Kernel Methods in Machine Learning

James Kwok

Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong

Joint work with Ivor Tsang, Pakming Cheung, Andras Kocsor, Jacek Zurada, Kimo Lai

November 2006, Nanjing

Outline

Popularity of Kernel Methods

Supervised learning

- Classification: Support vector machines (SVM)
- Regression: Support vector regression

Unsupervised learning

- Novelty detection: One-class SVM / Support vector domain description
- Clustering: Kernel clustering
- Principal component analysis: Kernel PCA

Other learning scenarios

- Semi-supervised learning, transductive learning, etc.

Applications

- Text classification, speaker adaptation, image fusion, texture classification ...

Basic Idea in Kernel Methods

Map the data from input space to feature space $\mathcal{F}$ using φ Apply a linear procedure in $\mathcal{F}$

- hyperplane classifier, linear regression, PCA, etc.

Only inner products in $\mathcal{F}$ are needed

- Kernel trick: $\varphi(\mathbf{x})'\varphi(\mathbf{y}) = k(\mathbf{x}, \mathbf{y})$

Support Vector Machines

Classification problem: $\{(\mathbf{x}_i, y_i)\}_{i=1}^N, x_i \in \mathbb{R}^m, y_i \in \{\pm 1\}$

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{w}\|^2 \qquad (\text{primal}) \\ \text{s.t.} & y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \geq 1 \end{array}$$

$$\begin{array}{ll} \max & \displaystyle\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \varphi(\mathbf{x}_i)' \varphi(\mathbf{x}_j) \\ \text{s.t.} & \displaystyle\sum_{i=1}^N \alpha_i y_i = 0, \quad \alpha_i \geq 0 \ \ (\text{dual}) \end{array}$$

Quadratic programming (QP) problem

Scale-up Problem

Problem 1

Need $O(m^2)$ memory just to write down $\mathbf{K}$ (m training examples)

- If $m = 20,000$ and it takes 4 bytes to represent a kernel entry, we would need 1.6Gbytes to store the kernel matrix

Problem 2

Involves inverting the kernel matrix $\mathbf{K}_{m \times m} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^m$

- Requires $O(m^3)$ time

Existing methods

- sampling, low-rank approximations, decomposition methods
- in practice, time complexities $O(m) - O(m^{2.3})$
- empirical observations and not theoretical guarantees

Observation

SVM implementations only approximate the optimal solution by an iterative strategy

1. Pick a subset of Lagrange multipliers
2. Optimize the reduced optimization problem
3. Repeat until all the Lagrange multipliers are "accurate enough" (loose KKT condition)

These near-optimal solutions are often good enough in practical applications

Approximation Algorithm

Approximation algorithms have been extensively used theoretical computer science

- E.g., for NP-complete problems such as vertex-cover problem, traveling-salesman problem, set-covering problem, ...

Denote

- C^*: cost of the optimal solution
- C: cost of the solution returned by approximation algorithm

Performance guarantee: Approximation ratio $\rho(n)$ for input size n

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$$

- large $\rho(n)$: solution is much worse than the optimal solution
- small $\rho(n)$: solution is more or less optimal

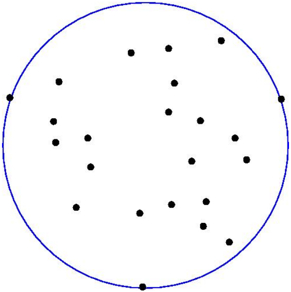
If the ratio does not depend on n, we may just write ρ and call the algorithm an ρ-approximation algorithm

The Minimum Enclosing Ball Problem

Problem in Computational Geometry

Given: $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, where each $\mathbf{x}_i \in \mathbb{R}^d$

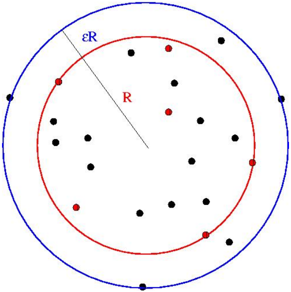
Minimum enclosing ball of $\mathcal{S}$ (MEB($\mathcal{S}$)): the smallest ball that contains all the points in $\mathcal{S}$



Finding exact MEBs is inefficient for large d

$(1+\epsilon)$-Approximation

Given an $\epsilon > 0$, a ball $B(\mathbf{c}, (1+\epsilon)R)$ is an $(1+\epsilon)$-approximation of MEB($\mathcal{S}$) if $R \leq r_{\text{MEB}(\mathcal{S})}$ and $\mathcal{S} \subset B(\mathbf{c}, (1+\epsilon)R)$



Approximate MEB Algorithm

Proposed by Bădoiu and Clarkson (2002)

A simple iterative scheme:

- At the tth iteration, the current estimate $B(\mathbf{c}_t, r_t)$ is expanded incrementally by including the furthest point outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)r_t)$
- Repeat until all the points in $\mathcal{S}$ are covered by $B(\mathbf{c}_t, (1+\epsilon)r_t)$

Surprising property

- Number of iterations (and hence the size of the final core-set) depends only on ϵ but not on d or m

MEB Problems and Kernel Methods

What is obvious

- MEB is equivalent to the hard-margin support vector data description (SVDD)
- The MEB problem can also be used to find the radius component of the radius-margin bound
  $\Rightarrow$ SVM parameter tuning

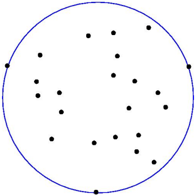
What is not so obvious

- Other kernel-related problems can also be viewed as MEB problems
- soft-margin one-class SVM, multi-class SVM, ranking SVM, SVR, Laplacian SVM, etc.

