2011/11/11 IBIS 2011:オンライン予測セッション

# オンライン凸最適化と線形識別モデル学習の最前線

株式会社Preferred Infrastructure
岡野原　大輔

# 背景：大規模、リアルタイムな分析の需要の高まり

- 高次元で疎な入力を扱う
  - 例：言語処理の例では、全単語の３つまでの組み合わせ～数百億次元、非零要素数は平均数百～数万
- 高スループット、低レイテンシな分析が求められる
  - SNS分析、アルゴリズムトレード、広告配信など
  - 例: 文書分類の場合、秒間1万文書程度が目標（SNSなど）

- データは大量、モデルの種類数も膨大
  - 全て集めてから処理というのは困難
  - ユーザー毎に異なるモデルを利用⇒分類器が数百万

# 背景：手法の発展

- 線形識別モデルとオンライン凸最適化による学習により、大量のデータを利用した高速な学習と識別が可能となっている
  - マシン１台でも秒間数万～数百万サンプルを処理可能
  - サンプル数に対してlinear、全特徴発火数に対しsublinear時間での学習
  - 自然言語処理、画像解析、行動分析など様々な分野で利用
- モデル、学習アルゴリズムは毎年進化している
  - 高精度、高い学習効率、コンパクトなモデル、高ノイズ耐性
- アルゴリズムの性能解析も進化
  - i.i.dを仮定しないRegret解析

# 宣伝：Jubatus

- NTT PF研とPreferred Infrastructureによる共同開発
  OSSで公開 http://jubat.us/
- 今回解説するオンライン学習モデルの多くがは分散並列化された上で
  サポートされている


リアルタイム
ストリーム

分散並列

深い解析



Preferred Infrastructure

# 今日の話

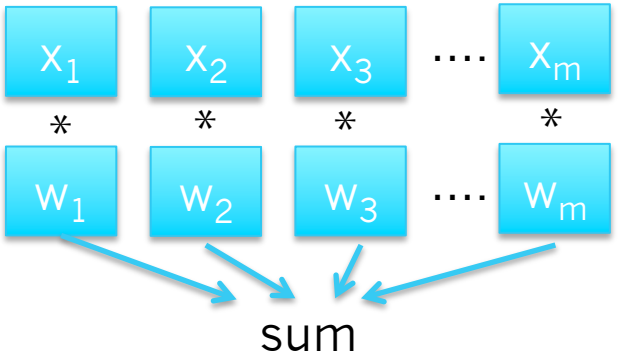
- 導入：線形識別モデル
- 学習手法
  - Perceptron, PA, CW, AROW, NHERD
- 凸コスト関数に対するオンライン学習のRegret解析
- SIMBA：sublinear learningでのSVMの学習

# 線形識別モデル

# 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できる.
  - 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動く
  - c.f. CRFs, 構造学習

# 学習 = 重みベクトルwの推定 = 凸最適化問題

訓練例$\{(x^{(i)}, y^{(i)})\}$ $(i=1 \ldots n)$ を利用して**w**を推定する

$$\mathbf{w}^* := \mathrm{argmin}_{\mathbf{w}} \Sigma_i L(x^{(i)}, y^{(i)}, \mathbf{w}) + C\, R(\mathbf{w})$$

- $L(\mathbf{x}, y, \mathbf{w})$：損失関数
  - 訓練例を正しく識別できているか
  - hinge-loss (SVM): $L(\mathbf{x}, y, \mathbf{w}) = [1 - y\mathbf{w}^T\mathbf{x}]$
  - log-loss (logistic回帰): $\log(1 + \exp(-y\mathbf{w}^T\mathbf{x}))$
- $R(\mathbf{w})$：正則化項

  **w**に対する事前知識．**w**がどのような形をとるかを決める
  - $C > 0$ 損失関数と正則化項の間のトレードオフパラメータ

L, Rが共に凸関数ならば、この問題は凸最適化問題で効率的に解ける

# 線形識別モデルの
# オンライン学習

# オンライン学習

- 各訓練例$(x^{(i)},y^{(i)})$に対して予測を行い、その結果に基づき、
  パラメータを即時更新する

- オンライン学習の特徴
  - 学習例が冗長な場合に収束が速い.
  - 訓練例を保存する必要は無く、その場で捨てられる
  - 実装は簡単な場合が多い
  - 凸最適化によるバッチ学習は確率的最急降下法(SGD)でオンライン化可能
    - その場合のアルゴリズムの性能はRegretで評価できる（後述）

