Módulo 3. IA y grandes volúmenes de datos #2. El problema de clasificación

**Clasificación binaria**

● Disponemos de *N* pares de entrenamiento (observaciones) 

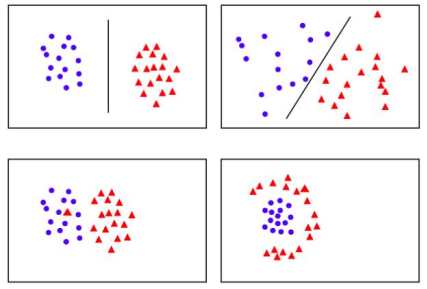
con *xi*ϵℝn, *yi*ϵ{-1, +1}.

● Aprender una *f*(*x*) tal que



es decir: *yi f*(*xi*) > 0 para una clasificación correcta.

**Separabilidad lineal**

linealmente 

separable

**no**

linealmente

separable

**Clasificadores lineales**

● La entrada es un vector **x***i* de dimensionalidad *n* ● La salida es una etiqueta *yi*ϵ{-1, +1}

● Clasificador = función de predicción + función de decisión *g*(*f*(**x**)) → {-1, +1}

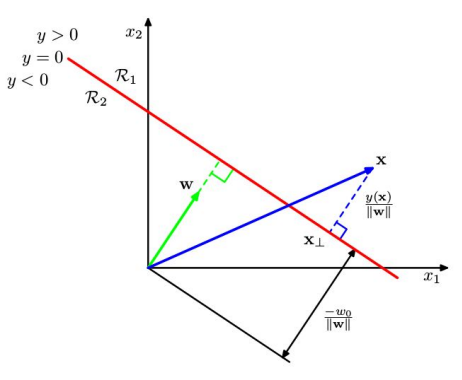
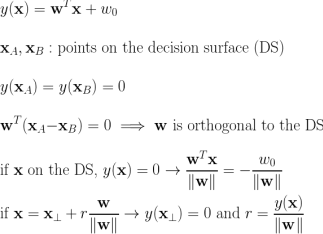
● Función de predicción **lineal**

*f*(**x**) = **w**T**x**+*w*0

● Función de decisión

*g*(*z*) = *sign*(*z*)

*g*(*f*(**x**)) = *sign*(**w**T**x**+*w*0)

puntos en la frontera de decisión (FD) 

es ortogonal a la FD

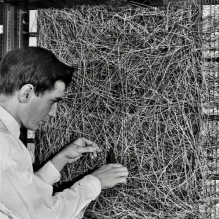
en la FD,

si

si

y

**El algoritmo del "perceptrón"**

● Propuesto por Rosemblatt en 1958 

● El objetivo es encontrar un hiperplano de

separación. **Si los datos son linealmente**

**separables, lo encuentra.**

● Es un algoritmo *online* (procesa un ejemplo

a la vez)

● Muchas variantes ...

**El algoritmo del "perceptrón"**

Entrada:

● una secuencia de pares de entrenamiento (**x**1,*y*1), (**x**2, *y*2) ... ● Una tasa de aprendizaje *r* (número pequeño y menor a 1)

Algoritmo:

● Inicializar **w**(0)ϵℝn

● Para cada ejemplo (**x***i*,*yi*)

○ Predecir *yi*'*=sign*(**w**T**x***i*)

○ Si *yi*' ≠ *yi*:

**w**(t+1) ← **w**(t) + *r* (*yi* **x***i*)

**El algoritmo del "perceptrón"**

Entrada:

● una secuencia de pares de entrenamiento (**x**1,*y*1), (**x**2, *y*2) ... ● Una tasa de aprendizaje *r* (número pequeño y menor a 1)

Algoritmo:

● Inicializar **w**(0)ϵℝn

● Para cada ejemplo (**x***i*,*yi*) ○ Predecir *yi*'*=sign*(**w**T**x***i*) ○ Si *yi*' ≠ *yi*:

