CGLAD: GLAD en problemas de Big Crowdsourced Data

Enrique G. Rodrigo
Departamento de Sistemas Informáticos
Universidad de Castilla-La Mancha
Albacete, España
Enrique.GRodrigo@uclm.es

Juan A. Aledo
Departamento de Matemáticas
Universidad de Castilla-La Mancha
Albacete, España
JuanAngel.Aledo@uclm.es

José A. Gámez
Departamento de Sistemas Informáticos
Universidad de Castilla-La Mancha
Albacete, España
Jose.Gamez@uclm.es

***Resumen*—En este artículo proponemos una mejora del algoritmo GLAD con el fin de mejorar su funcionamiento en problemas con grandes conjuntos de datos, en términos de eficiencia y de precisión del modelo resultante. El algoritmo GLAD permite aprender a partir de datos procedentes de múltiples anotadores, teniendo en cuenta su capacidad y la dificultad de las instancias que se predicen. Sin embargo, debido al número de parámetros del modelo, este no escala bien a grandes cantidades de datos, sobre todo si se requiere que el tiempo de ejecución sea bajo. Nuestra propuesta, que llamamos CGLAD, resuelve en gran medida estos problemas mediante *clustering* a partir de vectores procedentes de factorización de matrices, lo que permite reducir el número de parámetros del modelo y, en general, facilitar el aprendizaje de modelos que siguen la estrategia de GLAD.**

***Index Terms*—aprendizaje automático no estándar, crowdsourcing, múltiples anotadores, débilmente supervisado**

I. Introducción

El algoritmo GLAD [5] permite abordar problemas de aprendizaje automático a partir de múltiples anotadores (por ejemplo, los que proceden de plataformas de *crowdsourcing*) [7]. Este problema se enmarca dentro del aprendizaje automático no estándar [2], el cual aborda el análisis de conjuntos de datos que difieren en ciertas características del aprendizaje de aprendizaje automático tradicional. Concretamente, en el aprendizaje a partir de múltiples anotadores no disponemos de la etiqueta verdadera de los ejemplos, si no que disponemos de varias anotaciones para cada ejemplo procedentes de anotadores de calidad desconocida, por lo que se obtiene un conjunto de datos como el de la Tabla I.

$\mathbf{Y_1}$	$\mathbf{Y_2}$	...	$\mathbf{Y_N}$
0	0	...	1
1	1	...	-
0	-	...	0
1	0	...	1
1	-	...	1
...	...	...	...

Tabla I: Ejemplo de conjunto de anotaciones

Este trabajo ha sido parcialmente financiado por la Agencia Estatal de Investigación (AEI) y el Fondo Europeo de Desarrollo Regional (FEDER, UE) mediante los proyectos TIN2016-77902-C3-1-P y TIN2016-82013-REDT. Enrique G. Rodrigo también ha sido financiado por el MECD mediante la beca FPU15/02281.

Aparte de estas anotaciones es posible obtener más características de las instancias, existiendo algoritmos [4] que permiten hacer uso de estas características a medida que se aprende un modelo. Sin embargo, gran parte de los algoritmos en esta línea trabaja en resolver únicamente el problema de agregación de estas etiquetas [8]. El objetivo último es poder aprender un modelo a partir de estas anotaciones que permita predecir la clase verdadera para cada instancia, que entonces podrá utilizarse en un algoritmo de aprendizaje automático tradicional. El enfoque más sencillo y empleado es el de usar la clase más frecuente como entrada de un algoritmo de aprendizaje automático, método comúnmente conocido como *Majority Voting* cuando tenemos una variable de salida de tipo discreto (en el caso continuo usaríamos la media). Sin embargo, en la actualidad, existen algoritmos más efectivos a la hora de agregar opiniones puesto que permiten tener en cuenta información tal como la experiencia de los anotadores [1] o, incluso, la dificultad de cada ejemplo [5]. Esta información no solo es útil para estimar la verdadera etiqueta, si no que puede resultar interesante en numerosos problemas.

