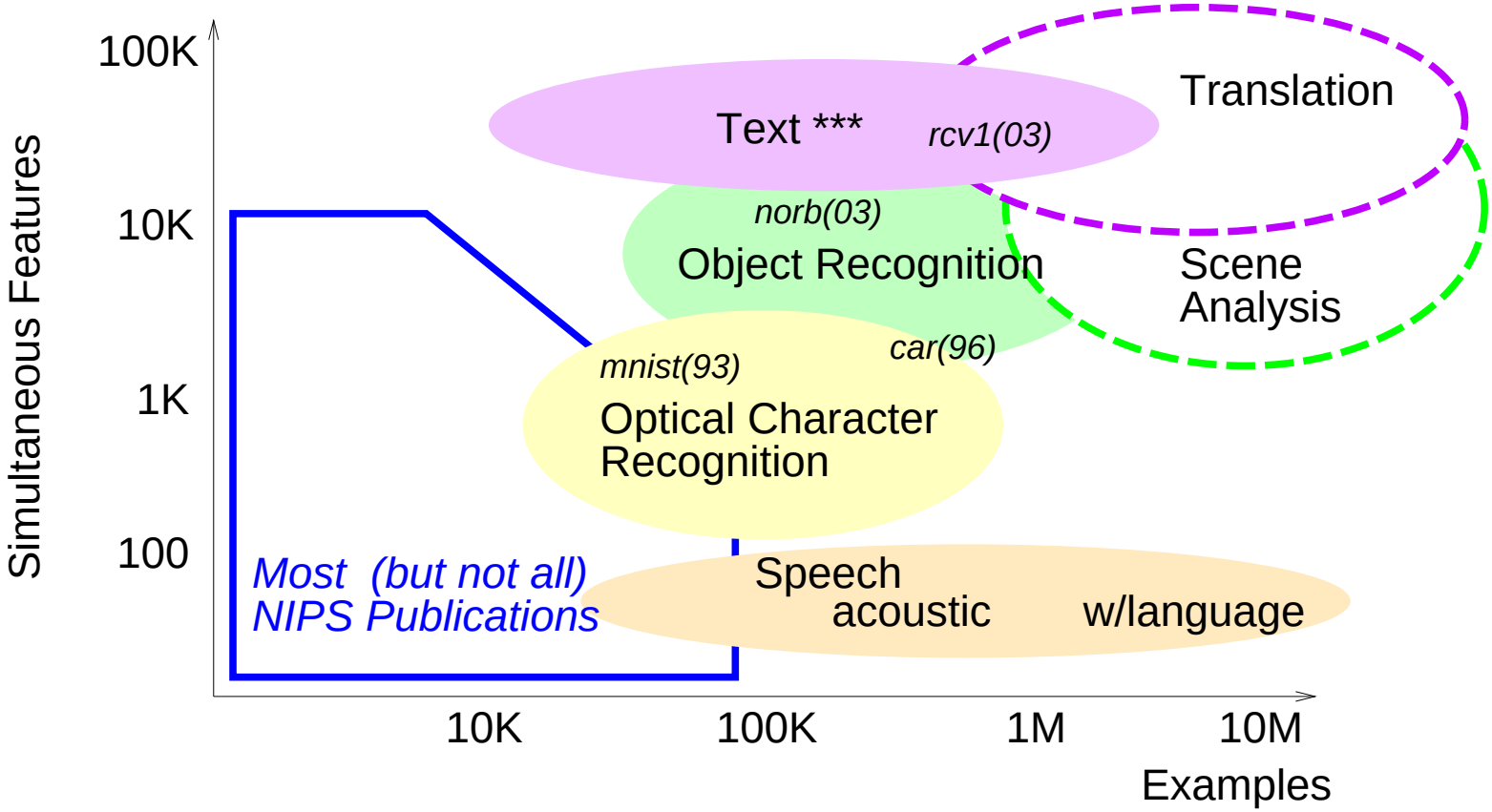
Fast Kernel Classifiers with Online and Active Learning

Léon Bottou,

Antoine Bordes[1], **Seyda Ertekin**[2],
Gaelle Loosli[3], **Jason Weston**,

NEC Labs America, Princeton NJ,
(1) also with ESPCI, (Paris, France.)
(2) also with Penn State University, (PA.)
(3) also with INSA Rouen, (France.)

Motivation



- 10 years: cpu speed $\times 100$, disk size $\times 1000$.
- linear vs non-linear

Summary

I. Quick Review

II. Online SVM algorithm

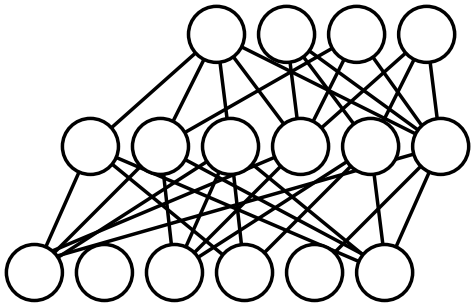
(with soft-margins, budget, etc.).

III. Active SVM algorithm.

IV. Discussion

(Practical uses, Convergence results, Open questions.)

Stochastic Gradient Learning



- Non linear, non convex.
- Practical tricks.

  e.g. (LeCun, B. , Müller, Orr 98)

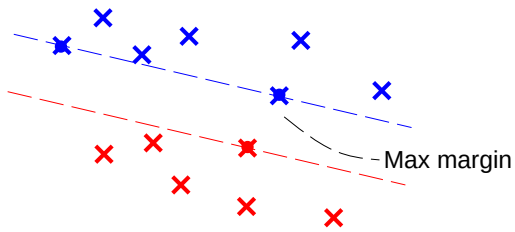
- Memory D^{β} with $1 \leq \beta \leq 2$.
- Speed ND^{α} with $1 < \alpha \leq 2$

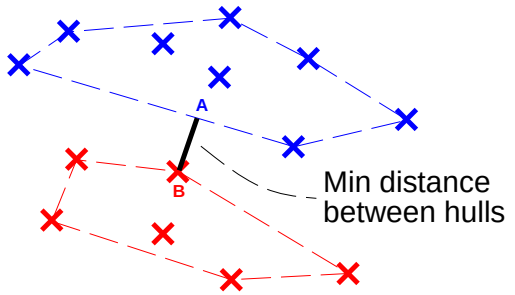
  (B. 91) (Murata & Amari 98)

  (B. & LeCun 04) ††

- How does D grows with N?

Learning in the dual



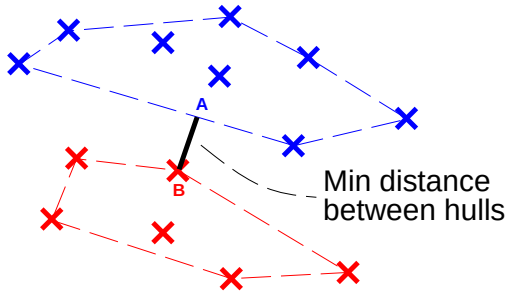


- Convex, Kernel trick.
- Memory NK
- Speed $N^{\alpha}K$ with $1 < \alpha \leq 2$
- Bad news $K \sim 2\mathcal{B}N$

  (Steinwart 04)
- K could be much smaller.

  (Burges 93) (Vincent & Bengio, 02)
- How to do it fast?
- How small?

