Tarea V

Joel Chacón Castillo

April 27, 2018

Ejercicio 1

En este ejercicio se demuestra la convergencia del algortimo perceptrón. Particulatmente, se considera el caso en que los datos son linealmente separables. La clase x_i se denota con y_i, donde y_i vive en $\{-1, 1\}$. Definiendo $z_i = x_i * y_i$, se busca un β tal que $\forall i : \beta^t z_i > 0$. Es decir, siempre se pueden rescalar las observaciones tal que la norma del los z_i sea menor que 1. Usando la notación anterior y eligiendo $\eta = 1$, en cada iteración se calcula:

$$\beta^{t+1} = \beta^t + z_i I(\beta z_i \leq 0) \tag{1}$$

A

Explicar que, si los datos son linealmente separables, entonces existe una β_{opt} tal que $\beta_{opt}^t z_i \geq 1$

Solución

Inicialmente se sabe que los datos son linealmente separables esto se define a continuación. Dos conjuntos de puntos A y B en un espacio n-dimensional son absolutamente linealmente separables si cada punto $(x_1, x_2, x_3, ..., x_n) \in A$ satisface $\sum_{i=1}^n \beta_i x_i > \beta_{n+1}$ y cada punto $(x_1, x_2, x_3, ..., x_n) \in B$ satisface $\sum_{i=1}^n \beta_i x_i > \beta_{n+1}$. Esto indica que si un conjunto de datos son linealmente separables, siempre existirá al menos un hiper-plano que los separe y además cuyo vector de pesos β sea ortogonal al plano. En esta formulación de puede observar que el valor β_{n+1} que indica la distancia entre el plano y el origen. Esto se puede observar en la figura 1, el hiper-plano divide de forma "lineal" a dos conjuntos de datos.

Particularmente, en el caso $\beta_{opt}^t z_i \geq 1$, se observa que el conjunto de puntos A se sigue la desigualdad:

$$\sum_{i=1}^{n} \beta_i a_i \geq 1 \tag{2}$$

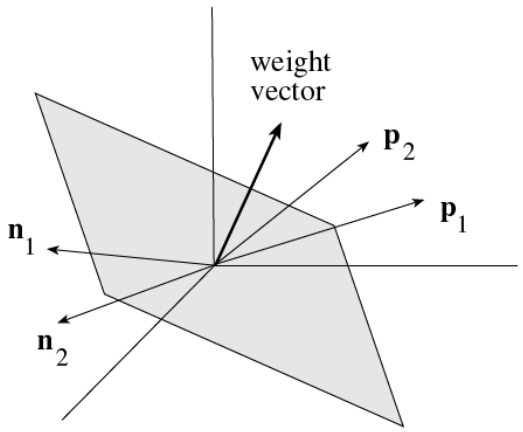

Figure 1: Ilustración del hiper-plano y un conjunto de datos linealmente separables.

y para el conjuto de puntos B que:

$$\sum_{i=1}^{n} \beta_i b_i < 1 \tag{3}$$

Esto se puede demostrar si se toma $\rho = max\{\sum_{i=1}^{n} \beta_i b_i - 1\}$. Además es claro que $\rho < \rho/2 < 0$. Si se establece que $\beta' = 1 + \rho/2$ para los puntos del conjunto A se establece que:

$$\sum_{i=1}^{n} \beta_i a_i - (\beta' - \frac{1}{2}\rho) \geq 0 \tag{4}$$

Esto significa que

$$\sum_{i=1}^{n} \beta_i a_i - \beta' \geq -\frac{1}{2}\rho > 0 \Rightarrow \sum_{i=1}^{n} \beta_i a_i > \beta' \tag{5}$$

De forma se establecen los mismos argumentos para el conjunto B:

$$\sum_{i=1}^{n} \beta_i b_1 - 1 = \sum_{i=1}^{n} \beta_i b_i - (\beta' - \frac{1}{2}\rho) \leq \rho \tag{6}$$

por lo tanto

$$\sum_{i=1}^{n} \beta_i b_i - \beta' \leq \frac{1}{2}\rho < 0 \tag{7}$$

Esto demuestra que los conjuntos A y B son linealmente separables, es decir existe una β_{op} la cual es ortogonal a un detemrinado hiperplano que separa el conjuno de datos.

