Natural Language Processing: Project 1

Seung-Hoon Na

April 13, 2024

1 Hidden Markov Model

본 주제에서는 hidden Markov model (HMM)을 위한 이론적인 내용들을 정리하고 numpy로 구현해보는 것을 목표로 한다.

$\mathbf{x} = \{x_1, \cdots, x_n\}$를 길이가 n인 **단어열**, 즉 n개의 연속된 단어(또는 토큰)으로 구성된 관측열 (**observation sequence**)이라고 하자. 여기에서는 $\mathbf{x}$는 "단어" 열을 의미하고 있으나, 일반적으로 순차성을 지니는 임의의 시퀀스가 될 수 있고, 예를 들면, 날짜별 날씨상태를 가리키는 순차열, 일별로 나열된 주가 (stock)열 등 다영한 순차열이 $\mathbf{x}$가 될 수 있다.

N-gram language model에서는 단어열 $\mathbf{x}$에 대한 확률 $P(\mathbf{x})$을 k-gram으로 구성된 n-gram probabilities을 이용하여 다음과 같이 근사시킨다.

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i | x_{i-1}, \cdots, x_{i-k+1}) \tag{1}$$

여기서, 표기의 단순성을 위해 $j < 0$인 경우, x_j는 해당 단어가 empty임을 의미한다고 하자.

HMM는 (직접 관측할 수 없지만) **은닉상태** (**hidden state**) 의 sequence $\mathbf{y} = \{y_1, \cdots, y_n\}$를 먼저 생성하고, 그 다음 은닉상태열을 조건으로 하여 관측열 $\mathbf{x}$를 생성하는 결합모델 (joint model)에 대한 Markov 근사 모델이다. $\mathbf{x}$, $\mathbf{y}$의 결합 확률을 구할때 Markov assumption을 적용하면, N-gram probabilities에 기반한 근사가 $\mathbf{y}$에 적용이 되고, 현재 각 단어가 생성될때는 해당 시점의 은닉상태에만 의존하고 다른 은닉상태나 이전단어들과는 조건부 독립 (conditional independence)이 만족된다.

우선, 은닉상태열과 관측열에 대한 결합 확률 $P(\mathbf{x}, \mathbf{y})$은 chain rule에 따라 다음과 같이 기술될 수 있다.

$$P(\mathbf{x}, \mathbf{y}) = P(\mathbf{x}|\mathbf{y})\, P(\mathbf{y}) \tag{2}$$

1.1 Bigram HMM (150 points)

Bigram HMM(이하 HMM)은 위의 식 Eq. (2)에서 $P(\mathbf{y})$를 Markov assumption에 따라 bi-gram language model근사를 은닉상태열에 적용하여, 다음과 같이 근사한다.

$$P(\mathbf{y}) \approx \prod_{i=1}^{n} q(y_i | y_{i-1}) \tag{3}$$

이때, sequence 시작과 끝 상태를 의미하는 START와 STOP를 특수상태로 도입하여, 다음과 같이 확장된 은닉상태 sequence인 $\tilde{\mathbf{y}} = (\mathsf{START}, y_1, \cdots, y_n, \mathsf{STOP})$에

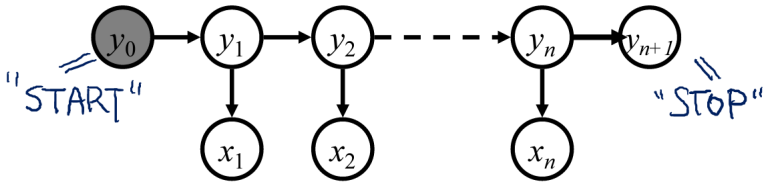
bigram language model을 적용하여 위의 식을 다음과 같이 정교화시킨다.

$$P(\tilde{\mathbf{y}}) \approx q(y_0) \prod_{i=1}^{n+1} q(y_i|y_{i-1}) \tag{4}$$

추가로, 현재 단어가 생성될때는 해당 시점의 은닉상태에만 의존한다는 가정을 통해, $P(\mathbf{x}|\mathbf{y})$를 다음과 같이 근사시킨다.

$$P(\mathbf{x}|\mathbf{y}) \approx \prod_{i=1}^{n} e(x_i|y_i) \tag{5}$$

1. 정리하면, Bigram HMM은 은닉상태열과 관측열에 대한 결합확률 Eq. (2)을 상태열상에서의 Markov assumption 및 생성과정의 은닉상태 참조 가정을 통해 Eq. (4)과 Eq. (5)를 이용하여 근사시킨 것이다. 다시말해, 결합확률 $P(\mathbf{x}, \mathbf{y}) = P(x_1, \cdots, x_n, y_1, \cdots, y_n)$을 근사시키기 위해, 랜덤변수들간 다음과 같은 Bayesian network를 가정한 것이다.



위의 Bayesian network을 참고하여, bigram HMM에서 사용된 Eq. (4)과 Eq. (5)의 근사식에서 가정한 모든 조건부 독립 가정을 $A \perp\!\!\!\perp B|C$의 형식으로 나열하시오.

2. (HMM의 Supervised Learning) 이제 HMM을 학습하기 위한 N개의 문장과 해당 states들이 annotation되어 있는 supervised dataset $\mathcal{D} = \left(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\right)_{j=1}^{N}$이 주어졌다고 하자.

   HMM의 파라미터 θ는 bigram **전이확률** (transition probability) $q(y_j|y_i)$과 단어 **생성 확률** (emission probability) $e(x|y)$로 구성되며, 이들 θ를 학습하기 위한 log-likelihood는 다음과 같다.

$$LogL = \sum_{j=1}^{N} \log P\left(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\right) \tag{6}$$

   Eq. (6)를 최대로 하는 θ의 Maximum likelihood estimation (MLE)가 다음과 같음을 유도하시오.

$$\begin{aligned} q(t|t') &= \frac{C(t', t)}{C(t)} \\ e(w|t) &= \frac{C(w, t)}{C(w)} \end{aligned} \tag{7}$$

   이때, $C(event)$는 학습데이터 $\mathcal{D}$상에서 $event$가 나타난 빈도수 (frequency)를 가리키며, 예를 들면 $C(t', t)$는 $\mathbf{y}_{j=1}^{(j)\,N}$상에서 bigram 상태전이 t', t가 출현한 빈도수이다.

유도과정을 보이면서, Eq. (7)의 빈도수 $C(t', t)$, $C(t)$, $C(w, t)$, $C(w)$ 또한 (수학적으로) 명확히 정의하시오.

3. (Inference) HMM의 θ가 학습된 이후에는, 추론 과정에서 새로운 test 입력 $\mathbf{x}$에 대해서, 결합확률 $P(\mathbf{x}, \mathbf{y})$를 최대로 하는 은닉상태열 $\mathbf{y}^*$를 찾는 조합 최적화 문제를 다루게 되며, 형식적으로 표현하면 다음과 같다.

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} P(\mathbf{x}, \mathbf{y}) \tag{8}$$

Viterbi알고리즘은 Eq. (8)의 최적화문제를 해결하기 위한 동적 프로그래밍 (dynamic programming)을 말한다. Viterbit 알고리즘을 유도하기 위한 핵심 과정으로 $\pi(i, y_i)$를 문장 시작부터 length i의 단어열 $x_1, \cdots, x_i$ 이 상태 (또는 태그) y_i로 끝날 **maximum** 결합 확률로 다음과 같이 정의하자.

$$\pi(i, y_i) = \max_{y_1, \cdots, y_{i-1}} P(x_1, , x_i, y_1, \cdots, y_i)$$

위의 정의에 따라 $\pi(i-1, y_{i-1})$를 시작부터 length $i-1$의 단어열이 상태 (또는 태그) y_{i-1}로 끝날 maximum 결합 확률이 된다.

Eq. (9)로 부터 $\pi(i, y_i)$을 바로 이전 length $i-1$의 순차열상의 $\pi(i-1, y_{i-1})$를 이용하여 재귀적으로 재기술하시오.

4. (Viterbi algorithm) 앞서의 $\pi(i, y_i)$에 대한 재귀식을 바탕으로, Eq. (8)의 최적화문제를 해결하기 위한 Viterbi 알고리즘을 유도하고, 최종 형태에 대한 pseudo-code를 기술하시오.

5. (HMM의 unsupervised learning) 이번에는 HMM을 unsupervised learning을 통해 학습하기 위해, N개의 문장들로 구성된 데이터셋 $\mathcal{D} = \left(\mathbf{x}^{(j)}\right)_{j=1}^{N}$로 주어졌다고 하자.

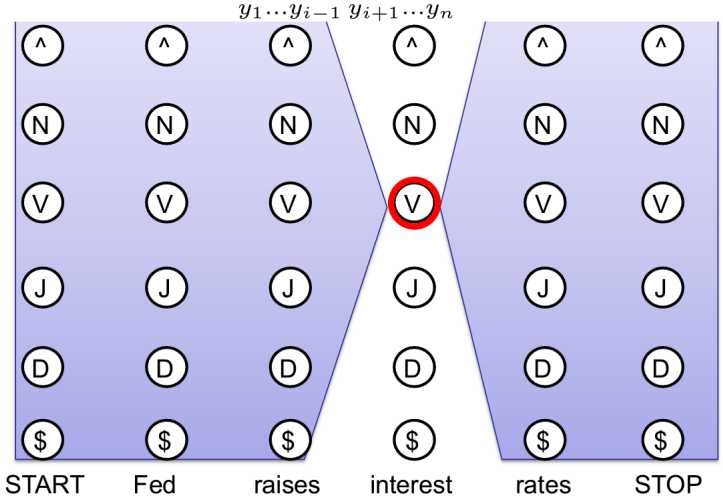
EM 알고리즘에 기반한 HMM의 unsupervised learning을 위해, 현재 t시점에 파라마터 θ^t가 주어졌다고 하자. 대표적인 unsupervised learning인 K-means 알고리즘은 K개의 centers를 무작위로 초기화하여 각 데이터별로 membership을 계산하여, 이렇게 얻어진 새로운 cluster members들을 바탕으로 K개의 centers를 계산하는 과정을 반복한다. K-means에 대한 EM버전에서는, 각 단계별로 데이터와 cluster간의 membership을 계산하는 과정인 E-step이 **확률적인 assignment**로 일반화되고, E-step을 부터 얻어진 **soft** cluster members로부터 membership정도를 가중치로 하는 가중 평균을 취하여 K개의 clusters를 업데이터하는 M-step을 반복적으로 수행하는 것이다.

HMM의 EM알고리즘을 위한 E-step은 간단히 말해, 주어진 문장 $\mathbf{x}_j$를 현재 θ^t상에서 확률적 태깅 (probabilistic tagging)을 수행하여, 파라미터를 구성하는 전이확률 $q(t|t')$과 생성확률 $e(w|t)$에 대한 **soft한 통계근거**를 확보하는 것이다.

이를 위해, 주어진 문장에 대해서 i번째의 상태(태그)가 y_i가 될 marginal probability인 $P(x_1, \cdots, x_n, y_i)$를 고려하자.

$$P(x_1, \cdots, x_n, y_i) = \sum_{y_1, \cdots, y_{i-1}} \sum_{y_{i+1}, \cdots, y_n} P(x_1, \cdots, x_n, y_1, \cdots, y_i, \cdots, y_n)$$

$$p(x_1 \dots x_n, y_i) = \sum_{y_1 \cdots y_{i-1}} \sum_{y_{i+1} \cdots y_n} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$



Eq. (9)의 summation파트는 i 앞부분과 뒷부분 크게 두개로 나뉘게 되며, Eq. (9)의 계산을 위한 동적알고리즘을 유도하기 위해, 다음과 같이 $P\left(x_1, \cdots, x_n, y_i\right)$ i의 앞부분과 뒷부분 파트로 분해한다.

$$P\left(x_1, \cdots, x_n, y_i\right) = P\left(x_1, \cdots, x_i, y_i\right) P\left(x_{i+1}, \cdots, x_n | y_i\right) \tag{9}$$

표기의 편의를 위해, 앞부분을 $\alpha(i, y_i)$와 뒷부분을 $\beta(i, y_i)$로 재기술하면 다음과 같다.

$$\begin{aligned} \alpha(i, y_i) &= P\left(x_1, \cdots, x_i, y_i\right) \\ \beta(i, y_i) &= P\left(x_{i+1}, \cdots, x_n | y_i\right) \end{aligned} \tag{10}$$

앞서 $\pi(i, y_i)$와 마찬가지로 이번에는 Bayesian network sum-product를 이용하여 $\alpha(i, y_i)$와 $\beta(i, y_i)$를 재귀적으로 표현하시오.

6. (Forward-backward algorithm) 앞서의 $\alpha(i, y_i)$와 $\beta(i, y_i)$의 재귀식을 바탕으로, E-step에서는 주어진 문장 $x_1, \cdots, x_n$에 대해 $p(y_1, \cdots, y_n | x_1, \cdots, x_n)$의 posterior probability의 계산을 수행하며, 이로부터 전이확률 및 생성확률의 근거가 되는 expected count가 유도가 된다.

   결국, HMM의 E-step단계의 핵심적 내용은 다음 posterior probabilities을 구하는 것이다.

$$\begin{aligned} P\left(y_i | x_1, \cdots, x_n\right) &= \frac{P\left(x_1, \cdots, x_n, y_i\right)}{P\left(x_1, \cdots, x_n\right)} \\ P\left(y_{i-1}, y_i | x_1, \cdots, x_n\right) &= \frac{P\left(x_1, \cdots, x_n, y_{i-1}, y_i\right)}{P\left(x_1, \cdots, x_n\right)} \end{aligned} \tag{11}$$

위의 정의에 따라, Eq. (11)의 $P\left(y_i | x_1, \cdots, x_n\right)$와 $P\left(y_{i-1}, y_i | x_1, \cdots, x_n\right)$를 $\alpha(i, y_i)$와 $\beta(i, y_i)$로 표현하고, 그 과정을 유도하시오.

7. (Forward-backward algorithm – M-step) M-step에서는 현재 파라미터로부터 얻어진 Eq. (11)의 $P(y_i|x_1, \cdots, x_n)$와 $P(y_{i-1}, y_i|x_1, \cdots, x_n)$를 바탕으로 다음 파라미터를 estimation하는 것이다.

   앞서의 supervised learning의 Eq. (7)의 MLE가, EM의 M-step에서는 다음과 같이 표현된다고 할때,

$$q^{(t+1)}(y'|y) = \frac{\gamma(y, y')}{\gamma(y)}$$
$$e^{(t+1)}(x|y) = \frac{\gamma(y, x)}{\gamma(y)} \tag{12}$$

   Expected counts에 대응되는 $\gamma(y, x)$, $\gamma(y)$, $\gamma(y, y')$를 수학적으로 정의하고, 이를 유도하시오.

   이로부터 최종 HMM의 EM-alorithm인 Forward-backward알고리즘을 정리하고 pseudo code를 기술하시오.

1.2 Bigram HMM의 구현 (400 points)

본 절에서는 앞절에서 유도한 Bigram을 실제 `python`코드 (행렬연산시 `numpy`사용 권장)로 구현해보고 영문 품사 태깅 (POS tagging)에 적용해보는 것을 목표로 한다.

1.2.1 HMM의 Supervised Learning 구현 및 POS tagging에 적용

HMM학습을 위해 제공된 영문 POS Tagging 데이터셋을 사용하고, 데이터셋은 다음과 같이 Training/dev/test셋을 위한 세개의 파일로 구성되어 있다.

- Training dataset (`tagged_train.txt`): 단어별 품사태그가 부착된 POS Tagging 태스크를 위한 학습데이터셋
- Development dataset (`tagged_dev.txt`): 단어별 품사태그가 부착된 POS Tagging 태스크를 위한 개발용 데이터셋
- Test dataset (`tagged_test.txt`): 단어별 품사태그가 부착된 POS Tagging 태스크를 위한 평가용 데이터셋
- Raw Training dataset (`raw_train.txt`): 코퍼스상 tokenizer가 적용된 학습데이터셋상에서 원문 문장셋 (EM algorithm에 기반한 HMM학습시 사용)

다음은 `raw_train.txt` 파일의 일부를 보여준다.

```
1130b36a9ba93d8fcbf07a2ac0048d4c281cd57c::29 Geordie Nicholson .
1130b36a9ba93d8fcbf07a2ac0048d4c281cd57c::30 Web site: burgessyachts.com .
1130b36a9ba93d8fcbf07a2ac0048d4c281cd57c::31 'Big Aron' Why we like it This
    capacious 2004 yacht was refitted in 2006.
```

다음은 `tagged_train.txt` 파일의 일부를 보여준다.

```
42c027e4ff9730fbb3de84c1af0d2c506e41c3e4::0 LONDON/NNP ,/, England/NNP (/(
    Reuters/NNP )/) --/: Harry/NNP Potter/NNP star/NN Daniel/NNP
    Radcliffe/NNP gains/NNS access/NN to/TO a/DT reported/VBN
     - 128) °20/CD million/CD (/( $/$ 41.1/CD million/CD )/) fortune/NN
```

```
    as/IN he/PRP turns/VBZ 18/CD on/IN Monday/NNP ,/, but/CC he/PRP
    insists/VBZ the/DT money/NN wo/MD n't/RB cast/VB a/DT spell/NN on/IN
    him/PRP ./.
42c027e4ff9730fbb3de84c1af0d2c506e41c3e4::1 Daniel/NNP Radcliffe/NNP as/IN
    Harry/NNP Potter/NNP in/IN ``/`` Harry/NNP Potter/NNP and/CC the/DT
    Order/NN of/IN the/DT Phoenix/NNP ''/'' To/TO the/DT disappointment/NN
    of/IN gossip/NN columnists/NNS around/IN the/DT world/NN ,/, the/DT
    young/JJ actor/NN says/VBZ he/PRP has/VBZ no/DT plans/NNS to/TO
    fritter/VB his/PRP$ cash/NN away/RB on/IN fast/JJ cars/NNS ,/, drink/NN
    and/CC celebrity/NN parties/NNS ./.
42c027e4ff9730fbb3de84c1af0d2c506e41c3e4::2 ``/`` I/PRP do/VBP n't/RB
    plan/VB to/TO be/VB one/CD of/IN those/DT people/NNS who/WP ,/, as/RB
    soon/RB as/IN they/PRP turn/VBP 18/CD ,/, suddenly/RB buy/VBP
    themselves/PRP a/DT massive/JJ sports/NNS car/NN collection/NN or/CC
    something/NN similar/JJ ,/, ''/'' he/PRP told/VBD an/DT Australian/JJ
    interviewer/NN earlier/RBR this/DT month/NN ./.
```

위의 파일에서 각 라인의 첫번째 token은 문장 고유 id를 의미한다.

참고로, 본 데이터셋은 공개 영문 Raw corpus상에서 Stanford의 POS tagger를 적용하여, 자동으로 얻은 데이터셋이다. 따라서, Tagging된 데이터셋상에서 오류가 있을 수 있다.

다음 상세 요구사항을 참조하여 python코드를 작성하시오.

1. **Supervised learning에 기반한 학습기** (Adding one smoothing): 위의 Training셋에 있는 모든 문장을 읽어들이고, 전이확률 $q(t|t')$ 및 생성확률 $e(w|t)$를 MLE 공식 Eq. (7)을 **adding one smoothing**을 이용하여 확장하여, 학습하는 python코드를 작성하시오.

   학습된 모델은 별도의 파일로 저장이 되어야 한다. 학습된 모델이 smoothing에 의존하기 때문에, Raw corpus상 고유한 각 빈도수 정보는 미리 계산하여 raw 통계정보를 별도의 파일에 저장한후에 (예: `HMM_counts.dat`), adding one smoothing을 적용한 모델 파일을 이로부터 유도하는 식으로 두 단계에 걸쳐서 저장해도 된다 (`HMM_addingone_model.dat`).

2. **Viterbi 태거**: 앞서 얻은 HMM 모델 파일을 읽어들여, 해당 모델의 전이확률 $q(t|t')$ 및 생성확률 $e(w|t)$를 바탕으로 주어진 입력 문장에 대해서 Viterbi algorithm을 이용하여 가장 높은 태그열을 출력하는 python코드를 작성하시오.

   태스트를 위해서, 인자로 model파일을 입력으로 받도록 하고, 콘솔환경에서 문장은 공백라인으로 끝나도록 하고, 엔터로 입력이 끝나면 태그열을 생성하도록 하라. 배치 처리를 위해 python argparse등에 기반하여 모델파일외에 입력파일과 출력파일을 추가 인자로 제시하도록 하여, 입력파일내에 있는 모든 문장에 대한 viterbi tagging결과를 출력파일에 각각 출력하도록 하라.

3. **학습된 HMM평가**: 위의 학습된 HMM모델을 바탕으로 제공된 Test셋상에서 적용하여, POS tagging 평가를 수행하시오. 평가 결과로 1) 총 단어수, 2) HMM에 기반하여 POS Tag를 정확히 맞춘 총 단어수, 3) Accuracy 의 수치가 포함되도록 출력하시오. 이때, Accuracy는 다음과 같이 정의된다.

$$Accuracy = \frac{\textit{예측된 POS Tag가 정답과 일치하는 총 단어수}}{\textit{테스트셋상 총 단어수}} \tag{13}$$

4. **Unsupervised learning에 기반한 HMM 학습기**: 이번에는 HMM에 대한 unsupervised learning을 위해, Training셋 상에 있는 모든 문장을 읽어들이고 (이때 문장만 읽어들어야 한다), 전이확률 $q(t|t')$ 및 생성확률 $e(w|t)$를 랜덤하게 초기화한후에, 은닉상태 갯수를 K로 취하고, forward-backward 알고리즘을 통한 EM알고리즘 기반 학습기를 python코드로 구현하시오.

   학습결과는 마찬가지로 앞서 supervised learning과 동일한 포맷의 model file로 저장되어야 한다.

5. **Unsupervised learning에 기반한 HMM평가**: 앞 EM알고리즘에 기반한 HMM을 $K = 10, 20, 30, 40, 50$의 옵션으로 두고 학습을 수행한후에 각 K에 대해서 Test셋상에서 평가를 수행하시오.

   이때, 평가를 위해, 각각의 은닉상태 t마다 Test셋상에서 있는 POS tag들중 하나로 매핑을 수행해야 하며, 이를 위해, 다음과 같이 t와 실제 POS tag y가 수반되는 전이확률 및 생성확률의 KL 분포를 이용하여 dissimilarity를 계산한다.

$$Dist(t, y) = \gamma_1 \cdot KL\left(q(\cdot|t) \| q(\cdot|y)\right) + (1 - \gamma_1) \cdot KL\left(e(\cdot|t) \| e(\cdot|y)\right) \quad (14)$$

   위의 distance 측도를 통해, 가장 유사한 POS tag y^*를 찾아, 이를 t에 대한 매핑된 POS tag로 삼는다.

$$y^*[t] = \underset{y}{\operatorname{argmin}}\, Dist(t, y) \quad (15)$$

   결국, 각 입력 문장에 대해서 Unsupervised learning으로 학습한 HMM을 이용하여 Viterbi 알고리즘을 통해 태깅된 은닉상태 t들을 모두 $y^*[t]$로 변환하고, Supervised Learning에 기반한 HMM모델 평가시 사용한 Accuracy를 측정하면 된다.

   $K = 10, 20, 30, 40, 50$일때 Accuracy를 평가하고, 비교하시오.

1.3 Trigram HMM (50 points)

Trigram HMM은 위의 식 Eq. (2)에서 $P(\mathbf{y})$를 Markov assumption에 따라 trigram language model근사를 은닉상태열에 적용하여 다음과 같이 bigram HMM을 정교화 한것이다.

$$P(\mathbf{y}) \approx \prod_{i=1}^{n} q\left(y_i | y_{i-1}, y_{i-2}\right) \quad (16)$$

나머지 단어생성하는 과정은 bigram HMM과 동일하다. 다음 물음에 답하시오.

1. Trigram HMM의 Supervised Learning: bigram HMM에서의 Supervised learning방식을 확장하여, Trigram HMM에서의 MLE 계산식을 기술하시오.

2. Trigram HMM의 Viterbi algorithm: Trigram HMM에서의 Viterbi algorithm에서 사용된 max-product 방식을 확장하여 Trigram HMM에서의 Viterbi algorithm을 기술하고, 유도과정을 기술하시오.

3. Trigram HMM에서 Smoothing: Trigram HMM에서는 전이확률이 Bigram HMM보다 Sparseness가 더 높다. 이를 위해, 어떠한 smoothing방식을 적용하는 것이 효과적일지 기술하시오.

2 제출 내용 및 평가 방식

Section 1.1과 Section 1.3에 대해서는 **문제 답안 상세 결과물**을 제출해야 한다.

Section 1.2의 구현 문제의 경우 **python**코드외에 결과보고서도 함께 제출해야 하며, 정리하면 다음과 같다.

- **코드 전체**
- **테스트 결과**: 각 내용별 테스트 코드 및 해당 로그 또는 출력 결과.
- **결과보고서**: 구현 방법을 요약한 보고서.

Section 1.2 구현 문항의 평가항목 및 배점은 다음과 같다.

- **각 세부내용의 구현 정확성 및 완결성 (80%)**
- **코드의 Readability 및 체계성 (10%)**
- **결과 보고서의 구체성 및 완결성 (10%)**