Kernel Trick



- **Decision Function:**
  $\widehat{y}(x) = \sum_{i \in \mathcal{S}} \alpha_i \; x_i \cdot x + b$
- Support Vectors: $\alpha_i \neq 0$

- **Kernel:** Replace $x \cdot y$ by a "Mercer Kernel" $K(x, y)$.
- Implicit mapping $\Phi(\cdot)$ such that $K(x, y) = \Phi(x) \cdot \Phi(y)$
- Typical kernels:

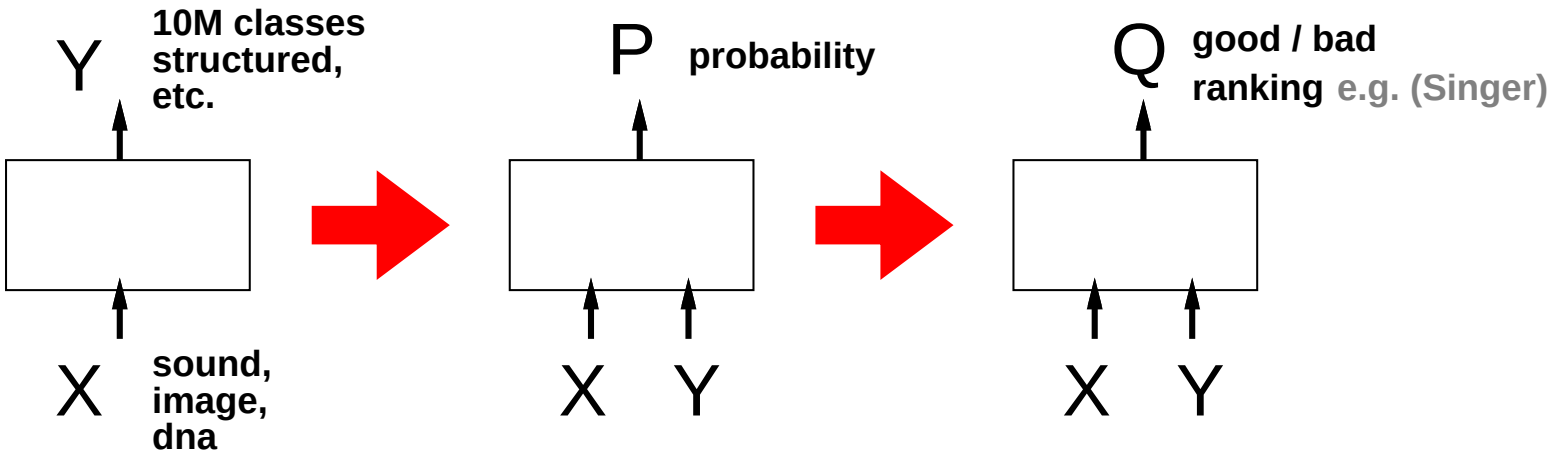
$$\begin{cases} \text{Linear} & K(x, y) = x \cdot y \\ \text{Polynomial} & K(x, y) = (\lambda + x \cdot y)^n \\ \text{RBF} & K(x, y) = \exp\left(-\gamma \|x - y\|^2\right) \end{cases}$$

(Aronzajn, 50) (Aizerman et al. 64) (Boser, Guyon, Vapnik, 91)

. . .

Binary Classifiers

Complicated problems can be turned into (almost) binary classifiers.



(Taskar et al. 04), (Altun et al. 04), (LeCun et al. 05)

Kernel Perceptrons (Aizerman et al., 64)

1) **Initialization**: $\mathcal{S} \leftarrow \emptyset$, $\boldsymbol{\alpha} \leftarrow \mathbf{0}$

2) **Online Iterations**:

Repeat a predefined number of times:

- Pick a fresh example $\boldsymbol{x}_t$
- Compute $\widehat{y}(\boldsymbol{x}_t)$
- If $y_t \ \widehat{y}(\boldsymbol{x}_t) \ \leq 0$ then $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}$, $\alpha_t \leftarrow y_t$

- Separable data: stops in finite time (Novikov,62).
- One SV for each mistake.
- Extension: SV removal step (Crammer et al., 04)

Support Vector Machines

- **Primal formulation:**

  Maximize margin (soft-margins, hinge loss)

- **Dual formulation:**

$$\max_{\boldsymbol{\alpha}} \; W(\boldsymbol{\alpha}) = \sum_i \alpha_i y_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j)$$

$$\text{with} \quad \begin{cases} \bullet \; \sum_i \alpha_i = 0 \\ \bullet \; \text{or} \quad \begin{array}{rl} 0 \leq \alpha_i \leq C & \text{when } y_i = +1 \\ -C \leq \alpha_i \leq 0 & \text{when } y_i = -1 \end{array} \end{cases}$$

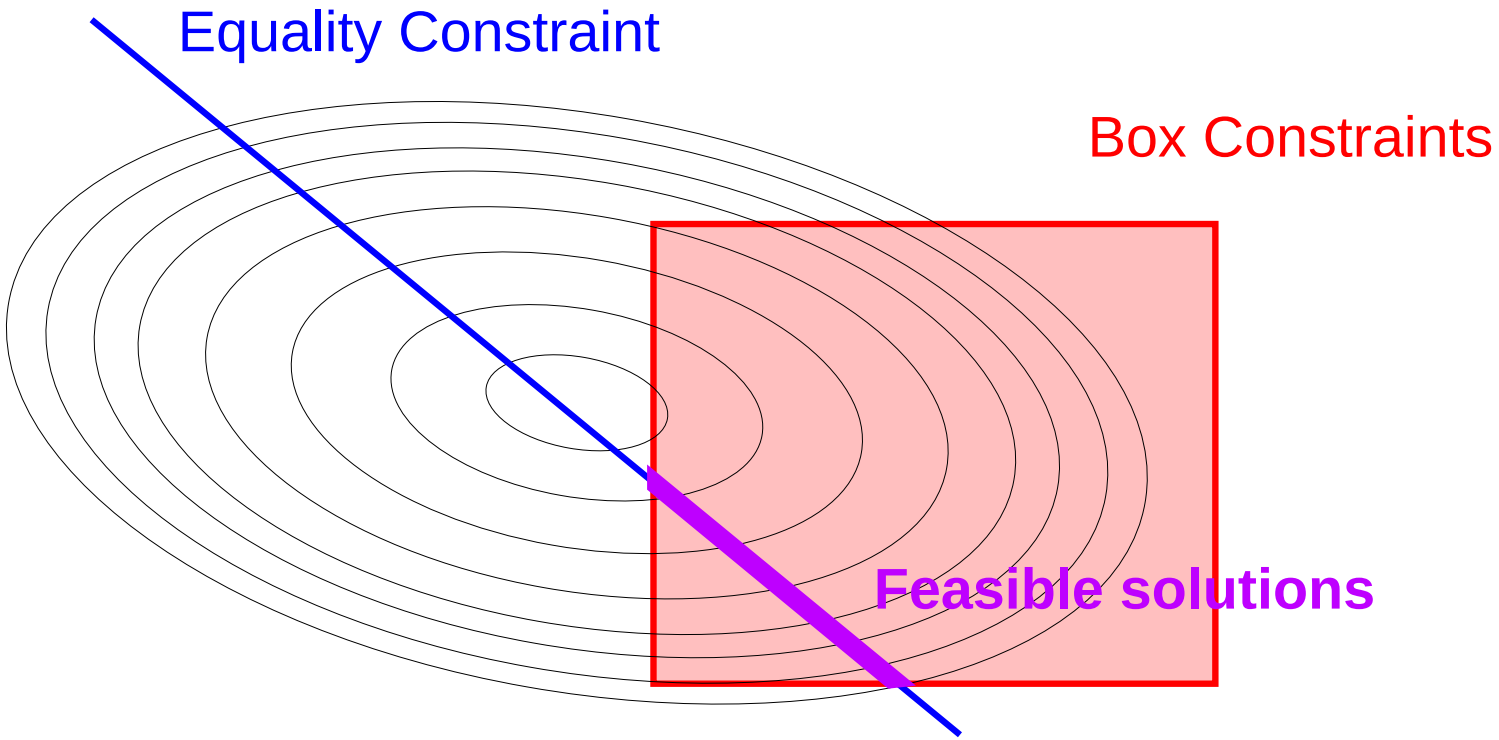
Roadmap

	Memory	Time	SVs
SVM	NK	$N^{\alpha}K$	$K \sim 2\mathcal{B}N$
Online SVM	K^2	NK †	$K \sim 2\mathcal{B}N$
Active SVM	K^2	NK	$K \ll 2\mathcal{B}N$ †

† caveats. . .

