Stochastic Subgradient Method

Lingjie Weng, Yutian Chen

Bren School of Information and Computer Science

UC Irvine

Subgradient

- Recall basic inequality for convex differentiable f :

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

- The gradient $\nabla f(x)$ at x determines a global under-estimator of f.

What if f is not differentiable?

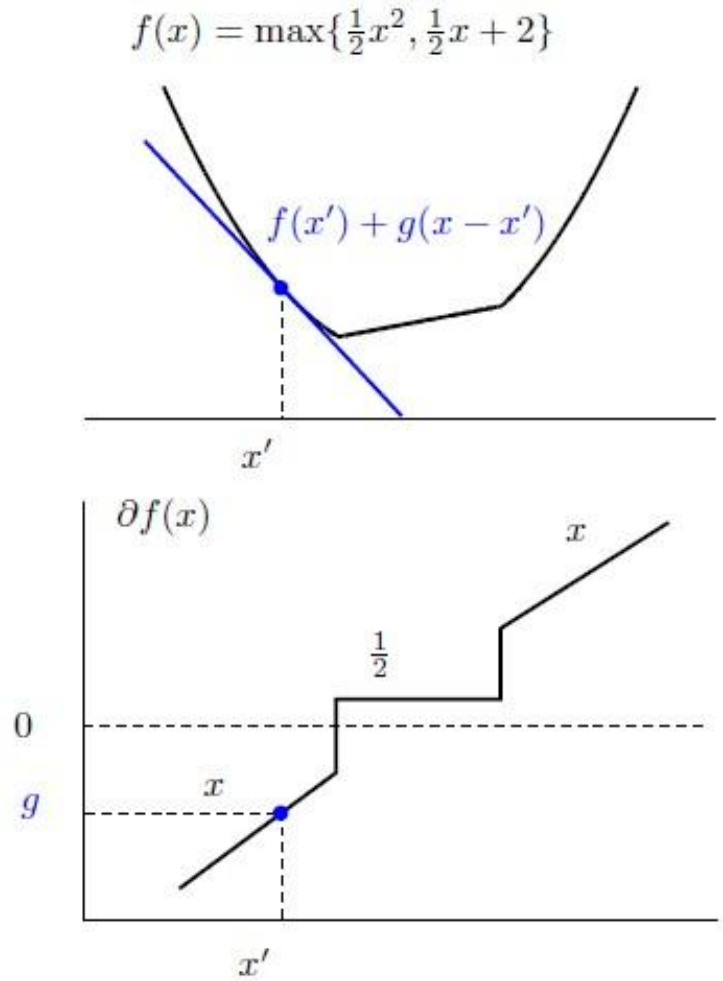
- We can still construct a global under-estimator : the vector g is a subgradient of f at x if

$$f(y) \geq f(x) + g^T (y - x) \quad \text{for all } y$$

- There can be more than one subgradient at given point: the collection of subgradients of f at x is the subdifferential $\partial f(x)$.

An example: non-differentiable function and its subgradients

$f = \max\{f_1, f_2\}$, with f_1, f_2 convex and differentiable



- $f_1(x') > f_2(x')$: unique subgradient $g = \nabla f_1(x')$
- $f_1(x') < f_2(x')$: unique subgradient $g = \nabla f_2(x')$
- $f_1(x') > f_2(x')$: subgradient form a line segment $[\nabla f_1(x'), \nabla f_2(x')]$

Subgradient methods

- Subgradient method is simple algorithm to minimize nondifferentiable convex function f

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$$

$x^{(k)}$ is the kth iterate

$g^{(k)}$ is any subgradient of f at $x^{(k)}$

$\alpha_k > 0$ is the kth step size

- Not a descent method, so we keep track of best point so far

$$f_{best}^{(k)} = \min\{f(x^{(1)}), \cdots, f(x^{(k)})\}$$

Why does it work ?--convergence

Assumptions:

