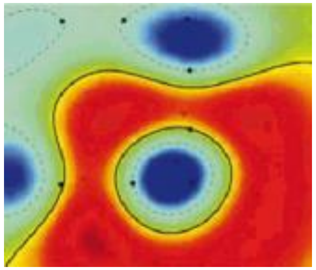
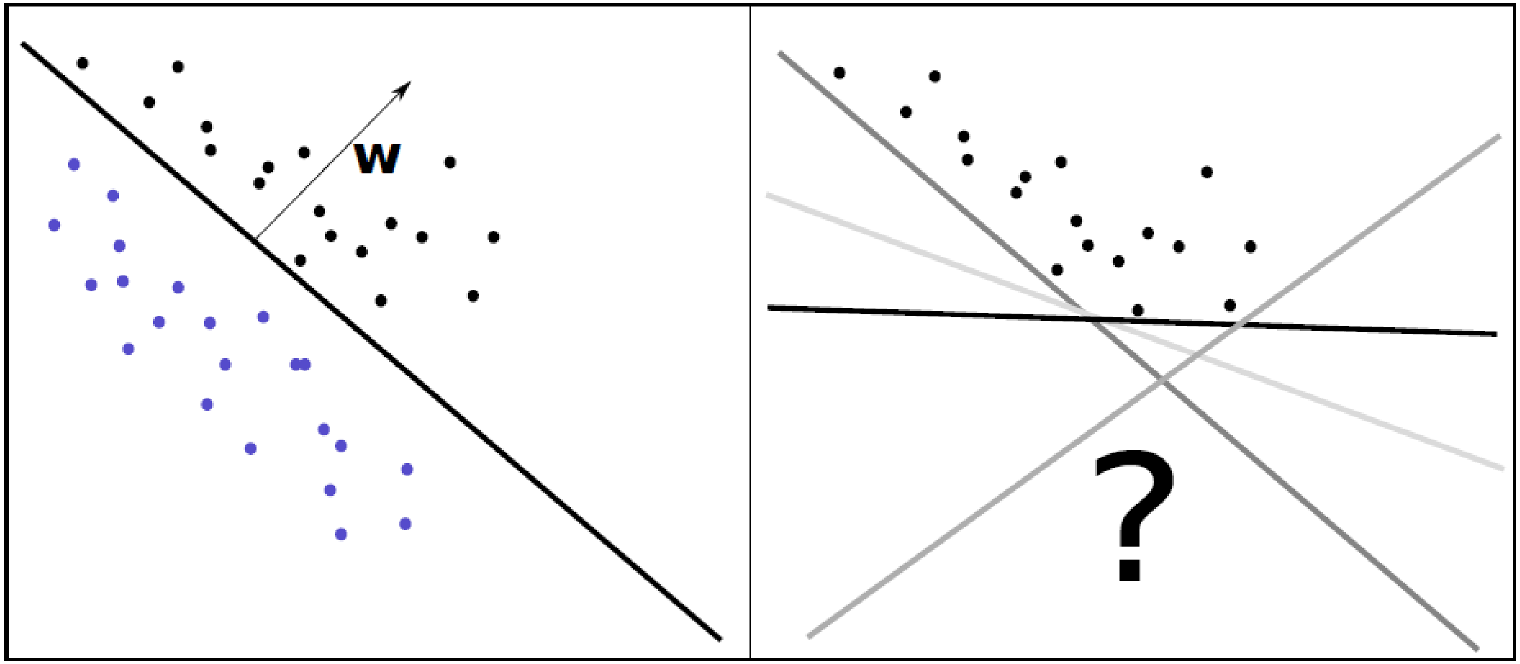
Anomaly Detection



Lecture by Klaus-Robert Müller, TUB 2023

Anomalies - introduction



Two-class (left): select a hyperplane such, that both classes are well separated
One-class (right): what is a good model?

Three approaches to Anomaly detection

- **Density-based:** Learn a density model of the inlier data $p(x)$, and then classify a new data point as 'outlier' when the probability assigned to it is low.
- **Reconstruction-based:** Learn a reconstruction model of the data $x \mapsto \mathrm{proj}_{\mathcal{D}}(x)$, and then classify as 'outlier' when the reconstruction error is high.
- **Boundary-based:** Learn a separating surface between the inlier data and the outlier data (e.g. a sphere enclosing all inliers).

