Introduction to Machine Learning

Fall 2022

University of Science and Technology of China

Lecturer: Jie Wang — Homework 5

Name: Yunqin Zhu — ID: PB20061372

Notice, to get the full credits, please present your solutions step by step.

Exercise 1: Proximal Operator

For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, we define its proximal operator at $\mathbf{x}$ by

$$\text{prox}_f(\mathbf{x}) = \underset{\mathbf{u} \in \textbf{dom } f}{\arg\min} \left\{ f(\mathbf{u}) + \frac{1}{2}\|\mathbf{u} - \mathbf{x}\|^2 \right\}.$$

1. Recall the convex optimization problem Lecture 08.

$$\min_{\mathbf{x} \in \mathbb{R}^n} F(\mathbf{x}).$$

Please rewrite $p(\mathbf{x}_c)$ using proximal operator.

Solution:

$$\begin{aligned} p(\mathbf{x}_c) &= \underset{\mathbf{x} \in \mathbb{R}^n}{\arg\min} \left\{ g(\mathbf{x}) + \frac{L}{2} \left\| \mathbf{x} - \left(\mathbf{x}_c - \frac{1}{L}\nabla f(\mathbf{x}_c) \right) \right\|^2 \right\} \\ &= \underset{\mathbf{x} \in \mathbb{R}^n}{\arg\min} \left\{ \frac{1}{L} g(\mathbf{x}) + \frac{1}{2} \left\| \mathbf{x} - \left(\mathbf{x}_c - \frac{1}{L}\nabla f(\mathbf{x}_c) \right) \right\|^2 \right\} \\ &= \text{prox}_{\frac{1}{L}g} \left(\mathbf{x}_c - \frac{1}{L}\nabla f(\mathbf{x}_c) \right) \end{aligned}$$

■

2. The proximal operator has the following properties.

(a) If f is proper and closed (which means $\textbf{epi}f$ is closed), then for any $\mathbf{x} \in \mathbb{R}^n$, $\text{prox}_f(\mathbf{x})$ exists and is unique. You can use the properties we have proved in Homework 4 directly.

(b) If f is proper and closed, then $\mathbf{u} = \text{prox}_f(\mathbf{x})$ if and only if $\mathbf{x} - \mathbf{u} \in \partial f(\mathbf{u})$.

(c) Please show that if $\mathbf{u} = \text{prox}_f(\mathbf{x}), \mathbf{v} = \text{prox}_f(\mathbf{y})$, then

$$\langle \mathbf{u} - \mathbf{v}, \mathbf{x} - \mathbf{y} \rangle \geq \|\mathbf{u} - \mathbf{v}\|_2^2,$$

which means prox_f is firmly nonexpansive. Then show that this implies nonexpansive

$$\left\|\text{prox}_f(\mathbf{x}) - \text{prox}_f(\mathbf{y})\right\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_2.$$

Solution:

(a) Note that $f(\mathbf{u}) + \frac{1}{2}\|\mathbf{u} - \mathbf{x}\|^2$ is proper, closed and strongly convex as a sum of the closed strongly convex function $\frac{1}{2}\|\mathbf{u} - \mathbf{x}\|^2$ on $\mathbb{R}^n$ and the proper closed convex function $f(\mathbf{u})$. Therefore, there exists a unique minimizer to the problem

$$\min_{\mathbf{u}\in\mathbf{dom}\, f} \left\{ f(\mathbf{u}) + \frac{1}{2}\|\mathbf{u}-\mathbf{x}\|^2 \right\}, \tag{1}$$

which is $\operatorname{prox}_f(\mathbf{x})$ by definition.

(b) We have shown that the convex optimization problem (1) admits a unique minimizer $\operatorname{prox}_f(\mathbf{x})$. By the optimality condition, $\mathbf{u} = \operatorname{prox}_f(\mathbf{x})$ if and only if $\mathbf{0} \in \partial f(\mathbf{u}) + \mathbf{u} - \mathbf{x}$, i.e. $\mathbf{x} - \mathbf{u} \in \partial f(\mathbf{u})$.

(c) According to (b), we have $\mathbf{x}-\mathbf{u} \in \partial f(\mathbf{u})$ and $\mathbf{y}-\mathbf{v} \in \partial f(\mathbf{v})$. Then,

$$\begin{aligned} \langle \mathbf{x}-\mathbf{u}, \mathbf{v}-\mathbf{u}\rangle &\le f(\mathbf{v}) - f(\mathbf{u}),\\ \langle \mathbf{y}-\mathbf{v}, \mathbf{u}-\mathbf{v}\rangle &\le f(\mathbf{u}) - f(\mathbf{v}). \end{aligned}$$

Summing up the two inequalities, we obtain

$$\langle (\mathbf{u}-\mathbf{v}) - (\mathbf{x}-\mathbf{y}), \mathbf{u}-\mathbf{v}\rangle \le 0.$$

Hence $\langle \mathbf{u}-\mathbf{v}, \mathbf{x}-\mathbf{y}\rangle \ge \|\mathbf{u}-\mathbf{v}\|_2^2$. Using the Cauchy-Schwarz inequality, it follows that

$$\|\mathbf{u}-\mathbf{v}\|_2^2 \le \|\mathbf{u}-\mathbf{v}\|_2\|\mathbf{x}-\mathbf{y}\|_2 \iff \|\mathbf{u}-\mathbf{v}\|_2 \le \|\mathbf{x}-\mathbf{y}\|_2. \qquad \blacksquare$$

3. The proximal operator satisfies the following equations.

(a) For $\lambda \neq 0$, $a \in \mathbb{R}^n$, we let $h(\mathbf{x}) = f(\lambda\mathbf{x}+\mathbf{a})$, then $\operatorname{prox}_h(\mathbf{x}) = \frac{1}{\lambda}\left(\operatorname{prox}_{\lambda^2 f}(\lambda\mathbf{x}+\mathbf{a}) - \mathbf{a}\right)$.

(b) For $\lambda > 0$, we let $h(\mathbf{x}) = \lambda f\left(\frac{\mathbf{x}}{\lambda}\right)$, then $\operatorname{prox}_h(\mathbf{x}) = \lambda \operatorname{prox}_{\lambda^{-1} f}\left(\frac{\mathbf{x}}{\lambda}\right)$.

(c) For $\mathbf{a} \in \mathbb{R}^n$, we let $h(\mathbf{x}) = f(\mathbf{x}) + \mathbf{a}^\top\mathbf{x}$, then $\operatorname{prox}_h(\mathbf{x}) = \operatorname{prox}_f(\mathbf{x}-\mathbf{a})$.

Solution:

(a)

$$\begin{aligned} \operatorname{prox}_h(\mathbf{x}) &= \operatorname*{arg\,min}_{\mathbf{u}\in\mathbf{dom}\, h} \left\{ f(\lambda\mathbf{u}+\mathbf{a}) + \frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2 \right\} \\ &= \frac{1}{\lambda}\left(\operatorname*{arg\,min}_{\mathbf{v}\in\mathbf{dom}\, f} \left\{ f(\mathbf{v}) + \frac{1}{2}\left\|\frac{\mathbf{v}-\mathbf{a}}{\lambda} - \mathbf{x}\right\|_2^2 \right\} - \mathbf{a}\right) \\ &= \frac{1}{\lambda}\left(\operatorname*{arg\,min}_{\mathbf{v}\in\mathbf{dom}\, f} \left\{ \lambda^2 f(\mathbf{v}) + \frac{1}{2}\left\|\mathbf{v} - (\lambda\mathbf{x}+\mathbf{a})\right\|_2^2 \right\} - \mathbf{a}\right) \\ &= \frac{1}{\lambda}\left(\operatorname{prox}_{\lambda^2 f}(\lambda\mathbf{x}+\mathbf{a}) - \mathbf{a}\right). \end{aligned}$$

(b)

$$\begin{aligned} \operatorname{prox}_h(\mathbf{x}) &= \operatorname*{arg\,min}_{\mathbf{u}\in\mathbf{dom}\, h} \left\{ \lambda f(\frac{\mathbf{u}}{\lambda}) + \frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2 \right\} \\ &= \lambda \operatorname*{arg\,min}_{\mathbf{v}\in\mathbf{dom}\, f} \left\{ \lambda f(\mathbf{v}) + \frac{1}{2}\left\|\lambda\mathbf{v}-\mathbf{x}\right\|_2^2 \right\} \\ &= \lambda \operatorname*{arg\,min}_{\mathbf{v}\in\mathbf{dom}\, f} \left\{ \lambda^{-1} f(\mathbf{v}) + \frac{1}{2}\left\|\mathbf{v}-\frac{\mathbf{x}}{\lambda}\right\|_2^2 \right\} \\ &= \lambda \operatorname{prox}_{\lambda^{-1} f}\left(\frac{\mathbf{x}}{\lambda}\right). \end{aligned}$$

(c)
$$\begin{aligned}\operatorname{prox}_h(\mathbf{x}) &= \operatorname*{arg\,min}_{\mathbf{u}\in\mathbf{dom}\,h}\left\{f(\mathbf{u})+\mathbf{a}^\top\mathbf{u}+\frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2\right\}\\ &= \operatorname*{arg\,min}_{\mathbf{u}\in\mathbf{dom}\,h}\left\{f(\mathbf{u})+\frac{1}{2}\|\mathbf{u}-(\mathbf{x}-\mathbf{a})\|_2^2\right\}\\ &= \operatorname{prox}_f(\mathbf{x}-\mathbf{a}).\end{aligned}$$ ■

4. Please find the proximal operator of the following functions.
   (a) $f(\mathbf{x}) = 0$.
   (b) $f(\mathbf{x}) = \|\mathbf{x}\|_2$
   (c) $f(\mathbf{x}) = I_C(\mathbf{x})$, where C is a convex set.
   (d) $f(\mathbf{x}) = \|\mathbf{x}\|_1$.

   Solution:

   (a) $\operatorname{prox}_f(\mathbf{x}) = \operatorname{arg\,min}_{\mathbf{u}}\left\{\frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2\right\} = \mathbf{x}$.

   (b) $\operatorname{prox}_f(\mathbf{x}) = \operatorname{arg\,min}_{\mathbf{u}}\left\{\|\mathbf{u}\|_2+\frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2\right\}$. Letting $\mathbf{x}-\mathbf{u}\in\partial\|\mathbf{u}\|_2$, we have

   $$\begin{cases}\mathbf{x}-\mathbf{u}=\frac{\mathbf{u}}{\|\mathbf{u}\|_2}, & \text{if } \mathbf{u}\neq\mathbf{0},\\ \mathbf{x}-\mathbf{u}\in B(0,1), & \text{if } \mathbf{u}=\mathbf{0}.\end{cases}$$

   Therefore, $\operatorname{prox}_f(\mathbf{x}) = \begin{cases}\mathbf{x}-\frac{\mathbf{x}}{\|\mathbf{x}\|_2}, & \text{if } \|\mathbf{x}\|_2>1,\\ \mathbf{0}, & \text{if } \|\mathbf{x}\|_2\le 1.\end{cases}$

   (c) $\operatorname{prox}_f(\mathbf{x}) = \operatorname{arg\,min}_{\mathbf{u}}\left\{I_C(\mathbf{x})+\frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2\right\} = \operatorname{arg\,min}_{\mathbf{u}\in C}\left\{\frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2\right\} = \Pi_C(\mathbf{x})$.

   (d) $\operatorname{prox}_f(\mathbf{x}) = \operatorname{arg\,min}_{\mathbf{u}}\left\{\|\mathbf{u}\|_1+\frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2\right\}$. Letting $\mathbf{x}-\mathbf{u}\in\partial\|\mathbf{u}\|_1$, we have

   $$\begin{cases}x_i-u_i=+1, & \text{if } u_i>0,\\ x_i-u_i\in[-1,+1], & \text{if } u_i=0,\\ x_i-u_i=-1, & \text{if } u_i<0.\end{cases}$$

   Therefore, $\mathbf{u}=\operatorname{prox}_f(\mathbf{x})$ satisfies

   $$\begin{cases}u_i=x_i-1, & \text{if } x_i>1,\\ u_i=0, & \text{if } |x_i|\le 1,\\ u_i=x_i+1, & \text{if } x_i<-1.\end{cases}$$ ■

5. (Optional) Consider the convex optimization problem

   $$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})+\tilde{I}_D(\mathbf{x}), \tag{2}$$

   where $D\subseteq\mathbb{R}^n$ is a closed convex set and $\tilde{I}_D(\mathbf{x})$ is the extended-value extension of its indicator function $I_D(\mathbf{x})$.

   (a) Write down the optimality condition and the proximal operator of the problem (2).

