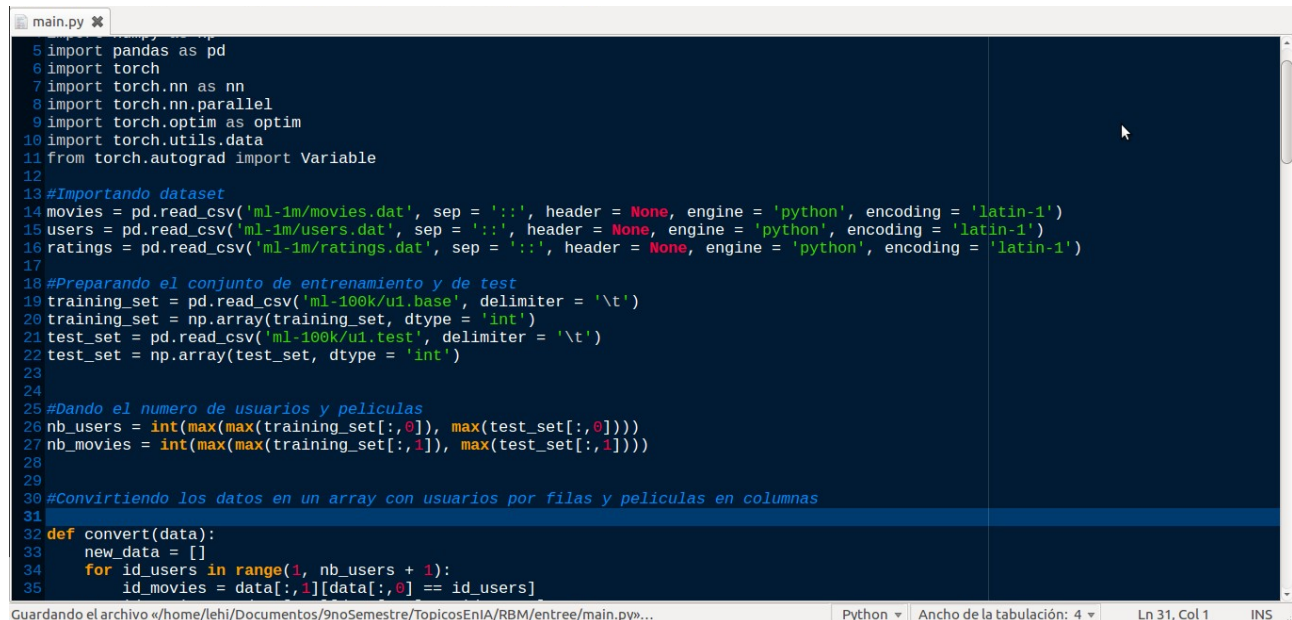


Laboratorio Boltzmann Machines

Para el presente Laboratorio, se utilizó el dataSet de Movie Lens, <https://grouplens.org/datasets/movielens/>, y le implementó según la practica :



```
main.py *
5 import pandas as pd
6 import torch
7 import torch.nn as nn
8 import torch.nn.parallel
9 import torch.optim as optim
10 import torch.utils.data
11 from torch.autograd import Variable
12
13 #Importando dataset
14 movies = pd.read_csv('ml-1m/movies.dat', sep = '::', header = None, engine = 'python', encoding = 'latin-1')
15 users = pd.read_csv('ml-1m/users.dat', sep = '::', header = None, engine = 'python', encoding = 'latin-1')
16 ratings = pd.read_csv('ml-1m/ratings.dat', sep = '::', header = None, engine = 'python', encoding = 'latin-1')
17
18 #Preparando el conjunto de entrenamiento y de test
19 training_set = pd.read_csv('ml-100k/u1.base', delimiter = '\t')
20 training_set = np.array(training_set, dtype = 'int')
21 test_set = pd.read_csv('ml-100k/u1.test', delimiter = '\t')
22 test_set = np.array(test_set, dtype = 'int')
23
24
25 #Dando el numero de usuarios y peliculas
26 nb_users = int(max(max(training_set[:,0]), max(test_set[:,0])))
27 nb_movies = int(max(max(training_set[:,1]), max(test_set[:,1])))
28
29
30 #Convirtiendo los datos en un array con usuarios por filas y peliculas en columnas
31
32 def convert(data):
33     new_data = []
34     for id_users in range(1, nb_users + 1):
35         id_movies = data[:,1][data[:,0] == id_users]
```

