Momento de Retroalimentación

Módulo 2. Análisis y Reporte sobre el desempeño del modelo.

(Portafolio Análisis)

Implementación de un modelo de clasificación de Perceptrón usando la librería Scikit-Learn.

Hayali Monserrat Marina Garduño, A01751188

Inteligencia Avanzada para la ciencia de datos I (TC3006C.101)

Introducción

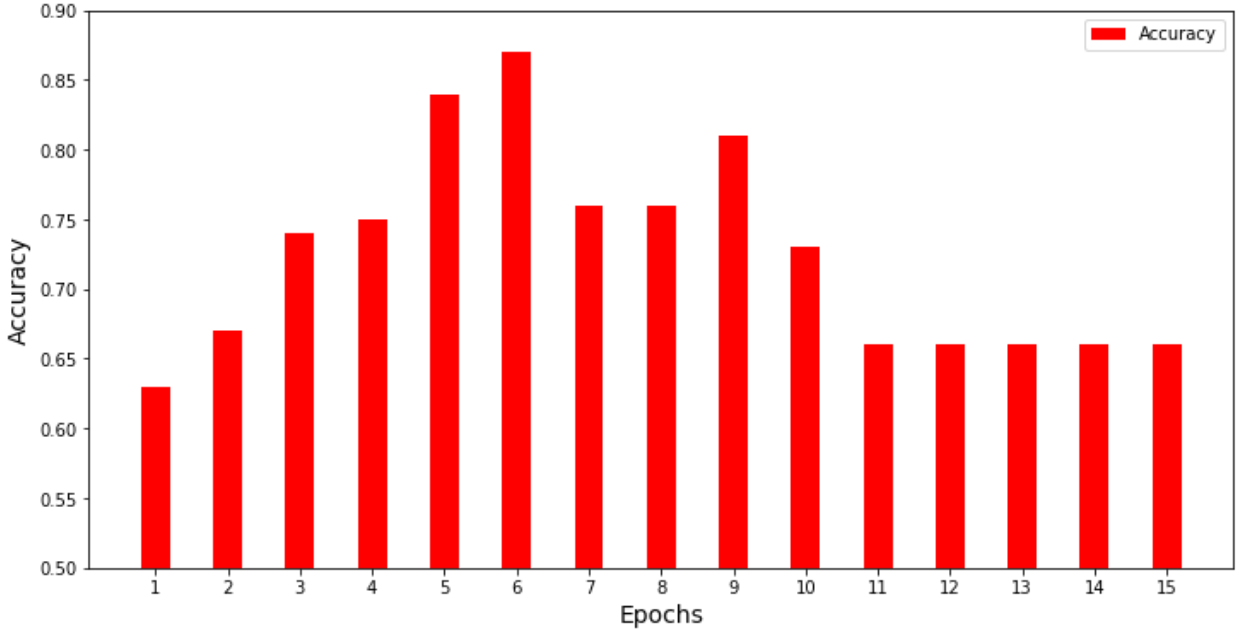
En este reporte se documentará el desarrollo de un modelo de clasificación utilizando el algoritmo de un Perceptrón. De forma muy general, un Perceptrón utiliza el algoritmo de backwards propagation para ajustar y modificar pesos que a su vez modifican el resultado final de una combinación lineal entre los mismos, y las entradas que se le asignan al perceptrón. Esta combinación luego se pasa por una función de activación para obtener nuestra clasificación. A medida que los pesos se ajustan utilizando gradientes, el error resultado final con respecto al nuestro valor esperado disminuye.

El perceptrón es un algoritmo de clasificación considerado de aprendizaje supervisado, pues requiere que los simples que ingresamos al algoritmo estén previamente clasificados correctamente para poder calcular el error y como mencionamos anteriormente, en respuesta poder ajustar los pesos.