Este tipo de algoritmos es especialmente interesante cuando el tamaño del problema es mayor, ya que es en este caso cuando es más complejo obtener un gran número de anotaciones fiables a partir de expertos (o, en general, de personas en las que podamos confiar) en un determinado problema. Es, por tanto, recomendable, que los algoritmos utilizados en la agregación de etiquetas puedan ser escalables a una gran cantidad de datos, de forma que puedan aplicarse también en estos casos.

En este artículo exponemos algunos problemas relacionados con la escalabilidad del algoritmo GLAD, uno de los algoritmos principales en la agregación de anotaciones cuando uno está interesado no solo en evaluar la calidad de los anotadores sino también en la dificultad de los ejemplos. Por otro lado, proponemos una mejora de este algoritmo, CGLAD, que permite abordar problemas de mayor envergadura y facilita en gran medida el uso de este en todos los contextos, añadiendo una mayor estabilidad frente a ligeros cambios en la configuración del algoritmo. Por último, ofrecemos una serie de comparativas con ambos métodos y exponemos nuestras conclusiones al respecto.

II. El algoritmo GLAD

El algoritmo GLAD [5] permite resolver problemas de aprendizaje con clase binaria usando múltiples anotaciones. A diferencia de otros algoritmos [1], [3], [4], permite estimar tanto la precisión de cada anotador como la dificultad de cada ejemplo. A continuación describiremos en qué consiste este algoritmo, y expondremos sus problemas de escalabilidad, los cuales pretende resolver nuestra propuesta.

II-A. Modelo

El modelo supone que la anotacion depende de tres elementos: la dificultad del ejemplo a anotar, la experiencia del anotador y la verdadera etiqueta. Los dos primeros elementos se modelan de la siguiente forma:

- **Dificultad de cada ejemplo**. Se modela usando un parámetro, $1/\beta_i \in [0, \infty)$, para cada ejemplo i, donde β_i es positivo. Si $1/\beta_i$ se acerca a 0, el ejemplo será más sencillo (anotadores con menos experiencia son capaces de anotarlo correctamente). Al contrario, si se acerca a ∞, el ejemplo sería tan ambiguo que incluso un anotador experimentado tendría solo un 50 % de probabilidad de etiquetar el ejemplo incorrectamente.
- **Experiencia del anotador**. Se modela usando un parámetro, $\alpha_j \in (-\infty, \infty)$, para cada anotador j. Si α_j se acercase a ∞, el anotador tendría tanta experiencia que siempre anotaría correctamente. Si α_j se acercase a $-\infty$ el anotador siempre anotaría incorrectamente, lo que significaría que el anotador es tan bueno como el anterior distinguiendo las clases, pero está invirtiendo la etiqueta, maliciosamente o por un malentendido. Por último, si α_j se acerca a 0, significa que el anotador no puede discriminar entre las clases (podría ser un *spammer*).

El modelo de anotación (generativo), se vale de los dos parámetros anteriores de forma que la probabilidad de que un anotador etiquete una instancia correctamente es

$$c_{ji} = p(y_i^j = y_i | \alpha_j, \beta_i) = \frac{1}{1 + e^{-\alpha_j \beta_i}}$$

donde y_i^j es la anotación del anotador j a la instancia i e y_i es la etiqueta verdadera de la instancia. De esta forma, los anotadores más experimentados (con alto valor de α_j) tienen una mayor probabilidad de etiquetar correctamente. Asimismo, si la dificultad de un ejemplo, $1/\beta_i$, es mayor, la probabilidad de que la etiqueta sea correcta se acerca a 0,5. Esto ocurre también si el valor de α_j se acerca a 0.

II-B. Inferencia

Las variables observadas son las etiquetas generadas por los anotadores. La variables ocultas son las verdaderas etiquetas y los parámetros del modelo de anotador $\boldsymbol{\alpha}$ y dificultad de las instancias $\boldsymbol{\beta}$. Para estimar las variables ocultas se puede utilizar el enfoque *Expectation-Maximization* para obtener estimaciones por máxima verosimilitud:

- **Paso E**: Sea $\mathcal{Y}_i = \{y_i^j\}$ el conjunto de anotaciones para el ejemplo i (no todos los anotadores tienen que anotar una instancia). Para obtener la probabilidad sobre la verdadera etiqueta, y_i, conocidos los valores de $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ y las anotaciones $\mathcal{Y}_i$ podemos utilizar:

$$p(y_i = k | \mathcal{Y}_i, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto p(y_i = k) \prod_j p_{ji}^k$$

donde

$$\begin{aligned} p_{ji}^k &= p(y_i^j = k | y_i, \alpha_j, \beta_i) \\ &= \begin{cases} (c_{ji})^k \cdot (1 - c_{ji})^{1-k} & \text{si } y_i = 1 \\ (c_{ji})^{1-k} \cdot (1 - c_{ji})^k & \text{si } y_i = 0 \end{cases} \end{aligned}$$

- **Paso M**: Se maximiza la siguiente función usando gradiente descendiente con respecto a los valores de $\boldsymbol{\alpha}$ y $\boldsymbol{\beta}$

$$Q(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_i E[ln(p(y_i = k))] + \sum_{ij} E[ln(p_{ji}^k)],$$

donde E es la esperanza con respecto a las estimaciones procedentes del anterior paso E.
A partir de esta se obtienen los siguientes gradientes para cada parámetro

$$\frac{\partial Q}{\partial \alpha_j} = \sum_i (p_{ji}^1 \, y_i^j + p_{ji}^0 \, (1 - y_i^j) - \sigma(\alpha_j \beta_i)) \beta_i,$$

$$\frac{\partial Q}{\partial \beta_i} = \sum_j (p_{ji}^1 \, y_i^j + p_{ji}^0 \, (1 - y_i^j) - \sigma(\alpha_j \beta_i)) \alpha_j,$$

donde $\sigma(x) = 1/(1 + e^{-x})$.

II-C. Pseudocódigo

A partir del modelado anterior podemos resumir el algoritmo mediante el pseudocódigo en el Algoritmo 1.

Algoritmo 1 Algoritmo GLAD sin optimización

1: **Paso E inicial**: Agregación por mayoría $\rightarrow y_i$
2: **for** $i = 0$ hasta converger **do**
3: **Paso M**: Optimizar mediante gradiente descendiente usando las siguientes fórmulas:

$$\frac{\partial Q}{\partial \alpha_j} = \sum_i (p_{ji}^1 \, y_i^j + p_{ji}^0 \, (1 - y_i^j) - \sigma(\alpha_j \beta_i)) \beta_i,$$

$$\frac{\partial Q}{\partial \beta_i} = \sum_j (p_{ji}^1 \, y_i^j + p_{ji}^0 \, (1 - y_i^j) - \sigma(\alpha_j \beta_i)) \alpha_j,$$

4: **Paso E**: Estimar la probabilidad de la clase para cada instancia.

$$p(y_i = k | \mathcal{Y}_i, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto p(y_i = k) \prod_j p(y_i^j = k | y_i, \alpha_j, \beta_i)$$

5: **end for**

El algoritmo comienza realizando una inicialización de las verdaderas etiquetas a la clase más frecuente presente en las anotaciones para cada ejemplo. Tras ello realiza varias iteraciones del algoritmo EM, estimando los parámetros del modelo en el paso M y volviendo a estimar la verdadera etiqueta en el paso E. Este algoritmo se detiene cuando se logra la convergencia de la función de verosimilitud o cuando se llega a un número máximo de iteraciones.

II-D. Problemas de escalabilidad

Al intentar utilizar este algoritmo en problemas con un tamaño considerable descubrimos que el modelo pierde precisión en comparación con problemas similares de menor envergadura. Para comprobarlo, hemos realizado diferentes pruebas con el algoritmo variando el tamaño de los conjuntos de datos así como la tasa de aprendizaje del algoritmo de gradiente descendiente. Esta tasa gobierna en gran medida la capacidad del algoritmo para converger hacia una solución, en especial si mantenemos constantes el umbral y el número máximo de iteraciones del algoritmo de gradiente descendiente. Para las pruebas se han utilizado conjuntos de datos simulados con las siguientes características:

- **Tamaño del conjunto**. Se ha generado una serie de conjuntos de datos con los siguientes números de instancias[1]: 5000, 10000, 20000, 40000, 80000, 160000, 320000, 640000, 1280000, 2560000.
- **Anotaciones**. Para cada uno de los conjuntos anteriores se generan 10 anotaciones realizadas por anotadores simulados usando una distribución de probabilidad discreta. Usamos las distribuciones de la Figura 1, generando 6 anotadores con una precisión alta, 2 anotadores aleatorios y 2 anotadores adversarios.

