Machine Learning: Kernel, Sparsity, Online, and Future Perspectives

Haiqin Yang, Irwin King, and Michael R. Lyu

Department of Computer Science & Engineering
The Chinese University of Hong Kong

June 10, 2012

A Motivated Example

How to Find?

Season	Age	Team	Lg	G	Min	Pts	FG%	3P%	FT%	...
2006-07	18	HARVARD	College	28	506	133	.415	.281	.818	...
2007-08	19	HARVARD	College	30	940	377	.448	.279	.621	...
2008-09	20	HARVARD	College	28	975	497	.502	.400	.744	...
2009-10	21	HARVARD	College	29	933	476	.519	.341	.755	...
2010-11	22	GSW	NBA	29	284	76	.389	.200	.760	...
2011-12	23	NYK	NBA	35	940	512	.437	.311	.793	...
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	... [1]

[1] Data from http://www.basketball-reference.com

What if machine learning/data mining techniques are applied?

Possible Results

Season	Age	Team	Lg	G	Min	Pts	FG%	3P%	FT%	...
2006-07	18	HARVARD	College	28	506	133	.415	.281	.818	...
2007-08	19	HARVARD	College	30	940	377	.448	.279	.621	...
2008-09	20	HARVARD	College	28	975	497	.502	.400	.744	...
2009-10	21	HARVARD	College	29	933	476	.519	.341	.755	...
2010-11	22	GSW	NBA	29	284	76	.389	.200	.760	...
2011-12	23	NYK	NBA	35	940	512	.437	.311	.793	...
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...

Russtroll Westbrook @sideeyespecial Following

Welp. Looks like I may have been wrong about Lin ascending to Teflon Don ranks

Reggie Miller @ReggieMillerTNT Follow

OK, I GIVE IN!!!!!!!!!! Its Legit!!!!! WOW!!!!!!!!

Spike Lee @SpikeLee Following

Floyd Mayweather I Hope You Watched Jeremy Hit The Gamewinning 3 Pointer With .005 Seconds Left.Our Guy Can BALL.PLAIN AND SIMPLE.RECOGNIZE.

Mat MacDonald @Mat_MacDonald Following

Jose Calderon's having his way with Lin. Like I said, don't forget about him. #RTZ

Word-of-Mouth Effect!

Outline

Pre-requisites Knowledge

- Calculus
- Linear algebra
- Probability theory
- Optimization
- Geometry

Outline

Outline

"I applied my heart to what I observed and learned a lesson from what I saw."

– Proverbs 24:32 (NIV)

"A few observations and much reasoning lead to error; many observations and a little reasoning lead to truth."

– Alexis Carrel

Supervised Learning

Learning from labeled observations

Horse

Donkey

- Given labeled data: $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}, \quad \mathbf{x}_i \in \mathbb{R}^d, \quad y_i \in \{\pm 1\}/\mathbb{R}$
- Classification: $f(\mathbf{x}) \to \{-1, +1\}$
- Regression: $f(\mathbf{x}) \to \mathbb{R}$

Semi-supervised/Transductive Learning

Learning from labeled and unlabeled observations

Horse


Donkey


Unlabeled data




- Given data: $\mathcal{L}$, and $\mathcal{U}_{\mathcal{L}} = \{(\mathbf{x}_j)\}_{j=1}^{U}, \quad \mathbf{x}_j \in \mathbb{R}^d$
- Learn $f(\mathbf{x}) \to \{-1, +1\}$
- Semi-supervised learning: In-class exam
- Transductive learning: Take-home exam

Unsupervised Learning

Learning patterns from unlabeled observations.




Learning from Universum

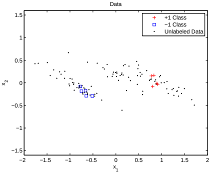
Learning from labeled and universum observations

Horse

Donkey

Universum (Mule)

Illustration



- Given data: $\mathcal{L}$, and $\mathcal{U}_0 = \{(\mathbf{x}_k)\}_{k=1}^{U}, \quad \mathbf{x}_k \in \mathbb{R}^d$
- Learn $f(\mathbf{x}) \to \{-1, +1\}$
- Criterion: Maximizing contraction on Universum

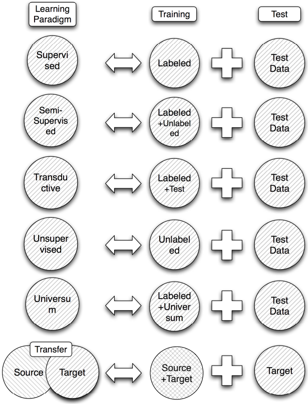
Transfer Learning

Transfer knowledge across domains, tasks, and distributions that are similar but not identical

Task 1: Learn to distinguish horse and donkey

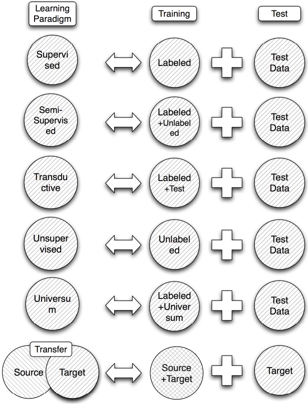
Transfer knowledge learned from Task 1 to distinguish sheep and goat

Summary of Learning Paradigms



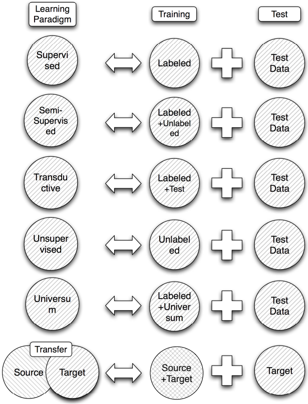
- Supervised learning
  Support vector machines (SVM), Lasso, etc.
- Semi-supervised/Transductive learning
  S^3VM, TSVM
- Learning from universum
  $\mathcal{U}$-SVM
- Transfer learning
  Multi-task learning
- ...

Summary of Learning Paradigms



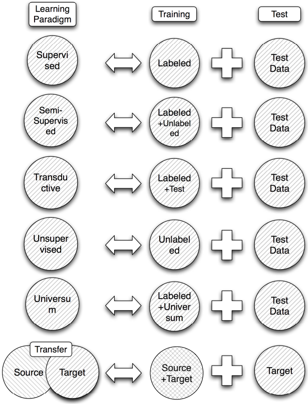
- Supervised learning
  Support vector machines (SVM), Lasso, etc.
- Semi-supervised/Transductive learning
  S^3VM, TSVM
- Learning from universum
  $\mathcal{U}$-SVM
- Transfer learning
  Multi-task learning
- ...

Summary of Learning Paradigms



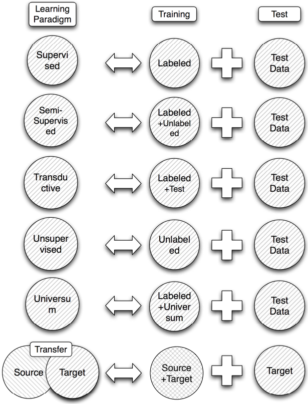
- Supervised learning
  Support vector machines (SVM), Lasso, etc.
- Semi-supervised/Transductive learning
  S^3VM, TSVM
- Learning from universum
  $\mathcal{U}$-SVM
- Transfer learning
  Multi-task learning
- ...

Summary of Learning Paradigms



- Supervised learning
  Support vector machines (SVM), Lasso, etc.
- Semi-supervised/Transductive learning
  S^3VM, TSVM
- Learning from universum
  $\mathcal{U}$-SVM
- Transfer learning
  Multi-task learning
- . . .

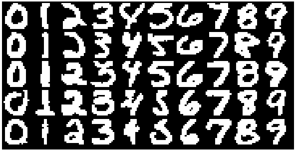
Summary of Learning Paradigms

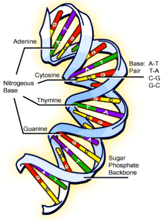


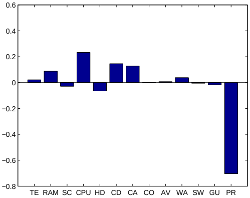
- Supervised learning
  Support vector machines (SVM), Lasso, etc.
- Semi-supervised/Transductive learning
  S^3VM, TSVM
- Learning from universum
  $\mathcal{U}$-SVM
- Transfer learning
  Multi-task learning
- ...

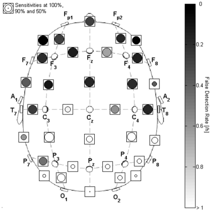
Applications

- Pattern recognition
- Computer vision
- Natural language processing
- Information retrieval
- Medical diagnosis
- Market decisions
- Bioinformatics
- ...









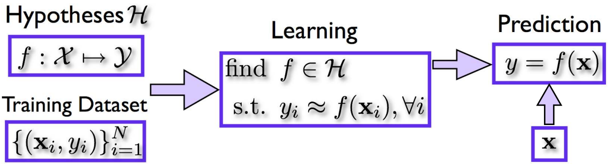
Outline

Supervised Learning Procedure

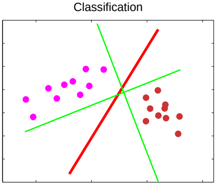
Data: N i.i.d. paired data sampled from $\mathcal{P}$ over $\mathcal{X} \times \mathcal{Y}$ as

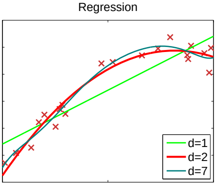
$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad \mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d, \quad y_i \in \mathcal{Y} \subseteq \mathbb{R}$$

Procedure:



Tasks:





Regularization

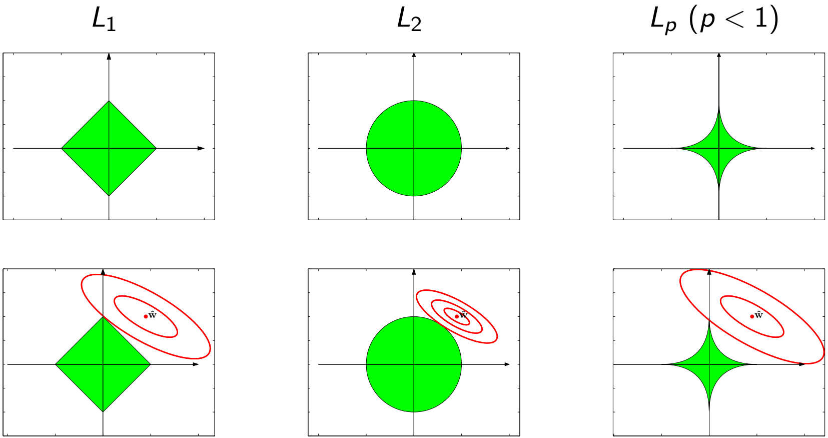
- Formulation

$$\boxed{f^* = \arg\min_{f \in \mathcal{H}} \left(R[f] + C\mathcal{R}^{\ell}_{\mathcal{D}}[f] \right)}$$

$R[f]$: Regularization, complexity of f
$\mathcal{R}^{\ell}_{\mathcal{D}}[f]$: Empirical risk, measured by square, hinge, etc.
$C \geq 0$: Trade-off parameter

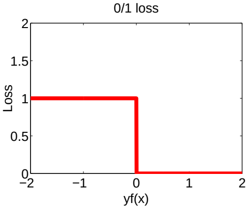
- Advantages
  - Controlling the functional complexity to avoid overfitting
  - Providing an intuitive and principled tool for learning from high-dimensional data
    - Lasso: Perform regression while selecting features
    - SVM: Regularization corresponds to maximum margin

Typical Regularizers

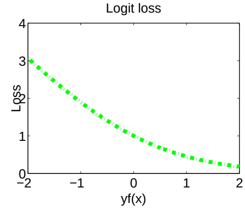


Typical Loss Functions

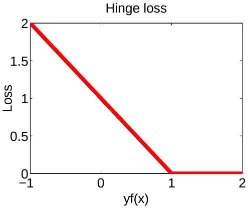
0/1-loss



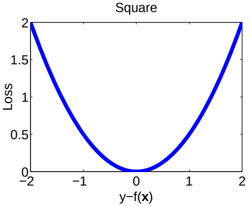
Logit loss



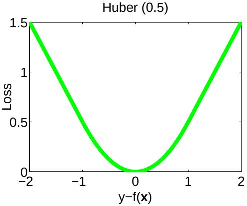
Hinge loss



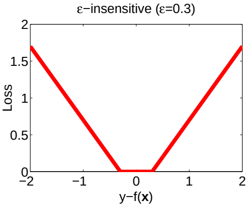
Square loss



Huber loss

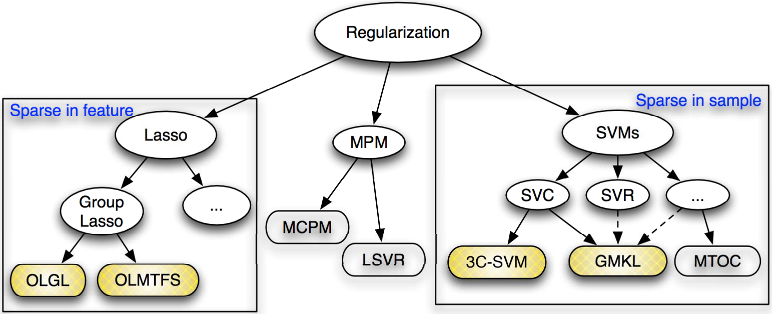


ϵ-insensitive loss



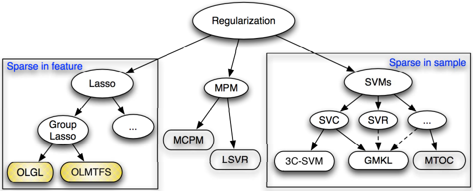
Outline

Overview



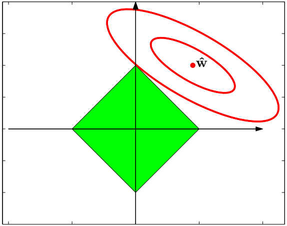
- Sparse learning models under regularization
  - Sparse in feature level
  - Sparse in sample level
- Online learning
- Semi-supervised learning
- Multiple kernel learning (MKL)

Sparse in Feature Level

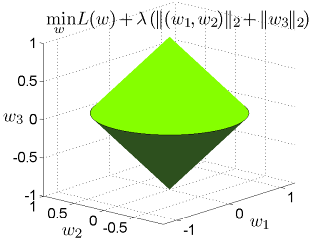


- Models

Lasso

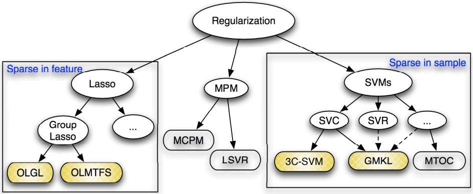


Group Lasso

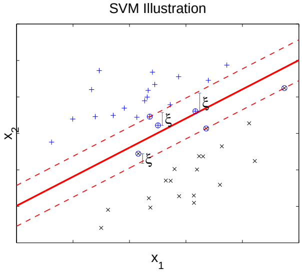


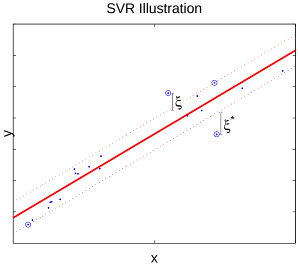
$f(\mathbf{x}) = \mathbf{w}^{\star T}\mathbf{x} + b^{\star}$, most elements of $\mathbf{w}^{\star}$ vanish!

Sparse in Sample Level



- Models



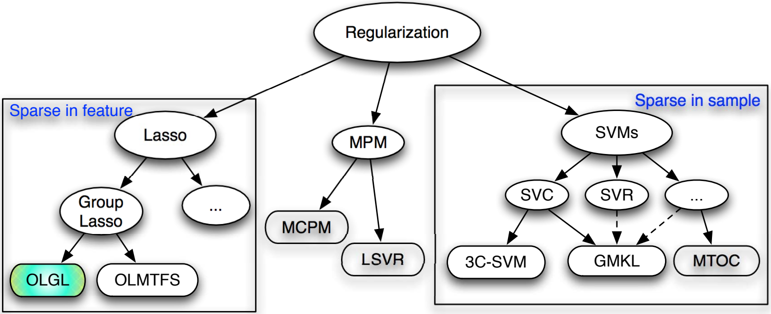


$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^\star K(\mathbf{x}_i, \mathbf{x}) + b^\star,$$ most elements of $\boldsymbol{\alpha}^\star$ vanish!

Outline

Outline

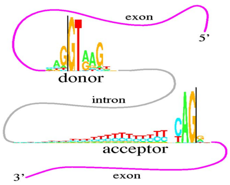
Online Learning for Group Lasso



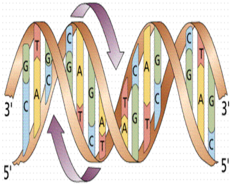
- H. Yang, Z. Xu, I. King, and M. R. Lyu. Online learning for group lasso. In *ICML*, pages 1191–1198, 2010.
- Toolbox: `http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=OLGL`

A Motivated Example

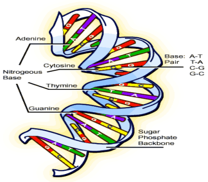
Data with group structure appear sequentially



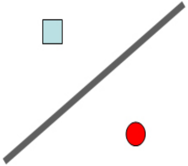
$\Longrightarrow$



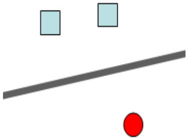
$\Longrightarrow$



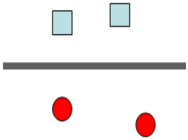
How to update the decision function adaptively?



