Solución del Netflix Prize usando la Máquina de Boltzmann Restringida

J. Agustín Barrachina
Miembro Estudiantil IEEE
Instituto Tecnológico de Buenos Aires

***Abstract*—Trabajo basado en el desafío Netflix Prize que consistió en mejorar el sistema de recomendaciones de películas disponibles. Se obtiene un *dataset* de 80 mil puntuaciones de mil usuarios para setecientos películas. Luego se utiliza un *testset* que tiene 20 mil puntuaciones a predecir. El objetivo es encontrar el algoritmo que reduzca el MSE en el *testset*. Para el mismo se aplicó una Máquina de Boltzmann Restingida.**

I. Introducción

El desafío *Netflizx Prize* consistió en mejorar el sistema de recomendaciones de películas disponibles. La empresa proveyó una base de datos de 100 millones de puntuaciones que 480 mil usuarios le dieron a 18 mil películas. El objetivo es entrenar un predictor y minimizar el error cuadrático para 1,4 millones de puntuaciones que los participantes desconocían.

En el caso particular de éste informe se posee un set de datos reducido de menos de 1000 usuarios con menos de 2000 películas (un total de 80.000 puntuaciones).

El trabajo se basó principalmente en el trabajo de Salakhutdinov et al. [1] por lo que se recomienda su consulta ante cualquier inquietud. En el mismo se aplica el algoritmo de la Máquina de Boltzmann Restringido (RBM por sus siglas en inglés) para solucionar el desafío.

En el informe se considerará pleno conocimiento sobre la Máquina de Boltzmann convencional y su implementación básica. Para su profundización se puede usar [2] capítulo 7.

II. Implementación Teórica

Definiremos primero las variables utilizadas, notación que se mantendrá a lo largo del informe.

- **M** Cantidad de Películas
- **N** Cantidad de Usuarios
- **K** Puntuación máxima (entre 1 y K). Para nuestro caso se usará $K = 5$.
- **F** Cantidad de categorías.
- **V** matriz en la que cada fila será una película diferente y cada columna representará la puntuación, es decir, $v_i^k = 1$ si el ranking de la película i es k y cero para otro caso.
- $\mathbf{h}_j$ coeficiente de cada valor escondido (hidden value) con $j = 1, ..., F$ valores.

El mayor problema de cómo aplicar Boltzmann es cómo lidiar con los datos eficientemente. Para cada uno de los N usuarios habrá $m < M$ películas con puntuación, ya que no estarán definidas todas las puntuaciones M para cada uno. Tal como propone [1], debido al caso específico que cumple dicha condición ($m < M$), se utilizó una RBM distinta para cada usuario. Luego se obtuvo la máquina de de Boltzmann final realizando un promedio de los coeficientes de cada una de ellas.

III. RBM

Para la implementación de la máquina de Boltzmann se basó el trabajo sobre el programa realizado por E. Chen [3].

Si bien en la siguiente sección se realizará alguna descripción sobre las ecuaciones y el método, el mismo se encuentra reducido y sin demostración de las ecuaciones. Se recomienda leer el trabajo de Hinton [4] para más desarrollo sobre el tema. En particular, esta sección se encuentra desarrollado en la sección 3 de dicho trabajo.

A. *Asociaciones Positivas*

Las asociaciones positivas con aquellas con las cuales se parte de los coeficientes *visibles* y se llega a los coeficientes *ocultos*.

Para el entrenamiento de la red se define la *energía de activación* de los *estados ocultos* a_j como:

$$a_j = \sum_{i=1}^{m} \sum_{k=1}^{K} v_i^k W_{ij}^k \tag{1}$$

Donde W_{ij}^k son los pesos que unen el valor de la puntuación k de la película i con el peso escondido j. En la figura1 se muestra un diagrama de cada índice de la película 1 hasta m para sus $K = 5$ valores de puntuación. Para prolijidad se encuentran dibujadas solo las lineas correspondientes a W_{i1}^k, es decir, solo para el coeficiente escondido $j = 1$.

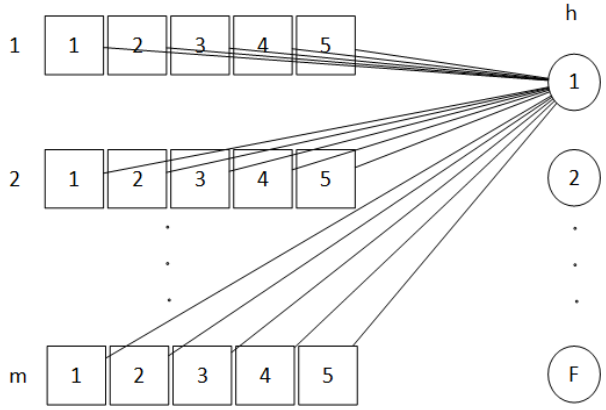

Fig. 1. Diagrama RBM implementada

Luego se debe computar la denominada *probabilidad de los estados ocultos*:

$$p(h_j = 1|V) = \sigma\left(b_j + a_j\right) \tag{2}$$

Donde b_i es el bias y $\sigma(x)$ es la función logística definida por:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