Guardando el archivo «/home/lehi/Documentos/9noSemestre/TopicosEnIA/RBM/entree/main.py»... Python ▾ Ancho de la tabulación: 4 ▾ Ln 31, Col 1 INS

Average Distance

```
nb_epoch = 10
for epoch in range(1, nb_epoch + 1):
    train_loss = 0
    s = 0.
    for id_user in range(0, nb_users - batch_size, batch_size):
        vk = training_set[id_user:id_user+batch_size]
        v0 = training_set[id_user:id_user+batch_size]
        ph0,_ = rbm.sample_h(v0)
        for k in range(10):
            _,hk = rbm.sample_h(vk)
            _,vk = rbm.sample_v(hk)
            vk[v0<0] = v0[v0<0]
            phk,_ = rbm.sample_h(vk)
            rbm.train(v0, vk, ph0, phk)
            train_loss += torch.mean(torch.abs(v0[v0>=0] - vk[v0>=0]))
        s += 1.
    print('epoch: '+str(epoch)+' loss: '+str(train_loss/s))

# Testing the RBM
test_loss = 0
s = 0.
for id_user in range(nb_users):
```

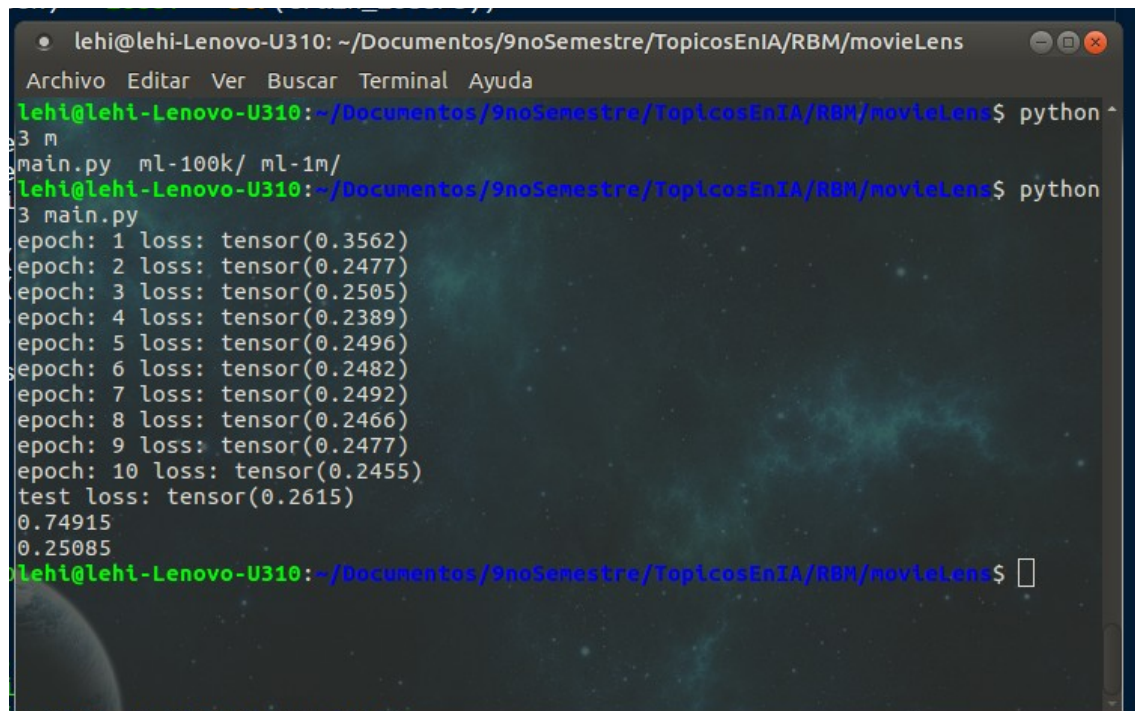
```

v = training_set[id_user:id_user+1]
vt = test_set[id_user:id_user+1]
if len(vt[vt>=0]) > 0:
    _,h = rbm.sample_h(v)
    _,v = rbm.sample_v(h)
    test_loss += torch.mean(torch.abs(vt[vt>=0] - v[vt>=0]))
    s += 1.
print('test loss: '+str(test_loss/s))