- $f^* = \inf_x f(x) > -\infty$, with $f(x^*) = f^*$
- $\| g^{(k)} \|_2 \leq G$ for all k (if f satisfies Lipschitz condition $|f(u) - f(v)| \leq G \| u - v \|_2$)
- $\| x^{(1)} - x^* \|_2 \leq R$

Convergence results for different step size

- Constant step size: $\alpha_k = \alpha$(positive constant).
  - Converges to a suboptimal solution: $f_{best}^{(k)} - f^* \leq \frac{R^2 + G^2 k\alpha^2}{2k\alpha} \xrightarrow{k\to\infty} \frac{G^2\alpha}{2}$
- Constant step length: $\alpha_k = \frac{\gamma}{\|g^{(k)}\|_2}$ (i.e. $\left\|x^{(k+1)} - x^{(k)}\right\| = \gamma$).
  - Converges to a suboptimal solution: $f_{best}^{(k)} - f^* \leq \frac{R^2 + \gamma^2 k}{2\gamma k/G} \xrightarrow{k\to\infty} \frac{G\gamma}{2}$

- Square summable but not summable $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$, $\sum_{k=1}^{\infty} \alpha_k = \infty$ (i.e. $\alpha_k = \frac{a}{b+k}$)
- Nonsummable diminishing step size $\lim_{k\to\infty} \alpha_k = 0$, and $\sum_{k=1}^{\infty} \alpha_k = \infty$ (i.e. $\alpha_k = \frac{a}{\sqrt{k}}$)
- Nonsummable diminishing step length $\alpha_k = \frac{\gamma_k}{\|g^{(k)}\|_2}$
  - Converges to the optimal solution: $\lim_{k\to\infty} f_{best}^{(k)} - f^* = 0$

Stopping criterion

- Terminating when $\frac{R^2+G^2\sum_{i=1}^{k}\alpha_i^2}{2\sum_{i=1}^{k}\alpha_i} \leq \varepsilon$ is really, really slow.
- $\alpha_i = (R/G)/\sqrt{k}, i = 1, \ldots, k$ minimizes the suboptimality bound.
- The number of steps to achieve a guaranteed accuracy of ε will be at least $(RG/\varepsilon)^2$, no matter what step sizes we use.

- There really isn't a good stopping criterion for the subgradient method
  - are used only for problems in which very high **accuracy** is **not** required, typically around 10%

Subgradient method for unconstrained problem

1. Set $k := 1$ and. Choose an infinite sequence of positive step size value $\{\alpha^{(k)}\}_{k=1}^{\infty}$.
2. Compute a subgradient $g^{(k)} \in \partial f(x^{(k)})$.
3. Update $x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$
4. If algorithm has not converged, then set $k := k + 1$ and go to step 2.

Properties:

- Simple and general. Any subgradient from $\partial f(x^{(k)})$ is sufficient.
- Convergence theorems exist for properly selected step size
- Slow convergence. It usually requires a lot of iterations
- No monotonic convergence. $-g^{(k)}$ need not be decreasing direction → line search cannot be used to find optimal α_k.
- No good stopping condition.

Example: L_2-norm Support Vector Machines

- $(\boldsymbol{w}^*, b^*, \xi^*) = \underset{\boldsymbol{w},b,\xi}{\operatorname{argmin}} \left[\frac{1}{2} \| \boldsymbol{w} \|^2 + C \sum_{i=1}^{m} \xi_i\right],$

  s.t. $\quad y_i(\boldsymbol{w}^T x_i + b) \geq 1 - \xi_i, \; i = 1, \dots, m$

  $\xi_i \geq 0, \quad i = 1, \dots, m$

- Note that $\xi_i \geq \max\{0, 1 - y_i(\boldsymbol{w}^T x_i + b)\}$ and the equality is attained in the optimum, i.e., we can optimize

  s.t. $\xi_i = \max\{0, 1 - y_i(\boldsymbol{w}^T x_i + b)\} = \ell(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle y_i)$ (hinge loss)

