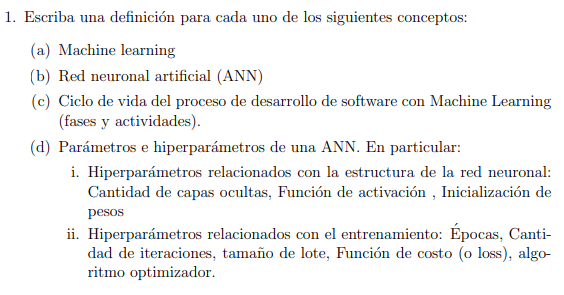
*Trabajo Práctico 8*



a)

El ML es un enfoque de la IA (no simbólica) que trata de generalizar comportamientos a partir de una información suministrada en forma de ejemplos (un proceso de inducción del conocimiento), en donde la máquina construye (aprende) su propio algoritmo para resolver un problema.

b)

Las redes neuronales son un modelo computacional basado en un gran conjunto de unidades neuronales simples (neuronas artificiales). Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida.

c)

**Definición de necesidades**

El primer paso para el aprendizaje automático consiste en definir un objetivo, es decir, plantear un problema que requiera una solución a medio-largo plazo.

Existen diferentes formas de estructurar el objetivo de un proyecto en el que se va a implementar tecnología Machine Learning pero siempre hay que establecerlo en base a las necesidades empresariales y a las posibilidades que tiene la compañía en función de los datos de los que disponemos.

**Conseguir los datos**

Dada la problemática que se quiere resolver, se investigará y obtendrán datos que se utilizarán para alimentar la máquina. Importa mucho la calidad y cantidad de información que se consiga, ya que impactará directamente en lo bien o mal que funcione el modelo.

**Preparar/normalizar los datos**

Es importante mezclar “las cartas” que se obtengan, ya que el orden en que se procesen los datos dentro de la máquina no debe de ser determinante.Se harán visualizaciones de los datos y revisarán si hay correlaciones entre las distintas características obtenidas. Se seleccionarán las características, las cuales impactarán directamente en los tiempos de ejecución y en los resultados, también se puede hacer reducción de dimensiones aplicando PCA si fuera necesario. Se deberá tener balanceada la cantidad de datos que hay en cada resultado(clase), para que sea representativo, ya que si no, el aprendizaje podrá ser tendencioso hacia un tipo de respuesta y cuando el modelo intente generalizar el conocimiento fallará.

Se separarán los datos en dos grupos: uno para entrenamiento y otro para evaluación del modelo. Podemos fraccionar aproximadamente en una proporción de 80/20 pero puede variar según el caso y el volumen de datos que se tenga.

**Crear el modelo y entrenarlo**

Una vez preprocesado los datos nos corresponde elegir el algoritmo más adecuado en relación al problema que deseamos resolver.

Es aquí donde debemos optar por un algoritmo de aprendizaje supervisado o un aprendizaje no supervisado.

Dentro de los algoritmos de aprendizaje supervisado tenemos: Linear Regression, Logistic Regression, Decision Tree Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree Classification, etc.

Dentro de los algoritmos de aprendizaje no supervisado tenemos: K-Means Clustering, Hierarchical Clustering, Principal Component Analysis, etc.

Elegido el algoritmo procedemos a separar los datos preprocesados. Un porcentaje de la data, comúnmente el 70 % del total, la utilizaremos como data de entrenamiento.

Es decir, será aquella a la cual aplicaremos el algoritmo seleccionado. Sobre esta data buscaremos alcanzar el objetivo planteado inicialmente.

**Evaluar el modelo y re-entrenarlo**

Dado que ya tenemos el modelo entrenado, lo siguiente es evaluarlo. Esto lo realizaremos con los datos restantes, aquella que no utilizamos para el entrenamiento. La cual llamaremos datos de evaluación.

Sobre los datos de evaluación procederemos a correr el algoritmo y a evaluar los resultados obtenidos.

Debemos ser conscientes que existe la posibilidad que el modelo funcione bien para los datos de entrenamiento y no para los datos de validación (problema de overfitting).

Por lo tanto volveremos a re-entrenar nuestro modelo hasta que se ajuste bien a las dos particiones (data de entrenamiento y data de validación). Todo esto con el propósito de ganar confianza en nuestro modelo.

**Despliegue**

En este punto ya se consiguió el porcentaje de exactitud deseado, por lo que es momento de utilizar el modelo para realizar predicciones reales en la vida real.

**Mantenimiento y evolución**

Tras la puesta en marcha del modelo de Machine Learning pueden darse múltiples situaciones que den lugar a equívocos. Por ello, el análisis de errores (Mantenimiento) es la última fase, aquella que permite modelar y cambiar los aspectos no relevantes para el proyecto.

