Universidade do Minho
Departamento de Informática
Mestrado Integrado em Engenharia Informática

TPC6

Segmentação automática de vasos da retina

Visão por Computador

Grupo de trabalho:

Ana Esmeralda Fernandes, A74321
Miguel Dias Miranda, A74726

Braga, 31 de Janeiro de 2018

Conteúdo

1. Meta 1

Com o propósito de utilizar máquinas de vetor de suporte para o processo de segmentação dos vasos da retina, é necessário fornecer a estes métodos de classificação dados de treino, para que os mesmos sejam utilizados no seu processo de aprendizagem supervisionada.

Seguindo a abordagem da técnica de *Ricci* e *Perfetti* [1], as variáveis de entrada que alimentam o processo de análise de padrões da *SVM* serão *samples* de pixeis retiradas das matrizes Img_{inv}, S e S_0 de um conjunto de imagens de treino. O significado destas matrizes e a sua forma de cálculo podem ser encontradas com mais detalhe no referido artigo.

1.1 Cálculo das características I_{inv}, S e S_0

Nesta secção, serão abordadas as principais decisões no processo de criação das referidas características, sendo para isso utilizada a linguagem de programação *python*. Nesta implementação, procurou-se tirar proveito dos métodos e operações que o *python* permite realizar, de forma eficiente e simples, sobre matrizes.

De forma muito breve, o cálculo das matrizes S e S0 obriga à determinação, para cada pixel da imagem, de um conjunto de 12 retas. Cada uma destas retas está inserida numa janela 15x15 e encontram-se espaçadas entre si em ângulos de 15. Para cada uma destas retas, serão calculados um conjunto de 15 pontos.

1.1.1 Descrição do algoritmo

1. Cálculo da matriz Img_invertida, utilizando o método *np.invert*;
2. Inicialização com zeros das matrizes S e S0, com as mesmas dimensões da imagem inicial;

3. Calcular previamente os valores dos pontos ocupados pelas 12 retas, dentro de uma janela 15x15, assumindo o centro do referencial como o centro da janela.

   A função criada para este efeito, designada *gera_Pontos_Retas* devolve duas matrizes 12x15. Cada linha representa uma das 12 retas e cada coluna representa, conforme a matriz, o valor de um ponto no eixo das abcissas (eixo X) ou no eixo das ordenadas (eixo Y).

4. Como o cálculo das retas só traz informação relevante quando aplicadas dentro da zona da retina, como forma de otimização, o algoritmo pode apenas operar na área da imagem ocupada da retina. Para isso utiliza-se uma máscara da retina da imagem a processar.

   A máscara pode ser obtida com recurso às técnicas implementadas no TPC5 ou através das máscaras disponibilizadas na pasta *DRIVE*;

5. Iterar a imagem. Como otimização, iterar apenas os índices da imagem que pertencem à zona da retina.

```
id_l, id_c = nonzero(mask_img_inv > 0)
for i,j in zip(id_l, id_c):
    // to do
```

6. Para cada pixel iterado:

   - Verificar se o pixel se encontra próximo da borda da imagem, ou seja, se a janela 15x15 com esse pixel no centro apanha pontos fora da área da retina;
   - Caso se verifique esta condição, a média N da janela é calculada com base apenas na média dos pixeis da janela que estão dentro da área da retina;
   - O valor de intensidade dos pixeis da janela 15x15 que ficam fora da área da retina, devem passar a ter o valor de N.

7. Calcular o valor de N, que representa a média de intensidades da janela 15x15 a utilizar;

8. Calcular a média das intensidades para cada linha, somando as intensidades dos pontos da janela 15x15 que definem essa reta L e dividindo pelo número de pontos contabilizados;

9. Tendo as médias para cada uma das 12 retas L, a posição (i,j) da matriz S toma o valor L_{max} - N, onde N representa a média da janela anteriormente calculada e L_{max} o valor máximo das intensidades médias calculadas para cada reta L.

10. Para a reta que apresenta a maior média de intensidades, calcula-se a sua reta perpendicular. Para esta reta, designada por L0, calcula-se também a sua média de intensidades, mas utilizando apenas os seus 3 pontos centrais. O valor da matriz S0 na posição (i,j) toma o valor L0 - N.

