Incrementally Optimizing AUC

Zhi-Hua Zhou

http://cs.nju.edu.cn/zhouzh/
Email: zhouzh@nju.edu.cn

LAMDA Group
National Key Laboratory for Novel Software Technology,
Nanjing University, China

What are New Aspects?

Today I will talk about one issue …

See discussion in:

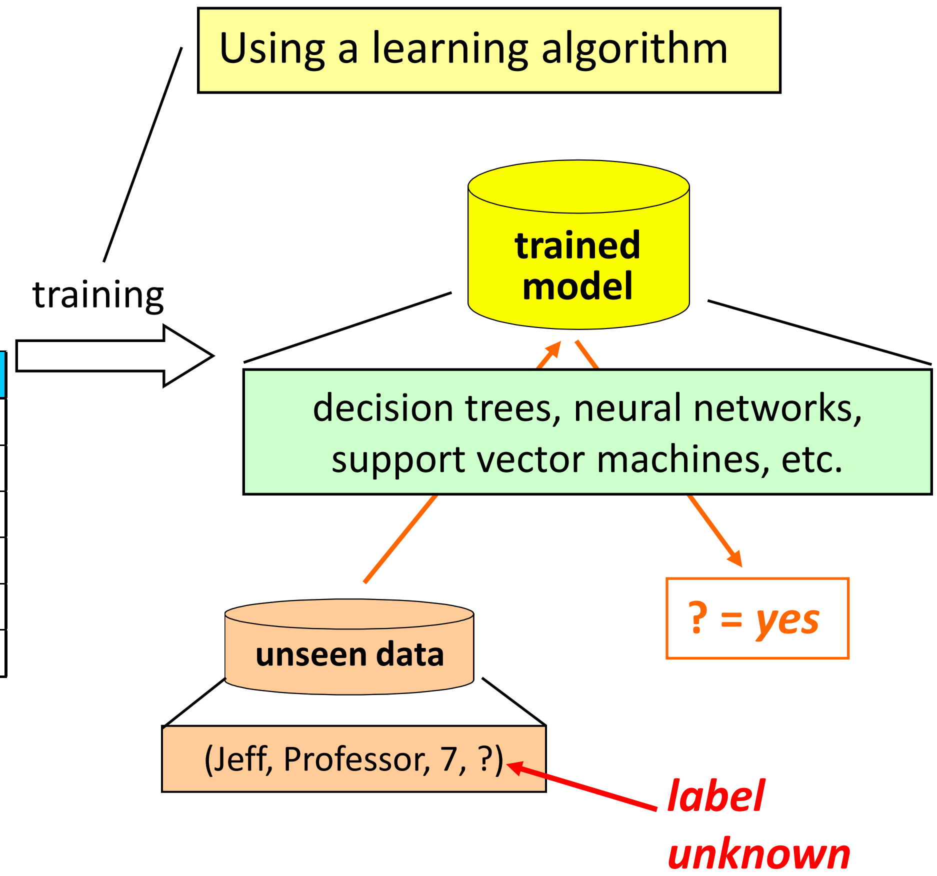
Z.-H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams. Big data opportunities and challenges: Discussions from data analytics perspectives. **IEEE Computational Intelligence Magazine**, 9(4): 62-74, November 2014.

A typical machine learning process

training data

label

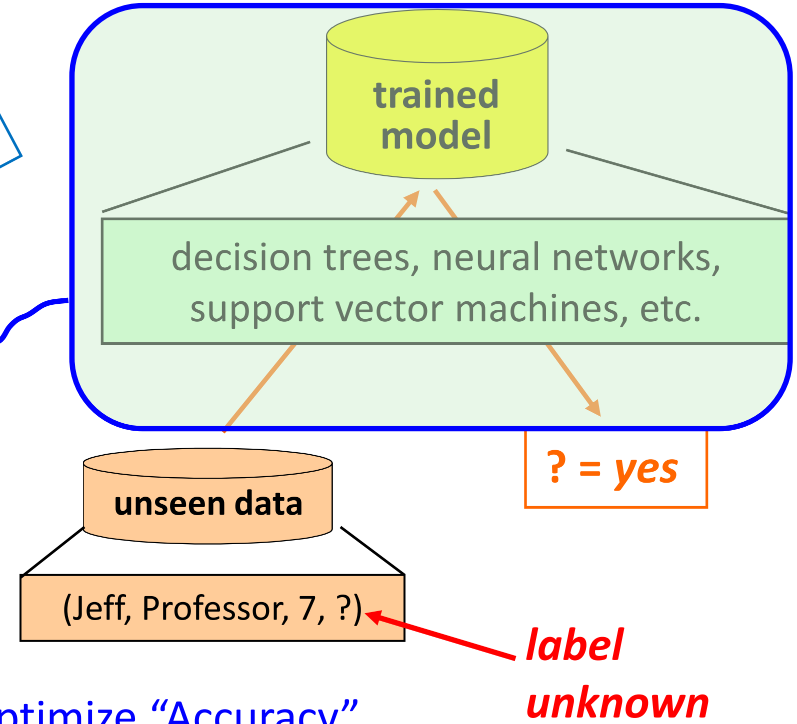
Name	Rank	Years	Tenured
Mike	Assistant Prof	3	no
Mary	Associate Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no



What model is good?

Performance measure

Related to the **performance measure** for evaluation



If we know which performance measure is concerned, we can take it as **optimization objective** for learning

The smaller #mistakes, the better ? → to optimize "Accuracy"

Both precision and recall are concerned ? → to optimize "F1"

… …

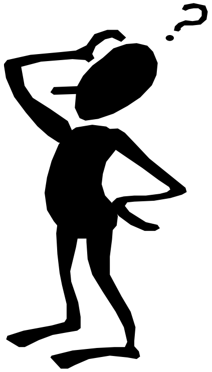
Optimizing the performance measure

Typically, we collect all training data, then do optimization

Now the data become bigger

Unwise to
"Get all data, then optimize"

What can we do ?



Incremental learning !

But …

Some performance measures are with good properties, easier to optimize, e.g.,

Accuracy:

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(y_i = f(\boldsymbol{x}_i))$$

It is **decomposable** over examples

Convex surrogate losses (hinge loss, exponential loss) make the optimization easier

