HW 6

Evan Ott
UT EID: eao466

October 17, 2016

Proximal operators

(A)

$$\begin{aligned}
f(x) &\approx \hat{f}(x; x_0) = f(x_0) + (x - x_0)^\top \nabla f(x_0) \\
\text{prox}_\gamma \hat{f}(x) &= \arg\min_z \left[\hat{f}(z) + \frac{1}{2\gamma}\|z - x\|_2^2\right] \\
&= \arg\min_z \left[f(x_0) + (z - x_0)^\top \nabla f(x_0) + \frac{1}{2\gamma}\|z - x\|_2^2\right] \\
0 &= \frac{\partial}{\partial z}\left[f(x_0) + (z - x_0)^\top \nabla f(x_0) + \frac{1}{2\gamma}\|z - x\|_2^2\right] \\
&= 0 + \nabla f(x_0) + \frac{1}{2\gamma}(2z^* - 2x) = \nabla f(x_0) + \frac{1}{\gamma}(z^* - x) \\
\text{prox}_\gamma \hat{f}(x) &= z^* = x - \gamma \nabla f(x_0)
\end{aligned}$$

which is indeed the gradient-descent step for $f(x)$ of size γ starting at x_0.

(B)

$$\begin{aligned}
l(x) &= \frac{1}{2}x^\top P x - q^\top x + r \\
\text{prox}_{1/\gamma} l(x) &= \arg\min_z \left[\hat{l}(z) + \frac{\gamma}{2}\|z - x\|_2^2\right] \\
&= \arg\min_z \left[\frac{1}{2}z^\top P z - q^\top z + r + \frac{\gamma}{2}\|z - x\|_2^2\right] \\
0 &= \frac{\partial}{\partial z}\left[\frac{1}{2}z^\top P z - q^\top z + r + \frac{\gamma}{2}\|z - x\|_2^2\right] \\
&= P z^* - q + \gamma(z^* - x) \\
\gamma x + q &= (P + \gamma I) z^* \\
\text{prox}_{1/\gamma} l(x) &= z^* = (P + \gamma I)^{-1}(\gamma x + q)
\end{aligned}$$

assuming $(P + \gamma I)^{-1}$ exists.

If we have $y|x \sim N(Ax, \Omega^{-1})$ with y having n rows, then

$$\begin{aligned}
L(y|x) &= \frac{1}{\sqrt{2\pi}^n}|\Omega|^{-1/2}\exp\left[-\frac{1}{2}(y - Ax)^\top \Omega^{-1}(y - Ax)\right] \\
n(y|x) = -\log L(y|x) &= \frac{1}{2}\log|\Omega| + \frac{n}{2}\log(2\pi) + \frac{1}{2}(y - Ax)^\top \Omega^{-1}(y - Ax) \\
&= \frac{1}{2}y^\top \Omega^{-1} y - (Ax)^\top \Omega^{-1} y + \frac{1}{2}(Ax)^\top \Omega^{-1} Ax + \frac{1}{2}\log|\Omega| + \frac{n}{2}\log(2\pi)
\end{aligned}$$

So $P = \Omega^{-1}$, $q = \Omega^{-1}Ax$ (because $\Omega = \Omega^\top$ since it is a covariance matrix), and $r = \frac{1}{2}(Ax)^\top \Omega^{-1} Ax + \frac{1}{2}\log|\Omega| + \frac{n}{2}\log(2\pi)$.

(C)

$$\begin{aligned}
\phi(x) &= \tau\|x\|_1 \\
\operatorname{prox}_\gamma \phi(x) &= \arg\min_z \left[\phi(z) + \frac{1}{2\gamma}\|z-x\|_2^2\right] \\
&= \arg\min_z \left[\tau\|z\|_1 + \frac{1}{2\gamma}\|z-x\|_2^2\right] \\
&= \arg\min_z \left[\tau\sum_{i=1}^n (|z_i|) + \frac{1}{2\gamma}\sum_{i=1}^n \left((z_i-x_i)^2\right)\right] \\
&= \arg\min_z \left[\sum_{i=1}^n \frac{1}{2\gamma}(z_i-x_i)^2 + \tau|z_i|\right] \\
&= \arg\min_z \left[\sum_{i=1}^n \frac{1}{2}(z_i-x_i)^2 + \tau\gamma|z_i|\right] \qquad \text{(multiplying by positive scalar yields same optimization)}
\end{aligned}$$