$\Longrightarrow$

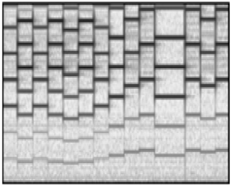


$\Longrightarrow$

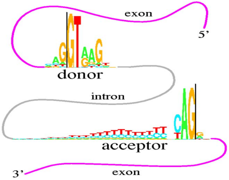


Motivations

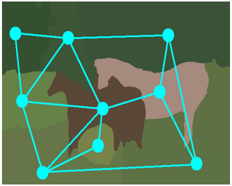
- Applications with group structure



McAuley et al., 2005



Meier et al., 2008



Harchaoui & Bach, 2007

- Group features
  - Continuous features represented by k-th order expansions
    $x_1 \Rightarrow \mathbf{x}_1 = [x_1, x_1^2, \ldots, x_1^k]$
  - Categorical features represented a group of dummy variables
    $x_2 \Rightarrow \mathbf{x}_2 = [x_{21}, x_{22}, \ldots, x_{2m}]$

Online Learning for Group Lasso

- Problems
  - Some features are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - Group lasso and its extensions (Yuan & Lin, 2006; Meier et al., 2008; Roth & Fischer, 2008; Jacob et al., 2009; etc.)
  - Online learning algorithms (Shalev-Shwartz & Singer, 2006; Zinkevich, 2003; Bottou & LeCun, 2003; Langford et al., 2009; Duchi & Singer, 2009; Xiao, 2009)

    Batch learned algorithms cannot solve the above problems!
- Our contributions
  - A novel online learning framework for the group lasso
  - Easy implementation: three lines of main codes
  - Efficient in both time complexity and memory cost, $\mathcal{O}(d)$
  - Sparsity in both the group level and the individual feature level
  - Easy extension to group lasso with overlap and graphical lasso

Models

Lasso: A shrinkage and selection method for linear regression

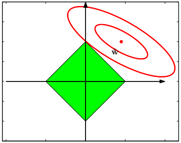
$$\min_{\mathbf{w}} \quad \|\mathbf{X}\mathbf{w} - \mathbf{Y}\|^2 + \lambda \|\mathbf{w}\|_1$$

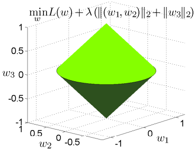
Group Lasso: Find important explanatory factors in a grouped manner

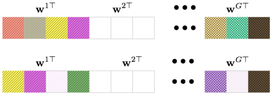
$$\min_{\mathbf{w}} \quad \|\mathbf{X}\mathbf{w} - \mathbf{Y}\|^2 + \lambda \sum_{g=1}^{G} \sqrt{d_g} \|\mathbf{w}^g\|_2$$

Sparse Group Lasso: Yield sparse solutions in the selected group

$$\min_{\mathbf{w}} \quad \|\mathbf{X}\mathbf{w} - \mathbf{Y}\|^2 + \lambda \sum_{g=1}^{G} (\sqrt{d_g} \|\mathbf{w}^g\|_2 + r_g \|\mathbf{w}^g\|_1)$$







Formulation Summary

- **Model framework**

$$\boxed{\min_{\mathbf{w}} \quad \sum_{i=1}^{N} \ell(\mathbf{w},\ \mathbf{z}_i) + \Omega_\lambda(\mathbf{w})}$$

$\ell(\cdot,\cdot)$: Loss function, e.g., square loss, logit loss, etc.
$\Omega_\lambda(\cdot)$: Regularization

- **Favorable properties**
  - Obtain sparse solution
  - Perform feature selection and classification/regression simultaneously
  - Attain good classification/regression performance

Online Learning Algorithm Framework for Group Lasso

Initialization: $\mathbf{w}_1 = \mathbf{w}_0$, $\bar{\mathbf{u}}_0 = \mathbf{0}$

for $t = 1, 2, 3, \ldots$

1. Compute the subgradient on $\mathbf{w}_t$, $\mathbf{u}_t \in \partial l_t$
2. Calculate the average subgradient $\bar{\mathbf{u}}_t$:
   $\bar{\mathbf{u}}_t = \frac{t-1}{t}\bar{\mathbf{u}}_{t-1} + \frac{1}{t}\mathbf{u}_t$
3. Update the next iteration $\mathbf{w}_{t+1}$:
   $$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \Upsilon(\mathbf{w}) \triangleq \left\{\bar{\mathbf{u}}_t^\top \mathbf{w} + \Omega_\lambda(\mathbf{w}) + \frac{\gamma}{\sqrt{t}} h(\mathbf{w})\right\}$$

end for

Remarks

- Motivated by the dual averaging method for Lasso (Xiao, 2009)
- $h(\mathbf{w})$: Make the new search point in the vincinity
- FOBOS (Duchi & Singer, 2009): $\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \{\frac{1}{2}\|\mathbf{w} - (\mathbf{w}_t - \eta_t \mathbf{u}_t)\|^2 + \eta_t \Omega(\mathbf{w})\}$
- Overlapped groups or graphical lasso

Updating Rules for Online Group Lasso

- **Group Lasso**: $\Omega_\lambda(\mathbf{w}) = \lambda \sum_{g=1}^{G} \sqrt{d_g}\|\mathbf{w}^g\|_2$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$

$$\boxed{\mathbf{w}_{t+1}^{g} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\bar{\mathbf{u}}_t^g\|_2}\right]_+ \cdot \bar{\mathbf{u}}_t^g}$$

- **Sparse Group Lasso**: $\Omega_{\lambda,\mathbf{r}}(\mathbf{w}) = \lambda \sum_{g=1}^{G} \left(\sqrt{d_g}\|\mathbf{w}^g\|_2 + r_g\|\mathbf{w}^g\|_1\right)$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$

$$\boxed{\mathbf{w}_{t+1}^{g} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\mathbf{c}_t^g\|_2}\right]_+ \cdot \mathbf{c}_t^g}, \quad c_t^{g,j} = \left[|\bar{u}_t^{g,j}| - \lambda r_g\right]_+ \cdot \mathrm{sign}\left(\bar{u}_t^{g,j}\right)$$

- **Enhanced Sparse Group Lasso**:
$\Omega_{\lambda,\mathbf{r}}(\mathbf{w}) = \lambda \sum_{g=1}^{G} \left(\sqrt{d_g}\|\mathbf{w}^g\|_2 + r_g\|\mathbf{w}^g\|_1\right)$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + \rho\|\mathbf{w}\|_1$

$$\boxed{\mathbf{w}_{t+1}^{g} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\tilde{\mathbf{c}}_t^g\|_2}\right]_+ \cdot \tilde{\mathbf{c}}_t^g,}, \quad \tilde{c}_t^{g,j} = \left[|\bar{u}_t^{g,j}| - \lambda r_g - \frac{\gamma\rho}{\sqrt{t}}\right]_+ \cdot \mathrm{sign}\left(\bar{u}_t^{g,j}\right)$$

Efficiency: $\mathcal{O}(d)$ in memory cost and time complexity

Updating Rules for Online Group Lasso

- **Group Lasso**: $\Omega_\lambda(\mathbf{w}) = \lambda \sum_{g=1}^{G} \sqrt{d_g}\|\mathbf{w}^g\|_2$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$

$$\boxed{\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\bar{\mathbf{u}}_t^g\|_2}\right]_+ \cdot \bar{\mathbf{u}}_t^g}$$

- **Sparse Group Lasso**: $\Omega_{\lambda,\mathbf{r}}(\mathbf{w}) = \lambda \sum_{g=1}^{G} \left(\sqrt{d_g}\|\mathbf{w}^g\|_2 + r_g\|\mathbf{w}^g\|_1\right)$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$

$$\boxed{\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\mathbf{c}_t^g\|_2}\right]_+ \cdot \mathbf{c}_t^g}, \quad c_t^{g,j} = \left[|\bar{u}_t^{g,j}| - \lambda r_g\right]_+ \cdot \mathrm{sign}\left(\bar{u}_t^{g,j}\right)$$

- **Enhanced Sparse Group Lasso**: $\Omega_{\lambda,\mathbf{r}}(\mathbf{w}) = \lambda \sum_{g=1}^{G} \left(\sqrt{d_g}\|\mathbf{w}^g\|_2 + r_g\|\mathbf{w}^g\|_1\right)$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + \rho\|\mathbf{w}\|_1$

$$\boxed{\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\tilde{\mathbf{c}}_t^g\|_2}\right]_+ \cdot \tilde{\mathbf{c}}_t^g,}, \quad \tilde{c}_t^{g,j} = \left[|\bar{u}_t^{g,j}| - \lambda r_g - \frac{\gamma\rho}{\sqrt{t}}\right]_+ \cdot \mathrm{sign}\left(\bar{u}_t^{g,j}\right)$$

Efficiency: $\mathcal{O}(d)$ in memory cost and time complexity

Updating Rules for Online Group Lasso

- **Group Lasso**: $\Omega_\lambda(\mathbf{w}) = \lambda \sum_{g=1}^{G} \sqrt{d_g}\|\mathbf{w}^g\|_2$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$

$$\boxed{\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\bar{\mathbf{u}}_t^g\|_2}\right]_+ \cdot \bar{\mathbf{u}}_t^g}$$

- **Sparse Group Lasso**: $\Omega_{\lambda,\mathbf{r}}(\mathbf{w}) = \lambda \sum_{g=1}^{G} \left(\sqrt{d_g}\|\mathbf{w}^g\|_2 + r_g\|\mathbf{w}^g\|_1\right)$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$

$$\boxed{\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\mathbf{c}_t^g\|_2}\right]_+ \cdot \mathbf{c}_t^g}, \quad c_t^{g,j} = \left[|\bar{u}_t^{g,j}| - \lambda r_g\right]_+ \cdot \mathrm{sign}\left(\bar{u}_t^{g,j}\right)$$

- **Enhanced Sparse Group Lasso**:
$\Omega_{\lambda,\mathbf{r}}(\mathbf{w}) = \lambda \sum_{g=1}^{G} \left(\sqrt{d_g}\|\mathbf{w}^g\|_2 + r_g\|\mathbf{w}^g\|_1\right)$, $h(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + \rho\|\mathbf{w}\|_1$

$$\boxed{\mathbf{w}_{t+1}^g = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda\sqrt{d_g}}{\|\tilde{\mathbf{c}}_t^g\|_2}\right]_+ \cdot \tilde{\mathbf{c}}_t^g,}, \quad \tilde{c}_t^{g,j} = \left[|\bar{u}_t^{g,j}| - \lambda r_g - \frac{\gamma\rho}{\sqrt{t}}\right]_+ \cdot \mathrm{sign}\left(\bar{u}_t^{g,j}\right)$$

Efficiency: $\mathcal{O}(d)$ in memory cost and time complexity

Average Regret for Group Lasso

- Definition

$$
\begin{aligned}
\bar{R}_T(\mathbf{w}) &:= \frac{1}{T}\sum_{t=1}^{T}\left(\Omega_\lambda(\mathbf{w}_t) + l_t(\mathbf{w}_t)\right) - S_T(\mathbf{w}) \\
S_T(\mathbf{w}) &:= \min_{\mathbf{w}} \frac{1}{T}\sum_{t=1}^{T}\left(\Omega_\lambda(\mathbf{w}) + l_t(\mathbf{w})\right)
\end{aligned}
$$

- Theoretical bounds

$\bar{R}_T \sim \mathcal{O}(1/\sqrt{T})$

$\bar{R}_T \sim \mathcal{O}(\log(T)/T)$ if $h(\cdot)$ is strongly convex

Average Regret for Group Lasso

- Definition

$$\bar{R}_T(\mathbf{w}) := \frac{1}{T}\sum_{t=1}^{T}(\Omega_\lambda(\mathbf{w}_t) + l_t(\mathbf{w}_t)) - S_T(\mathbf{w})$$
$$S_T(\mathbf{w}) := \min_{\mathbf{w}} \frac{1}{T}\sum_{t=1}^{T}(\Omega_\lambda(\mathbf{w}) + l_t(\mathbf{w}))$$

- Theoretical bounds

$\bar{R}_T \sim \mathcal{O}(1/\sqrt{T})$

$\bar{R}_T \sim \mathcal{O}(\log(T)/T)$ if $h(\cdot)$ is strongly convex

Summary

Summary

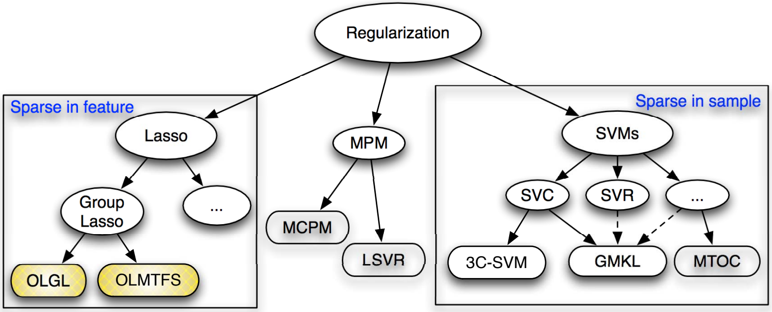
- A novel online learning algorithm framework for group lasso
- Apply this framework for variant group lasso models
- Provide closed-form solutions to update the models
- Provide the convergence rate of the average regret

Future work

- Evaluate on more datasets and compare with more other online frameworks
- Study lazy update schemes to handle high-dimensional data
- Derive a faster convergence rate for the online learning algorithm

Outline

Online Learning for Multi-Task Feature Selection

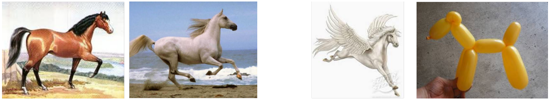


- H. Yang, I. King, and M. R. Lyu. Online learning for multi-task feature selection. In *CIKM2010*, pages 1693–1696, 2010.
- Toolbox: `http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=OLMTFS`

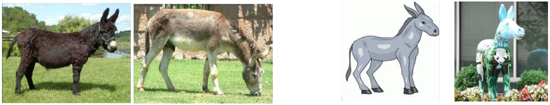
An Example of Multi-Task Learning

Given several similar, but not identical tasks

Task 1: Learn to recognize real horses



Task 2: Learn to recognize real donkeys



Task 3: Learn to recognize real mules

How to learn these tasks simultaneously to achieve better performance?

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples (< 100's), large variables (> 1000's)

    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: > 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests;
      2) Automatic classifying related web page categories
- Observation III: Redundant/irrelevant features exist

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples ($<$ 100's), large variables ($>$1000's)

    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: $>$ 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests;
      2) Automatic classifying related web page categories

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples ($<$ 100's), large variables ($>$1000's)
  - Text categorization from documents in multiple related categories
    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: $>$ 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests;
      2) Automatic classifying related web page categories

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples ($<$ 100's), large variables ($>$1000's)
  - **Text categorization** from documents in multiple related categories
    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: $>$ 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests; 2) Automatic classifying related web page categories

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples ($<$ 100's), large variables ($>$1000's)
  - **Text categorization** from documents in multiple related categories
    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: $>$ 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests;
      2) Automatic classifying related web page categories
- **Observation III**: Redundant/irrelevant features exist

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples ($<$ 100's), large variables ($>$1000's)
  - **Text categorization** from documents in multiple related categories
    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: $>$ 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests;
      2) Automatic classifying related web page categories
- **Observation III**: Redundant/irrelevant features exist

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - ♦ Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - ♦ Tasks: Distinguish healthy from unhealthy for different diseases
    - ♦ Problems: few samples ($<$ 100's), large variables ($>$1000's)
  - **Text categorization** from documents in multiple related categories
    - ♦ Features: A vector of vocabulary on word frequency counts
    - ♦ Vocabulary: $>$ 10000's words
    - ♦ Tasks: 1) Detecting spam-emails from persons with same interests;
      2) Automatic classifying related web page categories
- **Observation III**: Redundant/irrelevant features exist

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples ($<$ 100's), large variables ($>$1000's)
  - **Text categorization** from documents in multiple related categories
    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: $>$ 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests;
      2) Automatic classifying related web page categories
- **Observation III**: Redundant/irrelevant features exist

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples ($<$ 100's), large variables ($>$1000's)
  - **Text categorization** from documents in multiple related categories
    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: $>$ 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests;
      2) Automatic classifying related web page categories
- **Observation III**: Redundant/irrelevant features exist

Learning multiple tasks simultaneously **CAN** improve the model performance!

Why Multi-Task Feature Selection?

- **Observation I**: Training data are limited for each task
- **Observation II**: Related tasks contain helpful information
  - **Gene selection** from microarray data in related diseases
    - Variables: Gene expression coefficients corresponding to the amount of mRNA in a patient's sample (e.g., tissue biopsy)
    - Tasks: Distinguish healthy from unhealthy for different diseases
    - Problems: few samples ($<$ 100's), large variables ($>$1000's)
  - **Text categorization** from documents in multiple related categories
    - Features: A vector of vocabulary on word frequency counts
    - Vocabulary: $>$ 10000's words
    - Tasks: 1) Detecting spam-emails from persons with same interests;
2) Automatic classifying related web page categories
- **Observation III**: Redundant/irrelevant features exist

Learning multiple tasks simultaneously **CAN** improve the model performance!

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{1,0}$-regularization based on MIC (Dhillon et al. 2009)

  Batch trained algorithms CANNOT solve the above problems!
- Our contributions
  - A novel online learning framework for multi-task feature selection
  - Easy implementation: three lines of main codes
  - Efficient in both time complexity and memory cost, $O(d \times Q)$
  - Find important features and important tasks that dominate the targets
  - Easily extend to nonlinear models

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{0,0}$-regularization based on MIC (Dhillon et al. 2009)

  Batch trained algorithms CANNOT solve the above problems!
- Our contributions
  - A novel online learning framework for multi-task feature selection
  - Easy implementation: three lines of main codes
  - Efficient in both time complexity and memory cost, $O(d \times Q)$
  - Find important features and important tasks that dominate the features
  - Easily extend to nonlinear models

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{0,0}$-regularization based on MIC (Dhillon et al. 2009)

Batch trained algorithms **CANNOT** solve the above problems!

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{0,0}$-regularization based on MIC (Dhillon et al. 2009)

  Batch trained algorithms **CANNOT** solve the above problems!
- Our contributions

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{0,0}$-regularization based on MIC (Dhillon et al. 2009)

Batch trained algorithms **CANNOT** solve the above problems!

- Our contributions
  - A novel online learning framework for multi-task feature selection
  - Easy implementation: three lines of main codes
  - Efficient in both time complexity and memory cost: $O(d \times Q)$
  - Find important features and important tasks that dominate the feature
  - Easily extend to nonlinear models

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{0,0}$-regularization based on MIC (Dhillon et al. 2009)

Batch trained algorithms **CANNOT** solve the above problems!

- Our contributions
  - A novel online learning framework for multi-task feature selection
  - Easy implementation: three lines of main codes
  - Efficient in both time complexity and memory cost: $O(d \times Q)$
  - Find important features and important tasks that dominate the featur
  - Easily extend to nonlinear models

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{0,0}$-regularization based on MIC (Dhillon et al. 2009)

Batch trained algorithms **CANNOT** solve the above problems!

