Centro de Investigación en Matemáticas, A.C.

CIMAT

KSG-EDA:

Un Algoritmo de Estimación de Distribución que Minimiza la Divergencia Kullback-Leibler Usando SGD

T E S I S

Que para obtener el grado de
Maestro en Ciencias
con especialidad en

Computación y Matemáticas Industriales

P r e s e n t a:

Luisa Fernanda Restrepo Torres

Directores de tesis:

Dr. Arturo Hernández Aguirre

Dr. Ignacio Segovia-Domínguez

Autorización de la versión final

Guanajuato, Gto. 21, Octubre de 2020

A mi familia

Agradecimientos

Agradezco a mi asesor el Dr. Arturo Hernandez Aguirre quien con su experiencia, paciencia y calidad humana me acompañó, apoyó y guió durante el último año de la maestría. Fue él quien hizo que me enamorara de la Optimización Estocástica y que finalmente mi tesis de grado fuera una aportación en este campo.

Agradezco a mi coasesor el Dr. Ignacio Segovia que con su disciplina, paciencia, constante acompañamiento y sus fabulosas ideas hizo que esta tesis se llevara a cabo. Gracias por no dejar que perdiera el horizonte.

Agradezco a mis padres Mirian Torres y Alberto Restrepo, mis hermanas Claudia y Alejandra y mi sobrina Alexandra porque son mi razón y motivación para lograr cada cosa que me he propuesto. Gracias por siempre confiar en mi y apoyar cada decisión que tomo.

Agradezco a mi novio Breitner Ocampo por amarme y permitirme amarlo. Gracias por cada llamada que me sacó una sonrisa y me desconectó del estrés de los trabajos pendientes. Gracias por ser mi escape.

Agradezco a mi mejor amigo David Cardona, por su apoyo incondicional, por escucharme, jugar conmigo y sobre todo por hacerme sentir qué se siente tener un hermano. Agradezco a mis amigos de aventura: Santiago, Leimar y Richard por los almuerzos de cada domingo y las experiencias vividas. También agradezco a Erick, Juan Luis y Mario quienes me brindaron apoyo no solamente con temas académicos sino con temas personales. Gracias por compartir conmigo sus experiencias y enseñarme mucho más de la cultura mexicana.

Agradezco al Centro de Investigación en Matemáticas (CIMAT), por abrirme las puertas de su comunidad, por brindarme un buen ambiente de trabajo, una oficina agradable y profesores excepcionales. Agradezco también al Consejo Nacional de Ciencia y Tecnología (CONACYT), por el apoyo económico con una beca completa de maestría cuyo número de registro de becario es 940478.

Resumen

El objetivo principal de este trabajo de investigación es crear un Algoritmo de Estimación de Distribución que actualice los parámetros de la distribución de búsqueda usando Descenso de Gradiente Estocástico. Para lograr este objetivo se hace un recorrido teórico por los conceptos básicos de los Algoritmos Evolutivos, Funciones de Densidad de Probabilidad, Algoritmos de Estimación de Distribución, gradientes y varios tipos de Descenso de Gradiente. También se hace un desarrollo detallado de un par de algoritmos que inspiraron el diseño y creación del aquí propuesto. Adicionalmente se comparan estos tres algoritmos en varias dimensiones y con diversas funciones del benchamark.

El algoritmo creado en esta tesis llamado KSG-EDA un algoritmo novedoso, que bajo los diversos experimentos realizados se muestra muy competitivo. Lo más importante para resaltar de este algoritmo es que tiene bases matemáticas fuertes: utiliza Descenso de Gradiente estocástico para explorar el espacio de los parámetros de la distribución de búsqueda; propone una táctica de exploración e intensificación con la intensión de mejorar la aptitud del mejor individuo actual de la población y seguir explorando el espacio de búsqueda.

Palabras Clave

Optimización, Algoritmos de Evolutivos, función de distribución, Algoritmos de Estimación de Distribución, Gradiente, Descenso de Gradiente Estocástico, Divergencia Kullback-Leibler.

Índice

Índice de figuras

Índice de tablas

Introducción

El hombre, a lo largo de la historia, de manera consciente o inconsciente, se ha enfrentado a problemas de optimización difíciles de resolver, por ejemplo, encontrar la ruta de mínima longitud para llegar de un lugar a otro, maximizar la ganancia de un trabajo, minimizar costos de producción, etc. Debido a la importancia de la optimización, este ha sido un tema muy atractivo para la investigación durante los últimos años. Para un problema de optimización dado, una tarea desafiante es diseñar algoritmos que puedan encontrar la solución óptima o, por lo menos, encontrar una solución aceptable lo más rápido posible. Uno de los conceptos más utilizados y mejor estudiados en la optimización numérica es el gradiente; el gradiente en cualquier punto del espacio de búsqueda indica la dirección a lo largo de la cual la función a optimizar debe mejorar. Es por eso que es natural diseñar algoritmos que usen la dirección del gradiente para encontrar óptimos locales de la función moviendo iterativamente una solución usando la información del gradiente. Para optimizar una función, existen muchas variantes de este tipo de algoritmos, los cuales van desde los más sencillos como el Descenso de Gradiente [31] a unos más avanzados, como los Gradientes Conjugados [17].

Los gradientes siguen siendo un tema importante en la investigación en varias áreas de optimización como es, por ejemplo, el caso de la optimización estocástica; la optimización estocástica es un escenario que se encuentra típicamente cuando se trata de simulaciones de

procesos reales que son de alta complejidad [3]. Los Algoritmos Evolutivos (EAs) son una poderosa metodología de optimización estocástica y un área de investigación muy activa. Se han desarrollado múltiples algoritmos EAs hibridizados para abordar problemas del mundo real. Un EA se denomina híbrido si se integra en su procedimiento un algoritmo de optimización local, los EAs híbridos a menudo se denominan como Algoritmos meméticos [26] [10]. Dada esta premisa, el diseño de un EA híbrido que integre algoritmos de optimización basados en gradientes es un tema interesante. Sin embargo, aunque algunos problemas pueden beneficiarse del uso de información de gradiente, en otros problemas, es mucho más complicado calcular el gradiente que resolver el problema de optimización asociado a este gradiente, por lo tanto, buscar estrategias para estimar el gradiente es una alternativa que vale la pena explorar [35][36].

En este trabajo se hace una hibridización entre una familia de EAs llamada Algoritmos de Estimación de Distribución (EDAs) con Descenso de Gradiente Estocástico, el cual es una variante del algoritmo Descenso de Gradiente [33]. Los EDAs son Algoritmos Evolutivos que se basan en un modelo probabilístico llamada distribución de búsqueda [25][11]. A lo largo de este trabajo se muestra que los EDAs y gradientes pueden combinarse para crear diversos algoritmos de optimización estocástica que ajustan los parámetros de la distribución de búsqueda con la información de un gradiente.

Las funciones objetivo a optimizar en el presente trabajo tienen restricciones de caja, es decir, restricciones donde los valores del espacio de búsqueda están acotados inferior y superiormente. Este tipo de restricciones son muy comunes en los modelos reales de optimización, por ejemplo, la capacidad máxima y mínima de un inventario, o su producción y transporte, también en problemas de inversiones es natural tener una restricción de cantidad mínima y máxima de inversión; hasta para hablar de un ejemplo cotidiano como ir de compras, se habla de la cantidad máxima de peso que puede cargar una persona.

Esta tesis está organizada de la siguiente manera: la *Introducción* contiene la motivación, el planteamiento del problema y la estructura de la tesis. En el Capítulo 1, llamado *Preliminares*, se revisan los conceptos que se van a usar en la tesis: uno de ellos son los EDAs. junto

con el papel tan importante que juega la distribución de búsqueda en ellos; se expone la distribución Boltzmann y la distribución Normal; se expone la Divergencia Kullback-Leibler; se explican los conceptos básicos del gradiente y SGD, el cual es una técnica de optimización basada en un gradiente estocástico. También se explican y ejemplifican varias versiones del SGD. En el Capítulo 2, llamado *Algoritmos de Estimación de Distribución con Gradiente*, se exponen los algoritmos ∇_d-NEDA y BUMDA. En este capítulo, estos algoritmos son desarrollados, explicados y complementados con demostraciones matemáticas que no aparecen en la literatura. En el Capítulo 3, llamado *KSG-EDA*, se construye un algoritmo de optimización estocástica continua (KSG-EDA) el cual usa SGD para minimizar la Divergencia Kullback-Leibler de la distribución Bolztmann a la Normal. En el Capítulo 4, llamado *Experimentos*, se documentan los resultados obtenidos al ejecutar el algoritmo KSG-EDA con funciones unimodales y multimodales del Benchmark. Estos resultados se comparan con los obtenidos por los algoritmos ∇_d-NEDA y BUMDA con ayuda de dos test estadísticos. Finalmente, en el Capítulo 5 llamado *Conclusiones*, se hacen algunos comentarios comparativos entre los algoritmos involucrados en este trabajo; también se muestran algunos caminos para mejorar el algoritmo propuesto en un futuro trabajo. Como parte final de la tesis anexamos el Apéndice A en donde se muestra cómo graficar una matriz de covarianza para facilitar la visualización en dimensión 2 del comportamiento de los algoritmos presentes en la tesis.

CAPÍTULO 1

Preliminares

Este capítulo contiene los conceptos básicos necesarios para entender el contenido de este trabajo de investigación. Se explican las ideas generales de un Algoritmo de Estimación de Distribución (EDA por su siglas en inglés: Estimation Distribution Algorithm), se exponen las dos distribuciones de probabilidad protagonistas en este trabajo, se explica un criterio de similitud de distribuciones y con un ejemplo se muestra el papel clave que desempeña una distribución de probabilidad en los EDAs. Posteriormente se introduce el gradiente de una función, se describe sus características más importantes y cómo puede usarse esta información en algoritmos de optimización.

1.1. Conceptos básicos EDAs

Los Algoritmos Genéticos (GA: Por sus siglas en inglés: Genetic Algorithm) son heurísticas inspiradas en la evolución de las especies, esto es, los individuos más aptos son sobrevivientes de pequeñas mutaciones fortuitas que les permitió adaptarse a su medio por medio de la selección natural(Holland [18]). En un GA los individuos más aptos y más fuertes tienen

mayor probabilidad de reproducirse y heredar su material genético a las siguiente generación. Los GAs pertenecen a una familia denominada Algoritmos Evolutivos [1] [11](EA: Evolution Algorithm), cuyos miembros son: AG (Holland y Goldberg [14]), Programación Evolutiva (Lawrence Fogel), Estrategias Evolutivas (Schewefel [4]), y Programación Genética (John Koza [23]). La principal característica de un EA es la *población* la cual representa un conjunto de candidatos a ser solución. Las componentes de la población son denominados *individuos*. La estructura de los individuos depende del dominio del problema, lo cual implica que para cada problema, es necesario tener una representación adecuada. Los problemas de optimización deben tener una medida que permita comparar los individuos que compiten, es decir, debe de existir un valor escalar asociado a cada individuo de la población que represente su calidad como solución, a la función que asigna este valor se le denomina *función de aptitud*. Un individuo con mayor aptitud representa una mejor solución a un problema, de la misma manera, un individuo con menor aptitud representa una peor solución. Otra característica muy importante en los EA es la *selección* el cual es un operador que escoge (preferiblemente) a los individuos con mayor aptitud de una población. El grado en el cual se elije a los individuos por su valor de aptitud se denomina *presión de selección*. La *reproducción* es el operador que le permite a individuos de una población intercambiar información generando nuevas posibles soluciones, mientras que la *mutación* es el operador que causa cambios aleatorios en los individuos. El pseudocódigo del AE se muestra en el Algoritmo 1.

Algorithm 1 Estructura general de un EA

1: $t = 0$
2: Inicializar población $P^{(t)}$ aleatoria en el dominio de $\mathcal{F}$
3: **while** No se cumpla criterio de paro **do**
4: Evaluar aptitud
5: $P_1 \leftarrow$ Selección$(P^{(t)})$ ▷ Seleccionar padres
6: $P_2 \leftarrow$ Reproducción(P_1) ▷ Generar hijos
7: $P_3 \leftarrow$ Mutación(P_2) ▷ Mutar hijos
8: $P^{(t+1)} \leftarrow P_3$ ▷ Reemplazar los padres con los nuevos hijos
9: $t = t + 1$
10: **end while**

Para muchos problemas de optimización real, la mejor función de aptitud es la función misma que se pretende resolver. Pues la finalidad de una función de aptitud es medir el desempeño de los individuos y es parte fundamental para modelar la lucha por la sobrevivencia de los individuos de una especie.

A continuación se definen formalmente algunos conceptos mencionados anteriormente para un problema de optimización real y continuo. Los individuos de una población P se expresan como objetos en $\mathbb{R}^d$, es decir, un individuo $x \in P$ es de la forma $(x_1, x_2, \cdots, x_d)$. Se denotará como *función objetivo* una función $\mathcal{F} : D \subseteq \mathbb{R}^d \to \mathbb{R}$ que se pretende minimizar sobre un dominio o *espacio de búsqueda* $D \subseteq \mathbb{R}^d$. El espacio de búsqueda comúnmente se encuentra acotado por un hipercubo, es decir, $D \subseteq [x_{min}, x_{max}]^d$. esta característica del espacio de búsqueda es llamada *condición de caja*. Se dice que un individuo $x \in D$ es mejor (o más apto) que un individuo $y \in D$ si $\mathcal{F}(x) < \mathcal{F}(y)$. Una de las intenciones principales de los EAs es encontrar un individuo óptimo (u óptimos) $x^\star \in D$ tal que $\mathcal{F}(x^\star) \leq \mathcal{F}(x)\ \forall x \in D$. De manera análoga se aplica para maximizar la función $\mathcal{F}$.

Los GAs simulan la evolución natural de una población y tienen 3 fases importantes: selección, cruza y mutación. Este tipo de algoritmo elige a los mejores individuos de la población, con la intención de que al reproducirse se generen mejores soluciones que puedan formar parte de la siguiente generación. Dichas soluciones "hijas"son susceptibles a experimentar mutaciones y de esta manera tener mayor diversidad en la población. Cuando un individuo se muta y sobrevive se simula la adaptación de él a su entorno. El pseudocódigo del GA se muestra en el Algoritmo 2.

Los Algoritmos de Estimación de Distribución [25] (EDAs), también llamados PMBGAs [30] (Probabilistic Model-Building Genetic Algorithms) o IDEAs [7] (Iterated Density Estimation Algorithms) son una corriente moderna de los EAs, en particular de los GAs. Una de las principales diferencias entre los EDAs y lo GAs, es la forma en que se regenera la pobla-

Algorithm 2 Estructura general de un GA

1: Inicializar población P aleatoria en el dominio de $\mathcal{F}$ de tamaño N
2: **while** No se cumpla criterio de paro **do**
3: P_1 Seleccionar N individuos mediante el operador de selección
4: P_2 Aplicar el operador cruza a P_1 con probabilidad Pc para generar hijos
5: Mutar los dos hijos con probabilidad Pm
6: Los hijos generados se convierten en la nueva población P
7: **end while**

ción: los EDAs utilizan una distribución de probabilidad, llamada *distribución de búsqueda*, la cual usualmente se construye con los mejores individuos de la población, y se usa para muestrear nuevas soluciones candidatas y poblar la siguiente generación, estas nuevas soluciones adquieren del modelo de probabilidad las características de los mejores individuos de la población. Es decir, en los GAs existe de manera implícita una distribución de probabilidad de la población [16] pero el algoritmo la ignora, mientras que en un EDA (ver Algoritmo 3) la distribución de probabilidad se hace explícita (ver paso 5) y se debe construir (aproximar) a partir de una población seleccionada (paso 4).

Algorithm 3 Estructura general de un EDA

1: Inicializar población P aleatoria en el dominio de $\mathcal{F}$
2: **while** No se cumpla criterio de paro **do**
3: Evaluar la aptitud de los individuos de P
4: $P_1 \leftarrow$ Seleccionar individuos de P
5: Estimar los parámetros θ del modelo $\mathcal{Q}(x, \theta)$ a partir de P_1
6: $H \leftarrow$ Muestrear individuos de $\mathcal{Q}(x, \theta)$
7: $P \leftarrow H$
8: **end while**

La estrategia de los EDAs es incrementar la probabilidad de muestrear el óptimo a partir de la distribución de búsqueda. El problema principal en estos algoritmos es estimar los parámetros de la distribución de búsqueda. De esta manera los EDAs transforman el problema de optimización en un problema de estimación de los parámetros de una distribución de probabilidad (el modelo) que permite muestrear sobre el óptimo y regiones promisorias con probabilidad creciente a lo largo de las generaciones. La distribución Normal Multivariada

es usada comúnmente como distribución de búsqueda para los EDAs, ya que es fácil de implementar y permite modelar de manera simple características de la población como la dependencia entre variables. Sin embargo existen otras distribuciones como la Boltzman [38], mezclas de distribuciones [36], etc. que conforman la distribución búsqueda de un EDA. En [25] se encuentran una alta variedad de EDAs tanto para optimización continua como para discreta. En la Secciones 1.1.1 y 1.1.3 se muestran las dos distribuciones que usan los EDAs descritos en este trabajo de investigación.

1.1.1. Distribución Normal

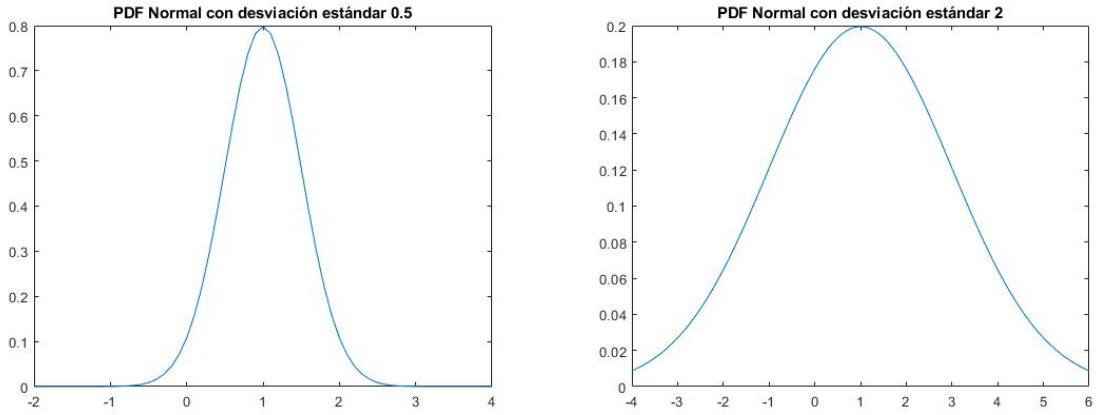
La distribución Normal [39] fue utilizada de manera empírica por primera vez por el francés Abraham de Moivre (1667-1754). Sin embargo, fue Carl Friedrich Gauss (1777-1855) quién formuló la ecuación de la curva y por eso se conoce también como "la campana de Gauss". La densidad de una variable normal está completamente determinada por dos parámetros, su media y su varianza, denotadas generalmente por μ y σ . Con esta notación, la densidad de la normal univariada

$$\mathcal{N}(x, \mu, \sigma) : \mathbb{R} \to \mathbb{R}$$

está descrita por la ecuación:

$$\mathcal{N}(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma}\right) \tag{1.1}$$

La densidad normal o Gaussiana tiene varias características atractivas: es simétrica con respecto a la media; es una densidad unimodal, es decir, tiene un único máximo ubicado justamente en μ. Otra propiedad interesante es que los valores que son más frecuentes o que tienen más probabilidad de aparecer están alrededor de la media, esto significa que cuando los valores observados se alejan de la media, la probabilidad de aparecer y su frecuencia descienden. Por su parte, la varianza es un parámetro que describe qué tan alejados de la media están las observaciones. En la figura 1.1 se muestran dos densidades normales con media



(a) Normal con $\mu = 1$ y $\sigma = 0.25$ **(b)** Normal con $\mu = 1$ y $\sigma = 4$

Figura 1.1: Funciones de densidad normal con media $\mu = 1$

igual media pero diferente varianza. Como se puede observar, al aumentar la varianza de 0.25 a 4, la ubicación de la gráfica no cambia, pero la forma de la gráfica sí: hay menos densidad en el centro y más densidad en las colas; también la probabilidad de la media es menor.

La densidad Normal también tiene su versión multivariada, es decir, su dominio puede ser $\mathbb{R}^d$ para $d \in \mathbb{N}$ cuya ecuación viene dada por la siguiente expresión:

$$N(x, \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) \tag{1.2}$$

El vector $\mu \in \mathbb{R}^d$ es la media y $\Sigma \in \mathbb{R}^{d\times d}$ es la matriz de covarianza, esta matriz es simétrica y definida positiva, siendo $|\Sigma|$ su determinante. Note que la ecuación 1.2 se reduce a la ecuación 1.1 si $d = 1$ (μ y Σ serían un escalar). Aunque ya no sea posible visualizar esta función de distribución para 3 o más dimensiones, la distribución normal multivariada sigue manteniendo las propiedades características de la normal univariada. Para ahondar más en este tema refiérase a [8], [19] y [22].

Para entender mejor el funcionamiento de un EDA y ejemplificar el papel que desarrolla una distribución búsqueda en ellos, considere el siguiente algoritmo llamado Algoritmo de

Estimación Normal Multivariada el cual utiliza una distribución Normal Multivariada como distribución de búsqueda.

1.1.2. Ejemplo de un EDA: Algoritmo de distribución Normal Multivariada (EMNA)

El Algoritmo de Estimación Normal Multivariada [37] (EMNA: Estimation Mutivariate Normal Algorithm) es un EDA que utiliza una distribución normal multivariada $\mathcal{N}(x, \mu, \Sigma)$ con media μ y varianza Σ para generar los nuevos individuos de la población. Ahora, la cuestión importante es cómo este algoritmo estima los parámetros μ y Σ. El método es bastante simple, el EMNA genera de forma aleatoria la población inicial; a una porción de los mejores individuos les calcula el promedio y la varianza de sus componentes, las expresiones obtenidas son los parámetros μ y Σ de la distribución normal de la que se muestrea la siguiente generación. La distribución generada de esta manera es comúnmente llamada Normal Empírica. El pseudocódigo del EMNA se muestra en el algoritmo 4.

Algorithm 4 EMNA

1: Inicializar población P aleatoria en el dominio de $\mathcal{F}$
2: **while** No se cumpla criterio de paro **do**
3: Evaluar la aptitud de los individuos de P
4: $PM \leftarrow$ los m mejores individuos de P
5: $\mu \leftarrow$ media de las componentes de los individuos de PM
6: $\Sigma \leftarrow$ matriz de covarianza asociada a PM
7: $P \leftarrow$ Muestrear individuos de $\mathcal{N}(\mu, \Sigma)$
8: **end while**

La capacidad de exploración de un EDA depende fuertemente de la diversidad de la población. Al seleccionar los mejores individuos (tener una presión de selección alta) para estimar los parámetros de la distribución, se puede perder diversidad y así generar una convergencia prematura del algoritmo la cual reduce su exploración y su habilidad para detectar las mejores regiones en el espacio de búsqueda. La convergencia anticipada de un EDA puede ocurrir porque las funciones de distribución creadas mediante parámetros estimados en cada

generación, junto con una alta presión de selección, tienden a presentar una menor desviación con respecto a su media; con una menor desviación, los individuos generados son más cercanos entre sí cada vez. Para aprovechar este comportamiento, en el Capítulo 3 se utiliza la metodología del EMNA, la cual genera la nueva población con la varianza empírica, con la intención de intensificar la búsqueda de un mejor individuo alrededor de un individuo que es 'suficientemente bueno' para finalmente mejorar la calidad del óptimo encontrado.

