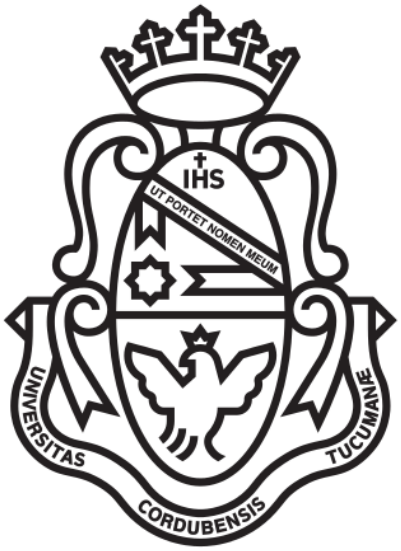


FACULTAD DE MATEMÁTICA, ASTRONOMÍA Y FÍSICA

UNIVERSIDAD NACIONAL DE CÓRDOBA

TRABAJO FINAL

Técnicas *embedding* para clasificación de imágenes en grandes bancos de datos

Maximiliano David Bustos

Director
Jorge Adrián Sánchez

6 de Agosto de 2015

Resumen

En este trabajo se considera el problema de clasificación de imágenes en gran escala mediante *embeddings* lineales. En un modelo embedding, además de generar una representación para las imágenes (entradas) se genera una representación para las clases o conceptos de interés (salidas). De esta forma, al comparar estas representaciones intermedias (imágenes y clases) en un espacio de representación común, es posible abordar de manera unificada problemas como los de clasificación y búsqueda de imágenes por contenido. Los métodos embedding son particularmente atractivos en cuanto permiten generar proyecciones a espacios de dimensionalidad reducida, lo que hace posible el abordaje de problemas en gran escala (millones de imágenes, cientos de miles de conceptos) de manera eficiente.

En particular, se analiza el algoritmo WSABIE propuesto por [Weston et al., 2011b] el cual, a diferencia de los esquemas tradicionales, aborda el problema de aprendizaje mediante la optimización de una función objetivo que tiene en cuenta no solo si una muestra fue bien o mal clasificada, sino cómo se ubicó su etiqueta verdadera respecto de las k mejores predicciones en una lista ordenada de posibles anotaciones.

Resumen en Inglés

In this work we consider the problem of large scale image classification using linear *embeddings*. In an embedding model, a representation of both images (inputs) and classes (outputs) is generated. Then, by comparing these intermediate representations (images and classes) in a common representation space, it is possible to solve problems like classification and image retrieval in a unified manner. Embedding methods are attractive because they allow the projection into spaces of low dimensionality where large scale problems (millions of images and hundreds of thousands of concepts) can be handled efficiently.

In particular, we analyze the WSABIE algorithm proposed by [Weston et al., 2011b] which, unlike traditional methods, approaches the learning problem through the optimization of an objective function that considers not only whether the sample was correctly classified, but also the rank of the true label with respect to the k best predictions in a sorted list of possible annotations.

Palabras clave: visión por computadora, aprendizaje automático, optimización estocástica, vectores de Fisher.

Clasificación:

- I.5 PATTERN RECOGNITION
- I.5.2 Design Methodology
- I.5.4 Applications

Agradecimientos

Quiero dedicar esta sección para agradecer a algunas personas que me apoyaron durante el transcurso de mi carrera. Primero que nada, a mis padres, quienes me brindaron la oportunidad de estudiar en Córdoba y fueron un soporte fundamental para avanzar en mis estudios. También a mis hermanos, con quienes viví día a día esta experiencia de estudiar lejos de casa, y a mi novia por su acompañamiento en todo este proceso.

Por otro lado, quiero agradecer a Jorge, mi director, por su paciencia, dedicación y buena disposición para la realización de este trabajo. A Facu y Mauro, por su gran amistad y por haber compartido toda la carrera juntos. A la comunidad de docentes de Fa.M.A.F, por su compromiso en brindar una educación de primer nivel, y al personal no docente, en especial a María José, Ignacio y Diego, por su ayuda incondicional en todos los trámites, certificados, becas, etc. que tuve durante estos años.

Índice general

Capítulo 1

Introducción

Contexto actual

En la última década, el fácil acceso a cámaras digitales y *smartphones* permitió que las personas produzcan grandes cantidades de datos multimedia, capturados con estos dispositivos. Además, en la actualidad existen numerosas plataformas web, tales como Facebook[1], Google Photos[2], Instagram[3], Flickr[4] y Youtube[5], entre otras, que permiten que estos datos se puedan subir Internet fácilmente en forma de repositorios personales de imágenes y videos, y compartirlos con otras personas.

Estas plataformas tienen cientos de millones de usuarios y por lo tanto almacenan grandes cantidades de datos. Por ejemplo, Instagram reporta 300 millones de usuarios activos mensualmente, 70 millones de fotos subidas cada día y más de 30 billones ya almacenadas.

Un problema que se presenta a los usuarios de estos servicios es cómo acceder a sus fotos y videos de manera fácil y rápida. Por ejemplo, a una persona que quiere mostrarle a alguien la foto que tomó en la cima de una montaña durante sus vacaciones hace tres años, probablemente le resulte tedioso buscar entre las cientos de imágenes de su repositorio personal.

Para mejorar la experiencia de sus usuarios, las empresas proveedoras de estos servicios ofrecen diversas funcionalidades en sus plataformas. Entre ellas, podemos mencionar la agrupación automática de imágenes según su contenido semántico, dentro Google Photos. Además, la posibilidad que este sitio ofrece para realizar búsquedas con consultas tales como "fotos de montañas" o "fotos de mi perro", y obtener imágenes del repositorio personal del usuario. Otra funcionalidad a destacar, es el reconocimiento automático de rostros en fotos de Facebook.

Para que esto sea posible, detrás de cada servicio hay una gran variedad de algoritmos que procesan datos, analizan consultas realizadas por el usuario y elaboran respuestas. Por ejemplo, algoritmos de *clasificación automática de imágenes*, área en la que se enfoca este trabajo, de reconocimiento de patrones, de procesamiento de lenguaje natural, entre otros.

[1] https://www.facebook.com/
[2] https://photos.google.com/
[3] https://www.instagram.com/
[4] https://www.flickr.com/
[5] https://www.youtube.com/

Para el diseño e implementación de estos algoritmos en contextos de gran escala como el que se presentó, la velocidad de cálculo y el consumo de recursos son factores importantes a tener en cuenta, por diversos motivos. Desde el punto de vista del usuario, implica una mejor experiencia durante el uso de los servicios. Por otro lado, para las empresas proveedoras representa mayor eficiencia en sus procesos, reducir costos y obtener una ventaja competitiva respecto de otros actores del mercado.

Por todo esto, el desarrollo de algoritmos que se adapten a estas condiciones es de gran interés en la actualidad.

Objetivos del trabajo

Este trabajo se enfoca en el problema de clasificación automática de imágenes en grandes bancos de datos mediante la utilización de *técnicas embedding*, las cuales buscan proyectar tanto imágenes como clases a un mismo espacio (de baja dimensionalidad) llamado *espacio embedding*. Luego se pueden realizar consultas en este espacio, utilizando una medida de *similitud* entre sus elementos.

Además, las técnicas embedding se pueden aplicar a otros problemas como el de recomendación, recuperación de información y ranking.

En particular, se analiza el algoritmo WSABIE, propuesto por [Weston et al., 2011b], que ha demostrado un buen desempeño en el problema de clasificación de imágenes en contextos de gran escala. El mismo se basa en aprender una representación conjunta sobre las imágenes y las clases de forma tal de optimizar la precisión en lo más alto de una lista *rankeada* de posibles resultados. Es escalable tanto en tiempo de cómputo como en memoria ya que el problema de aprendizaje se aborda mediante técnicas de *optimización estocástica*, y por otro lado, utiliza un espacio embedding, lo que reduce los requerimientos de almacenamiento.

El objetivo de este trabajo es mostrar el buen rendimiento de WSABIE en el problema de clasificación de imágenes en gran escala, como un ejemplo de aplicación de técnicas embedding en estos contextos. Para ello, se lo compara con un algoritmo estándar en la literatura, teniendo en cuenta la utilización de recursos, velocidad de entrenamiento y evaluación, y la calidad de los resultados obtenidos.

Estructura del trabajo

La estructura de este trabajo es la siguiente. Primero, se introducen conceptos de aprendizaje automático. Luego, se explora el problema de clasificación mediante aprendizaje supervisado, presentando modelos lineales y técnicas para aprender sus parámetros. Además, se analizan las debilidades de éstas cuando se trabaja con bancos de datos de gran escala.

En el siguiente capítulo se presenta el enfoque de modelos embedding lineales y el algoritmo WSABIE como caso particular para el problema de clasificación de imágenes en gran escala.

Luego, se dedica un capítulo a los experimentos realizados en cuatro bancos de datos de diferente tamaño (el más grande con 1,4M de imágenes). Aquí se compara el rendimiento de WSABIE con un esquema de referencia de la literatura, respecto de la calidad de los resultados, el tiempo de entrenamiento y evaluación, y la utilización de memoria.

Finalmente, se presentan las conclusiones y posibles direcciones de trabajo futuro.

Capítulo 2

Marco Teórico

2.1. Aprendizaje Automático

2.1.1. Introducción

El *aprendizaje automático* (o *machine learning*, por su denominación en inglés) es un subcampo de las ciencias de la computación y la matemática que se enfoca en la construcción y estudio de algoritmos que pueden *aprender* y realizar *predicciones* sobre datos.

Tom M. Mitchell elaboró una definición más formal de este concepto de aprendizaje: "se dice que un programa de computadora aprende de una experiencia E con respecto a una clase de tarea T y medición de desempeño P, si su desempeño en la tarea T, medido por P, mejora con la experiencia E" [Mitchell, 1997].

A manera de ejemplo, se puede considerar el problema de reconocer dígitos manuscritos [Bishop, 2006]. Se necesita un programa de computadora que tome como entrada una imagen y devuelva como salida el dígito manuscrito en la misma.

Para empezar, se considera un conjunto grande de N imágenes de dígitos manuscritos $\{x_1, ..., x_N\}$, llamado *conjunto de entrenamiento*. Los dígitos representados en cada una de estas N imágenes se conocen de antemano (por ejemplo, inspeccionando cada imagen manualmente). Cada dígito, 0, 1, 2, ... 9, es una *categoría o clase*, y se representa utilizando un *vector objetivo* que se puede codificar de diversas maneras. En este caso, una representación posible sería utilizando un vector de 10 dimensiones, en el que la categoría i se representa con un uno en la posición i del vector, y los demás ceros. Por ejemplo, la categoría del dígito 7 tendría la siguiente representación: 0000000100.

El algoritmo de aprendizaje automático en cuestión debe *aprender* una función paramétrica $y_\theta(x)$ que tome una nueva imagen x como entrada y que genere un vector y como salida, el cual codifique una categoría como se explicó en el párrafo anterior. La forma de esta función $y_\theta(x)$ se determina durante la *fase de entrenamiento o aprendizaje*, mediante la minimización o maximización de una *función objetivo*, utilizando los datos del conjunto de entrenamiento. Luego, durante la *fase de evaluación*, se mide la habilidad del modelo de categorizar correctamente nuevos ejemplos de imágenes nunca antes vistos. Estas imágenes, junto con sus categorías verdaderas, forman lo que se conoce como *conjunto de*

evaluación.

La habilidad de categorizar correctamente nuevas imágenes que difieran de las que se utilizaron durante la fase de entrenamiento, se conoce como *generalización*. En la práctica, la variabilidad de las imágenes de entrada será tal que los datos de entrenamiento constituirán solo una pequeña fracción del conjunto de posibilidades. Por lo tanto, la generalización es uno de los objetivos centrales en los algoritmos de aprendizaje automático.

En aplicaciones prácticas, usualmente se procesan los elementos de entrada con el objetivo de que el problema sea más fácil de resolver. Por ejemplo, en el problema de reconocer dígitos manuscritos, las imágenes se podrían escalar de manera que cada dígito quede contenido en un cuadro de un tamaño prefijado. Hay que tener precaución de no descartar información importante, pues eso afectaría el rendimiento general del sistema. Esta etapa de procesamiento se denomina *extracción de características* y se discutirá en la Sección 4.1 para el caso de las imágenes de nuestros experimentos.

