Experimento 01 - Comparação entre algoritmos de busca/otimização de hiper-parâmetros

Augusto César Benvenuto de Almeida

Abstract

Neste experimento foi realizado uma comparação entre diversas técnicas de busca de hiper-parâmetros, entre elas podemos destacar o Particle Swarm Optimization (PSO) e o Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Analisando os dados coletados e aplicando testes estatísticos não-paramétricos é possível perceber que existem claras diferenças entre o desempenho dos algoritmos. Entre os testados, os que apresentaram melhor desempenho foi o PSO, Random Search e Grid Search. Porém o PSO conseguiu sempre ser o mais rápido dentre os métodos testados.

1. Motivação

Como o espaço de busca dos parâmetros para cada técnica de Machine Learning costuma ser contínuo e de alta dimensionalidade, para atingir os valores ótimos é necessário uma busca exaustiva de baixa granularidade, demandando assim um alto custo computacional. Para resolver esse problema existem técnicas que servem para realizar essa busca de maneira a diminuir os custos de tempo e processamento.

2. Objetivo

Este experimento tem como objetivo verificar quais das funções disponíveis atingem o melhor resultado dentre as séries de retorno das ações componentes do índice IBOVESPA.

As técnicas a serem comparadas serão:

2.1. Random Search (RS)

Random Search é a técnica mais simples de todos os algoritmos testados, nela os parâmetros são gerados de forma aleatória seguindo uma distribuição uniforme num dado limite.

2.2. Grid Search (GS)

Grid Search é um método de busca não direcionado que consiste em testar um conjunto de valores pré-definidos para cada parâmetro. Uma grade é construída a partir do produto cartesiano desses conjuntos de valores.

Essa técnica também é bastante simples, porém permite que os parâmetros sejam gerados em diversas combinações diferentes.

2.3. Particle Swarm Optimization (PSO)

Particle Swarm Optimization [1] [2] (PSO) é um método heurístico de otimização, simula um conjunto de partículas que se movem no espaço de busca.

2.4. Nelder-Mead (NM)

Nelder-Mead [3] é um método numérico utilizado para encontrar os parâmetros que minimizam uma função objetivo em um espaço multi-dimensional.

Apresenta uma limitação pois precisa de um ponto inicial para iniciar a busca, o que pode levar a convergências de ótimos locais.

2.5. Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Covariance Matrix Adaptation Evolutionary Strategy [4] [5] é um algoritmo evolucionário para otimização de funções contínuas, ele consegue se adaptar dinamicamente ao espaço de busca, permitindo assim buscas eficientes.

3. Metodologia

Primeiro passo para realizar o experimento foi conseguir a base de dados, para tal foi utilizado o sistema do Yahoo! Finance [6] para conseguir as séries do IBOVESPA (43 dos 70 componentes do índice).

Os dados adquiridos apresentam dados temporais e diversos dados financeiros, será utilizado e avaliado no experimento e tese apenas a série de preços ajustados de fechamento, ajustes necessários para equalizar o preço no tempo devido a ações de desdobramento ou grupamento [7].

Vale notar que os dados utilizados apresentam apenas valores colhidos nas quartas-feiras, isso foi feito para evitar um efeito[8, p. 404] conhecido no mercado no início e fim de semana.

Com as séries em mãos é necessário realizar uma transformação nos dados para conseguir a série de retornos, a operação aplicada é:

$$r_t = \frac{p_t - p_{t-1}}{p_{t-1}} \tag{1}$$

Após calculada, precisamos ainda criar a matriz contendo os padrões de entrada e saídas para os algoritmos de Machine Learning, nessa fase é necessário um parâmetro p extra responsável pela dimensão do problema, nesse caso é a quantidade de lags usados para prever o próximo valor.

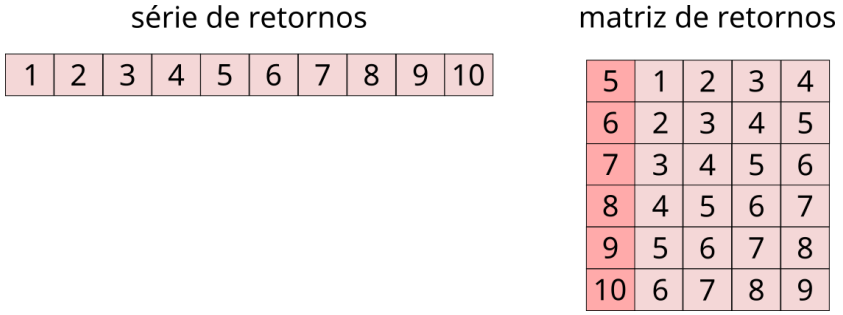

Figure 1: Exemplo de matriz gerada com parâmetro $p = 4$, valor de saída na primeira coluna e de entrada nas demais.

