex function and let $\mathbf{w}^\star \in \operatorname{argmin}_{\mathbf{w}:\|\mathbf{w}\|\leq B} f(\mathbf{w})$. Assume that SGD is run for T iterations with $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$. Assume also that for all t, $\|\mathbf{v}_t\| \leq \rho$ with probability 1. Then,*

$$\mathbb{E}\left[f(\bar{\mathbf{w}})\right] - f(\mathbf{w}^\star) \leq \frac{B\rho}{\sqrt{T}}.$$

Therefore, for any $\epsilon > 0$, to achieve $\mathbb{E}[f(\bar{\mathbf{w}})] - f(\mathbf{w}^\star) \leq \epsilon$, it suffices to run the SGD algorithm for a number of iterations that satisfies

$$T \geq \frac{B^2\rho^2}{\epsilon^2}.$$

SGD Analysis

LEMMA 14.1 *Let $\mathbf{v}_1, \ldots, \mathbf{v}_T$ be an arbitrary sequence of vectors. Any algorithm with an initialization $\mathbf{w}^{(1)} = 0$ and an update rule of the form*

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \mathbf{v}_t \tag{14.4}$$

satisfies

$$\sum_{t=1}^{T} \langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t \rangle \leq \frac{\|\mathbf{w}^\star\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|\mathbf{v}_t\|^2. \tag{14.5}$$

In particular, for every $B, \rho > 0$, if for all t we have that $\|\mathbf{v}_t\| \leq \rho$ and if we set $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$, then for every $\mathbf{w}^\star$ with $\|\mathbf{w}^\star\| \leq B$ we have

$$\frac{1}{T} \sum_{t=1}^{T} \langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t \rangle \leq \frac{B\rho}{\sqrt{T}}.$$

SGD Analysis

Proof Using algebraic manipulations (completing the square), we obtain:

$$\begin{aligned}\langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t \rangle &= \frac{1}{\eta} \langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \eta \mathbf{v}_t \rangle \\ &= \frac{1}{2\eta} (-\|\mathbf{w}^{(t)} - \mathbf{w}^\star - \eta \mathbf{v}_t\|^2 + \|\mathbf{w}^{(t)} - \mathbf{w}^\star\|^2 + \eta^2 \|\mathbf{v}_t\|^2) \\ &= \frac{1}{2\eta} (-\|\mathbf{w}^{(t+1)} - \mathbf{w}^\star\|^2 + \|\mathbf{w}^{(t)} - \mathbf{w}^\star\|^2) + \frac{\eta}{2} \|\mathbf{v}_t\|^2,\end{aligned}$$

SGD Analysis

where the last equality follows from the definition of the update rule. Summing the equality over t, we have

$$\sum_{t=1}^{T} \langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t \rangle = \frac{1}{2\eta} \sum_{t=1}^{T} \left(-\|\mathbf{w}^{(t+1)} - \mathbf{w}^\star\|^2 + \|\mathbf{w}^{(t)} - \mathbf{w}^\star\|^2 \right) + \frac{\eta}{2} \sum_{t=1}^{T} \|\mathbf{v}_t\|^2. \tag{14.6}$$

The first sum on the right-hand side is a telescopic sum that collapses to

$$\|\mathbf{w}^{(1)} - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(T+1)} - \mathbf{w}^\star\|^2.$$

SGD Analysis

Plugging this in Equation (14.6), we have

$$\begin{aligned}\sum_{t=1}^{T}\langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t\rangle &= \frac{1}{2\eta}(\|\mathbf{w}^{(1)} - \mathbf{w}^\star\|^2 - \|\mathbf{w}^{(T+1)} - \mathbf{w}^\star\|^2) + \frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2 \\ &\leq \frac{1}{2\eta}\|\mathbf{w}^{(1)} - \mathbf{w}^\star\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2 \\ &= \frac{1}{2\eta}\|\mathbf{w}^\star\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{v}_t\|^2,\end{aligned}$$

where the last equality is due to the definition $\mathbf{w}^{(1)} = 0$. This proves the first part of the lemma (Equation (14.5)). The second part follows by upper bounding $\|\mathbf{w}^\star\|$ by B, $\|\mathbf{v}_t\|$ by ρ, dividing by T, and plugging in the value of η. □

