Introduction to Optimization
AIL 7014
Assignment 2

Name: Trishanku Sarma
Entry ID: 2025AIY7584

September 28, 2025

1 Directional Derivative and Sub-gradients

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, let $x \in \operatorname{dom} f$, and let $d \in \mathbb{R}^n$. Prove that

$$f'(x; d) \;=\; \sup_{g \in \partial f(x)} g^\top d,$$

where the **directional derivative** is defined by

$$f'(x; d) \;=\; \lim_{t \downarrow 0} \frac{f(x + td) - f(x)}{t},$$

and the **subdifferential** of f at x is

$$\partial f(x) \;=\; \big\{ g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \;\; \forall\, y \in \mathbb{R}^n \big\}.$$

Proof

We prove the equality by showing both inequalities.

Step 1: "$\geq$" inequality.
Take any $g_x \in \partial f(x)$. By the subgradient inequality, for all $t > 0$,

$$f(x + td) \;\geq\; f(x) + g_x^\top (x + td - x) = f(x) + t\, g_x^\top d.$$

Dividing by t,

$$\frac{f(x+td)-f(x)}{t} \;\geq\; g_x^\top d.$$

Taking the limit as $t \downarrow 0$,

$$f'(x;d) \;\geq\; g_x^\top d.$$

Since this holds for every $g_x \in \partial f(x)$,

$$f'(x;d) \;\geq \sup_{g_x \in \partial f(x)} g_x^\top d.$$

Step 2: "$\leq$" inequality.
Define

$$\varphi(t) := \frac{f(x+td)-f(x)}{t}, \quad t>0.$$

Since f is convex and $x+td$ is affine in t, the function $\psi(t) := f(x+td)$ is convex in t. Therefore, by the first-order condition for convexity or monotonicity condition, the difference quotients $\varphi(t)$ are non-decreasing in t. Hence

$$f'(x;d) = \inf_{t>0} \varphi(t).$$

Now fix $t>0$. For any $g_{x+td} \in \partial f(x+td)$, the subgradient inequality at $x+td$ with $y=x$ gives

$$f(x) \;\geq\; f(x+td) + g_{x+td}^\top (x-(x+td)) = f(x+td) - t\, g_{x+td}^\top d.$$

Rearranging,

$$\frac{f(x+td)-f(x)}{t} \;\leq\; g_{x+td}^\top d.$$

Thus

$$\varphi(t) \;\leq \sup_{g_{x+td} \in \partial f(x+td)} g_{x+td}^\top d.$$

Taking the infimum over $t>0$,

$$f'(x;d) = \inf_{t>0} \varphi(t) \;\leq \inf_{t>0} \sup_{g_{x+td} \in \partial f(x+td)} g_{x+td}^\top d.$$

By outer semicontinuity of the subdifferential mapping, as $t \downarrow 0$ the sets $\partial f(x+td)$ converge to $\partial f(x)$. Hence, $g_{x+td} \xrightarrow{t\downarrow 0} g_x$

$$f'(x;d) \;\leq \sup_{g_x \in \partial f(x)} g_x^\top d.$$

Step 3: Combine.
From Step 1 and Step 2 we conclude

$$f'(x;d) = \sup_{g \in \partial f(x)} g^\top d.$$

2 Moreau's decomposition and conjugates

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper, closed (lower semicontinuous), convex function. Recall the convex conjugate

$$f^*(y) := \sup_{x \in \mathbb{R}^n} \{\langle x, y\rangle - f(x)\},$$

and the biconjugate $f^{**} := (f^*)^*$. We prove:

(a) (Fenchel–Moreau) $f^{**} = f$.

(b) $y \in \partial f(x) \iff x \in \partial f^*(y)$.

(c) (Moreau decomposition)

$$\operatorname{prox}_f(x) + \operatorname{prox}_{f^*}(x) = x \quad \text{for all } x \in \mathbb{R}^n,$$

where $\operatorname{prox}_f(x) = \arg\min_u \{f(u) + \frac{1}{2}\|u - x\|^2\}$.

