

Clasificadores Discriminativos Básicos

Aprendizaje Automático INF-398 II-2021

Ricardo Nanculef

DI UTFSM Campus San Joaquín

Table of contents

1. Teoría de Decisión
2. El Perceptrón
3. El Regresor Logístico

Teoría de Decisión

Dado un espacio de entrada \mathbb{X} , un conjunto finito de categorías (clases, o etiquetas) $\mathbb{Y} = \{c_1, c_2, \dots, c_K\}$, y un conjunto finito de ejemplos $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^n \subset \mathbb{X} \times \mathbb{Y}$, un **problema de clasificación** consiste en encontrar una función $f : \mathbb{X} \rightarrow \mathbb{Y}$ que minimice la probabilidad de asignar incorrectamente una categoría a un patrón de entrada $x \in \mathbb{X}$, es decir que minimice:

$$M(f) = P(y \neq f(x)) = \sum_x \sum_{y \neq f(x)} p(x, y) , \quad (1)$$

donde $p(x, y)$ denota la función de probabilidad con la cuál se generan los ejemplos.

Clasificación como Aprendizaje Supervisado

Es fácil demostrar que un problema de clasificación es un caso especial de aprendizaje desde ejemplos si elegimos como función de pérdida la denominada **misclassification loss**:

$$L(f(x), y) = I(f(x) \neq y) . \quad (2)$$

Aquí $I(\cdot)$ es una función indicatriz, es decir una función que toma el valor 1 si su argumento es verdadero y 0 si es falso. En efecto, con esta elección de L ,

$$\begin{aligned} R(f) &= \mathbb{E}(L(f(x), y)) = \sum_{x,y} I(f(x) \neq y) p(x, y) \\ &= \sum_x \sum_{y \neq f(x)} p(x, y) = P(y \neq f(x)) . \end{aligned} \quad (3)$$

Notar que \mathbb{Y} es finito y por lo tanto discreto.

Implementación de un Clasificador

Decisiones necesarias para obtener un clasificador:

- Una representación de los datos (dominio para f)
- Un espacio de hipótesis desde donde elegir f .
- Un criterio de entrenamiento $\hat{R}(f)$ que dependa de los datos/ejemplos (experiencia).
- Un algoritmo para minimizar $\hat{R}(f)$.

Criterio de entrenamiento simple: fracción de errores de entrenamiento,

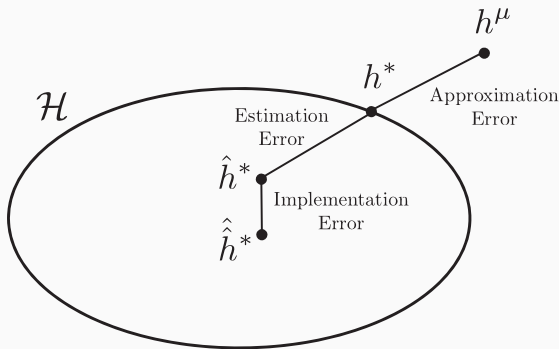
$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n I(f(x^{(i)}) \neq y^{(i)}). \quad (4)$$

Advertencia: posible overfitting!

Tipos de Errores

Las elecciones anteriores generan en realidad al menos 3 tipos de error:

(i) error de estimación, (ii) error de aproximación, y (iii) error de optimización.



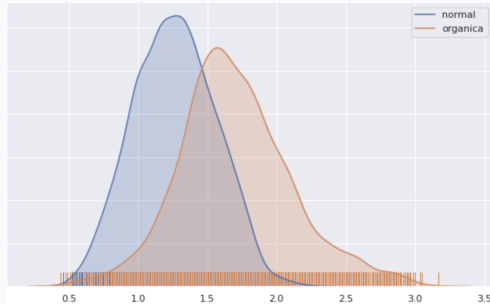
Fuentes de error:

- (i) **Estadístico**: Conjunto de ejemplos es finito.
- (ii) **Funcional**: Espacio de hipótesis es reducido.
- (iii) **Computacional**: Algoritmo de optimización es imperfecto.

Imaginemos que superamos estos problemas (i.e. tenemos datos infinitos, hipótesis arbitrarias y algoritmos completos), ¿Hay alguna otra fuente de error?

Separabilidad de las Clases

Sí, existe un error asociado a la aleatoriedad de las clases que queremos reconocer.

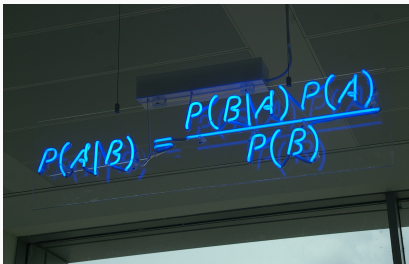


Curva azul: precio de paltas normales. Curva naranja: precio de paltas orgánicas. Tarea: clasificar una palta en base a su precio.

Error & Clasificador de Bayes

Supongamos que eliminamos/ignoramos las fuentes de error i)-iii). ¿Cuál el mejor clasificador que podemos construir?

En reconocimiento de patrones, este clasificador se denomina **Clasificador de Bayes** (Bayes Classifier).


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



Este clasificador nos dará una **cota inferior** para el error de clasificación que nos podemos esperar de **cualquier otro clasificador**.

La cota se denomina **Error de Bayes**:

$$B^* = \min_f \mathbb{E}(L(f(x), y)) = \min_f P(y \neq f(x)). \quad (5)$$

¿Cómo obtener este valor mínimo?

Tenemos que

$$\begin{aligned}\mathbb{E}(L(f(x), y)) &= \sum_{x,y} I(f(x) \neq y)p(x, y) \\ &= \sum_x \sum_y I(f(x) \neq y)p(y|x)p(x) \\ &= \sum_x B(x)p(x).\end{aligned}\tag{6}$$

Es claro que la “suma” $\mathbb{E}(L(f(x), y)) = \sum_x B(x)p(x)$ se puede minimizar si $\forall x$ elegimos el valor mínimo de $B(x)$.

Ahora, si $f(x)$ puede asignar sólo 1 categoría a x (es determinista), tenemos que

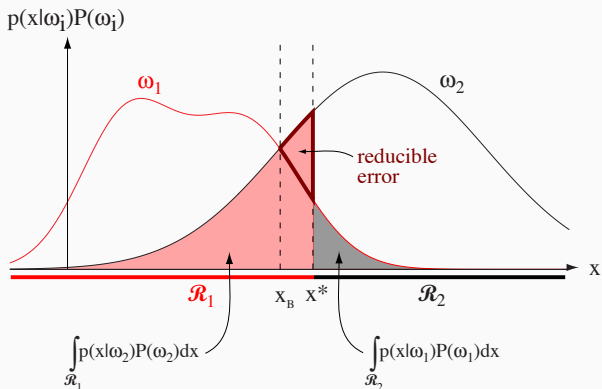
$$B(x) = \sum_y I(f(x) \neq y) p(y|x) = \sum_{y:f(x) \neq y} p(y|x) = 1 - p(f(x)|x).$$

De aquí, es claro que el mínimo de $B(x)$ se logra implementando la regla

$$f^*(x) = \arg \max_{y \in \mathbb{Y}} p(y|x). \quad (7)$$

Error & Clasificador de Bayes

Gráficamente,



Decisión A-Priori vs. Decisión A-Posteriori

En palabras: la regla de decisión óptima (para la *loss* elegida) consiste en elegir la categoría c_j que maximiza la probabilidad $P(c_j|x)$, denominada *a-posteriori* (después de ver x).

