基于 RNN-RBM 的音乐生成

一 背景介绍

音乐生成是让计算机自动创作音乐的技术。音乐创作一直被认为是一种高难度，依赖人类突发灵感的高级思维活动。另一方面，音乐创作又需要遵循严格的规律，如节拍性、强弱性等。这意思着创造音乐是一项既要循规蹈矩，又要寻求新意的艰苦劳动，仅有对音高、节拍等具有敏锐感觉，且思维活跃度极高的少数人能够胜任。幸运的是，这种在严格框架下进行有限创新的工作，计算机具有天然优势，它可以充分保证生成作品的合规性，同时在合规下探索各种可能的创新。让计算机自动生成音乐，可极大减少人类进行音乐创作的工作量，且有望产生挣脱传统思路束缚新颖音乐。即便用机器生成的音乐还不能与人类的音乐家相比，但机器作品可以为人类提供候选或初级作品，使作曲家创作更加容易；同时，计算机生成的音乐还可以为作曲家提供灵感和刺激，激发他们不断创造新的音乐，仿止因长期创作带来的风格惰性和思维困顿，帮助作曲家永褒创作青春。因此，自动音乐生成具有非常广阔的应用前景。

音乐生成一般可以有以下两种方式：

1、 经典的概率模型

该方法用语言模型或者 HMM 模型训练字符化的乐谱，然后生成一段乐谱，能取得比较不错的效果。

2、 神经网络（NN）模型

该方法学习字符化的乐谱利用神经网络，相当于学习一个序列，然后用模型生成一个序列，这种方法一个时刻只能生成一个音符，不过可以获得一些令人满意的音乐片段。

二 实验原理介绍

在本次实验中，即用 RNN-RBM 和 LSTM-RBM 产生 model 复调音乐，训练过程中采用的是 midi 格式的音频文件，接着用建好的 model 来产生复调音乐。对音乐建模的难点在与每首乐曲中帧间是高度时间相关的（这样样本的维度会很高），用普通的网络模型是不能搞定的，这种情况下可以采用 RNN 来处理，而 RNN 在处理长序列的时候，会出现梯度下降或者梯度爆炸，而 LSTM 更好的解决这个问题。

下面就 RNN、LSTM 和 RBM 原理介绍，最后对 RNN-RBM 神经网络做介绍。

多层反馈 RNN（Recurrent neural Network、循环神经网络）神经网络是一种节点定向连接成环的人工神经网络。这种网络的内部状态可以展示动态时序行为。不同于前馈神经网络的是，RNN 可以利用它内部的记忆来处理任意时序的输入序列，这让它可以更容易处理如不分段的手写识别、语音识别等。其结构如下图 1 所示：

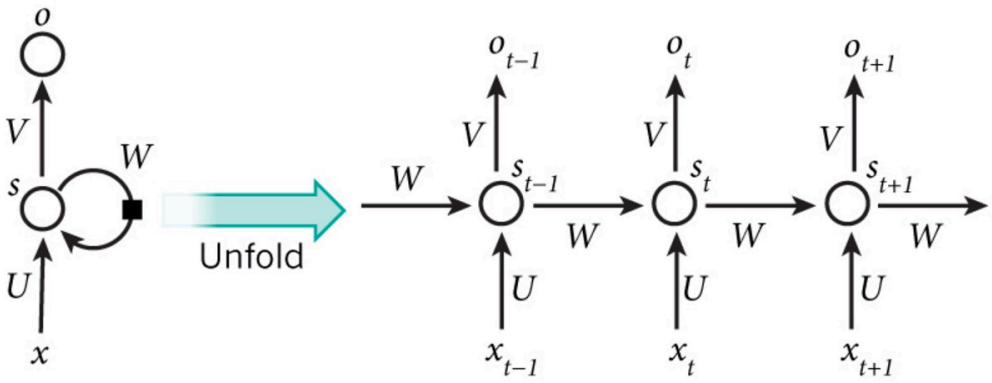

图 1 RNN 结构

RNN 公式如下：

$$s_t = \sigma\left(Wx_t + Us_{t-1} + b_s\right) \quad (1)$$

$$o_t = g\left(Vs_t + b_o\right) \quad (2)$$

其中，$W,U,V \in R^{n\times m}, R^{n\times n}, R^{n\times k}$ 属于权重矩阵，$b_s, b_o \in R^n, R^k$ 属于偏置矩阵，m,n,k 分别为输入 x,隐层 s,输出 o 的维度。