**w**(t+1) ← **w**(t) + *r* (*yi* **x***i*)

Actualiza solo cuando comete un error Error en positivos:

**w**(t+1) ← **w**(t) + *r* **x***i*

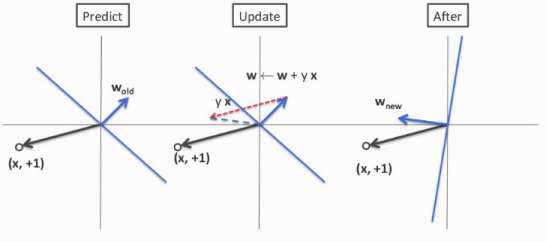
Error en negativos:

**w**(t+1) ← **w**(t) - *r* **x***i*

Si *yi***w**T**x***i* ≤ 0 → error

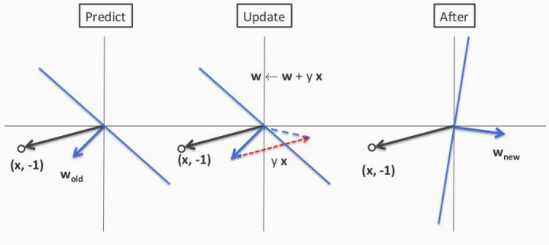
**Dinámica de actualización**

Error en ejemplo **positivo**:



**Dinámica de actualización**

Error en ejemplo **negativo**:



**El algoritmo estándar**

Dado un conjunto D={(**x**i, *y*i), i=1, ..., N}, *y*i ∊ {-1, +1}, taza de entrenamiento *r* y número de épocas *T*

1. Inicializar **w**(0)

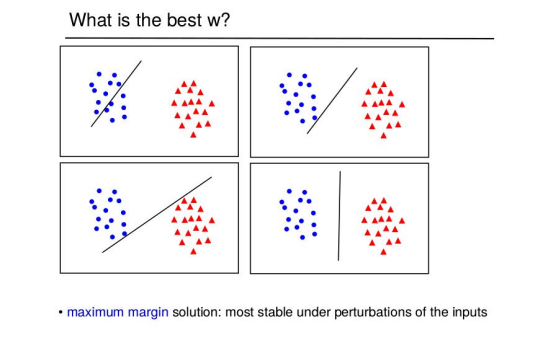
2. Para época *t*=1, …, *T*

a. *barajar* el conjunto de entrenamiento D

r, T : hiperparámetros

b. Para cada muestra de entrenamiento (**x**i, *y*i)∊D ■ si *yi***w**(t)T**x***i* ≤ 0, actualizar **w**(t+1) ← **w**(t) + *r* (*yi* **x***i*) 3. Retornar **w**(T)

**Predicción:** *sgn*(**w**T**x)**

**¿Cuál es el mejor *w*?** 

Solución de **margen máximo**: el hiperplano más estable ante perturbaciones de la entrada

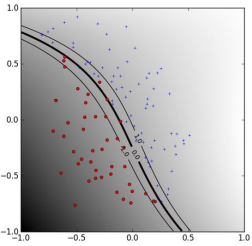
**Generalización en clasificación**

● Complejidad del modelo ⇔ complejidad de la frontera de decisión Regresión

Clasificación

Regresión logística

**Clasificación basada en probabilidades**

● Objetivo: dar una estimación de probabilidad de que una instancia *x* sea de una clase *y*, es decir, p(*y*|*x*) 

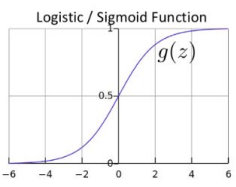
● Recordar:

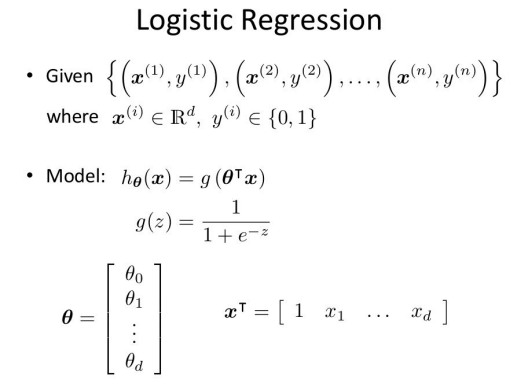
0 ≤ p(*evento*) ≤ 1

p(*evento*) + p(¬*evento*) = 1

**Regresión logística**

● Aproximación probabilística al problema de clasificación ● La función de predicción h*w*(*x*) debe dar una aproximación de p(*y*=1|*x*,*w*) ● 0 ≤ h*w*(*x*) ≤ 1



**Regresión logística** 

Dados

donde

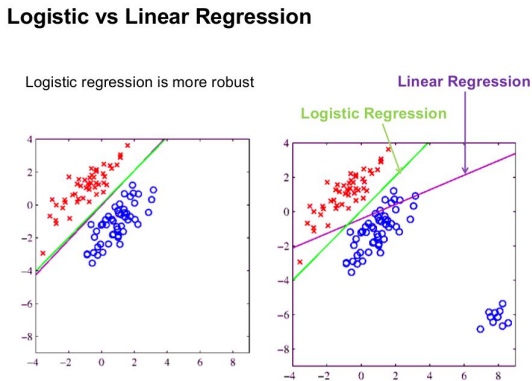
Modelo:

**Regresión logística. Función de costo**

● Conjunto de entrenamiento

● *y*: observaciones discretas ￫ muestras de una distribución Bernoulli 

● Encontrar el ***w*** que maximice la verosimilitud de las etiquetas en el conjunto de entrenamiento

**Regresión lineal vs. regresión logística**

Problemas multiclase

**Clasificación multiclase**

● Una muestra puede pertenecer a 1 (o más) de K clases ○ Datos de entrenamiento {(**x***i*, *y*i)}, *y*i=1,…, K

● Distintos tipos de problemas:

○ multiclase: **x** pertenece solo a una categoría

○ multietiqueta: **x** puede pertenecer a más de una categoría

● A veces es más fácil descomponer el problema multiclase en una serie de problemas binarios. **Distintas estrategias: OVA, AVA, ...**

**Estrategia uno contra todos (OVA)**

**● Asumimos que cada clase es separable del resto** ● Dado un conjunto de entrenamiento D={(**x**i, *y*i)}, *y*i =1,...,K ○ Descomponer el problema en K problemas binarios. Para la

clase *k*, crear un problema tal que:

■ Ejemplos cuya etiqueta es *y*i =k son ejemplos positivos ■ Ejemplos cuya etiqueta es *y*i ≠k son ejemplos negativos ○ Generar K clasificadores binarios con **función de predicción**

*f*k(**x**), k=1,...,K.

● Predicción (*winner takes all*): k\* = argmax *f*k(**x**)

**Estrategia uno contra todos (OVA)**

**Estrategia todos contra todos (AVA)**

**● Asumimos que cada clase par de clases es separable** ● Dado un conjunto de entrenamiento D={(**x**i, *y*i)}, *y*i =1,...,K ○ Descomponer el problema en K(K-1)/2 problemas binarios.

Para el par de clases *(i, j), i*≠*j*, crear un problema tal que: ■ Ejemplos cuya etiqueta es *y*i =*i* son ejemplos positivos ■ Ejemplos cuya etiqueta es *y*i =*j* son ejemplos negativos ○ Generar K(K-1)/2 clasificadores binarios con **función de**

**decisión** *g*(i, j)(**x**)

● Predicción (*voting*): cada clase recibe K-1 “votos”

**Estrategia todos contra todos (AVA)**

**Regresión logística multiclase** ● Para dos clases: 

peso asignado

a y=0

● Para *C* clases (*c*=1,...,*C*):

(función **softmax**)

peso asignado a y=1

Métricas en clasificación

**Importancia de las métricas**

● La función de costo es solo un *proxy* al problema en el mundo real ● Las métricas ayudan a capturar objetivos reales en forma cuantitativa (no todos los errores son iguales)

● Ayudan a la organización el trabajo de los equipos en función de los requerimientos del problema

● Permiten cuantificar diferencias en:

○ performance deseada vs modelo base

○ performance deseada vs actual

○ evolución en el tiempo

● Deberían ser el objetivo del entrenamiento, pero a veces es difícil.

