Universidad Católica de Temuco
Facultad de Ingeniería
Departamento de Ciencias Matemáticas y Físicas

Una propuesta de modelo probabilístico adaptativo de contexto aplicado al reconocimiento visual de objetos.

Por

Norman Vergaray Illescas

Profesor Guía

Dr. Billy Peralta Márquez

Actividad Formativa Equivalente, para optar al grado de
Magíster en Matemáticas Aplicadas (Profesional).

Temuco - 9 de enero de 2018

Universidad Católica de Temuco
Facultad de Ingeniería
Departamento de Ciencias Matemáticas y Físicas

COMISIÓN EVALUADORA

Profesor Guía:

.....
Dr. Billy Peralta Márquez

Profesor informante:

.....
Dr. Emilio Cariaga López

Profesor informante:

.....
Dr. Julio Rojas Mora

Profesor informante:

.....
-

Director del Programa (Ministro de fe):

.....
-

Temuco .....

Perfil de Egreso

Magíster en Matemáticas Aplicadas.
Universidad Católica de Temuco.

El egresado del Magíster en Matemáticas Aplicadas es un profesional posgraduado que posee la competencia de aplicar la matemática al análisis de sistemas y procesos complejos en el ámbito de los fenómenos de transporte.

Formula ecuaciones diferenciales como modelos matemáticos, en el ámbito de los fenómenos de transporte, para obtener una relación cuantitativa entre las variables relevantes del sistema.

Resuelve ecuaciones diferenciales como modelos matemáticos, utilizando técnicas numéricas y analíticas, para obtener valores cuantitativos de la variable respuesta del sistema.

Utiliza programas computacionales en la resolución, análisis y aplicación de ecuaciones diferenciales al mejoramiento de sistemas complejos en el ámbito de los fenómenos de transporte.

Agradecimientos

A mi director de Tesis, Dr. Billy Peralta Márquez, por darme la oportunidad de participar en investigación dentro del área de visión computacional, así como su guía a lo largo de este proceso. Al director del Magíster, Dr. Emilio Cariaga López, por su orientación en temas numéricos y la supervisión de mi avance académico. A los doctores Paul Grassia y Stefan Berres por los conocimientos entregados con respecto modelos probabilísticos, así como al resto de académicos del Departamento de Ciencias Matemáticas y Físicas por estar dispuestos a responder mis inquietudes. Además, al Dr. Julio Rojas por dar una segunda opinión desde el punto de vista informático, dentro del marco del desarrollo de esta Tesis.

Dedicada en memoria de mi madre
Rebeca Illescas Encina
y a mi padrastro
Víctor Basaez Pérez.

Abstract

Within the increasing automation of tasks, it is necessary to obtain relevant information from images, to have clarity of the actions that must be performed. In this process, it is possible to detect objects individually, however this mode can generate a considerable error due to the enormous number of ways in which an object can be presented. It is therefore necessary to improve the level of accuracy, and a way to achieve this is taking into account the relationships between objects as well. Until now, context-oriented models generate relationships between objects without discriminating between input images. It is necessary that the model operates on each particular image, to reduce the error and obtain a conclusive result. In response to the above, this work proposes an adaptive probabilistic context-model.

The model in question is a functional that processes the occurrence probability of the objects in each image, and generates the relations between the detectable objects using a Bayesian network in the form of a tree. This model is updated with the characteristics of each image, requiring a minimization of the quadratic error through a numerical approximation, obtained by Newton-Raphson method. Comparisons were made between different heuristic proposals, as well as tests on different context-oriented databases, in order to validate the results. On the other hand, tests were performed using features extracted through histograms of orient gradients and Deep Learning. It was found that an improvement in the prediction of the order of 20% is feasible in the testing process. This result was possible using histograms of oriented gradients, which had a higher performance than deep learning. In a later instance, this proposal will be enriched by adding more variables to the adaptive model, considering more aspects of the image.

Resumen

Dentro de la creciente automatización de tareas, se requiere la obtención de información relevante a partir de imágenes, para tener claridad de las acciones que se deben realizar. En este proceso, es posible detectar los objetos de manera individual, sin embargo esta modalidad puede generar un error considerable debido a la enorme cantidad de formas en que un objeto se puede presentar. Se requiere, por tanto, mejorar el nivel de exactitud, y una forma de conseguirlo es tomando en cuenta también las relaciones entre los objetos. Hasta ahora, los modelos orientados a contexto generan relaciones entre los objetos sin discriminar entre las imágenes de entrada. Es necesario que el modelo opere sobre cada imagen en particular, para reducir el error y obtener un resultado concluyente. En respuesta a lo anterior, en este trabajo se propone un modelo probabilístico adaptativo de contexto.

El modelo en cuestión es un funcional que procesa la probabilidad de aparición de los objetos en cada imagen, y genera las relaciones entre los objetos detectables utilizando una red bayesiana en forma de árbol. Dicho modelo se va actualizando con las caraterísticas de cada imagen, requiriéndose una minimización del error cuadrático a través de una aproximación numérica, obtenida por medio del método de Newton-Raphson. Se llevó a cabo comparaciones entre distintas propuestas de heurísticas, así como pruebas sobre distintas bases de datos orientadas a contexto, con el fin de validar los resultados. Por otro lado, se llevaron a cabo pruebas empleando características extraídas a través de histogramas de gradientes orientados y Aprendizaje Profundo. Se halló que es factible una mejora en la predicción del orden de un 20 %, en el proceso de testeo. Este resultado fue posible empleando histogramas de gradientes orientados, que tuvo un desempeño superior al del Aprendizaje Profundo. En una instancia posterior, se enriquecerá esta propuesta agregando más variables al modelo adaptativo, considerando más aspectos de la imagen.

Índice

1 Introducción **1**

2 Objetivos **3**

3 Materiales y métodos **4**
3.1 Modelo conceptual 4
3.1.1 Aprendizaje de Máquina 4
3.1.2 Aprendizaje Predictivo 4
3.1.3 Aprendizaje Descriptivo 5
3.1.4 Preliminares de probabilidad 6
3.1.5 Redes Bayesianas 7
3.2 Modelo matemático 10
3.3 Solución numérica 11
3.3.1 Minimización a través de mínimos cuadrados 11
3.3.2 Minimización por medio de heurística 12
3.3.3 Descripción del método numérico 13
3.4 Simulación computacional 15

4 Resultados **19**
4.1 Ajuste de f por medio de mínimos cuadrados. 19
4.2 Ajuste de f por medio de heurística 22
4.2.1 Comparación del comportamiento de distintos pesos empleando *HOG* 22
4.2.2 Comparación del comportamiento de distintos pesos empleando Aprendizaje Profundo 25
4.2.3 Análisis de convergencia 28

5 Conclusiones **30**

6 Referencias **31**

7 Códigos MATLAB y Python **34**
7.1 Implementación del modelo probabilístico adaptativo de contexto . . . 34
7.2 Extracción de características empleando HOG 37
7.3 Función que evalúa E 38
7.4 Cálculo de $\frac{\partial E}{\partial \theta}$ 39
7.5 Cálculo de la heurística A 40
7.6 Cálculo de f 40
7.7 Obtención de $\frac{\partial f}{\partial \theta}$ 41
7.8 Determinación de la Máxima Verosimilitud 41
7.9 Extracción de características empleando Aprendizaje Profundo 42

Tabla de Vocabulario Técnico Inglés - Español

Cellsize	–	Tamaño de celda
Deep Learning	–	Aprendizaje Profundo
CRF	–	Campo Condicional Aleatorio
Damped inexact Newton − Raphson (*DINR*)	–	Newton-Raphson inexacto disminuido
Features	–	Características
Framework	–	Marco de trabajo
Knowledge Discovery	–	Descubrimiento de conocimiento
Maximum Likelihood	–	Máxima Verosimilitud
Set	–	Conjunto
Reshaping	–	Transformación

1 Introducción

En la actualidad existe un creciente desarrollo del aprendizaje de máquina, con el fin de sustituir por computadores tareas típicamente llevadas a cabo por humanos (Murphy, 2012). Dentro de este contexto es que se encuentra la visión computacional, que permite la obtención de información a partir de imágenes digitales. La detección de objetos en visión computacional plantea una serie de desafíos, y puesto que la detección individual no suele entregar toda la información, es necesario tomar en cuenta otros factores relativos al contexto en que se encuentran.

Lo anterior reviste especial importancia teniendo en cuenta el gran número de aplicaciones de la visión computacional (Szeliski, 2010). Algunas de ellas se encuentran en el reconocimento facial y de huellas dactilares, así como la detección de intrusos. Por otra parte, se emplea en la conducción automática, en el modelamiento 3D a partir de fotografías aéreas y en la imagenología médica. En este último caso, se trabaja con imágenes antes y durante las operaciones, como también en el estudio de los cambios en la morfología del cerebro a medida que envejece. Además, se utiliza en el reconocimiento óptico de caracteres, con el fin de procesar las letras de las frases escritas en una carta que indican su destino y reconocer placas de patentes. También, permite la detección de fallas en maquinaria a través de visión estéreo, especialmente en la aeronáutica y automóviles, así como hallar defectos en piezas fundidas de acero mediante rayos X. Finalmente, es posible reconocer de forma automática los artículos que se compran en un supermercado a través de cajas automáticas.

Hasta ahora, se han implementado diversas estrategias para resolver el problema. Entre las más simples, Rabinovich, Vedaldi, Galleguillos, Wiewiora y Belongie (2007) utilizan detectores locales para asignar inicialmente una etiqueta de objeto a cada segmento de imagen, y ajustar dichas etiquetas por medio de un campo condicional aleatorio, que corresponde a un tipo de método de modelamiento estadístico aplicado en reconocimento de patrones y aprendizaje de máquinas. Por otra parte, Torralba, Murphy y Freeman (2005) combinan un meta-algoritmo que reduce el sesgo y varianza en un contexto de aprendizaje supervisado, y también, un campo condicional aleatorio. Hasta aquí, la idea es detectar primero objetos de fácil reconocimiento, para luego detectar objetos más difíciles a través de la información de contexto.

Otras estrategias más complejas incluyen la inclusión de una característica global llamada *gist*, que puede determinar la presencia o ausencia de objetos y sus ubicaciones sin emplear un detector local (Torralba, 2003). También, Heitz, Gould, Saxena y Koller (2008) introducen un modelo de clasificación en cascada, el cual relaciona la categorización de la escena global, segmentación de la imagen en regiones significativas y su posterior etiquetado, detección de objetos y reconstrucción 3D. Choi, Lim, Torralba y Willsky (2010) abordan el problema utilizando detección local de objetos, determinando las relaciones entre los objetos y reconociendo la escena global. Finalmente, Peralta et al. (2012) utilizan un modelo jerárquico para la detección de objetos combinando dis-

tribuciones de probabilidad, las cuales obtienen un resultado diferente según la imagen de entrada.

En los trabajos expuestos, se puede apreciar que ya se tiene conciencia de que se requiere considerar tanto la identificación de la escena global, como de cada uno de los objetos y las relaciones entre ellos dentro de un modelo jerárquico. Sin embargo, los modelos desarrollados operan sobre el total de imágenes a procesar, y no capturando la particularidad de cada una de ellas. Es en respuesta a lo anterior que se propone en este trabajo un modelo jerárquico probabilístico de contexto, para la detección de objetos visuales, de carácter dinámico, que genera un resultado distinto para cada imagen, tomando en cuenta sus propias características.

En el Capítulo 3 de este informe se expone el modelo conceptual, analizando los fundamentos teóricos de la solución propuesta. Lo anterior incluye abordar el aprendizaje de máquina y la diferencia entre aprendizaje predictivo y descriptivo, para luego pasar a una definición de las redes bayesianas, las cuales se construyen de acuerdo a las probabilidades condicionales de aparición de cada uno de los objetos posibles en cada imagen. Además, se efectúa una breve introducción al concepto de Aprendizaje Profundo. Más tarde se describen las dos alternativas de modelos matemáticos probabilísticos, una que implementa el método de mínimos cuadrados y otra que opera a través de heurísticas. Posteriormente se expone el modelo numérico, basado en el método de Newton-Raphson y la regla de Armijo para el descenso de gradiente. Finalmente, se dan algunas explicaciones de la implementación computacional indicando los recursos utilizados.

Prosiguiendo, en el Capítulo 4 se compara el rendimiento de ambos modelos matemáticos, incluyendo el análisis sobre el impacto de distintos pesos para la heurística del segundo modelo. En ambos casos, se expondrá el comportamiento de las soluciones empleando histogramas de gradientes orientados y Aprendizaje Profundo para el cálculo de las características de cada imagen. Finalmente, en el Capítulo 5 se presenta la discusión de los resultados obtenidos así como las conclusiones extraídas, junto a las perspectivas del trabajo, indicando las mejoras necesarias para generar otros aportes.