The term being minimized for each component z_i is exactly $S_{\tau\gamma}(x_i)$ from the notation last week, and there are no interaction terms between the z_i and z_j for $i \neq j$, so

$$\left(\operatorname{prox}_\gamma \phi(x)\right)_i = S_{\tau\gamma}(x_i)$$

The proximal gradient method

(A)

$$\begin{aligned}
\hat{x} &= \arg\min_x \left\{\tilde{l}(x;x_0) + \phi(x)\right\} \\
&= \arg\min_x \left\{l(x_0) + (x-x_0)^\top \nabla l(x_0) + \frac{1}{2\gamma}\|x-x_0\|_2^2 + \phi(x)\right\} \\
&= \arg\min_z \left\{l(x_0) + (z-x_0)^\top \nabla l(x_0) + \frac{1}{2\gamma}\|z-x_0\|_2^2 + \phi(z)\right\} \\
&= \arg\min_z \left\{\phi(z) + l(x_0) + (z-x_0)^\top \nabla l(x_0) + \frac{1}{2\gamma}\left(z^\top z - 2x_0^\top z + x_0^\top x_0\right)\right\} \\
&= \arg\min_z \left\{\phi(z) + (z-x_0)^\top \nabla l(x_0) + \frac{1}{2\gamma}\left(z^\top z - 2x_0^\top z + x_0^\top x_0\right)\right\} \qquad \text{(add/subtract a constant for same optimization)} \\
&= \arg\min_z \left\{\phi(z) + \frac{\gamma}{2}\left[\nabla l(x_0)\right]^\top \nabla l(x_0) + 2\frac{1}{2\gamma}(z-x_0)^\top \gamma\nabla l(x_0) + \frac{1}{2\gamma}\left(z^\top z - 2x_0^\top z + x_0^\top x_0\right)\right\} \\
&= \arg\min_z \left\{\phi(z) + \frac{1}{2\gamma}\left[\gamma\nabla l(x_0)\right]^\top \gamma\nabla l(x_0) + 2\frac{1}{2\gamma}(z-x_0)^\top \gamma\nabla l(x_0) + \frac{1}{2\gamma}\left(z^\top z - 2x_0^\top z + x_0^\top x_0\right)\right\} \\
&= \arg\min_z \left\{\phi(z) + \frac{1}{2\gamma}\left(\left[\gamma\nabla l(x_0)\right]^\top \gamma\nabla l(x_0) + 2(z-x_0)^\top \gamma\nabla l(x_0) + z^\top z - 2x_0^\top z + x_0^\top x_0\right)\right\} \\
&= \arg\min_z \left[\phi(z) + \frac{1}{2\gamma}\|z - x_0 + \gamma\nabla l(x_0)\|_2^2\right] \\
&= \arg\min_z \left[\phi(z) + \frac{1}{2\gamma}\|z - (x_0 - \gamma\nabla l(x_0))\|_2^2\right] \\
u &= x_0 - \gamma\nabla l(x_0) \\
\hat{x} &= \operatorname{prox}_\gamma \phi(u)
\end{aligned}$$

(B)

Now, we want to play around with our results to cast the lasso regression into a proximal gradient problem.

$$\begin{aligned}
\hat{\beta} &= \arg\min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\} \\
l(\beta|X,y) &= \|y - X\beta\|_2^2 = y^\top y - 2y^\top X\beta + \beta^\top X^\top X\beta \\
\hat{\beta} &= \arg\min_{\beta} \left\{ l(\beta|X,y) + \lambda \|\beta\|_1 \right\} \\
l(\beta|X,y) \approx \hat{l}(\beta|X,y;\beta_0) &= l(\beta_0|X,y) + (\beta - \beta_0)^\top \nabla l(\beta_0|X,y) \\
\nabla l(\beta|X,y) &= 0 - 2X^\top y + 2X^\top X\beta \\
\hat{l}(\beta|X,y;\beta_0) &= \|y - X\beta_0\|_2^2 + (\beta - \beta_0)^\top \left(-2X^\top y + 2X^\top X\beta_0\right)
\end{aligned}$$

Now, in the linear approximation to $l(\beta|X,y)$, we add in the regularization:

$$\tilde{l}(\beta|X,y;\beta_0) = \|y - X\beta_0\|_2^2 + (\beta - \beta_0)^\top \left(-2X^\top y + 2X^\top X\beta_0\right) + \frac{1}{2\gamma}\|\beta - \beta_0\|_2^2$$

