

OTIMIZAÇÃO EVOLUCIONISTA DE HIPERPARÂMETROS PARA MÁQUINAS DE BOLTZMANN RESTRITAS CONVOLUCIONAIS

Apresentação Final - Projeto e Implementação de Sistemas 2

Guilherme Camargo de Oliveira

2 de julho de 2020

Universidade Estadual Paulista “Júlio de Mesquita Filho” (UNESP)
Faculdade de Ciências (FC) / Departamento de Computação (DCo)
Bauru, SP - Brasil



Sumário da Apresentação

1. Introdução
2. Proposta do Trabalho
3. Fundamentação Teórica
 - 3.1 Máquinas Boltzmann Restrita
 - 3.2 Máquina de Boltzmann Restrita Convolucional
 - 3.3 Otimização por meta-heurísticas
4. Metodologia
5. Resultados dos experimentos
6. Software desenvolvido
7. Conclusão
 - 7.1 Trabalhos Futuros

Introdução

Introdução

Técnicas de aprendizado de máquina têm sido amplamente utilizadas nas mais diversas aplicações, principalmente aquelas baseadas em aprendizado em profundidade. Entretanto, essas técnicas possuem diversos hiperparâmetros que requerem seu ajuste de maneira personalizada para cada base, sendo essenciais para um bom desempenho da técnica.

Problema

O desafio com hiperparâmetros é que não há nenhum número mágico que funcione em todos os lugares. Os melhores números dependem de cada tarefa e de cada conjunto de dados. Podemos dividir em dois tipos:

- Hiperparâmetros de otimização estão mais relacionados ao processo de otimização e treinamento. Exemplos: taxa de treinamento, tamanho do mini-batch, número de épocas;
- Hiperparâmetros do modelo estão relacionados com sua estrutura. O número de unidades ocultas mede a “capacidade” do modelo.

O problema de ajuste de hiperparâmetros em modelos de aprendizado em profundidade como uma tarefa de otimização por meta-heurísticas têm recebido atenção apenas recentemente.



Proposta do Trabalho

Proposta do trabalho

Este trabalho tem como objetivo selecionar os hiperparâmetros de uma CRBM através de duas técnicas meta-heurísticas, Programação Genética e Algoritmo Genético, validando os resultados em dois conjuntos de imagens.

Objetivos Específicos:

- Obter conhecimentos de CRBMs;
- Desenvolver uma rede de Máquinas de Boltzmann Restrita Convolucional;
- Estudar o comportamento da aplicação Programação Genética em CRBMs;
- Estudar o comportamento da aplicação Algoritmo Genético em CRBMs;
- Validar os resultados;
- Desenvolver uma interface gráfica para a ilustração dos resultados.

Fundamentação Teórica

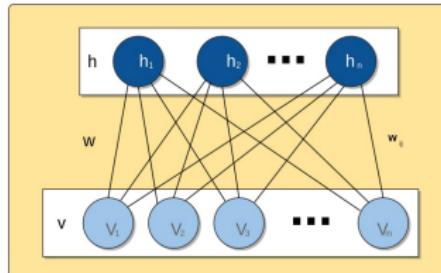
Máquinas Boltzmann Restrita

Máquinas Boltzmann Restritas [3] (RBMs) são redes neurais que pertencem aos chamados Modelos Baseados em Energia.

- Grande popularidade nos últimos anos no contexto do Prêmio Netflix [5], onde as RBMs alcançaram um alto desempenho na filtragem colaborativa e venceram a competição.
- Aplicações de RBMs: na redução de dimensionalidade [10], classificação [7], extração de características [1].

Inventor: Geoffrey Hinton [4]

Figura 1: Arquitetura de uma RBM.



Máquina de Boltzmann Restrita Convolucional

Problema: Imagens (alta dimensão, translação invariável). Inspirado pela Rede Neural Convolucional [8], o modelo apresenta bom desempenho em uma variedade de tarefas de reconhecimento visual. Características da CRBM [9]:

- Semelhante a RBM;
- A camada oculta consiste em K "grupos";
- Os pesos são compartilhados entre todos os locais de uma imagem;
- Cada um dos grupos K está associado a um filtro.

Figura 2: Arquitetura de uma CRBM.



