Spyros Chatzivasileiadis

Data-driven applications for power system security assessment

Spot the difference!

North East Blackout 2003

- North East Blackout 2003
- Affected 55 million people in the US and Canada
- Estimated cost of 7-10 billion USD
- Took 2 days (!) for full restoration!

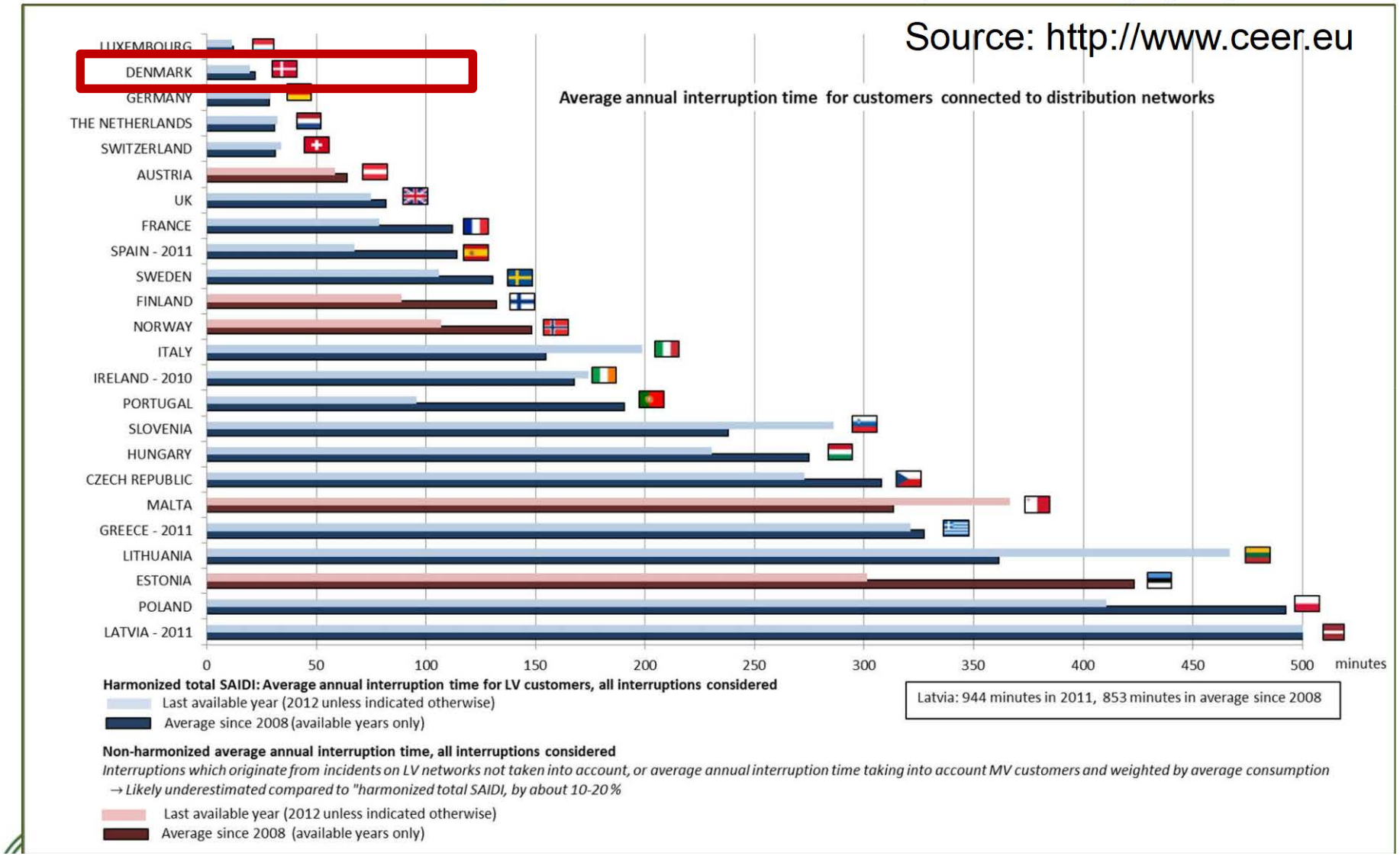
Blackouts are rare but costly!

- Frequency of power interruptions
  - 1 hour per year
- Economic damage from power interruptions
  - about 80 billion USD/year (US only, 2005)
- Total electric energy cost in the US:
  - 370 billion USD/year

India Blackout 2012 → affected 700 million people (!)

Affected regions in red

Average annual interruption time in European countries
(2012 data)



Source: M. Bollen, W. Friedl, PES GM 2014

http://grouper.ieee.org/groups/td/dist/sd/doc/2014-08%20European%20Benchmarking%20Report%20on%20Quality%20of%20Supply-%20Mathew%20Bollen+Werner%20Friedl.pdf

Operators run every day a security assessment

Energinet Control Room, Denmark

- Security = ability to withstand disturbances
- Security Assessment:
  - Screen contingency list every 15 mins
  - Prepare contingency plans for critical scenarios
- Run both:
  - Steady-state, i.e. power flows to check N-1 and violation of limits
  - Dynamic simulations

Challenges

- Dynamic simulations are hard
  - System of differential-algebraic equations with 10k degrees of freedom

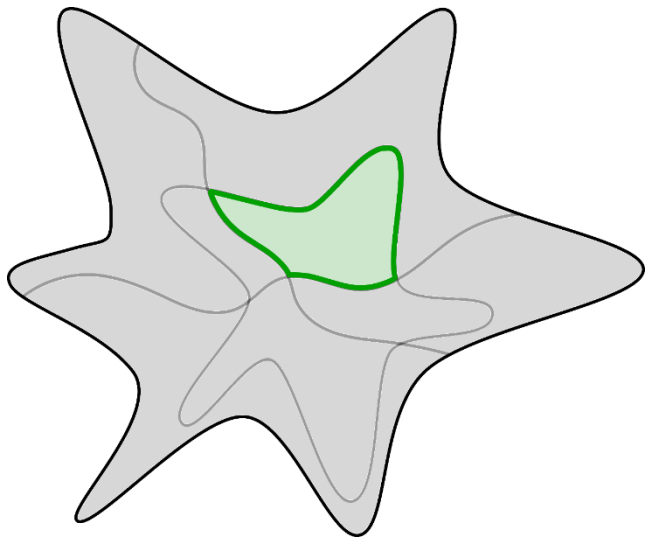
- Checking for N-k contingencies is a hard combinatorial problem
  - Usually computationally impossible to check even for all N-2 in a realistic systems with thousands of buses

- The safe operating region is a non-linear non-convex region
  - Impossible to use analytical tricks to determine it

So…. what do we do?

So… what do we do?

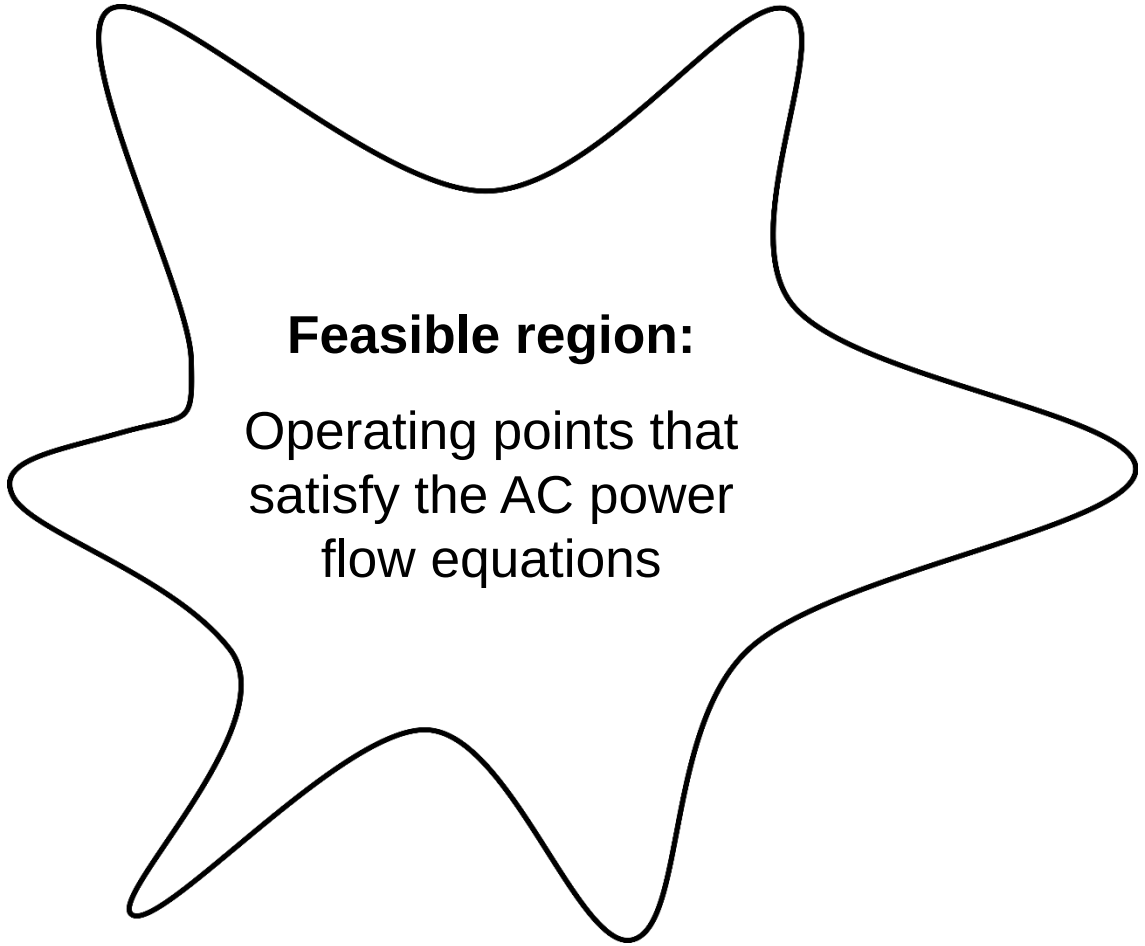
Identifying the power system security region



- Run a lot of simulations assessing each operating point
  - Several approaches for efficient approximations to boost computation speed
- Stability certificates
  - Extract **sufficient conditions** for sub-areas of the security region
- Machine learning approaches
  - Train for a given dataset and **infer** for all new points

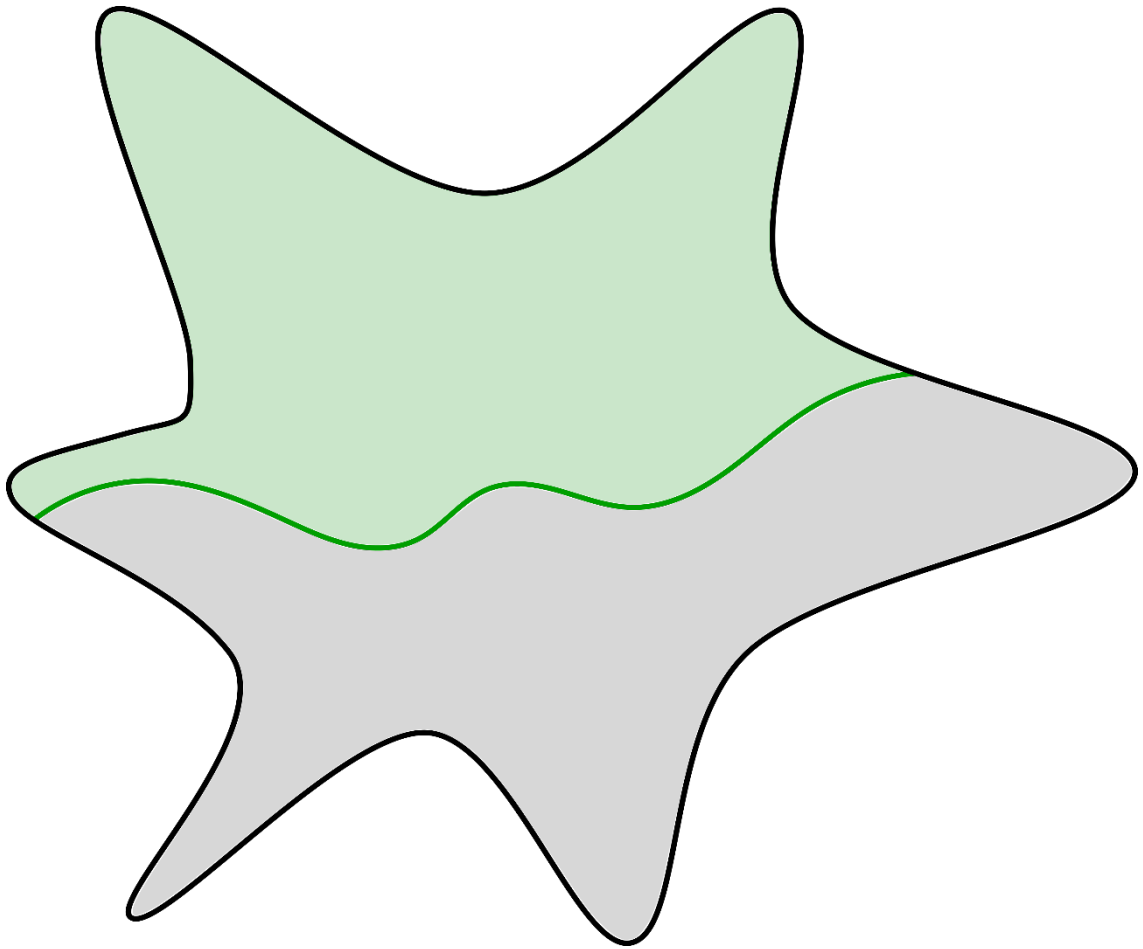
Power System Security Assessment

The safe operating region of power system operations



- Non-linear and non-convex AC power flow equations
- Component limits

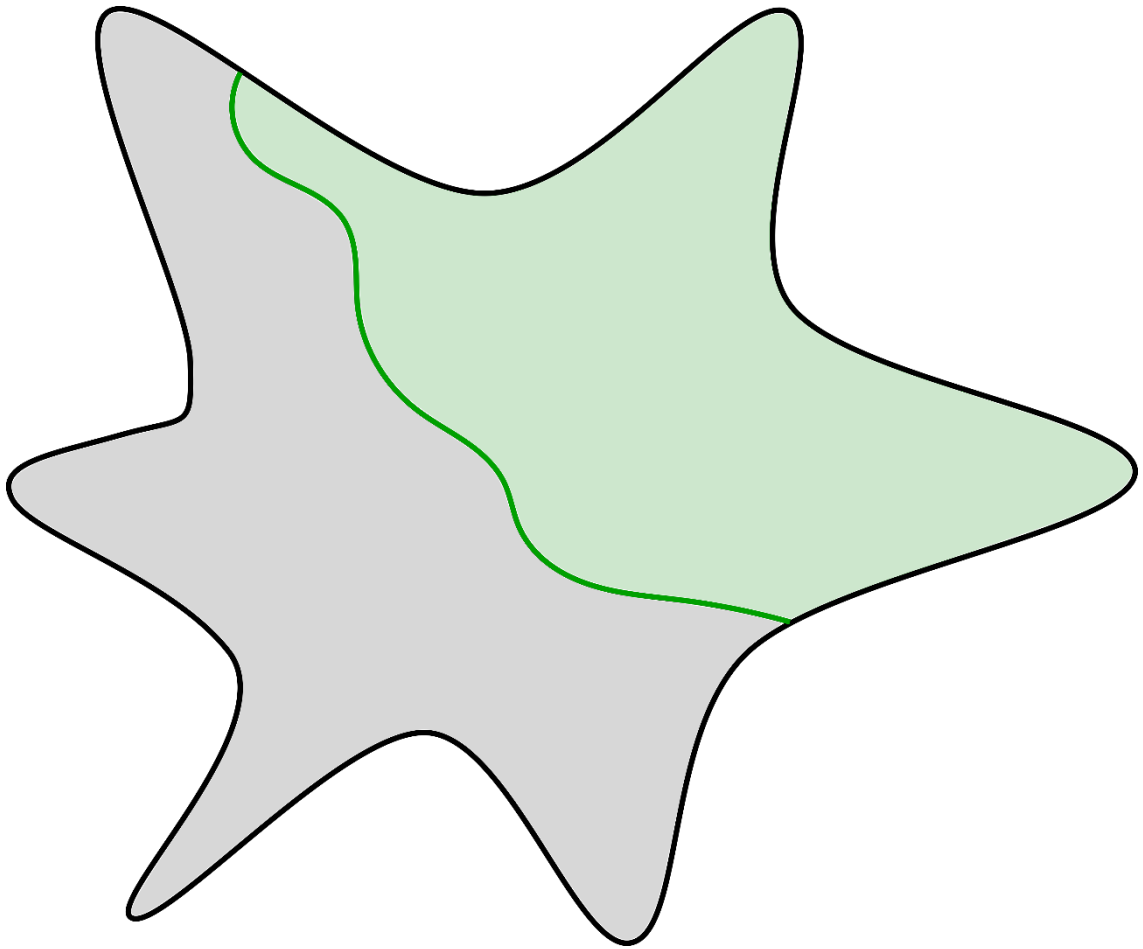
The feasible space of power system operations



