Aprendizaje Supervisado

Cristian Cardellino

Presentación

¿Quién Soy?

Cristian Cardellino

Profesor en FAMAF-UNC
Research Scientist en Mercado Libre
Doctor en Ciencias de la Computación, en Aprendizaje Semi-Supervisado y Aprendizaje Profundo para Procesamiento de Lenguaje Natural
https://crscardellino.ar
ccardellino@unc.edu.ar
Twitter/GitHub/LinkedIn/Medium: @crscardellino

Contenidos

Temario del Curso

- ¿Qué es aprendizaje supervisado?
- Aprendizaje supervisado.
  - Repaso: Regresión Lineal y Polinomial, Regresión Logística, Naive Bayes.
- Support Vector Machines.
  - Repaso: Perceptrón.
  - SVC/SVR. Datos no linealmente separables. Función de costo.
- Ensemble learning.
  - Repaso: Decision Trees
  - Random Forest, Bagging, Boosting, Voting.
- Redes neuronales.
  - Perceptrón multicapa.
- Sistemas de recomendación.
  - Filtrado colaborativo.
- Prácticas de reproducibilidad

Primera Clase

Temario de la Clase

- ¿Qué es aprendizaje supervisado?
- Aprendizaje supervisado.
  - Repaso: Regresión Lineal y Polinomial, Regresión Logística, Naive Bayes.
- Support Vector Machines.
  - Repaso: Perceptrón.
  - SVC/SVR. Datos no linealmente separables. Función de costo.
- Ensemble learning.
  - Repaso: Decision Trees
  - Random Forest, Bagging, Boosting, Voting.
- Redes neuronales.
  - Perceptrón multicapa. Redes convolucionales. Redes recurrentes.
- Sistemas de recomendación.
  - Filtrado colaborativo.
- Prácticas de reproducibilidad

Motivación

Ejemplos de aprendizaje supervisado

Una banco recibe nuevos pedidos para acceder a una tarjeta de crédito. Cada pedido tiene información acerca del aplicante:

- Recibo de sueldo
- Edad
- Estado Civil
- Informe de Veraz
- Situación crediticia según el BCRA
- ...

Problema: Determinar si aceptar o rechazar el pedido.

El servicio de correo de una universidad recibe cientos de mails por día.

Problema: Clasificar cada mail como correo basura (spam) o correo deseado, para filtrar y aligerar el servicio.

Descripción del problema: Aprendizaje supervisado

Datos: Se dispone de un conjunto de registros (o ejemplos, o instancias) descritos por *n* atributos: A_1, A_2, ... , A_n y cada instancia está anotada con una etiqueta, pudiendo ser una clase (e.g. Spam / No Spam), o un valor numérico (e.g. score crediticio).

Objetivo: Aprender un modelo (o función) a partir de los datos, buscando predecir sus etiquetas a partir de los atributos. Este modelo puede ser utilizado para predecir las etiquetas de nuevos registros sin anotar.

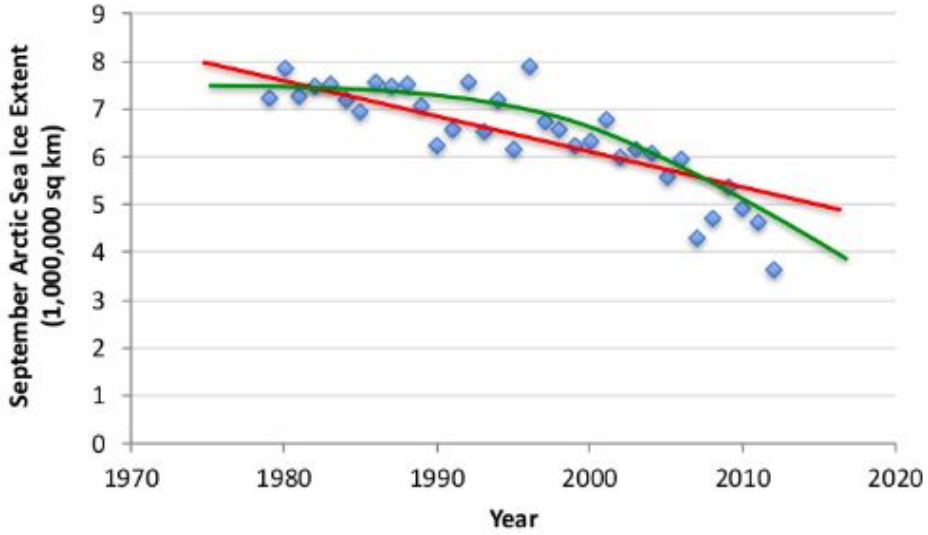
Aprendizaje Supervisado

Regresión

Dados $(x_1,y_1), (x_2, y_2), ..., (x_n, y_n)$

Aprender una $f(x)$ que permita predecir y a partir de x

Si $y \in \mathbb{R}^n$: Es un problema de **regresión**.