Desarrollo

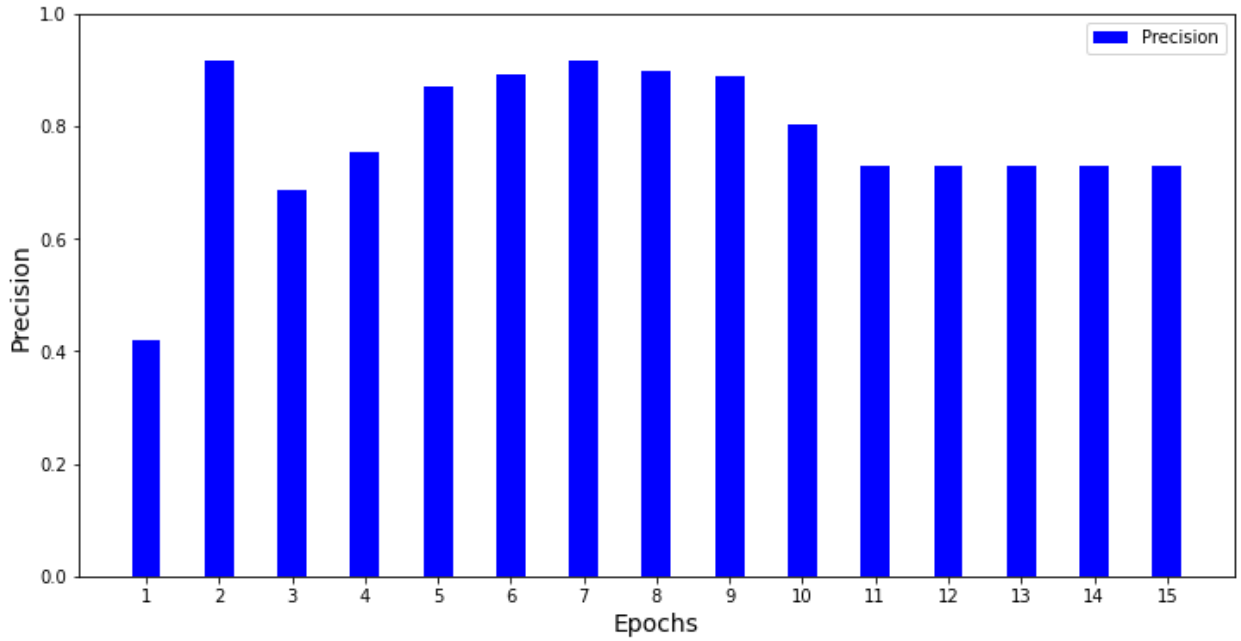
La primera parte al desarrollar el Perceptrón fue crear nuestro dataset utilizando la biblioteca de Scikit Learn. Generamos un dataset con 100 elementos, cada uno definido por dos componentes o features, y divididos en dos clases o categorías.

Después definimos nuestro clasificador Perceptrón igualmente con la biblioteca de Scikit Learn pero restringiendo el numero de iteraciones que el algoritmo daba al dataset. Lo anterior se hizo con el objetivo de poder correr diferentes tests de accuracy y otras métricas para así poder visualizar los efectos del overfitting y underfitting.

