4
Construção dos Classificadores

4.1.
Modelagem

O aprendizado supervisionado contém a etapa de modelagem, nessa etapa definimos quais serão as características encaminhadas ao classificador para o treinamento.

As características são valores obtidos através de informações presentes na imagem, cada *feature* possui uma característica específica. Um conjunto de valores de diferentes características faz parte de um exemplo.

Além do conjunto de valores das características, um exemplo contém a informação de qual classe ou categoria pertence esse mesmo conjunto de dados. A classe de um exemplo é uma das possíveis categorias presentes na imagem.

Abaixo, apresentamos uma figura exemplificando um conjunto de exemplos que contém a classe e os valores de cada *feature*. Esse conjunto de exemplos é denominado conjunto de treinamento.

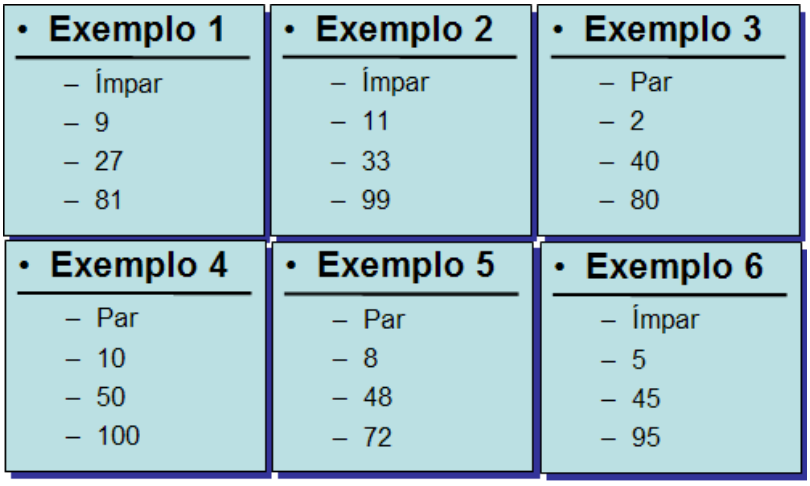

Figura 16 Exemplificação de um conjunto de exemplos

4.2. Conjunto de Treinamento

O aprendizado supervisionado consiste em fornecer ao classificador um conjunto de informações organizadas de acordo com a modelagem definida. Esse conjunto é denominado *Conjunto de Treinamento*, através desse conjunto modelado, o classificador é capaz de aprender sobre o problema.

O conjunto de treinamento é construído por uma intervenção externa humana, que define algumas informações do contexto geral como exemplos de treinamento ou testes. Apesar de o conjunto de treinamento ser composto de vários exemplos de treinamento e testes, apenas os exemplos de treinamento são utilizados pelo classificador na aprendizagem do problema.

Para o problema de classificação de imagens de sensoriamento remoto, tanto os exemplos de treinamento quanto os de teste são etiquetados da seguinte forma: primeiro, alguns pixels na imagem são selecionados aleatoriamente, e; após a seleção, o especialista define uma classe para cada exemplo escolhido.

Caso haja dúvida na etiquetagem do especialista, ou até mesmo para confirmar a marcação realizada pelo perito, as classes dos exemplos são obtidas em campo, ou seja, as informações sobre as categorias desses pixels são obtidas no local referente aos exemplos selecionados na imagem.

Após essa fase de etiquetagem dos exemplos indicados, garantimos que temos as classes de uma pequena parte do conjunto de dados, e a partir desse conjunto de treinamento construído, iremos realizar a classificação do restante da imagem.

Na Figura 17, exibimos uma imagem com um conjunto de treinamento etiquetado por um especialista.

