1G2-2in

ダイバージェンス最小化原理を利用した線形分類器のロバスト分散学習

Robust Training of Linear Classifiers

小宮山 純平[*1] 大岩 秀和[*1] 中川 裕志[*2]
Komiyama Junpei Oiwa Hidekazu Nakagawa Hiroshi

[*1]東京大学大学院 情報理工学系研究科
Graduation School of Information Science and Technology, the University of Tokyo

[*2]東京大学 情報基盤センター
Information Technology Center, the University of Tokyo

We consider a training of a linear classifier in an environment where data is distributed over many shards. Iterative parameter mixture (IPM) is the state-of-the-art training method for such a task. In the case where all shards are clean, IPM is guaranteed to converge in finite rounds and thus it is able to utilize the all data over the shards. However, the existence of flawed data negatively affects the training result. In this paper, we discuss a method to relieve the effect of such flawed data.

1. はじめに

線形分類器は，機械学習における最も基本的な概念の 1 つである．近年進展の著しいオンライン学習アルゴリズムは，この線形分類器を高速に学習するための方法として，非常に有力であることが知られている [Crammer et al., 2006, Dredze et al., 2008]．オンライン学習におけるアルゴリズムは，通常単一のマシンですべてのデータをストリーム的に処理する状況を仮定している．言い換えると，すべてのデータが単一のマシンからアクセス可能であることが前提とされている．

分散計算環境における機械学習は，データ量の増加に伴いその重要さが増加している．MapReduce をはじめとした近年の分散環境における特徴は，データが複数のシャード（部分データ）に分かれており，それぞれのノード（分散環境におけるマシン）は 1 つのシャードへのアクセスしかできないことである．このような分散環境において全体としての最適化を行うための枠組みとして，Iterative Parameter Mixture（IPM）[Mann et al., 2009] が近年注目されている．IPM では，各イテレーションごとに個々のシャード上で独立した学習を行い，イテレーションの最後で学習結果を混合する．この手法の長所としてその理論的な保障が挙げられる．つまり，線型分離可能な学習データにおける単一マシンでの損失上限を，IPM は分散学習においても引き継ぐことが可能である．

学習データの線形分離可能性は，学習データに関する理論解析のための仮定として多くの場合妥当である．しかし，以下に示すようにデータに汚染がある場合にはその限りではないことが考えられる．たとえば，

- Web サーバは悪意のあるユーザに攻撃を受けることは珍しくないし [Meyer and Whateley, 2004]，
- データフォーマットの整っていないファイルの受け渡しによって学習パラメータが不正な値になることも想定される [Dekel et al., 2010]．
- また，センサーデータはキャリブレーションの失敗により不正な値になることが考えられる．

連絡先: 小宮山 純平，〒 113-0033 東京都文京区本郷 7-3-1，東京大学 総合図書館内 情報基盤センター 学術情報研究部門，junpei_komiyama@mist.i.u-tokyo.ac.jp

これらの問題をまとめてデータ汚染と呼ぶことにする．データ汚染が起こる場合において，理論的保証の枠組みが成り立たず，IPM の学習結果が悪化することが本研究における問題提起である．

本稿では，IPM においてデータ汚染に対してロバストな分散学習手法を提案する．それぞれのシャードにおける学習結果を混合するにあたって，IPM はそれぞれの学習結果の重みを選ぶ自由度がある．データ汚染がない場合，もっともシンプルで妥当な方法は，各ワーカーの結果に均等に重み付けする方法である．また，データ汚染が起こっているシャードが既知の場合，そのシャードに対応するワーカーの結果の重みを 0 にすることによって，汚染データの影響を回避することができるであろう．しかし，我々の想定する問題ではどのシャードが汚染されているかを得ることが難しい．そのため，各結果の統計的な正常性から各シャードの重みを決定するのが妥当であると考えられる．

本研究の貢献は以下のようになる．

- ダイバージェンスの最小化による重みの自動決定原理を提案する（節 3.1）．
- 次に，この原理を用いて，KL ダイバージェンス最小化手法（KL-IPM）を提案する．さらに，データ汚染に対してロバストな手法として，Beta ダイバージェンス最小化手法（Beta-IPM）を提案する（節 3.3）．
- 実験的に Beta-IPM のロバスト性を示すためにこれらの IPM 手法のノイズ化での分類学習における精度を実データにおいて比較する（節 4）．Beta-IPM におけるハイパーパラメータの適切な選択によって，問題のあるシャードの重みを抑え，トレーニング結果を向上させられることを示す．

2. 問題設定

$\mathcal{X} \in \mathbb{R}^d$ を入力空間，$\mathcal{Y} = \{-1, 1\}$ を出力空間とする．それぞれのデータ点は，入出力ペア $(\mathbf{x}_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ で表現される．線形分類器は，パラメータベクトル $\mathbf{w} \in \mathbb{R}^d$ と対応する．パラメータベクトル $\mathbf{w}$ を持つ線形分類器は，入力 $\mathbf{x}_t$ が与えられたときに出力が $\hat{y}_t = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}_t)$ であると予測する．デー

Algorithm 1 Iterative Parameter Mixture（IPM）

1: シャード: $\mathcal{T}_1, ..., \mathcal{T}_M$, $\mathbf{w}^{(\mathrm{avg},0)} = \mathbf{0}$
2: **for** $n = 1, ..., N$ **do**
3: $\quad \mathbf{w}^{(i,n)} = \mathrm{SingleIterationTrain}(\mathcal{T}_i, \mathbf{w}^{(\mathrm{avg},n-1)})$
4: $\quad \mathbf{w}^{(\mathrm{avg},n)} = \sum_i \mu_{i,n} \mathbf{w}^{(i,n)}$
5: **end for**

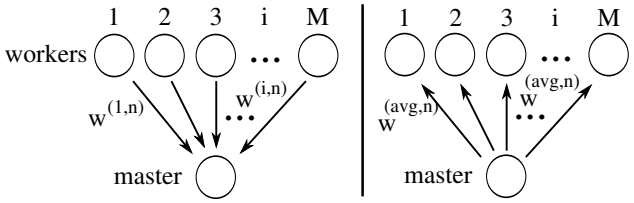

図 1: IPM におけるマスター・ワーカー間の通信

タセットにおける入力と出力の間の関係をうまく表現した線形分類器を求めるのが本研究における学習の目的である．

本研究で考える分散環境では，トレーニングデータは M 個の部分データセット（シャード）に分かれている．分散学習は1つのマスターノードと，各シャードにつき1つのワーカーノードによって行われる．それぞれのワーカーは，自分のシャードにのみアクセスすることができる．また，マスターは各ワーカーと学習器（パラメータベクトル）のやりとりをすることができる．IPM（Algorithm 1）は以下のようなアルゴリズムである．それぞれのエポック $n = 1, 2, ...$ において，各ワーカーはそれぞれのシャードにおいて1イテレーションの学習（“SingleIterationTrain”，Algorithm 1）を行い，学習終了時のパラメータベクトル $\mathbf{w}^{(i,n)}$ をマスターノードに送る．マスターはすべてのワーカーがトレーニングを終了するまで待ち，各パラメータベクトルを受け取る（図 1 左）．受け取ったパラメータベクトルを重み $\mu_{i,n}$ をつけて加算し，結果として混合ベクトル $\mathbf{w}^{(\mathrm{avg},n)}$ を生成する．このとき，混合の重みは総和が 1 になるように任意のものを選ぶことができる．マスターは混合ベクトル $\mathbf{w}^{(\mathrm{avg},n)}$ を各ワーカーに送り返す（図 1 右）．各ワーカーは混合ベクトルを受け取り，それを自らのパラメータベクトルとして次のイテレーションを開始する．この過程を N エポック繰り返し，マスターは最終的なパラメータベクトルを出力する．

2.1 オンライン学習アルゴリズムによるワーカーの学習

各ワーカーの1イテレーションの学習ルーチンとして，今回はオンライン学習を考える．具体的には，perceptron と Passive Aggressive（PA）法 [Crammer et al., 2006] を扱う．

3. ダイバージェンス最小化原理

本節では，本研究の主提案である IPM におけるダイバージェンス最小化原理における重みの決定手法（KL-IPM と Beta-IPM）を解説する．一言で言えば，提案手法はノイズのあるガウス分布に従う観測データからのロバストな推定手法を重み決定に利用している．KL-IPM と Beta-IPM の式導出は紙面の都合上省略する．

3.1 ダイバージェンス最小化原理とその統計的仮定

本研究では以下のような統計的仮定をおく．エポック n にワーカーの返すそれぞれのパラメータベクトルがあるガウス分布 Q_n から独立に引かれたものであると仮定する．我々の提案は，このパラメータベクトルを混合した $\mathbf{w}^{(\mathrm{avg},n)}$ はガウス分布 Q_n の中心であるべきと考えることである．しかし，実際にはパラメータベクトルがノイズを含む可能性のある分布 P_n から引かれている可能性がある．より正確に書くと，エポック n の時点でのワーカーの返すパラメータベクトルの集合 $\mathbf{w}^{(\cdot,n)}$ が

$$\mathbf{w}^{(\cdot,n)} \sim P_n = (1-\epsilon)Q_n + \epsilon R_n. \tag{1}$$

という形で与えられると考える．ここで，R_n はノイズ分布，ϵ はノイズの混入割合である．Q_n の平均 $\boldsymbol{\mu}$ と分散 $\boldsymbol{\Sigma}$ を，以下のダイバージェンスを最小化するように決定する:

$$\arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} D(P_n || Q_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})). \tag{2}$$

ここで，D は P_n と Q_n の間のダイバージェンスである．我々の目的は，ノイズに対して耐性のあるダイバージェンスを選択することによって，高ノイズ下でもノイズ分布 R_n の影響を減らし，P_n から Q_n を正しく推定することである．

3.2 KL ダイバージェンスと Beta ダイバージェンス

KL ダイバージェンスは，ある確率分布と別の確率分布の間の乖離度を測る指標として最も基本的な指標である．$\mathbb{R}^d$ 上の2つの確率分布 P と Q の間の KL ダイバージェンスは以下のように定義される．

$$D_{KL}(P||Q) = \int P(\mathbf{w}) \log \frac{P(\mathbf{w})}{Q(\mathbf{w})} d\mathbf{w}, \tag{3}$$

KL ダイバージェンスは非負であり，2つの確率分布がほとんど至るところで $P = Q$ である場合にのみ 0 の値をとる．KL ダイバージェンスは情報理論において最も基本的な指標であるが，はずれ値やデータ汚染に対してのロバスト性は持っていない．

次に，[Basu et al., 1998] と [Eguchi and Kano, 2001] によって導入された Beta ダイバージェンスについて説明する．Beta ダイバージェンスは KL ダイバージェンスの拡張であり，パラメータ $\beta > 0$ を持つ．確率分布 P と Q の間の Beta ダイバージェンスは

$$D_\beta(P||Q) = \int \left\{ P(\mathbf{w}) \frac{P^\beta(\mathbf{w}) - Q^\beta(\mathbf{w})}{\beta} \right\} - \frac{P^{\beta+1}(\mathbf{w}) - Q^{\beta+1}(\mathbf{w})}{\beta+1} d\mathbf{w}. \tag{4}$$

と定義される．また，β が 0 に近づく極限において，$\lim_{\beta\to 0} D_\beta(P||Q) = D_{KL}(P||Q)$ と一貫して定義することができる．つまり，Beta ダイバージェンスは KL ダイバージェンスの自然な拡張であると考えることができる．Beta ダイバージェンスはノイズ分布に対してロバスト性を持つことが知られており，いくつかの分野での応用例がある．たとえば，信号処理 [Minami and Eguchi, 2002]，独立成分分析 [Mollah et al., 2007]，そして非負行列分解 [Simsekli et al., 2013] などである．β の大きさは任意の正の値に設定できるが，大きければ大きいほどロバスト性が増し，計算効率が下がるというトレードオフがある．

3.3 KL-IPM と Beta-IPM

KL および Beta ダイバージェンスを最小化させるような重みの自動決定手法を提案する．まず最初に，KL ダイバージェンスの最小化方法は以下のようになる．

$$\mathbf{w}^{(\mathrm{avg},n)} = \arg\min_{\boldsymbol{\mu}} D_{KL}(P||Q(\boldsymbol{\mu}, \boldsymbol{\Sigma})), \tag{5}$$

ここで，$Q = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ はガウス分布であり，P は $\{\mathbf{w}^{(1,n)}, ..., \mathbf{w}^{(M,n)}\}$ などが引かれる仮想的な確率分布である．

提案 1.（KL-IPM）*KL-IPM* を，各エポック n に以下のようにパラメータ混合を行う方式と定義する．

$$\mathbf{w}^{(\mathrm{avg},n)} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{w}^{(i,n)} \tag{6}$$

言い換えると，もっとも単純にすべてのワーカーに等しい重みをつける方法は KL ダイバージェンスの最小化とみなすことができる．しかし，この方法はデータ汚染を含むシャードが存在する場合にその影響を受けてしまう可能性がある．この問題を解決するのが，次に述べる Beta-IPM である．Beta-IPM は次の最小化問題の解である:

$$\mathbf{w}^{(\mathrm{avg},n)} = \arg\min_{\boldsymbol{\mu}} D_{\beta}(P||Q(\boldsymbol{\mu}, \boldsymbol{\Sigma})). \tag{7}$$

提案 2.（Beta-IPM）$\boldsymbol{\mu}_c$ と $\boldsymbol{\Sigma}_c$ をそれぞれパラメータベクトル集合 $\{\mathbf{w}^{(i,n)}\}$ の経験平均と経験共分散とする．つまり，

$$\boldsymbol{\mu}_c = \frac{1}{M}\sum_{i=1}^{M}\mathbf{w}^{(i,n)}, \tag{8}$$

と

$$\boldsymbol{\Sigma}_c = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{w}^{(i,n)} - \boldsymbol{\mu}_c)(\mathbf{w}^{(i,n)} - \boldsymbol{\mu}_c)^{\top}. \tag{9}$$

である．*Beta-IPM* は，各エポック n において以下のように混合パラメータ $\mathbf{w}^{(\mathrm{avg},n)}$ を決定する*:*

$$\mathbf{w}^{(\mathrm{avg},n)} = \frac{\sum_{i=1}^{M}\exp S(\mathbf{w}^{(i,n)}|\boldsymbol{\mu}_c, \frac{1}{\beta}\boldsymbol{\Sigma}_c)\mathbf{w}^{(i,n)}}{\sum_{j=1}^{M}\exp S(\mathbf{w}^{(j,n)}|\boldsymbol{\mu}_c, \frac{1}{\beta}\boldsymbol{\Sigma}_c)}, \tag{10}$$

ここで，$S(\mathbf{w}^{(i,n)}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -(1/2)(\mathbf{w}^{(i,n)} - \boldsymbol{\mu})^{\top}\Sigma^{-1}(\mathbf{w}^{(i,n)} - \boldsymbol{\mu})$ は多変量 *Gauss* 関数の指数部である．

つまり，Beta-IPM においてはそれぞれのパラメータベクトルをその平均からの距離に応じて重みづける方式である．Beta-IPM は，実数パラメータ $\beta \geq 0$ の自由度があり，$\beta \to 0$ の極限で KL-IPM と一致する．KL-IPM and Beta-IPM の導出は紙面の制約から省略する．ただし，Beta-IPM の最小化問題は非凸であり，我々はテイラー展開のような一次近似を行い上記の式を導出した．

4. 実験評価

我々は，各種データセットでの評価を行った．これらの目的は，KL-IPM と Beta-IPM がデータ汚染環境でどのように振る舞うかを調べることである．

4.1 実験設定

実験には１６の２値分類データセットを利用した（表 1）．データセットの次元（素性数）およびデータ数は大きく幅がある．結果がデータセットの次元の順番に並んでいることに注意されたい．zeta と ocr データセットは Pascal large-scale learning challenge webpage[*1]，imdb と citeseer データセットは Paul Komarek's webpage[*2]から取得した．その他のデータセットは全て LIBSVM dataset repository[*3]から取得した．

表 1: 各データセットの素性数とデータ数

	素性数（次元）	データ数
ijcnn1.tr	22	49,990
mushrooms	112	8,124
a8a	123	22,696
ocr	1,156	3,500,000
epsilon	2,000	400,000
zeta	2,000	500,000
gisette	5,000	6,000
real-sim	20,958	72,309
rcv1	47,236	20,242
citeseer	105,354	181,395
imdb	685,569	167,773
news20	1,355,191	19,996
url	3,231,961	2,396,130
webspam	16,609,143	350,000
kdda	20,216,830	8,407,752
kddb	29,890,095	19,264,097

データシャード: それぞれのデータセットにおいて，80%のデータとトレーニングデータ，残りの 20%をテストデータとした．トレーニングデータは 100 個のシャードに均等に分割され，それぞれのワーカーに割り当てられた．全てのアルゴリズムの評価は，トレーニング後のテストデータ分類精度によって行った．

アルゴリズムのデータ汚染に対するロバスト性を調べるために，我々はクリーン設定（データ汚染無し）および 2 つのデータ汚染の設定を検証した．つまり

設定 1 - 敵対的なラベル付け: この設定では，100 個のシャードのうち 30 個のシャードが敵対的なデータ（ラベルを反転したデータ）を返す：

設定 2 - ランダムなラベル付け: この設定では，100 個のシャードのうち 80 個のシャードがランダムなデータを返す：それぞれのデータは確率 p で 1，それ以外の確率で 0 を返す．p の値はシャードによって個別で，0.1 と 0.9 の間に設定した．

アルゴリズム: 我々は，KL-IPM を perceptron および PA と組み合わせたもの（KL-IPM-perceptron と KL-IPM-PA），そして Beta-IPM を perceptron および PA（Beta-IPM-perceptron と Beta-IPM-PA）と組み合わせたものの４つのアルゴリズムを評価した． それぞれの β の値は $\{10^{-1}, 10^{-2}, ..., 10^{-8}\}$ の中から最も結果の良いものを選択した．我々のデータセットは高次元のデータセットを含むため，Beta-IPM のガウシアンの非対角項は削除した．また，分散が 0 の feature は削除した．それぞれのパラメータベクトルは，Beta-IPM での重み決定時のみ正規化した．

4.2 実験結果

各データセットでの実験結果を表 2 に示す．

- ほぼ全てのデータセットで，KL-IPM の分類精度はクリーン設定からデータ汚染の影響によって低下が見られる．とくに，設定２：ランダムラベルの場合には学習器の８割が正しくないデータを返すため，汚染の影響が大きい．

∗1 http://largescale.ml.tu-berlin.de/

∗2 http://komarix.org/ac/ds/

∗3 http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

表 2: 各設定における 50 エポック終了時での分類精度．太字は各データセットでの最大値．

クリーン設定（データ汚染なし）								
	ijcnn1.tr	mushrooms	a8a	ocr	epsilon	zeta	gisette	real-sim
KL-IPM-perceptron	0.913	0.999	0.845	0.763	0.899	0.628	0.947	0.968
KL-IPM-PA	0.912	0.999	0.845	0.762	0.899	0.694	0.958	0.975
	rcv1	citeseer	imdb	news20	url	webspam	kdda	kddb
KL-IPM-perceptron	0.960	0.976	0.981	0.953	0.986	0.990	0.881	0.886
KL-IPM-PA	0.966	0.977	0.985	0.958	0.986	0.990	0.882	0.887

設定 1 ：敵対的ラベル付け								
	ijcnn1.tr	mushrooms	a8a	ocr	epsilon	zeta	gisette	real-sim
KL-IPM-perceptron	**0.908**	0.937	0.838	0.760	0.881	0.582	0.854	0.824
KL-IPM-PA	**0.908**	0.983	0.837	0.760	0.886	0.651	0.912	0.904
Beta-IPM-perceptron	**0.908**	**0.998**	**0.846**	**0.763**	**0.898**	**0.665**	0.935	0.961
Beta-IPM-PA	**0.908**	0.989	**0.846**	0.762	**0.898**	0.663	**0.957**	**0.972**
	rcv1	citeseer	imdb	news20	url	webspam	kdda	kddb
KL-IPM-perceptron	0.762	0.976	0.980	0.703	0.983	0.987	0.743	0.759
KL-IPM-PA	0.871	**0.977**	**0.984**	0.844	0.983	0.987	0.689	0.712
Beta-IPM-perceptron	0.955	0.976	0.981	0.945	**0.986**	**0.991**	**0.876**	**0.882**
Beta-IPM-PA	**0.962**	**0.977**	**0.984**	**0.950**	**0.986**	**0.991**	0.676	0.693

設定 2 ：ランダムなラベル付け								
	ijcnn1.tr	mushrooms	a8a	ocr	epsilon	zeta	gisette	real-sim
KL-IPM-perceptron	0.886	0.858	0.820	0.750	0.737	0.516	0.698	0.680
KL-IPM-PA	0.855	0.942	0.817	0.674	0.758	0.545	0.827	0.741
Beta-IPM-perceptron	0.911	0.980	0.825	**0.755**	0.886	**0.642**	0.888	0.948
Beta-IPM-PA	**0.913**	**0.999**	**0.830**	0.723	**0.890**	0.624	**0.942**	**0.958**
	rcv1	citeseer	imdb	news20	url	webspam	kdda	kddb
KL-IPM-perceptron	0.600	0.657	0.611	0.644	0.971	0.951	0.739	0.734
KL-IPM-PA	0.701	0.685	0.684	**0.730**	0.971	0.960	0.761	0.757
Beta-IPM-perceptron	**0.919**	0.836	0.826	0.717	0.981	**0.986**	0.853	0.833
Beta-IPM-PA	0.910	**0.916**	**0.943**	**0.730**	**0.985**	0.985	**0.868**	**0.868**

- Beta-IPM はほぼすべてのデータセットで KL-IPM 以上の分類精度を示し，rcv1 や kdda などいくつかのデータセットではその差は非常に大きい．多くのデータセットにおいて，Beta-IPM はデータ汚染のない場合の分類精度をかなり良く復元している．これは，Beta-IPM がデータ汚染の影響を軽減させられることの経験的な証拠であると考えられる．

参考文献

[Basu et al., 1998] Basu, A., Harris, I. R., Hjort, N. L., and Jones, M. C. (1998). Robust and efficient estimation by minimising a density power divergence. *Biometrika*, 85(3):549–559.

[Crammer et al., 2006] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

[Dekel et al., 2010] Dekel, O., Shamir, O., and Xiao, L. (2010). Learning to classify with missing and corrupted features. *Mach. Learn.*, 81(2):149–178.

[Dredze et al., 2008] Dredze, M., Crammer, K., and Pereira, F. (2008). Confidence-weighted linear classification. In *ICML*, pages 264–271.

[Eguchi and Kano, 2001] Eguchi, S. and Kano, Y. (2001). Robustifying maximum likelihood estimation. In *Technical report, Institute of Statistical Mathematics, June 2001*.

[Mann et al., 2009] Mann, G., McDonald, R. T., Mohri, M., Silberman, N., and Walker, D. (2009). Efficient large-scale distributed training of conditional maximum entropy models. In *NIPS*, pages 1231–1239.

[Meyer and Whateley, 2004] Meyer, T. A. and Whateley, B. (2004). Spambayes: Effective open-source, bayesian based, email classification system. In *CEAS*.

[Minami and Eguchi, 2002] Minami, M. and Eguchi, S. (2002). Robust blind source separation by beta divergence. *Neural Computation*, 14:1859–1886.

[Mollah et al., 2007] Mollah, M. N., Eguchi, S., and Minami, M. (2007). Robust prewhitening for ica by minimizing beta-divergence and its application to fastica. *Neural Processing Letters*, 25(2):91–110.

[Simsekli et al., 2013] Simsekli, U., Taylan Cemgil, A., and Kenan Yilmaz, Y. (2013). Learning the beta-divergence in tweedie compound poisson matrix factorization models. In *ICML*, volume 28, pages 1409–1417.