However, many other important performance measures, e.g.,

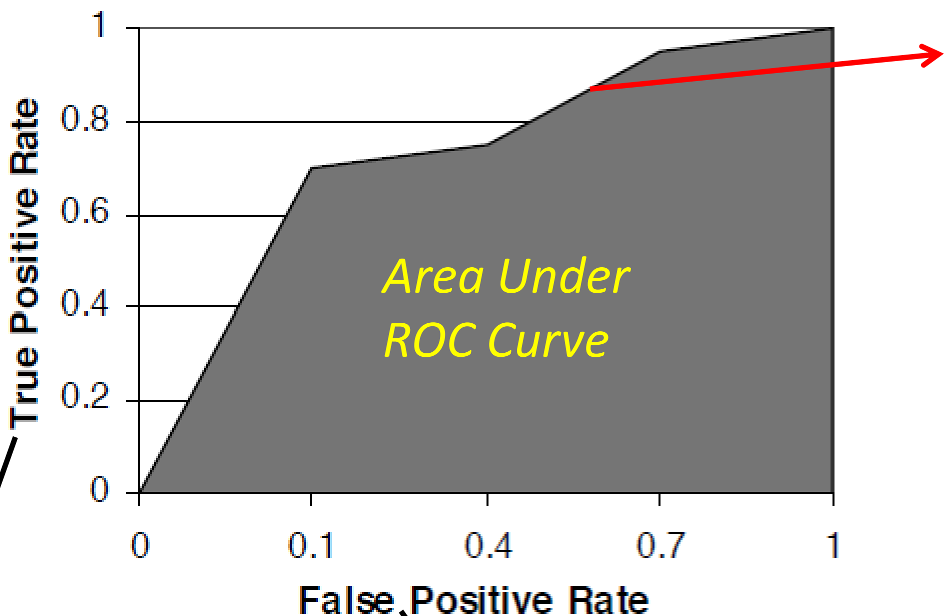
F1-score, PRBEP, MAP, AUC …

non-decomposable, non-linear, non-smooth, non-convex, …

Quite challenging

We focus on AUC as an example

AUC: **A**rea **U**nder the ROC **C**urve



ROC (Receiver Operating Characteristic) Curve [Green & Swets, Book 66; Spackman, IWML'89]

$$tpr = \frac{TP}{TP+FN} = \frac{TP}{m_+}$$

$$fpr = \frac{FP}{FP+TN} = \frac{FP}{m_-}$$

The bigger, the better

[Metz, SNM 78; Hanley & McNeil, Radiology 83]

AUC (con’t)

- ranking positive instances higher than negative ones
- insensitive to class distribution
- independent to classification threshold
- . . .

Widely used in various learning tasks, e.g.:

- ✓ Learning to Rank
- ✓ Class-imbalanced learning
- ✓ Cost-sensitive learning
- ✓ . . .

Optimizing the AUC

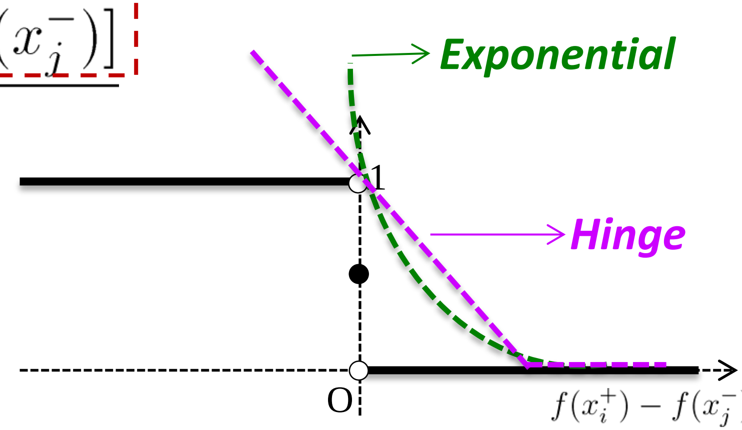
For function f and sample $S = \{(x_1^+, +1) \ldots (x_{n_+}^+, +1), (x_1^-, -1) \ldots (x_{n_-}^-, -1)\}$

AUC is given by

$$\sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \frac{\mathbb{I}[f(x_i^+) < f(x_j^-)] + \frac{1}{2}\mathbb{I}[f(x_i^+) = f(x_j^-)]}{n_+ n_-}$$

surrogate loss

$$\ell(f(x_i^+) - f(x_j^-))$$



Exponential [Freund et al., JMLR03; Rudin & Schapire JMLR09] Hinge [Joachims, KDD'06]

Pairwise loss

requires to store all data, and scan data many times

How to do incremental optimization ?

A natural idea: Online learning, stochastic gradient descent

easier for univariate losses (e.g., univariate Hinge loss, univariate exponential loss, univariate least square loss, etc.), **however, the performance is worse** (will show in exps)

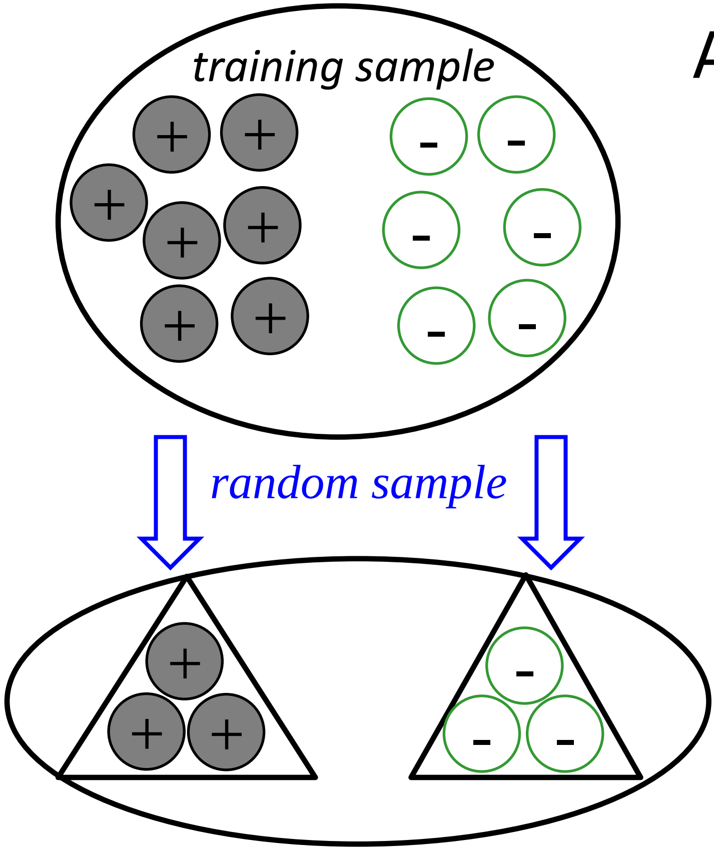
Pairwise loss: To encourage $f(\boldsymbol{x}_i^+) > f(\boldsymbol{x}_j^-)$

Univariate loss: To encourage $f(\boldsymbol{x}_i^+) > 0,\ f(\boldsymbol{x}_j^-) < 0$

univariate losses enforce unnecessary constraints

What can we do with pairwise loss?

To overcome the obstacle of pairwise calculation involving all data



A simple idea: To use a buffer

By using pairwise hinge loss, online AUC optimization with a buffer size $O(\sqrt{n_+ + n_-})$

[Zhao et al., ICML' 11]

Scan the buffer many times;
buffer size dependent to data size

AUC optimization approaches revisited

Current approaches:

Either: i) store all data, ii) scan all data many times

Or: i) buffer dependent to data size, ii) Scan buffer many times

Deficiencies, e.g.:

- For big data, buffer size will be big
- For streaming data, we do not know how many instances we will receive, and thus difficult to decide the buffer size

Can we have "One-Pass" approach:

- ✓ scan data only once
- ✓ storage not dependent to data size

OPAUC: One-Pass AUC Optimization

❶ We prove: Least square loss is consistent with AUC

❷ We derive: Two statistics are sufficient for AUC opt.