Esta regla contrasta con la elección “no informada” o *a-priori*, que consistiría en elegir la clase c_j que maximiza $p(c_j)$.

La relación entre las decisiones *a-priori* y *a-posteriori* viene dada por:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \propto p(x|y)p(y). \quad (8)$$

Decisión A-Priori vs. Decisión A-Posteriori

Ejemplo

Consideremos una palta que pesa $x = 1.4$.

La elección *a priori* corresponde a elegir el tipo de palta que aparece con mayor frecuencia.

La elección *a posteriori* corresponde en cambio a elegir la clase más frecuente de entre las paltas que pesan $x = 1.4$.

En general, la elección anterior difiere de elegir la clase en la cuál es más probable que $x = 1.4$.

Decisión A-Priori vs. Decisión A-Posteriori

Ejemplo

Supongamos que un 40% de las paltas son orgánicas (1).

Supongamos que de entre las paltas orgánicas (1), un 65% pesa $x = 1.4$.

Supongamos que de entre las paltas ordinarias (0), un 35% pesa $x = 1.4$.

Si una palta pesa $x = 1.4$ ¿Cómo debemos clasificar la palta?

Decisión A-Priori vs. Decisión A-Posteriori

$$\begin{aligned}P(y = 1|x = 1.4) &= \frac{P(x = 1.4|y = 1)P(y = 1)}{P(x = 1.4)} \\&= \frac{P(x = 1.4|y = 1)P(y = 1)}{P(x = 1.4|y = 1)P(y = 1) + P(x = 1.4|y = 0)P(y = 0)} \\&= \frac{0.65 \cdot 0.4}{0.65 \cdot 0.4 + 0.35 \cdot 0.6} = \frac{0.26}{0.47} = 0.553\end{aligned}$$

$$P(y = 0|x = 1.4) = 1 - P(y = 1|x = 1.4) = 0.446$$

Clasificadores Probabilistas vs. No-Probabilistas

La discusión anterior sugiere que podemos construir un clasificador estimando directamente $\text{argmax } p(y|x)$ o bien estimando $p(y|x) \forall y$ y luego encontrando la clase y^* que maximiza esa probabilidad.

Este segundo camino dará origen a una familia de **clasificadores probabilistas**, que no se entrenan para minimizar

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n I(f(x^{(i)}) \neq y^{(i)}), \quad (9)$$

sino que optimizan un **criterio “variacional”**

$$E(\theta) = D(p(y|x) || q_{\theta}(y|x)), \quad (10)$$

donde $q_{\theta}(y|x)$ es la aproximación a $p(y|x)$ implementada por la máquina y θ son los parámetros de los que depende esta aproximación.

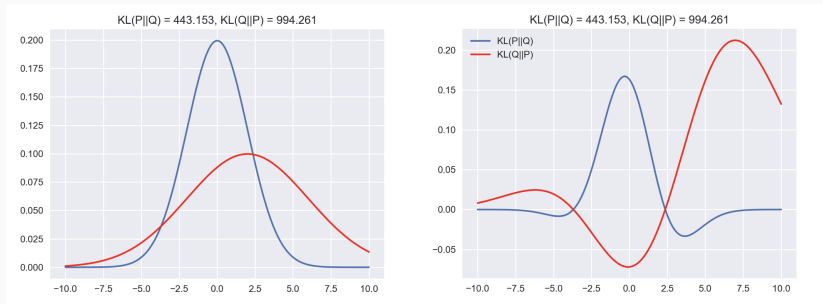
Divergencias Clásicas

Ejemplos clásicos de divergencias:

$$KL(p(y|x)||q_\theta(y|x)) = \mathbb{E}_{p(y|x)} (\log(p(y|x)/q_\theta(y|x))) \quad (11)$$

$$JS(p(y|x)||q_\theta(y|x)) = \frac{1}{2} KL(p(y|x)||m) + \frac{1}{2} KL(q_\theta(y|x)||m).$$

con $m = \frac{1}{2}p(y|x) + \frac{1}{2}q_\theta(y|x)$



Clasificadores Discriminativos vs. Generativos

La discusión anterior también sugiere que podemos construir un clasificador **probabilista** estimando $p(y|x)$ o bien estimando $p(y)$ y $p(x|y)$ por separado para luego obtener $p(y|x)$ vía regla de Bayes,

$$p(y|x) = \frac{\overbrace{p(x|y)}^{\text{likelihood}} \overbrace{p(y)}^{\text{prior}}}{\underbrace{p(x)}_{\text{evidence}}} \propto p(x|y)p(y).$$

Los clasificadores del primer tipo se denominan **discriminativos**.

Los clasificadores del segundo tipo se denominan **generativos**.

Clasificadores Discriminativos vs. Generativos

- **Clasificadores Discriminativos.** Aprenden directamente $p(y|x)$ o una transformación monótona de $p(y|x)$ que permite identificar la clase más probable *a-posteriori*.
- **Clasificadores Generativos.** Aprenden separadamente el *a-priori* $p(y)$ y la *verosimilitud* $p(x|y)$, para luego explotar la regla de Bayes.

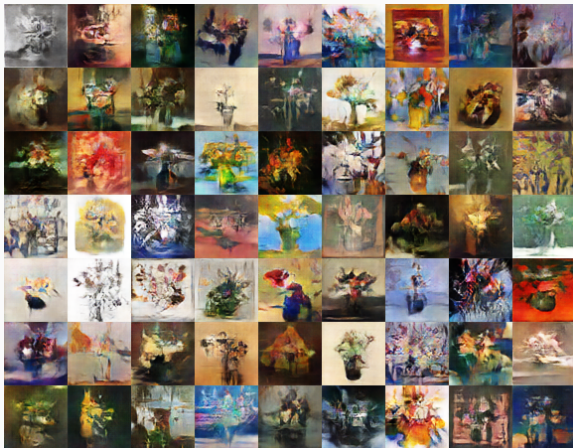
Cuando se tienen “suficientes datos” los clasificadores discriminativos tienden a obtener mejores resultados predictivos.

Los clasificadores generativos permiten introducir más fácilmente información *a-priori* sobre el problema estudiado.

Los clasificadores generativos pueden también ser usados para generar/simular data sintética vía el muestreo de $p(x|y)$.

Flores Sintéticas

Creadas por una red neuronal generativa entrenada con 100.000 imágenes de pinturas de variados artistas.



Otras Funciones de Costo

Todo lo que hemos dicho sobre la optimalidad del clasificador de Bayes debe reconsiderarse cuando el objetivo **no es minimizar el error de clasificación**.

