传统的时间序列预测研究主要是采用统计分析方法以及计量经济模型对其进行研究，到上世纪80年代之后，机器学习开始被逐渐应用与金融市场，到目前为止，也取得了一定的成功，原因在于传统的传统的统计计量方法可以模拟一个平稳的时间序列，但是通常需要将大量的数据进行预处理成为平稳序列。或者

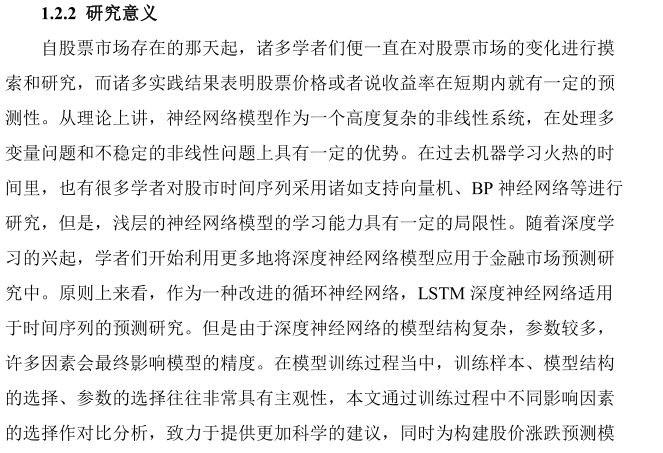
与传统的统计计量方法相比较，神经网络机器学习通常可以学习一个非线性平稳的序列过程。但是即使这样，也有一定的局限性，比如传统的BP神经网络模型，不能处理时间序列数据。同时，输入数据个数难以确定的问题目前也没有很好的解决办法。但传统的BP神经网络模型并不适用于时间序列训练预测的问题，同时还存在输入数据个数难以确定，容易陷入局部最优和网络结构难于确定等问题。而RNN的内在机理就具备了时序的概念，可以实现时间序列一次多个输入输出，而且隐藏层之间实现自联接；展开后相当于时序之间影响，不过同时，神经网络也具备一定的缺点：梯度下降快，无法稳定收敛到最优解等。

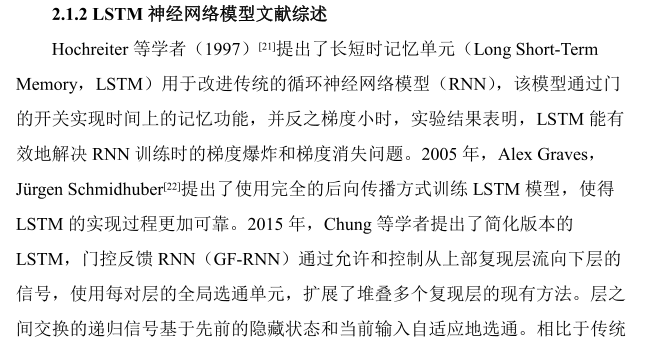
LSTM神经网络的实现使这些问题有了进一步解答。

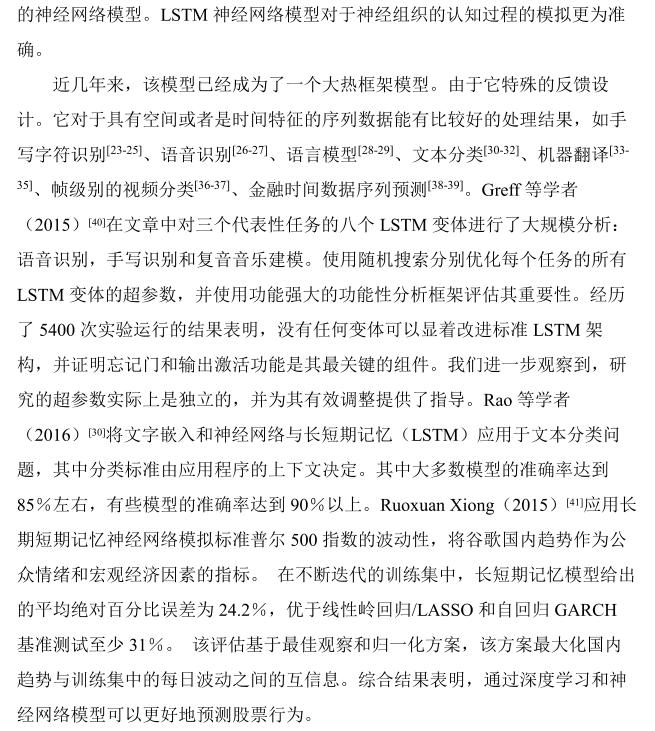
LSTM可以对之前的输入输出进行记忆，选择性传导到下一输出过程中。其次，时序序列之间的内部影响的特征都使得它适用于股票价格时间序列这类随机的非平稳时间序列。

