Notas Escuela de Verano 2022

Alatorre Zamora, Miguel Ángel
miguel.azamora@academicos.udg.mx

Becerra López, Fernando Ignacio
ignacio.becerra@academicos.udg.mx

Fregoso Becerra, Emilia
emilia.fbecerra@academicos.udg.mx

Guerrero Arroyo, Edgar Alejandro
edgar.guerreo@academicos.udg.mx

Licea Salazar, Juan Antonio
antonio.licea@academicos.udg.mx

Palafox González, Abel
abel.palafox@academicos.udg.mx

1 de julio de 2022

Índice general

Capítulo 1

Introducción

1.1. Aplicaciones reales de problemas de regresión y clasificación

El aprendizaje automático nace cuando los problemas a resolver por las ciencias son de alta complejidad y cuando poseen una enorme cantidad de datos. De acuerdo a Pedregosa et al (2011), existen al menos cuatro tipos de problemas que resuelve el aprendizaje automático: a) Clasificación; b) Regresión; c) Agrupación, y d) Reducción de dimensionalidad. Estos problemas suelen etiquetarse también como problemas de machine learning. Es posible que los problemas de clasificación y regresión sean de los más comunes en el aprendizaje automático. En el caso de los problemas de clasificación, el resultado que se busca obtener es una clase entre un número limitado de clases, mediante la predicción o etiquetado de objetos sobre un conjunto de clases prefijadas. Debe aclararse que la palabra "clase" se refiere a categorías arbitrarias, que dependen del problema. Para el caso de los problemas de regresión, el resultado que se busca obtener es un valor numérico, dentro de un conjunto infinito de posibles resultados. Existen técnicas que son específicas para la clasificación y otras que lo son para la regresión, pero a pesar de ello, la mayoría de las técnicas funcionan con ambos tipos de problemas (Martínez Heras, 2020). La técnica de regresión logística es un motivo de confusión frecuente: el nombre hace pensar en problemas de regresión. Sin embargo, esta técnica solo funciona en problemas de clasificación. Un tipo clásico de un problema de clasificación es cuando se quiere detectar si un correo es spam o no. En este caso solamente hay dos clases, y el algoritmo adecuado deberá decidir a qué clase pertenece. Otros ejemplos comunes o típicos son también aquellos que se responden con un "SI" o un "NO": a) ¿Comprarán mis clientes este producto? B) ¿Es este comportamiento una

anomalía?, c) ¿Un tumor es maligno o benigno?, etc. Algunas técnicas que se emplean cotidianamente para resolver problemas de clasificación son: regresión logística, máquinas de vectores de soporte, árboles de decisión, bosques aleatorios, redes neuronales y aprendizaje profundo. Los problemas de regresión, en cambio, pueden vincularse con el mundo de las predicciones. Y precisamente, en la regresión se predice o se estima un valor, que podría ser una cantidad pecuniaria o un tiempo: En cuanto se podría vender un bien, cuantos productos se pueden vender, cuánto tiempo puede durar un empleado en un trabajo, cuanto tiempo tardaría un vehículo para llegar a un punto, son algunos ejemplos de cuestiones a resolver con regresión. Entre las técnicas de aprendizaje automático que se pueden emplear en regresión están: regresión lineal y regresión no lineal, máquinas de vectores de soporte, árboles de decisión, bosques aleatorios, redes neuronales y aprendizaje profundo. Puede observarse que prácticamente son las mismas técnicas que se emplean para clasificación.

1.2. Problemas de clasificación binaria: una clase. Función de activación logística

Los problemas de clasificación binaria son relativamente simples, pues solamente se tiene dos clases, y el algoritmo debe decidir por una de ellas. En el proceso de evolución de las redes neuronales, las funciones que gobiernan el comportamiento de las neuronas artificiales se conocen como funciones de activación. Estas funciones transforman la combinación de entradas, pesos y sesgos. Cuando una neurona artificial pasa de un valor distinto de cero a otro, se dice que se ha activado. Las funciones de activación se usan para propagar la salida de los nodos de una capa hacia la siguiente capa. Las funciones más importantes usadas como funciones de activación son la familia de funciones sigmoideas. Una de estas es la función logística, que convierte variables independientes de rango casi infinito en probabilidades simples entre [0,1] (1.1):

$$\begin{cases} \phi(x) = \frac{1}{1+e^{-x}}, & x \in \mathbb{R}, \\ Im(\phi) \in [0,1], \end{cases} \tag{1.1}$$

1.3. Problema de clasificación múltiple: un número discreto de clases. Función de activación softmax

La función softmax (1.2) es una generalización de la regresión logística que se puede aplicar a datos continuos, especialmente para sistemas de clasificación multinomial, por lo que se considera como el recurso principal utilizado en las capas de salida de un clasificador. Esta función de activación calcula la distribución de probabilidades del evento sobre 'n' eventos diferentes, y devuelve esta distribución de cada una de las clases existentes en el modelo. En términos generales, esta función calculará las probabilidades de cada clase objetivo sobre todas las clases-objetivo posibles. Luego, las probabilidades calculadas serán útiles para determinar la clase objetivo para las entradas dadas. La principal ventaja de usar Softmax es que se tiene un rango de probabilidades de salida de 0 a 1, y la suma de todas las probabilidades será igual a uno.

$$\phi(\mathbf{x}) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}, \quad i = 1, 2, \ldots, k. \tag{1.2}$$

La fórmula calcula la exponencial del valor de entrada dado y la suma de los valores exponenciales de todos los valores en las entradas. Luego, la relación de la exponencial del valor de entrada y la suma de los valores exponenciales es la salida de la función Softmax. Este tipo de función de activación es muy utilizado en el modelo de regresión logística de clasificación múltiple y en diferentes niveles de capa de cara a la construcción de redes neuronales.

1.4. Problema de regresión: un número continuo de clases. Función de activación lineal

Las transformadas lineales son básicamente representadas por la función de identidad, donde la variable dependiente tiene una relación directa y proporcional con la variable independiente. En términos prácticos, significa que la función pasa la señal sin cambios, y se puede describir mediante:

$$\begin{cases} \phi(x) = Wx, & x \in \mathbb{R}, \\ Im(\phi) \in \mathbb{R}. \end{cases} \tag{1.3}$$

Capítulo 2

Modelo del Perceptrón Multicapa (MLP)

2.1. Definición de un MLP

El perceptrón de Rosenblatt (figura 2.1), también llamado perceptrón simple, puede verse como la forma más simple de red neuronal. Podemos decir que la neurona propuesta por Rosenblatt consiste en realizar una combinación lineal de las entradas con los pesos sinápticos $w_i, i = 1, ..., n$ sumada con un sesgo o bias b, a la que se aplica una función de activación f que produce la salida y.

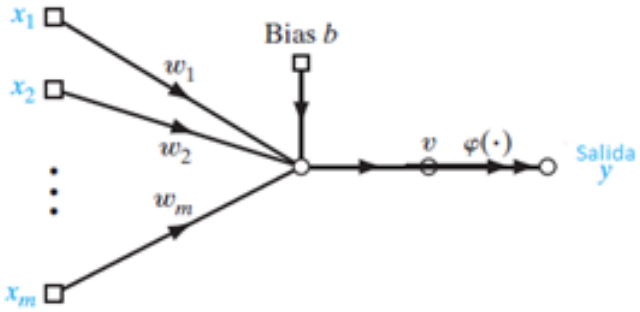

Figura 2.1: Perceptrón

Visto de otra forma siendo $W = (w_1, w_2, \ldots, w_n)$ el vector de los pesos sinápticos, $x = (x_1, x_2, \ldots, x_n)$ el vector de entradas y b el sesgo del modelo, la salida y se calcula como:

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.1}$$

A la función f se le llama función de activación. En el caso del perceptrón simple, se utiliza la función escalón unitario (o de Heaviside)

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.2}$$

lo que muestra el objetivo del perceptrón de hacer una clasificación binaria. Podemos observar que la frontera entre las entradas que se considerarán 0 o 1 de salida se da cuando

$$\sum_{i=1}^{m} w_i x_i + b = 0 \tag{2.3}$$

Este hiperplano es llamado la frontera de decisión, y divide el espacio n-dimensional en dos regiones de clasificación (figura 2.2).

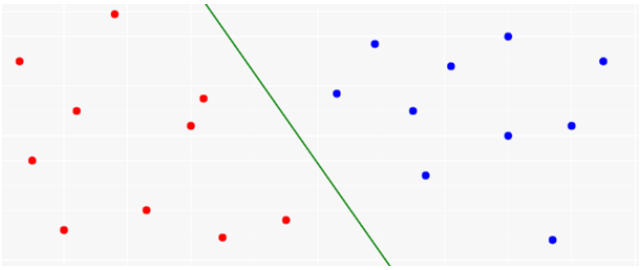

Figura 2.2: Frontera de desición

2.1.1. Red neuronal multicapa

Se puede pensar que el perceptrón simple es una red neuronal de un sólo elemento. El perceptrón multicapa (MLP) nace al conectar varios de estos elementos (neuronas) entre sí. Se organizan en forma de capas o grupos de neuronas que se conectan con otras capas unidireccionalmente. Las entradas se consideran una capa (de entrada), y se asigna tantas neuronas como valores en los vectores de salida se requieran (capa de salida). El resto de neuronas se organizan en capas (llamadas ocultas). En la figura 2.3 se puede observar un ejemplo con dos capas ocultas.

