Daniel Felipe Tamayo García

Juan David Ospina Arango

Técnicas de Aprendizaje Estadístico

Marzo de 2019

Resumen

En este trabajo se realizo con el conjunto de datos MINST y se aplicaron cinco algoritmos para reconocer dígitos escritos a mano. Los algoritmos fueron adaptados de diferentes paquetes de R; las tasas de error, entrenamiento y tiempos de prueba se estudiaron y se compararon los diferentes algoritmos de aprendizaje.

Introducción

El propósito de este trabajo es explotar algunas técnicas de aprendizaje: Arboles de decisión, Redes Neuronales, Maquinas de soporte vectorial, y usarlos en un problema en especifico.

El conjunto de datos MINST incluye miles de dígitos escritos a mano, estos fueron utilizados para el entrenamiento y pruebas de las diferentes técnicas de aprendizaje. El reconocimiento de imágenes es un tema interesante y que esta tomando mucha importancia en el mundo real. El reconocimiento de digitos es relevante por ejemplo en el problema que vamos a abarcar que es el envío y recepción de postales.

Reconocimiento de Dígitos

La fuente de los datos utilizada es el conjunto de datos MNIST ("Modified National Institute of Standards and Technology"), descargada de la pagina oficial del curso "Técnicas de Aprendizaje Estadístico", con el objetivo de reconocer cual es el dígito escrito en una imagen. El conjunto de datos MNIST es clásico en la comunidad de aprendizaje automático se ha estudiado por mucho tiempo. Este conjunto utilizado tiene 60000 imágenes y un conjunto de prueba de 10000 imágenes, donde cada imagen tiene 28-x28 pixeles con niveles de gris de 0-254.

Preparación de datos

Generalmente para el reconocimiento de objetos tenemos que calcular vectores de características a partir de imágenes, la forma mas sencilla de construir un vector de características es usar pixeles en bruto, obviamente, el pixel sin formato no es un buen vector de características considerando las distorsiones de los dígitos escritos a mano, pero se decidió usarlos para simplificar las preguntas.
Se cargaron el archivo *mnist_train.csv* en R como marco de datos, el conjunto de datos incluyo 42000 observaciones y 785 variables. La primera variable se etiqueta, lo que nos dice que numero de dígito es en realidad; las siguientes 784 variables son los niveles de gris de cada pixel en las imágenes. Las imágenes originalmente son bi-dimensionales y se han aplanado en un vector.

Podemos ver el conjunto de lo cargado desde MNIST (ver Tabla 1)

0	1	2	3	4	5	6	7	8	9
5923	6742	5958	6131	5842	5420	5918	6265	5851	5949

Tabla 1. *Los números de cada dígito en este subconjunto de datos*

Para visualizar los datos de la imagen, mostramos las primeras imágenes en el conjunto de datos con sus etiquetas. Las imágenes se normalizaron y se centraron.

Imagen 1. *Imágenes del conjunto de datos. Cada imagen tiene 28x28 pixeles con 255 niveles de gris*

Division de datos:

El conjunto de datos se dividió aleatoriamente en un conjunto de entrenamiento y un conjunto de prueba. El set de entrenamiento incluyen en el archivo *mnist_train.csv* y el set de prueba se incluyen el archivo *mnist_test.csv*

Para estudiar el algoritmo, en función del tamaño del entrenamiento, se crearon 4 conjuntos de datos del conjunto “train”, ellos incluyeron 20%, 40%, 60% y 80% de las imágenes del conjunto de entrenamiento, respectivamente. Al traficar el rendimiento del algoritmo a estos cinco conjuntos de catos (los 4 que creamos y el conjunto completo), podemos obtener una idea general de cómo el tamaño de estos influye en el rendimiento del algoritmo de aprendizaje.

Entrenamiento y prueba de datos

Ahora el conjunto de datos esta preparado, así que se empieza a entrenar y probar usando cada uno de los algoritmos de aprendizaje, lo que se utiliza para cada algoritmo es lo siguiente.

- Árbol de decisión: utilizamos el paquete ***rpart***
- Redes Neuronales: utilizamos el paquete ***nnet,*** y para potenciar utilizamos ***adarga***
- SVM: utilizamos el paquete ***kernlab***

Árboles de decisión

El nombre "***rpart***" significa "Recursive Partitioning and Regression Trees" (Partición Recursiva y Árboles de Regresión). Cuando el parámetro del método se establece en "*class*", puede funcionar en problemas de clasificación, rpart utiliza un conjunto de parámetros para controlar el crecimiento del árbol de decisión. Ajustamos el parámetro *"cp"* a 0.0 para lograr un árbol completo. Aunque el método tiene otras configuraciones predeterminadas para limitar el crecimiento del árbol: cómo el numero mínimo de observaciones n un nodo, la profundidad maxima de cualquier nodo, entre otros. Por lo tanto, establecer a cp en 0.0 puede o no dar un ajuste excesivo, pero se realizara para probar y usaremos el conjunto de datos de prueba para "podarlo".

