LABORATÓRIOS DIDÁTICOS DE GEOPROCESSAMENTO

Sensoriamento Remoto

Cálculo do *NDVI* e classificação não-supervisionada de imagens no GEE

Docente: Prof.ª Dr.ª Mariana Abrantes Giannotti

Elaboração do roteiro: Leonardo Alves Godoy

Monitor: Christian de Nazareth Teixeira

Objetivos

- Cálculo do *NDVI* no GEE
- Classificação não supervisionada de imagens com *k-means* no GEE

1 Introdução

Este laboratório prático tem como objetivo apresentar o cálculo de um índice de vegetação no GEE e incluí-lo como um dos atributos em uma classificação não-supervisionada utilizando o algoritmo conhecido como *k-means*. Ao final, na parte extra, uma forma de buscar um número ótimo de *clusters* para o *k-means* é apresentada.

2 Referências teóricas

2.1 *NDVI*

Os Índices de vegetação, em sensoriamento remoto, são usados para avaliar coberturas vegetais. Podem ser obtidos através de cálculos com os dados da reflectância nas bandas do infravermelho próximo (*NIR*) e do vermelho visível (*RED*). Um dos índices de vegetação mais usados é o *NDVI* (*Normalized Difference Vegetation Index*). A fórmula do *NDVI* é dada por (*NIR*-*RED*)/(*NIR*+*RED*), conforme apresentado por TUCKER (1979). Na diferença entre *NIR* e *RED*, quanto maior, mais forte está a vegetação. As áreas sem vegetação devem obter valores baixos de *NDVI* (LILLESAND; KIEFER; CHIPMAN, 2015).

2.2 Classificação de imagens

O objetivo de uma classificação de imagens é separar os *pixels* dentro de uma imagem em classes (LILLESAND; KIEFER; CHIPMAN, 2015), ou seja, agrupá-los de acordo com algum critério. No caso de imagens de satélite, procura-se padrões espectrais semelhantes de acordo com a reflectância em cada *pixel* (LILLESAND; KIEFER; CHIPMAN, 2015). A classificação pode ser feita *pixel* a *pixel*

individualmente, por regiões ou por objetos (LILLESAND; KIEFER; CHIPMAN, 2015). Para este laboratório trabalharemos com um método *pixel* a *pixel*.

2.3 Classificação não-supervisionada

A classificação não-supervisionada, não utiliza nenhum tipo de conhecimento prévio como base para classificação, **não** tendo assim o processo chamado de treinamento (LILLESAND; KIEFER; CHIPMAN, 2015). Estes tipos de algoritmos checam os *pixels* e os combinam em grupos baseados em alguma forma na qual eles possam ser comparados. A questão que fica, nesse caso, é que as classes não possuem uma identidade, apenas suas divisões. Especificamente no caso de imagens de satélite, pode-se pensar em termos de classes espectrais, onde não se sabe o que cada uma representa na superfície e esta associação é feita a posteriori pelo analista.

2.4 *K-means*

Neste laboratório é utilizado um método de agrupamento conhecido como *k-means*. O *k-means* trabalha com base em medidas de similaridade entre os objetos (*pixels* no caso de uma imagem) a fim de agrupá-los quando são similares, ou próximos entre si. Também existem outros métodos que consideram as medidas de dissimilaridade que ao invés da proximidade, trabalha com a distância (diferença) entre os objetos.

Como os atributos (bandas) possuem valores numéricos e contínuos se utiliza uma métrica, ou medida, de distância. No caso, o usual, para o *k-means* com dados numéricos reais é o uso da distância euclidiana no espaço.

O princípio do *k-means* é buscar uma similaridade alta dentro dos grupos (*clusters*) e uma similaridade baixa entre os objetos de *clusters* distintos (BOEHMKE; GREENWELL, 2019).

