Project work

Balogh Nóra, gyulai László, Vargha Noémi

In this project we have implemented 2 dimensional Ising model with Monte-Carlo simulation. We aimed to train neural networks based on the simulation results. The main parts of our work:

1. Implementing Monte-Carlo simulation code for the Ising model
2. Generating training data
3.

Feladatok:

1. training data generáló függvény
2. adat feldolgozás
   - fejléc: size,h,k,M,E,[mátrix]
   - kimenet: megnézni, hogy mit kér a háló (kép, vagy adatsor)
   - labels/tareget:
     - A. M, E
     - A. K
     - A. h
   - train_test_split
3. neurális háló

Ideas:

[https://arxiv.org/pdf/1706.09779.pdf (https://arxiv.org/pdf/1706.09779.pdf)](https://arxiv.org/pdf/1706.09779.pdf)

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv
from numpy.random import rand
from numpy.random import randint
```

In [2]:
```
import random
```

In [3]:
```
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers.core import Dense
from keras.optimizers import SGD, Adam
from sklearn.preprocessing import MinMaxScaler
```

```
Using TensorFlow backend.
```

1. Monte-Carlo simulation code for the Ising model

We used a square lattice od size LxL=10x10. Each cell in the square lattice has spin $\sigma_i = \pm 1$ which is initialized randomly. The Hamiltonian of the system is:

$$H = -K \sum_{\langle i,j \rangle} \sigma_i \sigma_J - h \sum_i \sigma_i$$

where the summation runs over the nearest neighbors, K is coupling strength between the nearest neighbors and h is the external field.

assiged each lattice cell a random spin of ± 1 with the function below:

Parameters of the 2D Ising model:

- hi: interaction with external field
- Kij: interactions between neighbours

In [4]:
```
def initialize(L):
    ''' generates a random spin (+1 or -1) configuration for initial conditio
n'''
    state = 2*np.random.randint(2, size=(L,L))-1
    return state #square lattice, cells filled randomly with +1 or -1
```

Magnetization is the average value of the spin. (Source: https://en.wikipedia.org/wiki/Ising_model (https://en.wikipedia.org/wiki/Ising_model)). The energy of a configuration is calculated from the Hamiltonian.

In [5]:
```
def magnetization(config):
    '''Magnetization of a given configuration'''
    mag = np.sum(config)/(config.shape[0]*config.shape[1])
    return mag

def Energy(config, h, K):
    '''Calculates energy of a given configuration'''
    energy = 0
    L=len(config)
    for i in range(L):
        for j in range(L):
            S = config[i,j]
            energy-=h*S
            #neighbors with periodic boundary conditions
            neighbors = config[(i+1)%L, j] + config[i,(j+1)%L] + config[(i-1)%
L, j] + config[i,(j-1)%L]
            energy -= K*neighbors*S
    return energy
```

One Monte Carlo timestep consists of LxL elementary step in which a spin is chosen randomly and flipped according to the Metropolis probabilities.

In each elementary step we randomly choose a lattice point with spin s. If the spin is flipped, the cost is $\Delta E = E_{flipped} - E_{original}$. From the Hamiltonian:

$$\Delta E = 2hs - 2Ks \cdot nb$$

where nb is the sum of the spins of the nearest neighbors of s. The function "mcmove" below calculates this cost and flips the spin if $\Delta E < 0$ and flips this spin with probability $p = \exp(\beta \cdot \Delta E)$ if $E > 0$. ($\beta = \frac{1}{k_b T}$)

In [6]:
```
def mcmove(config, beta, h, K):
    '''Monte Carlo move using Metropolis algorithm '''
    L=len(config)
    for i in range(L):
        for j in range(L):
                a = np.random.randint(0, L)
                b = np.random.randint(0, L)
                s =  config[a, b]
                neighbors = config[(a+1)%L,b] + config[a,(b+1)%L] + config[(a-
1)%L,b] + config[a,(b-1)%L]
                #cost=difference between energies
                cost = 2*h*s + 2*K*neighbors*s
                if cost < 0:
                    s *= -1
                elif np.random.rand() < np.exp(-cost*beta):
                    s *= -1
                config[a, b] = s
    return config
```

Use Monte carlo step as: config=mcmove(config, beta, h, K)

2. Training data generation

2.1 Generation of lattices for energy and magnetization prediction

In [7]:
```
K=1
beta=0.2
L=10
h=5
timesteps = 500
iterations = 20
config_data=[]


for i in range(iterations):
    config=initialize(L)
    E,M=np.zeros(timesteps), np.zeros(timesteps)
    M0=np.zeros(timesteps)
    for t in range(timesteps):
        E[t]=Energy(config, h, K)
        M[t]=magnetization(config)
        row1 = [E[t]]+[M[t]]+list(config.flatten())
        config_data.append(row1)
        config=mcmove(config, beta, h, K)
```

In [8]:
```
len(config_data)
```

Out[8]:
```
10000
```

We will store in a .csv file the following features of each generated lattice:

- energy
- magnetization
- config: the lattice configuration (array of +1, -1 values)

In [9]:
```
with open('training_data.csv', 'w+', newline='') as writeFile:
    writer = csv.writer(writeFile)
    writer.writerows(config_data)
writeFile.close()
```

2.2. Training data generation for coupling strength prediction

In each iteration we use different coupling strength and step

In [10]:
```
K=1
beta=0.2
L=10
h=5
timesteps = 10
iterations = 5000
config_data=[]

