Inteligencia artificial

->reconocimiento de patrones (eso es lo que quiero hacer)

***Machine Learning***

Paradigma del aprendizaje:

Aprendizaje supervisado, no supervisado, reforzado

Técnicas para diferentes tipos de aplicaciones:

árbol de decisión, modelo de regresión, de clasificación, técnica de cluster, redes neuronales

Redes neuronales: aprendizaje de forma jerarquizada del mismo, en las primeras capas se aprenden conceptos más abstractos y en las capas superiores se obtienen resultados más concretos. ¿Cuántas capas debo usar? Por lo general se tiene a aumentar a medida que se incrementa la complejidad de los resultados esperados, esto da pie al Deep Learning

**Aprendizaje supervisado y no supervisado**

El aprendizaje consiste en encontrar una relación entre (variables de entrada y las variables en la salida) los datos que ingresan con los datos que tenemos a la salida.

El aprendizaje supervisado es el cual se le señala claramente a la “caja negra de momento” que es lo que debería obtener a la salida y esta debería resolver el dilema entregándonos posteriormente la solución a datos que no se le entrego anteriormente.

**Modelos**

Reconstrucción simplificada que nos permite representar la realidad, mapas, diagramas, partituras.

Probabilidad

Reducir la complejidad de un fenómeno en base a los datos, el uso de probabilidad nos permite generar modelos probabilísticos, que nos permiten manejar grandes cantidades de información y interpretarla de forma que no sea tan complicado usar estos modelos.

-Modelos

Se pueden ajustar los modelos con el objetivo de poder representar de mejor manera los datos que observamos, (En mi caso, lo que pretendo es que los datos obtenidos por parte de las antenas se relacionen de alguna manera tal que me permitan anticiparme a variaciones de las mismas), como revisamos si nuestro modelo es correcto, es usar el error de los datos

-Datos: de donde extraemos la información del modelo (multidimensional, si mi posición, potencia, ángulos, son necesarios el uso de las matemáticas para poder representar la realidad de los datos obtenidos.

-parámetros: son los valores que puedo modificar dentro de mi modelo con el fin de poder generar una mejor representación de la realidad, (permite ajustarnos a los datos)

-error: definir una función de error esto nos permite ver que tanto nuestro modelo se acerca a la realidad que queremos interpretar, en el caso del aprendizaje supervisado se revisa en base a los datos a la salida del modelo y en el del no supervisado se obtienen de los datos a la entrada del modelo. Reajustando el modelo (optimización)

**Regresión lineal y mínimos cuadrados**

Modelo de regresión simple

Planos y hiperplanos que representen estos datos

Representación vectorial -> con matrices

La gpu esta acostumbradas a trabajar con matrices, modelos se entrenan más eficientemente

Recta que mejor hace esto, pero de forma automática, como lo hacemos

Mínimo cuadrados ordinarios, (Uso del error)

El error cuadrático medio

Gráfico

Descripción generada automáticamente

El error cuadrático medio

Mientras que el error cuadrático medio vectorial

Pero si aplicamos la derivada obtenemos con respecto a

Mínimo error cuadrático medio

De esta forma minimizamos la ecuación de coste, pero lo que a mi me preocupa es encontrar esa matriz inversa u.u

Es aquí donde aparece el método del gradiente :D

**Método del Gradiente**

La función de coste se basa en las propiedades de las funciones cóncavas, convexa y no convexa.

Buscamos los mínimos globales -> es fácil en las funciones convexa porque solo hay única solución

Pero tenemos en este caso funciones no convexa, donde encontraremos múltiples mínimos locales y globales. En nuestro caso tenemos que buscar desde un punto si la pendiente es negativa, si existe un cambio de pendiente, tenemos un mínimo encontrado.

Usaremos derivadas parciales de esta función multidimensional de la cual obtendremos una pendiente, una matriz de derivadas parciales denominada gradiente, repetimos este proceso varias veces hasta que obtener la convergencia de la misma

Diagrama, Gráfico de superficie

Descripción generada automáticamente

Siendo el radio de aprendizaje define cuanto afecta el gradiente a la actualización de nuestro parámetro en el aprendizaje

De no escogerse apropiadamente este parámetro es posible encontrarnos con un sistema que aprenda demasiado lento o quedar atrapado en un mínimo local (demasiado pequeño) o de ser muy grande es posible que sea incapaz de converger a el mínimo, y en el peor de los casos es de que quede en un bucle infinito.

