Modelos de clasificación: Bayesiano ingenuo, máquina de soportes vectorial y regresión logística

Fernando Carranza
fernandocarranza86@gmail.com

Primer Cuatrimestre de 2025

- Unidad 1: Introducción
- **Unidad 2: Los algoritmos supervisados**
  - i) Aprendizaje supervisado y no supervisado.
  - ii) Métricas usuales para medir el rendimiento de modelos de clasificación (accuracy, precisión y cobertura).
  - iii) Anotación como tarea a resolver por un modelo predictivo.
  - iv) Datos estructurados y no estructurados: manejo de estructuras de almacenamiento de datos (json, csv).
  - v) Vectorización (conversión de un texto en tanto dato no estructurado en un arreglo numérico estructurado; CountVectorizer, TfidfVectorizer).
  - **vi) Modelos de clasificación: Bayesiano ingenuo, Regresión Logística, Máquina de soporte vectorial (Support Vector Machines).**
- Unidad 3: Anotación morfológica y de clase de palabra
- Unidad 4: Anotación sintáctica
- Unidad 5: Anotación para propósitos específicos

Presentación

Estructura y temas de esta clase:

1. Introducción
2. La clasificación automática
3. Bayesiano Ingenuo
   - Nociones de probabilidad
   - Ejemplo no lingüístico de bayesiano ingenuo
   - Bayesiano ingenuo en una tarea de PLN
4. Support Vector Machine
5. Regresión logística
6. Recapitulación
7. Bibliografía

Bibliografía para la preparación de esta clase:

- Jurafsky y Martin. 2025. "Naive Bayes, Text Classification, and Sentiment". *Speech and Language processing*. Draft.
- Tan, Michael Steinbach y Vipin Kumar. 2005. "Classification: Alternative Techniques". *Data Mining*.Pearson.
- Jurafsky y Martin. 2025. "Logistic Regression". *Speech and Language processing*. Draft.

- Los algoritmos de clasificación son algoritmos predictivos que toman una serie de datos y los asocian a una serie de categorías.
- Los algoritmos de clasificación pueden estar basados en reglas o recurrir a aprendizaje automático.
- De entre los que recurren a aprendizaje automático, existen los supervisados y los no supervisados. Los supervisados son aquellos que se entrenan a partir de datos anotados previamente que el modelo usa para descubrir patrones estadísticos. Los algoritmos no supervisados de clasificación o *clustering* elaboran la clasificación solo en función de la similitud o no entre los datos.

En lo que sigue nos vamos a concentrar en los supervisados.

De entre los tipos de clasificación, el que más concierne al procesamiento del lenguaje natural es la **clasificación de textos** (*text categorization*). Algunos ejemplos de clasificación de textos:

- **Análisis de sentimiento**: Extracción de la orientación positiva o negativa de un texto con respecto a un objeto o tema, clasificación que puede ser binaria o más compleja.
- **Detección de Spam**: Identificación de si un texto es o no spam, una clasificación binaria.
- **Language ID**: Identificación de la lengua de un texto
- **Atribución de autoría**: Identificación del autor de un texto.
- **Topic labeling**: Identificación del tema de un texto.
- **Intent classification**: Clasificación de intenciones.

También vamos a poder usar la clasificación para determinar algunas de las siguientes cuestiones:

- Cuál es la clase de palabra para un determinado token.
- Cuál es la función sintáctica de un determinado token con respecto a otro.
- A qué clase semántica responde determinado *chunk*.

Clasificación y modelos de lenguaje

Los modelos de lenguaje también pueden pensarse como algoritmos de clasificación. Cada palabra o sucesiones de palabras se pueden pensar como un documento y la siguiente como la clase. De este modo, dada una determinada palabra (documento) se predice la siguiente (la clase) y luego se toma la siguiente como el documento para la próxima clase y así sucesivamente.

- Existen dos grandes tipos de algoritmos de clasificación supervisados: los algoritmos generativos, como el bayesiano ingenuo, y los discriminativos, como la regresión logística.

- Los algoritmos de clasificación generativos construyen modelos de respecto de cómo una clase es capaz de generar o no los datos de input.
- Los algoritmos discriminativos aprenden qué rasgos del input son los más útiles para discriminar entre las diferentes clases.

- Para entrenar un algoritmo de clasificación, es necesario contar con datos estructurados. Por definición, un dato estructurado es uno con una sintaxis claramente definida. Es muy común que los datos estructurados aparezcan en documentos como json o csv, etc.

Ejemplo de json


```
{
  "tarea":"clasificacion de sentimiento",
  "datos":[
    {
      "texto":"Me parece una mierda",
      "sentimiento":0
    },{
      "texto":"me encanta",
      "sentimiento":0
    },
    {
      "texto":"sirve para lo que tiene que servir",
      "sentimiento":0.5
    },
    {
      "texto":"hace muy bien su tarea",
      "sentimiento":1
    }
  ]
}
```

Ejemplo de csv

```
"me parece una mierda",0
"me encanta",1
"sirve para lo que tiene que servir",0.5
"no sirve para nada",0
"hace muy bien su tarea",1
```

Visualización en forma de cuadro usando “,” como delimitador:

"me parece una mierda"	0
"me encanta"	1
"sirve para lo que tiene que servir"	0.5
"no sirve para nada"	0
"hace muy bien su tarea"	1

- Para entrenar un algoritmo supervisado, normalmente se divide el conjunto de datos (*data set*) en partes y se entrena con una (el *train set*) y se mide la capacidad predictiva con otra (el *test set*). Si no fuera así, no tendríamos cómo evaluar la efectividad del modelo, ya que evaluar un modelo con un dato que ya conoce sería hacer trampa.
- También se suele recurrir a un *validation set*, *development set* o *devset* para ajustar los hiperparámetros del modelo.