E,M=np.zeros(iterations), np.zeros(iterations)
#Klist=[j/1000 for j in range(iterations)]
for i in range(iterations):
    K=random.random()*2  # k is a random value between 0 and 2
    config=initialize(L)
    for t in range(timesteps):
        config=mcmove(config, beta, h, K)
    E[i]=Energy(config, h, K)
    M[i]=magnetization(config)
    row1 = [K]+[E[i]]+[M[i]]+list(config.flatten())
    config_data.append(row1)
```

In [187]:
```
len(config_data)
```

Out[187]:
```
5000
```

In [188]:
```
with open('training_data2.csv', 'w+', newline='') as writeFile:
    writer = csv.writer(writeFile)
    writer.writerows(config_data)
writeFile.close()
```

3. Neural Network learns to calculate energy and magnetization

3.1. Loading the data

In [49]:
```
traindata = pd.read_csv("training_data.csv", names=["E"]+["M"]+[i for i in ran
ge(L*L)])
```

In [53]:
```
trainlabelsE=traindata["E"].values
trainlabelsM=traindata["M"].values
```

In [192]:
```
train_attrs=traindata.drop(columns=["E", "M"])
```

In [ ]:
```

```

Loading the data

In [189]:
```
traindata2 = pd.read_csv("training_data2.csv", names=["K"]+["E"]+["M"]+[i for
i in range(L*L)])
```

In [190]:
```
trainlabelsK=traindata2["K"].values
```

In [191]:
```
train_attrs2=traindata2.drop(columns=["K"])
```

In [89]:
```

```

In [91]:
```
max(trainlabelsM)
```

Out[91]:
```
1.0
```

In [92]:
```
min(trainlabelsM)
```

Out[92]:
```
-0.12
```

In [93]:
```
type(trainlabelsM)
```

Out[93]:
```
numpy.ndarray
```

In [94]:
```
trainlabelsM
```

Out[94]:
```
array([0.02, 0.52, 0.74, ..., 0.96, 0.92, 0.96])
```

In [95]:
```
trainlabelsM=(trainlabelsM-(-1))/(1-(-1))
```

In [96]:
```
trainlabelsM
```

Out[96]:
```
array([0.51, 0.76, 0.87, ..., 0.98, 0.96, 0.98])
```

In [97]:
```
X_train, X_test, Y_train, Y_test = train_test_split(train_attrs, trainlabelsM,
test_size=0.33, random_state=0)
```

In [98]:
```
X_train.shape, Y_train.shape, X_test.shape, Y_test.shape
```

Out[98]:
```
((6700, 100), (6700,), (3300, 100), (3300,))
```

In [99]:
```
model = Sequential()
model.add(Dense(100, input_dim=100, activation='relu'))
model.add(Dense(1, activation='relu'))
```

In [100]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_9 (Dense)              (None, 100)               10100     
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 101       
=================================================================
Total params: 10,201
Trainable params: 10,201
Non-trainable params: 0
_________________________________________________________________
```

