MACHINE LEARNING FOR COMPUTATIONAL FINANCE
ASSIGNMENT 2

KARTHIK IYER

Problem 1: *Calculate prox operator*

Recall that

$$\operatorname{prox}_{\alpha f}(z) = \arg\min_x \frac{1}{2\alpha}\|x - z\|^2 + f(x) \tag{0.1}$$

Suppose f is convex and $\alpha > 0$.

(1) Show $\operatorname{prox}_{\alpha f}$ is a single-valued mapping (i.e. there's a unique solution to (0.1)).

Solution: Let $h_z(x) = \frac{1}{2\alpha}\|x - z\|^2 + f(x)$. We wish to find $\arg\min_x h_z(x)$. Note that $h_z(x)$ being sum of two convex functions is itself convex, ($||.||^2$ is strongly convex while f is assumed to be convex.) Moreover, the function $\frac{1}{2\alpha}\|x - z\|^2$ is strictly convex (essentially because any norm coming from an inner product is strictly convex) and hence h_z is strictly convex. We know that a (non-constant) strictly convex function has atmost one global minimizer. Clearly, h_z is non-constant and convex and hence has at most one global minimizer.

Moreover, h_z is also coercive (and continuous) and hence has at least one global minimizer. (See Definition 1.1 and Theorem 1.3 in [1]). h_z is coercive since it is the sum of a strongly convex function and a covex function. A strongly convex function has a quadratic lower bound. If a convex function has a lower bound, then the sum of a strongly convex function and a convex function must shoot off to ∞ as $||x|| \longrightarrow \infty$ thereby showing that the sum is coercive. If the convex function does not have a lower bound, then it must shoot off to $-\infty$ but it cannot have a faster than linear growth while doing so. In either case, the strongly convex part will *dominate* and h_z will shoot off to $+\infty$ as $||x|| \longrightarrow \infty$. Thus h_z has a unique global minimizer thereby proving that $\operatorname{prox}_{\alpha f}$ is a single-valued mapping. ■

(2) Compute $\operatorname{prox}_{\alpha f}$, for $f(x) = \beta\|x\|_1 + (1 - \beta)\|x\|_2^2$.

Solution: Let $h_z = \frac{1}{2\alpha}\|x - z\|^2 + \beta\|x\|_1 + (1-\beta)\|x\|_2^2$. Note that $h_z(x) = \sum_{i=1}^n \frac{1}{2\alpha}\|x_i - z_i\|^2 + \beta\|x_i\|_1 + (1 - \beta)\|x_i\|_2^2$. Since h_z is sum of n one variable 'independent' functions, minimizing h_z is equivalent to minimizing $h_z^i = \frac{1}{2\alpha}\|x_i - z_i\|^2 + \beta\|x_i\|_1 + (1 - \beta)\|x_i\|_2^2$ for each $i = 1, 2, ..., n$.

Cnsider now the problem of minimizing the one variable function $g_z(x) = \frac{1}{2\alpha}\|x - z\|^2 + \beta\|x\|_1 + (1 - \beta)\|x\|_2^2$.

Case 1: Assume that the minimizer $x > 0$.

In this case, $g_z(x) = \frac{1}{2\alpha}\|x - z\|^2 + \beta x_1 + (1 - \beta)x_2^2$. g is minimized when $x = \frac{z-\alpha\beta}{1+2\alpha(1-\beta)}$. Since x is aassumed to be positive, this forces $z > \alpha\beta$.

Case 2: Assume that the minimizer $x < 0$.

Similar to case 1, we can show that g is minimized when $x = \frac{z+\alpha\beta}{1+2\alpha(1-\beta)}$. Since x is aassumed to be negative , this forces $z < -\alpha\beta$.

Case 3: Assume that the minimizer $x = 0$. Since prox is a well-defined mapping, this implies that $x = 0$ for all other values of z i.e when $|z| \leq \alpha\beta$.

