Coordinate Descent Algorithm for Ramp Loss Linear Programming Support Vector Machines

Xiangming Xi

Tsinghua University

July 14, 2014



Index

1 Ramp Loss Linear Programming Support Vector Machines (Ramp-LPSVM)

2 Coordinate Descent Algorithm (CDA)
- Algorithm Framework
- Implementation Issues

3 Numerical Experiments

4 Conclusion

Outline

1 Ramp Loss Linear Programming Support Vector Machines (Ramp-LPSVM)

2 Coordinate Descent Algorithm (CDA)
- Algorithm Framework
- Implementation Issues

3 Numerical Experiments

4 Conclusion

Generally, a classification problem can be expressed as follows,

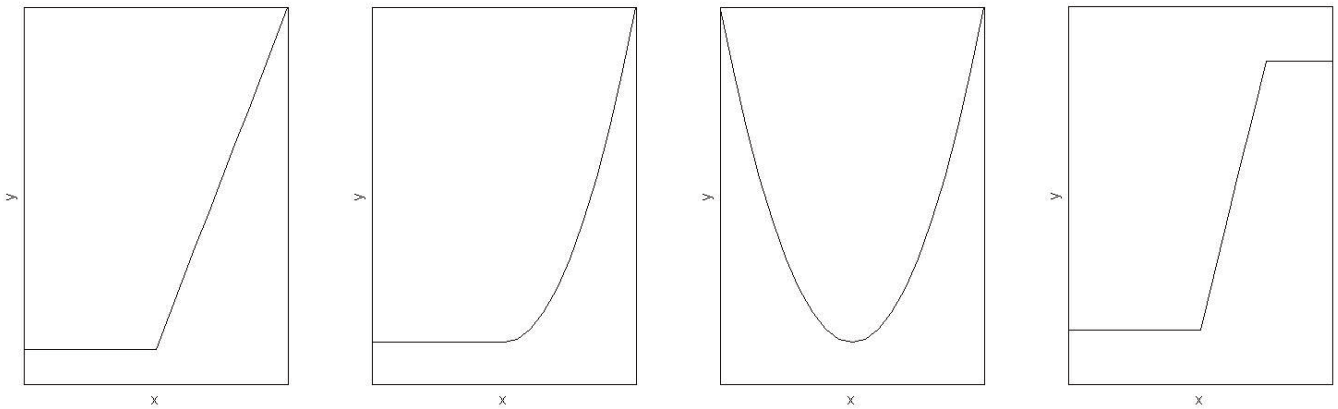
$$\min_{g,b} \quad \mu||g||_{\mathcal{K}}^2 + \frac{1}{m}\sum_{i=1}^{m} L(1 - y_i(g(x_i) + b)) \tag{1}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ belongs to the Reproducing Kernel Hilbert Space induced by the Mercer kernel $\mathcal{K}$ with norm $||\cdot||_{\mathcal{K}}$, $L(\cdot)$ represents the loss function, $b \in \mathbb{R}$ is the offset, and μ is the regularization parameter.

The regularization term: $\begin{cases} l_1\ regularization\ (1-norm); \\ l_2\ regularization\ (2-norm); \\ l_\infty\ regularization\ (\infty-norm). \end{cases}$

The regularization term:
$$\begin{cases} l_1\ regularization\ (1-norm); \\ l_2\ regularization\ (2-norm); \\ l_\infty\ regularization\ (\infty-norm). \end{cases}$$

Loss function:
$$\begin{cases} hinge\ loss\ function: \max\{u,0\}; \\ squared\ hinge\ loss\ function: \max\{u,0\}^2; \\ least\ square\ loss\ function: u^2. \\ \ldots \end{cases}$$



Ramp loss linear programming support vector machine (ramp-LPSVM):

$$\min_{\alpha \geqslant 0, b} \quad f(\alpha, b) = \mu \sum_{i=1}^{m} \alpha_i + \frac{1}{m} \sum_{i=1}^{m} L_{\mathsf{ramp}}(1 - y_i(\sum_{j=1}^{m} \alpha_j y_j \mathcal{K}(x, x_j)) + b))$$

where $L_{\mathsf{ramp}}(u) = \min\{1, \max\{u, 0\}\}$.

