Question 1

Décrivez et implémentez deux moyens d'améliorer chacune des deux versions linéaires de l'approche de factorisation, la version sans Pytorch (MatrixFactorizationCustomLinearModel) et la version avec Pytorch (MatrixFactorizationPytorchLinearModel). Le code se trouve à ce lien https://github.com/maxbrenner-ai/matrix-factorization-for-collaborative-filtering.

Améliorations

1. **Diminution de *learning rate*** : Le learning rate est un hyperparamètre qui contrôle la taille des pas que le modèle fait lors de l'optimisation. Si le learning rate est trop grand, le modèle peut ne pas converger. Si le *learning rate* est trop petit, le modèle peut converger trop lentement. Il est donc important de trouver un bon *learning rate*. Le *learning rate* qui semble fonctionner le mieux est **0.01**.

2. **Utilisation de l'optimisation ADAM** : L'optimisation ADAM est une méthode d'optimisation qui est souvent utilisée pour l'entraînement des réseaux de neurones. Elle est souvent plus rapide et plus efficace que la descente de gradient stochastique. Elle consiste à adapter le *learning rate* pour chaque paramètre du modèle.

Le code se situe sous-dessous.

Nouvelle performance

performance

```python
import numpy as np
from math import sqrt
import torch
import pandas as pd
import torch.nn.functional as F
import random
from collections import OrderedDict
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rc('font', **{'family': 'DejaVu Sans', 'size'   : 16})
matplotlib.rc('figure', figsize=(15, 7.5))
```

```
C:\Users\nicol\AppData\Local\Temp\ipykernel_31680\103376597.py:4:
DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major
release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type,
and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at
https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd

use_toy_problem = False  # use toy problem or real dataset
K = 32  # num latent features for U and V
train_frac = 0.7  # fraction of R to use as train vs val
num_epochs = 100
lr = 0.01  # learning rate
'''
4 models:
    1. From-scratch linear: 'scratch-linear'
    2. Pytorch linear: 'pytorch-linear'
    3. Pytorch non-linear (simple): 'pytorch-nonlinear-simple'
    4. Pytorch non-linear (complex): 'pytorch-nonlinear-complex'
'''
# Running multiple models will plot the performance on the same plot
model_definitions = [
    'scratch-linear',
    'pytorch-linear'
]  # can specify multiple models
run_all_models = True  # or can just set this to True to run all
models

"""
Toy problem
- Underlying model: Each item has a feature of is action, is comedy,
is rated R, is big-budget

Item 1: action, rated R
Item 2: action, comedy, rated R, big-budget
Item 3: comedy, big-budget
Item 4: big-budget
"""
def make_toy_data():
    R = np.array([
        [4, 4, 3, 4],  # likes action, dislikes comedy, likes big-
budget
        [1, 2, 3, 2],  # likes comedy, dislikes action, dislikes R,
dislikes big-budget
        [4, 4, 3, 3],  # likes R
        [4, 4, 3, 2],  # likes action, likes comedy, dislikes big-
budget
        [2, 4, 5, 4],  # dislikes action, likes comedy, likes big-
budget
    ], dtype=np.float32)
    return R
```

```
def get_dataset(use_toy_problem):
    if use_toy_problem:
        return make_toy_data()
    # ow Movie lens dataset, made smaller
    df = pd.read_csv('ratings.csv').astype(int)
    num_users = len(df.userId.unique())
    num_items = len(df.movieId.unique())
    user_df_dict = dict(zip(df.userId.unique(),
list(range(num_users))))
    items_df_dict = dict(zip(df.movieId.unique(),
list(range(num_items))))
    R = np.full((num_users, num_items), np.nan, dtype=np.float32)
    for indx, df_row in df.iterrows():
        R[user_df_dict[df_row.userId], items_df_dict[df_row.movieId]]
= float(df_row.rating)
    # Make R smaller if you want
#     R = R[:, :610]
    return R

# Get/make dataset
R = get_dataset(use_toy_problem)
num_users, num_items = R.shape

def make_train_and_val_splits(R, num_rows, num_cols, train_frac):
    nan_full = lambda: np.full((num_rows, num_cols), np.nan)
    R_train, R_val = nan_full(), nan_full()
    did_train, did_val = False, False
    for row in range(num_rows):
        for col in range(num_cols):
            if np.random.random() < train_frac:
                R_train[row, col] = R[row, col]
                did_train = True
            else:
                R_val[row, col] = R[row, col]
                did_val = True
    assert did_train and did_val
    return R_train, R_val

# Make train/val splits
R_train, R_val = make_train_and_val_splits(R, num_users, num_items,
train_frac)

def convert_tensor_to_np(t):
    if not torch.is_tensor(t):
        return t
    return t.detach().numpy()

class MatrixFactorizationLinearModel(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.is_linear = True

    def forward(self):
        raise NotImplementedError()

    def calculate_loss(self, R, R_hat):
        raise NotImplementedError()

    def optimize(self, loss, R, R_hat, N):
        raise NotImplementedError()

    @staticmethod
    def mask_out_nans(R, R_hat):
        lib = np if isinstance(R, np.ndarray) else torch
        R_mask = lib.isnan(R)
        R[R_mask] = R_hat[R_mask]
        N = (R.shape[0] * R.shape[1]) - R_mask.sum()
        return R, N

class
MatrixFactorizationCustomLinearModel(MatrixFactorizationLinearModel):
    def __init__(self, num_users, num_items, K, lr):
        super().__init__()
        self.lr = lr
        self.U =
MatrixFactorizationCustomLinearModel.xavier_uniform_array((num_users,
K))
        self.V =
MatrixFactorizationCustomLinearModel.xavier_uniform_array((num_items,
K))
        self.m_u_t = 0
        self.m_v_t = 0
        self.v_u_t = 0
        self.v_v_t = 0
        self.beta1 = 0.9
        self.beta2 = 0.999

    @staticmethod
    def xavier_uniform_array(arr_shape):
        limit = sqrt(6. / sqrt(arr_shape[0] + arr_shape[1]))
        weights = np.random.uniform(-limit, limit, size=arr_shape)
        return weights

    def forward(self):
        P = np.concatenate((self.U, np.ones((self.U.shape[0], 1))), 1)
        Q = np.concatenate((self.V, np.ones((self.V.shape[0], 1))), 1)
        return P @ Q.T

    @staticmethod
    def _calc_mse(R, R_hat, N):
        return np.square(R - R_hat).sum() / N