2.1.2. Categorización de problemas de aprendizaje automático

Los problemas de aprendizaje automático suelen categorizarse en tres grandes conjuntos:

- **Aprendizaje supervisado**: son problemas en los que el conjunto de entrenamiento consiste en datos de entrada (vectores de dimensionalidad conocida) junto con sus respectivos valores de salida. En el ejemplo de categorización de dígitos manuscritos presentado anteriormente, estos valores de salida corresponden a los vectores objetivo.

  A su vez se pueden distinguir dos grandes subcategorías de problemas: *regresión* y *clasificación*. En el primer caso, la salida esperada consiste en una o más *variables continuas*. Un ejemplo de problema de regresión sería el de la predicción del valor de un inmueble con determinadas características (ubicación, cantidad de habitaciones, tipo de terreno), contando con un conjunto de entrenamiento con inmuebles, sus características y sus respectivos valores. Por otro lado, cuando se busca asignar a un vector de entrada un valor entre un *conjunto finito de posibilidades o categorías*, estamos ante un problema de clasificación, tema central de este trabajo.

- **Aprendizaje no supervisado**: en este caso, el conjunto de entrenamiento consiste de vectores de entrada pero de los cuales no se conocen sus vectores de salida correspondientes. Entre los problemas más comunes de este tipo, se pueden mencionar: *agrupamiento o clustering* (agrupar vectores de acuerdo a un criterio particular, como pueden ser la distancia o similitud), *reducción de dimensionalidad* (proyectar vectores de alta dimensionalidad a dimensiones más pequeñas para facilitar su manipulación o visualización), entre otros.

- **Aprendizaje por refuerzo**: consiste en el problema de encontrar acciones adecuadas a realizar en una situación dada, con el objetivo de *maximizar una recompensa*. En este caso, el algoritmo de aprendizaje no cuenta con ejemplos de salidas óptimas, sino que debe descubrirlos mediante un

proceso de prueba y error. Por ejemplo, usando técnicas de aprendizaje por refuerzo, una *red neuronal* puede aprender a jugar el juego de *backgammon* con un buen desempeño [Tesauro, 1994].

2.2. Clasificación mediante Aprendizaje Supervisado

2.2.1. Introducción

Muchos problemas en la actualidad se pueden ver como un problema de clasificación. Por ejemplo, el filtro de spam de nuestro sistema de correo electrónico, puede verse como un clasificador que agrupa correos en dos clases, "spam" y "no spam" [Awad and ELseuofi, 2011]. En medicina, se utilizan clasificadores para determinar si un tumor es maligno o benigno [Griffith et al., 2013]. Diversas instituciones financieras utilizan modelos de clasificación mediante aprendizaje supervisado para determinar el *puntaje de crédito* de una persona que solicita un préstamo [Van Gestel et al., 2003].

Además, cabe observar que estos problemas están presentes a diferentes escalas: filtrar millones de correos electrónicos cada hora; diagnosticar miles de pacientes por semana; atender centenas de clientes en un banco por día. En particular, en este trabajo se hace énfasis en los problemas de *clasificación de gran escala*, concepto que se abordará en la Sección 2.2.3.

En esta sección se presentará con más detalles el problema de clasificación mediante aprendizaje supervisado, ya mencionado en la Sección 2.1.2. Primero se establecerá una definición formal de este problema, seguido de una caracterización de los contextos de gran escala. Luego, se presentará el enfoque de modelos lineales, empezando por los clasificadores binarios y la utilización de estos para el caso multiclase. Finalmente, se discutirán técnicas para aprender estos modelos.

2.2.2. Definición formal del problema de clasificación

El objetivo de la clasificación es tomar un vector de entrada x y asignarlo a una de las K clases discretas, identificadas como C_k, donde $k = 1, ..., K$. En el escenario más común, estas clases son disjuntas, de manera que a cada vector de entrada se le asigna una y sólo una de ellas. Por otro lado, cuando se puede asignar más de una clase a un elemento dado, se habla de un problema de *clasificación multi-etiqueta*, pero este caso está fuera del alcance de este trabajo.

Cuando $K = 2$, el problema es de *clasificación binaria*, mientras que $K > 2$ da origen al problema de *clasificación multiclase*.

El espacio de vectores de entrada está dividido en K *regiones de decisión*, R_k con $k = 1, ..., K$, de manera que todos los elementos de R_k son asignados a la clase C_k. Los límites de estas regiones se llaman *fronteras de decisión*.

Representación de clases

En el caso de un problema de clasificación binario, una representación conveniente consiste en tener una variable $t \in \{-1, 1\}$ tal que $t = 1$ representa

la clase C_1 (también llamada *clase positiva*) y $t = -1$ representa la clase C_2 (también llamada *clase negativa*).

Para el caso de múltiples clases, es conveniente utilizar una codificación *1-en-K,* donde t es un vector de longitud K, en el que la clase C_j es representada por $t_j = 1$ y $t_i = 0 \, \forall i \neq j$.

2.2.3. Clasificación en gran escala

Hasta aquí se ha explicado el problema de clasificación en general. A continuación se presentan algunas particularidades del problema de clasificación cuando los conjuntos de entrenamiento y evaluación son grandes, es decir, del orden de los millones de elementos. Esta es una situación común hoy en día, debido al crecimiento acelerado en la cantidad de datos disponibles que requieren análisis. La clasificación de correos electrónicos como spam, la categorización automática de noticias, la clasificación automática de imágenes en servicios como *Google Photos* y *Flickr*, son ejemplos de este problema en un ambiente de gran escala.

La escala de un problema de clasificación mediante aprendizaje supervisado se puede medir en tres dimensiones, tal como describe [Akata, 2014]: el número de clases K, la cantidad de elementos N que conforman el conjunto de entrenamiento, y la dimensión D del descriptor de cada elemento (es decir, la dimensión del vector obtenido luego de la etapa de pre-procesamiento). En las siguientes sub-secciones se detallan algunos desafíos asociados al aprendizaje en gran escala respecto a cada una de estas tres variables.

Número de clases (K)

El incremento de la cantidad de clases no sólo hace que el problema requiera más poder de cómputo, sino también incrementa su complejidad pues se hace más difícil determinar a qué clase pertenece un nuevo elemento nunca antes visto. Por ejemplo, en el problema de clasificación de imágenes, las clases "flor" y "árbol" son simples de distinguir. Pero a medida que K se incrementa, podríamos tener clases como "orquídea", "jazmín" y "rosa", haciendo que el conjunto sea más denso. Si una imagen de de la clase "rosa" es clasificada como "jazmín" u "orquídea", entonces esto será considerado incorrecto.

Dimensión de la representación (D)

Los datos se representan utilizando vectores. La utilización de representaciones de gran dimensionalidad es fundamental para construir clasificadores de gran escala, como analiza [Sánchez et al., 2013] para el caso de clasificación de imágenes. Pero incrementar la dimensionalidad trae un incremento en el uso de memoria y tiempo de cómputo, que debe ser tenido en cuenta a la hora de entrenar un clasificador y poner un sistema en producción.

Cantidad de elementos (N)

Durante la fase de entrenamiento, procesar todos los descriptores de los elementos de un banco de datos de gran escala requiere grandes costos computacionales. Por ejemplo, si tenemos un descriptor de dimensionalidad $\mathcal{O}(10^5)$, donde cada entrada es un número de punto flotante de precisión doble (8 *bytes*),

entonces este descriptor ocupa aproximadamente 0.76MB de memoria. Luego, si tenemos un millón de elementos, no se los puede cargar a todos en memoria simultáneamente, lo que representa un desafío a la hora de aprender un clasificador.

2.2.4. Modelos lineales para clasificación

En esta sección se presenta el enfoque de clasificación mediante *modelos lineales*. Usualmente son elegidos por su simpleza, facilidad de implementación y por su buen desempeño en contextos de gran escala. Cabe aclarar que existe una gran familia de *modelos no lineales* que también se utilizan en el problema de clasificación, pero una descripción de los mismos está fuera del alcance de este trabajo.

En los modelos lineales, las fronteras de decisión son funciones lineales del vector de entrada x y por lo tanto están definidas por hiperplanos de dimensión $D-1$, dentro del espacio de entrada de dimensión D. Un conjunto de datos cuyas clases pueden ser separadas por *fronteras de decisión lineales* se dice que es *linealmente separable*.

Se llama *función discriminante* a aquella que toma como entrada un vector x y le asigna una de las K clases, denotadas como C_k. En particular, las *funciones discriminantes lineales*, son aquellas para las cuales las fronteras de decisión son hiperplanos.

Clasificación binaria mediante discriminantes lineales

La representación más simple de una función discriminante lineal se obtiene tomando una función lineal del vector de entrada:

$$y(x) = w^T x + w_0 \tag{2.1}$$

donde w es llamado el *vector peso*, y w_0 es un *sesgo*.

Un vector de entrada x es asignado a la clase C_1 si $y(x) \geq 0$, y a la clase C_2 en caso contrario. Por lo tanto, la frontera de decisión está definida por $y(x) = 0$, que corresponde a un hiperplano de dimensión $D-1$ dentro del espacio de entrada de dimensión D.

Sean x_A y x_B dos vectores de entrada, ambos en la frontera de decisión, es decir:

$$y(x_A) = y(x_B) = 0. \tag{2.2}$$

Entonces, aplicando la defínción 2.1, se obtiene que:

$$w^T(x_A - x_B) = 0. \tag{2.3}$$

Por lo tanto, el vector w es *ortogonal* a todo vector contenido en la frontera de decisión y además, w determina la orientación de dicha frontera.

Además, si x es un punto en la frontera de decisión, entonces $y(x) = 0$, y se puede ver que la *distancia normal* del origen a la frontera de decisión es:

$$\frac{w^T x}{\|w\|} = -\frac{w_0}{\|w\|}. \tag{2.4}$$

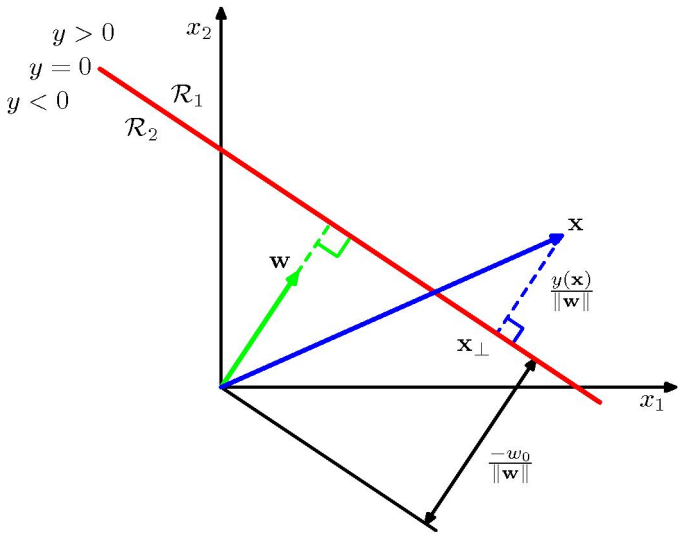


Figura 2.1: Geometría de un discriminante lineal en dos dimensiones. La frontera de decisión, en rojo, es perpendicular a w, y su desplazamiento desde el origen es controlado por w_0.[1]

Por lo tanto, el sesgo w_0 determina la ubicación de la frontera de decisión respecto del origen.

2.2.5. Clasificación multiclase

Como ya se explicó antes, este es el caso en el que hay más de dos clases posibles. Un enfoque a este problema consiste en la utilización de múltiples clasificadores binarios, combinados de distintas maneras, como se detalla a continuación. Además, se analiza la escalabilidad de cada una de estas técnicas.

Uno-contra-todos

La técnica *uno-contra-todos* consiste en entrenar un clasificador binario por cada clase, con los elementos pertenecientes a esa clase como ejemplos positivos y todos los demás como ejemplos negativos. Además, se requiere que cada uno de los clasificadores produzca un valor de confianza de que el elemento pertenece a la clase. Luego, para determinar la clase de un elemento, se evalúan todos los clasificadores y se elige el que reporta mayor nivel de confianza. Es decir, dado un elemento x del cual se quiere predecir su clase $\hat{y}$, se tiene que:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\arg\max}\ y_k(x) \tag{2.5}$$

donde $y_k(.)$ es el clasificador binario correspondiente a la clase C_k.