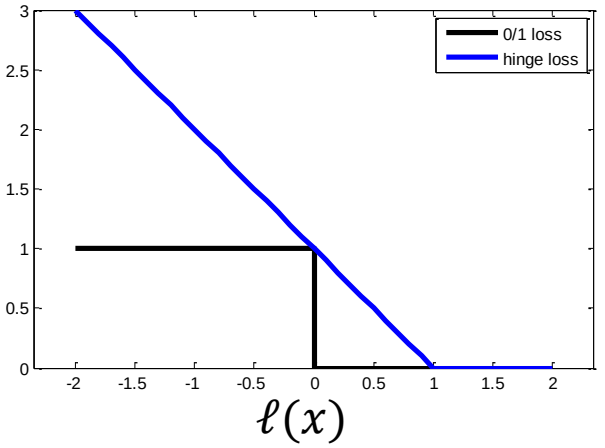
- We can reformulate the problem as an unconstrained convex problem

$$(\boldsymbol{w}^*, b^*) = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \left[\frac{1}{2} \| \boldsymbol{w} \|^2 + C\ell(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle y_i)\right]$$

Example L_2-norm Support Vector Machines(cont')

- Function $f_0(\boldsymbol{w}, b) = \frac{1}{2} \| \boldsymbol{w} \|^2$ is differentiable and thus $\partial_w f_0(\boldsymbol{w}, b) = \mathrm{w}$, $\partial_b f_0(\boldsymbol{w}, b) = 0$.
- Function $f_i(\boldsymbol{w}, b) = \max\{0, 1 - y_i(\boldsymbol{w}^T x_i + b)\} = \ell(y_i(\boldsymbol{w}^T x_i + b))$is a point-wise maximum thus subdifferential is convex combination of gradients of active functions

i-th function	$\partial_{\boldsymbol{w}} f_i(\boldsymbol{w}, b)$	$\partial_b f_i(\boldsymbol{w}, b)$
$I_+ = \{i \mid y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) > 1\}$	$\mathbf{0}$	0
$I_0 = \{i \mid y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) = 1\}$	$\mathbf{Co}\{\mathbf{0}, -y_i \boldsymbol{x}_i\}$	$\mathbf{Co}\{0, -y_i\}$
$I_- = \{i \mid y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) < 1\}$	$-y_i \boldsymbol{x}_i$	$-y_i$

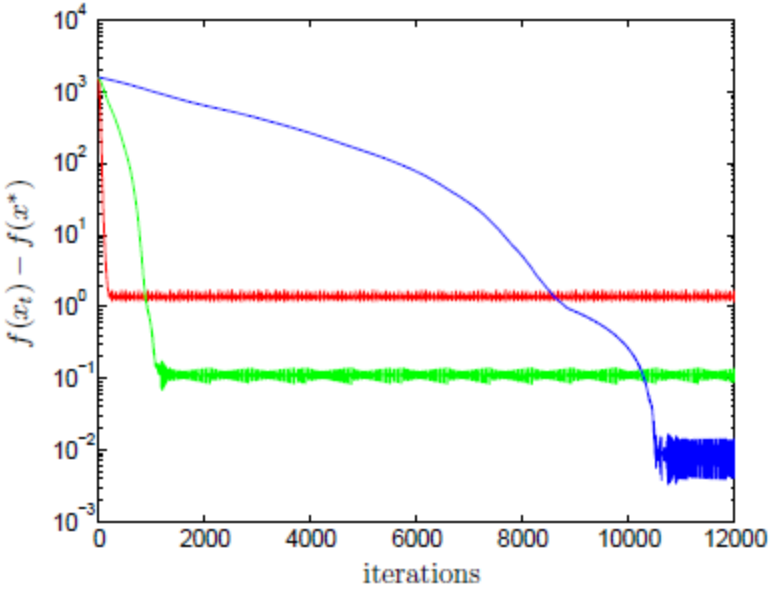


Constant step length: $\alpha_t = \frac{\gamma}{\|g^t\|}$

Converges to a suboptimal solution:

$$\lim_{t\to\infty} f_{best}^{(t)} - f^* \leq \frac{G^2\alpha}{2}$$

$\gamma = 0.1$, $\gamma = 0.01$, $\gamma = 0.001$



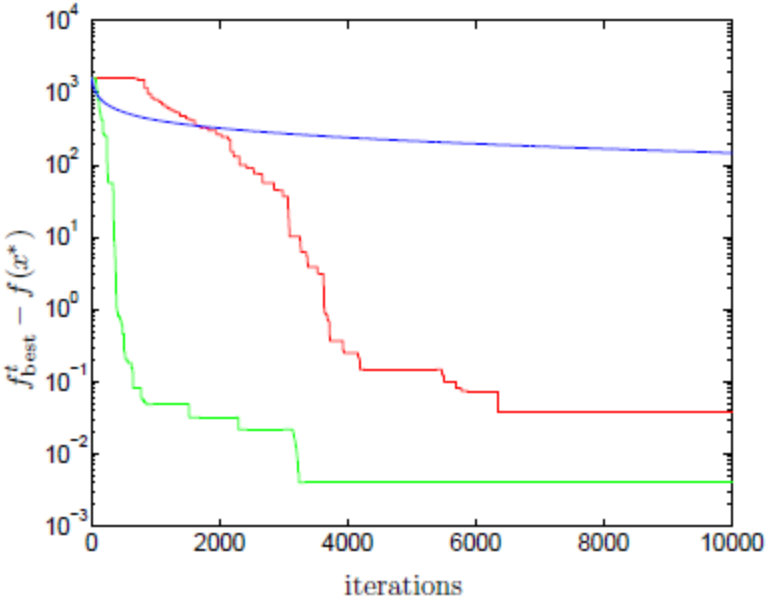
The value of : $f_{best}^{(k)} - f^*$ versus iteration number t

Diminishing step size: $\alpha_t = \frac{\gamma}{t}$

Converges to a optimal solution:

$$\lim_{t\to\infty} f_{best}^{(t)} - f^* = 0$$

$\gamma = 10$, $\gamma = 1$, $\gamma = 0.1$



The value of : $f_{best}^{(k)} - f^*$ versus iteration number t

Trade-off: larger γ gives faster convergence, but larger final suboptimality

Subgradient method for constrained problem

- Let us consider the constraint optimization problem

  minimize $f(x)$

  s.t. $x \in S$ (S is a convex set)

- Projected subgradient method is given by $x^{(k+1)} = P_S(x^{(k)} - \alpha_k g^{(k)})$

  P_S is (Euclidean) projection on S,

1. Set $k := 1$ and. Choose an infinite sequence of positive step size value $\{\alpha^{(k)}\}_{k=1}^{\infty}$.
2. Compute a subgradient $g^{(k)} \in \partial f(x^{(k)})$.
3. Update$x^{(k+1)} = P_S(x^{(k)} - \alpha_k g^{(k)})$
4. If algorithm has not converged, then set $k := k + 1$ and go to step 2.

Example L_2-norm Support Vector Machines

- $(\boldsymbol{w}^*, b^*) = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \left[\frac{1}{2} \| \boldsymbol{w} \|^2 + C \sum_{i=1}^{m} \max\{0, 1 - y_i(\boldsymbol{w}^T x_i + b)\}\right]$
- An alternative formulation of the same problem is

$$(\boldsymbol{w}^*, b^*) = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \max\{0, 1 - y_i(\boldsymbol{w}^T x_i + b)\}$$

$$\text{S.t. } \|\boldsymbol{w}\| \leq W$$

- Note that the regularization constant C > 0 is replaced here by a less abstract constant $W > 0$.