Clase	Negativa	Positiva
Negativa	0.8	0.2
Positiva	0.1	0.9

(a) Precisión alta

Clase	Negativa	Positiva
Negativa	0.5	0.5
Positiva	0.5	0.5

(b) Aleatorio

Clase	Negativa	Positiva
Negativa	0.2	0.8
Positiva	0.8	0.2

(c) Adversario

Figura 1: Tipos de anotadores generados

Para cada conjunto de datos recogemos la precisión obtenida (al tener la información de la clase verdadera podemos evaluar directamente sobre ella) para cada conjunto de datos y 3 valores de la tasa de aprendizaje: 0.1, 0.01, 0.001. Resumimos los resultados en la Figura 2. Podemos observar que al aumentar el tamaño del conjunto de datos, la precisión del método parece disminuir. Variando la tasa de aprendizaje podemos conseguir resolver algunos casos más (aumentando el tiempo de aprendizaje).

Creemos que la razón fundamental de este problema es el aumento de parámetros que se estiman. En cuanto al modelo de anotador no hay problema, ya que en los conjuntos no varía el número de anotadores. Sin embargo, al aumentar el número de instancias del problema, aumenta de forma lineal el número de parámetros del modelo de dificultad (uno por instancia). Para tamaños de problema pequeños, como puede verse en la Figura 2, el algoritmo de gradiente descendiente converge, pero a medida que aumentamos el número de parámetros la convergencia es más compleja. Parece necesario, por tanto, reducir el número de parámetros del modelo de dificultad para conseguir un aprendizaje más estable y, sobre todo, para poder utilizar el algoritmo en contextos de grandes volúmenes de datos.

[1]Los datasets están disponibles en `.csv` y `.parquet` en el siguiente enlace: http://bit.ly/caepia2018-cglad

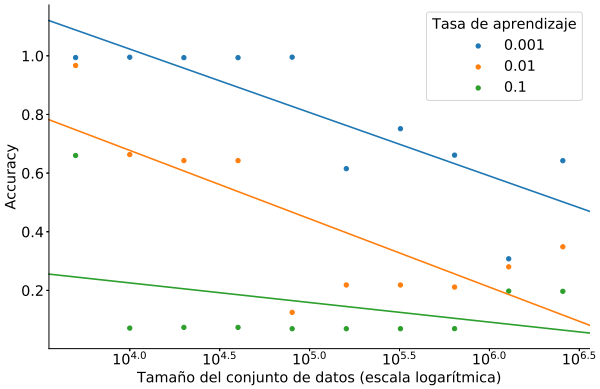

Figura 2: Problemas de escalabilidad del algoritmo GLAD.

III. CGLAD

En este artículo proponemos una mejora al algoritmo GLAD que permite reducir el número de parámetros necesarios para aprender el modelo. Para ello, añadimos un paso previo al algoritmo EM que reduce los parámetros mediante *clustering*.

III-A. Modelo

El modelo usa la misma estructura que GLAD, pero calculando dificultades de *clusters* de instancias en vez de instancias concretas.

- **Dificultad de cada *cluster***. Se modela usando un parámetro, $1/\beta_t \in [0, \infty)$, para cada *cluster* k, donde β_t es positivo. Si $1/\beta_t$ se acerca a 0, los ejemplos que pertenecen al *cluster* t serán más sencillos, mientras que serán más ambiguos si se acerca a ∞.
- **Experiencia del anotador**. Se modela usando un parámetro, $\alpha_j \in (-\infty, \infty)$, para cada anotador j. El significado de este parámetro es el mismo que en el caso de GLAD.

