Estimación de un modelo de aprendizaje con regresión logística utilizando métodos de optimizacion numérica

1 Introducción

Tomando como referencia el ejercicio planteado en clase, el cual toma como referencia el artículo publicado por Colubri et. al (2019), se busca entrenar un modelo de regresión logística que permite pronosticar la supervivencia a enfermedad por virus del ébola. Para tal propósito, se utilizan datos recolectados por el Cuerpo Internacional de Medicina (IMC) durante 2014 y 2016 en Liberia y Sierra Leona. El fin de este trabajo es explorar los resultados de pronóstico y desempeño computacional derivados de la utilización de diferentes métodos de optimización numérica aplicados a machine learning, y de la implementación de métodos de cómputo en paralelo durante el proceso de estimación.

2 Objetivos

El objetivo principal de este trabajo es evaluar el rendimiento de los métodos de optimización mencionados en la sección 4. Dado lo anterior, definimos rendimiento en términos de métricas de desempeño del modelo, y también en métricas de desempeño durante el proceso de computo. A continuación, se describe el planteamiento del problema a resolver.

2.1 Problema a resolver

El método de *regresión logística* asume que $Pr[y_i|x_i, \beta] \sim Bernoulli(\mu_i)$, con:

$$\mu_i = \sigma(\beta^T x_i)$$
$$\sigma(z) = (1 + exp(-z))^{-1}$$
$$\beta \in \mathbb{R}^p$$

El objetivo es encontrar el modelo $\hat{\beta} \in \mathbb{R}^p$ que mejor se ajuste al conjunto de datos. Para estimar $\hat{\beta}$, implementaremos y compararemos tres métodos: método de Newton, método Broyden, Fletcher, Goldfarb y Shanno(BFGS) y el método del descenso de gradiente estocástico (SGD, por sus siglas en inglés).

Dado lo anterior, se resuelve el problema de manera iterativa por medio de la resolución de los siguientes objetivos intermedios:

- Modelar $Pr[y|x, \hat{\beta}]$
- Predecir la etiqueta $\hat{y} \in 0, 1$ de un nuevo dato x por medio de:

$$\hat{y} = \begin{cases} 0, & \text{si } \sigma(\hat{\beta}^T x) \geq 0.5 \\ 1, & \text{si } \sigma(\hat{\beta}^T x) < 0.5 \end{cases} \tag{1}$$

- Computar la función de pérdida correspondiente a la *log-verosimilitud negativa* (y el riesgo empírico en SGD):

$$F(\beta) := LVN(\beta) = -\sum_{i=1}^{m} [y_i log\mu_i + (1 - y_i)log(1 - \mu_i)] \tag{2}$$

- Comparar tiempo de convergencia entre los tres métodos y las soluciones alanzadas en términos de las métricas de desempeño del modelo.

3 Implementación

La implementación de este ejercicio se realizará por medio de código escrito en lenguaje Python. En particular, con el fin de minimizar la función de pérdida de log-verosimilitud negativa (y el riesgo empírico en SGD), se incluirán módulos propios para resolver el problema. Los pasos a seguir son los siguientes:

1. Implementar *grad_F* y hess_F en Python.
2. Implementar el método de máximo descenso para minimizar $F(\beta)$. Elegimos un β^0 aleatorio y una tolerancia de $\epsilon = 10^{-8}$.
3. Implementar un clasificador de regresión logística para obtener $\hat{y}$.
4. Implementar el método de Newton para minimizar $F(\beta)$
5. Implementar el método BFGS
6. Implementar el método SGD
7. Implementar variabilidad en tasa de aprendizaje (condición de Armijo).
8. Paralelizar procesos; por ejemplo: resolver en paralelo la dirección del descenso.
9. Dockerizar ambiente
10. Comparar tiempos de ejecución
11. Unittest

Con el fin de generar un entorno aislado que permita evaluar el desempeño del proceso de entrenamiento en término de recursos computacionales, se utilizará una instancia EC2 de Amazon Web Services (AWS). Adicionalmente, para tener un control del entorno virtual asociado, se creará una imagen de docker asociada al repositorio donde se encontrarán todos los códigos de este proyecto.

4 Algoritmos

4.1 Método de Newton

1. Inicializar β^0
2. Mientras "no converge"
   (a) Resolver d^k en $(\nabla^2 F(\beta^k))d^k = -\nabla F(\beta^k)$
   (b) Utilizar Armijo para encontrar una tasa de aprendizaje α^k
   (c) $\beta^{k+1} \leftarrow \beta^k + \alpha^k d^k$

4.2 Método BFGS

1. Aproximar la inversa $\nabla^2 F(\beta^k)$ por medio de una función de rango-2 dada por:

$$\begin{aligned} H^{k+1} &= H^k + \frac{w^k (w^k)^T}{(w^k)^T z^k} - \frac{H^k z^k (H^k z^k)^T}{(z^k)^T H^k z^k} \\ z^k &= \beta^{k+1} - \beta^k \\ w^k &= \nabla F(\beta^(k+1) - \nabla F(\beta^k) \end{aligned} \tag{3}$$

2. Reescribir $F(\beta)$ como función de riesgo empírico.
3. Derivar $\nabla l(\beta^T x_i, y_i)$.

4.3 Método SGD

1. Inicializar β^0
2. Mientras "no converge"
   (a) Reordenar los datos D de manera aleatoria (para garantizar $\mathbb{E}[\nabla l(\beta^T x, y)] = \nabla F(\beta)$
   (b) Para cada $(x_i, y_i) \in D, \beta^{k+1} \leftarrow \beta^k - \eta \nabla l(\beta^T x_i, y_i)$

Implementar el método de gradiente estocástico con minilotes de cardinalidad $q \in [m]$:

$$\beta \leftarrow \beta - \eta \frac{1}{|S|} \sum_{j \in S} \nabla l(\beta^T x_i, y_i),$$

para minilote $S \sim S \subset D : |S| = q$

5 Resultados

6 Conclusiones

7 Referencias

- Colubri, A., Hartley, M. A., Siakor, M., Wolfman, V., Felix, A., Sesay, T., ... Sabeti, P. C. (2019). Machine-learning Prognostic Models from the 2014–16 Ebola Outbreak: Data-harmonization Challenges, Validation Strategies, and mHealth Applications. EClinicalMedicine, 11, 54-6º4.
- Nocedal, J., Wright, S. (2006). Numerical optimization. Springer Science Business Media.

Appendix A Funciones de Python

Appendix B Código de Python