Computer Aided Medical Procedures

Fast Training of Support Vector Machines for Survival Analysis

Sebastian Pölsterl[1], Nassir Navab[1,2], and Amin Katouzian[1]

1: Technische Universität München, Munich, Germany

2: Johns Hopkins University, Baltimore MD, USA

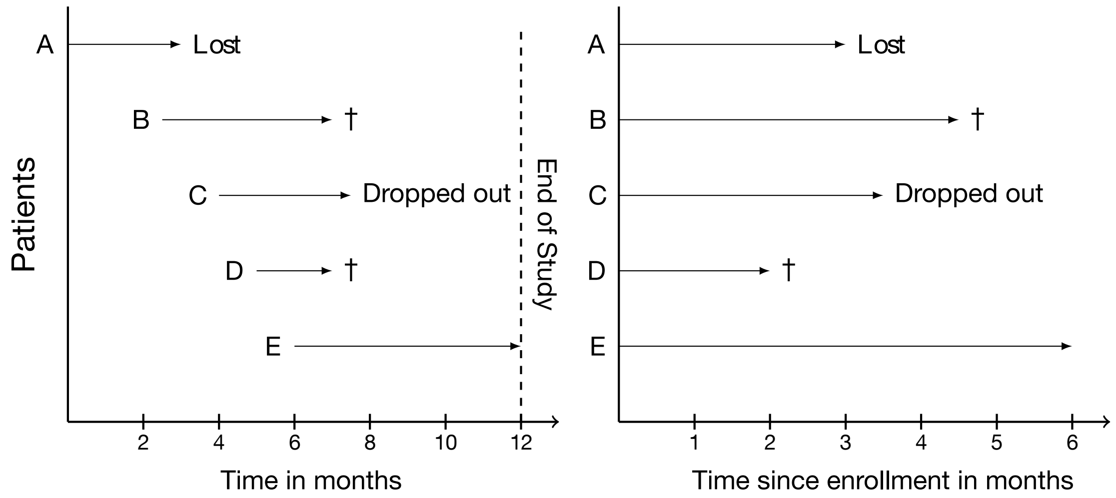
Technische Universität München

Johns Hopkins Whiting School of Engineering

Survival Analysis

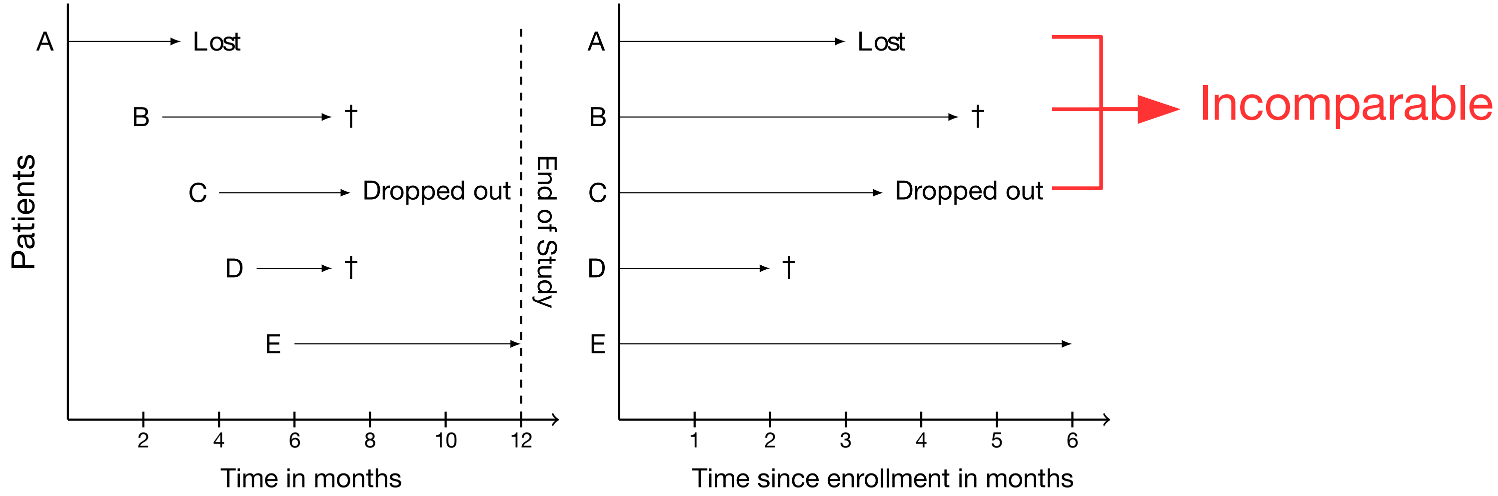
- **Objective**: to establish a connection between covariates and the time between the start of the study and an event.
- Possible formulation: Rank subjects according to observed survival time.
- Usually, parts of survival data can only be partially observed – they are **censored**.
- Survival data consists of *n* triplets:
  - $\boldsymbol{x}_i \in \mathbb{R}^d$ a *d*-dimensional feature vector
  - $y_i > 0$ time of event *or* time of censoring
  - $\delta_i \in \{0, 1\}$ event indicator