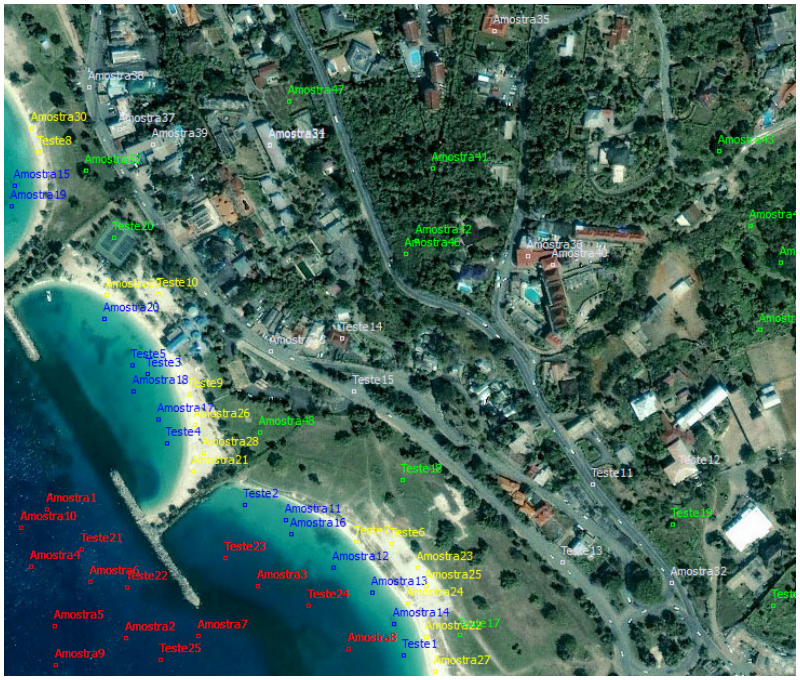

Figura 17 Região de Montego Bay, Jamaica, etiquetada por um especialista

Abaixo apresentamos a tabela, que informa os valores RGB e HSB de alguns exemplos do conjunto de treinamento da figura acima.

Exemplo	Classe	R	G	B	H	S	B
Amostra 5	Água	1	33	71	142	233	34
Amostra 15	Costa	0	144	128	116	240	68
Amostra 25	Areia	239	249	224	56	162	223
Amostra 35	Urbano	171	126	105	13	68	130
Amostra 45	Vegetação	60	117	72	81	77	83
Teste 3	Costa	17	162	145	115	199	84
Teste 13	Urbano	113	145	130	101	30	121
Teste 23	Água	0	67	94	131	240	44

Tabela 2 Informações RGB e HSB de alguns exemplos do conjunto de treinamento

4.3. Treinamento

O objetivo da fase de treinamento é fornecer conhecimentos ao classificador de acordo com a modelagem pré-determinada, ou seja, prover as informações ao classificador de acordo com a modelagem escolhida. Após o treinamento do classificador, o mesmo está apto a realizar a predição do conjunto de dados.

A etapa de treinamento do classificador pode ser executada por várias técnicas de aprendizado de máquina. Desenvolvemos a etapa de treinamento do classificador de imagens empregando a metodologia de Support Vector Machines.

Implementamos o classificador de imagens baseado em Support Vector Machine, através da biblioteca LIBSVM[21], que oferece suporte de execução da metodologia citada.

A biblioteca LIBSVM proporciona uma API (Application Programming Interface) para diversas linguagens de programação e a utilizada no protótipo foi a API de C++, a mesma oferece cinco tipos de implementação do SVM (*C-SVC*, *nu-SVC*, *one-class SVM*, *epsilon-SVR* e *nu-SVR*) e quatro tipos de Kernel (Linear, Polinomial, Função de Base Radial e Sigmoid), com a possibilidade de ajuste dos parâmetros de cada função de Kernel.

A classificação multiclasses desenvolvida pela LIBSVM utiliza a decomposição *One-Against-One*.

4.4. Os Classificadores

A modelagem é fundamental na etapa de treinamento dos classificadores, pois, nessa fase, definimos quais são as características que o classificador utilizará para aprender sobre o problema.

A fim de resolver o problema de classificação de imagens de sensoriamento remoto, construímos três modelos. Esses modelos contêm informações dos canais de cores RGB, HSB e Infravermelho, valor altimétricos e, junto das características básicas, foram incluídas a informação do contexto de um pixel selecionado.

O contexto de um pixel é a informação relacionada a uma região vizinha ao pixel, ou seja, além do pixel selecionado, informações dos pixels vizinhos foram incluídas aos modelos. Dividimos as informações de contexto em Contexto de Cores e Contexto Altimétrico.