Por ejemplo, en un problema de detección automática de riesgo de cancer, no da lo mismo un **falso positivo (FP)** que un **falso negativo (FN)**

Matriz de Confusión (Caso Binario):

	Pred +	Pred -
Actual +	True Positive	False Negative
Actual -	False Positive	True Negative

Otras Funciones de Costo

Cost-sensitive classification: En el caso de que tengamos una matriz de costos de la forma C_{kj} que representa el costo de predecir c_j cuando la clase verdadera es c_k ,

	Pred c_1	Pred c_2	...	Pred c_K
Actual c_1	C_{11}	C_{12}	...	C_{1K}
Actual c_2	C_{21}	C_{22}	...	C_{2K}
...		...		
Actual c_K	C_{K1}	C_{K2}	...	C_{KK}

la regla de clasificación óptima es:

$$f(x) = \arg \min_j C_j^* = \sum_k C_{kj} p(y = c_k | x).$$

Cost-Sensitive Classification

Ejemplo

Supongamos que el 40% de las paltas son orgánicas (1).

Supongamos que clasificar una palta como orgánica cuando no lo es nos cuesta 2 y que clasificar una palta como corriente cuando no lo es nos cuesta 1.

Si una palta pesa $x = 1.4$ ¿Cómo debemos clasificar la palta?

Cost-Sensitive Classification

Habíamos visto que

$$P(y = 1|x = 1.4) = 0.553$$

$$P(y = 0|x = 1.4) = 0.446$$

Asumiendo $C_{jj} = 0, \forall j$, tenemos

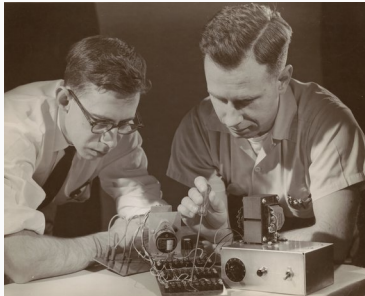
$$C_1^* = C_{01}P(y = 0|x = 1.4) + C_{11}P(y = 1|x = 1.4) = 2 \cdot 0.446 = 0.892$$

$$C_0^* = C_{00}P(y = 0|x = 1.4) + C_{10}P(y = 1|x = 1.4) = 1 \cdot 0.553 = 0.553$$

Conviene clasificarla como normal (0).

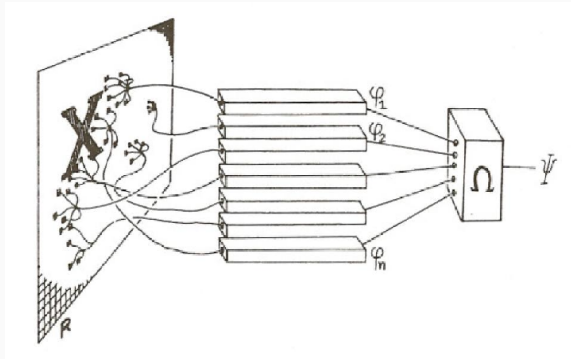
El Perceptrón

Un perceptrón ... “Inventado” por Frank Rosenblatt hacia 1957, el perceptrón es el primer algoritmo de aprendizaje automático del que tengamos noticias.



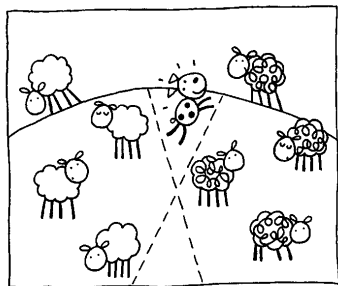
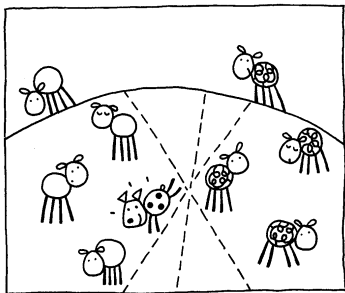
El Perceptrón

El método fue diseñado para aprender *parte* (última capa) de los parámetros de una red neuronal artificial construida (físicamente) para reconocer caracteres (imágenes de 20×20).



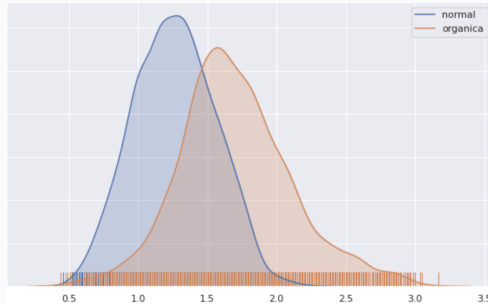
El Perceptrón

En el caso de dos clases, el método se puede ver como un método basado en un *hiperplano separador*.



Perceptrón como Método Discriminativo

Idea: en vez de aprender “toda la distribución” a posteriori $p(y|x)$, el perceptrón intenta aprender aquello esencial para tomar una decisión.



Perceptrón como Método Discriminativo

Justificación Formal: Es posible “imitar” el clasificador de Bayes sólo aproximando las fronteras de decisión. Para verlo, consideremos por simplicidad, un problema binario con clases $\mathbb{Y} = \{c_1, c_2\}$.

Log-Odds: Notemos que

$$\begin{aligned} p(y = c_1|x) > p(y = c_2|x) &\Leftrightarrow p(y = c_1|x) > 1 - p(y = c_1|x) \quad (12) \\ &\Leftrightarrow \frac{p(y = c_1|x)}{1 - p(y = c_1|x)} > 1 \\ &\Leftrightarrow o_1(x) = \log \left(\frac{p(y = c_1|x)}{1 - p(y = c_1|x)} \right) > 0 \end{aligned}$$

Perceptrón como Método Discriminativo

Conocer log-odds es suficiente para elegir entre c_1 y c_2 :

$$p(y = c_1|x) > p(y = c_2|x) \Leftrightarrow o_1(x) = \log \left(\frac{p(y = c_1|x)}{1 - p(y = c_1|x)} \right) > 0,$$

$$p(y = c_1|x) < p(y = c_2|x) \Leftrightarrow o_1(x) = \log \left(\frac{p(y = c_1|x)}{1 - p(y = c_1|x)} \right) < 0,$$

$$p(y = c_1|x) = p(y = c_2|x) \Leftrightarrow o_1(x) = \log \left(\frac{p(y = c_1|x)}{1 - p(y = c_1|x)} \right) = 0.$$

Idea: Aprender los log-odds! usando como espacio de hipótesis, el espacio de las funciones lineales:

$$f(x) = \sum_i w_i x_i + b. \quad (13)$$

De log-odds a decisiones:

$$p(y = c_1|x) \geq p(y = c_2|x) \Leftrightarrow o_1(x) = \log \left(\frac{p(y = c_1|x)}{1 - p(y = c_1|x)} \right) > 0.$$

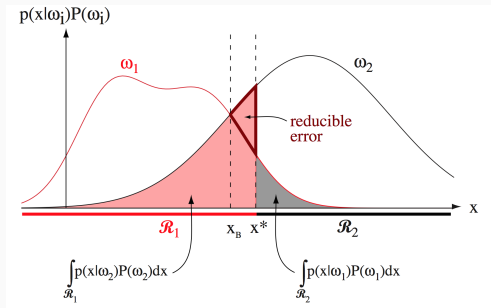
Idea: $f(x) \approx o_1(x)$.

