Entregável 2.1. Resultados de execução de PyCCD numa máquina da DGT com acesso aos dados Sentinel-2 Theia. Estimativas de necessidades computacionais para processamento para o território de Portugal Continental com a frequência temporal de 2 meses.

Índice

Índice das Tabelas

1 Aspetos Computacionais

Ao aplicar o código conforme a lógica presente no repositório do GitHub (https://code.usgs.gov/lcmap/pyccd), verificou-se que o tempo de processamento de 10.000 pixels foi de aproximadamente 3 horas, reforçando que estes testes foram para o período de março de 2017 a dezembro de 2021. Para tentar reduzir o tempo de processamento, implementou-se algumas melhorias, incluindo a leitura dos dados utilizando a biblioteca Python xarray, distribuição do processamento de pontos em paralelo pelos vários processadores da máquina e a exclusão de bandas não utilizadas na deteção – como o RED, NIR e SWIR1. Depois destas melhorias, conseguiu-se uma redução significativa no tempo de processamento dos dados Theia/MAJA. Os resultados são reportados para "*batchs*" de 10000 pixels Sentinel-2 com resolução de 10 m, que correspondem no conjunto a uma área de 1 km^2. Abaixo, a tabela apresenta os tempos de leitura de dados e de computação e o desempenho dos métodos de deteção de alterações para alguns valores dos parâmetros:

'ALPHA'	'LASSO_MAX_ITER'	Tempo de processamento CCD	Leitura de dados / Escolha de pontos	Desempenho
0	1000	15.39 mins	~ 10 mins	F1-score = 80.24% Omission error = 16.59% Commission error = 22.69%
20	1000	17.55 mins		F1-score = 81.82% Omission error = 16.22% Commission error = 20.05%
0	25000	16.06 mins		F1-score = 80.24% Omission error = 16.59% Commission error = 22.69%
20	25000	18.18 mins		F1-score = 81.82% Omission error = 16.22% Commission error = 20.05%

Tabela 1 – Tempos de processamento/desempenho do CCD para dados Theia numa máquina equipada com processador i7-8700 3.20GHz 6 Core(s) e 64 GB de RAM.

Observou-se que ao reduzir o parâmetro **'LASSO_MAX_ITER'** de 25.000 para 1.000, houve uma diminuição leve no tempo de computação, mantendo o desempenho. Por outro lado, ao alterar o valor de **'ALPHA'** de 20 para 0, isto é, ao substituir a função Lasso por uma regressão linear, houve uma diminuição de aproximadamente ~ 2 minutos no tempo de computação, porém o F1-Score e o erro de comissão pioraram.

Após uma análise detalhada dos resultados, foi essencial compreender quais as partes do algoritmo que estavam a consumir mais tempo de computação. Ao realizar um perfil do código, constatou-se que a função Lasso representa aproximadamente 77% do tempo total de processamento:

Ficheiro	% do tempo de execução	Tempo de execução (Total = 33m:52.568s)
pyccd_theia/ccd/models/lasso.py	77.39	25m:26.544s

Tabela 2 – Perfil do código detalhando o ficheiro que requer mais tempo de processamento.

Com base na premissa de selecionar os pixels apropriados para a execução do CCD, constatou-se que a área de Portugal Continental, onde há simultaneamente floresta ou

mato de acordo com a COS2018 e a COSc 2023, é composta por 352 milhões e 116 mil pixels. Destes pixels são excluídos aqueles com NDVI acima de 0,7 ou cujo NDVI tenha aumentado ou permanecido igual em relação ao mês anterior. Esta regra foi aplicada aos compósitos mensais da DGT (https://smos.dgterritorio.gov.pt/sites/default/files/documents/Ficha_de_produto_MIAEV_v_20-12-2022.pdf). A Tabela 3 apresenta o tempo de computação necessário para processar a totalidade do território português que satisfaz esses critérios, usando uma máquina com velocidade relativamente baixa de 3,2 GHz. Os tempos estimados foram calculados com base no valor de referência de 15,39 minutos para 10.000 pixels.

	Número de pixels	**Tempo de computação PyCCD**
Outubro 2023	56.409.000	58 dias
Novembro 2023	45.207.000	47 dias
Dezembro 2023	54.558.000	56 dias
Janeiro 2024	81.819.000	85 dias

Tabela 3 – Tempos de execução estimados de PyCCD para o número de pixels existentes para cada mês para uma máquina com 6 cores e velocidade de relógio de 3.2 GHz.

2 Estratégias a explorar para reduzir o tempo de computação

2.1 Melhorar a eficiência do algoritmo

O objetivo futuro será adaptar o código de PyCCD de modo a reduzir significativamente o tempo de computação.

Uma das pistas a explorar que foi identificada no trabalho realizado até ao momento está relacionado com a decomposição do tempo de processamento e da identificação da rotina Lasso como sendo particularmente pesada computacionalmente. Uma solução poderá ser a substituição da regressão Lasso da biblioteca Python scikit-learn pela implementação disponível nas bibliotecas Python cuML e statsmodels.

A outra direção de pesquisa está relacionada com a paralelização de partes do código de PyCCD. Após a identificação das partes mais pesadas do processo (leitura de dados, regularização Lasso), irão ser testadas alterações do código para aumentar a eficiência dessas rotinas.