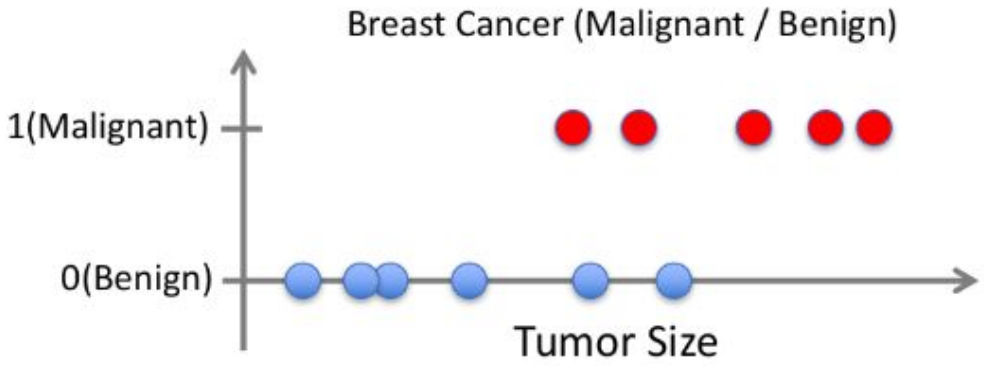
Filminas del curso previo

Clasificación

Dados $(x_1,y_1), (x_2, y_2), ..., (x_n, y_n)$

Aprender una $f(x)$ que permita predecir y a partir de x

Si y es categórica: Es un problema de **clasificación**.



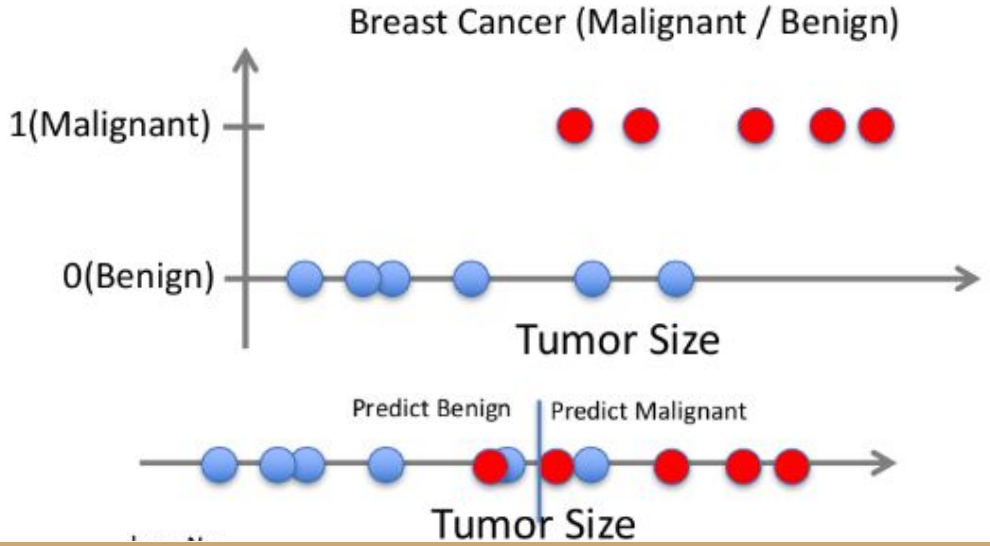
Filminas del curso previo

Clasificación

Dados (x_1,y_1), (x_2, y_2), ..., (x_n, y_n)

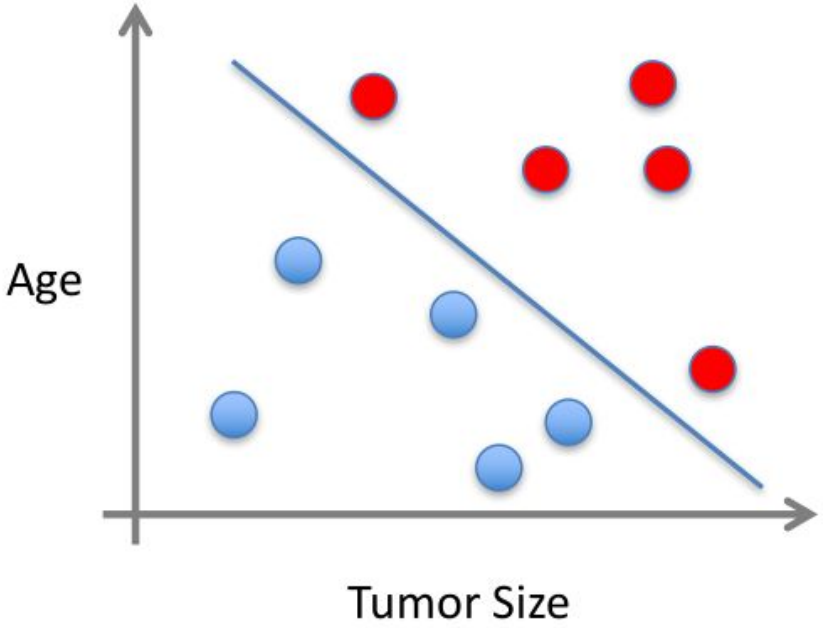
Aprender una $f(x)$ que permita predecir y a partir de x

Si y es categórica: Es un problema de **clasificación**.



Filminas del curso previo

Aprendizaje Supervisado



- La variable x puede ser multidimensional.
- Cada dimensión corresponde a un atributo:
  - Edad del paciente
  - Tamaño del tumor
  - Uniformidad en la forma de la célula
  - Etcétera
- La regresión busca “acercar” los datos a una función (lineal, polinomial, etc.)
- La clasificación busca separar los datos mediante ciertos “bordes”.

