Entregável 1.2 . Seleção e justificação das metodologias a operacionalizar

Índice

Índice das Figuras

Índice das Tabelas

1 Introdução

O desenvolvimento de metodologias eficazes à escala nacional e eficientes a nível computacional para a criação sistemática de um produto nacional em formato vetorial de delimitação de manchas superiores a 0.5 ha de perda recente de floresta e mato com base em análise automática de imagens de satélite requere ferramentas que permitem explorar a informação temporal e espacial contida nos dados de muito grande dimensão que consistem em séries de imagens Sentinel-2 para Portugal Continental.

Como discutido no Entregável 1, as técnicas que parecem ser mais adaptadas ao problema incluem algoritmos de deteção de alterações ao nível do pixel, entre os quais se destaca o designado *Continuous Change Detection and Classification* (CCDC) que não só permite identificar perdas de vegetação (tarefa 3) como caracteriza o padrão temporal da ocupação do solo com o objetivo de caracterizar o fator de alteração (tarefa 4).

Neste relatório, descreve-se as adaptações realizadas e os resultados de experiências computacionais realizadas com o CCDC sobre uma base de dados de referência para:

1. Adaptação do algoritmo PyCCD (aberto) a dados Sentinel-2;
2. Parametrização do método;
3. Avaliação do desempenho;
4. Avaliação de recursos computacionais necessários.

Os resultados do presente relatório permitem:

1. Aplicar CCDC para deteção automática de alterações a uma nova coleção de dados de referência (dados Navigator que será objeto do Entregável 2.1), e posterior validação;
2. Definir direções de pesquisa no quadro do projeto para permitir reduzir o tempo de execução do método com o objetivo de o aplicar à generalidade do território continental português, em áreas onde se prevê à partida a possibilidade de ocorrência de perdas de vegetação e mato.

Recorde-se que CCDC é uma técnica de deteção e classificação de mudanças contínuas que tem sido amplamente utilizada em monitorização ambiental, especialmente na deteção de mudanças na cobertura terrestre ao longo do tempo (Zhu et al., 2014). Esta técnica é particularmente útil em aplicações de deteção remota, onde é necessário identificar e quantificar mudanças em grandes áreas de forma contínua e precisa (Awty-Carroll et al., 2019).

O CCDC é projetado para lidar com séries temporais de imagens de deteção remota, como imagens de satélite, que são adquiridas em intervalos regulares ao longo do tempo. A ideia fundamental do CCDC é modelar as séries temporais de dados de deteção remota de forma a capturar tanto a variabilidade sazonal quanto as mudanças de longo prazo na cobertura terrestre. Isso é feito através da combinação de técnicas de modelos de estatística, como séries temporais de decomposição e modelos de regressão (Xian et al., 2022, Zhu et al., 2014).

Os scripts em Python, objeto deste relatório, estão disponível em: https://github.com/manuelcampagnolo/S2CHANGE/tree/main/scripts/pyccd_theia/notebooks.

2 Dados

Os testes realizados e descritos neste relatório usaram imagens Sentinel-2 fornecidos pela Theia, uma plataforma francesa que desempenha um papel importante na distribuição e processamento de dados de deteção remota. Os dados disponibilizados pela Theia são pré-processados e corrigidos usando o algoritmo de correção atmosférica MAJA, proporcionando uma base sólida para análises subsequentes. Além disso, foram realizados testes com dados do Sentinel-2 provenientes do *Google Earth Engine* (GEE), os quais foram processados utilizando o algoritmo de correção atmosférica s2cloudless (Skakun et al., 2022).

Ambos os conjuntos de dados foram criteriosamente selecionados, descartando imagens com cobertura de nuvens superior a 60%, visando garantir a qualidade e a consistência dos resultados obtidos.

Na Figura 1, apresenta-se a série temporal de disponibilidade de imagens para os dados Theia/MAJA (representados por círculos vermelhos) e os dados GEE/s2cloudless (representados por círculos azuis) entre 2017 e o final de 2021.