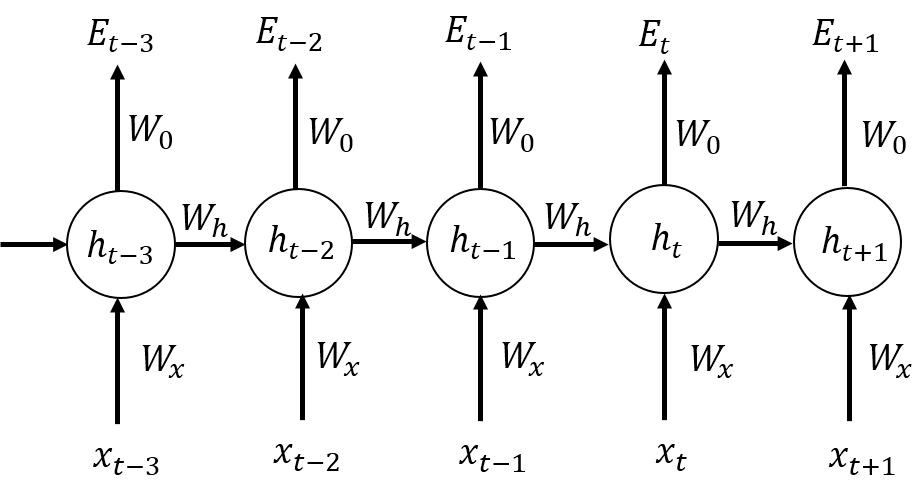
研究意义

自股票存在的那天起，诸多学者们便一直在对股票市场的变化进行摸索和研究，而诸多实践结果表明股票价格或者说收益率在短期内就有一定的预测性。







RNNは

RNNの概要は前の図　で示した通り、

はｔ時刻の入力

はｔ時刻の中間層の出力

は誤差

は入力層から中間層の重み

は中間層同士の重み

は中間層から出力層への重み

時刻ｔの中間層の計算をする際に入力層からの入力と前の時刻の中間層の出力に重み付けを行った結果を加算したものを中間層への入力として扱う。この際にポイントが2つある。

一つ目は、一般的なRNNでは時刻のｔ＝0のときには時刻t-1の中間層が存在しないため、過去の時間からの入力はすべての値が0のものを時刻t-1の中間層からの入力として設定する。

2つ目は、入力層から中間層への重み、中間層から中間層への重み、中間層からその上の層への重みは別の時間軸関でそれぞれ同じものを用いる。すなわち、一つのニューラルネットワークを構築して、その中間層の計算にとある特定の重みを用いて時間展開をしていくということになる。その際の活性化関数としてはよくtanhが用いられる。

そのため、ある時刻tにおける中間層からの出力を、時刻tにおける入力、時刻入力層から中間層への重みとバイアス、中間層から中間層への重みを用いて書くと以下のようになる。

RNNでは各時刻が重みを共有して、中間状態

t时刻の誤差関数 定义为交叉熵损失

在这里，是时刻t上正确的值， 为预测出来的值。使用一整个序列作为一次训练，每个时刻的误差累计构成总的误差。

だから、時刻tにおける誤差をとすると、合計誤差は次のように表すことができる。

これから、すべての重み、、、についてそれぞれの勾配を計算する

まずは、連鎖律を用いると、中間層から出力層への重みに関する誤差関数の偏微分は以下のようになる。

次は、中間層から中間層への重みについての勾配を計算すると以下のようになる。

はｔ時刻の中間層の出力を表すため、式（１）によると、はの関数であり、に関する計算はとに依存する。また前の時刻t-1のときに、であり、 に関する計算はとに依存する。だからを展開する必要がある。

