Regresión Lineal

En la operación de sistemas fotovoltaicos, la potencia entregada depende directamente de la irradiancia solar disponible, pero solo cuando esta supera cierto umbral a partir del cual los equipos comienzan a generar de manera efectiva. Por debajo de dicho nivel la producción es prácticamente nula, mientras que por encima se observa una relación creciente y aproximadamente lineal entre irradiancia y potencia. Además, factores ambientales y de medición introducen inevitablemente cierto grado de variabilidad. Sea X la irradiancia neta centrada y escalada entre $[-1, 1]$ e Y la potencia normalizada, se desea predecir la potencia producida en función de la irradiancia.

(**a**) *Simulación*:

1. Utilizando `random.uniform` (numpy), generar un dataset de 500 pares de muestras (X, Y) de la siguiente forma: $X \sim \mathcal{U}(-1, 1)$, $Y|X = x \sim \mathcal{N}(r(x), \sigma^2)$ con $\sigma^2 = 0.04$ y $r(x) = x\mathbf{1}\{x > 0\}$ (función conocida como ReLU).
2. Graficar los datos utilizando `scatter` (matplotlib).

(**b**) *Análisis Teórico*:

1. Calcular analíticamente $\mathbf{E}[X]$, $\mathbf{E}[Y]$, $\text{var}(X)$, $\text{var}(Y)$ y $\mathbf{E}[XY]$. : No es necesario ni deseable resolver integrales con respecto a "y" para computar estas magnitudes.
2. Indicar la función de regresión óptima y el error bayesaino asociado.
3. ¿Cuál es el mejor predictor sin observar X? Comparar el error esperado que comete en comparación con el regresor óptimo.

(**c**) *Regresión Lineal:*

1. Implementar una regresión lineal (matricial) a partir de los datos generados previamente. El código debe estar estructurado de la siguiente manera:

```
class RegLineal:
    # Opcional, para inicializar atributos o declarar hiperparámetros
    def __init__(self,...

    # Etapa de entrenamiento
    def fit(self,X,y):

    # Etapa de testeo
    def predict(self,X):

    # Cómputo del error
    def err_predict(self,X,y):
```

   A su vez, se debe poder extraer el atributo `RegLineal.params` (parámetros encontrados durante el entrenamiento).
2. Entrenar la regresión lineal con el dataset generado anteriormente.
3. Indicar el error de entrenamiento.
4. Utilizar el regresor para predecir la potencia producida para una irradiancia de 0.5.
5. Graficar el regresor obtenido superpuesto al óptimo y al scatter.

(**d**) *Gradiente Descendente:*

1. Agregar un método `fit_gradient(self,X,y,learning_rate)` a la implementación desarrollada anteriormente que reemplace el entrenamiento matricial por gradiente descendente.
2. Repetir los puntos 2, 3, 4 y 5 del inciso (**c**).
3. Graficar la evolución del parámetro w (pendiente de la recta) en función del número de iteraciones.

(**e**) **OPCIONAL:** ¿Como modificaría el código para efectuar una regresión cuadŕatica? Implementar la clase `RegCuadratica` y comparar con los otros regresores.