De forma resumida, o *k-means* segue os seguintes passos:

1. Recebe um valor de entrada k <= n (onde n é o número de objetos do conjunto de dados), que é o número de grupos a ser formado e os dados que serão tratados;

2. Seleciona de forma aleatória - ou direcionada e otimizada, como no caso do *k-means++* descrito em ARTHUR e VASSILVITSKII (2006) - k objetos que serão as centróides iniciais;
3. Calcula a distância dos objetos para as centróides;
4. Cada objeto é associado com a centróide da qual ele tem a menor distância, fazendo assim parte desse grupo;
5. Recalcula as centróides de cada grupo;
6. Verifica se o centróide se deslocou consideravelmente:
    a. se não, termina o algoritmo,
    b. se sim volta para o passo 3.
7. Um critério de máximo de interações pode ser utilizado para a parada do algoritmo após um certo número de execuções.

3 Roteiro prático

3.1 Preparação dos dados no GEE

Inicialmente deve ser importada uma coleção de imagens que contenha valores de reflectância em seus *pixels*, não os números digitais (*DN*). Os valores de reflectância, como já estudado na disciplina, representam uma proporção, portanto não possuem unidade e estão entre 0 e 1 (ou 0 e 100). Para este laboratório didático deve ser feita a seleção de uma única imagem para o trabalho na região da cidade de São Paulo. Deve ser recortada alguma parte da imagem com vegetação (aqui, pode ser usado o ***"Map.addLayer()"*** para encontrar essa região) para se trabalhar utilizando o método ***"clip()"*** e uma *feature* com uma geometria, como um retângulo. Essa sugestão é para se obter um processamento mais rápido dos dados de estudo. As bandas que serão utilizadas devem ser escolhidas apropriadamente, identificando os seus comprimentos de onda nas informações da coleção. No exemplo da Figura 1 é criada uma lista com as bandas úteis para a execução da classificação na coleção ***"USGS Landsat 8 Collection 2 Tier 1 TOA Reflectance"***. Também é conveniente definir em uma variável a escala desejada (resolução espacial) para o processamento dos *pixels*. Essa, pode ser a resolução das bandas utilizadas - Figura 3.

Caso tenha optado pela coleção sugerida, com a reflectância de superfície, deve-se observar a informação nos metadados das bandas das imagens - ou em

seu manual disponível na *web*[1] - que indica que os valores estão em uma escala de 10.000 vezes o valor da reflectância. Sendo assim os valores precisam ser escalonados, como na Figura 2. Cabe aqui uma observação a respeito de um passo do pré-processamento, que deve ser levado em consideração. Caso verifique mais apropriadamente os valores dos *pixels*, após a reescala, provavelmente encontrará valores fora do intervalo entre 0 e 1. O ideal é se fazer o tratamento desses valores durante o pré-processamento dos dados. Uma das maneiras seria simplesmente mascarar os *pixels* (sendo esses *pixels* anulados e não utilizados nos cálculos). Para fazer isso, utiliza-se as informações da banda ***"radsat_qa"***, que apresenta, em uma sequência codificada de *bits*, quais bandas estavam muito saturadas (fenômeno chamado de *oversaturation*[2]) no *pixel* relativo durante aquela coleta dos dados. Informações similares a essas do *Landsat 8*, podem ser encontradas nos manuais do produtor das imagens das respectivas coleções. Portanto, basta importar a coleção, criar as *geometries* de referência, declarar no código as bandas selecionadas, especificar no código os parâmetros para o ***"clip()"*** e filtros de nuvens e datas como apresentado nos laboratórios anteriores e adicionar esse recorte ao mapa - Figura 1. Na sequência, ajustar os valores para o intervalo adequado - conforme Figura 2 - e declarar os parâmetros de ajuste de escala, conforme Figura 3.

Figura 1 - Bandas do *Landsat 8* para serem utilizadas no cálculo das componentes principais

```
Imports (3 entries)
  var colecaoLandsat8SR: (Deprecated) ImageCollection "USGS Landsat 8 Surface Reflectance Tier 1 [deprecated]"
  var saoPaulo: Point (-46.89, -23.64)
  var retanguloSelecao: Polygon, 4 vertices
1  //Seleção das bandas a serem trabalhadas
2  var bandas =ee.List(['B2','B3','B4','B5','B6','B7']);
3
4  //Seleção da imagem a ser trabalhada
5  var imagem = colecaoLandsat8SR.filterBounds(saoPaulo)
6                                .filterMetadata('CLOUD_COVER', 'less_than', 20)
7                                .filterDate('2019-01-01','2020-01-01')
8                                .first()
9                                .select(bandas)
10                               .clip(retanguloSelecao);
11
12 //Adicionando imagem com todas as bandas utilizadas
13 Map.addLayer(imagem.select(bandas),"","Recorte de São Paulo com vegetação");
```

```
//Seleção das bandas a serem trabalhadas
var bandas =ee.List(['B2','B3','B4','B5','B6','B7']);
```

