CC3045 – Inteligencia Aritificial

# Laboratorio 2

## Instrucciones

- Esta es una actividad en grupos de no más de 3 integrantes.
    - Recuerden unirse al grupo de canvas
- No se permitirá ni se aceptará cualquier indicio de copia. De presentarse, se procederá según el reglamento correspondiente.
- Tendrán hasta el día indicado en Canvas.
    - No se confíen, aprovechen el tiempo en clase para entender todos los ejercicios y avanzar lo más posible.
- NOTA: Limiten el uso de IA generativa. Intenten primero buscar en fuentes de internet y si en verdad necesitan usarla, asegúrense de colcar el prompt que utilizar para cada task donde corresponda, así como una explicación de por qué ese prompt funcionó.

## Introducción:

En la industria, aproximadamente el 80% de los errores en IA no son de código, son conceptuales. De nada sirve saber programar un descenso de gradiente si lo aplican sobre una función no convexa, o si usas un KNN sobre datos desbalanceados sin entender las consecuencias. Por eso, en este laboratorio tendremos dos fase:

1. Demostrar que entienden las limitaciones matemáticas de los modelos
2. Construir un sistema de detección de Phishing desde cero, sin la ayuda de librerías como scikitlearn para el entrenamiento, para enteder realmente cómo aprende la máquina.

## Task 1

Responda las siguientes preguntas justificando su respuesta basándose en el material visto en clase. Se espera que demuestren análisis del caso y no solamente un “copy-paste”.

### El Problema de la Convexidad (Regresión Logística)

Durante la clase, se habló que no podemos usar el Error Cuadrático Medio (MSE) para la Regresión Logística. Con esto en mente considere el escenario donde usted asume el rol de Lead AI Engineer y un Junior le presenta un modelo de clasificación binaria que usa una función sigmoide, pero insiste en entrenarlo usando la fórmula de MSE porque “es lo que usó en regresión lineal y funcionaba bien”. Responda

- Explíquele técnicamente (y gráficamente si es necesario) por qué su modelo probablemente se quedará atascado y no encontrará la solución óptima. Mencione el concepto de "mínimos locales" vs "mínimo global".

El uso de MSE con regresión logística genera una función no convexa con múltiples mínimos locales. Cuando se combina la sigmoide $\sigma(z) = 1/(1 + e^{(-z)})$ con $MSE = (1/n) \Sigma(y_i - \sigma(w \cdot x_i + b))^2$, la superficie de error tiene múltiples puntos donde el gradiente se anula. Un mínimo local es donde $f(x^*) \leq f(x)$ solo en una vecindad, mientras que el mínimo global satisface esto para todo el dominio. El descenso del gradiente puede quedar atascado en un mínimo local donde $\nabla f(w) \approx 0$, deteniendo las actualizaciones aunque exista un mínimo global mejor en otra región.

La solución es Log Loss: $J(w,b) = -(1/m) \Sigma[y_i \cdot \log(\sigma(z_i)) + (1-y_i) \cdot \log(1-\sigma(z_i))]$. Esta función es convexa con la sigmoide, garantizando un único mínimo global y convergencia independiente de la inicialización.

## El Problema de la K en KNN

Refierase a las slides correspondientes al tema de KNN. Con esto, considere el escenario donde estamos clasificando enfermedades raras. Tenemos un dataset de 1,000 pacientes sanos (Clase A) y solo 10 pacientes con la enfermedad (Clase B). Responda:

- Si usted elige un K=15 (un número mayor a la cantidad total de casos positivos), ¿qué es lo más probable que ocurra matemáticamente con todas las predicciones para nuevos pacientes enfermos? ¿Por qué el algoritmo de "votación mayoritaria" falla aquí y qué técnica simple de pre-procesamiento (vista en el lab anterior) sería obligatoria antes de usar KNN?

Con 1,000 muestras clase A versus 10 clase B y K=15, todas las predicciones convergen a la clase mayoritaria. La densidad espacial de A es ~100 veces mayor que B. Como K=15 > 10 (total de clase B), incluso si un punto está cerca de todas las muestras de B, los 15 vecinos incluirán al menos 5 de clase A, resultando en votaciones típicas de 14:1 o 15:0 a favor de A.

KNN asume densidades comparables, pero la votación mayoritaria subestima $P(y=B|x)$ porque está dominada por el prior $P(y=A) \approx 0.99$ en lugar de la likelihood real. La solución es balancear las clases mediante oversampling (SMOTE genera muestras sintéticas: $x_{new} = x_i + \lambda(x_j - x_i)$), undersampling (reducir clase mayoritaria), o técnicas híbridas. Con ratio 1:1, la votación refleja similitud geométrica real.

## Polimonios y la Realidad

Observe las gráficas de las diapositivas 38 – 40. Con esto considere y responda. Si su modelo de regresión polinomial tiene una pérdida (Loss) de casi 0.0 en el set de entrenamiento (orden 8), pero al probarlo en producción con datos nuevos el error es gigante:

- ¿Qué nombre técnico recibe este fenómeno?
  **Overfitting (alto variance, bajo bias)**
- Basado en la diapositiva 41, ¿qué está modelando su algoritmo que *no* debería modelar?

