OPERATOR SPLITTING METHODS

References:

L. Vandenberghe, “Optimization Methods for Large-Scale Systems,” lecture notes, `http://www.seas.ucla.edu/~vandenbe/ee236c.html`

S. Boyd, “Convex Optimization II”, lecture notes, `http://ee364b.stanford.edu`

PROXIMAL MAPPING

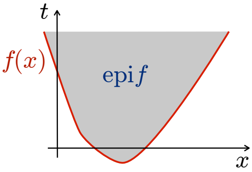
- The **proximal mapping** (or **proximal operator**) of a convex function $f: \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is defined as

$$\operatorname{prox}_f(v) = \arg\min_x \left(f(x) + \frac{1}{2}\|x - v\|_2^2 \right)$$

- We assume also that f is **closed** and **proper**, that is its **epigraph**

$$\operatorname{epi} f = \{(x,t) : \ f(x) \leq t\} \subseteq \mathbb{R}^{n+1}$$

is nonempty, closed and convex.



PROXIMAL MAPPING

- We often use the proximal operator on the scaled function λf with $\lambda > 0$

$$\boxed{\operatorname{prox}_{\lambda f}(v) = \arg\min_x \left(f(x) + \frac{1}{2\lambda}\|x - v\|_2^2 \right)}$$

- The **proximal point** $\operatorname{prox}_{\lambda f}(v)$ of v is a tradeoff between being close to v and minimizing f

- f can be **nonsmooth** and **extended real-valued** ($f(x) = +\infty$ for some x)

- Example: **indicator function** of a convex set $\mathcal{C}$:

$$f(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ +\infty & \text{if } x \notin \mathcal{C} \end{cases} \quad \Longrightarrow \quad \underbrace{\operatorname{prox}_{\lambda f}(v) = \Pi_{\mathcal{C}}(v)}_{\text{projection of } v \text{ on } \mathcal{C}}$$

PROXIMAL POINT ALGORITHM

(Rockafellar, 1976)

- When v is a minimizer of f ($v = x^* \in \arg\min_x f(x)$) we get

$$\mathrm{prox}_{\lambda f}(x^*) = x^*$$

as both terms $f(x)$ and $\frac{1}{\lambda}\|x - x^*\|_2^2$ are minimized at x^*

- The **proximal point algorithm** simply iterates

$$\boxed{x^{k+1} = \mathrm{prox}_{\lambda f}(x^k)}$$

- If f has a minimum, the algorithm converges to an optimizer x^* of f

  (Bauschke, Combettes, 2011)

- The parameter λ may be changed during iterations, as long as $\lambda_k > 0$ and $\sum_{k=0}^{\infty} \lambda_k = +\infty$

PROXIMAL GRADIENT METHOD

(Combettes, Wajs, 2005)

- We want to solve the unconstrained optimization problem

$$\min_x f(x) + g(x)$$

  where

  - $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable with $\operatorname{dom} f = \mathbb{R}^n$
  - $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is convex (possibly non-smooth) with an inexpensive proximal operator

- The **proximal gradient algorithm** (or **forward backward splitting**) iterates

$$x^{k+1} = \operatorname{prox}_{\lambda_k g}\left(x^k - \lambda_k \nabla f(x^k)\right)$$

PROXIMAL GRADIENT METHOD - INTERPRETATION

- The proximal gradient step has the following interpretation:

$$
\begin{aligned}
x^{k+1} &= \mathrm{prox}_{\lambda_k g}\left(x^k - \lambda_k \nabla f(x^k)\right) \\
&= \arg\min_x \left(g(x) + \frac{1}{2\lambda_k}\|x - x^k + \lambda_k \nabla f(x^k)\|_2^2\right) \\
&= \arg\min_x \left(g(x) + \underbrace{f(x^k) + \nabla f(x^k)'(x - x^k) + \frac{1}{2\lambda_k}\|x - x^k\|_2^2}_{\text{simple quadratic model of } f(x) \text{ around } x^k}\right)
\end{aligned}
$$

PROXIMAL GRADIENT METHOD - CONVERGENCE

- If ∇f is **Lipschitz continuous** with constant $L > 0$

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \ \forall x, y \in \mathbb{R}^n$$

then the algorithm converges for all constant $\lambda_k \equiv \lambda \in (0, \frac{1}{L}]$

- Convergence rate: $f(x^k) + g(x^k) - (f(x^*) + g(x^*)) \leq \dfrac{1}{2\lambda k}\|x^0 - x^*\|_2^2$

- If f is strongly convex with parameter $m > 0$[1] then

$$\|x^k - x^*\|_2^2 \leq \left(1 - \frac{m}{L}\right)^k \|x^0 - x^*\|_2^2$$

linear convergence

[1] Remember that f is **strongly convex** with parameter $m > 0$ if and only if $f(y) \geq f(x) + \nabla f(x)'(y - x) + \frac{m}{2}\|y - x\|_2^2$, or equivalently $f(x) - \frac{m}{2}x'x$ convex, or $\nabla^2 f(x) \succeq mI, \forall x \in \mathbb{R}^n$.

