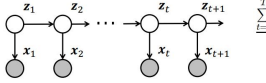
Baum-Welch算法是EM算法的一种，其解决了隐马尔科夫模型（Hidden Markov Model，HMM）三大主要问题中的学习问题。HMM的学习问题可以按如下方式定义：给定观测序列$X=\{x_1,...,x_T\}$，在隐藏序列$Z=\{z_1,...,z_T\}$未知的情况下，如何估计模型的最佳参数θ，使得$P(X\,|\,\theta)$最大。



$$\frac{\sum_{t=1}^{T}P(z_t=j,x_t=k|X,\theta^{(i)})}{\sum_{t=1}^{T}P(z_t=j|X,\theta^{(i)})}=\frac{\sum_{t=1}^{T}\gamma_t(j)1(x_t=k)}{\sum_{t=1}^{T}\gamma_t(j)}.$$

参数$\theta=\{\pi,A,B\}$，包括初始概率分布$\pi=[\pi_i]_N$，转移（Transition）矩阵$A=[a_{ij}]_{N\times N}$，观测/发射（Emission）矩阵$B=[b_j(k)]_{N\times M}$，其中N表示隐状态总数，M表示可观测状态总数。在本题中我们将利用先前的知识完成该算法的推导。

(1) 首先进行**E**步的计算，在这一步中我们根据当前的网络参数θ^{old}得到隐变量Z的后验概率$q(Z)=P(Z,X\,|\,\theta^{(i)})$，然后固定住$q(Z)$，并列出ELBO得到优化目标。请根据ELBO写出$J(\theta)$，并证明：

$$\underset{\theta}{\operatorname{argmax}}\,J(\theta)=\underset{\theta}{\operatorname{argmax}}\sum_{Z}P(X,Z\,|\,\theta^{old})\log P(X,Z\,|\,\theta)$$

(2) 使用模型的参数θ（包括π,A,B）来表示$P(X,Z\,|\,\theta)$.

(3) 令$Q(\theta,\theta^{old})=\sum_Z P(X,Z\,|\,\theta^{old})\log P(X,Z\,|\,\theta)$，应用(2)中的结果，将$Q(\theta,\theta^{old})$拆分成三项之和，其中每一项仅与$\pi,A,B$中的一个参数有关。

(4) 上一小题实现了参数之间的解耦，可以进入**M**步的计算，此时暂时不需要考虑θ^{old}的处理。我们以π的求解为例。请首先证明

$$\sum_Z P(X,Z\,|\,\theta^{old})\log\pi_{z_1}=\sum_{i=1}^{N}P(X,z_1=i\,|\,\theta^{old})\log\pi_i,$$

下取到极大值时π_i的取值。

然后使用拉格朗日乘子法，求$\sum_Z P(X,Z\,|\,\theta^{old})\log\pi_{z_1}$在$\sum_{i=1}^{N}\pi_i=1$

(5) 参照上一小题的过程，求$Q(\theta,\theta^{old})$取极大值时a_{ij}和$b_j(k)$的取值。

$$\gamma_t^{old}(i)=P(z_t=i\,|\,X,\theta^{old})\qquad \xi_t^{old}(i,j)=P(z_t=i,z_{t+1}=j\,|\,X,\theta^{old})$$

回顾课上讲过的前向-后向算法，对于给定参数θ^{old}的HMM，我们可以求出$\alpha_t^{old}(i)=P(x_{1:t},z_t=i\,|\,\theta^{old})$，$\beta_t^{old}(i)=P(x_{t+1:T}\,|\,z_t=i,\theta^{old})$请使用$\alpha_t^{old}(i),\beta_t^{old}(i),a_{ij}^{old},b_j^{old}(k)$表示出$\gamma_t^{old}(i)$和$\xi_t^{old}(i,j)$.

(7) 回顾(4)(5)的结果，请使用$\gamma_t^{old}(i)$和$\xi_t^{old}(i,j)$表示出$\pi_i,a_{ij},b_j(k)$，并给出Baum-Welch算法的流程。

(a) $\because J(\theta)=\mathscr{L}(Q^*,\theta)=\sum_Z Q^*(Z)\log\frac{P(X,Z|\theta)}{Q^*(Z)}$，其中$Q^*(Z)=P(Z|X,\theta^{(i)})$，$\therefore$

$$\arg\max_\theta J(\theta)=\arg\max_\theta\sum_Z Q^*(Z)\log\frac{P(X,Z|\theta)}{Q^*(Z)}=\arg\max_\theta\sum_Z Q^*(Z)\log(P(X,Z|\theta))-$$
$$\arg\max_\theta Q^*(Z)\log Q^*(Z)=\arg\max_\theta\sum_Z P(Z|X,\theta^{(i)})\log(P(X,Z|\theta))=$$
$$\arg\max_\theta\sum_Z\frac{P(Z,X|\theta^{(i)})}{P(X)}\log(P(X,Z|\theta))=\frac{1}{P(X)}\arg\max_\theta\sum_Z P(Z,X|\theta^{(i)})\log(P(X,Z|\theta))=$$
$$\arg\max_\theta\sum_Z P(Z,X|\theta^{(i)})\log(P(X,Z|\theta)).$$

(b) $P(X,Z|\theta)=\pi_{z_1}\prod_{t=1}^{T-1}b_{z_t}(x_t)a_{z_t,z_{t+1}}\cdot b_{z_N}(x_N)=\pi_{z_1}\prod_{t=1}^{T-1}a_{z_t,z_{t+1}}\prod_{t=1}^{T}b_{z_t}(x_t)$，$\therefore Q(\theta,\theta^{(i)})=$

$$\sum_Z P(X,Z|\theta^{(i)})\log P(X,Z|\theta)=\sum_Z P(X,Z|\theta^{(i)})\log\pi_{z_1}+\sum_Z P(X,Z|\theta^{(i)})\sum_{t=1}^{T-1}\log a_{z_t,z_{t+1}}+$$
$$\sum_Z P(X,Z|\theta^{(i)})\sum_{t=1}^{T}\log b_{z_t}(x_t).$$

(c) $\sum_Z\log\pi_{z_1}P(X,Z|\theta^{(i)})=\sum_{i=1}^{N}\sum_{z_1=i}\log\pi_{z_1}P(X,Z|\theta^{(i)})=\sum_{i=1}^{N}\log\pi_i P(X,z_1=i|\theta^{(i)})$，∴在约束条件$\sum_{i=1}^{N}\pi_i=1$下，$Q(\theta|\theta^{(i)})$第一项$\sum_Z P(X,Z|\theta^{(i)})\log\pi_{z_1}$拉格朗日函数的形式为

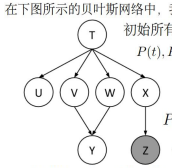
$$\sum_{i=1}^{N}\log\pi_i P(X,z_1=i|\theta^{(i)})+\gamma(\sum_{i=1}^{N}\pi_i-1)=0$$，对各π_i分别求偏导得$\frac{\partial}{\partial\pi_i}LHS=\frac{P(X,z_1=i|\theta^{(i)})}{\pi_i}+\gamma=0$，即$\pi_i=-\frac{P(X,z_1=i|\theta^{(i)})}{\gamma}$，对$i$遍历$1,2,\cdots,N$求和，得$1=\sum_{i=1}^{N}\pi_i=-\frac{1}{\gamma}\sum_{i=1}^{N}P(X,z_1=i|\theta^{(i)})=-\frac{1}{\gamma}P(X|\theta^{(i)})$，$\gamma=-P(X|\theta^{(i)})$，$\therefore\pi_i=\frac{P(X,z_1=i|\theta^{(i)})}{P(X|\theta^{(i)})}$. 在约束条件$\sum_{j=1}^{N}\sum_{i=1}^{N}a_{ij}=1$下，$Q(\theta|\theta^{(i)})$第二项$\sum_Z P(X,Z|\theta^{(i)})\sum_{t=1}^{T-1}\log z_t,z_{t+1}$拉格朗日函数的形式为$\sum_{i=1}^{N}\sum_{j=1}^{N}P(X,z_t=j,z_{t+1}=k|\theta^{(i)})\sum_{t=1}^{T-1}\log a_{ij}+\gamma(\sum_{i=1}^{N}\sum_{j=1}^{N}a_{ij}-1)=0$，对各$a_{ij}$分别求偏导得$\frac{\partial}{\partial a_{ij}}LHS=\frac{\sum_{t=0}^{T-1}P(X,z_t=i,z_{t+1}=j|\theta^{(i)})}{a_{ij}}+\gamma=0$，同上进行变换有$a_{ij}=\frac{\sum_{t=0}^{T-1}P(X,z_t=i,z_{t+1}=j|\theta^{(i)})}{\sum_{t=0}^{T-1}P(X,a_{z_t}=j|\theta^{(i)})}$. 在约束条件$\sum_{k=1}^{M}b_j(k)=1$下，$Q(\theta,\theta^{(i)})$第三项$\sum_Z P(X,Z|\theta^{(i)})\sum_{t=1}^{T}\log b_{z_t}(x_t)$拉格朗日函数的形式为$\sum_{k=1}^{M}P(X,z_t=j|\theta^{(i)})\sum_{t=1}^{T}\log b_{z_t}(x_t)+\gamma(\sum_{k=1}^{M}b_j(k)-1)=0$，对各$b_j(k)$分别求偏导得$\frac{\partial}{\partial b_j(k)}LHS=\frac{P(X,z_t=j,x_t=k|\theta^{(i)})}{b_j(k)}+\gamma=0$，同上变换有$b_j(k)=\frac{\sum_{t=1}^{T}P(X,z_t=j,x_t=k|\theta^{(i)})}{\sum_{t=1}^{T}P(X,z_t=j|\theta^{(i)})}$.

(d) $\gamma_t(i)=P(z_t=q_i|X,\theta)=\frac{P(z_t=q_i,X|\theta)}{P(X|\theta)}=\frac{P(z_t=q_i,X|\theta)}{\sum_{j=1}^{N}P(z_t=q_i,X|\theta)}=\frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}$，$\xi(i,j)=P(z_t=q_i,z_{t+1}=q_j|X,\theta)=\frac{P(z_t=q_i,z_{t+1}=q_j,X|\theta)}{P(X|\theta)}=\frac{P(z_t=q_i,z_{t+1}=q_j,X|\theta)}{\sum_{i=1}^{N}\sum_{j=1}^{N}P(z_t=q_i,z_{t+1}=q_j,X|\theta)}=\frac{\alpha_t(i)\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(k)\beta_{t+1}(l)}$

(e) $\pi_i=\frac{P(X,z_1=i|\theta^{(i)})}{P(X|\theta^{(i)})}=P(z_t=q_i|X,\theta^{(i)})=\gamma_1(i)$，$a_{ij}=\frac{\sum_{t=0}^{T-1}P(X,z_t=i,z_{t+1}=j|\theta^{(i)})}{\sum_{t=0}^{T-1}P(X,a_{z_t}=j|\theta^{(i)})}=\frac{\sum_{t=0}^{T-1}\frac{1}{P(X,\theta^{(i)})}P(X,z_t=i,z_{t+1}=j|\theta^{(i)})}{\sum_{t=0}^{T-1}\frac{1}{P(X,\theta^{(i)})}P(X,a_{z_t}=j|\theta^{(i)})}=\frac{\sum_{t=0}^{T-1}P(z_t=i,z_{t+1}=j|X,\theta^{(i)})}{\sum_{t=0}^{T-1}P(a_{z_t}=j|X,\theta^{(i)})}=\frac{\sum_{t=0}^{T-1}\xi(i,j)}{\sum_{t=0}^{T-1}\gamma_t(i)}$，$b_j(k)=\frac{\sum_{t=1}^{T}P(X,z_t=j,x_t=k|\theta^{(i)})}{\sum_{t=1}^{T}P(X,z_t=j|\theta^{(i)})}=\frac{\sum_{t=1}^{T}\gamma_t(j)1(x_t=k)}{\sum_{t=1}^{T}\gamma_t(j)}$.

```
def Baum-Welch(X, pi, A, B, pi0, A0, B0, max_iter)
    model.initialize(pi0, A0, B0)
    for i in range(max_iter):
        alpha = model.forward(X)
        beta = model.backward(X)
        gamma = model.compute_gamma(alpha, beta)
        xi = model.compute_xi(alpha, beta)
        pi = model.compute_pi(gamma)
        A = model.compute_a(gamman, xi)
        B = model.compute_b(gamma)
        model.update(pi, A, B)
    return model
```

在下图所示的贝叶斯网络中，我们想要通过变量消除法计算$P(Y|Z=z)$



初始所有的因子如下：

$$P(t),P(u\,|\,t),P(v\,|\,t),P(w\,|\,t),P(x\,|\,t),P(y\,|\,v,w),P(z\,|\,x)$$

当消除X时，我们产生一个新的因子$\tau_1(z,t)$，$\tau_1(z,t)=\sum P(x\,|\,t)P(z\,|\,x)$剩余因子如下：

$$P(t),P(u\,|\,t),P(v\,|\,t),P(w\,|\,t),P(y\,|\,v,w),\tau_1(z\,|\,t)$$

令变量消除法的顺序如下：X,T,U,V,W。

(a) 当消除T时，产生新的因子τ_2，请写出其表达式并列出所有剩余因子。

(b) 当消除U时，产生新的因子τ_3，请写出其表达式并列出所有剩余因子。

(c) 当消除V时，产生新的因子τ_4，请写出其表达式并列出所有剩余因子。

(d) 当消除W时，产生新的因子τ_5，请写出其表达式并列出所有剩余因子。

(e) 如何利用剩余因子计算$P(Y=y|Z=z)$?

(f) 因子大小是变量消除法计算复杂度的关键因素。例如，假设所有的变量都是二元变量，则因子$P(x\,|\,t)$和$\tau_1(z,t)$的大小都是2^2，它们都一共有2^2种取值需要维护。在以上变量消除的过程中，最大的因子是哪个，其大小是多少？是否存在更优的变量消除顺序，使得最大因子的大小优于以上顺序？

(a) $\tau_2(u,v,w,z)=\sum_t P(u|t)P(v|t)P(w|t)\tau_1(z,t)$，剩余因子为$P(y|v,w),\tau_2(u,v,z,t)$

(b) $\tau_3(v,w,z)=\sum_u\tau_2(u,v,w,t)$，剩余因子为$P(y|v,w),\tau_3(v,w,z)$

© $\tau_4(y,w,z)=\sum_v P(y|v,w)\tau_3(v,w,z)$，剩余因子为$\tau_4(y,w,z)$

(d) $\tau_5(y,z)=\sum_w\tau_4(y,w,z)$，剩余因子为$\tau_5(y,z)$

(e) 可以将$Z=z$先行代入，安装变量消除 的顺序依次解出新的因子，最后$\tau(Y,z)$在$Y=y$处的值即为$P(Y=y|Z=z)\propto 16$.

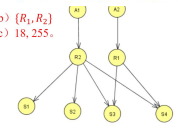
(f)最大的因子为$\tau_2(u,v,w,z)$，大小为$2^4=16$.存在，若按照U,V,W,T,X的顺序消除，则产生的因子大小依次为2, 4, 4, 4, 4，最大值为$\beta=\min(\beta,v)$

贝叶斯网络被广泛应用于医学诊断中，医生会通常根据患者的历史行为与症状判断病因。在新冠疫情流行期间，准确有效的医学检测是疫情防控的重中之重。对于新的病人，医生需要判断该病人患有普通肺炎(R_1)或新冠肺炎(R_2)。医生可以了解到该病人是否为密切接触者(A_1)，以及是否吸烟(A_2)，同时，可以获取肺部影像结果(S_1)，核酸检测结果(S_2)，是否干咳(S_3)以及是否呼吸困难(S_4)。已知密切接触者患有新冠肺炎的概率会更高，吸烟会提升患有普通肺炎的概率。新冠肺炎患者的四种症状均可能为阳性，而普通肺炎只可能导致干咳或者呼吸困难。

(a) 请画出最符合上述描述的贝叶斯网络图，并写出对应的条件概率乘积的分解形式。（3 分） (b) 请写出呼吸困难(S_4)的马尔可夫边界

(c) 请计算描述（1）中的贝叶斯网络，需要多少个独立参数？如果取消所有独立性假设，又需要多少个独立参数？（注：所有变量均为二值变量）（2 分）

(d) 在实际情况中，医生要根据所有的行为历史(A_1,A_2)与症状(S_1, S_2, S_3, S_4)判断病人是否患有新冠肺炎(R_2)。请写出对应该场景的条件概率，并利用贝叶斯公式与消元法写出计算过程。（7 分）



（b）$\{R_1,R_2\}$
（c）18, 255。

需要计算

$$P(R_2|A_1,A_2,S_1,S_2,S_3,S_4)=\frac{P(R_2,A_1,A_2,S_1,S_2,S_3,S_4)}{P(A_1,A_2,S_1,S_2,S_3,S_4)}$$

首先计算$P(R_2,A_1,A_2,S_1,S_2,S_3,S_4)$:

$$P(R_2,A_1,A_2,S_1,S_2,S_3,S_4)$$
$$=\sum_{R_1}P(S_1,S_2,S_3,S_4,R_1,R_2,A_1,A_2)$$
$$=\sum_{R_1}p(S_1|R_2)p(S_2|R_2)p(S_3|R_1,R_2)p(S_4|R_1,R_2)p(S_4|R_1,R_2)p(R_1|A_2)p(R_2|A_1)p(A_1)p(A_2)$$

将所有和R_1有关的因子提出，记做$\tau(R_2,S_3,S_4,A_2)$:

$$\tau(R_2,S_1,S_2,A_2)=\sum_{R_1}p(S_3|R_1,R_2)p(S_4|R_1,R_2)p(R_1|A_2)$$

故分子的消去结果为：

$$P(R_2,A_1,A_2,S_1,S_2,S_3,S_4)=p(R_2|A_1)p(A_1)p(A_2)p(S_1|R_2)p(S_2|R_2)\tau(R_2,S_3,S_4,A_2)$$

进一步计算分母：

$$p(A_1,A_2,S_1,S_2,S_3,S_4)$$
$$=\sum_{R_2}p(R_2,A_1,A_2,S_1,S_2,S_3,S_4)$$
$$=\sum_{R_2}p(R_2|A_1)p(A_1)p(A_2)p(S_1|R_2)p(S_2|R_2)\tau(R_2,S_3,S_4,A_2)$$

将与R_2有关的因子提出，记为$\tau(A_1,A_2,S_1,S_2,S_3,S_4)$:

$$\tau(A_1,A_2,S_1,S_2,S_3,S_4)=\sum_{R_2}p(R_2|A_1)p(S_1|R_2)p(S_2|R_2)\tau(R_2,S_3,S_4,A_2)$$

故分母消去的结果为：$p(A_1)p(A_2)\tau(A_1,A_2,S_1,S_2,S_3,S_4)$

因此最终该条件概率的消去结果为：$p(R_2|A_1,A_2,S_1,S_2,S_3,S_4)=\frac{p(R_2|A_1)p(S_1|R_2)p(S_2|R_2)\tau(R_2,S_3,S_4,A_2)}{\tau(A_1,A_2,S_1,S_2,S_3,S_4)}$

General Tree Search

```
1. function TREE-SEARCH(problem) returns a solution, or failure
2.   initialize the frontier as a specific work list (stack, queue, priority queue)
3.   add initial state of problem to frontier                    (1). Initialization
4.   loop do
5.     if the frontier is empty then
6.       return failure                                          Frontier: leaf nodes for expansion
7.     choose a node and remove it from the frontier
8.     if the node contains a goal state then                    Strategy: How to choose a leaf node to expand
9.       return the corresponding solution
10.                                                              (2). Selection
11.    for each resulting child from node
12.      add child to the frontier                               (3). Expansion
```

General Graph Search

```
1. function GRAPH-SEARCH(problem) returns a solution, or failure
2.   initialize the explored set to be empty
3.   initialize the frontier as a specific work list (stack, queue, priority queue)
4.   add initial state of problem to frontier                    (1). Initialization
5.   loop do
6.     if the frontier is empty then
7.       return failure                                          Strategy: How to choose a leaf node to expand
8.     choose a node and remove it from the frontier
9.     if the node contains a goal state then
10.      return the corresponding solution                       (2). Selection
11.    add the node state to the explored set                    Idea: never expand a state twice
12.    for each resulting child from node
13.      if the child state is not already in the frontier or explored set then
14.        add child to the frontier                             (3). Expansion
```

Alpha-Beta Pruning

Move Ordering

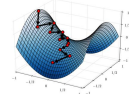
```
function MIN-VALUE(s, α, β) returns value
  if Terminal-Test(s) then return Utility(s)
  initialize v = +∞
  for each successor s' of state s:
    v = min(v, MAX-VALUE(s', α, β))
    if v ≤ α
      return v                          General case for alpha-beta pruning
    β = min(β, v)                       - If m is better than n for Player (n < m = α)
  return v                                we will never get to n in play
                                        - α: MAX's best option on path to root
                                        - β: MIN's best option on path to root
```

Worst ordering: $O(b^m)$ time
Best ordering: $O(b^{0.5m})$ time
Random ordering: $O(b^{0.75m})$ time

Minimax Efficiency
Time: $O(b^m)$
Space: $O(bm)$

SGD

- In each iteration t ($\leq T$):
  - Randomly sample a minibatch of $m \ll n$ points $\{(x_i,y_i)\}_{i=1}^m$
  - Set $J^t(\theta^t)=\frac{1}{m}\sum_{i=1}^m\ell(\theta^t;x_i,y_i)$
  - Compute gradient on minibatch: $\Delta^t=\nabla_\theta J^t(\theta^t)$
  - Update parameters with learning rate η: $\theta^{t+1}=\theta^t-\eta\Delta^t$



- For linear regression: $\boldsymbol{w}^{t+1}\leftarrow\boldsymbol{w}^t-2\eta\boldsymbol{X}_m^T(\boldsymbol{X}_m\boldsymbol{w}^t-\boldsymbol{y})$
  - Computational complexity $O(dmT)$. How about large dimension d?
- Theoretically, convergence rate is $O(1/\sqrt{T})$ under convex condition.

Basis Function

Feature map: $\boldsymbol{x}=(x_1,...,x_d)\in\mathcal{X}\xrightarrow{\Phi}\boldsymbol{z}=(z_1,...,z_{\tilde{d}})\in\mathcal{Z}$

$\{\phi_j\}_{0\leq j\leq\tilde{d}}$ is called basis functions. Radial basis functions (RBF)

Polynomial basis functions

1-D vector: $\boldsymbol{z}'=(1,x_1,x_1^2,x_1^3)$

2-D vector: $\boldsymbol{z}'=(1,x_2,x_1x_2,x_1^2)$

$$\phi_j(x)=\left\{\exp\left(-\frac{||x-\mu_j||_2^2}{2\sigma^2}\right):j=1,...,\tilde{d}\right\}$$

Softmax Regression: Cross-Entropy Loss

Loss for each data point (x,y):

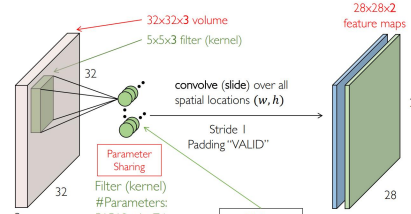
$$\ell(h(\boldsymbol{x}_i),y_i)=\begin{cases}-\log\left[\frac{\exp(\boldsymbol{w}_1^T\boldsymbol{x})}{\sum_{r=1}^C\exp(\boldsymbol{w}_r^T\boldsymbol{x})}\right] & y_i=1\\ -\log\left[\frac{\exp(\boldsymbol{w}_2^T\boldsymbol{x})}{\sum_{r=1}^C\exp(\boldsymbol{w}_r^T\boldsymbol{x})}\right] & y_i=2\\ \vdots & \\ -\log\left[\frac{\exp(\boldsymbol{w}_C^T\boldsymbol{x})}{\sum_{r=1}^C\exp(\boldsymbol{w}_r^T\boldsymbol{x})}\right] & y_i=C\end{cases}$$

Equivalent to maximum log-likelihood estimation

These two hypotheses have same outputs

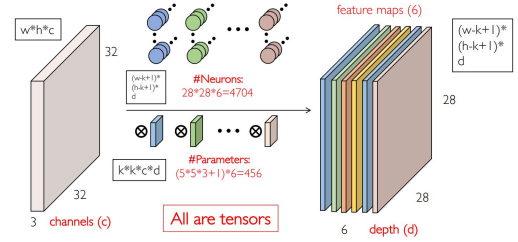
$$\begin{pmatrix}\boldsymbol{w}_1\\ \boldsymbol{w}_2\\ \cdots\\ \boldsymbol{w}_C\end{pmatrix}\quad\begin{pmatrix}\boldsymbol{w}_1+V\\ \boldsymbol{w}_2+V\\ \cdots\\ \boldsymbol{w}_C+V\end{pmatrix}$$

Convolution: Parameter Sharing



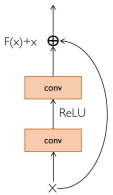
Not that no "sliding" in practice, we add neuron for each location (w,h)

Convolution: Multiple Feature Maps



We stack all feature maps up to get a "new image" (representation) of size 28×28×6.
(However, we cannot "see" the new image since it has more than 3 R/G/B channels)

ResNet



- Copying the learned layers from the shallower model
- Setting additional layers to identity mapping.

Solution:
Mapping F(x) = H(x) – x, instead of H(x) directly

Padding

Add P circles of zeros outside of the original map

$$\text{height}_{new}=\lceil(\text{height-filter+2*pad})/\text{stride}\rceil+1$$
$$\text{width}_{new}=\lceil(\text{width-filter+2*pad})/\text{stride}\rceil+1$$

Optimization of State Error Rate

- Viterbi algorithm is used to minimize the sequence error rate. Now, we focus on state error rate:

$$P(\boldsymbol{z}_t|\boldsymbol{x}_{1:T})=P(\boldsymbol{x}_{1:T},\boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T})$$
$$=P(\boldsymbol{x}_{1:t},\boldsymbol{z}_t)P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T})$$
$$=P(\boldsymbol{x}_{1:t},\boldsymbol{z}_t)P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T})$$
$$=P(\boldsymbol{x}_{1:t},\boldsymbol{z}_t)P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)/P(\boldsymbol{x}_{1:T})$$
$$\propto P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)$$

Conditional independency of HMM

$\equiv\alpha_t(\boldsymbol{z}_t)$, we evaluate it with forward algorithm

$\equiv\beta_t(\boldsymbol{z}_t)$, we evaluate it with backward algorithm

Forward Algorithm

- Physics meaning: estimate each hidden state online.

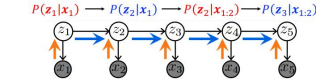
Define $\alpha_t(\boldsymbol{z}_t)=P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$

Prior distribution $P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})=\sum_{\boldsymbol{z}_{t-1}}P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\alpha_t(\boldsymbol{z}_{t-1})$

Posterior distribution $\alpha_t(\boldsymbol{z}_t)=\boldsymbol{c}_t P(\boldsymbol{x}_t|\boldsymbol{z}_t)P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})$

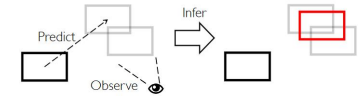
Normalizer

- Recursively proceed forward through the HMM network:



Forward Algorithm

- We have two inaccurate methods to estimate the hidden state of HMM:
  - From the new observation, using emission equation.
  - From current estimation, using transition equation.
- The forward algorithm "combines" the results of two methods!



Forward Algorithm: Prior Distribution

- Prior Distribution: Predict the next state before observation

$$P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})=\sum_{\boldsymbol{z}_{t-1}}P(\boldsymbol{z}_t,\boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1})\qquad\text{(Total probability)}$$
$$=\sum_{\boldsymbol{z}_{t-1}}P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1},\boldsymbol{x}_{1:t-1})P(\boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1})\qquad\boldsymbol{z}_t\text{只依赖}\boldsymbol{z}_{t-1}$$
$$=\sum_{\boldsymbol{z}_{t-1}}P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\alpha_{t-1}(\boldsymbol{z}_{t-1})$$

- Basic idea: The beliefs get pushed through the transition matrix.

Forward Algorithm: Posterior Distribution

- Posterior Distribution: After an observation, the belief is updated

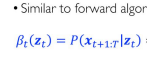
$$\alpha_t(\boldsymbol{z}_t)=P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})=\frac{P(\boldsymbol{z}_t,\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})}{P(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})}$$
$$\propto P(\boldsymbol{z}_t,\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})\qquad\text{(Normalize of }\boldsymbol{x}_t\text{)}$$
$$=P(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1},\boldsymbol{z}_t)P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})\qquad\text{条件概率+边缘分布}$$
$$=P(\boldsymbol{x}_t|\boldsymbol{z}_t)P(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})\qquad\text{(Markov dependency)}$$

- Basic idea: The beliefs reweighted by emission matrix.
- Due to the $\propto$, we need to renormalize after the above calculation.

Backward Algorithm

- Similar to forward algorithm, we can deduce backward algorithm:

$$\beta_t(\boldsymbol{z}_t)=P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t)=\sum_{\boldsymbol{z}_{t+1}}P(\boldsymbol{x}_{t+1:T},\boldsymbol{z}_{t+1}|\boldsymbol{z}_t)\qquad\text{加法公式}$$
$$=\sum_{\boldsymbol{z}_{t+1}}P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_{t+1},\boldsymbol{z}_t)P(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t)$$
$$=\sum_{\boldsymbol{z}_{t+1}}P(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_{t+1})P(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t)\qquad\text{条件独立性}$$
$$=\sum_{\boldsymbol{z}_{t+1}}P(\boldsymbol{x}_{t+2:T}|\boldsymbol{z}_{t+1})P(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{t+1})P(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t)\qquad\text{概率乘法公式}$$



Recursively run this algorithm backward

$\beta_{t+1}(\boldsymbol{z}_{t+1})$ Emission Transition

Example: Weather Forecasting

z_{t-1} \ z_t	Rain	Sun
Rain	0.7	0.3
Sun	0.1	0.9

z_t \ x_t	Umbrella	Not umbrella
Rain	0.9	0.1
Sun	0.2	0.8

$$P(\boldsymbol{z}_2|\boldsymbol{x}_1)=P(\boldsymbol{z}_2|\boldsymbol{z}_{1,\text{rain}})\alpha_1(\boldsymbol{z}_{1,\text{rain}})+P(\boldsymbol{z}_2|\boldsymbol{z}_{1,\text{sun}})\alpha_1(\boldsymbol{z}_{1,\text{sun}})$$
$$=\begin{pmatrix}0.3\\0.7\end{pmatrix}\times0.5+\begin{pmatrix}0.9\\0.1\end{pmatrix}\times0.5=\begin{pmatrix}0.6\\0.4\end{pmatrix}$$

$$\alpha_2(\boldsymbol{z}_2)\propto P(\boldsymbol{x}_2|\boldsymbol{z}_2)P(\boldsymbol{z}_2|\boldsymbol{x}_1)=\begin{pmatrix}0.2\\0.9\end{pmatrix}\times\begin{pmatrix}0.6\\0.4\end{pmatrix}=\begin{pmatrix}0.12\\0.36\end{pmatrix}$$

P(sun)=0.25
P(rain)=0.75

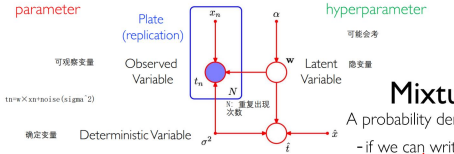
Example: Polynomial Regression

- The polynomial regression model, jointly showing also a new input value $\hat{x}$ together with the corresponding model prediction $\hat{t}$.

$$p(\hat{t}, \boldsymbol{t}_{1:N}, \boldsymbol{w}|\hat{\boldsymbol{x}}, \boldsymbol{x}_{1:N}, \alpha, \sigma^2) = \left[\prod_{n=1}^{N} p(t_n|x_n, \boldsymbol{w}, \sigma^2)\right] \times p(\boldsymbol{w}|\alpha) p(\hat{t}|\hat{\boldsymbol{x}}, \boldsymbol{w}, \sigma^2)$$

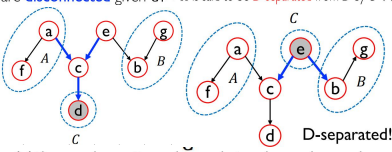
给定x_hat, t_hat发生的概率 | 随机变量 | 参数 | 通常认为是均值为w*x_hat方差为1.sim的高斯分布

parameter | hyperparameter



D-separation

All paths from any node in A (node set) to any node in B (node set) are disconnected given C,⇔ A is said to be D-separated from B by C ⇔ $(A \perp B) \mid C$

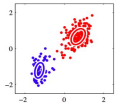


D-separated!

Although c is a head-to-head node, a and b are connected due to d is observed.

Gaussian Discriminant Analysis

- Alternative methods for dataset with all continuous features:
  - Using parametric distribution to represent $p(X = \boldsymbol{x}|Y = y)$.
- A common assumption in classification:
  - We always assume that data points in a class is a cluster.



- So we can model $p(X = \boldsymbol{x}|Y = y)$ by Gaussian distribution:

$$p(X = \boldsymbol{x}|Y = +1) \propto \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_+)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_+)\right)$$

$$p(X = \boldsymbol{x}|Y = -1) \propto \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_-)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_-)\right)$$

Usually share $\boldsymbol{\Sigma}$

We still model $p(Y = y)$ as Bernoulli distribution: $\text{Bernoulli}(\phi)$.

Now we use MLE to find the best parameter estimation:

$$\ell(\phi, \boldsymbol{\mu}_+, \boldsymbol{\mu}_-, \boldsymbol{\Sigma}) = \log \prod_{i=1}^{n} p(\boldsymbol{x}_i, y_i; \phi, \boldsymbol{\mu}_+, \boldsymbol{\mu}_-, \boldsymbol{\Sigma})$$

$$= \log \prod_{i=1}^{n} p(\boldsymbol{x}_i|y_i; \boldsymbol{\mu}_+, \boldsymbol{\mu}_-, \boldsymbol{\Sigma}) + \log \prod_{i=1}^{n} p(y_i|\phi)$$

The computing process is very similar to the process of Gaussian.

- The main difference is that $\boldsymbol{\mu}_+, \boldsymbol{\mu}_-$ are different.

$$\phi = \frac{\sum_{i=1}^{n} \mathbf{1}\{y_i = +1\}}{n}, \boldsymbol{\mu}_+ = \frac{\sum_{i=1}^{n} \mathbf{1}\{y_i = +1\} \boldsymbol{x}_i}{\sum_{i=1}^{n} \mathbf{1}\{y_i = +1\}}, \boldsymbol{\mu}_- = \frac{\sum_{i=1}^{n} \mathbf{1}\{y_i = -1\} \boldsymbol{x}_i}{\sum_{i=1}^{n} \mathbf{1}\{y_i = -1\}}$$

$$\boldsymbol{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu}_{y_i})(\boldsymbol{x}_i - \boldsymbol{\mu}_{y_i})^T$$

Gaussian Mixture Model

Parameters of Gaussian Mixture Model (GMM):

- Cluster probabilities: $\pi = (\pi_1, \ldots, \pi_k)$.
- Cluster means: $\mu = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k)$.
- Cluster covariance matrices: $\Sigma = (\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k)$

Density: $p(\boldsymbol{x}) = \sum_{z=1}^{k} \pi_z \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$

Generating process of GMM:

- First generate cluster index:
  - $z \sim (\pi_1, \ldots, \pi_k)$
- Then generate data:
  - $x \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$.

Mixture Distribution

A probability density $p(x)$ represents a mixture distribution

- if we can write it as a convex combination of probability densities:
- where $w_i \geq 0, \sum_{i=1}^{k} w_i = 1$, and each p_i is a probability density.

$$p(x) = \sum_{i=1}^{k} w_i p_i(x)$$

More constructively, let S be a set of probability distributions:

- Choose a distribution randomly from S.
- Sample $\boldsymbol{x}$ from the chosen distribution.

- Then $\boldsymbol{x}$ has a mixture distribution.

Expectation-Maximization (EM): Algorithm

Choose initial θ^{old}.

Expectation Step

Let $q^*(z) = p(z|x, \theta^{\text{old}})$. [$q^*$ gives best lower bound at θ^{old}]

Let $J(\theta) \coloneqq \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log\left(\frac{p(x, z|\theta)}{q^*(z)}\right)$

Expectation w.r.t. $z \sim q^*(z)$

Maximization Step

$\theta^{\text{new}} = \underset{\theta}{\text{argmax}}\, J(\theta)$ You can use SGD

Go to the Expectation Step, until converged.

EM for MAP

- Suppose we have a prior $p(\theta)$.
- Want to find MAP estimate: $\hat{\theta}_{\text{MAP}} = \underset{\theta}{\text{argmax}}\, p(\theta|x)$

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

$$\log p(\theta|x) = \log p(x|\theta) + \log p(\theta) - \log p(x)$$

- Still can use our evidence lower bound on $\log p(x, \theta)$.

$$J(\theta) \coloneqq \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log\left(\frac{p(x, z|\theta)}{q^*(z)}\right)$$

- Maximization step becomes $\theta^{\text{new}} = \underset{\theta}{\text{argmax}}\, [J(\theta) + \log p(\theta)]$

GMM: E-Step

- Denote probability (responsibility) that $\boldsymbol{x}_i$ comes from cluster j by $\gamma_i^j = p(z = j|\boldsymbol{x} = \boldsymbol{x}_i, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ $q^*(z) = p(z|x, \theta^{\text{old}})$
  - The vector $(\gamma_i^1, \ldots, \gamma_i^k)$ is exactly the soft assignment for $\boldsymbol{x}_i$.
- From probabilistic computation:

$$\gamma_i^j = p(z = j|\boldsymbol{x}_i) = \frac{p(z = j, \boldsymbol{x}_i)}{p(\boldsymbol{x}_i)} = \frac{\pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{c=1}^{k} \pi_c \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}$$

- If we know $\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \pi_j$ for all clusters $j = 1, \ldots k$, then easy to compute:

$$\gamma_i^j = \frac{\pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{c=1}^{k} \pi_c \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}$$

GMM: M-Step

$$\underset{\theta}{\text{argmax}} \mathcal{L}(q^*, \theta) = \underset{\theta}{\text{argmax}} \sum_z q^*(z) \log\left(\frac{p(x, z|\theta)}{q^*(z)}\right)$$

$\theta = \{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ $= \underset{\theta}{\text{argmax}} \sum_z p(z|x, \theta^{\text{old}}) \log(p(x, z|\theta))$

- So we have the loss function for Gaussian Mixture Model parameters:

By MLE $\underset{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}}{\text{argmax}} \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_i^j \log[\pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)]$

- Let $n_c = \sum_{i=1}^{n} \gamma_i^c$ be the number of points soft-assigned to cluster c.

$$\pi_c^{\text{new}} = \frac{n_c}{n}, \boldsymbol{\mu}_c^{\text{new}} = \frac{1}{n_c} \sum_{i=1}^{n} \gamma_i^c \boldsymbol{x}_i, \boldsymbol{\Sigma}_c^{\text{new}} = \frac{1}{n_c} \sum_{i=1}^{n} \gamma_i^c (\boldsymbol{x}_i - \boldsymbol{\mu}_c^{\text{new}})(\boldsymbol{x}_i - \boldsymbol{\mu}_c^{\text{new}})^T$$

EM for GMM: Overview

Initialize parameters $\boldsymbol{\pi}, \boldsymbol{\Sigma}, \boldsymbol{\mu}$.

$$\gamma_i^j = \frac{\pi_j \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{c=1}^{k} \pi_c \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}$$

E-step. Evaluate all responsibilities using current parameters:

M-step. Re-estimate the parameters using the responsibilities:

$$\pi_c^{\text{new}} = \frac{n_c}{n} \quad \boldsymbol{\mu}_c^{\text{new}} = \frac{1}{n_c} \sum_{i=1}^{n} \gamma_i^c \boldsymbol{x}_i, \boldsymbol{\Sigma}_c^{\text{new}} = \frac{1}{n_c} \sum_{i=1}^{n} \gamma_i^c (\boldsymbol{x}_i - \boldsymbol{\mu}_c^{\text{new}})(\boldsymbol{x}_i - \boldsymbol{\mu}_c^{\text{new}})^T$$

Repeat E-step and M-step until log-likelihood converges.

EM and Variational Methods

- When E-step is difficult:
  - Hard to take expectation w.r.t. $q^*(z) = p(z|x, \theta^{\text{old}})$
  - For example, hierarchical latent variable models (next lectures).
- Solution: Restrict to distributions Q that are easy to work with.
- The evidence lower bound (ELBO) now looser:

$$q^* = \underset{q \in Q}{\text{argmin}}\, \text{KL}\left[q(z), p(z|x, \theta^{\text{old}})\right]$$

- Find an easy-to-work variational distribution q^* to approximate the inference distribution $p(z|x, \theta^{\text{old}})$.
  - This group of methods are called variational methods.

Dirichlet-Multinomial Model

- Assume that we have a vocabulary of V words.
- Generative Process

What is the dimension of $\boldsymbol{\theta}$ here?

$\theta \sim \text{Dir}(\alpha)$ [draw distribution over words]
For each word $n \in \{1, \ldots, N\}$ [draw word]
$w_n \sim \text{Mult}(1, \theta)$ [draw word]

Categorical Distribution

- Example corpus

the	he	is	the	and	the	she	she	is
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9

- You will see the benefit of using conjugate prior for a distribution.

Dirichlet-Multinomial Mixture Model

- Generative Process

We have K topics.

Topics → Documents → Mixture

We have V words.

- Example corpus

the	he	is	the	and	the	she	she	is
w_{11}	w_{12}	w_{13}	w_{21}	w_{22}	w_{23}	w_{31}	w_{32}	w_{33}	w_{34}

Document 1 | Document 2 | Document 3

Dirichlet-Multinomial Mixture Model

What is the dimension of $\boldsymbol{\theta}$?
What is the dimension of $\boldsymbol{\beta_k}$?

- Generative Process

For each topic $k \in \{1, \ldots, K\}$: [draw distribution over words]
$\boldsymbol{\beta}_k \sim \text{Dir}(\eta)$ [draw distribution over topics]
$\boldsymbol{\theta} \sim \text{Dir}(\alpha)$
For each document $d \in \{1, \ldots, D\}$ [draw topic assignment]
$z_d \sim \text{Mult}(1, \boldsymbol{\theta})$
For each word $n \in \{1, \ldots, N_d\}$ [draw word]
$w_{dn} \sim \text{Mult}(1, \boldsymbol{\beta}_{z_d})$

- Example corpus

the	he	is	the	and	the	she	she	is
w_{11}	w_{12}	w_{13}	w_{21}	w_{22}	w_{23}	w_{31}	w_{32}	w_{33}	w_{34}

Document 1 | Document 2 | Document 3

Topics → Documents → Mixture

Mixture vs. Admixture

Topics → Documents → Admixture

There should be multiple topics in a document:
Each document can be modeled by a new distribution of topics!

Latent Dirichlet Allocation (LDA)

Each document has a single topic z

Dirichlet-Multinomial Mixture Model

For each topic $k \in \{1, \ldots, K\}$: [draw distribution over words]
$\boldsymbol{\beta}_k \sim \text{Dir}(\eta)$ [draw distribution over topics]
$\boldsymbol{\theta} \sim \text{Dir}(\alpha)$
For each document $d \in \{1, \ldots, D\}$ [draw topic assignment]
$z_d \sim \text{Mult}(1, \boldsymbol{\theta})$
For each word $n \in \{1, \ldots, N_d\}$ [draw word]
$w_{dn} \sim \text{Mult}(1, \boldsymbol{\beta}_{z_d})$

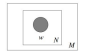
Each document has a topic distribution $\boldsymbol{\theta}$
Each word has a topic z

Latent Dirichlet Allocation

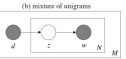
For each topic $k \in \{1, \ldots, K\}$: [draw distribution over words]
$\boldsymbol{\beta}_k \sim \text{Dir}(\eta)$ [draw distribution over topics]
For each document $d \in \{1, \ldots, D\}$
$\boldsymbol{\theta}_d \sim \text{Dir}(\alpha)$
For each word $n \in \{1, \ldots, N_d\}$ [draw topic assignment]
$z_{dn} \sim \text{Mult}(1, \boldsymbol{\theta}_d)$ [draw word]
$w_{dn} \sim \text{Mult}(1, \boldsymbol{\beta}_{z_{dn}})$

Probabilistic Graphical Model

- Box with number on the bottom right:
  - Hierarchical samples and Number of samples.
- White/Gray circle: Hidden/Observed Variables.
- Arrow: Generating path (conditional prob.).
- Read (b):
  - We generate M samples of z.
  - Each contains N kid samples w.
  - E.g. M clusters, each with N samples in it.
  - w is observed, but z is hidden.
- Can you read (c) Probabilistic Latent Semantic Indexing (PLSI) now?

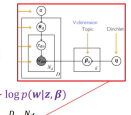

(a) unigram


(b) mixture of unigrams


(c) pLSI/aspect model

Maximum Likelihood Estimation

- If we know all the latent variables:
  - (We do not know them in fact.)
  - Let's try to compute MLE as an exercise!



$$\log p(\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\eta})$$

$$= \log p(\boldsymbol{\beta}|\boldsymbol{\eta}) + \log p(\boldsymbol{\theta}|\boldsymbol{\alpha}) + \log p(\boldsymbol{z}|\boldsymbol{\theta}) + \log p(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\beta})$$

$$= \sum_{k=1}^{K} \log p(\vec{\boldsymbol{\beta}}_k|\boldsymbol{\eta}) + \sum_{d=1}^{D} \log p(\vec{\boldsymbol{\theta}}_d|\boldsymbol{\alpha}) + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \log p(z_{d,n}|\vec{\boldsymbol{\theta}}_d)$$

$$+ \sum_{d=1}^{D} \sum_{n=1}^{N_d} \log p(w_{d,n}|z_{d,n}, \vec{\boldsymbol{\beta}}_{1:K})$$

PGM to simplify learning and inference!

- Recall the form of Dirichlet Distribution: $p(\vec{\boldsymbol{\theta}}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$

$$= \sum_{k=1}^{K} \left(\sum_{v=1}^{V} (\eta_v - 1) \log \beta_{kv} - \log B(\boldsymbol{\eta})\right) + \sum_{d=1}^{D} \sum_{k=1}^{K} (\alpha_k - 1) \log \theta_{dk} - \log B(\boldsymbol{\alpha})$$

$$+ \sum_{d=1}^{D} \sum_{n=1}^{N_d} \log \theta_{d,z_{d,n}} + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \log \beta_{z_{d,n} w_{d,n}}$$

- We will meet some more complex terms later 😊
- We need to solve for parameters $\boldsymbol{\alpha}, \boldsymbol{\eta}$.

Newton Method

$$\underset{\boldsymbol{\eta}}{\text{argmax}} \sum_{k=1}^{K} \left(\sum_{v=1}^{V} (\eta_v - 1) \log \beta_{kv} - \log B(\boldsymbol{\eta})\right)$$

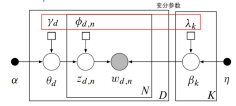
- Suppose E-step is difficult:
  - Hard to take expectation w.r.t. $q^*(z) = p(z|x, \theta^{\text{old}})$
  - For example, topic models.
- Solution: Restrict to distributions Q that are easy to work with.
- The ELBO now looser: Intractable! We compute this

$$q^* = \underset{q \in Q}{\text{argmin}}\, \text{KL}\left[q(z), p(z|x, \theta^{\text{old}})\right] = \underset{q \in Q}{\text{argmax}}\, \mathcal{L}(q, \theta^{\text{old}})$$

- Find an easy-to-work variational distribution q^* to approximate the inference distribution $p(z|x, \theta^{\text{old}})$.
  - This group of methods are called Variational Methods.

Variational Inference for LDA

- Mean field assumption in LDA:



$$q(\vec{\boldsymbol{\theta}}_{1:D}, \boldsymbol{z}_{1:D,1:N_d}, \vec{\boldsymbol{\beta}}_{1:K}) = \prod_{k=1}^{K} q(\vec{\boldsymbol{\beta}}_k|\vec{\lambda}_k) \prod_{d=1}^{D} q(\vec{\boldsymbol{\theta}}_d|\vec{\gamma}_d) \prod_{d=1}^{D} \prod_{n=1}^{N} q(z_{d,n}|\vec{\boldsymbol{\phi}}_{d,n})$$

Dirichlet Distribution | Dirichlet Distribution | Multinomial Distribution

- In LDA, we have known the form of q, and what left is:

$$p(\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\eta}) = p(\boldsymbol{\beta}|\boldsymbol{\eta}) p(\boldsymbol{\theta}|\boldsymbol{\alpha}) p(\boldsymbol{z}|\boldsymbol{\theta}) p(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\beta})$$

- The objective function $\mathcal{L}$ in VI of LDA (Solvable!) turns out to be:

$$\mathcal{L} = \sum_{k=1}^{K} E_q[\log p(\vec{\boldsymbol{\beta}}_k|\boldsymbol{\eta})] + \sum_{d=1}^{D} E_q[\log p(\vec{\boldsymbol{\theta}}_d|\vec{\boldsymbol{\alpha}})] +$$

$$\sum_{d=1}^{D} \sum_{n=1}^{N} E_q[\log p(Z_{d,n}|\vec{\boldsymbol{\theta}}_d)] +$$

$$\sum_{d=1}^{D} \sum_{n=1}^{N} E_q[\log p(w_{d,n}|Z_{d,n}, \vec{\boldsymbol{\beta}}_{1:K})] - E_q \log q$$

- We solve one of the terms as an example:

All latent variables are gone!

$$E_q[\log p(\boldsymbol{\theta}|\boldsymbol{\alpha})]$$

$$= E_q[\log\left\{\exp\left\{\left(\sum_{i=1}^{K} (\alpha_i - 1) \log \theta_i\right) + \log \Gamma\left(\sum_{i=1}^{K} \alpha_i\right) - \sum_{i=1}^{K} \log \Gamma(\alpha_i)\right\}\right\}],$$

$$= \left(\sum_{i=1}^{K} (\alpha_i - 1) E_q[\log \theta_i]\right) + \log \Gamma\left(\sum_{i=1}^{K} \alpha_i\right) - \sum_{i=1}^{K} \log \Gamma(\alpha_i),$$

$$= \left(\sum_{i=1}^{K} (\alpha_i - 1)(\Psi(\gamma_i) - \Psi(\sum_j \gamma_j))\right) + \log \Gamma\left(\sum_{i=1}^{K} \alpha_i\right) - \sum_{i=1}^{K} \log \Gamma(\alpha_i).$$

- Ψ is the first derivative of the log Gamma function Γ.
- Iterate the following steps (Descend each variable iteratively):

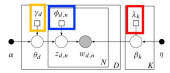
(1) For each topic k and term v:

$$\lambda_{k,v}^{(t+1)} = \eta + \sum_{d=1}^{D} \sum_{n=1}^{N} \mathbf{1}(w_{d,n} = v) \phi_{n,k}^{(t)}$$ 隐变量对应的变分分布

(2) For each document d:

(a) Update γ_d:

$$\gamma_{d,k}^{(t+1)} = \alpha_k + \sum_{n=1}^{N} \phi_{d,n,k}^{(t)}$$



(b) For each word n, update $\vec{\phi}_{d,n}$:

$$\vec{\phi}_{d,n}^{(t+1)} \propto \exp\left\{\Psi(\gamma_{d,k}^{(t+1)}) + \Psi(\lambda_{k,w_n}^{(t+1)}) - \Psi(\sum_{w=1}^{V} \lambda_{k,v}^{(t+1)})\right\},$$

where Ψ is the digamma function, the first derivative of the log Γ function.

$\sum_{k=1}^{V} \phi_{dnk} = 1$

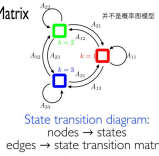
Digamma function Ψ can only be computed by approximation, e.g. Taylor

- Markov chain is defined via a state transition matrix:
  - Assume the state is discrete with K possible outcomes:

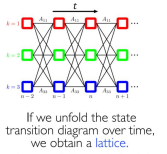
$$A_{ij} = P(x_{t+1,j} = 1 | x_{t,i} = 1) \quad i, j \in \{1, 2, \cdots, K\}$$

对j求和为1

State Transition Matrix



State transition diagram: nodes → states, edges → state transition matrix



If we unfold the state transition diagram over time, we obtain a lattice.

Viterbi Algorithm

In HMMs, Viterbi algorithm is used for minimizing sequence error rate.

利用贝叶斯决策准则把误差最小转化为最大化后验

Shannon Award (1991)

$$\tilde{\boldsymbol{Z}} = \underset{\boldsymbol{Z}}{\arg\max}\, P(\boldsymbol{Z}|\boldsymbol{X}) \propto \underset{\boldsymbol{Z}}{\arg\max}\, P(\boldsymbol{Z}, \boldsymbol{X})$$

图模型定义

$$= \underset{\boldsymbol{Z}}{\arg\max}\left[P(\boldsymbol{z}_1)\left[\prod_t P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\right] \cdot \left[\prod_t P(\boldsymbol{x}_t|\boldsymbol{z}_t)\right]\right]$$

Transition | Emission

取负对数似然

$$= \underset{\boldsymbol{Z}}{\arg\min}\left[\phi_1(\boldsymbol{z}_1) + \sum_{t=2}^{T} (\phi_t(\boldsymbol{z}_t) + \psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}))\right]$$

minimizing negative log-probability.

高斯分布

$$\phi_t(\boldsymbol{z}_t) = \begin{cases} -\log P(\boldsymbol{z}_1|\boldsymbol{z}_1) - \log P(\boldsymbol{z}_1) & t = 1 \\ -\log P(\boldsymbol{z}_t|\boldsymbol{z}_t) & t \geq 2 \end{cases}$$

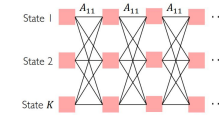
多变分布 i由于离

$$\psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}) = -\log P(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$$

Lattice Diagram

$$\tilde{\boldsymbol{Z}} = \underset{\boldsymbol{Z}}{\arg\min}\left[\phi_1(\boldsymbol{z}_1) + \sum_{t=2}^{T} (\phi_t(\boldsymbol{z}_t) + \psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}))\right]$$

- Node weight: $\phi_t(\boldsymbol{z}_t)$, edge weight: $\psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1})$
- Row for each possible state, column for each time step.
- State transition diagram determines allowable paths (K^T in total).
- MAP estimate is the shortest path through weighted edges in the graph.



Viterbi Algorithm

$$\tilde{\boldsymbol{Z}} = \underset{\boldsymbol{Z}}{\arg\min}\left[\phi_1(\boldsymbol{z}_1) + \sum_{t=2}^{T} (\phi_t(\boldsymbol{z}_t) + \psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}))\right]$$

- Define cost of shortest path ending with a specified state $\boldsymbol{z}_t$:

$$f_t(\boldsymbol{z}_t) = \min_{\boldsymbol{Z}} \left[\phi_1(\boldsymbol{z}_1) + \sum_{t'=2}^{t} (\phi_{t'}(\boldsymbol{z}_{t'}) + \psi_{t'}(\boldsymbol{z}_{t'}, \boldsymbol{z}_{t'-1}))\right]$$

- We can recursively write this minimization as follows:

$$f_t(\boldsymbol{z}_t) = \phi_t(\boldsymbol{z}_t) + \min_{\boldsymbol{z}_{t-1}} [\psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}) + f_{t-1}(\boldsymbol{z}_{t-1})]$$

- Interpretation: every path up to time t has the following form:
  1. A path to some state at time $t - 1$.
  2. A cost for the transition from time $t - 1$ to time t.

Dynamical programming

Initiation: $f_1(\boldsymbol{z}_1) = \phi_1(\boldsymbol{z}_1)$

At time step $t - 1$, keep the shortest path and corresponding cost $f_{t-1}(\boldsymbol{z}_{t-1})$ ending with each state $\boldsymbol{z}_{t-1}$:

From time t to $t + 1$:

- Evaluate the weights $\psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1})$ of every edge from t to $t + 1$.
- K paths arriving at each node k, $\mathcal{O}(K^2)$ computation cost in total.

Update the best path ending with each state $\boldsymbol{z}_t$.

$$f_t(\boldsymbol{z}_t) = \phi_t(\boldsymbol{z}_t) + \min_{\boldsymbol{z}_{t-1}} [\psi_t(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}) + f_{t-1}(\boldsymbol{z}_{t-1})]$$

Algorithm stop:

- When reaching the end T, we have K paths through this lattice.
- Select the best one as the final choice.

Overall computation cost: $\mathcal{O}(TK^2)$

7 伯努利混合模型的EM 算法（15分）

考虑$x=(x_1,\cdots,x_D)\in\{0,1\}^D$，设$P(x_d=1)=p_d, 1\le d\le D$，令$p=(p_1,\cdots,p_D)$，则$x$ 服从如下伯努利分布$P(x\mid p)=\prod_{d=1}^{D}p_d^{x_d}(1-p_d)^{1-x_d}$。

现考虑K 个伯努利分布的混合，混合分布参数为$\boldsymbol{p}=\left\{p^{(1)},\cdots,p^{(K)}\right\}$，对应权重为$\boldsymbol{\pi}=\{\pi_1,\cdots,\pi_K\}$。在混合分布中采样$X=\left\{x^{(i)}\right\}_{i=1,\cdots,n}$，则有

$$P\left(x^{(i)}\mid\boldsymbol{p},\boldsymbol{\pi}\right)=\sum_k\pi_kP\left(x^{(i)}\mid p^{(k)}\right).$$

考虑EM算法中的Expectation步骤，对于$x^{(i)}$，引入隐变量$z^{(i)}\in\{0,1\}^K$，如果$x^{(i)}$取自第k个伯努利分布，则有$z_k^{(i)}=1$，否则为0。

(a) 请说明下面公式：

$$P\left(z^{(i)}\mid\boldsymbol{\pi}\right)=\prod_{k=1}^{K}\pi_k^{z_k^{(i)}},$$

$$P\left(x^{(i)}\mid z^{(i)},\boldsymbol{p},\boldsymbol{\pi}\right)=\prod_{k=1}^{K}\left[P\left(x^{(i)}\mid p^{(k)}\right)\right]^{z_k^{(i)}}.$$

(b) 对应$X=\left\{x^{(i)}\right\}_{i=1,\cdots,n}$，令$Z=\left\{z^{(i)}\right\}_{i=1,\cdots,n}$，请基于(a)中结论，试证明数据与隐变量的似然$P(Z,X\mid\boldsymbol{p},\boldsymbol{\pi})$ 为

$$P(Z,X\mid\boldsymbol{p},\boldsymbol{\pi})=\prod_{i=1}^{n}\left[\prod_{k=1}^{K}\left[P\left(x^{(i)}\mid p^{(k)}\right)\right]^{z_k^{(i)}}\right]\left[\prod_{k=1}^{K}\pi_k^{z_k^{(i)}}\right].$$

(c) 令$\eta\left(z_k^{(i)}\right)=\mathbb{E}\left[z_k^{(i)}\mid x^{(i)},\boldsymbol{p},\boldsymbol{\pi}\right]$，试证明

$$\eta\left(z_k^{(i)}\right)=\frac{\pi_k\prod_{d=1}^{D}\left(p_d^{(k)}\right)^{x_d^{(i)}}\left(1-p_d^{(k)}\right)^{1-x_d^{(i)}}}{\sum_{j=1}^{K}\pi_j\prod_{d=1}^{D}\left(p_d^{(j)}\right)^{x_d^{(i)}}\left(1-p_d^{(j)}\right)^{1-x_d^{(i)}}}.$$

(d) 令$\tilde{\boldsymbol{p}},\tilde{\boldsymbol{\pi}}$ 作为我们想要最大化的新参数，$\boldsymbol{p}$ 和$\boldsymbol{\pi}$ 来自上一次迭代，基于(b)(c)中结论，试证明Maximization步骤为最大化下面的期望：

$$\begin{aligned}&\mathbb{E}[\log P(Z,X\mid\tilde{\boldsymbol{p}},\tilde{\boldsymbol{\pi}})\mid X,\boldsymbol{p},\boldsymbol{\pi}]\\&=\sum_{i=1}^{n}\sum_{k=1}^{K}\eta\left(z_k^{(i)}\right)\left[\log\tilde{\pi}_k+\sum_{d=1}^{D}\left(x_d^{(i)}\log\tilde{p}_d^{(k)}+\left(1-x_d^{(i)}\right)\log\left(1-\tilde{p}_d^{(k)}\right)\right)\right]\end{aligned}$$

(a) $x^{(i)}$必然恰好取K个伯努利分布的其中一个，由于这K个分布对应权重分别为$\{\pi_1,\cdots,\pi_K\}$，取在第k个的概率为π_k。此时只有$z_{k_0}^{(i)}=1$，其余$K-1$维上均为0，作为指数时幂的取值为1，$\therefore P(z^{(i)}|\pi)=\pi_{k_0}=\prod_{k=1}^{K}\pi_k^{z_k^{(i)}}$.

同理，$P(x^{(i)}|z^{(i)},p,\pi)=P(x^{(i)}|p^{(k_0)})=\prod_{k=1}^{K}[P(x^{(i)}|p^{(k)})]$，即$z^{(i)},p,\pi$的条件下满足$x=x^{(i)}$的$x$必然来自某个具体的第$k_0$个伯努利分布。

(b) 通过似然的表示及简单代入可得，$P(Z,X,|p,\pi)=p(z^{(i)}|\pi)\prod_{i=1}^{n}P(x^{(i)}|z^{(i)},p,\pi)=\prod_{k=1}^{K}[\prod_{k=1}^{K}[P(x^{(i)}|p^{(k)})]^{z_k^{(i)}}][\prod_{k=1}^{K}\pi_k^{z_k^{(i)}}]$.

© 注意到$z_k^{(i)}=k$当且仅当$x^{(i)}$取自第k个伯努利分布，故$\eta(z_k^{(i)})=E[z_k^{(i)}|x^{(i)},p,\pi]=p(k|x^{(i)},p,\pi)=\frac{\pi_kp(x^{(i)}|p^{(k)})}{\sum_{j=1}^{K}\pi_jp(x^{(i)}|p^{(k)})}$，由伯努利分布的形式，有$p(x^{(i)}|p^{(k)})=\prod_{d=1}^{D}((p_d^{(k)})^{x_d^{(i)}}(1-p_d^{(k)})^{1-x_d^{(i)}})$，代入期望表达式中即得$E[z_k^{(i)}|x^{(i)},p,\pi]=\frac{\pi_k\prod_{d=1}^{D}((p_d^{(k)})^{x_d^{(i)}}(1-p_d^{(k)})^{1-x_d^{(i)}})}{\sum_{j=1}^{K}\pi_j\prod_{d=1}^{D}((p_d^{(k)})^{x_d^{(i)}}(1-p_d^{(k)})^{1-x_d^{(i)}})}$.

(d) 对(b)中的$P(Z,X|p,\pi)$取对数，有$\log P(Z,X|p,\pi)=\sum_{i=1}^{n}\log\prod_{k=1}^{K}[P(x^{(i)}|p^{(k)})]^{z_k^{(i)}}+\sum_{k=1}^{K}z_k^{(i)}\log\pi_k=\sum_{k=1}^{K}z_k^{(i)}(\log\pi_k+\sum_{i=1}^{n}\log P(x^{(i)}|p^{(k)}))$，其中$\log P(x^{(i)}|p^{(k)})=\sum_{d=1}^{D}(x_d\log p_d+(1-x_d)\log(1-p_d))$，恰与结果中后一方括号内求和的通项一致，而整体求和中的$z_k^{(i)}$项使得和恰好等于其中使得$x^{(i)}$取第k个伯努利分布的第k项，故$\log P(Z,X|p,\pi)$与结果中方括号内的形式一致（只是p和π为个别取值而非均值），而$E[\log P(Z,X|\tilde{p},\tilde{\pi})|X,p,\pi]=\sum_{j=1}^{K}\eta(z_k^{(i)})\log P(Z,X|p,\pi)$，故$E[\log P(Z,X|\tilde{p},\tilde{\pi})|X,p,\pi]=\sum_{i=1}^{n}\sum_{k=1}^{K}\eta(z_k^{(i)})[\log\tilde{\pi}_k+\sum_{d=1}^{D}(x_d^{(i)}\log\tilde{p}_d^{(k)}+(1-x_d^{(i)})\log(1-\tilde{p}_d^{(k)}))]$，即求得该形式。