Filminas del curso previo

Aprendizaje supervisado

Functions $\mathcal{F}$

$f : \mathcal{X} \to \mathcal{Y}$

Training data

$\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$

LEARNING

find $\hat{f} \in \mathcal{F}$
s.t. $y_i \approx \hat{f}(x_i)$

Learning machine

New data

PREDICTION

$y = \hat{f}(x)$

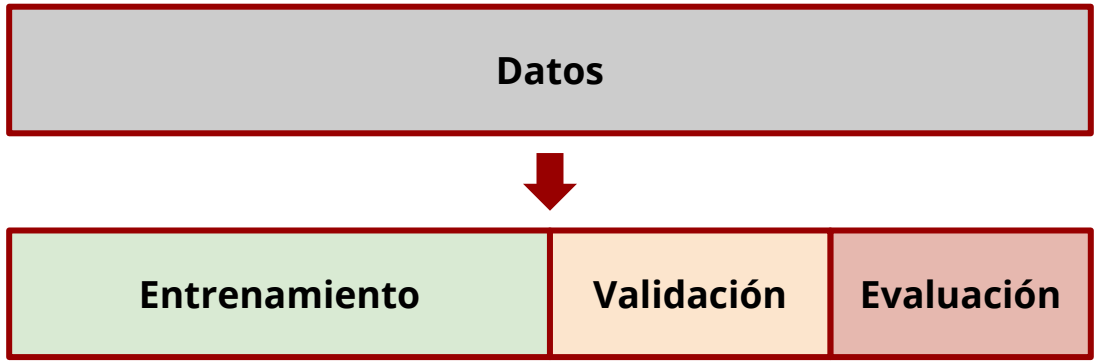
x

Filminas del curso previo

Elección de *hiperparámetros*

Dividir el conjunto total de ejemplos en tres subconjuntos

- **Entrenamiento**: aprendizaje de variables del modelo
- **Validación**: ajuste/elección de hiperparámetros
- **Evaluación**: estimación final del desempeño del modelo entrenado (y con hiperparámetros elegidos adecuadamente)



Filminas del curso previo

Regresión Lineal y Polinomial

Regresión Lineal

Busca ajustar los datos de entrenamiento mediante una función que sea un hiperplano.

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

Los valores θ son los pesos de los atributos (o *features* en inglés).

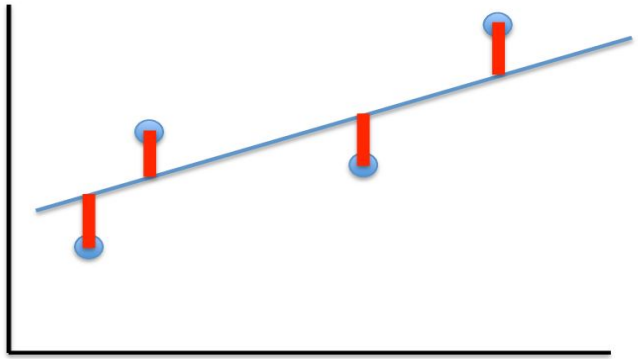
Se entrena minimizando la suma del error cuadrático.

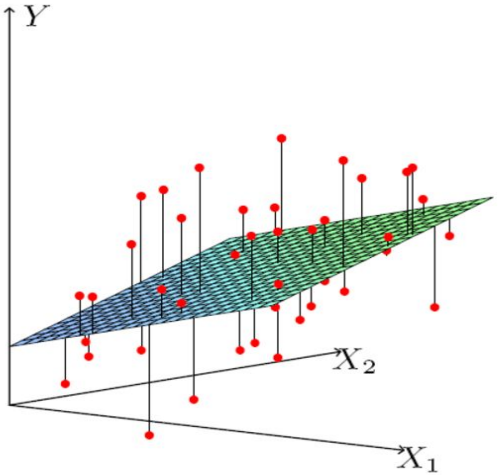
Regresión Lineal: Función de costo

Función objetivo:

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

Se resuelve mediante: $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$





Regresión Polinomial

Busca ajustar los datos de entrenamiento mediante una función polinomial:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Mientras más alto el grado del polinomio, más se ajusta a los datos (pero se vuelve más complejo y tiende a sobreajustar).

Demo Time
(demo_1_linear_regression)

Regresión Logística

Regresión Logística

Usa un enfoque probabilístico.

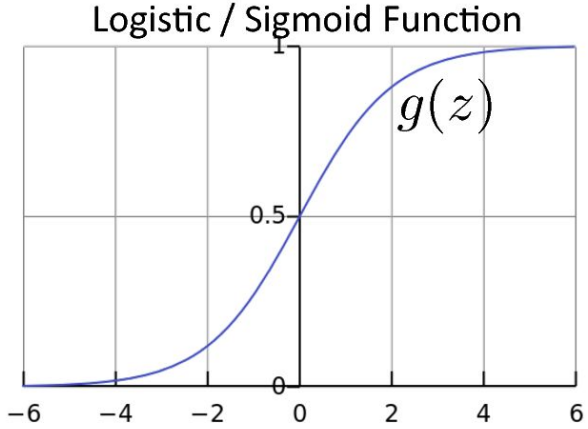
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) \quad \text{debería ser} \quad p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$$

Modelo de regresión logística

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{x}\right)$$

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

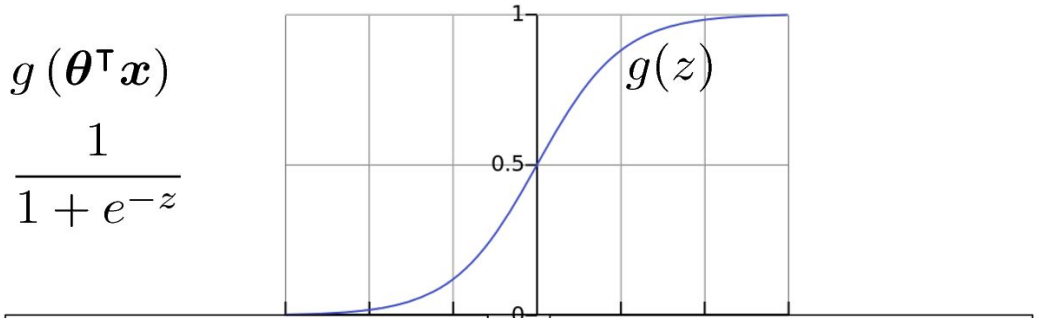
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{x}}}$$



