Capítulo 6 - Aplicações de SVM Usando Imagens

Introdução

Usar técnicas de classificação, como a SVM, para aplicações em imagens é de grande interesse e importância, tanto para a academia como para o mercado. Como exemplo, na medicina, é possível usar software para auxiliar o diagnóstico médico, chamado CAD (*Computer Aided Diagnosis*), analisando imagens de raio x, ultrassom, ressonância, tomografia, microscópio eletrônico, raio laser, etc. Este último tipo de imagens, conhecido como tomografia de coerência óptica (OCT, sigla em inglês), auxilia médicos no diagnóstico usando laser de baixa frequência, criando imagens tridimensionais de tumores.

Outro fator que motiva criar Sistemas Especialistas para análise de imagens é a existência de sistemas de gestão de exames, implantados nos principais hospitais e laboratórios, onde os exames são realizados e os resultados são digitalizados, inclusive as imagens dos exames. Assim, já existe uma gigantesca base de imagens, que poderia treinar vários Sistemas Especialistas, quando o diagnóstico já existe no prontuário do paciente.

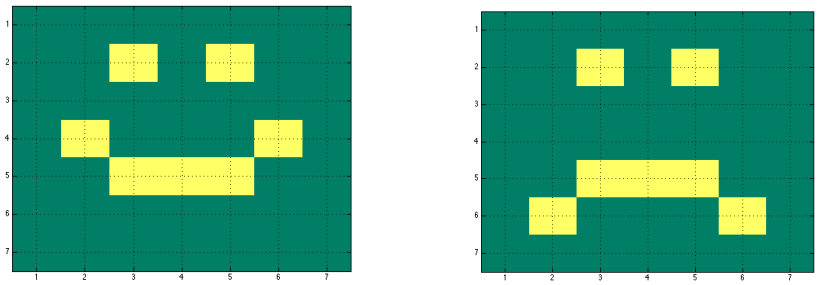
Além das imagens médicas, também existem grandes demandas para análise de imagens em dispositivos móveis, para os quais aplicações estão começando a surgir, como o reconhecedor de faces em aplicativos com câmeras em *smartphones*.

Outra aplicação recente é a busca por imagens na *web*, como o aplicativo *Google Goggles,* que, com base numa foto de um objeto retorna páginas na *web* com imagens semelhantes. Esta busca por imagem ainda apresenta muitos erros, parecendo ser uma busca pelo histograma da imagem (isto será explicado neste capítulo).

Desta forma, se ressalta a importância em criar sistemas sofisticados para a análise de imagens, e este capítulo faz uma introdução à classificação de imagens usando SVM.

Introdução à Classificação de Imagens Usando Weka e MATLAB

O leitor pode reproduzir os experimentos deste capítulo usando o software MATLAB®, versão 2010, para criar as imagens e gerar o arquivo ".arff", que deverão ser lidos pelo Weka. Foram feitos testes de classificação em imagens usando o classificador SVM. Inicialmente foram geradas imagens sintéticas simples de uma face feliz e outra triste, como ilustra a Figura 6.1.



(a) (b)

Figura 6.1 – Face (a) Feliz e Face (b) Triste.

Essas imagens são matrizes quadradas de dimensão 7x7, onde o canto superior esquerdo é convencionado no MATLAB® como o pixel (1,1), a primeira coordenada é o eixo vertical e a segunda coordenada, o eixo horizontal. Assim, os pixels dos olhos e das bocas das duas imagens da Figura 6.1 são definidos com o valor 1 (um) e o restante com o valor 0 (zero). Definimos uma imagem com dimensões 7x7 (Figura 6.1(a)) no MATLAB®, com os seguintes comandos:

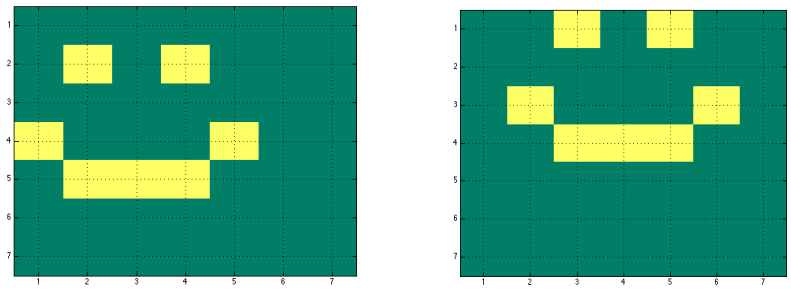
```
img_a = zeros(7,7); % cria imagem img_a com dimensões 7x7
img_a(2,3) = 1; img_a(2,5) = 1; % define os olhos
img_a(4,2) = 1; img_a(4,6) = 1; img_a(5,3:5) = 1; % define a boca feliz
```

