



UNIVERSIDAD DE CHILE

Deep Learning

Deeper, Better, _____, Stronger than Machine Learning

Valentin Barriere

Universidad de Chile – DCC

CC6204, Primavera 2025

Perceptron y Gradient Descendiente

Perceptron

Outline : Perceptron

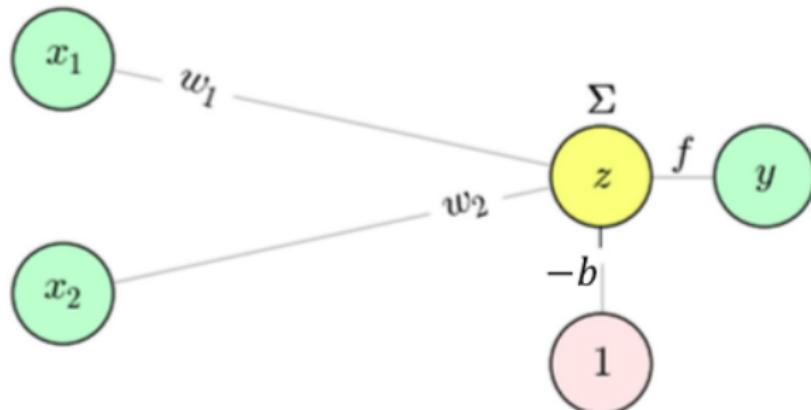
Perceptron

Gradiente Descendiente

Marco matemático

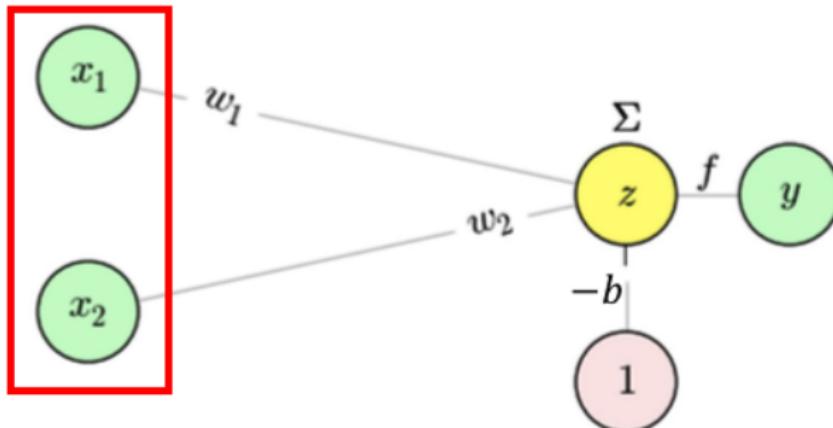
Optimización

Perceptrón – Presentación



$$y = f(x_1 w_1 + x_2 w_2 - b)$$

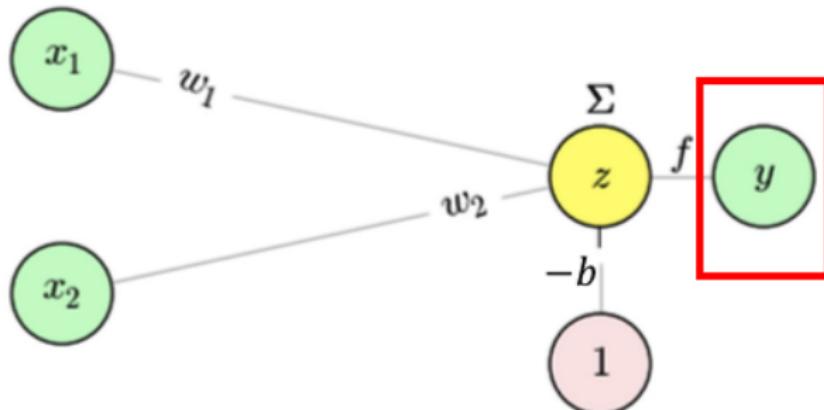
Perceptrón – Presentación



$$y = f(x_1 w_1 + x_2 w_2 - b)$$

Entradas binarias

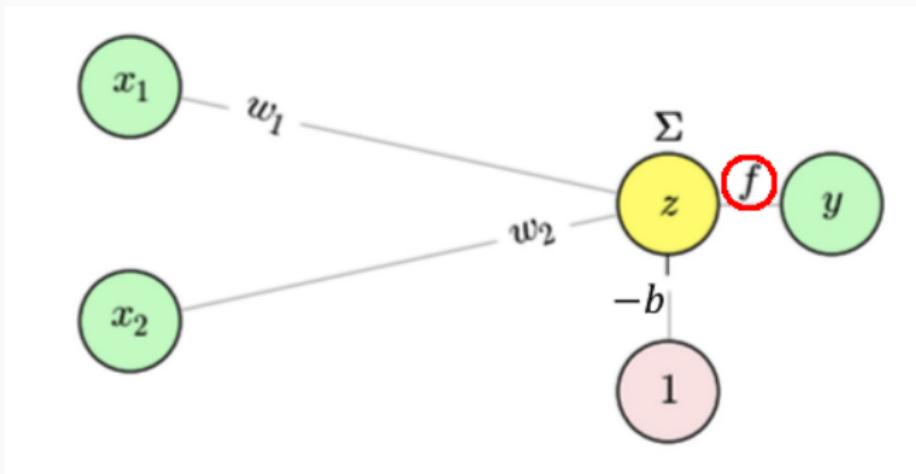
Perceptrón – Presentación



$$y = f(x_1 w_1 + x_2 w_2 - b)$$

Salida binaria

