Smart Nord

Support Vector Decoder for Constraint-Handling

in Distributed Real Power Planning — Jörg Bremer

About me

- Home
  - Department for Computing Science, Environmental Informatics
- Research interest
  - Computational Intelligence
  - Agent-based systems
  - Smart Grid
- Current Project
  - Smart Nord
  - TP1: Decentralized co-ordination of active power provision

Background

The idea of a Smart Grid

The idea of a Smart Grid

- Connect the electricity grid and ICT, to . . .
  - . . . build an intelligent energy network with interacting intelligent generators, storages, loads and transportation equipment
  - . . . integrate new distributed resources ad hoc into control flow
- Enable self-x properties by agent-based control
  - Self-organization of small units to jointly gain enough power
    - Role of power plants
  - Adopts flexibly to new developments
  - Scales well with the expected huge number of controllable energy entities

Lots of interesting computational problems, but. . .

Agenda

Motivation

Introduction

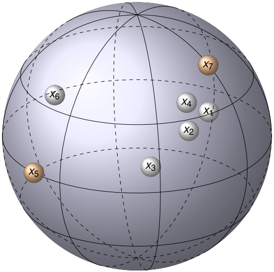
- Distributed Real Power Planning
- Basic Solution Idea

Solution

- Constraints
- SVDD
- Decoder
- Optimization

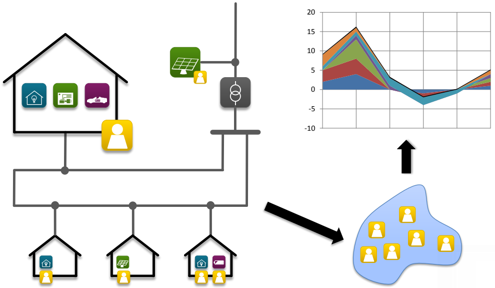
Evaluation

Conclusion

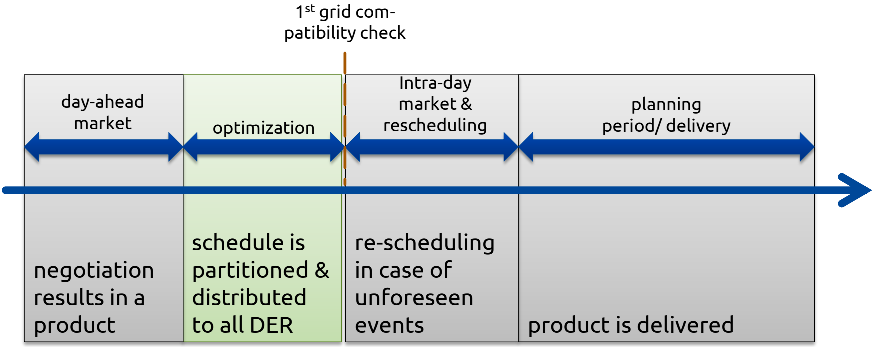


Use case

- Starting point: We already have a coalition with controllable energy resources
- A coalition of energy units wants to jointly operate a given active power schedule!
  - Question: How can we coordinate the group to achive this?



Timeline


1st grid compatibility check
day-ahead market
optimization
Intra-day market & rescheduling
planning period/ delivery
negotiation results in a product
schedule is partitioned & distributed to all DER
re-scheduling in case of unforeseen events
product is delivered

Setting

So, what do we have?

- Product from some energy market
  - Defines time frame and schedule
  - . . .
- A set of distributed energy resources
  - μ-CHP, photovoltaics, batteries, controllable consumer, . . .
  - Individually owned and operated
  - Individually configured
  - Setting mostly private
  - Each may offer a set of operable schedules

Distributed Real Power Planning

- We want: Exactly one schedule for each unit in the coalition
- Combinatorial problem: Sum of these schedules should resemble a wanted joint schedule
- Scheduling algorithm must know for each unit, which schedules are operable and which are not

Basic idea

Question

How can we model these restricted search spaces s.t. search algorithms can work on it independently of underlying unit?

Basic idea

Question

How can we model these restricted search spaces s.t. search algorithms can work on it independently of underlying unit?

Solution

Learn topographic traits of the solution (sub-) space and derive a decoder for constraint-free problem formulation!

Basic idea

Question

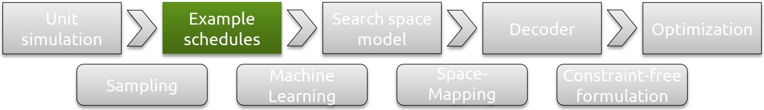
How can we model these restricted search spaces s.t. search algorithms can work on it independently of underlying unit?

Solution

Learn topographic traits of the solution (sub-) space and derive a decoder for constraint-free problem formulation!

Stopover

Let’s start with the constraints!



Geometric Constraint Interpretation

- What kind of constraints do we have to deal with?

- Operability of schedules (for given time frame) . . .

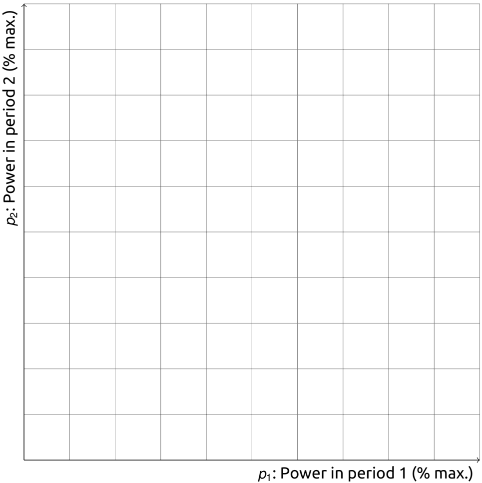
  . . . is restricted by technical constraints (min/max power output, buffer charging, etc.)

  . . . may depend on economical or ecological limiting factors (start-up cost, primary energy cost, user profiles, etc.)

  . . . depends on current operational state

  . . . let's have a look at a single unit first. . .

Constraints (Example: μ-CHP)



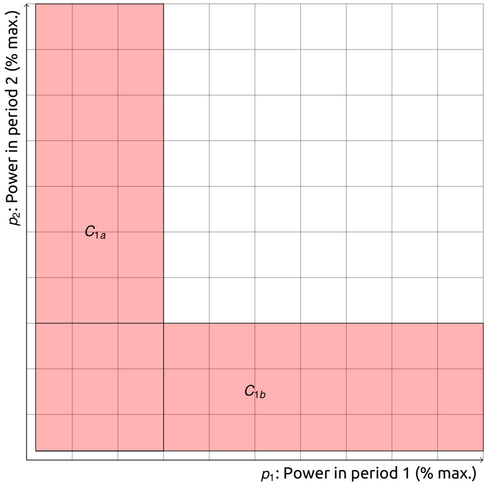
Illustrative example

- Each point in plane is schedule for 2 time periods

  X-axis: Mean active power during period 1
  Y-axis: Mean active power during period 2

- Output always between 0 and 100%
- Without further constraints: each schedule operable

Constraints (Example: μ-CHP)