Hard-Margin SVDD

Denote:

- Kernel k; feature map φ
- MEB in the feature space: $B(\mathbf{c}, R)$



$$(\text{primal}) : \min_{R,\mathbf{c}} R^2 \quad : \quad \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 \leq R^2, \quad i = 1, \ldots, m$$

$$(\text{dual}) \quad \max_{\boldsymbol{\alpha}} \quad \boldsymbol{\alpha}'\text{diag}(\mathbf{K}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \quad : \quad \boldsymbol{\alpha} \geq \mathbf{0}, \quad \boldsymbol{\alpha}'\mathbf{1} = 1$$

- $\boldsymbol{\alpha} = [\alpha_i, \ldots, \alpha_m]'$: Lagrange multipliers
- $\mathbf{K}_{m \times m} = [k(\mathbf{x}_i, \mathbf{x}_j)]$: kernel matrix
- $\mathbf{0} = [0, \ldots, 0]'$, $\mathbf{1} = [1, \ldots, 1]'$

Kernel Methods as MEB Problems

$$\text{Assume} \quad k(\mathbf{x}, \mathbf{x}) = \kappa, \quad \text{a constant} \tag{1}$$

Holds for

1. isotropic kernel $k(\mathbf{x}, \mathbf{y}) = K(\|\mathbf{x} - \mathbf{y}\|)$ (e.g., Gaussian)
2. dot product kernel $k(\mathbf{x}, \mathbf{y}) = K(\mathbf{x}'\mathbf{y})$ (e.g., polynomial) with normalized inputs
3. any normalized kernel $k(\mathbf{x}, \mathbf{y}) = \frac{K(\mathbf{x},\mathbf{y})}{\sqrt{K(\mathbf{x},\mathbf{x})}\sqrt{K(\mathbf{y},\mathbf{y})}}$

Combine with $\boldsymbol{\alpha}'\mathbf{1} = 1$, we have $\boldsymbol{\alpha}'\text{diag}(\mathbf{K}) = \kappa$

$$\max_{\boldsymbol{\alpha}} -\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \quad : \quad \boldsymbol{\alpha} \geq \mathbf{0}, \quad \boldsymbol{\alpha}'\mathbf{1} = 1 \tag{2}$$

Conversely, whenever the kernel k satisfies (1),

Any QP of the form in (2) $\leftrightarrow$ a MEB problem

Two-Class SVM

$\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$ with $y_i \in \{-1, 1\}$

$$\text{(primal)} \quad \min_{\mathbf{w},b,\rho,\xi_i} \quad \|\mathbf{w}\|^2 + b^2 - 2\rho + C\sum_{i=1}^m \xi_i{}^2 \quad : \quad y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \geq \rho - \xi_i$$

$$\text{(dual)} \quad \max_{\boldsymbol{\alpha}} \quad -\boldsymbol{\alpha}'\left(\mathbf{K} \odot \mathbf{y}\mathbf{y}' + \mathbf{y}\mathbf{y}' + \frac{1}{C}\mathbf{I}\right)\boldsymbol{\alpha} \; : \; \boldsymbol{\alpha} \geq \mathbf{0}, \;\; \boldsymbol{\alpha}'\mathbf{1} = 1$$

$$\tilde{\mathbf{K}} = \left[y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}\right], \;\; \text{with} \;\; \tilde{k}(\mathbf{z}, \mathbf{z}) = \kappa + 1 + \frac{1}{C} \;\; \text{(constant)}$$

Core Vector Machine (CVM)

At the tth iteration, denote

- $\mathcal{S}_t$: core-set; $\mathbf{c}_t$: ball's center; R_t: ball's radius

Given an $\epsilon > 0$

1. Initialize $\mathcal{S}_0$, $\mathbf{c}_0$ and R_0
2. Terminate if there is no training point $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ falls outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$
3. Find (core vector) $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ is furthest away from $\mathbf{c}_t$. Set $S_{t+1} = \mathcal{S}_t \cup \{\mathbf{z}\}$
4. Find the new MEB($\mathcal{S}_{t+1}$) and set $\mathbf{c}_{t+1} = \mathbf{c}_{\text{MEB}(\mathcal{S}_{t+1})}$ and $R_{t+1} = r_{\text{MEB}(\mathcal{S}_{t+1})}$
5. Increment t by 1 and go back to Step 2

Convergence to (Approximate) Optimality

When $\epsilon = 0$

- CVM outputs the exact solution of the kernel problem

When $\epsilon > 0$

CVM is an $(1 + \epsilon)^2$-approximation algorithm

Time Complexity

CVM converges in at most $2/\epsilon$ iterations [Bădoiu and Clarkson, 2002]

No probabilistic speedup:

- Overall time for $\tau = O(1/\epsilon)$ iterations: $\boxed{O\left(\frac{m}{\epsilon^2} + \frac{1}{\epsilon^4}\right)}$
- linear in m for a fixed ϵ

