Proyecto #1

MSCS

*Teoría y Algoritmos para el aprendizaje automático*

*https://colab.research.google.com/drive/1bwvaA\_vZ6Uz3sCqwM5r8zw6HpH69vhAW#scrollTo=z-74\_gRc60Gl*

1. **Integrantes**

* Álvaro Flores Blas
* Carlos Palma Arzubialde

1. **Introducción.**

**El presente trabajo, implementará modelos de clasificación no supervisada para analizar un set de imágenes en blanco y negro que muestran caras haciendo expresiones. Para ello se utilizará la librería PyWavelets**

Entre los métodos de clasificación de entrenamiento supervisado, los más populares son Support Vector Machine (SVM), Decision Tree (DT) y K-Nearest Neighbors (KNN); las tres técnicas precisan de datos etiquetados y las tres tienen una versión adaptable a regresión.

1. **Explicación de los modelos.**

Se necesitó vectorizar 135 imágenes por 7 tipos de emoción, para ellos se utilizó una aproximación hiperbólica basada en el paper de RA DeVore, SV Konyagin and VN Temlyakov. “Hyperbolic wavelet approximation,”. Dicha aproximación logra descomponer la imagen filtrándola para que disminuya su escala y ruido (HAAR transformation). Con el componente principal “aproximado” y vectorizado se pasa a realizar las clasificaciones con los siguientes modelos.

El primer método SVM, este método divide los datos con un hiperplano con el fin de clasificar cuales caen en cada lado de esta división. El modelo es altamente adaptable, se puede definir cuan estricto es el margen de separación; si no permite que ningún dato quede dentro de los márgenes del vector se denomina como Hard margin y cuando si permite se denomina Soft margin, este último funciona mejor en el caso de los outliers. Otra adaptabilidad del modelo se da cuando los datos presentan un comportamiento polinomial y no es linealmente separable; para ello existe una modificación denominada el Kernel Trick que permite transformar los datos a dimensiones superiores.

En este ejemplo podemos observar una separación entre el pixel 0 y el pixel 8 como ejemplo.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

El método DT, parte de la idea de elegir valores particulares de las características para separar los datos; de tal manera que en un proceso repetitivo se logre refinar hasta obtener datos con las mismas etiquetas. Para separar los datos se debe entrenar el modelo con la ayuda de dos criterios que permitirán obtener los mejores resultados: el Gini Impurity y la Information Entropy.

Gini Impurity (Gini Gain):

El Gini Impurity es la probabilidad de clasificar incorrectamente un elemento elegido aleatoriamente en el set de datos, si fuese etiquetado aleatoriamente con alguna clase. Es calculado con la fórmula al final del párrafo; donde C es el número de clases y p(i) es la probabilidad de aleatoriamente elegir un elemento de la clase i.

Para encontrar las mejores particiones se parte del cálculo sobre la data sin separaciones y se va removiendo Gini Impurity a este cálculo inicial (se conoce como Gini Gain); para ello se hace un cálculo ponderado por cada elemento a clasificar. El objetivo es maximizar el Gini Gain.

Entropy (Information Gain):

La entropía es la medida de la impureza de la aleatoriedad de los datos. Una alta entropía significa baja pureza o alto desorden en los datos. Esta se mide con la fórmula al final del párrafo; donde C es el número de clases y p(i) es la probabilidad de aleatoriamente elegir un elemento de la clase i.

**El proceso de hallar las mejores particiones coincide con disminuir la entropía en el modelo añadiendo separaciones. Entonces el proceso parte de hacer el cálculo de la entropía inicial sin separaciones y ver como mejora con separaciones nuevas, a esto se le denomina Information Gain. En concepto es similar al Gini Gain.**

**Este es un ejemplo del árbol entrenado para este trabajo, no se limitó la altura del árbol.**

**Diagrama

Descripción generada automáticamente**

El último método utilizado es el KNN, se le considera el algoritmo holgazán porque técnicamente no entrena el modelo para hacer predicciones. En vez de eso predice la clase de una observación, basado en la proporción mas grande de K vecinos cercanos. La etiqueta predicha es la mas representada entre los vecinos, como si fue por votación. La vecindad suele ser definida por una distancia normalmente euclidiana y la cantidad de k vecinos elegida se determina en el entrenamiento, en lo que se denomina el método codo. Formalmente, se trata de encontrar la moda de las etiquetas de los k vecinos más cercanos.

Variando el número de vecinos encontramos que el modelo funciona mejor con un número bajo de vecinos. El mejor modelo sería para dos vecinos.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

1. **Experimentación.**

En cuanto a la experimentación empezamos con un modelo separado por 65% para el entrenamiento y 35% para la prueba; esta es una separación clásica que podría generar problemas por no contar con muchos datos.

Los resultados en este caso nos indican los siguientes resultados de buena predicción:

SVM: 98%, DT: 83% y KNN (2 vecinos): 72%.

Para resolver el problema de pocos datos se implementó el modelo de cross validation usando 10 hojas de separación (K-Fold Cross Validation).

Los resultados en este caso nos indican los siguientes resultados de buena predicción:

SVM: 96%, DT(Gini): 86%, DT(Entropy): 86%,

KNN (2 vecinos y distancia Manhattan y Euclideana): 91% y 91%

KNN (3 vecinos y distancia Manhattan y Euclideana): 86% y 86%

Bootstrapping es otra metodología que puede ser utilizada en caso de tener pocos datos. En este caso implementamos un programa que nos permitía armar sets de experimentación seleccionando elementos con reemplazo para el entrenamiento, se realizó 10 repeticiones.

Los resultados obtenidos en este caso fueron los siguientes:

SVM: 95%, DT(Gini): 76%, DT(Entropy): 76%,

KNN (2 vecinos y distancia Manhattan y Euclideana): 65% y 64%

KNN (3 vecinos y distancia Manhattan y Euclideana): 55% y 55%

1. **Conclusiones.**

El trabajo no permitió experimentar con distintas aristas de los modelos previamente estudiados y poder encontrar cual tiene mayor poder de predicción.

Para este dataset pudimos encontrar que los modelos obtenían mejor predicción al ser entrenados y testeados con el modelo de K-Fold cross validation (10 hojas). Bajo este proceso los modelos de clasificación casi en todos los casos ganaron poder de predicción, salvo SVM que ya contaba con un alto poder de predicción.

Debemos notar que entre los modelos de Machine Learning utilizados en este trabajo el modelo de SVM con soft margins fue el que tuvo mejor ajuste.

Para futuros trabajos se podría experimentar utilizando el kernel trick para el modelo SVM.