PROXIMAL GRADIENT METHOD WITH LINE SEARCH

- If L is not known one can choose λ_k by line search, for example:

  (Beck, Teboulle, 2009)

  - Choose $\beta \in (0,1)$ (e.g., $\beta = \frac{1}{2}$) and set $\lambda \leftarrow \lambda_{k-1}$
  - Repeat

    $z \leftarrow \text{prox}_{\lambda g}(x^k - \lambda \nabla f(x^k))$

    break if $f(z) \leq f(x^k) + \nabla f(x^k)'(z - x^k) + \frac{1}{2\lambda}\|z - x^k\|_2^2$

    update $\lambda \leftarrow \beta\lambda$

  - Return $\lambda_k \leftarrow \lambda$, $x^{k+1} \leftarrow z$

ACCELERATED PROXIMAL GRADIENT METHOD

(Nesterov, 1983) (Beck, Teboulle, 2008)

- The **accelerated** (or **fast**) **proximal gradient algorithm** iterates the following

$$\begin{array}{rcl} y^{k+1} & = & x^k + \beta_k(x^k - x^{k-1}) \\ x^{k+1} & = & \mathrm{prox}_{\lambda_k g}\left(y^{k+1} - \lambda_k \nabla f(y^{k+1})\right) \end{array}$$

extrapolation step

- Possible choices for β_k (with $\beta_0 = 0$) are for example

$$\beta_k = \frac{k-1}{k+2}, \quad \beta_k = \frac{k}{k+3}, \quad \left\{ \begin{array}{rcl} \beta_k & = & \frac{\alpha_k}{\alpha_{k-1}} - \alpha_k \\ \alpha_{k+1} & = & \frac{1}{2}\left(\sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2\right) \\ \alpha_0 & = & \alpha_{-1} = 1^2 \end{array} \right.$$

- Thanks to adding the "momentum term" y^k the initial error $f(x^0) + g(x^0) - (f(x^*) + g(x^*))$ reduces as $1/k^2$

- Same line-search procedure is applicable to select varying λ_k

[2] Any α_k satisfying $\alpha_k^2(1 - \alpha_{k+1}) \leq \alpha_{k+1}^2$ would work

SPECIAL CASES

- Special cases of the (non-accelerated) proximal gradient method:
  - For $g(x) = 0$, $\text{prox}_{\lambda g}(v) = v$ we obtain the standard **gradient descent** method

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

  - For $f(x) = 0$ we obtain the standard **proximal point** method

$$x^{k+1} = \text{prox}_{\lambda_k g}(x^k)$$

  - For $g(x)$ = indicator function of a convex set $\mathcal{C}$ we obtain the **gradient projection** method (Bertsekas, 1999)

$$x^{k+1} = \Pi_{\mathcal{C}}(x^k - \lambda_k \nabla f(x^k))$$

- The accelerated version of the algorithm gives a fast version of the above

(FAST) GRADIENT PROJECTION FOR BOX-CONSTRAINED QP

- Consider the convex box-constrained QP

$$\begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} & \ell \leq x \leq u \end{array}$$

- Since $\|\nabla f(x) - \nabla f(y)\|_2 = \|Q(x-y)\|_2 \leq \lambda_{\max}(Q)\|x-y\|_2$ we can choose any $\lambda \leq \frac{1}{\lambda_{\max}(Q)}$

- The **gradient projection method for box-constrained QP** is

$$x^{k+1} = \max\{\ell, \min\{u, x^k - \lambda(Qx^k + c)\}\}$$

- The **fast gradient projection method for box-constrained QP** is

$$\begin{array}{rcl} y^{k+1} & = & x^k + \beta_k(x^k - x^{k-1}) \\ x^{k+1} & = & \max\{\ell, \min\{u, y^{k+1} - \lambda(Qy^{k+1} + c)\}\} \end{array}$$

DUAL GRADIENT PROJECTION FOR QP

- Consider the strictly convex QP and its dual

$$\begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} & Ax \le b \end{array} \quad \Longrightarrow \quad \begin{array}{ll} \min & \frac{1}{2}y'Hy + d'y \\ \text{s.t.} & y \ge 0 \end{array} \qquad \begin{array}{ll} H & = AQ^{-1}A' \\ d & = b + AQ^{-1}c \end{array}$$

- Take $\lambda \le \frac{1}{\lambda_{\max}(H)}$ ([3]) and apply the proximal gradient method to the dual QP:

$$y^{k+1} = \max\{y^k - \lambda(Hy^k + d), 0\}\} \qquad y_0 = 0$$

dual gradient projection method for QP

- The primal solution is related to the dual solution by

$$x^k = -Q^{-1}(c + A'y^k)$$

[3] Since for any matrix M the largest singular value $\sigma_{\max}(M) = \sqrt{\lambda_{\max}(M'M)}$, we have that $\lambda_{\max}(H) = \sigma^2_{\max}((AC^{-1})') = \sigma^2_{\max}(AC^{-1})$, where $C'C = Q$

ACCELERATED DUAL GRADIENT PROJECTION FOR QP (GPAD)

(Patrinos, Bemporad, 2014)

- The **dual accelerated gradient projection** (GPAD) for QP can be written as

$$\begin{aligned} w^k &= y^k + \beta_k(y^k - y^{k-1}) \\ x^k &= -Kw^k - g \\ s^k &= \tfrac{1}{L}Ax^k - \tfrac{1}{L}b \\ y^{k+1} &= \max\left\{w^k + s^k, 0\right\} \end{aligned}$$

$$\begin{aligned} K &= Q^{-1}A' \\ g &= Q^{-1}c \\ L &\geq \lambda_{\max}(AQ^{-1}A') \end{aligned}$$