を展開すると

そして、を展開すると、

式（８）を式（7）に代入すると

これにより、式(6)のという部分が展開された。

式（9）を式（6）に代入すると

最後に、入力層から中間層への重み重みに関する誤差関数の偏微分を計算する。の計算と同様に、連鎖律を用いると、重みに関する誤差関数の偏微分は以下のようになる

を展開すると、

-

RNNの勾配に関する問題点

先ほど計算した勾配の特性上、普通のRNNは長時間のデータを意識した学習をすることができないという問題が起こる。

重みに関する誤差関数の偏微分との計算に関しては、時刻tだけではなくて、時刻kからtまで、すべての時刻を一緒に計算する必要があるから、また、総乗の部分が入っているため、時刻ｊと時刻ｋの距離が大きく離れた場合は勾配爆発と勾配消失の問題が発生する。

なぜなら、の計算結果は

tanh関数の導関数は｛｝

图 tan 函数和tan函数的导数

よって：

勾配爆発が発生するの原因は

>4のとき、 、もしJとKの間の距離が大きすぎると、連乗も多すぎて，勾配爆発が発生し，無限に近づいてしまう。

勾配消失が発生する原因は

<4のとき、であるため、距离过大，会导致连乘项过多，产生梯度消失，趋近于0

勾配消失が起こると、ある程度先の時刻以降の誤差は全く考慮しないで重みを更新するという現象が起こっている。実際には4－５時刻程先の誤差を考慮した勾配を計算するとほぼ0になる。

この勾配消失の問題に対する対策として、簡単な方法としては活性化関数としてReLUを用いることが挙げられる。

しかし、ReLUは正の領域での勾配が常に１になるので勾配消失が起こりにくくなる。しかし、マイナスの値を全て0にしてしまうので長期時刻順伝播させる際に値がほとんど0になってしまう可能性がある。

RNNもう1つの問題点として、入力層―中間層、中間層―中間層、中間層―出力層で常に共通の重みを用いるために、重要な入力を通すために重みを大きくするように学習が進んでしまうと逆に時系列上にある不必要な情報も大きく通すようになっていまい、

そのため、RNNの重みは学習の際に常に矛盾したアップデートをされていることにより、学習がなかなか進まない。

これらの問題を、時間展開される中間層の構造を変化させることにより回避したのがLong　Short　Term　Memory（LSTM）である。

改良したRNN―LSTM

LSTM的定义和结构

LSTMはHochreiter&Schmidhuber在1997年提出的，目前在NLP领域已经取得了很好的应用效果，远远优于其他算法。

従来のRNNの中間層出力は前の時刻の出力と今の入力の重み付き線形和に活性化関数を作用されるだけの形だったため、時系列が長くなればなるほど活性化関数にネストされて、勾配消失が起こる。

LSTMは勾配爆発問題と勾配消失問題をCEC（Constant　Error　Carousel）とゲートという概念を導入するにより解決することができる。

CECは過去の情報を保存するためのユニットである。例えば、前の時刻ｔ－１の出力　と今の入力　の重み付き線形和をそのまま通すではなく、前の時刻ｔ－１のCECの値を加算して出力される。

ゲートに関しては入力ゲート、忘却ゲートと出力ゲートという3つのゲートの導入することによって各時刻ごとに入出力を制御することが可能になる。

例えば、ｔ時刻のRNNの入力ｘｔと中間層の出力ｈｔ－１しかないのに対して、LSTMのゲートを制御するファクターが3つある。一つ目はｔ時刻における入力ｘｔ、もう1つは前の時刻ｔ－１における中間層の出力ｈｔ－１、そして最後がCECの値Cｔ－１とCｔである。

