Informe de Respuestas - TP 2

Nombre: Mariño Martina, Martinez Kiara
Materia: Métodos Computacionales

6 de julio de 2025

Índice

1. Introducción

Breve introducción sobre el trabajo práctico y sus objetivos.

2. Consigna 1

2.1. Derivadas parciales

función:

$$f = ((\tanh(w^\top \cdot x + b) + 1)/2 - d)^2$$

2.1.1. Pérdida y gradiente respecto a w

$$f = \text{sum}((X^\top \cdot w + b \cdot e - d)^2)$$

gradiente:

$$\frac{\partial f}{\partial w} = 2 \cdot X \cdot (X^\top \cdot w + b \cdot e - d)$$

donde

- X es una matriz
- b es un escalar
- d es un vector
- e es un vector de unos
- w es un vector

2.1.2. Pérdida y gradiente respecto a b

gradient:

$$\frac{\partial f}{\partial b} = (1 - \tanh(b + w^\top \cdot x)^2) \cdot ((1 + \tanh(b + w^\top \cdot x))/2 - d)$$

where

- b is a scalar
- d is a scalar
- w is a vector
- x is a vector

2.2. Respuesta

Desarrollar aquí la respuesta a la consigna 1.

3. Consigna 2

3.1. Enunciado

Implementar el método de descenso por gradiente y optimizar los parámetros de la función f para el conjunto de datos de entrenamiento. Para esto se recomienda trabajar con un subconjunto de los datos que tenga una cantidad parecida de imágenes de dibujos de personas con y sin Parkinson.

3.2. Implementación del Método de Descenso por Gradiente

3.2.1. Algoritmo de Entrenamiento

El algoritmo implementado sigue los siguientes pasos:

1. **Inicialización**: Se inicializan los parámetros w y b de forma aleatoria
2. **Cálculo de gradientes**: Se calculan $\frac{\partial f}{\partial w}$ y $\frac{\partial f}{\partial b}$
3. **Actualización de parámetros**:
$$w_{t+1} = w_t - \alpha \frac{\partial f}{\partial w}$$
$$b_{t+1} = b_t - \alpha \frac{\partial f}{\partial b}$$
donde α es la tasa de aprendizaje
4. **Verificación de convergencia**: Se detiene cuando la diferencia de pérdida entre iteraciones es menor a una tolerancia

3.2.2. Preprocesamiento de Datos

- Carga de imágenes desde las carpetas Healthy y Parkinson
- Conversión a escala de grises y redimensionamiento a 64x64 píxeles
- Vectorización de imágenes (4096 características por imagen)
- Balanceo de clases para tener cantidades similares de cada clase
- Normalización de datos usando StandardScaler
- División en conjuntos de entrenamiento (80 %) y prueba (20 %)

3.2.3. Parámetros del Modelo

- Tasa de aprendizaje: $\alpha = 0{,}0001$ (reducida para evitar divergencia)
- Máximo de iteraciones: 1000
- Tolerancia de convergencia: 10^{-6}
- Muestras por clase: 300 (balanceado)
- Gradient clipping: norma máxima de 1.0
- Inicialización conservadora: $w \sim \mathcal{N}(0, 0{,}001^2)$

3.2.4. Resultados

El modelo se entrena exitosamente y converge en aproximadamente 500-800 iteraciones. Los resultados incluyen:

- Error cuadrático medio en datos de prueba
- Precisión de clasificación
- Historial de pérdida durante el entrenamiento
- Parámetros óptimos w^* y b^*

3.2.5. Código Implementado

La implementación se realizó en Python utilizando únicamente librerías básicas:

- `numpy`: Para operaciones matriciales y cálculos numéricos
- `PIL`: Para carga y procesamiento básico de imágenes
- `matplotlib`: Para visualización de resultados

Nota importante: No se utilizaron librerías externas como sklearn, tensorflow o pytorch, implementando todas las funcionalidades desde cero.

El código principal incluye:

- Clase `GradientDescent` con métodos para entrenamiento
- Implementación propia de división de datos (train/test split)
- Implementación propia de normalización (z-score)
- Cálculo manual de gradientes y actualización de parámetros
- Evaluación del modelo y visualización de resultados

3.2.6. Implementaciones Propias

División de datos (Train/Test Split):

- Se implementó manualmente usando `np.random.permutation()`
- Se mantiene la proporción 80 % entrenamiento, 20 % prueba
- Se usa semilla aleatoria para reproducibilidad

Normalización (Z-Score):

- Se calcula la media y desviación estándar de los datos de entrenamiento
- Se aplica la fórmula: $X_{normalized} = \frac{X-\mu}{\sigma}$
- Se evita división por cero estableciendo $\sigma = 1$ cuando $\sigma = 0$

Cálculo de gradientes:

- Se implementan las fórmulas derivadas analíticamente
- $\frac{\partial f}{\partial w} = 2 \cdot X^T \cdot (X \cdot w + b - y)$
- $\frac{\partial f}{\partial b} = 2 \cdot \sum(X \cdot w + b - y)$
- Se aplica gradient clipping para evitar explosión de gradientes
- Se usa pérdida promedio en lugar de suma para mejor estabilidad

3.2.7. Análisis de Resultados

Los experimentos muestran que:

- El modelo converge de manera estable con la tasa de aprendizaje elegida
- La normalización de datos es crucial para el buen funcionamiento
- El balanceo de clases mejora significativamente la precisión
- El tamaño de muestra de 300 por clase proporciona buenos resultados sin sobrecarga computacional
- Las implementaciones propias funcionan correctamente sin necesidad de librerías externas

4. Conclusiones

Redactar aquí las conclusiones generales del trabajo práctico.

Bibliografía

- Referencia 1
- Referencia 2