Después del entrenamiento, obtuvimos un árbol con muchas divisiones, lo cual nos fue difícil mostrarlo con etiquetas, así que simplemente se mostró con las ramas sin esta etiquetas (Imagen 2).

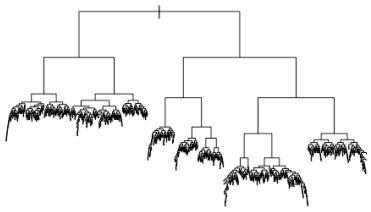

Imagen 2. *Arbol de decisión antes de "podar"*

Al comprobar el "*cptable",* encontramos que el árbol alcanzó su error mínimo, lo cual es una medida de error de validaciones internas cruzadas), en torno al tamaño 69. En realidad, el xerror tiene una disminución limitada después del tamaño 45. (Imagen 3)

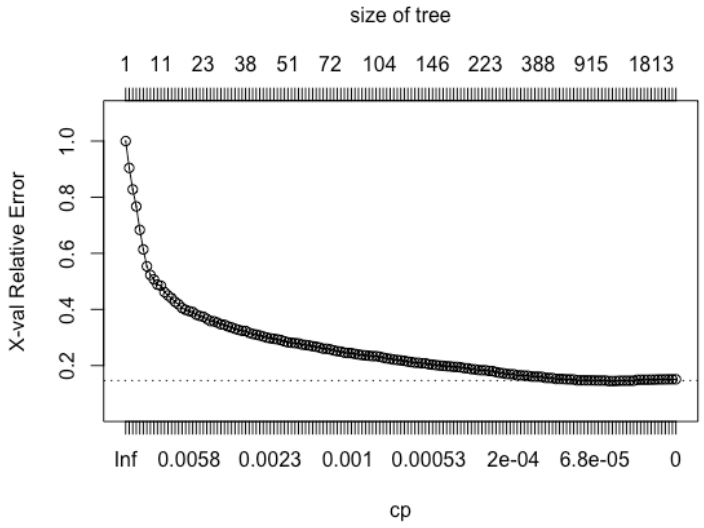

Imagen 3. *Xerror vs CP*

Luego se empezo a "podar" el árbol. Calculamos las tasas de error de las predicciones, basadas en los datos de prueba utilizando tamaños de árbol de 32 a 83, obtuvimos como resultado que el árbol con el tamaño 69 arrojo una mejor tasa de error en los datos de prueba (Imagen 4)

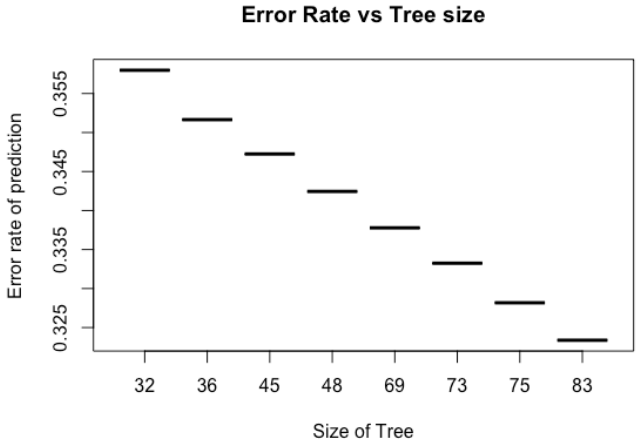

Imagen 4. *El tamaño del árbol se grafica a la tasa de error de predicción*

Luego de "podar" el árbol, se obtuvo uno ligeramente diferente con el primero (Imagen 5); la mejor tasa de error es 35.6 %. Tomó 2.88 para el entrenamiento y 0.11 para la prueba.

