Parametric and Non-parametric Piecewise Linear Models and their Optimization

Xiaolin Huang

Department of Automation
School of Electronics, Information, and Electrical Engineering
Shanghai Jiao Tong University

SEIEE 2-517, Dongchuan Road 800, Shanghai, P.R. China,
xiaolinhuang@sjtu.edu.cn

2018-6-25
1. Representations for Continuous Piecewise Linear Functions
2. Parametric Models: PWL Neural Networks
3. Non-parametric Models: PWL Kernels
4. Optimization and Training for PWL Models
5. Conclusion
Outline

1. Representations for Continuous Piecewise Linear Functions
2. Parametric Models: PWL Neural Networks
3. Non-parametric Models: PWL Kernels
4. Optimization and Training for PWL Models
5. Conclusion
Continuous Piecewise Linear Function

- definition: \( f(x) : D \in \mathbb{R}^d \mapsto \mathbb{R} \) is a piecewise linear function, if there exist a finite number of affine/line functions \( p_i(x) \):

\[
    f(x) \in \{ p_1(x), p_2(x), \ldots, p_M(x) \}
\]

moreover, if \( f(x) \) is continuous, it is called continuous piecewise linear function.
Continuous Piecewise Linear Function

- definition: $f(x) : D \in \mathbb{R}^d \mapsto \mathbb{R}$ is a piecewise linear function, if there exist a finite number of affine/line functions $p_i(x)$:

$$f(x) \in \{p_1(x), p_2(x), \ldots, p_M(x)\}.$$ 

Moreover, if $f(x)$ is continuous, it is called continuous piecewise linear function.
Representations

- piecewise representation

\[ f(x) = p_i(x), \forall x \in \Omega_i. \]

with continuity condition

\[ p_i(x) = p_j(x), \forall x \in \Omega_i \cap \Omega_j, \forall i, j \]

- piecewise representations by boolean variables

- vertex representation

\[ x(j) = \sum_{k=0}^{K} d_{jk}^k \lambda_{jk}^k, \forall j \quad f(x) = \sum_{j=1}^{d} \sum_{k=0}^{K} f^j(d_{jk}^k) \lambda_{jk}^k, \]

where, \( d_{jk}^k \) are breakpoints, satisfying: \( \sum_{k=1}^{K} \lambda_{jk}^k = 1, \lambda_{jk}^k \geq 0 \)

and \( \{\lambda_{jk}^k\} \) is SOS2: special ordered set of type 2.
Compact Representations

- to represent a CPWL function as a sum/composition of basic PWL functions;

\[ f(x) = \sum_{m=1}^{M} w_m B_m(x). \]

- composition of (finite) CPWL functions are still CPWL;
- sum of (finite) CPWL functions are still CPWL.

- properties
  - capability to represent all CPWL functions;
  - capability to approach any continuous function;
  - continuity is naturally guaranteed;
  - machine learning is applicable;
  - optimized as a regular non-smooth function.
Canonical Representation and Hinging Hyperplanes

- **canonical CPWL representation**\(^1\)

\[
f(x) = a_0^\top x + b_0 + \sum_{m=1}^{M} w_m |a_m^\top x + b_m|.
\]

- **hinging hyperplanes**\(^2\)

\[
f(x) = a_0^\top x + b_0 + \sum_{m=1}^{M} w_m \max\{0, a_m^\top x + b_m\}.
\]


Learning and Optimization

- learn \(a\) and \(b\) from samples \(\{x_i, y_i\}_{i=1}^N\):

\[
\min_{a, b, w} \sum_{i=1}^N \left( y_i - \left( a_0^\top x + b_0 + \sum_{m=1}^M w_m \max\{0, a_m^\top x_i + b_m\} \right) \right)^2
\]

- the function \(f(x)\) is PWL w.r.t. \(x\) and parameters \(a, b\);
- the problem is piecewise quadratic to \(a, b\), for squared error;
- the problem is piecewise linear to \(a, b\), for absolute error.

Hinge Finding Algorithm\(^3\)

- for the \(m\)-th hinging hyperplane, select active set 
  \(\mathcal{I}_m = \{i : a_m^\top x_i + b_m > 0\}\);
- least squares on \(x_i, i \in \mathcal{I}_m\) to update \(a_m\) and \(b_m\).