# 学習アルゴリズムの基本形

- これから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 $y\mathbf{w}_i^T\mathbf{x} > E$ ?
3. 正しく分類できないのであれば$\mathbf{w}_i$を次のように更新

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

   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）

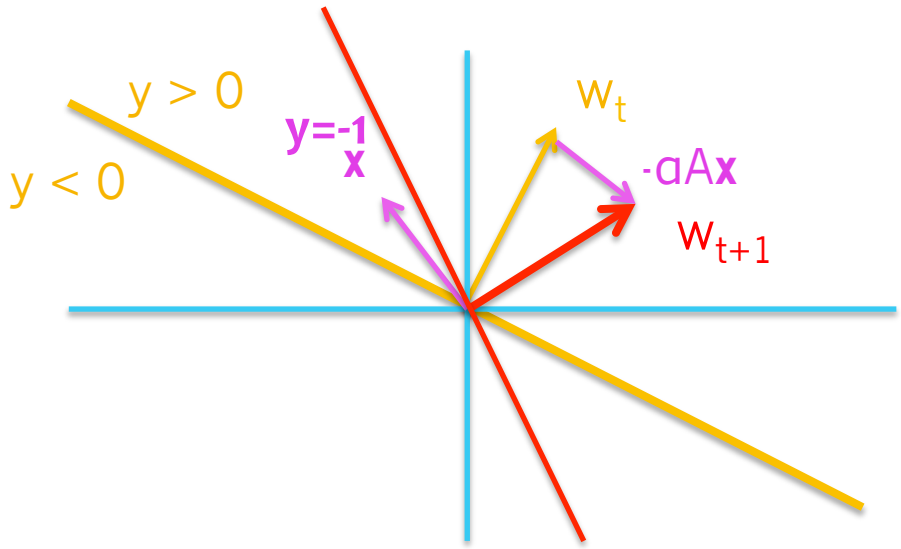
- E, α, Aをいかに適切に設定するかで、学習性能が変わってくる
  - E: 更新条件
  - α : ステップ幅
  - A : 各特徴毎の更新幅． c.f. マハラノビス距離

# 更新の意味

- $\alpha > 0$、Aが半正定値行列の時

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

$\alpha\mathbf{x}^{T}A\mathbf{x}$：常に正




- 今の訓練例は少なくとも正しく分類されるように更新

# 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

登場した年

1958
2002
2006
2009
2010

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$\mathbf{w}_i$で分類できるかを調べ、誤って分類した場合は $\mathbf{w}_{i+1} := \mathbf{w}_i + y\mathbf{x}$ と更新
  - $E = 0, \alpha = 1, A = I$ に対応
- 単純な更新式だが、自然言語処理など多くのタスクで強力である
  - 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）の
  - 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 $\{(x^{(i)}, y^{(i)})\}_{i=1\ldots N}$, $|x^{(i)}|_2<R$, がある重み$\mathbf{u}$でマージン$\gamma$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、

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

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]

- SVMのオンライン版
  - 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**w**を探す
  - 現在の訓練例(**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、$L(\mathbf{x}, y, \mathbf{w}) = [1 - y\mathbf{w}^T\mathbf{x}]$　(hinge–loss)

- この問題は閉じた式で得られる

# Passive Aggressive (続)

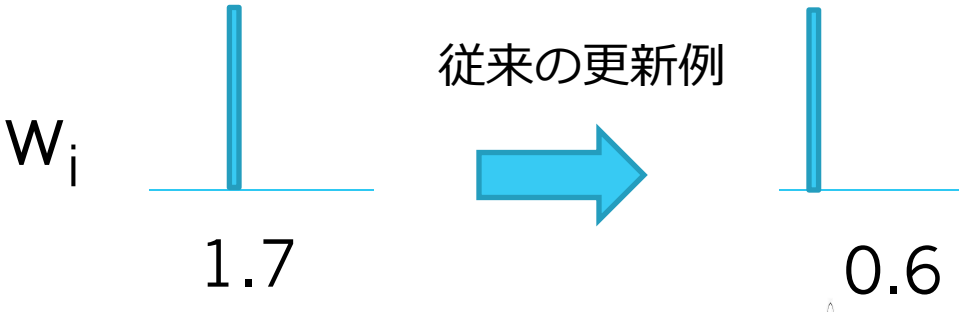
$\mathbf{w}_{i+1} := \mathbf{w}_i + y\, l(\mathbf{x}, y, \mathbf{w})/(|\mathbf{x}|^2 + 1/C)\ \mathbf{x}$

