

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Estadística y computación para metagenómica

Victor Muñiz Sánchez

victor_m@cimat.mx

Centro de Investigación en Matemáticas.
Unidad Monterrey.

Modelos de clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Modelos de clasificación

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Clasificación lineal I

Ya mencionamos antes que tradicionalmente consideramos dos problemas de predicción en ML:

Clasificación:

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n; \quad \mathbf{x} \in \mathbb{R}^d, y \in \{1, 2, \dots, K\},$$

Regresión:

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n; \quad \mathbf{x} \in \mathbb{R}^d, y \in \mathbb{R}.$$

Muchos de los métodos de ML pueden abordar ambos problemas. Algunos han surgido como métodos de clasificación y posteriormente se han adaptado a problemas de regresión. En la otra dirección resulta más complicado...

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Clasificación lineal II

En ambos casos, lo que queremos es encontrar una función f tal que, por ejemplo, para un problema de clasificación:

$$f : \mathbb{R}^d \mapsto \{1, 2, \dots, K\},$$

Una forma de representar un clasificador es mediante funciones discriminantes $f(\mathbf{x})$. Para el caso general de clasificación en K clases, el clasificador **asigna** un objeto \mathbf{x} a la clase k si

$$f_i(\mathbf{x}) > f_j(\mathbf{x}), \quad \forall j \neq i, i = 1, 2, \dots, K$$

Clasificación lineal III

Victor Muñiz

Modelos de
clasificación

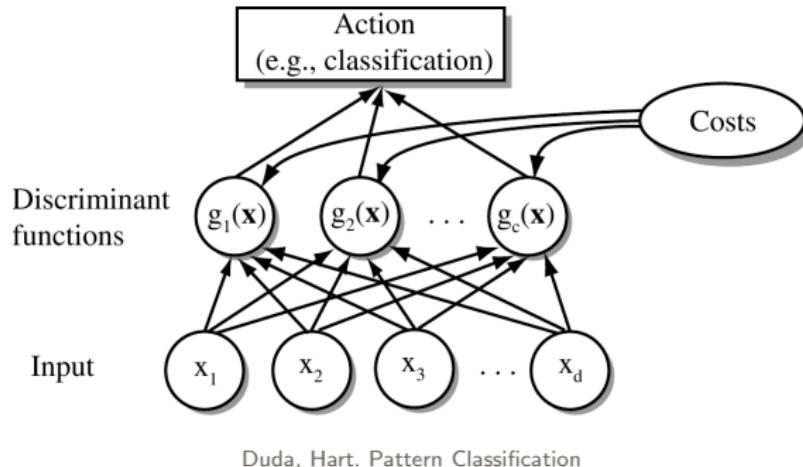
Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP



Si pudiéramos de alguna forma asignar $f_i(\mathbf{x}) = P(y = i|\mathbf{x})$, podríamos usar el criterio del clasificador óptimo Bayesiano, ya que la función discriminante máxima corresponderá a la probabilidad posterior máxima.

Clasificación lineal IV

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

En éste caso, las siguientes expresiones son equivalentes:

$$\begin{aligned}g_i(\mathbf{x}) &= P(y = i | \mathbf{x}) = \frac{P(\mathbf{x}|y=i)P(y=i)}{P(\mathbf{x})} \\g_i(\mathbf{x}) &= P(\mathbf{x}|y = i)P(y = i) \\g_i(\mathbf{x}) &= \log P(\mathbf{x}|y = i) + \log P(y = i)\end{aligned}$$

Sin embargo, no siempre es fácil asignar ésas probabilidades.

Clasificación lineal V

Modelos de
clasificación

Clasificación lineal

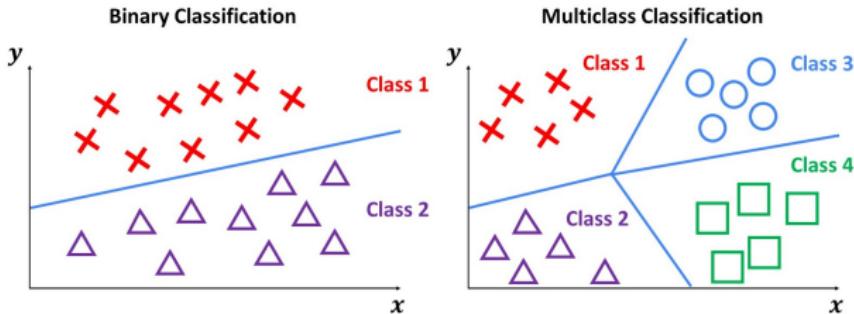
Regresión logística

Redes neuronales

Introducción

MLP

Geométricamente, la solución corresponde a una partición del espacio de las variables independientes.



Un ejemplo donde podemos asignar probabilidades directamente, es asumir cierta distribución de probabilidad $P(\mathbf{x}|y)$. El caso más simple, es cuando asumimos una distribución Gaussiana.

Clasificación lineal VI

Ejemplo: funciones discriminantes para la distribución normal (LDA, QDA)

Sin duda, el caso más estudiado es cuando se considera que nuestros datos provienen de una distribución normal:

$$X|y_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i).$$

Es decir,

$$P(\mathbf{x}|y = i) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp \left[\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) \right],$$

Simplificando la notación $y = i$ como y_i , las funciones discriminantes tienen la forma:

$$\begin{aligned} g_i(\mathbf{x}) &= \log P(\mathbf{x}|y_i) + \log P(y_i) \\ &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) - \frac{d}{2}\log 2\pi - \frac{1}{2}\log |\boldsymbol{\Sigma}_i| \\ &\quad + \log P(y_i) \end{aligned}$$

3 casos principales:

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Clasificación lineal VII

- Varianzas iguales (LDA): $\Sigma_i = \sigma^2 \mathbf{I}$.

$$g_i(\mathbf{x}) = -\frac{\|\mathbf{x} - \boldsymbol{\mu}_i\|^2}{2\sigma^2} + \log P(y_i)$$

$$g_i(\mathbf{x}) = \mathbf{w}'_i \mathbf{x} + w_{i0},$$

con

$$\begin{aligned}\mathbf{w}_i &= \frac{1}{\sigma^2} \boldsymbol{\mu}_i \\ w_{i0} &= -\frac{1}{2\sigma^2} \boldsymbol{\mu}'_i \boldsymbol{\mu}_i + \log P(y_i)\end{aligned}$$

Clasificación lineal VIII

Victor Muñiz

Modelos de
clasificación

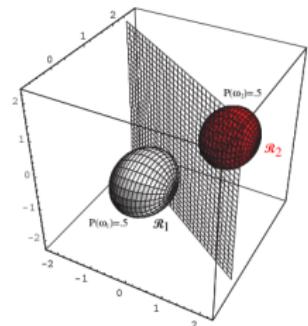
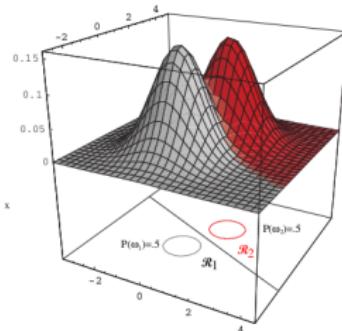
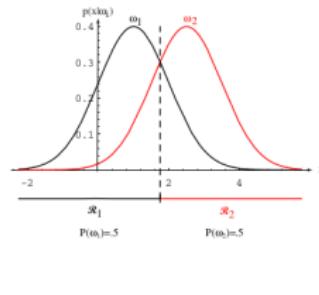
Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP



Duda, Hart. Pattern Classification

Clasificación lineal IX

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

- Varianzas iguales (LDA): $\Sigma_i = \Sigma$.

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) + \log P(y_i).$$

$$g_i(\mathbf{x}) = \mathbf{w}'_i \mathbf{x} + w_{i0},$$

donde

$$\begin{aligned}\mathbf{w}_i &= \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i \\ w_{i0} &= -\frac{1}{2} \boldsymbol{\mu}'_i \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i + \log P(y_i).\end{aligned}$$

Clasificación lineal X

Victor Muñiz

Modelos de
clasificación

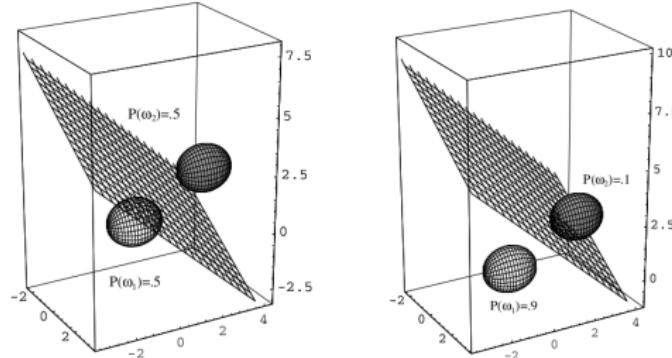
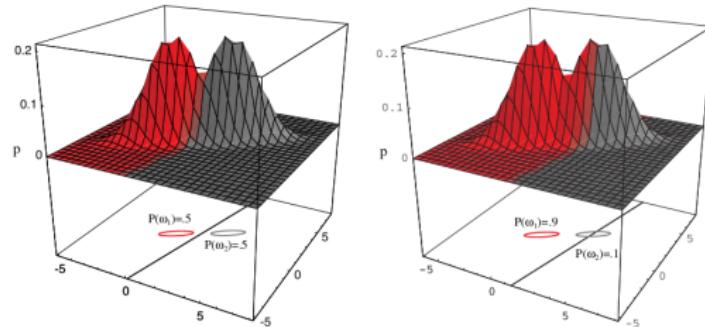
Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP



Clasificación lineal XI

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Duda, Hart. Pattern Classification