- Our contributions
  - A novel online learning framework for multi-task feature selection
  - Easy implementation: three lines of main codes
  - Efficient in both time complexity and memory cost, $\mathcal{O}(d \times Q)$
  - Find important features and important tasks that dominate the features
  - Easily extend to nonlinear models

Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{0,0}$-regularization based on MIC (Dhillon et al. 2009)

  Batch trained algorithms **CANNOT** solve the above problems!
- Our contributions
  - A novel online learning framework for multi-task feature selection
  - Easy implementation: three lines of main codes
  - Efficient in both time complexity and memory cost, $\mathcal{O}(d \times Q)$
  - Find important features and important tasks that dominate the features
  - Easily extend to nonlinear models

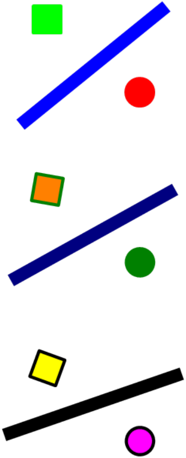
Problems and Contributions

- Problems
  - Features among tasks are redundant or irrelevant
  - Data come in sequence
  - Massive data
- Related work
  - A generalized L_1-norm single-task regularization (Argyriou et al. 2008)
  - Mixed norms of L_1, L_2, and L_∞ norms (Obozinski et al. 2009)
  - Nesterov's method on MTFS (Liu et al. 2009)
  - $L_{0,0}$-regularization based on MIC (Dhillon et al. 2009)

  Batch trained algorithms **CANNOT** solve the above problems!
- Our contributions
  - A novel online learning framework for multi-task feature selection
  - Easy implementation: three lines of main codes
  - Efficient in both time complexity and memory cost, $\mathcal{O}(d \times Q)$
  - Find important features and important tasks that dominate the features
  - Easily extend to nonlinear models

Idea Illustration

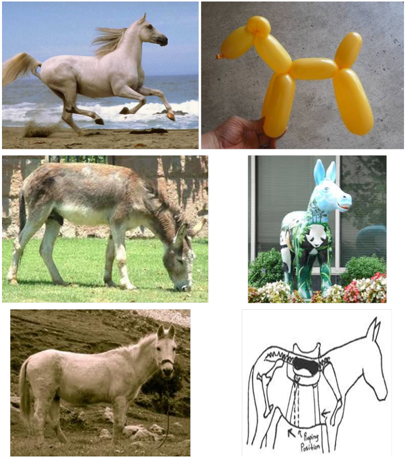
Multi-task data appear sequentially

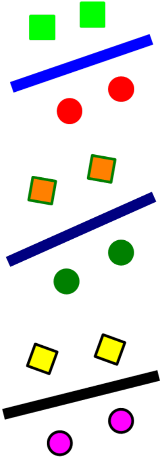


How to update the decision functions adaptively?

Idea Illustration

Multi-task data appear sequentially

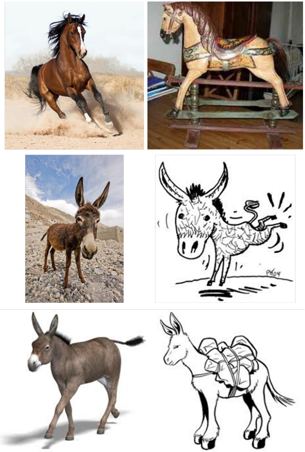


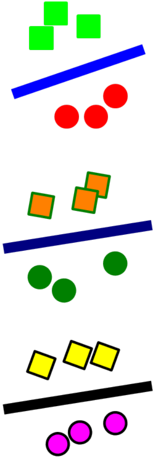


How to update the decision functions adaptively?

Idea Illustration

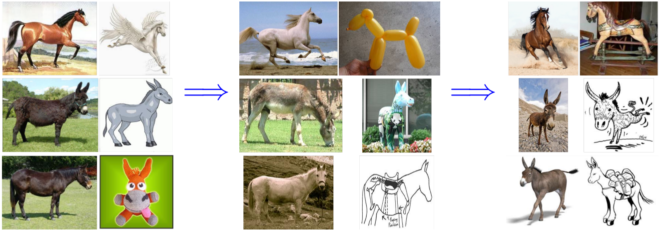
Multi-task data appear sequentially

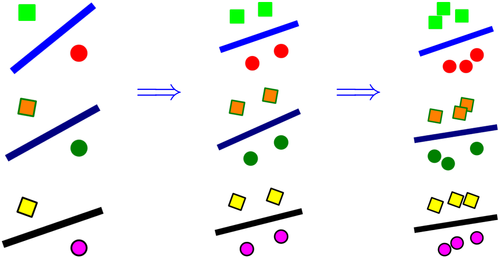




How to update the decision functions adaptively?

Idea Illustration





Multi-Task Feature Selection

- **Data**

  i.i.d. observations of $\mathcal{D} = \bigcup_{q=1}^{Q} \mathcal{D}_q$
  $\mathcal{D}_q = \{\mathbf{z}_i^q = (\mathbf{x}_i^q, y_i^q)\}_{i=1}^{N_q}$ sampled from $\mathcal{P}_q$, $q = 1, \ldots, Q$
  $\mathbf{x} \in \mathbb{R}^d$–input variable, $y \in \mathbb{R}$–response

- **Model**

$$f_q(\mathbf{x}) = \mathbf{w}^{q\top}\mathbf{x}, \quad q = 1, \ldots, Q$$

- **Objective**

$$\min_{\mathbf{W}} \quad \sum_{q=1}^{Q} \frac{1}{N_q} \sum_{i=1}^{N_q} \ell^q(\mathbf{W}_{\bullet q}, \mathbf{z}_i^q) + \Omega_\lambda(\mathbf{W})$$

$$\mathbf{W} = \left(\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^Q\right) = (\mathbf{W}_{\bullet 1}, \ldots, \mathbf{W}_{\bullet Q}) = \left(\mathbf{W}_{1\bullet}^\top, \ldots, \mathbf{W}_{d\bullet}^\top\right)^\top$$

Multi-Task Feature Selection

- **Data**

i.i.d. observations of $\mathcal{D} = \bigcup_{q=1}^{Q} \mathcal{D}_q$
$\mathcal{D}_q = \{\mathbf{z}_i^q = (\mathbf{x}_i^q, y_i^q)\}_{i=1}^{N_q}$ sampled from $\mathcal{P}_q$, $q = 1, \ldots, Q$
$\mathbf{x} \in \mathbb{R}^d$–input variable, $y \in \mathbb{R}$–response

- **Model**

$$\boxed{f_q(\mathbf{x}) = \mathbf{w}^{q\top}\mathbf{x}, \quad q = 1, \ldots, Q}$$

- **Objective**

$$\boxed{\min_{\mathbf{W}} \quad \sum_{q=1}^{Q} \frac{1}{N_q} \sum_{i=1}^{N_q} \ell^q(\mathbf{W}_{\bullet q}, \mathbf{z}_i^q) + \Omega_\lambda(\mathbf{W})}$$

$$\mathbf{W} = \left(\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^Q\right) = \left(\mathbf{W}_{\bullet 1}, \ldots, \mathbf{W}_{\bullet Q}\right) = \left(\mathbf{W}_{1\bullet}^\top, \ldots, \mathbf{W}_{d\bullet}^\top\right)^\top$$

Multi-Task Feature Selection

- **Data**

$$\boxed{\begin{array}{l}\text{i.i.d. observations of } \mathcal{D} = \bigcup_{q=1}^{Q} \mathcal{D}_q \\ \mathcal{D}_q = \{\mathbf{z}_i^q = (\mathbf{x}_i^q, y_i^q)\}_{i=1}^{N_q} \text{ sampled from } \mathcal{P}_q,\ q = 1, \ldots, Q \\ \mathbf{x} \in \mathbb{R}^d\text{–input variable, } y \in \mathbb{R}\text{–response}\end{array}}$$

- **Model**

$$\boxed{f_q(\mathbf{x}) = \mathbf{w}^{q\top}\mathbf{x}, \quad q = 1, \ldots, Q}$$

- **Objective**

$$\boxed{\min_{\mathbf{W}} \quad \sum_{q=1}^{Q} \frac{1}{N_q} \sum_{i=1}^{N_q} \ell^q(\mathbf{W}_{\bullet q}, \mathbf{z}_i^q) + \Omega_\lambda(\mathbf{W})}$$

$$\mathbf{W} = \left(\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^Q\right) = (\mathbf{W}_{\bullet 1}, \ldots, \mathbf{W}_{\bullet Q}) = \left(\mathbf{W}_{1\bullet}^\top, \ldots, \mathbf{W}_{d\bullet}^\top\right)^\top$$

Multi-Task Feature Selection

- Different regularization achieves different properties
- Regularization
  - **iMTFS:** $\Omega_\lambda(\mathbf{W}) = \lambda \sum_{q=1}^{Q} \|\mathbf{W}_{\bullet q}\|_1 = \lambda \sum_{j=1}^{d} \|\mathbf{W}_{j\bullet}^\top\|_1$
  - **aMTFS:** $\Omega_\lambda(\mathbf{W}) = \lambda \sum_{j=1}^{d} \|\mathbf{W}_{j\bullet}^\top\|_2$
  - **MTFTS:** $\Omega_{\lambda,\mathbf{r}} = \lambda \sum_{j=1}^{d} \left(r_j \|\mathbf{W}_{j\bullet}^\top\|_1 + \|\mathbf{W}_{j\bullet}^\top\|_2 \right)$

$$
\overset{\textbf{iMTFS}}{\begin{pmatrix} x & 0 & 0 & x & x \\ 0 & x & x & x & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x & 0 & x & x & x \end{pmatrix}},
\overset{\textbf{aMTFS}}{\begin{pmatrix} x & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x & x & x & x & x \end{pmatrix}},
\overset{\textbf{MTFTS}}{\begin{pmatrix} x & 0 & x & x & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & x & 0 & x & x \end{pmatrix}}
$$

Multi-Task Feature Selection

- Different regularization achieves different properties
- Regularization
  - **iMTFS:** $\Omega_\lambda(\mathbf{W}) = \lambda \sum_{q=1}^{Q} \|\mathbf{W}_{\bullet q}\|_1 = \lambda \sum_{j=1}^{d} \|\mathbf{W}_{j\bullet}^\top\|_1$
  - **aMTFS:** $\Omega_\lambda(\mathbf{W}) = \lambda \sum_{j=1}^{d} \|\mathbf{W}_{j\bullet}^\top\|_2$
  - **MTFTS:** $\Omega_{\lambda,\mathbf{r}} = \lambda \sum_{j=1}^{d} \left(r_j \|\mathbf{W}_{j\bullet}^\top\|_1 + \|\mathbf{W}_{j\bullet}^\top\|_2 \right)$

$$\overset{\textbf{iMTFS}}{\begin{pmatrix} x & 0 & 0 & x & x \\ 0 & x & x & x & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x & 0 & x & x & x \end{pmatrix}}, \quad \overset{\textbf{aMTFS}}{\begin{pmatrix} x & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x & x & x & x & x \end{pmatrix}}, \quad \overset{\textbf{MTFTS}}{\begin{pmatrix} x & 0 & x & x & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & x & 0 & x & x \end{pmatrix}}$$

Multi-Task Feature Selection

- Different regularization achieves different properties
- Regularization
  - $\boxed{\textbf{iMTFS: } \Omega_\lambda(\mathbf{W}) = \lambda \sum_{q=1}^{Q} \|\mathbf{W}_{\bullet q}\|_1 = \lambda \sum_{j=1}^{d} \left\|\mathbf{W}_{j\bullet}^\top\right\|_1}$
  - $\boxed{\textbf{aMTFS: } \Omega_\lambda(\mathbf{W}) = \lambda \sum_{j=1}^{d} \left\|\mathbf{W}_{j\bullet}^\top\right\|_2}$
  - $\boxed{\textbf{MTFTS: } \Omega_{\lambda,\mathbf{r}} = \lambda \sum_{j=1}^{d} \left(r_j \left\|\mathbf{W}_{j\bullet}^\top\right\|_1 + \left\|\mathbf{W}_{j\bullet}^\top\right\|_2 \right)}$

$$
\underset{\textbf{iMTFS}}{\begin{pmatrix} x & 0 & 0 & x & x \\ 0 & x & x & x & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x & 0 & x & x & x \end{pmatrix}},
\underset{\textbf{aMTFS}}{\begin{pmatrix} x & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x & x & x & x & x \end{pmatrix}},
\underset{\textbf{MTFTS}}{\begin{pmatrix} x & 0 & x & x & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & x & 0 & x & x \end{pmatrix}}
$$

Online Learning Algorithm Framework for MTFS

Initialization: $\mathbf{W}_1 = \mathbf{W}_0$, $\bar{\mathbf{G}}_0 = \mathbf{0}$

for $t = 1, 2, 3, \ldots$

1. Compute the subgradient on $\mathbf{W}_t$, $\mathbf{G}_t \in \partial\, l_t$
2. Calculate the average subgradient $\bar{\mathbf{G}}_t$:
   $\bar{\mathbf{G}}_t = \frac{t-1}{t}\bar{\mathbf{G}}_{t-1} + \frac{1}{t}\mathbf{G}_t$
3. Update the next iteration $\mathbf{W}_{t+1}$:
   $$\mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \Upsilon(\mathbf{W}) \triangleq \left\{\bar{\mathbf{G}}_t^\top \mathbf{W} + \Omega(\mathbf{W}) + \frac{\gamma}{\sqrt{t}} h(\mathbf{W})\right\}$$

end for

Remarks

- $\mathbf{W}$: a matrix, not a vector
- Easily extend to non-linear case
- Motivated by the success of dual averaging method (Xiao, 2009; Yang et al. 2010)

Updating Rules for Online MTFS

Define: $h(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2$

- **iMTFS**: For $i = 1, \ldots, d$ and $q = 1, \ldots, Q$,

$$\boxed{(W_{i,q})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[|(\bar{G}_{i,q})_t| - \lambda\right]_+ \cdot \mathrm{sign}\left((\bar{G}_{i,q})_t\right)}$$

- **aMTFS**: For $j = 1, \ldots, d$,

$$\boxed{(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{G}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{G}}_{j\bullet})_t}$$

- **MTFTS**: For $j = 1, \ldots, d$,

$$\boxed{(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{U}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{U}}_{j\bullet})_t}$$

  where the q-th element of $(\bar{\mathbf{U}}_{j\bullet})_t$ is calculated by

$$(\bar{U}_{j,q})_t = \left[|(\bar{G}_{j,q})_t| - \lambda r_j\right]_+ \cdot \mathrm{sign}\left((\bar{G}_{j,q})_t\right), \; q = 1, \ldots, Q.$$

Efficiency: $\mathcal{O}(d \times Q)$ in memory cost and time complexity

Updating Rules for Online MTFS

Define: $h(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2$

- **iMTFS**: For $i = 1, \ldots, d$ and $q = 1, \ldots, Q$,

$$\boxed{(W_{i,q})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[|(\bar{G}_{i,q})_t| - \lambda\right]_+ \cdot \mathrm{sign}\left((\bar{G}_{i,q})_t\right)}$$

- **aMTFS**: For $j = 1, \ldots, d$,

$$\boxed{(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{G}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{G}}_{j\bullet})_t}$$

- **MTFTS**: For $j = 1, \ldots, d$,

$$\boxed{(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{U}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{U}}_{j\bullet})_t}$$

where the q-th element of $(\bar{\mathbf{U}}_{j\bullet})_t$ is calculated by

$$(\bar{U}_{j,q})_t = \left[|(\bar{G}_{j,q})_t| - \lambda r_j\right]_+ \cdot \mathrm{sign}\left((\bar{G}_{j,q})_t\right), \; q = 1, \ldots, Q.$$

Efficiency: $\mathcal{O}(d \times Q)$ in memory cost and time complexity

Updating Rules for Online MTFS

Define: $h(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2$

- **iMTFS**: For $i = 1, \ldots, d$ and $q = 1, \ldots, Q$,

$$\boxed{(W_{i,q})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[|(\bar{G}_{i,q})_t| - \lambda\right]_+ \cdot \operatorname{sign}\left((\bar{G}_{i,q})_t\right)}$$

- **aMTFS**: For $j = 1, \ldots, d$,

$$\boxed{(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{G}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{G}}_{j\bullet})_t}$$

- **MTFTS**: For $j = 1, \ldots, d$,

$$\boxed{(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{U}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{U}}_{j\bullet})_t}$$

  where the q-th element of $(\bar{\mathbf{U}}_{j\bullet})_t$ is calculated by

$$(\bar{U}_{j,q})_t = \left[|(\bar{G}_{j,q})_t| - \lambda r_j\right]_+ \cdot \operatorname{sign}\left((\bar{G}_{j,q})_t\right), \ q = 1, \ldots, Q.$$

Efficiency: $\mathcal{O}(d \times Q)$ in memory cost and time complexity

Average Regret for MTFS

- Definition

$$\bar{R}_T(\mathbf{W}) := \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{T}\sum_{t=1}^{T}\left(\Omega_\lambda(\mathbf{W}_t) + l_t(\mathbf{W}_t)\right) - S_T(\mathbf{W})$$

$$S_T(\mathbf{W}) := \min_{\mathbf{W}} \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{T}\sum_{t=1}^{T}\left(\Omega_\lambda(\mathbf{W}) + l_t(\mathbf{W})\right)$$

- Theoretical bounds

$$\bar{R}_T \sim \mathcal{O}(1/\sqrt{T})$$

$$\bar{R}_T \sim \mathcal{O}(\log(T)/T) \quad \text{if } h(\cdot) \text{ is strongly convex}$$

Average Regret for MTFS

- Definition

$$\begin{aligned} \bar{R}_T(\mathbf{W}) &:= \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{T}\sum_{t=1}^{T}\left(\Omega_\lambda(\mathbf{W}_t) + l_t(\mathbf{W}_t)\right) - S_T(\mathbf{W}) \\ S_T(\mathbf{W}) &:= \min_{\mathbf{W}}\frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{T}\sum_{t=1}^{T}\left(\Omega_\lambda(\mathbf{W}) + l_t(\mathbf{W})\right) \end{aligned}$$

- Theoretical bounds

$$\begin{aligned} &\bar{R}_T \sim \mathcal{O}(1/\sqrt{T}) \\ &\bar{R}_T \sim \mathcal{O}(\log(T)/T) \quad \text{if } h(\cdot) \text{ is strongly convex} \end{aligned}$$

Experimental Setup for Online MTFS

- **Data**
  - ★ Computer survey data
- **Comparison algorithms**
  - ★ iMTFS
  - ★ aMTFS
  - ★ DA-iMTFS
  - ★ DA-aMTFS
  - ★ DA-MTFTS
- **Platform**
  - ★ PC with 2.13 GHz dual-core CPU
  - ★ Batch-mode algorithms: Matlab
  - ★ Online-mode algorithms: Matlab

Conjoint Analysis

- **Description**
  - **Objective:** Predict rating by estimating respondents' partworths vectors
  - **Data:** Ratings on personal computers of 180 students for 20 different PC, $Q = 180$
  - **Features:** Telephone hot line (TE), amount of memory (RAM), screen size (SC), CPU speed (CPU), hard disk (HD), CDROM/multimedia (CD), cache (CA), color (CO), availability (AV), warranty (WA), software (SW), guarantee (GU) and price (PR); $d = 14$
- **Setup**
  - Evaluation: Root mean square errors (RMSEs)
  - Loss: Square loss
  - Parameters setting: Cross validation (hierarchical and grid search)

Conjoint Analysis Results

Accuracy

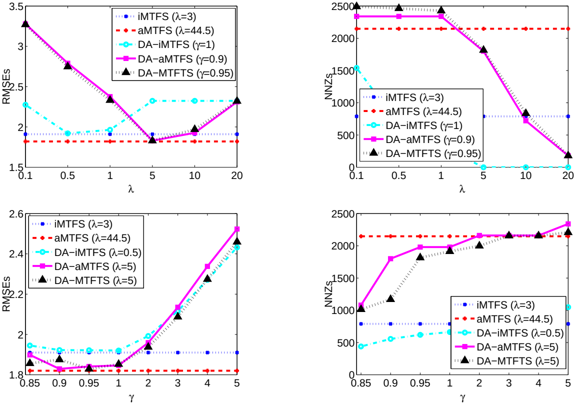
- Learning partworths vectors across respondents can help to improve the performance
- Online learning algorithms attain nearly the same accuracies as batch-trained algorithms

Method	RMSEs	NNZs	Parameters
aMTFS	**1.82**	2148	$\lambda = 44.5$
iMTFS	1.91	789	$\lambda = 3$
DA-aMTFS	2.04	540	$\lambda = 20.0, \gamma = 0.9$, ep=1
DA-aMTFS	**1.83**	1800	$\lambda = 5, \gamma = 0.9$, ep=20
DA-iMTFS	2.43	199	$\lambda = 2.0, \gamma = 2.0$, ep=1
DA-iMTFS	1.92	662	$\lambda = 0.5, \gamma = 1.0$, ep=20

Effect of λ and γ

Results

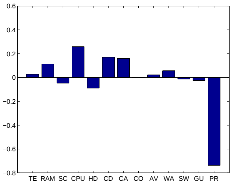
- NNZs decreases as λ increases
- NNZs increases as γ increases



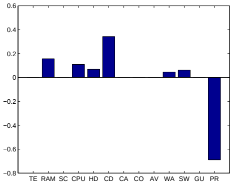
Learned Features

Results

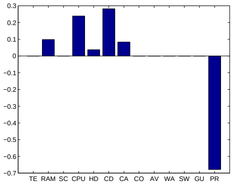
- Features learned from the online algorithms are consistent to those learned from the batch-trained algorithm
- Ratings are strongly negative to the price and positive to the RAM, the CPU speed, CDROM, etc.



aMTFS



DA-aMTFS



DA-MTFTS

Summary

Summary

- A novel online learning algorithm framework for multi-task feature selection
- Apply this framework for variant multi-task feature selection models
- Provide closed-form solutions to update the models
- Provide the convergence rate of the average regret
- Experimental results demonstrate the proposed algorithms in both efficiency and effectiveness

Outline

How to Define Data Similarity?