(b) Find the relationship between (2) and the constrained optimization problem

$$\min_{\mathbf{x}\in D} f(\mathbf{x}).$$

Solution:

(a) The optimality condition is $\mathbf{0} \in \partial f(\mathbf{x}) + \partial \tilde{I}_D(\mathbf{x})$, i.e. $-\mathbf{g} \in N_D(\mathbf{x})$ for some $\mathbf{g} \in \partial f(\mathbf{x})$, where $N_D(\mathbf{x})$ is the normal cone of D at $\mathbf{x}$. To be explicit,

$$\text{there exists } \mathbf{g} \in \partial f(\mathbf{x}) \text{ such that } \langle \mathbf{g}, \mathbf{y} - \mathbf{x}\rangle \geq 0 \text{ for all } \mathbf{y} \in D.$$

The proximal operator is $\operatorname{prox}_{f+\tilde{I}_D}(\mathbf{x}) = \arg\min_{\mathbf{u}\in D}\left\{f(\mathbf{u}) + \frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2\right\}$. Moreover, $\mathbf{u} = \operatorname{prox}_{f+\tilde{I}_D}(\mathbf{x})$ iff. $\mathbf{u} \in D$ and $\mathbf{x} - \mathbf{u} - \mathbf{g} \in N_D(\mathbf{u})$ for some $\mathbf{g} \in \partial f(\mathbf{u})$. That is,

$$\text{there exists } \mathbf{g} \in \partial f(\mathbf{u}) \text{ such that } \langle \mathbf{g} + \mathbf{u} - \mathbf{x}, \mathbf{v} - \mathbf{u}\rangle \geq 0 \text{ for all } \mathbf{v} \in D.$$

(b) Suppose that $D \subseteq \mathbf{dom}\, f$. Then $f(\mathbf{x}) + \tilde{I}_D(\mathbf{x}) = f(\mathbf{x}) < +\infty$ for any $\mathbf{x} \in D$, whereas $f(\mathbf{x}) + \tilde{I}_D(\mathbf{x}) = +\infty$ for any $\mathbf{x} \notin D$. It follows that $\mathop{\mathbf{argmin}}_{\mathbf{x}\in\mathbb{R}^n}\left\{f(\mathbf{x}) + \tilde{I}_D(\mathbf{x})\right\} = \mathop{\mathbf{argmin}}_{\mathbf{x}\in D} f(\mathbf{x})$, i.e. the constrained optimization problem shares the same optimal solution as the unconstrained one. ■

6. (Optional) Write down the proximal operator of the following convex optimization problems.

$$\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1\|\mathbf{w}\|_1 + \lambda_2 I_{\mathbb{R}^n_+}(\mathbf{w}),$$

Solution:
Let $F(\mathbf{w}) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1\|\mathbf{w}\|_1$. The proximal operator is

$$\operatorname{prox}_{F+\lambda_2 I_{\mathbb{R}^n_+}}(\mathbf{w}) = \arg\min_{\mathbf{u}\in\mathbb{R}^n_+}\left\{\frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{u}\|_2^2 + \lambda_1\|\mathbf{u}\|_1 + \frac{1}{2}\|\mathbf{u}-\mathbf{w}\|_2^2\right\}.$$

By the optimality condition, $\mathbf{u} = \operatorname{prox}_{F+\lambda_2 I_{\mathbb{R}^n_+}}(\mathbf{w})$ if and only if $\mathbf{u} \in \mathbb{R}^n_+$ and $\mathbf{g} + \mathbf{u} - \mathbf{w} \in -N_{\mathbb{R}^n_+}(\mathbf{u})$ for some $\mathbf{g} \in \partial F(\mathbf{u})$. Since $\partial F(\mathbf{u}) + \mathbf{u} - \mathbf{w} = \lambda_1 \partial\|\mathbf{u}\|_1 + \tilde{\mathbf{u}} - \mathbf{w}$, where $\tilde{\mathbf{u}} = \left(\frac{2}{n}\mathbf{X}^\top\mathbf{X} + \mathbf{I}\right)\mathbf{u} - \frac{2}{n}\mathbf{X}^\top\mathbf{y}$, we obtain

$$\begin{cases} \lambda_1 + \tilde{u}_i - w_i \geq 0, & \text{if } u_i = 0, \\ \lambda_1 + \tilde{u}_i - w_i = 0, & \text{if } u_i > 0, \end{cases}$$

Therefore, $\lambda\mathbf{1} + \tilde{\mathbf{u}} - \mathbf{w} = \mathbf{0}$, and thus

$$\operatorname{prox}_{F+\lambda_2 I_{\mathbb{R}^n_+}}(\mathbf{w}) = \left(\frac{2}{n}\mathbf{X}^\top\mathbf{X} + \mathbf{I}\right)^{-1}\left(\frac{2}{n}\mathbf{X}^\top\mathbf{y} + \mathbf{w} - \lambda_1\mathbf{1}\right).$$ ■

Exercise 2: Proximal Gradient

Consider the following convex optimization problem

$$\begin{aligned} &\min_{\mathbf{x}} F(\mathbf{x}) \\ &\text{s.t.}\, \mathbf{x} \in D \end{aligned} \tag{3}$$

where $F : \mathbb{R}^n \to \bar{\mathbb{R}}$ is a proper convex function and $D \subseteq \mathbb{R}^n$ is a nonempty convex set with $D \subseteq$ **dom** F. Suppose that the problem (3) is solvable, and **we do not require the differentiability of** F.

1. If $\mathbf{x} \in \textbf{int}\,(\textbf{dom } F) \cap D$ and there exists a $\mathbf{g} \in \partial F(\mathbf{x})$ such that

$$\langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle \geq 0, \forall\, \mathbf{y} \in D,$$

show that $\mathbf{x}$ is optimal.

Solution:
By the definition of subdifferential,

$$F(\mathbf{y}) \geq F(\mathbf{x}) + \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle \geq F(\mathbf{x}), \forall\, \mathbf{y} \in D,$$

which implies that $\mathbf{x}$ is optimal. ■

2. (Optional) If $\mathbf{x} \in \textbf{int}\,(\textbf{dom } F)$ and $\mathbf{x}$ is optimal, show that $\mathbf{x} \in D$ and there exists a $\mathbf{g} \in \partial F(\mathbf{x})$ such that

$$\langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle \geq 0, \forall\, \mathbf{y} \in D.$$

Solution:
By Exercise 1.5(b), the problem is equivalent to $\min_{\mathbf{x}} F(\mathbf{x}) + I_D(\mathbf{x})$. Since $\mathbf{x}$ is optimal, we have

$$F(\mathbf{y}) \geq F(\mathbf{x}) = F(\mathbf{x}) + \langle \mathbf{0}, \mathbf{y} - \mathbf{x} \rangle, \forall\, \mathbf{y} \in D,$$

which implies that $\mathbf{0} \in \partial F(\mathbf{x}) + \partial I_D(\mathbf{x})$. Thus, there exists a $\mathbf{g} \in \partial F(\mathbf{x})$ such that $\mathbf{g} \in -N_D(\mathbf{x})$, where $\partial I_D(\mathbf{x}) = N_D(\mathbf{x})$ is the normal cone of D at $\mathbf{x}$. More explicitly, $\langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle \geq 0, \forall\, \mathbf{y} \in D$. ■

3. Please give an example to show that $\partial F(\mathbf{x})$ can be empty.

Solution:
$F(x) = -\sqrt{x}$ is convex on **dom** $F = [0, \infty)$ but $\partial F(0) = \emptyset$. ■

4. If $\mathbf{x}^*$ is an interior point of D, show that

$$\mathbf{x}^* \in \mathop{\textbf{argmin}}_{\mathbf{x} \in D} F(\mathbf{x}) \iff 0 \in \partial F(\mathbf{x}^*).$$

You can use the conclusion of Problems 1 and 2.

Solution:
By Problem 2, $\mathbf{x}$ is optimal if and only if $F(\mathbf{y}) \geq F(\mathbf{x}) = F(\mathbf{x}) + \langle \mathbf{0}, \mathbf{y} - \mathbf{x} \rangle$, $\forall\, \mathbf{y} \in D$, which is equivalent to $\mathbf{0} \in \partial F(\mathbf{x}) + \partial I_D(\mathbf{x})$. For any $\mathbf{x}^* \in \textbf{int}\ D$, we have $\partial I_D(\mathbf{x}^*) = N_D(\mathbf{x}^*) = \mathbf{0}$, and thus $\mathbf{x}^* \in \textbf{argmin}_{\mathbf{x}\in D}\ F(\mathbf{x}) \iff 0 \in \partial F(\mathbf{x}^*)$. ■

In many cases, the function F can be decomposed into $F = f + g$, where $g : \mathbb{R}^n \to \bar{\mathbb{R}}$ is a continuous convex function, and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex and continuously differentiable function, whose gradient is Lipschitz continuous with the constant L. We can use ISTA, which has been introduced in Lecture 08, to find $\min\limits_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x})$.

5. For a **given** point $\mathbf{x}_c$, we consider the following quadratic approximation of F:

$$Q(\mathbf{x}; \mathbf{x}_c) = f(\mathbf{x}_c) + \langle \nabla f(\mathbf{x}_c), \mathbf{x} - \mathbf{x}_c \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}_c\|^2 + g(\mathbf{x}).$$

Please show that it always admits a unique minimizer

$$p(\mathbf{x}_c) = \arg\min_{\mathbf{x}\in\mathbb{R}^n} Q(\mathbf{x}; \mathbf{x}_c).$$

Solution:
By Exercise 1.1, $p(\mathbf{x}_c) = \mathrm{prox}_{\frac{1}{L}g}\left(\mathbf{x}_c - \frac{1}{L}\nabla f(\mathbf{x}_c)\right)$. Since $\frac{1}{L}g$ is proper, convex and closed, by Exercise 1.2(a), $\mathrm{prox}_{\frac{1}{L}g}(\mathbf{x})$ exists and is unique, so is $p(\mathbf{x}_c)$. Note that the closedness of $\frac{1}{L}g$ follows from its continuity. ■

6. (Optional) We can think of the update step of ISTA, i.e., $\mathbf{x}^+ = p(\mathbf{x})$, as two steps:

(a) Take a step in the opposite direction of the gradient of f at $\mathbf{x}$, i.e.,

$$\mathbf{z} = \mathbf{x} - \frac{1}{L}\nabla f(\mathbf{x}).$$

(b) Project $\mathbf{z}$ on some set C, i.e.,

$$\mathbf{x}^+ = p(\mathbf{x}) = \Pi_C(\mathbf{z}).$$

Find the set C. Is it closed, open or neither? Is it convex or not?

Solution:
Let $C = \{\mathbf{x} : g(\mathbf{x}) \leq g(\mathbf{x}^+)\}$. Then, on the one hand,

$$\min_{\mathbf{x}} \left\{ \|\mathbf{x} - \mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x}) \right\} \leq \min_{\mathbf{x}\in C} \left\{ \|\mathbf{x} - \mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x}) \right\} \leq \min_{\mathbf{x}\in C} \|\mathbf{x} - \mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x}^+).$$

On the other hand, noting that $\mathbf{x}^+ = \arg\min_{\mathbf{x}} \left\{ \|\mathbf{x} - \mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x}) \right\}$,

$$\min_{\mathbf{x}} \left\{ \|\mathbf{x} - \mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x}) \right\} = \|\mathbf{x}^+ - \mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x}^+) \geq \min_{\mathbf{x}\in C} \|\mathbf{x} - \mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x}^+).$$

Therefore, $\min_{\mathbf{x}} \left\{\|\mathbf{x}-\mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x})\right\} = \min_{\mathbf{x}\in C} \|\mathbf{x}-\mathbf{z}\|^2 + \frac{2}{L}g(\mathbf{x}^+)$, and thus

$$\mathbf{x}^+ = \arg\min_{\mathbf{x}\in C} \|\mathbf{x}-\mathbf{z}\|^2 = \Pi_C(\mathbf{z}).$$

Note that C is a level set of the closed convex function g, so it is closed and convex. ■

7. Consider the Lasso problem

$$\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{n}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1.$$

Suppose that $\hat{\mathbf{w}}$ solves the problem. Write down the optimality condition at $\hat{\mathbf{w}}$.

Solution:
The optimality condition is $\mathbf{0} \in \nabla f(\hat{\mathbf{w}}) + \partial g(\hat{\mathbf{w}})$, where $f(\mathbf{w}) = \frac{1}{n}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2$ and $g(\mathbf{w}) = \lambda\|\mathbf{w}\|_1$. So

$$\mathbf{0} \in \frac{2}{n}\left(\mathbf{X}^\top\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}^\top\mathbf{y}\right) + \lambda\partial\|\mathbf{w}\|_1.$$

Letting $\tilde{\mathbf{w}} = \frac{2}{n}\left(\mathbf{X}^\top\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}^\top\mathbf{y}\right)$, we obtain

$$\begin{cases} 0 = \tilde{w}_i + \lambda, & \text{if } \hat{w}_i > 0, \\ 0 \in \tilde{w}_i + \lambda[-1,1], & \text{if } \hat{w}_i = 0, \\ 0 = \tilde{w}_i - \lambda, & \text{if } \hat{w}_i < 0. \end{cases}$$ ■

8. If we use ISTA to solve the Lasso problem, show that

$$w_i^+ = \begin{cases} z_i + \dfrac{\lambda}{L}, & \text{if } z_i < -\dfrac{\lambda}{L}, \\ 0, & \text{if } |z_i| \le \dfrac{\lambda}{L}, \\ z_i - \dfrac{\lambda}{L}, & \text{if } z_i > \dfrac{\lambda}{L}, \end{cases}$$

where $\mathbf{z} = \mathbf{w}_k - \dfrac{2}{Ln}\mathbf{X}^\top(\mathbf{X}\mathbf{w}_k - \mathbf{y})$.

Solution:
By Exercise 1.1, $\mathbf{w}^+ = p(\mathbf{w}_k) = \operatorname{prox}_{\frac{\lambda}{L}\|\cdot\|_1}(\mathbf{z})$. The optimality condition becomes

$$\mathbf{z} - \mathbf{w}^+ \in \frac{\lambda}{L}\partial\|\mathbf{w}^+\|_1.$$

And hence

$$\begin{cases} z_i - w_i^+ = \frac{\lambda}{L}, & \text{if } w_i^+ > 0, \\ z_i - w_i^+ \in \frac{\lambda}{L}[-1,1], & \text{if } w_i^+ = 0, \\ z_i - w_i^+ = -\frac{\lambda}{L}, & \text{if } w_i^+ < 0. \end{cases}$$

The result follows. ■

Exercise 3: Projected Gradient Descent (Optional)

Consider the following problem

$$\min_{x \in D} f(x), \tag{4}$$

where $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is proper, continuously differentiable and strongly convex with convexity parameter $\mu > 0$. We assume that the gradient of f is Lipschitz with a constant $L > 0$.

