MATH466/MATH766
Math of machine learning
02/17-02/19 Lecture 11-12 Optimization Algorithm (cont.)

References:

- Convex Optimization by Stephen Boyd and Lieven Vandenberghe
- Lecture notes by Prof. Ryan Tibshirani
  https://www.stat.cmu.edu/~ryantibs/convexopt-F15/
- L. Bottou and O. Bousquet, The tradeoffs of large scale learning, 2007, Neurips
- L. Bottou, Stochastic gradient descent tricks, 2012
- Ch 8 of Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville.

- SAG: Schmidt et.al. 2017, minimizing finite sums with the stochastic average gradient
- SAGA: Defazio, Bach and Lacoste-Julien, 2014, SAGA: a fast incremental gradient method with support for non-strongly convex composite objects
- SVRG: Johnson and Zhang, 2013, Accelerating stochastic gradient descent using predictive variance reduction
- SARAH: Nguyen et.al. 2017, SARAH: a novel method for machine learning problems using stochastic recursive gradient

- Y. Nesterov. Introductory lecture on convex programming
- AdaGrad: Duchi, Hazan and Singer, 2011, Adaptive subgradient methods for online learning and stochastic optimization
- AdaDelta: Zeiler, 2012. Adadelta: an adaptive learning rate method.
- RMSProp: Tieleman and Hinton 2012, Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report.
- ADAM: Kingma and Ba, 2014, Adam: A method for stochastic optimization

Todays contents:

- proximal operator and proximal gradient method
- stochastic gradient method
- variants of SGD
- optimization for training deep models

Important concepts:

- proximal operator
- stochastic gradient

Recommend reading:

- Ch 8 of Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville.stochas
- L. Bottou, Stochastic gradient descent tricks, 2012

Warm up: $f(x) = \frac{1}{2} x^T A x$

A is symmetric, w/. eigen-decomposition $U \Lambda U^T$

$\nabla f(x) = \frac{1}{2}(A + A^T)x = Ax$, $\nabla^2 f(x) = A$

$f(x)$ is cvx $\Leftrightarrow$ _____

$f(x)$ is μ-cvx $\Leftrightarrow$ _____

$f(x)$ is L-smooth $\Rightarrow$ _____

0. Review

- convergence rate

$x_k = \frac{1}{\sqrt{k}}$, $k=1,2,\cdots$; $x_k = \frac{1}{k^2}$, $k=1,2,\cdots$; $x_k = \frac{1}{2^k}$, $k=1,2,\cdots$; $x_k = e^{-2^k}$, $k=1,2,\cdots$

- unconstrained opt $x^{k+1} = x^k + t^k p^k$

  (t^k: stepsize; p^k: search direction)

  $\min_{x \in \mathbb{R}^n} f(x)$ (backtracking line search) (descent direction: e.g. negative grad)

  ($\frac{1}{L}$ for L-smooth f) (subgradient)

- SubGrad Thm. Assume that $f(x) \geq f(x^*)$ for any x

  and f is G-Lipschitz continuous i.e. $|f(x) - f(y)| \leq G\|x-y\|_2$

  Then $f(x_{best}^{(k)}) - f(x^*) \leq \frac{R^2 + G^2 \sum_{k=1}^{k} t_k^2}{2 \sum_{k=1}^{k} t_k}$, $(R = \|x^{(1)} - x^*\|)$

If we pick a fixed step-size t, then $\lim_{k \to +\infty} f(x_{best}^{(k)}) \leq f(x^*) + \frac{G^2 t}{2}$

The optimal value may NOT be achieved in the limit.

Smaller step-size may reduce the gap but requires more iterations.