Right Censoring



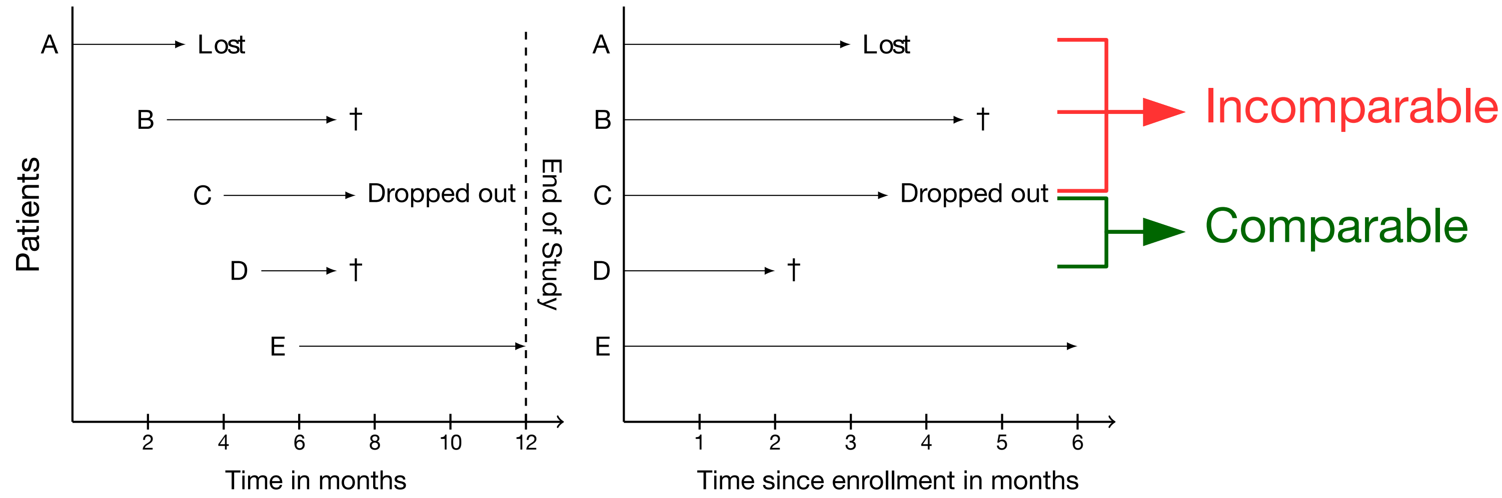
- Only events that occur while the study is running can be recorded (records are **uncensored**).
- For individuals that remained event-free during the study period, it is unknown whether an event has or has not occurred after the study ended (records are **right censored**).

Right Censoring



- Only events that occur while the study is running can be recorded (records are **uncensored**).
- For individuals that remained event-free during the study period, it is unknown whether an event has or has not occurred after the study ended (records are **right censored**).

Right Censoring



- Only events that occur while the study is running can be recorded (records are **uncensored**).
- For individuals that remained event-free during the study period, it is unknown whether an event has or has not occurred after the study ended (records are **right censored**).

Overview

- **Problem**:
  - Naive training algorithms for linear Survival Support Vector Machines require $O(n^4)$ time and $O(n^2)$ space (Van Belle et al., 2007; Evers et al., 2008).
- **Proposed Solution**:
  - Perform optimization in the primal using truncated Newton optimization.
  - Use order statistics trees to lower time and space requirements.
  - Approach extends to hybrid ranking-regression objective function as well as non-linear Survival SVM.

Survival SVM

- Objective function depends on a quadratic number of pairwise comparisons

$$\mathcal{P} = \{(i,j) \mid y_i > y_j \wedge \delta_j = 1\}_{i,j=1,\ldots,n}$$

$$f(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{\gamma}{2} \sum_{i,j \in \mathcal{P}} \max(0, 1 - \boldsymbol{w}^T(\boldsymbol{x}_i - \boldsymbol{x}_j))^2$$

- Closely related to RankSVM (Herbrich et al., 2000), where

$$\mathcal{P} = \{(i,j) \mid q_i = q_j \wedge y_i > y_j\}_{i,j=1,\ldots,n}$$

- Ties in survival time are *not* common, i.e., number of relevance levels *r* for RankSVM is *O(n)*.