- To implement subgradient method we need only a projection operator

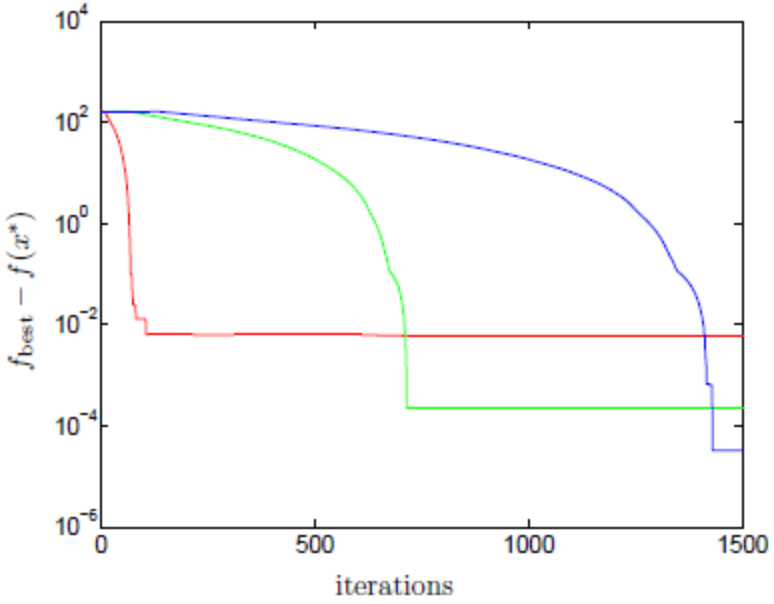
$$P_S(\boldsymbol{w}, b) = \{\mathrm{w} \rightarrow \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} W, \mathrm{b} \rightarrow \mathrm{b}\}$$

- So after update $\boldsymbol{w} := \boldsymbol{w} - \alpha_k \boldsymbol{w}'$ and $b := b - \alpha_k b'$,

  we project w:= $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} W$

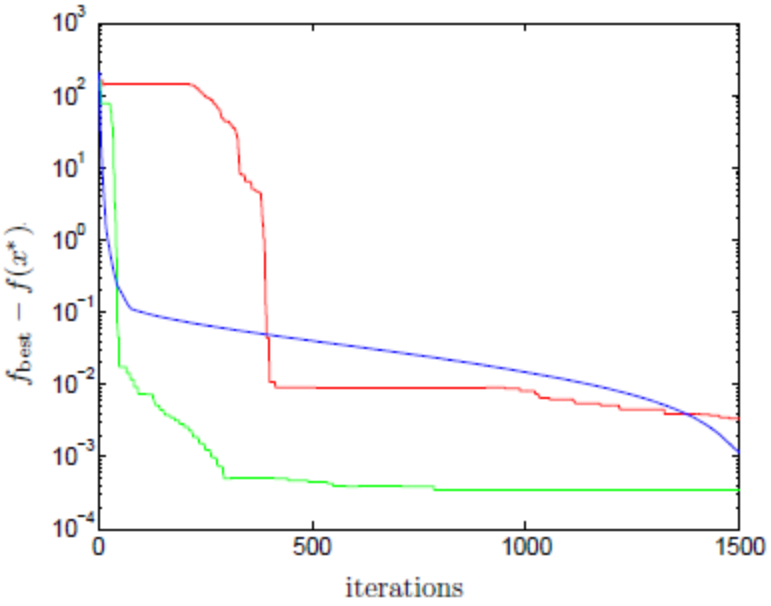
Constant step length: $\alpha_t = \frac{\gamma}{\|g^t\|}$

$\gamma = 0.1$, $\gamma = 0.01$, $\gamma = 0.005$



Diminishing step size: $\alpha_t = \frac{\gamma}{t}$

$\gamma = 10$, $\gamma = 1$, $\gamma = 0.1$



Noisy unbiased subgradient

- Random vector $\tilde{g} \in \boldsymbol{R}^n$ is a **noisy unbiased subgradient** for $f: \boldsymbol{R}^n \to \boldsymbol{R}$ at x if for all z

$$f(z) \geq f(x) + (\boldsymbol{E}\tilde{g})^T(z - x)$$

$i.e., g = \boldsymbol{E}\tilde{g} \in \partial f(x)$

- The noise can represent error in computing g, measurement noise, Monte Carlo sampling error, etc
- If x is also random, $\tilde{g}$ is a noisy unbiased subgradient of f at x if

$$\forall z \;\; f(z) \geq f(x) + \boldsymbol{E}(\tilde{g}|x)^T(z - x)$$

holds almost surely

- Same as $\boldsymbol{E}(\tilde{g}|x) \in \partial f(x)$ (a.s.)