- Non-linear and non-convex AC power flow equations
- Component limits

+ N-1 security criterion

The feasible space of power system operations

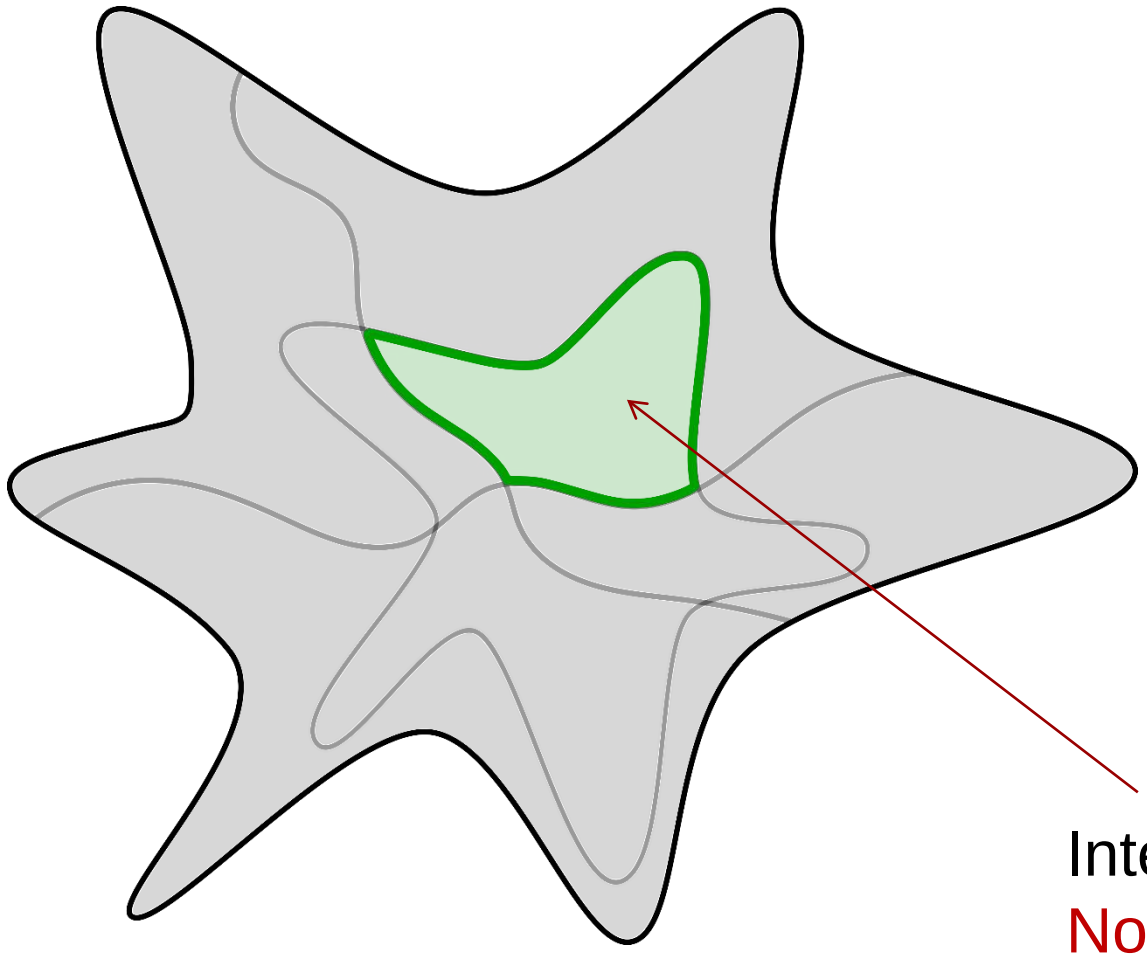


- Non-linear and non-convex AC power flow equations
- Component limits

+ N-1 security criterion

+ Stability Limits

The feasible space of power system operations



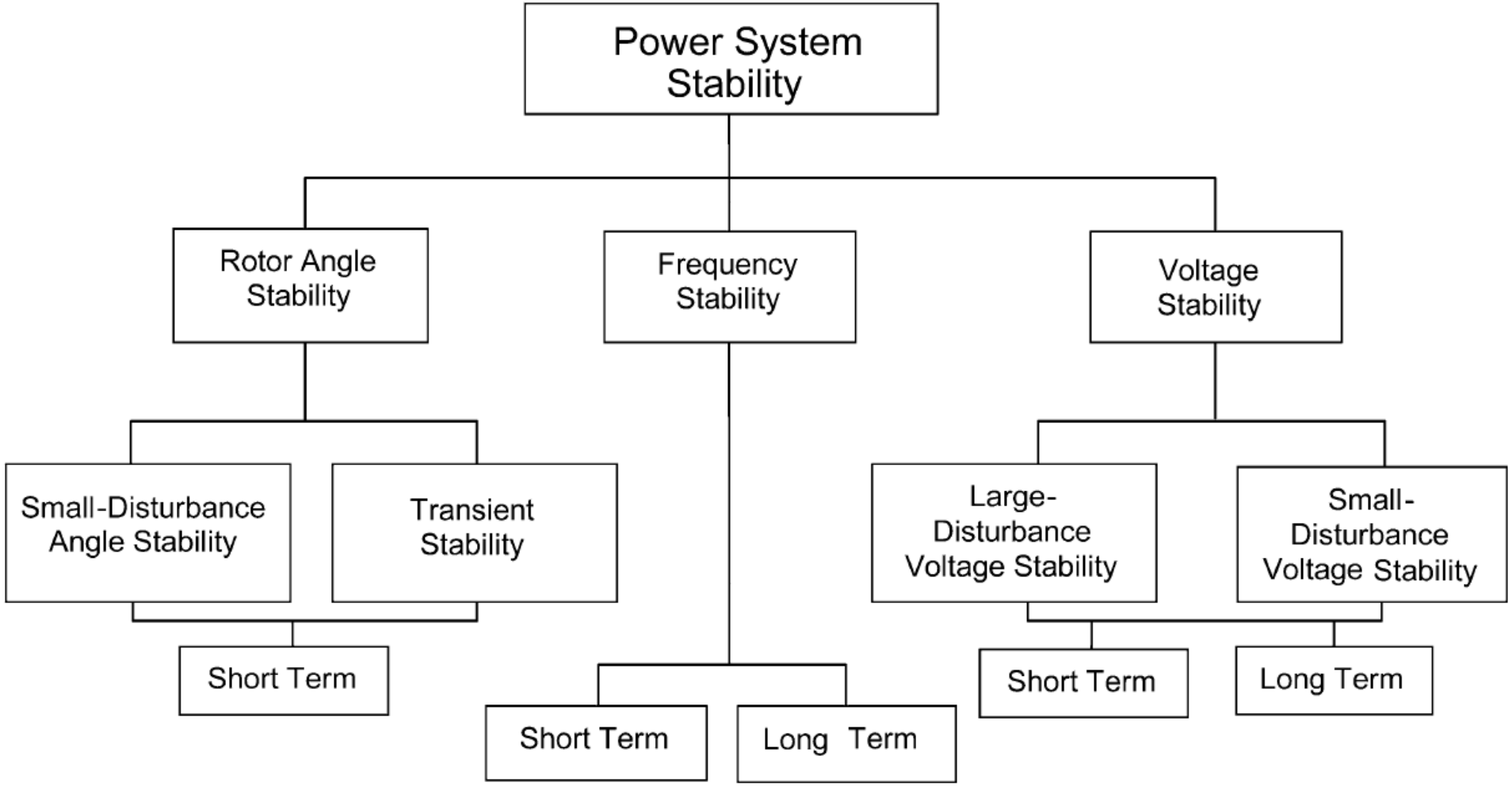
- Non-linear and non-convex AC power flow equations
- Component limits

+ N-1 security criterion

+ Stability Limits

Intersection of all security/stability criteria:
Non-linear and non-convex security region

Operators have to check for all instability types



AND steady-state security!

N-1 contingency analysis through power flows

Kundur et al., IEEE TPWRS, 2004

Focus of this lecture:
Machine Learning for Power System Security Assessment

- Decision Trees
  - First proposed by Louis Wehenkel (Univ of Liege) in the ‘90s
  - Very successful
  - Applications in the industry
  - Research is still ongoing; latest focus is on interpretability (remember Dolores’s talk? ☺)

- Neural networks

- Deep Neural Networks (same as neural networks but deep ☺)
  - One paper on feature extraction (Sun, Konstantelos, Strbac, 2018)
  - One paper inspired by image processing (Hidalgo, Hancharou, Thams, Chatzivasileiadis, 2019)

Machine learning applications

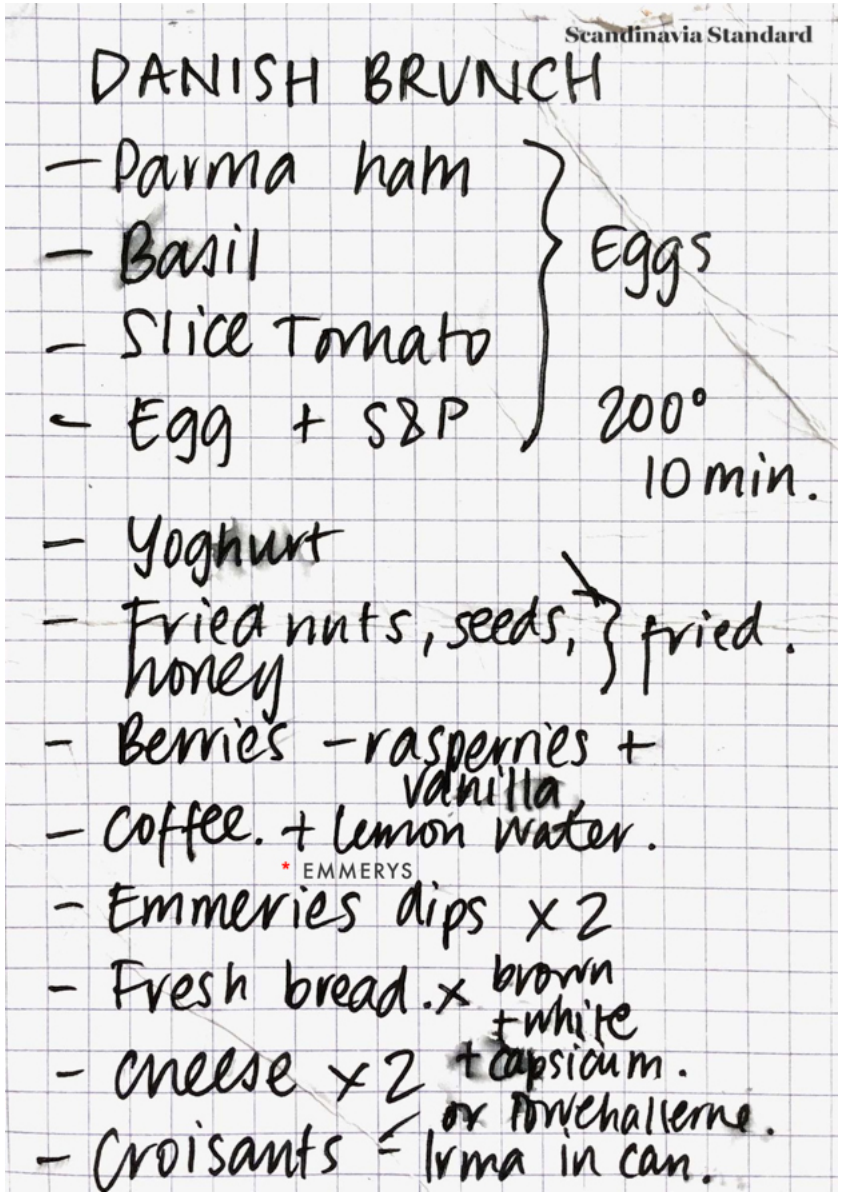
(for power system security assessment)

A <u>very</u> short overview

The ingredients

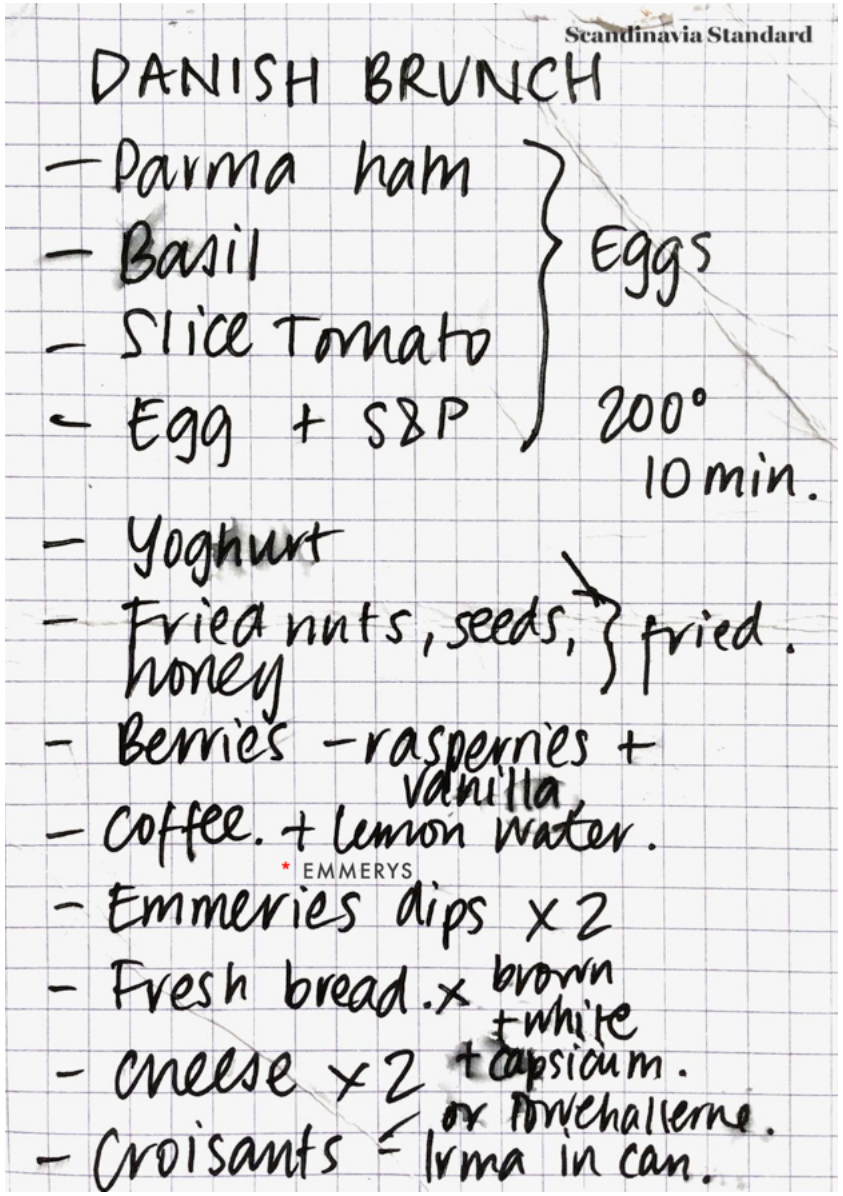
What do we need?

If we want to apply machine learning approaches for power system security



The ingredients

- A training database
- A training algorithm
- A test database
  - To test accuracy of the approach



Test Database

Test Database

Traditionally:

- Split training database to e.g. 80% training samples and 20% test samples
- Train with the 80%
- Test with the 20%

Modern toolboxes have this integrated and automatized →only need to provide a training database

Point to remember:

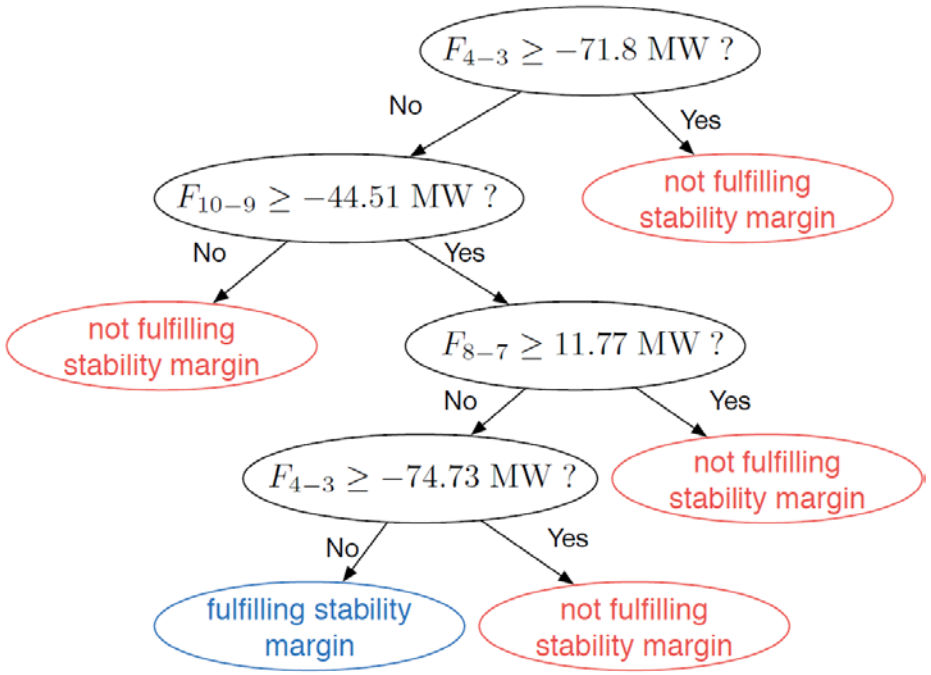
The test database determines the performance of your method. If the test data come from the same simulations as your training data, the accuracy can be deceivingly high. Would it be equally high in reality?