Now, we let $l(\beta|X,y) \approx \tilde{l}(\beta|X,y;\beta_0)$ when β is near β_0. This is now exactly the form of surrogate optimization referenced above so

$$\begin{aligned}
\phi(\beta) &= \lambda\|\beta\|_1 \\
u^{(t)} &= \beta^{(t)} - \gamma^{(t)} \nabla l(\beta^{(t)}|X,y) = \beta^{(t)} - \gamma^{(t)} \left(2X^\top X\beta^{(t)} - 2X^\top y\right) \\
\beta^{(t+1)} &= \operatorname{prox}_{\gamma^{(t)}}\phi(u^{(t)}) \\
\beta_i^{(t+1)} &= S_{\lambda\gamma^{(t)}}\left(u_i^{(t)}\right) = \operatorname{sign}\left(u_i^{(t)}\right)\left(\left|u_i^{(t)}\right| - \lambda\gamma^{(t)}\right)_+
\end{aligned}$$

So, to go from step t to step $t+1$, we just compute $u^{(t)}$ then use its components to compute $\beta_i^{(t+1)}$.

There's a relatively high one-time cost to compute $X^\top X$ and $X^\top y$, and (depending on how big p, the number of elements of β, is) this cost carries over each iteration to compute $X^\top X\beta^{(t)}$. That's a $O(p^2)$ calculation (at least in the dense case). Beyond that, the rest of the operations are $O(p)$.

Notes from class Oct. 17

Looking today at dual descent, which is the minimal pre-requisite to understand ADMM (hw 7).

Standard-form convex optimization problem

Note: x will be what we're optimizing.

Minimize $f_0(x)$ subject to $f_i(x) \leq 0$ for $i = 1, \ldots, m$ and $Ax = b$ (affine), with f_0 and all f_i being convex.

Convex set is geometric: take two points in the set, any point on the line between them is also in the set. Convex function is similar: look at the affine transformation (I think linear approximation at a point) is a global under-estimator or not.

Linear program (LP)

Minimize $c^\top x + d$ subject to $Gx \preceq h$ and $Ax = b$ ($\preceq$ means pointwise inequality [applies the $\leq$ operator element-wise].

Quadratic program (QP)

Minimize $\frac{1}{2}x^\top Px + q^\top x + r$ subject to $Gx \preceq h$ and $Ax = b$.

For example, constrained least squares:

minimize $\frac{1}{2}\|Ax-b\|_2^2$ (which is the optimization way of writing $X\beta - y$), constrained by $l \preceq x \preceq u$. That is, $x \preceq u$ and $-x \preceq l$ which is an example of a QP (G would be a block matrix with identity and negative identity).

Slack variables

Similar in idea to latent variables used in MCMC that augment the model, put it in a bigger space, and rewrite the problem.

Example:

$$\underset{x\in\mathbb{R}^D}{\text{minimize}}\ \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1$$

We rewrite it in as

$$\begin{aligned}\underset{x\in\mathbb{R}^D,\ z\in\mathbb{R}^D}{\text{minimize}}\ &\frac{1}{2}\|Ax-b\|_2^2 + \lambda\|z\|_1\\ \text{subject to }&x = z\end{aligned}$$

Example:

$$\underset{x\in\mathbb{R}^D}{\text{minimize}}\ \frac{1}{2}\|x-y\|_2^2 + \lambda\|Dx\|_1$$

where D is an "oriented edge matrix" (see spatial smoothing). Can rewrite as

$$\begin{aligned}\underset{x\in\mathbb{R}^D,\ z\in\mathbb{R}^D}{\text{minimize}}\ &\frac{1}{2}\|x-y\|_2^2 + \lambda\|z\|_1\\ \text{subject to }&Dx = z \ \ \text{feasibility constraint}\end{aligned}$$

Often, most algorithms don't enforce the feasibility constraint until convergence.

Lagrangian

Minimize $f_0(x)$ subject to $f_i(x) =\leq 0$ and $h_i(x) = 0$ (not necessarily convex). Would like to cast this into an unconstrained optimization.