In [119]:
```
model.compile(loss='mean_squared_error', optimizer='adam')
```

In [120]:

```
history = model.fit(x=X_train, y=Y_train, batch_size=32, epochs=100, validatio
n_data=(X_test, Y_test))
```

```
Train on 6700 samples, validate on 3300 samples
Epoch 1/100
6700/6700 [==============================] - 0s 63us/step - loss: 9.1564e-04
- val_loss: 8.7617e-05
Epoch 2/100
6700/6700 [==============================] - 0s 24us/step - loss: 2.0900e-05
- val_loss: 6.7064e-05
Epoch 3/100
6700/6700 [==============================] - 0s 23us/step - loss: 2.1068e-05
- val_loss: 1.0166e-04
Epoch 4/100
6700/6700 [==============================] - 0s 22us/step - loss: 2.6724e-05
- val_loss: 6.3427e-05
Epoch 5/100
6700/6700 [==============================] - 0s 22us/step - loss: 1.2448e-05
- val_loss: 7.0155e-05
Epoch 6/100
6700/6700 [==============================] - 0s 23us/step - loss: 1.1392e-05
- val_loss: 5.7603e-05
Epoch 7/100
6700/6700 [==============================] - 0s 22us/step - loss: 4.1275e-05
- val_loss: 1.6797e-04
Epoch 8/100
6700/6700 [==============================] - 0s 22us/step - loss: 8.0378e-05
- val_loss: 6.5883e-05
Epoch 9/100
6700/6700 [==============================] - 0s 22us/step - loss: 1.2076e-04
- val_loss: 9.1684e-05
Epoch 10/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.5603e-05
- val_loss: 1.5618e-04
Epoch 11/100
6700/6700 [==============================] - 0s 22us/step - loss: 1.1303e-04
- val_loss: 9.4105e-05
Epoch 12/100
6700/6700 [==============================] - 0s 22us/step - loss: 6.0583e-05
- val_loss: 8.1875e-05
Epoch 13/100
6700/6700 [==============================] - 0s 22us/step - loss: 7.0207e-05
- val_loss: 1.5580e-04
Epoch 14/100
6700/6700 [==============================] - 0s 23us/step - loss: 6.5184e-05
- val_loss: 6.3218e-05
Epoch 15/100
6700/6700 [==============================] - 0s 22us/step - loss: 9.4633e-05
- val_loss: 1.5816e-04
Epoch 16/100
6700/6700 [==============================] - 0s 22us/step - loss: 5.4746e-05
- val_loss: 9.8629e-05
Epoch 17/100
6700/6700 [==============================] - 0s 22us/step - loss: 5.9973e-05
- val_loss: 2.0471e-04
Epoch 18/100
6700/6700 [==============================] - 0s 22us/step - loss: 6.0595e-05
- val_loss: 1.4119e-04
Epoch 19/100
6700/6700 [==============================] - 0s 24us/step - loss: 8.5788e-05
```

```
- val_loss: 1.4809e-04
Epoch 20/100
6700/6700 [==============================] - 0s 23us/step - loss: 5.0184e-05
- val_loss: 1.4935e-04
Epoch 21/100
6700/6700 [==============================] - 0s 25us/step - loss: 7.8490e-05
- val_loss: 1.7583e-04
Epoch 22/100
6700/6700 [==============================] - 0s 25us/step - loss: 6.4175e-05
- val_loss: 9.5871e-05
Epoch 23/100
6700/6700 [==============================] - 0s 22us/step - loss: 7.0334e-05
- val_loss: 1.0095e-04
Epoch 24/100
6700/6700 [==============================] - 0s 22us/step - loss: 4.6352e-05
- val_loss: 3.7959e-04
Epoch 25/100
6700/6700 [==============================] - 0s 22us/step - loss: 6.3040e-05
- val_loss: 1.8743e-04
Epoch 26/100
6700/6700 [==============================] - 0s 22us/step - loss: 7.4954e-05
- val_loss: 1.2332e-04
Epoch 27/100
6700/6700 [==============================] - 0s 23us/step - loss: 7.1813e-05
- val_loss: 1.4792e-04
Epoch 28/100
6700/6700 [==============================] - 0s 22us/step - loss: 5.5691e-05
- val_loss: 1.0767e-04
Epoch 29/100
6700/6700 [==============================] - 0s 24us/step - loss: 4.9191e-05
- val_loss: 8.6910e-05
Epoch 30/100
6700/6700 [==============================] - 0s 25us/step - loss: 1.2072e-04
- val_loss: 9.8178e-05
Epoch 31/100
6700/6700 [==============================] - 0s 22us/step - loss: 2.0761e-05
- val_loss: 1.0596e-04
Epoch 32/100
6700/6700 [==============================] - 0s 22us/step - loss: 6.3649e-05
- val_loss: 2.1489e-04
Epoch 33/100
6700/6700 [==============================] - 0s 26us/step - loss: 6.0590e-05
- val_loss: 9.4615e-05
Epoch 34/100
6700/6700 [==============================] - 0s 24us/step - loss: 3.8090e-05
- val_loss: 9.2219e-05
Epoch 35/100
6700/6700 [==============================] - 0s 22us/step - loss: 7.9631e-05
- val_loss: 1.3629e-04
Epoch 36/100
6700/6700 [==============================] - 0s 22us/step - loss: 5.3275e-05
- val_loss: 3.1267e-04
Epoch 37/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.8345e-05
- val_loss: 1.7964e-04
Epoch 38/100
6700/6700 [==============================] - 0s 22us/step - loss: 7.7989e-05
```

```
- val_loss: 8.7461e-05
Epoch 39/100
6700/6700 [==============================] - 0s 23us/step - loss: 6.6540e-05
- val_loss: 1.1150e-04
Epoch 40/100
6700/6700 [==============================] - 0s 23us/step - loss: 6.2521e-05
- val_loss: 1.8994e-04
Epoch 41/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.7425e-05
- val_loss: 1.0586e-04
Epoch 42/100
6700/6700 [==============================] - 0s 23us/step - loss: 8.3088e-05
- val_loss: 8.7077e-05
Epoch 43/100
6700/6700 [==============================] - 0s 23us/step - loss: 2.1691e-05
- val_loss: 1.4305e-04
Epoch 44/100
6700/6700 [==============================] - 0s 25us/step - loss: 1.0308e-04
- val_loss: 1.0411e-04
Epoch 45/100
6700/6700 [==============================] - 0s 22us/step - loss: 2.2447e-05
- val_loss: 7.2787e-05
Epoch 46/100
6700/6700 [==============================] - 0s 26us/step - loss: 3.3206e-05
- val_loss: 1.3500e-04
Epoch 47/100
