確率的HPSG・パラメータ推定

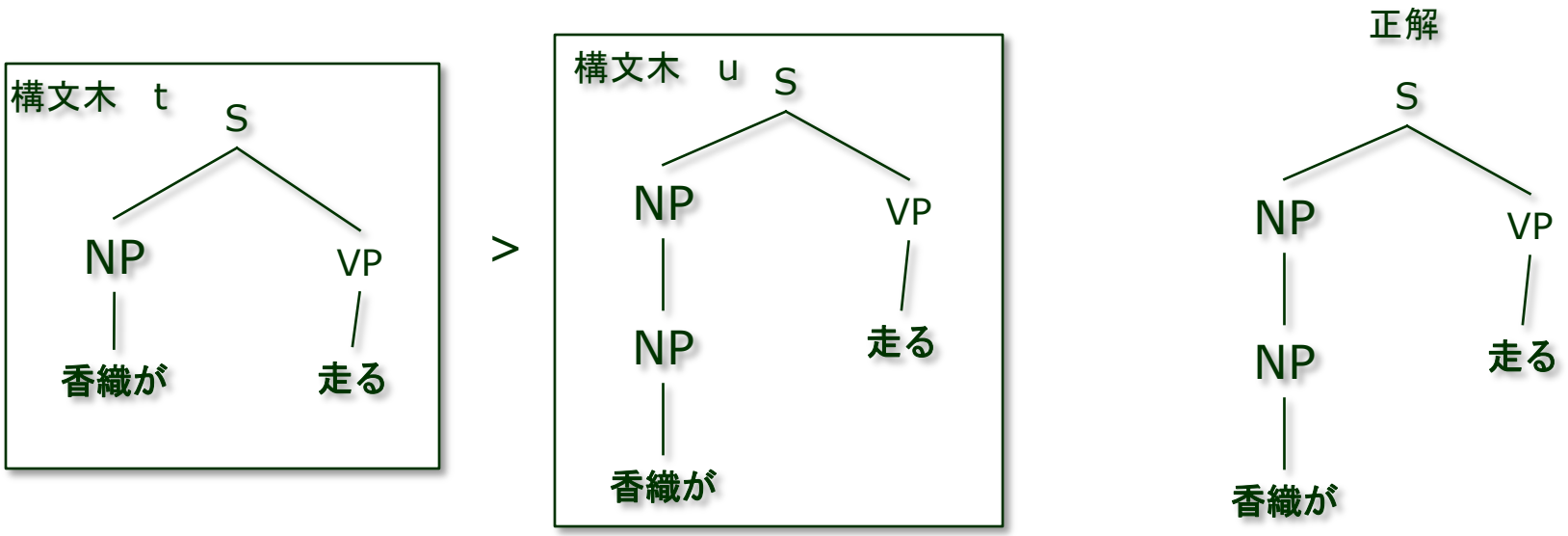
二宮　崇

今日の講義の予定

- 識別モデル
- 確率的HPSG
- 最大エントロピーモデル (多クラスロジスティック回帰)
- 最適化
  - GIS, IIS, CG
  - パーセプトロン
- 教科書
  - Yusuke Miyao (2006) From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model, Ph.D Thesis, University of Tokyo
  - Jun'ichi Kazama (2004) Improving Maximum Entropy Natural Language Processing by Uncertainty-aware Extensions and Unsupervised Learning, Ph.D. Thesis, University of Tokyo
  - 北研二(著) 辻井潤一(編) 言語と計算4 確率的言語モデル 東大出版会
  - Jorge Nocedal, Stephen Wright (1999) "Numerical Optimization" Springer, 1st edition 1999, 2nd edition 2006
  - Cristopher M. Bishop "PATTERN RECOGNITION AND MACHINE LEARNING" Springer, 2006

PCFGの問題

- 独立性の仮定



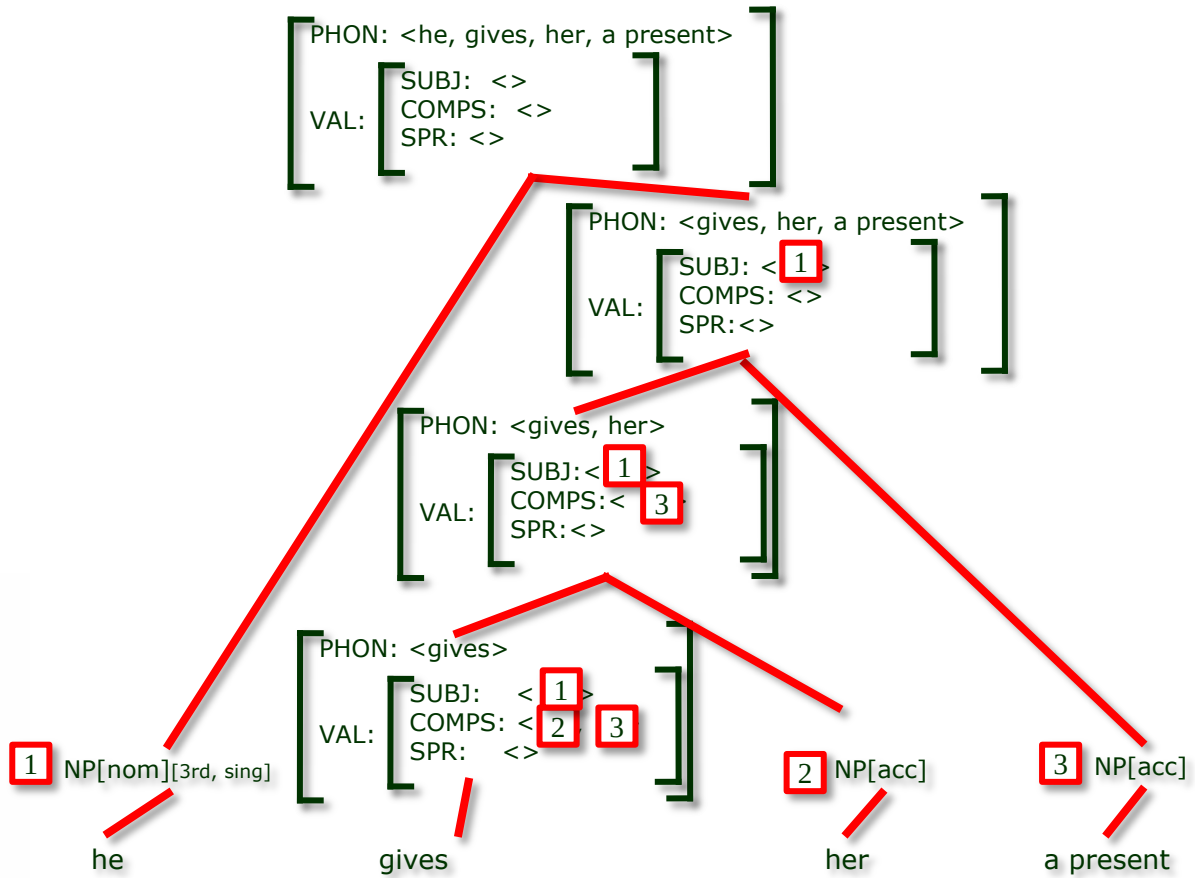
$P(t) = \theta_{S \to NP\ NP} \times \theta_{NP \to 香織} \times \theta_{V \to 走る}$

$P(u) = \theta_{S \to NP\ NP} \times \theta_{NP \to NP} \times \theta_{NP \to 香織} \times \theta_{V \to 走る}$

常にP(t) > P(u) →正解がuであっても必ずtが選ばれる

HPSGの確率モデル？

- PCFG: 各書換規則に対応するパラメータ
- HPSG: ？？



生成モデルから識別モデルへ

- 識別モデル

  直接

$$\tilde{t} = \arg\max_{t} p(t \mid s; \theta)$$

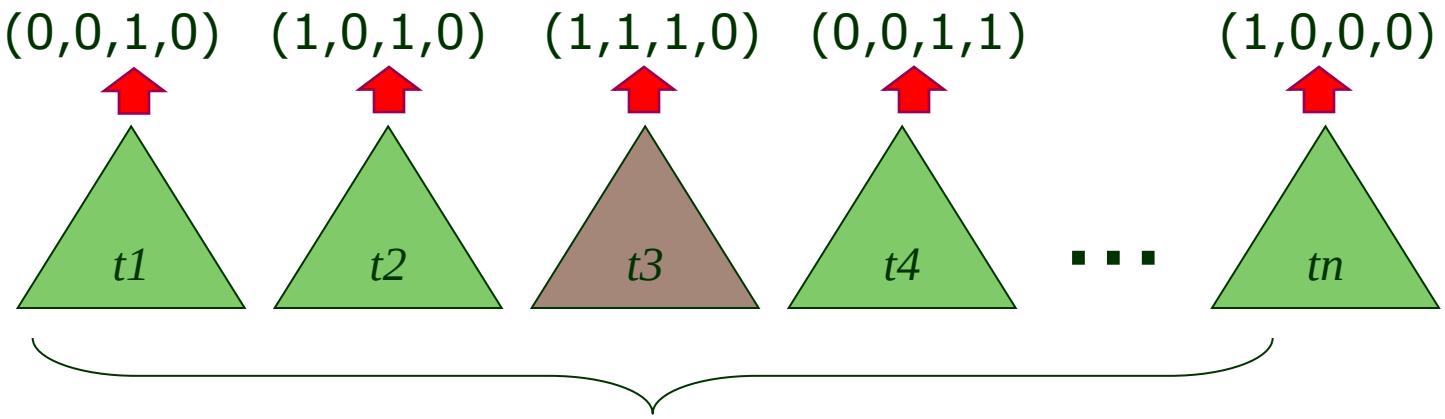
  を解く

  - 独立な事象を仮定しない
  - 「条件部の確率」をモデルにいれない

識別モデル

$p(t \mid s)$ s = "A blue eye girl with white hair and skin walked"

