Encode What You Need

Quezada, Diego
diego.quezadac@sansano.usm.cl

20 de diciembre de 2022

Resumen

La familia de modelos de *Autoencoder* ha demostrado ser eficiente en la construcción de codificadores que permiten obtener una representación de menor dimensionalidad que capturan y codifican aquellos atributos ocultos que permiten posteriormente reconstruir la entrada. Sin embargo, estas codificaciones no consideran la tarea final en la cual estas representaciones serán utilizadas pues la reducción se aborda como un problema *self-supervised* independiente. En la presente investigación se propone **Fish**: una simple y novedosa arquitectura neuronal que permite construir un codificador *ad-hoc* a la tarea supervisada.

***Keywords*:** *Representation Learning*, *Autoencoders*, *Dimensionality Reduction*.

1. Motivación

En el contexto de aprendizaje supervisado, se aprende una **hipótesis** $h\colon \mathcal{X} \to \mathcal{Y}$ utilizando un conjunto $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M}$ de datos entrenamiento. Utilizar una representación de menor dimensionalidad ***ad-hoc*** nos debería guiar hacia una mejor hipótesis h que si se usara una representación de menor dimensionalidad estándar como las obtenidas mediante un *Autoencoder Undercomplete*.

2. Estado del arte

La **reducción de dimensionalidad** es un proceso que proyecta datos de alta dimensión en espacio dimensional mucho más pequeño [Sarveniazi, 2014]. Mediante algoritmos clásicos tales como *PCA*,*LDA* o *K-Means* se puede realizar esta tarea obtiendo incluso resultados del estado del arte [Coates y Ng, 2012]. Desde la introducción del pre-entrenamiento no supervisado, y considerando el interés actual por el aprendizaje profundo, muchos esquemas en donde se apilan capas de atributos para construir representaciones profundas han sido propuestos: *sparse-coding*, *RBMs*, *sparse RBMs* y *Autoencoders* por ejemplo [Coates *et al*., 2011].

El objetivo del **aprendizaje de representaciones** (*representation learning*) es aprender representaciones de los datos que permitan extraer información útil a partir de ellas [Bengio *et al*., 2012]. En este sentido, se define una buena representación como aquella que es útil cuando se utiliza como entrada para un modelo supervisado.

En la comunidad del aprendizaje de representaciones se han desarollado dos amplias líneas paralelas de investigación: una basada en modelos gráficos probabilísticos y la otra en redes neuronales artificiales, siendo representadas respectivamente por las Máquinas de *Boltzmann* restringidas y los *Autoencoders* junto a todas sus variantes.

Hasta la fecha y según el conocimiento del autor, todos los algoritmos propuestos para aprender representaciones son no supervisados, o bien, auto supervisados y, por ende, no realizan una codificación *ad-hoc* a un problema supervisado.

3. Descripción

3.1. Introducción

En la presente investigación, se compara el desempeño de dos algoritmos de reducción de dimensionalidad para resolver un problema de clasificación.

Algoritmo 1: Entrenar **por separado** un *Autoencoder Undercomplete* $(g \circ f_1)(x)$ y una red neuronal $(h_1 \circ f_1)(\boldsymbol{x})$ totalmente conectada que cuyas primeras capas están definidas por el codificador $f_1(\boldsymbol{x})$ con sus pesos congelados.

Algoritmo 2: Entrenar **simultaneamente** una red neuronal $(h_2 \circ f_2)(\boldsymbol{x})$ totalmente conectada cuyas primeras capas aprendan una representación de menor dimensionalidad.

Problema de clasificación: Clasificar *tweets* como tóxicos o no tóxicos.

En 1 se presenta una arquitectura estándar de un *Autoencoder Undercomplete*. Consiste en una red neuronal artificial totalmente conectada que es entrenada para codificar la entrada **X** a una representación de menor dimensionalidad mediante el codificador f_1 y luego reconstruir **X** minimizando una función de pérdida $\ell(\boldsymbol{x}, g(f_1(\boldsymbol{x})))$ como la pérdida cuadrática que penaliza $g(f_1(\boldsymbol{x}))$ en función de cuánto difere con $\boldsymbol{x}$.

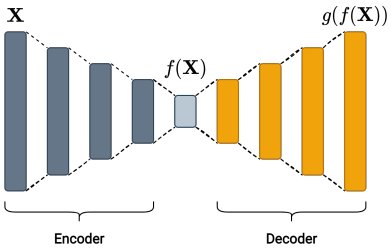

Figura 1: *Autoencoder Undercomplete*

Como en el contexto de aprendizaje supervisado la tarea es conseguir una hipótesis h_1 que permita *mapear* desde un conjunto de entrada $\mathcal{X}$ hacia un conjunto de salida $\mathcal{Y}$, una vez entrenado el *Autoencoder* $(g \circ f_1)(x)$ se utiliza el codificador f_1 para codificar las entradas a un nuevo modelo h_1.