Observa-se que no início da série, os dados Theia/MAJA são mais consistentes em comparação com os GEE/s2cloudless. No entanto, para datas mais recentes os dados GEE/s2cloudless têm maior densidade relativamente aos dados Theia/MAJA, o que pode ser explicado por serem ambas máscaras conservativas, mas MAJA usar um buffer de maior dimensão para nuvens (240 m em vez de 160 m) e incluir deteção de sombras (Skakun et al, 2022).

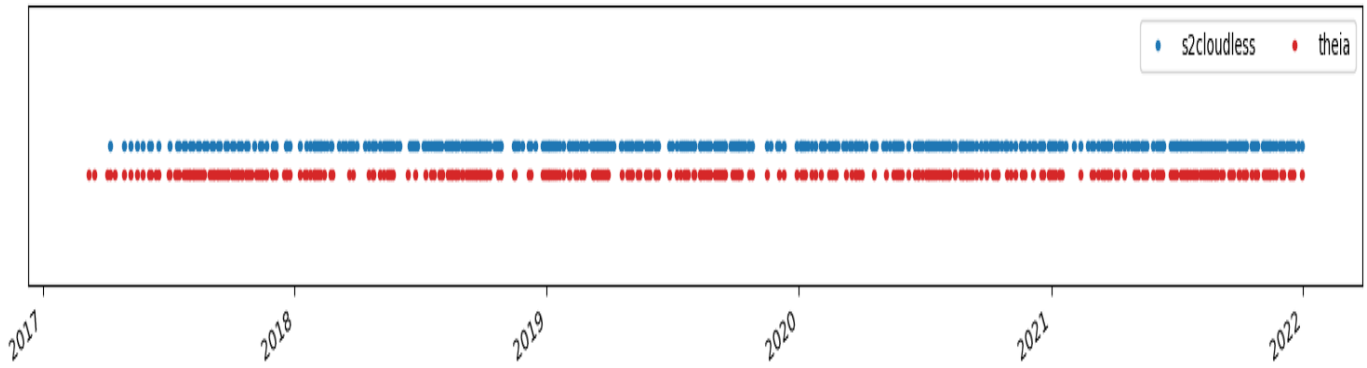

Figura 1– Disponibilidade de dados Theia/MAJA e GEE/s2cloudless.

Estes testes foram realizados sobre a base de dados de referência BDR-300 (Moraes et al, ND) extraída sobre o tile T29TNE Sentinel-2 para o período compreendido entre março de 2018 e dezembro de 2021. Esta região de Portugal Continental é caracterizada pela presença de densas florestas e vastas áreas agrícolas, representando um ambiente diversificado. Além disso, esta área foi severamente afetada por incêndios florestais de grande magnitude em 2017, resultando numa paisagem em constante mudança. Assim, dada a complexidade e a dinâmica desse cenário, o PyCCD demonstra um forte potencial para detetar e monitorizar mudanças nesta região.

Para a maiorias das experiências realizadas e descritas neste relatório, foram escolhidos aleatoriamente 10000 pontos entre os aproximadamente 350000 pontos

existentes em BDR-300. A Figura 2 mostra a distribuição espacial de uma mostra de 10000 pontos extraídos da BDR-300, sendo que o número dentro de cada hexágono representa a quantidade de pontos escolhidos dentro do respetivo buffer. A distribuição dos pontos foi feita de maneira a assegurar que todos os buffers tivessem pelo menos um ponto para serem incorporados nos dados de entrada do algoritmo de deteção de alterações (Moraes et al, ND).