---

\(^3\)Ernst, Hinging hyperplane trees for approximation and identification, IEEE-CDC, 1998.
Applications on Time-series Segmentation

- number of subregions are controlled by number of basis function

$$\min_{w,e} \frac{1}{2} \sum_{m=1}^{M} w_m^2 + \gamma \frac{1}{2} \sum_{i=1}^{N} e_i^2 + \sum_{m=1}^{M} \mu_m |w_m|$$

s.t. 
$$y(t_i) = e_i + w_0 + \sum_{m=1}^{M} w_m \phi_m(t_i), i = 1, 2, \ldots, N,$$

---

\(^4\)Huang, Matijás, Suykens, Hinging hyperplanes for time-series segmentation, IEEE-TNNLS, 2013
Limitation of Hinging Hyperplanes
Towards Full Representation Capability

- high level CPWL representation\(^5\)
- generalized hinging hyperplane\(^6\)
  \[ B_m(x) = \max \{ a_{m0}^T x + b_{m0}, a_{m1}^T x + b_{m1}, \ldots, a_{md}^T x + b_{md} \} \]
  - in deep neural networks, max pooling and maxout\(^7\) share the theoretical discussion of GHH.
- adaptive hinging hyperplanes\(^8\)
- irredundant lattice representation\(^9\)
- smoothing hinging hyperplanes\(^10\)

\(^7\) Goodfellow, Warde-Farley, Mirza, Courville, Bengio, Maxout networks, 2013.
Compact Representation for Subregion

- linear subregion, $\Omega_i$, where a CPWL function is linear:
  - $\Omega_i$ is a polyhedron;
  - $\Omega_i$ could be represented by upper/lower boundary function
    \[ \Omega_i = \{ \mathbf{x}^{(d)} \mid \mathcal{L}_i(\mathbf{x}^{(d-1)}) \leq x(1) \leq \mathcal{U}_i(\mathbf{x}^{(d-1)}) \} , \]
  - upper/lower boundaries are PWL functions in a lower space.

\[ A_1(x) - A_3(x) < 0 \quad A_3(x) - A_4(x) < 0 \quad A_2(x) - A_1(x) > 0 \quad A_1(x) - A_2(x) > 0 \quad A_4(x) - A_2(x) < 0 \quad A_2(x) - A_4(x) < 0 \]

\[ x'_2 = -0.5x'_1 \quad x'_2 = 1 - x'_1 \quad x'_2 = (1-x'_1)/3 \quad x'_2 = 0.25 - 0.5x'_1 \]

\[ x'_2 = 0 \quad x'_2 = 1 \quad x'_2 = x'_1 \]
a continuous piecewise linear function in $\mathbb{R}^d$ can be represented by the boundary functions\footnote{Wang, Huang, Junaid, Configuration of continuous piecewise linear neural networks, IEEE-TNN, 2008},

$$f(x) = \sum_{m=1}^{M} w_m \max \left\{ 0, \{ x(1) - L_m(x^{(d-1)}), U_m(x^{(d-1)}) - L_m(x^{(d-1)}) \} \right\},$$

- recursive definition leads to deep structure;
- initialization and training by back propagation.
Deep PWL Neural Network

- Linear modules in convolutional neural networks
  - Convolutional operator: \( \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} w_{ij} F_{ij} \)
  - Fully connected layer: \( \sum_i \sum_j w_{ij} F_{ij} \)
  - Averaging pooling: \( \frac{1}{K} \sum_i F_i \)

- Piecewise linear modules in convolutional neural networks
  - ReLu: \( \max\{0, u\} \)
  - LeakyReLu: \( \max\{-\tau u, u\} \)
  - Max pooling: \( \max\{F_1, F_2, \ldots, F_n\} \)
Outline

1. Representations for Continuous Piecewise Linear Functions
2. Parametric Models: PWL Neural Networks
3. Non-parametric Models: PWL Kernels
4. Optimization and Training for PWL Models
5. Conclusion
Non-parametric Models

- support vector machine learns a discriminant function from training data \( \{x_i, y_i\}_{i=1}^N \), \( x_i \in \mathbb{R}^d \), \( y_i \in \{-1, +1\} \).

\[
\min_{w, b} \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^N \max \{1 - y_i (w^T \phi(x_i) + b), 0\}.
\]

- dual problem

\[
\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i \alpha_i \mathcal{K}(x_i, x_j) \alpha_j y_j - \sum_{i=1}^N \alpha_i
\]

s.t. \( \sum_{i=1}^N y_i \alpha_i = 0 \), \( 0 \leq \alpha_i \leq C \), \( \forall i \).