Perceptrón – Presentación

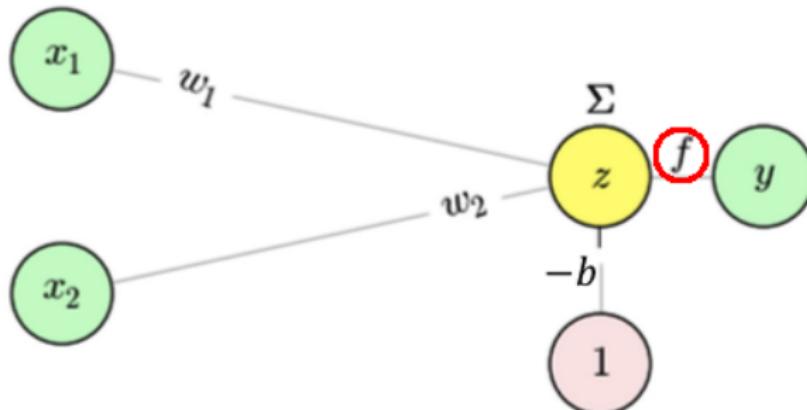


$$y = f(x_1 w_1 + x_2 w_2 - b)$$

Porque en teoría f es la función de Heaviside:

$$f(z) = \mathbb{1}_{\mathbb{R}_+} = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases}$$

Perceptrón – Presentación



$$y = f(x_1 w_1 + x_2 w_2 - b)$$

Lo que da:

$$y = \begin{cases} 1 & \text{si } x_1 w_1 + x_2 w_2 \geq b \\ 0 & \text{si } x_1 w_1 + x_2 w_2 < b \end{cases}$$

Funciones de activación

Función de Heaviside

- En teoría f es la función de Heaviside: $f(z) = \mathbb{1}_{\mathbb{R}_+} = \begin{cases} 0 & \text{si } z < 0 \\ 1 & \text{si } z \geq 0 \end{cases}$
- **Problema:** esta función **no es derivable**.