\Rightarrow **Función de Decisión:** Si codificamos la clase c_1 como $+1$ y la clase c_2 como -1 , la regla de clasificación queda

$$\text{sign } f(x) = \text{sign} \left(\sum_i w_i x_i + b \right). \quad (14)$$

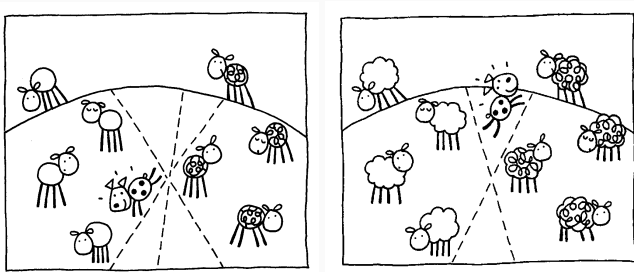
Fronteras de Clasificación

Si $\sigma_1(x) = 0 \Leftrightarrow p(c_1|x) = p(c_2|x)$, i.e., el conjunto de puntos $\{x : f(x) = 0\}$ representa una aproximación de la **frontera de decisión**: región en que cambia decisión de c_1 a c_2 (en el dibujo: ω_1 a ω_2).



Hiperplano Separador

El perceptrón busca aprender la frontera usando una función lineal i.e. trazando un hiperplano sobre el espacio de características. Si el hiperplano es consistente con todos los ejemplos, se denomina **hiperplano separador**.



Algoritmo de Entrenamiento de Rosenblatt

¿Cómo aprender tal función desde un conjunto de ejemplos?

Algorithm 1: El Perceptrón.

```
1  $w, b \leftarrow 0, 0$ 
2 do
3   mistakes  $\leftarrow$  false
4   for  $i = 1, \dots, n$  do
5     if  $y^{(i)} \text{sign}(w^T x^{(i)} + b) < 0^\dagger$  then
6        $w \leftarrow w + \eta y^{(i)} x^{(i)}$ 
7        $b \leftarrow b + \eta y^{(i)}$ 
8       mistakes  $\leftarrow$  true
9 while mistakes;
```

† Es importante notar que en este algoritmo, se define $\text{sign}(0) = 1$ para romper un empate entre ambas clases.

Algoritmo de Entrenamiento de Rosenblatt

Inicio. Los parámetros (w, b) se inicializan a valores convenientes.

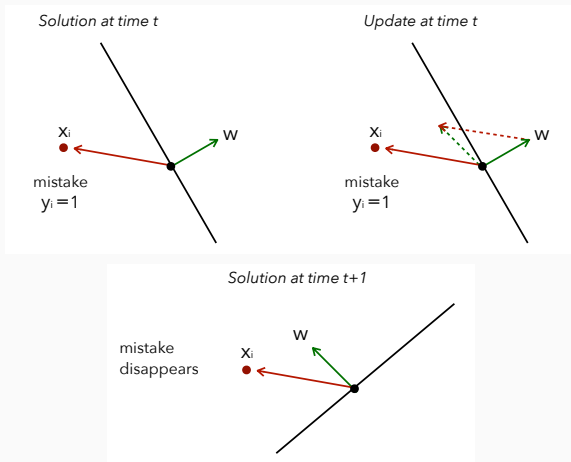
Loops. Luego, el algoritmo realiza una serie de iteraciones sobre los ejemplos. En cada iteración, se ajusta (w, b) *si y sólo si* $y^{(i)}\text{sign}(w^T x^{(i)} + b) < 0$, definiendo $\text{sign}(0) = 1$.

Cómo se ajustan los parámetros? Como la regla de clasificación es $\hat{y}^{(i)} = f(x) = \text{sign}(w^T x^{(i)} + b)$, el producto $y^{(i)}\text{sign}(w^T x^{(i)} + b) < 0$ *si y sólo si* la clase $\hat{y}^{(i)}$ predicha por modelo es incorrecta.

$y^{(i)}$	Signo de $(w^T x^{(i)} + b)$	
	+1	-1
+1	correct	mistake
-1	mistake	correct

Geometría del Perceptrón

Cómo se ajustan los parámetros? El algoritmo “mueve” (w, b) de manera que $y^{(i)}f(x^{(i)})$ sea menos negativo, es decir, el perceptrón trata de corregir el error.



Aprendizaje en el Perceptrón

En efecto, denotemos por $(w^{(t+1)}, b^{(t+1)})$ el estado de w, b después del t -ésimo update y por $(x^{(t)}, y^{(t)})$ el ejemplo usado en tal ajuste. Se tiene que

$$\begin{aligned}y^{(t)} w^{(t+1)T} x^{(t)} &= y^{(t)} \left(w^{(t)T} + \eta y^{(t)} x^{(t)} \right)^T x^{(t)} \\&= y^{(t)} w^{(t)T} x^{(t)} + \eta \|x^{(t)}\|^2 > y^{(t)} w^{(t)T} x^{(t)} \\y^{(t+1)} b^{(t)} &= y^{(t)} b^{(t)} + \eta y^{(t)} y^{(t)} > y^{(t)} b^{(t)}\end{aligned}$$

Por lo tanto, el ejemplo se clasifica mejor (o menos mal) en la siguiente iteración:

$$y^{(t)} f(x^{(t)}) > y^{(t-1)} f(x^{(t-1)})$$

No es muy difícil mostrar que este algoritmo converge a un hiperplano separador si el problema es *linealmente separable*.

Definición

En un problema de clasificación binario, un conjunto de ejemplos $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$ (con al menos un ejemplo de cada clase) se dice linealmente separable, si existen γ, w^*, b^* tal que $\forall i = 1, \dots, n$

$$y^{(i)} (w^{*T} x^{(i)} + b^*) > \gamma > 0. \quad (15)$$

Notemos que podemos perfectamente asumir $\|w^*\|^2 + b^{*2} = 1$, ya que en otro caso basta normalizar por una constante.

Notación. Como de costumbre, denotemos por $(w^{(t+1)}, b^{(t+1)})$ el estado de w, b después del t -ésimo update y por $(x^{(t)}, y^{(t)})$ el ejemplo usado en tal update. Además, para simplificar la notación, definamos $\underline{w} = (w, b)$ y $\underline{x} = (x, 1)$.

Cota Inferior a $\underline{w}^{*T} \underline{w}^{(t)}$. Observemos primero que:

$$\begin{aligned}\underline{w}^{*T} \underline{w}^{(t+1)} &= \underline{w}^{*T} \left(w^{(t)} + \eta y^{(t)} \underline{x}^{(t)} \right) \\ &= \underline{w}^{*T} w^{(t)} + \eta y^{(t)} \underline{w}^{*T} \underline{x}^{(t)} > \underline{w}^{*T} w^{(t)} + \eta \gamma.\end{aligned}$$

Como $\underline{w}^{(0)} = 0$, se sigue por inducción que $\underline{w}^{*T} \underline{w}^{(t)} > t\eta\gamma$.