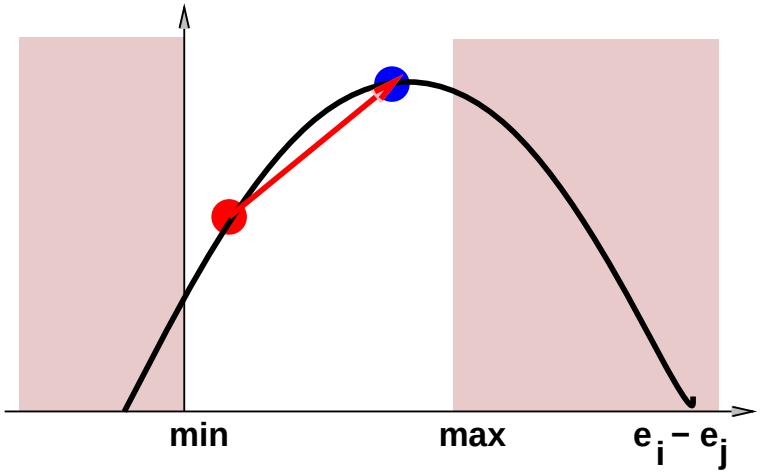
Solving the SVM

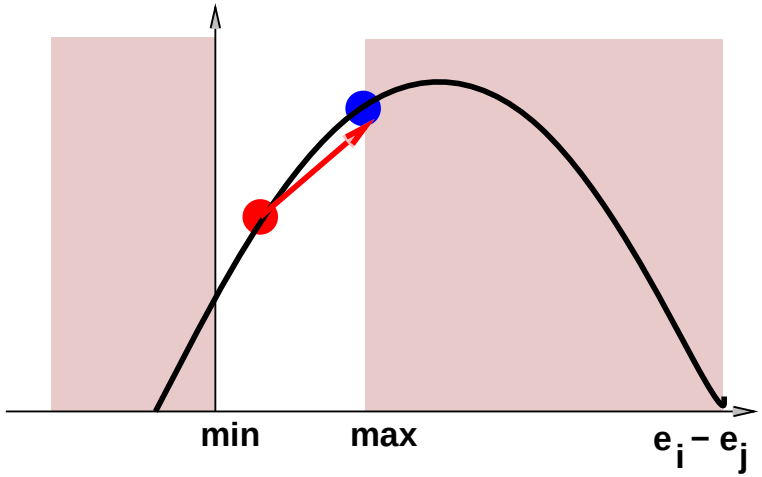
Quadratic Programming Problem.



Sequential Newton steps

- Choose a feasible direction in α-space.





- Iterate

SMO Algorithm

Idea: Perform Newton steps in $\boldsymbol{\alpha}$ space along directions that only involve two coefficients.

1) **Initialization**: $\mathcal{S} \leftarrow \emptyset$, $\boldsymbol{\alpha} \leftarrow \mathbf{0}$, $\boldsymbol{g} = (y_i)$

2) **SMO Iterations**:

Repeat until convergence:

- Select two examples x_i and x_j using gradients.
- Perform a Newton step along $\boldsymbol{u} = \boldsymbol{e}_i - \boldsymbol{e}_j$ clipping to constraints.
- Update gradients $g_s \leftarrow g_s - \Delta_{\alpha_i} K(\boldsymbol{x}_i, \boldsymbol{x}_s) - \Delta_{\alpha_j} K(\boldsymbol{x}_j, \boldsymbol{x}_s)$

Particular case of **Sequential Direction Search.**

Reorganizing SMO $\rightarrow$ Online SVM

State variables.

— Kernel expansion.

  – Set of support vectors $\mathcal{S}$.

  – Coefficients α_i, $i \in \mathcal{S}$

— Gradients w.r.t Support Vector Coefficients.

$$g_i = \frac{dW}{d\alpha_i},\ i \in \mathcal{S}$$

PROCESS

Pick a direction $\boldsymbol{e}_i - \boldsymbol{e}_j$ defined by two examples:

- One random example not in expansion $\mathcal{S}$.
- One matching example from the expansion $\mathcal{S}$, such that $\boldsymbol{e}_i - \boldsymbol{e}_j$ is the feasible direction with the strongest gradient.

Effect: First example might be added to $\mathcal{S}$.

PROCESS(k)

1) Bail out if $k \in \mathcal{S}$.

2) $\alpha_k \leftarrow 0$, $\quad g_k \leftarrow y_k - \sum_{s \in \mathcal{S}} \alpha_s K_{ks}$, $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$

3) If $y_k = +1$ then

$\quad i \leftarrow k$, $\quad j \leftarrow \arg\min_{s \in S} g_s$ with $\alpha_s > A_s$

else

$\quad j \leftarrow k$, $\quad i \leftarrow \arg\max_{s \in S} g_s$ with $\alpha_s < B_s$

4) Bail out if $\boldsymbol{e}_i - \boldsymbol{e}_j$ is not a feasible direction.

5) $\lambda \leftarrow \min \left\{ \dfrac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}},\ B_i - \alpha_i,\ \alpha_j - A_j \right\}$

$\alpha_i \leftarrow \alpha_i + \lambda$, $\quad \alpha_j \leftarrow \alpha_j - \lambda$

$g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall\, s \in \mathcal{S}$

REPROCESS

Pick a direction $\boldsymbol{e}_i - \boldsymbol{e}_j$ defined by two examples:

- Both examples chosen from the expansion $\mathcal{S}$, such that $\boldsymbol{e}_i - \boldsymbol{e}_j$ is the feasible direction with the strongest gradient.

Effect: Examples might be removed from $\mathcal{S}$.

REPROCESS

1) $i \leftarrow \arg\max_{s \in \mathcal{S}} g_s$ with $\alpha_s < B_s$
   $j \leftarrow \arg\min_{s \in \mathcal{S}} g_s$ with $\alpha_s > A_s$
2) Bail out if $\boldsymbol{e}_i - \boldsymbol{e}_j$ is not a feasible direction.
3) $\lambda \leftarrow \min \left\{ \dfrac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}},\ B_i - \alpha_i,\ \alpha_j - A_j \right\}$

