10장

그룹화 및 모델 피팅

이전 장에서는 다양한 클러스터링 방법과 유사성을 측정하는 다양한 방법을 사용하여 서로 "닮은" 픽셀을 함께 수집했습니다. 이 보기는 토큰(예: 포인트, 에지 포인트, 슈퍼픽셀)에 적용될 수 있습니다. 본질적으로 로컬 뷰입니다.

보다 포괄적인 관점을 강조하는 대안은 픽셀, 토큰 또는 일부 모델을 준수하기 때문에 무엇이든 함께 수집하는 것입니다. 이 접근 방식은 클러스터링 접근 방식과 다소 유사해 보이지만 메커니즘과 결과는 상당히 다른 경향이 있습니다. 클러스터링 접근 방식에서 우리가 생성하는 결과는 로컬 구조를 가질 수 있지만 반드시 글로벌 구조를 가질 필요는 없습니다.

예를 들어, 추가할 토큰이 앞의 두 토큰에 의해 형성된 라인에 가까운지 여부를 테스트하여 토큰을 집합적으로 라인으로 조립하려고 하면 꽤 구부러진 모양을 얻을 수 있습니다. 모든 토큰이 라인의 매개변수에 대해 "동의"하는지 확인해야 합니다. 로컬 일관성이 충분하지 않습니다.

이러한 문제는 다소 어려우며 한 종류의 모델에 대한 공격 전략은 다른 종류의 모델에 대한 전략으로 확장되는 경향이 있습니다. 이 장에서 우리는 주로 하나의 핵심 문제에 집중하는데, 이는 세부적으로 수행할 수 있을 만큼 간단합니다. 우리는 일련의 토큰으로 표시되는 모든 라인을 찾으려고 합니다. 이 문제는 일반적으로 피팅 또는 때때로 그룹화라고 합니다. 여기에는 세 가지 중요한 하위 문제가 있습니다. 모든 토큰이 모델에 동의하면 모델은 무엇입니까?

특정 모델에 기여하는 토큰과 그렇지 않은 토큰은 무엇입니까? 그리고 모델의 인스턴스는 몇 개입니까?

10.1 허프 변환

토큰 세트에 구조를 맞추려고 한다고 가정합니다(예: 점 세트에 선).
동일한 구조에 있을 수 있는 토큰을 클러스터링하는 한 가지 방법은 각 토큰이 있는 모든 구조를 기록한 다음 많은 표를 얻은 구조를 찾는 것입니다.
이 (매우 일반적인) 기술은 Hough 변환으로 알려져 있습니다. 허프 변환으로 구조를 맞추려면 각 이미지 토큰을 가져와 해당 토큰을 통과할 수 있는 모든 구조를 결정합니다. 우리는 이 세트를 기록하고(이것을 투표라고 생각해야 합니다) 각 토큰에 대해 프로세스를 반복합니다. 우리는 투표를 보고 무엇이 존재하는지 결정합니다. 예를 들어, 선 위에 있는 점을 그룹화하는 경우 각 점을 선택하고 통과할 수 있는 모든 선에 투표합니다. 이제 각 지점에 대해 이 작업을 수행합니다. 존재하는 선(또는 선)은 많은 점을 통과하므로 많은 투표를 하기 때문에 스스로를 분명하게 보여야 합니다.

10.1.1 허프 변환을 사용한 피팅 라인

선은 다음과 같이 점(x, y)의 집합으로 쉽게 매개변수화됩니다.

$$x \cos \theta + y \text{ 죄 } \theta + r = 0.$$

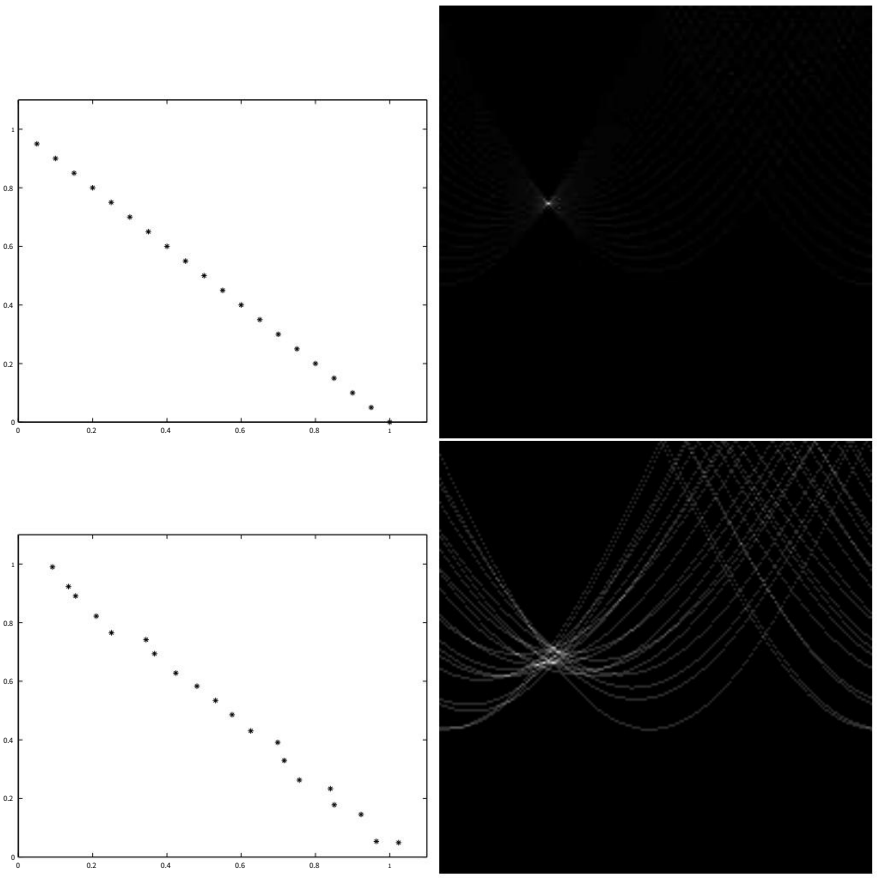

그림 10.1: Hough 변환은 토큰과 같은 각 점을 해당 점을 통과하는 가능한 선의 곡선(또는 다른 파라메트릭 곡선)에 매핑합니다. 이 그림은 선에 대한 Hough 변환을 보여줍니다. 왼쪽 열에는 포인트가 표시되고 오른쪽 열에는 해당 누산기 배열이 표시됩니다(투표 수는 회색 수준으로 표시되고 투표 수가 많을수록 밝은 점으로 표시됨). 맨 위 행은 선에서 그려진 20개의 점 세트를 사용하여 발생하는 일을 보여줍니다. 오른쪽 상단에는 이러한 포인트의 Hough 변환을 위한 누산기 배열이 있습니다. 각 포인트에 해당하는 것은 누산기 배열의 투표 곡선입니다. 가장 큰 투표 세트는 20입니다(가장 밝은 점에 해당). 누산기 배열의 수평 변수는 θ이고 수직 변수는 r입니다. 각 방향으로 200단계가 있고 r은 [0, 1.55] 범위에 있습니다.

아래쪽에서 이러한 점은 각 요소가 [0, 0.05] 범위에서 균일한 랜덤 벡터로 오프셋되었습니다. 이것은 점 옆에 표시된 누산기 배열의 곡선을 상쇄하고 최대 투표는 이제 6입니다(이 이미지에서 가장 밝은 값에 해당합니다. 이 값은 상단 이미지와 동일한 축척에서 보기 어렵습니다).

이제 (θ, r) 쌍은 고유한 선을 나타내며, 여기서 $r \geq 0$은 선에서 원점까지의 수직 거리이고 $0 \leq \theta < 2\pi$입니다. 우리는 쌍(θ, r) 라인 공간의 집합을 호출합니다. 공간은 반 무한 실린더로 시각화할 수 있습니다. 가족이 있다

모든 포인트 토큰을 통과하는 라인. 특히, r = x0 cos θ + y0 sin θ 에 의해 주어진 선 공간에서 곡선 위에 놓인 선은 모두 (x0, y0) 에서 점 토큰을 통과합니다 .

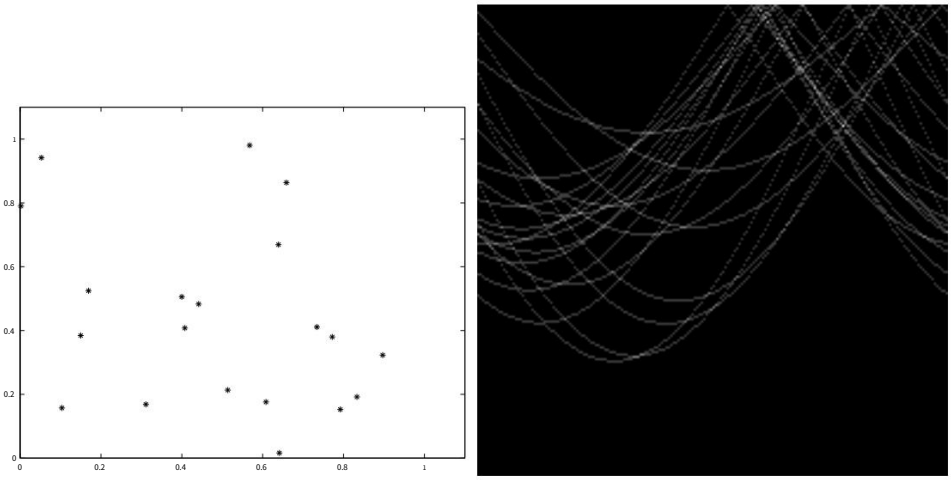

그림 10.2: 임의의 점 집합에 대한 Hough 변환은 누산기 배열에서 매우 큰 투표 집합으로 이어질 수 있습니다. 그림 10.1에서와 같이 왼쪽 열에는 포인트가 표시되고 오른쪽 열에는 해당 누산기 배열이 표시됩니다(투표 수는 회색 수준으로 표시되고 투표 수가 많을수록 밝은 점으로 표시됨). 이 경우 데이터 포인트는 노이즈 포인트입니다(두 좌표 모두 [0, 1] 범위의 균일한 난수임). 이 경우 누산기 배열에는 많은 중첩 지점이 포함되며 최대 투표는 이제 4입니다(그림 10.1의 6과 비교).

이미지의 크기가 알려져 있기 때문에 r > R의 선에는 관심이 없는 일부 R이 있습니다. 이 선은 원점에서 너무 멀리 떨어져 있어 볼 수 없습니다. 이는 우리가 관심을 갖는 선이 평면의 제한된 하위 집합을 형성하고 이를 편리한 그리드로 이산화한다는 것을 의미합니다. 그리드 요소는 투표를 하는 버킷으로 생각할 수 있습니다. 이 버킷 그리드를 누산기 어레이라고 합니다. 각 포인트 토큰에 대해 포인트 토큰에 해당하는 곡선의 모든 그리드 요소에 대해 형성된 총계에 투표를 추가합니다.

동일 선상에 있는 포인트 토큰이 많으면 해당 라인에 해당하는 그리드 요소에 투표가 많을 것으로 예상됩니다.

10.1.2 허프 변환 사용

Hough 변환은 매우 일반적인 절차입니다. 설명된 절차를 사용하여 예를 들어 원을 평면의 점에 맞추거나 구 또는 타원체를 3차원 데이터에 맞출 수 있습니다. 이것은 원칙적으로 작동하지만 실제로는 설명된 Hough 변환을 사용하기가 어렵습니다. 심지어 라인을 찾는 경우에도 마찬가지입니다. 여러 가지 어려움이 있습니다.

- 격자 치수: 선에 대한 누산기 배열의 치수는 2이지만 평면의 원에 대한 치수는 3(중심 위치 및 반지름)입니다. 평면의 축 정렬 타원의 경우 차원이 4입니다. 일반 타원의 경우

비행기, 다섯; 3D 구체의 경우 4개; 3D에서 축 정렬 타원체의 경우 7개; 3D의 일반적인 타원체의 경우 10입니다. 아주 간단한 구조라도 관리할 수 없는 양의 저장 공간을 차지하는 고차원 누산기 배열이 될 수 있습니다.

- 양자화 오류: 적절한 그리드 크기를 선택하기 어렵습니다. 그리드가 너무 거칠면 하나의 버킷에 상당히 다른 많은 구조가 대응하기 때문에 잘못된 투표 값을 얻을 수 있습니다. 그리드가 너무 미세하면 정확하게 정렬되지 않은 토큰으로 인한 투표가 다른 버킷에 들어가고 어떤 버킷도 큰 투표권을 가지지 않기 때문에 구조를 찾을 수 없게 될 수 있습니다(그림 10.1).

- 노이즈: Hough 변환의 매력은 일부 구조 가까이에 있는 널리 분리된 토큰을 연결한다는 것입니다. 합리적으로 균일하게 분포된 대규모 토큰 세트에서 꽤 좋은 팬텀 구조를 많이 찾을 수 있기 때문에 이것은 또한 약점입니다(그림 10.2). 이는 텍스처 영역이 검색된 라인과 관련된 것보다 더 큰 투표 배열에서 피크를 생성할 수 있음을 의미합니다.

이러한 어려움은 Hough 변환을 분포에서 모드를 찾으려는 시도로 인식함으로써 어느 정도 피할 수 있습니다. 분포는 투표 배열로 표시되며 일부 문제는 해당 배열의 셀에서 생성됩니다. 그러나 모드를 찾기 위해 반드시 누산기 배열을 사용할 필요는 없습니다. 대신 평균 이동을 적용할 수 있습니다. 섹션 9.3.4의 알고리즘을 직접 적용할 수 있습니다. 예를 들어 가장자리 감지기를 조정하여 텍스처를 부드럽게 하거나 고대비 가장자리를 보장하는 조명을 사용하거나 가장자리 포인트가 있는 더 복잡한 구조의 토큰을 사용하여 관련 없는 토큰을 최소화하는 것도 유용할 수 있습니다.

Hough 변환의 자연스러운 적용 중 하나는 객체 인식입니다. 자세한 내용은 18.4.2절로 미루지만 일반적인 그림은 다음과 같습니다. 객체가 알려진 부품으로 만들어졌다고 상상해 보십시오. 이러한 부품은 서로에 대해 약간 움직일 수 있지만 자체적으로 감지할 수 있을 만큼 충분히 큽니다.

그런 다음 감지된 각 부품이 물체의 위치(및 아마도 상태)에 대한 의견을 가질 것으로 기대할 수 있습니다. 즉, 먼저 부품을 감지한 다음 감지된 각 부품이 객체 인스턴스의 위치(및 상태)에 투표하도록 허용하고 마지막으로 Hough 변환(대부분 평균 이동 형식)을 사용하여 많은 부품이 있는 인스턴스를 찾을 수 있음을 의미합니다. 탐지기는 동의합니다. 이 접근법은 다양한 형태로 성공적으로 적용되었습니다(Maji and Malik 2009).

10.2 피팅 라인 및 평면

많은 응용 프로그램에서 객체는 직선의 존재로 특징지어집니다. 예를 들어 건물 사진을 사용하여 건물 모델을 만들고 싶을 수 있습니다(19장의 응용 프로그램에서와 같이). 이 응용 프로그램은 건물의 다면체 모델을 사용하므로 이미지의 직선이 중요합니다. 유사하게, 많은 산업 부품에는 한 형태 또는 다른 형태의 직선 모서리가 있습니다. 이미지에서 산업 부품을 인식하려면 직선이 도움이 될 수 있습니다. 두 경우 모두 이미지의 모든 직선에 대한 보고서는 매우 유용한 분할입니다. 이 모든 것은 라인을 평면 토큰 세트에 맞추는 것이 매우 유용하다는 것을 의미합니다. 평면 피팅

3D의 토큰도 유용하며 평면에 선을 맞추는 방법은 거의 변경 없이 적용됩니다.

10.2.1 단일 라인 맞추기

먼저 특정 선에 속하는 모든 점이 알려져 있고 선의 매개변수를 찾아야 한다고 가정합니다. 우리는 표기법을 채택

$$\overline{\text{안으로}} = \frac{\text{UI}}{\text{케이}}$$

프레젠테이션을 단순화합니다.