La validación cruzada (*cross validation*) consiste en dividir el *data set* en partes y entrenar sucesivamente modelos con todas las partes menos una hasta agotar todas las combinaciones posibles y validar cada uno de esos modelos con los datos que quedaron afuera.

- Existen distintas técnicas para entrenar los modelos estadísticos. Una librería en Python muy útil para indagar es sklearn (`https://scikit-learn.org/stable/index.html`)
- Si el algoritmo distribuye el conocimiento que adquiere en la etapa de entrenamiento en capas ocultas (*hidden layers*), se habla de aprendizaje profundo (*deep learning*).
- El aprendizaje profundo puede ser supervisado o no supervisado.

Algunas técnicas estadísticas típicas de clasificación incluyen al bayesiano ingenuo (*Naive Bayesian*), las *support vector machines* (SVM) y la regresión logística (*logistic regression*), entre otras.

Bayesiano ingenuo

El bayesiano ingenuo es un clasificador estadístico que utiliza el teorema de Bayes como base.
El teorema bayesiano tiene la siguiente forma:

$$(1) \quad P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}$$

- **Resultado**: Un posible valor que puede arrojar el fenómeno en cuestión.
- **Espacio de muestra**: El conjunto total de todos los resultados que puede arrojar el fenómeno. En las fórmulas, se suele referir mediante la variable Ω
- **Evento**: Un subconjunto de los resultados que puede arrojar el fenómeno.

Algunos axiomas:

- **No negatividad**: La probabilidad de un elemento no puede ser un número negativo.
- **Normalización**: La sumatoria de la probabilidad de todos los resultados que conforman el espacio de muestra tiene que ser igual a 1.

Probabilidad

(2) P(Y) es la posibilidad de encontrarnos con el resultado Y.

Supongamos que queremos determinar cuál es la probabilidad de que en esta clase un estudiante tenga una computadora. ¿Cómo hacemos?

La probabilidad se puede obtener mediante estadística contando. La probabilidad de que se dé un resultado o un evento y se calcula contando la cantidad de veces que se da el resultado y dividida la cantidad de instancias totales de posibles resultados dentro del espacio de muestra Ω.

(3) $P(Y|X)$ es la probabilidad de encontrarnos con el resultado Y dado el resultado X.

Supongamos que queremos determinar cuál es es la probabilidad de que alguien que trajo computadora haya traído mate. ¿Cómo hacemos?

La probabilidad condicionada se puede obtener mediante estadística contando. La probabilidad de que se dé un resultado o un evento y dado un resultado o evento x se calcula contando la cantidad de veces que se da el resultado o evento y dividida la cantidad de instancias del resultado o evento x.

Un bayesiano ingenuo es un algoritmo generativo que caracteriza clases en función de los datos, vectorizados en función de una serie de rasgos que los caracterizan.

id	A	B	C
Tid	home owner	Marital status	Defaulted Borrower
1	Yes	Single	No
2	No	Married	No
3	No	Single	No
4	Yes	Married	No
5	No	Divorced	Yes
6	No	Married	No
7	Yes	Divorced	No
8	No	Single	Yes
9	No	Married	No
10	No	Single	Yes

Cuadro: Tomado de Tan *et al.* (2006)

- Vamos a usar C como variable para el conjunto de las clases, c como variable de clase, X como variable para el dato vectorizado y x para cada valor en el vector.
- La tarea de entrenamiento consiste en calcular la probabilidad P(Yes|X) y P(No|X) dado el corpus de entrenamiento.
- Es ingenuo porque asume que la presencia de cada atributo en una clase dada es independiente del resto de los atributos. Esto permite que la probabilidad no se compute directamente con B (con todo el vector, lo cual requeriría muchos datos), sino con cada uno de los valores de B. Esa "ingenuidad" es la que permite multiplicar las probabilidades posteriores entre sí por cada valor de X.
- $\hat{c}$ es la clase predicha por el modelo.

