Clasificadores Discriminativos Básicos

Aprendizaje Automático INF-393 II-2018

Ricardo Ñanculef

UTFSM Campus San Joaquín

Table of contents

Clasificación en el Modelo Supervisado

Definición

Dado un espacio de entrada $\mathbb{X}$, un conjunto finito de categorías (clases, o etiquetas) $\mathbb{Y} = \{c_1, c_2, \ldots, c_K\}$, 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 $f : \mathbb{X} \to \mathbb{Y}$ que minimice la probabilidad de asignar incorrectamente una categoría a un patrón de entrada $x \in \mathbb{X}$, es decir que minimice:

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

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

Clasificación como Aprendizaje Supervisado

Con la definición anterior, un problema de clasificación es un caso especial de aprendizaje supervisado donde $\mathbb{Y}$ es finito y la función de pérdida es la denominada misclassification loss:

$$L(f(x), y) = I(f(x) \neq y) \ , \tag{2}$$

con $I(\cdot)$ 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 obtenemos que:

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

Entrenamiento de un Clasificador

Recordemos los componentes esenciales de un método de aprendizaje:

- Una representación (típicamente vectorial) de los datos.
- Un espacio de hipótesis.
- Un criterio de entrenamiento (función de error o mal desempeño).
- Un algoritmo para minimizar el error i.e. mejorar el desempeño a partir de los ejemplos (experiencia).

Criterio de entrenamiento simple: fracción de datos mal clasificados en el conjunto de entrenamiento, es decir,

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

Posible overfitting! Depende de la complejidad del espacio de hipótesis relativa al número de ejemplos disponibles.

Clasificador Óptimo

Tal como hicimos para el modelo de regresión, es útil preguntarse cuál es óptimo de este criterio de aprendizaje si ignoramos las limitaciones propias de un conjunto finito de ejemplos, un espacio de hipótesis reducido y un algoritmo de optimización imperfecto.

En teoría de clasificación / reconocimiento de patrones, el resultado se denominará Regla Óptima de Decisión ó Clasificador Bayesiano y nos dará una cota inferior para el error de clasificación (medio) que podemos obtener sobre (infinitos) futuros casos,

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

Clasificador Óptimo

Notemos que, definiendo $B(x) = \sum_y I(f(x) \neq y)p(y|x)$, obtenemos

$$\begin{aligned} \mathbb{E}(L(f(x), y)) &= \sum\nolimits_{x,y} I(f(x) \neq y)p(x, y) \\ &= \sum\nolimits_x \sum\nolimits_y I(f(x) \neq y)p(y|x)p(x) \\ &= \sum\nolimits_x B(x)p(x)\,. \end{aligned} \tag{6}$$

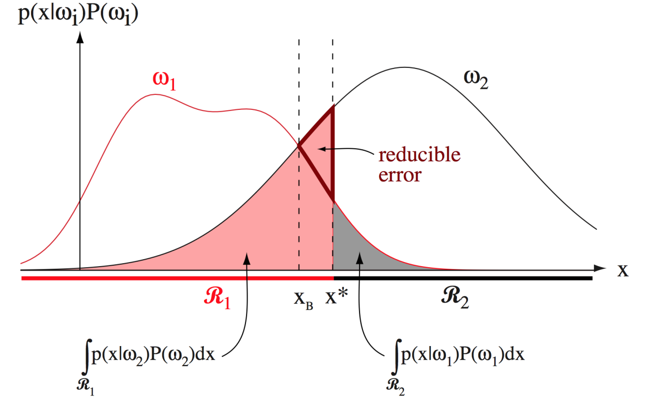
Claramente, la "suma" anterior se puede minimizar si $\forall x$ elegimos el valor mínimo de $B(x)$. Notemos ahora que como $f(x)$ puede asignar sólo 1 categoría a x,

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

de modo que el mínimo de $B(x)$ se logra implementando la regla

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

Error de Bayes



En este dibujo las clases son ω_1 y ω_2.

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 ambas probabilidades viene dada por:

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

En esta importante relación, el término $p(x|y)$ se denomina *la verosimilitud* y $p(x)$ *la evidencia*.

Decisión A-Priori vs. Decisión A-Posteriori

Ejemplo

Consideremos el problema de clasificar correo SPAM y supongamos que $x \in \{0,1\}$ representa la presencia o ausencia de una palabra en el cuerpo del correo. Supongamos que debemos decidir si un correo entrante que contiene la palabra ($x = 1$) es SPAM o no-SPAM (HAM).