[1] Esta figura fue obtenida de `http://research.microsoft.com/en-us/um/people/cmbishop/prml/webfigs.htm`. Sus derechos pertenecen a Christopher M. Bishop.

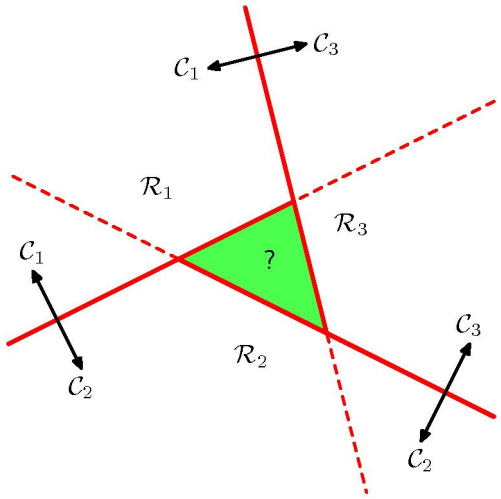


Figura 2.2: Problema de de clasificación en tres clases, en un espacio de dos dimensiones. La región verde es *ambigua* pues un vector perteneciente a ella recibiría un voto para cada clase.[2]

Escalabilidad Esta técnica es paralelizable en la fase de entrenamiento pues cada clasificador es independiente del resto. Es decir, los parámetros elegidos para uno no afectan los parámetros de otro y por lo tanto, se pueden procesar por separado y en simultáneo. Lo mismo aplica a la fase de evaluación.

Para aprender cada clasificador se necesita acceder a todo el conjunto de entrenamiento, lo cual podría ser restrictivo cuando N es grande. Además, utilizando un discriminante lineal como el de la Ecuación 2.1, se necesitan $\mathcal{O}(K)$ productos punto para evaluar un solo vector, factor a tener en cuenta en problemas de gran escala con muchas clases.

Uno-contra-uno

La técnica de *uno-contra-uno* consiste en utilizar $K(K-1)/2$ clasificadores binarios, uno por cada posible par de clases. Se denota por y_{ij} al clasificador que distingue entre las clases C_i y C_j, donde $i \neq j$. Cada uno de estos clasificadores es entrenado utilizando los datos de las dos clases correspondientes. Un nuevo vector se clasifica de acuerdo a una mayoría de *votos* entre las funciones y_{ij}. Es decir, se aplica cada clasificador de manera que si y_{ij} elige la clase C_i, entonces se incrementa en uno la cantidad de votos de C_i, y en caso contrario se incrementa la cantidad de votos de C_j. El resultado es la clase que obtuvo más votos en este proceso.

Esta técnica tiene un problema de *regiones ambiguas*, como muestra la Figura 2.2. Si se tiene un vector en la región verde, entonces al evaluar cada uno de los tres clasificadores, resultará que cada clase cuenta con exactamente un voto.

Escalabilidad Esta técnica también es paralelizable en la fase de entrenamiento por la misma independencia que se explicó para el caso anterior. Por

[2]Esta figura fue obtenida de `http://research.microsoft.com/en-us/um/people/cmbishop/prml/webfigs.htm`. Sus derechos pertenecen a Christopher M. Bishop.

otro lado, para entrenar el clasificador y_{ij} sólo se necesita acceder a los ejemplos de entrenamiento pertenecientes a las clases C_i y C_j, a diferencia de uno-contra-todos, donde se necesita todo el conjunto. La fase de evaluación también se puede ejecutar en paralelo por la independencia mencionada.

Una desventaja es que la cantidad de clasificadores tiene un *crecimiento cuadrático* en el número de clases, haciendo que su aplicación no sea factible si se tiene, por ejemplo, $K = 10^5$. Si se utiliza un discriminante lineal como el de la Ecuación 2.1, se necesitan $\mathcal{O}(K^2)$ productos punto para evaluar un solo vector, lo cual podría ser costoso computacionalmente.

2.2.6. Fase de aprendizaje

Desde el enfoque de aprendizaje supervisado, una vez que está definido el modelo hay que determinar el mecanismo para estimar sus parámetros.

Se consideran dos espacios de objetos, $\mathcal{X}$ e $\mathcal{Y}$, y una función $h_\theta \colon \mathcal{X} \to \mathcal{Y}$ (llamada *hipótesis*), que dado un $x \in \mathcal{X}$ devuelva un $y \in \mathcal{Y}$.

Se cuenta con un conjunto de entrenamiento $\mathcal{S} = \{(x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})\}$, donde $x^{(i)} \in \mathcal{X}$ es un valor de entrada e $y^{(i)} \in \mathcal{Y}$ es la respuesta esperada de $h(x^{(i)})$. Además, se considera que hay una distribución de probabilidad conjunta $P(x, y)$ sobre $\mathcal{X}$ e $\mathcal{Y}$ y que el conjunto de entrenamiento $\mathcal{S}$ consiste de N instancias *independientes e idénticamente distribuídas* de dicha distribución.

Funciones de pérdida

Una *función de pérdida* es una función real, no negativa $L(\hat{y}, y)$, que mide cuán diferente es la predicción $\hat{y}$ obtenida por la hipótesis h_θ, con respecto a la salida esperada y. Existen diversas funciones de pérdida que se utilizan en distintos contextos. A continuación se presentan algunas aplicadas en el problema de clasificación:

- **Función de pérdida 0-1**: $L(\hat{y}, y) = \mathbb{I}(y \neq \hat{y})$, donde $\mathbb{I}(.)$ es la función indicadora.
- **Función de pérdida *hinge***: se utiliza para clasificación de *margen máximo*, es decir, cuando se busca maximizar la distancia entre las fronteras de decisión y los datos de entrenamiento. Esta es la base de un popular método de clasificación llamado *Máquina de Vectores de Soporte* (*Support Vector Machine o SVM*, en la literatura en inglés).

  Se considera $\mathcal{Y} = \{-1, 1\}$, que es una configuración usual en clasificación binaria, e $\hat{y} = w^T x + w_0$. Entonces, $L(\hat{y}, y) = max(0, 1 - \hat{y}y)$. En este caso, el parámetro $\hat{y}$ es el puntaje asignado a x por la función discriminante y no una clase de $\mathcal{Y}$.
- **Función de pérdida logística**: con una configuración análoga a la de pérdida hinge, definimos $L(\hat{y}, y) = log(1 + exp(-\hat{y}y))$.

Minimización del Riesgo Empírico

El riesgo asociado a la hipótesis h_θ se define como la esperanza de la función de pérdida:

$$R(h_\theta) = \mathbb{E}_{(x,y)\sim P}[L(h_\theta(x), y)] = \int L(h_\theta(x), y) dP(x, y). \tag{2.6}$$

El objetivo en la etapa de aprendizaje es encontrar una hipótesis h^*_θ entre una familia de funciones $\mathcal{H}$ dada por θ, para la cuál el riesgo $R(h^*_\theta)$ sea mínimo:

$$h^*_\theta = \underset{h_\theta \in \mathcal{H}}{\arg\min}\, R(h_\theta). \tag{2.7}$$

Por lo general, el riesgo $R(h_\theta)$ no puede computarse ya que la distribución $P(x, y)$ es desconocida. Sin embargo, se puede calcular una aproximación, llamada *riesgo empírico*, calculando la media de la función de pérdida en el conjunto de entrenamiento:

$$R_{emp}(h_\theta) = \frac{1}{N} \sum_{i=1}^{N} L(h_\theta(x^{(i)}), y^{(i)}). \tag{2.8}$$

El *principio de minimización del riesgo empírico* establece que el algoritmo de aprendizaje debería elegir una hipótesis $\hat{h}_\theta$ que minimice el riesgo empírico:

$$\hat{h}_\theta = \underset{h_\theta \in \mathcal{H}}{\arg\min}\, R_{emp}(h_\theta). \tag{2.9}$$

El problema de *sobre ajuste*

Como ya se explicó en la Sección 2.1.1, uno de los principales objetivos de los algoritmos de aprendizaje automático es su capacidad de *generalizar* a ejemplos nunca antes vistos. Sin embargo, si la fase de aprendizaje se realiza durante demasiado tiempo o si los ejemplos del conjunto de entrenamiento son raros, el modelo aprendido podría ajustarse específicamente a ciertas características aleatorias de estos datos, que en realidad no contribuyen a la generalización. Esto se conoce como *sobre ajuste* (*overfitting* en la literatura en inglés), y es un problema importante y ampliamente discutido en el campo de aprendizaje automático. En el proceso de sobre ajuste, el desempeño del algoritmo en el conjunto de entrenamiento sigue mejorando, pero en el conjunto de evaluación empeora.

En la Figura 2.3, se puede ver un ejemplo de este problema en el caso de clasificación binaria. A medida que la fase de aprendizaje avanza, el clasificador se ajusta cada vez más al ruido aleatorio del conjunto de entrenamiento.

Una técnica para evitar el sobre ajuste de modelos es utilizar *regularización*. Esencialmente, consiste en penalizar los parámetros del modelo para evitar su crecimiento desmedido.

Aplicando un regularizador al riesgo empírico de la Ecuación 2.8 se obtiene:

$$R'(h_\theta) = R_{emp}(h_\theta) + \lambda C(h_\theta) \tag{2.10}$$

donde $C(h_\theta)$ mide la complejidad de la función $h_\theta(x)$ y λ controla la importancia que se le da a la penalización.

Sobre ajuste en el problema de clasificación

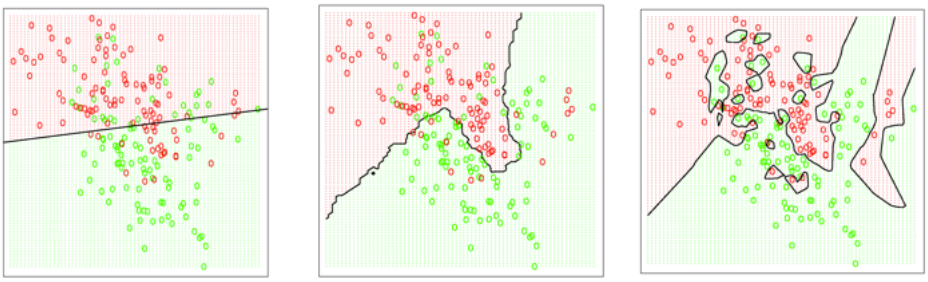


Figura 2.3: A medida que la fase de aprendizaje avanza (imágenes de derecha a izquierda), el clasificador se ajusta cada vez más al ruido aleatorio del conjunto de entrenamiento.

Descenso de Gradiente

El método de *descenso de gradiente* permite encontrar *mínimos locales* de una función. Está basado en la observación de que si una función multivariable $F(w)$ está definida y es diferenciable en un vecino de un punto a, entonces $F(w)$ decrece más rápido si se avanza desde a en la dirección del gradiente negativo de F en a, es decir, $-\nabla F(a)$. Si $b = a - \gamma \nabla F(a)$ para un γ suficientemente pequeño, entonces $F(a) \geq F(b)$

El método se inicializa con una estimación w_0 de un mínimo local de $F(w)$, y construye una secuencia $w_0, w_1, w_2, ...$ tal que:

$$w_{n+1} = w_n - \gamma_n \nabla F(w_n),\ n \geq 0 \tag{2.11}$$

donde γ_n representa el tamaño de cada paso y en la literatura de aprendizaje automático se lo denomina *tasa de aprendizaje*.

Entonces, se tiene que $F(w_0) \geq F(w_1) \geq F(w_2) \geq ....$ Existen extensos estudios sobre las condiciones que debe reunir $F(w)$ para asegurar la *convergencia* del método, pero están fuera del alcance de este trabajo.

Una particularidad es que cuando $F(w)$ es convexa, todo mínimo local es *mínimo global*, por lo tanto en este caso el método de descenso de gradiente puede converger a una *solución global*.

Descenso de Gradiente Estocástico

En las técnicas de aprendizaje automático se considera el problema de minimizar una función objetivo que tiene forma de suma:

$$Q(w) = \sum_{i=1}^{N} Q_i(w) \tag{2.12}$$

donde queremos estimar el parámetro w^* que minimiza $Q(w)$. Por ejemplo, esto está presente en la minimización del riesgo empírico 2.8.