LSTM（Long-Short Term Memory,LSTM）是一种时间递归神经网络，论文首次发表于 1997 年。由于独特的设计结构，LSTM 适合于处理和预测时间序列中间隔和延迟非常长的重要事件。这种结构主要是解决了 RNN 记忆力短暂的问题。与 RNN 相比，LSTM 在隐层做了较大的改进，一个 Cell 由三个 Gate（input、forget、output）和一个 cell 单元组成。Gate 使用一个 sigmoid 激活函数，而 input 和 cell state 通常会使用 tanh 来转换。LSTM 其结构如图 2 所示：

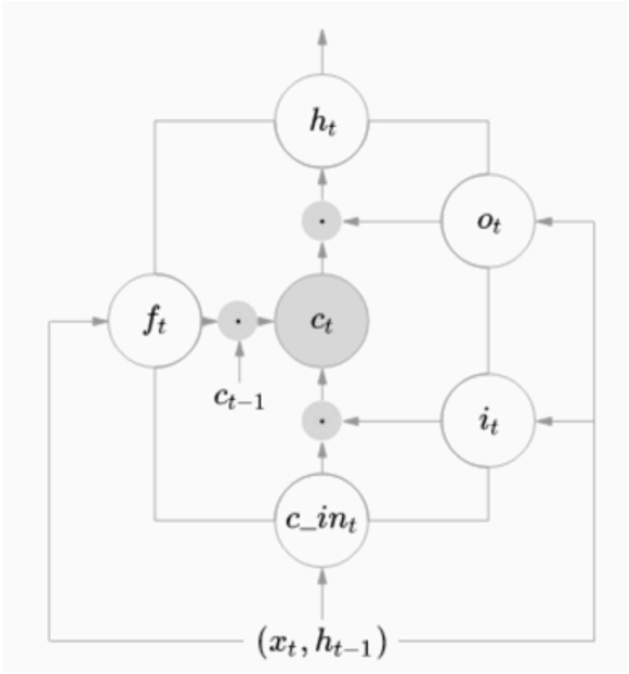

图 2 LSTM's cell 结构图

LSTM 生成隐层的向量公式如下：

$$h_t = \begin{cases} o_t * \tanh(C_t) & ,if\ t > 0 \\ 0 & ,if\ t = 0 \end{cases} \quad (3)$$

其中：

$$o_t = \sigma\left(W_o E_{x_t} + U_o h_{t-1} + b_o\right) \quad (4)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (5)$$

$$\tilde{C}_t = \sigma\left(W_C E_{x_t} + U_C h_{t-1} + b_C\right) \quad (6)$$

$$i_t = \sigma\left(W_i E_{x_t} + U_t h_{t-1} + b_i\right) \quad (7)$$

$$f_t = \sigma\left(W_f E_{x_t} + U_f h_{t-1} + b_f\right) \quad (8)$$

$E \in R^{m \times K_x}$ 是对应的输入向量矩阵。$W_o, W_C, W_i, W_f \in R^{n \times m}$ 和 $U_o, U_C, U_t, U_f \in R^{n \times n}$ 是权重矩阵，b_o，b_C，b_i，$b_f \in \mathrm{R}^n$ 是偏置向量，m 是词向量的维度，n 是隐层数。

受限波尔兹曼机（RBM）是一类具有两层结构、对称连接且无自反馈的随机神经网络模型昏 层间全连接昏 层内无连接。RBM 具有一个可见层，一个隐层，层内无连接，其结构如图 3 所示。RBM 具有很好的性质，在给定可见层单元状态(输入数据)时， 各隐单元的激活条件独立，反之，在给定隐单元状态时， 可见层单元的激活亦条件独立。这样一来， 尽管 RBM 所表示的分布仍无法有效计算，但通过 Gibbings 采样可以得到服从 RBM 所表示分布的随机样本。