Regresión Logística

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

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

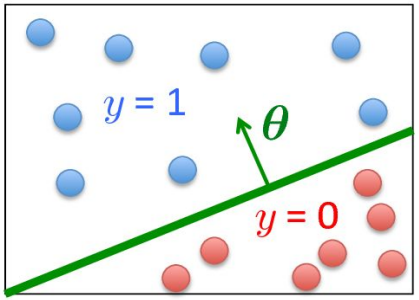


$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ debería tener *valores* ***negativos*** *grandes para instancias negativas* y *valores* ***positivos*** *grandes para instancias positivas*.

Definir un umbral y…

Predecir $y = 1$ si $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$

Predecir $y = 0$ si $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Based on slide by Andrew Ng

Regresión Logística: Función de costo

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

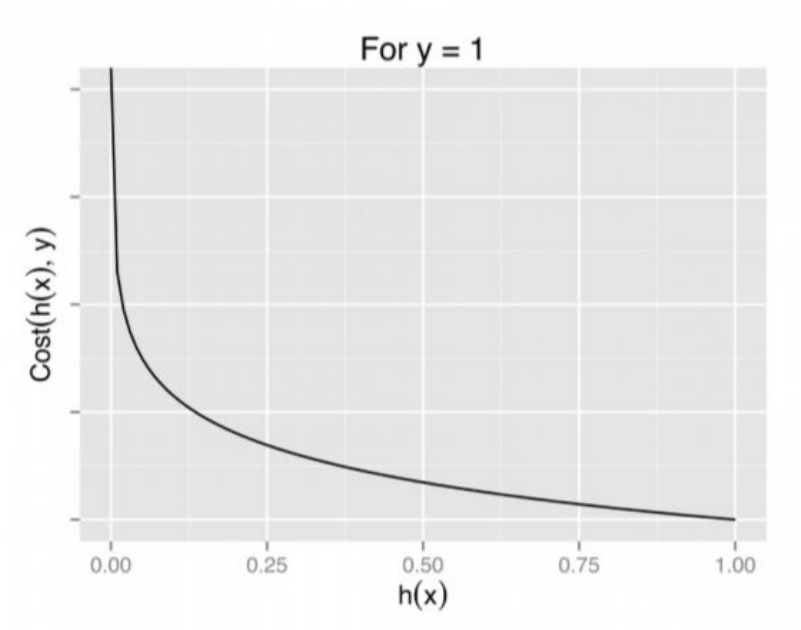
Costo de una sola instancia de los datos

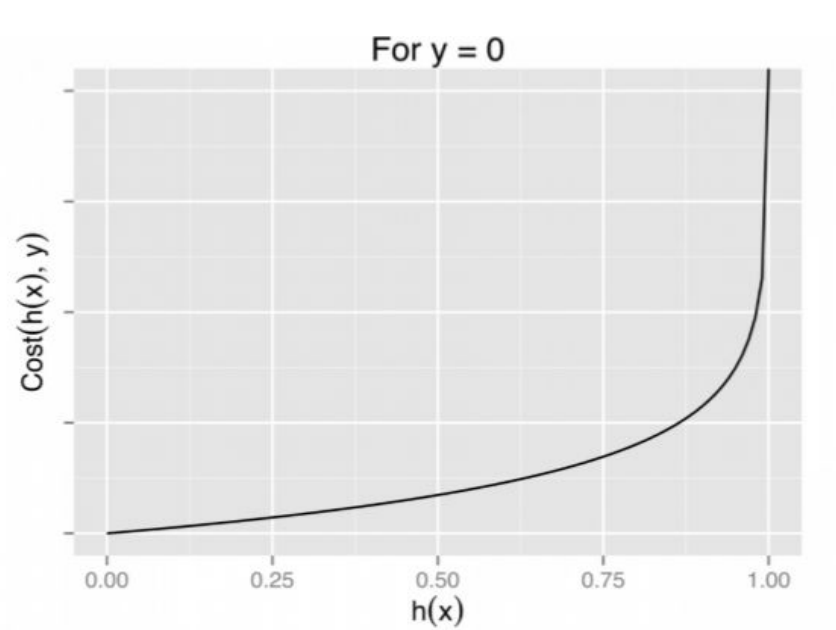
$$\operatorname{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Se reescribe la función de coste como

$$J(\boldsymbol{\theta}) = \sum_{i=1}\operatorname{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}\right)$$

¿Por qué esta función de costo?





Demo Time
(demo_2_logistic_regression)

Naive Bayes

Naive Bayes

Es un clasificador basado en el teorema de Bayes, con una asunción “naive” sobre los datos.

Es muy sencillo de programar y entender.

Sirve mucho como baseline y, aunque simplista, puede tener resultados que sobrepasan a algoritmos mucho más complejos.

Es rápido de entrenar y funciona con datos de mucha dimensionalidad (e.g. es muy útil a la hora de clasificar documentos).

Teorema de Bayes

El algoritmo de “Naive Bayes” está fuertemente ligado al teorema de Bayes.