Solution: diminishing step-size e.g. $t_k = \frac{1}{k}$

Convergence rate: To have $\frac{R^2 + G^2 \sum_{k=1}^{k} t_k^2}{2 \sum_{k=1}^{k} t_k} \leq \varepsilon$, need at least $\left(\frac{RG}{\varepsilon}\right)^2$ iterations

$$\frac{R^2 + G^2 \sum_{k=1}^{k} t_k^2}{2 \sum_{k=1}^{k} t_k} \geq \frac{RG}{\sqrt{K}}, \quad "=" \text{ when } t_k = \frac{R/G}{\sqrt{k}}$$

	L-smooth (or G-cts)	L-smooth (or G-cts) & μ-strongly cvx
GD	$O(\frac{1}{\varepsilon})$	$O(\log \frac{1}{\varepsilon})$
subGrad	$O(\frac{1}{\varepsilon^2})$	$O(\frac{1}{\varepsilon})$

lose efficiency or linear cvg due to non-smoothness

Q: Is it possible to obtain the same convergence rate as GD for nonsmooth obj ?

1. Proximal Method and Proximal Gradient

1.1. Proximal operator and proximal method

Def. (Proximal Operator) Let r be a closed convex function.

i.e. $\forall \alpha \in \mathbb{R}$, $\{x \in \mathrm{dom}(r) \mid r(x) \le \alpha\}$ is a closed convex set

The proximal mapping of r at x is defined as

$$\mathrm{prox}_r(x) = \arg\min_y \; r(y) + \frac{1}{2}\|y - x\|_2^2$$

Proximal method: If f is convex,

consider $x^{(k+1)} = \mathrm{prox}_{tf}(x^{(k)}) = \arg\min_x \; f(x) + \frac{1}{2t}\|x - x^{(k)}\|^2$

why this works:

① (intuition) from definition

② if f is differentiable, the update is equivalent to

$$x^{(k+1)} = x^{(k)} - t\nabla f(x^{(k+1)})$$

(Recall GD: $x^{(k+1)} = x^{(k)} - t\nabla f(x^{(k)})$)

Both simulate $dx(t) = -\nabla f(x(t))\,dt$ (gradient flow)

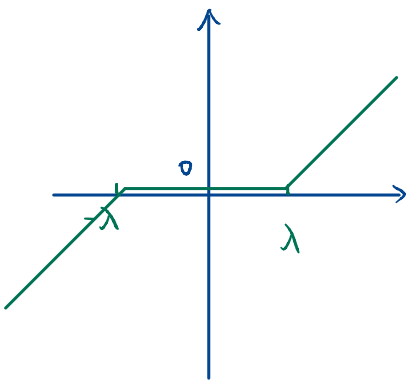
GD $\Leftrightarrow$ forward Euler PD $\Leftrightarrow$ backward Euler

Prop. If r is a closed convex function, then for any $x \in \mathrm{dom}(r)$, $\mathrm{prox}_r(x)$ exist and is unique.

Remark. For nonconvex function, we can also define proximal operator, but $\mathrm{prox}_r(x)$ may not exist or may not be unique.

e.g. For $\lambda > 0$, $r(x) = \lambda |\cdot|_1$.

$\text{prox}_{\lambda|\cdot|_1}(x) = \text{sgn}(x)\max\{0, |x| - \lambda\}$ (soft-threshold)



Pf. $\text{prox}_{\lambda|\cdot|_1}(x) = \underset{y}{\text{argmin}}\ \lambda|y| + \frac{1}{2}(y-x)^2 =: f(y)$

$$f(y) = \begin{cases} \frac{1}{2}[y-(x-\lambda)]^2 + \frac{1}{2}(x^2-(x-\lambda)^2) & y > 0 \\ \frac{1}{2}[y-(x+\lambda)]^2 + \frac{1}{2}(x^2-(x+\lambda)^2) & y \le 0 \end{cases}$$

① if $x \ge \lambda$, when $y > 0$, $f(y) \ge f(x-\lambda) = \frac{1}{2}\lambda(2x-\lambda)$ $\Rightarrow y^* = x-\lambda$

when $y \le 0$, $f(y) \ge f(0) = \frac{1}{2}x^2 \ge \frac{1}{2}\lambda x \ge \frac{1}{2}\lambda(2x-\lambda)$