6700/6700 [==============================] - 0s 26us/step - loss: 7.0208e-05
- val_loss: 8.4790e-05
Epoch 48/100
6700/6700 [==============================] - 0s 23us/step - loss: 4.1142e-05
- val_loss: 7.0077e-05
Epoch 49/100
6700/6700 [==============================] - 0s 22us/step - loss: 5.8883e-05
- val_loss: 9.1758e-05
Epoch 50/100
6700/6700 [==============================] - 0s 23us/step - loss: 4.1245e-05
- val_loss: 0.0010
Epoch 51/100
6700/6700 [==============================] - 0s 23us/step - loss: 6.4801e-05
- val_loss: 6.9781e-05
Epoch 52/100
6700/6700 [==============================] - 0s 23us/step - loss: 4.3144e-05
- val_loss: 8.2047e-05
Epoch 53/100
6700/6700 [==============================] - 0s 23us/step - loss: 5.2509e-05
- val_loss: 1.4120e-04
Epoch 54/100
6700/6700 [==============================] - 0s 22us/step - loss: 5.9102e-05
- val_loss: 1.6049e-04
Epoch 55/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.1468e-05
- val_loss: 8.2004e-05
Epoch 56/100
6700/6700 [==============================] - 0s 22us/step - loss: 6.2887e-05
- val_loss: 1.5445e-04
Epoch 57/100
6700/6700 [==============================] - 0s 22us/step - loss: 2.8193e-05
```

```
- val_loss: 1.1979e-04
Epoch 58/100
6700/6700 [==============================] - 0s 23us/step - loss: 6.3159e-05
- val_loss: 8.9022e-05
Epoch 59/100
6700/6700 [==============================] - 0s 22us/step - loss: 4.7430e-05
- val_loss: 6.5298e-05
Epoch 60/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.2163e-05
- val_loss: 1.4843e-04
Epoch 61/100
6700/6700 [==============================] - 0s 22us/step - loss: 8.7798e-05
- val_loss: 1.5636e-04
Epoch 62/100
6700/6700 [==============================] - 0s 24us/step - loss: 2.2588e-05
- val_loss: 6.3540e-05
Epoch 63/100
6700/6700 [==============================] - 0s 24us/step - loss: 3.0758e-05
- val_loss: 5.5897e-05
Epoch 64/100
6700/6700 [==============================] - 0s 24us/step - loss: 7.1744e-05
- val_loss: 6.7922e-05
Epoch 65/100
6700/6700 [==============================] - 0s 24us/step - loss: 2.2162e-05
- val_loss: 1.0959e-04
Epoch 66/100
6700/6700 [==============================] - 0s 24us/step - loss: 6.3420e-05
- val_loss: 9.6461e-05
Epoch 67/100
6700/6700 [==============================] - 0s 22us/step - loss: 2.3396e-05
- val_loss: 2.9059e-04
Epoch 68/100
6700/6700 [==============================] - 0s 24us/step - loss: 7.2875e-05
- val_loss: 5.8689e-05
Epoch 69/100
6700/6700 [==============================] - 0s 23us/step - loss: 3.4696e-05
- val_loss: 5.1729e-05
Epoch 70/100
6700/6700 [==============================] - 0s 24us/step - loss: 2.7825e-05
- val_loss: 4.8700e-05
Epoch 71/100
6700/6700 [==============================] - 0s 24us/step - loss: 4.4659e-05
- val_loss: 4.8446e-05
Epoch 72/100
6700/6700 [==============================] - 0s 23us/step - loss: 5.6822e-05
- val_loss: 6.2089e-05
Epoch 73/100
6700/6700 [==============================] - 0s 24us/step - loss: 3.1455e-05
- val_loss: 4.8126e-05
Epoch 74/100
6700/6700 [==============================] - 0s 23us/step - loss: 3.8039e-05
- val_loss: 5.0330e-05
Epoch 75/100
6700/6700 [==============================] - 0s 23us/step - loss: 4.8658e-05
- val_loss: 7.1178e-05
Epoch 76/100
6700/6700 [==============================] - 0s 23us/step - loss: 2.5795e-05
```

```
- val_loss: 4.5331e-05
Epoch 77/100
6700/6700 [==============================] - 0s 22us/step - loss: 4.8680e-05
- val_loss: 4.4901e-05
Epoch 78/100
6700/6700 [==============================] - 0s 23us/step - loss: 6.1085e-05
- val_loss: 4.8984e-05
Epoch 79/100
6700/6700 [==============================] - 0s 23us/step - loss: 1.5676e-05
- val_loss: 6.3148e-05
Epoch 80/100
6700/6700 [==============================] - 0s 24us/step - loss: 5.4571e-05
- val_loss: 6.3398e-05
Epoch 81/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.5151e-05
- val_loss: 2.1019e-04
Epoch 82/100
6700/6700 [==============================] - 0s 23us/step - loss: 3.1313e-05
- val_loss: 8.3907e-05
Epoch 83/100
6700/6700 [==============================] - 0s 24us/step - loss: 6.1366e-05
- val_loss: 8.1717e-05
Epoch 84/100
6700/6700 [==============================] - 0s 23us/step - loss: 1.6458e-05
- val_loss: 4.2932e-05
Epoch 85/100
6700/6700 [==============================] - 0s 23us/step - loss: 5.0256e-05
- val_loss: 2.5407e-04
Epoch 86/100
6700/6700 [==============================] - 0s 24us/step - loss: 2.4775e-05
- val_loss: 4.9913e-05
Epoch 87/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.6504e-05
- val_loss: 3.9553e-05
Epoch 88/100
6700/6700 [==============================] - 0s 24us/step - loss: 4.4981e-05
- val_loss: 9.8802e-05
Epoch 89/100
6700/6700 [==============================] - 0s 23us/step - loss: 2.9835e-05
- val_loss: 1.0966e-04
Epoch 90/100
6700/6700 [==============================] - 0s 23us/step - loss: 4.2493e-05
- val_loss: 3.9500e-05
Epoch 91/100
6700/6700 [==============================] - 0s 23us/step - loss: 1.9816e-05
- val_loss: 4.1127e-05
Epoch 92/100
6700/6700 [==============================] - 0s 23us/step - loss: 5.5570e-05
- val_loss: 5.2840e-05
Epoch 93/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.1197e-05
- val_loss: 4.3631e-05
Epoch 94/100
6700/6700 [==============================] - 0s 22us/step - loss: 3.4779e-05
- val_loss: 5.5369e-05
Epoch 95/100
6700/6700 [==============================] - 0s 23us/step - loss: 2.4841e-05
```