- kernel trick

\[
\mathcal{K}(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]

and

\[
f(x) = w^T \phi(x) + b = \sum_{i=1}^N y_i \alpha_i \mathcal{K}(x_i, x) + b.
\]
Non-parametric Models

- sparsity and support vectors
  - only a part of samples, support vector, have $\alpha_i \neq 0$;
  - sparsity is beneficial for storage and computation;
  - if $K(x_i, x)$ is piecewise linear, then only $\alpha_i \neq 0$ provides non-convexity.
Piecewise Linear Kernels

- multiconlitron\textsuperscript{12}: a separable model;
- intersection kernel\textsuperscript{13, 14} $\mathcal{K}(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{d} \min\{\mathbf{u}(i), \mathbf{v}(i)\}$
  - additive kernel;
  - subregion structure;
- truncated $\ell_1$ kernel (TL1 kernel)\textsuperscript{15}:
  $$\mathcal{K}(\mathbf{u}, \mathbf{v}) = \max\{0, \rho - \|\mathbf{u} - \mathbf{v}\|_{\ell_1}\}$$
  - non-separable functions
  - flexible subregion structure;
  - non-PSD (positive semi-definite) kernel.

\textsuperscript{12}Li, Liu, Yang, Fu, Li, Multiconlitron: A general piecewise linear classifier, IEEE-TNN, 2011
\textsuperscript{13}Maji, Berg, Malik, Classification using intersection kernel support vector machines is efficient, CVPR, 2008
\textsuperscript{14}Maji, Berg, Malik, Efficient classification for additive kernel SVMs, IEEE-TPAMI, 2013
\textsuperscript{15}Huang, Suykens, Wang, Hornegger, Maier, Classification with truncated l1 distance kernel, IEEE-TNNLS, 2018.
Indefinite Learning

- indefinite learning

\[
\min_{\alpha_i} \quad \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i \alpha_i K(x_i, x_j) \alpha_j y_j - \sum_{i=1}^{N} \alpha_i \\
\text{s.t.} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C, \forall i.
\]

- there is no \(\phi\) such that \(K(u, v) = \phi(u)^{\top} \phi(v)\);
- the kernel matrix \(K: K_{ij} = K(x_i, x_j)\) is non-PSD and the problem is non-convex;
- non-separable PWL kernels are likely to be indefinite.
Indefinite Learning

- reproducing kernel Hilbert space (RKHS) → reproducing kernel Kreĭn spaces (RKKS)\(^{16}\)
  - feature space interpretation\(^{17}\)
  - generalized representer theorem\(^{18}\)
- convex problem → non-convex problem;
  - kernel generated model;
  - eigenvalue cutting\(^{19}\)/flipping\(^{20}\)/squaring\(^{21}\);
  - finding the nearest PSD kernel\(^{22}\), e.g., \(\min_{\tilde{\mathcal{K}} \succeq 0} \| \tilde{\mathcal{K}} - \mathcal{K} \|_F\)
  - non-convex optimization\(^{23}\).

---


\(^{17}\) Y. Ying, C. Campbell, M. Girolami, Analysis of SVM with indefinite kernels, NIPS 2009

\(^{18}\) C. S. Ong, X. Mary, S. Canu, A. J. Smola, Learning with non-positive kernels, ICML 2004

\(^{19}\) E. Pekalska, et al., Kernel discriminant analysis for PSD/indefinite kernels, TPAMI, 2009.

\(^{20}\) V. Roth, J. Laub, M. Kawanabe, J. M. Buhmann, Optimal cluster preserving embedding of nonmetric proximity data, TPAMI, 2003

\(^{21}\) H. Sun et al., LS regression with indefinite kernels and coefficient regul., ACHA, 2011

\(^{22}\) R. Luss, et al., SVM classification with indefinite kernels, NIPS 2008.