11. Como as imagens utilizadas para os dois processos de segmentação são conhecidas e constantes nesta exploração do problema, cada característica gerada é guardada num ficheiro no formato *.gif*. Assim, dentro da diretoria de saída fornecida, serão criadas 4 pastas:

    - Pasta *S_img/* que contem as imagens S geradas. Cada imagem é guardada usando a numeração da *DRIVE*, por exemplo "23_S.gif"ou "03_S.gif";
    - Pasta *S0_img/* que contem as imagens S0 geradas. Cada imagem é guardada usando a numeração da *DRIVE*, por exemplo "34_S0.gif"ou "08_S0.gif";

- Pasta *I_Inv/* que contem as imagens I_inv geradas. Cada imagem é guardada usando a numeração da *DRIVE*, por exemplo "12_I_Inv.gif";
- Pasta *Vasos/* onde são copiadas as imagens dos vasos da retina existentes na *DRIVE*. Estas imagens serão necessárias para o processo de segmentação por *SVMs*, na construção da matriz target, como forma de saber se os índices são relativos a pixeis dos vasos ou do fundo da retina.

Ao guardar estas características em ficheiros, evita-se ter que processar novamente todas as imagens, sempre que se procurarem explorar os parâmetros das técnicas de segmentação. (Figura 1.1)

Este algoritmo é codificado na função *gere_caracteristicas* que se encontra devidamente explicada com mais detalhe no módulo *Meta1.py* que a contêm.

1.1.2 Análise de Resultados

Para executar a geração das características S, S_0 e I_{inv}, deve utilizar-se um comando com a estrutura: **python TPC6.py** "./**diretoria_input**/" "./**diretoria_output**/" **1**, supondo que a linha de comandos se encontra na pasta com acesso às diretorias e aos módulos *python* desenvolvidos.

- A opção (1) indica que é para gerar as caracteristicas;
- A diretoria *input* deve ter dentro dela as pastas *test* e *training*. As pastas *test* e *training* devem obedecer às sub pastas e nomes de ficheiros da base de dados DRIVE;
- A diretoria *output* pode ser uma qualquer. Dentro dela serão criadas as pastas Treino e Teste e, dentro de cada uma destas, são criadas as sub-pastas e ficheiros descritos no último tópico da secção 1.1.1.

Exemplo: **python TPC6.py** "./**DRIVE**/" "./**Caracteristicas**/" **1**

Este comando processa as 20 imagens da pasta *training* e as 20 imagens da pasta *test*. Para cada imagem, são geradas as características I_{inv}, **S**, **S0**. É ainda calculada e guardada uma matriz 1000x2, com as coordenadas (i,j) dos 1000 índices aleatoriamente selecionados dentro da área da retina para serem utilizados como dados para a *SVM*. Cada imagem terá um conjunto distinto de 1000 pontos a utilizar para treino ou teste da *SVM*, conforme o seu tipo.

O tempo de processamento sequencial ronda os 65 segundos por imagem, demorando assim cerca de 23 minutos para processar todas as 20 imagens de treino e outros 23 minutos para processar todas as 20 imagens de teste. O tempo de processamento **sequencial** das 40 imagens foi de **43 minutos**.

Numa tentativa de explorar o paralelismo de *threads* do CPU, foi criada uma versão da função de geração que lança um grupo de 20 threads para tratar paralelamente das 20 imagens de treino e, posteriormente ao término destas, executa outro grupo de 20 *threads* para processar as 20 imagens de treino.

- É deixado a cargo do sistema operativo a melhor gestão das threads e particionamento dos recursos do CPU;
- De realçar que esta alternativa aumenta o tempo de processamento por imagem para os 10 minutos, mas ao fim de 10 minutos estão preparadas 20 imagens simultaneamente.
- Esta técnica implica ainda que durante a gestão e execução das *threads* o CPU é utilizado de forma intensiva nos 100%.
- Nesta alteração, o tempo de processamento **paralelo** das 40 imagens foi de **21 minutos**.

