1. **Loss functions**
In our first look at the regression problem in this course, we considered how to predict $y \in \mathbb{R}$ given $x \in \mathbb{R}^d$ by finding a vector $w$ minimizing the least squares loss function
\[
\|y - Xw\|^2.
\]
This problem is also called $\ell_2$ regression, and the loss is sometimes also called a quadratic loss. However, now that we have grown more sophisticated both in modeling and in optimization, we understand that the quadratic loss is not always the best choice.

Please list at least two cases where we should use a loss function that is not quadratic. For each, state the input space $\mathcal{X}$, the output space $\mathcal{Y}$, describe the loss function and regularizer you would use for this problem (and, optionally, any feature transformations), and explain why your choice of loss function and regularizer make sense for this problem. Feel free to use a problem you’ve encountered in your class project.

2. **Proximal Gradient Method**
The proximal operator of a function $r : \mathbb{R}^d \to \mathbb{R}$ is defined as
\[
\text{prox}_r(z) = \arg\min_w \left( r(w) + \frac{1}{2} \|w - z\|_2^2 \right).
\]
In class, we saw how to use the $\ell_1$ regularizer to encourage sparsity. In this problem, we will see a different regularizer that enforces sparsity.

(a) Define the $k$-sparse indicator $1_k : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$,
\[
1_k(w) = \begin{cases} 
0 & \text{nnz}(w) \leq k \\
\infty & \text{otherwise}
\end{cases}
\]
where $\text{nnz}(w) =$ the number of non-zero entries of $w$.
Compute the proximal operator of the $k$-sparse indicator $1_k$. 

(b) A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if the line between any two points on the function lies (weakly) above the graph of the function. Formally, $\forall x, y \in \mathbb{R}^d$ and $\forall t \in [0,1]$, 
$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y).$$

Find a counterexample in two dimensions that shows the 1-sparse indicator is not convex. (That is, $d = 2$ and $k = 1$.)

(c) Our goal now is to solve the Sparse Least Squares (SLS) problem,
$$\min ||Xw - y||^2 + 1_k(w).$$

Write pseudocode describing how the proximal gradient method could be used to solve this problem. (That is, write the prox and gradient steps explicitly for this loss function and regularizer.)

(d) Code the proximal gradient method for the SLS problem and run it on the instance in ProxGradHomework.ipynb. Recall that the Lipschitz constant of the gradient of the least squares objective is $L = 2||X||^2$, where $||X||$ is the maximum singular value of $X$. Make sure to use an appropriate step size to ensure convergence. Plot the objective value as a function of the number of iterations. You may want to plot the $y$ axis of the plot on a log scale using `semilogy()` instead of the `plot()` command in the package `PyPlot`. (This is called a convergence plot; it helps us understand how quickly the method finds a solution, and the quality of that solution.)

(e) Run the algorithm starting at multiple locations and create a histogram of the objective value. Use 100 iterations for each run. What do you observe?

(f) Solve the LASSO problem ($\ell_1$ regularized least squares regression) using the proximal gradient method on this problem. You may use the code from the demo in class, found at https://github.com/ORIE4741/demos/blob/master/ProximalGradient.ipynb.

(g) Does LASSO converge to the same place starting from different initial vectors $w^0$?

(h) Compare the SLS solution with the LASSO solution. Which is more sparse? Which achieves a better objective value? Which method is more reliable?


(a) Write pseudocode for the stochastic proximal gradient method applied to the Sparse Least Squares problem above.

(b) Code the stochastic proximal gradient method for the Sparse Least Squares problem.
Using the same data used in problem 2, plot the objective value as a function of the number of iterations.

How long does the stochastic proximal gradient method take compared to the standard proximal gradient method? Compare both the number of iterations and the time required for convergence. You may find Julia’s `@time` macro useful: place it in front of line of code to evaluate the running time of that line.

Run the stochastic algorithm starting at multiple locations and create a histogram of the final objective values. What do you observe?


In class we have defined the subgradient $\partial f$ for a function $f : \mathbb{R} \rightarrow \mathbb{R}$ in the following way:

- If $f$ is differentiable at $x$, $\partial f = \{\nabla f(x)\}$, which is a singleton set containing the gradient.
- If $f$ is not differentiable at $x$, $\partial f$ is any convex combination of the gradient of $f$ at nearby points. Formally,

\[
\partial f = \text{conv} \left( \left\{ \lim_{y \to x} \nabla f(y) \right\} \right),
\]

where $\text{conv}(A)$ is the convex hull of set $A$.

(This definition works for “proper” convex functions, which are differentiable almost everywhere.)

Suppose we want to use proximal gradient method to solve

\[
\min \|y - Xw\|_1 + \|w\|_2^2,
\]

where $X = 1$ and $y = 0$. (This is a particularly simple learning problem with only one example and with no covariates! $n = 1$ and $d = 0$.)

(a) Write a formula to evaluate the subgradient of the loss function $\ell(w) = \|y - Xw\|_1$.

(b) Write a formula to evaluate the proximal operator of the regularizer $r(w) = \|w\|_2^2$.

(c) Run the proximal gradient method with constant step size $\alpha = 1$ and starting point $w^0 = 1$. Repeat the iteration for several steps and record $w^0, w^1, w^2, \ldots$. Make a plot of the value of the iterate $w^t$ as a function of the iteration $t$. Do the iterates seem to converge?

(d) Repeat the previous question with several different step sizes and starting points. Describe what you observe and explain why you think it is happening.

(e) Do you see a problem with using the proximal gradient method for this function? How would you fix it?