SGD Analysis

- Proof of theorem:

$$\mathop{\mathbb{E}}_{\mathbf{v}_{1:T}}[f(\bar{\mathbf{w}}) - f(\mathbf{w}^\star)] \leq \mathop{\mathbb{E}}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T}(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^\star))\right].$$

Since Lemma 14.1 holds for any sequence $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_T$, it applies to SGD as well. By taking expectation of the bound in the lemma we have

$$\mathop{\mathbb{E}}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t\rangle\right] \leq \frac{B\,\rho}{\sqrt{T}}. \tag{14.9}$$

It is left to show that

$$\mathop{\mathbb{E}}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T}(f(\mathbf{w}^{(t)}) - f(\mathbf{w}^\star))\right] \leq \mathop{\mathbb{E}}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t\rangle\right], \tag{14.10}$$

SGD Analysis

Using the linearity of the expectation we have

$$\mathop{\mathbb{E}}_{\mathbf{v}_{1:T}}\left[\frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{w}^{(t)}-\mathbf{w}^{\star},\mathbf{v}_t\rangle\right]=\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{\mathbf{v}_{1:T}}[\langle \mathbf{w}^{(t)}-\mathbf{w}^{\star},\mathbf{v}_t\rangle].$$

Next, we recall the *law of total expectation*: For every two random variables α, β, and a function g, $\mathbb{E}_{\alpha}[g(\alpha)] = \mathbb{E}_{\beta}\,\mathbb{E}_{\alpha}[g(\alpha)|\beta]$. Setting $\alpha = \mathbf{v}_{1:t}$ and $\beta = \mathbf{v}_{1:t-1}$ we get that

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathbf{v}_{1:T}}[\langle \mathbf{w}^{(t)}-\mathbf{w}^{\star},\mathbf{v}_t\rangle] &= \mathop{\mathbb{E}}_{\mathbf{v}_{1:t}}[\langle \mathbf{w}^{(t)}-\mathbf{w}^{\star},\mathbf{v}_t\rangle] \\ &= \mathop{\mathbb{E}}_{\mathbf{v}_{1:t-1}}\mathop{\mathbb{E}}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)}-\mathbf{w}^{\star},\mathbf{v}_t\rangle \,\middle|\, \mathbf{v}_{1:t-1}\right].\end{aligned}$$

Once we know $\mathbf{v}_{1:t-1}$, the value of $\mathbf{w}^{(t)}$ is not random any more and therefore

$$\mathop{\mathbb{E}}_{\mathbf{v}_{1:t-1}}\mathop{\mathbb{E}}_{\mathbf{v}_{1:t}}\left[\langle \mathbf{w}^{(t)}-\mathbf{w}^{\star},\mathbf{v}_t\rangle \,\middle|\, \mathbf{v}_{1:t-1}\right]=\mathop{\mathbb{E}}_{\mathbf{v}_{1:t-1}}\left\langle \mathbf{w}^{(t)}-\mathbf{w}^{\star},\mathop{\mathbb{E}}_{\mathbf{v}_t}[\mathbf{v}_t \,|\, \mathbf{v}_{1:t-1}]\right\rangle .$$