Análisis de Convergencia/Término

Cota Inferior a $\|\underline{w}^{(t)}\|$. Como $\|\underline{w}^*\| = 1$ y $\underline{w}^{*T} \underline{w}^{(t)} < \|\underline{w}^*\| \|\underline{w}^{(t)}\|$,

$$\|\underline{w}^{(t)}\| > t\eta\gamma.$$

Cota Superior a $\|\underline{w}^{(t)}\|$. Notemos ahora que el perceptron realiza un update solo si $y^{(t)} \text{sign}(\underline{w}^{(t)T} \underline{x}^{(t)} + b) < 0$. Esto significa que

$$y^{(t)} \underline{w}^{(t)T} \underline{x}^{(t)} \leq 0.$$

Por lo tanto, si $R^2 = \max_i \|\underline{x}^{(i)}\|^2$, tenemos que

$$\begin{aligned}\|\underline{w}^{(t+1)}\|^2 &= \|\underline{w}^{(t)} + \eta y^{(t)} \underline{x}^{(t)}\|^2 \\ &= \|\underline{w}^{(t)}\|^2 + \eta^2 \|\underline{x}^{(t)}\|^2 + 2\eta y^{(t)} \underline{w}^{(t)T} \underline{x}^{(t)} \\ &\leq \|\underline{w}^{(t)}\|^2 + \eta^2 R^2.\end{aligned}$$

Como $\underline{w}^{(0)} = 0$, se sigue por inducción que $\|\underline{w}^{(t)}\|^2 \leq t\eta^2 R^2$.

Análisis de Convergencia/Término

Combinando las últimas dos cotas, obtenemos que

$$t^2 \eta^2 \gamma^2 < \|\underline{w}^{(t)}\|^2 \leq t \eta^2 R^2.$$

Conclusión I. Obtenemos entonces que el perceptron converge después de efectuar no más de

$$t < \frac{R^2}{\gamma^2}$$

updates y por lo tanto se detiene en un hiperplano separador!

Conclusión II. A partir de la primera y tercera cota, obtenemos también

$$\frac{\underline{w}^{*T} \underline{w}^{(t+1)}}{\|\underline{w}^*\| \|\underline{w}^{(t+1)}\|} > \frac{t \eta \gamma}{\sqrt{t \eta^2 R^2}} = \mathcal{O}(1/\sqrt{t})$$

Es decir, el perceptrón se acerca un hiperplano separador a tasa $\mathcal{O}(1/\sqrt{t})$ (updates).

Extensiones Multi-clases

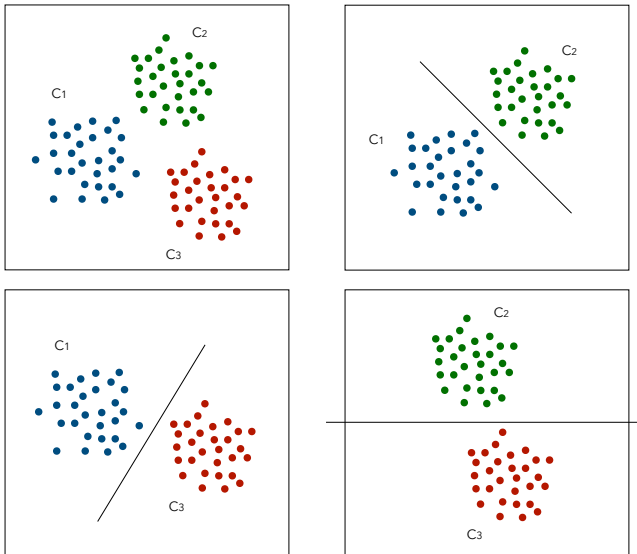
En general, hay dos formas de extender un **clasificador binario** al caso de múltiples clases:

- Descomponer el problema: se entrenan y combinan múltiples clasificadores binarios.
- El problema se resuelve directamente modificando el criterio de entrenamiento específico.

Tres métodos genéricos en la primera categoría:

- **One-versus-One (OVO):** Se entrenan $K(K - 1)/2$ clasificadores binarios para distinguir pares de clases.
- **One-versus-the-Rest (OVR):** Se entrenan K clasificadores binarios para distinguir una clase del resto.
- **Error correcting output codes (ECOC):** Se re-codifican las clases como cadenas binarias de largo M y luego se entrenan M clasificadores binarios para predecir cada bit.

One-versus-One (OVO)



One-versus-One (OVO)

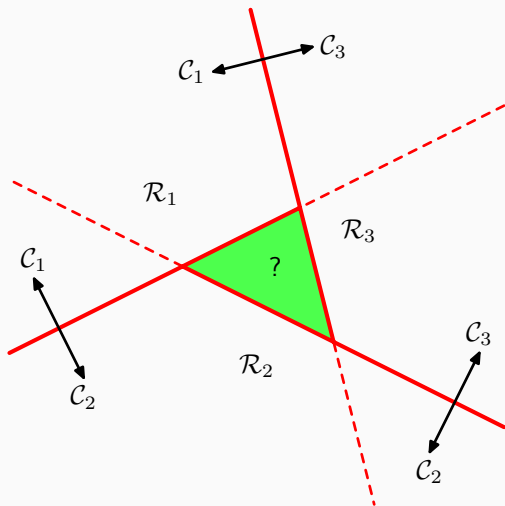
Para $i, j = 1, \dots, K$, $i < j$, se entrena un clasificador C_{ij} con los datos de la clase i re-etiquetados como positivos $+1$ y los datos de la clase j re-etiquetados como negativos.

Función de Decisión: Se realiza un “torneo” en el que cada C_{ij} “vota” por la clase i o la clase j . Finalmente se elige la clase con más votos.

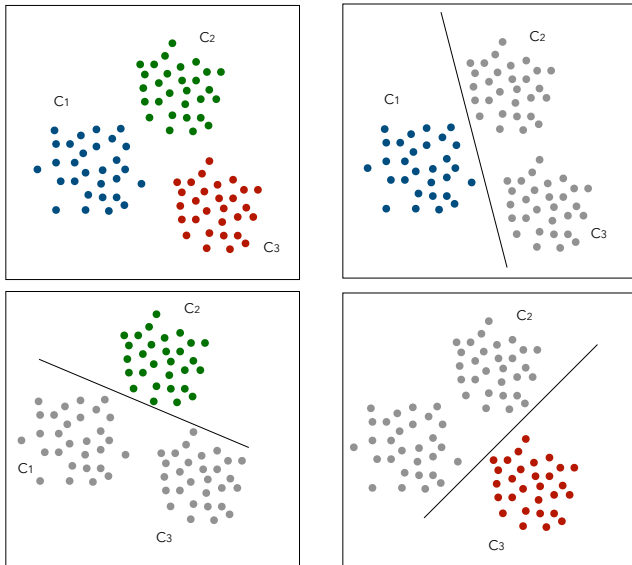
Problemas:

- Aunque los problemas son más pequeños, se debe entrenar un gran número de clasificadores.
- Pueden existir regiones del espacio característico sin clases dominantes.
- Pueden existir regiones del espacio con más de 1 clase dominante.