    def calculate_loss(self, R, R_hat):
        # Mask out nans by setting to R_hat values
        R, N = MatrixFactorizationLinearModel.mask_out_nans(R, R_hat)
        # MSE and masked R
        loss = MatrixFactorizationCustomLinearModel._calc_mse(R,
R_hat, N)
        return loss, loss, R, N

    @staticmethod
    def _calc_mse_gradient(R, R_hat, M, N):
        return (R - R_hat) @ M * -2 / N

    def optimize(self, loss, R, R_hat, N):
        # Calc gradient of MSE for each factor matrix ie parameter
        # MSE grad
        U_grad =
MatrixFactorizationCustomLinearModel._calc_mse_gradient(R, R_hat,
self.V, N)
        V_grad =
MatrixFactorizationCustomLinearModel._calc_mse_gradient(R.T, R_hat.T,
self.U, N)
        # Update U and V based on gradient and step size

        # Adam optimizer
        self.m_u_t = self.beta1 * self.m_u_t + (1 - self.beta1) *
U_grad
        self.v_u_t = self.beta2 * self.v_u_t + (1 - self.beta2) *
np.square(U_grad)
        self.m_v_t = self.beta1 * self.m_v_t + (1 - self.beta1) *
V_grad
        self.v_v_t = self.beta2 * self.v_v_t + (1 - self.beta2) *
np.square(V_grad)
        m_hat_u_t = self.m_u_t / (1 - self.beta1)
        v_hat_u_t = self.v_u_t / (1 - self.beta2)
        m_hat_v_t = self.m_v_t / (1 - self.beta1)
        v_hat_v_t = self.v_v_t / (1 - self.beta2)
        self.U -= self.lr * m_hat_u_t / (np.sqrt(v_hat_u_t) + 1e-8)
        self.V -= self.lr * m_hat_v_t / (np.sqrt(v_hat_v_t) + 1e-8)

class
MatrixFactorizationPytorchLinearModel(MatrixFactorizationLinearModel):
    def __init__(self, num_users, num_items, K, lr):
        super().__init__()
        self.U = torch.nn.Parameter(torch.zeros((num_users, K)))
        self.V = torch.nn.Parameter(torch.zeros((num_items, K)))
        torch.nn.init.xavier_uniform_(self.U)
        torch.nn.init.xavier_uniform_(self.V)
        self.MSE = torch.nn.MSELoss()
        self.optimizer = torch.optim.Adam(
            self.parameters(),
            lr=lr,
        )


    def forward(self):
        P = torch.cat((self.U, torch.ones((self.U.shape[0], 1))), 1)
        Q = torch.cat((self.V, torch.ones((self.V.shape[0], 1))), 1)
        return P @ Q.T

    def calculate_loss(self, R, R_hat):
        R = torch.tensor(R, dtype=torch.float)
        select_mask = ~torch.isnan(R)
        R = torch.masked_select(R, select_mask)
        R_hat = self.forward()
        R_hat = torch.masked_select(R_hat, select_mask)
        t_loss = self.MSE(R_hat, R)
        np_loss = convert_tensor_to_np(t_loss)
        return t_loss, np_loss, R, R.size()

    def optimize(self, loss, R, R_hat, N):
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

def train_linear_model(MF_model, R_train, R_val, num_epochs):
    train_losses, val_losses = [], []
    # Training
    for curr_epoch in range(num_epochs):
        # Train
        # Reconstruct R_hat from latent factor matrices
        R_hat = MF_model.forward()
        # Calc MSE loss of this reconstruction
        # And create a nan masked R_train for future use
        train_loss, np_train_loss, R_train_masked, N =
MF_model.calculate_loss(R_train.copy(), R_hat)
        # Calc grad and update
        MF_model.optimize(train_loss, R_train_masked, R_hat, N)
        train_losses.append(np_train_loss)

        # Eval
        with torch.no_grad():
            val_loss, np_val_loss, _, _ =
MF_model.calculate_loss(R_val.copy(), R_hat)
        val_losses.append(np_val_loss)
        print(f'Epoch {curr_epoch+1} losses -> Train: {np_train_loss}
- Val: {np_val_loss}')
```

```python
    return train_losses, val_losses

class MatrixFactorizationPytorchNonLinearModel(torch.nn.Module):
    def __init__(self, num_users, num_items, K, lr):
        super().__init__()
        self.is_linear = False

        self.U_layer = torch.nn.Linear(num_users, K)
        torch.nn.init.xavier_uniform_(self.U_layer.weight)
        self.V_layer = torch.nn.Linear(num_items, K)
        torch.nn.init.xavier_uniform_(self.V_layer.weight)

        self.MSE = torch.nn.MSELoss()

    def forward(self, U_one_hots, V_one_hots):
        raise NotImplementedError()

    def calculate_loss(self, R, R_hat):
        t_loss = self.MSE(R, R_hat)
        np_loss = convert_tensor_to_np(t_loss)
        return t_loss, np_loss

    def optimize(self, loss):
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

class SimpleNonLinearModel(MatrixFactorizationPytorchNonLinearModel):
    def __init__(self, num_users, num_items, K, lr):
        super().__init__(num_users, num_items, K, lr)
        self.fc1 = torch.nn.Linear(2 * K, K)
        self.fc2 = torch.nn.Linear(K, 1)

        self.optimizer = torch.optim.SGD(self.parameters(), lr=lr)

    def forward(self, U_one_hots, V_one_hots):
        U_latent = self.U_layer(U_one_hots)
        V_latent = self.V_layer(V_one_hots)
        x = torch.cat((U_latent, V_latent), dim=1)
        # Non-linear transform
        x = torch.sigmoid(self.fc1(x))
        R_hat = self.fc2(x)
        return R_hat

class ComplexNonLinearModel(MatrixFactorizationPytorchNonLinearModel):
    def __init__(self, num_users, num_items, K, lr):
        super().__init__(num_users, num_items, K, lr)
        drop_p = 0.25
        self.U_dropout = torch.nn.Dropout(p=drop_p)
        self.V_dropout = torch.nn.Dropout(p=drop_p)

        self.fc1 = torch.nn.Linear(2 * K, K)
        self.dropout1 = torch.nn.Dropout(p=drop_p)
        self.fc2 = torch.nn.Linear(K, K // 2)
        self.dropout2 = torch.nn.Dropout(p=drop_p)
        self.fc3 = torch.nn.Linear(K // 2, 1)

        self.optimizer = torch.optim.Adam(self.parameters(), lr=lr)

    def forward(self, U_one_hots, V_one_hots):
        U_latent = self.U_layer(U_one_hots)
        U_latent = self.U_dropout(U_latent)
        V_latent = self.V_layer(V_one_hots)
        V_latent = self.V_dropout(V_latent)
        x = torch.cat((U_latent, V_latent), dim=1)
        # Non-linear transform
        x = torch.sigmoid(self.fc1(x))
        x = self.dropout1(x)
        x = torch.sigmoid(self.fc2(x))
        x = self.dropout2(x)
        R_hat = self.fc3(x)
        return R_hat

def create_inputs_and_targets(R, num_users, num_items):
    def create_one_hot(length, pos):
        one_hot = np.zeros(shape=(length,), dtype=np.float32)
        one_hot[pos] = 1.
