Optimización

Primer Cuatrimestre 2025

Entrega N°1

Ejercicio 1 (Redes Neuronales Multicapa) El dataset MNIST (del paquete MLDatasets) consta de 70,000 imágenes de dígitos escritos a mano (28×28 píxeles) con etiquetas en $\{0, 1, 2, \ldots, 9\}$. Se desea entrenar una red neuronal multicapa para clasificar los dígitos en sus respectivas categorías. Para ello, se considerará una red neuronal con una capa oculta, donde la entrada x_i es el vector de características de la i-ésima imagen (aplanada en un vector de dimensión 784) y la salida es un vector de probabilidades sobre las 10 clases, calculado mediante la función softmax. La arquitectura de la red se define mediante los siguientes parámetros:

1. Capa oculta con h neuronas y activación sigmoidal:

$$a_i^{(1)} = \sigma(W^{(1)} x_i + b^{(1)}),$$

   donde $W^{(1)}$ es una matriz de pesos de dimensión $h \times 784$ y $b^{(1)}$ es el vector de sesgos de dimensión h.

2. Capa de salida con 10 neuronas y activación softmax:

$$p_i = \text{softmax}(W^{(2)} a_i^{(1)} + b^{(2)}),$$

   donde $W^{(2)}$ es una matriz de pesos de dimensión $10 \times h$ y $b^{(2)}$ es el vector de sesgos de dimensión 10.

Se definen las etiquetas y_i en formato one-hot, de modo que $y_{ij} = 1$ si la imagen i pertenece a la clase j, y 0 en caso contrario. Suponiendo una distribución categórica para la variable respuesta, la log-verosimilitud viene dada por:

$$\ell(W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}) = -\sum_{i=1}^{n} \sum_{j=1}^{10} y_{ij} \ln(p_{ij}).$$

(a) Importe el dataset MNIST, normalice los valores de los píxeles en el rango $[0, 1]$, divida el conjunto en datos de entrenamiento y prueba, y convierta las etiquetas a formato one-hot.

(b) Escriba la función log-verosimilitud en Julia y calcule los gradientes para cada dato usando *back propagation* respecto a los parámetros $W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}$.

(c) Implemente el método de descenso por gradiente con paso constante, iniciando desde valores aleatorios para los parámetros. Experimente con diferentes valores del paso para determinar cuál permite una convergencia estable del algoritmo.

(d) Realizar un gráfico de los valores obtenidos de la función de pérdida utilizando los datos de entrenamiento.

(e) BONUS: Implementar el Método de Gradiente Estocástico, comparar los resultados con lo anterior implementando una función `accuracy` que mida cuan preciso es el modelo, es decir $\frac{\#\text{casos favorables}}{\#\text{casos totales}}$. *Sugerencia:* Para poder programar `accuracy` debemos primero **predecir** los datos. Nuestra red devuelve un vector de probabilidades y debemos predecir a cada dato con la clase que mayor probabilidad tiene de haber sido.

Ejercicio 2 El objetivo es implementar el método de Newton usando la modificación propuesta por Levenberg-Marquardt. Esta consiste en tomar como dirección de descenso $-(\mathbf{H}_k + \mu_k \mathbf{I})^{-1}\mathbf{g}_k$, donde μ_k se elige en cada paso de manera tal que si $\mathbf{H}_k$ es definida positiva, $\mu_k = 0$, y en caso contrario, μ_k es tal que $\mathbf{H}_k + \mu_k \mathbf{I}$ resulte definida positiva. En la práctica, esto equivale a un método intermedio entre Newton ($\mu_k = 0$) y el gradiente (μ_k grande). La elección de μ_k puede resultar compleja, dado que uno querría que $\mu_k \sim -\lambda$, donde λ es el autovalor negativo de $\mathbf{H}_k$ de máximo módulo, y el cálculo de autovalores es un problema complicado.

Implementar el algoritmo de Levenberg-Marquardt tomando μ_k como el mínimo autovalor de $\mathbf{H}_k$ + un valor ε_0.

Aplicar el método a la función de Beale:

$$f(x, y) = (1{,}5 - x + xy)^2 + (2{,}25 - x + xy^2)^2 + (2{,}625 - x + xy^3)^2,$$

cuyo mínimo se encuentra en $(3, 0{,}5)$ y a la función de Ackley:

$$f(x, y) = -20e^{-0{,}2\sqrt{0{,}5(x^2+y^2)}} - e^{0{,}5(\cos 2\pi x + \cos 2\pi y)} + e + 20,$$

cuyo mínimo absoluto está en el origen.

Sugerencia: Puede ser útil la función `ForwardDiff.hessian`