❸ We present: The OPAUC approach

Consistency of pairwise least square loss

Thm For finite instance space and least square loss $\ell(t) = (1-t)^2$, the surrogate loss $\Psi(f, x, x') = \ell(f(x) - f(x'))$ is consistent with AUC

Proof Outline: For $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ with margin probability p_i and conditional probability $\xi_i = \Pr[y_i = 1 | x_i]$

- Our goal is to minimize the expected risk

$$R_\Psi(f) = C_0 + \sum_{i \neq j} p_i p_j \left(\xi_i(1-\xi_j)\ell(f(\mathbf{x}_i) - f(\mathbf{x}_j)) + \xi_j(1-\xi_i)\ell(f(\mathbf{x}_j) - f(\mathbf{x}_i))\right)$$

- Based on sub-gradient conditions, we obtain *n* linear equations

$$\sum_{k \neq i} p_k(\xi_i + \xi_k - 2\xi_i\xi_k)(f(\mathbf{x}_i) - f(\mathbf{x}_k)) = \sum_{k \neq i} p_k(\xi_i - \xi_k) \text{ for each } 1 \leq i \leq n$$

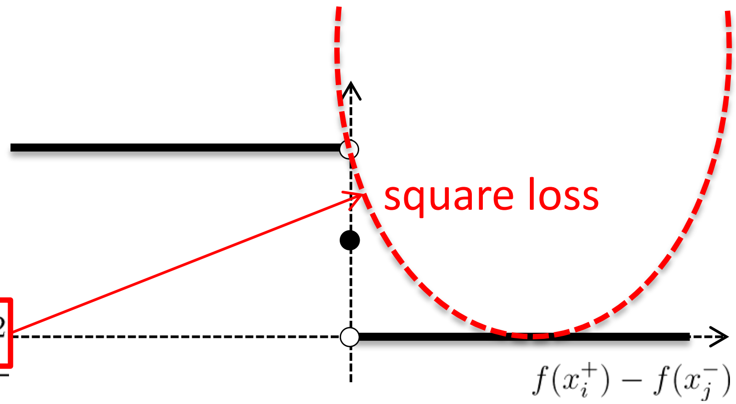
- Solving those linear equations, we get a Bayes solution

$$f(\mathbf{x}_i) - f(\mathbf{x}_j) = (\xi_i - \xi_j)\frac{\prod_{k \neq i,j} \sum_{l=1}^{n} p_l(\xi_l + \xi_k - 2\xi_l\xi_k)}{\sum_{\substack{s_i \geq 0 \\ s_1 + \cdots + s_n = n-2}} p_1^{s_1} \cdots p_n^{s_n} \Gamma(s_1, s_2, \cdots, s_n)}$$

where $\Gamma > 0$ is a polynomial in $(\xi_l + \xi_k - 2\xi_l\xi_k)$

Least square loss for AUC optimization

In each iteration, we receive a training example (x_t, y_t), and aim to optimize the least square loss



$$\mathcal{L}_t(w) = \frac{\lambda}{2}|w|^2 + \frac{\sum_{i=1}^{t-1} \mathbb{I}[y_i \neq y_t](1 - y_t(x_t - x_i)^\top w)^2}{2|\{i \in [t-1] : y_i y_t = -1\}|}$$

For stochastic gradient descent

$$w_{t+1} = w_t - \eta_t \nabla \mathcal{L}_t(w_t)$$

it is sufficient to calculate the gradient $\nabla \mathcal{L}_t(w_t)$

Two statistics and update

If $y_t = +1$ (similarly for $y_t = -1$), we have the gradient:

$$\nabla \mathcal{L}_t(w) = \lambda w - x_t + \underbrace{\sum_{i:\, y_i=-1} \frac{x_i}{n_t^-}}_{\text{neg. mean}} + \Big(x_t - \underbrace{\sum_{i:\, y_i=-1} \frac{x_i}{n_t^-}}_{\text{neg. mean}}\Big)\Big(x_t - \underbrace{\sum_{i:\, y_i=-1} \frac{x_i}{n_t^-}}_{\text{neg. mean}}\Big)^\top w$$

$$+ \underbrace{\Big(\sum_{i:\, y_i=-1} \frac{x_i x_i^\top}{n_t^-} - \sum_{i:\, y_i=-1} \frac{x_i}{n_t^-}\Big(\sum_{i:\, y_i=-1} \frac{x_i}{n_t^-}\Big)^\top\Big)}_{\text{neg. covariance}} w$$

Store the mean and covariance are sufficient !

Simple algebraic calculation for updating mean and covariance

Mean: One vector addition $c_t^+ = (1 - 1/T_t^+) c_{t-1}^+ + x_t / T_t^+$

Covariance: Four matrices additions

$$S_t^+ = (1 - 1/T_t^+) S_{t-1}^+ + x_t x_t^\top / T_t^+ + c_{t-1}^+ [c_{t-1}^+]^\top - c_t^+ [c_t^+]^\top$$

OPAUC

Learning And Mining from DatA
http://lamda.nju.edu.cn

Algorithm 1 The OPAUC Algorithm

Input: The regularization parameter $\lambda > 0$ and stepsizes $\{\eta_t\}_{t=1}^{n_+ + n_-}$.
Initialization: Set $\mathbf{w}_0 = \mathbf{0}$, $\mathbf{c}_0^+ = \mathbf{c}_0^- = \mathbf{0}$ and $S_0^+ = S_0^- = [\mathbf{0}]_{d \times d}$

for $t = 1, 2, \dots, n_+ + n_-$ **do**
  Receive a training example $(\mathbf{x}_t, y_t)$
  if $y_t = +1$ **then**
    Update the mean [$O(d)$] and covariance matrices [$O(d\times d)$] of positive instances
    Calculate the gradient $\nabla \mathcal{L}_t(\mathbf{w}_{t-1})$ from Eq. (4)
  else
    Update the mean [$O(d)$] and covariance matrices [$O(d\times d)$] of negative instances
    Calculate the gradient $\nabla \mathcal{L}_t(\mathbf{w}_{t-1})$ from Eq. (5)
  end if
  $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \nabla \mathcal{L}_t(\mathbf{w}_{t-1})$
end for

Storage: *$O(d\times d)$,* independent to data size

Scan data only once

How to handle large *d* ?

[Gao & Zhou, ICML'13]

To handle very high dimensions

Inspired by **random projection**

A straightforward idea:

i) high-dim. data $\xrightarrow{\text{rand. proj.}}$ low-dim. data

ii) apply OPAUC on the low-dim. data

Does NOT work !