```

Resultados

Con esta metrica obtuvimos una distancia promedio de 0.26, que es equivalente a que el 74% de las predicciones son correctas.



```

lehi@lehi-Lenovo-U310: ~/Documentos/9noSemestre/TopicosEnIA/RBM/movieLens
Archivo Editar Ver Buscar Terminal Ayuda
lehi@lehi-Lenovo-U310:~/Documentos/9noSemestre/TopicosEnIA/RBM/movieLens$ python main.py
3 m
main.py ml-100k/ ml-1m/
lehi@lehi-Lenovo-U310:~/Documentos/9noSemestre/TopicosEnIA/RBM/movieLens$ python main.py
3 main.py
epoch: 1 loss: tensor(0.3562)
epoch: 2 loss: tensor(0.2477)
epoch: 3 loss: tensor(0.2505)
epoch: 4 loss: tensor(0.2389)
epoch: 5 loss: tensor(0.2496)
epoch: 6 loss: tensor(0.2482)
epoch: 7 loss: tensor(0.2492)
epoch: 8 loss: tensor(0.2466)
epoch: 9 loss: tensor(0.2477)
epoch: 10 loss: tensor(0.2455)
test loss: tensor(0.2615)
0.74915
0.25085
lehi@lehi-Lenovo-U310:~/Documentos/9noSemestre/TopicosEnIA/RBM/movieLens$

```

RMSE (Root Mean Squared Error)

```

nb_epoch = 10
for epoch in range(1, nb_epoch + 1):
    train_loss = 0
    s = 0.
    for id_user in range(0, nb_users - batch_size, batch_size):
        vk = training_set[id_user:id_user+batch_size]
        v0 = training_set[id_user:id_user+batch_size]
        ph0,_ = rbm.sample_h(v0)
        for k in range(10):
            _,hk = rbm.sample_h(vk)
            _,vk = rbm.sample_v(hk)
            vk[v0<0] = v0[v0<0]
        phk,_ = rbm.sample_h(vk)
        rbm.train(v0, vk, ph0, phk)
        train_loss += np.sqrt(torch.mean((v0[v0>=0] - vk[v0>=0])**2)) # RMSE here

```

```

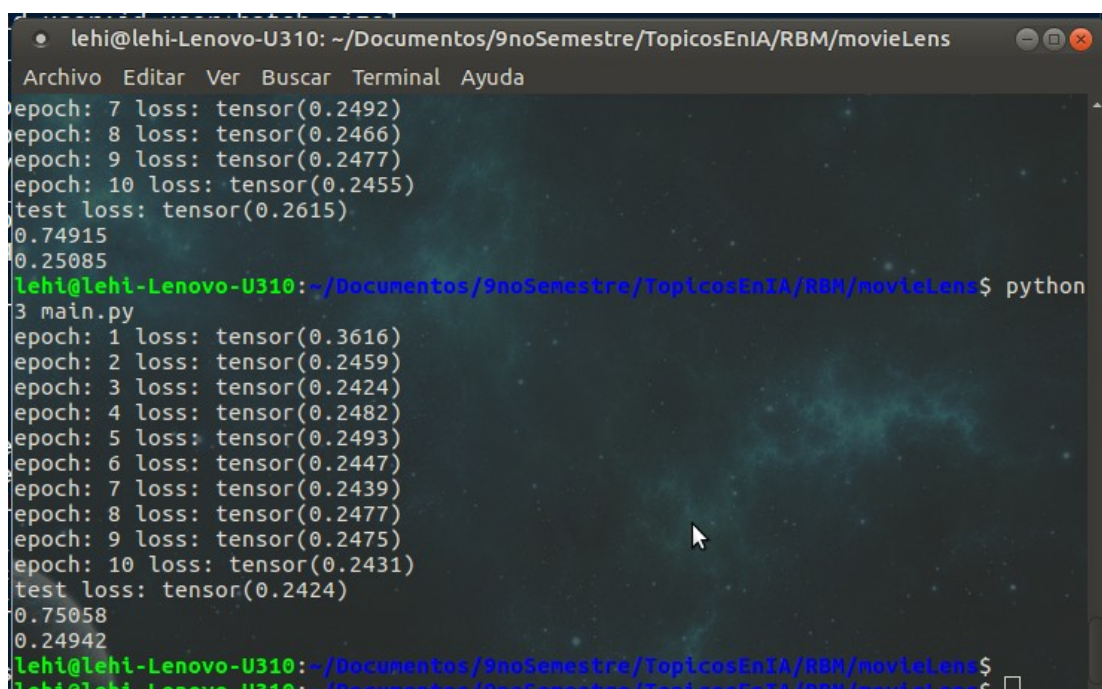
        s += 1.
print('epoch: '+str(epoch)+' loss: '+str(train_loss/s))

test_loss = 0
s = 0.
for id_user in range(nb_users):
    v = training_set[id_user:id_user+1]
    vt = test_set[id_user:id_user+1]
    if len(vt[vt>=0]) > 0:
        _,h = rbm.sample_h(v)
        _,v = rbm.sample_v(h)
        test_loss += np.sqrt(torch.mean((vt[vt>=0] - v[vt>=0])**2)) # RMSE herh
        s += 1.
print('test loss: '+str(test_loss/s))