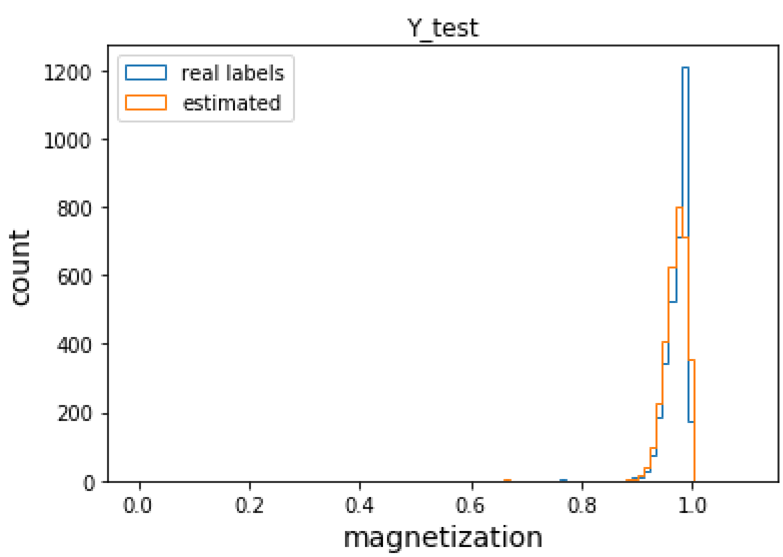
```
- val_loss: 7.7598e-05
Epoch 96/100
6700/6700 [==============================] - 0s 24us/step - loss: 2.8025e-05
- val_loss: 3.8177e-05
Epoch 97/100
6700/6700 [==============================] - 0s 25us/step - loss: 4.1205e-05
- val_loss: 4.2259e-05
Epoch 98/100
6700/6700 [==============================] - 0s 25us/step - loss: 4.6747e-05
- val_loss: 4.9653e-05
Epoch 99/100
6700/6700 [==============================] - 0s 23us/step - loss: 2.7804e-05
- val_loss: 6.7565e-05
Epoch 100/100
6700/6700 [==============================] - 0s 25us/step - loss: 5.6127e-05
- val_loss: 4.0504e-05
```

In [121]:
```
preds = model.predict(X_test)
print('MSE : ', mean_squared_error(preds, Y_test))
```

```
MSE :  4.050433846230874e-05
```

In [125]:
```
Y_hist = plt.hist(Y_test, 100, (0,1.1),histtype="step",alpha= 1.0)
plt.title("Y_test")
pred_hist = plt.hist(preds, 100, (0,1.1),histtype="step",alpha= 1.0)
plt.legend(["real labels", "estimated"], loc='upper left')
plt.xlabel("magnetization", fontsize = 14)
plt.ylabel("count",  fontsize = 14)
```

Out[125]:
```
Text(0, 0.5, 'count')
```



In [145]:
```
trainlabelsE=(trainlabelsE-min(trainlabelsE))/(max(trainlabelsE)-min(trainlabe
lsE))
```