El modelo de anotación es equivalente al caso de GLAD, pero utiliza el parámetro de dificultad del *cluster*. Si denotamos como $\phi(i)$ la aplicación que mapea cada ejemplo i a uno de los *clusters*, el modelo de anotación sería el siguiente:

$$c_{ji} = p(y_i^j = y_i | \alpha_j, \beta_{\phi(i)}) = \frac{1}{1 + e^{-\alpha_j \beta_{\phi(i)}}}$$

donde y_i^j es la anotación del anotador j a la instancia i e y_i es la etiqueta verdadera de la instancia. Salvo por el uso del

cluster de dificultades, la interpretación de esta expresión es similar al caso de GLAD.

III-B. Inferencia

Usamos el enfoque EM para aprender los parámetros del modelo (conocidos los *cluster* en los que se dividen las instancias). La interpretación de estos es similar al caso del algoritmo tradicional.

- **Paso E**: Sea $\mathcal{Y}_i = \{y_i^j\}$ el conjunto de anotaciones para el ejemplo i. Para obtener la probabilidad sobre la verdadera etiqueta, y_i, conocidos los valores de $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ y las anotaciones $\mathcal{Y}_i$ podemos utilizar:

$$p(y_i = k|\mathcal{Y}_i, \alpha_j, \beta_{\phi(i)}) \propto p(y_i = k) \prod_j p_{ji}^k$$

donde

$$\begin{aligned} p_{ji}^k &= p(y_i^j = k|y_i, \alpha_j, \beta_{\phi(i)}) \\ &= \begin{cases} (c_{ji})^k \cdot (1 - c_{ji})^{1-k} & \text{si } y_i = 1 \\ (c_{ji})^{1-k} \cdot (1 - c_{ji})^k & \text{si } y_i = 0 \end{cases} \end{aligned}$$

- **Paso M**: Se maximiza la siguiente función usando gradiente descendiente con respecto a los valores de $\boldsymbol{\alpha}$ y $\boldsymbol{\beta}$

$$Q(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_i E[ln(p(y_i = k))] + \sum_{ij} E[ln(p_{ji}^k)].$$

A partir de esta expresión se obtienen los siguientes gradientes para cada parámetro

$$\frac{\partial Q}{\partial \alpha_j} = \sum_i (p_{ji}^1 \, y_i^j + p_{ji}^0 \, (1 - y_i^j) - \sigma(\alpha_j \beta_{\phi(i)})) \beta_{\phi(i)},$$

$$\frac{\partial Q}{\partial \beta_t} = \sum_i \delta(\phi(i), t) \sum_j (p_{ji}^1 \, y_i^j + p_{ji}^0 \, (1 - y_i^j) - \sigma(\alpha_j \beta_t)) \alpha_j,$$

donde $\delta(\phi(i), t) = 1$ si $\phi(i) = t$ y 0 en otro caso.

III-C. Inicialización

A diferencia del algoritmo GLAD, en la inicialización, aparte de obtener una estimación de las etiquetas verdaderas usando la clase de mayor frecuencia, debemos obtener los *clusters* para cada instancia $\phi(i)$. Para ello utilizamos un enfoque basado en factorización de matrices y K-Means.

- **Factorización de la matriz de anotacion**. Al igual que en problemas donde se pueden aplicar técnicas de filtrado colaborativo, podemos visualizar la matriz de anotaciones como una matriz donde las filas representan a los anotadores y las columnas a los ejemplos. Podemos obtener dos matrices que nos permitan estimar esta usando factorización de matrices. Esto nos proporciona vectores de tamaño R (tamaño que debe elegir el usuario del método) que describen tanto a los anotadores como a los ejemplos. Para la factorización en esta propuesta usamos ALS [9], obteniendo dos matrices, A y D, que representan a los anotadores y a las instancias respectivamente.
- **Clustering**. Una vez obtenidos los vectores, podemos aplicar cualquier algoritmo de *clustering* sobre estos para obtener agrupaciones sobre las instancias. Estas agrupaciones estarán basadas en las diferencias en los vectores encontrados en el apartado anterior, originados por patrones de anotación diferentes. Para este paso, en esta propuesta utilizamos K-Means, aunque se podrían utilizar otros métodos [6]. Como nuestro objetivo no es interpretar los *clusters*, si no simplemente utilizarlos para aliviar la complejidad de la inferencia, podemos utilizar un número de *clusters* K alto (su elección óptima dependerá del problema a resolver, en nuestro caso, para los experimentos, lo hemos fijado en 32).