```

Resultados

Con esta metrica obtuvimos una distancia promedio de 0.24, que es equivalente a que el 76% de las predicciones son correctas.



```

lehi@lehi-Lenovo-U310: ~/Documentos/9noSemestre/TopicosEnIA/RBM/movieLens
Archivo Editar Ver Buscar Terminal Ayuda
epoch: 7 loss: tensor(0.2492)
epoch: 8 loss: tensor(0.2466)
epoch: 9 loss: tensor(0.2477)
epoch: 10 loss: tensor(0.2455)
test loss: tensor(0.2615)
0.74915
0.25085
lehi@lehi-Lenovo-U310:~/Documentos/9noSemestre/TopicosEnIA/RBM/movieLens$ python
3 main.py
epoch: 1 loss: tensor(0.3616)
epoch: 2 loss: tensor(0.2459)
epoch: 3 loss: tensor(0.2424)
epoch: 4 loss: tensor(0.2482)
epoch: 5 loss: tensor(0.2493)
epoch: 6 loss: tensor(0.2447)
epoch: 7 loss: tensor(0.2439)
epoch: 8 loss: tensor(0.2477)
epoch: 9 loss: tensor(0.2475)
epoch: 10 loss: tensor(0.2431)
test loss: tensor(0.2424)
0.75058
0.24942
lehi@lehi-Lenovo-U310:~/Documentos/9noSemestre/TopicosEnIA/RBM/movieLens$
lehi@lehi-Lenovo-U310:~/Documentos/9noSemestre/TopicosEnIA/RBM/movieLens$

```

¿Qué interpretación daría a los resultados del punto anterior?

Las dos formas de evaluar nuestra RBM son con el RMSE y la distancia media.

- El RMSE (Root Mean Squared Error) se calcula como la raíz cuadrada de la media de las diferencias al cuadrado entre las predicciones generada y el objetivos deseado.
- La Distancia Promedio es más intuitiva. Se calcula con la sumatoria de la diferencia.

Usando la base de datos de EnTree Chicago



Data Set Characteristics:	Transactional, Sequential	Number of Instances:	50672	Area:	N/A
Attribute Characteristics:	Categorical	Number of Attributes:	N/A	Date Donated	2000-03-09
Associated Tasks:	Recommender-Systems	Missing Values?	Yes	Number of Web Hits:	80279

Dataset :

<https://archive.ics.uci.edu/ml/datasets/Entree+Chicago+Recommendation+Data>

No se puede realizar un entregamiento con este tipo de dataset, debido a que es de tipo transaccional y no contiene datos multivariable. Este formato es usado en otros tipos de sistema de recomendaciones.