Chapter 14 : Part 2

**Deep Learning**

**BM: Restricted Boltzmann Machines (RBM)**

Introduction

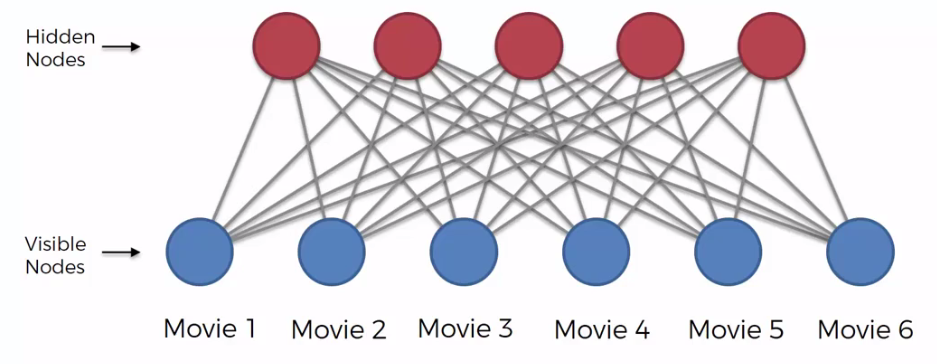
**14.2.1 Restricted Boltzmann Machines (RBM)**

And we're going to see how Restricted Boltzmann Machines (RBM) learns, and how it is applied in practice.

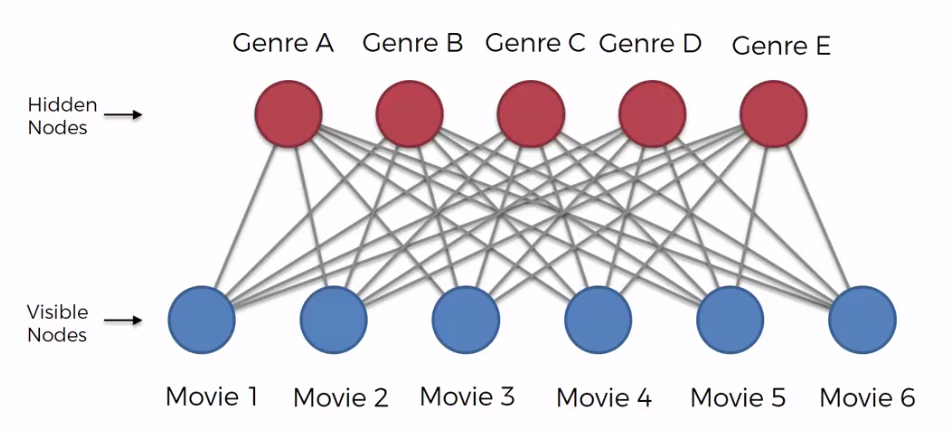
|  |  |  |
| --- | --- | --- |
| * BM: Here we've got the *standard Boltzmann machine*, we've got all of these intra connections. Every *single* *node* connects to *each* *other*. * In *theory* this is a great model and you can *solve* lots of different *problems*. But in *practice* it's very *hard to implement*. We simply cannot compute a full Boltzmann machine, the reason is as you *increase number of nodes*, the number of *connections* between them *grows exponentially*. * RBM: Therefore, a different type of architecture was proposed which is called the *Restricted Boltzmann Machine (Rbm)*. | |  |
| * Here in RBM, we've got exactly the same concept with the simple restriction that * Hidden nodes *cannot* *connect* to each other and * Visible nodes *cannot* *connect* to each other. * Everything else is the same to BM. We've also got undirected connections. |  | |