El modelo captura el ruido estocástico $\varepsilon_i$ además de la función verdadera $f(x)$. Los datos son $y_i = f(x_i) + \varepsilon_i$, y un polinomio orden 8 con 9 grados de libertad ajusta ambos componentes: $\hat{y}(x) \approx f(x) + \hat{\varepsilon}(x)$.

Según el bias-variance tradeoff: $E[(y - \hat{y})^2] = Bias^2 + Variance + \sigma^2$. En alta complejidad, $Bias^2 \rightarrow 0$ pero $Variance \rightarrow \infty$, resultando en $L_{train} \approx 0$ pero $L_{test} >> L_{train}$ porque el ruido de entrenamiento no correlaciona con el de test. Las soluciones incluyen regularización (L2 penaliza $||w||^2_2$), validación cruzada para seleccionar orden óptimo, early stopping cuando $L_{val}$ aumenta, y más datos ($n >> d$). El objetivo es minimizar el error de generalización balanceando bias y variance.

## Task 2 – Preparación de Datos

Ahora aplicaremos la teoría a un problema real de ciberseguridad. Vamos a detectar sitios web maliciosos.

Dataset: Web Page Phishing Detection Dataset (Kaggle)

Instrucciones:

1. Carga y Limpieza:
    a. Descargue y cargue el dataset. Elimine columnas que sean IDs únicos o irrelevantes (ej: url_length si no está normalizada puede meter ruido, analícelo).
    b. Codificación: Convierta las variables categóricas (como el status del dominio o protocolo) a numéricas.
2. Selección de Features
    a. Para poder graficar la frontera de decisión en 2D en las siguientes partes, seleccione las dos variables independientes que tengan la mayor correlación con la variable objetivo (phishing vs legitimate). Trabajaremos principalmente con estas dos para los gráficos.
3. Escalado (Must):
    a. Recuerden: El Descenso del Gradiente tarda siglos en converger si los datos no están escalados, y KNN colapsa si una dimensión tiene magnitudes mayores a otra.
    b. Implemente normalización (StandardScaler o MinMax) en sus features.
4. Split:
    a. 80% Training, 20% Testing.

## Task 3 – Implementación desde Cero

Reglas importantes:

- Puede usar numpy, pandas, matplotlib.
- No tiene permitido usar sklearn (fit/predict), tensorflow, pytorch para esta sección.
- Deben implementar las fórmulas matemáticas matriciales.

### Regresión Logística con Descenso del Gradiente

Implemente un clasificador binario para detectar Phishing. Para ello, tome en cuenta lo siguiente:

1. Implemente la función sigmoide g(z) y la predicción $\hat{y} = g(X \cdot w + b)$ (Hipótesis) 2. Implemente la función de Log Loss (Binary Cross-Entropy) (Costo)
3. Implemente el bucle de Gradient Descent (Entrenamiento):
    a. En cada época, actualice los pesos w y el sesgo b.
    b. Guarde el historial de pérdida.
    c.

Debe asegurarse de entregar dentro de su notebook:

- Gráfica de Costo: Muestre cómo el error baja en cada época. (Si no baja, su learning rate es muy alto o su gradiente está mal).
- Threshold de Decisión: Haga un scatter plot de las 2 variables seleccionadas (puntos azules=legítimo, rojos=phishing) y dibuje la línea que aprendió su modelo para separarlos.

### K-Nearest Neighbours Manual

Implemente un clasificador de KNN. Para ello tome en cuenta lo siguiente:

1. Cree una función `predict_knn(X_train, y_train, X_test, k)` que use Distancia Euclidiana.
2. Clasifique el set de pruebas usando K=3

Debe asegurarse de entregar dentro de su notebook:

- Genere un mapa de color de fondo (decision boundary) que muestre qué áreas del plano 2D pertenecen a "Phishing" y cuáles a "Legítimo" según su KNN. Superponga los puntos reales.

## Task 4 – Comparación

En esta parte sí deben utilizar librerías como scikit-learn. Con esto en cuenta, realice y responda lo siguiente:

1. Benchmark:
    a. Entrene un LogisticRegression y un KNeighborsClassifier de sklearn con los mismos datos.
    b. Compare los resultados (Accuracy, Precision, Recall) de sus implementaciones manuales vs. las de sklearn.
2. Análisis de Cierre:
    a. ¿Cuál implementación fue mejor y por qué cree que sucedió?
    b. En el contexto de Phishing, ¿qué error es más costoso: bloquear a un usuario legítimo (Falso Positivo) o dejar pasar un ataque (Falso Negativo)? Basado en esto, ¿qué métrica debería priorizar?

### Entregas en Canvas

1. Documento PDF con las respuestas a cada task
2. Archivo .ipynb, o link a repositorio de GitHub (No se acepta entregas en otros medios)
    a. El código debe estar comentado explicando la relación con las fórmulas de las diapositivas.

## Evaluación

1. [1.0 pt] Task 1
2. [0.5 pt] Task 2
3. [2.0 pt] Task 3
4. [0.5 pt] Task 3

Total 4 pts