A commonly used approach to solve the constrained optimization problem (4) is the so-called *projected gradient descent*, in which each iteration improves the current estimation $\mathbf{x}_k$ of the optimum by

$$\mathbf{x}_{k+1} = \Pi_D(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)),$$

where $\alpha > 0$ is the step size.

1. Show that

$$f(\mathbf{y}) \geq f(\mathbf{x}) - \frac{1}{2\mu}\|\nabla f(\mathbf{x})\|_2^2,\ \forall\, \mathbf{x}, \mathbf{y} \in D.$$

Solution:
By strong convexity, we have

$$\begin{aligned} f(\mathbf{y}) &\geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \\ &= f(\mathbf{x}) + \frac{1}{2\mu}\left\|\nabla f(\mathbf{x}) + \mu\left(\mathbf{y} - \mathbf{x}\right)\right\|_2^2 - \frac{1}{2\mu}\|\nabla f(\mathbf{x})\|_2^2 \\ &\geq f(\mathbf{x}) - \frac{1}{2\mu}\|\nabla f(\mathbf{x})\|_2^2. \end{aligned}$$

■

2. Consider the problem (4) and the sequence generated by the *projected gradient descent* algorithm. Suppose that $\mathbf{x}^*$ is the solution to the problem (4).

(a) Find the range of α such that the function values $f(\mathbf{x}_k)$ converge linearly to $f(\mathbf{x}^*)$.

(b) When does the (projected) gradient descent always achieve the optimal solution in one iteration wherever the intial point $\mathbf{x}_0$ is?

Solution:

(a) Lipschitz continuity of ∇f implies that

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \leq \langle \nabla f(\mathbf{x}_k), \mathbf{x}_{k+1} - \mathbf{x}_k \rangle + \frac{L}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2. \tag{5}$$

Note that $\mathbf{x}_{k+1} - \mathbf{x}_k = \Pi_D(\mathbf{x}_k - \alpha\nabla f(\mathbf{x}_k)) - \Pi_D(\mathbf{x}_k)$. Using both the nonexpansiveness and the firm nonexpansiveness of Π_D, the above inequality becomes

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \leq -\frac{1}{\alpha}\|\alpha\nabla f(\mathbf{x}_k)\|_2^2 + \frac{L}{2}\|\alpha\nabla f(\mathbf{x}_k)\|_2^2 = -\alpha(1 - \frac{L}{2}\alpha)\|\nabla f(\mathbf{x}_k)\|_2^2,$$

where the equality holds if $\mathbf{x}_k - \alpha\nabla f(\mathbf{x}_k) \in D$ and the equality in (5) holds. To ensure that $\{f(\mathbf{x}_k)\}$ converges to f^* monotoneously for an arbitrary f, we must have $\alpha(1 - \frac{L}{2}\alpha) > 0$, i.e. $0 < \alpha < \frac{2}{L}$, such that $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ strictly unless the sufficient optimality condition $\nabla f(\mathbf{x}_k) = \mathbf{0}$ is satisfied. Then, we can use the result in Problem 1 to derive

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \le -\mu\alpha(2 - L\alpha)(f(\mathbf{x}_k) - f(\mathbf{x}^*)).$$

Subtracting $f(\mathbf{x}^*)$ from both sides and rearranging, we have

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \le (1 - \mu\alpha(2 - L\alpha))(f(\mathbf{x}_k) - f(\mathbf{x}^*)),$$

which leads to

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \le (1 - \mu\alpha(2 - L\alpha))^k(f(\mathbf{x}_0) - f(\mathbf{x}^*)) \xrightarrow{\text{linear}} 0.$$

Here, we have used the fact that $0 < \mu\alpha(2 - L\alpha) \le L\alpha(2 - L\alpha) \le 1$. To conclude, the range of α is $(0, \frac{2}{L})$.

(b) When $1 - \mu\alpha(2 - L\alpha) = 0$, i.e. $\alpha = \frac{1}{L} = \frac{1}{\mu}$, the (projected) gradient descent algorithm achieves the optimal solution in one iteration. ■

Exercise 4: [2] ISTA with Backtracking

Suppose that we would like to apply ISTA to solve the convex optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}), \tag{6}$$

where $g : \mathbb{R}^n \to \bar{\mathbb{R}}$ is a continuous convex function, and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex and continuously differentiable function, whose gradient is Lipschitz continuous with the constant L. We assume that the problem (6) is solvable, i.e., there exists $\mathbf{x}^*$ such that

$$F(\mathbf{x}^*) = F^* = \min_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x}).$$

In practice, however, a possible drawback of ISTA is that the Lipschitz constant L is not always known or computable. For instance, if $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$, the Lipschitz constant for ∇f depends on $\lambda_{\max}(\mathbf{A}^\top\mathbf{A})$, which is not always easily computable for large-scale problems. To tackle this problem, we always equip ISTA with the backtracking stepsize rule as shown in Algorithm 1.

Note that in Algorithm 1, Q_L and p_L are defined as

$$\begin{aligned} Q_L(\mathbf{x};\mathbf{x}_c) &= f(\mathbf{x}_c) + \langle \nabla f(\mathbf{x}_c), \mathbf{x} - \mathbf{x}_c\rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{x}_c\|_2^2 + g(\mathbf{x}) \\ p_L(\mathbf{x}_c) &= \mathop{\mathbf{argmin}}_{\mathbf{x}\in\mathbb{R}^n} Q_L(\mathbf{x};\mathbf{x}_c). \end{aligned}$$

Algorithm 1 ISTA with Backtracking

```
1: Input: An initial point $\mathbf{x}_0$, an initial constant $L_0 > 0$, a threshold $\eta > 1$, and $k = 1$.
2: while the termination condition does not hold do
3:     Find the smallest non-negative integer $i_k$ such that with $\tilde{L} = \eta^{i_k} L_{k-1}$
           $F(p_{\tilde{L}}(\mathbf{x}_{k-1})) \le Q_{\tilde{L}}(p_{\tilde{L}}(\mathbf{x}_{k-1});\mathbf{x}_{k-1}).$   (7)
4:     $L_k \leftarrow \eta^{i_k} L_{k-1}$, $\mathbf{x}_k \leftarrow p_{L_k}(\mathbf{x}_{k-1})$,
5:     $k \leftarrow k + 1$,
```

1. Show that the sequence $\{F(\mathbf{x}_k)\}$ produced by Algorithm 1 is non-increasing.

   Solution:
   The inequality (7) shows that

$$F(\mathbf{x}_k) \le f(\mathbf{x}_{k-1}) + \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{x}_k - \mathbf{x}_{k-1}\rangle + \frac{L_k}{2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2 + g(\mathbf{x}_k). \tag{8}$$

   Since $\mathbf{x}_k = \operatorname{prox}_{\frac{1}{L_k}g}(\mathbf{x}_{k-1} - \frac{1}{L_k}\nabla f(\mathbf{x}_{k-1}))$, by Exercise 1.2(b), we have

$$\mathbf{x}_{k-1} - \frac{1}{L_k}\nabla f(\mathbf{x}_{k-1}) - \mathbf{x}_k \in \frac{1}{L_k}\partial g(\mathbf{x}_k).$$

Using the definition of subdifferential, we obtain

$$\left\langle \mathbf{x}_{k-1} - \frac{1}{L_k}\nabla f(\mathbf{x}_{k-1}) - \mathbf{x}_k, \mathbf{x} - \mathbf{x}_k \right\rangle \leq \frac{1}{L_k} g(\mathbf{x}) - \frac{1}{L_k} g(\mathbf{x}_k), \quad \forall \mathbf{x} \in \partial g(\mathbf{x}_k). \tag{9}$$

Combining (8) and (9) yields

$$\begin{aligned} F(\mathbf{x}_k) &\leq f(\mathbf{x}_{k-1}) + \langle \nabla f(\mathbf{x}_{k-1}), \mathbf{x} - \mathbf{x}_{k-1} \rangle + g(\mathbf{x}) \\ &\qquad - \frac{L_k}{2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2 + L_k \langle \mathbf{x}_{k-1} - \mathbf{x}_k, \mathbf{x}_{k-1} - \mathbf{x} \rangle \\ &\leq F(\mathbf{x}) - \frac{L_k}{2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2 + L_k \langle \mathbf{x}_{k-1} - \mathbf{x}_k, \mathbf{x}_{k-1} - \mathbf{x} \rangle \\ &= F(\mathbf{x}) - \frac{L_k}{2}\left(\|\mathbf{x}_k - \mathbf{x}\|_2^2 - \|\mathbf{x}_{k-1} - \mathbf{x}\|_2^2\right). \end{aligned} \tag{10}$$

Letting $\mathbf{x} = \mathbf{x}_{k-1}$ in the above inequality yields

$$F(\mathbf{x}_k) \leq F(\mathbf{x}_{k-1}) - \frac{L_k}{2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2 \leq F(\mathbf{x}_{k-1}),$$

which implies that the sequence $\{F(\mathbf{x}_k)\}$ is non-increasing. ■

2. Show that the inequality (7) is satisfied for any $\tilde{L} \geq L$, where L is the Lipschitz constant of ∇f, thus showing that for Algorithm 1 one has $L_k \leq \eta L$ for every $k \geq 1$.

   Solution:
   By Newtom-Leibniz formula, we have

$$\begin{aligned} f(\mathbf{y}) &= f(\mathbf{x}) + \int_{\mathbf{y}}^{\mathbf{x}} \nabla f(\mathbf{z}) \mathrm{d}\mathbf{z} \\ &= f(\mathbf{x}) + \int_0^1 \langle \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})), \mathbf{y} - \mathbf{x} \rangle \,\mathrm{d}t \\ &= f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \int_0^1 \langle \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \,\mathrm{d}t \\ &\leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \int_0^1 \|\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x})\|_2 \|\mathbf{y} - \mathbf{x}\|_2 \mathrm{d}t \\ &\leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \int_0^1 L\|\mathbf{y} - \mathbf{x}\|_2 \mathrm{d}t \\ &= f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \\ &\leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\tilde{L}}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \end{aligned}$$

   for any $\tilde{L} \geq L$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Letting $\mathbf{x} = \mathbf{x}_{k-1}$, $\mathbf{y} = p_{\tilde{L}}(\mathbf{x}_{k-1})$ and adding $g(p_{\tilde{L}}(\mathbf{x}_{k-1}))$ to both sides of the above inequality yields the inequality (7).

   Consider the case where $L_0 \leq \eta L$. If $L_k > \eta L$ for some $k \geq 1$, then $\frac{L_k}{\eta} > L$ is a smaller $\tilde{L}$ that satisfies the inequality (7), contradicting Line 3 of Algorithm 1. Therefore, $L_k \leq \eta L$ for every $k \geq 1$. ■

3. Let $\{\mathbf{x}_k\}$ be the sequence generated by Algorithm 1. Show that for any $k \geq 1$ we have

$$F(\mathbf{x}_k) - F(\mathbf{x}^*) \leq \frac{\eta L \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2k}, \forall \mathbf{x}^* \in \operatorname*{\mathbf{argmin}}_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x}).$$

The above result means that the number of iterations of Algorithm 1 required to obtain an ε-optimal solution, i.e., an $\hat{\mathbf{x}}$ such that $F(\hat{\mathbf{x}}) - F(\mathbf{x}^*) \leq \varepsilon$, is at most

$$\left\lceil \frac{\eta L \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2\varepsilon} \right\rceil.$$

Solution:
Invoking the inequality (10) with $\mathbf{x} = \mathbf{x}^* \in \mathbf{argmin}_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x})$, we have

$$\begin{cases} F(\mathbf{x}_k) - F(\mathbf{x}^*) \leq \frac{L_k}{2}\|\mathbf{x}_{k-1} - \mathbf{x}^*\|_2^2 - \frac{L_k}{2}\|\mathbf{x}_k - \mathbf{x}^*\|_2^2, \\ F(\mathbf{x}_{k-1}) - F(\mathbf{x}^*) \leq \frac{L_{k-1}}{2}\|\mathbf{x}_{k-2} - \mathbf{x}^*\|_2^2 - \frac{L_{k-1}}{2}\|\mathbf{x}_{k-1} - \mathbf{x}^*\|_2^2, \\ \quad \vdots \\ F(\mathbf{x}_2) - F(\mathbf{x}^*) \leq \frac{L_2}{2}\|\mathbf{x}_1 - \mathbf{x}^*\|_2^2 - \frac{L_2}{2}\|\mathbf{x}_2 - \mathbf{x}^*\|_2^2, \\ F(\mathbf{x}_1) - F(\mathbf{x}^*) \leq \frac{L_1}{2}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2 - \frac{L_1}{2}\|\mathbf{x}_1 - \mathbf{x}^*\|_2^2. \end{cases}$$

Summing up the above inequalities yields

$$\begin{aligned} \sum_{i=1}^k F(\mathbf{x}_i) - kF(\mathbf{x}^*) &\leq \sum_{i=1}^k \frac{L_i}{2}\left(\|\mathbf{x}_{i-1} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}_i - \mathbf{x}^*\|_2^2\right), \\ &\leq \frac{\eta L}{2}\left(\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2 - \|\mathbf{x}_k - \mathbf{x}^*\|_2^2\right) \leq \frac{\eta L \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2}. \end{aligned}$$

Dividing both sides k, we obtain

$$\frac{1}{k}\sum_{i=1}^k F(\mathbf{x}_i) - F(\mathbf{x}^*) \leq \frac{\eta L \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2k}.$$

As $\{F(\mathbf{x}_i)\}$ is non-increasing, the average $\frac{1}{k}\sum_{i=1}^k F(\mathbf{x}_i) \geq F(\mathbf{x}_k)$, which completes the proof.