B

Utilizando lo anterior, verifica que si se obtiene β^{t+1} utilizando la ecuación anterior para una z_i mal clasificada entonces:

$$0 \leq ||\beta^{t+1} - \beta_{opt}||^2 \leq ||\beta^t - \beta_{op}||^2 - 1 \tag{8}$$

Solución

Primeramente se comprueba la parte izquierda de la desigualdad, que consiste en demostrar que la norma de la diferencia elevada al cuadrado es un valor positivo, esto considerando la desigualdad del triángulo o desigualdad de Minkowski.

$$\begin{aligned} 0 &\leq ||\beta^{t+1} - \beta_{opt}||^2 \\ 0 &\leq ||\beta^{t+1} - \beta_{opt}|| \end{aligned} \tag{9}$$

Si se define cero de la siguiente forma:

$$0 = (\beta^{t+1} - \beta_{opt}) - (\beta^{t+1} - \beta_{opt}) \tag{10}$$

entonces mediante la desigualdad del triángulo se establece:

$$\begin{aligned} ||0|| &\leq ||\beta^{t+1} - \beta_{opt}|| + || - \beta^{t+1} + \beta_{opt}|| \\ 0 &\leq 2||\beta^{t+1} - \beta_{opt}|| \end{aligned} \tag{11}$$

Por lo tanto se comprueba que la norma del vector (compuesto por diferencias) es positivo. Ahora se procede con demostrar la parte derecha de la desigualdad

$$||\beta^{t+1} - \beta_{opt}||^2 \leq ||\beta^t - \beta_{op}||^2 - 1 \tag{12}$$

Inicialmente se desarrolla la parte izquierda de la desigualdad

$$\begin{aligned} ||\beta^{t+1}||^2 - 2\beta^{t+1}\beta_{opt} + ||\beta_{opt}||^2 &\leq ||\beta^t - \beta_{op}||^2 - 1 \\ ||\beta^{t+1}||^2 &\leq 2\beta^{t+1}\beta_{opt} - ||\beta_{opt}||^2 + ||\beta^t - \beta_{op}||^2 - 1 \end{aligned} \tag{13}$$

Es esta parte se introduce una cota superior a $||\beta^{t+1}||^2$, es decir dado que son desigualdades se puede reeplantear el problema en otro similar en este caso utiliza una cota superior de la siguiente forma:

$$||\beta^{t+1}||^2 = ||\beta^t||^2 + 2\beta^t z_i + ||z_i||^2 \tag{14}$$

Se sabe que $\beta^t z_i$ es negativo o cero (de otra forma no se necesitaría corregirse β^t usando z_i), podemos deducir que

$$||\beta^{t+1}||^2 \leq ||\beta^t||^2 + ||z_i||^2 \tag{15}$$

Por lo tanto:

$$||\beta^{t+1}||^2 \leq ||\beta^t||^2 + ||z_i||^2 \leq 2\beta^{t+1}\beta_{opt} - ||\beta_{opt}||^2 + ||\beta^t - \beta_{op}||^2 - 1 \tag{16}$$

De esta forma nos limitamos a demostrar la parte derecha de la desigualdad de la siguiente forma:

$$\begin{aligned}
||\beta^t||^2 + ||z_i||^2 &\leq 2\beta^{t+1}\beta_{opt} - ||\beta_{opt}||^2 + ||\beta^t - \beta_{op}||^2 - 1 \\
||\beta^t||^2 + ||z_i||^2 &\leq 2\beta^{t+1}\beta_{opt} - ||\beta_{opt}||^2 + ||\beta^t||^2 - 2\beta^t\beta_{op} + ||\beta_{op}||^2 - 1 \\
||z_i||^2 &\leq 2\beta^{t+1}\beta_{opt} - 2\beta^t\beta_{op} - 1 \\
||z_i||^2 &\leq 2\beta^t_{opt}(\beta^t + z_i) - 2\beta^t\beta_{op} - 1 \\
||z_i||^2 &\leq 2\beta^t_{opt}\beta^t + \beta^t_{opt}z_i - 2\beta^t\beta_{op} - 1 \\
||z_i||^2 &\leq \beta^t_{opt}z_i - 1 \\
||z_i||^2 + 1 &\leq \beta^t_{opt}z_i
\end{aligned} \tag{17}$$

