



人工知能特論II 第11回
二宮 崇

今日の講義の予定

- 確率的識別モデル
 - 最大エントロピーモデル(多クラスロジスティック回帰、対数線形モデル)
 - パラメータ推定
 - 自然言語処理での素性ベクトル
- 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
 - Nello Cristianini and John Shawe-Taylor "An Introduction to Support Vector Machines and other kernel-based learning methods", Cambridge University Press, 2000.



最大エントロピーモデル、多クラスロジスティック回帰、対数線形モデル

確率的識別モデル



問題設定

- x : 入力
- y : 出力
- 訓練データ
 - $(x_i, y_i) \quad i=1, \dots, N$
 - 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
 - $f_j(x, y) \quad j=1, \dots, M$
 - 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)
 - $(f_1(x, y), f_2(x, y), \dots, f_M(x, y))$

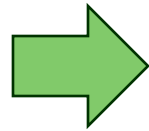


全体の流れ(1/2)

- 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
$\langle f_1(x_1, y_1), f_2(x_1, y_1), \dots, f_M(x_1, y_1) \rangle$
$\langle f_1(x_2, y_2), f_2(x_2, y_2), \dots, f_M(x_2, y_2) \rangle$
...
$\langle f_1(x_N, y_N), f_2(x_N, y_N), \dots, f_M(x_N, y_N) \rangle$



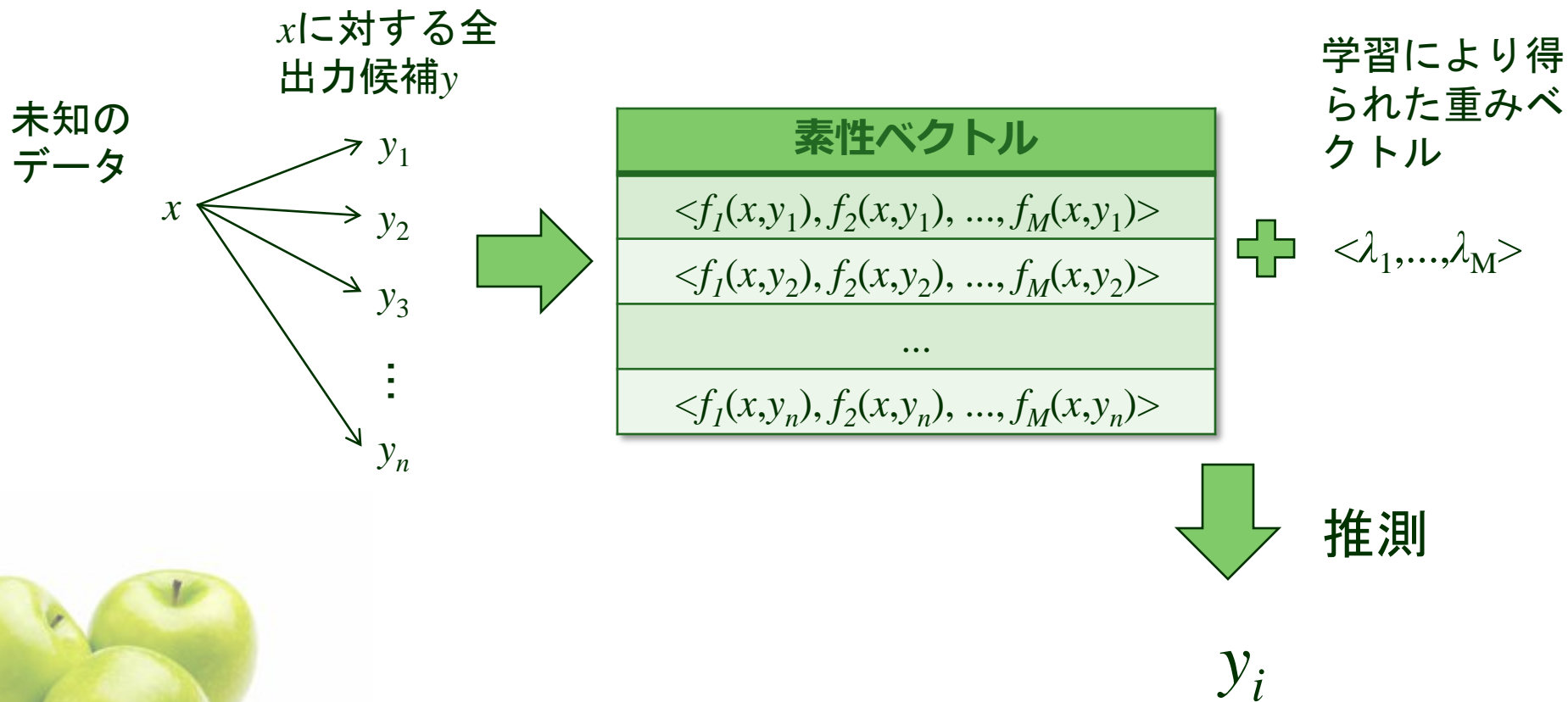
学習

$\langle \lambda_1, \lambda_2, \dots, \lambda_M \rangle$



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

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

ただし

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

重み

素性関数

分配関数
(Partition function)



直感的理解

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

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

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

となる。ちなみに、

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

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

パラメータ推定



パラメータ推定

- 訓練データに対する対数尤度

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

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

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

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



パラメータ推定

- GIS (Generalized Iterative Scaling)
- IIS (Improved Iterative Scaling)
- 勾配に基づく数値最適化
 - 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)
- オンラインアルゴリズム
 - パーセプトロンなど



パラメータ推定: 勾配ベースのアルゴリズム

- 目的関数

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

- 勾配

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

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



パラメータ推定: 最急降下法

- パラメータ更新式

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

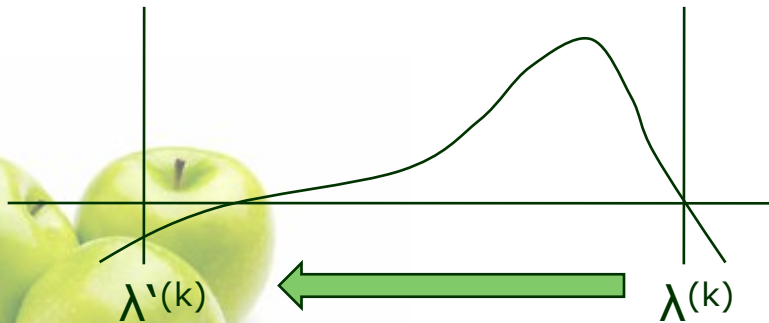
- α は適当な小さな値もしくはは一次元最適化(直線探索 ともい
う) (one-dimensional line search) で決定
- 収束が非常に遅い

黄金分割にすると、 $L(\lambda)$ の計算が2回で
はなくて1回で済む

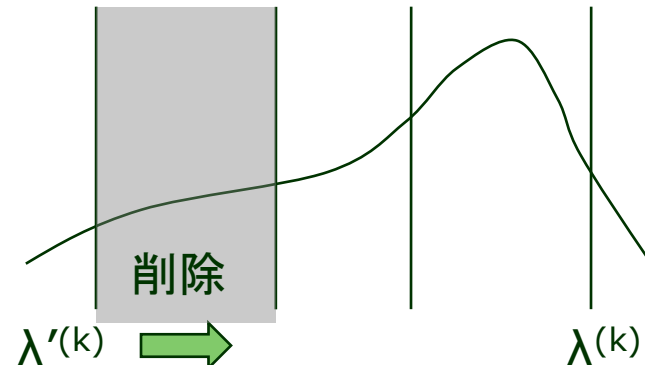
一次元最適化

1. 候補領域の決定

あるステップ幅を \mathbf{g} 方向に2乗しながら探索し、 $L(\lambda') < L(\lambda)$ 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 $L(\lambda)$ を計算し、その大きさ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

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

$\Leftrightarrow \sum_j \lambda_j f_j(x_i, y_i) - \sum_j \lambda_j f_j(x_i, y')$ 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 $D = \langle x_1, y_1 \rangle \dots \langle x_N, y_N \rangle$, feature functions $f = \{f_j\}$, initial parameters $\lambda = \{\lambda_j\}$

Output: optimal parameters λ

loop until λ converges

foreach $\langle x, y \rangle \in D$

$z' := \operatorname{argmax}_z p(z|x;\lambda)$

if($y \neq z'$)

foreach $f_j \in f$

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

自然言語処理の識別モデル



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
- 自然言語処理の場合、疎なベクトルになる
- 疎なベクトルなら数百万次元ぐらい扱える



素性に関する注意その1

- 単語の素性と素性値
 - 例: 今みている単語が“apple”であった時の素性値

appleに対応する次元

各次元が単語に対応する

(0,0,0,0,0,.....,0,1,0,.....,0,0,0,0,0,0)

(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
 - 一つ前の単語が“the”で、今見ている単語が“flies”。一つ前の品詞が動詞で、今見ている品詞が冠詞などなど。
 - SVM: 多項式カーネル
 - 素性の組み合わせを手で指示する。自動的に組み合わせる研究もある。c.f. 素性選択



まとめ

- 識別モデル
 - 最大エントロピーモデル(多クラスロジスティック回帰、対数線形モデル)
- 識別モデルのパラメータ推定
 - 勾配ベースのアルゴリズム
 - パーセプトロン
- 自然言語処理での素性ベクトル



今までの講義資料

<http://aiweb.cs.ehime-u.ac.jp/~ninomiya/ai2/>