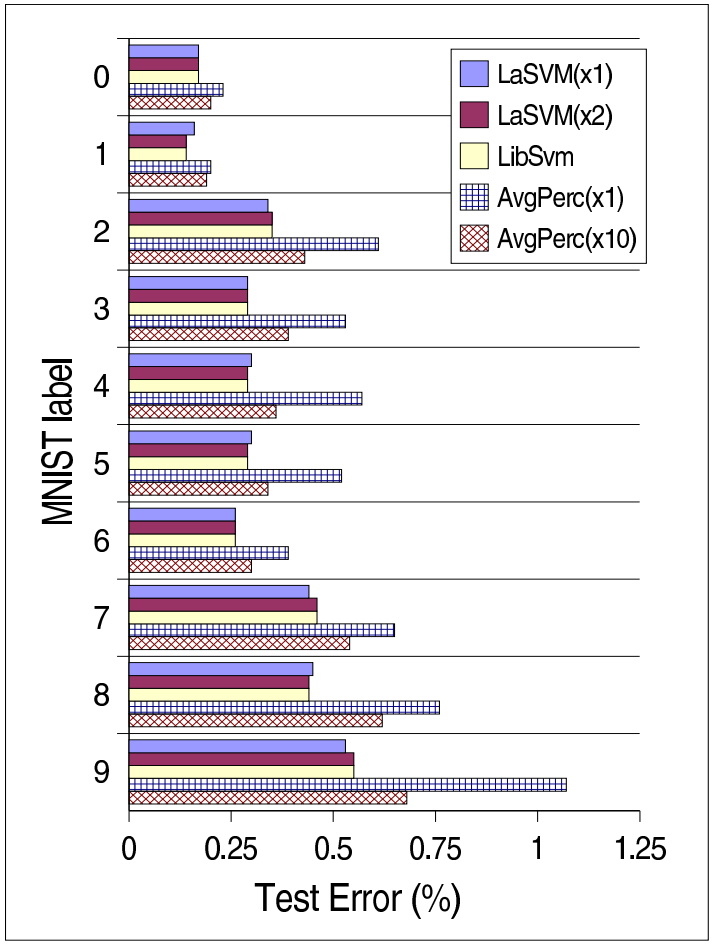
   $\alpha_i \leftarrow \alpha_i + \lambda$, $\alpha_j \leftarrow \alpha_j - \lambda$
   $g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall\, s \in \mathcal{S}$
4) $i \leftarrow \arg\max_{s \in \mathcal{S}} g_s$ with $\alpha_s < B_s$
   $j \leftarrow \arg\min_{s \in \mathcal{S}} g_s$ with $\alpha_s > A_s$
   For all $s \in \mathcal{S}$ such that $\alpha_s = 0$
      If $y_s = -1$ and $g_s \geq g_i$ then $\mathcal{S} = \mathcal{S} - \{s\}$
      If $y_s = +1$ and $g_s \leq g_j$ then $\mathcal{S} = \mathcal{S} - \{s\}$
5) $b \leftarrow (g_i + g_j)/2$, $\delta \leftarrow g_i - g_j$

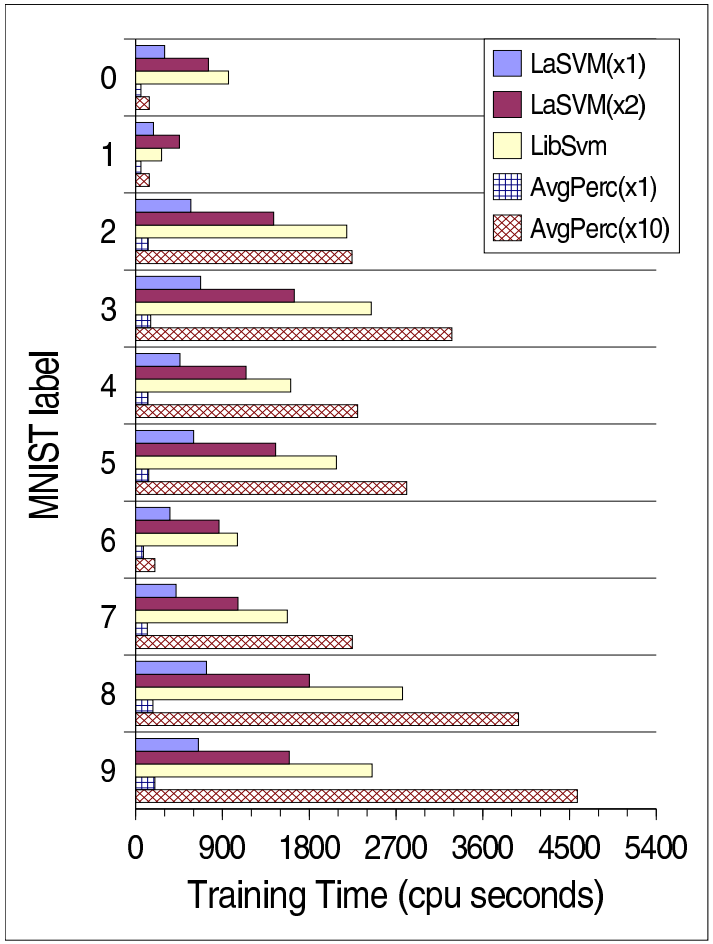
LASVM

1) **Initialization**:
   Seed $\mathcal{S}$ with a few examples of each class.
   Set $\boldsymbol{\alpha} \leftarrow \mathbf{0}$ and compute $\boldsymbol{g}$.
2) **Online Iterations**:
   Repeat a predefined number of times:
   - Pick an example k_t
   - Run PROCESS(k_t).
   - Run REPROCESS once.
3) **Finishing**:
   Repeat REPROCESS until $\delta \leq \tau$.