- Varianzas Σ_i son arbitrarias (QDA):

$$g_i(\mathbf{x}) = \mathbf{x}' \mathbf{W}_i' \mathbf{x} + \mathbf{w}_i' \mathbf{x} + w_{i0},$$

con

$$\mathbf{W}_i = -\frac{1}{2} \boldsymbol{\Sigma}_i^{-1}$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i$$

$$w_{i0} = -\frac{1}{2} \boldsymbol{\mu}_i' \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i - \frac{1}{2} \log |\boldsymbol{\Sigma}_i| + \log P(y_i)$$

Clasificación lineal XII

Modelos de
clasificación

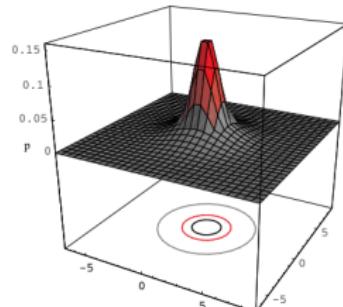
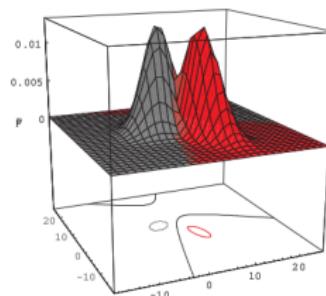
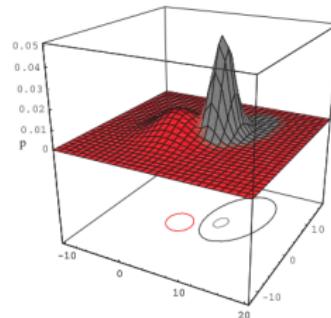
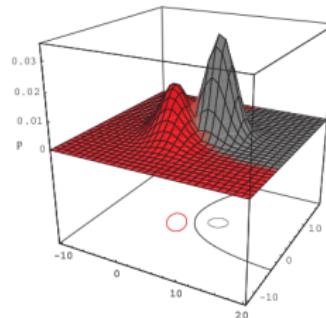
Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP



Duda, Hart. Pattern Classification

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

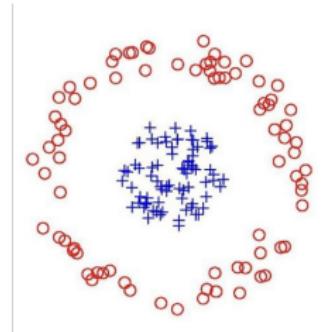
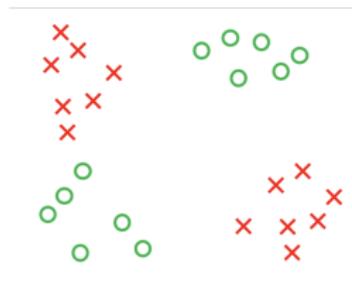
Clasificación lineal XIII

Resumen entonces, el hiperplano:

Divide el espacio de los datos. Su orientación está dada por β y su localización por β_0 .

Problemas:

- Solo para dos clases linealmente separables



- La solución no es única

Clasificación lineal XIV

Victor Muñiz

Modelos de
clasificación

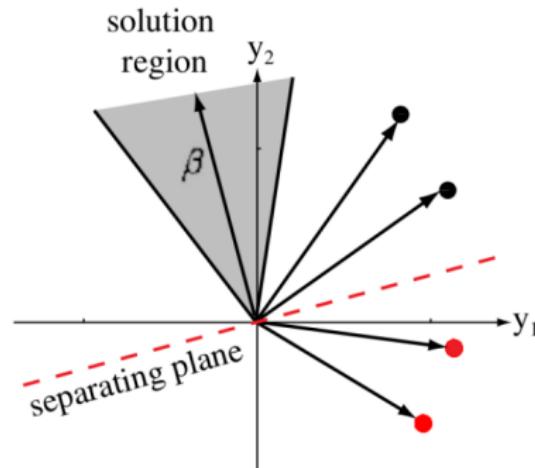
Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Métodos de clasificación

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Regresión logística

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Regresión logística

Para clasificación multiclase en general, recuerda que, según la regla de Bayes, asignamos una categoría estimada mediante

$$\hat{y} = \arg \max_k P(y = k | \mathbf{x}).$$

Nosotros modelamos las probabilidades posteriores mediante una función discriminante

$$f_k(\mathbf{x}) = P(y = k | \mathbf{x}).$$

Para ambos casos, ya vimos que la frontera de decisión, por ejemplo, para dos clases con etiquetas, 0 y 1 es

$$P(y = 1 | \mathbf{x}) = P(y = 0 | \mathbf{x}) \iff f_1(\mathbf{x}) = f_0(\mathbf{x}).$$

Regresión logística

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Para que la relación anterior sea válida, hace falta resolver un punto importante, ya que, en general

$$f_k(\mathbf{x}) \in \mathbb{R}$$

y por definición

$$P(y = k | \mathbf{x}) \in [0, 1]$$

Regresión logística

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Para que la relación anterior sea válida, hace falta resolver un punto importante, ya que, en general

$$f_k(\mathbf{x}) \in \mathbb{R}$$

y por definición

$$P(y = k|\mathbf{x}) \in [0, 1]$$

Para el caso binario, es fácil resolver tal relación, ya que en este caso Y es una v.a. Bernoulli con parámetro $p = P(y = 1)$, y en consecuencia, $P(y = 0) = 1 - p$.

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Regresión logística

En regresión logística, modelamos **linealmente** el logaritmo de los momios (odds ratio) de las probabilidades posteriores respecto a una probabilidad de referencia:

$$\log \frac{P(y = 1|\mathbf{x})}{P(y = 0|\mathbf{x})} = \beta_0 + \boldsymbol{\beta}'\mathbf{x}$$

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Regresión logística

En regresión logística, modelamos **linealmente** el logaritmo de los momios (odds ratio) de las probabilidades posteriores respecto a una probabilidad de referencia:

$$\log \frac{P(y = 1|\mathbf{x})}{P(y = 0|\mathbf{x})} = \beta_0 + \boldsymbol{\beta}'\mathbf{x}$$

¿Porqué?

Regresión logística

Modelos de
clasificación

Clasificación lineal

Regresión logística

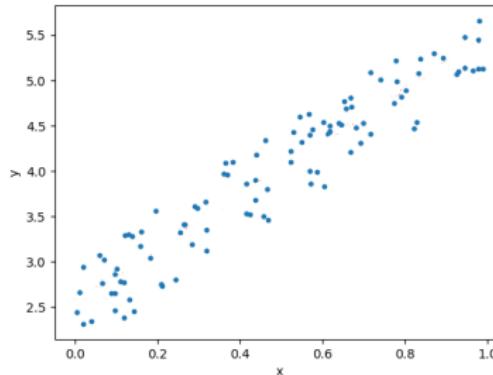
Redes neuronales

Introducción

MLP

GLM revisitado.

Recuerda el modelo de regresión lineal simple:



$$\hat{y} = f(\mathbf{x}) + \epsilon.$$

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

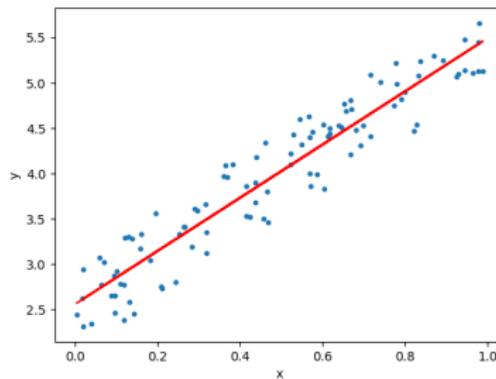
Introducción

MLP

Regresión logística

GLM revisitado.

Recuerda el modelo de regresión lineal simple:



$$\hat{y} = f(\mathbf{x}) + \epsilon.$$

En OLS:

$$\hat{y} = \boldsymbol{\beta}' \mathbf{x} + \epsilon, \quad \text{con } \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Regresión logística

Modelos de
clasificación

Clasificación lineal

Regresión logística

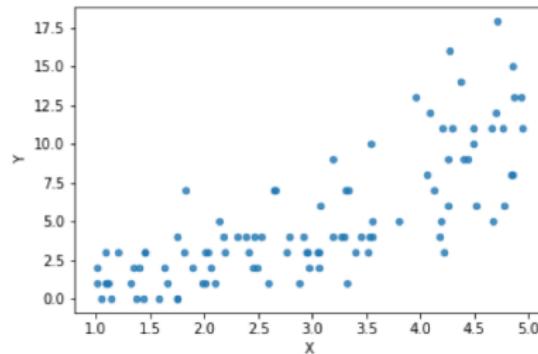
Redes neuronales

Introducción

MLP

GLM revisitado.

Pero no siempre son válidos éstos supuestos...



$$\hat{y} = f(\mathbf{x}) + \epsilon.$$

Regresión logística

GLM revisitado.

La generalización (GLM, McCullagh et al.)

- ➊ El componente aleatorio: La distribución de probabilidad de la variable dependiente. E.g., en OLS:

$$\hat{y} \sim \mathcal{N}(\mu, \sigma^2)$$

Regresión logística

GLM revisitado.

La generalización (GLM, McCullagh et al.)

- ➊ El componente aleatorio: La distribución de probabilidad de la variable dependiente. E.g., en OLS:

$$\hat{y} \sim \mathcal{N}(\mu, \sigma^2)$$

- ➋ El componente sistemático: modela las variables independientes. E.g., en OLS:

$$\eta = \beta' \mathbf{x}$$

Regresión logística

GLM revisitado.

La generalización (GLM, McCullagh et al.)

- ➊ El componente aleatorio: La distribución de probabilidad de la variable dependiente. E.g., en OLS:

$$\hat{y} \sim \mathcal{N}(\mu, \sigma^2)$$

- ➋ El componente sistemático: modela las variables independientes. E.g., en OLS:

$$\eta = \beta' \mathbf{x}$$

- ➌ El **enlace** entre los componentes aleatorio y sistemático. Indica cómo se relaciona $E(\hat{y}) = \mu$ con η . E.g., en OLS:

$$\eta = g(\mu) = \mu,$$

con $g(\cdot)$ la función link.