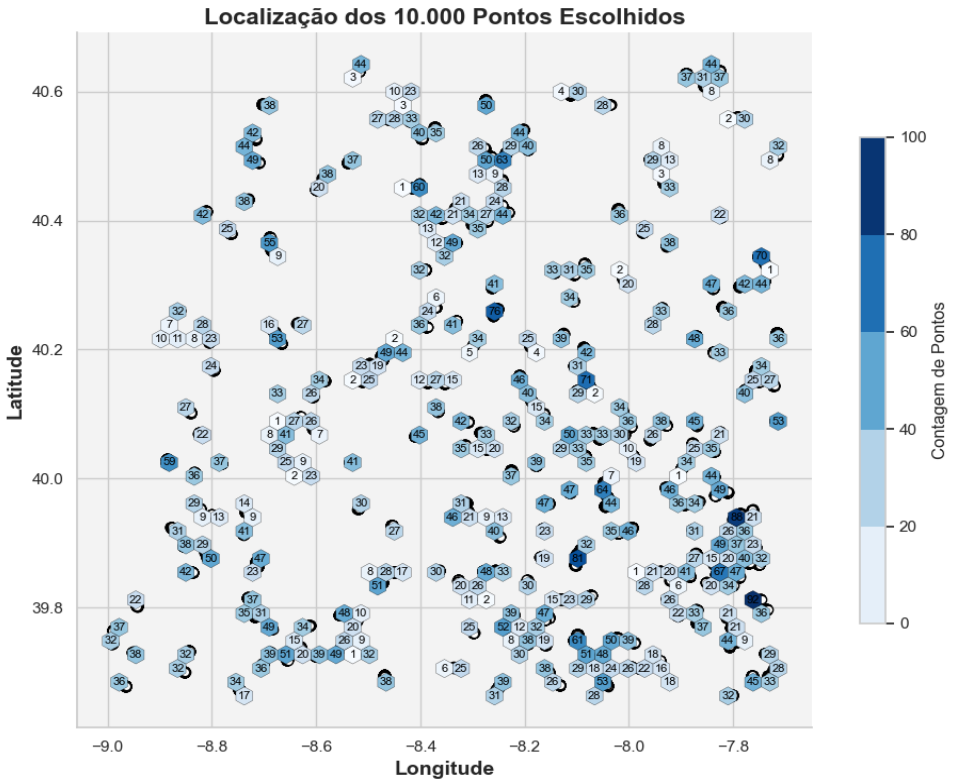

Figura 2 – Localização de uma amostra de 10 .000 pontos para input do PyCDD.

3 CCD

3.1 Adaptação do algoritmo PyCCD

Para implementar eficientemente o CCDC, foi feita uma adaptação de um código Python disponível em https://code.usgs.gov/lcmap/pyccd, originalmente projetado para dados do satélite LANDSAT, mas que teve que ser adaptado, no quadro deste projeto, para suportar dados do satélite Sentinel-2. Esse algoritmo é designado por PyCCD.

O código do PyCCD aberto é muito complexo e não está documentado em detalhe, sendo a interpretação do código e a sua melhoria uma tarefa tecnicamente difícil. No entanto, entende-se que esse investimento é necessário para poder dispor para a prossecução do projeto de um código que possa correr em diversos ambientes computacionais, desde workstations locais a clusters de computação avançada como, por exemplo, os clusters da Rede Nacional de Computação Avançada (RNCA). A escolha de um código na linguagem Python facilita a portabilidade do código entre plataformas.

Durante o processo de adaptação do código, ajustamos o modelo para incorporar as diferentes características dos dois satélites. Por exemplo, o LANDSAT possui uma banda térmica, ao passo que o Sentinel-2 não. Além disso, os dados do Sentinel-2

utilizados no algoritmo já incluíam correções atmosféricas, diferentemente dos dados do LANDSAT, onde esse processamento era feito dentro do próprio algoritmo.

Adicionalmente, foram adicionados alguns parâmetros ao modelo que anteriormente não estavam incorporados, como **'CHISQUAREPROB'** e **'ALPHA'** (este parâmetro determina a intensidade da regularização aplicada ao modelo).