        return one_hot
    # Inputs are one hot vectors
    U_one_hots, V_one_hots = [], []
    # Targets are flattened R without nans
    R_targets = []
    for row in range(num_users):
        row_one_hot = create_one_hot(num_users, row)
        for col in range(num_items):
            value = R[row, col]
            if not np.isnan(value):
                U_one_hots.append(row_one_hot)
                V_one_hots.append(create_one_hot(num_items, col))
                R_targets.append(value)
    return torch.tensor(U_one_hots),torch.tensor(V_one_hots),
torch.tensor(R_targets, dtype=torch.float32).unsqueeze(1)

def train_nonlinear_model(MF_model, R_train, R_val, num_users,
num_items, num_epochs):
    # Dont use matrices here as there can be a lot of "white space" in
a sparse ranking matrix
    U_one_hots_train, V_one_hots_train, R_targets_train =
create_inputs_and_targets(R_train, num_users, num_items)
    U_one_hots_val, V_one_hots_val, R_targets_val =
create_inputs_and_targets(R_val, num_users, num_items)

    train_losses, val_losses = [], []
    # Training
    for curr_epoch in range(num_epochs):
        # Train
        MF_model.train()  # train mode
        # Reconstruct R_hat from latent factor matrices
        R_hat_train = MF_model.forward(U_one_hots_train,
V_one_hots_train)
        # Calc MSE loss of this reconstruction
        train_loss, np_train_loss =
MF_model.calculate_loss(R_targets_train, R_hat_train)
        # Calc grad and update
        MF_model.optimize(train_loss)
        train_losses.append(np_train_loss)

        # Eval
        MF_model.eval()  # eval mode
        R_hat_val = MF_model.forward(U_one_hots_val, V_one_hots_val)
        val_loss, np_val_loss = MF_model.calculate_loss(R_targets_val,
R_hat_val)
        val_losses.append(np_val_loss)
        print(f'Epoch {curr_epoch+1} losses -> Train: {np_train_loss}
- Val: {np_val_loss}')

    return train_losses, val_losses

model_dict = {'scratch-linear': MatrixFactorizationCustomLinearModel,
              'pytorch-linear': MatrixFactorizationPytorchLinearModel,
              'pytorch-nonlinear-simple': SimpleNonLinearModel,
              'pytorch-nonlinear-complex': ComplexNonLinearModel}

def train_model(model_definition, performance_dict):
    model = model_dict[model_definition](num_users, num_items, K, lr)
    print(f'Training {model_definition} model...')
    if model.is_linear:
        train_losses, val_losses = train_linear_model(model, R_train,
R_val, num_epochs)
    else:
        train_losses, val_losses = \
            train_nonlinear_model(model, R_train, R_val, num_users,
num_items, num_epochs)
    print('...completed training\n')
    performance_dict[model_definition] = (train_losses, val_losses)

def train_models():
    performance_dict = OrderedDict()
    # Run all models, or only the specified models
    model_list = list(model_dict.keys()) if run_all_models else
model_definitions
    for model_definition in model_list:
        train_model(model_definition, performance_dict)
    return performance_dict

# Make and train model(s)
performance_dict = train_models()
```

```
Training scratch-linear model...

Epoch 1 losses -> Train: 6.75786315270035 - Val: 6.766292079237437
Epoch 2 losses -> Train: 6.6514548888978995 - Val: 6.768482220181947
Epoch 3 losses -> Train: 6.511780521432257 - Val: 6.771244246093349
Epoch 4 losses -> Train: 6.3483530168029745 - Val: 6.77044178710062
Epoch 5 losses -> Train: 6.161807993813085 - Val: 6.759899112698363
Epoch 6 losses -> Train: 5.949879759913468 - Val: 6.732700691648807
Epoch 7 losses -> Train: 5.709603537187919 - Val: 6.681672102807967
Epoch 8 losses -> Train: 5.438370010997503 - Val: 6.599668506584614
Epoch 9 losses -> Train: 5.134479499832334 - Val: 6.479987491699928
Epoch 10 losses -> Train: 4.797564660371959 - Val: 6.31694661257323
Epoch 11 losses -> Train: 4.428983556217339 - Val: 6.106382980656953
Epoch 12 losses -> Train: 4.0321627615711 - Val: 5.846152369439078
Epoch 13 losses -> Train: 3.612925820842101 - Val: 5.53674513710015
Epoch 14 losses -> Train: 3.1798042040799706 - Val: 5.181958594717941
Epoch 15 losses -> Train: 2.7442438106982974 - Val: 4.7895077304015325
Epoch 16 losses -> Train: 2.3205772884852007 - Val: 4.371370416964784
Epoch 17 losses -> Train: 1.9255620145804526 - Val: 3.9437910935554457
Epoch 18 losses -> Train: 1.5771481237198013 - Val: 3.5265946247902606
Epoch 19 losses -> Train: 1.2920885527720964 - Val: 3.1413683455465184
Epoch 20 losses -> Train: 1.0823610015939042 - Val: 2.808335774123015
Epoch 21 losses -> Train: 0.9511361174163755 - Val: 2.5421850513848856
Epoch 22 losses -> Train: 0.8898249686788409 - Val: 2.348001430274351
Epoch 23 losses -> Train: 0.8783725699689241 - Val: 2.219494610488995
Epoch 24 losses -> Train: 0.8901717025531615 - Val: 2.141394178720843
Epoch 25 losses -> Train: 0.8999319703611777 - Val: 2.0953544530578383
Epoch 26 losses -> Train: 0.8904510236194141 - Val: 2.066128338499944
Epoch 27 losses -> Train: 0.8553320233364475 - Val: 2.0449270735996645
Epoch 28 losses -> Train: 0.7976105376319336 - Val: 2.029370363488114
Epoch 29 losses -> Train: 0.726103946111371 - Val: 2.02126814111813
Epoch 30 losses -> Train: 0.6514259884783339 - Val: 2.0238298982992493
Epoch 31 losses -> Train: 0.5828892739692164 - Val: 2.039450238721318
Epoch 32 losses -> Train: 0.5267485263111032 - Val: 2.0685025291411114
Epoch 33 losses -> Train: 0.48570952376024334 - Val:
2.1090658221326533
Epoch 34 losses -> Train: 0.4593598222094029 - Val: 2.157369645415103
Epoch 35 losses -> Train: 0.4450980121257389 - Val: 2.2086298498657184
Epoch 36 losses -> Train: 0.43918883828491856 - Val:
2.2579295537598965
```

```
Epoch 37 losses -> Train: 0.43768825984483795 - Val:
2.3009248246383582
Epoch 38 losses -> Train: 0.4371090899728419 - Val: 2.3342941039821046
Epoch 39 losses -> Train: 0.4348052824102608 - Val: 2.3559315868457773