Horse

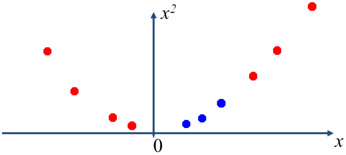
Donkey

What are Kernels?

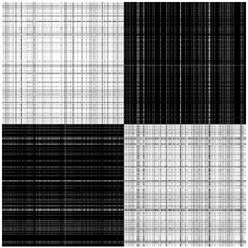
- Similarity defined in original space: $\mathbf{x}_i^T \mathbf{x}_j$
- Similarity defined in kernel space: $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

Original space | Mapping | 2D space

$\Longrightarrow$



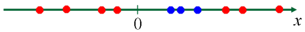
From linear kernel | From RBF kernel



What are Kernels?

- Similarity defined in original space: $\mathbf{x}_i^T \mathbf{x}_j$
- Similarity defined in kernel space: $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

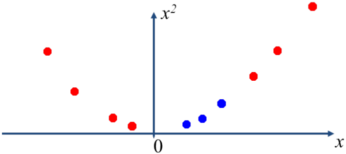
Original space



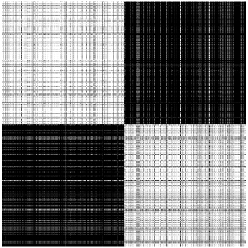
Mapping

$\Longrightarrow$

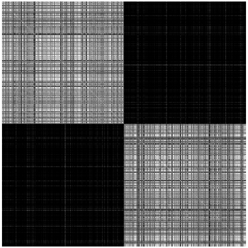
2D space



From linear kernel



From RBF kernel



The Kernel Trick: An Example

- Suppose the vectors $\mathbf{x} = [x_1; x_2] \in \mathbb{R}^2$
- Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
- Question: Show $\phi(\mathbf{x})$, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

$$
\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{j2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1;\ x_{i1}^2;\ \sqrt{2}x_{i1}x_{i2};\ x_{i2}^2;\ \sqrt{2}x_{i1};\ \sqrt{2}x_{i2}]^T \\
&\quad \times [1;\ x_{j1}^2;\ \sqrt{2}x_{j1}x_{j2};\ x_{j2}^2;\ \sqrt{2}x_{j1};\ \sqrt{2}x_{j2}] \\
&= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)
\end{aligned}
$$

where $\phi(\mathbf{x}) = [1; x_1^2; \sqrt{2}x_1x_2; x_2^2; \sqrt{2}x_1; \sqrt{2}x_2] \in \mathbb{R}^6$

The Kernel Trick: An Example

- Suppose the vectors $\mathbf{x} = [x_1; x_2] \in \mathbb{R}^2$
- Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
- Question: Show $\phi(\mathbf{x})$, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

$$
\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1;\ x_{i1}^2;\ \sqrt{2}x_{i1}x_{i2};\ x_{i2}^2;\ \sqrt{2}x_{i1};\ \sqrt{2}x_{i2}]^T \\
&\quad \times [1;\ x_{j1}^2;\ \sqrt{2}x_{j1}x_{j2};\ x_{j2}^2;\ \sqrt{2}x_{j1};\ \sqrt{2}x_{j2}] \\
&= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)
\end{aligned}
$$

where $\phi(\mathbf{x}) = [1; x_1^2; \sqrt{2}x_1x_2; x_2^2; \sqrt{2}x_1; \sqrt{2}x_2] \in \mathbb{R}^6$

The Kernel Trick: An Example

- Suppose the vectors $\mathbf{x} = [x_1; x_2] \in \mathbb{R}^2$
- Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
- Question: Show $\phi(\mathbf{x})$, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

$$
\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{j2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1;\ x_{i1}^2;\ \sqrt{2}x_{i1}x_{i2};\ x_{i2}^2;\ \sqrt{2}x_{i1};\ \sqrt{2}x_{i2}]^T \\
&\quad \times [1;\ x_{j1}^2;\ \sqrt{2}x_{j1}x_{j2};\ x_{j2}^2;\ \sqrt{2}x_{j1};\ \sqrt{2}x_{j2}] \\
&= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)
\end{aligned}
$$

where $\phi(\mathbf{x}) = [1; x_1^2; \sqrt{2}x_1x_2; x_2^2; \sqrt{2}x_1; \sqrt{2}x_2] \in \mathbb{R}^6$

The Kernel Trick: An Example

- Suppose the vectors $\mathbf{x} = [x_1; x_2] \in \mathbb{R}^2$
- Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
- Question: Show $\phi(\mathbf{x})$, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

$$
\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{j2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1;\ x_{i1}^2;\ \sqrt{2}x_{i1}x_{i2};\ x_{i2}^2;\ \sqrt{2}x_{i1};\ \sqrt{2}x_{i2}]^T \\
&\quad \times [1;\ x_{j1}^2;\ \sqrt{2}x_{j1}x_{j2};\ x_{j2}^2;\ \sqrt{2}x_{j1};\ \sqrt{2}x_{j2}] \\
&= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)
\end{aligned}
$$

where $\phi(\mathbf{x}) = [1; x_1^2; \sqrt{2}x_1x_2; x_2^2; \sqrt{2}x_1; \sqrt{2}x_2] \in \mathbb{R}^6$

The Kernel Trick: An Example

- Suppose the vectors $\mathbf{x} = [x_1; x_2] \in \mathbb{R}^2$
- Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
- Question: Show $\phi(\mathbf{x})$, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

$$
\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{j2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1;\ x_{i1}^2;\ \sqrt{2}x_{i1}x_{i2};\ x_{i2}^2;\ \sqrt{2}x_{i1};\ \sqrt{2}x_{i2}]^T \\
&\quad \times [1;\ x_{j1}^2;\ \sqrt{2}x_{j1}x_{j2};\ x_{j2}^2;\ \sqrt{2}x_{j1};\ \sqrt{2}x_{j2}] \\
&= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)
\end{aligned}
$$

where $\phi(\mathbf{x}) = [1; x_1^2; \sqrt{2}x_1x_2; x_2^2; \sqrt{2}x_1; \sqrt{2}x_2] \in \mathbb{R}^6$

The Kernel Trick: An Example

- Suppose the vectors $\mathbf{x} = [x_1; x_2] \in \mathbb{R}^2$
- Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
- Question: Show $\phi(\mathbf{x})$, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\ &= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{j2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\ &= [1;\ x_{i1}^2;\ \sqrt{2}x_{i1}x_{i2};\ x_{i2}^2;\ \sqrt{2}x_{i1};\ \sqrt{2}x_{i2}]^T \\ &\quad \times [1;\ x_{j1}^2;\ \sqrt{2}x_{j1}x_{j2};\ x_{j2}^2;\ \sqrt{2}x_{j1};\ \sqrt{2}x_{j2}] \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \end{aligned}$$

where $\phi(\mathbf{x}) = [1; x_1^2; \sqrt{2}x_1x_2; x_2^2; \sqrt{2}x_1; \sqrt{2}x_2] \in \mathbb{R}^6$

The Kernel Trick: An Example

- Suppose the vectors $\mathbf{x} = [x_1; x_2] \in \mathbb{R}^2$
- Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
- Question: Show $\phi(\mathbf{x})$, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

$$
\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{j2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1;\ x_{i1}^2;\ \sqrt{2}x_{i1}x_{i2};\ x_{i2}^2;\ \sqrt{2}x_{i1};\ \sqrt{2}x_{i2}]^T \\
&\quad \times [1;\ x_{j1}^2;\ \sqrt{2}x_{j1}x_{j2};\ x_{j2}^2;\ \sqrt{2}x_{j1};\ \sqrt{2}x_{j2}] \\
&= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)
\end{aligned}
$$

where $\phi(\mathbf{x}) = [1; x_1^2; \sqrt{2}x_1x_2; x_2^2; \sqrt{2}x_1; \sqrt{2}x_2] \in \mathbb{R}^6$

What Functions are Kernels?

- Functions that satisfy *Mercer's condition* can be kernel functions. That is

$$\forall \text{ square integrable functions } g(x), \iint K(x,y)g(x)g(y)dxdy \geq 0$$

- Examples of typical kernel functions:
  - Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
  - Gaussian/Radial-Basis Function (RBF) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

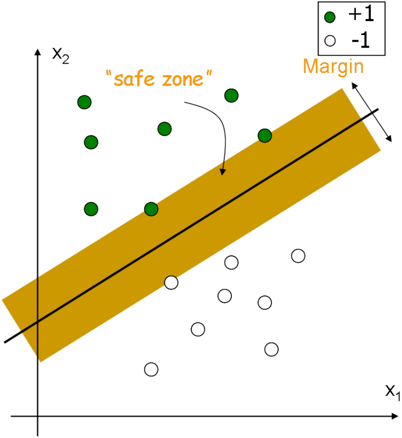
  - Hyperbolic tangent:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i^T \mathbf{x}_j + c), \text{ for some } \kappa > 0, \text{ and } c < 0$$

What is the relation between Kernel and SVM?

SVM–Maximum Margin Linear Classifier

- A linear classifier with the maximum margin
- Margin is defined as the width that the boundary could be increased by before hitting a data point
- Why it is the best?
  - Robust to outliers
  - Strong generalization ability



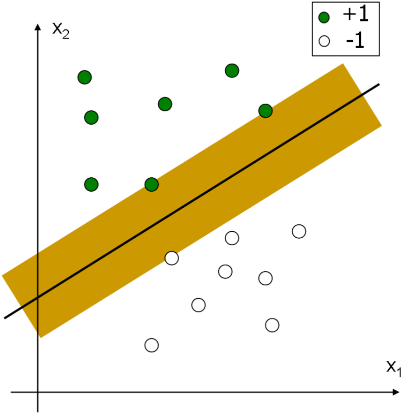
SVM–Maximum Margin Linear Classifier

- Given data, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$

$$\text{For } y_i = +1, \qquad \mathbf{w}^T\mathbf{x}_i + b > 0$$
$$\text{For } y_i = -1, \qquad \mathbf{w}^T\mathbf{x}_i + b < 0$$

- Scaling on both $\mathbf{w}$ and b yields

$$\text{For } y_i = +1, \qquad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$
$$\text{For } y_i = -1, \qquad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



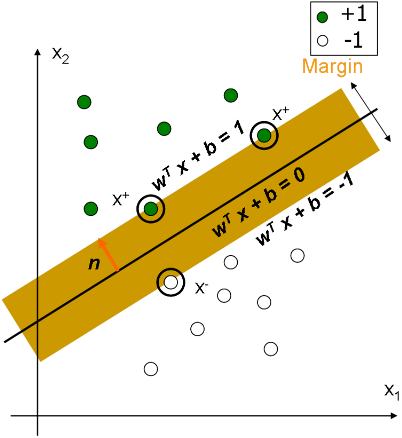
SVM–Maximum Margin Linear Classifier

- **Support vectors**: Data points closest to the hyperplane
- Support vectors satisfy

$$
\begin{aligned}
\mathbf{w}^T\mathbf{x}^+ + b &= 1 \\
\mathbf{w}^T\mathbf{x}^- + b &= -1
\end{aligned}
$$

- The margin width is

$$
\begin{aligned}
M &= (\mathbf{x}^+ - \mathbf{x}^-)^T\mathbf{n} \\
&= (\mathbf{x}^+ - \mathbf{x}^-)^T\frac{\mathbf{w}}{\|\mathbf{w}\|} \\
&= \frac{2}{\|\mathbf{w}\|}
\end{aligned}
$$



SVM–Maximum Margin Linear Classifier

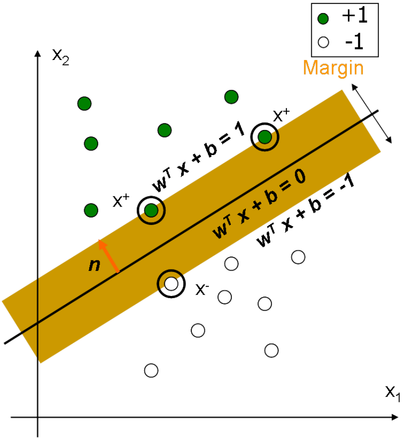
- Formulation

$$\max_{\mathbf{w}} \quad \frac{2}{\|\mathbf{w}\|}$$

such that

For $y_i = +1, \quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$

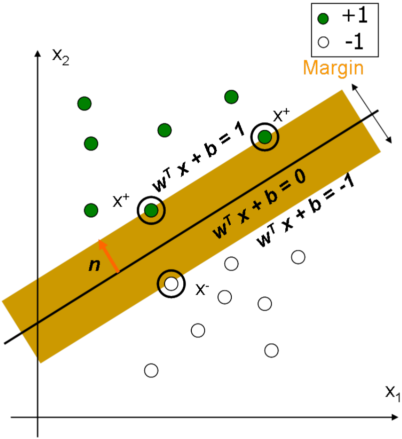
For $y_i = -1, \quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$



SVM–Maximum Margin Linear Classifier

- Formulation

$$\begin{array}{ll} \min_{\mathbf{w}} & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \\ & i = 1, \ldots, N \end{array}$$



How to Solve the Optimization Problem?

- Quadratic programming with linear constraints

$$\begin{array}{ll} \min_{\mathbf{w}} & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, N \end{array}$$

- Lagrangian multipliers

$$\begin{array}{ll} \min & \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right) \\ \text{s.t.} & \boldsymbol{\alpha} \geq \mathbf{0} \end{array}$$

- Optimal condition

$$\begin{array}{lll} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 & \Longrightarrow & \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \Longrightarrow & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{array}$$

How to Solve the Optimization Problem?

- Quadratic programming with linear constraints

$$\begin{array}{ll} \min_{\mathbf{w}} & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, N \end{array}$$

- Lagrangian multipliers

$$\begin{array}{ll} \min & \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right) \\ \text{s.t.} & \boldsymbol{\alpha} \geq \mathbf{0} \end{array}$$

- Optimal condition

$$\begin{array}{lll} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 & \Longrightarrow & \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \Longrightarrow & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{array}$$

How to Solve the Optimization Problem?

- Quadratic programming with linear constraints

$$\boxed{\begin{array}{ll} \min\limits_{\mathbf{w}} & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, N \end{array}}$$

- Lagrangian multipliers

$$\boxed{\begin{array}{ll} \min & \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum\limits_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right) \\ \text{s.t.} & \boldsymbol{\alpha} \geq \mathbf{0} \end{array}}$$

- Optimal condition

$$\boxed{\begin{array}{lcl} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 & \Longrightarrow & \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \Longrightarrow & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{array}}$$

How to Solve the Optimization Problem?

- Lagrangian multipliers

$$\begin{aligned} \min \quad & \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) \\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq \mathbf{0} \end{aligned}$$

- Dual problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq \mathbf{0}, \ \text{and} \ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned}$$

How to Solve the Optimization Problem?

- Lagrangian multipliers

$$
\begin{array}{ll}
\min & \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) \\
\text{s.t.} & \boldsymbol{\alpha} \geq \mathbf{0}
\end{array}
$$

- **Dual problem**

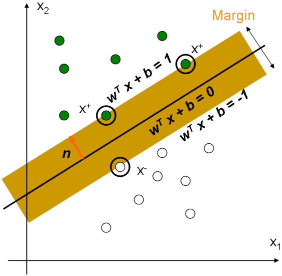
$$
\begin{array}{ll}
\max\limits_{\boldsymbol{\alpha}} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\
\text{s.t.} & \boldsymbol{\alpha} \geq \mathbf{0}, \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0
\end{array}
$$

SVM Solution

- KKT conditions are
  $\alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1\right) = 0, \quad i = 1, \ldots, N$
- **Support vectors**: $\alpha_i \neq 0$
- The solution is

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = \sum_{k \in SV} \alpha_k y_k \mathbf{x}_k$$

  Extract b from
  $\alpha_k \left(y_k(\mathbf{w}^T \mathbf{x}_k + b) - 1\right) = 0$,
  where $k \in SV$



SVM Solution

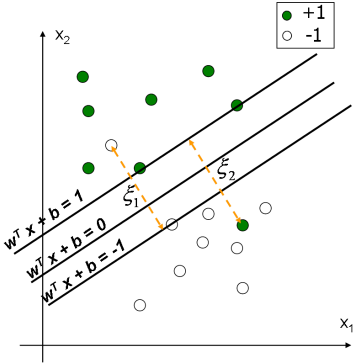
- The linear classifier is

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i \in SV} \alpha_i \mathbf{x}_i^T \mathbf{x} + b$$

- The score is decided by the *dot product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$
- It is noticed that solving the optimization problem also involved computing the *dot products* $\mathbf{x}_i^T\mathbf{x}_j$ between all pairs of training data points

SVM–Non-separable Case

- What if data is not linear separable? (noisy data, outlier, etc.)
- Slack variables ξ_i are introduced to allow misclassification on difficult or noisy data points



SVM–Non-separable Case

- Formulation

$$\boxed{\begin{array}{ll} \min_{\mathbf{w}} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \ldots, N \end{array}}$$

- Parameter C is to balance the margin and the errors, which can be also viewed as a way to control over-fitting.