Stochastic subgradient method

- **Stochastic subgradient method** is the subgradient method, using noisy unbiased subgradients

$$x^{(k+1)} = x^{(k)} - \alpha_k \tilde{g}^{(k)}$$

- $x^{(k)}$ is kth iterate
- $\tilde{g}^{(k)}$ is any noisy unbiased subgradient of (convex) f at $x^{(k)}$
- $\alpha_k > 0$ is the kth step size
- Define $f_{best}^{(k)} = \min\{f(x^{(1)}), \cdots, f(x^{(k)})\}$

Assumptions

- $f^* = \inf_x f(x) > -\infty$, with $f(x^*) = f^*$
- $\boldsymbol{E} \parallel g^{(k)} \parallel_2^2 \leq G^2$ for all k
- $\boldsymbol{E} \parallel x^{(1)} - x^* \parallel_2^2 \leq R^2$
- Step sizes are square-summable but not summable

$$\alpha_k \geq 0, \sum_{k=1}^{\infty} \alpha_k^2 = \parallel \alpha \parallel_2^2 < \infty, \sum_{k=1}^{\infty} \alpha_k = \infty$$

These assumptions are stronger than needed, just to simplify proofs

Convergence results

- Convergence in expectation:

$$\boldsymbol{E} f_{best}^{(k)} - f^* \leq \frac{R^2 + G^2 \|\alpha\|_2^2}{2 \sum_{i=1}^{k} \alpha_i}$$

$$\lim_{k \to \infty} \boldsymbol{E} f_{best}^{(k)} = f^*$$

- Convergence in probability (Markov's inequality): for any $\epsilon > 0$,

$$\lim_{k \to \infty} \mathbf{Prob}\left(f_{best}^{(k)} \geq f^* + \epsilon\right) = 0$$

- Almost sure convergence:

$$\lim_{k \to \infty} f_{best}^{(k)} = f^* \text{ (a.s.)}$$

Example: Machine learning problem

- Minimizing loss function

$$\min_{\boldsymbol{w}} \hat{L}(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} loss(w, (x_i, y_i))$$

Subgradient estimate: $\tilde{g}^{(k)} = \nabla_{\boldsymbol{w}} loss(\boldsymbol{w}^{(k)}, (x_i, y_i))$

- Examples: L_2 regularized linear classification with hinge loss, aka. SVM

$$\min_{\|\boldsymbol{w}\|_2 \le B} \frac{1}{m}\sum_{i=1}^{m} \ell(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle y_i) \quad \equiv \quad \min \frac{1}{m}\sum_{i=1}^{m} \ell(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle y_i) + \frac{\lambda}{2} \| \boldsymbol{w} \|^2$$

Initialize at $\boldsymbol{w}^{(\mathbf{0})}$

Iteration k:

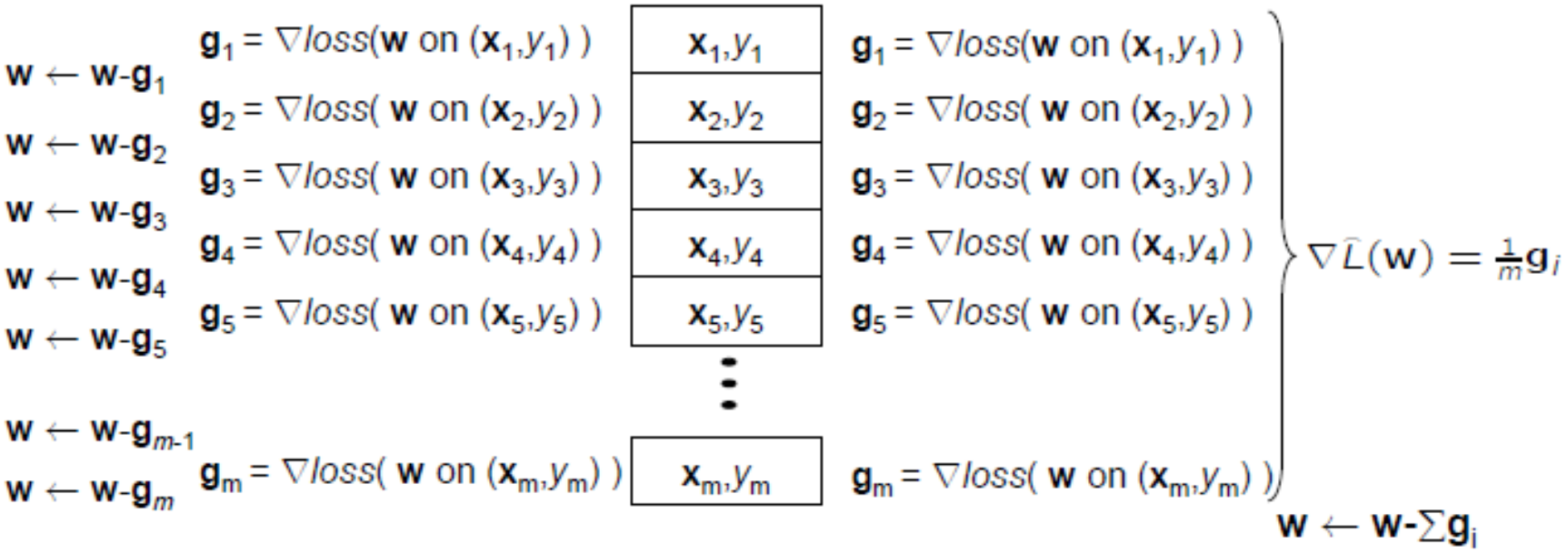
$\boldsymbol{w}^{(k+1)} \leftarrow \boldsymbol{w}^{(k)} - \alpha^{(k)} \tilde{g}^{(k)}$

If $\| \boldsymbol{w}^{(k+1)} \| > B$

$$\boldsymbol{w}^{(k+1)} \leftarrow B \frac{\boldsymbol{w}^{(k+1)}}{\|\boldsymbol{w}^{(k+1)}\|}$$

Stochastic vs Batch Gradient Descent

$$\min_{\mathbf{w}} \bar{L}(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} loss(\mathbf{w} \text{ on } (\mathbf{x}_i, y_i))$$



$$\bar{\boldsymbol{w}} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{w}_i$$

Stochastic vs Batch Gradient Descent

- Runtime to guarantee an error ϵ ($\alpha^{(k)} = \frac{B/G}{\sqrt{k}}$)
  - Stochastic Gradient Descent: $\mathcal{O}(\frac{G^2B^2}{\epsilon^2}d)$
  - Batch Gradient Descent: $\mathcal{O}\left(\frac{G^2B^2}{\epsilon^2}md\right)$

- Optimal runtime if only using gradients and only assuming Lipchitz condition.
- Slower than second order method

Stochastic programming

minimize $\boldsymbol{E} f_0(x, \omega)$
subject to $\boldsymbol{E} f_i(x, \omega) \leq 0, i = 1, \cdots, m$

If $f_i(x, \omega)$ is convex in x for each ω, problem is convex

- "cerntainty-equivalent" problem

  minimize $f_0(x, \boldsymbol{E}\omega)$
  subject to $f_i(x, \boldsymbol{E}\omega) \leq 0, i = 1, \cdots, m$

(if $f_i(x, \omega)$ is convex in ω, gives a lower bound on optimal value of stochastic problem)