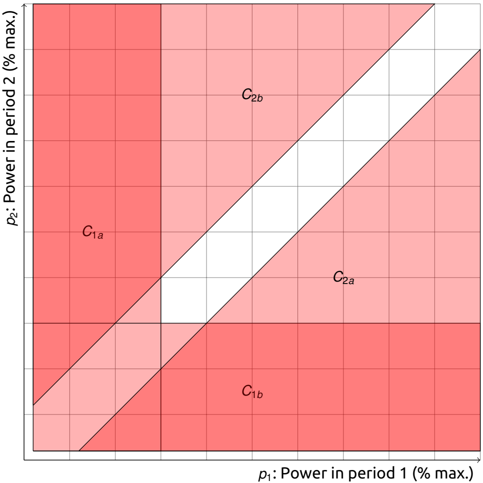
Constraint C_1: Modulation

- Only within given range
- OFF is additional option

(exaggerated depiction)

⇒ Red area drops off the solution space

Constraints (Example: μ-CHP)

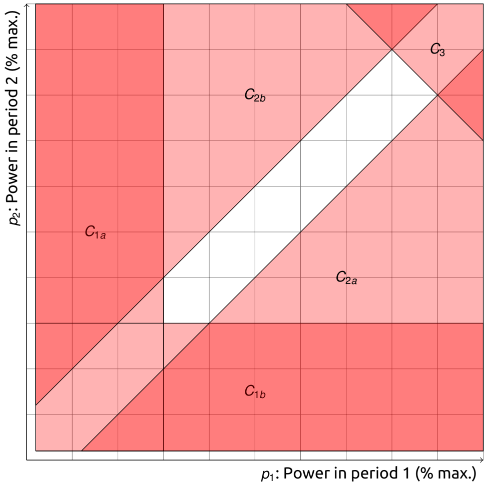


Constraint C_2: Inertia

- No instantaneous changes

⇒ Additional areas drop off

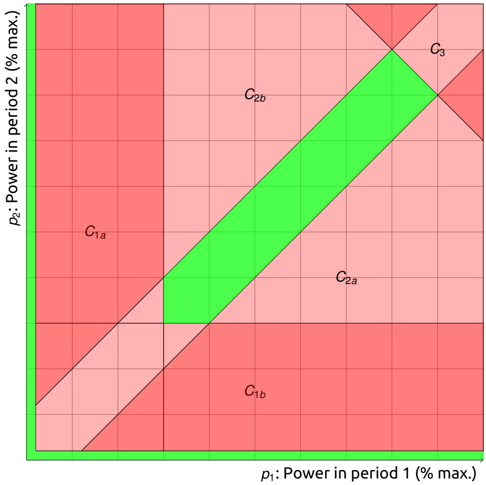
Constraints (Example: μ-CHP)



Constraint C_3: Buffer capacity

- Use or store concurrently produced thermal energy
- But: Buffer store has limited capacity

Constraints (Example: μ-CHP)



Superposition of all constraints

- Remaining region is solution space
- Only take schedules from this region
- Dimension: *96 and more* not unusual!

Stopover

How can we describe the solution space?

Basic Idea of Description

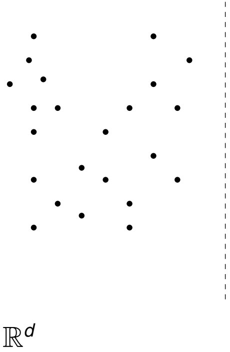
- Solution space is a region in $\mathbb{R}^d$
  - Elements are active power schedules $p = (p_1, \ldots, p_d) \in \mathbb{R}^d$
  - with mean active power during period $0 \leq i \leq d$
- Structure is abstraction for
  - The unit (or rather its future control capabilities)
  - The set of constraints

Idea:

Have the region learned from a set of example schedules

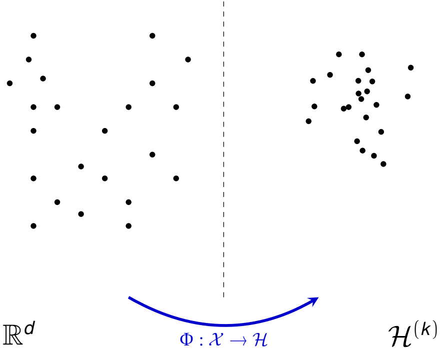
A support vector approach works very well here. . .

SVDD: Basic Idea



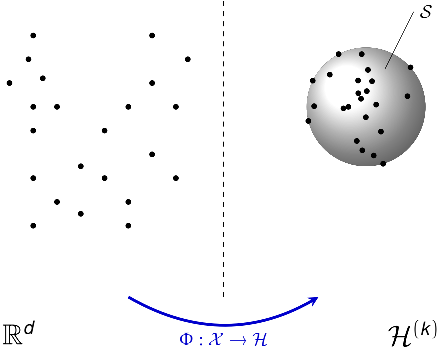
- Given: sample $\mathcal{X} \in \mathbb{R}^d$
- Here: set of schedules
- Wanted: enclosing surface
  - not necessarily connected

SVDD: Basic Idea



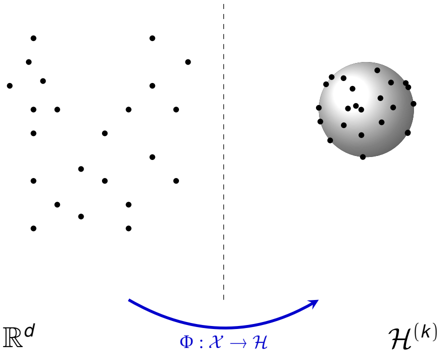
- Step 1: Map to some Hilbert space $\mathcal{H}^k$ with $k >> d$
- Let Φ be a mapping that can do it
  - Φ is unknown
  - And many Φ would do

SVDD: Basic Idea



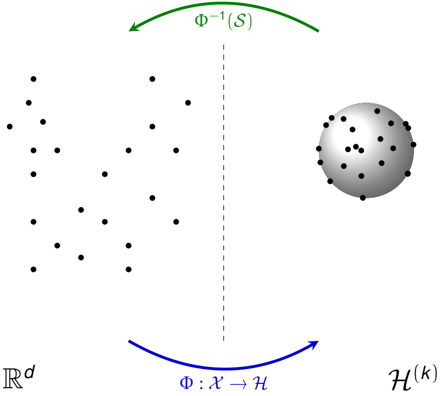
- $\exists$ a hyper-sphere $\mathcal{S}$, that contains all images of $\Phi(\mathcal{X})$
- Different Φ imply different spheres of different size
- But then, there is a smallest one, so. . .