To find an ε-optimal solution, we let $\frac{\eta L}{2k}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2 \leq \varepsilon$ and yield

$$k \geq \frac{\eta L \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2\varepsilon} \implies k \geq \left\lceil \frac{\eta L \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2\varepsilon} \right\rceil. \quad ■$$

Exercise 5: Programming Exercise: Naive Bayes Classifier

We provide you with a data set that contains spam and non-spam emails ("`hw5_nb.zip`"). Please use the Naive Bayes Classifier to detect the spam emails. Finish the following exercises by programming. You can use your favorite programming language.

1. Remove all the tokens that contain non-alphabetic characters.
2. Train the Naive Bayes Classifier on the training set according to Algorithm 2.
3. Test the Naive Bayes Classifier on the test set according to Algorithm 3. You may encounter a problem that the likelihood probabilities you calculate approach 0. How do you deal with this problem?
4. Compute the confusion matrix, accuracy, precision, recall, and F-score.
5. Without the Laplace smoothing technique, complete the steps again.

Algorithm 2 Training Naive Bayes Classifier

```
Input: The training set with the labels $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$.
1: $\mathcal{V} \leftarrow$ the set of distinct words and other tokens found in $\mathcal{D}$
2: for each target value $c$ in the labels set $\mathcal{C}$ do
3:     $\mathcal{D}_c \leftarrow$ the training samples whose labels are $c$
4:     $P(c) \leftarrow \frac{|\mathcal{D}_c|}{|\mathcal{D}|}$
5:     $T_c \leftarrow$ a single document by concatenating all training samples in $\mathcal{D}_c$
6:     $n_c \leftarrow |T_c|$
7:     for each word $w_k$ in the vocabulary $\mathcal{V}$ do
8:         $n_{c,k} \leftarrow$ the number of times the word $w_k$ occurs in $T_c$
9:         $P(w_k|c) = \frac{n_{c,k}+1}{n_c+|\mathcal{V}|}$
```

Algorithm 3 Testing Naive Bayes Classifier

```
Input: An email $\mathbf{x}$. Let $x_i$ be the $i^{th}$ token in $\mathbf{x}$ . $\mathcal{I} = \emptyset$.
1: for $i = 1, \dots, |\mathbf{x}|$ do
2:     if $\exists\, w_{k_i} \in \mathcal{V}$ such that $w_{k_i} = x_i$ then
3:         $\mathcal{I} \leftarrow \mathcal{I} \cup i$
4: predict the label of $\mathbf{x}$ by $\hat{y} = \arg\max_{c \in \mathcal{C}} P(c) \prod_{i \in \mathcal{I}} P(w_{k_i}|c)$
```

Solution:
A python implementation is given below.

```python
# Naive Bayes Classifier
import numpy as np
import pandas as pd
import os
import re
```

```python
def to_tokens(text):
    '''
    Extract alphabetic tokens from a text, convert them to lowercase, and return the
↪   counts of tokens in a Series.
    '''
    # Extract the tokens
    tokens = re.findall('[a-z]+', text.lower())
    # Count the tokens
    return pd.Series(tokens).value_counts()


class TextClassifier:
    '''
    Multinomial Naive Bayes for text classification.
    '''

    def __init__(self, smoothing=0):
        '''
        Initialize the classifier.
        '''
        self.smoothing = smoothing

    def fit(self, train_data):
        '''
        Train the classifier on the given training data.
        '''
        # Compute the log of joint probabilities
        self.label_probs = train_data['label'].value_counts() / len(train_data)
        # Initialize the counts for each token
        self.token_counts = {
            label: pd.Series(dtype='int')
            for label in self.label_probs.index
        }
        documents = train_data.groupby('label')['sample'].sum()
        self.token_counts = documents.apply(to_tokens).T.fillna(0)
        # Compute the conditional probabilities for each token
        self.token_probs = (self.token_counts + self.smoothing) / (
            self.token_counts.sum() + self.smoothing * len(self.token_counts))

        return self

    def predict(self, text):
        '''
        Classify the given text using the trained classifier.
        '''
        # Compute the log of posterior probabilities for each label
        log_probs = np.log(self.label_probs)
        for label in self.label_probs.index:
            for token, count in to_tokens(text).items():
                if token in self.token_probs[label]:
                    log_probs[label] += count * np.log(
                        self.token_probs[label][token])

        # Return the label with the highest log-probability
        return log_probs.idxmax()


def load_data(dir):
    '''
    Load the data from the given directory.
    '''
    data = []
    for filename in os.listdir(dir):
        with open(dir + filename, 'r') as f:
            sample = f.read()
        label = 'spam' if 'spmsg' in filename else 'notspam'
        data.append((label, sample))
    return pd.DataFrame(data, columns=['label', 'sample'])


def evaluate(label, pred, p, n):
    '''
    Evaluate the performance of a binary classifier.
    '''
    # Compute the confusion matrix
    TP = np.sum((label == p) & (pred == p))
    FP = np.sum((label == n) & (pred == p))
    TN = np.sum((label == n) & (pred == n))
    FN = np.sum((label == p) & (pred == n))
    confusion_matrix = pd.DataFrame([[TP, FP], [FN, TN]],
                                    index=[p, n],
                                    columns=[p, n])

    # Compute the accuracy, precision, recall, and F1 score
    accuracy = (TP + TN) / (TP + FP + TN + FN)
    precision = TP / (TP + FP)
    recall = TP / (TP + FN)
    f1_score = 2 * TP / (2 * TP + FP + FN)

    # Print the results
    print(f'Confusion matrix:\n{confusion_matrix}\n')
    print(f'Accuracy:  {accuracy:.2f}')
    print(f'Precision: {precision:.2f}')
    print(f'Recall:    {recall:.2f}')
    print(f'F1 score:  {f1_score:.2f}')
    print()

    # Return all the results in a Dataframe
    return pd.DataFrame(
        [[TP, FP, TN, FN, accuracy, precision, recall, f1_score]],
        index=['result'],
        columns=[
            'TP', 'FP', 'TN', 'FN', 'accuracy', 'precision', 'recall',
            'f1_score'
        ])


def main(smoothing):
    # Load the training and test data
    train_data = load_data('hw5_nb/train-mails/')
    test_data = load_data('hw5_nb/test-mails/')
    # Train the classifier
    clf = TextClassifier(smoothing).fit(train_data)
    # Test and evaluate the classifier
    test_data['prediction'] = test_data['sample'].apply(clf.predict)
    return evaluate(test_data['label'], test_data['prediction'], 'spam',
                    'notspam')


# CASE1: Using the Laplace smoothing technique
print('CASE1: Using the Laplace smoothing technique\n')
res1 = main(smoothing=1)
# CASE2: Without using the Laplace smoothing techniqu
print('CASE2: Without using the Laplace smoothing technique\n')
res2 = main(smoothing=0)

# Save the DataFrame
df = pd.concat([res1, res2])
df.index = ['Laplace smoothing', 'No smoothing']
df.to_csv('naive_bayes.csv')
```

The results are shown in Table 1. We notice that the Laplace smoothing technique improves the performance of the classifier significantly by preventing the classifier from assigning zero probability to unseen tokens.

Table 1: Results of Naive Bayes Classifier

	TP	FP	TN	FN	Accuracy	Precision	Recall	F1 score
Laplace smoothing	47	0	242	2	0.9931	1.0	0.9592	0.9792
No smoothing	8	0	242	41	0.8591	1.0	0.1633	0.2807

Answer to Problem 3: We compute the log-likelihoods by summing up the log-probabilities of each token conditional on the label, and then choose the label with the highest log-joint probability. With this technique, we avoid multiplying many small probabilities and thus avoid the underflow problem. ■

Exercise 6: Logistic Regression and Newton's Method

Given the training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$. Let

$$\mathcal{I}^+ = \{i : i \in [n], y_i = 1\},$$
$$\mathcal{I}^- = \{i : i \in [n], y_i = 0\},$$

where $[n] = \{1, 2, \ldots, n\}$. We assume that $\mathcal{I}^+$ and $\mathcal{I}^-$ are not empty.
Then, we can formulate the logistic regression of the form.

$$\min_{\mathbf{w}} \ L(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^n \left(y_i \log\left(\frac{\exp(\langle \mathbf{w}, \bar{\mathbf{x}}_i\rangle)}{1+\exp(\langle \mathbf{w}, \bar{\mathbf{x}}_i\rangle)}\right) + (1-y_i)\log\left(\frac{1}{1+\exp(\langle \mathbf{w}, \bar{\mathbf{x}}_i\rangle)}\right)\right), \tag{11}$$

where $\mathbf{w} \in \mathbb{R}^{d+1}$ is the model parameter to be estimated and $\bar{\mathbf{x}}_i^\top = (1, \mathbf{x}_i^\top)$.

1. (a) Suppose that the training data is strictly linearly separable, that is, there exists $\hat{\mathbf{w}} \in \mathbb{R}^{d+1}$ such that

$$\langle \hat{\mathbf{w}}, \bar{\mathbf{x}}_i\rangle > 0, \, \forall\, i \in \mathcal{I}^+,$$
$$\langle \hat{\mathbf{w}}, \bar{\mathbf{x}}_i\rangle < 0, \, \forall\, i \in \mathcal{I}^-.$$

Show that problem (11) has no solution.

(b) Suppose that the training data is NOT linearly separable, that is, for all $\mathbf{w} \in \mathbb{R}^{d+1}$, there exists $i \in [n]$ such that

$$\langle \mathbf{w}, \bar{\mathbf{x}}_i\rangle < 0, \text{ if } i \in \mathcal{I}^+,$$

or

$$\langle \mathbf{w}, \bar{\mathbf{x}}_i\rangle > 0, \text{ if } i \in \mathcal{I}^-.$$

Show that problem (11) always admits a solution.

Solution:

(a) We can write the objective function $L(\hat{\mathbf{w}})$ as

$$L(\mathbf{w}) = \frac{1}{n}\left(\sum_{i\in\mathcal{I}^+} \log\left(1+\exp(-\langle \mathbf{w}, \bar{\mathbf{x}}_i\rangle)\right) + \sum_{i\in\mathcal{I}^-} \log\left(1+\exp(\langle \mathbf{w}, \bar{\mathbf{x}}_i\rangle)\right)\right) > 0.$$

Let $\hat{\mathbf{w}}$ be a solution such that $\langle \hat{\mathbf{w}}, \bar{\mathbf{x}}_i\rangle > 0$ if $y_i = 1$, and $\langle \hat{\mathbf{w}}, \bar{\mathbf{x}}_i\rangle < 0$ if $y_i = 0$. Then, for any $\alpha > 1$, we have

$$L(\alpha\hat{\mathbf{w}}) = \frac{1}{n}\left(\sum_{i\in\mathcal{I}^+} \log\left(1+\exp(-\alpha\langle \hat{\mathbf{w}}, \bar{\mathbf{x}}_i\rangle)\right) + \sum_{i\in\mathcal{I}^-} \log\left(1+\exp(\alpha\langle \hat{\mathbf{w}}, \bar{\mathbf{x}}_i\rangle)\right)\right) \xrightarrow{\alpha\to+\infty} 0.$$

Hence $\inf_{\mathbf{w}} L(\mathbf{w}) = 0$ cannot be attained, implying that the problem is not solvable.

(b) Given any $\mathbf{w}$ and $\alpha \to +\infty$, there exists $i \in \mathcal{I}^+$ such that $\log\left(1+\exp(-\alpha\langle \hat{\mathbf{w}}, \bar{\mathbf{x}}_i\rangle)\right) \to +\infty$, or $i \in \mathcal{I}^-$ such that $\log\left(1+\exp(\alpha\langle \hat{\mathbf{w}}, \bar{\mathbf{x}}_i\rangle)\right) \to +\infty$. Therefore,

$$\lim_{\alpha \to +\infty} L(\alpha\hat{\mathbf{w}}) = +\infty,$$

implying that the objective function is coercive[1], and hence it attains its minimum over **dom** L, i.e. the problem is solvable. ■

2. Suppose that $\bar{\mathbf{X}} = (\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_n)^\top \in \mathbb{R}^{n\times(d+1)}$ and $\operatorname{rank}\left(\bar{\mathbf{X}}\right) = d+1$. Show that $L(\mathbf{w})$ is strictly convex, i.e., for all $\mathbf{w}_1 \neq \mathbf{w}_2$,

$$L(t\mathbf{w}_1 + (1-t)\mathbf{w}_2) < tL(\mathbf{w}_1) + (1-t)L(\mathbf{w}_2), \forall\, t \in (0,1).$$

Solution:
The gradient of $L(\mathbf{w})$ is

$$\nabla L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+\exp(-\langle \mathbf{w}, \bar{\mathbf{x}}_i\rangle)} - y_i\right)\bar{\mathbf{x}}_i = \frac{1}{n}\bar{\mathbf{X}}^\top\left(\sigma(\bar{\mathbf{X}}\mathbf{w}) - \mathbf{y}\right),$$

where we define σ as the sigmoid function

$$\sigma(\mathbf{z}) = \operatorname{sigmoid}(\mathbf{z}) = \left(\frac{1}{1+\exp(-z_1)}, \frac{1}{1+\exp(-z_2)}, \ldots, \frac{1}{1+\exp(-z_n)}\right).$$

Note that

$$\begin{aligned}\nabla\sigma(\mathbf{z}) &= \sigma(\mathbf{z})^\top\mathbf{I}(1-\sigma(\mathbf{z}))\\ &= \begin{pmatrix} \sigma(z_1)(1-\sigma(z_1)) & 0 & \ldots & 0\\ 0 & \sigma(z_2)(1-\sigma(z_2)) & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \sigma(z_n)(1-\sigma(z_n))\end{pmatrix} \succ 0.\end{aligned}$$

So the Hessian of $L(\mathbf{w})$ satisfies

$$\nabla^2 L(\mathbf{w}) = \frac{1}{n}\bar{\mathbf{X}}^\top\mathbf{\Sigma}_{\mathbf{w}}\bar{\mathbf{X}} \succeq 0,$$

where we define $\mathbf{\Sigma}_{\mathbf{w}} = \nabla\sigma(\mathbf{z})\big|_{\mathbf{z}=\bar{\mathbf{X}}\mathbf{w}}$ for notational convenience. This implies that $L(\mathbf{w})$ is convex. If $\operatorname{rank}\left(\bar{\mathbf{X}}\right) = d+1$, then $\nabla^2 L(\mathbf{w}) \succ 0$, implying that $L(\mathbf{w})$ is strictly convex. ■

3. In real applications, a widely-used method to learn the parameters' values of logistic regression is to solve the optimization problem (11) with a regularization term, e.g.,

$$\min_{\mathbf{w}} F(\mathbf{w}) = L(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2,\ \lambda > 0.$$

The Newton's method is an iterative method for optimization problems. We use the Newton's method to fit the regularized logistic regression by the following algorithm.