¹Unidad de Rectificación Lineal

²Unidad Exponencial Lineal

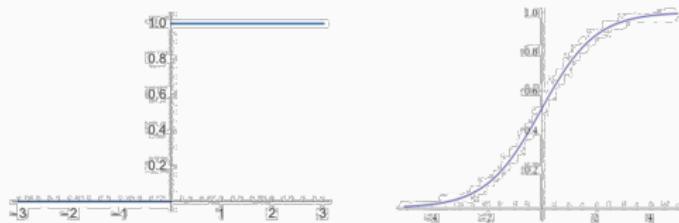
Funciones de activación

Función de Heaviside

- En teoría f es la función de Heaviside: $f(z) = \mathbb{1}_{\mathbb{R}_+} = \begin{cases} 0 & \text{si } z < 0 \\ 1 & \text{si } z \geq 0 \end{cases}$
- Problema:** esta función **no es derivable**.

Solución: Se puede reemplazar Heaviside por una función Sigmoidal!

- Es continua, derivable, de derivada continua
- Tiene un comportamiento similar



$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Funciones de activación

Función de Heaviside

- En teoría f es la función de Heaviside: $f(z) = \mathbb{1}_{\mathbb{R}_+} = \begin{cases} 0 & \text{si } z < 0 \\ 1 & \text{si } z \geq 0 \end{cases}$
- Problema:** esta función **no es derivable**.

En la práctica se utilizan funciones derivables como:

Nombre	Gráfico	Ecuación	Valores en $\pm\infty$	Valores derivada en $\pm\infty$
Sigmoide		$f(z) = \frac{1}{1+e^{-z}}$	0; 1	0; 0
Tangente hiperbólica		$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	-1; 1	0; 0
ReLU ¹		$f(z) = \begin{cases} 0 & \text{si } z < 0 \\ z & \text{si } z \geq 0 \end{cases}$	0; z	0; 1
ELU ²		$f(z) = \begin{cases} \alpha(e^z - 1) & \text{si } z < 0 \\ z & \text{si } z \geq 0 \end{cases}$	$-\alpha; z$	0; 1

¹Unidad de Rectificación Lineal

²Unidad Exponencial Lineal

Gradiente Descendiente

Outline : Gradiente Descendiente

Perceptron

Gradiente Descendiente

Marco matemático

Optimización

Outline : Marco matemático

Perceptron

Gradiente Descendiente

Marco matemático

Optimización

Aprendizaje supervisado I

Marco matemático

- Medida de entrada: $\mathbf{X} = (X^{(1)}, X^{(2)}, \dots, X^{(d)}) \in \mathcal{X}$
- Medida de salida: $Y \in \mathcal{Y}$
- $(\mathbf{X}, Y) \sim \mathbf{P}$ donde \mathbf{P} es desconocido
- Conjunto de entrenamiento: $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)\}$
- A menudo:
 - $\mathbf{X} \in \mathbb{R}^d$ y $Y \in [1, \dots, C]$ para una tarea de clasificación
 - $\mathbf{X} \in \mathbb{R}^d$ y $Y \in \mathbb{R}$ para una tarea de regresión

Un clasificador es una función en $\mathcal{F} = \{f : \mathcal{X} \rightarrow \mathcal{Y}\}$ (medible).

Objetivo

Construir un clasificador \hat{f} **satisfactorio** utilizando los datos de entrenamiento. **Atención:** \hat{f} depende de \mathcal{D}_n .

Aprendizaje, tarea, medida de rendimiento

- Conjunto de entrenamiento: $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)\}$
- Clasificador: $f : \mathcal{X} \rightarrow \mathcal{Y}$ (medible).
- Función de costo/pérdida: $\ell(Y, f(\mathbf{X}))$ mide la calidad de la *predicción* de f en relación con Y . Este función es **non negativa, cerca de cero cuando $f(\mathbf{X}) \approx Y$.**
- Riesgo:

$$\mathcal{R}(f) = \mathbb{E}[\ell(Y, f(\mathbf{X}))]$$

Objetivo

Aprender a construir un **clasificador** $f \in \mathcal{F}$ desde los datos de entrenamiento \mathcal{D}_n de manera que el **riesgo** $\mathcal{R}(f)$ sea **pequeño en promedio** en relación con \mathcal{D}_n .