1.1.3. Distribución Boltzmann

En la práctica, los EDAs suponen que las mejores regiones para buscar el óptimo son aquellas que contienen a los mejores individuos. Por lo tanto, la distribución de búsqueda debe reflejar los valores de aptitud de los individuos, esto implica que cuanto mejor sea el valor de aptitud de un individuo, mayor debe ser la probabilidad de muestrear sobre la región alrededor de dicho individuo. Con la intensión de muestrear sobre las regiones más prometedoras de una función $\mathcal{F}$, es deseable que la función de densidad de probabilidad $p(x)$ cumpla con las siguientes características:

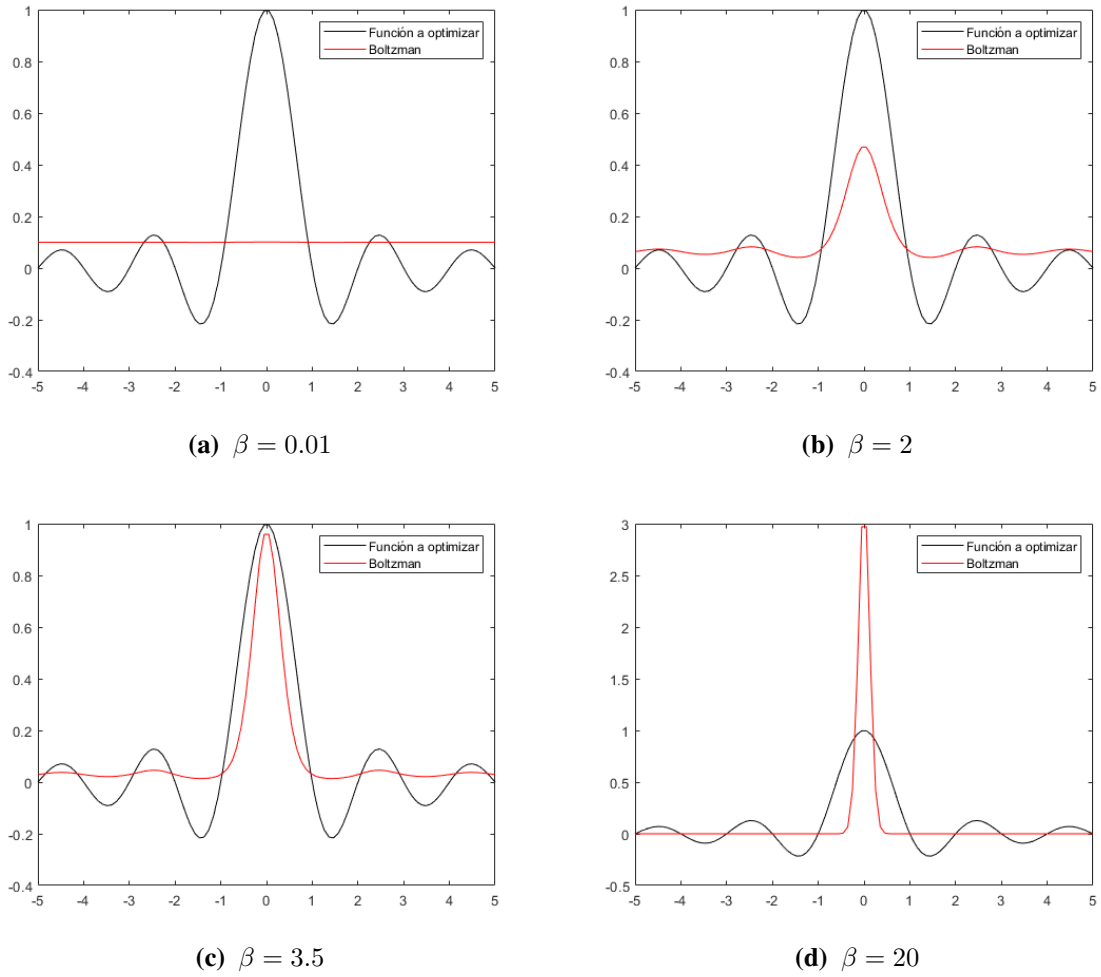
- Si el individuo x_1 es mejor que x_2 entonces $p(x_1) > p(x_2)$.
- Cuando $\mathcal{F}$ crece/decrece, la función de densidad de probabilidad crece/decrece.
- Si x^* es el único máximo entonces $p(x^*) > p(x)$ para todo $x \neq x^*$ en el espacio de búsqueda.

Una distribución que cumple con las condiciones anteriores es la distribución Boltzmann. La densidad Boltzmann [38], [27] de una función a optimizar $\mathcal{F}$ está definida por:

$$p(x) = \int_x \frac{e^{\beta \mathcal{F}(x)}}{Z} dx, \quad \text{con } Z = \int_{\mathbb{R}^d} e^{\beta \mathcal{F}(x)} dx \tag{1.3}$$

El parámetro β [29] está relacionado con la varianza de la población generada y la presión de selección: si β es lo suficientemente grande, el óptimo tiene probabilidad casi 1; si es muy

pequeño (pero positivo) la probabilidad es casi la misma para todos los individuos, es decir, la distribución Boltzmann se comporta como una distribución uniforme sobre el dominio de $\mathcal{F}$.



(a) $\beta = 0.01$ **(b)** $\beta = 2$

(c) $\beta = 3.5$ **(d)** $\beta = 20$

Figura 1.2: Densidades Boltzmann con diferentes valores de β.

La Figura 1.2 ilustra el comportamiento de la densidad Boltzmann para diferentes valores de β. En la Figura 1.2a se evidencia que para $\beta = 0.01$ la densidad Bolztmann se comporta como una densidad uniforme, la probabilidad de cada individuo es la misma. En la Figura 1.2b para $\beta = 2$ se muestra cómo la densidad Bolztmann se comporta de manera similar a la función a optimizar, el máximo de la función tiene una probabilidad mayor que cualquier

otro, sin embargo los máximos locales también son detectados por la densidad. En la Figura 1.2c para $\beta = 3.5$ se ve cómo los máximos locales empiezan a ser ignorados por la distribución y el máximo tiene una probabilidad significativamente mayor que los demás. Para $\beta = 20$ la distribución ubica su masa de probabilidad en una región muy pequeña alrededor del máximo global sin tener en cuenta que existen máximos locales.

La distribución Boltzmann en términos teóricos parece ser la distribución ideal para muestrear individuos óptimos, sin embargo no es posible muestrear de ella directamente, pues para esto, el parámetro Z que es el valor de normalización para que la expresión (1.3) sea una distribución (el valor de la integral sea 1), debe ser conocido y éste está determinado por todos los valores de la función $\mathcal{F}$ sobre el dominio. Además de conocer Z se debe saber la función de densidad. Existen algoritmos que permiten muestrear de distribuciones desconocidas, como el de metrópolis, pero su costo computacional es muy elevado . Es por esto que es interesante encontrar una distribución que se parezca a la distribución Boltzmann de la que sí sea posible muestrear. A continuación se expone una expresión que permita medir la similitud o diferencia de dos distribuciones.

1.1.4. Divergencia Kullback-Leibler

La divergencia Kullback- Leibler (KL) fue originalmente introducida por Solomon Kullback y Richard Leibler en 1951 [24]. Es una "medida"de la similitud entre dos funciones de distribución de probabilidad. Para las distribuciones $\mathcal{P}$ y $\mathcal{Q}$ de variable aleatoria continua, la divergencia KL de $\mathcal{P}$ a $\mathcal{Q}$ se define como la integral:

$$KL_{\mathcal{P},\mathcal{Q}} = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} \, \mathrm{d}x \tag{1.4}$$

donde p y q representan las densidades de $\mathcal{P}$ y $\mathcal{Q}$ respectivamente. Algunas propiedades importantes de esta divergencia son las siguientes:

1. Es siempre positiva.

2. Es cero si y sólo si $\mathcal{Q}$ y $\mathcal{P}$ son la misma distribución.

3. No es simétrica.

Dadas las propiedades 1. y 2. si se quiere aproximar una distribución $\mathcal{P}$ desconocida con una distribución $\mathcal{Q}(\theta)$, se busca el parámetro θ que minimiza la función

$$KL_{\mathcal{P},\mathcal{Q}(\theta)} = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x,\theta)} \, \mathrm{d}x \tag{1.5}$$

Es importante tener en cuenta que por la propiedad 3. la divergencia $KL_{\mathcal{P},\mathcal{Q}}$ no es una distancia entre $\mathcal{P}$ y $\mathcal{Q}$. A pesar de eso, es la expresión que se usa para cuantizar la divergencia de $\mathcal{P}$ a $\mathcal{Q}$ porque es capaz de capturar las similitudes entre las distribuciones y determina una función suave sin cambios bruscos. También la divergencia KL provee expresiones matemáticas más fáciles de usar que otras divergencias como la [6] o que incluso usar expresiones que sí representan una distancia entre las distribuciones como la *Hellinger distance* [5]

En la Sección 2.2 y en el Capítulo 3 se pretende encontrar una distribución Normal que se aproxime a la Boltzmann en el sentido de la divergencia KL.

1.2. Gradiente

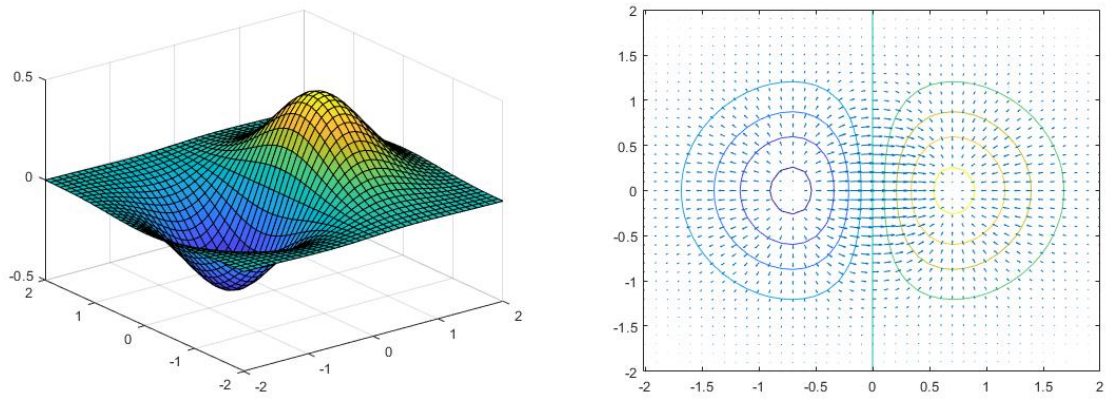
El gradiente de una función diferenciable

$$\mathcal{F} : D \subseteq \mathbb{R}^d \to \mathbb{R}$$

se denota como $\nabla\mathcal{F}$ y es el vector cuyas entradas son las derivadas parciales de $\mathcal{F}$

$$\nabla\mathcal{F}(x_1, x_2, ..., x_d) = \begin{bmatrix} \frac{\partial\mathcal{F}(x_1,x_2,...,x_d)}{\partial x_1} \\ \frac{\partial\mathcal{F}(x_1,x_2,...,x_d)}{\partial x_2} \\ \vdots \\ \frac{\partial\mathcal{F}(x_1,x_2,...,x_d)}{\partial x_d} \end{bmatrix}$$

Esto significa que $\nabla\mathcal{F}$ es una función vectorial. Para ejemplificar el campo vectorial $\nabla\mathcal{F}$, considere la función $\mathcal{F}(x, y) = xe^{-x^2-y^2}$



(a) Superficie

(b) Curvas de nivel y campo vectorial

Figura 1.3: (a) Superficie, (b) curvas de nivel y campo vectorial de $\mathcal{F}(x, y)$

La figura 1.3b muestra el campo vectorial $\nabla(xe^{-x^2-y^2})$, pero ¿cuál es su interpretación?

El campo vectorial está definido de la siguiente manera:

$$\nabla\mathcal{F}(x, y) = \begin{bmatrix} \frac{\partial\mathcal{F}(x,y)}{\partial x} \\ \frac{\partial\mathcal{F}(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} e^{-x^2-y^2}(1 - 2x^2) \\ -2xye^{-x^2-y^2} \end{bmatrix}$$

Considere a la gráfica de $xe^{-x^2-y^2}$ como si fuera una zona montañosa, con altos y bajos, si una persona quiere caminar por este terreno y está parada en el punto (x_0, y_0, z_0) (con $z_0 = \mathcal{F}(x_0, y_0)$) la pendiente de la colina depende de la dirección en la que camine, por ejemplo, si camina en dirección positiva del eje x la pendiente es $\dfrac{\partial\mathcal{F}(x_0, y_0)}{\partial x}$ o si lo hace

en dirección positiva al eje y la pendiente es $\dfrac{\partial \mathcal{F}(x_0, y_0)}{\partial y}$. Existen dos zonas interesantes en este terreno, la cima de la montaña y el fondo de la depresión; pero, ¿qué dirección debe tomar alguien para llegar más rápido a alguna de ellas? La respuesta la tiene el gradiente: si la persona camina en dirección del gradiente estará dirigiéndose directamente hacia la cima de la montaña, pero si camina en dirección opuesta a él llegará justamente al fondo de la depresión. La longitud del vector $\nabla \mathcal{F}(x_0, y_0)$ expresa cuál es la pendiente de la montaña en esa dirección, cuando la magnitud de $\nabla \mathcal{F}(x, y)$ es muy pequeña, como pasa en la cima o en el fondo entonces se está atravesando por un lugar prácticamente plano, estas regiones son atractivas pues ahí se encuentran puntos que pueden ser máximos, mínimos o puntos silla de la función $\mathcal{F}$. Los puntos anteriores se encuentran resolviendo el siguiente sistema de ecuaciones

$$\nabla \mathcal{F}(x, y) = \begin{bmatrix} e^{-x^2-y^2}(1 - 2x^2) \\ -2xye^{-x^2-y^2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{1.6}$$

Los puntos que satisfacen la ecuación 1.6 son $\left(\dfrac{1}{\sqrt{2}}, 0\right)$ y $\left(\dfrac{-1}{\sqrt{2}}, 0\right)$, los cuales corresponden al máximo y mínimo de la función $\mathcal{F}$, respectivamente, y se pueden observar en la figura 1.3a. Es importante mencionar que el gradiente de una función en un punto modela la taza local de cambio, es decir, dado un punto x en el espacio, se puede hablar del comportamiento de la función, con ayuda del gradiente, en una vecindad pequeña de ese punto pero se desconoce el comportamiento de la función en lugares alejados de este punto, es decir, no puede asegurar nada de manera global.

En la figura 1.3b se observan las curvas de nivel de la función $\mathcal{F}(x, y) = xe^{-x^2-y^2}$, cada vector $\nabla \mathcal{F}$ que toca a una curva de nivel apunta la en dirección en la que se llegará a la siguiente curva de nivel con el menor tamaño de paso. Además, entre más acercamiento se le hace la figura 1.3b, cada curva se parecerá más a una recta y las curvas de nivel parecerán paralelas; como se sabe que el camino más corto entre dos rectas paralelas es una línea recta perpendicular a ambas, el gradiente se ve (y es) perpendicular a la curva de nivel.

Cuando las entradas de una función $\mathcal{F}$ pertenecen a un espacio de más de dos dimensiones, ya no es fácil visualizar su gráfica ni el campo vectorial gradiente asociado a ella. Sin embargo, la misma idea se mantiene: el gradiente de la función $\mathcal{F}$ es el vector que apunta en la dirección que $\mathcal{F}$ crece más rápido.

Ahora bien, fue relativamente sencillo resolver la Ecuación 1.6 para encontrar el máximo y el mínimo de $\mathcal{F}$ porque la expresión matemática de la función fue dada y además esta función era diferenciable. Un tema interesante es cómo aproximar el gradiente de una función de la que no se conoce su expresión matemática, una función que se comporta como una caja negra: solo se sabe sus valores de entrada y de salida pero no su proceso. En la práctica se presentan frecuentemente este tipo de funciones, por eso, en el siguiente capítulo se muestran diversos métodos que aproximan el gradiente de una función.

1.2.1. Descenso de gradiente

Hasta ahora se ha visto en qué dirección moverse sobre una función $\mathcal{F}$ para llegar a puntos máximos o mínimos, pero no se sabe cuánto. Una estrategia es la siguiente: si la función cambia "mucho" entonces avanzar mucho, y si cambia "poco" avanzar poco conservando la dirección. En conclusión, lo más indicado es generar un paso proporcional al gradiente. Para poder controlar el tamaño del paso a dar se utiliza un parámetro cuantificador conocido como tasa de aprendizaje y usualmente denotado por α :

$$\alpha\nabla\mathcal{F} \quad \text{para ascender y} \quad -\alpha\nabla\mathcal{F} \quad \text{para descender}$$

Dicho esto, dada una posición x_k, para calcular nueva posición x_{k+1} en la cual se espera que $\mathcal{F}$ tome un menor valor, se usa la siguiente expresión:

$$x_{k+1} = x_k - \alpha\nabla\mathcal{F}(x_k) \tag{1.7}$$

En la Figura 1.4 se puede observar el recorrido iterativo de las nuevas posiciones de x_k para α fijo. Se nota que cada paso es más grande al principio que al final porque el gradiente es mayor; a medida que se aproxima a la zona de valor mínimo los pasos son más pequeños.

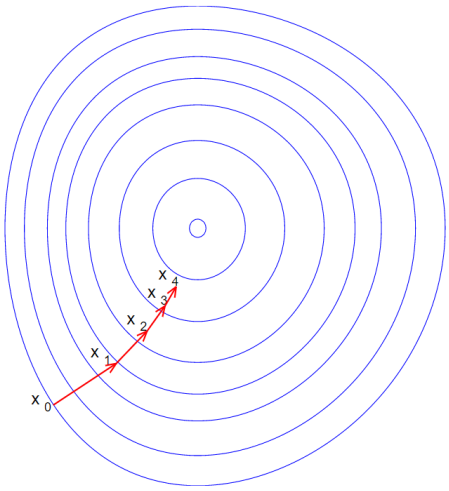


Figura 1.4: Recorrido iterativo de un punto a través de las curvas de nivel de una función

Elegir α adecuada puede ser difícil y depende del problema: una tasa de aprendizaje que es demasiado pequeña conduce a una convergencia muy lenta, mientras que una tasa de aprendizaje que es demasiado grande puede dificultar la convergencia y hacer que la función objetivo fluctúe alrededor de un mínimo o incluso diverja.

Lo métodos que usan la Ecuación (1.7) son intuitivos y fáciles de entender pero aunque se avance en dirección contraria al gradiente, esta estrategia no garantiza que se llegue al mínimo global o ni siquiera a un mínimo local.

La técnica anterior es conocida como *Descenso de Gradiente* (GD: Gradient Descent,Cauchy [9] 1847). El GD es uno de los algoritmos más populares para optimizar, y por mucho, es la forma más común de optimizar redes neuronales. La mayoría de las bibliotecas de Deep Learning de última generación contienen varias implementaciones de GD por ejemplo Lasagne, Caffe y Keras en Python.

En resumen, GD es una estrategia para optimizar una función objetivo $\mathcal{F}$ parametrizada por un conjunto de vectores $x \in \mathbb{R}^d$ los cuales actualiza en dirección del gradiente $\nabla\mathcal{F}$ (o en dirección contraria). También, usa una tasa de aprendizaje α que determina el tamaño de paso que se da en la dirección del gradiente la cual puede ser fija o variable. Es importante notar que este método depende fuertemente del punto inicial x_0.

Hasta este momento se explicaron los conceptos necesarios para desarrollar la técnica que se usa en este trabajo de investigación: Descenso de Gradiente Estocástico

1.2.2. Descenso de gradiente estocástico

El método Descenso de Gradiente Estocástico (SGD por sus siglas en inglés: Stochastic Gradient Descent) es precedido por el método GD y fue propuesto por Robbins y Monro en 1951 [33], aunque en este documento no aparece como un método de gradiente sino como una cadena de Markov. Posteriormente, Kiefer y Wolfowitz publicaron su artículo, *Stochastic Estimation of the Maximum of a Regression Function* [20], el cual es más reconocido para las personas familiarizadas con la variante ML de aproximación estocástica, es decir, con SGD. Los documentos anteriores son ampliamente citados como precursores del SGD en el artículo de revisión de Nocedal, Bottou y Curtis [41]. En el artículo anterior se proporciona una breve perspectiva histórica de SGD desde el punto de vista del Aprendizaje Automático.

El objetivo principal de SGD es encontrar los parámetros de un modelo que minimice una función de error usando solamente una muestra de toda la información disponible. La palabra ‘estocástico‘ significa un proceso que está vinculado a una probabilidad aleatoria. Por lo tanto, en el SGD, se seleccionan algunas muestras al azar en lugar del conjunto de datos completo en cada iteración. Para entender esto, considere el siguiente ejemplo:

Ejemplo: Ajuste de datos a una recta con SGD

Si se tiene un conjunto de datos en $\mathbb{R}^2$ compuestos por el peso y la altura de un conjunto de personas, se pretende encontrar una recta que minimice el error cuadrático entre dicha recta y los puntos. Es decir, se quiere construir un modelo, en este caso una recta $\mathcal{L}(x, m, b)$ cuyos parámetros son la pendiente m y el intercepto b, que minimice el error cuadrático entre la recta y los datos. La función de error se denota por $L(m, b)$ y depende de la pendiente y el intercepto.

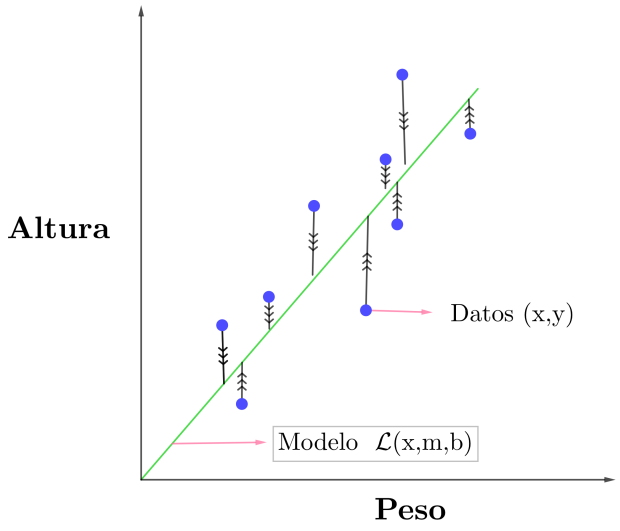


Figura 1.5: Recta que minimiza el error cuadrático

$$\text{Modelo} \rightarrow \mathcal{L}(x, m, b) = mx + b$$

$$\text{Función de error} \rightarrow L(m, b) = \sum_{i=1}^{n} (y_i - \mathcal{L}(x_i, m, b))^2$$

En este caso se considerará el error cuadrático como función de error pero puede ser cualquier otro: el error cuadrático medio, la suma del valor absoluto entre los valores reales y los predecidos, etc.

Para actualizar los parámetros de la recta, el típico GD cambia los parámetros derivando

a la función de error con respecto a los parámetros de la recta de la siguiente manera: Dados m_0, b_0 y los tamaño de paso α_{b0} y α_{m1}

$$m_{k+1} = m_k - \alpha_m \frac{\partial \mathrm{L}(m_k, b_k)}{\partial m} = m_k - \alpha_{mk} \sum_i -2x_i(y_i - (m_k x_i + b_k)),$$

$$b_{k+1} = b_k - \alpha_b \frac{\partial \mathrm{L}(m_k, b_k)}{\partial b} = b_k - \alpha_{bk} \sum_i -2(y_i - (m_k x_i + b_k)).$$

Este proceso se repite hasta que las derivadas parciales sean muy pequeñas (una tolerancia permitida por el usuario) o se cumplan un máximo de iteraciones. Note que las expresiones anteriores usan la totalidad de los datos en cada iteración, es por eso que si se tienen muchos datos, calcular cada derivada parcial puede ser muy costoso computacionalmente.

La idea del SGD es aproximar los gradientes de la función de error usando un subconjunto pequeño de datos que son elegidos de manera aleatoria en cada iteración. Esto permite hacer menos evaluaciones del modelo y en general se han obtenido resultados competentes comparados con el mismo GD. Esta técnica es muy atractiva en muchísimas áreas que usan optimización estocástica y por ser tan simple, rápida y efectiva se utiliza para el entrenamiento de redes neuronales.

El ejemplo anterior da pie a la siguiente sección en la que se hace un desarrollo teórico y formal de SGD.

Planteamiento formal de SGD

En primera instancia considere el siguiente planteamiento de aprendizaje supervisado. Sea $z = (x, y)$ una muestra determinada por un valor de entrada x y un valor de salida y. Sea $\mathrm{L}(\hat{y}, y)$ una función de pérdida que mide el costo de predecir $\hat{y}$ cuando el valor real es y. Sea $\mathbb{F}$ una familia de funciones $f_\theta(x)$ parametrizadas por un vector θ, el propósito es encontrar una

función $f \in \mathbb{F}$ que minimice el costo promedio de $\mathrm{L}(f_\theta(x), y) = \mathrm{L}(z, \theta)$ sobre un conjunto de muestras z. Si se supone que este promedio se toma sobre una densidad de probabilidad $dP(z)$ desconocida, el promedio de $\mathrm{L}(z, \theta)$ se escribe como:

$$\mathbb{E}[\mathrm{L}(z, \theta)] = \mathbb{E}[\mathrm{L}(f_\theta(x), y)] = \int \mathrm{L}(f_\theta(x), y) dP(z)$$

Como no se conoce la expresión para dP, la ley de los grandes números brinda una estimación para $\mathbb{E}(\mathrm{L}(f(x), y))$ usando una muestra $Z = \{z_1, z_2, \cdots, z_N\}$ y su expresión es la siguiente:

$$\mathbb{E}_N[\mathrm{L}(f_\theta(x), y)] = \frac{1}{N} \sum_{i=1}^{N} \mathrm{L}(f_\theta(x_i), y_i)$$

$\mathbb{E}_N$ es el riesgo empírico y mide el comportamiento de la muestra Z. $\mathbb{E}$ es el riesgo esperado y mide el comportamiento de manera general, eso quiere decir que es el comportamiento esperado de futuras muestras.

Existen múltiples métodos para minimizar el riesgo empírico $\mathbb{E}_N[f_\theta]$, uno de ellos es GD, que como se vio en la Sección 1.2.1, es un algoritmo de optimización iterativo en el que en cada iteración se actualiza el vector de parámetros θ usando el gradiente de $\mathbb{E}_N[f_\theta]$ con respecto a θ. Una fórmula de actualización sencilla del parámetro θ es la siguiente: para una $\theta^{(0)}$ dada,

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \mathrm{L}(z_i, \theta^{(t)}) \tag{1.8}$$

Note que para cada actualización del parámetro θ, la Fórmula (1.8) utiliza la totalidad la muestra Z lo que significa evaluar el gradiente de la función de costo L para todas las muestras. Esto puede ser un proceso de alto costo computacional y es aquí donde se motiva la formulación del método descenso de gradiente estocástico.