En el caso del MLP, la función de activación es cualquier función de tipo sigmoide (ver figura 2.4). Un ejemplo puede ser la función logística

$$f(x) = \frac{1}{1 + e^{-ax}} \tag{2.4}$$

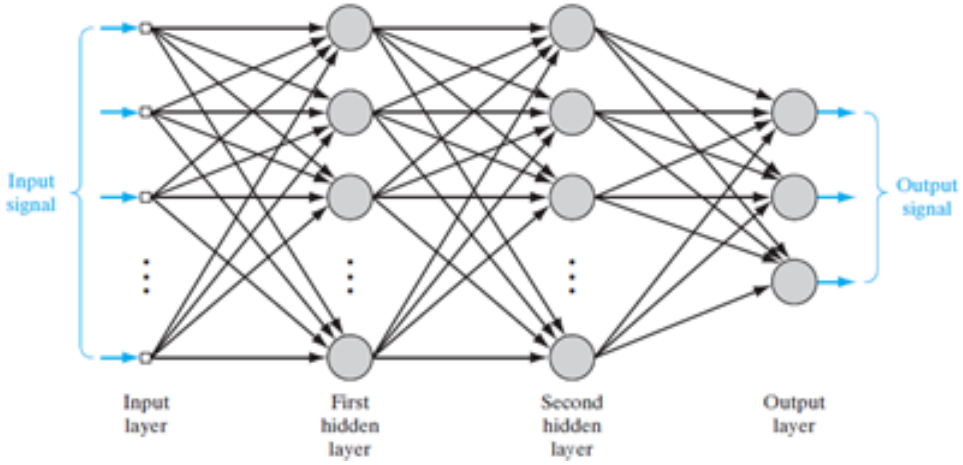

Figura 2.3: Perceptrón multicapa

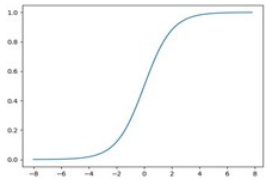

Figura 2.4: Función tipo sigmoide

De manera matemática podemos decir que la respuesta de cada neurona que conforman la red está dado por:

$$f(\sum_j w_{ij}^k x_j + b_j^k) \tag{2.5}$$

donde i refiere a la i-ésima neurona de la k-ésima capa. Podemos escribir lo anterior de forma matricial como

$$f(WX + B) \tag{2.6}$$

de forma que el modelo general F se puede ver como la composición

$$F(X) = \sigma(W^n\sigma(...\sigma(W^2\sigma(W^1X + B^1) + B^2)... + B^{n-1}) + B^n) \tag{2.7}$$

Algo importante es el generar un que en este tipo de modelos nos permite crear fronteras de decisión en las que no necesariamente esta sea una función línea.

2.2. Problema directo

El problema directo o *propagación hacia adelante* consiste en *evaluar* nuestro MLP. En otras palabras, asumimos conocidos los parámetros de la red $\{\mathbf{W}, \mathbf{b}\}$ y los valores de entrada $\mathbf{X}$, para después calcular la salida $\hat{\mathbf{y}}$. De esta manera, lo que tenemos que hacer es evaluar la función

$$MLP(\mathbf{X}; \mathbf{W}, \mathbf{b}) \to \hat{\mathbf{y}}. \tag{2.8}$$

Esta evaluación hacia adelante nos será de utilidad tanto al estimar los parámetros de nuestra red como para evaluar nuevos puntos en nuestro modelo ya entrenado, lo último con el fin de obtener su clasificación/valor correspondiente.

2.2.1. Planteamiento del problema directo

Consideremos un MLP con L capas ($L-2$ capas ocultas). Denotemos por n_ℓ el número de neuronas en la ℓ-ésima capa, donde $\ell = 1, \ldots, L-1$. Partimos de que las entradas $\mathbf{a_1}$ son conocidas por ser precisamente los valores de entrada $\mathbf{X}$, es decir $\mathbf{a}_1 = \mathbf{X}$. Para continuar, consideremos la capa de entrada $\mathbf{a}_k$ de nuestro MLP. Para obtener las entradas $\mathbf{a}_{\ell+1}$ de la capa $(\ell+1)$-ésima lo que debemos hacer es evaluar la combinación lineal de pesos y sesgos con los valores de la capa anterior $\mathbf{a}_\ell$, y evaluarlos posteriormente en nuestra función de activación. Esto es:

$$\begin{cases} \mathbf{z}_{\ell+1} = \mathbf{W}_{\ell+1}\mathbf{a}_\ell + \mathbf{b}_{\ell+1} \\ \mathbf{a}_{\ell+1} = \sigma_{\ell+1}(\mathbf{z}_{\ell+1}). \end{cases} \tag{2.9}$$

Es importante tomarse un momento para realizar algunas aclaraciones importantes:

I) Asumimos que hay m datos con n_1 características cada uno. Esto quiere decir que $\mathbf{a}_1 = \mathbf{X} \in \mathbb{R}^{n_1 \times m}$.

II) La expresión $\mathbf{W}_{\ell+1}\mathbf{a}_\ell$ es un producto de matrices en donde $\mathbf{W}_{\ell+1} \in \mathbb{R}^{n_{\ell+1} \times n_\ell}$ y $\mathbf{a}_\ell \in \mathbb{R}^{n_\ell \times m}$, por lo que su producto vive en el espacio $\mathbb{R}^{n_{\ell+1} \times m}$.

III) La matriz de sesgos $\mathbf{b}_{\ell+1}$ contiene un sesgo diferente por cada una de las $n_{\ell+1}$ neuronas de la capa $\ell+1$. Dichos valores se pueden pensar como un vector de $n_{\ell+1}$ sesgos. Sin embargo, para que la operación $\mathbf{W}_{\ell+1}\mathbf{a}_\ell + \mathbf{b}_{\ell+1}$ pueda realizarse, ambos sumandos deben de vivir en el mismo espacio $\mathbb{R}^{n_{\ell+1} \times m}$. De este modo, el vector de $n_{\ell+1}$ sesgos se escribe como matriz

repitiendo sus valores para los m datos. Luego entonces, $\mathbf{b}_{\ell+1}$ tiene la forma siguiente:

$$\mathbf{b}_{\ell+1} = \begin{pmatrix} b_{\ell+1}^1 & b_{\ell+1}^1 & \cdots & b_{\ell+1}^1 \\ b_{\ell+1}^2 & b_{\ell+1}^2 & \cdots & b_{\ell+1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ b_{\ell+1}^{n_{\ell+1}} & b_{\ell+1}^{n_{\ell+1}} & \cdots & b_{\ell+1}^{n_{\ell+1}} \end{pmatrix} \in \mathbb{R}^{n_{\ell+1} \times m}. \tag{2.10}$$

Dicho de otras palabras, los valores de los sesgos $\mathbf{b}_{\ell+1}$ son intependientes de los datos, y se deben de repetir en forma de renglón para realizar las operaciones matriciales correspondientes.

IV) Con lo anterior, resulta claro que $\mathbf{z}_{\ell+1} \in \mathbb{R}^{n_{\ell+1} \times m}$.

V) Finalmente, la evaluación de $\mathbf{z}_{\ell+1}$ en las funciones de activación $\sigma_{\ell+1}$ es realizada entrada por entrada

$$\begin{cases} \sigma_{\ell+1}^j : \mathbb{R} \to \mathbb{R}, \\ a_{\ell+1}^{i,j} = \sigma_{\ell+1}^j(z_{\ell+1}^{i,j}), \quad i = 1, 2 \ldots, m, \quad j = 1, 2 \ldots, n_{\ell+1}, \end{cases} \tag{2.11}$$

y de este modo hemos calculado $\mathbf{a}_{\ell+1}$, el cual tiene las mismas dimensiones que $\mathbf{z}_{\ell+1}$.

VI) Aunque las funciones de activación $\sigma_{\ell+1}^j$ pueden ser diferentes de una neurona a otra y de una capa a otra, en la práctica se asumen que todas las funciones de activación de una misma capa son de la misma forma. Es decir, $\sigma_{\ell+1}^j = \sigma_{\ell+1}^k$ para todos $j, k = 1, 2, \ldots, n_{\ell+1}$. De esta manera, se hace referencia a las funciones de activación por capas (i.e. "las funciones de activación de la capa ℓ").

El problema directo se puede pensar como la composición sucesiva de funciones de activación de las combinaciones lineales de los valores de las capas. Esto es:

$$\begin{aligned} MLP(\mathbf{X}; \mathbf{W}, \mathbf{b}) &= \mathbf{a}_L \\ &= \sigma_L(\mathbf{W}_L \mathbf{a}_{L-1} + \mathbf{b}_L) \\ &= \sigma_L(\mathbf{W}_L \sigma_{L-1}(\mathbf{W}_{L-1} \mathbf{a}_{L-2} + \mathbf{b}_{L-1}) + \mathbf{b}_L) \\ &= \ldots \\ &= \sigma_L(\mathbf{W}_L \sigma_{L-1}(\mathbf{W}_{L-1}(\ldots \sigma_2(\mathbf{W}_2 \mathbf{a}_1 + \mathbf{b}_2) \ldots) + \mathbf{b}_{L-1}) + \mathbf{b}_L), \end{aligned} \tag{2.12}$$

en donde partimos de que $\mathbf{a}_1 = \mathbf{X}$. Finalmente, tenemos que

$$MLP(\mathbf{X}; \mathbf{W}, \mathbf{b}) = \hat{\mathbf{y}} = \mathbf{a}_L. \tag{2.13}$$

2.3. Problema inverso

El problema inverso de nuestro modelo se refiere al proceso se estimar los parámetros $\theta := \{\mathbf{W}, \mathbf{b}\}$ de nuestro MLP a partir de un conjunto de datos $\{\mathbf{X}, \mathbf{y}\}$. Dicho problema se puede plantear como el siguiente problema de minimización:

$$\arg\min_{\theta \in \mathbb{R}^{\mathcal{N}}} \; Cost(\theta; \mathbf{X}, \mathbf{y}). \tag{2.14}$$

en donde $Cost : \mathbb{R}^{\mathcal{N}} \to \mathbb{R}$ es nuestra función de error/costo/objetivo. Dicha función mide el error que se comete al aproximar nuestros datos $\mathbf{y}$ con la salida $\hat{\mathbf{y}}$ del modelo $MLP(\mathbf{X}; \theta)$ utilizando los parámetros θ. De esta manera, lo que buscamos es encontrar los parámetros θ (los pesos y sesgos de nuestro MLP) que minimizan dicho error. Formalmente hablando, la función de costo es una métrica que nos permite medir la siguiente distancia

$$Cost(\theta; \hat{\mathbf{X}}, \hat{\mathbf{y}}) = dist(\mathbf{y}, \hat{\mathbf{y}}). \tag{2.15}$$

Dependiendo del tipo de problema que estemos resolviendo algunas funciones de costo pueden ser más convenientes que otras. Lo anterior depende de la naturaleza del número de clases (si es discreto o continuo).