3.2 Parâmetros do PyCCD

Os parâmetros principais escolhidos para o modelo PyCCD foram os seguintes:

PEEK_SIZE	MIN_YEARS	DETECTION_BANDS	LASSO_MAX_ITER	CHISQUAREPROB	ALPHA
6	1	NDVI, Green, SWIR2	1000, 25000	0.999	0 a 200

Tabela 1 – Parâmetros escolhidos para o CCD.

Os valores dos parâmetros foram selecionados com base nos resultados de (Moraes et al., ND) mas adaptados para os novos dados de entrada, em particular as imagens Theia. Nesse estudo, os autores testaram o CCD implementado em GEE com os mesmos valores para os parâmetros mencionados anteriormente, atingindo um bom desempenho do modelo.

Foram aqui considerados 3 combinações de dados e de algoritmos: imagens Theia/MAJA com PyCCD, imagens GEE/s2cloudness com algoritmo PyCCD, e imagens GEE/s2cloudness com algoritmo eCCD, o algoritmo que é executado em GEE e cujo código não é aberto. O objetivo foi verificar que as soluções locais (usando PyCCD) tinham o mesmo nível de desempenho do que o produto de alterações obtido em ambiente GEE, com o algoritmo eCCD.

A Figura 3 ilustra o resultado obtido quando o CCD é implementado num determinado pixel. Inicialmente, os dados revelam uma série temporal consistente de valores de NDVI, variando entre 7500 e 8500, indicando uma cobertura vegetal saudável e densa, predominantemente composta por pinheiro-bravo. Entretanto, a 28 de julho de 2020 (designada por data de quebra) observa-se uma mudança abrupta. Os valores de NDVI sofrem uma queda acentuada, atingindo a faixa de 1000. Essa redução significativa e repentina nos valores do NDVI é um claro indicador de uma situação adversa, especificamente um evento de fogo.

Figura 3 – Resultado obtido pelo CCD para um pixel onde houve uma ocorrência de fogo na data identificada.

3.3 Escolha do valor do parâmetro de regularização *alpha*

Para melhorar o desempenho do modelo, foi feito uma série de testes para determinar o valor ideal do parâmetro *alpha*, que desempenha um papel fundamental na função do Lasso da biblioteca Python scikit-learn, determinando a intensidade da regularização aplicada ao modelo. Quando este parâmetro é definido como zero, a regressão Lasso é desativada, resultando numa regressão linear.

Esta pesquisa foi realizada em ambos os conjuntos de dados, Theia/MAJA (apresentado nos gráficos superiores da Figura 4) e GEE/s2cloudless (exibido nos gráficos inferiores da Figura 4), explorando uma ampla faixa de valores de *alpha*, variando de 0 a 200.

Ao investigar os diferentes conjuntos de dados, observámos que o valor ideal de *alpha* pode variar para cada um dos conjuntos de dados, por várias razões (cf. documentação da função sklearn.linear_model.lasso):

I. Tamanho do conjunto de dados**:** Conjuntos de dados de diferentes tamanhos podem exigir diferentes níveis de regularização para se ajustar aos padrões específicos de cada conjunto.
II. Distribuição dos dados: Variações na distribuição dos dados entre os conjuntos podem afetar o comportamento do modelo, levando a necessidades distintas de regularização.
III. Complexidade dos dados: Conjuntos de dados mais complexos, ou com maior presença de ruído, podem requerer valores de alpha diferentes para controlar adequadamente o ajuste do modelo.

Além disso, a escolha do valor de *alpha* influencia o equilíbrio entre sobreajustamento e subajustamento do modelo. Valores mais altos de *alpha* tendem a reduzir o sobreajustamento, resultando em modelos mais simples com menos coeficientes significativos. Por outro lado, valores mais baixos de *alpha* oferecem maior flexibilidade ao modelo, mas podem aumentar o risco de sobreajustamento, levando a modelos mais complexos com mais coeficientes significativos.