Ramp loss linear programming support vector machine (ramp-LPSVM):

$$\min_{\alpha \geqslant 0, b} \quad f(\alpha, b) = \mu \sum_{i=1}^{m} \alpha_i + \frac{1}{m} \sum_{i=1}^{m} L_{\mathsf{ramp}}(1 - y_i(\sum_{j=1}^{m} \alpha_j y_j \mathcal{K}(x, x_j)) + b))$$

where $L_{\mathsf{ramp}}(u) = \min\{1, \max\{u, 0\}\}$.

Advantages	**Disadvantages**
Sparsity	
Piecewise linearity	Non-differentiability
Robustness	

Outline

1 Ramp Loss Linear Programming Support Vector Machines (Ramp-LPSVM)

2 Coordinate Descent Algorithm (CDA)
- Algorithm Framework
- Implementation Issues

3 Numerical Experiments

4 Conclusion

Basic concept: Update one (block of) coordinate each time, and iterate the optimization until it converges.
Discussed problems: Convex and differentiable problems.

$$\begin{aligned}\min_{\alpha\geqslant 0,b} \quad & f(\alpha,b)=\mu\sum_{i=1}^{m}\alpha_i+\frac{1}{m}\sum_{i=1}^{m}\max\left\{1-y_i(\sum_{j=1}^{m}\alpha_j y_j\mathcal{K}(x_i,x_j)+b),0\right\}\\ & -\frac{1}{m}\sum_{i=1}^{m}\max\left\{-y_i(\sum_{j=1}^{m}\alpha_j y_j\mathcal{K}(x_i,x_j))+b),0\right\}\end{aligned}$$

$$\begin{aligned}\min_{\alpha\geqslant 0,b} \quad & f(\alpha,b)=\mu\sum_{i=1}^{m}\alpha_i+\frac{1}{m}\sum_{i=1}^{m}\max\left\{1-y_i(\sum_{j=1}^{m}\alpha_j y_j\mathcal{K}(x_i,x_j)+b),0\right\}\\ & -\frac{1}{m}\sum_{i=1}^{m}\max\left\{-y_i(\sum_{j=1}^{m}\alpha_j y_j\mathcal{K}(x_i,x_j))+b),0\right\}\end{aligned}$$

$$\begin{aligned}\min_{\alpha_j} \quad & h(\alpha_j)=\mu\alpha_j+\frac{1}{m}\sum_{i=1}^{m}\max\left\{1-y_i y_j\mathcal{K}(x,x_j)\alpha_j-q_{i,},0\right\}\\ & -\frac{1}{m}\sum_{i=1}^{m}\max\left\{-y_i y_j\mathcal{K}(x,x_j)\alpha_j-q_{i,j},0\right\}+p_j\\ \text{s.t.} \quad & \alpha_j\geqslant 0\end{aligned}$$

where $p_j=\mu\sum_{i=1,i\neq j}^{m}\alpha_i$, $q_{i,j}=y_i(\sum_{j=1,j\neq j}^{m}\alpha_j y_j\mathcal{K}(x_i,x_j)+b)$.