Para conseguir alguma relevância estatística serão feitas 50 execuções de um cross-validation (CV) proposto por Hyndman [9] para cada algoritmo de busca, a variável a ser analisada será o valor da raiz quadrada do erro médio quadrático (RMSE) resultante de cada de cada CV.

Para exemplificar, imagine que a matriz de retornos seja dividida em k-folds (5, por exemplo) sem que haja permutação dos padrões para que não ocorra perda de informação de autocorrelação entre os dados, consequência inerente das séries temporais.

Então acontecerão 4 processos de treinamento/teste de forma sequencial, no primeiro passo usa-se o fold-1 para treinamento do regressor e o fold-2 para teste, é então calculado o RMSE do conjunto de teste e seu resultado é salvo. No próximo passo o fold-2 é agregado ao anterior e ambos são usados como treinamento para testar o fold-3, e assim por diante. Desse jeito é possível representar o comportamento temporal no cross-validation e se aproximar do que ocorre no mundo real. O resultado final do CV é exatamente a média dos RMSEs calculados nessa etapa.

Exemplo:

1 : folds treinamento [1], fold teste [2]

2 : folds treinamento [1 2], fold teste [3]
3 : folds treinamento [1 2 3], fold teste [4]
4 : folds treinamento [1 2 3 4], fold teste [5]

Após a execução das 50 iterações para cada algoritmo de busca será gerada então uma matriz de 50 linhas e 5 colunas contendo todos os resultados obtidos, a primeira análise feita é se a distribuição dos valores de RMSE de cada técnica é igual, para tal foi usado um teste estatístico não-paramétrico, o **Kruskal-Wallis H-test**. Esse teste é o equivalente não-paramétrico do ANOVA e sua hipótese nula afirma que as distribuições dos grupos são iguais.

Se a hipótese for rejeitada é criada então uma matriz de 5 linhas e 5 colunas para verificar onde estão as diferenças entre os algoritmos, cada elemento da matriz é o resultado de um teste não-paramétrico entre os pares de técnicas i e j, o teste é o mesmo realizado anteriormente, porém quando o número de grupos é 2 (teste pareado) ele é conhecido por **Mann–Whitney U test**, equivalente não paramétrico do t-test.

Exemplo:

	rs	gs	pso	nm	cma-es
rs	0	0	1	0	1
gs	1	0	1	0	1
pso	0	0	0	0	0
nm	1	1	1	0	1
cma-es	0	0	0	0	0

Table 1: Matriz contendo o resultado dos testes pareados

Se a matriz for criada, ou seja, existe diferença estatística entre os algoritmos, é calculado um vetor com a soma de cada linha da matriz para criar um ranking dessa técnica em relação as outras.

A matriz é formada por 0's e 1's, onde uma linha representa o desempenho de uma técnica em relação as outras, se for 0 temos a possibilidade de que as técnicas não apresentam diferença estatística entre si ou então o i-ésimo algoritmo é melhor que o j-ésimo (apresenta um RMSE médio menor, nesse caso melhor pois tratamos de erro), se for 1 só existe a possibilidade que existe diferença estatística entre as técnicas e que a i-ésima apresenta um pior desempenho (RMSE médio maior).

Então cada elemento do vetor varia entre 0 e 4, onde o primeiro representa

que aquele algoritmo não foi pior que nenhum outro e 4 indica que ele foi pior que todos os outros.

Esse experimento foi feito utilizando 3 diferentes regressores, para verificar se os resultados eram consistentes entre diferentes técnicas de Machine Learning, foram escolhidos o **elmK** [10], **elmR** [10] e **svr** [11].

4. Resultados

Nesta seção serão apresentados os resultados finais colhidos após processar 43 séries componentes do IBOVESPA nos 3 regressores propostos.

4.1. elmK

Search Function	Ranking Sums
random search	39
grid search	49
particle swarm	27
nelder-mead	163
cma-es	97

Table 2: Soma dos rankings encontrados para as 43 séries

O resultado do *elmK* mostra que o algoritmo que apresentou melhor desempenho foi o PSO, com o Random Search e Grid Search em seguida, respectivamente.

Além disso é feita mais uma checagem para dar uma força estatística ao resultado, é criada uma matriz contendo o vetor de rankings de cada uma das séries verificadas, formando assim uma matriz de 43 linhas e 5 colunas. É feito um teste não-paramétrico para verificar se os rankings dos algoritmos apresentam uma mesma distribuição.

Rankings have same distribution	False

Table 3: Resultado do teste não-paramétrico descrito acima

A hipótese nula foi negada, então é criada uma matriz quadrada contendo os resultados dos testes não-paramétricos pareados. A partir dela, gera-se um vetor final contendo o ranking definitivo.