Related Work – Survival SVM

- Van Belle et al., 2007: Explicitly construct all pairwise comparisons of samples to transform ranking problem into classification problem and use standard dual SVM solver.

$$O(dn^4)$$

- Van Belle et al., 2008: Reduces number of samples *n* by clustering data according to survival times using *k*-nearest neighbor search.

$$O(d\tilde{n}^4) \qquad \tilde{n} < n$$

Related Work – Rank SVM

- Airola et al., 2011: Combines cutting plane optimization with red-black tree based approach to subgradient calculations.

$$O(nd + n \log n + d + nr)$$

- Lee et al., 2014: Combines truncated Newton optimization with order statistics trees to compute gradient and Hessian.

$$O(nd + n \log n + d + n \log r)$$

The Objective Function (1)

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\sum_{i,j\in\mathcal{P}} \max(0, 1 - (\boldsymbol{w}^T\boldsymbol{x}_i - \boldsymbol{w}^T\boldsymbol{x}_j))^2 \quad (1)$$

$$= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\left(p_w + \boldsymbol{w}^T\boldsymbol{X}^T\left(\boldsymbol{A}_{\boldsymbol{w}}^T\boldsymbol{A}_{\boldsymbol{w}}\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{A}_{\boldsymbol{w}}^T\right)\right) \quad (2)$$

- $\boldsymbol{A}_{\boldsymbol{w}}$ is a $p_{\boldsymbol{w}} \times n$ sparse matrix with each row having one entry that is 1, one entry that is -1, and the remainder all zeros.
- $p_{\boldsymbol{w}}$ denotes the number of support vectors:

$$p_{\boldsymbol{w}} = |\{(i,j) \in \mathcal{P} \mid \boldsymbol{w}^T\boldsymbol{x}_j > \boldsymbol{w}^T\boldsymbol{x}_i - 1\}|$$

The Objective Function (2)

- $\boldsymbol{A_w}$ is a $p_{\boldsymbol{w}} \times n$ sparse matrix with each row having one entry that is 1, one entry that is -1, and the remainder all zeros.
- For some $s \in \{1, \ldots, n\}$, $k \in \{1, \ldots, p_w\}$ and $q \in \{1, \ldots, n\}$,

$$(\boldsymbol{A_w})_{k,q} = \begin{cases} 1 & \text{if } y_q < y_s \wedge \delta_q = 1 \wedge \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_q + 1, \\ -1 & \text{if } y_q > y_s \wedge \delta_s = 1 \wedge \boldsymbol{w}^T \boldsymbol{x}_s > \boldsymbol{w}^T \boldsymbol{x}_q - 1, \\ 0 & \text{else.} \end{cases}$$

- Example: $\mathcal{P} = \{(i,j) \mid y_i > y_j \wedge \delta_j = 1\} = \{(1,2); (1,3); (2,3); (4,2); (4,3)\}$

i	1	2	3	4
y_i	9	6	5	8
δ_i	0	1	1	0
$\boldsymbol{w}^T \boldsymbol{x}_i$	-0.7	-0.1	0.15	1.6

$$\boldsymbol{A_w} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \end{pmatrix}$$

Truncated Newton Optimization

- **Problem**: Explicitly storing the Hessian matrix can be prohibitive for high-dimensional survival data.
- **Proposed Solution**:
  - Optimization in primal.
  - Avoid constructing Hessian matrix by using truncated Newton optimization, which only requires computation of Hessian-vector product:

$$\boldsymbol{H}\boldsymbol{v} = \boldsymbol{v} + \gamma \boldsymbol{X}^T \boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{X} \boldsymbol{v}$$

Calculation of Search Direction (1)

- In each iteration of Newton's method, $\boldsymbol{A}_{\boldsymbol{w}}$ has to be recomputed due to its dependency on $\boldsymbol{w}$

$$(\boldsymbol{A}_{\boldsymbol{w}})_{k,q} = \begin{cases} 1 & \text{if } y_q < y_s \land \delta_q = 1 \land \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_q + 1 \qquad (1) \\ -1 & \text{if } y_q > y_s \land \delta_s = 1 \land \boldsymbol{w}^T \boldsymbol{x}_s > \boldsymbol{w}^T \boldsymbol{x}_q - 1 \qquad (2) \\ 0 & \text{else} \end{cases}$$

$$(\boldsymbol{A}_{\boldsymbol{w}})_{k,i} \cdot (\boldsymbol{A}_{\boldsymbol{w}})_{k,j} = \begin{cases} 1 & \text{if } i = j,\ (\boldsymbol{A}_{\boldsymbol{w}})_{k,i} = (\boldsymbol{A}_{\boldsymbol{w}})_{k,j} = 1, \\ & \quad \text{and (1) holds for } q = i, \\ 1 & \text{if } i = j,\ (\boldsymbol{A}_{\boldsymbol{w}})_{k,i} = (\boldsymbol{A}_{\boldsymbol{w}})_{k,j} = -1, \\ & \quad \text{and (2) holds for } q = i, \\ \ldots & \end{cases}$$