El Teorema de Bayes establece:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

El teorema establece que se puede encontrar la probabilidad de **A** (e.g. una clase objetivo) dada la ocurrencia de **B** (e.g. un conjunto de features). Es decir, **B** es la evidencia y **A** es la hipótesis.

Naive Bayes: Asunciones

Bajo el teorema de Bayes, la principal asunción es que los atributos son independientes entre sí.

La presencia de un feature no afecta a los otros. Esta asunción es “naive”, por eso el nombre del algoritmo.

Una segunda asunción, es que todos los atributos tienen el mismo efecto en la salida del algoritmo.

Naive Bayes: Utilizando el Teorema de Bayes

En base a lo establecido, se puede utilizar el teorema de Bayes para calcular la probabilidad de una clase y de la siguiente manera:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Donde y representa la clase y X representa el vector de atributos:

$$X = (x_1, x_2, x_3, ..., x_n)$$

Naive Bayes: Utilizando el Teorema de Bayes

Utilizando sustitución en la fórmula anterior obtenemos:

$$P(y|x_1,\ldots,x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

Dado que el denominador es estático para todas las entradas del conjunto de datos se puede ignorar y se establece una proporcionalidad:

$$P(y|x_1,\ldots,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Naive Bayes: Predicción de la clase

En base a lo visto previamente, la predicción de la clase objetivo es sencillamente aquella con mayor probabilidad:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Naive Bayes: Algoritmo

- Calcular la probabilidad para cada clase

  $$P(y) / \forall\ y \in Y$$

- Calcular la probabilidad condicional de cada atributo dada cada clase:

  $$P(x_i|y) / 0 \leq i \leq n, \forall\ y \in Y$$

- Calcular la clase de acuerdo a la que maximice la probabilidad (o, en este caso la proporción):

  $$y = \text{argmax}_y\ P(y) \prod^{n}_{i=1} P(x_i|y)$$

Naive Bayes: Tipos de algoritmos

Bernoulli Naive Bayes: Para casos donde los atributos son variables binarias (e.g. si una palabra ocurre o no en un documento).

Multinomial Naive Bayes: Para casos donde los atributos representan frecuencias (e.g. la cantidad de veces que una palabra ocurre en un documento).

Gaussian Naive Bayes: Para casos donde los atributos toman valores continuos, se asume que los valores son muestras de una distribución gaussiana (esto se usa para calcular las probabilidades condicionales en el algoritmo).

Naive Bayes: Relación con Regresión Logística

Naive Bayes es un modelo **generativo**, es decir, que intenta modelar la probabilidad $p(y,\boldsymbol{x})$ donde "y" representa la etiqueta y "$\boldsymbol{x}$" representa los atributos. A grandes rasgos, trata de generar los atributos del modelo dada la etiqueta.

La Regresión Logística es un modelo **discriminativo**, es decir, que intenta modelar la probabilidad $p(y|\boldsymbol{x})$. A grandes rasgos, determina la etiqueta "y" a partir del vector de atributos "$\boldsymbol{x}$". Esto quiere decir que no le hace falta hacer ninguna asunción sobre $p(\boldsymbol{x})$ ya que no es necesaria para modelar.

Demo Time
(demo_3_naive_bayes)

Algoritmo del perceptrón

El algoritmo del "perceptrón"

Propuesto por Frank Rosenblatt en 1958

El objetivo es encontrar un hiperplano de separación

- Sólo encuentra la solución si los datos son linealmente separables

Es un algoritmo *online* (procesa un ejemplo a la vez)

Filminas del curso previo

El algoritmo del "perceptrón"

Entrada:

- Una conjunto de entrenamiento $(x_1,y_1), (x_2, y_2), ..., (x_n, y_n)$
- Una tasa de aprendizaje r

Algoritmo:

- Inicializar $w^{(0)} \in \mathbb{R}^n$
- Para cada ejemplo (x_i,y_i)
  - Predecir $y_i' = signo(w^T x_i + w_0)$
  - Si $y_i' \neq y_i$:

$$w^{(t+1)} \leftarrow w^{(t)} + r\,(y_i\, x_i)$$

Filminas del curso previo

El algoritmo del "perceptrón"

Entrada:

- Una conjunto de entrenamiento $(x_1,y_1), (x_2, y_2), ..., (x_n, y_n)$
- Una tasa de aprendizaje r

Algoritmo:

- Inicializar $w^{(0)} \in \mathbb{R}^n$
- Para cada ejemplo (x_i,y_i)
  - Predecir $y_i' = signo(w^T x_i + w_0)$
  - Si $y_i' \neq y_i$:

$$w^{(t+1)} \leftarrow w^{(t)} + r\,(y_i\,x_i)$$

Actualiza solo cuando comete un error

Error en positivos:

$$w^{(t+1)} \leftarrow w^{(t)} + r\,x_i$$

Error en negativos:

$$w^{(t+1)} \leftarrow w^{(t)} - r\,x_i$$

Si $y_i w^T x_i \leq 0 \rightarrow$ error

Filminas del curso previo

Demo Time
(demo_4_perceptron)

Introducción a Support Vector Machines

Fronteras de decisión en clasificación

Un clasificador busca separar los datos de una y otra clase de la mejor manera.

Esta separación se da mediante una frontera de decisión.

¿Qué determina que tan “buena” es una frontera de decisión?

¿Qué es una “buena” frontera de decisión?