SGD Analysis

Since $\mathbf{w}^{(t)}$ only depends on $\mathbf{v}_{1:t-1}$ and SGD requires that $\mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t \,|\, \mathbf{w}^{(t)}] \in \partial f(\mathbf{w}^{(t)})$ we obtain that $\mathbb{E}_{\mathbf{v}_t}[\mathbf{v}_t \,|\, \mathbf{v}_{1:t-1}] \in \partial f(\mathbf{w}^{(t)})$. Thus,

$$\mathop{\mathbb{E}}_{\mathbf{v}_{1:\,t-1}} \langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathop{\mathbb{E}}_{\mathbf{v}_t}[\mathbf{v}_t \,|\, \mathbf{v}_{1:\,t-1}] \rangle \;\geq\; \mathop{\mathbb{E}}_{\mathbf{v}_{1:\,t-1}} [f(\mathbf{w}^{(t)}) - f(\mathbf{w}^\star)].$$

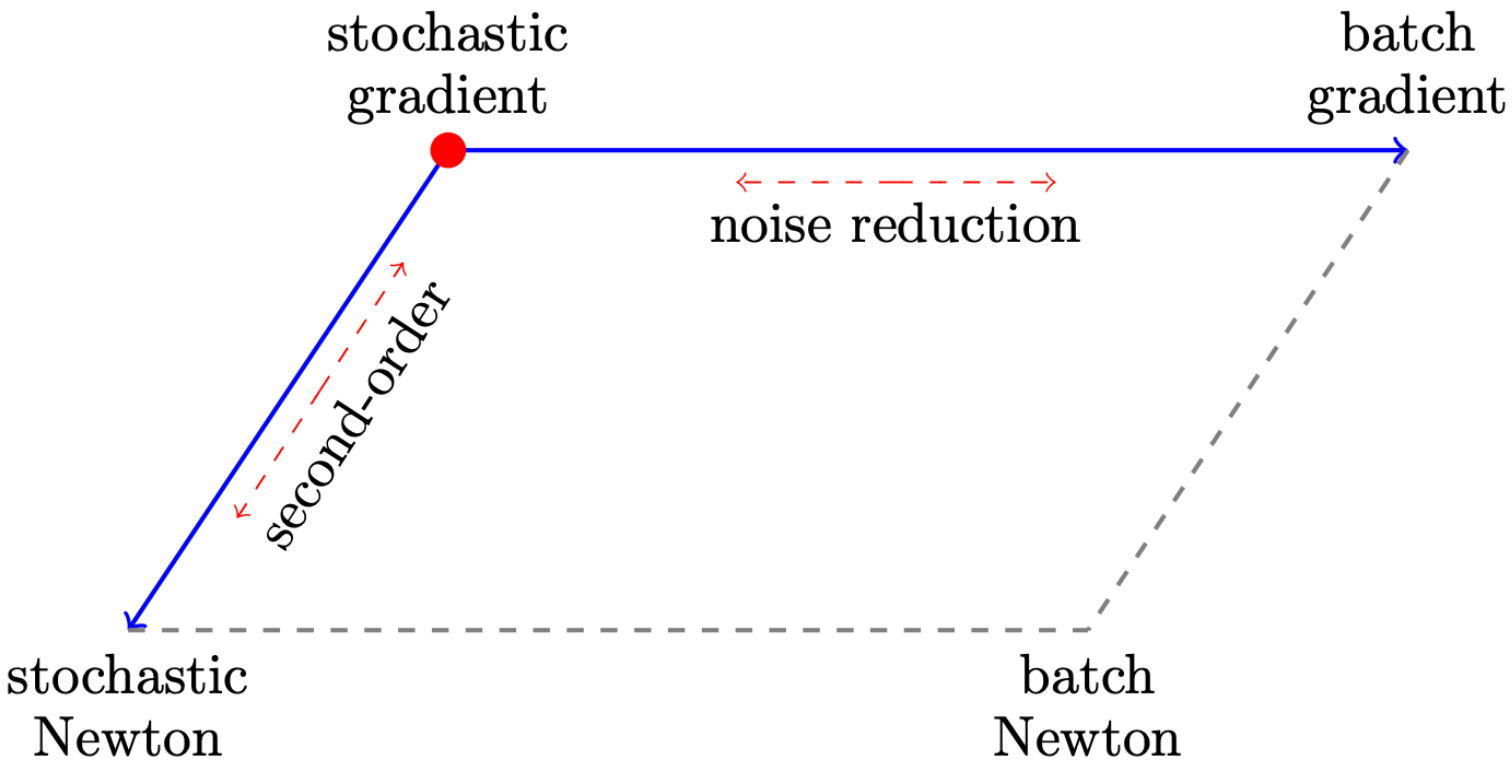
Overall, we have shown that

$$\begin{aligned} \mathop{\mathbb{E}}_{\mathbf{v}_{1:T}} [\langle \mathbf{w}^{(t)} - \mathbf{w}^\star, \mathbf{v}_t \rangle] &\geq \mathop{\mathbb{E}}_{\mathbf{v}_{1:t-1}} [f(\mathbf{w}^{(t)}) - f(\mathbf{w}^\star)] \\ &= \mathop{\mathbb{E}}_{\mathbf{v}_{1:T}} [f(\mathbf{w}^{(t)}) - f(\mathbf{w}^\star)] \,. \end{aligned}$$

Summing over t, dividing by T, and using the linearity of expectation, we get that Equation (14.10) holds, which concludes our proof. □

LINEAR RATE METHODS

Improving SGD



Slides taken from Jorge Nocedal

Stochastic Averaged Gradient

- **Can we have a rate of $O(\rho^t)$ with only 1 gradient evaluation per iteration?**
  - YES! The stochastic average gradient (SAG) algorithm:
    - Randomly select i_t from $\{1, 2, \ldots, N\}$ and compute $f'_{i_t}(x^t)$.

$$x^{t+1} = x^t - \frac{\alpha^t}{N} \sum_{i=1}^{N} y_i^t$$

    - **Memory**: $y_i^t = \nabla f_i(x^t)$ from the last t where i was selected.
      [Le Roux et al., 2012]
  - Stochastic variant of increment average gradient (IAG).