Respectivamente, de forma semelhante para a imagem da Figura 6.1(b):

```
img_b = zeros(7,7); % cria imagem img_b com dimensões 7x7
img_b(2,3) = 1; img_b(2,5) = 1; % define os olhos
img_b(6,2) = 1; img_b(6,6) = 1; img_b(5,3:5) = 1; % define a boca triste
```

Em seguida, para gerar outras amostras das imagens feliz e triste, foram geradas translações aleatórias de 1 pixel na horizontal, Figura 6.2(a) (respectivamente, para a direção vertical, Figura 6.2(b)). Para a translação horizontal (respectivamente vertical), foram utilizados os seguintes comandos no MATLAB®:

```
trans = 1; % para translações de um pixel na horizontal ou na vertical
% Horizontal
img = imdilate(img,translate(strel([1]),[0 -trans+round(rand*2*trans)]));
% Vertical
img = imdilate(img,translate(strel([1]),[-trans+round(rand*2*trans) 0]));
```



(a) (b)

Figura 6.2 - Translação (a) Horizontal e (b) Vertical.

Nestes comandos, foi utilizada a dilatação com elementos estruturantes especiais, onde o objetivo do comando *–trans+round(rand*2*trans)* é fornecer um número aleatório entre *-trans* e *trans* (para detalhes, consulte o *help* do MATLAB®). Assim, a imagem *img* é transladada de forma aleatória para a esquerda ou direita. Cálculo análogo é realizado para a translação vertical. Veja no código a seguir a função completa construída no MATLAB® para gerar imagens aleatórias e também o arquivo ".arff", que será usado no Weka para obter o resultado da classificação usando SVM:

```
function imagens_aleatorias(str, resposta, NUM_IMGS, img)
% str          – string com o nome do arquivo ".arff"
% resposta     – string com valores "feliz" ou "triste"
% NUM_IMGS     – número de amostras aleatórias geradas nesta função
% img          – imagem de entrada feliz ou triste
img_ori = img;
for i=1 : NUM_IMGS
    img = img_ori;
    trans = 1; % translação +/- trans pixels na horizontal/vertical
    % translação horizontal
    img = imdilate(img,translate(strel([1]),[0 -trans+round(rand*2*trans)]));
    % translação vertical
    img = imdilate(img,translate(strel([1]),[-trans+round(rand*2*trans) 0]));

    % varre primeiro as linhas da imagem e cada pixel define um atributo
    x1 = img(1,1);
    x2 = img(1,2);
    x3 = img(1,3);
    …
    x49= img(7,7);
    fid = fopen(str,'r'); % lê arquivo str
    if fid == -1 % se arquivo não existe, incluir cabeçalho
        fid = fopen(str,'at+');
        fprintf(fid,'%s\n','@RELATION face');
        fprintf(fid,'%s\n','@ATTRIBUTE class    {triste,feliz}');
        fprintf(fid,'%s\n','@ATTRIBUTE x1 INTEGER');
```

```
        fprintf(fid,'%s\n','@ATTRIBUTE x2 INTEGER');
        …
        fprintf(fid,'%s\n','@ATTRIBUTE x49 INTEGER');
        fprintf(fid,'%s\n','@DATA');
    else % senão, incluír os atributos da nova imagem feliz ou triste
        fclose(fid);
        fid = fopen(str,'at+');
    end
% o próximo comando imprime “feliz” ou “triste”, mais os 49 atributos (pixels)
fprintf(fid,'%s,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d
,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5
d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d,%5d\n',...
resposta,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21
,x22,x23,x24,x25,x26,x27,x28,x29,x30,x31,x32,x33,x34,x35,x36,x37,x38,x39,x40,x41,x4
2,x43,x44,x45,x46,x47,x48,x49);
    fprintf(fid,'\n');
    fclose(fid); % fecha arquivo
end
```

Um exemplo de uso desta função é `imagens_aleatorias(‘face50.arff’, ‘feliz’, 50, img_a)`, e após a execução deste comando no MATLAB®, esta função salva no disco o seguinte fragmento de arquivo “.arff”, gerando 50 imagens da face feliz (Figura 6.1 (a)) com translações aleatórias de um pixel para a horizontal ou vertical. Este arquivo `‘face50.arff’` é usado como entrada de dados no Weka:

```
@RELATION face
@ATTRIBUTE class    {triste,feliz}
@ATTRIBUTE x1 INTEGER
@ATTRIBUTE x2 INTEGER
…
@ATTRIBUTE x49 INTEGER
@DATA
feliz,
0,    0,    1,    0,    1,    0,    0,
0,    0,    0,    0,    0,    0,    0,
0,    1,    0,    0,    0,    1,    0,
0,    0,    1,    1,    1,    0,    0,
0,    0,    0,    0,    0,    0,    0,
0,    0,    0,    0,    0,    0,    0,
0,    0,    0,    0,    0,    0,    0

feliz,
0,    0,    0,    0,    0,    0,    0,
0,    0,    1,    0,    1,    0,    0,
0,    0,    0,    0,    0,    0,    0,
0,    1,    0,    0,    0,    1,    0,
0,    0,    1,    1,    1,    0,    0,
0,    0,    0,    0,    0,    0,    0,
0,    0,    0,    0,    0,    0,    0
…

triste,
0,    0,    0,    0,    0,    0,    0,
0,    0,    1,    0,    1,    0,    0,
0,    0,    0,    0,    0,    0,    0,
0,    0,    0,    0,    0,    0,    0,
0,    0,    1,    1,    1,    0,    0,
0,    1,    0,    0,    0,    1,    0,
0,    0,    0,    0,    0,    0,    0
```