2.3.1. Funciones de costo

Cuando el número de clases es discreto comúnmente se utiliza, como medida de distancia $dist(\mathbf{y}, \hat{\mathbf{y}})$, la función de pérdida de entropía cruzada (CEL), la cual está dada por la expresión:

$$CEL(\mathbf{y}, \hat{\mathbf{y}}) = - < \mathbf{y}, log(\hat{\mathbf{y}}) >= -\sum_{i=1}^{m} y_i log(\hat{y}_i). \tag{2.16}$$

Por otra parte, si consideramos que el número de clases es continuo, entonces usualmente se utiliza la función del error cuadrático medio (MSE):

$$MSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{m} ||\mathbf{y} - \hat{\mathbf{y}}||_2^2 = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2. \tag{2.17}$$

2.3.2. Esquema general de solución del problema inverso

Al resolver el problema (2.14) usualmente se utilizan estrategias de optimización locales, iterativas y que utilizan información total o parcial del gradiente de la función de costo respecto a los parámetros θ. De esta manera, es natural pensar que hay un ciclo iterativo con una serie de pasos establecidos. Dicho algoritmo general se muestra en la figura 2.5.

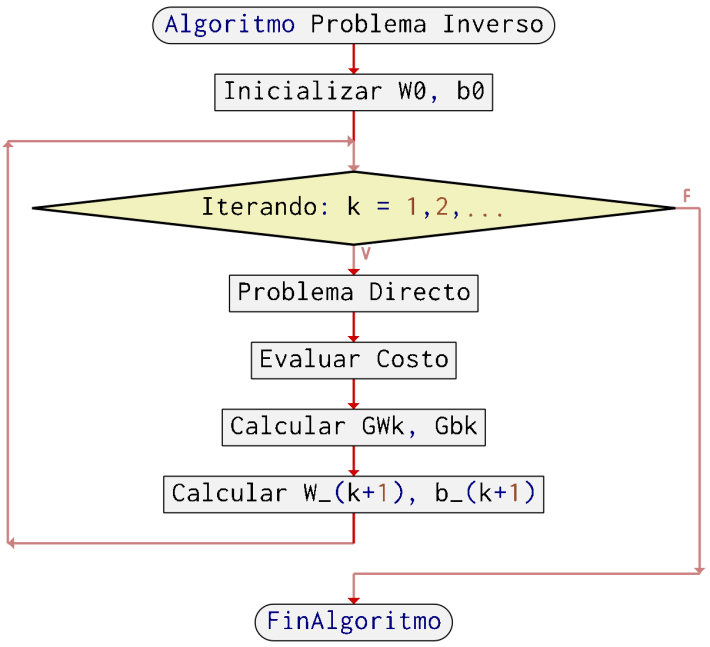

Figura 2.5: Algoritmo general de solución del problema inverso. Las variables $\mathbf{W}_0$ y $\mathbf{b}_0$ denotan respectivamente los pesos y sesgos iniciales de la red. Asímismo, $\mathbf{GW}_k$ y $\mathbf{Gb}_k$ denotan respectivamente los gradientes de $\mathbf{W}_k$ y $\mathbf{b}_k$.

Hasta el momento hemos hablado de como evaluar el problema directo para calcular $\hat{\mathbf{y}}$ y de como obtener el error de nuestro modelo con los parámetros actuales $\theta_k = \{\mathbf{W}_k, \mathbf{b}_k\}$. En las secciones siguientes platicaremos sobre como calcular los gradientes de los pesos y sesgos así como también el proceso para actualizar a los parámetros del modelo $\theta_{k+1} = \{\mathbf{W}_{k+1}, \mathbf{b}_{k+1}\}$.

2.4. Propagación hacia atrás

El algoritmo de propagación hacia atrás ("Back propagation" o *Backprop*) consiste en combinar el uso de la regla de la cadena y de la estructura de las operaciones del problema directo para calcular de manera eficiente las derivadas parciales de la función de costo respecto de sus parámetros. Dicho de otro modo, este algoritmo nos permite calcular los gradientes $\nabla \mathbf{W}$ y $\nabla \mathbf{b}$. El algoritmo de Backprop se puede ver como un caso particular de la derivación automática. De este modo, las derivadas parciales obtenidas tienes una precisión a nivel máquina. El uso de Backprop para estimar los gradientes es una práctica común por ser preciso y computacionalmente económico, en comparación con diferenciación numérica o cálculo simbólico.

2.4.1. Fórmulas de cálculo de derivadas

Dentro del cálculo de las derivadas de la función de costo (2.15) el mayor reto se concentra en calcular las derivadas de la red respecto de los pesos y sesgos. De este modo, se pueden obtener las siguientes expresiones mediante el algoritmo de Backprop:

$$\delta_L = \sigma_L'(\mathbf{z}_L) * (\mathbf{a}_L - \mathbf{y}), \tag{2.18}$$

$$\delta_\ell = \sigma_\ell'(\mathbf{z}_\ell) * \mathbf{W}_{\ell+1}^T \delta_{\ell+1}, \qquad \ell = 2, 3 \ldots, L-1, \tag{2.19}$$

$$\nabla_{\mathbf{b}_\ell} Cost = \delta_\ell, \qquad \ell = 2, 3 \ldots, L, \tag{2.20}$$

$$\nabla_{\mathbf{W}_\ell} Cost = \delta_\ell \mathbf{a}_\ell^T, \qquad \ell = 2, 3 \ldots, L. \tag{2.21}$$

Hay varias aclaraciones que hay que hacer respecto a las ecuaciones (2.18)-(2.21):

I) La operacion $*$ denota el producto de Hadamard, el cual es solo el producto entrada por entrada preservando la misma dimensión.

II) Denotamos por σ_ℓ' a la derivada de la funciones de activación σ_ℓ.

III) Recordemos que $\mathbf{y} \in \mathbb{R}^{n_L \times m}$. En el caso de regresión tenemos que $n_L = 1$. Por otra parte, en el caso de clásificación tenemos que $\mathbf{y}$ es un vector de tamaño m con los índices de la clase a la que pertenece cada uno de los datos $\mathbf{X}_i$. Luego entonces, podemos decir que hay n_L clases y por tanto $\mathbf{y}$ se puede pensar como una matriz de tamaño $n_L \times m$. Cada columna de $\mathbf{y}$ sería un vector de tamaño n_L donde todas las entradas son cero excepto la correspondiente al índice que indica a cuál clase pertenece. Dicha representación de $\mathbf{y}$ como matriz se conoce popularmente como "one hot".

IV) Puesto que $\mathbf{a}_L, \mathbf{y} \in \mathbb{R}^{n_L \times m}$, tenemos entonces que $\mathbf{a}_L - \mathbf{y} \in \mathbb{R}^{n_L \times m}$. Sabemos además, que $\mathbf{z}_L \in \mathbb{R}^{n_L \times m}$ y que la evaluación de las funciones σ'_L se realizan entrada por entrada. De este modo $\sigma'_L(\mathbf{z}_L) \in \mathbb{R}^{n_L \times m}$. De esta manera, tenemos que $\delta_L = \sigma'_L(\mathbf{z}_L) * (\mathbf{a}_L - \mathbf{y}) \in \mathbb{R}^{n_L \times m}$.

V) Extendiendo lo anterior para las capas internas, tenemos que $\sigma'_\ell(\mathbf{z}_\ell) \in \mathbb{R}^{n_\ell \times m}$, $\delta_{\ell+1} \in \mathbb{R}^{n_{\ell+1} \times m}$ y como además sabemos que $\mathbf{W}_{\ell+1}^T \in \mathbb{R}^{n_\ell \times n_{\ell+1}}$, entonces podemos concluir que $\delta_\ell = \sigma'_\ell(\mathbf{z}_\ell) * \mathbf{W}_{\ell+1}^T \delta_{\ell+1} \in \mathbb{R}^{n_\ell \times m}$.

VI) El gradiente $\nabla_{\mathbf{b}_\ell} Cost$ denota el vector de derivadas parciales de la función de costo respecto a los sesgos de la capa ℓ-ésima. Dicho esto, remarcamos que $\nabla_{\mathbf{b}_\ell} Cost \in \mathbb{R}^{n_\ell \times m}$.