SVDD: Basic Idea



- Step 2: Find smallest sphere
- Minimize:
  $\|\Phi(x_i) - a\|^2 \leq R^2 + \xi_i \quad \forall i$
- Use: Mercer's theorem and substitute dot-products in $\mathcal{H}$ with kernel in $\mathbb{R}^d$
- Result:
  - Set of support vectors
    - Mapped directly onto the surface of the sphere
    - Here: subset of example schedules
  - Distance-function

SVDD: Basic Idea

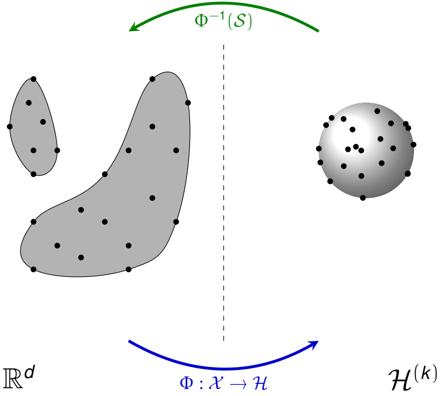


- Result:
  - Set of support vectors
    - Mapped directly onto the surface of the sphere
    - Here: subset of example schedules
  - Distance-function
    - All calculations can be done in data space ($\Phi(x) \cdot \Phi(y) = k(x, y)$)
    - We only need support vectors (non-zero weights)

Distance to center:

$$R^2(x) = \|\Phi(x) - a\|^2 = k(x, x) - 2 \sum_i \beta_i k(x_i, x) + \sum_{i,j} \beta_i \beta_j k(x_i, x_j)$$

SVDD: Basic Idea



- Step 3: Determine pre-image of sphere
- Set of points $\{x | R^2(x) \leq R^2(s)\}$ defines solution space
- Decision boundary separates feasible and in-feasible solutions

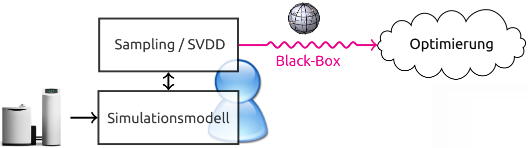
Distance to center:

$R^2(x) = ||\Phi(x) - a||^2 = k(x, x) - 2\sum_i \beta_i k(x_i, x) + \sum_{i,j} \beta_i \beta_j k(x_i, x_j)$

Modeling the Solution Space

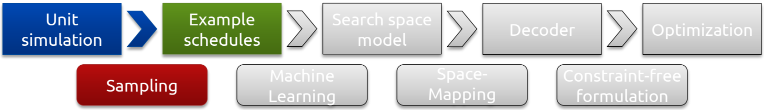
- Model consists of:
  - Set of support vectors $SV = \{x_i \in \mathcal{X} \mid \beta_i \neq 0\}$
  - Associated weights: $w = (\beta_1, \ldots, \beta_n) \quad \forall \beta \neq 0$
  - Some additional unit parameters: e.g. max. power, . . .
- Model is a black-box
  - Decision function: $R^2(x) = 1 - 2\sum_i w_i k_{\mathcal{G}}(s_i, x) + \sum_{i,j} w_i w_j k_{\mathcal{G}}(s_i, s_j)$
  - Solution space: $\{x | R(x) \leq R_{\mathcal{S}}\}$
  - Decision: *feasible or not*

. . . and ordering according to proximity to feasibility



Stopover

But, where does the sample come from?



Sampling

- We need a set of example schedules
  - As a stencil for the region that they reside in
  - We have: simulation model of the respective unit that

    ... can check feasibility of given schedule

    ... and thus may serve as a characteristic function
  - Naïve approach: Generate random schedule and check with simulation model

- But:
  - Example: if $1/3$ is infeasible in each time period
  - Then for a whole day, a fraction of $(\frac{2}{3})^{96} \approx 1.25 \times 10^{-17}$ is feasible

  $\Rightarrow$ Correct guessing very unlikely

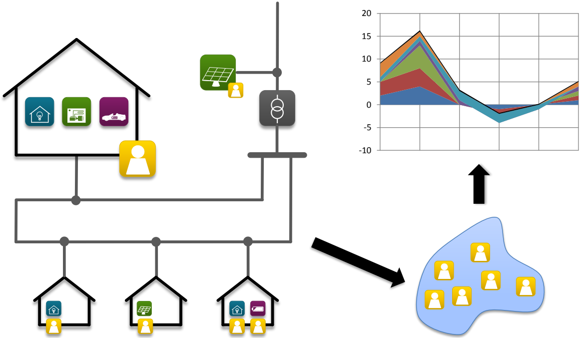
Successive Sampling

- Solution: Successively build schedules for the sample
- In co-operation with simulation model:
  - Guessing a 1-dimensional schedule correctly
    ... quite likely
    ... even more with more than one try
  - Simulation model determines follow-up state
  - Repeat until schedule complete
  - Probability: $P^d_{(n)} = \left(\sum_{i=1}^{n} B(i|P,n)\right)^d$, current example: 0.314
- Advantage: Interface only comprises evaluation of given schedule
- For correct density: e.g. kernel density estimation

Stopover

How can we use this model for optimization?

Optimization Problem



- Each unit delivers a search space model
- Search is defined on this set of models

Integration Problems

Problem

Integration of model into optimization showed up to be not so easy!

- Use as blackbox model still implies a need for constraint handling for each non-linear distance-function
- Using distance as external penalty mostly got stuck in infeasible solutions

  . . . too many penalties!

So, we went for another idea. . .

Decoder: Concept

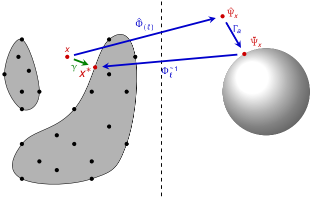
Idea:

Build a decoder based on the SVDD-model!

- In general: A decoder gives hints on how to construct feasible solutions
- Here: Build a mapping that
  - … makes an arbitrary solution feasible (solution repair)
  - … maps space of *all* schedules to feasible region for constraint free problem formulation
  - … that can be automatically derived from model

Decoder: Idea

$$
\begin{array}{ccc}
x \in [0,1]^d & \xrightarrow{\hat{\Phi}_\ell} & \hat{\Psi}_x \in \mathcal{H}^{(\ell)} \\
\downarrow \gamma & & \downarrow \Gamma_a \\
x^* \in \mathcal{F}_{[0,1]} \subseteq [0,1]^d & \xleftarrow[\Phi_\ell^{\sim 1}]{} & \tilde{\Psi}_x \in \mathcal{S}^{(\ell)} \in \mathcal{H}^{(\ell)}
\end{array}
$$

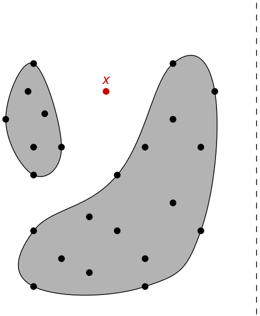


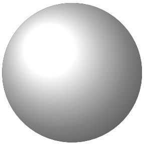
Construct: Mapping $\gamma = \Phi_\ell^{\sim 1} \circ \Gamma_a \circ \hat{\Phi}_\ell$

Step by step:

1. Empirical mapping into sub-space of $\mathcal{H}$
2. Adjustment towards sphere center
3. Find pre-image of adjusted image

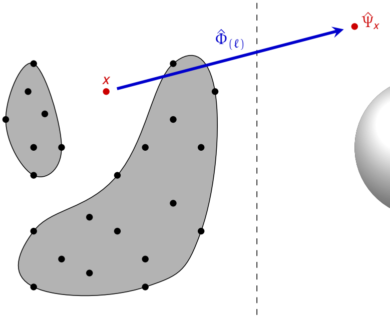
Decoder: Implementation





- Let x be an arbitrary (infeasible) Solution candidate

Decoder: Implementation



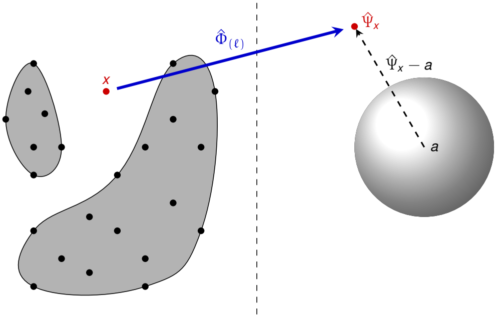
Empirical kernel map

$$\hat{\Phi}_\ell(x): \quad \mathbb{R}^d \to \mathcal{H}^{(\ell)}$$

$$x \mapsto K^{-\frac{1}{2}} \begin{pmatrix} k(s_1, x) \\ \dots \\ k(s_\ell, x) \end{pmatrix}$$