Una vez completados los pasos anteriores, obtendríamos la función ϕ, con la que ya tendríamos todos los componentes para implementar el algoritmo.

III-D. Pseudocódigo

El pseudocódigo el algoritmo CGLAD se muestra en el Algoritmo 2

Algoritmo 2 Algoritmo CGLAD

1: **Inicialización**:
2: $(A, D) = ALS(X)$
3: $\phi(i) = KMeans(D)$
4: **Paso E inicial**: Agregación por mayoría $\rightarrow y_i$
5: **for** $i = 0$ hasta converger **do**
6: **Paso M**: Optimizar mediante gradiente descendiente:

$$\frac{\partial Q}{\partial \alpha_j} = \sum_i (p_{ji}^1 \, y_i^j + p_{ji}^0 \, (1 - y_i^j) - \sigma(\alpha_j \beta_{\phi(i)})) \beta_{\phi(i)},$$

$$\frac{\partial Q}{\partial \beta_t} = \sum_i \delta(\phi(i), t) \sum_j (p_{ji}^1 \, y_i^j + p_{ji}^0 \, (1 - y_i^j) - \sigma(\alpha_j \beta_t)) \alpha_j,$$

7: **Paso E**:

$$p(y_i = k|\mathcal{Y}_i, \alpha_j, \beta_i) \propto p(y_i = k) \prod_j p_{ji}^k)$$

8: **end for**

El algoritmo comienza aprendiendo los *clusters* dado el conjunto de anotaciones, pasando posteriormente al algoritmo EM para estimar los parámetros del modelo. Al igual que en GLAD, iteramos hasta que se produzca la convergencia o hasta realizar un máximo de iteraciones.

III-E. Escalabilidad

Aplicamos este algoritmo a los mismos conjuntos de datos que utilizamos para analizar los problemas de escalabilidad del algoritmo GLAD. También usamos la misma configuración de parámetros para el algoritmo de gradiente descendiente y el algoritmo EM. Obtenemos los resultados que mostramos en la Figura 3. Como se puede ver, obtenemos un modelo más estable con respecto al tamaño de los conjuntos de datos. A diferencia del modelo original, obtenemos resultados aceptables para todos los conjuntos de datos y comparables a los resultados que obtiene GLAD en los problemas pequeños. Asimismo, se logra un algoritmo más estable con respecto a la elección de la tasa de aprendizaje.

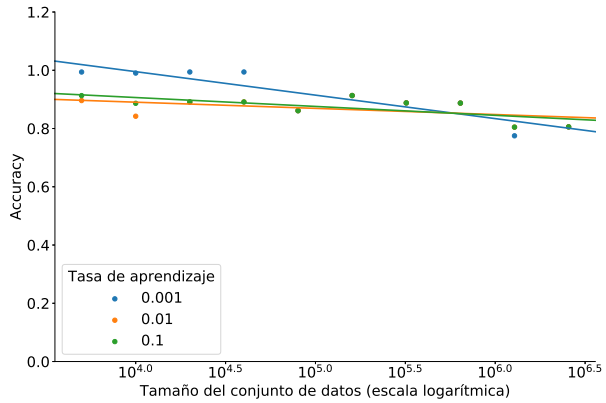

Figura 3: Escalabilidad en calidad del modelo en al propuesta CGLAD.