VII) El gradiente $\nabla_{\mathbf{W}_\ell} Cost$ denota la matriz de derivadas parciales de la función de costo respecto a los pesos de la capa ℓ-ésima. Recordemos que $\mathbf{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ por lo que tiene sentido esperar que $\nabla_{\mathbf{W}_\ell} Cost$ tenga las mismas dimensiones. En efecto podemos ver fácilmente que $\delta_\ell \mathbf{a}_\ell^T$ tiene las dimensiones esperadas.

VIII) Los gradientes de las ecuaciones (2.20) y (2.21) están evaluados en el punto $\mathbf{X}$ aunque no se indíque de manera explícita. Esto es fácil de ver si recordamos que el algoritmo de backprop lo realizamos después de haber evaluado el problema directo, es decir, cuando hemos propagado hacia adelante el punto $\mathbf{X}$ hasta calcular $\mathbf{a}_L$, quien aparece en el lado derecho de la ecuación (2.18).

Las demostraciones de las fórmulas (2.18)-(2.21) no son difíciles de obtener pero son un poco largas. Para evitar perder claridad en la lectura, el lector interesado puede consultar las demostraciones a dichas fórmulas en [Higham C. F., 2019].

Una estrategia útil al momento de realizar la implementación de las fórmulas (2.18)-(2.21) es comparar los gradientes obtenidos con los resultantes de aproximar dichos gradientes mediante diferencias finitas. Lo anterior se puede realizar para algunas iteraciones solo para verificar que el algoritmo de Backprop está correctamente implementado. Los gradientes se pueden comparar utilizando la norma relativa de su diferencia por ejemplo.

Capítulo 3

Optimización Numérica

3.1. Qué es minimizar

Constantemente buscamos las condiciones que permitan obtener el máximo beneficio posible. Por ejemplo, cuando maximizamos la cantidad de gastos que podemos cubrir con el presupuesto semanal, cuando encontramos la mejor forma de reducir el tiempo de traslado de la casa a la escuela, al elegir la fila del supermercado con el menor tiempo de espera, o bien, al encontrar el flujo de procesamiento que minimiza el costo de ensamblado en una fábrica de autos.

Encontrar las mejores condiciones posibles, para realizar una tarea, o bien, en las que un fenómeno particular se presenta, es lo que entendemos por *Optimizar*.

Inclusive, la naturaleza optimiza, pues los fenómenos físicos tienden a alcanzar estados de mínima energía. De esta forma, la optimización es una herramienta indispensable en la toma de decisiones y es de vital importancia en el análisis de nuestro entorno.

En la práctica, el punto de partida es identificar las cantidades que pueden ser cuantificables en el problema de estudio. En los ejemplos anteriores, estas cantidades pueden ser dinero, distancia, tiempo, ganancia, energía potencial, o alguna combinación de estas que puedan ser expresadas por un solo valor numérico al que llamaremos *objetivo*. Este objetivo depende de diferentes formas de las características del problema de estudio, a las que llamamos *variables* o *incógnitas*. En el contexto los ejemplos anteriores, las variables pueden ser: ingresos, medio de transporte, tráfico, cantidad de personas en el supermercado, afluencia de gente, cantidad de personal, costo de proveedores, etc.

En otras palabras, podemos observar que, al asociar las condiciones de un

fenómeno con incógnitas, y a su vez con cantidades, encontrar condiciones óptimas se traduce en buscar valores *mínimos* (o máximos según el contexto) de tales incógnitas. En este sentido, es que podemos hablar de optimizar o minimizar un problema.

3.2. Planteamiento del problema general de minimización

Supongamos que un fenómeno a estudiar, se puede describir por una función, digamos

$$\begin{aligned} f : \Omega &\rightarrow \Psi \\ x &\mapsto y, \end{aligned} \tag{3.1}$$

donde las incógnitas x estén definidas en un conjunto $x \in \Omega$, y se les asocia un valor en un conjunto de $y \in \Psi$, al que nos hemos referido como cantidad objetivo. Análogamente, nos podemos referir a f como función objetivo. Cabe señalar que, además de la definición de la relación entre x y y, i.e. $f(x) = y$, determinar los conjuntos Ω y Ψ también son parte del proceso de modelado. Una consideración fundamental en este punto, es el hecho de que la forma en la que se cuantifican la cantidad objetivo y las incógnitas, y por consiguiente, la forma en la que se aborda el problema de Optimización, está íntimamente relacionada con la definición de los conjuntos Ω y Ψ.

Empezaremos considerando problemas de optimización definidos en los números reales. Es decir, $f : \mathbb{R} \rightarrow \mathbb{R}$. Entonces, en el sentido más general, el problema de Optimización es:

$$\min_{x \in \mathbb{R}} f(x). \tag{3.2}$$

Notemos que la cantidad de objetivo y concentra en un sólo valor la relación entre las diferentes incógnitas x del problema definido por f. Por tanto, tiene sentido hablar de un orden para $f(x)$ y más aún, de valores mínimos y máximos. El valor x^* que resuelve el problema (3.2) lo llamaremos el *óptimo* de la función f. Nos referiremos coloquialmente como minimizar la función objetivo f, al procedimiento de encontrar el valor x^*. Veremos a continuación características y condiciones sobre x^*.

Dependiendo del problema de estudio, y por consiguiente de la función f, es posible que exista más de una solución al problema (3.2). En los casos donde exista una sola solución, se dice que ésta es el *mínimo global* u *óptimo global*. Cuando se tienen varias soluciones, se dice que cada una de duchas

soluciones es un *mínimo local*, o *minimizador local*. Las definiciones formales vienen a continuación.

Definición 3.2.1. Un punto $x^* \in \mathbb{R}$ es el *mínimo global* de $f : \mathbb{R} \to \mathbb{R}$ si y sólo si

$$f(x^*) \leq f(x), \quad \forall x \in \mathbb{R}.$$

Definición 3.2.2. Un punto $x^* \in \mathbb{R}$ es un *mínimo local* de $f : \mathbb{R} \to \mathbb{R}$ si existe una vecindad $\mathcal{N}$ alrededor de x^* tal que

$$f(x^*) \leq f(x), \quad \forall x \in \mathcal{N}.$$

A partir de las definiciones 3.2.1 y 3.2.2, resulta natural ver que, para identificar un mínimo global, en particular debe ser mínimo local. Consecuentemente, la forma para identificar mínimos locales y globales x^*, es examinando los puntos en la vecindad de x^* y comparar su valor bajo f para asegurarse que no se tiene otro punto con valor menor. Sin embargo, cuando la función f es *suave* (i.e. continua y con todas sus derivadas continuas) hay formas mas eficientes, y con sustento teórico, para determinar mínimos locales. En particular, si f es dos veces continuamente diferenciable, la herramienta que nos proporcionará las bases para identificar bajo que condiciones un punto es mínimo local, es el Teorema de Taylor. Enunciaremos a continuación el Teorema de Taylor en una variable.

Teorema 3.2.1 (Teorema de Taylor)**.** Supongamos que $f : \mathbb{R} \to \mathbb{R}$ es continuamente diferenciable sobre un intervalo abierto que contiene a los puntos a y x. Entonces

$$f(x) = f(a) + f'(a)(x-a) + \frac{f'(a)}{2}(x-a)^2 + \ldots + \frac{f^n(a)}{n}(x-a)^n + R^n(x),$$

donde $R^n(x) = \frac{f^{n+1}(c)}{(n+1)}(x-a)^{n+1}$, para algún número c entre a y x.

La demostración del teorema de Taylor puede ser consultada en [Spivak, 2019].

3.3. Condiciones de Optimalidad

El teorema de Taylor proporciona una aproximación local de la función f en un punto en términos de las derivadas de la misma. Es a partir de éstas, las derivadas de la función que se pueden establecer las condiciones para identificar cuándo un punto es mínimo local. Estas condiciones se conocen como *condiciones de optimalidad* y las enunciaremos a continuación

Teorema 3.3.1 (Condición de optimalidad (necesaria) de primer orden)**.** Si x^* es un mínimo local de f y f es continuamente diferenciable en una vecindad abierta de x^*, entonces $f'(x^*) = 0$.

Demostración. Haremos la demostración por contradicción. Supongamos que x^* es un mínimo local y que $f(x^*) \neq 0$. Como x^* es un mínimo local existe un abierto que contiene a x^*, tal que

$$f(x^*) < f(x), \quad \forall x \in (a, b).$$

Ahora supongamos que $f(x^*) < 0$ y tomamos $x \in (a, b)$ tal que $h = x - x^* > 0$. Luego por el teorema de Taylor a primer orden tenemos

$$f(x) = f(x^*) + f'(x^*)h.$$

Reescribiendo la última ecuación,

$$f(x) - f(x^*) = f'(x^*)h < 0,$$

obtenemos que $f(x) < f(x^*)$. Lo cual contradice que x^* sea un mínimo local. Para el caso que $f(x^*) > 0$, bastaría elegir $h < 0$. Por lo tanto, si x^* es mínimo local entonces $f'(x^*) = 0$. □

En el caso que la función f es dos veces continuamente diferenciable, el teorema de Taylor proporciona una aproximación local de orden dos, misma que da lugar a la *condición de optimalidad de segundo orden.*

Teorema 3.3.2 (Condición de optimalidad (necesaria) de segundo orden)**.** Sea x^* un mínimo local de f, f dos veces continuamente diferenciable en una vecindad abierta de x^*, entonces $f'(x^*) = 0$ y $f''(x^*) > 0$.