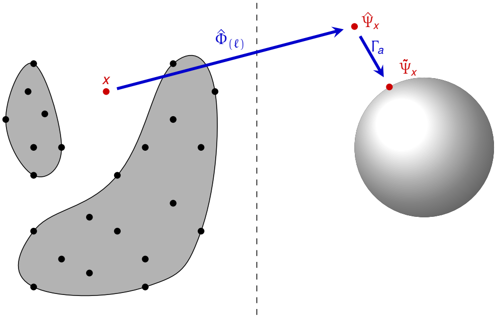
- Step 1: Map to kernel space
  - spanned by support vectors
- Infeasible solution $\Rightarrow$ Image outside of sphere

Decoder: Implementation



- Image, center and distance of image are known

Decoder: Implementation



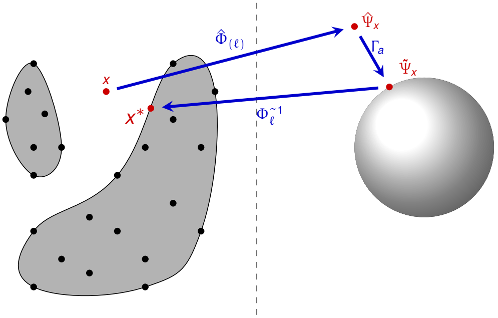
Adjustment in $\mathcal{H}$

$$\Gamma_a(\hat{\Psi}_x): \quad \mathcal{H}^{(\ell)} \to \mathcal{H}^{(\ell)}$$

$$\hat{\Psi}_x \mapsto a + \frac{(\hat{\Psi}_x - a) \cdot R_S}{R_x}$$

- Step 2: Move image towards feasibility

Decoder: Implementation



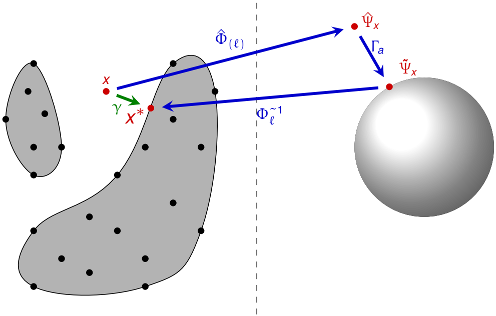
Pre-image

$$\Phi_\ell^{\sim 1}: \quad \mathcal{H}^{(\ell)} \longrightarrow \mathbb{R}^d$$

$$x_{n+1}^* = \frac{\sum_{i=1}^{\ell} \left(\tilde{w}_i^{\Gamma_a} e^{-\|s_i - x_n^*\|^2/2\sigma^2} s_i\right)}{\sum_{i=1}^{\ell} \left(\tilde{w}_i^{\Gamma_a} e^{-\|s_i - x_n^*\|^2/2\sigma^2}\right)}$$

- Step 3: Find pre-image
- Resulting in a point at outskirts of feasible region

Decoder: Implementation

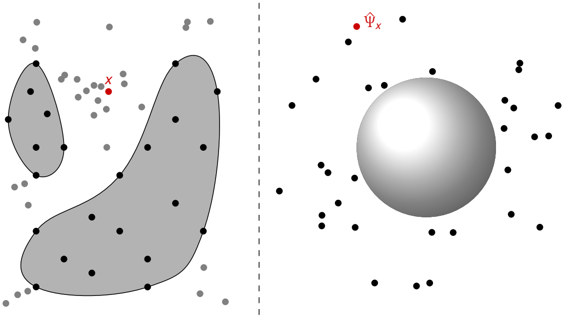


Sought mapping

$$\gamma = \Phi_\ell^{\sim 1} \circ \Gamma_a \circ \hat{\Phi}_\ell$$

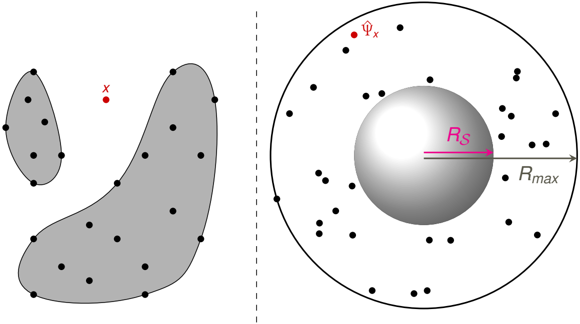
- Concatenation yields the mapping
- But: so far only surface is used

Decoder: Implementation



- Actually, we do not have really arbitrary points (Box-Constraint)
    - Points are from $[0, 1]^d$

Decoder: Implementation



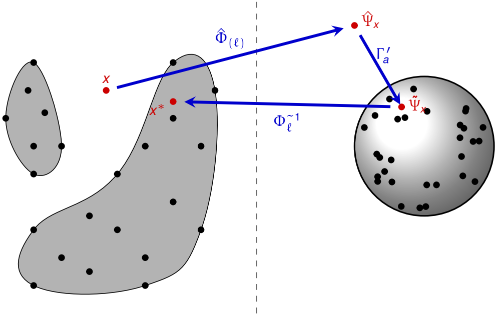
Improvement

$$\tilde{\Psi}_x = a + \frac{R_S}{R_{max}} \cdot (\hat{\Psi}_x - a)$$

But: R_{max} unknown!

- $\exists$ a larger sphere that contains all images
- Might be rescaled to the smaller one

Decoder: Implementation



Improvement

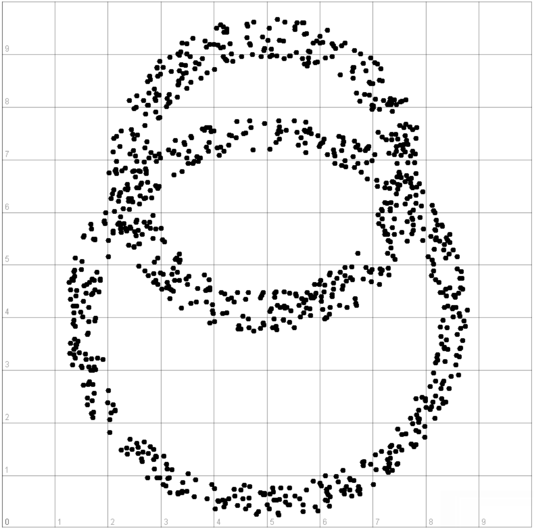
$$\tilde{\Psi}_x = a + \frac{R_{\mathcal{S}}}{R_{max}} \cdot (\hat{\Psi}_x - a)$$

But: R_{max} unknown!