La elección *a priori* corresponde a elegir la categoría que aparece con mayor frecuencia. La elección *a posteriori* corresponde en cambio a elegir la clase más frecuente de entre los correos que sí contienen la palabra.

$$f^*(x) = \begin{cases} \text{SPAM} & \textit{Si } p(\text{SPAM}|x=1) \geq 0.5 \\ \text{HAM} & \textit{Si } p(\text{HAM}|x=1) > 0.5 \end{cases}$$

Clasificadores Probabilistas vs. No-Probabilistas

La discusión anterior nos lleva a concluir que un modo de construir un clasificador consiste en aproximar $p(y|x)$. Esto sugiere que, en vez de utilizar el criterio de entrenamiento básico

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

podríamos utilizar un **criterio de entrenamiento "variacional"**, es decir minimizar una medida de distancia D entre $p(y|x)$ y un modelo explícito para $p(y|x)$, digamos $q_\theta(y|x)$, implementado por el learner

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

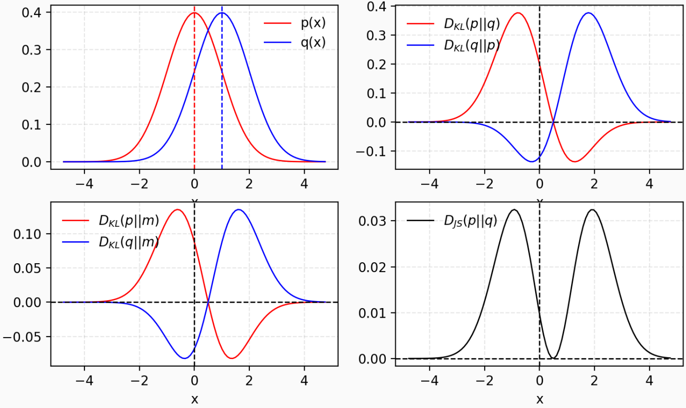
Ejemplos clásicos de divergencias:

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

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

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

Divergencias Clásicas



Clasificadores Discriminativos vs. Generativos

Recordemos que

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

A partir de esta observación aparecen dos tipos de clasificadores:

- **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.

Si bien los clasificadores discriminativos tienden a tener mejores resultados predictivos, los clasificadores generativos pueden 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 revisarse si lo que interesa no es minimizar el error de clasificación.

Por ejemplo, en un problema de detección automática de riesgo de cancer, ¿Es simétrica la función de costo?

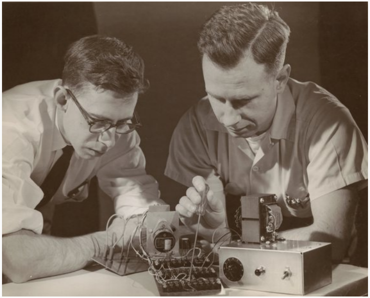
Cost-sensitive learning: En el caso de que tengamos una matriz de costos de la forma C_{jk} que representa el costo de predecir c_j cuando la clase es c_k, la regla de clasificación óptima es:

$$f(x) = \arg\min_j \sum_k p(y = c_j|x)\, C_{jk} \, .$$

El Perceptrón

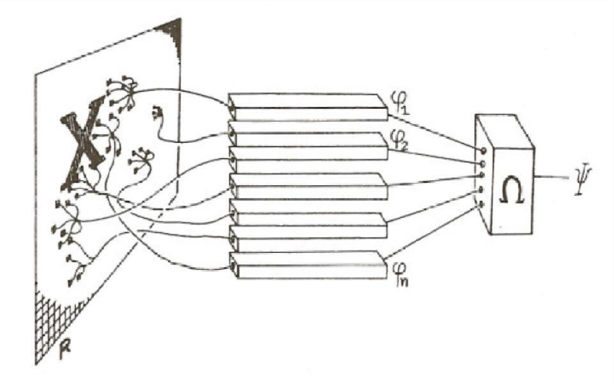
Érase una vez ...

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).



Perceptrón como Método Discriminativo

Hoy, podemos estudiar el perceptrón como un clásico algoritmo discriminativo. De hecho, un poco más audaz. En vez de aprender "toda la distribución" a posteriori $p(y|x)$, el perceptrón intenta aprender aquello esencial para tomar una decisión.