El SGD es un algoritmo que simplifica GD drásticamente. En lugar de calcular exactamente el gradiente de $\mathbb{E}_n(f_\theta)$ en cada iteración, se hace estimación de él usando una única

una muestra z_t, la cual se elige de manera aleatoria. Así, la Ecuación (1.8) queda de la forma:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \nabla_\theta \mathrm{L}(z_t, \theta^{(t)}) \tag{1.9}$$

El proceso estocástico $\{\theta^{(t)}, t = 1...\}$ depende de las muestras z_t tomadas de manera aleatoria, lo que se espera es que la Ecuación (1.9) se comporte como la Ecuación (1.8). Sin embargo, en la práctica es más usual usar un subconjunto aleatorio de muestras en lugar de una sola. Es decir, dado un conjunto de muestras $Z = \{z_1, z_2, \cdots, z_N\}$ se toman de manera aleatoria $n << N$ elementos de Z de tal manera que la actualización de los parámetros es:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathrm{L}(z_i, \theta^{(t)}) \tag{1.10}$$

La estructura general del SGD presentada utiliza una tasa de aprendizaje general, es decir, es la misma para todo el conjunto de parámetros θ. En general, dado que cada parámetro tiene un ritmo particular de descenso, a continuación se exponen tres algoritmos que modifican la tasa de aprendizaje α para cada componente de θ.

- **AdaGrad (Adaptive Gradient Algorithm)**

  AdaGrad [13] es un algoritmo de optimización tipo SGD el cual adapta la tasa de aprendizaje de los parámetros, realizando actualizaciones pequeñas (es decir, un α pequeña) para los parámetros asociados con las características que ocurren con frecuencia, y actualizaciones más grandes (es decir, un α grande) para los parámetros asociados con las características poco frecuentes. Por esta razón, es muy adecuado para tratar con datos escasos. Dean y Col. [12] descubrieron que AdaGrad mejoró enormemente la robustez del SGD y lo usaron para entrenar redes neuronales a gran escala en Google.

  Anteriormente, se mostró el SGD básico (Ecuación (1.9)) que actualiza todos los parámetros $\theta \in \mathbb{R}^d$ a la vez, es decir, cada parámetro $\theta_i, i = 1, 2, \cdots, d$ usa la misma tasa de aprendizaje α. El AdaGrad usa una tasa de aprendizaje diferente para cada parámetro θ_i en cada iteración t, es por eso que primero se va mostrar la actualización por compo-

nente y luego se va a generalizar vectorizando. Por simplicidad, se va a usar $g_i^{(t)}$ para denotar el gradiente de L en la iteración t con respecto al parámetro θ_i

SGD actualiza cada parámetro θ_i en cada iteración t de la forma:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha \cdot g_i^{(t)}$$

El AdaGrad modifica esta tasa general α con la siguiente expresión:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \frac{\alpha}{\sqrt{G_{ii}^{(t)} + \epsilon}} \cdot g_i^{(t)}$$

Donde $G^{(t)} \in \mathbb{R}^{d \times d}$ es una matriz diagonal donde cada elemento de la diagonal $G_{i,i}^{(t)}$ es la suma de los cuadrados de los gradientes con respecto a θ_i anteriores. El término ϵ impide la división por cero (usualmente es 10^{-8}). Sin la raíz en el denominador, el algoritmo se comporta inestable. Como $G^{(t)}$ contiene en la diagonal la suma de los cuadrados de los anteriores gradientes con respecto a todos los parámetros θ, lo anterior se puede vectorizar cambiando el producto de números por el producto matriz-vector de la siguiente manera:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\alpha}{\sqrt{G^{(t)} + \epsilon}} \cdot g^{(t)}$$

Una de los principales fortalezas del algoritmo AdaGrad es la adaptación del tamaño de paso de acuerdo a las características de las variables y no lo deja simplemente en un valor fijo. Por el contrario, la principal debilidad es la acumulación de los cuadrados de los gradientes en el denominador: como cada término añadido es positivo, la suma acumulada crece durante las iteraciones. Esto causa que la tasa de aprendizaje se vuelva, eventualmente, infinitesimalmente pequeña y desde ese momento el algoritmo se estanca.

- **AdaDelta**

El algoritmo AdaDelta [42] intenta resolver la mayor debilidad de AdaGrad reduciendo de manera monótona el tamaño de paso. En lugar de acumular todos los gradientes cuadrados pasados, AdaDelta restringe el número de gradientes pasados acumulados a un tamaño fijo k. En lugar de almacenar de manera ineficiente k gradientes cuadrados anteriores, la suma de los gradientes se define de forma recursiva como un promedio de todos los gradientes cuadrados anteriores. El promedio $\mathbb{E}[g^2]^{(t)}$ depende del anterior promedio y del gradiente actual:

$$\mathbb{E}[g^2]^{(t)} = 0.9\mathbb{E}[g^2]^{(t-1)} + 0.1g^{2(t)}$$

Así, la fórmula de actualización de parámetros de forma vectorizada es:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\alpha}{\sqrt{\mathbb{E}[g^2]^{(t)} + \epsilon}} \cdot g^{(t)}$$

- **Adam (Adaptive Moment Estimation)**

  El Adam [21] es otro algoritmo de optimización tipo SGD que adapta tamaños de paso en caso iteración por parámetro usando estimaciones de la media y la varianza del gradiente. El Adam combina las ventajas de los algoritmos AdaGrad y RMSProp. Las fórmulas de actualización para la media y la varianza del gradiente son las siguientes:

$$m^{(t)} = \beta_1 m^{(t-1)} + (1 - \beta_1)g^{(t)}$$

$$v^{(t)} = \beta_2 v^{(t-1)} + (1 - \beta_2)g^{2(t)}$$

  $m^{(t)}$ y $v^{(t)}$ son las estimaciones del primer momento (la media) y del segundo momento (la varianza) de los gradientes en el tiempo t, de ahí el nombre del algoritmo. Para $m^{(0)}$ y $v^{(0)}$ se proponen los vectores 0. Los autores del Adam notaron que durante las primeras iteraciones o cuando β_1 y β_2 estaban muy cercanos a 1, m y v estaban muy cercanos al vector 0. Para contrarrestar este comportamiento sesgado, calculan las

siguientes expresiones:

$$\hat{m}^{(t)} = \frac{m^{(t)}}{1-\beta_1} \quad \hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2} \tag{1.11}$$

Luego, la fórmula de actualización de los parámetros es

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\alpha}{\sqrt{\hat{v}^{(t)}} + \epsilon} \cdot \hat{m}^{(t)}$$

Los autores proponen $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ y $\epsilon = 10^{-8}$

Ejemplo: ajuste de parámetros usando GD, SGD, AdaDelta y Adam

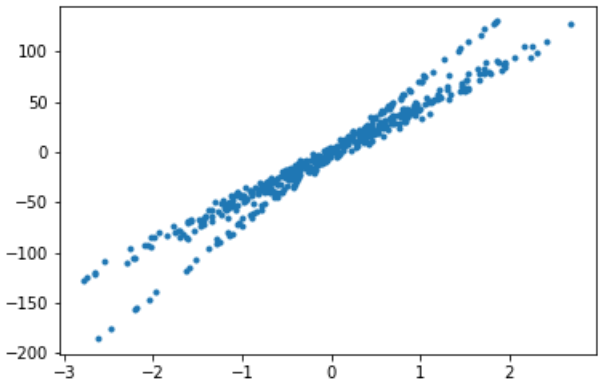


Figura 1.6: Datos $z_i = (x_i, y_i)$ para ajustar una recta

Para ejemplificar el GD, SGD, AdaDelta y Adam, considere los datos $Z = \{z_1, z_2, \cdots, z_N\}$ de la Figura 1.6 a los cuales se les pretende ajustar una recta (ejemplo basado en lo desarrollado [32]). Sea $\theta = [\theta_0, \theta_1]$ el vector de parámetros de la familia de rectas

$$f_\theta(x) = \theta_1 x + \theta_0 \tag{1.12}$$

Sea

$$\mathrm{L}(Z,\theta) = \frac{1}{2\kappa N}\sum_{i=1}^{N} 1 - \exp(-\kappa[f_\theta(x_i) - y_i]^2) \tag{1.13}$$

La función de costo a minimizar. Reemplazando la Ecuación (1.12) en la Ecuación (1.13) se obtiene

$$\mathrm{L}(Z,\theta) = \frac{1}{2\kappa N}\sum_{i=1}^{N} 1 - \exp(-\kappa[\theta_1 x_i + \theta_0 - y_i]^2) \tag{1.14}$$

Las derivadas parciales de (1.14) son

$$\frac{\partial \mathrm{L}(Z,\theta)}{\partial \theta_0} = -\frac{1}{N}\sum_{i=1}^{N}[\theta_1 x_i + \theta_0 - y_i]\exp(-\kappa[\theta_1 x_i + \theta_0 - y_i]^2)$$

$$\frac{\partial \mathrm{L}(z,\theta)}{\partial \theta_1} = -\frac{1}{N}\sum_{i=1}^{N} x_i[\theta_1 x_i + \theta_0 - y_i]\exp(-\kappa[\theta_1 x_i + \theta_0 - y_i]^2)$$

donde $\nabla_\theta \mathrm{L}(z,\theta) = \left(\frac{\partial \mathrm{L}(z,\theta)}{\partial \theta_0}, \frac{\partial \mathrm{L}(z,\theta)}{\partial \theta_1}\right)$

Después de ejecutar el GD, SGD, AdaDelta y Adam se obtiene la Figura 1.7 la cual muestra el recorrido del parámetro θ a lo largo de las iteraciones.

Los 4 algoritmos tienen la misma condición inicial para los parámetros

$$\theta^{(0)} = [\theta_0^{(0)}, \theta_1^{(0)}] = [0.546, 0.128]$$

El algoritmo Adam muestra cambios bruscos al inicio de las itereaciones pero se estabiliza alrededor de la iteración 25. El AdaDelta da unos pasos más lentos, llega a la misma región que el Adam, pero se queda oscilando hasta el final de las iteraciones. Los Algoritmos GD y SGD muestran un recorrido muy similar, aunque el SGD muestra un poco más de ruido en su trayectoria; también se observa que obtienen una convergencia tardía con respecto al Adam y AdaGrad.

El Adam y AdaDelta bajo circunstancias 'buenas', dan resultados muy parecidos, en [21] muestran que las fórmulas en 1.11 ayudan al Adam a superar ligeramente a AdaDelta en el final de la optimización a medida que los gradientes se vuelven más sparse. El Adam podría

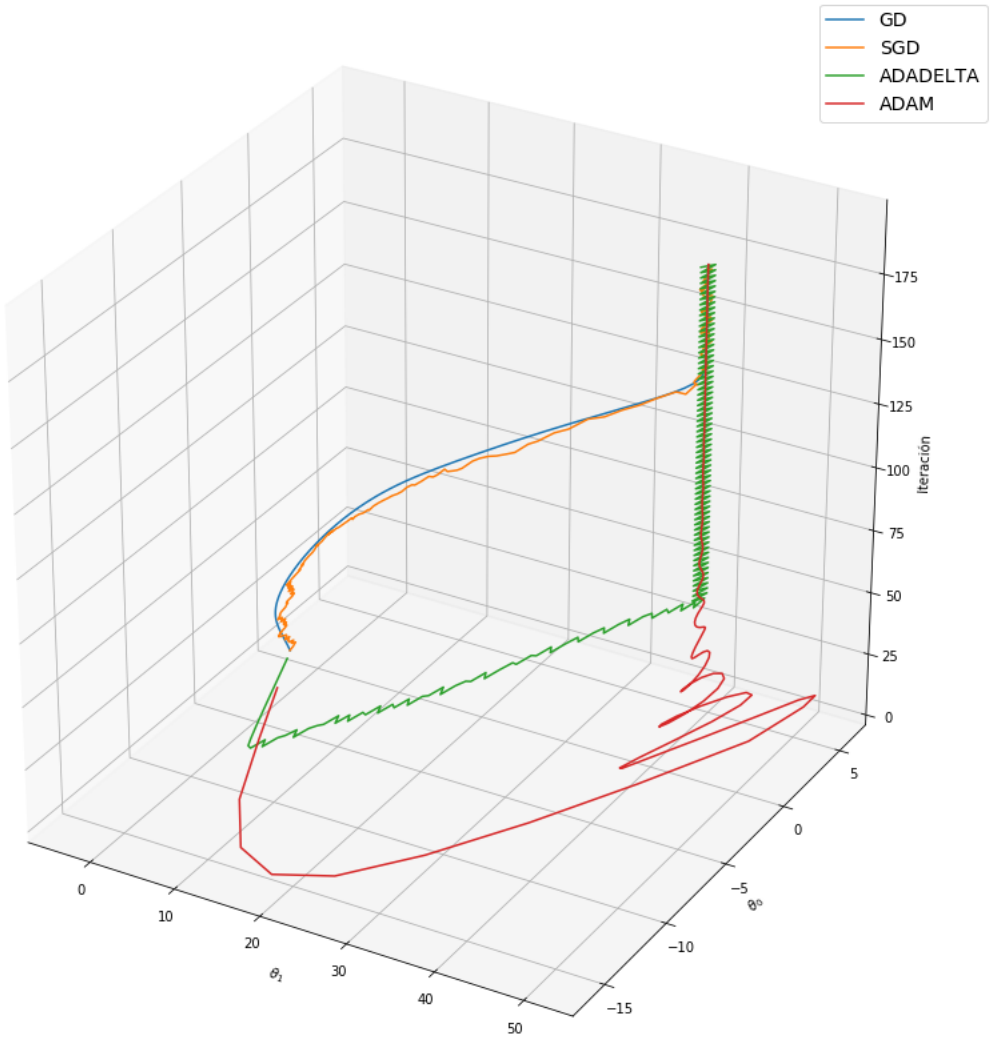


Figura 1.7: Comportamiento del GD, SGD, AdaDelta y Adam

Algoritmo	$f_\theta(x)$
GD	$45.68x - 0.28$
SGD	$44.56x + 0.99$
AdaDelta	$45.32x + 0.07$
Adam	$45.67x - 0.27$

Tabla 1.1: Recta que minimiza la función de costo (1.13) según cada algoritmo

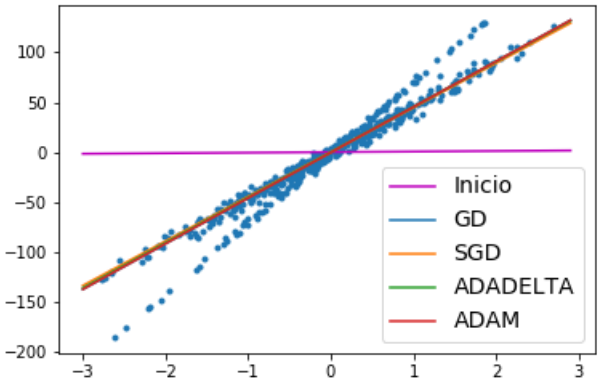


Figura 1.8: Rectas ajustadas a los datos usando GD, SGD, AdaDelta y Adam

ser una mejor opción en general. Para conocer otros tipos de SGD explicados de manera detallada, refiérase a [34].

1.3. Resumen

Este capítulo se presentaron los dos conceptos principales que se usan en este trabajo de investigación: EDAs y gradientes. Se explicó que los EDAs cambian el problema de optimización de una función $\mathcal{F}$ por un problema de ajuste de parámetros de una distribución de probabilidad que muestreará con mayor probabilidad el óptimo de $\mathcal{F}$. Se mostró el papel decisivo que juega una distribución de probabilidad en los EDAs y expuso la distribución Boltzmann y la Normal. También se explicó que el vector gradiente $\nabla\mathcal{F}(x)$ en un vector x modela la tasa local de mayor crecimiento, la cual tiene asociada una magnitud y una dirección en x, y cómo diversos algoritmos basados en descenso de gradiente, como SGD, usan esta información para actualizar parámetros de una función a optimizar. Posteriormente se verá cómo los EDAs y SGD se unen para crear el algoritmo propuesto en este trabajo de investigación.

CAPÍTULO 2

Algoritmos de Estimación de Distribución con Gradiente

En este capítulo se presentan dos EDAs que usan un gradiente estimado. Uno de ellos utiliza un gradiente estimado de la función a optimizar mientras que el otro calcula un gradiente con respecto a los parámetros de distribución de búsqueda que usa. Se desarrollan estos algoritmos de manera detallada pues por su alto contenido matemático, son la inspiración para crear el algoritmo que se presenta en el Capítulo 3.

2.1. ∇_d-NEDA

Es ∇_d-NEDA un EDA desarrollado por Segovia y Hernández [36] en el cual se construye una distribución de búsqueda basada en un gradiente estimado sobre el espacio de búsqueda de la función a optimizar. Para empezar a describir paso por paso la deducción del ∇_d-NEDA, se mostrará otra manera de estimar un gradiente sobre el espacio de búsqueda.

2.1.1. Estimación del gradiente

Como se vio en el capítulo anterior, bajo condiciones ideales el gradiente de una función objetivo tiene una propiedad importante: el vector gradiente en cualquier punto del espacio de búsqueda, siempre apunta en la dirección del máximo aumento de la función. Ahora bien, si la función objetivo no es diferenciable o si la expresión matemática de la función es desconocida, lo cual ocurre con frecuencia en aplicaciones reales, no es posible calcular de manera explícita un vector que indique una dirección de ascenso o de descenso (como lo hace el gradiente), es por eso que se tiene que recurrir a otras tácticas para tener una aproximación del gradiente de la función.

Se han desarrollado diferentes métodos para estimar el gradiente de una función objetivo. Por ejemplo, Salomon en el artículo [35] para aproximar el gradiente de una función $\mathcal{F}$ en un individuo x, genera n individuos $t_i = x + z_i$ aplicando mutaciones al z_i a x que se distribuyen con una distribución Gaussiana con media 0 y una varianza σ_i. La expresión que usa para calcular el gradiente estimado g_x llamado EGS (Evolutionay Gradient Search) es:

$$g_x = \sum_{i=1}^{n} (\mathcal{F}(t_i) - \mathcal{F}(x))(t_i - x) \tag{2.1}$$

Este método tiene varias propiedades importantes, una de ellas es que es invariante a rotaciones porque los individuos t_i son generados sin una orientación preferida. Sin embargo, está fuertemente influenciado por el número n de individuos que se generan. Las ideas presentadas por Salomon se enfocan en usar esta estimación de gradiente para aplicar el método de descenso de gradiente, como consecuencia, esta manera de optimizar depende de manera estricta del punto inicial x.

Por su parte, Segovia y Hernández construyen otra estimación del gradiente llamado (EGE: Expected Gradient Estimate). En su artículo [36], proponen un gradiente estimado que únicamente usa la información que siempre se tiene en los Algoritmos Evolutivos: los

individuos de la población y su evaluación en la función objetivo, sin necesidad de generar nuevos individuos cada vez que se necesite calcular un gradiente.

La propuesta de los autores de [36] sugiere que para calcular el gradiente estimado para un individuo x, se requiere de él mismo y de una vecindad $N_x = \{x_1, x_2, \cdots, x_r\}$ de tamaño r donde individuos de esta vecindad pueden ser elegidos de diferentes maneras, una de ellas es considerando la distancia euclidiana, es decir, los r individuos de la población más cercanos a x. Con lo anterior se puede iniciar con la deducción para el gradiente estimado de $\mathcal{F}$ en el punto x.

Suponga que para $x_k \in N_x$ se cumple la siguiente relación:

$$\frac{x_k - x}{\|x_k - x\|} = \pm \frac{\nabla \mathcal{F}(x)}{\|\nabla \mathcal{F}(x)\|}$$

Es decir, se supone que el punto x_k está sobre la línea definida por el gradiente real de $\mathcal{F}$ en x. El signo $+$ indica que x_k está en la misma dirección que $\nabla \mathcal{F}(x)$ y el signo - en dirección contraria.

Sean $u_+ = \frac{\nabla \mathcal{F}(x)}{\|\nabla \mathcal{F}(x)\|}$ y $u_- = -\frac{\nabla \mathcal{F}(x)}{\|\nabla \mathcal{F}(x)\|}$ vectores unitarios. De la definición de derivada direccional se sabe que:

$$\begin{aligned} \lim_{h \to 0} \frac{\mathcal{F}(x + hu_+) - \mathcal{F}(x)}{h} &= \nabla \mathcal{F}(x) \cdot u_+ \\ &= \|\nabla \mathcal{F}(x)\| \|u_+\| \cos(\alpha) \\ &= \|\nabla \mathcal{F}(x)\| \end{aligned}$$

como $\|u_+\| = 1$, u_+ es paralelo $\nabla \mathcal{F}(x)$ y están orientados en la misma dirección, entonces $\cos(\alpha) = 1$.

Así

$$\left(\lim_{h\to 0}\frac{\mathcal{F}(x+hu_+)-\mathcal{F}(x)}{h}\right)u_+ = \|\nabla\mathcal{F}(x)\|\frac{\nabla\mathcal{F}(x)}{\|\nabla\mathcal{F}(x)\|}$$
$$= \nabla\mathcal{F}(x)$$

De manera análoga, para u_-

$$\lim_{h\to 0}\frac{\mathcal{F}(x+hu_-)-\mathcal{F}(x)}{h} = \nabla\mathcal{F}(x)\cdot u_-$$
$$= \|\nabla\mathcal{F}(x)\|\|u_-\|\cos(\alpha)$$
$$= -\|\nabla\mathcal{F}(x)\|$$

pues $\|u_+\| = 1$, como u_+ es paralelo $\nabla\mathcal{F}(x)$ y están orientados en dirección contraria, $\cos(\alpha) = -1$.

$$\left(\lim_{h\to 0}\frac{\mathcal{F}(x+hu_-)-\mathcal{F}(x)}{h}\right)u_- = -\|\nabla\mathcal{F}(x)\|\frac{-\nabla\mathcal{F}(x)}{\|\nabla\mathcal{F}(x)\|}$$
$$= \nabla\mathcal{F}(x)$$

De cualquier forma, considerando u_- o u_+, se tiene que

$$\left(\lim_{h\to 0}\frac{\mathcal{F}(x+hu_\pm)-\mathcal{F}(x)}{h}\right)u_\pm = \|\nabla\mathcal{F}(x)\|\frac{\nabla\mathcal{F}(x)}{\|\nabla\mathcal{F}(x)\|}$$
$$= \nabla\mathcal{F}(x)$$

Desde este punto se puede considerar una aproximación del límite y así tener una estimación para $\nabla\mathcal{F}(x)$ la cual considera solo un individuo de N_x.

Como $u = \frac{x_k - x}{\|x_k - x\|} = \frac{\nabla \mathcal{F}(x)}{\|\nabla \mathcal{F}(x)\|}$, sea $l = \|x_k - x\|$, entonces la aproximación del gradiente g_k, que usa solo el vecino x_k de x tiene la forma

$$\begin{aligned} g_k &= \frac{\mathcal{F}(x + lu) - \mathcal{F}(x)}{l} u \\ &= \frac{\mathcal{F}(x + (x_k - x)) - \mathcal{F}(x)}{l} u \\ &= \frac{\mathcal{F}(x_k) - \mathcal{F}(x)}{l} u \\ &= \frac{\mathcal{F}(x_k) - \mathcal{F}(x)}{\|x_k - x\|} \frac{x_k - x}{\|x_k - x\|} \\ &= \frac{\mathcal{F}(x_k) - \mathcal{F}(x)}{\|x_k - x\|^2} (x_k - x) \end{aligned}$$

Es importante notar que se hizo una suposición muy fuerte sobre el individuo x_k de N_x y no hay manera de asegurar que este individuo esté sobre la linea que define el gradiente de $\mathcal{F}$ en x. Sin embargo, como ya se tiene una expresión para el gradiente usando solo un individuo de la vecindad N_x, es natural preguntarse por una expresión que contenga todos los individuos de ella y el desarrollo es el siguiente: los elementos en N_x vienen de un proceso aleatorio, y así, la diferencia entre los valores $\mathcal{F}(x_k)$ y $\mathcal{F}(x)$ también, luego cada estimación g_k surge de un proceso aleatorio. Si se supone que $\mathcal{P}$ es la incertidumbre del modelo que describe el comportamiento de g_k, un vector representativo para este modelo es la esperanza la cual se puede aproximar usando r elementos de N_x como sigue:

$$\mathbf{E}(g) = \int_{\mathcal{R}^d} g \mathcal{P} dg \approx \frac{1}{r} \sum_{k=1}^{r} g_k$$

Lo anterior motiva la siguiente definición:

Definición 1 (EGE). Sea $N_x = \{x_1, x_2, \cdots, x_r\}$ el conjunto de vecinos de x en la población,

el *Gradiente Estimado Esperado* de $\mathcal{F}$ en x está definido por

$$\hat{\nabla\mathcal{F}}(x) = \frac{1}{r}\sum_{k=1}^{r}\frac{\mathcal{F}(x_k) - \mathcal{F}(x)}{\|x_k - x\|^2}(x_k - x) \tag{2.2}$$

donde $x \neq x_k$ para todo $k = 1, 2, \cdots, r$ y $\mathcal{F}$ calcula el valor de la función objetivo.