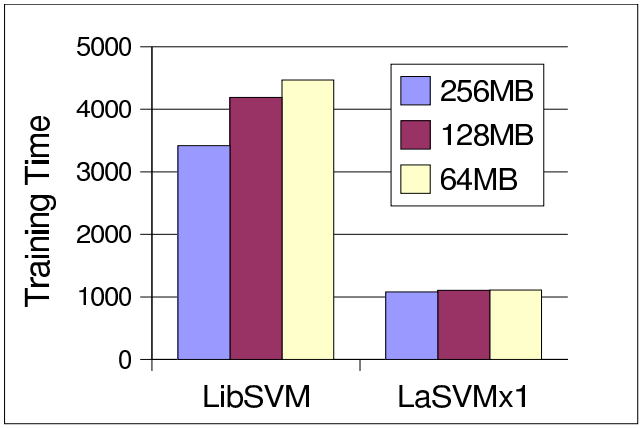
LASVM on MNIST

60000+10000 handwritten digits



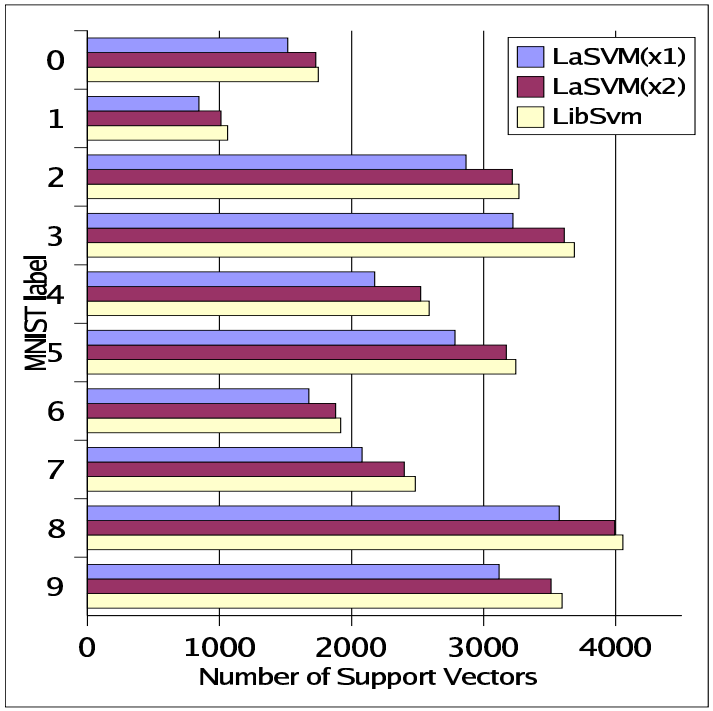


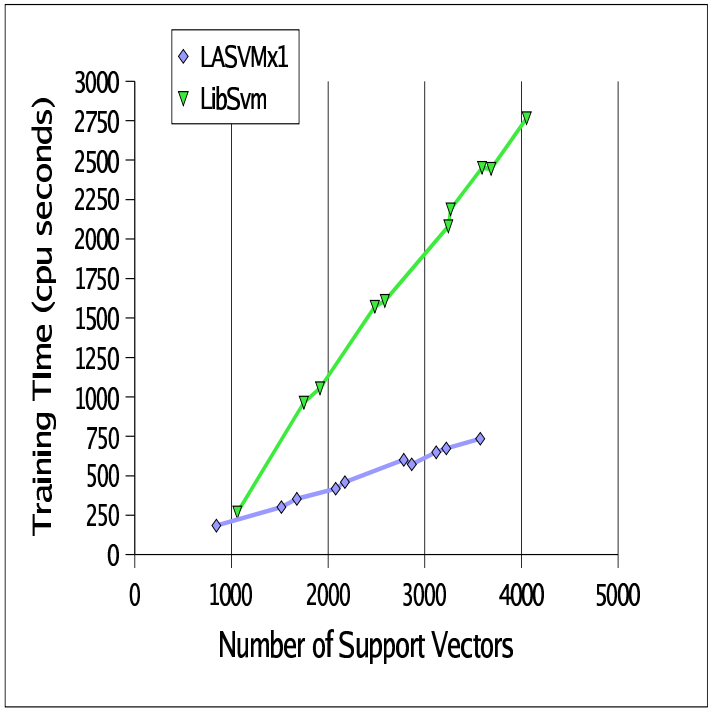
LASVM on MNIST



Algorithm	Error	Time
LIBSVM	**1.36%**	17400s
LASVM×1	1.42%	**4950s**
LASVM×2	**1.36%**	12210s

LASVM on MNIST





LASVM on Various Datasets

	Train Size	Test Size	γ	C	Cache	τ	Notes
Waveform[1]	4000	1000	0.05	1	40M	0.001	Artificial data, 21 dims.
Banana[1]	4000	1300	0.5	316	40M	0.001	Artificial data, 2 dims.
Reuters[2]	7700	3299	1	1	40M	0.001	Topic "moneyfx" vs. rest.
USPS[3]	7329	2000	2	1000	40M	0.001	Class "0" vs. rest.
Adult[3]	32562	16282	0.005	100	40M	0.001	As in (Platt, 99).
Forest[3] (100k)	100000	50000	1	3	512M	0.001	As in (Collobert et al. 02)
Forest[3] (521k)	521012	50000	1	3	1250M	0.01	As in (Collobert et al. 02).

[1] `http://mlg.anu.edu.au/~raetsch/data/index.html`

[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578`

[3] `ftp://ftp.ics.uci.edu/pub/machine-learning-databases`

LASVM on Various Datasets

	LIBSVM				LASVM×1			
Dataset	**Error**	**SV**	**KCalc**	**Time**	**Error**	**SV**	**KCalc**	**Time**
Waveform	8.82%	1006	4.2M	3.2s	**8.68%**	**948**	**2.2M**	**2.7s**
Banana	9.96%	873	6.8M	9.9s	9.98%	869	6.7M	10.0s
Reuters	2.76%	1493	11.8M	**24s**	2.76%	1504	**9.2M**	31.4s
USPS	0.41%	236	1.97M	**13.5s**	0.43%	**201**	**1.08M**	15.9s
Adult	14.90%	11327	1760M	1079s	14.94%	11268	**626M**	**809s**
Forest (100k)	**8.03%**	43251	27569M	14598s	8.15%	**41750**	**18939M**	**10310s**
Forest (521k)	4.84%	124782	316750M	159443s	4.83%	122064	**188744M**	**137183s**

The Finishing step

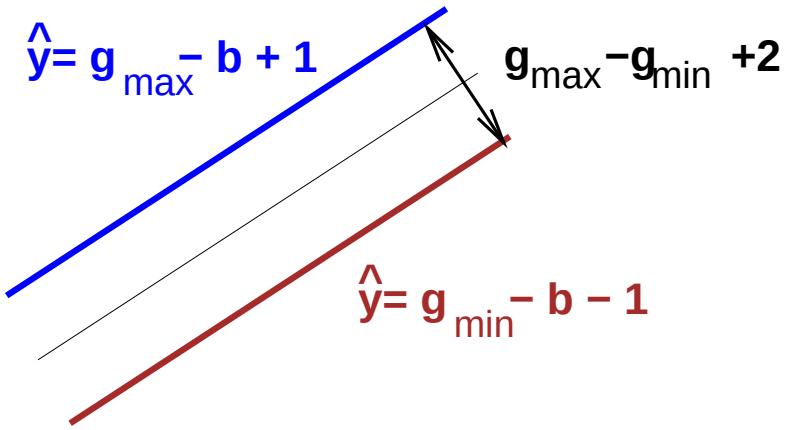
Impact of the finishing step.

Dataset	Relative Variation Error	SV	Time
Waveform	-0%	-0%	+4%
Banana	-79%	-74%	+185%
Reuters	0%	-0%	+3%
USPS	0%	-2%	+0%
USPS+N%	-67%	-33%	+7%
Adult	-13%	-19%	+80%
Forest (100k)	-1%	-24%	+248%
Forest (521k)	-2%	-24%	+84%

The collection of SVs

$$g_k \;=\; y_k - \sum_i \alpha_i K(x_i, x_k) \;=\; y_k - \hat{y}(x_k) + b$$

- PROCESS with a positive example k makes a SV if $\mathcal{S}$ contains a matching example i such that $g_k - g_i > 0$.
- REPROCESS reduces $g_{max} - g_{min}$



Variations on REPROCESS

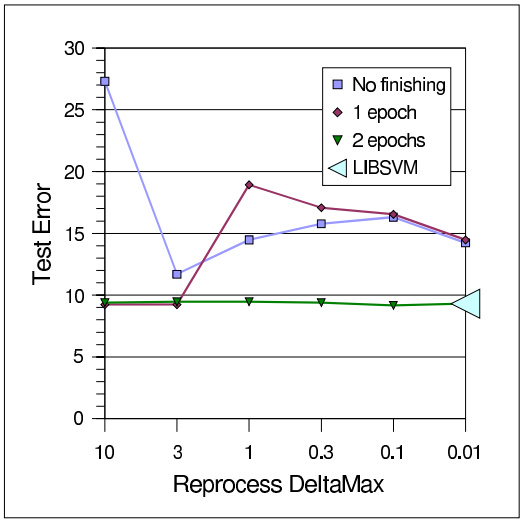
More REPROCESS $\Longrightarrow$

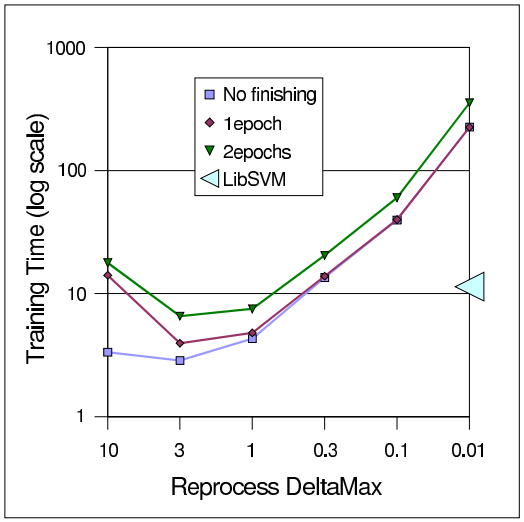
PROCESS collects less SVs.

