H2I Workshop
Spectral Classifier for HSIs

Boyuan Sun

16/09/2021
Outline

• Introduction to spectral classifier
• Classifier design and training
• Training platforms
• Classifier implementation in Pytorch
Introduction to spectral classifier
Introduction to spectral classifier

• RGB images vs HSIs
Introduction to spectral classifier

- Convolution in 2d
Introduction to spectral classifier

• Convolution in ‘3d’
Introduction to spectral classifier

- Data shape transformation
Classifier design and training

• Design

  • Use Cifar10Net architecture
  • Change parameters of Cifar10Net
    • Convolution kernel size
    • Pooling size
    • Input and output of fully connected layers
Training

• Direct training
  • Put all data together and fed to the network
  • Observation: imbalanced performance between categories
Training

• Training by phases

  • Data selection and grouping
    • Group data by metric
    • 3 groups

• Each phase trained by different data group

  • Train from scratch -> Finetune the network
Data selection and grouping

• Process training data process
  • Apply classifier to training data
  • Evaluate the outputs(probability) for training data
  • Group the data: Strong, Intermediate, Weak
Training by phases

• Phase 1:
  • Train the network from scratch
  • Data: Group strong feature

• Phase 2:
  • Finetune the network for 2 fully connected layers
  • Data: Group intermediate feature

• Phase 3:
  • Finetune the network for last fully connected layer
  • Data: Group weak feature
Training Platforms

• Pytorch: @Facebook, adapted from torch
• Tensorflow: @Google, integrated with Keras
• MxNet: @Apache Software Foundation
• CNTK: @Microsoft
• …
Training Platform

• Pytorch:

The five steps in developing as follows:
• 1. Prepare the Data
• 2. Define the Model
• 3. Train the Model
• 4. Evaluate the Model
• 5. Make Predictions
Prepare the Data

- Numerical input data and numerical output data
- Python libraries: numpy, pandas
- Pytorch build-in class: `DATASET` and `DATALOADER`

```python
# dataset definition
class CSVDataset(Dataset):
    # load the dataset
    def __init__(self, path):
        # store the inputs and outputs
        self.X = ...
        self.y = ...

    # number of rows in the dataset
    def __len__(self):
        return len(self.X)

    # get a row at an index
    def __getitem__(self, idx):
        return [self.X[idx], self.y[idx]]

# create the dataset
dataset = CSVDataset(...)
# select rows from the dataset
train, test = random_split(dataset, [[...], [...]])
# create a data loader for train and test sets
train_dl = DataLoader(train, batch_size=32, shuffle=True)
test_dl = DataLoader(test, batch_size=1024, shuffle=False)
```
Define the Model

• Defines the layers of the model
• Define forward() function that defines how to forward propagate input through the defined layers of the model.
• Layers: Conv2d, Linear, Maxpool..

```python
# model definition
class MLP(Module):
    # define model elements
def __init__(self, n_inputs):
    super(MLP, self).__init__()
    self.layer = Linear(n_inputs, 1)
    self.activation = Sigmoid()

    # forward propagate input
    def forward(self, X):
        X = self.layer(X)
        X = self.activation(X)
        return X
```
Train the Model

• Define loss function
  • BCELoss: Binary cross-entropy loss for binary classification.
  • CrossEntropyLoss: Categorical cross-entropy loss for multi-class classification.
  • MSELoss: Mean squared loss for regression.

• Define optimization algorithm
  • SGD: stochastic gradient descent
  • Adam: A method for stochastic optimization

```python
# define the optimization
criterion = MSELoss()
optimizer = SGD(model.parameters(), lr=0.01, momentum=0.9)
# enumerate epochs
for epoch in range(100):
    # enumerate mini batches
    for i, (inputs, targets) in enumerate(train_dl):
        ...

    # clear the gradients
    optimizer.zero_grad()
    # compute the model output
    yhat = model(inputs)
    # calculate loss
    loss = criterion(yhat, targets)
    # credit assignment
    loss.backward()
    # update model weights
    optimizer.step()
```
Evaluate the Model

• Prepare testing data
• Load the model
• Set the model into evaluation mode
• Evaluation
• Save the model
Make Predictions

• Prepare testing data
• Load the model
• Set the model into evaluation mode
• Evaluation
Classifier implementation in Pytorch
Thanks!

🏠 https://h2i.inf.unibz.it/
✉️ h2i-fesr@googlegroups.com