Project 1. HMM Modeling using Character Trajectories Data Set

Sangjun Son
Department of Computer Science and Engineering, Seoul National University

1. Introduction

은닉 마르코프 모델 (Hidden markov model, 이하 HMM)은 순차적인 데이터를 다루는 데 강점을 지녀 개체명 인식, 포스태깅 등 단어의 연쇄로 나타나는 언어구조 처리에 과거 많은 주목을 받았던 기법입니다[1]. 마르코프 과정 (Markov process)에 기반하여 상태들이 특정한 확률 분포에 따라 결과들을 추론하는 특성에 맞게 HMM은 직접 관측할 수 없는 데이터 열과 이에 의존하는 다른 데이터들이 있을 때 데이터 열을 복원하는 것을 목표로 하는 다양한 분야에 적용 가능합니다[2]. 예를 들어 다중 서열 정렬, 단백질 2차 구조 예측과 같은 유전자 서열 분석[3]과 음성 인식 (Speech to Text)에도 사용되고 있습니다.

HMM 모델링을 위해 학습에 사용할 데이터의 특성은 시간에 따라 측정되어야 한다 (Time-serial)는 점입니다. 이번 프로젝트에서는 사람의 생물학적인 움직임을 기록한 여러가지 데이터 중에 가장 단순한 형태의 분류 가능한 필기체 데이터 (Character trajectories data set)[4]을 사용합니다. 총 20개의 알파벳에 대해 필기체로 작성한 2858개의 샘플 데이터가 있으며 각각 펜의 좌표 (x, y)와 펜 촉의 세기를 측정하여 기록하였습니다[5]. 2858개 중 90%인 2572개를 무작위 샘플링하여 학습을 위한 데이터 (Train data set)로, 나머지 10%인 286개를 검증을 위한 데이터 (Test data set)로 모델링과 실험검증을 진행하였습니다.

2. Related works

특성 추출 (Feature extraction)은 패턴 인식 (Pattern recognition)에 꼭 필요한 요소입니다. 필기체 데이터를 HMM에 적용하기 위해서 새로운 함수 (Metric)를 정의할 필요가 있습니다. 변위 데이터의 연속성 때문에 HMM의 이산 모델 (Discrete model)을 적용할 수 없기 때문입니다. 따라서 관측치를 이산화하면서 필기체 알파벳을 특성 함수 (Feature metric)에 제대로 표현하기 위해서는 여러 함수를 설정하고 검증하는 시도가 필요할 것입니다.

필기체 인식모델 사례 중 알파벳이 아닌 숫자에 대해 94%의 정확도를 보여주는 HMM 모델이 있습니다[6]. 가속도 측정기 (Accelerator)를 이용하여 펜 촉의 시간별 가속도를 측정하였고 2가지 방식을 소개합니다. Trajectory Orientation (TO)에서 가속도를 동경으로 표현하였고 6개의 영역으로 나누어 해당하는 위치의 번호를, Curve Feature (CF)에서는 가속도의 변화량을 특성 함수로 설정하였습니다. 각각의 특성 함수를 HMM의 관측치로 넣어 주어 학습하는 모델입니다.

3. Preliminaries

1) Hidden Markov Model

HMM은 각 상태가 마르코프 체인 (Markov Chain)을 따르되 은닉되어 있다고 가정합니다. 하지만 주어진 모델 λ에 대한 정보는 관측치 O외에는 없습니다. 이를 토대로 은닉되어 있는 상태 (Hidden state)를 추론하게 됩니다. 은닉 상태끼리의 전환되는 확률은 전이확률 (Transition probability) A라 하며 각 상태에서 O를 관측할 확률을 방출확률 (Emission probability) B라고 합니다. 또한 우리 모델에서 알파벳 라벨을 결정짓는데 쓰이는 우도 (Likelihood) $p(O \mid \lambda)$는 모델 λ이 주어졌을 때 여러 관측치를 임의로 표현하였는데 그 때 관측치 O가 뽑힐 확률을 의미합니다.

우도를 구하기 위해서는 전방확률 (Forward probability)과 후방확률 (Backward probability)을 이용하여 계산할 수 있습니다. 전방확률 $\alpha_t(j)$이란 모델 λ가 주어졌을 때 j번째 상태와 $o_1, ..., o_t$가 나타날 확률을 의미하고 같은 맥락으로 후방확률 $\beta_t(j)$를 j번째 상태와 $o_{t+1}, ..., o_T$가 나타날 확률로 정의할 수 있습니다.

$$\begin{aligned}\alpha_t(j) &= P(o_1, ..., o_t, q_t = j \mid \lambda) \\ &= \sum_{i=1}^{n} \alpha_{t-1}(i) \times a_{ij} \times b_j(o_t)\end{aligned}$$

$$\begin{aligned}\beta_t(j) &= P(o_{t+1}, ..., o_T, q_t = j \mid \lambda) \\ &= \sum_{i=1}^{n} \beta_{t+1}(i) \times a_{ij} \times b_j(o_{t+1})\end{aligned}$$

$$p(O \mid \lambda) = \sum_{s=1}^{n} \alpha_t(s) \times \beta_t(s)$$

2) EM Algorithm

HMM의 변수는 전이확률 A와 방출확률 B이며 이들을 관측치 O를 토대로 업데이트하는 알고리즘이 바로 EM 알고리즘 (Expectation-maximization algorithm)입니다. 다른 용어로는 'Baum-Welch' 또는 'Forward, Backward'이 통용됩니다.

E-step: 모델 λ를 고정시킨 상태에서 관측치 O를 바탕으로 전방확률 α와 후방확률 β 업데이트
M-step: *E*-step에서 구한 전방확률 α와 후방확률 β를 바탕으로 전이확률 A, B를 업데이트

4. Proposed Method

*vel4acc6iter1*에 대한 전반적인 모델링에 대해 설명합니다. 속도의 방향과 가속도의 방향을 관측값으로 하는 모델이며 DHMM (Discrete HMM)을 알파벳 레이블 별로 단일 수행합니다.

1) Data visualization & Feature extraction

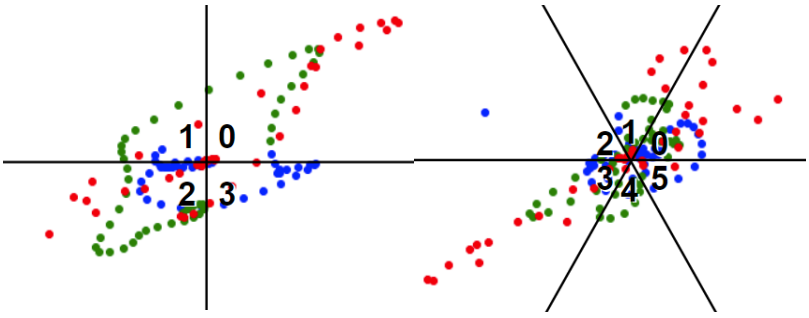

Fig 1. alphabet 'a's plots of speed (left) and acceleration (right)
w. overlays: separators of feature metric

해당 알파벳 별로 특정할 특성을 찾기 위해 속도와 가속도를 계산한 뒤 각각의 벡터에 대한 동경을 구합니다. Fig 1.과 같이 각각의 동경을 속도의 경우 4등분한 영역의 번호 N_V, 가속도의 경우 6등분한 영역의 번호 N_A로 표시합니다. 특성 함수 (Feature metric)를 $6N_V + N_A$로 선택하여 시간에 따라 어떤 관측 값 (범위: 0 ~ 23)을 표현합니다. Fig 1.에서와 같은 샘플 데이터를 이용한다면 아래와 같은 함수(metric)이 나옵니다.

```
array([ 0, 23, 22, 21, 21, 21, 22, 22, 22, 22, 15,
15, 15, 15, 14, 14, 14, 14, 13, 13, 13, 13, 7, 7,
8, 8, 9, 8, 7, 6, 6, 6, 6, 6, 0, 0, 0, 0, 5, 4, 4,
4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 3, 3, 21,
21, 21, 21, 21, 21, 20, 20, 13, 14, 14, 14, 14,
15, 15, 15, 15, 15, 13, 12, 12, 12, 17, 14, 14,
13, 12, 12, 12, 12, 12, 12, 6, 6, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 5, 4, 4, 4, 4, 22, 21, 21,
21, 15, 14, 13, 13, 13, 13, 7, 6, 11, 10, 4, 4, 4,
4, 4, 16, 15, 15, 15, 15, 15, 15, 15, 15, 15])
```

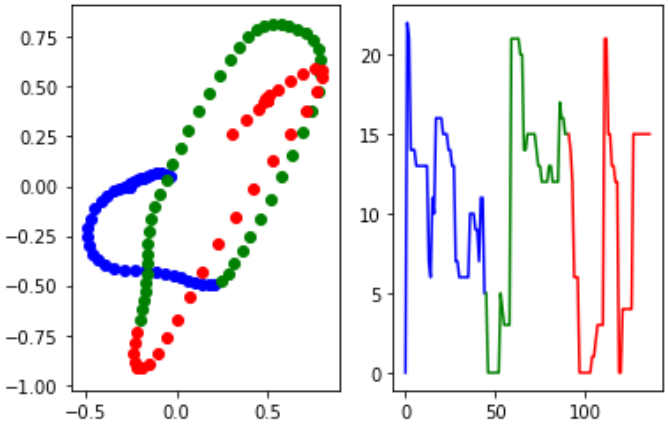

Fig 2. alphabet 'a's plots of its displacement vectors (left)
and feature metric graph according to drawing time (right)
The pen moves from blue points to red points via green points.

2) Train test set split & Discrete HMM

sklearn.model_selection 패키지를 이용해 특성 함수 (Feature metric)로 표현한 모든 2858개의 샘플을 학습과 검증 데이터로 나눕니다[7].

20개의 모든 알파벳 레이블에 대해 분류할 것이기 때문에 HMM 모델 또한 알파벳의 개수만큼 만들고 각각의 관측값에 대해 우도가 큰 HMM에 대해 레이블링이 되는 ovr 방식 (One vs rest)을 취합니다.

이산 관측값 (Discrete observations)를 바탕으로 학습을 시키기 위해서 *hmmlearn* 패키지의 *Multinomial HMM*을 사용합니다. 관측치의 다양성은 은닉 상태의 다양성을 따라가게 하기 위해서 *n_components*의 값을 특성 함수의 값의 범위로 초기화 해줍니다. 오버피팅을 막기 위해 *n_iter*의 값을 교차검증 (Cross validation)을 통해 구한 최적 값인 1로 설정해줍니다.

model	# of samples	elapsed time	model	# of samples	elapsed time
['a']	152	3.03s	['o']	128	2.16s
['b']	127	2.47s	['p']	117	2.18s
['c']	125	1.53s	['q']	117	2.49s
['d']	143	2.57s	['r']	103	1.81s
['e']	166	2.65s	['s']	119	2.20s
['g']	127	2.31s	['u']	123	2.53s
['h']	115	1.92s	['v']	133	2.33s
['l']	159	1.66s	['w']	109	1.99s
['m']	120	2.55s	['y']	125	2.22s
['n']	114	1.92s	['z']	150	3.27s

Table 1. elapsed time while training each alphabet samples

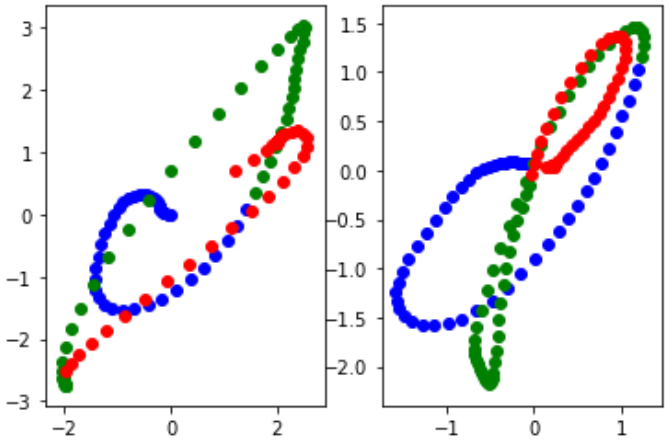

Fig 3. plots of real displacement of 'd' (left) and
an arbitrary sample of misclassified label 'q' (right)

286개의 테스트 데이터에 대해 잘못된 분류를 수행한 경우는 82개이며 정확도 (Accuracy)는 약 71.33%으로 측정되었습니다. Fig 3.은 잘못 분류된 알파벳에 대해 원래 갖고 있던 레이블과 잘못 분류한 레이블을 비교한 도식이며 복잡도가 높은 알파벳에 대해 좋은 성능을 보이지 않는 것을 판단할 수 있습니다.

Fig 3. empirical analysis for higher accuracy considering speed-accuracy tradeoffs

5. Empirical analysis

HMM 모델이름을 다음과 같이 정의하겠습니다. 특성 추출 방식(Feature extraction)에 따라 분류됩니다. 가속도 벡터 동경으로 #등분하여 분류할 경우 *acc#*, 속도 벡터 경우 *vel#*, 우리의 Proposed Method와 같이 속도 벡터를 #등분하고 가속도는 *$*등분하여 분류하는 경우 *vel#acc$*라고 칭하겠습니다. 또한 HMM의 종료조건을 결정하는 반복 횟수 (Iteration)을 #번 한다면 모델 이름의 마지막에 *iter#*을 붙이도록 하겠습니다.

1) Iteration Variation (*acc6*)

가속도 벡터의 동경을 6분할로 하고 HMM의 학습인자 (Learning parameter) 중 반복 횟수를 상용 로그 스케일로 늘려가며 실험을 진행하였습니다. 반복횟수가 커질수록 시간은 늘어나고 정확도는 증가할 것을 예상하였으나 반대로 정확도가 감소하는 것을 확인였습니다. 오히려 언더피팅 문제가 생각보다 빨리 해결되어 오버피팅이 일어나지 않도록 반복횟수를 최소화 하였습니다.

2) Acceleration Variation (*iter1*)

오버피팅이 빨리 일어난다는 것을 보고 모델의 복잡도를 늘려야겠다는 생각을 하였고 가속도 벡터의 동경을 더 많이 쪼개어 실험을 진행하였습니다. 예상과 마찬가지로 모델이 커질수록 시간과 정확도가 증가하는 추세를 보여주었습니다. 하지만 아무리 높여도 목표성능인 50%를 넘진 못하였습니다. 가속도의 동경만 가지고는 모델의 복잡도를 따라갈 수 없으며 새로운 특성 함수 (Feature metric)을 도입해야 했습니다.

3) Speed Variation (*iter1*)

Acceleration Variation과 마찬가지로 속도 벡터의 동경의 분할 수를 늘려 모델 복잡도를 늘려보았습니다. 실행시간은 비슷하였으나 예상 외로 가속도 보다 10% 가량의 높은 정확도를 보였으며 끝내 12, 16, 24분할로

쪼갠 *vel12iter1*, *vel16iter1*, *vel24iter1*가 목표 성능(정확도 50%)에 도달하였습니다.

4) Iteration Variation (*vel4acc6*)

Acceleration Variation과 Speed Variation을 섞어서 특성을 추출하게 되면 더 복잡한 모델로 학습하지 않을까하는 생각에 실험을 진행하였습니다. 관측치 범위가 너무 커지지 않도록 하면서 어느 정도는 복잡하게 만들게 된 *vel4acc6iter1* 모델의 학습 결과는 70%의 정확도를 보여주었습니다. 반복횟수를 늘려가며 새로운 모델을 만들었으나 마찬가지로 정확도는 높아지지 않았습니다.

6. Conclusion

필기체로 쓰여진 글자를 시간에 따른 변위가 주어졌을 때 가속도의 방향과 속도의 방향을 결합하여 관측을 하였을 때 새로운 관측치가 들어왔을 때에도 정확도(우도)가 높아지는 것을 볼 수 있었습니다. 실제로 변위를 찍어 필기체 데이터를 그래프에 도식화하여 눈으로 보고 알파벳이 무엇인지 분류하기 힘든 데이터가 많았습니다. 하지만 HMM을 사용하여 육안으로도 구분하기 힘든 데이터에 대해 은닉 상태를 찾아 학습을 하여 높은 분류 성능을 낸다는 점을 확인할 수 있었습니다.

알파벳 필기체 데이터를 사용하면서 라벨을 결정짓는데 특성 추출 (Feature extraction)이 모델을 결정짓는 주요 요인으로 작용하였습니다. 현재까지 학습한 모델 중에 가장 높은 성능의 *vel4acc6iter1*는 테스트 데이터에 대해 70%의 정확도를 보여주었고 잘못 분류된 나머지 30%의 데이터는 특정 알파벳에 대해 특히 많은 분포를 가진다는 것을 도출해 낼 수 있었습니다. 특히 ‘m’, ‘n’, ‘h’, ‘p’, ‘g’와 같이 복잡한 알파벳처럼 특징을 확실히 구분하지 못한다는 점에서 미루어 보아 모델의 복잡도를 높여야 될 것입니다.

현 모델에서 펜 촉의 세기를 사용하지 않고 펜의 변위를 바탕으로 속도와 가속도를 구하고 방향성만 가지고 특성 함수 (Feature metric)을 구성하였다는 점이 좋은 결과를 가져오지 못하였습니다. Related works 언급한 TO/CF 모델처럼 가속도 변화량까지 고려하거나 실제 펜 촉의 세기나 가속도 크기를 구간화하여 특성 함수 (metric)로 표현한다면 시간 - 정확도 절충안 (Speed-accuracy tradeoff)를 찾을 수 있을 것입니다.

Reference

[1] ratgo’s blog, 은닉마코프모델
[2] Wikipedia, Hidden Markov model
[3] Byung-Jun Yoon, Hidden Markov Models and their Applications in Biological Sequence Analysis
[4] B.H. Williams, M.Toussaint, and A.J. Storkey. Extracting motion primitives from natural handwriting data. In ICANN, volume 2, pages 634–643, 2006.
[5] UCI Machine Learning Repository, Character Trajectories Data Set
[6] Chun Yuan. et al, A Handwritten Character Recognition System Based on Acceleration, January 2011
[7] API Documentation of sklearn.model_selection.train_test_split (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)
[8] API Documentation of hmmlearn.MultinomialHMM (https://hmmlearn.readthedocs.io/en/latest/api.html#multinomialhmm)