En 2 se presenta la arquitectura neuronal **Fish** propuesta. Consiste en una clásica red neuronal artificial totalmente conectada que es entrenada para aprender **simultaneamente** la codificación *ad-hoc* f_2 y la tarea supervisada $h_2 \colon \mathcal{X} \to \mathcal{Y}$. Primero, sus primeras capas se encogen para lograr una representación de menor dimensionalidad; Segundo, una capa expande esta representación; Tercero, las siguientes capas se encogen hasta la capa de salida.

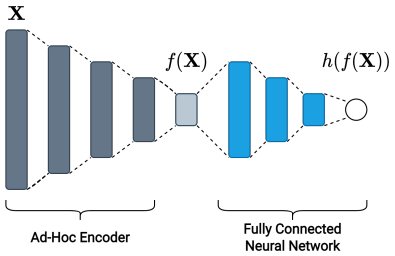

Figura 2: Red Neuronal Fish

3.2. Datos

El conjunto de datos utilizado consiste de 312735 *tweets* con las siguientes etiquetas *toxic*, *severe toxic*, *obscene*, *threat*, *insult* y *identity hate*. En la presente investigación, solo se utilizará la etiqueta ***toxic***, la cual indica simplemente si un *tweet* es tóxico o no.

Los datos fueron publicados para una competencia de *Kaggle* [cjadams, 2017] organizada por *Jigsaw*: "una unidad de Google que explora las amenazas a las sociedades abiertas y crea tecnología que inspira soluciones escalables".

Un ejemplo de un *tweet* tóxico:

have a cupcake since you have no friends or life award wow you must live a very sad loney life i guess if i had no friend or anyone that cared about me i would spend all my

time pretending like a know it all with a stick in my butt as i ride my high horse so here is a cupcake you fruitloop diggus go ban me and change another article to whatever you what it to say so you feel like you have some power in the world cause god knows it must suck to be you stop trying to pick up little boys on the internet i m telling chris hanson

Un ejemplo de un *tweet* no tóxico:

sorry i confused you with another user greetings i just wanted to drop a note and tell you i was sorry i confused you with another user over at the rfa talk page your usernames are close and i misread the username

3.3. Preprocesamiento

A continuación, se detalla el paso a paso del proceso de preprocesamiento:

1. Concatenar los conjunto de datos crudos de entrenamiento y prueba.
2. Transformar a minúscula cada *tweet*.
3. *Tokenizar* cada *tweet* utilizando ***Spacy*** [Honnibal *et al.*, 2020].
4. Definir los nuevos conjuntos de entrenamiento, validación y prueba utilizando el $80\,\%$, $10\,\%$ y $10\,\%$ de los datos respectivamente.

Solo el $10\,\%$ de los *tweets* en el conjunto de datos obtenido en 2 son tóxicos. En el punto 4 del preprocesamiento los conjuntos de datos son creados manteniendo esta proporción; el conjunto de entrenamiento consta de 178839 *tweets*, mientras que el conjunto de validación y prueba de 22355.

Para *tokenizar* cada *tweet* se utilizó el *tokenizador* de la *pipeline* ***en_core_web_sm*** de ***Spacy***. En el conjunto de entrenamiento el promedio de *tokens* por *tweet* es 64. La siguiente figura presenta la distribución de *tweets* en el mismo conjunto de datos.

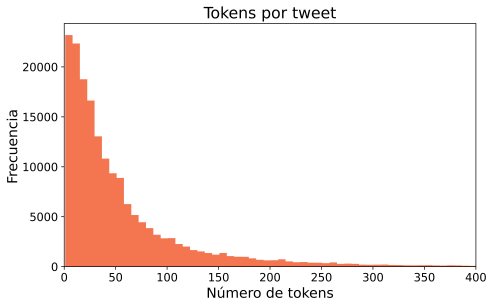

Figura 3: Distribución del número de *tokens* por *tweet*

3.4. Desbalance de clases

Tal como se mencionó en la sección anterior, solo el $10\,\%$ de los *tweets* son tóxicos. Es necesario manejar este desbalance de alguna forma para que en el proceso de entrenamiento los modelos aprendan a diferenciar entre *tweets* tóxicos y no tóxicos. En esta ocasión, se optó por balancear cada *batch* utilizando el atributo *sampler* de la clase *DataLoader* de **PyTorch** [Paszke *et al.*, 2019].

3.5. Arquitectura

Para que las comparaciones entre arquitecturas fueran justas, se consideraron redes neuronales artificiales con el mismo número de capas. f_1 y f_2 codifican la entrada utilizando una capa de *embeddings* seguida de 4 capas con función de activación *ReLU* [Agarap, 2018]. Luego, h_1 y h_2 generan la salida utilizando 3 capas con función de activación *ReLU* seguida de una capa con función de activación *Softmax*.