In [153]:
```
max(trainlabelsE)
```

Out[153]:
```
1.0
```

In [154]:
```
X_trainE, X_testE, Y_trainE, Y_testE = train_test_split(train_attrs, trainlabe
lsE, test_size=0.33, random_state=0)
```

In [160]:
```
model2 = Sequential()
model2.add(Dense(100, input_dim=100, activation='relu'))
model2.add(Dense(50, activation='relu'))
model2.add(Dense(1, activation='relu'))
```

In [161]:
```
model2.compile(loss='mean_squared_error', optimizer='adam')
```

In [162]:
```
history = model2.fit(x=X_trainE, y=Y_trainE, batch_size=64, epochs=100, valida
tion_data=(X_testE, Y_testE))
```

```
Train on 6700 samples, validate on 3300 samples
Epoch 1/100
6700/6700 [==============================] - 1s 82us/step - loss: 0.0094 - va
l_loss: 0.0090
Epoch 2/100
6700/6700 [==============================] - 0s 18us/step - loss: 0.0105 - va
l_loss: 0.0074
Epoch 3/100
6700/6700 [==============================] - 0s 17us/step - loss: 0.0095 - va
l_loss: 0.0089
Epoch 4/100
6700/6700 [==============================] - 0s 17us/step - loss: 0.0100 - va
l_loss: 0.0092
Epoch 5/100
6700/6700 [==============================] - 0s 16us/step - loss: 0.0097 - va
l_loss: 0.0089
Epoch 6/100
6700/6700 [==============================] - 0s 17us/step - loss: 0.0096 - va
l_loss: 0.0091
Epoch 7/100
6700/6700 [==============================] - 0s 17us/step - loss: 0.0097 - va
l_loss: 0.0089
Epoch 8/100
6700/6700 [==============================] - 0s 17us/step - loss: 0.0097 - va
l_loss: 0.0092
Epoch 9/100
6700/6700 [==============================] - 0s 15us/step - loss: 0.0098 - va
l_loss: 0.0090
Epoch 10/100
6700/6700 [==============================] - 0s 16us/step - loss: 0.0091 - va
l_loss: 0.0089
Epoch 11/100
6700/6700 [==============================] - 0s 15us/step - loss: 0.0093 - va
l_loss: 0.0088
Epoch 12/100
6700/6700 [==============================] - 0s 15us/step - loss: 0.0063 - va
l_loss: 0.0012
Epoch 13/100
6700/6700 [==============================] - 0s 15us/step - loss: 0.0015 - va
l_loss: 7.5545e-04
Epoch 14/100
6700/6700 [==============================] - 0s 15us/step - loss: 0.0010 - va
l_loss: 6.9252e-04
Epoch 15/100
6700/6700 [==============================] - 0s 15us/step - loss: 6.2906e-04
- val_loss: 5.5176e-04
Epoch 16/100
6700/6700 [==============================] - 0s 16us/step - loss: 4.5927e-04
- val_loss: 5.5094e-04
Epoch 17/100
6700/6700 [==============================] - 0s 15us/step - loss: 3.3547e-04
- val_loss: 4.8892e-04
Epoch 18/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.8392e-04
- val_loss: 4.5270e-04
Epoch 19/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.6080e-04
```

```
- val_loss: 4.5833e-04
Epoch 20/100
6700/6700 [==============================] - 0s 16us/step - loss: 2.5111e-04
- val_loss: 4.3447e-04
Epoch 21/100
6700/6700 [==============================] - 0s 17us/step - loss: 2.2448e-04
- val_loss: 4.4103e-04
Epoch 22/100
6700/6700 [==============================] - 0s 16us/step - loss: 2.0556e-04
- val_loss: 4.2933e-04
Epoch 23/100
6700/6700 [==============================] - 0s 16us/step - loss: 2.2052e-04
- val_loss: 5.1419e-04
Epoch 24/100
6700/6700 [==============================] - 0s 18us/step - loss: 2.2452e-04
- val_loss: 4.0414e-04
Epoch 25/100
6700/6700 [==============================] - 0s 18us/step - loss: 2.2017e-04
- val_loss: 4.7547e-04
Epoch 26/100
6700/6700 [==============================] - 0s 16us/step - loss: 2.2498e-04
- val_loss: 3.8460e-04
Epoch 27/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.3055e-04
- val_loss: 4.9639e-04
Epoch 28/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.6711e-04
- val_loss: 4.2343e-04
Epoch 29/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.3805e-04
- val_loss: 3.9542e-04
Epoch 30/100
6700/6700 [==============================] - 0s 16us/step - loss: 3.0298e-04
- val_loss: 4.0094e-04
Epoch 31/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.6440e-04
- val_loss: 4.0989e-04
Epoch 32/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.6506e-04
- val_loss: 4.1327e-04
Epoch 33/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.8915e-04
- val_loss: 3.7762e-04
Epoch 34/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.1111e-04
- val_loss: 4.1666e-04
Epoch 35/100
6700/6700 [==============================] - 0s 18us/step - loss: 2.1523e-04
- val_loss: 4.2575e-04
Epoch 36/100
6700/6700 [==============================] - 0s 18us/step - loss: 2.1876e-04
- val_loss: 3.8560e-04
Epoch 37/100
6700/6700 [==============================] - 0s 16us/step - loss: 2.0990e-04
- val_loss: 3.5149e-04
Epoch 38/100
6700/6700 [==============================] - 0s 16us/step - loss: 2.2510e-04
```

```
- val_loss: 4.0713e-04
Epoch 39/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.7759e-04
- val_loss: 3.5108e-04
Epoch 40/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.5148e-04
- val_loss: 3.6441e-04
Epoch 41/100
6700/6700 [==============================] - 0s 17us/step - loss: 3.3376e-04
- val_loss: 3.4608e-04
Epoch 42/100
6700/6700 [==============================] - 0s 18us/step - loss: 2.8588e-04
- val_loss: 3.1324e-04
Epoch 43/100
6700/6700 [==============================] - 0s 16us/step - loss: 2.0134e-04
- val_loss: 3.0541e-04
Epoch 44/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.7826e-04
- val_loss: 2.9672e-04