SVM–Non-separable Case

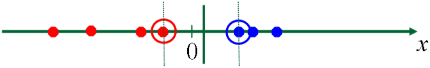
- Formulation–Lagrangian dual problem

$$\boxed{\begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} & \mathbf{0} \le \boldsymbol{\alpha} \le C\mathbf{1}_N, \\ & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{array}}$$

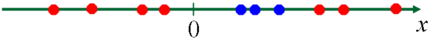
- How to seek the optimal $\boldsymbol{\alpha}$?
  - Convexity: The optimization is convex; every local optimal is the global optimal!
  - Optimization techniques: Sequential minimal optimization (SMO), etc.

Non-linear SVMs

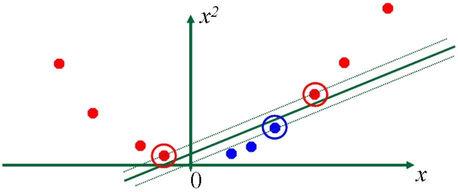
- Datasets that are linearly separable with noise work out great:



- But what are we going to do if the dataset is just too hard?

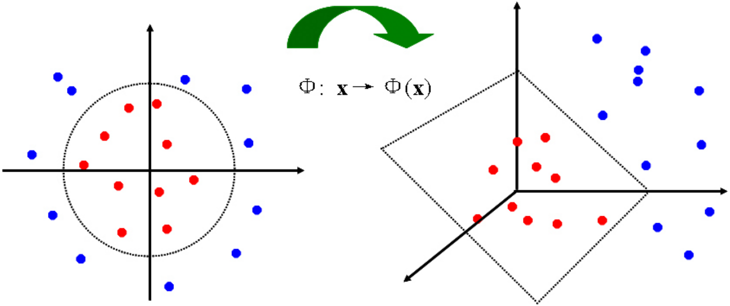


- How about mapping data to a higher-dimensional space:



Non-linear SVMs: Feature Space

- Idea: Make the data separable by mapping it to a (higher-dimensional) feature space



Non-linear SVMs: The Kernel Trick

- With the mapping, the discriminant function becomes

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in SV} \alpha_i \boxed{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b$$

- Only the *dot product* of feature vectors are needed. No need to know the mapping explicitly.
- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Non-linear SVMs: Optimization

- Formulation-Lagrangian Dual problem

$$\boxed{\begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & \sum\limits_{i=1}^{N} \alpha_i - \frac{1}{2} \sum\limits_{i=1}^{N} \sum\limits_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_N, \\ & \sum\limits_{i=1}^{N} \alpha_i y_i = 0 \end{array}}$$

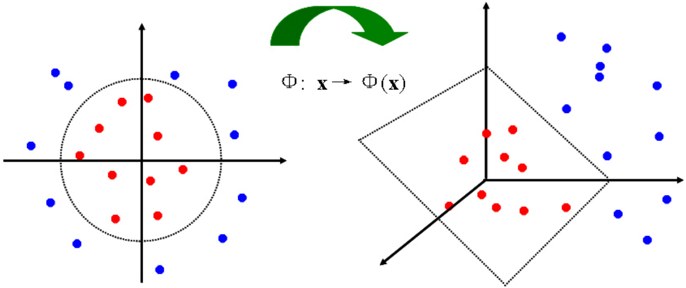
- The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in SV} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- The optimization technique is the same as the linear SVM

Non-linear SVMs–Overview

- SVM seeks a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product (similarity measurement) in the feature space



Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution
    - Only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
    - Complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution

Packages

- LibSVM: A Library for Support Vector Machines
  - An integrated software for SVM; core codes are written in C++
  - Implementation includes: C-SVC, ν-SVC, ϵ-SVR, ν-SVR, one-class SVM, multi-class classification
  - Link: `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
  - R package: `http://cran.r-project.org/web/packages/e1071/index.html`
- SVMlight
  - An SVM package in C
  - Link: `http://svmlight.joachims.org/`

R Packages for SVM

- Link `http://cran.r-project.org/web/packages/e1071/index.html`

An Example

```
> # load library, class, a dependence for the SVM library
> library(class)

> # load library, SVM
> library(e1071)

> # load library, mlbench, a collection of some datasets from the UCI repository
> library(mlbench)

> # load data
> data(Glass, package = "mlbench")

> # get the index of all data
> index <- 1:nrow(Glass)

> # generate test index
> testindex <- sample(index, trunc(length(index)/3))

> # generate test set
> testset <- Glass[testindex, ]

> # generate trainin set
> trainset <- Glass[-testindex, ]
```

An Example (2)

```
> # train svm on the training set
> # cost=100: the penalizing parameter for C-classication
> # gamma=1: the radial basis function-specific kernel parameter
> # Output values include SV, index, coefs, rho, sigma, probA, probB
> svm.model <- svm(Type~ ., data = trainset, cost = 100, gamma = 1)

> # show output coefficients
> svm.model$coefs
```

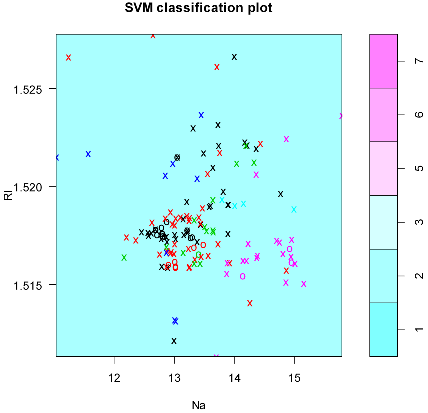
```
> # generate a scatter plot of the data
> # of a svm fit for classification model
> # in two dimensions: RI and Na
> plot(svm.model, trainset, RI~Na)

> # a vector of predicted values,
> # for classification: a vector of labels
> svm.pred <- predict(svm.model, testset[, -10])

> # a cross-tabulation of the true
> # versus the predicted values
> table(pred = svm.pred, true = testset[, 10])
```

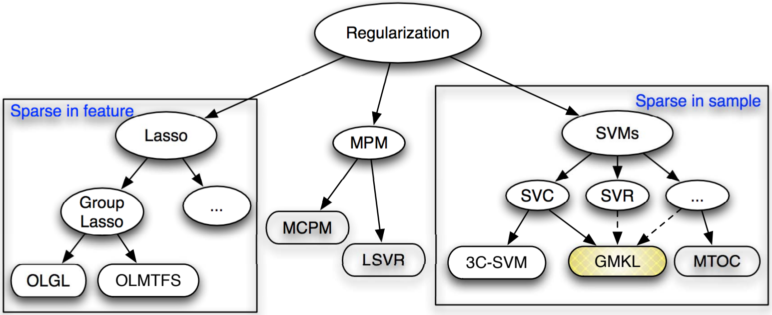
	true					
pred	1	2	3	5	6	7
1	16	3	1	0	1	0
2	7	23	3	3	2	1
3	0	1	1	0	0	0
5	0	0	0	2	0	0
6	0	0	0	0	1	0
7	0	0	0	0	0	6

SVM Plot Figure



Outline

Sparse Generalized Multiple Kernel Learning

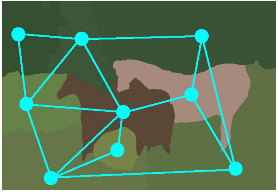


- H. Yang, Z. Xu, J. Ye, I. King, and M. R. Lyu. Efficient sparse generalized multiple kernel learning. *IEEE Transactions on Neural Networks*, 22(3):433–446, March 2011.
- Toolbox: `http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=GMKL`

How to Measure Data Similarity More Accurately?

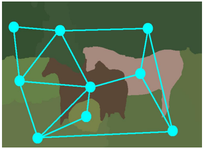
Labeled: Horse

Labeled: Donkey

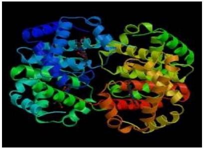


- Data characteristics
  - Multi-source
  - Heterogeneous

Why Multiple Kernel Learning?



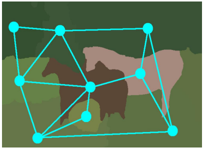
Harchaoui & Bach, 2007



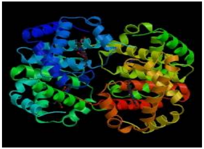
Zien & Ong, 2007

- Applications: Multi-source data fusion (web classification, genome fusion); Image annotation; Text mining; etc.
- Characteristics: Complex tasks; Heterogenous–various medias (text, images, etc.); Huge data
- Solution: Kernel methods⇒Multiple kernels learning
  - Learning combinations of kernels: $K = \sum_{q=1}^{Q} \theta_q K_q$, $\theta_q \geq 0$

Why Multiple Kernel Learning?



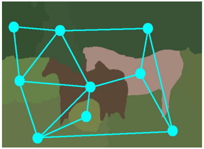
Harchaoui & Bach, 2007



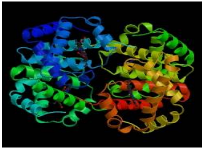
Zien & Ong, 2007

- Applications: Multi-source data fusion (web classification, genome fusion); Image annotation; Text mining; etc.
- Characteristics: Complex tasks; Heterogenous–various medias (text, images, etc.); Huge data
- Solution: Kernel methods⇒Multiple kernels learning
  - Learning combinations of kernels: $\mathcal{K} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q, \ \theta_q \geq 0$

Why Multiple Kernel Learning?



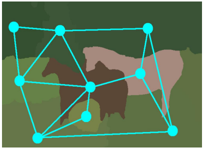
Harchaoui & Bach, 2007



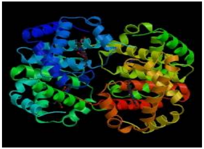
Zien & Ong, 2007

- Applications: Multi-source data fusion (web classification, genome fusion); Image annotation; Text mining; etc.
- Characteristics: Complex tasks; Heterogenous–various medias (text, images, etc.); Huge data
- Solution: Kernel methods⇒Multiple kernels learning
  - Learning combinations of kernels: $\mathcal{K} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q$, $\theta_q \geq 0$
    - Summing kernels corresponds to concatenating feature spaces

Why Multiple Kernel Learning?



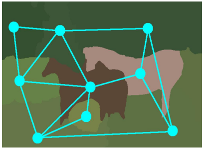
Harchaoui & Bach, 2007



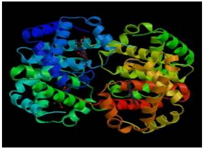
Zien & Ong, 2007

- Applications: Multi-source data fusion (web classification, genome fusion); Image annotation; Text mining; etc.
- Characteristics: Complex tasks; Heterogenous–various medias (text, images, etc.); Huge data
- Solution: Kernel methods⇒Multiple kernels learning
  - Learning combinations of kernels: $\mathcal{K} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q$, $\theta_q \geq 0$
    - Summing kernels corresponds to concatenating feature spaces
    - E.g., $k_1(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_1(\mathbf{z}_1), \phi_1(\mathbf{z}_2) \rangle$, $k_2(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_2(\mathbf{z}_1), \phi_2(\mathbf{z}_2) \rangle$

Why Multiple Kernel Learning?



Harchaoui & Bach, 2007

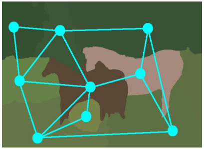


Zien & Ong, 2007

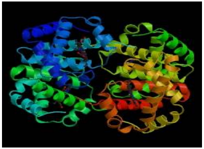
- Applications: Multi-source data fusion (web classification, genome fusion); Image annotation; Text mining; etc.
- Characteristics: Complex tasks; Heterogenous–various medias (text, images, etc.); Huge data
- Solution: Kernel methods⇒Multiple kernels learning
  - Learning combinations of kernels: $\mathcal{K} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q$, $\theta_q \geq 0$
    - Summing kernels corresponds to concatenating feature spaces
    - E.g., $k_1(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_1(\mathbf{z}_1), \phi_1(\mathbf{z}_2) \rangle$, $k_2(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_2(\mathbf{z}_1), \phi_2(\mathbf{z}_2) \rangle$

$$k_1(\mathbf{z}_1, \mathbf{z}_2) + k_2(\mathbf{z}_1, \mathbf{z}_2) = \left\langle \begin{pmatrix} \phi_1(\mathbf{z}_1) \\ \phi_2(\mathbf{z}_1) \end{pmatrix}, \begin{pmatrix} \phi_1(\mathbf{z}_2) \\ \phi_2(\mathbf{z}_2) \end{pmatrix} \right\rangle$$

Why Multiple Kernel Learning?



Harchaoui & Bach, 2007

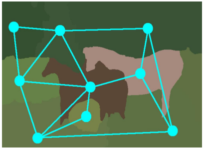


Zien & Ong, 2007

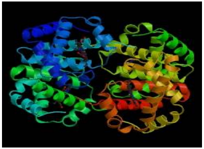
- Applications: Multi-source data fusion (web classification, genome fusion); Image annotation; Text mining; etc.
- Characteristics: Complex tasks; Heterogenous–various medias (text, images, etc.); Huge data
- Solution: Kernel methods⇒Multiple kernels learning
  - Learning combinations of kernels: $\mathcal{K} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q$, $\theta_q \geq 0$
    - Summing kernels corresponds to concatenating feature spaces
    - E.g., $k_1(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_1(\mathbf{z}_1), \phi_1(\mathbf{z}_2) \rangle$, $k_2(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_2(\mathbf{z}_1), \phi_2(\mathbf{z}_2) \rangle$

$$k_1(\mathbf{z}_1, \mathbf{z}_2) + k_2(\mathbf{z}_1, \mathbf{z}_2) = \left\langle \begin{pmatrix} \phi_1(\mathbf{z}_1) \\ \phi_2(\mathbf{z}_1) \end{pmatrix}, \begin{pmatrix} \phi_1(\mathbf{z}_2) \\ \phi_2(\mathbf{z}_2) \end{pmatrix} \right\rangle$$

Why Multiple Kernel Learning?



Harchaoui & Bach, 2007

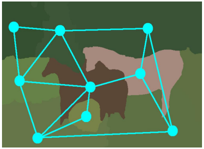


Zien & Ong, 2007

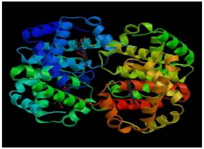
- Applications: Multi-source data fusion (web classification, genome fusion); Image annotation; Text mining; etc.
- Characteristics: Complex tasks; Heterogenous–various medias (text, images, etc.); Huge data
- Solution: Kernel methods⇒Multiple kernels learning
  - Learning combinations of kernels: $\mathcal{K} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q$, $\theta_q \geq 0$
    - Summing kernels corresponds to concatenating feature spaces
    - E.g., $k_1(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_1(\mathbf{z}_1), \phi_1(\mathbf{z}_2) \rangle$, $k_2(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_2(\mathbf{z}_1), \phi_2(\mathbf{z}_2) \rangle$

$$k_1(\mathbf{z}_1, \mathbf{z}_2) + k_2(\mathbf{z}_1, \mathbf{z}_2) = \left\langle \begin{pmatrix} \phi_1(\mathbf{z}_1) \\ \phi_2(\mathbf{z}_1) \end{pmatrix}, \begin{pmatrix} \phi_1(\mathbf{z}_2) \\ \phi_2(\mathbf{z}_2) \end{pmatrix} \right\rangle$$

Why Multiple Kernel Learning?



Harchaoui & Bach, 2007



Zien & Ong, 2007

- Applications: Multi-source data fusion (web classification, genome fusion); Image annotation; Text mining; etc.
- Characteristics: Complex tasks; Heterogenous–various medias (text, images, etc.); Huge data
- Solution: Kernel methods⇒Multiple kernels learning
  - Learning combinations of kernels: $\mathcal{K} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q$, $\theta_q \geq 0$
    - Summing kernels corresponds to concatenating feature spaces
    - E.g., $k_1(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_1(\mathbf{z}_1), \phi_1(\mathbf{z}_2) \rangle$, $k_2(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_2(\mathbf{z}_1), \phi_2(\mathbf{z}_2) \rangle$

$$k_1(\mathbf{z}_1, \mathbf{z}_2) + k_2(\mathbf{z}_1, \mathbf{z}_2) = \left\langle \begin{pmatrix} \phi_1(\mathbf{z}_1) \\ \phi_2(\mathbf{z}_1) \end{pmatrix}, \begin{pmatrix} \phi_1(\mathbf{z}_2) \\ \phi_2(\mathbf{z}_2) \end{pmatrix} \right\rangle$$

MKL–Related Work

- **Formulation:** Learning combinations of kernels

$$\boxed{\mathcal{K} = \sum_{q=1}^{Q} \theta_q \mathbf{K}_q, \quad \theta_q \geq 0}$$

  - L_1-MKL (Bach et al. 2004; Lanckriet et al. 2004, etc.): $\boxed{\|\boldsymbol{\theta}\|_1 \leq 1}$
  - L_2-MKL, L_p-MKL (Cortes et al. 2009; Kloft et al. 2010; Xu et al. 2010; etc.): $\boxed{\|\boldsymbol{\theta}\|_p \leq 1, p \neq 1}$

- **Speedup methods**
  - Semi-Definite Programming (SDP) (Lanckriet et al. 2004)
  - Second-Order Cone Programming (SOCP) (Bach et al. 2004)
  - Semi-Infinite Linear Program (SILP) (Sonnenburg et al. 2006)
  - Subgradient method (Rakotomamonjy et al. 2008)
  - Level method (Xu et al. 2009; Liu et al. 2009)

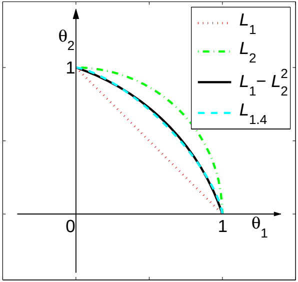
Problems and Our Contributions

- Properties and problems
  - L_1-MKL yields sparse solutions, but discard some useful information
  - L_p-MKL ($p > 1$) yields non-sparse solutions, but prone to noise
- Contributions
  - Generalize L_1-MKL and L_p-MKL
  - Theoretical analysis on the properties of grouping effect and sparsity
  - Solved by the level method

Our Generalized MKL

- Formulation

$$\min_{\boldsymbol{\theta}\in\Theta}\ \max_{\boldsymbol{\alpha}\in\mathcal{A}}\quad \mathcal{D}(\boldsymbol{\theta},\boldsymbol{\alpha}) = \mathbf{1}_N^\top\boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\left(\sum_{q=1}^{Q}\theta_q\mathbf{K}_q\right)(\boldsymbol{\alpha}\circ\mathbf{y})$$
$$\Theta = \{\boldsymbol{\theta}\in\mathbb{R}_+^Q : v\|\boldsymbol{\theta}\|_1 + (1-v)\|\boldsymbol{\theta}\|_p^p \le 1\},\ (p=2)$$
$$\mathcal{A} = \{\boldsymbol{\alpha}\in\mathbb{R}_+^N,\ \boldsymbol{\alpha}^\top\mathbf{y}=0,\ \boldsymbol{\alpha}\le C\mathbf{1}_N\}$$