Se computa para las probabilidades de la ecuación 2 los F valores de estado ocultos como 1 o 0 según la probabilidad en cuestión (que denominaremos h_j). Queda entonces definidos los coeficientes de los estados ocultos (ecuación 4)

$$h_j = \begin{cases} 1, & p(h_j = 1|V) \\ 0, & 1 - p(h_j = 1|V) \end{cases} \tag{4}$$

Se llegó entonces a los coeficientes de los estados ocultos a partir de los valores visibles. En la siguiente sección se explicará el camino inverso, es decir, pasar de los estados ocultos a partir de los estados visibles.

B. Asociaciones Negativas

Se realizó hasta ahora el calculo de los valores denominados *positivos*. Es necesario ahora hacer el cálculo de los valores *negativos*. Para ello se calcula las energías de activación de los valores visibles:

$$a_i^k = \sum_j^F h_j W_{ij}^k \tag{5}$$

Ésta energía se puede considerar como la inversa a la calculada en la ecuación 1. De forma similar, se calcula el "inverso" de la ecuación 2 se define la siguiente probabilidad:

$$p(v_i^k = 1|h) = \frac{e^{\left(b_i^k + \sum_{j=1}^F h_j W_{ij}^k\right)}}{\sum_{l=1}^K e^{\left(b_i^l + \sum_{j=1}^F h_j W_{ij}^l\right)}} \tag{6}$$

Se obtiene a partir de estos datos, una matriz que representa la estimación de la matriz V la cual denominaremos $\hat{V}$.

C. Entrenamiento de la Red

A partir de los datos h_j obtenidos en la sección III-A se puede calcular las *asociaciones positivas* como:

$$positive_{ij}^k = v_i^k * h_j \tag{7}$$

De forma análoga, con la matriz $\hat{V}$ obtenida en la sección III-B, se vuelven a obtener los valores h_j (que denominaremos $\hat{h}_j$) tal como se hizo en la sección III-A. A partir de los mismos se obtienen los valores asociativos *negativos* de forma análoga a la ecuación 7:

$$negative_{ij}^k = \hat{v}_i^k * \hat{h}_j \tag{8}$$

Luego se actualizan los coeficientes de la forma:

$$\triangle W_{ij}^k = W_{ij}^k + L * (postive - negative) \tag{9}$$

Donde L es el denominado "Learning Rate". Nótese que como cada red tendrá un solo entrenamiento, se omite el factor de división por cada caso de entrenamiento en la ecuación 9.

El algoritmo descrito se repite para cada usuario y luego se obtiene una máquina de Boltzmann final promediando los coeficientes de cada usuario.

D. Valores Iniciales

Como *Netflix* utiliza 23 categorías de películas para su motor de búsqueda, se decidió comenzar con dicho valor de F, el cual puede ser un mínimo para utilizar.

Los pesos iniciales de los coeficientes W_{ij}^k suelen ser aleatorios de media cero con distribución Gaussiana y desvío estándar de alrededor 0.01. ([4] sección 8).

En la misma sección recomienda para el bias de las unidades visibles, usar $log\left[p_i/(1-p_i)\right]$ donde p_i es la proporción de vectores de entrenamiento sobre la cual el vector i se encuentra. En nuestro caso sin embargo, ese número es muy cercano a cero ya que se utilizan casi 1000 usuarios, por lo cual se iniciará el bias con cero. Algo similar ocurre para el caso del bias de las unidades escondidas. Sin embargo, existe la posibilidad de elegir ese numero en -4 si se desea mayor dispersión.

El parámetro L (Learning rate) es muy difícil de establecer. Un valor muy grande hará que el error se incremente drásticamente y los pesos diverjan. Se recomienda comparar los mismos con el histograma de la variación de los pesos y con los pesos mismos y tratar que la variación sea al rededor de 10^{-3} veces los pesos. Sin embargo en éste trabajo se usará el valor por defecto que presenta la función realizada por [3] y se variará luego acorde a los resultados.

La cantidad de iteraciones que se realizan con cada entrenamiento (*epochs*) se desea que sea lo más alta posible. Sin embargo, como el algoritmo tiene un tiempo de computación muy alto, el mismo se redujo mucho para poder correr el algoritmo en un tiempo razonable.

IV. Posibles Mejoras

A. Reducción del MSE

El trabajo [1] menciona numerosas otras variantes de RBM como el RBM con unidades ocultas gausianas, RBM condicional y RBM concional factorizado. Todos estos casos prueban tener mejores resultados que para el caso RBM convencional aplicado. Sería de interés aplicar estos métodos para mejorar el rendimiento del código.