Regresión logística

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Regresando... En regresión logística, modelamos **linealmente** el logaritmo de los momios (odds ratio) de las probabilidades posteriores respecto a una probabilidad de referencia:

$$\log \frac{P(y = 1|\mathbf{x})}{P(y = 0|\mathbf{x})} = \boldsymbol{\beta}'\mathbf{x} + \beta_0$$

Entonces:

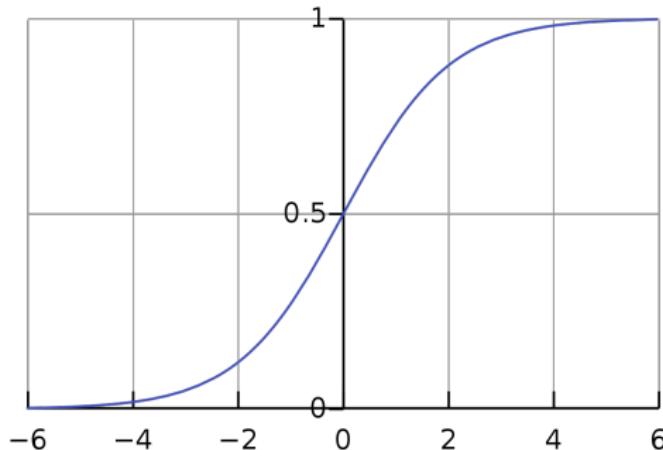
$$P(y = 1|\mathbf{x}) = \frac{e^{\boldsymbol{\beta}'\mathbf{x} + \beta_0}}{1 + e^{\boldsymbol{\beta}'\mathbf{x} + \beta_0}}$$

$$P(y = 0|\mathbf{x}) = \frac{1}{1 + e^{\boldsymbol{\beta}'\mathbf{x} + \beta_0}}$$

Regresión logística

De esa forma, podemos comparar y “umbralizar” la probabilidad de que x pertenezca a alguna categoría mediante la función sigmoide:

$$\sigma(u) = \frac{e^u}{1 + e^u} = \frac{1}{1 + e^{-u}} \in (0, 1)$$



Regresión logística

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Para K clases (multilogit), es fácil ver que:

$$P(y = k|\mathbf{x}) = \frac{e^{\beta_{k0} + \boldsymbol{\beta}'_k \mathbf{x}}}{1 + \sum_{i=1}^{K-1} e^{\beta_{i0} + \boldsymbol{\beta}'_i \mathbf{x}}}, \quad k = 1, 2, \dots, K-1$$

y

$$P(y = K|\mathbf{x}) = \frac{1}{1 + \sum_{i=1}^{K-1} e^{\beta_{i0} + \boldsymbol{\beta}'_i \mathbf{x}}}$$

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Regresión logística

Ajuste: máxima verosimilitud.

$$l(\boldsymbol{\theta}) = \sum_{i=1}^n \log P_k(\mathbf{x}_i; \boldsymbol{\theta}),$$

donde $P_k(\mathbf{x}_i; \boldsymbol{\theta}) = P(y = k | \mathbf{x}_i; \boldsymbol{\theta})$.

Regresión logística

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Ajuste: máxima verosimilitud.

$$l(\boldsymbol{\theta}) = \sum_{i=1}^n \log P_k(\mathbf{x}_i; \boldsymbol{\theta}),$$

donde $P_k(\mathbf{x}_i; \boldsymbol{\theta}) = P(y = k | \mathbf{x}_i; \boldsymbol{\theta})$.

Los métodos de ajuste generalmente son dos:

- Iteratively reweighted least squares (IRLS), usando Newton-Raphson
- Gradiente descendente estocástico

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Redes neuronales

Introducción

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Considera una función discriminante lineal ajustada en el conjunto de datos de entrenamiento.

Para iniciar, considera un problema de clasificación binario, donde $y \in \{0, 1\}$.

Ajustamos un modelo lineal

$$f(\mathbf{x}) = \beta_0 + \mathbf{x}'\boldsymbol{\beta},$$

y obtenemos

$$\hat{y} = \begin{cases} 0 & \text{si } f(\mathbf{x}) < \theta \\ 1 & \text{si } f(\mathbf{x}) > \theta \end{cases}$$

Introducción

Victor Muñiz

Modelos de
clasificación

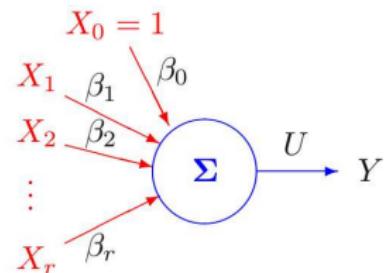
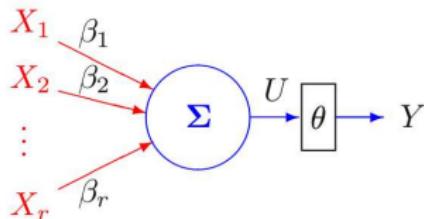
Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP



Aquí, $U = \mathbf{x}'\boldsymbol{\beta}$, el cual es comparado con el valor de referencia o umbral θ (figura izquierda).

Si hacemos $\beta_0 = -\theta$ y $x_0 = 1$, entonces la umbralización se realiza en cero (figura derecha).

U es una función que realiza una transformación a $f(\mathbf{x})$. En éste caso, es una función identidad, pero puede tener otras formas.

Modelos de
clasificación

Clasificación lineal

Regresión logística

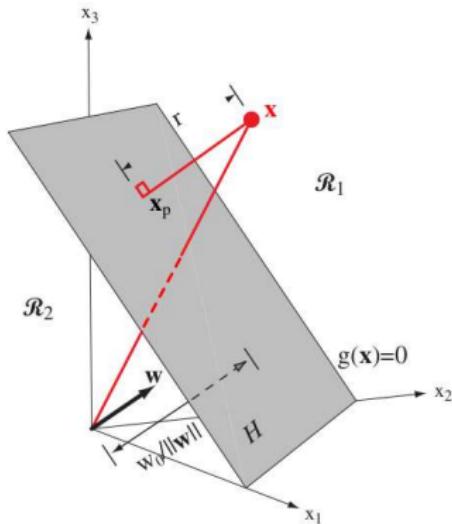
Redes neuronales

Introducción

MLP

Introducción

Geométricamente (solo cambia g por f y w por β):



El hiperplano:

Divide el espacio de los datos. Su orientación está dada por β y su localización por β_0 .

Introducción

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Si $f(\mathbf{x}) = 0$, implica una re-etiquetación de las categorías como $y \in \{-1, 1\}$, y

$$\hat{y} = sign(g(\mathbf{x})).$$

Introducción

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Si $f(\mathbf{x}) = 0$, implica una re-etiquetación de las categorías como $y \in \{-1, 1\}$, y

$$\hat{y} = \text{sign}(g(\mathbf{x})).$$

El modelo anterior es llamado **perceptrón**, y se ajusta minimizando la siguiente función de costo:

$$L(\boldsymbol{\beta}) = - \sum_{i \in MC} (\mathbf{x}' \boldsymbol{\beta}) y_i,$$

donde $y \in \{-1, 1\}$ y MC es el conjunto de observaciones mal clasificadas.

Introducción

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

La optimización para el modelo perceptrón, se basa en descenso por gradiente:

- 1: Input: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\boldsymbol{\beta}^0$, learning rate η^0 criterio de paro ϵ .
- 2: **repeat**
- 3: Calcular

$$\boldsymbol{\beta}^t = \boldsymbol{\beta}^{t-1} - \eta^t \nabla L(\boldsymbol{\beta})$$

- 4: **until**

$$|\eta^t \nabla L(\boldsymbol{\beta})| < \epsilon$$

- 5: Output: solución $\boldsymbol{\beta}$

Lo que dio origen a un modelo muy general de ajuste llamado **backpropagation**, que veremos un poco más adelante.

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Perceptron

Antecedentes:

Frank Rosenblatt. "The Perceptron - A Perceiving and Recognizing Automaton." Report 85-460-1. Cornell Aeronautical Laboratory. 1957



Perceptron

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

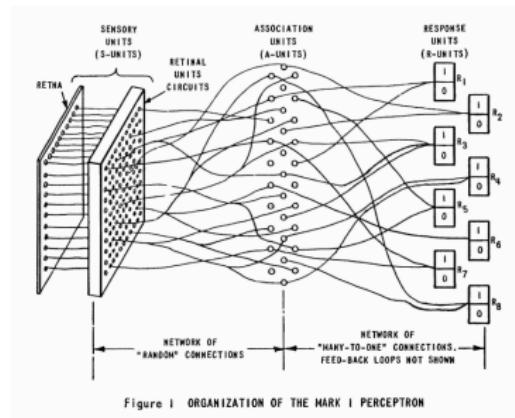
Regresión logística

Redes neuronales

Introducción

MLP

Mark I Perceptron, Rosenblatt 1958. Diseñado para reconocer imágenes. Una aplicación fue intentar reconocer el género de una persona a través de su fotografía.



Fuente: Mark I Perceptron Operators' Manual, Cornell Aeronautical Lab., Inc. 1960

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Perceptron

Antecedentes.

Configuración: arreglo de 20×20 fotoceldas de sulfuro de cadmio (imágenes de 400 "pixeles"). Configuración de diferentes características de entrada mediante conexiones por cables. Pesos adaptativos mediante potenciómetros para los inputs.