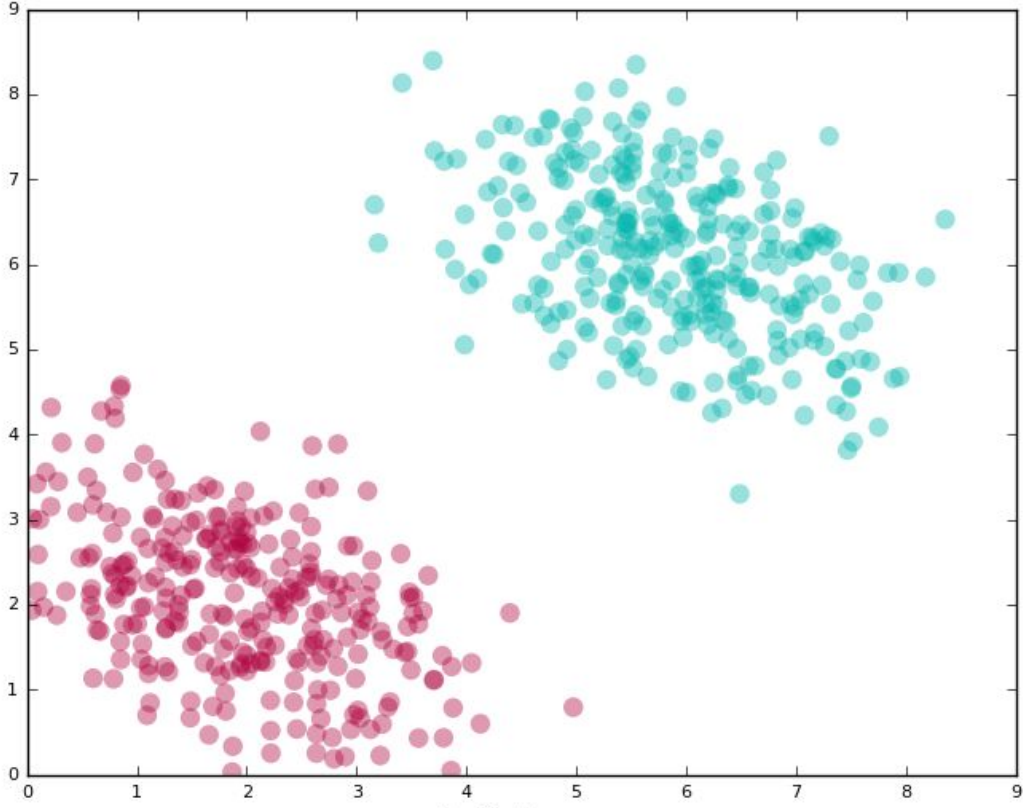
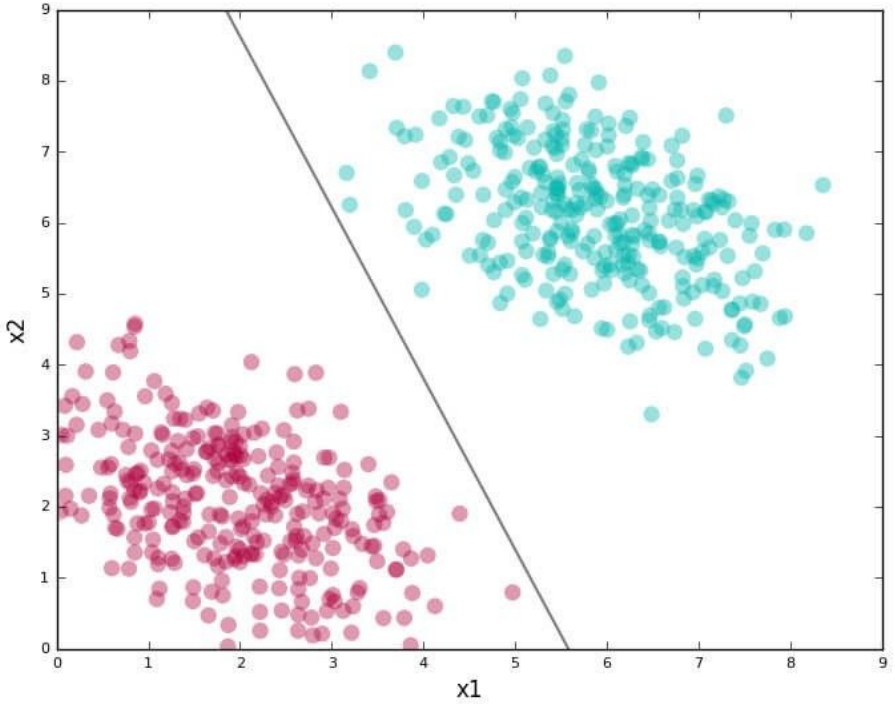


Image from https://blog.statsbot.co/

¿Qué es una “buena” frontera de decisión?