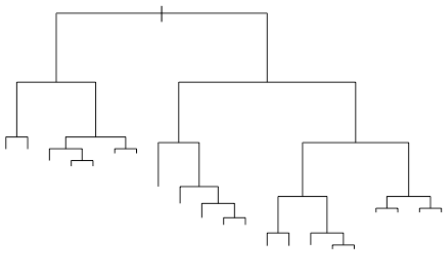

Imagen 5. *Árbol después de "podar"*

A continuación, se entreno el modelo con diferentes tamaños de conjunto de datos de entrenamiento, desde "train1" a "train4" y encontramos que mientras el tamaño de entrenamiento sea más grande, la precision de la predicción será mucho mejor (Imagen 6).

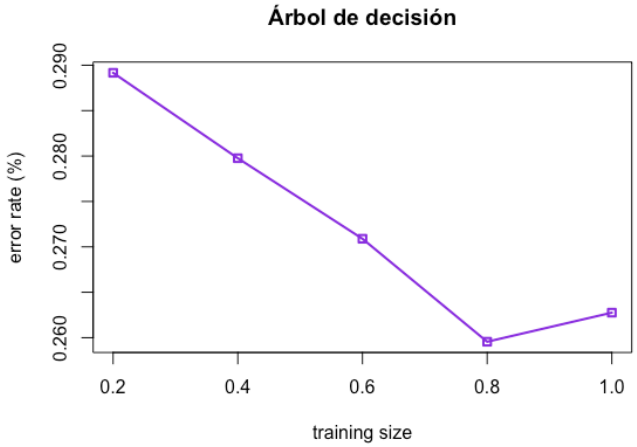

Imagen 6. *Árbol de decisión, el rendimiento es mejor con mayor tamaño de entrenamiento*

Redes Neuronales

El paquete "***nnet***" proporciona métodos para utilizar redes neuronales de avance con una sola capa oculta. En este proceso de entrenamiento, el numero de unidades en la capa oculta se establece en 15, el máximo de iteración se establece en 50, 100, 150 y la caída de peso se establece en 0.1. Este algoritmo consumo mucho tiempo, así que no se pudieron probar demasiados parámetros diferentes. Los demás parámetros se ajustaran automáticamente.

Este entrenamiento es lento y cuando la iteración maxima es larga, la tasa de error es menor, el resultado se puede observar en la Imagen 7.

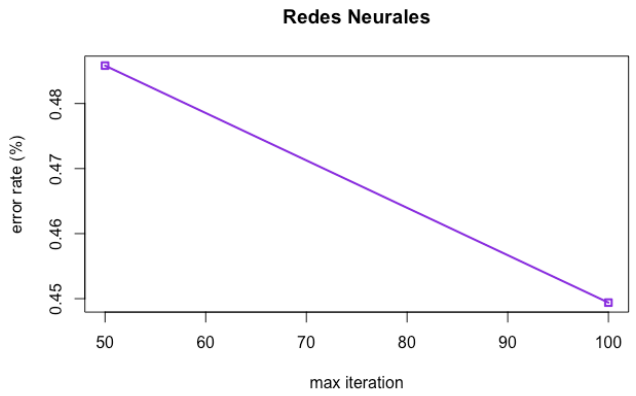

Imagen 7. *La tasa de error de las redes neurales disminuye con las interacciones máximas*

Máquinas de soporte vectorial

Para maquinas de soporte vectorial, utilizamos el paquete "***kernlab***" y probamos dos kernals, uno es un kernal lineal y el otro es Bernal gaussiano, para ambos se midió la tasa de error de la predicción en función del tamaño del entrenamiento. La imagen 10 representa los resultados

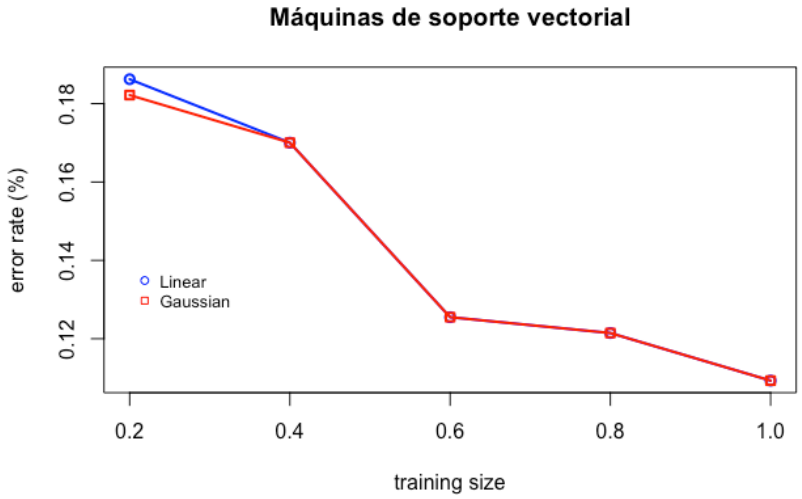

Imagen 8. *Las tasas de error se trazan a los tamaños de entrenamiento en dos kernal's diferentes*

Resultados y comparaciones

Hemos estudiado varios algoritmos de aprendizaje detalladamente en el conjunto de datos de dígitos. Compararemos estos en términos de tiempo de entrenamiento, tiempo de prueba y tiempo de error. Utilizamos los parámetros optimizados probados anteriormente para medir el tiempo de entrenamiento, el tiempo de prueba y la tasa de error en diferentes algoritmos de aprendizaje. Estos resultados los podemos observar en la imagen 11.