Preliminary (Fenchel–Young inequality)

From the definition of $f^*(y) = \sup_z\{\langle z, y\rangle - f(z)\}$
or

$$\langle z, y\rangle - f(z) \le f^*(y)$$

Rearranging the inequality with $z = x$. we get, $\langle x, y\rangle \le f^*(y) + f(x)$

(a) Proof that $f^{**} = f$.
Step 1. $f^{**} \le f$. For any x and y, Taking the supremum in y and Fenchel–Young gives

$$f^{**}(x) = \sup_y\{\langle x, y\rangle - f^*(y)\} \le \langle x, y\rangle - (\langle x, y\rangle - f(x)) \le f(x).$$

Step 2. $f^{**} \ge f$ *(when* f *is closed and convex).*
Fix any $x_0 \in \mathbb{R}^d$ with $f(x_0) < +\infty$. Let $\varepsilon > 0$ and set

$$p := (x_0,\ f(x_0) - \varepsilon).$$

Since $p \notin \operatorname{epi}(f)$ (it lies strictly below the epigraph) and $\operatorname{epi}(f)$ is closed and convex, the separating hyperplane theorem gives the following.

Setup (what we know from separating hyper plane theorem)

There exists $(y, \beta) \in \mathbb{R}^d \times \mathbb{R}$, $(y, \beta) \ne 0$, and a scalar γ such that

$$\langle (y, \beta), (z, t)\rangle \ge \gamma, \quad \text{for all } (z, t) \in \operatorname{epi}(f),$$

$$\langle y, z\rangle + \beta t \ \ge\ \gamma \tag{1}$$

and for the external point p

$$\langle (y, \beta), p\rangle \le \gamma$$

$$\langle y, x_0\rangle + \beta\big(f(x_0) - \varepsilon\big) \ \le\ \gamma. \tag{2}$$

From the two inequalities, we get for every $(z,t) \in \mathrm{epi}(f)$,

$$\langle y, z\rangle + \beta t \;\geq\; \langle y, x_0\rangle + \beta\big(f(x_0) - \varepsilon\big). \tag{S}$$

Observe that $\beta < 0$. Indeed, since $\mathrm{epi}(f)$ contains points with arbitrarily large t, a nonnegative β would make the left-hand side of the inequality unbounded above and prevent a finite γ. Hence $-\beta > 0$.

Algebraic rearrangement. From (S),

$$\langle y, z\rangle - \langle y, x_0\rangle \;\geq\; \beta\big(f(x_0) - \varepsilon - t\big).$$

Multiply by -1 (flip inequality) and then divide by $-\beta > 0$ (no flip):

$$\frac{\langle y, x_0\rangle - \langle y, z\rangle}{-\beta} \;\leq\; f(x_0) - \varepsilon - t.$$

Define $\tilde{y} := \dfrac{y}{-\beta}$. Then $\dfrac{\langle y, x_0\rangle - \langle y, z\rangle}{-\beta} = \langle \tilde{y}, x_0\rangle - \langle \tilde{y}, z\rangle$, so

$$t \;\leq\; f(x_0) - \varepsilon - \big(\langle \tilde{y}, x_0\rangle - \langle \tilde{y}, z\rangle\big).$$

Rearrange to get the useful form

$$\langle \tilde{y}, z\rangle - t \;\leq\; \langle \tilde{y}, x_0\rangle - (f(x_0) - \varepsilon) \qquad \text{for all } (z,t) \in \mathrm{epi}(f).$$

Apply to $t = f(z)$ and take supremum. Plugging $t = f(z)$ (since $(z, f(z)) \in \mathrm{epi}(f)$) yields

$$\langle \tilde{y}, z\rangle - f(z) \;\leq\; \langle \tilde{y}, x_0\rangle - f(x_0) + \varepsilon \qquad \forall z.$$

Taking the supremum over z gives

$$f^*(\tilde{y}) \;\leq\; \langle \tilde{y}, x_0\rangle - f(x_0) + \varepsilon,$$

hence

$$\langle \tilde{y}, x_0\rangle - f^*(\tilde{y}) \;\geq\; f(x_0) - \varepsilon.$$

But $f^{**}(x_0) = \sup_y\{\langle x_0, y\rangle - f^*(y)\}$, so

$$f^{**}(x_0) \;\geq\; \langle \tilde{y}, x_0\rangle - f^*(\tilde{y}) \;\geq\; f(x_0) - \varepsilon.$$

Letting $\varepsilon \downarrow 0$ yields $f^{**}(x_0) \geq f(x_0)$.

Thus, Combining Step 1 and step 2, we obtain $f^{**} = f$ for closed convex f.

(b) Subgradient equivalence: $y \in \partial f(x) \iff x \in \partial f^\star(y)$.

Proof.