- **Termination criteria**: when the following two conditions are met

$$\begin{aligned} s_i^k &\leq \tfrac{1}{L}\epsilon_A,\ i = 1,\ldots,m && \text{primal feasibility} \\ -(w^k)'s^k &\leq \tfrac{1}{L}\epsilon_f && \text{optimality} \end{aligned}$$

the solution $x^k = -Kw^k - g$ satisfies $A_i x^k - b_i \leq \epsilon_A$ and, if $w^k \geq 0$,

$$f(x^k) - f(x^*) \leq f(x^k) - \underbrace{q(w^k)}_{\text{dual fcn}} = -(w^k)'s^k L \leq \epsilon_f$$

RESTART IN FAST GRADIENT PROJECTION

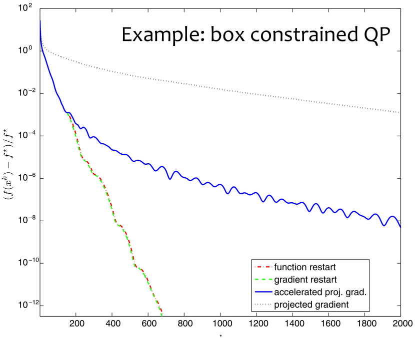
- Fast gradient projection methods can be sped up by adaptively restarting the sequence of coefficients β_k (O'Donoghue, Candés , 2013)

- Restart conditions:
  - **function restart** whenever

$$f(y^k) > f(y^{k-1})$$

  - **gradient restart** whenever

$$\nabla f(w^{k-1})'(y_k - y_{k-1}) > 0$$



PROXIMAL OPERATORS - EXAMPLES

- **indicator function** of a convex set $\mathcal{C}$:

$$f(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ +\infty & \text{if } x \notin \mathcal{C} \end{cases} \quad \Rightarrow \quad \underbrace{\mathrm{prox}_{\lambda f}(v) = \Pi_{\mathcal{C}}(v)}_{\text{projection of } v \text{ on } \mathcal{C}}$$

- **1-norm**: $\mathrm{prox}_{\lambda f}$ is called the **soft-threshold (shrinkage) operator** $S_\lambda : \mathbb{R}^n \to \mathbb{R}^n$

$$f(x) = \|x\|_1 \quad \Rightarrow \quad [\mathrm{prox}_{\lambda f}(v)]_i = [S_\lambda(v)]_i \triangleq \begin{cases} v_i + \lambda & \text{if} \quad v_i \leq -\lambda \\ 0 & \text{if} \quad |v_i| \leq \lambda \\ v_i - \lambda & \text{if} \quad v_i \geq \lambda \end{cases}$$

- **Euclidean norm**:

$$f(x) = \|x\|_2 \quad \Rightarrow \quad \mathrm{prox}_{\lambda f}(v) = \begin{cases} (1 - \lambda/\|v\|_2)v & \text{if } \|v\|_2 \geq \lambda \\ 0 & \text{otherwise} \end{cases}$$

PROXIMAL OPERATORS - EXAMPLES

- **quadratic function**: $Q \succeq 0$

$$f(x) = \frac{1}{2}x'Qx + c'x \quad \Rightarrow \quad \mathrm{prox}_{\lambda f}(v) = (I + \lambda Q)^{-1}(v - \lambda c)$$

- **logarithmic barrier**:

$$f(x) = -\sum_{i=1}^{n} \log x_i \quad \Rightarrow \quad [\mathrm{prox}_{\lambda f}(v)]_i = \frac{v_i + \sqrt{v_i^2 + 4\lambda}}{2},\ i = 1, \ldots, n$$

- Many other examples exist for which the proximal operator can be computed analytically or determined efficiently (for example by bisection)

PROXIMAL OPERATORS - CALCULUS RULES

- **separable sum**:

$$f(x) = \sum_{i=1}^{n} f_i(x_i) \quad \Longrightarrow \quad [\mathrm{prox}_{\lambda f}(v)]_i = \mathrm{prox}_{\lambda f_i}(v_i)$$

- **postcomposition**:

$$f(x) = \alpha\phi(x) + b,\ \alpha > 0 \quad \Longrightarrow \quad \mathrm{prox}_{\lambda f}(v) = \mathrm{prox}_{\alpha\lambda\phi}(v)$$

- **precomposition**:

$$f(x) = \phi(\alpha x + b),\ \alpha \neq 0 \quad \Longrightarrow \quad \mathrm{prox}_{\lambda f}(v) = \frac{1}{\alpha}\left(\mathrm{prox}_{\alpha^2\lambda\phi}(\alpha v + b) - b\right)$$

PROXIMAL OPERATORS - CALCULUS RULES

- **affine addition**:

$$f(x) = \phi(x) + a'x + b \quad \Longrightarrow \quad \mathrm{prox}_{\lambda f}(v) = \mathrm{prox}_{\lambda\phi}(v - \lambda a)$$

- **regularization**: by setting $\tilde{\lambda} = \frac{\lambda}{1+\lambda\rho}$

$$f(x) = \phi(x) + \frac{\rho}{2}\|x - a\|_2^2 \quad \Longrightarrow \quad \mathrm{prox}_{\lambda f}(v) = \mathrm{prox}_{\tilde{\lambda}\phi}\left(\frac{\tilde{\lambda}}{\lambda}v + \rho\tilde{\lambda}a\right)$$