Regiones Ambiguas en OVO



One-versus-the-Rest (OVR)



One-versus-the-Rest (OVR)

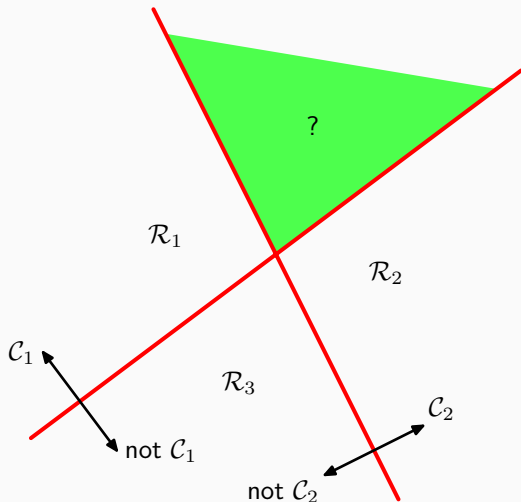
Para $i = 1, \dots, K - 1$, se entrena un clasificador C_i con los datos de la clase i re-etiquetados como positivos $+1$ y todos los demás datos re-etiquetados como negativos.

Función de Decisión: Se realiza un “torneo” en el que cada C_i “vota” por la clase i o no. Si ninguna clase $i = 1, \dots, K - 1$ obtiene votos, el dato se asigna a la clase K . Sino, se elige la clase con más votos.

Problemas:

- Intrínsecamente desbalanceado.
- Pueden existir regiones del espacio con más de 1 clase dominante.

Regiones Ambiguas en OVR



El problema de las regiones ambiguas se puede resolver entrenando K clasificadores y pidiendo a cada clasificador C_i que asigne una “confianza” $f_i(x)$ a su decisión (positiva para los datos asignados a la clase i).

Función de Decisión Modificada: Se asigna la clase k usando la siguiente regla de decisión

$$\arg \max_i f_i(x) \quad (16)$$

Se puede hacer lo mismo con OVO.

Error correcting output codes (ECOC)

Las etiquetas se re-codifican usando cadenas binarias de largo M

Clase k	Código c_k
1	0 0 1 ... 1
2	1 0 1 ... 0
 0
K	0 1 1 ... 1

Se entrena un clasificador C_i para predecir el bit i -ésimo.

Función de Decisión: Para clasificar un dato nuevo x , se construye una cadena binaria $c(x)$ usando el clasificador C_i para determinar el bit i -ésimo. Finalmente, se elige la clase k cuya cadena c_k sea la más cercana a $c(x)$ (Hamming).

El Regresor Logístico

Simplificación: Consideremos por simplicidad, un problema binario con clases $\mathbb{Y} = \{c_1, c_2\}$.

Al igual que el perceptrón, un regresor logístico es un método discriminativo que se enfoca en aproximar los log-odds

$$o(x) = \log \left(\frac{p(y = c_1|x)}{1 - p(y = c_1|x)} \right),$$

usando funciones lineales, es decir $o(x) \approx f(x)$ con

$$f(x) = \sum_i w_i x_i + b, \tag{17}$$

la función por aprender.

La diferencia está en la forma en que el regresor logístico se entrenará para lograr tal objetivo.

Esencialmente el regresor logístico adoptará un enfoque probabilista, **reconstruyendo $p(y|x)$ a partir del log-odd aprendido:**

$$f(x) \approx \log \left(\frac{p(y = c_1|x)}{1 - p(y = c_1|x)} \right) \Rightarrow \exp(f(x)) \approx \left(\frac{p(y = c_1|x)}{1 - p(y = c_1|x)} \right) \quad (18)$$

$$\Rightarrow \exp(f(x)) - \exp(f(x))p(y = c_1|x) \approx p(y = c_1|x)$$

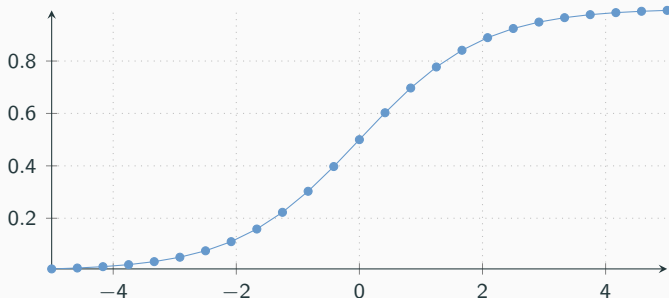
$$\Rightarrow p(y = c_1|x) \approx \frac{\exp(f(x))}{1 + \exp(f(x))} = \frac{1}{1 + \exp(-f(x))}.$$

Transformación Logística

La función

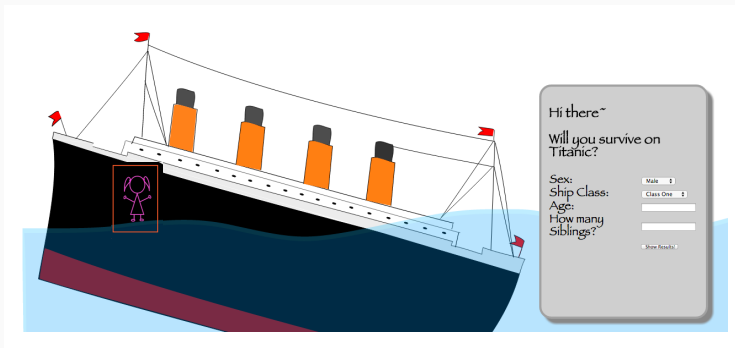
$$\sigma(\xi) = \frac{1}{1 + \exp(-\xi)}, \quad (19)$$

aplicada a $f(x)$ se denomina **función sigmoideal** o **función logística**.



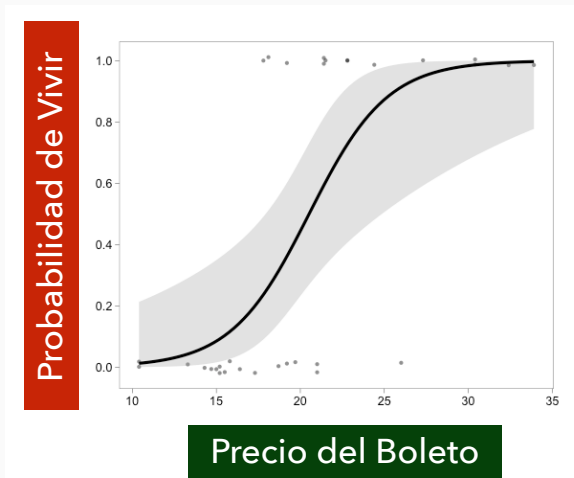
Interpretación (vía Modelo Lineal)

Ejemplo: Machine Learning from Disaster (Kaggle). Supongamos que x representa diferentes atributos acerca de un pasajero del TITANIC y que debemos “predecir” si sobrevivirá al naufragio o no.



Interpretación (vía Modelo Lineal)

Ejemplo: Machine Learning from Disaster (Kaggle). Supongamos que x representa diferentes atributos acerca de un pasajero del TITANIC y que debemos “predecir” si sobrevivirá al naufragio o no.



Entrenamiento del Clasificador

La ventaja del enfoque probabilístico es que nos permitirá expresar el entrenamiento como un problema de estimación clásico.

Codificación de las Clases: En vez de codificar las clases como $+1$ y -1 , nos convendrá ahora usar las etiquetas 1 (c_1) y 0 (c_2).