($\Rightarrow$) Assume $y \in \partial f(x)$. By the definition of subgradient,

$$f(z) \;\geq\; f(x) + \langle y, z - x\rangle, \qquad \forall z.$$

Rearrange to isolate the conjugate form:

$$\langle z, y\rangle - f(z) \;\leq\; \langle x, y\rangle - f(x), \qquad \forall z.$$

Taking the supremum over z on the left-hand side gives

$$f^\star(y) \;\le\; \langle x, y\rangle - f(x).$$

On the other hand, Fenchel–Young inequality (FY) ensures

$$\langle x, y\rangle \;\le\; f(x) + f^\star(y) \;\iff\; \langle x, y\rangle - f(x) \;\le\; f^\star(y).$$

Combining the two inequalities forces equality:

$$f^\star(y) \;=\; \langle x, y\rangle - f(x).$$

Now, For every w,

$$f^*(w) \;\ge\; \langle x, w\rangle - f(x).$$

Moreover,

$$\langle x, w\rangle - f(x) \;=\; \langle x, y\rangle - f(x) + \langle x, w - y\rangle \;=\; f^*(y) + \langle x, w - y\rangle.$$

$$f^\star(w) \;\ge\; f^\star(y) + \langle x, w - y\rangle, \qquad \forall w.$$

But this is exactly what the above equality implies. Hence

$$x \in \partial f^\star(y).$$

($\Leftarrow$) Conversely, assume $x \in \partial f^\star(y)$. Then by definition,

$$f^\star(w) \;\ge\; f^\star(y) + \langle x, w - y\rangle, \qquad \forall w.$$

Rearrange:

$$\langle x, y\rangle - f^\star(y) \;\ge\; \langle x, w\rangle - f^\star(w), \qquad \forall w.$$

Taking supremum over w on the right-hand side gives

$$\langle x, y\rangle - f^\star(y) \;\ge\; f^{\star\star}(x).$$

Since f is closed and convex, we know $f^{\star\star} = f$, so

$$\langle x, y\rangle - f^\star(y) \;\ge\; f(x).$$

Fenchel–Young inequality (FY) gives the reverse inequality:

$$\langle x, y\rangle - f^\star(y) \;\le\; f(x).$$

Thus equality must hold:

$$f(x) = \langle x, y\rangle - f^\star(y).$$

By the definition of the conjugate,

$$f^\star(y) = \sup_z \{\langle z, y\rangle - f(z)\}.$$

Hence, the fact that equality is achieved at $z = x$ means

$$\langle z, y\rangle - f(z) \;\le\; \langle x, y\rangle - f(x), \qquad \forall z.$$

Rearranging gives

$$f(z) \;\geq\; f(x) + \langle y, z - x\rangle, \qquad \forall z,$$

which is precisely the definition of $y \in \partial f(x)$.

Therefore,

$$y \in \partial f(x) \quad \Longleftrightarrow \quad f(x) + f^\star(y) = \langle x, y\rangle \quad \Longleftrightarrow \quad x \in \partial f^\star(y).$$

(c) Moreau decomposition.

Theorem. Let $f : \mathbb{R}^n \to (-\infty, +\infty]$ be proper, closed, and convex. Then for all $x \in \mathbb{R}^n$,

$$\mathrm{prox}_f(x) + \mathrm{prox}_{f^*}(x) = x.$$

Proof.

Existence and uniqueness of $\mathrm{prox}_f(x)$. Consider the function

$$u \;\mapsto\; f(u) + \tfrac{1}{2}\|u - x\|^2.$$

It is convex (sum of convex functions) and *strongly convex* due to the quadratic term. Therefore it has a unique minimizer, which we denote

$$p := \mathrm{prox}_f(x).$$

First-order optimality condition. The subgradient condition for the minimizer p is

$$0 \in \partial f(p) + (p - x) \quad \Longleftrightarrow \quad x - p \in \partial f(p).$$

Applying subgradient equivalence. From part (b), the subgradient equivalence

$$y \in \partial f(x) \;\Longleftrightarrow\; x \in \partial f^*(y)$$

applied to $x = p$ and $y = x - p$ gives

$$p \in \partial f^*(x - p).$$

Set

$$q := x - p.$$

Since q is $\mathrm{prox}_{f^*}(x)$. By definition,

$$\mathrm{prox}_{f^*}(x) := \arg\min_v \Big(f^*(v) + \tfrac{1}{2}\|v - x\|^2 \Big),$$

and the first-order optimality condition is

$$0 \in \partial f^*(v) + (v - x).$$

Taking $v = q$. Since $p \in \partial f^*(q)$, we have

$$p + (q - x) = p + (x - p - x) = 0,$$

so q satisfies the optimality condition. Hence,

$$\mathrm{prox}_{f^*}(x) = q = x - p.$$

Thus, we can conclude the decomposition. Combining the two proximal operators:

$$\mathrm{prox}_f(x) + \mathrm{prox}_{f^*}(x) = p + q = p + (x - p) = x.$$

Uniqueness. Strong convexity of the quadratic term guarantees that each proximal minimizer is unique. Therefore, the decomposition holds exactly for all $x \in \mathbb{R}^n$.