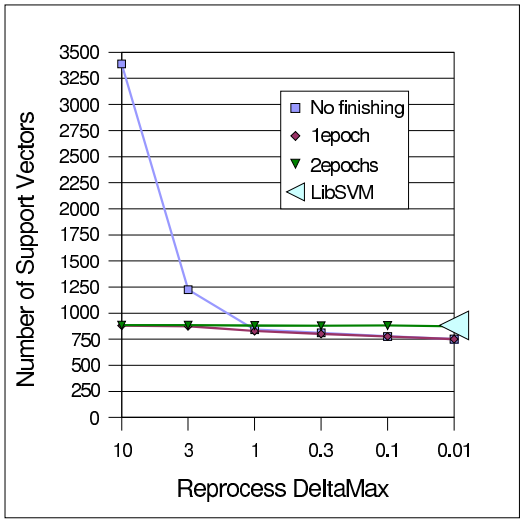
Experiment: Repeat REPROCESS until

$$g_{max} - g_{min} < \Delta$$

Variations on REPROCESS







Roadmap

	Memory	Time	SVs
SVM	NK	$N^{\alpha}K$	$K \sim 2\mathcal{B}N$
Online SVM	K^2	NK †	$K \sim 2\mathcal{B}N$
Active SVM	K^2	NK	$K \ll 2\mathcal{B}N$ †

† caveats. . .

Example selection

picking examples to PROCESS

Gradient Selection:

Strongest gradient.

$$k_G \quad = \quad \arg\min_{k \notin \mathcal{S}} \quad y_k \, \widehat{y}(x_k)$$

Active Selection:

Do not rely on labels.

$$k_A \quad = \quad \arg\min_{k \notin \mathcal{S}} \quad \max_{y=\pm 1} \quad y \widehat{y}(x_k) \quad = \quad \arg\min_{k \notin \mathcal{S}} \quad |\widehat{y}(x_k)|$$

Sampling

Given an example selection criterion:

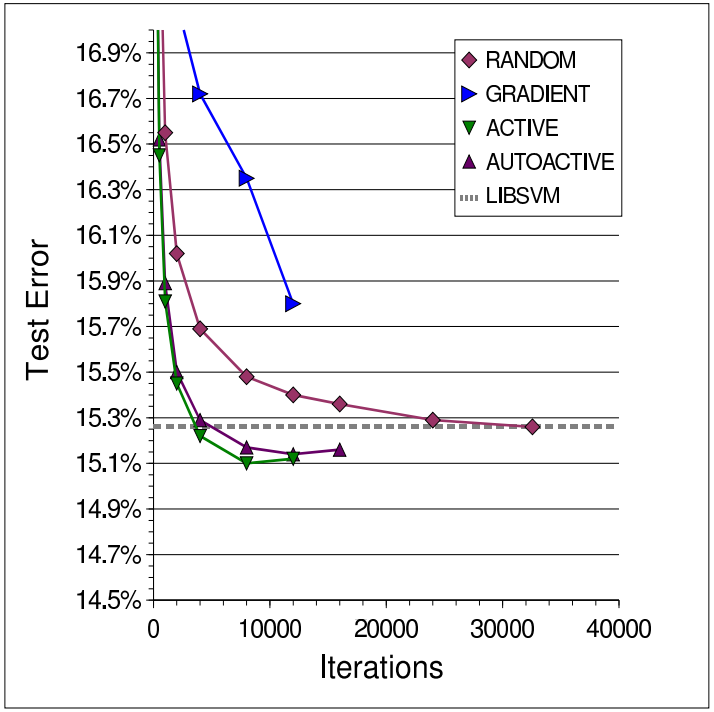
– Pick K random examples

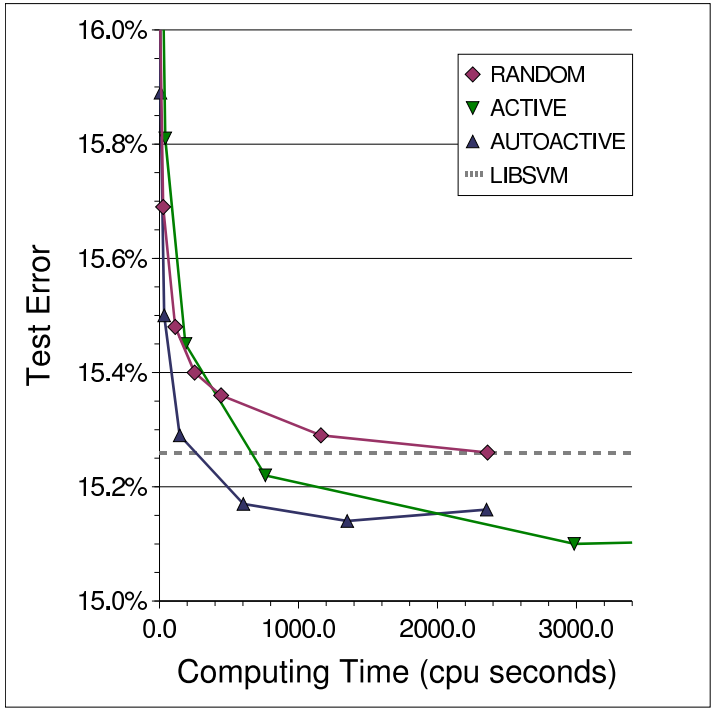
– Choose the one with best criterion.

- When $K = 59$, we have 95% chances to be among the 5% best regardless of the total number of examples.

- Heuristics to adapt K.