Recordatorios de Machine Learning

- I Tener datos etiquetados
- II Extraer los descriptores = transformar documentos en vectores
- III Crear un modelo matemático f_θ
- VI Implementar una función de costo (error) ℓ a minimizar
- V Encontrar los parámetros θ^* de manera que $\ell(f_{\theta^*}(\mathbf{X}_i), Y_i)$ sea pequeño
- VI Probar f_{θ^*} en nuevos datos con una métrica de evaluación adecuada

Recordatorios de Machine Learning

I Tener datos etiquetados

- Conjunto de datos de tamaño n , $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- Y son las etiquetas (por ejemplo: monto del préstamo concedido)

II Extraer los descriptores = transformar documentos en vectores

- X es un vector de observaciones (por ejemplo: edad, sexo, salario)
- Y son las etiquetas (por ejemplo: monto del préstamo concedido)

III Crear un modelo matemático f_θ

- Modelo f_θ tal que $f_\theta(X)$ esté cerca de Y (para regresión)
- θ es el conjunto de parámetros del modelo matemático

VI Implementar una función de costo (error) ℓ a minimizar

V Encontrar los parámetros θ^* de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

VI Probar f_{θ^*} en nuevos datos con una métrica de evaluación adecuada

Recordatorios de Machine Learning

I Tener datos etiquetados

- Conjunto de datos de tamaño n , $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- Y son las etiquetas (por ejemplo: monto del préstamo concedido)

II Extraer los descriptores = transformar documentos en vectores

- X es un vector de observaciones (por ejemplo: edad, sexo, salario)
- Y son las etiquetas (por ejemplo: monto del préstamo concedido)

III Crear un modelo matemático f_θ

- Modelo f_θ tal que $f_\theta(X)$ esté cerca de Y (para regresión)
- θ es el conjunto de parámetros del modelo matemático

VI Implementar una función de costo (error) ℓ a minimizar

- Cuanto más se equivoque el modelo, mayor será el costo
- En general, se desea tener un costo pequeño

V Encontrar los parámetros θ^* de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

VI Probar f_{θ^*} en nuevos datos con una métrica de evaluación adecuada

Recordatorios de Machine Learning

I Tener datos etiquetados

- Conjunto de datos de tamaño n , $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- Y son las etiquetas (por ejemplo: monto del préstamo concedido)

II Extraer los descriptores = transformar documentos en vectores

- X es un vector de observaciones (por ejemplo: edad, sexo, salario)
- Y son las etiquetas (por ejemplo: monto del préstamo concedido)

III Crear un modelo matemático f_θ

- Modelo f_θ tal que $f_\theta(X)$ esté cerca de Y (para regresión)
- θ es el conjunto de parámetros del modelo matemático

VI Implementar una función de costo (error) ℓ a minimizar

- Cuanto más se equivoque el modelo, mayor será el costo
- En general, se desea tener un costo pequeño

V Encontrar los parámetros θ^* de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

- $\theta^* = \arg \min_{\theta} \sum_i \ell(f_{\theta}(X_i), Y_i)$

VI Probar f_{θ^*} en nuevos datos con una métrica de evaluación adecuada

Recordatorios de Machine Learning

I Tener datos etiquetados

- Conjunto de datos de tamaño n , $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- Y son las etiquetas (por ejemplo: monto del préstamo concedido)

II Extraer los descriptores = transformar documentos en vectores

- X es un vector de observaciones (por ejemplo: edad, sexo, salario)
- Y son las etiquetas (por ejemplo: monto del préstamo concedido)

III Crear un modelo matemático f_θ

- Modelo f_θ tal que $f_\theta(X)$ esté cerca de Y (para regresión)
- θ es el conjunto de parámetros del modelo matemático