2 Objetivos

Objetivo general

Proponer un nuevo modelo jerárquico probabilístico de contexto, para la detección de objetos visuales.

Objetivos específicos

1. Definir un modelo matemático que represente el modelo probabilístico jerárquico adaptativo.
2. Implementar el modelo matemático propuesto reduciendo el costo computacional, con el fin de que pueda operar con los recursos disponibles.
3. Validar el modelo por medio del empleo de bases de datos reales de objetos visuales, comparando con el estado del arte.

3 Materiales y métodos

En esta sección se detalla el modelo de contexto jerárquico probabilístico de carácter adaptativo, así como la obtención de la solución numérica empleando el método de Newton-Raphson y descenso de gradiente basado en la regla de Armijo.

3.1 Modelo conceptual

3.1.1 Aprendizaje de Máquina

El Aprendizaje de Máquina (*Machine Learning*) corresponde al conjunto de métodos que pueden detectar automáticamente patrones a partir de datos, y que posteriormente emplea los patrones descubiertos para predecir datos futuros o llevar a cabo otros tipos de decisión que operen sobre incerteza, tal como planear la forma de recolectar más datos (Murphy, 2012). Esto es de vital importancia en el análisis de grandes volúmenes de datos. Luego, para trabajar con problemas que operan bajo incerteza se recurre a teoría de probabilidades.

El Aprendizaje de Máquina se divide frecuentemente en *aprendizaje predictivo* y *aprendizaje descriptivo*.

3.1.2 Aprendizaje Predictivo

También llamado *aprendizaje supervisado*, tiene como objetivo es aprender un mapeo a partir de entradas $\mathbf{x}$ hacia las salidas y, tomando como insumo un conjunto etiquetado de pares entrada-salida $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. $\mathcal{D}$ es denominado *conjunto de entrenamiento* y N es el número de ejemplos de entrenamiento. En otras palabras, $\mathbf{x}$ corresponde a la variable independente e y, a los datos observados que se desea pronosticar. $\mathcal{D}$ por tanto, corresponde a una *base de conocimiento* sobre la cual el modelo debe realizar aprendizaje, y está compuesta por pares de datos $(\mathbf{x}_i, y_i)$ con el valor evaluado y el valor observado.

En la configuración más simple, cada entrada de entrenamiento $\mathbf{x}_i$ correspondiente a un ejemplo de entrenamiento i es un vector D-dimensional de números. Éstos son llamados **características** o **atributos**. Además, suelen estar almacenados en una matriz de diseño $\mathbf{X}$ de dimensiones $N \times D$.

De forma similar, la salida o *variable de respuesta* puede en principio ser cualquiera, pero la mayoría de los métodos asume que y_i es una variable *categórica* que puede hallarse dentro de un conjunto discreto de salidas $y_i \in \{1, ..., C\}$, o bien, ser continua de valores reales. Cuando y_i es categórica el problema se conoce como de *clasificación* o *reconocimento de patrones*, mientras que al ser y_i real, se conoce como *regresión*.

En el caso de un problema de clasificación, C corresponde al número de clases. Si $C = 2$, se denomina *clasificación binaria*, donde se puede asumir que $y_i \in \{0, 1\}$. En el caso de que $C > 2$, se trata de *clasificación multiclase*. En el caso de que las etiquetas no sean mutuamente exclusivas, se denomina problema de *clasificación multietiqueta*, siendo visto de mejor forma como la predicción múltiple de clases binarias.

A modo de ejemplo, supongamos que queremos reconocer si la parte de una moneda que estemos observando corresponde a *cara* o a *sello*. Debemos realizar sucesivas observaciones sobre monedas encontradas $i \in \{1, ..., N\}$ capturando la imagen, y en cada una de ellas se debe extraer las características de dicha imagen y almacenarlas en un vector de dimensión D. Éstas corresponderán a $\mathbf{x}_i$. Además, se han de relacionar con los datos observados (si la moneda muestra *cara* o *sello*), lo cual es posible denotar como $y_i \in \{0, 1\}$ (0 correspondería a *sello* y 1 a *cara*). De esta manera se construye la base de datos $\mathcal{D}$.

Una manera de formalizar el problema es por medio de una *función de aproximación*. Se asume $y = f(\mathbf{x})$ para una función desconocida f, y el objetivo es estimar la función f dado un conjunto de entrenamiento etiquetado, para posteriormente, realizar predicciones utilizando $\hat{y} = \hat{f}(\mathbf{x})$ sobre conjuntos de entrenamientos nuevos. Además, y_i suele ser una única variable que se está tratando de predecir, lo cual permite el uso de modelos univariados (con parámetros de entrada dependientes).

En casos donde no existe certeza acerca de la clasificación de un objeto, es preferible que la función retorne una probabilidad. Dicha función se denota como como $p(y|\mathbf{x}, \mathcal{D})$ e implementa una *distribución de probabilidad* sobre posibles etiquetas, donde se tienen como parámetros un vector de entrada $\mathbf{x}$ y un conjunto de entrenamiento $\mathcal{D}$. En esta notación se expone de forma explícita que la probabilidad es condicional para $\mathbf{x}$ y $\mathcal{D}$.

Volviendo al reconocimiento de monedas, la función de aproximación f debe entregar un resultado $\hat{y} \in \{0, 1\}$ tomando como insumo la base aprendida $\mathcal{D}$ y las caraterísticas de un conjunto nuevo de imágenes $\mathbf{x}$. Si se emplea una distribución de probabilidad, es posible definir el evento *cara* de interés, e $\hat{y}$ indicar la probabilidad de que la moneda la muestre dando $\mathcal{D}$ y un $\mathbf{x}$ nuevo como parámetros.

3.1.3 Aprendizaje Descriptivo

Denominado además como *aprendizaje no supervisado*, consiste en hallar patrones de interés a partir de los datos. En este caso sólo se entregan las entradas como parámetro $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$. También se denomina *descubrimiento de conocimento*, y es un problema mucho menos bien definido, donde no se tiene certeza sobre los tipos de patrones que se pueden encontrar, además de no contar con una métrica de error exacta (al contrario del aprendizaje predictivo donde se puede comparar la predicción para un determinado $\mathbf{x}$ con el valor observado).

Se procede formalizando el proceso a través de una *estimación de densidad incondicional*, esto es, construyendo modelos de la forma $p(\mathbf{x}_i|\theta)$, donde θ es el parámetro de aprendizaje. Dichos modelos son probabilísticos multivariados, al contrario del aprendizaje predictivo.

Prosiguiendo con el ejemplo anterior respecto al reconocimiento de monedas, no existen datos observados acerca de si lo que se muestra en cada ocasión es *cara* o *sello*. Solamente se obtienen las características para la imagen de cada observación i y en base a ello el proceso debe determinar si existen patrones de interés a partir de los cuales se pueda clasificar las monedas.

3.1.4 Preliminares de probabilidad

Probabilidad condicional.

Se define la probabilidad condicional de un evento A, cuando un evento B es verdadero, de la manera que sigue:

$$p(A|B) = \frac{p(A,B)}{p(B)} \text{ si } p(B) > 0. \tag{1}$$

Probabilidad conjunta.

La probabilidad de que ocurran dos eventos conjuntamente, se define de la siguiente manera:

$$p(A,B) = p(A \cap B) = p(A|B)p(B). \tag{2}$$

Esto es llamado en ocasiones como *regla del producto*. Luego, dada una *distribución conjunta* con dos eventos $p(A,B)$, es posible definir la *distribución marginal*:

$$p(A) = \sum_b p(A,B) = \sum_b p(A|B=b)p(B=b) \tag{3}$$

donde se efectúa una suma sobre todos los estados posibles de B. Es posible definir $p(B)$ de forma similar. Esto se llama la *regla de la suma* o la *regla de la probabilidad total*. Dicha regla del producto se puede aplicar múltiples veces generando la *regla de la cadena* de la probabilidad.

Regla de la cadena.

Gracias a esta regla, es posible representar una distribución de probabilidad conjunta $p(\mathbf{x})$ a partir de las probabilidades condicionales para cada una de las variables, donde V es el número de variables. La regla funciona de la siguiente manera

$$p(x_{1:V}) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)p(x_4|x_1, x_2, x_3)...p(x_V|x_{1:V-1}). \tag{4}$$

El principal problema que trae consigo esta operación es la complejidad del proceso cuando V aumenta. En el caso de que todas las variables tengan K estados se puede representar $p(x_1)$ como una tabla de $O(K)$ números representando una distribución discreta. Para representar $p(x_2|x_1)$ se necesita una tabla de $O(K^2)$ números para denotar $p(x_2 = j|x_1 = i)$. Igualmente, para describir $p(x_3|x_1, x_2)$ se necesita una tabla 3D de $O(K^3)$ números, por lo que el problema se vuelve de costo $O(K^V)$. Estas tablas se denominan *tablas de probabilidad condicional* (CPT). En el caso de $K = 2$ (variable de valores binarios) el proceso se simplifica.

3.1.5 Redes Bayesianas

Introducción a la interpretación bayesiana de la probabilidad.

Existen dos maneras de enfocar la probabilidad en el contexto del aprendizaje de máquina (Murphy, 2012). Una de ellas es la *interpretación frecuentista*, en la cual la probabilidad representa la ocurrencia de un suceso en base a experimentación sucesiva. En el caso de una moneda, la probabilidad de que al lanzarla se obtenga cara o sello se obtiene al realizar muchos intentos. Mientras más lanzamientos, la probabilidad irá tendiendo a 0.5 para cada suceso.

El segundo enfoque es la *interpretación bayesiana*. En este caso, la probabilidad es utilizada para cuantificar la incerteza sobre un suceso. Aquí se define la probabilidad en base a información concerniente al suceso. Se obtiene la probabilidad de que al lanzar una moneda salga cara en la mitad de los intentos, de acuerdo a la lógica. Como son dos sucesos, la probabilidad es 0.5. No se requiere en este enfoque realizar sucesivos intentos. Y como su nombre lo indica, está basado en el *Teorema de Bayes*, el cual se describe a continuación.

Teorema de Bayes.

Propuesto por Thomas Bayes (1702-1761), este teorema combina la probabilidad condicional con las reglas de la suma y el producto

$$p(X = x|Y = y) = \frac{p(Y = y|X = x)}{p(Y = y)} = \frac{p(Y = y|X = x)p(X = x)}{\sum_{x'} p(Y = y|X = x')p(X = x')}, \tag{5}$$

donde X e Y corresponden a *variables aleatorias discretas* que operan sobre un conjunto de eventos, y x e y representan los distintos posibles que pueden tomar X e Y.

Independencia condicional.

Frecuentemente, existe influencia entre muchas variables entre sí. Un caso típico es que dos variables influyan entre sí por medio de de una tercera. Con respecto a lo anterior, se puede decir que X e Y son *independientes condicionalmente* (CI), dada la variable Z si la probabilidad conjunta puede ser escrita como el producto de condicionales marginales:

$$X \perp Y|Z \Leftrightarrow p(X, Y|Z) = p(X|Z)p(Y|Z). \tag{6}$$

Lo anterior reviste especial importancia en la creación de grafos y redes bayesianas. En esos casos es importante conocer cuando se tiene un evento Z que depende de X e Y, pero X e Y son independientes ente sí. En ese caso, dicho supuesto se puede representar como un grafo X - Z - Y.

Modelos a través de grafos.

Un *modelo a través de grafo* es una forma de representar una distribución conjunta de probabilidades haciendo suposiciones de independencia condicional. En particular, los nodos en el grafo representan variables aleatorias, y la existencia de una arista entre dos nodos representa una dependencia condicional, mientras que su ausencia indica que existe una independencia condicional entre las dos variables. Existen varios tipos de modelos, dependiendo de si el grafo es dirigido (la probabilidad se da en un solo sentido) o no es dirigido, o bien una combinación de ambos tipos.

Como parte de la terminología básica acerca de grafos, se expone lo siguiente:

- Grafo : $G(\mathcal{V}, \mathcal{E})$ consiste en un conjunto de *nodos* o *vertices* $\mathcal{V} = \{1, ..., V\}$, y un conjunto de *aristas* $\mathcal{E} = \{(s, i) : s, i \in \mathcal{V}\}$.
- Matriz de adyacencia : Indica la existencia o no de aristas dentro del grafo. Se asigna $G(s, i) = 1$ para denotar que $(s, i) \in \mathcal{E}$, es decir, que $s \to i$ es un lado del grafo.

