Práctica de Laboratorio 3
Tópicos de Inteligencia Artificial ...
CComp9-1

Alexander Giuliano Pinto De la Gala
alexander.pinto@ucsp.edu.pe

Universidad Católica San Pablo

1 Introducción

El presente Informe describe la experiencia y resultados de la tercera práctica de Laboratorio del curso de Tópicos de Inteligencia Artificial referida a Máquina de Vectores de Soporte (SVM) y Perceptrón Multicapa (MLP).

1.1 Perceptrón Multicapa

Un perceptrón es un clasificador lineal, que produce una salida basado en varias entradas formando una combinación lineal usando sus pesos de entrada, y usualmente un función de activación no lineal a la salida.

Un perceptrón multicapa está formado por perceptrones (neuronas) alineados en capas.

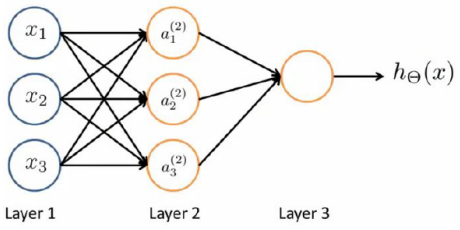


Fig. 1: Ejemplo de Multilayer Perceptrón

Cada una de estas unidades logísticas tiene una "activación" puede ser representada como a_i^j, donde i es la unidad y j es la capa que corresponde. La valor de activación es igual a aplicar una función no lineal (g) a la suma de los productos de los pesos (Θ) y sus valores de entrada (x) para el ejemplo de la Figura 1 :

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3) \tag{1}$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3) \tag{2}$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3) \tag{3}$$

$$h_\theta(x) = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}) \tag{4}$$

Otra forma de representar la activación es de la forma:

$$z^{(j)} = \Theta^{(j)}a^{(j-1)} \tag{5}$$
$$a^{(j)} = g(z^{(j)}) \tag{6}$$

El bias puede entonces ser representado como $a_0^{(j)} = 1$. Por lo tanto la salida del ejemplo queda como $z^{(3)} = \Theta^{(2)}a^{(2)}$ y la hipótesis como $h_\Theta(x) = a^{(3)} = g(z^{(3)})$

Para nuestra implementación consideramos la función g como la sigmoidea:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{7}$$

La función de costo a minimizar ahora está definida por:

$$J(\theta) = \frac{-1}{m}\big[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta(x^{(i)})_k + (1 - y_k^{(i)}) \log(1 - h_\theta(x^{(i)})_k)\big] \tag{8}$$

El cálculo de la Gradiente está dado por:

$$\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_{ij}}J(\theta) \tag{9}$$
$$D_{ij} = \frac{\partial}{\partial\theta_{ij}}J(\theta) \tag{10}$$

El algoritmo de *backpropagation* nos permite corregir los errores de cada neurona de capa en un sentido inverso, para esto utilizaremos el método directo, tomando como intuición que $\delta_j^{(l)}$ es igual a error del nodo j en la capa l.

Entonces en la capa de salida tenemos el error $\delta_j^{(l)} = a_j^{(l)} - y_j$, mientras que para que las capas escondidas es igual a:

$$\delta_j^{(l)} = (\Theta^{(l)})^T\delta^{(l)} * g'(z^{(l-1)}) \tag{11}$$

Para el caso de la función sigmoidea $g'(x) = x(1 - x)$

Luego hacemos el cálculo de las gradientes asociadas a cada capa:

$$\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)}(a^{(l)})^T \tag{12}$$

Finalmente $D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)}$ el promedio de las gradientes anteriores.

1.2 Máquina de Vectores de Soporte (SVM)

SVM es usualmente considerado un modelo de clasificación, pero es empleado también en modelos de regresión. Puede utilizar tanto variables continuas como categóricas. SVM construye un hiperplano en un espacio multidimensional para separar diferentes clases. Los *support vectors* son definidos como los puntos de datos más cercanos que recaen a la superficie de decisión o hiperplano [**?**]. Estos son los puntos más difíciles de clasificar y tienen influencia directa en la ubicación óptima de la superficie de decisión. En general existen infinitas soluciones para ubicar el hiperplano.