Utilizando la técnica de descenso de gradiente, el método realizaría las siguientes iteraciones:

$$w_{n+1} = w_n - \gamma_n \nabla Q(w_n) = w_n - \gamma_n \sum_{i=1}^{N} \nabla Q_i(w_n). \tag{2.13}$$

Cuando se utiliza un conjunto de entrenamiento de gran escala, la evaluación de la suma de los gradientes se vuelve costosa computacionalmente. El método de *descenso de gradiente estocástico* propone economizar el costo de cada iteración utilizando sólo un subconjunto de las funciones sumandos $\nabla Q_i(w)$ en cada paso. Es decir, el verdadero gradiente $\nabla Q(w)$ se aproxima por el gradiente en un sólo elemento:

$$w_{n+1} = w_n - \gamma_n \nabla Q_i(w_n). \tag{2.14}$$

A medida que el algoritmo avanza a través del conjunto de entrenamiento, realiza esta actualización de w para cada elemento.

El *pseudo-código* del método de descenso de gradiente estocástico se puede ver en el Algoritmo 1.

Algorithm 1 Descenso de Gradiente Estocástico

Elegir una taza de aprendizaje γ.
Elegir un vector w inicial.
Ordenar aleatoriamente los elementos del conjunto de entrenamiento.
repeat
  for $i = 1, ..., N$ **do**
    $w = w - \gamma \nabla Q_i(w)$
  end for
until Se encuentra un mínimo aproximado.

Capítulo 3

Modelos *embedding* lineales

3.1. Introducción

En el capítulo anterior, se presentó un abordaje del problema de clasificación utilizando aprendizaje supervisado sobre clasificadores lineales. En este capítulo, se describe un enfoque diferente, el de *modelos embedding*, que como se mostrará en los experimentos, tiene mejor rendimiento en contextos de gran escala.

En general, las técnicas embedding consisten en aprender una función de mapeo de los datos en un nuevo espacio, posiblemente de menor dimensión, llamado *espacio embedding*. Luego, midiendo la similitud de los vectores en este nuevo espacio, se pueden resolver una gran variedad de problemas, como el de clasificación, recuperación de información, ranking, recomendación, entre otros.

En este trabajo se estudian *modelos embedding lineales*, es decir, aquellos cuya función de mapeo es lineal. Estos modelos han tenido gran éxito en los problemas ya mencionados. Por ejemplo, la técnica de descomposición en valores singulares (SVD) para recomendación [Billsus and Pazzani, 1998, Koren et al., 2009], indexación semántica latente (LSI) para recuperación de información [Deerwester et al., 1990] y WSABIE, el algoritmo principal estudiado en este trabajo, para clasificación de imágenes en gran escala [Weston et al., 2011b].

Un punto a destacar es que utilizando baja dimensionalidad en el espacio embedding, estos modelos son escalables a grandes conjuntos de datos. Mostrar esto es uno de los principales objetivos de este trabajo.

3.2. Definición formal

Los modelos embedding lineales se pueden definir de la siguiente forma:

$$f_{EMB}(x, y) = x^T V^T W y = \sum_{ij} x_i V_i^T W_j y_j \tag{3.1}$$

donde $x \in \mathbb{R}^D$ e $y \in \mathbb{R}^K$ son los vectores que representan los datos de entrada y salida respectivamente, V_i y W_j son las columnas *i-ésima* y *j-ésima* de V y W respectivamente.

Los vectores de entrada pueden representar imágenes, videos, música, documentos de texto, etc. Por otro lado, dependiendo del problema en consideración, los valores de salida pueden ser la representación de una etiqueta o clase (en el

caso de anotación o clasificación), un documento (en el caso de recuperación de información), o un ítem (en el caso de recomendación).

Como ya se explicó en la Sección 2.2.2, para el problema de clasificación la salida generalmente se representa con un vector que tiene un único elemento igual a uno, que indica una clase particular, y el resto igual a cero.

Un punto a destacar es que, independientemente del tipo de objetos de entrada y salida que tenga el problema, estos son representados por elementos en el mismo espacio embedding. Por ejemplo, en el caso de clasificación de imágenes, tanto las imágenes como las anotaciones se representan con nuevos vectores dentro este espacio, como se explicará en 3.4.1.

La función f_{EMB} de la Ecuación 3.1 asigna un puntaje a un par de vectores de entrada y salida dados. Para ello, primero realiza el mapeo de ambos en el espacio embedding de dimensión d, utilizando las matrices V (de dimensión $d \times D$) y W (de dimensión $d \times K$). Luego, utiliza el *producto punto o producto escalar* para obtener una medida de la similitud de las representaciones. Cabe destacar que se pueden utilizar otras métricas tales como la *distancia Euclidea* o la *similitud coseno*, cambiando f_{EMB} según corresponda.

3.3. Clasificación mediante *aprender a rankear*

En esta sección se introduce el enfoque al problema de clasificación como un *problema de rankeo* de elementos, utilizando los modelos embedding lineales antes presentados.

Aprender a rankear se refiere al conjunto de técnicas de aprendizaje automático para entrenar modelos en un problema de ranking. Estas técnicas son útiles en muchas aplicaciones de recuperación de información, procesamiento de lenguaje natural y minería de datos.

Combinando los modelos embedding lineales, presentados en la sección anterior, con técnicas de aprender a rankear, el problema de clasificación se puede resolver de la siguiente manera: para realizar una nueva predicción sobre un vector x se computa $f_{EMB}(x, y)$ para cada etiqueta o clase y, se ordenan los puntajes obtenidos de mayor a menor y se elige el primero. Formalmente, la etiqueta predecida $\hat{y}$, se define como:

$$\hat{y} = \arg\max_{y} f_{EMB}(x, y). \tag{3.2}$$

Es posible extender esta técnica al problema de clasificación multi-etiqueta, es decir, cuando a un elemento se le puede asignar más de una clase (Sección 2.2.2). Para lograrlo, si se quieren asignar k etiquetas se consideran los primeros k puntajes mayores.

En otras palabras, estamos ante un problema de ranking, en el que el objetivo es aprender a rankear la lista de posibles anotaciones, de manera que las que se encuentren en las primeras posiciones sean las correctas.

3.4. *WSABIE*: clasificación de imágenes en gran escala

En esta sección se presenta el algoritmo *WSABIE* (anotación para escala web mediante embeddings de imágenes, por sus siglas en inglés) propuesto por [Weston et al., 2011b]. El mismo utiliza modelos embedding lineales en combinación con el enfoque de aprender a rankear para el problema de clasificación de imágenes en gran escala, que es tema principal de este trabajo.

Esencialmente, WSABIE aprende una representación conjunta de imágenes y anotaciones en un espacio embedding de baja dimensionalidad y al mismo tiempo optimiza la precisión en los primeros elementos de una lista rankeada de anotaciones. Es decir que, para una imagen dada, se calcula un ranking de todas las anotaciones, buscando que aquellas que ocupan las primeras posiciones sean las que mejor describan su contenido semántico.

Por otro lado, se busca que tanto los tiempos de entrenamiento y evaluación, así como el uso de memoria, sean escalables a conjuntos con millones de datos. Como se mostrará en los experimentos realizados (Capítulo 4), la baja dimensionalidad del espacio embedding permite que la obtención de una lista de anotaciones rankeada para una imagen en particular sea muy rápida. Además, esta baja dimensionalidad implica menor uso de memoria.

3.4.1. Modelo de representación conjunta de imágenes y anotaciones

El algoritmo aprende una representación conjunta de imágenes y anotaciones en un mismo espacio embedding de dimensión d. Para ello, se necesitan dos funciones de mapeo, una para imágenes y otra para anotaciones.

Consideramos que una imagen está representada por un vector $x \in \mathbb{R}^D$. Definimos la siguiente función lineal, que mapea imágenes en el espacio de representación conjunta:

$$\begin{aligned} \Phi(x)\colon \mathbb{R}^D &\to \mathbb{R}^d \\ \Phi(x) &= Vx \end{aligned} \tag{3.3}$$

donde V es una matriz de dimensión $d \times D$.

Análogamente, consideramos que las anotaciones están representadas por $i \in Y = \{1, ..., K\}$, índices en un diccionario de anotaciones posibles. Utilizamos esta codificación en lugar de la vectorial explicada con anterioridad, pues así se simplifica la notación. La función de mapeo en este caso se define como:

$$\begin{aligned} \Psi(i)\colon \{1, ..., K\} &\to \mathbb{R}^d \\ \Psi(i) &= W_i \end{aligned} \tag{3.4}$$

donde W_i representa la *i-ésima* columna de la matriz W, de dimensión $d \times K$.

La Figura 3.1 ilustra el rol de estas dos funciones.

Utilizando las funciones de mapeo, se puede definir el siguiente modelo, que tiene un estructura acorde a la definición de modelos embedding lineales, presentada en la Sección 3.2.

$$f(x, i) = \Phi(x)^T \Psi(i) = x^T V^T W_i. \tag{3.5}$$

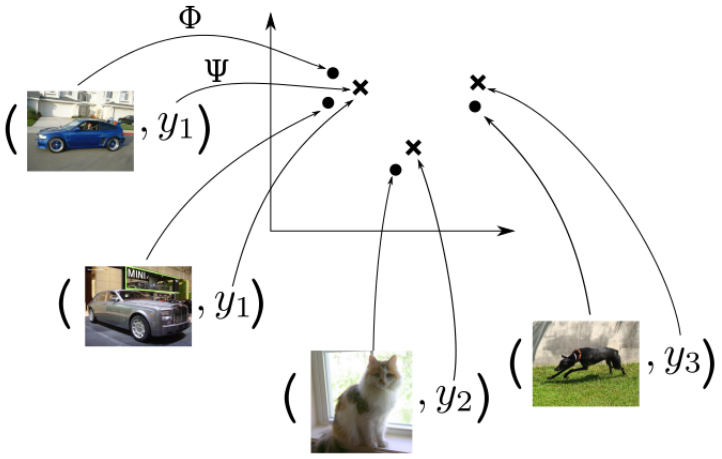


Figura 3.1: La figura muestra un espacio embedding de dos dimensiones ($d = 2$), junto con las proyecciones de cuatro imágenes y tres anotaciones o clases, a través de las funciones Φ y Ψ. Como se mencionó, la idea es que las imágenes de la misma clase tengan una representación *similar* a la representación de su clase correspondiente.

El ranking de anotaciones se calcula de acuerdo a la magnitud decreciente de $f(x, i)$, de la misma manera que se explica en la Sección 3.3.

Además, se utilizan los siguientes regularizadores:

$$\|V_i\| \leq C,\ i = 1, ..., D. \tag{3.6}$$

$$\|W_i\| \leq C,\ i = 1, ..., K. \tag{3.7}$$

3.4.2. Función de pérdida WARP

Se considera la siguiente función de error de a pares:

$$err(x, y, \bar{y}) = L(rank_y(x))max(0, l + f(x, \bar{y}) - f(x, y)) \tag{3.8}$$

donde $rank_y(x)$ representa el ranking de la anotación correcta y de la imágen x, penalizado por un margen l:

$$rank_y(x) = \sum_{\bar{y} \neq y} \mathbb{I}(f(x, y) \leq l + f(x, \bar{y})) \tag{3.9}$$

donde $\mathbb{I}$ es la función indicadora.

Por otro lado, $L(.)$ transforma este ranking en una pérdida:

$$L(k) = \sum_{j=1}^{k} \alpha_j, con\, \alpha_1 \geq \alpha_2 \geq ... \geq 0. \tag{3.10}$$

Diferentes elecciones de los α_j son posibles. Por ejemplo:

- $\alpha_j = \dfrac{1}{(K-1)}$ optimiza el ranking medio.

- $\alpha_1 = 1, \alpha_j = 0$, optimiza la precisión en 1, es decir, la proporción de anotaciones correctas que están en la primera posición del ranking.
- Valores mayores para los k primeros α_j, optimiza la precisión en k.

Consideramos un conjunto de entrenamiento $\mathcal{S} = \{(x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})\}$. En [Weston et al., 2011b] se define la siguiente función objetivo, conocida como *función de pérdida WARP* (weighted approximate-rank pairwise, por su denominación en inglés):

$$\sum_{(x,y)\in S} \sum_{\bar{y}\neq y} err(x, y, \bar{y}) = \sum_{(x,y)\in S} \sum_{\bar{y}\neq y} L(rank_y(x)) max(0, l + f(x, \bar{y}) - f(x, y)). \tag{3.11}$$

3.4.3. Entrenamiento utilizando descenso de gradiente estocástico

La forma de la función de pérdida WARP sugiere que se puede utilizar la técnica de descenso de gradiente estocástico, presentada en la Sección 2.2.6, para entrenar este modelo.