	Precision		F-score	
	CGLAD	GLAD	CGLAD	GLAD
Instancias				
5000	**0.9940**	**0.9940**	**0.9941**	**0.9941**
10000	0.9904	**0.9952**	0.9903	**0.9952**
20000	**0.9941**	0.9937	**0.9942**	0.9938
40000	**0.9941**	0.9937	**0.9942**	0.9938
80000	**0.8731**	0.8521	**0.8796**	0.8520
160000	**0.9132**	0.6151	**0.9191**	0.6153
320000	**0.8879**	0.7513	**0.8939**	0.7516
640000	**0.8873**	0.6613	**0.8974**	0.6566
1280000	**0.7750**	0.3077	**0.7500**	0.3086
2560000	**0.8056**	0.6423	**0.7844**	0.6418

Tabla II: Resultados GLAD y CGLAD

	CGLAD (Speedup)	GLAD
Instancias		
5000	2797.0s (1.14)	3208.0s
10000	1862.0s (2.03)	3781.0s
20000	719.0s (**5.52**)	3973.0s
40000	723.0s (**5.43**)	3925.0s

Tabla III: Resultados en tiempo GLAD y CGLAD

IV. Comparativa

Hemos realizado varias comparativas tanto entre GLAD y nuestra propuesta, CGLAD, como entre nuestra propuesta y otros algoritmos presentes en la literatura.

IV-A. CGLAD contra GLAD

En esta sección comparamos tanto GLAD y CGLAD entre sí en términos de precisión, así como en tiempo de cómputo. Para esta comparativa utilizamos los mismos conjuntos de datos que se usaron para exponer los resultados de escalabilidad en las secciones anteriores. En la Tabla II se muestran los resultados obtenidos por ambos algoritmos en estos conjuntos. Como se puede ver en la tabla, nuestra propuesta obtiene mejores resultados en la mayor parte de los conjuntos de datos, especialmente cuando los tamaños de los conjuntos de datos son mayores. Para obtener estos resultados usamos la misma configuración para ambos algoritmos, con la salvedad de que, en el caso de CGLAD, tenemos que definir el tamaño de los vectores en la factorización, 8, y el número de *clusters*, que en nuestro caso es 32. El resto de parámetros, comunes, son los siguientes:

- Maximo número de iteraciones EM: 5.
- Threshold EM: 0.1
- Máximo número de iteraciones de gradiente descendiente: 100
- Threshold gradiente descendiente: 0.1
- Tasa de aprendizaje de gradiente descendiente: 0.001

También es interesante analizar el tiempo de ejecución para los cuatro primeros casos, que presentan una precisión similar. Esto se puede ver en la Tabla III.

Como se puede ver, a pesar de que nuestra propuesta presenta una precisión similar en los casos más pequeños, esta consigue un tiempo de ejecución mucho menor. Cabe decir que usamos los primeros casos, y no los posteriores, ya que al usar CGLAD el proceso de gradiente descendiente para optimizar parámetros, para casos más grandes el algoritmo GLAD no converge, lo que lleva a que el proceso pare tras pocas iteraciones. Para estos casos, nuestra propuesta (que sí optimiza los parámetros iterando) y GLAD no se pueden comparar.

IV-B. CGLAD contra otros algoritmos del área

Existen otros algoritmos en el área del aprendizaje a partir de múltiples anotadores que podrían ser interesantes para realizar una comparativa. Sin embargo, hay que tener en cuenta que estos no presentan información de la dificultad de los ejemplos como proporcionan GLAD y CGLAD. Específicamente, vamos a comparar nuestra propuesta con MajorityVoting, el algoritmo más sencillo, que únicamente consiste en utilizar la clase más frecuente y DawidSkene [1], que utiliza como modelo una matriz de confusión para cada anotador y el algoritmo EM para realizar la estimación de esta matriz y la clase verdadera. Para estas pruebas hemos generado otros conjuntos de datos con anotadores simulados, cuyas anotaciones no solo dependen de la clase verdadera de una instancia, si no también de la dificultad de cada instancia (generada aleatoriamente). Se han generado conjuntos de los siguientes tamaños: 5000, 10000, 20000, 40000 y 80000. Para la ejecución se utilizan los siguientes parámetros:

- Maximo número de iteraciones EM: 5.
- Threshold EM: 0.1
- Máximo número de iteraciones de gradiente descendiente: 100
- Threshold gradiente descendiente: 0.1
- Tasa de aprendizaje de gradiente descendiente: 0.0003
- Tamaño de los vectores para la factorización: 8
- Tamaño de los vectores para la factorización: 32

Podemos ver los resultados en la Tabla IV. Aunque CGLAD obtiene resultados comparables e incluso mejores en varios problemas, podemos ver que no es mejor sistemáticamente que el algoritmo de DawidSkene, siendo este último mucho más rápido y sencillo. Sin embargo, CGLAD no solo nos permite modelar la precisión de los anotadores sino también la dificultad de los ejemplos, lo que puede ser interesante en

muchos problemas, incluso aunque no se obtenga una mejora en la precisión del modelo.