Kernel Density Estimation (density based)

Kernel density estimation is a density-based anomaly model, based on the probability function:

$$p(\boldsymbol{x}) = \frac{1}{Z}\sum_{i=1}^{N} k(\boldsymbol{x}, \boldsymbol{x}_i)$$

where we typically choose $k(\boldsymbol{x}, \boldsymbol{x}_i) = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}_i\|^2)$ i.e. the Gaussian kernel.

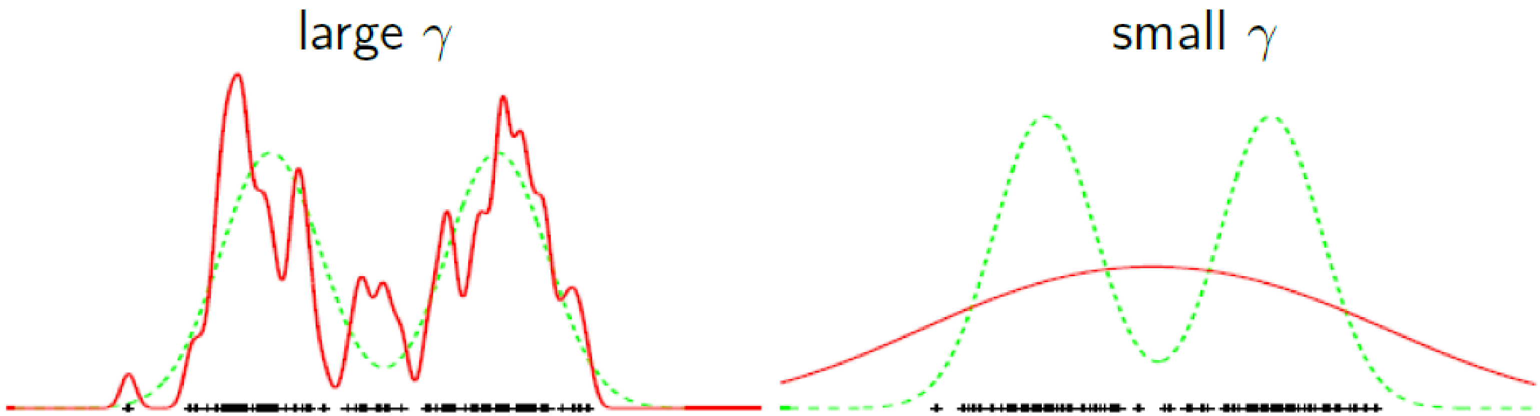


Image source: Raykar et al. 2006. Fast optimal bandwidth selection for kernel density estimation

Kernel Density Estimation (density based)

Feature space view of Kernel Density Estimation:

- Kernel density estimation can also be interpreted as some prediction model in kernel feature space. Assume k induces a feature map $\phi : \mathbb{R}^d \to \mathcal{H}$, the probability function can be written as:

$$
\begin{aligned}
p(\boldsymbol{x}) &= \frac{1}{Z} \sum_{i=1}^{N} k(\boldsymbol{x}, \boldsymbol{x}_i) \\
&= \frac{1}{Z} \sum_{i=1}^{N} \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}_i) \rangle \\
&= \left\langle \phi(\boldsymbol{x}), \underbrace{\frac{1}{Z} \sum_{i=1}^{N} \phi(\boldsymbol{x}_i)}_{\text{mean}} \right\rangle
\end{aligned}
$$

i.e. a data point is an inlier if it aligns with the data mean in feature space.