素性ベクトル
(特徴ベクトル)

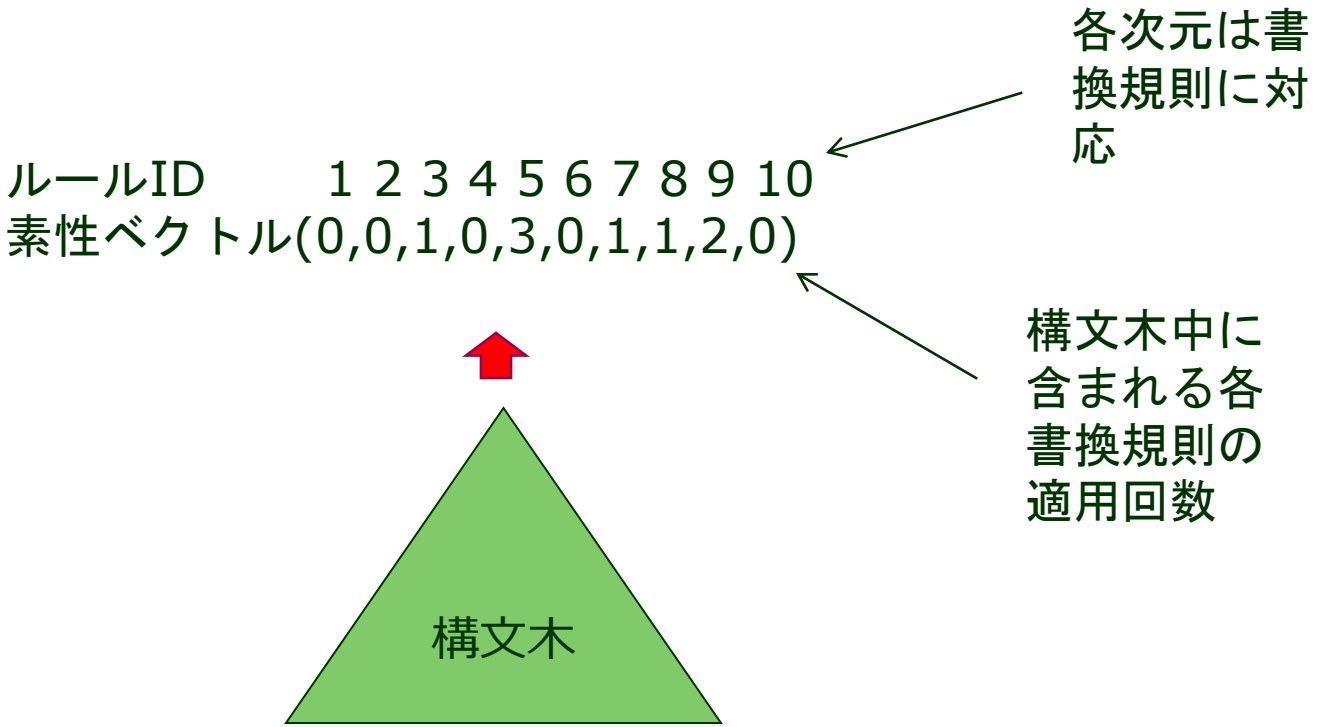


文法Gによりsから導出出来る全ての構文木集合

$p(t3|s)$ はt1,t2,t3,..,tnからt3を選択する確率

CFGの識別モデルの例

- 構文木生成に用いられた各書換規則の適用回数

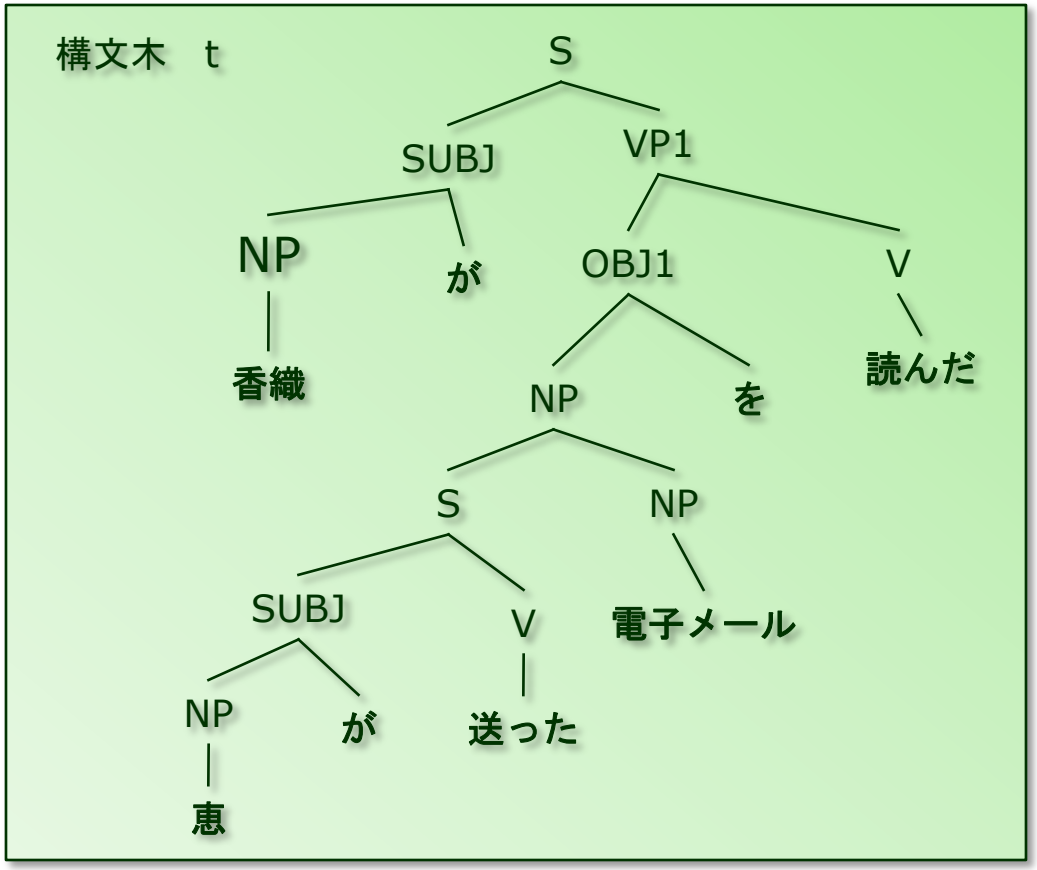


構文木の素性ベクトル

簡単なCFGの例	ID
S → SUBJ　VP1	1
S → SUBJ　V	2
SUBJ → NP　が	3
VP1 → OBJ1　V	4
OBJ1 → NP　を	5
NP → S　NP	6
V → 送った	7
V → 読んだ	8
NP → 香織	9
NP → 恵	10
NP → 電子メール	11
NP → プレゼント	12
NP → 香織　NP1	13
NP → 恵　NP1	14
NP1 → と　NP	15

ID 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

素性ベクトル(1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0)