With probabilistic speedup:

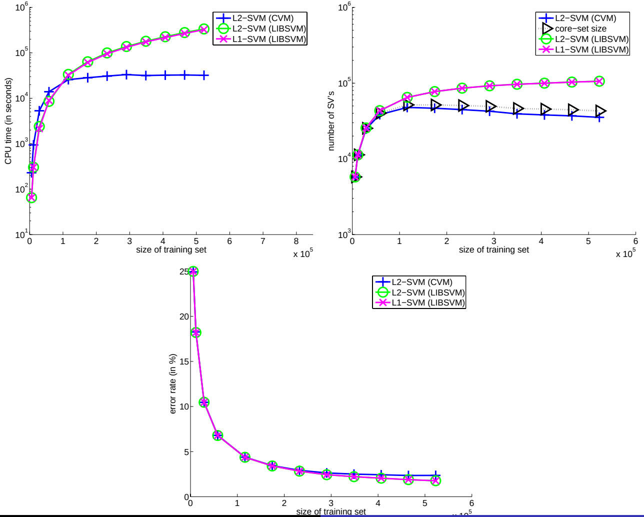
- Overall time: $\boxed{O\left(\frac{1}{\epsilon^4}\right)}$
- independent of m for a fixed ϵ

Space Complexity

Space complexity for the for the whole procedure: $O(1/\epsilon^2)$

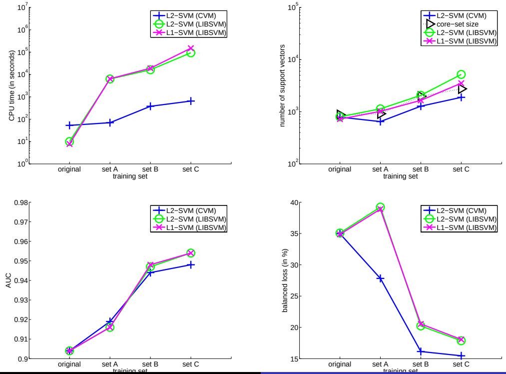
- independent of m for a fixed ϵ

Forest Cover Type Data (522,911 patterns)



Extended MIT Face Data

training set	# faces	# nonfaces	total
original	2,429	4,548	6,977
set A	2,429	481,914	484,343
set B	19,432 (blur+flip)	481,914	501,346
set C	408,072 (rotate)	481,914	889,986



KDDCUP-99 Intrusion Detection (4,898,431 patterns)

Used in KDD-99's Knowledge Discovery and Data Mining Tools Competition: Separate normal connections from attacks

method		# train patns input to SVM	# test errors	SVM training time (in sec)	other proc time (in sec)
random sampling	0.001%	47	25,713	0.000991	500.02
	0.01%	515	25,030	0.120689	502.59
	0.1%	4,917	25,531	6.944	504.54
	1%	49,204	25,700	604.54	509.19
	5%	245,364	25,587	15827.3	524.31
active learning		747	21,634	94192.213	
CB-SVM (KDD'03)		4,090	20,938	7.639	4745.483
CVM		4,898,431	19,513	1.4	

AUC	ℓ_{bal}	# core vectors	# support vectors
0.977	0.042	55	20

Limitations

1. $k(\mathbf{x}, \mathbf{x}) =$ constant for any pattern $\mathbf{x}$
2. The QP problem is of the form

$$\max -\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \quad \text{s.t.} \quad \boldsymbol{\alpha}'\mathbf{1} = 1, \quad \boldsymbol{\alpha} \geq \mathbf{0}$$

Condition 1 holds for kernels, including

- Isotropic kernel (e.g., Gaussian kernel)
- Dot product kernel (e.g., polynomial kernel) with normalized input
- Any normalized kernel

Condition 2 holds for kernel methods including the one-class and two-class SVMs

- there are still some popular kernel methods that violate these conditions and so cannot be used

Motivating Example

Example (L2-support vector regression (SVR))

Training set: $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$

Find $f(\mathbf{x}) = \mathbf{w}'\varphi(\mathbf{x}) + b$ in $\mathcal{F}$ that minimizes $\bar{\varepsilon}$-insensitive loss

Primal

$$\min \quad \|\mathbf{w}\|^2 + b^2 + \frac{C}{\mu m}\sum_{i=1}^m (\xi_i^2 + \xi_i^{*2}) + 2C\bar{\varepsilon}$$

$$\text{s.t.} \quad y_i - (\mathbf{w}'\varphi(\mathbf{x}_i) + b) \leq \bar{\varepsilon} + \xi_i, \quad (\mathbf{w}'\varphi(\mathbf{x}_i) + b) - y_i \leq \bar{\varepsilon} + \xi_i^*$$

Dual

$$\max \quad [\boldsymbol{\lambda}' \ \ \boldsymbol{\lambda}^{*\prime}] \begin{bmatrix} \frac{2}{C}\mathbf{y} \\ -\frac{2}{C}\mathbf{y} \end{bmatrix} - [\boldsymbol{\lambda}' \ \ \boldsymbol{\lambda}^{*\prime}]\tilde{\mathbf{K}} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^* \end{bmatrix}$$

$$\text{s.t.} \quad [\boldsymbol{\lambda}' \ \ \boldsymbol{\lambda}^{*\prime}]\mathbf{1} = 1, \quad \boldsymbol{\lambda}, \boldsymbol{\lambda}^* \geq \mathbf{0}$$