Ahora bien, tendiendo en cuenta que se puede aproximar el gradiente de varias formas, el paso a seguir es ver cómo se usa esto en un algoritmo de optimización estocástica. Como se dijo anteriormente, el EGS descrito en la primera parte se usa en un algoritmo evolutivo con descenso de gradiente, sin embargo, el gradiente no solo puede usarse de esta manera, en la siguiente sección se describe el ∇_d-EDA un EDA que usa la aproximación de gradiente EGE de la Ecuación (2.2).

2.1.2. Densidad dirigida por gradiente estimado

Ahora se describirá de manera detallada el algoritmo ∇_d-NEDA que se desarrolla en [36], el cual es un algoritmo de optimización estocástica que tiene como objetivo construir una función de densidad que contiene información del gradiente estimado de la función a optimizar; la intención es incrementar la probabilidad de muestrear el óptimo. En la siguiente definición se describen dos propiedades deseables de tal densidad.

Definición 2. Sea z un individuo en el dominio de $\mathcal{F}$ y sea $G(z)$ la función que calcula el gradiente estimado en z usando la Ecuación (2.2). La función de densidad $\mathcal{P}(x, \theta)$ es una densidad controlada por el gradiente para un individuo z si cumple las siguientes condiciones:

1. La densidad multivariada $\mathcal{P}(x, \theta)$ es una función unimodal.
2. $\frac{G(z)}{\|z\|} = \frac{\nabla\mathcal{P}(z,\theta)}{\|\mathcal{P}(z,\theta)\|}$.

La primera condición dice que es deseable que exista solo una masa de probabilidad para

muestrear el óptimo; la segunda pide que la densidad tenga la misma dirección del gradiente, es decir, se quiere el ángulo que forma el vector $G(z)$ y $\nabla\mathcal{P}(z,\theta)$ sea cero, para esto considere el producto punto entre ellos

$$G(z)^T\nabla\mathcal{P}(z,\theta) = \|G(z)\|\|\nabla\mathcal{P}(z,\theta)\|cos(\alpha)$$

donde α es el ángulo entre los vectores. Si se necesita que $G(z)$ y $\nabla\mathcal{P}(z,\theta)$ estén en la misma dirección, se deben encontrar los parámetros $\hat{\theta}$ que minimicen el ángulo α, lo cual es equivalente a maximizar el cociente

$$\frac{G(z)^T\nabla\mathcal{P}(z,\theta)}{\|G(z)^T\|\|\nabla\mathcal{P}(z,\theta)\|}$$

Así las cosas, lo que se quiere encontrar es $\hat{\theta}$ tal que cumpla lo siguiente

$$\hat{\theta} = \max_{\theta} \frac{G(z)^T\nabla\mathcal{P}(z,\theta)}{\|G(z)^T\|\|\nabla\mathcal{P}(z,\theta)\|} \tag{2.3}$$

Es claro que esto aplica para cualquier densidad multivariada en $\mathbb{R}^d$, por simplicidad, se considera una densidad Gaussiana $\mathcal{N} = \mathcal{N}(x,\mu,\Sigma)$. Sea

$$\mathcal{P}(x,\theta) = \mathcal{N}(x,\theta:(\mu,\Sigma)) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right)$$

cuyo gradiente está determinado por $\nabla\mathcal{N} = \mathcal{N}\Sigma^{-1}(\mu - x)$.

Si se denomina $Z_G = \dfrac{G(z)}{\|G(z)\|} = (Z_{G_1}, Z_{G_2}, \cdots, Z_{G_d})$, La Ecuación (2.3) es de la siguiente

forma

$$\begin{aligned}\hat{\mu}, \hat{\Sigma} &= \max_{\mu,\Sigma} \frac{G(z)^T}{\|G(z)\|} \frac{\nabla \mathcal{P}(z,\theta)}{\|\nabla \mathcal{P}(z,\theta)\|} \\ &= \max_{\mu,\Sigma} Z_G^T \frac{\mathcal{N}\Sigma^{-1}(\mu - z)}{\|\mathcal{N}\Sigma^{-1}(\mu - z)\|} \\ &= \max_{\mu,\Sigma} Z_G^T \frac{\Sigma^{-1}(\mu - z)}{\|\Sigma^{-1}(\mu - z)\|} \end{aligned} \tag{2.4}$$

Pues $\mathcal{N} > 0$. Sea $x = \dfrac{\Sigma^{-1}(\mu - z)}{\|\Sigma^{-1}(\mu - z)\|} = (x_1, x_2, \cdots, x_d)$. La norma de x y de Z_G es 1. Según lo anterior, se pretende maximizar el producto punto entre Z_G y x con respecto a x (pues al fijar z, Z_G está fijo) y ambos están en esfera unitaria (tienen norma 1).

Descrito lo anterior como un problema de optimización con restricciones, se quiere maximizar $f(x) = Z_G \cdot x$ sujeto a $g(x) = \|x\|^2 = 1$. Calculando el gradiente de f y g obtenemos $\nabla f(x) = Z_G, \nabla g(x) = 2x$. Usando la técnica de multiplicadores de Lagrange, tomando $\nabla f(x) = \lambda \nabla g(x)$ se tiene que $Z_G = 2\lambda x$, así $x = \dfrac{Z_G}{2\lambda}$, reemplazando en la restricción $g(x)$ se tiene que $\sum_i \left(\dfrac{Z_{G_i}}{2\lambda}\right)^2 = 1$, $\frac{1}{4\lambda^2} \sum_i Z_{G_i}^2 = 1$, por hipótesis $\sum_i Z_{G_i}^2 = 1$ luego $\frac{1}{4\lambda^2} = 1$, así $\lambda = \pm\frac{1}{2}$ como se necesita que x esté en la misma dirección que Z_G entonces $\lambda = \frac{1}{2}$. Así las cosas, el vector x que maximiza f sujeto a g es $x = Z_G$. Ahora bien, existen infinitos vectores $\Sigma^{-1}(\mu - z)$ tal que $\dfrac{\Sigma^{-1}(\mu - z)}{\|\Sigma^{-1}(\mu - z)\|} = Z_G$, sin embargo, si se supone que $\|\Sigma^{-1}(\mu - z)\| = 1$ se obtiene la siguiente relación:

$$\Sigma^{-1}(\mu - z) = Z_G,$$

es claro que también existen infinitas μ y Σ que cumplen la igualdad anterior. Pero dado que Σ debe ser una matriz de covarianza, es decir, simétrica y definida positiva, es conveniente fijarla cumpliendo con esas restricciones, de esta manera despejando μ se obtiene lo siguiente

$$\mu = z + \Sigma Z_G$$

Con lo anterior se motiva la siguiente definición.

Definición 3. (Densidad controlada por el gradiente) Sea Σ_0 una matriz de covarianza fija, la densidad Gaussiana $N(z, \mu, \Sigma)$ dirigida por el gradiente $G(z)$ tiene parámetros:

$$\mu = z + \Sigma_0 Z_G$$

Conociendo los parámetros de la densidad deseada, es preciso ahora describir el algoritmo que la utiliza.

EDA - Densidad controlada por el gradiente

Esta sección presenta un EDA, el ∇_dNEDA el cual usa la función de densidad descrita en la Definición 3, este modelo calcula la esperanza y la varianza de un modelo que combina dos gaussianas multivariadas : La densidad Normal empírica y la densidad controlada por el gradiente, la primera densidad promueve la intensificación de la búsqueda del óptimo, mientras que la segunda permite que las muestras se generen sobre posibles regiones prometedoras, es decir, permite la exploración.

Mezcla de modelos

Sea $N_1(\mu_1, \Sigma_1)$ la densidad normal empírica y $N_2(\mu_2, \Sigma_2)$ la densidad controlada por el gradiente . Sea $\mathbf{Z}$ una variable aletoria que se distrbuye como la mezcla de las normales N_1 y N_2, esto es, $\mathbf{Z} \sim N(\mu_{new}, \Sigma_{new})$ donde $E[\mathbf{Z}] = \mu_{new}$ es esperanza y $Cov[\mathbf{Z}] = \Sigma_{new}$ la covarianza de $\mathbf{Z}$ respectivamente. Las expresiones para μ_{new} y Σ_{new} son las siguientes:

$$\mu_{new} = p\mu_1 + (1-p)\mu_2 \tag{2.5}$$

$$\Sigma_{new} = p\Sigma_1 - (1-p)\Sigma_2 + p(\mu_1 - \mu_{new})(\mu_1 - \mu_{new})^T + (1-p)(\mu_2 - \mu_{new})(\mu_2 - \mu_{new})^T \tag{2.6}$$

con $p \in [0, 1]$, note que si $p = 0$ estamos en el caso de la distribución controlada por el

gradiente y si $p = 1$ con la normal empírica.

Un primer paso para justificar las ecuaciones de arriba es hacer una transformación de la expresión de Σ_{new} como sigue:

$$\begin{aligned}
\Sigma_{new} &= p\Sigma_1 - (1-p)\Sigma_2 + p(\mu_1 - \mu_{new})(\mu_1 - \mu_{new})^T + (1-p)(\mu_2 - \mu_{new})(\mu_2 - \mu_{new})^T \\
&= p\Sigma_1 - (1-p)\Sigma_2 + p(\mu_1 - p\mu_1 - (1-p)\mu_2)(\mu_1 - p\mu_1 - (1-p)\mu_2)^T \\
&\quad + (1-p)(\mu_2 - p\mu_1 - (1-p)\mu_2)(\mu_2 - p\mu_1 - (1-p)\mu_2)^T \\
&= p\Sigma_1 - (1-p)\Sigma_2 + p((1-p)\mu_1 - (1-p)\mu_2)((1-p)\mu_1 - (1-p)\mu_2)^T \\
&\quad + (1-p)(-p\mu_1 + p\mu_2)(-p\mu_1 + p\mu_2)^T \\
&= p\Sigma_1 - (1-p)\Sigma_2 + p((1-p)(\mu_1 - \mu_2))((1-p)(\mu_1 - \mu_2))^T \\
&\quad + (1-p)((-p)(\mu_1 - \mu_2))((-p)(\mu_1 - \mu_2))^T \\
&= p\Sigma_1 - (1-p)\Sigma_2 + p(1-p)^2(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \\
&\quad + (1-p)(-p)^2(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \\
&= p\Sigma_1 - (1-p)\Sigma_2 + p(1-p)(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T((1-p) + p)) \\
&= p\Sigma_1 - (1-p)\Sigma_2 + p(1-p)(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T
\end{aligned} \tag{2.7}$$

Con lo anterior se pueden justificar las expresiones (2.5) y (2.6) usando la ecuación para la esperanza y la covarianza total:

Sea

$$I = \begin{cases} 1 & \text{si la variable se distribuye } N_1(\mu_1, \Sigma_1) \\ 0 & \text{si la variable se distribuye } N_2(\mu_2, \Sigma_2) \end{cases}$$

$$E[\mathbf{Z}|I=1] = \mu_1, \quad E[\mathbf{Z}|I=0] = \mu_2$$
$$Cov[\mathbf{Z}|I=1] = \mathbf{\Sigma}_1, \quad Cov[\mathbf{Z}|I=0] = \mathbf{\Sigma}_2$$

Sea $p = P(I = 1)$

$$\begin{aligned} E[\mathbf{Z}] &= E[E[\mathbf{Z}|I]] \\ &= E[p\mu_1 + (1-p)\mu_2] \\ &= p\mu_1 + (1-p)\mu_2 \end{aligned}$$

Con lo que queda justificada la expresión para μ_{new}.

Por otro lado es cierto que

$$\begin{aligned} Cov[\mathbf{Z}|I=1] &= E[\mathbf{Z}\mathbf{Z}^T|I=1] - \mu_1\mu_1^T = \mathbf{\Sigma}_1 \\ Cov[\mathbf{Z}|I=0] &= E[\mathbf{Z}\mathbf{Z}^T|I=0] - \mu_2\mu_2^T = \mathbf{\Sigma}_2 \\ E[\mathbf{Z}\mathbf{Z}^T|I=1] &= \mathbf{\Sigma}_1 + \mu_1\mu_1^T \\ E[\mathbf{Z}\mathbf{Z}^T|I=0] &= \mathbf{\Sigma}_2 + \mu_2\mu_2^T \end{aligned}$$

Usando la expresión de arriba, la covarianza total de $\mathbf{Z}$ es:

$$\begin{aligned} Cov[\mathbf{Z}] &= E[\mathbf{Z}\mathbf{Z}^T] - E[\mathbf{Z}]E[\mathbf{Z}]^T \\ &= E[E[\mathbf{Z}\mathbf{Z}^T|I]] - (p\mu_1 + (1-p)\mu_2)(p\mu_1 + (1-p)\mu_2)^T \\ &= E[p(\mathbf{\Sigma}_1 + \mu_1\mu_1^T) + (1-p)(\mathbf{\Sigma}_2 + \mu_2\mu_2^T)] - (p\mu_1 + (1-p)\mu_2)(p\mu_1^T + (1-p)\mu_2^T) \\ &= p\mathbf{\Sigma}_1 + p\mu_1\mu_1^T + (1-p)\mathbf{\Sigma}_2 + (1-p)\mu_2\mu_2^T - (p\mu_1 + (1-p)\mu_2)(p\mu_1^T + (1-p)\mu_2^T) \\ &= p\mathbf{\Sigma}_1 + (1-p)\mathbf{\Sigma}_2 + p(1-p)(\mu_1\mu_1^T) - p(1-p)\mu_1^T\mu_2 - p(1-p)\mu_1\mu_2^T + (p-p^2)(\mu_2\mu_2^T) \\ &= p\mathbf{\Sigma}_1 + (1-p)\mathbf{\Sigma}_2 + p(1-p)\left[(\mu_1\mu_1^T) - \mu_1^T\mu_2 - \mu_1\mu_2^T\right] + p(1-p)\left[(\mu_2\mu_2^T)\right] \\ &= p\mathbf{\Sigma}_1 + (1-p)\mathbf{\Sigma}_2 + p(1-p)(\mu_1 - \mu_2)(\mu_1^T - \mu_2^T) \\ &= p\mathbf{\Sigma}_1 + (1-p)\mathbf{\Sigma}_2 + p(1-p)(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \end{aligned}$$

Lo cual coincide con la expresión para la covarianza de (2.7).

Ejemplo

Para ilustrar cómo funciona el ∇_dNEDA considere la esfera en dimensión 2 , es decir, si $x = (x_1, x_2)$ la función $f(x) = x_1^2 + x_2^2$. Se sabe que el mínimo global es $\hat{x} = (0, 0)$ Ahora, considere una población de tamaño $N = 11$ generada de manera aleatoria en el cuadrado $[2, 8] \times [2, 8]$

x_1	x_2	$f(x)$
2.9250	7.7385	68.4397
7.6140	6.9123	105.7522
6.3696	3.0549	49.9037
4.1622	3.1327	27.1382
2.0072	3.8985	19.2272
6.1977	5.7515	71.4916
5.2584	4.6342	49.1265
3.7246	5.0100	38.9720
6.5693	6.5744	86.3788
5.4563	6.4860	71.8395
5.8732	2.7393	41.9984

x_1	x_2	$f(x)$
2.0072	**3.8985**	19.2272
4.1622	3.1327	27.1382
3.7246	5.0100	38.9720
5.8732	2.7393	41.9984
5.2584	4.6342	49.1265
6.3696	3.0549	49.9037
2.9250	7.7385	68.4397
6.1977	5.7515	71.4916
5.4563	6.4860	71.8395
6.5693	6.5744	86.3788
7.6140	6.9123	105.7522

En el cuadro de la izquierda observamos la población generada y su evaluación en la función $f(x)$. El cuadro de la derecha está ordenado con respecto al valor de aptitud $f(x)$ de menor a mayor. Como la intención es encontrar el individuo x tal que $f(x)$ sea mínimo, el mejor individuo actual es el que tiene función de aptitud más pequeña y está en negrita, llamémoslo z = (2.0072 , 3.8985).

Cálculo del gradiente

Para calcular el gradiente en z vamos a usar un tamaño de vecindad $r = 3$, estos están en azul $N_z = \{(4.1622, 3.1327); (3.7246, 5.0100); (5.8732, 2.7393)\}$.

Usando la expresión (2.2), el gradiente estimado en z es de la forma

$$\nabla f(z) = \frac{1}{3}\left(\frac{27.1382 - 19.2272}{\sqrt{(4.1622 \text{ - } 2.0072)^2 + (3.1327 \text{ - } 3.8985)^2}}(4.162 \text{ - } 2.0072, 3.1327 \text{ - } 3.8985)\right.$$
$$+ \frac{38.9720 - 19.2272}{\sqrt{(3.7246 \text{ - } 2.0072)^2 + (5.0100 \text{ - } 3.8985)^2}}(3.7246 \text{ - } 2.0072, 5.0100 \text{ - } 3.8985)$$
$$\left. + \frac{41.9984 - 19.2272}{\sqrt{(5.8732 \text{ - } 2.0072)^2 + (2.7393 \text{ - } 3.8985)^2}}(5.8732 \text{ - } 2.0072, 2.7393 \text{ - } 3.8985)\right)$$

Después de simplificaciones, el vector $\nabla f(z) =$ (5.5889 , 0.8219). Como queremos una dirección de descenso, el vector que vamos a usar es $-\nabla f(z)$. El vector $Z_G = \frac{\nabla f(z)}{\|\nabla f(z)\|} =$ (0.9894 , 0.1455).

Cálculo de las distribuciones normales:

- La normal empírica $\mathcal{N}_{\text{est}}(\mu_{\text{est}}, \Sigma_{\text{est}})$ tiene como parámetro μ_{est} la media de las componentes de la población, es decir, calculamos el promedio de las x_1 y el promedio de las x_2 y ese es el vector μ_{est} = (5.1052 , 5.0848). el parámetro Σ_{est}. Para la Σ_{est} calculamos covarianza de las variables x_1 y x_2 obteniendo la matriz $\begin{pmatrix} 2.6466 & 0.3075 \\ 0.3075 & 2.7325 \end{pmatrix}$

- La normal dirigida por el gradiente $\mathcal{N}_g(\mu_{\text{g}}, \Sigma_{\text{g}})$ depende fuertemente de la matriz Σ_{g}, vamos a considerar la matriz $\Sigma_{\text{g}} = \begin{pmatrix} 2.6466 & 0 \\ 0 & 2.7325 \end{pmatrix}$, esta matriz diagonal tiene los valores de la diagonal de Σ_{est}, así las cosas $\mu_g = z - \Sigma_g Z_G =$ (-0.6112 , 3.5009).

  La matriz Σ_g puede ser cualquier matriz definida positiva: La identidad, La misma matriz Σ_{est}, entre muchas otras. Es conveniente que sea una matriz que dependa de la población: una población dispersa permite exploración y una población aglomerada, intensificación en la búsqueda del óptimo.

- La mezcla de las normales $\mathcal{N}_{\text{new}}(\mu_{\text{new}}, \Sigma_{\text{new}})$ Usando las fórmulas (2.5) y (2.6) tenemos

que $\mu_{\text{new}} = (2.2470, 4.2929)$ y $\Sigma_{\text{new}} = \begin{pmatrix} 10.8160 & 2.4173 \\ 2.4173 & 3.3597 \end{pmatrix}$ con $p = 0.5$

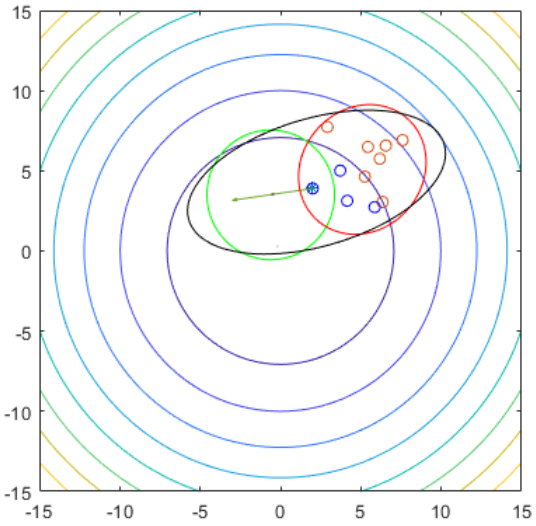


Figura 2.1: Ilustración en 2D de ∇_dEDA

Elipse roja: Normal empírica, Elipse verde: Normal controlada por le gradiente, Elipse negra: mezcla de las normales con parámetro 0.5. Vector verde: gradiente estimado en el mejor individuo cuya vecindad son los círculos pequeños azules dentro de la elipse roja. En la Figura 2.1 se observa el vector gradiente que justamente apunta a una región donde la función de aptitud va a mejorar.

Las elipses son una región de confianza del 95 %, es decir, que al muestrearse de la normal correspondiente, un 95 % de las muestras estarán en esa región, en el Apéndice A se muestra cómo se obtienen estas representaciones de las distribuciones normales.

Con todo lo anterior ya se puede formular el Algoritmo ∇_d-NEDA, ver Algoritmo 5 Sea $\beta = 1 - p$.

Algorithm 5 ∇_d-NEDA

1: P_t =U(dominio), calcule la aptitud de los individuos de P_t,P_{best} = los mejores $r+1$ individuos.
2: **while** No se cumpla criterio de paro **do**
3: Calcule el gradiente estimado $G(x_{best})$
4: Calcule μ, Σ empíricas
5: Tome Σ_g definida positiva
6: Calcule μ_{new} y Σ_{new} con las fórmulas (2.5) y (2.6)
7: $\mathcal{S} = \mathcal{N}(\mu_{new}, \Sigma_{new}, M)$
8: Calcule la aptitud de los individuos de $\mathcal{S}$
9: P_{t+1} = Mejores individuos de $\{P_t \cup S\}$
10: Si existe un mejor individuo, este reemplaza el peor de P_{best}
11: **if** $M_{sur}/M > 1/2$: **then**
12: $\beta = \beta + 0.05$
13: **else**
14: $\beta = \beta - 0.05$
15: **end if**
16: **end while**

Un dato curioso de este algoritmo es que la vecindad que usa para calcular el gradiente del mejor individuo es el historial del mismo, es decir, al reemplazar el mejor individuo actual por el peor en P_{best}, después de actualizar r veces al mejor individuo, tendremos el historial del mejor individuo como vecindad para el cálculo de su gradiente.
La exploración de este algoritmo depende fuertemente del parámetro β el cual se adapta de acuerdo a la calidad de los individuos muestreados: si más de la mitad de los individuos nuevos pasan a la siguiente generación, el algoritmo se permite explorar más en dirección al gradiente estimado, si por el contrario sobrevive menos de la mitad, se intensifica la búsqueda, reduciendo la exploración, con la intención de encontrar mejores individuos.

∇_d-NEDA es un algoritmo con un fundamento matemático fuerte en el que se tiene en cuenta la información proporcionada por gradiente de la función a optimizar, es comparable con algoritmos con muy buen desempeño tales como el CMA-ES(Evolution Strategy with Covariance Matrix Adaptation) [15] y el xNes [40]. En el Capítulo 4 se muestran los resultados de este algoritmo con algunos problemas del benchmark.

2.2. BUMDA

En esta sección se describe el BUMDA, un EDA que utiliza la información del gradiente calculado sobre el espacio de los parámetros de la distribución de búsqueda. El objetivo de este algoritmo es encontrar los parámetros μ, σ de una distribución Normal $\mathcal{N}(\mu, \sigma)$ que minimizan la divergencia KL 1.4 entre la Normal y la distribución Bolztmann 1.3.

Inicialmente se hará el cálculo matemático de los parámetros μ y σ que minimizan la divergencia KL.

2.2.1. Aproximación de la Distribucón Boltzmann con un Modelo Gaussiano usando Divergencia Kullback-Leibler

Recuerde que la divergencia Kullback- Liebler (KL) entre dos distribuciones $\mathcal{Q}$ y $\mathcal{P}$ está dada por la Ecuación

$$KL_{\mathcal{Q},\mathcal{P}} = \int_x q(x) \ln \frac{q(x)}{p(x)} \, \mathrm{d}x \tag{2.8}$$

Donde q, p son las densidades de $\mathcal{Q}$ y $\mathcal{P}$ respectivamente. Dicho esto, suponga inicialmente que la función objetivo es una función con dominio en los reales, es decir

$$\mathcal{F} : \mathbb{R} \to \mathbb{R}.$$

Lo que se quiere entonces es aproximar la distribución Boltzmann univariada definida en la Sección 1.1.3

$$p = p(x) = \frac{\exp(\beta g(x))}{Z}$$

con una Gaussiana univariada (Sección 1.1.1)

$$q = q(x) = N(x, \mu, \sigma) = \frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma}\right)}{\sqrt{2\pi\sigma}}$$

con parámetros media μ y varianza σ minimizando la expresión (2.8) y para ello se calculan las derivadas parciales de la misma.