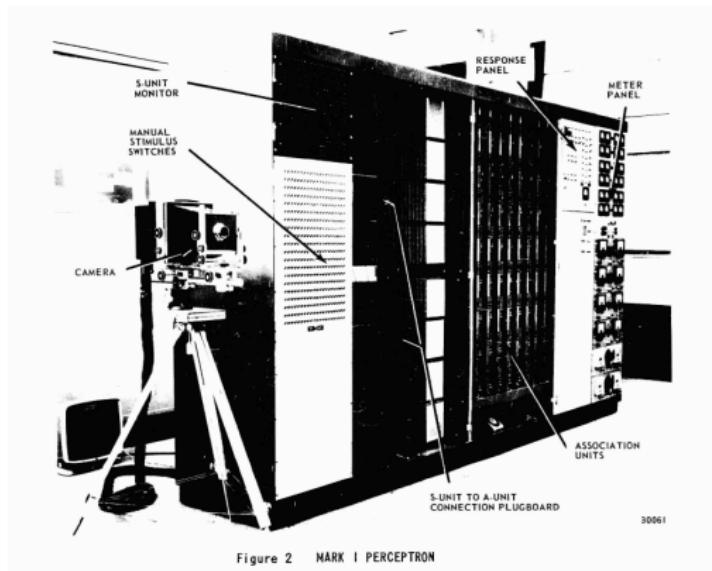


Figure 2 MARK I PERCEPTRON

Fuente: Mark I Perceptron Operators' Manual, Cornell Aeronautical Lab., Inc. 1960

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

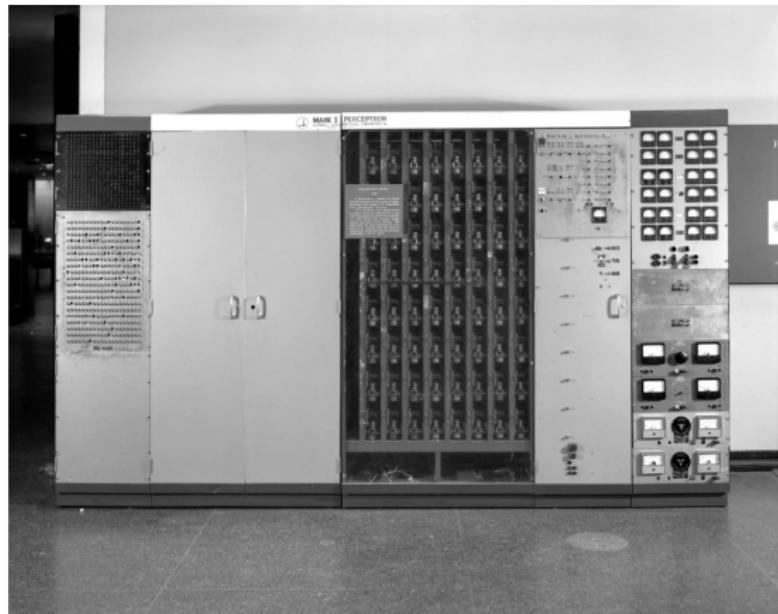
Introducción

MLP

Perceptron

Antecedentes.

Mark I Perceptron, Rosenblatt 1958.



Fuente: National Museum of American History & Smithsonian Institution.

https://americanhistory.si.edu/collections/search/object/nmah_334414

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

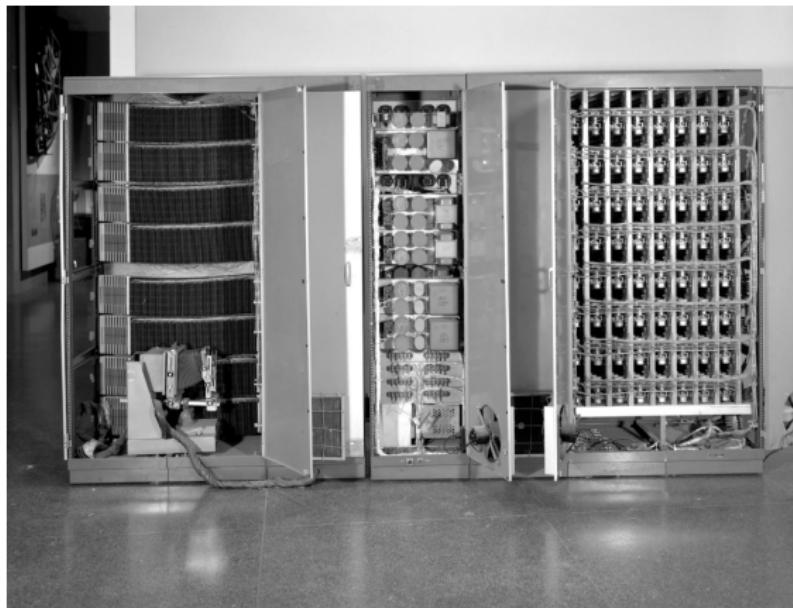
Introducción

MLP

Perceptron

Antecedentes.

Mark I Perceptron, Rosenblatt 1958.



Fuente: National Museum of American History & Smithsonian Institution.

https://americanhistory.si.edu/collections/search/object/nmah_334414

▶ YouTube Video

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

El algoritmo perceptrón

- Usa descenso por gradiente
- Define un tamaño de paso (learning rate) para el descenso
- La versión en línea actualiza los pesos β en dirección del gradiente, solo si encuentra un dato mal clasificado
- Alternativamente usa métodos Newton o Quasi-Newton para calcular el tamaño de paso

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

El algoritmo perceptrón

Perceptrón con método de Newton.

1: Input: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\boldsymbol{\beta}^0$, criterio de paro ϵ .

2: **repeat**

3: Calcular

$$\boldsymbol{\beta}^t = \boldsymbol{\beta}^{t-1} - H^{-1}(\boldsymbol{\beta}) \nabla L(\boldsymbol{\beta})$$

4: **until**

$$|H^{-1}(\boldsymbol{\beta}) \nabla L(\boldsymbol{\beta})| < \epsilon$$

5: Output: solución $\boldsymbol{\beta}$

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

El algoritmo perceptrón

Problemas ya mencionados:

- La solución no es única
- Solo para dos clases linealmente separables

El algoritmo perceptrón

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

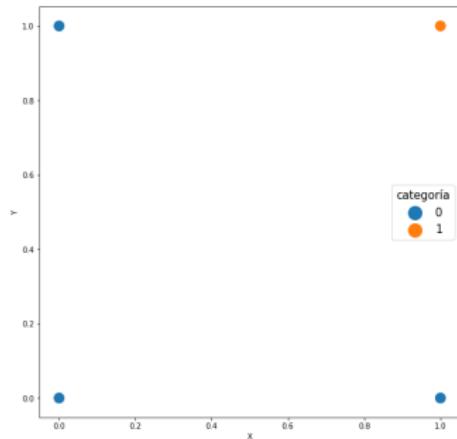
Introducción

MLP

Un ejemplo ilustrativo: compuertas lógicas.

Compuertas lógica AND.

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

El algoritmo perceptrón

Compuerta lógica AND.

Solución por mínimos cuadrados.

$$\min_{\beta} \sum_i (\hat{y}_i - y_i)^2,$$

donde

$$\hat{y}_i = \beta_0 + \beta' \mathbf{x}_i.$$

Como hacemos generalmente en ML, centramos los datos, lo que implica una re-etiquetación de la respuesta como $y \in \{-1, 1\}$.

En este caso,

$$\hat{y}_i = \text{signo}(\beta_0 + \beta' \mathbf{x}_i).$$

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

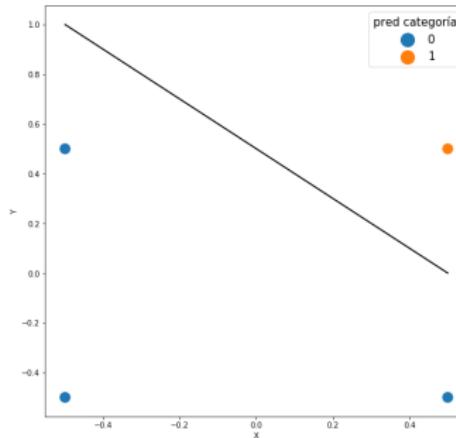
MLP

El algoritmo perceptrón

Compuerta lógica AND.

La solución tiene una forma cerrada usando las ecuaciones normales:

$$\beta_{\text{AND}} = \left(-\frac{1}{2}, 1, 1 \right)'$$



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

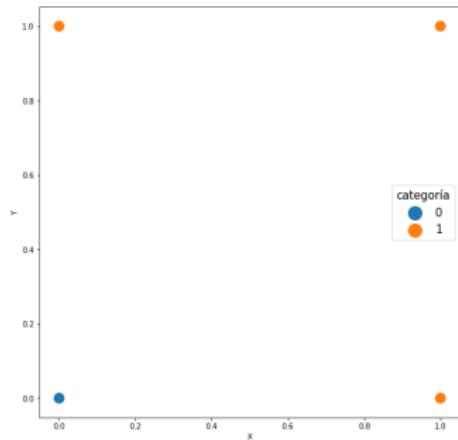
Introducción

MLP

El algoritmo perceptrón

Compuerta lógica OR.

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

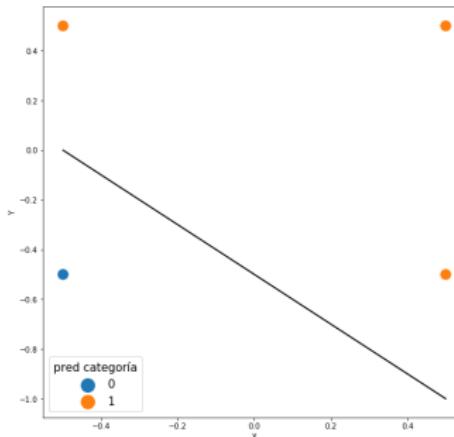
MLP

El algoritmo perceptrón

Compuerta lógica OR.