\(^{23}\) F. Schleif, P. Tino, Indefinite proximity learning: A review, Neural Computation, 2015
LS-SVM

- primal problem:

\[
\begin{align*}
\min_{w, b, \xi} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{M} \xi_i^2 \\
\text{s.t.} & \quad y_i(w^T \phi(x_i) + b) = 1 - \xi_i, \quad \forall i \in \{1, \ldots, m\}
\end{align*}
\]

- dual problem:

\[
\begin{bmatrix}
0 \\
y^T \\
y \quad H + \frac{1}{\gamma} I
\end{bmatrix} [b, \alpha_1, \ldots, \alpha_N]^T =
\begin{bmatrix}
0 \\
1
\end{bmatrix},
\]

where \( I \) is an identity matrix, \( 1 \) is an all ones vector with the proper dimension, and \( H \) is given by

\[
H_{ij} = y_iy_j K_{ij} = y_iy_j K(x_i, x_j).
\]
choose a non-PSD kernel $K$, the dual problem of LS-SVM is still easy to solve, but it lacks of feature space interpretation.\(^{24}\)

**Theorem**

The dual problem of

\[
\begin{align*}
\min & \quad \frac{1}{2} (\mathbf{w}_+^T \mathbf{w}_+ + \mathbf{w}_-^T \mathbf{w}_-) + \frac{\gamma}{2} \sum_{i=1}^{N} \xi_i^2 \\
\text{s.t.} & \quad y_i (\mathbf{w}_+^T \phi_+ (\mathbf{x}_i) + \mathbf{w}_-^T \phi_- (\mathbf{x}_i) + b) = 1 - \xi_i, \quad \forall i \in \{1, 2, \ldots, N\}
\end{align*}
\]

is

\[
\begin{bmatrix}
0 \\ \mathbf{y}^T \\
\mathbf{y} \\
\mathbf{H} + \frac{1}{\gamma} \mathbf{I}
\end{bmatrix}
\begin{bmatrix}
b \\ \alpha_1, \ldots, \alpha_M
\end{bmatrix}^T =
\begin{bmatrix}
0 \\ 1
\end{bmatrix}.
\]

\(^{24}\) Huang, Maier, Hornegger, Suykens, Indefinite kernels in least squares support vector machine and principal component analysis, Applied and Comput. Harmonic Analysis, 2017
Learning Performance of PWL Kernel

Table: Average Accuracy and Standard Deviation on Test Data

<table>
<thead>
<tr>
<th>dataset</th>
<th>$M$</th>
<th>RBF kernel ($\sigma$ by cross-validation)</th>
<th>TL1 kernel ($\rho = 0.7n$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qsar</td>
<td>528</td>
<td>86.92 ± 1.31%</td>
<td>86.05 ± 1.21%</td>
</tr>
<tr>
<td>Splice</td>
<td>1000</td>
<td>89.83 ± 0.09%</td>
<td>92.74 ± 0.02%</td>
</tr>
<tr>
<td>Guide3</td>
<td>1243</td>
<td>84.15 ± 3.45%</td>
<td>97.56 ± 0.00%</td>
</tr>
<tr>
<td>Madelon</td>
<td>2000</td>
<td>58.83 ± 0.00%</td>
<td>61.33 ± 0.00%</td>
</tr>
<tr>
<td>Spamb.</td>
<td>2300</td>
<td>93.32 ± 0.60%</td>
<td>94.05 ± 0.56%</td>
</tr>
<tr>
<td>ML-prove</td>
<td>3059</td>
<td>72.48 ± 0.32%</td>
<td>79.08 ± 0.00%</td>
</tr>
<tr>
<td>Guide1</td>
<td>3089</td>
<td>96.84 ± 0.16%</td>
<td>97.12 ± 0.04%</td>
</tr>
<tr>
<td>Wilt</td>
<td>4339</td>
<td>85.80 ± 0.74%</td>
<td>86.80 ± 0.44%</td>
</tr>
<tr>
<td>Phish.</td>
<td>5528</td>
<td>95.92 ± 0.30%</td>
<td>93.83 ± 0.48%</td>
</tr>
<tr>
<td>Magic</td>
<td>9510</td>
<td>86.48 ± 0.45%</td>
<td>86.04 ± 0.43%</td>
</tr>
<tr>
<td>RNA</td>
<td>59535</td>
<td>96.66 ± 0.20%</td>
<td>95.74 ± 0.22%</td>
</tr>
</tbody>
</table>
Indefinite kernel PCA

- primal problem:
  \[
  \max_{\mathbf{w}, \xi} \quad \frac{\gamma}{2} \sum_{i=1}^{M} \xi_i^2 - \frac{1}{2} \mathbf{w}^T \mathbf{w} \\
  \text{s.t.} \quad \xi_i = \mathbf{w}^T (\phi(x_i) - \hat{\mu}_\phi), \forall i \in \{1, \ldots, M\},
  \]
  where \( \hat{\mu}_\phi \) is the centering term, i.e., \( \hat{\mu}_\phi = \frac{1}{m} \sum_{i=1}^{M} \phi(x_i) \).

