MO431 - Tarefa 4

Jônatas Trabuco Belotti
RA: 230260
jonatas.t.belotti@hotmail.com

I. Introdução

O objetivo desse trabalho é realizar a otimização dos hiperparâmetros de uma Máquina de Vetores de Suporte (SVM, do inglês *Support Vector Machine*) que será aplicada em uma regressão de um banco de dados. A base de dados utilizada na regressão possui um total de 506 registros divididos em dois conjuntos, um conjunto de treinamento com 404 amostras e um conjunto de testes com 102.

Todos os códigos deste trabalho foram escritos utilizando a linguagem de programação **Python 3.6.9** com as seguintes bibliotecas: **Numpy 1.18.2**, **Scikit-Learn 0.22.2**, **Hyperopt 0.2.4**, **Pyswarms 1.1.0**, **Simanneal 0.5.0** e **cma 3.0.3**. Sendo estas obrigatórias para execução dos códigos. Todo o código desenvolvido juntamente com os arquivos utilizados para o treinamento e teste da SVM podem ser acessados no repositório do GitHub `https://github.com/jonatastbelotti/mo431_trabalho4`.

II. Estudo de caso

Foi utilizada a implementação do SVM do pacote **Scikit-Learn** através da classe **SVR**, que é uma implementação de SVM para regressão. Dentre todos os hiperparâmetros desse modelo foram otimizados apenas os 3 considerados mais importantes:

- **C** - no intervalo de busca entre 2^{-5} e 2^{15}, com distribuição uniforme nos expoentes;
- **gamma** - intervalo de busca entre 2^{-15} e 2^{3}, com distribuição uniforme nos expoentes;
- **epsilon** - com intervalo de busca entre $0,05$ e 1.0, com distribuição de probabilidade uniforme.

Como medida de erro utilizada para avaliar o desempenho das regressões realizadas pela SVM foi utilizado Erro Absoluto Médio (MAE, do inglês Mean Absolute Error) através da função **mean_absolute_error** do pacote **Scikit-Learn**.

A busca pelos melhores hiperparâmetros para o SVM de regressão foi realizada com os seguintes algoritmos de otimização:

- **RS -** *Random Search* com 125 combinações dos hiperparâmetros. Como implementação foi utilizada a classe **RandomizedSearchCV** do pacote **Scikit-Learn**.
- **GS -** *Grid Seach* com um grid de busca de $5 \times 5 \times 5$. Aqui também foi utilizada uma implementação do pacote **Scikit-Learn** através da classe **GridSearchCV**.
- **OB -** Otimização Bayesiana com 125 chamadas para a função. Utilizou-se a implementação do pacote **Hyperopt** que usa um regressor (TPE) para modelar a distribuição de probabilidades que é muito mais rápido que a implementação padrão.
- **PSO -** Otimização por Enxame de Partículas com apenas 11 partículas e 11 iterações. Como implementação foi utilizada a função **pso** do pacote **Pyswarm**.
- **SA -** *Simulated Annealing* com 125 passos. Foi utilizada a classe **Annealer** do pacote **Simanneal**.
- **CMA-ES -** Estratégia de Evolução da Adaptação da Matriz de Covariância com 125 chamadas da função. Aqui utilizou-se a implementação da classe **CMAEvolutionStrategy** do pacote **CMA**.

Todos os algoritmos realizaram a otimização dos hiperparâmetros levando em consideração apenas o MAE para o conjunto de treinamento. Uma vez que a otimização foi finalizada e os valores dos hiperparâmetros foram encontrados a avaliação final do desempenho é realizada através do MAE para o conjunto de teste.

Os melhores valores para os 3 hiperparâmetros encontrados por cada algoritmo de otimização de otimização juntamente com o MAE da SVM para o conjunto de treinamento e de teste utilizando esses hiperparâmetros são apresentados na Tabela I.

Observando os dados da Tabela I logo nota-se a grande variedade dos valores para os hiperparâmetros. Nenhum valor de **C**, **gamma** ou **epsilon** foi encontrado por mais de um algoritmo.

Ainda observando os dados da Tabela I é possível verificar que a configuração que resultou no menor MAE para o conjunto de treinamento não é a mesma com o menor MAE para o conjunto de teste. Uma explicação para isso é que quando a SVM tem erros extremamente baixos para o conjunto de treinamento, ela pode estar sobretreinada, ou seja, em vez de aprender os padrões que moldam os dados do conjunto de treinamento ela simplesmente decorou os dados. Desse modo, quando a mesma SVM que apresentou resultados tão bons no treinamento é submetida ao conjunto de teste, conjunto esse que ela ainda não teve contato seu desempenho é muito inferior, visto que como a SVM decorou os dados de treinamento ela não tem a capacidade de generalizar seu aprendizado para dados novos. Um exemplo claro desse comportamento pode ser observado pelos resultados do CMA-ES, que obteve o melhor MAE para o conjunto de treinamento, mas o pior para o de teste.

Como o objetivo desse trabalho é analisar os desempenhos dos algoritmos de otimização na busca pelos melhores hiperparâmetros e a falha mencionada acima é referente ao modelo de regressão e não dos algoritmos de otimização será

TABELA I: Resultados da execução de cada algoritmo.

Algoritmo	C	gamma	epsilon	MAE (Treinamento)	MAE (Teste)
RS	1490.233819	0.000108	0.789122	1.698516	2.834550
GS	30.800390	0.001193	0.716517	1.685725	3.565599
OB	396.923914	0.231660	0.050924	0.050550	5.642941
PSO	173.922902	0.210269	0.050000	0.049485	5.614816
SA	1061.366533	0.338134	0.099685	0.098761	5.716623
CMA-ES	833.666135	0.525374	0.000004	0.000068	5.744949

levada em consideração apenas o MAE de treinamento como medida de desempenho para os algoritmos de otimização testados.

Analisando com esse quesito verifica-se que o algoritmo que encontrou os melhores valores para os hiperparâmetros foi o CMA-ES, com um MAE para o conjunto de treinamento de $0,000068$; seguido pelo PSO e OB com MAEs de $0,049485$ e 0.050550 respectivamente.

Por sua vez os piores resultados ficaram por conta dos algoritmos RS e GS com MAEs de $1,698516$ e $1,685725$ respectivamente.