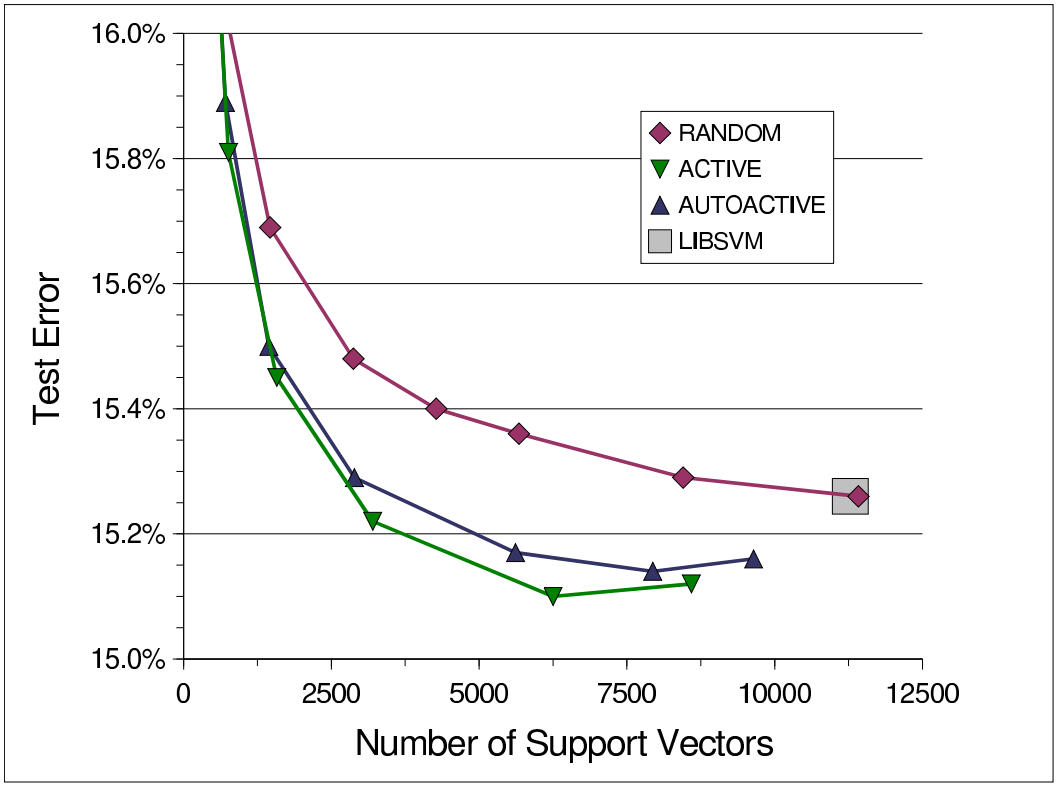
Example selection on Adult





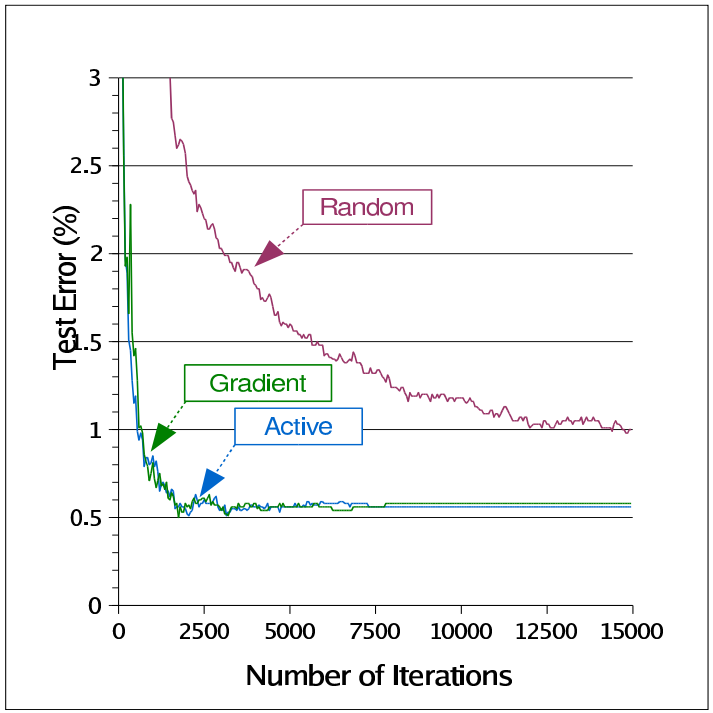
Average over 65 folds: accuracy$\approx$ 0.25%.