Demostración. Supongamos que x^* es un mínimo local, entonces existe un abierto que contiene a x^*, tal que

$$f(x^*) < f(x), \quad \forall x \in (a, b).$$

Ahora, por el teorema de Taylor se tiene que,

$$f(x) = f(x^*) + f'(x^*)(x - x^*) + \frac{f''(x)}{2}(x - x^*)^2 \quad \text{para algún } x \in (a, b).$$

Como consecuencia del teorema 3.3.1, tenemos que $f'(x^*) = 0$. Entonces

$$f(x) - f(x^*) = f''(x^*)(x - x^*)^2,$$

como $f(x) - f(x^*) > 0$ nos queda que $f''(x^*) > 0$. □

Los teoremas 3.3.1 y 3.3.2 proporcionan información sobre una función una vez que tenemos localizado un mínimo local. Sin embargo, aun necesitamos algún criterio para, a partir de la información local, asegurar que un punto es un mínimo. Para esto, enunciaremos las *condiciones de optimalidad suficientes*.

Teorema 3.3.3 (Condiciones de optimalidad (suficientes) de segundo orden)**.** Sea f continua en una vecindad abierta de x^* tal que $f'(x^*) = 0$ y $f''(x^*) > 0$. Entonces x^* es un mínimo de f.

Demostración. Como $f''(x^*) > 0$ y continua en x^*, podemos elegir un radio r tal que $f''(x)$ permanezca positiva para todo $x \in \mathcal{D}_r$, con $D_r = \{x||x - x^*| < r\}$. Sea cualquier h distinto de cero tal que $h < r$. Notemos que $x^* + p \in D_r$ pues

$$|(x^* + h) - x^*| = h < r.$$

Luego, por el teorema de Taylor se tiene que

$$f(x^* + h) = f(x^*) + hf'(x^*) + \frac{1}{2}h^2 f''(x) \quad \text{para algún } x \in D_r,$$

y como $f'(x^*) = 0$

$$f(x^* + h) = f(x^*) + \frac{1}{2}h^2 f''(x) \quad \text{para algún } x \in D_r,$$

además, como $h^2 f''(x) > 0$, esto implica que

$$f(x^* + h) > f(x^*).$$

Por tanto x^* es un mínimo local de f. □

Las condiciones de optimalidad suficientes incorporan de forma natural la información de las primera y segunda derivadas para identificar el mínimo de la función en una región. Es a partir de éstas condiciones que los métodos de optimización son diseñados, es decir, típicamente los métodos de optimización identifican puntos tales que la derivada de la función es cero (o aproximadamente cero) y la segunda derivada (cuando está disponible) es positiva. En la práctica, el mínimo de una función no siempre es alcanzado, esto es debido a principalmente a limitantes numéricas. Existen puntos que no pueden ser representados de manera exacta en precisión finita, no siempre se tiene la primera o segunda derivada analítica de la función, el costo computacional de evaluar la función en cuestión es muy elevado, etc. Este tipo de contrariedades deben ser así mismo consideradas como parte de la implementación de los métodos de optimización en cada aplicación. Se recomienda la lectura de [Quarteroni et al., 2010].

3.4. Métodos de búsqueda por líneas

La relevancia de las condiciones de optimalidad, es que éstas nos permiten garantizar el haber encontrado o no, solución al problema de optimización. En este sentido, los métodos numéricos para resolver problemas de optimización, o comúnmente llamados *Métodos de Optimización*, tienen como objetivo utilizar procedimientos computacionales o algoritmos, para encontrar puntos x^* que satisfagan las condiciones de optimalidad.

Típicamente, los métodos de optimización son iterativos, requieren de un punto inicial x_0, a partir del cual construyen una sucesión finita $\{x_k\}_{k=1}^{N}$ que termina cuando el algoritmo no puede progresar más o cuando se cumplen las condiciones de optimalidad. Para decidir como pasar de una iteración x_k a x_{k+1} los métodos de optimización utilizan un esquema básico definido como:

$$x_{k+1} = x_k + \alpha_k p_k. \tag{3.3}$$

Donde p_k es un vector dirección, conocido como *dirección de descenso*, de la misma dimensión que x_k y α_k es un escalar.

Existen dos estrategias principales para construir la iteración (3.3), que dan origen a dos familias de métodos de optimización: *Métodos de Búsqueda por Lineas* y *Métodos de Región de Confianza*.

La característica principal de los métodos de Región de Confianza, consiste en que éstos, primero construyen el escalar α_k, que corresponde al radio de una vecindad llamada región de confianza, y posteriormente, definen una dirección de descenso p_k dentro de esa región. Mas detalles pueden consultarse en [Wright et al., 1999].

Por el contrario, los métodos de optimización de Búsqueda por Líneas, en primer instancia definen o construyen una dirección de descenso p_k, y posteriormente, definen el tamaño de paso α_k. El algoritmo general de éste tipo de métodos se describe en el Algoritmo 1. En este grupo de estrategias se encuentra el método que describiremos a continuación.

3.4.1. Método de Descenso más pronunciado

Existe una amplia variedad de métodos de optimización de Búsqueda por Líneas. La diferencia principal está en la forma como se construyen tanto la dirección de descenso, como el tamaño de paso (pasos 3 y 4 del Algoritmo 1 manteniendo siempre el orden de ejecución de estos pasos). En este documento nos restringiremos al método de *Descenso más pronunciado*, también conocido como *Descenso de paso fijo*, *Descenso de gradiente* o bien *Steepest Descend*.

Definiremos primero qué es una dirección de descenso.

Entrada: f, f', x_0
Salida: x^*
1 $k = 0$;
2 **mientras** $f'(x_k) \neq 0$ {*Condición de optimalidad de primer orden*} **hacer**
3 Construir p_k a partir de x_k, $f(x_k)$ y $f'(x_k)$;
4 Construir $\alpha_k > 0$ a partir de x_k, $f(x_k)$, $f'(x_k)$ y p_k ;
5 $x_{k+1} = x_k + \alpha_k p_k$;
6 $k = k + 1$;
7 **fin**
8 **devolver** x_k

Algoritmo 1: Algoritmo general para un método de optimización de búsqueda por líneas.

Definición 3.4.1. Se dice que la dirección p de $\mathbb{R}$ es una dirección de descenso en x si

$$f(x + \alpha p) < f(x),$$

para α positivo suficientemente pequeño.

Lema 3.4.1. Considere un punto $x \in \mathbb{R}^n$. Cualquier dirección que satisface

$$f'(x)p < 0,$$

es una dirección de descenso.

Demostración. Por el Teorema de Taylor a primer orden, tenemos para α pequeño,

$$f(x + \alpha p) = f(x) + \alpha f'(x)p.$$

Como $f'(x)p < 0$, implica que;

$$\begin{aligned} f(x + \alpha p) - f(x) &= \alpha f'(x)p, \\ f(x + \alpha p) - f(x) &< 0, \\ f(x + \alpha p) &< f(x). \end{aligned}$$

Por tanto, p es una dirección de descenso. □

3.4.2. Dirección de descenso

Ahora, una pregunta razonable es, ¿cuál es la mejor dirección de descenso? Esta debería ser aquella que nos proporciona la mayor diferencia entre $f(x + \alpha p)$ y $f(x)$ para α suficientemente pequeño. Pero como esta diferencia

depende de $\alpha f'(x)p$. Como vimos en el lema 3.4.1, como $f'(x)p < 0$ necesitamos encontrar el valor más negativo (o el mínimo) que puede alcanzar $f'(x)p$. Notemos que, como estamos buscando una dirección, en particular buscaremos que p sea dirección unitaria, es decir $|p| = 1$. Para encontrarla, podemos plantear el problema de optimización

$$\min_p f'(x)p, \text{ con } |p| = 1.$$

Intuitivamente, podemos pensar la derivada de una función como un indicador del crecimiento o decrecimiento de una función. Es decir, como:

$$f'(x) = \lim_{h \to 0} \frac{f(x-h) - f(x)}{h},$$

tenemos que, si $f'(x) > 0$, los valores de $f(x-h)$ son mayores a $f(x)$ y por tanto la función es localmente creciente, y viceversa. En ese sentido, podemos considerar elegir una dirección de descenso en términos de la derivada de la función.

Observemos que, si $p = cf'(x)$, con $c > 0$, entonces $f'(x)p > 0$, por lo cual, de acuerdo al lema 3.4.1, p no puede ser dirección de descenso. Por otro lado, si $c < 0$:

$$\begin{aligned} f'(x)p &= f'(x)(cf'(x)), \\ &= c(f'(x))^2, \end{aligned}$$

tenemos dirección de descenso. Luego:

$$\begin{aligned} |p| &= 1, \\ |cf'(x)| &= 1, \\ |c||f'(x)| &= 1, \\ |c| &= \frac{1}{|f'(x)|}. \end{aligned}$$

Entonces, debe ser que $c = -\frac{1}{|f'(x)|}$. De esta forma, llegamos a que:

$$p = -\frac{f'(x)}{|f'(x)|}, \tag{3.4}$$

es la mejor dirección de descenso.

En la práctica, el escalar que corresponde a $1/f'(x)$ se traslada al valor del tamaño de paso. De esta forma, la dirección de descenso que se construye en el paso 3 del Algoritmo 1 se toma como:

$$p_k = -f'(x_k). \tag{3.5}$$

Cabe señalar que, si bien la dirección de descenso tomada como (3.5) es la mejor, no es la única que se puede tomar. Es decir, basta con que se parezca a la dirección de menos la derivada de la función. Por supuesto, entre menos se parezca la dirección de descenso a menos la derivada, el desempeño del método numérico será menos eficiente. En ocasiones, no es posible calcular la derivada de una función en todos los puntos x_k, en esos casos se puede recurrir a una aproximación numérica de la derivada de la función, como se describirá más adelante.