    [Blatt et al., 2007]
  - Assumes gradients of non-selected examples don't change.
  - Assumption becomes accurate as $||x^{t+1} - x^t|| \to 0$.

Slides taken from Mark Schmidt

SAG Convergence Rate

- If each f_i' is $L-$continuous and f is strongly-convex, with $\alpha_t = 1/16L$ SAG has

$$\mathbb{E}[f(x^t) - f(x^*)] \leqslant \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^t C,$$

where

$$C = [f(x^0) - f(x^*)] + \frac{4L}{N}\|x^0 - x^*\|^2 + \frac{\sigma^2}{16L}.$$

- Linear convergence rate but only 1 gradient per iteration.
  - For well-conditioned problems, constant reduction per pass:

$$\left(1 - \frac{1}{8N}\right)^N \leq \exp\left(-\frac{1}{8}\right) = 0.8825.$$

  - For ill-conditioned problems, almost same as deterministic method (but N times faster).

SAG Convergence Rate

- Assume that $N = 700000$, $L = 0.25$, $\mu = 1/N$:
  - Gradient method has rate $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$.
  - Accelerated gradient method has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.99761$.
  - SAG (N iterations) has rate $\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^N = 0.88250$.
  - *Fastest possible* first-order method: $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2 = 0.99048$.
- SAG beats two lower bounds:
  - Stochastic gradient bound (of $O(1/t)$).
  - Deterministic gradient bound (for typical L, μ, and N).
- Number of f_i' evaluations to reach ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$.
  - Gradient: $O(N\frac{L}{\mu}\log(1/\epsilon))$.
  - Accelerated: $O(N\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$.
  - SAG: $O(\max\{N, \frac{L}{\mu}\}\log(1/\epsilon))$.