[1] O em:
https://www.usgs.gov/media/files/landsat-8-collection-1-land-surface-reflectance-code-product-guide
[2] Manual disponível em:
https://www.usgs.gov/core-science-systems/nli/landsat/landsat-8-data-users-handbook

```
//Seleção da imagem a ser trabalhada
var imagem = colecaoLandsat8SR.filterBounds(saoPaulo)
                                  .filterMetadata('CLOUD_COVER', 'less_than',
20)
                                  .filterDate('2019-01-01','2020-01-01')
                                  .first()
                                  .select(bandas)
                                  .clip(retanguloSelecao);

//Adicionando imagem com todas as bandas utilizadas
Map.addLayer(imagem.select(bandas),"","Recorte de São Paulo com vegetação");
```

Figura 2 - Ajustando os valores dos *pixels* para o intervalo adequado

```
15  //Escala de reflectância
16  imagem = imagem.multiply(0.0001);
```

```
//Escala de reflectância
imagem = imagem.multiply(0.0001);
```

Figura 3 - Escala para ser utilizada no processamento da imagem

```
18  //Escala de pixels
19  var escala = 30;
```

```
//Escala de pixels
var escala = 30;
```

3.2 Calculando o *NDVI* no GEE

Para o cálculo do *NDVI*, basta fazer a aritmética das bandas, como já apresentado em laboratório anterior. Para deixar o código com melhor legibilidade, primeiramente pode-se extrair as bandas do *NIR* e do vermelho e colocá-las em variáveis separadas - Figura 4. Na Figura 5, é apresentada a aritmética das bandas conforme a orientação a objetos utilizada no GEE. A fórmula do *NDVI* primeiro faz a diferença entre os valores da banda *NIR* e da banda do vermelho visível, depois o valor resultante é dividido pela soma das mesmas duas bandas. Ainda, na Figura 5,

o resultado da operação aritmética é uma banda que é renomeada para *NDVI* com o método ***"rename()"***. Além disso, como pode ser observado, o método ***"addBands()"*** é acionado para adicionar as bandas dos cálculos efetuados. O segundo parâmetro do método ***"addBands()"*** é uma lista com o nome das bandas que devem ser adicionadas à imagem que chamou o método, neste caso apenas ***"NDVI"***. Finalmente o resultado da operação (um objeto alocado, mas sem nome) é atribuído à variável imagem, sobrescrevendo seus dados anteriores. Para visualizar o *NDVI* é possível usar a função: ***"Map.addLayer(imagem, {bands: ['NDVI'], min: -1, max: 1}, 'NDVI');"***.

Figura 4 - Extraindo as bandas do vermelho e do infravermelho próximo

```
21  // Extraindo as bandas do vermelho e do infravermelho próximo
22  var imagemBandaNir = imagem.select('B5');
23  var imagemBandaVermelho = imagem.select('B4');
```

```
// Extraindo as bandas do vermelho e do infravermelho próximo
var imagemBandaNir = imagem.select('B5');
var imagemBandaVermelho = imagem.select('B4');
```

Figura 5 - Código para cálculo do *NDVI* no GEE e exibição do mapa

```
25  // Código para cálculo do NDVI no GEE
26  imagem = imagem.addBands(
27                          imagemBandaNir.subtract(imagemBandaVermelho)
28                                        .divide(imagemBandaNir.add(imagemBandaVermelho))
29                                        .rename('NDVI'),['NDVI']);
30
31  //Adicionando imagem NDVI
32  Map.addLayer(imagem, {bands: ['NDVI'], min: -1, max: 1}, 'NDVI');
```

```
// Código para cálculo do NDVI no GEE
imagem = imagem.addBands(
                          imagemBandaNir.subtract(imagemBandaVermelho)

.divide(imagemBandaNir.add(imagemBandaVermelho))
                                        .rename('NDVI'),['NDVI']);

//Adicionando imagem NDVI
Map.addLayer(imagem, {bands: ['NDVI'], min: -1, max: 1}, 'NDVI');
```

Para o prosseguimento do script, a lista com os nomes da banda deve ser atualizada com o nome da banda *NDVI* - Figura 6.