Otimização por meta-heurísticas

Principais características:

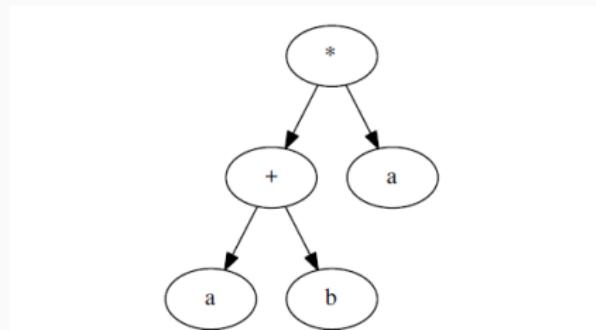
1. Método para resolver uma classe geral de otimização de problemas.
2. Combina medidas de utilidade de forma abstrata e esperançosa, sem utilizar insights mais profundos sobre sua estrutura interna.
3. não exigem hipóteses sobre o problema de otimização nem qualquer tipo de conhecimento prévio sobre a função objetivo. -"caixa preta".
4. obtém conhecimento sobre a estrutura de um problema utilizando estatísticas obtidas de soluções possíveis avaliadas no passado. Este conhecimento é usado para construir novas soluções candidatas que provavelmente tem uma utilidade alta.

Obs: Mesmo em uma única abordagem, há muitos variantes que incorporam diferentes representações de soluções e diferentes técnicas de modificação ou construção de novas soluções.

Programação Genética estrutura

Programação Genética é baseado na Teoria da Evolução de Darwin e foi criado por John Koza 1992 [6]. Considerado método meta-heurístico evolucionista.

- Na programação genética, os indivíduos da população não são seqüências de bits, mas sim programas de computador armazenados na forma de árvores sintáticas.
- Dois tipos de nós: função e terminal.
- Imagine uma árvore de expressão, onde os operadores encontram-se representados pelos nós do tipo função, e os operandos são modelados pelos nós terminais (nós folha):



Programação Genética operadores

Figura 3: cruzamento

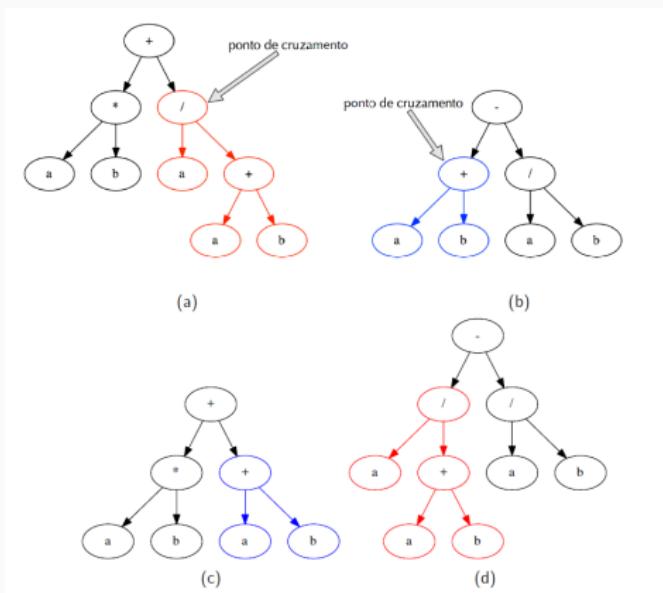
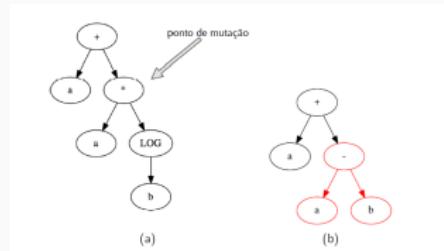


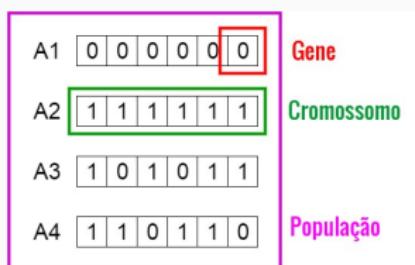
Figura 4: mutação



Algoritmo Genético

Algoritmo Genético criado por J. HOLLAND (1975) [11] com o nome de planos adaptativos, enfatizam a recombinação como o principal operador de busca e aplicam mutação com baixas probabilidades (operador secundário). Operam com representação binária de indivíduos e possuem seleção probabilística proporcional ao *fitness*.

- Tradicionalmente, os indivíduos são representados genotípicamente por vetores binários, onde cada elemento de um vetor denota a presença (1) ou ausência (0) de uma determinada característica: o seu genótipo. Os elementos podem ser combinados formando as características reais do indivíduo, ou o seu fenótipo.