SAG Implementation

- Basic SAG algorithm:
  - while(1)
  - Sample i from $\{1, 2, \ldots, N\}$.
  - Compute $f_i'(x)$.
  - $d = d - y_i + f_i'(x)$.
  - $y_i = f_i'(x)$.
  - $x = x - \frac{\alpha}{N} d$.
- Practical variants of the basic algorithm allow:
  - Regularization.
  - Sparse gradients.
  - Automatic step-size selection.
    - Common to use adaptive step-size procedure to estimate L.
  - Termination criterion.
    - Can use $\|x^{t+1} - x^t\|/\alpha = \frac{1}{n} d \approx \|\nabla f(x^t)\|$ to decide when to stop.
  - Acceleration [Lin et al., 2015].
  - Adaptive non-uniform sampling [Schmidt et al., 2013].

SAG Implementation

- Does re-shuffling and doing full passes work better?
  - For classic SG: Maybe?
    - Noncommutative arithmetic-geometric mean inequality conjecture.

      [Recht & Ré, 2012]
  - For SAG: NO.
  - Performance is intermediate between IAG and SAG.
- Can non-uniform sampling help?
  - For classic SG methods, can only improve constants.
  - For SAG, bias sampling towards Lipschitz constants L_i,

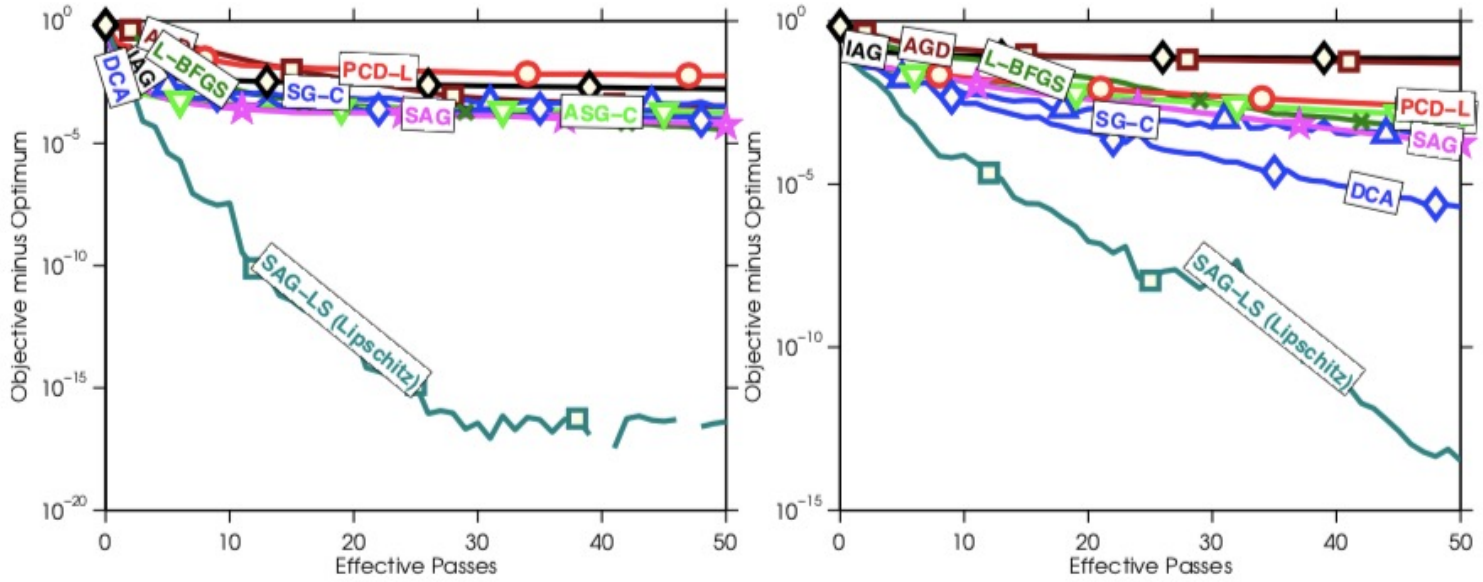
    $$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i \|x - y\|.$$

    improves rate to depend on L_{mean} instead of L_{max}.