- **Moreau decomposition**: for all functions f it always holds that

$$\boxed{v = \mathrm{prox}_f(v) + \mathrm{prox}_{f^*}(v)}$$

where f^* is the **convex conjugate** (or **Fenchel conjugate**) of f

$$f^*(y) = \sup_x \{y'x - f(x)\}$$

- Calculus rules also exist for computing convex conjugate functions

CONJUGATE FUNCTION AND LAGRANGE DUAL

- Consider the convex optimization problem with linear constraints

$$\begin{array}{rl} \min & f(x) \\ \text{s.t.} & A_i x \leq b_i,\ i \in I \\ & A_i x = b_i,\ i \in E \end{array}$$

- The dual function for the problem is

$$\begin{aligned} q(\lambda) &= \inf_x \{f(x) + \lambda'(Ax - b)\} = -\sup_x \{(-A'\lambda)'x - f(x)\} - b'\lambda \\ &= -f^*(-A'\lambda) - b'\lambda \end{aligned}$$

- If we know the conjugate function f^* we can compute the dual function easily

RELATION WITH INTEGRATION METHODS FOR ODES

(Bruck, 1975) (Botsaris, Jacobson, 1976) (Eckstein, 1989)

- Let f smooth and convex, $\arg\min_x f(x) \neq \emptyset$, and the solution $x(t)$ of the ordinary differential equation (ODE)

$$\frac{dx(t)}{dt} = -\nabla f(x(t)), \quad x(0) = x_0$$

exist. Then $\lim_{t\to\infty} x(t) = x^* \in \arg\min_x f(x)$.

- **gradient descent** = **forward Euler method** for integrating the ODE

$$x^{k+1} = x^k - \lambda_k \frac{dx(x^k)}{dt} = x^k - \lambda_k \nabla f(x^k)$$

- **proximal point method** = **backward Euler method**

$$x^{k+1} = x^k - \lambda_k \nabla f(x^{k+1}) = \arg\min_x \{f(x) + \frac{1}{2\lambda_k}\|x - x^k\|_2^2\} = \text{prox}_{\lambda_k f}(x^k)$$

- **Newton's method** = numerical integration of $\frac{dx}{dt} = -(\nabla^2 f(x))^{-1}\nabla f(x)$

ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM)

(Gabay, Mercier, 1976) (Glowinski, Marrocco, 1975) (Douglas, Rachford, 1956) (Boyd et al., 2010)

- We want to solve the optimization problem

$$\begin{array}{rl} \min_{x,z} & f(x)+g(z) \\ \text{s.t.} & Ax+Bz=c \end{array}$$

$$\begin{array}{l} x\in\mathbb{R}^n,\ z\in\mathbb{R}^m \\ A\in\mathbb{R}^{p\times n},\ B\in\mathbb{R}^{p\times m} \\ c\in\mathbb{R}^p \end{array}$$

where $f:\mathbb{R}^n\to\mathbb{R}\cup\{+\infty\}$, $g:\mathbb{R}^m\to\mathbb{R}\cup\{+\infty\}$ are closed, proper, and convex (possibly non-smooth)

- For a scalar $\rho>0$ we form the **augmented Lagrangian**

$$\mathcal{L}_\rho(x,z,y)=f(x)+g(z)+y'(Ax+Bz-c)+\frac{\rho}{2}\|Ax+Bz-c\|_2^2$$

ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM)

- The **Alternating Direction Method of Multipliers (ADMM)** iterates the following steps

$$
\begin{array}{rcl}
x^{k+1} & = & \arg\min_x \mathcal{L}_\rho(x, z^k, y^k) \\
z^{k+1} & = & \arg\min_z \mathcal{L}_\rho(x^{k+1}, z, y^k) \\
y^{k+1} & = & y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)
\end{array}
$$

- The name “alternating direction” comes from minimizing the augmented Lagrangian $\mathcal{L}_\rho$ first with respect to x and then to z

ADMM - CONVERGENCE

- Assuming that the unaugmented Lagrangian $\mathcal{L}_0$ ($\rho = 0$) has a saddle point, i.e., $\exists (x^*, z^*, y^*)$ such that

$$\mathcal{L}_0(x^*, z^*, y) \leq \mathcal{L}_0(x^*, z^*, y^*) \leq \mathcal{L}_0(x, z, y^*)$$

we have that

$\lim_{k\to\infty} Ax^k + Bz^k - c = 0$ residual convergence

$\lim_{k\to\infty} f(x^k) + g(z^k) = f(x^*) + g(z^*)$ objective convergence

$\lim_{k\to\infty} y^k = y^*$ dual variable convergence

- ADMM has a builtin “integral action”, namely y^k integrates the **primal residual** $r^k = Ax^k + Bz^k - c$

ADMM - STOPPING CRITERIA

- We call **dual residual** the quantity $s^k = \rho A'B(z^{k+1} - z^k)$

- A reasonable termination criterion is to stop the ADMM iterations when

$$\|r^k\|_2 \le \epsilon_{\rm pri} \quad \text{and} \quad \|s^k\|_2 \le \epsilon_{\rm dual}$$

  with

$$\begin{array}{rcl}\epsilon_{\rm pri} & = & \sqrt{p}\epsilon_{\rm abs} + \epsilon_{\rm rel}\max\{\|Ax^k\|_2, \|Bz_k\|_2, \|c\|_2\}\\ \epsilon_{\rm dual} & = & \sqrt{n}\epsilon_{\rm abs} + \epsilon_{\rm rel}\|A'y^k\|_2\end{array}$$

  and $\epsilon_{\rm abs} > 0$ is an absolute tolerance, $\epsilon_{\rm rel} > 0$ a relative tolerance (for example $\epsilon_{\rm rel} = 10^{-3}$ or 10^{-4})