**14.2.2 How a RBM trained**

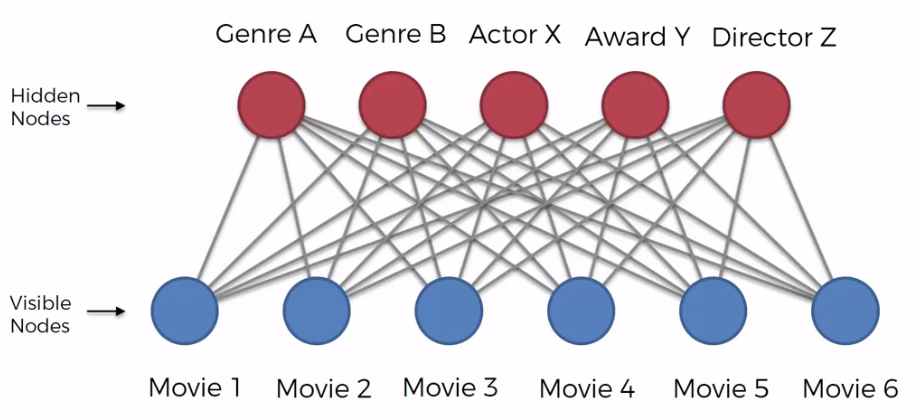
Now we're going to talk about how a *RBM* *works*, how it's *trained* and then how it's *applied* in practice. For example, here we use RBM for movie recommendation system.



* Let's say our *RBM - recommender system* is going to be working on six movies.
* As you remember, a *Boltzmann machine* is a *generative* *type* of *model*, it always constantly generates states. By training the BM through feeding it training data and through a process called contrastive divergence (it will be discussed next), the Boltzmann machine become a representation of our specific system *(rather being a recommender system for any kind of possible impossible movies)*.
* We make it the *recommender* *system* that is *associated* with our *specific set of movies* that we are *feeding* into this system and with our *specific* *training* *data*.
* Through that process, *RBM* is going to *learn* how to *allocate* its *hidden* *nodes* to certain *features*.



* This process is very similar that we discussed in the CNN (for processing images). For example, through the training process, the RBM might identify some genres A, B, C, D and E.
* But the important thing to understand here is that RBM doesn't know about those genres. It's just identifying *certain* *features*. Actually it ***doesn't have to be genres***, for example, it could identify that *genre A* and *B* are important for the *recommender* *system* but there can be other important features such as an actor, an award or a director.
* Hence those genres could be *Genre of certain class* A or B (eg: Action or Drama) or *Genre of Actors* (Tim Robbins or Morgan Freeman), *Genre of Awards* (Oscars or Golden globe), *Genre of Directors* (Tarantino or Cameron).



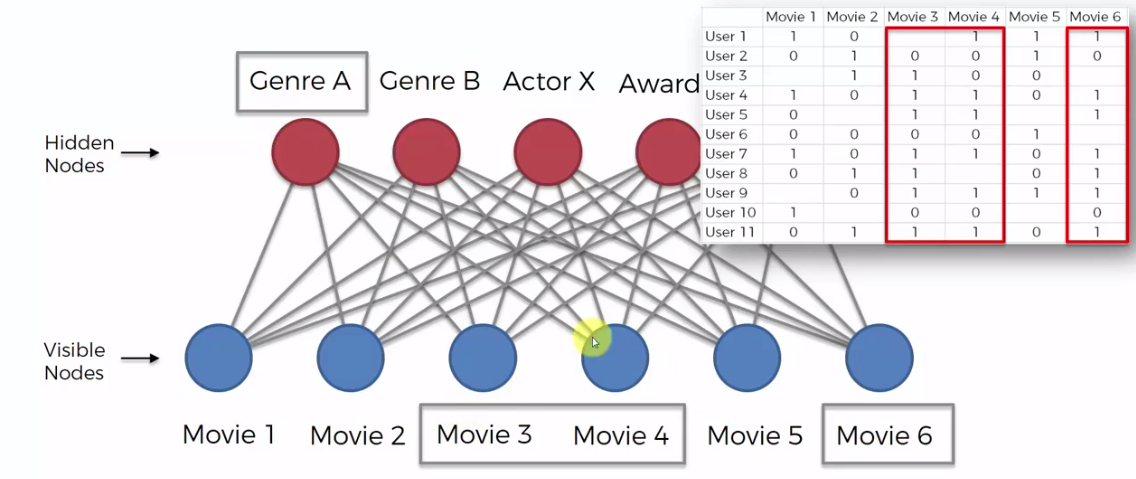
|  |  |
| --- | --- |
| * How RBM identify an important feature: During the training process, we're feeding in lots and lots of rows to the RBM and for example, these rows could look something like ***Table in Right-hand***, where we've got *movies* as *columns* and then the *users* as *rows*. * We've got the ratings ***1 = user liked it***, and ***0 = user didn't like it***. * The empty cells means that person hasn't watched that movie. * Through this process as we're feeding this data to this RBM, it's able to understand better our system. |  |

