Learning RNN

# 1.Introduction

In this paper I will introduce RNN technology and explain RNN and explain how to do back propagation to compute the weight of RNN. This tutorial will using character prediction as example to show how to use RNN to predict next character based on current input and hidden state.

Exmaple code: <https://github.com/weixsong/min-char-rnn>

# 2.Simple RNN Theory

RNN 的图片表示如下：



基本的RNN的python代码如下：

**class** **RNN**:

*# ...*

**def** **step**(self, x):

*# update the hidden state*

self**.**h **=** np**.**tanh(np**.**dot(self**.**W\_hh, self**.**h) **+** np**.**dot(self**.**W\_xh, x))

*# compute the output vector*

y **=** np**.**dot(self**.**W\_hy, self**.**h)

**return** y

其中，t时刻的hidden state的计算公式为： ht=tanh(Whh\*ht-1 + Wxh\*xt), xt表示t时刻的RNN输入。

特别注意：这里的RNN的hidden layer采用了tanh作为激活函数（Activation function）, Output layer则没有采用任何激活函数，即为线性的output.

# 3. Using RNN to predict next character

我们利用RNN来进行下一个character的预测，利用已知的字母序列来预测下一个可能出现的字母，首先我们有以下若干RNN使用的假设，用来初始化我们的RNN模型：

1. 假设我们一共有V个character, 即字典的大小为V，那么RNN的输入为one hot vector, dim=V
2. 假设我们当前的RNN的hidden layer的size为H,则我们的RNN的权重矩阵的维度分别为：
   1. Wxh: H\*V, input to hidden layer weight
   2. Whh: H\*H, hidden to hidden layer weight
   3. Why: V\*H, hidden to output layer weight