Introdução ao Processamento de Imagem Digital (MO443/MC920)

Prof. Alexandre Xavier Falcão

Segundo semestre de 2003

1 Segmentação por clustering

Técnicas de clustering têm sido usadas em problemas de reconhecimento de padrões de diversas áreas. Estamos interessados na aplicação dessas técnicas para segmentação de imagens. Neste caso, cada pixel $p \in D_I$ de uma imagem $\hat{I} = (D_I, I)$ tem associado um vetor de características de imagem, tais como brilho, variância e magnitude de gradiente [1]. Para um número n de características, cada pixel p é mapeado em um ponto no espaço $\Re^n$ de características. Técnicas de clustering se baseiam na premissa que regiões de interesse para segmentação formam aglomerados de pontos separáveis em $\Re^n$, e procuram identificar esses aglomerados particionando o espaço $\Re^n$ em k regiões $R_1, R_2, \ldots, R_k$ tais que $\cap_{i=1}^{k} R_i = \emptyset$ e $\cup_{i=1}^{k} R_i = D_I$. Se cada região R_i estiver associada a um único objeto O_i, $i = 1, 2, \ldots, k$, incluindo o fundo, então a partição obtida é a solução da segmentação. Caso contrário, quando um objeto possui múltiplas regiões de pixels com características distintas, temos ainda que associar quais regiões pertencem a quais objetos. Informações globais sobre o problema— tais como características das regiões, vizinhança entre regiões, e a forma gerada pela união de regiões— poderiam ser usadas para completar a segmentação.

O particionamento em regiões requer o cálculo de distância entre pontos do $\Re^n$. Esta distância pode ser traduzida em uma dissimilaridade $d(p, q)$ entre os pixels $p, q \in D_I$ que esses pontos representam. Por exemplo podemos ter em $\Re^2$,

$$d(p, q) = [I(p) - I(q)]^2 + [G(p) - G(q)]^2, \tag{1}$$

onde $G(p)$ é a magnitude do gradiente de Sobel calculado em p. Existem várias funções de dissimilaridade, mas elas normalmente satisfazem os axiomas métricos.

1. $d(p, q) \geq 0$, e se $d(p, q) = 0$, então $p = q$.
2. $d(p, q) = d(q, p)$.

[1] No caso de imagens coloridas, temos ainda propriedades relacionadas com os componentes de cor que podem ser usados como características.

3. $d(p,r) \leq d(p,q) + d(q,r)$.

As regiões R_i são definidas dinaminamente durante o algoritmo de clustering. Isto permite que a dissimilaridade entre dois pixels varie durante o algoritmo e seja uma medida mais global do que local. Alguns variantes definem $d(p,q)$ como a dissimilaridade entre regiões R_i e R_j, onde $1 \leq i,j \leq k$, $i \neq j$, $p \in R_i$ e $q \in R_j$, levando-se em conta as características de todos os pixels que pertencem a essas regiões no dado instante.

Os métodos de clustering mais comuns podem ser divididos em particionais e hierárquicos. Métodos particionais definem um valor fixo para k e maximizam um critério de dissimilaridade entre regiões, enquanto que métodos hierárquicos consideram todas as possibilidades para $1 \leq k \leq |D_I|$. Métodos hierárquicos aglomerativos partem da partição onde cada pixel forma uma região ($k = |D_I|$) para o caso onde todos os pixels formam uma mesma região ($k = 1$), e os métodos divisivos fazem o caminho inverso.

1.1 Clustering por partição

Métodos particionais iniciam com um conjunto $S = \{p_1, p_2, \ldots, p_k\}$ de pixels representativos de cada região R_i, $p_i \in R_i$, $i = 1, 2, \ldots, k$. Esses pixels podem ser eleitos com base em algum critério automático de escolha ou selecionados manualmente pelo usuário. Uma partição é obtida por associar cada pixel $p \in D_I \backslash S$ à região cujo representante é o mais próximo em S. Após particionar a imagem, novos representativos são eleitos com base em uma função critério f aplicada a cada região. Se os representativos eleitos por f forem diferentes dos que estão em S, o algoritmo substitui esses representativos em S e repete o particionamento. O algoritmo pára quando os pixels em S são reeleitos novamente por f. Um algoritmo exemplo é dado abaixo.

Clustering por partição:

Entrada: Imagem cinza $\hat{I} = (D_I, I)$, número de regiões k, conjunto $S = \{p_1, p_2, \ldots, p_k\}$ de pixels representativos, função de dissimilaridade d, e uma função critério f.
Saída: Imagem rotulada $\hat{L} = (D_I, L)$, onde $L(p) = p_i$ para um representativo $p_i \in R_i$, $i = 1, 2, \ldots, k$.
Auxiliares: Conjunto S' e conjuntos de regiões $\{R_1, R_2, \ldots, R_k\}$.