* And *adjust* *itself* to be a better *representation* of our *system*, and *understand* and *reflect* all of the *intra* *connectivity* that might be present in the data.
* Because ultimately, people have *biases*, *preferences*, *tastes* and that is what is *reflected* in the *data*.
* If somebody liked *Movie-2* and *Movie-3* and didn't like *Movie-1* just means that that's what's their preferences.
* Somebody else might have liked *Movie-1* and might have not liked *Movie-2* but liked *Movie-3*.
* So basically the data is talking about the preferences of people, their tastes and how they're biased towards different movies and that's what the RBM is trying to find out.

|  |  |
| --- | --- |
| * *RBM* would *identify* those in the *training* and it would assign a *node* to look out for *certain* *feature* (even *without* *knowing* what that *feature* is, since all the input are **1**'s and **0**'s ). * It's not getting the genre of the movies or list of actors or list of awards, it's only getting just these **1**'s and **0**'s. * From that kind of data it can *establish* that there probably is *some* *feature* that these *movies* have in *common* that is making people *like* them. |  |

* So people who *like* these *movies*, actually they *like* *that* *feature* and therefore any *other* *movie* with that *feature*, is highly likely to be *enjoyed* by those specific *people*.

In our understanding, as humans that feature might be Genre, specific Actors or Award winning or some specific Directors. But RBM doesn't aware of those things.



* In short ward RBM takes those *input*, and through the *training* *process* it understands what *features* among these *movies* and it's assigning its *hidden* *nodes* or the *weights* are being *assigned* in such a way that the *hidden* *nodes* are becoming *reflective* of those specific *features*. So that's how the training of the RBM happens.

**14.2.3 Trained RBM in action**

Consider we've trained up our RBM, it is able to pick out these *certain* *features* and based on data of *thousands* of users and their *ratings*.

* Now we're going to look at specific features. Let's say as ***features***, we are considering ***Drama*** or ***Action***, ***Leonardo*** ***DiCaprio*** as the actor, ***Oscar*** as an award (whether or not the movie has won an ***Oscar*** for the ***Best*** ***Picture***), and ***Quentin*** ***Tarantino*** as director.
* These *named-features* are just for our *learning* *purpose*. In reality, the *RBM* has no idea about those *names*, *Genres*, *Actors* or *Directors*. It's just *picking* out a *feature*.
* Let's look at a couple of movies. We're going to input a new row into this RBM-recommender system and we're going to see how it *predict* whether a *person* will *like* certain movies or *not*
* We've got movies ***The Matrix***, the ***Fight Club***, ***Forrest Gump***, ***Pulp Fiction***, ***Titanic*** and ***The Departed***.

|  |  |
| --- | --- |
| * Now the person that we're trying to make a recommendation for gives following rating : * The Matrix: Seen & Didn't liked it (**0**), * Fight Club: didn't Seen the Fight Club. * Forrest Gump: Seen & liked it (**1**) * Pulp Fiction: Seen & Didn't liked it (**0**) * Titanic: Seen & liked it (**0**) * The Departed: they haven't seen that movie |  |

* Now since the user haven't seen *Fight Club* and *The Departed* our *RBM* will *predict* that the user will *like* those movies or *not*, so that we can *recommend* movies to that *user*.
* We're gonna go through this *step by step* to see how *RBM* takes those decisions. We're going to assess which of these nodes (features) are going to activate for this specific user.
* As in the *CNN* *analogy*, there, we would feed in a *picture* into our *CNN* and it would, *certain* *features* would *highlight*. Certain features would *light* up if they're *present* in that *picture*.
* Same thing here we're *feeding* in a *row* into our *RBM* and certain *features* are going to *light* *up* if they are *present* in this user's *tastes* and *preferences* and *likes* and *biases*.