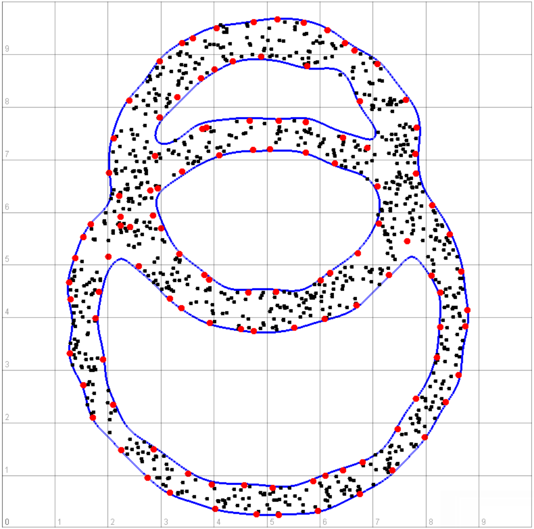
- Enables constraint free formulation

... by mapping to different search space

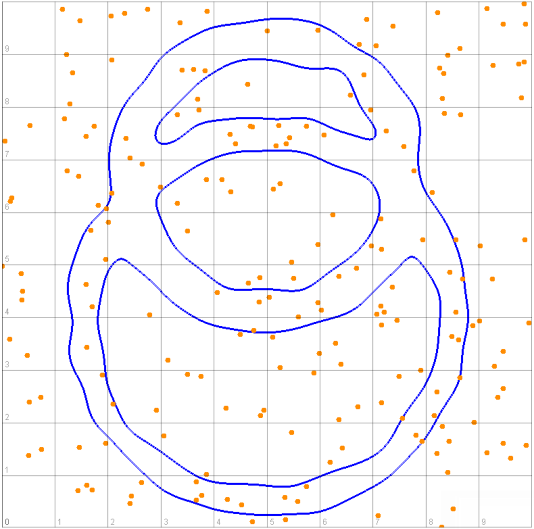
Decoder: Results



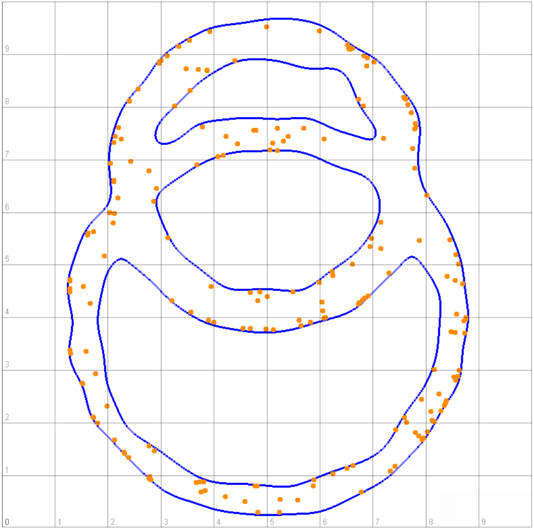
Decoder: Results



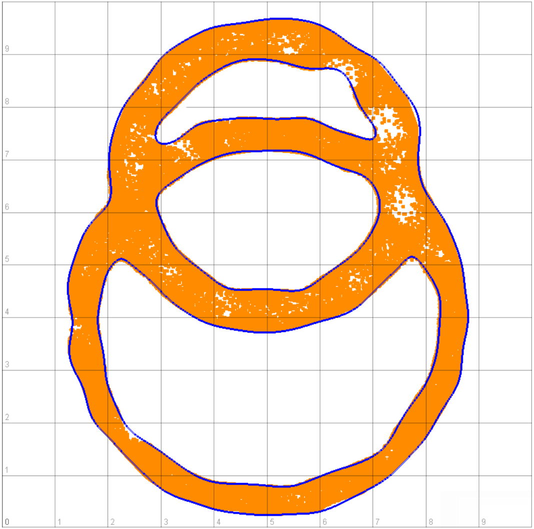
Decoder: Results



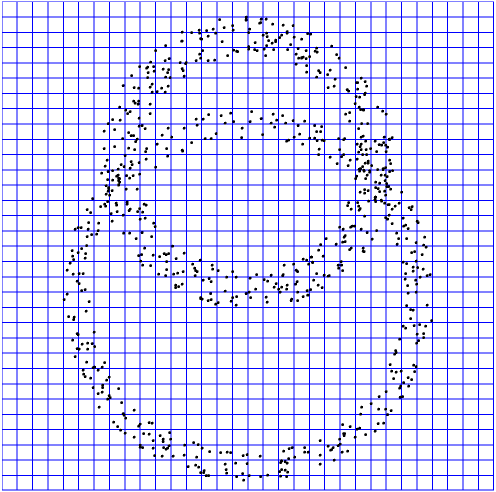
Decoder: Results

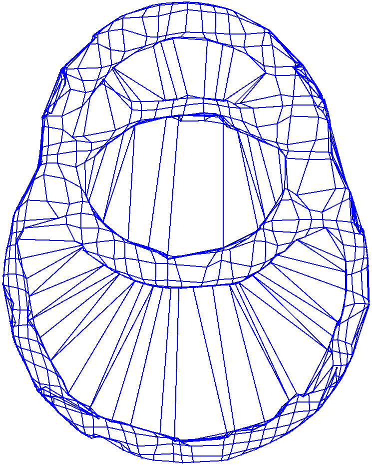


Decoder: Results



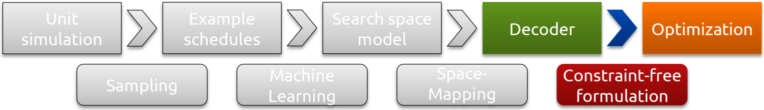
Decoder: Results





Stopover

Now we can integrate model and optimization!



Optimization

- Involved: n distributed energy units $\longrightarrow$
  - n search space models and
  - thus n decoder mappings $\gamma_1 \ldots \gamma_n$
- Also given by market $\longrightarrow$ target schedule ζ

- Configuration: $\sigma = (x_1, \ldots, x_n)$, with $x_i \in [0, 1]^d$
- Error: $\mathcal{H} = \|\zeta - \sum P_i^{max} \cdot \gamma_i(x_i)\|$
- Configuration may evolve in $([0, 1]^d)^n$

- Let $\tau = (\tau_1, \ldots, \tau_n)$ be configuration with smallest found error $\mathcal{H}$
- Then, $(P_1^{max} \cdot \gamma_1(\tau_1), \ldots, P_n^{max} \cdot \gamma_n(\tau_n))$ is solution

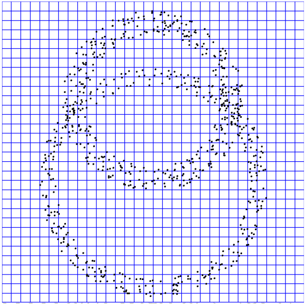
Optimization

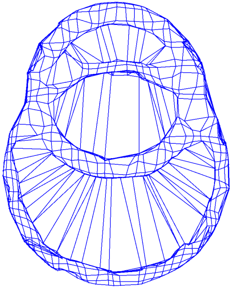
- Involved: n distributed energy units $\rightarrow$
  - n search space models and
  - thus n decoder mappings $\gamma_1 \ldots \gamma_n$
- Also given by market $\rightarrow$ target schedule ζ
- Configuration: $\sigma = (x_1, \ldots, x_n)$, with $x_i \in [0, 1]^d$
- Error: $\mathcal{H} = \|\zeta - \sum P_i^{max} \cdot \gamma_i(x_i)\|$
- Configuration may evolve in $([0, 1]^d)^n$
- Let $\tau = (\tau_1, \ldots, \tau_n)$ be configuration with smallest found error $\mathcal{H}$
- Then, $(P_1^{max} \cdot \gamma_1(\tau_1), \ldots, P_n^{max} \cdot \gamma_n(\tau_n))$ is solution