El teorema bayesiano aplicado al problema de calcular la probabilidad condicionada de una clase dado un vector es como sigue:

$$(4) \quad \mathrm{P(c|X)} = \frac{\mathrm{P(X|c)} \times \mathrm{P(c)}}{\mathrm{P(X)}}$$

Dada la ingenuidad del modelo, multiplicamos todas las dimensiones de X:

$$(5) \quad \mathrm{P(c|X)} = \frac{P(c) \prod_{x \in X} P(x|c)}{\mathrm{P(X)}}$$

Dado que la P(X) se mantiene fija para todas las clases, se la elimina de la cuenta

$$(6) \quad P(c|X) = P(c) \prod_{x \in X} P(x|c)$$

Dado que lo que queremos es predecir la clase dado un vector, necesitamos repetir esta operación para todas las clases y seleccionar aquella que maximice la posibilidad (la que tenga una mayor probabilidad). Esa será la clase a predecir, o $\hat{c}$:

(7)

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{x \in X} P(x|c)$$

id	A	B	C
Tid	home owner	Marital status	Defaulted Borrower
1	Yes	Single	No
2	No	Married	No
3	No	Single	No
4	Yes	Married	No
5	No	Divorced	Yes
6	No	Married	No
7	Yes	Divorced	No
8	No	Single	Yes
9	No	Married	No
10	No	Single	Yes

$P(A{=}Yes|C{=}No) = 3/7$
$P(A{=}No|NC{=}o) = 4/7$
$P(A{=}YesIC{=}Yes) = 0$
$P(A{=}No|C{=}Yes) = 1$
$P(B{=}Single|C{=}No) = 2/7$
$P(B{=}Divorced|C{=}No) = 1/7$
$P(B{=}Married|C{=}No){=} 4/7$
$P(B{=}Single|C{=}Yes){=} 2/3$
$P(B{=}Divorced|C{=}Yes){=} 1/3$
$P(B{=}Married|C{=}Yes){=} 0$
$P(C{=}Yes) = 0.3$
$P(C{=}No) = 0.7$

Supongamos que queremos predecir si va a ser Yes o No el siguiente deudor:

id	A	B	C
Tid	home owner	Marital status	Defaulted Borrower
11	No	Married	?

Para eso tenemos que completar la fórmula con los datos que sacamos de la tabla:

(8) a. $P(X|C{=}No) = P(A{=}No|C{=}No) \times P(B{=}Married|C{=}No) = \frac{4}{7} \times \frac{4}{7} = \frac{16}{49} = 0.3265$

b. $P(X|C{=}Yes) = P(A{=}No|C{=}Yes) \times P(B{=}Married|C{=}Yes) = 1 \times 0 = 0$

(9) a.

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{x \in X} P(x|c)$$

(10) a. $\text{P(C=Yes|X=[No, Married])} = P(C = Yes) \prod_{x \in X} P(x|C = Yes)$

b. $\text{P(C=Yes|X=[No, Married])} = 0{,}3 \times 1 \times 0$

c. $\text{P(C=Yes|X=[No, Married])} = 0$

(11) a. $\text{P(C=No|X=[No, Married])} = P(C = No) \prod_{x \in X} P(x|C = No)$

b. $\text{P(C=No|X=[No, Married])} = 0{,}7 \times \frac{4}{7} \times \frac{4}{7}$

c. $\text{P(C=No|X=[No, Married])} = 0{,}228571429$

Dado que $\text{P(No|X)} > \text{P(Yes|X)}$, se predice que 11 no va a ser un *defaulted borrower*.

Para aplicar esto a una tarea de procesamiento del lenguaje natural, tenemos que vectorizar los datos lingüísticos con los que queramos trabajar. Por ejemplo, para una tarea de análisis de sentimiento, una forma es asumir que cada feature es la frecuencia de las palabras en un modelo de bolsa de palabras (BOW).

d1= I love this movie! It's sweet, but with satirical humor. The dialogue is great...

$$d1 = \begin{matrix} \text{it} & 6 \\ \text{I} & 5 \\ \text{the} & 4 \\ \text{to} & 3 \\ \text{and} & 3 \\ \text{seen} & 2 \\ \text{yet} & 1 \\ \ldots & \ldots \end{matrix}$$

Los documentos se pueden representar entonces como una matriz de documento-término:

doc/words	it	I	the	to	and	yet
d1	2	1	2	1	1	1
d2	1	1	0	0	2	0
d3	3	4	1	0	0	1

A esta matriz se le puede aplicar, por ejemplo, tf-idf

TF-IDF

tf-idf parte de las siguientes ideas:

- Parte de la idea de *Bag of words* (BOW)
- Ajusta la importancia de palabras comunes en el corpus
- Reduce el peso de las palabras muy frecuentes (como *stopwords*), a la vez que aumenta la importancia de las menos frecuentes (distintivas para un documento).

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \cdot \log\left(\frac{N}{\text{df}(t)}\right)$$

donde:

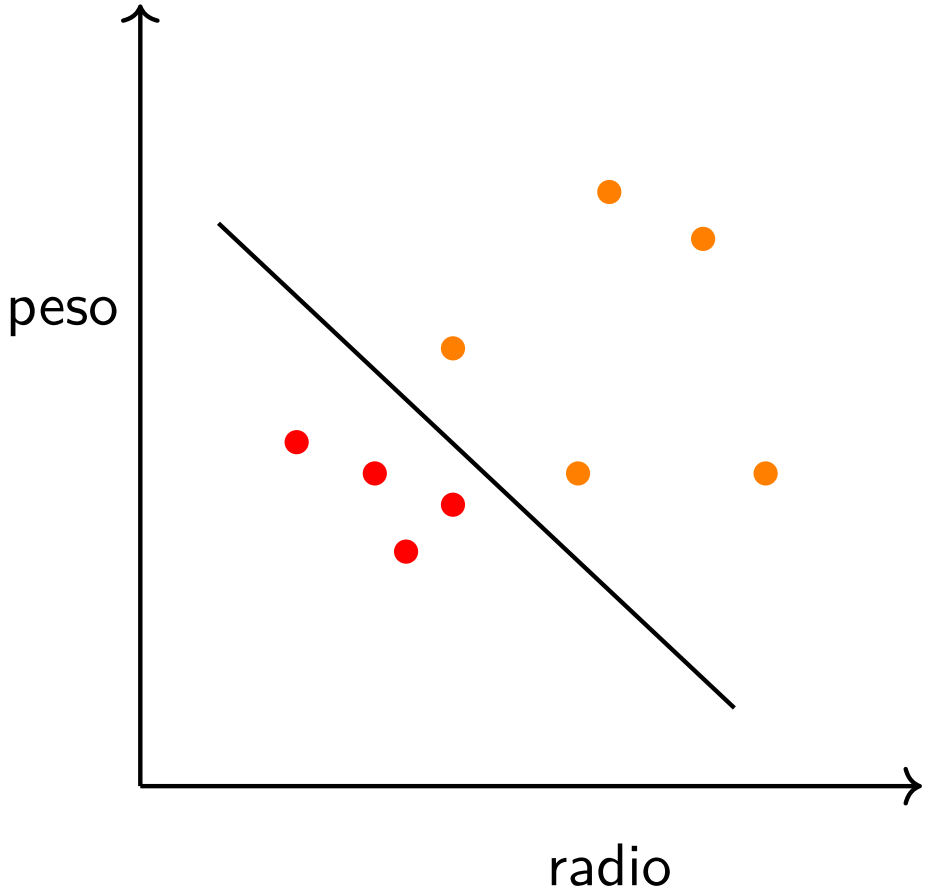
- $\text{tf}(t, d)$: Frecuencia de término t en el documento d
- N: Número total de documentos
- $\text{df}(t)$: Número de documentos que contienen el término t

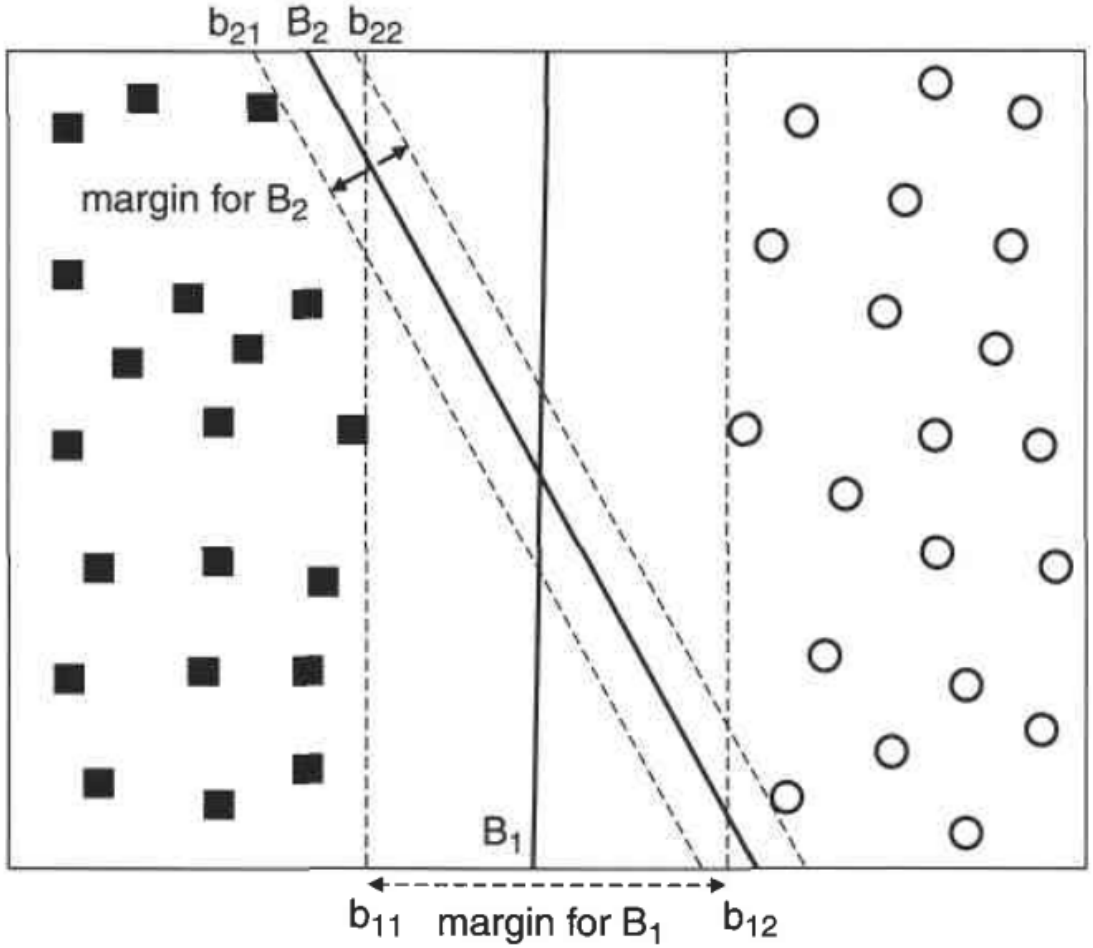
A modo de ejemplo, supongamos que obtenemos una matriz como esta:

doc/words	it	I	the	to	and	yet
d1	0.4	0.8	0.9	0.8	0.7	0.8
d2	0.2	0.6	0.8	0	0.1	0
d3	0.1	0.5	1	0	0	0.1