VI Implementar una función de costo (error) ℓ a minimizar

- Cuanto más se equivoque el modelo, mayor será el costo
- En general, se desea tener un costo pequeño

V Encontrar los parámetros θ^* de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

- $\theta^* = \arg \min_{\theta} \sum_i \ell(f_{\theta}(X_i), Y_i)$

VI Probar f_{θ^*} en nuevos datos con una métrica de evaluación adecuada

Outline : Optimización

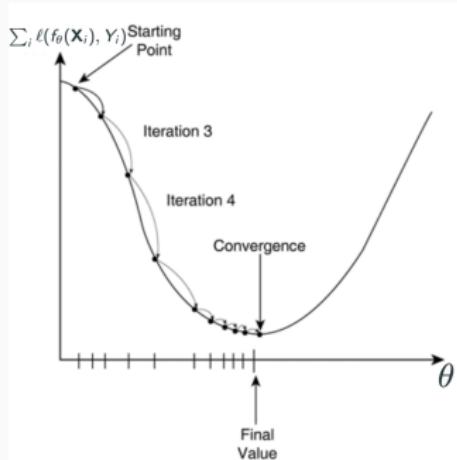
Perceptron

Gradiente Descendiente

Marco matemático

Optimización

Optimización



Optimización de la función de costo

- Sirve para converger al valor mínimo de la función de costo en el conjunto de datos de entrenamiento
- Mejor caso: rápido y preciso
- A menudo se hacen aproximaciones para ser más rápidos

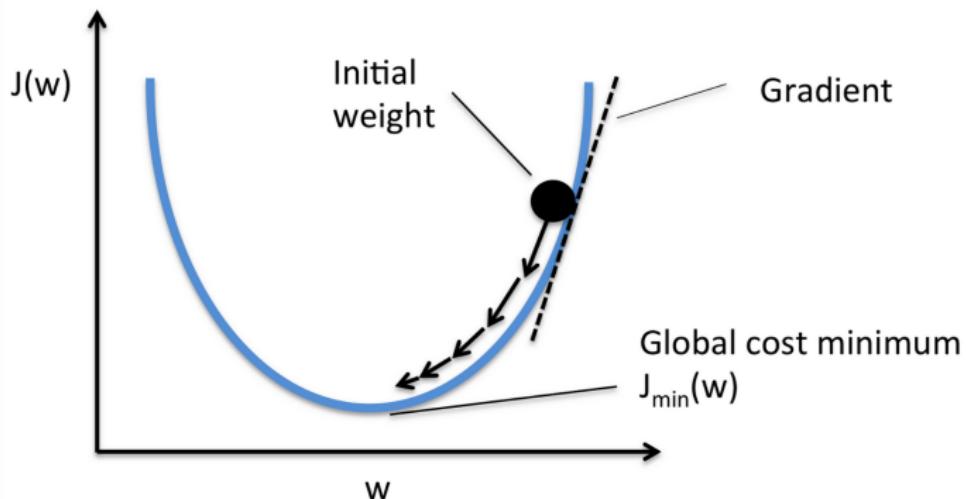
Gradiente

Una función multivariante $f(x)$ puede escribirse como una serie de Taylor:

$$f(x + \delta_x) = f(x) + \nabla_x f(x)\delta_x + \mathcal{O}(\|\delta_x^2\|)$$

Definición

El gradiente de una función $\nabla_x f(x) = (\frac{\partial f}{\partial x_i})_{i=1..n}$ es su derivativa según cada dimensión. Es una **aproximación lineal de la función al nivel local**.