Our approach

$$\begin{pmatrix} \star & \cdots & \star \\ \vdots & d \times d & \vdots \\ \star & \cdots & \star \end{pmatrix} = \begin{pmatrix} \triangle & \cdots & \triangle \\ \vdots & d \times n_t^+ & \vdots \\ \triangle & \cdots & \triangle \end{pmatrix} \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & n_t^+ \times n_t^+ & \vdots \\ 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \triangle & \cdots & \triangle \\ \vdots & d \times n_t^+ & \vdots \\ \triangle & \cdots & \triangle \end{pmatrix}^\top$$

$$\begin{pmatrix} 1 & \cdots & 0 \\ \vdots & n_t^+ \times n_t^+ & \vdots \\ 0 & \cdots & 1 \end{pmatrix} \approx \begin{pmatrix} \otimes & \cdots & \otimes \\ \vdots & n_t^+ \times \tau & \vdots \\ \otimes & \cdots & \otimes \end{pmatrix} \begin{pmatrix} \otimes & \cdots & \otimes \\ \vdots & n_t^+ \times \tau & \vdots \\ \otimes & \cdots & \otimes \end{pmatrix}^\top$$

$\otimes \sim N(0,1)$

τ is small

$$\begin{pmatrix} \star & \cdots & \star \\ \vdots & d \times d & \vdots \\ \star & \cdots & \star \end{pmatrix} \approx \begin{pmatrix} * & \cdots & * \\ \vdots & d \times \tau & \vdots \\ * & \cdots & * \end{pmatrix} \begin{pmatrix} * & \cdots & * \\ \vdots & d \times \tau & \vdots \\ * & \cdots & * \end{pmatrix}^\top$$

Low-rank approx. of covariance matrix

$OPAUC_r$

Theoretical results about convergence rates

Our theoretic analysis disclosed:

- For OPAUC (full covariance)
  - separable case: $O(1/T)$ $(T = n_+ + n_-)$
  - non-separable case: $O(1/\sqrt{T})$
- For OPAUC_r (approximated covariance)
  - separable case: $O(1/T)$
  - non-separable case: $O(1/\sqrt{T})$ + small constant

In contrast, previous approaches:

- Best convergence rate is at most $O(1/\sqrt{T})$ [Zhao et al., ICML'11]
- Dependent to $\frac{n_+}{n_-}$

Experimental data sets (I)

Benchmark data sets

datasets	#inst	#feat	datasets	#inst	#feat
diabetes	768	8	w8a	49,749	300
fourclass	862	2	kddcup04	50,000	65
german	1,000	24	mnist	60,000	780
splice	3,175	60	connect-4	67,557	126
usps	9,298	256	acoustic	78,823	50
letter	15,000	16	ijcnn1	141,691	22
magic04	19,020	10	epsilon	400,000	2,000
a9a	32,561	123	covtype	581,012	54

Comparison with existing methods

Online methods:

- **OAMseq**: pairwise hinge loss, sequential update, buffer size 100 [Zhao et al., ICML’11]
- **OAMgra**: pairwise hinge loss, gradient update, buffer size 100 [Zhao et al., ICML’11]

Batch methods:

- **SVM-perf**: structure SVM [Joachims, ICML’05]
- **SVM-OR**: pairwise hinge loss [Joachims, KDD’06]
- **Uni-Log**: univariate logistic loss [Kotlowski et al., ICML’11]

Results: Existing online methods

datasets	OPAUC	OAM_{seq}	OAM_{gra}
diabetes	.8309±.0350	.8264±.0367	.8262± .0338
fourclass	.8310±.0251	.8306±.0247	.8295±.0251
german	.7978±.0347	.7747±.0411•	.7723±.0358•
splice	.9232±.0099	.8594±.0194•	.8864±.0166•
usps	.9620±.0040	.9310±.0159•	.9348±.0122•
letter	.8114±.0065	.7549±.0344•	.7603±.0346•
magic04	.8383±.0077	.8238±.0146•	.8259±.0169•
a9a	.9002±.0047	.8420±.0174•	.8571±.0173•
w8a	.9633±.0035	.9304±.0074•	.9418±.0070•
kddcup04	.7912±.0039	.6918±.0412•	.7097±.0420•
mnist	.9242±.0021	.8615±.0087•	.8643±.0112•
connect-4	.8760±.0023	.7807±.0258•	.8128±.0230•
acoustic	.8192±.0032	.7113±.0590•	.7711±.0217•
ijcnn1	.9269±.0021	.9209±.0079•	.9100±.0092•
epsilon	.9550±.0007	.8816±.0042•	.8659±.0176•
covtype	.8244±.0014	.7361±.0317•	.7403±.0289•
win/tie/loss		**14/2/0**	**14/2/0**

OPAUC significantly better

Results: Existing batch methods

datasets	OPAUC	SVM-perf	batch SVM-OR	batch Uni-Log
diabetes	.8309±.0350	.8325±.0220	.8326±.0328	.8330±.0322
fourclass	.8310±.0251	.8221±.0381	.8305±.0311	.8288±.0307
german	.7978±.0347	.7952±.0340	.7935±.0348	.7995±.0344
splice	.9232±.0099	.9235±.0091	.9239±.0089	.9208±.0107•
usps	.9620±.0040	.9600±.0054•	.9630±.0047◦	.9637±.0041◦
letter	.8114±.0065	.8028±.0074•	.8144±.0064◦	.8121±.0061
magic04	.8383±.0077	.8427±.0078◦	.8426±.0074◦	.8378±.0073
a9a	.9002±.0047	.9033±.0039	.9009±.0036	.9033±.0025◦
w8a	.9633±.0035	.9626±.0042	.9495±.0082•	.9421±.0062•
kddcup04	.7912±.0039	.7935±.0037◦	.7903±.0039•	.7900±.0039•
mnist	.9242±.0021	.9338±.0022◦	.9340±.0020◦	.9334±.0021◦
connect-4	.8760±.0023	.8794±.0024◦	.8749±.0025•	.8784±.0026◦
acoustic	.8192±.0032	.8102±.0032•	.8262±.0032◦	.8253±.0032◦
ijcnn1	.9269±.0021	.9314±.0025◦	.9337±.0024◦	.9282±.0023◦
epsilon	.9550±.0007	.8640±.0049•	.8643±.0053•	.8647±.0150•
covtype	.8244±.0014	.8271±.0011◦	.8248±.0013	.8246±.0010
win/tie/loss		**4/6/6**	**4/6/6**	**4/6/6**

OPAUC:

- scan once
- store statistics

Batch:

- scan many times
- store whole data

OPAUC highly competitive

Additional comparison methods

How about online methods for other univariate surrogate loss?

- **Online Uni-Exp**：optimize univariate exponential loss
- **Online Uni-Squ**：optimize univariate least square loss

How about batch methods for least square loss?

- **LS-SVM**：optimize pairwise least square loss
- **Batch Uni-Squ**：optimize univariate least square loss