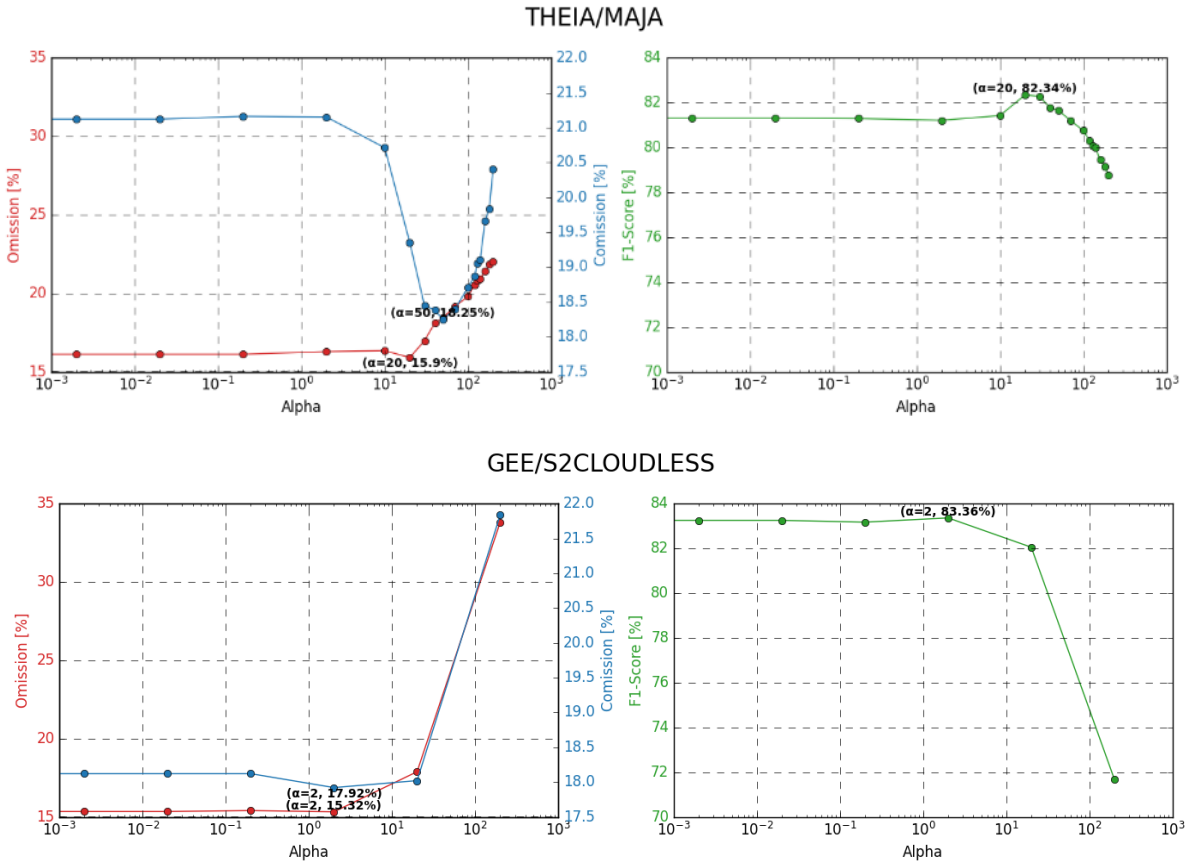

Figura 4 – Desempenho do modelo com diferentes valores de alpha para dados Theia/MAJA (gráficos superiores) e GEE/s2cloudless (gráficos inferiores).

Ao analisar os dados Theia/MAJA, o valor ideal de *alpha* para maximizar o F1-Score e minimizar a Omissão é de 20, resultando em taxas de 82.34% e 15.9%, respetivamente. Embora o valor que minimiza o erro de comissão seja 50, a escolha recaiu sobre 20 devido à mínima discrepância observada nessa métrica entre os valores de 20 e 50, resultando numa taxa de 18.38% (em vez dos 18.25% que seriam obtidos com *alpha* = 50).

Em relação aos dados GEE/s2cloudless, o valor ideal de *alpha* para otimizar tanto o F1-Score quanto a Omissão e a Comissão é 2, produzindo taxas de 83.36%, 15.32% e 17.92%, respetivamente.