Figura 6 - Adicionando um novo nome de banda na lista com todas as bandas

```
34  //Adicionando uma nova banda na lista com todas as bandas
35  bandas = bandas.add('NDVI');
```


```
//Adicionando uma nova banda na lista com todas as bandas
bandas = bandas.add('NDVI');
```


3.3 Executando o algoritmo do *k-means* no GEE

Há duas maneiras de se executar o *k-means* no GEE. Uma delas está dentro de ***"ee.Algorithms"***. A outra é baseada na implementação do pacote *WEKA*[3] dentro dos algoritmos da classe ***"ee.Clusterer"***. Aqui, será apresentada a implementação baseada no *WEKA* - mais poderosa - e como atividade extra opcional, ao fim do laboratório, a opção em ***"ee.Algorithms"*** será demonstrada.

3.3.1 *WEKA k-means*

Esta alternativa oferecida pelo GEE está na classe ***"ee.Clusterer"***. Neste caso, há uma estratégia para a execução da classificação, que diverge do *k-means* básico. Primeiramente, deve-se indicar para que o algoritmo faça uma seleção de amostra de *pixels*, como na Figura 7, com o método ***"sample()"*** chamado a partir da imagem que será classificada. O parâmetro é um dicionário de dados. De forma redundante, aqui, foi passada como região de abrangência (chave ***"region"***) o parâmetro do retângulo utilizado para seleção da imagem (o padrão seria utilizar a imagem toda, portanto nesse caso desnecessário, mas didático). Em ***"scale"*** a escala é ajustada como desejado, aqui sendo fornecido o próprio valor da resolução dos *pixels* da imagem. Finalmente, um número de *pixels* que devem ser amostrados é ajustado em ***"numPixels"***. O resultado será colocado em uma *FeatureCollection*. Na sequência, Figura 8, a amostra gerada é passada como parâmetro para o

[3] https://weka.sourceforge.io/doc.dev/weka/clusterers/SimpleKMeans.html#SimpleKMeans--

método ***"train()"*** do objeto do tipo ***"clusterer"*** gerado por ***"ee.Clusterer.wekaKMeans()"***. Isso tudo ocorre em uma mesma linha de código, mas o *clusterer* gerado poderia ser primeiramente atribuído a uma variável. O parâmetro do método que gera o *clusterer* (***"wekaKMeans()"***) é o número de *clusters* desejado. O resultado do código da Figura 8 é, também, um objeto *clusterer*, porém instanciado com base no treinamento, ou seja, já com os *clusters* calculados. Com este objeto alocado, pode ser chamado a partir da imagem o método ***"cluster()"*** - Figura 9, que aplica as características do *clusterer* gerado na imagem. Esse método aplica um algoritmo que busca iterativamente, para cada *pixel*, a centróide do cluster com menor distância a esse *pixel* e o atribui ao *cluster*. No GEE, o processo feito com ***"train()"*** é chamado treinamento, mas tem um sentido distinto do treinamento na classificação supervisionada que será visto nas próximas aulas. Aqui é simplesmente o cálculo das centróides da amostra, sem precisar rodar o algoritmo para toda a base de dados, o que em imagens muito grandes poderia ser lentíssimo. Para visualizar a classificação no mapa, basta utilizar o código da Figura 10. O resultado será uma imagem onde cada cor (gerada aleatoriamente) representa uma classe. Neste caso, uma imagem com cores geradas aleatoriamente é criada a partir da imagem original com a chamada do método ***"randomVisualizer()"***. A imagem para visualização é gerada com três bandas associadas às cores vermelho, verde e azul com uma sequência de valor para cada cluster encontrado.