Este análisis permitirá mejorar el rendimiento del modelo de aprendizaje automático y profundizar aún más en las fases previas a la implementación y entrenamiento.

Además, con el análisis de errores, obtendremos las claves de los fallos durante el proceso y generaremos nuevas conclusiones sobre qué necesita el proyecto.

**Vuelta a la fase 1)**

d)

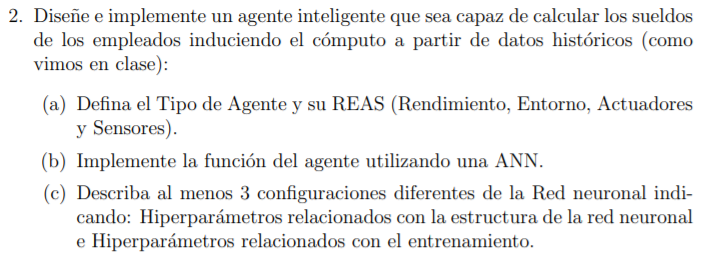
* Los parámetros del modelo son los pesos y los umbrales. El objetivo del entrenamiento es aprender los valores de estos parámetros.
* Los hiperparámetros son parámetros externos establecidos por el operador de la red neuronal.

Hiperparámetros relacionados con la estructura de la red neuronal:

* Cantidad de capas ocultas: una red neuronal puede tener varias capas ocultas, cada una de estas capas contiene neuronas que están conectadas con las neuronas de las siguientes capas.
* Función de activación: las funciones de activación se encuentran en cada neurona e indican cuando una neurona se activa o se apaga. Dependiendo de la función de activación que se use, esta tendrá ciertos límites, recordemos que primero se calcula la función Z = WX + b y el resultado de esta función se le pasa a la función de activación, esta última busca si los datos tienen los patrones que busca la neurona o no los tiene.
* Inicialización de pesos: es la asignación inicial de los pesos, hay varias posibilidades de inicialización siendo las más comunes las que introducen valores aleatorios pequeños.

Hiperparametros relacionados con el entrenamiento:

* Épocas: este es el número de veces que se ejecutarán los algoritmos de forwardpropagation y backpropagation. En cada ciclo (epoch o época) todos los datos de entrenamiento pasan por la red neuronal para que esta aprenda sobre ellos.
* Cantidad de iteraciones: significa que al final de cada ronda, tenemos que calcular la tasa de precisión. Cuando la tasa de precisión ya no mejora, termina el cálculo y determina el número de iteraciones.
* Tamaño de lote: es el número de datos que tiene cada iteración de un ciclo (epoch), esto es útil porqué la red neuronal actualiza los parámetros W y b más veces, también cuando se tienen grandes cantidades de datos se necesitan computadoras con más memoria y la red neuronal tarda más en ejecutar cada ciclo, si dividimos los ciclos en iteraciones con un número de datos más pequeño ya no es necesario cargar todos los datos en la memoria al mismo tiempo y la red neuronal se entrena más rápido.
* Función de costo (o loss): la función de pérdida, también conocida como función de costo, es la función que nos dice que tan buena es la red neuronal, un resultado alto indica que la red neuronal tiene un desempeño pobre y un resultado bajo indica que la red neuronal está haciendo un buen trabajo. Esta es la función que optimizamos o minimizamos cuando realizamos el backpropagation.
* Algoritmo optimizador: es el algoritmo encargado de encontrar el conjunto de pesos que minimizan la función de error. El algoritmo más famoso se llama gradient descent, el cual utiliza las derivadas para que la pendiente le indique si incrementar o decrementar el peso.



a) Es un agente que aprende, ya que se va a utilizar una red neuronal artificial, la cual usa

mecanismos de aprendizaje para ir mejorando la eficacia para calcular el sueldo que le corresponde a un empleado a partir de su categoría, ausencias e hijos, y posee todos los elementos que tiene un agente que aprende, el elemento de actuación (la ANN), el elemento de aprendizaje (el algoritmo optimizador, Adam en este caso), la crítica (la función de costo, Mean squared error en este caso) y el generador de problemas (los datos de entrenamiento).