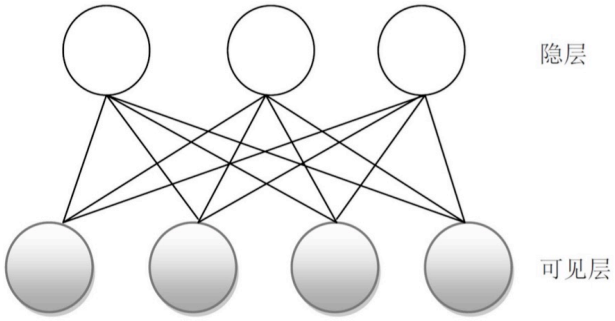

图 3 RBM 结构图

RBM 是一种能量模型，可以给出可见层(v)和隐藏层(h)联合概率分布。其概率分布函数为：

$$p(v,h) = \exp\left(-b_v^T v - b_h^T h - h^T W v\right) / Z \quad (9)$$

其中，b_v, b_h, W 是 RMB 模型的参数。

由于给定可见层(v)或隐藏层(h)，生成可见层(v)和隐藏层(h)是条件独立的。

$$p(h_i = 1 | v) = \sigma\left(b_h + W v\right)_i \quad (10)$$

$$p\left(v_j=1\,|\,v\right)=\sigma\left(b_v+Wh\right)_j \qquad (11)$$

RNN(LSTM)-RBM 是综合了 RNN(LSTM)和 RBM 的优点，很适合用来产生复调音乐。因为复调音乐具有高纬度时序依赖的特性，RNN(LSTM)模块学习时序上的依赖关系，保证风格的统一性，而 RBM 模块学习给定上一个高维向量数据，产生一个符合给定条件的高维向量。如下图 4 所示：

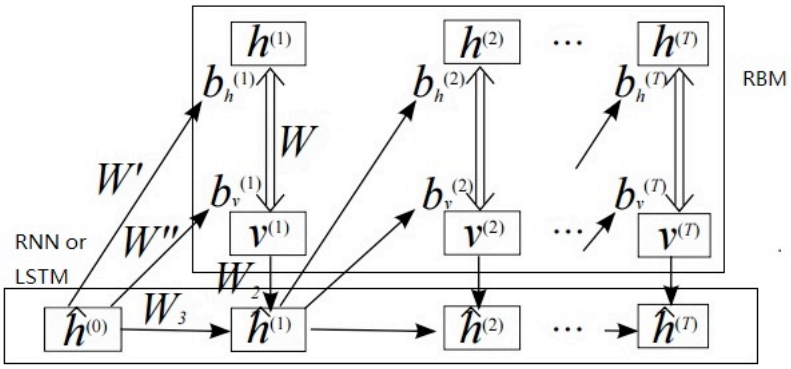

图 4 RNN(LSTM)-RBM 结构图

其主要公式如下：

$$b_h^{(t)}=b_h+W'\hat{h}^{(t-1)} \qquad (12)$$

$$b_v^{(t)}=b_v+W'\hat{h}^{(t-1)} \qquad (13)$$

$$\hat{h}^{(t)}=\sigma\left(W_2v^{(t)}+W_3\hat{h}^{(t-1)}+b_{\hat{h}}\right) \qquad (14)$$

RNN-RBM 使用交叉熵代价函数如下：

$$L\left(\left\{v^{(t)}\right\}\right)=\frac{1}{T}\sum_{t=1}^{T}\sum_{j}^{n_v}-v_j^{(t)}\log y_j^{(t)}-\left(1-v_j^{(t)}\right)\log\left(1-y_j^{(t)}\right) \qquad (15)$$

三 主要代码

在本实验中，主要的 RNN 和 RBM 模型代码如下图 5 和 6 所示：

```
def recurrence(v_t, u_tm1,c_tm1):
    bv_t = bv + T.dot(u_tm1, Wuv)
    bh_t = bh + T.dot(u_tm1, Wuh)
    generate = v_t is None
    if generate:
        v_t, _, _,mean_t, updates = build_rbm(T.zeros((n_visible,)), W, bv_t,
                                              bh_t, k=25)
    u_t = T.tanh(bu + T.dot(v_t, Wvu) + T.dot(u_tm1, Wuu))

    # u_t,c = lstm(v_t,u_tm1,c_tm1,Wvu,Wuu,bu)

    return ([v_t, u_t,c,mean_t], updates) if generate else [u_t, c,bv_t, bh_t]
```