Combining everything together, we obtain

$$(\text{prox}_{\alpha f}(z))_i = \begin{cases} \frac{z_i-\alpha\beta}{1+2\alpha(1-\beta)}, \text{ when } z_i > \alpha\beta \\ \frac{z_i+\alpha\beta}{1+2\alpha(1-\beta)}, \text{ when } z_i < -\alpha\beta \\ 0 \text{ when } |z_i| \leq \alpha\beta \end{cases}$$

■

(3) Compute $\text{prox}_{\alpha f}$, for $f(x) = \|x\|_2$. (no square)

Solution: Let $h_z(x) = \frac{1}{2\alpha}\|x - z\|^2 + \|x\|$. Assume that the minimizer x is such that $||x|| > 0$. In this case, h_z becomes a differentiable function of x. Setting the gradient of h_z to 0 gives us $x = z\frac{||x||}{\alpha+||x||}$. This implies that $||z|| = ||x|| + \alpha$. Since we have assumed that $||x|| > 0$, this forces $||z|| > \alpha$.

And for the other possibility (viz $||z|| \leq \alpha$, we obtain $||x|| = 0$ i.e $x = 0$.) Hence

$$\text{prox}_{\alpha f}(z) = \begin{cases} z\left(1 - \frac{\alpha}{||z||}\right) \text{ when } ||z|| > \alpha \\ 0 \text{ when } ||z|| \leq \alpha \end{cases}$$

■

(4) Compute $\text{prox}_{\alpha f}$, for $f(x) = \|x\|_2^2 + \delta_{\mathbb{R}_+^n}(x)$, where

$$\delta_{\mathbb{R}_+^n}(x) = \sum_{i=1}^{n} \begin{cases} 0, & x_i \geq 0 \\ \infty, & x_i < 0 \end{cases}.$$

Proof. Solution: Let $h_z(x) = \frac{1}{2\alpha}\|x-z\|^2+\|x\|_2^2+\delta_{\mathbb{R}_+^n}(x)$. The minimizer for this function, first of all exists (by part (1)) and cannot possibly be outside the first octant since outside the first octant, h_z is ∞. We can thus recast the minimization problem as $\arg\min_{x\in\delta_{\mathbb{R}_+^n}} \frac{1}{2\alpha}\|x - z\|^2 + \|x\|_2^2$.

By the seperability of $\frac{1}{2\alpha}\|x - z\|^2 + \|x\|_2^2$ and $\delta_{\mathbb{R}_+^n}$, the minimization can be thought of as n separate independent minimizations of the form $\arg\min_{x\in\delta_{\mathbb{R}_+}} \frac{1}{2\alpha}|x - z|^2 + |x|_2^2$. Let us now concentrate on the one variable minimization $\arg\min_{x\in\delta_{\mathbb{R}_+}} \frac{1}{2\alpha}|x - z|^2 + |x|_2^2$.

If $z \leq 0$, then we can think of the minimization problem as finding the point $x \in \mathbb{R}_+$ for which the distance of z from $\mathbb{R}_+$ is minimum and also that $|x|$ is minimum. Clearly, 0 is the desired candidate.

Now, assume that $z > 0$. In this case, there are two possible choices for the minimizer x. Either $x > 0$ or $x = 0$. (Since $\mathbb{R}_+$ is a closed set, we separate in to 2 cases to ensure that we can differentiate without worry.)

If $x > 0$, then the first derivative test implies that $x_{min} = \frac{z}{1+2\alpha}$ and this is our candidate for the minimizer. ($x = 0$ cannot be a minimizer since the value of $\frac{1}{2\alpha}|x - z|^2 + |x|_2^2$ at 0 is bigger than the value at x_{min}.)

Combining everything together, we obtain

$$(\mathrm{prox}_{\alpha f}(z))_i = \begin{cases} \frac{z_i}{1+2\alpha}, & \text{when } z_i > 0 \\ 0 \text{ when } z_i \leq 0 \end{cases}$$

■

Problem 2: *Bias-Variance Decomposition*

We know that generalized error could be decomposed into irreducible error, error from bias, and error from variance, using linear regression as a specific example. Consider more general setting:

- $\{y_i, a_i\}_{i=1}^m$ is the training data
- the model is
$$y_i = f(a_i) + \epsilon_i, \quad i = 1, \dots, m$$
where ϵ_i are i.i.d. random variables with mean 0 and variance σ_ϵ^2.
- f is the true data generating mechanism, and $\hat{f}$ is the estimated model.
- $\{y_0, a_0\}$ is a test data point, which also satisfies
$$y_0 = f(a_0) + \epsilon_0.$$