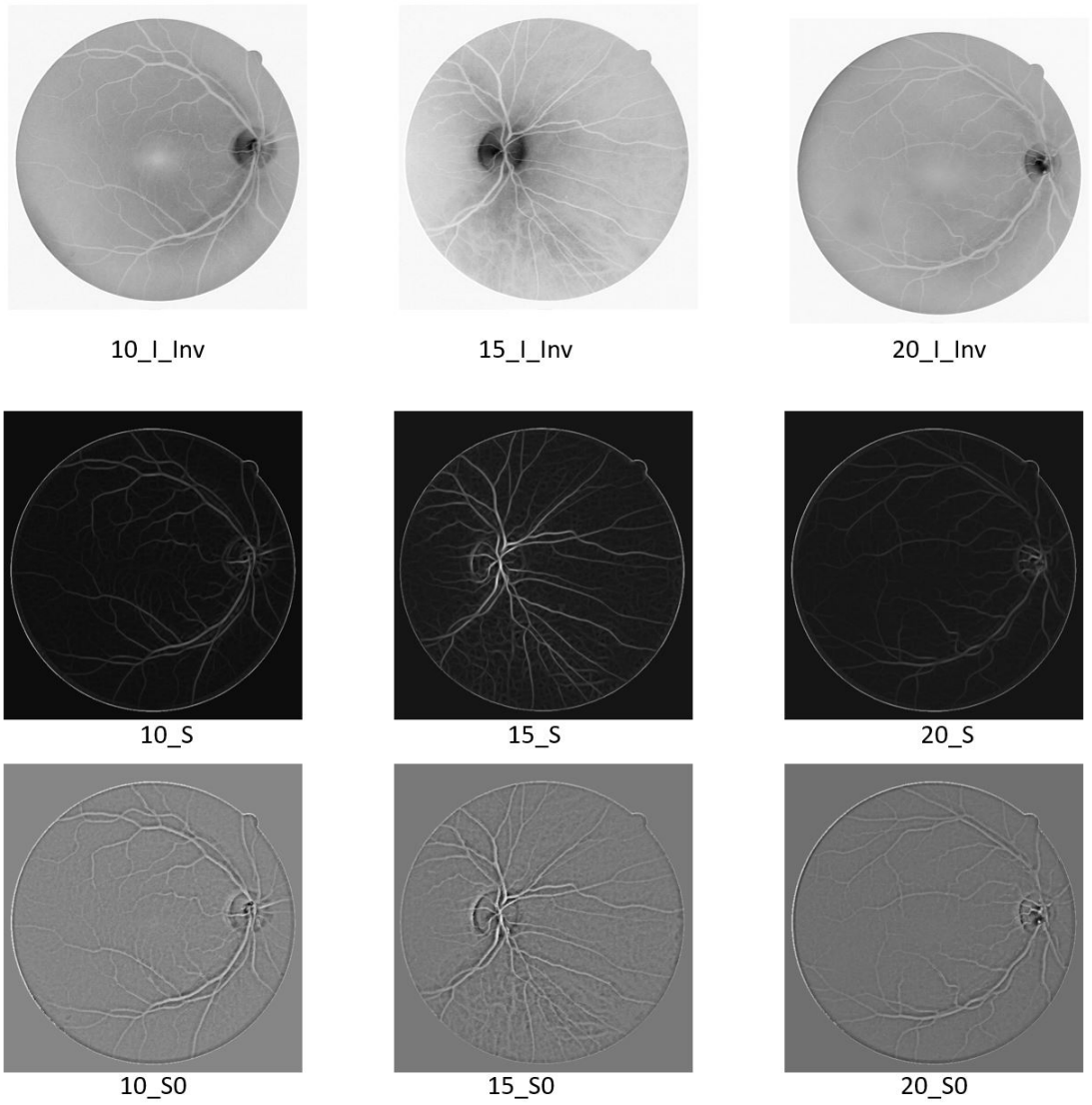

Figura 1.1: Exemplos das características de três imagens da pasta *DRIVE/test*

1.2 Segmentação baseada no *threshold* em S

Para realizar segmentação baseada nas imagens S, deve utilizar-se um comando com a estrutura: **python TPC6.py** "./**diretoria_input**/" "./**diretoria_output**/" **2**, supondo que a linha de comandos se encontra na pasta com acesso às diretorias e aos módulos *python* desenvolvidos.