Algoritmo Genético operadores

Figura 5: cruzamento

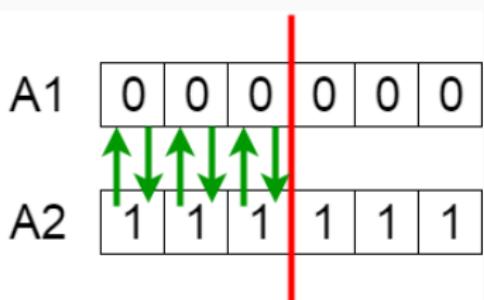


Figura 6: mutação

Antes da Mutação

A5	1	1	1	0	0	0
----	---	---	---	---	---	---

Depois da Mutação

A5	1	1	0	1	1	0
----	---	---	---	---	---	---

Metodología

Metodologia

O projeto pode ser descrito em 3 etapas principais:

- desenvolvimento da rede CRBM,;
- aplicação de GP e GA nos hiperparâmetros da CRBM;
- análise descritiva dos resultados sobre os respectivos conjuntos de dados.

Para desenvolver a Programação Genética e o Algoritmo Genético, iremos utilizar o framework “Distributed Evolutionary Algorithms in Python” [2], que incorpora as estruturas e ferramentas de dados necessárias para implementar as técnicas de computação evolutiva mais comuns, como algoritmo genético, programação genética, estratégias de evolução, otimização de enxame de partículas, evolução diferencial, fluxo de tráfego e estimativa do algoritmo de distribuição.



Metodologia

1. Para validação, foi utilizado os conjuntos de base (Figura 7): MNIST, CalTech, Silhouettes, MPEG e Semeion.
2. Os parâmetros do Algoritmo Genético e Programação Genética foram configurados de acordo com a Tabela 1.
3. Durante a etapa de aprendizagem de uma CRBM, optamos por otimizar três hiperparâmetros: o tamanho do filtro, o número de filtros e os k-passos da amostragem de Gibbs.
4. O critério para comparar o erro e a função custo foi utilizando a medida de Raiz do Erro Quadrático Médio (Root Mean Squared Error - RMSE) entre a imagem de entrada e a imagem de saída de uma CRBM (crença da rede):

$$e_{RMSE} = \sqrt{\frac{\sum_{i=1}^n (v_e^i - v_r^i)^2}{n}} \quad (1)$$

5. Por fim, os resultados obtidos através dessas análises serão exibidos através de uma interface gráfica.

Configuração experimental

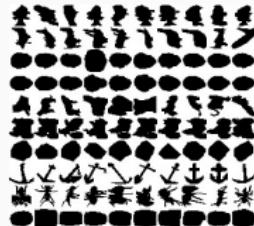
Tabela 1: Configuração dos parâmetros para este projeto.

Parâmetros	Valores
Taxa de cruzamento	0.4
Taxa de mutação	0.1
População por geração	30
Número de gerações	6

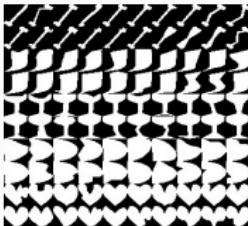
Figura 7: Exemplo de amostras de treinamento das bases.

0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9

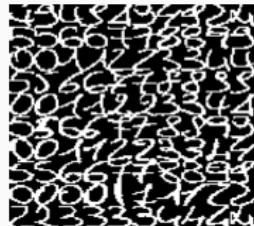
(a)



(b)



(c)



(d)

(a) MNIST, (b) CalTech 101 Silhouettes, (c) MPEG-7 e (d) Semeion.

Desenvolvimento da CRBM

Foi realizado uma busca no github, porém diversos problemas surgiram como projetos muito antigos, não úteis, sem documentação e precisava configurar ambientes com bibliotecas específicas. Resolveu-se, então, começar uma nova implementação em Python 3 utilizando os operadores do Pytorch. Várias versões foram realizadas:

- Versão 1: Reproduzir mesmos resultados da implementação do Mojtaba Solgi¹, mas rodando em Python 3 com Pytorch;
- Versão 2: Otimização Vetorização;
- Versão 3: Otimização Lote.

Cada versão tem uma “subversão”, onde a cada modificação era necessário testar a rede.



Figura 8: Imagens dos filtros da nova rede implementada.

