TRABAJO FINAL APRENDIZAJE COMPUTACIONAL

1. INTRODUCCION

En este trabajo se va a presentar la modalidad de la utilización de uno o más modelos para datos concretos. En este caso hemos hecho una prueba con varios modelos enfrentándolos todos ellos a dos problemas de visión por computador clásicos, los cuatro modelos utilizados para estos problemas son:

* Una máquina de vectores soporte.
* Un perceptrón con solo una capa.
* Un perceptrón multicapa.
* Una red neuronal convolucional profunda.

Los dos problemas a los cuales hemos enfrentado estas estructuras han sido por un lado MNIST y por otro CIFAR-10.

El problema se ha elaborado usando los paquetes keras y scikit-Learn de Python ya que son los paquetes en los cuales el alumno que elabora esto se siente mas cómodo trabajando y también porque es la única forma que conoce de utilizar estructuras y algoritmos variados de una forma sencilla. Todo el código se puede encontrar en el siguiente [enlace](https://github.com/MagicElyas/MachineLearningFinalAssignment) y en la bibliografía también está enlazado.

En las siguientes secciones vamos a tratar el problema que se resuelve, cada uno de los modelos que estamos utilizando para resolverlo y para terminar se hará un estudio estadístico comparativo de las distintas ejecuciones que se tienen de las distintas resoluciones de los problemas. Además del código tanto en el enlace como en la entrega se encontrarán todos los ficheros csv generados y los convertidos a xlsx con las estadísticas extraídas de las distintas resoluciones de los distintos problemas, además de esto en el repositorio enlazado se encontrarán los comandos necesarios a ejecutar en consolas, así como los requisitos para hacer funcionar todo el código disponible.

1. EL PROBLEMA

Hoy en día tenemos más poder de computación que nunca a nuestra disposición y por tanto tenemos la capacidad de resolver problemas cada vez mas complejos. En el caso de este documento nos enfrentamos a un problema ya clásico de la inteligencia artificial, aunque se engloba dentro del ámbito de la visión por computador, hablamos del problema de clasificación de imágenes. En este trabajo nos hemos centrado en dos conjuntos de datos que también ya son un clásico como lo son MNIST que es el conjunto de datos por antonomasia de la visión por computado y CIFAR-10 otro conjunto de datos muy conocido, pero más difícil de clasificar por norma general que MNIST. El trabajo que se lleva a cabo con el consecuente experimento entra dentro del concepto de aprendizaje supervisado, pues en este caso tenemos

MNIST es un conjunto de datos que contiene imágenes de tamaño 28\*28 que reflejan números escritos a mano, se compone de 10 clases, es decir cada imagen contiene un número en el intervalo [0-9].



Ilustración 1 Ejemplo del conjunto de datos MNIST

Como se ha dicho es ya un clásico del análisis de imágenes y de las pruebas de los modelos de Inteligencia Artificial, es tan usado que no tenemos ni que descargarlo a parte pues viene integrado en keras como conjunto de datos.

CIFAR-10 es otro conjunto de datos muy conocido que se compone de 60000 imágenes de 32\*32 pixeles que también pertenecen a 10 clases, pero a diferencia de MNIST estas imágenes están en color con lo cual tenemos que tratar el triple de información, dado que las imágenes tienen los canales [RGB].

Como en el caso anterior este conjunto de datos es tan usado a la hora de realizar pruebas de modelos de aprendizaje computacional que está integrado en muchos paquetes, como es el ejemplo de keras.

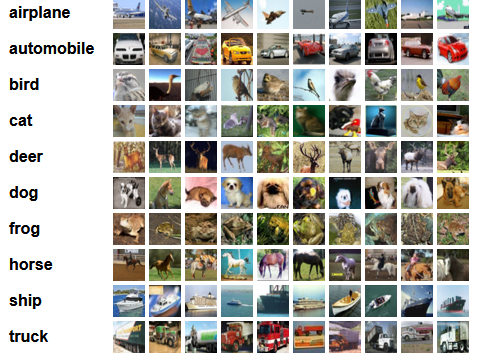


Ilustración 2 Ejemplo del conjunto de datos CIFAR-10

A la hora de realizar la minería de datos y la limpieza de los mismos, dado que los problemas a resolver seleccionados son conjuntos de datos cerrados, se realiza una percepción estadística en primer lugar para ver si hay algún problema con alguna clase minoritaria o hay problemas con la variabilidad de los datos y quitando que en MNIST la variabilidad es menor porque las imágenes tienen un solo canal, por lo demás son conjuntos de datos perfectos y equilibrados para realizar pruebas con los siguientes modelos.