Función de Verosimilitud: Denotemos por $q_\theta(y|x)$ la aproximación del modelo a $p(y|x)$, es decir

$$q_\theta(y = 1|x) = \frac{1}{1 + \exp(-f(x; \theta))} \quad (20)$$

$$q_\theta(y = 0|x) = 1 - q_\theta(y = 1|x) = \frac{1}{1 + \exp(f(x; \theta))}.$$

Entonces, la función de verosimilitud correspondiente a los ejemplos $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$ viene dada por

$$\mathcal{L}(\theta) = \prod_{i=1}^n \left(q_\theta(y^{(i)} = 1|x^{(i)}) \right)^{y^{(i)}} \left(q_\theta(y^{(i)} = 0|x^{(i)}) \right)^{1-y^{(i)}}. \quad (21)$$

Tomando logaritmo, obtenemos

$$\ell(\theta) = \sum_{i=1}^n y^{(i)} \log \left(q_{\theta}(y^{(i)} = 1 | x^{(i)}) \right) + (1 - y^{(i)}) \log \left(q_{\theta}(y^{(i)} = 0 | x^{(i)}) \right) .$$

Los estimadores máximo verosímiles de los parámetros del clasificador se obtienen maximizando $\ell(\theta)$, o minimizando

$$E(\theta) = - \left(\sum_{i=1}^n y^{(i)} \log \left(q_{\theta}(y^{(i)} = 1 | x^{(i)}) \right) + (1 - y^{(i)}) \log \left(q_{\theta}(y^{(i)} = 0 | x^{(i)}) \right) \right) .$$

Codificación de las Clases: Si interpretamos las etiquetas $y^{(i)}$ como las verdaderas probabilidades $p(y|x)$, tenemos

$$\begin{aligned}\ell(\theta) = \sum_{i=1}^n & p(y^{(i)} = 1|x^{(i)}) \log \left(q_{\theta}(y^{(i)} = 1|x^{(i)}) \right) \\ & + p(y^{(i)} = 0|x^{(i)}) \log \left(q_{\theta}(y^{(i)} = 0|x^{(i)}) \right),\end{aligned}\quad (22)$$

es decir

$$\ell(\theta) = \sum_{i=1}^n \sum_{y^{(i)}} p(y^{(i)}|x^{(i)}) \log q_{\theta}(y^{(i)}|x^{(i)}) \quad (23)$$

es decir,

$$\ell(\theta) = \sum_{i=1}^n \mathbb{E}_{p(y^{(i)}|x^{(i)})} \log q_{\theta}(y^{(i)}|x^{(i)}). \quad (24)$$

Cross-entropy: La función objetivo

$$\ell(\theta) = \sum_{i=1}^n \sum_{y^{(i)}} p(y^{(i)}|x^{(i)}) \log q_{\theta}(y^{(i)}|x^{(i)}) = \sum_{i=1}^n \mathbb{E}_{p(y^{(i)}|x^{(i)})} \log q_{\theta}(y^{(i)}|x^{(i)}), \quad (25)$$

se puede escribir como una suma de “errores” sobre el conjunto de entrenamiento

$$\ell(\theta) = \sum_{i=1}^n L \left(p(y^{(i)}|x^{(i)}), q_{\theta}(y^{(i)}|x^{(i)}) \right), \quad (26)$$

con una función de costo $L(y, f(x))$ denominada *cross-entropy loss*.

Para propósitos de optimización en θ , da la mismo sumar una constante a la f.o. de modo que

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^n \mathbb{E}_{p(y^{(i)}|x^{(i)})} \log q_{\theta}(y^{(i)}|x^{(i)}) = \sum_{i=1}^n \mathbb{E}_{p(y^{(i)}|x^{(i)})} \log \frac{q_{\theta}(y^{(i)}|x^{(i)})}{p(y^{(i)}|x^{(i)})} + \text{cte.} \\ &= - \sum_{i=1}^n \mathbb{E}_{p(y^{(i)}|x^{(i)})} \log \frac{p(y^{(i)}|x^{(i)})}{q_{\theta}(y^{(i)}|x^{(i)})} \\ &= - \sum_{i=1}^n KL(p(y^{(i)}|x^{(i)})||q_{\theta}(y^{(i)}|x^{(i)})) \approx - \mathbb{E}_{x,y} KL(p(y|x)||q_{\theta}(y|x)).\end{aligned}$$

Conclusión: Entrenar el regresor logístico para maximizar la verosimilitud de los datos, es equivalente a minimizar la divergencia KL entre $p(y|x)$ y el modelo que implementa la máquina.

En la práctica, para entrenar el regresor logístico necesitamos encontrar el(un) máximo de:

$$\ell(\theta) = \sum_{i=1}^n y^{(i)} \log \left(q_{\theta}(y^{(i)} = 1 | x^{(i)}) \right) + (1 - y^{(i)}) \log \left(q_{\theta}(y^{(i)} = 0 | x^{(i)}) \right)$$

Como

$$q_{\theta}(y^{(i)} = 1 | x^{(i)}) = \frac{1}{1 + \exp(-\underline{w}^T \underline{x}^{(i)})},$$

con $\underline{w} = (w, b)$ y $\underline{x} = (x, 1)$. Obtenemos que

$$\ell(\theta) = \ell(\underline{w}) = \sum_{i=1}^n y^{(i)} \underline{w}^T \underline{x}^{(i)} - \log \left(1 + \exp(\underline{w}^T \underline{x}^{(i)}) \right), \quad (27)$$

Obtenemos que las primeras y segundas derivadas de

$$\ell(\underline{w}) = \sum_{i=1}^n y^{(i)} \underline{w}^T \underline{x}^{(i)} - \log \left(1 + \exp(\underline{w}^T \underline{x}^{(i)}) \right), \quad (28)$$

vienen dadas por

$$\frac{\partial \ell}{\partial \underline{w}} = \sum_{i=1}^n x^{(i)} (y^{(i)} - \hat{y}^{(i)}), \quad (29)$$

con $\hat{y}^{(i)} = q_{\theta}(y^{(i)} = 1 | x^{(i)})$.

Lamentablemente, la ecuación

$$\frac{\partial \ell}{\partial \underline{w}} = \sum_{i=1}^n x^{(i)}(y^{(i)} - \hat{y}^{(i)}) = 0, \quad (30)$$

no se puede resolver analíticamente. Sin embargo,

$$\frac{\partial^2 \ell}{\partial \underline{w} \partial \underline{w}^T} = - \sum_{i=1}^n x^{(i)} x^{(i)T} \hat{y}^{(i)} (1 - \hat{y}^{(i)}), \quad (31)$$

Matricialmente

$$\frac{\partial \ell}{\partial \underline{w}} = X^T(Y - \hat{Y}), \quad \frac{\partial^2 \ell}{\partial \underline{w} \partial \underline{w}^T} = -X^T D X. \quad (32)$$

con D una matriz diagonal, con entradas $D_{ii} = \hat{y}^{(i)}(1 - \hat{y}^{(i)})$ y X es una matriz que contiene los x de entrenamiento en las filas.

