Detección de barcos vía clasificación

Javier Castro M.

Pregunta 1

La clasificación multiclase propone clasificar según la regla $x \to argmax\left\{l_k(x) = a_k^T x + b_k | k = 1, ..., N\right\}$. El máximo sobre el conjunto finito de índices permite que todo x sea clasificado. No obstante, dado que el máximo puede no ser único, la definición dada en el enunciado permite que hayan x asignados a más de una clase. Para que esto no pase, cada vez que se de este caso se asignará simplemente al mayor índice. Si $N = 2$, según lo expresado, $x \in \mathcal{C}_1$ ssi $a_1^T x + b_1 > a_2^T x + b_2$ y $x \in \mathcal{C}_2$ ssi $a_1^T x + b_1 \leq a_2^T x + b_2$. Restando en las desigualdades obtenemos que $x \in \mathcal{C}_1$ ssi $(a_1 - a_2)^T x + (b_1 - b_2) > 0$ que concuerda con lo visto para dos clases. Ahora, si tenemos más clases la región de decisión y las clases quedan definidas por

$$R = \bigcup_{i,j} \left\{x \in \mathbb{R}^M | l_i(x) = l_j(x)\right\}$$
$$\mathcal{C}_i = \left\{x \in \mathbb{R}^M | l_j(x) \leq l_i(x) \forall j \leq i, l_j(x) < l_i(x) \forall j > i\right\}$$

Pregunta 2

Se cuenta con un set de datos $D = (I_i, y_i)_{i=1}^N$ donde I_i representa una imagen e y_i indica si la imagen representa ($y_i = 1$) o no ($y_i = 0$) a un barco. Las imágenes I_i se representan por un vector 3-dimensional del cuál se obtienen su vector de características x_i usando un método que busca resaltar los bordes en las imágenes. Trabajaremos con los vectores características, con esto nuestros datos son de la forma $(x, y) = (x_i, y_i)_{i=1}^N$. Definimos entonces las clases en cuestión, $\mathcal{C}_1$ representa las imágenes que muestran un barco y $\mathcal{C}_0$ las que no. Para la parte **b)** se busca atacar el problema desde el punto de vista de la **regresión logística** (RL), este plantea encontrar $p(\mathcal{C}_1|w, x_i)$ buscando el peso w que maximice la log-verosimilitud (1).

$$l(w) = \sum_{i=1}^{N} y_i ln(\sigma(w^T x_i)) + (1 - y_i) ln(1 - \sigma(w^T x_i)) \quad (1)$$

$$\nabla l(w) = \sum_{i=1}^{N} (y_i - \sigma_i) x_i \quad (2)$$

Procedemos a maximizar la función $l(w)$ calculada con datos de entrenamiento para encontrar el peso w óptimo. Dada la forma de la función objetivo en (1), se usa el método de gradiente estocástico (SGD), es cuál tiene como regla de actualización

$$w_{n+1} = w_n + \eta(y_i - \sigma_i) x_i$$

lo cual se repite recorriendo todos los datos $i \in \{1, ..., N\}$ de forma aleatoria. A cada una de estas pasadas por los datos se le denomina **época**. En la figura 1 se muestran los resultados de aplicar este algoritmo de optimización.

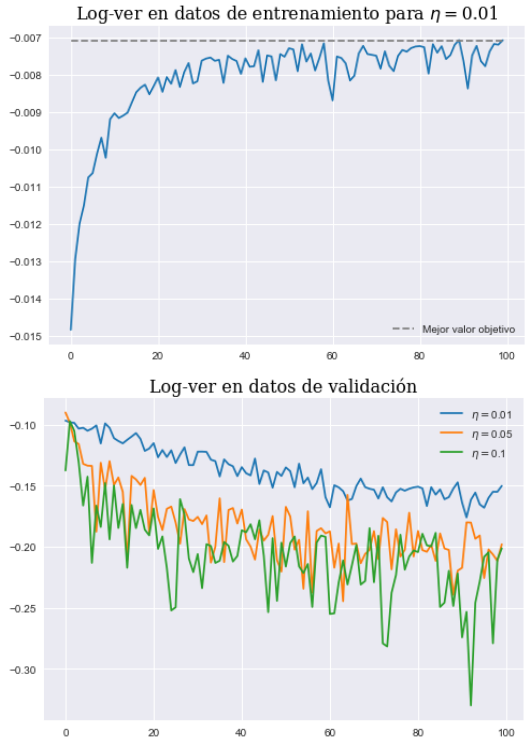

Figure 1: Log-verosimilitud normalizada por la cantidad de datos vs cantidad de épocas. Además, se agregó una recta que representa el valor objetivo alcanzado en el último paso de *GSD*

En la figura 1 vemos que para los datos de entrenamiento el comportamiento es el esperado, mientras más épocas se pasa optimizando más aumenta la *log-verosimilitud* hasta el punto en que alcanza cierta estabilidad, lo cual pasa al rededor $e \approx 30$ para $\eta = 0.01$ y $e \approx 10$ para $\eta \in \{0.05, 0.1\}$. Otro punto importante a resaltar es lo "*estocástico*" de las curvas, es decir, el ruido que se puede observar lo que es propio del método *GSD* para optimización. En cuanto a esto, se observa que mientras más grande es η más ruido tendrán las curvas, esto se explica pues η representa el tamaño del paso en la dirección (espacial) que se escoge avanzar en el algoritmo, por lo tanto un η pequeño resultará en avances más "cortos" hacia el óptimo pero de forma más estable. Mientras que un η más grande

nos entregará una convergencia más rápida pero a su vez más caótica.