1. LOS MODELOS

A la hora de enfrentarnos a problemas de visión por computador las mejores opciones hoy son las redes neuronales convolucionales o eso es lo que se recomienda desde muchos sitios, sin embargo, se ha decidido darle un punto de vista distinto, y hemos añadido una red neuronal convolucional produnda pero solo en uno de los modelos, en el resto hemos probado cosas que no se suelen usar con el análisis de imágenes.

En primer lugar, tenemos las máquinas de vectores soporte que son un modelo de clasificación que en principio no están indicadas para imágenes, pero como en todos los modelos, podemos pasarles el tipo de datos que queramos, en este caso se han escogido para darle variedad al trabajo y que no fueran todo redes neuronales. Además de esto, hace tiempo se probaron con otro tipo de datos y funcionaron mejor que las otras alternativas dentro del paquete scikit learn y en el problema de clasificación de imágenes simplificadas que utilizan desde scikit learn para mostrarte el poder de los distintos modelos, este modelo era de los que mejor se comportaba.

En segundo lugar, tenemos un perceptrón unicapa, (Single-Layer Perceptron) que conecta las entradas con las salidas sin capas ocultas. En estas dos capas que se tienen en la primera se han puesto 25 neuronas y en la segunda se han puesto 10 pues 10 son las posibles salidas del conjunto. No se ha adaptado el enfoque tradicional de tener una neurona de entrada para cada uno de los valores de entrada por motivos de complejidad computacional y de tiempo de ejecución.

En tercer lugar, tenemos el perceptrón multicapa que sigue la misma estructura que el anterior pero además incorpora una capa oculta, la capa oculta es una capa densa, es decir, este clasificador es lo que se conoce como una red neuronal completamente conectada (Fully-Connected Network. La introducción de la capa oculta en teoría lo que debería de darnos es un análisis mejor del problema pues si la dimensionalidad del problema sobrepasa lo que la red anterior es capaz de procesar va a venir bien porque la extracción de características va a ser mucho mejor.

El último modelo que tenemos es una red neuronal convolucional profunda. Esta red tiene muchas capas (50) y por tanto se necesita una potencia de computación importante para poder utilizarla, en este caso no es problema porque se dispone de una GPU de alta gama que nos va a permitir ejecutar el modelo sin muchos problemas, si bien se necesita tiempo para realizar las 30 ejecuciones para realizar la estadística.

Con estos modelos y estos conjuntos de datos nos enfrentamos al experimento.

1. EL EXPERIMENTO

El experimento que se realiza es la evaluación de los distintos modelos que se han citado anteriormente con relación a los problemas que también se han mencionado anteriormente. En este apartado vamos a analizar que parámetros se han tenido en cuenta y como hemos enfocado el análisis estadístico, aunque el estudio estadístico propiamente dicho se verá en la siguiente sección

En relación a los datos se ha realizado un preprocesamiento en todos los casos porque esto simplifica el problema y por tanto el tiempo de ejecución que de otra forma sería mucho más extenso.

Dicho preprocesamiento consiste en primer lugar, salvo en el caso de la red neuronal, en realizar un flatten de las imágenes, esto es, convertir la matriz en las que vienen codificadas las imágenes en vectores planos conteniendo toda la información de estas, esto es para facilitar las entradas a la red neuronal o a la máquina de vectores soporte que corresponda.

En segundo lugar, se realiza una normalización de la imagen, esto es, desplazar el rango de los valores que se tienen en los vectores o en las matrices en el caso que corresponda del rango [0-255] al rango [0-1]. Esto se hace porque las redes neuronales tienen un problema que se conoce como gradient vanishing que consiste en una perdida de información debido a trabajar con números grandes y por tanto se pierda la convergencia, por tanto, podemos hacer que el clasificador con el que estemos trabajando converja mucho más rápido simplemente realizando una normalización de los datos.