**Clasificación binaria**

● Entrada: *x*, salida: *y* (valores 0/1 o -1/+1)

● Predicción del modelo: *ŷ*=*h*(*x*)

● Dos tipos de modelos:

○ Modelos que predicen directamente una variable categórica (kNN, árboles de decisión)

○ Modelos que predicen un puntaje (*score*) (SVM, regresión logística) ■ Se necesita elegir un umbral (func. de decisión)

■ Nos enfocaremos en esta última clase de modelos. Los anteriores se pueden ver como un caso especial.

**Modelos basados en *scores***

**Umbral** ⟶ **clasificador** ⟶ **métrica puntual**

**Matriz de confusión**

● la suma total es fija (muestra)

● la suma por columnas es fija

(muestras por clase)

● la calidad del modelo y el valor de

umbral deciden el agrupamiento de

filas

● queremos que los elementos

diagonales tengan valores grandes y

los no diagonales valores chicos

**Matriz de confusión**

*● true positives* (TP) = 9

*● true negatives* (TN) = 8

*● false positives* (FP) = 2

*● false negatives* (FN) = 1

**Métricas puntuales: exactitud (*accuracy*) **

● acc=(TP+TN)/(TP+FP+TN+FN)

● equivalente al costo 0/1

**Métricas puntuales: precisión**

****

● Prec=TP/(TP+FP)

● Prec 100% = todos bajo el umbral

salvo el de score más alto (siempre

que sea correcto)

**Métricas puntuales: sensitividad (*recall*)**

****

● Recall=TP/(TP+FN)

● Recall 100% = todos los puntos por

encima del umbral

**Métricas puntuales: F1-score**

****

**Métricas puntuales: cambio del umbral**

****

# umbrales efectivos = # ejemplos + 1

**Métricas resumen: curvas ROC (rotada)**

*specificity*=tnr=TN/Neg=TN/(TN+FP) 

*sensitivity*=tpr=TP/Pos=TP/(TP+FN)

métrica AUC=área bajo la curva ROC

random guessing

ROC=Receiver Operating Characteristic

**Métricas resumen: curvas PR**

*precision*=TP/(pos predichos) 

*recall*=TP/(pos verdaderos)

AUPR=área bajo la curva PR

*fracción de muestras positivas en el dataset*

PR=Precision-Recall

**Curvas ROC y PR en validación cruzada**

Opción 1:

● Asumir que magnitudes de los *scores* son comparables entre corridas ● Acumular predicciones de todas las corridas

● Trazar la curva usando predicciones acumuladas

Opción 2:

● Trazar las curvas individuales para cada partición

● Considerar la “curva promedio”

**Resumen curvas ROC y PR**

● Permiten evaluación cuantitativa a distintos niveles de “confianza” ● Asumen problemas binarios

● Se pueden resumir en medidas del tipo “área bajo la curva” ● Las curvas ROC son insensibles a cambios en la distribución de clases en el conjunto de test

● Las curvas PR muestran la fracción de las predicciones que son FP ● Las curvas PR son útiles en problemas con una proporción de muestras negativas muy alta

● Permiten determinar umbrales óptimos para distintos puntos de operación

**Problemas multiclase**

● Problema con N clases => matriz de confusión de NxN

● La mayoría de las métricas se analizan como N problemas binarios (OVA) ○ El desbalance crece con el

número de clases

● Variantes multiclase de métricas AUC

○ micro vs. macro average