Se aplica entonces el bayesiano ingenuo sobre la matriz que obtenemos como resultado

Los datos se presentan como vectores. Supongamos que tenemos naranjas y cerezas y conocemos su peso y su radio. Queremos entrenar el modelo para que prediga a partir del peso y el radio si una fruta que no forma parte del entrenamiento es naranja o cereza. Para eso hay que encontrar la función que permita dibujar el hiperplano que divide los datos que son naranjas de los que son cerezas.





(Tomado de Tan *et al.* 2006: 258.)

Asumiendo que los datos se puedan separar mediante una función lineal, para definir este hiperplano se debe encontrar la función lineal cuyos puntos cumplan la siguiente condición:

$\mathbf{w} \cdot \mathbf{x} + b = 0$

En donde tanto w como x son vectores.

La operación $\cdot$ se denomina producto punto. Esta operación consiste en tomar dos vectores necesariamente del mismo tamaño y sumar los productos resultantes de tomar el primer valor del primer vector y multiplicarlo con el primero del segundo, el segundo del primero con el segundo del segundo y así sucesivamente.
producto punto$(v, w) = v \cdot w = \sum_{i=1}^{n} v_1 w_i = v_1 w_1 + v_2 w_2 + ... + V_n w_n$
La longitud de los vectores dependerá de la cantidad de dimensiones.

$$\text{producto punto}([2,3,4],[1,2,2]) = [2,3,4] \cdot [1,2,2] = 2{\times}1 + 3{\times}2 + 4{\times}2$$
$$\text{producto punto}([2,3,4],[1,2,2]) = [2,3,4] \cdot [1,2,2] = 2 + 6 + 8 = 16$$

$$\text{producto punto}([5, 2, 2], [3, 2, 1]) \quad =$$

producto punto$([2, 3, 4], [1, 2, 2, 5]) =$

Dado que el producto punto entre dos

Para cualquier punto que esté sobre el hiperplano, se va a cumplir la siguiente condición:

(12) $\mathbf{w} \cdot \mathbf{x} + b = 0$

Cualquier punto que cumpla con esa condición va a ser un límite de decisión.
Podemos seleccionar dos puntos cualesquiera x1 y x2 y trazar el vector que va de x2 a x1 restándolos:

(13)
a. $\mathbf{w} \cdot \mathbf{x_1} + b = 0$
b. $\mathbf{w} \cdot \mathbf{x_2} + b = 0$
c. $(\mathbf{w} \cdot \mathbf{x_1} + b) - (\mathbf{w} \cdot \mathbf{x_2} + b) = 0$
d. $\mathbf{w} \cdot \mathbf{x_1} - \mathbf{x_2} = 0$

El vector resultante es paralelo al hiperplano de decisión. Dado que el producto punto entre dos vectores es igual a su coseno y que 0 es el resultado del coseno para los ángulos de 90°, podemos concluir que w es perpendicular al límite de decisión.

Para calcular la función que dibuja al hiperplano, se deben encontrar los puntos de dos clases diferentes que sean más cercanos (esto es un hiperparámetro, podemos seleccionar los más lejanos o podemos señalar los más cercanos pero solo en la medida en que estén separados por una distancia mínima).
Para estos puntos se cumple la siguiente condición:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$$