- dual problem:
  \[
  \Omega \alpha = \lambda \alpha,
  \]
  where the centered kernel matrix \( \Omega \) is induced from \( \mathcal{K} \):
  \[
  \Omega_{ij} = \mathcal{K}(x_i, x_j) - \frac{1}{M} \sum_{r=1}^{M} \mathcal{K}(x_i, x_r) \\
  - \frac{1}{M} \sum_{r=1}^{M} \mathcal{K}(x_j, x_r) + \frac{1}{M^2} \sum_{r=1}^{M} \sum_{s=1}^{M} \mathcal{K}(x_r, x_s).
  \]

- a non-PSD kernel can also be directly used.
Indefinite kernel PCA with PWL kernel

**Figure:** Reduce data of two classes in three dimensional space into two dimensional space
Outline

1. Representations for Continuous Piecewise Linear Functions
2. Parametric Models: PWL Neural Networks
3. Non-parametric Models: PWL Kernels
4. Optimization and Training for PWL Models
5. Conclusion
Example 1: Chiller Plants Optimization

- operation optimization for centrifugal chiller plants
Example 1: Chiller Plants Optimization

- model the input-output relationship by PWL functions;
- surrogate optimization via sub linear programings;

\[
\begin{align*}
\min_{\alpha} & \quad f_0(\alpha) \\
\text{s.t.} & \quad f_i(\alpha) \leq 0 \\
& \quad h_i(\alpha) = 0.
\end{align*}
\]

<table>
<thead>
<tr>
<th>Temperature</th>
<th>600 kW</th>
<th>1500 kW</th>
<th>2400 kW</th>
<th>3300 kW</th>
</tr>
</thead>
<tbody>
<tr>
<td>10°C</td>
<td>37.74%</td>
<td>30.69%</td>
<td>12.91%</td>
<td>19.85%</td>
</tr>
<tr>
<td>15°C</td>
<td>14.05%</td>
<td>17.28%</td>
<td>17.68%</td>
<td>09.79%</td>
</tr>
<tr>
<td>20°C</td>
<td>25.30%</td>
<td>02.11%</td>
<td>06.13%</td>
<td>16.92%</td>
</tr>
<tr>
<td>25°C</td>
<td>24.51%</td>
<td>10.87%</td>
<td>10.04%</td>
<td>09.99%</td>
</tr>
</tbody>
</table>
Example 2: PVC Production Process Optimization

- PVC production process

![Diagram of PVC production process]
Example 2: PVC Production Process Optimization

- equations: 13871
- optimization variables: 5064 (discrete), 8119 (continuous)
- optimization time (MILP): around 2000 s (MINLP: around 14000 s)

<table>
<thead>
<tr>
<th></th>
<th>Optimization Model</th>
<th>Current Model</th>
<th>improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>absolute value ($)</td>
</tr>
<tr>
<td>total cost</td>
<td>156,796,000</td>
<td>166,392,500</td>
<td>9,596,500</td>
</tr>
<tr>
<td>energy cost</td>
<td>68,424,300</td>
<td>79,406,633.5</td>
<td>10,982,334</td>
</tr>
<tr>
<td>coal cost</td>
<td>68,424,300</td>
<td>75,949,933.5</td>
<td>7,525,634</td>
</tr>
<tr>
<td>inventory cost</td>
<td>1,602,600</td>
<td>229,380</td>
<td>1,373,220</td>
</tr>
<tr>
<td>material cost</td>
<td>86,712,400</td>
<td>86,712,386.5</td>
<td>14</td>
</tr>
<tr>
<td>switching cost</td>
<td>56,700</td>
<td>44,100</td>
<td>12,600</td>
</tr>
</tbody>
</table>
PWL Optimization Problems

- optimization based on learned PWL models;
  - for unknown or complicated function \( g(x) \), model a surrogate PWL function \( f(x) \) and optimize \( f(x) \);
  - discussion on specific PWL model, e.g., local optimality\(^{25}\) and global heuristic;

- training for PWL models
  - piecewise linear penalty/loss → piecewise linear optimization
    - \( \ell_1 \)-norm regularization term, total variation, non-convex sparsity enhancer\(^{26,27}\)
    \[ p(u) = \sum_{k=1}^{K} |u[k]| \]
    - absolute loss, quantile loss (k-th maximum loss), hinge loss, ramp loss, ...
  - smooth penalties/loss → piecewise smooth optimization
    - \( \ell_2 \)-norm regularization term
    - squared loss, sigmoid loss, logarithmic loss, ...