El margen es un brecha (*gap*) entre las dos líneas sobre los puntos más cercanos de las diferentes clases. El margen es calculado como la distancia perpendicular desde la línea que conforma los vectores de soporte. Si el margen es mayor entre las clases se considera un buen margen, si es más pequeño es un mal margen. La figura 2 detalla estos conceptos.

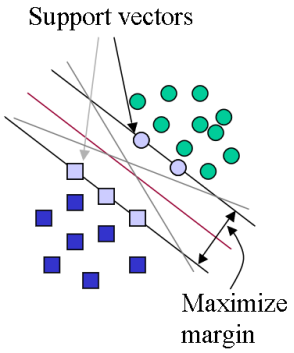


Fig. 2: SVM: Vectores de soporte y margen

El algoritmo de SVM puede expresarse como: 1. Generar hiperplanos los cuales segregan las clases de la mejor manera; 2. Seleccionar el hiperplano correcto con la máxima segregación desde sus puntos más cercanos; y 3. Repetir hasta convergencia o número de iteraciones.

Debido a que no todos los problemas pueden resolverse utilizando un hiperplano lineal, SVM utiliza lo que se conoce como *kernel trick*, lo cual es una transformación del espacio de entrada un espacio dimensional mayor, de manera que se pueda aplicar una separación lineal. Las ecuaciones 13 tratan esta transformación.

$$\text{Kernel Lineal :} \quad K(x, x_i) = \sum(x \cdot x_i) \tag{13}$$

$$\text{Kernel Polinomal :} \quad K(x, x_i) = 1 + \sum(x \cdot x_i)^d \tag{14}$$

$$\text{Kernel Función Radial :} \quad K(x, x_i) = \exp(\gamma * \sum(x - x_i^2) \tag{15}$$

1.3 Métrica

La métrica utilizada será *Accuracy* definido por:

$$Accuracy = \frac{\text{número de predicciones correctas}}{\text{número de predicciones hechas}} \tag{16}$$

Un *accuracy* igual uno (1) indica un modelo perfecto con predicciones sin errores. Un *accuracy* igual a cero (0) indica que no se obtuvo ninguna predicción acertada.

2 Implementación

La implementación[1] fue realizada en Python 3.6, con uso de bibliotecas como numpy y pandas para el caso de MLP. Para el caso de SVM se utilizó adicionalmente la librería sklearn.

La data utilizada corresponde a dos *datasets* "Iris" (iris) y "Enfermedad Cardiaca" (enfermedad_cardiaca).

La data fue normalizada por medio de la puntuación estándar definida por:

$$\frac{X - \mu}{\sigma} \tag{17}$$

donde:
μ es la media aritmética
σ es la desviación estándar del conjunto de datos.

3 Experimentos y Resultados

3.1 Experimento 1

Consiste en la búsqueda de los mejores parámetros de entrenamiento para los conjuntos "Enfermedad Cardiaca" e "Iris" utilizando validación cruzada(k) (*k-fold cross validation*, con k = 3).

En la validación cruzada se entrenó con los parámetros específicos según tabla 1 y se calculó el promedio de los *accuracies* obtenido al ejecutar el algoritmo del gradiente descendiente k veces. En cada vez, uno de los *folds* es usado como conjunto de prueba y el resto, los otros dos, como conjunto de entrenamiento. Los *folds* son conjuntos disjuntos dos a dos del conjunto de datos original.

Las Tablas 2 y 3 resumen los resultados del experimento. Cada fila representa la variación de la tasa de aprendizaje, y las columnas la variación del número de iteraciones.