Properties

$$\min_{\boldsymbol{\theta}\geq\mathbf{0}} \quad \mathcal{D}(\boldsymbol{\theta},\boldsymbol{\alpha}^\star) + \lambda\left(v\|\boldsymbol{\theta}\|_1 + (1-v)\|\boldsymbol{\theta}\|_2^2\right)$$

$$\text{where} \quad \mathcal{D}(\boldsymbol{\theta},\boldsymbol{\alpha}) = \mathbf{1}_N^\top\boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\left(\sum_{q=1}^{Q}\theta_q\mathbf{K}_q\right)(\boldsymbol{\alpha}\circ\mathbf{y})$$

- $v\|\boldsymbol{\theta}^\star\|_1 + (1-v)\|\boldsymbol{\theta}^\star\|_2^2 \Leftrightarrow 1$
- For $\mathbf{K}_i = \mathbf{K}_j$,

  $v \neq 1 \quad \theta_q^\star = \max\left\{0, \frac{1}{2(1-v)}\left(\frac{1}{2\lambda}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\mathbf{K}_q(\boldsymbol{\alpha}\circ\mathbf{y}) - v\right)\right\}$ Sparsity

  $v = 1 \quad \theta_i$ and θ_j are not unique
- $\frac{(\boldsymbol{\alpha}^\star\circ\mathbf{y})^\top\mathbf{K}_i(\boldsymbol{\alpha}^\star\circ\mathbf{y})}{(\boldsymbol{\alpha}^\star\circ\mathbf{y})^\top\mathbf{K}_j(\boldsymbol{\alpha}^\star\circ\mathbf{y})} \approx 1 \Rightarrow \theta_i^\star \approx \theta_j^\star$ Grouping effect

	L_1-MKL	L_2-MKL	GMKL	Lasso	Elastic net	Group Lasso
Sparsity	✓	×	✓	✓	✓	✓
Non-linearity	✓	✓	✓	×	×	×
Grouping	×	✓	✓	×	✓	×

Algorithm–Level Method

Given: predefined tolerant error $\delta > 0$
Initialization: Let $t = 0$ and $\boldsymbol{\theta}^0 = c\mathbf{1}_q$;
Repeat

1. Solve the $\boxed{\text{dual problem}}$ of an SVM with $\sum_{q=1}^{Q} \theta_q^t \mathbf{K}_q$ to get $\boldsymbol{\alpha}$;
2. Construct the $\boxed{\text{cutting plane}}$ model, $h^t(\boldsymbol{\theta}) = \max_{1 \le i \le t} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^i)$;
3. Calculate the $\boxed{\text{lower bound}}$ and the $\boxed{\text{upper bound}}$ of the cutting plane $\underline{\mathcal{D}}^t = \min_{\boldsymbol{\theta} \in \Theta} h^t(\boldsymbol{\theta})$, $\overline{\mathcal{D}}^t = \min_{1 \le i \le t} \mathcal{D}(\boldsymbol{\theta}^i, \boldsymbol{\alpha}^i)$ and the gap, $\Delta^t = \overline{\mathcal{D}}^t - \underline{\mathcal{D}}^t$;
4. $\boxed{\text{Project}}$ $\boldsymbol{\theta}^t$ onto the level set by solving
$$\min_{\boldsymbol{\theta} \in \Theta} \|\boldsymbol{\theta} - \boldsymbol{\theta}^t\|_2^2 \quad \text{s.t.} \quad \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha}^i) \le \underline{\mathcal{D}}^t + \tau \Delta^t, i \le t.$$
5. Update $t = t + 1$;

until $\Delta^t \le \delta$.

- Formulation:

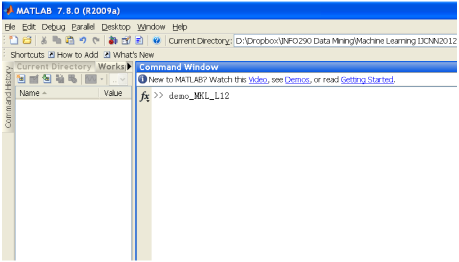
$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \mathcal{D}(\boldsymbol{\theta}, \boldsymbol{\alpha})$$
$$\Theta = \{\boldsymbol{\theta} \in \mathbb{R}_+^Q : v\|\boldsymbol{\theta}\|_1 + (1-v)\|\boldsymbol{\theta}\|_p \le 1\}$$
$$\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}_+^N,\ \boldsymbol{\alpha}^\top \mathbf{y} = 0,\ \boldsymbol{\alpha} \le C\mathbf{1}_N\}$$

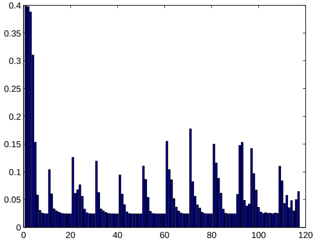
- Convergence rate

$$\boxed{\mathcal{O}(\delta^{-2})}$$

Demo

- Download codes from `http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=gmkl`
- **Note:** Required toolbox, Mosek from `http://www.mosek.com`
- In Matlab, type "demo_MKL_L12"
- See "Readme.txt" if needed





Experiments

- Datasets
  - Two toy datasets
  - Eight UCI datasets
  - Three protein subcellular localization data
- Algorithms
  - GMKL
  - L_1-norm MKL (SimpleMKL)
  - L_2-norm MKL
  - Uniformly Weighted MKL (UW-MKL)
- Platform
  - Mosek to solve the QCQP
  - Matlab
  - PC with Intel Core 2 Duo 2.13GHz CPU and 3GB memory.
- Objectives
  - Select important features in a group manner: two toy examples
  - Test efficiency: eight UCI datasets
  - Solve the proteins subcellular localization problem: three datasets

Datasets

Dataset	# Classes	# Training (N)	# Test	# Dim	# Kernel (Q)
Toy1	2	150	150	20	273
Toy2	2	150	150	20	273
Breast	2	341	342	10	143
Heart	2	135	135	13	182
Ionosphere	2	175	176	33	442
Liver	2	172	173	6	91
Pima	2	384	384	8	117
Sonar	2	104	104	60	793
Wdbc	2	284	285	30	403
Wpbc	2	99	99	33	442
Plant	4	470	470		69
Psort+	4	270	271		69
Psort-	5	722	722		69

Experimental Setup

- Preprocessing
  - Construct base kernels
  - Normalize base kernels
- Stopping criteria
  - $\#$ iterations ≤ 500, $\max|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}| \leq 0.001$
  - L_1-MKL: duality gap ≤ 0.01
  - GMKL, L_2-MKL: $\tau = 0.90$ to 0.99 when $\Delta^t/\mathcal{V}^t \leq 0.01$

Toy Data Description

- **Generation scheme**

  - **Toy 1**

    $$Y_i = \text{sign}\left(\sum_{j=1}^{3} f_1(x_{ij}) + \epsilon_i\right)$$

  - **Toy 2**

    $$Y_i = \text{sign}\left(\sum_{j=1}^{3} f_1(x_{ij}) + \sum_{j=4}^{6} f_2(x_{ij}) + \sum_{j=7}^{9} f_3(x_{ij}) + \sum_{j=10}^{12} f_4(x_{ij}) + \epsilon_i\right)$$

  $f_1(a) = -2\sin(2a) + 1 - \cos(2)$, $f_2(a) = a^2 - \frac{1}{3}$,
  $f_3(a) = a - \frac{1}{2}$, $f_4(a) = e^{-a} + e^{-1} - 1$

- **Remarks**
  - The outputs (labels) are dominated by only some features
  - Each mapping acts on three features equally, implicitly incorporating grouping effect
  - Each mapping is with zero mean on the corresponding feature, which yields zero mean on the output

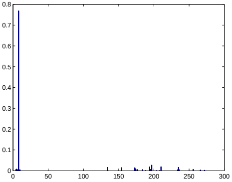
Toy Data Results

Dataset	Method	Accuracy	# Kernel	Time (s)
Toy 1	GMKL	**70.4**±3.3	36.8±5.0	2.9±0.2
	L_1-MKL	69.2±4.5	22.1±5.2	4.4±1.2
	L_2-MKL	68.2±3.0	273	2.9±0.4
	UW-MKL	66.3±5.3	273	–
Toy 2	GMKL	**72.9**±3.2	43.4±7.1	2.8±0.1
	L_1-MKL	72.3±3.1	30.2±8.1	4.9±1.3
	L_2-MKL	71.9±3.6	273	2.9±0.1
	UW-MKL	71.6±4.0	273	–

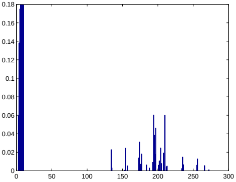
Remarks

- GMKL obtains significant improvement on the accuracy
- The non-sparse MKL models are prone to the noise
- GMKL selects more kernels, about 1.5 times of that selected by the L_1-MKL; while the L_2-MKL selects all kernels
- GMKL and L_2-MKL cost similar same, and cost less time than L_1-MKL

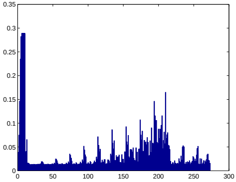
Selected Kernels on Toy Data



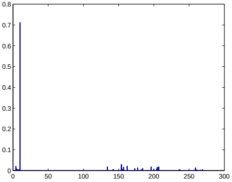
L_1-MKL on Toy 1



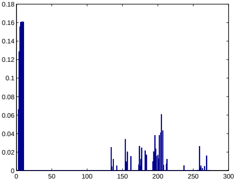
GMKL on Toy 1



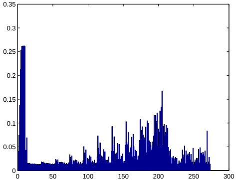
L_2-MKL on Toy 1



L_1-MKL on Toy 2

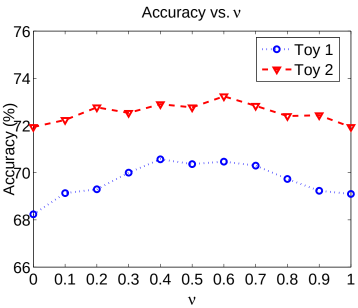


GMKL on Toy 2

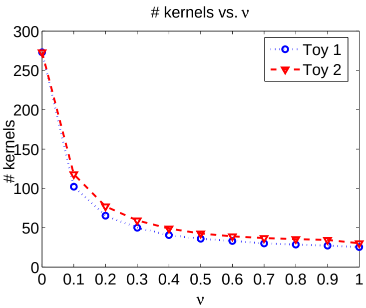


L_2-MKL on Toy 2

Effect of v on Toy Data



Accuracy vs. v



No. of selected kernels vs. v

Remarks

- $v = 0$: L_2-MKL
- $v = 1$: L_1-MKL
- The best accuracy is achieved when v is about 0.5
- The number of selected kernels decreases as v increases

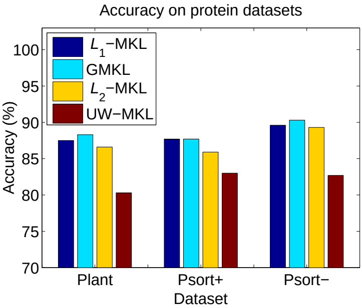
Results on UCI datasets

Dataset	Method	Accuracy	# Kernel	Time (s)	Dataset	Method	Accuracy	# Kernel	Time (s)
Breast	GMKL	**97.2**±0.5	61.1±6.5	2.8±0.5	Pima	GMKL	†**76.9**±1.6	27.1±2.4	3.8±0.2
	L_1-MKL	97.0±0.7	18.6±3.8	23.0±3.9		L_1-MKL	76.5±1.9	18.7±2.7	24.8±3.4
	L_2-MKL	96.9±0.4	143	5.1±0.3		L_2-MKL	76.0±1.8	117	6.2±1.0
	UW-MKL	**97.2**±0.5	143	–		UW-MKL	76.2±1.7	117	–
Heart	GMKL	**83.9**±1.9	38.5±5.4	1.4±0.1	Sonar	GMKL	80.4±4.1	81.1±6.5	12.4±0.6
	L_1-MKL	83.4±2.6	29.7±4.6	3.5±0.7		L_1-MKL	80.4±4.2	60.3±7.4	16.7±2.0
	L_2-MKL	82.8±2.5	182	1.7±0.1		L_2-MKL	†**83.8**±3.7	793	3.9±0.3
	UW-MKL	**83.9**±1.9	182	–		UW-MKL	81.5±4.3	793	–
Ionosphere	GMKL	91.8±1.7	66.5±7.2	5.1±0.3	Wdbc	GMKL	†**96.0**±1.1	79.7±7.6	6.6±0.8
	L_1-MKL	91.5±2.1	38.4±5.0	19.2±3.3		L_1-MKL	95.3±1.4	34.9±8.9	37.8±5.8
	L_2-MKL	**92.0**±1.8	442	4.0±0.4		L_2-MKL	95.9±0.7	403	7.8±1.6
	UW-MKL	89.9±1.8	442	–		UW-MKL	93.9±1.0	403	–
Liver	GMKL	67.6±1.8	19.5±1.7	1.0±0.0	Wpbc	GMKL	**76.7**±3.3	275.4±96.9	1.3±1.0
	L_1-MKL	64.3±2.8	9.2±3.0	1.7±0.4		L_1-MKL	76.6±2.8	40.4±10.2	4.8±1.0
	L_2-MKL	†**69.7**±2.2	91	1.4±0.0		L_2-MKL	76.3±3.7	442	1.6±0.2
	UW-MKL	67.2±4.6	91	–		UW-MKL	76.6±2.9	442	–

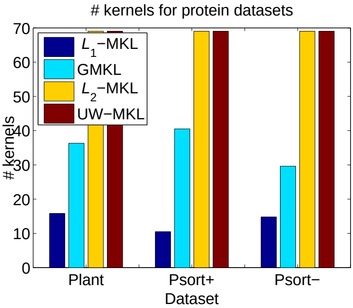
Remarks

- GMKL achieves highest accuracy on five datasets, while L_2-MKL obtains the highest accuracy for the rest three datasets
- GMKL selects more kernels, but achieves better results than L_1-MKL
- GMKL and L_2-MKL cost less time than L_1-MKL

Results on Protein Subcellular Localization Data



Accuracy

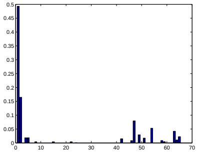


No. of selected kernels

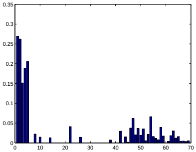
Significant test:

Dataset	GMKL vs. L_1-MKL	GMKL vs. L_2-MKL	GMKL vs. UW-MKL
Plant	0.109	0.109	0.002
Psort+	0.754	0.022	0.002
Psort-	0.022	0.002	0.002

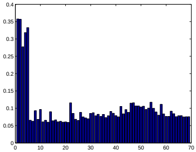
Kernel Weights on Protein Data



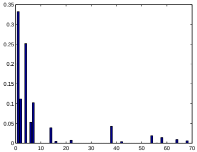
L_1-MKL on Plant



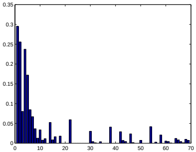
GMKL on Plant



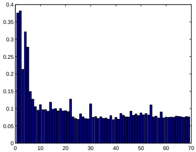
L_2-MKL on Plant



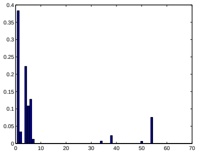
L_1-MKL on Psort+



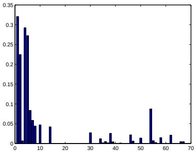
GMKL on Psort+



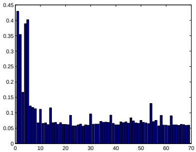
L_2-MKL on Psort+



L_1-MKL on Psort-



GMKL on Psort-



L_2-MKL on Psort-

Summary

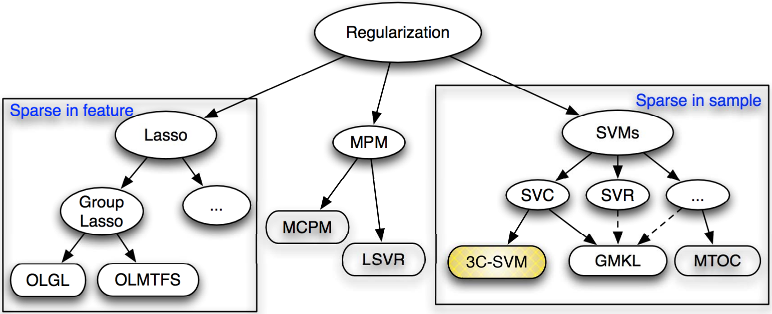
- A generalized multiple kernel learning (GMKL) model by imposing L_1-norm and L_2-norm regularization on the kernel weights
- Properties of sparsity and grouping effect are analyzed theoretically
- The model is solved by the level method and the convergence rate is provided
- Experiments on both synthetic and real-world datasets are conducted to demonstrate the effectiveness and efficiency of the model

Future work

- Apply GMKL in other applications, e.g., regression, multiclass classifications
- Apply techniques, e.g., warm start, to speed up GMKL
- Extend GMKL to include the uniformly-weighted MKL as a special case

Outline

Tri-Class Support Vector Machine



- H. Yang, S. Zhu, I. King, and M. R. Lyu. Can irrelevant data help semi-supervised learning, why and how? In *CIKM*, pages 937–946, 2011.
- Toolbox: `http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=3CSVM`

A Motivated Example–Classifying Horse and Donkey

Horse

Donkey

Relevant unlabeled

Relevant unlabeled

Irrelevant unlabeled

How to learn the decision function utilizing the labeled and (mixed) unlabeled data

A Motivated Example–Classifying Horse and Donkey

Horse

Donkey

Relevant unlabeled

Relevant unlabeled

Irrelevant unlabeled

How to learn the decision function utilizing the labeled and (mixed) unlabeled data

Why Semi-Supervised/Transductive Learning?

Labeled: Horse

Labeled: Donkey

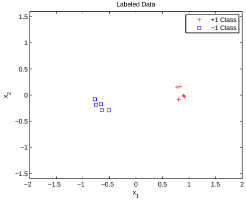
Unlabeled: Horse

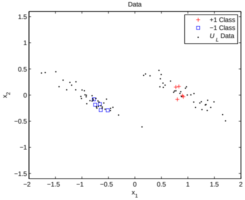
Unlabeled: Donkey

- Labeling data are precious, costly and time consuming to obtain
- Many unlabeled data are easy to collect and may provide useful information
- Close to natural human learning
  - Children master the acoustic-to-phonetic mapping of a language with few feedback
  - People recognize objects by small samples

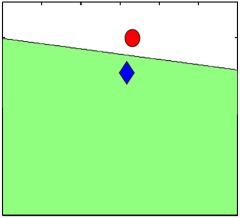
Assumptions on Semi-Supervised/Transductive Learning

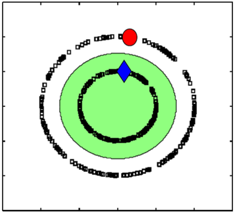
Case I: Following the same distribution





Case II: On a Riemannian manifold





Problem–Learning from Labeled and Mixed Unlabeled Data

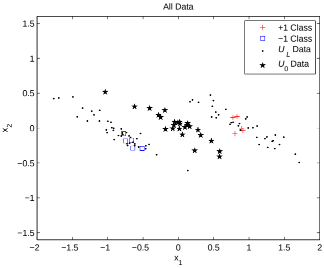
Labeled: Horse

Labeled: Donkey

Relevant unlabeled

Relevant unlabeled

Irrelevant unlabeled



How to utilize all labeled, relevant unlabeled, and irrelevant unlabeled data to improve performance in SSL?