3 Implementing SGD, SAGA, and Prox-SGD

3.1.1 Data Preparation

Load the sklearn diabetes dataset and split into train/test sets with test_size=0.2 and random_state=42.

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split

data = load_diabetes()
X, y = data.data, data.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Listing 1: Train/Test Split

3.1.2 Built-in Lasso Regression

Use sklearn's Lasso with $\alpha = 0.2$ and report test MSE.

```
# Initialize Lasso with alpha = 0.2, random_state = 42
lassoModel = Lasso(alpha=0.2, random_state=42, max_iter=50000, tol
    =1e-4)
# Fit on training data
lassoModel.fit(X_train, y_train)
# Predict on test data
y_pred_lasso = lassoModel.predict(X_test)
# Compute MSE
mseForLasso = mean_squared_error(y_test, y_pred_lasso)
print("Test MSE (Lasso, alpha = 0.2):", mseForLasso)
```

Listing 2: Sklearn Lasso Example

Test MSE (Lasso, α=0.2): 2820.3666683620127

3.1 Lasso Regression: Loss and Gradients

The Lasso regression objective function is:

$$f(w) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - x_i^\top w)^2 + \alpha \|w\|_1,$$

where n is the number of training examples, $x_i \in \mathbb{R}^d$ is the feature vector, $y_i \in \mathbb{R}$ is the target, $w \in \mathbb{R}^d$ is the weight vector including bias w_0, and α controls L1 sparsity.

The gradient can be split into:

$$\nabla f_{\text{smooth}}(w) = \frac{1}{n} \sum_{i=1}^{n} x_i (x_i^\top w - y_i), \quad \nabla f_{\text{L1}}(w) = \alpha \, \text{sign}(w), \quad \text{with } \text{sign}(0) = 0.$$

And $\nabla f(w) = \nabla f_{\text{smooth}}(w) + \nabla f_{\text{L1}}(w)$

3.2 Update Rules for Lasso

Notation: w_k — weights at iteration k, η — step size, i — randomly selected example, g_i — stored gradient for example i, $\bar{g}$ — average of stored gradients.

- **SGD:**

$$w_{k+1} = w_k - \eta\big(\nabla f_{i(smooth)}(w_k) + \alpha\,\mathrm{sign}(w_k)\big)$$

  Where

$$\nabla f_{\mathrm{i(smooth)}}(w) = x_i(x_i^\top w_k - y_i),$$

  Gradient step on a single example; L1 handled via sign term. Bias w_0 is not penalized.

- **SAGA (Variance-reduced SGD):** SAGA reduces variance of SGD by storing gradients. The update at iteration t proceeds as follows:

  1. **Initialization:**

$$g_i = 0 \quad \forall i \in \{1, \ldots, n\}, \quad w_0 \text{ initialized randomly or to zero}$$

  2. **Iteration t:**
     (a) Sample an index i_t uniformly from $\{1, \ldots, n\}$.
     (b) Compute variance-reduced gradient:

$$G_t = \nabla f_{i_t(smooth)}(w_t) - g_{i_t} + \frac{1}{n}\sum_{i=1}^{n} g_i$$

     (c) Update weights:

$$w_{t+1} = w_t - \eta_t G_t$$

     (d) Update stored gradient:

$$g_{i_t} \leftarrow \nabla f_{i_t}(w_t)$$

  Bias w_0 is not penalized; L1 regularization can be applied separately using a proximal operator if desired.

- **Proximal SGD:**

$$w_{k+1} = \mathrm{prox}_{\eta\alpha}\big(w_k - \eta\nabla f_{i(smooth)}(w_k)\big), \quad \mathrm{prox}_{\eta\alpha}(v) = \mathrm{sign}(v)\max(|v| - \eta\alpha, 0)$$

  Gradient step on smooth loss, then apply soft-thresholding to enforce sparsity. Bias w_0 is not penalized.