1. Elegir un par (x, y) del conjunto de entrenamiento.
2. Para este par (x, y), encontrar un $\bar{y}$ tal que $l + f(x, \bar{y}) > f(x, y)$.
3. En cada iteración, realizar la siguiente actualización de parámetros β (notación que representa a las matrices V y W), los cuales definen una familia de posibles funciones $f \in F$ (pues f está definida por V y W):

$$\beta_{t+1} = \beta_t - \gamma_t \frac{\partial err(x, y, \bar{y})}{\partial \beta_t} \tag{3.12}$$

   donde γ_t es la tasa de aprendizaje.

Sin embargo, hay dos problemas que hacen que este procedimiento sea ineficiente:

1. En el paso 2, necesitamos computar los valores de $f(x, i)$ con $i = 1, ..., K$ para saber qué etiquetas $\bar{y}$ cumplen la condición $l + f(x, \bar{y}) > f(x, y)$. Esto es costoso cuando la cantidad de clases K es grande.
2. $rank_y(x)$ es desconocido si no se computa $f(x, i)$ para cada i, lo cual nuevamente es costoso.

[Weston et al., 2011b] propone una solución para ambos problemas. En el paso 2, muestrear etiquetas i de manera uniforme y con reposición hasta encontrar una que cumpla la condición $l + f(x, \bar{y}) > f(x, y)$. Si existen $k = rank_y(x)$ de estas etiquetas, la variable aleatoria N_k que cuenta el número de ensayos realizados hasta encontrar una de ellas, sigue una *distribución geométrica* de parámetro $p = \frac{k}{K-1}$ (probabilidad de éxito). Entonces, $E[N_k] = \frac{1}{p} = \frac{K-1}{k}$.

Por lo tanto, $k = \frac{K-1}{E[N_k]}$. El valor de $rank_y(x)$ de la Ecuación 3.9 puede ser aproximado por:

$$rank_y(x) \approx \left\lfloor \frac{K-1}{n} \right\rfloor \tag{3.13}$$

donde $\lfloor . \rfloor$ es la función piso y n el número de ensayos realizados en la etapa de muestreo es un estimador puntual de $E[N_k]$.

Finalmente, aplicando este método, el gradiente queda así:

$$\frac{\partial L\left(\left\lfloor \frac{K-1}{n} \right\rfloor\right) max(0, l + f(x, \bar{y}) - f(x, y))}{\partial \beta_t}. \tag{3.14}$$

Reemplazando β por W y calculando las derivadas parciales, la actualización de parámetros en cada iteración de descenso de gradiente estocástico resulta:

$$W_y \leftarrow W_y + \gamma L\left(\left\lfloor \frac{K-1}{n} \right\rfloor\right) Vx$$

$$W_{\bar{y}} \leftarrow W_{\bar{y}} - \gamma L\left(\left\lfloor \frac{K-1}{n} \right\rfloor\right) Vx.$$

Análogamente, reemplazando β por V, se obtiene:

$$V_{ij} \leftarrow V_{ij} - \gamma L\left(\left\lfloor \frac{K-1}{n} \right\rfloor\right) x_j(W_{i\bar{y}} - W_{iy}).$$

En este trabajo se utiliza una tasa de aprendizaje γ constante e independiente del número de iteración, aunque podrían analizarse otras técnicas más complejas como AdaGrad [Duchi et al., 2010].

En el Algoritmo 2 se puede ver pseudo-código de WSABIE, que consiste del espacio embedding para anotaciones e imágenes explicado en la Sección 3.4.1 y entrenado con la función de pérdida WARP.

Parámetros del algoritmo

A continuación se resume la lista de parámetros ajustables que tiene el algoritmo WSABIE:

- Dimensión del espacio embedding (d) en 3.3 y 3.4.
- Tasa de aprendizaje para descenso de gradiente estocástico (γ) en 3.12.
- Margen de separación de clases (l) en 3.8.
- Constante de regularización (C) en 3.6 y 3.7.
- Cantidad de iteraciones del descenso de gradiente estocástico: se puede prefijar o bien utilizar un conjunto de validación para terminar el ciclo cuando el error en el mismo no mejora.

Algorithm 2 WSABIE

Require: conjunto de entrenamiento $(x^{(i)}, y^{(i)}), y^{(i)} \in \{1, ..., K\}$.
  repeat
    Elegir aleatoriamente un elemento (x, y) del conjunto de entrenamiento.
    Calcular: $f(x, y) = \Phi(x)^T \Psi(y)$
    Inicializar: $n = 0$
    repeat
      Elegir una anotación aleatoria $\bar{y} \in \{1, ..., K\} \setminus y$
      Calcular: $f(x, \bar{y}) = \Phi(x)^T \Psi(\bar{y})$
      $n = n + 1$.
    until $f(x, \bar{y}) + l > f(x, y) \vee n \geq K - 1$
    if $f(x, \bar{y}) + l > f(x, y)$ **then**
      Realizar una iteración de descenso de gradiente para minimizar

$$L\left(\left\lfloor \frac{K-1}{n} \right\rfloor\right) max(0, l + f(x, \bar{y}) - f(x, y))$$

      Proyectar V y W para que se cumplan las condiciones 3.6 y 3.7.
    end if
  until Error de validación no mejora.

3.4.4. Análisis de consumo de memoria

- **Etapa de entrenamiento:** el algoritmo necesita mantener en memoria las matrices V y W, lo cual implica un consumo de $\mathcal{O}(dD + dK)$. Como el entrenamiento es *online*, los vectores se pueden ir leyendo a medida que se necesitan. Es decir, WSABIE no requiere cargar todo el conjunto de entrenamiento en memoria, lo cual es una propiedad deseable para problemas de gran escala.

- **Etapa de evaluación:** sólo se necesitan las dos matrices V y W, por lo tanto el consumo de memoria es $\mathcal{O}(dD + dK)$.

3.4.5. Otras aplicaciones de la función de pérdida WARP y el algoritmo WSABIE

Además del problema de clasificación de imágenes, la función de pérdida WARP y el algoritmo WSABIE fueron aplicados exitosamente en otros contextos de gran escala:

- Anotación y recuperación de música [Weston et al., 2011a]: se proyectan clips de audio, artistas y etiquetas en un mismo espacio embedding, y se consideran las siguientes tareas: predicción de artistas (a partir de un clip de audio, determinar su artista intérprete), predicción de canciones (dada una canción, determinar posibles artistas intérpretes), obtener artistas similares (dado un artista, construir una lista de artistas similares), obtener canciones similares (dada una canción, devolver una lista de canciones similares) y predicción de etiquetas (dada una canción, devolver una lista de etiquetas que la describan).

- Sistema de recomendación para *Google Play Music*[1] y *YouTube* [Weston et al., 2013b]: se utiliza una versión generalizada de la función de pérdida WARP para resolver tres problemas de gran escala y utilizando datos de sistemas en producción: recomendación de artistas y pistas en Google Play Music, y recomendación de videos en YouTube.

[1] http://music.google.com/

Capítulo 4

Experimentos

En este capítulo se describen los experimentos realizados en este trabajo. Primero se explica el modelo de representación de imágenes, se definen las métricas utilizadas y finalmente se presentan los resultados obtenidos en los distintos bancos de datos, organizados en tres conjuntos: experimentos de pequeña escala (Pascal VOC2007 [Everingham et al., 2009] y CUB200-2011 [Welinder et al., 2010, Wah et al., 2011]), experimentos de mediana escala (SUN397 [Xiao et al., 2010, 2014]) y experimentos de gran escala (ILSVRC'10 [Russakovsky et al., 2015]). En el Cuadro 4.1 se puede ver un resumen de las dimensiones de dichos conjuntos.

4.1. Representación de imágenes

Como se mencionó en la Sección 2.1.1, las imágenes deben ser pre-procesadas antes de ser utilizadas por el algoritmo de aprendizaje automático.

En problemas de clasificación y búsqueda de imágenes por contenido la representación de la información visual es un problema de fundamental importancia. En la literatura, el método más utilizado consiste en extraer de la imagen un conjunto de características locales (descriptores), proyectarlas a un espacio de gran dimensionalidad y computar, a partir de estas proyecciones, una representación vectorial que permita caracterizar la información visual a mayores niveles de abstracción; esto es, permitir diferenciar entre conceptos dados a nivel semántico, capturar la noción de similitud en el contenido de dos imágenes o regiones

Banco de datos	N	K	**Originales**	**FV 2,5K**	**FV 40K**
Pascal VOC2007	9.963	20	875MB	197,6MB	2,9GB
CUB200-2011	11.788	200	1,2GB	232,4MB	3,25GB
SUN397	108.754	397	39GB	2,12GB	32,6GB
ILSVRC'10	∼1,4M	1.000	155GB	28GB	∼420GB

Cuadro 4.1: Resumen de los bancos de datos utilizados en los experimentos junto con sus dimensiones (N y K), y los costos de almacenamiento asociados.

arbitrarias, etc. Dentro de esta clase de modelos, quizá el más emblemático ha sido el de *Bolsa de Palabras Visuales* (BoVW, por sus siglas en inglés) [Csurka et al., 2004, Sivic and Zisserman, 2003]. En el modelo BoVW, el conjunto de descriptores locales se codifica con ayuda de una representación auxiliar conocida como vocabulario visual, la cual se obtiene empleando técnicas de clustering sobre un conjunto grande de descriptores en un conjunto de entrenamiento. La representación del contenido de una imagen (o una región de la misma) se obtiene como el promedio de las representaciones locales resultado de este proceso de codificación.

Modelos más recientes, tales como *VLAD* [Jegou et al., 2010], *Super Vector* [Zhou et al., 2010] o *vectores de Fisher (FV)* [Perronnin and Dance, 2007], generalizan el modelo BoVW mediante la inclusión de estadísticas de orden superior o el uso de vocabularios probabilísticos. Entre estos, el modelo de vectores de Fisher ha demostrado ser el de mejor rendimiento en problemas de clasificación [Chatfield et al., 2011, Huang et al., 2014], superando ampliamente los niveles de desempeño alcanzados empleando modelos BoVW.

En este trabajo se emplearon vectores de Fisher, los cuales se presentan a continuación, junto con la configuración particular utilizada en los experimentos.

Vectores de Fisher Gaussianos

Sea $X = \{x_t, t = 1, ..., T\}$ el conjunto de los descriptores locales D_ℓ- *dimensionales* extraídos de una imagen dada. Además, sea $u_\lambda \colon \mathbb{R}^{D_\ell} \to \mathbb{R}_+$ una función de *distribución de probabilidad* con parámetros λ modelando el proceso de generación de *descriptores de bajo nivel* de cualquier imagen. En este caso, u_λ se define como una mezcla de M Gaussianas con covarianzas diagonales: $u_\lambda(x) = \sum_{i=1}^{M} w_i u_i(x)$, $\lambda = \{w_i, \mu_i, \sigma_i, i = 1, ..., M\}$. w_i, μ_i y σ_i^2 denotan, respectivamente, el peso de la mezcla, y los vectores media y varianza correspondientes al *i-ésimo* componente de la misma.

El *vector de Fisher* se define como $\mathcal{G}_\lambda^X = L_\lambda G_\lambda^X$, donde G_λ^X corresponde al gradiente (promediado) del logaritmo de la verosimilitud de X, es decir, $\frac{1}{T} \sum_{t=1}^{T} \nabla_\lambda \log u_\lambda(x_t)$ y L_λ es un normalizador diagonal. La representación de la imagen es la concatenación de derivadas parciales normalizadas, resultando en un vector de dimensión $D = 2MD_\ell$. Siguiendo a Perronnin et al. [2010], se aplica la transformación $f(z) = sign(z)\sqrt{|z|}$ en cada dimensión y se normaliza el vector resultante con la norma L_2, ya que se ha mostrado que esto mejora la exactitud de la clasificación.