Algorithm 4 Newton's Method for Logistic Regression

Input: The twice-differentiable objective function $F(\mathbf{w})$, the initial point $\mathbf{w_0}$, the degree of precision ϵ.

Output: $\hat{\mathbf{w}}$,the first point satisfying $\|\hat{\mathbf{g}}\|_2 < \epsilon$.

```
1: Calculate the gradient $\mathbf{g}(\mathbf{w}) = \nabla F(\mathbf{w})$ and the Hessian matrix $\mathbb{H}(\mathbf{w})$ of the
   input $F(\mathbf{w})$.
2: while $\|\mathbf{g}_k(\mathbf{w}_k)\|_2 \geq \epsilon$ do
3:     Let $\mathbb{H}(\mathbf{w}_k) = \mathbb{H}_k$ and $\mathbf{g}(\mathbf{w}_k) = \mathbf{g}_k$ to simplify notations.
       Calculate the Hessian matrix $\mathbb{H}_k$, and the let $\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbb{H}_k^{-1}\mathbf{g}_k$.
4:     $k \leftarrow k + 1$.
5:     Calculate the gradient $\mathbf{g}_{k+1}$.
```

(a) Please calculate the gradient $\mathbf{g}(\mathbf{w})$ and the Hessian matrix $\mathbb{H}(\mathbf{w})$ of the regularized logistic regression.

(b) Please show that the Hessian matrix $\mathbb{H}(\mathbf{w})$ is invertible.

(c) (Bonus) Please show the local convergence of Newton's method in logistic regression, i.e.,

$$\frac{\|\mathbf{w}_{k+1} - \mathbf{w}^*\|}{\|\mathbf{w}_k - \mathbf{w}^*\|^2} < B,$$

for some $B \in \mathbb{R}$, if the initial point is closed enough to $\mathbf{w}^* = \arg\min_{\mathbf{w}} F(\mathbf{w})$.

Solution:

(a) Using the results in Problem 2, we have

$$\mathbf{g}(\mathbf{w}) = \nabla L(\mathbf{w}) + \lambda\mathbf{w} = \frac{1}{n}\bar{\mathbf{X}}^\top(\sigma(\bar{\mathbf{X}}\mathbf{w}) - \mathbf{y}) + \lambda\mathbf{w},$$
$$\mathbb{H}(\mathbf{w}) = \nabla^2 L(\mathbf{w}) + \lambda\mathbf{I} = \frac{1}{n}\bar{\mathbf{X}}^\top\sigma(\mathbf{z})^\top\mathbf{I}(1 - \sigma(\mathbf{z}))\bar{\mathbf{X}} + \lambda\mathbf{I}.$$

(b) We have shown in Problem 2 that $\nabla^2 L(\mathbf{w}) \succeq 0$, and thus $\mathbb{H}(\mathbf{w}) \succ 0$ is invertible.

(c) To prove the local convergence, we assume the following statements hold.

i. $\forall\mathbf{w},\ \|\mathbb{H}(\mathbf{w})^{-1}\| \leq \frac{1}{\mu}$ for some $\mu > 0$.

ii. $\forall\mathbf{w}_1, \mathbf{w}_2,\ \|\mathbb{H}(\mathbf{w}_1) - \mathbb{H}(\mathbf{w}_2)\| \leq L_{\mathbb{H}}\|\mathbf{w}_1 - \mathbf{w}_2\|$ for some $L_{\mathbb{H}} > 0$.

The Newton step implies that

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbb{H}_k^{-1}\mathbf{g}_k = \mathbf{w}_k - \mathbb{H}_k^{-1}\int_{\mathbf{w}^*}^{\mathbf{w}_k}\mathbb{H}(\mathbf{w})\mathrm{d}\mathbf{w} = \mathbf{w}^* + \mathbb{H}_k^{-1}\int_{\mathbf{w}^*}^{\mathbf{w}_k}(\mathbb{H}_k - \mathbb{H}(\mathbf{w}))\mathrm{d}\mathbf{w}.$$

Subtracting $\mathbf{w}^*$ from both sides and taking the norm, we have

$$\begin{aligned}\|\mathbf{w}_{k+1} - \mathbf{w}^*\| &= \left\|\mathbb{H}_k^{-1}\int_{\mathbf{w}^*}^{\mathbf{w}_k}(\mathbb{H}_k - \mathbb{H}(\mathbf{w}))\mathrm{d}\mathbf{w}\right\| \leq \|\mathbb{H}_k^{-1}\|\int_{\mathbf{w}^*}^{\mathbf{w}_k}\|\mathbb{H}_k - \mathbb{H}(\mathbf{w})\|\|\mathrm{d}\mathbf{w}\| \\ &\leq \frac{1}{\mu}\int_{\mathbf{w}^*}^{\mathbf{w}_k}L_{\mathbb{H}}\|\mathbf{w}_k - \mathbf{w}\|\|\mathrm{d}\mathbf{w}\| = \frac{L_{\mathbb{H}}}{2\mu}\|\mathbf{w}_k - \mathbf{w}^*\|^2,\end{aligned}$$

from which the local convergence follows.

We now show that the statements i and ii hold. In regularized logistic regression, we have $\|\mathbb{H}(\mathbf{w})\| \geq \lambda$, and thus $\|\mathbb{H}(\mathbf{w})^{-1}\| \leq \frac{1}{\lambda}$. To establish the second statement, we have

$$\begin{aligned}\|\mathbb{H}(\mathbf{w}_1) - \mathbb{H}(\mathbf{w}_2)\|_2 &= \frac{1}{n}\left\|\bar{\mathbf{X}}^\top(\mathbf{\Sigma}_{\mathbf{w}_1} - \mathbf{\Sigma}_{\mathbf{w}_2})\bar{\mathbf{X}}\right\| \leq \frac{1}{n}\|\bar{\mathbf{X}}\|^2\|\mathbf{\Sigma}_{\mathbf{w}_1} - \mathbf{\Sigma}_{\mathbf{w}_2}\| \\ &\leq \frac{1}{n}\|\bar{\mathbf{X}}\|^2\left\|\mathbf{1} + \sigma(\bar{\mathbf{X}}\mathbf{w}_1) + \sigma(\bar{\mathbf{X}}\mathbf{w}_2)\right\|\left\|\sigma(\bar{\mathbf{X}}\mathbf{w}_1) - \sigma(\bar{\mathbf{X}}\mathbf{w}_2)\right\| \\ &\leq \frac{1}{4n}\|\bar{\mathbf{X}}\|^3\left\|\mathbf{1} + \sigma(\bar{\mathbf{X}}\mathbf{w}_1) + \sigma(\bar{\mathbf{X}}\mathbf{w}_2)\right\|\|\mathbf{w}_1 - \mathbf{w}_2\|,\end{aligned}$$

where we use the Lipschitz continuity of σ (see Exercise 8.1). Since $\mathbf{w}$ is bounded in a neighborhood of $\mathbf{w}^*$, we conclude that $\|\mathbb{H}(\mathbf{w}_1) - \mathbb{H}(\mathbf{w}_2)\|_2 \leq L_{\mathbb{H}}\|\mathbf{w}_1 - \mathbf{w}_2\|$ for some $L_{\mathbb{H}} > 0$. The proof is complete. ■

Exercise 7: Convergence of Stochastic Gradient Descent for Convex Function

Consider an optimization problem

$$\min_{\mathbf{w}} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w}), \tag{12}$$

where the objective function F is continuously differentiable and strongly convex with convexity parameter $\mu > 0$. Suppose that the gradient of F, i.e., ∇F, is Lipschitz continuous with Lipschitz constant L, and F can attain its minimum F^* at $\mathbf{w}^*$. We use the stochastic gradient descent(SGD) algorithm introduced in Lecture 12 to solve the problem (12). Let the solution sequence generated by SGD be $\{\mathbf{w}_k\}$.

1. Please show that $\forall \mathbf{w} \in \mathbf{dom}\ F$, the following inequality

$$F(\mathbf{w}) - F^* \le \frac{1}{2\mu}\|\nabla F(\mathbf{w})\|^2 \tag{13}$$

holds, and interpret the role of strong convexity based on this.

Solution:
By strong convexity, we have

$$\begin{aligned} F(\mathbf{w}^*) &\ge F(\mathbf{w}) + \langle \nabla F(\mathbf{w}), \mathbf{w}^* - \mathbf{w}\rangle + \frac{\mu}{2}\|\mathbf{w}^* - \mathbf{w}\|^2 \\ &= F(\mathbf{w}) + \frac{1}{2\mu}\left\|\nabla F(\mathbf{w}) + \mu\left(\mathbf{w}^* - \mathbf{w}\right)\right\|^2 - \frac{1}{2\mu}\|\nabla F(\mathbf{w})\|^2 \\ &\ge F(\mathbf{w}) - \frac{1}{2\mu}\|\nabla F(\mathbf{w})\|^2, \end{aligned}$$

i.e. $F(\mathbf{w}) - F^* \le \frac{1}{2\mu}\|\nabla F(\mathbf{w})\|^2$. The strong convexity parameter μ controls the upper bound of the optimality gap. ■

2. Recall that with a fixed stepsize $\alpha \in [0, \frac{1}{LM_G}]$ where M_G (as well as the following M) is a parameter regarding the upper bound of the variance of stochastic gradient in SGD, the sequence $\{\mathbb{E}[F(\mathbf{w}_k)]\}$ generated by SGD converges to a neighborhood of F^* with a linear rate, i.e,

$$\mathbb{E}_{\xi_0:\xi_{k-1}}[F(\mathbf{w}_k) - F^*] \le \frac{LM}{2\mu}\alpha + (1-\mu\alpha)^k\left(F(\mathbf{w}_0) - F^* - \frac{LM}{2\mu}\alpha\right) \xrightarrow{\text{linear}} \frac{LM}{2\mu}\alpha.$$

This means that the expected optimality gap, i.e., $\mathbb{E}_{\xi_0:\xi_{k-1}}[F(\mathbf{w}_k) - F^*]$, fails to converge to zero. In order to alleviate this problem, we consider a strategy of diminishing stepsize α_k. Suppose that the SGD method is run with a stepsize sequence $\{\alpha_k\}$ such that, for all $k \in \mathbb{N}$, $\alpha_k = \frac{\beta}{\gamma+k}$ for some $\beta > \frac{1}{\mu}$ and $\gamma > 0$ satisfying $\alpha_0 \le \frac{1}{LM_G}$. Please show that $\forall k \in \mathbb{N}$, we have

$$\mathbb{E}_{\xi_0:\xi_{k-1}}[F(\mathbf{w}_k) - F^*] \le \frac{\tau}{\gamma+k},$$

where $\tau = \max\{\frac{\beta^2 LM}{2(\beta\mu-1)}, \gamma(F(\mathbf{w}_0) - F^*)\}$.

Solution:
The following result has been shown in Lecture 11:

$$\mathbb{E}_{\xi_k}[F(\mathbf{w}_{k+1}) - F(\mathbf{w}_k)] \le -\alpha_k(1 - \frac{L}{2}M_G\alpha_k)\|\nabla F(\mathbf{w}_k)\|^2 + \frac{L}{2}M\alpha_k^2. \tag{14}$$

As $0 < \alpha_k < \alpha_0 \le \frac{1}{LM_G}$, we have

$$\mathbb{E}_{\xi_k}[F(\mathbf{w}_{k+1}) - F(\mathbf{w}_k)] \le -\frac{1}{2}\alpha_k\|\nabla F(\mathbf{w}_k)\|^2 + \frac{L}{2}M\alpha_k^2.$$

Using the result in Problem 1, we have

$$\mathbb{E}_{\xi_k}[F(\mathbf{w}_{k+1}) - F(\mathbf{w}_k)] \le \alpha_k\mu(F(\mathbf{w}^*) - F(\mathbf{w}_k)) + \frac{L}{2}M\alpha_k^2.$$

Subtracting F^* from both sides, taking expectations and rearranging, we obtain

$$\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_{k+1}) - F(\mathbf{w}^*)] \le (1 - \alpha_k\mu)\mathbb{E}_{\xi_0:\xi_{k-1}}[F(\mathbf{w}_k) - F(\mathbf{w}^*)] + \frac{L}{2}M\alpha_k^2.$$

To prove by induction, we assume that $\mathbb{E}_{\xi_0:\xi_{k-1}}[F(\mathbf{w}_k) - F(\mathbf{w}^*)] \le \frac{\tau}{\gamma+k}$. Then,

$$\begin{aligned}
\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_{k+1}) - F(\mathbf{w}^*)] &\le (1 - \alpha_k\mu)\frac{\tau}{\gamma+k} + \frac{L}{2}M\alpha_k^2 \\
&= \left(1 - \frac{\beta\mu}{\gamma+k}\right)\frac{\tau}{\gamma+k} + \frac{L}{2}M\left(\frac{\beta}{\gamma+k}\right)^2 \\
&= \left(1 - \frac{1}{(\gamma+k)^2}\right)\tau + \frac{1}{(\gamma+k)^2}\left((1-\beta\mu)\tau + \frac{\beta^2 LM}{2}\right) \\
&\le \left(1 - \frac{1}{(\gamma+k)^2}\right)\tau \le \frac{\tau}{\gamma+k+1}.
\end{aligned}$$

Moreover, by definition of τ, $F(\mathbf{w}_0) - F^* \le \frac{\tau}{\gamma}$, which completes the proof. ■

3. In practice, for the same problem, SGD enjoys less time cost but more iteration steps than gradient descent methods and may suffer from non-convergence. As a trade-off between SGD and gradient descent approaches, consider using mini-batch samples to estimate the full gradient. Taking k^{th} iteration as an example, instead of picking a single sample, we randomly select a subset $\mathcal{S}_k$ of the sample indices to compute the update direction

$$\mathbf{g}_k(\xi_k) = \frac{1}{|\mathcal{S}_k|}\sum_{i\in\mathcal{S}_k}\nabla f_i(\mathbf{w}_k)$$

where ξ_k is the selected samples. For simplicity, suppose that the mini-batches in all iterations are of constant size, i.e., $|\mathcal{S}_k| = n_m$, and the stepsize α is fixed. Please show that for mini-batch SGD, there holds

$$\mathbb{E}_{\xi_0:\xi_{k-1}}[F(\mathbf{w}_k) - F^*] \le \frac{LM}{2\mu n_m}\alpha + (1-\mu\alpha)^k(F(\mathbf{w}_0) - F^* - \frac{LM}{2\mu n_m}\alpha) \xrightarrow{\text{linear}} \frac{LM}{2\mu n_m}\alpha.$$

Moreover, point out the advantage of mini-batch SGD compared to SGD in terms of the number of the iteration step.

Solution:
Note that $\mathbb{E}_{\xi_k}[\mathbf{g}_k(\xi_k)] = \nabla F(\mathbf{w}_k)$ still holds. Using the result in Lecture 11, we have

$$\mathbb{E}_{\xi_k}[F(\mathbf{w})_{k+1} - F(\mathbf{w}_k)] \leq -\alpha(1 - \frac{L}{2}\alpha)\|\nabla F(\mathbf{w}_k)\|^2 + \frac{L}{2}\alpha^2 \mathbb{V}_{\xi_k}[\mathbf{g}_k(\xi_k)],$$

where the variance satisfies

$$\begin{aligned}\mathbb{V}_{\xi_k}[\mathbf{g}_k(\xi_k)] &= \mathbb{V}_{\xi_k}\left[\frac{1}{n_m}\sum_{i\in\mathcal{S}_k}\nabla f_i(\mathbf{w}_k)\right] = \frac{1}{n_m^2}\sum_{i\in\mathcal{S}_k}\mathbb{V}_{\xi_k}\left[\nabla f_i(\mathbf{w}_k)\right] \\ &\leq \frac{1}{n_m^2}\sum_{i\in\mathcal{S}_k}\left(M + M_V\|\nabla F(\mathbf{w}_k)\|^2\right) = \frac{M}{n_m} + \frac{M_V}{n_m}\|\nabla F(\mathbf{w}_k)\|^2.\end{aligned}$$

Thus, the equation (14) becomes

$$\mathbb{E}_{\xi_k}[F(\mathbf{w}_{k+1}) - F(\mathbf{w}_k)] \leq -\alpha(1 - \frac{LM_G}{2n_m}\alpha)\|\nabla F(\mathbf{w}_k)\|^2 + \frac{LM}{2n_m}\alpha^2.$$

Letting $0 < \alpha < \frac{n_m}{LM_G}$, we have

$$\mathbb{E}_{\xi_k}[F(\mathbf{w}_{k+1}) - F(\mathbf{w}_k)] \leq -\frac{\alpha}{2}\|\nabla F(\mathbf{w}_k)\|^2 + \frac{LM}{2n_m}\alpha^2.$$

Using the result in Problem 1, we have

$$\mathbb{E}_{\xi_k}[F(\mathbf{w}_{k+1}) - F(\mathbf{w}_k)] \leq \mu\alpha(F(\mathbf{w}_k) - F^*) + \frac{LM}{2n_m}\alpha^2.$$

Subtracting F^* from both sides, taking expectations and rearranging, we obtain

$$\mathbb{E}_{\xi_k}[F(\mathbf{w}_{k+1}) - F^*] \leq (1 - \mu\alpha)\mathbb{E}_{\xi_{k-1}}[F(\mathbf{w}_k) - F^*] + \frac{LM}{2n_m}\alpha^2,$$

which is equivalent to

$$\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_{k+1}) - F^*] - \frac{LM}{2\mu n_m}\alpha \leq (1 - \mu\alpha)\left(\mathbb{E}_{\xi_0:\xi_{k-1}}[F(\mathbf{w}_k) - F^*] - \frac{LM}{2\mu n_m}\alpha\right).$$

The result follows immediately, i.e.

$$\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_{k+1}) - F^*] \leq \frac{LM}{2\mu n_m}\alpha + (1 - \mu\alpha)^{k+1}(F(\mathbf{w}_0) - F^* - \frac{LM}{2\mu n_m}\alpha) \xrightarrow{\text{linear}} \frac{LM}{2\mu n_m}\alpha.$$

To obtain the same optimality gap using SGD, we need to choose a stepsize $\frac{\alpha}{n_m}$ and then

$$\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_{k+1}) - F^*] \leq \frac{LM}{2\mu n_m}\alpha + (1 - \frac{\mu\alpha}{n_m})^{k+1}(F(\mathbf{w}_0) - F^* - \frac{LM}{2\mu n_m}\alpha) \xrightarrow{\text{linear}} \frac{LM}{2\mu n_m}\alpha.$$

As $1 - \mu\alpha < 1 - \frac{\mu\alpha}{n_m}$, we see that the mini-batch SGD needs fewer iteration steps. ■

4. Notice that in real applications, F is not always strongly convex. Let F be convex and continuously differentiable, and the second moment of stochastic gradient $\mathbf{g}$ be bounded, i.e.,

$$\mathbb{E}_\xi[\|\mathbf{g}(\xi)\|_2^2] \leq G^2.$$

We denote $\{\mathbf{w}_k\}$ as a sequence generated by SGD algorithm with a fixed stepsize α. Besides, define $\tilde{\mathbf{w}}_K = \frac{1}{K+1}\sum_{k=0}^K \mathbf{w}_k$ and $F^* = F(\mathbf{w}^*)$.

(a) If X is a random variable and $h : \mathbb{R} \to \mathbb{R}$ is a convex function, please show that

$$h(\mathbb{E}[X]) \leq \mathbb{E}[h(X)].$$

(b) Suppose that the stochastic gradient at k^{th} iteration is $\mathbf{g}_k$. Please show that

$$\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_k) - F^*] \leq \mathbb{E}_{\xi_0:\xi_k}[\langle \mathbf{g}_k, \mathbf{w}_k - \mathbf{w}^*\rangle].$$

(c) Please show that

$$\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_k) - F^*] \leq \frac{1}{2\alpha}\mathbb{E}_{\xi_0:\xi_k}[\|\mathbf{w}_k - \mathbf{w}^*\|_2^2 - \|\mathbf{w}_{k+1} - \mathbf{w}^*\|_2^2 + \alpha^2\|\mathbf{g}_k\|_2^2].$$

(d) Please show that

$$\mathbb{E}_{\xi_0:\xi_K}[F(\tilde{\mathbf{w}}_K) - F^*] \leq \frac{\|\mathbf{w}_0 - \mathbf{w}^*\|_2^2 + \alpha^2 G^2(K+1)}{2\alpha(K+1)} \to \frac{\alpha G^2}{2}.$$

Solution:

(a) Denote the cumulative distribution function of X by $F_X(x)$. Since $\int \mathrm{d}F_X(x) = 1$, by convexity of h, we have

$$h(\mathbb{E}[X]) = h\left(\int x \ \mathrm{d}F_X(x)\right) \leq \int h(x) \ \mathrm{d}F_X(x) = \mathbb{E}[h(X)].$$

(b) The full gradient $\nabla F(\mathbf{w})$ satisfies

$$F^* \geq F(\mathbf{w}_k) + \langle \nabla F(\mathbf{w}_k), \mathbf{w}^* - \mathbf{w}_k\rangle.$$

Rearranging the terms and taking expectation on both sides, we obtain

$$\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_k) - F^*] \leq \mathbb{E}_{\xi_0:\xi_k}[\langle \nabla F(\mathbf{w}_k), \mathbf{w}_k - \mathbf{w}^*\rangle] = \mathbb{E}_{\xi_0:\xi_k}[\langle \mathbb{E}_{\xi_k}[\mathbf{g}_k], \mathbf{w}_k - \mathbf{w}^*\rangle].$$

It is easy to see that $\mathbb{E}_{\xi_0:\xi_k}[\langle \cdot, \mathbf{w}_k - \mathbf{w}^*\rangle]$ is convex as a linear combinition of affine functions. Therefore,

$$\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_k) - F^*] \leq \mathbb{E}_{\xi_k}\left\{\mathbb{E}_{\xi_0:\xi_k}[\langle \mathbf{g}_k, \mathbf{w}_k - \mathbf{w}^*\rangle]\right\} = \mathbb{E}_{\xi_0:\xi_k}[\langle \mathbf{g}_k, \mathbf{w}_k - \mathbf{w}^*\rangle].$$

(c) Note that

$$2\langle \mathbf{w}_k - \mathbf{w}_{k+1}, \mathbf{w}_k - \mathbf{w}^*\rangle = \|\mathbf{w}_k - \mathbf{w}^*\|_2^2 - \|\mathbf{w}_{k+1} - \mathbf{w}^*\|_2^2 + \|\mathbf{w}_{k+1} - \mathbf{w}_k\|_2^2,$$

where $\mathbf{w}_{k+1} - \mathbf{w}_k = -\alpha \mathbf{g}_k$. Therefore,

$$2\alpha \langle \mathbf{g}_k, \mathbf{w}_k - \mathbf{w}^* \rangle = \|\mathbf{w}_k - \mathbf{w}^*\|_2^2 - \|\mathbf{w}_{k+1} - \mathbf{w}^*\|_2^2 + \alpha^2 \|\mathbf{g}_k\|_2^2,$$

Plugging this into the inequality in (b), we have

$$\begin{aligned}\mathbb{E}_{\xi_0:\xi_k}[F(\mathbf{w}_k) - F^*] &\le \frac{1}{\alpha}\mathbb{E}_{\xi_0:\xi_k}[\langle \mathbf{w}_k - \mathbf{w}_{k+1}, \mathbf{w}_k - \mathbf{w}^* \rangle] \\ &= \frac{1}{2\alpha}\mathbb{E}_{\xi_0:\xi_k}[\|\mathbf{w}_k - \mathbf{w}^*\|_2^2 - \|\mathbf{w}_{k+1} - \mathbf{w}^*\|_2^2 + \alpha^2\|\mathbf{g}_k\|_2^2].\end{aligned}$$

(d) Summing up the following inequalities,

$$\begin{cases}\mathbb{E}_{\xi_0:\xi_K}[F(\mathbf{w}_K) - F^*] \le \frac{1}{2\alpha}\mathbb{E}_{\xi_0:\xi_K}[\|\mathbf{w}_K - \mathbf{w}^*\|_2^2 - \|\mathbf{w}_{K+1} - \mathbf{w}^*\|_2^2 + \alpha^2\|\mathbf{g}_K\|_2^2], \\ \quad \vdots \\ \mathbb{E}_{\xi_0:\xi_1}[F(\mathbf{w}_1) - F^*] \le \frac{1}{2\alpha}\mathbb{E}_{\xi_0:\xi_1}[\|\mathbf{w}_1 - \mathbf{w}^*\|_2^2 - \|\mathbf{w}_2 - \mathbf{w}^*\|_2^2 + \alpha^2\|\mathbf{g}_1\|_2^2], \\ \mathbb{E}_{\xi_0}[F(\mathbf{w}_0) - F^*] \le \frac{1}{2\alpha}\mathbb{E}_{\xi_0}[\|\mathbf{w}_0 - \mathbf{w}^*\|_2^2 - \|\mathbf{w}_1 - \mathbf{w}^*\|_2^2 + \alpha^2\|\mathbf{g}_0\|_2^2],\end{cases}$$

we have

$$\begin{aligned}&\mathbb{E}_{\xi_0:\xi_K}\left[\sum_k^{K+1} F(\mathbf{w}_k) - (K+1)F^*\right] \\ \le&\frac{1}{2\alpha}\left\{\mathbb{E}_{\xi_0:\xi_K}\left[\|\mathbf{w}_0 - \mathbf{w}^*\|_2^2 - \|\mathbf{w}_{K+1} - \mathbf{w}^*\|_2^2\right] + \mathbb{E}_{\xi_0:\xi_K}\left[\alpha^2\sum_k^{K+1}\|\mathbf{g}_k\|_2^2\right]\right\} \\ \le&\frac{1}{2\alpha}\left\{\|\mathbf{w}_0 - \mathbf{w}^*\|_2^2 + \alpha^2(K+1)G^2\right\}.\end{aligned}$$

Dividing both sides by $K+1$, we obtain

$$\mathbb{E}_{\xi_0:\xi_K}[F(\tilde{\mathbf{w}}_K) - F^*] \le \frac{\|\mathbf{w}_0 - \mathbf{w}^*\|_2^2 + \alpha^2 G^2(K+1)}{2\alpha(K+1)} \to \frac{\alpha G^2}{2},$$

as desired. ■

Exercise 8: Programming Exercise: Logistic Regression (Optional)

We provide you with a dataset of handwritten digits[1] that contains a training set of 60000 examples and a test set of 2022 examples ("hw5_lr.mat"). Each image in this dataset has 28×28 pixels and the associated label is the handwritten digit—that is, an integer from the set $\{0, 1, \cdots, 9\}$—in the image. In this exercise, you need to build a logistic regression classifier to *predict if a given image has the handwritten digit* 6 *in it or not.* You can use your favorite programming language to finish this exercise.