- Grafo no dirigido : La dependencia se da en ambos sentidos. $G(s,i) = 1$ y $G(i,s) = 1$.
- Grafo dirigido : La dependencia sólo se da en un sentido. $G(s,i) = 0$ ó $G(i,s) = 0$.
- Padre : En un grafo dirigido, los *padres* de un nodo es el conjunto de todos los nodos que lo apuntan: $pa(s) \cong \{i : G(i,s) = 1\}$.
- Hijo : En un grafo dirigido, los *hijos* de un nodo es el conjunto de todos los nodos a los cuales él apunta : $hi(s) \cong \{i : G(s,i) = 1\}$.
- Raíz : En un grafo dirigido, es un nodo sin padres.
- Hoja : En un grafo dirigido, es un nodo sin hijos.

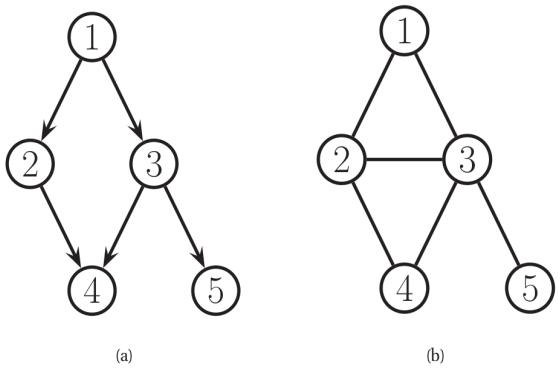

Figura 1: (a) Un grafo dirigido con cinco nodos, numerado en orden topológico. El nodo 1 es la raíz, mientras que los nodos 4 y 5 son las raíces. (b) Un grafo no dirigido. Tomada de Murphy (2012).

Descripcion de una red bayesiana.

Una red bayesiana corresponde a un tipo especial de *grafo dirigido*. Cada nodo es un *suceso*, y las aristas representan las *dependencias condicionales*. Su propiedad principal es que los nodos pueden ser ordenados de tal forma que los nodos padres pueden venir antes que los nodos hijos. Esto se denomina ordenamiento topológico. Dado un ordenamiento, es factible definir la *propiedad ordenada de Markov* que corresponde a la suposición de que una nodo solamente depende de sus dos padres, y no de todos sus predecesores:

$$x_s \perp \mathbf{x}_{\mathrm{pred}(s)\setminus\mathrm{pa}(s)} | \mathbf{x}_{\mathrm{pa}(s)}, \tag{7}$$

donde pa(s) son los padres del nodo s, y pred(s) son los predecesores del nodo s en el ordenamiento. Esta es la generalización de la propiedad de Markov de primer orden desde las cadenas de Markov a los grafos dirigidos.

En general se tiene que

$$p(\mathbf{x}_{1:V}|G) = \prod_{i=1}^{V} p(x_i|\mathbf{x}_{\mathrm{pa}(i)}), \tag{8}$$

donde V es el número de vértices, G indica que los supuestos de independencia condicional del grafo deben ser correctos y cada termino $p(x_i|\mathbf{x}_{\mathrm{pa}(i)})$ es una distribución de probabilidad condicional.

3.2 Modelo matemático

Se propone un modelo predictivo de contexto, capaz de variar de acuerdo a la imagen de entrada. El número de imágenes a procesar se denota como i y el número de objetos factibles de ser reconocidos, como n. Se trabajará con datos binarios que indican la existencia real de cada uno de los objetos para cada una de las imágenes, los que se denotarán como $\beta^0 \in \{0,1\}^n$. De esta forma, un valor de 1 en la entrada (i,n) indica que el objeto n se encuentra en la imagen i, mientras que un valor de 0 en dicha entrada indica que no se encuentra en la imagen. Luego, cada objeto corresponderá a un nodo de la red bayesiana, que tendrá forma de árbol. Para obtener la probabilidad de aparición de los objetos, es decir, las conexiones entre los nodos, es necesario considerar cuatro escenarios que se pueden dar entre dos objetos arbitrarios a y b

- La existencia del objeto a en la imagen i, cuando b también está presente en ella.
- La existencia del objeto a en la imagen i, cuando b no está presente en ella.
- La no existencia del objeto a en la imagen i, cuando b está presente en ella.
- La no existencia del objeto a en la imagen i, cuando b no está presente en ella.

Por consiguiente, se tienen cuatro combinaciones lógicas entre dos objetos. La matriz de probabilidades, por tanto, se obtendrá calculando las co-ocurrencias entre el total de objetos, para el total de imágenes. Esto quiere decir, que se contará en cuantas imágenes ocurre cada uno de estos casos, tomando en cuenta todos los pares posibles entre los objetos. El resultado será una matriz de orden $2n$ y se empleará como constante para todas ellas. Los cambios en la red bayesiana estarán dados por las características de cada imagen. De esta manera,

$$\tau_i^*(\beta^0, X_i, \theta) = \underset{\tau}{\operatorname{argmax}}\ \tau(cc(\beta^0) + X_i\theta), \tag{9}$$

donde τ_i^* corresponde al árbol de dependencia entre objetos calculado para la imagen i

de acuerdo al algoritmo explicitado en Chow y Liu (1968). Además, $cc(\beta^0) \in \mathbb{N}^{2n,2n}$ calcula las co-ocurrencias entre los objetos, considerando las cuatro combinaciones lógicas binarias

- $X \in \mathbb{R}^{i,4n}$ es una matriz correspondiente al total de características para el conjunto de imágenes. Asimismo, X_i consiste en el vector de características de la imagen i de dimensionalidad $4n$.
- $\theta \in \mathbb{R}^n$ es el vector de ponderación para cada uno de los objetos reconocibles.

El vector de características tiene dimensión de $(1, 4n)$ debido a que efectuando una transformación sobre él, se genera una matriz de orden $2n$ capaz de operar con las que genera $cc(\beta^0)$.

A su vez,

$$b_i^*(\tau_i^*(\beta^0, X_i, \theta)) = \underset{b}{\operatorname{argmax}}\ p(b; \tau_i^*(\beta^0, X_i, \theta)), \tag{10}$$

donde $b_i^* \in \{0, 1\}^n$ consiste en la aproximación de la existencia de cada uno de los objetos para la imagen i expresada en forma binaria. Como resultado, se denota el modelo matemático de la siguiente forma

$$f(\beta^0, X_i, \theta) = b_i^*(\tau_i^*(\beta^0, X_i, \theta)), \tag{11}$$

el cual opera sobre cada imagen i. A partir de ahora, f reemplazará la notación b_i^* por razones de simplicidad.

3.3 Solución numérica

Con el fin de lograr una buena aproximación de la existencia de los objetos, se requiere minimizar el error entre la existencia real de los objetos β^0 y la aproximada f. Para esto, se propone generar una nueva ecuación empleando dos métodos, el de mínimos cuadrados y el de minimización mediante heurística.

3.3.1 Minimización a través de mínimos cuadrados

Se trata de un método de estimación de parámetros, en este caso θ

$$\underset{\theta}{\text{mín}} E(\beta^0, X, \theta) = \underset{\theta}{\text{mín}} \sum_{i=1}^{n} \|\beta_i - f(\beta^0, X_i, \theta)\|_2^2, \tag{12}$$

donde $\beta_i \in \{0,1\}^n$ indica la existencia real del objeto n para la imagen i y se busca la reducción del error cuadrático. $\|\cdot\|_2$ consiste en la norma euclidiana entre β_i y $f(\beta^0, x_i, \theta)$. Se busca que x influya en el mayor número posible de estadísticos.

Para obtener el mínimo de E se deriva con respecto a θ igualando a 0:

$$0 = \sum_{i=1}^{n} \left\{ 2(\beta_i - f(\beta^0, X_i, \theta)) \frac{\partial f(\beta^0, X_i, \theta)}{\partial \theta} \right\}.$$

3.3.2 Minimización por medio de heurística

Corresponde a otro método de minimización

$$\min_{\theta} E(\beta^0, X, \theta) = \min_{\theta} \sum_{i=1}^{n} \|A(\beta_i, f(\beta^0, X_i, \theta))(\beta_i - f(\beta^0, X_i, \theta))\|_2, \tag{13}$$

donde $A(\beta_i, f(\beta^0, X_i, \theta))$ es la función heurística que guía la minimización. El objetivo de dicha función es lograr que el modelo logre detectar la mayor cantidad de objetos posibles que existan en la imagen, para lo cual se premia la aparición de objetos en la aproximación b_i^* ($b_i^n = 1$) y se castiga con un peso más elevado si un objeto existe en la realidad y luego no es reconocido en la aproximación ($\beta_i^n = 1$ cambia a $b_i^n = 0$).

Por consiguiente, la función heurística queda denotada de la siguiente forma

$$A_{ii} = \begin{cases} 1, & \beta_i = 0 \\ peso, & \beta_i = 1; \end{cases} \tag{14}$$

donde $peso \in \{5, 10, 20\}$. Luego, igual que en la minimización por mínimos cuadrados, se obtiene el mínimo de E derivando con respecto a θ e igualando a 0:

$$\begin{aligned} \frac{\partial E}{\partial \theta} &= \frac{\partial}{\partial \theta} \sum_{i=1}^{n} \{\|A(\beta_i, f(\beta^0, X_i, \theta))(\beta_i - f(\beta^0, X_i, \theta))\|_2\} \\ 0 &= \sum_{i=1}^{n} \left\{ diag(A(\beta_i, f(\beta^0, X_i, \theta)))^T \frac{\partial f(\beta^0, X_i, \theta)}{\partial \theta} \right\}. \end{aligned}$$

3.3.3 Descripción del método numérico

En ambas formas, el término $\frac{\partial}{\partial\theta}f(\beta^0, x_i, \theta)$ no es diferenciable analíticamente. Es por esta razón que se debe diferenciar numéricamente por medio de diferencias finitas centradas:

$$\frac{\partial f(\beta^0, X_i, \theta)}{\partial\theta} \cong \frac{f(\beta^0, X_i, \theta + \overrightarrow{\epsilon}) - f(\beta^0, X_i, \theta - \overrightarrow{\epsilon})}{2\epsilon}, \tag{15}$$

donde ϵ es constante con valor 0,1 y $\overrightarrow{\epsilon} = \epsilon \times e_i$, $e_i = [0\ 0 \cdots 1 \cdots 0\ 0]^T$ con la entrada 1 en la i-ésima posición.

Para la obtención del θ óptimo se recurre al método de Newton-Raphson para aproximarlo mediante sucesivas iteraciones. Se utilizará la variante *Newton-Raphson inexacto disminuido* de acuerdo a lo señalado por Niessner et al. (2005). En su versión original opera de la siguiente manera:

Elegir $\mathbf{u}^{n+1,0}$

Fijar $r = 0$

MIENTRAS $\frac{||\mathbf{f}(\mathbf{u})^{n+1,r}||_2}{||\mathbf{f}(\mathbf{u})^{n+1,0}||_2} > \epsilon_{nl}$ REPETIR

Encontrar $\mathbf{s}_n$ tal que $\mathbf{K}(\mathbf{u}^{n+1,r})\mathbf{s}_n = -\mathbf{f}(\mathbf{u}^{n+1,r})$ con un error inferior a ϵ_{lin}

Hallar η adecuado

$\mathbf{u}^{n+1,r+1} = \mathbf{u}^{n+1,r} + \eta\mathbf{s}_n$

$r = r + 1$

FIN MIENTRAS

Luego,

$$\mathbf{K}(\mathbf{u}^{n+1,r})\mathbf{s}_n = -\mathbf{f}(\mathbf{u}^{n+1,r}) \tag{16}$$

corresponde a la versión lineal del método, donde $\mathbf{K}(\mathbf{u}^{n+1,r}) = \left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right)^{-1}_{n+1,r}$ y $\mathbf{s}_n = \mathbf{u}^{n+1,r+1} - \mathbf{u}^{n+1,r}$. Además, $\mathbf{f}(\mathbf{u})^{n+1,r}$ representa el vector lado derecho en el instante de tiempo $n+1$ y en el nivel de iteración (elemento de sucesión convergente) r. La constante ϵ_{nl} corresponde a la cota de error para la solución del sistema no lineal, mientras que ϵ_{lin} consiste en la cota de error para la solución del sistema lineal.