- A opção (2) indica que é para segmentar segundo a característica S previamente calculada;
- diretoria *input* deve ter dentro dela a pasta com as imagens S, usando os nomes das imagens geradas pelo algoritmo da secção anterior;
- A diretoria *output* indica a pasta onde serão guardadas as imagens geradas pela segmentação.

Exemplo: **python TPC6.py** "./**Caracteristicas/Treino/S_img**/" "./**Segmentacao_S**/" **2**

Como o nome indica, o processo de segmentação de vasos baseado num *threshold* em S, utiliza a imagem S criada para cada imagem. Sobre essa imagem, é realiza uma binarização, segundo um valor de intensidade de corte determinado pelo algoritmo de *Otsu*.

Apesar de no artigo de *Ricci* [1] o processo desta segmentação obter resultados satisfatórios, na implementação desenvolvida os resultados apresentam fraco rigor. Apesar de se terem explorado diferentes formas de calcular o *threshold* da imagem S, os resultados finais continuam satisfatórios.

Contudo, deve ser realçado o facto de que a implementação feita segue a metodologia especificada pelo artigo.

2. Meta 2

2.1 Segmentação baseada em *Support Vector Machine*

Na secção anterior foi descrito o processo de calculo das imagens Img_Inv, S e S0. Estas matrizes são previamente calculadas para cada uma das imagens a utilizar no processo de treino e teste, e guardadas num respetivo ficheiro *.gif*.

A utilização das informações destas características, para o processo de aprendizagem e teste da *SVM*, baseia-se nas seguintes considerações:

- As 20 imagens utilizadas para o processo de treino, encontram-se na base de dados *DRIVE*, dentro da pasta *training*. De forma semelhante, as 20 imagens utilizadas para testar os parâmetros da *SVM*, encontram-se na base de dados *DRIVE*, dentro da pasta *test*.

  Sobre estas 40 imagens, foram previamente calculadas as características S, S0 e I_inv, segundo o algoritmo descrito na secção 1.1.1 anterior.

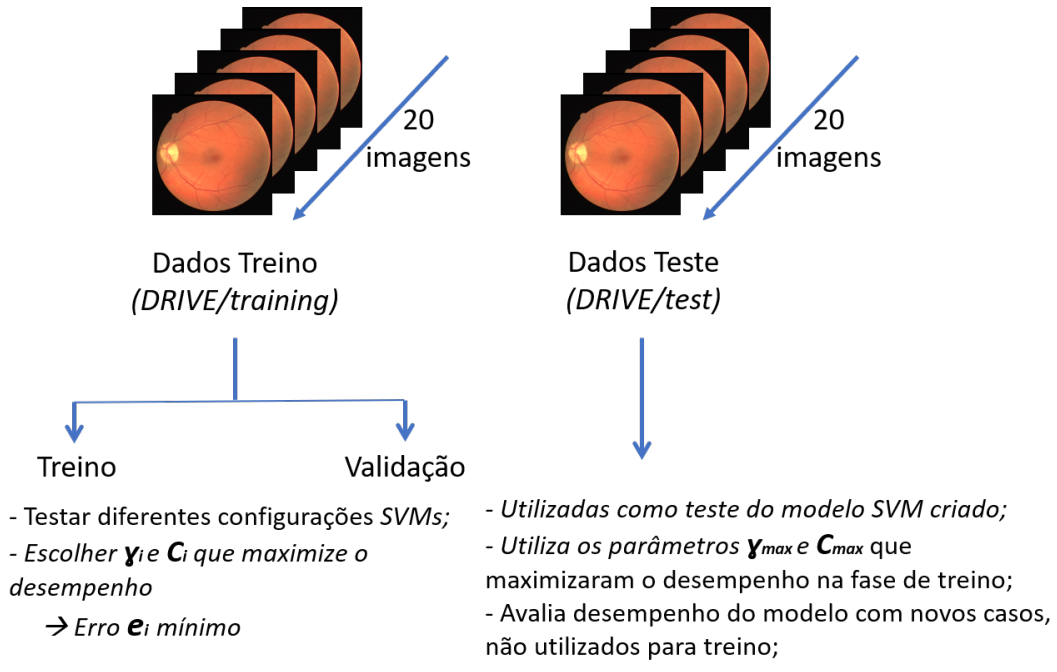

Figura 2.1: Divisão das imagens entre conjuntos de treino e teste

- Para cada imagem, e com recurso à mascara da retina de cada uma delas, são recolhidos 1000 índices (i,j) de forma aleatória.