En tercer y último lugar se realiza una colocación categórica de los valores de los target, es decir de las clases a las que pertenecen cada uno de los ejemplos que se desean clasificar, esto se hace para evitar las distintas dicotomías que surgen a la hora de enfrentarse a un valor entero pues una única salida tiene el problema de la precisión pues no se sabe a que clase pertenece la predicción. Con este preprocesamiento aliviamos esto y lo que obtenemos es por cada ejemplo un vector de N elementos con las probabilidades de que el elemento X que pasamos pertenezca a cada una de las clases.

Una vez tenemos todos los datos preprocesados llega el momento de construir el modelo, el entrenamiento, las pruebas y el análisis.

En cuanto a entrenamiento en los 3 modelos que son redes neuronales de algún tipo se han decidido usar los mismos optimizadores y las mismas funciones de pérdida todo a enfocar el experimento en ver qué algoritmos y estructuras neuronales dan un mejor resultado con el resto de los parámetros iguales. El optimizador elegido es el optimizador ADAM y la función de pérdida es a la que nos referimos como categorical\_crossentropy.

Una vez especificadas las condiciones del experimento se procede al análisis estadístico de los distintos datos obtenidos gracias a las ejecuciones que se han realizado de los distintos modelos.

1. ANÁLISIS ESTADISTICO DE LOS RESULTADOS

Una vez realizado el experimento y una vez que tenemos todos los datos que necesitamos para realizar el análisis estadístico lo único que nos queda es realizar la comparativa de las precisiones y errores cometidos por los distintos algoritmos y estructuras neuronales utilizadas en este experimento.

Antes de ver el análisis estadístico se quieren hacer un par de aclaraciones.

En primer lugar, se han reducido el numero de muestras con las que se trabajan en el caso de la maquina de vectores soporte pues si le pasábamos todos los datos que se debían pasar se convertía en un problema impracticable.

En el caso de la red neuronal convolucional profunda no se ha podido realizar la prueba con MNIST pues el tamaño mínimo de entrada era 32\*32 y las imágenes que proporciona MNIST son de tamaño 28\*28.

Una vez aclarados estos dos temas se procede al análisis, en primer lugar, vamos a ver los resultados de la máquina de vectores soporte que solo se pudo probar con MNIST pues la cantidad de datos que proporcionaba CIFAR-10 no es tratable para este modelo. Los resultados obtenidos son los siguientes:

|  |  |  |  |
| --- | --- | --- | --- |
|  | ACC | MSE | EXECUTION TIME |
| MEDIA | 0,909319149 | 1,605085 | 19,15838859 |
| VARIANZA | 5,4345E-05 | 0,031754 | 0,639432799 |
| DESVIACIÓN TÍPICA | 0,007451601 | 0,180123 | 0,808290487 |

Tabla 1 Evaluación de rendimiento del modelo SVM en MNIST

Los datos de los cuales se han sacado estas medidas se encuentran en el fichero svm.xlsx disponible en el repositorio.

Como podemos ver la tasa de acierto es bastante alta para ser un modelo al que le hemos dado tan pocos datos y puede estar ahí el secreto, ya que a la hora de realizar las predicciones al tener que hacer menos predicciones y no cubrirse el espacio muestral completo puede que se introduzca un sesgo, aun así, para haber usado un 10% de todos los datos disponibles es un resultado mas que razonable.

Importante fijarse en el tiempo medio de entrenamiento necesitado para estas ejecuciones, en torno a los 20 segundos y esto es en gran medida gracias a que se han normalizado los datos antes de introducirlos en el modelo.

Hay una prueba en la que no se normalizan los datos, pero no llego a incluirse pues el tiempo de ejecución de dicha prueba fue de mas de media hora, por tanto, queda claro que normalizar los datos al menos en las máquinas de vectores soporte tiene sentido .

Una ultima cosa a destacar, la poca desviación típica que hay entre los valores de tasa de aciertos, lo cual indica que el modelo es muy consistente.

En segundo lugar, vamos a estudiar los resultados del perceptrón con sin capas ocultas respecto a las medidas que se obtienen con el conjunto de datos de MNIST y también con el conjunto de datos de CIFAR-10.

Este modelo consta de 25 neuronas de entrada que quedan conectadas a 10 neuronas de salida como se dijo con anterioridad a la hora de describir el experimento. Esta red neuronal corresponde lo que se conoce como una red completamente conectada o Fully Connected Network.