Ahora bien se sabe que $\beta^t_{opt}z_i \geq 1$ y además $||z_i||^2 < 1$, de aqui ya se puede visualizar que la desigualdad se cumple. Sin embargo, si se puede transforma el problema de la siguiente forma:

$$\begin{aligned}
||z_i||^2 < 1 \leq \beta^t_{opt}z_i - 1 \leftrightarrow 1 &\leq \beta^t_{opt}z_i - 1 \\
0 &\leq \beta^t_{opt}z_i
\end{aligned} \tag{18}$$

Por lo tanto queda demostrada la desigualdad

C

Explicar que lo anterior significa que en un tiempo finito β^t debe converger.

Solución

Ejercicio 2

Este ejercicio es sobre el uso de métodos de clasificación para detectar billetes falsos. Basado en eel conjunto de datos que se anexan, resumir, visualizar y analizar los datos. Construir algunos clasificadores interesantes basado en el $K - NN$ y redes neuronales. Estimar el poder predictivo (dividir muchas veces los datos en un conjunto de prueba y de entrenamiento).

Solución

A partir de aquí se considera $w = \beta$. La demostración del algoritmo de perceptrón consiste en asumir que los puntos están acotados, es decir existe una distancia máxima en tre el origen y el punto mas alejado R y existe una cota inferior que es la distancia mínima entre el hiper-plano (que separa los puntos) y los puntos γ, esto se puede observar en la figura 2. Formalmente se define una esfera que envuelve a todos los puntos como $R = max||x||$ y el margen funcional como $\gamma = minw^T x$. Se define el teorema de *Block, Novikoff*: Dado el conjunto de entrenamiento S y esté

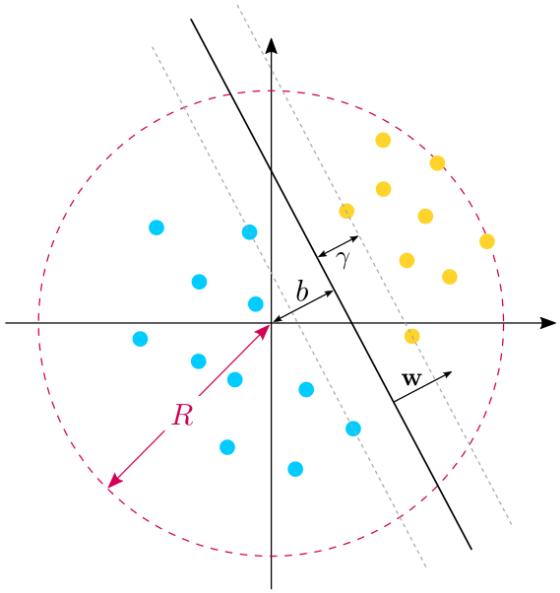

Figure 2: Conjunto de datos que consiste en dos clases que están separadas por un hiperplano $w^t x + b = 0$. El margen funcional γ es la mínima distancia que existe del hiper-plano a cualquiera de las muestras. La distancia del hiperplano al origen es denotado por b.

situado dentro de una esfera de radio R en relación al origen. Si se asume que el conjunto de datos es linelamente separable, y w_{opt}, $||w_{opt}|| = 1$, se define el hiper-plano separando los dos conjuntos de puntos, y además considerando un margen funcional $\rho > 0$. Si se inicializa un el vector normal al plano con cero. Entonces el número de actualizaciones (k) del algoritmo perceptrón es acotado por $k \leq \left(\frac{2R}{\gamma}\right)^2$.

Específicamente en nuestro caso se considera que la mínima distancia que existe entre el hiperplano es la unidad, es decir que $\gamma = 1$, que es lo demostrado en el punto anterior.