최소 제곱으로 알려진 피팅 라인에 대한 간단한 전략이 있습니다. 이 절차는 오랜 전통을 가지고 있지만(이것이 우리가 설명하는 유일한 이유입니다!) 상당한 편견이 있습니다. 대부분의 독자는 이 아이디어를 보았을 것이지만 많은 사람들이 이것이 야기하는 문제에 익숙하지 않을 것입니다. 이 접근 방식에서는 선을 y = ax + b로 나타냅니다. 각 데이터 포인트에는 (xi , yi) 가 있습니다 . 측정된 각 x 좌표에 대해 측정된 y 좌표를 가장 잘 예측하는 선을 선택하기로 결정합니다. 이것은 우리가 최소화하는 선을 선택하고 싶다는 것을 의미합니다.

$$\sum_{\text{나}} (\text{yi - 축 - b})^{2}.$$

차별화에 의해 문제에 대한 솔루션으로 라인이 제공됩니다.

$$\overline{\text{와}}^{2} = \overline{\text{배}}^{2}\ \overline{\text{엑스}}\ \text{ㅏ} \times 1 \text{비}.$$

이것은 고전적인 문제에 대한 표준 선형 솔루션이지만 비전 응용 프로그램에는 그다지 도움이 되지 않습니다. 모델이 매우 좋지 않기 때문입니다. 어려운 점은 측정 오류가 좌표 프레임에 따라 다르다는 것입니다. 우리는 선으로부터의 수직 오프셋을 오류로 계산하고 있습니다. 즉, 수직선 근처에서 상당히 큰 오류 값과 매우 재미있는 맞춤이 발생한다는 의미입니다(그림 10.3). 실제로 프로세스는 좌표 프레임에 너무 의존적이어서 수직선을 전혀 나타내지 않습니다.

점과 선 사이의 실제 거리(수직 거리가 아닌)로 작업할 수 있습니다. 이것은 전체 최소 제곱으로 알려진 문제로 이어집니다.

ax + by + c = 0인 점의 집합으로 선을 나타낼 수 있습니다. 모든 선은 이런 방식으로 나타낼 수 있으며 선을 값(a, b, c)의 삼중으로 생각할 수 있습니다. λ = 0인 경우 λ(a, b, c)로 표시된 직선은 (a, b, c)로 표시된 직선과 같습니다. 연습문제에서 간단하지만 매우 유용한 점(u, v)에서 선(a, b, c)까지의 수직 거리가 다음과 같이 주어진다는 결과를 증명해야 합니다.

$$\text{abs(au + bv + c) 만약 a}^{2\,2+\text{비}} = 1.$$

경험상 이 사실은 기억할 가치가 있을 만큼 충분히 유용합니다. 점과 선 사이의 수직 거리의 합을 최소화하려면 다음을 최소화해야 합니다.

$$\sum_{\text{나}} (\text{축} + \text{바이아} + \text{c})^{2},$$

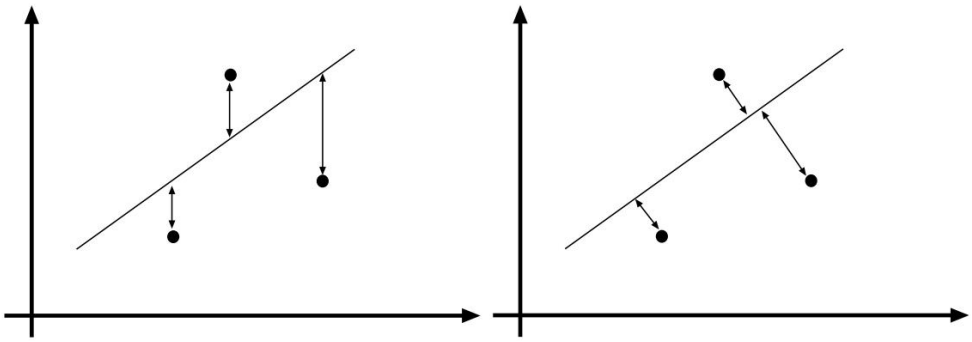

그림 10.3: 왼쪽: 최소 제곱은 선과 토큰 사이의 제곱된 수직 거리의 합을 최소화하는 선을 찾습니다(오류가 y 좌표에만 나타난다고 가정하기 때문입니다). 이것은 적합하지 않은 비용으로 (매우 약간) 더 간단한 수학적 문제를 생성합니다. 오른쪽: 총 최소 제곱은 토큰과 선 사이의 제곱된 수직 거리의 합을 최소화하는 선을 찾습니다. 이것은 예를 들어 수직에 가까운 선을 어려움 없이 맞출 수 있음을 의미합니다.

에 따라 $^{2\,2+비}$ = 1. 이제 라그랑주 승수 λ를 사용하면 다음과 같은 솔루션이 있습니다.

$$\begin{matrix} \overline{배^2} & \overline{xy} & x\ \overline{2} \\ \overline{xy} & yy & xy\ 1 \end{matrix} \begin{matrix} 아 \\ 비 \\ 씨 \end{matrix} = \lambda \begin{matrix} 2a \\ 2b \\ 0 \end{matrix} .$$

이것은

$$c = \overline{ax} \text{ 에 의해,}$$

고유값 문제를 얻기 위해 이것을 다시 대체할 수 있습니다.

$$\begin{matrix} \overline{x2} - xx & \overline{xy} - xy \\ \overline{2xy} - xyy & -년 \end{matrix} \begin{matrix} 아 \\ 비 \end{matrix} = \mu \begin{matrix} 아 \\ 비 \end{matrix} .$$

이것은 2D 고유값 문제이기 때문에 최대 두 가지 솔루션을 닫힌 형식으로 얻을 수 있습니다(관심 있는 사람들을 위해 일반적으로 수치로 수행됩니다!). 척도는 a = 1이라는 제약 조건에서 얻은 2입니다. 이 문제에 대한 두 가지 솔루션은 직각의 선입니다. 하나는 제곱 거리의 합을 최대화하고 다른 하나는 이를 최소화합니다.

10.2.2 평면 맞추기

피팅 평면은 피팅 라인과 매우 유사합니다. 평면을 z = ux + vy + w로 표현한 다음 최소 제곱을 적용할 수 있습니다. 이것은 수직 평면을 잘 나타내지 않기 때문에 최소 제곱선 피팅과 마찬가지로 편향됩니다. 전체 최소 제곱은 라인 피팅과 마찬가지로 더 나은 전략입니다. 평면을 ax + by + cz + d = 0으로 표현합니다. 그러면 점 xi = (xi , yi , zi) 에서 평면까지 의 거리는 (axi + byi + czi + d) = 1이 되며 이제 위의 분석을 약간 변경하여 사용할 수 있습니다.

^{2} $^{2\,경우\ +비^{2\,2+c}}$

10.2.3 여러 줄 맞추기

이제 토큰 세트(예: 포인트)가 있고 이 토큰 세트에 여러 줄을 맞추고 싶다고 가정합니다. 이 문제는 큰 조합 공간을 검색하는 것과 관련될 수 있기 때문에 어려울 수 있습니다. 한 가지 접근 방식은 우리가 고립된 점을 거의 만나지 않는다는 것을 알아차리는 것입니다. 대신 많은 문제에서 선을 가장자리 점에 맞춥니다. 모서리 점의 방향을 선의 다음 점 위치에 대한 힌트로 사용할 수 있습니다. 고립된 지점에 갇혀 있으면 k-평균을 적용할 수 있습니다.

증분 라인 피팅 알고리즘은 에지 포인트의 연결된 곡선을 취하고 곡선을 따라 점의 런에 라인을 맞춥니다. 에지 포인트의 연결된 곡선은 출력이 방향을 제공하는 에지 검출기에서 쉽게 얻을 수 있습니다(연습 참조). 증분 필터는 모서리 점 곡선의 한쪽 끝에서 시작하여 곡선을 따라 이동하면서 선에 잘 맞는 픽셀 실행을 잘라냅니다(알고리즘의 구조는 알고리즘 10.1에 표시됨). 증분 라인 피팅은 기본 통계 모델이 없음에도 불구하고 잘 작동할 수 있습니다. 한 가지 기능은 닫힌 곡선을 형성하는 선 그룹을 보고한다는 것입니다. 이것은 알고리즘이 더 이상의 소란 없이 자연 그룹을 보고한다는 것을 의미하기 때문에 관심 라인이 폐곡선을 형성할 것으로 합리적으로 예상될 수 있을 때(예: 일부 객체 인식 응용 프로그램에서) 매력적입니다. 이 전략을 사용할 때 가장자리의 폐색으로 인해 둘 이상의 선분이 경계에 맞춰질 수 있습니다. 이 어려움은 선을 후처리하여 (대략) 일치하는 쌍을 찾음으로써 해결할 수 있지만 두 선이 일치하는 시기를 결정하는 현명한 기준을 제공하기 어렵기 때문에 프로세스가 다소 매력적이지 않습니다.

```
곡선을 따라 순서대로 모든 점을 곡선 목록에 넣습니다.
라인 포인트 목록 비우기
회선 목록 비우기
곡선에 너무 적은 점이 있을 때까지
    곡선의 처음 몇 점을 선 점 목록으로 전송
    라인 포인트 목록에 라인 맞추기
    피팅 라인이 충분하면서
        곡선의 다음 점 전송
            라인 포인트 목록으로 이동하고 라인을 다시 맞춥니다.
    끝
    마지막 점을 곡선으로 다시 전송
    리핏 라인
    회선 목록 끝에 회선 연결
```

알고리즘 10.1: 증분 라인 피팅.

이제 포인트가 어떤 라인에 놓여 있는지에 대한 힌트를 가지고 있지 않다고 가정합니다(즉, 색상 정보 또는 도움이 되는 정보가 없으며, 결정적으로 포인트가 연결되어 있지 않음). 또한, 우리가 얼마나 많은 라인이 있는지 알고 있다고 가정합니다. 수정된 버전의 k-평균을 사용하여 어떤 점이 어떤 선에 있는지 확인하려고 시도할 수 있습니다. 이 경우 모델은 k개의 라인이 있고 각 라인은 다음을 생성합니다.

데이터 포인트의 일부 하위 집합. 라인 및 데이터 포인트에 대한 최상의 솔루션은 다음을 최소화하여 얻을 수 있습니다.

$$\text{i 번째 라인으}^{2}$$

$$\text{로 인한 dist}(l_i , x_j) \; l_i \in \text{lines} \; x_j \in \text{data}$$

서신과 라인 모두를 통해. 다시 말하지만, 이 공간을 검색하기에는 너무 많은 대응이 있습니다.

이 문제를 처리하기 위해 k-평균을 수정하는 것은 쉽습니다. 두 단계는 다음과 같습니다.

- 각 점을 가장 가까운 선에 할당합니다.
- 각 라인에 할당된 포인트에 가장 적합한 라인을 맞춥니다.

결과는 알고리즘 10.2입니다. 수렴은 선의 변화 크기, 레이블이 뒤집혔는지 여부(아마도 가장 좋은 테스트) 또는 선에서 점까지의 수직 거리의 합을 확인하여 테스트할 수 있습니다.

k개의 라인을 가정합니다(아마도 균일하게 무작위로).
또는
점에 대한 선의 할당을 가정한 다음 이 할당을 사용하여 선을 맞
춥니다.

수렴까지
각 점을 가장 가까운 선에 할당
리핏 라인 끝

알고리즘 10.2: K-평균 라인 피팅.

10.3 곡선 구조 맞추기

2D의 곡선은 2D의 선과 다릅니다. 평면의 모든 토큰에 대해 가장 가까운 선에 고유한 단일 점이 있습니다. 곡선의 경우에는 그렇지 않습니다.
곡선이 휘기 때문에 토큰에 가장 가까운 것처럼 국부적으로 보이는 곡선에 둘 이상의 점이 있을 수 있습니다(그림 10.4). 이것은 점과 곡선 사이의 최소 거리를 찾는 것이 매우 어려울 수 있음을 의미합니다. 3D 표면에서도 유사한 효과가 발생합니다. 이 어려움을 무시한다면 피팅 곡선은 피팅 라인과 유사합니다. 곡선 선택의 함수로 점과 곡선 사이의 거리 제곱합을 최소화합니다.

곡선이 암시적이라고 가정하고 형식이 $\phi(x, y) = 0$이라고 가정합니다. 암시적 곡선의 가장 가까운 점에서 데이터 점까지의 벡터는 곡선에 수직이므로 가장 가까운 점은 다음을 모두 찾아 주어집니다. 다음 속성을 가진 (u, v):

1. (u, v)는 곡선의 한 점이며, 이는 $\phi(u, v) = 0$임을 의미합니다. 2. $s = (dx, dy)$ (u, v)
는 곡선에 수직입니다.

그림 10.4: 마치 토큰에 가장 가까운 것처럼 국지적으로 보이는 곡선에는 둘 이상의 점이 있을 수 있습니다. 이로 인해 곡선을 점에 맞추는 것이 매우 어렵습니다. 왼쪽에는 곡선과 토큰이 있습니다. 파선은 토큰을 로컬 테스트에서 가장 근접할 수 있는 곡선의 두 지점에 연결합니다. 로컬 테스트는 점선과 곡선의 접선이 직각인지 확인합니다. 중앙과 오른쪽에는 곡선 일부의 복사본이 표시됩니다. 각각에 대해 곡선의 일부가 없기 때문에 토큰에 대한 세그먼트의 가장 가까운 지점이 다릅니다. 결과적으로 지점이 가장 가깝다는 것을 보장하는 로컬 테스트를 수행할 수 없습니다. 모든 후보자를 확인해야 합니다.

이러한 모든 s가 주어지면 가장 짧은 길이는 데이터 포인트에서 곡선까지의 거리입니다.

두 번째 기준은 법선을 결정하기 위해 약간의 작업이 필요합니다. 암시적 곡선의 법선은 곡선을 가장 빨리 떠나는 방향입니다. 이 방향을 따라 ϕ의 값도 가장 빠르게 변해야 합니다. 이것은 점 (u, v)에서의 법선이 다음임을 의미합니다.

$$\frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y} \quad ,$$

(u, v)에서 평가됩니다. 곡선의 접선이 T이면 Ts = 0이어야 합니다.
우리는 2D에서 작업하고 있기 때문에 법선에서 탄젠트를 결정할 수 있습니다.

$$(u, v) \{dx - u\} - \frac{\partial \phi}{\partial y} \frac{\partial \phi}{\partial x}(u, v) \{dy - v\} = 0$$

점 (u, v)에서. 우리는 이제 두 개의 미지수에 대한 두 개의 방정식을 가지고 있고, 원칙적으로 그것들을 풀 수 있습니다. 그러나 많은 솔루션이 있을 수 있기 때문에 보기보다 쉬운 경우는 거의 없습니다. 우리는 ϕ가 차수 d의 다항식인 경우에 d를 기대하지만, 그 중 일부는 d^2 복잡할 수 있습니다.

파라메트릭 곡선을 사용해도 상황이 개선되지 않습니다. 매개변수 곡선의 점 좌표는 매개변수의 함수이므로 t가 매개변수인 경우 곡선은 (x(t), y(t))로 쓸 수 있습니다. 데이터 포인트 (dx, dy)가 있다고 가정합니다.
매개변수 곡선에서 가장 가까운 지점은 매개변수 값으로 식별할 수 있으며 이를 τ로 작성합니다. 이 점은 곡선의 한쪽 또는 다른 쪽 끝에 있을 수 있습니다. 그렇지 않으면 데이터 포인트에서 가장 가까운 포인트까지의 벡터가 곡선에 수직입니다. 이는 s(τ) = (dx, dy) (x(τ), y(τ))가 탄젠트 벡터에 수직이므로 s(τ).T = 0입니다.
탄젠트 벡터는 다음과 같습니다.

$$\frac{dx}{dt}(\tau), \frac{dy}{dt}(\tau),$$

이는 τ가 다음 방정식을 충족해야 함을 의미합니다.

$$\frac{dx}{dt}(\tau)\{dx - x(\tau)\} + \frac{dy}{dt}(\tau)\{dy - y(\tau)\} = 0.$$

이제 이것은 두 개가 아닌 하나의 방정식이지만 상황은 파라메트릭 곡선의 경우보다 훨씬 낫지 않습니다. x(t)와 y(t)가 다항식인 경우는 거의 항상 있습니다. 왜냐하면 일반적으로 다항식에 대해 근 찾기를 수행하는 것이 더 쉽기 때문입니다. 최악의 경우 x(t)와 y(t)는 다항식의 비율입니다. 이 경우에도 방정식의 왼쪽을 재정렬하여 다항식을 얻을 수 있기 때문입니다. 그러나 우리는 여전히 많은 수의 뿌리에 직면해 있습니다. 근본적인 문제는 기하학적입니다. 곡선에는 국지적으로 주어진 데이터 포인트에 가장 가까운 포인트로 나타나는 많은 포인트가 있을 수 있습니다. 곡선이 평평하지 않기 때문입니다(그림 10.4). 각각을 차례로 확인하지 않고는 가장 가까운 것을 알 수 있는 방법이 없습니다. 경우에 따라(예: 원) 이 문제를 회피할 수 있습니다. 이 어려움은 곡선 표면을 3D의 점에 맞출 때도 적용됩니다.