  Ao utilizar a máscara da respetiva imagem, garante-se que os pixeis escolhidos, encontram-se dentro da área da retina, sendo assim fundo ou vaso da retina, mas nunca *background* externo à área da retina.

- A matriz *features* representa os atributos que a *SVM* irá analisar para estimar uma classificação. As suas colunas são referentes a valores de S, S0 e I_inv, respetivamente, seguindo assim a ordem especificada no artigo.

- Cada matriz *features* tem associada uma matriz *target*. A matriz *target* indica, para cada linha da matriz *features* qual é classificação esperada, considerando os valores dos atributos S, S0 e Img_Invertida dessa linha.

  Se o ponto utilizado para preencher uma linha da matriz *features* for retirado de um vaso da retina, então a respetiva linha na matriz *target* toma o valor 1. Em oposição, se o ponto utilizado pertencer ao fundo da retina, a respetiva linha da matriz *target* toma o valor 0.

- Cada coluna da matriz *features* deve ser normalizada, utilizando a sua média e desvio padrão, segundo a formula:

$$x = \frac{x_i - \mu_i}{\sigma_i},$$

  sendo μ_i = média coluna, σ_i = desvio padrão coluna, i = 1,2,3

  Esta normalização deve ser realizada separadamente para cada imagem (conjunto de 1000 pixeis) de forma a compensar variações de iluminação e contraste entre cada imagem.

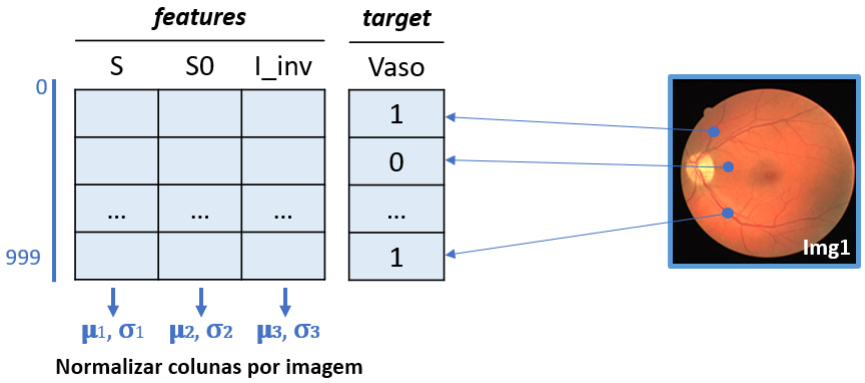

Figura 2.2: Normalização dos índices selecionadas por imagem, recorrendo à média e desvio padrão por coluna

- Os dados devem ser "baralhados"aleatoriamente, para evitar tendências de aprendizagem da *SVM*.

  Para isso, rearranja-se aleatoriamente as matrizes *features* e *target* pela mesma ordem, para manter o mapeamento correto entre os casos de treino e o resultado esperado. Ou seja, se a linha i trocar pela linha j na matriz *features*, a mesma troca deve acontecer na matriz *target*.

A seguinte imagem procura ilustrar este processo esquematicamente.