ADMM - VARIANTS

- Convergence sometimes can be improved by introducing **over-relaxation**, that is replacing Ax^{k+1} with

$$\alpha Ax^{k+1} - (1-\alpha)(Bz^k - c)$$

when updating z^{k+1}, y^{k+1}, where $\alpha \in (1,2)$ (typically $\alpha \in [1.5, 1.8]$)

- By introducing the **scaled dual variable** $u = \frac{1}{\rho}y$, ADMM can be expressed in the simplified **scaled form**

$$\begin{aligned}
x^{k+1} &= \arg\min_x \left\{f(x) + \tfrac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2\right\} \\
z^{k+1} &= \arg\min_z \left\{g(z) + \tfrac{\rho}{2}\|Ax^{k+1} + Bz - c + u^k\|_2^2\right\} \\
u^{k+1} &= u^k + Ax^{k+1} + Bz^{k+1} - c
\end{aligned}$$

SCALED ADMM AND PROXIMAL OPERATORS

- Consider the convex problem

$$\min_x \quad f(x) + g(x)$$

$$\begin{array}{rl} \min_{x,z} & f(x) + g(z) \\ \text{s.t.} & x - z = 0 \end{array}$$

- The augmented Lagrangian is

$$\mathcal{L}_\rho(x, z, y) = f(x) + g(z) + y'(x - z) + \frac{\rho}{2}\|x - z\|_2^2$$

- Since $y = \rho u$ and adding $\frac{\rho}{2}\|u\|_2^2$ does not change the minimizer with respect to x and z, we get

$$\arg\min_{x,z} \mathcal{L}_\rho(x, z, y) = \arg\min_{x,z} \left\{ f(x) + g(z) + \frac{\rho}{2}\|x - z + u\|_2^2 \right\}$$

SCALED ADMM AND PROXIMAL OPERATORS

- By letting $\lambda = \frac{1}{\rho}$, the scaled ADMM iterations can be rewritten as

$$
\begin{array}{rcl}
x^{k+1} & = & \arg\min_x \mathcal{L}_\rho(x, z^k, y^k) & = & \mathrm{prox}_{\lambda f}(z^k - u^k) \\
z^{k+1} & = & \arg\min_z \mathcal{L}_\rho(x^{k+1}, z, y^k) & = & \mathrm{prox}_{\lambda g}(x^{k+1} + u^k) \\
u^{k+1} & & & = & u^k + x^{k+1} - z^{k+1}
\end{array}
$$

- The proximal operator calculus can be used for ADMM algorithms too

- An accelerated version of ADMM also exists

ADMM FOR CONSTRAINED CONVEX OPTIMIZATION

- Consider the convex problem with $f, \mathcal{C}$ convex

$$\begin{array}{rl} \min & f(x) \\ \text{s.t.} & x \in \mathcal{C} \end{array} \quad \Rightarrow \quad \begin{array}{rl} \min & f(x) + g(z) \\ \text{s.t.} & x - z = 0 \end{array}$$

  where g is the indicator function of the set $\mathcal{C}$

- The scaled ADMM iterations to solve the problem are

$$\begin{array}{rcl} x^{k+1} & = & \arg\min_x \{f(x) + \frac{\rho}{2}\|x - z^k + u^k\|_2^2\} = \mathrm{prox}_{\frac{1}{\rho}f}(z^k - u^k) \\ z^{k+1} & = & \Pi_{\mathcal{C}}(x^{k+1} + u^k) \\ u^{k+1} & = & u^k + x^{k+1} - z^{k+1} \end{array}$$

- ADMM can be applied to nonconvex $\mathcal{C}$ (e.g., $\mathcal{C} = \{0,1\}^{n_1} \times \mathbb{R}^{n-n_1}$). No guarantee of convergence to a global minimum, but it can be a good heuristic.

(Boyd, Parikh, Chu, Peleato, Eckstein, 2010) (Takapoui, Moehle, Boyd, Bemporad, 2017)

ADMM FOR LINEAR AND QUADRATIC PROGRAMMING

- Consider the standard form QP with Hessian $Q = Q' \succeq 0$

$$\begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array} \quad \Rightarrow \quad \begin{array}{ll} \min & f(x) + g(z) \\ \text{s.t.} & x - z = 0 \end{array}$$

- f is the sum of $\frac{1}{2}x'Qx + c'x$ and the indicator function of $\{x: \ Ax = b\}$

- g is the indicator function of $\mathbb{R}^n_+ = \{x: \ x_i \geq 0, \ i = 1, \ldots, n\}$

- The problem is an LP in standard form when $Q = 0$

- The update for x^{k+1} requires solving

$$\begin{array}{rcll} x^{k+1} & = & \arg\min_x & \frac{1}{2}x'Qx + c'x + \frac{\rho}{2}\|x - z^k + u^k\|_2^2 \\ & & \text{s.t.} & Ax = b \end{array}$$

that is solving the linear system

$$\begin{bmatrix} Q + \rho I & A' \\ A & 0 \end{bmatrix} \begin{bmatrix} x^{k+1} \\ \nu \end{bmatrix} = \begin{bmatrix} \rho(z^k - u^k) - c \\ b \end{bmatrix}$$

- Note that the symmetric matrix $\left[\begin{smallmatrix} Q+\rho I & A' \\ A & 0 \end{smallmatrix}\right]$ can be factorized at start and cached

