Bloque 2
Aprendizaje Automático

Práctica 2:

Sesión 3
Aplicación de los algoritmos de Perceptrón y Regresión Logística a varios datasets

Sesiones de la práctica 3

Sesión 1:

- Familiarizarse con el entorno de trabajo (Google Colab)
- Analizar conjuntos de datos (datasets): **iris**, **digits**, **olivetti**, **openml**

Sesión 2:

- Aplicación del algoritmo del Perceptron a tareas de clasificación: dataset **iris**.
- **Ejercicio: A**plicar Perceptrón a **digits** y **olivetti**

Sesión 3:

- Aplicación de Regresión Logística a tareas de clasificación: dataset **iris**.

- **Ejemplo de examen:**
  - Aplicación de Regresión Logística a un dataset de OpenML.

Sesión 4 (examen):

- Se pedirá la aplicación de Regresión Logística para una tarea diferente de OpenML
- Hay que subir la solución del Ejercicio

Sesión 3: Regresión Logística (RL)

- Vamos a utilizar regresión logística aplicado a tareas de clasificación
  - Usaremos la función `LogisticRegression` de sklearn

Aprendizaje por mínima NLL: aprendizaje por máxima verosimilitud planteado como un problema de minimización

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}}\ \mathrm{NLL}(\mathbf{W})$$

Descenso por gradiente aplicado a regresión logística:

$$\mathbf{W}_0 = \mathbf{0}; \quad \mathbf{W}_{i+1} = \mathbf{W}_i - \eta_i \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n (\boldsymbol{\mu}_n - \boldsymbol{y}_n)^t$$

Sesión 3: Regresión Logística (RL)

En sklearn:

```
LogisticRegression(random_state=23,
          solver=solver,
          tol=tol,
          C=C,
          max_iter=max_iter).fit(X_train, y_train)
```

Donde:

- El argumento **solver** especifica qué algoritmo utilizar para la optimización de los parámetros del modelo.
- El argumento **tol** especifica la tolerancia para la detención del procedimiento de optimización.
- El argumento **C** es el inverso de la fuerza de regularización. La regularización puede ayudar a prevenir el sobreajuste reduciendo la magnitud de los parámetros del modelo.
- **Max_iter** especifica el número máximo de iteraciones que el solucionador debe usar para encontrar los parámetros del modelo.

Sesión 3: Regresión Logística (RL)

Nociones importantes:

Subajuste (Underfitting):

- Ocurre cuando el modelo no logra capturar la complejidad del conjunto de datos
- Esto sucede porque el modelo es demasiado simple para representar los datos.
- Produce % de error alto en entrenamiento y test

Sobreajuste (Overfitting):

- Ocurre cuando el modelo aprende no solo los patrones generales de los datos, sino también el ruido y las irregularidades del conjunto de entrenamiento
- El modelo podría tener dificultades para clasificar correctamente, ya que solo "recuerda" los datos de entrenamiento.
- Produce % de error bajo en entrenamiento pero alto en test.

Sesión 3: Regresión Logística (RL)

Solver

La función LogisticRegression de sklearn ofrece la posibilidad de utilizar varios solvers:

liblinear: buena opción para datasets pequeños; se limita a resolver problemas de una clase frente al resto, no para determinar la probabilidad de cada clase

sag, saga: buenas opciones para datasets grandes

sag, saga, lbfgs, newton-cg: solo estos solvers manejan pérdida multinomial para problemas multi-clase

newton-cholesky: buena opción cuando número de muestras >> número de características; se limita a resolver problemas de clasificación binaria y problemas de una clase frente al resto

Sesión 3: Regresión Logística (RL)

Solver (RECOMENDACIONES)

Utilizar **sag, saga, lbfgs, newton-cg** que son para problemas multi-clase

La librería sklearn utiliza **lbfgs** por defecto

Se puede priorizar **sag** o **saga** para datasets grandes

Sesión 3: Regresión Logística (RL)

Regularización

Añade una penalización para evitar el sobreajuste (overfitting), es decir, para conseguir que el algoritmo generalice bien en conjuntos de test

Penaliza los coeficientes altos de la función lineal (sin incluir el término independiente)

> cuando una característica de las muestras solo ocurre en una clase (y en el resto de clases los valores de dicha característica son 0) el algoritmo de RL asignará coeficientes muy altos a dicha característica. En este caso, el modelo probablemente se ajustará demasiado al conjunto de entrenamiento.

$$\text{argmin}_W \ \text{NLL}(\mathbf{W}) + \ \lambda \ \text{R}(\mathbf{W})$$

parámetro $\lambda \ \in \ \mathbb{R}^{>0}$

término de regularización

Sesión 3: Regresión Logística (RL)

Regularización

Hay varios tipos de regularización $\mathrm{R}(\mathbf{W})$. La librería sklearn aplica por defecto una regularización llamada regularización L2 o Gauss.