[1] El código de la implementación se encuentra disponible en https://github.com/giulianodelagala/MLP-SVM

Table 1: Parámetros para el Experimento 1

Parámetros	Valores
Tasa de Aprendizaje	0.1, 0.25, 0.5, 0.75, 1.0
Iteraciones	[500,3500] en incrementos de 500
Número de capas	1, 2 y 3
Número de neuronas por capas	5, 10 y 15

Table 2: Accuracy de modelo MLP para el conjunto "Iris" para diferentes parámetros

Número de capas escondidas: 1							
Número de neuronas por capa: 5							
	500	1000	1500	2000	2500	3000	3500
0.10	0.853350	0.866830	0.893791	0.907271	0.900327	0.913399	0.933415
0.25	0.873366	0.920343	0.926879	0.953431	0.966503	0.966503	0.973039
0.50	0.913399	0.953431	0.973039	0.973039	0.973039	0.973039	0.973039
0.75	0.933415	0.973039	0.973039	0.973039	0.979984	0.979984	0.979984
1.00	0.940359	0.973039	0.973039	0.979984	0.979984	0.979984	0.979984
Número de neuronas por capa: 10							
	500	1000	1500	2000	2500	3000	3500
0.10	0.907271	0.920752	0.913807	0.920343	0.940359	0.940359	0.953431
0.25	0.913807	0.933415	0.933415	0.966503	0.973039	0.973039	0.973039
0.50	0.940359	0.973039	0.973039	0.973039	0.979984	0.979984	0.979984
0.75	0.959967	0.973039	0.973039	0.979984	0.979984	0.979984	0.979984
1.00	0.966503	0.973039	0.979984	0.979984	0.979984	0.979984	0.979984
Número de neuronas por capa: 15							
	500	1000	1500	2000	2500	3000	3500
0.10	0.906863	0.927288	0.940359	0.940359	0.940359	0.966503	0.946895
0.25	0.920343	0.940359	0.959967	0.966503	0.973039	0.973039	0.973039
0.50	0.933415	0.966503	0.973039	0.973039	0.973039	0.979984	0.979984
0.75	0.953431	0.973039	0.973039	0.979984	0.979984	0.979984	0.986520
1.00	0.966503	0.973039	0.979984	0.973039	0.973039	0.979984	0.979984
Número de capas escondidas: 2							
Número de neuronas por capa: 5							
	500	1000	1500	2000	2500	3000	3500
0.10	0.880310	0.973039	0.973039	0.973039	0.966503	0.979984	0.979984
0.25	0.919935	0.979984	0.973039	0.973039	0.979984	0.986520	0.986520
0.50	0.979984	0.979984	0.979984	0.973039	0.986520	0.979575	0.986520
0.75	0.979984	0.979984	**0.986520**	0.979575	0.979575	0.979575	0.972631
1.00	0.979984	0.979575	0.979575	0.979575	0.986520	0.986520	0.979575
Número de neuronas por capa: 10							
	500	1000	1500	2000	2500	3000	3500
0.10	0.920343	0.966503	0.979984	0.979984	0.973039	0.979984	0.979984
0.25	0.973039	0.973039	0.973039	0.979984	0.979984	0.973039	0.979575
0.50	0.973448	0.979984	0.979984	0.979575	0.986520	0.979575	0.979575
0.75	0.979984	0.979984	0.986520	0.979575	0.979575	0.979575	0.986520
1.00	0.979984	0.979575	0.986520	0.979575	0.986520	0.973039	0.979167
Número de neuronas por capa: 15							
	500	1000	1500	2000	2500	3000	3500
0.10	0.927288	0.973039	0.973039	0.973039	0.979984	0.979984	0.979984
0.25	0.979984	0.979984	0.979984	0.973039	0.986520	0.979984	0.979575
0.50	0.979984	0.979984	0.979575	0.979575	0.979575	0.986520	0.979575
0.75	0.973039	0.986520	0.986520	0.979575	0.979575	0.973039	0.973039
1.00	0.979984	0.986520	0.986520	0.972631	0.972631	0.973039	0.979167
Número de capas escondidas: 3							
Número de neuronas por capa: 5							
	500	1000	1500	2000	2500	3000	3500
0.10	0.746324	0.872958	0.959967	0.973039	0.973039	0.979984	0.973039
0.25	0.946078	0.979984	0.973039	0.986520	0.986520	0.986520	0.986520
0.50	0.979984	0.986520	0.979575	0.986520	0.973039	0.979575	0.973039
0.75	0.979984	0.979575	0.986520	0.973039	0.986520	0.986520	0.979167
1.00	0.979575	0.986520	0.979984	0.986520	0.986520	0.979575	0.986520
Número de neuronas por capa: 10							
	500	1000	1500	2000	2500	3000	3500
0.10	0.880310	0.966503	0.979984	0.979984	0.979984	0.979984	0.979984
0.25	0.979984	0.973039	0.986520	0.979984	0.986520	0.979575	0.986520
0.50	0.979984	0.979575	0.979575	0.986520	0.986520	0.973039	0.973039
0.75	0.979984	0.986520	0.979575	0.979575	0.972631	0.972631	0.973039
1.00	0.979575	0.979575	0.972631	0.973039	0.979167	0.973039	0.973039
Número de neuronas por capa: 15							
	500	1000	1500	2000	2500	3000	3500
0.10	0.959967	0.973039	0.979984	0.973039	0.973039	0.979984	0.986520
0.25	0.979984	0.979984	0.979984	0.979575	0.986520	0.979575	0.986520
0.50	0.979984	0.986520	0.979575	0.973039	0.979575	0.972631	0.973039
0.75	0.986520	0.986520	0.966095	0.973039	0.973039	0.973039	0.973039
1.00	0.973039	0.973039	0.972631	0.973039	0.973039	0.966095	0.966095