② if $x \le -\lambda$, when $y > 0$, $f(y) > f(0) = \frac{1}{2}x^2 \ge -\frac{1}{2}\lambda x \ge -\frac{1}{2}\lambda(2x+\lambda)$ $\Rightarrow y^* = x+\lambda$

when $y \le 0$, $f(y) \ge f(x+\lambda) = -\frac{1}{2}\lambda(2x+\lambda)$

③ if $-\lambda < x < \lambda$ when $y > 0$, $f(y) > f(0)$. $\Rightarrow y^* = 0$

when $y \le 0$, $f(y) \ge f(0)$

1.2. Proximal Gradient Method

A common setup $\min_{x \in \mathbb{R}^n} f(x) = h(x) + r(x)$

h is convex, differentiable, $\text{dom}(g) = \mathbb{R}^n$

r is convex, not necessarily differentiable

e.g. LASSO $\min_w \frac{1}{2}\|Xw - y\|_2^2 + \lambda\|w\|_1$.

Alg. ProxGD

Initialization $x^{(0)}$

While (???)

| $x^{(k+1)} = \text{prox}_{tr}(x^{(k)} - t\nabla h(x^{(k)}))$

Thm. If h is convex, differentiable and L-smooth,

r is convex and $\text{prox}_{tr}(x)$ can be evaluated

then proximal gradient descent with fixed step size $t \le \frac{1}{L}$

satisfies $$f(x^{(k)}) - f^* \le \frac{\|x^{(0)} - x^*\|_2^2}{2tk}$$

Can be $O(c^k)$ for strongly convex case.

e.g. LASSO

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda \|w\|_1 \qquad (\lambda > 0)$$

h ↑ , r ↑ cvx

$$\nabla h = X^T(Xw - y), \quad \nabla^2 h = X^T X \text{ p.s.d. } \|X^T X\|_2\text{-smooth}$$

$\Rightarrow h$ cvx

$$w^{k+1} = \text{prox}_{\tau r}(w^k - \tau \nabla h(w^k))$$

$$= \text{prox}_{\tau\lambda\|\cdot\|_1}(w^k - \tau X^T(Xw^k - y)) \qquad \tau = \frac{1}{\|X^T X\|_2}$$

Comparison of GD, subGrad, ProxGD

	L-smooth	L-smooth & μ-strongly cvx
GD	$O(\frac{1}{\varepsilon})$	$O(\log \frac{1}{\varepsilon})$
subGrad	$O(\frac{1}{\varepsilon^2})$	$O(\frac{1}{\varepsilon})$
ProxGD	$O(\frac{1}{\varepsilon})$	$O(\log \frac{1}{\varepsilon})$

ProxGD gain efficiency

What is sacrificed?

2. Stochastic Gradient Descent (SGD)

Motivation. Recall $\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(w; x_i, y_i)$

where $\ell(w; x_i, y_i) = \frac{1}{2}(w^T x_i - y_i)^2$ linear regression

$\ell(w; x_i, y_i) = \frac{1}{2}(w^T x_i - y_i)^2 + \frac{\lambda}{2}\|w\|_2^2$ ridge regression

$\ell(w; x_i, y_i) = \max\{0, 1 - y_i(\langle w, x_i\rangle + b)\} + \frac{\lambda}{2}\|w\|^2$ SVM

cost of each iteration $O(nd)$ ↑ can be very large

Problem Formulation $\min_w f(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w)$

Stochastic Gradient Update:

randomly pick a batch of samples $\{i_1, i_2, \cdots, i_b\}$

$$w^{(k+1)} = w^{(k)} - t^{(k)} \sum_{j=1}^{b} \nabla f_{i_j}(w^{(k)})$$

- Similar to SubGrad, SGD is not a descent method
- Convergence: If f is L-smooth, $f(w) \geq f^*$ for any w

batch size $b=1$ $\mathbb{E}_i[\|\nabla f_i(w)\|^2] \leq G^2$.