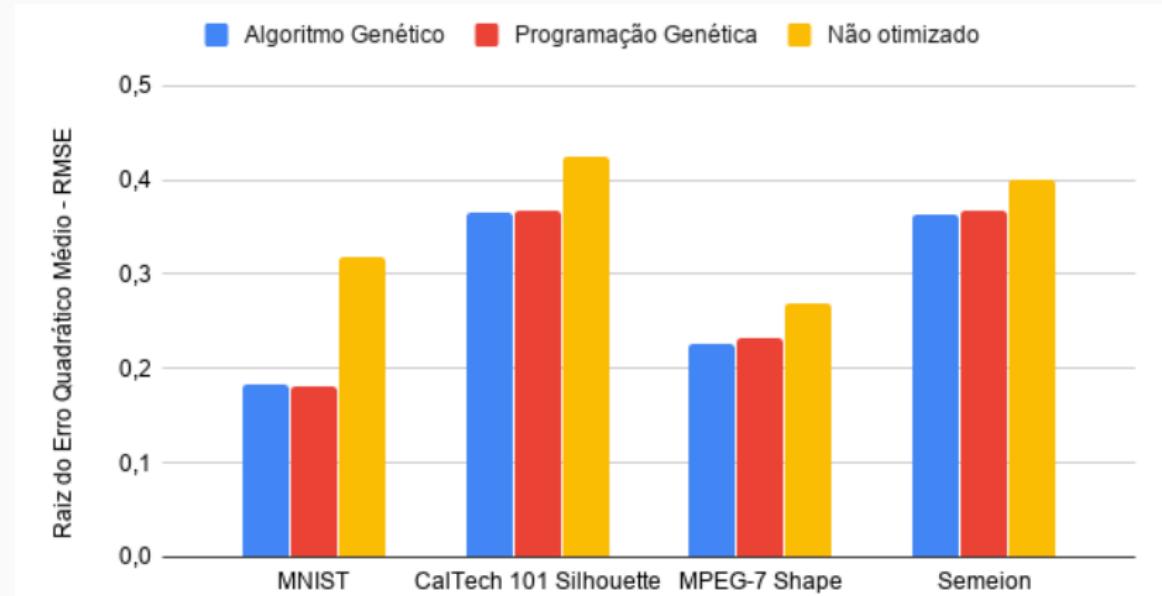


Figura 9: Imagens dos filtros, retirado do artigo[9]

¹<https://github.com/msolgi14/CDBN>.

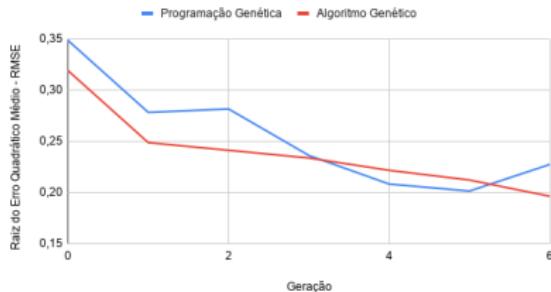
Resultados dos experimentos

Resultados dos experimentos

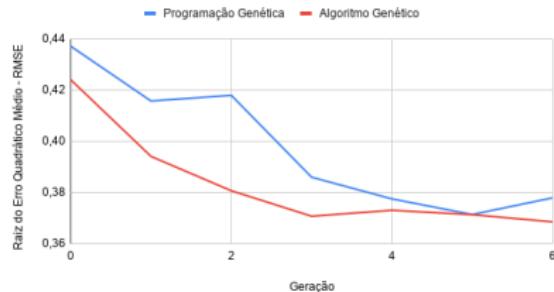


Resultados dos experimentos

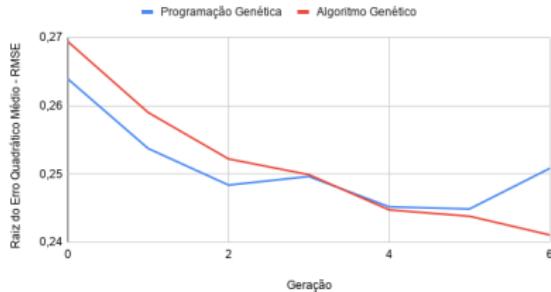
Validação no Banco de Dados MNIST



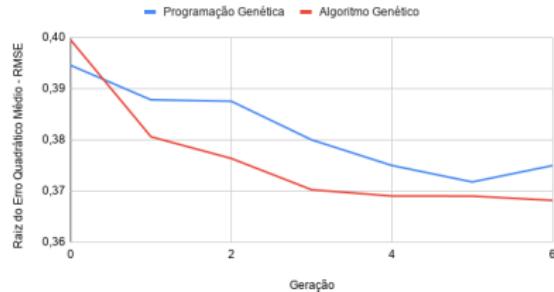
Validação no Banco de Dados CalTech 101 Silhouette