이 매우 일반적인 문제를 처리하기 위한 두 가지 전략이 있습니다. 하나는 점과 곡선(또는 3D에서는 점과 표면) 사이의 거리를 근사치로 대체하는 것인데, 때때로 효과적입니다. 다른 하나는 곡선이나 표면의 표현을 수정하는 것입니다. 예를 들어 샘플 집합 또는 선분 집합이 있는 곡선을 나타낼 수 있습니다. 마찬가지로 표면은 샘플 세트 또는 메시로 표시될 수 있습니다. 그런 다음 12장의 방법을 사용하여 이러한 표현을 데이터에 등록하거나 데이터에 맞게 변형할 수도 있습니다.

10.4 견고성

설명된 모든 선 맞춤 방법에는 제곱 오차 항이 포함됩니다. 이것은 매우 부적절한 단일 데이터 포인트가 많은 좋은 데이터 포인트로 인해 데이터 포인트를 지배하는 오류를 제공할 수 있기 때문에 실제로 적합하지 않을 수 있습니다. 이러한 오류는 피팅 프로세스에서 상당한 편향을 초래할 수 있습니다(그림 10.5). 이 효과는 오류의 제곱으로 인해 발생합니다. 실제로 이러한 데이터 포인트(일반적으로 이상치라고 함)를 피하는 것은 어렵습니다. 데이터 포인트 수집 또는 기록 오류는 이상값의 중요한 원인 중 하나입니다. 또 다른 일반적인 원인은 모델의 문제입니다. 드물지만 중요한 일부 효과가 무시되었거나 효과의 크기가 심하게 과소평가되었을 수 있습니다. 마지막으로, 대응 오류는 특히 이상값을 생성하기 쉽습니다. 실제 시력 문제에는 일반적으로 이상값이 포함됩니다.

이 문제는 추정치에 대한 먼 지점의 영향을 줄이거나(10.4.1절) 외곽 지점을 식별하고 무시함으로써 해결할 수 있습니다. 외곽 지점을 식별하는 방법에는 두 가지가 있습니다. 좋은 점을 찾을 수 있습니다. 좋은 점의 작은 세트는 우리가 맞추고자 하는 것을 식별할 것입니다. 다른 좋은 점은 동의하고 동의하지 않는 점은 나쁜 점입니다. 이것은 섹션 10.4.2에 설명된 매우 중요한 접근 방식의 기초입니다. 또는 이를 데이터 누락 문제로 간주하고 섹션 10.5에서 설명한 EM 알고리즘을 사용할 수 있습니다.

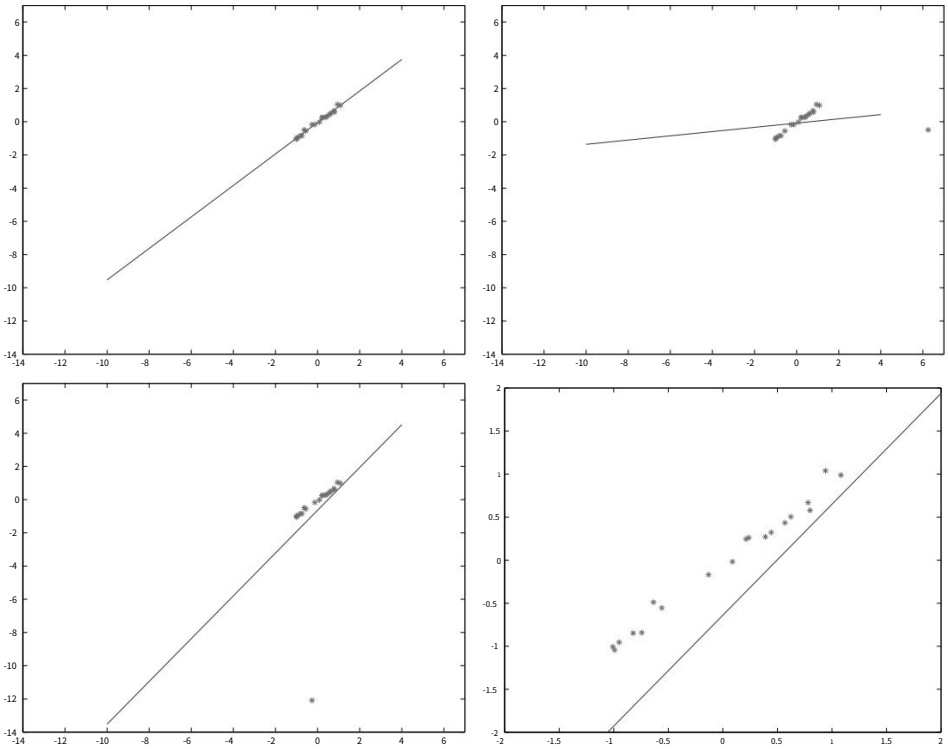

그림 10.5: 제곱 오차가 있는 라인 피팅은 x 및 y 좌표 모두에서 이상값에 매우 민감합니다. 최소 제곱을 사용하는 예를 보여줍니다. 왼쪽 상단에는 일련의 점에 대한 선의 최소 자승 적합도가 있습니다. 오른쪽 상단은 동일한 점 집합을 보여주지만 한 점의 x 좌표가 손상되었습니다. 이는 포인트가 있어야 할 위치에서 수평으로 변환되었음을 의미합니다. 결과적으로 실제 선에 엄청난 오류 항을 제공하고 선의 방향을 크게 변경하여 더 나은 최소 제곱 적합을 얻습니다. 이렇게 하면 대부분의 지점에서 오류가 커지지만 이상값에서 매우 큰 오류가 줄어듭니다. 왼쪽 하단은 동일한 점 집합을 보여주지만 한 점의 y 좌표가 손상되었습니다. 이 특별한 경우 x 절편이 변경되었습니다. 이 세 그림은 비교를 위해 동일한 축 세트에 있지만 이 축 선택은 세 번째 경우에 얼마나 적합하지 않은지 명확하게 보여주지 않습니다. 오른쪽 하단은 이 사례의 세부 사항을 보여주며, 선이 분명히 적합하지 않습니다.

10.4.1 M-추정기

M-추정기는 제곱 오차 항을 더 잘 작동하는 항으로 대체하여 매개변수를 추정합니다. 이것은 우리가 형식의 표현을 최소화한다는 것을 의미합니다.

$$\rho(r_i(x_i, \theta); \sigma),$$

여기서 θ는 피팅되는 모델의 매개변수이고(예를 들어 선의 경우 방향과 y 절편이 있을 수 있음) $r_i(x_i, \theta)$는 i번째 데이터에 대한 모델의 잔여 오차입니다. 가리키다. 이 표기법을 사용하여 우리의 최소

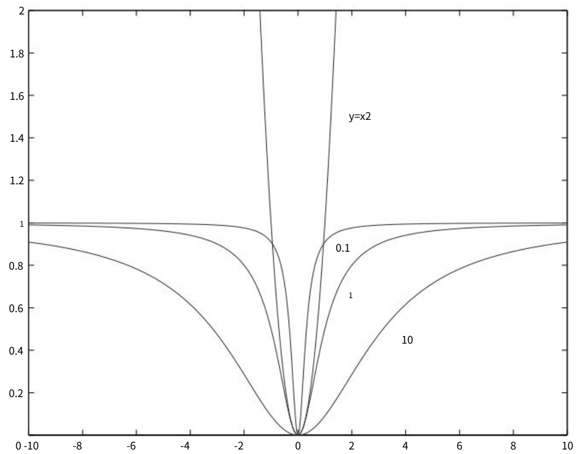

그림 10.6: 함수 ρ(x; σ) = x = 0.1, 1, 10, 비교를 위한 플롯 $(y = x^2)$ 2차 항을 제곱 대체 플롯 값이 피팅에 미치는 영향이 줄어듭니다. 피팅된 곡선에서 σ^2의 몇 배 떨어진 지점은 피팅된 곡선의 계수에 거의 영향을 미치지 않을 것입니다. ρ의 값이 1에 가까워지고 피팅된 곡선으로부터의 거리에 따라 매우 느리게 변경되기 때문입니다. .

제곱 및 총 최소 제곱 라인 피팅 오류(잔차 오류의 형태만 다름)는 둘 다 ρ(u; σ) = u ρ(u; σ)가 범위의 일부에 대해 u처럼 보이다가 평평해집니다. 우리는 ρ(u; σ)가 단조롭게 증가하고 큰 ^{2}. M-추정기의 요령은 u에 대해 일정한 값에 가깝다고 예상합니다. 일반적인 선택은

$$\rho(u; \sigma) = \frac{인치^2}{시^2 + 유^2}.$$

매개변수 σ는 함수가 평평해지는 지점을 제어하며 그림 10.6에 다양한 예를 표시했습니다. 사용할 수 있는 다른 많은 M-추정기가 있습니다. 일반적으로 다음과 같이 정의되는 영향 기능 측면에서 논의됩니다.

$$\frac{\partial \rho}{\partial \theta}.$$

최소화 기준이 다음과 같기 때문에 이는 자연스러운 현상입니다.

$$\sum_{나} \rho(r_i(x_i, \theta); \sigma) = 0 \frac{\partial \rho}{\partial \theta}$$

솔루션에서. 우리가 고려하는 종류의 문제에 대해 우리는 좋은 영향 함수가 반대칭(약간의 과대 예측과 약간의 과소 예측 사이에 차이가 없음)을 기대하고 큰 값으로 끝납니다. 이상치의 영향을 제한하기 때문입니다. .

M-추정기를 사용하는 데는 두 가지 까다로운 문제가 있습니다. 첫째, 최소화 문제는 비선형이며 반복적으로 해결해야 합니다. 표준 어려움이 적용됩니다. 하나 이상의 로컬 최소값이 있을 수 있고 방법이 분기될 수 있으며 방법의 동작이 시작점에 크게 의존할 수 있습니다.

s = 1에서 s = k인 경우
무작위로 균일하게 선택된 r개의 구별되는 점의 하위 집합을 그립니다.
매개변수 θ의 초기 세트를 얻기 위해 최소 제곱을 사용하여 이 점 세트에 맞춥니다.
추정 p θ를 사용하여
수렴할 때까지 (보통 |θ n θ | 작다):
θ를 사용하여 최소화 단계를 수행하여 θ n을 ,
얻습니다.
이제 σ end end를 계
산합
니다.
잔차의 중앙값을 사용하여 이 k 시행 세트의 가장 적합한 것을 보고합니다.
기준으로

알고리즘 10.3: M-Estimator를 사용하여 최소 제곱 모델에 적합

이 문제를 처리하기 위한 일반적인 전략은 데이터 세트의 하위 샘플을 그리고 최소 제곱을 사용하여 해당 하위 샘플에 맞추고 이를 피팅 프로세스의 시작점으로 사용하는 것입니다. 우리는 이 작업을 많은 수의 서로 다른 하위 샘플에 대해 수행하여 좋은 데이터 포인트로 완전히 구성된 하위 샘플이 하나 이상 있을 가능성이 높도록 합니다(알고리즘 10.3).

둘째, 그림 10.7과 10.8에서 알 수 있듯이 추정기는 σ의 합리적인 추정치를 필요로 하며, 이를 종종 척도라고 합니다. 일반적으로 척도 추정치는 솔루션 방법의 각 반복에서 제공됩니다. 대중적인 규모 추정치는

$$\text{피}^{(n)} = 1.4826 \text{ 중앙값 } |r \quad {}^{(n)}_{\text{나}} (xi ;\theta (n-1))| .$$

알고리즘 10.3에서 일반적인 M-estimator를 요약합니다.

10.4.2 RANSAC: 좋은 점 찾기

비용 함수 수정에 대한 대안은 데이터 포인트 모음에서 좋은 포인트를 검색하는 것입니다. 이것은 반복 프로세스로 매우 쉽게 수행됩니다. 먼저 점의 작은 하위 집합을 선택하고 해당 하위 집합에 맞춘 다음 결과 개체에 맞는 다른 점이 몇 개인지 확인합니다. 찾고 있는 구조를 찾을 가능성이 높을 때까지 이 프로세스를 계속합니다.

예를 들어 약 50% 이상치로 구성된 데이터 세트에 선을 맞추는 중이라고 가정합니다. 두 점에만 선을 맞출 수 있습니다. 한 쌍의 점을 균일하고 무작위로 그리면 이 쌍의 약 1/4은 전적으로 좋은 데이터 점으로 구성됩니다. 우리는 다른 점들의 큰 집합이 그러한 쌍에 맞는 선 가까이에 있다는 것을 알아차림으로써 이러한 좋은 쌍을 식별할 수 있습니다. 물론 현재 선에 가까운 지점에 선을 맞추면 선의 더 나은 추정치를 얻을 수 있습니다.

Fischler와 Bolles(1981)는 이 접근 방식을 알고리즘으로 공식화했습니다. 즉, 많은 데이터 포인트가 일치하는 적합성으로 이어지는 임의의 샘플을 검색합니다.
알고리즘은 일반적으로 RANdom SAMple Consensus의 경우 RANSAC라고 하며 다음과 같습니다.

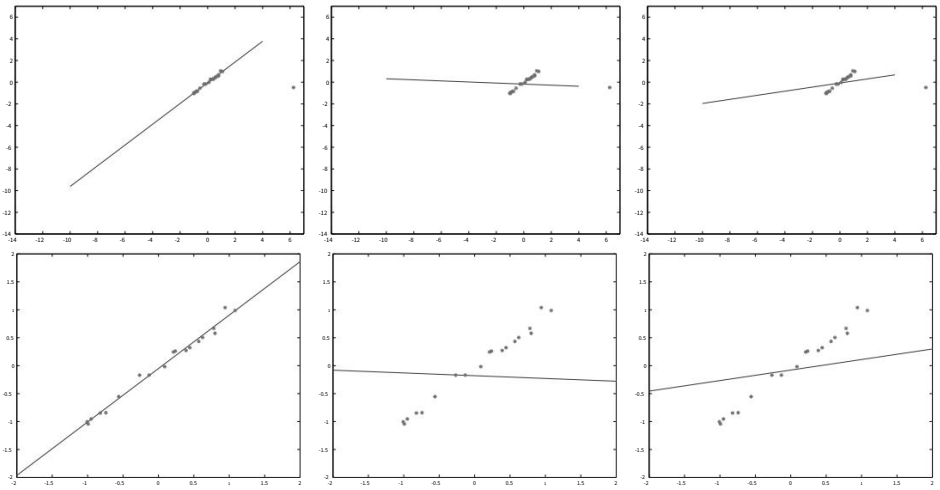

그림 10.7: 맨 위 행은 먼 지점의 기여를 덜 강조하는 가중치 함수(그림 10.6의 함수 φ)를 사용하여 그림 10.5의 두 번째 데이터 세트에 맞는 선을 보여줍니다. 왼쪽에서 μ는 대략 올바른 값을 가집니다. 특이치의 기여도가 낮아졌으며 적합도가 좋습니다. 중앙에서 μ의 값이 너무 작아 피팅이 모든 데이터 포인트의 위치에 민감하지 않아 데이터와의 관계가 모호합니다. 오른쪽에서 μ의 값이 너무 커서 이상값이 최소 제곱에서와 거의 동일한 기여를 한다는 것을 의미합니다. 맨 아래 행은 적합선의 근접 촬영과 동일한 사례에 대한 외부에 있지 않은 데이터 포인트를 보여줍니다.

알고리즘 10.4에 표시됩니다. 이 알고리즘을 실용화하려면 세 가지 매개변수를 선택해야 합니다.

필요한 샘플 수 샘플은 데이터 세트에서 균일하고 무작위

로 추출된 점 세트로 구성됩니다. 각 샘플에는 관심 있는 추상화에 맞는 데 필요한 최소 포인트 수가 포함되어 있습니다. 예를 들어 선을 맞추려면 점 쌍을 그립니다. 원을 맞추려면 세 개의 점을 그리는 식입니다. 우리는 n개의 데이터 포인트를 그려야 하고 w는 이러한 포인트 중 좋은 부분이라고 가정합니다(이 숫자에 대한 합리적인 추정만 필요함). 이제 1점을 얻는 데 필요한 무승부 k의 예상 값은 다음과 같습니다.

$$E[k] = 1P(\text{한 번의 추첨에서 하나의 좋은 샘플}) + 2P(\text{두 번의 추첨에서 하나의 좋은 샘플}) + \ldots n\,)\,2w^{n} + 2(1 - w^{n})w^{n} + 3(1 - \ldots$$

$$n = w \quad w \quad + \ldots$$

승 -n =

(여기서 마지막 단계는 대수 시리즈를 약간 조작합니다). 우리는 우리가 좋은 샘플을 보았다는 것을 상당히 확신하고 싶기 때문에 w -n 샘플보다 더 많은 샘플을 뽑기를 원합니다. 자연스러운 일은 여기에 몇 가지 표준 편차를 추가하는 것입니다.

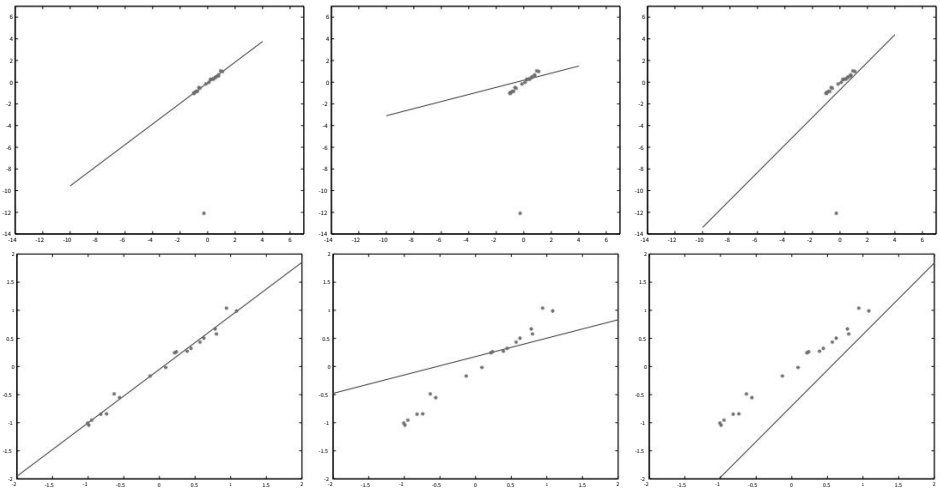

그림 10.8: 맨 위 행은 먼 지점의 기여도를 강조하지 않는 가중치 함수(그림 10.6의 함수 ϕ)를 사용하여 그림 10.5의 세 번째 데이터 세트에 맞는 선을 보여줍니다. 왼쪽에서 μ는 대략 올바른 값을 가집니다. 특이치의 기여도가 낮아졌으며 적합도가 좋습니다. 중앙에서 μ의 값이 너무 작아 피팅이 모든 데이터 포인트의 위치에 민감하지 않아 데이터와의 관계가 모호합니다. 오른쪽에서 μ의 값이 너무 커서 이상값이 최소 제곱에서와 거의 동일한 기여를 한다는 것을 의미합니다. 맨 아래 행은 동일한 경우에 대해 적합선과 외부에 있지 않은 데이터 포인트의 클로즈업을 보여줍니다.

숫자. k의 표준 편차는 다음과 같이 구할 수 있습니다.

$$SD(\text{케이}) = \frac{\sqrt{1 - wn}}{\text{남}}.$$

이 문제에 대한 또 다른 접근 방식은 나쁜 샘플만 볼 수 있는 낮은 확률 z를 보장하는 많은 샘플을 보는 것입니다. 이 경우, 우리는

$$(1 - wn)^{\text{케이}} = \text{지},$$

의미하는 것은

$$k = \frac{\log(z)}{\log(1 - wn)}.$$

w가 알려지지 않은 데이터를 처리해야 하는 것이 일반적입니다. 그러나 각 피팅 시도에는 w에 대한 정보가 포함됩니다. 특히 n개의 데이터 포인트가 필요한 경우 성공적인 피팅 확률은 w n이라고 가정할 수 있습니다. 피팅 시도의 긴 시퀀스를 관찰하면 이 시퀀스에서 w를 추정할 수 있습니다. 이것은 우리가 상대적으로 낮은 w 추정치로 시작하여 일련의 적합 시도를 생성한 다음 w 추정치를 개선함을 시사합니다. w의 새로운 추정치가 예측하는 것보다 더 많은 피팅 시도가 있는 경우 프로세스가 중지될 수 있습니다. w의 추정치를 업데이트하는 문제는 일련의 일치가 주어지면 동전이 앞면 또는 뒷면이 나올 확률을 추정하는 것으로 축소됩니다.

```
다음을 결정합니
    다. n - 필요한 최소 포인트 수(예: 선의 경우 n = 2, 원의 경우 n = 3) k - 필요한 반복 횟수 t -
      잘 맞는 포인트를 식별하
    는 데 사용되는 임계값 d - 숫자 모델이 잘 맞는지 확인
    하는 데 필요한 인근 포인트 수


k 반복이 발생할 때까지
    데이터에서 균일하고 무작위로 n 포인트의 샘플을 그립니다.

    해당 n 포인트 세트에 맞추기
    샘플 외부의 각 데이터 포인트에 대해
      포인트에서 구조물까지의 거리 테스트
        t에 대하여; 점에서 구조물까지의 거리가 t보다 작으면 점이 가까운 것입니다.

    끝
    구조물 가까이에 d개 이상의 점이 있는 경우
      그러면 잘 맞습니다. 이 모든 점을 사용하여 구조를 다시 맞춥니다. 좋은 맞춤
      모음에 결과를 추가합니다.
끝
피팅 오류를 기준으로 사용하여 이 컬렉션에서 가장
  잘 맞는 것을 사용합니다.
```

알고리즘 10.4: RANSAC: 무작위 샘플 합의를 사용하여 구조를 피팅합니다.

점이 가까운지 여부 확인 샘플에 맞는 선에 점이 가까운
지 여부를 확인해야 합니다.

점과 적합선 사이의 거리를 결정하고 임계값 d에 대해 해당 거리를 테스트하여 이를 수행합니다. 거리가 임계값 미만이면 점이 가까운 것입니다. 일반적으로 이 매개변수를 지정하는 것은 모델링 프로세스의 일부입니다. 이 매개변수의 값을 얻는 것은 비교적 간단합니다. 우리는 일반적으로 규모 추정치만 필요하며 동일한 값이 여러 다른 실험에 적용됩니다. 매개변수는 종종 몇 가지 값을 시도하고 어떤 일이 발생하는지 확인하여 결정됩니다. 또 다른 접근 방식은 몇 가지 특징적인 데이터 세트를 살펴보고 눈으로 선을 맞추고 편차의 평균 크기를 추정하는 것입니다.

일치해야 하는 점의 수 두 데이터 점의 임의 샘플에 선을 맞추었고 그
선이 좋은지 여부를 알아야 한다고 가정합니다. 우리는 선의 일정 거리 내에 있는 점의 수를 세어 이를 수행합니다(거리는 이전 섹션에서 결정됨). 특히, 이상값이 이 포인트 모음에 있을 확률을 알고 있다고 가정합니다. 이 확률을 y로 씁니다. 근처의 모든 포인트가 작을 확률(예: 0.05 미만)이 되도록 몇 개의 포인트 t를 선택하려고 합니다. $y \leq (1 \quad w)$(선이 이상값이기 때문에 y 일부 이상값은 선에서 멀리 떨어져 있어야 함)이므로 (1 w)가 되도록 t를 선택할 수 있습니다.

티
,

티

작다.

10.5 확률 모델을 사용한 피팅

우리가 설명한 피팅 절차에서 확률 모델을 구축하는 것은 간단합니다. 그렇게 하면 새로운 종류의 모델과 새로운 알고리즘이 생성됩니다. 둘 다 실제로 매우 유용합니다. 핵심은 관찰된 데이터를 생성 모델에 의해 생성된 것으로 보는 것입니다. 생성 모델은 각 데이터 포인트가 생성된 방법을 지정합니다.

가장 간단한 경우, 최소 제곱을 사용한 라인 피팅에서는 자연 생성 모델을 사용하여 섹션 10.2.1에서 작업한 것과 동일한 방정식을 복구할 수 있습니다.
데이터 생성을 위한 우리의 모델은 x 좌표가 균일하게 분포되고 y 좌표는 (a) x 좌표에 해당하는 선에서 점 axi + b를 찾은 다음 (b) 제로 평균 정규 분포 랜덤을 추가하여 생성됩니다 변하기 쉬운. 이제 x가 확률 분포 p의 표본임을 의미하기 위해 x p라고 씁니다. 특정 범위의 값에 대한 균일 분포에 대해 U(R)을 작성하십시오.

아르 자형? N(μ, p를 쓴다. ^{2}) 평균 mu 및 분산 σ를 갖는 정규 분포의 경우 ^{2}.
표기법으로 다음과 같이 쓸 수 있습니다.

$$xi \quad U(R)\ yi$$
$$N(축 + b, \sigma \quad ^2).$$

우리는 이 모델의 알려지지 않은 매개변수를 간단한 방법으로 추정할 수 있습니다.
중요한 매개변수는 a와 b입니다(σ를 아는 것이 유용할 수 있음). 확률 모델에서 매개변수를 추정하는 일반적인 방법은 일반적으로 음의 로그 가능성으로 작업하고 이를 최소화하여 데이터의 가능성을 최대화하는 것입니다. 이 경우 데이터의 로그 우도는 다음과 같습니다.

$$L(a, b, \sigma) = \sum_{\text{아이}\in\text{데이터}} \log P(xi , yi \,|a, b, \sigma)$$
$$= \sum_{i\in data} \log P(yi \,|xi , a, b, \sigma) + \log P(xi)$$
$$= \sum_{\text{아이}\in\text{데이터}} - \frac{(yi \quad (축 + b))2}{2\sigma 2} - \frac{1}{2} 로그\ 2\pi\sigma 2 + Kb$$

여기서 Kb 는 log P(xi)를 나타내는 상수입니다 . 이제 a와 b의 함수로 음의 로그 가능도를 최소화하기 위해 a 와 b의 함수로 i∈data(yi (axi + b))2를 최소화할 수 있습니다(최소 제곱 라인 피팅에 대해 수행한 작업). 섹션 10.2.1에서).

이제 전체 최소제곱 선 맞춤을 고려하십시오. 다시, 자연 생성 모델에서 10.2.1절에서 작업한 방정식을 복구할 수 있습니다. 이 경우 데이터 포인트(xi , yi)를 생성 하기 위해 라인을 따라 무작위로 균일하게 포인트(ui , vi) 를 생성합니다. 그런 다음 거리 ξi (여기서 ξi N(0, σ2))를 샘플링하고 그 거리만큼 선에 수직인 점(ui , vi) 을 이동합니다 . 선이 ax + by + c = 0이고 a = 1이면, 우리는 (xi , yi) = (ui , vi) + ξi(a, b)를 가집니다. 우리는 다음과 같이 쓸 수 있습니다.

$$^2 \quad 2 + b$$

이 모델에서 데이터의 로그 우도는 다음과 같습니다.

$$L(a,b,c,\sigma) = \sum_{i \in data} \log P(x_i, y_i | a,b,c,\sigma)$$

$$= \sum_{\text{데이터}} \log P(\xi_i | \sigma) + \log P(u_i, v_i | a,b,c). \text{ 아이}\in$$

그러나 $P(u_i, v_i | a,b,c)$는 이 점이 선을 따라 균일하게 분포되어 있기 때문에 일정합니다. ξ_i 는 (x_i, y_i) 에서 선까지의 수직 거리 ($||(ax_i + by_i + c)|| \; a^2 + b^2 = 1$인 경우)이므로 최대화해야 합니다.

$$\text{로그 } P(\xi_i | \sigma) = \sum_{\text{데이터}} \text{아이}\in\text{데이터} \; - \frac{\text{엑스}^2}{\text{나는 2초 2}} - \frac{1}{2} \text{로그 2x2}$$

$$= \sum_{\text{아이}\in\text{데이터}} - \frac{(\text{축} + \text{바이}_i + c)^2}{2\sigma^2} - \text{ㅏ로그 2πs2 2}$$

(다시, $a^2 + b^2 = 1$에 따라). 고정된(그려봐^2 아마도 알려지지 않은) σ의 경우 이것은 섹션 10.2.1에서 작업했던 문제를 산출합니다. 지금까지 생성 모델은 우리가 이미 알고 있는 것을 재현했을 뿐이지만 강력한 트릭을 통해 훨씬 더 흥미롭게 만들었습니다.

10.5.1 누락된 데이터 문제

많은 중요한 시력 문제는 데이터의 유용한 요소가 누락된 문제로 표현될 수 있습니다. 예를 들어, 측정값이 나온 여러 소스 중 어느 것을 결정하는 문제로 세분화를 생각할 수 있습니다. 이것은 일반적인 견해입니다. 보다 구체적으로 말하자면, 일련의 토큰에 라인을 맞추는 것은 토큰을 이상치와 인라이어로 분할한 다음 라인을 인라이어에 맞추는 것과 관련됩니다. 이미지를 영역으로 분할하는 것은 이미지 픽셀을 생성한 색상 및 텍스처 픽셀의 소스를 결정하는 것을 포함합니다. 한 세트의 라인을 토큰 세트에 맞추는 것은 어떤 토큰이 어떤 라인에 있는지 결정하는 것을 포함합니다. 모션 시퀀스를 이동 영역으로 분할하는 것은 이동 픽셀을 모션 모델에 할당하는 것을 포함합니다. 이러한 각 문제는 현재 누락된 데이터(각각 포인트가 인라이어인지 아웃라이어인지, 픽셀이 어느 지역에서 왔는지, 토큰이 어느 라인에서 왔는지, 어떤 모션 모델이 픽셀에서 온다).

누락 데이터 문제는 일부 데이터가 누락된 통계적 문제입니다.
누락된 데이터가 중요한 두 가지 자연스러운 맥락이 있습니다. 첫째, 데이터 벡터의 일부 용어가 어떤 경우에는 누락되고 다른 경우에는 존재합니다(아마도 설문 조사에 응답한 사람이 질문에 당황했을 것입니다). 우리의 응용 프로그램에서 훨씬 더 일반적인 두 번째에서는 값을 알 수 없는 일부 변수를 사용하여 다시 작성하여 추론 문제를 훨씬 더 간단하게 만들 수 있습니다.

다행히 누락된 데이터 문제를 처리하기 위한 효과적인 알고리즘이 있습니다. 본질적으로 우리는 누락된 데이터에 대한 기대를 합니다. 두 가지 예를 통해 이 방법과 적절한 알고리즘을 시연합니다.

예: 이상치 및 라인 피팅 xi = (xi , yi) 에 있는 토큰 세
트에 라인을 맞추려고 합니다 . 일부 토큰은 이상치일 수 있지만 어떤 토큰인지 알 수 없습니다. 이것은 우리가 먼저 토큰을 생성하는 프로세스를 모델링할 수 있다는 것을 의미합니다. 라인에서 나올지 이상치인지 선택한 다음 원래 선택에 따라 조건이 지정된 토큰을 선택할 수 있습니다.

첫 번째 선택은 무작위이며 P(token comes from line) = π라고 쓸 수 있습니다.
선 모델에서 포인트를 생성하는 방법에 대한 두 가지 모델을 이미 제공했습니다. 이상값은 평면에서 균일하고 무작위로 발생하는 것으로 모델링합니다. 이것은 우리가 토큰을 생성할 확률을 다음과 같이 쓸 수 있음을 의미합니다.

$$\begin{aligned} P(xi \mid a, b, c, \pi) &= P(xi, \text{직선} \mid a, b, c, \pi) + P(xi, \text{특이치} \mid a, b, c, \pi) \\ &= P(xi \mid \text{선}, a, b, c)P(\text{선}) + P(xi \mid \text{이상치}, a, b, c)P(\text{이상치}) \\ &= P(xi \mid \text{선}, a, b, c)\pi + P(xi \mid \text{이상치}, a, b, c)(1 - \pi). \end{aligned}$$

모든 데이터 항목에 대해 그것이 선에서 왔는지 이상치인지를 안다면 선을 맞추는 것은 간단할 것입니다. 우리는 모든 이상값을 무시하고 섹션 10.2.1의 방법을 다른 지점에 적용합니다. 유사하게, 우리가 선을 안다면 어떤 점이 이상치이고 어떤 점이 아닌지 추정하는 것이 간단할 것입니다(이상치는 선에서 멀리 떨어져 있습니다). 어려움은 우리가 그렇게 하지 않는다는 것입니다. 이 어려움을 해결하는 열쇠는 반복적인 재추정(10.5.3절)이며, 이 문제 클래스에 대한 표준 알고리즘을 제공합니다. 그림 10.9는 표준 알고리즘을 사용한 일반적인 결과를 보여줍니다.

위 방정식을 아주 조금만 조작하면("선"을 "배경"으로, "이상치"를 "전경"으로 대체) 배경 빼기 문제도 나타낼 수 있습니다. 비디오의 각 프레임에 있는 이미지를 동일하게 모델링하고 자동 게인 제어를 고려하기 위해 일부 상수를 곱했지만 노이즈가 추가되었습니다. 노이즈는 일정한 소스에서 오는 것으로 모델링합니다. 그림 10.10과 10.11은 이러한 문제에 대한 표준 알고리즘으로 얻은 결과를 보여줍니다(10.5.3절).

예: 이미지 분할 이미지의 각 픽셀에서 위치, 색
상 및 질감 정보를 포함할 수 있는 d차원 특징 벡터 x를 계산합니다. 우리는 이미지가 g 세그먼트를 포함하고 각 픽셀이 이러한 세그먼트 중 하나에 의해 생성된다고 생각합니다. 따라서 픽셀을 생성하려면 이미지 세그먼트를 선택한 다음 해당 세그먼트의 모델에서 픽셀을 생성합니다. 우리는 l번째 세그먼트가 확률로 선택되고 특정 세그먼트에 따라 달라지는 알려진 공분산 Σ 및 알 수 없는 평균 θl = (µl) 을 사용하여 l번째 세그먼트와 관련된 밀도를 가우시안으로 모델링한다고 가정합니다. 이러한 매개변수를 매개변수 벡터로 캡슐
재발 , 화하여 Θ = (π1,..., πg, θ1,...,θg)를 얻습니다 . 이것은 픽셀 벡터 x를 생성할 확률을 다음과 같이 쓸 수 있음을 의미합니다.

$$p(x|\Theta) = \sum_{\text{나}} p(x|\theta l)\pi l .$$

각 세그먼트의 평균을 개별적으로 추정할 수 있기 때문에 어떤 세그먼트가 어떤 픽셀을 생성했는지 알면 이 모델을 맞추는 것이 간단할 것입니다. 마찬가지로, 평균을 알면 픽셀을 생성한 세그먼트를 추정할 수 있습니다. 이것은 매우 일반적인 상황입니다.

10.5.2 혼합 모델과 숨겨진 변수

앞의 각 예는 혼합 모델로 알려진 일반적인 모델 형식의 인스턴스입니다. 여기서 데이터 항목은 먼저 혼합 구성 요소(선 또는 이상값, 픽셀이 나오는 세그먼트)를 선택한 다음 다음을 생성하여 생성됩니다. 해당 구성 요소의 데이터 항목. l번째 구성요소 θl 에 대한 매개변수를 호출 하고 Θ = (π1,... , , l번째 성분 πl을 선택할 확률 , 제발 , θ1,...,θl). 그런 다음 x를 생성할 확률을 쓸 수 있습니다.

$$p(x|\Theta) = \sum_{\text{제발}} p(x|\theta_j)\pi_j .$$

이것은 확률 모델의 가중 합계 또는 혼합입니다. πl 은 일반적으로 혼합 가중치라고 합니다. 이 모델을 확률의 g "블롭" 세트로 구성된 x 공간의 밀도로 시각화할 수 있으며, 각각은 모델의 구성 요소와 연관됩니다. (a) 각 블롭의 매개변수, (b) 혼합 가중치, 일반적으로 (c) 각 토큰의 구성 요소를 결정해야 합니다. 일반 혼합 모델에 대한 데이터의 로그 우도는 다음과 같습니다.

$$\text{엘}(\Theta) = \sum_{\text{그 i}\in\text{관측}} \text{로} \sum_{j=1}^{g} \pi_j p_j(\text{일부}\,|\theta_j) .$$

이 함수는 대수 내부의 합계 때문에 최대화하기 어렵습니다. 마지막 두 예와 마찬가지로 각 토큰이 나오는 혼합 구성 요소를 알면 구성 요소를 독립적으로 추정할 수 있기 때문에 문제가 단순화됩니다.

이제 새로운 변수 세트를 소개합니다. 각 데이터 항목에 대해 각 데이터 항목이 어느 구성 요소에서 왔는지 알려주는 표시 변수 벡터(구성 요소당 하나)가 있습니다. i번째 데이터 항목과 관련된 벡터에 대해 δi를 쓰고 δi 의 j번째 구성 요소에 대해 δij를 씁니다 . 그런 다음

$$\delta_{ij} = \begin{cases} \text{항목 i가 구성요소 j에서 나온 경우 1 그렇지 않은} \\ \text{경우 0} \end{cases} .$$

이러한 변수는 알 수 없습니다. 이러한 변수를 알고 있다면 전체 데이터 로그 우도를 최대화할 수 있습니다.

$$LC(\Theta) = \sum_{i\in\text{관측}} \log P(x_i, \delta_i|\Theta),$$

이는 매우 쉬울 것입니다(구성 요소를 독립적으로 추정하는 것으로 귀결되기 때문입니다). 우리는 δ를 누락된 데이터의 일부로 간주합니다(이를 완전한 데이터 로그 우도라고 부르는 이유입니다). 혼합 모델에 대한 Lc(Θ) 의 형식은 다음과 같은 깔끔한 트릭을 포함하므로 기억할 가치가 있습니다.

δij를 사용하여 용어를 켜고 끕니다. 우리는

$$LC(\Theta) = \sum_{i \in \text{관측}} \log P(x_i, \delta_i | \Theta)$$

$$= \sum_{\text{그 } i \in \text{관측값}} \log \prod_{j \in \text{구성요소}} [p_j(x_i|\theta_j)\pi_j]^{\text{디지}}$$

$$= \sum_{i \in \text{관측}} \sum_{j \in \text{성분}} [(\log p_j(x_i|\theta_j) + \log \pi_j)\delta_{ij}]$$

(δij 는 1 또는 0이고, δij = 1이라는 점 에 유의하십시오 . 이는 각 데이터 포인트가 정확히 하나의 모델에서 나오도록 요구하는 것과 같습니다).

10.5.3 혼합 모델에 대한 EM 알고리즘

각각의 예에서 누락된 데이터를 알고 있으면 매개변수를 효과적으로 추정할 수 있습니다. 마찬가지로 매개변수를 알고 있으면 누락된 데이터가 뒤따를 것입니다. 이것은 반복 알고리즘을 제안합니다.

1. 매개변수에 대한 추측을 사용하여 누락된 데이터의 추정치를 얻습니다.
2. 추정값을 사용하여 자유 매개변수의 최대 우도 추정값을 형성합니다.
   누락된 데이터의

수렴될 때까지 (바라건대!) 이 절차를 반복합니다. 라인 피팅의 경우 알고리즘은 다음과 같습니다.

1. 어떤 점이 선 위에 있고 어떤 점이 선에 없는지에 대한 추정치를 얻습니다.
   선 추정치를 사용합니다.
2. 이 정보를 사용하여 라인의 수정된 추정치를 구성합니다.

이미지 분할의 경우 다음과 같습니다.

1. θl 의 추정치를 사용하여 각 픽셀의 특징 벡터가 나온 구성 요소의 추정치를 구합니다 .

2. 이 추정치를 사용하여 θl 과 혼합 가중치를 업데이트합니다.

누락된 데이터에 대해 주어진 절차가 수렴되면 좋겠지만 그렇게 믿을 특별한 이유는 없습니다. 실제로 각 단계에서 적절한 선택이 주어지면 그렇게 합니다. 이는 일반적인 알고리즘인 EM(기대 최대화) 알고리즘의 예임을 보여줌으로써 가장 쉽게 알 수 있습니다.

EM의 핵심 아이디어는 각 누락된 값에 대한 기대값을 대체하여 누락된 데이터(및 Θ)에 대한 작업 값 세트를 얻는 것입니다. 특히 매개변수를 어떤 값으로 고정한 다음 xi 값 과 매개변수 값이 주어지면 각 δij 의 예상 값을 계산합니다. 그런 다음 δij 의 예상 값을 작업하기 훨씬 쉬운 전체 데이터 로그 우도에 연결하고 이를 최대화하여 매개변수 값을 얻습니다. 이 시점에서 δij 의 예상 값이 변경되었을 수 있습니다. 기대 단계와 최대화 단계를 번갈아 가며 알고리즘을 얻고 수렴될 때까지 반복합니다. 더

공식적으로 주어진 Θ(s) , 다음과 같이 Θ(s+1)을 형성합니다 .

1. 불완전한 데이터와 매개변수의 현재 값을 사용하여 완전한 데이터 로그 우도에 대한 기대값을 계산합니다. 즉, 우리는 계산

$$Q(\Theta; \Theta(s)) = E_{\delta|x,\Theta(s)} Lc(\Theta).$$

이 객체는 Θ 및 δ의 함수를 기대하여 얻은 Θ의 함수입니다. 예상은 P(δ|x, Θ(s)) 에 대한 것입니다 . 이를 E-스텝이라고 합니다.

2. 이 객체를 Θ의 함수로 최대화합니다. 즉, Θ (s+1) 을 계산합니다.

$$= \text{인수 최대 Th } Q(\Theta; \Theta(s)).$$

이것은 M 단계로 알려져 있습니다.

불완전한 데이터 대수 우도가 각 단계에서 증가하고 인컴의 (로컬) 최대값으로 수렴한다는 것을 보여줄 수 로 수렴된다는 보장은 없으며 올바른 로 있습니다. (1996)). 물론 이 알고리즘이 올바른 로컬 최대값으 컬 최대값을 찾는 것이 다소 번거로울 수 있습니다.

EM은 혼합 모델을 찾는 것보다 훨씬 쉽습니다. 먼저,
섹션 10.5.2는 혼합물에 대한 전체 데이터 로그 우도 모델이 다음과 같다는 것을 나타냅니다.

$$Lc(\Theta) = \sum_{i \in \text{관측값}} \sum_{j \in \text{구성요소}} [(\log p_j(x_i|\theta_j) + \log \pi_j)\, \delta_{ij}].$$

이것은 δ에서 선형입니다. 기대치를 취하는 것은 선형이기 때문에 Q(Θ; Θ(s))는 δij 의 기대값을 대입하여 Lc(Θ) 에서 얻을 수 있습니다 . 이제 다음 과 같이 씁니다 . _ _ _ _ 이들은 일반적으로 소프트

웨이트라고 합니다. 우리

는 이제 쓸 수 있습니다

$$Q(\Theta; \Theta(s)) = \sum_{i \in \text{관측값}} \sum_{j \in \text{구성요소}} [(\log p_j(x_i|\theta_j) + \log \pi_j)\, \alpha_{ij}] \cdot$$

둘째, i번째 결측 변수는 주어진 i번째 데이터 포인트와 모델의 매개변수에 대해 다른 모든 변수와 조건부로 독립적이라는 점에 유의하십시오. 이것이 혼란스럽다면 예제에 대해 생각해 보십시오. 라인 피팅의 경우 특정 포인트가 아웃라이어인지 여부를 확인하는 데 필요한 유일한 정보는 해당 포인트와 라인 추정치입니다. 다른 점은 그것에 대해 말할 것이 없습니다. 마지막으로

$$\begin{aligned}
\alpha_{ij} &= E_{\delta|x,\Theta(s)}[\delta_{ij}] \\
&= E_{\delta i|xi,\Theta(s)}[\delta_{ij}]\ \Theta \\
&= 1 \cdot P(\delta_{ij} = 1|x_i, \Theta(s)) + 0 \cdot P(\delta_{ij} = 0|x_i, \Theta(s)) \\
&= P(\delta_{ij} = 1|x_i, \quad (s)).
\end{aligned}$$

이제 계산을 해야 합니다

$$P(\delta ij = 1|xi, \quad \Theta(들)) = \frac{P(xi, \delta ij = 1|\Theta(s))}{P(xi|\Theta(들))}$$

$$= \frac{P(xi|\delta ij = 1, \Theta(s))P(\delta ij = 1|\Theta(s))}{P(xi|\Theta(s))\, pj(xi|}$$

$$= \frac{\Theta(s))\pi j\ \delta il = 1|\Theta(s))}{\sum P(xi,}$$

$$= \frac{pj(xi|\Theta(s))\pi j}{\sum pl(xi|\Theta(s))\pi l}$$

분자는 모델 j에서 데이터 포인트를 얻을 확률이고 분모는 해당 포인트를 얻을 확률이기 때문입니다. 우리의 단계는 다음과 같습니다.

E-Step 각 i, j에 대해 소프트 가중치를 계산합니다.

$$\alpha ij = P(\delta ij = 1|xi, \quad \Theta(s)) = \frac{pj(xi|\Theta(s))\pi j}{\sum pl(xi|\Theta(s))\pi l}.$$

그런 다음

$$Q(\Theta; \Theta(s)) = [(\log pl(xi|\theta l) \log \pi l)\, \alpha ij]. \quad i\in 관측값\ j\in 구성요소$$

M-Step 우리는 극대화해야 합니다

$$Q(\Theta; \Theta(s)) = \sum_{i\in 관측값\ j\in 구성요소} [(\log pl(xi|\theta l) \log \pi l)\, \alpha ij].$$

Θ의 함수로. 이는 각 데이터 포인트를 가중치 αij를 사용하여 j번째 모델에 할당한 다음 각 모델의 우도를 개별적으로 최대화하는 것과 동일합니다. 프로세스는 각 모델이 각 데이터 포인트의 일부를 설명하는 것처럼 작동하므로 이러한 용어를 소프트 가중치라고 합니다. 이는 예제에 대한 방정식을 공부할 때 더 분명해질 것입니다(연습 참조).

10.5.4 EM 알고리즘의 어려움

EM은 지역 최소값에 갇히는 경향이 있습니다. 이러한 로컬 최소값은 일반적으로 연구 중인 문제의 조합적 측면과 관련이 있습니다. 아웃라이어에 영향을 받는 피팅 라인의 예에서 알고리즘은 본질적으로 포인트가 아웃라이어인지 여부를 결정하려고 시도합니다. 일부 잘못된 라벨은 안정적일 수 있습니다. 예를 들어 이상값이 하나만 있는 경우 알고리즘은 해당 지점과 다른 지점을 통과하는 선을 찾고 나머지 모든 지점을 이상값으로 표시할 수 있습니다(그림 10.9).

한 가지 유용한 전략은 알고리즘의 최종 구성이 시작점의 결정론적 기능이라는 점을 인식하고 신중하게 선택한 시작점을 사용하는 것입니다. 하나는 다양한 (무작위로 선택된) 구성에서 시작하여 훑어볼 수 있습니다.

