TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Implementación optimizada sobre sistemas heterogéneos de algoritmos de Deep Learning para clasificación de imágenes

Autor
David Sánchez Pérez

Directores
José Miguel Mantas Ruiz

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, mes de Febrero 2024

Implementación optimizada sobre sistemas heterogéneos de algoritmos de Deep Learning para clasificación de imágenes

Autor
David Sánchez Pérez

Directores
José Miguel Mantas Ruiz

Título del Proyecto: Subtítulo del proyecto

Nombre Apellido1 Apellido2 (alumno)

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3, ...

Resumen

Poner aquí el resumen.

Project Title: Project Subtitle

First name, Family name (student)

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Nombre Apellido1 Apellido2**, alumno de la titulación TITULACIÓN de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI XXXXXXXXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Nombre Apellido1 Apellido2

Granada a X de mes de 201 .

D. **Nombre Apellido1 Apellido2 (tutor1)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

D. **Nombre Apellido1 Apellido2 (tutor2)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Nombre Apellido1 Apellido2 (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Agradecimientos

Poner aquí agradecimientos...

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Resumen

1.2. Estado del arte

1.3. Objetivos

Capítulo 2

Conceptos previos

2.1. Machine Learning

Se entiende como el campo de las ciencias de computación que en vez de enfocarse en el diseño de algoritmos explícitos, optan por el estudio de técnicas de aprendizaje. Este enfoque tiene un gran éxito en tareas computacionales donde no es factible diseñar un algoritmo de forma explícita. [1]
En vez de averiguar las distintas reglas a seguir para llegar a una solución, esta alternativa permite simplemente suministrar ejemplos de lo que debería pasar en distintas situaciones, y dejar que la máquina aprenda y extraiga ella misma sus propias conclusiones. De esta forma, el procedimiento en aprendizaje supervisado consiste en 'entrenar' con una muestra de N ejemplos, extraer información de ellos, y posteriormente poder evaluar de forma 'correcta' (bajo un margen de error controlado) otra muestra de M ejemplos, siendo M >N. [2]
Este enfoque ha contribuido en el avance de áreas como reconocimiento de voz, visión por ordenador, procesamiento de lenguaje natural, etc.

2.2. Deep Learning

2.3. Tipos de aprendizaje

2.3.1. Aprendizaje Supervisado

Es el que se empleará en este proyecto. Se caracteriza por la presencia de una etiqueta 'correcta' y_i asociada a cada dato de entrada x_i. Posteriormente, la red empleará ambos valores para, a partir de x_i, tratar de deducir y_i. [2]
Aunque se tratará de impedirlo, siempre hay ruido en los datos empleados, implicando que algunas etiquetas de Y=$\{y_1, y_2, ..., y_N\}$ pueden ser erróneas.

2.3.2. Aprendizaje No Supervisado

En este tipo de aprendizaje, los datos no contienen ninguna información respecto a lo que debe predecir la red. De esta forma, el conjunto de datos D se compone exclusivamente de valores X=$\{x_1, x_2, ..., x_N\}$. [2]

2.3.3. Aprendizaje Por Refuerzo

En este caso tampoco existe un y_i 'correcto' asociado a cada x_i. En su lugar, se asocia a cada x_i una etiqueta con un valor posible de y_i, además de una medida que indica como de bueno es el mismo. [2]

2.4. Tipos de problemas en machine learning

2.4.1. Clasificación

2.4.2. Regresión

2.5. División de datos en entrenamiento y test

En aprendizaje supervisado, al disponer de N ejemplos se dice que se dispone de una muestra D = $\{d_1 = (x_1, y_1), d_2 = (x_2, y_2), ..., d_N = (x_N, y_N)\}$.

Dicho conjunto de datos D se suele dividir en 2 subconjuntos, entrenamiento y test. El objetivo de ello es comprobar si realmente el programa 'aprende' o solo memoriza.
El procedimiento consiste en entrenar una red determinada empleando exclusivamente los datos del conjunto de entrenamiento. Una vez terminado todo el entrenamiento y nunca antes de ello, se accede al subconjunto test y se comparan las predicciones sobre este con los valores reales que se encuentran en cada y_i correspondiente. De esta forma, se observa el comportamiento de la red 'fuera de la muestra' y da una idea de como generaliza. Es decir, si ha aprendido o por el contrario simplemente ha logrado memorizar todos los ejemplos con los que se entrenó.

2.6. Entrenamiento

2.7. Redes Neuronales Totalmente Conectadas

2.7.1. Neurona

2.7.2. Estructura por capas

2.7.3. Función de activación

RELU para capas intermedias, sigmoide para última capa.