De la misma forma, centramos y resolvemos por mínimos cuadrados.

$$\beta_{\text{OR}} = \left(\frac{1}{2}, 1, 1 \right)'$$



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

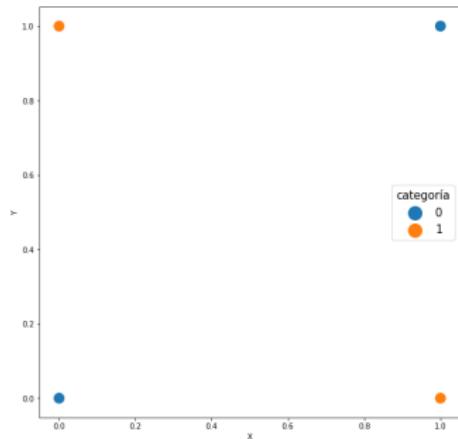
Introducción

MLP

El algoritmo perceptrón

Compuerta lógica XOR.

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

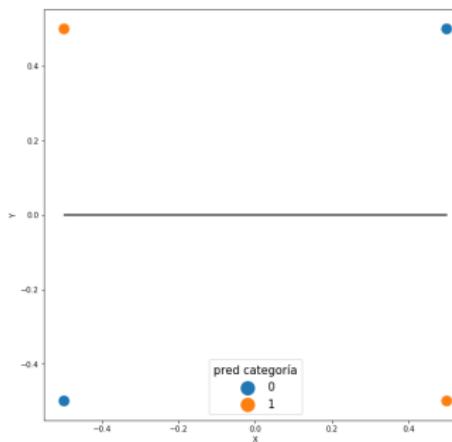
MLP

El algoritmo perceptrón

Compuerta lógica XOR.

De la misma forma, centramos y resolvemos por mínimos cuadrados.

$$\beta_{\text{XOR}} = (0, 0, 0)'$$



El no poder resolver éste problema hizo que se desestimara el uso de redes neuronales por algún tiempo durante los 1960's.

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

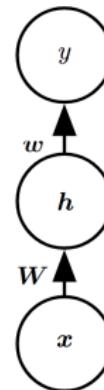
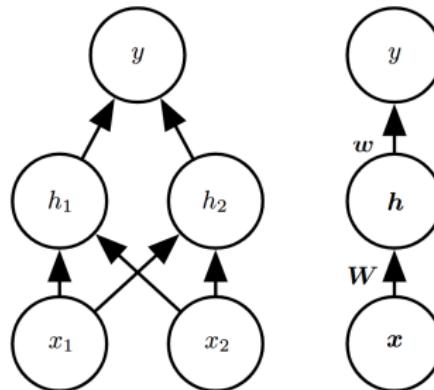
Introducción

MLP

El algoritmo perceptrón

Compuerta lógica XOR.

Una red con una capa oculta y dos unidades ocultas.



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

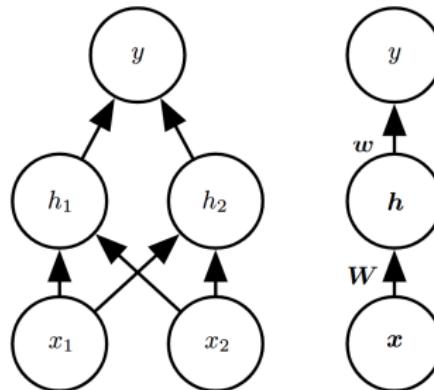
Introducción

MLP

El algoritmo perceptrón

Compuerta lógica XOR.

Una red con una capa oculta y dos unidades ocultas.



$$\begin{aligned}\mathbf{h} &= f^{(1)}(\mathbf{x}; \mathbf{W}, \mathbf{c}) \\ y &= f^{(2)}(\mathbf{h}; \mathbf{w}, b),\end{aligned}$$

por lo tanto

$$y = f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, b) = f^{(2)}(f^{(1)}(\mathbf{x})).$$

El algoritmo perceptrón

Compuerta lógica XOR.

Para obtener representaciones útiles de x en la tarea de clasificación, elegimos $\mathbf{h} = f^{(1)}(\cdot)$ no lineal (o semi-lineal):
 $\text{Relu}(u) = \max(0, u)$. Entonces

$$f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, b) = \mathbf{w}^\top \max\{0, \mathbf{W}^\top \mathbf{x} + \mathbf{c}\} + b.$$

$$\mathbf{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

$$\mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

$$\mathbf{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix},$$

con $b = 0$.

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

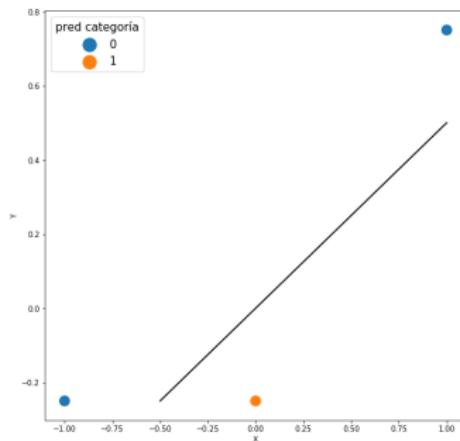
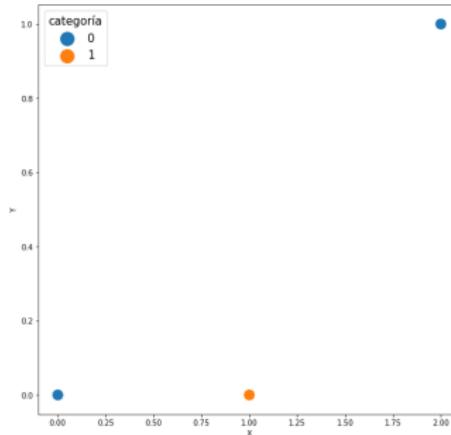
El algoritmo perceptrón

Compuerta lógica XOR.

Puedes comprobar que

$$\mathbf{h} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}.$$

Con $\beta_{\text{XOR}} = (0, 2, -4)'$.



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

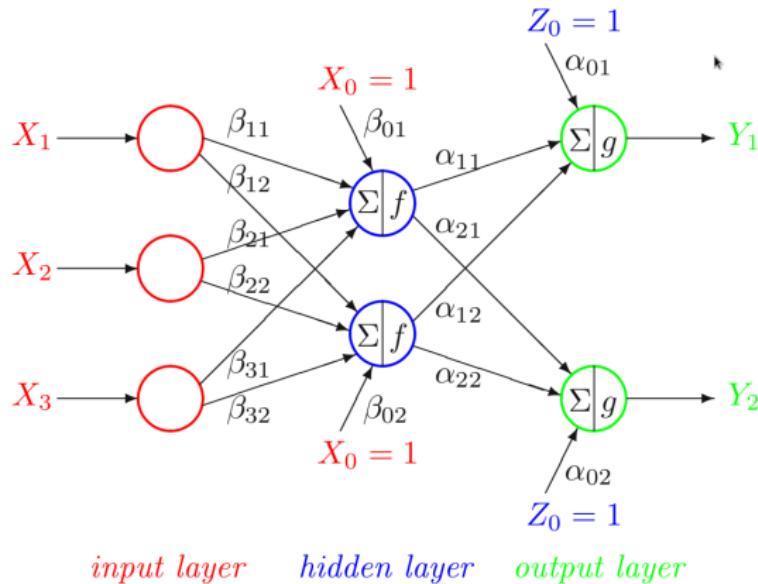
Introducción

MLP

MLP: Multilayer Perceptron (Feedforward Neural Networks)

Multilayer Perceptron

Esquema: Red neuronal (perceptron multicapa) con una capa oculta, $d = 3$ nodos de entrada, $t = 2$ nodos ocultos y $K = 2$ nodos de salida.



Modelos de
clasificación

Clasificación lineal

Regresión logística

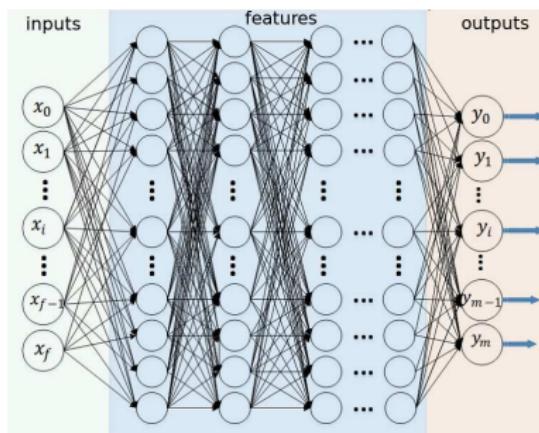
Redes neuronales

Introducción

MLP

Multilayer Perceptron

Capas ocultas. Nos permiten obtener representaciones útiles de los datos. Estas representaciones se “aprenden” mediante el ajuste del modelo (optimización).



Multilayer Perceptron

Veamos un ejemplo simple. MLP con una sola capa oculta.

Sea

$$\mathbf{x} = (x_1, x_2, \dots, x_d) \text{ nodos de entrada}$$

$$\mathbf{z} = (z_1, z_2, \dots, z_t) \text{ nodos ocultos (una capa)}$$

$$U_j = \beta_{0j} + \mathbf{x}' \boldsymbol{\beta}_j$$

$$V_k = \alpha_{0k} + \mathbf{z}' \boldsymbol{\alpha}_k,$$

donde

$$\boldsymbol{\beta}_j = (\beta_{1j}, \beta_{2j}, \dots, \beta_{dj})'$$

$$\boldsymbol{\alpha}_k = (\alpha_{1k}, \alpha_{2k}, \dots, \alpha_{tk})'$$

entonces

$$\begin{aligned} Z_j &= f_j(U_j), & j &= 1, 2, \dots, t \\ y_k &= g_k(V_k), & k &= 1, 2, \dots, K \end{aligned}$$

Multilayer Perceptron

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

y poniendo las ecuaciones juntas, el valor de la salida k de la red puede expresarse como

$$\hat{y}_k = \mu_k(\mathbf{x}) + \epsilon_k,$$

donde

$$\mu_k(\mathbf{x}) = g_k \left(\alpha_{0k} + \sum_{j=1}^t \alpha_{jk} f_j \left(\beta_{0j} + \sum_{i=1}^d \beta_{ij} x_i \right) \right)$$

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Multilayer Perceptron

y poniendo las ecuaciones juntas, el valor de la salida k de la red puede expresarse como

$$\hat{y}_k = \mu_k(\mathbf{x}) + \epsilon_k,$$

donde

$$\mu_k(\mathbf{x}) = g_k \left(\alpha_{0k} + \sum_{j=1}^t \alpha_{jk} f_j \left(\beta_{0j} + \sum_{i=1}^d \beta_{ij} x_i \right) \right)$$

Aquí, $f(\cdot)$, $g(\cdot)$, son **funciones de activación**:

Feedforward Neural Networks

Observaciones.