Ideally → use a different real-life dataset

(Unfortunately, not always possible)

Training a neural network

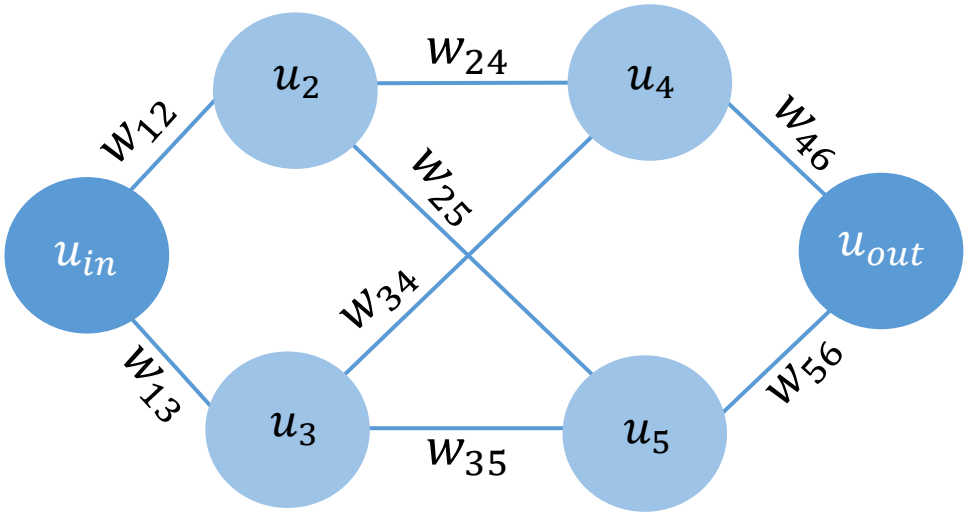
Decision Trees



- Decision rules
  - If line_flow>FF then …. else…

F. Thams, L. Halilbašić, P. Pinson, S. Chatzivasileiadis, and R. Eriksson, "Data-driven security-constrained OPF," in *10th IREP Symposium – Bulk Power Systems Dynamics and Control*, 2017.

Neural Networks



- Linear weights
- On every node: a non-linear activation function
  - e.g **RELU:** $\boldsymbol{u_j = \max(0, w_{ij}x_i + b_i)}$
  - Many other possibilities, e.g. sigmoid function and others

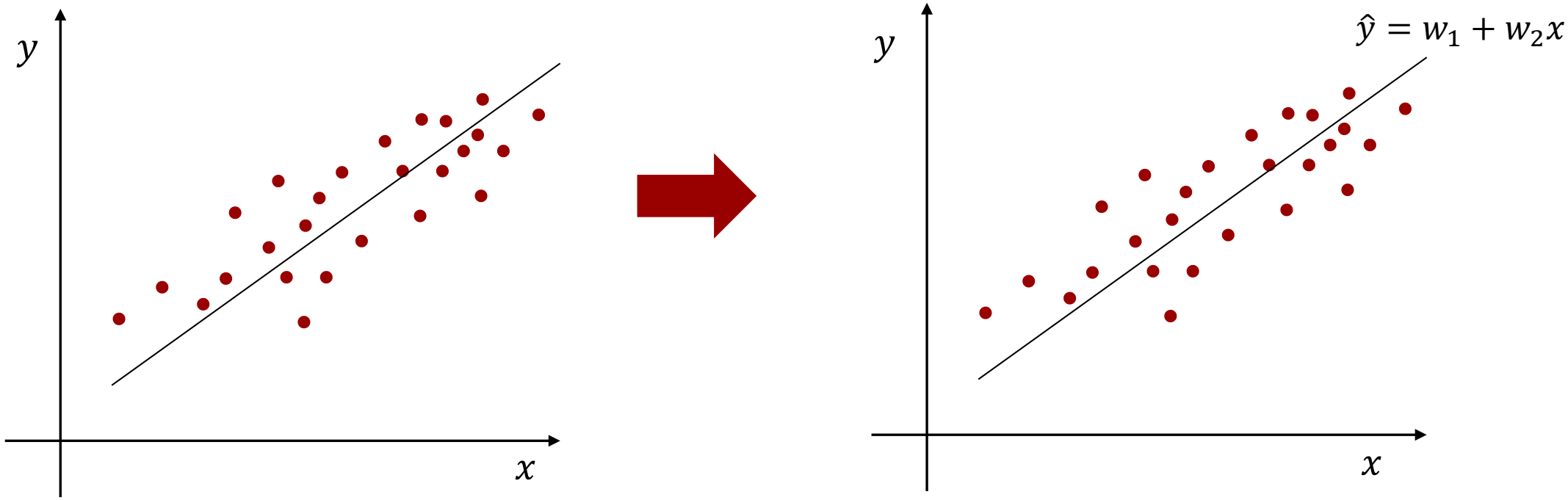
Training a neural network

- Batch size vs Iterations vs Epochs
- Loss function
- Evaluation of performance
  - Confusion matrix
  - Performance metrics

Batch size vs Iterations vs Epochs

Example from linear regression;
neural network works similarly

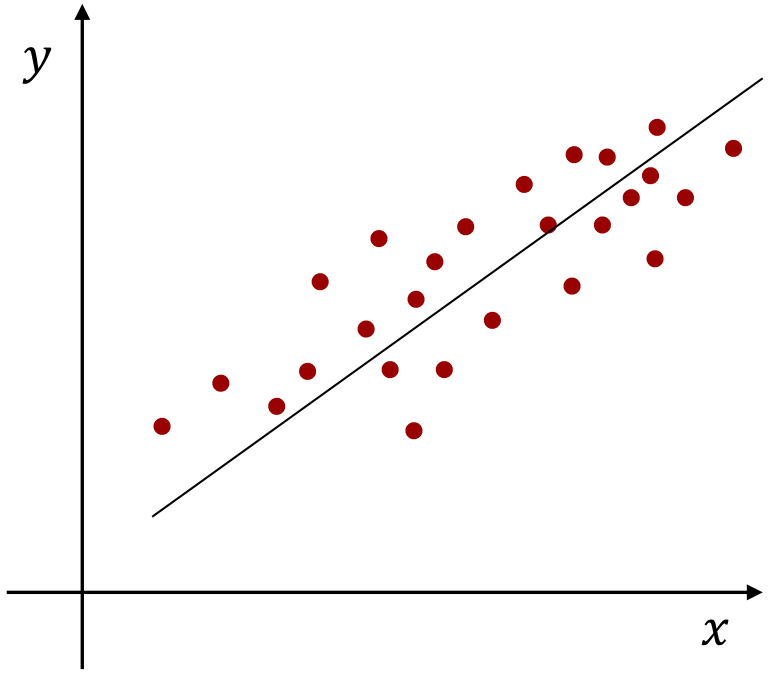
Goal: *estimate* w_1, w_2 *to fit* $\hat{y} = w_1 + w_2 x$



Batch size vs Iterations vs Epochs

Example from linear regression;
neural network works similarly

Goal: *estimate* w_1, w_2 *to fit* $\hat{y} = w_1 + w_2 x$

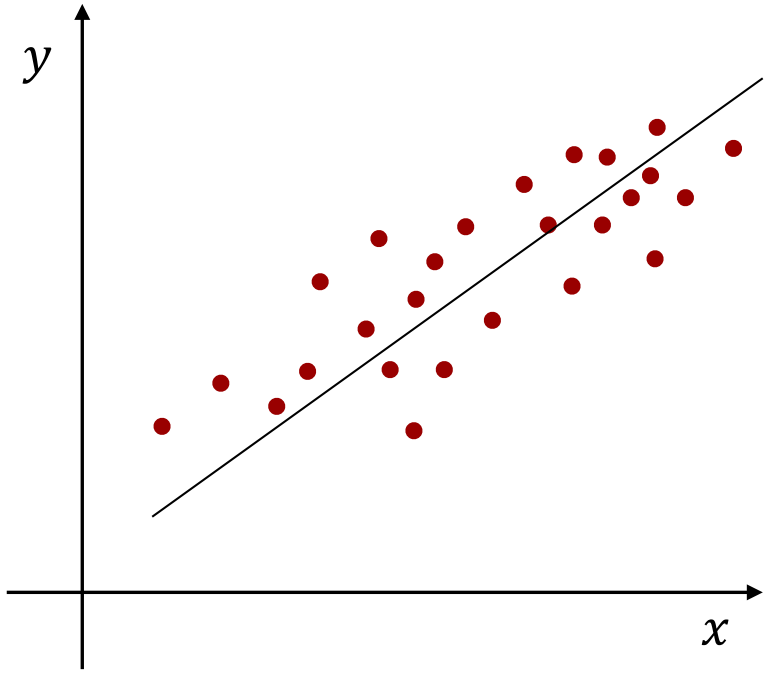


- Ideally we want to <u>enter all our data</u> in our linear regression algorithm to estimate w_1, w_2
- But, the size of data can be immense!
- That's why we need to split our dataset in batches

Batch size vs Iterations vs Epochs

Example from linear regression;
neural network works similarly

Goal: *estimate* w_1, w_2 *to fit* $\hat{y} = w_1 + w_2 x$



- Ideally we want to <u>enter all our data</u> in our linear regression algorithm to estimate w_1, w_2
- But, the size of data can be immense!
- That's why we need to split our dataset in batches
- Total #data = batch_size * iterations

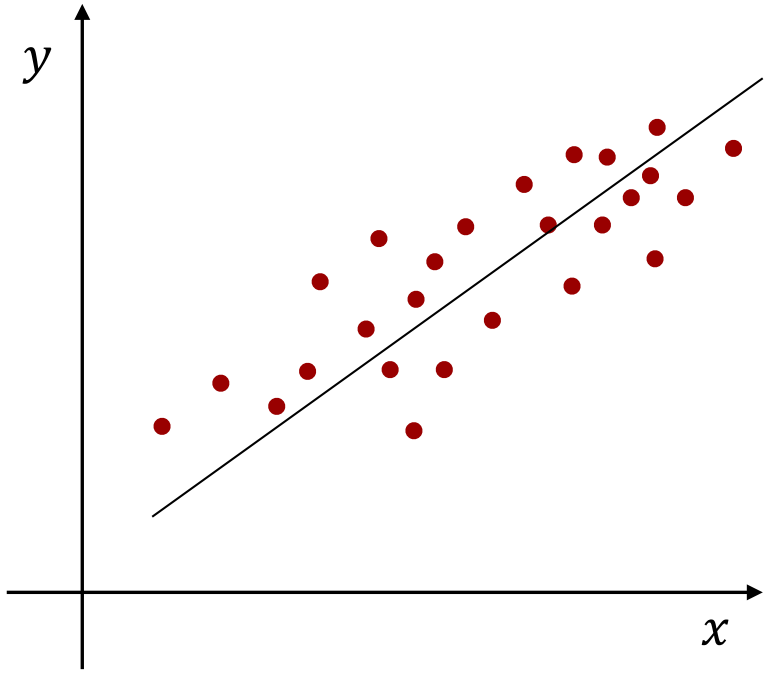
  e.g. if 1000 datapoints and batch_size=200,

  then 1000=200*5 iterations

Batch size vs Iterations vs Epochs

Example from linear regression;
neural network works similarly

Goal: *estimate* w_1, w_2 *to fit* $\hat{y} = w_1 + w_2 x$



More info:
https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

- Ideally we want to enter all our data in our linear regression algorithm to estimate w_1, w_2
- But, the size of data can be immense!
- That's why we need to split our dataset in batches
- Total #data = batch_size * iterations

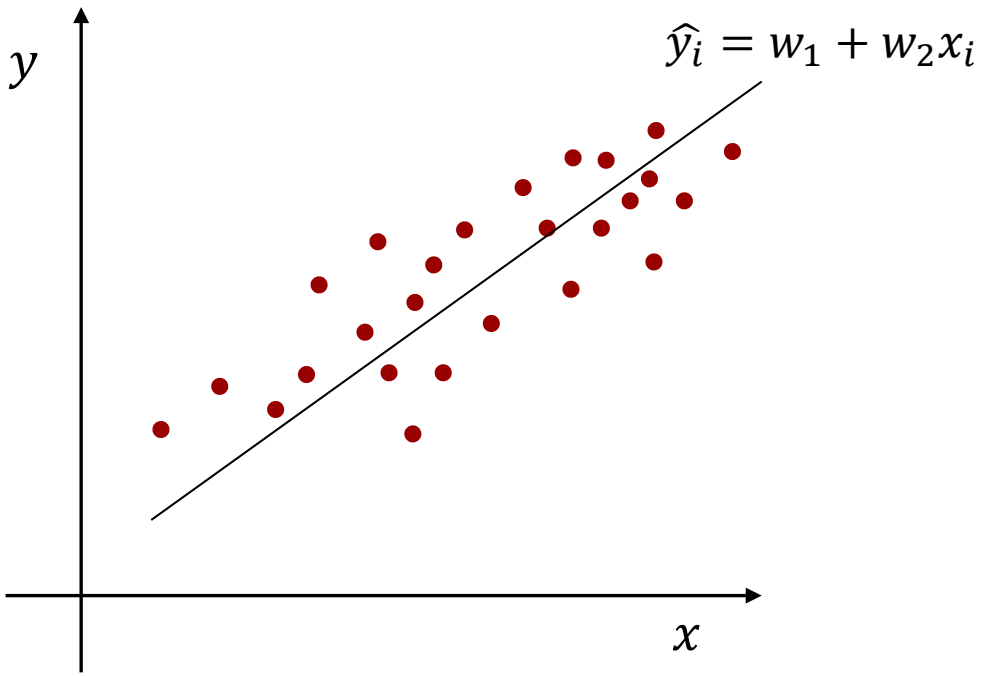
  e.g. if 1000 datapoints and batch_size=200,

  then 1000=200*5 iterations
- Epochs: passing the whole dataset only once is not enough!
  - 1 Epoch = passing the dataset through the neural network ONCE

Loss function:
To find the best estimates for w_1, w_2 we need to run an optimization!

y_i: actual/correct value

$\hat{y}_i$: estimated value



$$\min_{w_1, w_2} \quad \|y_i - \hat{y}_i\|$$

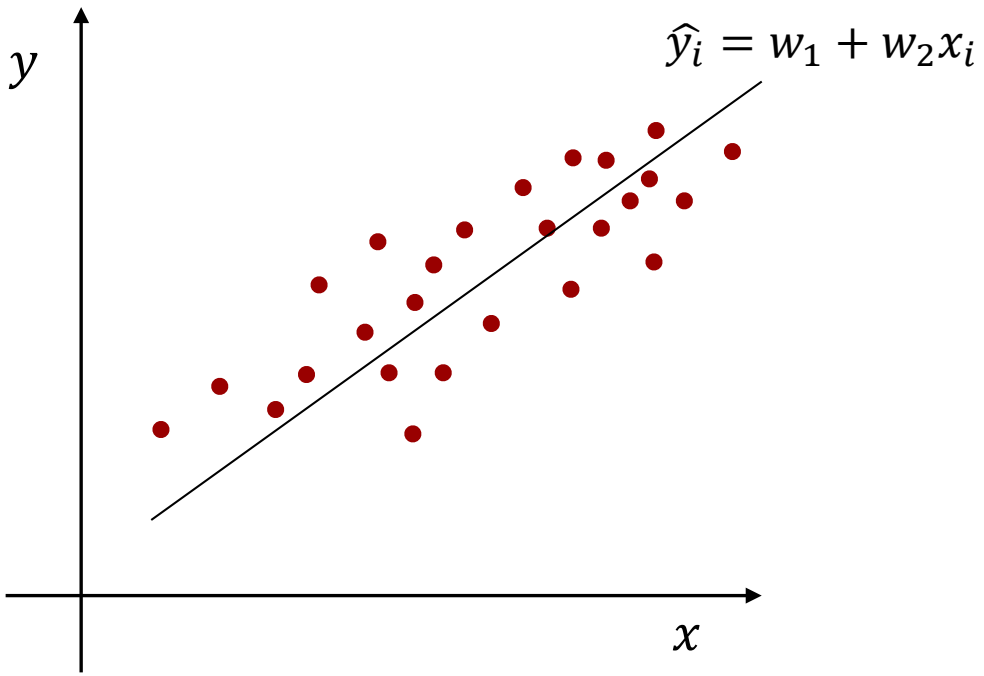
s.t.

$$\hat{y}_i = w_1 + w_2 x_i \qquad \forall i$$

Loss function:
To find the best estimates for w_1, w_2 we need to run an optimization!

y_i: actual/correct value

$\hat{y}_i$: estimated value



$$\min_{w_1, w_2} \quad \|y_i - \hat{y}_i\|$$

s.t.