Análisis de los datos

Inicialmente, se verifican las características de los datos, es decir, es analizado el estado de la información. Primero se observa que el rango de los datos es similar y además existe información de 610 muestras de billetes reales y 762 muestras de billetes falsos. En la figura 3 se puede observar que entre las cuatro variables existe muy poca linealidad, por lo tanto la información que se obtendría al aplicar PCA puede ser incorrecta. Además, se puede observar que los datos ya fueron pre-procesados (la varianza tiene valores negativos posiblemente se aplico una transformación logarítmica), por lo tanto se elige no normalizar o estandarizar. Se observa también que no existen datos atípicos, otro indicador de que los datos son pre-procesados. En la parte inferior del gráfico se observa que la distribución que existe entre los billetes falsos y los reales es distinta. Principalmente en "Variance" y "Skewness" es distinta, quizas podríamos centrarnos únicamente en estas dos variables ya que aparentemente se podría construir apropiadamente un clasificador.

Clasificadores basados en el $K - NN$

Para esto contruyen varios clasificadores en base al incrementando del número de vecinos más cercanos, en función a lo que se pide se cree que es más apropiado utilizar el método de validación cruzada que sirve para evaluar los resultados de un análisis estadístico entre datos de entrenamiento y de prueba. El método de validación cruzada se repite 30 veces, y el número de vecinos son veinte, es decir se realizaron $30 * 20 = 600$ ejecuciones, también se toman un 70% de datos para entrenamiento y un 30% para prueba. Principalmente se utilizan las funciones *createDataPartitio*, *trainControl*, *train*, *predict* y *confusionMatrix* de R. En la imágen 4 se puede observar la exactitud que existe considerando un número de vecinos distintos. Se puede observar que existe una exactitud del 100% si se consideran entre veinte distintos tamaños de vecinos, esto puede indicar un posible

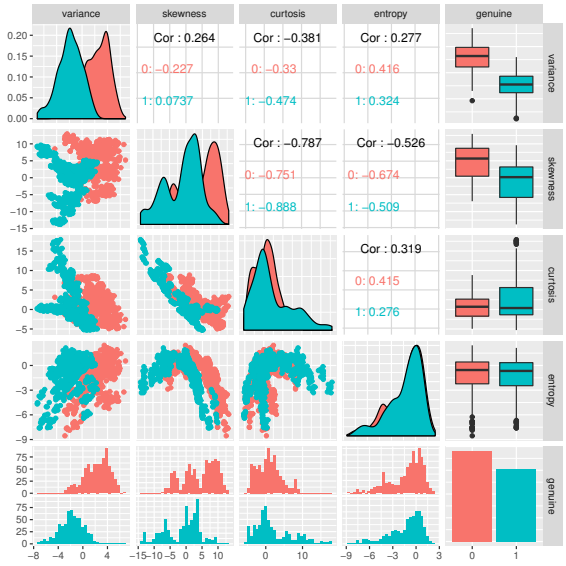

Figure 3: Información general de los billetes.

sobre-ajuste. (Creo que sería buena idea utilizar la validación cruzada para tunear algoritmos evolutivos). Además se puede apreciar que al aumentar el número de vecinos más cercanos existe un problema el clasificador tiende a perder exactitud. En la tabla 1 se puede observar la exactitud que existe al considerar distintos tamaños de vecinos k. El valor con mayor exactitud es asignado con $k = 13$. Para probar esto, se ejecuta el clasificador con la configuración previamente mencionada para determinar la exactitud de clasificación. Por lo tanto se prueba el modelo con los datos de prueba (30%). En la tabla 2 se muestra la matriz de confusión, que muestra el rendimiento de los clasificadores en un conjunto de prueba (clases también conocidas). Esta tabla indica que la exactitud de clasificación en los datos de prueba son muy altos (100%), ninguna instancia se clasificó de forma equivocada.

Clasificadores basados en redes neuronales

De igual forma se utiliza la validación cruzada como parte de este análisis. En este caso se consideran distintos números de capas y además se consideran distintos tamaños de pesos para realizar los correspondientes aprenddizajes. De igual forma se consideran 30 iteraciones en la validación cruzada y veinte tamaños de capas ocultas. Se establecieron tres razones de decrecimiento en los pesos que son 0, 0.001 y 0.1 respectivamente. En la figura 5 se pueden observar las tres configuraciones, es posible obtener una exactitud elevada del 100% al menos una vez. Además se puede observar que con un número pequeño de capas ocultas no se logra la exactitud deseada.