Results: Additional comparisons

Learning And Mining from DatA
http://lamda.nju.edu.cn

datasets	OPAUC	online Uni-Exp	online Uni-Squ	batch LS-SVM	batch Uni-Squ
diabetes	.8309±.0350	.8215±.0309•	.8258±.0354	.8325±.0329	.8332±.0323
fourclass	.8310±.0251	.8281±.0305	.8292±.0304	.8309±.0309	.8297±.0310
german	.7978±.0347	.7908±.0367	.7899±.0349	.7994±.0343	.7990±.0342
splice	.9232±.0099	.8931±.0213•	.9153±.0132•	.9245±.0092∘	.9211±.0107•
usps	.9620±.0040	.9538±.0045•	.9563±.0041•	.9634±.0045∘	.9617±.0043
letter	.8114±.0065	.8113±.0074	.8053±.0081•	.8124±.0065∘	.8112±.0061
magic04	.8383±.0077	.8354±.0099•	.8344±.0086•	.8379±0.0078	.8338±.0073•
a9a	.9002±.0047	.9005±.0024	.8949±.0025•	.8982±.0028•	.8967±.0028•
w8a	.9633±.0035	.7693±.0986•	.8847±.0130•	.9495±.0092•	.9075±.0104•
kddcup04	.7912±.0039	.7851±.0050•	.7850±.0042•	.7898±.0039•	.7926±.0038
mnist	.9242±.0021	.7932±.0245•	.9156±.0027•	.9336±.0025∘	.9279±.0021∘
connect-4	.8760±.0023	.8702±.0025•	.8685±.0033•	.8739±.0026•	.8760±.0024
acoustic	.8192±.0032	.8171±.0034•	.8193±.0035	.8210±.0033∘	.8222±.0031∘
ijcnn1	.9269±.0021	.9264±.0035	.9022±.0041•	.9320±.0037∘	.9038±.0025•
epsilon	.9550±.0007	.9488±.0012•	.9480±.0021•	.8644±.0050•	.8653±.0073•
covtype	.8244±.0014	.8236±.0017	.8236±.0020	.8222±.0014•	.8242±.0012
win/tie/loss		10/6/0	11/5/0	6/4/6	6/8/2

OPAUC:
significantly better than online

highly competitive with batch

Experimental data sets (II)

Very high-dimensional data sets

datasets	#inst	#feat	datasets	#inst	#feat
real-sim	72,309	20,985	sector.lvr	9,619	55,197
rcv1v2	23,149	47,236	news20	15,935	62,061
rcv	20,278	47,236	ecml2012	456,886	98,519
sector	9,619	55,197	news20.binary	19,996	1,355,191

Learning And Mining from DatA
http://lamda.nju.edu.cn

Results: High dimensional data (OPAUCr)

We also compared with:

- OPAUC^{f}: randomly select 1,000 features, then apply OPAUC
- OPAUC^{rp}: randomly project to 1,000 dim., then apply OPAUC
- OPAUC^{pca}: project to 1,000 dim. by PCA, then apply OPAUC

Does NOT work !

•/○ indicates OPAUCr is significantly better/worse

dim	1,355,191	98,519	62,061	55,197	55,197	*47,236*	*47,236*	*20,278*
datasets	news20.binary	ecml2012	news20	sector	sector.lvr	rcv	rcv1v2	real-sim
OPAUCr	.6389±.0136	.9828±.0008	.8871±.0083	.9292±.0081	.9962±.0011	.9831±.0016	.9686±.0029	.9789±.0010
OAM$_{seq}$	.6314±.0131•	N/A	.8543±.0099•	.9163±.0087•	.9965±.0064	.9885±.0010○	.9686±.0026	.9840±.0061○
OAM$_{gra}$	.6351±.0135•	.9657±.0055•	.8346±.0094•	.9043±.0100•	.9955±.0059•	.9852±.0019○	.9604±.0025•	.9762±.0062•
online Uni-Exp	.6347±.0092•	.9820±.0016•	.8880±.0047	.9215±.0034•	.9969±.0093	.9907±.0012○	.9822±.0042○	.9914±.0011○
online Uni-Squ	.6237±.0104•	.9530±.0041•	.8878±.0066	.9203±.0043•	.9669±.0260	.9918±.0010○	.9818±.0014○	.9920±.0009○
OPAUC^{f}	.5068±.0086•	.6601±.0036•	.5958±.0118•	.6228±.0145•	.6813±.0444•	.7297±.0069•	.6875±.0101•	.8105±.0042•
OPAUC^{rp}	.6212±.0072•	.9355±.0047•	.7885±.0079•	.7286±.0619•	.9863±.0258•	.9450±.0039•	.9353±.0053•	.9444±.0036•
OPAUC^{pca}	N/A	N/A	.8878±.0115	.8853±.0114•	.9893±.0288•	.9796±.0020•	.9752±.0020○	.9834±.0009○

OPAUCr: highly competitive , especially for very high-dim data

Another setting:

We have learned a model to do something (such as related to accuracy), but have not stored the data.

Now, we want to learn a model for another thing (such as related to AUC) … **What can we do?**

The problem setting

Suppose we have received *m* training examples, and constructed a classifier from these training examples; however, we have not stored these *m* examples

Now, we want to construct a classifier optimizing AUC

A straightforward option:

Using only the *n* examples received after the *m* examples to optimize AUC

The *m* examples ignored

What can we do ?

To exploit the *m* exps by adaptation !

Close relation between some measures

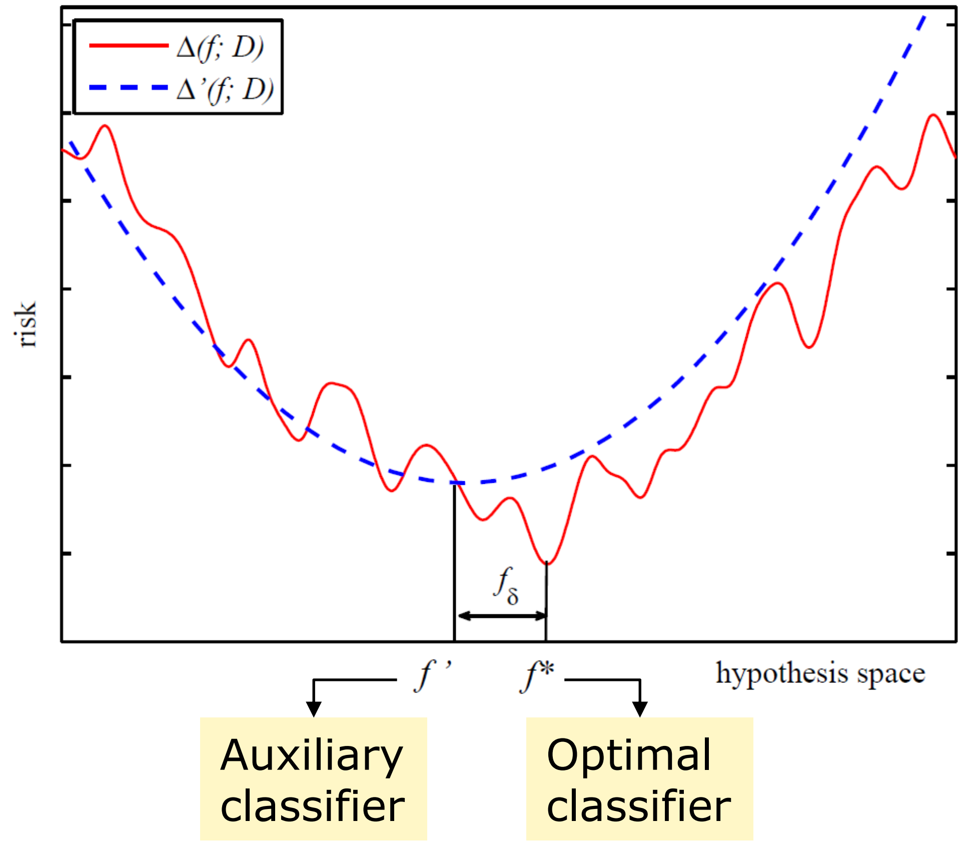
Many performance measures are closely-related

e.g., F1-score & PRBEP, AUC & Accuracy [Cortes & Mohri, NIPS'04]