$y_i = 1$ para una clase y -1 para la otra:
Clase 1 =

$$1(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$$

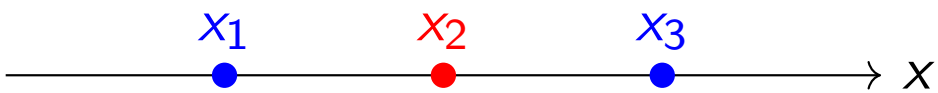
Clase -1 =

$$-1(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$$

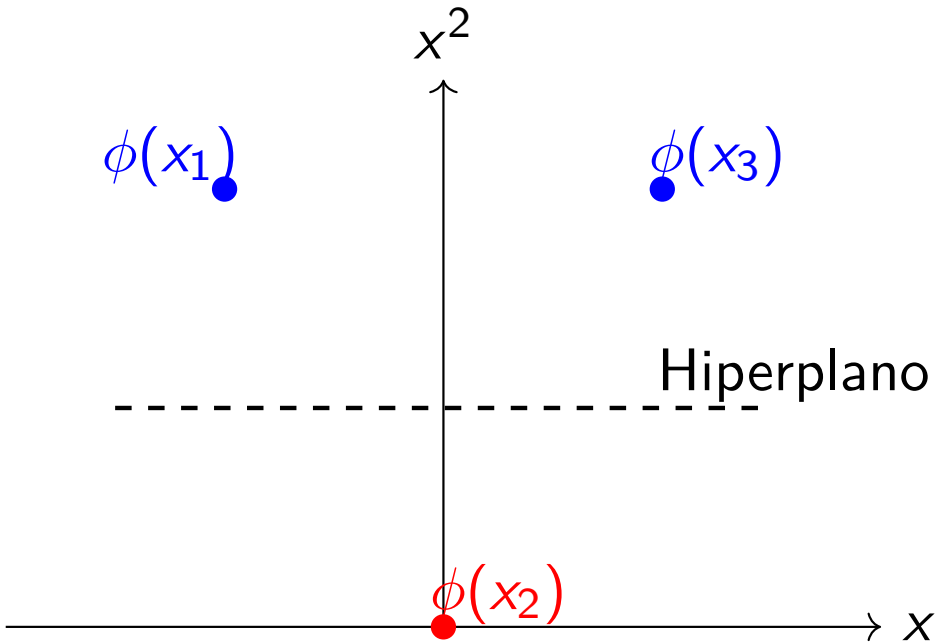
El margen de la decisión está dado por la distancia entre estos dos hiperplanos, es decir, 2, cada uno de los cuales está a una distancia de $\frac{2}{|\mathbf{w}|}$.

- La máquina de soporte vectorial tiene por objetivo encontrar los valores para w tal que cada punto de soporte esté a $\frac{2}{|\mathbf{w}|}$ de la línea de decisión y el b que, una vez calculado w dé $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$ para cada uno de los x que actúa de sorporte. Esto le va a permitir trazar la línea que separa unos datos de otros.
- En el caso en que no sea posible trazar una línea, se aplica alguna transformación sobre el espacio vectorial para que los datos sean separables por una línea. Esto se conoce con el nombre del truco del kernel.

Espacio original (1D)



Espacio transformado (2D)



Regresión logística

- La regresión logística es un clasificador que utiliza aprendizaje supervisado, por lo que requiere de datos anotados.
- En su forma más simple permite decidir si un dato pertenece a una de dos clases. Recurriendo a la regresión logística multinomial se puede agregar más clases. Para presentar sucintamente el algoritmo, nos vamos a limitar a una tarea de decisión con solo dos clases.

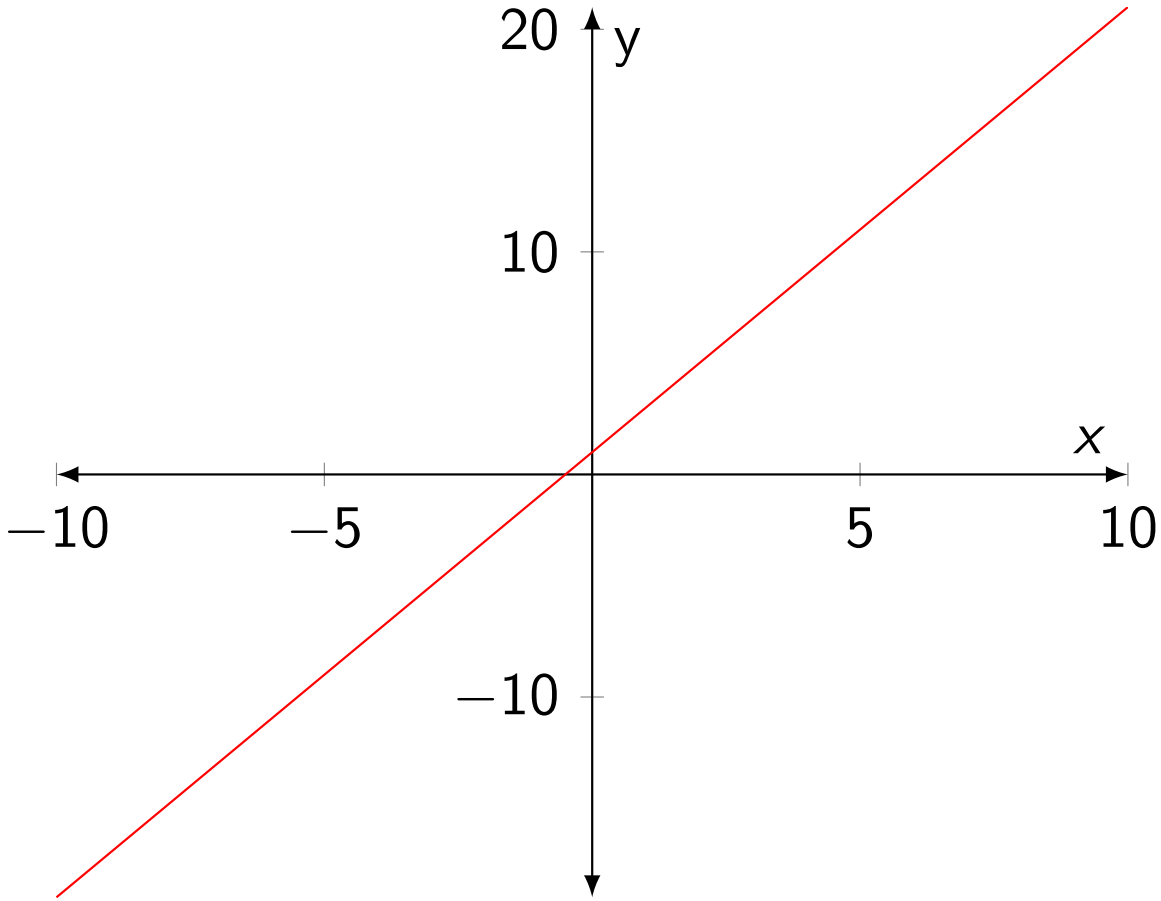
- Cada dato anotado consiste en un vector, esto es, una lista de valores [x1, x2..., xn].
- El entrenamiento consiste en llegar a desarrollar un vector con valores denominados pesos de la misma cardinalidad que el vector que contiene los datos. Este vector de pesos tiene la forma [w1, w2,..., wn]
- También se utiliza un valor numérico denominado bias (sesgo), que se representa con la letra b.