Calculation of Search Direction (2)

- In each iteration of Newton's method, $\boldsymbol{A}_{\boldsymbol{w}}$ has to be recomputed due to its dependency on $\boldsymbol{w}$

$$(\boldsymbol{A}_{\boldsymbol{w}})_{k,q} = \begin{cases} 1 & \text{if } y_q < y_s \wedge \delta_q = 1 \wedge \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_q + 1 \qquad (1) \\ -1 & \text{if } y_q > y_s \wedge \delta_s = 1 \wedge \boldsymbol{w}^T \boldsymbol{x}_s > \boldsymbol{w}^T \boldsymbol{x}_q - 1 \qquad (2) \\ 0 & \text{else} \end{cases}$$

$$(\boldsymbol{A}_{\boldsymbol{w}})_{k,i} \cdot (\boldsymbol{A}_{\boldsymbol{w}})_{k,j} = \begin{cases} \ldots \\ -1 & \text{if } i \neq j,\ (\boldsymbol{A}_{\boldsymbol{w}})_{k,i} = 1, (\boldsymbol{A}_{\boldsymbol{w}})_{k,j} = -1, \\ & \quad \text{and (1) holds for } q = i, s = j \\ & \quad \Leftrightarrow \text{ (2) holds for } q = j, s = i, \\ -1 & \text{if } i \neq j,\ (\boldsymbol{A}_{\boldsymbol{w}})_{k,i} = -1, (\boldsymbol{A}_{\boldsymbol{w}})_{k,j} = 1, \\ & \quad \text{and (1) holds for } q = j, s = i, \\ & \quad \Leftrightarrow \text{(2) holds for } q = i, s = j \end{cases}$$

Calculation of Search Direction (3)

- $\left(\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A}_{\boldsymbol{w}}\right)_{i,j}$ can compactly be expressed as:

$$(\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A}_{\boldsymbol{w}})_{i,j} = \begin{cases} l_i^+ + l_i^- & \text{if } i = j, \\ -1 & \text{if } i \neq j, \text{ and } j \in \mathrm{SV}_i^+ \text{ or } i \in \mathrm{SV}_j^-, \\ 0 & \text{else,} \end{cases}$$

$$\mathrm{SV}_i^+ = \{s \mid y_s > y_i \wedge \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_i + 1 \wedge \delta_i = 1\}$$

$$\mathrm{SV}_i^- = \{s \mid y_s < y_i \wedge \boldsymbol{w}^T \boldsymbol{x}_s > \boldsymbol{w}^T \boldsymbol{x}_i - 1 \wedge \delta_s = 1\}$$

$$l_i^+ = |\mathrm{SV}_i^+|$$

$$l_i^- = |\mathrm{SV}_i^-|$$

Calculation of Search Direction (4)

$$\boldsymbol{H}\boldsymbol{v} = \boldsymbol{v} + \gamma \boldsymbol{X}^T \boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{X} \boldsymbol{v}$$

$$\begin{aligned}(\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{X} \boldsymbol{v})_i &= (l_i^+ + l_i^-)\boldsymbol{x}_i^T \boldsymbol{v} - \sum_{s \in \mathrm{SV}_i^+} \boldsymbol{x}_s \boldsymbol{v} - \sum_{s \in \mathrm{SV}_i^-} \boldsymbol{x}_s \boldsymbol{v} \\ &= (l_i^+ + l_i^-)\boldsymbol{x}_i^T \boldsymbol{v} - \sigma_i^+ - \sigma_i^- .\end{aligned}$$

- Assume that l_i^+, l_i^-, σ_i^+, and σ_i^- have been computed.
- Hessian-vector product can be computed in $O(nd + d)$ instead of $O(nd + p + d)$

Order Statistics Trees

- **Problem**: Order depends on survival times and predicted scores

$$\mathrm{SV}_i^+ = \{ s \mid y_s > y_i \wedge \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_i + 1 \wedge \delta_i = 1 \}$$

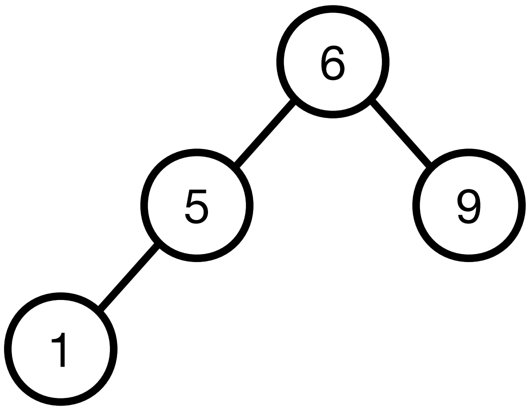
- **Solution**:
  - Sort survival data according to $\boldsymbol{w}^T \boldsymbol{x}_i$.
  - Incrementally add y_i and $\boldsymbol{w}^T \boldsymbol{x}_i$ to an order statistics tree (balanced binary search tree).