Framework of the Coordinate Descent Algorithm:

- Initialize ramp-LPSVM with $\boldsymbol{\alpha}_0$, $k := 0$, η and other parameters;
- REPEAT
  - FOR $j = 1$ to m
    - Solve the corresponding subproblem, with an optimal solution $\boldsymbol{\alpha}^{\star}_{k,j}$;
    - IF $h(\boldsymbol{\alpha}^{\star}_{k,j}) < h(\boldsymbol{\alpha}_{k,j})$
      Update $\boldsymbol{\alpha}_{k+1,j} := \boldsymbol{\alpha}^{\star}_{k,j}$;
  - END FOR
  - $k := k + 1$;
- UNTIL $||\boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k|| \leqslant \eta$

Implementation Issues

$$\begin{aligned} \min_{\alpha_j} \quad & h(\alpha_j) = \mu\alpha_j + \tfrac{1}{m}\sum_{i=1}^{m} \max\{1 - y_i y_j \mathcal{K}(x, x_j)\alpha_j - q_{i,j}, 0\} \\ & -\tfrac{1}{m}\sum_{i=1}^{m} \max\{-y_i y_j \mathcal{K}(x, x_j)\alpha_j - q_{i,j}, 0\} + p_j \\ \text{s.t.} \quad & \alpha_j \geqslant 0 \end{aligned}$$

$$\begin{aligned} \min_{\alpha_j} \quad & h(\alpha_j) = \mu\alpha_j + \tfrac{1}{m}\sum_{i=1}^{m} \max\{1 - y_i y_j \mathcal{K}(x, x_j)\alpha_j - q_{i,j}, 0\} \\ & -\tfrac{1}{m}\sum_{i=1}^{m} \max\{-y_i y_j \mathcal{K}(x, x_j)\alpha_j - q_{i,j}, 0\} + p_j \\ \text{s.t.} \quad & \alpha_j \geqslant 0 \end{aligned}$$

NP-hard

$$\begin{array}{ll}\min\limits_{\alpha_j} & h(\alpha_j)=\mu\alpha_j+\frac{1}{m}\sum_{i=1}^{m}\max\left\{1-y_iy_j\mathcal{K}(x,x_j)\alpha_j-q_{i,j},0\right\}\\ & -\frac{1}{m}\sum_{i=1}^{m}\max\left\{-y_iy_j\mathcal{K}(x,x_j)\alpha_j-q_{i,j},0\right\}+p_j\\ \text{s.t.} & \alpha_j\geqslant 0\end{array}$$

NP-hard

$$\begin{aligned}&\max\left\{1-y_iy_j\mathcal{K}(x,x_j)\alpha_j-q_{i,j},0\right\}-\max\left\{-y_iy_j\mathcal{K}(x,x_j)\alpha_j-q_{i,j},0\right\}\\ =\ &\begin{cases}1,\ \text{if}\ -y_iy_j\mathcal{K}(x,x_j)\alpha_j-q_{i,j}>0\\ \max\left\{1-y_iy_j\mathcal{K}(x,x_j)\alpha_j-q_{i,j},0\right\},\ \text{if}\ -y_iy_j\mathcal{K}(x,x_j)\alpha_j-q_{i,j}\leqslant 0\end{cases}\end{aligned}$$

Algorithm for subproblems

- Given $\boldsymbol{\alpha}_k$, sets $S = \{1, \cdots, m\}$.
- Randomly initialize non-empty set $S_0 \subset S$, $t := 0$;
- REPEAT
  - Solve

$$\begin{aligned} \min_{\boldsymbol{\alpha}_{k,j}} \quad & \mu\boldsymbol{\alpha}_{k,j} + \frac{1}{m}\sum_{i\in S_t} \max\{1 - y_i y_j \mathcal{K}(x, x_j)\boldsymbol{\alpha}_{k,j} - q_{i,j}, 0\} \\ & + p_j + \frac{1}{m}\sum_{i\in S\setminus S_t} 1 \\ \text{s.t.} \quad & \boldsymbol{\alpha}_{k,j} \geqslant 0 \end{aligned}$$

    with an optimum $\boldsymbol{\alpha}^\star_{k,j}$
  - Let $S_{t+1} = \{i \in S \mid -y_i y_j \mathcal{K}(x, x_j)\boldsymbol{\alpha}^\star_{k,j} - q_{i,j} \leqslant 0\}$.
  - $t := t + 1$
- UNTIL S_t equals to S_{t-1}