En primera instancia vamos a ver los resultados arrojados por el empleo de este clasificador en MNIST:

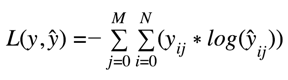
|  |  |  |  |
| --- | --- | --- | --- |
|  | ACC | LOSS | EXECUTION TIME |
| MEDIA | 0,9640394 | 0,123556815 | 12,15973945 |
| VARIANZA | 3,3533E-06 | 3,8894E-05 | 0,168470882 |
| DESVIACIÓN TÍPICA | 0,0018596 | 0,006333201 | 0,416816023 |

Tabla 2 Evaluación de rendimiento del modelo SLP en MNIST

Como podemos observar, lo que mas llama la atención es el aumento del 6% en tasa de aciertos, lo cual es bastante sorprendente en un modelo que antes alcanzaba ya el 90% aunque con una menor cantidad de datos.

Estas métricas están sacadas de la fase de pruebas la cual consiste en la predicción de 10000 imágenes que están ordenadas aleatoriamente, y en la fase de entrenamiento se emplean 60000 imágenes con sus correspondientes predicciones porque como se dijo anteriormente en este trabajo estamos realizando aprendizaje supervisado.

Algo que también sorprende de este modelo, es el valor medio tan bajo de la función de pérdida. Se debe aclarar que la función de pérdida utilizada en este caso no es el error cuadrático medio dado que para análisis de imágenes Keras no permite usarlo. En este caso se ha optado por la función de pérdida que tiene por nombre Categorical Crossentropy y que tiene la siguiente fórmula:



Donde el valor que se pretende predecir es ŷ. Para ello se compara la distribución de las predicciones con la distribución real donde la clase representada por la entrada esta puesta a 1 y el resto a 0. Cuanto mas cercanas las predicciones al valor real, menor el valor de la función de pérdida.

Algo que también resulta sorprendente es el valor de la función de pérdida, así como su desviación típica tan pequeña lo cual puede indicar buena generalización, aunque esto último no siempre es así.

El tiempo de ejecución es menor en este modelo que en el anterior y esto es porque tensorflow reparte el trabajo a hacer entre varios procesadores agilizando mucho mas el tiempo tanto de entrenamiento como de pruebas. Dicho esto, más tarde se justificará la introducción de la GPU para el procesamiento del último de los modelos.

Vamos a analizar los datos correspondientes a las pruebas con este modelo en el conjunto de datos de CIFAR-10

|  |  |  |  |
| --- | --- | --- | --- |
|  | ACC | LOSS | EXECUTION TIME |
| MEDIA | 0,264370967 | 1,939969913 | 23,98820283 |
| VARIANZA | 0,007973288 | 0,029123712 | 0,497584589 |
| DESVIACIÓN TÍPICA | 0,090769293 | 0,173477671 | 0,717057 |

Tabla 3 Evaluación de desempeño de SLP en CIFAR-10

Como podemos ver aquí la cosa ha cambiado mucho respecto a MNIST. La tasa de acierto es mucho menor y el valor de la función de pérdida es mucho más grande

Esto es debido a la mayor cantidad de datos que se tienen que procesar por parte del modelo, a la hora de realizar la clasificación de manera correcta. Triplicando la cantidad de datos a analizar la complejidad que adquiere la clasificación es mayor y por tanto el algoritmo debe tener una mayor capacidad de almacenaje y procesamiento de esa información.

Aun así, el aumento del valor de la función de pérdida es alarmante pues vemos como se dispara su valor aumentando 15 veces.

Estos datos junto con la duplicación del tiempo de ejecución lo único que nos indican es que la función a aproximar ahora por este sistema es más compleja, y que también tenemos mas datos a analizar por una red neuronal que puede ser insuficiente.

Ahora vamos a evaluar el modelo MLP que consiste en un perceptrón multicapa con 25 neuronas en la capa de entrada, 12 en la capa oculta y 10 en la capa de salida. En principio dichas neuronas disponibles en la capa oculta deberían ayudar

|  |  |  |  |
| --- | --- | --- | --- |
|  | ACC | LOSS | EXECUTION TIME |
| MEDIA | 0,961503227 | 0,135380589 | 12,86003716 |
| VARIANZA | 5,18497E-06 | 7,15504E-05 | 0,156778725 |
| DESVIACIÓN TÍPICA | 0,002277053 | 0,008458745 | 0,395952933 |

Tabla 4 Evaluación de desempeño del modelo MLP para MNIST

En este caso como podemos ver se empeora un poco la tasa de acierto y la función de perdida, pero en cambio se mejora la consistencia de los resultados en ambos campos.