- The update for z^{k+1} is simply

$$z^{k+1} = \max\{x^{k+1} + u^k, 0\}$$

ADMM FOR QUADRATIC PROGRAMMING

- Consider the QP with Hessian $Q = Q' \succeq 0$, A full column rank or $Q = Q' \succ 0$

$$\begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} & \ell \leq Ax \leq u \end{array} \quad \Longrightarrow \quad \begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x + g(z) \\ \text{s.t.} & Ax - z = 0 \end{array}$$

where g is the indicator function of $\{z : \ \ell \leq z \leq u\}$

- The scaled ADMM iterations to solve the QP are

$$\begin{array}{rcl} x^{k+1} & = & -(Q + \rho A'A)^{-1}(\rho A'(u^k - z^k) + c) \\ z^{k+1} & = & \min\{\max\{Ax^{k+1} + u^k, \ell\}, u\} \\ u^{k+1} & = & u^k + Ax^{k+1} - z^{k+1} \end{array}$$

- We can factorize $Q + \rho A'A$ at start and cache the factorization
- The dual QP solution is also available, as $y^k = \rho u^k$

REGULARIZED ADMM FOR QUADRATIC PROGRAMMING

(Stellato, Banjac, Goulart, Bemporad, Boyd, 2020)

- Consider the QP with Hessian $Q = Q' \succeq 0$

$$\begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x \\ \text{s.t.} & \ell \le Ax \le u \end{array} \quad \Rightarrow \quad \begin{array}{ll} \min & \frac{1}{2}x'Qx + c'x + g(z) \\ \text{s.t.} & Ax - z = 0 \end{array}$$

where g is the indicator function of $\{z : \ell \le z \le u\}$

- Chosen any $\epsilon > 0$, more robust "regularized" ADMM iterations are

$$\begin{array}{rcl} x^{k+1} & = & -(Q + \rho A^T A + \epsilon I)^{-1}(c - \epsilon x^k + \rho A^T (u^k - z^k)) \\ z^{k+1} & = & \min\{\max\{Ax^{k+1} + u^k, \ell\}, u\} \\ u^{k+1} & = & u^k + Ax^{k+1} - z^{k+1} \end{array}$$

- See the osQP solver `https://github.com/oxfordcontrol/osqp`

SUPPLEMENTARY MATERIAL

- By Farkas lemma

  either $\begin{bmatrix} A \\ -A \end{bmatrix} x \leq \begin{bmatrix} u \\ -\ell \end{bmatrix}$ **or** $\begin{bmatrix} A' & -A' \end{bmatrix} \begin{bmatrix} y^+ \\ y^- \end{bmatrix} = 0,\ \begin{bmatrix} u \\ -\ell \end{bmatrix}' \begin{bmatrix} y^+ \\ y^- \end{bmatrix} < 0,\ y^+, y^- \geq 0$

  Then the QP is **infeasible** if a dual vector y exists such that

$$A'y = 0,\ u' \max(y, 0) - l' \max(-y, 0) < 0$$

- The QP is **unbounded** if a primal vector x exists such that

$$Qx = 0, \quad c'x < 0, \quad \begin{cases} A_i x = 0 & l_i, u_i \in \mathbb{R} \\ A_i x \geq 0 & l_i \in \mathbb{R},\ u_i = +\infty \\ A_i x \leq 0 & l_i = -\infty,\ u_i \in \mathbb{R} \end{cases}$$

- In ADMM iterations, y^k (x^k) diverge if the problem is infeasible (unbounded)

DETECTION OF INFEASIBILITY AND UNBOUNDEDNESS

SUPPLEMENTARY MATERIAL

- One can show that
  - $w^k = \frac{y^k}{\|u' \max(y^k,0)+l' \max(-y^k,0)\|}$ asymptotically satisfies Farkas lemma if the QP is infeasible
  - $v^k = \frac{x^k}{-c'x^k}$ asymptotically satisfies the conditions for recognizing unboundedness of the QP

- Alternatively, the increments

$$\delta x^k = x^k - x^{k-1}, \quad \delta y^k = y^k - y^{k-1}, \quad \delta z^k = z^k - z^{k-1}$$

always converge and δy^k (δx^k) also works for recognizing infeasibility (unboundedness) (Banjac, Goulart, Stellato, Boyd, 2017)

ADMM FOR LASSO

- Consider the LASSO problem

$$\min \frac{1}{2}\|Ax-b\|_2^2+\tau\|x\|_1 \quad \Longrightarrow \quad \begin{array}{ll} \min & \frac{1}{2}\|Ax-b\|_2^2+\tau\|z\|_1 \\ \text{s.t.} & x-z=0 \end{array}$$

- The iteration for z is $z^{k+1} = \text{prox}_{\frac{1}{\rho}(\tau\|\cdot\|_1)}(x^{k+1}+u^k)$ = $S_{\frac{\tau}{\rho}}(x^{k+1}+u^k)$ (soft-threshold operator)

- The scaled ADMM iterations to solve the LASSO problem become

$$\begin{array}{rcl} x^{k+1} & = & (A'A+\rho I)^{-1}(A'b+\rho(z^k-u^k)) \\ z^{k+1} & = & S_{\frac{\tau}{\rho}}(x^{k+1}+u^k) \\ u^{k+1} & = & u^k+x^{k+1}-z^{k+1} \end{array}$$