1. $S' \leftarrow \emptyset$ e $R_i \leftarrow \emptyset$ para $i = 1, 2, \ldots, k$.
2. Para todo pixel $p \in D_I$, faça
3. Se $p \in S$ então $L(p) \leftarrow p$. Caso contrário, $L(p) \leftarrow nil$.
4. Para todo pixel $p \in D_I$, tal que $L(p) = nil$,
5. Encontre $p_i \in S$ tal que $d(p, p_i) = \min_{\forall q \in S}\{d(p,q)\}$.
6. Faça $L(p) \leftarrow p_i$.

7. Para cada pixel $p_i \in S$ associe um único conjunto R_i, $i = 1, 2, \ldots, k$.

8. Para todo pixel $p \in D_I$,

9. Encontre $p_i \in S$ tal que $L(p) = p_i$ e insira p em R_i.

10. Para $i = 1, 2, \ldots, k$ eleja um novo representativo $\hat{p}_i \in R_i$ baseado no critério $f(R_i)$ aplicado a todos os pixels em R_i, e insira $\hat{p}_i$ em S'.

11. Se $S' = S$, então páre. Caso contrário, faça $S \leftarrow S'$ e volte para a etapa 1.

O único aspecto não esclarecido no algoritmo acima é a função critério f usada na linha 10. O algoritmo *k-medoids*, por exemplo, usa como representativo de cada região R_i, o pixel $\hat{p}_i \in R_i$ que minimiza a distância média entre ele e os demais em R_i. Isto é,

$$f(R_i) \;=\; \frac{1}{|R_i|} \sum_{\forall q \in R_i} d(\hat{p}_i, q) = \min_{\forall p \in R_i} \left\{ \frac{1}{|R_i|} \sum_{\forall q \in R_i} d(p, q) \right\}. \tag{2}$$

Já o algoritmo *k-means* escolhe como representativo o pixel que minimiza a distância quadrática média entre ele e os demais pixels em R_i. Neste caso, o pixel $\hat{p}_i \in R_i$ escolhido é o mais próximo do centróide de R_i. Observe que as funções critério usadas nesses dois algoritmos tendem a encontrar aglomerados hiperesféricos e compactos em $\Re^n$. Elas não se aplicam a aglomerados alongados por exemplo.

1.2 Clustering hierárquico

Métodos hierárquicos são normalmente mais eficientes do que os particionais, porém não revertem decisões tomadas durante a construção da hierarquia. Por exemplo, uma vez decidido que dois pixels pertencem a uma mesma região em um nível da hierarquia durante um algoritmo aglomerativo, eles pertencerão a esta mesma região deste nível em diante.

Métodos aglomerativos partem da situação onde cada pixel forma uma região R_i, $i = 1, 2, \ldots, k$ e $k = |D_I|$. A cada passo, avaliam uma função de dissimilaridade $d(R_i, R_j)$ para todo par (R_i, R_j), $i \neq j$, $i, j = 1, 2, \ldots, k$, e unem os pares de regiões com **menor dissimilaridade**, reduzindo o número k de regiões. Este processo guarda a hierarquia entre as regiões e é repetido até $k = 1$.

Métodos divisivos partem da situação onde todos os pixels formam uma única região. A cada passo, avaliam uma função de dissimilaridade $d(p, q)$ entre os pixels de cada região, e dividem as regiões separando os pixels com **maior dissimilaridade**, aumentando o número k de regiões. Este processo guarda a hierarquia entre as regiões e é repetido até $k = |D_I|$.

A premissa no clustering hierárquico é que a solução desejada estará em um nível da hierarquia. Muito embora os métodos divisivos tendam a obter regiões maiores que os aglomerativos, o custo computacional alto para fazer as primeiras divisões faz com que a maioria dos algoritmos publicados sejam aglomerativos.

No contexto de métodos aglomerativos, existem várias formas de definir a dissimilaridade $d(R_i, R_j)$ entre duas regiões como função da dissimilaridade entre seus pixels. Uma alternativa para obter aglomerados hiperesféricos e compactos é definir $d(R_i, R_j)$ como a média das

dissimilaridades entre todos os pares de pixels (p, q), $p \in R_i$ e $q \in R_j$. O caso mais popular é o algoritmo *single-linkage* que define $d(R_i, R_j)$ como a menor dissimilaridade entre um pixel $p \in R_i$ e um pixel $q \in R_j$. Este critério é adequado para obter aglomerados alongados. Já no caso de aglomerados compactos e hiperesféricos, mas não bem separados, a escolha de $d(R_i, R_j)$ como a maior dissimilaridade entre um pixel $p \in R_i$ e um pixel $q \in R_j$ é mais adequada. Este é o caso do algoritmo *complete-linkage*.

Em particular, o algoritmo *single-linkage* tem uma relação estreita com algoritmos que constroem uma árvore de peso mínimo para um grafo de entrada (e.g. Algoritmos de Prim e de Kruskal), onde no nosso caso, os vértices são os pixels e as arestas são definidas por uma relação de adjacência simétrica entre os pixels. Se o grafo for completo (i.e. todos os pixels da imagem são considerados adjacentes entre si), o algoritmo é dito sem restrição de conexidade. Alta eficiência pode ser obtida com restrição de conexidade para relações de adjacência pequenas (e.g. vizinhança-8). Uma vez construída a árvore de peso mínimo, a segmentação hierárquica é obtida removendo arestas da árvore com peso maior que um dado limiar T, para T variando do peso de aresta mínimo ao máximo. Isto é, cada solução possível é uma floresta de peso mínimo, onde as regiões R_i são árvores de peso mínimo. Esta hierarquia de regiões é conhecida como dendrograma. Se o valor de T for conhecido *a priori*, podemos modificar o algoritmo para gerar o resultado da segmentação em um mapa de rótulos. Um exemplo deste variante aplicado ao algoritmo de Prim é apresentado a seguir.

Segmentação de imagens como uma floresta de peso mínimo:

Entrada: Imagem cinza $\hat{I} = (D_I, I)$, função de dissimilaridade d, e um limiar T.
Saída: Imagem rotulada $\hat{L} = (D_I, L)$, onde $L(p) = p_i$ para um representativo $p_i \in R_i$, $i = 1, 2, \ldots, k$. Imagem de predecessores $\hat{P} = (D_I, P)$ representando a floresta de peso mínimo. Imagem de custos $\hat{C} = (D_I, C)$ onde $C(p)$ é o peso da aresta que liga o pixel p com seu predecessor $P(p)$.
Auxiliares: Fila de prioridades Q e variável c.

1. Para todo pixel $p \in D_I$, faça $C(p) \leftarrow +\infty$, $L(p) \leftarrow p$, $P(p) \leftarrow nil$, e insira p em Q.
2. Enquanto $Q \neq \emptyset$, faça
3. Remova p de Q tal que $C(p)$ é mínimo.
4. Se $P(p) = nil$ então $C(p) \leftarrow 0$.
5. Para todo $q \in A(p)$, tal que $q \in Q$, faça
6. $c \leftarrow d(p, q)$.
7. Se $c < C(q)$ e $c < T$, então
8. Remova q de Q, faça $C(q) \leftarrow c$, $P(q) \leftarrow p$, e $L(q) \leftarrow L(p)$, e insira q em Q.

Observe que o algoritmo acima é essencialmente o algoritmo de Dijkstra modificado para uma função de custo $f(\pi)$ não-suave, definida como o peso da última aresta no caminho π, ou 0 caso π seja trivial. Com memória adicional, poderíamos também modificar $d(p, q)$ para representar a dissimilaridade entre as regiões com raíz $L(p)$ e $L(q)$. Se removermos o teste $c < T$ da linha 7 teremos uma árvore de peso mínimo em P.

Uma solução parecida é usada por métodos que procuram reduzir a supersegmentação criada pela transformada clássica de watershed de uma imagem de gradiente através da união de bacias. Por exemplo, constrói-se um grafo de vizinhança entre as bacias, calcula-se uma árvore de peso mínimo para este grafo usando a dissimilaridade entre bacias vizinhas, e depois escolhe-se a floresta de peso mínimo que resolve a segmentação. A floresta pode resultar da limiarização do peso das arestas ou da escolha de sementes pelo usuário. Note que em uma árvore de peso mínimo só existe um caminho que liga dois nós. Se escolhermos uma semente s_1 com rótulo objeto, por exemplo, e outra s_2 com rótulo fundo, podemos partir a árvore em duas por remover a aresta de maior peso ao longo do caminho que liga s_1 a s_2. Nesta abordagem, porém, o mapa de predecessores P deve ser transformado em um grafo não orientado e conexo, onde existe um caminho entre qualquer par de nós.

2 Exercícios

1. Proponha e analise algumas funções de dissimilaridade e critério para o algoritmo de clustering por partição.

2. Proponha um algoritmo de clustering hierárquico usando a abordagem divisiva.

3. Proponha variações do algoritmo apresentado acima para segmentação como uma floresta de peso mínimo.