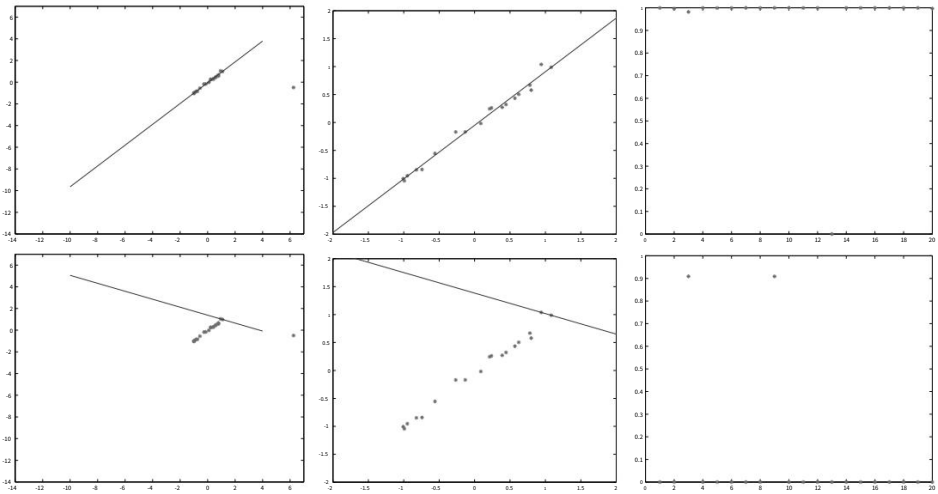

그림 10.9: EM을 사용하여 이상값을 거부할 수 있습니다. 여기서 우리는 그림 10.5의 두 번째 데이터 세트에 대한 선 맞춤을 보여줍니다. 상단 행은 올바른 로컬 최소값을 보여주고 하단 행은 또 다른 로컬 최소값을 보여줍니다. 첫 번째 열은 그림 10.5와 동일한 축을 사용하여 데이터 포인트에 겹쳐진 선을 보여줍니다. 두 번째 열은 데이터 포인트 주변의 영역을 나타내는 선의 상세 보기를 보여줍니다. 세 번째 열은 점의 인덱스에 대해 플로팅된 노이즈 모델이 아닌 선에서 점이 나올 확률의 플롯을 보여줍니다. 올바른 로컬 최소값에서는 하나를 제외한 모든 포인트가 라인과 연결되어 있는 반면, 잘못된 로컬 최소값에서는 라인과 연결된 두 개의 포인트가 있고 나머지는 노이즈에 할당됩니다.

RANSAC과 같은 최적의 결과를 찾는 결과입니다. 가장 적합한 것을 찾기 위해 Hough 변환과 같은 것을 사용하여 데이터를 사전 처리할 수 있습니다. 둘 다 보장되지 않습니다.

두 번째 어려움은 일부 포인트의 예상 가중치가 매우 작다는 것입니다. 이것은 우리에게 수치적인 문제를 제시합니다. 작은 가중치를 0과 동일하다고 간주하면 어떤 일이 발생하는지 명확하지 않습니다(이는 일반적으로 현명한 일이 아닙니다). 차례로, 우리는 매우 작은 숫자를 더하고 0이 아닌 결과를 도출할 수 있는 숫자 표현을 채택해야 할 수도 있습니다. 이 문제는 이 책의 범위를 벗어나지만 자세히 다루지 않기 때문에 문제의 가치를 과소평가해서는 안 됩니다.

10.6 매개변수 추정에 의한 움직임 분할

움직이는 카메라에 의해 생성된 모션 시퀀스의 두 프레임을 고려하십시오. 작은 움직임의 경우 비교적 적은 수의 새 포인트를 볼 수 있고 상대적으로 적은 포인트를 잃을 수 있으므로 첫 번째 프레임의 각 포인트를 두 번째 프레임(겹쳐진)의 해당 포인트에 화살표로 연결할 수 있습니다. 머리는 두 번째 프레임의 지점에 있으며 경과 시간이 짧으면 화살표 필드는 이미지의 순간적인 움직임으로 생각할 수 있습니다. 화살표는 원래 Gibson(1950)에 기인한 개념인 광학 흐름으로 알려져 있습니다. 옵티컬의 구조

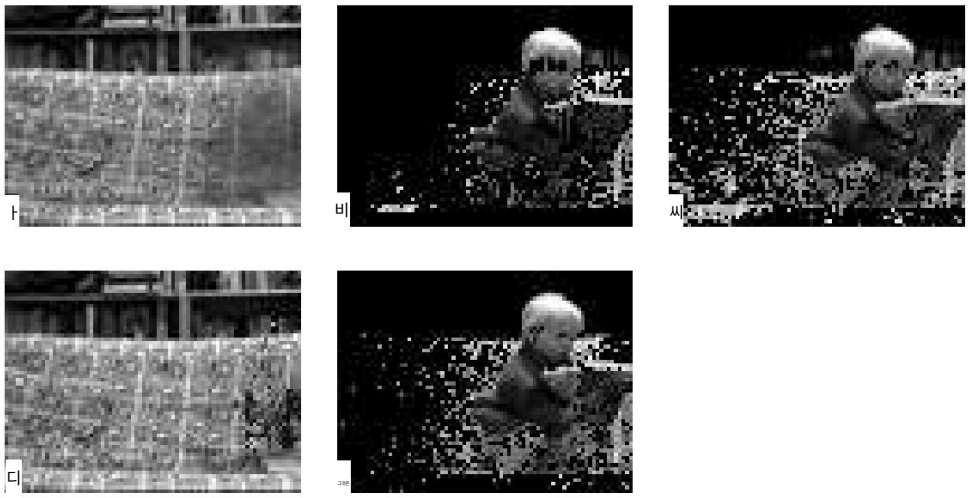

그림 10.10: EM을 사용하여 그림 9.8의 시퀀스에 대한 배경 빼기. (a), (b) 및 (c)는 비교를 위해 그림 9.9에서 가져온 것입니다. (d)는 추정된 배경을 보여주고 (e)는 추정된 전경을 보여줍니다. 각 경우에 약간의 초과 픽셀과 일부 누락 픽셀이 있습니다.

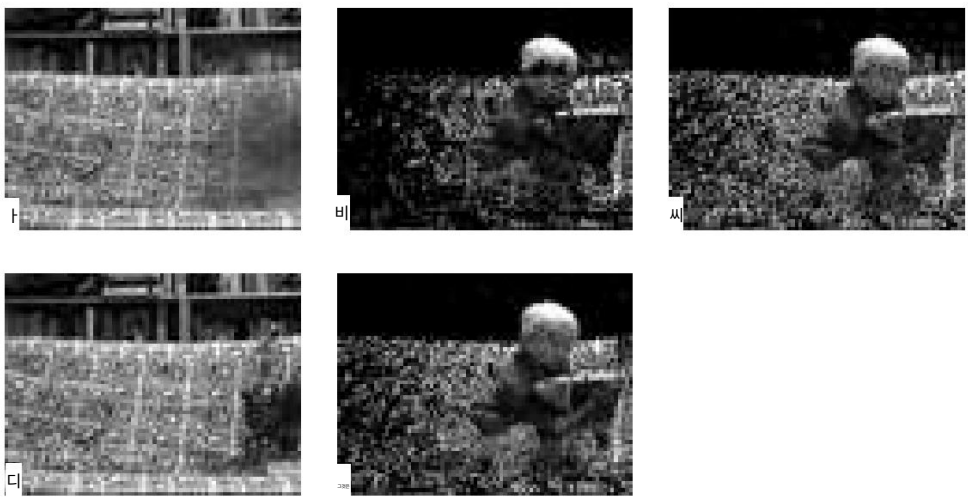

그림 10.11: EM을 사용하여 그림 9.8의 시퀀스에 대한 배경 빼기. (a), (b) 및 (c)는 비교를 위해 그림 9.10에서 가져온 것입니다. (d)는 추정된 배경을 나타내고, (e)는 추정된 전경을 나타낸다. 소파의 패턴이 어린이로 오인된 문제 픽셀의 수가 눈에 띄게 증가했음을 알 수 있습니다. 작은 움직임으로 인해 소파의 높은 공간 주파수 패턴이 어긋나 큰 차이가 발생할 수 있기 때문입니다.

흐름 필드는 장면에 대해 매우 유익할 수 있으며(10.6.1절) 광학 흐름의 매우 간단한 파라메트릭 모델은 종종 좋은 표현입니다(10.6.2절).

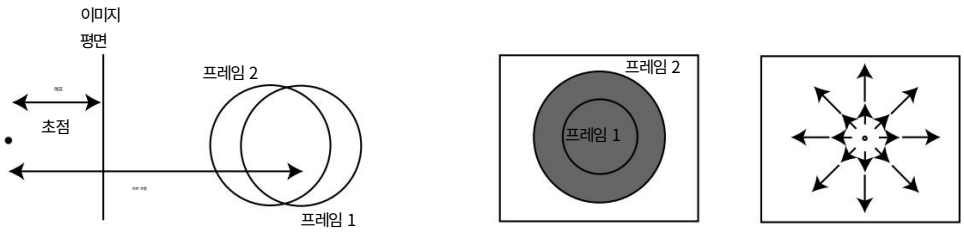

그림 10.12: 반지름이 R인 구가 속도 V로 Z축을 따라 카메라에 접근합니다(왼쪽 측면 보기). 이미지는 구가 가까워질수록 커지는 원입니다(중앙). 흐름은 확장 초점에 대한 방사형이며 접촉 시간의 추정치를 제공합니다(오른쪽). 이 추정치는 다른 개체에도 적용됩니다.

결과적으로 모션 시퀀스는 종종 내부적으로 유사한 모션을 갖는 큰 영역으로 구성됩니다. 차례로 이것은 우리에게 분할 원칙을 제공합니다. 우리는 시퀀스를 만들기 위해 구성되는 일련의 이동 레이어로 모션 시퀀스를 분해하려고 합니다(섹션 10.6.3).

10.6.1 광학 흐름 및 모션

흐름은 일반적으로 에고모션이라고 하는 시청자의 모션과 3D 장면 사이의 관계에 대해 특히 유익합니다. 예를 들어, 움직이는 자동차에서 볼 때 멀리 있는 물체는 가까운 물체보다 겉보기 움직임이 훨씬 느리므로 겉보기 움직임의 속도는 거리에 대해 알 수 있습니다. 이것은 멀리 있는 물체의 흐름 화살표가 가까운 물체의 흐름 화살표보다 짧다는 것을 의미합니다.

또 다른 예로, egomotion이 어떤 방향으로의 순수한 이동이라고 가정합니다.
그러면 확장 초점으로 알려진 해당 방향의 이미지 지점이 움직이지 않고 모든 광학 흐름이 해당 지점에서 멀어집니다(그림 10.12).
이것은 단순히 그러한 유동장을 관찰하는 것만으로도 우리가 어떻게 움직이고 있는지에 대해 알 수 있음을 의미합니다. 더 간단한 관찰은 우리가 무언가를 얼마나 빨리 칠 것인지를 알려줍니다.
확장의 초점에 카메라가 있다고 가정하고 세계를 카메라로 이동시킵니다. 중심이 운동 방향을 따라 있고 깊이 Z에 있는 반지름 R의 구는 반지름 r = fR/Z의 원형 이미지 영역을 생성합니다. 속도 V = dZ/dt로 Z축 아래로 이동하면 이미지에서 이 영역의 성장 속도는 dr/dt = fRV /Z가 됩니다.

2 . 이것은

$$\text{접촉 시간} = \frac{Z}{\text{안에}} = \frac{\quad}{(\frac{\text{박사}}{dt})}.$$

빼기 기호는 구가 Z축 아래로 움직이기 때문에 Z가 점점 작아지고 V가 음수이기 때문입니다. 이 인수가 작동하기 위해 개체가 구형일 필요는 없으며 카메라가 구형이면 우리가 이동하는 방향을 바라볼 필요도 없습니다. 이것은 빠르게 번역하는 동물이 무언가를 아주 빠르고 쉽게 칠 때까지 걸리는 시간을 추정할 수 있음을 의미합니다.

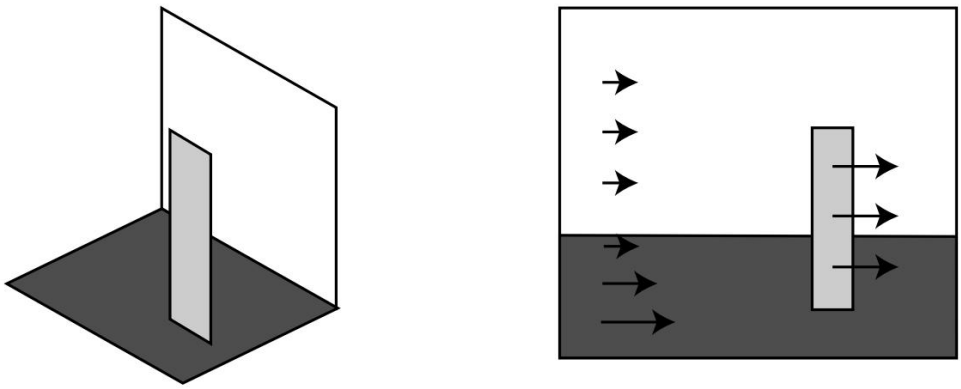

그림 10.13: 광학 흐름 필드는 장면을 구조화하거나 분할하는 데 사용할 수 있습니다. 왼쪽에는 매우 간단한 장면이 있습니다. 이제 이미지 평면이 흰색 사각형과 평행하고 왼쪽으로 이동하는 카메라로 이 장면을 본다고 상상해 보십시오. 오른쪽 이미지와 같은 흐름 필드를 볼 수 있습니다. 흰색 사각형의 흐름은 일정하고(평면이 변환 방향 및 이미지 평면과 평행하기 때문에) 작고(멀리 있음); 밝은 회색 직사각형에서는 일정하지만 더 큽니다. 경사면에서는 먼 지점에서는 작고 가까운 지점에서는 큽니다. 이러한 흐름 필드의 파라메트릭 모델을 사용하면 다른 구조가 다른 흐름 필드에 해당하기 때문에 이와 같이 장면을 분할할 수 있습니다.

10.6.2 흐름 모델

매우 간단한 파라메트릭 흐름 모델은 장면의 일부를 함께 그룹화할 수 있습니다(그림 10.13). 매개변수가 선형인 모델을 구축하는 것이 도움이 됩니다. 매개변수 벡터의 i번째 구성요소에 대한 θi , i번째 흐름 기반 벡터 필드에 대한 Fi , 픽셀 x에서 흐름 벡터에 대한 v(x)를 쓰면 다음과 같습니다.

$$v(x) = \sum_{i} \theta_i F_i$$

아핀 운동 모델에서 우리는

$$v(x) = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}.$$

흐름에 본질적으로 2D 효과가 포함되는 경우(이는 특히 사람 팔다리의 측면 보기에 적합함) 변환, 회전 및 일부 아핀 효과를 인코딩하는 기본 흐름 세트로 충분할 것입니다. x의 성분에 대해 (x, y)를 씁니다.

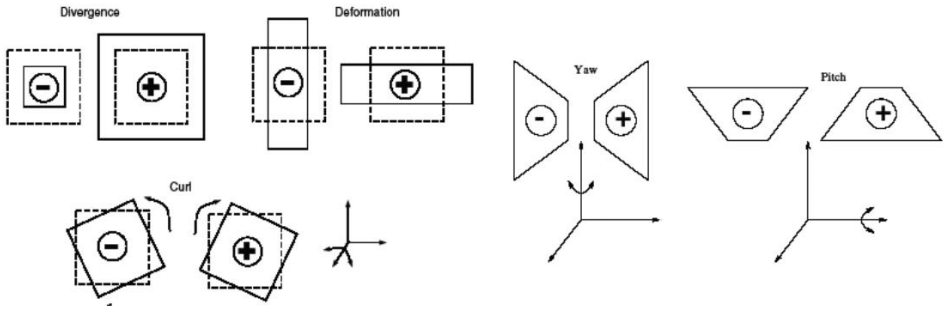

그림 10.14: 모델에 의해 생성된 일반적인 흐름 (u(x), v(x) = (θ1 + θ2x + θ3y +). θi 의 다른 값은 다른 흐름, θ7x 를 제공하동 이 2 + θ8xy, θ4 + θ5x + θ6y + θyxy + θ8y 2 며 모델은 2D 그림의 일반적인 흐름을 생성할 수 있습니다. 3D로 이미지 크기가 조정될 때 발산이 발생합니다(예: θ = (0, 1, 0, 0, 0, 1, 0, 0). 한 방향이 축소되고 다른 방향이 커질 때 변형이 발생합니다(예: 약 회전 직교 카메라의 보기 평면에 평행한 축)(예: θ = (0, 1, 0, 0, 0, 1, 0, 0) 컬은 평면 회전에서 발생할 수 있습니다. 예: θ = (0, 0, -1, 0, 1, 0, 0, 0) Yaw 모델은 원근 카메라에서 수직 축을 기준으로 회전합니다(예: θ = (0, 0, 0, 0, 0, 1, 0)). .

마지막으로 투시 카메라의 수평축을 기준으로 피치 모델이 회전합니다. 예를 들어 θ = (0, 0, 0, 0, 0, 0, 1). 이 그림은 원래 "Cardboard People: A Parameterized Model of Articulated Image Motion"의 그림 2로 S. Ju, M. Black, Y.

야콥, IEEE Int. 회의 얼굴 및 제스처, 1996 c IEEE, 1996.

간단한 모델을 사용하여 이러한 흐름을 얻을 수 있습니다.

$$v(x) = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \end{bmatrix}.$$

이 모델은 θ에서 선형이며 2D 직사각형의 3D 동작으로 인한 흐름의 합리적인 인코딩을 제공합니다(그림 10.14 참조). 또는 추적하려는 흐름 유형의 예제 풀의 특이값 분해를 통해 기본 흐름을 얻을 수 있으며 대부분의 변동을 설명하는 기본 흐름 집합을 찾으려고 시도할 수 있습니다(예: Ju et 참조). 등(1996)).

10.6.3 레이어를 사용한 모션 분할

이제 파라메트릭 흐름 모델을 사용하여 비디오 시퀀스를 분할하려고 합니다. 시퀀스에 두 개의 프레임만 있고 k 세그먼트가 있다는 것을 알고 있다고 가정합니다(그렇지 않으면 섹션 10.7의 방법을 사용하여 다른 세그먼트 수를 검색해야 합니다). 우리는 k개의 파라메트릭 흐름 모델이 혼합된 두 프레임에 대한 흐름 모델을 추정할 것입니다. 첫 번째 프레임에 있는 각 픽셀의 모션은 이 혼합에서 나오며 픽셀을 어떤 위치로 가져갈 것입니다.

그림 10.15: MPEG 꽃밭 시퀀스의 프레임 1, 15 및 30. 모션 분할 알고리즘을 시연하는 데 자주 사용됩니다. 이 시퀀스는 나무가 집보다 카메라에 훨씬 더 가깝고 꽃밭이 지면에 있는 변환 카메라에서 가져온 것처럼 보입니다. 결과적으로 나무는 프레임 전체에서 빠르게 이동하고 집은 느리게 이동하는 것처럼 보입니다. 비행기는 아핀 모션 필드를 생성합니다. 이 그림은 원래 J. Wang과 EH Adelson, IEEE Transactions on Image Processing, 1994, c IEEE, 1994에 의해 "Representing moving images with layer"에서 그림 6으로 게시되었습니다.

동일한 밝기 값을 가질 것으로 예상되는 두 번째 프레임의 픽셀입니다.
각 픽셀을 흐름 모델에 할당하여 첫 번째 이미지(또는 두 번째 이미지, 흐름 모델이 있는 경우에는 문제가 되지 않음)를 분할할 수 있으므로 흐름이 첫 번째 모델에서 나온 픽셀은 세그먼트 1에 있게 됩니다. 등등. 이 모델은 고유하고 내부적으로 일관된 모션 필드 세트를 흐름 모델당 하나씩 캡슐화합니다. 예를 들어 깊이가 다른 일련의 단단한 물체와 움직이는 카메라에서 가져올 수 있습니다(그림 10.15). 별도의 모션 필드는 종종 레이어라고 하며 모델은 레이어드 모션 모델이라고 합니다.

한 쌍의 이미지가 주어지면 (a) 픽셀이 속한 모션 필드와 (b) 각 필드의 매개변수 값을 결정하고자 합니다. 이 모든 것은 우리가 첫 번째를 알면 두 번째가 쉬울 것이고 두 번째를 알면 첫 번째가 쉬울 것이라는 점에서 처음 두 가지 예와 상당히 비슷해 보일 것입니다. 이것은 다시 누락 데이터 문제입니다. 누락 데이터는 픽셀이 속한 모션 필드이고 매개변수는 각 필드의 매개변수와 혼합 가중치입니다.

문제를 해결하기 위해 우리는 관찰의 확률적 모델도 필요합니다. 이미지 2의 픽셀 강도는 이미지 1의 픽셀을 해당 픽셀의 흐름 화살표를 따라 이동한 다음 (x, y) 첫 번째 이미지에서 매개변수가 θl 인 l번째 모션 필드에 속합니다 . 이것은 이 픽셀 이 두 번째 프레임에서 (x, y) + v(x, y; θl) 로 이동하여 이 두 픽셀의 강도가 측정 노이즈까지 동일하다는 것을 의미합니다 . 2 . 이제 가정

x, y번째 픽셀 등에서 첫 번째 이미지의 이미지 강도에 대해 I1(x, y)를 씁니다 . 누락된 데이터는 픽셀이 속한 모션 필드입니다. 이를 지표 변수 Vxy,j로 나타낼 수 있습니다. 여기서

$$\text{아버지, j} = \begin{cases} 1, & \text{x, y번째 픽셀이 j번째 모션 필드에 속하는 경우} \\ 0, & \text{그렇지 않은 경우} \end{cases} .$$

전체 데이터 로그 우도는 다음과 같습니다.

$$L(V, \Theta) = - \sum_{xy,\,j} V_{xy,j} \frac{(I1(x, y) \quad I2(x + v1(x, y; \theta j), y + v2(x, y; \theta j)))2}{2초 2} + 씨,$$

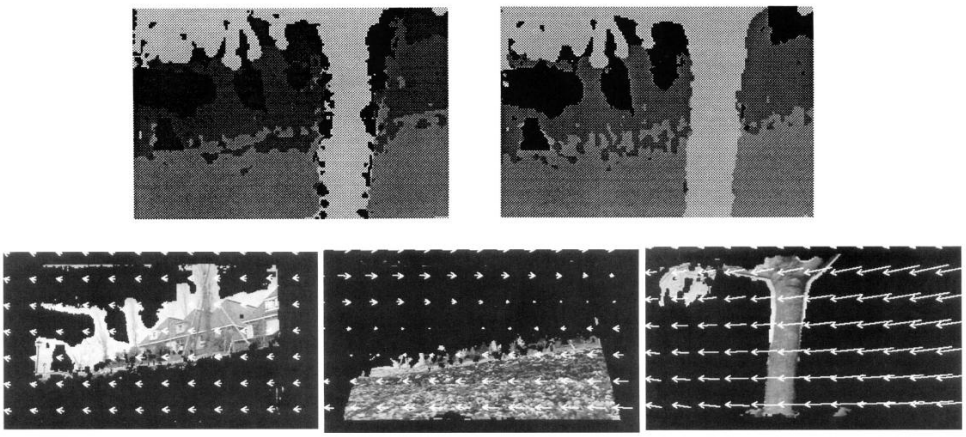

그림 10.16: 왼쪽 상단에는 이미지 움직임의 로컬 추정치를 클러스터링하여 얻은 화원 시퀀스의 프레임에서 레이어 픽셀이 속하는 것을 나타내는 amap이 있습니다.
각 그레이 레벨은 레이어에 해당하며 각 레이어는 서로 다른 아핀 모션 모델로 움직입니다. 이 지도는 픽셀 이웃의 움직임이 미래 및 과거 프레임의 이웃과 일치하는 정도를 확인하여 오른쪽 상단에 지도가 표시되도록 개선할 수 있습니다. 세 개의 레이어와 해당 모션 모델이 하단에 표시됩니다. 이 그림은 원래 J. Wang과 EH Adelson, IEEE Transactions on Image Processing, 1994, c IEEE, 1994에 의해 "Representing moving images with layer"에서 그림 11 및 12로 게시되었습니다.

여기서 Θ = (θ1,..., θk). 여기에서 EM 알고리즘을 설정하는 것은 간단합니다. 이전과 마찬가지로 결정적인 문제는 결정하는 것입니다.

$$P\{Vxy,j = 1|I1, I2, \Theta\}.$$

이러한 확률은 종종 각 픽셀에 최대 확률 계층을 나타내는 회색 수준을 할당하는 지원 맵(그림 10.16)으로 표현됩니다.

계층화된 모션 표현은 여러 가지 이유로 유용합니다. 첫째, "동일한 방식으로" 움직이는 포인트를 함께 클러스터링합니다. 둘째, 모션 경계를 노출합니다. 마지막으로, 흥미로운 방식으로 계층에서 새로운 시퀀스를 재구성할 수 있습니다(그림 10.17).

10.7 모델 선택: 어떤 모델이 가장 적합합니까?

지금까지 모델에 포함된 구성 요소의 수를 알고 있다고 가정했습니다. 예를 들어, 우리는 단일 라인을 피팅한다고 가정했습니다. 이미지 분할 예에서 세그먼트 수를 알고 있다고 가정했습니다. 일반 혼합 모델의 경우 성분의 수가 알려져 있다고 가정했습니다. 일반적으로 이것은 안전한 가정이 아닙니다.

구성 요소(예: 선, 세그먼트 등)의 수가 다른 모델을 적합하고 어떤 모델이 가장 적합한지 확인할 수 있습니다. 더 많은 구성 요소가 있는 모델이 항상 가장 적합하기 때문에 이 전략은 실패합니다. 극단적인 경우에 점에 대한 선의 정말 좋은 맞춤은 한 쌍의 점을 통과하는 하나의 선을 포함합니다. 이 표현은 데이터에 완벽하게 맞지만 거의 모든 경우에 쓸모가 없습니다.

그림 10.17: 레이어 측면에서 모션을 나타내는 한 가지 기능은 일부 레이어 없이 모션 시퀀스를 재구성할 수 있다는 것입니다. 이 예에서 MPEG 가든 시퀀스는 트리 레이어가 생략된 상태로 재구성되었습니다. 왼쪽 그림은 프레임 1, 가운데 그림은 프레임 15, 오른쪽 그림은 프레임 30을 보여줍니다.

이 그림은 원래 J. Wang과 EH Adelson, IEEE Transactions on Image Processing, 1994, c IEEE, 1994에 의해 "Representing moving images with layer"에서 그림 13으로 게시되었습니다.

사례. 조작하기에 너무 복잡하고 새로운 데이터를 예측하는 데 매우 열악하기 때문에 쓸모가 없습니다.

이 점을 보는 또 다른 방법은 편향과 분산 사이의 절충안입니다.
데이터 포인트는 우리가 나타내려고 하는 일부 기본 프로세스에서 가져온 샘플입니다. 예를 들어 한 줄로 많은 데이터 포인트를 나타내는 것은 데이터 세트를 생성한 모델의 모든 복잡성을 나타낼 수 없기 때문에 편향된 표현입니다. 기본 프로세스에 대한 일부 정보는 불가피하게 손실됩니다. 그러나 약간의 주의를 기울이면 데이터 포인트를 나타내는 데 사용되는 선의 속성을 매우 정확하게 추정할 수 있으므로 우리가 맞는 모델의 추정치에는 거의 차이가 없습니다. 또는 데이터 요소를 연결하는 지그재그 선 집합으로 데이터 요소를 나타내는 경우 표현에는 편향이 없지만 동일한 소스의 새로운 데이터 요소 샘플마다 다를 수 있습니다. 결과적으로 우리가 적합하는 모델의 추정치는 샘플마다 크게 바뀝니다. 그것은 분산에 의해 압도됩니다.

우리는 절충안을 원합니다. 피팅 오차는 매개변수의 개수에 따라 작아지므로 구성 요소의 개수에 따라 증가하는 피팅 오차에 항을 추가해야 합니다. 이 페널티는 증가하는 매개변수 수로 인해 발생하는 피팅 오류(동일하게 음의 로그 우도) 감소를 보상합니다. 대신, 우리는 다양한 기법 중에서 선택할 수 있으며, 각 기법은 서로 다른 극한 원리와 기준의 서로 다른 대략적인 추정치에 따라 서로 다른 할인을 사용합니다.

이 점을 보는 또 다른 방법은 모델에서 미래 샘플을 예측하려는 것입니다. 데이터 세트는 모델 계열의 구성원인 파라메트릭 모델의 샘플입니다. 매개변수를 적절히 선택하면 모델(테스트 세트)과 데이터 세트(종종 훈련 세트라고 함)에서 향후 샘플을 예측할 수 있습니다. 안타깝게도 이러한 향후 샘플은 사용할 수 없습니다. 또한, 데이터 세트를 사용하여 얻은 모델 매개변수의 추정치는 선택된 매개변수가 모델이 가능한 전체 데이터 세트가 아니라 훈련 세트에 최적임을 보장하기 때문에 편향될 가능성이 있습니다. 그 효과를 선택 편향이라고 합니다. 학습 세트는 모델에서 가져올 수 있는 전체 데이터 세트의 하위 집합입니다. 무한정인 경우에만 모델을 정확하게 나타냅니다.

크기가 큰. 이것이 음의 로그 우도가 모델 선택에 대한 잘못된 지침인 이유입니다. 점점 더 편향되기 때문에 피팅이 더 좋아 보입니다.

이제 최적의 매개변수 선택을 Θ 로 , 데이터 세트에 대한 적합의 로그 우도를 L(x; Θ), 자유 매개변수의 수는 p, 데이터 항목의 수는 N으로 씁니다. 우리는 로그 우도로부터 점수를 계산하고 너무 많은 매개변수를 권장하지 않는 페널티를 계산할 것입니다. 점수에는 몇 가지 가능성이 있지만 절차에는 이 점수를 최적화하는 모델을 찾기 위해 모델 공간을 검색하는 작업이 포함됩니다(예: 구성 요소 수를 늘릴 수 있음).

AIC: 정보 기준

Akaike는 널리 알려진 AIC("Akaike 정보 기준"이 아니라 "정보 기준")라고 하는 페널티를 제안했습니다.

$$2L(x; \Theta) + 2p.$$

AIC에 대한 통계적 논쟁 모음집이 있습니다. 첫 번째 요점은 데이터 포인트 수에 용어가 없다는 것입니다. 데이터 포인트의 수가 증가함에 따라 실제 모델의 매개변수에 대한 추정치가 더 좋아져야 하므로 이는 의심스럽습니다. 둘째, AIC가 과대적합하는 경향이 있는 경험이 있습니다. 즉, 훈련 세트에는 잘 맞지만 테스트 세트에서는 잘 수행되지 않는 매개변수가 너무 많은 모델을 선택하는 것입니다.

베이지안 방법 및 Schwartz의 BIC

간단히 하기 위해 데이터에 D, 모델에 M, 데이터에 θ를 씁니다.
매개변수. Bayes의 규칙은 다음을 생성합니다.

$$P(M|D) = \frac{P(D|M)}{P} P(M)P(D)$$

$$= \frac{P(D|Mi\ , \theta)P(\theta)d\theta P(M)}{P(D)}.$$

이제 사후가 큰 모델을 선택할 수 있습니다. 이 사후 계산은 어려울 수 있지만 일련의 근사를 통해 기준을 얻을 수 있습니다.

$$L(D; \theta) + \frac{P}{2} \log N$$

(여기서 N은 데이터 항목의 수입니다). 다시 이 점수를 최소화하는 모델을 선택합니다. 이를 Bayes 정보 기준 또는 BIC라고 합니다. 여기에는 데이터 항목 수에 대한 용어가 있습니다.

설명 길이 모델은 본질적으로 통계

적이지 않은 기준으로 선택할 수 있습니다. 결국 우리는 모델을 선택하고 있으며 왜 그것을 선택하고 싶은지 말할 수 있습니다. 다소 자연스러운 기준은 데이터 세트를 가장 선명하게 인코딩하는 모델을 선택하는 것입니다. 이 최소 설명 길이 기준은 데이터 세트의 가장 효율적인 전송을 허용하는 모델을 선택합니다. 데이터셋을 전송하기 위해서는 모델 파라미터를 코딩하여 전송한 후 모델 파라미터가 주어진 데이터를 코딩하여 전송한다. 데이터가 모델에 적합하지 않으면 잡음과 같은 신호를 코딩해야 하기 때문에 후자의 항이 커집니다.

실제로 사용되는 기준의 도출은 오히려 우리의 필요를 넘어섭니다. 자세한 내용은 Rissanen(1983), (1987), Wallace and Freeman(1987)에 나와 있습니다. Kolmogorov로 인해 정보 이론에 뿌리를 두고 Cover와 Thomas(1991)에서 설명된 유사한 아이디어가 있습니다. 놀랍게도 BIC는 이 분석에서 도출되어 다음과 같이 산출됩니다.

$$L(D; \theta) + \log N. \frac{p}{2}$$

다시 이 점수를 최소화하는 모델을 선택합니다.

10.7.1 교차 검증을 사용한 모델 선택

모델 선택의 주요 어려움은 측정할 수 없는 양, 즉 훈련 세트에 없는 데이터를 예측하는 모델의 능력을 사용해야 한다는 것입니다. 훈련 세트가 충분히 큰 경우 훈련 세트를 두 개의 구성 요소로 분할하고 하나는 모델에 적합하고 다른 하나는 적합을 테스트하는 데 사용할 수 있습니다. 이 접근 방식을 교차 검증이라고 합니다.

교차 유효성 검사를 사용하여 데이터 세트를 교육 및 테스트 데이터로 분할하고 다양한 모델을 교육 데이터에 맞춘 다음 테스트 데이터에서 가장 잘 수행되는 모델을 선택하여 모델의 구성 요소 수를 결정할 수 있습니다. 성능을 평가하기 위해 테스트 데이터의 로그 우도를 살펴볼 수 있습니다. 매개변수가 너무 많은 모델은 훈련 데이터 세트에 잘 맞지만 테스트 세트를 잘못 예측하기 때문에 이 프로세스가 구성 요소 수를 추정할 것으로 예상합니다.

두 구성 요소로 분할된 단일 선택 항목을 사용하면 다른 형태의 선택 편향이 도입되며 가장 안전한 방법은 이러한 모든 분할에 대한 추정치를 평균화하는 것입니다. 분할 수가 엄청나기 때문에 테스트 세트가 크면 다루기 어려워집니다. 가장 일반적인 버전은 leave-one-out 교차 검증입니다. 이 접근법에서 우리는 훈련 세트의 각 N-1 세트에 모델을 맞추고 나머지 데이터 포인트에 대한 오류를 계산하고 이러한 오류를 합산하여 모델 오류의 추정치를 얻습니다. 그런 다음 이 추정치를 최소화하는 모델이 선택됩니다.

10.8 참고 사항

최소 제곱 피팅의 기원은 가우스 자신이 방법을 발명했다고 믿지만 우리에게는 불투명합니다. 총 최소 제곱은 Deming(1943)에 의한 것으로 보입니다. 최소 제곱법 또는 근사법을 사용하여 곡선 또는 곡면을 피팅하는 방법에 대한 많은 문헌이 있습니다(원뿔형에 대한 작업으로 시작할 수 있음(Bookstein 1979, Fitzgibbon et al. 1999, Kanatani 2006, Kanatani 1994, Porrill 1990, Sampson 1982) ; (Taubin 1991))에서 더 복잡한 문제.

허프 변환

Hough 변환은 Hough(1962)에 기인합니다(Keith Price의 훌륭한 참고 문헌 설명: "가장 많이 인용되고 가장 적게 읽은 참조"). Hough 변환에 대한 많은 문헌이 있었는데, 이는 이론적 의미가 있는 것으로 여겨졌습니다. 관심 있는 사람은 Ballard(1981)로 시작한 다음 Ballard(1984)로 시작할 수 있습니다.

그런 다음 주제가 구식으로 보이기 시작했지만 평균 이동 방법과 Maji 및 Malik(2009)의 관찰로 인해 모든 토큰이 동일한 표를 가질 필요는 없으며 가중치를 학습할 수 있다는 관찰에 의해 부활했습니다. 여러 조각이라는 생각

객체의 위치에 투표함으로써 서로를 강화할 수 있는 객체의 위치는 매우 오래된 것입니다(Ballard(1981); Ballard(1984) 참조). 가장 중요한 최신 버전은 Bourdev et al. (2010).

랜삭

RANSAC은 구현 및 사용이 매우 쉽고 매우 효과적인 매우 중요한 알고리즘입니다. 원본 논문(Fischler and Bolles 1981)은 여전히 읽을 가치가 있습니다.
데이터와 문제에 대해 무엇을 알고 있는지에 따라 다양한 변형이 있습니다. Torr 및 Davidson(2003) 및 Torr 및 Zisserman(2000)을 참조하십시오.

EM 및 결측변수 모델

EM은 Dempster et al.에 의해 통계 문헌에 처음 공식적으로 설명되었습니다.
(1977). 아주 좋은 요약 참조는 수많은 변형을 설명하는 McLachlan and Krishnan(1996)입니다. 예를 들어 Q(u;u (s)) 의 최대값을 찾을 필요가 없습니다 . 필요한 것은 더 나은 가치를 얻는 것입니다. 또 다른 예로 확률적 통합 방법을 사용하여 기대치를 추정할 수 있습니다.

누락된 변수 모델은 모든 종류의 장소에서 발생하는 것 같습니다. 컴퓨터 비전에서 우리가 알고 있는 모든 모델은 혼합 모델에서 발생하므로(따라서 누락된 변수에서 선형인 완전한 데이터 로그 우도를 가짐) 이 경우에 집중했습니다. 세분화를 위해 누락 변수 모델을 사용하는 것은 자연스러운 일입니다(예를 들어 Belongie et al.(1998a), Feng 및 Perona(1998), Vasconcelos 및 Lippman(1997), Adelson 및 Weiss(1996) 또는 Wells et al. (1996)). 다양한 형태의 레이어드 모션이 현재 존재합니다(Delaert et al.(2000); Wang and Adelson(1994); Adelson and Weiss(1996); Tao et al.(2000); Weiss(1997) 참조). 또한 동일한 깊이에 있는 층을 구성하거나 (Brostow 및 Essa(1999); Torr 등(1999b); 또는 Baker 등(1998) 참조) 다른 공통 속성을 가질 수 있습니다. 다른 흥미로운 사례로는 투명도, 반사성 등으로 인한 움직임이 있습니다(Darrell 및 Simoncelli(1993), Black 및 Anandan(1996), Jepson 및 Black(1993), Hsu 등(1994) 또는 Szeliski 등(2000)). 결과 표현은 상당히 효율적인 이미지 기반 렌더링에 사용될 수 있습니다(Shade et al.(1998) 참조).

EM은 매우 성공적인 추론 알고리즘이지만 마술은 아닙니다. 우리가 설명한 종류의 문제에 대한 주요 어려움 원인은 지역 최대값입니다. 매우 많은 수의 결측 변수가 있는 문제가 많은 수의 로컬 최대값을 갖는 것이 일반적입니다. 이는 정답에 가까운 최적화를 시작하여 처리할 수 있으며, 이는 요점을 놓치는 것입니다.

모델 선택

모델 선택은 마땅히 받아야 할 만큼 많은 관심을 받지 못한 주제입니다.
중요한 작업이 진행 중이며 문제는 어떤 카메라 모델(또는 영상, 원근법 등)을 적용할지입니다(Torr(1999), Torr(1997), Kinoshita 및 Lindenbaum(2000) 또는 Maybank 및 Sturm(1999) 참조)). 유사하게, 범위 데이터의 분할에 대한 작업이 있는데, 여기서 질문은 데이터가 맞춰져야 하는 파라메트릭 표면 세트(즉, 2개의 평면이 있는지 3개의 평면이 있는지 등)입니다(Bubna and Stewart 2000). 재구성 문제에서 때때로 다음 여부를 결정해야 합니다.

축퇴된 카메라 모션 시퀀스가 존재합니다(Torr et al. 1999a). 분할의 표준 문제는 얼마나 많은 세그먼트가 존재하는지입니다(Raja et al.
(1998); Belongiet al. (1998a); 및 Adelson 및 Weiss(1996)). 모델을 예측적으로 사용하는 경우 모델 예측보다 가중 평균을 계산하는 것이 때때로 더 좋습니다(실제 베이지안은 모델 선택을 수행하지 않음)(Torr and Zisserman 1998, Ripley 1996). 사용 가능한 방법 중 일부만 설명했습니다. 한 가지 중요한 누락은 Kanatani의 기하학적 정보 기준입니다(Kanatani 1998).

문제

10.1. 간단하지만 매우 유용한 결과인 수직 거리 2 2 + b = 1을 증명하십시오.
점 (u, v)에서 toaline (a, b, c)는 abs(au + bv + c)로 주어진다.

10.2. 고유값 문제 도출

$$\begin{matrix} \overline{2\,x - xx} & \overline{xy - xy} \\ \overline{2\,xy - xyy} & \overline{yy} \end{matrix} \begin{matrix} \overline{\quad} & \overline{\quad} \\ \overline{\quad} & \overline{\quad} \end{matrix} \begin{matrix} \text{ㅏ} \\ \text{비} \end{matrix} = \mu \begin{matrix} \text{ㅏ} \\ \text{비} \end{matrix}$$

총 최소 제곱에 대한 생성 모델에서. 이것은 간단한 연습입니다. 가능성을 최대화하고 약간의 조작만 하면 됩니다. 하지만 올바르게 수행하고 기억할 가치가 있습니다. 이 기술은 매우 유용합니다.

10.3. 반환하는 에지 검출기에서 에지 포인트의 곡선을 얻는 방법
정위? 재귀 알고리즘을 제공하십시오.

10.4. 암시적 곡선은 ϕ(x, y) = 0으로 제공됩니다. 이 곡선의 경우 모든 점(u, v)에 대해 곡선에 정확히 한 점(x0, y0) 이 로컬 가까운 방정식. 이 곡선이 직선임을 보여라. (힌트: 곡선의 각 점에서 법선은 평행해야 합니다.)

10.5. 증분 맞춤의 약간 더 안정적인 변형은 선을 맞출 때 선 점 목록에서 처음 몇 픽셀과 마지막 몇 픽셀을 자릅니다. 이러한 픽셀은 모서리에서 왔을 수 있기 때문입니다. (ㄱ) 이것이 개선으로 이어지는 이유는 무엇입니까? (b) 생략할 픽셀 수를 어떻게 결정해야 합니까?

10.6. 초점 거리가 f인 고정 카메라가 있다고 가정합니다. 월드 포인트의 좌표는 대문자로, 이미지 포인트의 좌표는 소문자로 쓴다. 카메라의 초점을 (0, 0, 0)에 놓고 이미지 평면을 Z = -f에 놓습니다. 평면 Z = aX + b에 |a|가 있는 평면 객체가 있습니다. > 0, | 비 | > 0. (a) 이 개체의 어떤 변환이 이미지에 정확히 일치하는 움직임 필드를 제공합니까?

아핀 모션 모델로 표현?

(b) 어떤 상황에서 아핀 모션 모델이 이 객체를 변환하여 생성된 이미지 흐름 필드에 근사할 수 있는 이유를 제공합니까?

10.7. 선 및 이상값 예제에 대한 표기법은 섹션 10.5.1을 참조하십시오. i번째 예의 표시 변수에 대해 δi를 씁니다 . 예가 행에서 나온 경우 δi = 1이고 그렇지 않은 경우 δi = 0입니다. 전체 최소 제곱을 사용하여 피팅하려고 한다고 가정합니다. 오류의 표준 편차인 σ를 알고 있다고 가정합니다.

이상값의 확률은 해당 위치와 무관하다고 가정합니다. 이 예에서 전체 데이터 로그 우도가 다음임을 보여줍니다.

$$LC(a, b, c, \pi) = \text{ㄴ} - \frac{(\text{축} + \text{바이i} + c)^2}{2\sigma 2} + \log \pi\, \delta i + [K + \log(1 \quad \pi)](1 \quad \delta i) + L$$

여기서 K는 이상값을 얻을 확률을 나타내는 상수이고 L은 a, b, c 또는 π에 의존하지 않습니다.

10.8. 선 및 이상값 예제에 대한 표기법은 섹션 10.5.1 및 이전 예제를 참조하십시오. 이 예에서는 αi = Eδ|x,Θ(s) [δi] 에 대한 표현식을 생성합니다 . 여기서 Θ = (a, b, c, π)입니다.

10.9. 이미지 분할 예제에 대한 표기법은 섹션 10.5.1을 참조하십시오. i번째 예의 표시 변수에 대해 δij를 씁니다 . 여기서 예가 j번째 세그먼트에서 온 경우 δij = 1이고 그렇지 않은 경우 δij = 0입니다. 이미지 세그먼트 확률 분포의 공분산인 Σ를 알고 있고 이것이 세그먼트별로 동일하다고 가정합니다. 이 예에서 전체 데이터 로그 가능도가 다음임을 보여줍니다.

$$Lc(\pi 1, \ldots, \pi g, \mu 1, \ldots, \mu g) = \sum_{ij} \left[-\frac{(xi - \mu j)^{T} \Sigma^{-1} (xi - \mu j)}{2} + \log \pi j \right] \delta ij + L$$

여기서 L은 μj 또는 πj 에 의존하지 않습니다 .

10.10. 이미지 분할 예제에 대한 표기법은 섹션 10.5.1을 참조하십시오. 을 위한
이 예에서는 αij = Eδ|x,Θ(s) [δij] 에 대한 표현식을 생성합니다 . 여기서 Θ=(π1,..., πg, μ1,...,μg)입니다.

10.11. 선 및 이상값 예제에 대한 표기법은 섹션 10.5.1을 참조하십시오. 이를 위해
예를 들어, M 단계에서 생성된 업데이트가

$$\pi(s+1) = \frac{\sum_{i} \alpha ij}{\sum_{i,j} \alpha ij}$$

그리고

$$\mu_j^{(s+1)} = \frac{\sum_{i} \alpha ij xi}{\sum_{i} \alpha ij}$$

10.12. 이미지 분할 예제에 대한 표기법은 섹션 10.5.1을 참조하십시오. 을 위한
이 예에서는 M 단계에서 생성된 업데이트가

$$\text{파이}_{\text{새}}^{(s+1)} = \frac{\sum_{i} \alpha ij}{\sum_{i,j} \alpha ij}$$

그리고

$$\mu_j^{(s+1)} = \frac{\sum_{i} \alpha ij xi}{\sum_{i} \alpha ij}$$

프로그래밍 실습

10.13. 증분 라인 피팅을 구현합니다. 라인 포인트 목록에서 처음 몇 픽셀과 마지막 몇 픽셀을 제외하여 얼마나 중요한 차이가 발생하는지 확인합니다(이것을 구축할 때 약간의 주의를 기울이십시오. 우리의 경험에 따르면 이것은 유용한 소프트웨어입니다).

10.14. Hough 변환 기반 라인 찾기를 구현합니다.

10.15. HT 라인 파인더로 라인을 센다. 얼마나 잘 작동합니까?

10.16. 증분 라인 피팅을 위한 알고리즘을 구현합니다.

10.17. 이미지 분할 예제에 대한 표기법은 섹션 10.5.1을 참조하십시오. αij = Eδ|x,Θ(s) [δij](여기서 Θ = (π1,..., πg, μ1,...,μg)) 에 대한 식을 사용하여 EM 알고리즘을 구현하여 이미지를 분할합니다. RGB 색상과 위치를 특징 벡터로 사용하면 충분합니다.