2.7.4. Función de error o pérdida

Como solo tenemos dos clases y estamos en clasificación binaria, usaremos Sigmoid Cross Entropy Loss.

$$H(x) = -\frac{1}{N}\sum_{i=1}^{N}[y_i * log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)] \tag{2.1}$$

y = etiqueta real
$\hat{y}$ = predicción

2.7.5. ForwardPropagation

2.7.6. Descenso del gradiente

2.7.7. BackPropagation

Regla de la cadena

2.8. Redes Neuronales Convolucionales

2.8.1. Tipos de capas

2.8.2. Estructura por capas

2.8.3. ForwardPropagation

Capítulo 3

Aportaciones

3.1. Redes Neuronales Totalmente Conectadas

3.1.1. BackPropagation con 1 capa oculta

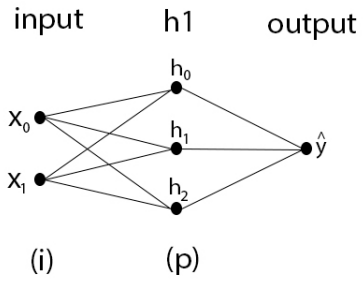

Figura 3.1: Red Neuronal totalmente conectada con 1 capa oculta

La Figura 3.1 se compone de puntos y líneas, representando neuronas y pesos que las conectan respectivamente. Cada punto es una neurona, y cada línea un peso.
El peso W_{ip} referencia al peso que une las neuronas X_i y h_p.
El peso $W_{p\hat{y}}$ representa el peso que conecta las neuronas h_p y $\hat{y}$.

Capa output

Sea la neurona n, se define como n_{in} el valor de dicha neurona antes de aplicar sobre ella su función de activación asociada, y n_{out} el obtenido tras aplicarla. De esta forma, si la última capa es $\hat{y}$ y solo tiene una neurona, $\hat{y_{in}}$ y $\hat{y_{out}}$ corresponderán con los valores antes y después de aplicar la función de activación respectivamente.

De esta forma, la función de pérdida (2.1) se convierte en:

$$H(x) = -\frac{1}{N}\sum_{i=1}^{N}[y_i * log(\hat{y}_{out_i}) + (1 - y_i) * log(1 - \hat{y}_{out_i})] \tag{3.1}$$

Para realizar el descenso del gradiente, se debe empezar calculando la derivada de la función de pérdida respecto a la predicción obtenida. Es decir, la derivada de la fórmula (3.1) respecto de las neuronas en la última capa de la red tras aplicar sus respectivas funciones de activación, que en este caso corresponde a $\hat{y}_{out}$.
Por simplicidad, podemos dividir esta derivada en 2 partes.
Parte izquierda:

$$f(x) = A * B \tag{3.2}$$
$$f'(x) = AB' + A'B \tag{3.3}$$
$$\frac{dy_i}{d\hat{y}_{out_i}} = 0 \tag{3.4}$$
$$\frac{dlog(\hat{y}_{out_i})}{d\hat{y}_{out_i}} = \frac{1}{\hat{y}_{out_i}} \tag{3.5}$$
$$\frac{dy_i * log(\hat{y}_{out_i})}{d\hat{y}_{out_i}} = y_i * \frac{1}{\hat{y}_{out_i}} + 0 * log(\hat{y}_{out_i}) = \frac{y_i}{\hat{y}_{out_i}} \tag{3.6}$$

Parte derecha:

$$\frac{d(1 - y_i)}{d\hat{y}_{out_i}} = 0 \tag{3.7}$$
$$\frac{dlog(1 - \hat{y}_{out_i})}{d\hat{y}_{out_i}} = \frac{1}{1 - \hat{y}_{out_i}} * (-1) \tag{3.8}$$
$$\frac{d(1 - y_i) * log(1 - \hat{y}_{out_i})}{d\hat{y}_{out_i}} = (1 - y_i) * \frac{1}{1 - \hat{y}_{out_i}} * (-1) + 0 * log(1 - \hat{y}_{out_i}) \tag{3.9}$$
$$\frac{d(1 - y_i) * log(1 - \hat{y}_{out_i})}{d\hat{y}_{out_i}} = -\frac{1 - y_i}{1 - \hat{y}_{out_i}} \tag{3.10}$$

Finalmente, se obtiene:

$$\frac{dH(x)}{d\hat{y}_{out_i}} = -\frac{1}{N}\sum_{i=1}^{N}[\frac{y_i}{\hat{y}_{out_i}} - \frac{1 - y_i}{1 - \hat{y}_{out_i}}] \tag{3.11}$$

Función activación de la capa output

En la capa output se emplea la función de activación sigmoide.

$$sigmoide(x) = \frac{1}{1 + e^{-x}} \tag{3.12}$$

$$sigmoide'(x) = \frac{sigmoide(x)}{1 - sigmoide(x)} \tag{3.13}$$

De esta forma,

$$\frac{d\hat{y}_{out}}{d\hat{y}_{in}} = \frac{dsigmoide(\hat{y}_{in})}{d\hat{y}_{in}} = sigmoide(\hat{y}_{in}) * (1 - sigmoide(\hat{y}_{in})) \tag{3.14}$$

Ahora, podemos calcular el gradiente completo hasta la capa output antes de aplicar su función de activación.

