Redes Convolucionais para Séries Temporais

Gabriel Mayrink

Agosto 2022

Introdução

Os dados financeiros, geralmente, são séries temporais e os modelos que mais utilizamos para tentar prever o comportamento de algum ativo levam em consideração a variável tempo. Para as arquiteturas de Redes Neurais mais simples os dados precisam ser estáticos, existem modelos de Redes Neurais mais complexos que levam o tempo em consideração, como por exemplo Redes Neurais Recorrentes(RNN), porém nesse trabalho tentaremos modelar e prever o comportamento de um ativo financeiro utilizando uma arquitetura que não leva o tempo em consideração, para isso precisaremos transformar os dados.

Metodologia

A ideia principal do trabalho é transformar a série temporal em uma imagem e depois usar uma Rede Neural Convolucional para lidar com esses novos dados. Veremos se esse modelo é capaz de prever o comportamento de um ativo e também vamos comparar com modelos mais simples.

Usaremos como referência o artigo "Deep Learning for Financial Times-Series Data Analytics: An image processing Based Approach e o estudo é divido em 7 partes: Data Collection, Data Preparation, Data Transformation, Data Labeling, Modeling, Performance Evaluation and Comparison e Going Further.

1 Data Collection

Os dados utilizados são os dados diários do câmbio entre as moedas Euro e Dolar, no período entre 08/12/2000 a 13/09/2018 para tentar prever o comportamento do ativo no período entre 03/01/2000 a 14/09/2018. Coletamos os dados no site https://finance.yahoo.com/ e os dados incluem as seguintes séries temporais: Abertura, Fechamento, Máximo, Minimo, Variação(%).

Como veremos na parte de Data Transformation precisamos de 18 dias(de funcionamento do mercado) antes do dia 03/01/2000 para poder calcular os indicadores financeiros, portanto os dados coletados são do dia 08/12/1999 até o dia 14/09/2018.

2 Data Prepation

É comum, em estudos que utilizam séries temporais, o uso de tecnicas para dividir os dados em janelas de tempo. Dividiremos os dados em 8 cenários e cada um dos cenários é dividido entre conjunto de treino e conjunto de teste. Usaremos a técnica de sliding windows nos 6 primeiros cenários e expanding windows nos 2 últimos cenários. A janela de tempo nos 6 primeiros cenários é de 8 anos, onde os 7 primeiros serão usados para o conjunto de treino e o último para conjunto de teste, além disso o deslocamento entre as janelas é de 2 anos. Como usaremos a tecnica de expanding windows nos 2 últimos cenários, então o 7° cenário terá uma janela de tempo de 11 anos, onde os 9 primeiros serão usados para treino e os 2 últimos para teste. O 8° cenário terá uma janela de 19 anos(o período todo), onde os 17 primeiros anos serão usados para treino e os 2 últimos para teste.

3 Data Transformation

A partir dos dados criaremos uma tabela 5x5 onde as 4 primeiras linhas são formadas pelo cálculo de 4 indicadores financeiros relativos à série e a última linha terá dados relativos ao gráfico de candlestick, além da trend de 1 dia.

Os indicadores utilizados serão:

- Relative Strength Index(RSI)
- Stochastic Oscilator(SO)
- Average True Range(ATR)
- Exponential Moving Average(EMA)

Os indicadores são calculados relativos à própria série e usam como parâmetro quantos dias para trás usaremos para o cálculo. A quantidade de dias usados são 3, 5, 8, 12 e 18, portanto a tabela fica como na figura 1.

RSI(3)	RSI(5)	RSI(8)	RSI(12)	RSI(18)
SO(3)	SO(5)	SO(8)	SO(12)	SO(18)
ATR(3)	ATR(5)	ATR(8)	ATR(12)	ATR(18)
EMA(3)	EMA(5)	EMA(8)	EMA(12)	EMA(18)
Abertura	Fechamento	Máximo	Mínimo	T1D

Figure 1: Estrutura da tabela

Após a criação da tabela, cada entrada é passada por uma função de normalização para que o valor de cada entrada seja levado em um valor no intervalo $[0, 255]$ que é o valor do cinza no pixel.

A função usada para isso é essa:

$$v_{px,i,j} = (v_{entrada,i,j} - min_{entrada,i,j})/(max_{entrada,i,j} - min_{entrada,i,j})(255-0)+0$$

onde $v_{entrada,i,j}$ é o valor da entrada (i, j) da tabela, $min_{entrada,i,j}$ é o valor minimo que a entrada (i, j) da tabela recebe, $max_{entrada,i,j}$ é o valor máximo que a entrada (i, j) tabela recebe e $v_{px,i,j} \in [0, 255]$ é o resultado da normalização.