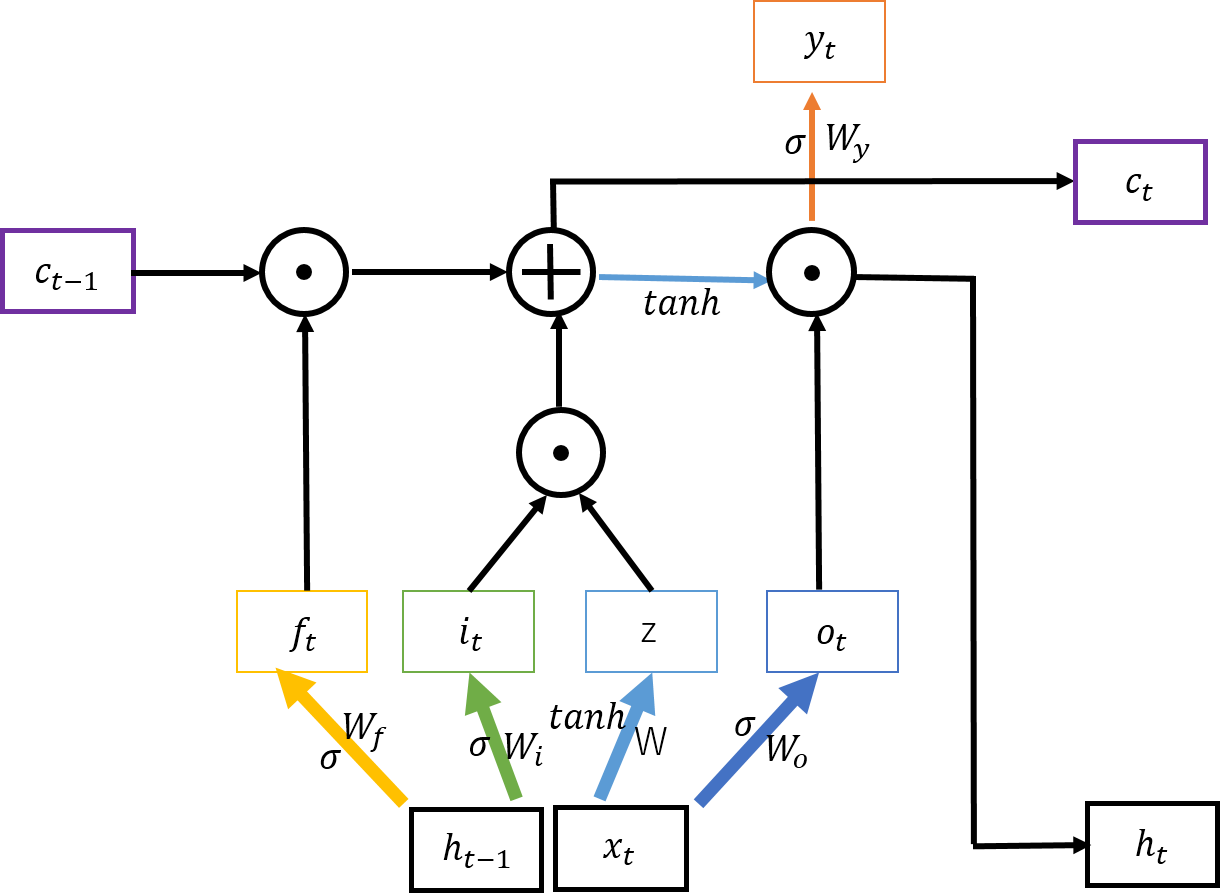
具体的にそれぞれのゲートの式は以下のようになる。

忘却ゲート

入力ゲート

入力データ

出力ゲート



⊙はアダマール積であり、

、、の計算に関してはそれぞれt時刻における入力と前の時刻における中間層の出力の重み付き線形和にシグモイド関数を作用され、0.0～1.0の間の値に調整された。ゲートの値が1.0に近い場合はそれぞれの入力を通過させ、0.0.に近い場合はシャットダウンする。

また、はtanh関数を作用されることにより－1.0～1.0の間の値になるように調整された。はただの入力データであり、ゲートを制御するわけではない。

これから、忘却ゲートを作用させた前の時刻のCECの値Cｔ－１と入力ゲートを作用させた入力を時刻ｔの入力を加算してCECを更新する

それに活性化関数TANHを作用させた上で、出力ゲートを作用させて中間層出力とする。

⊙tanh()

最後の結果の出力は

=

これがLSTMの仕組みである。

LSTM为什么解决了梯度消失，数学分析。