- Since $\rho>0$, $A'A+\rho I$ is always invertible and can be factorized once

CONSENSUS ADMM

- Consider the **separable problem**

$$\boxed{\min_x f(x) = \sum_{i=1}^{N} f_i(x)} \quad x \in \mathbb{R}^n, \quad f_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$$

  with f_i convex and possibly non-smooth

- This may represent a model fitting problem, where x are the parameters of the model and $f_i(x)$ are the losses associated with the ith datapoint

- The problem can be rewritten as the **global consensus problem**

$$\begin{array}{ll} \min & \displaystyle\sum_{i=1}^{N} f_i(x_i) \\ \text{s.t.} & x_i = z, \quad i = 1, \ldots, N \end{array}$$

CONSENSUS ADMM

- Recall the scaled ADMM iterations:

$$\begin{cases} x^{k+1} & = & \arg\min_x \left\{ f(x) + \frac{\rho}{2} \|Ax + Bz^k - c + u^k\|_2^2 \right\} \\ z^{k+1} & = & \arg\min_z \left\{ g(z) + \frac{\rho}{2} \|Ax^{k+1} + Bz - c + u^k\|_2^2 \right\} \\ u^{k+1} & = & u^k + Ax^{k+1} + Bz^{k+1} - c \end{cases}$$

- Here $x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, u = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}, A = I_{nN}, B = -\begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}, c = 0, g(z) = 0$

- In general, if $w = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix}$ then $\|w\|_2^2 = \sum_{i=1}^N \|w_i\|_2^2$. Therefore

$$\left\| \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} - \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix} z + \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix} \right\|_2^2 = \sum_{i=1}^N \|x_i - z + u_i\|_2^2$$

CONSENSUS ADMM

- Moreover $\arg\min_z \sum_{i=1}^N \|x_i - z + u_i\|_2^2 = \arg\min_z \sum_{i=1}^N z'z - 2(x_i + u_i)'z = \frac{1}{N}\sum_{i=1}^N x_i + u_i$
- The scaled ADMM iterations for the consensus problem are therefore

$$
\begin{aligned}
x_i^{k+1} &= \arg\min_{x_i} \left\{ f_i(x_i) + \tfrac{\rho}{2}\|x_i - z^k + u_i^k\|_2^2 \right\} && \text{local/parallel} \\
z^{k+1} &= \tfrac{1}{N}\textstyle\sum_{i=1}^N x_i^{k+1} + u_i^k && \text{global/centralized} \\
u_i^{k+1} &= u_i^k + x_i^{k+1} - z^{k+1} && \text{local/parallel}
\end{aligned}
$$

- The 1st and 3rd steps can be run in parallel, the 2nd step averages $x_i^{k+1} + u_i^k$
- The objectives f_i do not need to be shared!
- A regularization term or indicator function of a constraint $g(z)$ can be included as well ($g(z) = \|z\|_2^2$, $g(z) = \|z\|_1$, ...)

STOCHASTIC GRADIENT METHODS

STOCHASTIC OPTIMIZATION PROBLEM

- We want to minimize

$$\min_x \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

- The problem may come from taking N samples $\xi_1,\ldots,\xi_i$ to approximate

expected value $\quad \min_x E_\xi[\bar f(x;\xi)] \quad \approx \quad \min_x \frac{1}{N}\sum_{i=1}^{N} \bar f(x;\xi_i)$ empirical mean

- In machine learning problems we want to optimize

$$\min_x \frac{1}{N}\sum_{i=1}^{N} \ell(h(u_i;x),y_i)$$

where $(u_1,y_1),\ldots,(u_N,y_N)$ is the training set, $h(u;x)$ a prediction function, $\ell(h,y)$ a loss function

Example: $h(u;x)=x'_{1:n-1}u+x_n$ and $\ell(h,y)=\|h-y\|_2^2$

STOCHASTIC GRADIENT METHOD

(Robbins, Monro, 1951)

- Let $f(x) = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} f_i(x)$
- We solve $\min_x f(x)$ by choosing an index $i_k \in \{1,\ldots,N\}$ **at random** and update

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

stochastic gradient (SG) method

- The step-size α_k is called **learning-rate** in machine learning
- **Pros**: every iteration is extremely cheap (only one gradient is computed)
- **Cons**: descent only in expectation
- The method is an **incremental** (or **online**) **optimization** method (cf. survey paper (Bertsekas, 2012))

STOCHASTIC GRADIENT METHOD

- More generally, the SG method can take the following form:

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k) \qquad \text{single gradient}$$

$$x^{k+1} = x^k - \frac{\alpha_k}{n_k}\sum_{j=1}^{n_k} \nabla f_{i_{k,j}}(x^k) \qquad \textbf{mini-batch}\ (n_k \ll N)$$

$$x^{k+1} = x^k - \frac{\alpha_k}{n_k} H_k \sum_{j=1}^{n_k} \nabla f_{i_{k,j}}(x^k) \qquad \textbf{scaled mini-batch}\ (H_k \in \mathbb{R}^{n\times n})$$

- For $n_k = N$ the resulting **batch gradient method** = **gradient descent** iterations

$$x^{k+1} = x^k - \frac{\alpha_k}{N}\sum_{i=1}^{N} \nabla f_i(x^k)$$

CONVERGENCE ANALYSIS

- If f is continuously differentiable and ∇f Lipschitz continuous with constant[4] L the expectations with respect to i_k (or equivalently ξ_k) satisfy

$$E[f(x^{k+1})] - f(x^k) \leq -\underbrace{\mu\alpha_k\|\nabla f(x^k)\|_2^2}_{\text{expected decrease}} + \underbrace{\frac{1}{2}\alpha_k^2 L E[\|\nabla f_{i_k}(x^k)\|_2^2]}_{\text{variance}} \qquad \mu > 0$$

