Parameter learning for MRFs

假定我们有一个马尔科夫随机场，现在有 C 个 clique。可以理解为 C 个传感器来感知这个 graph，每个传感器能理解一部分 graph 的知识。而每个传感器是 log-linear 的，而 combine 这些传感器ϕ_c的参数θ_c是模型的参数，也就是这个模型可以看作是一个$M(\theta_1,\theta_2,\ldots\theta_C)$ ，只要我们得到了每一个参数，那么这个模型能产生什么数据也确定了。或者说如果观察到了这个模型的数据，反过来能算出每一个参数。

首先我们给出已知θ，y 的条件概率：

$$p(y|\theta)=\frac{1}{Z(\theta)}\exp\left(\sum_c\theta_c\phi_c(y)\right)$$ 【1】

举个例子，假定我们有 2 个样本，y1 和 y2，和三个参数$\theta_1,\theta_2,\theta_3$。
那么

$p(y1|<\theta_1,\theta_2,\theta_3>)\ \approx\ \exp(\theta_1\phi_c(y1)+\theta_2\phi_c(y1)+\theta_3\phi_c(y1))$

$p(y2|<\theta_1,\theta_2,\theta_3>)\ \approx\ \exp(\theta_1\phi_c(y2)+\theta_2\phi_c(y2)+\theta_3\phi_c(y2))$

我们假定

$\theta_1\phi_c(y1)+\theta_2\phi_c(y1)+\theta_3\phi_c(y1)=A$ ；

$\theta_1\phi_c(y2)+\theta_2\phi_c(y2)+\theta_3\phi_c(y2)=B$

则有

$p(y1|<\theta_1,\theta_2,\theta_3>)$ = exp(A) / {exp(A)+exp(B)}

$p(y2|<\theta_1,\theta_2,\theta_3>)$ = exp(B) / {exp(A)+exp(B)}

其中$Z(\theta)$ = exp(A)+exp(B)

样本的 log—likelihood 为

$$\ell(\theta)=\frac{1}{N}\sum_i\ \log(p(y_i|\theta)=\frac{1}{N}\sum_i\ \left[\sum_c\theta_c\phi_c(y_i)-\log(Z(\theta))\right]$$ 【2】

求每个θ_c的偏导(梯度)

$$\frac{\partial\ell(\theta)}{\partial(\theta_c)}=\frac{1}{N}\sum_i\ \left[\phi_c(y_i)-\frac{\partial\log(Z(\theta))}{\partial(\theta_c)}\right]$$ 【3】

$$\frac{\partial\log(Z(\theta))}{\partial(\theta_c)}=\sum_{y_i}\phi_c(y_i)\,p(y_i|\theta)$$ 【4】

上面这个$\log(Z(\theta))$的求偏导有点费解，我们举个例子

如上例子：

$p(y1|<\theta_1,\theta_2,\theta_3>)$ = exp(A) / {exp(A)+exp(B)}

$p(y2|<\theta_1,\theta_2,\theta_3>)$ = exp(B) / {exp(A)+exp(B)}

$Z(\theta)$ = exp(A)+exp(B)

则有 $\log(Z(\theta))$ = log(exp(A)+exp(B))

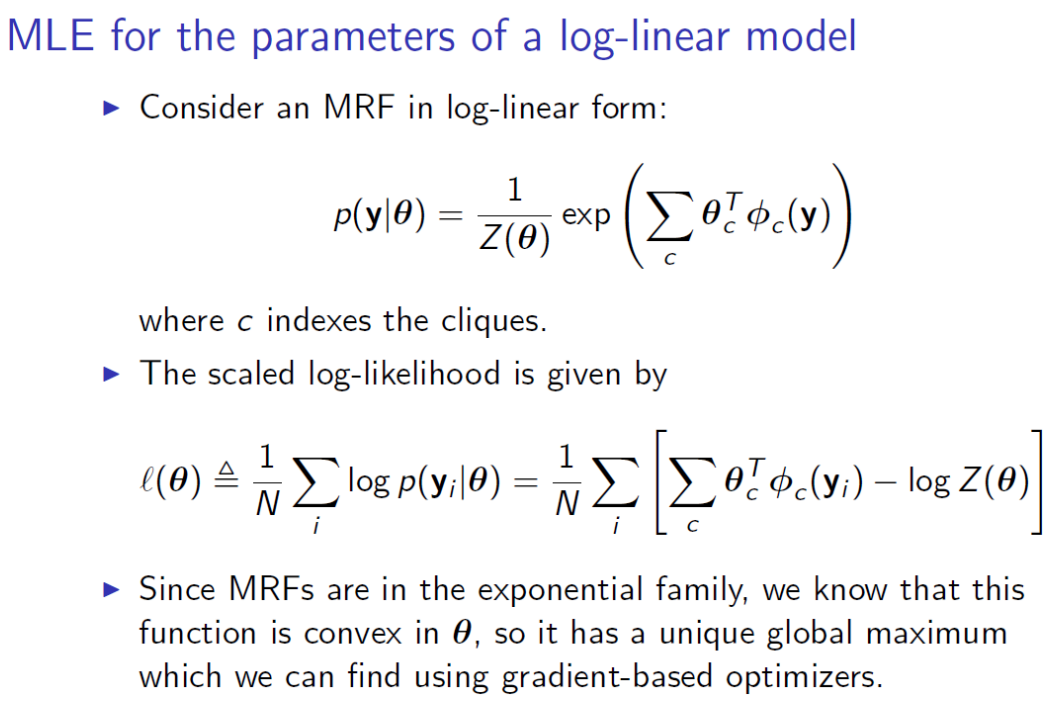
$$\frac{\partial\log(Z(\theta))}{\partial(\theta_1)}=\frac{\exp(A)*\phi_1(y1)+\exp(B)*\phi_1(y2)}{\exp(A)+\exp(B)}$$
$$=p(y1|<\theta_1,\theta_2,\theta_3>)*\phi_1(y1)+p(y2|<\theta_1,\theta_2,\theta_3>)*\phi_1(y2)$$

由公式【3】【4】得到θ_c的梯度为

$$\frac{\partial \ell(\theta)}{\partial(\theta_c)} = [\frac{1}{N}\sum_i \phi_c(y_i)] - [\sum_{y_i} \phi_c(y_i)\, \mathrm{p}(y_i|\theta)]$$

也就是每个参数θ_c的梯度,等于它 touch 到的这个传感器的均值 – 这个传感器的期望。就是它每次梯度下降的 step。

美帝的几个 slides 如下



MLE for the parameters of a log-linear model

- Consider an MRF in log-linear form:

$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{y})\right)$$

where c indexes the cliques.

- The scaled log-likelihood is given by

$$\ell(\boldsymbol{\theta}) \triangleq \frac{1}{N}\sum_i \log p(\mathbf{y}_i|\boldsymbol{\theta}) = \frac{1}{N}\sum_i \left[\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{y}_i) - \log Z(\boldsymbol{\theta})\right]$$

- Since MRFs are in the exponential family, we know that this function is convex in $\boldsymbol{\theta}$, so it has a unique global maximum which we can find using gradient-based optimizers.

Gradient descent

- In particular, the derivative for the weights of a particular clique, c, is given by

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N}\sum_i \left[\phi_c(\mathbf{y}_i) - \frac{\partial}{\partial \boldsymbol{\theta}_c}\log Z(\boldsymbol{\theta})\right]$$

- One can show that the derivative of the log partition function wrt $\boldsymbol{\theta}_c$ is the expectation of the c'th feature under the model, i.e.,

$$\frac{\partial \log Z(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_c} = \mathbb{E}\left[\phi_c(\mathbf{y})|\boldsymbol{\theta}\right] = \sum_{\mathbf{y}} \phi_c(\mathbf{y}) p(\mathbf{y}|\boldsymbol{\theta})$$

  where

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \left[\frac{1}{N}\sum_i \phi_c(\mathbf{y}_i)\right] - \mathbb{E}\left[\phi_c(\mathbf{y})\right]$$

- In the first term, we fix $\mathbf{y}$ to its observed values; this is sometimes called the **clamped term**. In the second term, $\mathbf{y}$ is free; this is sometimes called the **unclamped term** or

Moment matching

- The gradient of the log likelihood can be rewritten as the expected feature vector according to the empirical distribution minus the model's expectation of the feature vector:

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \mathbb{E}_{p_{\text{emp}}}\left[\phi_c(\mathbf{y})\right] - \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}\left[\phi_c(\mathbf{y})\right]$$

- At the optimum, the gradient will be zero, so the empirical distribution of the features will match the model's predictions:

$$\mathbb{E}_{p_{\text{emp}}}\left[\phi_c(\mathbf{y})\right] = \mathbb{E}_{p(\cdot|\boldsymbol{\theta})}\left[\phi_c(\mathbf{y})\right]$$

  This is called **moment matching**.

一个例子：

Probabilistic model of English spelling

- For trigram features, a table has $26^3 = 17,576$ parameters in it. However, most of these triples will never occur.
- An alternative approach is to define indicator functions that look for certain "special" triples, such as "ing", "qu-", etc. Then we can define the potential on each trigram as follows:

$$\psi(y_{t-1}, y_t, y_{t+1}) = \exp(\sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1}))$$

  where k indexes the different features, corresponding to "ing", "qu-", etc., and ϕ_k is the corresponding binary **feature function**.
- By tying the parameters across locations, we can define the probability of a word of any length using

$$p(\mathbf{y}|\boldsymbol{\theta}) \propto \exp(\sum_t \sum_k \theta_k \phi_k(y_{t-1}, y_t, y_{t+1}))$$

Feature induction for UGMs

- Consider the following method for learning features for a log-linear model of English spelling

  ```
  "Inducing features of random fields",
  S. Della Pietra and V. Della Pietra and J. Lafferty
  PAMI 1997
  ```
- Greedily add feature which bests improves likelihood. Features are atomic or conjunctions of previous features.
- Initially the model has no features, which represents the uniform distribution.
- The algorithm starts by choosing to add the feature

$$\phi_1(\mathbf{y}) = \sum_t \mathbb{I}(y_t \in \{a, \ldots, z\})$$

  which checks if any letter is lower case or not.

First feature

- After the feature is added, the parameters are (re)-fit by maximum likelihood (see later). For this feature, it turns out that $\hat{\theta}_1 = 1.944$, which means that a word with a lowercase letter in any position is about $e^{1.944} \approx 7$ times more likely than the same word without a lowercase letter in that position.
- Some samples from this model, generated using (annealed) Gibbs sampling (see later) are shown below.

```
m, r, xevo, ijjiir, b, to, jz, gsr, wq, vf, x, ga, msm
pcp, d, oziVlal, hzagh, yzop, io, advzmxnv,
emx, kayerf, mlj, rawzyb, jp, ag, ctdnnnbg, wgdw, t, kg
cy, spxcq, uzflbbf, dxtkkn, cxwx, jpd, ztzh, lv, zhpkvr
zcngotcnx, igcump, zjcjs, lqpWiqu, cefmfhc, o, lb, fdc
yopxmvk, by, fz,, t, govyccm, ijyiduwfzo, 6xr, duh, ej
pjw, l, fl, w
```

Next few features

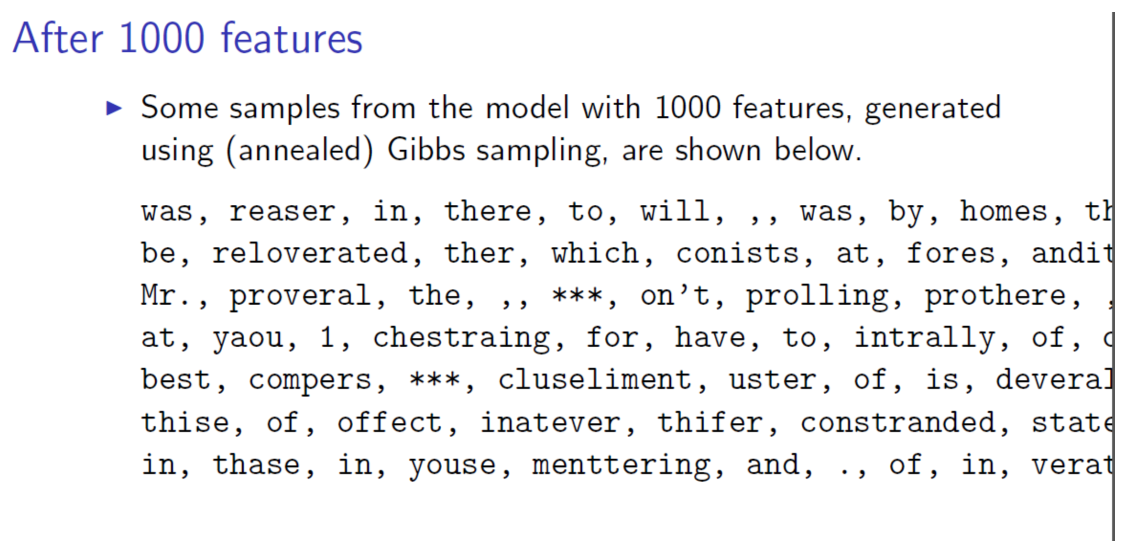
- The second feature added by the algorithm checks if two adjacent characters are lower case:

$$\phi_2(\mathbf{y}) = \sum_{s \sim t} \mathbb{I}(y_s \in \{a, \ldots, z\}, y_t \in \{a, \ldots, z\})$$

  Now the model has the form

$$p(\mathbf{y}) = \frac{1}{Z} \exp(\theta_1 \phi_1(\mathbf{y}) + \theta_2 \phi_2(\mathbf{y}))$$

- Continuing in this way, the algorithm adds features for the strings `s>` and `ing>`, where $>$ represents the end of word, and for various regular expressions such as `[0-9]`, etc.



After 1000 features

- Some samples from the model with 1000 features, generated using (annealed) Gibbs sampling, are shown below.

```
was, reaser, in, there, to, will, ,, was, by, homes, th
be, reloverated, ther, which, conists, at, fores, andit
Mr., proveral, the, ,, ***, on't, prolling, prothere, ,
at, yaou, 1, chestraing, for, have, to, intrally, of, c
best, compers, ***, cluseliment, uster, of, is, deveral
thise, of, offect, inatever, thifer, constranded, state
in, thase, in, youse, menttering, and, ., of, in, verat
```

我的试验（只用了一个特征）

通过试验可以看到，在迭代足够多次数后，每个样本的 likelihood 都是 1/5。而且参数也稳定收敛。

https://github.com/pennyliang/MachineLearning-C---code/blob/master/MRF/main.cpp