哈爾濱工業大學

HARBIN INSTITUTE OF TECHNOLOGY

模式识别与深度学习

实验报告

RNN 网络函数预测与情感分析

黄海

1160300329

计算机科学与技术

指导教师 左旺孟

2019 年 6 月 6 日

目录

Chapter 1

正弦预测

1.1 解释

通过搭建基本的 RNN 网络，通过产生足够长度的时间序列，对每一组数据进行预测。这里采用的策略是把每个 2π 分成两个部分，前一个部分进行训练，后一个部分进行预测，最后使用标准正弦进行拟合更正。

1.2 代码

以下是训练的部分代码

```
for STEP in RANGE(200):
    START, END = STEP * NP.PI, (STEP + 1) * NP.PI
    STEPS = NP.LINSPACE(START, END, 5, DTYPE=NP.FLOAT32)
    S.EXTEND(STEPS)
    X_NP = NP.SIN(STEPS - NP.PI)
    Y_NP = NP.SIN(STEPS)
    X = VARIABLE(TORCH.FROM_NUMPY(X_NP[NP.NEWAXIS, :, NP.NEWAXIS]))
    Y = VARIABLE(TORCH.FROM_NUMPY(Y_NP[NP.NEWAXIS, :, NP.NEWAXIS]))
    PREDICTION, SELF.H_STATE = SELF.MODULE(X, SELF.H_STATE)
    P = PREDICTION
    SELF.H_STATE = VARIABLE(SELF.H_STATE.DATA)
```

```
LOSS = SELF.LOSS_FUNC(PREDICTION, Y)
SELF.OPTIMIZER.ZERO_GRAD()
LOSS.BACKWARD()
SELF.OPTIMIZER.STEP()
ORIGINAL_SIN.EXTEND(Y_NP.FLATTEN())
TRAIN_SET.EXTEND(PREDICTION.DATA.NUMPY().FLATTEN())
```

1.3 实验结果

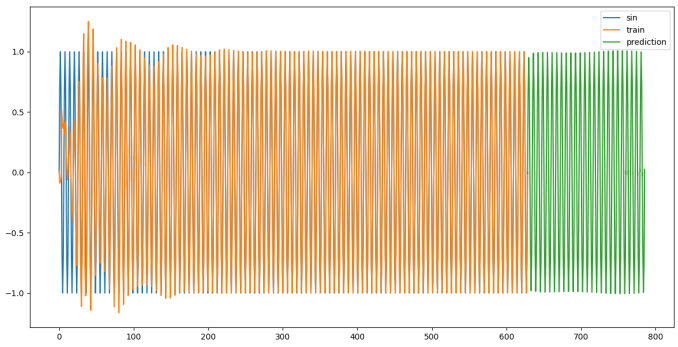

图 1.1: 训练以及预测结果

Chapter 2

文本情感分析

2.1 解释

文本情感分析的主要难点在于如何组织数据并进行训练，在这次实验中，处理过后的数据分成了两块，前 4000 个语句为训练使用，后 1000 个为测试。这之后便大同小异。调整好网络的参数，将输入调整为 50 个单词，对于不够长度的句子进行补齐。最后输出 1 维向量用来表示结果

2.2 代码

以下展示了部分的训练代码

```
for I in RANGE(4000):
    print(I)
    X_NP = NP.ARRAY(SELF.NEGTRAINNUM[I])
    Y_N = [SELF.NEGTRAINLABELS[I]]
    Y_NP = NP.ARRAY(Y_N)
    X_NP = X_NP.ASTYPE(NP.FLOAT32)
    Y_NP = Y_NP.ASTYPE(NP.FLOAT32)
    X = TORCH.FROM_NUMPY(X_NP[NP.NEWAXIS, :])
    Y = TORCH.FROM_NUMPY(Y_NP[NP.NEWAXIS, :, NP.NEWAXIS])
    X = VARIABLE(X)
```

```
        y = Variable(y)
        prediction = self.module(x)
        loss = self.loss_func(prediction, y)
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
        # print(loss.item())

        x_np = np.array(self.postrainnum[i])
        y_n = [self.postrainlabels[i]]
        y_np = np.array(y_n)
        x_np = x_np.astype(np.float32)
        y_np = y_np.astype(np.float32)
        x = torch.from_numpy(x_np[np.newaxis, :])
        y = torch.from_numpy(y_np[np.newaxis, :, np.newaxis])
        x = Variable(x)
        y = Variable(y)
        prediction = self.module(x)
        loss = self.loss_func(prediction, y)
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
        # print(loss.item())
    print("End Train.")
```