一、研究背景

従来型の時系列予測の研究に関しては主に統計分析手法と計量経済学モデルを用いて行っている。1980年代以降、金融市場の分析に機械学習の手法が徐々に適用されるようになった。しかし、複雑な関数近似をしなければ分類や回帰ができない場合に、従来型の機械学習手法ではうまくいかないケースも多く、このような問題に対して、ディープラーニング手法を使用するケースが増えている。それに、近年ではディープラーニングの領域がブームになり、その中でベースの手法はニューラルネットワークである。

ニューラルネットワークは脳の神経回路をモデルした機械学習手法の一つである。生きもの脳はニューロンと呼ばれる細胞が膨大な数集まって構成されている。そしてそれぞれのニューロンが電気信号を受け取ったり、出力したりして脳は活動している。ニューラルネットワークの場合はモデル化されたニューロンが複数の入力を受け取って、それらの値を重み付きで足し合わせて、出力する。一つひとつの人工ニューロンは単純な仕組みだが、それを多数組み合わせる事で複雑な関数近似を行うことができる。

ニューロンを横に並べたり多層化したものがニューラルネットワークになる。ニューラルネットワークの各層には名前がある。入力データが入って来る層のことを入力層、最終的に求めたい値が出てくる層を出力層、それ以外の中間にある層のことを中間層と呼ぶ。

　ディープラーニングの手法を用いることにより、従来に比べて飛躍的に認識精度が向上するケースもあり、現在世の中でディープラーニングはたいへん注目されています。例えば、最近ではレコメンデーションや自動運転の分野など、幅広い分野で利用されている。

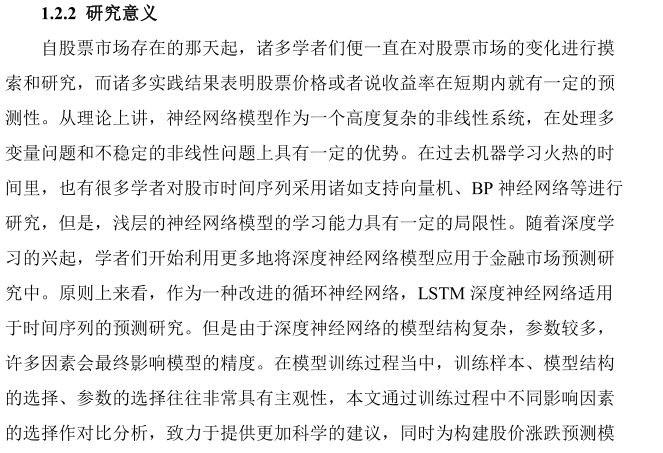
しかし、時系列データの予測に関しては、入力は一つ一つ一連の入力データとして扱われ、それぞれに対する一連の出力データが必要となる。単純なニューラルネットワークでは、出力は出力層からだけ発生するため、一連の入力一つ一つに対して出力を生成することはできない。通常のニューラルネットワークでは、ある層の出力は、次の層の入力に利用されるのみであり、時系列データを自然に扱うことができない。Recurrent Neural Network（RNN）では、ある層の出力は、次の層の入力として利用されるだけでなく、一般的なニューラルネットワークの出力層のような出力としても利用される。また、各層の入力として、前の層の入力のみではなく、時系列のデータポイントも入力とする。