Validação no Banco de Dados MPEG-7 Shape

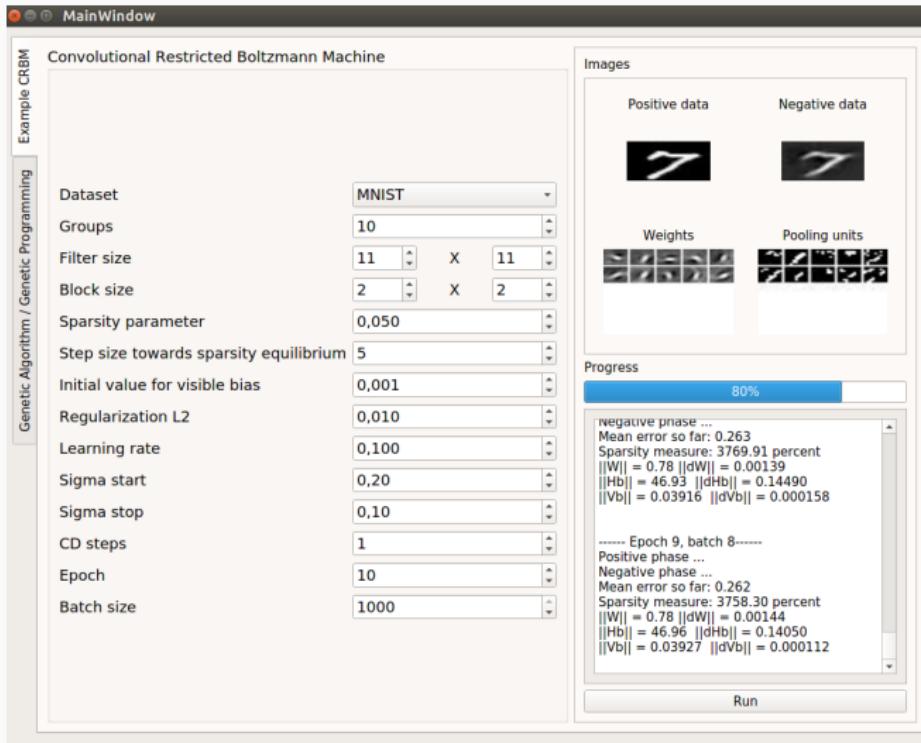


Validação no Banco de Dados Semeion



Software desenvolvido

Software Desenvolvido



Software Desenvolvido

MainWindow

Example CRBM

Genetic Programming

Parameters

Meta-Heuristic	Genetic Programming
Probability of crossover	0,40
Probability of mutation	0,10
Population	30
Number of generations	6

Hyperparameters - bound

	Min	Max
Groups	1	25
Filter size	5	15
CD steps	1	3

Fixed hyperparameters

Dataset	Semeion
Block size	2 X 2
Sparcity parameter	0,050
Step size towards sparsity equilibrium	5
Initial value for visible bias	0,001
Regularization L2	0,010
Learning rate	0,100
Sigma start	0,20
Sigma stop	0,10
Epoch	20
Batch size	100

Results

gen	avg	std
0.0	0.394249300...	0.019112708...
1.0	0.389542416...	0.019160829...
2.0	0.386246566...	0.019277685...
3.0	0.383765256...	0.015401799...
4.0	0.381688281...	0.013260446...
5.0	0.3803633521...	0.016235877...
6.0	0.378874208...	0.017937520...