Outline

1 Ramp Loss Linear Programming Support Vector Machines (Ramp-LPSVM)

2 Coordinate Descent Algorithm (CDA)
- Algorithm Framework
- Implementation Issues

3 Numerical Experiments

4 Conclusion

Configuration

- Kernel function: linear, $\mathcal{K}(x, x_j) = x' * x_j$;
- To test robustness, change labels of r percent of training data;
- Compare to C-SVM which is implemented via sequential minimal optimization (SMO)

Results

Data	m	r	C-SVM Accuracy	C-SVM SV	C-SVM Time(s)	Alg. I Accuracy	Alg. I SV	Alg. I Time(s)
Spect	100	0.00	88.77%	#67	0.056	89.30%	#43	0.062
		0.05	87.70%	#66	0.070	90.91%	#57	0.049
		0.10	86.63%	#73	0.070	88.77%	#79	0.044
Brst.	100	0.00	96.67%	#42	0.076	97.06%	#76	0.098
		0.05	96.33%	#60	0.081	96.18%	#64	0.077
		0.10	95.33%	#66	0.110	96.56%	#39	0.083
Hbrm.	200	0.00	76.64%	#115	0.140	75.61%	#6	0.050
		0.05	71.03%	#125	0.056	72.90%	#6	0.050
		0.10	72.90%	#132	0.059	73.55%	#7	0.054
Monk1	200	0.00	79.63%	# 79	0.062	76.16%	#99	0.099
		0.05	76.39%	#84	0.071	75.69%	#101	0.086
		0.10	71.99%	#85	0.115	75.23%	#101	0.092
Monk2	300	0.00	87.73%	#153	0.085	82.41%	#136	0.122
		0.05	83.33%	#154	0.038	81.02%	#141	0.389
		0.10	78.94%	#151	0.079	80.32%	#143	0.599
Pima.	500	0.00	72.12%	#259	0.157	71.75%	#328	0.215
		0.05	71.00%	#75	0.108	71.38%	#71	0.130
		0.10	72.12%	#70	0.123	70.19%	#67	0.252

Results. cont

Data	m	r	C-SVM Accuracy	C-SVM SV	C-SVM Time(s)	Alg. I Accuracy	Alg. I SV	Alg. I Time(s)
Spam.	1e3	0.00	87.48%	#372	1.730	80.07%	#313	2.278
		0.05	89.73%	#489	2.050	77.82%	#373	3.188
		0.10	89.01%	#536	2.156	77.37%	#427	3.288
Cod.	5e3	0.00	89.61%	#2814	23.14	87.14%	#1728	25.62
		0.05	89.61%	#2814	23.14	87.14%	#1728	26.25
		0.10	89.61%	#2814	23.14	87.14%	#1728	25.98
Cod.	1e4	0.00	91.08%	#4374	55.44	93.75%	#7080	48.44
		0.05	90.00%	#4366	57.03	90.04%	#7080	57.12
		0.10	90.16%	#4531	55.25	89.11%	#7080	50.99
Cod.	2e4	0.00	92.60%	#15700	130.67	91.58%	#10441	141.55
		0.05	90.33%	#16210	139.11	91.58%	#11012	146.01
		0.10	90.33%	#15369	128.48	91.58%	#10310	152.30
IJCNN	4e4	0.00	89.40%	#23018	175.87	91.49%	#8441	348.49
		0.05	86.39%	#19320	167.54	89.01%	#10441	340.88
		0.10	85.77%	#21364	174.11	86.35%	#13007	354.09

Outline

1 Ramp Loss Linear Programming Support Vector Machines (Ramp-LPSVM)

2 Coordinate Descent Algorithm (CDA)
- Algorithm Framework
- Implementation Issues

3 Numerical Experiments

4 Conclusion

- The performance of the modified coordinate descent algorithm for ramp-LPSVM
- Application on large-scale SVM problems with considerable performances.

Thank you!