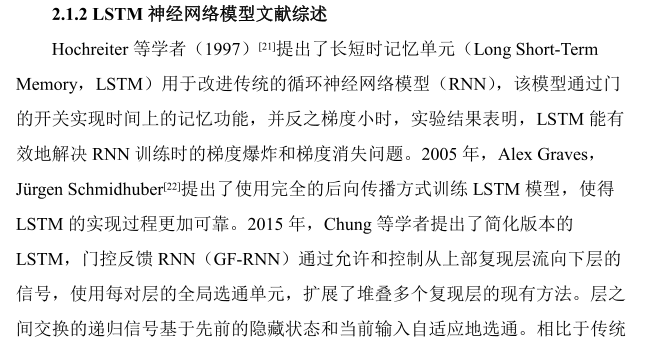
だから、RNNは時系列データの予測問題を解決することができる。

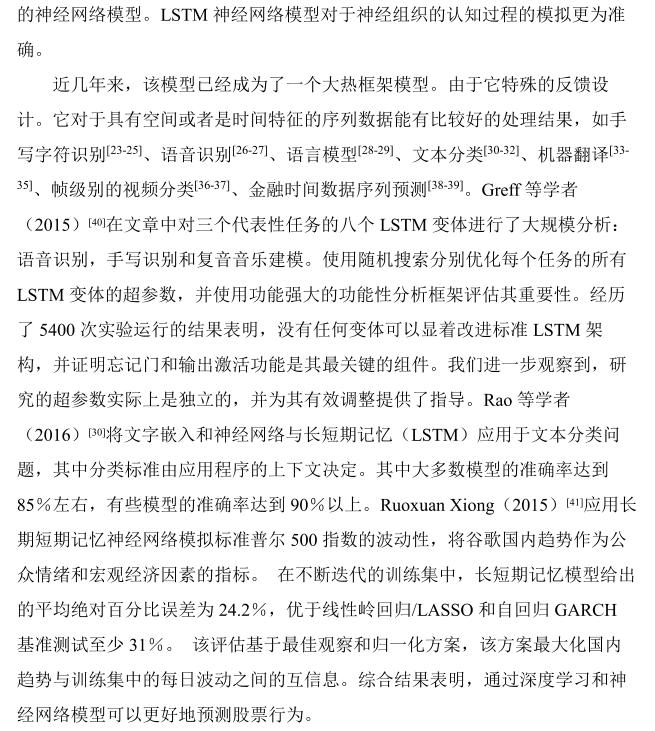
RNNの内部メカニズムには、時系列の複数の入出力を一度に実現し、隠れ層間の自己接続を実現できるタイミングの概念があります。拡張後は、タイミング間の影響に相当しますが、ニューラルネットワークにはいくつかの欠点もあります。 ：勾配降下は高速で、最適なソリューションに着実に収束できません。

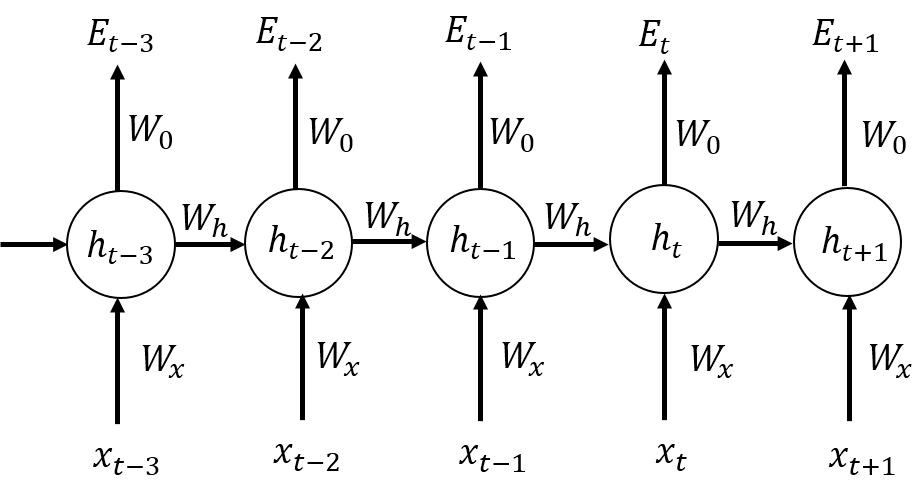
LSTMニューラルネットワークの実装は、これらの質問にさらに答えています。

LSTMは、以前の入力と出力を記憶し、それを次の出力プロセスに選択的に実行できます。第二に、時系列間の内部影響の特性はすべて、株価時系列などのランダムな非定常時系列に適しています。，