Table 3: Accuracy de modelo MLP para el conjunto "Enfermedad Cardiaca" para diferentes parámetros

Número de capas escondidas: 1							
Número de neuronas por capa: 5							
	500	1000	1500	2000	2500	3000	3500
0.10	0.848185	0.877888	0.854785	0.858086	0.897690	0.877888	0.887789
0.25	0.851485	0.861386	0.881188	0.881188	0.887789	0.884488	0.891089
0.50	0.881188	0.904290	0.884488	0.858086	0.891089	0.884488	0.884488
0.75	0.904290	0.874587	0.871287	0.894389	0.900990	0.900990	0.887789
1.00	0.874587	0.897690	0.877888	0.894389	0.907591	0.897690	0.904290
Número de neuronas por capa: 10							
	500	1000	1500	2000	2500	3000	3500
0.10	0.851485	0.881188	0.864686	0.858086	0.871287	0.871287	0.897690
0.25	0.867987	0.891089	0.874587	0.907591	0.891089	0.900990	0.910891
0.50	0.887789	0.907591	0.914191	0.927393	0.917492	0.914191	0.917492
0.75	0.904290	0.900990	0.924092	0.904290	0.933993	0.914191	0.910891
1.00	0.914191	0.897690	0.924092	0.907591	0.904290	0.904290	0.940594
Número de neuronas por capa: 15							
	500	1000	1500	2000	2500	3000	3500
0.10	0.851485	0.871287	0.884488	0.897690	0.881188	0.887789	0.900990
0.25	0.871287	0.894389	0.917492	0.937294	0.937294	0.914191	0.914191
0.50	0.904290	0.897690	**0.943894**	0.917492	0.920792	0.900990	0.924092
0.75	0.924092	0.910891	0.924092	0.924092	0.927393	0.924092	0.920792
1.00	0.907591	0.924092	0.917492	0.910891	0.907591	0.910891	0.910891
Número de capas escondidas: 2							
Número de neuronas por capa: 5							
	500	1000	1500	2000	2500	3000	3500
0.10	0.825083	0.861386	0.871287	0.887789	0.897690	0.900990	0.894389
0.25	0.894389	0.861386	0.884488	0.897690	0.894389	0.874587	0.910891
0.50	0.891089	0.871287	0.884488	0.907591	0.900990	0.874587	0.917492
0.75	0.907591	0.874587	0.907591	0.904290	0.904290	0.910891	0.894389
1.00	0.871287	0.891089	0.877888	0.894389	0.887789	0.874587	0.887789
Número de neuronas por capa: 10							
	500	1000	1500	2000	2500	3000	3500
0.10	0.818482	0.861386	0.897690	0.910891	0.894389	0.900990	0.904290
0.25	0.877888	0.897690	0.917492	0.914191	0.924092	0.930693	0.900990
0.50	0.891089	0.907591	0.933993	0.914191	0.904290	0.924092	0.910891
0.75	0.887789	0.914191	0.917492	0.910891	0.924092	0.917492	0.907591