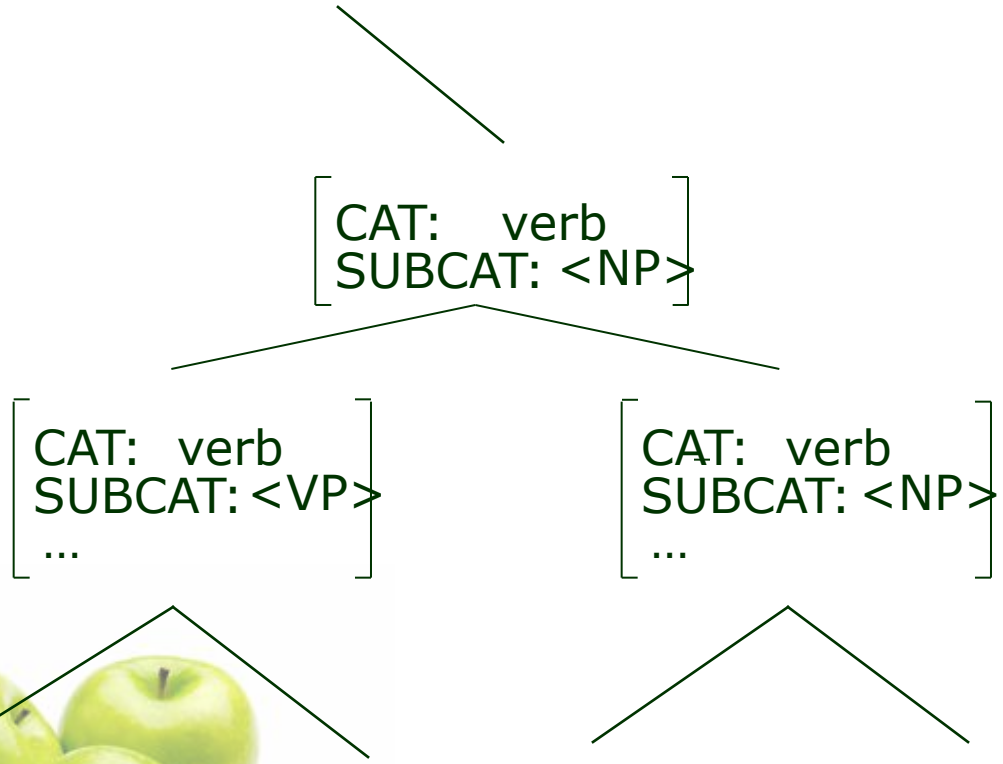


識別モデルのいいところ

- 独立性を仮定していない
  - (戦略として) 思いつく限りいろんな素性をいれる
  - 訓練データに対してより良い予測ができる
  - 逆にoverfittingする可能性がある
    - c.f. 正規分布の事前分布によるMAP推定でoverfittingを緩和
- CFGなら、ルールだけでなく、head wordなどいろんな素性をいれれば良い
- 疎なベクトルなら数百万次元ぐらい

確率的HPSG

- 「....を満たすブランチ（分岐）はいくつあるか？」という素性の集合



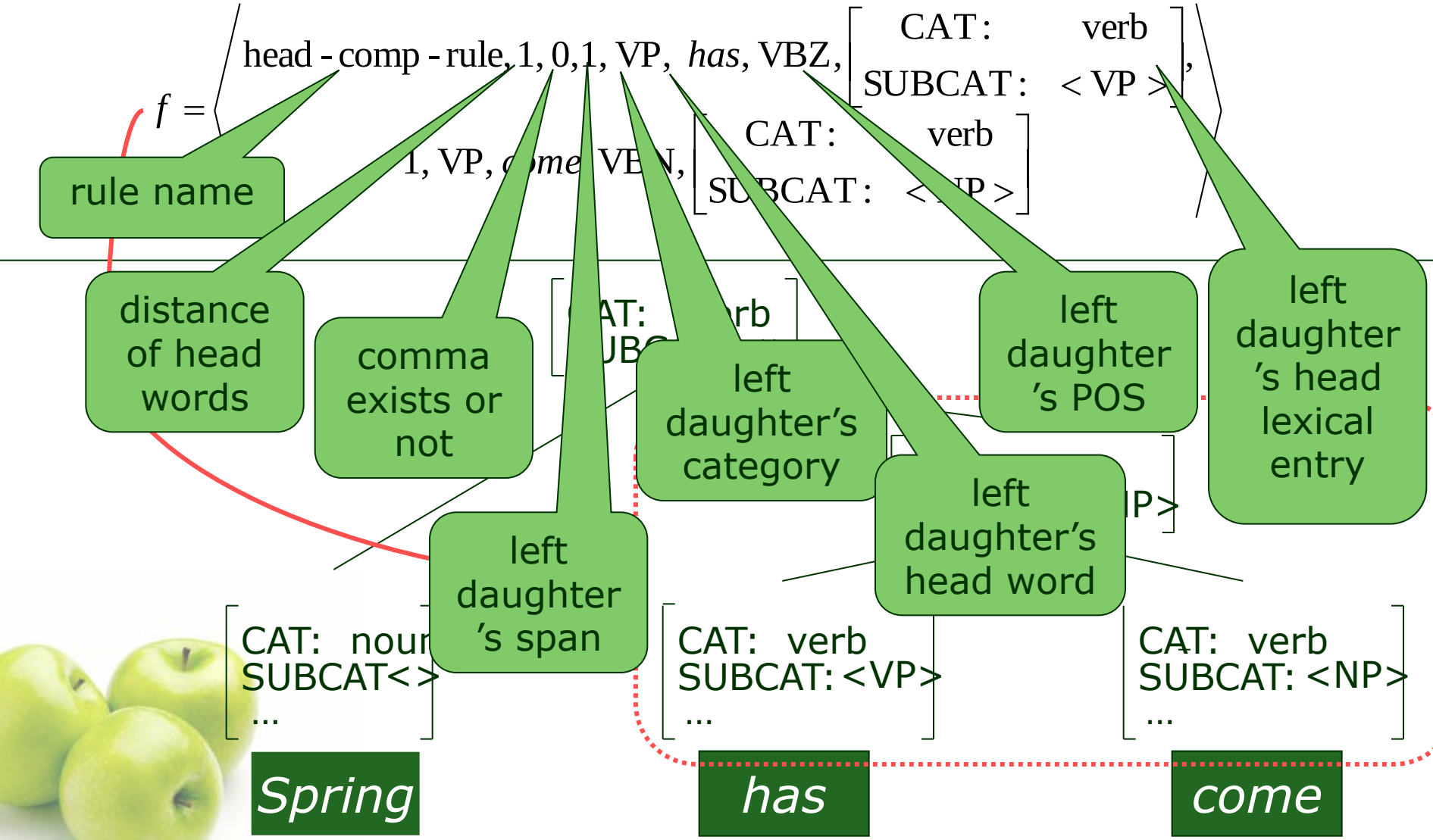
親のcatがverbで左娘のcatがverbで右娘のcatがverbであるか？
→yes
→ +1

親のcatがverbで左娘のcatがnounで右娘のcatがverbであるか？
→no
→+0

確率的HPSG

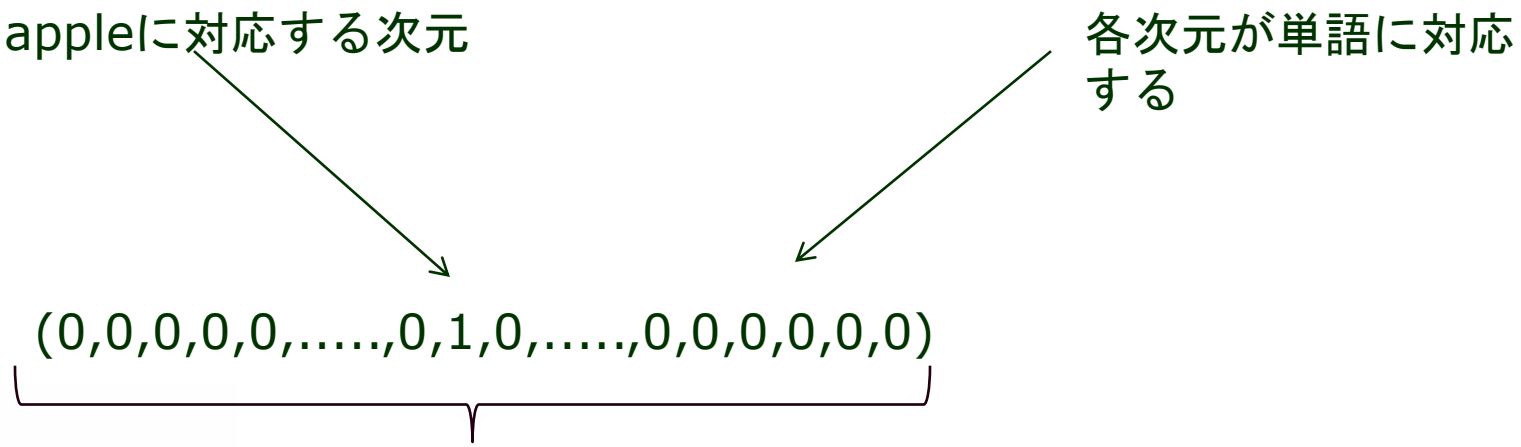
- ブランチの周辺状況を素性にしている
- 親のカテゴリーと左娘のカテゴリーと右娘のカテゴリーの全ての組み合わせを列挙して素性にすれば、先ほどの例のCFGと同じ素性になる
- カテゴリーだけでなく、head wordや、距離などいろいろな素性をいれられる

確率的HPSGの素性の実例



素性に関する注意その1

- 単語の素性と素性値
  - 例: head wordが``apple''であった時の素性値



(訓練データに出現した)単語の数だけ次元がある！

素性に関する注意その2

- 素性の組み合わせ
  - 最大エントロピー法(ロジスティック回帰)では、素性同士の共起情報が別素性として自動的に組み込まれるわけではない
    - 右娘と左娘のcatが同時にverb
    - SVM: 多項式カーネル
  - 素性の組み合わせを手で指示しないといけない⇒自動的に行うなら「素性選択」を行う

素性に関する注意その2: 確率的HPSGの素性組み合わせの実例

RULE	DIST	COMMA	SPAN	SYM	WORD	POS	LE
✔	✔	✔			✔	✔	✔
✔	✔	✔			✔	✔	
✔	✔	✔			✔		✔
✔	✔	✔		✔	✔		
✔		✔	✔		✔	✔	✔
✔		✔	✔		✔	✔	
✔		✔	✔		✔		✔
✔		✔	✔	✔	✔		
✔	✔	✔				✔	✔
✔	✔	✔				✔	
✔	✔	✔					✔
✔	✔	✔		✔			
✔		✔	✔			✔	✔
✔		✔	✔			✔	
✔		✔	✔				✔
✔		✔	✔	✔			

識別モデルの学習

問題設定

- x: 入力
- y: 出力
- 訓練データ
  - (x_i, y_i) i=1,...,N
  - 例
    - xは文で、yはxに対する正解の構文木
    - xは競馬情報で、yは1位の馬
- 問題
  - ある未知の入力xに対する出力yの予測

素性関数

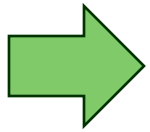
- 入力や出力から特徴を抽出する素性関数 (feature function) を複数定義
  - $f_j(x, y)$ j=1,...,M
  - 注意
    - 人手で定義
    - Mは特にいくつでもかまわないが、増やした分だけ計算時間・空間がかかったり、overfittingしてしまう
    - 良い素性関数をできるだけたくさん見つける、ということが人間がしなくてはいけない重要な仕事
- 素性ベクトル (または特徴ベクトル, feature vector)
  - $(f_1(x,y), f_2(x,y), ..., f_M(x, y))$

全体の流れ(1/2)

- Estimation (推定、パラメータ推定)
  - 各素性 f_j に対する重み λ_j を学習

訓練データ

入力	出力
x_1	y_1
x_2	y_2
…	…
x_N	y_N



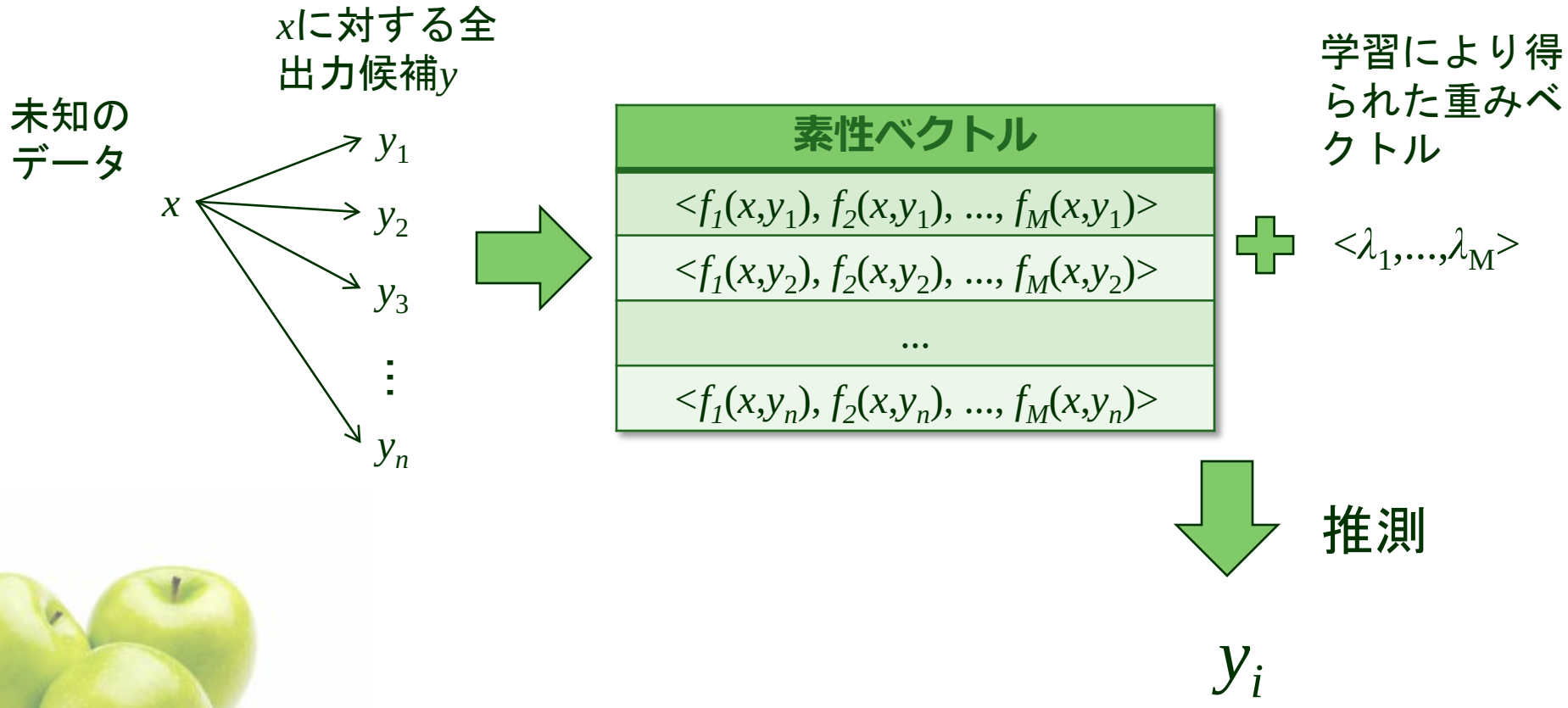
素性ベクトル
$<f_1(x_1,y_1), f_2(x_1,y_1), \ldots, f_M(x_1,y_1)>$
$<f_1(x_2,y_2), f_2(x_2,y_2), \ldots, f_M(x_2,y_2)>$
…
$<f_1(x_N,y_N), f_2(x_N,y_N), \ldots, f_M(x_N,y_N)>$

学習

$<\lambda_1, \lambda_2, \ldots, \lambda_M>$

全体の流れ(2/2)

- Inference (推測、推定)
  - 未知のデータxに対する出力yの推定



最大エントロピーモデル (Maximum Entropy model)
多クラスロジスティック回帰 (Multi-class Logistic Regression)
対数線形モデル (Log-linear Model)

- 確率モデル

重み

素性関数

$$p(y \mid x;\lambda) = \frac{1}{Z(x,\lambda)} \exp\left(\sum_j \lambda_j f_j(x,y)\right)$$

ただし

分配関数
(Partition function)

$$Z(x,\lambda) = \sum_{y' \in Y(x)} \exp\left(\sum_j \lambda_j f_j(x,y')\right)$$

直感的理解

- スコアの対数=各素性の(値×重み)の和
- p(y | x)= (xyのスコア)/(xに対する候補集合y'のスコアの和)

$$s(x,y,\lambda)=\exp\left(\sum_j \lambda_j f_j(x,y)\right)$$ とおくと、

$$p(y\mid x;\lambda)=\frac{1}{Z(x,\lambda)}\exp\left(\sum_j \lambda_j f_j(x,y)\right)=\frac{s(x,y,\lambda)}{\sum_{y'\in Y(x)} s(x,y',\lambda)}$$

となる。ちなみに、

$$s(x,y,\lambda)=e^{\lambda_1 f_1(x,y)+\lambda_2 f_2(x,y)+\cdots+\lambda_M f_M(x,y)}=e^{\lambda_1 f_1(x,y)}e^{\lambda_2 f_2(x,y)}\cdots e^{\lambda_M f_M(x,y)}$$

$$\log s(x,y,\lambda)=\sum_j \lambda_j f_j(x,y)$$

パラメータ推定

- 訓練データに対する対数尤度

$$\log\left(\prod_{i=1}^{N} p(y_i \mid x_i;\lambda)\right)$$

$$= \sum_{i=1}^{N} \log p(y_i \mid x_i;\lambda)$$

$$= \sum_{i=1}^{N} \log \frac{1}{Z(x_i,\lambda)} \exp\left(\sum_j \lambda_j f_j(x_i,y_i)\right)$$

$$= -\sum_{i=1}^{N} \log Z(x_i,\lambda) + \sum_{i=1}^{N}\sum_j \lambda_j f_j(x_i,y_i)$$

$$\log ab = \log a + \log b$$

$$\log_e \exp(x) = \log_e e^x = x$$

Zはパラメータを含むexpの足し算になっているから、これの極値を求めるのは難しい…

パラメータ推定

EMの時と
同じ

- パラメータ更新式に変形
  - 新しいパラメータと古いパラメータによるデータ全体に対する対数尤度の差を正（もしくは正が保証されている中で最大にする）にするよう更新
  - 古いパラメータ: λ
  - 新しいパラメータ: λ'

$$L(\lambda,\lambda') = \log\left(\prod_{i=1}^{N} p(y_i \mid x_i;\lambda')\right) - \log\left(\prod_{i=1}^{N} p(y_i \mid x_i;\lambda)\right)$$

パラメータ更新式の導出

$$
\begin{aligned}
L(\lambda,\lambda') &= \log\left(\prod_{i=1}^{N} p(y_i \mid x_i;\lambda')\right) - \log\left(\prod_{i=1}^{N} p(y_i \mid x_i;\lambda)\right) \\
&= \sum_{i=1}^{N} \log p(y_i \mid x_i;\lambda') - \sum_{i=1}^{N} \log p(y_i \mid x_i;\lambda) \\
&= \sum_{i=1}^{N} \log \frac{1}{Z(x_i,\lambda')} \exp\left(\sum_j \lambda'_j f_j(x_i,y_i)\right) - \sum_{i=1}^{N} \log \frac{1}{Z(x_i,\lambda)} \exp\left(\sum_j \lambda_j f_j(x_i,y_i)\right) \\
&= \sum_{i=1}^{N} \log \frac{Z(x_i,\lambda)}{Z(x_i,\lambda')} + \sum_{i=1}^{N}\sum_j (\lambda'_j - \lambda_j) f_j(x_i,y_i) \\
&= \sum_{i=1}^{N}\sum_j (\lambda'_j - \lambda_j) f_j(x_i,y_i) - \sum_{i=1}^{N} \log \frac{Z(x_i,\lambda')}{Z(x_i,\lambda)} \\
&\geq \sum_{i=1}^{N}\sum_j (\lambda'_j - \lambda_j) f_j(x_i,y_i) + \sum_{i=1}^{N}\left(1 - \frac{Z(x_i,\lambda')}{Z(x_i,\lambda)}\right)
\end{aligned}
$$

$-\log x \geq 1-x$ より

パラメータ更新式の導出

$$
\begin{aligned}
L(\lambda,\lambda') &\geq \sum_{i=1}^{N}\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y_i)+N-\sum_{i=1}^{N}\frac{Z(x_i,\lambda')}{Z(x_i,\lambda)} \\
&= \sum_{i=1}^{N}\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y_i)+N-\sum_{i=1}^{N}\frac{1}{Z(x_i,\lambda)}\left(\sum_{y\in Y(x_i)}\exp\left(\sum_{j}\lambda'_j f_j(x_i,y)\right)\right) \\
&= \sum_{i=1}^{N}\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y_i)+N-\sum_{i=1}^{N}\frac{1}{Z(x_i,\lambda)}\left(\sum_{y\in Y(x_i)}\exp\left(\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y)+\sum_{j}\lambda_j f_j(x_i,y)\right)\right) \\
&= \sum_{i=1}^{N}\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y_i)+N-\sum_{i=1}^{N}\sum_{y\in Y(x_i)}\frac{1}{Z(x_i,\lambda)}\exp\left(\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y)\right)\exp\left(\sum_{j}\lambda_j f_j(x_i,y)\right) \\
&= \sum_{i=1}^{N}\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y_i)+N-\sum_{i=1}^{N}\sum_{y\in Y(x_i)}p(y\mid x_i;\lambda)\exp\left(\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y)\right) \\
&= \sum_{i=1}^{N}\sum_{j}\Delta\lambda_j f_j(x_i,y_i)+N-\sum_{i=1}^{N}\sum_{y\in Y(x_i)}p(y\mid x_i;\lambda)\exp\left(\sum_{j}\Delta\lambda_j f_j(x_i,y)\right)
\end{aligned}
$$

ただし、$\lambda'_j-\lambda_j=\Delta\lambda_j$

パラメータ更新式の導出: Generalized Iterative Scaling (GIS)

$$L(\lambda,\lambda') \geq \sum_{i=1}^{N}\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y_i)+N-\sum_{i=1}^{N}\sum_{y\in Y(x_i)}p(y\mid x_i;\lambda)\exp\left(\sum_{j}(\lambda'_j-\lambda_j)f_j(x_i,y)\right)$$

$$=\sum_{i=1}^{N}\sum_{j}\Delta\lambda_j f_j(x_i,y_i)+N-\sum_{i=1}^{N}\sum_{y\in Y(x_i)}p(y\mid x_i;\lambda)\exp\left(C\sum_{j}\Delta\lambda_j\frac{f_j(x_i,y)}{C}\right)$$

ただし、$\lambda'_j-\lambda_j=\Delta\lambda_j$

$$C=\max_{x,y}\sum_{j}f_j(x,y)$$

ジェンセンの不等式

$$L(\lambda,\lambda') \geq \sum_{i=1}^{N}\sum_{j}\Delta\lambda_j f_j(x_i,y_i)+N-\sum_{i=1}^{N}\sum_{y\in Y(x_i)}p(y\mid x_i;\lambda)\frac{f_j(x_i,y)}{C}\exp\left(C\sum_{j}\Delta\lambda_j\right)$$

↑

この最後の式を$A(\lambda,\lambda')$とおこう

パラメータ更新式の導出: Generalized Iterative Scaling (GIS)

$$A(\lambda,\lambda') = \sum_{i=1}^{N}\sum_{j}\Delta\lambda_j f_j(x_i,y_i) + N - \sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda)\frac{f_j(x_i,y)}{C}\exp\left(C\sum_j \Delta\lambda_j\right)$$

ここで、Aを最大化 (=極値を求める)

$$\frac{\partial A(\lambda,\lambda')}{\partial \Delta\lambda_j} = \sum_{i=1}^{N} f_j(x_i,y_i) - \sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda) f_j(x_i,y)\exp(C\Delta\lambda_j) = 0$$

$$\exp(C\Delta\lambda_j) = \frac{\sum_{i=1}^{N} f_j(x_i,y_i)}{\sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda) f_j(x_i,y)}$$

$$\Delta\lambda_j = \frac{1}{C}\log\frac{\sum_{i=1}^{N} f_j(x_i,y_i)}{\sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda) f_j(x_i,y)}$$

GISのパラメータ更新式

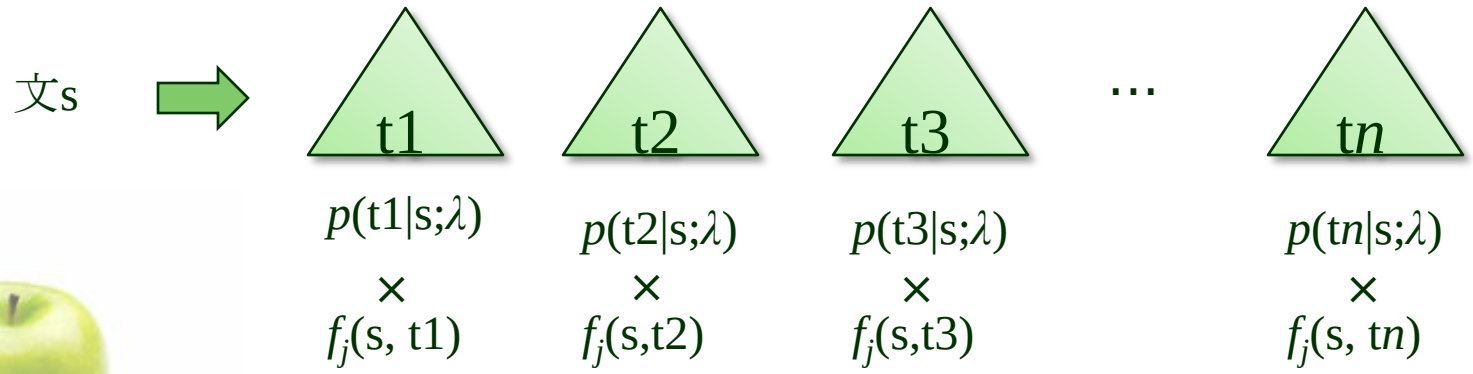
パラメータ更新式の直感的理解

$$\Delta\lambda_j = \frac{1}{C}\log\frac{\sum_{i=1}^{N} f_j(x_i, y_i)}{\sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y \mid x_i;\lambda) f_j(x_i, y)}$$

訓練データに対する素性値の合計

正解候補集合に対する素性値の期待値を合計

パージングなら、、、



GISアルゴリズム

Input: training data $D=\{\langle x,y\rangle\}$, feature functions $f=\{f_j\}$, initial parameters $\lambda=\{\lambda_j\}$
Output: optimal parameters λ

foreach $\langle x,y\rangle \in D$
  foreach $f_j \in f$ such that $f_j(x,y) \neq 0$
    $\mu'_j := f_j(x,y)$

$C := -\infty$
loop until λ converges
  foreach $\langle x,y\rangle \in D$
    $R := \{\}$; $Z := 0$
    foreach $y' \in Y(x)$
      $C := \max(\sum_j f_j(x,y'), C)$; $S := \exp(\sum_k \lambda_k f_k(x,y'))$; $Z := Z + S$
      $R := R \cup \{\langle y', S\rangle\}$
    foreach $\langle y', S\rangle \in R$
      foreach $f_j \in f$ such that $f_j(x,y') \neq 0$
        $\mu_j := \mu_j + f_j(x,y') \cdot 1/Z \cdot S$
  foreach $fj \in f$
    $\Delta\lambda_j := 1/C \cdot \log(\mu'_j/\mu_j)$
    $\lambda_j := \lambda_j + \Delta\lambda_j$