3.4.3. Tamaño de paso y condiciones de Wolfe

De forma más o menos general, los métodos de Búsqueda por Lineas, utilizan la dirección de descenso de la ecuación (3.5). Por lo que la principal diferencia entre los distintos métodos está en la forma que se elige el tamaño de paso (paso 4 en el Algoritmo 1).

La forma más simple es elegir

$$\alpha_k = C, \tag{3.6}$$

con $C > 0$ una constante que se mantiene fija, para todas las iteraciones del algoritmo. Es precisamente esta elección la que da el nombre al método de "Descenso de paso fijo".

El método de descenso de paso fijo es ampliamente usado, por simplicidad, aunque es necesario en ocasiones realizar experimentos para ajustar un valor de la constante C adecuado.

Recordemos que la mejor dirección de descenso está dada por p_k como en la ecuación (3.4). Entones, si tomamos $C = 1$, en la actualización del paso 5 del Algoritmo 1 tenemos:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k, && (3.7)\\ &= x_k + 1 p_k, && (3.8)\\ &= x_k + \frac{|p_k|}{|p_k|} p_k, && (3.9)\\ &= x_k + |p_k| \frac{p_k}{|p_k|}. && (3.10) \end{aligned}$$

Lo cual indica que en realidad nos "estamos moviendo" en las iteraciones tomando la mejor dirección de descenso, con tamaño de paso igual a la magnitud de la derivada de la función. Por esta razón, en general, no es recomendable tomar tamaño de paso con $C > 1$, pues puede generar divergencia cuando se trabaja con funciones con derivadas muy grandes. Podemos pensar en $C = 1$ como una cota superior para el tamaño de paso.

Consideremos ahora la función $f(x) = x^2$. Tomamos un x_0 arbitrario y el tamaño de paso $\alpha_k = 1$. Entonces, siguiendo el Algoritmo 1 tenemos:

$$\begin{aligned} p_0 &= -f'(x_0) = -2x_0, \\ x_1 &= x_0 + 1(-2x_0) = -x_0, \\ p_1 &= -f'(x_1) = -2(-x_0) = 2x_0, \\ x_2 &= x_1 + 1(2x_0) = -x_0 + 2x_0 = x_0, \\ \dots x_{2k-1} &= -x_0, \\ x_{2k} &= x_0. \end{aligned}$$

Es decir, el método se queda oscilando entre los valores x_0 y $-x_0$, independientemente del valor del punto inicial. Este es una limitante del método de descenso de paso fijo, que en funciones localmente simétricas y convexas, puede llegar a tener problemas de convergencia. De esta forma, valores de $\alpha_k = 1$ podrían no ser recomendables.

Sin embargo, mencionando nuevamente el hecho de que la mejor dirección de descenso se obtiene con (3.4), tomar valores de α_k muy pequeños resulta en limitar el avance que puede tener el método en cada iteración y por consecuencia, en la necesidad de un número grande de iteraciones para alcanzar convergencia del método. En otras palabras, requerimos elegir el tamaño de paso lo más grande posible, garantizando que el método desciende.

Ante esta problemática, la elección del tamaño de paso se vuelve determinante en el desempeño del método. Por lo cual, se ha tratado de formular criterios que contribuyan en mejorar la elección del tamaño de paso, de forma que pueda ser variable a lo largo de las iteraciones del método y de esta forma se sobreponga a las limitaciones mencionadas en el caso del tamaño de paso constante.

El primer criterio que mencionaremos, aparece en la literatura como *Condición de Armijo*. La idea de la Condición de Armijo, es identificar si el paso siguiente en las iteraciones del método de descenso es efectivamente un descenso en la función. Para lo cual, se traza una linea recta, que pasa por $f(x_k)$ y con pendiente similar a f en ese punto. Si el tamaño de paso elegido queda por debajo de esa recta, entonces tendremos suficiente descenso, en caso contrario, será necesario tomar un tamaño de paso distinto. Esto se ilustra en la Figura 3.1.

La Condición de Armijo consiste entonces como elegir α tal que satisfaga:

$$f(x_k + \alpha p_k) < f(x_k) + c_1 \alpha f'(x_k) p_k, \tag{3.11}$$

para alguna constante $c_1 \in (0, 1)$. Observemos que el lado derecho de la desigualdad corresponde a una recta para α, que podemos denotar $l(\alpha)$, tomando x_k y p_k fijos. El lado izquierdo de la desigualdad define una función

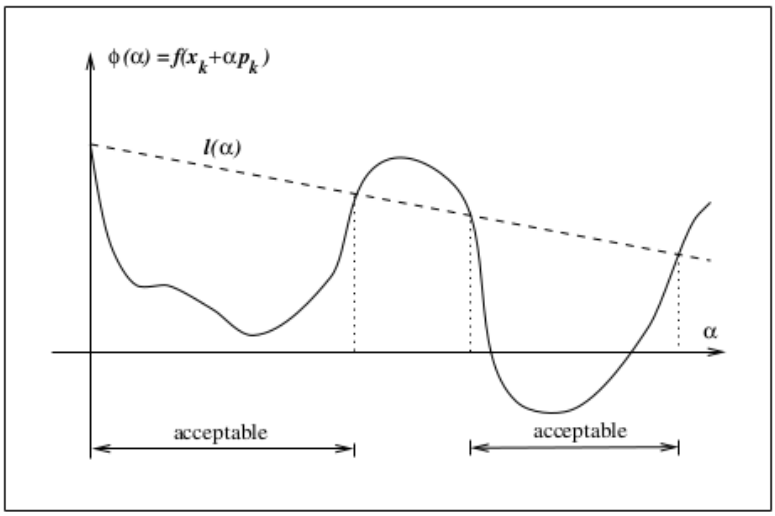

Figura 3.1: Condición de suficiente descenso. Si el tamaño de paso elegido no proporciona un punto que quede por debajo de la recta correspondiente un modelo lineal aproximante a la función f, no se puede garantizar suficiente descenso. Fuente: [Wright et al., 1999].

$\phi(\alpha)$. Observar nuevamente la Figura 3.1, las regiones donde se cumple la condición de Armijo corresponden a los valores aceptables para α. La constante c_1 nos permite controlar qué tan estricto estableceremos la condición de suficiente descenso y dependerá de la aplicación. Por ejemplo, valores de c_1 cercanos a cero indican que la condición es poco rigurosa y se utilizaría en funciones planas con gradientes muy cercanos a cero. En la práctica c_1 se elige con valores pequeños, digamos $c_1 = 1 \times 10^{-4}$.

La condición de Armijo (3.11) puede ser utilizada entonces como un criterio para ajustar el tamaño de paso de forma variable a lo largo de las iteraciones. Sin embargo, por sí sola, no garantiza que el método realiza un progreso razonable, pues puede suceder que se elijan tamaños de paso muy pequeños. En el ejemplo de la Figura 3.1, se observan dos regiones aceptables, una de ellas tiene valores más pequeños (se traduce en menor progreso del método) que la otra. Para delimitar los valores pequeños del tamaño de paso, se incorpora una condición adicional, relacionada con la curvatura de la función:

$$f'(x_k + \alpha_k p_k)p_k \geq c_2 f'(x_k)p_k, \tag{3.12}$$

para alguna constante $c_2 \in (c_1, 1)$. Observemos que el lado izquierdo de esta desigualdad es la derivada de $\phi(\alpha)$, entonces, esta condición de curvatura asegura que la pendiente de ϕ en α_k es c_2 veces mayor que la pendiente inicial $\phi(0)$.

Intuitivamente, lo que nos dice la condición (3.12) es que si tenemos una función cuya derivada es muy grande, entonces podemos incrementar significativamente el valor del tamaño de paso y tomar ventaja de esto para lograr un mayor descenso. Por el contrario, si la derivada de la función es pequeña, o incluso que cambie de signo, corresponderá a funciones muy planas donde no se espera poder incrementar en gran medida el tamaño de paso. Los valores recomendados de c_2 son cercanos a 1. Por ejemplo $c_2 = 0{,}9$.

Las condiciones (3.11) y (3.12) se conocen como *Condiciones de Wolfe*. Estas condiciones son criterios que se incorporan para asegurar suficiente descenso pero no son en sí mismas un procedimiento para encontrar el valor de α_k. Buscar el valor óptimo para α_k que cumpla las Condiciones de Wolfe, puede conducir a un problema igual o más complejo que resolver el problema de optimización original. Por lo tanto, a fin de mantener la simplicidad del método de optimización y no incrementar el costo computacional, recordemos que se debe encontrar un tamaño de paso en cada iteración, se recurre a estrategias de búsqueda basadas en heurísticas, que permitan encontrar un tamaño de paso adecuado.

El algoritmo *Backtracking* es un procedimiento simple, que se basa en tomar un tamaño de paso $\bar{\alpha}$ suficientemente grande, y después probar si éste satisface la condición de suficiente descenso (puede ser sólo Armijo o las condiciones de Wolfe). En caso de que no se satisfagan, se reduce el valor de $\bar{\alpha}$ y se vuelve a probar. Los pasos se listan en el Algoritmo 2.

Entrada: $\bar{\alpha}$, ρ, $c_1 \in (0,1)$, $c_2 \in (c_1,1)$, x_k, p_k
Salida: α_k
1 **mientras** *no se cumple suficiente descenso* **hacer**
2 $\quad \bar{\alpha} = \rho\bar{\alpha}$;
3 **fin**
4 **devolver** $\alpha_k = \bar{\alpha}$

Algoritmo 2: Algoritmo Backtracking para optimización.