3.3 Implementation from Scratch

3.3.1 Hyperparameters

- Batch size: 1
- Step size: 0.01(fixed)
- Maximum Number of iterations: 200000
- Regularization $\alpha = 0.2$

Convergence Check

The stopping criterion checks both stabilization of the loss and changes in model parameters. Given a weight vector w, previous weight vector w_{prev}, and the loss history $\{\mathcal{L}_1, \ldots, \mathcal{L}_t\}$:

1. Compute the rolling average of the previous *window* losses:

$$\text{avg\_prev\_loss} = \frac{1}{window} \sum_{i=t-window}^{t-1} \mathcal{L}_i$$

2. Let the current loss be:

$$\text{curr\_loss} = \mathcal{L}_t$$

3. Check loss stabilization:

$$|\text{avg\_prev\_loss} - \text{curr\_loss}| < \text{tol\_loss} \quad \Rightarrow \quad \text{converged}$$

4. Check parameter change:

$$\|w - w_{\text{prev}}\|_2 < \text{tol\_w} \quad \Rightarrow \quad \text{converged}$$

5. If either condition is satisfied, set a flag:

$$\text{converged} = \text{True}, \quad \text{reason} = \text{“loss stabilized” or “param change”}$$

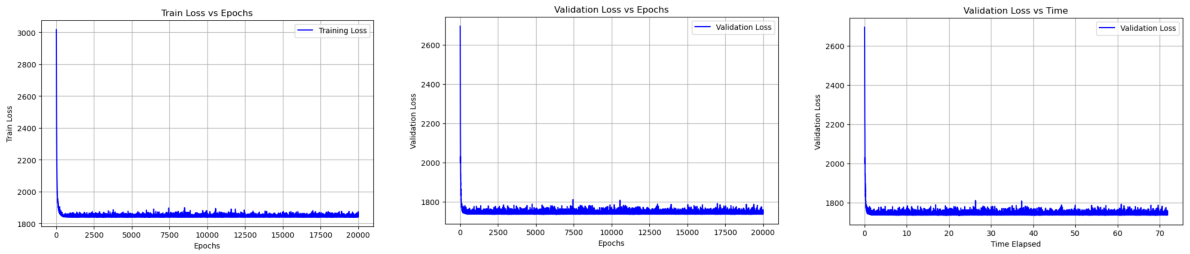
Here, *tol_loss* and *tol_w* are user-defined tolerances, and *window* is the number of previous epochs considered for rolling average.

3.3.2 Results obtained after experiments with differnet set of algorithms

Method	**Iterations Ran**	**Time (s)**	**Test MSE**
Regular SGD (fixed-step-size)	20000(max)	72.375268459	2840.06976999
Regular SGD (Decaying step size)	19360	69.980312824	2831.53896928
SAGA (variance-reduced)	896	11.0953724384	2820.93520790
Proximal GD	9291	38.366797924	2834.51455110
Backtracking LS (SGD + line search)	20000 (max)	223.11412596	2823.84566215

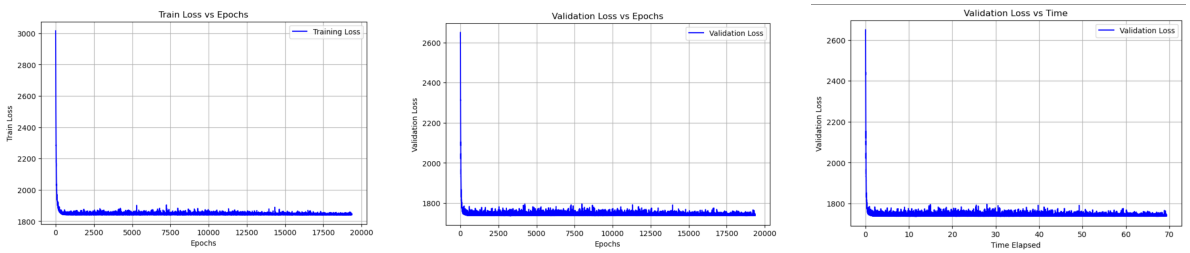
Table 1: Comparison of optimization methods: convergence epochs, wall-clock time, and test MSE.