Primero, es necesario encontrar una expresión general para la derivada parcial de $KL_{\mathcal{Q},\mathcal{P}}$ respecto a cualquier parámetro θ

$$\begin{aligned}\frac{\partial K_{Q,P}}{\partial\theta} &= \int_x \frac{\partial}{\partial\theta}\left(q\log\frac{q}{p}\right)dx \\ &= \int_x \frac{\partial q}{\partial\theta}\log\frac{q}{p} + q\frac{\partial}{\partial\theta}\left(\log\frac{q}{p}\right)dx \\ &= \int_x \frac{\partial q}{\partial\theta}\log\frac{q}{p} + \frac{\partial q}{\partial\theta}dx \\ &= \int_x \left(1+\log\frac{q}{p}\right)\frac{\partial q}{\partial\theta}dx \end{aligned} \tag{2.9}$$

Con lo anterior, para reemplazar θ por μ, se necesitan dos expresiones importantes

$$\begin{aligned}\frac{\partial q}{\partial\mu} &= q\left(\frac{x-\mu}{\sigma}\right), \\ \log\frac{q}{p} &= \log\left(\frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma}\right)}{\sqrt{2\pi\sigma}}\right) - \log\frac{\exp(\beta\mathcal{F}(x))}{Z} \\ &= -\frac{(x-\mu)^2}{2\sigma} - \log\sqrt{2\pi\sigma} - \beta\mathcal{F}(x) + \log Z\end{aligned}$$

Reemplazando en la integral (2.9) se obtiene:

$$\begin{aligned}\int_x \left(1+\log\frac{q}{p}\right)\frac{\partial q}{\partial\mu}dx = \\ \int_x (1+\log Z - \frac{(x-\mu)^2}{2\sigma} - \log\sqrt{2\pi\sigma} - \beta\mathcal{F}(x))q\left(\frac{x-\mu}{\sigma}\right)dx = \\ \int_x \left(1+\log Z - \frac{(x-\mu)^2}{2\sigma} - \log\sqrt{2\pi\sigma}\right)q\left(\frac{x-\mu}{\sigma}\right)dx \qquad (2.10) \\ -\int_x \beta\mathcal{F}(x)q\left(\frac{x-\mu}{\sigma}\right)dx \qquad (2.11)\end{aligned}$$

$(x-\mu)$ es una función impar con respecto a μ, se sabe que q y $\frac{(x-\mu)^2}{2\sigma}$ son funciones pares respecto a μ y $1+\log Z - \log\sqrt{2\pi\sigma}$ es una constante. El μ que se busca será la media de los

datos, es decir, el lugar donde se integra está centrado en μ, entonces la integral en (2.10) es 0.

Así las cosas, hasta ahora se tiene que

$$\frac{\partial K_{\mathcal{Q},\mathcal{P}}}{\partial \mu} = -\frac{\beta}{\sigma}\int_x q(x-\mu)\mathcal{F}(x)dx$$

Sabemos que

$$\int_x q(x-\mu)\mathcal{F}(x)dx = \mathbb{E}[(x-\mu)\mathcal{F}(x)]$$

Y por la ley de los grandes números, se tiene que

$$\int_x q(x-\mu)\mathcal{F}(x)dx \approx \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)\mathcal{F}(x_i)$$

Luego $\frac{\partial K_{\mathcal{Q},\mathcal{P}}}{\partial \mu} \approx \sum_{i=1}^{n}(x_i-\mu)\mathcal{F}(x_i)$, igualando a 0 y despejando μ se obtiene

$$\mu \approx \frac{\sum_{i=1}^{n}\mathcal{F}(x_i)x_i}{\sum_{i=1}^{n}\mathcal{F}(x_i)} \tag{2.12}$$

Finalmente, la expresión para σ se calcula de manera similar:

$$\frac{\partial q}{\partial \sigma} = q\left(\frac{(x-\sigma)^2}{2\sigma^2} - \frac{1}{2\sigma}\right) \tag{2.13}$$

Reemplazando en (2.9)

$$\begin{aligned}
\int_x \left(1+\log\frac{q}{p}\right)\frac{\partial q}{\partial \sigma}dx = \\
\int_x \left(1+\log\frac{q}{p}\right) q\left(\frac{(x-\sigma)^2}{2\sigma^2} - \frac{1}{2\sigma}\right)dx = \\
\int_x (1+\log Z - \log\sqrt{2\pi\sigma})q\left(\frac{(x-\sigma)^2}{2\sigma^2} - \frac{1}{2\sigma}\right)dx \\
-\int_x \frac{(x-\mu)^2}{2\sigma} + \beta\mathcal{F}(x)q\left(\frac{(x-\sigma)^2}{2\sigma^2} - \frac{1}{2\sigma}\right)dx
\end{aligned} \tag{2.14}$$

Usando que

$$\int_x q dx = 1$$

$$\int_x q(x-\mu)^2 dx = \sigma$$

$$\int_x (x-\mu)^4 q dx = 3\sigma^2$$

se obtiene que (2.14) es igual a:

$$-\frac{3}{4\sigma} - \frac{\beta}{2\sigma^2}\int_x \mathcal{F}(x) q(x-\mu)^2 dx + \frac{1}{4\sigma} + \frac{\beta}{2\sigma}\int_x \mathcal{F}(x) q dx$$

Igualando a 0, despejando σ y usando la ley de los grandes números se tiene que :

$$\begin{aligned} \sigma &= \frac{\int_x \mathcal{F}(x)(x-\mu)^2 q dx}{\frac{3+4\sqrt{2}}{8\sqrt{2}\beta} + \int_x \mathcal{F}(x) q dx} \\ &\approx \frac{\sum_i \mathcal{F}(x_i)(x_i-\mu)^2}{T + \sum_i \mathcal{F}(x_i)} \end{aligned} \tag{2.15}$$

donde $T = \frac{3+4\sqrt{2}}{8\sqrt{2}\beta}$

Por cuestiones de simplicidad, en [38] trabajan con una expresión para σ como la siguiente:

$$\sigma \approx \frac{\sum_i \mathcal{F}(x_i)(x_i-\mu)^2}{1 + \sum_i \mathcal{F}(x_i)} \tag{2.16}$$

Análisis de las expresiones para los parámetros encontrados

Ya teniendo las expresiones de los parámetros del modelo Gaussiano que minimiza (1.4) observaciones acerca de las expresiones encontradas para ellos.

- Como $sup_x \mathcal{F}(x) = -inf_x(-\mathcal{F}(x))$, la expresión para la distribución Boltzmann $p(x) = \frac{\exp(\beta\mathcal{F}(x))}{Z}$ quedaría entonces $p(x) = \frac{\exp(\beta(-\mathcal{F}(x)))}{Z} = \frac{\exp(-\beta\mathcal{F}(x))}{Z}$ y como las expresiones en (2.5) y (26) no dependen de β, el análisis para encontrar los parámetros μ y σ es el

mismo en los dos casos.

- Como la expresión de μ depende en el numerador y en el denominador de la función g, si estamos en un problema minimización donde el óptimo es 0 se puede obtener a tener un problema numérico del tipo $\frac{0}{0}$ así que lo prudente es hacer a transformaciones algebraicas de la función para evitar este problema. Por ejemplo considere la función $\mathcal{F}(x) = x^2$, $\mathcal{F}$ es no negativa y su óptimo es el 0 por lo cual bastará con encontrar el mínimo de la función $\mathcal{F}^*(x) = x^2 + c$ donde $c > 0$, la cual tiene el mismo mínimo de la función $\mathcal{F}$.

- Existe una restricción para el parámetro σ que no es tenida en cuenta, σ por definición tiene que ser mayor o igual que cero, en este caso, la expresión en (26) depende de la función $\mathcal{F}$ y de los valores de su dominio. Para el caso donde ésta es una función positiva y tenemos un dominio positivo, no hay problema, pero cuando toma algunos valores negativos y otros positivos o tiene dominio con números positivos y negativos, se puede correr el riesgo de tener una varianza negativa.

Después de comentar algunas adversidades que puede presentar el método que se base en expresiones para μ y σ presentadas en (2.12) y (26) respectivamente, se presentará el algoritmo BUMDA.

2.2.2. EDA-Boltzman

Como es común en los EDAs es necesario contar con una parte de la población para actualizar los parámetros de la distribución de búsqueda y así generar nuevos individuos. El BUMBA utiliza un método de truncamiento para selección de ciertos individuos de la población: Considere una población inicial $x_1, x_2, ..., x_N \in \mathbb{R}$

Método de truncamiento maximizar funciones

- Para una generación inicial $t = 0$ sea $\mathcal{F}(x_i, 0)$ $i = 1, ..., N$, los valores de la función objetivo en la población inicial. Defina $\phi_0 = \min \mathcal{F}(x_i, 0)$

- Para $t > 0$ sea:$\phi_t = \max(\phi_{t-1}, \min(\mathcal{F}(x_i, t)|\mathcal{F}(x_i, t) > \phi_{t-1}))$
- Si para los individuos ordenados de manera decreciente se tiene que $\mathcal{F}(x_{N/2}) \geq \theta_t$ se define $\phi_t = \mathcal{F}(x_{N/2})$
- Se trunca la población de manera tal que $\mathcal{F}(x_s, t) \geq \phi_t$ donde x_s son todos los individuos para los cuales la función objetivo es mayor o igual que ϕ_t

Método de truncamiento minimizar funciones

- Para una generación inicial $t = 0$ sea $\mathcal{F}(x_i, 0)$ $i = 1, ..., N$, los valores de la función objetivo en la población inicial. defina $\phi_0 = \max \mathcal{F}(x_i, 0)$
- Para $t > 0$ sea: $\phi_t = \min(\phi_{t-1}, \max(\mathcal{F}(x_i, t)|g(x_i, t) > \phi_{t-1})$
- Si para los individuos ordenados de manera creciente se tiene que $\mathcal{F}(x_{N/2}) \leq \phi_t$ defina $\phi_t = g(x_{N/2})$
- Se trunca la población de manera tal que $\mathcal{F}(x_s, t) \leq \phi_t$ donde x_s son todos los individuos para los cuales la función objetivo es menor o igual que θ_t

Con el método de selección de individuos definido y las fórmulas de actualización de parámetros deducidas en la sección anterior el algoritmo BUMDA en una dimensión es el presentado en el Algoritmo 6:

Algorithm 6 BUMDA una dimensión

1: $N \leftarrow$ Número de individuos de la población.
2: $minvar \leftarrow$ Mínima varianza permitida.
3: $t \leftarrow 0$
4: $P_t \leftarrow$ U(dominio) de tamaño N.
5: **while** $\sigma > minvar$ **do**
6: Evalúe y trunque la población P_t en un tamaño n
7: $\mathcal{F}^*(x) = \mathcal{F}(x) - \mathcal{F}(x_n) + 1$ (Para maximizar o)
8: $\mathcal{F}^*(x) = \mathcal{F}(x_n) - \mathcal{F}(x) + 1$ (Para minimizar)
9: $\mu \leftarrow \frac{\sum_{i=1}^{n} \mathcal{F}^*(x_i) x_i}{\sum_{i=1}^{n} \mathcal{F}^*(x_i)}$
10: $\sigma \leftarrow \frac{\sum_{i}^{n} \mathcal{F}^*(x_i)(x_i - \mu)^2}{1 + \sum_{i}^{n} \mathcal{F}^*(x_i)}$
11: $P_{t+1} \leftarrow$ Genere $N-1$ individuos de $\mathcal{N}(\mu, \sigma)$ e inserte el individuo élite.
12: $t \leftarrow t + 1$
13: **end while**

De manera generalizada, desarrollan el BUMDA por dimensión de manera independiente para cada componente de los individuos.

El seudocódigo está descrito en el Algoritmo 7

Algorithm 7 BUMDA D dimensiones

1: $N \leftarrow$ Número de individuos de la población.
2: $minvar \leftarrow$ Mínima varianza permitida.
3: $t \leftarrow 0$.
4: $P_t \leftarrow$ U(dominio) de tamaño N.
5: **while** $\sigma > minvar$ **do**
6: Evalúe y trunque la población P_t en un tamaño n
7: **for** d=1,...,D **do**
8: $\mu[d] \leftarrow \frac{\sum_{i=1}^{nselec} \mathcal{F}^*(X_i) X_{id}}{\sum_{i=1}^{nselec} \mathcal{F}^*(X_i)}$
9: $\sigma[d] \leftarrow \frac{\sum_{i}^{nselec} \mathcal{F}^*(X_i)(X_{id} - \mu[d])^2}{1 + \sum_{i}^{nselec} \mathcal{F}^*(X_i)}$ ▷ Con $\mathcal{F}^*$ definida en el Algoritmo 4
10: **end for**
11: $P_{t+1} \leftarrow$ Genere $N-1$ individuos de $\mathcal{N}(\mu, \sigma)$ e inserte el individuo élite.
12: $t \leftarrow t + 1$
13: **end while**

El criterio de paro del BUMDA es que la varianza sea menor que un valor permitido, pues en este algoritmo la varianza tiende a ser 0 para un número grande de generaciones, sin embargo, la varianza puede crecer o disminuir según los individuos seleccionados y su

valor en la función $\mathcal{F}$. Es importante notar que el BUMDA solo necesita el tamaño de la población N y el número de individuos n para el truncamiento de la población; la estimación de los parámetros la distribución de búsqueda μ y σ se hace rápidamente pues no se necesita información adicional a la de los individuos y su evaluación en la función. En el Capítulo 4 se muestran los resultados de este algoritmo con algunos problemas del benchmark en diferentes dimensiones.

2.3. Resumen

En este capítulo se describieron dos algoritmos de optimización estocástica con una base matemática fuerte: el ∇_d-NEDA y el BUMDA. Estos algoritmos tienen en común que son dos EDAs que usan el gradiente, sin embargo lo usan de la manera diferente, el ∇_d- NEDA calcula el gradiente sobre el espacio de búsqueda y el BUMDA lo hace sobre el espacio de los parámetros. Esto muestra la versatilidad del gradiente e inspira el desarrollo del siguiente capítulo en el que se muestra la aportación más importante de este trabajo de investigación. En el Capítulo 4 se comparan el algoritmo desarrollado en el Capítulo 3 con estos dos algoritmos.

CAPÍTULO 3

KSG-EDA

En este capítulo se desarrolla un nuevo EDA, el cual utiliza una distribución Normal Multivariada como distribución de búsqueda y actualiza los parámetros de la misma usando descenso de gradiente estocástico para minimizar la divergencia Kullback-Leibler entre la distribución Boltzmann y dicha Normal.

La distribución Boltzmann, como se vio en el capítulo 2, es una distribución de búsqueda ideal para los EDAs pues tiene su masa de probabilidad concentrada al rededor del óptimo de la función a optimizar. Para ilustrar, en la Figura 3.1 en rojo se puede ver la función de densidad Boltzmann asociada a la función a optimizar en negro. Como se sabe que un EDA no puede usar directamente esta distribución de búsqueda soñada, se quiere encontrar una densidad Normal Multivariada, que se vaya adaptando a lo largo de las iteraciones, que empiece por ejemplo en la Normal verde, que se vaya ajustando hacia una distribución que se asemeja a la distribución Boltzman y de la que es posible muestrear el óptimo con mayor probabilidad. Ahora bien, ¿cómo aproximar los parámetros de esta distribución Normal para

que se parezca a la Boltzman? La idea que se desarrolla en este trabajo es ajustar los parámetros de la normal usando SGD.

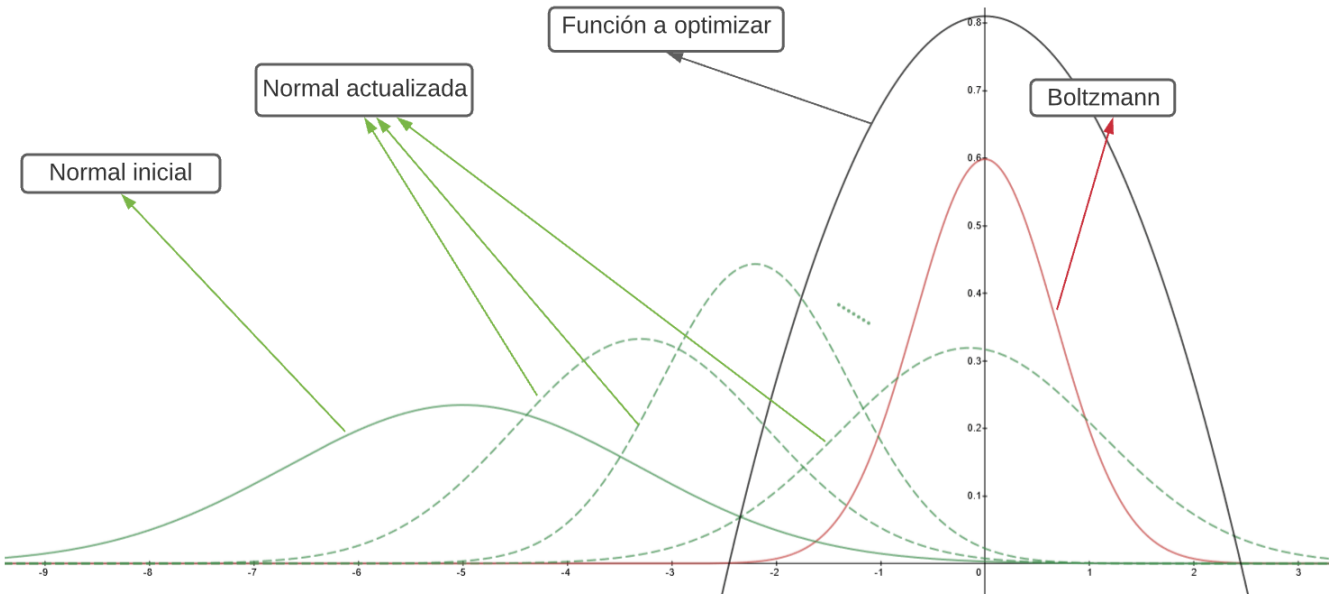


Figura 3.1: Explicación del problema

Para el SGD, tal cual se vio en el capítulo 1, se necesita unos *datos* a los cuales se les ajusta un modelo minimizando una función de error. En este trabajo, se van a considerar los datos como la distribución Boltzmann a la cual se le quiere ajustar como modelo una Normal multivariada. Como se quiere ajustar una distribución a otra, la función error que se propone es la divergencia Kullback-Leibler.

Para desarrollar las expresiones a usar en la actualización de los parámetros de la Normal Multivariada $\mathcal{N}(x, \mu, \sigma)$, primero es necesario hacerlo en $\mathbb{R}$ para poder extender esta idea a más dimensiones.

3.1. Deducción de fórmulas

Deducción del ajuste de una Normal Multivariada a una Boltmann La distribución Boltzmann de la Función a optimizar $\mathcal{F} : \mathbb{R} \to \mathbb{R}$

$$\mathcal{P}(x, \mathcal{F}) = \int_x \frac{\exp(\beta \mathcal{F}(x))}{Z} dx, \quad p(x) = \frac{\exp(\beta \mathcal{F}(x))}{Z} \tag{3.1}$$

Donde $Z = \int_{-\infty}^{\infty} \exp(\beta \mathcal{F}(x)) dx$.

Se desea aproximar $\mathcal{P}$ con una distribución Normal $\mathcal{Q}$ cuya función de densidad es:

$$q(x, \mu, \sigma) = \frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma}\right)}{\sqrt{2\pi\sigma}} \tag{3.2}$$

La divergencia Kullback-Leibler de $\mathcal{P}$ a $\mathcal{Q}$ está dada por

$$KL_{\mathcal{P},\mathcal{Q}}(x, \mu, \sigma) = \int_x p(x) \log\left(\frac{p(x)}{q(x, \mu, \sigma)}\right) dx \tag{3.3}$$

Por simplicidad, se denota 3.3 como

$$KL_{\mathcal{P},\mathcal{Q}} = \int_x p \log\left(\frac{p}{q}\right) dx$$

Ahora bien, para encontrar la fórmula de actualización de los parámetros μ y σ de la Normal, es necesario encontrar una expresión general de la derivada de la anterior expresión con

respecto a cualquier parámetro θ

$$\begin{aligned}\frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial\theta} &= \int_x \frac{\partial}{\partial\theta}\left(p\log\left(\frac{p}{q}\right)\right)dx \\ &= \int_x p\frac{\partial}{\partial\theta}\left(\log\left(\frac{p}{q}\right)\right)dx \\ &= \int_x p\frac{\partial}{\partial\theta}\left(\log(p)-\log(q)\right)dx \\ &= \int_x p\frac{\partial}{\partial\theta}\left(-\log(q)\right)dx\end{aligned}$$

Fijando un individuo x y reemplazando θ por μ y σ se obtienen las siguientes expresiones

$$\begin{aligned}\frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial\mu} &= -\int_x p\frac{\partial}{\partial\mu}\left(\log\left(\frac{1}{\sqrt{2\pi\sigma}}\exp\left(\frac{-(x-\mu)^2}{2\sigma}\right)\right)\right)dx \\ &= -\int_x p\frac{\partial}{\partial\mu}\left(\log\left(\frac{1}{\sqrt{2\pi\sigma}}\right)+\log\left(\exp\left(\frac{-(x-\mu)^2}{2\sigma}\right)\right)\right)dx \\ &= -\int_x p\frac{\partial}{\partial\mu}\left(\frac{-(x-\mu)^2}{2\sigma}\right)dx \\ &= \int_x -p\frac{2(x-\mu)}{2\sigma}dx \\ &= \int_x -p\frac{(x-\mu)}{\sigma}dx \end{aligned} \tag{3.4}$$

$$\begin{aligned}\frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial\sigma} &= -\int_x p\frac{\partial}{\partial\sigma}\left(\log\left(\frac{1}{\sqrt{2\pi\sigma}}\right)-\frac{(x-\mu)^2}{2\sigma}\right)dx \\ &= -\int_x p\left[\sqrt{2\pi\sigma}\frac{1}{\sqrt{2\pi}}\frac{-1}{2}\frac{1}{\sigma^{3/2}}-\frac{(x-\mu)^2}{2\sigma^2}(-1)\right]dx \\ &= -\int_x p\left[\frac{\sigma^{1/2}}{\sigma^{3/2}}\left(\frac{-1}{2}\right)+\frac{(x-\mu)^2}{2\sigma^2}\right]dx \\ &= \int_x p\left[\frac{1}{2\sigma}-\frac{(x-\mu)^2}{2\sigma^2}\right]dx \end{aligned} \tag{3.5}$$

Dada una muestra $\{x_1, x_2, \cdots, x_n\}$, una buena aproximación para 3.4 y 3.5 es

$$\frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial \mu} \sim \sum_{i=1}^{n} \frac{\exp(\beta \mathcal{F}(x_i))}{\sum_{j=1}^{n} \exp(\beta \mathcal{F}(x_j))} \frac{(x_i - \mu)}{\sigma} \tag{3.6}$$

$$\frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial \sigma} \sim \sum_{i=1}^{n} \frac{\exp(\beta \mathcal{F}(x_i))}{\sum_{j=1}^{n} \exp(\beta \mathcal{F}(x_j))} \left[\frac{1}{2\sigma} - \frac{(x_i - \mu)^2}{2\sigma^2} \right] \tag{3.7}$$

Las expresiones (3.6) y (3.7) son las aproximaciones de gradientes de la función de error con respecto a los parámetros del modelo. El siguiente paso es mostrar un algoritmo simple (natural) servirá como base para el algoritmo propuesto.

Algorithm 8 Algoritmo base

1: $N \leftarrow$ Cantidad de individuos en la población
2: $n \leftarrow$ Cantidad de individuos para actualizar los parámetros.
3: $\alpha_\mu \leftarrow$ Tamaño de paso para μ
4: $\alpha_\sigma \leftarrow$ Tamaño de paso para σ
5: $t \leftarrow 0$ Contador de generaciones, inicializado en 0.
6: $P^{(t)} \leftarrow \mathcal{U}(Dominio)$ Población inicial de tamaño N
7: $f^{(t)} \leftarrow \mathcal{F}(P^{(t)})$
8: $P_m^{(t)} \leftarrow$ Muestra aleatoria de $P^{(t)}$ de tamaño n
9: $f_m^{(t)} \leftarrow \mathcal{F}(P_m^{(t)})$
10: $\mu^{(t)} \leftarrow$ Media($P_m^{(t)}$)
11: $\sigma^{(t)} \leftarrow$ Varianza($P_m^{(t)}$)
12: **while** No se cumpla criterio de paro **do**
13: $\quad t \leftarrow t + 1$
14: $\quad \dfrac{\partial KL_{P,Q}}{\partial \mu} \leftarrow$ Ecuación (3) usando $P_m^{(t-1)}$ y $f_m^{(t-1)}$
15: $\quad \dfrac{\partial KL_{P,Q}}{\partial \sigma} \leftarrow$ Ecuación (4) usando $P_m^{(t-1)}$ y $f_m^{(t-1)}$
16: $\quad \mu^{(t)} \leftarrow \mu^{(t-1)} - \alpha_\mu \dfrac{\partial t_{P,Q}}{\partial \mu}$
17: $\quad \sigma^{(t)} \leftarrow \sigma^{(t-1)} - \alpha_\sigma \dfrac{\partial t_{P,Q}}{\partial \sigma}$
18: $\quad \mathcal{S} \leftarrow$ Simule N individuos de $\mathcal{N}(\mu^{(t)}, \sigma^{(t)})$
19: $\quad P^{(t)} \leftarrow \mathcal{S}$
20: $\quad f^{(t)} \leftarrow \mathcal{F}(P^{(t)})$
21: $\quad P_m^{(t)} \leftarrow$ Muestra aleatoria de $P^{(t)}$ con tamaño n
22: $\quad f_m^{(t)} \leftarrow \mathcal{F}(P_m^{(t)})$
23: **end while**

Básicamente, a este algoritmo en el paso 1 se le ingresa el número N de individuos de la población, el cual se mantiene a lo largo de las generaciones, en el 2 se ingresa el parámetro n, que debe ser mucho menor que N y se usa para calcular la aproximación del gradiente, Asimismo, en los pasos 3 y 4 se guardan los tamaños de paso α_μ y α_σ los cuales se mantienen constantes y que se necesitan para la actualización de μ y σ en los pasos 14 y 15 del algoritmo. En el paso 5 está inicializado t el cual cuenta el número de generaciones. El paso 6 del Algoritmo consiste en crear una población inicial con tamaño N de manera uniforme sobre el dominio de la función $\mathcal{F}$ a minimizar, en el 7 se evalúa esta población. Los pasos 8 y 9 toman una muestra aleatoria de la población con sus respectivas evaluaciones de función.