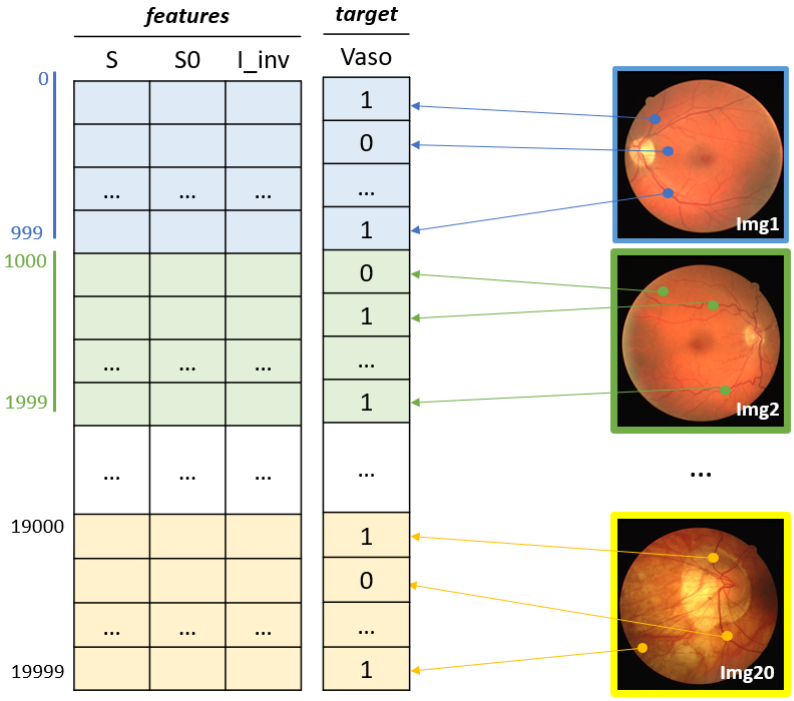

Figura 2.3: Preparação dos atributos para treinar a *SVM*

2.1.1 Análise de Resultados

Para executar as diferentes configurações de *SVMs*, utilizando as informações das características S, S0 e I_{inv}, deve utilizar-se um comando com a estrutura: **python TPC6.py** "./**diretoria_input**/" **3**, supondo que a linha de comandos se encontra na pasta com acesso às diretorias e aos módulos *python* desenvolvidos.

- A opção (3) indica que é para realizar a aprendizagem e teste das diferentes *SVMs*;
- A diretoria *input* deve conter dentro dela as pastas *Treino* e *Teste* geradas pelos cálculos das caraterísticas S, S0 e I_{inv}. (Último tópico da secção 1.1.1);

Exemplo: **python TPC6.py** "./**Caracteristicas**/" **3**

Para efeitos de aprendizagem e avaliação dos desempenhos de classificação, cada técnica de *SVM* utilizada recebe 4 argumentos: As matrizes *features* e *target* que representam os 20000 exemplos de treino, gerados pelas características da 20 imagens de treino, e as matrizes *features* e *target* que representam os 20000 exemplos de teste, gerados pelas características da 20 imagens de teste.

Foram utilizadas as seguintes configurações:

- **SVM Linear**

  Taxa de sucesso em treino de 87.275% e taxa de sucesso em teste de 87.03%

- **SVM SVC Linear** (*kernel SVC linear*)

  Taxa de sucesso em treino de 88.09% e taxa de sucesso em teste de 88.005%

- **SVM RBF** - *tuning* hiperparâmetros

  Utiliza apenas as 20 imagens de treino (Figura 2.1), que divide em sub conjuntos de treino e teste da SVM.

Usada para estimar os melhores valores de *C* e *Gamma*, conforme os melhores resultados obtidos na previsão de treino.

Obtidos os melhores valores, a SVM RBF é novamente utilizada com os melhores hiperplanos *C* e *Gamma*, mas utilizando as 40 imagens da DRIVE.

Nesta abordagem, os melhores valores para os parâmetros foram *C*=8.6 e *Gamma*=0.1, com taxa de sucesso em treino de 93.25% e taxa de sucesso em teste de 93.29%.

- **SVM RBF** - com todas as 40 imagens Utilizando os melhores valores para os parâmetros *C* e *Gamma*, esta SVM foi ensinada com as 20 imagens de treino e testada com as 20 imagens de teste.

  Nesta abordagem, taxa de sucesso obtida em treino foi de 93.21% e taxa de sucesso em teste de 93.36%.

Sendo que o uso de SVMs lineares exige um tempo de execução muito menor que as abordagens não lineares, este tipo de *SVM* é suficiente para o processo de classificação e devem ser utilizadas na segmentação de vasos da retina, com recurso a técnicas de aprendizagem supervisionada. Estas mesmas conclusões foram observadas por *Ricci* [1] no final da secção *II* do seu artigo.

Bibliografia

[1] Elisa Ricci and Renzo Perfetti. Retinal blood vessel segmentation using line operators and support vector classification. *IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 26*, 10 October 2007.