Then $\min\left\{\mathbb{E}\left[\|\nabla f(w^k)\|^2\right]\right\}_{k=0}^{k-1} \leq \dfrac{f(w^0) - f^* + \frac{G^2 L}{2}\sum_{k=0}^{k-1} t_k^2}{\sum_{k=0}^{k-1} t_k}$

Alg SGD

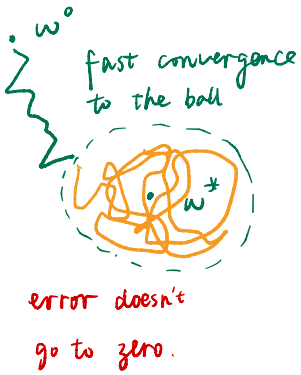
For $\ell = 1, 2, 3, \cdots$ (epoch)

  For $k = 1, 2, 3, \cdots n/b$

    randomly pick a batch of samples

    do stochastic gradient update

(each epoch touches n samples)



If $t_k = t$ is a constant

RHS may not $\to 0$ when $k \to +\infty$

Even when f is strongly convex

cst step-size cannot guarantee $\|\nabla f(w^k)\| \to 0$

it only give a faster cvg rate to the noisy ball.

To have SGD converging to an optimizer / stationary point.

need $\sum_{k=1}^{\infty} t_k = \infty$ and $\sum_{k=1}^{\infty} t_k^2 < \infty$, typical choice $t_k = \frac{1}{1+\alpha k}$

- Convergence rate comparison

$f(x^k) - f^*$

obj	step-size	GD	SGD
cvx	diminish	$O(\frac{1}{\sqrt{k}})$	$O(\frac{1}{\sqrt{k}})$
cvx, L-smooth	cst	$O(\frac{1}{k})$	$O(\frac{1}{\sqrt{k}})$
μ-cvx, L-smooth	cst	$O(e^{-k})$	$O(\frac{1}{k})$

Remark: We can also apply "stochastic" strategy to subgradient method and proximal gradient method, and other algorithms.

Pro: Simple and efficient

Con: Hard to achieve high accuracy

3. Optimization for Training Deep Models

Stochastic type algorithms are the most common algorithms in ML especially deep learning. Step-size are often chosen as constant.

Learning ≠ Pure Optimization

3.1. Error Decomposition in Machine Learning [Bottou & Bousquet 2017]

$$E(f) = \int \ell(f(x), y)\, dP(x, y) = \mathbb{E}[\ell(f(x), y)]$$

$$f^*(x) = \operatorname*{argmin}_{\hat{y}} \mathbb{E}[\ell(\hat{y}, y) \mid x]$$

① Select $\mathcal{F}$, $f^*_{\mathcal{F}} = \operatorname*{argmin}_{f \in \mathcal{F}} E(f)$ — approximation error (VC dim)

② $E_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$

$f_n = \operatorname*{argmin}_{f \in \mathcal{F}} E_n(f)$ — estimation error (concentration ineqn)

③ optimization algorithm output $\tilde{f}_n$

$E_n(\tilde{f}_n) < E_n(f_n) + \rho$ — optimization error (optimization)

$$\mathcal{E} := \mathbb{E}[E(\tilde{f}_n) - E(f^*)]$$

$$= \underbrace{\mathbb{E}[E(\tilde{f}_n) - E(f_n)]}_{\text{opt err}} + \underbrace{\mathbb{E}[E(f_n) - E(f^*_{\mathcal{F}})]}_{\text{est err}} + \underbrace{\mathbb{E}[E(f^*_{\mathcal{F}}) - E(f^*)]}_{\text{app err}}$$

3.2. Challenges and Tricks

Check recommand reading for more details

- local minima, saddle points
- exploding gradients, vanishing gradients
- inexact gradient
- ⋮