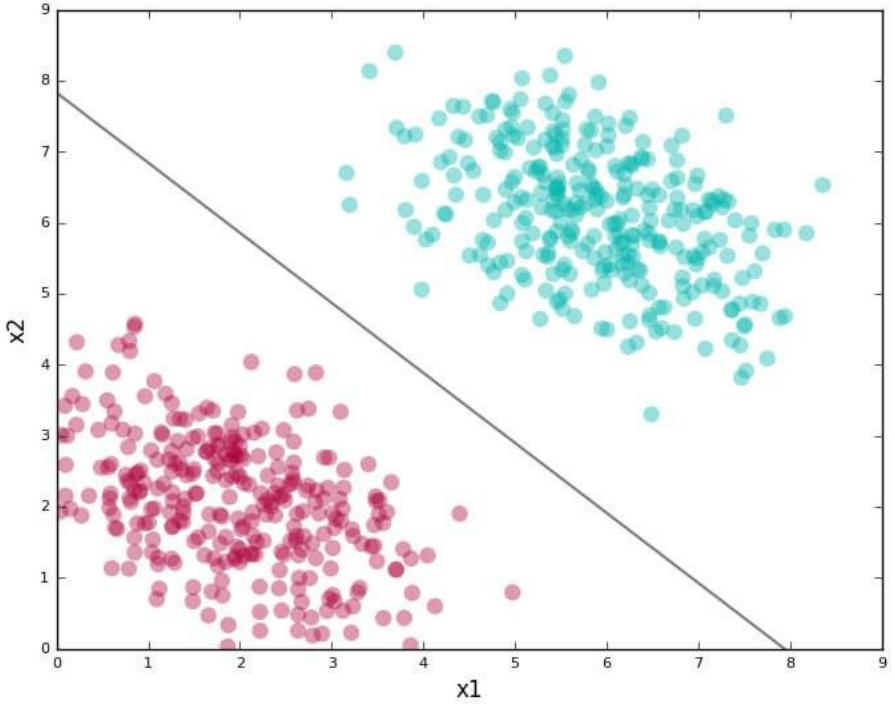


Image from https://blog.statsbot.co/

Margen de la frontera

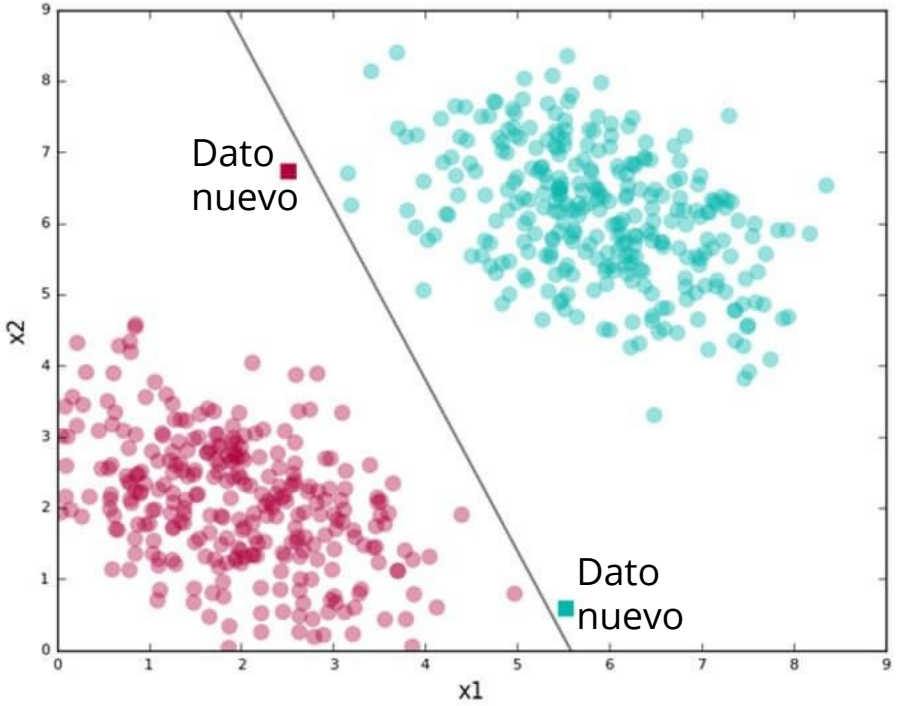
En el gráfico anterior, cualquiera de las líneas separan los datos correctamente.

Buscamos una línea que capture el patrón general entre los datos.

En el gráfico de la izquierda, la línea de separación está algo sesgada. Tiene menos margen entre ella y ambos clústeres de datos.

La línea en el gráfico de la derecha, en cambio, se encuentra bien a la mitad de ambos clústeres.

¿Qué pasa al clasificar nuevos datos?



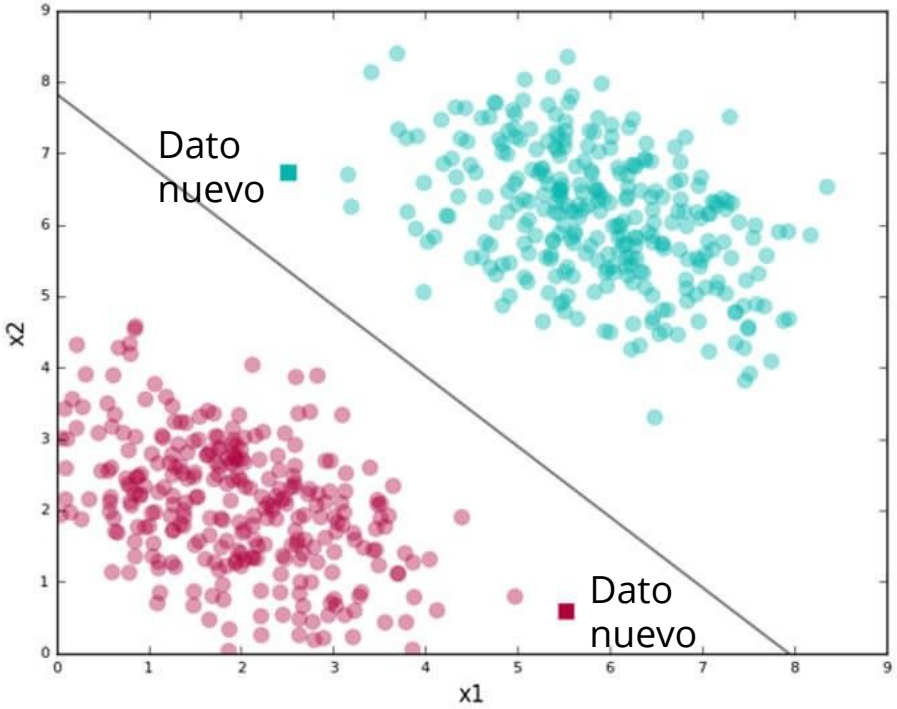


Image from https://blog.statsbot.co/

Support Vector Machines

Es un algoritmo que busca separar los datos mediante la mejor frontera de decisión. Esta frontera de decisión es conocida como **hiperplano**.