Epoch 45/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.5794e-04
- val_loss: 2.9298e-04
Epoch 46/100
6700/6700 [==============================] - 0s 19us/step - loss: 1.2362e-04
- val_loss: 3.2053e-04
Epoch 47/100
6700/6700 [==============================] - 0s 17us/step - loss: 1.1335e-04
- val_loss: 2.8504e-04
Epoch 48/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.1240e-04
- val_loss: 2.9895e-04
Epoch 49/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.1190e-04
- val_loss: 2.9085e-04
Epoch 50/100
6700/6700 [==============================] - 0s 18us/step - loss: 1.0612e-04
- val_loss: 2.9828e-04
Epoch 51/100
6700/6700 [==============================] - 0s 17us/step - loss: 1.3354e-04
- val_loss: 3.2643e-04
Epoch 52/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.1942e-04
- val_loss: 2.6882e-04
Epoch 53/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.5285e-04
- val_loss: 3.0364e-04
Epoch 54/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.5926e-04
- val_loss: 2.7916e-04
Epoch 55/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.6895e-04
- val_loss: 2.8350e-04
Epoch 56/100
6700/6700 [==============================] - 0s 14us/step - loss: 2.1591e-04
- val_loss: 2.9074e-04
Epoch 57/100
6700/6700 [==============================] - 0s 15us/step - loss: 2.2131e-04
```

```
- val_loss: 2.8316e-04
Epoch 58/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.2239e-04
- val_loss: 2.4534e-04
Epoch 59/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.4152e-04
- val_loss: 2.9019e-04
Epoch 60/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.5623e-04
- val_loss: 2.5816e-04
Epoch 61/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.2447e-04
- val_loss: 2.5405e-04
Epoch 62/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.1393e-04
- val_loss: 2.6260e-04
Epoch 63/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.1870e-04
- val_loss: 3.4765e-04
Epoch 64/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.3640e-04
- val_loss: 2.4325e-04
Epoch 65/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.1465e-04
- val_loss: 2.3780e-04
Epoch 66/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.1040e-04
- val_loss: 2.7461e-04
Epoch 67/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.2320e-04
- val_loss: 2.4664e-04
Epoch 68/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.3218e-04
- val_loss: 2.4993e-04
Epoch 69/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.2674e-04
- val_loss: 2.6135e-04
Epoch 70/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.2310e-04
- val_loss: 2.9199e-04
Epoch 71/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.2553e-04
- val_loss: 2.5324e-04
Epoch 72/100
6700/6700 [==============================] - 0s 15us/step - loss: 9.0784e-05
- val_loss: 2.2701e-04
Epoch 73/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.0455e-04
- val_loss: 2.5564e-04
Epoch 74/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.0674e-04
- val_loss: 2.3021e-04
Epoch 75/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.3226e-04
- val_loss: 3.0951e-04
Epoch 76/100
6700/6700 [==============================] - 0s 15us/step - loss: 9.3432e-05
```

```
- val_loss: 2.7257e-04
Epoch 77/100
6700/6700 [==============================] - 0s 15us/step - loss: 8.2300e-05
- val_loss: 2.3492e-04
Epoch 78/100
6700/6700 [==============================] - 0s 18us/step - loss: 6.7351e-05
- val_loss: 2.3420e-04
Epoch 79/100
6700/6700 [==============================] - 0s 18us/step - loss: 7.0664e-05
- val_loss: 2.2806e-04
Epoch 80/100
6700/6700 [==============================] - 0s 15us/step - loss: 7.1051e-05
- val_loss: 2.2800e-04
Epoch 81/100
6700/6700 [==============================] - 0s 15us/step - loss: 7.7821e-05
- val_loss: 2.3837e-04
Epoch 82/100
6700/6700 [==============================] - 0s 17us/step - loss: 7.6138e-05
- val_loss: 2.2694e-04
Epoch 83/100
6700/6700 [==============================] - 0s 18us/step - loss: 8.9835e-05
- val_loss: 2.9379e-04
Epoch 84/100
6700/6700 [==============================] - 0s 16us/step - loss: 1.1373e-04
- val_loss: 2.3660e-04
Epoch 85/100
6700/6700 [==============================] - 0s 14us/step - loss: 1.4650e-04
- val_loss: 3.0196e-04
Epoch 86/100
6700/6700 [==============================] - 0s 17us/step - loss: 1.3386e-04
- val_loss: 2.0295e-04
Epoch 87/100
6700/6700 [==============================] - 0s 18us/step - loss: 7.3734e-05
- val_loss: 2.2751e-04
Epoch 88/100
6700/6700 [==============================] - 0s 16us/step - loss: 6.4350e-05
- val_loss: 2.6713e-04
Epoch 89/100
6700/6700 [==============================] - 0s 15us/step - loss: 5.8253e-05
- val_loss: 2.1083e-04
Epoch 90/100
6700/6700 [==============================] - 0s 15us/step - loss: 4.7856e-05
- val_loss: 2.3516e-04
Epoch 91/100
6700/6700 [==============================] - 0s 15us/step - loss: 4.7491e-05
- val_loss: 2.3422e-04
Epoch 92/100
6700/6700 [==============================] - 0s 15us/step - loss: 4.6728e-05
- val_loss: 1.9595e-04
Epoch 93/100
6700/6700 [==============================] - 0s 16us/step - loss: 4.8041e-05
- val_loss: 2.3064e-04
Epoch 94/100
6700/6700 [==============================] - 0s 19us/step - loss: 5.7827e-05
- val_loss: 2.3389e-04
Epoch 95/100
6700/6700 [==============================] - 0s 17us/step - loss: 6.5566e-05
```