- precondition, early stopping, experiment w/ step-size, ...
- variance-reduced, momentum accelerated, adaptive SGD

3.3. Variance Reduced SGD

$$\min_{w \in \mathbb{R}^d} f(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

Recall $\min \left\{ \mathbb{E}\left[\|\nabla f(w^k)\|^2 \right] \right\}_{k=0}^{k-1} \leq \dfrac{f(w^0) - f^* + \frac{\alpha^2 L}{2} \sum_{k=0}^{k-1} t_k^2}{\sum_{k=0}^{k-1} t_k}$

where $\alpha^2 \geq \mathbb{E}_i\left[\|\nabla f_i(w)\|^2 \right]$ = variance + bias^2

If we stick to unbiased estimator of $\nabla f(w)$

reduce variance → improve the constant in the cvg rate $\frac{C}{\sqrt{k}}$

→ get higher accuracy

- ASGD (Average SGD)

update $w^{k+1} = w^k - \alpha_k \nabla f_{i_k}(w^k)$

return $\frac{1}{K} \sum_{k=1}^{K} w^k$

this do not reduce α^2, but reduce the variance of the output.

- SAG, SAGA: Use an **average** of previous stochastic gradient

SAG: Initialize w^0, $g_i^0 = \nabla f_i(w^0)$, $i=1,\cdots,n$ ③ → high memory cost!

for $k = 1, 2, \cdots$

- uniformly randomly choose i_k from $\{1, 2, \cdots, N\}$
- $g_i^k = \begin{cases} \nabla f_i(w^{k-1}) & \text{if } i = i_k \\ g_i^{k-1} & \text{o.w.} \end{cases}$
- $w^k = w^{k-1} - t_k \frac{1}{n} \sum_{i=1}^{n} g_i^k$

① implementation to reduce computational complexity

$$p^0 = \frac{1}{n} \sum_{i=1}^{n} g_i^0, \quad p^k = p^{k-1} + \frac{1}{n} (g_{i_k}^k - g_{i_k}^{k-1})$$

② SAGA update with direction $p^k = p^{k-1} + (g_{i_k}^k - g_{i_k}^{k-1})$ unbiased

- SVRG, SARAH: Set a **check-point** and compute **full gradient** every m iterations.

SVRG: outer-loop

for $k = 1, 2, \cdots$

- $w = w^{k-1}$, $g = \nabla f(w)$ $O(nd)$
- inner-loop
- $\tilde{w}^0 = w$
- for $s = 1, \cdots, m$
  - pick i_s
  - $\tilde{g}^s = g + (\nabla f_{i_s}(\tilde{w}^{s-1}) - \nabla f_{i_s}(w))$
  - $\tilde{w}^s = \tilde{w}^{s-1} - t_s \tilde{g}^s$
- $w^k = \tilde{w}^m$ or $\tilde{w}^s$, s randomly picked from $\{1, \cdots, m\}$

① total cost: $O(K(nd+m))$ ↑ reduced comparing to SGD

② SARAH: $\tilde{g}^s = \tilde{g}^{s-1} + (\nabla f_{i_s}(\tilde{w}^{s-1}) - \nabla f_{i_s}(w))$

3.4. SGD with Adaptive Step-size and/or Direction

$$w^{k+1} = w^k - t_k \, g^k$$

t_k: learning rate; g^k: gradient.

SGD: $g^k = \nabla f_{i_k}(w^k)$

10 Stochastic Gradient Descent Optimisation Algorithms + Cheatsheet | by Raimi Karim | Towards

Optimiser	Year	Learning Rate	Gradient
Momentum	1964		✓
AdaGrad	2011	✓	
RMSprop	2012	✓	
Adadelta	2012	✓	
Nesterov	2013		✓
Adam	2014	✓	✓
AdaMax	2015	✓	✓
Nadam	2015	✓	✓
AMSGrad	2018	✓	✓

- SGD w/. momentum

$$g^k = \gamma g^{k-1} + (1-\gamma) \nabla f_{i_k}(w^k)$$ typical choice $\gamma = 0.9$.