4 Desempenho do CCD sobre dados de referência

Após o modelo ter processado uma amostra de pixels, é realizado um teste de validação utilizando os resultados fornecidos pelos analistas. Esse teste de validação abrange o período de análise compreendido entre 12/09/2017 e 30/09/2021, no qual cada deteção automática feita pelo PyCCD foi classificada como verdadeiro positivo (VP), verdadeiro negativo (VN), falso positivo (FP) ou falso negativo (FN) – Figura 5.

- **Verdadeiro Positivo (VP):** Refere-se a uma deteção automática pelo PyCCD que coincide exatamente com a quebra identificada pelo analista.
- **Verdadeiro Negativo (VN):** Pode ocorrer de duas maneiras:

1. A deteção pelo PyCCD ocorre fora do período de análise, portanto, durante este período, não há deteções identificadas nem pelo PyCCD nem pelos analistas.
2. O PyCCD não identifica nenhuma deteção ao longo da série temporal.

- **Falso Positivo (FP):** Ocorre quando o PyCCD deteta algo automaticamente, mas o analista não identifica nenhuma quebra.
- **Falso Negativo (FN):** Pode surgir de duas maneiras:
  1. O PyCCD não identifica nenhuma deteção automática ao longo de toda a série, enquanto o analista identifica.
  2. O PyCCD identifica antes do período de análise dos analistas.

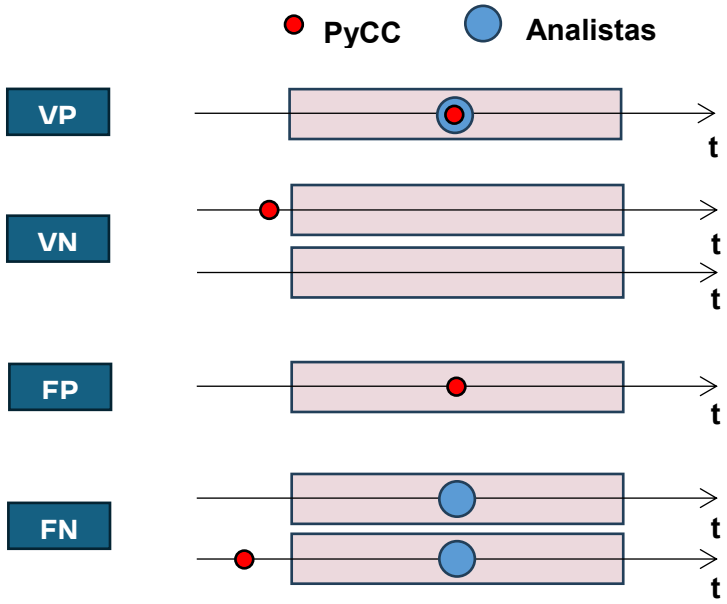

Figura 5 – Classificação de deteções automáticas pelo PyCCD em relação às identificações dos analistas. VP (Verdadeiro Positivo), VN (Verdadeiro Negativo), FP (Falso Positivo), FN (Falso Negativo). O retângulo rosa representa a tolerância na incerteza temporal (30 dias) – cf. (Moraes et al., ND).

Relacionando estas definições com os erros de omissão e comissão calculados para os dados Theia/MAJA e GEE/s2cloudless, podemos entender melhor a precisão das deteções efetuadas pelo CCD em relação às identificações realizadas pelos analistas (Tabela 2).

	THEIA/MAJA	GEE/S2CLOUDLESS
Erro de omissão [%]	15.9	15.32
Erro de comissão [%]	18.38	17.92

Tabela 2 – Erros de omissão e comissão para dados Theia/MAJA e GEE/s2cloudless.