Para los experimentos, los FV se calcularon en base a un vocabulario de 16 o 256 Gaussianas. En el primer caso, cada vector resulta de una dimensión $D = 2,5K$ y ocupa aproximadamente 20KB para su almacenamiento. Con 256 Gaussianas se obtienen vectores de dimensión $D = 40K$ y un costo de almacenamiento de 300KB. Se utilizará la notación FV 2,5K y FV 40K para hacer referencia a cada una de estas configuraciones. En ambos, la representación de números es con *aritmética flotante de precisión doble* (8 bytes). En el Cuadro 4.1 se resume el costo total de almacenamiento para cada banco de datos.

Características de bajo nivel

Cada imagen se escaló para que tuviera un tamaño máximo de 10^5 píxeles y se la transformó a escala de grises (pues no se utilizaron descriptores de colores).

Se computaron descriptores SIFT [Lowe, 2004] de 128 dimensiones, extraídos de *parches* de las imágenes distribuídos uniformemente en ellas (parches de 6×6 píxeles, tomados cada 4 píxeles, osea, con superposición). Los descriptores resultantes fueron reducidos a 80 dimensiones utilizando *Descomposición en Componentes Principales* (PCA). Para dar cuenta de las variaciones en escala, se extrajeron parches en 5 resoluciones diferentes usando un factor de escala de $1,414$ entre niveles.

Modelo Generativo

Se entrenó un *Modelo de Mezcla de Gaussianas* (GMM) bajo un criterio de *Máxima Verosimilitud* (ML) usando el algoritmo de *Maximización de la Esperanza* (EM), tal como se explica en [Sánchez et al., 2013]. Se utilizaron un millón de muestras aleatorias en cada banco de datos. Las iteraciones de EM se inicializaron ejecutando *k-means* y utilizando las estadísticas de la asignación de clústers (cantidad relativa, vectores de media y varianza) como estimaciones iniciales.

4.2. Métricas

En esta sección se presentan las métricas utilizadas en los experimentos: *precisión*, *exhaustividad*, *exactitud*, y *precisión promedio interpolada (mAP)*. La elección de las mismas varía según la especificación provista por cada banco de datos, por ejemplo, en Pascal VOC 2007 su protocolo indica que se debe utilizar *mAP*. Es importante respetar esto pues así se pueden comparar los resultados obtenidos con otros publicados en la literatura.

4.2.1. Precisión, exhaustividad y exactitud

Si se ejecuta un clasificador binario como el de la Sección 2.2.4 sobre un conjunto de evaluación, existen cuatro resultados posibles para cada elemento clasificado:

- **Verdadero Positivo**: el elemento se clasificó como *positivo*, y su valor real es *positivo*.
- **Verdadero Negativo**: el elemento se clasificó como *negativo*, y su valor real es *negativo*.
- **Falso Positivo**: el elemento se clasificó como *positivo*, pero su valor real es *negativo*.
- **Falso Negativo**: el elemento se clasificó como *negativo*, pero su valor real es *positivo*.

Estas cuatro posibilidades se pueden visualizar en el Cuadro 4.2.

Considerando los resultados del clasificador ejecutado sobre todo el conjunto de evaluación, se definen:

- **VP**: cantidad de elementos que resultaron *Verdaderos Positivos*.
- **VN**: cantidad de elementos que resultaron *Verdaderos Negativos*.

Resultado de la clasificación	**Valor real: positivo**	**Valor real: negativo**
Positivo	*Verdadero Positivo*	*Falso Positivo*
Negativo	*Falso Negativo*	*Verdadero Negativo*

Cuadro 4.2: Los cuatro posibles resultados al aplicar un clasificador binario sobre un elemento del cual su clase es *conocida* con anterioridad.

- **FP**: cantidad de elementos que resultaron *Falsos Positivos*.
- **FN**: cantidad de elementos que resultaron *Falsos Negativos*.

Con estos valores se pueden definir las métricas que se detallan a continuación.

Precisión

La *precisión* del clasificador se define como el número de *Verdaderos Positivos* (es decir, la cantidad de elementos clasificados correctamente en la clase positiva) dividido por el total de elementos clasificados en la clase positiva (esto es, la suma de los *Verdaderos Positivos* y los *Falsos Positivos*). Se puede expresar como:

$$Precision = \frac{VP}{VP + FP}. \tag{4.1}$$

Una precisión de 1.0 para un clasificador significa que todo elemento asignado a la clase positiva pertenece realmente a dicha clase, aunque no dice nada acerca de los elementos de la clase positiva que fueron asignados incorrectamente a la clase negativa.

Exhaustividad

La *exhaustividad* de un clasificador (*recall* en la literatura en inglés) se define como el número de *Verdaderos Positivos* dividido por el número total de elementos que realmente pertenecen a la clase positiva (es decir, la suma de los *Verdaderos Positivos* y los *Falsos Negativos*). Se puede expresar de la siguiente manera:

$$Exhaustividad = \frac{VP}{VP + FN}. \tag{4.2}$$

Un valor de 1.0 de exhaustividad indica que todos los elementos de la clase positiva fueron asignados a la clase positiva, aunque no dice nada acerca de cuántos elementos fueron además incorrectamente asignados en dicha clase.

Por lo general, las métricas de precisión y exhaustividad no se analizan por separado. Una forma de análisis es observar el valor de una, *fijando* un nivel en la otra (por ejemplo, la precisión con un nivel de exhaustividad de 0.80). Además, existen métricas que combinan ambos valores, como es la *medida-F* o *valor-F*:

$$F_1 = 2\,\frac{Precision \times Exhaustividad}{Precision + Exhaustividad}. \tag{4.3}$$

Exactitud

La *exactitud* (*accuracy* en la literatura en inglés) se define como la proporción de resultados verdaderos (*Verdaderos Positivos* y *Verdaderos Negativos*) entre el total de elementos examinados (es decir, el tamaño del conjunto de evaluación). Expresado como una fórmula resulta:

$$Exactitud = \frac{VP + VN}{VP + FP + VN + FN}. \tag{4.4}$$

4.2.2. Precisión Promedio Interpolada

La *Precisión Promedio Interpolada* (AP) [Manning et al., 2008] es una métrica definida en el contexto de recuperación de información y se calcula a partir de una lista rankeada de resultados devuelta por un algoritmo.

En el contexto de ranking, se pueden redefinir los conceptos de precisión y exhaustividad considerando los primeros k resultados de la lista rankeada de resultados.

La precisión se define como la proporción de todos los ejemplos por encima de ese k que pertenecen a la clase positiva.

La exhaustividad se define como la proporción de todos los ejemplos positivos que fueron rankeados por encima del rango dado.

Como ya se mencionó antes, se puede calcular la precisión para un nivel de exhaustividad dado. Esto permite definir una curva de precisión/exhaustividad. La AP resume la forma de esta curva:

$$AP = \frac{1}{11} \sum_{r \in \{0,0,1,\ldots,1\}} p_{interp}(r). \tag{4.5}$$

La precisión a cada nivel de exhaustividad r es *interpolada* tomando la precisión máxima para valores de exhaustividad mayores que r:

$$p_{interp}(r) = \max_{\tilde{r}:\, \tilde{r} \geq r} p(\tilde{r}) \tag{4.6}$$

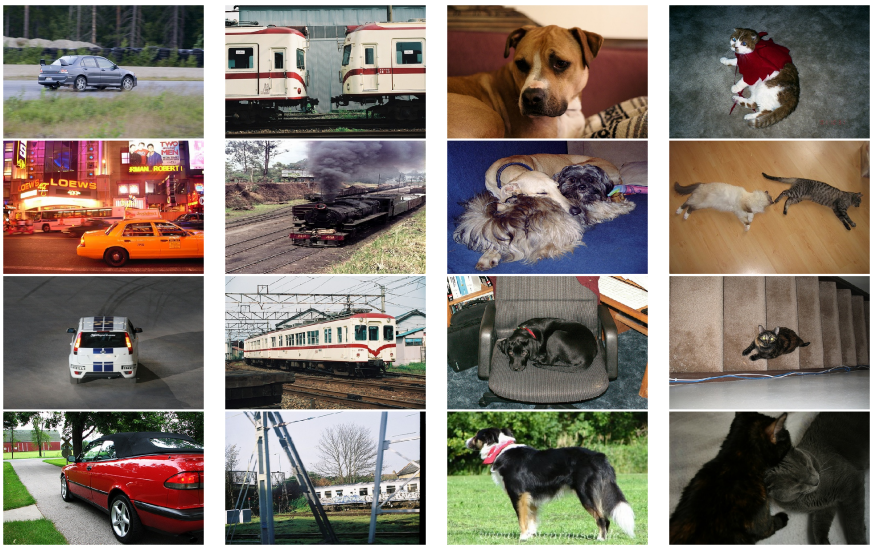
donde $p(\tilde{r})$ es la precisión medida con un nivel de exhaustividad r.

Además, se define la *precisión promedio media (mAP)* como la media de la precisión promedio de cada clase:

$$mAP = \frac{1}{K} \sum_{k=1}^{K} AP(k). \tag{4.7}$$

4.3. Base de comparación

Para tener una base de comparación (*baseline* de la literatura en inglés), se utiliza la técnica de uno-contra-todos con *SVM lineales*, tal como se explica en la Sección 2.2.5. Esta es utilizada ampliamente en la literatura, y ha demostrado ser el estado del arte para el problema de clasificación en gran escala con modelos lineales [Sánchez et al., 2013] (utilizando modelos no lineales se pueden obtener aún mejores resultados).



(a) Clase *car* (b) Clase *train* (c) Clase *dog* (d) Clase *cat*

Figura 4.1: Ejemplos de imágenes de las clases *car*, *train*, *dog* y *cat* de Pascal VOC 2007.

Dado que el foco es comparar algoritmos de clasificación, se mantiene invariante la técnica de representación de imágenes, es decir, se utiliza la configuración de FV descripta en la Sección 4.1.

4.4. Experimentos en PASCAL VOC 2007

4.4.1. Descripción del banco de datos

En esta sección se describen los experimentos realizados en PASCAL VOC 2007 [Everingham et al., 2009]. Este banco de datos es lo suficientemente pequeño como para poder probar numerosos experimentos en tiempo razonable y realizar las correcciones pertinentes, antes de hacer pruebas en conjuntos más grandes como los que se presentarán más adelante. Cuenta con 20 clases de objetos y 9.963 imágenes. Cada imagen puede tener más de una clase, en caso de tener presente más de un objeto en la misma (por ejemplo, en la imágen de una persona junto con su perro).

Uno de los objetivos al diseñar este banco de datos fue que exista gran variabilidad en las imágenes, en términos de tamaños de los objetos, orientación, posición, iluminación y oclusión. Además, se buscó evitar favorecer imágenes en la que los objetos estuvieran centrados o con buena iluminación.

Para lograrlo se utilizaron imágenes de *Flickr*, una plataforma web que permite a sus usuarios subir fotografías y compartirlas con otros. Las imágenes se seleccionaron de manera aleatoria entre los resultados devueltos por consultas con determinadas palabras claves a esta plataforma. Luego se realizó la anotación de las mismas de manera manual. En la Figura 4.1 se muestran ejemplos de algunas imágenes de este banco de datos.

Algoritmo y parámetros	***mAP* en conjunto *test***
WSABIE - FV 2,5K $d = 20$; $\gamma = 0{,}01$; $l = 5$; $C = 5$	45,3
WSABIE - FV 40K $d = 20$; $\gamma = 0{,}1$; $l = 5$; $C = 5$	47,38
OvR SVM - FV 2,5K $\alpha = 1 \times 10^{-4}$	52,65
OvR SVM - FV 40K $\alpha = 1 \times 10^{-4}$	55,73

Cuadro 4.3: Experimentos en Pascal VOC 2007. Resultados de WSABIE y de la base de comparación para dos magnitudes de FV.

Las imágenes están divididas en tres conjuntos: *train*, *val* y *test*. El protocolo de evaluación utilizado consiste en ajustar parámetros entrenando en *train* y evaluando en *val*, y luego con la mejor configuración encontrada, volver a entrenar utilizando ambos *train* y *val* y evaluar en *test*.

La métrica que se utiliza es la Precisión Promedio Interpolada (mAP), explicada en la Sección 4.2.2.

4.4.2. Resultados

En el Cuadro 4.3 se presentan los mejores resultados de WSABIE y de la base de comparación definida en la Sección 4.3, reportados como la *mAP* en el conjunto *test*. Como ya se explicó, los parámetros fueron obtenidos utilizando el conjunto de validación *val*.