**REAS**

Rendimiento: Calcular un sueldo aproximado de manera correcta +100, Hacer un mal cálculo de un sueldo -100

Entorno: un listado de información de los empleados y sus atributos

Actuadores: una consola para mostrar los resultados

Sensores: un input donde el usuario ingresa los datos

b)

función Agente-ANN(percepción) devuelve una acción

variables estáticas: ANN, una red neuronal artificial

Decir(ANN, Construir-Sentencia-De-Percepción(percepción))

acción ← Preguntar(ANN, Pedir-Acción())

Decir(ANN, Construir-Sentencia-De-Acción(acción))

devolver acción

c)

Configuración 1

**Hiperparámetros relacionados con la estructura de la red neuronal:**

* Cantidad de capas ocultas: 75
* Función de costo (o loss): Mean squared error
* Función de activación: Relu
* Inicialización de pesos: Random normal

**Hiperparámetros relacionados con el entrenamiento:**

* Épocas: 50
* Tamaño de lote: 10000
* Algoritmo optimizador: Adam

Configuración 2

**Hiperparámetros relacionados con la estructura de la red neuronal:**

* Cantidad de capas ocultas: 75
* Función de costo (o loss): Mean squared error
* Función de activación: Softmax
* Inicialización de pesos: Random normal

**Hiperparámetros relacionados con el entrenamiento:**

* Épocas: 100
* Tamaño de lote: 5000
* Algoritmo optimizador: Adam

Configuración 3

**Hiperparámetros relacionados con la estructura de la red neuronal:**

* Cantidad de capas ocultas: 85
* Función de costo (o loss): Mean squared error
* Función de activación: Relu
* Inicialización de pesos: Uniform

**Hiperparámetros relacionados con el entrenamiento:**

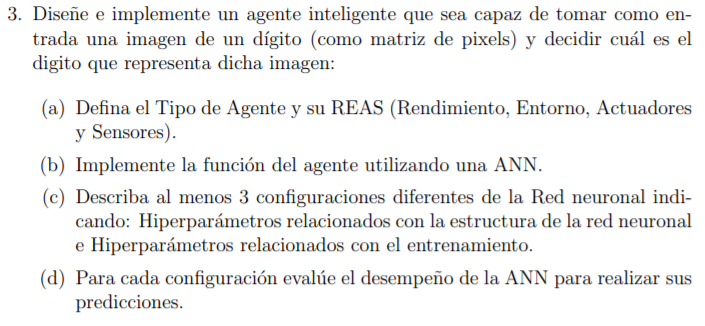
* Épocas: 50
* Tamaño de lote: 10000
* Algoritmo optimizador: Adam

d)

Configuración 1: Loss: 2642579.75

Configuración 2: Loss: 3366977024

Configuración 3: Loss: 3367082752



a) Es un agente que aprende, ya que se va a utilizar una red neuronal artificial, la cual usa

mecanismos de aprendizaje para ir mejorando la eficacia para determinar el número ingresado a partir de una matriz de píxeles, y posee todos los elementos que tiene un agente que aprende, el elemento de actuación (la ANN), el elemento de aprendizaje (el algoritmo optimizador, sdg en este caso), la crítica (la función de costo, Sparse categorical crossentropy en este caso) y el generador de problemas (los datos de entrenamiento).

**REAS**

Rendimiento: Identificar correctamente el dígito +100, Identificar de forma errónea el dígito -100

Entorno: un conjunto de imágenes (como matrices de píxeles) de dígitos escritos a mano

Actuadores: una consola para mostrar los resultados

Sensores: un input donde el usuario ingresa los datos o la imagen

b)

función Agente-ANN(percepción) devuelve una acción

variables estáticas: ANN, una red neuronal artificial

Decir(ANN, Construir-Sentencia-De-Percepción(percepción))

acción ← Preguntar(ANN, Pedir-Acción())

Decir(ANN, Construir-Sentencia-De-Acción(acción))

devolver acción

c)

Configuración 1:

**Hiperparámetros relacionados con la estructura de la red neuronal:**

* Cantidad de capas ocultas: 15
* Función de costo (o loss): Sparse categorical crossentropy
* Función de activación: Relu y Softmax
* Inicialización de pesos: Por defecto

**Hiperparámetros relacionados con el entrenamiento:**

* Épocas: 50
* Tamaño de lote: 1024
* Algoritmo optimizador: sgd

Configuración 2:

**Hiperparámetros relacionados con la estructura de la red neuronal:**

* Cantidad de capas ocultas: 30
* Función de costo (o loss): Sparse categorical crossentropy
* Función de activación: Relu y Softmax
* Inicialización de pesos: Por defecto

**Hiperparámetros relacionados con el entrenamiento:**

* Épocas: 30
* Tamaño de lote: 1024
* Algoritmo optimizador: sgd

Configuración 3:

**Hiperparámetros relacionados con la estructura de la red neuronal:**

* Cantidad de capas ocultas: 15
* Función de costo (o loss): Sparse categorical crossentropy
* Función de activación: Relu y Softmax
* Inicialización de pesos: Por defecto

**Hiperparámetros relacionados con el entrenamiento:**

* Épocas: 100
* Tamaño de lote: 2048
* Algoritmo optimizador: sgd

d)

Configuración 1: Loss: 0.3490 - Accuracy: 0.9021

Configuración 2: Loss: 0.3968 - Accuracy: 0.8955

Configuración 3: Loss: 0.3397 - Accuracy: 0.9057