1.00	0.914191	0.907591	0.924092	0.910891	0.927393	0.920792	0.907591
Número de neuronas por capa: 15							
	500	1000	1500	2000	2500	3000	3500
0.10	0.848185	0.877888	0.904290	0.900990	0.904290	0.914191	0.894389
0.25	0.900990	0.907591	0.914191	0.910891	0.910891	0.917492	0.924092
0.50	0.907591	0.920792	0.917492	0.927393	0.937294	0.900990	0.927393
0.75	0.907591	0.937294	0.930693	0.891089	0.927393	0.924092	0.910891
1.00	0.914191	0.914191	0.943894	0.924092	0.914191	0.907591	0.927393
Número de capas escondidas: 3							
Número de neuronas por capa: 5							
	500	1000	1500	2000	2500	3000	3500
0.10	0.818482	0.851485	0.874587	0.891089	0.864686	0.871287	0.874587
0.25	0.854785	0.891089	0.917492	0.900990	0.907591	0.907591	0.907591
0.50	0.897690	0.900990	0.907591	0.854785	0.904290	0.930693	0.904290
0.75	0.884488	0.914191	0.910891	0.924092	0.910891	0.874587	0.871287
1.00	0.910891	0.894389	0.887789	0.900990	0.900990	0.910891	0.914191
Número de neuronas por capa: 10							
	500	1000	1500	2000	2500	3000	3500
0.10	0.818482	0.877888	0.900990	0.887789	0.910891	0.881188	0.904290
0.25	0.874587	0.914191	0.907591	0.897690	0.900990	0.917492	0.914191
0.50	0.884488	0.914191	0.924092	0.924092	0.914191	0.894389	0.897690
0.75	0.920792	0.924092	0.900990	0.894389	0.920792	0.910891	0.920792
1.00	0.904290	0.937294	0.924092	0.927393	0.920792	0.907591	0.910891
Número de neuronas por capa: 15							
	500	1000	1500	2000	2500	3000	3500
0.10	0.848185	0.900990	0.904290	0.897690	0.917492	0.910891	0.910891
0.25	0.858086	0.914191	0.900990	0.943894	0.930693	0.924092	0.924092
0.50	0.897690	0.924092	0.924092	0.907591	0.920792	0.917492	0.933993
0.75	0.917492	0.930693	0.914191	0.920792	0.940594	0.917492	0.917492
1.00	0.907591	0.940594	0.897690	0.917492	0.914191	0.930693	0.894389

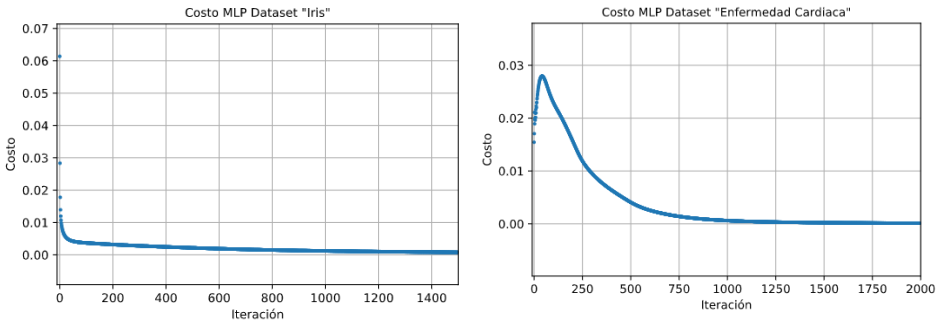
De la Tabla 2 para el conjunto de datos "Iris", se aprecia que el mejor accuracy obtenido es de 0.986520, para una arquitectura de 2 capas con 5 neuronas cada una y una tasa de aprendizaje de 0.75 para 1500 iteraciones. Se dan resultados similares en otras arquitecturas pero con mayor costo computacional. Otro punto a favor es que los resultados se mantienen estableces para un mayor número de iteraciones.