Progress

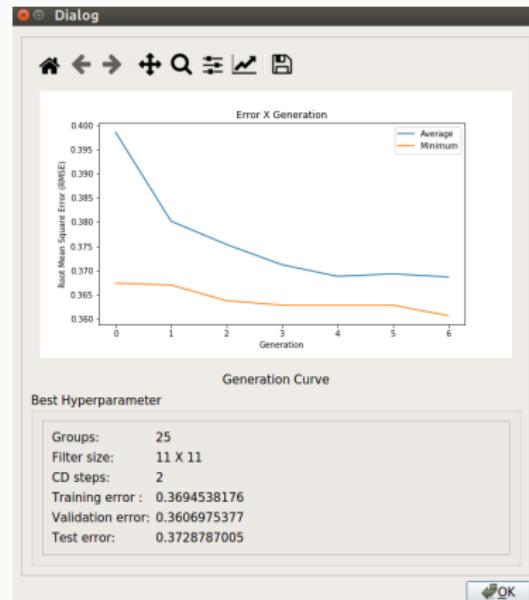
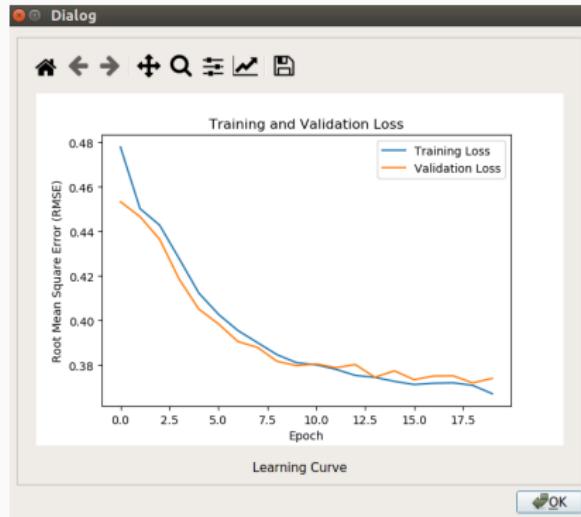
100%

Test error

```
----- Epoch 19, batch 0 -----  
Positive phase ...  
Negative phase ...  
Mean error so far: 0.371  
----- Epoch 19, batch 1 -----  
Positive phase ...  
Negative phase ...  
Mean error so far: 0.372  
----- Epoch 19, batch 2 -----  
Positive phase ...  
Negative phase ...  
Mean error so far: 0.372
```

Run

Software Desenvolvido



Conclusão

Conclusão

- Mostraram bom desempenho e resultados similares, porém requer um poder computacional.
- Algoritmo Genético se mostrou um pouco mais eficiente em achar o melhor indivíduo após as gerações enquanto a Programação Genética apresentou flutuações mais evidentes.
- Ao lidar com problemas mais complexos como otimizar todos os hiperparâmetros, um número maior de gerações pode ser necessário para gerar estruturas mais adequadas.
- O operador de cruzamento foi aplicado a uma taxa de 40% e observamos que uma operação com taxas acima dessa porcentagem, os valores flutuam bastante dificultando a convergência.

Trabalhos Futuros

- Outras métricas podem ser analisadas como acurácia e pseudo-verossimilhança;
- Novos modelos de aprendizado de máquina podem ser introduzidos para validar e eficiência dos métodos GP e GA.
- Também é possível introduzir o conceito de validação cruzada com k -fold para calcular o desempenho da rede.
- A introdução de uma nova função que avalia o desempenho da rede pode ser proposta para resolver problema de *overfitting* e *Underfitting*:

Função objetiva modificada

$$\min 100 \cdot (e_{train} - e_{valid})^2 + e_{valid}. \quad (2)$$

Referências

- [1] X. Cai, S. Hu, and X. Lin.

Feature extraction using restricted boltzmann machine for stock price prediction.

In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, volume 3, pages 80–83. IEEE, 2012.

- [2] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné.

DEAP: Evolutionary algorithms made easy.

Journal of Machine Learning Research, 13:2171–2175, jul 2012.

- [3] G. E. Hinton.

A practical guide to training restricted boltzmann machines.

In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

- [4] G. E. Hinton, S. Osindero, and Y.-W. Teh.
A fast learning algorithm for deep belief nets.
Neural computation, 18(7):1527–1554, 2006.
- [5] Y. Koren.
The bellkor solution to the netflix grand prize.
Netflix prize documentation, 81(2009):1–10, 2009.
- [6] J. R. Koza, D. Andre, M. A. Keane, and F. H. Bennett III.
Genetic programming III: Darwinian invention and problem solving, volume 3.
Morgan Kaufmann, 1999.
- [7] H. Larochelle and Y. Bengio.
Classification using discriminative restricted boltzmann machines.
In *Proceedings of the 25th international conference on Machine learning*, pages 536–543, 2008.

- [8] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back.
Face recognition: A convolutional neural-network approach.
IEEE transactions on neural networks, 8(1):98–113, 1997.
- [9] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng.
Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.
In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616, 2009.
- [10] A. K. Noulas, B. J. Kröse, et al.
Deep belief networks for dimensionality reduction.
In *Belgian-Netherlands Conference on Artificial Intelligence*, pages 185–191. Citeseer, 2008.
- [11] D. Whitley.
A genetic algorithm tutorial.
Statistics and computing, 4(2):65–85, 1994.

Perguntas?

Obrigado pela atenção!