图 5 RNN 代码

```
def gibbs_step(v):
    mean_h = T.nnet.sigmoid(T.dot(v, W) + bh)
    h = rng.binomial(size=mean_h.shape, n=1, p=mean_h,
                     dtype=theano.config.floatX)
    mean_v = T.nnet.sigmoid(T.dot(h, W.T) + bv)

    v = rng.binomial(size=mean_v.shape, n=1, p=mean_v,
                     dtype=theano.config.floatX)
    return mean_v, v

(mean_chain, chain), updates = theano.scan(lambda v: gibbs_step(v), outputs_info=[None, v],
                           n_steps=k)
v_sample = chain[-1]

mean_v = gibbs_step(v_sample)[0]
monitor = T.xlogx.xlogy0(v, mean_v) + T.xlogx.xlogy0(1 - v, 1 - mean_v)
monitor = monitor.sum() / v.shape[0]

def free_energy(v):
    return -(v * bv).sum() - T.log(1 + T.exp(T.dot(v, W) + bh)).sum()
cost = (free_energy(v) - free_energy(v_sample)) / v.shape[0]
```

图 6 RBM 和 cost 函数代码

四 输入以及参数设置

在本试验中，源数据是 midi 格式的钢琴曲，总共有各类风格一万四千多首，数据来源是从虫虫钢琴网站。

1、 把 midi 格式的钢琴曲转换成一个 88 维的向量数组，每一个 88 维向量依次作为 RNN 可见层输入
2、 RNN 隐层数设为 100
3、 RBM 隐层数设为 150

五 实验步骤及结果

在本次实验中，主要是基于论文《Modeling Temporal Dependencies in High-Dimensional Sequences》的方法，进去拓展和改进，下面是具体的实验步骤以及每步的结果。

5.1. 复现论文中的结果

在这个阶段，我通过对神经网络，python 和 theano 有了初步的学习和认识，运行了论文中提供的代码和数据，还原了论文中的结果。利用论文提供的数据集中，获得了比较悦耳的结果。图 5 是复现文论结果生成的挑选的一段音乐旋律表示。

图 7　在论文数据集上生成的结果

5.2. 用自己数据集替换论文的数据集

在复现论文的结果之后，我开始尝试用自己找的 midi 格式数据集，替换原来的数据集。这样做的目的是为了实验论文中的方法在不同的数据集上的泛化能力。我分别在周杰伦和邓丽君的数据上构建模型，并分别生成了结果，如下图 6 和 7 所示。

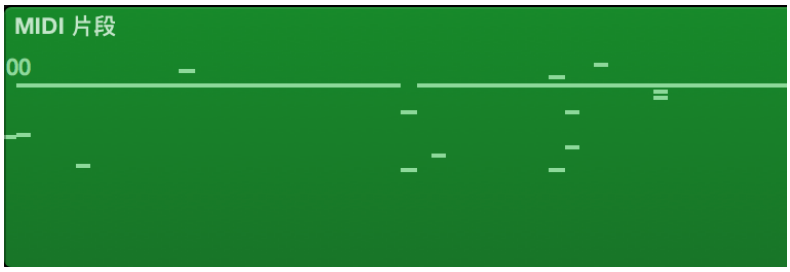

图 8　在邓丽君数据集上生成的结果

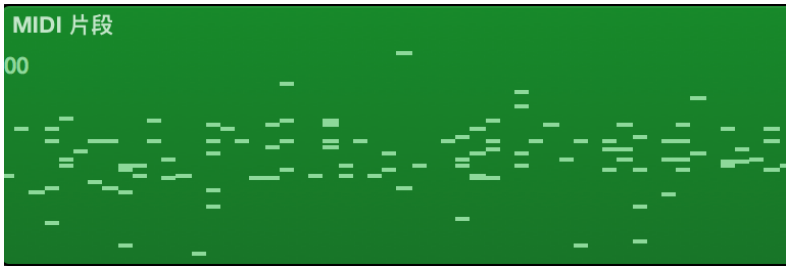

图 9 在周杰伦数据集上生成的结果

由上面的音符可以看出，生成的结果很差，在邓丽君数据集上的音符过于稀疏，而在周杰伦数据集上的音符过于杂乱，缺乏旋律性。把这两个数据集跟论文的数据集比较结果之后，分析得出在邓丽君的数据量过小，而在周杰伦的数据上 midi 格式的数据过于多样化，并不全是钢琴曲，而有一部分是交响曲或者吉它曲。