Com isso temos o resultado final:

Search Function	Rank	Rank Mean	Time Mean
particle swarm	0	0.6279	1.6810
random search	0	0.9069	1.7933
grid search	1	1.1395	1.8438
cma-es	3	2.2558	1.6764
nelder-mead	4	3.7907	0.1941

Table 4: Análise final para o *elmK*

O PSO e Random Search apresentaram os melhores resultados, apesar de não existir uma diferença estatística entre os dois, o primeiro conseguiu uma menor média nos rankings e um menor tempo médio de execução.

4.2. elmR

Search Function	Ranking Sums
random search	76
grid search	17
particle swarm	40
nelder-mead	172
cma-es	63

Table 5: Soma dos rankings encontrados para as 43 séries

O resultado mostra que o algoritmo que apresentou melhor desempenho foi o GS, com o PSO e CMA-ES em seguida, respectivamente. Realizando mais uma vez um teste para verificar se os rankings apresentam uma mesma distribuição temos:

Rankings have same distribution	False

Table 6: Resultado do teste não-paramétrico entre os rankings

A hipótese nula foi negada, com isso temos o resultado final:

O GS apresentou o melhor resultado, em seguida vem o PSO. Esse resultado pode ser entendido pelo fato de no *elmR* o espaço de busca ser menor, apenas 1 dimensão do parâmetro de regularização C, pois o número de neurônios está fixo em 500. Isso deve ajudar um pouco o grid search, deixando a sua busca bem mais refinada do que o PSO. Mas apesar disso o PSO ainda conseguiu ser mais rápido.

Search Function	Rank	Rank Mean	Time Mean
grid search	0	0.3953	3.8715
particle swarm	1	0.9302	3.4902
cma-es	2	1.4651	3.6026
random search	2	1.7674	3.8385
nelder-mead	4	4.0000	0.4778

Table 7: Table caption

4.3. svr

Search Function	Ranking Sums
random search	18
grid search	10
particle swarm	30
nelder-mead	128
cma-es	89

Table 8: Soma dos rankings encontrados para 34 séries

O resultado mostra que o algoritmo que apresentou melhor desempenho foi o GS, com o RS e PSO em seguida, respectivamente. Realizando mais uma vez um teste para verificar se os rankings apresentam uma mesma distribuição temos:

Rankings have same distribution	False

Table 9: Resultado do teste não-paramétrico entre os rankings

A hipótese nula foi negada, com isso temos o resultado final:

Na terceira análise o PSO também fica em segundo lugar, atrás do Grid e Random Search, porém chega a ser 1 ordem de grandeza mais rápido. Isso deve ocorrer pois o *svr* precisa resolver um problema de otimização para conseguir seus vetores de suporte, assim o GS deve entrar em áreas onde os parâmetros pedidos são muito difíceis e de convergência demorada.

Search Function	Rank	Rank Mean	Time Mean
grid search	0	0.2941	38.9919
random search	0	0.5294	5.7148
particle swarm	1	0.8823	2.7880
cma-es	3	2.6176	0.2987
nelder-mead	4	3.7647	0.1305

Table 10: Table caption

5. Conclusão

Fazendo uma análise mais simplificada somando os rankings dos resultados finais temos que o GS tem 1 ponto, enquanto que o RS e PSO atingem 2 pontos, porém o PSO sempre consegue ser o mais rápido, num ambiente dinâmico e de alta frequência como da previsão de séries temporais, talvez seja melhor optar pela velocidade e pegar um método *bom*, quase ótimo.

Todos os códigos, base de dados, resultados e documentos desse experimento estão disponíveis em `http://www.sharelatex.com`

References

[1] J. Kennedy, Particle swarm optimization, Encyclopedia of Machine Learning (2010) 760–766.

[2] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (2002) 58–73.

[3] J. A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal 7 (1965) 308–313.

[4] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2001) 159–195.

[5] Wikipedia, Cma-es — wikipedia, the free encyclopedia, 2014. [Online; accessed 13-April-2015].

[6] Y. Finance, Yahoo finance - business finance, stock market, quotes, news, 2015. [Online; accessed 13-April-2015].

[7] Investpedia, O que é desdobramento (split) e grupamento (inplit)?, 2009. [Online; accessed 13-April-2015].

[8] E. J. Elton, M. J. Gruber, S. J. Brown, W. N. Goetzmann, Modern Portfolio Theory and Investment Analysis, Wiley, seventh edition, 2007.

[9] R. Hyndman, Why every statistician should know about cross-validation, 2010. [Online; accessed 13-April-2015].

[10] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (2012) 513–529.

[11] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik, et al., Support vector regression machines, Advances in neural information processing systems 9 (1997) 155–161.