RNN_project

June 19, 2017

1 Artificial Intelligence Nanodegree

1.1 Recurrent Neural Network Projects

Welcome to the Recurrent Neural Network Project in the Artificial Intelligence Nanodegree! In this notebook, some template code has already been provided for you, and you will need to implement additional functionality to successfully complete this project. You will not need to modify the included code beyond what is requested. Sections that begin with **'Implementation'** in the header indicate that the following block of code will require additional functionality which you must provide. Instructions will be provided for each section and the specifics of the implementation are marked in the code block with a 'TODO' statement. Please be sure to read the instructions carefully!

In addition to implementing code, there will be questions that you must answer which relate to the project and your implementation. Each section where you will answer a question is preceded by a **'Question X'** header. Carefully read each question and provide thorough answers in the following text boxes that begin with **'Answer:'**. Your project submission will be evaluated based on your answers to each of the questions and the implementation you provide.

> **Note:** Code and Markdown cells can be executed using the **Shift + Enter** keyboard shortcut. In addition, Markdown cells can be edited by typically double-clicking the cell to enter edit mode.

1.1.1 Implementation TODOs in this notebook

This notebook contains two problems, cut into a variety of TODOs. Make sure to complete each section containing a TODO marker throughout the notebook. For convinence we provide links to each of these sections below.

2 Problem 1: Perform time series prediction

In this project you will perform time series prediction using a Recurrent Neural Network regressor. In particular you will re-create the figure shown in the notes - where the stock price of Apple was

forecasted (or predicted) 7 days in advance. In completing this exercise you will learn how to construct RNNs using Keras, which will also aid in completing the second project in this notebook.

The particular network architecture we will employ for our RNN is known as Long Term Short Memory (LTSM), which helps significantly avoid technical problems with optimization of RNNs.

2.1 1.1 Getting started

First we must load in our time series - a history of around 140 days of Apple's stock price. Then we need to perform a number of pre-processing steps to prepare it for use with an RNN model. First off, it is good practice to normalize time series - by normalizing its range. This helps us avoid serious numerical issues associated how common activation functions (like tanh) transform very large (positive or negative) numbers, as well as helping us to avoid related issues when computing derivatives.

Here we normalize the series to lie in the range [0,1] [using this scikit function](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html), but it is also commonplace to normalize by a series standard deviation.

In [1]:
```
### Load in necessary libraries for data input and normalization
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

### load in and normalize the dataset
dataset = np.loadtxt('datasets/normalized_apple_prices.csv')
```

Lets take a quick look at the (normalized) time series we'll be performing predictions on.

In [2]:
```
# lets take a look at our time series
plt.plot(dataset)
plt.xlabel('time period')
plt.ylabel('normalized series value')
```

Out[2]:
```
<matplotlib.text.Text at 0x7f0052460240>
```