Usando a função anterior para gerar 50 amostras para a imagem feliz e 50 para a triste, o classificar SVM implementado no Weka obteve 100% de acerto, como mostra a seguinte matriz de confusão:

```
a       b   <-- classified as
50      0   |  a = triste
0       50 |  b = feliz
```

Para translações horizontais e verticais ao mesmo tempo, como mostra a Figura 6.3, foram gerados dois conjuntos de amostras ao acaso de 50 translações cada. Na primeira, também foi obtido 100% de acerto. Porém, no segundo conjunto de amostras aleatórias geradas foi encontrada a seguinte matriz de confusão:

```
 a      b   <-- classified as
47      3   |  a = triste
1       49 |  b = feliz
```

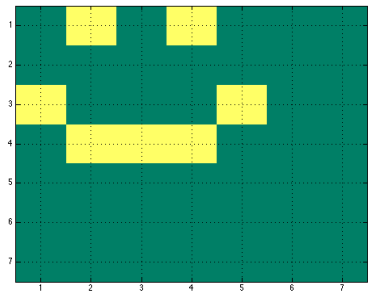


Figura 6.3 – Face Transladada Horizontal e Vertical.

Neste caso com 96% de acerto. Ou seja, três imagens triste foram classificadas como feliz e uma imagem feliz foi classificada como triste. Um dos casos é apresentado na Figura 6.4.

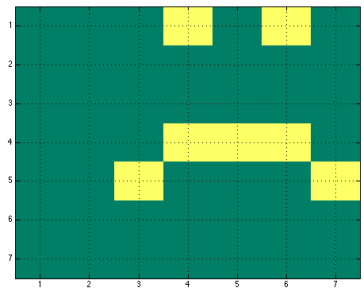


Figura 6.4 – Caso de Erro, com Translações Horizontais e Verticais.

Agora, aumentando o número de imagens para 1000 amostras de translações aleatórias de um pixel na vertical e na horizontal foi obtido 100% de acerto. Este processo foi repetido 5 vezes e em todos os casos o resultado foi o mesmo. Ou

seja, quando maior o número de amostras, melhor será o resultado da classificação.

Em seguida, foram realizados testes de translação com mais de um pixel na horizontal e na vertical. Em uma das simulações foi usada uma translação com 2 pixels. Neste caso, a boca e/ou os olhos poderão ser excluídos da imagem, como mostra a Figura 6.5, dificultando ainda mais o processo de classificação. Como resultado, foi obtida a seguinte matriz de confusão, considerando 100 amostras:

```
 a    b   <-- classified as
37   13 |  a = triste
 8   42 |  b = feliz
```

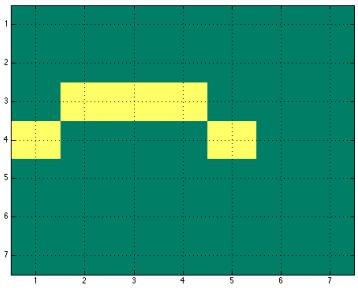


Figura 6.5 – Foto Sem os Olhos.

Como esperado, foi obtido 79% de acerto. Rodamos novamente para outras 100 amostras, obtivemos 72% de acerto. Aumentando para 1000 amostras obtivemos 94.8%, com a seguinte matriz de confusão:

```
  a    b   <-- classified as
472   28 |  a = triste
 24  476 |  b = feliz
```

No lugar de fazer translações, também podemos utilizar rotações e reflexão. Finalmente, foram utilizados ruídos aleatórios de um pixel, dois pixels etc., de acordo com o seguinte trecho de código no MATLAB®:

```
% adiciona ruídos de um pixel na imagem 7x7
num = sum(sum(img));
aux = img;
while (num~=8) % para adicinar dois pixels de ruído, mudar de 8 para 9
    img = aux;
    img = max(rand(7)>0.95,img);
    num = sum(sum(img));
end
```

Para imagens com ruídos de um pixel a mais, ou seja, para 8 pixels com valor 1, foi obtido 100% de acerto para 100 amostras (o mesmo se repetiu para 9, 10 e 11 pixels, com geração de vários conjuntos de 100 amostras diferentes). Já para 12 pixels, ou seja com 5 pixels a mais de ruído, foi obtido 99% de acerto. Foram repetidos vários testes com 12 pixels com valor 1 e o resultado se manteve próximo a 99% de acerto. A Figura 6.6 ilustra um dos resultados classificados erroneamente.