3.5. Extensión a $\mathbb{R}^n$

En este capítulo se han revisado aspectos teóricos relacionados con Optimización Numérica en una dimensión. Sin embargo, los problemas de optimización con mucha frecuencia son problemas que se plantean en espacios de dimensiones mayores. Los conceptos mencionados hasta ahora se extienden de manera natural a el caso n-dimensional. Esta sección está enfocada en presentar entonces tales versiones n-dimensionales.

Consideramos entonces una función $f : \mathbb{R}^n \leftarrow \mathbb{R}$, y el vector $\mathbf{x} \in \mathbb{R}$. Un problema de optimización se define entonces como:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}).$$

Las definiciones para mínimo/máximo local/global son inmediatas.

El teorema de Taylor queda como:

Teorema 3.5.1 (Teorema de Taylor ($\mathbb{R}^n$))**.** Supongamos que $f : \mathbb{R}^n \to \mathbb{R}$ es continuamente diferenciable sobre un intervalo abierto que contiene a los puntos $\mathbf{a}$ y
$mathbfx$. Entonces

$$f(\mathbf{x}) = f(\mathbf{a}) + \nabla f(\mathbf{a})(\mathbf{x}-\mathbf{a}) + \frac{\nabla^2 f(\mathbf{a})}{2}(\mathbf{x}-\mathbf{a})^2 + \ldots + \frac{f^n(\mathbf{a})}{n}(\mathbf{x}-\mathbf{a})^n + R^n(\mathbf{x}),$$

donde $R^n(\mathbf{x}) = \frac{f^{n+1}(\mathbf{c})}{(n+1)}(\mathbf{x}-\mathbf{a})^{n+1}$, para algún número $\mathbf{c}$ entre $\mathbf{a}$ y $\mathbf{x}$.

Observamos que las primera y segunda derivadas de f cambiaron al gradiente ∇f y la matriz Hessiana $\nabla^2 f$.

A partir de esto, entonces, las condiciones de optimalidad se definen como:

Teorema 3.5.2 (Condición de optimalidad (necesaria) de primer orden ($\mathbb{R}^n$))**.** Si $\mathbf{x}^*$ es un mínimo local de f y f es continuamente diferenciable en una vecindad abierta de $\mathbf{x}^*$, entonces $\nabla f(\mathbf{x}^*) = 0$.

En el caso que la función f es dos veces continuamente diferenciable, tenemos:

Teorema 3.5.3 (Condición de optimalidad (necesaria) de segundo orden ($\mathbb{R}^n$))**.** Sea $\mathbf{x}^*$ un mínimo local de f, f dos veces continuamente diferenciable en una vecindad abierta de $\mathbf{x}^*$, entonces $\nabla f(\mathbf{x}^*) = 0$ y $\nabla^2 f(\mathbf{x}^*)$ es positiva definida.

En este caso notamos que la condición de convexidad local, cambia de ser un escalar ($f''(x) > 0$) a que la matriz Hessiana sea positiva definida, aunque conceptualmente son equivalentes.

Teorema 3.5.4 (Condiciones de optimalidad (suficientes) de segundo orden ($\mathbb{R}^n$))**.** Sea f continua en una vecindad abierta de $\mathbf{x}^*$ tal que $\nabla f(\mathbf{x}^*) = 0$ y $\nabla^2 f(\mathbf{x}^*)$ positiva semidefinida. Entonces $\mathbf{x}^*$ es un mínimo local de f.

El esquema general tanto de los métodos de optimización de Búsqueda por Líneas, como los de Región de Confianza se mantiene sin cambios al pasar al caso n-dimensional. Es decir, el Algoritmo 1 se mantiene igual, y por tanto, se mantiene el esquema iterativo definido por:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k.$$

La dirección de máximo descenso en el caso n-dimensional es:

$$\mathbf{p} = -\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|},$$

con la consideración de que el criterio para suficiente descenso del lema 3.4.1 correspondiente es:

$$\nabla f(\mathbf{x})^T \mathbf{p} < 0.$$

Entonces, de forma análoga, se pretende tomar la dirección de descenso como:

$$\mathbf{p}_k = -\nabla f(\mathbf{x}).$$

El efecto del tamaño de paso $\alpha_k \in \mathbb{R}$ tiene la misma interpretación y limitantes que en el caso unidimensional. Es decir, los valores grandes de α_k generan inestabilidad y divergencia de los métodos de descenso de paso fijo, y los valores pequeños no aseguran suficiente descenso. Por lo cual, se incorporan las condiciones de Wolfe, mismas que quedan de la siguiente forma.

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k, \tag{3.13}$$

$$\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k \geq c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k. \tag{3.14}$$

El algoritmo 2 también se utiliza exactamente en la misma forma.

3.6. Convergencia de los métodos de Búsqueda por Líneas

Una característica propia de los métodos numéricos deterministas, es que tienen convergencia monótona. Es decir, el valor que se obtiene a lo largo de las iteraciones, se aproxima paulatinamente al punto de convergencia del método. En nuestro caso, al valor mínimo de la función. La velocidad, o factor con que esta aproximación se da, a menudo se puede estudiar analíticamente para caracterizar el desempeño del método.

Teorema 3.6.1. Considere la iteración $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, con $\mathbf{p}_k$ dirección de descenso y tamaño de paso α_k que satisface (3.13)-(3.14). Suponiendo que f está acotada por abajo en $\mathbf{R}^n$ y que es continuamente diferenciable en un conjunto abierto $\mathcal{N}$ conteniendo la curva de nivel $\mathcal{L} = \{\mathbf{x} : f(\mathbf{x}) \leq f(\mathbf{x}_0)\}$, donde $\mathbf{x}_0$ es el punto inicial de la iteración. Supongamos también que el gradiente ∇f es Lipschitz continuo en $\mathcal{N}$, esto es, que existe una constante $L > 0$ tal que:

$$\|\nabla f(\mathbf{x}) - \nabla f(\tilde{\mathbf{x}})\| \leq L\|\mathbf{x} - \tilde{\mathbf{x}}\|, \text{ for all } \mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{N}.$$

Entonces:

$$\sum_{k \geq 0} \cos^2 \theta_k \|\nabla f(\mathbf{x}_k)\|^2 < \infty.$$

Demostración. A partir de (3.14) y $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, con $\mathbf{p}_k$ tenemos

$$\begin{aligned}
\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k &\geq c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k, \\
\nabla f(\mathbf{x}_{k+1})^T \mathbf{p}_k &\geq c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k, \\
\nabla f(\mathbf{x}_{k+1})^T \mathbf{p}_k - \nabla f(\mathbf{x}_k)^T \mathbf{p}_k &\geq c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k - \nabla f(\mathbf{x}_k)^T \mathbf{p}_k, \\
\left(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\right)^T \mathbf{p}_k &\geq (c_2 - 1) \nabla f(\mathbf{x}_k)^T \mathbf{p}_k.
\end{aligned}$$

Por otro lado, la condición de Lipschitz implica que:

$$\begin{aligned}
\left(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\right)^T \mathbf{p}_k &\leq L\|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2, \\
\left(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\right)^T \mathbf{p}_k &\leq L\|\mathbf{x}_k + \alpha_k \mathbf{p}_k - \mathbf{x}_k\|^2, \\
\left(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\right)^T \mathbf{p}_k &\leq \alpha_k L\|\mathbf{p}_k\|^2.
\end{aligned}$$

Combinando ambas desigualdades, se tiene que:

$$\begin{aligned}
\alpha_k &\geq \frac{\left(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\right)^T \mathbf{p}_k}{L\|\mathbf{p}_k\|^2} \\
\alpha_k &\geq \frac{(c_2 - 1)\nabla f(\mathbf{x}_k)^T \mathbf{p}_k}{L\|\mathbf{p}_k\|^2}.
\end{aligned}$$

Sustituyendo en la condición (3.13) tenemos:

$$\begin{aligned}
f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) &\leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k, \\
f(\mathbf{x}_{k+1}) &\leq f(\mathbf{x}_k) + c_1 \frac{(c_2 - 1)\nabla f(\mathbf{x}_k)^T \mathbf{p}_k}{L\|\mathbf{p}_k\|^2} \nabla f(\mathbf{x}_k)^T \mathbf{p}_k, \\
&= f(\mathbf{x}_k) - c_1 \frac{(1 - c_2)}{\left(\nabla f(\mathbf{x}_k)^T \mathbf{p}_k\right)^2} L\|\mathbf{p}_k\|^2.
\end{aligned}$$

Pero, notemos que:

$$\frac{-\nabla f(\mathbf{x}_k)^T \mathbf{p}_k}{\|\nabla f(\mathbf{x}_k)\|\|\mathbf{p}_k\|} = \cos\theta_k,$$

donde θ_k es el ángulo formado por p_k y $-\nabla f(\mathbf{x}_k)$. Entonces

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - c\cos^2\theta_k \|\nabla f(\mathbf{x}_k)\|^2,$$

con $c = c_1(1-c_2)/L$. Realizando la suma de estas expresiones, sobre todos los índices menores o iguales a k, obtenemos:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_0) - c\sum_{j=0}^{k} \cos^2\theta_j \|\nabla f(\mathbf{x}_j)\|^2.$$

Como f es acotaba por abajo, tenemos que $f(\mathbf{x}_0) - f(\mathbf{x}_{k+1})$ es menor que una constante positiva para todo k. Por lo tanto, tomando el límite cuando k tiende a infinito, llegamos a que

$$\sum_{k\geq 0} \cos^2\theta_k \|\nabla f(\mathbf{x}_k)\|^2 < \infty.$$

□

Observemos que las restricciones del teorema no son tan restrictivas. Si la función f no fuera acotada por debajo, el problema de optimización no estaría bien definido, y la condición de suavidad (es decir, continuidad Lipschitz del gradiente) también es necesaria para satisfacer las condiciones de optimalidad.