Para generar $\mathbf{u}^{n+1,r+1}$, se debe escoger el número más pequeño $q \in \{0, 1, ..., 50\}$ para el factor $\eta = \left(\frac{1}{2}\right)^q$ tal que

$$||\mathbf{f}(\mathbf{u}^{n+1,r+1})||_2 \leq \left[1 - \frac{1}{4}\eta\right] ||\mathbf{f}(\mathbf{u}^{n+1,r})||_2. \tag{17}$$

En la implementación del modelo que se está presentando en este trabajo se ha de simplificar este método. Puesto que no se requiere resolución de sistema lineal, sólo se necesita definir ϵ_{nl}. A su vez, el descenso de gradiente se aplicará con una leve modificación

$$\theta^{(t+1)} = \theta^{(t)} - \alpha\eta\frac{\partial E}{\partial \theta}, \tag{18}$$

donde α se inicializa en 0,01 para suavizar el descenso y obtener un número razonable de iteraciones. Este valor se irá disminuyendo a medida que el método se va acercando al óptimo. El objetivo es generar valores sucesivos de θ hasta que $||\theta^{(t+1)} - \theta^{(t)}||_2^2 < \epsilon_{nl}$, $\epsilon_{nl} = 10^{-5}$, reemplazando el error relativo original. Logrado esto, $\theta^{(t+1)}$ será el vector de ponderaciones definitivo. En suma, el método *Newton-Raphson inexacto disminuido* modificado para este modelo es

Elegir $\theta^{(t)}$

Fijar $r = 0$, $\alpha = 0,01$

MIENTRAS $||\theta^{(t+1)} - \theta^{(t)}||_2^2 < 10^{-5}$ REPETIR

Encontrar $\dfrac{\partial E(\beta^0, X, \theta)}{\partial \theta}$

Hallar η adecuado

$\theta^{(t+1)} = \theta^{(t)} - \alpha\eta\dfrac{\partial E}{\partial \theta}$

$\alpha = \alpha * \eta$

$r = r + 1$

FIN MIENTRAS

Y para definir el η correspondiente se elige el número más pequeño $q \in \{0, 1, ..., 50\}$ para el factor $\eta = \left(\frac{1}{2}\right)^q$ tal que

$$E(\beta^0, X, \theta^{(t+1)}) \leq \left[1 - \frac{1}{4}\eta\right] E(\beta^0, X, \theta^{(t)}), \tag{19}$$

donde no se calcula la norma euclidiana sobre E debido a que dicha función ya la incluye en su evaluación.

3.4 Simulación computacional

Con respecto a la implementación, la obtención de resultados será mediante programación en MATLAB a través de modos secuencial y paralelo. Se trabajará empleando la base de datos de contexto ubicada en Choi et al. (2010), utilizando el grupo de entrenamiento para ejecutar el modelo adaptativo, y evaluando el resultado con el grupo de testeo. El grupo de entrenamiento estará integrado por 4367 imágenes, mientras que el de testeo, de 4317. De las imágenes del grupo de entrenamiento se seleccionarán de forma aleatoria 30 en total para la extracción de características, esto por limitaciones dadas por el recurso de cómputo.

En cuanto a los recursos disponibles para la ejecución del modelo matemático y el algoritmo numérico, se utilizará un servidor de las siguientes características:

- Servidor HP Proliant DL180 Gen9.
- 2 procesadores Intel Xeon SixCore E5-2609 (1,9 GHz, 15 MB cache L3 c/u).
- 80 GB de memoria RDIMM DDR4 (ampliable a 512 GB).
- 2 discos HP SATA 1 TB 7200 RPM espejados (RAID 1).

Como se expuso anteriormente, el modelo requerirá la extracción de características para cada imagen, esto tanto en el caso del entrenamiento como en el de testeo. En un primer caso, se empleará la función *HOG* de la librería *VLFeat*. En un segundo caso, se extraerán dichas características mediante Aprendizaje Profundo utilizando el marco de trabajo *Caffe*.

Caffe es un marco de trabajo desarrollado por Yangquing Jia durante su estudios de doctorado en la UC Berkeley. Opera bajo el concepto de *Aprendizaje Profundo*, ya que es un modelo composicional que es representado como una conexión de capas interconectadas que operan sobre porciones de datos. Esta interconexión se puede expresar como una red que define dicho modelo desde la entrada de datos hasta la salida de ellos. Los datos fluyen a través de la red por medio de pasos hacia adelante y hacia atrás, por lo que Caffe emplea el tipo de dato masa, para el almacenamiento, comunicación y manipulación de la información, debido a que éste tiene menos restricciones para su uso. Un tipo de dato masa, es una envoltura de datos que es procesada por *Caffe*, y bajo la cubierta provee capacidad de sincronización entre el procesador y la tarjeta gráfica. Matemáticamente, es un vector N-dimensional almacenado en memoria contigua empleando lenguaje de programación C. Por otro lado, cada capa se transforma en el fundamento del proceso de cálculo.

Para la extracción de características mediante *Caffe* se hará uso del siguiente equipo:

- Procesador: Intel CoreTM i7-4790K CPU (4.00GHz, 8 núcleos).
- Memoria del Sistema: 16 GB (DDR3).
- Tarjeta Gráfica: NVIDIA GeForce GTX 970/PCIe/SSE2.
- Disco: 2 TB.

El cambio de PC se justifica debido al uso de una tarjeta gráfica (en este caso GeForce GTX 970) en la ejecución de *Caffe*. Esto es conveniente ya que *Caffe* posee una arquitectura de cálculo paralelo CUDA, la cual proporciona una reducción del tiempo de cálculo.

Para implementar el descenso de gradiente empleado en el método de Newton-Raphson inexacto disminuido, y así obtener $\theta^{(t+1)}$, se utiliza el siguiente procedimiento

PARA $q \in 0, 1, 2, 3, 4, 5, 6$ REPETIR

$\eta \leftarrow \left(\frac{1}{2}\right)^q$

$\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha\eta\dfrac{\partial E}{\partial \theta}$

$error_t \leftarrow E(\beta^0, X, \theta^{(t+1)})$

$error_{t+1} \leftarrow E(\beta^0, X, \theta^{(t)})$

SI $(error_{t+1} \leq (1 - 10^{-4} * \eta) * error_t)$ ENTONCES

$\theta^{(t+1)}$ es óptimo

$\alpha \leftarrow \alpha * \eta$

Salir PARA

FIN SI

FIN PARA

Luego, se debe tener en cuenta que para la extracción de características en HOG se debe hacer la conversión de dimensiones desde la matriz que entrega la función (32*32*31) a un vector de longitud $4n$. Para esto se lleva a cabo un procedimiento recursivo empleando un tamaño de celda de 2. Finalmente, cuando ya no se puede seguir reduciendo por medio de la función, se seleccionan los valores de forma aleatoria. El procedimiento es el siguiente:

FUNCION *ExtraerFeatures*(identificacion_objetos,identificacion_imagenes)

Inicializar VLFeat

$N \leftarrow$ Obtener numero de objetos

$M \leftarrow$ Obtener numero de imágenes

Obtener ruta donde se encuentran las imágenes

$cellSize \leftarrow 2$

$feat \leftarrow$ vector de ceros de dimensiones $M \times 4n$

PARA k desde 1 hasta M REPETIR

Obtener nombre de la imagen

$currentimage \leftarrow$ Leer la imagen

$resized \leftarrow$ imresize($currentimage$,[256 256])

$hog \leftarrow$ vl_hog(im2single($resized$),$cellSize$)

$dim_hog \leftarrow 128$

MIENTRAS $31 * (dim_hog/2)^2 >= 4n$ HACER

$dim_hog \leftarrow dim_hog/2$

$hog \leftarrow$ vl_hog(hog,$cellSize$)

FIN MIENTRAS

$hog \leftarrow$ ConvertirAVector(hog)

PARA j desde 1 hasta $31 * dim_hog^2 - 4n$ REPETIR

Eliminar $hog(random(31 * dim_hog^2 + 1 - j))$

FIN PARA

$feat(k, :) \leftarrow 2{,}5 * hog$

FIN PARA

Devolver $feat$

FIN *ExtraerFeatures*

Finalmente, en la extracción de características utilizando Aprendizaje Profundo sobre el marco de trabajo *Caffe* se debe tener en cuenta que el vector de características obtenido para cada imagen contiene 4096 elementos. Por medio del algoritmo *Random Subspace* se reducirá cada vector a la dimensión $4n$ que se requiere. El proceso se

expone a continuación:

FUNCION *RandomSubspace*($features$,n)

  $M \leftarrow$ Obtener numero de imagenes

  $size_original \leftarrow$ Obtener la longitud del vector de características para cada imagen

  Inicializar vector *feat* como vector vacío

  PARA i desde 1 hasta M REPETIR

    $positions \leftarrow$ Obtener $4n$ posiciones aleatorias dentro de $size_original$

    $ft \leftarrow$ vector de ceros de largo $4n$

      PARA j desde 1 hasta $4n$ REPETIR

        $ft(j) \leftarrow features(i, positions(j))$

      FIN PARA

    Agregar *ft* a *feat*

  FIN PARA

  Devolver *feat*

FIN *RandomSubspace*

4 Resultados

En este capítulo se analiza la implementación de (11) a través de las variantes (12) y (13), comparando los resultados obtenidos en ambas modalidades.

4.1 Ajuste de f por medio de mínimos cuadrados.

En las figuras 2 y 3 se compara la evolución de la minimización de la alternativa (12) en la etapa de entrenamiento, extrayendo características para cada imagen tanto por *HOG* como por Aprendizaje Profundo.

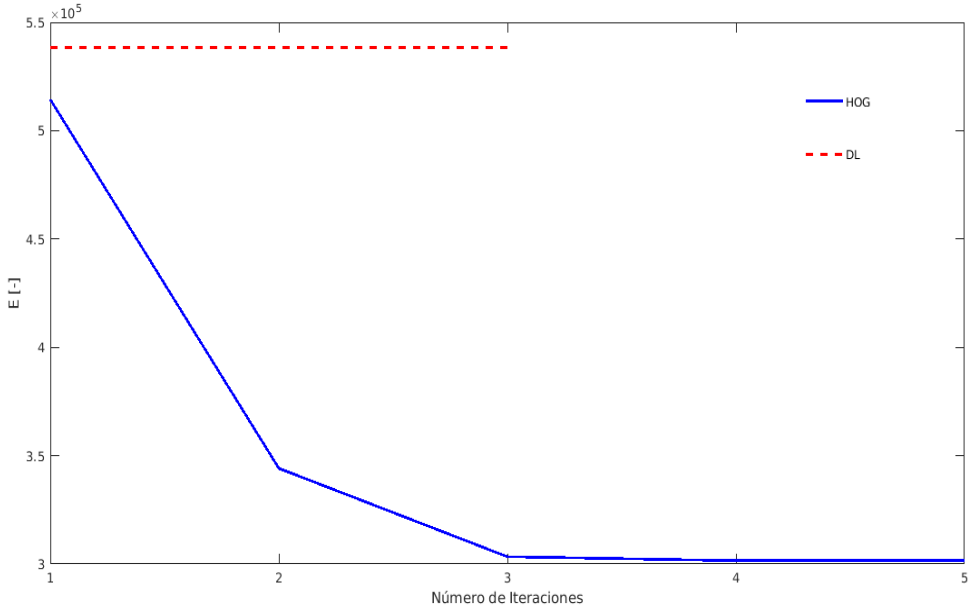

Figura 2: Evolución de el valor de E al emplear *HOG* y Aprendizaje Profundo en la etapa de entrenamiento, utilizando mínimos cuadrados. Valores absolutos.

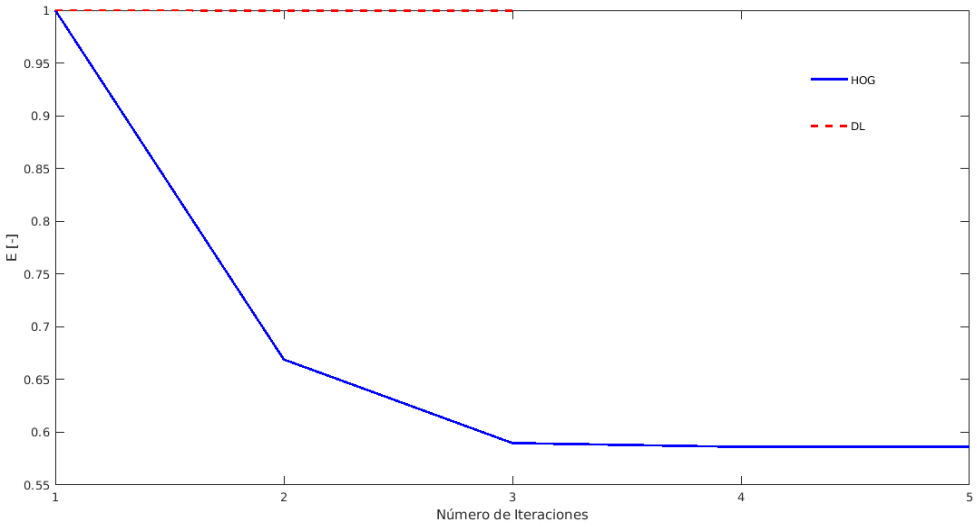

Figura 3: Evolución de el valor de E al emplear HOG y Aprendizaje Profundo en la etapa de entrenamiento, utilizando mínimos cuadrados. Comportamiento a partir del valor inicial de E.