    (with bigger step size)
  - Adaptively estimate L_i as you go. (see paper/code).
  - Slowly learns to ignore well-classified examples.

SAG with Non-Uniform Sampling

- protein ($n = 145751$, $p = 74$) and sido ($n = 12678$, $p = 4932$)



- Adaptive non-uniform sampling helps a lot.

Stochastic Variance Reduced GD

SVRG algorithm:

- Start with x_0
- for $s = 0, 1, 2 \ldots$
  - $d_s = \frac{1}{N} \sum_{i=1}^{N} f_i'(x_s)$
  - $x^0 = x_s$
  - for $t = 1, 2, \ldots m$
    - Randomly pick $i_t \in \{1, 2, \ldots, N\}$
    - $x^t = x^{t-1} - \alpha_t (f_{i_t}'(x^{t-1}) - f_{i_t}'(x_s) + d_s)$.
  - $x_{s+1} = x^t$ for random $t \in \{1, 2, \ldots, m\}$.

Requires 2 gradients per iteration and occasional full passes,
but only requires storing d_s and x_s.

Practical issues similar to SAG (acceleration versions, automatic step-size/termination, handles sparsity/regularization, non-uniform sampling, mini-batches).

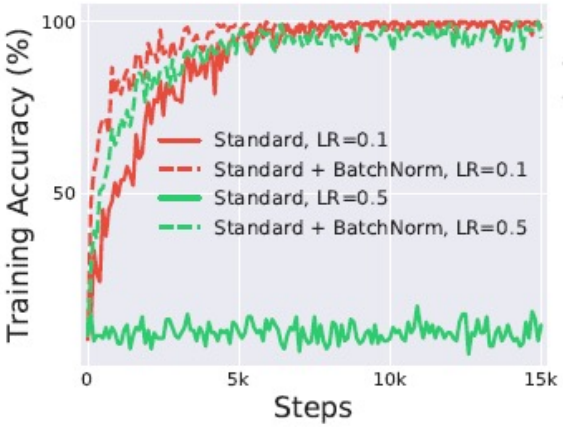
BATCH NORMALIZATION

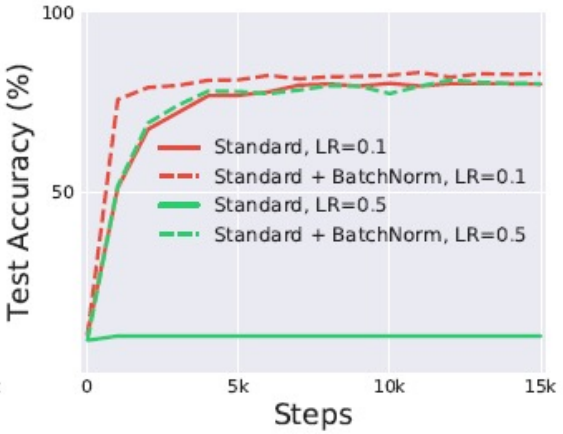
Slides taken from Jude Shavlik: http://pages.cs.wisc.edu/~shavlik/cs638_cs838.html

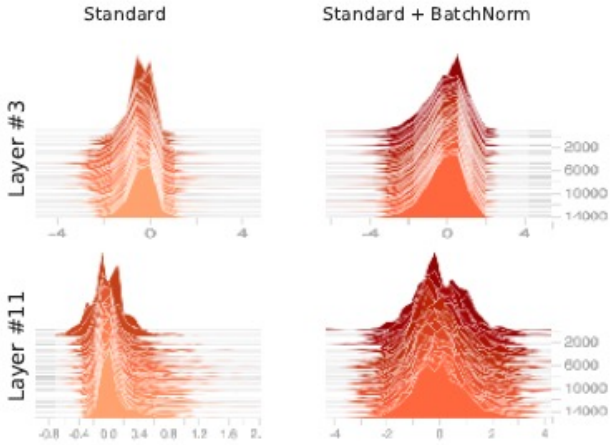
Batch normalization: Other benefits in practice