$\lambda = \frac{1}{\mathrm{C}}$ → parámetro C que utiliza la función LogisticRegression de sklearn

- Por defecto, C=1.0
- Para valores próximos a 0, λ es muy grande y por tanto se aplica máxima regularización lo cual puede conllevar una posibilidad de **sub-ajuste** al conjunto de entrenamiento
- Para valores muy altos de C, , λ es muy pequeño y por tanto se aplica mínima regularización lo cual puede conllevar una posibilidad de **sobre-ajuste** al conjunto de entrenamiento

- C → 0 puede provocar **subajuste**
- C → valores altos puede provocar **sobreajuste**

Sesión 3: Regresión Logística (RL)

Tolerancia

Umbral de tolerancia (tol) para el criterio de parada (para acabar el entrenamiento)

El algoritmo RL dejará de buscar un mínimo una vez se alcanza cierta tolerancia, es decir, cuando se está suficientemente cerca del objetivo.

Por defecto, **tol** = 0.0001 ($1e^{-4}$)

Early stopping

Una forma indirecta de aplicar 'regularización' es controlando el máximo número de iteraciones que se ejecuta el algoritmo de RL.

Sesión 3

Ejemplo en Iris (irisRL.ipynb)

- Aplicar el **algoritmo RL** al dataset Iris para diferentes valores de **solver, tolerancia, C y max_iter**
- Mostrar los resultados del siguiente modo:

```
#   Iter     solver        C         tol        Ete
#-------------------------------------------------------
      10      lbfgs    0.010      0.0001      0.133
      10      lbfgs    0.010      0.0100      0.133
      10      lbfgs    0.010      1.0000      0.133
      10      lbfgs    0.010    100.0000      0.600
      10      lbfgs    0.010  10000.0000      0.600
#-------------------------------------------------------
      10      lbfgs    0.100      0.0001      0.000
      10      lbfgs    0.100      0.0100      0.000
      10      lbfgs    0.100      1.0000      0.000
      10      lbfgs    0.100    100.0000      0.600
      10      lbfgs    0.100  10000.0000      0.600
#-------------------------------------------------------
      10      lbfgs    1.000      0.0001      0.000
      10      lbfgs    1.000      0.0100      0.000
      10      lbfgs    1.000      1.0000      0.000
      10      lbfgs    1.000    100.0000      0.600
      10      lbfgs    1.000  10000.0000      0.600
#-------------------------------------------------------
      10      lbfgs   10.000      0.0001      0.000
      10      lbfgs   10.000      0.0100      0.000
      10      lbfgs   10.000      1.0000      0.000
      10      lbfgs   10.000    100.0000      0.600
      10      lbfgs   10.000  10000.0000      0.600
#-------------------------------------------------------
      10      lbfgs  100.000      0.0001      0.000
      10      lbfgs  100.000      0.0100      0.000
      10      lbfgs  100.000      1.0000      0.000
```

- Indica cuál crees que es la mejor combinación de iteraciones, solver, C y tolerancia

Sesión 3

Ejemplo de examen

- Aplicar el algoritmo RL a un dataset de OpenML para diferentes valores de solver, tolerancia, C y max_iter
    - Consultar 01_ejemplo_examen.ipynb

- ¿Cómo será el examen?
    - Misma idea del ejemplo
    - Se proporcionará un dataset identificado con su ID
    - Se pedirá ejecutar el algoritmo RL para diferentes valores de solver, tolerancia, C y max_iter y observar el error de entrenamiento y de test
    - Se planteará alguna cuestión respecto a que parámetros usar

Bloque 2
Aprendizaje Automático

Práctica 2:

Aplicación de los algoritmos de Perceptrón y Regresión Logística a varios datasets