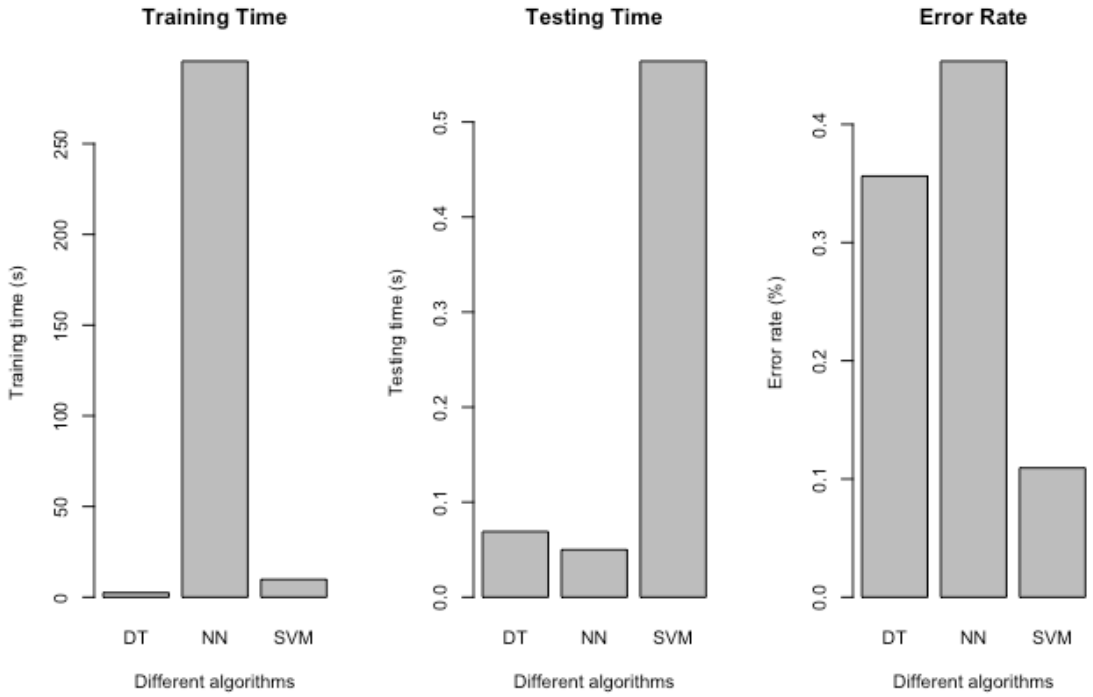

Imagen 9. *Comparación entre diferentes algoritmos de aprendizaje*

De la anterior grafica podemos concluir que:

- La red neuronal tiene un tiempo de entrenamiento muy mayor (1148 s) pero tiene el peor rendimiento (45.3 %)
- El árbol de decisiones sin impulso es fácil de entrenar y realizar las pruebas, pero el rendimiento no es optimo (35.6 % de tasa de error)
- El modelo SVM ofrece el mejor rendimiento en términos de tasa de error (10.9%) y el tiempo de entrenamiento es corto (23.8%). Toma un tiempo de prueba ligeramente largo (1.48 s) en el conjunto de prueba que en otros modelos.

Estas conclusiones pueden variar al momento de ajustar los modelos de una manera mas avanzadas, como por ejemplo ejecutar el modelo de red neuronal en mas iteraciones, esto nos puede dar un mejor rendimiento; sin embargo, estos resultados nos dan una idea general de los algoritmos en el conjunto de datos MNIST.

Referencias

- kernlab package: Author(s): Alexandros Karatzoglou, Alex Smola, Kurt Hornik
- nnet package: Author(s): Brian Ripley: ripley@stats.ox.ac.uk.
- rpart package: Author(s): Terry Therneau [aut], Beth Atkinson [aut], Brian Ripley [aut, trl, cre]