**Introducción al machine learning, perceptrón y su modelo mejorado Adaline(ADAptative LINear Element).**

Tratemos de entender primero cómo funciona el cerebro humano, para poder entender una máquina artificial de aprendizaje. Mc Cullak y Pitts descubrieron una célula nerviosa como una simple puerta lógica con salidas binarias, múltiples señales llegan a las dendritas, a continuación, se integran en el cuerpo de la célula y, si la señal acumulada supera un umbral determinado, se genera una señal de salida que será transmitida por el axón**(ver imagen)**

Diagrama

Descripción generada automáticamente

Solo unos años después, Frank Rosenblat publicó el primer concepto de regla de aprendizaje del perceptrón, basado en el modelo de la neurona, con esta regla del perceptrón, Rosenblatt propuso un algoritmo que podía automáticamente aprender los coeficientes de peso óptimo que luego se multiplican con las características de entrada para tomar la decisión si una neurona se activa o no.

Volviendo al machine learning, veremos que un algoritmo como este puede utilizarse para predecir si un objeto de un conjunto pertenece a una etiqueta o a otra, es decir, este primer modelo propuesto por Frank Rosenblat puede clasificar binariamente a un conjunto.

**Definición formal del primer modelo de una neurona artificial perceptrón.**

Clasificaremos binariamente a un conjunto a través de hacer clase a 2 clases 1(clase positiva) y -1(clase negativa). También podemos definir una función de decisión

, donde es el vector de pesos y es el vector de características o los valores de entrada y es la entrada de la red. Ahora si la entrada de red de una muestra concreta es mayor que un umbral definido , la clase pertenece a 1(clase positiva). Para definir esto elegimos la función de decisión de la siguiente manera:

Para que podamos utilizar el umbral definido, definimos a y a de tal manera que cuando realicemos el producto punto, aparezca (el umbral) como termino constante en la suma. Así se cumple que si la ‘’la suma restante del producto punto’’ es mayor que (la suma total es mayor o igual a 0) y se clasifica en la etiqueta 1(clase positiva), y si es negativa(la suma total es menor que 0) se clasifica en la etiqueta -1(clase negativa). Con lo anterior podemos clasificar a las etiquetas de acuerdo a si los pesos superan o no a un umbral, habitualmente a se le denomina habitualmente sesgo.

Pasemos a definir ahora **la regla de aprendizaje del perceptrón.** La idea general que hay detrás de la neurona y del modelo del perceptrón umbralizado es utilizar un enfoque reduccionista para imitar de manera simple como trabaja una neurona en el cerebro, si se excita o no. Los pasos se pueden resumir de la siguiente manera:

1. Iniciar los pesos a 0
2. Para cada muestra de entrenamiento:
   1. Calcular el valor de salida(lo predice )
   2. Actualice los pesos

La actualización de pesos, para cada peso se define como (:

Note que, en caso de una predicción errónea, los pesos se verán empujados a aumentar o a descender según si está abajo o arriba de la etiqueta verdadera, es de notar también que la actualización de peso i es proporcional a la característica

**De manera gráfica y resumiendo el perceptrón tiene el siguiente algoritmo:**

Diagrama

Descripción generada automáticamente

Antes de proseguir con un algoritmo de clasificación más potente, es de notar la convergencia del perceptrón, ya que esta misma solo está garantizada si las 2 clases son linealmente separables**(ver imagen)** por un hiperplano y si el rango de aprendizaje es lo suficientemente pequeño, esto lo demostró matemáticamente Rosenblat. Si los datos no son linealmente separables por un hiperplano, podemos decidir un número de pasos máximos sobre el conjunto de datos de entrenamiento(épocas) y/o un umbral para el número de clasificaciones erróneas toleradas. De otro modo el perceptrón nunca deja de actualizar los pesos.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

**Neuronas lineales adaptativas(ADALINE)**

Veamos otro modelo de red neuronal de capa única, las neuronas lineales adaptativas, en inglés ADAptative LINear Element. Adaline fue publicado por Bernard Widrow y su alumno Ted Hoff, pocos años después de que se haya creado el perceptrón.

El algoritmo es especialmente útil porque puede minimizar una función de coste de una función continua, a diferencia del modelo del perceptrón. Esto sienta las bases para algoritmos más avanzados como la regresión logística, máquinas de vectores de soporte etc.