- BN reduces training times. (Because of less Covariate Shift, less exploding/vanishing gradients.)
- BN reduces demand for regularization, e.g. dropout or L2 norm.
  - Because the means and variances are calculated over batches and therefore every normalized value depends on the current batch. I.e. the network can no longer just memorize values and their correct answers.)
- BN allows higher learning rates. (Because of less danger of exploding/vanishing gradients.)
- BN enables training with saturating nonlinearities in deep networks, e.g. sigmoid. (Because the normalization prevents them from getting stuck in saturating ranges, e.g. very high/low values for sigmoid.)

Internal Covariate Shift







Why the naïve approach Does not work?

- Normalizes layer inputs to zero mean and unit variance. *whitening*.
- Naive method: Train on a batch. Update model parameters. Then normalize. Doesn't work: Leads to exploding biases while distribution parameters (mean, variance) don't change.

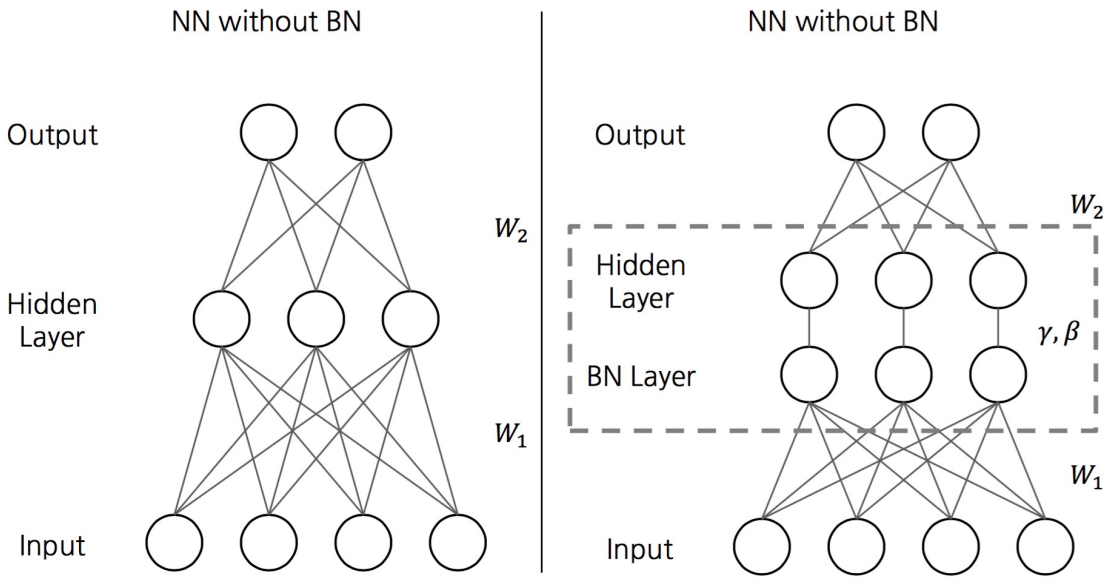
  - If we do it this way gradient always ignores the effect that the normalization for the next batch would have
  - i.e. : "The issue with the above approach is that the gradient descent optimization does not take into account the fact that the normalization takes place"

The proposed solution:

To add an extra regularization layer

we introduce, for each activation $x^{(k)}$, a pair of parameters $\gamma^{(k)}, \beta^{(k)}$, which scale and shift the normalized value:

$$y^{(k)} = \gamma^{(k)} \widehat{x}^{(k)} + \beta^{(k)}.$$



A new layer is added so the gradient can “see” the normalization and make adjustments if needed.

