Facultad de Ciencias, Facultad de Minas
Universidad Nacional de Colombia, Sede Medellín
2019

STATISOFT

CLASIFICACIÓN DE IMÁGENES DE PERSONAS

Heber Esteban Bermúdez [1] John Bryan Yepez [2], Simon Zapata [3]

Resumen

En este informe se aborda la creación de un modelo de clasificación para predecir en imágenes de personas si estas tiene gafas de sol (gafas oscuras), como también la creación de una aplicación dinámica para la implementación del modelo.
Se crea una aplicación shiny en el lenguaje estadístico R, donde el usuario puede cargar una imagen de cualquier formato o tamaño ya sea a color o en escala de grises y la aplicación permite saber si la persona de la imagen tiene gafas de sol o no, como también se muestra la probabilidad hallada por el modelo para hacer esta clasificación.

Introducción

Para crear el modelo de clasificación de imágenes se decidió implementar un modelo de redes neuronales convolucionales en el software estadístico R, donde se hizo uso de la librería keras para este objetivo.
El modelo toma la imagen ingresada por el usuario y la redimensiona para convertirla en una matriz de 3 dimensiones, con dimensiones 100x100x3 (ancho y largo en píxeles, y las 3 componentes RGB) para ser evaluada en el modelo predictivo y arojar la calasificación. (Ver Metodología)

Definiciones

Red neuronal convolucional Una red neuronal convolucional es un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico. Este tipo de red es una variación de un perceptron multicapa, sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones

[1] hebermudezg@unal.edu.co
[2] jbyepezh@unal.edu.co
[3]sizapatagu@unal.edu.co

Aplicacion Shiny Es una herramienta para crear fácilmente aplicaciones web interactivas (apps) que permiten a los usuarios interactuar con sus datos sin tener que manipular el código. La programación Reactiva enfatiza el uso de: Valores que cambian en el tiempo.

Metodología

Para entrenar el modelo de redes neuronales convlucionales primero se hizo una partición el conjunto de datos dejando 420 imágenes para entrenar el modelo y las restantes 204 para validar el rendimiento del mismo, una vez obtenida las 420 imágenes de entrenamiento redimensionadas y transformada en matrices se pasa a un modelo de red neuronal secuencial, pero primero se prepara los datos combinando las matrices del de entrenamiento y validación y luego codificando los vectores en matrices de clase binaria mediante la to categorical de la función Keras como se muestra a continuación

```
> # codificando los datos de entrenamiento y validacion
> trainLabels <- to_categorical(trainy)
> testLabels <- to_categorical(testy)
```

El modelo se entrena con las matrices R, a continuación se muestra el código del modelo de entrada única con 2 clases (clasificación binaria), con 32 filtros y tamaño del kernel y la forma de entrada, se inicia con un modelo secuencial y luego se agregan capas usando el operador de tuberías y por último se compila el modelo con la función de pérdida, el optimizador adecuado.

```
> model <- keras_model_sequential()
> model %>%
+         layer_conv_2d(filters = 32,
+                       kernel_size = c(3,3),
+                       activation = 'relu',
+                       input_shape = c(100, 100, 3)) %>%
+         layer_conv_2d(filters = 32,
+                       kernel_size = c(3,3),
+                       activation = 'relu') %>%
+         layer_max_pooling_2d(pool_size = c(2,2)) %>%
+         layer_dropout(rate = 0.25) %>%
+         layer_conv_2d(filters = 64,
+                       kernel_size = c(3,3),
+                       activation = 'relu') %>%
+         layer_conv_2d(filters = 64,
+                       kernel_size = c(3,3),
```

```
+                         activation = 'relu') %>%
+          layer_max_pooling_2d(pool_size = c(2,2)) %>%
+          layer_dropout(rate = 0.25) %>%
+          layer_flatten() %>%
+          layer_dense(units = 256, activation = 'relu') %>%
+          layer_dropout(rate=0.25) %>%
+          layer_dense(units = 2, activation = 'softmax') %>%
+
+          compile(loss = 'categorical_crossentropy',
+                  optimizer = optimizer_sgd(lr = 0.01,
+                                            decay = 1e-6,
+                                            momentum = 0.0,
+                                            nesterov = T),
+                  metrics = c('accuracy'))
```

Se Usa la función fit para entrenar el modelo durante 1 épocas(repeticion) utilizando conjutnos de 32 imágenes:

```
> history <- model %>%
+          fit(test,
+              testLabels,
+              epochs = 1,
+              batch_size = 32,
+              validation_split = 0.2,
+              validation_data = list(train, trainLabels))
```

Diagnóstico

Para el diagnostico del modelo a continuación se muestra la matriz de confusión y la tasa de clasificación de imágenes correctamente clasificadas.

A continuación se muestra la matriz de confusión obtenida con los datos de entrenamiento.

Tabla 1: Matriz de confusión datos de entrenamiento

	0	1
0	208	3
1	4	205

A continuación se muestra la matriz de confusión obtenida con los datos de validación

Tabla 2: Matriz de confusión datos de validación

	0	1
0	101	0
1	0	103

Conclusiones

Los modelos implementados con redes neuronales covolucionales para la clasificación de imágenes tienen un excelente rendimiento y es muy recomendable usarlo en estos temas, se puede notar que en la matriz de confusión para los datos de validación obtuvo error cero.

NOTA 1: El código del este trabajo se puede consultar en el siguiente repositorio https://github.com/hebermudezg/TAE_Clasificacion_Imagenes

NOTA 2: La aplicación shiny se puede consultar en: https://colnalitycs.shinyapps.io/StatiSoft_clasificar_Imagen/

Referencias

[1] Redes neuronales con R- https://keras.rstudio.com/

[2] RStudio Team: (2019). Integrated Development for R. RStudio, Inc., Boston, MA URL http://www.rstudio.com/.