Define

$$I_{\text{-}}(u) = \begin{cases} 0 & u \leq 0 \\ \infty & \text{o.w.}\end{cases}$$

$$I_0(u) = \begin{cases} 0 & u = 0 \\ \infty & \text{o.w.}\end{cases}$$

So now

$$\underset{x\in\mathbb{R}^D}{\text{minimize}} f_0(x) + \sum_{i=1}^{m} I_{\text{-}}(f_i(x)) + \sum_{i=1}^{p} I_0(h_i(x))$$

in other words, any time we're in a case where the constraint is not satisfied, our objective jumps to ∞.

The Lagrangian just linearizes $I_{\text{-}}(u)$ and $I_0(u)$:

$$L(x, \lambda, nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

where λ_i and ν_i are the Lagrange multipliers (also called dual variables). $L(x,\lambda,\nu)$ is the "primal variable," the thing we actually care about.

Now, let's look at the (Lagrange) dual function:

$$g(\lambda,\nu) = \inf_x L(x,\lambda,\nu)$$

Why is this useful?

Fact

For any $\lambda \succeq 0, \nu$, we have $g(\lambda, \nu) \leq p^*$ which is the optimal value of the primal problem ($f_0(x^*)$).

Proof

For any $\lambda \succeq 0, \nu$, we have the following.

$$\sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{x}) \leq 0$$

for any feasibile $\tilde{x}$ (all the h_i are 0, all the $f_i \leq 0$ so this has to be true.

Therefore $L(\tilde{x}, \lambda, \nu) = f_0(\tilde{x}) + \sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{x}) \leq f_0(\tilde{x})$. And

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) \leq L(\tilde{x}, \lambda, \nu) \leq f_0(\tilde{x})$$

this works for any feasible $\tilde{x}$ so it must be true for the optimal value x^*.

Dual problem

$$\underset{\lambda \succeq 0,\ \nu \in \mathbb{R}^p}{\text{maximize}}\ g(\lambda, \nu)$$

This is essentially a minimax problem: we're maximizing the lower bound. Let's say that the optimal value is $d^* = g(\lambda^*,\ \nu^*)$.

Cool thing: strong (Lagrangian) duality is that $p^* = d^*$ which is kind of incredible, and this is actually true sometimes. When? That's the \$1,000,000 question for math careers. However, in stats, it's true in basically all interesting convex problems (mostly). More particularly, under Slater's conditions (see Boyd §5.2).

Now: slight change of notation to match the paper rather than matching the textbook. Dual variables λ, ν will now be denoted y^*.

If strong duality holds, then

$$x^* = \arg\min_x L(x, y^*)$$

where y^* is a dual optimal solution (assuming that L has one minimum). Why do we care? Sometimes it's easier to solve the dual problem than solving the primal problem.

Dual ascent

Now, let's assume that these conditions all hold (strong duality, one minimum, Slater's conditions).

Dual ascent: solve the dual problem by gradient ascent.

y is the dual variable.

Example

minimize $f(x)$ subject to $Ax = b$. What's the lagrangian? Well, $Ax - b = 0$ so

$$L(x, y) = f(x) + y^\top (Ax - b)$$

Dual ascent here is $y^{t+1} = y^t + \alpha^t \nabla g(y)$ where g is the dual function $g(y) = \inf_x L(x, y)$.

How do we evaluate the gradient of the dual function? Think it's called the envelope formula (this is a general property of functions, with some regularities). For $g(y) = \inf_x L(x, y)$, we have:

$$\nabla g(y) = \nabla_y L(x, y)|_{x = \hat{x}(y)}$$

where $\hat{x}(y) = \arg\min_x L(x, y)$

So in this case,

$$\begin{aligned}\nabla g(y) &= \nabla_y \left[f(x) + y^\top (Ax - b)\right] \\ &= Ax - b\end{aligned}$$

which is exactly the residuals of the feasibility constraints. So when $\nabla g(y) = 0$ this gives the extremely interpretable result of having a solution when all constraints are met.

So dual ascent becomes:

$$\begin{aligned}x^{(t+1)} &= \arg\min_x L(x, y^{(t)}) \\ y^{(t+1)} &= y^{(t)} + \alpha^{(t)} \left(Ax^{(t+1)} - b\right)\end{aligned}$$

which is nice because we never actually have to use the dual function. At convergence, $x^{(T)} = x^*$ and $y^{(T)} = y^*$.

This is most of the understanding we need for ADMM, but leaves out the method of multipliers (related to augmented Lagrangian).