Como se puede apreciar, utilizando HOG se obtiene una convergencia entre los valores reales de aparición y los pronosticados para cada imagen, superior al 40 %. Esto tomando en cuenta el conjunto de entrenamiento. Para el Aprendizaje Profundo no hubo mejora para el modelo predictivo. Ahora, en las figuras 4 y 5 se compara para ambas metodologías de extracción de características el comportamiento del ajuste de mínimos cuadrados para el conjunto de testeo.

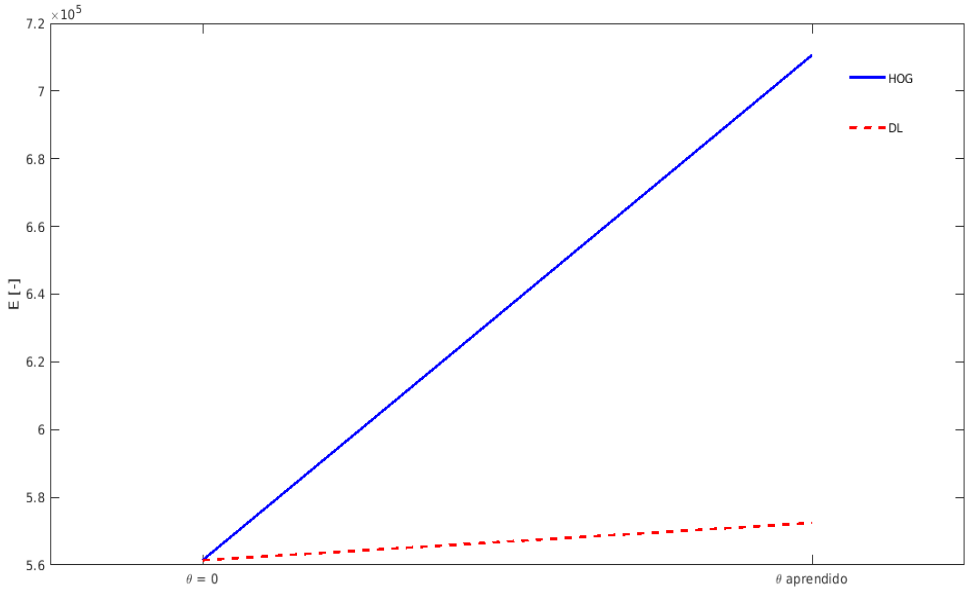

Figura 4: Evolución de el valor de E al emplear HOG y Aprendizaje Profundo en la etapa de testeo, utilizando mínimos cuadrados. Valores absolutos.

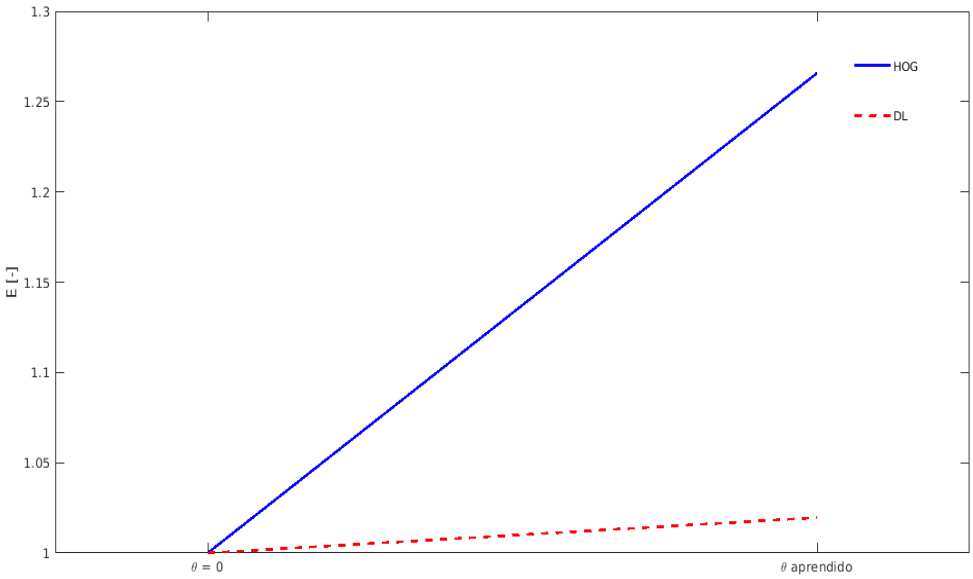

Figura 5: Evolución de el valor de E al emplear HOG y Aprendizaje Profundo en la etapa de testeo, utilizando mínimos cuadrados. Comportamiento a partir del valor inicial de E.

Se puede deducir en base a lo anterior, que en ambos casos, con el θ obtenido de la etapa de entrenamiento existe un aumento del error entre los valores reales de existencia de objetos y los pronosticados. En el caso de Aprendizaje Profundo el aumento es menor debido a la escasa convergencia que hubo en la etapa de entrenamiento. Por consiguiente, el ajuste de mínimos cuadrados no permite obtener resultados replicables en otras bases de datos orientadas a contexto distintas a la de entrenamiento.

4.2 Ajuste de f por medio de heurística

En este instante, se expone el desempeño de la heurística para lograr una mayor exactitud en el resultado obtenido a partir del modelo. Se hará la comparación para distintos pesos a evaluar en (14), en este particular $peso \in \{5, 10, 20\}$. La idea es ver cuánto se debe premiar la aparición de nuevos objetos en la aproximación en cada imagen por sobre la información de objetos ya detectados en las mismas imágenes o en otras.

4.2.1 Comparación del comportamiento de distintos pesos empleando *HOG*

Se evalúa el desempeño del modelo dinámico para los pesos señalados utilizando *HOG*. El resultado se visualiza en las figuras 6 y 7.

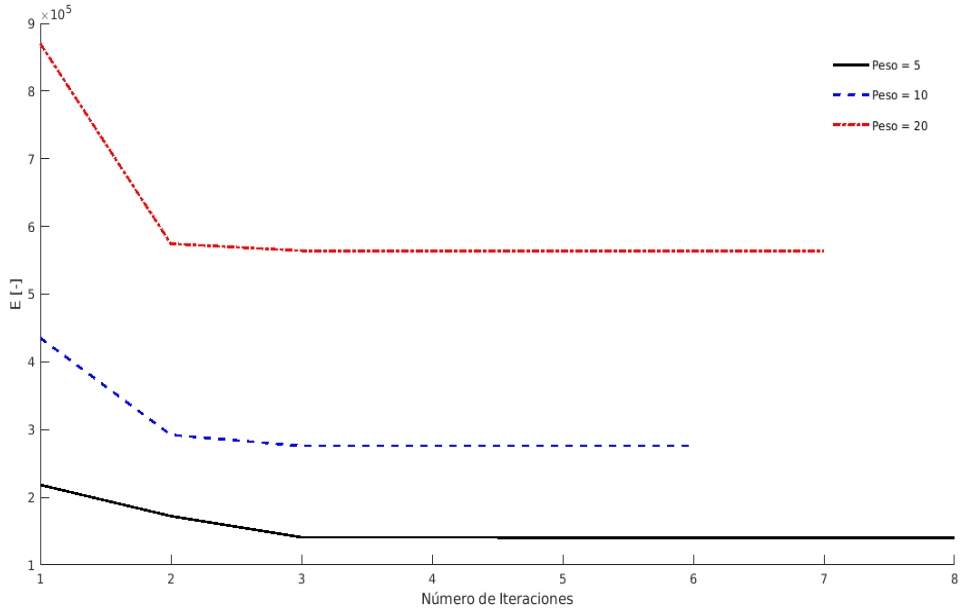

Figura 6: Evolución de el valor de E al emplear *HOG* para distintos pesos en la heurística, en la etapa de entrenamiento. Valores absolutos.

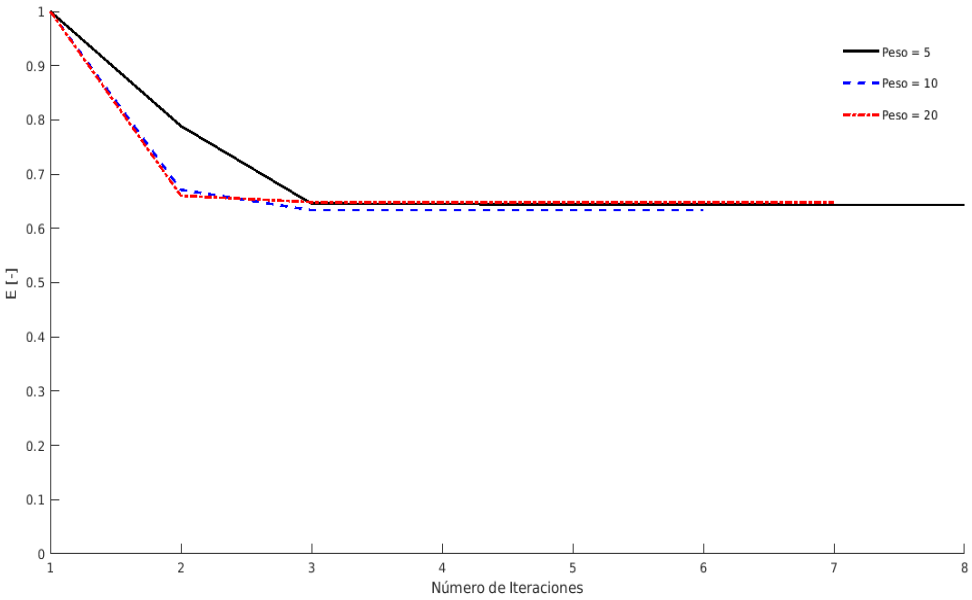

Figura 7: Evolución de el valor de E al emplear HOG para distintos pesos en la heurística, en la etapa de entrenamiento. Comportamiento a partir del valor inicial de E.

Como se puede visualizar, en el conjunto de entrenamiento se genera una reducción del error de la predicción mayor a un 30 % tomando en consideración las tres alternativas de pesos. La variación de la reducción entre los pesos no es significativa. Prosiguiendo, en las figuras 8 y 9 se muestra el resultado de la utilización de los pesos de la heurística para la mejora de la predicción, esta vez con el conjunto de testeo.

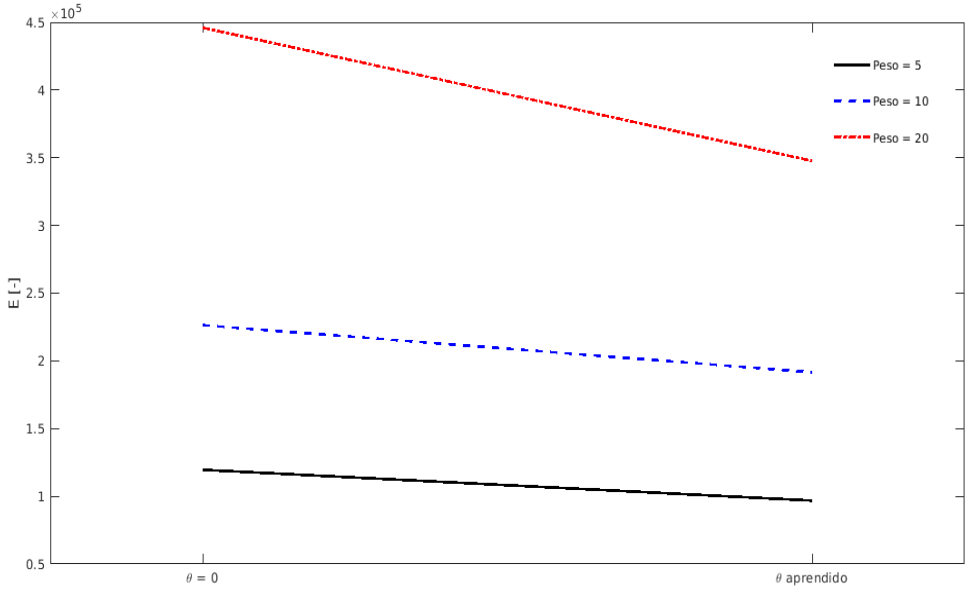

Figura 8: Evolución de el valor de E al emplear HOG para distintos pesos en la heurística, en la etapa de testeo. Valores absolutos.