Observar primero que hay una diferencia en los resultados obtenidos con FV 2,5K y FV 40K. Esto está en línea con lo que reportan [Sánchez et al., 2013], es decir, mayor dimensionalidad en la representación permite obtener mejores resultados en la clasificación.

La base de comparación presenta un mejor rendimiento que WSABIE, con una diferencia de 7,35 puntos en el caso de FV 2,5K, y de 8,35 con FV 40K. El hecho de que la diferencia no sea tan significativa da un indicio de que la implementación que se realizó en este trabajo funciona correctamente. Asegurar esto es muy importante antes de emprender experimentos más grandes, donde *debuggear* es más costoso. Por otro lado, WSABIE está pensado para problemas de gran escala, por lo cual era esperado un rendimiento inferior en un experimento sobre un conjunto pequeño.

Observar además que la tasa de aprendizaje es diferente en los dos modelos WSABIE. Mientras que para el caso de FV 2,5K se utilizó $\gamma = 0{,}01$, durante el proceso de validación de parámetros se observó que el caso FV 40K convergía más rápido con $\gamma = 0{,}1$.

Cabe destacar que se obtuvo un resultado levemente mejor (48.34 *mAP* con FV de 2,5K), pero utilizando una dimensionalidad $d = 100$. Sin embargo, se buscó que la dimensión del espacio de representación sea menor o igual que K, que es veinte en este caso, para que el tamaño de los modelos de WSABIE sea menor o igual que el de SVM con esquema uno-contra-todos.

Finalmente, mientras que para la técnica de uno-contra-todos utilizando SVM hay sólo dos parámetros que configurar y validar (número de iteracio-

nes y α), para WSABIE se necesitan cinco, tal como se listó en la Sección 3.4.3. Esto hace que el proceso de validación en el segundo algoritmo sea más costoso, pues hay más combinaciones que se pueden probar.

4.5. Experimentos en CUB-200-2011

4.5.1. Descripción del banco de datos

En esta sección se describen los experimentos realizados en Caltech-UCSD Birds 200-2011 (CUB-200-2011) [Wah et al., 2011], que es una extensión de CUB-200 [Welinder et al., 2010] con casi el doble de imágenes. Este banco de datos consiste en 11.788 imágenes de 200 especies de aves, en su mayoría de América del Norte, y fue diseñado para estudiar el problema de *categorización subordinada*, es decir, clasificación en subclases más granulares de una clase general (i.e., aves en este caso).

Para la construcción de este banco de datos, los autores partieron de un conjunto de especies de aves, compiladas de una guía online. Luego, se recolectaron todas las imágenes presentes en el artículo de *Wikipedia*[1] de cada una de dichas especies. Se descartaron las especies para las que dicho artículo no existía o no contenía imágenes. Las categorías restantes fueron utilizadas como términos de búsqueda en *Flickr*, de donde se obtuvieron las imágenes.

Luego, se utilizó *Amazon Mechanical Turk*[2] *(AMT)* para realizar una verificación manual de que las imágenes obtenidas de Flickr realmente correspondieran a la especie de ave asignada. Además, cada una de ellas se anotó con un cuadro delimitador (es decir, la región de la imagen donde se encuentra el ave), se realizó una localización de sus partes (por ejemplo, las coordenadas del píxel donde se encuentra el pico del ave) y se establecieron etiquetas de atributos (por ejemplo, la forma del pico).

En la Figura 4.2 se muestran ejemplos de algunas imágenes de este banco de datos.

Las imágenes están divididas en dos conjuntos, uno para entrenamiento y otro para test. Los autores no definen un protocolo de evaluación, pero recomiendan que se utilice la partición de datos que ellos proponen. No se utilizaron los datos de cuadro delimitador, localización de partes o etiquetas de atributos, sólo la categoría a la que pertenece cada imagen. Las métricas que se reportan son la precisión, exhaustividad y exactitud, explicadas en la Sección 4.2.1

4.5.2. Resultados

En el Cuadro 4.4 se presentan los mejores resultados de WSABIE y de la base de comparación definida en la Sección 4.3.

Nuevamente se observa una diferencia entre los resultados de FV de 2,5K y FV de 40K.

Al igual que en el banco de datos anterior, la base de comparación presenta un mejor rendimiento que WSABIE.

En este caso, sin embargo, se pudo utilizar una dimensión de espacio embedding cuatro veces menor que K. Por lo tanto, el modelo resultante del algoritmo

[1] http://www.wikipedia.org/
[2] https://www.mturk.com/

(a) *Common Yellowthroat* (b) *Elegant Tern* (c) *Magnolia Warbler* (d) *Pileated Woodpecker*

Figura 4.2: Ejemplos de imágenes de las clases *Common Yellowthroat*, *Elegant Tern*, *Magnolia Warbler* y *Pileated Woodpecker* de CUB-200-2011.

Algoritmo y parámetros	**Exactitud**	**Precisión**	**Exhaustividad**
WSABIE - FV de 2,5K $d = 50$; $\gamma = 0{,}01$; $l = 10$; $C = 5$	8,8	8,0	9,0
WSABIE - FV de 40K $d = 50$; $\gamma = 0{,}01$; $l = 10$; $C = 5$	12,7	13,0	13,0
OvR SVM - FV de 2,5K $\alpha = 1 \times 10^{-4}$	10,08	10,0	10,0
OvR SVM - FV de 40K $\alpha = 1 \times 10^{-4}$	16,32	16,0	16,0

Cuadro 4.4: Experimentos en CUB200-2011. Resultados de WSABIE y de la base de comparación para dos magnitudes de FV.

WSABIE ocupa menos espacio en memoria que el de la base de comparación, aunque esta diferencia sea despreciable debido a la baja magnitud de este banco de datos.

4.6. Experimentos en SUN397

4.6.1. Descripción del banco de datos

En esta sección se describen los experimentos realizados en SUN397 [Xiao et al., 2010, 2014]. Este banco de datos consiste en 108.754 imágenes de 397 categorías de escenas, por ejemplo: bosque, estación de servicio, habitación, edificio, iglesia, entre otros.

Para la construcción del banco de datos, los autores comenzaron por elaborar una taxonomía de escenas. Primero, seleccionaron términos de *WordNet* [Fellbaum, 1998] que describieran escenas, lugares y ambientes. Con este proceso se obtuvieron 2.500 términos iniciales, que luego de combinar sinónimos y separar escenas con diferentes identidades visuales (por ejemplo, la vista interior y exterior de una iglesia), se llegó a una lista refinada de 908 categorías.

Para cada categoría se obtuvieron imágenes utilizando motores de búsqueda, las cuales fueron examinadas manualmente para confirmar que correspondieran a la categoría asignada. Además se descartaron imágenes degeneradas (en blanco y negro, con rotación incorrecta, colores distorsionados, etc) y duplicadas.

Finalmente se seleccionaron las 397 categorías para las cuales existían al menos 100 imágenes.

En la Figura 4.3 se muestran ejemplos de algunas imágenes de este banco de datos.

Junto con el banco de datos se proveen 10 particiones, cada una de las cuales contiene 50 imágenes para entrenamiento y 50 imágenes para evaluación por categoría.

En este trabajo se utiliza la partición número 1, y se reporta la precisión, exhaustividad y exactitud obtenida en la misma.

4.6.2. Resultados

En el Cuadro 4.5 se presentan los mejores resultados de WSABIE y de la base de comparación definida en la Sección 4.3.

La mayor dimensionalidad de FV nuevamente produce una mejora en las métricas. Además, la base de comparación tiene mejor rendimiento que WSABIE, al igual que en los experimentos anteriores. De la misma manera que en los experimentos sobre CUB-200-2011, se pudo utilizar una dimensión de espacio embedding aproximadamente cuatro veces menor que K.

4.7. Experimentos en ILSVRC'10

4.7.1. Descripción del banco de datos

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) es una competencia de clasificación y detección de cientos de categorías de objetos y millones de imágenes. Se viene realizando anualmente desde el 2010 hasta la actuali-

(a) *Abbey* (b) *Library* (c) *Nuclear Power Plant* (d) *Windfarm*

Figura 4.3: Ejemplos de imágenes de las clases *abbey*, *library*, *nuclear power plant* y *windfarm* de SUN397.

Algoritmo y parámetros	**Exactitud**	**Precisión**	**Exhaustividad**
WSABIE - FV de 2,5K $d = 100$; $\gamma = 0{,}07$; $l = 1$; $C = 5$	26,25	28,0	26,0
WSABIE - FV de 40K $d = 100$; $\gamma = 0{,}07$; $l = 1$; $C = 5$	31,86	31,0	31,0
OvR SVM - FV de 2,5K $\alpha = 1 \times 10^{-5}$	32,54	33,0	33,0
OvR SVM - FV de 40K $\alpha = 1 \times 10^{-4}$	39,69	40,0	40,0

Cuadro 4.5: Experimentos en SUN397. Resultados de WSABIE y de la base de comparación para dos magnitudes de FV.

(a) *Apiary* (b) *Bonsai* (c) *French fries* (d) *Jellyfish*

Figura 4.4: Ejemplos de imágenes de las clases *apiary*, *bonsai*, *french fries* y *jellyfish* de ILVRC'10.

dad, contando con la participación de más de 50 instituciones, y se ha convertido en un punto de referencia estándar en la literatura de reconocimiento de objetos en gran escala.

En este trabajo se utilizó el banco de datos de la edición 2010 de esta competencia (ILSVRC'10), que cuenta con 1.000 clases de objetos y aproximadamente 1,4 millones de imágenes. El mismo es un subconjunto del banco de datos *ImageNet* [Deng et al., 2009], que al momento de escritura cuenta con 14.197.122 imágenes y 21.841 categorías verificadas manualmente. A continuación se describe brevemente su proceso de construcción.

ImageNet está organizado de acuerdo a la jerarquía definida en *WordNet* [Fellbaum, 1998], en el que cada concepto es descripto por múltiples palabras o frases llamadas *synsets* (del inglés *synonym set*, conjunto de sinónimos).

Las imágenes se obtuvieron de los resultados de motores de búsqueda en los que se emplearon las palabras de los synsets como consulta. Para cada synset se recolectaron aproximadamente 10K imágenes, que luego fueron verificadas y filtradas manualmente, utilizando *Amazon Mechanical Turk*. Se implementó un mecanismo de control de calidad, pues estos procesos manuales son propensos a errores y además, en ocasiones los usuarios de AMT no siguen las consignas planteadas. La solución consistió en tener múltiples usuarios independientes para clasificar una misma imagen, considerándola positiva sólo si había consenso entre ellos. El resultado final de este proceso fue de entre 500 y 1.000 imágenes por synset.

En la Figura 4.4 se muestran ejemplos de algunas imágenes del subconjunto ILSVRC'10.

Las imágenes de ILSVRC'10 están divididas en tres conjuntos: 1.2 millones de imágenes para entrenamiento, 50.000 imágenes para validación y 150.000 imágenes para evaluación. Se reportan la exactitud y la *exactitud en 5*. Esta última es estándar en ILVRC'10 y consiste en contabilizar un resultado como correcto si la clase verdadera está entre los primeros 5 lugares de la lista rankeada devuelta por el algoritmo.

4.7.2. Resultados

Los experimentos se corrieron sobre una computadora personal con las siguientes prestaciones: procesador Intel Core i7-4710HQ 2.5GHz, 16 GB DDR3 de RAM y disco SSD de 500 GB. Realizar un experimento en un banco de datos del tamaño de ILSVRC'10 utilizando el *hardware* descripto, es un desafío interesante por diversas razones:

- Las imágenes originales pesan 155GB en total, por lo tanto obtenerlas desde Internet y manipularlas son procesos costosos. La descarga, con velocidad promedio de 1MB/s, demoró aproximadamente 45 horas.
- El procesamiento de las imágenes para obtener los FV con 16 Gaussianas llevó 10,5 horas, ejecutando 7 hilos en paralelo en el hardware antes descripto.
- Como ya se mostró en el Cuadro 4.1, los FV 2,5K para este banco de datos pesan aproximadamente 28GB, por lo tanto, no entran todos juntos en la memoria de la computadora utilizada. Es por esto que se realizó un procesamiento *online* durante el entrenamiento de WSABIE, es decir, leyendo los vectores a medida que el algoritmo los necesitaba en lugar de cargarlos todos en memoria al inicio.
- En los experimentos anteriores, el entrenamiento de los SVM lineales se realizó leyendo todos los FV en memoria antes de iniciar el algoritmo. En este caso, se realizó una lectura en bloques por las limitaciones de memoria. En cada iteración, se leía una cantidad prefijada de FV y se los pasaba al algoritmo para su procesamiento.
- El caso de FV con 256 Gaussianas no pudo realizarse por limitaciones de recursos.

