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 use 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
3. Output layer的维度为V，输出为每个字母可能出现的概率，output layer做了一个softmax操作，获得每个字母可能出现的概率分布，在预测下一个字母的时候就可以在这个得到的字母分布的基础上进行采样。
4. 设Hidden layer的输入为u,输出为h, output layer的输入为u’, 这里，因为hidden layer采用tanh作为激活函数，所以h = tanh(u), output layer没有采用激活函数，所以 y = u’ .

接下来我们讲的所有的操作，forward propagation & back propagation都是以本节的设定为基础的。

# 4. Loss Function

在NN中常用的两种Loss function有：Sum of squared error (Quadratic error) & cross entropy error。

假设t为训练样本的真实值，y为神经网络的输出，我们的训练样本为(x, t)，一个样本。我们下面的公式也是针对一个样本而言，对于所有的样本也就很简单了。

## 4.1 Sum of Squared error (Quadratic error)

(1)

当神经网络的output layer没有采用激活函数的时候，我们应该采用Quadratic error，这样能够比较快速的进行梯度下降参数估计。

## 4.2 Cross Entropy Error

(2)

如果当output layer采用sigmoid激活函数的时候，我们应该采用cross entropy error进行参数估计，这是因为如果采用sigmoid激活函数，采用cross entropy error能够在求导的时候消掉sigmoid函数，这样能够加快梯度下降的速度。接下来的参数估计推导部分会有详细的推导说明。如果output layer没有采用sigmoid函数，但是却使用cross entropy error来进行参数估计，那么就会得到不是很舒服的偏导数公式，可能对梯度下降的速度有所影响。

# 5. Forward Propagation

离得太远了，再把图拿过来看一下



Forward propagation 的计算过程如下：

1. u = Wxh \* x (u 为 hidden layer的输入) (3)
2. ht = tanh(Whh\*ht-1 + u) (h 为hidden layer的输出) (4)

= tanh(zt + u) (zt = Whh\*ht-1)

1. u’ = Why\*h (u’为output layer的输入) (5)
2. y = u’ (y 为output layer的输出, output layer 为linear layer, 所以 y = u’) (6)

上面的计算公式中，具体的每个神经元的计算如下：

1. (7)
2. (8)
3. (9)
4. (10)
5. (11)