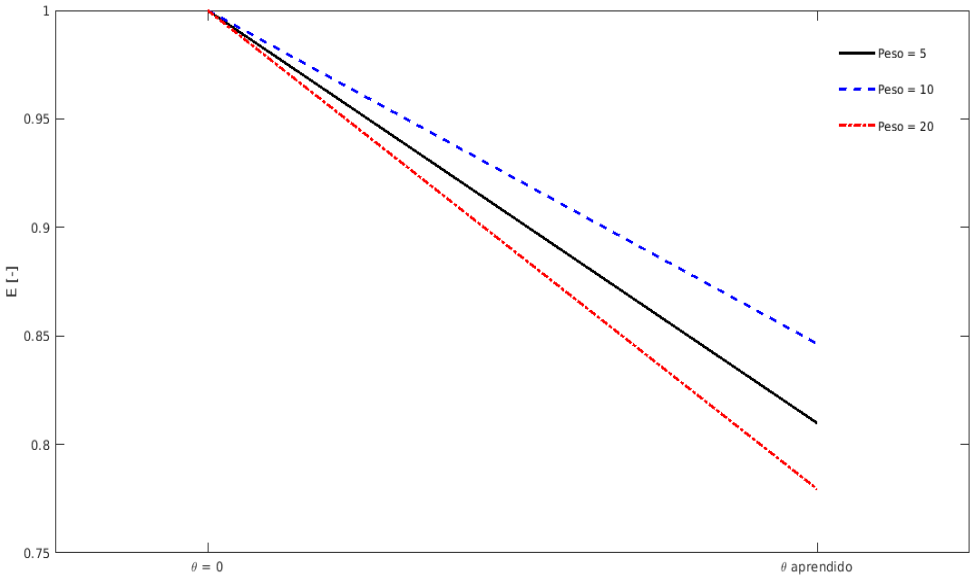

Figura 9: Evolución de el valor de E al emplear HOG para distintos pesos en la heurística, en la etapa de testeo. Comportamiento a partir del valor inicial de E.

Se observa que para el conjunto de testeo se genera una reducción del error predictivo en torno a un 20 %. El peso $A_{ii} = 20$ es el que genera la reducción mayor, lo que indicaría a primera vista que se debe premiar la detección de nuevos objetos en la predicción. Sin embargo, no existe una correlación entre el valor del peso y la mejora de la aproximación para $A_{ii} = 5$ y $A_{ii} = 10$, por lo que se debe probar con pesos mayores y menores para evaluar un posible rango de inestabilidad en los resultados.

4.2.2 Comparación del comportamiento de distintos pesos empleando Aprendizaje Profundo

Se muestra la operación del modelo para los tres pesos en evaluación utlizando Aprendizaje Profundo. En las figuras 10 y 11 se detalla cómo opera.

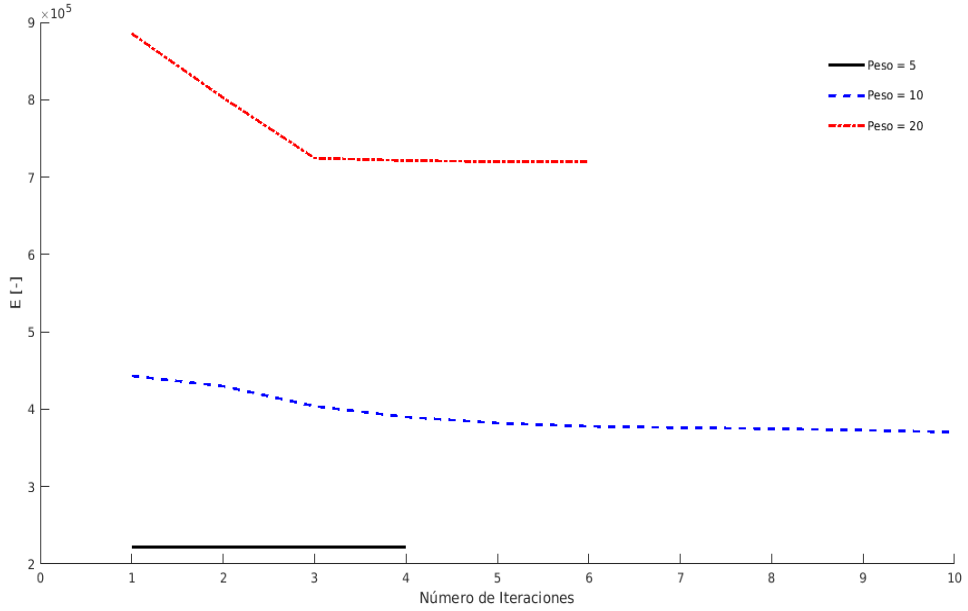

Figura 10: Evolución de el valor de E al emplear Aprendizaje Profundo para distintos pesos en la heurística, en la etapa de entrenamiento. Valores absolutos.

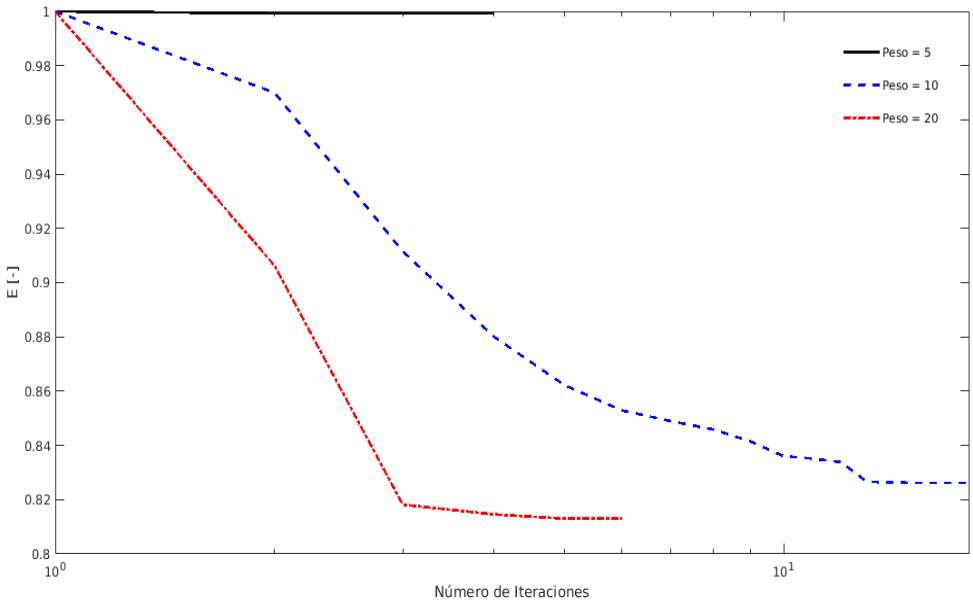

Figura 11: Evolución de el valor de E al emplear Aprendizaje Profundo para distintos pesos en la heurística, en la etapa de entrenamiento. Comportamiento a partir del valor inicial de E.

Se visualiza una mejora de la predicción en torno a un 18 % para el peso más alto $A_{ii} = 20$. Existe una clara relación entre el peso empleado y la mejora del resultado. En este caso Aprendizaje Profundo sí responde al premio que se entrega a la nueva información que entrega la aproximación. A continuación, en las figuras 12 y 13 se comparan los pesos de la heurística para la mejora de la predicción, con el conjunto de testeo.

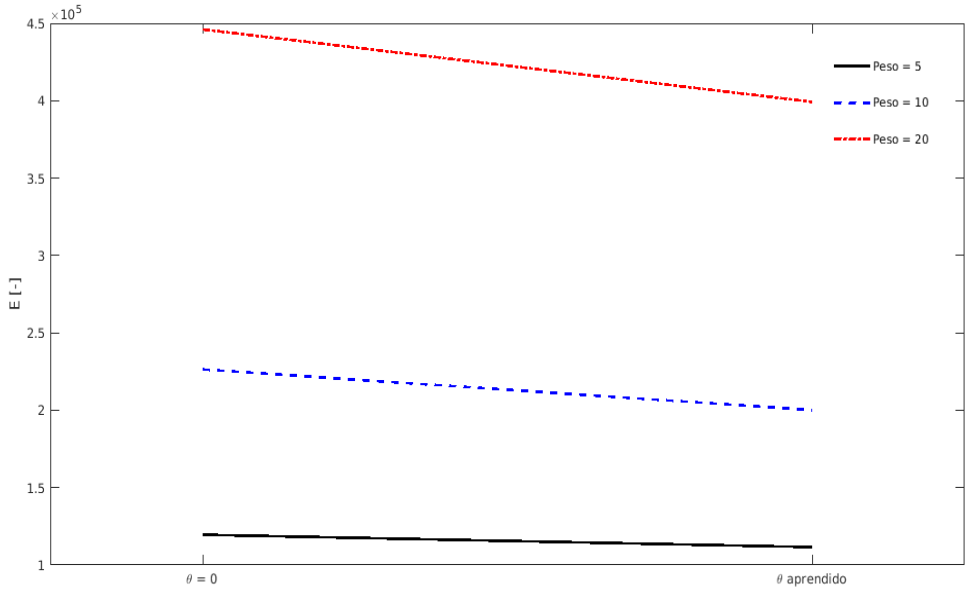

Figura 12: Evolución de el valor de E al emplear Aprendizaje Profundo para distintos pesos en la heurística, en la etapa de testeo. Valores absolutos.

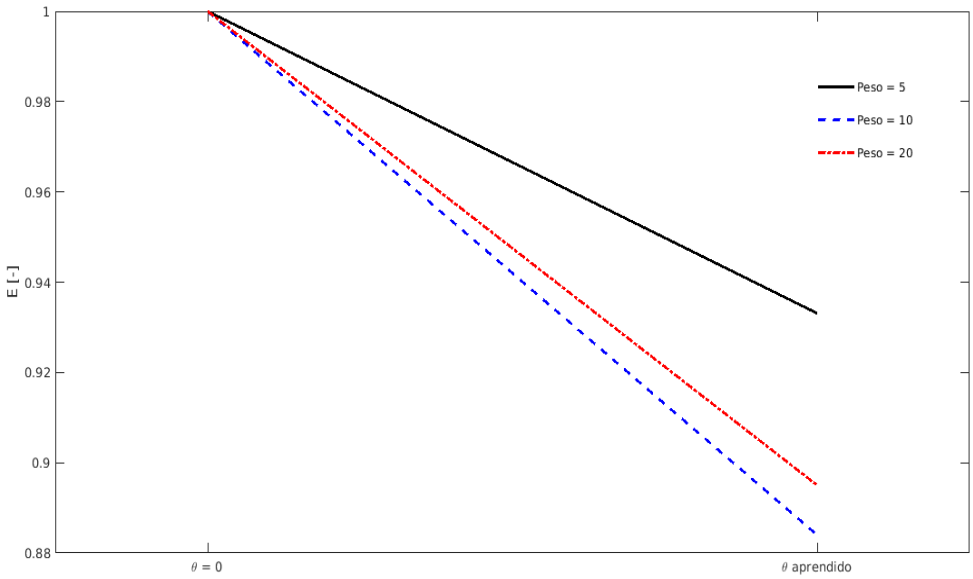

Figura 13: Evolución de el valor de E al emplear Aprendizaje Profundo para distintos pesos en la heurística, en la etapa de testeo. Comportamiento a partir del valor inicial de E.

Para el conjunto de testeo se genera una reducción del error predictivo en torno a un 10 %. No existe una relación estricta entre el valor del peso y la mejora de los resultados. Sin embargo, existe una diferencia clara entre los pesos más altos ($A_{ii} = 10$ y $A_{ii} = 20$) con el peso más bajo. Esto indica que en la etapa de testeo, si bien no se conserva la relación entre valor de peso y reducción de error de predicción de forma estricta, sí se mantiene un sesgo a favor de los pesos más altos, sólo que a partir de cierto límite el aporte no es destacable.

4.2.3 Análisis de convergencia

Se compara la velocidad de la convergencia del modelo en la etapa de entrenamiento, entre los distintos pesos. El proceso se replica tanto para *HOG* como para el Aprendizaje Profundo. Esto reviste de importancia para evaluar cual alternativa es la más eficiente en términos de recursos computacionales y cuánto, para determinar si se debe priorizar exactitud o economía de tiempo de cómputo.