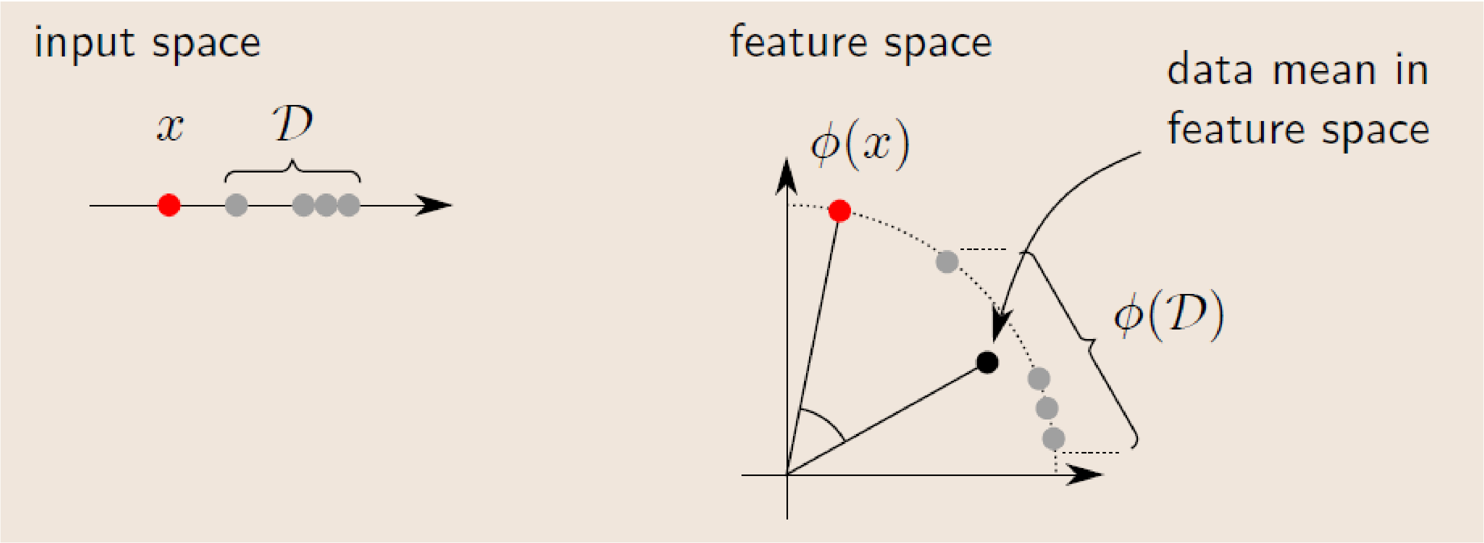
Kernel Density Estimation (density based)

Feature space view of Kernel Density Estimation:

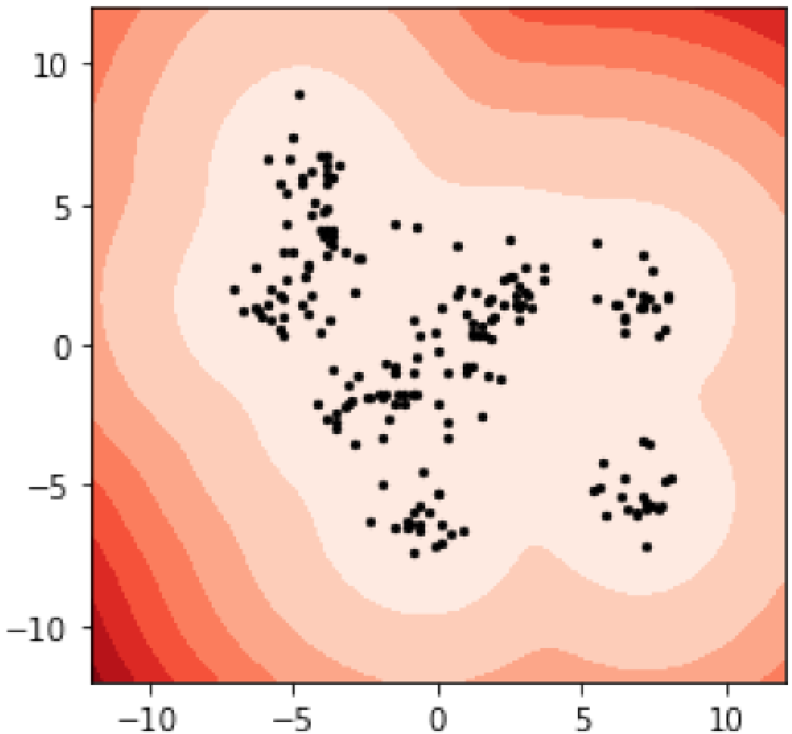
- KDE can be rewritten as:

$$p(\boldsymbol{x}) = \left\langle \phi(\boldsymbol{x}), \underbrace{\frac{1}{Z}\sum_{i=1}^{N}\phi(\boldsymbol{x}_i)}_{\text{mean}} \right\rangle$$

- Visual intuition (one-dimensional input space):