⇨ If we have a classifier f' which optimizes accuracy, it can be regarded as a rough estimation of the classifier f^* which optimizes AUC, and thus it will be a good start point to find f^* in the function space

We adapt the "auxiliary classifier" f' for achieving f^*

Intuitive illustration



Auxiliary classifier (e.g., *f'* which maximizes accuracy) **can be helpful in finding the optimal classifier** (e.g., *f** which maximizes AUC)

The general strategy

We take the **classifier adaptation** strategy:

Auxiliary classifier f' —classifier adaptation→ **Target classifier** f → perform better w.r.t. concerned performance measure

In the function-level classifier adaption framework

Auxiliary classifier f' + **Delta function** f_δ => **Target classifier** f

Taking $f_\delta(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x})$, it follows

$$f(\mathbf{x}) = \mathrm{sign}\left[f'(\mathbf{x}) + \mathbf{w}^\top \Phi(\mathbf{x})\right]$$

The multivariate formulation

We take a multivariate formulation and considers to map a tuple of n instances $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ to a tuple of n labels $\bar{y} = (y_1, \ldots, y_n)$

$\Delta(\bar{y}, \bar{y}')$ is the loss by mapping $\bar{\mathbf{x}}$ to $\bar{y}'$ when its ground-truth is $\bar{y}$

Based on regularized risk minimization, we have the problem

$$\min_{\mathbf{w}} \Omega(\mathbf{w}) + C \cdot \Delta(\bar{y}, \bar{y}')$$

Regularizer Empirical risk

Usually non-convex, non-smooth, non-decomposable

The CAPO framework

A convex upper-bound over the empirical risk

$$R(\mathbf{w}; D) = \max_{\bar{y}' \in \mathcal{Y}^n} \left[F(\bar{\mathbf{x}}, \bar{y}') - F(\bar{\mathbf{x}}, \bar{y}) + \Delta(\bar{y}, \bar{y}')\right]$$

$$F(\bar{\mathbf{x}}, \bar{y}) = \begin{bmatrix} 1 \\ \mathbf{w} \end{bmatrix}^\top \Upsilon(\bar{\mathbf{x}}, \bar{y})$$

$$\Upsilon(\bar{\mathbf{x}}, \bar{y}) = \sum\nolimits_{i=1}^{n} y_i \begin{bmatrix} f'(\mathbf{x}_i) \\ \Phi(\mathbf{x}_i) \end{bmatrix}$$

By taking $\Omega(\mathbf{w}) = \|\mathbf{w}\|^2$, the optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, \xi \geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \forall\ \bar{y}' \in \mathcal{Y}^n \setminus \bar{y} : \\ & \begin{bmatrix} 1 \\ \mathbf{w} \end{bmatrix}^\top \left[\Upsilon(\bar{\mathbf{x}}, \bar{y}) - \Upsilon(\bar{\mathbf{x}}, \bar{y}')\right] \geq \Delta(\bar{y}, \bar{y}') - \xi \end{aligned}$$

⇨ CAPO finds the target classifier f **near the auxiliary classifier** f' such that f **minimizes the upper-bound of the empirical risk**

CAPO with multiple auxiliary classifiers

For multiple auxiliary classifiers, we construct an ensemble:

$$f(\mathbf{x}) = \mathrm{sign}\left[\sum\nolimits_{i=1}^{m} a_i f^i(\mathbf{x}) + \mathbf{w}^\top \Phi(\mathbf{x})\right]$$

Following the same strategy, we get :

$$\begin{aligned} \min_{\mathbf{a},\mathbf{w},\xi\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}B\|\mathbf{a}\|^2 + C\xi \\ \text{s.t.} \quad & \forall\, \bar{y}' \in \mathcal{Y}^n \setminus \bar{y} : \\ & \begin{bmatrix}\mathbf{a}\\ \mathbf{w}\end{bmatrix}^\top [\Psi(\bar{\mathbf{x}},\bar{y}) - \Psi(\bar{\mathbf{x}},\bar{y}')] \geq \Delta(\bar{y},\bar{y}') - \xi. \end{aligned}$$

$$\Psi(\bar{\mathbf{x}},\bar{y}) = \sum\nolimits_{i=1}^{n} y_i \begin{bmatrix}\mathbf{f}_i \\ \Upsilon(\mathbf{x}_i)\end{bmatrix}$$

⇨ It learns **an ensemble of auxiliary classifiers** and seeks the target classifier **near the ensemble**, such that **the upper-bound of the empirical risk is minimized**

The solution & algorithm

Single auxiliary classifier

$$\begin{aligned} \min_{\mathbf{w},\xi\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\xi \\ \text{s.t.} \quad & \forall\, \bar{y}' \in \mathcal{Y}^n \setminus \bar{y}: \\ & \begin{bmatrix} 1 \\ \mathbf{w} \end{bmatrix}^\top [\Upsilon(\bar{\mathrm{x}}, \bar{y}) - \Upsilon(\bar{\mathrm{x}}, \bar{y}')] \geq \Delta(\bar{y}, \bar{y}') - \xi \end{aligned}$$

Multiple auxiliary classifiers

$$\begin{aligned} \min_{\mathbf{a},\mathbf{w},\xi\geq 0} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}B\|\mathbf{a}\|^2 + C\xi \\ \text{s.t.} \quad & \forall\, \bar{y}' \in \mathcal{Y}^n \setminus \bar{y}: \\ & \begin{bmatrix} \mathbf{a} \\ \mathbf{w} \end{bmatrix}^\top [\Psi(\bar{\mathrm{x}}, \bar{y}) - \Psi(\bar{\mathrm{x}}, \bar{y}')] \geq \Delta(\bar{y}, \bar{y}') - \xi. \end{aligned}$$

By taking linear delta function, the problem can be efficiently solved by cutting-plane algorithm, which is similar to linear SVM-perf

The auxiliary classifier contributes to CAPO in two aspects:

- It injects nonlinearity, which is quite needed in practice;
- It provides an estimate of the target classifier, making the classifier adaption procedure more efficient

Experiments

- **20 tasks**
  - 5 datasets from different domains

    Face, text, gene, OCR…

DATA SET	#FEATURE	#TRAIN	#TEST
IJCNN1	22	49,990	91,701
Mitfaces	361	6,977	24,045
Reuters	8,315	7,770	3,299
Splice	60	1,000	2,175
USPS*	256	7,291	2,007

  - 4 performance measures (in addition to AUC, we also consider other performance measures)

    Accuracy, F1-score, PRBEP, AUC

  - 5 * 4 = 20

Compared methods

We compare their performance on 20 tasks

- **CAPO**
  - Auxiliary classifiers: CVM, RBF-NN, C4.5; (all with default parameters)
  - **CAPO_{cvm}, CAPO_{nn}, CAPO_{dt}, CAPO*** (B=1)
- **SVMperf** [Joachims, ICML'05]
  - Linear kernel & RBF kernel
- **SVM with cost model**
  - Linear kernel & RBF kernel, implemented by SVMlight

The parameter C, RBF kernel width, cost weights are selected via CV on training sets

