NANYANG
TECHNOLOGICAL
UNIVERSITY

2013 SIAM International Conference on DATA MINING (SDM13)

May 2-4, 2013

Sheraton Austin Hotel at the Capitol
Austin, Texas, USA

SDM2013 TUTORIAL:

Online Learning for Big Data Mining:

Methods and Applications

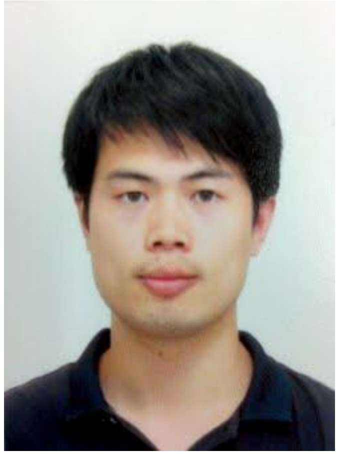
Steven C.H. Hoi

School of Computer Engineering
Nanyang Technological University
Singapore

4 May, 2013

http://LIBOL.stevenhoi.org/

Acknowledgements

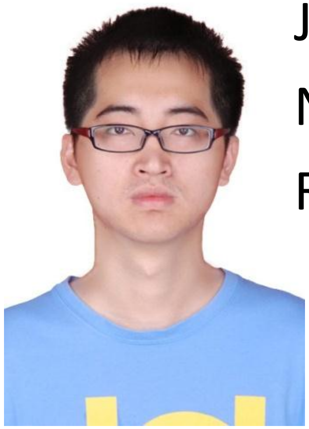


Peilin Zhao

NTU

PhD

Student



Jialei Wang

NTU

RA

Rong Jin

MSU

Coauthor

Bin Li, NTU,

PhD Student

Hao Xia, NTU,

PhD Student

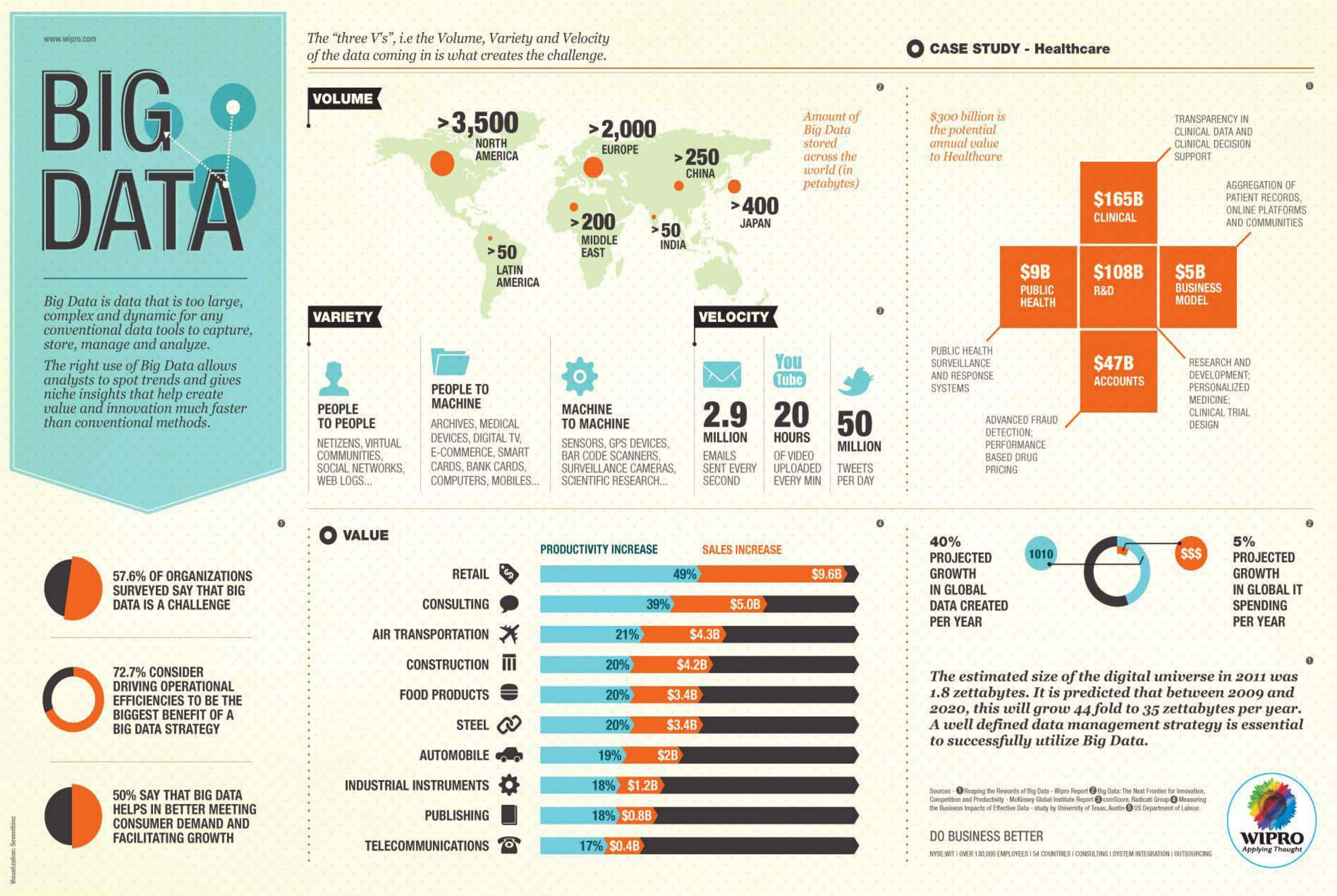
Pengcheng Wu, NTU,
PhD Student

Dayong Wang, NTU,

PhD Student

Agenda

- Introduction (25 min)
  - Big Data Mining: Opportunities & Challenges
  - Online Learning: What, Why, Where
- Online Learning
  - Overview (5 min)
  - Traditional Linear Online Learning (30 min)
  - Non-traditional Linear Online Learning (30 min)
  - Kernel-based Online Learning (30 min)
  - Online Multiple Kernel Learning (30 min)
- Discussions + Q&A (30 min)



www.wipro.com

BIG DATA

Big Data is data that is too large, complex and dynamic for any conventional data tools to capture, store, manage and analyze.

The right use of Big Data allows analysts to spot trends and gives niche insights that help create value and innovation much faster than conventional methods.

The "three V's", i.e the Volume, Variety and Velocity of the data coming in is what creates the challenge.

VOLUME

Amount of Big Data stored across the world (in petabytes) ❷

VARIETY

PEOPLE TO PEOPLE
NETIZENS, VIRTUAL COMMUNITIES, SOCIAL NETWORKS, WEB LOGS...

PEOPLE TO MACHINE
ARCHIVES, MEDICAL DEVICES, DIGITAL TV, E-COMMERCE, SMART CARDS, BANK CARDS, COMPUTERS, MOBILES...

MACHINE TO MACHINE
SENSORS, GPS DEVICES, BAR CODE SCANNERS, SURVEILLANCE CAMERAS, SCIENTIFIC RESEARCH...

VELOCITY ❸

2.9 MILLION EMAILS SENT EVERY SECOND

You Tube **20 HOURS** OF VIDEO UPLOADED EVERY MIN

50 MILLION TWEETS PER DAY

CASE STUDY - Healthcare ❺

$300 billion is the potential annual value to Healthcare

$165B CLINICAL – TRANSPARENCY IN CLINICAL DATA AND CLINICAL DECISION SUPPORT

$5B BUSINESS MODEL – AGGREGATION OF PATIENT RECORDS, ONLINE PLATFORMS AND COMMUNITIES

$9B PUBLIC HEALTH – PUBLIC HEALTH SURVEILLANCE AND RESPONSE SYSTEMS

$108B R&D – RESEARCH AND DEVELOPMENT; PERSONALIZED MEDICINE; CLINICAL TRIAL DESIGN

$47B ACCOUNTS – ADVANCED FRAUD DETECTION; PERFORMANCE BASED DRUG PRICING

❶

57.6% OF ORGANIZATIONS SURVEYED SAY THAT BIG DATA IS A CHALLENGE

72.7% CONSIDER DRIVING OPERATIONAL EFFICIENCIES TO BE THE BIGGEST BENEFIT OF A BIG DATA STRATEGY

50% SAY THAT BIG DATA HELPS IN BETTER MEETING CONSUMER DEMAND AND FACILITATING GROWTH

VALUE ❹

	PRODUCTIVITY INCREASE	SALES INCREASE
RETAIL	49%	$9.6B
CONSULTING	39%	$5.0B
AIR TRANSPORTATION	21%	$4.3B
CONSTRUCTION	20%	$4.2B
FOOD PRODUCTS	20%	$3.4B
STEEL	20%	$3.4B
AUTOMOBILE	19%	$2B
INDUSTRIAL INSTRUMENTS	18%	$1.2B
PUBLISHING	18%	$0.8B
TELECOMMUNICATIONS	17%	$0.4B

❷

40% PROJECTED GROWTH IN GLOBAL DATA CREATED PER YEAR

5% PROJECTED GROWTH IN GLOBAL IT SPENDING PER YEAR

❶

The estimated size of the digital universe in 2011 was 1.8 zettabytes. It is predicted that between 2009 and 2020, this will grow 44 fold to 35 zettabytes per year. A well defined data management strategy is essential to successfully utilize Big Data.

Sources - ❶ Reaping the Rewards of Big Data - Wipro Report ❷ Big Data: The Next Frontier for Innovation, Competition and Productivity - McKinsey Global Institute Report ❸ comScore, Radicati Group ❹ Measuring the Business Impacts of Effective Data - study by University of Texas, Austin ❺ US Department of Labour.

DO BUSINESS BETTER

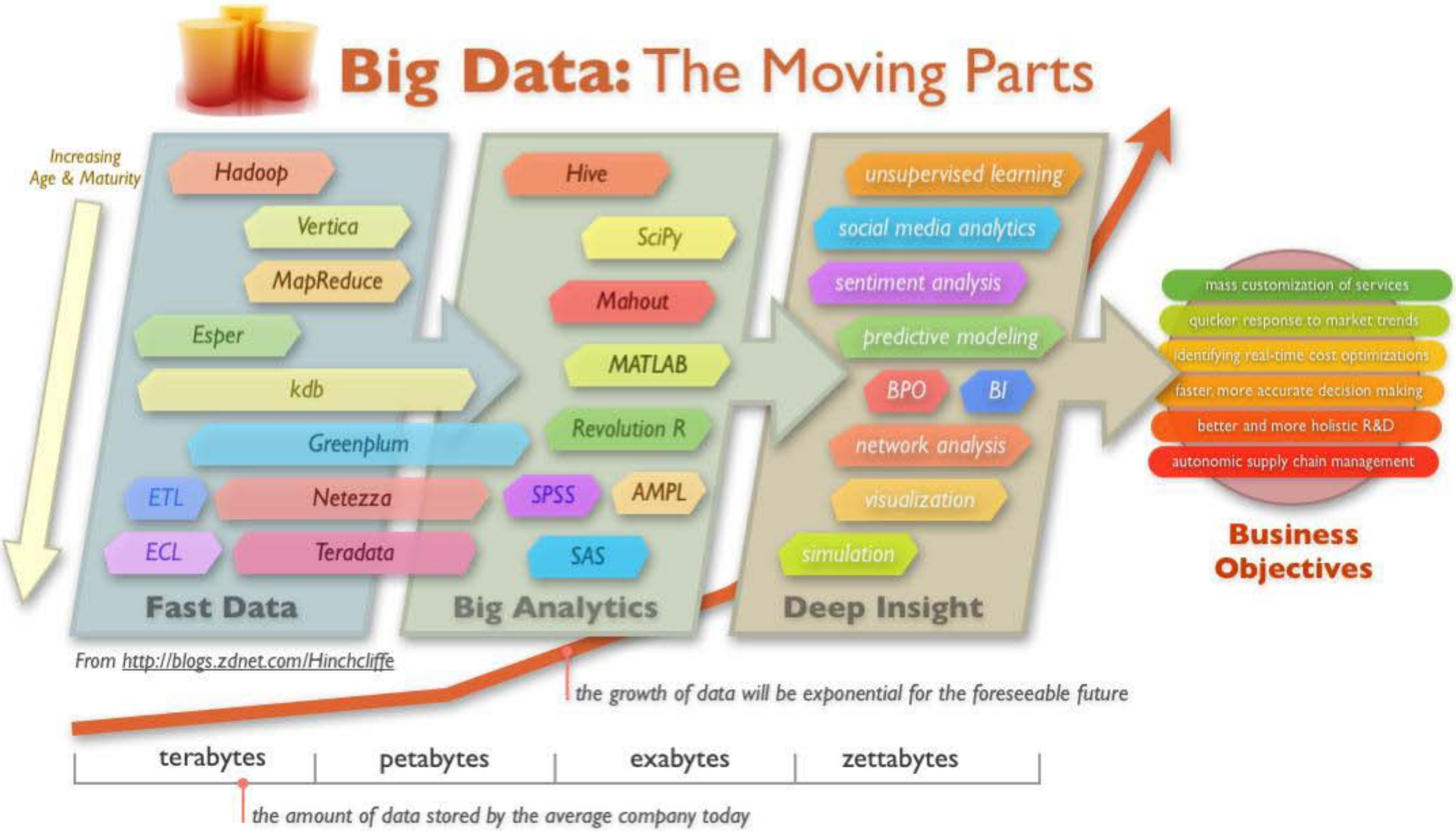
NYSE:WIT | OVER 130,000 EMPLOYEES | 54 COUNTRIES | CONSULTING | SYSTEM INTEGRATION | OUTSOURCING

WIPRO Applying Thought

Visualization: Seventhinc

From http://visual.ly/big-data

Big Data Mining: Opportunities

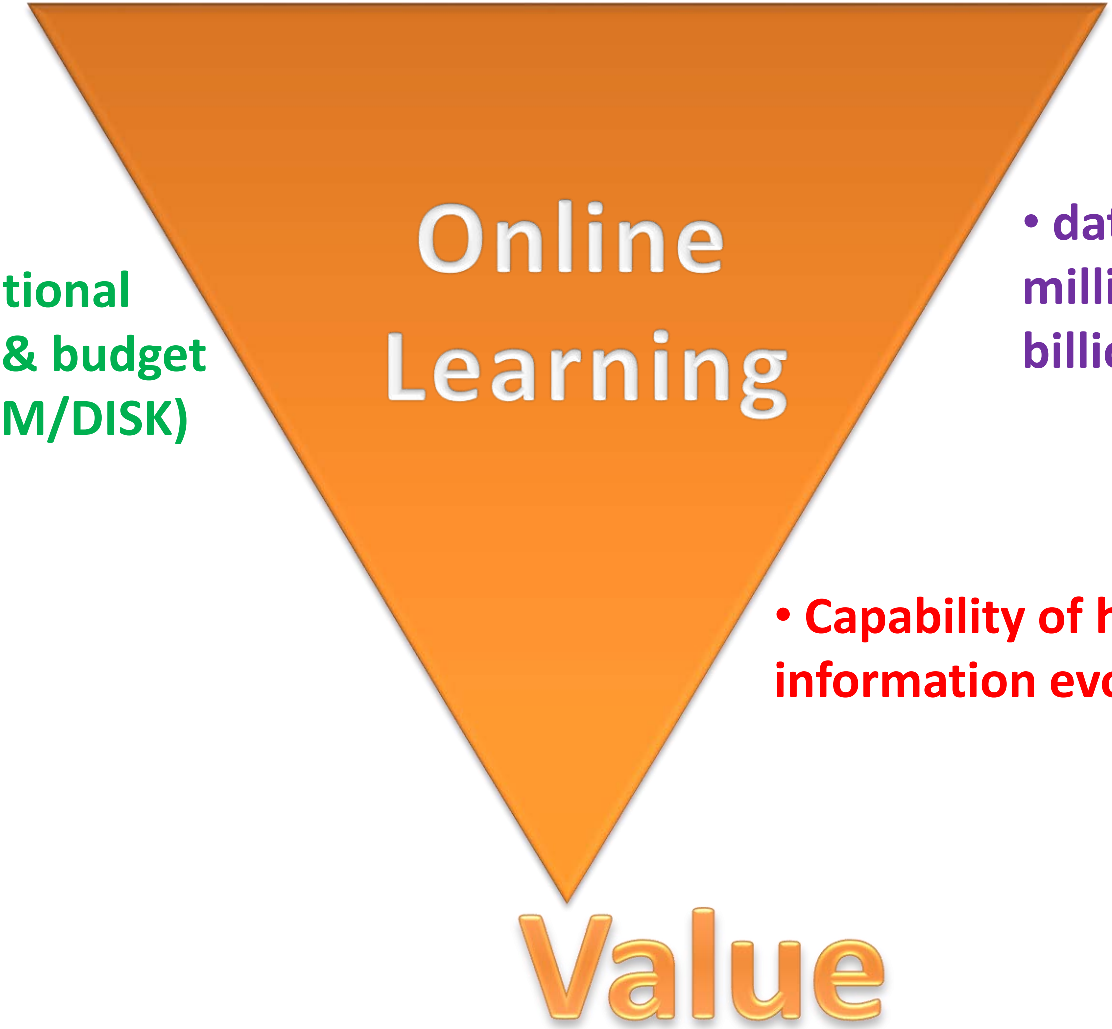


From http://blogs.zdnet.com/Hinchcliffe

the growth of data will be exponential for the foreseeable future

the amount of data stored by the average company today

Big Data Mining: Challenges



- Limited computational capacity & budget (CPU/RAM/DISK)

- data size in millions or even billions scale

- Capability of handling diverse information evolving dynamically

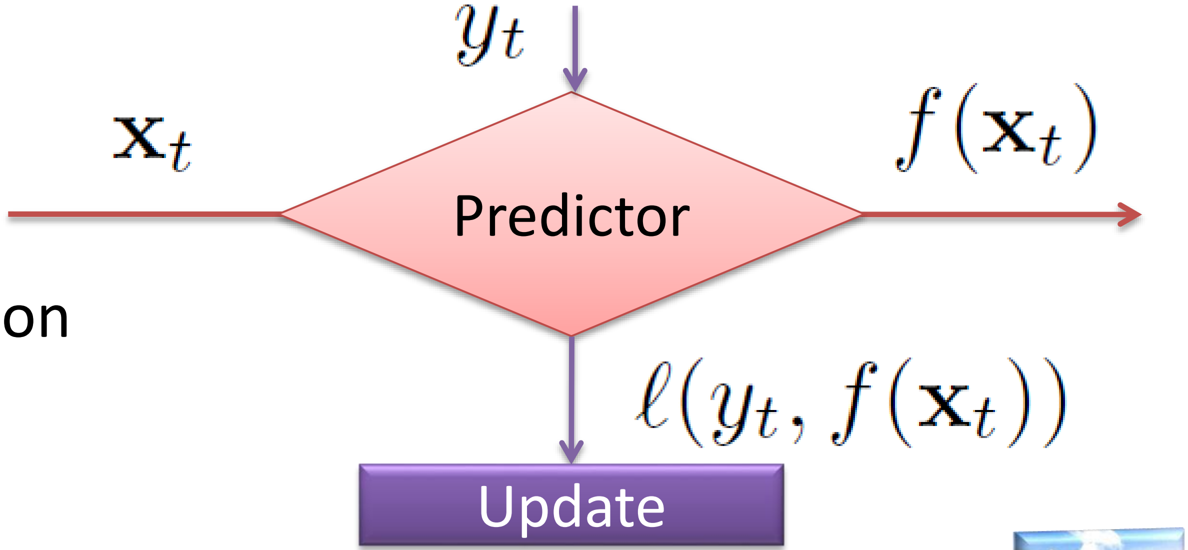
What is Online Learning?

Batch/Offline Learning vs. Online Learning

- Learn a model from a **batch** of training data
- A test data set is used to validate the model

- Learn a model **incrementally** from a **sequence** of instances
- Make the sequence of online predictions accurately

Example:
Online Classification



Why Online Learning?

✓ Avoid re-training when adding new data

✓ High efficiency

✓ Excellent scalability

✓ Strong adaptability to changing environments

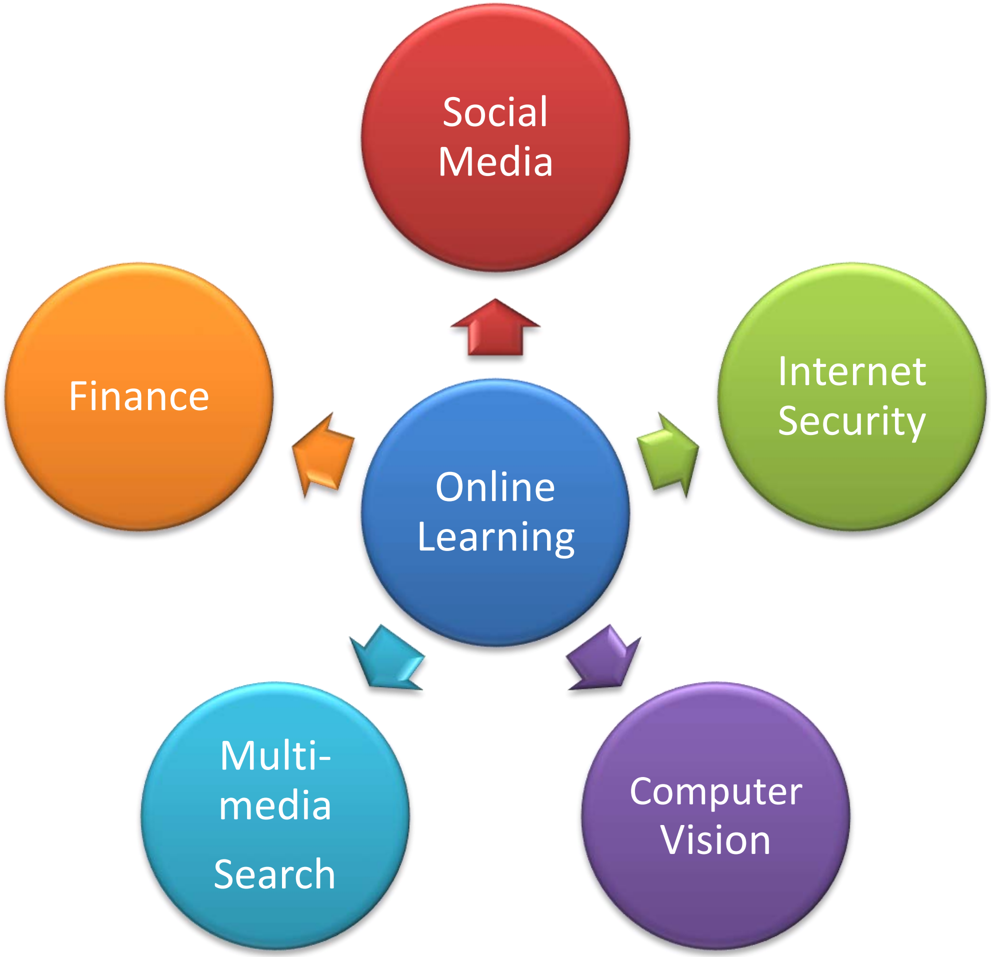
✓ Simple to understand

✓ Trivial to implement

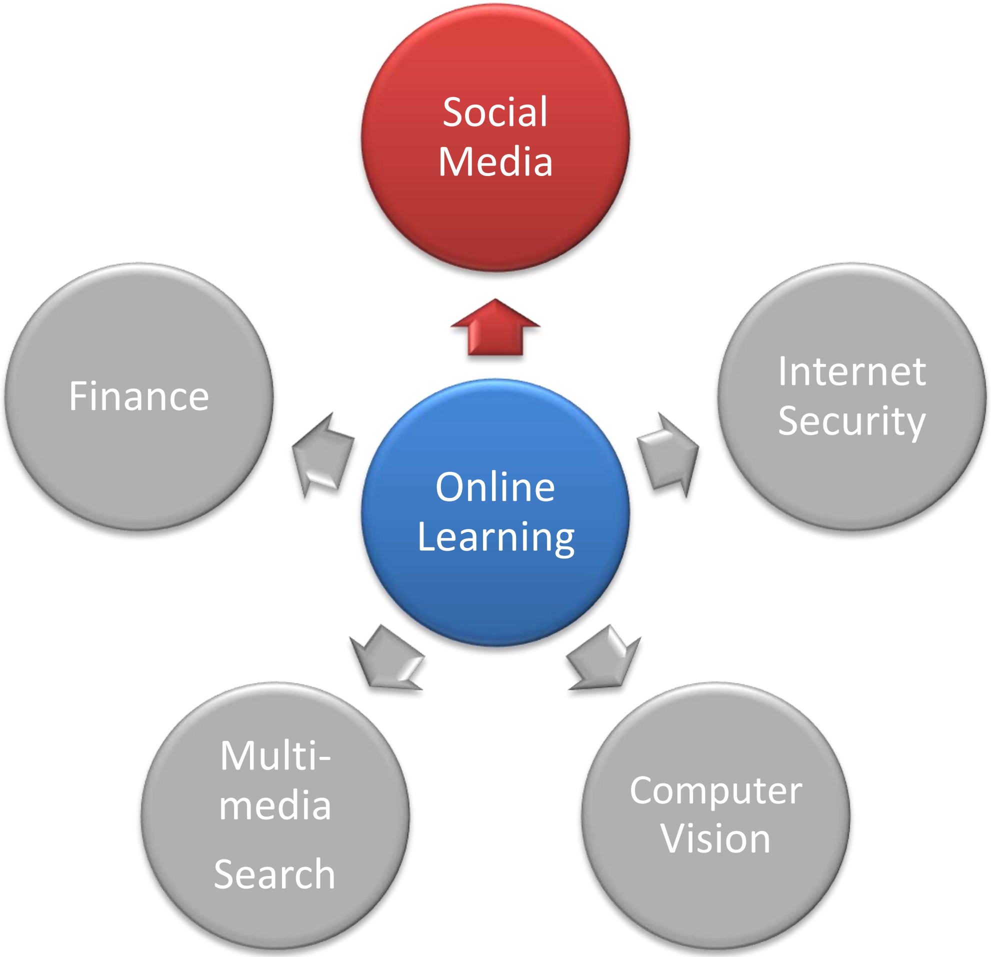
✓ Easy to be parallelized

✓ Theoretical guarantee

Where to apply Online Learning?

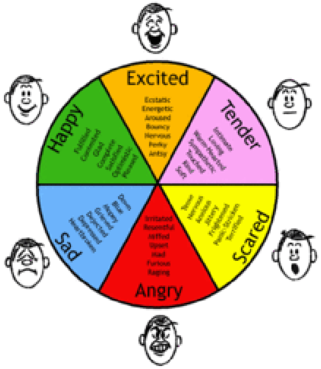


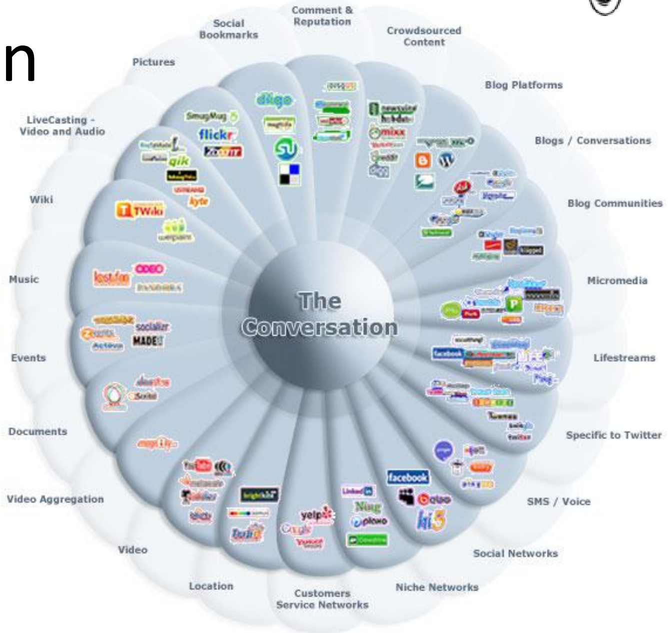
Online Learning : Applications

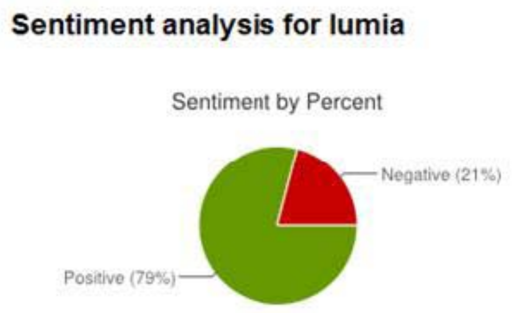


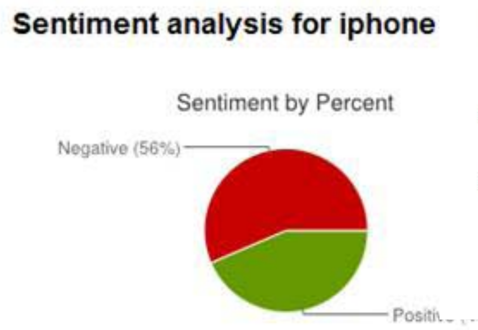
Online Learning for Social Media

- Online mining of social media streams
- Business intelligence applications
  - – Public emotion analytics
  - – Product sentiment detection
  - – Track brand sentiments

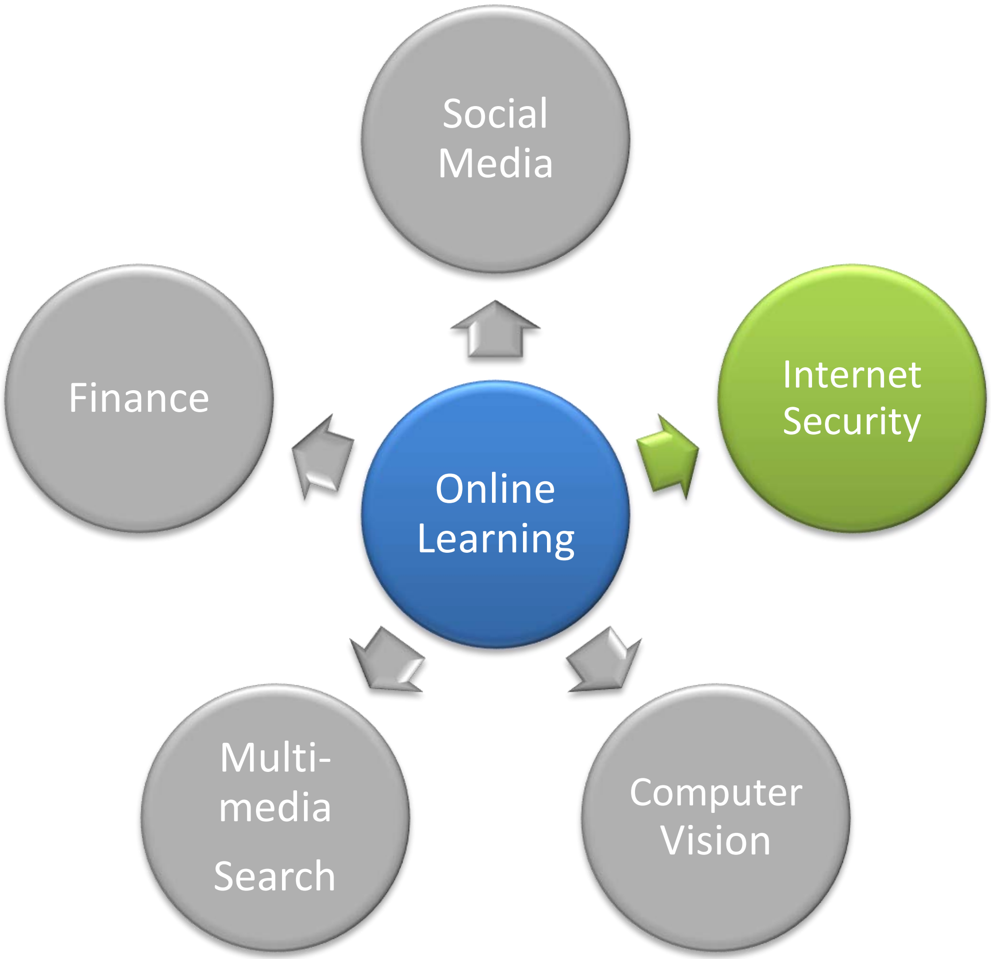





Sentiment analysis for lumia

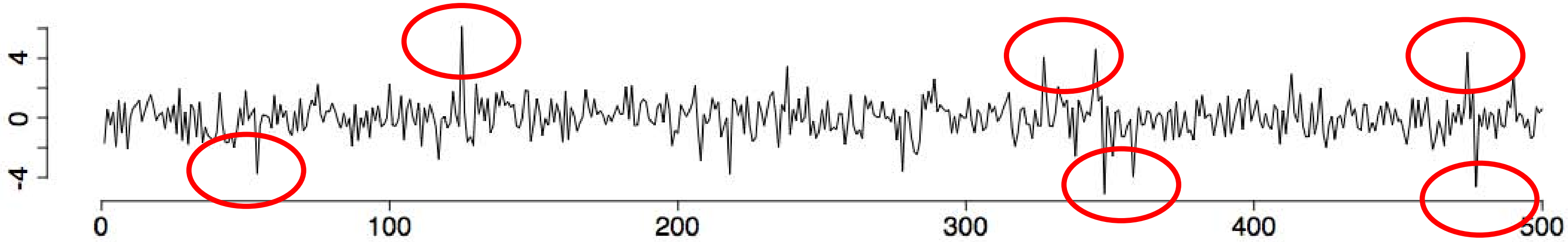

Sentiment analysis for iphone

Online Learning : Applications

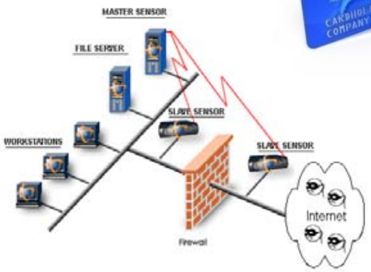


Online Learning for Internet Security

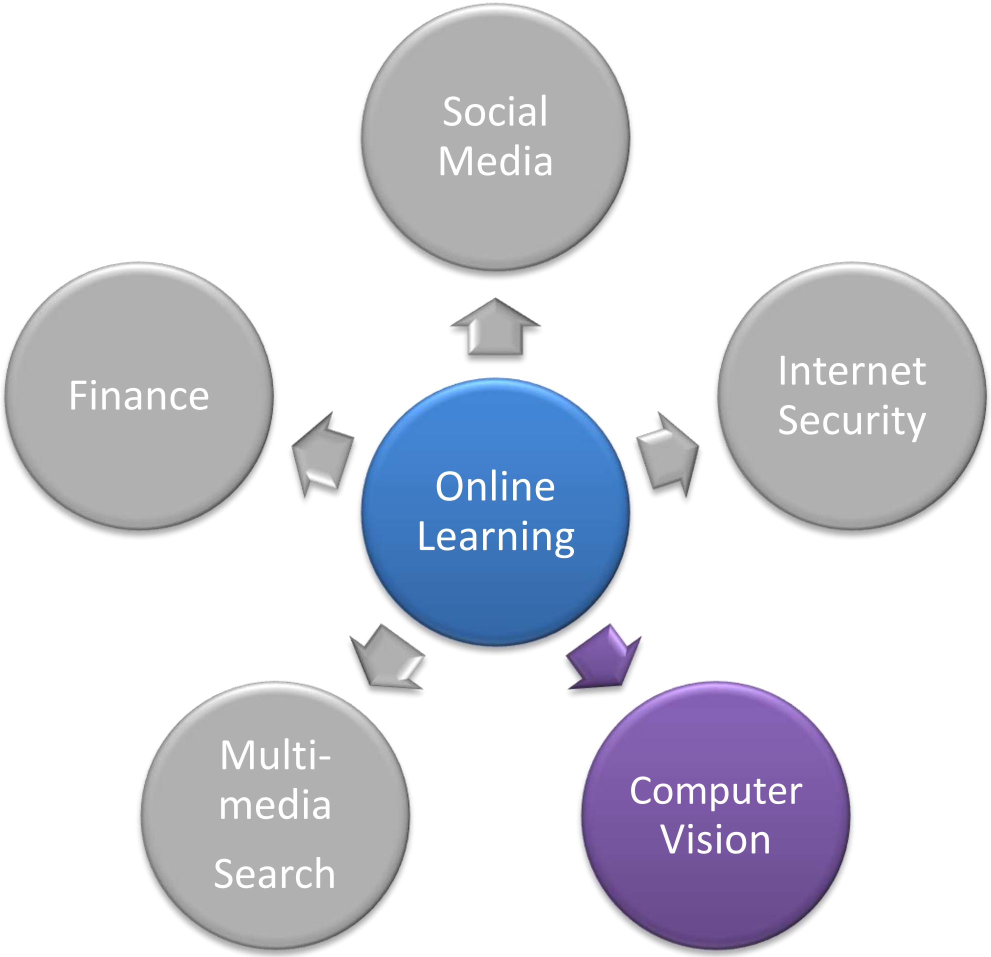
- Online Anomaly Detection (outlier/intrusion)



- Example
  - Detection of **fraud** credit card transactions
  - Network **intrusion detection** systems
  - **Spam email** filtering, etc.



Online Learning : Applications



Online Learning

Social Media

Internet Security

Computer Vision

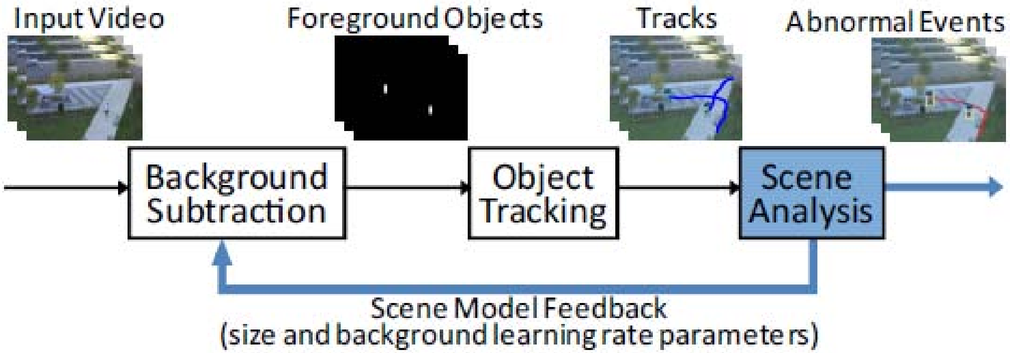
Multi-media

Search

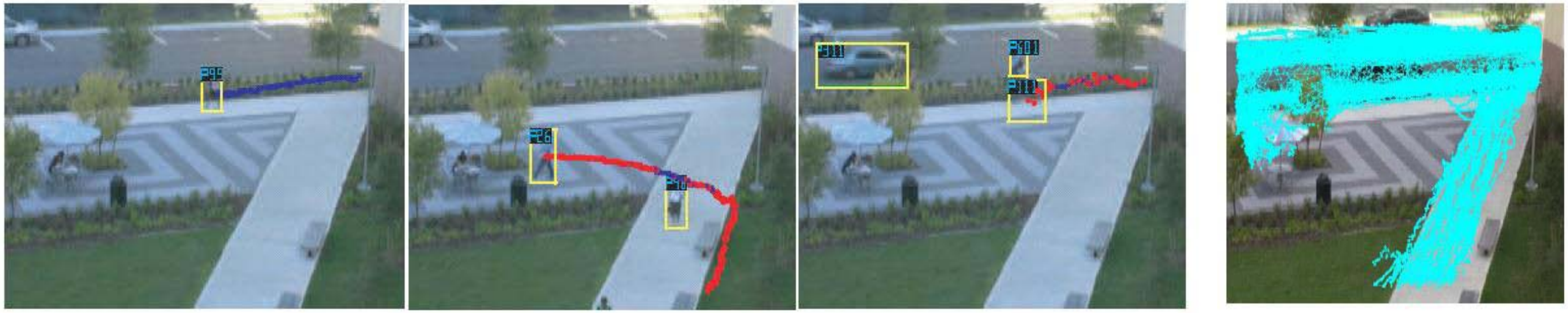
Finance

Online Learning for Computer Vision

- Video surveillance application by online learning
  - Real-time object tracking
  - Detect anomalous events from real-time video streams



(Basharat et al. 2008)

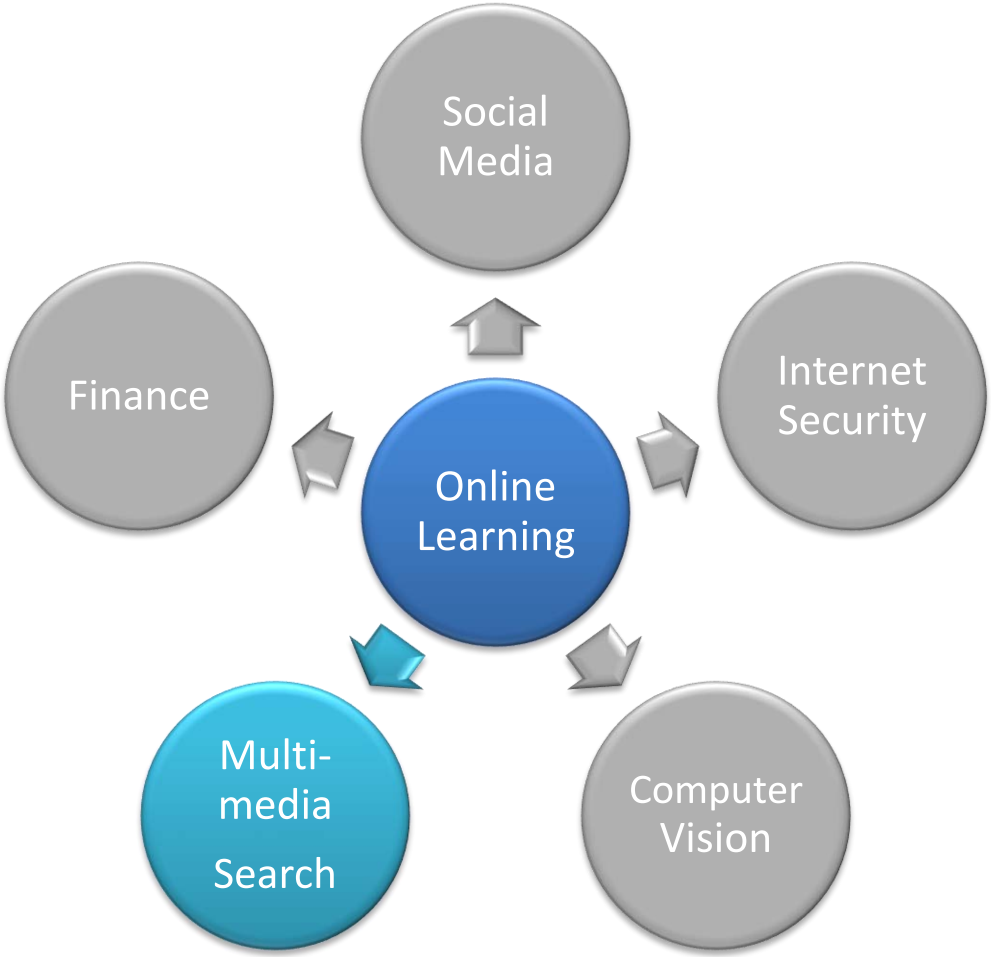


(a) Normal Track

(b) Unusual Path

(c) Bicycle on sidewalk (abnormal size and speed)

Online Learning : Applications



Online Learning

Social Media

Internet Security

Computer Vision

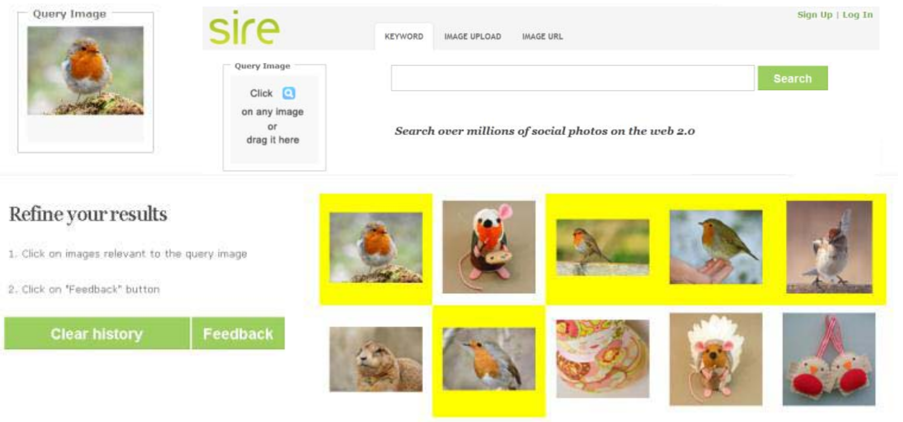
Multi-media

Search

Finance

Online Learning for Multimedia Search

- Web-scale Content-based Multimedia Retrieval
  - Interactive Image/Video Search via online relevance feedback



Query Image

sire

Sign Up | Log In

KEYWORD IMAGE UPLOAD IMAGE URL

Query Image

Click on any image or drag it here

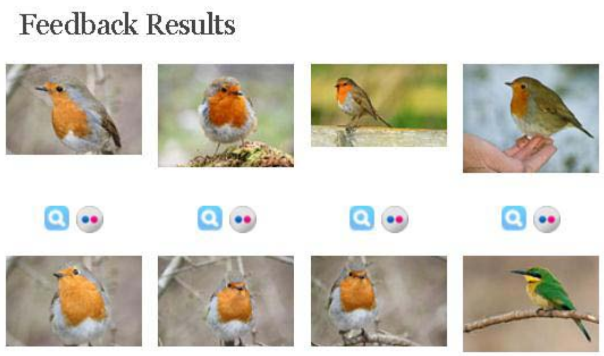
Search

Search over millions of social photos on the web 2.0

Refine your results

1. Click on images relevant to the query image
2. Click on "Feedback" button

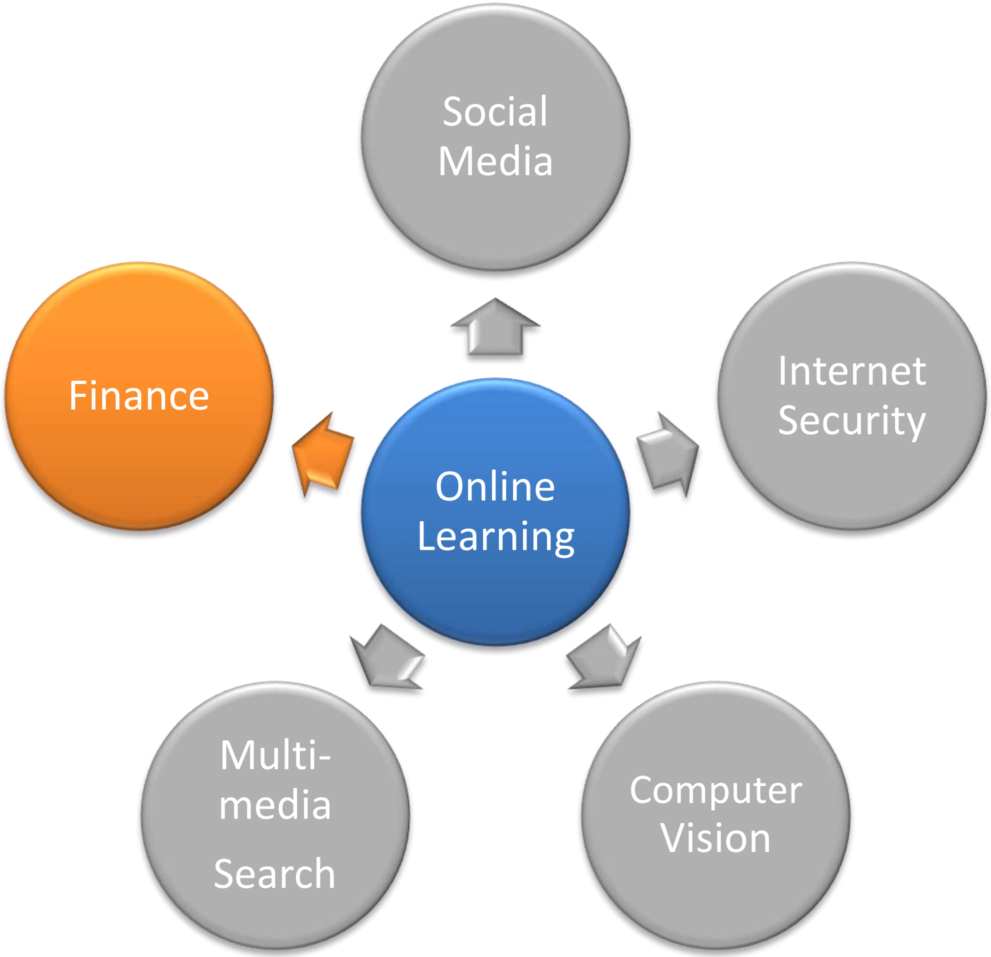
Clear history | Feedback



Feedback Results

- Collaborative Multimedia Search & Annotation
  - Mining massive side info (e.g., search log data) incrementally
  - Example: distance metric learning, online kernel learning, etc.

Online Learning : Applications



Online Learning

Social Media

Internet Security

Computer Vision

Multi-media

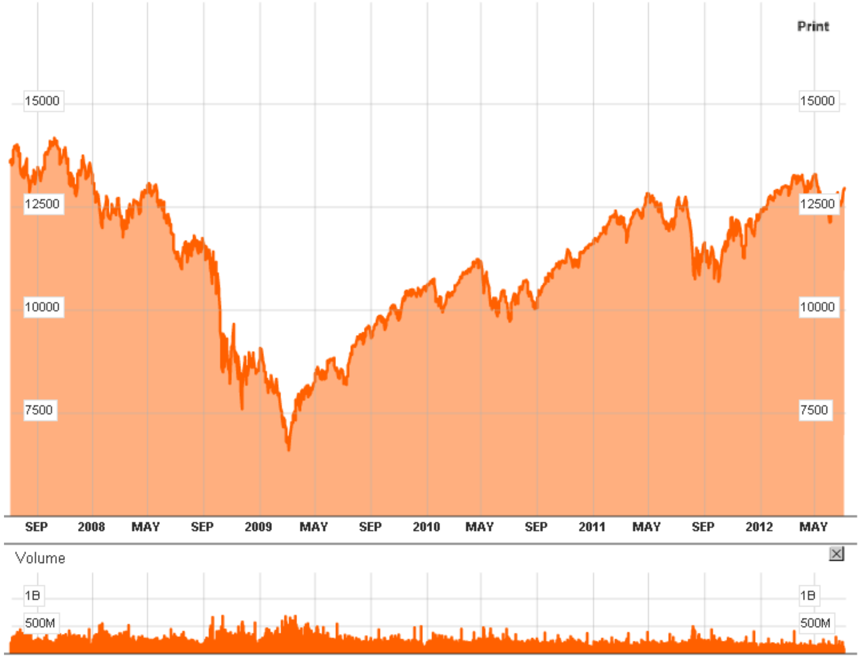
Search

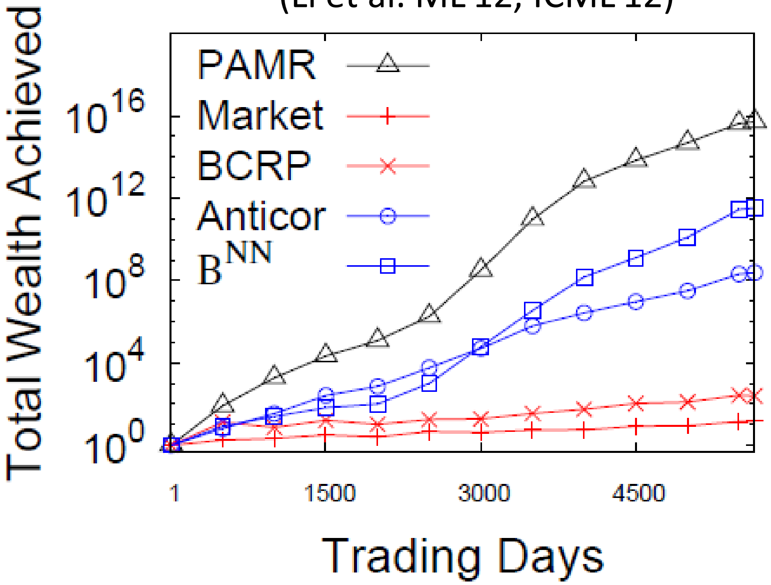
Finance

Online Learning for Finance

- On-line Portfolio Selection
  - Sequential decisions of investing wealth among assets

(Li et al. ML'12, ICML'12)





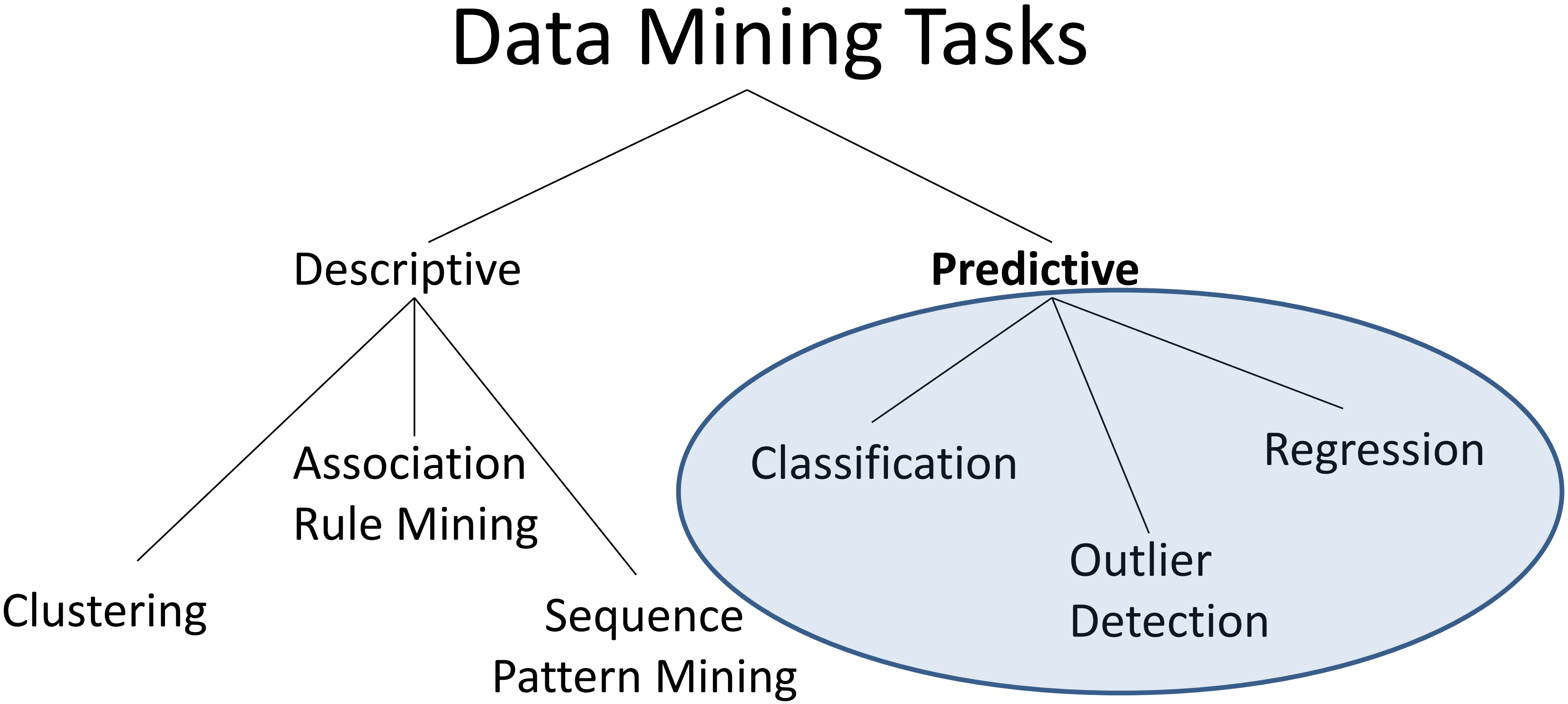
(a) NYSE (O) dataset

Agenda

- Introduction (25 min)
  - Big Data Mining: Opportunities & Challenges
  - Online Learning: What, Why, Where
- Online Learning
  - Overview (5 min)
  - Traditional Linear Online Learning (30 min)
  - Non-traditional Linear Online Learning (30 min)
  - Kernel-based Online Learning (30 min)
  - Online Multiple Kernel Learning (30 min)
- Discussions + Q&A (30 min)

Online Learning: Overview

- Big Data Mining: topic coverage



Data Mining Tasks

Descriptive

Clustering

Association Rule Mining

Sequence Pattern Mining

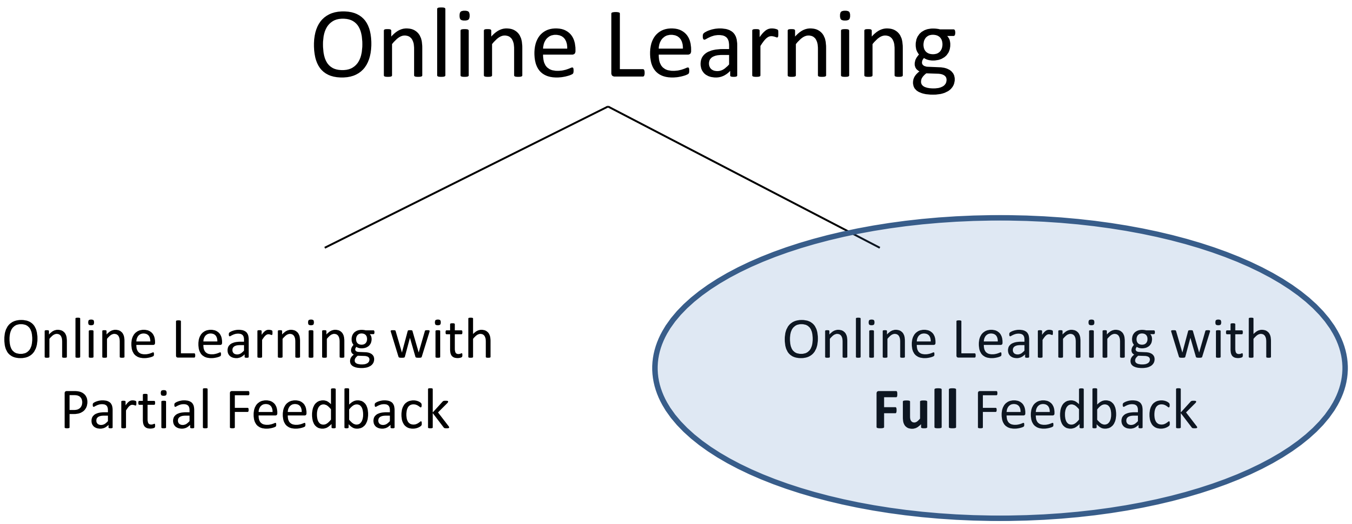
Predictive

Classification

Outlier Detection

Regression

Online Learning: Overview



Online Learning

Online Learning with Partial Feedback

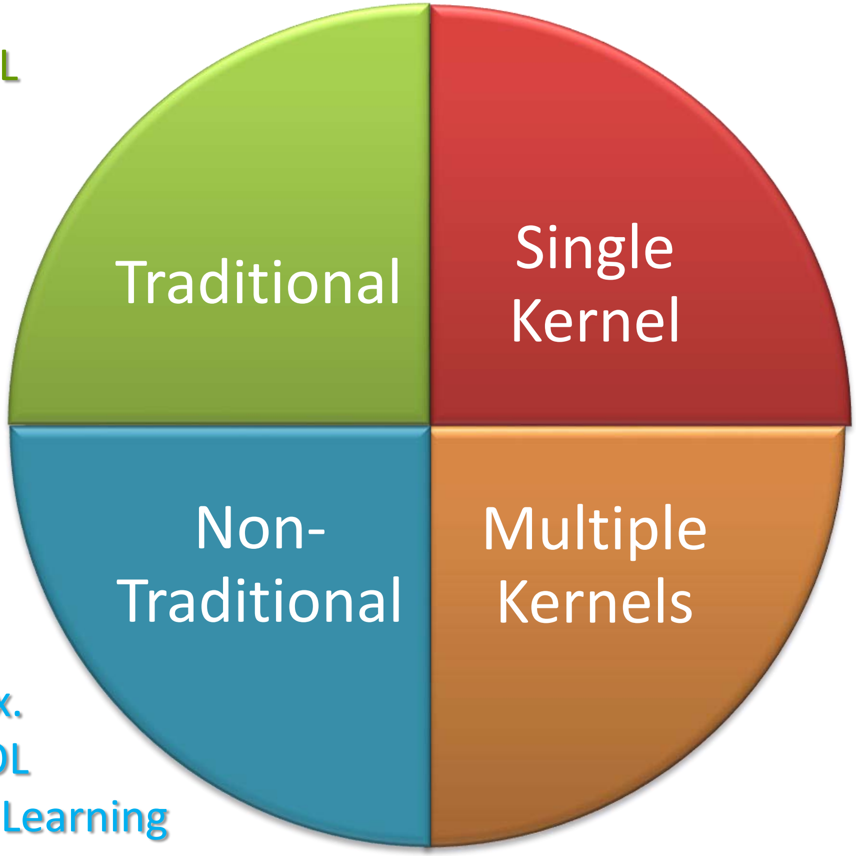
Online Learning with **Full** Feedback

- Not covered in this tutorial
  - Bandits
    - ACML12 Tutorial: http://www.princeton.edu/~sbubeck/tutorial.html
    - ICML Tutorial: https://sites.google.com/site/banditstutorial/
    - Prediction, Learning, and Games (Nicolo Cesa-Bianchi & Gabor Lugosi)
  - Reinforcement learning
    - http://chercheurs.lille.inria.fr/~ghavamza/ICML2012-Tutorial.html
    - http://hunch.net/~jl/projects/RL/RLTheoryTutorial.pdf

Online Learning: Overview

✓First order OL
✓Second order OL
✓Sparse OL

Linear
Methods



Traditional

Single Kernel

Non-Traditional

Multiple Kernels

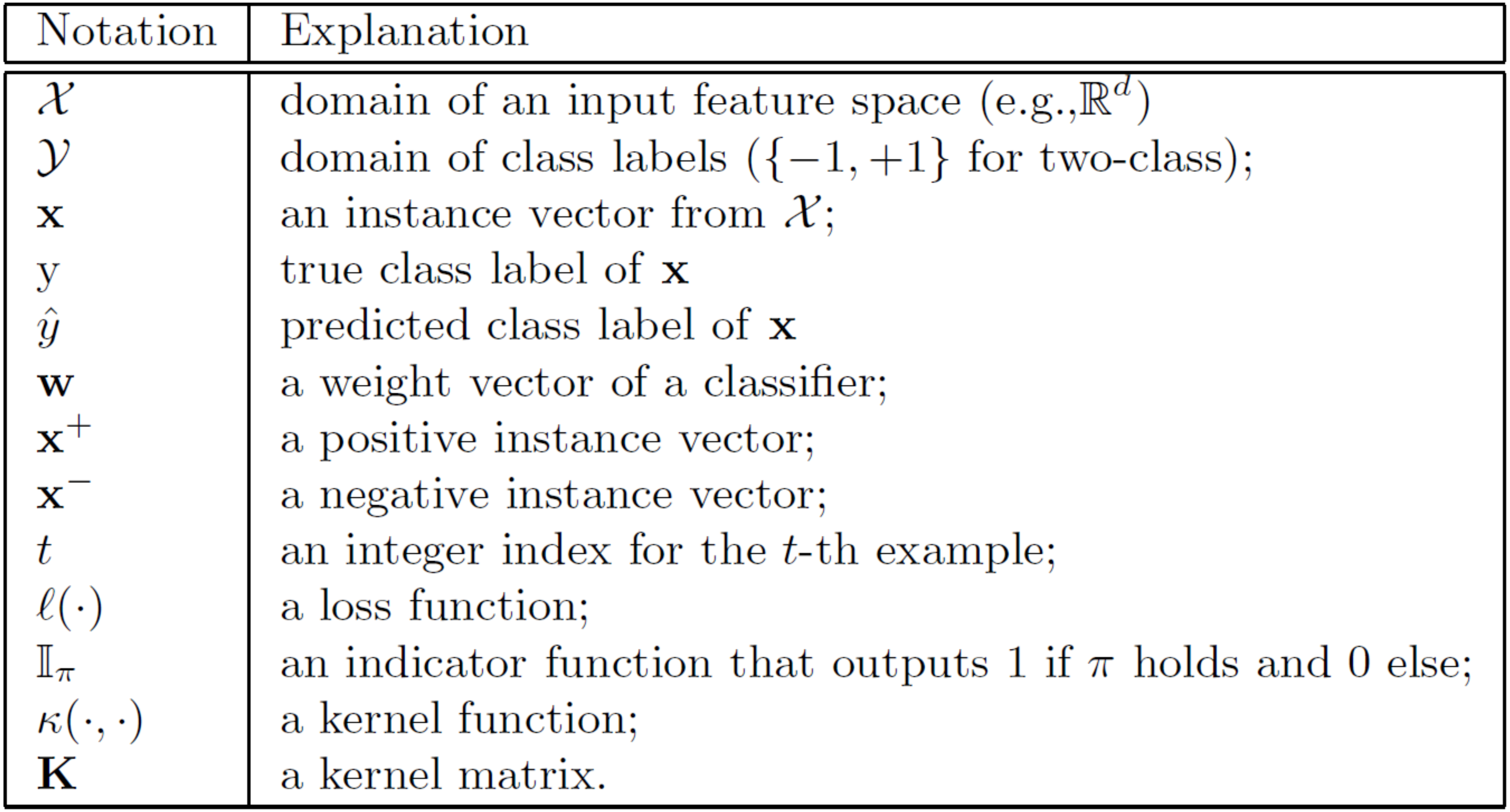
✓Kernel OL
✓Budget OL

Non-Linear
Methods

✓Online MKL
✓OMKC
✓OMDL

✓Online AUC Max.
✓Cost-Sensitive OL
✓Online Transfer Learning

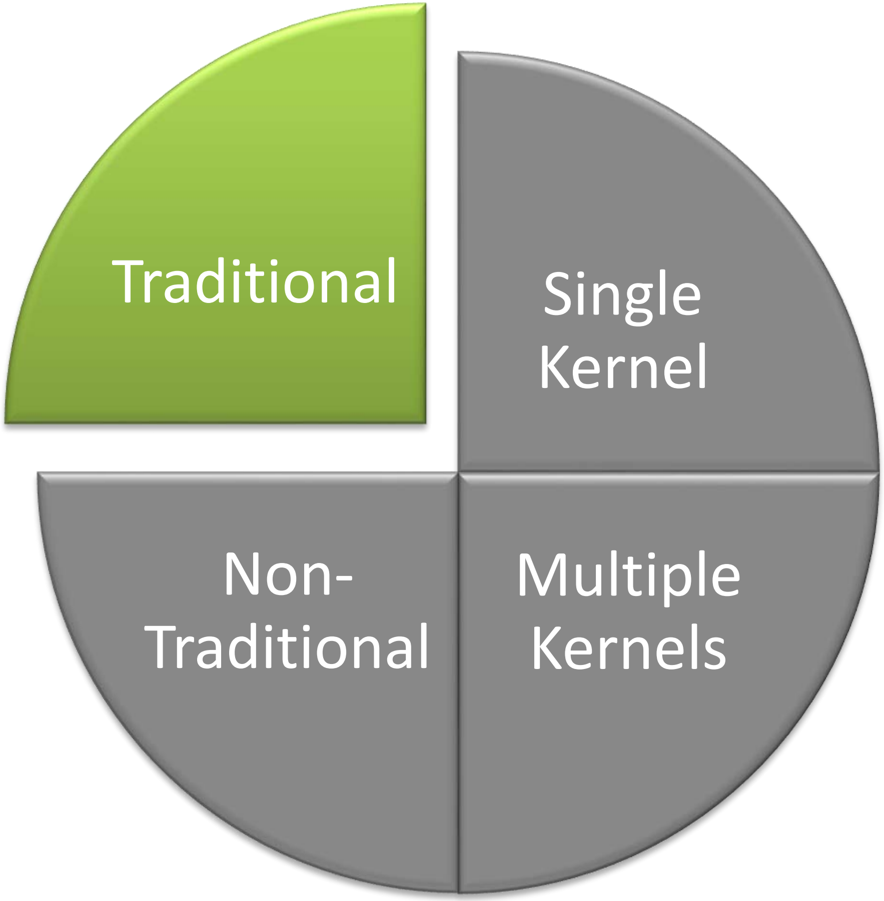
Notation



Notation	Explanation
$\mathcal{X}$	domain of an input feature space (e.g.,$\mathbb{R}^d$)
$\mathcal{Y}$	domain of class labels ($\{-1, +1\}$ for two-class);
$\mathbf{x}$	an instance vector from $\mathcal{X}$;
y	true class label of $\mathbf{x}$
$\hat{y}$	predicted class label of $\mathbf{x}$
$\mathbf{w}$	a weight vector of a classifier;
$\mathbf{x}^+$	a positive instance vector;
$\mathbf{x}^-$	a negative instance vector;
t	an integer index for the t-th example;
$\ell(\cdot)$	a loss function;
$\mathbb{I}_\pi$	an indicator function that outputs 1 if π holds and 0 else;
$\kappa(\cdot, \cdot)$	a kernel function;
$\mathbf{K}$	a kernel matrix.

Online Learning: Overview

Linear
Methods



Traditional

Single Kernel

Non-Traditional

Multiple Kernels

Non-Linear
Methods

Traditional Linear Online Learning

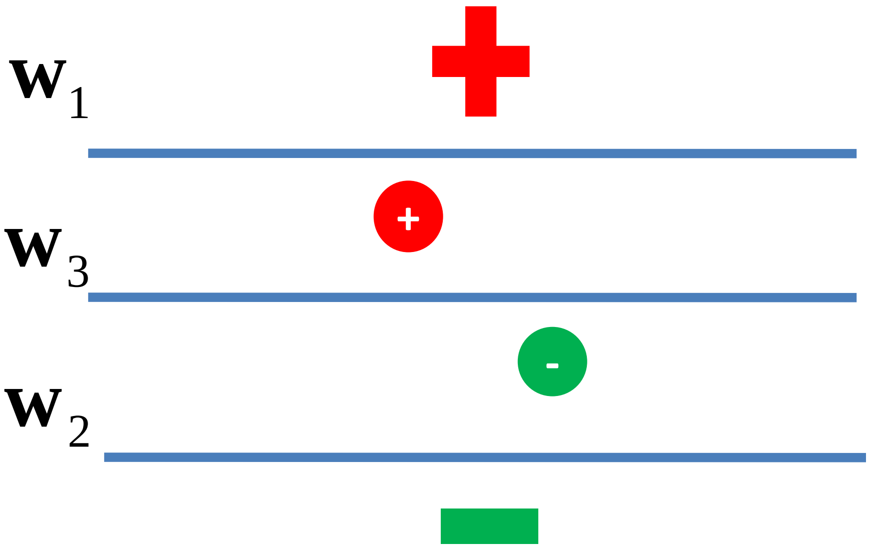
- Online learning protocol for classification

The following scenario is **repeated** indefinitely:

- The algorithm **receives** an unlabeled example
- The algorithm **predicts** a classification of this example;
- The algorithm is then **told** the correct answer
- The algorithm **updates** the classifier when appropriate

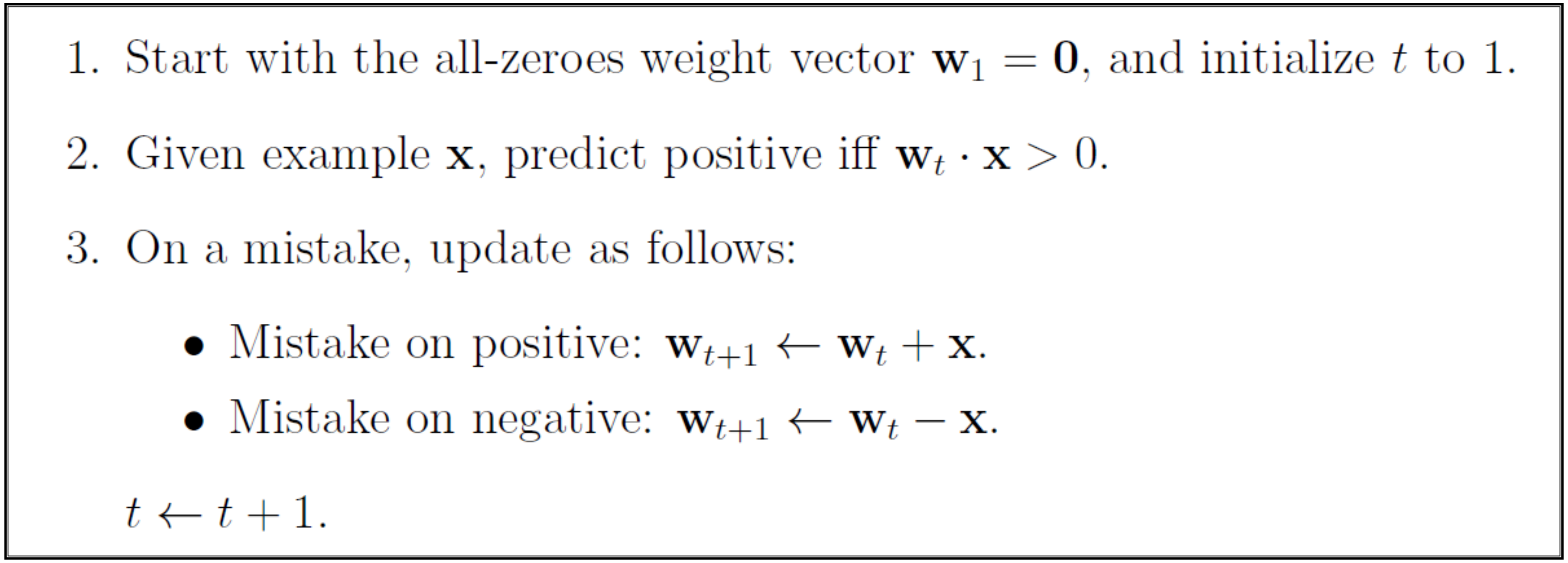
- Objective
  - To minimize the **mistake** rate of online classification

Perceptron Algorithm (Rosenblatt Frank, 1958)



$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}_t$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}_t$$



1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.

Traditional Linear Online Learning (cont')

- First Order Learning methods
  - Perceptron (Rosenblatt, Frank, 1958)
  - Online Gradient Descent (Zinkevich et al., 2003)
  - Passive Aggressive learning (Crammer et al., 2006)
  - Others (including but not limited)
    - MIRA: Margin Infused Relaxed Algorithm (Crammer and Singer, 2003)
    - NORMA: Naive Online R-reg Minimization Algorithm (Kivinen et al., 2002)
    - ROMMA: Relaxed Online Maximum Margin Algorithm (Li and Long, 2002)
    - ALMA: A New Approximate Maximal Margin Classification Algorithm (Gentile, 2001)

Online Gradient Descent

- Online Convex Optimization (Zinkevich et al., 2003)
  - Consider a convex objective function

$$f : S \to \mathbb{R}$$

where $S \subset \mathbb{R}^n$ is a bounded convex set

  - The update by Online Gradient Descent (OGD) or Stochastic Gradient Descent (SGD):

$$\mathbf{w}_{t+1} \leftarrow \Pi_S(\mathbf{w}_t - \eta \nabla f(\mathbf{w}_t))$$

where η is called the learning rate

Online Gradient Descent (OGD) algorithm

- Repeat from t=1,2,…
  - An unlabeled example $\mathbf{x}_t$ *arrives*
  - Make a prediction based on existing weights

$$\hat{y}_t = \mathrm{sgn}(\mathbf{w}_t^\top \mathbf{x}_t)$$

  - Observe the true class label $y_t \in \{+1, -1\}$
  - Update the weights by the OGD rule:

$$\mathbf{w}_{t+1} \leftarrow \Pi_S(\mathbf{w}_t - \eta \nabla f(\mathbf{w}_t))$$

where $\eta > 0$ is a learning rate

Passive Aggressive Online Learning

- Passive Aggressive learning (Crammer et al., 2006)
  - **PA**

$$\mathbf{w}_{t+1} = \arg\min \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t))) = 0.$$

$$\ell(\mathbf{w}; (\mathbf{x}, y)) = \begin{cases} 0 & y(\mathbf{w} \cdot \mathbf{x}) \geq 1 \\ 1 - y(\mathbf{w} \cdot \mathbf{x}) & \text{otherwise} \end{cases}$$

  - **PA-I**

$$\mathbf{w}_{t+1} = \arg\min \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi \quad \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t))) \leq \xi \quad \text{and} \quad \xi \geq 0.$$

  - **PA-II**

$$\mathbf{w}_{t+1} = \arg\min \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 \quad \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t))) \leq \xi.$$

Passive Aggressive Online Learning

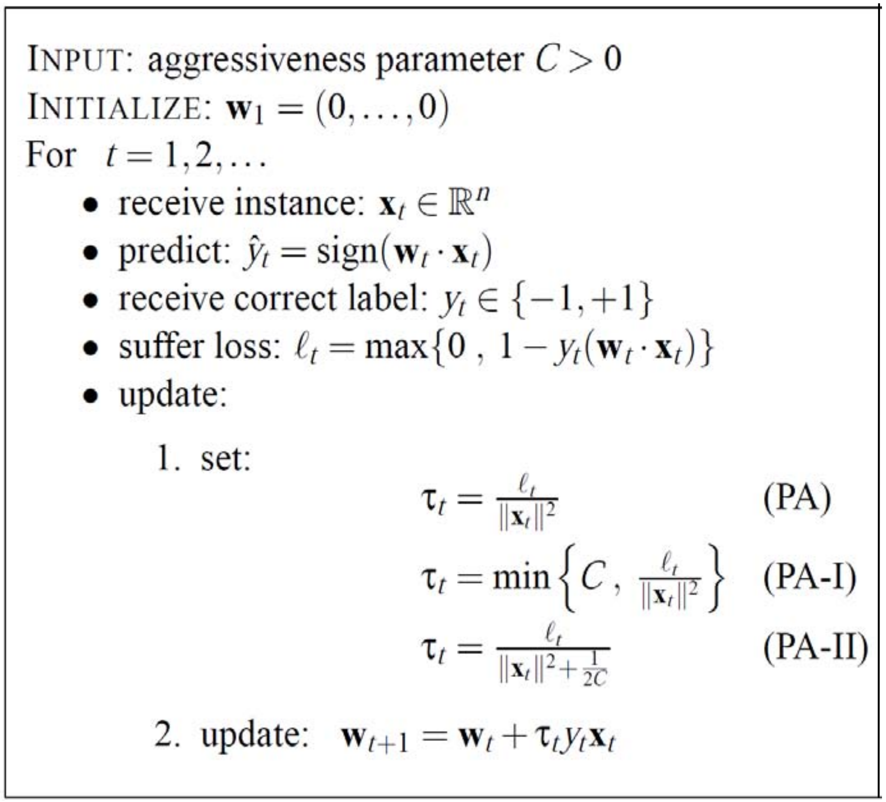
- Closed-form solutions can be derived:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$$

$$\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2} \qquad \text{(PA)}$$

$$\tau_t = \min\left\{C\,,\ \frac{\ell_t}{\|\mathbf{x}_t\|^2}\right\} \qquad \text{(PA-I)}$$

$$\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} \qquad \text{(PA-II)}$$



```
INPUT: aggressiveness parameter $C > 0$
INITIALIZE: $\mathbf{w}_1 = (0,\ldots,0)$
For  $t = 1,2,\ldots$
    • receive instance: $\mathbf{x}_t \in \mathbb{R}^n$
    • predict: $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
    • receive correct label: $y_t \in \{-1,+1\}$
    • suffer loss: $\ell_t = \max\{0\ ,\ 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\}$
    • update:
        1. set:
                $\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2}$                                  (PA)
                $\tau_t = \min\left\{C\,,\ \frac{\ell_t}{\|\mathbf{x}_t\|^2}\right\}$          (PA-I)
                $\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}}$                    (PA-II)
        2. update:  $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$
```

Traditional Linear Online Learning (cont')

- **First-Order** methods
  - Learn a **linear** weight vector (first order) of model
- Pros and Cons
  - ☺ Simple and easy to implement
  - ☺ Efficient and scalable for high-dimensional data
  - ☹ Relatively slow convergence rate

Second Order Online Learning methods

- Key idea
  - Update the weight vector **w** by maintaining and exploring second order information in addition to the first order information
- Some representative methods
  - **SOP:** Second order Perceptron (Cesa-Bianchi et al, 2005)
  - **CW:** Confidence Weighted learning (Dredze et al, 2008)
  - **AROW:** Adaptive Regularization of Weights (Crammer, 2009)
  - **SCW:** Soft Confidence Weighted (**SCW**) (Wang et al, 2012)
  - Others (but not limited)
    - IELLIP:Online Learning by Ellipsoid Method (Yang et al., 2009)
    - NHERD: Gaussian Herding (Crammer & Lee 2010)
    - NAROW: New variant of AROW algorithm (Orabona & Crammer 2010)

SOP: Second Order Perceptron

- SOP: Second order Perceptron (Cesa-Bianch et al. 2005)
  - Whiten Perceptron (Not incremental!!)
    - Correlation matrix $M = \sum_{t=1}^{T} \boldsymbol{x}_t \boldsymbol{x}_t^\top$
    - Simply run a standard Perceptron for the following

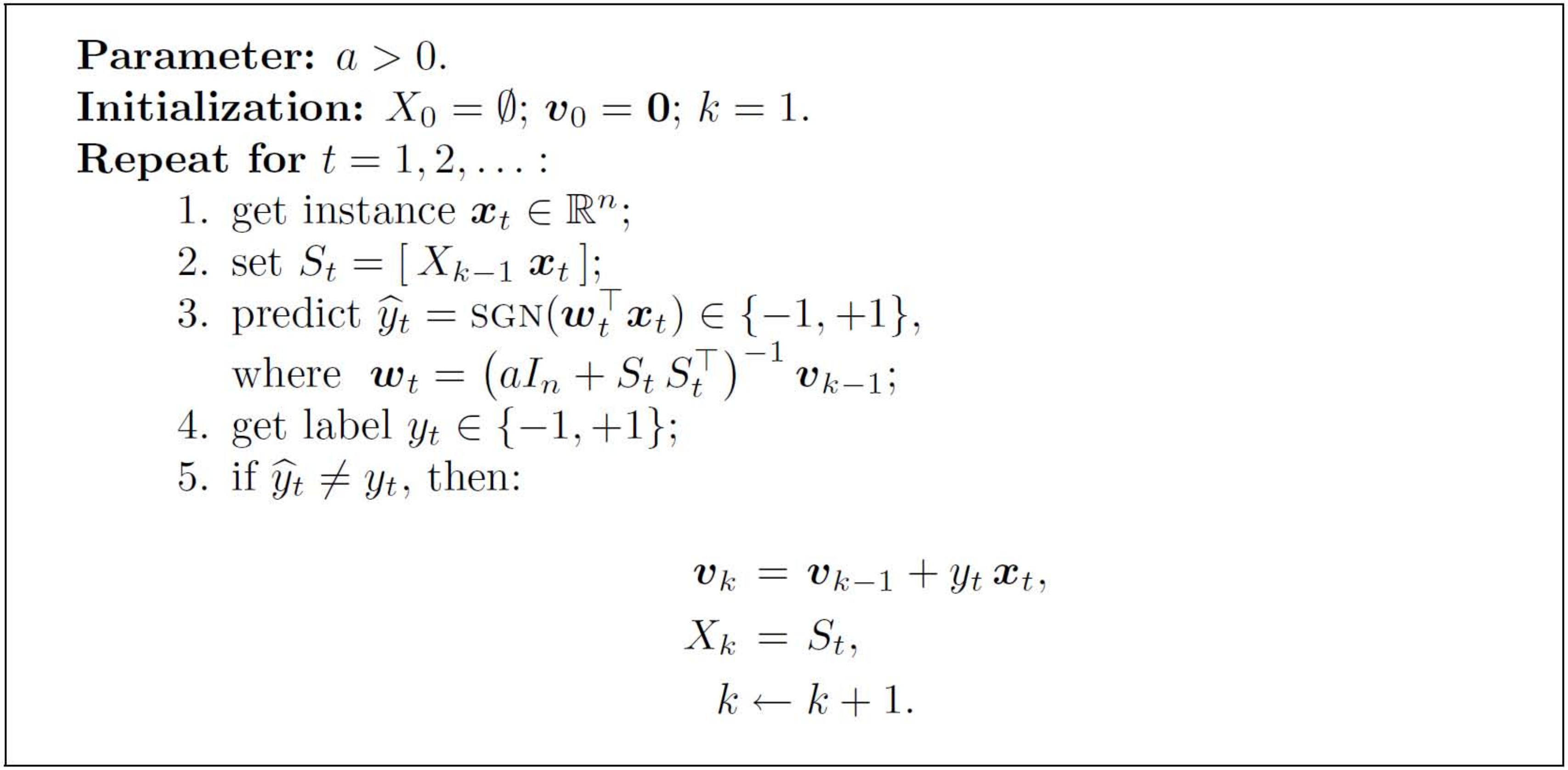
$$(M^{-1/2}\boldsymbol{x}_1, y_1), (M^{-1/2}\boldsymbol{x}_2, y_2), \ldots, (M^{-1/2}\boldsymbol{x}_T, y_T)$$

$$\begin{aligned}\sum_{t=1}^{T}\left(M^{-1/2}\boldsymbol{x}_t\right)\left(M^{-1/2}\boldsymbol{x}_t\right)^\top &= \sum_{t=1}^{T} M^{-1/2}\boldsymbol{x}_t\,\boldsymbol{x}_t^\top M^{-1/2} \\ &= M^{-1/2} M\, M^{-1/2} \\ &= I_n .\end{aligned}$$

  - Online algorithm (an incremental variant of Whiten Perceptron)
    - Augmented matrix: $S_t = [\, X_{k-1} \;\; \boldsymbol{x}_t \,]$ $X_0 = \emptyset$ (the empty matrix)
    - Correlation matrix: $aI_n + S_t S_t^\top$

SOP: Second Order Perceptron

- SOP: Second order Perceptron (Cesa-Bianch et al. 2005)



```
Parameter: $a > 0$.
Initialization: $X_0 = \emptyset$; $\boldsymbol{v}_0 = \mathbf{0}$; $k = 1$.
Repeat for $t = 1, 2, \dots$ :
    1. get instance $\boldsymbol{x}_t \in \mathbb{R}^n$;
    2. set $S_t = [\, X_{k-1} \; \boldsymbol{x}_t \,]$;
    3. predict $\widehat{y}_t = \mathrm{SGN}(\boldsymbol{w}_t^\top \boldsymbol{x}_t) \in \{-1, +1\}$,
       where $\boldsymbol{w}_t = \left(aI_n + S_t \, S_t^\top\right)^{-1} \boldsymbol{v}_{k-1}$;
    4. get label $y_t \in \{-1, +1\}$;
    5. if $\widehat{y}_t \neq y_t$, then:
           $\boldsymbol{v}_k = \boldsymbol{v}_{k-1} + y_t \, \boldsymbol{x}_t$,
           $X_k = S_t$,
           $k \leftarrow k + 1$.
```

CW: Confidence Weighted learning

- CW: Confidence Weighted learning (Dredze et al. 2008)
  - Draw a parameter vector $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$
  - The margin is viewed as a random variable:

$$M \sim \mathcal{N}\left(y_i(\boldsymbol{\mu} \cdot \boldsymbol{x}_i)\ ,\ \boldsymbol{x}_i^{\top} \Sigma \boldsymbol{x}_i\right)$$

  - The probability of a correct prediction is

$$\Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)}[M \geq 0] = \Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)}[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i) \geq 0]$$

  - Optimization of CW

$$\begin{aligned}(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}) = \min\ & \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \parallel \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right) \\ & \text{s.t. } \Pr\left[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i) \geq 0\right] \geq \eta\ .\end{aligned}$$

$$\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0) \parallel \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)\right) = \frac{1}{2}\left(\log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right) + \mathrm{Tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\top} \Sigma_1^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - d\right)$$

CW: Confidence Weighted learning

$\Pr\left[y_t\left(\boldsymbol{w}\cdot\boldsymbol{x}_t\right)\geq 0\right]\geq\eta$ can be written as

$y_t\left(\boldsymbol{\mu}\cdot\boldsymbol{x}_t\right)\ \geq\ \phi\sqrt{\boldsymbol{x}_t^\top\Sigma\boldsymbol{x}_t}$ $\phi=\Phi^{-1}\left(\eta\right)$ Φ is the cumulative function of the normal distribution.

$$(\boldsymbol{\mu}_{i+1},\Sigma_{i+1})=\min\frac{1}{2}\log\left(\frac{\det\Sigma_i}{\det\Sigma}\right)+\frac{1}{2}\mathrm{Tr}\left(\Sigma_i^{-1}\Sigma\right)+\frac{1}{2}\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}\right)^\top\Sigma_i^{-1}\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}\right)$$

$$\text{s.t.}\quad y_i(\boldsymbol{\mu}\cdot\boldsymbol{x}_i)\geq\phi\left(\boldsymbol{x}_i^\top\Sigma\boldsymbol{x}_i\right)\ .$$

Algorithm 1 Variance Algorithm (Approximate)

```
Input: confidence parameter $\phi = \Phi^{-1}(\eta)$
       initial variance parameter $a > 0$
Initialize: $\boldsymbol{\mu}_1 = \mathbf{0}$ , $\Sigma_1 = aI$
for $i = 1, 2 \ldots$ do
    Receive $\boldsymbol{x}_i \in \mathbb{R}^d$ , $y_i \in \{+1, -1\}$
    Set the following variables:
        $\alpha_i$ as in Lemma 1
        $\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i y_i \Sigma_i \boldsymbol{x}_i$ (11)
        $\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha_i\phi\,\mathrm{diag}(\boldsymbol{x}_i)$ (17)
end for
```

$$\begin{aligned}\mathcal{L}\ &=\ \frac{1}{2}\log\left(\frac{\det\Sigma_i}{\det\Sigma}\right)+\frac{1}{2}\mathrm{Tr}\left(\Sigma_i^{-1}\Sigma\right)\\ &\quad+\frac{1}{2}\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}\right)^\top\Sigma_i^{-1}\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}\right)\\ &\quad+\alpha\left(-y_i\left(\boldsymbol{\mu}\cdot\boldsymbol{x}_i\right)+\phi\left(\boldsymbol{x}_i^\top\Sigma\boldsymbol{x}_i\right)\right)\end{aligned}$$

Lemma 1: The optimal value of the Lagrange multiplier is given by $\alpha_i=\max\left\{\gamma_i,0\right\}$

$$\gamma_i=\frac{-(1+2\phi M_i)+\sqrt{(1+2\phi M_i)^2-8\phi\left(M_i-\phi V_i\right)}}{4\phi V_i}$$

$M_i=y_i\left(\boldsymbol{x}_i\cdot\boldsymbol{\mu}_i\right)\ V_i\ =\ \boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i$

AROW: Adaptive Regularization of Weights

- AROW (Crammer et al. 2009)
  - Extension of CW learning
  - Key properties: large margin training, confidence weighting, and the capacity to handle non-separable data
- Formulations

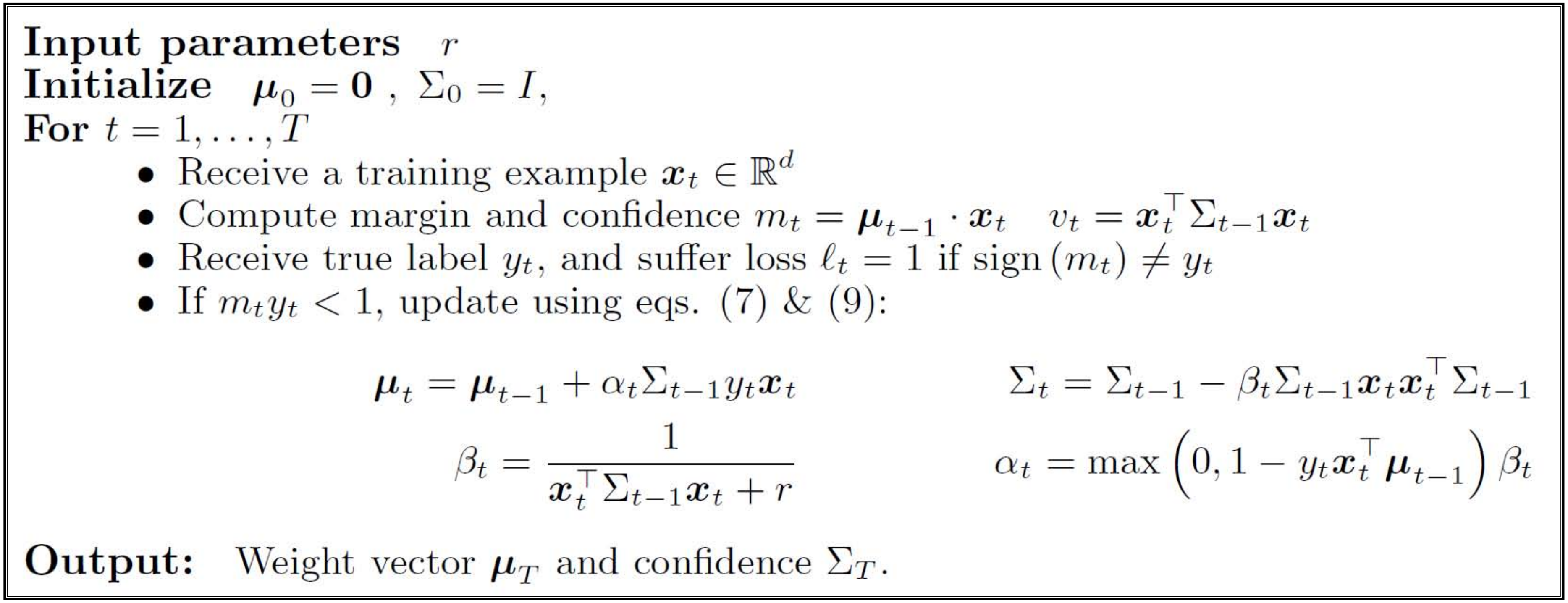
$$\mathcal{C}\left(\boldsymbol{\mu},\Sigma\right) = \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu},\Sigma\right) \parallel \mathcal{N}\left(\boldsymbol{\mu}_{t-1},\Sigma_{t-1}\right)\right) + \lambda_1 \ell_{\mathrm{h}^2}\left(y_t, \boldsymbol{\mu}\cdot\boldsymbol{x}_t\right) + \lambda_2 \boldsymbol{x}_t^{\top} \Sigma \boldsymbol{x}_t$$

$$\ell_{\mathrm{h}^2}\left(y_t, \boldsymbol{\mu}\cdot\boldsymbol{x}_t\right) = \left(\max\{0, 1 - y_t(\boldsymbol{\mu}\cdot\boldsymbol{x}_t)\}\right)^2$$

$$\begin{aligned}\mathcal{C}\left(\boldsymbol{\mu},\Sigma\right) = \frac{1}{2}\log\left(\frac{\det \Sigma_{t-1}}{\det \Sigma}\right) + \frac{1}{2}\mathrm{Tr}\left(\Sigma_{t-1}^{-1}\Sigma\right) + \frac{1}{2}\left(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}\right)^{\top}\Sigma_{t-1}^{-1}\left(\boldsymbol{\mu}_{t-1} - \boldsymbol{\mu}\right) - \frac{d}{2} \\ + \frac{1}{2r}\ell_{\mathrm{h}^2}\left(y_t, \boldsymbol{\mu}\cdot\boldsymbol{x}_t\right) + \frac{1}{2r}\boldsymbol{x}_t^{\top}\Sigma\boldsymbol{x}_t\end{aligned}$$

AROW: Adaptive Regularization of Weights

- AROW algorithm (Crammer et al. 2009)



```
Input parameters $r$
Initialize $\boldsymbol{\mu}_0 = \mathbf{0}$ , $\Sigma_0 = I$,
For $t = 1, \ldots, T$
    • Receive a training example $\boldsymbol{x}_t \in \mathbb{R}^d$
    • Compute margin and confidence $m_t = \boldsymbol{\mu}_{t-1} \cdot \boldsymbol{x}_t \quad v_t = \boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t$
    • Receive true label $y_t$, and suffer loss $\ell_t = 1$ if sign$(m_t) \neq y_t$
    • If $m_t y_t < 1$, update using eqs. (7) & (9):

          $\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \alpha_t \Sigma_{t-1} y_t \boldsymbol{x}_t \qquad \Sigma_t = \Sigma_{t-1} - \beta_t \Sigma_{t-1} \boldsymbol{x}_t \boldsymbol{x}_t^\top \Sigma_{t-1}$
          $\beta_t = \frac{1}{\boldsymbol{x}_t^\top \Sigma_{t-1} \boldsymbol{x}_t + r} \qquad \alpha_t = \max\left(0, 1 - y_t \boldsymbol{x}_t^\top \boldsymbol{\mu}_{t-1}\right) \beta_t$

Output: Weight vector $\boldsymbol{\mu}_T$ and confidence $\Sigma_T$.
```

SCW: Soft Confidence Weighted learning

- SCW (Wang et al. 2012)
  - Four salient properties
    - ☺ Large margin, Non-separable, Confidence weighted (2nd order), Adaptive margin
  - Formulation
    - SCW-I

$$(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu},\Sigma} D_{KL}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \| \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)\right) + C\ell^{\phi}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma); (\mathbf{x}_t, y_t)\right)$$

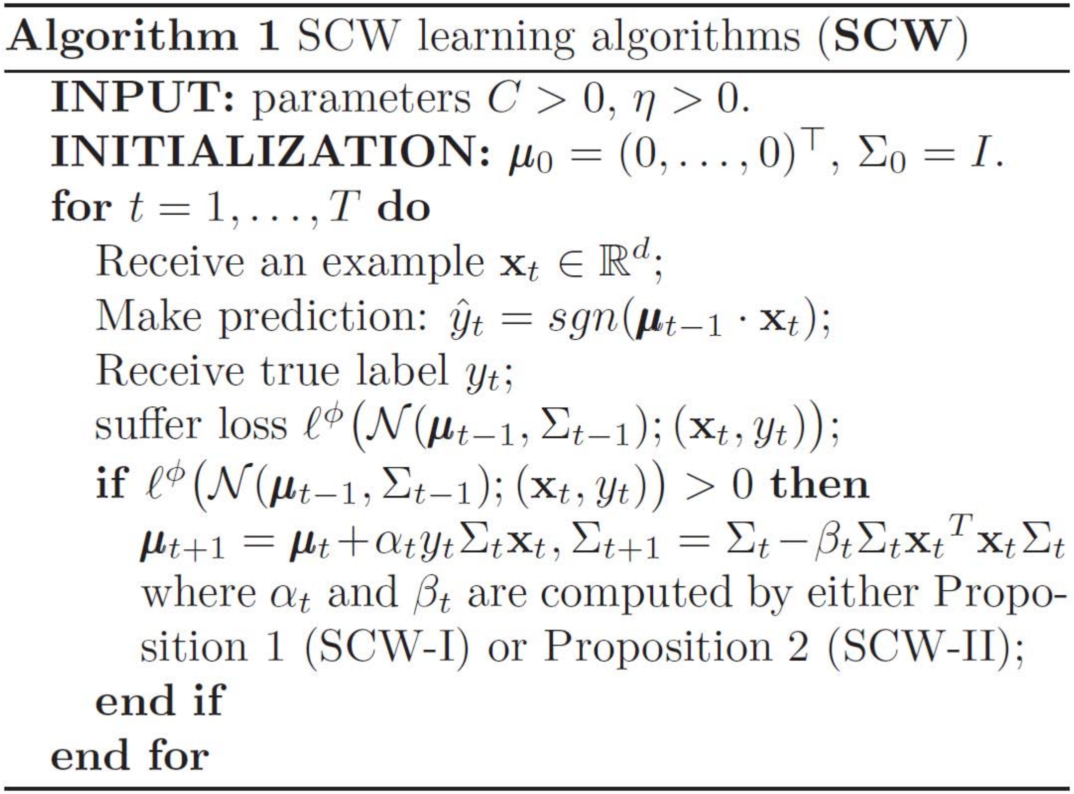
$$\ell^{\phi}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma); (\mathbf{x}_t, y_t)\right) = \max\left(0, \phi\sqrt{\mathbf{x}_t^{\top} \Sigma \mathbf{x}_t} - y_t \boldsymbol{\mu} \cdot \mathbf{x}_t\right)$$

    - SCW-II

$$(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu},\Sigma} D_{KL}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \| \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)\right) + C\ell^{\phi}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma); (\mathbf{x}_t, y_t)\right)^2$$

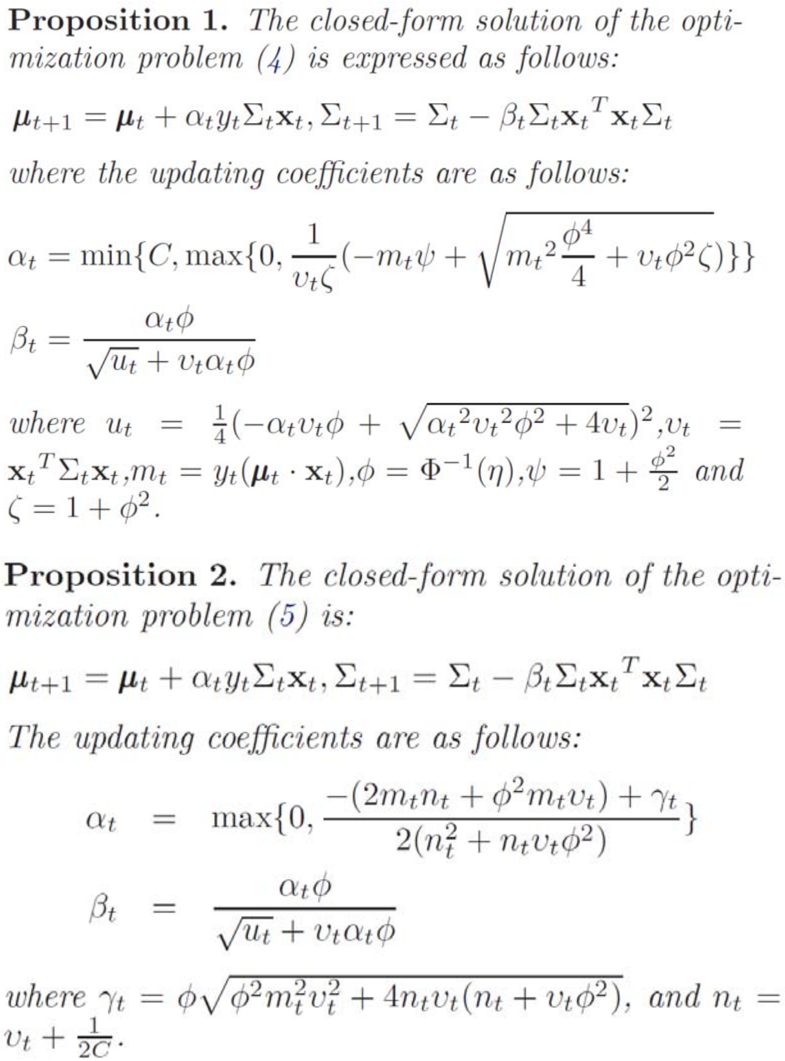
SCW: Soft Confidence Weighted learning

- SCW Algorithms



Algorithm 1 SCW learning algorithms (**SCW**)

```
INPUT: parameters $C > 0$, $\eta > 0$.
INITIALIZATION: $\boldsymbol{\mu}_0 = (0, \dots, 0)^\top$, $\Sigma_0 = I$.
for $t = 1, \dots, T$ do
    Receive an example $\mathbf{x}_t \in \mathbb{R}^d$;
    Make prediction: $\hat{y}_t = sgn(\boldsymbol{\mu}_{t-1} \cdot \mathbf{x}_t)$;
    Receive true label $y_t$;
    suffer loss $\ell^\phi(\mathcal{N}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}); (\mathbf{x}_t, y_t))$;
    if $\ell^\phi(\mathcal{N}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1}); (\mathbf{x}_t, y_t)) > 0$ then
        $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t y_t \Sigma_t \mathbf{x}_t, \Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t {\mathbf{x}_t}^T \mathbf{x}_t \Sigma_t$
        where $\alpha_t$ and $\beta_t$ are computed by either Proposition 1 (SCW-I) or Proposition 2 (SCW-II);
    end if
end for
```



Proposition 1. *The closed-form solution of the optimization problem (4) is expressed as follows:*

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t y_t \Sigma_t \mathbf{x}_t, \Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t {\mathbf{x}_t}^T \mathbf{x}_t \Sigma_t$$

where the updating coefficients are as follows:

$$\alpha_t = \min\{C, \max\{0, \frac{1}{v_t \zeta}(-m_t \psi + \sqrt{{m_t}^2 \frac{\phi^4}{4} + v_t \phi^2 \zeta})\}\}$$

$$\beta_t = \frac{\alpha_t \phi}{\sqrt{u_t} + v_t \alpha_t \phi}$$

where $u_t = \frac{1}{4}(-\alpha_t v_t \phi + \sqrt{{\alpha_t}^2 {v_t}^2 \phi^2 + 4 v_t})^2$,$v_t = {\mathbf{x}_t}^T \Sigma_t \mathbf{x}_t$,$m_t = y_t(\boldsymbol{\mu}_t \cdot \mathbf{x}_t)$,$\phi = \Phi^{-1}(\eta)$,$\psi = 1 + \frac{\phi^2}{2}$ *and* $\zeta = 1 + \phi^2$.

Proposition 2. *The closed-form solution of the optimization problem (5) is:*

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t y_t \Sigma_t \mathbf{x}_t, \Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t {\mathbf{x}_t}^T \mathbf{x}_t \Sigma_t$$

The updating coefficients are as follows:

$$\alpha_t = \max\{0, \frac{-(2 m_t n_t + \phi^2 m_t v_t) + \gamma_t}{2(n_t^2 + n_t v_t \phi^2)}\}$$

$$\beta_t = \frac{\alpha_t \phi}{\sqrt{u_t} + v_t \alpha_t \phi}$$

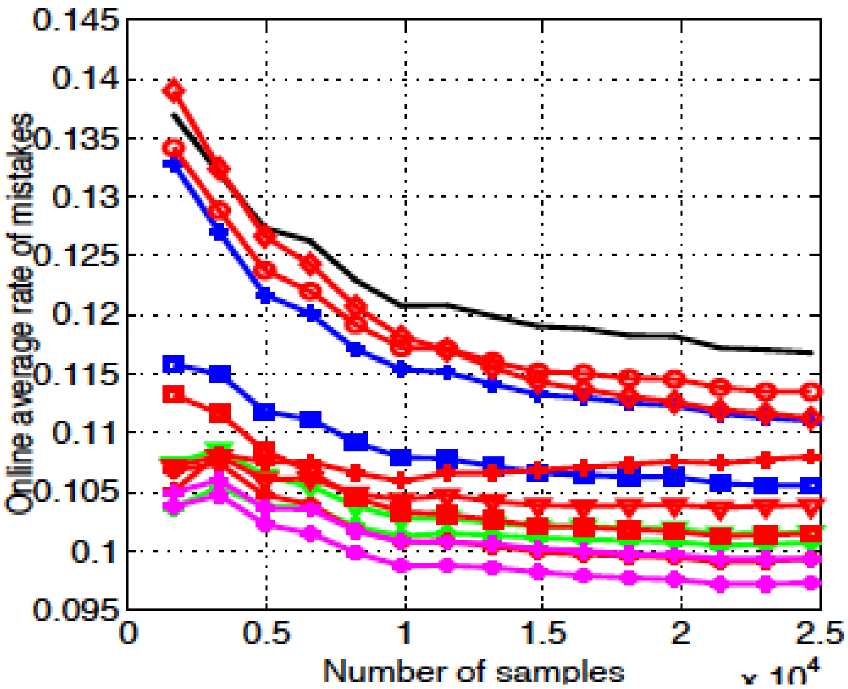
where $\gamma_t = \phi \sqrt{\phi^2 m_t^2 v_t^2 + 4 n_t v_t (n_t + v_t \phi^2)}$, *and* $n_t = v_t + \frac{1}{2C}$.

Traditional Linear Online Learning (cont')

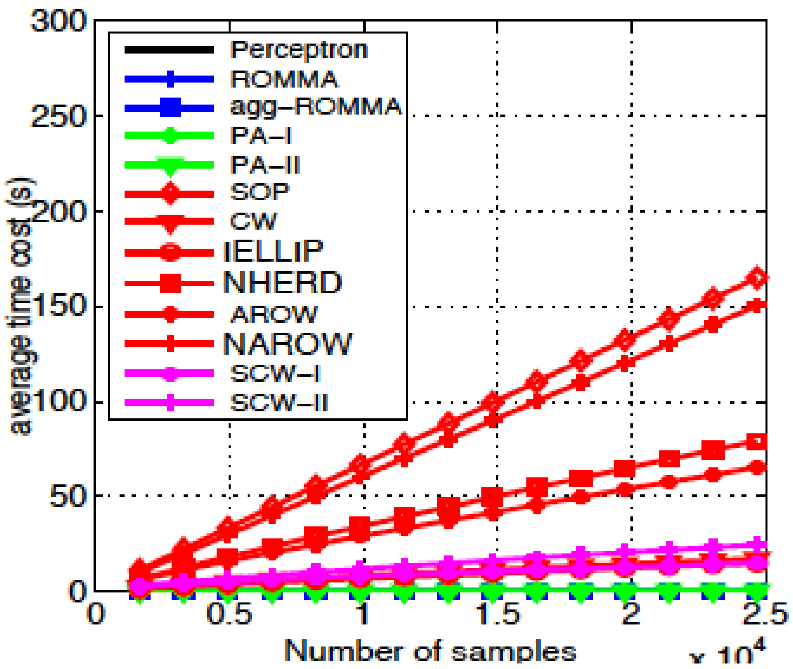
- **Second-Order** Methods
  - Learn both **first order** and **second order** info
- Pros and Cons
  - ☺ Faster convergence rate
  - ☹ Expensive for high-dimensional data
  - ☹ Relatively sensitive to noise

Traditional Linear Online Learning (cont')

- Empirical Results (Wang et al., ICML'12)



Online Mistake Rate



Online Time Cost

Sparse Online Learning

- Motivation
  - How to induce **Sparsity** in the weights of online learning algorithms for high-dimensional data
  - Space constraints (memory overflow)
  - Test-time constraints (test computational cost)
- Some popular existing work
  - **Truncated gradient** (Langford et al., 2009)
  - FOBOS: Forward Looking Subgradients (Duchi and Singer 2009)
  - Bayesian sparse online learning (Balakrishnan and Madigan 2008)
  - etc.

Truncated gradient (Langford et al., 2009)

- Main Idea
  - **Truncated gradient**: impose sparsity by modifying the stochastic gradient descent

$$\sum_{i=1}^{t} L(w_i, z_i) \qquad z_i = (x_i, y_i)$$

- Stochastic Gradient Descent

$$f(w_i) = w_i - \eta \nabla_1 L(w_i, z_i)$$

- Simple Coefficient Rounding

$$f(w_i) = T_0(w_i - \eta \nabla_1 L(w_i, z_i), \theta)$$

Truncated gradient (Langford et al., 2009)

Simple Coefficient Rounding vs. Less aggregative truncation

$$T_0(v_j,\theta) = \begin{cases} 0 & \text{if } |v_j| \le \theta \\ v_j & \text{otherwise} \end{cases}$$

$$T_1(v_j,\alpha,\theta) = \begin{cases} \max(0, v_j - \alpha) & \text{if } v_j \in [0,\theta] \\ \min(0, v_j + \alpha) & \text{if } v_j \in [-\theta, 0] \\ v_j & \text{otherwise} \end{cases}$$

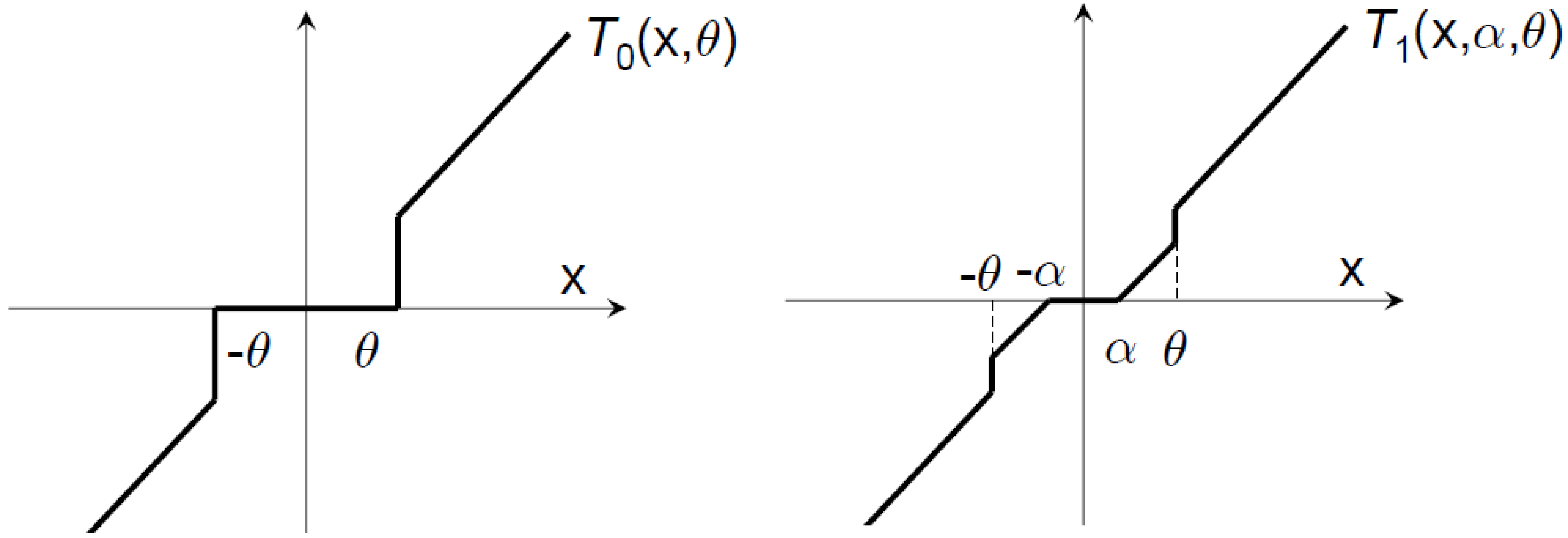
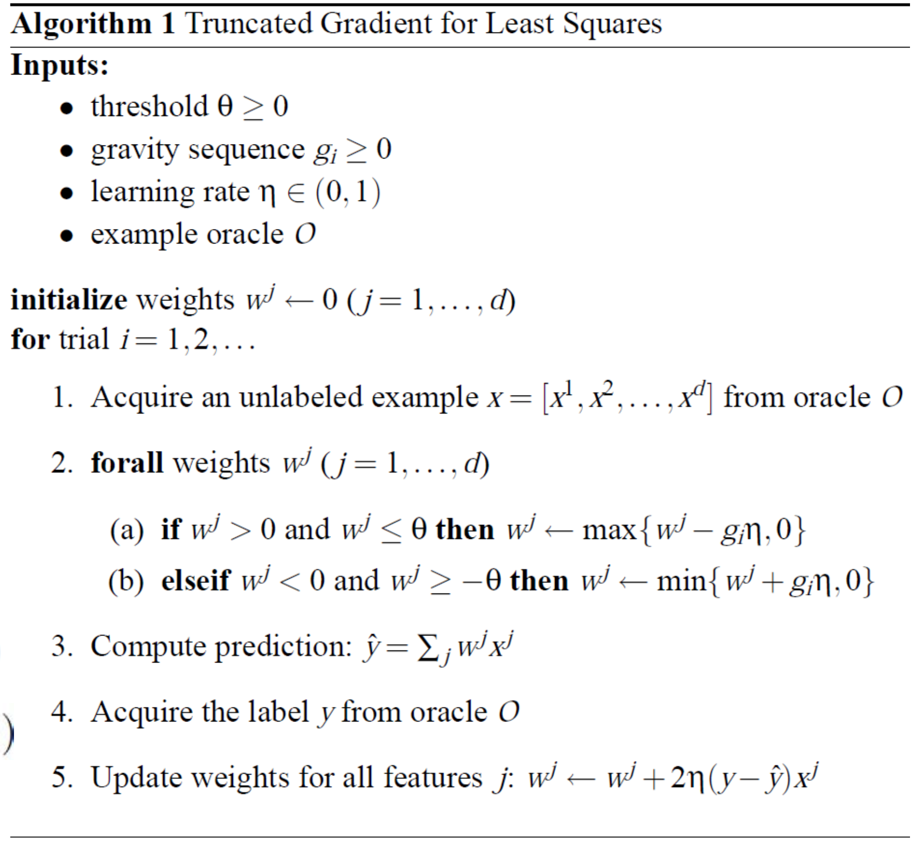


Illustration of the two truncation functions, T0 and T1

Truncated gradient (Langford et al., 2009)

$$f(w_i) = T_1(w_i - \eta \nabla_1 L(w_i, z_i), \eta g_i, \theta)$$

- The amount of shrinkage is measured by a *gravity* parameter $g_i > 0$
- The truncation can be performed every K online steps
- When $g_i = 0$ the update rule is identical to the standard SGD
- Loss Functions: $L(w,z) = \phi(w^T x, y)$
  - Logistic $\phi(p,y) = \ln(1+\exp(-py))$
  - SVM (hinge) $\phi(p,y) = \max(0, 1-py)$
  - Least Square $\phi(p,y) = (p-y)^2$

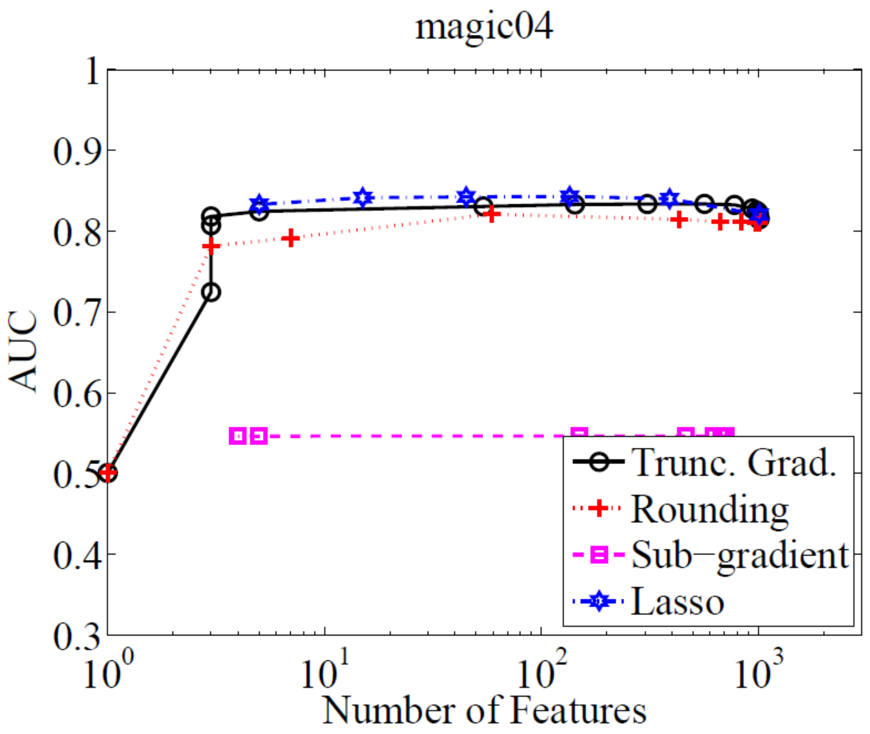


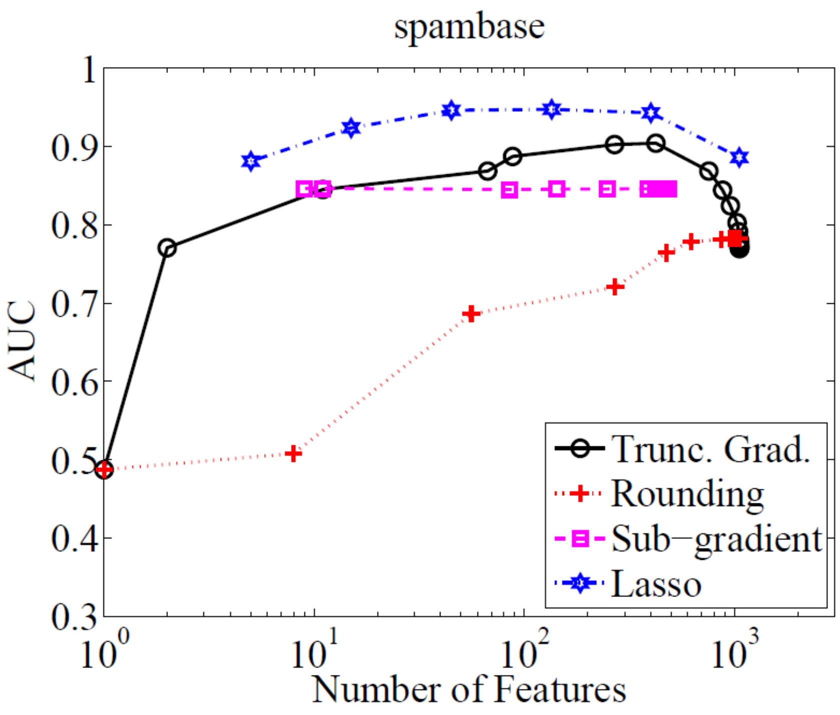
Algorithm 1 Truncated Gradient for Least Squares

```
Inputs:
  • threshold $\theta \geq 0$
  • gravity sequence $g_i \geq 0$
  • learning rate $\eta \in (0,1)$
  • example oracle $O$
initialize weights $w^j \leftarrow 0$ $(j = 1, \ldots, d)$
for trial $i = 1, 2, \ldots$
  1. Acquire an unlabeled example $x = [x^1, x^2, \ldots, x^d]$ from oracle $O$
  2. forall weights $w^j$ $(j = 1, \ldots, d)$
       (a) if $w^j > 0$ and $w^j \leq \theta$ then $w^j \leftarrow \max\{w^j - g_i\eta, 0\}$
       (b) elseif $w^j < 0$ and $w^j \geq -\theta$ then $w^j \leftarrow \min\{w^j + g_i\eta, 0\}$
  3. Compute prediction: $\hat{y} = \sum_j w^j x^j$
  4. Acquire the label $y$ from oracle $O$
  5. Update weights for all features $j$: $w^j \leftarrow w^j + 2\eta(y - \hat{y})x^j$
```

Truncated gradient (Langford et al., 2009)

- Comparison to other baselines





Variants of Sparse Online Learning

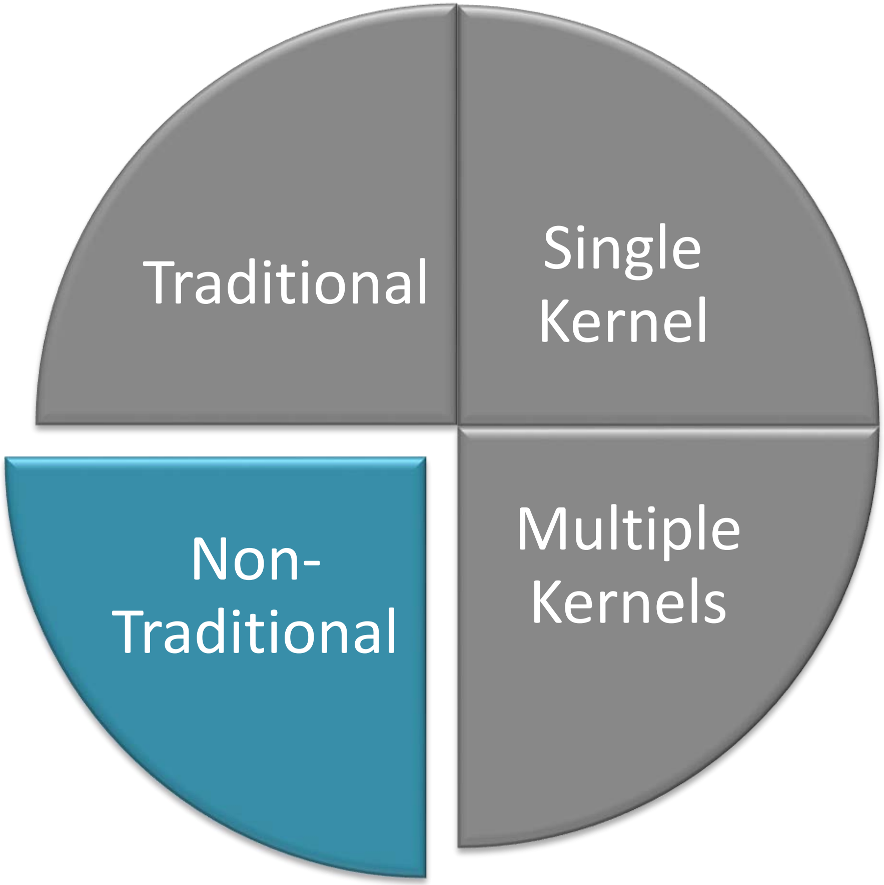
- Online Feature Selection (OFS)
  - A variant of Sparse Online Learning
  - The key difference is that OFS focuses on selecting a fixed subset of features in online learning process
  - Could be used as an alternative tool for batch feature selection when dealing with big data
- Existing Work
  - Online Feature Selection (Hoi et al, 2012) proposed an OFS scheme by exploring the Sparse Projection to choose a fixed set of active features in online learning

Summary of Traditional Linear OL

- Pros
  - ☺ Efficient for computation & memory
  - ☺ Extremely scalable
  - ☺ Theoretical bounds on the mistake rate
- Cons
  - ☹ Learn **Linear** prediction models
  - ☹ Optimize the **mistake rate** only

Online Learning: Overview

Linear Methods



Non-Linear Methods

Non-Traditional Linear OL

- Online AUC Maximization
- Cost-Sensitive Online Learning
- Online Transfer Learning

Online AUC Maximization

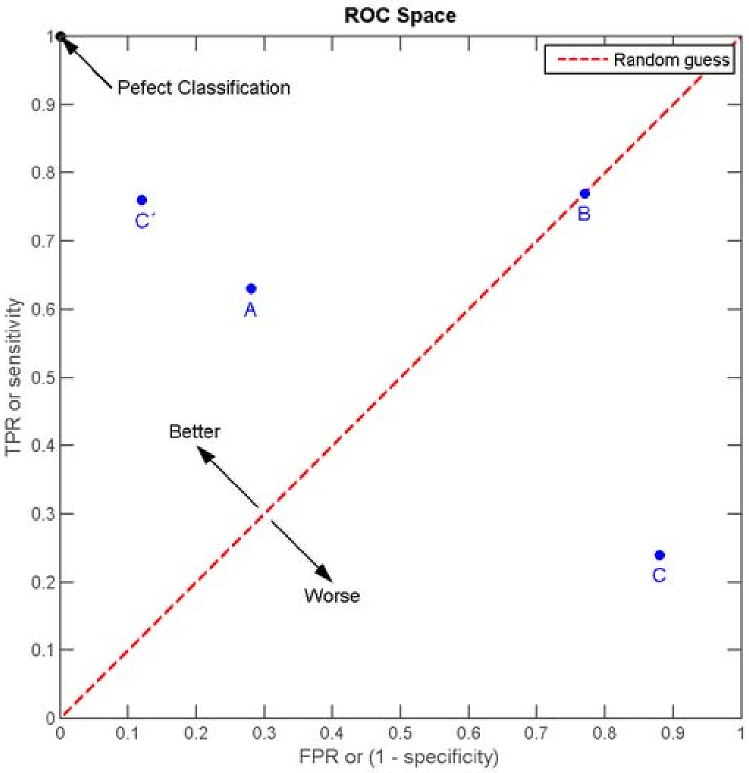
- Motivation
  - The **mistake rate** (or classification accuracy) measure could be misleading for many real-world applications
- Example:

  Consider a set of **10,000** instances with only **10** "**positive**" and **9,990** "**negative**". A naïve classifier that simply declares every instance as "**negative**" has **99.9%** accuracy.
- Many applications (e.g., anomaly detection) often adopt other metrics, e.g., AUC (area under the ROC curve).

Can online learning directly optimize AUC?

Online AUC Maximization

- What is AUC?
  - **AUC** (Area Under the ROC Curve)
  - **ROC** (Receiver Operating Characteristic) curve details the rate of **True Positives (TP)** against **False Positives (FP)** over the range of possible thresholds.
  - AUC measures the probability for a randomly drawn **positive** instance to have a **higher** decision value than a randomly sampled **negative** instance
  - ROC was first used in World War II for the analysis of radar signals.



Online AUC Maximization

- Motivation
  - To develop an online learning algorithm for training an online classifier to maximize the AUC metric instead of mistake rate/accuracy
  - "Online AUC Maximization" (Zhao et al., ICML'11)
- Key Challenge
  - In math, AUC is expressed as a **sum of pairwise losses** between instances from different classes, which is **quadratic** in the number of received training examples
  - Hard to directly solve the AUC optimization efficiently

Formulation

- A data set $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{-1, +1\} \mid i \in [T]\}$
- Positive instances $\mathcal{D}_+ = \{(\mathbf{x}_i^+, +1) \mid i \in [T_+]\}$
- Negative instances $\mathcal{D}_- = \{(\mathbf{x}_j^-, -1) \mid j \in [T_-]\}$
- Given a classifier **w**, its **AUC** on the dataset D:

$$\mathrm{AUC}(\mathbf{w}) = \frac{\sum_{i=1}^{T_+}\sum_{j=1}^{T_-} \mathbb{I}_{(\mathbf{w}\cdot\mathbf{x}_i^+ > \mathbf{w}\cdot\mathbf{x}_j^-)}}{T_+T_-} = 1 - \frac{\boxed{\sum_{i=1}^{T_+}\sum_{j=1}^{T_-} \mathbb{I}_{(\mathbf{w}\cdot\mathbf{x}_i^+ \le \mathbf{w}\cdot\mathbf{x}_j^-)}}}{T_+T_-}$$

$$\max_w \mathrm{AUC}(\mathbf{w}) \Leftrightarrow \min_w \sum_{i=1}^{T_+}\sum_{j=1}^{T_-} \mathbb{I}_{(\mathbf{w}\cdot\mathbf{x}_i^+ - \mathbf{w}\cdot\mathbf{x}_j^- \le 0)}$$

Formulation (cont’)

- Replace the indicator function $\mathbb{I}$ with its convex surrogate, i.e., the hinge loss function

$$\ell(\mathbf{w}, \mathbf{x}_i^+ - \mathbf{x}_j^-) = \max\{0, 1 - \mathbf{w} \cdot (\mathbf{x}_i^+ - \mathbf{x}_j^-)\}$$

- Find the optimal classifier **w** by minimizing

$$\boxed{\mathcal{R}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{T_+}\sum_{j=1}^{T_-}\ell(\mathbf{w}, \mathbf{x}_i^+ - \mathbf{x}_j^-)} \qquad \textbf{(1)}$$

- It is not difficult to show that

$$\mathcal{R}(\mathbf{w}) \geq \frac{1}{2}\|\mathbf{w}\|_2^2 + CT_+T_-(1 - \mathrm{AUC}(\mathbf{w}))$$

Formulation (cont')

- Re-writing objective function **(1)** into:

$$\boxed{\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{T_+}\sum_{j=1}^{T_-}\ell(\mathbf{w},\mathbf{x}_i^+ - \mathbf{x}_j^-)}$$

$$\mathcal{L}_t(\mathbf{w};x_t,y_t) = \mathbb{I}_{(y_t=+1)}\boxed{h_+^t(\mathbf{w})} + \mathbb{I}_{(y_t=-1)}\boxed{h_-^t(\mathbf{w})}$$

$$\underline{h_+^t(\mathbf{w}) \quad = \quad \sum_{t'=1}^{t-1}\mathbb{I}_{(y_{t'}=-1)}\ell(\mathbf{w},\mathbf{x}_t - \mathbf{x}_{t'})} \qquad \underline{h_-^t(\mathbf{w}) \quad = \quad \sum_{t'=1}^{t-1}\mathbb{I}_{(y_{t'}=+1)}\ell(\mathbf{w},\mathbf{x}_{t'} - \mathbf{x}_t)}$$

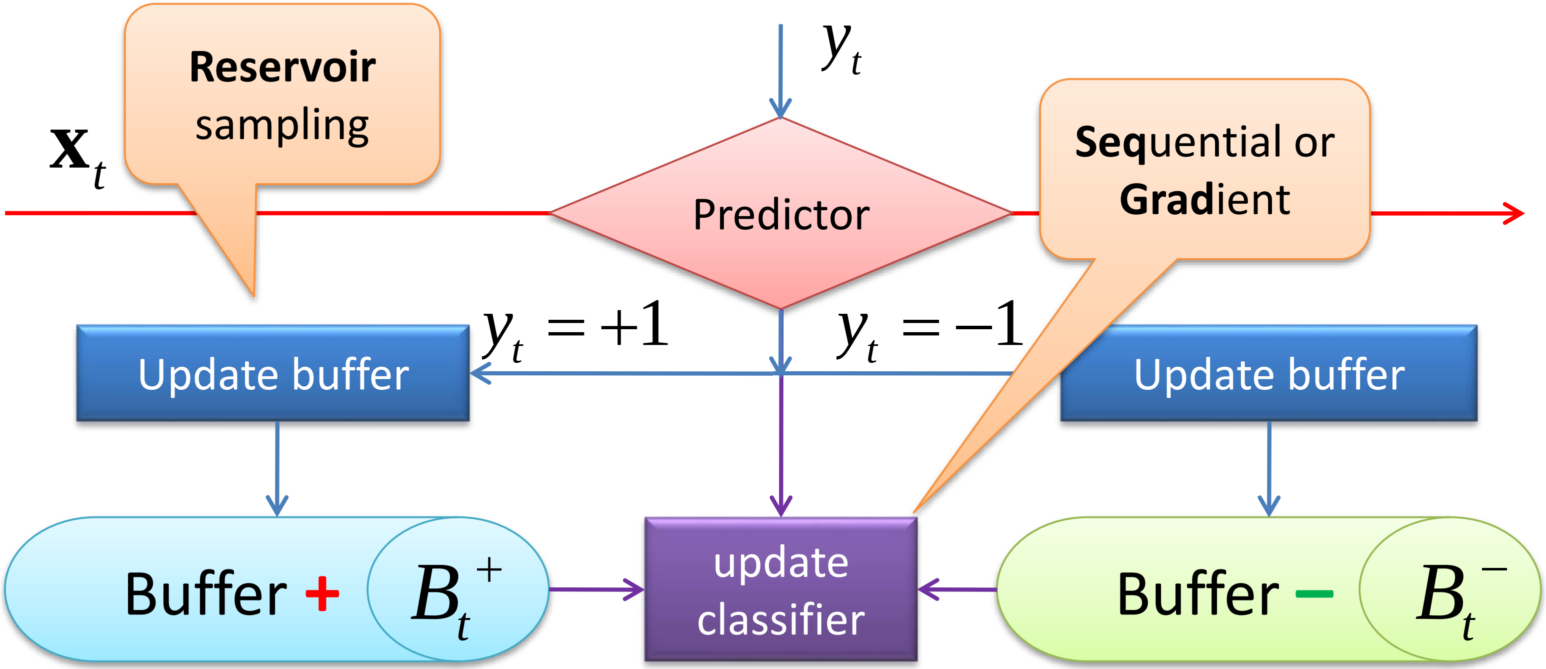
- In online learning task, given (x_t, y_t), we may do online update:

$$\mathbf{w}_{t+1} \quad = \quad \mathbf{w}_t \; - \; C\nabla\mathcal{L}_t(\mathbf{w})$$

The loss function is related to **all** received examples.
Have to **store all the received training examples!!**

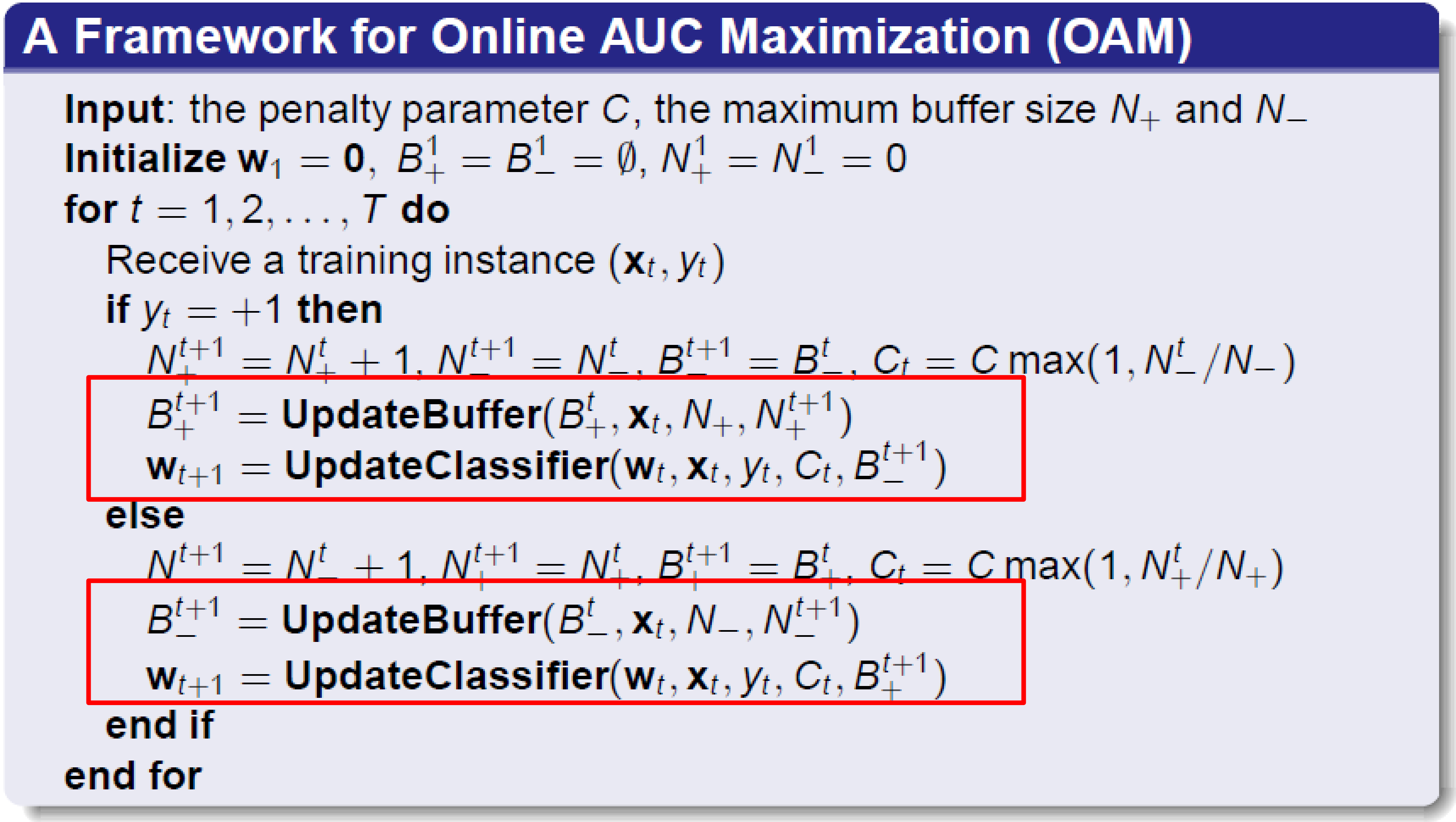
Main Idea of OAM

- **Cache** a **small** number of received examples;
- Two **buffers** of fixed size, B_t^+ *and* B_t^-, *to* cache the positive and negative instances;



Flow of the proposed online AUC maximization process

OAM Framework

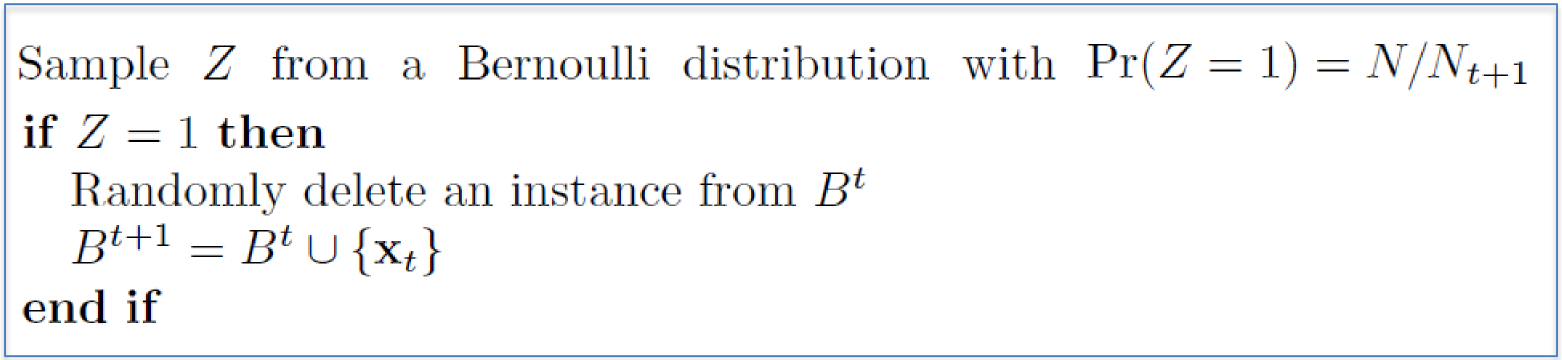


A Framework for Online AUC Maximization (OAM)

```
**Input**: the penalty parameter $C$, the maximum buffer size $N_+$ and $N_-$
**Initialize** $\mathbf{w}_1 = \mathbf{0},\ B_+^1 = B_-^1 = \emptyset, N_+^1 = N_-^1 = 0$
**for** $t = 1, 2, \ldots, T$ **do**
    Receive a training instance $(\mathbf{x}_t, y_t)$
    **if** $y_t = +1$ **then**
        $N_+^{t+1} = N_+^t + 1, N_-^{t+1} = N_-^t, B_-^{t+1} = B_-^t, C_t = C\max(1, N_-^t/N_-)$
        $B_+^{t+1} = \textbf{UpdateBuffer}(B_+^t, \mathbf{x}_t, N_+, N_+^{t+1})$
        $\mathbf{w}_{t+1} = \textbf{UpdateClassifier}(\mathbf{w}_t, \mathbf{x}_t, y_t, C_t, B_-^{t+1})$
    **else**
        $N_-^{t+1} = N_-^t + 1, N_+^{t+1} = N_+^t, B_+^{t+1} = B_+^t, C_t = C\max(1, N_+^t/N_+)$
        $B_-^{t+1} = \textbf{UpdateBuffer}(B_-^t, \mathbf{x}_t, N_-, N_-^{t+1})$
        $\mathbf{w}_{t+1} = \textbf{UpdateClassifier}(\mathbf{w}_t, \mathbf{x}_t, y_t, C_t, B_+^{t+1})$
    **end if**
**end for**
```

Update Buffer

- **Reservoir Sampling** (J. S. Vitter, 1985)
  - A family of classical sampling algorithms for randomly choosing k samples from a data set with n items, where n is either a very large or unknown number.
  - In general, it takes a **random** sample set of the **desired size** in only **one pass** over the underlying dataset.
  - The UpdateBuffer algorithm is simple and very efficient:



```
Sample $Z$ from a Bernoulli distribution with $\Pr(Z = 1) = N/N_{t+1}$
if $Z = 1$ then
    Randomly delete an instance from $B^t$
    $B^{t+1} = B^t \cup \{\mathbf{x}_t\}$
end if
```

Update Classifier

- Algorithm 1: Sequential update by PA:
  - Follow the idea of Passive aggressive learning (Crammer et al.'06)
  - For each **x** in buffer B, update the classifier:

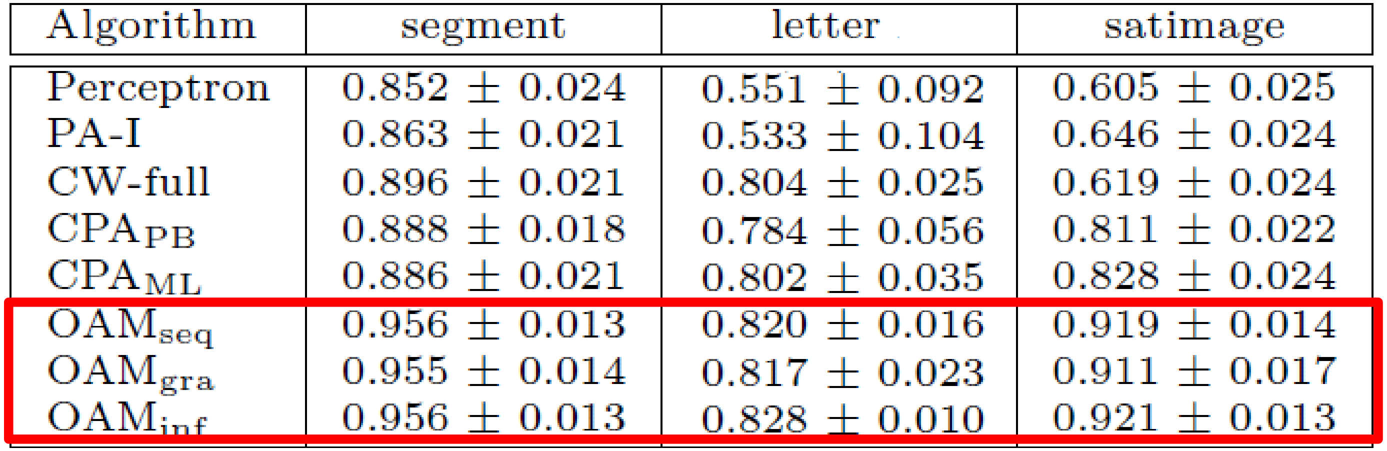
$$\mathbf{w}^{i+1} = \arg\min_{\mathbf{w}} |\mathbf{w} - \mathbf{w}^{i}|_2^2 + C_t \ell(\mathbf{w}, y_t(\mathbf{x}_t - \mathbf{x}))$$

- Algorithm 2: Gradient-based update
  - Follow the idea of online gradient descent
  - For each **x** in buffer B, update the classifier:

```
if $y_t \mathbf{w}_t \cdot (\mathbf{x}_t - \mathbf{x}) \leq 1$ then
    $\mathbf{w}_{t+1} = \mathbf{w}_{t+1} + C_t y_t (\mathbf{x}_t - \mathbf{x})/2$
end if
```

Empirical Results of OAM

- Comparisons
  - Traditional algorithms:
    - Perceptron, PA, Cost-sensitive PA (CPA), CW
  - The proposed OAM algorithms:
    (i) OAM-seq, OAM-gra, (ii) OAM-inf (infinite buffer size)
- Evaluation of AUC for Classification tasks



Algorithm	segment	letter	satimage
Perceptron	0.852 ± 0.024	0.551 ± 0.092	0.605 ± 0.025
PA-I	0.863 ± 0.021	0.533 ± 0.104	0.646 ± 0.024
CW-full	0.896 ± 0.021	0.804 ± 0.025	0.619 ± 0.024
$\mathrm{CPA}_{\mathrm{PB}}$	0.888 ± 0.018	0.784 ± 0.056	0.811 ± 0.022
$\mathrm{CPA}_{\mathrm{ML}}$	0.886 ± 0.021	0.802 ± 0.035	0.828 ± 0.024
$\mathrm{OAM}_{\mathrm{seq}}$	0.956 ± 0.013	0.820 ± 0.016	0.919 ± 0.014
$\mathrm{OAM}_{\mathrm{gra}}$	0.955 ± 0.014	0.817 ± 0.023	0.911 ± 0.017
$\mathrm{OAM}_{\mathrm{inf}}$	0.956 ± 0.013	0.828 ± 0.010	0.921 ± 0.013

Cost-Sensitive Online Learning

- Motivation
  - Beyond optimizing the mistake rate or accuracy
  - Attempt to optimize the **cost-sensitive** measures
    - **Sum** $sum = \eta_p \times sensitivity + \eta_n \times specificity$

$$sensitivity = \frac{T_p - M_p}{T_p}, \quad specificity = \frac{T_n - M_n}{T_n}, \quad \begin{array}{c} \eta_p + \eta_n = 1 \\ 0 \le \eta_p, \eta_n \le 1 \end{array}$$

    - **Cost** $cost = c_p \times M_p + c_n \times M_n$

$$0 \le c_p, c_n \le 1 \quad c_p + c_n = 1$$

- Existing Work
  - Cost-sensitive Online Gradient Descent (Wang et al. 2012)
  - Cost-Sensitive Double Updating Online Learning (Zhao et al. 2013)

Cost-Sensitive Online Learning

- Our goal is to design online learning algorithms to optimize the cost-sensitive metrics directly
- **Proposition 1:**
  - Consider a cost-sensitive classification problem, the goal of maximizing the weighted sum or minimizing the weighted cost is equivalent to minimizing the following objective:

$$\rho \sum_{y_t=+1} \mathrm{I}_{(y_t f(\mathbf{x}_t)<0)} + \sum_{y_t=-1} \mathrm{I}_{(y_t f(\mathbf{x}_t)<0)}$$

  - where $\rho = \frac{\eta_p T_n}{\eta_n T_p}$ for the maximization of the **weighted sum**;
  - and $\rho = \frac{c_p}{c_n}$ for the minimization of the **cost.**

Cost-Sensitive Online Learning

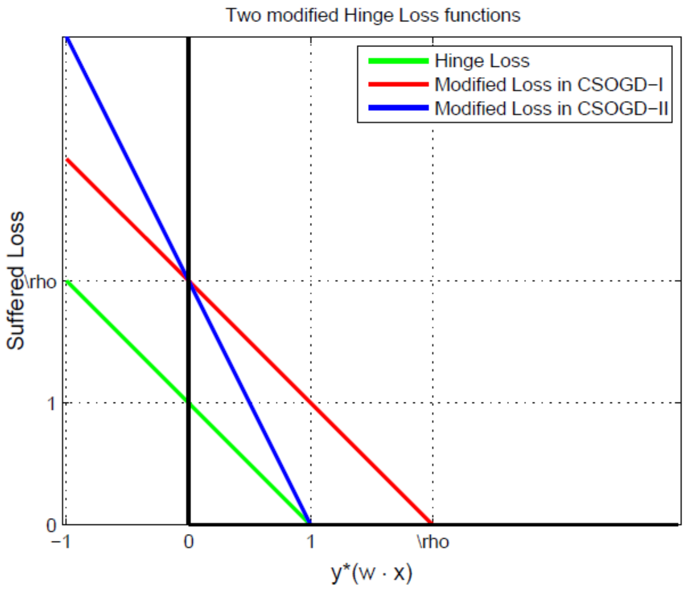
- Convex Relaxation
  - Modified Hinge Loss:

Replace the indicator function with modified hinge loss:

$$
\begin{aligned}
\ell^{I}(\mathbf{w};(\mathbf{x},y)) &= \max(0,(\rho * \mathrm{I}_{(y=1)} + \mathrm{I}_{(y=-1)}) - y(\mathbf{w}\cdot\mathbf{x})) \\
\ell^{II}(\mathbf{w};(\mathbf{x},y)) &= (\rho * \mathrm{I}_{(y=1)}+\mathrm{I}_{(y=-1)})*\max(0,1-y(\mathbf{w}\cdot\mathbf{x}))
\end{aligned}
$$

- sum $\rho = \frac{\eta_p T_n}{\eta_n T_p}$

- cost $\rho = \frac{c_p}{c_n}$



Cost-Sensitive Online Gradient Descent

- CSOGD: Cost-Sensitive Online Gradient Descent
  - Cost-sensitive objective functions

$$\mathcal{F}_T^*(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{t=1}^{T} \ell^*(\mathbf{w};(\mathbf{x}_t, y_t)) \quad * \in \{I, II\}$$

$$\begin{aligned} \ell^I(\mathbf{w};(\mathbf{x},y)) &= \max(0, (\rho * \mathrm{I}_{(y=1)} + \mathrm{I}_{(y=-1)}) - y(\mathbf{w}\cdot\mathbf{x})) \\ \ell^{II}(\mathbf{w};(\mathbf{x},y)) &= (\rho * \mathrm{I}_{(y=1)} + \mathrm{I}_{(y=-1)}) * \max(0, 1 - y(\mathbf{w}\cdot\mathbf{x})) \end{aligned}$$

    - where $\rho = \frac{\eta_p T_n}{\eta_n T_p}$ for optimizing sum or $\rho = \frac{c_p}{c_n}$ for cost
  - Update by Online Gradient Descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \nabla \ell_t(\mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + \lambda y_t \mathbf{x}_t & \text{if } \ell_t(\mathbf{w}_t) > 0 \\ \mathbf{w}_t & \text{otherwise} \end{cases} \qquad \mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + \lambda \rho_t y_t \mathbf{x}_t & \text{if } \ell_t(\mathbf{w}_t) > 0 \\ \mathbf{w}_t & \text{otherwise} \end{cases}$$

Cost-Sensitive Online Gradient Descent

- CSOGD: Cost-Sensitive Online Gradient Descent
  - Formulate the cost-sensitive objective functions

$$\mathcal{F}_T^*(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{t=1}^{T} \ell^*(\mathbf{w};(\mathbf{x}_t, y_t)) \quad * \in \{I, II\}$$

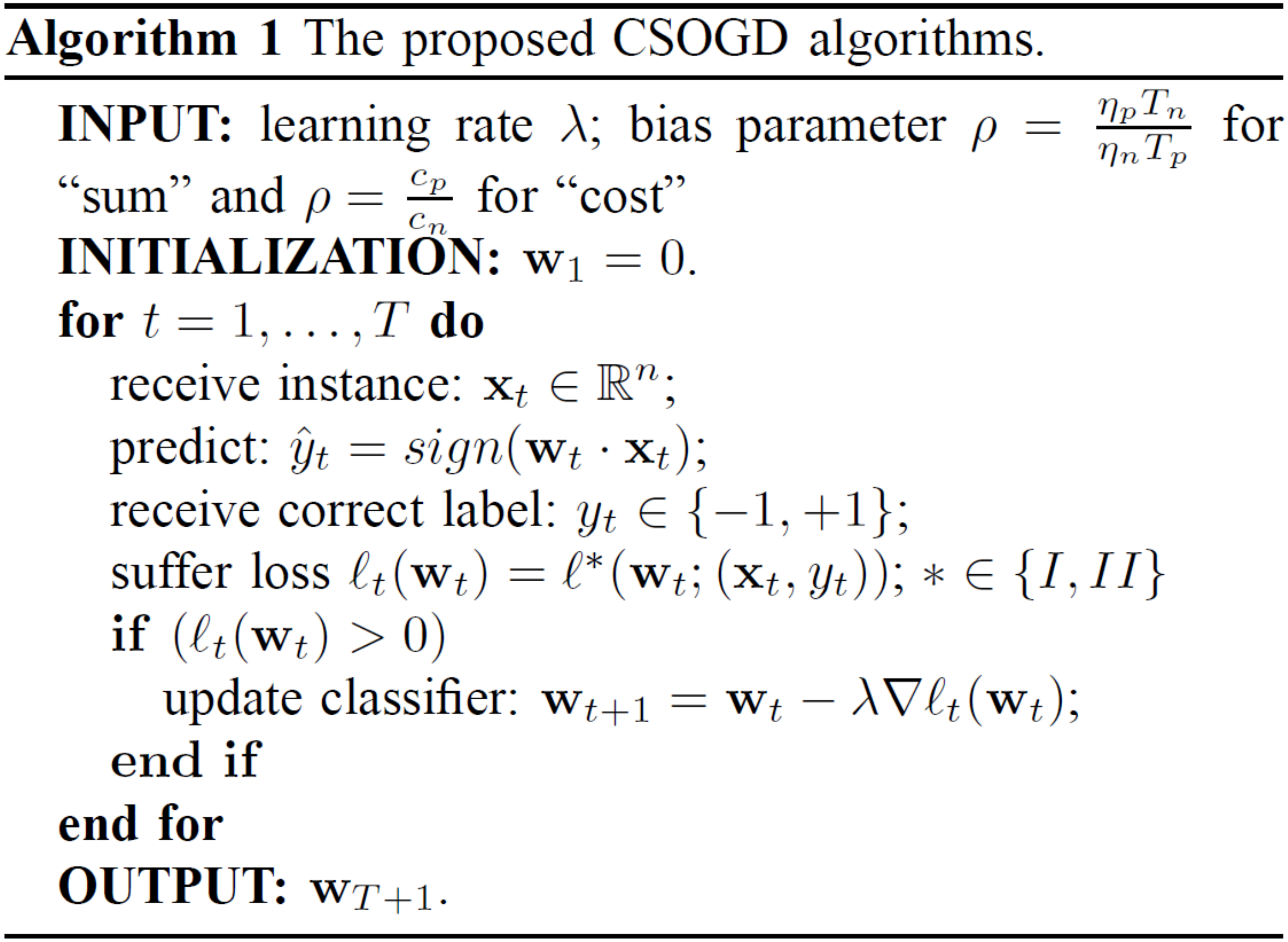
$$\begin{aligned} \ell^I(\mathbf{w};(\mathbf{x},y)) &= \max(0, (\rho * \mathrm{I}_{(y=1)} + \mathrm{I}_{(y=-1)}) - y(\mathbf{w}\cdot\mathbf{x})) \\ \ell^{II}(\mathbf{w};(\mathbf{x},y)) &= (\rho * \mathrm{I}_{(y=1)} + \mathrm{I}_{(y=-1)}) * \max(0, 1 - y(\mathbf{w}\cdot\mathbf{x})) \end{aligned}$$

    - where $\rho = \frac{\eta_p T_n}{\eta_n T_p}$ for optimizing sum or $\rho = \frac{c_p}{c_n}$ for cost
  - Update by Online Gradient Descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \nabla \ell_t(\mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + \lambda y_t \mathbf{x}_t & \text{if } \ell_t(\mathbf{w}_t) > 0 \\ \mathbf{w}_t & \text{otherwise} \end{cases} \qquad \mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + \lambda \rho_t y_t \mathbf{x}_t & \text{if } \ell_t(\mathbf{w}_t) > 0 \\ \mathbf{w}_t & \text{otherwise} \end{cases}$$

Cost-Sensitive Online Gradient Descent

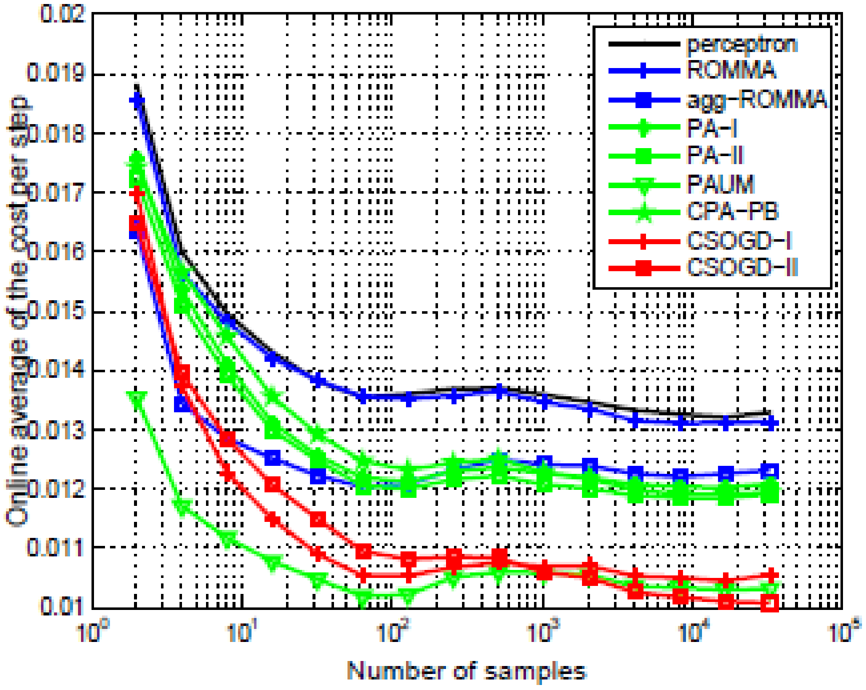


Algorithm 1 The proposed CSOGD algorithms.

```
INPUT: learning rate $\lambda$; bias parameter $\rho = \frac{\eta_p T_n}{\eta_n T_p}$ for "sum" and $\rho = \frac{c_p}{c_n}$ for "cost"
INITIALIZATION: $\mathbf{w}_1 = 0$.
for $t = 1, \ldots, T$ do
    receive instance: $\mathbf{x}_t \in \mathbb{R}^n$;
    predict: $\hat{y}_t = sign(\mathbf{w}_t \cdot \mathbf{x}_t)$;
    receive correct label: $y_t \in \{-1, +1\}$;
    suffer loss $\ell_t(\mathbf{w}_t) = \ell^*(\mathbf{w}_t; (\mathbf{x}_t, y_t))$; $* \in \{I, II\}$
    if $(\ell_t(\mathbf{w}_t) > 0)$
        update classifier: $\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda \nabla \ell_t(\mathbf{w}_t)$;
    end if
end for
OUTPUT: $\mathbf{w}_{T+1}$.
```

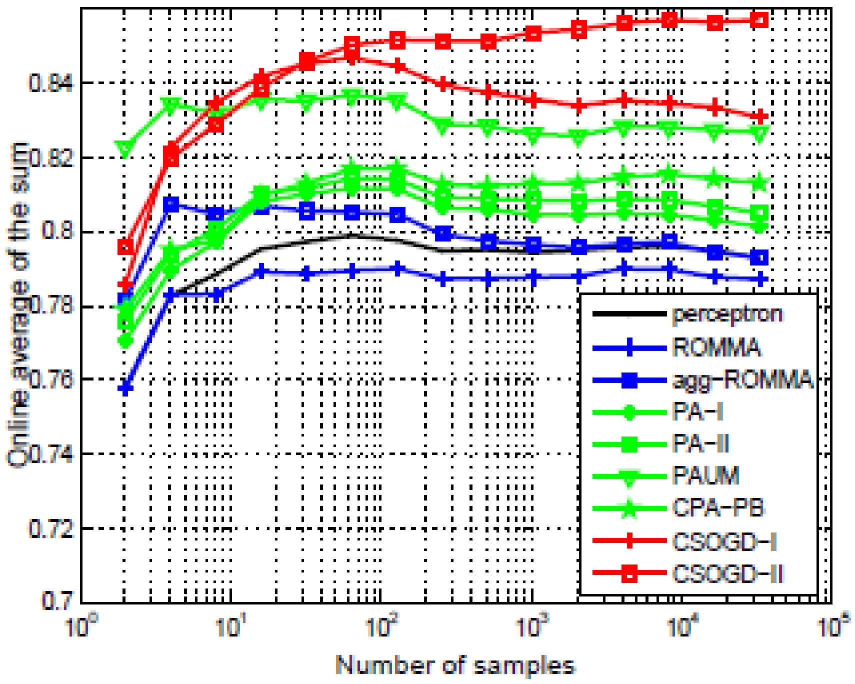
Cost-Sensitive Online Classification

Cost



(f) *w8a*

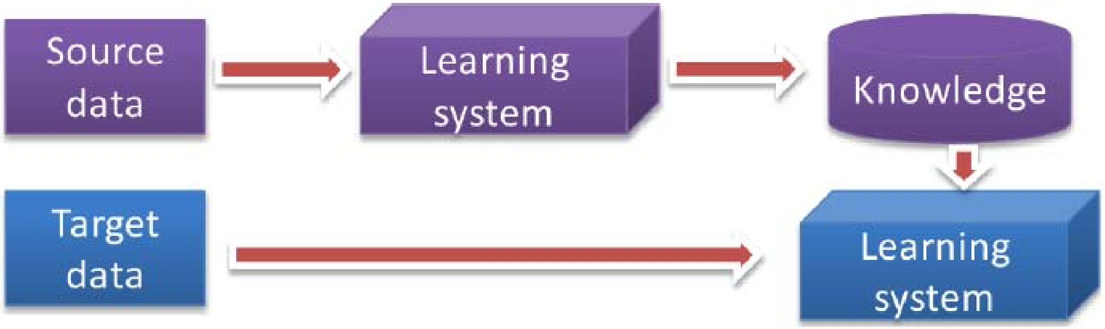
Sum

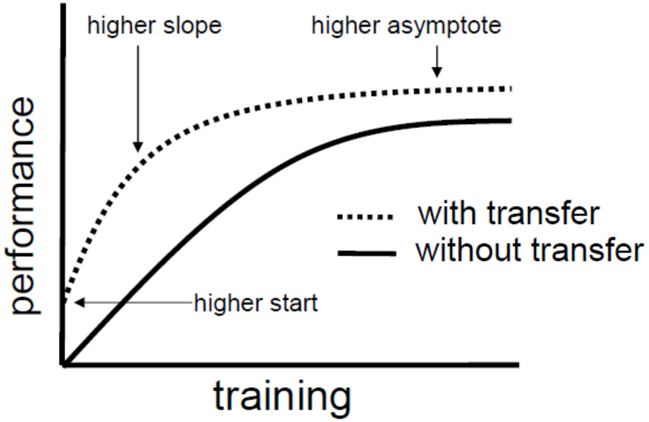


(f) *w8a*

Online Transfer Learning

- Transfer learning (TL)
  – Extract knowledge from one or more source tasks and then apply them to solve target tasks





  – Three ways which transfer might improve learning
  – Two Types of TL tasks
    - Homogeneous vs Heterogeneous TL

Online Transfer Learning (Zhao and Hoi 2011)

- Online Transfer learning (OTL)
  - Assume training data for target domain arrives sequentially
  - Assume a classifier was learnt from a source domain
  - online algorithms for transferring knowledge from source domain to target domain
- Settings
  - Old/source data space: $\mathcal{X}_1 \times \mathcal{Y}_1, \mathcal{X}_1 = \mathbb{R}^m \text{ and } \mathcal{Y}_1 = \{-1, +1\}$
  - New/target domain: $\mathcal{X}_2 \times \mathcal{Y}_2$
  - A sequence of examples from new/target domain

$$\{\mathbf{x}_t^{(2)}, y_t^{(2)} | t = 1, \ldots\}$$

  - OTL on Homogeneous domains $\mathcal{X}_1 = \mathcal{X}_2$
  - OTL on heterogeneous domains $\mathcal{X}_1 \neq \mathcal{X}_2$

Online Transfer Learning (Zhao and Hoi 2011)

- OTL on Homogeneous domains $\mathcal{X}_1 = \mathcal{X}_2$
  - Key Ideas: explore ensemble learning by combining both source and target classifiers

$$\hat{y}_t = \operatorname{sign}\left(\alpha_{1,t}\Pi(\mathbf{v}^\top \mathbf{x}_t) + \alpha_{2,t}\Pi(\mathbf{w}_t^\top \mathbf{x}_t) - \frac{1}{2}\right)$$

$$\alpha_{1,t+1} = \frac{\alpha_{1,t}s_t(\mathbf{v})}{\alpha_{1,t}s_t(\mathbf{v}) + \alpha_{2,t}s_t(\mathbf{w}_t)}, \quad \alpha_{2,t+1} = \frac{\alpha_{2,t}s_t(\mathbf{w}_t)}{\alpha_{1,t}s_t(\mathbf{v}) + \alpha_{2,t}s_t(\mathbf{w}_t)}$$

$$s_t(\mathbf{u}) = \exp\{-\eta\ell^*(\Pi(\mathbf{u}^\top \mathbf{x}_t), \Pi(y_t))\} \qquad \ell^*(z, y) = (z - y)^2$$

  - Update rules using any existing OL algorithms (e.g., PA)

Online Transfer Learning (Zhao and Hoi 2011)

- OTL on Heterogeneous domains $\mathcal{X}_1 \neq \mathcal{X}_2$
  - Assumption: not completely different
  - Each instance $\mathbf{x}_t \in \mathcal{X}$ in target domain can be split into two views: $\mathbf{x}_{1,t} \in \mathcal{X}_s$ $\mathbf{x}_{2,t} \in \mathcal{X}/\mathcal{X}_s$
  - The key idea is to use a co-regularization principle for online optimizing two classifiers $\mathbf{w}_{1,t}$ and $\mathbf{w}_{2,t}$

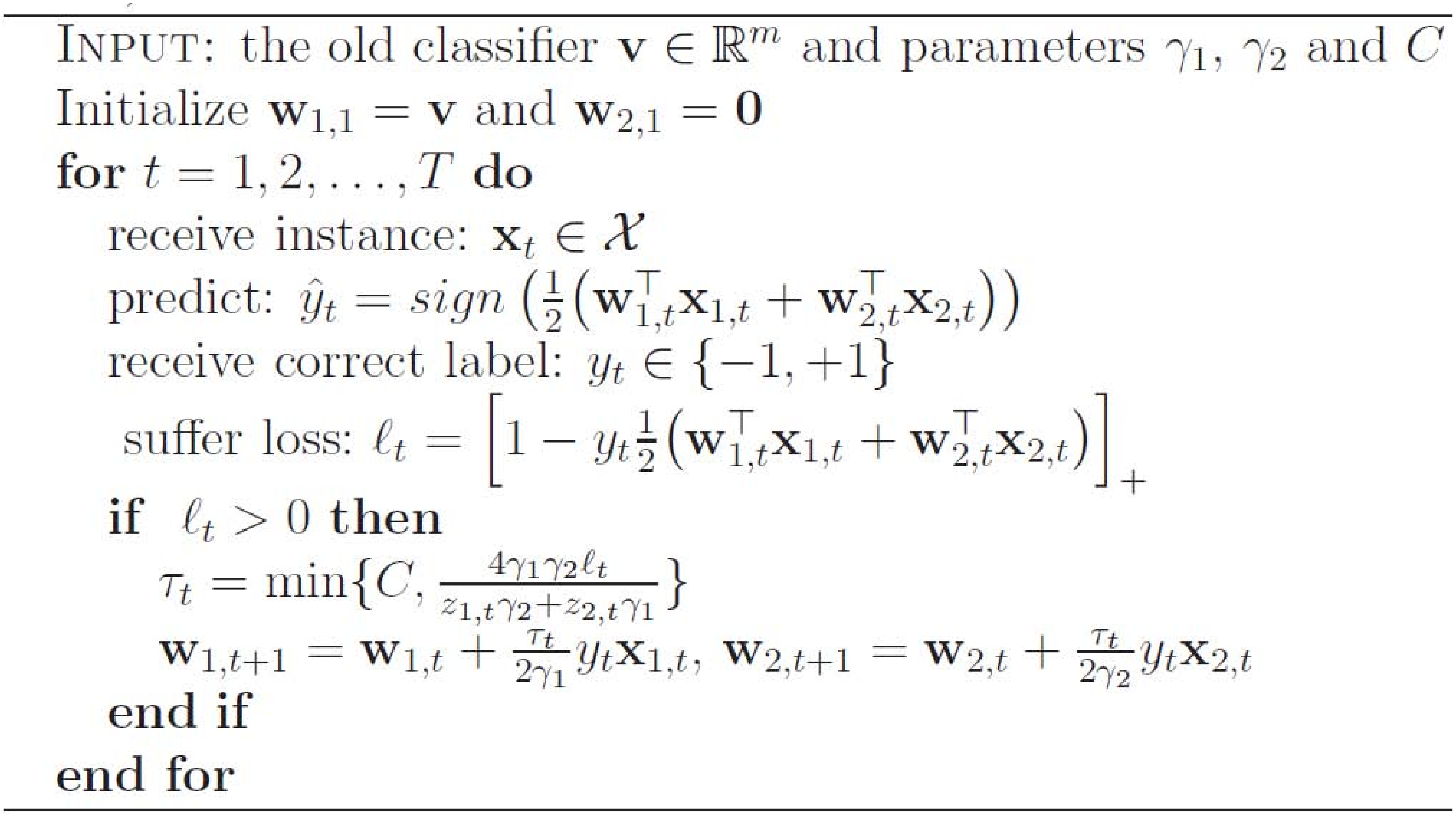
$$(\mathbf{w}_{1,t+1}, \mathbf{w}_{2,t+1}) = \arg\min_{\mathbf{w}_1 \in \mathbb{R}^m, \mathbf{w}_2 \in \mathbb{R}^{n-m}} \frac{\gamma_1}{2}\|\mathbf{w}_1 - \mathbf{w}_{1,t}\|^2 + \frac{\gamma_2}{2}\|\mathbf{w}_2 - \mathbf{w}_{2,t}\|^2 + C\ell(\mathbf{w}_1, \mathbf{w}_2; t)$$

  - Prediction can be made by

$$\hat{y}_t = sign\left(\frac{1}{2}\left(\mathbf{w}_{1,t}^\top \mathbf{x}_{1,t} + \mathbf{w}_{2,t}^\top \mathbf{x}_{2,t}\right)\right)$$

Online Transfer Learning (Zhao and Hoi 2011)

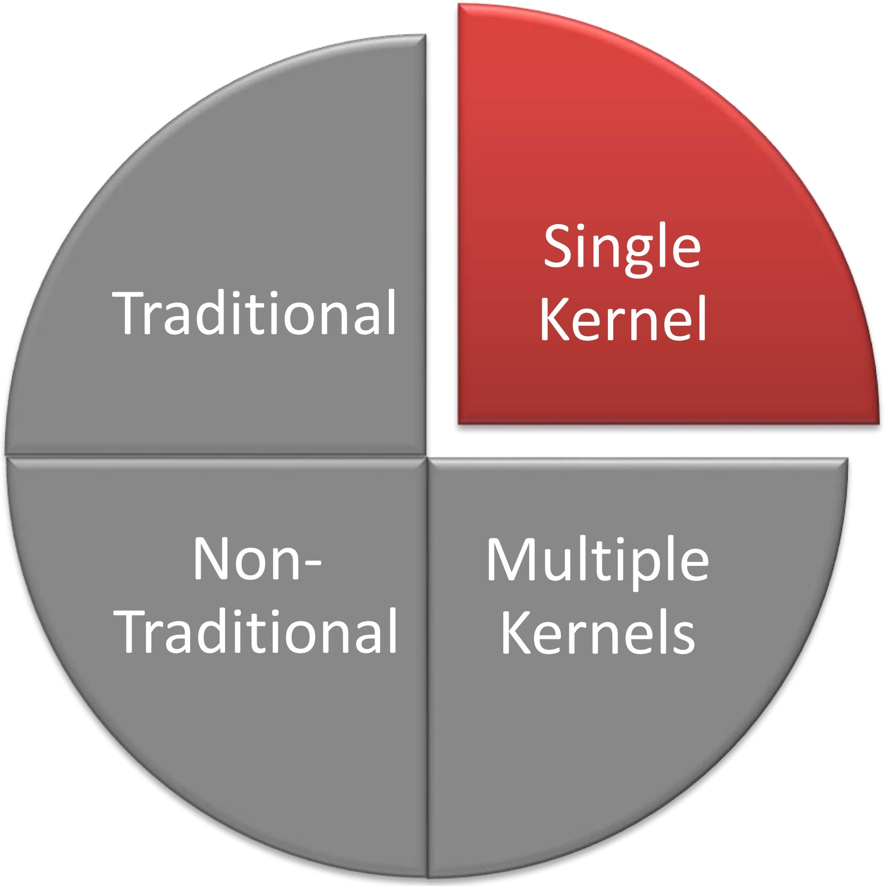
- Heterogeneous OTL algorithm



```
INPUT: the old classifier $\mathbf{v} \in \mathbb{R}^m$ and parameters $\gamma_1$, $\gamma_2$ and $C$
Initialize $\mathbf{w}_{1,1} = \mathbf{v}$ and $\mathbf{w}_{2,1} = \mathbf{0}$
for $t = 1, 2, \ldots, T$ do
  receive instance: $\mathbf{x}_t \in \mathcal{X}$
  predict: $\hat{y}_t = sign\left(\frac{1}{2}(\mathbf{w}_{1,t}^\top \mathbf{x}_{1,t} + \mathbf{w}_{2,t}^\top \mathbf{x}_{2,t})\right)$
  receive correct label: $y_t \in \{-1, +1\}$
  suffer loss: $\ell_t = \left[1 - y_t \frac{1}{2}(\mathbf{w}_{1,t}^\top \mathbf{x}_{1,t} + \mathbf{w}_{2,t}^\top \mathbf{x}_{2,t})\right]_+$
  if $\ell_t > 0$ then
    $\tau_t = \min\{C, \frac{4\gamma_1\gamma_2\ell_t}{z_{1,t}\gamma_2 + z_{2,t}\gamma_1}\}$
    $\mathbf{w}_{1,t+1} = \mathbf{w}_{1,t} + \frac{\tau_t}{2\gamma_1} y_t \mathbf{x}_{1,t}$, $\mathbf{w}_{2,t+1} = \mathbf{w}_{2,t} + \frac{\tau_t}{2\gamma_2} y_t \mathbf{x}_{2,t}$
  end if
end for
```

Online Learning: Overview

Linear Methods



Traditional

Single Kernel

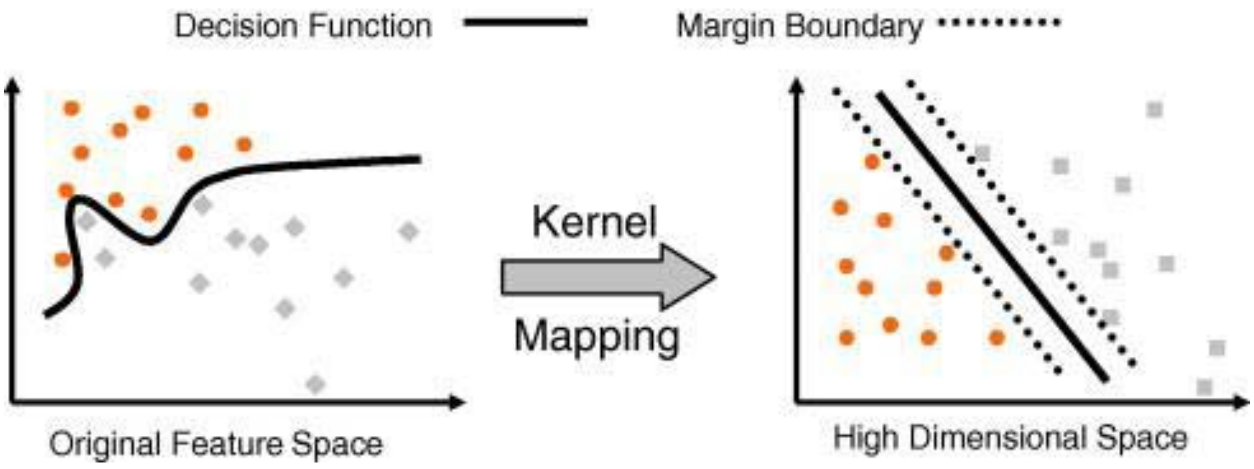
Non-Traditional

Multiple Kernels

Non-Linear Methods

Kernel-based Online Learning

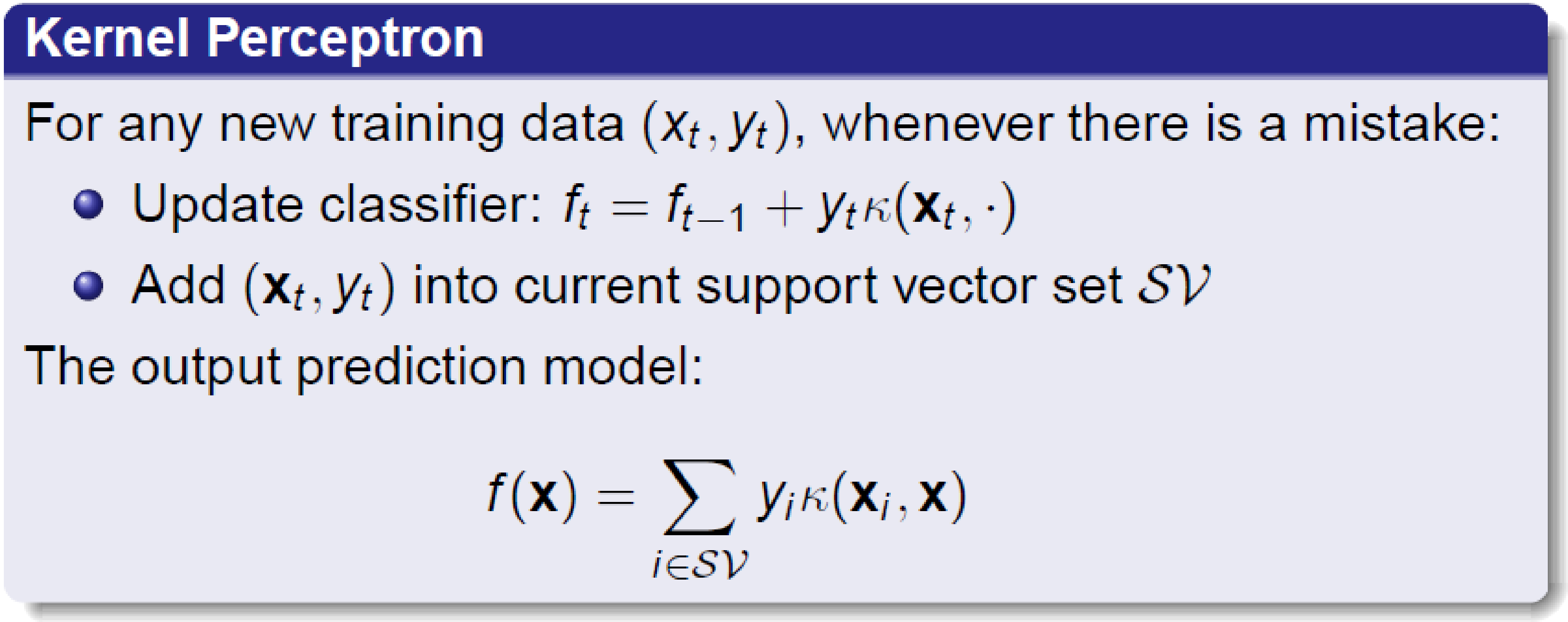
- Motivation
  - Linear classifier is limited in certain situations



- Objective
  - Learn a **non-linear** model for online classification tasks using the **kernel** trick

Kernel-based Online Learning

- Kernel Perceptron



Kernel Perceptron

For any new training data (x_t, y_t), whenever there is a mistake:

- Update classifier: $f_t = f_{t-1} + y_t \kappa(\mathbf{x}_t, \cdot)$
- Add $(\mathbf{x}_t, y_t)$ into current support vector set $\mathcal{SV}$

The output prediction model:

$$f(\mathbf{x}) = \sum_{i \in \mathcal{SV}} y_i \kappa(\mathbf{x}_i, \mathbf{x})$$

- Related Work
  - Single Updating: Kernel PA, Kernel OGD, etc.
  - Double Updating Online Learning (Zhao et al, 2011)
  - Others (e.g., Online SVM by fully optimal update)

Double Updating Online Learning (DUOL)

- Motivation
  - When a new support vector (SV) is added, the weights of existing SVs remain unchanged (i.e., only the update is applied for a single SV)
  - How to update the weights of existing SVs in an efficient and effective approach
- Main idea
  - Update the weight for one more existing SV in addition to the update of the new SV
- Challenge
  - which existing SV should be updated and how to update?

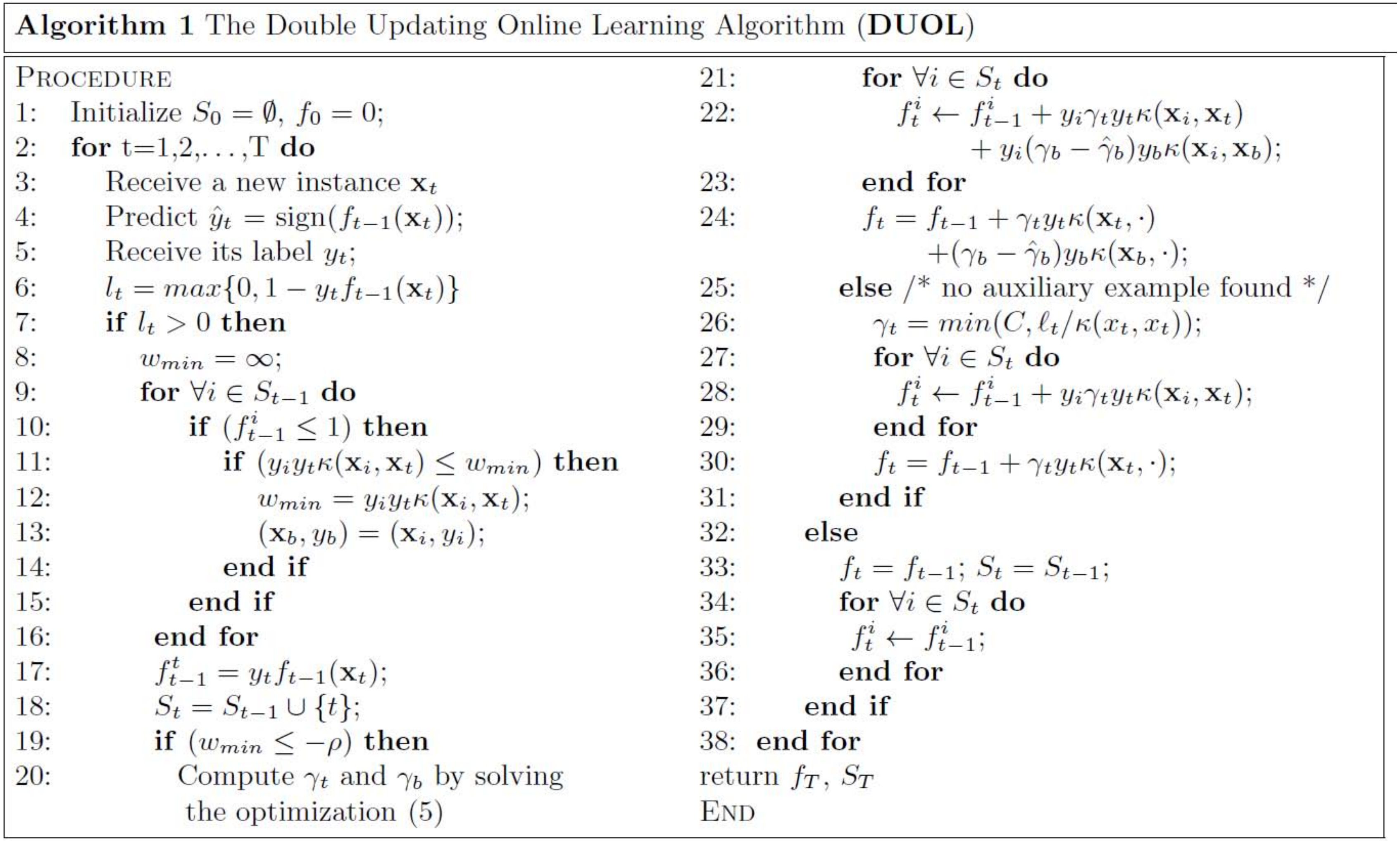
Double Updating Online Learning (DUOL)

- Denote a new Support Vector as: (x_a, y_a)
- Choose an **auxiliary** example $(x_b, y_b) \in \mathcal{D}$ from existing SVs:
  - Misclassified: $y_b f(x_b) \leq 0$
  - Conflict most with new SV: $k(x_b, x_a) y_a y_b \leq -\rho$
- Update the current hypothesis by

$$f(\cdot) = \sum_{i \in \mathcal{S} - \{b\}} \gamma_i k(x_i, \cdot) + \hat{\gamma}_b k(x_b, \cdot) + \gamma_a (x_a, \cdot)$$

- How to optimize the weights of the two SVs
- DUOL formulates the problem as a simple QP task of large margin optimization, and gives closed-form solutions.

Double Updating Online Learning (DUOL)



Algorithm 1 The Double Updating Online Learning Algorithm (**DUOL**)

```
PROCEDURE
  Initialize $S_0 = \emptyset,\ f_0 = 0$;
  for t=1,2,...,T do
    Receive a new instance $\mathbf{x}_t$
    Predict $\hat{y}_t = \mathrm{sign}(f_{t-1}(\mathbf{x}_t))$;
    Receive its label $y_t$;
    $l_t = max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$
    if $l_t > 0$ then
      $w_{min} = \infty$;
      for $\forall i \in S_{t-1}$ do
        if $(f^i_{t-1} \leq 1)$ then
          if $(y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) \leq w_{min})$ then
            $w_{min} = y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
            $(\mathbf{x}_b, y_b) = (\mathbf{x}_i, y_i)$;
          end if
        end if
      end for
      $f^t_{t-1} = y_t f_{t-1}(\mathbf{x}_t)$;
      $S_t = S_{t-1} \cup \{t\}$;
      if $(w_{min} \leq -\rho)$ then
        Compute $\gamma_t$ and $\gamma_b$ by solving
         the optimization (5)
        for $\forall i \in S_t$ do
          $f^i_t \leftarrow f^i_{t-1} + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$
                 $+ y_i(\gamma_b - \hat{\gamma}_b) y_b \kappa(\mathbf{x}_i, \mathbf{x}_b)$;
        end for
        $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$
             $+(\gamma_b - \hat{\gamma}_b) y_b \kappa(\mathbf{x}_b, \cdot)$;
      else /* no auxiliary example found */
        $\gamma_t = min(C, \ell_t / \kappa(x_t, x_t))$;
        for $\forall i \in S_t$ do
          $f^i_t \leftarrow f^i_{t-1} + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
        end for
        $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$;
      end if
    else
      $f_t = f_{t-1}$; $S_t = S_{t-1}$;
      for $\forall i \in S_t$ do
        $f^i_t \leftarrow f^i_{t-1}$;
      end for
    end if
  end for
return $f_T$, $S_T$
END
```

Kernel-based Online Learning

- Challenge
  - The number of support vectors with the kernel-based classification model is often **unbounded!**
  - Non-scalable and inefficient in practice!!
- Question
  - Can we bound the number of support vectors?
- Solution
  - “Budget Online Learning”

Budget Online Learning

- Problem
  - Kernel-based Online Learning by **bounding** the number of support vectors for a given **budget** B
- Related Work in literature
  - Randomized Budget Perceptron (Cavallanti et al.,2007)
  - Forgetron (Dekel et al.,2005)
  - Projectron (Orabona et al.,2008)
  - Bounded Online Gradient Descent (Zhao et al 2012)
  - Others

RBP: Randomized Budget Perceptron

(Cavallanti et al.,2007)

- Idea: maintaining budget by means of randomization
- Repeat whenever there is a mistake at round *t*:
  - If the number of SVs <= B, then apply Kernel Perceptron

$$f_t \leftarrow f_t + y_t k(\mathbf{x}_t, \cdot)$$

$$\mathcal{S}_t \leftarrow \mathcal{S}_{t-1} + \{t\}$$

  - Otherwise randomly discard one existing support vector

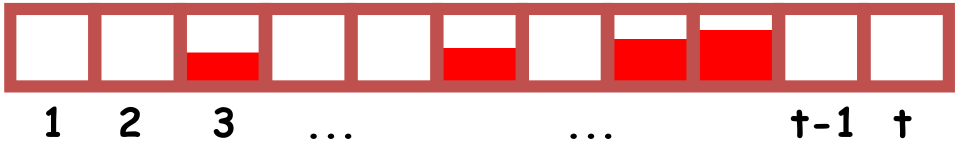
$$r \leftarrow \text{random} \ \in \ \mathcal{S}_{t-1}$$

$$f_t \leftarrow f_t + y_t k(\mathbf{x}_t, \cdot) - y_r k(\mathbf{x}_r, \cdot)$$

$$\mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \cup \{t\} - \{r\}$$

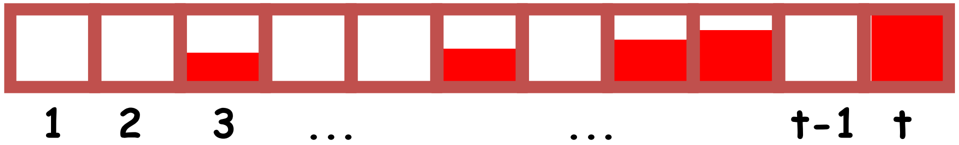
Forgetron (Dekel et al.,2005)

$$f_t = \sum_{i \in \mathcal{S}_t} \alpha_{i,t} y_i k(\mathbf{x}_i, \cdot)$$



Step (1) - Perceptron

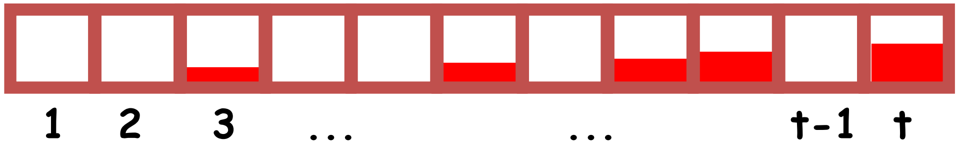
$$f'_t(\mathbf{x}) = f_t(\mathbf{x}) + y_t K(\mathbf{x}_t, \mathbf{x})$$



Step (2) – Shrinking

$$f''_t = \phi_t f'_t \qquad \phi_t \in (0, 1]$$

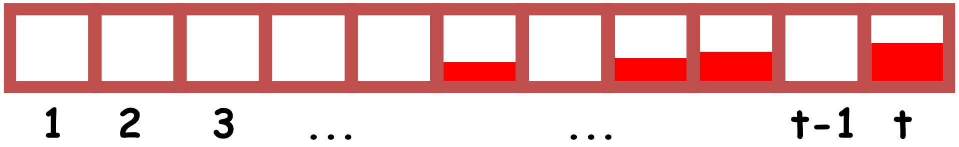
$$f''_t = \sum_{i \in \mathcal{S}'_t} \alpha_{i,t+1} y_i k(\mathbf{x}_i, \cdot)$$



Step (3) – Remove Oldest

$$r = \min \mathcal{S}_t$$

$$\mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \cup \{t\} - \{r\}$$



Projectron (Orabona et al., 2008)

- The new hypothesis

$$f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$$

is projected onto the space spanned by $\mathcal{S}_{t-1}$

$$f''_t = P_{t-1} f'_t = P_{t-1}\left(f_{t-1} + y_t k(\mathbf{x}_t, \cdot)\right)$$

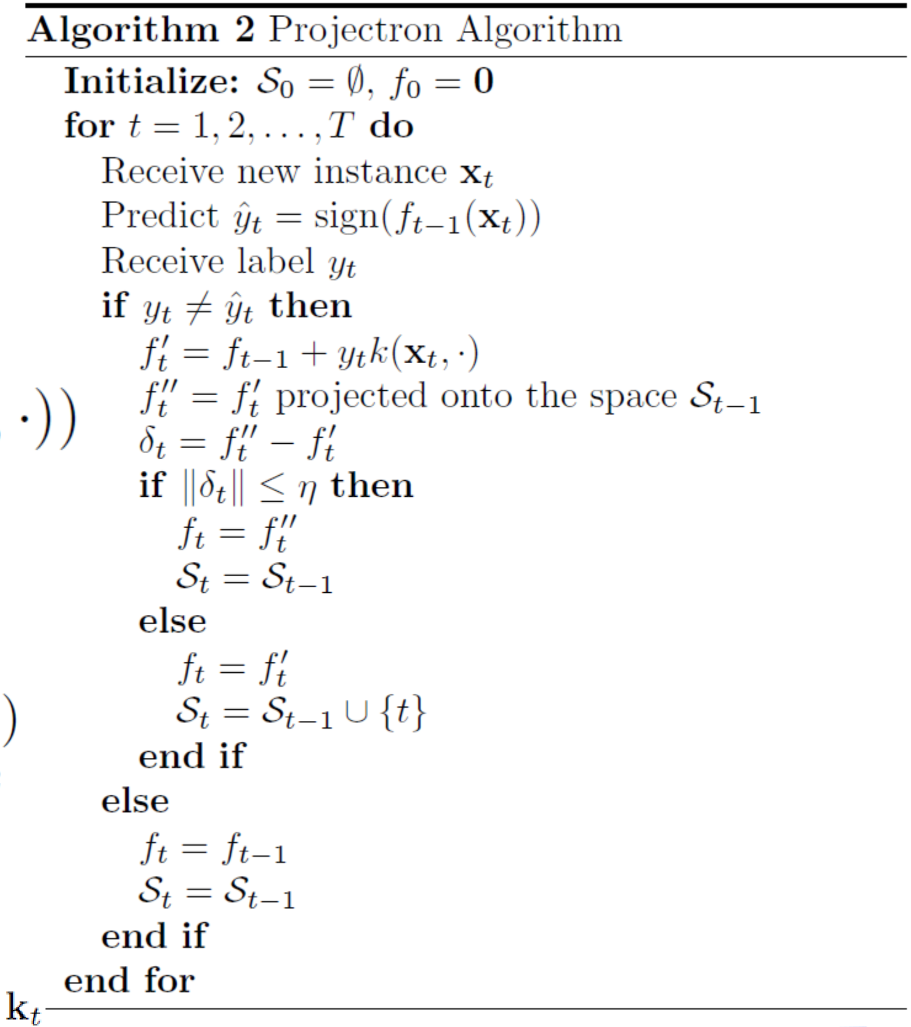
- How to solve the projection?

$$f''_t = f_{t-1} + y_t P_{t-1} k(\mathbf{x}_t, \cdot)$$

$$\delta_t = f''_t - f'_t = y_t P_{t-1} k(\mathbf{x}_t, \cdot) - y_t k(\mathbf{x}_t, \cdot)$$

$$\|\delta_t\|^2 = \min_{(d_1,\dots,d_{t-1})} \left\| \sum_{j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot) \right\|^2$$

$$f''_t = f_{t-1} + y_t \sum_{j \in \mathcal{S}_{t-1}} \mathbf{d}^\star_j k(\mathbf{x}_j, \cdot) \qquad \mathbf{d}^\star = \mathbf{K}^{-1}_{t-1} \mathbf{k}_t$$



Algorithm 2 Projectron Algorithm

```
Initialize: $\mathcal{S}_0 = \emptyset$, $f_0 = \mathbf{0}$
for $t = 1, 2, \dots, T$ do
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$
  Receive label $y_t$
  if $y_t \neq \hat{y}_t$ then
    $f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$
    $f''_t = f'_t$ projected onto the space $\mathcal{S}_{t-1}$
    $\delta_t = f''_t - f'_t$
    if $\|\delta_t\| \leq \eta$ then
      $f_t = f''_t$
      $\mathcal{S}_t = \mathcal{S}_{t-1}$
    else
      $f_t = f'_t$
      $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \{t\}$
    end if
  else
    $f_t = f_{t-1}$
    $\mathcal{S}_t = \mathcal{S}_{t-1}$
  end if
end for
```

Bounded Online Gradient Descent

(Zhao et al 2012)

- Limitations of previous work
  - Perceptron-based, heuristic or expensive update
- Motivation of BOGD
  - Learn the kernel-based model using online gradient descent by constraining the SV size less than a predefined budget B
- Challenges
  - How to efficiently maintain the budget?
  - How to minimize the impact due to the budget maintenance?

Bounded Online Gradient Descent

(Zhao et al 2012)

- Main idea of the BOGD algorithms
  - A **stochastic budget maintenance** strategy to guarantee
    - One existing SV will be discarded by **multinomial** sampling
    - **Unbiased** estimation with only **B** SVs;
- Formulation
  - Current hypothesis $f_t(\cdot) = \sum_{i=1}^{B} \alpha_i^t y_i^t \kappa(\mathbf{x}_i^t, \cdot)$
  - Construct an unbiased estimator (to ensure $\mathrm{E}[\widehat{f}_t(\cdot)] = f_t(\cdot)$)

$$\widehat{f}_t(\cdot) = \sum_{i=1}^{B} \left(a_i^t Z_i^t + b_i^t\right) y_i^t \kappa(\mathbf{x}_i^t, \cdot)$$

$\sum_{i=1}^{B} Z_i^t = 1$. $Z_i^t = 1$ indicates the i-th SV is selected for removal

Bounded Online Gradient Descent

(Zhao et al 2012)

- Z is obtained according a sampling distribution:

$$\mathbf{p}^t = (p_1^t, \ldots, p_B^t)$$

- The final weights of SV:

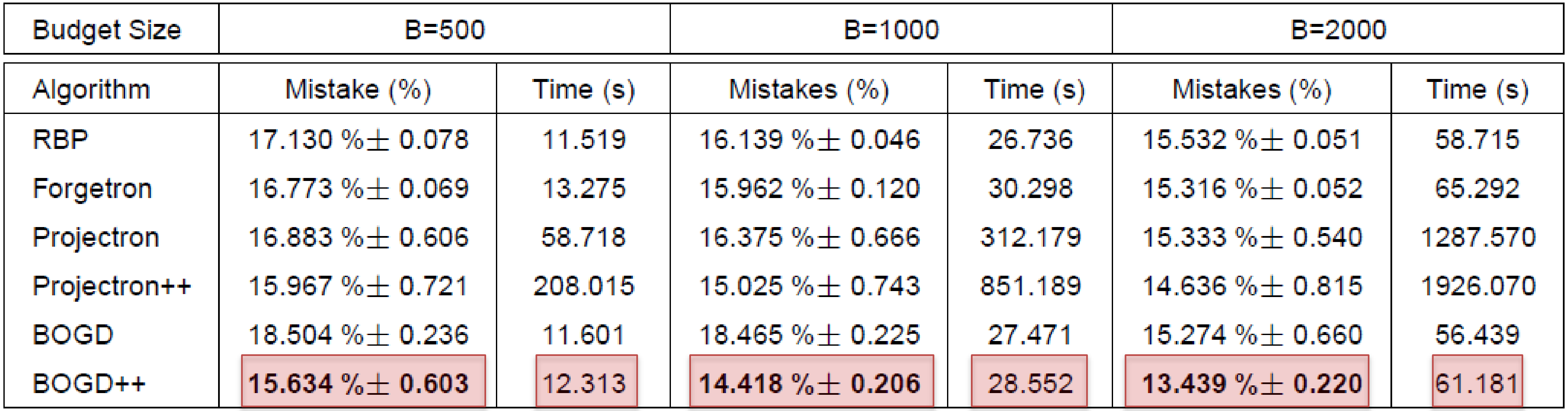
$$\alpha_i^{t+1} = \min\left((1 - Z_i^t)\frac{1 - \lambda\eta}{1 - p_i^t}\alpha_i^t, \gamma\eta\right),\ i \in [B]$$

- How to choose a proper sampling **p**?
  - – Uniform sampling $p_i^t = 1/B \text{ for any } i \in [B]$
  - – Non-uniform sampling $p_i = 1 - s\alpha_i\sqrt{\kappa(\mathbf{x}_i, \mathbf{x}_i)}$

$$s = \frac{(B-1)}{\sum_{i=1}^{B} \alpha_i \sqrt{\kappa(\mathbf{x}_i, \mathbf{x}_i)}}$$

Empirical Results of BOGD

- Comparison
  - Baseline: Forgetron, RBP, Projectron, Projectron++
  - Our algorithms: BOGD (uniform), BOGD++ (non-uniform)
- Evaluation of budget online learning algorithms



Budget Size	B=500		B=1000		B=2000	
Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
RBP	17.130 %± 0.078	11.519	16.139 %± 0.046	26.736	15.532 %± 0.051	58.715
Forgetron	16.773 %± 0.069	13.275	15.962 %± 0.120	30.298	15.316 %± 0.052	65.292
Projectron	16.883 %± 0.606	58.718	16.375 %± 0.666	312.179	15.333 %± 0.540	1287.570
Projectron++	15.967 %± 0.721	208.015	15.025 %± 0.743	851.189	14.636 %± 0.815	1926.070
BOGD	18.504 %± 0.236	11.601	18.465 %± 0.225	27.471	15.274 %± 0.660	56.439
BOGD++	**15.634 %± 0.603**	12.313	**14.418 %± 0.206**	28.552	**13.439 %± 0.220**	61.181

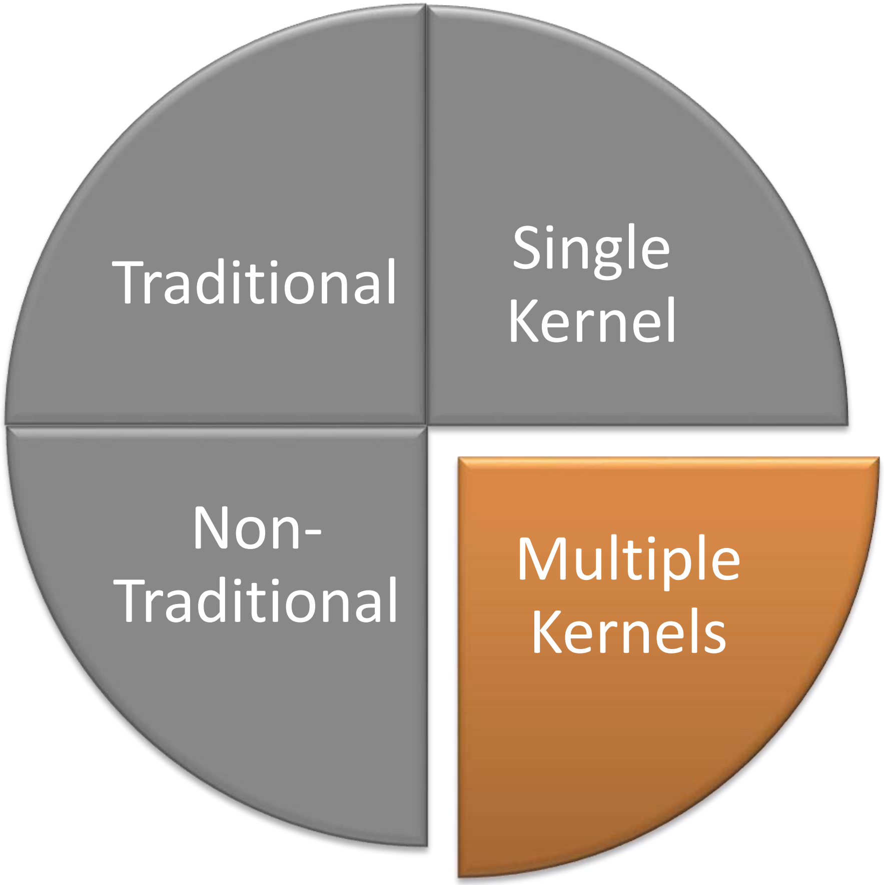
Experimental result of varied budget sizes on the *codrna* data set (n=271617)

Summary

- Pros of BOGD
  - ☺ Very efficient due to stochastic strategy
  - ☺ Rather scalable
  - ☺ State-of-the-art performance
  - ☺ Theoretical guarantee
- Cons of existing BOL
  - ☹ Predefined budget size (optimal budget size)?
  - ☹ Only learn with a single kernel

Online Learning: Overview

Linear Methods



Traditional

Single Kernel

Non-Traditional

Multiple Kernels

Non-Linear Methods

Online Multiple Kernel Learning

- Motivation
  - **Kernel** defines a **similarity measure** for two objects
  - Many ways to define a kernel function
  - Example: Multimedia Applications
    - Image: color, texture, shape, local features, etc
    - Video: visual, textural, audio features, etc.
- Problem
  - Can we learn a kernel-based model incrementally from a sequence of (multi-modal) instances using multiple kernels in an online learning setting?

Kernel and Multimodal Representation

- **Kernel trick**
  - Mapping observations from a general set S into an inner product space V
- **Kernel function** K(**x**, **x**’) for any **x**, **x**’ from S
  - Expressed as an inner product between two objects in V

$$\langle\varphi(x),\varphi(x')\rangle =: k(x,x')$$

  - Defines a similarity measure between any two objects
  - Example: linear, polynomial, Gaussian, etc.
  - Kernels on structured data: tree, graph, etc,
- Multi-modal representations for image applications
  - Color features (color histogram, color moment, etc)
  - Texture features (Gabor, GIST, etc)
  - Edge features (edge direction histogram, etc)
  - Local features (bag of SIFT features, bag of words, etc)

What is MKL?

- Multiple Kernel Learning (MKL)
  - Kernel method by an optimal combination of multiple kernels
- Batch MKL Formulation

$$\min_{\theta\in\Delta}\ \min_{f\in\mathcal{H}_{K(\theta)}}\ \frac{1}{2}\|f\|^2_{\mathcal{H}_{K(\theta)}} + C\sum_{i=1}^{n} l(f(x_i), y_i)$$

$$\Delta = \{\theta \in \mathbb{R}^m_+ | \theta^T e_m = 1\} \quad K(\theta)(\cdot,\cdot) \quad = \quad \sum_{i=1}^m \theta_i \kappa_i(\cdot,\cdot)$$

$$\min_{\theta\in\Delta}\max_{\alpha\in\Xi}\left\{\alpha^\top \mathbf{1}_n - \frac{1}{2}(\alpha\circ\mathbf{y})^\top\left(\sum_{i=1}^{m}\theta_i K^i\right)(\alpha\circ\mathbf{y})\right\}$$

- Hard to solve the convex-concave optimization for big data!

Can we avoid having to directly solve the optimization?

Online MKL (Hoi et al., ML'12)

- Objective
  - Learn a kernel-based predictor with multiple kernels from a sequence of (multi-modal) data examples
  - Avoid the need of solving complicated optimizations
- Main idea: a simple two-step online learning

  At each learning iteration, if there is a mistake:

  - Step 1: Online learning with each single kernel
    - Kernel Perceptron (Rosenblatt Frank, 1958, Freund 1999)

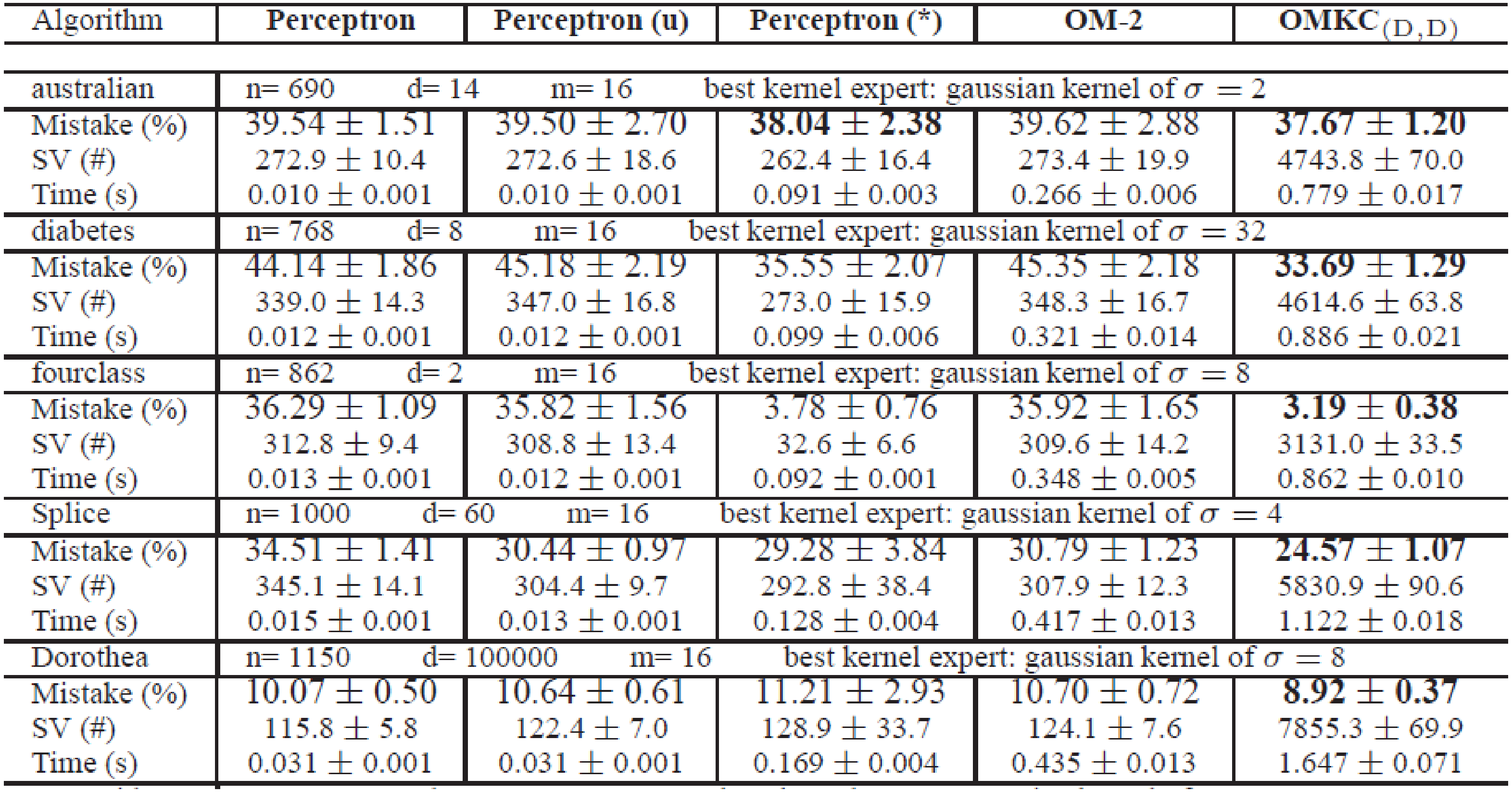
$$f_{t+1,i}(x) = f_{t,i}(x) + z_i(t) y_t \kappa_i(x_t, x)$$

  - Step 2: Learning to update the combination weights
    - Hedging algorithm

$$\theta_i(t+1) = \theta_i(t)\beta^{z_i(t)}$$

Online MKL for Classification

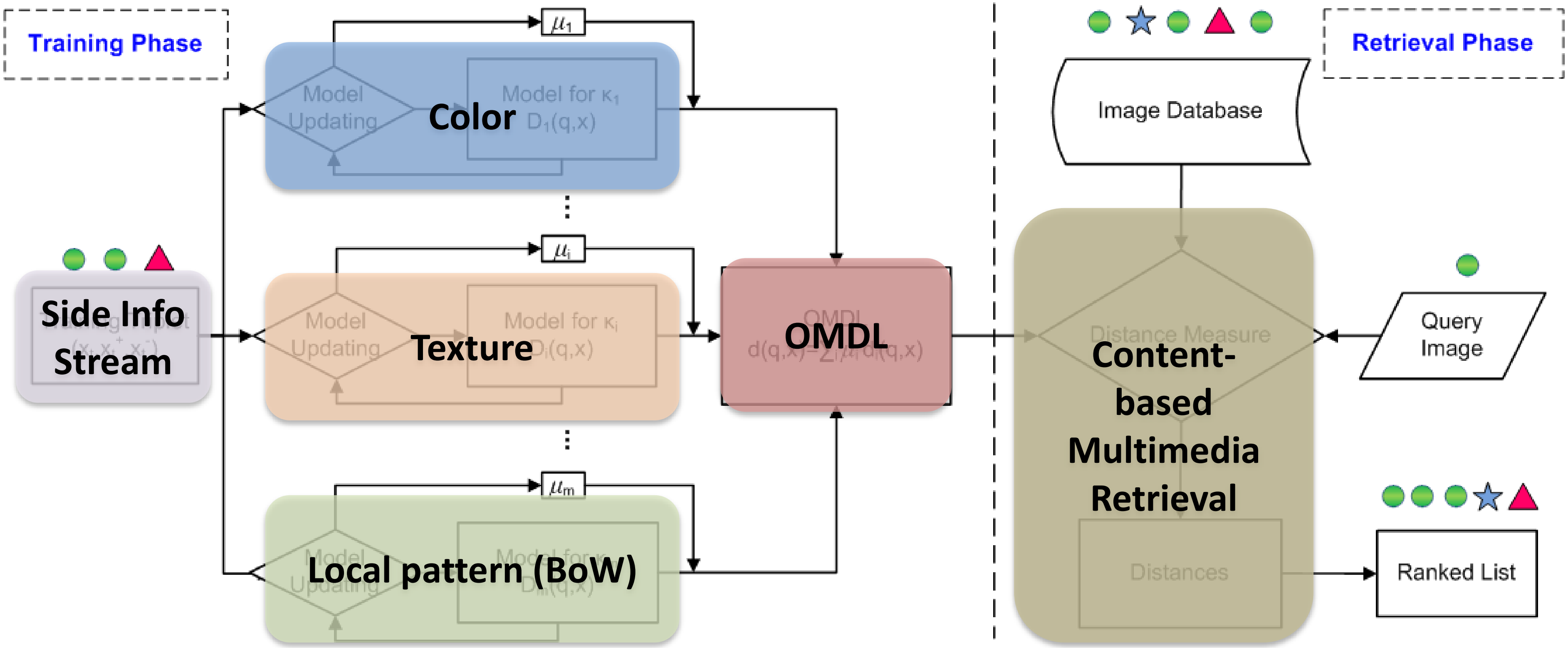
- Empirical Results of OMKC for classification



Algorithm	Perceptron	Perceptron (u)	Perceptron (*)	OM-2	OMKC$_{(D,D)}$
australian	n= 690 d= 14 m= 16 best kernel expert: gaussian kernel of $\sigma = 2$				
Mistake (%)	39.54 ± 1.51	39.50 ± 2.70	$\mathbf{38.04 \pm 2.38}$	39.62 ± 2.88	$\mathbf{37.67 \pm 1.20}$
SV (#)	272.9 ± 10.4	272.6 ± 18.6	262.4 ± 16.4	273.4 ± 19.9	4743.8 ± 70.0
Time (s)	0.010 ± 0.001	0.010 ± 0.001	0.091 ± 0.003	0.266 ± 0.006	0.779 ± 0.017
diabetes	n= 768 d= 8 m= 16 best kernel expert: gaussian kernel of $\sigma = 32$				
Mistake (%)	44.14 ± 1.86	45.18 ± 2.19	35.55 ± 2.07	45.35 ± 2.18	$\mathbf{33.69 \pm 1.29}$
SV (#)	339.0 ± 14.3	347.0 ± 16.8	273.0 ± 15.9	348.3 ± 16.7	4614.6 ± 63.8
Time (s)	0.012 ± 0.001	0.012 ± 0.001	0.099 ± 0.006	0.321 ± 0.014	0.886 ± 0.021
fourclass	n= 862 d= 2 m= 16 best kernel expert: gaussian kernel of $\sigma = 8$				
Mistake (%)	36.29 ± 1.09	35.82 ± 1.56	3.78 ± 0.76	35.92 ± 1.65	$\mathbf{3.19 \pm 0.38}$
SV (#)	312.8 ± 9.4	308.8 ± 13.4	32.6 ± 6.6	309.6 ± 14.2	3131.0 ± 33.5
Time (s)	0.013 ± 0.001	0.012 ± 0.001	0.092 ± 0.001	0.348 ± 0.005	0.862 ± 0.010
Splice	n= 1000 d= 60 m= 16 best kernel expert: gaussian kernel of $\sigma = 4$				
Mistake (%)	34.51 ± 1.41	30.44 ± 0.97	29.28 ± 3.84	30.79 ± 1.23	$\mathbf{24.57 \pm 1.07}$
SV (#)	345.1 ± 14.1	304.4 ± 9.7	292.8 ± 38.4	307.9 ± 12.3	5830.9 ± 90.6
Time (s)	0.015 ± 0.001	0.013 ± 0.001	0.128 ± 0.004	0.417 ± 0.013	1.122 ± 0.018
Dorothea	n= 1150 d= 100000 m= 16 best kernel expert: gaussian kernel of $\sigma = 8$				
Mistake (%)	10.07 ± 0.50	10.64 ± 0.61	11.21 ± 2.93	10.70 ± 0.72	$\mathbf{8.92 \pm 0.37}$
SV (#)	115.8 ± 5.8	122.4 ± 7.0	128.9 ± 33.7	124.1 ± 7.6	7855.3 ± 69.9
Time (s)	0.031 ± 0.001	0.031 ± 0.001	0.169 ± 0.004	0.435 ± 0.013	1.647 ± 0.071

Online MKL for Multimedia Retrieval

- Online Multi-Modal Distance Learning (Xia et al 2013)
  - Goal: Learning multi-kernel distance for multimedia retrieval


Training Phase

Side Info Stream

Color

Texture

Local pattern (BoW)

OMDL

Content-based Multimedia Retrieval

Retrieval Phase

Problem Settings

Consider a multi-modal object retrieval task

- A set of **n** multimedia objects $\mathcal{X} = \{\mathbf{x}_i | i = 1, 2, \ldots, n\}$
- A collection of triplet constraints

$$\mathcal{C} = \{(i_t, j_t, k_t), t = 1, \ldots, T\}$$

  where each triplet (i_t, j_t, k_t) indicates that object $\mathbf{x}_{i_t}$ **similar** to object $\mathbf{x}_{j_t}$, but **dissimilar** to object $\mathbf{x}_{k_t}$

- For simplicity, we will simply denote $\mathbf{x}_{i_t}$ as $\mathbf{x}_i$ for the rest of discussion.

- A set of **m** predefined kernel functions

$$\mathbf{K}^p : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \qquad p = 1, 2, \ldots, m$$

Formulation

- Kernel-based distance measure

$$d(\mathbf{x}_i, \mathbf{x}_j) \doteq \sum_{p=1}^{m} \mu_p \left(\mathbf{K}_i^p - \mathbf{K}_j^p\right)^\top \mathbf{W}^p \left(\mathbf{K}_i^p - \mathbf{K}_j^p\right)$$

$$\mathbf{W} \in \mathbb{R}^{n \times n} \qquad \mathbf{K}_i = [k(\mathbf{x}_i, \mathbf{x}_1), \ldots, k(\mathbf{x}_i, \mathbf{x}_n)]^\top$$

- For each triple constraint (i_t, j_t, k_t)

$$d(\mathbf{x}_i, \mathbf{x}_j) + 1 \leq d(\mathbf{x}_i, \mathbf{x}_k) + \xi_{ijk}, \ \xi_{ijk} \geq 0$$

- Graph Laplacian Regularizer

$$\begin{aligned} \Omega(\mathbf{V}, \mathbf{S}) &= \frac{1}{2} \sum_{i,j=1}^{n} S_{ij} \left\| \frac{\mathbf{v}_i}{\sqrt{D_i}} - \frac{\mathbf{v}_j}{\sqrt{D_j}} \right\|_2^2 \\ &= tr(\mathbf{V}\mathbf{L}\mathbf{V}^\top) = tr(\mathbf{L}\mathbf{K}) \end{aligned}$$

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{1/2}$$

Formulation (cont’)

Graph Laplacian Regularization on Target Kernel

Loss of violating the Triplet Constraints

$$\min_{\mathbf{W}^p,\xi} \quad \sum_{p=1}^{m} \boxed{tr(\mathbf{K}^p\mathbf{W}^p\mathbf{K}^p\mathbf{L}^p)} + \frac{\beta}{|\mathcal{C}|} \sum_{\forall(i,j,k)\in\mathcal{C}} \boxed{\xi_{ijk}}$$

$$\begin{aligned}
s.t. \quad & d(\mathbf{x}_i,\mathbf{x}_j) \doteq \sum_{p=1}^{m} (\mathbf{K}_i^p - \mathbf{K}_j^p)^\top \mathbf{W}^p (\mathbf{K}_i^p - \mathbf{K}_j^p) \\
& d(\mathbf{x}_i,\mathbf{x}_j) + 1 \leq d(\mathbf{x}_i,\mathbf{x}_k) + \boxed{\xi_{ijk}}, \; \xi_{ijk} \geq 0, \forall(i,j,k) \in \mathcal{C} \\
& \mathbf{W}^p \succeq \mathbf{0} \quad p = 1, 2, \ldots, m
\end{aligned}$$

- Difficult to solve the above optimization directly.

Online Learning with A Single Kernel

- For a received triplet (i_t, j_t, k_t) at round t

$$\begin{array}{ll} \min\limits_{\mathbf{W},\xi} & \frac{1}{2}\|\mathbf{W}-\mathbf{W}_{t-1}\|_F^2 + C_1 tr(\mathbf{KWKL}) + C_2\xi \\ s.t. & d(\mathbf{x}_i,\mathbf{x}_j) \doteq (\mathbf{K}_i - \mathbf{K}_j)^\top \mathbf{W}(\mathbf{K}_i - \mathbf{K}_j) \\ & d(\mathbf{x}_i,\mathbf{x}_j) + 1 \leq d(\mathbf{x}_i,\mathbf{x}_k) + \xi,\ \xi \geq 0 \\ & \mathbf{W} \succeq \mathbf{0} \end{array}$$

- Closed-form solutions can be derived with PSD.

Online Learning with Multiple Kernels

- Multi-modal distance functions

$$
\begin{aligned}
d_p(\mathbf{x}_i, \mathbf{x}_j) &= (\mathbf{K}_i^p - \mathbf{K}_j^p)^\top \mathbf{W}^p (\mathbf{K}_i^p - \mathbf{K}_j^p) \\
d(\mathbf{x}_i, \mathbf{x}_j) &= \sum_{p=1}^{m} \mu_p d_p(\mathbf{x}_i, \mathbf{x}_j)
\end{aligned}
$$

- How to optimize the combination weights μ_p ?
- Online Learning for the Optimal Combination

$$\mu_p(t) = \mu_p(t-1)\eta^{z_p(t)} \qquad p = 1, 2, \ldots, m$$

$z_p(t)$ outputs 1 when $d_p(\mathbf{x}_i, \mathbf{x}_j) > d_p(\mathbf{x}_i, \mathbf{x}_k)$ and 0 otherwise

$\eta \in (0, 1)$ is a decay factor

- The above approach follows the idea of Hedging algorithm for learning with expert advice

OMDL-LR: Low-rank Approximation

- Instead of learning a high dimensional matrix **W** which is d × d, we learn a matrix $\mathbf{W}_{LR}$ of dimension $d_{LR} \times d_{LR}$ $(d_{LR} \ll d)$
- We generate a $d \times d_{LR}$ projection matrix **P** based on a Gaussian distribution, and use $\mathbf{P}\mathbf{W}_{LR}\mathbf{P}^\top$ to approximate **W**.

$$\mathbf{W} \approx \mathbf{P}\mathbf{W}_{LR}\mathbf{P}^\top$$

- Once the random projection matrix **P** is chosen, it is straightforward for improving the rest of the algorithm by projecting the columns of kernel values accordingly,

$$\mathbf{K}_i \leftarrow \mathbf{P}^\top \mathbf{K}_i$$

- One could also try other low-rank approximation methods, e.g., Nystrom (Williams and Seeger, 2001).

Results (mAP) for Multi-modal Image Retrieval

	Algorithm	Metric	Caltech10	Caltech20	Caltech50	Corel5000	ImageCLEF	ImageCLEF+
Selecting Best Modality	Kernel-Best	mAP	0.2315	0.1657	0.1010	0.1789	0.4090	0.0959
		std	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000
	RCA-Best	mAP	0.2343	0.1714	0.1012	0.1801	0.4709	0.1098
		std	±0.0003	±0.0001	±0.0001	±0.0001	±0.0004	±0.0002
	KRCA-Best	mAP	0.2489	0.1867	0.1087	0.2113	0.5497	0.1078
		std	±0.0004	±0.0003	±0.0001	±0.0002	±0.0012	±0.0008
	LMNN-Best	mAP	0.2365	0.1696	0.1049	0.1909	0.4939	0.1069
		std	±0.0013	±0.0014	±0.0002	±0.0002	±0.0013	±0.0019
	OASIS-Best	mAP	0.2472	0.1852	0.1010	0.1797	0.4325	0.1062
		std	±0.0036	±0.0052	±0.0000	±0.0054	±0.0087	±0.0115
Concatenating all the features	Kernel-Con	mAP	0.2115	0.1609	0.0998	0.2518	0.3959	0.1598
		std	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000
	RCA-Con	mAP	0.2173	0.1699	0.1040	0.2591	0.5044	0.2176
		std	±0.0001	±0.0002	±0.0000	±0.0002	±0.0006	±0.0009
	KRCA-Con	mAP	0.2404	0.1955	0.1114	0.3240	0.5797	0.1936
		std	±0.0008	±0.0001	±0.0001	±0.0002	±0.0012	±0.0005
	LMNN-Con	mAP	0.2419	0.1781	0.1097	0.2768	0.5373	0.2272
		std	±0.0009	±0.0013	±0.0002	±0.0005	±0.0021	±0.0030
	OASIS-Con	mAP	0.2350	0.1633	0.1001	0.2518	0.4518	0.1560
		std	±0.0044	±0.0043	±0.0004	±0.0077	±0.0119	±0.0047
Simple combined kernel	Kernel-U	mAP	0.2536	0.2133	0.1247	0.3310	0.4415	0.2554
		std	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000
	Kernel-W	mAP	0.2557	0.2148	0.1253	0.3321	0.4602	0.2682
		std	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000	±0.0000
	OMDL-LR	mAP	**0.3377**	**0.2498**	**0.1356**	**0.3639**	**0.6136**	**0.3147**
		std	±**0.0047**	±**0.0025**	±**0.0004**	±**0.0068**	±**0.0098**	±**0.0220**

Summary of OMKL

- Pros
  - ☺ Nonlinear models for tough applications
  - ☺ Avoid solving complicated optimization directly
  - ☺ Handle multi-modal data
  - ☺ Theoretical guarantee
- Cons
  - ☹ Scalability has to be further improved

Discussions and Open Issues

- Challenges of Big Data
  - Volume
    - Explosive growing data: from million to billion scales
    - From a single machine to multiple machines in parallel
  - Velocity
    - Data arrives extremely fast
    - From a normal scheme to a real-time solution
  - Variety
    - Heterogeneous data and diverse sources
    - From centralized approach to distributed solutions

Discussions and Open Issues

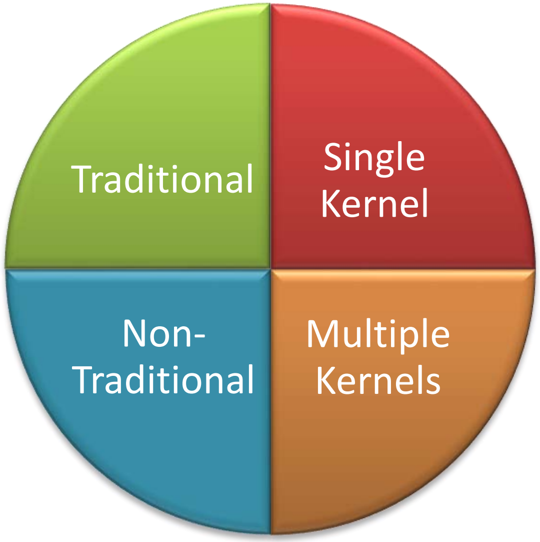
- Other Issues
  - High-dimensionality
  - Data sparsity
  - Structural data
  - Noise and incomplete data
  - Concept drifting
  - Domain adaption
  - Incorporation of background knowledge
  - Parallel & distributed computing
- User interaction
  - Interactive OL vs Passive OL
  - Human computation

Discussions and Open Issues

- Applications of Big Data Mining
  - Web Search and Mining
  - Social Network and Social Media
  - Speech Recognition & Mining (e.g., SIRI)
  - Multimedia Retrieval
  - Computer Vision
  - Medical and Healthcare Informatics
  - Financial Engineering
  - etc

Conclusion

- Introduction of emerging opportunities and challenges for big data mining
- Introduction of online learning, widely applied for various real-word applications with big data mining
- Survey of classical and state-of-the-art online learning techniques



Take-Home Message

- Online learning is promising for big data mining
- More challenges and opportunities ahead:
  - More smart online learning algorithms
  - Handle more real-world challenges, e.g., concept drifting, noise, sparse data, high-dimensional issues, etc.
  - Scale up for mining billions of instances using distributed computing facilities & parallel programming (e.g., Hadoop)

LIBOL: An open-source Library of Online Learning Algorithms
http://libol.stevenhoi.org

THANKS!

References

- Steven C.H. Hoi, Rong Jin, Tianbao Yang, Peilin Zhao, "Online Multiple Kernel Classification", Machine Learning (ML), 2013.
- Hao Xia, Pengcheng Wu, Steven C.H. Hoi, "Online Multi-modal Distance Learning for Scalable Multimedia Retrieval“, ACM Intl. Conf. on Web Search and Data Mining (WSDM),2013.
- Peilin Zhao, Jialei Wang, Pengcheng Wu, Rong Jin, Steven C.H. Hoi, "Fast Bounded Online Gradient Descent Algorithms for Scalable Kernel-Based Online Learning", The 29th International Conference on Machine Learning (ICML), June 26 - July 1, 2012.
- Jialei Wang, Steven C.H. Hoi, "Exact Soft Confidence-Weighted Learning ", The 29th International Conference on Machine Learning (ICML), June 26 - July 1, 2012.
- Bin Li, Steven C.H. Hoi, "On-line Portfolio Selection with Moving Average Reversion", The 29th International Conference on Machine Learning (ICML), June 26-July 1, 2012.
- Jialei Wang, Peilin Zhao, Steven C.H. Hoi, "Cost-Sensitive Online Classification“, IEEE International Conference on Data Mining (ICDM), 2012.
- Bin Li, Peilin Zhao, Steven C.H. Hoi, V. Gopalkrishnan, "PAMR: Passive-Aggressive Mean Reversion Strategy for Portfolio Selection", Machine Learning, vol.87, no.2, pp.221-258, 2012.
- Steven C.H. Hoi, Jialei Wang, Peilin Zhao, Rong Jin, Online Feature Selection for Big Data Mining, ACM SIGKDD Workshop on Big Data Mining (BigMine), Beijing, China, 2012
- Peilin Zhao, Steven C.H. Hoi, Rong Jin, "Double Updating Online Learning", Journal of Machine Learning Research (JMLR), 2011.
- Peilin Zhao, Steven C.H. Hoi, Rong Jin, Tianbo Yang, "Online AUC Maximization" The 28th International Conference on Machine Learning (ICML), 2011.

References

- Duchi, John C., Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. JMLR, 12:2121–2159, 2011.
- Jin, R., Hoi, S. C. H. and Yang, T. Online multiple kernel learning: Algorithms and mistake bounds, *ALT, pp. 390–404., 2010.*
- Peilin Zhao and Steven C.H. Hoi, "OTL: A Framework of Online Transfer Learning" The 27th International Conference on Machine Learning, Haifa, Israel, 21-24 June, 2010
- Crammer, Koby and D.Lee, Daniel. Learning via gaussian herding. In NIPS, pp. 345–352, 2010.
- Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems (NIPS), 2009.*
- M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *ICML, pages 264–271, 2008.*
- Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput., 37(5):1342–1372, 2008. ISSN 0097-5397.*
- Francesco Orabona, Joseph Keshet, and Barbara Caputo. The projectron: a bounded kernel-based perceptron. In *ICML, pages 720–727, 2008.*
- Crammer, Koby, Dredze, Mark, and Pereira, Fernando. Exact convex confidence-weighted learning. In NIPS, pp. 345–352, 2008.
- Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning, 69(2-3):143–167, 2007.*
- N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games. Cambridge University* Press, 2006.

References

- Crammer, Koby, Dekel, Ofer, Keshet, Joseph, Shalev-Shwartz, Shai, and Singer, Yoram. Online passive aggressive algorithms. JMLR, 7:551–585, 2006.
- Cesa-Bianchi, Nicol`o, Conconi, Alex, and Gentile, Claudio. A second-order perceptron algorithm. SIAM J. Comput., 34(3):640–668, 2005.
- Koby Crammer, Jaz S. Kandola, and Yoram Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems (NIPS), 2003.*
- Claudio Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research, 2:213–242, 2001.*
- Jyrki Kivinen, Alex J. Smola, and Robert C. Williamson. Online learning with kernels. In *Advances in Neural Information Processing Systems (NIPS), pages 785–792, 2001.*
- Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Mach. Learn., 37(3):277–296, 1999.*
- Yi Li and Philip M. Long. The relaxed online maximum margin algorithm. In *Advances in Neural Information Processing Systems (NIPS), pages 498–504, 1999.*
- *Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119--139, (1997).*
- F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65:386–407, 1958.*