① smaller γ, forget previous information faster

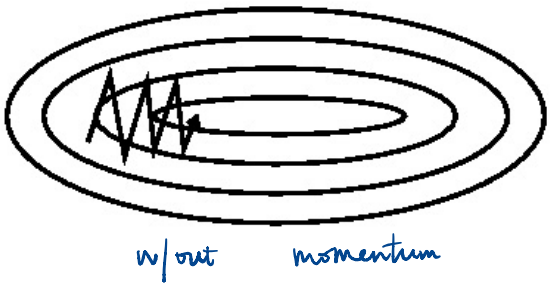
$$g^{k+1} = \gamma(\gamma g^{k-1} + (1-\gamma)\nabla f_{i_k}(w^k)) + (1-\gamma)\nabla f_{i_{k+1}}(w^{k+1})$$

$$= \gamma^2 g^{k-1} + \gamma(1-\gamma)\nabla f_{i_k}(w^k) + (1-\gamma)\nabla f_{i_{k+1}}(w^{k+1})$$

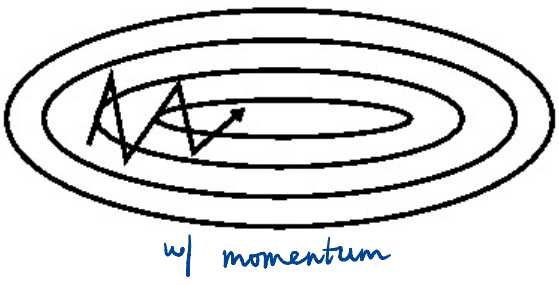
② move quickly in directions with consistent gradients

-- slowly ——— ~~~ inconsistent - - - .

(consistency among samples)



w/out momentum



w/ momentum

https://willamette.edu/~gorr/classes/cs449/momrate.html

- AdaGrad

Let $G_k := \frac{1}{k}\sum_{s=1}^{k}\left(\nabla f_{i_s}(w^s)\,\nabla f_{i_s}(w^s)^T\right)$ (matrix)

$(G_k)_{d,d} = \frac{1}{k}\sum_{s=1}^{k}\left(\nabla_{w_d} f_{i_s}(w^s)\right)^2$ ~ mean^2 + variance

$$w_d^{k+1} = w_d^k - t_k \frac{1}{\sqrt{\varepsilon + (G_k)_{d,d}}} \nabla_{w_d} f_{i_k}(w^k)$$

i.e. $w^{k+1} = w^k - t_k \left(\mathrm{diag}(G_k)\right)^{-\frac{1}{2}} \nabla f_{i_k}(w^k)$

① Pro. adaptive learning rate

lower frequency → smaller mean

more consistent → lower variance

→ larger learning rate

② Con. learning rate sensitive to initialization and decay fast

③ Variants: AdaDelta

* RMSProp: average on recent terms

introduce decay rate γ as momentum method do

✡ . ADAM (one of the most popular alg in deep learning)

SGD $\quad g^{(k)} = \nabla f_{i_k}(w^{(k)})$

momentum — 1-st order moment $\quad m^{(k)} = \beta_1 m^{(k-1)} + (1-\beta_1) g^{(k)}$

(biased)

biase corrected $\quad \hat{m}^{(k)} = \dfrac{m^{(k)}}{(1-\beta_1)^k}$

RMSProp — 2-nd order moment $\quad v^{(k)} = \beta_2 v^{(k-1)} + (1-\beta_2) g^{(k)} \odot g^{(k)}$

(biased)

biase corrected $\quad \hat{v}^{(k)} = \dfrac{v^{(k)}}{(1-\beta_2)^k}$

$$w^{k+1} = w^k - t_k \hat{m}^k \oslash \sqrt{\hat{v}^k + \varepsilon}$$