- $\tilde{\mathbf{K}} = [\tilde{k}(\mathbf{z}_i, \mathbf{z}_j)] = \begin{bmatrix} \mathbf{K} + \mathbf{1}\mathbf{1}' + \frac{\mu m}{C}\mathbf{I} & -(\mathbf{K} + \mathbf{1}\mathbf{1}') \\ -(\mathbf{K} + \mathbf{1}\mathbf{1}') & \mathbf{K} + \mathbf{1}\mathbf{1}' + \frac{\mu m}{C}\mathbf{I} \end{bmatrix}$

Center-Constrained MEB Problem

Modifications to the original MEB problem:

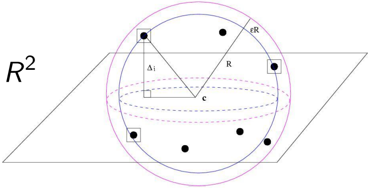
1. Augment an extra $\Delta_i \in \mathbb{R}$ to each $\varphi(\mathbf{x}_i) \rightarrow \begin{bmatrix} \varphi(\mathbf{x}_i) \\ \Delta_i \end{bmatrix}$
2. Constrain the last coordinate of the ball's center to zero $\begin{bmatrix} \mathbf{c} \\ 0 \end{bmatrix}$

Finding the center-constrained MEB

Primal:

$$\min R^2 \quad \text{s.t.} \quad \left\| \begin{bmatrix} \mathbf{c} \\ 0 \end{bmatrix} - \begin{bmatrix} \varphi(\mathbf{x}_i) \\ \Delta_i \end{bmatrix} \right\|^2 \leq R^2$$

where $\boldsymbol{\Delta} = [\Delta_1^2, \ldots, \Delta_m^2]' \geq \mathbf{0}$



Dual:

$$\max \;\; \boldsymbol{\alpha}'(\text{diag}(\mathbf{K}) + \boldsymbol{\Delta}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \quad \text{s.t.} \;\; \boldsymbol{\alpha}'\mathbf{1} = 1, \;\; \boldsymbol{\alpha} \geq \mathbf{0}$$

Goal: Transform the dual of SVR to this form

SVR as a Center-Constrained MEB Problem

SVR's dual:

$$\begin{aligned} \max \quad & \overbrace{[\boldsymbol{\lambda}' \ \ \boldsymbol{\lambda}^{*\prime}]}^{\tilde{\boldsymbol{\alpha}}'} \begin{bmatrix} \frac{2}{C}\mathbf{y} \\ -\frac{2}{C}\mathbf{y} \end{bmatrix} - [\boldsymbol{\lambda}' \ \ \boldsymbol{\lambda}^{*\prime}]\tilde{\mathbf{K}} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^* \end{bmatrix} \\ \text{s.t.} \quad & [\boldsymbol{\lambda}' \ \ \boldsymbol{\lambda}^{*\prime}]\mathbf{1} = 1, \ \ \boldsymbol{\lambda}, \boldsymbol{\lambda}^* \geq \mathbf{0} \end{aligned}$$

Define $\boldsymbol{\Delta} = -\text{diag}(\tilde{\mathbf{K}}) + \eta\mathbf{1} + \frac{2}{C}\begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \end{bmatrix}$ for η large enough such that $\boldsymbol{\Delta} \geq \mathbf{0}$

$$\max \ \tilde{\boldsymbol{\alpha}}'(\text{diag}(\tilde{\mathbf{K}}) + \boldsymbol{\Delta} - \eta\mathbf{1}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} \quad : \quad \tilde{\boldsymbol{\alpha}}'\mathbf{1}=1, \quad \tilde{\boldsymbol{\alpha}} \geq \mathbf{0}$$

Using the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$

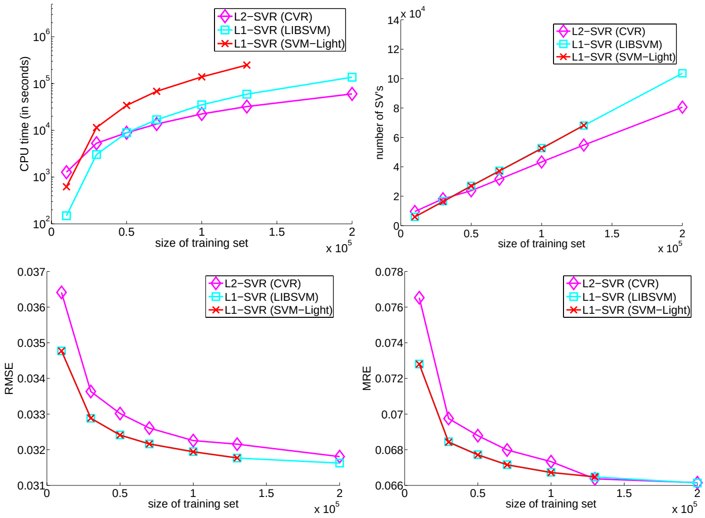
$$\max \ \tilde{\boldsymbol{\alpha}}'(\text{diag}(\tilde{\mathbf{K}}) + \boldsymbol{\Delta}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} \quad : \quad \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1, \quad \tilde{\boldsymbol{\alpha}} \geq \mathbf{0}$$

which is thus of the required form!

