Prácticos Redes Neuronales

Grupo 3: Emiliano Bodean - Zacarias Ojeda

2020-02-01

Contents

Chapter 1

Guía 1

1.1 Ejercicio 1:

Realice un programa que permita el entrenamiento y prueba de un perceptrón simple con una cantidad variable de entradas.

1.1.1 Resolución del problema OR

- Lectura de los patrones de entrenamiento
- Selección de parámetros y entrenamiento de perceptrón

Utilizamos las funciones implementadas en el archivo "PerceptronSimple.R"

El entrenamiento nos devuelve la tasa obtenida en cada época y al final mostramos los valores de los pesos w que definen la recta del modelo.

$ pesos = [w_0, w_1, w_2, ...w_n] $

- Graficas

En la gráfica vemos como la recta del modelo separa el dominio según la clase.

- Prueba con datos de test

Por ser este un problema sencillo donde las clases son separables por una recta, la tasa de aciertos en test nos da un 100%.

1.1.2 Resolución del problema XOR

- Lectura de los patrones de entrenamiento
- Selección de parámetros y entrenamiento de perceptrón

Vemos que luego de 10 épocas el perceptrón simple no logra resolver el problema del XOR, ni tampoco mejorar su tasa de aciertos.

- Graficas

En la gráfica vemos la recta obtenida que deja todos los patrones clasificados de igual manera.

- Prueba con datos de test

Este problema no pudo ser resuelto por le perceptrón simple, la tasa de aciertos obtenida es peor que el azar.

1.1.3 Preguntas:

- ¿Pueden ser resueltos ambos problemas (OR y XOR) empleando un perceptrón simple? Justifique su respuesta.

En los resultados obtenidos verificamos que no se puede resolver el problema XOR con el perceptrón simple porque no es posible dividir la clase con una recta que separe el dominio en dos.

- ¿Qué efecto tiene la tasa de aprendizaje en el entrenamiento del perceptrón? Explique cómo este parámetro afecta a la actualización de la frontera de decisión

La tasa de aprendizaje nos permite variar cuanto se actualizan los valores de los pesos. Una tasa muy baja puede generar que el algoritmo demore mucho en llegar al mínimo, y una tasa muy alta puede generar que el algoritmo diverja y nunca llegue a un mínimo. En nuestro caso, se implementó una función que posee una tasa de aprendizaje de 0,05 (nu=0.05) por defecto y no fue necesario cambiarlo.

1.2 Ejercicio 2:

1.2.1 Punto a:

- Lectura de datos
- Grafica
- Generamos las particiones

Utilizamos la función implementada en el archivo “Particiones.R”

Esta función nos devuelve un listado de 5 listas de ID para entrenamiento, de aproximadamente 800 elementos, y 5 listas de ID para prueba, de aproximadamente 200 elementos.

- Generamos los modelos con el perceptrón simple.

1.2.2 Punto b:

- Lectura de datos
- Generamos las particiones

Llamamos a la función dos veces con dos semillas para obtener las 10 particiones. La función implementada genera las particiones sin reemplazo, con una relación 80/20 no se pueden generar más de 5 particiones.

- Generamos los modelos con el perceptrón simple.

– Datos con desviaciones del 10%

– Datos con desviaciones del 50%

– Datos con desviaciones del 70%

1.2.3 Preguntas:

- ¿Qué beneficio supone el uso de validación cruzada?

El uso de validación cruzada nos permite usar todos los datos para test y poder evaluar de mejor manera si el método utilizado para la generación del modelo es bueno.

- ¿Qué ocurre con la tasa de acierto del perceptrón para los diferentes datasets del ejercicio 2b? Analice el desempeño al incrementarse la dispersión de los datos.

Al aumentar la dispersión de los datos, disminuye la tasa de aciertos porque el plano no permite separar correctamente las clases.

1.3 Ejercicio 3

El algoritmo implementado se encuentra en el archivo “PerceptronMulticapa.R”

- Lectura de datos
- Gráfica de datos con clase
- Entrenamiento de perceptrón
- Gráfica con clasificación del perceptron

1.3.1 Punto b

Incorporación del termino de momento.

Se agrega una variable alfa a la función, si la variable es cero no aplica el termino de momento.