3.3.3 Training Curves



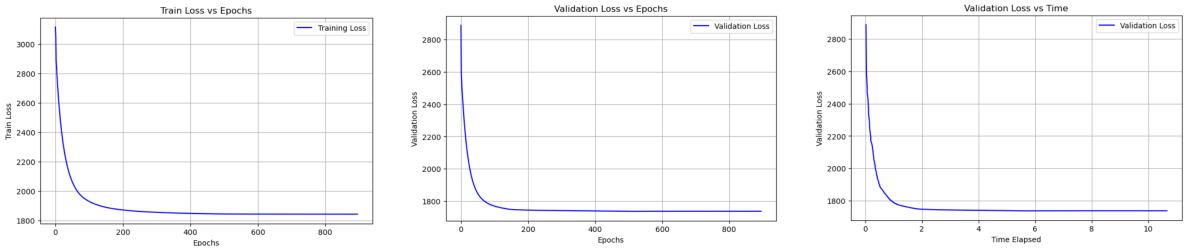
(a) train loss vs epochs (b) validation loss vs epochs (c) validation loss vs time elapsed

Figure 1: Plot for SGD With fixed step size = 0.01 and maxEpochs = 20000



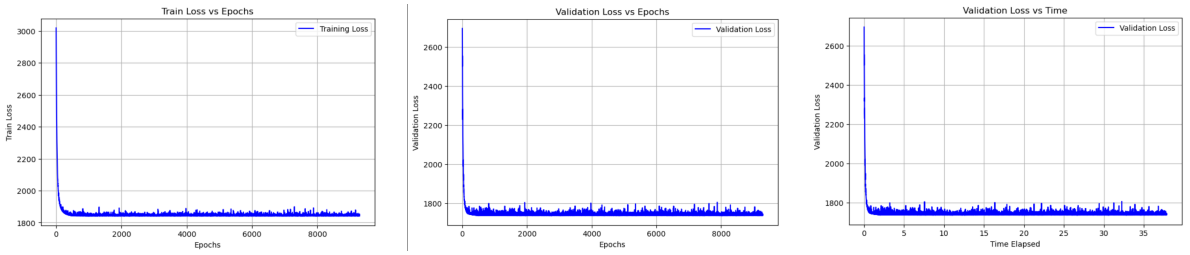
(a) train loss vs epochs (b) validation loss vs epochs (c) validation loss vs time elapsed

Figure 2: Plot for SGD With decaying step size after 10000 epocs



(a) train loss vs epochs (b) validation loss vs epochs (c) validation loss vs time elapsed

Figure 3: Plot for SAGA



(a) train loss vs epochs (b) validation loss vs epochs (c) validation loss vs time elapsed

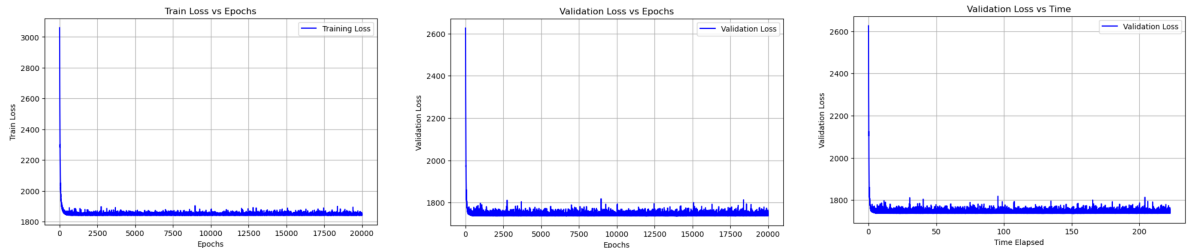
Figure 4: Plot for SGD With Proximal GD

3.4 Backtracking Line Search on SGD

Implement backtracking line search and compare convergence.

```
def backtracking_line_search(X, y, w, grad_w, alpha=0.2, gamma
   =0.02, tau=0.5):
    step_size = 0.01
    while True:
        w_new = w - step_size * grad_w
        f_new = lasso_objective_loss(X, y, w_new, X.shape[0],
           alpha)
        f_curr = lasso_objective_loss(X, y, w, X.shape[0], alpha)
        if f_new <= f_curr - gamma * step_size * (grad_w.T @
           grad_w):
            break
        step_size *= tau
    return step_size
```

Listing 3: Backtracking Line Search



(a) train loss vs epochs (b) validation loss vs epochs (c) validation loss vs time elapsed

Figure 5: Plot for Backtracking line search(maxm epochs = 20000)

3.4.1 Observations and Conclusion

On implementing back tracking line search, it was observed that the test MSE got slightly improved than Regular SGD but still SAGA performed the best of all models.(Table 3.3.2). But failed to converge in 20000 epochs but decaying step size did converged under these iterations.