Entrenamiento vía Gradiente

Un método ampliamente empleado en ML para optimizar es *gradiente descendente*. Dado un criterio de entrenamiento de la forma

$$\min_{\underline{\beta} \in \mathbb{R}^p} J(\underline{\beta}), \quad (33)$$

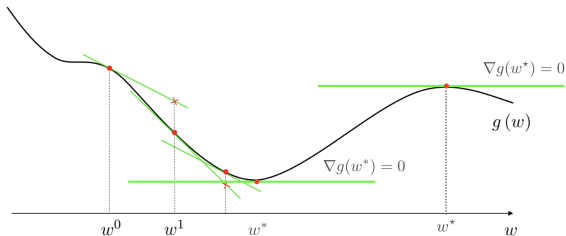
con $J(\underline{\beta})$ diferenciable, este método genera una sucesión de soluciones $\underline{\beta}^{(0)}, \underline{\beta}^{(1)}, \dots, \underline{\beta}^{(t)}$ que se obtienen iterando la siguiente regla

$$\underline{\beta}^{(t+1)} = \underline{\beta}^{(t)} - \eta_t \left. \frac{\partial J}{\partial \underline{\beta}} \right|_{\underline{\beta} = \underline{\beta}^{(t)}}, \quad (34)$$

donde $\eta_t \in \mathbb{R}$ se denomina *la tasa de aprendizaje*. Con frecuencia se usa un valor fijo $\eta_t = \eta \in (0, 1)$.

Entrenamiento vía Gradiente

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta_t \nabla J(\Theta^{(t)})$$



Para el clasificador estudiado, el algoritmo (vectorizado) toma la forma

Algorithm 2: Gradiente Descendente (o Ascendente) para el Regresor Logístico.

```
1  $\underline{w} \leftarrow 0$   
2 do  
3    $\hat{Y} \leftarrow \frac{1}{1 + \exp(-X\underline{w})}$   
4    $\underline{w} \leftarrow \underline{w} + \eta_t X^T (Y - \hat{Y})$   
5 while not convergence;
```

- Una iteración del algoritmo anterior es lineal en el número de atributos (p) y en el número de ejemplos.
- Un algoritmo de segundo orden es lineal en el número de ejemplos y prácticamente cúbica en el número de atributos.
- En términos de memoria, un método clásico de segundo orden necesita almacenar la matriz $X^T X$ y por lo tanto es cuadrático en p , mientras que el algoritmo anterior es lineal en p .

Una variante aún más flexible del método del gradiente, consiste en usar lo que se denomina *gradiente descendente estocástico*.

Supongamos que en cada iteración t , construimos una aproximación \tilde{G}_t del verdadero gradiente $G_t = \partial J / \partial \underline{\beta} \big|_{\underline{\beta}^{(t)}}$ tal que

$$\mathbb{E}(\tilde{G}_t) = G_t \quad (35)$$

Bajo condiciones razonables, es posible demostrar que la regla

$$\underline{\beta}^{(t+1)} = \underline{\beta}^{(t)} - \eta_t \tilde{G}_t, \quad (36)$$

converge (en un sentido estocástico) al óptimo de la función objetivo.

En nuestro problema, debiésemos aproximar el gradiente

$$G_t = \frac{\partial J}{\partial \underline{\beta}} = \sum_{i=1}^n e^{(i)} \underline{x}^{(i)} \quad (37)$$

Una forma sencilla de hacerlo es elegir B ejemplos de entrenamiento al azar $\{(x^{(i^*)}, y^{(i^*)})\}_{i^*=1}^B$ y estimar el gradiente como

$$\tilde{G}_t = \frac{n}{B} \sum_{i^*=1}^B (y^{(i^*)} - \hat{y}^{(i^*)}) x^{(i^*)} \quad (38)$$

No es difícil demostrar que:

$$\mathbb{E}(\tilde{G}_t) = G_t. \quad (39)$$

Gradiente Estocástico versus Solución Clásica

- Una iteración del algoritmo anterior es lineal en el número de atributos e independiente del número de ejemplos.
- Entrenar via gradiente estocástico puede ser muy ventajoso en problemas con muchísimos datos y muchos atributos.

Como \underline{w} es un estimador MLE, se obtiene inmediatamente que si los log-odds son efectivamente lineales ($o = X\underline{w}^*$ para algún \underline{w}^*).

$$\lim_{n \rightarrow \infty} \underline{w} = \underline{w}^*. \quad (40)$$

Además,

$$\underline{w} \xrightarrow[n \rightarrow \infty]{D} \mathcal{N}(\underline{w}^*, X^T D X^{-1}). \quad (41)$$

lo que nos permite hacer inferencias sobre los coeficientes obtenidos (intervalos de confianza o contrastes).

Múltiples Clases

Para manejar el caso multi-class $\mathbb{Y} = \{c_1, c_2, \dots, c_K\}$, el clasificador logístico modela múltiples log-odds. Tomando una clase como referencia, digamos la K -ésima, se modelan $K - 1$ log-odds

$$o_1(x) = \log \left(\frac{p(y = c_1|x)}{p(y = c_K|x)} \right) \quad (42)$$

$$o_2(x) = \log \left(\frac{p(y = c_2|x)}{p(y = c_K|x)} \right)$$

...

$$o_{K-1}(x) = \log \left(\frac{p(y = c_{K-1}|x)}{p(y = c_K|x)} \right)$$

usando funciones lineales, es decir $o_j(x) \approx f_j(x)$ con

$$f_j(x) = \sum_i w_{ji}x_i + b_j. \quad (43)$$

Es posible invertir las ecuaciones anteriores para obtener una estimación de $p(y = c_j|x)$ para cada clase ...

$$o_1(x) = \log \left(\frac{p(y = c_1|x)}{p(y = c_K|x)} \right)$$

$$o_2(x) = \log \left(\frac{p(y = c_2|x)}{p(y = c_K|x)} \right)$$

...

$$o_{K-1}(x) = \log \left(\frac{p(y = c_{K-1}|x)}{p(y = c_K|x)} \right)$$

$$\exp o_1(x) = \frac{p(y = c_1|x)}{p(y = c_K|x)}$$

$$\exp o_2(x) = \frac{p(y = c_2|x)}{p(y = c_K|x)}$$

...

$$\exp o_{K-1}(x) = \frac{p(y = c_{K-1}|x)}{p(y = c_K|x)}$$

$$p(y = c_K | x) \sum_{j=1}^{K-1} \exp o_j(x) = \sum_{j=1}^{K-1} p(y = c_j | x)$$

Y obtenemos así que

$$p(y = c_K | x) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp o_j(x)} \quad (44)$$

$$p(y = c_k | x) = \frac{\exp o_k(x)}{1 + \sum_{j=1}^{K-1} \exp o_j(x)} \quad \forall k \neq K \quad (45)$$

El clasificador logístico aproxima los log-odds mediante funciones lineales $f_j(x) = \sum_i w_{ji}x_i + b_j$ para obtener aproximaciones de las probabilidades anteriores.

$$p(y = c_K|x) \approx q_\theta(c_K|x) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp f_j(x)} \quad (46)$$

$$p(y = c_k|x) \approx q_\theta(c_k|x) \frac{\exp f_k(x)}{1 + \sum_{j=1}^{K-1} \exp f_j(x)} \quad \forall k \neq K \quad (47)$$

Como tanto en el caso binario como multi-class tenemos un modelo explícito de $p(y = c_k|x)$ la decisión se toma, “imitando” el clasificador de Bayes con nuestras estimaciones,

$$f^*(x) = \arg \max_j q_\theta(c_j|x). \quad (48)$$

1. Teoría de Decisión
2. El Perceptrón
3. El Regresor Logístico