- Para regresión: $g_k(V_k) = V_k$ (identidad).
- Para clasificación: $g_k(V_k) = \frac{e^{V_k}}{\sum_{k=1}^K e^{V_k}}$ (soft-max o multilogit)

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Feedforward Neural Networks

Observaciones.

- Para regresión: $g_k(V_k) = V_k$ (identidad).
- Para clasificación: $g_k(V_k) = \frac{e^{V_k}}{\sum_{k=1}^K e^{V_k}}$ (soft-max o multilogit)
- Si f_j es la función identidad, obtenemos un **modelo lineal**, entonces, la red neuronal es un modelo lineal generalizado (modelo aditivo)

$$\mu_k(\mathbf{x}) = g_k \left(\sum_{j=1}^t \alpha_{jk} (\boldsymbol{\beta}'_j \mathbf{x}) \right)$$

Feedforward Neural Networks

Observaciones.

- Para regresión: $g_k(V_k) = V_k$ (identidad).
- Para clasificación: $g_k(V_k) = \frac{e^{V_k}}{\sum_{k=1}^K e^{V_k}}$ (soft-max o multilogit)
- Si f_j es la función identidad, obtenemos un **modelo lineal**, entonces, la red neuronal es un modelo lineal generalizado (modelo aditivo)

$$\mu_k(\mathbf{x}) = g_k \left(\sum_{j=1}^t \alpha_{jk} (\boldsymbol{\beta}'_j \mathbf{x}) \right)$$

- Si usamos soft-max, la red neuronal es un modelo de regresión logística Multilogit (una capa oculta).

Feedforward Neural Networks

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Veremos entonces algunos aspectos sobre:

- Representación y capas ocultas
- Capas de salida
- Funciones de costo
- Ajuste
- Optimización

Feedforward Neural Networks

Modelos de clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

- Capas y unidades ocultas.

Nos permiten obtener representaciones útiles de los datos. Estas representaciones se “aprenden” mediante un proceso de optimización.

Ejemplo: XOR

Feedforward Neural Networks

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

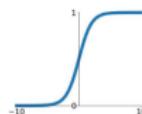
Introducción

MLP

- Capas y unidades ocultas.

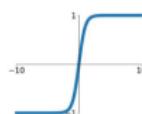
Sigmoid

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



tanh

$$\tanh(x)$$



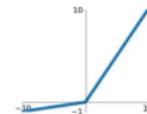
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

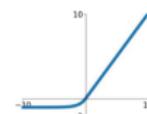


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



<https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044>

Feedforward Neural Networks

Victor Muñiz

Modelos de clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

- Unidades de salida.

Una **unidad**, ya sea oculta o de salida, realiza operaciones concretas (transformaciones) a un conjunto de variables o características de entrada para obtener una representación útil de ellas.

En el caso particular de las unidades de salida, éstas transformaciones tienen como objetivo, obtener una respuesta acorde a la tarea que pretendemos realizar con la red, por ejemplo, clasificación o regresión.

Feedforward Neural Networks

Modelos de clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

• Unidades de salida.

Tipo de salida	Distribución	Capa de salida	Función de costo
Binaria	Bernoulli	Sigmoid	Binary cross-entropy
Discreta	Multinomial	Softmax	Discrete cross-entropy
Contínua	Gausiana	Lineal	MSE (Gaussian cross-entropy)
Contínua	Mezcla de Gausianas	Mezcla de densidades	Cross-entropy

Feedforward Neural Networks

Victor Muñiz

Modelos de clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

• Funciones de costo

El ajuste de los parámetros (que pueden ser realmente muchos) se realiza resolviendo iterativamente (o recursivamente) un proceso de optimización. Para esto, debemos definir una función de costo apropiada según la tarea que queremos resolver.

Básicamente, hablamos de dos funciones:

- Error cuadrático medio:

$$L(\boldsymbol{\theta}) = \|\mathbf{y} - f(\mathbf{x}; \boldsymbol{\theta})\|$$

- Cross entropy:

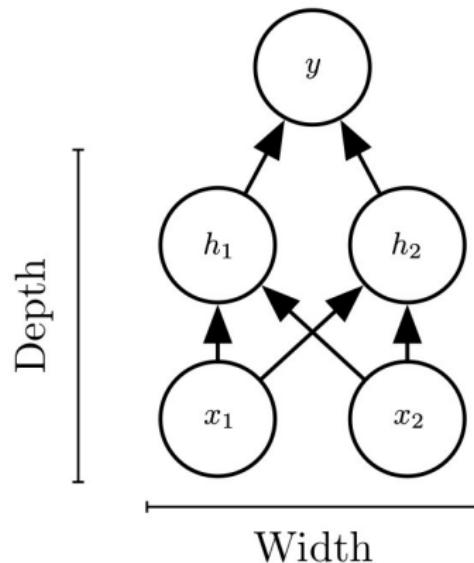
$$L(\boldsymbol{\theta}) = -E_{\mathbf{x}} \log P_{\text{model}}(\mathbf{y} | \mathbf{x})$$

Feedforward Neural Networks

Modelos de clasificación

- Clasificación lineal
- Regresión logística
- Redes neuronales
- Introducción
- MLP

● Arquitectura



Feedforward Neural Networks

Teorema de aproximación universal

Cualquier función continua definida en un subconjunto compacto de \mathbb{R}^d puede ser aproximada uniformemente (en una métrica apropiada), con una red neuronal, usando la función lineal en la capa de salida y al menos una capa oculta con la función sigmoide.

Feedforward Neural Networks

Teorema de aproximación universal

Cualquier función continua definida en un subconjunto compacto de \mathbb{R}^d puede ser aproximada uniformemente (en una métrica apropiada), con una red neuronal, usando la función lineal en la capa de salida y al menos una capa oculta con la función sigmoide.

Entonces, según el Teorema anterior, sin importar la función que tratamos de aprender, una red MLP suficientemente grande (muchas unidades ocultas) será capaz de representarla.

Feedforward Neural Networks

Teorema de aproximación universal

Cualquier función continua definida en un subconjunto compacto de \mathbb{R}^d puede ser aproximada uniformemente (en una métrica apropiada), con una red neuronal, usando la función lineal en la capa de salida y al menos una capa oculta con la función sigmoide.

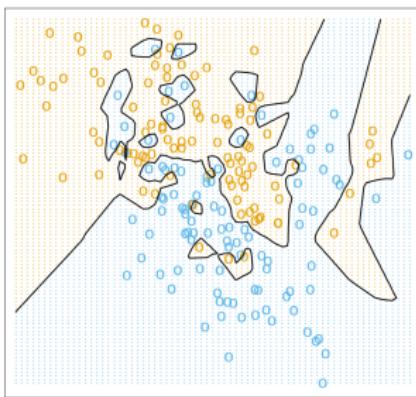
Entonces, según el Teorema anterior, sin importar la función que tratamos de aprender, una red MLP suficientemente grande (muchas unidades ocultas) será capaz de representarla.

¿Porqué usar más capas entonces?

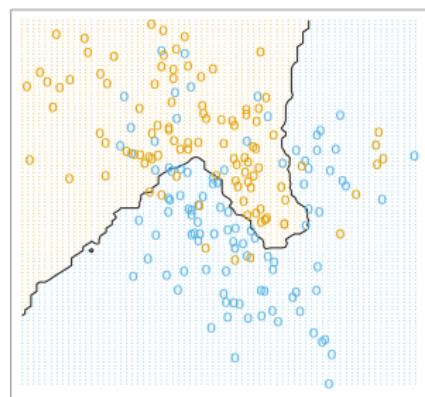
¿Qué implicaciones tiene?

Feedforward Neural Networks

Según el Teorema anterior, sin importar la función que tratamos de aprender, una red MLP suficientemente grande (muchas unidades ocultas) será capaz de representarla.



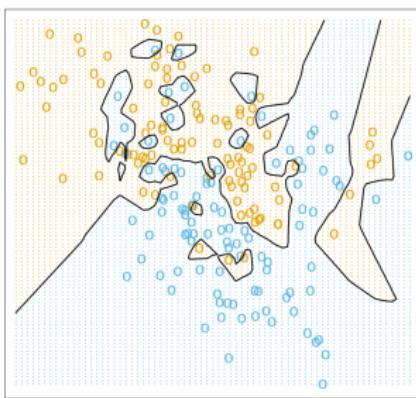
modelo 1



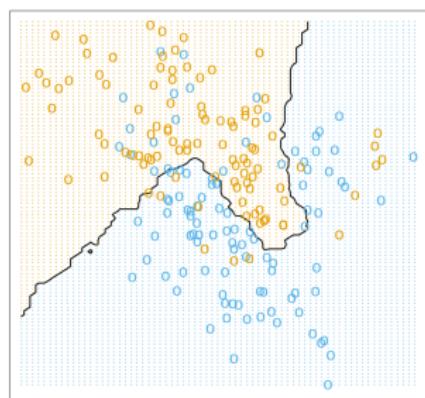
modelo 2

Feedforward Neural Networks

Según el Teorema anterior, sin importar la función que tratamos de aprender, una red MLP suficientemente grande (muchas unidades ocultas) será capaz de representarla.



modelo 1



modelo 2

Mensaje: es necesario regularizar.

