Part III: Typical Unsupervised Learning Methods: EM and Viterbi Algorithms

- Basic Unsupervised Learning Methods: EM & Viterbi
- Characteristics of the Unsupervised Methods
- An Example: Part-of-speech Tagging
- More Details and Differences Between EM & Viterbi
- Potential Problems with the Unsupervised Methods

Basic Unsupervised Learning: EM/Viterbi Algorithm

1. Develop a Model:

- Select Potentially Useful Features
- Build a statistical Language Model with those adopted features

2. Prepare a Training Corpus

3. Set up Initial Conditions:

- EM: Guessing Initial Model Parameter (uniformly, or heuristically), and then calculating the initial expectation
- Viterbi: Guessing Initial Labels

Basic Unsupervised Learning (cont.):

4. Re-Estimate Model Parameters via MLE

- EM: Using the expectation (which implies weighting every possibility)
- Viterbi: Using the guessed labels (which implies using only one possibility: a simplified case of EM)

5. Re-Generating Prediction according to new Model Parameters

- EM: Re-estimate the Expectation of Sufficient Statistic
- Viterbi: Re-Labeling

6. Repeat the Prediction and Estimation Steps until the joint likelihood value of the training corpus converge

Characteristics of the Unsupervised Learning Algorithms

- Joint likelihood values of the training corpus monotonically increases with iterations

$$\text{EM: } P\left([w_1^n]_0|\Lambda_0\right) \le P\left([w_1^n]_0|\Lambda_1\right) \le P\left([w_1^n]_1|\Lambda_1\right) \cdots$$

- Likelihood values will converge to a local/global maximum given sufficiently large iterations

Viterbi Training (I):

- Example Task: Tagging a Corpus w_1^n (=w_1, w_2,…, w_n) with the appropriate tag sequence c_1^n (=c_1, c_2,…, c_n)
- 1. Decide Statistical Language Model:

$$\hat{c}_1^n = \arg\max_{c_1^n} P(c_1,\cdots,c_n|w_1,\cdots,w_n,\Lambda)$$
$$= \arg\max_{c_1^n} P(w_1^n|c_1^n,\Lambda)\times P(c_1^n|\Lambda)$$

  - Bi-gram Language Model:

$$\hat{c}_1^n \equiv \arg\max_{c_1^n} \prod_{i=1}^{n} P(w_i|c_i)\times P(c_i|c_{i-1})$$

Viterbi Training (II):

- 2. Get Untagged Corpora:
  - Example:
    - The current design of …
    - det adj/n v/n p …
- 3. Make Initial Guess (based on initial parameter set Λ_0):
  - i.e., prior distribution of unigram, $P(c_{ki} \mid w_i)$
  - The Current Design of ...
  - $[c_1^n]_0$: [det n v p ...]

Viterbi Training (III):

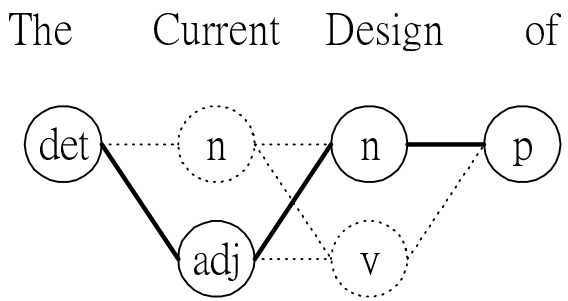
- 4. Maximum Likelihood Estimation

$$\Lambda_1 = \arg\max_{\Lambda} P\left(\left[c_1^n\right]_0 | w_1^n, \Lambda\right)$$

$$P\left(c_i = \mathrm{n} | c_{i-1} = \mathrm{det}\right) = \frac{\#\left[\mathrm{det}, \mathrm{n}\right]}{\#\left[\mathrm{det}\right]}$$

$$P\left(w_i = \mathrm{design} | c_i = \mathrm{v}\right) = \frac{\#\left[\mathrm{design}, \mathrm{v}\right]}{\#\left[\mathrm{v}\right]}$$

- 5. Re-tagging: Select the path with maximum probability



$$\left[c_1^n\right]_1 = \arg\max_{\left\{c_1^n\right\}} P\left(c_1^n | w_1^n, \Lambda_1\right)$$

Viterbi Training (IV):

- 6. Re-estimation: Estimate parameters that maximize the likelihood value

$$\Lambda_2 = \arg\max_{\Lambda} P\left(\left[c_1^n\right]_1 | w_1^n, \Lambda\right)$$

- 7. Repeat: Λ_1 => Λ_2 => Λ_3 => => Λ^* (optimal parameters)

- Likelihood Value is Monotonically Increasing

$$P\left(\left[c_1^n\right]_0 | w_1^n, \Lambda_0\right) \le P\left(\left[c_1^n\right]_0 | w_1^n, \Lambda_1\right)$$
$$\le P\left(\left[c_1^n\right]_1 | w_1^n, \Lambda_1\right) \le P\left(\left[c_1^n\right]_1 | w_1^n, \Lambda_2\right) \le \cdots$$

EM Training (I)

- Example Task: Tagging a Corpus w_1^n (=w_1, w_2,…, w_n) with the appropriate tag sequence c_1^n (=c_1, c_2,…, c_n)

1: Set up Language Model, which is the same as that in the Viterbi Training

- Bi-gram Language Model:

$$\hat{c}_1^n \equiv \arg\max_{c_1^n} \prod_{i=1}^{n} P(c_i|c_{i-1}) \times P(w_i|c_i)$$

2: Prepare Training Corpus, which is the same as that in the Viterbi Training

EM Training (II)

3: Guessing Initial Model Parameter (uniformly, or heuristically), and then calculating the initial expectation of the sufficient statistics ([$N_{c(i)}$, $N_{c(i),\,c(j)}$, $N_{w(k)}$]; for every possible combination)

- E[$N_{c(i)}$] : Expected Number of transitions from a specific POS c(i) (position independent: the position index is ignored)

$$E\left[N_{c(i)}\right] = \sum_{c_1^n}\left[\text{Number of times that } c(i) \text{ occurs under } c_1^n\right] \times P\left(c_1^n\right)$$

- E[$N_{c(i),\,c(j)}$]: Expected Number of transitions from a specific POS c(i) to another POS c(j) (ignoring the position indexes)

$$E\left[N_{c(i),c(j)}\right] = \sum_{c_1^n}\left[\#[c(i),c(j)] \text{ pairs occurs under } c_1^n\right] \times P\left(c_1^n\right)$$

EM Training (III)

3: Calculate the expectation of the sufficient statistics (cont.)

- E[$N_{w(k)}$]: Expected Number of word w(k) appears in the corpus (position independent: the position index is ignored)

$$E\left[N_{w(k)}\right] = \sum_{c_1^n} \left[\text{Number of times that } w(k) \text{ occurs under } c_1^n\right] \times P\left(c_1^n\right)$$
$$= \left[\text{Number of times that } w(k) \text{ occurs in the corpus}\right]$$

4: Using the expectation to do Maximum Likelihood Estimation

$$P\left(c_i = c(j) | c_{i-1} = c(i)\right) = \frac{E\left[N_{c(i),c(j)}\right]}{E\left[N_{c(i)}\right]}; \quad P\left(w_i = w(k) | c_i = c(j)\right) = \frac{E\left[N_{w(k)}\right]}{E\left[N_{c(j)}\right]}$$

EM Training (IV)

5: Re-calculate the expectation of the sufficient statistics

- Repeat the calculations as in Step (3).

6: Re-estimate the parameters

- Repeat the calculations as in Step (4).

7: Repeat the above procedures.

- Likelihood Value Monotonically Increases

$$P\left([w_1^n]_0 \mid \Lambda_0\right) \le P\left([w_1^n]_0 \mid \Lambda_1\right) \le P\left([w_1^n]_1 \mid \Lambda_1\right) \le \cdots$$

EM versus Viterbi (I)

- EM is a kind of soft-labeling (that is, an observation can belong to several different classes simultaneously with various associated probabilities).
  - Example: every possible tag are assign to a given word-position (I.e., every possible tag sequence are associated with the given word sequence) .
- Viterbi is a kind of hard-labeling (that is, an observation can only belong to one class).
  - Example: only one tag can be assigned to a given word-position (I.e., only one tag sequence is associated with the given word sequence).

EM versus Viterbi (II)

- Soft-labeling versus Hard-labeling (cont.)
  - Comparison:
    - Hard-Labeling is a special case of the Soft-Labeling
    - Soft-Labeling carries more information than the Hard-Labeling does, given the same amount of training data (i.e., more efficient).
    - The advantage diminishes when the corpus size goes larger
    - Soft-Labeling annotation is very difficult to be performed by human

EM versus Viterbi (III)

- Optimization Criteria are Different
  - EM: optimize

$$\hat{\Lambda} = \arg\max_{\Lambda} P\left(w_1^n|\Lambda\right) = \arg\max_{\Lambda} \sum_{c_1^n} P\left(w_1^n, c_1^n|\Lambda\right)$$

  - Viterbi: optimize

$$\hat{\Lambda} = \arg\max_{\Lambda} \left\{\max_{c_1^n} P\left(w_1^n, c_1^n|\Lambda\right)\right\} = \arg\max_{\Lambda} \left\{\max_{c_1^n} \left[P\left(w_1^n|c_1^n, \Lambda\right) \times P\left(c_1^n|\Lambda\right)\right]\right\}$$

  - In general, they would converge to different parameter points (I.e., obtain different parameter sets)

EM versus Viterbi (IV)

- Characteristics Comparison:
  - EM can deliver better performance, but requires heavier computation
  - Viterbi is simple and quick, but with inferior performance
  - The performance difference is usually small and tolerable for most NLP and Speech Recognition tasks conducted in the community
    - In many cases, the parameters will be adjusted again by using an adaptive learning algorithm any way
    - Initial difference will not make effect after adaptive learning

More on EM Training [Dempster 77]

- EM (Expectation and Maximization) algorithm: an unsupervised training process which consists of an expectation step followed by a maximization step.

- There is a many-to-one mapping **x**→**y** from ***X*** to ***Y***.

  - **x**: is the *complete data* with density **x** ~ $f(\mathbf{x}|\Phi)$ depending on the parameter set Φ.

  - **y**: the *incomplete data* with the sampling density $g(\mathbf{y}|\Phi)$

$$g(y|\Phi) = \int_{X(y)} f(x|\Phi)\,dx$$

More on EM Training (cont.)

- The EM training procedure in the *p*-th iteration:
  - E-step: Estimate the complete-data sufficient statistics **t(x)** by finding

$$t^{(p)} = E\left[t(X)\middle|y, \Phi^{(p)}\right]$$

  - M-step: Determine $\Phi^{(p+1)}$ which maximizes.

$$f\left(t^{(p)}\middle|\Phi^{(p+1)}\right)$$

Example Task:

- The observed (incomplete) data y: 197 animals which are distributed multinomially into 4 categories.
  - $y=(y_1, y_2, y_3, y_4)=(125,18,20,34)$.
  - cell probabilities: $(1/2+1/4\pi, 1/4(1-\pi), 1/4(1-\pi), 1/4\pi)$
- The complete data (multinomially distributed):
  - $x=(x_1, x_2, x_3, x_4, x_5)$.
  - cell probabilities: $(1/2, 1/4\pi, 1/4(1-\pi), 1/4(1-\pi), 1/4\pi)$ [with one parameter π]
  - $y_1 = x_1+x_2$, e.g., (125, 0), or (124,1), or …
  - $x_3 = y_2$, $x_4 = y_3$, $x_5 = y_4$.

$$f(x|\pi)==\frac{(x_1+x_2+x_3+x_4+x_5)!}{x_1!x_2!x_3!x_4!x_5!}\left(\frac{1}{2}\right)^{x_1}\left(\frac{1}{4}\pi\right)^{x_2}\left(\frac{1}{4}-\frac{1}{4}\pi\right)^{x_3}\left(\frac{1}{4}-\frac{1}{4}\pi\right)^{x_4}\left(\frac{1}{4}\pi\right)^{x_5}$$

E-Step:

- To find $\pi^{(p+1)}$ from $\pi^{(p)}$ ($\pi^{(p)}$ denotes the value of π after *p* iterations):

- E-Step:

$$x_1^{(p)} = E\left[X_1 | X_1 + X_2 = 125, \pi^{(p)} \right] = 125 \times \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{4}\pi^{(p)}}$$

$$x_2^{(p)} = E\left[X_2 | X_1 + X_2 = 125, \pi^{(p)} \right] = 125 \times \frac{\frac{1}{4}\pi^{(p)}}{\frac{1}{2} + \frac{1}{4}\pi^{(p)}}$$

M-Step:

- M-Step:

$$Let \ \frac{\partial f(x|\pi)}{\partial \pi} = 0 \Rightarrow \pi = \frac{x_2 + x_5}{x_2 + x_3 + x_4 + x_5}$$

$$\pi^{(p+1)} = \frac{x_2^{(p)} + 34}{x_2^{(p)} + 18 + 20 + 34}$$

  - x_1^p and x_2^p are usually not integers.
  - The re-estimation process converge to π^* when p>5, where $\pi^* \approx$ 0.6268214980 is the MLE of π.

Common Problems with Unsupervised Learning (I)

- *Ad Hoc* Features Selection:
  - Feature Space determines the Upper Bound of performance
  - Over simplified bigram, trigram models may fail to gain success on complicated NLP tasks
  - Class-based features not used, resulting in a large number of parameters (that is, no refined model; e.g., words vs. tags or chunks)
- Feature Dependency Overlooked: Model Deficiency
  - Inappropriate independence assumptions, inappropriate dependency relationship assumed (to be described in the afternoon session)
- Over Fitting of Model:
  - Using high Model Complexity with Small Training Corpus
- Un-hinted Initial Guess:
  - Trapped in undesired Local Maximum

Common Problems with Unsupervised Learning (II)

- Unseen and Untrained Events not Well Estimated
  - Give poor performance when the case involves the unseen event
- Mismatch between ML Estimation & Human Preference not Compensated
  - Maximizing training set likelihood does not implies good testing set performance
- System Sensitivity (versus statistical characteristics variation) not considered
  - System Sensitivity : the degree of variation of system performance that will be caused by the variation of the statistical characteristics between the training set and the testing set
  - Statistical characteristics variation: variation between inherited statistical characteristics (implied by the parameters) of the training set and the testing set