- 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（$\mu, \Sigma$)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）
- $\arg\min_{\mu, \Sigma} D_{KL}(N(\boldsymbol{\mu}, \Sigma) \,||\, N(\boldsymbol{\mu}_i, \Sigma_i))$ s.t. $\Pr_{\mathbf{w}\sim N(\mu,\Sigma)}[y\mathbf{w}_i^T\mathbf{x} \geqq 0] \geqq \eta$
- この最適化問題は閉じた解を持つ
  - E, α, Aを$\mathbf{x}$, y, $\boldsymbol{\mu}$, 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

| 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フォルダ
EnronのUser Bの上位10フォルダ
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

CWでは1が常に最優先

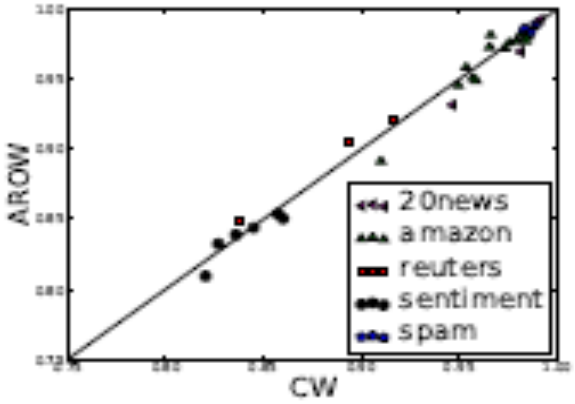
$$\arg\min_{\mu,\Sigma} D_{KL}(N(\mu,\Sigma) \,||\, N(\mu_i,\Sigma_i)) + \lambda_1 L(x, y, \mu) + \lambda_2 x^T \Sigma x$$

条件2　　条件1　　条件3

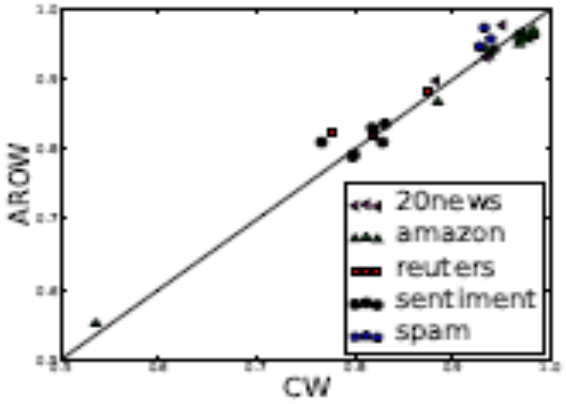
- E, α, Aは閉じた式で求めることができる

Preferred Infrastructure

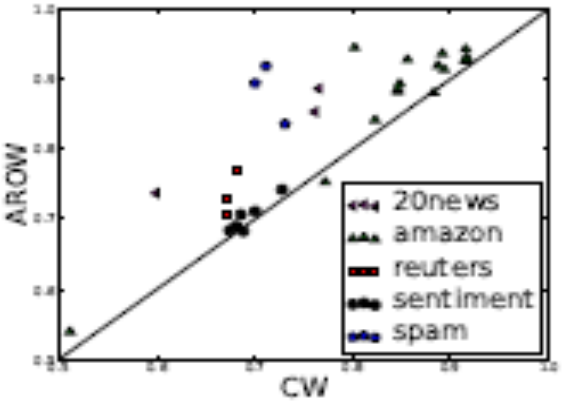
# AROWの実験結果




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

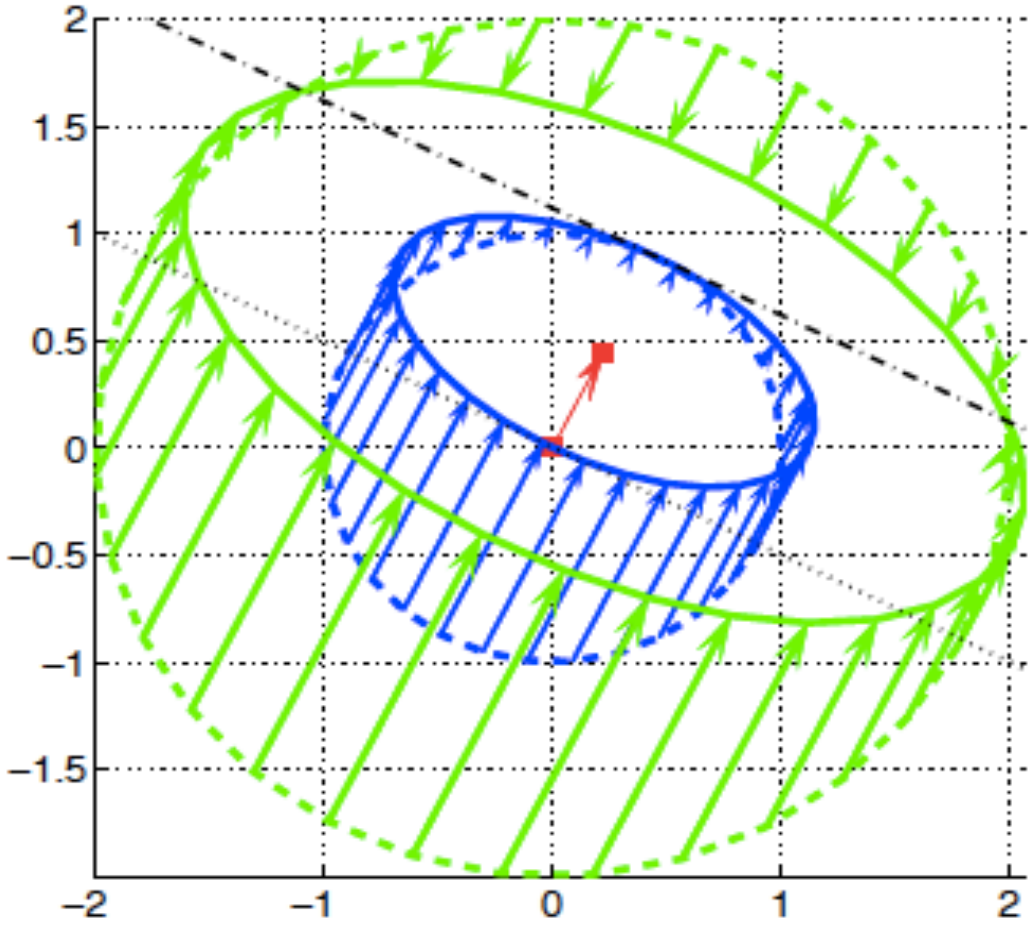
# NHERDの更新例

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

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

# 線形学習のまとめ

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

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

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

いずれのアルゴリズムも更新時間は$O(|\mathbf{x}|_o)$

Preferred Infrastructure

# オンライン凸最適化の Regret解析

# 一般の損失関数の場合

- 先ほど紹介したのは分類専門のモデル
  - 分類さえできれば良い．特にモデルは仮定していない
- 損失関数を利用した学習については確率的勾配降下法(SGD)を利用してオンライン化可能

  $\mathbf{w} := \mathbf{w} - \tau \partial L(\mathbf{x}, y, \mathbf{w})$

  - 正則化項がある場合でもFOBOS (FOrward Backward Splitting) [Duchi+ 09]などを利用して更新式を少し変更して組み込むことが可能（付録資料）

- 損失関数の性能はRegret解析で統一的に調べられる

# Regret解析

- アルゴリズムAが$\mathbf{x}_t$を選択した後、$\mathbf{f}_t$が明かされコスト$\mathbf{f}_t(\mathbf{x}_t)$が発生
- $\mathrm{Regret}_T(A) := \sum_{i=1\ldots T} \mathbf{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}} \sum_{i=1\ldots T} \mathbf{f}_t(\mathbf{x})$
  - 最適点（全tで同じ）を常に選んでいた場合と比べて、アルゴリズムAが選んだ点によるコストはどれだけ多く発生したか？
  - $\mathrm{Regret}_T(A) = o(T)$であるならば、Tを増やすほど最適点との1回あたりの差は小さくなっていき、最適点に近づく ($\mathrm{Regret}_T(A) / T \to 0$)

- オンラインアルゴリズムの性能解析手法としてRegret解析は強力
  - **f**についてi.i.dなどの仮定はいらない
    - Aの挙動を見てから、コストが大きくなるように$f_i$を選んでも良い
  - 今回紹介する解析は確率的な議論ではなく常に成り立つ

# 教師有学習におけるRegret解析

- アルゴリズムが選ぶパラメータはモデルパラメータ**w**に対応
- 訓練例 (x, y)に対する損失をコスト関数とする$\mathbf{f}_t(\mathbf{w}) := L(x^{(t)}, y^{(t)}, \mathbf{w})$
- 正則化はパラメータの値域Kで表現
  - $|\mathbf{w}|_1 < C, |\mathbf{w}|_2 < C$
- Regretが小さい = テストデータを正しく予測できている

$$\sum_{i=1\ldots T} L(x^{(t)}, y^{(t)}, \mathbf{w}_t) - \min_{\mathbf{w}} \sum_{i=1\ldots T} L(x^{(t)}, y^{(t)}, \mathbf{w}) \quad \text{s.t.} \quad \mathbf{w} \in K$$

* Regret解析の場合のパラメータは**x**であることに注意

# 一般の線形コスト関数の場合
# Regularized Follow The Leader (RTFL) :

$$\mathbf{x}_{t+1} = \mathrm{argmin}_{x \in K} \left( \eta \sum_{i=1 \ldots t} \mathbf{f}_i^T \mathbf{x} + R(\mathbf{x}) \right)$$ RTFL

- 但し、$R(\mathbf{x})$は強凸、$\eta > 0$, $K$は凸な集合であるとする
- 今までのコスト + 正則化項が最小である点を次の予想に使う
  - 次のコスト関数はこれまでのコスト関数とは関係無いことに注意
  - 正則化項が無いと$\mathrm{Regret} = O(T)$となる
- この時、$\mathrm{Regret}_T(\mathrm{RTFL}) = O(T^{1/2})$
- RTFLは様々なアルゴリズムの一般化 [Hazan 11]
  - Kがsimplex上の点、$R(\mathbf{x}) = \mathbf{x}\log\mathbf{x}$ の時 multiplicative updateに対応
  - Kが単位円、$R(\mathbf{x}) = |\mathbf{x}|_2^2$の時、RTFLは最急降下法に対応
  - 過去のコスト関数を覚えておく必要はない

# コスト関数が強凸である場合
## OGD : Online Gradient Descent

$$\mathbf{y}_t = \mathbf{x}_{t-1} - \eta_{t-1} \nabla \mathbf{f}_{t-1}(\mathbf{x}_{t-1})$$
$$\mathbf{x}_t = \mathrm{argmin}_{x \in K} |\mathbf{x} - \mathbf{y}_t|_2$$

OGD

- コスト関数$f_t(x)$が強凸である場合は更にRegret boundを強くできる
  - $f(x)$が$\alpha$強凸 $\Leftrightarrow$ $f''(x) \geqq \alpha$ for all $x$
- この時、$\mathrm{Regret}_T(\mathrm{OGD}) = O(\log T)$ [Hazan+ ML 07]
  - 線形コスト関数に対するRTFLの$O(T^{1/2})$に比べて$O(\log T)$は遙かに小さい
  - 証明は初等的でシンプル（次項）

# OGDのRegretがO(logT)であることの証明 (1/2)

$\nabla_t := \nabla f_t(x_t)$

$x^* := \mathrm{argmin}_x \sum_{i=1\ldots T} f_t(x)$

$d_t := x_t - x^*$

- fがα強凸であることをふまえて、$x_t$の周りでfオイラー展開

$$f_t(x^*) \geqq f_t(x_t) + \nabla_t^T d_t + (\alpha/2)\, d_t^2$$

$$2(f_t(x_t) - f_t(x^*)) \leqq -2\nabla_t^T d_t - \alpha\, d_t^2$$

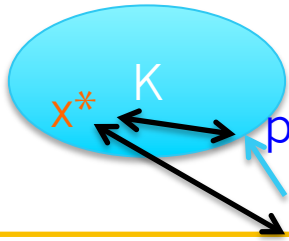
- $\nabla_t^T d_t$ を上から抑えるために、$|d_{t+1}|^2=|x_{t+1}-x^*|^2$について調べる

$$|d_{t+1}|^2 = |\mathrm{proj}(x_t - \eta_t \nabla_t) - x^*|^2$$
$$\leqq |x_t - \eta_t \nabla_t - x^*|^2$$
$$= d_t^2 - 2\eta_t \nabla_t^T d_t + \eta_t^2 |\nabla_t|^2$$
$$2\nabla_t^T d_t = (d_t^2 - d_{t+1}^2)/\eta_t + \eta_t |\nabla_t|^2$$

$\mathrm{proj}(y) := \arg\min_{x \in K} |x - y|_2$

$x^* \in K$なので

$|\mathrm{proj}(y) - x^*| \leqq |y - x^*|$



# OGDのRegretがO(logT)であることの証明 (2/2)

- $\nabla_t^T d_t$ を代入して、t=1…nについて両辺和をとり、$\eta_t = 1/\alpha t$ とおく.
  また、$G = \max_t |\nabla_t|^2$ とおく

$$2\sum_{t=1}^{n} f_t(x_t) - f_t(x^*) \leqq \sum_{t=1}^{n} d_t^2 (1/\eta_{t+1} - 1/\eta_t - \alpha) + |\nabla_t|^2 \sum_{t=1}^{n} \eta_t$$
$$\leqq G^2 \sum_{t=1}^{n} (1/\alpha t)$$
$$\leqq G^2/\alpha \, (1 + \log T)$$

# 凸最適化の解析について

- コスト関数については強凸以外に仮定はおいていないことに注意
  - 途中で変えても良い。毎回ランダムな凸関数を使っても良い
- 確率的な解析ではなく常に成り立つ
  - 質問：悪意があるクエリに対してもなぜ成り立つか

    ⇒悪意があるクエリはこれまでのコスト関数の傾向とは離れている.

    最適値$x^*$のコスト和もどんどん悪くなる⇒差は離れない

# 線形より高速な学習

# 線形時間より高速な学習は可能？

- 学習は、入力に必要なO(nm)時間より速くすることは困難なように思える
  - n：学習サンプル数
  - m：1サンプルあたりの平均非零特徴数

- 任意のサンプル、特徴を自由にアクセスできる環境では線形時間より計算量を改善できるか？
  ⇒できる

# SIMBA [Hazan+ NIPS 11]
## (Sublinear IMportance-sampling Bi-stochastic Algorithm)

- SVMをsub-linear時間で学習する
  - 訓練例数をn, 特徴種類数をmとした時、ε近似の学習時間は$O((n+m)/\varepsilon^2)$
  - 実用的にも、高速なSVM学習器であるPegasos [S. Shwartz+ ICML 2007] と比べて特徴アクセス数にして100倍近く高速に求めることができる
  - オンラインではない（データを重み付きサンプリング）
- primal-dual法を利用
  - importance-weighted samplingを利用し、怪しそうなサンプルを優先的に学習する.
  - どのサンプルが怪しいかどうかは、特徴側を利用してサンプル
  - 主変数側、双対変数側の両方で確率的勾配法で最適化を行う

# SIMBA（概要）

- 次の鞍点問題に対するprimal-dual法を利用 [Clarkson+ FOCS 10]

  $\max_{z \in K} \min_{i \in [1,n]} c_i(z)$

  - SVMの場合、$K = z \in c_i(z) = y_i(\mathbf{w}^T\mathbf{x}_i + b) + \xi_i$
- この問題を次のように変形

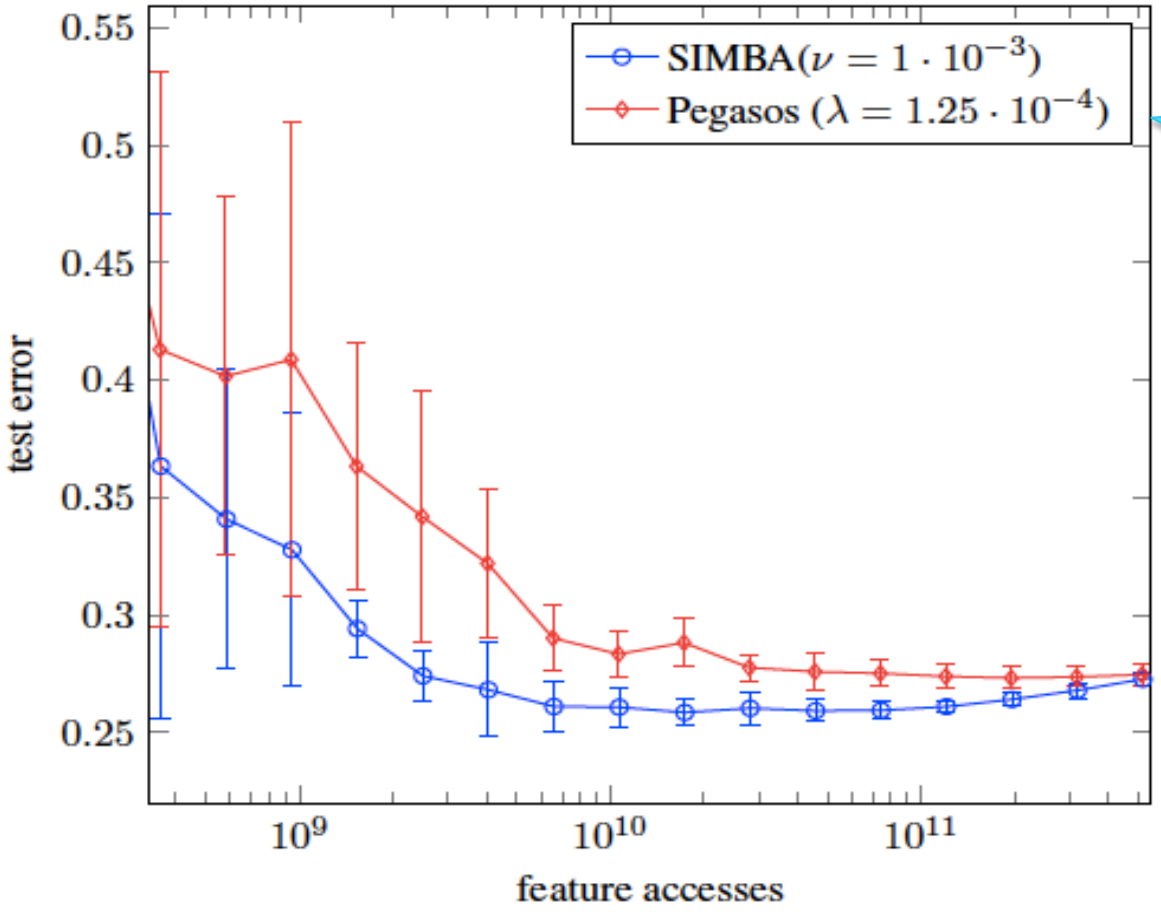
  $\max_{z \in K} \min_{p \in \Delta n} p_i c_i(z)$

  - 主変数 z、双対変数 p これらを確率的に更新
- 主変数を更新
  - $i \in [1, n]$をpに従って選び、$c_i(z)$を用いて勾配を求めzを更新 $O(d)$ time
- 双対変数を更新
  - $c_i(z)$が小さい$p_i$が大きくなるようにpを更新 $O(n)$ time

- 上記問題のε近似は確率1/2以上で$O(\varepsilon^{-2}(n + m))$時間で得られる

Preferred Infrastructure

# PegasosとSIMBAの比較




Pegasos:
代表的なSVMの
高速学習手法

News20の分類問題： 特徴発火数1,355,191、 訓練例8000
特徴へのアクセス回数で比較（注：横軸は対数）
最適値へPegasosの1/10で収束

Preferred Infrastructure

# まとめ

- 線形識別器の学習アルゴリズムは毎年着実に進展している
  - 同じ計算量で、より高精度な分類、データの変化に追従.
- 性能解析としてRegret解析が有効
  - 問題設定に仮定は少ない
  - 悪意のある問題設定、データの変化に対しても追従可能

- 今回扱えなかったオンライン最適化のトピック
  - 並列分散環境の場合
  - 線形識別器以外の場合（行列分解）
  - 正則化付の場合（特にnon-smoothな$L_1$, $L_0$, $L_\infty$）

# 付録資料

# 正則化

# 正則化

- 正則化の方法で、重みベクトルがどのような形をとるかが変わる
- L2-norm $|w|_2^2 = \sum_i w_i^2$
  - 大きすぎる重みにペナルティがかかり、小さくなるとペナルティは無視できるほど小さくなる
- L1-norm $|w|_1 = \sum_i |w_i|$
  - 多くの重みが0になるようにペナルティがかかる。
- L∞-norm $|w|_\infty = \max_i \{|w_i|\}$
  - 全ての重みが等しくなるようなペナルティがかかる

- これらの重みをグループ毎にかけることで、グループ毎に異なる性質をもたせることができる
  - Group Lassoなど

# FOBOS (1/4)
# (FOrward Backward Splitting) [Duchi+ 09]

- f(**w**) + r(**w**)を最小にする**w**をSGDで求める
  - f(**w**)は微分可能　（例：損失関数の和）
  - r(**w**)は微分不可能（例: $|\mathbf{w}|_1$）
- 最初にf(**w**)のみを最小化し、その結果に近いもので r(**w**)を最小化する**w**を求める

$$\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \eta_t \boldsymbol{g}_t^f$$

f(**w**)のみでの勾配降下法 ($g_t^f = \partial f(\mathbf{w}_t)$)

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \frac{1}{2} \left\| \boldsymbol{w} - \boldsymbol{w}_{t+\frac{1}{2}} \right\|^2 + \eta_{t+\frac{1}{2}} r(\boldsymbol{w}) \right\}$$

Preferred Infrastructure

# FOBOS (2/4)

- 先ほどの二番目のステップは
  単純な閉じた解が得られる場合が多い
- 例1．$r(\mathbf{w}) = |\mathbf{w}|_1$の場合

$$w_{t+1,j} = \mathrm{sign}\left(w_{t+\frac{1}{2},j}\right)\left[|w_{t+\frac{1}{2},j}| - \tilde{\lambda}\right]_+$$