Los hiperparámetros de interés para la presente investigación son el tamaño del *embedding* y el número de neuronas en la quinta capa, es decir, aquella que **expande** la codificación. Se experimentó con tres valores distintos para cada hiperparámetro, lo que genera 9 modelos distintos para cada arquitectura.

3.6. Entrenamiento

Para la primera arquitectura, se entrenaron tres *Autoencoders Undercomplete* con *embedings* de tamaño 128, 256 y 512 utilizando la función de pérdida cuadrática para penalizar las salidas. Luego, para cada *Autoencoder* se utilizó el codificador con los pesos congelados para entrenar tres redes neuronales artificiales con un diferente número de neuronas en la capa de **expansión** utilizando la función de pérdida *Cross Entropy*. Para ambos entrenamientos se utilizó el optimizador *SGD* con una taza de aprendizaje que **decae** en un ratio de $0{,}1$ cada 20 epochs.

De manera análoga, para la segunda arquitectura se entrenaron 9 redes neuronales artificiales; una por cada combinación de hiperparámetros. Para mantener las comparaciones justas, se utilizó el mismo optimizador y decaimento de la taza de aprendizaje.

Para ambas arquitecturas se consideró un entrenamiento de 60 *epochs*. Al realizar más iteraciones el *trade-off* entre un elevado tiempo de computación y una despreciable mejora en la generalización del modelo no era favorable. Además, ambos modelos comenzaban a sufrir *overfitting* a partir de la *epoch* 80 aproximadamente.

4. Resultados

En 1 se detallan los resultados obtenidos para ambas arquitecturas en el conjunto de prueba. Solo para tres configuraciones, la arquitectura que utiliza un *Autoencoder* tiene un mejor *F1-Score* que la arquitectura propuesta *Fish*.

El uso de *F1 Score* es necesario pues, tal como se mencionó en la sección 2.3, el conjunto de datos de prueba mantiene la proporción del $10\,\%$ de *tweets* tóxicos del conjunto de datos original. Mientras que métricas como el *accuracy* ocultan la verdadera capacidad de generalización de modelos cuando se miden en un conjunto de datos desbalanceado, *F1 Score* informa una media armónica entre la *precision* y el *recall*: dos métricas que miden con mayor robustez el desempeño en este tipo de conjuntos de datos.

Embedding	Neuronas	F1 Score FNN	F1 Score Fish
128	48	0.723873	0.726558
128	64	0.734102	0.720681
128	88	0.707602	0.715352
256	64	0.720902	0.717698
256	128	0.719521	0.725211
256	192	0.720952	0.714286
512	64	0.669626	0.724931
512	128	0.709508	0.722067
512	256	0.717080	0.720137

Tabla 1: *F1-score* en el conjunto de prueba para ambas arquitecturas

5. Innovación

En la presente investigación se innova con la propuesta de *Fish*: una arquitectura novedosa que permite obtener una representación de menor dimensionalidad de los datos de entrada mediante un codificador *ad-hoc* a un problema supevisado .

6. Conclusión

En los experimentos realizados, **Fish** mediante sus representaciones de menor dimensionalidad *ad-hoc* al problema supervisado supera, en general, al rendimiento de un modelo compuesto por un codificador de un *Autoencoder* seguido de una red neuronal artificial totalmente conectada. Este resultado es **clave**, pues valida la motivación inicial de esta investigación al informar que las codificaciones obtenidas mediante un *Autoencoder* no necesariamente capturan los mejores atributos ocultos para, posteriormente, resolver un problema supervisado.

La arquitectura **Fish** requiere considerablemente un me-

nor tiempo de computación para su entrenamiento sin sacrificar su rendimiento. Esto es de esperarse pues el codificador f_2 es aprendido de manera simultánea con la hipótesis h_2 que aprende a resolver el problema supervisado.

El siguiente paso para esta investigación es estudiar la calidad de la representaciones y analizar su capacidad para ser transferidas a problemas supervisados similares.

El código fuente que permite reproducir los resultados de esta investigación se encuentran disponible en *https://github.com/diegoquezadac/Fish*.

Referencias

[Agarap, 2018] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375.

[Bengio *et al.*, 2012] Bengio, Y., Courville, A. C., y Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538.

[cjadams, 2017] cjadams, Jeffrey Sorensen, J. E. L. D. M. M. n. W. C. (2017). Toxic comment classification challenge.

[Coates *et al.*, 2011] Coates, A., Ng, A., y Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223.

[Coates y Ng, 2012] Coates, A. y Ng, A. Y. (2012). *Learning Feature Representations with K-Means*, pp. 561–580. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Honnibal *et al.*, 2020] Honnibal, M., Montani, I., Van Landeghem, S., y Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

[Paszke *et al.*, 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., y Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703.

[Sarveniazi, 2014] Sarveniazi, A. (2014). An actual survey of dimensionality reduction. *American Journal of Computational Mathematics*, 04:55–72.