经过上面的分析之后，在自己下载的 midi 格式的数据中，使用操作 midi 格式的 python API 分别挑选出了 500 首和 2000 首符合要求的钢琴曲，生成的结果如下图 8 和 9 所示。

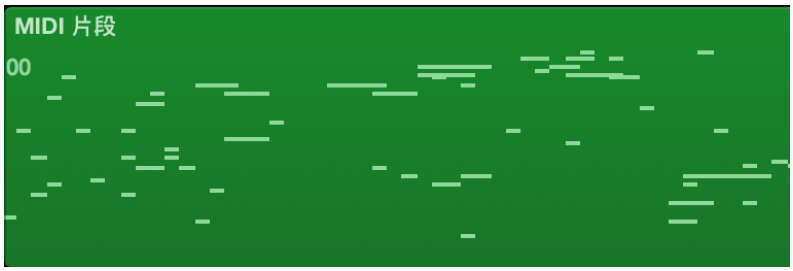

图 10　在 500 首数据集上生成的结果

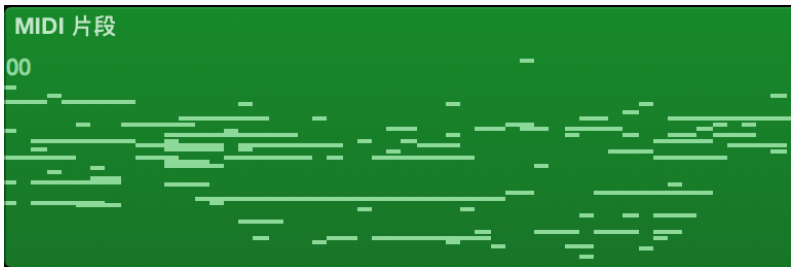

图 11 在 500 首数据集上生成的结果

通过对比可以看出，500 首和 2000 首在生成的音符旋律明显要比邓丽君和周杰伦数据集上的结果要好，这也验证了分析之后的假设。而且，2000 首的数据集比 500 首的数据集生成的音乐片段旋律性更好，说明增加数据集可以达到更好的效果。

5.3．对模型进行改进

在上一步中，用论文原模型代码跑不同的数据集，产生了一些问题，也解决了一些问题。不过，最后分别在 500 首和 2000 首数据集上跑的结果虽然比邓丽君和周杰伦的数据集上有

不错的结果，但是总体来说还是差强人意，比在论文提供的数据集上的结果还是有一定的差距。

在反复把自己的数据集和论文提供的数据集进行比对之后，发现论文提供的数据集中的钢琴曲在时长上远远小于自己的数据集。经过分析，得出这样的结论，由于自己的数据集上的时长过长和 RNN 自身的局限性，导致 RNN-RBM 模型不能很好的记住所学习钢琴曲的整体结构。LSTM 在由于独特的设计结构，LSTM 适合于处理和预测时间序列中间隔和延迟非常长的重要事件。于是，决定把 RNN-RBM 换成 LSTM-RBM。关键代码如下图 10 所示。

```
def lstm(X, h, c, W, U, b):
    g_on = T.dot(X,W) + T.dot(h,U) + b
    i_on = T.nnet.sigmoid(g_on[:n_hidden_recurrent])
    f_on = T.nnet.sigmoid(g_on[n_hidden_recurrent:2*n_hidden_recurrent])
    o_on = T.nnet.sigmoid(g_on[2*n_hidden_recurrent:3*n_hidden_recurrent])
    c = f_on * c + i_on * T.tanh(g_on[3*n_hidden_recurrent:])
    h = o_on * T.tanh(c)
    return h, c
```