O Contexto de Cores representa os canais de cores RGB e HSB dos pixels vizinhos ao pixel selecionado, os pixels que formam o contexto são definidos pelo nível de vizinhança. Definimos nível de vizinhança v como os v primeiros pixels vizinhos ao pixel selecionado em todas as direções.

O Contexto Altimétrico são os valores de altimetria dos pixels a uma distância fixa do pixel selecionado. Abaixo, exibimos uma figura que representa o

contexto de cores com nível de vizinhança 2(dois) e o contexto de altimetria com nível de vizinhança 5(cinco).

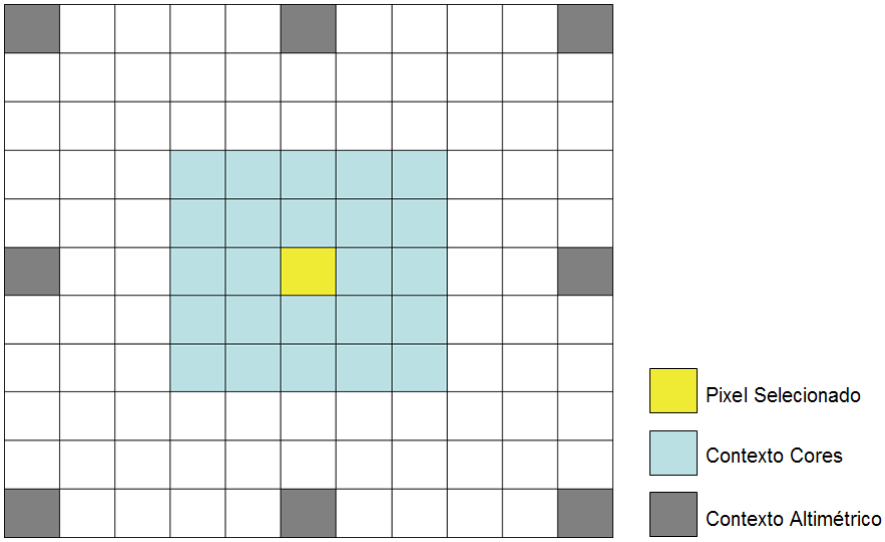

Figura 18 Exemplificação do contexto de um pixel, com contexto de cores em nível dois e contexto de altimetria em nível cinco.

Definimos o primeiro modelo, **RGB+HSB**, que contém as seguintes características: os valores dos canais de cores RGB e HSB do pixel selecionado e de seu contexto, e a média e a variância dos valores dos canais de cores RGB e HSB do pixel selecionado e de seu contexto.

Numeramos as características do modelo **RGB+HSB** da seguinte forma:

- Canais RGB do pixel selecionado: 3 características;
- Canais HSB do pixel selecionado: 3 características;
- Canais RGB do contexto de cores:
  - Nível de Vizinhança 1: 24 características
  - Nível de Vizinhança 2: 72 características
  - Nível de Vizinhança 3: 144 características
  - Nível de Vizinhança 5: 360 características
  - Nível de Vizinhança 10: 1320 características
- Canais HSB do contexto de cores:
  - Nível de Vizinhança 1: 24 características
  - Nível de Vizinhança 2: 72 características
  - Nível de Vizinhança 3: 144 características
  - Nível de Vizinhança 5: 360 características
  - Nível de Vizinhança 10: 1320 características
- Média e Variância dos canais RGB: 6 características;

- Média e Variância dos canais HSB: 6 características;

O modelo **RGB+HSB** possui no mínimo 66 características ou no máximo 2658 características, de acordo com o nível de vizinhança do contexto de cores.

O segundo modelo, **RGB+HSB+Altimetria**, é uma evolução do primeiro, ou seja, incluímos o valor de altimetria do pixel selecionado e de seu contexto altimétrico, a média, a variância do valor de altitude do pixel e de seu contexto altimétrico e os valores de mínimo e máximo do pixel selecionado e de seu contexto altimétrico.