Kernel Density Estimation (density based)



- Input data in two dimensions ($\boldsymbol{x} \in \mathbb{R}^2$).
- The outlier score is computed from the probability function as:

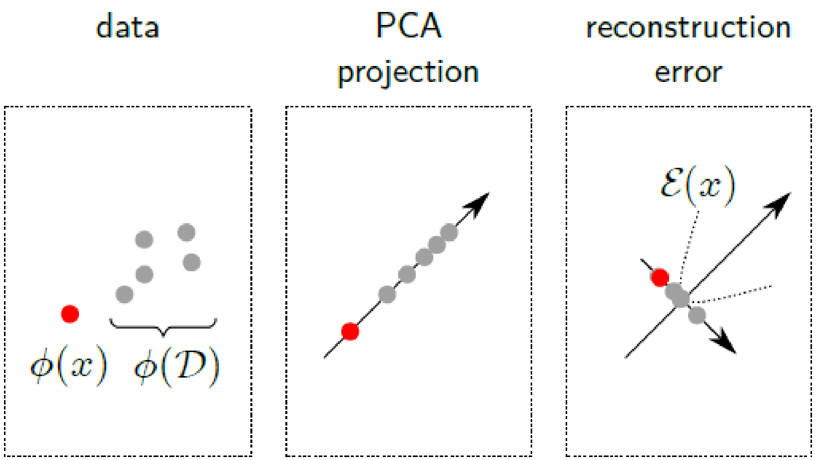
$$o(\boldsymbol{x}) = -\log p(\boldsymbol{x}).$$

  The more red, the higher the outlier score.
- Outlier score grows in every direction where there is no data.

Kernel PCA (reconstruction based)

Uncentered PCA projection in feature space can be written as:

$$\phi(\boldsymbol{x}) = \underbrace{\sum_{i=1}^{a} u_i u_i^\top \phi(\boldsymbol{x})}_{\text{PCA model}} + \underbrace{\sum_{i=a+1}^{h} u_i u_i^\top \phi(\boldsymbol{x})}_{\text{residuals}}$$



where $u_1, \ldots, u_a$ are the principal components. Reconstruction error is given by the square distance between the data and its projection in PCA space:

$$o(\boldsymbol{x}) = \left\| \phi(\boldsymbol{x}) - \sum_{i=1}^{a} u_i u_i^\top \phi(\boldsymbol{x}) \right\|^2$$

$$= k(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i=1}^{a} (u_i^\top \phi(\boldsymbol{x}))^2$$

Kernel PCA (reconstruction based)

Question: Can we compute the projections

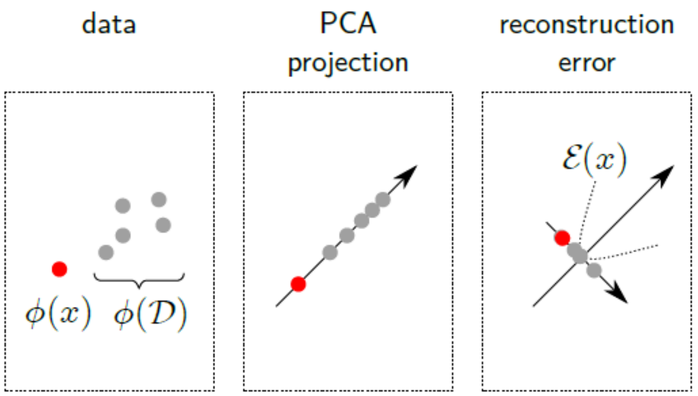
$$\text{proj}_i(\boldsymbol{x}) = u_i^\top \phi(\boldsymbol{x}) \quad ?$$

Answer: Using the decomposition $K = V \Lambda V^\top$, the projection on the ith principal component is given by:

$$\text{proj}_i(\boldsymbol{x}) = k(\boldsymbol{x}, X) \cdot V_{:,i} \cdot \lambda_i^{-0.5},$$