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

Log-Odds: Notemos que

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

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:

$$f(x) = \sum_i w_i x_i + b\,. \tag{13}$$

Perceptrón como Método Discriminativo

De log-odds a decisiones:

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

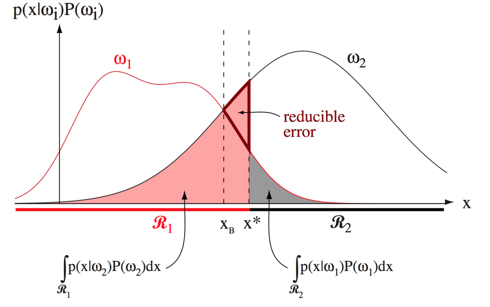
Idea: $f(x) \approx o_1(x)$.

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

$$\operatorname{sign} f(x) = \operatorname{sign}\left(\sum_i w_i x_i + b\right)\,. \tag{14}$$

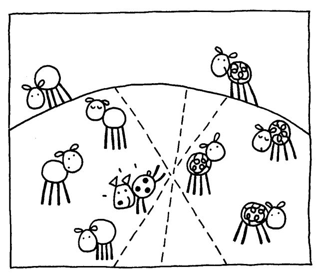
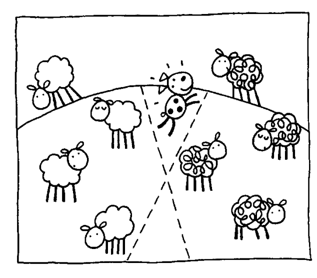
Fronteras de Clasificación

Recordemos que $o_1(x) = 0 \iff p(c_1|x) = p(c_2|x)$. Entonces, el conjunto de puntos $\{x : f(x) = 0\}$ representa una aproximación de la frontera de decisión: la zona del espacio de entrada en que el clasificador óptimo 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 un 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.

$w, b \leftarrow 0, 0$
do
    mistakes $\leftarrow$ `false`
    for $i = 1, \ldots, n$ **do**
        if $y^{(i)} sign\left(w^T x^{(i)} + b\right) < 0^{\dagger}$ **then**
            $w \leftarrow w + \eta y^{(i)} x^{(i)}$
            $b \leftarrow b + \eta y^{(i)}$
            mistakes $\leftarrow$ `true`
while *mistakes*;

† Es importante notar que en este algoritmo, se define 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.

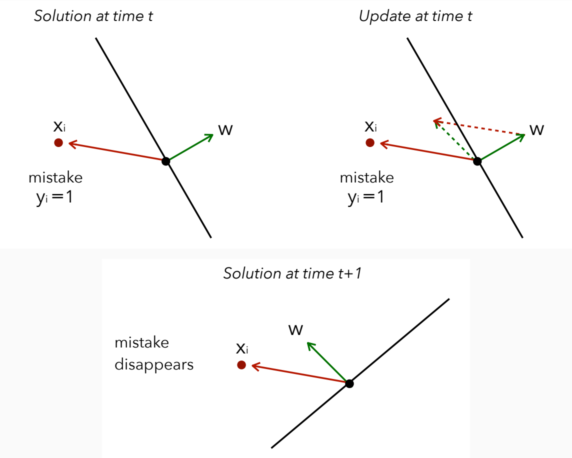
Loops. Luego, el algoritmo realiza una serie de iteraciones sobre los ejemplos. En cada iteración, se ajusta (w, b) *si y sólo si* $y^{(i)}\text{sign}(w^T x^{(i)} + b) < 0$, definiendo $\text{sign}(0) = 1$.

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.

	Signo de $(w^T x^{(i)} + b)$	
$y^{(i)}$	$+1$	-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 $\left(w^{(t+1)}, b^{(t+1)}\right)$ 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

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

$$y^{(t)} b^{(t)} = y^{(t)} b^{(t)} + \eta y^{(t)} y^{(t)} > y^{(t)} b^{(t)}$$

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

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

Separabilidad Lineal & Convergencia

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 $\forall i = 1, \ldots, n$

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

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

Análisis de Convergencia/Término

Notación. Como de costumbre, denotemos por $\left(w^{(t+1)}, b^{(t+1)}\right)$ 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 $\underline{w}^{*T}\underline{w}^{(t+1)}$. 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 $\underline{w}^{*T}\underline{w}^{(t+1)} > t\eta\gamma$.