Como resultado obtemos uma imagem 5x5 como na figura 2

Figure 2: Exemplo de imagem criada

4 Data Labeling

O objetivo do estudo é modelar o comportamento do ativo para poder prever seus futuros valores e então lucrar, logo o label de cada imagem para hoje(dia) é dado da seguinte forma:

$$\begin{cases} 1, & Fechamento[dia+1] - Abertura[dia+1] > 0 \\ 0, & Fechamento[dia+1] - Abertura[dia+1] \leq 0 \end{cases}$$

Ou seja, se amanhã(dia + 1) o preço sobre então devemos comprar(1) hoje(dia), e se amanhã(dia + 1) o preço desce então devemos vender(0) hoje(dia)

5 Data Modeling

Após os cenários, com as imagens e os labels, serem criados, em cada cenário, o conjunto de treino é passado no modelo para treiná-lo e em seguida o conjunto de teste é usado para verificar a acurácia.

Usaremos como modelo uma Rede Neural Convolucional com a seguinte arquitetura:

- 1 camada convolucional(20 features maps, kernel=(2,2), stride(1,1), função de ativação Relu)
- 1 camada de maxpooling(2,2)
- 1 camada fully connected(20 neurônios, função de ativação Relu)
- 1 camada de dropout(0.25)
- 1 camada fully connected(20 neurônios, função de ativação Relu)
- 1 camada output(2 neurônios, função de ativação softmax)

O algoritmo de otimização utilizado é o Adam com learning rate de 0.001
O batch size foi de 32
e
Foram usadas 100 épocas para cada conjunto de treino.

6 Performance Evaluation

Os resultados obtidos nos cenários estão na tabela abaixo:

Scenario	Training Data	Test Data	Train Accuracy	Test Accuracy
1	2000-2007	2008	0.595512	0.467181
2	2002-2009	2010	0.591680	0.513410
3	2004-2011	2012	0.595512	0.523077
4	2006-2013	2014	0.590685	0.555556
5	2008-2015	2016	0.590038	0.494253
6	2010-2017	2018	0.571194	0.480769
7	2000-2008	2009-2010	0.584851	0.531549
8	2000-2016	2017-2018	0.609008	0.492337

Figure 3: Resultados do modelo nos cenários

	Train Accuracy	Test Accuracy
count	8.000000	8.000000
mean	0.577502	0.517159
std	0.014167	0.027171
min	0.556103	0.478927
25%	0.567283	0.500033
50%	0.579682	0.519157
75%	0.584452	0.536717
max	0.597508	0.553846

Figure 4: Resultados do modelo nos cenários

Podemos ver na figura 5, por exemplo, o plot da acurácia no conjunto de treino e no conjunto de teste do cenário, conforme as épocas passam 2.

Através dos resultados e dos plots podemos ver que a Rede tem uma tendência a overfitting, porém a acurácia média no conjunto de teste é maior que 50% o que não é um resultado ruim, pois se assumirmos que os ganhos e as perdas são iguais então no longo prazo o modelo gerará lucro.

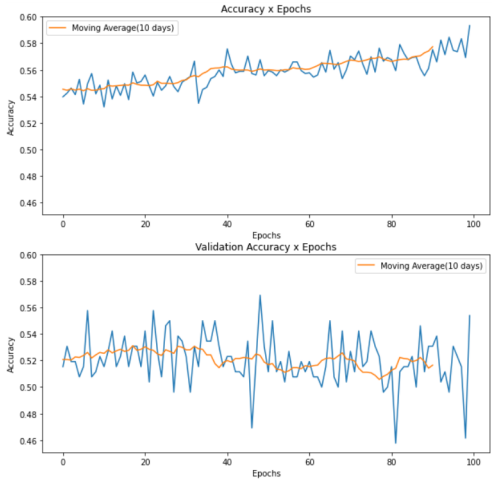

Figure 5: Plots das acurácias no conjunto de treino e teste do cenário 2

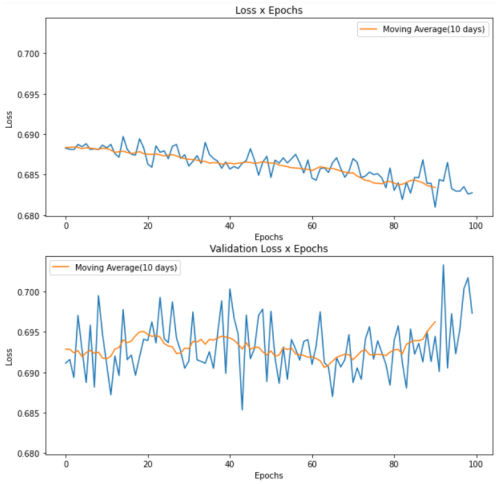

Figure 6: Plots das acurácias no conjunto de treino e teste do cenário 2

7 Performance Comparison

Usamos 2 modelos mais simples para a comparação:

1. Persistence Model
2. Autoregressive Model

O primeiro é o modelo mais simples que podemos pensar: Prevemos para o dia seguinte exatamente o que aconteceu hoje, ou seja, se hoje sabemos que temos que comprar então amanhã também compraremos e vice-versa.