- 例2. $r(\mathbf{w}) = |\mathbf{w}|_2^2$の場合

$$w_{t+1} = w_{t+1/2}\,/(1+\lambda)$$

$\mathbf{w}_{t+1/2} = (0.5, -1.0, 2.0, -0.7, 1.4)$、$\lambda=1$ の時
$\mathbf{w}_{t+1} = (\ 0,\ \ 0, 1.0,\ \ 0,\ 0.4)$

# FOBOS (3/4)

- 例3. Berhu正則化（0からγまで$L_1$, γから$L_2$)

$$r(\boldsymbol{w}) = \lambda \sum_{j=1}^{n} b(w_j) = \lambda \sum_{j=1}^{n} \left[ |w_j| \, [\![ |w_j| \leq \gamma ]\!] + \frac{w_j^2 + \gamma^2}{2\gamma} [\![ |w_j| > \gamma ]\!] \right]$$

$$w_{t+1,j} = \begin{cases} 0 & \left| w_{t+\frac{1}{2},j} \right| \leq \tilde{\lambda} \\ \mathrm{sign}(w_{t+\frac{1}{2},j}) \left[ |w_{t+\frac{1}{2},j}| - \tilde{\lambda} \right] & \tilde{\lambda} < \left| w_{t+\frac{1}{2},j} \right| \leq \tilde{\lambda} + \gamma \\ \frac{w_{t+\frac{1}{2},j}}{1+\tilde{\lambda}/\gamma} & \gamma + \tilde{\lambda} < \left| w_{t+\frac{1}{2},j} \right| \end{cases}$$

- 複雑な正則化も混ぜたり自由にできる
  - $L_1$と$L_2$と$L_\infty$を同時に与える

# FOBOS (4/4)

- バッチ学習と殆ど同じ結果が得られる
  - 同時期に提案された似た手法も同様の結果
  - 理論的にも近い結果が得られることが保障
- 実装は非常に簡単
  - 既存のオンライン学習に１行加えるだけ
  - 「$L_1$は引く、$L_2$は割る」

# Efficient Sparse Modeling with Automatic Feature Grouping [Zhong+ ICML 2011]

- $L_1$正則化は、全ての特汎化性能を失う場合が多い
  - 冗長な特徴$f_1$と$f_2$がある場合、$L_1$正則化は$f_1$側にのみ重みを与えて、$f_2$側にoを与えてしまう。$f_1$と$f_2$に等しく重みを与えたい
- 特徴のグループを見つけること自体が大変
  - 特徴にあらかじめ構造があるのであればGroup Lassoなどで、正則化することができる
- 全ての特徴ペアに対してL∞正則化を適用する c.f. OSCAR
  - $|\mathbf{w}|_1 + \lambda \sum_{i<j} \max(|w_i|, |w_j|)$
  - 徴ペアは重みが等しくなろうとする

- ペア数は特徴数がmの時$O(m^2)$であり、計算量がそのままだと大きいが、$O(m \log m)$時間で求められるアルゴリズムを提案

Preferred Infrastructure

# 出典


- [Collins EMNLP 02] “Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. Michael Collins, EMNLP 2002,
- [Crammer, JMLR 06] “Online Passive-Aggressive Algorithms”, Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer, Journal of Machine Learning, 2006
- [Dredze+, ICML 08] “Confidence-Weighted Linear Classification”, Mark Dredze, Koby Crammer and Fernando Pereira , ICML 2008
- [Crammer+ NIPS 08] “Exact Convex Confidence-Weighted Learning”, Koby Crammer, Mark Dredze and Fernando Pereira, NIPS 2008

- [Duchi+ 09] “Online and Batch Learning using Forward Backward Splitting”, John Duchi and Yoram Singer, JMLR
- [S. S.-Shwartz 07] “Online Learning: Theory, Algorithms, and Applications”, S. Shalev-Shwartz, Ph. D thesis 2007
- [Hazan+ ML 07] “Logarithmic regret algorithms for online convex optimization”, E. Hazan, A. Agarwal and S. Kale. Machine Learning 2007
- [Hazan+ 11] “The convex optimization approach to regret minimization”, http://ie.technion.ac.il/~ehazan/papers/opt_book.pdf

- [Clarkson, FOCS 10] “Sublinear optimization for machine learning”, K. L. Clarkson, E. Hazan and D. P. Woodruff, FOCS 2010
- [Shalev+ ICML 07] “Pegasos: Primal estimated sub-gradient solver for SVM”,S. Shalev-Shwartz and N. Srebro, ICML 2007
- [Hazan+ NIPS 11] “Beating SGD: Learning SVMs in Sublinear Time”, E. Hazan, T. Koren, and N. Srebro, NIPS 2011