En la tabla 3, se puede observar que el mejor tamaño del modelo es con un número dos capas ($size = 2$) y una razón de aprendizaje del 0.1, hasta aquí son los resultados obtenidos con un 70% de entrenamiento y la validación cruzada. En la tabla 4 se muestran los resultados al aplicar este modelo con esta configuración (dos capas y razón de aprendizaje del 0.1) al 30% de datos de prueba. Se puede observar que la exactitud es del 100%. Por lo tanto se puede observar que no es necesario aplicar un número elevado de capas en un red neuronal para obtener los mejores resultados.

Ejercicio 3

Considera la siguiente función que surge de una red de base radial:

$$f_{\sigma,\beta,\mu} = \sum_{j=1}^{p} \beta_j exp(-||in - \mu_j||^2/\sigma_j) \tag{19}$$

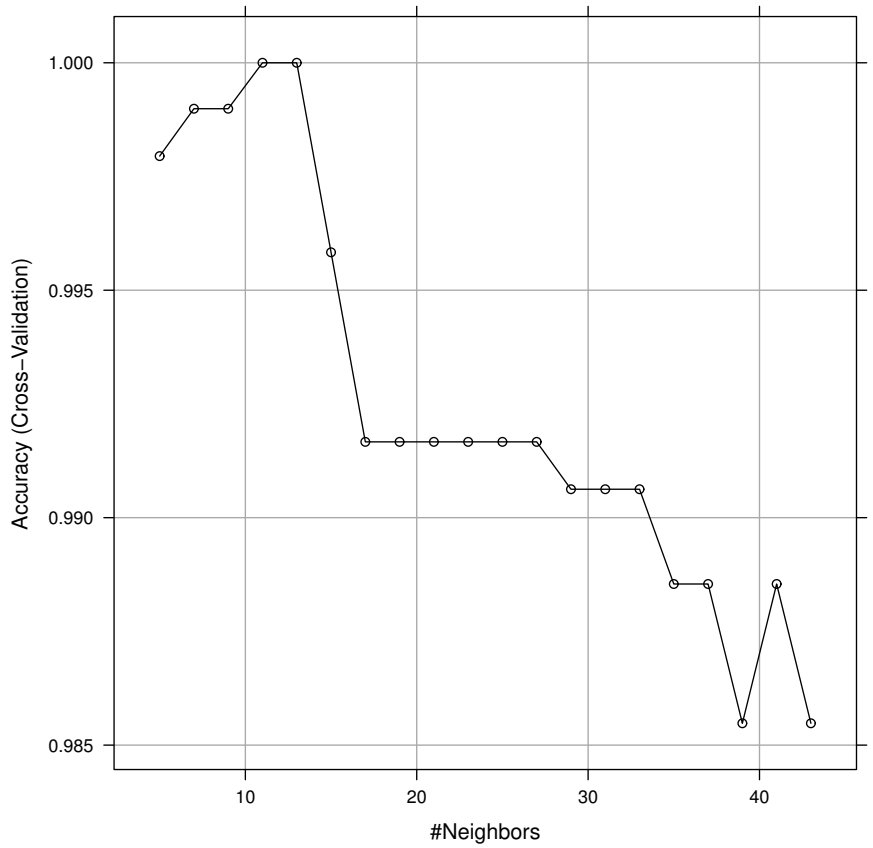

Figure 4: Análisis del clasificador K-NN.

Table 1: k-Nearest Neighbors, 961 samples. 4 predictors. 2 classes: '0', '1'. No pre-processing. Resampling: Cross-Validated (30 fold). Summary of sample sizes: 929, 929, 929, 929, 928, 929, ... Resampling results across tuning parameters:.

k	Accuracy	Kappa
5	0.9979482	0.9958518
7	0.9989899	0.9979516
9	0.9989899	0.9979516
11	1	1
13	1	1
15	0.9958333	0.991601
17	0.9916667	0.9832021
19	0.9916667	0.9832021
21	0.9916667	0.9832021
23	0.9916667	0.9832021
25	0.9916667	0.9832021
27	0.9916667	0.9832021
29	0.990625	0.9811352
31	0.990625	0.9811352
33	0.990625	0.9811352
35	0.9885417	0.9769685
37	0.9885417	0.9769685
39	0.9854798	0.9708395
41	0.9885417	0.9769685
43	0.9854798	0.9708723