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関数の導関数の値の範囲がはであるため

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

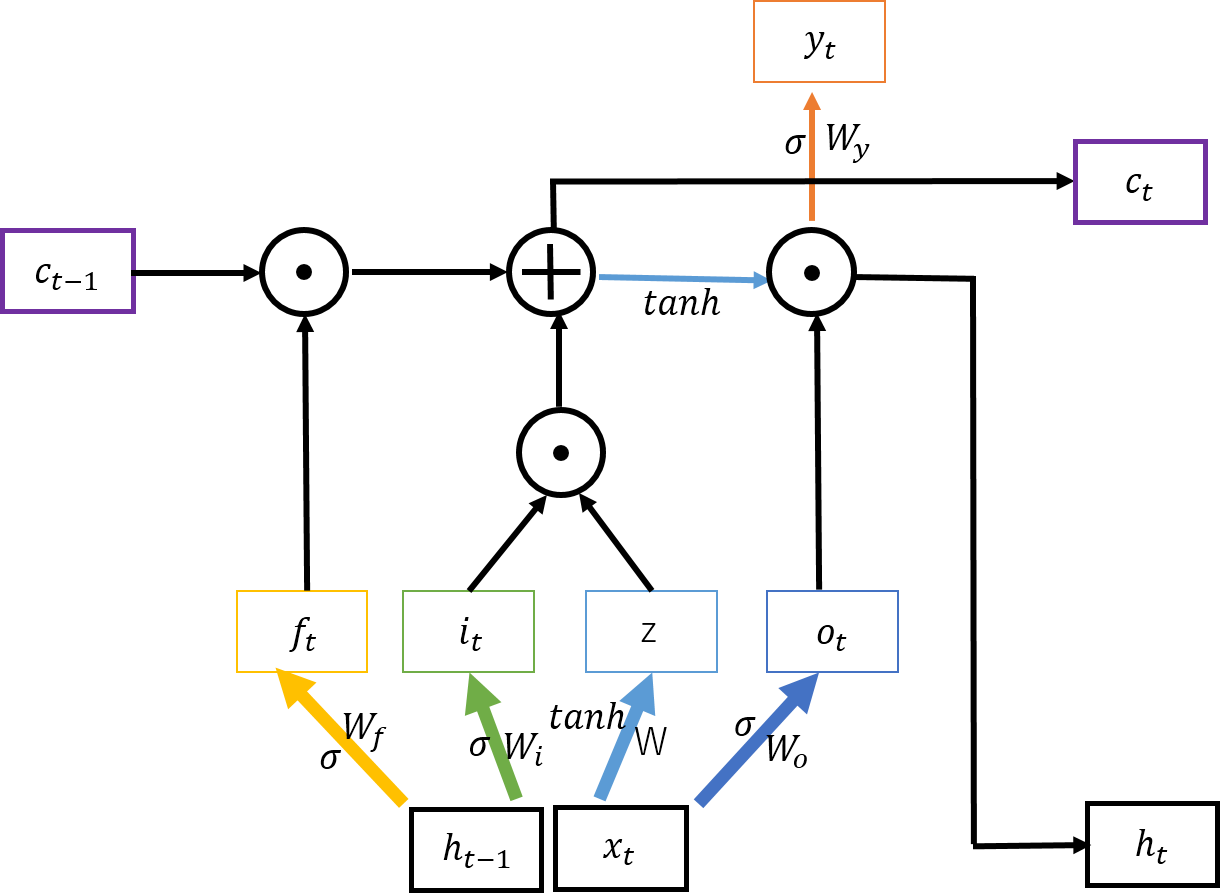
所谓的遗忘门就是要决定从记忆细胞中是否丢弃某些信息；通过一个sigmoid函数处理；遗忘门接受来自输入 和上一层隐状态 的值进行加权计算处理；

入力ゲート

入力データ

更新们就是需要确定什么样的信息能够存入细胞状态中, 还需要通过tanhtanh 计算记忆细胞的候选值 ˜c(t)c(t)~；

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。