1.3.2 Punto c

1.3.3 Preguntas:

- ¿Cuál es la arquitectura mínima que emplearía para resolver este problema? ¿Por qué?

La arquitectura mínima es una capa de 3 neuronas y una capa de 1 neurona, esta fue la arquitectura utilizado para resolver la guía. Al estar una de las clases rodeada o contenida dentro de la otra clase, se requiere una arquitectura que genere una zona cerrada, para esto se requieren al menos tres rectas que definen una zona triangular. La arquitectura utilizada genera tres rectas con la primera capa de 3 neuronas y luego una capa de 1 neurona que a partir de la salida de las anteriores clasifica según si está dentro o fuera de esta zona triangular.

- ¿El número de épocas requerido para entrenar un perceptrón multicapa se modifica al emplear el término de momento? Analice el efecto de este término de momento sobre el entrenamiento.

El número de época al utilizar termino de momento se reduce. En nuestro caso, se ve que disminuye de 296 a 275. Para lograr esto se agrega un término a la corrección de los pesos utilizando el error del patrón anterior.

- ¿Qué conclusión puede sacar a partir del ejercicio 3c?

Del ejercicio 3c podemos concluir que es muy importante conocer todas las herramientas y entender bien el problema, en este caso el problema se podía resolver con una tasa de acierto mayor al 90% utilizando un perceptrón simple. Esto reduce los tiempos de entrenamiento y la complejidad del modelo.

1.4 Ejercicio 4

En ejercicio 4 quedó un head(20) para disminuir la cantidad de combinaciones de leave2out

- Lectura de los patrones de entrenamiento
- Selección de parámetros y entrenamiento de perceptrón Utilizamos las funciones implementadas en el archivo “PerceptronMulticapa.R”

1.4.1 Leave One Out

La media de los errores leave_one_out es: 1.9801029 El desvio standard de los errores leave_one_out es: 0.93459

1.4.2 Leave 2 Out

La media de los errores leave_one_out es: 1.9801029 El desvio standard de los errores leave_one_out es: 0.93459

Chapter 2

Guía 2

2.1 Ejercicio 1

2.1.1 Resolución del problema XOR con una red neuronal RBF

- Lectura de los patrones de entrenamiento
- Selección de parámetros y entrenamiento de perceptrón

En este caso se utilizan 4 gausianas por la distribución de los datos.

- Prueba con datos de test

2.1.2 Resolución del problema Iris con una red neuronal RBF

- Lectura de los patrones de entrenamiento
- Selección de parámetros y entrenamiento de perceptrón
- Prueba con datos

Cantidad de parámetros:
En una red MLP con una estructura (3,1), tenemos los siguientes parámetros:
$numParamMLP = ParmetrosdeCapa1 + ParmetrosdeCapa2$
$numParamMLP = [(4entradas + 1) * 3neuronas] + [(3entradas + 1) * 1neurona]$
$numParamMLP = 5 * 3 + 4 * 1 = 19parmetros$
Una red RBF con 19 parámetros podría tener la siguiente distribución:
$numParamRBF = ParmetrosdeGausianas + ParmetrosdePerceptrones$
$numParamRBF = [3centros] + [(3entradas + 1) * 3neurona]$
$numParamRBF = 3 + 4 * 3 = 15parmetros$

- Prueba con datos

2.2 Ejercicio 2

- Lectura de datos
- Preprocesamiento de los datos

Generamos un dataset que contenga seis valores consecutivos en cada registro, cinco tomados como datos de entrada y un sexto valor tomado como clase.

Antes de generar el modelo, tenemos que definir el número de gausianas. Utilizamos la gráfica de Elbow para definir el k a utilizar en el modelo.

Mirando la gráfica anterior tomamos un valor de k = 4, es donde la gráfica hace el codo y queda aproximadamente constante.

- Normalizamos los datos.
- Dividimos los datos en Train y Test, utilizando un 70% para entrenamiento.
- Generamos el modelo con los datos de entrenamiento.
- Aplicamos el modelo a los datos de Train y Test
- Grafica de error en Test
- Generamos el modelo a aplicar para realizar las predicciones con todos los datos.

Aplicamos el modelo a los mismos datos de entrenamiento para graficar error en train.