Os resultados dos erros de omissão para os dados Theia e GEE revelam que o algoritmo PyCCD enfrenta dificuldades na deteção precisa de mudanças em ambas as bases de dados, perdendo cerca de 15.9% e 15.32% das mudanças reais identificadas pelos analistas, respetivamente. Da mesma forma, os erros de comissão indicam que aproximadamente 18.38% das mudanças identificadas pelo PyCCD nos dados Theia/MAJA e cerca de 17.92% nos dados GEE/s2cloudless não correspondem a mudanças reais observadas pelos analistas. Estes resultados terão que ser comparados com os resultados sobre a nova base de dados de referência a ser explorada no seguimento do projeto.

Refira-se que a comparação dos resultados GEE/s2cloudness e GEE/eCCD (com o algoritmo de código não aberto do GEE) deram resultados muito semelhantes, o que

é uma indicação de o que o desempenho de PyCCD é comparável ao que se obtém usando o GEE, tendo a vantagem de poder ser executado em plataformas locais.

5 Aspetos computacionais

Ao aplicar o código conforme a lógica presente no repositório do GitHub (https://code.usgs.gov/lcmap/pyccd), verificámos que o tempo de processamento de 10.000 pontos 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, implementámos 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, conseguimos uma redução significativa no tempo de processamento dos dados Theia/MAJA. Os resultados são reportados para "batches" de 10000 pixels Sentinel-2 com resolução de 10 m, que correspondem no conjunto a uma área de 1 km². 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 3 – 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.

Observámos 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 como mencionado anteriormente, 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 4 – 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, excluindo aqueles com NDVI acima de 0.7 ou que não tenham baixado de valor relativamente ao mês anterior, a Tabela 5 apresenta o tempo de computação que seria necessário para processar numa máquina semelhante (que tem uma velocidade relativamente baixa de 3.2 GHz) a totalidade do território português que satisfaz esses critérios. Os tempos de execução estimados tempos foram calculados usando o valor de referência de 15.39 mins para 10000 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 5 – 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.

6 Estratégias a explorar para reduzir o tempo de computação

6.1 Melhorar a eficiência do algoritmo

O nosso objetivo futuro será adaptar o código de PyCCD de modo 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.

6.2 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).

Essa estratégia é ilustrada na Figura 6, 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 que 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.