Advantages

1. Allows a more general QP formulation
2. Can be used with any linear/nonlinear kernels
   - no longer require "$k(\mathbf{x}, \mathbf{x}) = \text{constant}$" on the kernel

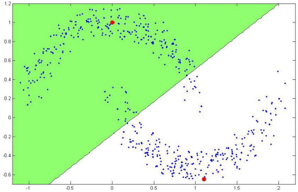
Friedman (200,000 Patterns)



Semi-Supervised Learning

Labeled patterns are rare, expensive and time consuming to collect

- supervised learning can have poor performance when only very few labeled patterns are available



Unlabeled data are abundant and readily available without any cost

- e.g., unlabeled webpages on the internet
- often has a manifold structure

Laplacian SVM

Incorporate a manifold regularizer [Belkin et al 2005]:

$$\min \quad \frac{1}{\ell}\sum_{i=1}^{\ell}\xi_i + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_k} + \frac{\lambda_G}{2}\|\nabla_G f\|^2$$
$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

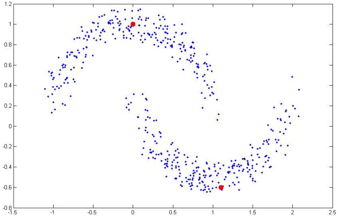
Sparse Laplacian SVM

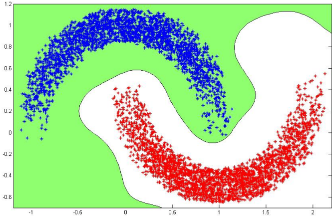
$$\min \quad \underbrace{\|\mathbf{w}\|^2 + b^2}_{\text{margin}} + \underbrace{\frac{C}{\ell\mu}\sum_{i=1}^{\ell}\xi_i^2}_{\text{error}} + 2C\epsilon + \underbrace{\frac{C\theta}{u\mu}\sum_{e\in\mathcal{E}}(\zeta_e^2 + \zeta_e^{*2})}_{\text{manifold regularizer}}$$

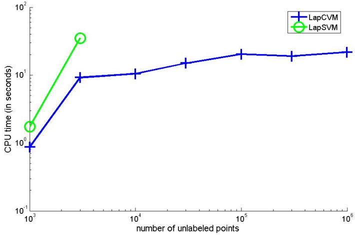
$$\text{s.t.} \quad y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \geq 1 - \epsilon - \xi_i,$$
$$-\mathbf{w}'\psi_e \leq \epsilon + \zeta_e, \mathbf{w}'\psi_e \leq \epsilon + \zeta_e^*, \quad e \in \mathcal{E}.$$

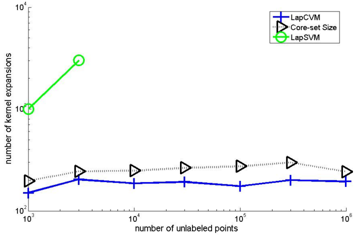
Dual: center-constrained MEB problem

Two Moons ($\ell = 2; u = 1,000,000$)

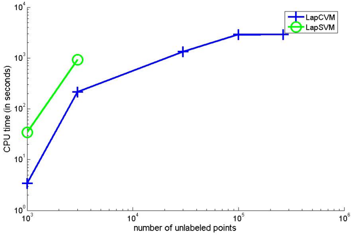


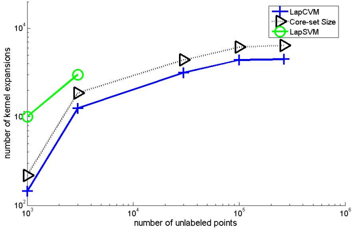


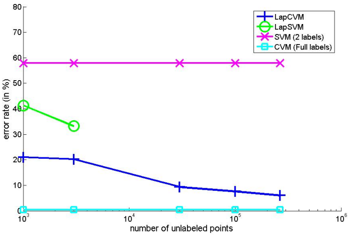




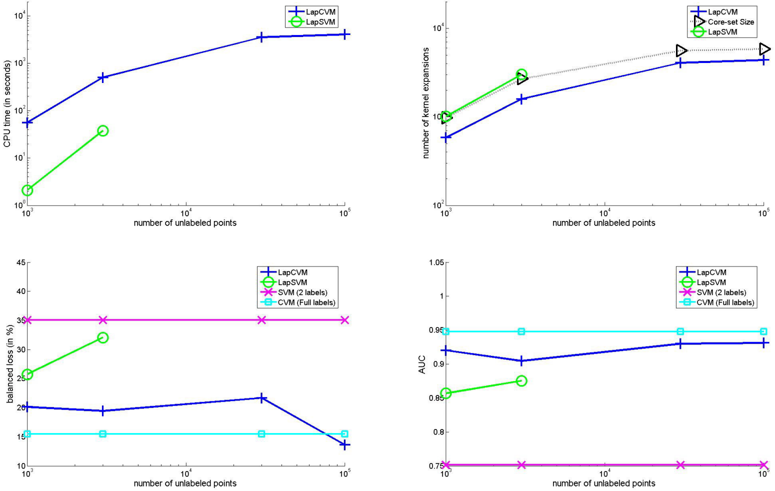
Extended USPS: 0-vs-1 ($\ell = 2; u = 266,077$)