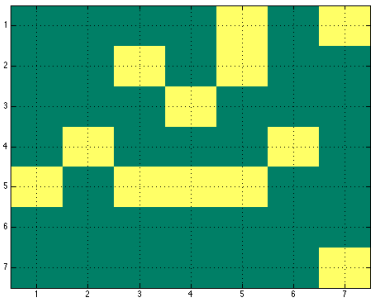


Figura 6.6 – Imagem com Ruídos Classificada Erroneamente.

Como conclusões destes testes, podemos ressaltar que não é possível considerar todos os pixels de uma imagem como atributos, pois para uma imagem de 1000x1000 pixels (que é um tamanho razoável para imagens tratadas atualmente) o número de atributos utilizados no Weka torna-se inviável. Além disso, para uma simples imagem sintética 7x7 verificamos que variações geométricas de translação e da inclusão de ruídos, a classificação fica comprometida, sendo necessário aumentar o número de amostras significativamente para melhorar a classificação.

Além disso, é aconselhável usar atributos mais significativos para diminuir a quantidade de atributos sem comprometer o desempenho da classificação. Por exemplo, para o caso de imagens da face, é possível ter como atributos: o número de componentes conexos (pixels com valores 1's conectados usando vizinhança 8, por exemplo, ou seja, tomando-se o centro de um subconjunto 3x3

da imagem e verificando se existe algum dos 8 vizinhos também com valor 1), assim, estes pixels com valores 1's pertencem a um componente conexo; a área de cada componente conexo; o fecho convexo dos objetos, isto é, as bordas convexas nos olhos e na bocas das imagens da Figura 6.1 – o fecho convexo da face triste possui área maior que a área da face feliz etc. Estes atributos, que representam a topologia dos objetos, podem ser mais representativos ao distinguir diferentes tipos de objetos.

O MATLAB® oferece também recursos para calcular medidas geométricas de cada objeto usando o comando **regionpropos** – para mais informações deste comando, consulte o *help* do MATLAB®. Porém, antes de analisar a topologia de cada objeto, a próxima seção apresenta uma técnica de classificação de imagens usando histogramas.

Classificação de Imagens Usando Histogramas

Como uma aplicação de classificação de imagens reais, em que não é possível considerar cada pixel da imagem com um atributo, nesta seção consideramos apenas os valores dos pixels das imagens, sem a preocupação com as formas que estes pixels possam representar na imagem.

Considere a imagem da Figura 6.7. Esta é uma imagem colorida, usando o padrão de cores RGB (do inglês: *Red* (vermelho), *Green* (verde) e *Blue* (azul), respectivamente). Esta imagem pode ser lida no MATLAB® usando o comando `img = imread('Lena.jpg');`, criando a variável `img`. Para visualizar esta imagem, digite o comando `imshow(img)`, como mostra a Figura 6.7.

Para verificar as dimensões desta variável `img`, basta digitar `whos` no MATLAB®. Na Figura 6.8, é possível observar que a imagem `img` possui dimensões 512x512x3. O último valor 3 representa as cores RGB, ou também chamadas de "as três bandas da imagem". O campo *Class* nesta Figura 6.8 representa o tipo de dados de cada pixel, em cada uma das três bandas. Neste exemplo o tipo é *uint8*, que significa um *byte* que pode assumir valores entre 0 e 255, representando tons de cinza.

Figura 6.7 – Imagem em Cores, com Dimensões 512x512x3.

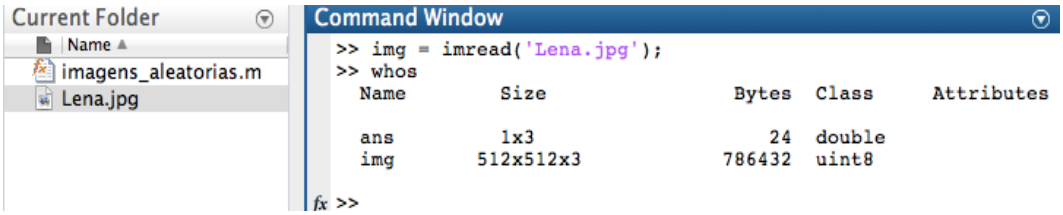


Figura 6.8 –Comando Usado para Ler uma Imagem e Mostrar as suas Características.

Para os nossos testes de classificação usando histogramas, é possível analisar apenas uma banda de cores, por exemplo, digitando o comando *imgNC=img(:,:,2);*. Neste comando é considerado apenas a segunda banda da imagem. Para visualizar esta imagem, digite *imshow(imgNC)*, como mostra a Figura 6.9.