$$\hat{y}_i = w_1 + w_2 x_i \qquad \forall i$$

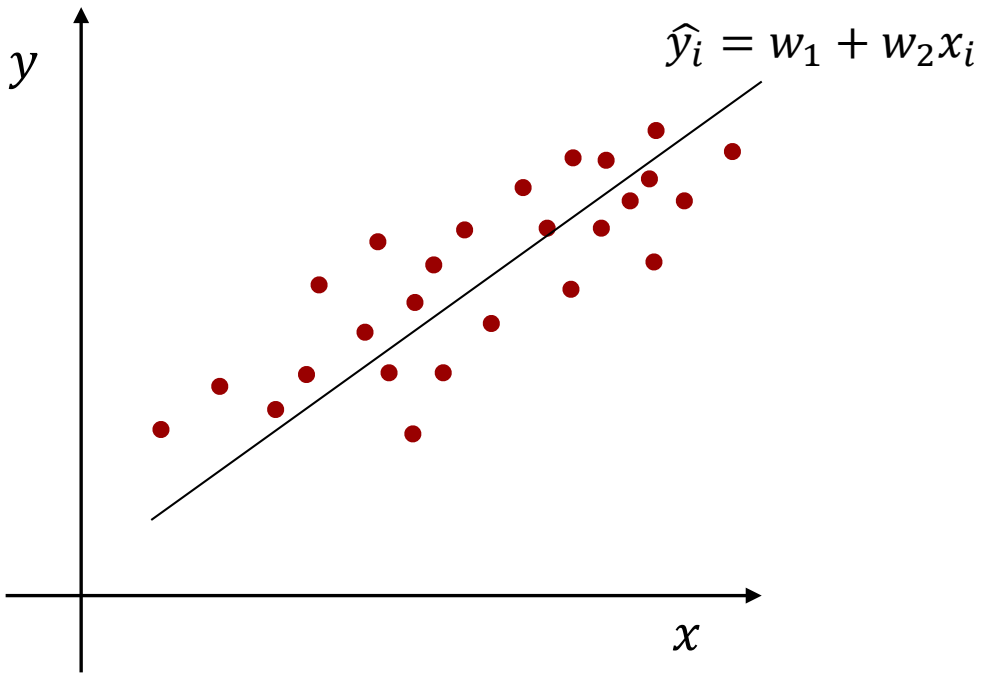
Rewrite:

$$\min_{w_1, w_2} \quad \|y_i - (w_1 + w_2 x_i)\| \quad \forall i$$

Loss function:
To find the best estimates for w_1, w_2 we need to run an optimization!

y_i: actual/correct value

$\hat{y}_i$: estimated value



$$\min_{w_1, w_2} \quad \|y_i - \hat{y}_i\|$$

s.t.

$$\hat{y}_i = w_1 + w_2 x_i \qquad \forall i$$

Rewrite:

$$\min_{w_1, w_2} \quad \|y_i - (w_1 + w_2 x_i)\| \quad \forall i$$

What cost function shall we use?

Cost function (often called "Loss function")

- A series of different options:
  https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
  https://isaacchanghau.github.io/post/loss_functions/

- Often used for classification problems: **cross-entropy**

- For **2 classes**, i.e. $y_i = \{0,1\}$ and $\widehat{y_i} = [0,1]$:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \widehat{y_i} + (1 - y_i)\log(1 - \widehat{y_i})$$

Cross-entropy: the insights

- Insight #1: you minimize over the estimation errors of all points in the batch

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i \; + (1 - y_i)\log(1 - \hat{y}_i)$$

- Insight #2:
  - Actual values: either 0 or 1
  - Estimated values: between 0 and 1

$$y_i = \{0,1\} \quad but \; \hat{y}_i = [0,1]$$

Cross-entropy: the insights

- Insight #1: you minimize over the estimation errors of all points in the batch

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i \ + (1 - y_i)\log(1 - \hat{y}_i)$$

- Insight #2:
  - Actual values: either 0 or 1
  - Estimated values: between 0 and 1

$$y_i = \{0,1\} \quad but \ \hat{y}_i = [0,1]$$

- Insight #3: Why the minus?

Hint: explain it for one point

$$\mathcal{L} = -(y_i \log \hat{y}_i \ + (1 - y_i)\log(1 - \hat{y}_i))$$

Cross-entropy: the insights

- Insight #1: you minimize over the estimation errors of all points in the batch

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i \ + (1 - y_i)\log(1 - \hat{y}_i)$$

- Insight #2:
  - Actual values: either 0 or 1
  - Estimated values: between 0 and 1

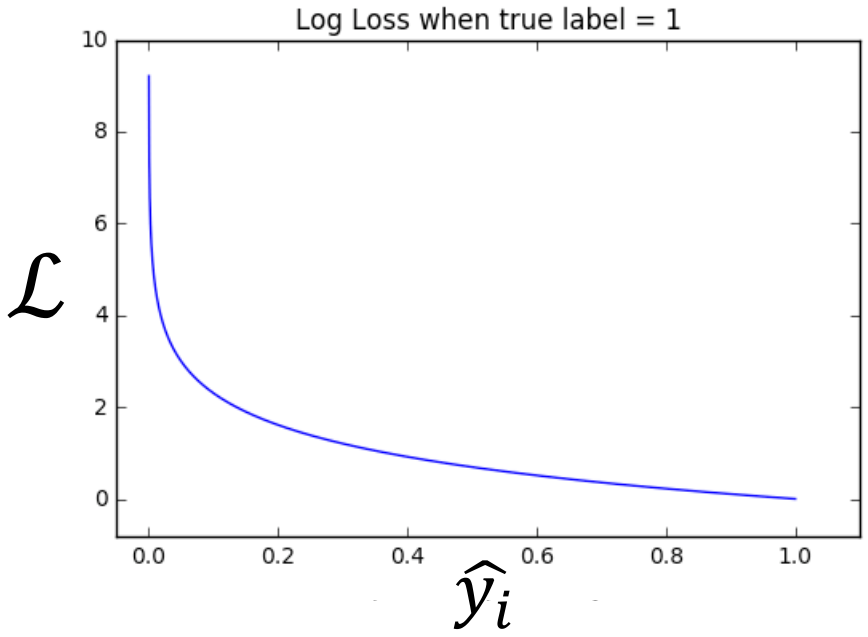
$$y_i = \{0,1\} \quad but \ \hat{y}_i = [0,1]$$

- Insight #3: Why the minus?

Hint: explain it for one point

$$\mathcal{L} = -(y_i \log \hat{y}_i \ + (1 - y_i)\log(1 - \hat{y}_i))$$

- Insight #4: y_i=1, $\mathcal{L}$ *when* $\hat{y}_i = [0,1]$



Evaluating the performance: Confusion matrix

		Target class (actual values)	
		1	0
Output class (predicted values)	1	True positive (TP)	False positive (FP)
	0	False negative (FN)	True Negative (TN)

Evaluating the performance: Confusion matrix

		Target class (actual values)	
Output class (predicted values)		1	0
	1	True positive (TP)	False positive (FP)
	0	False negative (FN)	True Negative (TN)

Classification for power system security

	Actually Safe	Actually Unsafe
Predicted safe	True positive (TP)	False positive (FP)
Predicted Unsafe	False negative (FN)	True Negative (TN)

Performance Metrics: Accuracy

	Actually Safe	Actually Unsafe
Predicted safe	True positive (TP)	False positive (FP)
Predicted Unsafe	False negative (FN)	True Negative (TN)

- Accuracy: The proportion of correct classifications in the whole dataset

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

- **Example:** Assume 1000 datapoints
  - Actually safe: 500 TP=480 FP=30
  - Actually unsafe: 500 FN=20 TN=470

Accuracy = ?

Performance Metrics: Accuracy

	Actually Safe	Actually Unsafe
Predicted safe	True positive (TP)	False positive (FP)
Predicted Unsafe	False negative (FN)	True Negative (TN)

- Accuracy: The proportion of correct classifications in the whole dataset

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

- **Example:** Assume 1000 datapoints
  - Actually safe: 500 TP=480 FP=30
  - Actually unsafe: 500 FN=20 TN=470

$$\text{Accuracy} = \frac{480+470}{480+20+470+30} = 95\%$$

Evaluating performance by measuring only accuracy is often not enough

Why?

Performance Metrics: Accuracy

	Actually Safe	Actually Unsafe
Predicted safe	True positive (TP)	False positive (FP)
Predicted Unsafe	False negative (FN)	True Negative (TN)

- Accuracy: The proportion of correct classifications in the whole dataset

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

- **Example:** Assume 1000 datapoints
  - Actually safe: 500 TP=480 FP=30
  - Actually unsafe: 500 FN=20 TN=470

$$\text{Accuracy} = \frac{480+470}{480+20+470+30} = 95\%$$

Evaluating performance by measuring only accuracy is often not enough

Why?

- **Example:** Assume 1000 datapoints
  - Actually safe: 20 TP= 1 FP=30
  - Actually unsafe: 980 FN= 19 TN=950

Accuracy = ?

Performance Metrics: Accuracy

	Actually Safe	Actually Unsafe
Predicted safe	True positive (TP)	False positive (FP)
Predicted Unsafe	False negative (FN)	True Negative (TN)

- Accuracy: The proportion of correct classifications in the whole dataset

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

- **Example:** Assume 1000 datapoints
  - Actually safe: 500 TP=480 FP=30
  - Actually unsafe: 500 FN=20 TN=470

$$\text{Accuracy} = \frac{480+470}{480+20+470+30} = 95\%$$

Evaluating performance by measuring only accuracy is often not enough

Why?

- **Example:** Assume 1000 datapoints
  - Actually safe: 20 TP= 1 FP=30
  - Actually unsafe: 980 FN= 19 TN=950

$$\text{Accuracy} = \frac{1+950}{1+19+950+30} = 95\%$$

- *95% accurate but we have misclassified almost all truly safe points!*
- *For heavily unbalanced data, accuracy is not sufficient!*

Performance metrics

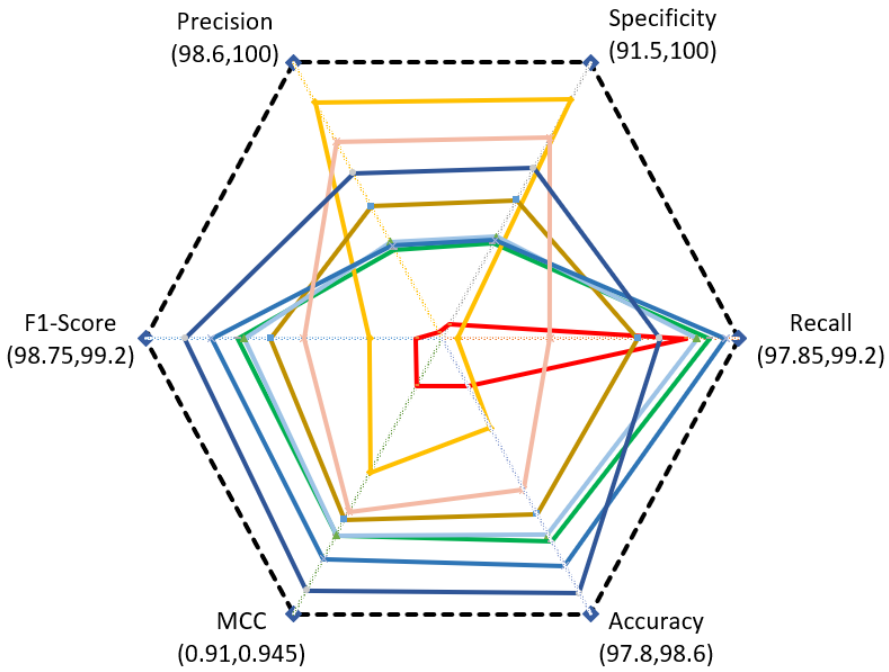
- Accuracy
- Recall: True Positive Rate $\frac{TP}{TP+FN}$
- Specificity: True Negative Rate $\frac{TN}{TN+FP}$
- Precision: Positive Predictive Value $\frac{TP}{TP+FP}$
- F1: harmonic mean of Precision and Recall $\mathrm{F1} = \frac{Precision \cdot Recall}{Precision+Recall}$
- **MCC (Matthews correlation coefficient)**

  (only for binary classification – 2 classes only)

  – MCC=1 → perfect prediction

  – MCC=0 → random (like flipping a coin)

  – MCC= –1→ Completely mistaken



Hidalgo-Arteaga, Hancharou, Thams, Chatzivasileiadis, Powertech 2019

$$\mathrm{MCC} = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Key hints for your implementation

- Regularization: Training of neural networks work better if you normalize your inputs
  - Try to normalize your active power setpoints (e.g. if PG1 =30 MW and PG1max = 100 MW, then PG1=0.3)

- 1-hot encoding: Neural networks work better if you use *one vector for each class*

Instead of:

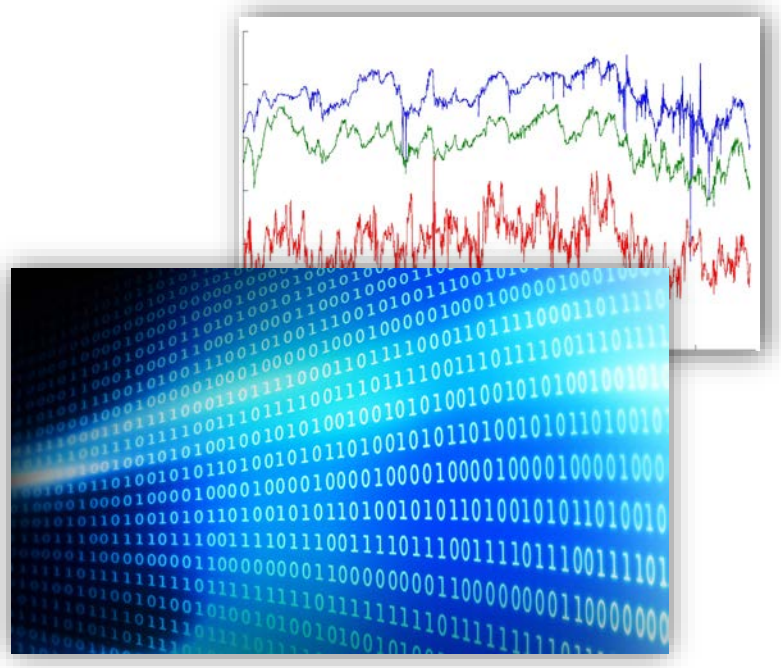
	Safe=1 Unsafe=0
x_1	0
x_2	1
x_3	0

	Safe	Unsafe
x_1	0	1
x_2	1	0
x_3	0	1

Training database

We need data!

- Data that accurately capture the whole security region
  - so that we can successfully use machine learning approaches for classification
- Historical data are insufficient
  - Contain very limited number of abnormal situations
- Need to generate simulation data
- **Assessing the stability of 100’000s of operating points is an extremely demanding task**
  - **Immense search space**



Efficient Database Generation

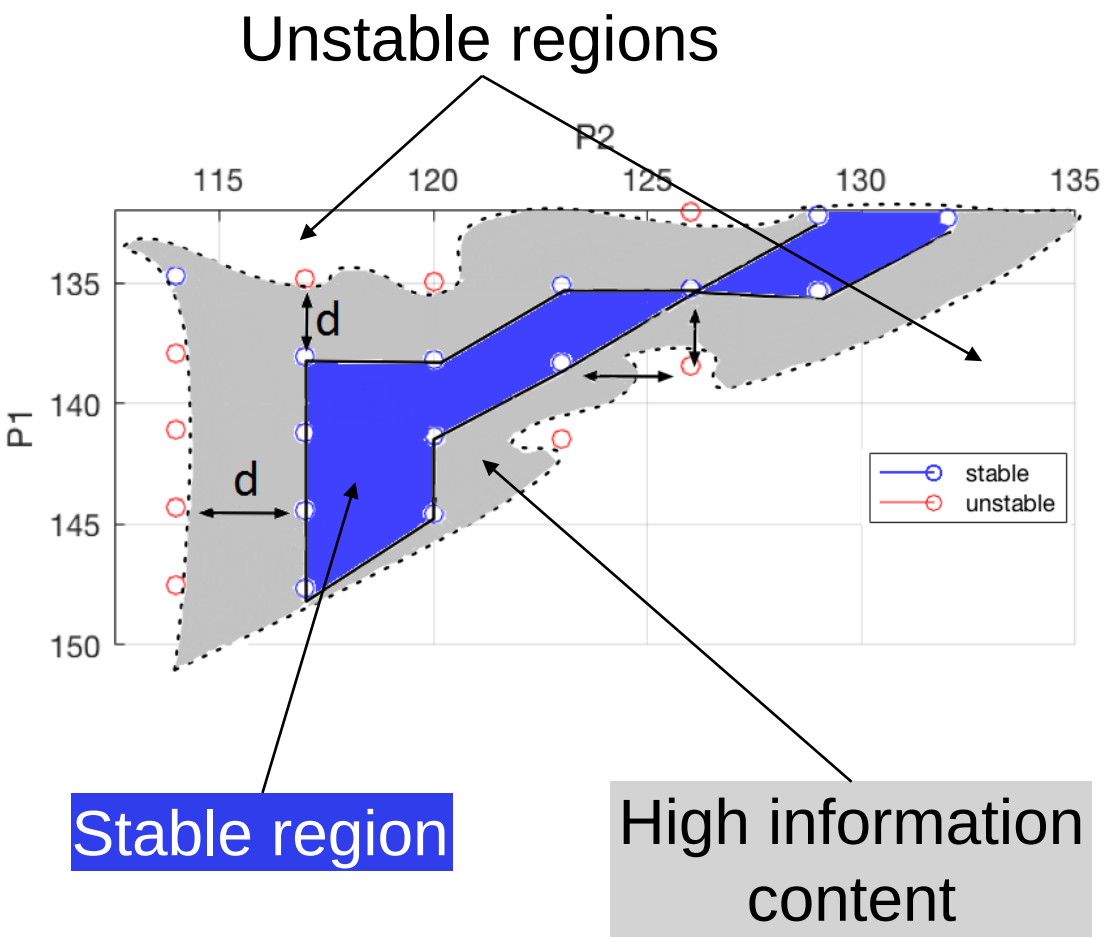
- Modular and highly efficient algorithm
- Can accommodate numerous definitions of power system security (e.g. N-1, N-k, small-signal stability, voltage stability, transient stability, **or a combination** of them)
- **10-20 times faster** than existing state-of-the-art approaches
- Our use case: N-1 security + small-signal stability
- Generated Database for NESTA 162-bus system online available!
  https://github.com/johnnyDEDK/OPs_Nesta162Bus (>500,000 points)