Extended MIT Face ($\ell = 10; u = 100,000$)



Multi-Instance Learning: Motivating Example

Content-based image retrieval: Classify/retrieve images based on content

- each image is a bag and each local image patch an instance
- an image is labeled positive when at least one of its segments is positive

Weak label information of the training data

- only the bags (but not the individual instances) have known labels

Kernel-Based MI Learning Methods

Design MI kernels that operate on bags

- the underlying quadratic programming (QP) problem only involves variables corresponding to the bags, but not instances

(More direct approach) Associate the variables with instances, but not with bags

- bag label information still used implicitly
- bag B_i: instances $\{\mathbf{x}_{ij}\}_{j=1}^{n_i}$

$$f(B_i) = \max_{j=1,\ldots,n_i} f(\mathbf{x}_{ij})$$

Problems

Mixed integer problem

- MI-SVM uses a simple optimization heuristic
- convergence properties unclear

Only the sign is important in classification

- $\text{sign}(f(B_i)) = \text{sign}(\max_{j=1,\dots,n_i} f(\mathbf{x}_{ij}))$
- $f(B_i) = \max_{j=1,\dots,n_i} f(\mathbf{x}_{ij})$ may be too restrictive

Cannot utilize both the bag and instance information simultaneously

- MI kernels: variables correspond only to the bags, but not instances
- MI-SVM: variables correspond only to the instances, but not bags

Proposed Approach

Introduce a loss function between $f(B_i)$ and the associated $f(\mathbf{x}_{ij})$'s

- allows both the bags and instances to directly participate in the optimization process
- the learned function is smooth over both bags and instances

Optimization technique

- MI-SVM uses an optimization heuristic
- we use constrained concave-convex procedure

Constrained Concave-Convex Procedure

An optimization tool for nonlinear optimization problems whose objective function can be expressed as a difference of convex functions

Constrained Concave-Convex Procedure (CCCP)

$$\begin{array}{ll} \min_{\mathbf{x}} & f_0(\mathbf{x}) - g_0(\mathbf{x}) \\ \text{s.t.} & f_i(\mathbf{x}) - g_i(\mathbf{x}) \leq c_i, \quad i = 1, \ldots, m, \end{array}$$

- f_i, g_i $(i = 0, \ldots, m)$ are real-valued, convex and differentiable functions on $\mathbb{R}^n$; $c_i \in \mathbb{R}$

Procedure:

1. start with an initial $\mathbf{x}^{(0)}$
2. replace $g_i(\mathbf{x})$ with its first-order Taylor expansion at $\mathbf{x}^{(t)}$
3. set $\mathbf{x}^{(t+1)}$ to the solution of the relaxed optimization problem:

$$\begin{array}{ll} \min\limits_{\mathbf{x}} & f_0(\mathbf{x}) - \left[g_0(\mathbf{x}^{(t)}) + \nabla g_0(\mathbf{x}^{(t)})'(\mathbf{x} - \mathbf{x}^{(t)}) \right] \\ \text{s.t.} & f_i(\mathbf{x}) - \left[g_i(\mathbf{x}^{(t)}) + \nabla g_i(\mathbf{x}^{(t)})'(\mathbf{x} - \mathbf{x}^{(t)}) \right] \leq c_i \end{array}$$

Converges to a local minimum solution

Regularization Framework

A set of training bags: $\{(B_1, y_1), \ldots, (B_m, y_m)\}$

- $B_i = \{\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{in_i}\}$: ith bag containing instances $\mathbf{x}_{ij}$'s
- $y_i \in \{\pm 1\}$

Define a loss function that depends on both the training bags and training instances:

$$V\left(\{B_i, y_i, f(B_i)\}_{i=1}^m, \{f(\mathbf{x}_{ij})\}_{j=1}^{n_i}{}_{i=1}^m\right)$$

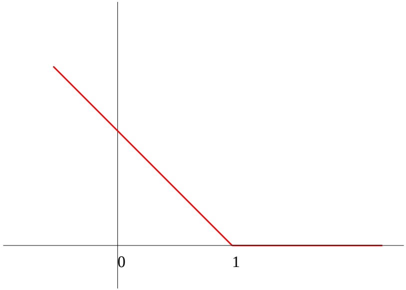
Split the loss function V into two parts

1. between each bag label and its bag prediction
$$V\left(\{B_i, y_i, f(B_i)\}_{i=1}^m, \{f(\mathbf{x}_{ij})\}_{j=1}^{n_i}{}_{i=1}^m\right)$$
2. between the predictions of each bag and its constituent instances
$$V\left(\{B_i, y_i, f(B_i)\}_{i=1}^m, \{f(\mathbf{x}_{ij})\}_{j=1}^{n_i}{}_{i=1}^m\right)$$