$$\begin{aligned} \mathrm{SV}_{i+1}^+ = & \{ s | \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_i + 1 \} \\ & \cup \{ s | \boldsymbol{w}^T \boldsymbol{x}_i + 1 \leq \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_{i+1} + 1 \wedge \delta_{i+1} = 1 \} \end{aligned}$$

Order Statistics Trees

$$\mathrm{SV}_i^+ = \{s \mid y_s > y_i \wedge \boldsymbol{w}^T\boldsymbol{x}_s < \boldsymbol{w}^T\boldsymbol{x}_i + 1 \wedge \delta_i = 1\}$$

i	1	2	3	4	5	6	7	8	9
$\boldsymbol{w}^T\boldsymbol{x}_i$	-0.7	-0.1	0.15	0.2	0.3	0.8	1.6	1.7	2.3
y_i	1	9	6	5	8	2	7	3	4
δ_i	0	0	1	0	1	1	1	0	0



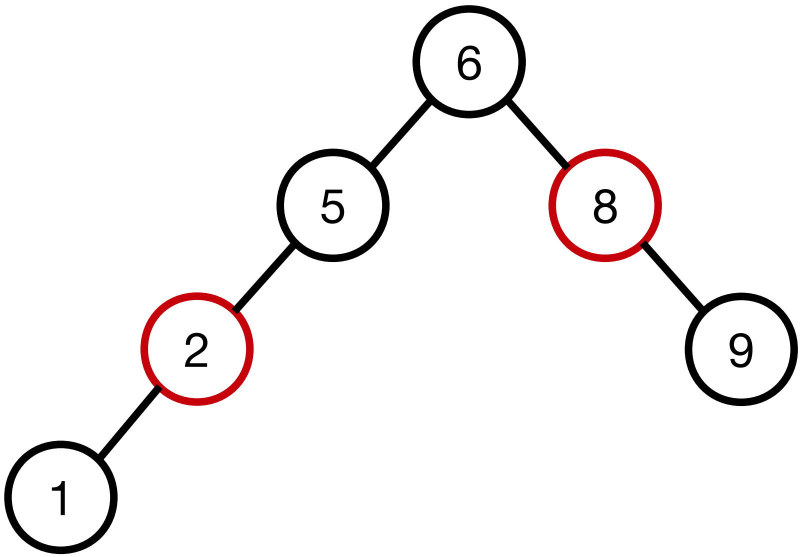
$$\mathrm{SV}_1^+ = \varnothing$$

(censored)

Order Statistics Trees

$$\mathrm{SV}_i^+ = \{s \mid y_s > y_i \wedge \boldsymbol{w}^T\boldsymbol{x}_s < \boldsymbol{w}^T\boldsymbol{x}_i + 1 \wedge \delta_i = 1\}$$

i	1	2	3	4	5	6	7	8	9
$\boldsymbol{w}^T\boldsymbol{x}_i$	-0.7	-0.1	0.15	0.2	0.3	0.8	1.6	1.7	2.3
y_i	1	9	6	5	8	2	7	3	4
δ_i	0	0	1	0	1	1	1	0	0



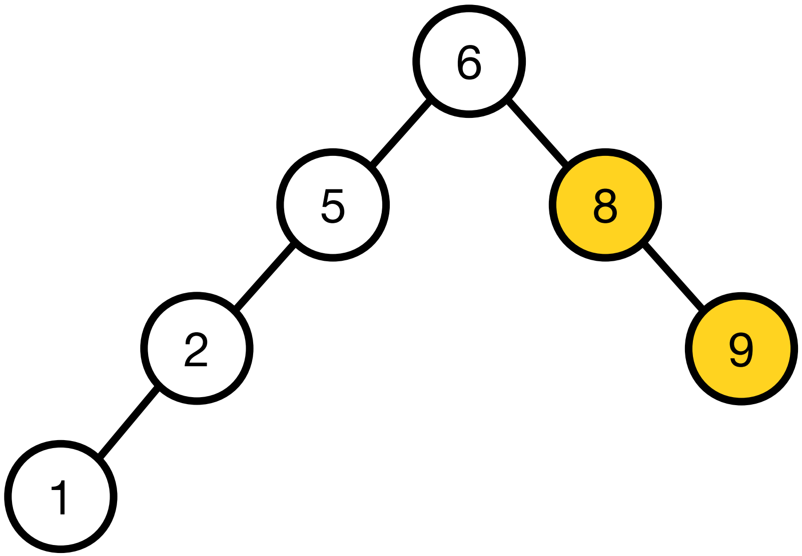
$$\mathrm{SV}_2^+ = \varnothing$$

(censored)