Gradiente

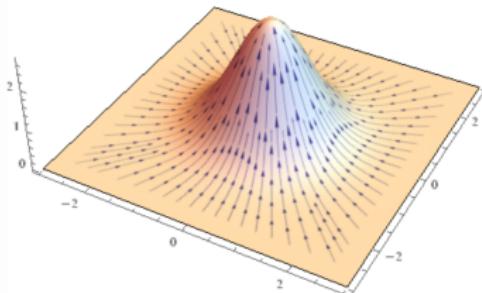
Una función multivariante $f(x)$ puede escribirse como una serie de Taylor:

$$f(x + \delta_x) = f(x) + \nabla_x f(x)\delta_x + \mathcal{O}(\|\delta_x\|^2)$$

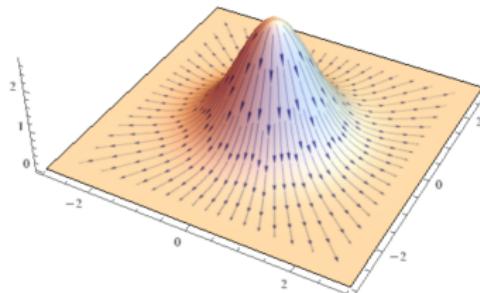
Definición

El gradiente de una función $\nabla_x f(x) = (\frac{\partial f}{\partial x_i})_{i=1..n}$ es su derivativa según cada dimensión. Es una **aproximación lineal de la función al nivel local**.

Moving towards gradient

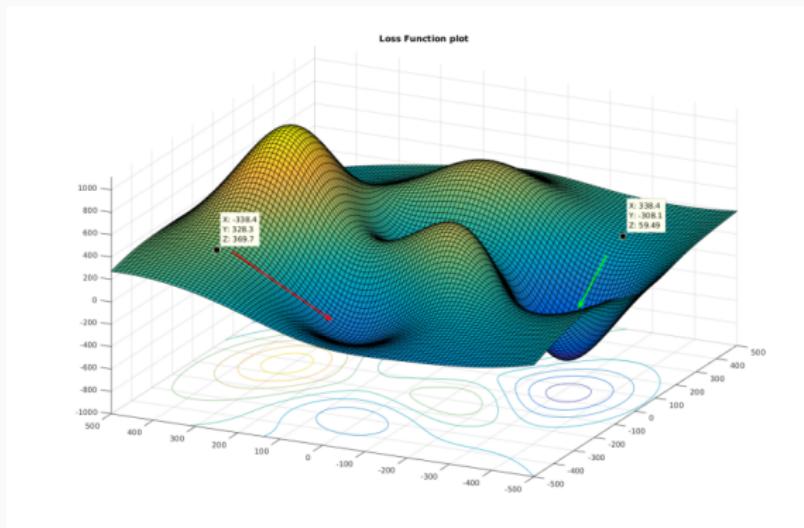


Moving opposite to gradient



Más detalles [aca](#). Idea: $\Delta f = f(x + \delta_x) - f(x) \approx \nabla_x f(x)\delta_x$.
 $\delta_x = -\lambda \nabla_x f(x) \Rightarrow \Delta f = -\lambda \|\nabla_x f(x)\|^2 < 0$.

Visualización de la función de costo: Loss Landscape



Se puede visualizar la función de costo $\ell : \mathbb{R}^d \rightarrow \mathbb{R}$ como una superficie:

- Los valores de los parámetros θ varían en el plano, y el valor de la función $\ell(\mathcal{D}_n; \theta)$ varía en altura.
- La convergencia se produce cuando se tienen parámetros que están en un hueco de esta superficie (mínimo local o global, dependiendo del modelo)

Optimización : visualización

Optimización : Gradiente descendiente

Descenso del gradiente

Después de cada cálculo de la función de costo $\ell(Y_i, f_\theta(\mathbf{X}_i); \theta)$, se calcula el gradiente de esta función para actualizar los parámetros θ :

$$\theta \leftarrow \theta - \alpha * \nabla_{\theta} \ell(Y_i, f_\theta(\mathbf{X}_i); \theta)$$