Loss Function V: 1st Part

Between each bag label y_i and its corresponding prediction $f(B_i)$

- hinge loss $(1 - y_i f(B_i))_+$ where $(z)_+ = \max(0, z)$



Loss Function V: 2nd Part

Between the predictions of each bag $f(B_i)$ and its constituent instances $\{f(\mathbf{x}_{ij}) \mid j = 1, \ldots, n_i\}$

$$\boxed{\ell(f(B_i), \max_j f(\mathbf{x}_{ij}))}$$

- $\ell(v_1, v_2) = \begin{cases} 0 & \text{if } v_1 = v_2, \\ \infty & \text{otherwise.} \end{cases}$
- L1 loss: $\ell(v_1, v_2) = |v_1 - v_2|$
- L2 loss: $\ell(v_1, v_2) = (v_1 - v_2)^2$

Combining

$$V = \frac{1}{m}\sum_{i=1}^{m}(1 - y_i f(B_i))_+ + \frac{\lambda}{m}\sum_{i=1}^{m}\ell(f(B_i), \max_j f(\mathbf{x}_{ij}))$$

- λ: trades off the two components

Special cases:

1. Only the first part: $\frac{1}{m}\sum_{i=1}^{m}(1 - y_i f(B_i))_+$
   - the same as that with the MI kernel
2. $\ell(v_1, v_2) = \begin{cases} 0 & \text{if } v_1 = v_2, \\ \infty & \text{otherwise.} \end{cases}$
   - same as the MI-SVM

Optimization Problem

Introduce

- $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_m]'$: slack variables for the errors on bags
- γ, λ: tradeoff parameters

$$\begin{aligned} \min_{f \in \mathcal{H}, \boldsymbol{\xi}} \quad & \frac{1}{2}\|f\|_{\mathcal{H}}^2 + \frac{\gamma}{m}\boldsymbol{\xi}'\mathbf{1} + \frac{\gamma\lambda}{m}\sum_{i=1}^{m} \ell\left(f(B_i), \max_{j=1,\ldots,n_i} f(\mathbf{x}_{ij})\right) \\ \text{s.t.} \quad & y_i f(B_i) \geq 1 - \xi_i, \\ & \boldsymbol{\xi} \geq \mathbf{0} \end{aligned}$$

Representer Theorem

$$f(\mathbf{x}) = \sum_{i=1}^{m}\left(\alpha_{i0} k(\mathbf{x}, B_i) + \sum_{j=1}^{n_i} \alpha_{ij} k(\mathbf{x}, \mathbf{x}_{ij})\right), \quad \alpha_{i0}, \alpha_{ij} \in \mathbb{R}$$

- $\boldsymbol{\alpha}$: vector for all the α_{i0}'s and α_{ij}'s

Using the L1 Loss for $\ell(\cdot,\cdot)$

- $\mathbf{K}$: kernel matrix; $\mathbf{k}_i$: ith column of $\mathbf{K}$

$$
\begin{aligned}
\min_{\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\delta},b} \quad & \frac{1}{2}\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} + \frac{\gamma}{m}\boldsymbol{\xi}'\mathbf{1} + \frac{\gamma\lambda}{m}\boldsymbol{\delta}'\mathbf{1} \\
\text{s.t.} \quad & y_i(\mathbf{k}'_{\mathcal{I}(B_i)}\boldsymbol{\alpha} + b) \geq 1 - \xi_i, \\
& \boldsymbol{\xi} \geq \mathbf{0}, \\
& \mathbf{k}'_{\mathcal{I}(\mathbf{x}_{ij})}\boldsymbol{\alpha} - \delta_i \leq \mathbf{k}'_{\mathcal{I}(B_i)}\boldsymbol{\alpha}, \\
& \mathbf{k}'_{\mathcal{I}(B_i)}\boldsymbol{\alpha} - \max_{j=1,\ldots,n_i} (\mathbf{k}'_{\mathcal{I}(\mathbf{x}_{ij})}\boldsymbol{\alpha}) \leq \delta_i
\end{aligned}
$$

Objective: quadratic; First three constraints: linear
Last constraint: nonlinear, but is a difference of two convex functions

Optimization using CCCP

Iterative procedure:

1. obtain $\boldsymbol{\alpha}$ from this QP
2. use this as $\boldsymbol{\alpha}^{(t+1)}$ and iterate

- $\boldsymbol{\alpha}^{(t)}$: estimate of $\boldsymbol{\alpha}$ at the tth iteration
- $\beta_{ij}^{(t)}$: estimate of β_{ij}

$$\begin{aligned}
\min_{\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\delta},b} \quad & \frac{1}{2}\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} + \frac{\gamma}{m}\boldsymbol{\xi}'\mathbf{1} + \frac{\gamma\lambda}{m}\boldsymbol{\delta}'\mathbf{1} \\
\text{s.t.} \quad & y_i(\mathbf{k}'_{\mathcal{I}(B_i)}\boldsymbol{\alpha} + b) \geq 1 - \xi_i, \\
& \boldsymbol{\xi} \geq \mathbf{0}, \\
& \mathbf{k}'_{\mathcal{I}(\mathbf{x}_{ij})}\boldsymbol{\alpha} - \delta_i \leq \mathbf{k}'_{\mathcal{I}(B_i)}\boldsymbol{\alpha}, \\
& \mathbf{k}'_{\mathcal{I}(B_i)}\boldsymbol{\alpha} - \sum_{j=1}^{n_i} \beta_{ij}^{(t)}\mathbf{k}'_{\mathcal{I}(\mathbf{x}_{ij})}\boldsymbol{\alpha} \leq \delta_i
\end{aligned}$$