Para visualizar o histograma de uma imagem, use o comando `imhist(imgNC).` A Figura 6.10 mostra o resultado deste comando.

Figura 6.9 –Imagem em Nível de Cinza.

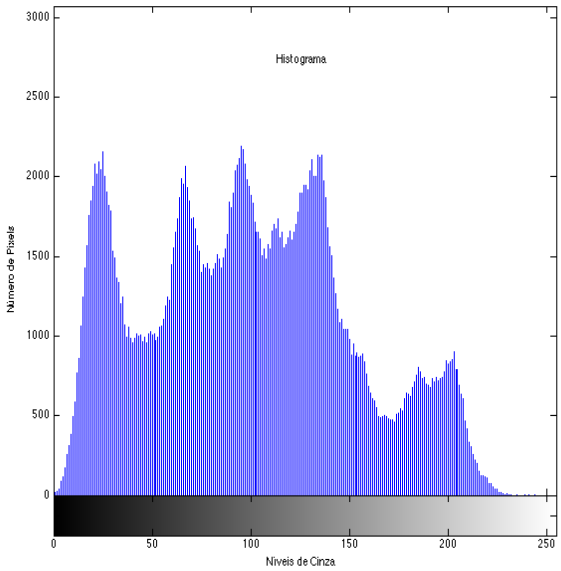
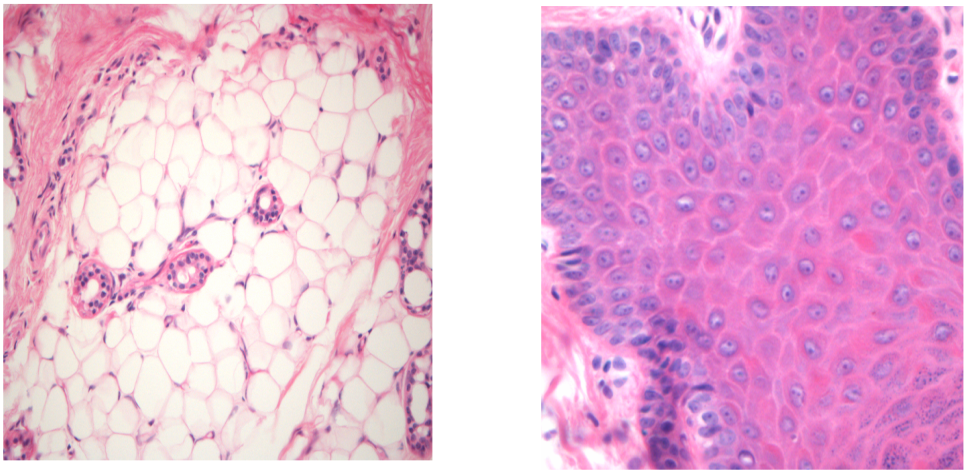


Figura 6.10 - Histograma da Imagem imgNC.

Agora, já é possível fazer um experimento completo de classificação em imagens usando como atributos o histograma de cada imagem, isto é, os 256 atributos

referentes à cor de cada pixel para cada imagem, como apresentado na Figura 6.10. Para estes testes serão usadas 10 imagens, sendo 5 do tecido adiposo (Figura 6.11(a)) e 5 do tecido epitelial (Figura 6.11(b))[9]. Estas imagens também foram classificadas usando medidas topológicas, como apresentado em (ZAMPIROLLI *et al.*, 2010)[10] e resumido na próxima seção. Usando apenas o histograma na segunda banda das imagens (veja Figuras 6.12 e 6.13), a respectiva matriz de confusão é apresentada a seguir, com 100% de acerto. Esta facilidade na classificação ocorre pois as imagens do tecido adiposo são mais claras (muitos pixels com valores próximos do 255), quando comparadas com as imagens do tecido epitelial, como observado nos histogramas da Figura 6.13.

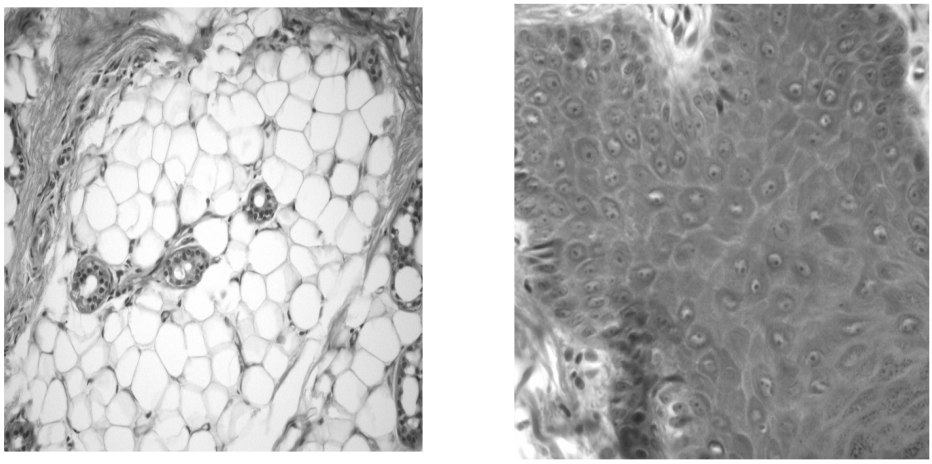
```
a b   <-- classified as
5 0 | a = epitelial
0 5 | b = adiposo
```