Dado que el valor objetivo de la optimización de (1) se estabiliza para $e \approx 30$ en los tres valores de η considerados, se tomará el valor de la *log-verosimilitud* en el set de validación y la estabilidad de la curva como punto de comparación. Esto pues, buscamos siempre maximizar (1) ya sea tomando datos de entrenamiento o validación. En cuanto a la cantidad de épocas, dado que no existe mucha variación en el óptimo alcanzado luego de $e \approx 30$, el punto de comparación será el tiempo de ejecución el cuál se buscará disminuir y claramente menos épocas implica menos tiempo de ejecución. Es por esto que se decide tomar $\eta = 0.01$ y $e = 40$. Dado que este método nos entrega la probabilidad de ser $\mathcal{C}_1$, para predecir proponemos un umbral de probabilidad p, de forma que si la probabilidad de ser $\mathcal{C}_1$ pasa este umbral entonces se clasificará como $\mathcal{C}_1$. Notar que esta discriminación deja fuera de $\mathcal{C}_1$ datos con a priori alta probabilidad de ser $\mathcal{C}_1$ pero permite obtener a, b como en la clasificación lineal.

Como idea para medir que tan buenos son los métodos, se propone un porcentaje de certeza/rendimiento que simplemente corresponde a dividir la cantidad de predicciones correctas por el tamaño del set de datos sobre el cuál se aplica la predicción que será el set de testeo.

Para la parte **c)** se pide implementar la **discriminación lineal de Fisher**, esta se basa en encontrar parámetros a, b y clasificar de manera tal que si $a^T x \geq -b$ entonces $c \to \mathcal{C}_1$ y en caso contrario $x \to \mathcal{C}_2$. Estos parámetros se calculan como

$$
\begin{aligned}
a &\propto S_w^{-1}(\mu_1 - \mu_2) \\
S_w &= \sum_{n \in \mathcal{C}_1} (x_n - \mu_1)(x_n - \mu_1)^T \\
&\quad + \sum_{n \in \mathcal{C}_2} (x_n - \mu_2)(x_n - \mu_2)^T \\
\mu_k &= \frac{1}{|\mathcal{C}_k|} \sum_{n \in \mathcal{C}_k} x_n \\
-b &= a^T \frac{1}{2}(\mu_1 + \mu_2)
\end{aligned}
$$

En base a las expresiones anteriores se obtiene la tabla 1 en dónde podemos comparar ambos métodos según su rendimiento sobre set de entrenamiento y validación.

	Train	**Validación**
RL-0.5	0.93%	0.94%
RL-0.95	0.84%	0.83%
Fisher	0.84%	0.80%

Table 1: Porcentaje de certeza. **RL**-p significa que el umbral que se usó es p

Vemos que **Fisher** y **RL**-0.95 tienen igual rendimiento sobre datos de entrenamiento pero al comparar con **RL**-0.5 sobre el de validación, **RL** lo supera. Vemos que el rendimiento de **RL** tiene fuerte dependencia del umbral de probabilidad p, dado que si se acepta un umbral más bajo la categoría de predicciones "correctas" aumenta agregando aquellos datos que tenían una alta probabilidad de ser de cierta clase pero no se aceptaba como correcta. Lo anterior no siempre sucede pero sugiere ajustar el umbral de forma de maximizar la certeza (figura 2).

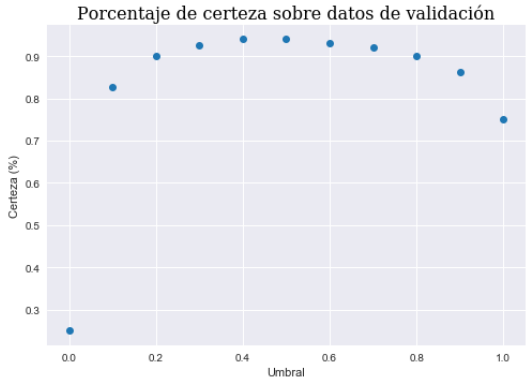

Figure 2: Umbral vs porcentaje de certeza

Desde ahora $p = 0.5$. Ajustados entonces los hiperparámetros η, e, p para **RL**, podemos comparar ambos métodos según el porcentaje de certeza sobre el set de testeo.

	RL-0.5	Fisher
%Certeza	0.93	0.83
Parámetro b	0	-0.012

Table 2: Comparación entre los métodos

Notemos que con **RL** obtenemos sólo peso $w = a$, pero con el método de clasificación usado podemos despejar [1] b que para $p = 0.5$ corresponde a 0. En base a la tabla 2 podemos concluir que **RL**-0.5 tiene mejor rendimiento sobre el set de testeo y por lo tanto es preferible, pero además se destaca su flexibilidad para poder ajustar el umbral p de forma de mejorar aún más su rendimiento.

No corresponde usar ciertos datos para testear que tan buenos son nuestros métodos si ya ocupamos esos datos para tomar decisiones sobre los parámetros del problema, la idea es testear sobre datos totalmente nuevos. Esto porque estaríamos sesgando la métrica de comparación puesto que los parámetros son tales que ya maximizan cierta métrica sobre estos datos. Por esto que es buena práctica dividir el set de datos en 3, para entrenar (ajustar parámetros), para validar (ajustar hiperparámetros) y testear (probar que tan bueno resulta el modelo).

[1] Notar que con **RL** obtenemos solo $a = w$, pero b se puede obtener tomando la inversa de σ a la desigualdad $\sigma(w^T x) > p$.