Ejemplo con un modelo lineal

- Sean $f_\theta(\mathbf{X}) = \theta^T \mathbf{X} = \sum_k^d \theta_k X^{(k)}$ y $\ell(Y, f_\theta(\mathbf{X})) = \frac{1}{2}(Y - f_\theta(\mathbf{X}))^2$

$$\nabla_{\theta} \ell(Y_i, f_\theta(\mathbf{X}_i); \theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_d} \end{pmatrix} . \ell(Y_i, f_\theta(\mathbf{X}_i); \theta) = \begin{pmatrix} X_i^{(1)}(Y - f_\theta(\mathbf{X}_i)) \\ \vdots \\ X_i^{(d)}(Y - f_\theta(\mathbf{X}_i)) \end{pmatrix}$$

Optimización : Gradiente descendiente

Descenso del gradiente

Después de cada cálculo de la función de costo $\ell(Y_i, f_\theta(\mathbf{X}_i); \theta)$, se calcula el gradiente de esta función para actualizar los parámetros θ :

$$\theta \leftarrow \theta - \alpha * \nabla_{\theta} \ell(Y_i, f_\theta(\mathbf{X}_i); \theta)$$

Ejemplo con un modelo lineal

- Sean $f_\theta(\mathbf{X}) = \theta^T \mathbf{X} = \sum_k^d \theta_k X^{(k)}$ y $\ell(Y, f_\theta(\mathbf{X})) = \frac{1}{2}(Y - f_\theta(\mathbf{X}))^2$

$$\begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \end{pmatrix} \leftarrow \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \end{pmatrix} - \alpha * \begin{pmatrix} X_i^{(1)}(Y - f_\theta(\mathbf{X}_i)) \\ \vdots \\ X_i^{(d)}(Y - f_\theta(\mathbf{X}_i)) \end{pmatrix}$$

Regla de la Cadena

Usando:

$$\frac{\partial \ell}{\partial \mathbf{w}_1} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_1} \quad (1)$$

Computemos las derivadas: $\ell = \frac{1}{2}(\hat{y} - y)^2$

Derivada de ℓ con respecto al valor de salida: $\frac{\partial \ell}{\partial \hat{y}} = \hat{y} - y$

Derivada de z con respecto a w_1

$$z = w_1 x_1 + w_2 x_2 + b \quad (2)$$

$$\frac{\partial z}{\partial w_1} = x_1 \quad (3)$$

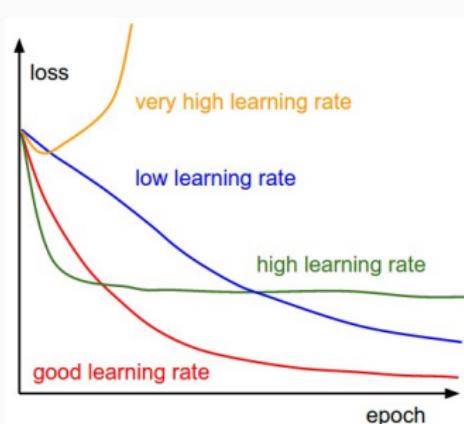
Sabiendo que $\hat{y} = (1 + e^{-z})^{-1}$, obtenemos: $\frac{\partial \hat{y}}{\partial z} = \sigma(z)(1 - \sigma(z))$ que da:

$$\frac{\partial \ell}{\partial w_1} = (\hat{y} - y)\sigma(z)(1 - \sigma(z)) \cdot x_1$$