(a) (b)

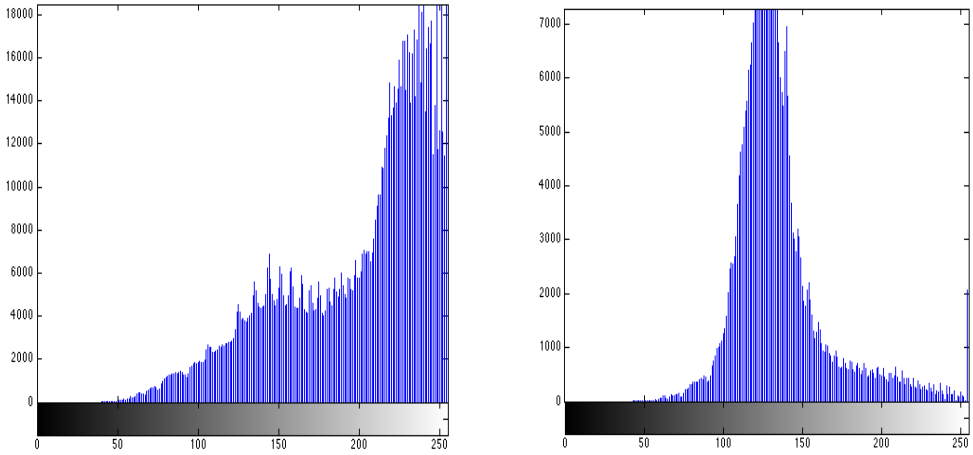
Figura 6.11 – Imagens dos Tecidos (a) Adiposo e (b) Epitelial.

[9] http://professor.ufabc.edu.br/~fzampirolli/cells)
[10] http://sibgrapi.sid.inpe.br/col/sid.inpe.br/sibgrapi/2010/08.28.15.30/doc/article_sibgrapi_v8.pdf.



(a) (b)

Figura 6.12 – Imagens em Níveis de Cinza dos Tecidos (a) Adiposo e (b) Epitelial.



(a) (b)

Figura 6.13 – Histogramas dos Tecidos (a) Adiposo e (b) Epitelial, das Imagens da Figura 6.12.

O código para ler as 10 imagens dos tecidos adiposo e epitelial é apresentado a seguir. Este código cria o arquivo "_MEASURES.arff", que pode ser lido pelo Weka.

```
function script_histograma

pasta = ['IMAGES/' ]; % pasta onde estão as imagens

d = dir(pasta); % comando para ler todos os arquivos de pasta
num=0;
for i = 3 : size(d,1) % para cada arquivo da pasta com final ".tif"
    str = [pasta d(i).name]
    if strcmp(str(end-3:end),'.tif')
        num = num + 1;
        addARFF(str); % função para adicionar o histograma em ".arff"
    end
end
disp(num)
end


% função para adicionar o histograma no arquivo "_MEASURES.arff"
function addARFF(str)
img = imread(str); % ler imagem ".tif" do disco

[cont,I] = imhist(img(:,:,2)); % calcular o histograma da imagem

str_meas = ['_MEASURES' '.arff']; % arquivo "_MEASURES.arff"
if fopen(str_meas,'r')==-1 % arquivo não existe, então crie cabeçalho
    fid = fopen(str_meas,'at+');
    fprintf(fid,'%s\n','@RELATION corpo');
    fprintf(fid,'%s\n','@ATTRIBUTE class    {epit,adip}');
    for Gi = 1 :  256
        type = num2str(Gi);
        fprintf(fid,'%s\n',['@ATTRIBUTE nc' type '  REAL']);
    end

    fprintf(fid,'%s\n','@DATA');
else % arquivo arff já existe, então inserir histograma da imagem
    fid = fopen(str_meas,'at+');
end

% adicionar em "_MEASURES.arff" "adip" ou "epit"
i=length(str);
while (i>0 && ~isequal(str(i),'/'))
 i = i - 1;
end
fprintf(fid,'%5s, ',str(i+1:i+4));

% adicionar em "_MEASURES.arff" os 256 valores do histograma da imagem
for Gi = 1 :  256
    if Gi == 256
        fprintf(fid,'%5d', cont(Gi));
    else
        fprintf(fid,'%5d, ', cont(Gi));
    end
end

fprintf(fid,'\n');
fclose(fid); % fechar arquivo
end
```

Classificação de Imagens Usando Atributos Topológicos

Nem todas as aplicações de classificação em imagens considerando a abordagem de histograma é possível. Por exemplo, nos histogramas das imagens feliz e triste da primeira seção deste capítulo, o número de pixels com valor 1 em ambas imagens é 7, e, com valor 0, 42. Assim, não é possível usar histogramas para classificar estas imagens.