Una forma de evitar el sobreajuste es añadir un término de regularización en los parámetros (weight decay).

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Feedforward Neural Networks

On the Number of Linear Regions of Deep Neural Networks

Guido Montúfar

Max Planck Institute for Mathematics in the Sciences

montufar@mis.mpg.de

Razvan Pascanu

Université de Montréal

pascanur@iro.umontreal.ca

Kyunghyun Cho

Université de Montréal

kyunghyun.cho@umontreal.ca

Yoshua Bengio

Université de Montréal, CIFAR Fellow

yoshua.bengio@umontreal.ca

Abstract

We study the complexity of functions computable by deep feedforward neural networks with piecewise linear activations in terms of the symmetries and the number of linear regions that they have. Deep networks are able to sequentially map portions of each layer's input-space to the same output. In this way, deep models compute functions that react equally to complicated patterns of different inputs. The compositional structure of these functions enables them to re-use pieces of computation exponentially often in terms of the network's depth. This paper investigates the complexity of such compositional maps and contributes new theoretical results regarding the advantage of depth for neural networks with piecewise linear activation functions. In particular, our analysis is not specific to a single family of models, and as an example, we employ it for rectifier and maxout networks. We improve complexity bounds from pre-existing work and investigate the behavior of units in higher layers.

(Montufar et al. NIPS, 2014).

Feedforward Neural Networks

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Entonces:

Feedforward Neural Networks

Modelos de clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Entonces:

- Una red con una capa oculta es suficiente para **representar** (no necesariamente aprender) cualquier función.

Feedforward Neural Networks

Modelos de clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Entonces:

- Una red con una capa oculta es suficiente para **representar** (no necesariamente aprender) cualquier función.
- Sin embargo, tal red puede ser muy larga (con muchas unidades) y generalmente, tiene problemas de generalización

Feedforward Neural Networks

Modelos de clasificación

Clasificación lineal
Regresión logística
Redes neuronales
Introducción
MLP

Entonces:

- Una red con una capa oculta es suficiente para **representar** (no necesariamente aprender) cualquier función.
- Sin embargo, tal red puede ser muy larga (con muchas unidades) y generalmente, tiene problemas de generalización
- Usar redes profundas puede reducir el número de unidades y puede reducir el error de generalización.

Feedforward Neural Networks

Modelos de clasificación

Clasificación lineal
Regresión logística
Redes neuronales
Introducción
MLP

Entonces:

- Una red con una capa oculta es suficiente para **representar** (no necesariamente aprender) cualquier función.
- Sin embargo, tal red puede ser muy larga (con muchas unidades) y generalmente, tiene problemas de generalización
- Usar redes profundas puede reducir el número de unidades y puede reducir el error de generalización.
- Por supuesto, hay otras consideraciones respecto a la arquitectura de las redes profundas. Nosotros nos enfocaremos solo en algunas.

Regularización

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Regularización:

“Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error”. Goodfellow et. al., 2016.

Regularización

Regularización:

“Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error”. Goodfellow et. al., 2016.

Restringir el modelo para reducir el error de generalización (no precisamente el de entrenamiento).

Regularización $\|\cdot\|_L$.

$$L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) + \alpha \Omega(\boldsymbol{\theta})$$

donde $\Omega(\boldsymbol{\theta})$ es una norma definida sobre los parámetros del modelo y $\alpha \in [0, \infty)$ es un hiperparámetro que pesa la contribución relativa de la penalización sobre la norma.

Regularización

- Regularización. $\|\cdot\|_{L2}$: Weight decay.
 - Reduce el vector de pesos en dirección del gradiente

Regularización

- Regularización. $\|\cdot\|_{L2}$: Weight decay.
 - Reduce el vector de pesos en dirección del gradiente
 - Re-escala el “peso óptimo” con un criterio dado por una aproximación al óptimo w^*

Modelos de
clasificación

Clasificación lineal

Regresión logística

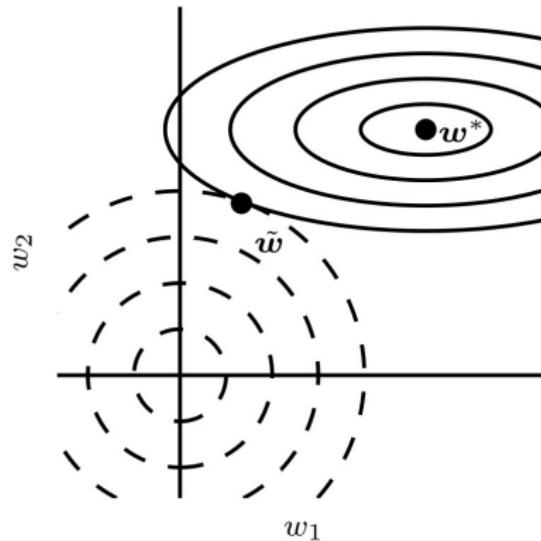
Redes neuronales

Introducción

MLP

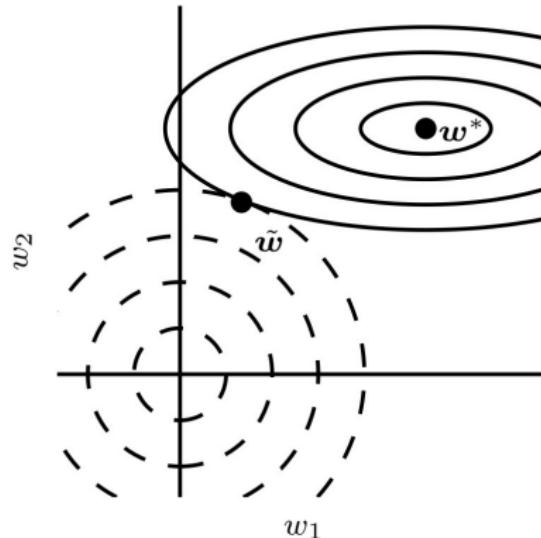
Regularización

- Regularización. $\|\cdot\|_{L2}$: Weight decay.
 - Reduce el vector de pesos en dirección del gradiente
 - Re-escala el “peso óptimo” con un criterio dado por una aproximación al óptimo w^*



Regularización

- Regularización. $\|\cdot\|_{L2}$: Weight decay.
 - Reduce el vector de pesos en dirección del gradiente
 - Re-escala el “peso óptimo” con un criterio dado por una aproximación al óptimo w^*



- Regularización. $\|\cdot\|_{L1}$: Weight decay.

Regularización

Modelos de
clasificación

Clasificación lineal

Regresión logística

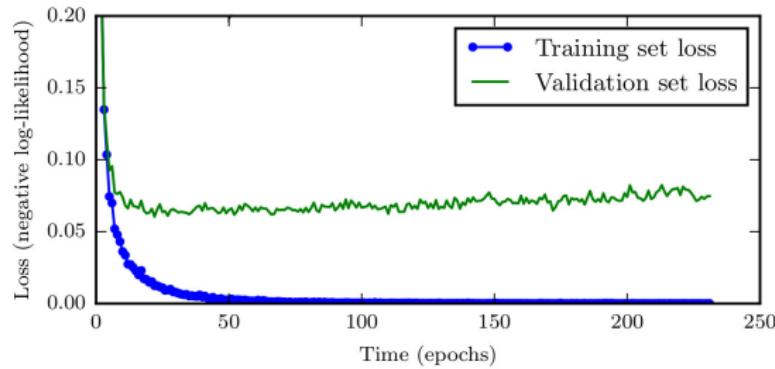
Redes neuronales

Introducción

MLP

Otros tipos de regularización para redes profundas:

- Early stopping



- Dropout
- Batch normalization
- Data augmentation
- Adversarial training

Feedforward Neural Networks

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Ajuste.

Nuestro objetivo es

$$\min_{\theta} L(f(\mathbf{x}; \theta) \mathbf{y}).$$

Feedforward Neural Networks

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Ajuste.

Nuestro objetivo es

$$\min_{\theta} L(f(\mathbf{x}; \theta) \mathbf{y}).$$

Sin embargo, para obtener una buena generalización, y por motivos computacionales, en la práctica resolvemos:

$$\min_{\theta} E(L(f(\mathbf{x}; \theta) \mathbf{y})) ,$$

donde $\mathbf{x} \sim F(\mathbf{x}) \equiv p_{data}(\mathbf{x})$ y es *iid*.

Deep feedforward networks

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Generalmente, minimizamos el **riesgo empírico**, es decir, basado en un conjunto de entrenamiento:

$$\min_{\theta} E(L(f(\mathbf{x}; \theta) \mathbf{y})) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \theta) \mathbf{y}_i),$$

con $\mathbf{x} \sim \hat{p}_{data}(\mathbf{x})$, es decir, basado en la distribución empírica de \mathbf{x} .

Muchos algoritmos se han propuesto para resolver el problema anterior.

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

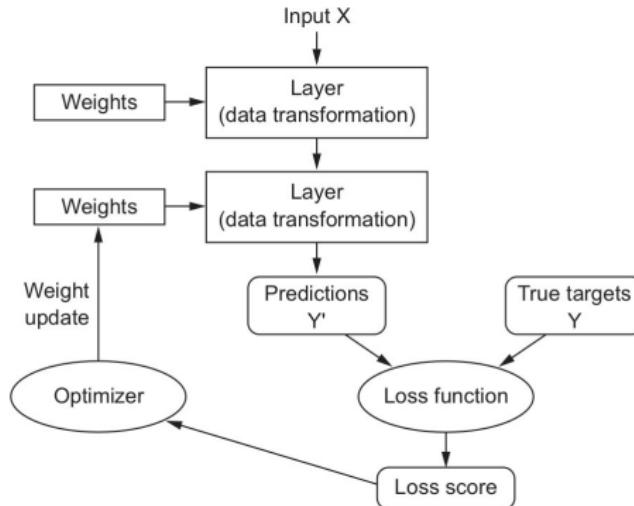
Introducción

MLP

Feedforward Neural Networks

En general, se utiliza un algoritmo de descenso por gradiente estocástico que minimiza alguna de las funciones de costo que ya vimos.