En el paso 10 y 11 se inicializan μ y σ con la media y la varianza de la población elegida en el paso 8. el paso 12 es el ciclo que se repetirá hasta un criterio de paro, en 13 se actualiza la generación t, en el 14 y 15 calculan el gradiente estimado usando las Ecuaciones (3.6 y (3.7) en las cuales los individuos x_i son los individuos de la población $P_m^{(t-1)}$ y $\mathcal{F}(x_i)$ es su respectiva evaluación $f_m^{(t)}$. En los pasos 16 y 17 se actualizan los parámetros μ y σ con SGD que son usados en el paso 18 para muestrear la nueva generación $P^{(t)}$. Nuevamente en el paso 21 se toma una muestra aleatoria de la población $P^{(t)}$ y se evalúa en la función en el paso 21 y se vuelve al paso 12 hasta que no se cumpla algún criterio de paro.

Existen varios puntos a considerar del Algoritmo base 8 dadas las expresiones (3.6) y (3.7) que usa:

- $\exp \beta\mathcal{F}(\cdot)$:

  En este trabajo de investigación se utilizan funciones a las que se les ingresa un vector $x \in \mathbb{R}^d$ y se conoce el valor $\mathcal{F}(x) \in \mathbb{R}$ por lo tanto, la imagen de la función $\mathcal{F}$ a optimizar se puede cubrir por un intervalo $[a, b] \in \mathbb{R}$, pero este intervalo puede ser muy 'grande' y como la distribución Boltzmann está compuesta por la función exponencial de $\mathcal{F}$, al ingresar un valor muy grande a la exponencial, se pueden ocasionar problemas numéricos desastrosos. Es por esto que es conveniente que los valores a y b estén controlados. Para efectos de este trabajo, de ser necesario, se usa la siguiente transformación de la imagen de $\mathcal{F} \geq 0$[1], llamada la función de energía de $\mathcal{F}$ para que $[a, b] \subseteq [0, 1]$.

  $$\mathcal{E}(x) = \frac{\text{máx}\,\mathcal{F} - \mathcal{F}(x)}{\text{máx}\,\mathcal{F}} \tag{3.8}$$

  La función de energía 3.8 tiene el mismo dominio de $\mathcal{F}$ pero su rango está acotado entre $[0, 1]$, es 0 cuando $\mathcal{F}$ alcanza el máximo y 1 cuando $\mathcal{F}$ sea 0. Así los problemas de minimización de $\mathcal{F}$ se convierten en problemas de maximización de $\mathcal{E}$ y los de maximización de $\mathcal{F}$ en minimización de $\mathcal{E}$.

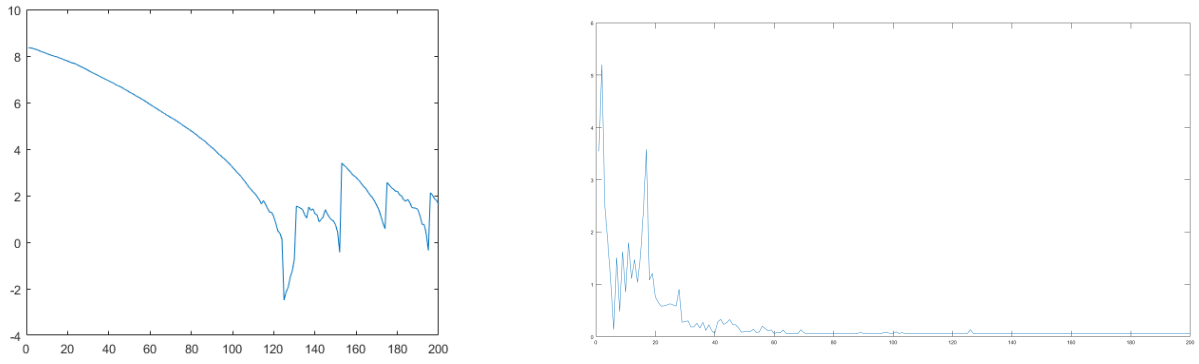
[1] Si $\mathcal{F} < 0$ se le suma una constante c lo suficientemente grande para que $\mathcal{F} + c \geq 0$

- σ como denominador en (3.6) y (3.7):

  En los EDAs un problema usual es la convergencia prematura, esto quiere decir que la varianza de los individuos de la población tiende a disminuir rápidamente haciendo que estos se concentren en una región muy reducida. Cuando σ está muy pequeña, hace que los gradientes (3.6) y (3.7) sean muy grandes o que incluso se indeterminen debido a los problemas numéricos que pueden ocurrir. En esta situación se hace imposible usar directamente las ecuaciones (3.6) y (3.7) para la actualización de los parámetros de la distribución de búsqueda. Este problema puede evitarse de varias maneras: usando un término de regularización, una función de penalización o imponiendo una restricción, es decir, una cota inferior para los valores de σ. La estrategia usada para evitar estos posibles problemas numéricos fue acotar inferiormente a σ y esto dio pie a crear una táctica de exploración e intensificación para el algoritmo que se explicará más adelante.

- Descenso de gradiente para el parámetro σ :

  El parámetro σ, al ser la varianza de la distribución Normal, tiene una restricción adicional y es que siempre tiene que ser mayor que 0. Al usar SGD para actualizarlo no se tienen en cuenta estas restricciones y se puede llegar a obtener una varianza negativa.



(a) σ usando SGD

(b) σ empírica

Figura 3.2: Comportamiento del parámetro σ

La Figura 3.2 muestra el comportamiento de σ calculado con SGD 3.2a y σ empírico 3.2b en una dimensión a lo largo de 200 iteraciones.

Dado que el parámetro σ tiene todas las restricciones comentadas, que necesita un tamaño de paso α_σ y que además es el parámetro que indica qué tan alejados de la media μ deben muestrearse nuevos individuos, la propuesta en este trabajo de investigación es usar la varianza empírica con una cota inferior $c_\sigma > 0$. Cuando σ sea menor que c_σ, se muestreará de la una Normal con varianza c_σ, esto le permitirá al algoritmo seguir explorando la zona en la que se encuentre. Ahora bien, se necesita muestrear de una Normal con una varianza suficientemente pequeña para para mejorar la calidad del óptimo en caso de que no se pueda salir de un óptimo local o se esté al rededor del óptimo global, la idea es usar una segunda normal con media el mejor individuo de la población actual y varianza mucho más pequeña que σ_e.

Antes de mostrar el algoritmo propuesto, hay que considerar que todos los cálculos que se desarrollaron hasta el momento están en una dimensión. El paso a seguir es cómo extender las fórmulas desarrolladas a más de una dimensión, después de eso se mostrará el algoritmo propuesto de manera general.

Caso: Función a optimizar $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}$

La propuesta para generalizar lo anterior, es extender de manera natural a más dimensiones, se va a suponer que las variables son independientes, esto quiere decir que tanto la media μ como la varianza σ de la distribución de búsqueda $\mathcal{N}(\mu, \sigma)$ son vectores de $\mathbb{R}^d$

$$\text{Población} \quad \begin{cases} X_1 = & (x_{11} & x_{12} & \cdots & x_{1d}) \\ X_2 = & (x_{21} & x_{22} & \cdots & x_{2d}) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ X_n = & (x_{n1} & x_{n2} & \cdots & x_{nd}) \end{cases}$$

$$\text{Parámetros} \quad \begin{cases} \mu & = & (\mu_1 & \mu_2 & \cdots & \mu_d) \\ \sigma & = & (\sigma_1 & \sigma_2 & \cdots & \sigma_d) \end{cases}$$

$$\text{Gradientes} \quad \begin{cases} \dfrac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial \mu} & = & \left(\dfrac{\partial KL_{\mathcal{P},\mathcal{Q}}(1)}{\partial \mu} & \dfrac{\partial KL_{\mathcal{P},\mathcal{Q}}(2)}{\partial \mu} & \cdots & \dfrac{\partial KL_{\mathcal{P},\mathcal{Q}}(d)}{\partial \mu} \right) \\ \dfrac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial \sigma} & = & \left(\dfrac{\partial KL_{\mathcal{P},\mathcal{Q}}(1)}{\partial \sigma} & \dfrac{\partial KL_{\mathcal{P},\mathcal{Q}}(2)}{\partial \sigma} & \cdots & \dfrac{\partial KL_{\mathcal{P},\mathcal{Q}}(d)}{\partial \sigma} \right) \end{cases}$$

La propuesta es calcular el gradiente por componentes usando las Ecuaciones (3.6), (3.7) y 3.8 Para $k = 1, 2, \cdots, d$ la componente k-ésima los gradientes de la divergencia Kulback-Leibler con respecto a μ y σ son:

$$\frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial \mu}(k) \sim \sum_{i=0}^{n} \frac{\exp(\beta \mathcal{E}(X_i))}{\sum_{j=0}^{n} \exp(\beta \mathcal{E}(X_j))} \frac{(x_{ik} - \mu_k)}{\sigma_k} \tag{3.9}$$

$$\frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial \sigma}(k) \sim \sum_{i=0}^{n} \frac{\exp(\beta \mathcal{E}(X_i))}{\sum_{j=0}^{n} \exp(\beta \mathcal{E}(X_j))} \left[-\frac{(x_{ik} - \mu_k)^2}{2\sigma_k^2} + \frac{1}{2\sigma_k} \right] \tag{3.10}$$

En la siguiente sección se describe el algoritmo que usa las expresiones la generalización hecha en esta sección. También tiene en cuenta todas consideraciones hechas con anterioridad.

3.2. Descripción del algoritmo

Se describirá paso por paso el KSG-EDA.

Algorithm 9 KSG-EDA

$N \leftarrow$ Cantidad de individuos en la población
$n \leftarrow$ Cantidad de individuos para actualizar los parámetros
$s \leftarrow$ Cantidad de individuos muestreados de la distribución de búsqueda
$\alpha_\mu \leftarrow$ Tamaño de paso para μ
$c_\sigma \leftarrow$ límite inferior de σ_k para $k = 1, \ldots, d$
$\beta_0 \leftarrow$ Parámetro de la distribución Boltzmann
$t \leftarrow 0$ Contador de generaciones
$P^{(t)} \leftarrow \mathcal{U}(Dominio)$, $f^{(t)} \leftarrow \mathcal{F}(P^{(t)})$
$P_m^{(t)} \leftarrow$ Muestra aleatoria de $P^{(t)}$ con tamaño s
$f_m^{(t)} \leftarrow \mathcal{F}(P_m^{(t)})$
$\mu^{(t)} \leftarrow$ Media($P_m^{(t)}$)
$\sigma^{(t)} \leftarrow$ Varianza($P_m^{(t)}$) por componentes
while No se cumpla criterio de paro **do**
    $\mathcal{E} \leftarrow \left(max(f_m^{(t)}) - f_m^{(t)}\right) / max(f_m^{(t)})$ ▷ Función de Energía
    for $k = 1, \cdots, d$ **do**
        $\frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial \mu}(k) \leftarrow$ Ecuación (3.9) Usando $P_m^{(t)}$ y $\mathcal{E}$
        $\mu_k^{(t)} \leftarrow \mu_k^{(t-1)} - \alpha_\mu \frac{\partial KL_{\mathcal{P},\mathcal{Q}}}{\partial \mu}$
    end for
    $\mathcal{S} \leftarrow$ Simule s individuos de $\mathcal{N}(\mu^{(t)}, \sigma^{(t)})$
    $\mathcal{S} \leftarrow$ Reinserción($\mathcal{S}$) ▷ Si es necesario
    $P^{(t+1)} \leftarrow \mathcal{S}$ Reemplaza los peores s individuos en $P^{(t)}$
    $P_m^{(t+1)} \leftarrow$ Muestra aleatoria de $P^{(t+1)}$ con tamaño n
    $f_m^{(t+1)} \leftarrow \mathcal{F}(P_m^{(t+1)})$
    $\sigma^{(t+1)} \leftarrow$ Varianza($P_m^{(t)}$) por componentes
    if Alguna $\sigma_k^{(t+1)} < c_\sigma$ **then**
        $\nu \leftarrow \sigma$
        $\nu(\nu < c_\sigma) = \nu(\nu < c_\sigma) * 10^{-4}$ ▷ Las $\nu_i < c_\sigma$ se multiplican por 10^{-4}
        $\mathcal{M} \leftarrow$ Simule $N/2$ individuos de una Normal con media el mejor individuo y varianza ν
        $\mathcal{M} \leftarrow$ Reinserción($\mathcal{M}$) ▷ Si es necesario
        $P^{(t+1)} \leftarrow \mathcal{M}$ Reemplaza los peores $N/2$ individuos en $P^{(t+1)}$
        $\beta = \beta_0$
        $\sigma^{(t+1)}(\sigma^{(t+1)} < c_\sigma) = c_\sigma$ ▷ Las $\sigma_i < c_\sigma$ se les asigna c_σ
    end if
    $t \leftarrow t + 1$
    $\beta = \beta + \epsilon$
end while

- **Inicialización de variables:** El Algoritmo 9 KSG-EDA en el paso 1 se le ingresa el número N de individuos de la población, el cual se mantiene a lo largo de las generaciones, en el 2 se ingresa el parámetro n, que debe ser mucho menor que N y se usa para calcular la aproximación del gradiente de la divergencia KL, Asimismo, en el paso 3 se guarda la cantidad s de individuos muestreados de la distribución de búsqueda, en 4 se guarda el tamaño de paso α_μ el cual se mantiene constante y se usa para la actualización del parámetro μ en el paso 15 del algoritmo. En el paso 5 se guarda del valor del límite inferior permitido para cada una de las componentes del parámetro σ. En el paso 6 se guarda el parámetro β de la Ecuación 3.1. En el 7 se está inicializado t el cual cuenta el número de generaciones. El paso 8 del Algoritmo consiste en crear una población inicial con tamaño N de manera uniforme sobre el dominio de la función $\mathcal{F}$ a minimizar y se evalúan en $\mathcal{F}$. Los pasos 9 y 10 toman una muestra aleatoria de la población $P_m^{(t)}$ de tamaño s con sus respectivas evaluaciones de función $f_m^{(t)}$. En el paso 11 y 12 se inicializan μ y σ con la media y la varianza de $P_m^{(t)}$.

- **ciclo que se repite:** El paso 13 es el ciclo que se repetirá hasta un criterio de paro. En 14 se calcula la función de energía $\mathcal{E}$ que transforma el problema de minimización de $\mathcal{F}$ en el de maximización de $\mathcal{E}$. En paso 15 está el ciclo que actualiza las componentes de μ: en el 16 se calcula el gradiente estimado de la divergencia KL con respecto a μ en el cual se usa la población $P_m^{(t)}$, mientras que en el paso 17 se actualiza el parámetro μ en dirección al gradiente (pues ya se está en un problema de maximización), los individuos x_i son los individuos de la población $P_m^{(t)}$. En el paso 19 y 20 se muestrean s nuevos individuos y si se salen del espacio de búsqueda se vuelven a reinsertar en la población. En el paso 21 se hace la selección de los individuos que harán parte de la siguiente generación: los s individuos generados reemplazan los peores s individuos en la población lo que hace que el KSG-EDA tenga diversidad en su población pero conserve sus mejores individuos. En el paso 22 se toma una muestra aleatoria de la población $P^{(t)}$ y se evalúa en la función en el paso 23. En el paso 24 se actualiza σ con la varianza empírica de $P_m^{(t+1)}$ y puede pasar que algunas componentes del parámetro

σ sean menor que el límite c_σ permitido

- **Mejora del óptimo actual:** si esto ocurre entonces se habilita el paso 26 en el cual se guarda el parámetro $\sigma^{(t+1)}$ en un vector auxiliar ν. en el paso 27 se reducen aún más las componentes de ν que son menores a c_σ por un factor de 10^{-4} y en el paso 28 se muestrean $N/2$ individuos al rededor del mejor individuo con la varianza calculada en el paso 27. Este paso intensifica la búsqueda al rededor del mejor poblador con la intensión de mejorar la calidad del óptimo actual, el paso 29 reinserta estos individuos generados si es necesario. En el paso 30 se reemplazan los individuos generados en el paso 29 por los peores individuos de la población actual. Se reduce el parámetro β a un valor fijo β_0 en el paso 31 con la intensión quitar presión de selección y en la próxima generación se pueda explorar un poco más en el espacio de búsqueda. El paso 32 fija las $\sigma^{(t+1)}$ componentes de sigma que son menores que c_σ en el valor c_σ.

En el paso 34 se actualiza la generación t y se vuelve al paso13 mientras no se cumpla el criterio de paro.

En el 35 Se incrementa un ϵ el parámetro β para aumentar la presión de selección en la próxima iteración.

Comparación entre Algoritmo base 8 y el Algoritmo 9 KSG-EDA

- En el Algoritmo 9 utiliza la función de energía descrita en la ecuación (3.8) mientras que el Algoritmo 8 usa la función $\mathcal{F}$ directamente.
- El SGD actualiza los parámetros de la distribución de búsqueda del KSG-EDA usando un subconjunto de la población denotado por $P_m^{(t)}$, una de las principales diferencias del Algoritmo 8 y el Algoritmo 9 es que el parámetro σ en el Algoritmo 9 se actualiza con la varianza empírica $P_m^{(t)}$, mientras que en el Algoritmo 8 se utiliza la Ecuación (3.10). Puede darse el caso que algunas dimensiones colapsen más rápido que otras, por eso el ajuste de σ Algoritmo 9 no es general para todas las dimensiones sino para las dimensiones colapsadas.

- En el condicional del paso 25 es donde al Algoritmo 9 KSG-EDA hace la parte de intensificación (de mejora del óptimo), es donde aparece una segunda densidad Normal de la que se muestrean la mitad de los individuos de la población. Esta parte no aparece en el Algoritmo 8.

- En el Algoritmo 9 aparece una modificación del parámetro β el cual ejerce presión de selección. Cuando el algoritmo ingresa al paso 25 es porque la varianza de la población $P_m^{(t)}$ es pequeña en alguna de sus componentes y el parámetro β se reinicia con la intensión de que el algoritmo siga explorando la región donde se encuentra. Fuera del condicional del paso 25 el parámetro se aumenta con la intensión de aumentar la presión de selección.

3.2.1. Descripción Gráfica del KSG-EDA

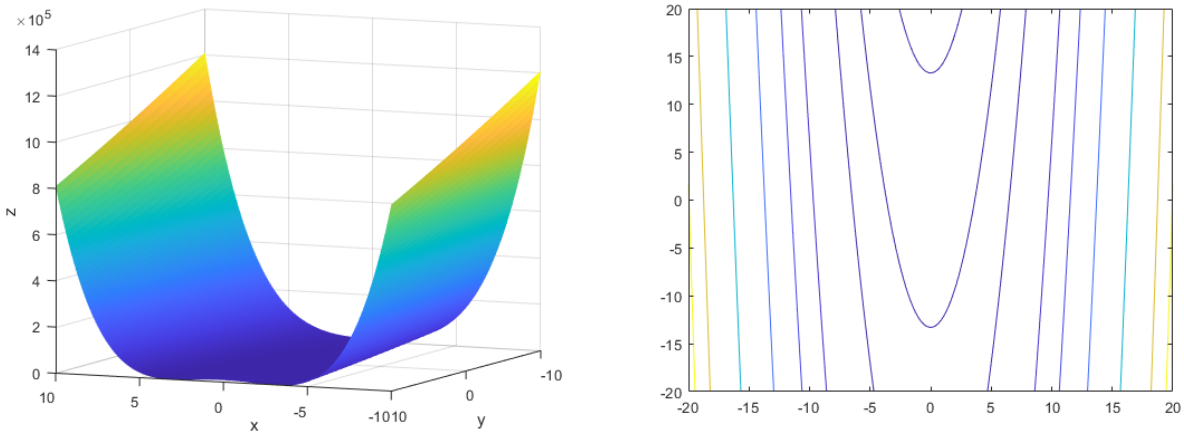
Para explicar de manera gráfica el comportamiento del KSG-EDA descrito en el Algoritmo 9, se probará en la función Rosenbrock. La función de Rosenbrock fue presentada por Howard H. Rosenbrock en 1960. Es una función no convexa utilizada para probar el rendimiento en algoritmos de optimización.

La función en dos dimensiones está definida por:

$$\mathcal{F}(x, y) = (a - x)^2 + b(y - x^2)^2$$

El mínimo de la funció tiene como coordenadas (a, a^2), donde $\mathcal{F}(a, a^2) = 0$.

En la Figura 3.3 se muestra el comportamiento de la función Rosenbrock en dimensión 2 con $a = 1$ y $b = 100$, donde $-10 < x < 10$, $-10 < y < 10$. En la Figura 3.3a se muestra la gráfica de la función mientras que en la Figura 3.3b se muestran las curvas de nivel.



(a) Superficie **(b)** Curvas de nivel

Figura 3.3: Función Rosenbrock en 2D

KSG-EDA con la función Rosenbrock en 2D

En la Figura 3.4 se muestran varias fases del algoritmo. La elipse roja representa la distribución Normal actual que corresponde a la distribución de la que se generaron los individuos representados por *. La elipse azul representa la Normal actualizada y de la que muestrearán individuos de la siguiente generación.

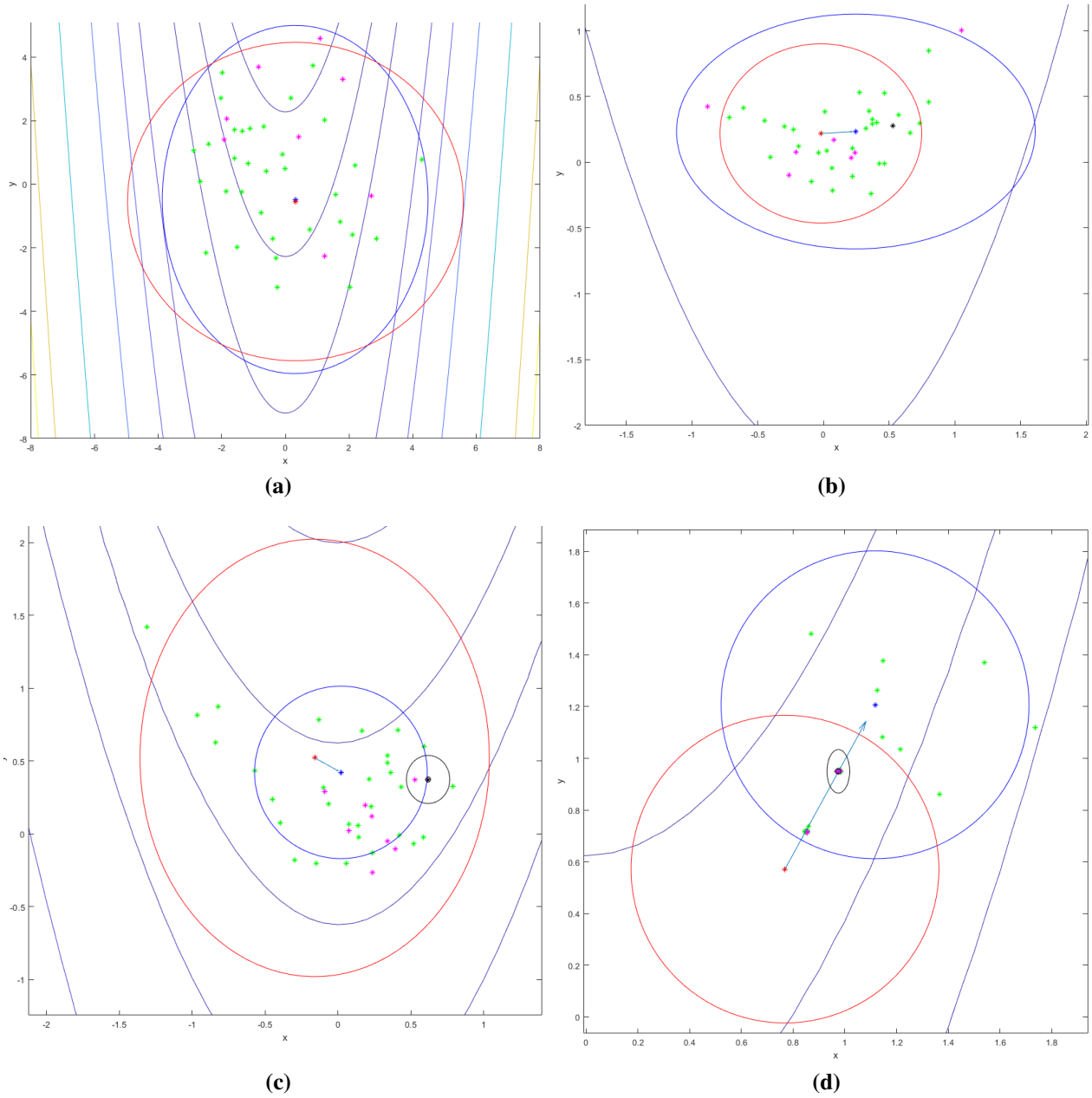


Figura 3.4: Comportamiento del KSG-EDA con la función Rosenbrock en 2D

A continuación se explicarán cada una de las gráficas:

Figura 3.4a En esta figura se muestra la fase inicial del Algoritmo 9, se representan las curvas de nivel de la función Rosenbrock y sobre ellas se muestra una población inicial generada de manera aleatoria sobre el rectángulo $[4, 4] \times [4, 4]$ los cuales aparecen como * y *. La

elipse roja representa la densidad Normal empírica cuya excentricidad está determinada por la varianza empírica y centro representado por un * es la media empírica. Los * color magenta son los individuos escogidos de manera aleatoria para actualizar los parámetros de la Normal inicial. La actualización de la Normal está representada por la elipse en color azul cuya media es el * azul.

