Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Actividad 2:

Regresiones

Aplicaciones en Ciencia de Datos e Inteligencia Artificial

Profesor : Francisco Pérez Galarce.
Ayudante : Yesenia Salinas

Fecha : 12 de noviembre de 2024

1 Introducción

Las regresiones lineales son ampliamente utilizadas tanto en estadística como en aprendizaje automático. Además de ser muy comunes en contextos prácticos, esta familia de modelos permite comprender conceptos relevantes en ciencia de datos, tales como el sobreajuste, la complejidad y la penalización. En esta actividad, a través de ejemplos prácticos, implementará en Python la regresión lineal simple, las regresiones polinomiales y las regresiones con penalización.

2 Instrucciones de la actividad

2.1 *Solución cerrada de regresión lineal (35 puntos)*

5 ptos Abrir entorno de programación, de preferencia utilizar Visual Studio Code, Google Colab[a] o Jupyter Notebook.

5 ptos Utilizando `Numpy` o `Scipy` genere 1,000 datos que distribuyan uniforme entre 0 y 3, $\mathcal{U}(0, 3)$. Este arreglo representará su variable independiente, también denominado a lo largo de este curso como feature, descriptor o input.

5 ptos A través de la misma librería genere los 1,000 datos de la variable dependiente, también conocida como target u output, de la siguiente forma $y = 5 + 2x + \mathcal{U}(2, 4)$

5 ptos Por medio de `matplotlib` o `seaborn` genere un gráfico de dispersión (scatter plot) entre la variable dependiente y la variable independiente. Ajuste título, nombre de ejes, tamaño de los valores en los ejes, tamaño de la fugura, colores de los y la transparencia de los puntos.

[a] https://colab.research.google.com/notebooks

5 ptos Genere una función que permita visualizar los datos simulados. Esta función debe recibir como argumentos la variable dependiente y, la variable independiente x y el número de datos a simular. La función debe permitir visualizar o guardar la imagen.

5 ptos Aplique las operaciones matriciales necesarias para ajustar una regresión lineal a través de su solución cerrada (use numpy.linalg). Cree una función que reciba la variable dependiente y la variable independiente, esta función debe retornar los parámetros optimizados.

5 ptos Utilizando `Matplotlib` o `seaborn` genere un gráfico que incluya la solución (recta óptima) y los datos utilizados. Cree una función que reciba x, y y los parámetros óptimos y cree dicha visualización.

5 ptos Cree una clase cuyo nombre sea `regressionLineal` que contenga los métodos antes generados. La clase debe ser inicializada con los atributos: tipo de datos (simulado o real) y tamaño de la base de datos.

2.2 *Regresión Polinomial (25 puntos)*

0 ptos Cargue la base de datos adjunta a la actividad (measurements.csv).

5 ptos Mediante `Matplotlib` o `seaborn` visualizar las variables contenidas en la base de datos con un scatter plot.

5 ptos Use la regresión lineal previamente implementada para predecir el consumo (*consume*) mediante la distancia recorrida (*distance*). Se recomienda usar la clase previamente implementada `regressionLineal`, sin embargo, también se permite usar funciones aisladas. El uso de métodos disponibles en sklearn también está permitido pero se evaluará con la mitad del puntaje asignado a este item.

5 ptos Genere una transformación polinomial de la variable independiente, para esta transformación se recomienda utilizar el método disponible por sklearn `preprocessing.PolynomialFeatures`).

5 ptos Ajuste regresiones polinomiales de diferentes grados.

5 ptos Grafique los modelos resultantes, ¿Qué observa en el error cuadrático medio y en la forma de las predicciones a medida que aumenta el grado del polinomio?

2.3 *Regresión con Penalización (bonus 10 puntos)*

2.5 ptos Separe la base de datos en una para ajustar el modelo y otra para testearlo (80%-20%).

2.5 ptos Utilizando la misma base de datos y una transformación polinomial de grado mayor a 5, ajuste en los datos de entrenamiento regresión ridge y Lasso (`from sklearn.linear_model import Lasso, ridge`).

2.5 ptos Modifique el parámetro de penalización y obtenga los valores de los parámetros y grafique las predicciones ¿Qué puede comentar al respecto? ¿Qué ocurre al aumentar el valor de alfa (penalización) en cada modelo?¿qué diferencia se observa entre ridge y Lasso?.

2.5 ptos Seleccione el modelo adecuado de acuerdo al error cuadrático medio.

3 Deploy (opcional)

- Incluya los tres modelos en una clase, de modo que se puedan ajustar, visualizar y predecir con las tres variantes. Considere los parámetros que requiere cada opción.
- Genere una función para guardar el modelo. Para esto puede guardar los coeficientes en un archivo o puede usar una librería como `Pickle`.
- Genere una función para cargar y predecir usando el modelo cargado.
- Cree un notebook que le permita usar la clase creada. Implemente tres ejemplos de uso.
- ¿Cómo podría testear la clase implementada?, proponga métodos para testear que todo está funcionando bien.

3.1 Entrega

- La actividad deberá entregarse en un archivo jupyter notebook (.ipynb), subirse a la plataforma del curso y subirse a su repositorio del curso en `Github`[b].
- La actividad debe realizarse en grupos de 2 o 3 personas que serán creados de forma aleatoria. Se trabajará en modalidad pair programming durante la clase, por lo tanto se recomienda tener instalado el complemento Live share.
- La actividad debe ser subida a la plataforma antes del miércoles 12 a las 21:59 P.M. Sin embargo, se recomienda entregarla al final de la clase. Aquellos estudiantes que envíen su actividad antes de la fecha indicada tendrán una bonificación de 0.5 décimas.

[b] https://github.com/