mlp-for-classification

May 4, 2024

1 MLP for Classification

1.1 Imports

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import torch
from torch import nn, optim
import torch.nn.functional as F
from torch.utils.data import DataLoader, TensorDataset
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

1.2 Load Data

[2]:
```
!cp "/content/drive/MyDrive/colab_projects/MLP for Classification/mobile_price.
 ↪zip" /content
!unzip "/content/drive/MyDrive/colab_projects/MLP for Classification/
 ↪mobile_price.zip"
```

```
Archive:  /content/drive/MyDrive/colab_projects/MLP for
Classification/mobile_price.zip
  inflating: test.csv
  inflating: train.csv
```

[3]:
```
df = pd.read_csv('/content/train.csv')
df.head()
```

[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	\
0	842	0	2.2	0	1	0	7	0.6	
1	1021	1	0.5	1	0	1	53	0.7	
2	563	1	0.5	1	2	1	41	0.9	
3	615	1	2.5	0	0	0	10	0.8	
4	1821	1	1.2	0	13	1	44	0.6	

	mobile_wt	n_cores	…	px_height	px_width	ram	sc_h	sc_w	talk_time	\
0	188	2	…	20	756	2549	9	7	19	
1	136	3	…	905	1988	2631	17	3	7	

```
2         145         5  …        1263       1716  2603     11      2          9
3         131         6  …        1216       1786  2769     16      8         11
4         141         2  …        1208       1212  1411      8      2         15

   three_g  touch_screen  wifi  price_range
0        0             0     1            1
1        1             1     0            2
2        1             1     0            2
3        1             0     0            2
4        1             1     0            1

[5 rows x 21 columns]
```

[4]:
```
X = df.drop('price_range', axis = 1)
```

[5]:
```
y = df['price_range']
```

[6]:
```
x_train, x_valid, y_train, y_valid = train_test_split(X, y, test_size=0.7,␣
↪random_state=42)
```

1.3 Preprocessing

1.3.1 Getting info about df

[7]:
```
df.shape
```

[7]:
```
(2000, 21)
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
```

```
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

1.3.2 Convert to Tensor

Note that since it's a classification problem, torch expects an integer as output and not a float.

[9]:
```
x_train = torch.FloatTensor(x_train.values)
y_train = torch.LongTensor(y_train.values)
```

[10]:
```
x_valid = torch.FloatTensor(x_valid.values)
y_valid = torch.LongTensor(y_valid.values)
```

1.3.3 Standardization

[11]:
```
mu = x_train.mean(dim=0)
std = x_train.std(dim=0)
```

[12]:
```
x_train = (x_train - mu) / std
x_valid = (x_valid - mu) / std
```

1.4 Dataloader

[13]:
```
train_data = TensorDataset(x_train, y_train)
train_loader = DataLoader(train_data, batch_size=100, shuffle=True)
```

[14]:
```
valid_data = TensorDataset(x_valid, y_valid)
valid_loader = DataLoader(valid_data, batch_size=200, shuffle=True)
```

1.5 Model

[15]:
```
num_feats = 20
num_class = 4
h1 = 64
h2 = 32

model = nn.Sequential(nn.Linear(num_feats, h1),
                      nn.ReLU(),
```

```
                      nn.Linear(h1, h2),
                      nn.ReLU(),
                      nn.Linear(h2, num_class))
```

1.6 Device

[16]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
```

[17]:
```
model = model.to(device)
```

1.7 Loss and Optimizer

[18]:
```
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)
```

[19]:
```
class AverageMeter(object):
    """Computes and stores the average and current value"""
    def __init__(self):
        self.reset()

    def reset(self):
        self.val = 0
        self.avg = 0
        self.sum = 0
        self.count = 0

    def update(self, val, n=1):
        self.val = val
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count
```

1.8 Torchmetrics

In order to see how well our model has predicted the corresponding classes of each sample, we can use the accuracy metric in torchmetrics

[20]:
```
!pip install torchmetrics
from torchmetrics import Accuracy
```

```
Collecting torchmetrics
  Downloading torchmetrics-1.3.2-py3-none-any.whl (841 kB)
                           841.5/841.5
kB 6.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>1.20.0 in
/usr/local/lib/python3.10/dist-packages (from torchmetrics) (1.25.2)
```

```
Requirement already satisfied: packaging>17.1 in /usr/local/lib/python3.10/dist-
packages (from torchmetrics) (24.0)
Requirement already satisfied: torch>=1.10.0 in /usr/local/lib/python3.10/dist-
packages (from torchmetrics) (2.2.1+cu121)
Collecting lightning-utilities>=0.8.0 (from torchmetrics)
  Downloading lightning_utilities-0.11.2-py3-none-any.whl (26 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from lightning-utilities>=0.8.0->torchmetrics) (67.7.2)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from lightning-
utilities>=0.8.0->torchmetrics) (4.11.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->torchmetrics) (3.14.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->torchmetrics) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->torchmetrics) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->torchmetrics) (3.1.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->torchmetrics) (2023.6.0)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl
(23.7 MB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl
(823 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl
(14.1 MB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7
MB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6
MB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6
MB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl
(56.5 MB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl
(124.2 MB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl
(196.0 MB)
```

```
Collecting nvidia-nccl-cu12==2.19.3 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl (166.0 MB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch>=1.10.0->torchmetrics)
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->torchmetrics) (2.2.0)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-
cu12==11.4.5.107->torch>=1.10.0->torchmetrics)
  Using cached nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl
(21.1 MB)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from
jinja2->torch>=1.10.0->torchmetrics) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->torch>=1.10.0->torchmetrics) (1.3.0)
Installing collected packages: nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-
nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12,
nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12, lightning-
utilities, nvidia-cusparse-cu12, nvidia-cudnn-cu12, nvidia-cusolver-cu12,
torchmetrics
Successfully installed lightning-utilities-0.11.2 nvidia-cublas-cu12-12.1.3.1
nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.1.105 nvidia-cuda-
runtime-cu12-12.1.105 nvidia-cudnn-cu12-8.9.2.26 nvidia-cufft-cu12-11.0.2.54
nvidia-curand-cu12-10.3.2.106 nvidia-cusolver-cu12-11.4.5.107 nvidia-cusparse-
cu12-12.1.0.106 nvidia-nccl-cu12-2.19.3 nvidia-nvjitlink-cu12-12.4.127 nvidia-
nvtx-cu12-12.1.105 torchmetrics-1.3.2
```

1.9 Model Training

[21]:
```python
num_epochs = 400

loss_train_hist = []
loss_valid_hist = []

acc_train_hist = []
acc_valid_hist = []

for epoch in range(num_epochs):
  loss_train = AverageMeter()
  acc_train = Accuracy(task='multiclass', num_classes=4).to(device)
  for i, (inputs, targets) in enumerate(train_loader):
    inputs = inputs.to(device)
    targets = targets.to(device)

    outputs = model(inputs)

    loss = loss_fn(outputs, targets)
```

```
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()

    loss_train.update(loss.item())
    acc_train(outputs, targets)

  with torch.no_grad():
    loss_valid = AverageMeter()
    acc_valid = Accuracy(task='multiclass', num_classes=4).to(device)
    for i, (inputs, targets) in enumerate(valid_loader):
      inputs = inputs.to(device)
      targets = targets.to(device)

      outputs = model(inputs)
      loss = loss_fn(outputs, targets)

      loss_valid.update(loss.item())
      acc_valid(outputs, targets)

  loss_train_hist.append(loss_train.avg)
  loss_valid_hist.append(loss_valid.avg)

  acc_train_hist.append(acc_train.compute())
  acc_valid_hist.append(acc_valid.compute())

  if epoch % 10 == 0:
    print(f'Epoch {epoch}')
    print(f'Train: Loss = {loss_train.avg:.4}, Acc = {acc_train.compute():.4}')
    print(f'Valid: Loss = {loss_valid.avg:.4}, Acc = {acc_valid.compute():.4}')
    print()
```

```
Epoch 0
Train: Loss = 1.394, Acc = 0.2367
Valid: Loss = 1.394, Acc = 0.2486

Epoch 10
Train: Loss = 1.381, Acc = 0.2517
Valid: Loss = 1.382, Acc = 0.2571

Epoch 20
Train: Loss = 1.369, Acc = 0.2783
Valid: Loss = 1.371, Acc = 0.2779

Epoch 30
Train: Loss = 1.356, Acc = 0.3167
```

```
Valid: Loss = 1.358, Acc = 0.3121

Epoch 40
Train: Loss = 1.34, Acc = 0.39
Valid: Loss = 1.342, Acc = 0.3714

Epoch 50
Train: Loss = 1.32, Acc = 0.46
Valid: Loss = 1.322, Acc = 0.4421

Epoch 60
Train: Loss = 1.293, Acc = 0.4883
Valid: Loss = 1.295, Acc = 0.4793

Epoch 70
Train: Loss = 1.258, Acc = 0.5317
Valid: Loss = 1.26, Acc = 0.5121

Epoch 80
Train: Loss = 1.213, Acc = 0.5467
Valid: Loss = 1.215, Acc = 0.5229

Epoch 90
Train: Loss = 1.157, Acc = 0.5583
Valid: Loss = 1.16, Acc = 0.5329

Epoch 100
Train: Loss = 1.093, Acc = 0.5633
Valid: Loss = 1.096, Acc = 0.5486

Epoch 110
Train: Loss = 1.026, Acc = 0.5883
Valid: Loss = 1.029, Acc = 0.5643

Epoch 120
Train: Loss = 0.9583, Acc = 0.61
Valid: Loss = 0.9624, Acc = 0.5829

Epoch 130
Train: Loss = 0.8943, Acc = 0.6433
Valid: Loss = 0.899, Acc = 0.6129

Epoch 140
Train: Loss = 0.8351, Acc = 0.6733
Valid: Loss = 0.8405, Acc = 0.6586

Epoch 150
Train: Loss = 0.7809, Acc = 0.725
```

```
Valid: Loss = 0.7872, Acc = 0.7

Epoch 160
Train: Loss = 0.7314, Acc = 0.775
Valid: Loss = 0.7387, Acc = 0.7429

Epoch 170
Train: Loss = 0.6859, Acc = 0.8083
Valid: Loss = 0.6946, Acc = 0.7743

Epoch 180
Train: Loss = 0.6443, Acc = 0.8333
Valid: Loss = 0.6546, Acc = 0.8029

Epoch 190
Train: Loss = 0.6062, Acc = 0.8483
Valid: Loss = 0.6183, Acc = 0.8214

Epoch 200
Train: Loss = 0.5711, Acc = 0.87
Valid: Loss = 0.5852, Acc = 0.84

Epoch 210
Train: Loss = 0.5389, Acc = 0.8883
Valid: Loss = 0.555, Acc = 0.8593

Epoch 220
Train: Loss = 0.509, Acc = 0.9033
Valid: Loss = 0.5274, Acc = 0.8686

Epoch 230
Train: Loss = 0.4814, Acc = 0.91
Valid: Loss = 0.502, Acc = 0.8736

Epoch 240
Train: Loss = 0.4558, Acc = 0.9167
Valid: Loss = 0.4787, Acc = 0.8843

Epoch 250
Train: Loss = 0.4319, Acc = 0.9167
Valid: Loss = 0.4574, Acc = 0.8914

Epoch 260
Train: Loss = 0.4097, Acc = 0.92
Valid: Loss = 0.4378, Acc = 0.8986

Epoch 270
Train: Loss = 0.3889, Acc = 0.9233
```

```
Valid: Loss = 0.4197, Acc = 0.9057

Epoch 280
Train: Loss = 0.37, Acc = 0.9317
Valid: Loss = 0.403, Acc = 0.9086

Epoch 290
Train: Loss = 0.3519, Acc = 0.9383
Valid: Loss = 0.3877, Acc = 0.9086

Epoch 300
Train: Loss = 0.335, Acc = 0.9433
Valid: Loss = 0.3734, Acc = 0.9107

Epoch 310
Train: Loss = 0.3193, Acc = 0.9467
Valid: Loss = 0.3604, Acc = 0.9107

Epoch 320
Train: Loss = 0.3045, Acc = 0.9517
Valid: Loss = 0.3482, Acc = 0.9114

Epoch 330
Train: Loss = 0.2907, Acc = 0.9533
Valid: Loss = 0.337, Acc = 0.9129

Epoch 340
Train: Loss = 0.2778, Acc = 0.9567
Valid: Loss = 0.3267, Acc = 0.9114

Epoch 350
Train: Loss = 0.2657, Acc = 0.9617
Valid: Loss = 0.3172, Acc = 0.9114

Epoch 360
Train: Loss = 0.2542, Acc = 0.9633
Valid: Loss = 0.3084, Acc = 0.9136

Epoch 370
Train: Loss = 0.2435, Acc = 0.965
Valid: Loss = 0.3002, Acc = 0.9136

Epoch 380
Train: Loss = 0.2334, Acc = 0.9667
Valid: Loss = 0.2926, Acc = 0.9157

Epoch 390
Train: Loss = 0.2238, Acc = 0.9683
```

```
Valid: Loss = 0.2856, Acc = 0.9157
```

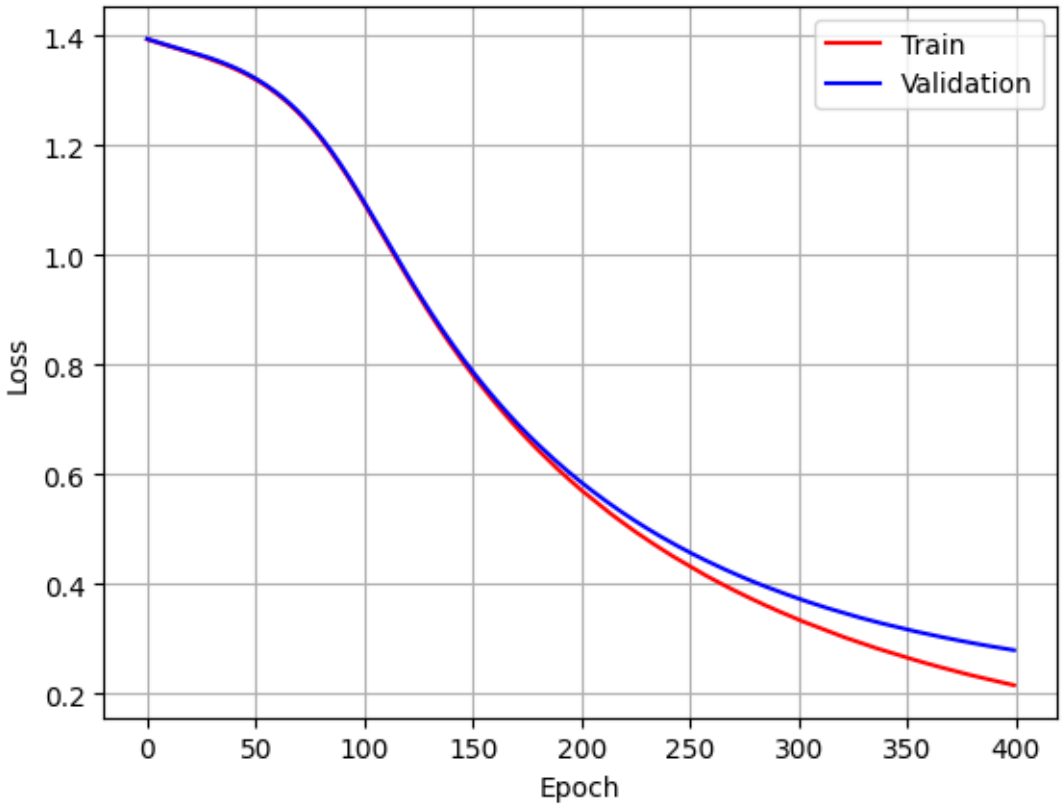
1.10 Evaluation

1.10.1 Loss

[22]:
```
plt.plot(range(num_epochs), loss_train_hist, 'r-', label='Train')
plt.plot(range(num_epochs), loss_valid_hist, 'b-', label='Validation')

plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.grid(True)
plt.legend()
```

[22]:
```
<matplotlib.legend.Legend at 0x7ed8b6051900>
```

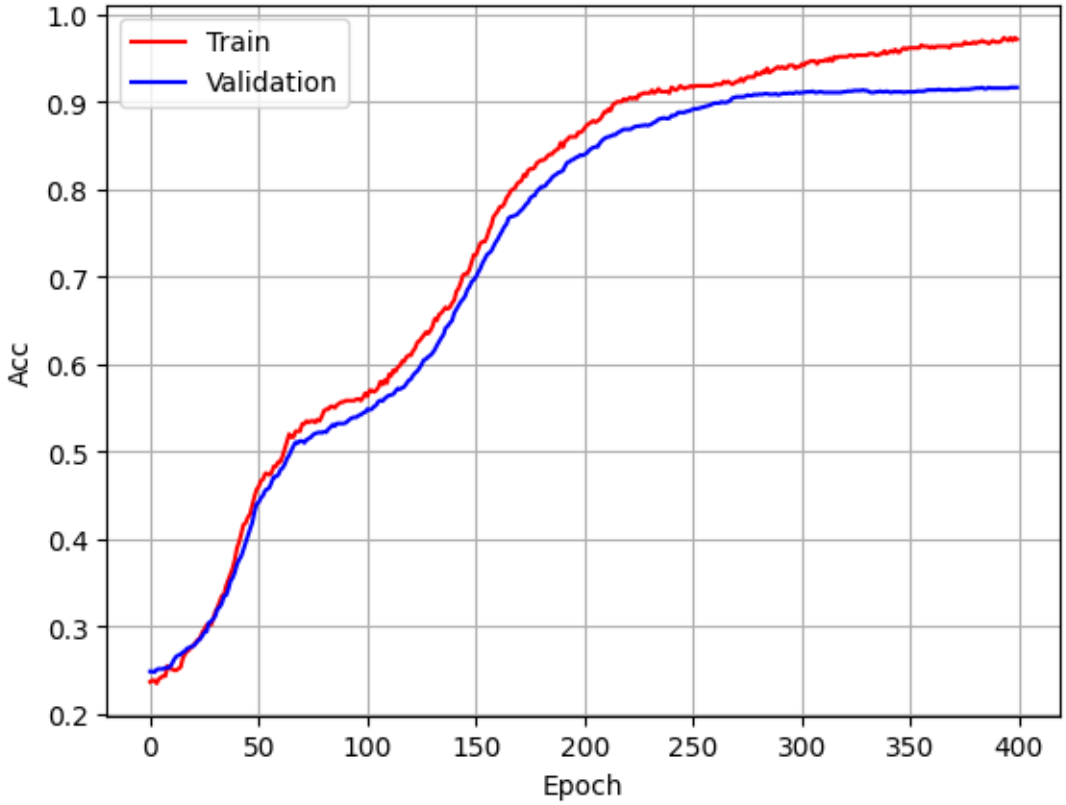


1.10.2 Accuracy

[23]:
```
plt.plot(range(num_epochs), acc_train_hist, 'r-', label='Train')
plt.plot(range(num_epochs), acc_valid_hist, 'b-', label='Validation')

plt.xlabel('Epoch')
plt.ylabel('Acc')
plt.grid(True)
plt.legend()
```

[23]:
```
<matplotlib.legend.Legend at 0x7ed8b6052e60>
```



1.11 Save Model

[24]:
```
torch.save(model, 'model.pth')
```

[25]:
```
mymodel = torch.load('model.pth')
```