El teorema anterior, también implica que:

$$\cos^2\theta_k \|\nabla f_k\|^2 \to 0.$$

Por otro lado, si la dirección de descenso se elige siempre parecida a menos la dirección del gradiente de la función, tenemos que $\theta_k < 90°$ por tanto, existe un δ tal que

$$\cos\theta_k \geq \delta > 0, \text{ para todo } k.$$

Por lo tanto, debe ser que:

$$\lim_{k\leftarrow\infty} \|\nabla f(\mathbf{x}_k)\| = 0.$$

En otras palabras, podemos estar seguros que la normas del gradiente convergen a cero siempre y cuando las direcciones de descenso sean similares a la dirección de menos el gradiente.

No obstante, es importante señalar que el resultado anterior, no nos garantiza convergencia hacia los mínimos locales de la función, sino que el método es traido por puntos críticos (donde el gradiente se anula). Para asegurar convergencia, tal y como se deduce de las condiciones de optimalidad, es necesario incorporar una condición de curvatura dada por la segunda derivada.

Estudiaremos la tasa de convergencia del método de descenso más pronunciado en un caso ideal, en que la función es cuadrática, de la forma:

$$f(x) = \frac{1}{2}x^T Qx - b^T x,$$

donde Q es una matriz simétrica y positiva definida. El gradiente entonces está dado por:

$$\nabla f(x) = Qx - b,$$

y el minimizador x^* es la solución (única) del sistema lineal $Qx = b$.

Dado un punto x_k, la longitud del tamaño de paso α_k que minimiza $f(x_k - \alpha\nabla f(x_k))$ se obtiene derivando la función:

$$f(x_k - \alpha\nabla f(x_k)) = \frac{1}{2}(x_k - \alpha\nabla f(x_k))^T Q(x_k - \alpha\nabla f(x_k)) - b^T(x_k - \alpha\nabla f(x_k)),$$

con respecto a α e igualando a cero, obtenemos:

$$\alpha_k = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k) Q \nabla f(x_k)}.$$

Con este tamaño de paso, la iteración para el método de descenso más pronunciado está dada por:

$$x_{k+1} = x_k - \left(\frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k) Q \nabla f(x_k)}\right) \nabla f(x_k). \tag{3.15}$$

Como $\nabla f(x_k) = Qx_k - b$, de la ecuación anterior se obtiene una expresión cerrada para x_{k+1} en términos de x_k.

Para cuantificar la tasa de convergencia consideremos la norma $\|x\|_Q^2 = x^T Qx$. Usando el hecho de que $Qx^* = b$, se puede probar que

$$\frac{1}{2}\|x - x^*\|_Q^2 = f(x) - f(x^*),$$

entonces esta norma mide la diferencia entre el punto actual x y el óptimo x^*. Usando la iteración (3.15) y observando que $\nabla f(x_k) = Q(x_k - x^*)$, se obtiene

$$\|x_{k+1} - x^*\|_Q^2 = \left\{1 - \frac{(\nabla f(x_k)^T \nabla f(x_k))^2}{(\nabla f(x_k)^T Q \nabla f(x_k))(\nabla f(x_k)^T Q^{-1} \nabla f(x_k))}\right\} \|x_k - x^*\|_Q^2,$$

lo cual nos da el decremento exacto en f en cada iteración.

La ecuación anterior, se puede acotar como la desigualdad:

$$\|x_{k+1} - x^*\|_Q^2 \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2 \|x_k - x^*\|_Q^2 = r^2 \|x_k - x^*\|_Q^2,$$

donde $0 \leq \lambda_1 \leq \ldots \leq \lambda_n$ son los eigenvalores de Q_n y r^2 es constante. Esto nos indica que la tasa de convergencia del método de descenso más pronunciado es cuadrática.

En otras palabras, elegir la dirección de descenso similar a menos la dirección del gradiente, asegura que el método converge a un punto crítico. Por otro lado, por la forma como se elige el tamaño de paso, satisfaciendo las condiciones de suficiente descenso (Armijo o Wolfe) se asegura que la función reduce su valor a lo largo de las iteraciones, con tasa cuadrática.

3.7. Aproximación del gradiente con diferencias finitas.

Cuando el usuario no puede proveer o calcular el gradiente de la función f de manera exacta, se puede optar por utilizar diferencias finitas para aproximar las derivadas parciales correspondientes. A pesar de que dicho procedimiento sólo produce una aproximación del gradiente, los resultados obtenidos mediante esta estrategia suelen ser adecuados en la mayoría de las ocasiones [Wright et al., 1999].

El método más sencillo consiste en usar diferencias finitas adelantadas

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + \epsilon e_i) - f(x)}{\epsilon}. \tag{3.16}$$

El gradiente puede ser construido aplicando la fórmula (3.16) para $i = 1, 2, \ldots, n$. Este proceso requiere la evaluación de f en $n + 1$ puntos: x y $x + \epsilon e_i$, $i = 1, 2, \ldots, n$.

Si f es dos veces continuamente diferenciable, entonces por el Teorema de Taylor tenemos que

$$f(x + p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x + tp) p, \quad \text{para algún } t \in (0, 1). \tag{3.17}$$

Suponiendo que el Hessiano está acotado, es decir, $\|\nabla^2 f(\cdot)\| \leq L$ dentro de la región de interés. De esta manera se obtiene que

$$\|f(x + p) - f(x) - \nabla f(x)^T p\| \leq \frac{L}{2} \|p\|^2. \tag{3.18}$$

Ahora, eligiendo $p = \epsilon e_i$, esto es un pequeño cambio a lo largo de las direcciones canónicas e_i. Entonces $\nabla f(x)^T p = \nabla f(x)^T e_i = \frac{\partial f}{\partial x_i}$, de manera que la ecuación (3.18) nos lleva a la estimación

$$\frac{\partial f}{\partial x_i}(x) = \frac{f(x+\epsilon)e_i - f(x)}{\epsilon} + \delta_\epsilon, \quad \text{donde } |\delta_\epsilon| \leq (L/2)\epsilon. \tag{3.19}$$

Para implementar fórmula (3.16) es necesario elegir el parámetro ϵ, que de acuerdo a la ecuación (3.19), el término de error δ_ϵ al aproximar las derivadas parciales mediante diferencias finitas hacia adelante tiende a cero conforme ϵ es cada vez más pequeño, de manera que debemos elegir ϵ tan pequeño como sea posible. Desafortunadamente la estimación (3.19) no toma en cuenta los errores de redondeo al implementar la aproximación (3.16) con aritmética finita. Si $\mathbf{u}$ es la unidad de redondeo ($\mathbf{u} \approx 1{,}1 \times 10^{-16}$) [Wright et al., 1999], podemos suponer que el error cometido en la evaluación de f está acotado por $\mathbf{u}$ de la siguiente manera:

$$|\text{fl}(f(x)) - f(x)| \leq \mathbf{u}L_f$$
$$|\text{fl}(f(x+\epsilon e_i)) - f(x+\epsilon e_i)| \leq \mathbf{u}L_f$$

donde $\text{fl}(\cdot)$ denota el valor calculado y L_f es una cota para los valores de $|f(\cdot)|$ en la región de interés. Si se usan los valores calculados en lugar de los valores exactos en la fórmula (3.19), se obtiene que el error está acotado por $(L/2)\epsilon + 2\mathbf{u}L_f/\epsilon$. El valor de ϵ que minimiza dicha cota para el error está dado por $\epsilon^2 = \frac{4L_f\mathbf{u}}{L}$. Suponiendo que la razón L_f/L está acotada por un valor de tamaño moderado, se puede concluir la elección $\epsilon = \sqrt{\mathbf{u}}$ es cercana al valor óptimo. Esta elección del valor de ϵ implica que el error al usar la fórmula de diferencias finitas adelantadas para aproximar las derivadas parciales es aproximadamente $\sqrt{\mathbf{u}}$.

Una elección más precisa para la aproximación del gradiente consiste en usar diferencias finitas centradas, de la siguiente manera

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x+\epsilon e_i) - f(x-\epsilon e_i)}{2\epsilon}. \tag{3.20}$$

A partir del teorema de Taylor es posible mostrar que

$$\frac{\partial f}{\partial x_i}(x) = \frac{f(x+\epsilon e_i) - f(x-\epsilon e_i)}{2\epsilon} + O(\epsilon^2). \tag{3.21}$$

De la ecuación (3.21) se puede notar que el error al aproximar las derivadas con diferencias finitas centradas es $O(\epsilon^2)$, en contraste con el error $O(\epsilon)$ de la fórmula de diferencias finitas adelantadas (3.19). Un análisis similar al realizado con las diferencias finitas adelantadas arroja que una elección óptima del parámetro ϵ es $\text{u}^{1/3}$, en cuyo caso el error sería aproximadamente $\text{u}^{2/3}$.

Bibliografía

[Higham C. F., 2019] Higham C. F., . H. D. J. (2019). Deep learning: An introduction for applied mathematicians. *Siam review*, 61(4):860–891.

[Quarteroni et al., 2010] Quarteroni, A., Sacco, R., and Saleri, F. (2010). *Numerical mathematics*, volume 37. Springer Science & Business Media.

[Spivak, 2019] Spivak, M. (2019). *Calculus*. Reverté.

[Wright et al., 1999] Wright, S., Nocedal, J., et al. (1999). Numerical optimization. *Springer Science*, 35(67-68):7.