Supongamos que el clasificador ya está entrenado. Para saber si un elemento pertenece o no a la clase, debe sumarse la multiplicación de cada valor en x con su respectivo peso en w y sumar b. La suma de la multiplicación se representa mediante la operación *producto punto*:

$$z = x \cdot w + b \tag{14}$$

El producto punto de [x1, x2,... xn] $\cdot$ [w1, w2,... wn] = x1 $\times$ w1 + x2 $\times$ w2 + ... xn $\times$ wn

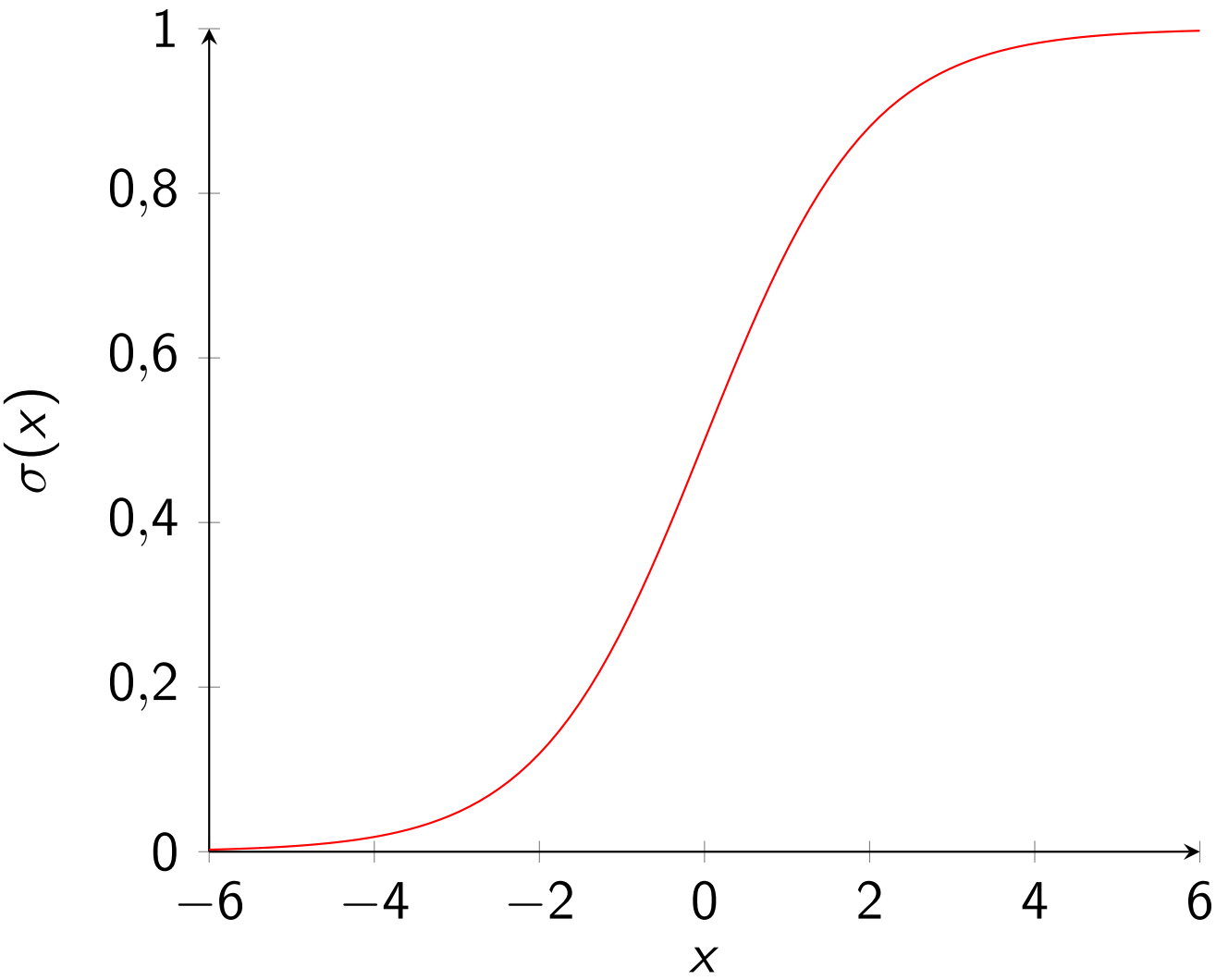
z es una función lineal con un valor que va de $-\infty$ a $+\infty$