Order Statistics Trees

$$\mathrm{SV}_i^+ = \{s \mid y_s > y_i \wedge \boldsymbol{w}^T\boldsymbol{x}_s < \boldsymbol{w}^T\boldsymbol{x}_i + 1 \wedge \delta_i = 1\}$$

i	1	2	3	4	5	6	7	8	9
$\boldsymbol{w}^T\boldsymbol{x}_i$	-0.7	-0.1	0.15	0.2	0.3	0.8	1.6	1.7	2.3
y_i	1	9	6	5	8	2	7	3	4
δ_i	0	0	1	0	1	1	1	0	0

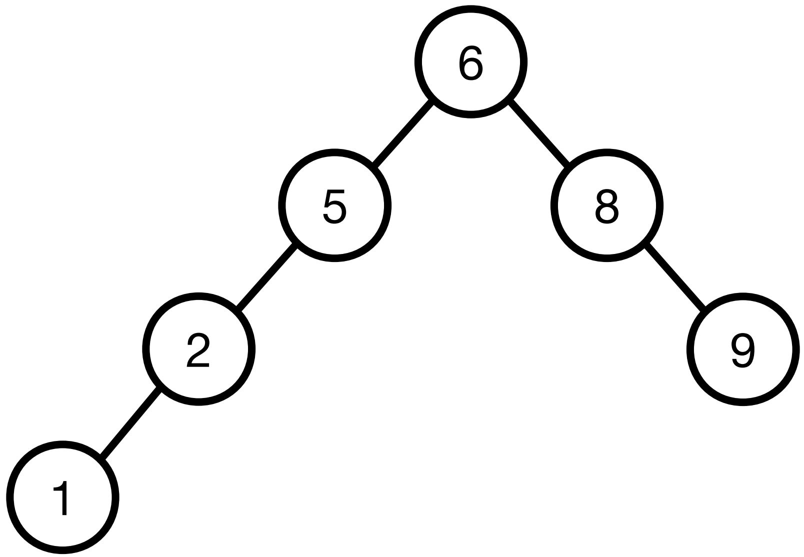


$$\mathrm{SV}_3^+ = \{2, 5\}$$

Order Statistics Trees

$$\mathrm{SV}_i^+ = \{s \mid y_s > y_i \wedge \boldsymbol{w}^T\boldsymbol{x}_s < \boldsymbol{w}^T\boldsymbol{x}_i + 1 \wedge \delta_i = 1\}$$

i	1	2	3	4	5	6	7	8	9
$\boldsymbol{w}^T\boldsymbol{x}_i$	-0.7	-0.1	0.15	0.2	0.3	0.8	1.6	1.7	2.3
y_i	1	9	6	5	8	2	7	3	4
δ_i	0	0	1	0	1	1	1	0	0



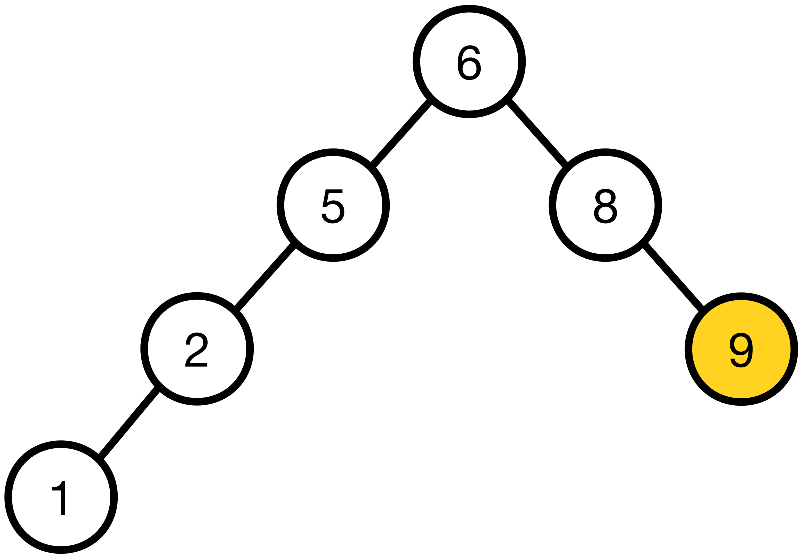
$$\mathrm{SV}_4^+ = \varnothing$$

(censored)