En el Cuadro 4.6 se presentan los resultados obtenidos con WSABIE y la base de comparación definida en la Sección 4.3.

Por primera vez en los experimentos de este trabajo, el algoritmo WSABIE tiene mejor rendimiento que la base de comparación. Además, esto se obtuvo utilizando dimensión $d < K$, logrando buenos resultados con valores de d que son la décima parte de K. Esto muestra el poder de WSABIE cuando la cantidad de clases escala: el tamaño de los modelos se mantiene pequeño gracias a la baja dimensionalidad del espacio embedding.

Tiempos de entrenamiento y evaluación

Como ya se mencionó, se quiere que el tiempo de entrenamiento del modelo también sea escalable. Los experimentos mostraron que WSABIE es ampliamente superior en este caso, como se muestra en el Cuadro 4.7

Algoritmo y parámetros	**Exactitud**	**Exactitud en 5**
WSABIE - FV de 2,5K $d = 100$; $\gamma = 0{,}01$; $l = 50$; $C = 50$	19,31	40,21
WSABIE - FV de 2,5K $d = 200$; $\gamma = 0{,}01$; $l = 50$; $C = 50$	20,08	40,78
WSABIE - FV de 2,5K $d = 300$; $\gamma = 0{,}01$; $l = 50$; $C = 50$	20,33	41,19
OvR SVM - FV de 2,5K $\alpha = 1 \times 10^{-5}$	17,76	30,28

Cuadro 4.6: Experimentos en ILVRC'10. Resultados de WSABIE y de la base de comparación para FV 2,5K.

Algoritmo y parámetros	**T. Entrenamiento**	**T. Evaluación**
WSABIE - FV de 2,5K $d = 100$; $\gamma = 0{,}01$; $l = 50$; $C = 50$	$4{,}7hrs$	$4{,}56mins$
WSABIE - FV de 2,5K $d = 200$; $\gamma = 0{,}01$; $l = 50$; $C = 50$	$8{,}22hrs$	$6{,}3mins$
WSABIE - FV de 2,5K $d = 300$; $\gamma = 0{,}01$; $l = 50$; $C = 50$	$9{,}45hrs$	$7mins$
OvR SVM - FV de 2,5K $\alpha = 1 \times 10^{-5}$	$61hrs$	$2hrs$

Cuadro 4.7: Experimentos en ILVRC'10. Tiempos de entrenamiento y evaluación de WSABIE y de la base de comparación para FV 2,5K.

Observar que a medida que la dimensión del espacio embedding crece, el tiempo de entrenamiento y evaluación también se incrementan. Por lo tanto, esto es un factor a tener en cuenta a la hora de elegir este parámetro. En este caso, la diferencia de desempeño entre los tres modelos de WSABIE es despreciable y por lo tanto podría considerarse que no se justifica el aumento en el tiempo de entrenamiento y evaluación.

Tener en cuenta que la técnica de uno-contra-todos con SVM lineales se puede paralelizar, tal como se explicó en la Sección 2.2.5, lo cual reduciría el tiempo de entrenamiento y evaluación.

Capítulo 5

Conclusiones y trabajos futuros

5.1. Conclusiones

En este trabajo se estudió el uso de espacios embedding para el problema de clasificación, en particular, el algoritmo WSABIE para el problema de clasificación de imágenes en gran escala. Se utilizó una base de comparación habitual en la literatura, como lo es uno-contra-todos con SVM lineales.

En los experimentos de pequeña y mediana escala, la base de comparación tuvo un rendimiento superior desde el punto de vista de la exactitud, precisión, exhaustividad o mAP (dependiendo el banco de datos en cuestión). Las diferencias en cuanto al tamaño de los modelos generados es despreciable en estos casos.

La cantidad de parámetros que se pueden configurar en WSABIE hace que la fase de evaluación sea más costosa, lleve más tiempo, y en el caso de los experimentos chicos, esto no lleva a mejoras importantes en los resultados.

Por lo tanto, la simpleza en el entrenamiento y el buen rendimiento de la técnica de uno-contra-todos nos permite concluir que es preferible sobre WSABIE para estos casos.

Esta situación cambia cuando se trabaja en bancos de datos más grandes, como es ILVRC'10, mostrado en la Sección 4.7. En este caso, el rendimiento de WSABIE es mejor desde el punto de vista de la exactitud, y además, utiliza modelos de menor tamaño y con una velocidad de entrenamiento y evaluación ampliamente superior.

5.2. Trabajos futuros

En esta sección se presenta una lista de posibles direcciónes en las que podría continuarse este trabajo:

- Realizar experimentos sobre *ImageNet Fall 2011* (14 millones de imágenes y 21k clases) [Deng et al., 2009]. Esto es un desafío pues al ser tanta cantidad de datos (el archivo TAR que contiene todas las imágenes pesa 1.2TB), se dificulta su manipulación, almacenamiento y procesamiento.

Por ejemplo, descargar todo el banco de datos desde Internet podría demorar unos 15 días con una velocidad de descarga de $1mb/s$.

- Construir bancos de datos haciendo consultas a motores de búsqueda o sitios que tienen *indexadas* una gran cantidad de imágenes (Flickr, plataformas e-commerce, etc) y luego utilizarlos para entrenar WSABIE. De esta manera, se podrían realizar a experimentos de escala aún mayor. Por ejemplo, [Gupta et al., 2014] reporta resultados de una versión mejorada de WSABIE, llamada *WSABIE++*, sobre un banco de datos de 40 millones de imágenes y 97k clases, lo que demandó un tiempo de entrenamiento de un mes aproximadamente. En la Universidad Nacional de Córdoba se cuenta con el *cluster Mendieta*[1], donde sería factible hacer experimentos de esta escala.

- Analizar posibilidades de paralelización de WSABIE, en particular en la etapa de muestreo, donde se busca un $\bar{y}$ tal que $l + f(x, \bar{y}) > f(x, y)$.

- Basado en la propuesta de [Weston et al., 2013a], explorar la utilización de modelos no lineales en combinación con los embeddings lineales definidos en la Sección 3.2.

- Para los experimentos, se implementó WSABIE en Python. Se podría realizar una re-implementación en C++ con el objetivo de hacer un manejo más eficiente de la memoria y acelerar la velocidad de cómputo. Además, dado que el algoritmo realiza numerosas multiplicaciones matriciales, la utilización de GPUs podría mejorar significativamente los tiempos de entrenamiento y evaluación.

- En este trabajo se consideró el problema de clasificación de imágenes, que esencialmente, consiste en obtener una lista de clases a partir de una imagen dada. Se puede explorar otro tipo de consultas, como imagen-imagen (obtener una lista de imágenes relacionadas a una imagen dada), o etiqueta-imagen (obtener una lista de imágenes a partir de una palabra). Esto se puede lograr utilizando la noción de *similitud* en el espacio embedding.

[1] http://ccad.unc.edu.ar/

Bibliografía

Zeynep Akata. *Contributions to large-scale learning for image classification.* PhD thesis, Université de Grenoble, 2014.

WA Awad and SM ELseuofi. Machine learning methods for e-mail classification. 2011.

Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 46–54. Morgan Kaufmann, 1998. ISBN 1-55860-556-8.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

Ken Chatfield, Victor S. Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In Jesse Hoey, Stephen J. McKenna, and Emanuele Trucco, editors, *British Machine Vision Conference, BMVC 2011, Dundee, UK, August 29 - September 2, 2011. Proceedings*, pages 1–12. BMVA Press, 2011. ISBN 1-901725-43-X. doi: 10.5244/C.25.76. URL `http://dx.doi.org/10.5244/C.25.76`.

Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. *Workshop on statistical learning in computer vision, ECCV*, 1(1-22):1–2, 2004.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990. doi: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9. URL `http://dx.doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPRW.2009.5206848. URL `http://dx.doi.org/10.1109/CVPRW.2009.5206848`.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In Adam Tauman Kalai

and Mehryar Mohri, editors, *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 257–269. Omnipress, 2010. ISBN 978-0-9822529-2-5. URL http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf#page=265.

Mark Everingham, Luc Van Gool, C. K. I. Williams, J. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge, 2009.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

Obi L Griffith, François Pepin, Oana M Enache, Laura M Heiser, Eric A Collisson, Paul T Spellman, and Joe W Gray. A robust prognostic signature for hormone-positive node-negative breast cancer. 2013.

Maya R. Gupta, Samy Bengio, and Jason Weston. Training highly multiclass linear classifiers. *Journal Machine Learning Research (JMLR)*, pages 1461–1492, 2014. URL http://jmlr.org/papers/volume15/gupta14a/gupta14a.pdf.

Yongzhen Huang, Zifeng Wu, Liang Wang, and Tieniu Tan. Feature coding in image classification: A comprehensive study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(3):493–506, 2014. doi: 10.1109/TPAMI.2013.113. URL http://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.113.

Herve Jegou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3304–3311, 2010. doi: 10.1109/CVPR.2010.5540039. URL http://dx.doi.org/10.1109/CVPR.2010.5540039.

Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263. URL http://doi.ieeecomputersociety.org/10.1109/MC.2009.263.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.

Florent Perronnin and Christopher R. Dance. Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*. IEEE Computer Society, 2007. ISBN 1-4244-1179-3. doi: 10.1109/CVPR.2007.383266. URL http://dx.doi.org/10.1109/CVPR.2007.383266.

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*, volume 6314 of *Lecture Notes in Computer Science*, pages 143–156. Springer, 2010. ISBN 978-3-642-15560-4. doi: 10.1007/978-3-642-15561-1_11. URL http://dx.doi.org/10.1007/978-3-642-15561-1_11.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015. doi: 10.1007/s11263-015-0816-y. URL http://dx.doi.org/10.1007/s11263-015-0816-y.

Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013. doi: 10.1007/s11263-013-0636-x. URL http://dx.doi.org/10.1007/s11263-013-0636-x.

Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*, pages 1470–1477. IEEE Computer Society, 2003. ISBN 0-7695-1950-4. doi: 10.1109/ICCV.2003.1238663. URL http://doi.ieeecomputersociety.org/10.1109/ICCV.2003.1238663.

Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.*, 6(2):215–219, March 1994. ISSN 0899-7667. doi: 10.1162/neco.1994.6.2.215. URL http://dx.doi.org/10.1162/neco.1994.6.2.215.

Tony Van Gestel, Bart Baesens, Joao Garcia, and Peter Van Dijcke. A support vector machine approach to credit scoring. *Bank en Financiewezen*, (2):73–82, 2003.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Jason Weston, Samy Bengio, and Philippe Hamel. Large-scale music annotation and retrieval: Learning to rank in joint semantic spaces. *CoRR*, abs/1105.5196, 2011a. URL http://arxiv.org/abs/1105.5196.

Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Th-*

ree, IJCAI'11, pages 2764–2770. AAAI Press, 2011b. ISBN 978-1-57735-515-1. doi: 10.5591/978-1-57735-516-8/IJCAI11-460. URL http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-460.

Jason Weston, Ron J. Weiss, and Hector Yee. Affinity weighted embedding. *CoRR*, abs/1301.4171, 2013a. URL http://arxiv.org/abs/1301.4171.

Jason Weston, Hector Yee, and Ron J. Weiss. Learning to rank recommendations with the k-order statistic loss. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 245–248, New York, NY, USA, 2013b. ACM. ISBN 978-1-4503-2409-0. doi: 10.1145/2507157.2507210. URL http://doi.acm.org/10.1145/2507157.2507210.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3485–3492, 2010. doi: 10.1109/CVPR.2010.5539970. URL http://dx.doi.org/10.1109/CVPR.2010.5539970.

Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *IJCV*, 2014.

Xi Zhou, Kai Yu, Tong Zhang, and Thomas S. Huang. Image classification using super-vector coding of local image descriptors. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision - ECCV 2010 - 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part V*, volume 6315 of *Lecture Notes in Computer Science*, pages 141–154. Springer, 2010. ISBN 978-3-642-15554-3. doi: 10.1007/978-3-642-15555-0_11. URL http://dx.doi.org/10.1007/978-3-642-15555-0_11.