Using the Loss Function in MI-SVM

With a particular choice of the subgradient

- identical to the optimization heuristic in MI-SVM
- MI-SVM: no convergence proof
- CCCP: guaranteed convergence

Classification: Image Categorization on Corel Images

Data set

- Used in Chen and Wang (JMLR 2004)
- 10 classes (beach, flowers, horses, etc.), with each class containing 100 images
- Each image: bag; Image segments: instance

Procedure

- Same as in (Chen and Wang)
- Randomly divided into a training and test set, each containing 50 images of each category
- Repeated 5 times, and report the average accuracy
- Model parameters selected by a validation set

Results

	accuracy (%)
DD-SVM (Chen and Wang 2004)	81.5 ± 3.0
Hist-SVM (Chapelle *et al.* 1999)	66.7 ± 2.2
MI-SVM (Andrews *et al.* 2003)	74.7 ± 0.6
SVM (MI kernel) (Gärtner *et al.* 2002)	84.1 ± 0.90
Our method	84.4 ± 1.38

- Results on DD-SVM, Hist-SVM and MI-SVM are from (Chen and Wang 2004)
- MI kernel used: normalized set kernel

$$\kappa(B_1, B_2) = \frac{k_{set}(B_1, B_2)}{\sqrt{k_{set}(B_1, B_1)}\sqrt{k_{set}(B_2, B_2)}}$$

$$k_{set}(B_1, B_2) = \sum_{\mathbf{x} \in B_1, \mathbf{z} \in B_2} k(\mathbf{x}, \mathbf{z}), \qquad k\text{: Gaussian kernel}$$

- Our method: use the L1 loss
  - significant at the 0.01 level of significance

Regression: Synthetic Musk Molecules

Predict the real-valued binding energies of musk molecules

Synthetic data sets generated by Dooly *et al.* (JMLR 2002)

- based on using an affinity model between the musk molecules and receptors
- LJ-16.30.2, LJ-80.166.1 and LJ-160.166.1
- LJ-16.30.2: # relevant features: 16; total # features: 30; # scale factors: 2

Make it more challenging

- created three more data sets (LJ-16-50-2, LJ-80-206-1 and LJ-160-566-1) by adding irrelevant features
- e.g., LJ-16-50-2 is generated by adding 20 more irrelevant features to LJ-16-30-2 while keeping its real-valued outputs intact

Results

data set	DD		citation-*k*NN		SVM (MI kernel)		our method	
	%err	MSE	%err	MSE	%err	MSE	%err	MSE
LJ-16.30.2	6.7	0.0240	16.7	0.0260	10	**0.0184**	10	0.0185
LJ-80.166.1	(not available)		8.6	0.0109	8.7	0.0135	4.3	**0.0097**
LJ-160.166.1	23.9	0.0852	4.3	**0.0014**	0	0.0054	0	0.0053
LJ-16-50-2	-	-	53.3	0.0916	40	0.0723	33.3	**0.0673**
LJ-80-206-1	-	-	30.4	0.0463	23.9	0.0325	22.8	**0.0321**
LJ-160-566-1	-	-	34.8	0.0566	37.0	0.0535	33.7	**0.0507**

- Results of DD, citation-*k*NN on the first three data sets are from (Dooly *et al.* 2002)
- DD: does not perform well
- Easier data sets: ours has comparable/better performance
- More challenging data sets
  - nearest neighbor-based and DD algorithms degrade with more irrelevant features
  - our SVM-based approach is consistently the best

Conclusion

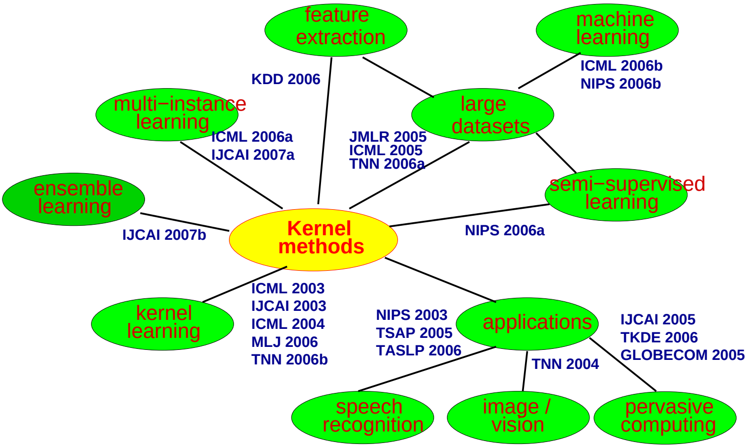
Kernel methods can now be used on massive data sets:

- novelty detection (unsupervised learning)
- classification/regression (supervised learning)
- manifold regularization (semi-supervised learning)
- maximum margin discriminant analysis (feature extraction)

Kernel methods can also be used for multi-instance learning in a disciplined manner

- allows a loss function between the outputs of a bag and its associated instances
- both bags and instances can now directly participate in the optimization process
- by using CCCP, no need to use optimization heuristics
- how to design MI kernels? $\longrightarrow$ marginalized kernel

Recent Research



http://www.cse.ust.hk/~jamesk

Google: james kwok