Algorithm Summary: Normalization via Mini-Batch Statistics

- Each feature (component) is normalized individually
- Normalization according to:
  - componentNormalizedValue = (componentOldValue - E[component]) / sqrt(Var(component))
- A new layer is added so the gradient can "see" the normalization and made adjustments if needed.
  - The new layer has the power to learn the identity function to de-normalize the features if necessary!
  - Full formula: newValue = gamma * componentNormalizedValue + beta (gamma and beta learned per component)
- E and Var are estimated for each mini batch.
- BN is fully differentiable.

The Batch Transformation: formally from the paper.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

The full algorithm as proposed in the paper

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$

Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

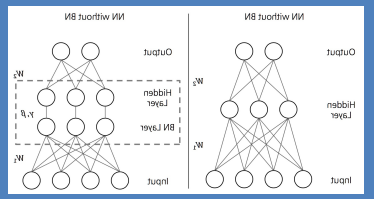
$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

Algorithm 2: Training a Batch-Normalized Network

Alg 1 (previous slide)

Architecture modification



Note that BN(x) is different during test…

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

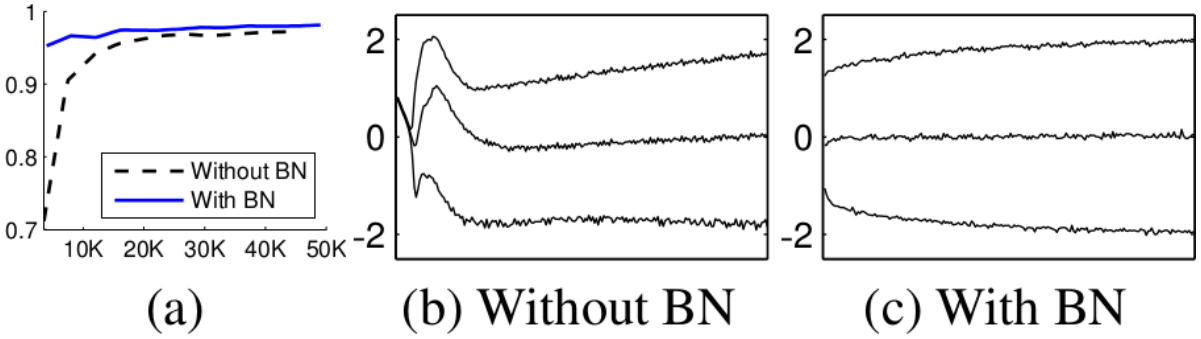
Vs.

$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

Batch normalization: Other benefits in practice

- BN reduces training times. (Because of less Covariate Shift, less exploding/vanishing gradients.)
- BN reduces demand for regularization, e.g. dropout or L2 norm.
  - Because the means and variances are calculated over batches and therefore every normalized value depends on the current batch. I.e. the network can no longer just memorize values and their correct answers.)
- BN allows higher learning rates. (Because of less danger of exploding/vanishing gradients.)
- BN enables training with saturating nonlinearities in deep networks, e.g. sigmoid. (Because the normalization prevents them from getting stuck in saturating ranges, e.g. very high/low values for sigmoid.)

Batch normalization: Better accuracy , faster.



BN applied to MNIST (a), and activations of a randomly selected neuron over time (b, c), where the middle line is the median activation, the top line is the 15th percentile and the bottom line is the 85th percentile.

References:

- **SGD convergence rate**:
  Shalev-Shwartz, S. and Ben-David, S., 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

- **Stochastic Averaged Gradient**:
  https://svan2016.sciencesconf.org/resource/page/id/6.html

- **First SGD in ML paper**:
- Léon Bottou and Olivier Bousquet: **The Tradeoffs of Large Scale Learning**, *Advances in Neural Information Processing Systems*, 20, MIT Press, Cambridge, MA, 2008.