Análisis de Convergencia/Término

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

$$\|\underline{w}^{(t+1)}\| > t\eta\gamma\,.$$

Cota Superior a $\|\underline{w}^{(t+1)}\|$. Notemos ahora que el perceptron realiza un update solo si $y^{(t)}\text{sign}\left(w^{(t)T}x^{(t)} + b\right) < 0$. 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 $\|\underline{w}^{(t+1)}\|^2 \leq t\eta^2 R^2$.

Análisis de Convergencia/Término

Combinando las últimas dos cotas, obtenemos que

$$t^2\eta^2\gamma^2 \ < \ \|\underline{w}^{(t+1)}\|^2 \ \leq \ t\eta^2 R^2 \, .$$

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).

El Regresor Logístico

Log-odds Lineales

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

$$f(x) = \sum_i w_i x_i + b\,, \tag{16}$$

la función por aprender.

Modelo Probabilístico

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**:

$$f(x) \approx= \log\left(\frac{p(y=c_1|x)}{1-p(y=c_1|x)}\right) \Rightarrow \exp(f(x)) \approx (\frac{p(y=c_1|x)}{1-p(y=c_1|x)} \tag{17}$$

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

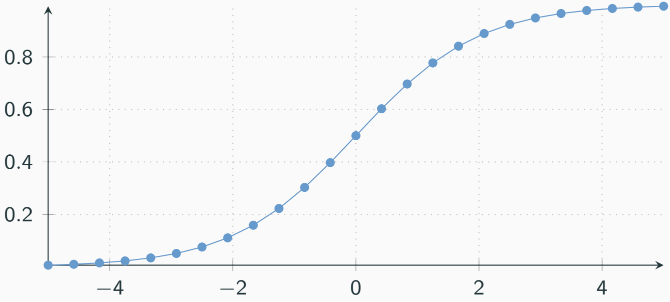
$$\Rightarrow p(y=c_1|x) \approx \frac{\exp(f(x))}{1+\exp(f(x))} = \frac{1}{1+\exp(-f(x))}\,.$$

Transformación Logística

La función

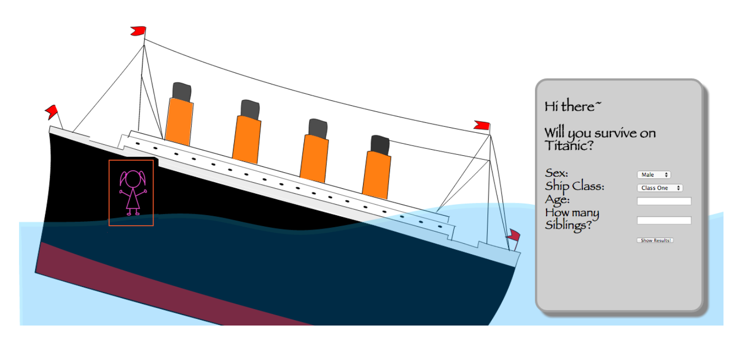
$$\sigma(\xi) = \frac{1}{1 + \exp(-\xi)}\,, \tag{18}$$

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



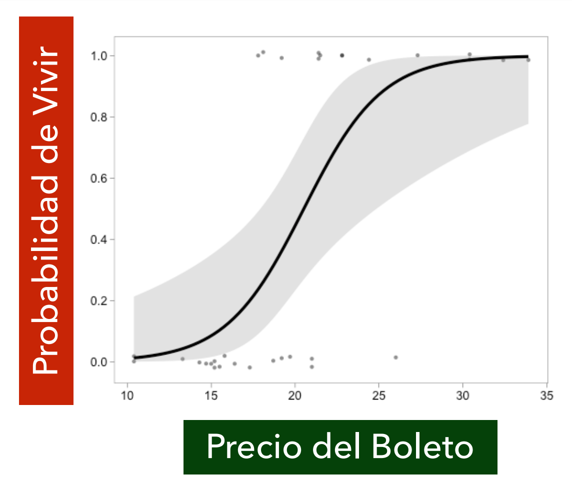
Interpretación (vía Modelo Lineal)