Figura 7 - Selecionando uma amostra para executar o *k-means*

```
37  //Selecionando uma amostra para executar o k-means
38  var amostraTreinamento = imagem.sample(
39                                          {
40                                            region: retanguloSelecao,
41                                            scale:escala,
42                                            numPixels: 5000
43                                          }
44                                        );
```


```
//Selecionando uma amostra para executar o k-means
var amostraTreinamento = imagem.sample(
                                        {
                                          region: retanguloSelecao,
```

```
                                    scale:escala,
                                    numPixels: 5000
                                  }
                                );
```

Figura 8 - Executando o *k-means* na amostra (exemplo com 5 clusteres desejados)

```
46  //Executando o k-means na amostra
47  var clustererImagem = ee.Clusterer.wekaKMeans(5)
48                                    .train(amostraTreinamento);
```

```
//Executando o k-means na amostra
var clustererImagem = ee.Clusterer.wekaKMeans(5)
                                  .train(amostraTreinamento)
```

Figura 9 - Classificando os *pixels* de acordo com o resultado do *k-means* da amostra

```
50  //Classificando os pixels de acordo com o resultado do k-means da amostra
51  var imagemClassificadaWeka = imagem.cluster(clustererImagem);
```

```
//Classificando os pixels de acordo com o resultado do k-means da amostra
var imagemClassificadaWeka = imagem.cluster(clustererImagem);
```

Figura 10 - Adicionando a imagem classificada no mapa do GEE

```
53  // Adicionando a imagem classificada no mapa do GEE
54  Map.addLayer(imagemClassificadaWeka.randomVisualizer(),"", "Segmentação por k-means");
```

```
// Adicionando a imagem classificada no mapa do GEE
Map.addLayer(imagemClassificadaWeka.randomVisualizer(),"", "Classificação por
k-means");
```

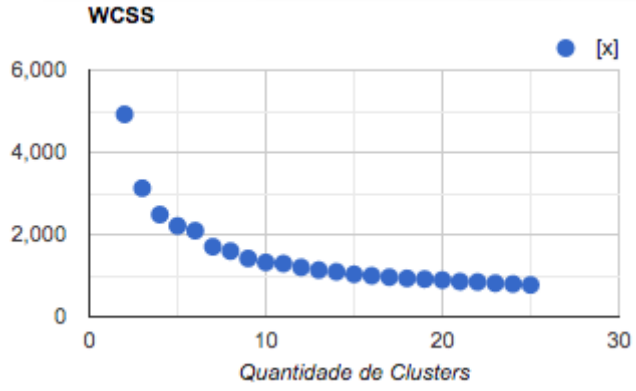
4 Indo além

4.1 Seleção de número de *clusters* ótimo

Ao fim desta seção é apresentado um código, nele pode ser observado um procedimento onde se tenta obter um número ótimo de clusters automaticamente.

Para isso, o *k-means* é processado diversas vezes, para diversos números de clusters a partir de dois e incrementando um a um, para a mesma imagem. Para cada uma das segmentações geradas, é calculada a soma da distância quadrática entre cada pixel e a centróide de seu respectivo cluster. Esse procedimento é feito para todos os clusters gerados por uma clusterização e a soma de todos os valores resultantes de cada cluster é feita. Essa é chamada Soma das Distâncias intra-cluster ou *Within Cluster Sum of Squares*, conhecido pela sigla *WCSS* (BOEHMKE ; GREENWELL, 2019). Faz-se isso para as clusterizações criadas de forma iterativa, e depois plota-se um gráfico em que no eixo x estão o número de clusters parametrizados para o *k-means* e no eixo y o *WCSS* - Figura 11. Visualmente, o que o gráfico apresenta, é uma distribuição cuja curva indica a partir de quando o esforço computacional deixa de fazer sentido dado o pouco ganho na diminuição da variação dos dados intra-cluster. Esse gráfico pode ser comparado com um outro gráfico onde o que se calcula é a variação entre clusters distintos, em que o objetivo é maximizar as distâncias. Ele também serve de base para o chamado *elbow method*, ou método do cotovelo (BOEHMKE ; GREENWELL, 2019). Fica como desafio aos interessados, buscar mais informações sobre os métodos de escolha de um número k de clusters ótimo.

Figura 11 - Gráfico do WCSS



Copie e cole os trechos de código da Figura 12 até a 17 - na ordem apresentada, em seu *script* - para efetuar o procedimento para avaliação de número ótimo de clusters.