Accuracy : 1
95% CI : (0.9911, 1)
No Information Rate : 0.5547
P-Value [Acc >NIR] : <2.2e-16

Kappa : 1
Mcnemar's Test P-Value : NA

Sensitivity : 1.0000
Specificity : 1.0000
Pos Pred Value : 1.0000
Neg Pred Value : 1.0000
Prevalence : 0.5547
Detection Rate : 0.5547
Detection Prevalence : 0.5547
Balanced Accuracy : 1.0000

'Positive' Class : 0

Table 2: Resultados obtenidos con *confusionMatrix*

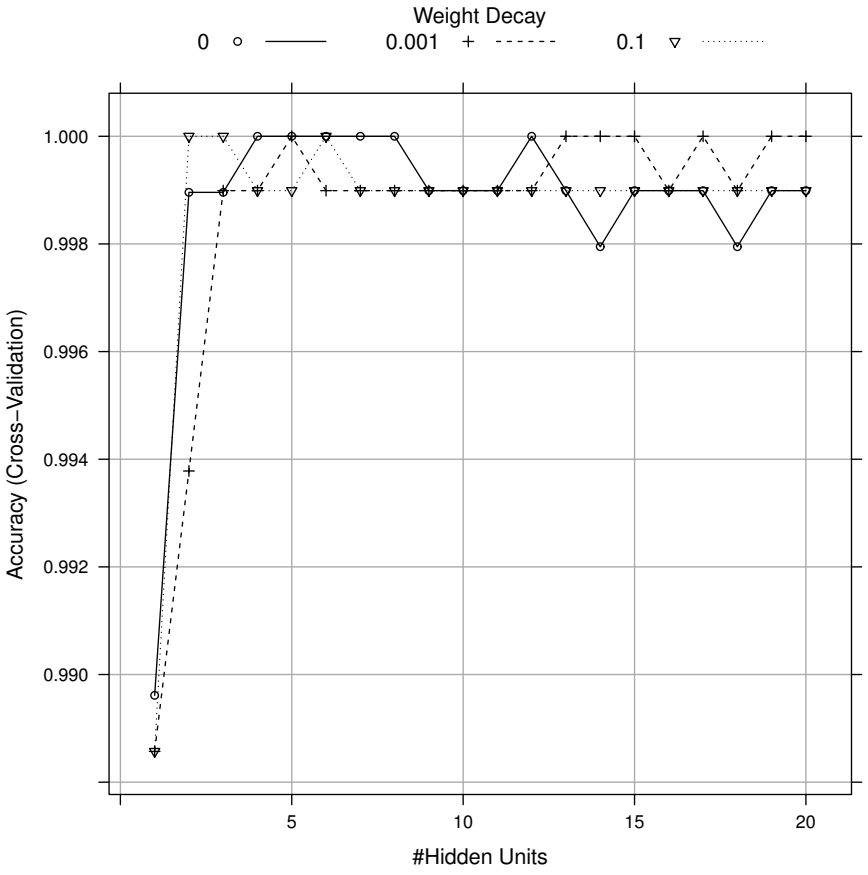

Figure 5: Análisis del clasificador redes neuronales.

Table 3: Neural Network, 961 samples, 4 predictors, 2 classes: '0', '1', No pre-processing, Resampling: Cross-Validated (30 fold), Summary of sample sizes: 929, 930, 929, 928, 928, 929, ... Resampling results across tuning parameters:

size	decay	Accuracy	Kappa	size	decay	Accuracy	Kappa	size	decay	Accuracy	Kappa
1	0	0.9896129	0.9789315	7	0.1	0.9989899	0.9979742	14	0.001	1	1
1	0.001	0.9885712	0.9767982	8	0	1	1	14	0.1	0.9989899	0.9979742
1	0.1	0.9885712	0.9767982	8	0.001	0.9989899	0.9979742	15	0	0.9989899	0.9979742
2	0	0.9989583	0.9978667	8	0.1	0.9989899	0.9979742	15	0.001	1	1
2	0.001	0.9937816	0.9870977	9	0	0.9989899	0.9979742	15	0.1	0.9989899	0.9979742
2	0.1	1	1	9	0.001	0.9989899	0.9979742	16	0	0.9989899	0.9979742
3	0	0.9989583	0.9978667	9	0.1	0.9989899	0.9979742	16	0.001	0.9989899	0.9979742
3	0.001	0.9989899	0.9979742	10	0	0.9989899	0.9979742	16	0.1	0.9989899	0.9979742
3	0.1	1	1	10	0.001	0.9989899	0.9979742	17	0	0.9989899	0.9979742
4	0	1	1	10	0.1	0.9989899	0.9979742	17	0.001	1	1
4	0.001	0.9989899	0.9979742	11	0	0.9989899	0.9979742	17	0.1	0.9989899	0.9979742
4	0.1	0.9989899	0.9979742	11	0.001	0.9989899	0.9979742	18	0	0.9979482	0.9958409
5	0	1	1	11	0.1	0.9989899	0.9979742	18	0.001	0.9989899	0.9979742
5	0.001	1	1	12	0	1	1	18	0.1	0.9989899	0.9979742
5	0.1	0.9989899	0.9979742	12	0.001	0.9989899	0.9979742	19	0	0.9989899	0.9979742
6	0	1	1	12	0.1	0.9989899	0.9979742	19	0.001	1	1
6	0.001	0.9989899	0.9979742	13	0	0.9989899	0.9979742	19	0.1	0.9989899	0.9979742
6	0.1	1	1	13	0.001	1	1	20	0	0.9989899	0.9979742
7	0	1	1	13	0.1	0.9989899	0.9979742	20	0.001	1	1
7	0.001	0.9989899	0.9979742	14	0	0.9979482	0.9958409	20	0.1	0.9989899	0.9979742

Accuracy : 1
95% CI : (0.9911, 1)
No Information Rate : 0.5547
P-Value [Acc >NIR] : <2.2e-16
Kappa : 1
Mcnemar's Test P-Value : NA
Sensitivity : 1.0000
Specificity : 1.0000
Pos Pred Value : 1.0000
Neg Pred Value : 1.0000
Prevalence : 0.5547
Detection Rate : 0.5547
Detection Prevalence : 0.5547
Balanced Accuracy : 1.0000
'Positive' Class : 0

Table 4: Resultados obtenidos con *confusionMatrix* en la red neuronal

donde $\sigma = (\sigma_1, \sigma_2, ..., \sigma_p)$, $\beta = (\beta_1, \beta_2, ..., \beta_p), \mu = (\mu_1, \mu_2, ..., \mu_p)$. Para un conjunto de datos $\{(in^d, out^d)\}$, define la función de costo:

$$E(\sigma, \beta, \mu) = \sum_d (out^d - f_{\sigma,\beta,\mu}(in^d))^2 \tag{20}$$

A

Calcular el gradiente de $E()$ con respecto a todos los parámetros.

Solución

En este caso se cambia la función objetivo a:

$$E(\sigma, \beta, \mu) = \frac{1}{2} \sum_d (out^d - f_{\sigma,\beta,\mu}(in^d))^2 \tag{21}$$

Además se define

$$\Phi(x, k) = exp(-\frac{||x - \mu_k||^2}{\sigma})$$

El cálculo del gradiente se indica a continuación:

$$\begin{aligned}
\nabla_\beta &= \sum_{d=1}^{N} \sum_{k=1}^{p} (out^d - f_{\sigma,\beta,\mu}(in^d)) \Phi(in^d, k) \\
\nabla_\mu &= \sum_{d=1}^{N} \sum_{k=1}^{p} (out^d - f_{\sigma,\beta,\mu}(in^d)) \Phi(in^d, k) exp(-||in - \mu_k||^2/\sigma_k) \\
\nabla_\sigma &= \sum_{d=1}^{N} \sum_{k=1}^{p} (out^d - f_{\sigma,\beta,\mu}(in^d)) \Phi(in^d, k) exp(-||in - \mu_k||^2/\sigma_k^2)
\end{aligned} \tag{22}$$

B

Implementa un algortimo que ajusta $f_{\sigma,\beta,\mu}(.)$ a un conjunto de datos $\{(in^d, out^d)\}$ dados, usando descenso de gradiente para encontrar los parámetros óptimos.