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

$$q_\theta(y=1|x) = \frac{1}{1+\exp(-f(x;\theta))} \tag{19}$$

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

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

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} \left(q_\theta(y^{(i)}=1|x^{(i)})\right)^{y^{(i)}} \left(q_\theta(y^{(i)}=0|x^{(i)})\right)^{1-y^{(i)}}\,. \tag{20}$$

Entrenamiento del Clasificador

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

$$\begin{aligned}\ell(\theta) = \sum_{i=1}^{n} & p(y^{(i)} = 1|x^{(i)}) \log \left(q_\theta(y^{(i)} = 1|x^{(i)}) \right) \\ & + p(y^{(i)} = 0|x^{(i)}) \log \left(q_\theta(y^{(i)} = 0|x^{(i)}) \right) ,\end{aligned} \tag{21}$$

es decir

$$\ell(\theta) = \sum_{i=1}^{n} \sum_{y^{(i)}} p(y^{(i)}|x^{(i)}) \log q_\theta(y^{(i)}|x^{(i)}) \tag{22}$$

es decir,

$$\ell(\theta) = \sum_{i=1}^{n} \mathbb{E}_{p(y^{(i)}|x^{(i)})} \log \frac{q_\theta(y^{(i)}|x^{(i)})}{p(y^{(i)}|x^{(i)})} . \tag{23}$$

Aprendizaje vía Divergencias

Para propósitos de optimización en θ,

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

$$= -\sum_{i=1}^{n} KL(p(y^{(i)}|x^{(i)})||q_\theta(y^{(i)}|x^{(i)})) \approx -\mathbb{E}_{x,y} KL(p(y|x)||q_\theta(y|x)) .$$

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 para $p(y|x)$.

Ecuaciones Normales

Para entrenar en la práctica el regresor logístico necesitamos encontrar el(un) máximo de:

$$
\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{n} y^{(i)} \log \left(q_\theta(y^{(i)} = 1 | x^{(i)}) \right) + (1 - y^{(i)}) \log \left(q_\theta(y^{(i)} = 0 | x^{(i)}) \right) \\
\ell(\underline{w}) &= \sum_{i=1}^{n} y^{(i)} \underline{w}^T \underline{x}^{(i)} - \log \left(1 + \exp(\underline{w}^T \underline{x}^{(i)}) \right) \ , \qquad (24)
\end{aligned}
$$

con $\underline{w} = (w, b)$ y $\underline{x} = (x, 1)$. Las primeras y segundas derivadas de esta f.o. vienen dadas por

$$
\frac{\partial \ell}{\partial \underline{w}} = \sum_{i=1}^{n} x^{(i)} (y^{(i)} - \hat{y}^{(i)}) \, , \qquad (25)
$$

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

Ecuaciones Normales

Lamentablemente, la ecuación

$$\frac{\partial \ell}{\partial \underline{w}} = \sum_{i=1}^{n} x^{(i)}(y^{(i)} - \hat{y}^{(i)}), \tag{26}$$

no se puede resolver analíticamente. Sin embargo,

$$\frac{\partial^2 \ell}{\partial \underline{w} \partial \underline{w}^T} = -\sum_{i=1}^{n} x^{(i)} x^{(i)T} \, \hat{y}^{(i)}(1 - \hat{y}^{(i)}), \tag{27}$$

Matricialmente

$$\frac{\partial \ell}{\partial \underline{w}} = X^T(Y - \hat{Y}), \quad \frac{\partial^2 \ell}{\partial \underline{w} \partial \underline{w}^T} = -X^T D X. \tag{28}$$

con D una matriz diagonal, con entradas $D_{ii} = \hat{y}^{(i)}(1 - \hat{y}^{(i)})$ y X es la misma matriz definida en el capítulo de regresión lineal.

Aprendizaje vía Gradiente

Iterar el siguiente algoritmo (vectorizado)

Algorithm 2: Gradiente Ascendente con Tasa de Aprendizaje Dinámica.

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

Resultados Asintóticos

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

$$\lim_{n\to\infty} \underline{w} = \underline{w}^* \,. \tag{29}$$

Además,

$$\underline{w} \xrightarrow[n\to\infty]{D} \mathcal{N}(\underline{w}^*, X^T D X^{-1})\,. \tag{30}$$

los que nos permite manipular los coeficientes obtenidos de manera análoga a como hacíamos en el caso de regresión lineal.

Qué hemos visto

1. Clasificación en el Modelo Supervisado
2. El Perceptrón
3. El Regresor Logístico