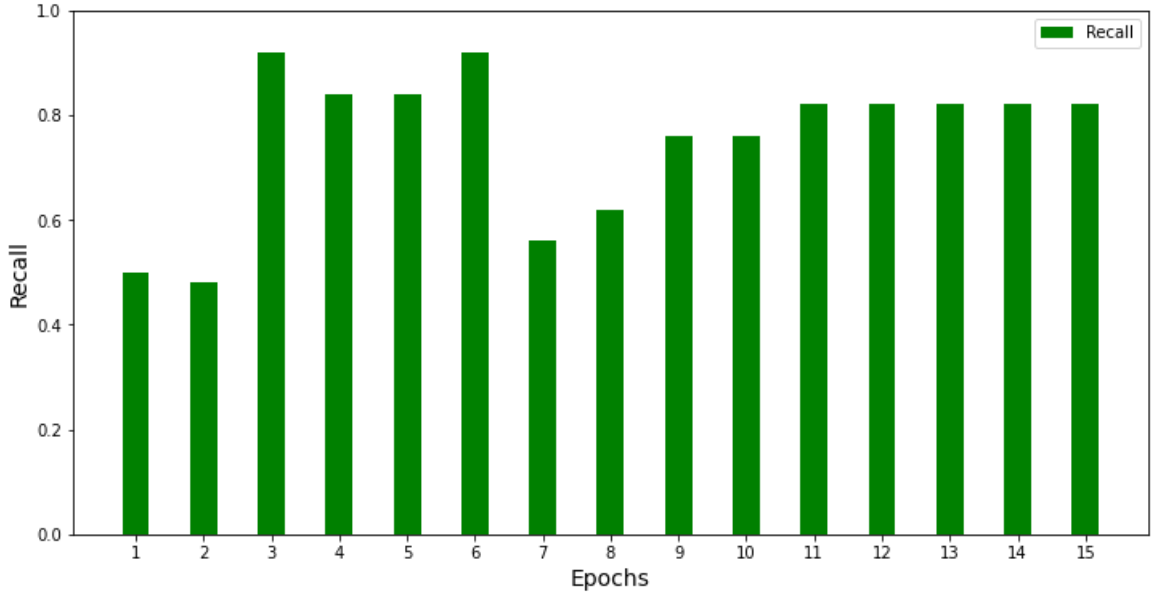


De esta forma, podemos apreciar la primera gráfica donde se muestra el cambio en el accuracy conforme aumentamos el numero de epochs. Es particularmente interesante y notable como el accuracy aumenta rápidamente desde menos del 65% hasta mas del 85% con un epoch=6. Sin

embargo, desde ese punto en adelante el accuracy baja hasta de nuevo el 65% en una muy notoria muestra de underfitting y overfitting. En especial es muy importante identificar estas tendencias, pues es muy fácil caer en un proceso de overfitting en donde se comienzan a ignorar las generalidades aprendidas por el modelo, en favor de particularidades de nuestro subset de training haciendo que aumente el accuracy al probar con el mismo subset de training, pero disminuyendo al probarlo con un subset de testing con información nueva.



De forma similar, al graficar los promedios de precisión con diferentes valores de epochs podemos notar la misma tendencia en donde inicialmente la precisión aumenta, en este caso hasta el apoch 8, punto a partir de donde la precisión comienza a disminuir en una muestra del mismo efecto de overfitting. En este caso el punto máximo de precisión se alcanzó con un epoch = 6.



Finalmente podemos también notar los efectos del overfitting al analizar nuestro recall. Si bien en este caso los resultados están ligeramente mas dispersos y la tendencia anteriormente mencionada es menos visible, aún así podemos notar como el recall varía conforme cambia el número de epochs.

Algo muy importante a notar, fue el uso de la técnica de validación cruzada con el que podemos utilizar todo el espectro de nuestro dataset minimizando así nuestro potencial bias. Para ello se utilizó la misma librería de SKLearn y en cada iteración se promediaron los resultados de los folds de la validación cruzada.

De esta forma, se obtuvo primero un registro de como varía el accuracy, precisión y recall con diferentes valores de epoch, concluyendo que un valor de 6 epochs nos da los mejores resultados.

Finalmente se corrió de nuevo el entrenamiento del modelo con 6 epochs y validación cruzada, confirmando los siguientes resultados:

```
Accuracy promedio: 87.0% con desviación estandar de 0.14
Precisión promedio: 89.29% con desviación estandar de 0.17
Recall promedio: 92.0% con desviación estandar de 0.07
```

Como se puede ver, tanto en accuracy, como en precisión y recall, los resultados son bastantes buenos, alrededor del 90%. Pero más notable aún, es que se presentan desviaciones estándar bastante baja lo que nos indica que esos resultados son bastante consistentes y estables.

Conclusión

Para finalizar este reporte, puedo concluir que este proyecto fue especialmente útil al mostrar el funcionamiento de un perceptrón y su desempeño. De la misma forma, se logró trabajar con métricas para medir el desempeño de nuestro modelo tales como accuracy, precisión y recall. Considerando las similitudes que un Perceptrón tiene con otro tipo de modelos mas complejos como una red neuronal, considero que fue muy útil poder ahondar mas en el desarrollo y desempeño de un modelo como este. Finalmente considero que en un futuro proyecto de seguimiento valdría la pena explorar mas a profundidad las limitaciones de este tipo de algoritmos por ejemplo con clasificaciones no binarias o dataset donde por su construcción una división con un hiperplano no sea ideal.