Figura 12 - Gera lista com número de *clusters* para testar

```
/* Gera lista com uma sequência dos números de cluster que serão usados para
 parametrizar diversas classificações feitas com o k-means automaticamente. */
var numeroClusters = ee.List.sequence(2, 25);
```

Figura 13 - Função que vai executar o *k-means* com dado número de *clusters*

```
// Função para gerar o k-means na imagem dado um número de clusters.
var executaKmeans = function (numeroCluster) {
  // Cria um objeto do tipo clusterer treinando com a amostra de treinamento.
  var clustererImagem = ee.Clusterer.wekaKMeans(numeroCluster)
                                    .train(amostraTreinamento);
  /* Atribui os pixels de entrada aos seus respectivos clusters dados
   pelo treinamento. */
  return imagem.cluster(clustererImagem);
};
```

Figura 14 - *Map* da função para cada número de *clusters* da lista

```
// Mapeia a função para cada número de clusters na lista.
var imagensSegmentadas = numeroClusters.map(executaKmeans);
```

Figura 15 - Função que calcula a soma das diferenças quadráticas intra-cluster

```
/* As função a seguir calcula para cada imagem clusterizada a soma dos
   quadrados intra-clusters (Within Cluster Sum of Squares). */
var wcss = function (imagemKmeans) {
  var maxIdCluster =  ee.Image(imagemKmeans).reduceRegion(ee.Reducer.max());
  var listaIdClusters = ee.List.sequence(0, maxIdCluster.get('cluster'));
  var calculaDiferencaQuadraticaCluster = function (idCluster) {
    var pixelsCluster = imagem.mask(ee.Image(imagemKmeans)
                              .eq(ee.Number(idCluster).toInt()));
    var mediaPixelsCluster = pixelsCluster.reduceRegion(ee.Reducer.mean());
    var diferencaQuadratica = pixelsCluster.subtract(
                                          mediaPixelsCluster.toImage()
                                        )
                                       .pow(2);
    var diferencaQuadraticaSomaBandas = diferencaQuadratica.reduceRegion(
                                                              ee.Reducer.sum()
                                                            );
    return diferencaQuadraticaSomaBandas.values().reduce(ee.Reducer.sum());
  };
  return
listaIdClusters.map(calculaDiferencaQuadraticaCluster).reduce(ee.Reducer.sum());
};
```

Figura 16 - Faz o *map* da função *wcss()* para as imagens clusterizadas

```
// Mapeia para todas as imagens o cálculo da soma dos quadrados intra-cluster.
var resultadoWcss = imagensSegmentadas.map(wcss);
```

Figura 17 - Plota o gráfico com os resultados

```
// Plota o gráfico com a soma dos quadrados dentro dos clusters.
print(ui.Chart.array.values(resultadoWcss, 0, numeroClusters)
                      .setOptions({
                                    title: 'WCSS',
                                    hAxis: {title: 'Quantidade de Clusters'}
                                  }));
```

4.2 Alternativa para execução do *k-means* no GEE

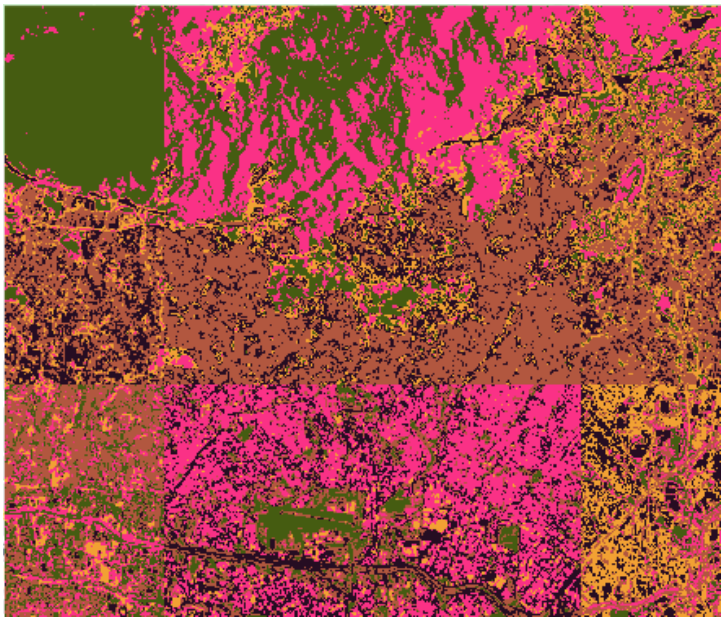
Na Figura 18 é apresentado o código para a execução do *k-means* implementado em ***"ee.Algorithms"***. É passado para o método ***"KMeans()"*** um dicionário de dados com os parâmetros para o algoritmo. Para a chave ***"image"*** é atribuída a imagem que se deseja classificar, em ***"numClusters"***, o número de clusters desejado é ajustado. O último parâmetro é particular desta alternativa. Ele apresenta se os números atribuídos como identificação de cada cluster devem ser únicos para toda a imagem (se for *true*) ou podem ser repetidos em cada pedaço da imagem (*false*). Esse conceito será entendido melhor no momento de se adicionar a imagem resultante ao mapa. O resultado da execução é uma imagem, onde em cada *pixel* há o número do *cluster* ao qual aquele *pixel* da imagem original pertence.