Problem–Learning from Labeled and Mixed Unlabeled Data

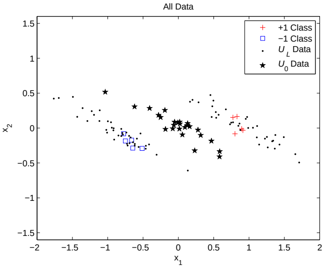
Labeled: Horse

Labeled: Donkey

Relevant unlabeled

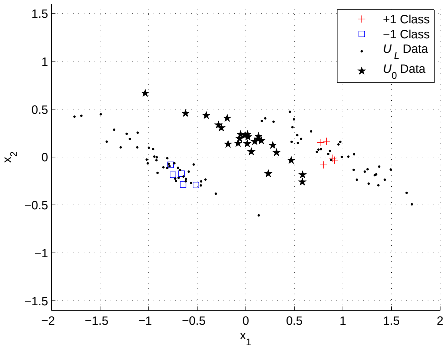
Relevant unlabeled

Irrelevant unlabeled



How to utilize all labeled, relevant unlabeled, and irrelevant unlabeled data to improve performance in SSL?

Setup of Tri-Class SVM (3C-SVM)



$$\mathcal{L} = \{(\mathbf{x}_i,\ y_i)\}_{i=1}^{L}$$
$$\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d,\ y_i \in \{-1,\ 0,\ 1\}$$
$$\mathcal{U} = \mathcal{U}_{\mathcal{L}} \cup \mathcal{U}_0 = \{\mathbf{x}_i\}_{i=1}^{U}$$

Objective: Seek

$$f_{\boldsymbol{\vartheta}}(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b,\ \boldsymbol{\vartheta} = (\mathbf{w},\ b)$$

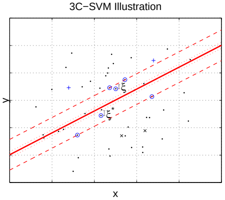
to separate the binary class data correctly with the help of (mixed) unlabeled data

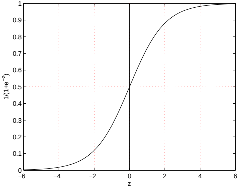
Model

- **Objective function:**

$$\min_{\vartheta} \quad \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2}_{\text{Margin}} + \underbrace{\sum_{\mathbf{x}_i \in \mathcal{L}} r_i \ell_L(f_\vartheta(\mathbf{x}_i), y_i)}_{\text{Empirical Risk Labeled Data}} + \underbrace{\sum_{\mathbf{x}_i \in \mathcal{U}} r_i \ell_U(f_\vartheta(\mathbf{x}_i))}_{\text{Empirical Risk Unlabeled Data}}$$

- **Facts:** if $f_\vartheta(\mathbf{x}_i) \gg 0$, more confident on $+1$-class
  if $f_\vartheta(\mathbf{x}_i) \ll 0$, more confident on -1-class
- **Principle:** rely more on labeled data and relevant data
  ignore irrelevant data





Model

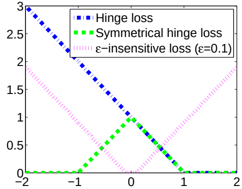
- **Objective function:**

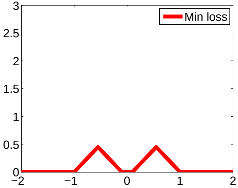
$$\min_{\boldsymbol{\vartheta}} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \overbrace{\sum_{\mathbf{x}_i \in \mathcal{L}_{\pm 1}} r_i H_1(y_i f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)) + \sum_{\mathbf{x}_i \in \mathcal{L}_0} r_i I_\varepsilon(f_{\boldsymbol{\vartheta}}(\mathbf{x}_i))}^{\text{Loss on labeled data}}$$

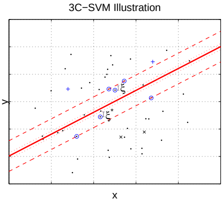
$$+ \overbrace{\sum_{\mathbf{x}_i \in \mathcal{U}} r_i \min\{H_1(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)|), I_\varepsilon(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_i)|)\}}^{\text{Loss on unlabeled data}} \, .$$

$$H_1(u) = \max\{0, 1-u\}, \quad I_\varepsilon(u) = \max\{0, |u|-\varepsilon\}$$

- **Illustration:**



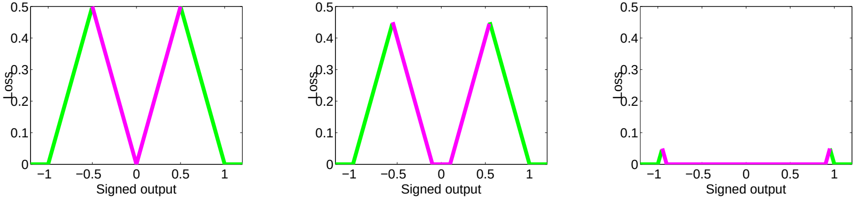




Model Generalization

- **Illustration:** $L_{\min}(u) = \min\{\max\{0, 1 - |u|\}, \max\{0, |u| - \varepsilon\}\}$

$\varepsilon = 0$ $\qquad\qquad$ $\varepsilon = 0.1$ $\qquad\qquad$ $\varepsilon = 0.9$



- **Model relationship:**

3C-SVM

$\mathcal{L}$	-1	0	1
$\mathcal{U}$	-1	0	1

SVM

$\mathcal{L}$	-1		1
$\mathcal{U}$			

S^{3}VM

$\mathcal{L}$	-1		1
$\mathcal{U}$	-1		1

$\mathcal{U}$-SVM

$\mathcal{L}$	-1	0	1
$\mathcal{U}$			

Theorem: How unlabeled irrelevant data help?

Objective function:

$$\min_{\vartheta} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{\mathbf{x}_i \in \mathcal{L}_{\pm 1}} r_i H_1(y_i f_{\vartheta}(\mathbf{x}_i)) + \sum_{\mathbf{x}_i \in \mathcal{L}_0} r_i I_{\varepsilon}(f_{\vartheta}(\mathbf{x}_i))$$
$$+ \sum_{\mathbf{x}_i \in \mathcal{U}} r_i \min\{H_1(|f_{\vartheta}(\mathbf{x}_i)|), I_{\varepsilon}(|f_{\vartheta}(\mathbf{x}_i)|)\}.$$

3C-SVM with $r_i = \infty$ for unlabeled data and $\varepsilon = 0$

Unlabeled data $\mathbf{x}_j$ satisfies
(a) $|\mathbf{w}^T\phi(\mathbf{x}_j) + b| \geq 1 \Rightarrow$ data lie on or out of the margin gap,
or
(b) $\mathbf{w}^T\phi(\mathbf{x}_j) + b = 0 \Rightarrow \mathbf{w}^T(\phi(\mathbf{x}_j) - \phi(\mathbf{x}_0)) = 0$, $\mathbf{x}_j, \mathbf{x}_0 \in \mathcal{U}_0$

Removing Min-Terms and Absolute Values

$$\min_{\vartheta} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{\mathbf{x}_i \in \mathcal{L}_{\pm 1}} r_i H_1(y_i f_{\vartheta}(\mathbf{x}_i)) + \sum_{\mathbf{x}_i \in \mathcal{L}_0} r_i I_{\varepsilon}(f_{\vartheta}(\mathbf{x}_i))$$

$$+ \sum_{\mathbf{x}_{k+L} \in \mathcal{U}} r_{k+L} \overbrace{\left(\underbrace{H_1(|f_{\vartheta}(\mathbf{x}_i)| + D(1-d_k))}_{Q_1} + \underbrace{I_{\varepsilon}(|f_{\vartheta}(\mathbf{x}_i)| - Dd_k)}_{Q_2} \right)}^{\min\{H_1(|f_{\vartheta}(\mathbf{x}_i)|), I_{\varepsilon}(|f_{\vartheta}(\mathbf{x}_i)|)\}}$$

- **Integer programming:** $\begin{cases} d_k = 0 \Rightarrow Q_1 = 0 \\ d_k = 1 \Rightarrow Q_2 = 0 \end{cases}$
- $H_1(|u| + a)$: Introducing non-convexity, solved by ramploss $H_{1-a}(u) - H_{\kappa}(u) + H_{1-a}(-u) - H_{\kappa}(-u)$
- $I_{\varepsilon}(|u| - a) = H_{-\varepsilon-a}(-u) + H_{-\varepsilon-a}(u)$
- Absolute terms are removed by introducing auxiliary labels

Concave-Convex Procedure

- **Objective function:** $Q^{\kappa}(\boldsymbol{\vartheta},\mathbf{d}){=}Q^{\kappa}_{vex}(\boldsymbol{\vartheta},\mathbf{d}){+}Q^{\kappa}_{cav}(\boldsymbol{\vartheta})$
- Each step

$$\boldsymbol{\vartheta}^{t+1} = \arg\min_{\boldsymbol{\vartheta}}\left(Q^{\kappa}_{vex}(\boldsymbol{\vartheta},\mathbf{d}^t){+}\frac{\partial Q^{\kappa}_{cav}(\boldsymbol{\vartheta}^t)}{\partial\boldsymbol{\vartheta}}\cdot\boldsymbol{\vartheta}\right),$$

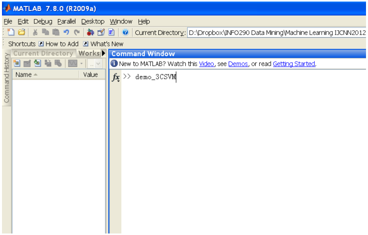
$$\overset{\text{Dual}}{\underset{\text{QP}}{\Longleftrightarrow}}\left\{\begin{array}{ll}\max\limits_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} & -\frac{\lambda}{2}\|\mathbf{w}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*)\|^2{+}\boldsymbol{\varrho}(\boldsymbol{\alpha},\boldsymbol{\alpha}^*)\\ \text{s.t.} & \mathbf{A}_e[\boldsymbol{\alpha};\boldsymbol{\alpha}^*]{=}\boldsymbol{\mu}^T\mathbf{Y}_{\bullet U}\,,\\ & \mathbf{A}[\boldsymbol{\alpha};\boldsymbol{\alpha}^*]{\le}\mathbf{0}\,,\\ & \mathbf{0}{\le}\boldsymbol{\alpha},\boldsymbol{\alpha}^*{\le}\mathbf{r}\,.\end{array}\right.$$

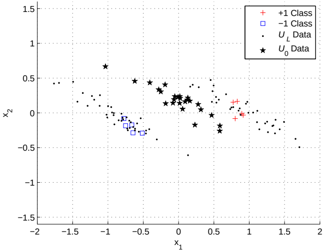
$$d_k = \begin{cases}1 & \text{if} \quad \xi_k{\le}\xi_k^*\\ 0 & \text{otherwise}\end{cases}, \quad \begin{array}{l}\xi_k{=}H_1(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L})|),\\ \xi_k^*{=}I_\varepsilon(|f_{\boldsymbol{\vartheta}}(\mathbf{x}_{k+L})|),\ k{=}1,\ldots,U.\end{array}$$

- **Solution:** w is linear combined by α and α^*
  b is attained by KKT condition

3CSVM Demo

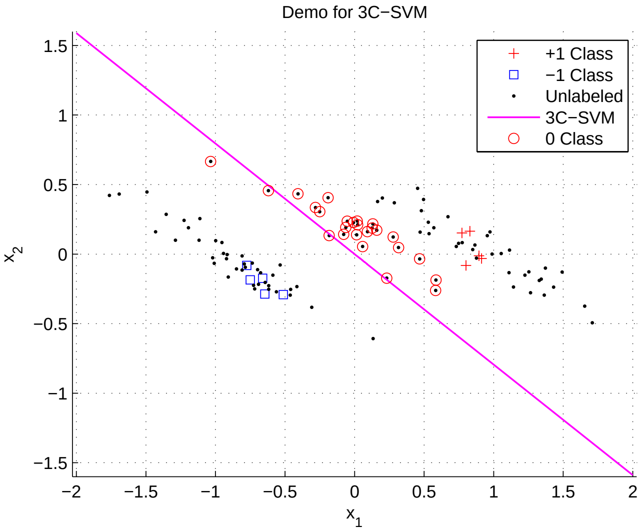
- Download codes from `http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=3csvm`
- **Note:** Required toolbox, Mosek from `http://www.mosek.com`
- In Matlab, type "demo_3CSVM"
- See "readme.txt" if needed





Video

3CSVM Result



Experimental Setup

- **Datasets**
  - Two toy datasets
  - Two real-world digit recognition datasets
- **Comparing algorithms**
  - SVMs
  - S^3VMs
  - $\mathcal{U}$-SVMs
  - 3C-SVMs
- **Platform**
  - Matlab 7.3
  - MOSEK 5.0

Data Generation

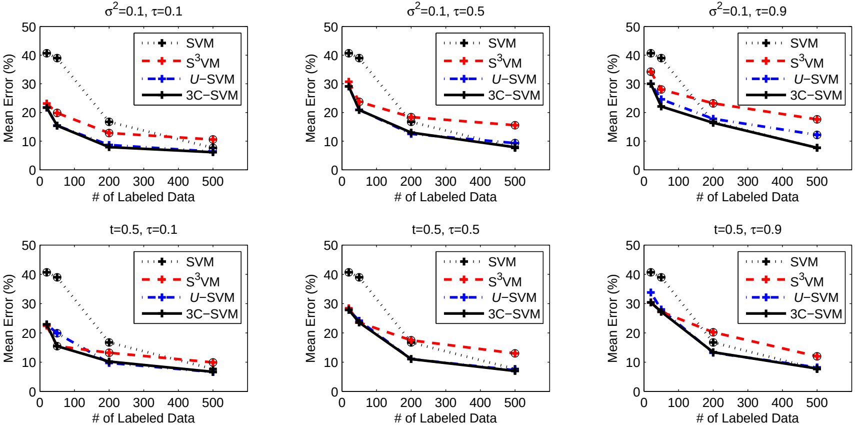
- Following scheme from Sinz et al., 2008
- $\pm$1-class: $c_i^{\pm} = \pm 0.3$, $i = 1, \ldots, 50$, $\sigma^2_{1,2} = 0.08$, $\sigma^2_{3,\ldots,50} = 10$
- Two Gaussians with the Bayes risk being approximately 5%
- First $\mathcal{U}_0$: zero mean, $\sigma^2_{1,2} = 0.1$, $\sigma^2_{3,\ldots,50} = 10$
- Second $\mathcal{U}_0$: variance values are the same as $\pm$1-class data, mean is $t \cdot \mathbf{c}^+$, $t = 0.5$

Test Procedure

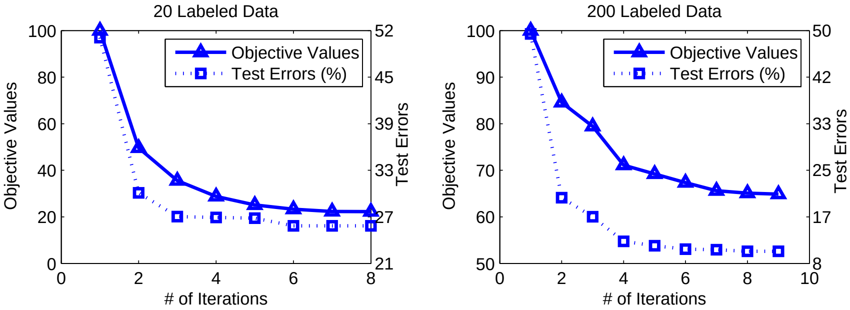
- $L = 20, 50, 200, 500$
- $U = 500 = (\tau U, (1-\tau)U)$, $\tau = 0.1, 0.5, 0.9$
- Labeled + Unlabeled/500 Test, ten-run average
- Hyperparameters
  - Linear kernel
  - Regularized parameters, forward tuning

	$C_{\mathcal{L}}$	$C_{\mathcal{U}}$	ε	κ
SVM	✓	×	×	×
$\mathcal{U}$-SVM	–	✓	✓	×
S^3VM	–	–	×	✓

Accuracy


σ²=0.1, τ=0.1
σ²=0.1, τ=0.5
σ²=0.1, τ=0.9
t=0.5, τ=0.1
t=0.5, τ=0.5
t=0.5, τ=0.9
Mean Error (%)
# of Labeled Data
SVM
S³VM
U−SVM
3C−SVM

Objective Function Values and Test Errors



Real-world Datasets

- Datasets:
  - Small size: USPS
  - Large size: MNIST
- Setup
  - ± 1-class: Digits "5" and "8"
  - $\mathcal{U}_0$: Other digits
  - $L = 20$
  - $U = 500 = (\tau U, (1-\tau)U)$, $\tau = 0.1, 0.5, 0.9$
  - RBF kernel: $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$, $\gamma = \frac{1}{0.3d}$
  - Other hyperparameters are set similar to those in the synthetic datasets

Accuracy Results

Dataset	Algorithm	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.9$
USPS	SVM	72.4± 15.9 (**0.7**)	72.4± 15.9 (**9.5**)	72.4± 15.9 (53.1)
	S^{3}VM	56.6 ± 5.9 (**0.0**)	54.5 ± 3.0 (**0.0**)	52.8 ± 6.9 (**0.0**)
	$\mathcal{U}$-SVM	83.1 ± 2.5 (**0.0**)	73.4 ± 4.4 (**0.0**)	64.2 ± 3.6 (**0.0**)
	3C-SVM	**87.2**±2.3	**80.6**±4.8	**75.4**±7.3
MNIST	SVM	70.9± 11.4 (**0.3**)	70.9± 11.4 (**0.8**)	70.9± 11.4 (13.6)
	S^{3}VM	58.9 ± 8.7 (**0.0**)	55.3 ± 8.1 (**0.0**)	53.2 ± 6.3 (**0.0**)
	$\mathcal{U}$-SVM	84.2 ± 2.2 (**0.2**)	80.0 ± 4.6 (**0.9**)	75.0 ± 3.9 (**1.0**)
	3C-SVM	**85.3**±1.6	**82.8**±2.9	**77.6**±3.9

Balance Constraint

- Ideally, $\frac{1}{U}\sum_{t=L+1}^{L+U} f_{\vartheta}(\mathbf{x}_t) = \frac{1}{L}\sum_{i=1}^{L} y_i$, but no improvement from experimental results
- A possible better on, $\frac{1}{U}\sum_{t=L+1}^{L+U} f_{\vartheta}(\mathbf{x}_t) = c$

  c: a user-specified constant, but need tuning

Summary

Summary

- A novel maxi-margin classifier, 3C-SVM, can distinguish data into -1, $+1$, and 0, three categories
- The model incorporates standard SVMs, S^{3}VMs, and $\mathcal{U}$-SVMs as specific cases
- It is solved by the CCCP, very efficient
- Effectiveness and efficiency are demonstrated

Future work

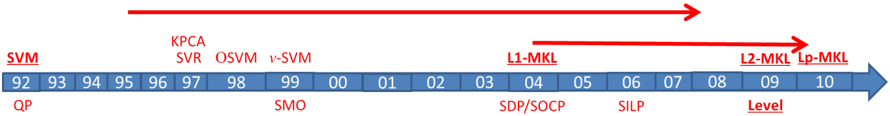
- Algorithm speedup
- Multi-class extension
- Theoretical analysis, generalization bound

Outline

Outline

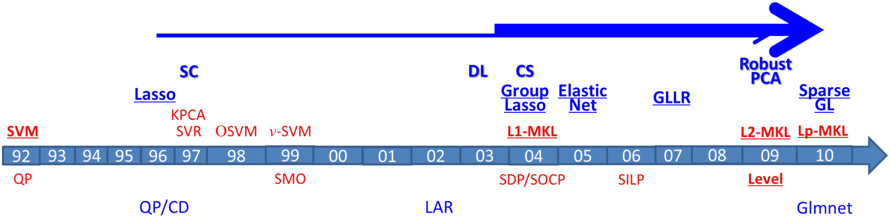
SVM and its Variants

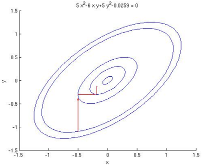
- SVM
  - In COLT'92 from VC theory
  - Many variants include SVR, ν-SVM, one-class SVM, etc.
- Kernel methods/learning
  - Kernel PCA, Kernel ICA, etc.
  - Multiple kernel learning: L_1-MKL, L_2-MKL, L_p-MKL

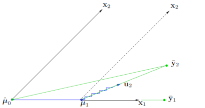


Sparse in Feature Level

- Lasso
  - Introduce in the mid of 90's
  - Many variants include Group Lasso, Elastic Net, etc.
- Sparse learning
  - Sparse coding, dictionary learning, compressive sensing, etc.