- Hay que convertir esta función en una función no lineal que nos sirva para calcular la probabilidad de que el dato pertenezca a una clase o a otra. Un rango muy útil para esto es el de 0 a 1.
- Para la regresión logística se utiliza la función sigmoide (porque tiene forma de s) o función logística:

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

El número al que se aplica esta función sigmoide recibe el nombre de *logit*


1
0,8
0,6
0,4
0,2
0
$\sigma(x)$
−6
−4
−2
0
2
4
6
x

El valor que la función sigmoide da para una u otra clase está normalizado (*i.e.*, los valores suman 1)
El *decision boundary* será el valor a partir del cual se discrimina a qué clase pertenece el dato. En este caso, supongamos que es 0.5:

$$\text{(15)} \quad \text{decision(x)} = \begin{cases} 1 \text{ si } \text{P(y=1|X)} > 0.5 \\ 0 \text{ en cualquier otro caso} \end{cases}$$

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

En el caso de más de una clase, se utiliza regresión logística multinomial, también llamada maxent classifier o softmax regression, porque utiliza softmax como función de activación en lugar de sigmoide.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Donde:

- z_i es el valor de entrada (logit) para la clase i
- K es el número total de clases
- e es la base del logaritmo natural

Una característica de la función softmax es que normaliza los resultados obtenidos, es decir, todos suman 1.
La regresión logística multinominal devuelve un vector y de tipo *one hot vector*, con un valor 1 para la clase predicha y 0 para el resto.

- El entrenamiento en la regresión logística consiste en encontrar los valores apropiados para los parámetros. Estos parámetros son los pesos y el sesgo.
- Para eso, se calcula la distancia entre las predicciones que hace el modelo y las respuestas que debería dar. Esta distancia se conoce como *loss function* o *cost function*.
- La más importante es la *cross-entropy loss*.

Se estiman los valores aleatoriamente. Supongamos que para nuestra tarea de análisis de sentimiento tenemos el dato x y definimos el siguiente w y el siguiente b:

1. x = [3, 2, 1, 3, 0, 4.19]
2. w = [2.5, -5.0, -1.2, 0.5, 2,0,0.7]
3. b = 0.1

Podemos calcular entonces el valor de la probabilidad p(+|x) de la siguiente forma:

1. $p(+|x) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$
2. $p(+|x) = \sigma([2{,}5, -5{,}0, -1{,}2, 0{,}5, 2, 0, 0{,}7] \cdot [3, 2, 1, 3, 0, 4{,}19] + 0{,}1)$
3. $p(+|x) = \sigma(,833)$
4. $p(+|x) = 0.7$

El valor de p(-|x) será entonces 0.3

La función de costo cross entropy para la clase $\hat{y}$ que obtuvo el modelo y la clase y que corresponde a la correcta se define como sigue:

$$L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \cdot \log(1 - \sigma(w \cdot x + b))]$$

Si la aplicamos al ejemplo de arriba en donde p(+|x) = 0.7, obtenemos el siguiente resultado

1. $L_{CE}(\hat{y}, y) = -[1 \log 0{,}7 + (1 - 1) \cdot \log(1 - 0{,}7)]$
2. $L_{CE}(\hat{y}, y) = -[\log 0{,}7 + 0 \cdot \log(1 - 0{,}7)]$

Dado que $0 \cdot \log(1 - 0{,}7) = 0$, el segundo término desaparece:

1. $L_{CE}(\hat{y}, y) = -\log 0{,}7$
2. $L_{CE}(\hat{y}, y) = 0{,}36$

Si el ejemplo hubiese sido negativo, entonces el cálculo sería como sigue:

1. $L_{CE}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \cdot \log(1 - \sigma(w \cdot x + b))]$
2. $L_{CE}(\hat{y}, y) = -[0 \log 0{,}3 + (1 - 0) \cdot \log(1 - 0{,}3)]$
3. $L_{CE}(\hat{y}, y) = -[\log 1 - 0{,}3)]$
4. $L_{CE}(\hat{y}, y) = -\log 0{,}3$
5. $L_{CE}(\hat{y}, y) = 1{,}2$

El valor de cross-entropy da de 0 a ∞.

- Esta distancia debe usarse como insumo para actualizar los parámetros. Estas funciones de optimización incluyen la *exact solution* y, la más conocida, el descenso de gradiente (*gradient descent*).

En esta clase introdujimos cómo funcionan el bayesiano ingenuo, las máquinas de soporte vectorial y la regresión logística y cómo se los puede aplicar a un problema lingüístico.

Bibliografía I

Tan, P.-N., Steinbach, M., y Kumar, V. (2006). *Introduction to data mining*. Pearson, Boston.