Numeramos as características do modelo **RGB+HSB+Altimetria** da seguinte forma:

- Características do modelo **RGB+HSB**;
- Valor Altimétrico do pixel selecionado: 1 feature;
- Valor Altimétrico do contexto altimétrico: 8 características;
- Média e Variância dos valores altimétricos: 2 características;
- Mínimo e Máximo dos valores altimétricos: 2 características;

Logo, o modelo **RGB+HSB+Altimetria** possui no mínimo 79 características ou no máximo 2671 características, de acordo com o nível de vizinhança do contexto de cores.

Por fim, o terceiro modelo, **RGB+HSB+Altimetria+IR**, que é uma evolução do segundo modelo, adicionamos o valor do canal infravermelho do pixel selecionado e de seu contexto de cores e a média e a variância do pixel selecionado e de seu contexto de cores.

Assim, numeramos as características do modelo **RGB+HSB+Altimetria+IR** da seguinte forma:

- Características do modelo **RGB+HSB+Altimetria**;
- Canal IR do pixel selecionado: 1 feature;
- Canais IR do contexto de cores:
  - Nível de Vizinhança 1: 8 características
  - Nível de Vizinhança 2: 24 características
  - Nível de Vizinhança 3: 48 características
  - Nível de Vizinhança 5: 120 características
  - Nível de Vizinhança 10: 440 características
- Média e Variância do canal IR: 2 características;

Logo, o modelo **RGB+HSB+Altimetria+IR** possui no mínimo 90 características ou no máximo 3114 características, de acordo com o nível de vizinhança do contexto de cores.

Criamos três classificadores, empregando a metodologia de aprendizado de máquina Support Vector Machines, utilizando os modelos **RGB+HSB**, **RGB+HSB+Altimetria** e **RGB+HSB+Altimetria+IR**.

Os três classificadores descritos acima foram construídos empregando o modelo de programação *multithread*, que permite a execução de múltiplas linhas de execução de forma independente, dentro de um conjunto simples de linhas de execução.

O problema de classificação de imagens pode facilmente ser implementado utilizando o modelo de programação *multithread*, basta dividir a área de classificação da imagem no número de *threads* em que a operação será executada, ou seja, cada *thread* fica responsável em classificar uma parte da imagem.

Atualmente, são disponibilizadas imagens muito grandes e com alta resolução e, através desse modelo de programação, é esperado que o tempo de execução seja reduzido, já que o mesmo pode ser demorado devido ao tamanho das imagens.

Abaixo, apresentamos um exemplo de como uma imagem pode ser dividida em quatro *threads* diferentes.

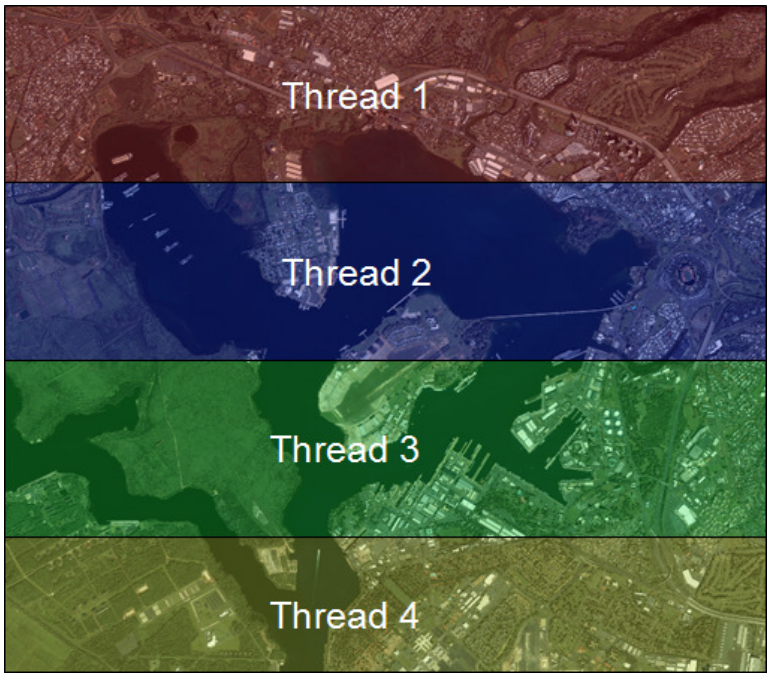

Figura 19 Exemplo de como uma imagem pode ser classificada separadamente em quatro *threads* distintas.