Order Statistics Trees

$$\mathrm{SV}_i^+ = \{ s \mid y_s > y_i \wedge \boldsymbol{w}^T \boldsymbol{x}_s < \boldsymbol{w}^T \boldsymbol{x}_i + 1 \wedge \delta_i = 1 \}$$

i	1	2	3	4	5	6	7	8	9
$\boldsymbol{w}^T \boldsymbol{x}_i$	-0.7	-0.1	0.15	0.2	0.3	0.8	1.6	1.7	2.3
y_i	1	9	6	5	8	2	7	3	4
δ_i	0	0	1	0	1	1	1	0	0



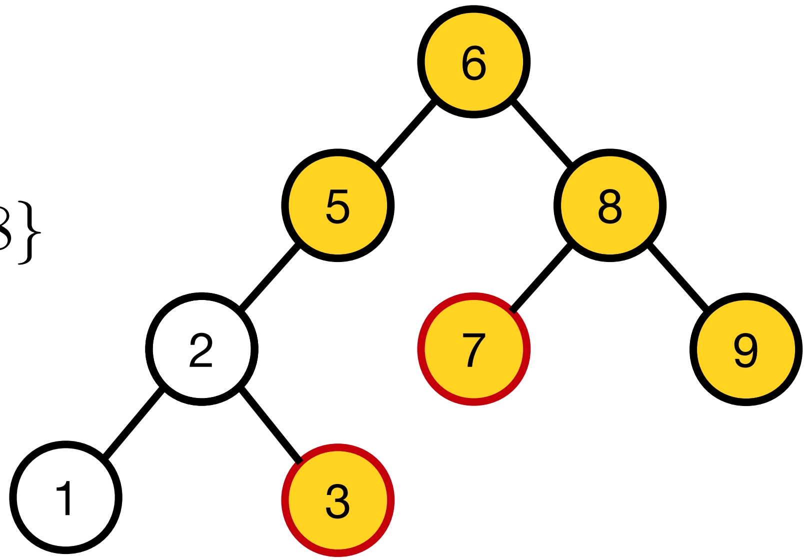
$$\mathrm{SV}_5^+ = \{2\}$$

Order Statistics Trees

$$\mathrm{SV}_i^+ = \{s \mid y_s > y_i \wedge \boldsymbol{w}^T\boldsymbol{x}_s < \boldsymbol{w}^T\boldsymbol{x}_i + 1 \wedge \delta_i = 1\}$$

i	1	2	3	4	5	6	7	8	9
$\boldsymbol{w}^T\boldsymbol{x}_i$	-0.7	-0.1	0.15	0.2	0.3	0.8	1.6	1.7	2.3
y_i	1	9	6	5	8	2	7	3	4
δ_i	0	0	1	0	1	1	1	0	0

$$\mathrm{SV}_5^+ = \{2, 3, 4, 5, 7, 8\}$$



Efficient Hessian-vector Product

- **Before**: Hessian-vector product required $O(nd + d + p)$

$$\boldsymbol{H}\boldsymbol{v} = \boldsymbol{v} + \gamma \boldsymbol{X}^T \boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{X} \boldsymbol{v}$$

$$(\boldsymbol{A}_{\boldsymbol{w}}^T \boldsymbol{A}_{\boldsymbol{w}} \boldsymbol{X} \boldsymbol{v})_i = (l_i^+ + l_i^-) \boldsymbol{x}_i^T \boldsymbol{v} - \sigma_i^+ - \sigma_i^- .$$

- **Now**: After sorting according to predicted scores, l_i^+, l_i^-, σ_i^+, and σ_i^- can be obtained in $O(\log n)$
- Hessian-vector product does not require constructing matrix of size $O(n^2)$ anymore
- Hessian-vector product requires $O(nd + d + n \log n)$