Based on this information show that,

$$\begin{aligned} \mathrm{Err}_{a_0} &= \mathbb{E}\left[\left(y_0 - \hat{f}(a_0)\right)^2\right] \\ &= \sigma_\epsilon^2 + \left(\mathbb{E}\left[\hat{f}(a_0)\right] - f(a_0)\right)^2 + \mathbb{E}\left[\hat{f}(a_0) - \mathbb{E}\left[\hat{f}(a_0)\right]\right]^2 \end{aligned}$$

Proof. Let us denote $\mathbb{E}\left[\hat{f}(a_0)\right] = c$. Then

$$\begin{aligned} &\left(\mathbb{E}\left[\hat{f}(a_0)\right] - f(a_0)\right)^2 + \mathbb{E}\left[\hat{f}(a_0) - \mathbb{E}\left[\hat{f}(a_0)\right]\right]^2 = (c - f(a_0))^2 + \mathbb{E}\left[\hat{f}(a_0) - c\right]^2 \\ &= c^2 + f^2(a_0) - 2cf(a_0) + \mathbb{E}\left[\hat{f}(a_0)^2\right] + c^2 - 2c\mathbb{E}\left[\hat{f}(a_0)\right] \\ &= f^2(a_0) - 2\mathbb{E}\left[\hat{f}(a_0)\right] f(a_0) + \mathbb{E}\left[\hat{f}(a_0)^2\right] \left(\text{since } \mathbb{E}\left[\hat{f}(a_0)\right] = c\right) \\ &= \mathbb{E}\left[\left(f(a_0) - \hat{f}(a_0)\right)^2\right] \end{aligned} \tag{0.2}$$

Hence,

$$\begin{aligned} \mathrm{Err}_{a_0} &= \mathbb{E}\left[\left(y_0 - \hat{f}(a_0)\right)^2\right] = \mathbb{E}\left[\left(y_0 - f(a_0) + f(a_0) - \hat{f}(a_0)\right)^2\right] \\ &= \mathbb{E}\left[(y_0 - f(a_0))^2 + \left(f(a_0) - \hat{f}(a_0)\right)^2 - 2\sigma_\epsilon\left(f(a_0) - \hat{f}(a_0)\right)\right] \\ &= \mathbb{E}\left[\epsilon_0^2 + \left(f(a_0) - \hat{f}(a_0)\right)^2 - 2\epsilon_0\left(f(a_0) - \hat{f}(a_0)\right)\right] \end{aligned} \tag{0.3}$$

Using (0.2), (0.3), $\mathbb{E}(\epsilon_0^2) = Var(\epsilon_0) = \sigma_\epsilon^2$ and the fact that $\mathbb{E}(\epsilon_0) = 0$ and that ϵ_0 and $\hat{f}(a_0)$ are independent random variables (there cannot be any dependence between the model error and the measurement error), we obtain the desired equality.

Thus the prediction error is composed of the irreducible error (the first term), the bias error (the second term) and the variance error (the third term). ■

Problem 3: *Newton's Method* Consider logistic regression problem,

$$\min_x f(x) := \sum_{i=1}^m \{\log(1 + \exp(\langle a_i, x \rangle)) - b_i \langle a_i, x \rangle\} + \frac{\lambda}{2}\|x\|_2^2$$

(1) What is the Hessian of this objective, $\nabla^2 f(x)$?

Proof. Let $A = [a_{ij}]_{m\times n}$. Note that, $\frac{\partial f}{\partial x_j} = \sum_{i=1}^m \left[\frac{a_{ij}}{1+e^{-\langle a_i, x\rangle}} - b_i a_{ij}\right] + \lambda x_j$. Let $p_i = \frac{1}{1+e^{-\langle a_i, x\rangle}}$, the probability vector. We can thus rewrite $\nabla f(x) = A^T(p - b) + \lambda x$.