Além disso, nas imagens dos tecidos adiposo e epitelial, apresentadas na seção anterior, a facilidade na classificação usando apenas histogramas não deve ocorrer se o objetivo for analisar a topologia das células em um único tipo de tecido celular, por exemplo, para tentar prever o início de um tumor. Mais ainda, analisar a topologia celular pode auxiliar no diagnóstico do câncer, para prever se um tumor é benigno ou maligno.

Desta forma, esta seção apresenta um estudo sobre classificação em imagens considerando os atributos relacionados à topologia de objetos. Como estudo de caso, serão consideradas as mesmas 10 imagens apresentadas na seção anterior, 5 do tecido adiposo e 5 do tecido epitelial. A parte de segmentação de imagens e cálculo dos atributos de cada objeto é avançada, porém o arquivo ".arff" contendo estes dados está disponível para o leitor poder reproduzir a classificação destas 10 imagens (http://professor.ufabc.edu.br/~fzampirolli/cells).

Segmentação

Uma das atividades mais importantes e complexas em visão computacional é encontrar, de forma automática, objetos em imagens, tendo como saída uma imagem em que cada objeto apresenta uma cor distinta, de forma a diferenciar um objeto de outro. Este processo pode ser chamado de segmentação em imagens.

A segmentação nas 10 imagens dos tecidos adiposo e epitelial foram obtidas através de um processo semiautomático: em primeiro lugar, algumas transformações em imagens foram aplicadas e possíveis marcadores foram identificados de forma automática; a seguir, um especialista usou uma interface para editar os marcadores, além de incluir e excluir certos marcadores. Todo este processo é detalhado em (ZAMPIROLLI *et al.*, 2010). Após a criação de um marcador para cada célula, foi usada uma transformação chamada *Watershed*,

para segmentar o contorno de cada célula, como apresentado na Figura 6.14. Para cada imagem, foram consideradas 81 células, totalizando 810 células.

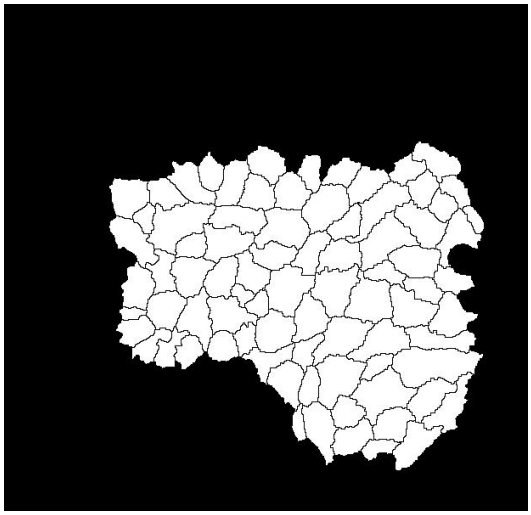


Figura 6.14 - Segmentação das Células da Figura 6.12(b).

Cálculo dos Atributos

Após a segmentação de cada imagem, é possível calcular vários atributos (como área, perímetro, etc.) para cada célula da imagem, usando por exemplo o comando ***regionprops*** (para mais detalhes, consulte o *help* do MATLAB®). Outras 5 medidas foram propostas por CHANG (2005). Também foram consideradas medidas em grafos, veja um exemplo de grafo na Figura 6.15. Consulte (ZAMPIROLLI *et al.*, 2010) para detalhes.

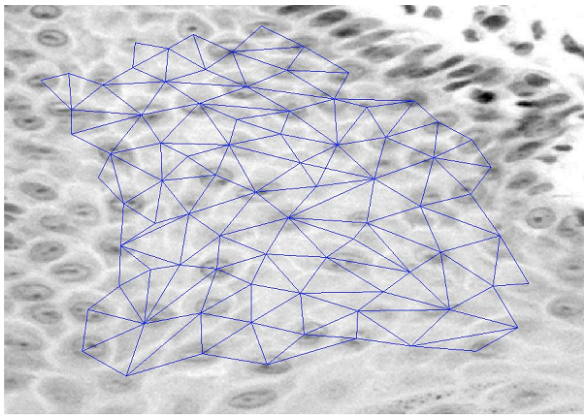


Figura 6.15 - Grafo de Vizinhança Gerado a partir da Imagem Segmentada.