O resultado que obtivemos é de 46,97% de acurácia média, um resultado bem pior que nosso modelo porém esperado dado sua simplicidade.

O segundo modelo que usamos para comparação é o autogressivo. No modelo autoregressivo assumimos que o valor da serie no tempo t é explicado, de forma linear pelos valores de p tempos anteriores:

$$X_t = c + \sum_{n=1}^{p} \phi_i X_{t-i} + \epsilon_t$$

No nosso caso usamos um modelo autoregressivo com 3 dias para trás e o resultado obtido foi de 48,50% de acurácia, um resultado um pouco melhor que o persistence model porém menor que 50% e pior que o nosso modelo.

8 Going Further

Uma última etapa do estudo foi tentar experimentar um modelo de Redes Neurais Convolucionais que possui a mesma arquitetura do modelo que criamos porém com previsão de 2 dias para frente.

Podemos ver os resultados desse modelo nas tabelas abaixo:

Scenario	Training Data	Test Data	Train Accuracy	Test Accuracy
1	2000-2007	2008	0.517241	0.552124
2	2002-2009	2010	0.542419	0.551724
3	2004-2011	2012	0.548440	0.519231
4	2006-2013	2014	0.562192	0.471264
5	2008-2015	2016	0.566502	0.471264
6	2010-2017	2018	0.578313	0.423077
7	2000-2008	2009-2010	0.539310	0.544933
8	2000-2016	2017-2018	0.591279	0.511494

Figure 7: Resultados do modelo com 2 dias de previsão nos cenários

	Train Accuracy	Test Accuracy
count	8.000000	8.000000
mean	0.555712	0.505639
std	0.023643	0.046641
min	0.517241	0.423077
25%	0.541642	0.471264
50%	0.555316	0.515363
75%	0.569455	0.546631
max	0.591279	0.552124

Figure 8: Resultados do modelo com 2 dias de previsão nos cenários

Como no caso de previsão de 1 dia, a Rede aparenta estar fazendo overfitting, e a acurácia média é de mais ou menos 50% o que não é muito bom para lucrar.

Podemos ver no plots abaixo a acurácia e os erros nos conjuntos de treino e teste conforme a Rede treina.

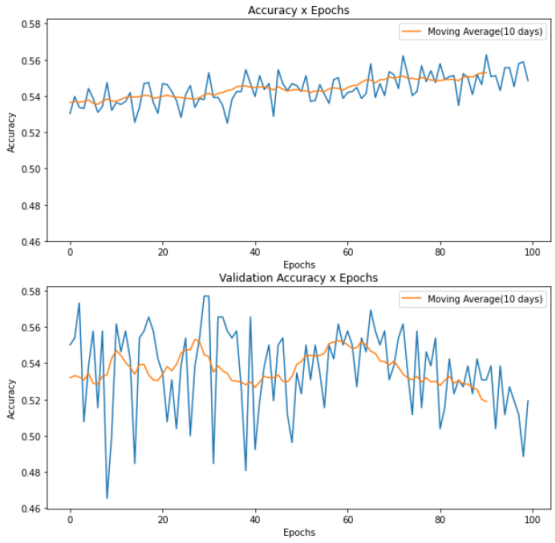

Figure 9: Plots das acurácias no conjunto de treino e teste do cenário 2 na Rede com 2 dias de previsão

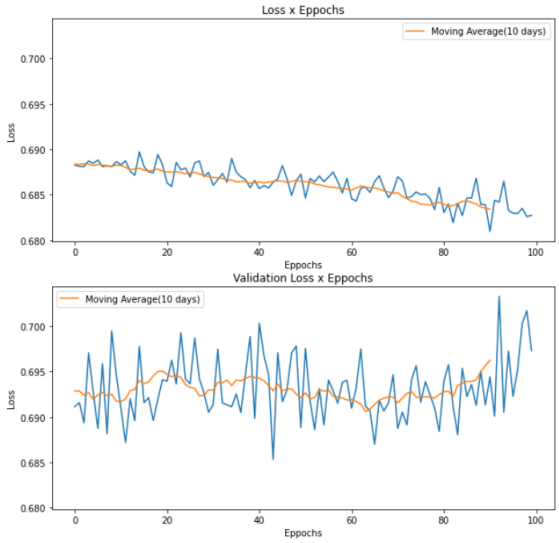

Figure 10: Plots das acurácias no conjunto de treino e teste do cenário 2 na Rede com 2 dias de previsão