Optimización : Importancia de la tasa de aprendizaje

Learning rate

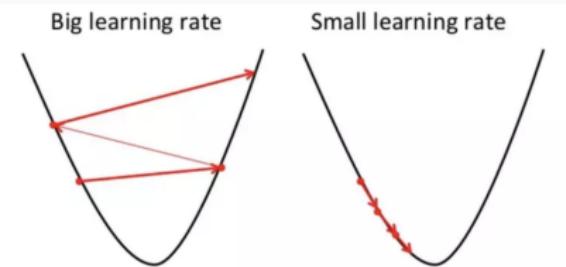
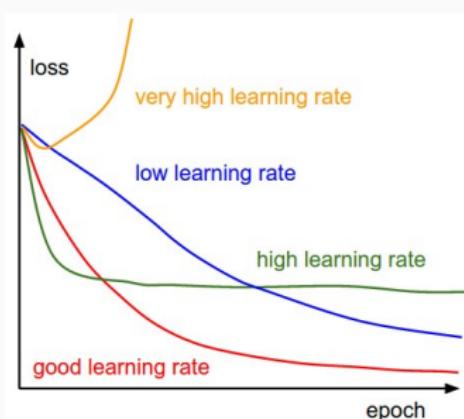
- Un α demasiado pequeño no avanza en el aprendizaje
- Un α demasiado pequeño alarga el tiempo de entrenamiento
- Un α demasiado grande no permite alcanzar el mínimo (damos vueltas alrededor del agujero de la superficie)
- Un α demasiado grande no permite nada
- Una solución: disminuir α con el tiempo



Optimización : Importancia de la tasa de aprendizaje

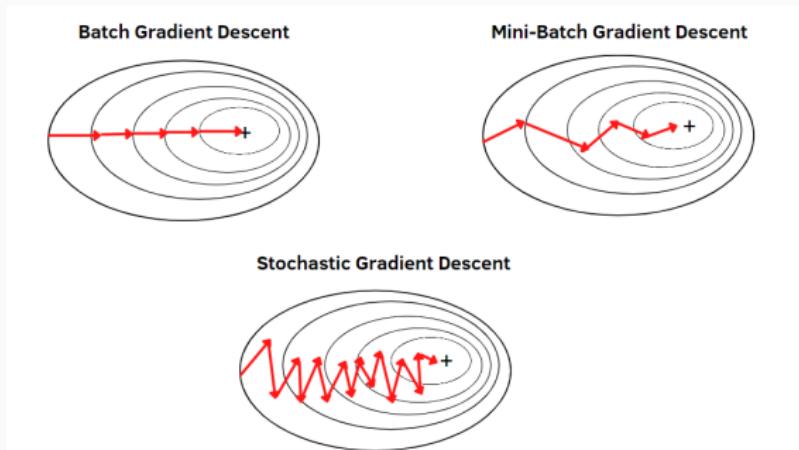
Learning rate

- Un α demasiado pequeño no avanza en el aprendizaje
- Un α demasiado pequeño alarga el tiempo de entrenamiento
- Un α demasiado grande no permite alcanzar el mínimo (damos vueltas alrededor del agujero de la superficie)
- Un α demasiado grande no permite nada
- Una solución: disminuir α con el tiempo



Gradiente descendiente estocástico // Stochastic Gradient Descent

- Estas operaciones se pueden hacer en paralelo \Rightarrow GPU!
- No se puede utilizar todo el dataset de train para updatear los parametros del modelo, se usa la técnica de los **mini-batch**
- Vamos a updatear los pesos del modelo calculando el error sobre **m** ejemplos que vamos a tomar de manera aleatoria en el dataset
- Mas **m** es grande, menos ruido en la convergencia



Algoritmo

Parametros: dataset \mathcal{D}_n , learning rate α , tamaño mini-batch m , cantidad de épocas

Para cada época e

Para cada mini-batch X_k

- Computar $f(X_k)$
- Computar el costo $\ell(f(X_k), Y)$
- Computar los gradientes
- Actualizar parámetros

Computar el costo promedio

Algoritmos de gradiente descendiente

Existen otros algoritmos existentes que se basan en un descenso del gradiente estocástico (SGD) con especificidades para mejorar su eficacia:

- Descenso del gradiente estocástico con momento
- Gradiente acelerado de Nestorov (NAG)
- Gradiente adaptativo (AdaGrad)
- Adam
- RMSprop

Finalmente, también existen otros métodos más clásicos de descenso: BFGS, L-BFGS, Quasi Newton, ...

Algoritmos de descenso del gradiente

Questions?

References i