4.5. Codificação

Apresentaremos a seguir, a codificação das principais etapas do problema de classificação de imagem. O primeiro pseudocódigo reflete a iteração do usuário com a aplicação, em que o mesmo é responsável por criar as classes e etiquetar os exemplos, construindo o conjunto de treinamento.

```
INICIO
  Dados de Entrada:
        Tarefa
        Imagens de Entrada
  Dados de Entrada e Saída:
        Conjunto de Treinamento
  SE Tarefa igual Criar Classe ENTÃO
        Definir Nome para Classe
        Definir Cor para Classe
        Adicionar Classe ao Conjunto de Treinamento
  SENÃO Tarefa igual Etiquetar Exemplo ENTÃO
        Definir Identificação do Exemplo
        Definir Posição do Exemplo
        Definir Tipo de Exemplo entre Treinamento e Teste
        Adicionar Exemplo a Classe
  FIM-SE
FIM
```

Algoritmo 1 Pseudo algoritmo referente à iteração do usuário com o protótipo

O segundo pseudocódigo reproduz a fase de aprendizagem do conjunto de treinamento criado pelo usuário, construindo um conjunto de paradigmas.

```
INICIO
  Dados de Entrada:
        Conjunto de Treinamento
        Modelo do Classificador
        Imagens de Entrada
  Dados de Saída:
        Conjunto de Paradigmas
  PARA Cada Exemplo de Treinamento do Conjunto de Treinamento FAÇA
        Paradigma.Rótulo ← Exemplo.Classe
```

PARA Cada Feature do Modelo FAÇA

Calcular o valor da Feature utilizando as informações das Imagens de Entrada e a Posição do Exemplo

Adicionar a Feature ao Paradigma

FIM-PARA

Adicionar o Paradigma ao Conjunto de Paradigmas

FIM-PARA

FIM

Algoritmo 2 Pseudo algoritmo referente à etapa de aprendizagem do conjunto de treinamento

Os próximos dois pseudocódigos reportam a etapa de treinamento do classificador e a etapa da classificação da imagem, empregando a lista de paradigmas e a imagem a ser classificada. A resposta dessa operação é a imagem classificada da imagem de entrada.

INICIO

Dados de Entrada:

Imagens de Entrada

Conjunto de Paradigmas

Número de Threads

Modelo do Classificador

Dados de Saída:

Imagem Classificada

Construir o Preditor baseado no Conjunto de Paradigmas, através da técnica de Support Vector Machines,

Criar as Threads, dado o Número de Threads

Dividir a Imagem de Entrada pelo Número de Threads

PARA Cada Thread FAÇA

Associar uma das regiões a Thread

Associar o Preditor a Thread

Associar o Modelo a Thread

Executar a Thread

FIM-PARA

Unificar as respostas de cada Thread, gerando a Imagem Classificada

FIM

Algoritmo 3 Pseudo algoritmo referente à etapa de classificação

O pseudocódigo abaixo detalha a execução de uma thread, responsável por classificar uma região da imagem de entrada.

INICIO
Dados de Entrada:
Imagens de Entrada
Região
Preditor
Modelo do Classificador
Dados de Saída:
Imagem Classificada da Região
PARA Cada Pixel da Região FAÇA
Criar Paradigma
PARA Cada Feature do Modelo FAÇA
Calcular o valor da Feature utilizando as informações das Imagens de Entrada e a Posição do Pixel
Adicionar a Feature ao Paradigma
FIM-PARA
Classificar o Paradigma, utilizando o Preditor
Atribuir a classe ao Pixel
FIM-PARA
FIM

Algoritmo 4 Pseudo algoritmo referente à execução de uma thread responsável em classificar uma parte da imagem de entrada