\(^{26}\) Wang, Yin, Sparse signal recon. via iterative support detection, SIAM. Imag. Sci. 2010
\(^{27}\) Huang, Van Huffel, Suykens, Two-level \( \ell_1 \) minimization for CS., Signal Processing, 2015
Example 3: Ramp-LPSVM

- $\ell_1$-norm penalty (sparsity) + ramp loss (robustness)$^{28}$;
- $p(u) = \sum_{i=1}^{n} |u(i)|$, $l_{\text{ramp}}(u) = \max\{0, \min\{u, 1\}\}$;
- ramp loss linear programming SVM

$$
\min_{\alpha} \mu \sum_{i} |\alpha_i| + \frac{1}{N} \sum_{i=1}^{N} l_{\text{ramp}} \left( 1 - y_i \left( \sum_{j=1}^{N} \alpha_i K(x_i, x_j) \right) \right)
$$

- difference of convex functions:

$$
\min_{\alpha} \mu \sum_{i} |\alpha_i| + \frac{1}{N} \sum_{i=1}^{N} \max \left\{ 1 - y_i \left( \sum_{j=1}^{N} \alpha_i K(x_i, x_j) \right), 0 \right\} - \frac{1}{N} \sum_{i=1}^{N} \max \left\{ -y_i \left( \sum_{j=1}^{N} \alpha_i K(x_i, x_j) \right), 0 \right\}
$$

$^{28}$Huang, Shi, Suykens, Ramp loss linear programming support vector machine, JMLR, 2014.
Example 3: Ramp-LPSVM

- solving ramp-LPSVM
  - linear programming for a local optimum;
  - hill detouring\(^{29}\), i.e., search on contour lines of a concave PWL function;

Example 3: Ramp-LPSVM

Table: Accuracy on Test Data and Number of SV (10% outliers)

<table>
<thead>
<tr>
<th></th>
<th>Spect</th>
<th>Monk1</th>
<th>Monk2</th>
<th>Monk3</th>
<th>Breast</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-SVM</td>
<td>81.42%</td>
<td>76.22%</td>
<td>72.41%</td>
<td>80.05%</td>
<td>89.69%</td>
</tr>
<tr>
<td>ramp-LPSVM</td>
<td>87.88%</td>
<td>79.33%</td>
<td>81.57%</td>
<td>83.43%</td>
<td>93.35%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Pima</th>
<th>Trans.</th>
<th>Haber.</th>
<th>Ionos.</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-SVM</td>
<td>61.66%</td>
<td>70.33%</td>
<td>70.65%</td>
<td>85.79%</td>
</tr>
<tr>
<td>ramp-LPSVM</td>
<td>68.51%</td>
<td>75.28%</td>
<td>74.62%</td>
<td>90.35%</td>
</tr>
</tbody>
</table>

- robustness to outliers is improved;
- sparsity is enhanced;
- algorithm is not applicable to large-scale problems.
Outline

1. Representations for Continuous Piecewise Linear Functions
2. Parametric Models: PWL Neural Networks
3. Non-parametric Models: PWL Kernels
4. Optimization and Training for PWL Models
5. Conclusion
Conclusion

- compact continuous piecewise linear models
  - parametric models and its link to neural networks: DP-CPLNN, AHH, SHH, ...
  - non-parametric models and indefinite learning: TL1 kernel, indefinite LS-SVM, and indefinite kPCA, ...
- optimization based on compact piecewise linear models
  - surrogate optimization for chiller plants and PVC production process
  - machine learning based on piecewise linear models, e.g., ramp-LPSVM

Outlook

- learning behavior and interpretation
  - deep piecewise linear neural networks
  - piecewise linear indefinite kernels
- piecewise linear optimization
  - fast local search and efficient global search
  - training piecewise linear neural networks
- conversation among different piecewise linear models
Acknowledgement

- Prof. Shuning Wang,
  Tsinghua University, Beijing, P.R. China
- Prof. Johan A.K. Suykens,
  Katholieke Universiteit Leuven, Leuven, Belgium
- Prof. Andreas Maier,
  Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
- Prof. Jun Xu,
  Harbin Institute of Technology (Shenzhen Campus), Shenzhen, P.R. China
Thanks