F. Thams, A. Venzke, R. Eriksson, and S. Chatzivasileiadis, ”Efficient database generation for data-driven security assessment of power systems”. Accepted in IEEE Trans. Power Systems, 2019. https://www.arxiv.org/abs/1806.0107.pdf

Efficient Database Generation

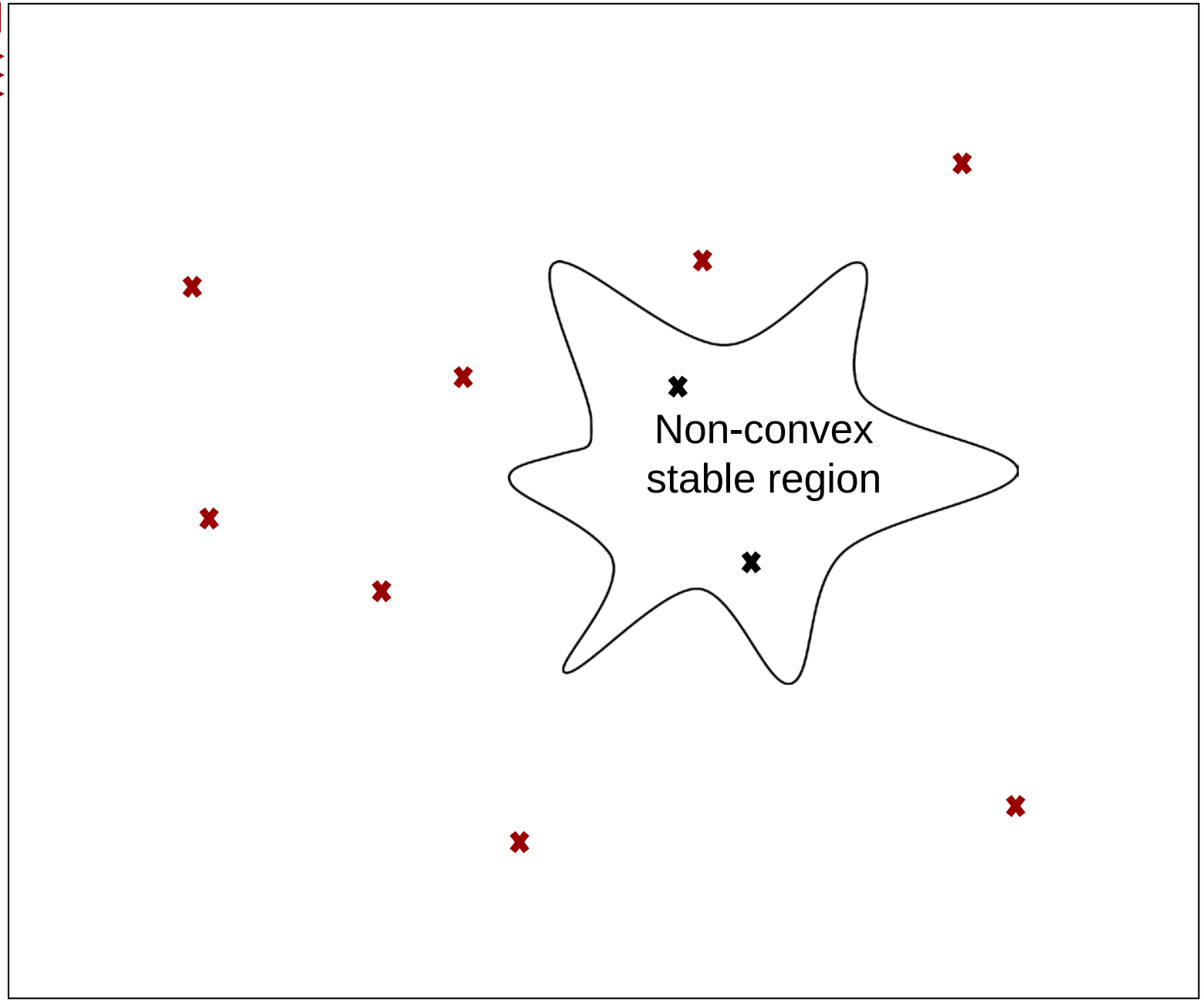
- The goal
  - **Focus** on the **boundary between stability and instability**
  - We call it: “high information content” region

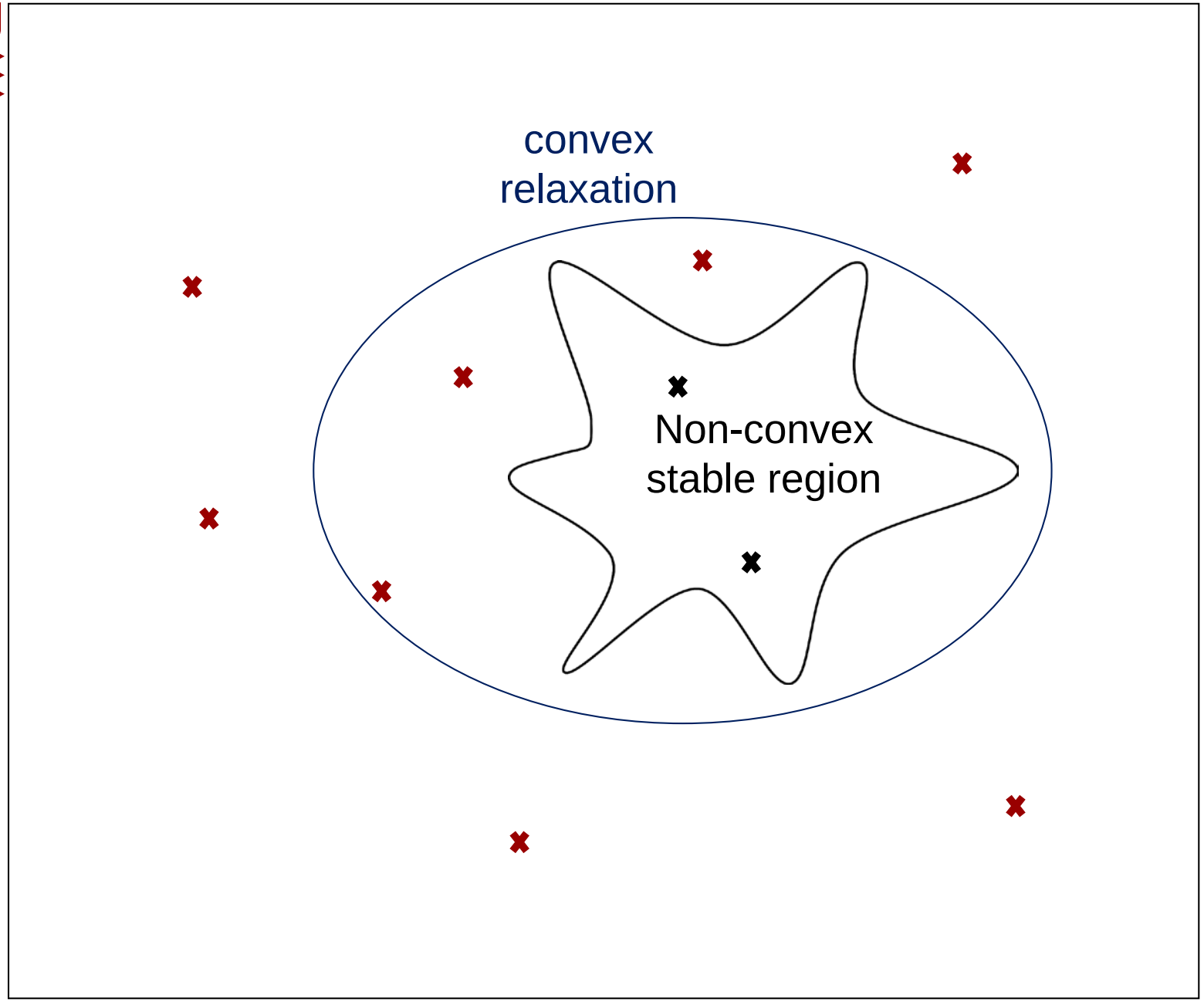
- How?
  1. Using convex relaxations
  2. And “Directed Walks”



Real data for the IEEE 14-bus system
N-1 security and small-signal stability

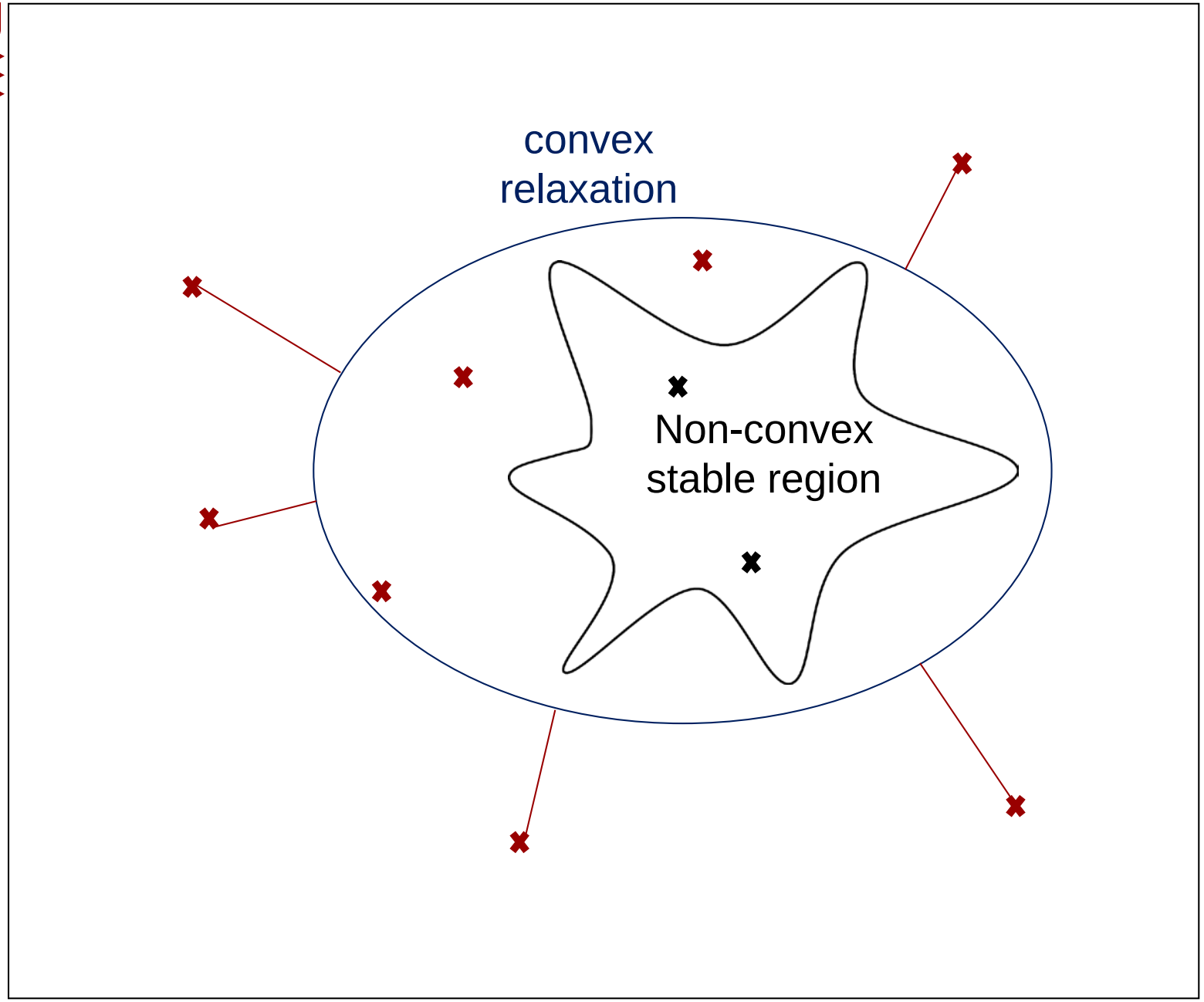
Convex relaxations to discard infeasible regions





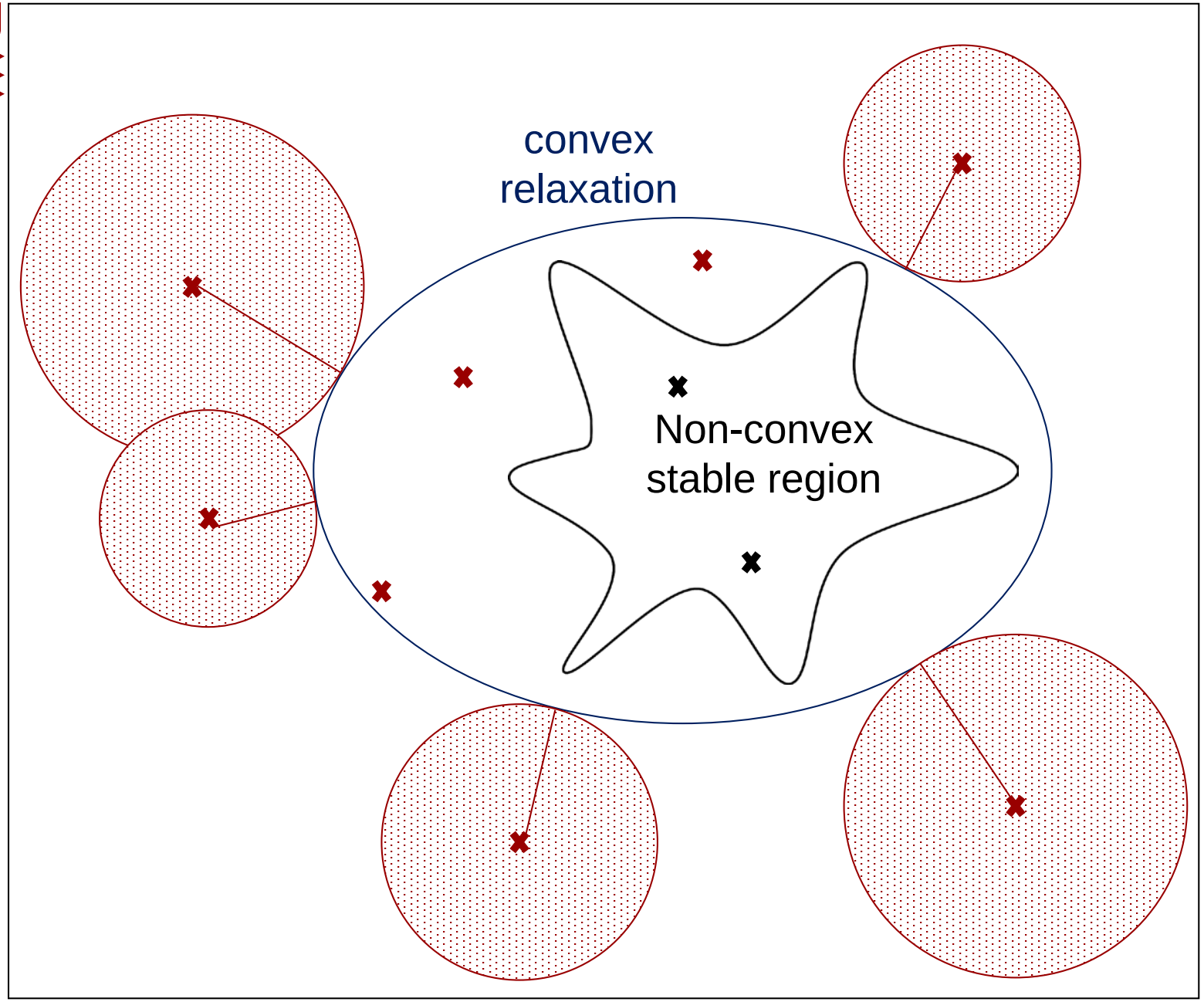
Convex relaxations to discard infeasible regions

- Certificate: if point infeasible for semidefinite relaxation → infeasible for the original problem