También se puede observar que incluso añadiendo una capa oculta que añade complejidad el tiempo de ejecución no se ve apenas afectado.

Veamos ahora los resultados de este modelo para el conjunto de datos de CIFAR-10.

|  |  |  |  |
| --- | --- | --- | --- |
|  | ACC | LOSS | EXECUTION TIME |
| MEDIA | 0,35837 | 1,753481528 | 30,3579555 |
| VARIANZA | 0,002502818 | 0,008042951 | 8,639593948 |
| DESVIACIÓN TÍPICA | 0,05002817 | 0,089682502 | 2,93931862 |

Tabla 5 Evaluación de rendimiento para MLP en CIFAR-10

Como podemos observar la mejoría es apreciable con tan solo añadir una capa oculta, aumentamos en torno a un 9% la tasa de acierto y reducimos en 0.2 el valor de la función de pérdida sin repercutir muy negativamente en el tiempo de ejecución en el aprendizaje, aunque como se puede ver la varianza de este tiempo es mayor, pero puede ser algunas ejecuciones porque el procesador estuviera haciendo alguna otra tarea y por tanto se ampliara como consecuencia el tiempo de entrenamiento.

Por último, vamos a ver los resultados extraídos por la red neuronal convolucional:

|  |  |  |  |
| --- | --- | --- | --- |
|  | ACC | LOSS | EXECUTION TIME |
| MEDIA | 0,446356668 | 2,651611742 | 791,1615357 |
| VARIANZA | 0,009241458 | 8,315988924 | 117,1865064 |
| DESVIACIÓN TÍPICA | 0,096132502 | 2,883745641 | 10,82527166 |

Tabla 6 Evaluación de rendimiento de la red neuronal convolucional profunda para CIFAR-10

Como podemos observar mejora bastante al clasificador sin capas ocultas y algo también mejora al que tiene una capa oculta, sin embargo, siendo esta un modelo que tiene muy buenas valoraciones entre la comunidad esperaba algo más de él, que acercara la tasa de acierto al 70% como mínimo.

Para este modelo se tuvo que utilizar la tarjeta gráfica (GTX 1070) pues con el procesador el problema podía tardar días en entrenarse (2210 segundos por época) por tanto se decide utilizar la tarjeta gráfica para agilizar el entrenamiento y como se puede ver se reduce mucho el tiempo de entrenamiento.

La poca tasa de acierto alcanzada respecto a la expectativa que se tenía sobre la misma puede deberse a que la red tiene un orden de complejidad muy superior al del problema y puede llegar al sobreajuste, de hecho, esta hipótesis tiene sentido pues durante la fase de entrenamiento se llegaban a tasas de acierto del 96% con lo cual puede ser indicativo que esta red neuronal es demasiado compleja para el problema que se intenta resolver.

1. CONCLUSIONES

En primer lugar, hay que destacar que lo que a priori puede parecer una mejor opción a la hora de resolver el problema concreto como esta arquitectura de red neuronal profunda puede ser contraproducente a la hora de ser evaluada frente a otras estructuras que pueden resultar mas sencillas o que no han sido probadas por los grandes expertos del aprendizaje automático.

En segundo lugar, este trabajo ha dejado patente que no porque un modelo parezca mas adecuado a primera vista tenga que serlo porque se puede introducir el sobreajuste de la función y puede terminar siendo fatal para la aplicación que se desea construir. Por tanto, de aquí se extrae que se debe buscar el modelo que mejor se adapte a nuestros datos o a la complejidad de estos.

En tercer lugar, la búsqueda de los datos y limpieza de estos no ha resultado muy compleja pues son conjuntos de datos arquetípicos y que son muy usados a la hora de probar y comparar distintos modelos de redes neuronales o de clasificadores no basados en neuronas artificiales. Por esto mismo se ha decidido centrar el trabajo en la exploración de los distintos modelos y no tanto en la búsqueda de conjuntos de datos variados y las técnicas de depuración y minería de estos.

Por último y como continuación de este trabajo convendría realizar otro experimento probando modelos neuronales intermedios en cuanto a complejidad y con capas distintas en busca del mejor modelo para resolver nuestro problema concreto

BIBLIOGRAFÍA

<https://github.com/MagicElyas/MachineLearningFinalAssignment>

<https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy>