SACSIS 2013 チュートリアル
2013/5/23 @ 仙台国際センター

機械学習の理論と実践

岡野原　大輔

株式会社Preferred Infrastructure
hillbig@preferred.jp

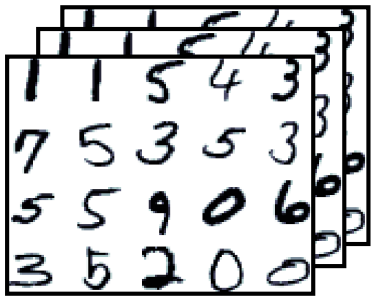
アジェンダ

- 導入編
  - 機械学習はどこでどのように何故使われているのか
  - 代表的な手法、理論、応用
- 実践編
  - 実際に機械学習を自分で作ってみる
  - 機械学習で求められる計算処理
- 発展編
  - Jubatus：リアルタイム・分散下での機械学習
  - 深層学習（DNN）：表現学習/古くて新しい機械学習

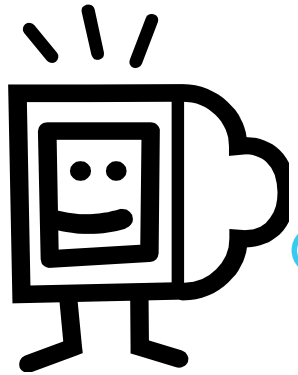
導入編

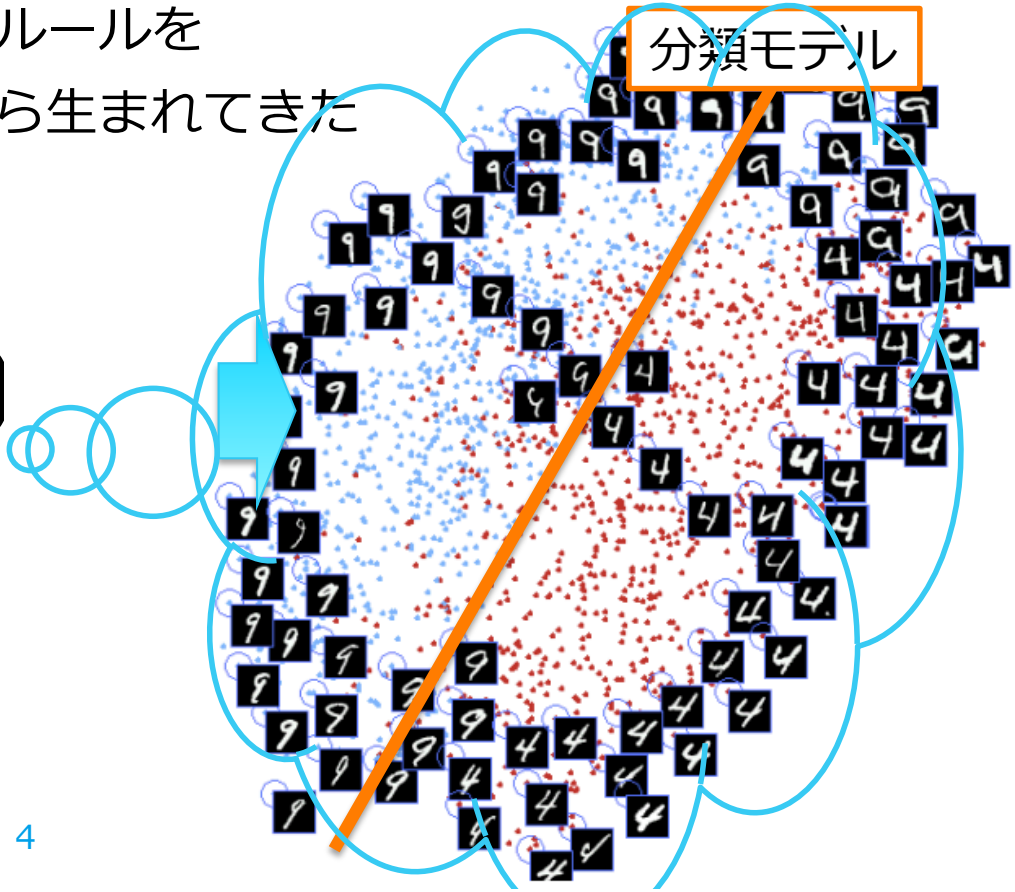
機械学習とは

- 経験（データ）によって賢くなるアルゴリズムの研究
  - データから知識・ルールを自動獲得する
  - データの適切な表現方法も獲得する
  - 人工知能の中で、人が知識やルールを
    明示的に与える方法の限界から生まれてきた



学習データ





BigData と 機械学習

- 巨大なデータがあらゆる分野で生まれている
  - Volume（量）, Velocity（生成速度）, Variety（多様性）
- データがどこでもある ⇒　機械学習がどこでも使える！

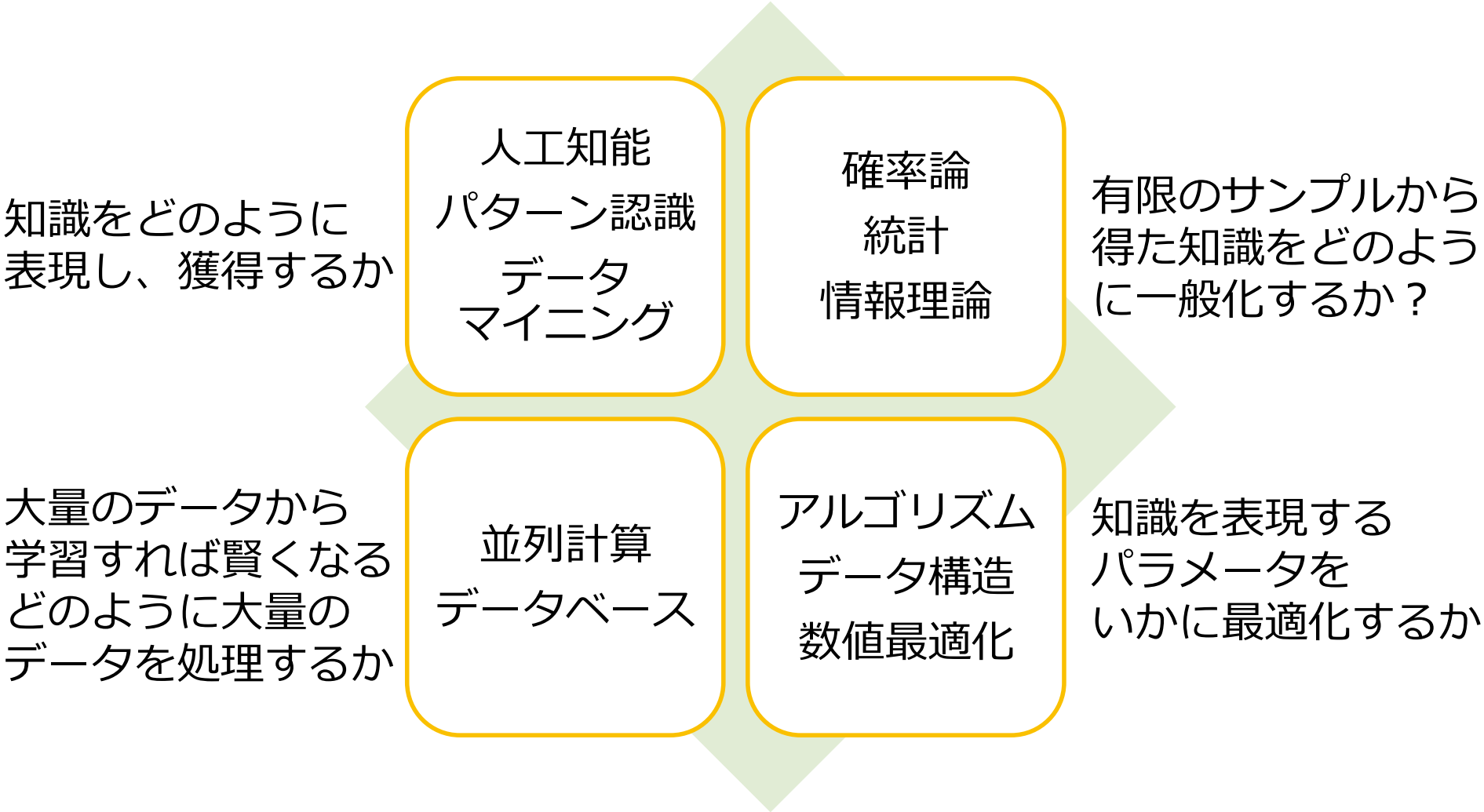
Preferred Infrastructure

機械学習の利用分野

はじめは金融、ウェブが中心だったがそこから他分野で急速に広がる

レコメンデーション クラスタリング	分類、識別	市場予測	評判分析
情報抽出	文字認識	ロボット	画像解析
遺伝子分析	検索ランキング	金融	医療診断

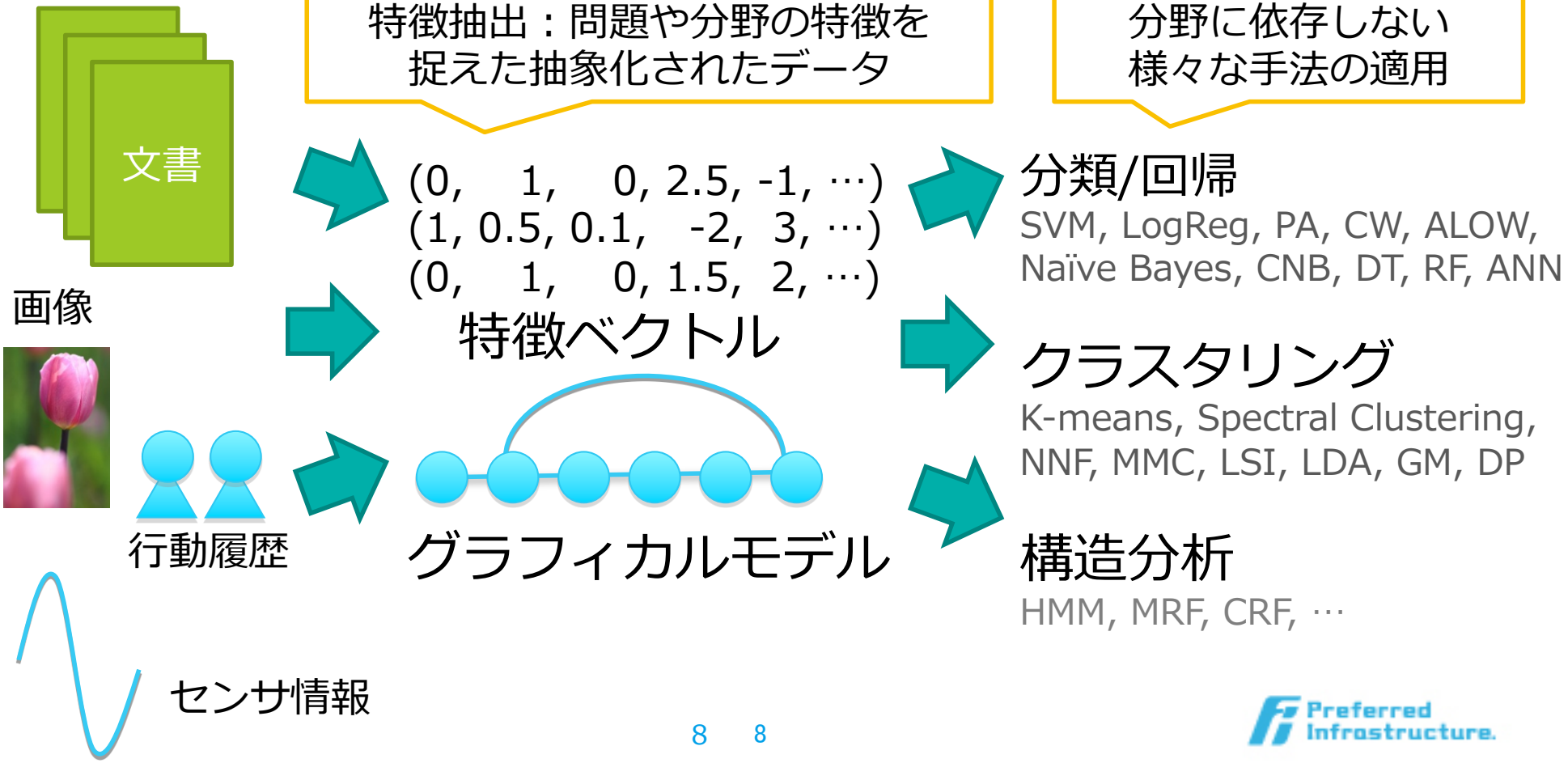
機械学習は総合格闘技



Preferred Infrastructure

機械学習が急速に広まった理由
特徴表現と分析処理の分離

- 異なる分野で同じ機械学習手法が適用可能

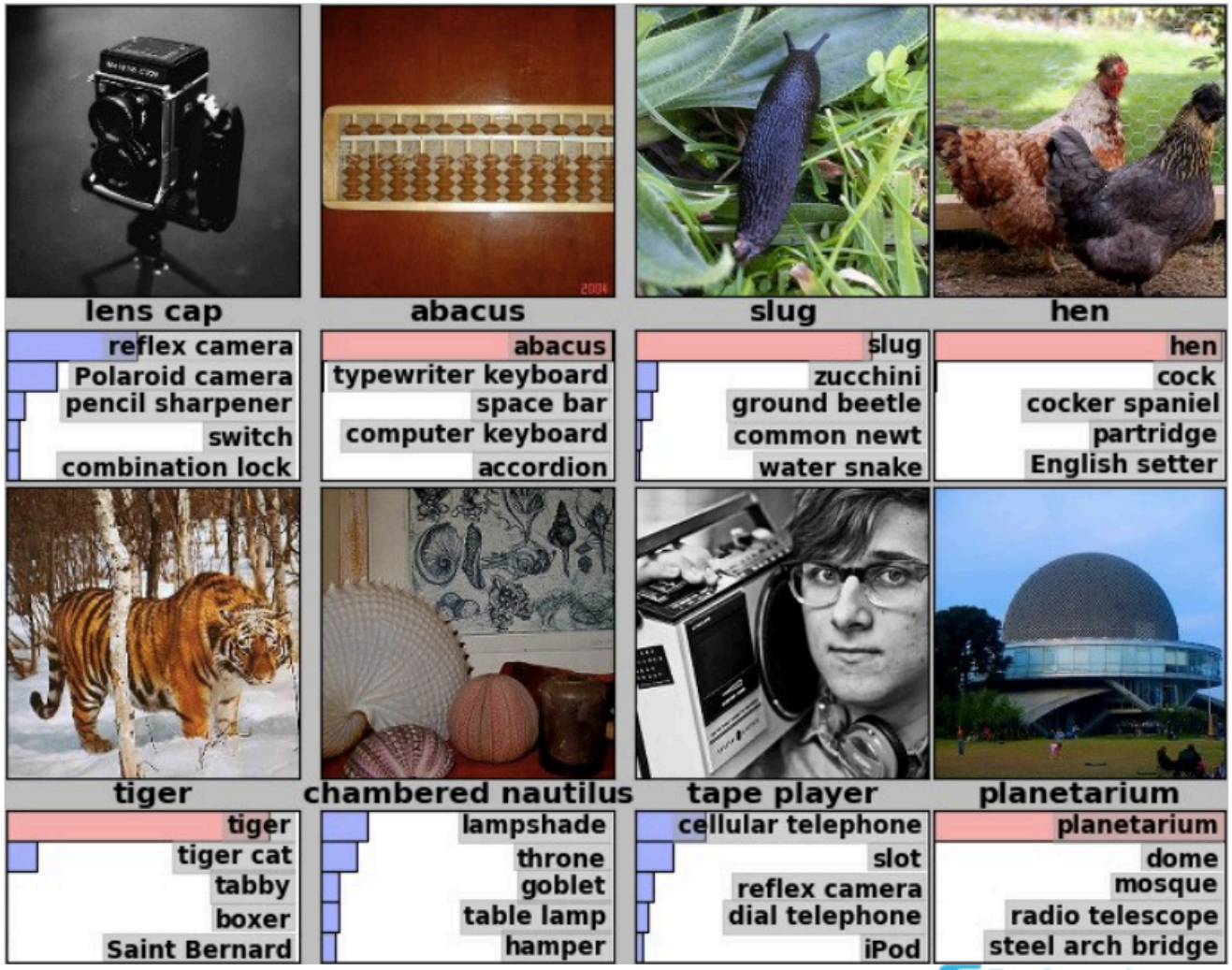


Preferred Infrastructure

最近の機械学習の進展の例（１）
一般物体認識（LSVRC2012優勝チーム）

一般物体のカテゴリ認識

間違えた場合も、納得のできる間違え方
（右下のtape playerを携帯電話と間違えている）

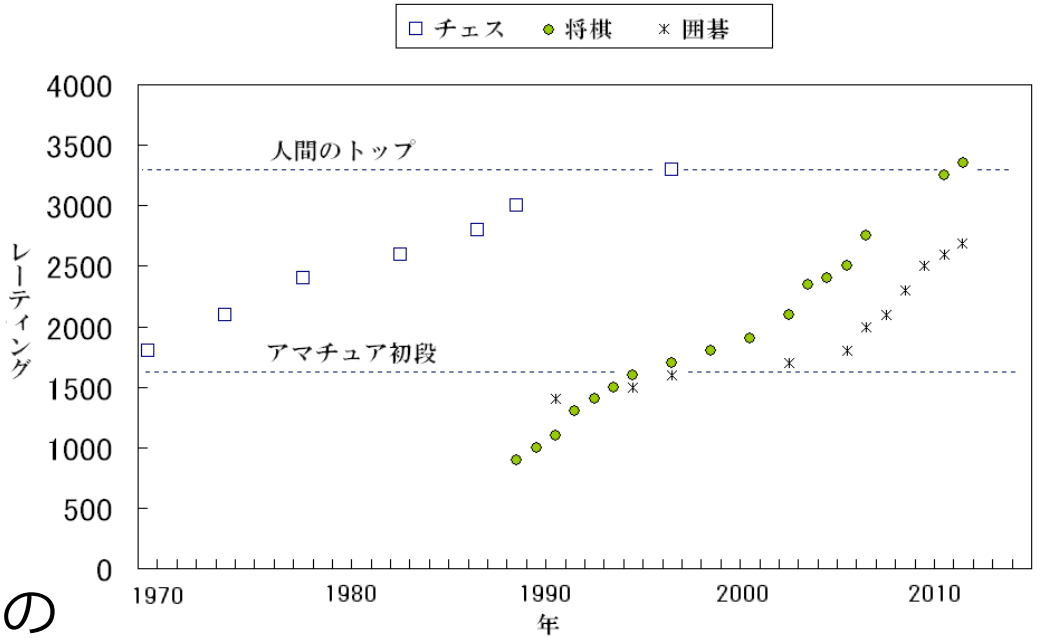


http://www.image-net.org/challenges/LSVRC/2012/ilsvrc2012.pdf

最近の機械学習の進展の例（２）
コンピュータ将棋・囲碁・チェス

分野の常識を変えた学習アルゴリズムの登場により
各分野で人のトップを超えてしまった

- チェス・将棋
  - Comparison training
    評価関数の自動学習
- 囲碁
  - Monte-Carlo Tree Search
    評価関数が不要
- ポーカー
  - Counterfactual Regret Minimization (CFR)
    大規模不完全情報ゲームの
    ナッシュ均衡解の高速計算

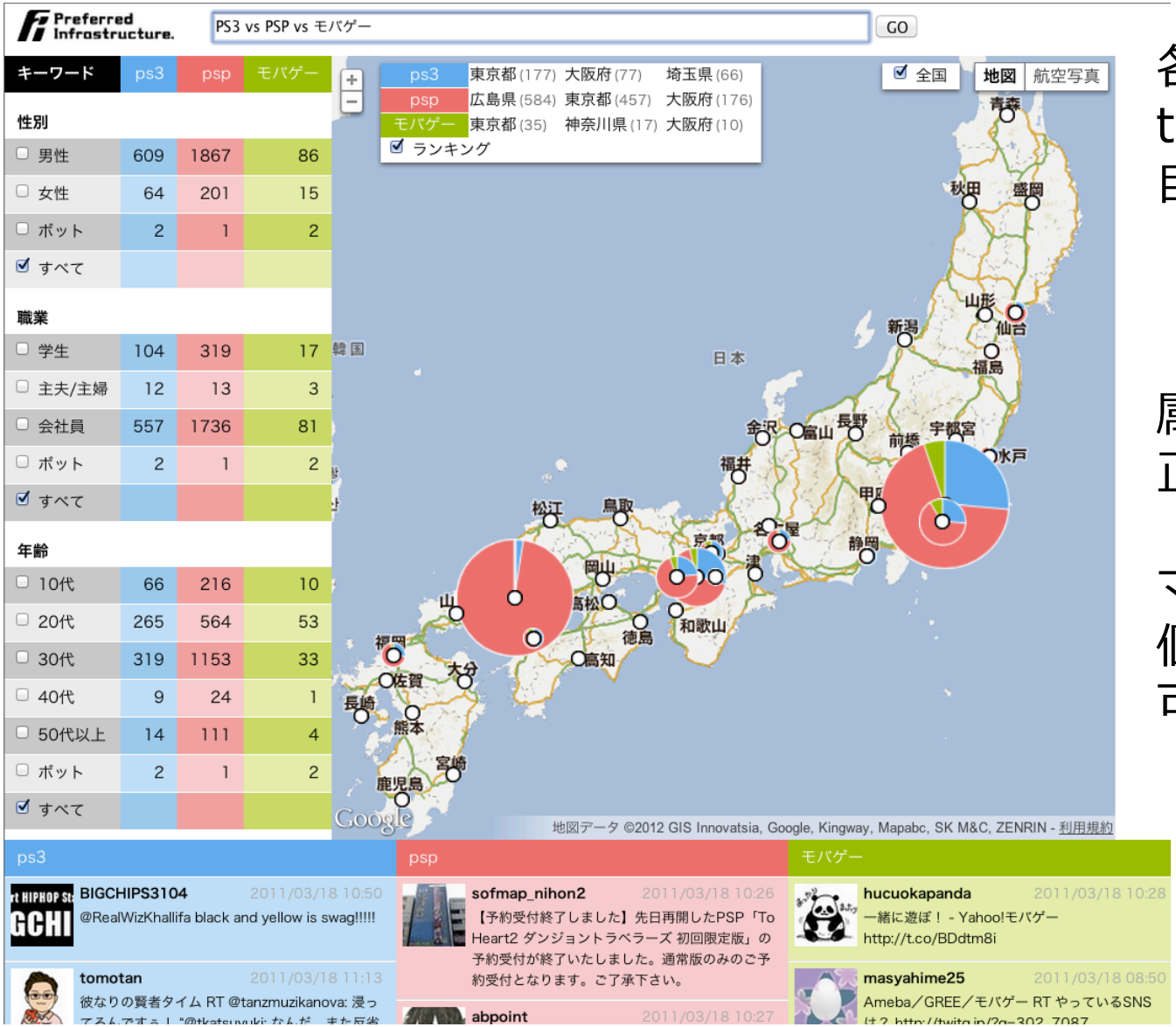


http://blog.livedoor.jp/yss_fpga/archives/53897129.html

詳細は鶴岡慶雅先生のチュートリアル「自然言語処理とAI」を参照

Preferred Infrastructure

最近の機械学習の進展の例（3）
SNS分析



各ユーザーの属性を
tweetの内容から
自動で推定
　性別・職業
　年齢・趣味

属性推定ルールは
正解データを元に学習

マスではなく
個人単位での分析が
可能になった

Preferred Infrastructure.

機械学習の長所 (1/4)
データがあればすぐ試せる

- 分類ルールを学習したい場合、正解事例がいくつかあれば学習可能

- 質問：データがどれくらいあったらうまくいくのか？
- 答え：正解事例が1つしかなくても動く、多ければ精度は高くなる
  - 以下のファクターで正解事例はより多く必要とする
    - 問題の複雑さ
    - 正解事例に含まれるノイズ量
    - 学習モデルの複雑さ
  - 実際は数十から数百、人手で作れる規模が殆ど

- すぐ試せる

機械学習の長所 (2/4)
メンテナンスフリー

- 学習の元となるデータを与え続ければ最適化される

- ルールベースの場合、ルールをメンテナンスしなければならない
  - 時間経過とともに運用コストが大きくなり、例外も次々と発生
  - 人の引き継ぎや、システム統合が発生した場合、メンテナンスは非常に困難になる

- 機械学習の場合は、ルールではなく、データをメンテナンスする必要がある
  - 自由度はルールベースより高く、スケールする

機械学習の長所 (3/4)
問題に対してスケールする

- 問題のデータサイズを増やしたり、分類対象数を増やしたり、他の似た問題にも展開可能
  - ある部署でうまくいっていた手法を他の部署や会社でも展開可能

- 例：ニュース記事の5カテゴリへの分類を次のように変更可能
  - カテゴリ数を5から100に増やす
  - 分類対象をニュース以外にもブログやメールにも増やす
  - 分類対象を日本語だけでなく、他100言語に増やす

機械学習の長所 (4/4)
人や人工システムを凌駕する性能を出す

- 速度、網羅性、可用性といった部分ではコンピュータが凌駕する

- 人はルールや評価関数をうまく表現できない場合も多い
  - 言語処理・ゲーム・画像認識・音声認識などは知識表現が大変
  - 人工知能の研究分野では、知識・ルール・評価関数を人が明示的に与えることに限界があり、データからの獲得に力を入れた

- 箱庭的な問題だと精度面でも人を凌駕する
  - 関係する特徴数が多い場合
    - 医療診断、広告最適化、スパム分類
  - 評価関数が分からない場合：
    - コンピュータ将棋/囲碁/チェス、機械翻訳、音声認識

機械学習の世界の分類

この二つの問題設定だけは
知っておいてほしいので説明

- 問題設定に基づく分類
  - 教師有学習 / 教師無学習 / 半教師有学習 / 強化学習 など ..
- 戦うドメインの違い
  - 特徴設計屋（各ドメイン毎に, NLP, Image, Bio, Music）
  - 学習アルゴリズム屋（SVM, xx Bayes, CW, ...）
  - 理論屋（統計的学習理論、経験過程、Regret最小化）
  - 最適化実装屋
- 好みの違い
  - Bayesian / Frequentist / Connectionist
  - [Non-|Semi-]Parametric

教師有学習 supervised learning

- 入力xから出力yへの関数 $f(x)=y$を教師事例$\{(x_i, y_i)\}$を元に学習
  - 出力yがカテゴリ値の場合は分類、連続値の場合は回帰と呼ぶ
  - 教師事例を介して、タスクを定義する
    - 教師事例は正解事例、訓練事例とも呼ばれる

- 例１：スパム分類
  - 入力:メール　　出力：スパムor普通
- 例２：株価予測
  - 入力:株価に関係する全情報　　出力: 明日の株価
- 例３：遺伝子と病気の相関分析
  - 入力：各遺伝子の発現量　　出力：患者の症状

本チュートリアルは基本的には教師有学習を扱います

教師無学習 unsupervised learning

- 入力データ$\{x_i\}$のみから学習する
  - クラスタリング、近傍探索、外れ値検出
  - データ表現を学習する場合が多い
    - クラスタリング、主成分分析、ベクトル量子化
  - タスクは事前分布、グラフィカルモデル、制約などで与える

- 例１：店舗クラスタリング
  入力：店舗情報　　出力：似た店舗グループに分類
- 例２：不正アクセス検出
  入力：パケット　　出力：通常と異なるアクセスパターン

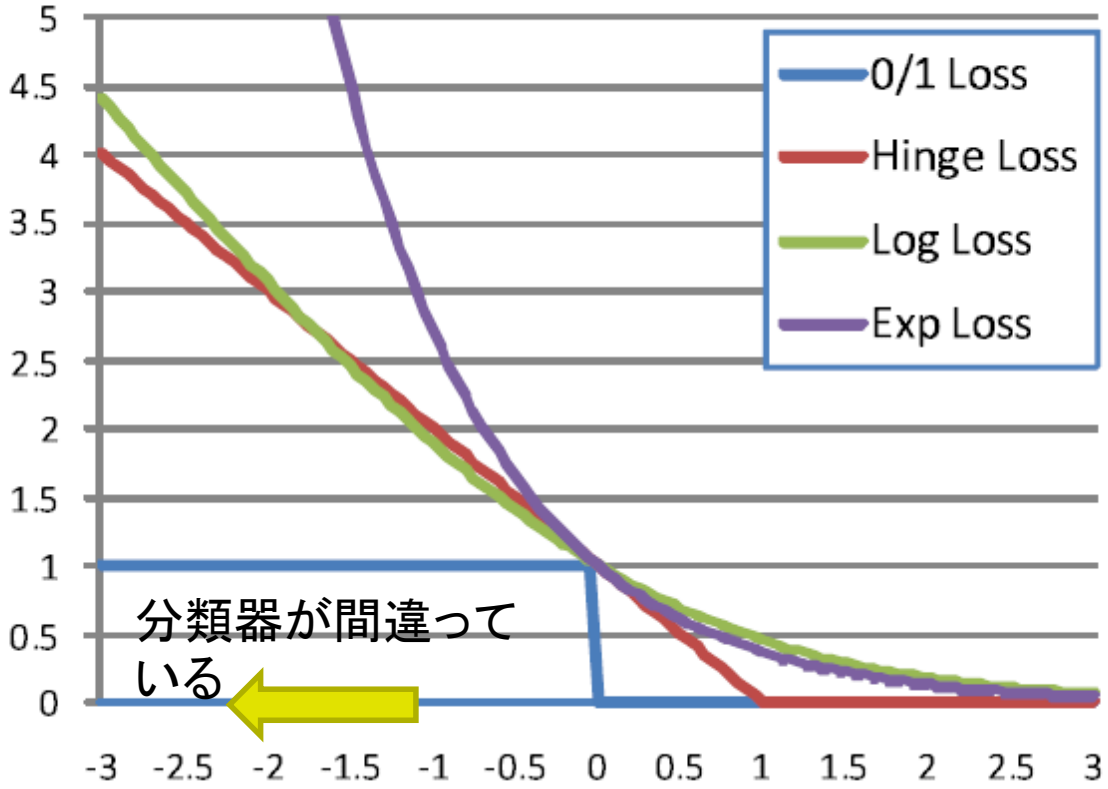
学習 = パラメータwを最適化問題として解く

$w^* = \mathrm{argmin}_w \Sigma_i L(f(x_i; w), y_i) + C\ R(w)$

- 教師事例$\{(x_i, y_i)\}$を利用し、関数を特徴付けるパラメータwを推定する
- $L(f(x_i; w), y)$：損失関数（次項）
  - 予測結果$f(x_i; w)$が正解yと違う場合大きな値をとる
- $R(w)$：正則化項
  - $C>0$はトレードオフパラメーター
- 多くの場合、上記の問題は凸最適化問題となる

損失関数の例

二値分類の場合
$m = y\mathbf{w}^T\mathbf{x}$とした時



$I(m > 0)$ (0/1 Loss)

$\max(0, 1 - m)$
(hinge-loss: SVM)

$\log(1 + \exp(-m))$
(log-loss: logistic 回帰)

$\exp(-m))$
(Exp-Loss:Ada-Boost)

$m = y\mathbf{w}^T\Phi(x)$の値

各損失関数は0/1 Lossの上界かつ、凸な関数

Preferred Infrastructure

機械学習を理解する上で重要な5概念

- 特徴表現
- 汎化・過学習
- 正則化
- 次元の呪い
- BiasとVariance

1.特徴抽出

- 入力データから特徴を抽出し特徴ベクトルで表す
  - テキスト、画像、音声、数値
  - 各領域の専門家による職人芸だったが近年変化（詳細は後述）

世の中ではビッグデータというキーワードが注目されていますが，一口にビッグデータといっても立場や観点によって定義は様々です．

特徴ベクトル

ビッグデータ	2
世の中	1
キーワード	1
定義	1

周囲が黄色	1
中心が茶	1
花びらの割合	0.7
高さ	150

他の要素は0とする

分析結果

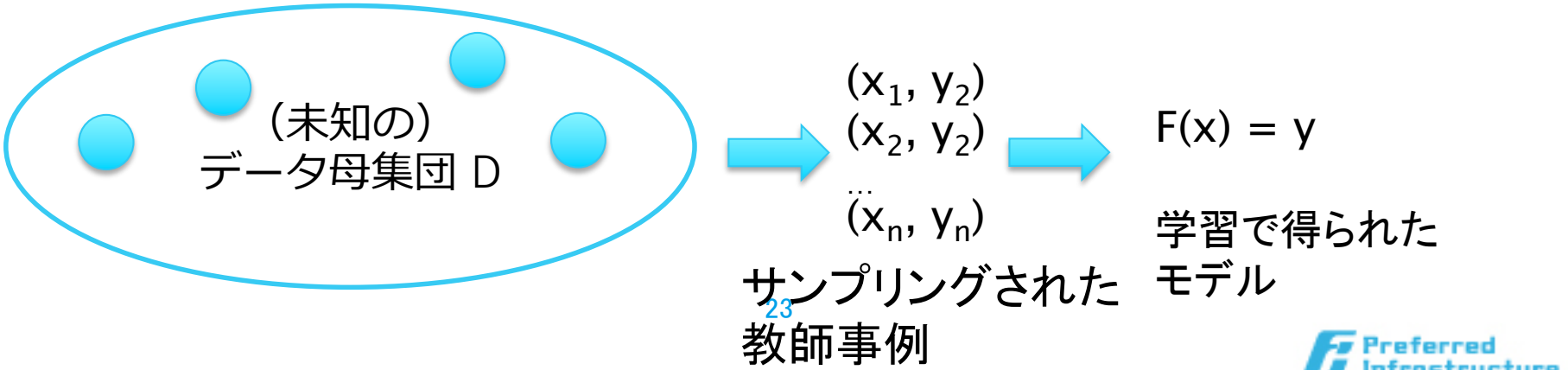
IT関連

ひまわり

2.汎化・過学習

*分類以外も同様である

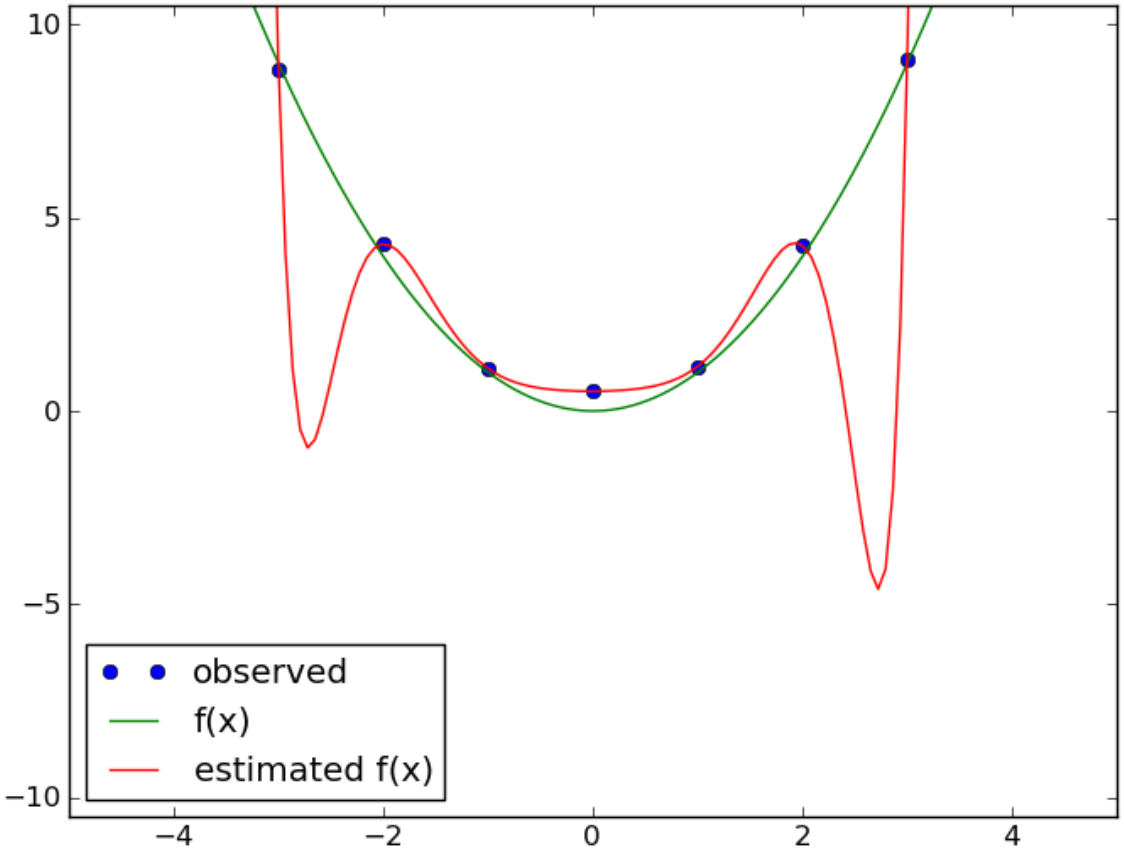
- 「未知のデータを正しく予測*できるような、ルールを学習する」
  - 教師事例を100%正しく分類できる分類器が得られても、未知のデータを正しく分類できるとは限らない
  - 教師事例だけでなく、未知のデータを正しく予測できる能力を汎化能力と呼ぶ
  - 教師事例のみを正しく予測でき、母集団のデータを正しく予測できない状態を過学習と呼ぶ



Preferred Infrastructure

過学習の例

http://research.preferred.jp/2010/12/subgradient-optimization-3/



真の予測したいモデルが $y = x^2$の時、そこから得られた教師事例（黒点）に対し学習させて得られたモデルが赤線。
教師事例は正しく推定できているが元のデータの推定誤差は大きい

3. 正則化

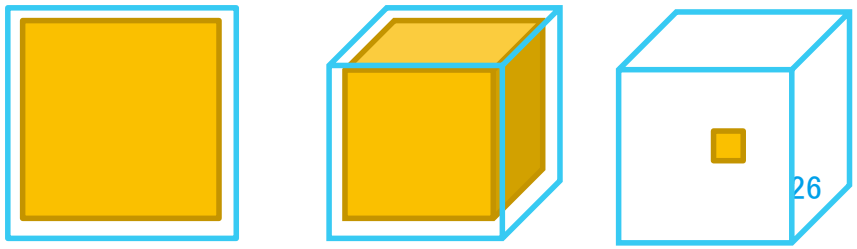
- 学習時に「教師事例を正しく分類する」以外の目標を導入したい
  - 過学習を防ぐことが可能　　例：単純なモデルを選ぶような正則化
  - 問題の事前知識を組み込む　例：少数の特徴のみが問題に関係

- 正則化の多くはパラメータへの制約として帰着できる
  - $L_\infty/L_2/L_1$正則化, Elastic-net, Grouped Lassoなど
  - 例：線形回帰の学習時にパラメータにL_2正則化を適用する

$$f(w) = \sum_i (y_i - w^T x_i)^2 + C|w|^2$$

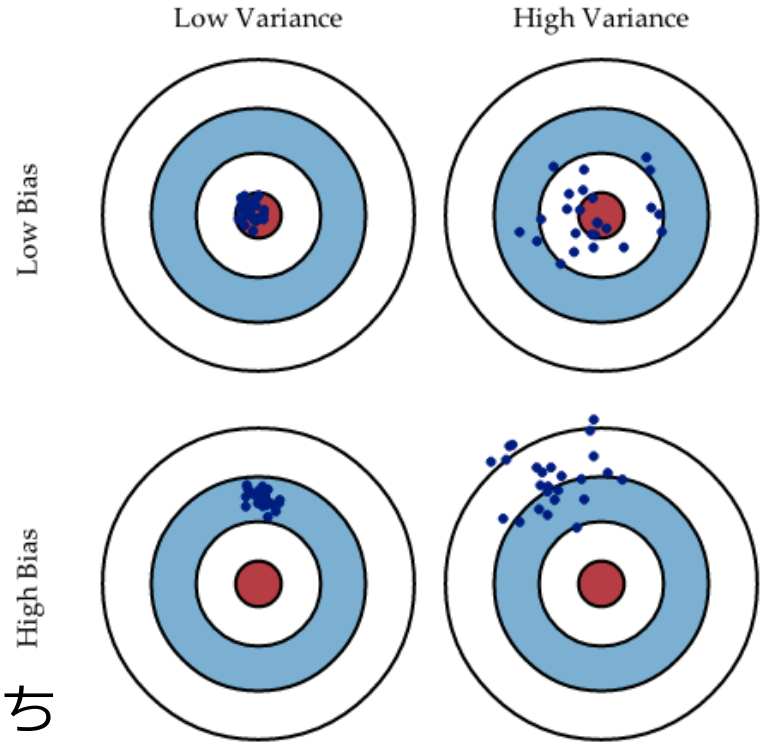
4. 次元の呪い

- 高次元データを扱う場合は、いろいろ面倒なことがでてくる
  - ちなみに自然言語処理やDNA解析では数百万次元は当たり前
- 対象問題の次元数の増加に伴い、必要なサンプル数が急増する
  - d次元で1辺の長さが10である超立方体中の単位超立方体の数
    d=3の場合 1000だが、d=10の場合 1000000000
- 高次元中においては、殆どの点が中心から遠い
  - 1辺の長さが1の超立方体の中で、1辺の長さが0.9の超立方体が占める体積の割合は d=3の場合 72%、d=100の場合 0.002%



5. Bias と Variance

- 学習結果による推定の誤差は次の二つに分解される
- Bias （バイアス）
  - 学習モデルによる期待値と、
    真の期待値の差
- Variance（バリアンス）
  - 推定結果のぶれぐあい
- この二つはトレードオフ
  - バイアス 小 ⇒ バリアンス 大
- 表現力が高い学習モデルは
  バイアス小, バリアンス大になりがち



図はhttp://scott.fortmann-roe.com/docs/BiasVariance.html

真に推定したいモデルが中央である場合、
各学習結果が各青点に対応する

実践編

実際に機械学習を作ってみる

- 機械学習の多くは実装は簡単！
- 今回はオンライン機械学習 + 線形分類器を作る

バッチ学習

- データ全体を受け取ってからパラメータを最適化
- 収束が遅い（特に初期収束）
- 訓練事例全体の保持が必要
- 実装は煩雑
  - 最適化ライブラリとかを利用

オンライン学習

- データを１つずつ受け取るたびにパラメータを即時更新
- 収束が速い
- データは捨てられる場合が多い
- 実装は単純
- Regret解析で性能解析可能

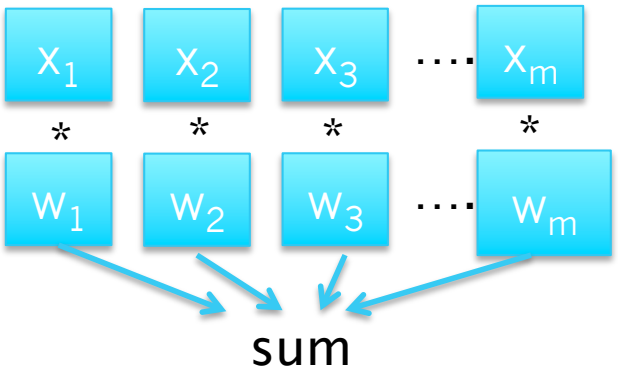
- 近年の殆どの場面でオンライン学習を採用
- 多くのバッチ学習は確率的最急降下法(SGD)でオンライン化可能

線形識別器（二値分類）

$f(\mathbf{x}; \mathbf{w}) := \mathrm{sign}(\mathbf{w}^T\mathbf{x})$

- 入力 $\mathbf{x} \in R^m$ から 出力 $y = \{-1, +1\}$ を予測する識別関数 $f(\mathbf{x}; \mathbf{w})$
  - $\mathbf{w} \in R^m$ が各特徴の重みであり、重み付き多数決で出力を推定

- 単純ベイズ法、SVMs、ロジスティック回帰など多くの分類器が線形識別器に属する
  - 分類の計算量は非零の特徴数に比例



sign(z) := +1 if z >= 0 and -1 otherwise

Preferred Infrastructure

線形識別器の多値分類への拡張

$$f(\mathbf{x}; \mathbf{w}) := \arg\max_y \mathbf{w}_y^T \mathbf{x}$$

- 入力 $\mathbf{x} \in R^m$ から 出力 $y = \{1, 2, \ldots, k\}$ を予測する識別関数 $f(\mathbf{x})$
  - 重みは特徴、出力候補毎に用意

- 以降紹介する二値分類の学習式は$y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}$の部分を

$\Delta := \mathbf{w}_{y^{(i)}}^T\mathbf{x}^{(i)} - \mathbf{w}_{y'}^T\mathbf{x}^{(i)}$に置き換えることで多クラス分類に拡張可能

  - 但しy'は最もスコアが高かった不正解ラベル $y' := \operatorname{argmax}_{y \neq y^{(i)}} \mathbf{w}_y^T\mathbf{x}^{(i)}$

- クラスラベル数が非常に大きくても、$\operatorname{argmax}_{y \neq y^{(i)}} \mathbf{w}_y^T\mathbf{x}^{(i)}$ さえ高速に求まれば学習可能
  - c.f. 構造学習

Preferred Infrastructure

学習アルゴリズムの基本形

- 紹介する学習アルゴリズムは全て次の繰り返しで表現できる

1. 訓練例 $(\mathbf{x}, y)$ を受け取る
2. 現在の識別器で正しく分類できるかを調べる $y\mathbf{w}^{\top}\mathbf{x} > E$?
3. 正しく分類できないのであれば$\mathbf{w}$を次のように更新

   $\mathbf{w} := \mathbf{w} + y\alpha A\mathbf{x}$

   但し、$\alpha > 0 \in R$, $A \in R^{m \times m}$ は半正定値行列（対角行列の場合が多い）

- E, α, Aをいかに適切に設定するかで、学習性能が変わってくる
  - E: 更新条件
  - α : ステップ幅
  - A : 各特徴毎の更新幅．c.f. マハラノビス距離

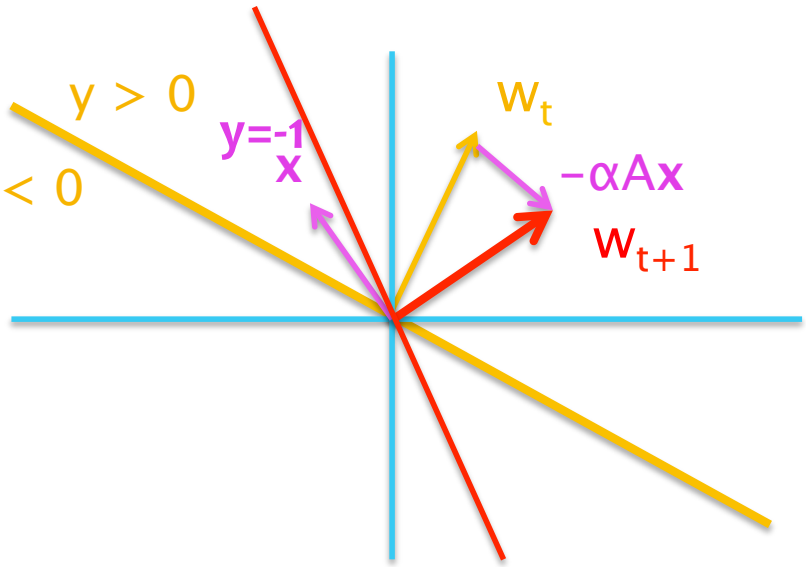
Preferred Infrastructure

更新の意味

- α> 0、Aが半正定値行列の時

$$y\mathbf{w}_{i+1}^{\mathsf{T}}\mathbf{x} = y(\mathbf{w}_i + y\alpha A\mathbf{x})^{\mathsf{T}}\mathbf{x}$$
$$= y\mathbf{w}_i^{\mathsf{T}}\mathbf{x} + y^2\alpha(A\mathbf{x})^{\mathsf{T}}\mathbf{x}$$
$$= y\mathbf{w}_i^{\mathsf{T}}\mathbf{x} + \alpha\mathbf{x}^{\mathsf{T}}A\mathbf{x}$$
$$\geqq y\mathbf{w}_i^{\mathsf{T}}\mathbf{x}$$

常に正

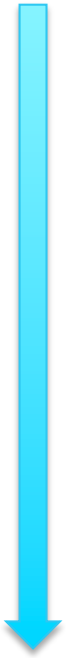


- 今の訓練例は少なくともより正しく分類されるように**w**を更新

線形識別器のオンライン学習アルゴリズム

次の方法を紹介

- Perceptron
- Passive-Aggressive
- Confidence Weighted Learning
- Adaptive Regularization of Weight Vector
- Normal HERD
- Soft Confidence Weighted Learning



登場した年

- 1958
- 2002
- 2006
- 2009
- 2010
- 2012

下に行くほど
より高精度、速く収束

Preferred Infrastructure

Perceptron [Rosenblatt Psy. Rev. 58], [Collins EMNLP 02]

- 訓練例($\mathbf{x}$, y)に対し、現在の線形識別器$\mathbf{w}$で分類できるかを調べ
  誤って分類した場合は $\mathbf{w} := \mathbf{w} + y\mathbf{x}$ と更新
- $E = 0, \alpha = 1, A = I$ に対応

- 単純な更新式だが、多くのタスクで強力な学習手法
- 最終的に得られた$\mathbf{w}$ではなく全ステップの平均$\mathbf{w}_a := \sum \mathbf{w}_i$を利用する
  Averaged Perceptronが良く使われる

- Perceptronは訓練例が線形分離可能な場合、有限回の更新で全ての訓練例を分類できるパラメータを見つけられる（次項）
- 線形分離可能で無い場合でも、多くの訓練例を正しく分類できる$\mathbf{w}$を見つけられる [Collins 02]

定理：訓練例 $\{(x^{(i)}, y^{(i)})\}_{i=1\ldots N}^{N}$, $|x^{(i)}|_2<R$, がある重み$\mathbf{u}$でマージンγで分類可能（$y^{(i)}\mathbf{u}^T\mathbf{x}^{(i)} \geqq \gamma$）ならば、Perceptronの更新回数は高々 $(R/\gamma)^2$回

訓練例数や特徴次元数に依存しない！

証明：

$\mathbf{w}_k$をk回目の更新直前の重みとする.

$$\mathbf{w}^T_{k+1}\mathbf{u} = \mathbf{w}^T_k\mathbf{u} + y^{(i)}\mathbf{x}^{(i)T}\mathbf{u}$$
$$\geqq \mathbf{w}^T_k\mathbf{u} + \gamma$$
$$\geqq k\gamma \quad (\mathbf{w}_o = \mathbf{0})$$

また、

$$|\mathbf{w}_{k+1}|^2=|\mathbf{w}_k|^2 + 2y^{(i)}\mathbf{w}_k{}^T\mathbf{x}^{(i)} + |\mathbf{x}^{(i)}|^2$$
$$\leqq |\mathbf{w}_k|^2+R^2 \qquad (\mathbf{w}_k\text{で間違えた訓練例なので、} y^{(i)}\mathbf{w}_k{}^T\mathbf{x}^{(i)}<0)$$
$$\leqq kR^2$$

上記２つより

$$kR^2 \geqq |\mathbf{w}_{k+1}|^2 \geqq |\mathbf{w}^T_{k+1}\mathbf{u}|^2 \geqq k^2\gamma^2 \Rightarrow (R/\gamma)^2 \geqq k$$

Preferred Infrastructure

Passive Aggressive [Crammer, JMLR 06]

- マージン学習のオンライン版
  - SVMと同じhinge lossを利用
  - Gmailの優先トレイの学習でも利用 [Aberdeen LCCC 2010]
- 次の２つの条件を満たす重み$\mathbf{w}$を探す
  - 現在の訓練例$(\mathbf{x}, y)$を正しく分類
  - 今までの重みベクトル$\mathbf{w}_i$に近い (= これまでの訓練例を正しく分類)
- $\mathbf{w}_{i+1} = \mathrm{argmin}_{\mathbf{w}} |\mathbf{w}-\mathbf{w}_i|^2/2 + C\, L(\mathbf{x}, y, \mathbf{w})^2$
  - 但し、$L(\mathbf{x}, y, \mathbf{w}) = [1 - y\mathbf{w}^\top\mathbf{x}]$　(hinge–loss)
  - この問題は閉じた式で得られる

Preferred Infrastructure

Passive Aggressive (続)

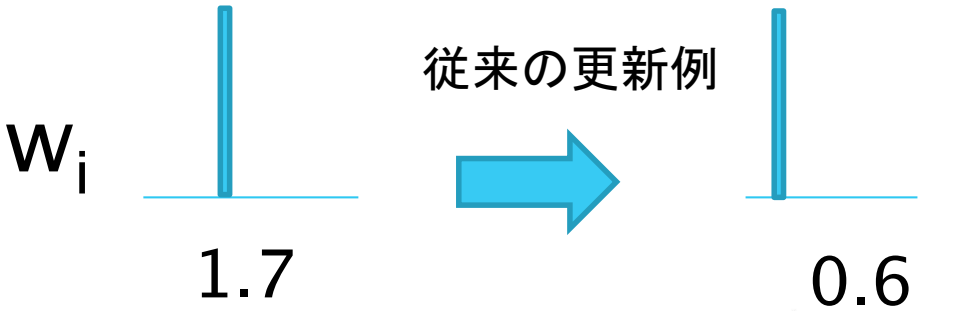
$\mathbf{w}_{i+1} := \mathbf{w}_i + y\, l(\mathbf{x}, y, \mathbf{w}) / (|\mathbf{x}|^2 + 1/C)\ \mathbf{x}$

- PAの最適化問題は閉じた解を持ち、次のように更新可能
  - $E = 1$
  - $\alpha = L(\mathbf{x}, y, \mathbf{w}) / (|\mathbf{x}|^2 + 1/C)$
  - $A = I$
- $\alpha \propto L(\mathbf{x}, y, \mathbf{w})$であり、誤った割合に比例した更新幅を使う

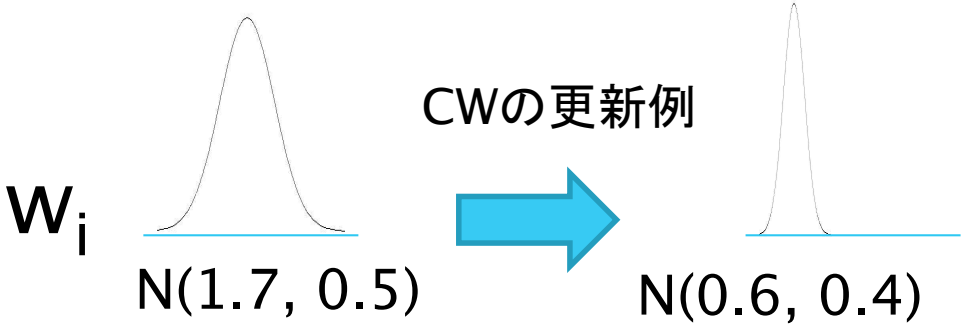
更新式
$\mathbf{w}_{i+1} := \mathbf{w}_i + \alpha A \mathbf{x}$

Confidence Weighted Algorithm (CW)
[K. Crammer, et. al, EMNLP 09]

- 重み**w**がガウス分布$N(\mu, \Sigma)$に従って分布しているとする
  - $\mu \in R^m$は現時点で最良の重みベクトル
  - $\Sigma \in R^{mxm}$は各重みの確信度を表す共分散行列



単一のパラメータ
に足し引きするだけ



パラメータ自体が分布を
持っている
（パラメータ間も）

Preferred Infrastructure

CW（続）

- PAと同じように次の２つを満たす分布（μ, Σ)を探す
  - 現在の訓練例を正しく分類
  - 今までの分布に近い（KL-Divergenceの条件で）
- $\arg\min_{\mu, \Sigma} D_{KL}(N(\mu, \Sigma) \,||\, N(\mu_i, \Sigma_i))$ s.t. $\Pr_{w \sim N(\mu,\Sigma)}[y \mathbf{w}_i^T \mathbf{x} \geqq 0] \geqq \eta$
  - この最適化問題は閉じた解を持つ
  - E, α, Aを$\mathbf{x}$, y, μ, Σ に関して閉じた式で与えることができる
  - PerceptronやPAと比べると複雑な式
- 高い学習効率
  - 自然言語処理の多くのタスクでは1回データを回すだけで収束

Preferred Infrastructure

Task	Perceptron	PA		CW			SVM		
		K=1	K=5	K=1	K=5	K=∞	1 vs. 1	MC	Maxent
20 News	81.07	88.59	88.60	**92.90	**92.78	**92.16	85.18	90.33	88.94
Amazon 7	74.93	76.55	76.72	**78.70	**78.04	**77.98	75.11	76.60	76.40
Amazon 3	92.26	92.47	93.29	†94.01	**94.29	93.54	92.83	93.60	93.60
Enron A	74.23	79.27	80.77	††83.83	†82.23	†82.40	80.23	82.60	82.80
Enron B	66.30	69.93	68.90	**73.57	**72.27	**71.80	65.97	71.87	69.47
NYTD	80.67	83.12	81.31	**84.57	*83.94	83.43	82.95	82.00	83.54
NYTO	78.47	81.93	81.22	†82.72	†82.55	82.02	82.13	81.01	82.53
NYTS	50.80	**56.19**	55.04	54.67	54.26	52.96	55.81	56.74	53.82
Reuters	92.10	93.12	93.30	93.60	**93.67**	93.60	92.97	93.32	93.40

[K. Crammer, et. al, EMNLP 09]

殆んどのタスクで既存のオンライン学習のみでなく通常の学習器より高精度

Task	Instances	Features	Labels	Bal.
20 News	18,828	252,115	20	Y
Amazon 7	13,580	686,724	7	Y
Amazon 3	7,000	494,481	3	Y
Enron A	3,000	13,559	10	N
Enron B	3,000	18,065	10	N
NYTD	10,000	108,671	26	N
NYTO	10,000	108,671	34	N
NYTS	10,000	114,316	20	N
Reuters	4,000	23,699	4	N

News Groupsのトピック
Amazonレビューの上位7タイプ
Amazonレビューの上位タイプ
EnronのUser Aの上位10フォル
EnronのUser Bの上位10フォル
NewYork Times

Preferred Infrastructure

Adaptive Regularization of Weight Vectors (AROW) [Crammer NIPS+ 09]

- CWは訓練例にノイズがある場合に急激に性能が劣化
  - 更新式では今の訓練例を必ず分類するようにしているため
- 学習時に三つの条件を同時に考慮し最適化

  条件1: 現在の訓練例を正しく分類

  条件2: **今までの分布に近い**(KL-Divergenceにおいて)

  条件3: 各特徴のConfidenceを更新毎に上げる

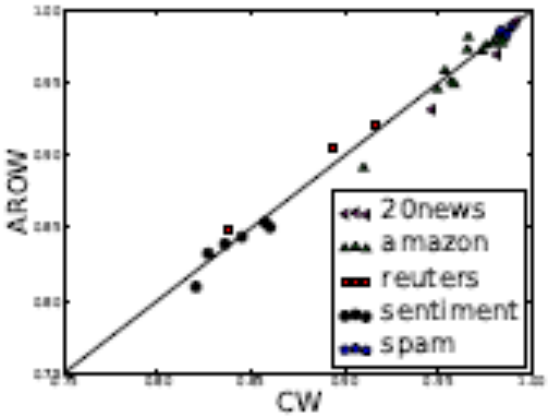
$$\arg\min_{\mu,\Sigma} D_{KL}(N(\boldsymbol{\mu}, \Sigma) \,||\, N(\boldsymbol{\mu}_i, \Sigma_i)) + \lambda_1 L(\mathbf{x}, y, \boldsymbol{\mu}) + \lambda_2 \mathbf{x}^T \Sigma \mathbf{x}$$

条件2　　条件1　　条件3

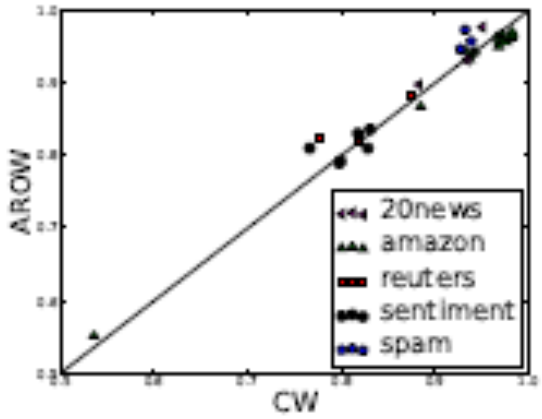
CWでは1が常に最優先

- E, α, Aは閉じた式で求めることができる

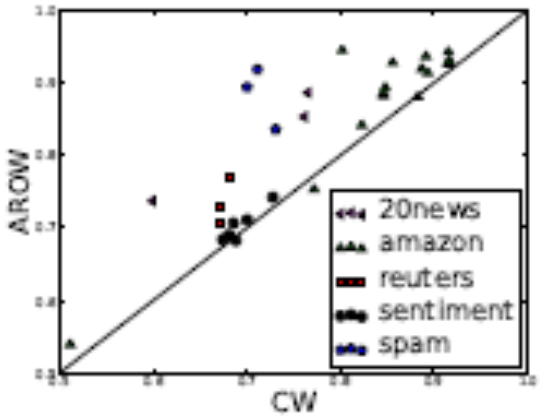
AROWの実験結果



ノイズ 0%



ノイズ 10%



ノイズ 30%

- 左上にあるほどAROW > CW
- ノイズ大⇒AROWの精度>CWの精度

Preferred Infrastructure

NHERD
[Crammer+ NIPS 10]

- 重みベクトル**w**がガウス分布N(**μ**, Σ)に従って分布しているとする
- 各重みベクトルをそれぞれPAの条件に従って更新した後にこれをガウス分布で近似する
  - CWの場合は、現在の分布にKL-divergenceで近い分布で正しく分類できるものを探していたがNHERDはマハラノビス距離上でのユークリッド距離
  - NHERDは重みベクトルの群れ(HERD)を正規化しながら更新

- α, E, Aは閉じた式で求めることができる
  - AROWと比べると積極的な更新を行う

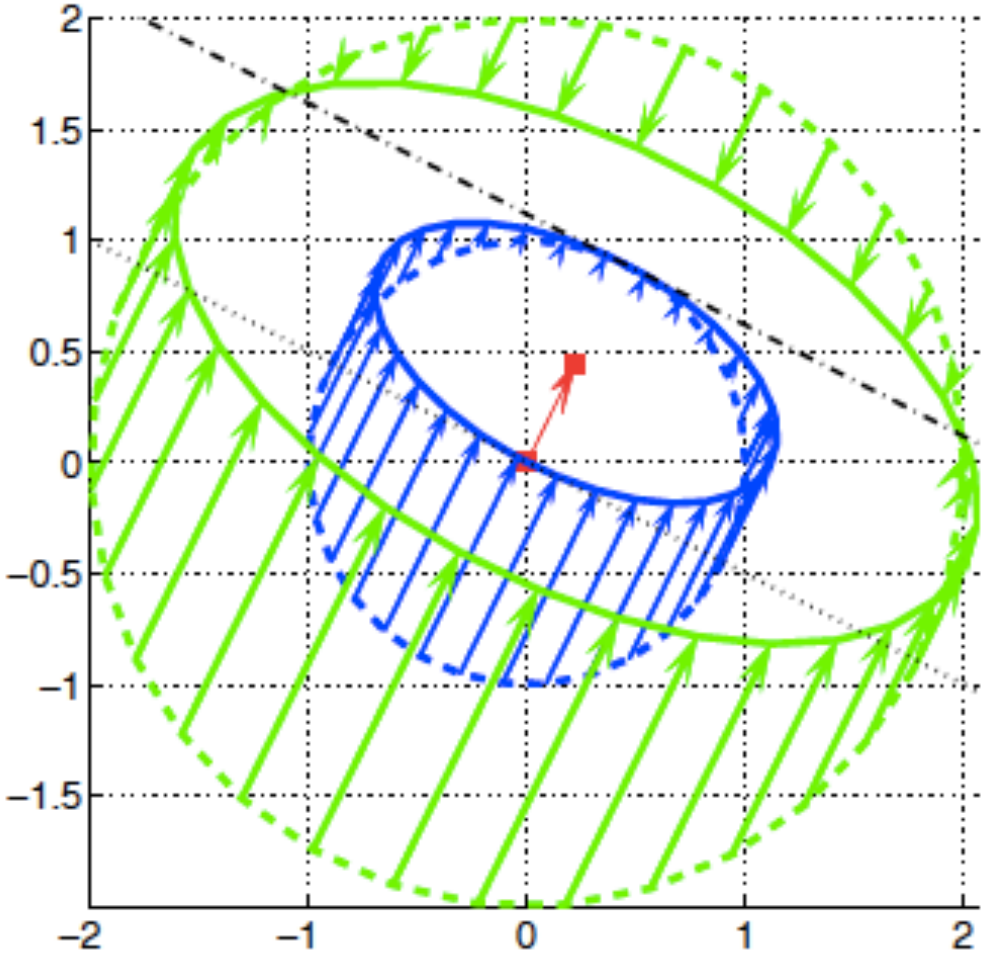
NHERDの更新例

$\mu = (0, 0)$, $\Sigma = I$

に訓練例
$x = (1, 2)$, $y = 1$
を与えた時の更新の様子

青は$|w|=1$, 緑は$|w|=2$の
重みベクトルの集合

真ん中の線は正しく分類、
上側のdash線はマージン1で
正しく分類



[Crammer+ NIPS 10]

Preferred Infrastructure

線形識別器のまとめ

Given (x, y)
If $yw^Tx < E$ then
$\mathbf{w} := \mathbf{w} + \alpha A\mathbf{x}$
Update(A)

$v = x^T\Sigma x$
$b = 1.f + 2yw^Txc$
$\gamma = (-b + (b^2\text{-}8C(yw^Tx - Cv))^{1/2}) / 4vC$
$\beta = (v + r)^{-1}$

	E	α	Update(A) ($a_{rr} :=$)
Perceptron	0	1	1
PA (PA-II)	1	$[1\text{-}yw^Tx] / (\|x\|^2 + 1/C)$	1
CW	γ	γ	$(a_{rr}^{-1} + 2\gamma x_r)^{-1}$
AROW	1	$[1\text{-}yw^Tx]\beta$	$a_{rr} - \beta(a_{rr}x_r)^2$
NHERD	1	$[1\text{-}yw^Tx]/(v + 1/C)$	$(a_{rr}^{-1} + (2C + C^2v)x_r^2)^{-1}$

いずれのアルゴリズムも更新時間は$O(|x|_o)$

Preferred Infrastructure

確率的勾配降下法
(SGD: Stochastic Gradient Descent)

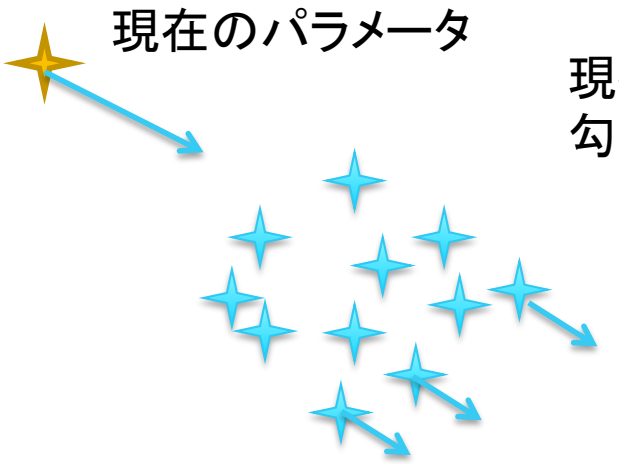
- 従来の勾配降下法
  - $F(\mathbf{w})$を最小化する$\mathbf{w}$を求める際、勾配 $\mathbf{v} = \partial F(\mathbf{w})/\partial \mathbf{w}$ を求め
    $\mathbf{w} := \mathbf{w} - \tau \mathbf{v}$ と更新
  - $\tau > 0$ は学習率と呼ばれる.

- SGD:確率的勾配降下法
  - $\partial F(\mathbf{w})/\partial \mathbf{w}$ を一部のデータから得られた勾配で近似 $\mathbf{v}' := \partial F(\mathbf{w})/\partial \mathbf{w}$
    その情報を元に$\mathbf{w}$を更新 $\mathbf{w} := \mathbf{w} - \tau \mathbf{v}'$
    - 例えば、1～128事例から$\mathbf{v}$を計算

- SGDの方が圧倒的に学習速度が速い。なぜか？

SGDが速い理由

- SGDは最適解から離れている時にはとても速い
  - 通常の勾配降下法と比べて数倍から数百倍速い
  - 少数のサンプルから求められた勾配を使っても、今の位置よりは良い領域にたどり着くのに十分な情報
  - 少ない計算量で大雑把な勾配を求め、更新回数を多くする



現在のパラメータが最適解と離れている場合、
勾配の方向は十分に一致する

各訓練事例の損失関数から
得られる勾配方向 $\partial L(f(x; \mathbf{w}), y)/\partial \mathbf{w}$

より実践的な機械学習へ (1/2)

- 機械学習の研究分野ではベンチマーク性能への固執、評価関数の優位性、応用との乖離が指摘されている [K. L Wagstaff ICML12]
  - 必ずしも機械学習に限った話ではないが・・

より実践的な機械学習へ (2/2)

- 今後は以下の研究課題も検討するべき

- ある問題が与えられた時どのように機械学習として定式化するか？
- データ収集をどのように行うか、どのように正解を作るか？
- データから、いかに特徴量を抽出するか？
- 複数の学習/推論アルゴリズムが存在する時、どれを選ぶか
- 結果をどのように解釈するか
- 世の中の問題における実際の要求スペックを満たせるか
  - サイズ、速度、リソース、安定性、可用性、分かりやすさ
- 開発コストとリターンをあらかじめ予測できるか

機械学習を利用する時の注意

- 機械学習をツールとして利用する際の注意を
  これまでの経験からいくつか紹介する

1. データを増やす >> 手法を頑張る
2. 最新のデータを使う
3. データを実際に観察する
4. 疎なパラメータ、密なパラメータ群で適切な計算を行う

1. データを増やす >> 手法を頑張る

- “大量のデータによる学習は、賢い学習手法を凌駕する”

- 学習データ量を増やすことによる精度の改善は、学習アルゴリズムの改善を凌駕する場合も多い
  - 統計的機械翻訳、音声認識、画像認識、コンピュータ将棋など
  - 例えば、機械翻訳で言えば正解事例（対訳文）の量の対数に精度（BLUEスコア）は比例する

- データ量の勝負にするところまで問題を頑張り、データ量を圧倒的に増やし、改善する

2. 最新のデータを使う

- 例えばレコメンドの場合、
  ユーザーの要求は直近の情報により現れている

- 性別 “女性”
- 過去に最も見たジャンル “化粧品”
- クエリ文字列 “チョコ”
- 直前に見たページ遷移列 “ミッキーマウス 手作りチョコ”
- 見ている時間 “2/13 19:00”

新鮮なデータ

直近の情報を利用できない場合
どんな手法を利用しても予測することは不可能

3. データを実際に観察する

- 機械学習をする際に、その生データの見たことがない人が多い
  - 与えられたものをブラックボックスとして扱っている
- 生のデータを観察し、中身を確認する
  - 必要に応じて、データの要約、可視化も必要
  - 1000ぐらいのデータだったら全部目でみてみる
- 分類結果も見てみる
  - 正解が間違っている、分類軸が間違っている、評価軸が間違っている場合も非常に多い

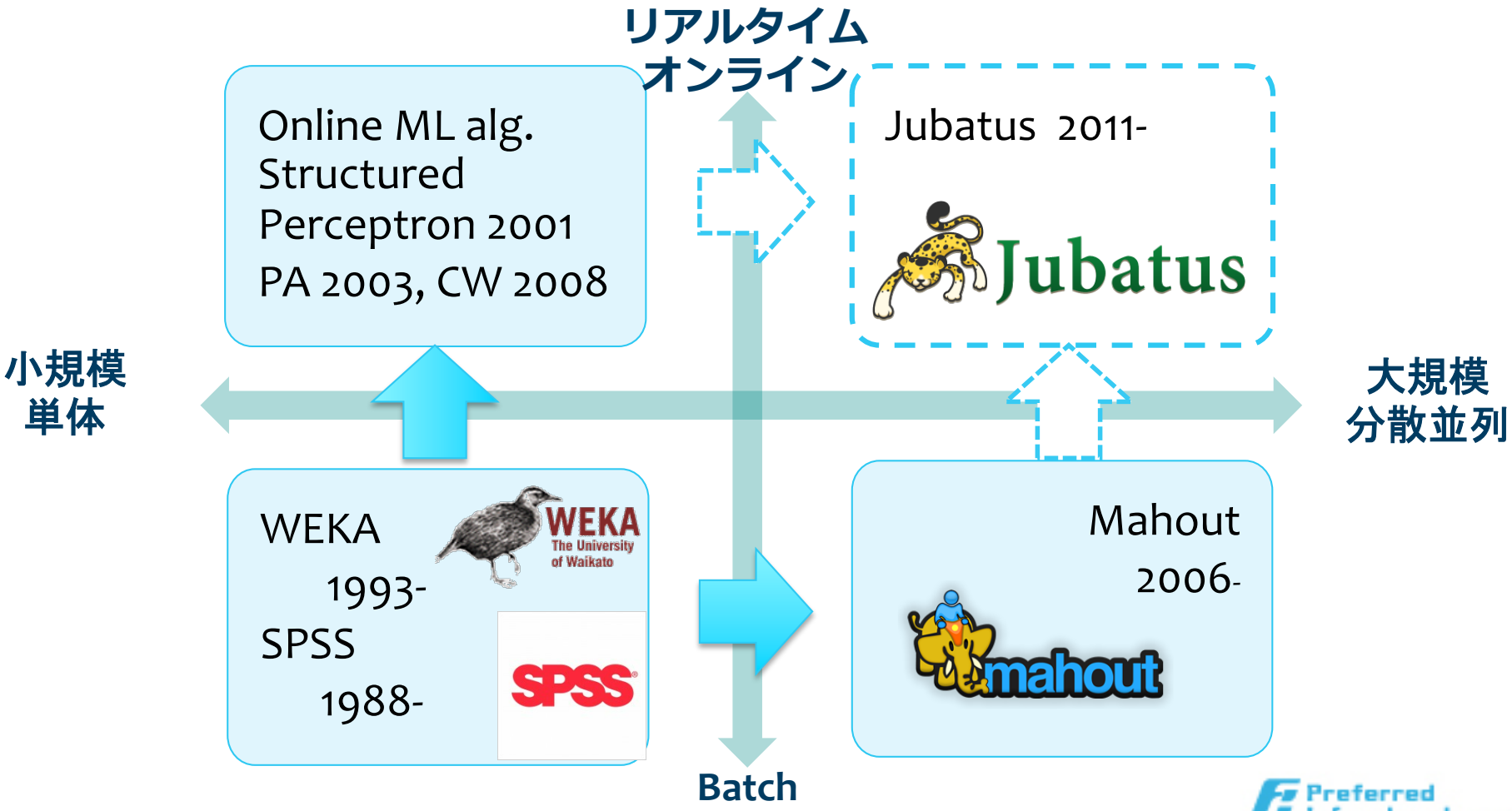
4. 疎なパラメータ、密なパラメータへの対応

- パラメータの性質に合わせて高速化を図る
  - 数十倍から数千倍、速度が変わってくる
- 疎なパラメータの場合
  - 自然言語処理/画像認識/行動履歴がある
  - 数千万次元中、1～100程度の値だけが非ゼロ
  - 非ゼロの要素数に比例する計算手法を活用する
- 密なパラメータの場合
  - 冗長な場合が多い、これを排除して高速化する
  - まず特異値分解を行い、上位の特異値に属する要素だけを利用する
  - 乱択化SVDなどを利用すれば、特異値の近似は1万 x 1万程度の上位10個の特異値分解は0.5秒で求められる (c.f. redsvd)
  - 特異値分解された上で様々な計算を行う、殆ど精度に差があい

発展編

機械学習はスケーラブルかつリアルタイムに

- Jubatusは２つの流れを融合している



Jubatus

- NTT SIC*とPreferred Infrastructureによる共同開発
- OSSで公開 http://jubat.us/

リアルタイム
ストリーム

分散並列

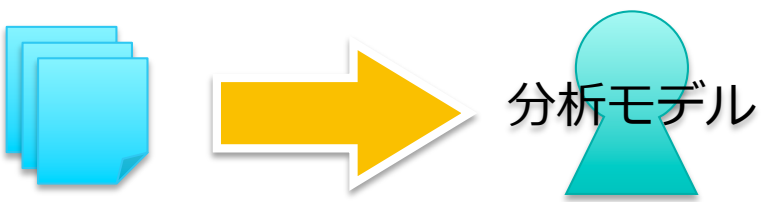
深い解析

* NTT研究所 サイバーコミュニケーション研究所
ソフトウェアイノベーションセンタ

Preferred Infrastructure

特徴1: リアルタイム / ストリーム処理

- 様々な処理をリアルタイム、ストリームで処理
  - 解析結果は、データ投入後すぐ返って来る
  - データを貯めることなく、その場で処理
  - 学習モデル更新もリアルタイムに行う
  - twitterの内容を分析して分類するのは6000QPS
  - 従来バッチで処理していた様々な解析をリアルタイム・ストリームで同様の精度で処理できるよう、新しく解析手法を開発



従来：バッチ処理

Jubatus：ストリーム処理

60

特徴2: 分散並列処理

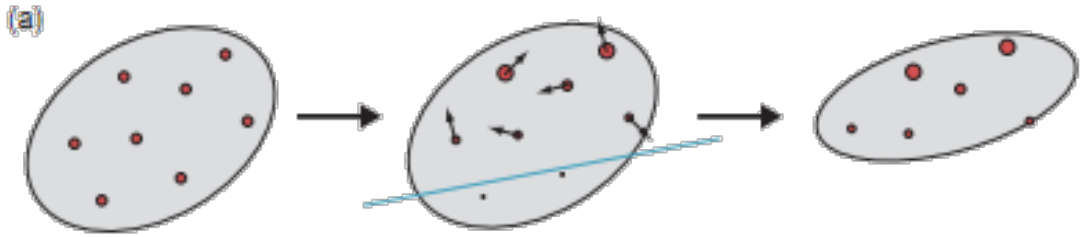
- スケールアウト：ノードを追加することで、性能向上ができる
  - 処理量に応じてシステムの大きさを柔軟に変更可能
  - 小さいデータから大きなデータの処理まで同じシステムで処理
  - 耐故障性も確保

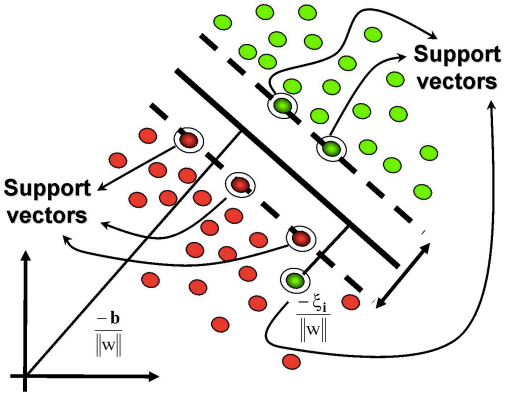
- 各ノードが完全に独立な処理なら簡単だが、それぞれが情報を蓄積し、それらを共有して処理するのは大変
  ⇒　モデルの緩やかな共有で解決（後述）

特徴3: 深い解析

- 単純な集計、統計処理だけではなく、分類・近傍探索・グラフ解析・外れ値検出など様々な機械学習手法をサポート
  - ユーザーはデータを投入すればこれらの分析処理を実現できる

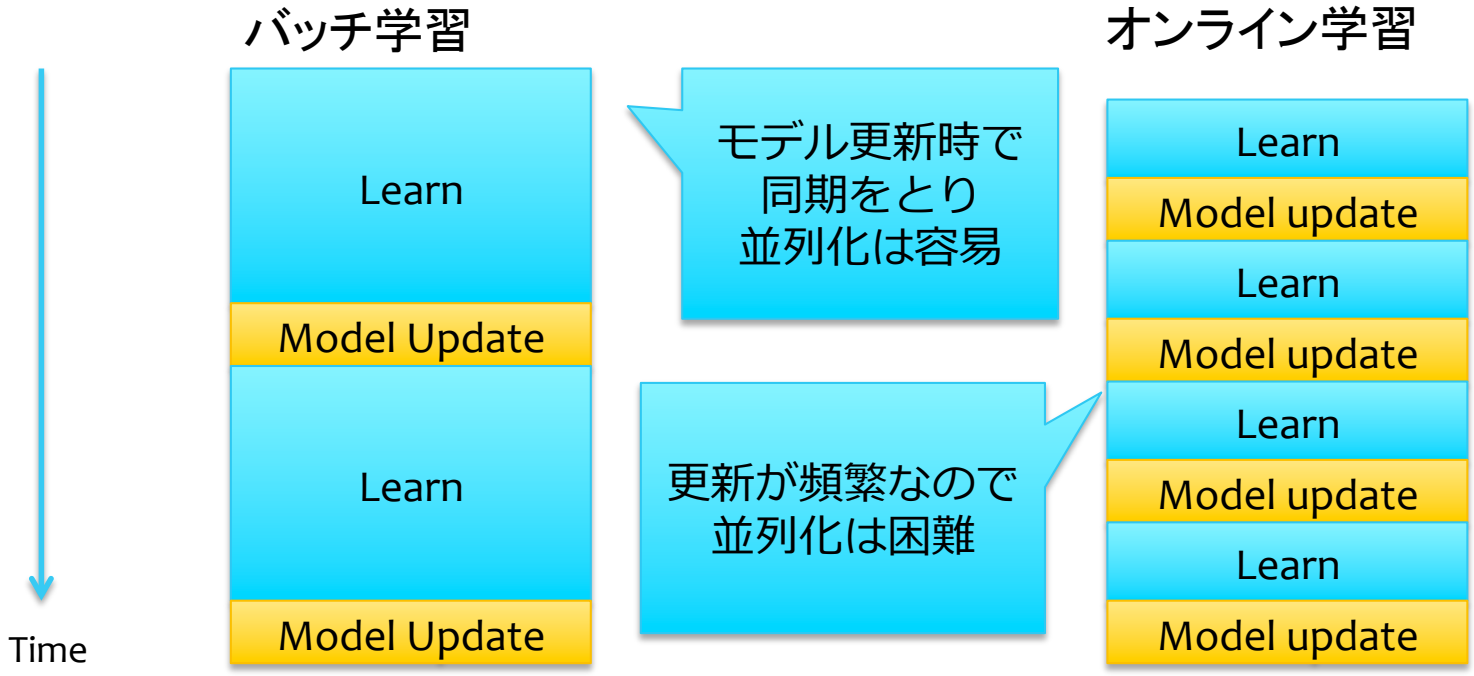
- 非定形データを扱えるように、データからの特徴抽出もサポート
  - 多くの機械学習ライブラリはここがサポートされていない
  - 特徴抽出はプラグイン化され、テキスト、画像、センサ、映像など様々な種類の情報を扱えるように





Preferred Infrastructure

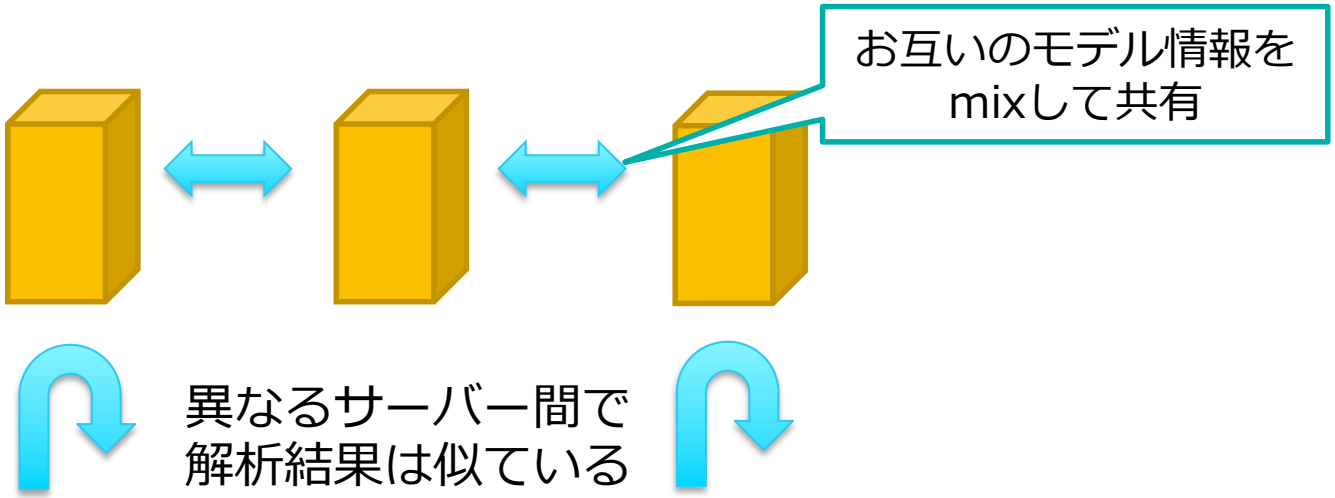
問題：分散とオンラインの融合は困難



- オンライン学習は頻繁に更新する
- オンライン学習をそのまま分散化した場合、モデルの同期コストが非常に大きくなってしまう

解決：緩やかなモデル情報の共有

- Jubatusは各サーバーのモデル情報を「緩やか」に共有する
- データ自体は共有せず、モデルのみ共有する
  - 既存システムと違う割り切り
- 全サーバー間で同時刻に同じモデルが存在することは保証しない
  - サーバー毎に解析が（実用上問題が無い程度に）異なることを許容
  - 時間が経つにつれ、モデル間の情報は共有されていく



Jubatusの優位性：M2Mデータ解析に重要な機能を揃える

- 従来のデータ分析システムと比較したJubatusのアドバンテージ

	Jubatus	Hadoop	CEP	RDBMS
大規模データ蓄積	対象外	◎ HDFS/HBase	対象外	○ 中規模まで
バッチ機械学習	○	○ Mahout	×	◎ SPSS等
ストリーム処理	○	×	◎	×
分散機械学習	◎	○ Mahout	×	×
オンライン機械学習	◎	×	×	×

重要度：高い

CEPとJubatusの比較

CEP

- 複雑なロジックを組むことはできず、ドメイン知識に基づいたルールの設定が必要
- その後も状況の変化とともにルールのメンテナンスが必要
- サーバー間のデータ共有は困難

Jubatus

- 様々な深い分析を行うことが可能
- ルールはデータから自動獲得する
- データや環境の変化に対応
- サーバー間で解析モデルを瞬時に共有

HadoopとJubatusの比較

Hadoop

- データ分析基盤を更に実現しなければならない
- データ解析は全処理終了時に得られる
- 様々なエコシステムが存在、プレイヤーは多い

Jubatus

- あらかじめデータ分析に必要な機能が備わる
- データ解析は瞬時に実現
- 分散によるオーバーヘッドは殆ど存在しない
  - 各モデルはローカルに保存

MahoutとJubatusの比較

Mahout

- バッチ機械学習をサポート
- 分散並列化されているのは一部で、スケーラブルではない
- 実装のクオリティはまちまちであり、統一感はなく商用レベルではない

Jubatus

- オンライン機械学習をサポート
- 全ての処理が分散並列化
- 同じ開発コミュニティが継続的にメンテナンスし、改善

Jubatusの機能概要

- Jubatusでは様々な分析手法をサポート
  - 多値分類・回帰
  - 統計
  - 近傍探索
  - グラフ解析
  - 外れ値検出
- これらを組合せることにより、多くの課題を解決が可能
  - スパムフィルタ（メール分類）
  - 電力消費量予測（回帰）
  - ユーザー属性推定（レコメンデーション）
  - ログからの異常検知（外れ値検出）
  - 攻撃の標的になりやすいハブノードの発見（グラフの中心性）

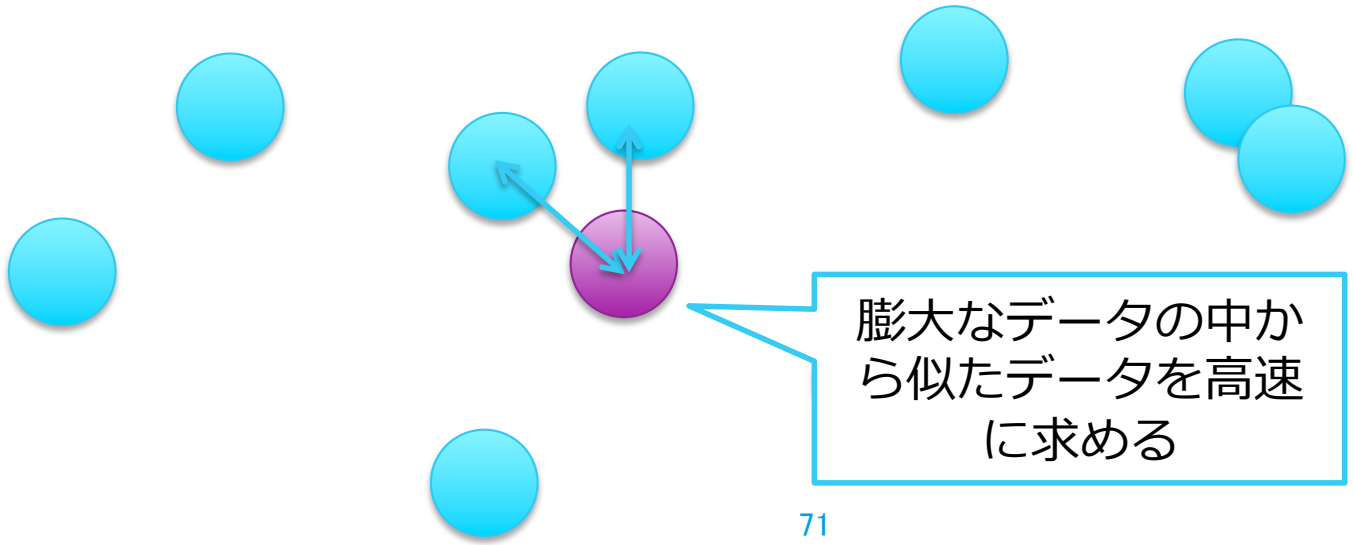
多クラス分類・回帰

- 入力xに対し、出力yを推定する
  - 正解データ{(x, y)}を利用しxからyの関数を学習

タスク	入力x	出力y
メール分類	メール	スパム or 普通 or 重要等
Twitterのユーザー分析	Tweet	ユーザーの性別、職業、年齢など
電気使用料需要の予測	パケット	各サーバーの予測使用量（連続値）
広告のコンバージョン予測	アクセス履歴、広告	クリック、コンバージョンするか
監視カメラ解析	監視カメラ画像	部屋の状態（明かりがついている？人がいるか？など）

近傍探索を用いた推薦（レコメンド）機能

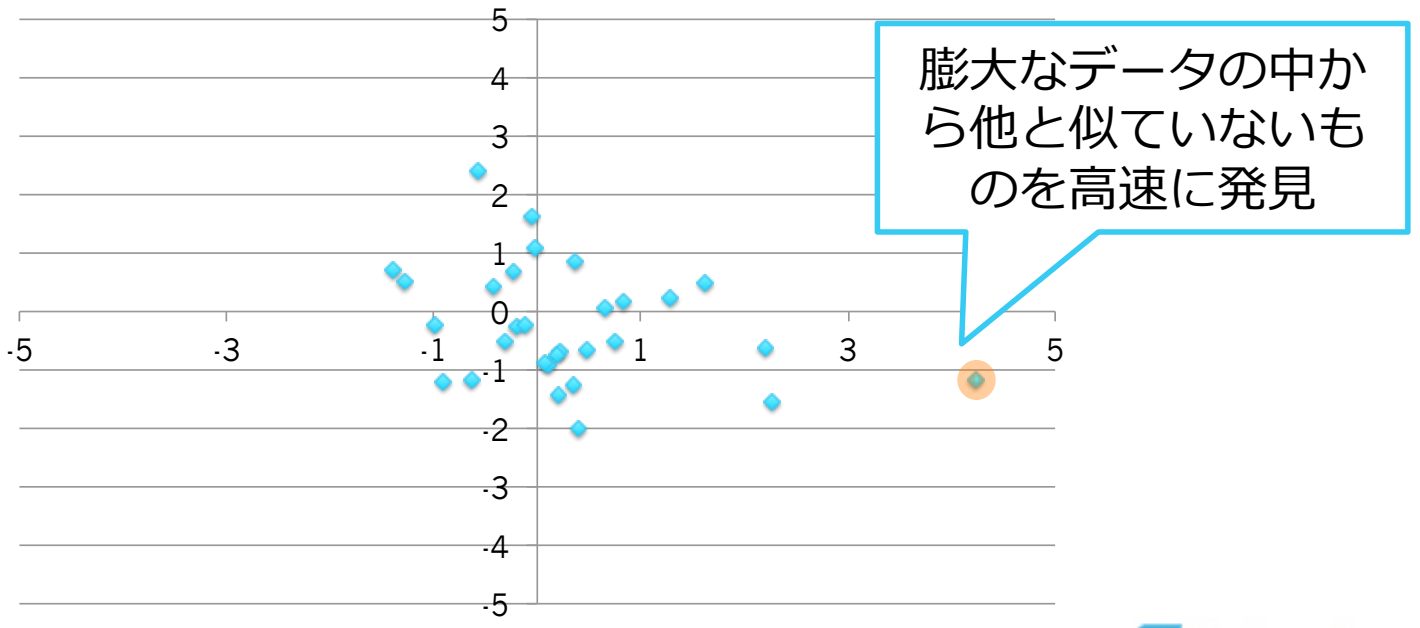
- 分散＋リアルタイム＋深い解析はレコメンドでも実現！
- 分散：データは分散環境で管理され、スケーラブル
- リアルタイム：登録したデータは即時に、推薦対象となる.
  推薦結果はすぐに返って来る
- 深い解析：各データはデータから特徴抽出を行った上で類似度計算



Preferred Infrastructure

外れ値検出

- 元データが高次元でも高精度な近傍探索によって外れ値検出を実現
- アルゴリズム: オンライン外れ値検出アルゴリズムを分散用に改良
- 距離計算: 裏側の近傍探索機能によって近似的に計算

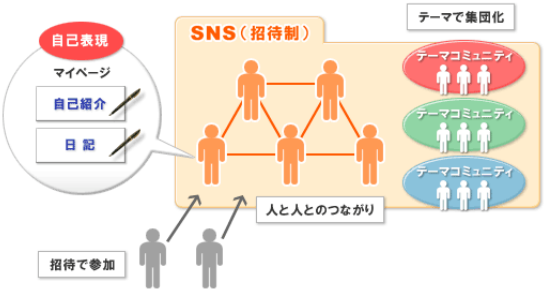


Preferred Infrastructure

グラフ解析

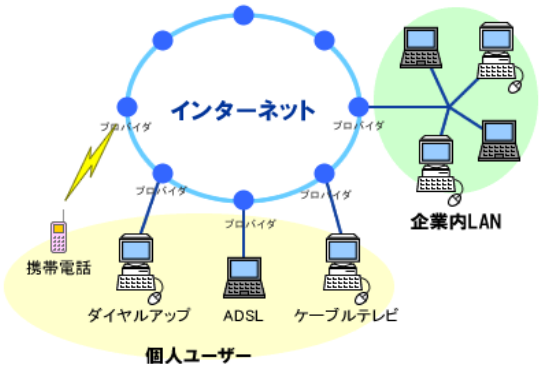
- 点と枝の集合で表されるグラフデータに対し、最短路探索、中心性の計算を提供
- グラフ中の点や枝の追加、削除、更新に対応
- 真の値に近い最短路経路を非常に高速に求める
- 中心性の計算も高速に求める

◆SNS



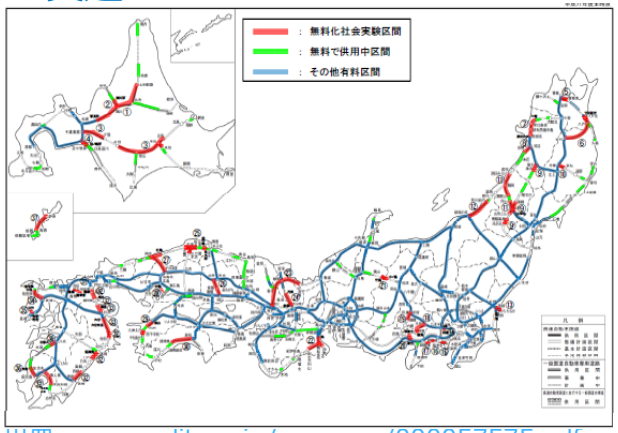
出典：http://idc.7-dj.com/sns/feat.html

◆インターネット



出典：http://www.soumu.go.jp/main_sosiki/joho_tsusin/security/kiso/illust/internet.gif

◆交通



出典：www.mlit.go.jp/common/000057575.pdf

Preferred Infrastructure

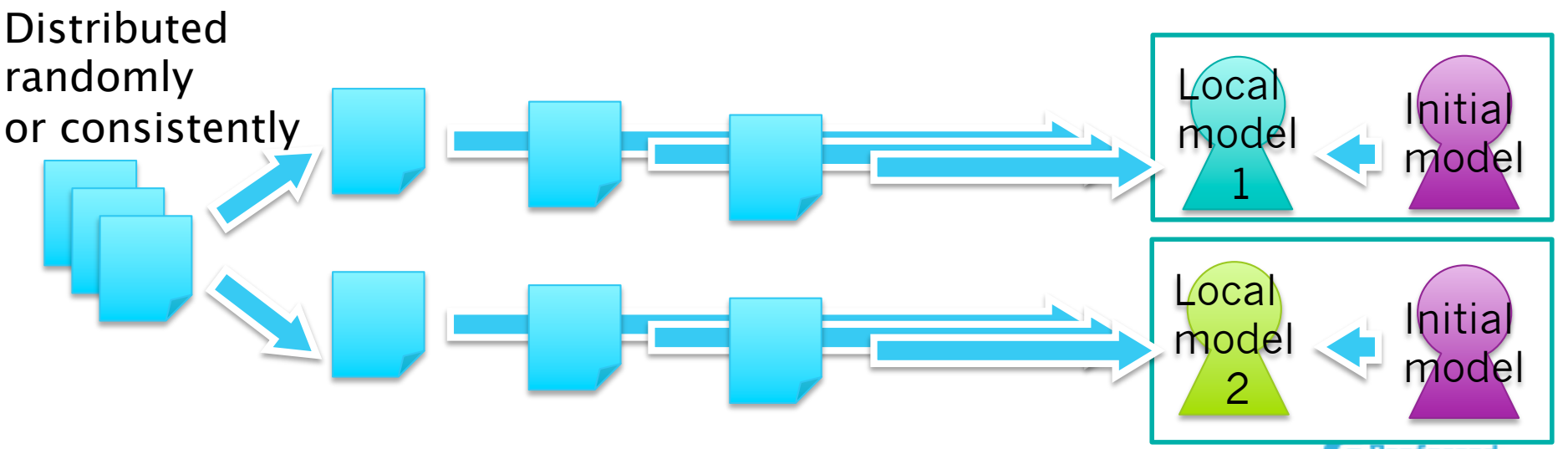
Jubatusの基本操作
UPDATE, ANALYZE, and MIX

1. UPDATE
   - データを受け取り，学習操作を行いローカルモデルを更新
2. ANALYZE
   - データを受け取り，ローカルモデルを利用し分析結果を返す
3. MIX (automatically executed in backend)
   - 複数のローカルモデルを混ぜ，その結果を返す

- C.f. Map-Shuffle-Reduce operations on Hadoop
- アルゴリズム設計者は次の問題を気にする必要はない
  - 分散ロジック
  - データ共有
  - 対故障性

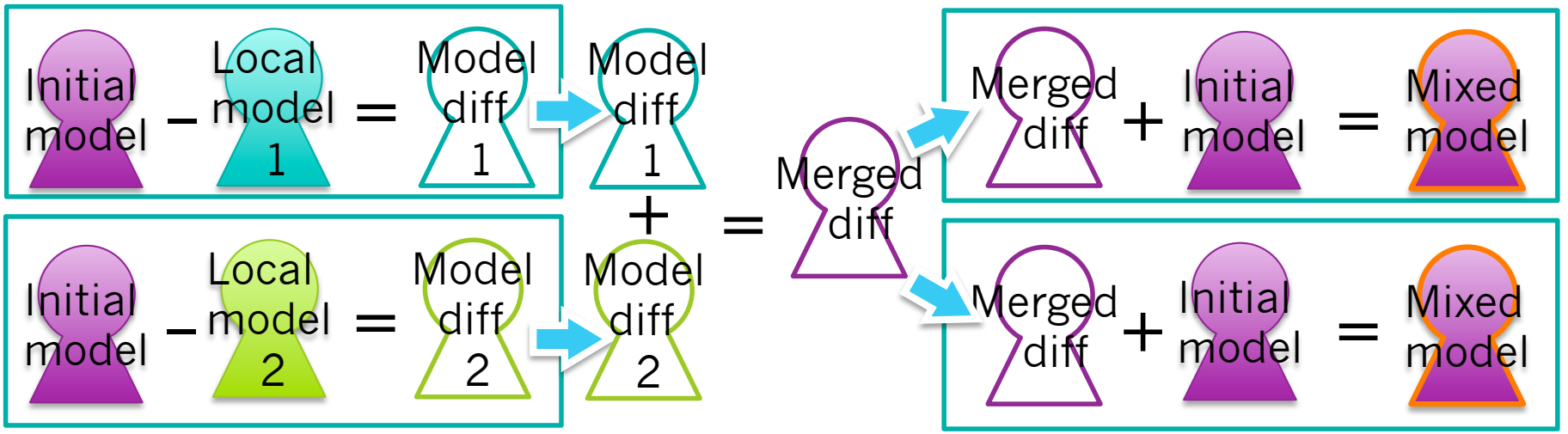
UPDATE

- データは任意のサーバーに送られる
- データに基づきローカルモデルがアップデートされる
- データは共有しない



MIX

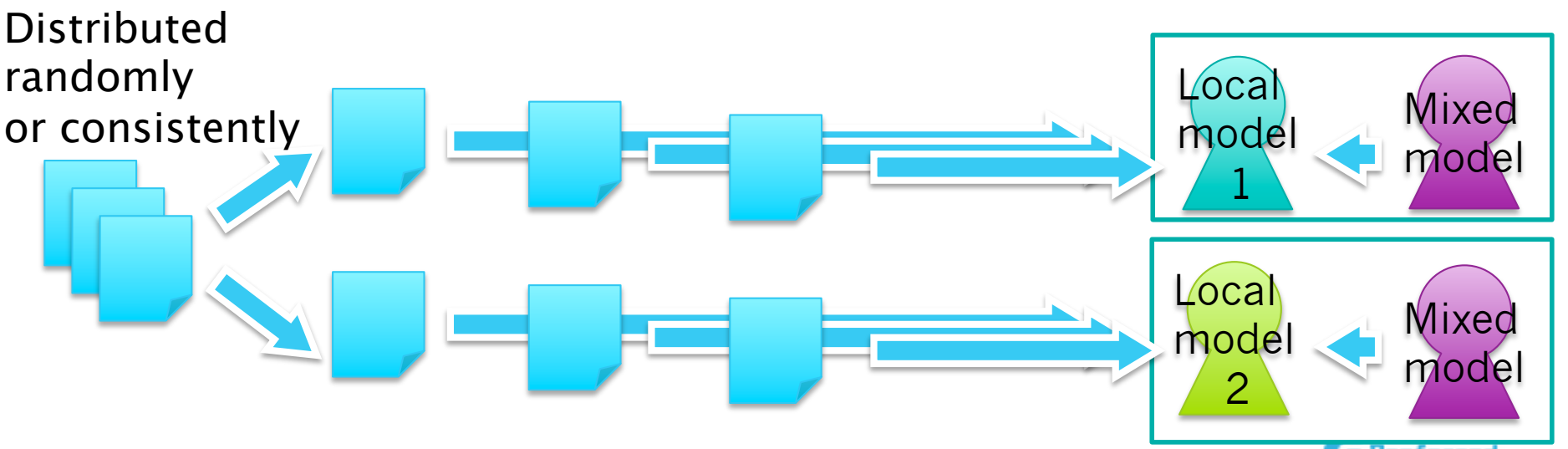
- 各サーバーはローカルモデルの差分を送る
- モデルの差分はマージされ、再度配布される
- モデルの差分はモデル自身と比べ非常に小さく転送コストは小さい



Preferred Infrastructure

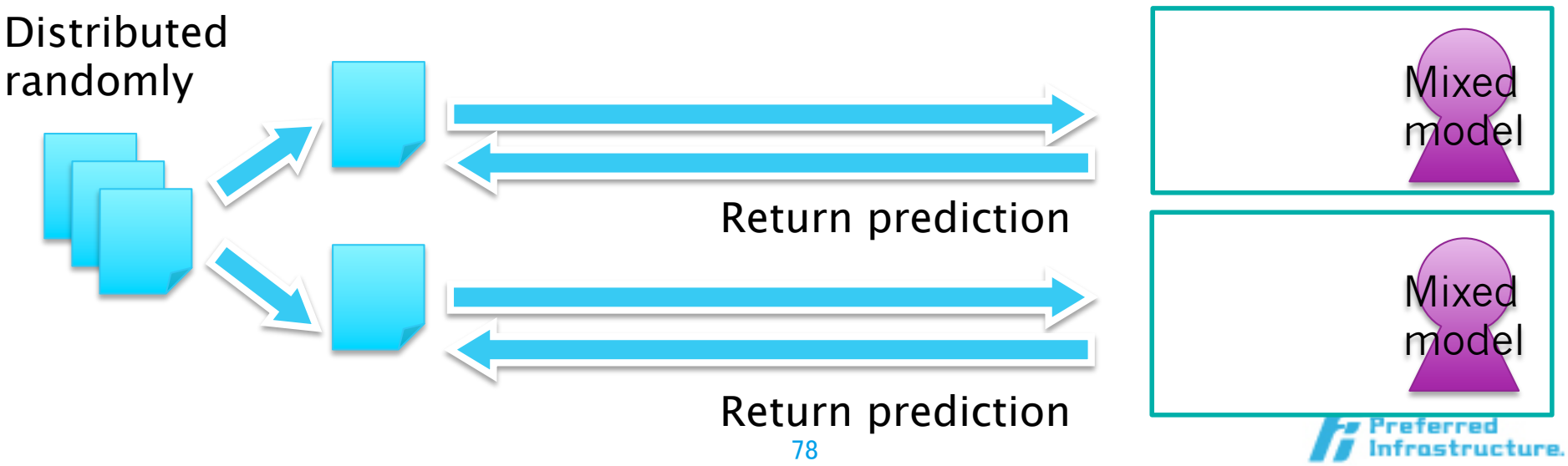
UPDATE (iteration)

- 各サーバーはMIX後のモデルから学習を行う
- MIX後のモデルは全サーバーの情報を持っており徐々に改善される



ANALYZE

- データは各サーバーにランダムに送られる
- 各サーバーは、ローカルモデルのみを元に分析を行う
  - 他のサーバーへの問い合わせは行わず，全てローカルで処理
- 結果はクライアントに返される



Preferred Infrastructure

まとめ：Jubatusがなぜリアルタイムで処理できるか

1. オンライン機械学習を採用している
   - オンライン機械学習を分散化するための手法は開発
2. UPDATE処理はローカルに行わる
   - UPDATE時に他のサーバーとの通信は必要ではない
3. モデル情報のみをMIXする
   - MIXは小さく、転送コストは小さくなり、低レイテンシで共有可能
4. ANALYZE処理はローカルに行われる各サーバーはMIX後のモデルを持ち，他に問い合わせしなくても全体の情報をしっている
   - 低レイテンシで推定可能
5. 全ての処理はメモリ上で行われる
   - モデル情報は小さくメモリ上で処理可能

Jubatusの今後

- 仮説検証を様々なドメインで実施中
  - ソーシャルデータ
  - セキュリティ
  - HEMS / BEMS
  - 車
  - 特にM2M分野と既存分野の融合領域
- 新機能開発
  - クラスタリング、時系列解析などのサポート
  - 秘匿化データマイニング，組み込み向けの改良
  - ドメインに応じた分析手法の開発も可能

- 一緒にJubatusの可能性を検証できるパートナーを探しています！

Deep Learning
深層学習

DeepLearning
深層学習

- 機械学習は次の２つのステップからなると説明した
  - STEP1 入力データからの特徴抽出
  - STEP2 特徴に対する学習・推論
- 特徴抽出は今でもドメイン知識や人手による試行錯誤が必要
  - 自然言語処理、画像処理、行動履歴 etc. 毎に異なる技
  - Feature Engineeringとも呼ばれる
  - どれを使うか、どう組み合わせるのか、値はどうするのか

- 特徴抽出も自動化できないか？
  - 特徴抽出は機械学習の実用上なボトルネック
  - 人手より最適化できないか？

⇒　深層学習

ニューラルネット(NN)の歴史 (1/2)

- 1940年代頃から何度もブームが
  - Perceptron, BackPropagation, ...
  - しかし90年代頃からの長い冬の時代
- 2006年からDeep Neural Netとしての復活
  - Hinton, BengioらによるPreTraining とAutoEncoderの登場
    - 深い階層を持った場合でも勾配が拡散せず学習できる
  - Ngらによる視覚野の働きに似た画像特徴抽出
    - 人の視覚認識の仕組みを部分的に再現できた
  - しかしまだ一部の研究者のみが注目している状況

ニューラルネット(NN)の歴史 (2/2)

- 2010年以降の発展
  - 多くの分野でのベンチマークテストによる圧勝
    - 一般物体画像認識、音声認識、薬物活性予測
    - これまでのstate-of-the-artを大きく凌駕すると共に、
      非専門家が達成したことに衝撃
  - 大規模NNからの教師無学習（後述）
    - Googleらがは1000万枚の画像を利用してパラメータ数が数十億
      からなる大規模NNを2000台（16000コア）で1週間で学習
    - 教師無し学習で知識のようなものが得られる

ブログ記事：「ニューラルネットの逆襲」
http://research.preferred.jp/2012/11/deep-learning/

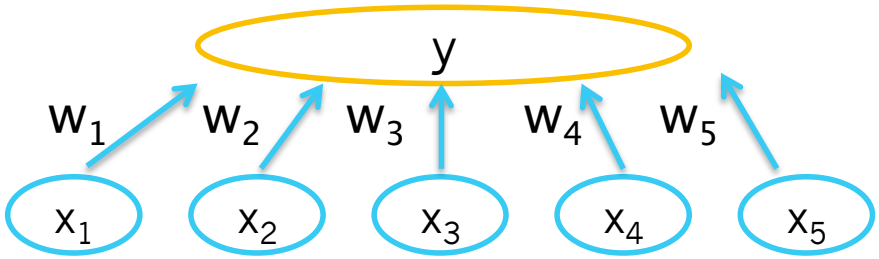
なぜ深層学習がこれほど成功したか

- Hinton曰く：「Bengioらが90年代に培った手法」
  + 「大規模データ」 + 「DropOut」
  - 特に大きなブレークスルーがあったわけではない
- 学習手法の改善
  - PreTraning, AutoEncoder, Dropout, Maxout、学習率調整
- 実装技術の改善
  - GPGPUや大規模クラスタの利用

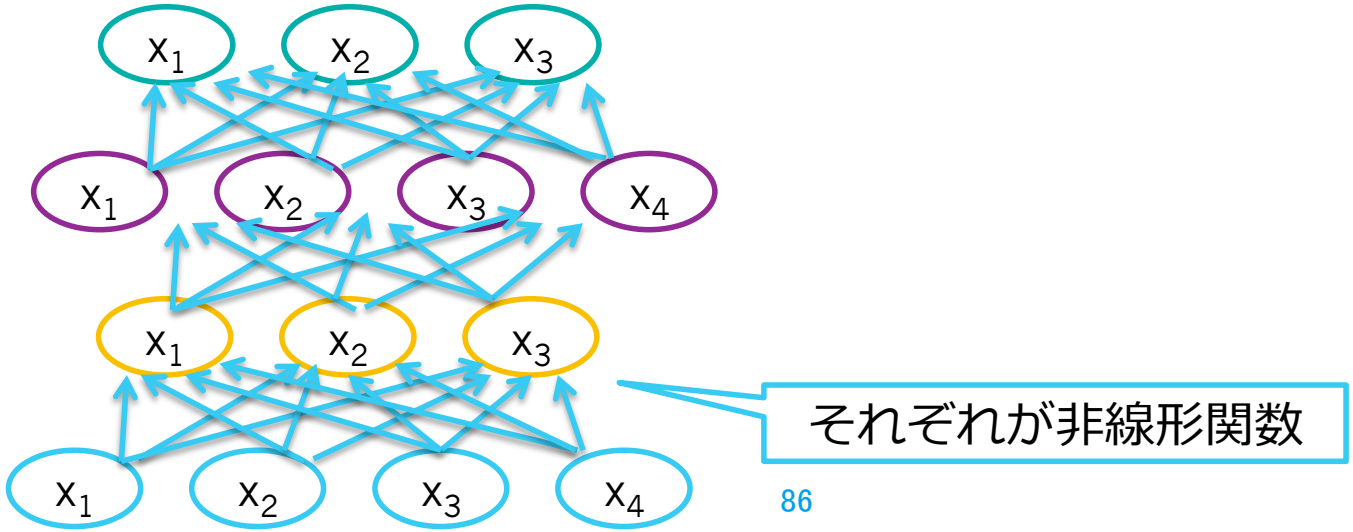
- ニューラルネットはこれまで注目されていなかっただけ
  + これまでの学習手法の煮詰まり感
  - 既存手法（線形分類器、ベイズ、カーネル法）に関して
    やれることはほぼやった。特徴抽出は比較的手付かず

NNの基礎

- これまで扱ってきた多くの学習器は１層の線形分類器



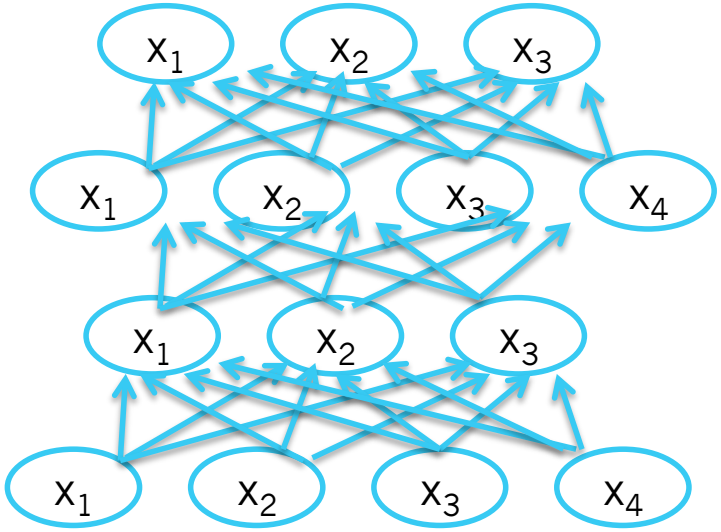
- 多層ニューラルネットは、各層が隣接層とくっついている



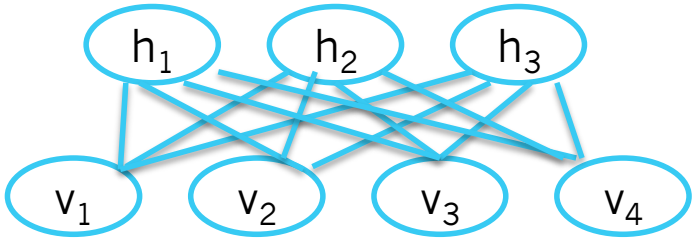
Preferred Infrastructure

NNの代表的なアーキテクチャー

- 多層NN

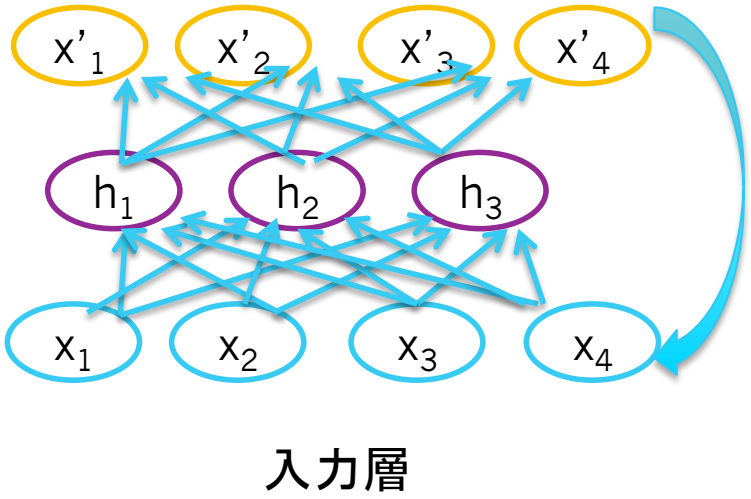


- Restricted Boltzmann Machine (RBM)
  - 観測層(v)と隠れ層(h)の間にのみ依存関係がある確率分布



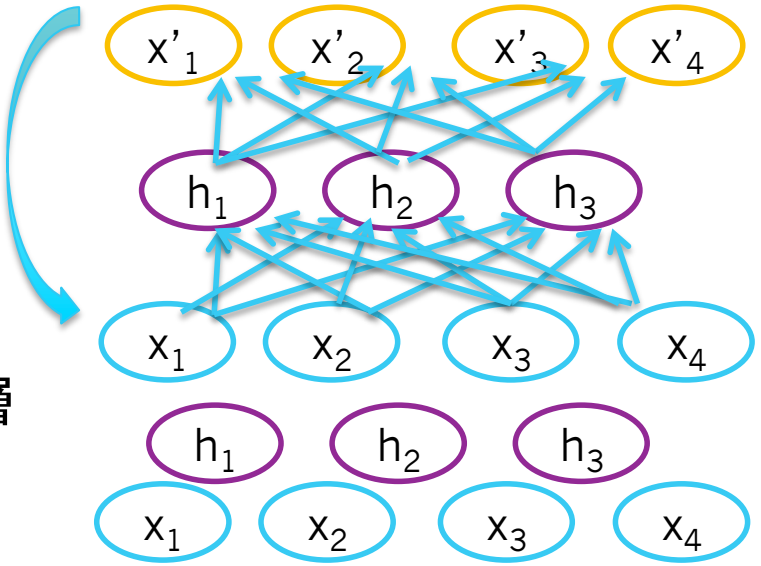
NNの工夫１：PreTraining, AutoEncoder

- 多層の場合BackPropagation時に更新項が拡散する問題があった
- 各層毎の重みをデータからの教師無学習で初期化しておき解決
  - 元のデータをうまく復元できるように各層をGreedyに学習
    ⇒ データを段階的に抽象化した表現を得る



学習して得られた出力を入力とみなし次の層を同じように学習
これを繰り返す

入力層と出力層の値が同じになるように学習
＝データの低次元表現を学習



Preferred Infrastructure

NNの工夫2：DropOut

- NNは表現力の高いモデルなので過学習しやすい
- 各データ毎の更新時にランダムに素子の半分を消した状態で学習
  - ランダムに各素子の出力を 0 とみなす
- 推定時には、各素子からの出力を1/2にして推定する
  - これは1層の場合は各学習時の平均に対応する
- 過学習を防ぐ役割を果たす
  - 複数モデルの結果の平均をとっていることに対応

- MaxOutと呼ばれる関数を利用した場合、多層であってもDropOutが良い近似となり高い学習性能を示す
  - 現在多くのタスクでMaxOut+DropOutが最高性能

学習の自動チューニング

- ニューラルネットには複数のハイパーパラメータが存在
  - 各層毎の学習率（SGDにおける $\theta - \mu \partial F(\theta)/\partial\theta$ のμ）
  - 各層のサイズ、つなぎ方
- ハイパーパラメーターの調整をしないと簡単に値が“吹っ飛ぶ“
  - 異なるハイパーパラメータで複数同時に試してうまくいった組み合わせを選ぶのがこれまでの主流だった

- 学習率自動チューニングする手法がいくつか提案される
  - AdaGrad [J. Duchi+, JMLR 2011]
  - vSGD-fd [T. Schaul+, Arxiv 2013]

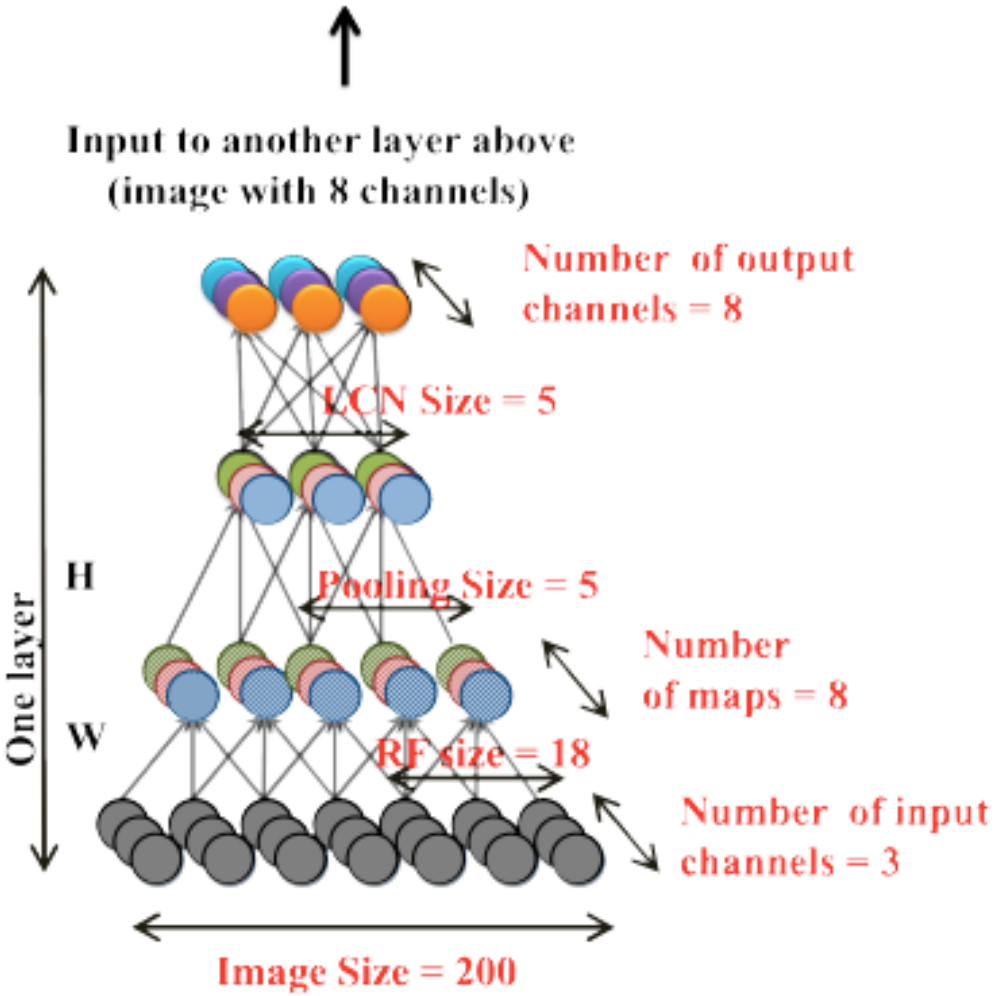
DistBelief
[J. Dean+, NIPS 13]

非同期での大規模学習をサポート
- オンラインの並列学習　(SGD)
- バッチ学習の並列学習 (L-BFGS)

Youtubeから得られた 200 x 200の画像 1000万枚に対して教師無し学習

(AutoEncoderと似たRICAと呼ばれる方法でDNNを学習)

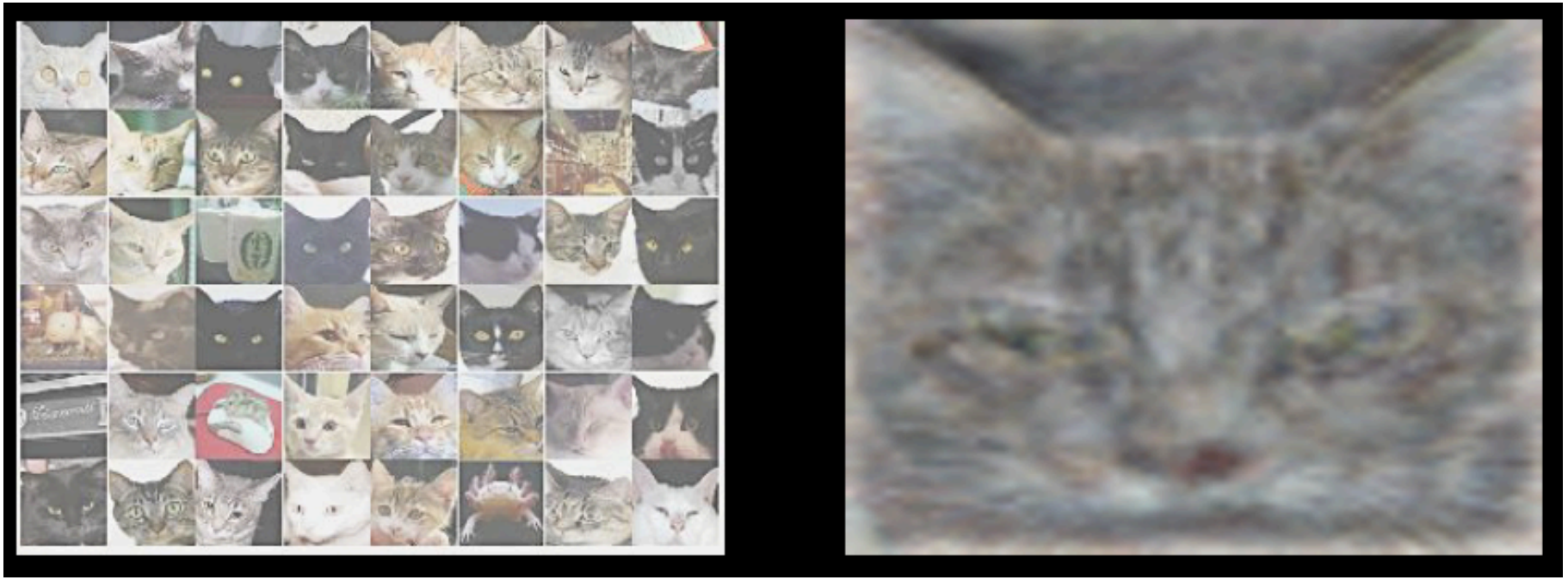
次項以降で学習で得られた最上部のニューロンが反応する画像を紹介



Preferred Infrastructure

右：学習で得られたニューロンが最も反応する画像
左：このニューロンに反応したテスト画像

右：学習で得られたニューロンが最も反応する画像
左：このニューロンに反応したテスト画像



まとめ

まとめ

- 機械学習についての考え方、使い方について紹介した
  - 機械学習は複数の分野にまたがる境界領域
- 研究から見た機械学習
  - 機械学習自体の問題に取りくむ　⇒　多様な問題が各分野で存在
  - 機械学習を利用　⇒　データが存在するところにどこでも適用可能
- 実践から見た機械学習
  - あらゆる産業分野での実用化が進む
  - 既存手法（人を含む）を凌駕する性能を達成、新しい産業へ

Preferred Infrastructure

補足資料

会社紹介

会社紹介
株式会社 Preferred Infrastructure

- 略称　　PFI
- 代表者　西川 徹
- 設立　　2006年3月
- 社員数　26名
- 所在地　〒113-0033 東京都文京区本郷2-40-1
- 事業概要
  - データ解析分野での製品開発, 販売, サービス提供
  - 様々な分野においての共同研究開発、共同事業

最先端の技術を最短路で実用化

リサーチとエンジニアリングの融合

世の中に必要とされる中で特に重要で困難な課題に対し解を提供

Preferred Infrastructure
メンバー構成

- フルタイム26人中23人がエンジニア/研究者
  - 以下の情報/理/工学博士
    - 自然言語処理/機械学習/計算量理論/データマイニング/文字列解析
  - ICPCプログラミングコンテスト 世界大会（=日本代表） 7名
  - 未踏プロジェクト 5名
  - TopCoder世界上位や、世界プログラミングコンテスト優勝者など
- 各種コミュニティへの働きかけ
  - 日本Hadoopユーザー会立ち上げ, 自然言語処理若手の会委員長
  - 日本語入力本, Haskel本, 各種雑誌記事, 専門書

その他、データ圧縮、UI/UX、セキュリティ、分散システム、
ソフトウェア工学など様々な分野の専門家

単純ベイズ法（Naïve Bayes）

- $p(y \mid x) \propto p(y)\, p(x \mid y)$ ベイズの定理より
  $\propto p(y)\, \Pi_i\, p(x_i \mid y)^{c(x_i)}$ 各特徴の生成確率が独立という仮定
- $\log p(y \mid x) \propto \underbrace{\log p(y)}_{b} + \Sigma_i\, c(x_i)\, \underbrace{\log p(x_i \mid y)}_{w_i}$ 両辺の対数をとる
- $\log p(y \mid x) = w^T x + b$
  - w_iとbはデータからの最尤推定で以下のように求められる
  - $b := C(y) / \Sigma_{y'} C(y')$
  - $w_i := C(x_i, y) / C(y)$
- 頻度はデータを操作しながら更新していけば良い