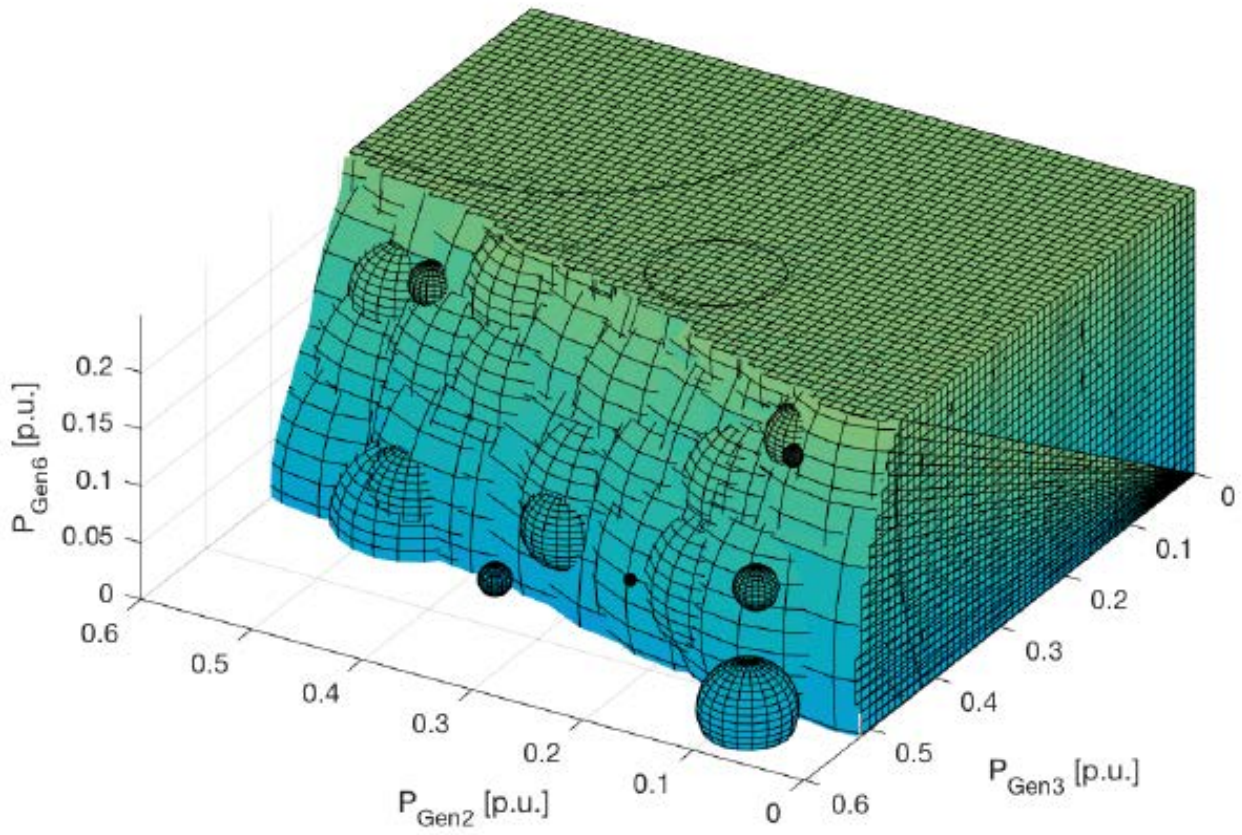
Convex relaxations to discard infeasible regions

- Certificate: if point infeasible for semidefinite relaxation → infeasible for the original problem
- If infeasible point: find minimum radius to feasibility



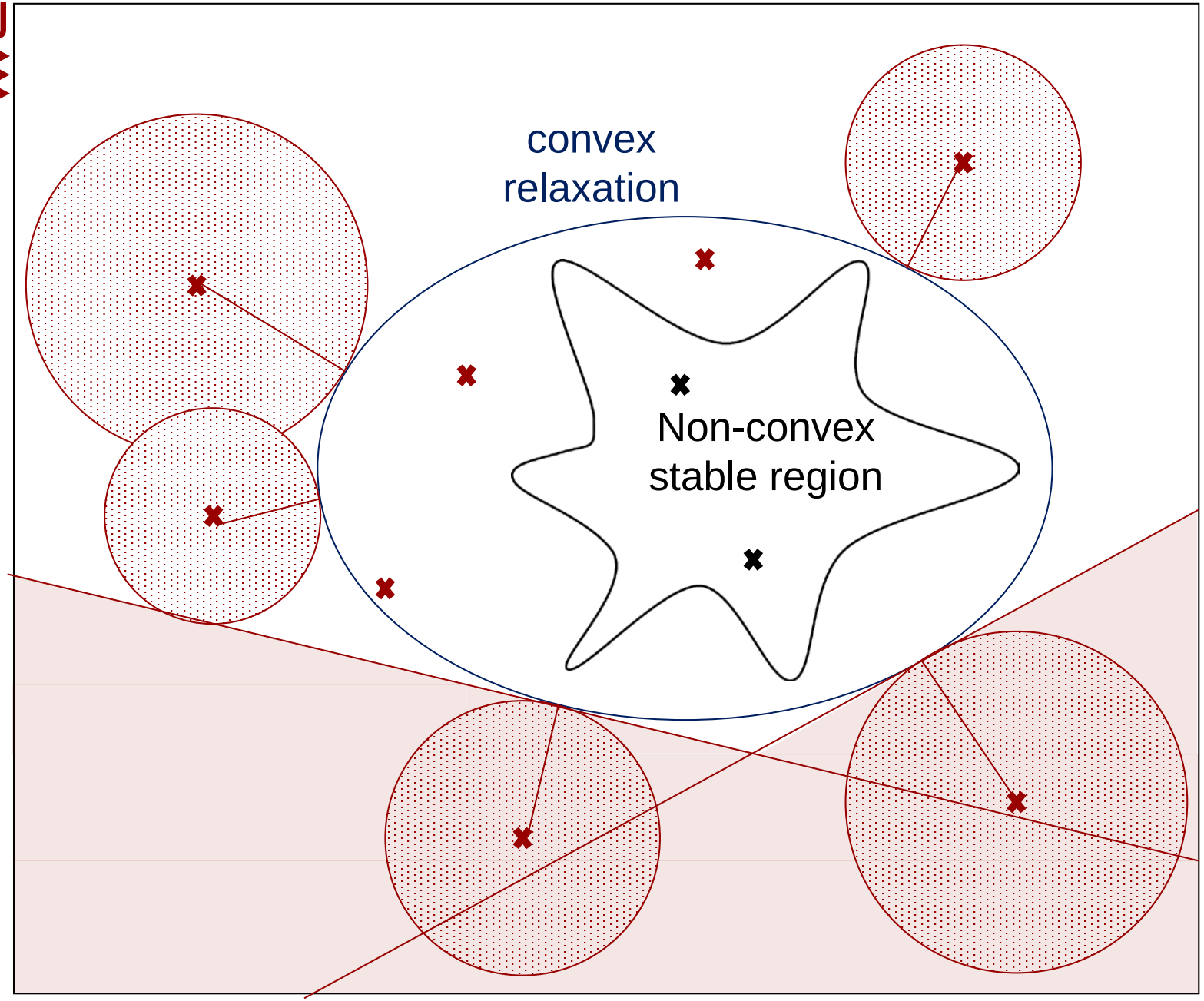
Convex relaxations to discard infeasible regions

- Certificate: if point infeasible for semidefinite relaxation → infeasible for the original problem
- If infeasible point: find minimum radius to feasibility
- Discard all points inside the (hyper)sphere



- 3D projection of hyperspheres
- IEEE 14-bus system
- Rapidly discarding (=classifying) large chunks of the search space as infeasible to focus on the boundary

F. Thams, A. Venzke, R. Eriksson, and S. Chatzivasileiadis, "Efficient database generation for data-driven security assessment of power systems". Accepted in IEEE Trans. Power Systems, 2019. https://www.arxiv.org/abs/1806.0107.pdf



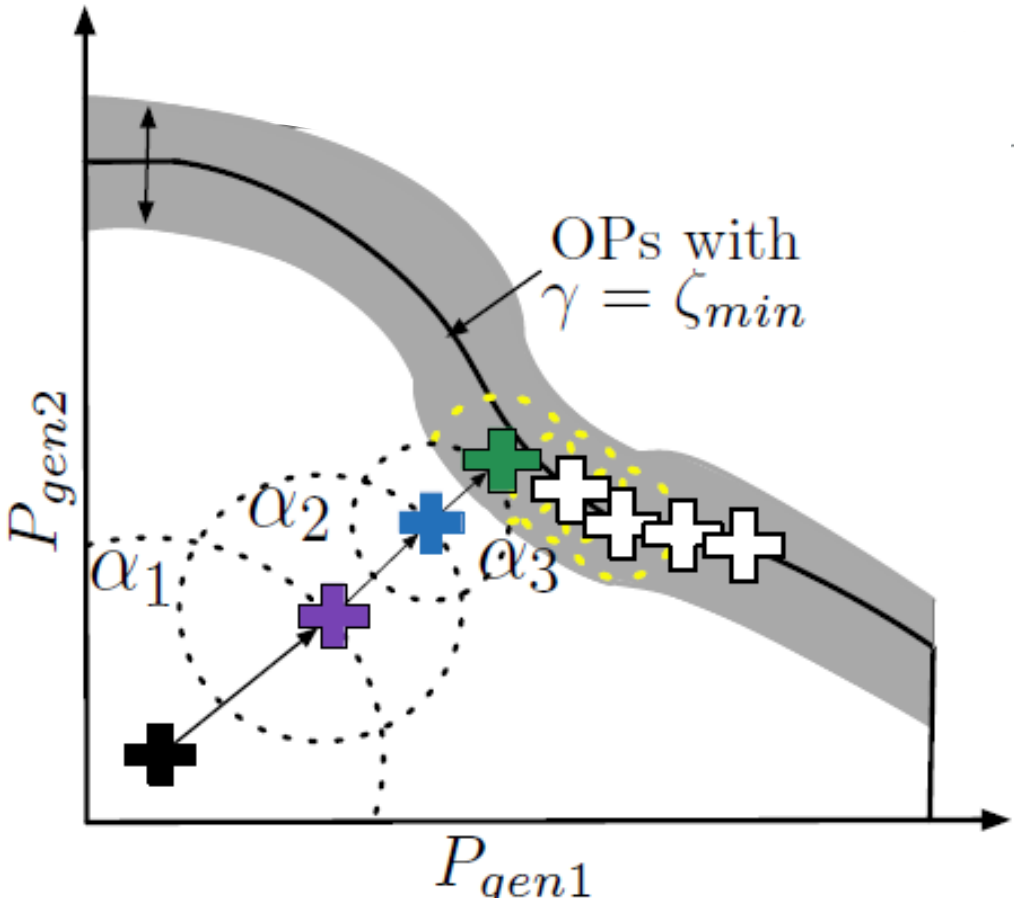
Convex relaxations to discard infeasible regions

- Currently working on hyperplanes
- Paper will appear soon

Directed Walks

- "Directed walks": steepest-descent based algorithm to explore the remaining search space, focusing on the area around the security boundary

1. Variable step-size
2. Parallel computation
3. Full N-1 contingency check



Results

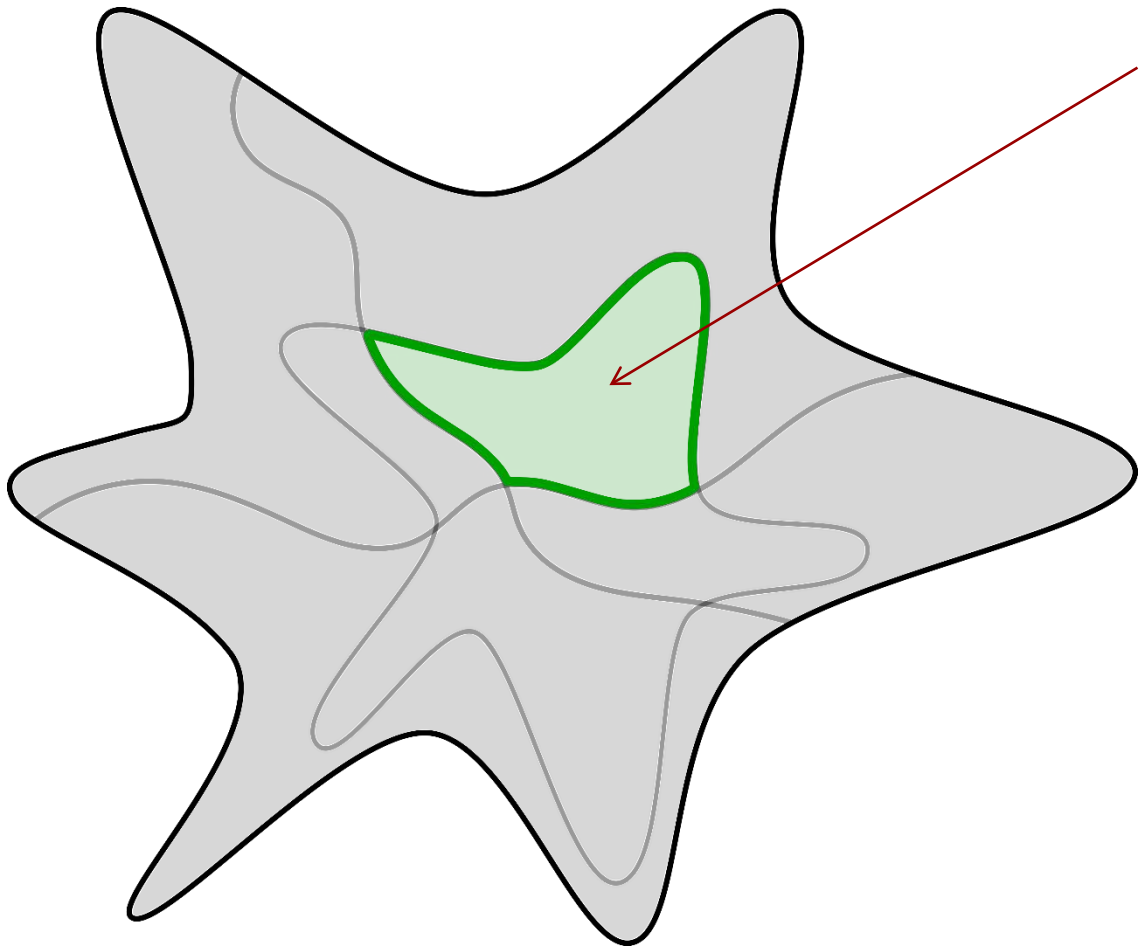
	Points close to the security boundary (within distance γ)	
	IEEE 14-bus	NESTA 162-bus
Brute Force	100% of points in **556.0 min**	*intractable*
Importance Sampling	100% of points in **37.0 min**	**901 points** in 35.7 hours
Proposed Method	100% of points in **3.8 min**	**183’295 points** in 37.1 hours

- Further benefits for the decision tree:
  - Higher accuracy
  - Better classification quality (Matthews correlation coefficient)

Generated Database for NESTA 162-bus system online available!
https://github.com/johnnyDEDK/OPs_Nesta162Bus

From decision trees to mixed integer linear programming

The feasible space of power system operations



Intersection of all security/stability criteria: Non-linear and non-convex security region

Electricity markets should determine setpoints that are only within this area

Optimization constraints should represent this area

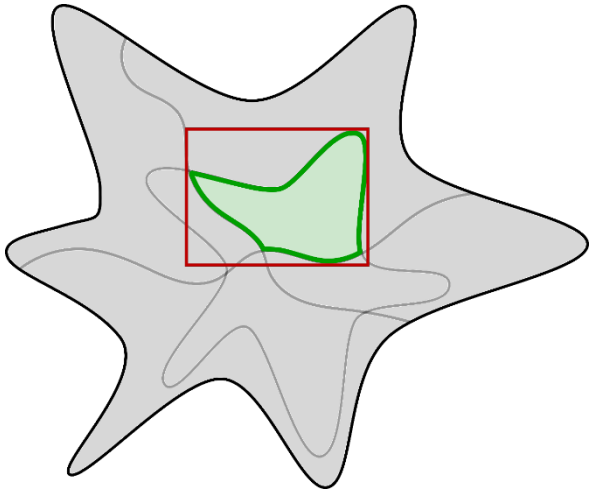
Impossible → differential and non-linear algebraic equations

What do TSOs and market operators do?

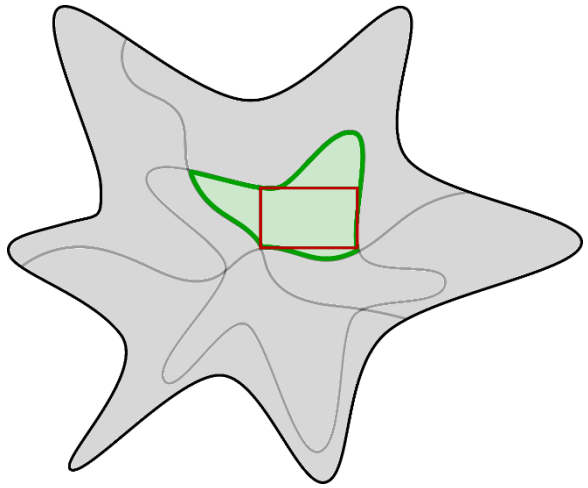
Linear approximations

Net Transfer Capacity

Inaccurate

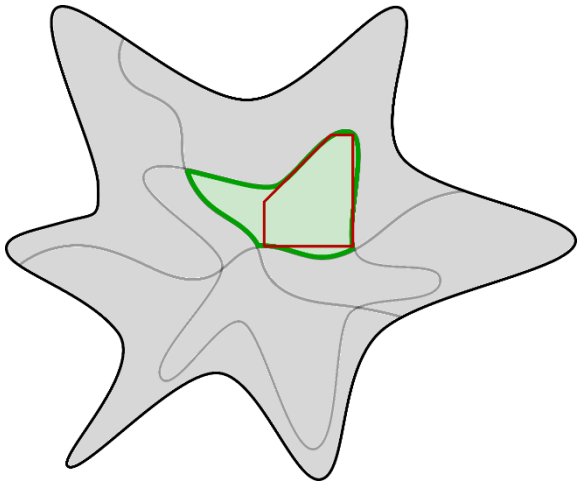


Too conservative

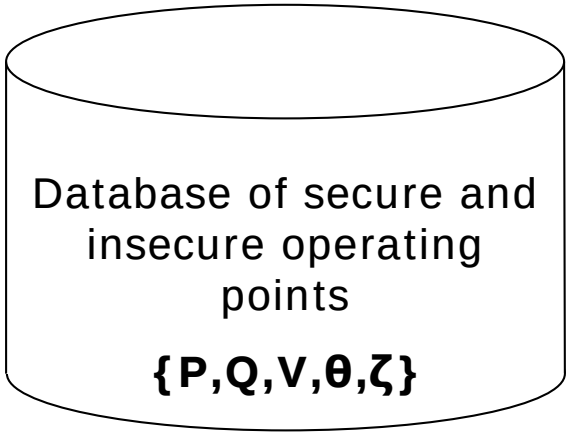


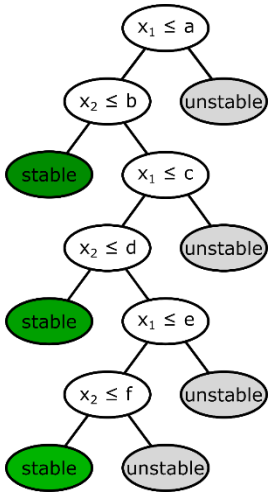
Flow-based market coupling

Single convex region



Our proposal: Data-driven Security Constrained OPF
How does it work?





$$\mathbf{PTDF} \cdot \left(\mathbf{P_G} - \mathbf{P_D}\right) \leq \mathbf{F}^{\max}_{\mathbf{L,p}} y_p + \mathbf{F}^{\max}_{\mathbf{L}}(1 - y_p)$$
$$\mathbf{PTDF} \cdot \left(\mathbf{P_G} - \mathbf{P_D}\right) \geq \mathbf{F}^{\min}_{\mathbf{L,p}} y_p - \mathbf{F}^{\max}_{\mathbf{L}}(1 - y_p)$$

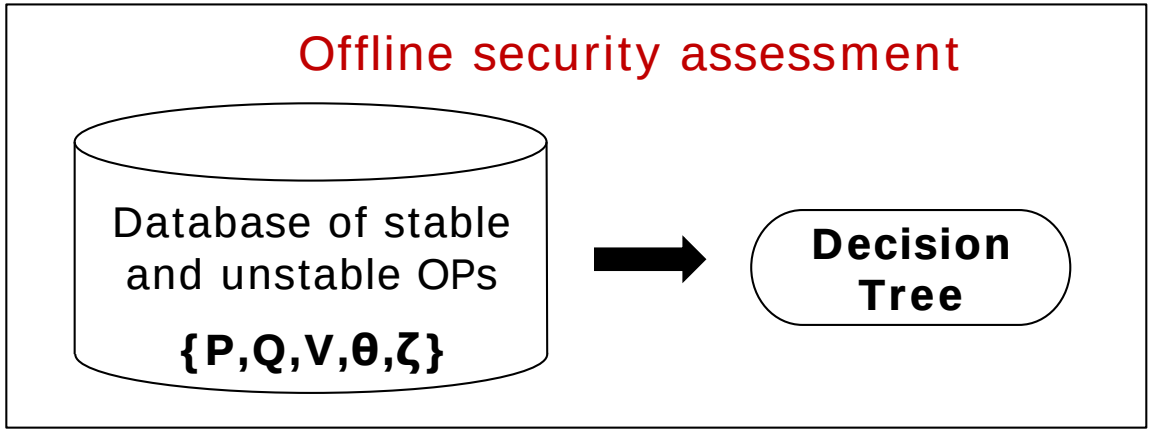
Operating points provided by the TSOs through simulated and real data

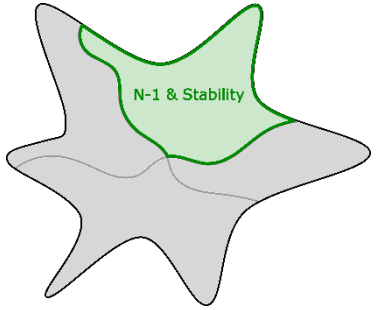
Train a decision tree to classify secure and insecure regions

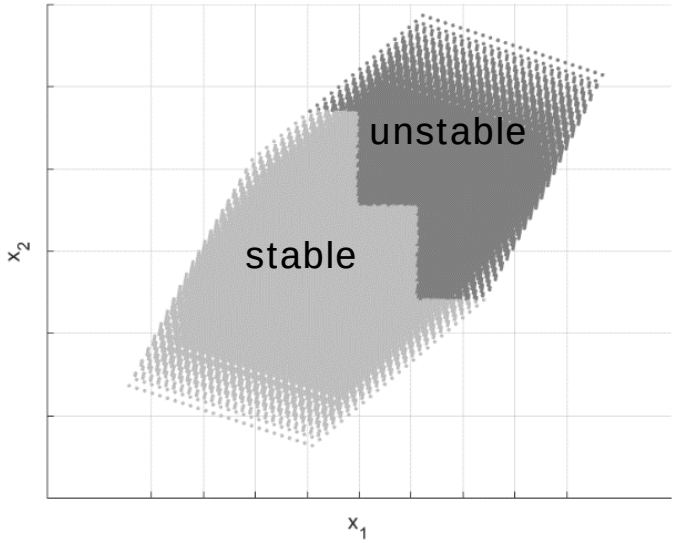
Exact reformulation to MILP

Thams et al IREP 2017,
Halilbasic et al PSCC 2018

Data-driven security-constrained OPF

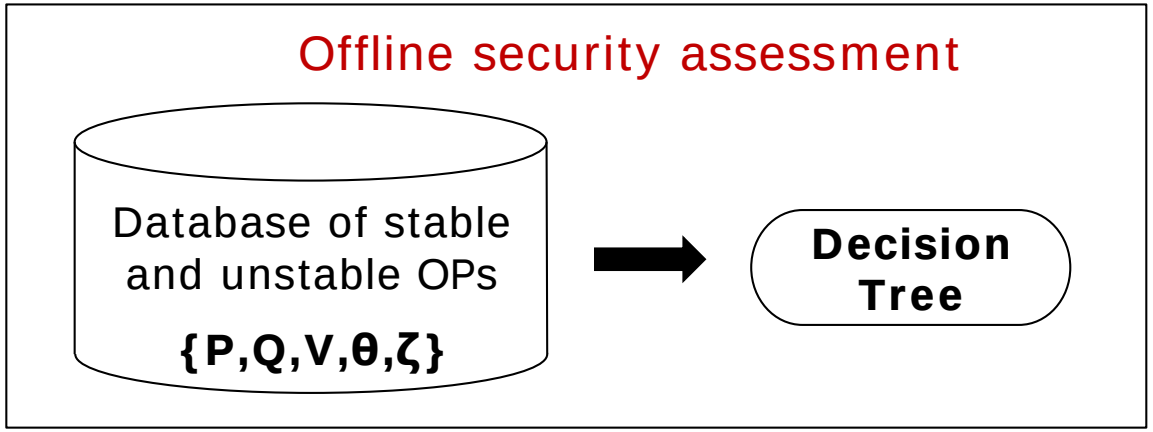


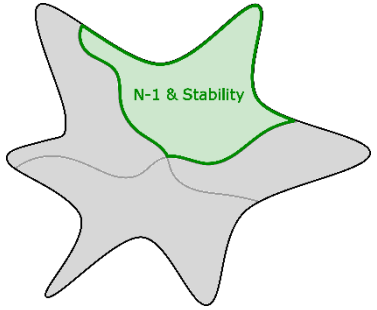




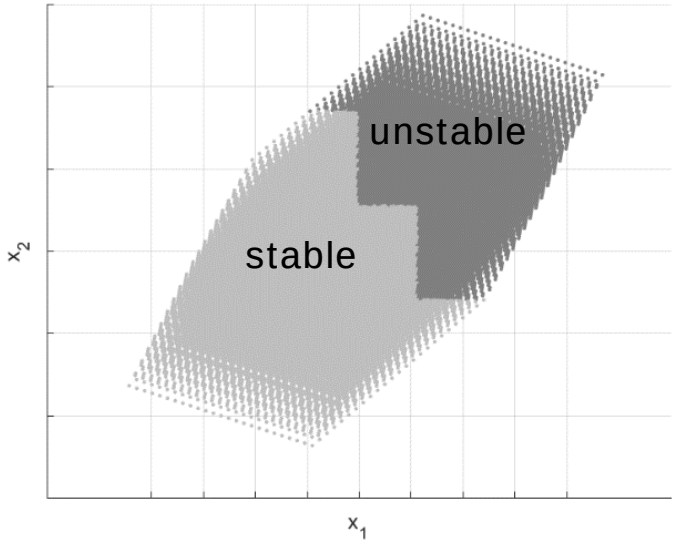
Thams et al IREP 2017,
Halilbasic et al PSCC 2018

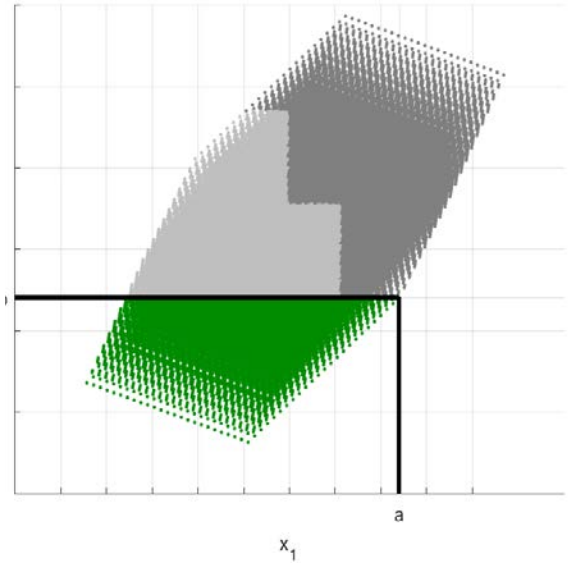
Data-driven security-constrained OPF

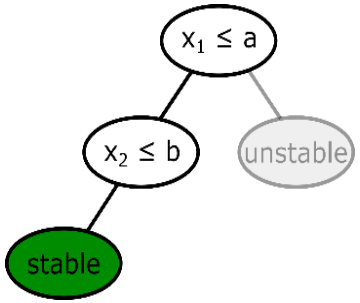




Partitioning the secure operating region

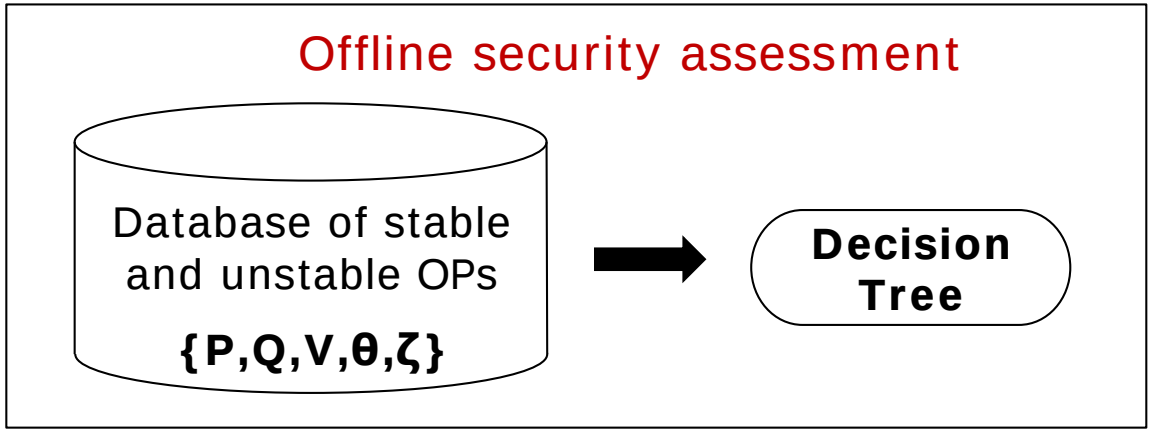


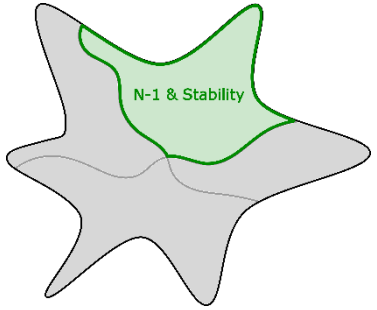




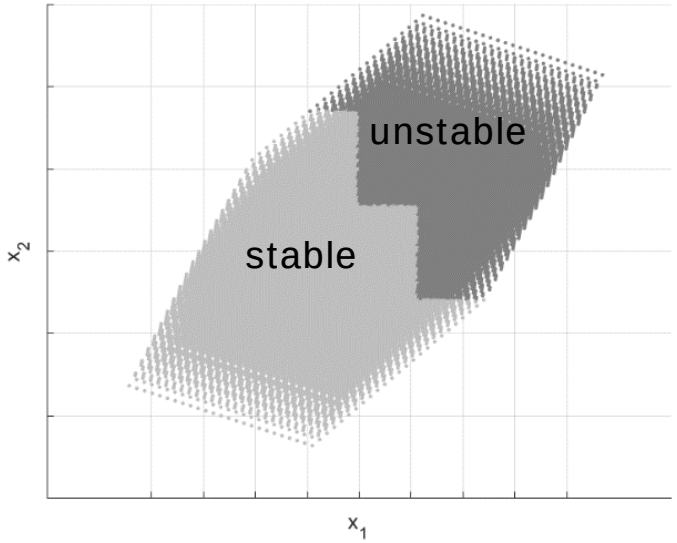
Thams et al IREP 2017,
Halilbasic et al PSCC 2018

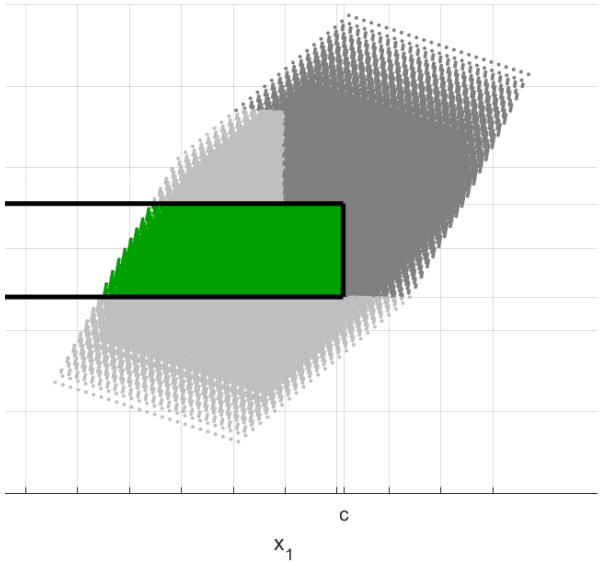
Data-driven security-constrained OPF

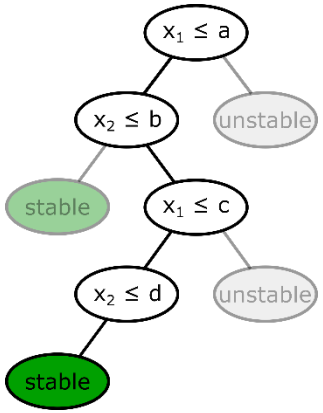




Partitioning the secure operating region

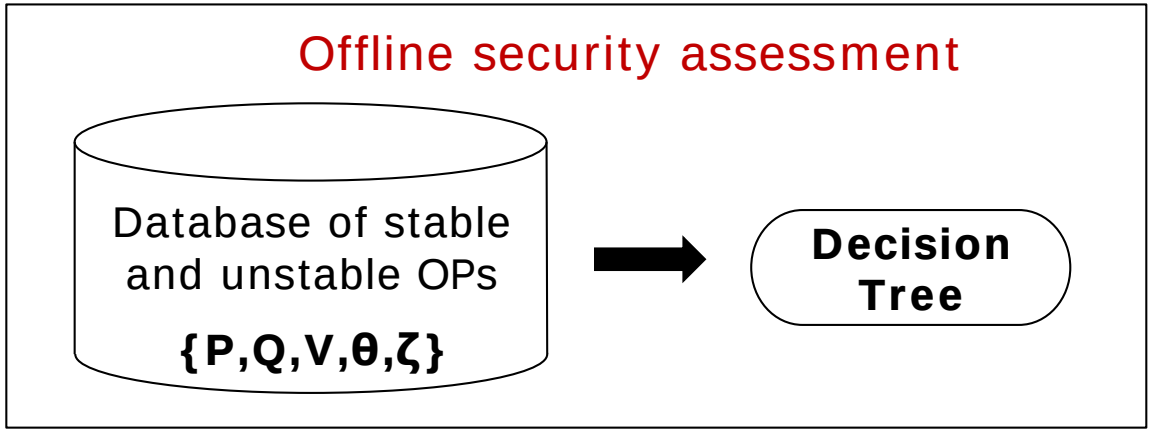


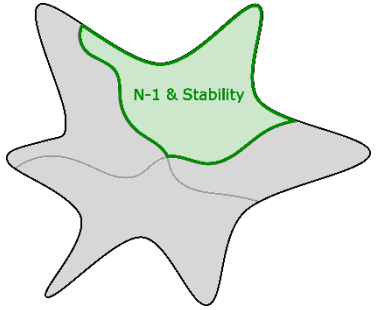




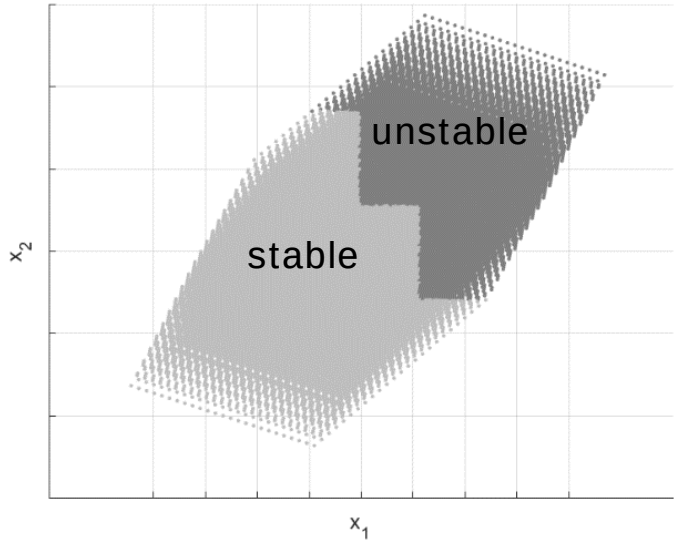
Thams et al IREP 2017,
Halilbasic et al PSCC 2018