Epoch 40 losses -> Train: 0.4291182638387638 - Val: 2.3649412962579484
Epoch 41 losses -> Train: 0.41935029994461825 - Val: 2.361510998405536
Epoch 42 losses -> Train: 0.40562907565474776 - Val: 2.346722920443019
Epoch 43 losses -> Train: 0.38871391135334243 - Val:
2.3223312160892187
Epoch 44 losses -> Train: 0.36977768834256547 - Val: 2.290532880795782
Epoch 45 losses -> Train: 0.35018680286840337 - Val: 2.253752918691604
Epoch 46 losses -> Train: 0.33129480469129796 - Val:
2.2144451843473933
Epoch 47 losses -> Train: 0.31426264829815603 - Val:
2.1749082517076883
Epoch 48 losses -> Train: 0.29991961773581227 - Val:
2.1371304467140755
Epoch 49 losses -> Train: 0.2886784736985404 - Val: 2.102676814765883
Epoch 50 losses -> Train: 0.2805150631880715 - Val: 2.0726231915804627
Epoch 51 losses -> Train: 0.27501569784065616 - Val: 2.047549004668104
Epoch 52 losses -> Train: 0.27148425946290217 - Val:
2.0275933713759513
Epoch 53 losses -> Train: 0.26908877663003034 - Val:
2.0125545105200744
Epoch 54 losses -> Train: 0.26701749142049225 - Val:
2.0020080343439774
Epoch 55 losses -> Train: 0.26461252959454024 - Val:
1.9954266681418884
Epoch 56 losses -> Train: 0.2614562129858944 - Val: 1.9922765134529425
Epoch 57 losses -> Train: 0.25739826938376303 - Val:
1.9920712089759731
Epoch 58 losses -> Train: 0.2525272957996579 - Val: 1.994388553148973
Epoch 59 losses -> Train: 0.24710218627094524 - Val: 1.99885806348829
Epoch 60 losses -> Train: 0.24146535547282524 - Val: 2.005128410311364
Epoch 61 losses -> Train: 0.23595909473609136 - Val: 2.012834518737202
Epoch 62 losses -> Train: 0.2308607932081987 - Val: 2.0215776505427994
Epoch 63 losses -> Train: 0.2263451701218771 - Val: 2.0309192933497733
Epoch 64 losses -> Train: 0.22247412065691363 - Val:
2.0403926548764986
Epoch 65 losses -> Train: 0.21920928906771486 - Val: 2.049530374982261
Epoch 66 losses -> Train: 0.21643944908387205 - Val:
2.0578965355081182
Epoch 67 losses -> Train: 0.21401423990579477 - Val:
2.0651171091604374
Epoch 68 losses -> Train: 0.21177692025551795 - Val:
2.0709068300348767
Epoch 69 losses -> Train: 0.20959084109822884 - Val:
2.0750860492526213
Epoch 70 losses -> Train: 0.20735681064189507 - Val:
```

```
2.0775869253577888
Epoch 71 losses -> Train: 0.20502066859312515 - Val:
2.0784516254496523
Epoch 72 losses -> Train: 0.2025720509859418 - Val: 2.077819795485111
Epoch 73 losses -> Train: 0.20003641882290055 - Val:
2.0759072823865474
Epoch 74 losses -> Train: 0.19746297520553704 - Val:
2.0729827362427344
Epoch 75 losses -> Train: 0.19491112251578324 - Val: 2.069343148459773
Epoch 76 losses -> Train: 0.19243784780860232 - Val:
2.0652904881578866
Epoch 77 losses -> Train: 0.1900878541835794 - Val: 2.0611132389320943
Epoch 78 losses -> Train: 0.18788753855994245 - Val:
2.0570708258702566
Epoch 79 losses -> Train: 0.18584314361264925 - Val:
2.0533813938058425
Epoch 80 losses -> Train: 0.1839427043956334 - Val: 2.050215923724298
Epoch 81 losses -> Train: 0.18216084065562513 - Val:
2.0476962646459715
Epoch 82 losses -> Train: 0.18046509357728063 - Val: 2.045896714203827
Epoch 83 losses -> Train: 0.17882243777687748 - Val:
2.0448498155039307
Epoch 84 losses -> Train: 0.1772047882396414 - Val: 2.0445526354224097
Epoch 85 losses -> Train: 0.17559266972125154 - Val: 2.044973188975398
Epoch 86 losses -> Train: 0.17397670655160422 - Val:
2.0460568118704434
Epoch 87 losses -> Train: 0.17235705433297605 - Val:
2.0477300409392507
Epoch 88 losses -> Train: 0.17074124178727884 - Val:
2.0499045038242008
Epoch 89 losses -> Train: 0.16914111259325507 - Val:
2.0524815395949534
Epoch 90 losses -> Train: 0.16756960184520947 - Val:
2.0553560432970377
Epoch 91 losses -> Train: 0.16603796376327534 - Val: 2.05842105013677
Epoch 92 losses -> Train: 0.16455386117148516 - Val:
2.0615718053536183
Epoch 93 losses -> Train: 0.1631204858798104 - Val: 2.064709040958148
Epoch 94 losses -> Train: 0.16173665951387584 - Val:
2.0677429910296614
Epoch 95 losses -> Train: 0.16039770164579037 - Val:
2.0705964157867824
Epoch 96 losses -> Train: 0.15909675105970017 - Val:
2.0732072744328125
Epoch 97 losses -> Train: 0.1578262179462155 - Val: 2.0755313041432997
Epoch 98 losses -> Train: 0.1565790876690823 - Val: 2.07754286094039
Epoch 99 losses -> Train: 0.1553498750973811 - Val: 2.079235054608907
Epoch 100 losses -> Train: 0.15413513391341838 - Val:
2.080618385505189
```

```
...completed training

Training pytorch-linear model...
Epoch 1 losses -> Train: 6.712428092956543 - Val: 6.714510917663574
Epoch 2 losses -> Train: 6.692338943481445 - Val: 6.71140718460083
Epoch 3 losses -> Train: 6.666724681854248 - Val: 6.702667236328125
Epoch 4 losses -> Train: 6.632691860198975 - Val: 6.686319351196289
Epoch 5 losses -> Train: 6.588453769683838 - Val: 6.660553932189941
Epoch 6 losses -> Train: 6.532397270202637 - Val: 6.623686790466309
Epoch 7 losses -> Train: 6.463062763214111 - Val: 6.574141979217529
Epoch 8 losses -> Train: 6.379121780395508 - Val: 6.510479927062988
Epoch 9 losses -> Train: 6.279392242431641 - Val: 6.431392669677734
Epoch 10 losses -> Train: 6.162842750549316 - Val: 6.335710525512695
Epoch 11 losses -> Train: 6.0286102294921875 - Val: 6.222421646118164
Epoch 12 losses -> Train: 5.87601900100708 - Val: 6.090707302093506
Epoch 13 losses -> Train: 5.704615116119385 - Val: 5.939980506896973
Epoch 14 losses -> Train: 5.5142011642456055 - Val: 5.769925594329834
Epoch 15 losses -> Train: 5.304874420166016 - Val: 5.580533981323242
Epoch 16 losses -> Train: 5.077064514160156 - Val: 5.37214469909668
Epoch 17 losses -> Train: 4.831574440002441 - Val: 5.14549446105957
Epoch 18 losses -> Train: 4.569622993469238 - Val: 4.901770114898682
Epoch 19 losses -> Train: 4.292886734008789 - Val: 4.642641067504883
Epoch 20 losses -> Train: 4.003533840179443 - Val: 4.370294094085693
Epoch 21 losses -> Train: 3.7042529582977295 - Val: 4.0874786376953125
Epoch 22 losses -> Train: 3.3982746601104736 - Val: 3.797513961791992
Epoch 23 losses -> Train: 3.0893735885620117 - Val: 3.5043129920959473
Epoch 24 losses -> Train: 2.7818474769592285 - Val: 3.212320566177368
Epoch 25 losses -> Train: 2.48046612739563 - Val: 2.92645263671875
Epoch 26 losses -> Train: 2.190371513366699 - Val: 2.651963472366333
Epoch 27 losses -> Train: 1.9169217348098755 - Val: 2.3942909240722656
Epoch 28 losses -> Train: 1.6654571294784546 - Val: 2.1587038040161133
Epoch 29 losses -> Train: 1.440975308418274 - Val: 1.949960470199585
Epoch 30 losses -> Train: 1.2477203607559204 - Val: 1.7718931436538696
Epoch 31 losses -> Train: 1.088704228401184 - Val: 1.6268081665039062
Epoch 32 losses -> Train: 0.9652141332626343 - Val: 1.5151236057281494
Epoch 33 losses -> Train: 0.8764094710350037 - Val: 1.435089111328125
Epoch 34 losses -> Train: 0.8191462755203247 - Val: 1.3828516006469727
Epoch 35 losses -> Train: 0.7881452441215515 - Val: 1.352867603302002
Epoch 36 losses -> Train: 0.7765475511550903 - Val: 1.3386831283569336
Epoch 37 losses -> Train: 0.7767753005027771 - Val: 1.3338361978530884
Epoch 38 losses -> Train: 0.7815200090408325 - Val: 1.332647681236267
Epoch 39 losses -> Train: 0.7846411466598511 - Val: 1.3309093713760376
Epoch 40 losses -> Train: 0.7817944288253784 - Val: 1.3261432647705078
Epoch 41 losses -> Train: 0.7706905007362366 - Val: 1.31752610206604
Epoch 42 losses -> Train: 0.7509924173355103 - Val: 1.3056206703186035
Epoch 43 losses -> Train: 0.723931074142456 - Val: 1.291896939277649
Epoch 44 losses -> Train: 0.6917673945426941 - Val: 1.2782095670700073
Epoch 45 losses -> Train: 0.6572213172912598 - Val: 1.266384243965149
Epoch 46 losses -> Train: 0.6229687333106995 - Val: 1.25789213180542
Epoch 47 losses -> Train: 0.591263473033905 - Val: 1.2536276578903198
Epoch 48 losses -> Train: 0.5637103319168091 - Val: 1.2538620233535767
Epoch 49 losses -> Train: 0.5411831736564636 - Val: 1.258304238319397
Epoch 50 losses -> Train: 0.5238615274429321 - Val: 1.2661974430084229
Epoch 51 losses -> Train: 0.5113513469696045 - Val: 1.276475191116333
Epoch 52 losses -> Train: 0.5028504133224487 - Val: 1.2879425287246704
Epoch 53 losses -> Train: 0.49732574820518494 - Val:
1.2994134426116943
Epoch 54 losses -> Train: 0.49367621541023254 - Val:
1.3098233938217163
Epoch 55 losses -> Train: 0.490866094827652 - Val: 1.3183258771896362
Epoch 56 losses -> Train: 0.4880201518535614 - Val: 1.3243284225463867
Epoch 57 losses -> Train: 0.48448172211647034 - Val:
1.3275017738342285
Epoch 58 losses -> Train: 0.47983527183532715 - Val:
1.3277901411056519
Epoch 59 losses -> Train: 0.4739019274711609 - Val: 1.3253587484359741
Epoch 60 losses -> Train: 0.46671268343925476 - Val:
1.3205503225326538
Epoch 61 losses -> Train: 0.4584667682647705 - Val: 1.3138548135757446
Epoch 62 losses -> Train: 0.4494816064834595 - Val: 1.3058332204818726
Epoch 63 losses -> Train: 0.4401389956474304 - Val: 1.2970666885375977
Epoch 64 losses -> Train: 0.43083298206329346 - Val:
1.2881174087524414
Epoch 65 losses -> Train: 0.4219233989715576 - Val: 1.2794862985610962
Epoch 66 losses -> Train: 0.41369807720184326 - Val:
1.2715705633163452
Epoch 67 losses -> Train: 0.40634819865226746 - Val:
1.2646608352661133
Epoch 68 losses -> Train: 0.39995667338371277 - Val:
1.2589293718338013
Epoch 69 losses -> Train: 0.3945010304450989 - Val: 1.2544469833374023
Epoch 70 losses -> Train: 0.38986897468566895 - Val:
1.2511823177337646
Epoch 71 losses -> Train: 0.38588348031044006 - Val:
1.2490440607070923
Epoch 72 losses -> Train: 0.3823337256908417 - Val: 1.2479125261306763
Epoch 73 losses -> Train: 0.37900710105895996 - Val:
1.2476345300674438
Epoch 74 losses -> Train: 0.37571680545806885 - Val:
1.2480827569961548
Epoch 75 losses -> Train: 0.37232232093811035 - Val:
1.2491405010223389
Epoch 76 losses -> Train: 0.3687405586242676 - Val: 1.2507213354110718
Epoch 77 losses -> Train: 0.3649468719959259 - Val: 1.2527599334716797
Epoch 78 losses -> Train: 0.36096692085266113 - Val:
1.2552107572555542
Epoch 79 losses -> Train: 0.35686376690864563 - Val:
1.2580344676971436
```

```
Epoch 80 losses -> Train: 0.35272032022476196 - Val:
1.2611912488937378
Epoch 81 losses -> Train: 0.34862327575683594 - Val:
1.2646301984786987
Epoch 82 losses -> Train: 0.3446485102176666 - Val: 1.268286943435669
Epoch 83 losses -> Train: 0.34085172414779663 - Val:
1.2720859050750732
Epoch 84 losses -> Train: 0.3372628688812256 - Val: 1.2759325504302979
Epoch 85 losses -> Train: 0.33388662338256836 - Val: 1.279735803604126
Epoch 86 losses -> Train: 0.3307056725025177 - Val: 1.28339684009552
Epoch 87 losses -> Train: 0.32768696546554565 - Val:
1.2868324518203735
Epoch 88 losses -> Train: 0.3247891366481781 - Val: 1.2899707555770874
Epoch 89 losses -> Train: 0.32196974754333496 - Val:
1.2927628755569458
Epoch 90 losses -> Train: 0.3191913962364197 - Val: 1.2951781749725342
Epoch 91 losses -> Train: 0.3164256513118744 - Val: 1.2972261905670166
Epoch 92 losses -> Train: 0.3136555850505829 - Val: 1.2989082336425781
Epoch 93 losses -> Train: 0.3108755946159363 - Val: 1.300285816192627
Epoch 94 losses -> Train: 0.30809006094932556 - Val:
1.3013981580734253
Epoch 95 losses -> Train: 0.30531036853790283 - Val:
1.3023139238357544
Epoch 96 losses -> Train: 0.3025519549846649 - Val: 1.303104043006897
Epoch 97 losses -> Train: 0.2998308539390564 - Val: 1.3038328886032104
Epoch 98 losses -> Train: 0.29716089367866516 - Val:
1.3045567274093628
Epoch 99 losses -> Train: 0.2945517897605896 - Val: 1.3053418397903442
Epoch 100 losses -> Train: 0.2920081615447998 - Val:
1.3062175512313843
...completed training

Training pytorch-nonlinear-simple model...

C:\Users\nicol\AppData\Local\Temp\ipykernel_31680\502854005.py:18:
UserWarning: Creating a tensor from a list of numpy.ndarrays is
extremely slow. Please consider converting the list to a single
numpy.ndarray with numpy.array() before converting to a tensor.
(Triggered internally at ..\torch\csrc\utils\tensor_new.cpp:278.)
  return torch.tensor(U_one_hots),torch.tensor(V_one_hots),
torch.tensor(R_targets, dtype=torch.float32).unsqueeze(1)

Epoch 1 losses -> Train: 16.634174346923828 - Val: 11.569181442260742
Epoch 2 losses -> Train: 11.564326286315918 - Val: 8.163403511047363
Epoch 3 losses -> Train: 8.160655975341797 - Val: 5.873081684112549
Epoch 4 losses -> Train: 5.872056484222412 - Val: 4.332304954528809
Epoch 5 losses -> Train: 4.332691669464111 - Val: 3.296081781387329
Epoch 6 losses -> Train: 3.297624349594116 - Val: 2.599635601043701
Epoch 7 losses -> Train: 2.602125406265259 - Val: 2.1319236755371094
Epoch 8 losses -> Train: 2.1351890563964844 - Val: 1.818076491355896
Epoch 9 losses -> Train: 1.8219767808914185 - Val: 1.6076371669769287
Epoch 10 losses -> Train: 1.6120574474334717 - Val: 1.4666297435760498
Epoch 11 losses -> Train: 1.4714754819869995 - Val: 1.372199535369873
Epoch 12 losses -> Train: 1.3773932456970215 - Val: 1.3089895248413086
Epoch 13 losses -> Train: 1.3144677877426147 - Val: 1.266692042350769
Epoch 14 losses -> Train: 1.2724030017852783 - Val: 1.2383943796157837
Epoch 15 losses -> Train: 1.2442954778671265 - Val: 1.2194643020629883
Epoch 16 losses -> Train: 1.2255208492279053 - Val: 1.2068005800247192
Epoch 17 losses -> Train: 1.2129840850830078 - Val: 1.1983273029327393
Epoch 18 losses -> Train: 1.2046146392822266 - Val: 1.1926562786102295
Epoch 19 losses -> Train: 1.1990283727645874 - Val: 1.1888588666915894
Epoch 20 losses -> Train: 1.1953002214431763 - Val: 1.1863142251968384
Epoch 21 losses -> Train: 1.1928123235702515 - Val: 1.1846081018447876
Epoch 22 losses -> Train: 1.1911522150039673 - Val: 1.1834625005722046
Epoch 23 losses -> Train: 1.1900444030761719 - Val: 1.1826924085617065
Epoch 24 losses -> Train: 1.189305067062378 - Val: 1.1821736097335815
Epoch 25 losses -> Train: 1.1888116598129272 - Val: 1.181823492050171
Epoch 26 losses -> Train: 1.1884820461273193 - Val: 1.181586503982544
Epoch 27 losses -> Train: 1.188261866569519 - Val: 1.1814255714416504
Epoch 28 losses -> Train: 1.1881145238876343 - Val: 1.1813156604766846
Epoch 29 losses -> Train: 1.1880159378051758 - Val: 1.1812403202056885
Epoch 30 losses -> Train: 1.1879496574401855 - Val: 1.1811882257461548
Epoch 31 losses -> Train: 1.1879050731658936 - Val: 1.181152105331421
Epoch 32 losses -> Train: 1.1878750324249268 - Val: 1.181126594543457
Epoch 33 losses -> Train: 1.187854528427124 - Val: 1.1811084747314453
Epoch 34 losses -> Train: 1.1878403425216675 - Val: 1.1810953617095947
Epoch 35 losses -> Train: 1.1878304481506348 - Val: 1.1810855865478516
Epoch 36 losses -> Train: 1.1878236532211304 - Val: 1.1810781955718994
Epoch 37 losses -> Train: 1.1878184080123901 - Val: 1.1810725927352905
Epoch 38 losses -> Train: 1.1878145933151245 - Val: 1.1810681819915771
Epoch 39 losses -> Train: 1.1878117322921753 - Val: 1.1810647249221802
Epoch 40 losses -> Train: 1.1878093481063843 - Val: 1.1810616254806519
Epoch 41 losses -> Train: 1.1878072023391724 - Val: 1.1810591220855713
Epoch 42 losses -> Train: 1.187805414199829 - Val: 1.1810567378997803
Epoch 43 losses -> Train: 1.1878037452697754 - Val: 1.1810545921325684
Epoch 44 losses -> Train: 1.1878023147583008 - Val: 1.181052803993225
Epoch 45 losses -> Train: 1.1878008842468262 - Val: 1.1810511350631714
Epoch 46 losses -> Train: 1.1877994537353516 - Val: 1.1810494661331177
Epoch 47 losses -> Train: 1.1877981424331665 - Val: 1.1810479164123535
Epoch 48 losses -> Train: 1.187796711921692 - Val: 1.181046485900879
Epoch 49 losses -> Train: 1.1877954006195068 - Val: 1.1810448169708252
Epoch 50 losses -> Train: 1.1877940893173218 - Val: 1.1810436248779297
Epoch 51 losses -> Train: 1.1877927780151367 - Val: 1.1810420751571655
Epoch 52 losses -> Train: 1.1877915859222412 - Val: 1.181040644645691
Epoch 53 losses -> Train: 1.1877903938293457 - Val: 1.1810393333435059
Epoch 54 losses -> Train: 1.1877890825271606 - Val: 1.1810379028320312
Epoch 55 losses -> Train: 1.1877877712249756 - Val: 1.1810367107391357
Epoch 56 losses -> Train: 1.1877864599227905 - Val: 1.1810352802276611
Epoch 57 losses -> Train: 1.1877851486206055 - Val: 1.1810338497161865
Epoch 58 losses -> Train: 1.18778395652771 - Val: 1.1810328960418701
Epoch 59 losses -> Train: 1.187782645225525 - Val: 1.181031584739685
Epoch 60 losses -> Train: 1.1877813339233398 - Val: 1.181030035018921
Epoch 61 losses -> Train: 1.1877800226211548 - Val: 1.181028962135315
Epoch 62 losses -> Train: 1.1877788305282593 - Val: 1.1810276508331299
Epoch 63 losses -> Train: 1.1877774000167847 - Val: 1.1810263395309448
Epoch 64 losses -> Train: 1.1877762079238892 - Val: 1.1810250282287598
Epoch 65 losses -> Train: 1.187774896621704 - Val: 1.1810237169265747
Epoch 66 losses -> Train: 1.187773585319519 - Val: 1.1810224056243896
Epoch 67 losses -> Train: 1.187772274017334 - Val: 1.1810210943222046
Epoch 68 losses -> Train: 1.187770962715149 - Val: 1.1810197830200195
Epoch 69 losses -> Train: 1.1877696514129639 - Val: 1.181018590927124
Epoch 70 losses -> Train: 1.187768578529358 - Val: 1.181017279624939
Epoch 71 losses -> Train: 1.1877671480178833 - Val: 1.181015968322754
Epoch 72 losses -> Train: 1.1877659559249878 - Val: 1.1810146570205688
Epoch 73 losses -> Train: 1.1877646446228027 - Val: 1.1810133457183838
Epoch 74 losses -> Train: 1.1877633333206177 - Val: 1.1810121536254883
Epoch 75 losses -> Train: 1.1877621412277222 - Val: 1.1810108423233032
Epoch 76 losses -> Train: 1.1877607107162476 - Val: 1.1810095310211182
Epoch 77 losses -> Train: 1.1877593994140625 - Val: 1.1810083389282227
Epoch 78 losses -> Train: 1.187758207321167 - Val: 1.181006908416748
Epoch 79 losses -> Train: 1.187756896018982 - Val: 1.181005597114563
Epoch 80 losses -> Train: 1.1877555847167969 - Val: 1.1810044050216675
Epoch 81 losses -> Train: 1.1877542734146118 - Val: 1.1810029745101929
Epoch 82 losses -> Train: 1.1877530813217163 - Val: 1.1810017824172974
Epoch 83 losses -> Train: 1.1877516508102417 - Val: 1.1810007095336914
Epoch 84 losses -> Train: 1.1877505779266357 - Val: 1.1809992790222168
Epoch 85 losses -> Train: 1.1877491474151611 - Val: 1.1809979677200317
Epoch 86 losses -> Train: 1.187747836112976 - Val: 1.1809967756271362
Epoch 87 losses -> Train: 1.187746524810791 - Val: 1.1809955835342407
Epoch 88 losses -> Train: 1.187745213508606 - Val: 1.1809941530227661
Epoch 89 losses -> Train: 1.187743902206421 - Val: 1.180992841720581
Epoch 90 losses -> Train: 1.1877427101135254 - Val: 1.180991530418396
Epoch 91 losses -> Train: 1.1877413988113403 - Val: 1.1809903383255005
Epoch 92 losses -> Train: 1.1877402067184448 - Val: 1.1809890270233154
Epoch 93 losses -> Train: 1.1877388954162598 - Val: 1.1809877157211304
Epoch 94 losses -> Train: 1.1877375841140747 - Val: 1.1809864044189453
Epoch 95 losses -> Train: 1.1877362728118896 - Val: 1.1809850931167603
Epoch 96 losses -> Train: 1.187734842300415 - Val: 1.1809837818145752
Epoch 97 losses -> Train: 1.1877336502075195 - Val: 1.1809824705123901
Epoch 98 losses -> Train: 1.1877323389053345 - Val: 1.1809812784194946
Epoch 99 losses -> Train: 1.187731146812439 - Val: 1.1809799671173096
Epoch 100 losses -> Train: 1.187729835510254 - Val: 1.1809786558151245
...completed training

Training pytorch-nonlinear-complex model...
Epoch 1 losses -> Train: 10.123239517211914 - Val: 9.059609413146973
Epoch 2 losses -> Train: 9.092995643615723 - Val: 8.082199096679688
Epoch 3 losses -> Train: 8.12230396270752 - Val: 7.1622395515441895
Epoch 4 losses -> Train: 7.218463897705078 - Val: 6.300007343292236
Epoch 5 losses -> Train: 6.363177299499512 - Val: 5.497217178344727
Epoch 6 losses -> Train: 5.5761799812316895 - Val: 4.757542610168457
Epoch 7 losses -> Train: 4.8454389572143555 - Val: 4.08626127243042
Epoch 8 losses -> Train: 4.19747161865234 - Val: 3.4894371032714844
Epoch 9 losses -> Train: 3.6149024963378906 - Val: 2.972163677215576
Epoch 10 losses -> Train: 3.1090359687805176 - Val: 2.5363948345184326
Epoch 11 losses -> Train: 2.6899149417877197 - Val: 2.179262638092041
Epoch 12 losses -> Train: 2.3510191440582275 - Val: 1.8933508396148682
Epoch 13 losses -> Train: 2.079758882522583 - Val: 1.6689093112945557
Epoch 14 losses -> Train: 1.8753385543823242 - Val: 1.4963444471359253
Epoch 15 losses -> Train: 1.7222723960876465 - Val: 1.3677945137023926
Epoch 16 losses -> Train: 1.604044795036316 - Val: 1.2770963907241821
Epoch 17 losses -> Train: 1.5210999250411987 - Val: 1.219011902809143
Epoch 18 losses -> Train: 1.488373875617981 - Val: 1.1883502006530762
Epoch 19 losses -> Train: 1.4739450216293335 - Val: 1.1795973777770996
Epoch 20 losses -> Train: 1.4722293615341187 - Val: 1.186952829360962
Epoch 21 losses -> Train: 1.490157127380371 - Val: 1.2047960758209229
Epoch 22 losses -> Train: 1.5093553066253662 - Val: 1.2280628681182861
Epoch 23 losses -> Train: 1.5488437414169312 - Val: 1.2524784803390503
Epoch 24 losses -> Train: 1.578831434249878 - Val: 1.2748390436172485
Epoch 25 losses -> Train: 1.6142475605010986 - Val: 1.2928191423416138
Epoch 26 losses -> Train: 1.6252751350402832 - Val: 1.3050169944763184
Epoch 27 losses -> Train: 1.6576602458953857 - Val: 1.3107404708862305
Epoch 28 losses -> Train: 1.649491786956787 - Val: 1.3101476430892944
Epoch 29 losses -> Train: 1.6628367900848389 - Val: 1.3037785291671753
Epoch 30 losses -> Train: 1.655250072479248 - Val: 1.29256165027618 4
Epoch 31 losses -> Train: 1.6251543760299683 - Val: 1.277818202972412
Epoch 32 losses -> Train: 1.6149629354476929 - Val: 1.2610124349594116
Epoch 33 losses -> Train: 1.596992015838623 - Val: 1.243543028831482
Epoch 34 losses -> Train: 1.5685124397277832 - Val: 1.2267464399337769
Epoch 35 losses -> Train: 1.5521509647369385 - Val: 1.211766004562378
Epoch 36 losses -> Train: 1.533454418182373 - Val: 1.199381709098816
Epoch 37 losses -> Train: 1.5169060230255127 - Val: 1.190057635307312
Epoch 38 losses -> Train: 1.4941328763961792 - Val: 1.1839408874511719
Epoch 39 losses -> Train: 1.4853458404541016 - Val: 1.1809120178222656
Epoch 40 losses -> Train: 1.4809229373931885 - Val: 1.180625557899475
Epoch 41 losses -> Train: 1.4704326391220093 - Val: 1.1825964450836182
Epoch 42 losses -> Train: 1.4706604480743408 - Val: 1.1862640380859375
Epoch 43 losses -> Train: 1.467061161994934 - Val: 1.1910642385482788
Epoch 44 losses -> Train: 1.4745358228683472 - Val: 1.1964370012283325
Epoch 45 losses -> Train: 1.4644503593444824 - Val: 1.2018755674362183
Epoch 46 losses -> Train: 1.4749246835708618 - Val: 1.2069714069366455
Epoch 47 losses -> Train: 1.4797580242156982 - Val: 1.2113440036773682
Epoch 48 losses -> Train: 1.4750237464904785 - Val: 1.2147297859191895
Epoch 49 losses -> Train: 1.4767494201660156 - Val: 1.2170026302337646
Epoch 50 losses -> Train: 1.4793291091918945 - Val: 1.218069314956665
Epoch 51 losses -> Train: 1.4781426191329956 - Val: 1.2179697751998901
Epoch 52 losses -> Train: 1.4739712476730347 - Val: 1.216796636581421
Epoch 53 losses -> Train: 1.4706014394760132 - Val: 1.21471107006073
Epoch 54 losses -> Train: 1.4664250612258911 - Val: 1.2118635177612305
Epoch 55 losses -> Train: 1.470912218093872 - Val: 1.2084357738494873
Epoch 56 losses -> Train: 1.4610079526901245 - Val: 1.2045745849609375
Epoch 57 losses -> Train: 1.4592700004577637 - Val: 1.2004947662353516
Epoch 58 losses -> Train: 1.4655358791351318 - Val: 1.1963443756103516
Epoch 59 losses -> Train: 1.4471285343170166 - Val: 1.1922701597213745
Epoch 60 losses -> Train: 1.455644130706787 - Val: 1.188331961631775
Epoch 61 losses -> Train: 1.4433385133743286 - Val: 1.1845773458480835
Epoch 62 losses -> Train: 1.44120192527771 - Val: 1.1810359954833984
Epoch 63 losses -> Train: 1.431127905845642 - Val: 1.1776801347732544
Epoch 64 losses -> Train: 1.4284111261367798 - Val: 1.1744763851165771
Epoch 65 losses -> Train: 1.4315441846847534 - Val: 1.171376347541809
Epoch 66 losses -> Train: 1.4276727437973022 - Val: 1.1683112382888794
Epoch 67 losses -> Train: 1.418640375137329 - Val: 1.1652095317840576
Epoch 68 losses -> Train: 1.401877760887146 - Val: 1.162005066871643
Epoch 69 losses -> Train: 1.4202134609222412 - Val: 1.1586424112319946
Epoch 70 losses -> Train: 1.39784836769104 - Val: 1.1550495624542236
Epoch 71 losses -> Train: 1.3987786769866943 - Val: 1.151172399520874
Epoch 72 losses -> Train: 1.3937678337097168 - Val: 1.1469477415084839
Epoch 73 losses -> Train: 1.3846646547317505 - Val: 1.1423163414001465
Epoch 74 losses -> Train: 1.3798235654830933 - Val: 1.1372193098068237
Epoch 75 losses -> Train: 1.3783149719238281 - Val: 1.131575345993042
Epoch 76 losses -> Train: 1.3546894788742065 - Val: 1.125305414199829
Epoch 77 losses -> Train: 1.3577027320861816 - Val: 1.1183682680130005
Epoch 78 losses -> Train: 1.3433163166046143 - Val: 1.1107653379440308
Epoch 79 losses -> Train: 1.3290127515792847 - Val: 1.102555394172668 5
Epoch 80 losses -> Train: 1.3001911640167236 - Val: 1.0938609838485718
Epoch 81 losses -> Train: 1.3021912574768066 - Val: 1.0848407745361328
Epoch 82 losses -> Train: 1.2872307300567627 - Val: 1.0757887363433838
Epoch 83 losses -> Train: 1.2855801582336426 - Val: 1.0670446157455444
Epoch 84 losses -> Train: 1.265789270401001 - Val: 1.058919906616211
Epoch 85 losses -> Train: 1.254834771156311 - Val: 1.0516152381896973
Epoch 86 losses -> Train: 1.24838125705719 - Val: 1.0451323986053467
Epoch 87 losses -> Train: 1.2394007444381714 - Val: 1.0392409563064575
Epoch 88 losses -> Train: 1.2233494520187378 - Val: 1.033652901649475
Epoch 89 losses -> Train: 1.217195987701416 - Val: 1.0281327962875366
Epoch 90 losses -> Train: 1.203571081161499 - Val: 1.0224930047988892
Epoch 91 losses -> Train: 1.1906851530075073 - Val: 1.016654372215271
Epoch 92 losses -> Train: 1.1864057779312134 - Val: 1.010629415512085
Epoch 93 losses -> Train: 1.17791140079498 3 - Val: 1.0044713020324707
Epoch 94 losses -> Train: 1.1688088178634644 - Val: 0.9982031583786011
Epoch 95 losses -> Train: 1.1630206108093262 - Val: 0.9918107390403748
Epoch 96 losses -> Train: 1.1495602130889893 - Val: 0.9852863550186157
Epoch 97 losses -> Train: 1.1364150047302246 - Val: 0.9786583781242371
Epoch 98 losses -> Train: 1.1301612854003906 - Val: 0.9719467759132385
Epoch 99 losses -> Train: 1.111933708190918 - Val: 0.965254008769989
Epoch 100 losses -> Train: 1.110291600227356 - Val: 0.958721935749054
```

```
...completed training
```

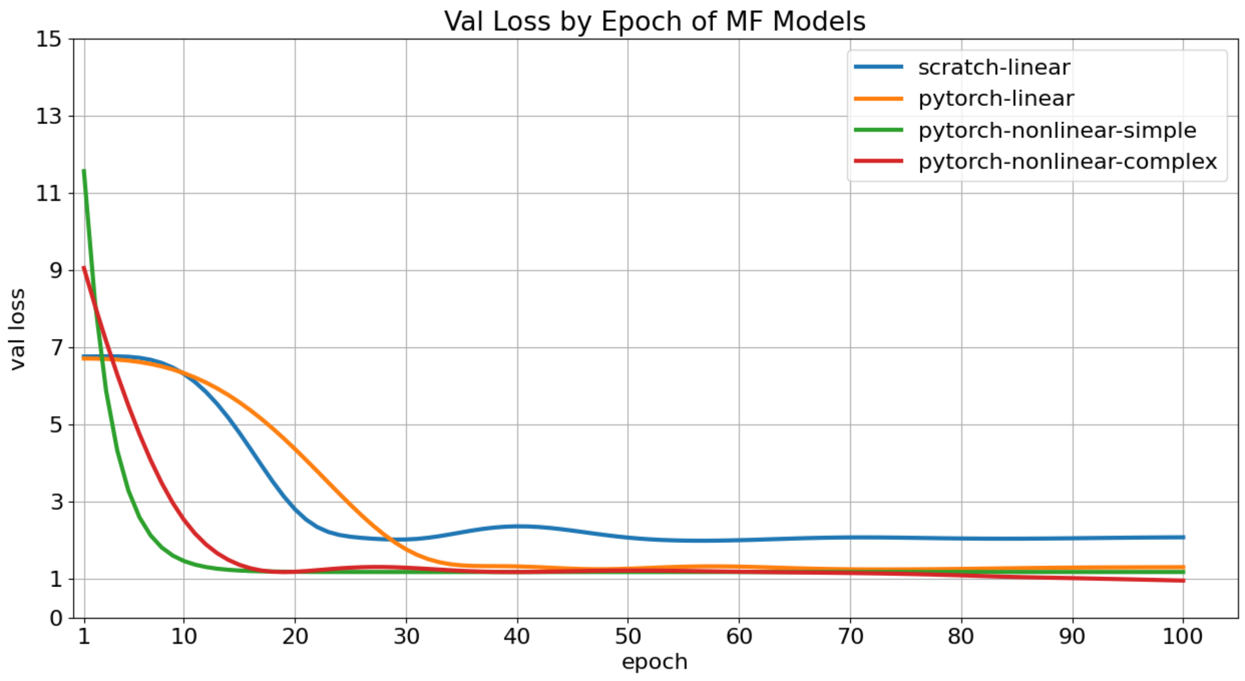
```
def plot_performance(performance_dict, save_file='performance.png'):
    fig, ax = plt.subplots()
    for model_name, (train_losses, val_losses) in 
list(performance_dict.items()):
        epochs = list(range(1, len(train_losses)+1))
        ax.plot(epochs, val_losses, label=model_name, linewidth=3)
    plt.xlabel('epoch')
    plt.ylabel('val loss')
    plt.xlim(0)
    plt.ylim(0, 15)

    plt.xticks([1] + list(range(10, 100, 10))+[100])

    plt.yticks([0, 1, 3, 5, 7, 9, 11, 13, 15])

    plt.legend()
    plt.grid()
    plt.title('Val Loss by Epoch of MF Models')
    fig.savefig(save_file)
    plt.show()

plot_performance(performance_dict)
```


Val Loss by Epoch of MF Models
scratch-linear
pytorch-linear
pytorch-nonlinear-simple
pytorch-nonlinear-complex
val loss
epoch