|  |  |
| --- | --- |
| * Drama: *Forrest* *Gump*, *Titanic* and *The Departed* are *Drama*. * (We don't have rating-data for The Departed, RBM can only learn from other two.) * Since this person liked Forest Gump and Titanic and based on that this node is gonna light up Green. * Symbolically green means node is activated and that means this person *likes Drama* *movies*. |  |
| * Action: The Action movies we have here are *The* *Matrix*, *Fight* *Club* and *Pulp* *Fiction* and *Departed* (it is also Drama). * We have four Action movies but out of them we only have *rating-data* for *The Matrix* and *Pulp* *Fiction* and both of these, this person *didn't like*. So it's gonna *light* up in *Red*. |  |
| * Dicaprio: *Leonardo DiCaprio* is present in *Titanic* and *The* *Departed*. * We only have *rating-data* for *Titanic* and user liked it. So the *DiCaprio* *node* is going to light up green. |  |
| * Oscar: Here we've got three Oscar movies. We only have rating-data for *Forrest* *Gump* and *Titanic*. The person liked both. The Oscar-node is gonna just light up green. |  |
| * Tarantino: The only Tarantino movie we have here is Pulp Fiction. The person *did not like* it. Therefore this Tarantino-node is gonna *light* up *Red*. |  |

* Now that's the first pass (forward-pass). Everything from our *visible* *nodes* goes into our *hidden* *nodes* and now we know which ones of our *hidden* *nodes* are *activated*.
* Backward Pass: When the backward pass happens, the Boltzmann machine try to *reconstruct* our *input*. It happens during training as well. So during training the test is also happening.
* *BM* first accept values into the *hidden* *nodes* and then it tries to *reconstruct* your *inputs* based on those *hidden* *nodes*. If during *training* the *reconstruction* is *incorrect*, then everything is adjusted (***weights*** are ***adjusted***) and then reconstruction happens again.
* Since we working with trained RBM, we're actually *inputting* a certain row and we want to get our predictions. So basically, there is not gonna be any *adjusting* of *weights*. We're just going to see how the BM basically *reconstructs* these *rows*.
* We're not going to care about the *movies* that we *already* have *ratings*, that's the training part of BM.
* Here we're only going to care about the movies that don't have ratings, we're gonna use RBM to reconstructs the ratings as predictions.

|  |  |
| --- | --- |
| * Fight Club: Fight Club is going to look at all of the *hidden-nodes* and based on training it's going to find out which *nodes* actually connect to *Fight Club*. * It's not a Drama movie. * It's an Action movie. * It *doesn't have DiCaprio* in it. * This movie *hasn't win* an *Oscar*. * *Tarantino* is *not* the *director* of this movie. |  |
| * Hence from all 5 hidden-nodes it only connects to Action (node). But this node is lit Red (not active). * *RBM* recognize these *associated* *connections*, based on the *weights* that it had determined *during* *training*. * Based on Action's (node) connection, we know this one lit up in Red and therefore *Fight Club* is going to be a movie that this *person* is *not* *going* to *like*. The predicted rating will be **0**. | |
| * The Departed: * It's a Drama movie. Connected to this node (active). * It's an Action movie. Connected to this node (not active). * It *does have DiCaprio* in it. Connected to this node (active). * This movie *has win* an *Oscar*. Connected to this node (active). * *Tarantino* is *not* the *director* of this movie. It's not connected to this node. The *weight* here is *low* or very *insignificant*. |  |
| * Among *5-hidden-nodes* 4 are connected and 3 of them are active, hence *The Departed* is going to be a movie that this *person* is *going* to *like*. The predicted rating will be **1**. | |

So there we go, that's how the RBM works. RBM is detecting some kind of sequence from the given data and filling the gaps (predicting unknown) according to presented data.