Figura 18 - Executando o *k-means* no GEE

```
// Executa o k-means implementado em ee.Algorithms na seleção.
var imagemClassificada = ee.Algorithms.Image
                                      .Segmentation
                                      .KMeans(
                                              {
                                                image: imagem,
                                                numClusters:5,
                                                uniqueLabels:false
                                              }
                                      );
```

Para visualizar a imagem classificada, basta utilizar a função ***"Map.addLayer()"*** como na Figura 10. O resultado da adição dessa imagem no mapa pode ser visto no exemplo da Figura 19. Na imagem adicionada, pode ser observado claramente limites em que a imagem se divide, gerando uma visualização confusa. Isso ocorre por causa do conceito de *tiles* do GEE. Uma *tile* nada mais é do que uma subdivisão da imagem e ela existe para tornar a computação mais eficiente. Essas *tiles* são carregadas desde as imagens importadas de uma coleção, por isso ao se adicionar uma imagem, algumas vezes, há a impressão de que ela vai sendo gerada aos poucos em quadrados. Cada quadrado desses é uma *tile* da imagem, que possui dimensões de 256x256 *pixels*. É possível reduzir as dimensões, porém não aumentar. Essa é uma característica inerente da plataforma do GEE. Se afastar a imagem no mapa com um *zoom* mais distante, pode ser observado que em certo momento essas *tiles* não serão mais observadas, sendo a classificação recalculada para a imagem toda (sempre que há um movimento de aproximação e distanciamento do mapa do GEE os mapas adicionados são recalculados). A explicação para isso é a re-escala dos *pixels* da imagem automaticamente (por exemplo de 30 metros para 60 metros) fazendo 256x256 *pixels* cobrirem a imagem toda no mapa.

Figura 19 - Visualização da classificação no mapa



5 Conclusões

Neste laboratório pode ser visto a simples forma de se calcular o *NDVI* no GEE. Outros índices podem ser calculados no GEE de forma similar com os conhecimentos aqui adquiridos. Também foram apresentados os passos necessários para a execução do algoritmo de classificação não-supervisionada *k-means* no GEE. Por fim, em uma etapa mais avançada, foi demonstrado como se buscar um número ótimo de clusters para o *k-means* e sua implementação no GEE.

6 Referências bibliográficas

ARTHUR, D.; VASSILVITSKII, S. k-means++: The advantages of careful seeding. Stanford, 2006.

BOEHMKE, B.; GREENWELL, B. M. Hands-on machine learning with R. CRC Press, 2019. Disponível em: https://bradleyboehmke.github.io/HOML/. Acesso em: 8 jun. 2021.

GARETH, J. et al. An introduction to statistical learning: with applications in R. Spinger, 2013. Disponível em: https://www.statlearning.com/. Acesso em: 8 jun. 2021.

LILLESAND, Thomas; KIEFER, Ralph W.; CHIPMAN, Jonathan. Remote sensing and image interpretation. John Wiley & Sons, 2015.

TUCKER, C. J. Red and photographic infrared linear combinations for monitoring vegetation. Remote sensing of Environment, v. 8, n. 2, p. 127-150, 1979.