Machine learning as stochastic programming

Minimizing generalization error:

$$\min_{\boldsymbol{w}\in\mathcal{W}} \boldsymbol{E}[f(\boldsymbol{w}, z)]$$

- Two approaches:
  - Sample Average Approximation (SAA):
    - Collect sample $z_1, \cdots, z_m$
    - Minimize $\hat{F}(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} f(\boldsymbol{w}, z_i)$, $\hat{\boldsymbol{w}} = \min_{\boldsymbol{w}} \hat{F}(\boldsymbol{w})$
    - Minimizing empirical risk
    - Stochastic/batch gradient descent, Newton method, …
  - Sample Approximation (SA):
    - Update $\boldsymbol{w}^{(k)}$ based on weak estimator to $\nabla F(\boldsymbol{w}^{(k)})$

      $$\tilde{g}^{(k)} = \nabla f(\boldsymbol{w}^{(k)}, z_k)$$

    - Stochastic gradient descent , $\bar{\boldsymbol{w}}^{(m)} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{w}_i$

Machine learning as stochastic programming

Minimizing generalization error:

$$\min_{\boldsymbol{w}\in\mathcal{W}} \boldsymbol{E}[f(\boldsymbol{w}, z)]$$

$$\min_{w} \hat{L}(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} loss(w, (x_i, y_i))$$

- Two approaches:
  - Sample Average Approximation (SAA):
    - Collect sample $z_1, \cdots, z_m$
    - Minimize $\hat{F}(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} f(\boldsymbol{w}, z_i)$, $\hat{\boldsymbol{w}} = \min_{\boldsymbol{w}} \hat{F}(\boldsymbol{w})$
    - Minimizing empirical risk
    - Stochastic/batch gradient descent, Newton method, …
  - Sample Approximation (SA):
    - Update $\boldsymbol{w}^{(k)}$ based on weak estimator to $\nabla F(\boldsymbol{w}^{(k)})$

      $$\tilde{g}^{(k)} = \nabla f(\boldsymbol{w}^{(k)}, z_k)$$

    - Stochastic gradient descent , $\bar{\boldsymbol{w}}^{(m)} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{w}_i$

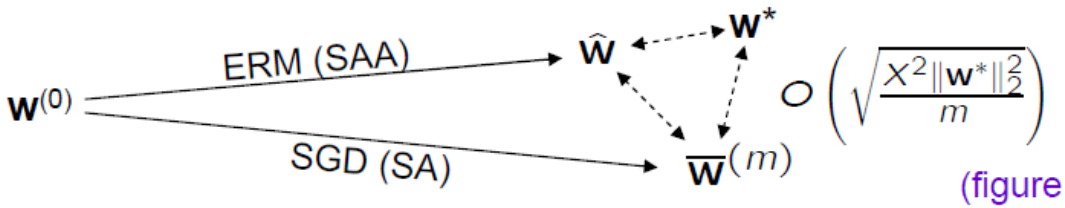
SAA vs SA

- Runtime to guarantee an error ϵ ($L(\boldsymbol{w}) \leq L(\boldsymbol{w}^*) + \epsilon$)
  - SAA (Empirical Risk Minimization)
    - Sample size to guarantee $L(\widehat{\boldsymbol{w}}) \leq L(\boldsymbol{w}^*) + \epsilon$: $m = \Omega(\frac{G^2B^2}{\epsilon^2})$
    - Using any method (stochastic/batch subgradient descent, Newton method, …), runtime:

$$\Omega(\frac{G^2B^2}{\epsilon^2}d)$$

  - SA (Single-Pass Stochastic Gradient Descent)
    - After k iteration, $L(\bar{\boldsymbol{w}}^{(k)}) \leq L(\boldsymbol{w}^*) + \mathcal{O}(\sqrt{\frac{G^2B^2}{k}})$
    - Runtime:

$$\mathcal{O}(\frac{G^2B^2}{\epsilon^2}d)$$



(figure adapted from Leon Bottou)

Reference

- S. Boyd, et. al. Subgradient Methods, Stochastic Subgradient Methods. Lecture slides and notes for EE364b, Stanford University, Spring Quarter 2007–08
- Professor Vojtech Franc's talks. A Short Tutorial on Subgradient Methods. IDA retreat, May 16, 2007
- Nathan Srebro and Ambuj Tewari, Stochastic Optimization: ICML 2010 Tutorial, July, 2010