Optimization

- Involved: n distributed energy units $\rightarrow$
  - n search space models and
  - thus n decoder mappings $\gamma_1 \dots \gamma_n$
- Also given by market $\rightarrow$ target schedule ζ
- Configuration: $\sigma = (x_1, \dots, x_n)$, with $x_i \in [0, 1]^d$
- Error: $\mathcal{H} = \|\zeta - \sum P_i^{max} \cdot \gamma_i(x_i)\|$
- Configuration may evolve in $([0, 1]^d)^n$
- Let $\tau = (\tau_1, \dots, \tau_n)$ be configuration with smallest found error $\mathcal{H}$
- Then, $(P_1^{max} \cdot \gamma_1(\tau_1), \dots, P_n^{max} \cdot \gamma_n(\tau_n))$ is solution

Optimization





- Configuration: $\sigma = (x_1, \ldots, x_n)$ with $x_i \in [0, 1]$
- Error: $\mathcal{H} = \|\zeta - \sum P_i^{max} \cdot \gamma_i(x_i)\|$
- Configuration may evolve in $([0, 1]^d)^n$

- Let $\tau = (\tau_1, \ldots, \tau_n)$ be configuration with smallest found error $\mathcal{H}$
- Then, $(P_1^{max} \cdot \gamma_1(\tau_1), \ldots, P_n^{max} \cdot \gamma_n(\tau_n))$ is solution

Advantages of the Decoder

- Using standard methods for optimization
- Algorithm does not need to know anything about unit, model or constraints
- Equal treatment of different types of unit
- Integrate new, so far unknown units

. . . and distributed algorithms?

Distributed Greedy

- Use case: Distributed approach with multi-agent system
- Blackboard approach
  - ∃ commonly known joint solution

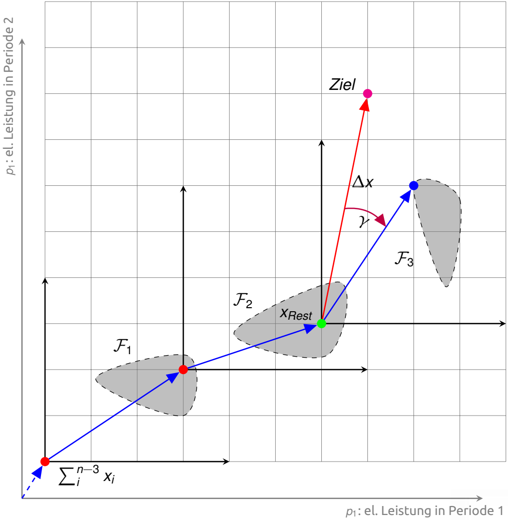
- Basic idea: every one does repeatedly his best for solution improvement
  - Difference between joint solution and own solution part: what the others do
  - Difference between target and 'what the others do': the missing part

  ... if this schedule is realizable: done!

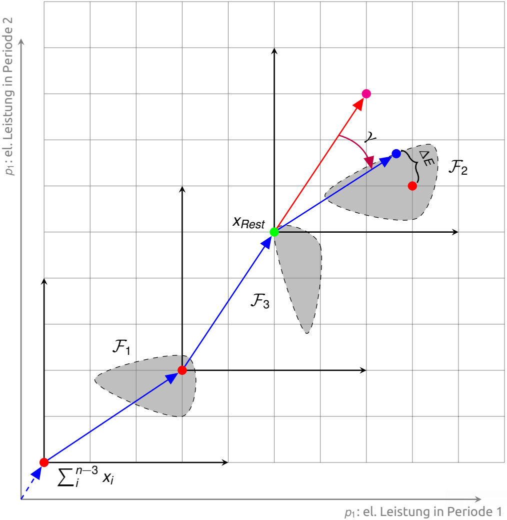
Algorithm

$\mathcal{A} \leftarrow$ List of all agents
$s_{target} \leftarrow$ target schedule
$S_a^t \leftarrow$ This agents contribution (initially zero)
if is initiator **then**
  $S \leftarrow zeros(n)$
else
  $S \leftarrow$aggregated schedule
  $S_a^{t+1} \leftarrow \gamma(s_{target} - (S - S_a))$
  $S \leftarrow S - S_a^t + S_a^{t+1}$
  if no stop criterion met **then**
    choose random agent $A \in \mathcal{A}$
    send message with S to A
  else
    publish solution S
  end if
end if

Greedy Ansatz

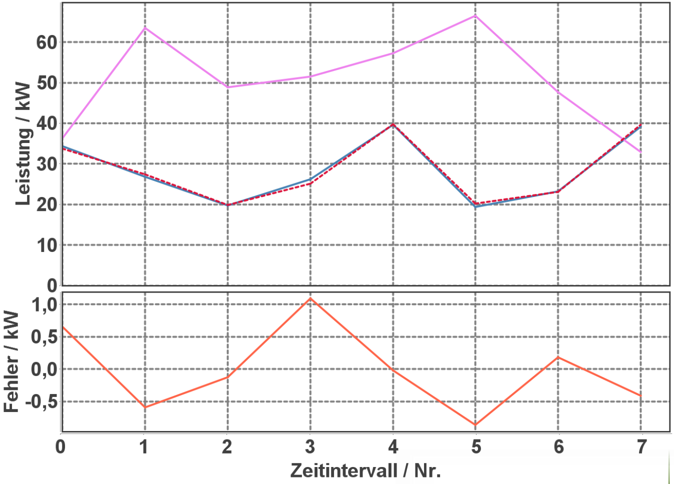


Greedy Ansatz



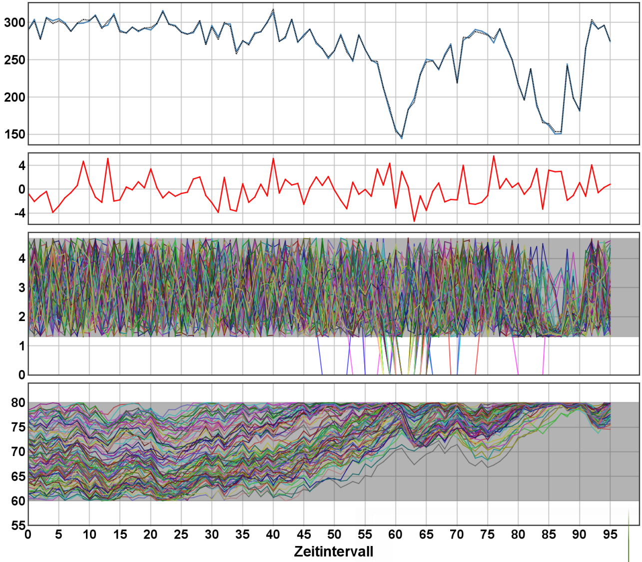
Results for Distributed Greedy

- worst case operation
- target and achieved schedule
- Remaining error ≈2.1% of worst case



Results for Distributed Greedy

- target and achieved schedule for 100 CHP / kW
- Error / kW
- individual schedules CHP / kW
- buffer temperature / °C
- grey areas: respective feasible region



Zwischenstopp

What about evaluation criteria?