Gráfico, Gráfico de superficie

Descripción generada automáticamente

Técnicas para mejorar la modificación dinámicamente los radios de aprendizaje.

**Redes neuronales**

Parte más básica de una red neuronal (unidad de procesamiento), en este caso es denominado neurona, consiste en una suma ponderada, a la cual, se le asignan a los valores de entrada los respectivos pesos, que son el grado de relevancia dentro del cálculo, también se incluye una variable independiente denominada sesgo o bias, el cual nos permite mover el hiperplano

Diagrama, Esquemático

Descripción generada automáticamente

Como en el ejemplo de nachos, lo que yo tengo en mi caso a la entrada son variables numéricas y variables binarias y a la salida en el primer caso tendría una variable binaria, si el valor es correcto o no, pero en este caso no podría representarlo pues el valor obtenido es un.

Si lo que quiero es obtener este valor de si o no es usar un umbral con la variable del error en este caso tenemos

Ahora si le asignamos al sesgo el valor inverso del umbral tendremos

Para poder resolver problemas más complejos es necesario añadir más neuronas

Falta una última componente La función de activación->

**Las redes**

Diagrama

Descripción generada automáticamente**+**

El uso de capaz nos permite un aprendizaje jerarquizado, donde obtendremos a la salida una salida más compleja que solamente un dato.

Arquitectura de la red <- es necesario descubrir algo más complejo (Aprendizaje profundo)

Pero en este caso al tener ecuaciones lineales, al operar con ellas infinitamente, el resultado al ser lineal es posible simplificarlo en una función lineal más simple, similar a haber usado una única neurona, para evitar esto es que se les añade un comportamiento no lineal a las funciones, esta a su vez se comportan de manera diferente, evitando que se puedan simplificar a una sola neurona, esta componente no lineal se denomina

Función de activación

Añade una deformación de lo anterior

Diagrama

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamenteImagen que contiene Diagrama

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**Backpropagation**

Que sucede si junto neuronas sin las no linealidades mencionadas anteriormente, ocurre que la linealidad de todas las neuronas de la misma estructura es equivalente a tener una sola neurona.

Lo que queremos es un algoritmo que modifique sus parámetros de forma automática, este es denominado como Backpropagation. El problema de usar directamente el descenso del gradiente, es que debemos calcular la matriz de las derivadas parciales de las pendientes (el problema surge al intentar calcular el coste de un peso, este a su vez influye dentro de la red de varias maneras, alterando sus valores y por tanto obteniendo derivadas parciales dependientes de múltiples caminos :c)

Vamos a calcular el descenso del gradiente(optimización de la función de coste del gradiente) por medio del backpropagation, pero que diantres es eso?????

Es revisar la retropropagación de errores, revisando desde el resultado final, hacia las neuronas anteriores y determinar que tanto influyeron en el resultado obtenido, de ser cierto este error es necesario implementar la corrección correspondiente.

*Imagen del café*

Matemáticas -:c

Underfitting (sub ajustado), es cuando nuestra red neuronal no posee un entrenamiento adecuado o su entrenamiento da como resultado un sistema que no obtiene buenos resultados. Un modelo bien ajustado debería dar buenos resultados??? La respuesta es que no debido a que no necesariamente es capaz de generalizar el aprendizaje adquirido.

Overfitting (sobre ajuste) ahora que sucede si sobre entrenamos a la red, sucede que empezamos a generalizar el ruido :O, se especializa en aprender la información que le entrega.

Que sucede u.u como resolvemos este problema, como lo detectamos.

Modelo de clasificación logístico

Modelos de los k-nn vecinos cercanos

Modelos de clusterizacion K-means

¿Cómo podemos identificarlo?

Visualmente, la frontera de decisión se ajusta mucho a los datos, o la línea de regresión es muy flexible adaptándose a los datos, en dos o tres dimensiones.

Con los datos podemos separar los datos en datos de entrenamiento y de validadción 80% -20% y revisamos el error (Estrategia denominada Hold Out)

Debo hacer que los datos estén distribuidos pero de forma idéntica, no es valido usar datos que tengan demasiada similitud

Y que los datos no sean dependientes, para que no se oculte el overfitting

Al usar librerías que calculen el error ellas estarán usando grafos computacionales (que es eso, ni idea u.u)

En TensorFlow los puntos de entrada de los datos del grafo computacional se denominan Placeholder