素性森 (Feature Forest)
畳み込み構文森のためのCRF (Packed Parse CRF)

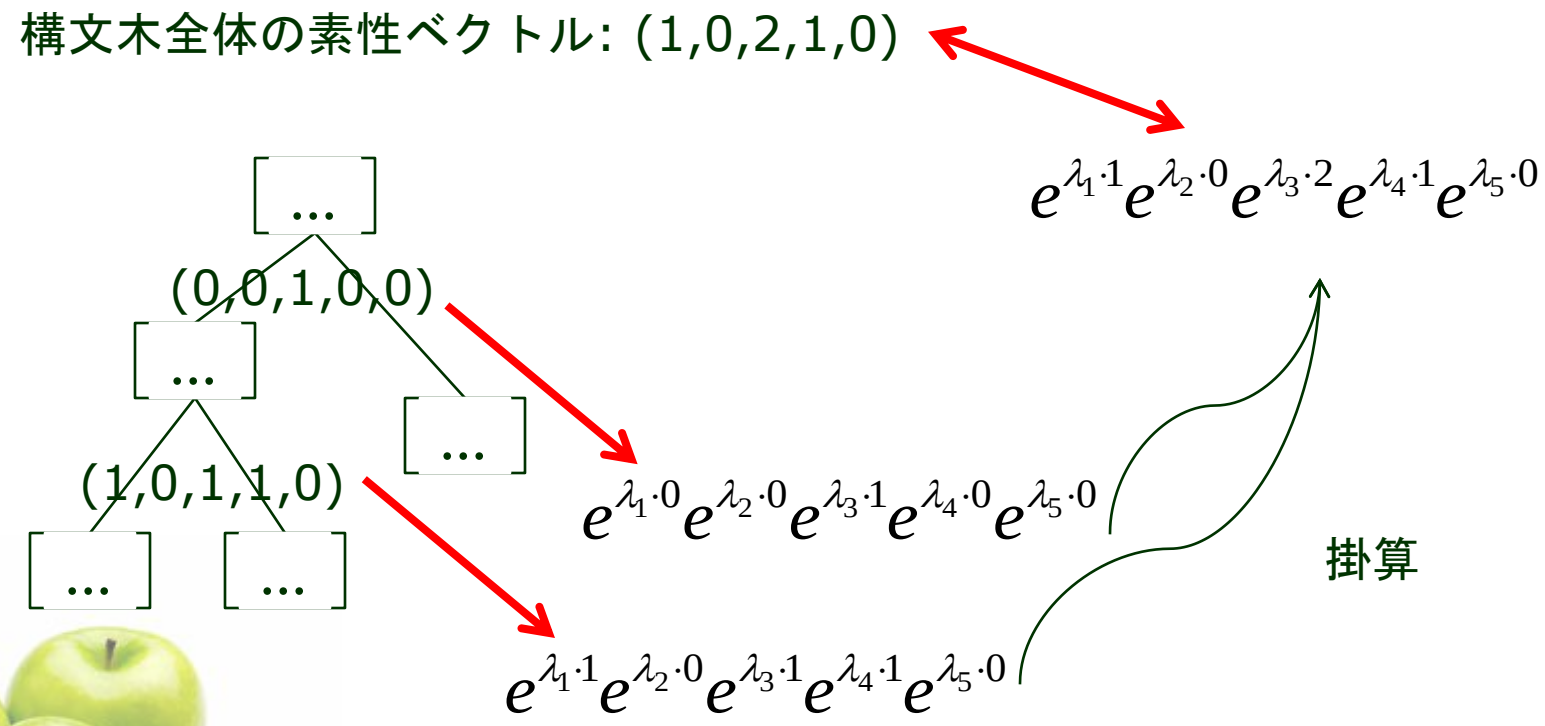
- 素性関数の期待値の計算: 「ある文xに対する全ての構文木集合Y(x)に対する確率」を計算しないといけない

$$\Delta\lambda_j = \frac{1}{C}\log\frac{\sum_{i=1}^{N} f_j(x_i, y_i)}{\sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y \mid x_i;\lambda) f_j(x_i, y)}$$

- 畳み込まれたデータ構造を展開することなく素性関数の期待値を計算
  - 内側外側アルゴリズム (構文木集合)
  - 前向き後向きアルゴリズム (系列ラベリング)

素性森

- 各ブランチのスコアの積＝全体のスコア



素性森

- 構文木の確率

$$p(y \mid x;\lambda) = \frac{1}{Z(x,\lambda)} \exp\left(\sum_j \lambda_j f_j(x,y)\right)$$

$Z(x,\lambda)$：構文木集合全体の確率の和（＝文全体に対する内側確率）

$$\exp\left(\sum_j \lambda_j f_j(x,y)\right) = \prod_j e^{\lambda_j f_j(x,y)} = \prod_c \prod_j e^{\lambda_j f_j(c)}$$

c：構文木の各ブランチ

PCFGの書換規則の確率に対応

- 内側外側アルゴリズムの適用
  - 書換規則の適用回数⇒素性値(素性の発火回数)
  - 書換規則の確率 θ_r ⇒ブランチのスコア $\prod_j e^{\lambda_j f_j(c)}$

EMと最大エントロピー法

	POSタガー	パーザー
データ構造	曖昧性のある畳み込まれた列	曖昧性のある畳み込まれた木構造
EMアルゴリズム	前向き後向きアルゴリズム	内側外側アルゴリズム
最大エントロピー法	MEMM Linear-Chain CRF	Feature Forest (Packed-Parse CRF)

その他のパラメータ推定アルゴリズム

パラメータ更新式の導出: Improved Iteretive Scaling (IIS)

GISでは $C = \max_{x,y} \sum_j f_j(x,y)$ としていたが、

$C(x,y) = \sum_j f_j(x,y)$ とする

$$L(\lambda,\lambda') \geq \sum_{i=1}^{N}\sum_j (\lambda'_j - \lambda_j) f_j(x_i,y_i) + N - \sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda)\exp\left(\sum_j (\lambda'_j - \lambda_j) f_j(x_i,y)\right)$$

$$= \sum_{i=1}^{N}\sum_j \Delta\lambda_j f_j(x_i,y_i) + N - \sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda)\exp\left(C(x,y)\sum_j \Delta\lambda_j \frac{f_j(x_i,y)}{C(x,y)}\right)$$

ただし、 $\lambda'_j - \lambda_j = \Delta\lambda_j$

$C(x,y) = \sum_j f_j(x,y)$

ジェンセンの不等式

$$L(\lambda,\lambda') \geq \sum_{i=1}^{N}\sum_j \Delta\lambda_j f_j(x_i,y_i) + N - \sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda)\frac{f_j(x_i,y)}{C(x,y)}\exp\left(C(x,y)\sum_j \Delta\lambda_j\right)$$

↑この最後の式を$A(\lambda,\lambda')$とおこう

パラメータ更新式の導出: Improved Iteretive Scaling (IIS)

$$A(\lambda,\lambda') = \sum_{i=1}^{N}\sum_{j}\Delta\lambda_j f_j(x_i,y_i) + N - \sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda)\frac{f_j(x_i,y)}{C(x_i,y)}\exp\left(C(x_i,y)\sum_j \Delta\lambda_j\right)$$

ここで、Aを最大化 (=極値を求める)

$$\frac{\partial A(\Delta\lambda)}{\partial\Delta\lambda_j} = \sum_{i=1}^{N} f_j(x_i,y_i) - \sum_{i=1}^{N}\sum_{y\in Y(x_i)} p(y\mid x_i;\lambda) f_j(x_i,y)\exp\left(C(x_i,y)\sum_j \Delta\lambda_j\right) = 0$$

・1変数の方程式になっているので、上の式をニュートン法で解けばよい
・上の式の$C(x_i,y)$が同じ項をまとめると$C(x_i,y)$が同じデータに対してのみモデル期待値を記憶しておくだけですむ
・$C(x_i,y)$を定数Cにしたのが**GIS**で、GISではニュートン法を使わなくても直接解析的に解ける。GISの収束はIISより遅い。
・$C(x_i,y)$のバリエーションが多いと、メモリが大量に必要。

パラメータ推定:勾配ベースのアルゴリズム

- 目的関数の勾配から勾配ベースの推定アルゴリズムでパラメータ推定が可能
  - 最急降下法 (steepest decent method)
  - 共役勾配法 (Conjugate Gradient, CG; Fletcher & Reeves 1964)
  - BFGS (L-BFGS) (Nocedal 1980)
  - 自然言語処理では、経験的に勾配ベースのアルゴリズムの方がIISより非常に速く収束するため、勾配ベースのアルゴリズムが望ましい (Malouf 2002)

パラメータ推定: 勾配ベースのアルゴリズム

- 目的関数

$$L(\lambda) = \log\left(\prod_{i=1}^{N} p(y_i \mid x_i; \lambda)\right) = -\sum_{i=1}^{N} \log Z(x_i, \lambda) + \sum_{i=1}^{N} \sum_{j} \lambda_j f_j(x_i, y_i)$$

- 勾配

$$\begin{aligned}
\frac{\partial L(\lambda)}{\partial \lambda_j} &= \sum_{i=1}^{N} f_j(x_i, y_i) - \sum_{i=1}^{N} \frac{1}{Z(x_i, \lambda)} \frac{\partial Z(x_i, \lambda)}{\partial \lambda_j} \\
&= \sum_{i=1}^{N} f_j(x_i, y_i) - \sum_{i=1}^{N} \frac{1}{Z(x_i, \lambda)} \sum_{y \in Y(x_i)} f_j(x_i, y) \exp\left(\sum_{j} \lambda_j f_j(x_i, y)\right) \\
&= \sum_{i=1}^{N} f_j(x_i, y_i) - \sum_{i=1}^{N} \sum_{y \in Y(x_i)} p(y \mid x_i; \lambda) f_j(x_i, y)
\end{aligned}$$

$$\mathbf{g} = \nabla L(\lambda) = \left\langle \frac{\partial L(\lambda)}{\partial \lambda_1}, \cdots, \frac{\partial L(\lambda)}{\partial \lambda_n} \right\rangle$$

パラメータ推定: 最急降下法

- パラメータ更新式

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \alpha^{(k)}\mathbf{g}^{(k)}$$

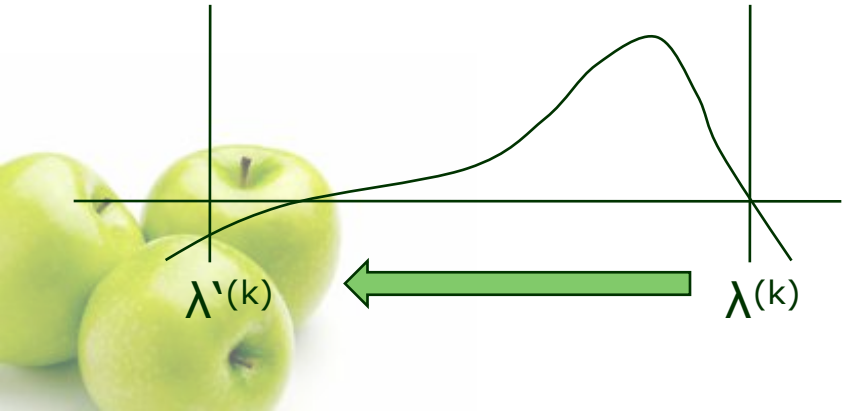
- αは適当な小さな値もしくは一次元最適化(直線探索 ともいう) (one-dimensional line search) で決定
- 収束が非常に遅い

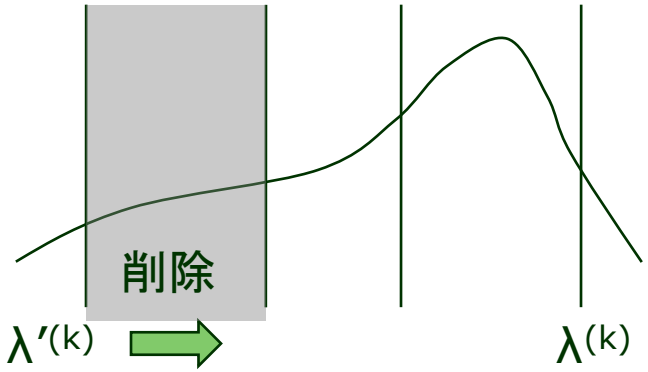
黄金分割にすると、L(λ)の計算が2回ではなくて1回で済む

一次元最適化

1. 候補領域の決定
あるステップ幅を**g**方向に2乗しながら探索し、L(λ')<L(λ)になったところで候補領域の決定

2. 候補領域を3分割(黄金分割)し、2つの中間点のL(λ)を計算し、その大小を比較することにより、左か右の領域を候補領域から削除。2.を繰り返す。





パラメータ推定: 共役勾配法 Conjugate Gradient (CG)

- 更新式

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$$

$$\mathbf{d}^{(k)} = -\mathbf{g}^{(k)} + \beta_{FR}\mathbf{d}^{(k-1)}$$

$$\beta_{FR} = \frac{\mathbf{g}^{(k)}\mathbf{g}^{(k)}}{\mathbf{g}^{(k-1)}\mathbf{g}^{(k-1)}}$$

- αは1次元最適化(one-dimensional line search)で求める
- 毎回、直交する方向に探索している
- n次元なら、n回の繰り返しで終了

パラメータ推定: 準ニュートン法

- 多次元のニュートン法
  - ヘシアンの逆行列の計算が重い…
- 準ニュートン法
  - ヘシアン逆行列を近似する
  - BFGS (Broyden 1970, Fletcher 1970, Goldfarb 1970, Shanno 1970)が有名。ただし、$|\lambda|^2$のサイズの行列を扱うので、巨大な次元の素性ベクトルには向かない
  - Limited-memory BFGS (L-BFGS) (Nocedal 1980)は少ないメモリでヘシアン逆行列を近似する。最大エントロピー法ではよく使われる。

パーセプトロン (Perceptron)

- 最大エントロピー法の問題点
  - Z(正解候補集合のスコアの和)の計算が重い
- パーセプトロン
  - 訓練データx_iに対しy_iを出力する確率が、正解候補集合$Y(x_i)$のどの要素の確率よりも高ければ良い

$$\log p(y_i \mid x_i;\lambda) - \log p(y' \mid x_i;\lambda) \text{を大きく} \left(y' = \arg\max_{y \in Y(x_i)} p(y \mid x_i;\lambda)\right)$$

$$\Leftrightarrow \sum_j \lambda_j f_j(x_i, y_i) - \sum_j \lambda_j f_j(x_i, y') \text{を大きく}$$

- 訓練データの正解と現在のパラメータで推測される最も確率の高い答えとだけ比較
- 実装もアルゴリズムも簡単！
- 最大エントロピーより性能は落ちるけど、メモリー使用量や学習時間の点で非常に有利

パーセプトロン: アルゴリズム

Input: training data D={<x,y>}, feature functions f={f_j}, initial parameters $\lambda=\{\lambda_j\}$

Output: optimal parameters λ

loop until λ converges

  foreach <x,y> $\in$ D

    z' := argmax_z $p(z|x;\lambda)$

    if($y \neq z'$)

      foreach $f_j \in f$

        $\lambda_j := \lambda_j + f_j(x, y) - f_j(x, z')$

おまけ

最大エントロピーモデルの理論的背景

最大エントロピーモデルの理論的背景

- 確率モデルはどこからきたのか？
- エントロピーを最大化？

$$p(y \mid x;\lambda) = \frac{1}{Z(x,\lambda)} \exp\left(\sum_j \lambda_j f_j(x,y)\right)$$

ただし

$$Z(x,\lambda) = \sum_{y' \in Y(x)} \exp\left(\sum_j \lambda_j f_j(x,y')\right)$$

経験確率（経験期待値）とモデル確率（モデル期待値）

- 経験確率
  - データ $\{<x_i, y_i>\}$が与えられた時、

$$\tilde{p}(x, y) = \frac{C(x, y)}{N}$$

$$\tilde{p}(x) = \frac{C(x)}{N}$$

- モデル確率
  - 求める確率分布
  - パラメータを含み、これを推定するのが目標

$$p_M(y \mid x)$$

$$p_M(x, y) = \tilde{p}(x) p_M(y \mid x)$$

経験確率

訓練データの列

x1	x2	x3	y
1	1	0	1
1	0	0	0
0	1	1	1
1	1	0	1
1	1	1	0
0	0	1	1
0	0	1	1
1	1	1	0
1	1	0	0
1	1	1	0
0	1	0	0
0	0	0	1
…	…	…	…

=

データの頻度(経験確率分布)

x1	x2	x3	y	freq(x,y)	p(x,y)
0	0	0	0	983428	983428/N
0	0	0	1	58123	58123/N
0	0	1	0	178237	178237/N
0	0	1	1	1323	1323/N
0	1	0	0	748	748/N
0	1	0	1	23	23/N
0	1	1	0	373	373/N
0	1	1	1	2384	2384/N
1	0	0	0	82	82/N
1	0	0	1	343781	343781/N
1	0	1	0	45854	45854/N
1	0	1	1	83472	83472/N
1	1	0	0	6474	6474/N
1	1	0	1	27	27/N
1	1	1	0	8239	8239/N
1	1	1	1	634	634/N

準備

- X: 入力xの全空間
- Y(x): 入力xに対する出力yの全空間
- F: 素性関数の集合
- エントロピー

$$H(p) = -\sum_{x \in X} p(x) \log p(x)$$

- 条件付きエントロピー

$$H(p) = -\sum_{x \in X} p(x) \sum_{y \in Y(x)} p(y \mid x) \log p(y \mid x)$$

準備

- カルバックライブラー距離(Kullback-Leibler distance)

$$KL(p,q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

  - 条件付き確率の場合

$$KL(p,q) = \sum_{x \in X} p(x) \sum_{y \in Y(x)} p(y \mid x) \log \frac{p(y \mid x)}{q(y \mid x)}$$

  - 二つの確率分布の近さを表す尺度
  - 相対エントロピー(relative entropy)とも呼ばれる
  - 一様分布との距離最小化⇔エントロピー最大化
  - $KL(p,q) \geqq 0$
  - p=qならばKL(p,q)=0

最大エントロピーモデル

- 素性値の制約
  - モデル期待値=経験期待値

$$\forall f_j \in F \quad \sum_{x \in X} \tilde{p}(x) \sum_{y \in Y(x)} p(y \mid x) f_j(x, y) = \sum_{x \in X, y \in Y(x)} \tilde{p}(x, y) f_j(x, y)$$

- 条件付き確率にするための制約

$$\forall x \in \mathbf{x} \quad \sum_{y \in Y(x)} p(y \mid x) = 1$$

- エントロピー最大化

$$H(p) = -\sum_{x \in X} \tilde{p}(x) \sum_{y \in Y(x)} p(y \mid x) \log p(y \mid x)$$

$$p_M(y \mid x) = \arg\max_{p} H(p)$$

解く

- H(p)を等式制約の元で最大化⇒ラグランジュの未定乗数法
  - ラグランジュ関数

$$\Lambda(p,\lambda,\kappa) = H(p) + \sum_j \lambda_j \left\{ \sum_{x\in X} \tilde{p}(x) \sum_{y\in Y(x)} p(y\mid x) f_j(x,y) - \sum_{x\in X, y\in Y(x)} \tilde{p}(x,y) f_j(x,y) \right\} + \sum_{x\in X} \kappa_x \left\{ \sum_{y\in Y(x)} p(y\mid x) - 1 \right\}$$

  - ラグランジュ関数をp(y | x)で偏微分

$$\frac{\partial \Lambda(p,\lambda,\kappa)}{\partial p(y\mid x)} = -\tilde{p}(x)(\log p(y\mid x) + 1) + \tilde{p}(x)\sum_j \lambda_j f_j(x,y) + \kappa_x = 0$$

解く

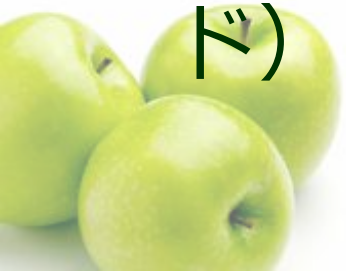
前スライドの等式をp(y | x)について解くと、

$$p(y \mid x) = \exp\left(\sum_j \lambda_j f_j(x,y) - 1 + \frac{\kappa_x}{\tilde{p}(x)}\right) = \exp\left(\sum_j \lambda_j f_j(x,y)\right)\exp\left(-1 + \frac{\kappa_x}{\tilde{p}(x)}\right)$$

次に、ラグランジュ関数をκ_xで偏微分

$$\frac{\partial \Lambda(p,\lambda,\kappa)}{\partial \kappa_x} = \sum_{y \in Y(x)} p(y \mid x) - 1 = 0$$

上の式を代入して解くと、、、（次スライド）



パラメトリックフォーム

$$\sum_{y \in Y(x)} \exp\left(\sum_j \lambda_j f_j(x,y)\right) \exp\left(-1 + \frac{\kappa_x}{\tilde{p}(x)}\right) - 1 = 0$$

$$\Leftrightarrow \exp\left(-1 + \frac{\kappa_x}{\tilde{p}(x)}\right) \sum_{y \in Y(x)} \exp\left(\sum_j \lambda_j f_j(x,y)\right) = 1$$

$$\Leftrightarrow \exp\left(-1 + \frac{\kappa_x}{\tilde{p}(x)}\right) = \frac{1}{\sum_{y \in Y(x)} \exp\left(\sum_j \lambda_j f_j(x,y)\right)}$$

p(y | x)の式に代入すると、

$$p(y \mid x) = \frac{1}{\sum_{y' \in Y(x)} \exp\left(\sum_j \lambda_j f_j(x,y')\right)} \exp\left(\sum_j \lambda_j f_j(x,y)\right)$$

パラメトリックフォーム

解く

- 最後にラグランジュ関数に求まったp(y | x)を代入、極値を求めてλを求める

$$\Lambda(p,\lambda,\kappa) = \sum_{x\in X}\tilde{p}(x)\sum_{y\in Y(x)}p(y\mid x)\log p(y\mid x) + \sum_j \lambda_j\left\{\sum_{x\in X}\tilde{p}(x)\sum_{y\in Y(x)}p(y\mid x)f_j(x,y) - \sum_{x\in X,y\in Y(x)}\tilde{p}(x,y)f_j(x,y)\right\} + \sum_{x\in X}\kappa_x\left\{\sum_{y\in Y(x)}p(y\mid x)-1\right\}$$

この項は0になることに注意

$$\Lambda(\lambda) = \sum_{x\in X}\tilde{p}(x)\sum_{y\in Y(x)}\frac{1}{Z(x)}\exp\left(\sum_j\lambda_j f_j(x,y)\right)\log\frac{1}{Z(x)}\exp\left(\sum_j\lambda_j f_j(x,y)\right) + \sum_j\lambda_j\left\{\sum_{x\in X}\tilde{p}(x)\sum_{y\in Y(x)}\frac{1}{Z(x)}\exp\left(\sum_j\lambda_j f_j(x,y)\right)f_j(x,y) - \sum_{x\in X,y\in Y(x)}\tilde{p}(x,y)f_j(x,y)\right\}$$

解く

$$
\begin{aligned}
\Lambda(\lambda) &= \sum_{x\in X}\tilde{p}(x)\sum_{y\in Y(x)}\frac{1}{Z(x)}\exp\left(\sum_j \lambda_j f_j(x,y)\right)\left(\log Z(x)-\sum_j \lambda_j f_j(x,y)\right)+ \\
&\quad \sum_{x\in X}\tilde{p}(x)\sum_j\left\{\sum_{y\in Y(x)}\frac{1}{Z(x)}\exp\left(\sum_j \lambda_j f_j(x,y)\right)\lambda_j f_j(x,y)\right\}-\sum_{x\in X,y\in Y(x)}\tilde{p}(x,y)\sum_j \lambda_j f_j(x,y) \\
&= \sum_{x\in X}\tilde{p}(x)\sum_{y\in Y(x)}\frac{1}{Z(x)}\exp\left(\sum_j \lambda_j f_j(x,y)\right)\left(\log Z(x)-\sum_j \lambda_j f_j(x,y)\right)+ \\
&\quad \sum_{x\in X}\tilde{p}(x)\sum_{y\in Y(x)}\frac{1}{Z(x)}\exp\left(\sum_j \lambda_j f_j(x,y)\right)\left(\sum_j \lambda_j f_j(x,y)\right)-\sum_{x\in X,y\in Y(x)}\tilde{p}(x,y)\sum_j \lambda_j f_j(x,y) \\
&= \sum_{x\in X}\tilde{p}(x)\sum_{y\in Y(x)}\frac{1}{Z(x)}\exp\left(\sum_j \lambda_j f_j(x,y)\right)\log Z(x)-\sum_{x\in X,y\in Y(x)}\tilde{p}(x,y)\sum_j \lambda_j f_j(x,y) \\
&= \sum_{x\in X}\tilde{p}(x)\log Z(x)\sum_{y\in Y(x)}\frac{1}{Z(x)}\exp\left(\sum_j \lambda_j f_j(x,y)\right)-\sum_{x\in X,y\in Y(x)}\tilde{p}(x,y)\sum_j \lambda_j f_j(x,y) \\
&= \sum_{x\in X}\tilde{p}(x)\log Z(x)-\sum_{x\in X,y\in Y(x)}\tilde{p}(x,y)\sum_j \lambda_j f_j(x,y)
\end{aligned}
$$

解けた

$$\Lambda(\lambda) = \sum_{x \in X} \tilde{p}(x) \log Z(x) - \sum_{x \in X, y \in Y(x)} \tilde{p}(x,y) \sum_j \lambda_j f_j(x,y)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \log Z(x_i) - \frac{1}{N} \sum_{i=1}^{N} \sum_j \lambda_j f_j(x_i, y_i)$$

p.23の数式をみてみると、ラグランジュ関数の極値と最尤推定の極値が一致

→エントロピー最大化により求まるモデルと最尤推定により求まるモデルは一致する

最大エントロピーモデル
ロジスティック回帰
対数線形モデル

- 最大エントロピーモデル(maximum entropy model)
  - 素性に対する制約＋エントロピー最大化によるモデル推定
- (多クラス)ロジスティック回帰 (multi-class logistic regression)
  - 多クラスのロジスティック回帰モデルに対する最尤推定
- 対数線形モデル(log-linear model)
  - log-linearで表現される確率モデルの最尤推定

自然言語処理の分野では上記の三つは同じ確率モデル、パラメータ推定を指す

最大エントロピーモデル ＝ ロジスティック回帰

まとめ

- 確率的HPSG
- 最適化
  - 最大エントロピー法 (GIS, IIS, CG)
  - パーセプトロン