2.2 Reaproveitar os segmentos já calculados

Para reduzir o tempo de computação, também foi considerada a possibilidade de executar o algoritmo apenas a partir da última data de quebra, em vez de iniciar o processo desde o início sempre que novas imagens de satélite estejam disponíveis. De facto, essa é a estratégia seguida pelo INPE (Karine Ferreira, INPE, comunicação pessoal) e pelo GEE - *Google Global Landsat-based CCDC Segments*.

Essa estratégia é ilustrada na Figura 1, em que se mostra um exemplo de ajustamento da série temporal de NDVI, com conjuntos de dados sucessivos. A modelação do primeiro

segmento não tem de ser refeita após a aquisição de novas imagens, dado que a modelação desse segmento não se altera.

No entanto, mais investigação é necessária para confirmar que este comportamento é preservado sob a aplicação do código PyCCD.

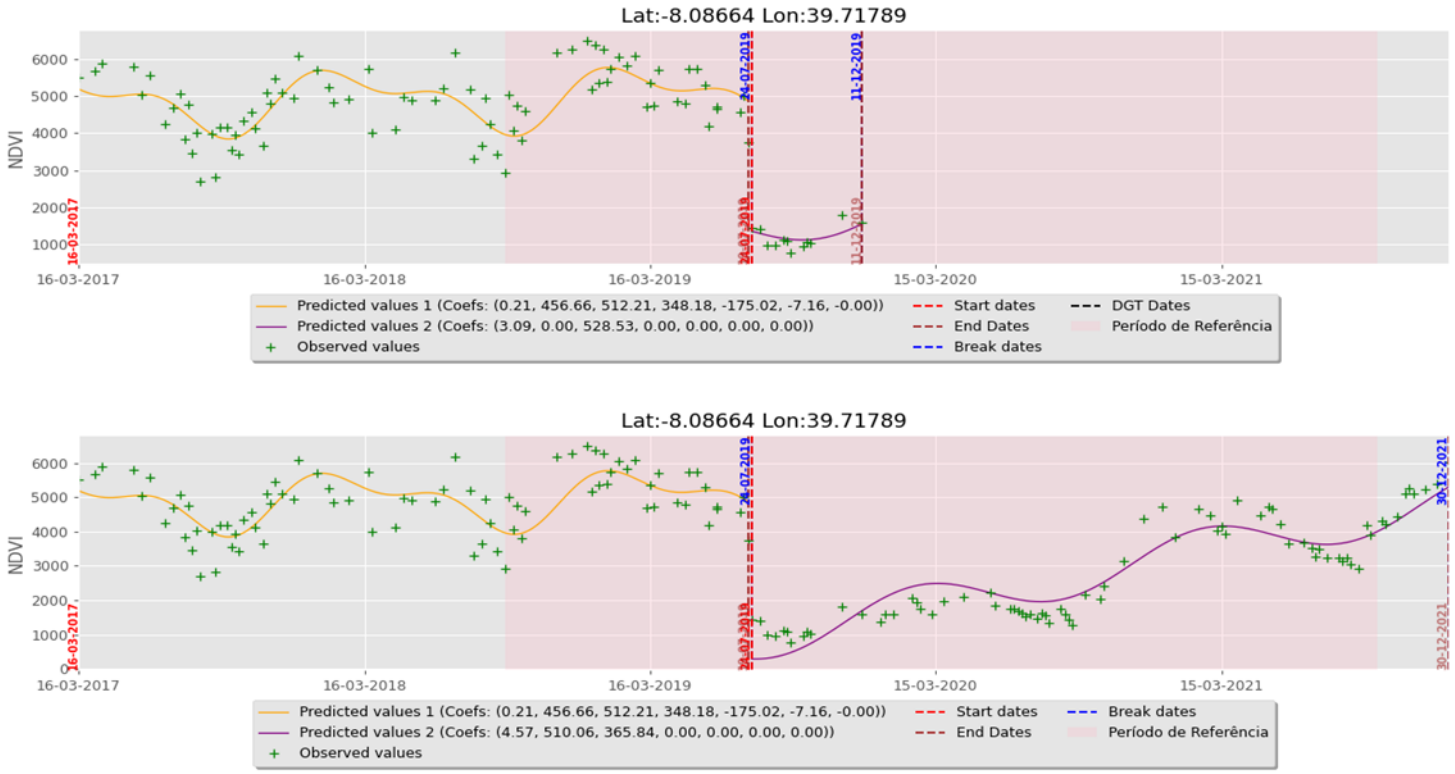

Figura 1 – Painel superior: CCD para o período entre março de 2017 e dezembro de 2019; painel inferior: o mesmo que em cima, mas para o período entre março de 2017 e dezembro de 2021.

3 Referências

1. Compósitos mensais da DGT. Disponível em: https://snig.dgterritorio.gov.pt/rndg/srv/eng/catalog.search#/search?anysnig=MIAEV&fast=index (Acedido a: 17 de junho 2024).
2. Google Global Landsat-based CCDC Segments (1999-2019). Disponível em: https://developers.google.com/earth-engine/datasets/catalog/GOOGLE_GLOBAL_CCDC_V1 (Acedido a: 17 de junho 2024).
3. INPE. Disponível em: https://www.gov.br/inpe/pt-br (Acedido a: 17 de junho 2024).