Grafica de error

- Gráfica del valor predicho y el valor real
- Predecimos un nuevo valor

Tomamos los últimos 5 valores del dataset y predecimos cual será el próximo valor.

2.3 Ejercicio 3

- Lectura de datos
- Graficamos los datos de entrada
- Inicialización de Grilla SOM

Implementamos una función con dos casos, una red cuadrada y una red lineal.

- Función de vecindad

Implementamos una función vecindad. Devuelve los ID de los vecinos de un nodo en un entorno cuadrado.

- Entrenamiento de la res SOM - Circulo

Implementamos la función de entrenamiento para redes SOM, y se la utiliza en el caso del Circulo.

La función dentro tiene tres etapas, - Ordenamiento topológico o global - Transición - Ajuste fino

La vecindad para el ajuste de los pesos decrece en forma lineal hasta 1 y la taza de aprendizaje decrece linealmente hasta 0,05.

- Inicialización de Grilla SOM - Te
- Entrenamiento de la res SOM - Te
- Inicialización de Grilla SOM Unidimensional - Te
- Entrenamiento de la res SOM Unidimensional - Te

Podemos observar que con la misma cantidad de neuronas, pero con la distribución lineal podemos obtener una red que se ubica por completo sobre los datos con el mismo entrenamiento.

2.4 Ejercicio 4

- Lectura de datos
- Graficamos los datos de entrada
- Inicialización de Grilla SOM - Clouds

Generamos una grilla SOM de 49 nodos, con esto tenemos un nodo cada 100 patrones aproximadamente.

- Entrenamiento de la res SOM - Clouds
- Etiquetado de neuronas

Para el etiquetado de neuronas, se evalúa la cantidad de patrones de cada clase en el entorno cercano de cada neurona. Se toma como entorno cercano un radio de la mitad de la distancia promedio de la neurona con sus vecinas.

- Clasificación con red SOM

Se asigna como clase ganadora a cada patrón a la clase de la neurona más cercana.

- Visualizamos los puntos clasificados
- Calculamos la tasa de aciertos

Obtuvimos una tasa de un 86% de acierto, para mejorar esto se podría entrenar una red SOM con mayor numeró de neuronas.

Chapter 3

Guia 3

3.1 Ejercicio 1

Implemente las estructuras de datos y algoritmos básicos para la solución de un problema mediante algoritmos genéticos. Pruebe estas rutinas para buscar el mínimo global de las siguientes funciones:

$$-x\sin(\sqrt{|x|})$$
$$x + \sin(3x) + 8\cos(5x)$$
$$(x^2 + y^2)^{0.25} * [\sin^2(50 * (x^2 + y^2)^{0.1}) + 1]$$

3.1.1 Preguntas

- ¿Corresponde al mínimo global el valor encontrado? Repita la búsqueda varias veces y determine el valor medio y desvío.
- ¿Se encuentra ahora el mínimo global dentro del intervalo?

3.1.2 Función 1

Se puede apreciar en la gráfica que el mínimo encontrado coincide con el mínimo global dentro del rango. El valor medio de los resultados de la funcion01 es **421.0046494**, el desvio estándar es **0.204572**

3.1.3 Función 2

Se puede apreciar en la gráfica que el mínimo encontrado coincide con el mínimo global dentro del rango. El valor medio de los resultados de la funcion01 es **1.8658132**, el desvio estándar es **0.0071057**

3.1.4 Función 3

Se puede apreciar en la gráfica que el mínimo encontrado coincide con el mínimo global dentro del ranggo (tanto para x como para y). El valor medio de los resultados de la funcion01 es **0.0210934, 0.0210934**, el desvio estándar es **0.0688038, 0.0688038**

3.2 Ejercicio 2

En el archivo desconocido1.csv se ha registrado información de un proceso que puede describirse mediante la ecuación:

$$y1 = a_1 x_1^3 + a_2 x_1^2 + a_3 x_1 + a_4$$

Se sabe que las mediciones contienen ruido, y que los parámetro del sistema se encuentran acotados en el intervalo $[-5,1.5]$. Utilice un algoritmo genético para determinar los parámetros delmodelo. Calcule el error cuadrático total obtenido de la comparción entre los datos provistos y la función aproximada mediante el algoritmo. ¿Qué puede concluir del ajuste?