- Initially f decreases because $\|\nabla f\|$ is large, then variance may dominate

- Therefore we need $\lim_{k\to\infty} \alpha_k = 0$

[4] $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x-y\|_2, \forall x, y \in \mathbb{R}^n$

CONVERGENCE ANALYSIS - STRONGLY CONVEX CASE

- Choose the learning rate $$\boxed{\alpha_k = \frac{\beta}{\gamma + k}} \qquad \beta, \gamma > 0$$
- When f is strongly convex[5] the convergence rate of stochastic gradient descent is **sublinear**
$$E[f(x^k) - f(x^*)] = O\left(\frac{1}{k}\right)$$
- Compare with the **linear** convergence rate of batch gradient
$$f(x^k) - f(x^*) = O(\rho^k), \quad 0 \leq \rho < 1$$
- However, one batch gradient step requires computing N gradients, one SG step only one gradient

[5] $f(y) \geq f(x) + \nabla f(x)'(y - x) + \frac{m}{2}\|y - x\|_2^2, m > 0$. Or equivalently $f(x) - \frac{m}{2}x'x$ convex, or $\nabla^2 f(x) \succeq mI, \forall x$

AVERAGED STOCHASTIC GRADIENT DESCENT

(Ruppert, 1988) (Polyak, Juditsky, 1992)

- Consider the L_2-regularized problem

$$\min_x \frac{\lambda}{2}\|x\|_2^2 + \frac{1}{N}\sum_{i=1}^{N} f_i(x), \quad \lambda > 0$$

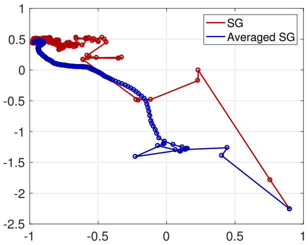
- The idea is to run standard gradient descent but take the average $\bar{x}^k$ after k_0 steps as the optimizer instead of x^k

$$\bar{x}^k = \frac{1}{k-k_0}\sum_{i=k_0+1}^{k} x^i \quad \Longrightarrow \quad \bar{x}^{k+1} = \bar{x}^k + \frac{1}{k+1-k_0}(x^{k+1} - \bar{x}^k)$$

- Choose learning rate

$$\boxed{\alpha_k = \frac{\alpha_0}{(1+\alpha_0\lambda k)^\sigma}}$$

$0 < \sigma < 1$, e.g., $\sigma = \frac{3}{4}$ (Bottou, 2012)



STOCHASTIC GRADIENT DESCENT METHODS

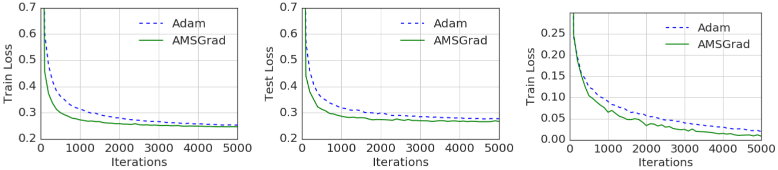
- Despite theory mostly covers the convex case, SGD methods are heavily used to solve nonconvex problems (especially for training deep neural networks)

- Several other popular variants exist with adaptive learning rates α_k:
  - **AdaGrad** (Duchi, Hazan, Singer, 2011)
  - **Adadelta** (Zeiler, 2012)
  - **Adam** (Kingma, Ba, 2015)
  - **Adamax** (Kingma, Ba, 2015)
  - **diffGrad** (Dubey, Chakraborty, Roy, Mukherjee, Singh, Chaudhuri, 2020)
  - ...

- Usually the parameters of the SGD algorithm are tuned on a smaller problem $\min_x \frac{1}{M}\sum_{j=1}^{M} f_{i_j}(x), I = \{i_1, \ldots, i_M\}, M \ll N$

ADAM, AMSGRAD

- Adam (and other variants) use scaling updates by square roots of exponential moving averages of squared past gradients
- An issue in Adam convergence proof has been pointed out and fixed by including a "long-term memory" of past gradients (=largest components encountered of scaling factors)
- The new SGD algorithm, called **AMSGrad**, guarantees convergence and also seems to improve empirical performance (Reddi, Kale, Kumar, 2018)



- **Update: AdamX** further fixes the proof of **AMSGrad** (Phuong, Phong, 2019)

RMSPROP

- **RMSProp**[6] keeps a moving average v_t of the component-wise squared gradient

$$v_j^k = \rho v_j^{k-1} + (1-\rho)[\nabla f_i(x^k)]_j^2$$

for $j = 1, \ldots, n$, where ρ = forgetting factor, and updates

$$x_j^{k+1} = x_j^k - \frac{\alpha}{\epsilon + \sqrt{v_j^k}}[\nabla f_i(x^k)]_j$$

with α = learning rate coefficient and $\epsilon > 0$ prevents division by zero

- Example: $\rho = 0.9$, $\alpha = 10^{-3}$, $\epsilon = 10^{-8}$
- RMSProp extends the **Rprop**[7] algorithm (Riedmiller, Braun, 1992) used in **batch** optimization to the on-line / mini-batch setting
- Heavily used in **deep learning**

---

[6] https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
[7] resilient backpropagation