	Métodos		
Instancias	MajorityVoting	DawidSkene	CGLAD
5000	0.8102	0.8462	**0.8610**
10000	0.8130	0.8443	**0.8469**
20000	0.8141	**0.8478**	0.8286
40000	0.8161	**0.8494**	0.8465
80000	0.8161	**0.8494**	0.8465

Tabla IV: Resultados de CGLAD con respecto a otros algoritmos del estado del arte

V. Conclusiones y trabajo futuro

En este artículo proponemos una optimización al algoritmo GLAD para el aprendizaje a partir de múltiples anotadores donde mejoramos su escalabilidad y estabilidad a la hora de resolver problemas grandes, así como su tiempo de ejecución y, en gran parte de problemas, sus resultados. Para comprobarlo se han realizado varios experimentos que permiten comparar ambos modelos tanto en problemas con conjuntos de datos relativamente pequeños como en conjuntos de datos grandes, donde GLAD no obtiene buenos resultados. Como se ha visto en la experimentación, para este tipo de problemas nuestra propuesta funciona mejor y es más robusta ante cambios en la tasa de aprendizaje. También se ha llevado a cabo una comparativa entre nuestra propuesta y otros algoritmos del área. Se ha observado que, aunque nuestra propuesta obtiene resultados comparables a otros algoritmos del estado del arte en varios problemas, existen otros algoritmos que por su simplicidad habría que considerar a la hora de resolver un problema a partir de múltiples anotadores si el único objetivo es realizar la estimación de la verdadera etiqueta. En contrapartida, si, aparte de la precisión en la estimación de la verdadera clase, es de interés estimar la dificultad de las instancias (porque es necesaria esta información o porque en el problema que afrontamos tiene importancia la dificultad de las instancias a la hora de estimar) nuestra propuesta es una alternativa más que adecuada al algoritmo GLAD, sobre todo si queremos abordar problemas con un número de instancias relativamente elevado.

Esta mejora, así como el análisis de los problemas de este algoritmo, abren alternativas a futuras líneas de trabajo. Por un lado, en este artículo solo utilizamos algoritmos sencillos tanto para la factorización como para el *clustering*. Existen multitud de técnicas de *clustering* más potentes que podrían ser de interés para mejorar los resultados del algoritmo. Asimismo, también podría ser de interés aplicar la idea de la dificultad de los ejemplos a otros algoritmos existentes en el área pero con mejores características que GLAD en lo que respecta a la escalabilidad así como a la capacidad de abordar problemas de clase discreta o, incluso, contínua. Por último, destacar que esta propuesta solo utiliza las anotaciones para realizar el *clustering* y, posteriormente, la estimación. En problemas donde las características de cada instancia estén disponibles, podría ser interesante utilizarlas, junto con las anotaciones, para elaborar *clusters* más informados, lo que a nuestro jucio podría influir positivamente en la estimación final de la clase verdadera.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la Agencia Estatal de Investigación (AEI) y el Fondo Europeo de Desarrollo Regional (FEDER, UE) mediante los proyectos TIN2016-77902-C3-1-P y TIN2016-82013-REDT. Enrique G. Rodrigo también ha sido financiado por el MECD mediante la beca FPU15/02281.

Referencias

[1] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, 2:20–28, 1979.

[2] Jerónimo Hernández-González, Inaki Inza, and Jose A Lozano. Weak supervision and other non-standard classification problems: a taxonomy. *Pattern Recognition Letters*, 69:49–55, 2016.

[3] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198. ACM, 2014.

[4] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

[5] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043, 2009.

[6] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[7] Jing Zhang, Xindong Wu, and Victor S Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016.

[8] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.

[9] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International Conference on Algorithmic Applications in Management*, pages 337–348. Springer, 2008.