En este caso, “mejor” se refiere a aquella que esté lo más separada posible de los puntos más cercanos a ella. Estos puntos son conocidos como **vectores de soporte**, y el espacio entre ellos y el hiperplano se conoce como **margen**.

En términos más técnicos, un algoritmo de SVM encuentra el hiperplano que devuelva el mayor margen entre sí mismo y los vectores de soporte.

Este tipo de clasificador a veces es conocido como “clasificador por márgenes” (margin classifier).

Support Vector Machines

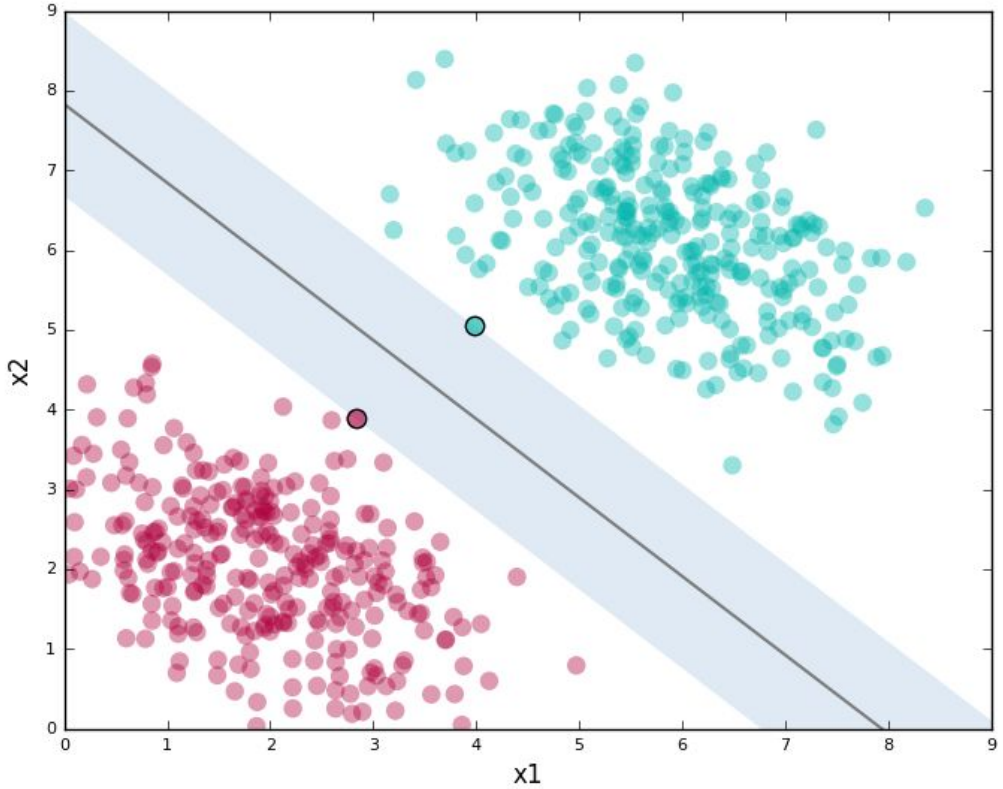


Image from https://blog.statsbot.co/

SVM: Función de Costo y Entrenamiento

SVM: Función de costo

Los SVM utilizan una función de costo conocida como hinge loss.

A diferencia de regresión logística, los datos se anotan con $\{-1, 1\}$ de acuerdo al valor de la etiqueta.

La función de costo de Hinge se define como:

$$c(x, y, f(x)) = max(0, 1 - y * f(x))$$

Donde el costo es 0 si el valor real y el predicho tienen el mismo signo y están dentro del margen de error (por lo general 1).

SVM: Función a optimizar

La función que buscamos minimizar es la siguiente:

$$min_w \sum_{i=1}^{n} max(0, 1 - y_i \langle x_i, w \rangle) + \lambda ||w||^2$$

Dónde $\lambda ||w||^2$ es el parámetro de regularización.

SVM: Gradientes

Tenemos dos factores en la función de costo que hay que derivar:

$$\frac{\delta}{\delta w_k}\lambda||w||^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} max(0, 1 - y_i\langle x_i, w\rangle) = \begin{cases} 0 & \text{si } y_i\langle x_i, w\rangle \geq 1 \\ -y_i x_{ik} & \text{c.c.} \end{cases}$$

SVM: Actualización de los pesos

Al actualizar los pesos, de acuerdo al signo de la predicción, tendremos para el caso donde el signo sea el mismo:

$$w = w - \alpha(2\lambda w)$$

Mientras que cuando el signo entre la predicción y el valor real es diferente:

$$w = w + \alpha(y_i x_i - 2\lambda w)$$

Fin de la primera clase