Both the performance and the used CPU time are reported

Results: Performance

Performance of auxiliary classifiers

	TASK	CAPO_{cvm}	CAPO_{dt}	CAPO_{nn}	CAPO*	$\text{SVM}^{\text{perf}}_{\text{lin}}$	$\text{SVM}^{\text{perf}}_{\text{rbf}}$	$\text{SVM}^{\text{light}}_{\text{lin}}$	$\text{SVM}^{\text{light}}_{\text{rbf}}$
IJCNN1	*Accuracy*	.9540 (.9521)	.9702 (.9702)	.9150 (.8914)	.9703	.9193	.9658	N/A	N/A
	F1	.7620 (.7544)	.8473 (.8471)	.5753 (.2643)	.8468	.5565	N/A	N/A	N/A
	PRBEP	.7723 (.7376)	.8470 (.8364)	.5692 (.3222)	.8605	.6016	N/A	N/A	N/A
	AUC	.9607 (.8839)	.9734 (.9464)	.9198 (.8658)	.9810	.9180	N/A	N/A	N/A
Mitfaces	*Accuracy*	.9842 (.9839)	.9458 (.9302)	.9696 (.9067)	.9841	.9727	.9840	.9733	N/A
	F1	.4658 (.4665)	.1605 (.1342)	.2281 (.1768)	.4514	.2056	N/A	.2015	N/A
	PRBEP	.5127 (.4979)	.1864 (.1822)	.2500 (.1059)	.4873	.2140	N/A	.2309	N/A
	AUC	.9148 (.9148)	.7991 (.7201)	.8368 (.7979)	.9137	.8533	N/A	.8450	N/A
Reuters	*Accuracy*	.9745 (.9745)	.9664 (.9660)	.9715 (.9315)	.9739	.9727	.9727	.9724	.9721
	F1	.7730 (.7729)	.6973 (.6890)	.7455 (.1439)	.7731	.7375	N/A	.7599	.7540
	PRBEP	.7654 (.7709)	.7207 (.6871)	.7151 (.3743)	.7765	.7598	N/A	.7709	.7598
	AUC	.9870 (.9363)	.9842 (.9144)	.9868 (.8322)	.9838	.9878	N/A	.9872	.9873
Splice	*Accuracy*	.8947 (.8947)	.9347 (.9347)	.9651 (.9651)	.9664	.8451	.8947	.8446	.8975
	F1	.8955 (.8943)	.9371 (.9362)	.9659 (.9659)	.9512	.8451	N/A	.8487	.8990
	PRBEP	.8762 (.8691)	.9363 (.9355)	.9576 (.9558)	.9584	.8532	N/A	.8523	.9036
	AUC	.9457 (.8992)	.9760 (.9307)	.9836 (.9667)	.9852	.9304	N/A	.9267	.9639
Usps*	*Accuracy*	.9691 (.9689)	.9233 (.9233)	.8520 (.7798)	.9676	.8411	.9706	N/A	N/A
	F1	.9611 (.9613)	.9060 (.9053)	.8188 (.7486)	.9617	.8012	N/A	N/A	N/A
	PRBEP	.9500 (.9488)	.9000 (.8898)	.8195 (.7500)	.9573	.7963	N/A	N/A	N/A
	AUC	.9731 (.9658)	.9557 (.9179)	.9137 (.7582)	.9843	.9052	N/A	N/A	N/A

not completed in 24 hours

CAPO methods, especially CAPO*, achieve the best performance on most tasks

Results: Performance

Performance of auxiliary classifiers

	Task	$CAPO_{cvm}$	$CAPO_{dt}$	$CAPO_{nn}$	$CAPO^{*}$	SVM^{perf}_{lin}	SVM^{perf}_{rbf}	SVM^{light}_{lin}	SVM^{light}_{rbf}
IJCNN1	*Accuracy*	.9540 (.9521)	.9702 (.9702)	.9150 (.8914)	**.9703**	.9193	.9658	N/A	N/A
	F1	.7620 (.7544)	**.8473** (.8471)	.5753 (.2643)	.8468	.5565	N/A	N/A	N/A
	PRBEP	.7723 (.7376)	.8470 (.8364)	.5692 (.3222)	**.8605**	.6016	N/A	N/A	N/A
	AUC	.9607 (.8839)	.9734 (.9464)	.9198 (.8658)	**.9810**	.9180	N/A	N/A	N/A
Mitfaces	*Accuracy*	**.9842** (.9839)	.9458 (.9302)	.9696 (.9067)	.9841	.9727	.9840	.9733	N/A
	F1	**.4658** (.4665)	.1605 (.1342)	.2281 (.1768)	.4514	.2056	N/A	.2015	N/A
	PRBEP	**.5127** (.4979)	.1864 (.1822)	.2500 (.1059)	.4873	.2140	N/A	.2309	N/A
	AUC	**.9148** (.9148)	.7991 (.7201)	.8368 (.7979)	.9137	.8533	N/A	.8450	N/A
Reuters	*Accuracy*	**.9745** (.9745)	.9664 (.9660)	.9715 (.9315)	.9739	.9727	.9727	.9724	.9721
	F1	.7730 (.7729)	.6973 (.6890)	.7455 (.1439)	**.7731**	.7375	N/A	.7599	.7540
	PRBEP	.7654 (.7709)	.7207 (.6871)	.7151 (.3743)	**.7765**	.7598	N/A	.7709	.7598
	AUC	.9870 (.9363)	.9842 (.9144)	.9868 (.8322)	.9838	**.9878**	N/A	.9872	.9873
Splice	*Accuracy*	.8947 (.8947)	.9347 (.9347)	.9651 (.9651)	**.9664**	.8451	.8947	.8446	.8975
	F1	.8955 (.8943)	.9371 (.9362)	**.9659** (.9659)	.9512	.8451	N/A	.8487	.8990
	PRBEP	.8762 (.8691)	.9363 (.9355)	.9576 (.9558)	**.9584**	.8532	N/A	.8523	.9036
	AUC	.9457 (.8992)	.9760 (.9307)	.9836 (.9667)	**.9852**	.9304	N/A	.9267	.9639
Usps*	*Accuracy*	.9691 (.9689)	.9233 (.9233)	.8520 (.7798)	.9676	.8411	**.9706**	N/A	N/A
	F1	.9611 (.9613)	.9060 (.9053)	.8188 (.7486)	**.9617**	.8012	N/A	N/A	N/A
	PRBEP	.9500 (.9488)	.9000 (.8898)	.8195 (.7500)	**.9573**	.7963	N/A	N/A	N/A
	AUC	.9731 (.9658)	.9557 (.9179)	.9137 (.7582)	**.9843**	.9052	N/A	N/A	N/A