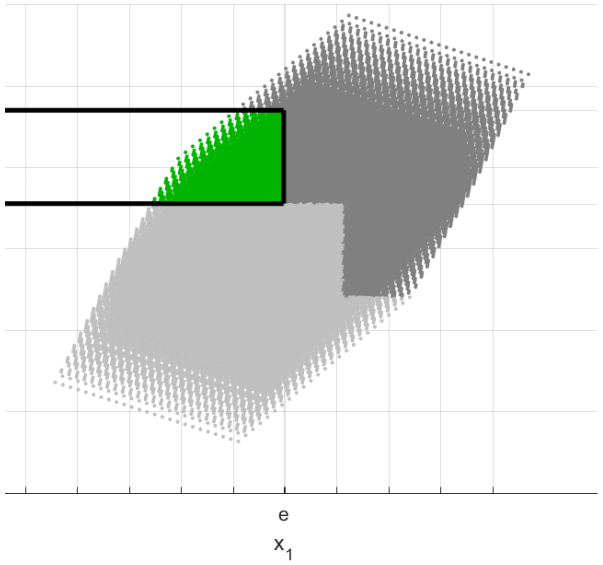
Data-driven security-constrained OPF

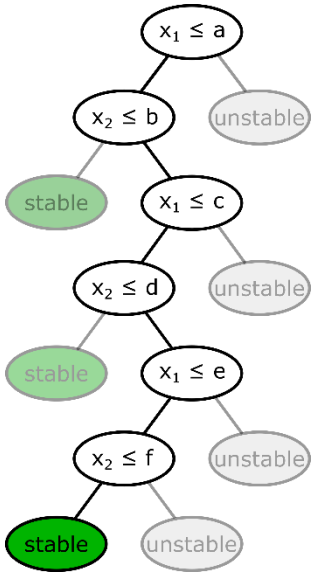




Partitioning the secure operating region







Thams et al IREP 2017,
Halilbasic et al PSCC 2018

Data-driven security-constrained OPF

Offline security assessment

Database of stable and unstable OPs

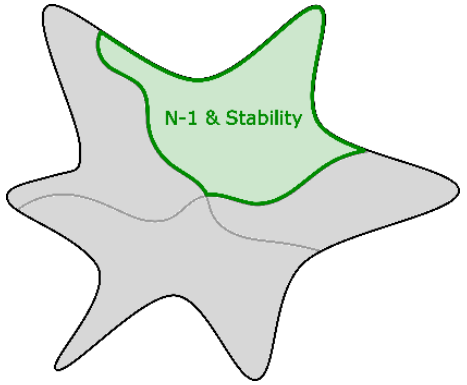
{P,Q,V,θ,ζ}

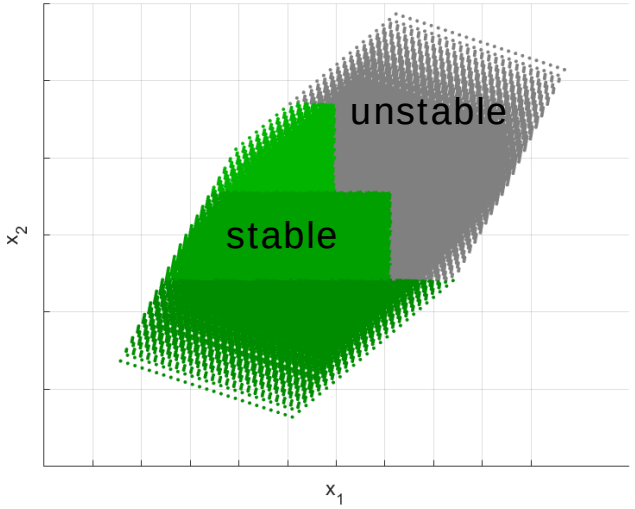
→ **Decision Tree**

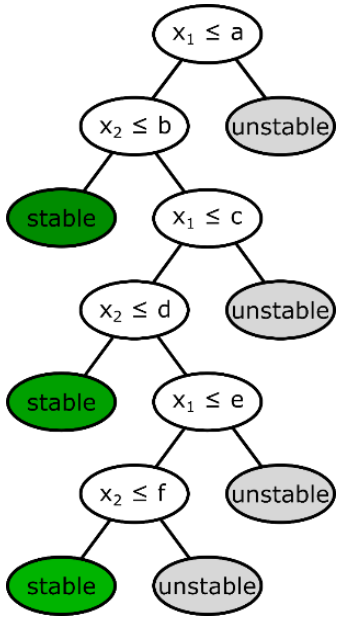
Optimization

Integer Programming to incorporate partitions (DT)

- DC-OPF (MILP)
- AC-OPF (MINLP)
- Relaxation (MIQCP, MISOCP)



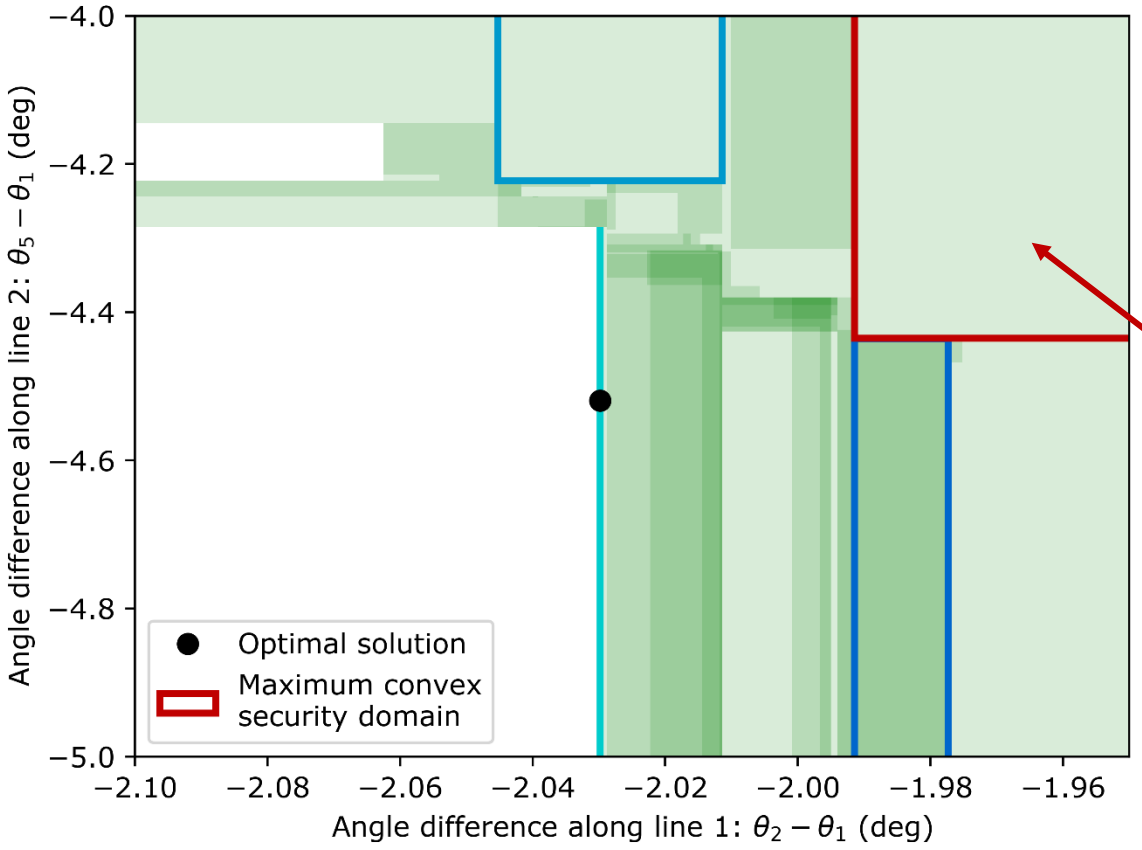


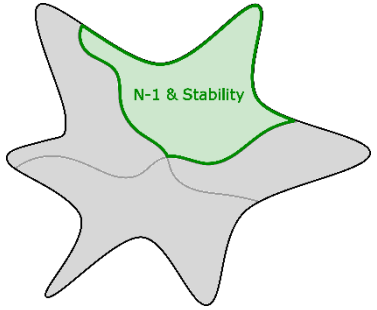


- Each leaf is a convex region
- FBMC corresponds to the leaf that maps the largest convex region

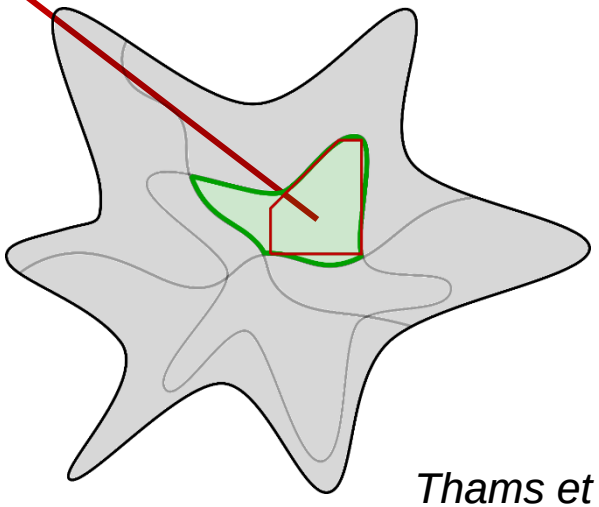
Thams et al IREP 2017,
Halilbasic et al PSCC 2018

We gain ~22% of the feasible space using data and Mixed Integer Programming



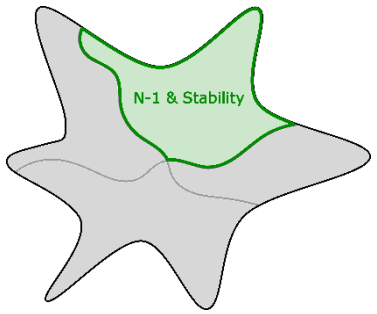


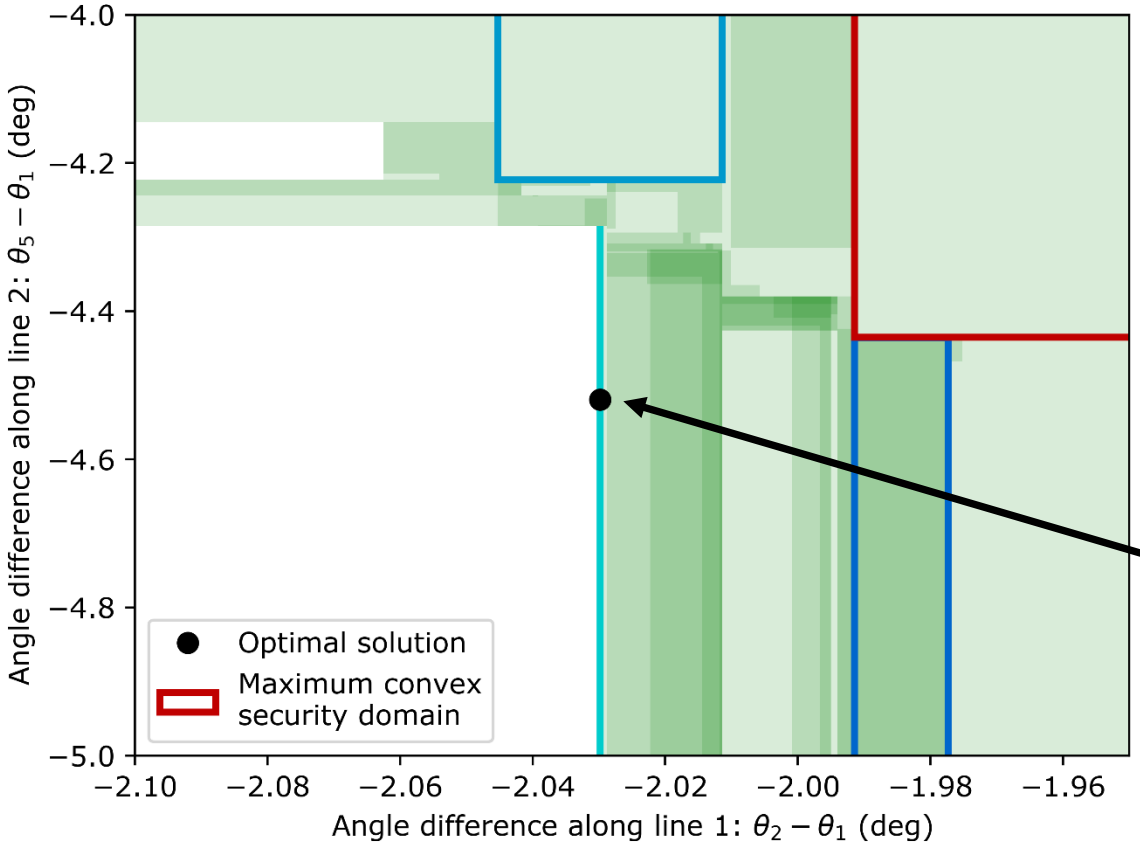
Largest convex region covers ~78%



Thams et al IREP 2017, Halilbasic et al PSCC 2018

MIP + convex AC-OPF approximation finds better solutions than nonconvex problem!





Optimum located at boundary of considered security region

Thams et al IREP 2017,
Halilbasic et al PSCC 2018

Thank you!

spchatz@elektro.dtu.dk

References:

L. Halilbašić, F. Thams, A. Venzke, S. Chatzivasileiadis, and P. Pinson, ”Data-driven security-constrained AC-OPF for operations and markets,” in *2018 Power Systems Computation Conference (PSCC)*, 2018. [.pdf]

F. Thams, L. Halilbašić, P. Pinson, S. Chatzivasileiadis, and R. Eriksson, ”Data-driven security-constrained OPF,” in *10th IREP Symposium – Bulk Power Systems Dynamics and Control*, 2017. [.pdf]

F. Thams, A. Venzke, R. Eriksson, and S. Chatzivasileiadis, ”Efficient database generation for data-driven security assessment of power systems”. *IEEE Trans. Power Systems, 2019*. arXiv: http://arxiv.org/abs/1806.01074.pdf .

L. Wehenkel, M. Pavella, E. Euxibie, and B. Heilbronn, “Decision treebased transient stability method a case study,”IEEE Transactions onPower Systems, vol. 9, no. 1, pp. 459–469, 1994.

N. Hatziargyriou, G. Contaxis, and N. Sideris, “A decision tree method for on-line steady state security assessment,” IEEE Transactions onPower Systems, vol. 9, no. 2, pp. 1052–1061, 1994.

I. Genc, R. Diao, V. Vittal, S. Kolluri, and S. Mandal, “Decision tree-based preventive and corrective control applications for dynamic securityenhancement in power systems,”IEEE Transactions on Power Systems,vol. 25, no. 3, pp. 1611–1619, 2010.

C. Hidalgo-Arteaga, F. Hancharou, F. Thams, S. Chatzivasileiadis, Deep Learning for Power System Security Assessment. In *IEEE Powertech 2019*, Milan, Italy, pages 1-6, June 2019. [.pdf]

Deep learning, NIPS Tutorial 2017: https://www.youtube.com/watch?v=YJnddoa8sHk