Solución

El método que se utiliza es el desenso del gradiente, este consiste en actualizar los valores conformado por la suma del gradiente con un tamaño de paso. Eso es utilizar de forma iterativa las siguientes ecuaciones:

$$\begin{aligned}
\beta &= \mu - \eta_\beta \nabla_\beta \\
\mu &= \mu - \eta_\mu \nabla_\mu \\
\sigma &= \sigma - \eta_\sigma \nabla_\sigma
\end{aligned} \tag{23}$$

El criterio de paro suele ser dado por la norma de los gradientes o un número máximo de iteraciones y el tamaño de paso de cada parámetro es fijado $\eta < 1$, sin embargo en la práctica es preferible utilizar valores $\eta < 0.1$. La aplicación utilizada consiste en interpolar funciones, así al inicio se proporciona una cantidad de datos de entrenamiento en base a una función dada y posteriormente dado el modelo construido se generan más puntos. En este caso se considera la función $y = x^2$. El error es medido como $\sum(\hat{y}(x) - y(x))^2$, es decir la diferencia entre el valor obtenido con el modelo y el real.

C

Implementar la versión anterior consiste en primero estimar (σ, μ) y después minimizar E sobre β, fijando $(\hat{\sigma}, \hat{\mu})$.

Solución

La implementación a utilizar es de interpolación, se basa en interpolar la función $f(x) = x^2$, el conjunto de entrenamiento esta compuesto de 100 puntos, y el conjunto de prueba de generar 1000 puntos, Para medir el rendimiento de cada variante se realizaron 30 ejecuciones con cada algoritmo, el número máximo de iteraciones es de 100. Se diseãron cuatro variantes de algoritmos, la primer variante consiste en aplicar evolución diferencial, es decir, los parámetros β, μ, σ son optimizados por medio de evolución diferencial, la configuración de evolución diferencial consiste en utilizar 10 individuos. La segunda variante consiste en aplicar el descenso del gradiente con la estimación del gradiente previa, el tamaño de paso es de $\eta = 0.001$ (para todos los parámetros). La tercer variante es implementar el gradiente con un tamaño de paso generados por una distribución normal $\eta = abs(N(0.001, 0.01))$. La cuarta y última variante consiste en aplicar el algoritmo de k-means para estimar el parámetro μ como se indica en el ejercicio.

En la tabla 5 se muestran los resultados en base a el error que es indicado en la ecuación (21), donde la mejor propuesta es la variante con descenso por gradiente y selección posteriormente está evolución diferencial. Particularmente evolución diferencial se ajusta adecuadamente a a construir el modelo debido a que no existe un sobre-ajuste, es decir, la solución no converge totalmente al óptimo en la parte de entrenamiento, por lo tanto en la parte de prueba no se considera un conjunto demasiado ajustado.

Una de las ventajas de evolución diferencial es que el conjunto de prueba puede ser probado con distinas configuraciones, ya que al ser un algoritmo evolutivo es proporcionad un conjunto de configuraciones óptimas. Además, evolución diferencial tiende puede mejorar en el sentido de que también puede optimizar el número de neuronas como parte del proceso de optimizar.

La mejor propuesta es la que se basa en el cálculo del gradiente con selección, específicamente se utilizan distintos tamaños de paso y sólo son actualizados los parámetros cuyo error sea mínimo, de esta manera se evitan desplazamientos elevados.

Table 5: Resultados de distintas variantes con *Radial Basis Function*

Implementación	Min	Max	Mediana	Media	std
Evolución diferencial	**0.1260239**	**1.890254**	**0.6438775**	0.5793472	0.3849773
Gradiente	0.1272871	16.01402	1.7076209	0.7516582	3.0062968
Gradiente con selección	0.1288594	13.07065	1.4559937	**0.505827**	2.6615571
Gradiente con k-means	5.212622	442.6889	205.735124	200.93085	77.19102

References