1. Normalize the data matrix and please find a Lipschitz constant of $\nabla L(\mathbf{w})$, where $L(\mathbf{w})$ is the objective function of the logistic regression after normalizing and $\mathbf{w}$ is the model parameter to be estimated.

   Solution:
   According to Exercise 6.2, we have

   $$\left\|\nabla^2 L(\mathbf{w})\right\|_2 = \frac{1}{n}\left\|\bar{\mathbf{X}}^\top \mathbf{\Sigma}_{\mathbf{w}} \bar{\mathbf{X}}\right\|_2 \leq \frac{1}{n}\|\mathbf{\Sigma}_{\mathbf{w}}\|_2 \left\|\bar{\mathbf{X}}\right\|_2^2,$$

   where $\bar{\mathbf{X}} \in \mathbb{R}^{n\times(d+1)}$ is the normalized data matrix with $n = 60000$ and $d = 28 \times 28$, respectively. As $\mathbf{\Sigma}_{\mathbf{w}}$ is a diagonal matrix with the i^{th} diagonal element $p_i(1-p_i)$ for some $0 \leq p_i \leq 1$, we have

   $$\|\mathbf{\Sigma}_{\mathbf{w}}\|_2 = \max_{i=1,\cdots,n} p_i(1-p_i) \leq \max_{0\leq p\leq 1} p(1-p) = \frac{1}{4}.$$

   We derive the upper bounds of $\left\|\bar{\mathbf{X}}\right\|_2^2$ as

   $$\left\|\bar{\mathbf{X}}\right\|_2^2 \leq \left\|\bar{\mathbf{X}}\right\|_1 \left\|\bar{\mathbf{X}}\right\|_\infty = n \max_{i=1,\cdots,n} \|\bar{\mathbf{x}}_i\|_1 \leq n(d+1).$$

   Hence $\left\|\nabla^2 L(\mathbf{w})\right\|_2 \leq \frac{\|\bar{\mathbf{X}}\|_2^2}{4n} \leq \frac{d+1}{4} = 196.25$, i.e. a Lipschitz constant of $\nabla L(\mathbf{w})$ is 196.25. Moreover, by computing $\frac{\|\bar{\mathbf{X}}\|_2^2}{4n}$, we can obtain a smaller Lipschitz constant of $\nabla L(\mathbf{w})$ as 9.753154. In the following, we will use this smaller Lipschitz constant to train the logistic regression classifier. ■

2. (a) Use the gradient descent algorithm (GD), which is a special case of ISTA introduced in Lecture 09, and SGD to train the logistic regression classifier on the training set, respectively. Evaluate the classification accuracy on the training set after each iteration. Stop the iteration when Accuracy $\geq 90\%$ or total steps are more than 5000. Please plot the accuracy of these two classifiers (the one trained by GD and the other trained by SGD) versus the iteration step on one graph.

   (b) Compare the total iteration counts and the total time cost of the two methods (GD and SGD), respectively. Please report your result.

   (c) Compare the confusion matrix, precision, recall and F1 score of the two classifiers (the one trained by GD and the other trained by SGD). Please report your result.

   (d) Use GD and SGD to train the logistic regression classifier with a 2-norm regularization term. Note that other experimental setup details is in line with 2(a). Please plot the accuracy of these two classifiers (the one trained by GD and the other trained by SGD) versus the iteration step on one graph and compare the confusion matrix, precision, recall and F1 score of the two classifiers.

Solution:
A python implementation is shown at the end of this exercise. The results using the original imbalanced training set are shown in Table 2. The plot of the accuracy and loss of the two classifiers versus the iteration step is shown in Figure 1.

Comments:

- The stochastic gradient descent is much faster than the full gradient descent when computing gradient, and needs less iterations to reach the same accuracy by introducing randomness.
- We only measure the time cost for computing the gradient and updating the model parameter. However, as we need to plot the accuracy score, we always compute the full probability vector in each iteration, which cause the stochastic gradient descent to be as slow as the full gradient descent in practice.
- The speed of convergence tends to be faster when the regularization term is set to be larger. This is due to the imbalanced nature of the training set. When the weights are all equal to zero, the model will predict the majority class for all the images, which will result in a high accuracy near 90%. This also explains why the recall is getting lower when the regularization term is set to be larger. ■

3. (a) The training set is imbalanced as the majority class has roughly nine times more images than the minority class. Imbalanced data can hurt the performance of the classifiers badly. Thus, please undersample the majority class such that the numbers of images in the two classes are roughly the same.

   (b) Use GD and SGD to train the logistic regression classifier on the new training set after undersampling. Stop the iteration when Accuracy $\geq 90\%$ or total steps are more than 5000.

   (c) Evaluate the two classifiers (the one trained with GD on the original training set and the other trained on the new training set after undersampling) on the test set. Compare the confusion matrix, precision, recall and F1 score of the two classifiers. Please report your result.

Solution:
The results using the undersampled training set are shown in Table 3. The plot of the accuracy and loss of the two classifiers versus the iteration step is shown in Figure 2.

Comments:

- After undersampling the majority class, the precision get higher and the recall get lower, implying that the model is more confident in predicting the minority class, but it is less confident in predicting the majority class.
- For $\lambda = 10$, the full gradient descent method finds the global minimum quickly but fails to satisfies the stopping criterion. The stochastic gradient descent oscillates around the global minimum and also fails to reach the target accuracy. This result implies the importance of a proper regularization term. ■

Table 2: The results of logistic regression on imbalanced training set.

Method	λ	TP	FP	TN	FN	Accu.	Prec.	Recall	F1	Time	Iter
GD	0	115	114	1727	66	0.9110	0.5022	0.6354	0.5610	19.0613	133
	0.01	113	112	1729	68	0.9110	0.5022	0.6243	0.5567	22.6330	122
	0.1	99	89	1752	82	0.9154	0.5266	0.5470	0.5366	16.0233	76
	1	18	11	1830	163	0.9139	0.6207	0.0994	0.1714	3.5002	22
	10	0	0	1841	181	0.9105	NaN	0.0000	0.0000	1.0105	6
SGD	0	55	33	1808	126	0.9214	0.6250	0.3039	0.4089	0.0237	92
	0.01	57	31	1810	124	0.9233	0.6477	0.3149	0.4238	0.0245	92
	0.1	41	23	1818	140	0.9194	0.6406	0.2265	0.3347	0.0180	73
	1	3	3	1838	178	0.9105	0.5000	0.0166	0.0321	0.0046	13
	10	0	1	1840	181	0.9100	0.0000	0.0000	0.0000	0.0012	4

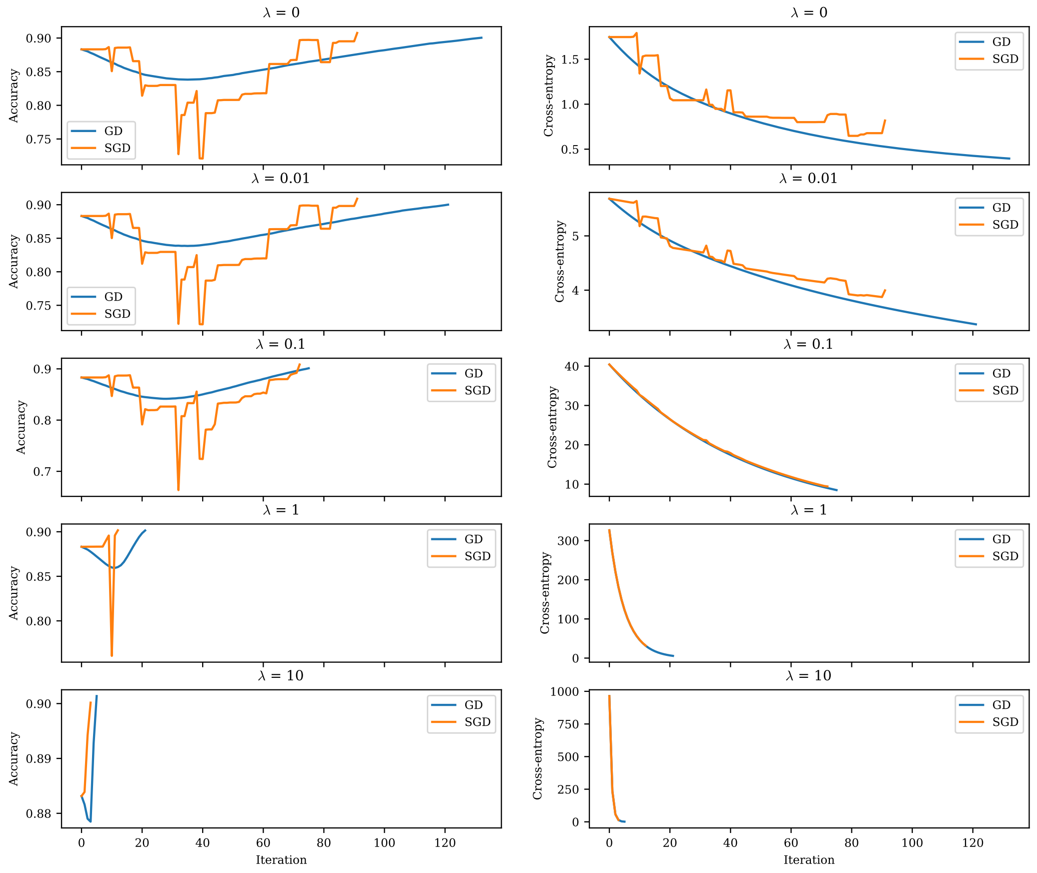

Figure 1: The accuracy and loss of logistic regression on imbalanced training set.

Table 3: The results of logistic regression on undersampled training set.

Method	λ	TP	FP	TN	FN	Accu.	Prec.	Recall	F1	Time	Iter
GD	0.0	161	223	1618	20	0.8798	0.4193	0.8895	0.5699	5.8963	155
	0.01	162	222	1619	19	0.8808	0.4219	0.8950	0.5735	5.1759	142
	0.1	164	226	1615	17	0.8798	0.4205	0.9061	0.5744	4.8542	86
	1.0	169	235	1606	12	0.8778	0.4183	0.9337	0.5778	1.0152	26
	10.0	178	411	1430	3	0.7953	0.3022	0.9834	0.4623	Failed	
SGD	0.0	166	194	1647	15	0.8966	0.4611	0.9171	0.6137	0.0543	191
	0.01	159	141	1700	22	0.9194	0.5300	0.8785	0.6611	0.0537	191
	0.1	141	81	1760	40	0.9402	0.6351	0.7790	0.6998	0.0709	156
	1.0	151	137	1704	30	0.9174	0.5243	0.8343	0.6439	0.0498	160
	10.0	0	0	1841	181	0.9105	NaN	0.0000	0.0000	Failed	