not completed in 24 hours

CAPO achieves performance improvements over auxiliary classifiers

Results: Time cost

	Task	CAPO_{cvm}	CAPO_{dt}	CAPO_{nn}	CAPO*	$\text{SVM}^{\text{perf}}_{\text{lin}}$	$\text{SVM}^{\text{perf}}_{\text{rbf}}$	$\text{SVM}^{\text{light}}_{\text{lin}}$	$\text{SVM}^{\text{light}}_{\text{rbf}}$
IJCNN1	*Accuracy*	9.3	11.1	9.9	11.2	10.0	96.6	N/A	N/A
	F1	9,451.5	9,011.5	14,809.3	6,652.8	12,281.3	N/A		
	PRBEP	1,507.9	1,033.3	2,276.2	1,005.1	2,034.0	N/A		
	AUC	88.0	38.0	124.0	40.6	112.6	N/A		
Mitfaces	*Accuracy*	9.5	11.2	23.7	9.0	27.2	27,089.3	6,114.7	N/A
	F1	465.6	802.5	1,211.5	379.0	1,189.4	N/A		
	PRBEP	126.9	183.4	241.6	119.6	234.4	N/A		
	AUC	37.7	48.5	74.0	30.6	79.3	N/A		
Reuters	*Accuracy*	5.7	2.1	2.6	3.9	2.3	39,813.1	283.1	53,113.8
	F1	68.7	67.4	64.3	67.6	60.2	N/A		
	PRBEP	10.8	13.1	11.9	10.6	11.4	N/A		
	AUC	18.9	8.6	8.7	3.9	8.1	N/A		
Splice	*Accuracy*	4.0	484.5	697.1	2.0	3,602.4	2,187.1	16,297.6	464.2
	F1	168.2	592.3	3,373.9	58.4	10,201.5	N/A		
	PRBEP	11.8	17.0	27.3	6.8	82.6	N/A		
	AUC	2.0	3.3	7.3	1.2	42.0	N/A		
Usps*	*Accuracy*	24.6	35.4	215.3	15.6	221.5	24,026.7	N/A	N/A
	F1	2,199.0	2,605.4	5,429.9	1,514.8	5,225.9	N/A		
	PRBEP	626.2	566.1	938.9	404.4	895.2	N/A		
	AUC	155.6	139.9	424.3	76.1	452.5	N/A		

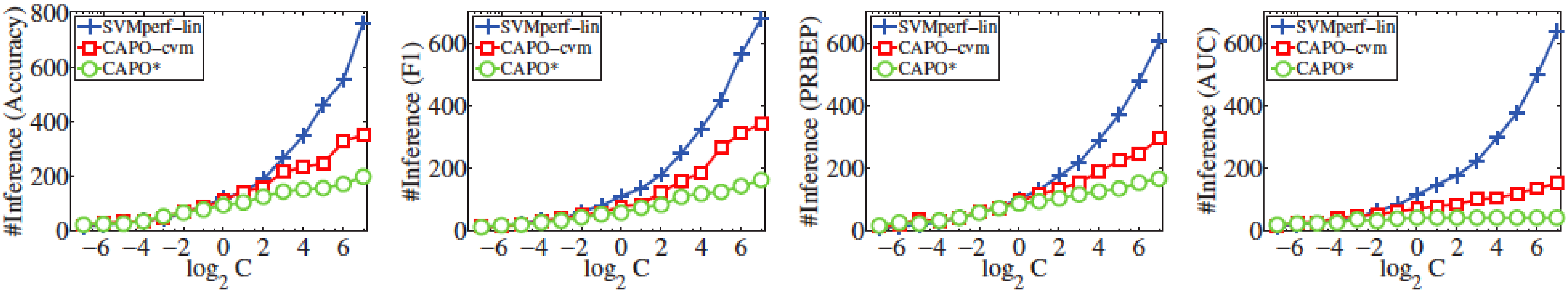
not completed in 24 hours

CAPO is even more efficient than linear SVMperf

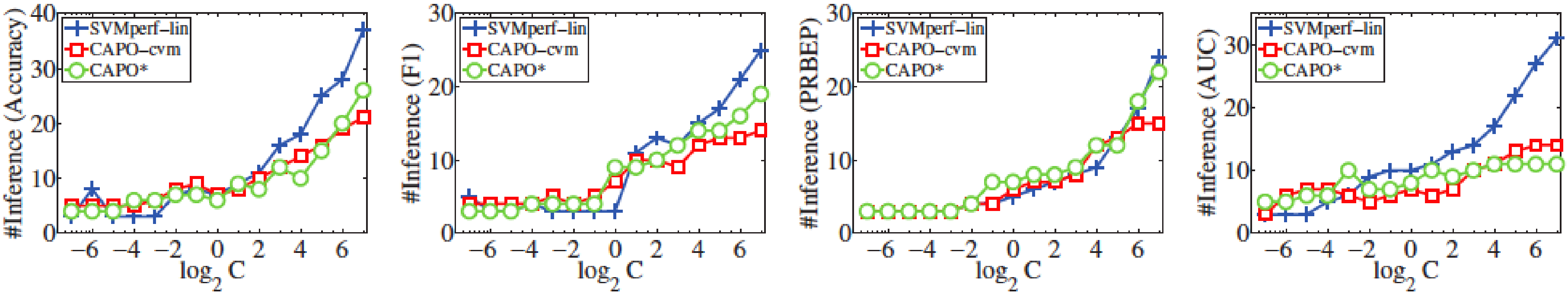
Learning And Mining from DatA
http://lamda.nju.edu.cn

Why CAPO more efficient

The number of inference iterations:



(a) Number of inferences on USPS*



(b) Number of inferences on Reuters

Technical Summary

- **One-pass optimization** *(one scan, storage independent to data size)*:
  - ✓ Pairwise least square loss is consistent with AUC
  - ✓ Two statistics are sufficient for AUC optimization
  - ✓ For high-dim data, sparse approx of covariance matrices

- **Adaptational optimization**:
  - ✓ Benefits from classifier optimizing closely-related measures
  - ✓ Optimizing a convex upper-bound over the empirical risk
  - ✓ Multiple auxiliary classifiers increase robustness

We take AUC for example, but the ideas can be extended to other performance measures (some are future work)

A further exploration of incremental learning

It is specified in Zhou & Chen, Hybrid decision tree, *Knowledge-Based Systems*, 2002, vol.15, no.8, pp.515-528

- ***E-IL*** *(Example-Incremental Learning):* ***New training examples*** *are provided after a learning system being trained*

- ***C-IL*** *(Class-Incremental Learning):* ***New output classes*** *are provid … a learning system being trained*

- ***A-IL*** *(Attribute-Incremental Learning):* ***New input att*** … *provided after a learning system being trained*

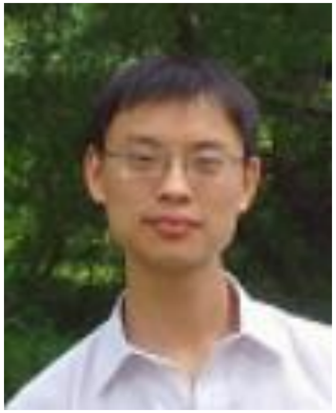
Few studies on C-IL, A-IL

A recent study on **C-IL**:

Da, Yu & Zhou. Learning with augmented class by exploiting unlabeled data. AAAI 2014

The talk involves joint work with

My students:



Wei Gao
(高尉)

Nan Li
(李楠)

My collaborators:

Rong Jin, Ivor W. Tsang, Shenghuo Zhu

Take-Home Messages

To handle big data:

- **Incremental learning is important**
- **Least square loss is even more useful than before**
- **Relevant, previously trained models can be helpful**

Codes available:

- OPAUC: http://lamda.nju.edu.cn/code_OPAUC.ashx
- CAPO: http://lamda.nju.edu.cn/code_CAPO.ashx

Thanks!