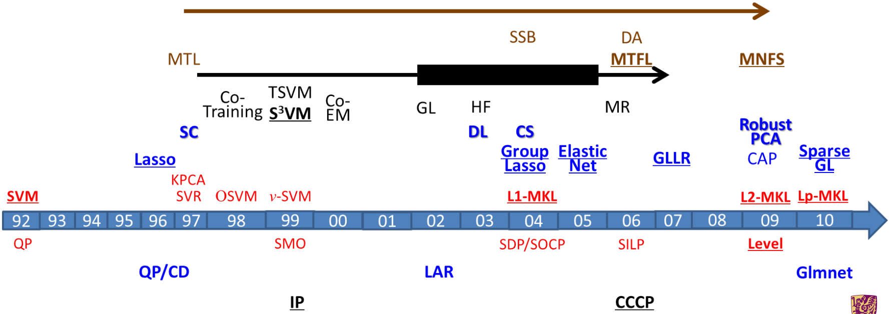




The LAR direction u_2 at step 2 makes an equal angle with x_1 and x_2.

Other Paradigms

- SSL
  - Co-training, Co-EM, tri-training, etc.
  - TSVM, S^3VM, etc.
  - Graph laplacian, harmonic function, manifold regularization, etc.
- Transfer learning
  - Multi-task learning, multi-task feature learning, mixed norm feature selection, etc.
  - Sample selection bias, domain adaptation, etc.



Outline

Perspectives

- Theory
    - Knowledge transfer
    - Concept drift
    - Sparse
    - ...
- Application-driven
    - Model interpretation
    - Scalability
    - Efficiency
    - ...

Outline

Conclusions

- Conclusions
  - Explore two families of sparse models
  - Provide promising solutions for large-scale applications in three main learning areas
    - Online learning framework for group lasso and multi-task feature selection
    - Multiple kernel learning model with sparsity and grouping effect to provide more accurate data similarity representation
    - Semi-supervised learning model to learn from mixture of relevant and irrelevant data
- Perpectives
  - Developing parsimonious learning models and efficient algorithms
  - Real-world applications with the following characteristics
    - Heterogeneous
    - Dynamic
    - Social relation or social information
    - ...

Questions?

https://www.cse.cuhk.edu.hk/irwin.king/confs/wcci2012-tutorial-machinelearning

{king,lyu,hqyang}@cse.cuhk.edu.hk

References-Our Work

- K. Huang, H. Yang, I. King, and M. R. Lyu. *Machine Learning: Modeling Data Locally and Globally*. Advanced Topics in Science and Tecnology in China: Machine Learning. Zhejiang University Press with Springer Verlag, first edition, 2008.
- Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, pages 1175–1182, 2010.
- Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*, pages 1825–1832, 2009.
- H. Yang, I. King, and M. R. Lyu. *Sparse Learning under Regularization Framework: Theory and Applications*. LAP Lambert Academic Publishing, first edition, 2011. In processing.
- H. Yang, Z. Xu, I. King, and M. R. Lyu. Online learning for group lasso. In *ICML*, pages 1191–1198, 2010.
- H. Yang, K. Huang, I. King, and M. R. Lyu. Efficient minimax clustering probability machine by generalized probability product kernel. In *WCCI2008*, 2008
- H. Yang, K. Huang, I. King, and M. R. Lyu. Localized support vector regression for time series prediction. *Neurocomputing*, 72(10-12):2659–2669, 2009.
- H. Yang, I. King, and M. R. Lyu. Multi-task learning for one-class classification. In *IJCNN*, Barcelona, Spain, 2010.
- H. Yang, I. King, and M. R. Lyu. Online learning for multi-task feature selection. In *CIKM*, pages 1693–1696, 2010.
- H. Yang, S. Zhu, I. King, and M. R. Lyu. Can irrelevant data help semi-supervised learning, why and how? In *CIKM*, pages 937–946, 2011.

References–Others' Work I

- A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *NIPS*, pages 41–48, 2006.
- F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, pages 41–48, New York, NY, USA, 2004. ACM.
- K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *NIPS*, 11:368C374, 1999.
- A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In COLT, 1998.
- B. E. Boser, I. M. Guyon, and V. Vapnik. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 144-152, 1992.
- L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, pages 161–168, 2007.
- L. Bottou and Y. LeCun. Large scale online learning. In *NIPS*, 2003.
- E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *CoRR*, abs/0912.3599, 2009.
- R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.
- R. Collobert, F. H. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 2006.
- R. Collobert, F. H. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *ICML*, pages 201–208, 2006.
- C. Cortes, M. Mohri, and A. Rostamizadeh. l_2 regularization for learning kernels. In *UAI*, 2009.
- K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.
- P. S. Dhillon, B. Tomasik, D. P. Foster, and L. H. Ungar. Multi-task feature selection using the multiple inclusion criterion (MIC). In *ECML/PKDD*, pages 276–289, 2009.
- D. L., Donoho, Compressed Sensing, IEEE Transactions on Information Theory, V. 52(4), 1289C1306, 2006
- J. Duchi and Y. Singer. Efficient learning using forward-backward splitting. *Journal of Machine Learning Research*, 10:2873–2898, 2009.
- B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

References–Others' Work II

- J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso, 2010.
- J. Friedman, T. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010.
- L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *ICML*, page 55, 2009.
- T. Joachims. Transductive inference for text classification using support vector machines. In ICML, pages 200C209. 1999.
- G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
- Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *CVPR*, 2007.
- M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien. Efficient and accurate lp-norm multiple kernel learning. In *NIPS*, pages 997–1005, 2010.
- K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. Lee, and T. J. Sejnowski. Dictionary learning algorithms for sparse representation. Neural Computation. 15(2): 349-396, 2003.
- J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.
- J. Liu, J. Chen, S. Chen, and J. Ye. Learning the optimal neighborhood kernel for classification. In *IJCAI*, pages 1144–1149, 2009.
- J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $l_{2,1}$ norm minimization. In *UAI*, 2009.
- McAuley, J., Ming, J., Stewart, D., and Hanna, P. Subband correlation and robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, 13(5-2):956–964, 2005.
- L. Meier, S. van de Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society, Series B*, 70(1):53–71, 2008.
- G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 2009.

References–Others' Work III

- J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf and C. J. C. Burges and A. J. Smola, editors, Advances in Kernel Methods–Support Vector Learning. MIT Press, 1999.
- A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:1179–1225, 2008.
- V. Roth and B. Fischer. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *ICML*, pages 848–855, 2008.
- Shalev-Shwartz, S. and Singer, Y. Online learning meets optimization in the dual. In *COLT*, pp. 423–437, 2006.
- F. H. Sinz, O. Chapelle, A. Agarwal, and B. Schölkopf. An analysis of inference with the universum. In *NIPS*, pages 1369–1376, 2008.
- S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.
- R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288, 1996.
- J. Weston, R. Collobert, F. H. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *ICML*, pages 1009–1016, 2006.
- L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, October 2010.
- V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.
- M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68(1):49–67, 2006.
- B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ICML*, 2004.
- A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *ICML*, pages 1191–1198, 2007.
- Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pp. 928–936, 2003.
- H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical, Society B*, 67:301–320, 2005.

Interpretation of Dual Average for Group Lasso

Objective: $\Upsilon(\mathbf{w}) = \min_{\mathbf{w}} \quad \sum_{i=1}^{N} \ell(\mathbf{w}) + \Omega(\mathbf{w})$

Since $\ell(\cdot)$ is convex, at T-step, we have

$$
\begin{aligned}
\Upsilon(\mathbf{w}) &= \frac{1}{T}\sum_{k=1}^{T}[\ell(\mathbf{w_k}) + \mathbf{u}_k^\top(\mathbf{w} - \mathbf{w}_k) + \underbrace{R_2(\mathbf{w})}_{\text{Second order}}] + \Omega(\mathbf{w}) \\
&= \frac{1}{T}\sum_{k=1}^{T}\ell(\mathbf{w_k}) + \bar{\mathbf{u}}_k^\top(\mathbf{w} - \mathbf{w}_k) + \underbrace{R_2(\mathbf{w})}_{\frac{\gamma}{\sqrt{t}}h(\mathbf{w})} + \Omega(\mathbf{w})
\end{aligned}
$$

Interpretation of Dual Average for MTFS

Objective: $\Upsilon(\mathbf{W}) = \min_{\mathbf{W}} \quad \sum_{i=1}^{N} \ell(\mathbf{W}) + \Omega(\mathbf{W})$

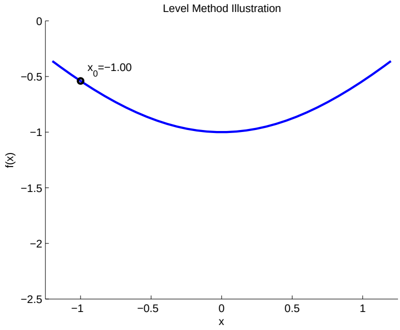
Since $\ell(\cdot)$ is convex, at T-step, we have

$$
\begin{aligned}
\Upsilon(\mathbf{W}) &= \frac{1}{T}\sum_{k=1}^{T}[\ell(\mathbf{W_k}) + \mathbf{G}_k^\top(\mathbf{W} - \mathbf{W}_k) + \underbrace{R_2(\mathbf{W})}_{\text{Second order}}] + \Omega(\mathbf{W}) \\
&= \frac{1}{T}\sum_{k=1}^{T}\ell(\mathbf{W_k}) + \bar{\mathbf{G}}_k^\top(\mathbf{W} - \mathbf{W}_k) + \underbrace{R_2(\mathbf{W})}_{\frac{\gamma}{\sqrt{t}}h(\mathbf{W})} + \Omega(\mathbf{W})
\end{aligned}
$$

Level Method Illustration

$$\min_{x} \{f(x) = -\cos(x) : x \in \mathcal{R}, \mathcal{R} = [-1.2, 1.2]\}$$

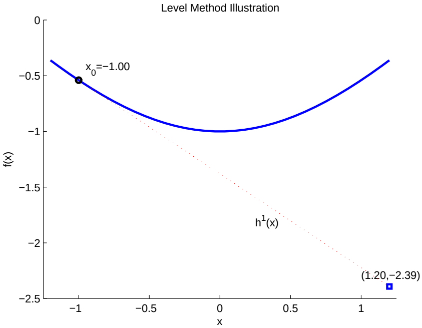
- Initialization: $x_0 = -1, \tau = 0.9$
- Construct a cutting plane
  $\mathcal{D}_1(x) = h^1(x)$
- Construct a level set level set $\mathcal{L}_1$
  $L_1 = \tau \times f(x_0) + (1-\tau) \times (-2.39)$
  $\mathcal{L}_1 = \{x \in \mathcal{R} : \mathcal{D}_1(x) \leq L_1\}$
- Project x_0 to $\mathcal{L}_1$
  $x_1 = \arg\min_{x}\{\|x - x_0\|_2^2 : x \in \mathcal{L}_1\}$



Level Method Illustration

$$\min_{x} \{f(x) = -\cos(x) : x \in \mathcal{R}, \mathcal{R} = [-1.2, 1.2]\}$$

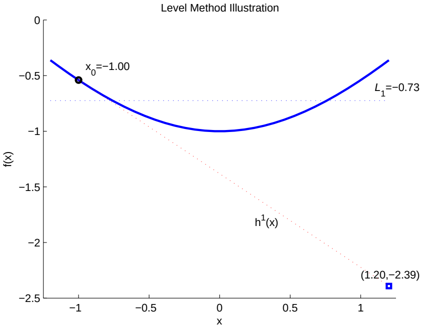
- Initialization: $x_0 = -1, \tau = 0.9$
- Construct a cutting plane
  $\mathcal{D}_1(x) = h^1(x)$
- Construct a level set level set $\mathcal{L}_1$
  $L_1 = \tau \times f(x_0) + (1-\tau) \times (-2.39)$
  $\mathcal{L}_1 = \{x \in \mathcal{R} : \mathcal{D}_1(x) \le L_1\}$
- Project x_0 to $\mathcal{L}_1$
  $x_1 = \arg\min_{x}\{\|x - x_0\|_2^2 : x \in \mathcal{L}_1\}$



Level Method Illustration

$$\min_{x} \{f(x) = -\cos(x) : x \in \mathcal{R}, \mathcal{R} = [-1.2, 1.2]\}$$

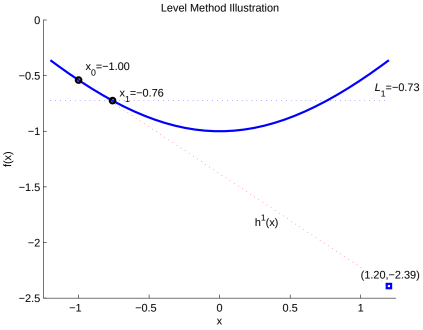
- Initialization: $x_0 = -1, \tau = 0.9$
- Construct a cutting plane
  $\mathcal{D}_1(x) = h^1(x)$
- Construct a level set level set $\mathcal{L}_1$
  $L_1 = \tau \times f(x_0) + (1-\tau) \times (-2.39)$
  $\mathcal{L}_1 = \{x \in \mathcal{R} : \mathcal{D}_1(x) \leq L_1\}$
- Project x_0 to $\mathcal{L}_1$
  $x_1 = \arg\min_{x}\{\|x - x_0\|_2^2 : x \in \mathcal{L}_1\}$



Level Method Illustration

$$\min_{x} \{f(x) = -\cos(x) : x \in \mathcal{R}, \mathcal{R} = [-1.2, 1.2]\}$$

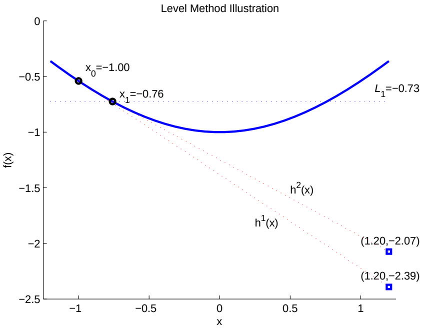
- Initialization: $x_0 = -1, \tau = 0.9$
- Construct a cutting plane
  $\mathcal{D}_1(x) = h^1(x)$
- Construct a level set level set $\mathcal{L}_1$
  $L_1 = \tau \times f(x_0) + (1-\tau) \times (-2.39)$
  $\mathcal{L}_1 = \{x \in \mathcal{R} : \mathcal{D}_1(x) \le L_1\}$
- Project x_0 to $\mathcal{L}_1$
  $x_1 = \arg\min_{x}\{\|x - x_0\|_2^2 : x \in \mathcal{L}_1\}$



Level Method Illustration

$$\min_x \{f(x) = -\cos(x) : x \in \mathcal{R}, \mathcal{R} = [-1.2, 1.2]\}$$

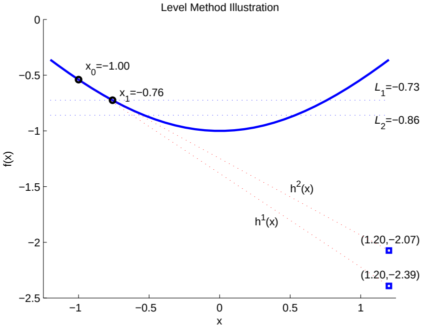
- Construct a new cutting plane $\mathcal{D}_2(x) = \min_x h^i(x)$
- Construct a new level set $\mathcal{L}_2$
  $L_2 = \tau \times f(x_1) + (1-\tau) \times (-2.07)$
  $\mathcal{L}_2 = \{x \in \mathcal{R} : \mathcal{D}_2(x) \leq L_2\}$
- Project x_1 to $\mathcal{L}_2$
  $x_2 = \arg\min_x \{\|x - x_1\|_2^2 : x \in \mathcal{L}_2\}$



Level Method Illustration

$$\min_x \{f(x) = -\cos(x) : x \in \mathcal{R}, \mathcal{R} = [-1.2, 1.2]\}$$

- Construct a new cutting plane $\mathcal{D}_2(x) = \min_x h^i(x)$
- Construct a new level set $\mathcal{L}_2$
  $L_2 = \tau \times f(x_1) + (1-\tau) \times (-2.07)$
  $\mathcal{L}_2 = \{x \in \mathcal{R} : \mathcal{D}_2(x) \leq L_2\}$
- Project x_1 to $\mathcal{L}_2$
  $x_2 = \arg\min_x \{\|x - x_1\|_2^2 : x \in \mathcal{L}_2\}$



Level Method Illustration

$$\min_x \{f(x) = -\cos(x) : x \in \mathcal{R}, \mathcal{R} = [-1.2, 1.2]\}$$

- Construct a new cutting plane $\mathcal{D}_2(x) = \min_x h^i(x)$
- Construct a new level set $\mathcal{L}_2$
  $L_2 = \tau \times f(x_1) + (1-\tau) \times (-2.07)$
  $\mathcal{L}_2 = \{x \in \mathcal{R} : \mathcal{D}_2(x) \le L_2\}$
- Project x_1 to $\mathcal{L}_2$
  $x_2 = \arg\min_x \{\|x - x_1\|_2^2 : x \in \mathcal{L}_2\}$