$$grad\_output = \frac{dH(x)}{d\hat{y}_{out}} * \frac{d\hat{y}_{out}}{d\hat{y}_{in}} = -\frac{1}{N}\sum_{i=1}^{N}[\frac{y_i}{\hat{y}_{out_i}} - \frac{1 - y_i}{1 - \hat{y}_{out_i}}] * \frac{d\hat{y}_{out}}{d\hat{y}_{in}} \tag{3.15}$$

$$grad\_output = -\frac{1}{N}\sum_{i=1}^{N}[\frac{y_i}{\hat{y}_{out_i}} - \frac{1 - y_i}{1 - \hat{y}_{out_i}}] * sigmoide(\hat{y}_{in}) * (1 - sigmoide(\hat{y}_{in})) \tag{3.16}$$

Pesos capas h1-output

Una vez calculado el gradiente hasta la capa output, se puede calcular el gradiente respecto a cada peso que se encuentra conectado a esta desde la capa anterior. Es decir, para cada $h_p \in h1$, se calcula $\frac{dH(x)}{dW_{p\hat{y}}}$. Usando la regla de la cadena, equivale a realizar lo siguiente:

$$\frac{d\hat{y}_{in}}{dW_{p\hat{y}_{in}}} = \frac{dh_{1out_p} * W_{p\hat{y}_{in}}}{dW_{p\hat{y}_{in}}} = h_{1out_p} \tag{3.17}$$

$$\frac{dH(x)}{dW_{p\hat{y}_{in}}} = \frac{dH(x)}{d\hat{y}_{out}} * \frac{d\hat{y}_{out}}{d\hat{y}_{in}} * \frac{d\hat{y}_{in}}{dW_{p\hat{y}_{in}}} = grad\_output * \frac{d\hat{y}_{in}}{dW_{p\hat{y}_{in}}} = grad\_output * h_{1out_p} \tag{3.18}$$

Capa oculta h1

$$\frac{d\hat{y}_{in}}{dh_{1out_p}} = \frac{dh_{1out_p} * dW_{p\hat{y}_{in}}}{dh_{1out_p}} = dW_{p\hat{y}_{in}} \tag{3.19}$$

$$\frac{dh_{1out_p}}{dh_{1in_p}} = \frac{drelu(h_{1in_p})}{dh_{1in_p}} = 0 \; si \; h_{1in_p} <= 0, \; 1 \; en \; caso \; contrario = deriv\_relu(h_{1in_p}) \tag{3.20}$$

$$\frac{dH(x)}{dh_{1in_p}} = grad\_output * \frac{d\hat{y}_{in}}{dh_{1out_p}} * \frac{dh_{1out_p}}{dh_{1in_p}} = grad\_output * W_{p\hat{y}_{in}} * deriv\_relu(h_{1in_p}) \tag{3.21}$$

$$\frac{dH(x)}{dh_{1in_p}} = grad\_h1p\_output \tag{3.22}$$

Pesos capas input-h1

$$\frac{h_{1in_p}}{W_{ip}} = \frac{dX_i * W_{ip}}{dW_{ip}} = X_i \tag{3.23}$$

$$\frac{dH(x)}{dW_{ip}} = grad\_h1p\_output * \frac{dh_{1in_p}}{dWip} = grad\_h1p\_output * X_i \tag{3.24}$$

Capa input

$$\frac{dh_{1in_p}}{dX_{out_i}} = \frac{dX_{out_i} * W_{ip}}{dX_{out_i}} = W_{ip} \tag{3.25}$$

$$\frac{dX_{out_i}}{dX_{in_i}} = 1 \tag{3.26}$$

Ahora se realiza una sumatoria con todos los 'caminos posibles' hacia la misma neurona

$$\frac{dH(x)}{dX_{in_i}} = \sum_{p=0}^{h_1.size()} grad\_h1p\_output * \frac{dh_{1in_p}}{dX_{out_i}} * \frac{dX_{out_i}}{dX_{in_i}} \tag{3.27}$$

$$\frac{dH(x)}{dX_{in_i}} = \sum_{p=0}^{h_1.size()} grad\_h1p\_output * W_{ip} \tag{3.28}$$

3.1.2. BackPropagation con 2 capas ocultas

Capítulo 4

Adaptación GPU

4.1. GPU en Redes Neuronales Totalmente Conectadas

4.2. GPU en Redes Neuronales Convolucionales

Capítulo 5

Comparación con distintas plataformas

5.1. cuDNN

Bibliografía

[1] Izzat El Hajj Wen-emi W.Hwu, David B.kirk. *Programming Massively Parallel Processors*. Morgan Kaufmann, 50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States, 4 edition, 2022.

[2] Hsuan-Tien Lin Yaser S. Abu-Mostafa, Malik Magdon-Ismail. *Learning From Data*. California Institute of Technology Pasadena, CA 91125, USA, 1 edition, 2012.