Note that $\frac{\partial p_i}{\partial x_j} = a_{ij}p_i(1-p_i)$. This combined with the form of $\nabla f(x)$ implies that $\nabla f^2(x) = A^T DA + \lambda I_{n\times n}$ where D is the $m \times m$ diagonal matrix with the jth entry $= p_j(1 - p_j)$. ■

(2) Implement Newton's method in **Jupyter Notebook**. Use the validation dataset to select λ. Report your best test error.

Proof. See the accompanying Jupyter notebook. We get the best test log loss as 0.69642675615051364 for $\lambda = 0.037$. ■

Problem 4: *FISTA* Using the data set from **Problem 3**, pick your favorite penalty $g(x)$ (not limited by the ones we saw on class) and solve logistic regression problem,

(1) What is your g? Calculate $\text{prox}_{\alpha g}$ based the g you provide. Implement the FISTA algorithm in **Jupyter Notebook**, and use validation to select the best λ.

Proof. We choose 3 different g and compare their performace.

(1) $g(x) = t||x||_1 + (1 - t)||x||^2; 0 \le t \le 1$.

For this g, we ran FISTA over a grid of values of t and λ and obtained the following best test log loss.

(i) $t \in [0, 1]$ (30 values, equally spaced) and $\lambda \in [0.01, 0.001]$ (21 values equally spaced).

The best test log loss is 0.69572123490420545 for $t = 0.13793103448275862$ and $\lambda = 0.00235$.

(ii) $t \in [0, 1]$ (30 values, equally spaced) and $\lambda \in [0.1, 0.001]$ (100 values equally spaced). This is very expensive computationally. I timed this in Python and this takes about 722 seconds. The best test log loss is still 0.69572123490420545 and occurs for $t = 0.27586206896551724$ and $\lambda = 0.03$.

(2) $g(x) = ||x||_2$.

For this g, we ran FISTA over a grid of values of λ; $\lambda \in [0.01, 0.001]$ (100 equally spaced values) and obtained a best test log loss of 0.71515597021017085. This is not promising. Let us look for 'smoother and more robust' penalities.

(3) $g(x) = \rho_c(x)$, where ρ_c is the Huber function with parameter c.

Computing the prox for Huber penalty is simple as it is decomposes in to n one variable optimizations where n is such that $x \in \mathbb{R}^n$.

Assume for the time being that $n = 1$. We split up in to 3 mutually exclusive and exhaustive cases. Either $x_0 = \arg\min_x \frac{1}{2\alpha}\|x - z\|^2 + \rho_c(x)$ is such that $|x_0| \le c$ or $x_0 > c$ or $x_0 < -c$. In each of these cases, we can differentiate with respect to x and compute the following expression for x_0:

$$x_0 = \begin{cases} \frac{z}{1+\alpha}, \text{ when } |z| < c(1+\alpha) \\ z - c\alpha, \text{ when } z > c(1+\alpha) \\ z + c\alpha \text{ when } z < -c(1+\alpha) \end{cases}$$

We can combine everything together and obtain the following expression for prox for ρ_c.

$$(\text{prox}_{\alpha\rho_c}(z))_i = \begin{cases} \frac{z_i}{1+\alpha}, \text{ when } |z_i| < c(1+\alpha) \\ z_i - c\alpha, \text{ when } z_i > c(1+\alpha) \\ z_i + c\alpha \text{ when } z_i < -c(1+\alpha) \end{cases}$$

We choose a large value of c (c = 40). (Choosing a small value of c implies we are essentially computing the prox for 1 norm, which we saw earlier leads to a large log loss. A large value of c implies that we are essentially computing the prox for 2 norm square.) We choose a grid of λ values ($\lambda \in [0.01, 0.001]$, 21 equally spaces points) and compute the test log loss for Huber penalty with $c = 40$ to obtain the best test log loss as 0.69642737105302122. (This is almost the same as that for 2 norm penalty, which is to be expected since c is large, implying that we mostly compute the prox for 2 norm square.)

Note that we are not actually achieving a good log loss by using logistic regression for this particular data set. For instance, instead of doing the minimization, if we just set the x to be always 0, then the log loss for this x will be $\log(2)$ which is slightly better than the best log loss we obtained using Huber, elastic net, and $\|x\|_2$ penalty. ■

References

[1] https://sites.math.washington.edu/ burke/crs/408/notes/nlp/unoc.pdf