Classificação

Após os cálculos dos vários atributos topológicos dos tecidos, foi gerado o arquivo ".arff", obtendo 99.8765% de acerto na classificação, com a seguinte matriz de confusão (em http://professor.ufabc.edu.br/~fzampirolli/cells, fazer *download* do aquivo ".arff", e seguir os seguintes passos no Weka: abrir arquivo ".arff" (*Explorer -> open file*) -> *Classify -> Choose -> function -> SMO -> (Nom) class -> Start*):

```
  a     b   <-- classified
404   1 |   a = adipose
  0 405 |   b = epitelial
```

A seguir são apresentados os pesos de cada atributo usando o SVM:

```
      -0.9332 * (normalized) Perimeter
+     -0.7119 * (normalized) Area
+     -4.2324 * (normalized) MeanDistNeighbors
+     -1.0041 * (normalized) MajorAxisLength
+     -1.054  * (normalized) MinorAxisLength
+      0.3752 * (normalized) Orientation
+     -0.8637 * (normalized) ConvexArea
+      0.002  * (normalized) Eccentricity
+     -0.9794 * (normalized) EquivDiameter
+      0.4476 * (normalized) Extent
+      1.4789 * (normalized) Solidity
+     -0.1844 * (normalized) FormFactor
+     -0.1537 * (normalized) Roundness
+      0.1897 * (normalized) AspectRation
+     -0.115  * (normalized) Convexity
+      4.0835 * (normalized) Solidity2
+      1.1104
```

Considerações Finais

O uso de aprendizado de máquina para classificar imagens é de grande importância na área de processamento digital de imagens porque geralmente produz resultados muito bons. Esta área está em crescimento devido ao avanço da tecnologia digital, demandando mais aplicações específicas. Este capítulo introduziu o uso de SVM aplicado no Reconhecimento de Padrões em imagens.

Existem outros classificadores, como Redes Neurais, KNN, etc., que podem ser usados para desenvolver sistemas especialistas usando imagens.

Além disso, é possível fazer o pré-processamento e a classificação usando apenas o MATLAB® , sem a necessidade de usar o Weka. Outra possibilidade é usar apenas linguagens de código aberto, como a linguagem Python, ou o Octave, no lugar do MATLAB® .

Para trabalhos futuros podemos pesquisar novos atributos, além dos obtidos do histograma e de atributos geométricos de objetos segmentados na imagem, criando novas aplicações em classificação em imagens.

Lista de Exercícios

1. Reproduzir os exemplos das faces feliz e triste da primeira seção com translações aleatórias de 1, 2 e 3 pixels para a horizontal e para a vertical, para 5, 10, 50 e 100 amostras, usando o código da função `imagens_aleatorias`. Esta função também gera o arquivo ".arff" para ser usado no Weka. Verifique se as classificações coincidem com as apresentadas neste capítulo. Altere esta função, agora para gerar ruídos aleatórios e também verifique a classificação no Weka.

2. Reproduzir os exemplos de classificação usando o histograma dos tecidos epitelial e adiposo. Para isto use a função `script_histograma`, apresentado na segunda seção. Construa a sua própria base de imagens com pelo menos dois tipos distintos de histogramas e use esta função para verificar a sua classificação (por exemplo, fotos digitais de pessoas e de paisagens).

3. Além dos atributos apresentados na terceira seção para analisar a topologia dos tecidos epitelial e adiposo, sugerir mais outros atributos para tentar melhorar a classificação.

4. Sugerir outras aplicações de classificação de imagens e novas técnicas usadas para gerar os atributos.

Referência Bibliográfica

CHANG, R.; WU, W.; MOON, W. K. & CHEN. D. **Automatic Ultrasound Segmentation and Morphology Based Diagnosis of Solid Breast Tumors**. Breast Cancer Research and Treatment, 89:179–18, 2005.

WOODS, R. E. & GONZALEZ, R. C. **Processamento Digital De Imagens** - 3ª Ed. Pearson Education – Br – 2011.

ZAMPIROLLI, F. A.; STRANSKY, B.; LORENA, A. C. & PAULON, F. L. D. M.: **Segmentation and Classification of Histological Images - Application of Graph Analysis and Machine Learning Methods.** In SIBGRAPI(2010)331-338.

Equipamento promete melhorar diagnóstico do câncer de pele. In http://noticias.bol.uol.com.br/ultimas-noticias/ciencia/2013/08/07/equipamento-promete-melhorar-diagnostico-do-cancer-de-pele.htm. Acessado em 08.08.13.

Hospital de Ribeirão Preto usa sistema nacional para arquivar e gerenciar imagens médicas. In http://revistapesquisa.fapesp.br/2013/07/12/acesso-digital. Acessado em 08.08.13.

Mineração de dados por imagens auxilia diagnóstico médico. In http://www.usp.br/agen/?p=100292. Acessado em 08.08.13.

MATLAB®. The Language of Technical Computing. In http://www.mathworks.com/products/matlab. Acessado em 15.08.13.

Weka. The Waikato University. In http://www.cs.waikato.ac.nz/ml/weka. Acessado em 03.03.13.