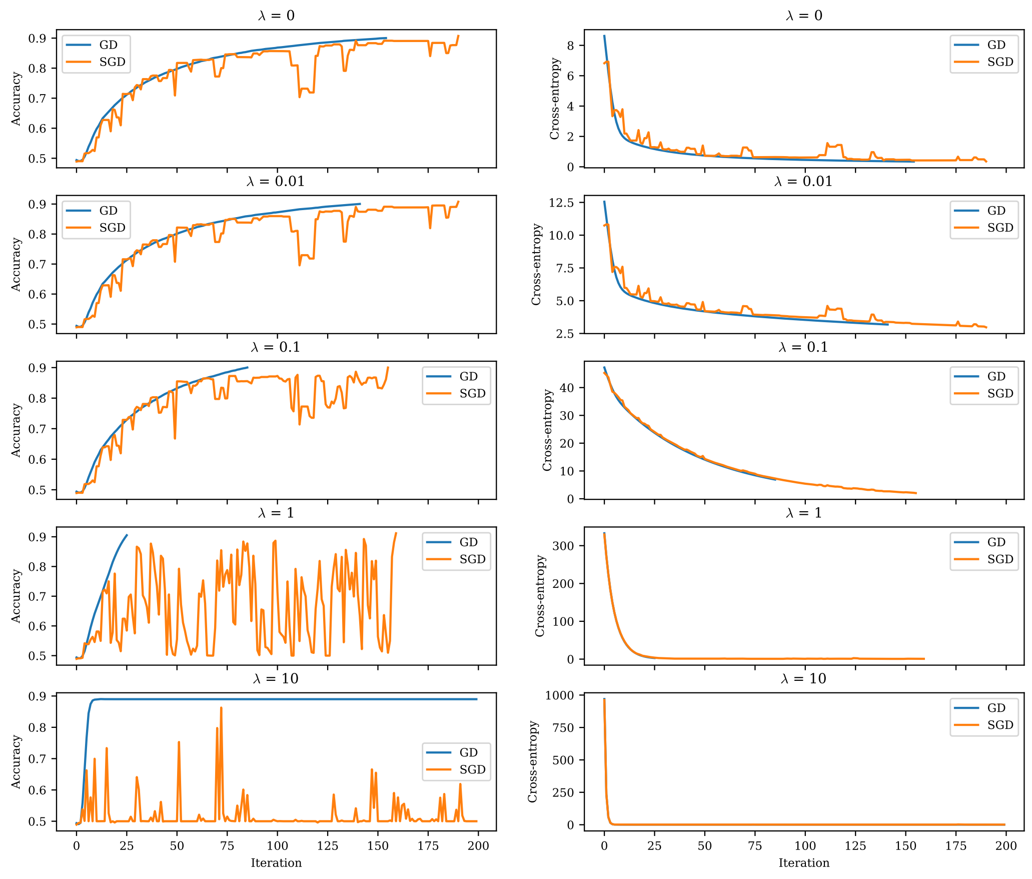

Figure 2: The accuracy and loss of logistic regression on undersampled training set.

```
# Logistic Regression

import os
import time
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rcParams['mathtext.fontset'] = 'cm'
mpl.rcParams['font.size'] = 8
mpl.rcParams['font.family'] = 'serif'

# Load MNITS dataset from hw5_lr.mat
import scipy.io as sio
mat = sio.loadmat('hw5_lr.mat')
train_data, test_data = mat['train_data'], mat['test_data']
train_label, test_label = mat['train_label'], mat['test_label']

# Normalize pixel values
train_data = (train_data / 255).reshape(-1, 28 * 28)
test_data = (test_data / 255).reshape(-1, 28 * 28)

# Convert label for binary classification (digit 6 or not)
train_label = (train_label == 6).astype(int).flatten()
test_label = (test_label == 6).astype(int).flatten()


def evaluate_binary_classifier(label, pred):
    '''
    Evaluate the performance of a binary classifier.
    '''
    '''
    Evaluate the performance of a binary classifier.
    '''
    # Compute the confusion matrix
    TP = np.sum((label == 1) & (pred == 1))
    FP = np.sum((label == 0) & (pred == 1))
    TN = np.sum((label == 0) & (pred == 0))
    FN = np.sum((label == 1) & (pred == 0))
    confusion_matrix = pd.DataFrame([[TP, FP], [FN, TN]],
                                    index=[1, 0],
                                    columns=[1, 0])

    # Compute the accuracy, precision, recall, and F1 score
    accuracy = (TP + TN) / (TP + FP + TN + FN)
    precision = TP / (TP + FP)
    recall = TP / (TP + FN)
    f1_score = 2 * TP / (2 * TP + FP + FN)

    # Print the results
    print(f'Confusion matrix:\n{confusion_matrix}\n')
    print(f'Accuracy:  {accuracy:.2f}')
    print(f'Precision: {precision:.2f}')
    print(f'Recall:    {recall:.2f}')
    print(f'F1 score:  {f1_score:.2f}')
    print()

    # Return all the results in a Dataframe
    return pd.DataFrame(
        [[TP, FP, TN, FN, accuracy, precision, recall, f1_score]],
        index=['result'],
        columns=[
            'TP', 'FP', 'TN', 'FN', 'Accuracy', 'Precision', 'Recall',
            'F1 score'
        ])


def sigmoid(x):
    '''
    Sigmoid function.
    '''
    if x >= 0:
        z = np.exp(-x)
        return 1 / (1 + z)
    else:
        z = np.exp(x)
        return z / (1 + z)


def log(x):
    '''
    Log function.
    '''
    if x == 0:
        return -1e10
    else:
        return np.log(x)


sigmoid = np.vectorize(sigmoid)
log = np.vectorize(log)


class BinaryClassifier:
    '''
    Binomial Logistic Regression using GD, SGD, or mini-batch SGD with L2 regularization.
    '''

    def __init__(self,
                 alpha=0.01,
                 learning_rate=None,
                 batch_size=None,
                 target_accuracy=0.90,
                 max_iter=5000,
                 random_state=0):
        self.alpha = alpha
        self.learning_rate = learning_rate
        self.batch_size = batch_size
        self.target_accuracy = target_accuracy
        self.max_iter = max_iter
        self.random_state = random_state

        self.n = None  # Number of samples
        self.d = None  # Number of features
        self.X = None  # Feature matrix with bias
        self.y = None  # Label vector
        self.w = None  # Weight vector
        self.L = None  # Lipschitz constant
        self.prob = None  # Predicted probabilities
        self.grad = None  # Gradient of loss
        self.loss_list = []  # Cross-entropy loss
        self.score_list = []  # Accuracy score
        self.total_iter = 0  # Total number of iterations
        self.total_time = 0  # Total time elapsed
        self.converged = False  # Whether the algorithm converged

    def fit(self,
            X,
            y,
            w0=None,
            verbose=False,
            record_loss=False,
            record_score=True):
        '''
        Train the classifier.
        '''
        # Set the random seed
        np.random.seed(self.random_state)
        # Initialize the parameters
        self.n = X.shape[0]
        self.d = X.shape[1]
        self.X = np.hstack((np.ones((self.n, 1)), X))
        self.y = y
        self.w = w0.copy()
        if self.w is None:
            self.w = np.random.randn(self.d + 1)
        if self.learning_rate is None:
            # Estimate the Lipschitz constant (See Problem 1)
            self.L = (self.d + 1) / 4
        else:
            self.L = 1 / self.learning_rate
        self.score_list.clear()
        self.loss_list.clear()
        self.total_time = 0
        self.converged = False
        # Minimize the loss function
        for self.total_iter in range(1, self.max_iter + 1):
            # Start timer
            start = time.time()
            # Compute the gradient of the loss function
            if self.batch_size is None:
                # Full gradient
                self.prob = sigmoid(self.X @ self.w)
                self.grad = self.X.T @ (self.prob - self.y) / self.n
            else:
                # Stochastic gradient
                idx = np.random.randint(self.X.shape[0], size=self.batch_size)
                self.prob = sigmoid(self.X[idx] @ self.w)
                self.grad = self.X[idx].T @ (self.prob -
                                             self.y[idx]) / self.batch_size
            # Update the weights
            self.w -= 1 / (self.L + self.alpha) * (self.grad +
                                                    self.alpha * self.w)
            # Stop timer
            duration = time.time() - start
            self.total_time += duration
            # As we need to evaluate the accuracy score in each iteration, we must
            ↪  compute the full probability vector here, which cause the stochastic
            ↪  gradient descent to be as slow as the full gradient descent.
            if self.batch_size is not None:
                self.prob = sigmoid(self.X @ self.w)
            # Compute the cross-entropy loss
            loss = self.loss()
            if record_loss:
                self.loss_list.append(self.loss())
            # Compute the accuracy score
            score = self.score()
            if record_score:
                self.score_list.append(score)
            # Print the progress
            if verbose:
                print(f'Iteration {self.total_iter:4d}: '
                      f'Loss = {loss:.4f}, '
                      f'Score = {score:.4f}, '
                      f'Time = {duration:.4f} sec')
            # Stop if the target accuracy is reached
            if self.target_accuracy < score:
                self.converged = True
                break

    def predict(self, X):
        '''
        Predict the labels.
        '''
        prob = self.predict_proba(X)
        return (prob > 0.5).astype(int)

    def predict_proba(self, X):
        '''
        Predict the probabilities of the positive class.
        '''
        X = np.hstack((np.ones((X.shape[0], 1)), X))
        return sigmoid(X @ self.w)

    def loss(self):
        '''
        Evaluate the cross-entropy loss on the training set.
        '''
        return -np.mean(self.y * log(self.prob) +
                        (1 - self.y) * log(1 - self.prob)
                        ) + self.alpha / 2 * np.sum(self.w**2)

    def score(self):
        '''
        Evaluate the accuracy on the training set.
        '''
        return np.mean((self.prob >= 0.5).astype(int) == self.y)

    def plot_loss(self, ax=None, **kwargs):
        '''
        Plot the loss function.
        '''
        if ax is None:
            ax = plt.gca()
        ax.plot(self.loss_list, **kwargs)

    def plot_score(self, ax=None, **kwargs):
        '''
        Plot the accuracy score.
        '''
        if ax is None:
            ax = plt.gca()
        ax.plot(self.score_list, **kwargs)

    def plot(self, ax=None, **kwargs):
        if ax is None:
            _, ax = plt.subplots(1, 2, figsize=(12, 4))
        self.plot_loss(ax[0], **kwargs)
        self.plot_score(ax[1], **kwargs)

    def report(self, data, label):
        '''
        Test the classifier on the given dataset.
        '''
        pred = self.predict(data)
        df = evaluate_binary_classifier(label, pred)
        df['Time'] = self.total_time
        df['Iter'] = self.total_iter
        df['Converged'] = self.converged
        return df


n = train_data.shape[0]
d = train_data.shape[1]
X = np.hstack((np.ones((n, 1)), train_data))
# Compute a Lipschitz constant
L = 0.25 * np.linalg.norm(X, ord=2)**2 / n
# Generate random initial weights
np.random.seed(0)
w0 = np.random.randn(d + 1)
```

```python
clf = BinaryClassifier(learning_rate=1 / L, random_state=0)

lamb_list = [0, 0.01, 0.1, 1, 10]

fig, ax = plt.subplots(len(lamb_list),
                       2,
                       figsize=(12, 2 * len(lamb_list)),
                       sharex=True)
for i in range(len(lamb_list)):
    ax[i, 0].set_title(f'$\lambda$ = {lamb_list[i]}')
    ax[i, 1].set_title(f'$\lambda$ = {lamb_list[i]}')
    ax[i, 0].set_ylabel('Accuracy')
    ax[i, 1].set_ylabel('Cross-entropy')
ax[len(lamb_list) - 1, 0].set_xlabel('Iteration')
ax[len(lamb_list) - 1, 1].set_xlabel('Iteration')

res1 = []
# CASE1: Imbalanced dataset & Full gradient
print('\nImbalanced dataset & Full gradient\n')
for i, lamb in enumerate(lamb_list):
    print(f'lambda = {lamb}')
    print('----------------')
    clf.alpha = lamb
    clf.batch_size = None
    clf.fit(train_data, train_label, w0=w0, verbose=True, record_loss=True)
    res1.append(clf.report(test_data, test_label))
    clf.plot_score(ax[i, 0], label='GD')
    clf.plot_loss(ax[i, 1], label='GD')
res1 = pd.concat(res1)

res2 = []
# CASE2: Imbalanced dataset & Stochastic gradient
print('\nImbalanced dataset & Stochastic gradient\n')
for i, lamb in enumerate(lamb_list):
    print(f'lambda = {lamb}')
    print('----------------')
    clf.alpha = lamb
    clf.batch_size = 1
    clf.fit(train_data, train_label, w0=w0, verbose=True, record_loss=True)
    res2.append(clf.report(test_data, test_label))
    clf.plot_score(ax[i, 0], label='SGD')
    clf.plot_loss(ax[i, 1], label='SGD')
    ax[i, 0].legend()
    ax[i, 1].legend()
res2 = pd.concat(res2)

fig.savefig('imbalanced.pdf')
res1.index = lamb_list
res2.index = lamb_list
res = pd.concat([res1, res2], keys=['GD', 'SGD'])
res.to_csv('imbalanced.csv')

fig, ax = plt.subplots(len(lamb_list),
                        2,
                        figsize=(12, 2 * len(lamb_list)),
                        sharex=True)
for i in range(len(lamb_list)):
    ax[i, 0].set_title(f'$\lambda$ = {lamb_list[i]}')
    ax[i, 1].set_title(f'$\lambda$ = {lamb_list[i]}')
    ax[i, 0].set_ylabel('Accuracy')
    ax[i, 1].set_ylabel('Cross-entropy')
ax[len(lamb_list) - 1, 0].set_xlabel('Iteration')
ax[len(lamb_list) - 1, 1].set_xlabel('Iteration')

clf.max_iter = 200

res3 = []
# CASE3: Undersampled dataset & Full gradient
print('\nUndersampled dataset & Full gradient\n')
for i, lamb in enumerate(lamb_list):
    print(f'lambda = {lamb}')
    print('----------------')
    clf.alpha = lamb
    clf.batch_size = None
    clf.fit(train_data_rus, train_label_rus, w0=w0, verbose=True, record_loss=True)
    res3.append(clf.report(test_data, test_label))
    clf.plot_score(ax[i, 0], label='GD')
    clf.plot_loss(ax[i, 1], label='GD')
res3 = pd.concat(res3)

res4 = []
# CASE4: Undersampled dataset & Stochastic gradient
print('\nUndersampled dataset & Stochastic gradient\n')
for i, lamb in enumerate(lamb_list):
    print(f'lambda = {lamb}')
    print('----------------')
    clf.alpha = lamb
    clf.batch_size = 1
    clf.fit(train_data_rus, train_label_rus, w0=w0, verbose=True, record_loss=True)
    res4.append(clf.report(test_data, test_label))
    clf.plot_score(ax[i, 0], label='SGD')
    clf.plot_loss(ax[i, 1], label='SGD')
    ax[i, 0].legend()
    ax[i, 1].legend()
res4 = pd.concat(res4)

fig.savefig('undersampled.pdf')
res3.index = lamb_list
res4.index = lamb_list
res = pd.concat([res3, res4], keys=['GD', 'SGD'])
res.to_csv('undersampled.csv')
```

References

[1] A. Beck. *First-Order Methods in Optimization.* MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Oct. 2017.

[2] A. Beck and M. Teboulle. A fast iterative shringkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.*, 2:183–202, 2009.