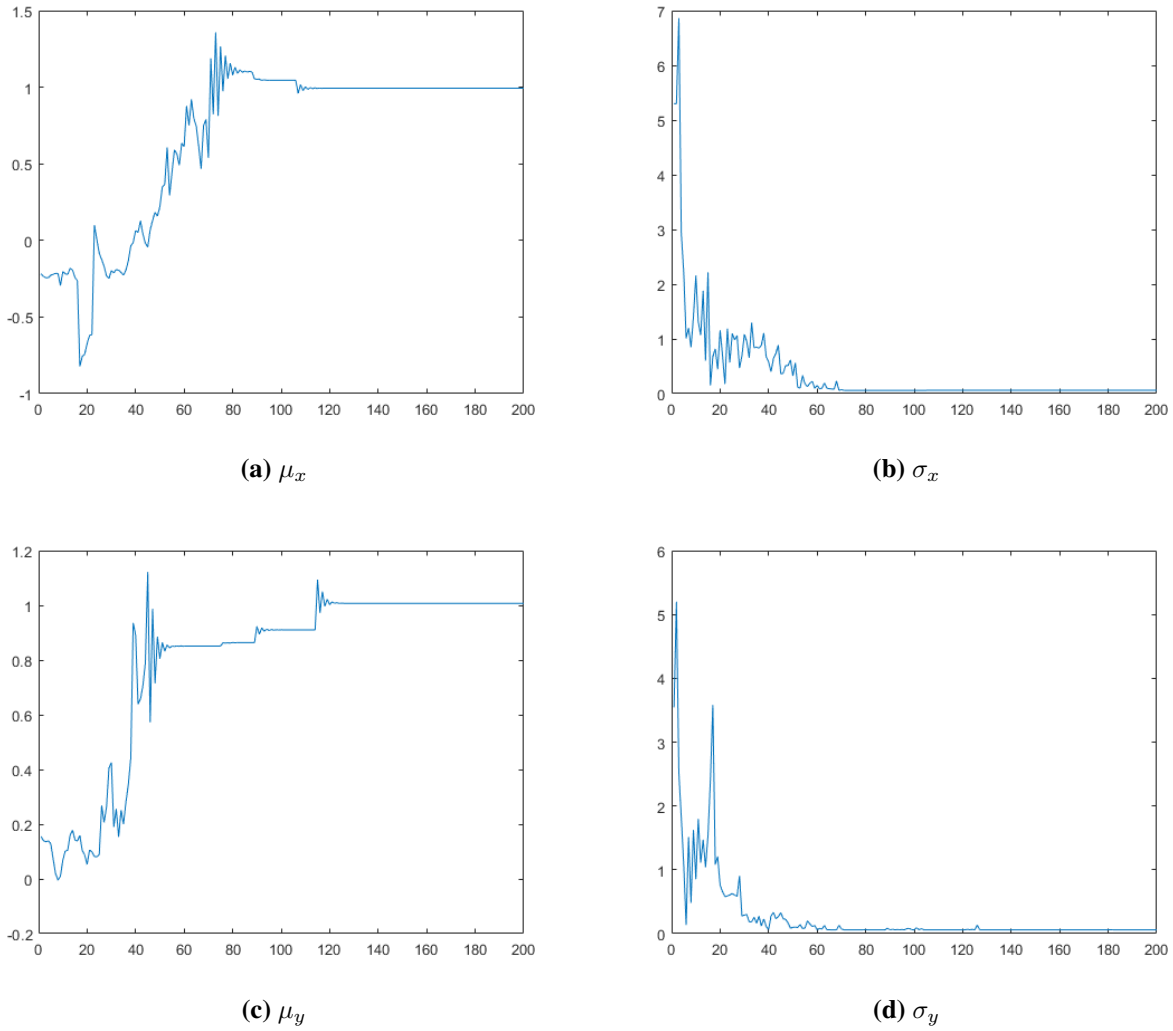
Figura 3.4b Este gráfico muestra el vector gradiente calculado con la Ecuación (3.9) el cual actualiza la media roja en la media azul. El mejor individuo en esta iteración se muestra en color negro.

Figura 3.4c En esta gráfica se muestra la primera vez que aparece la elipse negra en el algoritmo, esta elipse representa la normal generada en el la línea 28 del Algoritmo 9 la cual se usa cuando la varianza actual disminuye mucho. Tener dos distribuciones normales para muestrear nuevos individuos es muy ventajoso, pues la grande ayuda a exploración, es decir, la búsqueda de una mejor región, mientras que la otra busca intensificar, es decir, mejorar la calidad del óptimo encontrado.

Figura 3.4d Esta imagen muestra la fase final del algoritmo donde la mayoría de individuos están generados al rededor del óptimo, sin embargo se sigue explorando para intentar salir de posibles óptimos locales.

Evolución de los parámetros μ y σ a lo largo de las iteraciones

La Figura 3.5 se muestra el comportamiento de los parámetros μ y σ a lo largo de las generaciones. En la Figura 3.5a y 3.5c se muestra la evolución de la media de la Normal y cómo se va aproximando a (1,1) que son la componentes del óptimo en dimensión 2. Después de la generación 120 el parámetro μ se estabiliza en (1,1) y no tiene cambios significativos. En la Figura 3.5b se muestra la varianza en x a lo largo de las iteraciones del algoritmo, se tienen grandes cambios al inicio, la varianza sube y baja en las dos dimensiones pero al rededor de la iteración 70 se hace muy pequeña en ambas dimensiones y no vuelve a cambiar significativamente.



(a) μ_x **(b)** σ_x

(c) μ_y **(d)** σ_y

Figura 3.5: Evolución de los parámetros en x a lo largo de las iteraciones

En esta prueba con la función Rosenbrock se observa cómo el algoritmo KSG-EDA va actualizando los parámetros de manera acertada, dirigiendo la búsqueda hacia regiones que prometen tener mejores individuos. Este ejemplo explica la manera en el que el KSG-EDA explora e intensifica la búsqueda de una mejor aproximación del óptimo global.

3.3. Resumen

En este capítulo se muestra la mayor aportación de este trabajo de investigación: el algortimo KSG-EDA, el cual es un EDA que utiliza SGD para minimizar la divergencia KL de la distribución la Normal a distribución Boltzmann.

Dado que los EAs tienen dos etapas importantes: la etapa de *exploración*, que permite recorrer el espacio de búsqueda y encontrar regiones promisorias, y la etapa de *intensificación* la cual posibilita la mejora de la calidad del óptimo encontrado. El KSG-EDA aborda estas dos etapas usando una matriz de covarianza adecuada en cada momento de la ejecución.

CAPÍTULO 4

Experimentos

En esta tesis se describieron 3 EDAs: ∇_d-NEDA, BUMDA y KSG-EDA. Ahora lo que se hará es comparar el rendimiento del KSG-EDA compitiendo con los mismos problemas en los que fueron probados el ∇_d-NEDA y el BUMDA. El KSG-EDA fue programado en MATLAB R2019b y fue ejecutado en el clúster de CIMAT.

Para comparar el BUMDA y el KSG-EDA se tomaron los resultados obtenidos de 50 corridas del BUMDA reportados en la tesis de maestría Mario Iván Jaenz Márquez [28]. Se ejecutaron 50 veces las mismas funciones en las dimensiones 10 y 40. Los criterios de paro aplicados son dos: cuando el error alcanzado $\mathcal{F}(x) - \mathcal{F}(x^*)$ es menor 10^{-6} o cuando el número de evaluaciones de función excede $10^4 \times d$.

Por su parte, para hacer una comparación entre el ∇_d-NEDA y KSG-EDA se tomaron los resultados reportados de 50 corridas del ∇_d-NEDA en el Apéndice B de [36], se ejecutaron 50 veces las mismas funciones $\mathcal{F}$, en las mismas dimensiones d y se usaron dos criterios de

paro: cuando el error alcanzado $\mathcal{F}(x) - \mathcal{F}(x^*)$ es menor 10^{-8} o el número de evaluaciones de función excede $10^4 \times d$.

Dado que se quiere comparar el rendimiento de algoritmos distintos, es interesante contar con una prueba estadística que permita asegurar, con un nivel de confianza, qué algoritmo es más exitoso y además comparar las medias de algunas características, por ejemplo, cuál usa menos evaluaciones de función o cuál se aproxima más al error mínimo pedido. Se dice que un algoritmo tuvo un éxito si la diferencia entre el mejor valor encontrado por el algoritmo, $\mathcal{F}(x)$ y el valor del óptimo real $\mathcal{F}(x^*)$ es menor que la cota establecida.

4.1. Test para comparar dos muestras binomiales

Suponga que $X_1 \sim$ BIN(n_1, p_1) y $X_2 \sim$ BIN(n_2, p_2) Sea $\hat{p}_1 = \dfrac{X_1}{n_1}$, $\hat{p}_2 = \dfrac{X_2}{n_2}$ y $\hat{p} = \dfrac{X_1 + X_2}{n_1 + n_2}$ en [2] se muestra que :

$$Z = \frac{\hat{p}_2 - \hat{p}_1 - (p_2 - p_1)}{\sqrt{\hat{p}(1-\hat{p})\left(\dfrac{1}{n_1} + \dfrac{1}{n_1}\right)}} \to z \sim \mathcal{N}(0,1) \tag{4.1}$$

Para comparar el rendimiento de dos algoritmos es necesario ejecutarlos varias veces en la misma función $\mathcal{F}$ y en la misma dimensión d. Suponga que para una función a optimizar $\mathcal{F}$ el Algoritmo A_i $(i = 1, 2)$ se ejecutó n_i veces y es X_i el número de éxitos obtenidos. Sea $\hat{p}_i = X_i/n_i$. ¿Cuál Algoritmo es más exitoso? es decir, se quiere saber qué valor p_i es mayor, para esto se hace hará una prueba de hipótesis. Considere las siguientes hipótesis: la hipótesis nula es que los algoritmos son igual de exitosos, la hipótesis alternativa es que el algoritmo 1 es más exitoso que el algoritmo 2, es decir:

$$\begin{cases} H_0 : p_1 = p_2 \\ H_a : p_1 > p_2 \end{cases}$$

Para contrastar las hipótesis hay que tener una regla de decisión: se rechaza H_0 a un nivel de significancia α si al aplicar la Ecuación (4.1) con $p_1 = p_2$

$$Z = \frac{\hat{p}_2 - \hat{p}_1}{\sqrt{\hat{p}(1-\hat{p})\left(\frac{1}{n_1} + \frac{1}{n_1}\right)}} \rightarrow z \sim \mathcal{N}(0,1)$$

se obtiene que $Z < -Z_{1-\alpha/2}$, donde $Z_{1-\alpha/2}$ es el percentil $1-\alpha$ de la distribución Normal Estándar.

4.2. z-test para comparar dos Medias

Considere dos muestras independientes de tamaño n_1 y n_2 de dos poblaciones que se distribuyen Normal $X_i \sim \mathcal{N}(\mu_1, \sigma_1)$ y $Y_i \sim \mathcal{N}(\mu_2, \sigma_2)$. Sea $\overline{X}, \overline{Y}, S_1$ y S_2 las medias y varianzas muestreales. Se desea saber qué población tiene menor media. Si las varianzas σ_1 y σ_2 son conocidas, entonces $\overline{X} - \overline{Y} \sim \mathcal{N}(\mu_2 - \mu_1, \sigma_1/n_1 + \sigma_2/n_2)$ esto es:

$$Z = \frac{\overline{X} - \overline{Y} - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \rightarrow z \sim \mathcal{N}(0,1) \tag{4.2}$$

La hipótesis nula es que los algoritmos se comportan igual, es decir, tienen la misma media μ, la hipótesis alternativa es que la media del algoritmo 1 es menor que la media del algoritmo 2, es decir, las hipótesis son

$$\begin{cases} H_0 : \mu_1 = \mu_2 \\ H_a : \mu_1 < \mu_2 \end{cases}$$

Para contrastar las hipótesis se debe contar con una regla de decisión: se rechaza H_0 a un nivel de significancia α si al aplicar la Ecuación (4.2) con $\mu_1 = \mu_2$

$$Z = \frac{\overline{X} - \overline{Y}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \rightarrow z \sim \mathcal{N}(0,1)$$

se obtiene que $Z < -Z_{1-\alpha/2}$, donde $Z_{1-\alpha/2}$ es el percentil $1-\alpha$ de la distribución Normal Estándar.

Las pruebas de hipótesis mostradas en 4.1 y 4.2 asumen que las muestras son independientes y que se distribuyen normal. Las muestras que se obtienen son el resultado de ejecutar dos algoritmos distintos, entonces la hipótesis de independencia se cumple. Como la cantidad de muestras es 50 para cada algoritmo, el teorema del límite central permite cumplir la hipótesis de normalidad. se usa $\alpha = 0.05$. En los experimentos, se tomará como mu_1 y p_1 a los valores obtenidos por el KSG-EDA.

El KSG-EDA descrito en el Algoritmo 9 contiene varios parámetros los cuales fueron determinados tras una búsqueda exhaustiva para la estabilidad y convergencia del algoritmo:

- N cantidad de individuos de la población: se usaron 30 individuos para todas las dimensiones.
- s cantidad de individuos muestreados de la distribución de búsqueda: se usó el $10\,\%$ de N, es decir solo 3 individuos
- α_μ tamaño de paso para el parámetro μ: fue de 0.1.
- c_σ cota interior para cada componente de σ fue 0.06.
- β_0 parámetro inicial del parámetro β: se usó 5. β va aumentando un $\epsilon = 0.02$ a lo largo de las iteraciones y se reinicia a 5 si alguna componente σ es menor que c_σ

Los experimentos se hicieron sobre un conjunto de funciones unimodales (U) y multimodales (M) del benchmark descritos en las Tablas 4.1 y 4.2 respectivamente. Las Tablas 4.4 y 4.5 muestran los resultados del KSG-EDA y el BUMDA en dimensiones 10 y 40. Las Tablas 4.7 y 4.8 muestran los resultados del KSG-EDA y el ∇_d-EDA en las dimensiones 30 y 50.

Tabla 4.1: Funciones de prueba unimodales del bechmark

Funciones	**Definición**	x^*	$\mathcal{F}(x^*)$	**Dominio**
Sphere	$\sum_{i=1}^{d} x_i^2$	$(0, 0, \ldots, 0)$	0	$[-600, 300]^d$
Different Powers	$\sum_{i=1}^{d} \lvert x_i \rvert^{2+10\frac{i-1}{d-i}}$	$(0, 0, \ldots, 0)$	0	$[-20, 10]^d$
Mishra 2	$(1+y)^y$ con $y = d - \frac{1}{2}\sum_{i=1}^{d-1}(x_i + x_{i+1})$	$(1, 1, \ldots, 1)$	2	$[0, 1]^d$
Ellipsoid	$\sum_{i=1}^{d} 10^{6\frac{i-1}{d-1}} x_i^2$	$(0, 0, \ldots, 0)$	0	$[-20, 10]^d$
Tablet	$10^6 x_1^2 + \sum_{i=2}^{d} x_i^2$	$(0, 0, \ldots, 0)$	0	$[-20, 10]^d$
Cigar Tablet	$x_1^2 + \sum_{i=2}^{d-1} 10^4 x_i^2 + 10^8 x_d^2$	$(0, 0, \ldots, 0)$	0	$[-20, 10]^d$
Two Axes	$\sum_{i=1}^{[d/2]} 10^6 x_i^2 + \sum_{i=[d/2]}^{d} x_i^2$	$(0, 0, \ldots, 0)$	0	$[-20, 10]^d$
Parabolic Ridge	$-x_1 + 100\sum_{i=2}^{d} x_i^2$	$(l^{up}, 0, \ldots 0)$	$-l^{up}$	$[-20, 10]^d$
Sharp Ridge	$-x_1 + 100\sqrt{\sum_{i=2}^{d} x_i^2}$	$(l^{up}, 0, \ldots 0)$	$-l^{up}$	$[-20, 10]^d$
Exponential	$-\exp\left(-\frac{1}{2}\sum_{i=1}^{d} x_i^2\right)$	$(0, 0, \ldots, 0)$	-1	$[-1, 0.5]^d$
Zakharov	$\sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{d} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{d} 0.5 i x_i\right)^4$	$(0, 0, \ldots, 0)$	0	$[-20, 10]^d$

Tabla 4.2: Funciones de prueba multimodades del bechmark

Funciones	**Definición**	x^*	$f(x^*)$	**Dominio**
Rosenbrock	$\sum_{i=1}^{d-1}\left[(1-x_i)^2+100\left(x_{i+1}-x_i^2\right)^2\right]$	$(1,1,\ldots,1)$	0	$[-20,10]^d$
Ackley	$-20\exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2})$ $-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(2\pi x_i\right)\right)+20+e$	$(0,0,\ldots,0)$	0	$[-20,10]^d$
Cosine Mixture	$\sum_{i=1}^{d}x_i^2-0.1\sum_{i=1}^{d}\cos\left(5\pi x_i\right)$	$(0,0,\ldots,0)$	$-0.1d$	$[-1,0.5]^d$
Whitley	$\sum_{i=1}^{d}\sum_{j=1}^{d}\left[\frac{1}{4000}\left(y_{ij}\right)^2-\cos\left(y_{ij}\right)+1\right]$	$(1,1,\ldots,1)$	0	$[-20,10]^d$
Xin-She Yang 2	$\left(\sum_{i=1}^{d}\lvert x_i\rvert\right)\exp\left(-\sum_{i=1}^{d}\sin\left(x_i^2\right)\right)$	$(0,0,\ldots,0)$	0	$[-2\pi,\pi]^d$
Alpine 1	$\sum_{i=1}^{d}\lvert x_i\sin\left(x_i\right)+0.1x_i\rvert$	$(0,0,\ldots,0)$	0	$[-20,10]^d$
Schaffer 6	$\sum_{i=1}^{d-1}\left[\frac{1}{2}+\frac{\sin^2\sqrt{x_i^2+x_{i+1}^2}-\frac{1}{2}}{\left(1+0.001\left(x_i^2+x_{i+1}^2\right)\right)^2}\right]$	$(0,0,\ldots,0)$	0	$[-20,10]^d$
Cosine Wave	$-\sum_{i=1}^{d-1}\left[\exp\left(-\frac{1}{8}y_i\right)\cos(4\sqrt{y_i})\right]$ with $y_i=x_i^2+x_{i+1}^2+\frac{1}{2}x_ix_{i+1}$	$(0,0,\ldots,0)$	$-d+1$	$[-5,3]^d$
Wavy	$1-\frac{1}{d}\sum_{i=1}^{d}\left[\cos\left(10\cdot x_i\right)\exp\left(-\frac{1}{2}x_i^2\right)\right]$	$(0,0,\ldots,0)$	0	$[-\pi,\pi/2]^d$
Griewangk	$1+\sum_{i=1}^{n}x_i^2/4000-\prod_{i=1}^{n}\cos\left(x_i/\sqrt{i}\right)$	$(0,0,\ldots,0)$	0	$[-600,300]^d$
Rastrigin	$10d+\sum_{i=1}^{d}\left[x_i^2-10\cos\left(2\pi x_i\right)\right]$	$(0,0,\ldots,0)$	0	$[-20,10]^d$
Bohachevsky	$\sum_{i=1}^{d-1}\left[x_i^2+2x_{i+1}^2-0.3\cos\left(3\pi x_i\right)-\right.$ $\left.0.4\cos\left(4\pi x_{i+1}\right)+0.7\right]$	$(0,0,\ldots,0)$	0	$[-20,10]^d$
Levy 8	$\sum_{i=1}^{d-1}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_{i+1}\right)\right]+$ $\sin^2\left(\pi y_1\right)+\left(y_d-1\right)^2$ with $y_i=1+0.25\left(x_i+1\right)$	$(-1,-1,\ldots,-1)$	0	$[-20,10]^d$

Tabla 4.3: Comparación KSG-EDA y BUMDA

Tabla 4.4: Dimensión 10

Funciones	KSG-EDA	BUMDA	Rechazo H_0
Sphere	100	100	SI
	8.39e-7 ± 1.50e-7	7.25e-7±1.73e-7	
	1.27e+4 ± 1.85e+3	1.54e+4±1.95e+2	Z =-1.03e+1
Ellipsoid	100	100	NO
	6.89e-7 ± 2.13e-7	7.40e-7±1.82e-7	
	2.69e+4 ± 6.26e+3	**1.07e+4±1.75e+2**	
Differentpowers	100	100	SI
	7.24e-7 ± 3.38e-7	5.32e-7±2.63e-7	
	4.33e+3 ± 8.44e+2	7.51e+3±2.55e+2	Z =-2.55e+1
Cigartablet	100	100	NO
	5.07e-7 ± 3.35e-7	7.74e-7±1.70e-7	
	4.88e+4 ± 9.93e+3	**1.71e+4±2.43e+2**	
Twoaxes	100	100	NO
	7.60e-7 ± 3.38e-7	7.35e-7±1.85e-7	
	4.91e+4 ± 8.96e+3	**1.65e+4±2.23e+2**	
Zakharov	**100**	0	SI**
	8.87e-7 ± 1.03e-7	9.38e+0±6.79e+0	$Z = -1$
	5.51e+4 ± 6.84e+3	1.00e+5±0.00e+0	
Ackley	100	100	NO
	8.22e-7 ± 2.17e-7	8.73e-7±1.08e-7	
	3.54e+4 ± 7.97e+3	**2.22e+4±3.09e+2**	
Rosenbrock	0	0	SI*
	5.32e+0 ± 2.04e+0	8.10e+0±7.72e-2	Z = -9.63e+0
	1.00e+5 ± 6.00e+0	1.00e+5±0.00e+0	
Griewangk	0	**100**	NO**
	1.02e-1 ± 5.10e-2	7.30e-7±2.10e-7	
	1.00e+5 ± 5.11e+0	1.70e+4±3.79e+2	
Rastrigin	**100**	37	SI**
	7.79e-7 ± 2.09e-7	6.63e-2±2.52e-1	$Z = -176$
	5.75e+4 ± 1.65e+4	3.01e+4±1.92e+4	
Bohachevsky	100	100	NO
	7.37e-7 ± 2.81e-7	7.75e-7±1.67e-7	
	1.58e+4 ± 1.64e+3	**1.47e+4±3.24e+2**	
Levy8	90	**100**	NO**
	1.35e+0 ± 3.34e+0	8.03e-7±1.49e-7	
	4.63e+4 ± 2.72e+3	1.14e+4±3.15e+2	

Tabla 4.5: Dimensión 40

Funciones	KSG-EDA	BUMDA	Rechazo H_0
Sphere	100	100	NO
	8.94e-7 ± 1.27e-7	8.78e-7 ± 9.68e-8	
	8.14e+4 ± 7.88e+3	**3.50e+4±3.60e+2**	
Ellipsoid	100	100	NO
	7.72e-7 ± 2.45e-7	8.71e-7±9.56e-8	
	1.75e+5 ± 2.53e+4	**4.96e+4±4.25e+2**	
Differentpowers	100	100	NO
	8.34e-7 ± 2.11e-7	7.63e-7±1.91e-7	
	3.11e+4 ± 4.59e+3	**7.69e+4±1.48e+3**	
Cigartablet	100	100	NO
	8.42e-7 ± 1.81e-7	9.05e-7±6.32e-8	
	1.56e+5 ± 2.09e+4	**7.76e+4±4.01e+2**	
Twoaxes	100	100	NO
	7.64e-7 ± 2.65e-7	8.60e-7±1.06e-7	
	2.91e+5 ± 4.54e+4	**7.66e+4±5.07e+2**	
Zakharov	0	0	SI*
	1.01e-2 ± 6.10e-3	3.78e+2±1.35e+2	Z = -1.98e+1
	4.00e+5 ± 7.26e+0	4.00e+5±0.00e+0	
Ackley	100	100	NO
	9.43e-7 ± 1.04e-7	9.41e-7±5.40e-8	
	2.32e+5 ± 2.14e+4	**4.92e+4±4.16e+2**	
Rosenbrock	0	0	NO*
	4.55e+1 ± 4.59e+1	**3.79e+1±7.98e-2**	
	2.32e+5 ± 2.14e+4	4.00e+5±0.00e+0	
Griewangk	50	**100**	NO**
	2.32e-2 ± 3.42e-2	8.74e-7±7.53e-8	
	4.00e+5 ± 7.83e+0	3.42e+4±3.20e+2	
Rastrigin	**44**	0	SI**
	3.39e-1 ± 6.23e-1	4.51e+0±1.82e+0	$Z = -531$
	2.32e+5 ± 1.70e+5	4.00e+5±0.00e+0	
Bohachevsky	100	100	NO
	8.99e-7 ± 1.33e-7	8.56e-7±8.54e-8	
	3.69e+5 ± 4.36e+4	**3.35e+4±3.50e+2**	
Levy8	80	**100**	NO**
	6.96e-1 ± 2.29e+0	8.40e-7±1.11e-7	
	1.28e+5 ± 1.52e+4	2.66e+4±3.02e+2	

En las tablas 4.4, 4.5, 4.7 y 4.8 aparecen 4 columnas, en la primera columna aparece el nombre de la función, en la segunda y la tercera los resultados de los algoritmos correspondientes:para cada función aparecen tres filas en la segunda y tercera columna, que corresponden a la tasa de éxito, a la media más o menos la desviación estándar de la distancia al óptimo y la media más o menos la desviación estándar de las evaluaciones de función. en la cuarta columna aparece si se rechaza la hipótesis nula o no dependiendo del caso:

- Cuando ambos algoritmos tienen una tasa de éxito del 100 % se comparan las medias del número de evaluaciones de función con el proceso descrito en la Sección 4.2, cuando la hipótesis nula se rechaza con un nivel de significancia de 0.05 se marca con un

SI y se muestra el valor Z que es menor que $-Z_{0.975} = -0.8340$. Cuando no se puede rechazarla hipótesis nula, se marca con un NO.