图 12 修改 RNN-RBM 到 LSTM-RBM 的关键代码

当把模型改为 LSTM-RBM 之后，分别在 500 首和 2000 首数据集上训练模型，生成的音乐片段，如下图 11 和 12 所示。

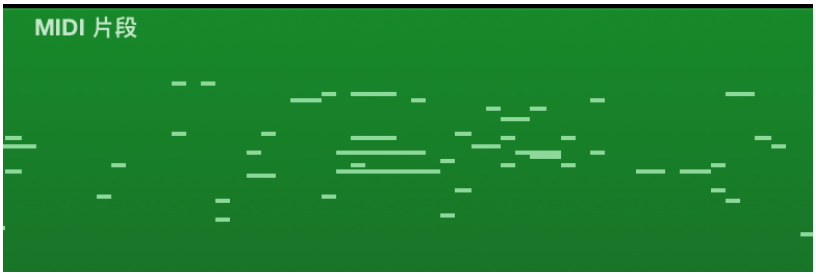

图 13 LSTM-RBM 在 500 首数据集上生成的结果

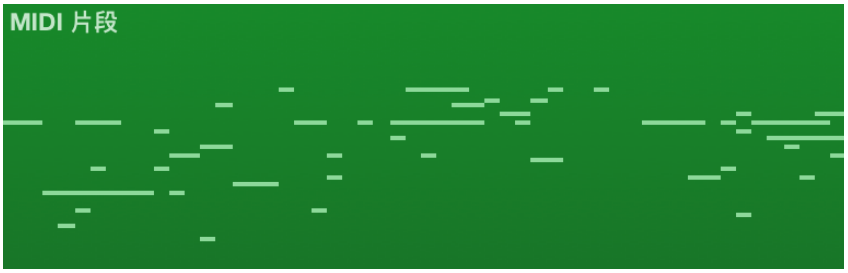

图 14 LSTM-RBM 在 2000 首数据集上生成的结果

从上图可以看出，LSTM-RBM 模型生成的音乐片段明显要比 RNN-RBM 生成的音乐片段旋律更加好一些，从而验证了分析之后的假设，也证明了 LSTM 确实比 RNN 有更轻的记忆能力。

虽然再经过几次修改之后，在自己的数据集上，表现的越来越好，不过在一些生成的音乐里边，有部分的音乐会出现不连续的情况。因为 RBM 在生成阶段，在生成每一个向量时，其实是先生成向量每一个维度的概率，然后根据每一维的概率，再经过一个二项分布生成最终得向量。根据这个过程，想出了只输出所有维的最大概率的向量这个方案，这一部分的关键代码如下图 13 所示：

```
note_data = self.generate_function()
piano_roll = note_data[0]
max = 0
mean_sample = note_data[1]

for index,list in enumerate(mean_sample):
    max = 0
    for i in range(len(list)):
        if(list[i] > list[max]):
            max = i
    for i in range(len(piano_roll[index])):
        if(i == max):
            piano_roll[index][i] = 1
        else:
            piano_roll[index][i] = 0
```

图 15 输出最大概率的关键代码

在修改之后，在 2000 首数据集上生成的音乐片段，产生了比较令人惊喜的旋律，其生成结果如下图 14 所示：

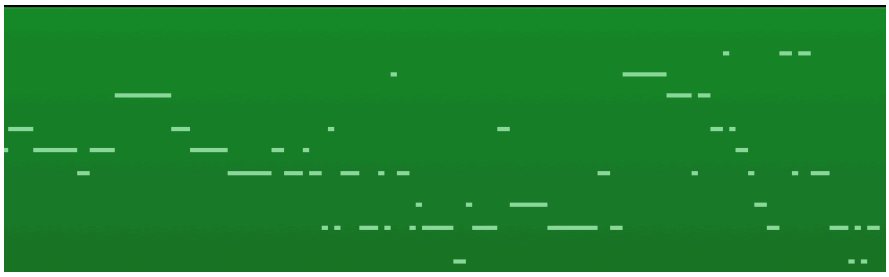

图 16 修改输出方式之后生成的结果

六 实验总结

1. 在这个实验过程中，我学习了神经网络，对神经网络有了比较深刻的认识，也使我对神经网络这一领域产生了浓厚的兴趣。
2. 在实验中，不断出现问题，分析之后，不断探索解决方案，对我的逻辑思维能力和动手能力有了比较大的提升。
3. 在这个实验过程中，我也阅读了其他关于音乐生成的论文，这些论文给我带来了不少的启发，其中的方法和思想可以借鉴到我以后的研究工作中。