Overall Complexity

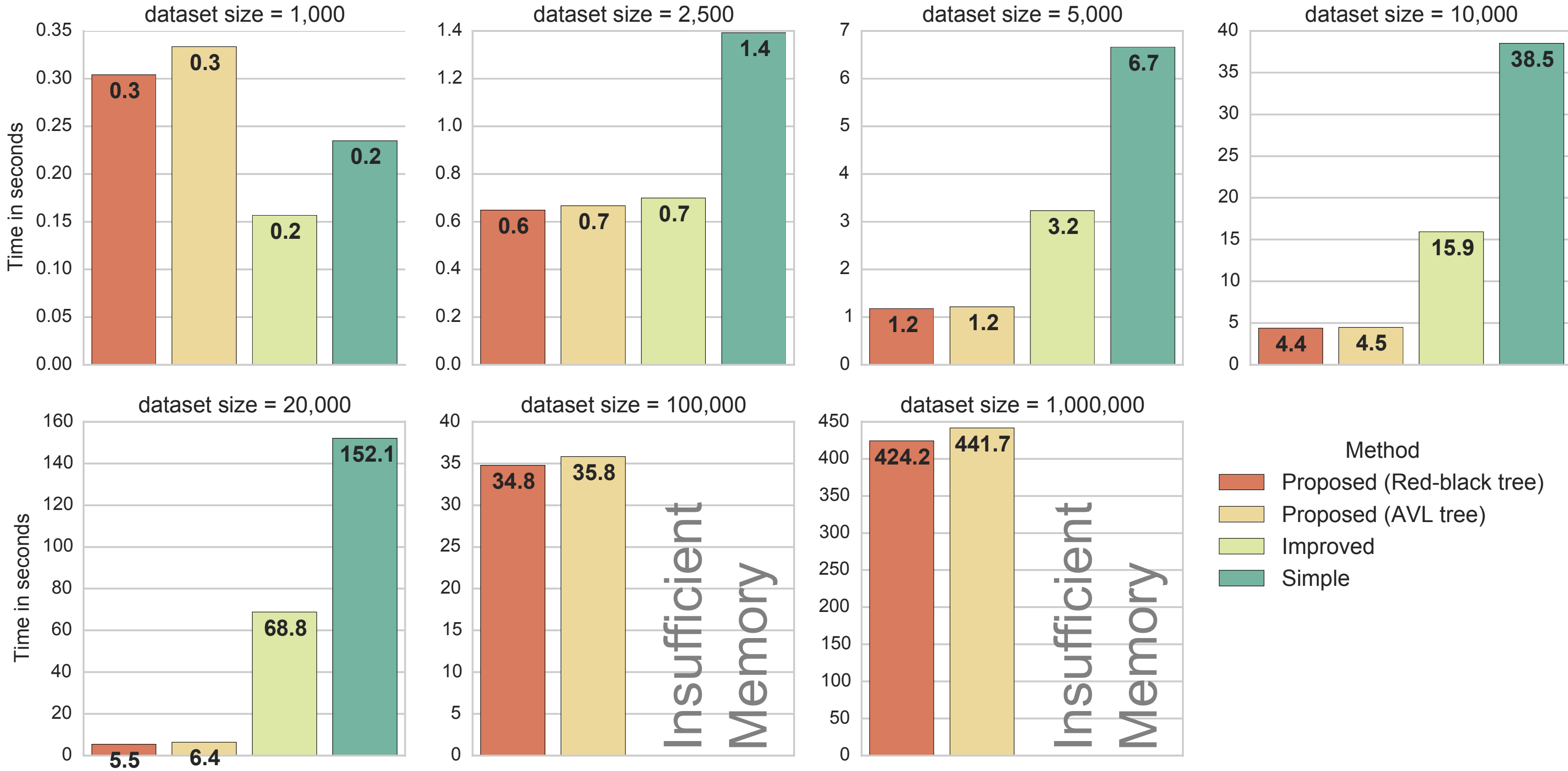
- **Time complexity**:

$$[O(n \log n) + O(nd + d + n \log n)] \cdot \bar{N}_{\mathrm{CG}} \cdot N_{\mathrm{Newton}}$$

- **Space complexity**: $O(n)$

  No need to explicitly construct all pairwise differences.

Training Time (in seconds)



Extensions

- Non-linear Survival SVM
  - Transform data with Kernel PCA before training in primal (Chapelle & Keerthi, 2009).
- Hybrid ranking-regression
  - Ranking approach cannot be used to predict exact time of event.
  - Use objective function that combines ranking and regression loss.

Conclusion

- Time complexity could be lowered from $O(dn^4)$ to $[O(n \log n) + O(nd + d + n \log n)] \cdot \bar{N}_{\text{CG}} \cdot N_{\text{Newton}}$
- Space complexity reduces from $O(n^2)$ to $O(n)$
- Same optimization scheme can be applied to non-linear Survival SVM and hybrid ranking-regression.
- Implementation is available online at https://github.com/tum-camp.

Bibliography

- Airola et al.: Training linear ranking SVMs in linearithmic time using red–black trees. Pattern Recogn. Lett. 32(9), 1328–36 (2011)
- Chapelle et al.: Efficient algorithms for ranking with SVMs. Information Retrieval 13(3), 201–5 (2009)
- Evers et al.: Sparse kernel methods for high-dimensional survival data. Bioinformatics 24(14), 1632–8 (2008)
- Herbrich et al.: Large Margin Rank Boundaries for Ordinal Regression. In: Advances in Large Margin Classifiers. pp. 115–32. (2000)
- Lee et al.: Large-Scale Linear RankSVM. Neural Comput. 26(4), 781–817 (2014)
- Van Belle et al.: Support Vector Machines for Survival Analysis. In: Proc. 3rd Int. Conf. Comput. Intell. Med. Healthc. pp. 1–8 (2007)
- Van Belle et al.: Survival SVM: a practical scalable algorithm. In: Proc. of 16th European Symposium on Artificial Neural Networks. pp. 89–94 (2008)