Extended model

Integrating individual performance indicators:

- We do have a many-objective problem
- Indicators must be assigned to individual schedules

Idea:

- Assumption: $\exists$ a (not necessarily known) relation between schedule and indicator value
- Indicators are concatenated to schedule
- Model and decoder can be used without changes
- SVDD concurrently learns relationship
- Decoder reconstructs indicator values

Indicator results

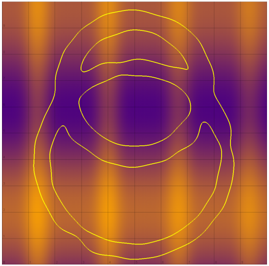
Error: Difference between original and reconstructed indicator value

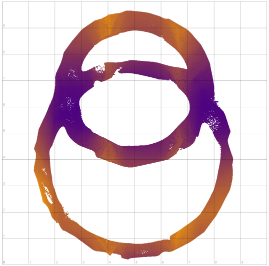
d	Rosenbrock	therm. Puffer
16	7.940e-5$\pm$2.104e-4	1.652e-4$\pm$6.948e-5
32	8.756e-5$\pm$1.894e-4	1.517e-4$\pm$7.549e-5
48	1.368e-4$\pm$2.393e-4	1.513e-4$\pm$7.188e-5
64	1.300e-4$\pm$2.383e-4	1.441e-4$\pm$7.260e-5
80	1.051e-4$\pm$2.448e-4	1.461e-4$\pm$7.433e-5
96	1.019e-4$\pm$1.988e-4	1.500e-4$\pm$7.721e-5

Example indicators

- Cost
- Preserving degrees of freedom
- Final state (for re-scheduling)

...





Conclusion

- Support Vector Decoder
  - … flexible Model
  - … allows for easy integration into optimization
  - … supports many use cases
    - Coaltion formation
    - Re-scheduling
    - Profit distribution
- Working prototype has been realized in Smart Nord
- Use case not restricted to power planning

Literature

Bremer, J., & Sonnenschein, M. (2013).
Constraint-Handling for Optimization with Support Vector Surrogate Models: A Noval Decoder Approach.
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART 2013)* Barcelona, Spain

Bremer, J., & Sonnenschein, M. (2012).
A Distributed Greedy Algorithm for Constraint-based Scheduling of Energy Resources.
In *1st International Workshop on Smart Energy Networks & Multi-Agent Systems (SEN-MAS 2012)* Wroclaw, Poland.

Bremer, J., Rapp, B., & Sonnenschein, M. (2011a).
Encoding distributed search spaces for virtual power plants.
In *IEEE Symposium Series in Computational Intelligence 2011 (SSCI 2011)* Paris, France.

Bremer, J., Rapp, B., & Sonnenschein, M. (2011b).
Including Environmental Performance Indicators into Kernel based Search Space Representations.
In *Information Technologies in Environmental Engineering (ITEE 2011).*

Bremer, J., Rapp, B., & Sonnenschein, M. (2010).
Support vector based encoding of distributed energy resources' feasible load spaces.
In *IEEE PES Conference on Innovative Smart Grid Technologies Europe* Chalmers Lindholmen, Gothenburg, Sweden.

Vielen Dank für Ihre Aufmerksamkeit!

Basisliteratur

Tax, David M. J. & Duin, R. P. W. (2004).
Support Vector Data Description.
In *Journal of Machine Learning* Kluwer Academic Publishers, Hingham, USA

B. Schölkopf and S. Mika and C. Burges and P. Knirsch and K.-R. Müller and G. Rätsch and A. Smola (1999).
Input space vs. feature space in kernel-based methods.
In *IEEE Transactions on Neural Networks*

Park, J., Kang, D., Kim, J., Kwok, J. T., Tsang,& Ivor W. (2007).
SVDD-Based Pattern Denoising.
In *Journal of Neural Computing* MIT Press, Cambridge, USA

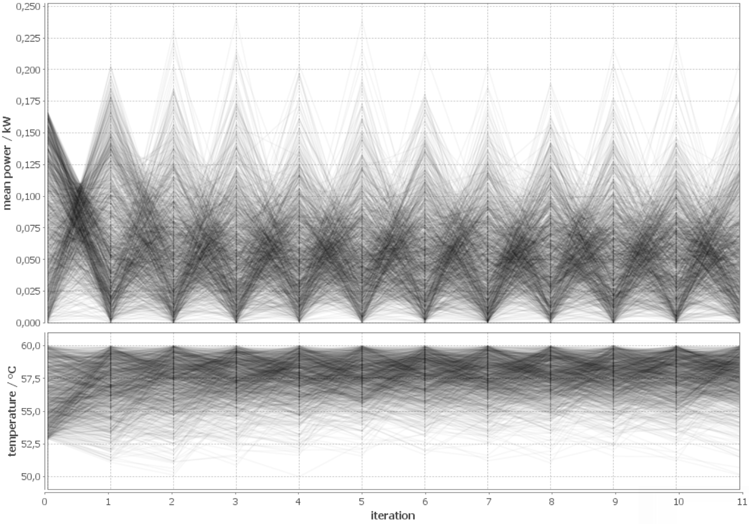
Kwok, J. T. Y. & Tsang, I. W. H. (2004).
The pre-image problem in kernel methods.
In *IEEE Transactions on Neural Networks*

Squire, W. (1975).
Computer implementation of the Schwarz-Christoffel transformation.
In *Journal of the Franklin Institute* Elsevier

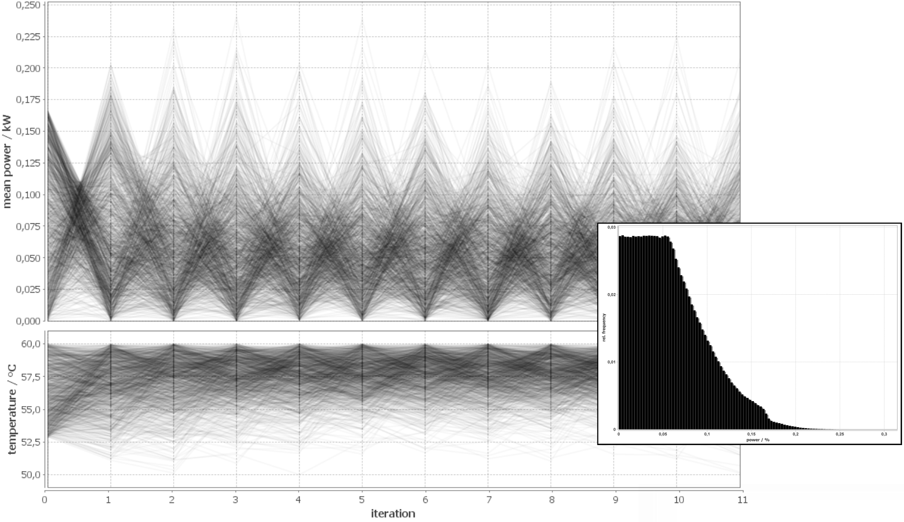
Ben-Hur, A., Siegelmann, H.T., Horn, D., & Vapnik, V. (2001).
Support Vector Clustering.
In *Journal of Machine Learning Research* IEEE Computer Society, Los Alamitos, USA

Platt, J.C. (1998).
Fast training of support vector machines using sequential minimal optimization.
In *Advances in Kernel Methods* MIT Press