Este algoritmo es llamado Backpropagation, y puede incluir parámetros de regularización o sobreajuste (weight-decay).



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Deep feedforward networks

Algunos conceptos importantes.

- Batch y mini batch.

Son una muestra, generalmente pequeña (entre 8 y 128) de los datos. Representan una partición de los datos de entrenamiento o prueba, y se almacenan en un tensor de dimensión arbitraria, pero el primer eje (axis) identifica el batch (tensorflow).

- Época.

Es un término para indicar que todas las muestras del conjunto de entrenamiento han sido usadas para actualizar los parámetros del modelo, es decir, han completado el paso forward y backward. En consecuencia, todos los batches han sido usados para el procedimiento de optimización.

- Iteración.

El número de batches necesario para completar una época.

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Feedforward Neural Networks

Resumiendo, el ajuste es:

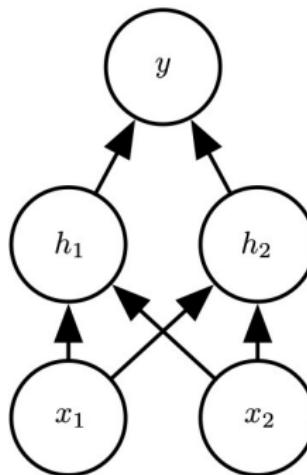
Compute activations

Forward prop

Compute loss

Back-prop

Compute derivatives



Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Feedforward Neural Networks

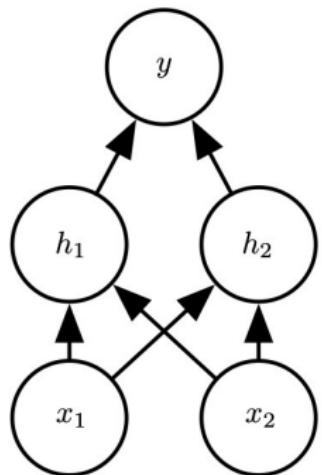
Resumiendo, el ajuste es:

Compute derivatives

Forward pass

Compute loss

Back-prop



- Valores iniciales: generalmente, se inicializa con valores cercanos a 0 aleatoriamente

Deep feedforward networks

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

● Back-propagation (Backprop)

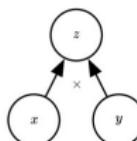
- Básicamente, es un algoritmo recursivo para calcular la regla de la cadena.
- Hay muchos algoritmos para cálculos recursivos, pero la regla de la cadena, tiene una característica, y es que varias expresiones pueden repetirse...
- Las implementaciones más eficientes, se basan en la construcción de grafos computacionales.

Deep feedforward networks

- Back-propagation (Backprop)

Computational graphs.

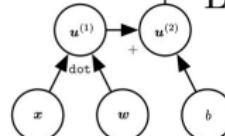
Multiplication



(a)

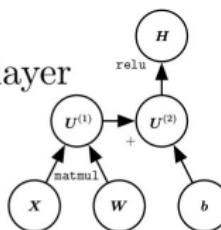
$$\hat{y}$$

σ



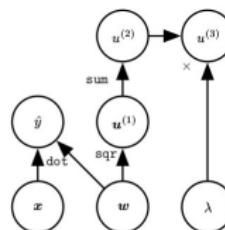
(b)

ReLU layer



(c)

Logistic regression



(d)

Linear regression
and weight decay

(Goodfellow et al, 2017)

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

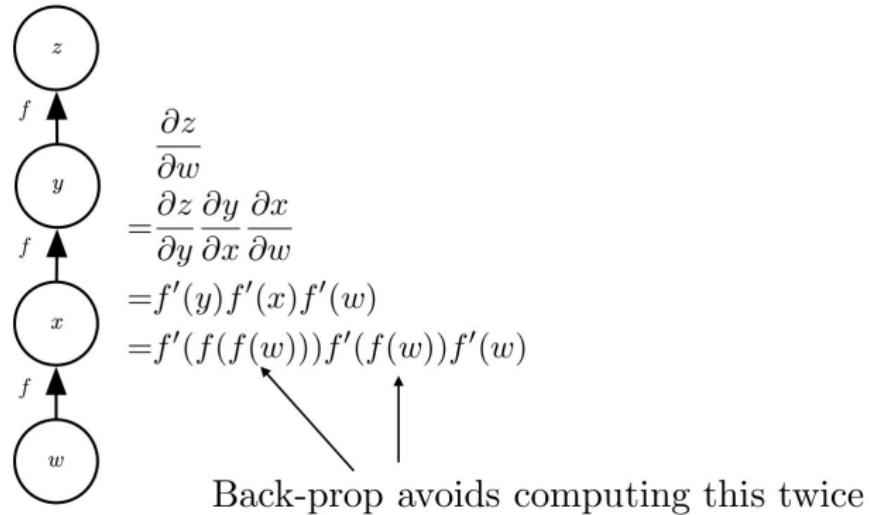
Introducción

MLP

Deep feedforward networks

- Back-propagation (Backprop)

Aplicación recursiva de la regla de la cadena. Sea $x = f(w), y = f(x), z = f(y)$:



(Goodfellow et al, 2017)

Modelos de
clasificación

Clasificación lineal

Regresión logística

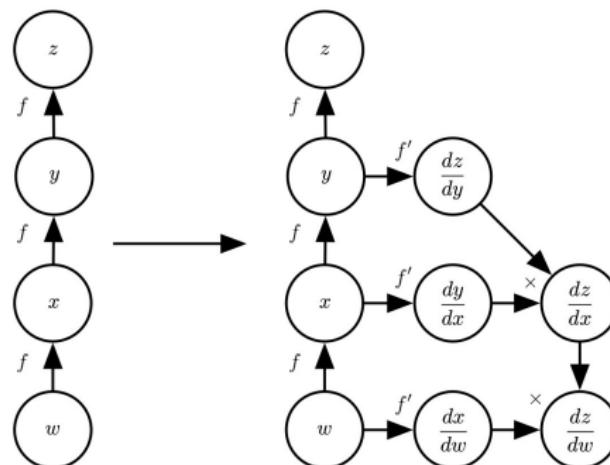
Redes neuronales

Introducción

MLP

Deep feedforward networks

Para esto, se utiliza diferenciación automática, relacionada a la llamada **symbol-to-symbol**. Para esto, se crea un grafo que describe las operaciones para calcular las derivadas.



Deep feedforward networks

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Así, para cada cálculo de derivadas en el grafo G , se crea una gráfica computacional “extra” que **indica** cómo calcular cada una de ellas.

Posteriormente, los nodos del grafo computacional se llenan con valores numéricos (symbol-to-number).

Pytorch utiliza también diferenciación automática en tiempo de ejecución (runtime), construyendo el grafo computacional a medida que se va ejecutando.

Feedforward Neural Networks

Victor Muñiz

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

<http://playground.tensorflow.org>

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Deep feedforward networks

Esquemas para datos con etiquetas desbalanceadas:

- Weighted cross-entropy loss function. Para un problema de clasificación con K categorías, y un par de entrenamiento (\mathbf{x}_i, y_i) , la definimos como:

$$L(\mathbf{x}_i, y_i) = -w_k \log \left(\frac{\exp(z_{i,y_i})}{\sum_{k=1}^K \exp(z_{i,k})} \right) y_i,$$

donde el score $z_{i,y_i} = \mathbf{x}'_i \boldsymbol{\beta}_{y_i}$, para algún $\boldsymbol{\beta}$, y pesos w_k , que es el peso para cada clase. Opciones para los pesos:

- $w_k = \frac{n}{K * n_k}$ (implementada en `sklearn.utils.class_weight`)
- $w_k = 1 - \frac{n_k}{n}$

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Deep feedforward networks

Esquemas para datos con etiquetas desbalanceadas:

- Weighted cross-entropy loss function. Para un problema de clasificación con K categorías, y un par de entrenamiento (\mathbf{x}_i, y_i) , la definimos como:

$$L(\mathbf{x}_i, y_i) = -w_k \log \left(\frac{\exp(z_{i,y_i})}{\sum_{k=1}^K \exp(z_{i,k})} \right) y_i,$$

donde el score $z_{i,y_i} = \mathbf{x}'_i \boldsymbol{\beta}_{y_i}$, para algún $\boldsymbol{\beta}$, y pesos w_k , que es el peso para cada clase. Opciones para los pesos:

- $w_k = \frac{n}{K * n_k}$ (implementada en `sklearn.utils.class_weight`)
- $w_k = 1 - \frac{\hat{n}_k}{n}$
- Aumento de datos (cuando es posible)

Modelos de
clasificación

Clasificación lineal

Regresión logística

Redes neuronales

Introducción

MLP

Deep feedforward networks

Esquemas para datos con etiquetas desbalanceadas:

- Weighted cross-entropy loss function. Para un problema de clasificación con K categorías, y un par de entrenamiento (\mathbf{x}_i, y_i) , la definimos como:

$$L(\mathbf{x}_i, y_i) = -w_k \log \left(\frac{\exp(z_{i,y_i})}{\sum_{k=1}^K \exp(z_{i,k})} \right) y_i,$$

donde el score $z_{i,y_i} = \mathbf{x}'_i \boldsymbol{\beta}_{y_i}$, para algún $\boldsymbol{\beta}$, y pesos w_k , que es el peso para cada clase. Opciones para los pesos:

- $w_k = \frac{n}{K * n_k}$ (implementada en `sklearn.utils.class_weight`)
- $w_k = 1 - \frac{\hat{n}_k}{n}$

- Aumento de datos (cuando es posible)
- Curriculum learning