- Cuando ambos algoritmos tienen una tasa de éxito del 0 % se comparan las medias de la distancia al óptimo con el proceso descrito en la Sección 4.2, cuando la hipótesis nula se rechaza a un nivel de significancia de 0.05 se marca con un SI* y se muestra el valor Z el cual es menor que $-Z_{0.975} = -0.8340$.Cuando no se puede rechazar la hipótesis nula, se marca con un NO*.

- Cuando los algoritmos tienen tasas de éxito diferentes, se compara la tasa de éxito con el procedimiento descrito en 4.1, cuando la hipótesis nula se rechaza a un nivel de significancia de 0.05 se marca con un SI** y se muestra el valor Z el cual es menor que $-Z_{0.975} = -0.8340$. Cuando no se puede rechazar la hipótesis nula, se marca con un NO**.

En las tablas 4.4, 4.5 se muestra que el KSG-EDA y el BUMDA se comportan de manera similar, sin embargo, el BUMDA, cuando ambas tasas de éxito son del 100 % el BUMDA utiliza menos evaluaciones de función que el KSG-EDA.

En las tablas 4.7 y 4.8 Se muestra que el KSG-EDA y el ∇_d-**NEDA** se comportan de manera similar en las funciones unimodales. En cuanto a las funciones multimodales como la función Acley, Cosine Mixture, Whitley, Xin-she Yan 2, Shaffer 6, Cosine Wave y Wavy el KSG-EDA obtiene mejores resultados.

Tabla 4.6: Funciones de prueba unimodales y multimodades del bechmark

Tabla 4.7: Dimensión 30

Funciones	KSG-EDA	∇_d-NEDA	Rechazo H_0
Sphere	100	100	SI
	8.89e-9 ± 1.22e-9	9.20e-9 ± 7.42e-10	
	1.03e+5 ± 1.19e+4	1.11e+5 ± 2.83e+4	Z =-1.97e+0
Different powers	100	100	NO
	6.98e-9 ± 2.54e-9	8.89e-9± 9.28e-10	
	3.4e+4 ± 4.53e+3	**1.14e+4±1.19e+3**	
Mishra 2	16	**100**	NO**
	6.42 e-8 ± 1.74e-7	9.46e-9±4.47e-10	
	2.98e+5 ± 5.46e+3	5.07e+3±1.76e+2	
Ellipsoid	100	100	NO
	8.42 e-9 ± 2.14e-9	9.07e-9±8.30e-10	
	1.86e+5 ± 2.82e+4	**1.88e+4±1.48e+3**	
Tablet	100	100	NO
	8.64e-9 ± 1.45e-9	8.87e-9±8.96e-10	
	1.22e+5 ± 1.47e+4	**1.26e+4±8.26e+2**	
Cigartablet	100	100	NO
	8.4e-9 ± 2.21e-9	8.86e-9±8.59e-10	
	1.77e+5 ± 1.84e+4	**1.58e+4±7.90e+2**	
Twoaxes	82	**100**	-
	1.42e-8 ± 2.45e-8	9.03e-9±8.46e-10	
	2.61e+5 ± 2.99e+4	1.92e+4±1.66e+3	
Parabolicridge	100	100	NO
	8.92e-9 ± 1.67e-9	8.91e-9±8.30e-10	
	1.59e+5 ± 2.21e+4	**1.32e+4±5.32e+2**	
Sharpridge	0	**100**	NO**
	2.54e-3 ± 9.67e-3	9.46e-9±4.60e-10	
	3.0e+5 ± 4.89e+0	2.20e+4±6.04e+2	
Exponential	100	100	NO
	8.70e-9 ± 1.70e-9	8.84e-9±1.03e-9	
	8.59e+4 ± 1.13e+4	**7.71e+3±2.69e+2**	
Rosenbrock	0	0	NO*
	2.73e+1 ± 3.55e+1	**1.12e+0±1.81e+0**	
	3.00e+5 ± 0.00e+0	3.00e+5±0.00e+0	
Ackley	**90**	84	-
	4.75e-8 ± 1.78e-7	2.64e-1±8.59e-1	
	2.96e+5 ± 7.45e+3	6.31e+4±1.04e+5	
Cosine Mixture	**100**	6	-
	8.21e-9 ± 1.71e-9	4.82e-1±2.75e-1	
	1.04e+5 ± 1.37e+4	2.83e+5±6.99e+4	
Whitley	6	4	-
	1.97e+01 ± 1.85E+01	3.11e+2±1.74e+2	
	2.93e+5 ± 4.25e+4	2.89e+5±5.63e+4	
Xin-she yang 2	100	100	SI
	6.83e-9 ± 1.86e-9	6.45e-9±2.34e-9	
	1.12e+3 ± 3.34e+2	1.95e+4±2.27e+4	Z =-5.72e+0
Alpine1	0	**100**	NO**
	1.24e-6 ± 2.80e-6	9.43e-9±4.06e-10	
	3.00e+5 ± 5.12e+0	1.75e+4±9.76e+2	
Schwefel 226	0	0	NO*
	5.16e+3 ± 1.09e+3	**2.17e+3±6.49e+2**	
	3.00e+5 ± 0.00e+0	3.00e+5±0.00e+0	
Schaffer 6	0	0	SI*
	2.85e-2 ± 1.01e-2	6.27e+0±1.27e+0	Z =-3.48e+1
	3.00e+5 ± 5.38e+0	3.00e+5±0.00e+0	
Cosine Wave	0	0	SI*
	9.07e+0 ± 1.64e+0	1.00e+1±1.66e+0	Z =-2.82e+0
	3.00e+5 ± 5.04e+0	3.00e+5±0.00e+0	
Wavy	**100**	0	SI**
	8.55 e-9 ± 1.83 e-9	3.22e-1±1.58e-1	
	1.80 e+5 ± 4.53 e+4	3.00e+5±0.00e+0	

Tabla 4.8: Dimensión 50

Funciones	KSG-EDA	∇_d-NEDA	Rechazo H_0
Sphere	100	100	SI
	8.40e-9 ± 2.00e-9	9.44e-9±5.65e-10	
	2.15e+5 ± 2.14e+4	2.89e+5±5.90e+4	Z =-8.34e+0
Differentpowers	100	100	NO
	8.20e-9 ± 2.25e-9	9.41e-9±4.80e-10	
	7.14e+4 ± 1.03e+4	**2.57e+4±1.71e+3**	
Mishra2	0	**100**	-
	1.45e-6 ± 7.19e-6	9.55e-9±3.91e-10	
	5.00e+5 ± 0.00e+	1.01e+4±3.86e+2	
Ellipsoid	100	100	NO
	8.50e-9 ± 2.26e-9	9.35e-9±7.72e-10	
	3.83e+5 ± 5.17e+4	**3.72e+4±2.49e+3**	
Tablet	100	100	NO
	9.08e-9 ± 1.55e-9	9.21e-9±7.19e-10	
	2.42e+5 ± 2.76e+4	**2.29e+4±1.30e+3**	
Cigartablet	100	100	NO
	8.83e-9 ± 1.49e-9	9.25e-9±6.87e-10	
	3.13e+5 ± 3.01e+4	**2.87e+4±1.07e+3**	
Twoaxes	6	**100**	-
	9.45e-8 ± 1.56e-7	9.12e-9±8.78e-10	
	4.99e+5 ± 7.48e+3	3.59e+4±2.55e+3	
Parabolicridge	100	100	NO
	8.99e-9 ± 1.26e-9	9.32e-9±6.31e-10	
	3.15e+5 ± 4.10e+4	**2.39e+4±9.87e+2**	
Sharpridge	0	**100**	-
	7.39e-3 ± 1.96e-2	9.66e-9±3.31e-10	
	5.00e+5 ± 4.98e+0	3.97e+4±1.34e+3	
Exponential	100	100	NO
	8.85e-9 ± 1.59e-9	9.26e-9±7.84e-10	
	1.76e+5 ± 1.83e+4	**1.41e+4±5.23e+2**	
Rosenbrock	0	0	NO*
	8.70e+1 ± 5.79e+1	2.81e+0±4.79e+0	
	5.00e+5 ± 0.00e+0	5.00e+5±0.00e+0	
Ackley	0	**72**	-
	1.81e-7 ± 2.02e-7	3.74e-1±6.63e-1	
	5.00e+5 ± 4.67e+0	1.63e+5±2.12e+5	
Cosine Mixtute	**100**	0	-
	8.97e-9 ± 1.14e-9	8.75e-1±3.67e-1	
	2.03e+5 ± 2.19e+4	5.00e+5±0.00e+0	
Whitley	0	**2**	-
	5.97e+1 ± 3.57e+1	8.03e+2±5.01e+2	
	5.00e+5 ± 5.22e+0	4.91e+5±6.28e+4	
Xinsheyang2	100	100	SI
	4.67e-9 ± 3.40e-9	4.54e-9±3.00e-9	
	5.18e+2 ± 2.58e+2	1.02e+3±7.30e+2	Z =-4.58e+0
Alpine1	0	**100**	-
	1.33e-5 ± 2.28e-5	9.39e-9±4.14e-10	
	5.00e+5 ± 4.77e+0	3.19e+4±2.24e+3	
Schwefel226	0	0	NO*
	8.41e+3 ± 1.71e+3	**4.12e+3±9.22e+2**	
	5.00e+5 ± 5.35e+0	5.00e+5±0.00e+0	
Schaffer6	0	0	SI*
	3.59e-2 ± 1.33e-2	1.39e+1±5.71e-1	Z =-1.72e+2
	5.00e+5 ± 5.25e+0	5.00e+5± 0.0e+0	
Cosine Wave	0	0	NO*
	1.65e+1 ± 2.55e+0	1.71e+1±2.13e+	
	5.00e+5 ± 5.15e+0	5.00e+5±0.00e+0	
Wavy	**70**	0	-
	5.03e-4 ± 2.16e-3	6.36e-1±5.63e-2	
	4.08e+5 ± 7.67e+4	5.00e+5±0.00e+0	

CAPÍTULO 5

Conclusiones

En esta tesis se propuso el KSG-EDA el cual es un EDA que utiliza información de un gradiente estocástico para actualizar los parámetros de la distribución de búsqueda. En este capítulo se escriben las conclusiones del trabajo comparando los tres algoritmos expuestos: KSG-EDA, ∇_d-NEDA y BUMDA así como el trabajo futuro

5.1. Comparación entre KSG-EDA, ∇_d-NEDA y BUMDA

Los algoritmos KSG-EDA y ∇_d-NEDA son EDAs basados en gradiente. Sin embargo el ∇_d-NEDA es un algoritmo que calcula el gradiente sobre el espacio de búsqueda de la función a optimizar, mientras que el KSG-EDA lo hace sobre el espacio de los parámetros de la distribución de búsqueda. Esto hace que desde el planteamiento del desarrollo de los algoritmos, estos sean completamente diferentes, sin embargo ambos algoritmos son competitivos y muestran que la fusión entre el gradiente y los EDAs dan muy buenos resultados.

Por otro lado, el BUMDA y el KSG-EDA parten de una idea general muy parecida: minimizar la divergencia KL para obtener los parámetros de una distribución Normal que se

parezca a la Boltzmann, en la Tabla 5.1 se muestran algunas de las múltiples características que los hacen algoritmos completamente diferentes.

Principal diferencia de estos algoritmos se da en la función de costo que minimizan. En la divergencia $KL_{\mathcal{P},\mathcal{Q}}$ generalmente $\mathcal{P}$ (Ecuación (3.1)) representa la 'verdadera' distribución de los datos, mientras que $\mathcal{Q}$ (Ecuación (3.2)) generalmente representa una teoría o un modelo. Como el BUMDA minimiza la divergencia $KL_{\mathcal{Q},\mathcal{P}}$ supone que el modelo es la distribución Normal, mientras que el KSG-EDA supone que el modelo es distribución Boltzmann. Partiendo de este punto, las expresiones que resultan para μ y para σ al minimizar $KL_{\mathcal{Q},\mathcal{P}}$ y $KL_{\mathcal{P},\mathcal{Q}}$ son diferentes.

Características	**KSG-EDA**	**BUMDA**
Parámetros	$\beta, N, n, s, \alpha_\mu, c_\sigma$	N
Función de costo a minimizar	$KL_{\mathcal{P},\mathcal{Q}}$	$KL_{\mathcal{Q},\mathcal{P}}$
# individuos para calcular el gradiente	$n << N$	N
Método de minimización	SGD	Máxima verosimilitud
Selección	$N - s$ Mejores + s Nuevos	Proceso de truncamiento
Criterio de paro	Máximo de iteraciones o distancia al mínimo $<$ cota	Máximo de iteraciones o Varianza $<$ cota

Tabla 5.1: Comparación KSG-EDA y BUMDA

5.2. Trabajo Futuro

El KSG-EDA minimiza la divergencia $KL_{\mathcal{P},\mathcal{Q}}$, sería interesante usar otras funciones para medir la similitud entre dos distribuciones como por ejemplo la distancia Bhattacharyya [6] o la distancia [5]. El KSG-EDA puede tener varios algoritmos derivados, por ejemplo implementarlo con variantes del SGD: el ADAM [21] Adagrad [13], el Adadelta [42], entre muchos otros.

EL KSG-EDA tiene muchos parámetros fijos, sería interesante que dependiendo de la dimensión o de incluso el tipo de problemas (unimodales o multimodales), se haga una adaptación de los parámetros. También sería interesante desarrollar una versión del KSG-EDA

para funciones multiobjetivo

APÉNDICE A

Representación de una distribución Normal bivariada como una elipse en 2D

En este capítulo se presenta un desarrollo detallado de cómo se grafica una matriz de convarianza desde su deducción matemática hasta un seudocódigo para su implementación.

¿Cómo dibujar una matriz de covarianza?

A.1. Variables independientes

Cuando las variables aleatorias se distribuyen $\mathcal{N}(\mu, \Sigma)$ y son independientes la matriz Σ es una matriz diagonal.

La ecuación de la elipse con los ejes principales son paralelos al eje x y y, con el eje mayor de longitud $2a$ y el eje menor de longitud $2b$ está definida por la siguiente ecuación

$$\left(\frac{X}{a}\right)^2 + \left(\frac{Y}{b}\right)^2 = 1$$

En este caso la longitud de los ejes está dado por la desviación estándar σ_x y σ_y y la

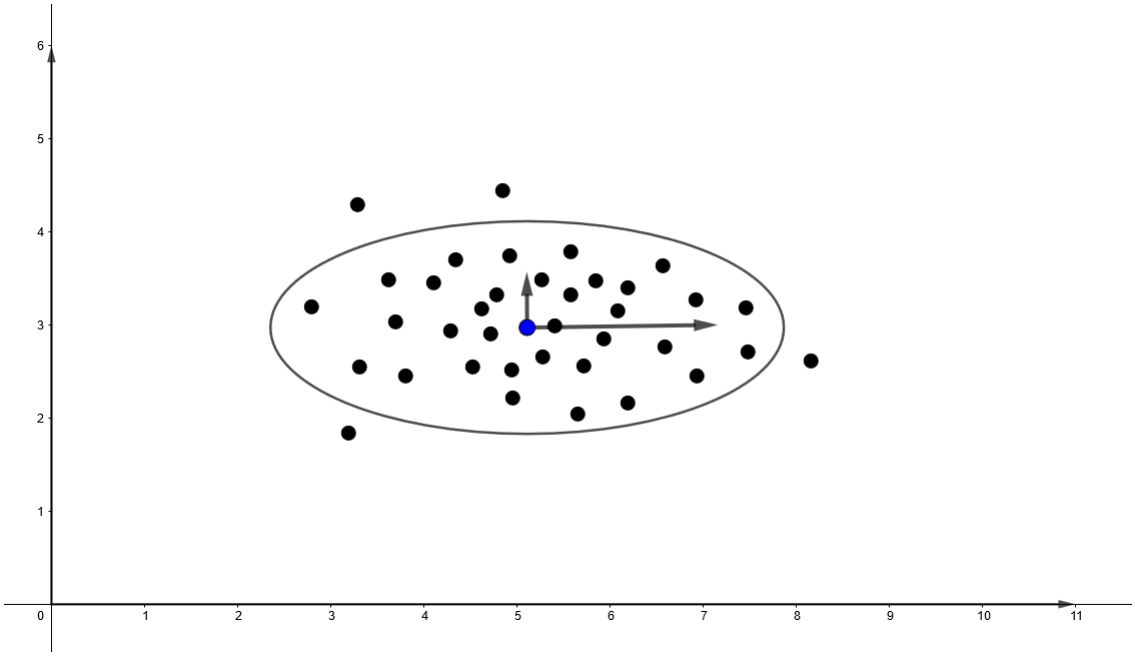


Figura A.1: Elipse con ejes paralelos a x y y:

ecuación anterior se puede reescribir como

$$\left(\frac{X}{\sigma_x}\right)^2 + \left(\frac{Y}{\sigma_y}\right)^2 = S \tag{A.1}$$

donde S define la escala de la elipse y puede ser cualquier número (por ejemplo 1). La pregunta es ¿cómo elegir S de modo que la elipse resultante represente el nivel de confianza elegido, digamos $95\,\%$?

$$95\,\% \leftrightarrow S = 5.991$$

En nuestro conjunto de datos 2D están muestreados de una Gaussiana con 0 covarianza, esto significa que los ejes son normalmente distribuidos, también.

La parte izquierda de la Ecuación A.1 representa la suma de las variables independientes distribuidas.

La suma de los cuadrados de una variable que se distribuye normal χ^2 la cual está definida en términos de grados de libertad las variables desconocidas, en nuestro caso son 2 y por lo tanto 2 grados de libertad.

$$P(S < 5.991) = 1 - 0.05 = 0.95)$$

$$\left(\frac{X}{\sigma_x}\right)^2 + \left(\frac{Y}{\sigma_y}\right)^2 = 5.991 \tag{A.2}$$

así la elipse tiene eje mayor $2\sigma_x\sqrt{5.991}$ y eje menor $2\sigma_y\sqrt{5.991}$.

La magnitud de los ejes de la elipse depende de la varianza de los datos

En general:

Cuando las matrices no son diagonales:

- Los eigenvalores representan la propagacón de los datos en la dirección de los eigenvectores.
- Los eigenvalores representan la varianza de los datos en dirección de los eigenvectores.

En este caso los eigenvectores representan la dirección de mayor propagación de los datos.

Similar al caso anterior, para crear una elipse con el $95\,\%$ de confianza, el eje mayor mediría $2\sqrt{5.991\lambda_i}$ donde $\lambda_i$son los eigenvalores.

Para obtener la dirección calculamos el ángulo del mayor eigenvector hacia el eje X (eje mayor).

$$\alpha = \text{arctan}(\frac{V_1(1)}{V_1(2)})$$

La matriz de rotación

$$R = \begin{bmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{bmatrix}$$

Bibliografía

[1] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Inc., USA, 1996.

[2] Lee Bain and Max Engelhardt. Introduction to probability and mathematical statistics. *Biometrics*, 49:673, 06 1993.

[3] Emre Barut and Warren B. Powell. Optimal learning for sequential sampling with non-parametric beliefs. *J. of Global Optimization*, 58(3):517–543, March 2014.

[4] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies –a comprehensive introduction. *Natural Computing: An International Journal*, 1(1):3–52, May 2002.

[5] A. BHATTACHARYYA. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta math. Soc.*, 35:99–109, 1943.

[6] A. Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 7(4):401–406, 1946.

[7] P.A.N. Bosman and D. Thierens. Continuous iterated density estimation evolutionary algorithms within the idea framework. *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, 01 2000.

[8] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[9] A. Cauchy. Methode generale pour la resolution des systemes d'equations simultanees. *C.R. Acad. Sci. Paris*, 25:536–538, 1847.

[10] R. Cheng and M. Gen. Parallel machine scheduling problems using memetic algorithms. In *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929)*, volume 4, pages 2665–2670 vol.4, 1996.

[11] Carlos A. Coello Coello. An introduction to evolutionary algorithms and their applications. In Félix F. Ramos, Victor Larios Rosillo, and Herwig Unger, editors, *Advanced Distributed Systems*, pages 425–442, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[12] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc V. Le, and Andrew Y. Ng. Large scale distributed deep networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1223–1231. Curran Associates, Inc., 2012.

[13] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[14] David E. Goldberg and John H. Holland. Genetic algorithms and machine learning. *Mach. Learn.*, 3(2–3):95–99, October 1988.

[15] Nikolaus Hansen. The cma evolution strategy: A tutorial, 2005.

[16] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):287–297, 1999.

[17] Magnus R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.

[18] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[19] William C Horrace. Some results on the multivariate truncated normal distribution. *Journal of Multivariate Analysis*, 94(1):209–221, 2005.

[20] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, 09 1952.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[22] Jayesh H Kotecha and Petar M Djuric. Gibbs sampling approach for generation of truncated multivariate gaussian random variables. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 3, pages 1757–1760. IEEE, 1999.

[23] John R. Koza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, USA, 2003.

[24] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.

[25] Jose Larrañaga, Pedro y Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Genetic algorithms and evolutionary computation ; 2, 01 2002.

[26] Wali Khan Mashwani, Abdullah Khan, Atila Göktaş, Yuksel Akay Unvan, Ozgur Yaniay, and Abdelouahed Hamdi. Hybrid differential evolutionary strawberry algorithm for real-parameter optimization problems. *Communications in Statistics - Theory and Methods*, 0(0):1–14, 2020.

[27] Heinz Mühlenbein, Thilo Mahnig, Pentti Kanerva, and Hideki Asoh. Evolutionary computation and beyond. 2001.

[28] Mario Iván Jaen Márquez. A Univariate Boltzmann based Estimation of Distribution Algorithm Using the Natural Gradient for Updating the Parameters. pages 53,54, Febrero 2016.

[29] Heinz Mühlenbein. *Convergence Theorems of Estimation of Distribution Algorithms*, volume 14, pages 91–108. 02 2012.

[30] Martin Pelikan. Probabilistic model-building genetic algorithms. In *Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '08, page 2389–2416, New York, NY, USA, 2008. Association for Computing Machinery.

[31] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing*. Cambridge University Press, USA, 1992.

[32] Mariano Rivera. *Descenso de Gradiente y Variaciones Sobre el Tema*, 2018. http://personal.cimat.mx:8181/ mrivera/cursos/optimizacion/descenso_grad_estocastico/descenso_grad_estocastico.html (visitado el 3 de julio de 2020).

[33] Sutton Robbins, Herbert y Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[34] Sebastian Ruder. An overview of gradient descent optimization algorithms. https://ruder.io/optimizing-gradient-descent/index.html#fn:6, 2016.

[35] R. Salomon. Evolutionary algorithms and gradient search: similarities and differences. *IEEE Transactions on Evolutionary Computation*, 2(2):45–55, 1998.

[36] Ignacio Segovia-Dominguez and Arturo Hernández Aguirre. Building multivariate density functions based on promising direction vectors. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013*, pages 1702–1709. IEEE, 2013.

[37] D. Tamayo-Vera, A. Bolufé-Röhler, and S. Chen. Estimation multivariate normal algorithm with thresheld convergence. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3425–3432, 2016.

[38] S. Ivvan Valdez, Arturo Hernández, and Salvador Botello. A boltzmann based estimation of distribution algorithm. *Information Sciences*, 236:126 – 137, 2013.

[39] Brian Wesolowski and Dorothy Musselwhite Thompson. Normal distribution. 01 2018.

[40] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(27):949–980, 2014.

[41] Léon Bottou y Frank E. Curtis and Jorge Nocedal. Optimization methods for large-scale machine learning, 2016.

[42] Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012.