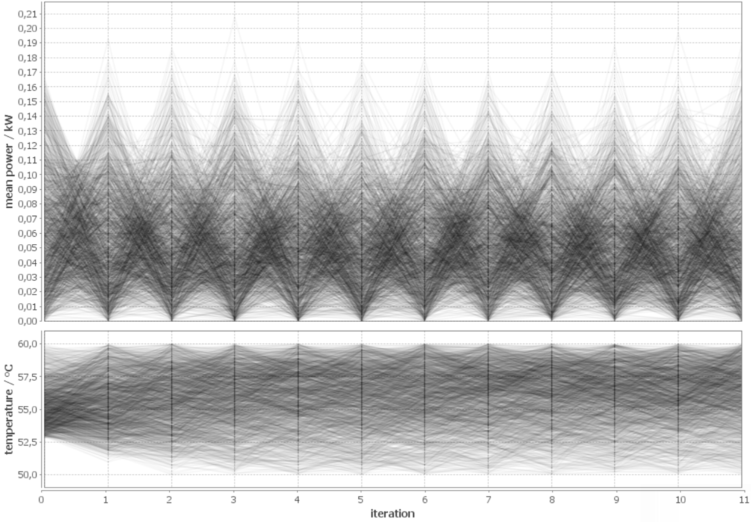
Kerndichteschätzer (Beispiel)



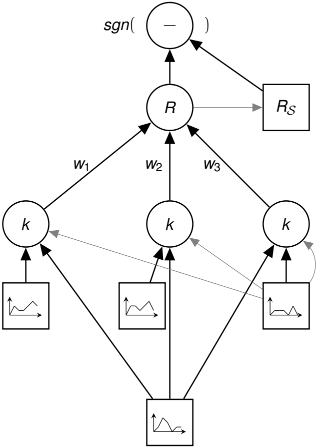
Kerndichteschätzer (Beispiel)



Kerndichteschätzer (Beispiel)



Klassifikationsprinzip



Klassifikation

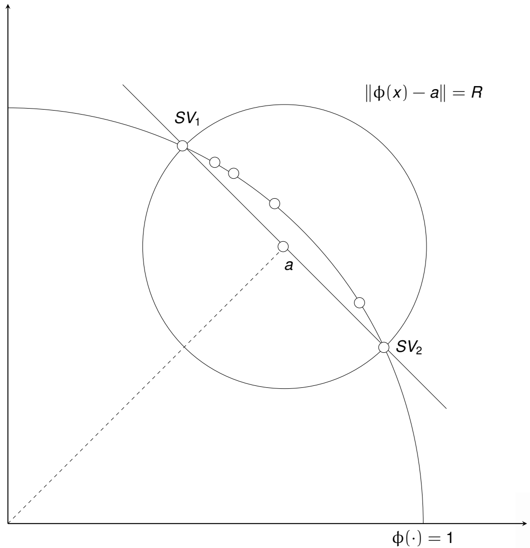
Abstandsbestimmung

Vergleich

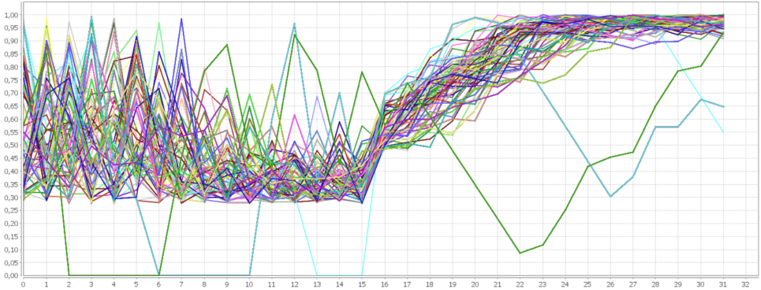
Supportvektoren

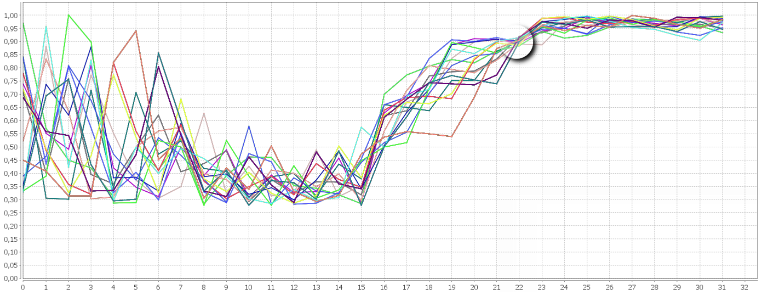
Eingabelastgang

Hypersphäre

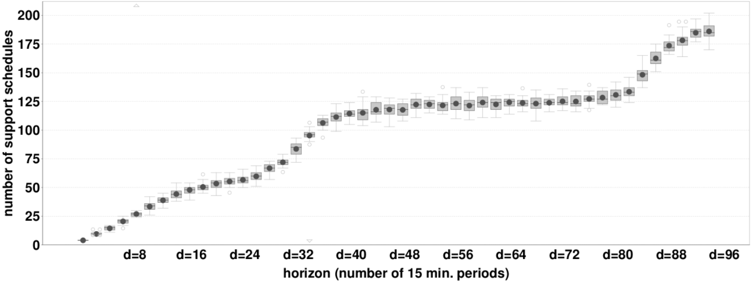


Zustand


1,00
0,95
0,90
0,85
0,80
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0,00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32


1,00
0,95
0,90
0,85
0,80
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0,00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

Anzahl


200
175
150
125
100
75
50
25
0
number of support schedules
d=8
d=16
d=24
d=32
d=40
d=48
d=56
d=64
d=72
d=80
d=88
d=96
horizon (number of 15 min. periods)

Optimierung

- Beteiligt: n dezentrale Anlagen $\rightarrow$
  - n Suchraummodelle und
  - somit n Dekoderabbildungen $\gamma_1 \dots \gamma_n$
- Ferner: Markt $\rightarrow$ Ziellastgang ζ
- Lösungskonfiguration: $\sigma = (x_1, \dots, x_n)$, wobei $x_i \in [0, 1]^d$
- Zu minimierender Fehler: $\mathcal{H} = \|\zeta - \sum P_i^{max} \cdot \gamma_i(x_i)\|$
- Variation der Lösungskonfiguration kann in $([0, 1]^d)^n$ erfolgen
- Sei $\tau = (\tau_1, \dots, \tau_n)$ die Konfiguration mit dem kleinsten gefundenen Fehler $\mathcal{H}$
- Dann ist $(P_1^{max} \cdot \gamma_1(\tau_1), \dots, P_n^{max} \cdot \gamma_n(\tau_n))$ die Lösung

Klassische Ansätze

Constraintbasierte Optimierung

$$f(x_1, x_2, \ldots, x_n) \rightarrow \min, \quad \text{s.t.} \quad R_i(x_i) \leq R_{S_i},\ 0 \leq i \leq n$$

- Problem: Nicht-lineare Nebenbedingungen; so noch keine Constraintbehandlung

Kombinatorisch (Multiple Choice Subset Sum)

$$f(x_1, x_2, \ldots, x_n) \rightarrow \min, \quad \text{s.t.} \quad x_i \in \mathcal{X}_2 \subset \mathcal{F}_i,\ 0 \leq i \leq n$$

- Problem: Erzeugen von Lösungskandidaten schwierig

Optimierung mit Penalty

$$f(x_1, x_2, \ldots, x_n) + p(R_1(x_1), R_2(x_2), \ldots, R_i(x_i)) \rightarrow \min$$

- Problem: Ungültige Lösungen möglich und mit wachsender Problemgröße immer wahrscheinlicher