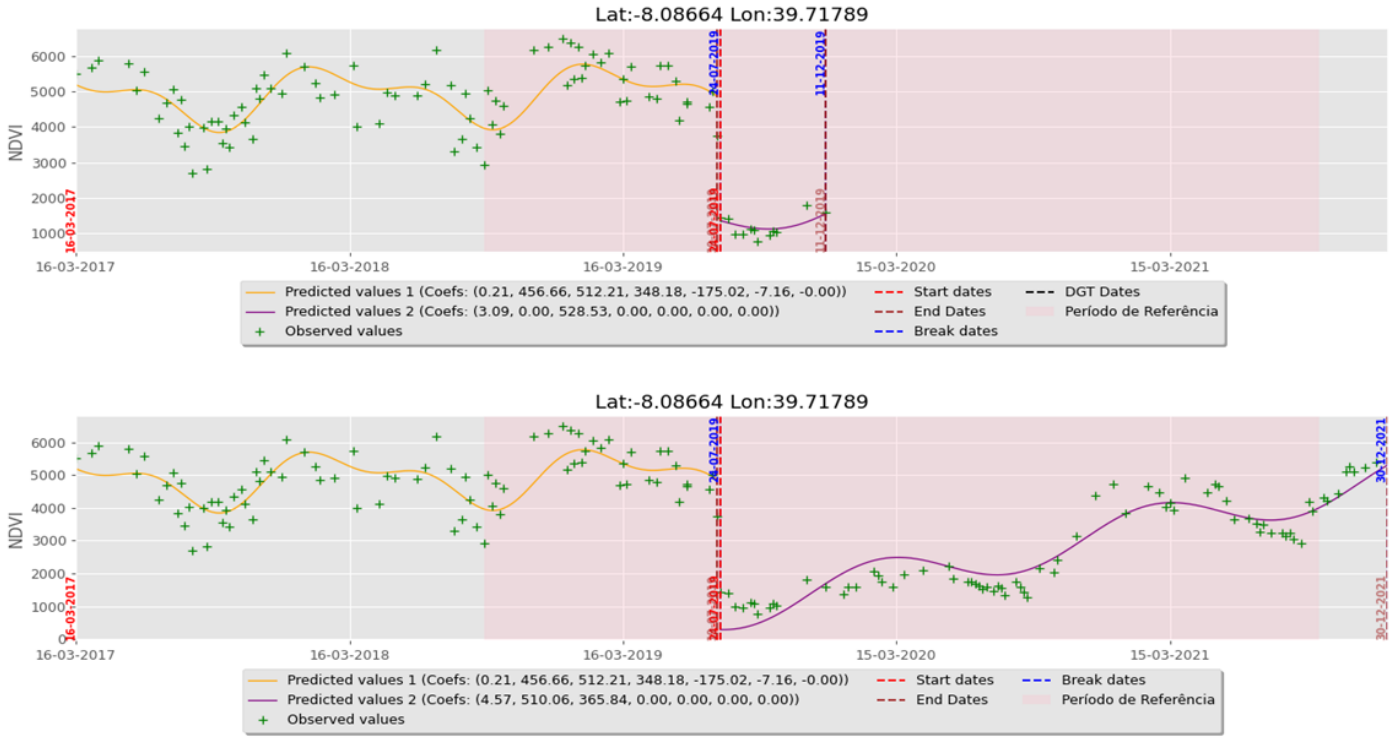

Figura 6 – Painel superior: CCD para o período entre março de 2017 e dezembro de 2019; painel inferior: o mesmo que em cima, mas para o período entre março de 2017 e dezembro de 2021

7 Referências

1. Awty-Carroll, Katie; Bunting, Pete; Hardy, Andy; Bell, Gemma (2019). Using Continuous Change Detection and Classification of Landsat Data to Investigate Long-Term Mangrove Dynamics in the Sundarbans Region. Remote Sensing, 11(23), 2833–. doi:10.3390/rs11232833
2. Código PyCCD para dados LANDSAT: https://code.usgs.gov/lcmap/pyccd (Acedido janeiro 2024)
3. Moraes, D. et al., "Continuous vegetation loss monitoring in a dynamic landscape of Central Portugal with Sentinel-2 data", (ND) em processo de revisão para o International Journal of Applied Earth Observation and Geoinformation
4. Skakun, Sergii & Wevers, Jan & Brockmann, Carsten & Doxani, Georgia & Aleksandrov, Matej & Batic, Matej & Frantz, David & Gascon, Ferran & Gómez-Chova, Luis & Hagolle, Olivier & López-Puigdollers, Dan & Louis, Jerome & Lubej, Matic & Mateo-Garcia, Gonzalo & Osman, Julien & Peressutti, Devis & Pflug, Bringfried & Puc, Jernej & Richter, Rudolf & Žust, Lojze. (2022). Cloud Mask Intercomparison eXercise (CMIX): An evaluation of cloud masking algorithms for Landsat 8 and Sentinel-2. Remote Sensing of Environment. 274. 112990. 10.1016/j.rse.2022.112990.
5. Sklearn.linear_model.lasso (no date) Scikit-learn: Machine Learning in Python. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html (Acedido a: 29 April 2024)

6. Xian, G. Z., Smith, K., Wellington, D., Horton, J., Zhou, Q., Li, C., Auch, R., Brown, J. F., Zhu, Z., & Reker, R. R. (2022). Implementation of the CCDC algorithm to produce the LCMAP Collection 1.0 annual land surface change product. Earth System Science Data, 14(1), 143–162. https://doi.org/10.5194/essd-14-143-2022
7. Zhu, Zhe; Woodcock, Curtis E. (2014). Continuous change detection and classification of land cover using all available Landsat data. Remote Sensing of Environment, 144(), 152–171. doi:10.1016/j.rse.2014.01.011