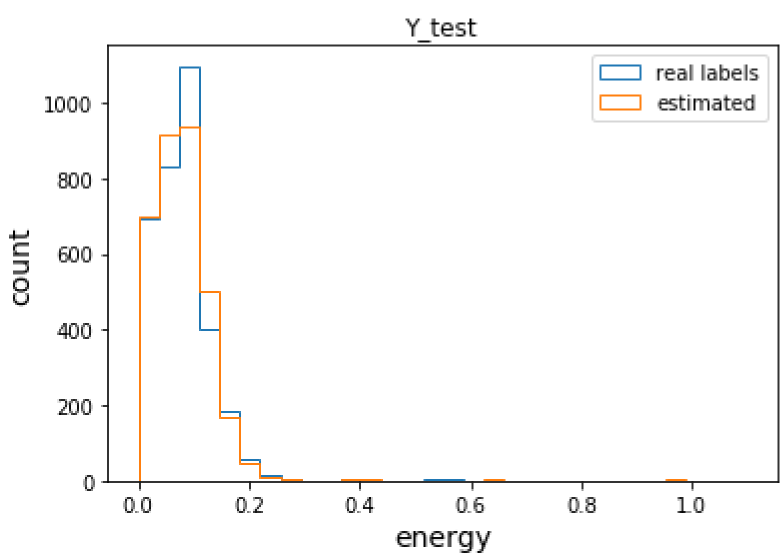
```
- val_loss: 2.0515e-04
Epoch 96/100
6700/6700 [==============================] - 0s 15us/step - loss: 7.9426e-05
- val_loss: 2.1732e-04
Epoch 97/100
6700/6700 [==============================] - 0s 15us/step - loss: 9.7711e-05
- val_loss: 1.9141e-04
Epoch 98/100
6700/6700 [==============================] - 0s 15us/step - loss: 8.7287e-05
- val_loss: 2.5921e-04
Epoch 99/100
6700/6700 [==============================] - 0s 15us/step - loss: 1.0740e-04
- val_loss: 2.3790e-04
Epoch 100/100
6700/6700 [==============================] - 0s 15us/step - loss: 6.9818e-05
- val_loss: 1.8921e-04
```

In [163]:
```
predsE = model2.predict(X_testE)
print('MSE : ', mean_squared_error(predsE, Y_testE))
```

```
MSE :  0.00018921456126465614
```

In [164]:
```
Y_hist = plt.hist(Y_testE, 30, (0,1.1),histtype="step",alpha= 1.0)
plt.title("Y_test")
pred_hist = plt.hist(predsE, 30, (0,1.1),histtype="step",alpha= 1.0)
plt.legend(["real labels", "estimated"])
plt.xlabel("energy", fontsize = 14)
plt.ylabel("count",  fontsize = 14)
```

Out[164]:
```
Text(0, 0.5, 'count')
```



Defining a model

In [ ]: