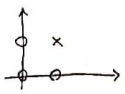
به نام خدا

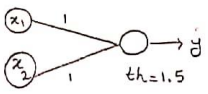
تمرین چهارم یادگیری ماشین

غزل زمانی‌نژاد

. .1

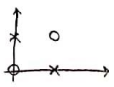
۱) معماری شبکه برای AND : مطابق تصویر، داده‌های linearly separable هستند و می‌توان با یک perceptron ساده آن‌ها را از هم جدا کرد.

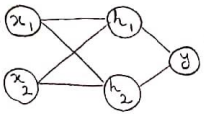




لایه ورودی 2 نورون
لایه خروجی 1 نورون

معماری شبکه XOR : این داده‌ها به صورت خطی تفکیک‌پذیر نیستند. باید یک لایه مخفی اضافه کنیم تا بتوانیم طبقه‌بندی را انجام دهیم.





لایه ورودی 2 نورون
لایه میانی 2 نورون
لایه خروجی 1 نورون

- الگوریتم backpropagation برنامه بهینه کردن وزن‌های میان نورون‌هاست. در هر مرحله خطای پیش‌بینی نسبت به مقدار واقعی محاسبه می‌شود و خطا به عقب منتشر می‌شود تا وزن‌ها به سمت مقدار بهینه، آپدیت شوند.
- در ابتدا وزن‌ها به صورت تصادفی مقداردهی می‌شوند. البته می‌توانیم از روش‌هایی مثل He یا Xavier هم استفاده کنیم که وزن‌ها را جوری مقداردهی می‌کنند که از توزیع نرمال پیروی کند. بدین ترتیب از vanishing/exploding gradients تا حدی جلوگیری می‌شود.
- در یک epoch :

۱) در forward pass ، خروجی شبکه را بر اساس مقادیر فعلی وزن‌ها محاسبه می‌کنیم

2) بر اساس loss function تعیین شده، میزان خطا بین خروجی و مقدار واقعی را بدست می‌آوریم.

3) گرادیان خطا نسبت به وزن‌ها را حساب می‌کنیم و بر اساس فرمول $w_i = w_i - \alpha \frac{\partial L}{\partial w_i}$ وزن‌ها را آپدیت می‌کنیم

تا رسیدن به convergence ، مراحل ۱ تا 3 را تکرار می‌کنیم.

2. الف) استراتژی: توپ‌ها را در ۳ گروه تقسیم می‌کنیم. دو گروه را با ترازو مقایسه می‌کنیم: اگر وزن این دو گروه برابر باشد، توپ سنگین در گروه سوم است. در غیر این صورت توپ متفاوت در کفه سنگین‌تر ترازو است. سپس باید ۳ توپی که از گروه سنگین مانده را مقایسه کنیم که در اینجا هم مشابه قبل عمل می‌کنیم. دو توپ را روی ترازو می‌گذاریم: اگر وزن‌شان برابر باشد، توپ سوم توپ سنگین است و در غیر این صورت کفه سنگین‌تر نشان‌دهنده توپ متفاوت است. در هر بار مقایسه ۳ حالت ممکن است اتفاق بیفتد: وزن دو کفه برابر باشد، کفه راست سنگین‌تر باشد، کفه چپ سنگین‌تر باشد که می‌توانیم آن را کد k-nary با $k=3$ در نظر بگیریم. از آنجایی که ۲ آزمایش نیاز داریم پس در اینجا $l_i=2$ است پس

$P(x=i) = 3 \wedge -2 = 1/9$

که برابر با احتمال خروج ۱ توپ از میان ۹ توپ است. سپس آنتروپی را محاسبه می‌کنیم:

$$-\sum_{i=1}^{9} \frac{1}{9} \log_3 \frac{1}{9} = 2$$

پس حداقل آزمایش مورد نیاز ۲ است و با آنتروپی آن را اثبات کردیم.

ب) در آزمایش اول، از مقایسه هر دو گروه توپ سه‌تایی، به این میزان gain می‌رسیم:

$H(X) - H(X|Y1) = 2 - 1 = 1$

که این بیشترین مقداریست زمانی که خروجی سه حالت می‌تواند داشته باشد. در آزمایش دوم، مقایسه هر دو توپ از گروه سه‌تایی به میزان

$H(X|Y1) - H(X|Y2, Y1) = 1 - 0 = 1$

اطلاعات دارد. بنابراین ID3 به درخت تصمیم بهینه خواهد رسید.

3) الف) total entropy $= -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} \simeq 0.811$

feature: Genre = {Drama, Horror}

$e(S_{Drama}) = -\frac{3}{3}\log\frac{3}{3} = 0$

$e(S_{Horror}) = -1\log 1 = 0$

$\rightarrow gain(S, Genre) = 0.811 - 0 = \boxed{0.811}$

feature: Age Rating = {PG-13, G}

$e(S_{PG-13}) = -\frac{3}{3}\log\frac{3}{3} = 0$

$e(S_G) = -1\log 1 = 0$

$gain(S, Age) = \boxed{0.811}$

feature: Language = {French, English}

$e(S_{French}) = -1\log 1 = 0$

$e(S_{English}) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} \simeq 0.923$

$gain(S, Language) = 0.811 - \frac{3}{4}.0923 = \boxed{0.118}$

feature: Source = {Book Adaptation}

$e(S_{Book}) = -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} \simeq 0.811 \rightarrow gain(S, Source) = 0.811 - 0.811 = \boxed{0}$

feature: Film Location = {Canada, USA}

$e(S_{Canada}) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} \simeq 0.923$

$e(S_{USA}) = -1\log 1 = 0$

$gain(S, Location) = 0.811 - \frac{3}{4} \times 0.923 = \boxed{0.118}$

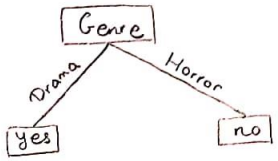
feature: Studio = {Warner, A24}

$e(S_{Warner}) = -\frac{2}{2}\log\frac{2}{2} = 0$

$e(S_{A24}) = \frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$

$gain(S, Studio) = 0.811 - \frac{1}{2}.1 = \boxed{0.311}$

در این مرحله بیشترین gain با Genre، Genre ← را انتخاب می‌کنیم. بین Age Rating و



ب) total entropy $= \frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} \simeq 0.97$

Feature: Genre $= \{$ Drama, Horror $\}$

$e(S_{Drama}) = -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} \simeq 0.811$

$e(S_{Horror}) = -1\log 1 = 0$

$gain(S, Genre) = 0.97 - \frac{4}{5}.0.811 \simeq \boxed{0.321}$

Feature: Age Rating $= \{$ PG-13, G $\}$

$e(S_{PG-13}) = -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} \simeq 0.811$

$e(S_{Horror}) = -1\log 1 = 0$

$gain(S, Age) = 0.97 - \frac{4}{5} \times 0.811 \simeq \boxed{0.321}$

Feature: Language $= \{$ French, English $\}$

$e(S_{French}) = \frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$

$e(S_{English}) = \frac{-2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} \simeq 0.923$

$gain(S, Language) = 0.97 - \frac{2}{5} \times 1 - \frac{3}{5} \times 0.923 \simeq \boxed{0.016}$

Feature: Source $= \{$ Book Adaptation, Original Screenplay $\}$

$e(S_{Book}) = -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} \simeq 0.811$

$e(S_{screenplay}) = -1\log 1 = 0$

$gain(S, Source) = 0.97 - \frac{4}{5} \times 0.811 \simeq \boxed{0.321}$

Feature: Film Location $= \{$ Canada, USA $\}$

$e(S_{Canada}) = -\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4} = 1$

$e(S_{USA}) = -1\log 1 = 0$

$gain(S, Location) = 0.97 - \frac{4}{5} \cdot 1 = \boxed{0.17}$

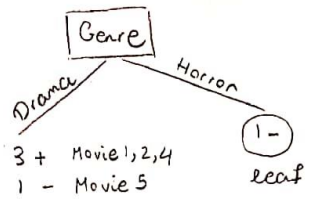
Feature: Studio $= \{$ Warner, A24 $\}$

$e(S_{warner}) = \frac{-2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} \simeq 0.923$

$e(S_{A24}) = \frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$

$gain(S, Studio) = 0.97 - \frac{3}{5} \times 0.923 - \frac{2}{5} \times 1 = \boxed{0.016}$

بیشترین gain را ۳ ویژگی: Genre، Age، Source دارند؛ Genre را انتخاب می‌کنیم



Movie 1, 2, 4, 5

⊕	1	PG-13	French	Book	Canada Warner
⊕	2	PG-13	English	Book	Canada Warner
⊕	4	PG-13	English	Book	Canada A24
⊖	5	PG-13	French	Screenplay	Canada Warner

total entropy $= \frac{-3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} \simeq 0.811$

Feature: Age = {PG-13}

$e(S_{PG-13}) = \frac{-3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} \simeq 0.811 \rightarrow$ gain(S, Age) =

$0.811 - 0.811 = \boxed{0}$

Feature: Language = {English, French}

$e(S_{English}) = \frac{-3}{3}\log\frac{3}{3} = 0$

$e(S_{French}) = -1\log 1 = 0$

} gain(S, Language) = 0.811

Feature: Source = {Book, Screenplay}

$e(S_{Book}) = \frac{-3}{3}\log\frac{3}{3} = 0$

$e(S_{screenplay}) = -1\log 1 = 0$

} gain(S, Source) = 0.811

Feature: Location = {canada}

$e(S_{canada}) = \frac{-3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} = 0.811 \rightarrow$ gain(S, Location) = 0

Feature: Studio = {Warner, A24}

$e(S_{warner}) = \frac{-2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} \simeq 0.923$

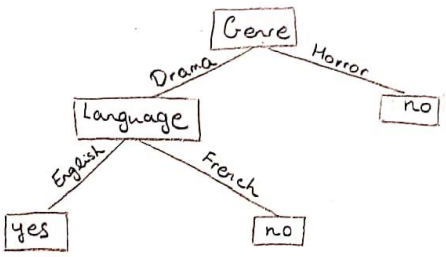
$e(S_{A24}) = -1\log 1 = 0$

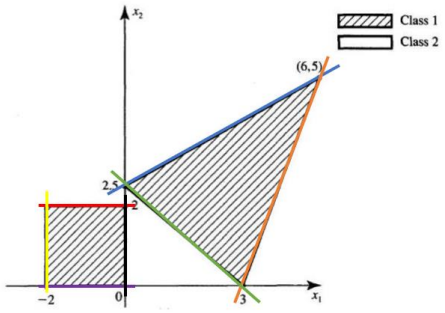
} gain(S, Studio) =

$0.811 - \frac{3}{4} \times 0.923 \simeq 0.118$

بیشترین gain را در ویژگی Source, Language دارند. یکی را انتخاب می‌کنیم ← Language

درخت نهایی





4) $w_1 x_1 + w_2 x_2 + b = 0$

معادله هریک از خطوط را بدست می‌آوریم. پس آنها را ترکیب می‌کنیم.

خط آبی : $y = \frac{2.5}{6} x + 2.5 \rightarrow 2.5 x_1 - 6 x_2 + 15 \geqslant 0$

خط نارنجی : $y = 5/3\ x - 5 \rightarrow 5 x_1 - 3 x_2 - 15 \leqslant 0$

خط سبز : $y = \frac{-2.5}{3} x + 2.5 \rightarrow 2.5 x_1 + 3 x_2 - 7.5 \leqslant 0$

} اشتراک 3 ناحیه

خط قرمز : $y = 2 \rightarrow 0 x_1 + x_2 - 2 \leqslant 0$

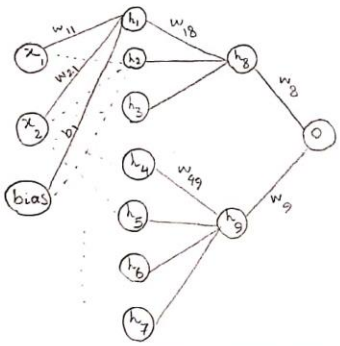
خط بنفش : $y = 0 \rightarrow 0 x_1 + x_2 + 0 \geqslant 0$

خط مشکی : $x = 0 \rightarrow x_1 + 0 x_2 + 0 \leqslant 0$

خط زرد : $x = -2 \rightarrow x_1 + 0 x_2 + 2 \geqslant 0$

} اشتراک 4 ناحیه

لایه هیدن اول :



$w_{11} = 2.5, w_{21} = -6, b_1 = 15.5$

$w_{12} = -5, w_{22} = 3, b_2 = 15.5$

$w_{13} = -2.5, w_{23} = -3, b_3 = 8$

$w_{14} = 0, w_{24} = -1, b_4 = 2.5$

$w_{15} = 0, w_{25} = 1, b_5 = 0.5$

$w_{16} = -1, w_{26} = 0, b_6 = -0.5$

$w_{17} = 1, w_{27} = 0, b_7 = 2.5$

وزن‌های نورون خروجی :

$w_8 = \frac{1}{2}, w_9 = \frac{1}{2}$

شرط active شدن تمامی نورون‌ها :

$$y = \begin{cases} 1 & w^T x + b \geqslant 0.5 \\ 0 & \text{otherwise} \end{cases}$$

لایه هیدن دوم :

$w_{18} = \frac{1}{6}, w_{19} = 0$

$w_{28} = \frac{1}{6}, w_{29} = 0$

$w_{38} = \frac{1}{6}, w_{39} = 0$

$w_{48} = 0, w_{49} = \frac{1}{8}$

$w_{58} = 0, w_{59} = \frac{1}{8}$

$w_{68} = 0, w_{69} = \frac{1}{8}$

$w_{78} = 0, w_{79} = \frac{1}{8}$

. .5

۵,۱: بخشی از داده آموزش را به عنوان اعتبارسنجی استفاده می‌کنیم. همچنین از transform برای نرمالیزه کردن تصاویر استفاده می‌کنیم.

```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5,), (0.5,))])

batch_size = 32

dataset = torchvision.datasets.MNIST(root='./data', train=True, download=True, transform=transform)
train_set, val_set = torch.utils.data.random_split(dataset, [0.8, 0.2])

train_loader = torch.utils.data.DataLoader(train_set, batch_size=batch_size, shuffle=True)
val_loader = torch.utils.data.DataLoader(val_set, batch_size=batch_size, shuffle=False)

test_set = torchvision.datasets.MNIST(root='./data', train=False, download=True, transform=transform)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=batch_size, shuffle=False)
```

پارامترهای شبکه و لایه های مختلف: از یک mlp دو لایه استفاده می‌کنیم. چون داده ورودی تصویر است و دو بعد دارد در ابتدا از یک لایه flatten استفاده می‌کنیم. سپس از یک لایه fully connected با ۲۸*۲۸ نورون ورودی و ۱۲۸ نورون خروجی، تابع فعالسازی relu، لایه fully connected دیگر با ۱۲۸ نورون ورودی و ۱۰ نورون خروجی استفاده می‌کنیم.

```
class MLP(nn.Module):

    def __init__(self, input_dim, h1, output_dim):
        super().__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(input_dim, h1)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(h1, output_dim)

    def forward(self, x):
        x = self.flatten(x)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

تابع train:

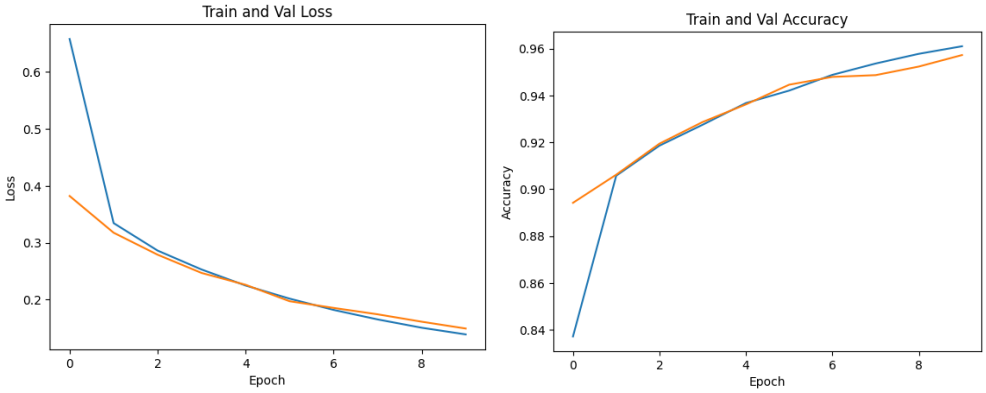
```python
def train(train_loader, val_loader, model, loss_fn, optimizer, epochs=10):

    train_losses = []
    val_losses = []
    train_accuracies = []
    val_accuracies = []

    for epoch in range(epochs):

        train_loss = 0
        train_acc = 0

        # train mode
        model.train()
        # iterate through batches
        for b, data in enumerate(train_loader):
            inputs, labels = data
            inputs = inputs.to(device)
            labels = labels.to(device)
            # zero the grads
            optimizer.zero_grad()
            # forward pass
            outputs = model(inputs)
            # find predictions for calculating accuracy
            preds = torch.argmax(outputs, dim=1)
            # calculate loss
            loss = loss_fn(outputs, labels)
            # backpropagation
            loss.backward()
            # update parameters
            optimizer.step()

            train_loss += loss.item()
            train_acc += torch.sum(labels == preds).item()

        train_loss /= len(train_loader)
        train_acc /= len(train_loader.dataset)

        # val mode
        model.eval()
        val_loss = 0
        val_acc = 0
        with torch.no_grad():
            for b, data in enumerate(val_loader):
                inputs, labels = data
                inputs = inputs.to(device)
                labels = labels.to(device)
                outputs = model(inputs)
                preds = torch.argmax(outputs, dim=1)
                loss = loss_fn(outputs, labels)
                val_loss += loss.item()
                val_acc += torch.sum(labels == preds).item()

        val_loss /= len(val_loader)
        val_acc /= len(val_loader.dataset)

        train_losses.append(train_loss)
        val_losses.append(val_loss)
        train_accuracies.append(train_acc)
        val_accuracies.append(val_acc)

        print(f'epoch {epoch}: train loss: {train_loss:.4f}, val loss: {val_loss:.4f}')

    return train_losses, val_losses, train_accuracies, val_accuracies
```

مدل را در ۱۰ ایپاک و با نرخ آموزش ۰,۰۰۰۱ آموزش می‌دهیم. نتیجه آموزش:

```python
mlp = MLP(28*28, 128, 10).to(device)
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(mlp.parameters(), lr=1e-4)

train_losses, val_losses, train_accuracies, val_accuracies = train(train_loader, val_loader, mlp, loss_fn, optimizer, 10)
```

```
epoch 0: train loss: 0.6577, val loss: 0.3820
epoch 1: train loss: 0.3344, val loss: 0.3176
epoch 2: train loss: 0.2862, val loss: 0.2790
epoch 3: train loss: 0.2530, val loss: 0.2469
epoch 4: train loss: 0.2248, val loss: 0.2261
epoch 5: train loss: 0.2018, val loss: 0.1973
epoch 6: train loss: 0.1820, val loss: 0.1855
epoch 7: train loss: 0.1653, val loss: 0.1743
epoch 8: train loss: 0.1507, val loss: 0.1613
epoch 9: train loss: 0.1391, val loss: 0.1494
```

Learning curve:



دقت طبقه‌بند: توانستیم به دقت ۹۵٪ بر روی داده آزمون برسیم.

```python
from sklearn.metrics import confusion_matrix

def test(test_loader, model):
    model.eval()

    all_labels = []
    all_predictions = []

    with torch.no_grad():
        for b, data in enumerate(test_loader):
            inputs, labels = data
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            preds = torch.argmax(outputs, dim=1)

            all_labels.extend(labels.cpu().numpy())
            all_predictions.extend(preds.cpu().numpy())

    return all_labels, all_predictions
```

```python
all_labels, all_predictions = test(test_loader, mlp)
test_acc = (np.array(all_labels) == np.array(all_predictions)).sum() / len(all_labels)

print("test accuracy:", test_acc)
print(confusion_matrix(all_labels, all_predictions))
```

```
test accuracy: 0.9577
[[ 962    0    1    1    0    4    8    2    2    0]
 [   0 1119    2    2    0    1    5    2    4    0]
 [   5    5  978   11    3    1    6   10   12    1]
 [   1    2    4  976    0    4    2    8    7    6]
 [   1    1    4    0  926    0   10    3    2   35]
 [   7    2    1   24    1  825   12    1   12    7]
 [   6    3    0    1    5    7  931    0    5    0]
 [   2    8   12    6    1    1    0  982    1   15]
 [   4    2    3   16    6    4   10    9  915    5]
 [   4    6    1   12    9    1    2   10    1  963]]
```

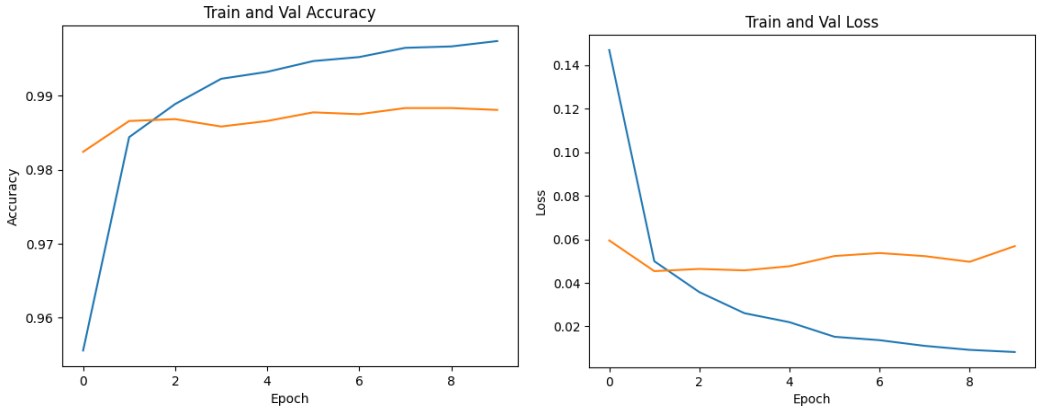
بهترین زمان متوقف کردن شبکه: مطابق نمودار خطا برای داده آموزش و اعتبارسنجی، از ایپاک پنجم به بعد میزان خطای آموزشی کاهش یافته ولی خطای اعتبارسنجی بسیار کم کاهش یافته. ممکن است مدل به سمت اورفیت شدن پیش رود پس بهتر است آموزش را در ایپاک ۵ متوقف کنیم.

۵,۲) پارامترهای شبکه و لایه های مختلف: شبکه cnn به صورت زیر است:

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
        self.relu1 = nn.ReLU()
        self.maxpool1 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.relu2 = nn.ReLU()
        self.maxpool2 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(64 * 7 * 7, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu1(x)
        x = self.maxpool1(x)

        x = self.conv2(x)
        x = self.relu2(x)
        x = self.maxpool2(x)

        x = self.flatten(x)
        x = self.fc1(x)

        return x
```

اولین لایه cnn کرنل ۳*۳ و ۳۲ کانال خروجی و تابع فعالسازی relu و بدنبال آن لایه max pooling است که به کاهش ابعاد کمک می‌کند. دومین لایه cnn کرنل ۳*۳ و ۶۴ کانال خروجی و تابع فعالسازی relu و بدنبال آن لایه max pooling است. در آخر از یک لایه تماما متصل برای پیش‌بینی خروجی استفاده می‌کنیم که دارای ۱۰ نورون خروجی است.

Learning curve:

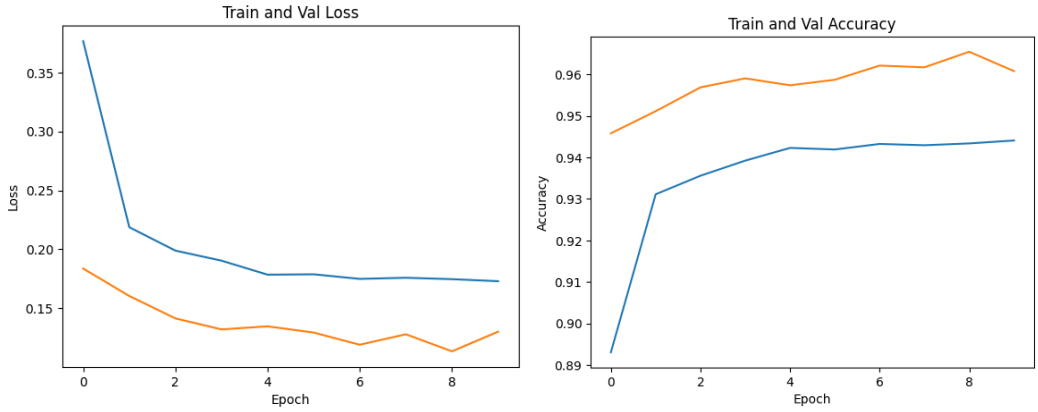


دقت طبقه‌بند:

```
test accuracy: 0.989
[[ 975    0    1    0    0    0    0    1    3    0]
 [   0 1130    2    0    0    0    1    1    1    0]
 [   0    0 1029    0    0    0    0    2    1    0]
 [   0    1    4 1001    0    4    0    0    0    0]
 [   0    0    1    0  975    0    0    0    1    5]
 [   0    1    0    5    0  882    3    0    1    0]
 [   3    1    1    0    2    2  946    0    3    0]
 [   0    4   20    0    0    0    0 1002    1    1]
 [   3    1    3    0    1    0    0    0  965    1]
 [   0    0    0    0   11    7    0    3    3  985]]
```

(۵,۳

Learning curve:



دقت طبقه‌بند:

```
test accuracy: 0.9624
[[ 957    0    1    1    4    5    6    1    1    4]
 [   0 1133    1    0    1    0    0    0    0    0]
 [   5    9  935   24    9   21    5   13    5    6]
 [   0    0    1  991    1   13    0    1    0    3]
 [   0    5    0    0  967    0    4    0    1    5]
 [   1    0    0   30    1  853    1    4    1    1]
 [   8    8    1    1    7    8  923    0    1    1]
 [   0   19    3    5   12    1    0  973    0   15]
 [   6    1    2    5    6    6    8    1  923   16]
 [   6    4    0    5    9    9    0    5    2  969]]
```

6. در ابتدا داده را می‌خوانیم و فاصله‌های اضافه را از مقادیر دیتافریم حذف می‌کنیم.

```python
import pandas as pd

columns = ['target', 'workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'native-country']
train_data = pd.read_csv('adult.train.10k.discrete', header=None)
train_data.columns = columns

for col in columns:
    train_data[col] = train_data[col].str.strip()
X_train = train_data.iloc[:, 1:]
y_train = train_data.iloc[:, 0]

test_data = pd.read_csv('adult.test.10k.discrete', header=None)
test_data.columns = columns

for col in columns:
    test_data[col] = test_data[col].str.strip()
X_test = test_data.iloc[:, 1:]
y_test = test_data.iloc[:, 0]
```

یک کلاس نود داریم که برای ساختن گره با property های مقدار، ویژگی و فرزندان از آن استفاده می‌کنیم.
از تابع calculate entropy برای محاسبه میزان آنتروپی بر حسب فرمول استفاده میکنیم.
به کمک تابع calculate gain، میزان آنتروپی کل را بدست می‌آوریم و بر اساس ویژگی خواسته شده در ورودی، آنتروپی مربوط به آن ویژگی و داده‌ها را حساب می‌کنیم. در نهایت میزان اطلاعات را بازمی گردانیم.

```python
import numpy as np

class Node:
    def __init__(self, value=None, feature=None, children=None):
        self.value = value
        self.feature = feature
        self.children = children


def calculate_entropy(X):
    # H(S) = - sigma(p*log(p))
    e = 0
    labels = X.unique()
    for l in labels:
        p = len(X[X==l]) / len(X)
        e -= p * np.log2(p) if p > 0 else 0
    return e


def calculate_gain(X, y, feature):
    total_entropy = calculate_entropy(y)
    conditional_entropy = 0

    unique_values = X[feature].unique()
    for value in unique_values:
        # find all rows with the specific value
        subset = X[X[feature]==value]
        subset_labels = y[X[feature]==value]
        if not subset.empty:
            e = calculate_entropy(subset_labels)
            conditional_entropy += e * len(subset)/len(X)

    return total_entropy - conditional_entropy

```

به کمک تابع find_most_informative، از میان ویژگی‌هایی که تا کنون استفاده نشده‌اند، ویژگی که بیشترین میزان اطلاعات را به ما می‌دهد پیدا می‌کنیم.
از تابع most_common_class برای پیدا کردن کلاسی که بیشترین تکرار را دارد استفاده می‌شود.
اصلی‌ترین تابع، build_tree است که به صورت recursive عمل می‌کند. شرط خاتمه (رسیدن به برگ) این است که در داده‌های موجود تمامی برچسب‌ها یکسان باشند و یا ویژگی‌ای باقی نمانده باشد. در غیر این صورت، بهترین ویژگی را پیدا می‌کنیم. و بر اساس مقادیر آن، باید ادامه درخت را تشکیل دهیم که به صورت بازگشتی این کار انجام می‌شود.

```python
def find_most_informative(X, y, features):
    most_gain = -float("inf")
    best_feature = None

    for feature in features:
        g = calculate_gain(X, y, feature)
        if g > most_gain:
            best_feature = feature
            most_gain = g

    return best_feature

def most_common_class(y):
    classes, counts = np.unique(y, return_counts=True)
    idx =  np.argmax(counts)
    return classes[idx]

def build_tree(X, y, features_left):

    # base case
    if len(y.unique()) == 1:
        return Node(value=y.iloc[0])

    if len(features_left) == 0 or len(X) == 0:
        return Node(value=most_common_class(y))

    best_feature = find_most_informative(X, y, features_left)
    unique_values = X[best_feature].unique()

    children = {}
    for val in unique_values:
        # find all rows which their feature value is 'val'
        subset = X[X[best_feature]==val]
        labels = y[X[best_feature]==val]
        if not subset.empty:
            new_features = [f for f in features_left if f != best_feature]
            children[val] = build_tree(subset, labels, new_features)

    return Node(feature=best_feature, children=children)
```

تابع predict برای پیش‌بینی کردن داده جدید به کار می‌رود و با traverseکردن درخت تا رسیدن به برگ عمل می‌کند.

```python
def predict(tree, data):
    try:
        if tree.value:
            return tree.value

        feature = tree.feature
        value = data[feature]
        return predict(tree.children[value], data)

    except:
        return '<=50K'
```

میزان دقت مدل برای داده آموزش:

```python
from sklearn.metrics import classification_report, accuracy_score

preds_train = []
for i in range(len(X_train)):
    preds_train.append(predict(root, X_train.iloc[i]))

print(accuracy_score(y_train, preds_train))
print(classification_report(y_train, preds_train))
```

```
0.8754
              precision    recall  f1-score   support

       <=50K       0.90      0.94      0.92      7550
        >50K       0.79      0.67      0.73      2450

    accuracy                           0.88     10000
   macro avg       0.84      0.81      0.82     10000
weighted avg       0.87      0.88      0.87     10000
```

میزان دقت مدل برای داده آزمون:

```
preds_test = []
for i in range(len(X_test)):
    preds_test.append(predict(root, X_test.iloc[i]))

print(accuracy_score(y_test, preds_test))
print(classification_report(y_test, preds_test))
```

```
0.8097
              precision    recall  f1-score   support

       <=50K       0.85      0.91      0.88      7539
        >50K       0.65      0.50      0.56      2461

    accuracy                           0.81     10000
   macro avg       0.75      0.71      0.72     10000
weighted avg       0.80      0.81      0.80     10000
```

به نظر می‌رسد در داده تست برای برخی ویژگی‌ها مقادیری وجود داشت که در داده آموزش نیامده بود.