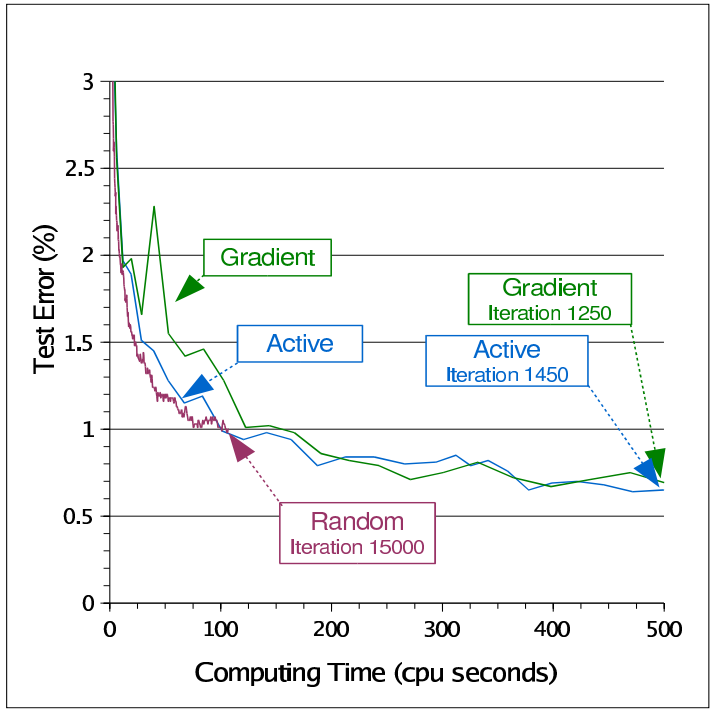
Example selection on Adult



Example selection on MNIST

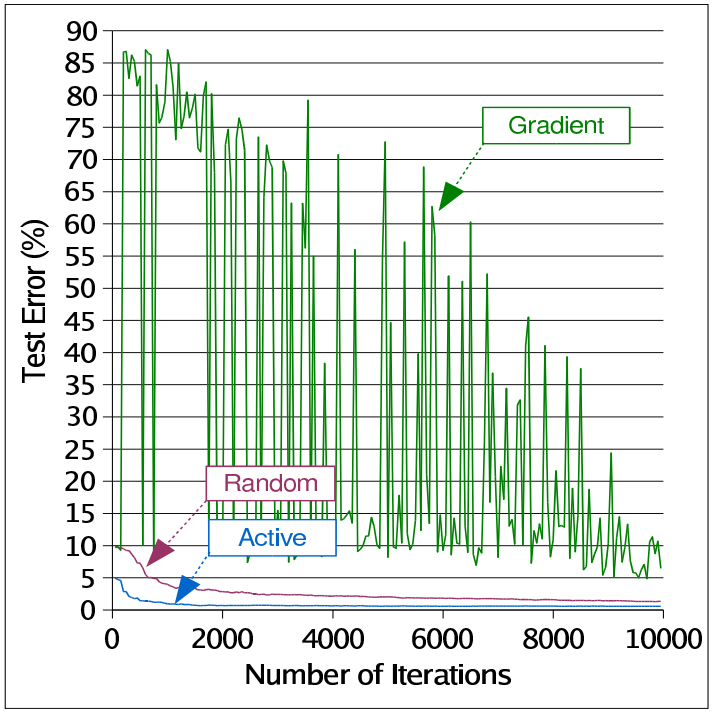
Class 8 versus rest

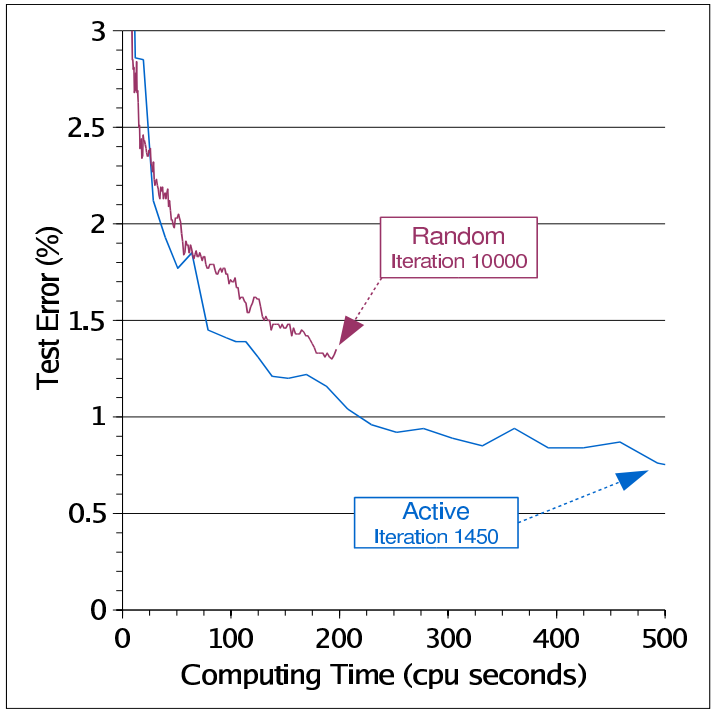




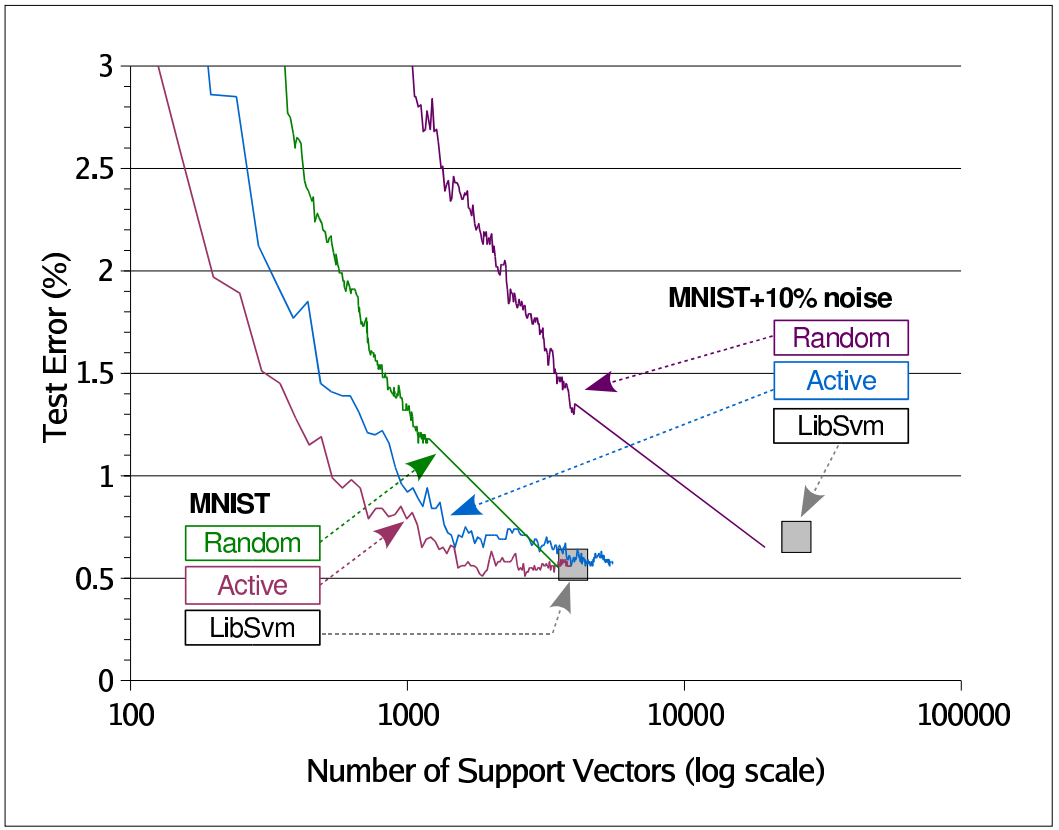
Example selection on MNIST

Class 8 versus rest, 10% label noise.





Example selection on MNIST



Discussion

Online

Accelerated learning

Disks like sequential accesses.

Active

Very fast learning on noisy data

Less support vectors

Parallel selection.

Large Scale MNIST

Examples

- 60,000 MNIST digits.
- 120 random deformations per digit.
- 7,200,000 training examples.

Results

- RBF kernel (no tricks).
- 10 one-vs-rest classifiers.
- each classifier trains in about 24h.
- test error rate (not final): 0.67%

Convergence Theory

"Witness directions for convex optimization"

- Feasible directions.
- Primal optimality criterion (Zoutendijk).
- Witness family of directions.
- Simplified optimality criterion.
- Finite witness family $\Rightarrow$ Convex is a polytope.
- Stochastic Sequential Direction Search.
  - strong convergence condition ($p_t > \pi > 0$)
- Approximate solutions of the convex problem.
- Approximate Sequential Direction Search
  - weak a.s. convergence condition ($\limsup p_t > 0$)
- Examples.

KKT theorems unnecessary!

Unsolved theoretical issues

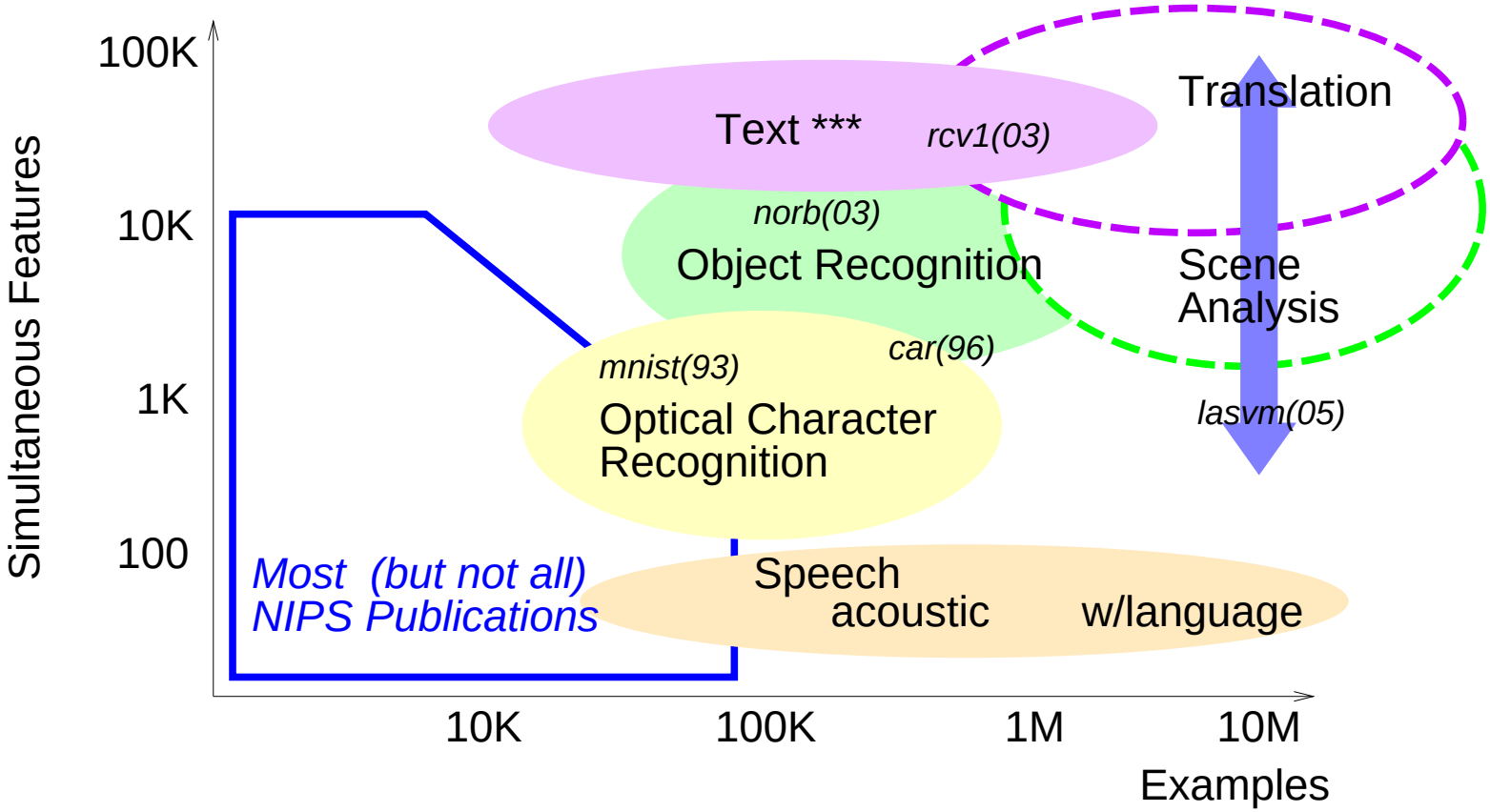
(i) Accuracy after one epoch vs. many epochs.

(ii) Number of SV using Active selection.

This is related to using a non-convex loss.

(iii) Less information $\longrightarrow$ higher accuracy.

Conclusion



Large scale SVMs are feasible!