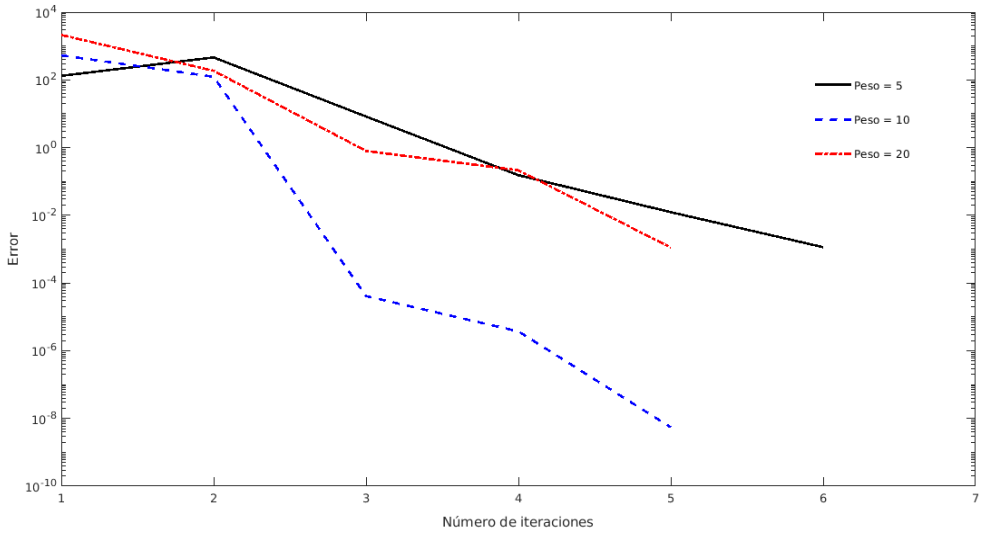

Figura 14: Evolución del error numérico al emplear distintos pesos para la heurística, utilizando características extraídas mediante *HOG*.

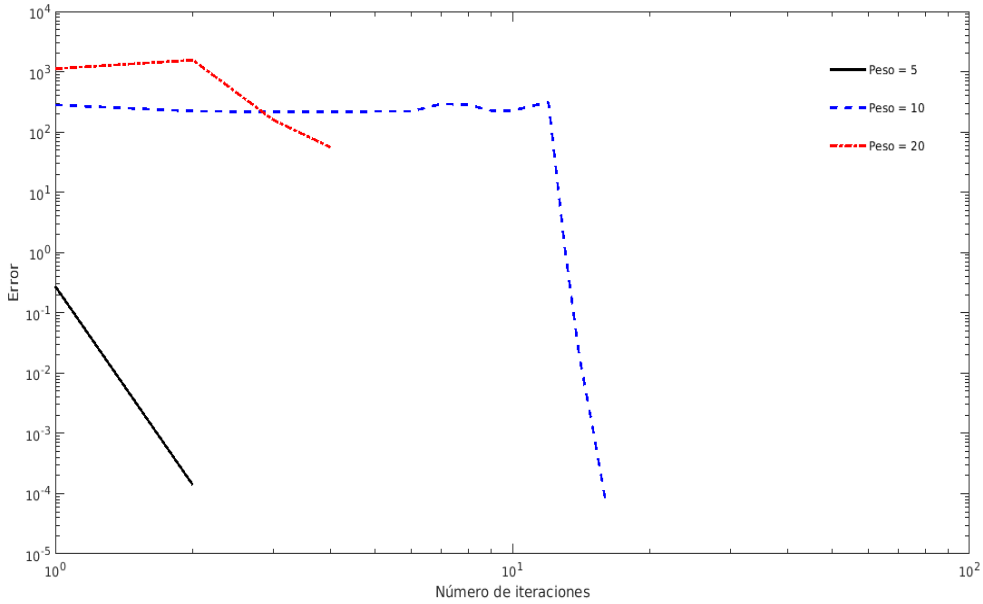

Figura 15: Evolución del error numérico al emplear distintos pesos para la heurística, utilizando características extraídas mediante Aprendizaje Profundo.

El error corresponde a la norma euclidiana al cuadrado de la diferencia entre los θ obtenidos en dos iteraciones consecutivas

$$||\theta^{(t+1)} - \theta^{(t)}||_2^2. \tag{20}$$

Se observa que para *HOG* la convergencia es más rápida con el peso $A_{ii} = 10$, siendo poco significativa la diferencia entre las otras dos alternativas. Por el contrario, empleando Aprendizaje Profundo la convergencia es más rápida con el peso más pequeño ($A_{ii} = 5$). En ambos casos las alternativas más económicas en cuanto a los recursos computacionales son las de peor desempeño en la etapa de testeo.

5 Conclusiones

Los resultados obtenidos a partir del trabajo permiten concluir que es factible implementar un modelo probabilistico adaptativo de contexto. Sin embargo, no todos los métodos para lograr el ajuste son válidos. El método de mínimos cuadrados no permite una minimización del error de predicción, sino que más bien lo aumenta, siendo este proceso evidente en el testeo. En el caso de la minimización a través de una función heurística sí se tiene una mejora en la predicción, aunque ésta va a estar condicionada al método de extracción de características y al peso con el que realice la búsqueda del óptimo.

Con respecto a los métodos de extracción de características, se concluye que el histograma de gradientes orientados (HOG) es el que permite disminuciones de error significativas, llegando a poco más de un 40 % en el proceso de entrenamiento y a un 20 % en el de testeo. Son resultados que permiten aplicar el modelo en otras bases de imágenes distintas a las de entrenamiento y testeo. Sin embargo, también cabe señalar que este método, con la implementación propuesta, no responde a los variaciones en el peso de heurística, es decir, independientemente de su valor el resultado es similar. Lo anterior ocurre con el método de reducción de dimensionalidad a partir de la enorme cantidad de datos que entrega HOG. Es posible que aumentando el volumen de datos empleado, si se pueda obtener cambios modificando el peso con el que opera la heurística.

En el caso del Aprendizaje Profundo se tiene que las mejoras en la exactitud del modelo son del orden de la mitad de las que se logran por medio de HOG. No obstante, tiene la virtud de responder a los cambios de propuestas de heurísticas. Al igual que en el caso anterior, realizar la conversión desde un gran volumen de datos a un vector de dimensión $4n$, donde el número de objetos factibles de ser reconocidos es bajo, no captura todos los patrones de cada imagen. Se recomienda mejorar el algoritmo de *Random Subspace* o reemplazarlo.

En relación al costo computacional, cabe destacar que fue necesario definir una muestra aleatoria de 30 imágenes a partir del conjunto de entrenamiento para la extracción de características. Sólo de esta forma se pudo ejecutar las simulaciones de acuerdo a los recursos disponibles y obtener resultados de interés.

Para finalizar, las perspectivas de este trabajo van en la dirección de realizar pruebas con más bases de datos orientadas a contexto, para así tener mayor claridad con respecto a los resultados. Por otra parte, se pueden integrar otras variables al modelo, como por ejemplo la localización esperada de un objeto dentro de la imagen, entre otros. Y por último, proponer heurísticas más complejas.

6 Referencias

[1] Cariaga, E., Vergara-Fernández, A., Lévano, M. y Vergaray, N. (2013). Numerical simulation of the water saturation on interface in a heterogeneous porous medium, *Revista Mexicana de Ingeniería Química*, *12*(3), 527-539.

[2] Choi, M. J., Lim, J. J., Torralba, A. y Willsky, A. S. (2010). *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. San Francisco, California, Estados Unidos.

[3] Choi, M. J., Lim, J. J., Torralba, A. y Willsky, A. S. (2010). Exploiting Hierarchical Context on a Large Database of Object Categories. Cambridge, Massachusetts, Estados Unidos: *MIT Computer Science and Artificial Intelligence Laboratory*. Recuperado de http://people.csail.mit.edu/myungjin/HContext.html.

[4] Chow, C. K. y Liu, C. N. (1968). Approximating Discrete Probability Distributions with Dependence Trees, *IEEE Transactions on Information Theory*, *14*(3), 462-467.

[5] Desai, C., Ramanan, D. y Fowlkes, C. (2009). Discriminative models for multi-class object layout. *International Conference on Computer Vision (ICCV)*. Kyoto, Japón.

[6] Galleguillos, C., Rabinovich, A. y Belongie, S. (2008). Object categorization using co-occurrence, location and appearance. *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Anchorage, Alabama, Estados Unidos.

[7] Goodfellow, I., Bengio, Y. y Courville, A. (2016). *Deep Learning*. Cambridge, Massachusetts, Estados Unidos: The MIT Press.

[8] Gould, S., Rodgers, J., Cohen, D., Elidan, G. y Koller, D. (2007). Multi-class segmentation with relative location prior. *International Journal of Computer Vision (IJCV)*, *80*(3), 300-316.

[9] He, X., Zemel, R. S. y Carreira-Perpiñán M. A. (2004). Multiscale conditional random fields for image labeling. *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Washington, DC, Estados Unidos.

[10] Heitz, G., Gould, S., Saxena, A. y Koller, D. (2008). Cascaded classification models: Combining models for holistic scene understanding. *Conference on Neural Information Processing Systems (NIPS)*. Vancouver, British Columbia, Canadá.

[11] Heitz, G. y Koller, D. (2008). Learning spatial context: Using stuff to find things. *Proc. 10th European Conference on Computer Vision*. Marsella, Francia.

[12] Jia, Y., Shelhamer, E., Donahue, J. Karayev, S., Long, J., Girshick, R., Guadarrama, S. y Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*.

[13] Jin, Y. y Geman, S. Context and hierarchy in a probabilistic image model. (2006)

2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2145-2152. Washington, DC, Estados Unidos.

[14] Li, L.-J., Socher, R. y Fei-Fei, L. (2009). Towards total scene understanding: classification, annotation and segmentation in an automatic framework. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Miami, Florida, Estados Unidos.

[15] Murphy, K. P. (2012). *Machine Learning. A Probabilistic Perspective.* Cambridge, Massachusetts, Estados Unidos: The MIT Press.

[16] Murphy, K. P., Torralba, A. y Freeman, W. T. (2003). Using the forest to see the trees: a graphical model relating features, objects and scenes. *Conference on Neural Information Processing Systems (NIPS)*. Vancouver, British Columbia, Canadá.

[17] Niessner J., Helmig R., Jakobs H. y Roberts J. (2005). Interface condition and linearization schemes in the Newton iterations for two-phase flow in heterogeneous porous media. *Advances in Water Resources*. 28: 671-687.

[18] Paillao, E. (2017). *Redes neuronales profundas aplicadas a la detección de atributos de personas en framework Caffe* (tesis de pregrado). Universidad Católica de Temuco, Temuco.

[19] Parikh, D. y Chen, T. Hierarchical semantics of objects (hSOs). (2008). *IEEE International Conference in Computer Vision (ICCV). 2007*. Rio de Janeiro, Brasil.

[20] Peralta, B., Espinace, P. y Soto, A. (2012). Adaptive hierarchical contexts for object recognition with conditional mixture of trees. *British Machine Vision Conference (BMVC) 2012*, *2012*, 1-11.

[21] Porway, J., Wang, K., Yao, B. y Zhu, S. C. (2008). A hierarchical and contextual model for aerial image understanding. *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, *2008*, 1-8.

[22] Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E. y Belongie, S. (2007). Objects in context. *2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Minneapolis, Minnesota, Estados Unidos.

[23] Sudderth, E. B., Torralba, A., Freeman, W. T. y Willsky, A. S. (2005). Learning hierarchical models of scenes, objects, and parts. *International Conference on Computer Vision (ICCV)*. Beijing, China.

[24] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications.* Cambridge, Massachusetts, Estados Unidos: Springer.

[25] Torralba, A. Contextual priming for object detection. (2003). *International Journal of Computer Vision (IJCV)*, *53*(2).

[26] Torralba, A., Murphy, K. P. y Freeman, W. T. (2005). Contextual models for object

detection using boosted random fields. *Conference on Neural Information Processing Systems (NIPS)*. Vancouver, British Columbia, Canadá.

[27] Tu, Z. (2008). Auto-context and its application to high-level vision tasks. *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Anchorage, Alabama, Estados Unidos.

[28] Vergaray, N. (2013). *Simulación numérica de la saturación de petróleo en un medio poroso heterogéneo bajo condiciones preindustriales de operación.* (tesis de pregrado). Universidad Católica de Temuco, Temuco.