B. Tiempo de computación

Por otra parte, el código tarda mucho tiempo con cada vez que se ejecuta. Las implementaciones de las sumatorias se utilizaron con ciclos 'for' sin tener en cuenta la localidad, se podrían dividir estos mismos de una forma coherente para así tenerla en cuenta y reducir el tiempo de ejecución. Otra opción sería implementarlas como producto matricial.

Ésta implementación sería una buena ocasión para aplicar *CUDA*, de forma de utilizar la placa de video para realizar

todas las multiplicaciones matriciales tan grandes y los ciclos 'for'.

Existe una librería llamada '*gnumpy*' [5], la cual aplica cuda para recrear una librería similar a '*numpy*'. De hecho, el mismo autor de la libraría presenta como un ejemplo de funcionamiento la aplicación de una red RBM.

V. Resultados

Se usaron dos métodos para obtener el resultado, es decir, la predicción del puntaje del usuario. En un principio se calculaba los valores v_i^k obteniendo los valores ocultos como se hace en III-A para obtener los h_j y a partir de los mismos, volver a obtener los $\hat{V}$.

Se utilizó luego un promedio de las probabilidades de la ecuación 6 con sus referentes pesos (entre 1 y K para cada caso) en lugar de obtener los valores V directamente a partir de la misma. Es decir, se predijo el puntaje de la película i de acuerdo a la siguiente ecuación.

$$prediction_i = \sum_{l=1}^{K} l * p(v_i^l = 1|h) \tag{10}$$

Lo cual tiene más sentido físico teórico porque no usa aleatoriedad sino que toma en cuenta el resultado de las probabilidades con su peso. Ya que la suma de probabilidades es 1, el valor de predicción de la ecuación 10 está acotado entre 1 y 5 lo cual también en lo que se espera.

El algoritmo no se comportó como se esperaba. El algoritmo tendía aproximadamente a 3 sin importar el caso. Se notó que variando el valor de L a valores muy pequeos respecto a los coeficientes W_{ij}^k, los valores obtenidos eran siempre 3.0 $\pm$ 0.09. La varianza aumentaba al subir L pero no se alcanzó una variación superior a 1.

Es posible que el error se encuentre en el bias utilizado tanto en valores iniciales como cuando se lo tuvo que aplicar en código. Se prestó poca atención al mismo y puede haber un error en éste aspecto.

El bias además es el encargado de separar aún más los valores de probabilidad de cada coeficiente, lo cual tiene también un sentido teórico que el error se encuentre allí.

A. Cómo seguir

Para debugear el código se recomienda reducir el set de datos a un número menor para reducir el tiempo de computación (que con todos los datos y un *epoch* = 1 el mismo ronda los 25 minutos).

Luego se recomienda revisar el código con principal atención al bias. Pero también contrastar mejor cada función con la teoría explicada en este informe ya que el código se hizo muy rápido y puede contener errores.

Se puede verificar que con cada entrenamiento, la predicción devuelva ese mismo valor o cercano antes de utilizar el set de datos.

VI. Conclusión

Fue decepcionante no obtener un buen resultado y no alcanzar una predicción aceptable.

El trabajo fue muy interesante como aplicación de la teoría vista en la materia a un problema real. El mismo permitió generalizar la máquina de Boltzmann y lograr comprender más su funcionamiento así como la aplicación de un ejemplo más complejo que los encontrados en las bibliografías convencionales.

El código presenta muchas posibilidades de mejora en el futuro como se explica en la sección IV y es de gran interés poder aplicarlo en algún futuro.

Acknowledgment

Se agradece a Nicolás Sutton, quien recomendó el uso del trabajo [1] para el desarrollo del trabajo, lo cual fue el pilar del mismo.

REFERENCES

[1] A. M. . G. H. Ruslan Salakhutdinov, "Restricted boltzmann machines for collaborative filtering," University of Toronto, 6 King's College Rd., Toronto, Ontario M5S 3G4, Canada, Tech. Rep., 2010.

[2] A. K. . R. G. P. John Hertz, *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, 1990, vol. 1.

[3] E. Chen. (2015, jul) Github repository restricted-boltzmann-machines. [Online]. Available: https://github.com/echen/restricted-boltzmann-machines

[4] G. Hinton, "A practical guide to trining restricted boltzmann machines," Department of Computer Science, University of Toronto, 6 King's College Rd, Toronto, Tech. Rep. 1, aug 2010.

[5] T. Tieleman, "Gnumpy: an easy way to use gpu boards in python," Department of Computer Science, University of Toronto, 6 Kings College Rd, Toronto, M5S 3G4, Canada, Tech. Rep., jul 2010.