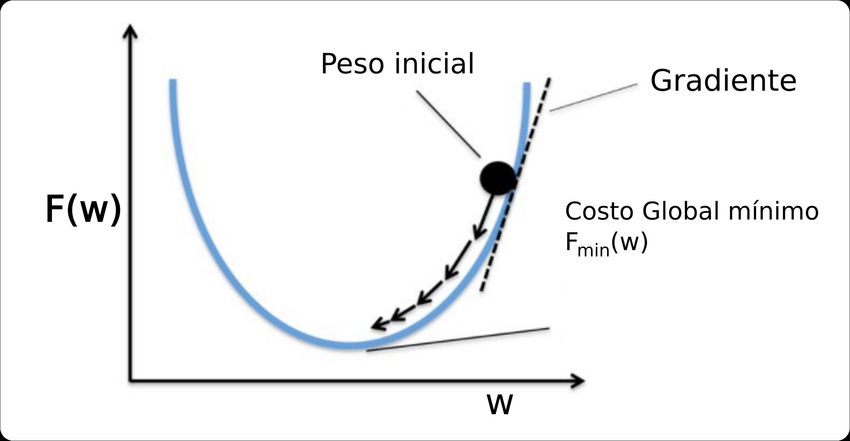
La diferencia es que los pesos se actualizan con una función continua a través de descenso de gradiente, para minimizar una función arbitraría llamada coste.

Podemos definir la función de coste como la mitad de la suma de los errores cuadráticos asociada al erro de clasificación de cada muestra i, el valor predicho se hace con la función definida anteriormente:

El término será de utilidad después para poder obtener una forma más simple del gradiente de esta función.

Aclarado esto expliquemos brevemente la idea del descenso por gradiente. La idea del descenso por gradiente es ir en la dirección de menor crecimiento de la curva, esta dirección es la dirección contraria al gradiente de la función, de esta manera podemos ir colina abajo para poder hallar el mínimo de la función que deseamos minimizar

Para que el método pueda funcionar necesitamos que los ‘’pasos’’ hacia la colina sean de poca longitud y además necesitamos que la función sea convexa . Si los pasos son muy largos podríamos sobrepasar el mínimo y tener un resultado erróneo, y si no es convexa la convergencia no está garantizada (ver imagen)



De manera análoga al perceptrón la actualización de pesos, para cada peso se define como (:

Donde la componente j del gradiente es:

La demostración del gradiente de la función de costos es así:

Texto, Carta

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

De forma vectorial la anterior ecuación del gradiente queda:

La idea de este algoritmo es minimizar mediante el gradiente descendiente, la función de error definida por el costo, este método es más eficiente y al utilizar una función continua que se quiere minimizar numéricamente, funciona evidentemente mejor que la actualización de pesos definida por el perceptrón ordinario.

Para mejorar la rapidez de la convergencia, podemos seguir un proceso de normalización para poder dotar a nuestros datos de una distribución normal estándar, este proceso cambia la media a 0 y la desviación estándar a 1. Para esto definimos un cambio de variable para cada dato x como sigue:

Una de las razones por las cuales esta normalización nos ayuda a mejorar el gradiente descendiente es que ayuda a converger más rápidamente al gradiente descendiente. Se tomará lo anterior como un hecho, los detalles matemáticos de este método son complicados y no es el tema principal del texto.

El método de minimizar la función de coste a través del gradiente descendiente es optimo para una cantidad de datos pequeñas, sin embargo, para grandes datos necesitamos utilizar un método con menos complejidad temporal, para esto se utiliza un método modificado, llamado gradiente descendiente estocástico. Este método en lugar de actualizar los pesos en base a la suma cuadrática de los errores, se actualiza los pesos de forma incremental para cada muestra de entrenamiento así:

Aunque el gradiente descendiente se puede considerar como una aproximación al descenso de gradiente, normalmente consigue la convergencia mucho más rápido debido a que las actualizaciones de pesos son más frecuentes. Al actualizarse los pesos a través de un solo ejemplo de comparación, el error al principio es mayor que el descenso por gradiente. El gradiente descendiente estocástico tiene la ventaja de que puede escapar de mínimos pocos profundos más fácilmente que el gradiente descendiente.

Para implementar Adaline, debemos presentar los datos en orden aleatorio; adicionalmente el rango de aprendizaje fijado se sustituye por un rango variable función del número de iteraciones

Donde son constantes. Con el rango de aprendizaje adaptativo podemos conseguir mejores resultados hacia el coste mínimo.

Para finalizar, se ha de aclarar que los anteriores modelos son modelos de una sola neurona y solo pueden clasificar binariamente a un conjunto; sin embargo, hay generalizaciones multiclase y de igual manera hay modelos multicapa y con algoritmos de optimización variados.