7 Códigos MATLAB y Python

7.1 Implementación del modelo probabilístico adaptativo de contexto

Subrutina *optimization.m*:

```
% Subrutina que implementa el modelo probabilístico adaptativo
% de contexto.

  disp('Learning the prior tree model...')

  N = size(b_samples,1);
  M = size(b_samples,2);

  % Cargar características (features) para todas las imágenes.
  %features = ExtractFeatures(names,Dtraining);
  features = load('features_train.txt');
  %features = load('train_sun_fine.txt');
  %features = features./max(max(features));

  % Inicializar Theta_0.
  Theta_t1 = zeros(1,N);
  for i=1:N
      Theta_t1(i) = rand;
  end

  prob_bij = computeBnStats(b_samples,features(1,:),zeros(1,N));
  mi = computeMutualInformationBin(prob_bij);
  adjmat = ChowLiu(mi,names);
  Beta_0 = MLikelihood(adjmat,prob_bij,root);

  % Inicializar parametros.
  epsilon = 1.0e-1;
  Alpha = 1.0e-2;

  err_i = 0;
  err_i1 = 0;
  esucesion = 0;
  eta = 0.0;
  dE = zeros(1,N);
  Theta = zeros(1,N);
```

```
index = randi([1 4367],1,30);

arch5 = fopen('convergencia.txt','w');
arch2 = fopen('salida_Theta.txt','w');
arch10 = fopen('sucesos.txt','w');

% Calcular valor inicial de E.
err_i = error_E(b_samples,names,features,Theta_t1,root);
initial_error = err_i;

fprintf(arch5,'  Funcion de Error.        Error.            Alpha.      ');
fprintf(arch5,'\n');
fprintf(arch5,'\n');
%fprintf(arch5,'%.15f %.15f %.15f',err_i,(norm(Theta_t1-Theta,2))^2,Alpha);
fprintf(arch5,'%.15f %.15f %.15f',err_i,abs(err_i1-err_i),Alpha);
fprintf(arch5,'\n');

%while (((norm(Theta_t1-Theta,2))^2)>1.0e-5)
%while (err_i>1.0e-4)
% Iterar hasta obtener un error adecuado.
while (abs(err_i1-err_i)>1.0e-4)

  Theta = Theta_t1;

  POOL = parpool('local',12);

  parfor j=1:30
    b_i = calc_functions(b_samples,names,features(index(j),:),Theta,root);
    dE = dE + E_diff(b_samples(:,index(j))',b_i,names,b_samples,...
              features(index(j),:),Theta,epsilon,root);
  end

  delete(POOL);
```

```
  % Aplicar descenso de gradiente mediante regla de Armijo.
  % Encontrar un q adecuado.
  for q=0:50
    eta = (1.0/2.0)^q;
    Theta_t1 = Theta - dE.*(Alpha*eta);
    err_i1 = error_E(b_samples,names,features,Theta_t1,root);

    fprintf(arch10,'q = %d',q);
    fprintf(arch10,'\n');

    if (err_i1 <= (1.0-1.0e-4*eta)*err_i)

        fprintf(arch10,'Elemento de sucesion %d',esucesion);
        fprintf(arch10,'\n')
        fprintf(arch10,'%.15f',err_i1);
        fprintf(arch10,'\n')

esucesion = esucesion + 1;
        Alpha = Alpha*eta;
break
    end
  end

  for j=1:N
      fprintf(arch2,'%.15f',Theta_t1(j));
      fprintf(arch2,' ');
  end

  fprintf(arch2,'\n');
  fprintf(arch2,'-------');
  fprintf(arch2,'\n');

  fprintf(arch5,'%.15f',err_i1);
  fprintf(arch5,' ');
  fprintf(arch5,'%.15f',(norm(Theta_t1-Theta,2))^2);
  fprintf(arch5,' ');
  fprintf(arch5,'%.15f',Alpha);
  fprintf(arch5,'\n');

  err_i = err_i1;

end
```

```
  fclose(arch2);
  fclose(arch5);
  fclose(arch10);
```

7.2 Extracción de características empleando HOG

Función *ExtractFeatures.m* que extrae características (*features*) para cada imagen empleando *HOG*.

```
% Función que extrae características (features) para cada imagen.
function feat = ExtractFeatures(names,Dtraining)

  run('/opt/vlfeat-0.9.20/toolbox/vl_setup');
  N = length(names);
  M = length(Dtraining);
  route = strcat('Images/',Dtraining(1).annotation.folder,'/');
  cellSize = 2;
  feat = zeros(M,4*N);

  POOL = parpool('local',12);

  % Repetir para cada imagen.
  parfor k=1:M

    currentfilename = Dtraining(k).annotation.filename;
    currentimage = imread(strcat(route,currentfilename));
    resized = imresize(currentimage,[256 256]);
    hog = vl_hog(im2single(resized),cellSize);

    dim_hog = 128;

    % Reducir el número de features hasta 4*N.
    while(31*(dim_hog/2)^2 >= 4*N)
      dim_hog = dim_hog/2;
      hog = vl_hog(hog,cellSize);
    end
    hog = hog(:);

    for j=1:31*dim_hog^2-4*N
      hog(randi(31*dim_hog^2+1-j)) = [];
    end
```

```
    % Dejar los valores de los features en el intervalo [0,1].
    feat(k,:) = 2.5*hog;

  end

  delete(POOL);

  return

end
```

7.3 Función que evalúa E

Se implementa por medio de *error_E.m*:

```
% Función que evalúa E.
function err = error_E(b_samples,names,features,Theta,root)

  N = size(b_samples,1);
  M = size(b_samples,2);
  err = 0;

  POOL = parpool('local',12);
  % Calcular para cada imagen.
  parfor i=1:M

      b_i = calc_functions(b_samples,names,features(i,:),Theta,root);
      res = b_samples(:,i)'-b_i;

      for j=1:N
          if (b_samples(j,i) == 1)
              res(j) = res(j)*10;
          end
      end

      e = norm(res,2);
      err = err + e;

  end

  delete(POOL);
```

```
  return

end
```

7.4 Cálculo de $\frac{\partial E}{\partial \theta}$

Se implementa por medio de *E_diff.m*:

```
% Función que calcula la derivada parcial de la función E con respecto a Theta.
function F = E_diff(Beta_0,b_i,names,b_samples,features,Theta,epsilon,root)

    N = size(Beta_0,2);
    F = zeros(1,N);

    Fdiff = [];
    %Evaluar la heurística.
    A = calc_A(Beta_0,b_i);

    %Repetir para cada objeto.
    for i=1:N

      epsilon_vec = zeros(1,N);
      epsilon_vec(i) = epsilon;
      fd = finite_diff(b_samples,names,features,Theta,epsilon_vec,epsilon,root);
      Fdiff = [Fdiff; fd];

    end

    F = (diag(A)')*Fdiff;

    return
end
```

7.5 Cálculo de la heurística A

Se ejecuta a través de la función *calc_A*:

```
% Funcion que evalúa la heurística A.
function A = calc_A(Beta_0,b_i)

  N = size(Beta_0,2);
  A = zeros(N,N);

  for i=1:N
      if (Beta_0(i) == 1)
          A(i,i) = 10;
      else
          A(i,i) = 1;
      end
  end

  return

end
```

7.6 Cálculo de f

Es obtenido utilizando la función *calc_functions.m*:

```
% Función que calcula b_i^*.
function b_i = calc_functions(samples,names,features,Theta,root)

  prob_bij = computeBnStats(samples,features,Theta);

  mi = computeMutualInformationBin(prob_bij);
  adjmat = ChowLiu(mi,names);
  b_i = MLikelihood(adjmat,prob_bij,root);

  return
end
```

7.7 Obtención de $\frac{\partial f}{\partial \theta}$

Se genera mediante diferencias finitas por medio de la función *finite_diff.m*:

```
% Función que calcula la diferencia finita de b_i^* con respecto
% a una entrada de Theta.
function F = finite_diff(b_samples,names,features,Theta,epsilon_vec,epsilon,root)

  df1 = calc_functions(b_samples,names,features,Theta+epsilon_vec,root);
  df2 = calc_functions(b_samples,names,features,Theta-epsilon_vec,root);

  F = (df1-df2)./(2*epsilon);

  return

end
```

7.8 Determinación de la Máxima Verosimilitud

Se tiene como resultado al emplear la función *MLikelihood.m* para el cálculo de f.

```
% Función que calcula el Maximum Likelihood para cada imagen, obteniendo b_i^*.
function b_i = MLikelihood(adjmat,prob_bij,root)

   N = size(adjmat,1);
   b_i = zeros(1,N);

   prior_adjmat = sparse(adjmat);
   tree_msg_order = treeMsgOrder(prior_adjmat, root);
   [prior_node_pot, prior_edge_pot] = marToPotBin(prob_bij, tree_msg_order);

   map_est = maxProductBin(adjmat,prior_node_pot,prior_edge_pot,tree_msg_order);
   b_i = map_est'-1;

end
```

7.9 Extracción de características empleando Aprendizaje Profundo

Código programado en Python bajo el nombre *features_sun_finetuning.py* que extrae *features* utilizando Aprendizaje de Máquina:

```
#!/usr/bin/env python
# encoding:utf-8

from PIL import Image
import numpy as np
import sys, os, caffe
from os import path

def getPATH(PATH):
    return path.expanduser(PATH)

def main():
    MEAN_FILE  = getPATH('/home/billy/Descargas/caffe-master/python/caffe\
                         /imagenet/ilsvrc_2012_mean.npy')
    MODEL_FILE = getPATH('/home/billy/Descargas/caffe-master/models\
                         /bvlc_reference_caffenet/deploy.prototxt')
    PRETRAINED = getPATH('/home/billy/Descargas/caffe-master/models/bvlc_\
                         reference_caffenet/bvlc_reference_caffenet.caffemodel')
    FOLDER_TRAIN = getPATH('/home/billy/Descargas/caffe-master/Sun_Database\
                           /BD_Train')
    FOLDER_TRAIN_RESIZED = getPATH('/home/billy/Descargas/caffe-master\
                                   /Sun_Database/BD_Train_resized')
    FOLDER_TEST = getPATH('/home/billy/Descargas/caffe-master/Sun_Database\
                          /BD_Test')
    FOLDER_TEST_RESIZED = getPATH('/home/billy/Descargas/caffe-master\
                                  /Sun_Database/BD_Test_resized')

    LAYER = 'fc7'
    #LAYER = 'fc6wi'
    #LAYER = 'fc7wi'
    INDEX = 4

    net = caffe.Classifier(MODEL_FILE, PRETRAINED, mean=np.load(MEAN_FILE),\
          channel_swap=(2,1,0),raw_scale=255)

    #caffe.set_mode_cpu()
    caffe.set_mode_gpu();
```

```
    caffe.set_device(0);

    d_train = {}
    d_test = {}

    train = open("train_sun_fine.txt", 'w');
    test = open("test_sun_fine.txt", 'w');
    comp_train  = open("comparative_table_train_sun_fine.name", 'w');
    comp_test  = open("comparative_table_test_sun_fine.name", 'w');
    f_train = open('names_images_train_BD_Sun.txt', 'r');
    f_test = open('names_images_test_BD_Sun.txt', 'r');

    flag = 0
    files = os.listdir(FOLDER_TRAIN)
    print len(files)#,files

    for filename in files:
        im_to_resize = Image.open(path.join(FOLDER_TRAIN, filename))
res = im_to_resize.resize((256, 256))
        res.save(path.join(FOLDER_TRAIN_RESIZED, filename), "JPEG")
        image = caffe.io.load_image(path.join(FOLDER_TRAIN_RESIZED, filename))
        net.predict([ image ])
        feat = net.blobs[LAYER].data[INDEX].flatten().tolist()
        comp_train.write('/'.join([FOLDER_TRAIN_RESIZED, filename]) + '\n')
        number = 1
        feat_edit = []
        for fe in feat:
            feat_edit.append(str(fe))
            number += 1

        line = ' '.join(feat_edit) + '\n'
        d_train[filename] = line
        #train.write(line)
        flag3 += 1
print 'train:',flag,len(files)

    for line in f_train:
        filename = line.replace("\n","").replace("\t","")
        train.write(d_train[filename])

    f_train.close()
    train.close()
    comp_train.close()
```

```
    flag = 0
    files = os.listdir(FOLDER_TEST)
    print len(files)#,files

    for filename in files:
        im_to_resize = Image.open(path.join(FOLDER_TEST, filename))
res = im_to_resize.resize((256, 256))
        res.save(path.join(FOLDER_TEST_RESIZED, filename), "JPEG")
        image = caffe.io.load_image(path.join(FOLDER_TEST_RESIZED, filename))
        net.predict([ image ])
        feat = net.blobs[LAYER].data[INDEX].flatten().tolist()
        comp_test.write('/'.join([FOLDER_TEST_RESIZED, filename]) + '\n')
        number = 1
        feat_edit = []
        for fe in feat:
            feat_edit.append(str(fe))
            number += 1

        line = ' '.join(feat_edit) + '\n'
        d_test[filename] = line
        #test.write(line)
        flag3 += 1
print 'test:',flag,len(files)

    for line in f_test:
        filename = line.replace("\n","").replace("\t","")
        test.write(d_test[filename])

    f_test.close()
    test.close()
    comp_test.close()
    print('Fin')

if __name__=="__main__":
    main()
```