De la Tabla 2 para el conjunto de datos "Enfermedad Cardiaca", se aprecia que el mejor accuracy obtenido es de 0.943894, para una arquitectura de 1 capa con 15 neuronas cada una y una tasa de aprendizaje de 0.50 para 1500 iteraciones. Se dan resultados similares en otras arquitecturas pero con mayor costo computacional. Sin embargo es un valor aislado. El comportamiento del accuracy para este dataset es muy errático, por lo que es posible que valores alto se consigan por un mejor conjunto inicial de pesos generados aleatoriamente. El accuracy se muestra mas estable para una de 2 capas y 15 neuronas.

La Figura 3 muestra la variación del costo para el entrenamiento de MLP para los dos datasets. Como se puede apreciar para el caso de "Iris", la convergencia es directa. Sin embargo para "Enfermedad Cardiaca" se aprecia que hay un comportamiento errático inicial, probablemente debido a que se encuentra en un mínimo local.

Fig. 3: Función de Costo para modelo MLP

(a) (b)



3.2 Experimento 2

En este experimento se usó la biblioteca Scikit-learn de Python para implementar una SVM usando los conjuntos de datos "Enfermedad Cardiaca" e "Iris". Encontrar los mejores valores para sus parámetros usando validación cruzada. En la Tabla 5, se muestra el Accuracy promedio obtenido al variar los parámetros de los kernels: lineal, polinomial, gaussiano, y el parámetro de regularización C.

Table 4: Parámetros para el Experimento 2

Parámetros	Valores
Kernel	linear, polinomial, gaussiano
Regularización	[0.5,5.0] en incrementos de 0.5

Table 5: Accuracy de modelo SVM para el conjunto "Iris" para diferentes parámetros

Dataset: "Iris"

	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
linear	**0.993464**	0.979575	0.979575	0.979575	0.979575	0.979575	0.979575	0.979575	0.973039	0.973039
poly	0.973856	**0.993464**	**0.993464**	0.986520	0.986520	0.986520	0.986520	0.986520	0.979575	0.979575
rbf	0.953840	0.960376	0.967320	0.967320	0.973856	**0.993464**	**0.993464**	**0.993464**	**0.993464**	**0.993464**

Dataset: "Enfermedad Cardiaca"

	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
linear	**0.848185**	0.841584	0.844884	0.844884	0.841584	0.841584	0.838284	0.838284	0.844884	0.844884
poly	0.673267	0.693069	0.696370	0.696370	0.706271	0.709571	0.709571	0.712871	0.706271	0.706271
rbf	0.633663	0.673267	0.689769	0.693069	0.699670	0.706271	0.706271	0.702970	0.712871	0.709571

De la Tabla 5, se aprecia que para el dataset "Iris", los tres kernel llegan a un accuracy de 0.993464, sin embargo el kernel polinomial, consigue un accuracy más estable para los diferentes variaciones del parámetro de regularización C, esto para el caso de un grado de polinomio igual a 2.

Para el caso del dataset "Enfermedad Cardiaca", es claro que el kernel lineal es el que consigue mejores resultados con accuracy de 0.848185 con un parámetro de regularización de 0.5. Los otros kernels presentan valores bastante menores de accuracy. Este es posible debido a que este dataset solo cuenta con dos clases.

4 Conclusiones

La presente ha descrito los experimentos de la Práctica de Laboratorio 3 del Curso de Tópicos de Inteligencia Artificial.

Los diversos experimentos han demostrado el poder de los modelos de Multilayer Perceptron y Support Vector Machine. MLP ha conseguido un mejor accuracy para ambos datasets, sin embargo SVM tiene otros parámetros los cuales podrían ser configurados correctamente y mejorar su desempeño.

Otro punto en contra de MLP es su comportamiento de caja negra, el cual no permite entender con claridad como es su proceso interior para con ello tomar mejores decisiones en cuanto a los parámetros, a diferencia de SVM que teniendo una base matemática más compleja es posible entender su proceso.

References

1. Andrew Ng, CS229 Lecture Notes