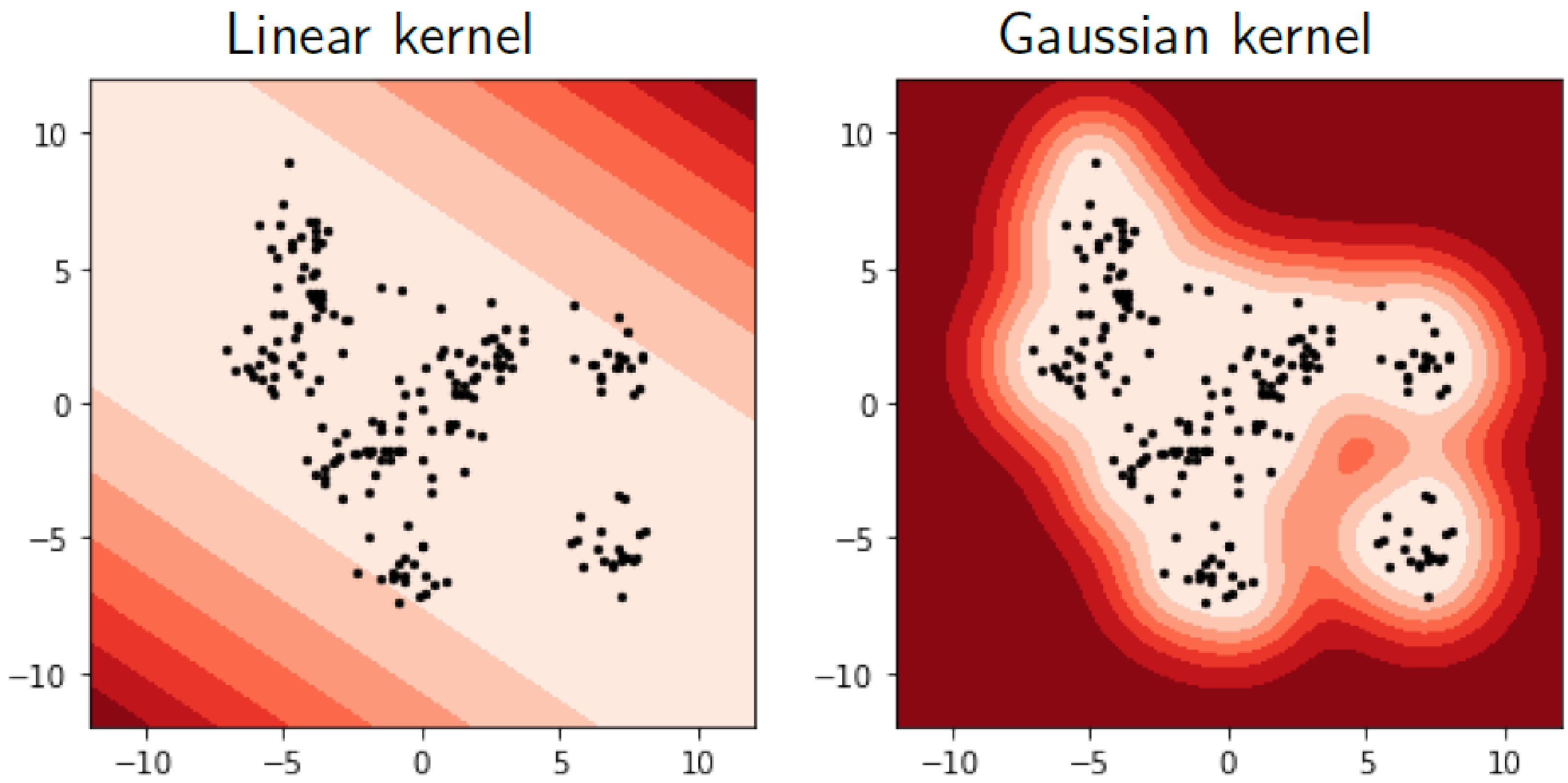
We can then compute an outlier score using this projection

$$o(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}') - \sum_{i=1}^{a} (\text{proj}_i(\boldsymbol{x}))^2.$$



Kernel PCA (reconstruction based)

Two-dimensional example, different choices of kernel functions:



The Gaussian kernel better captures the shape of the data.

What is one class learning?

Objective

- Learn common properties of the examples and be able to tell if a test point belongs to the class or not
- Assuming we know the data distribution $p(\mathbf{x})$, the task is, to reject all data points with $p(x) < \nu$ given a pre-defined threshold ν.
- Unfortunately, we usually don't know $p(\cdot)$
- Therefore, we need to estimate it ...

Applications

- Anomaly Detection

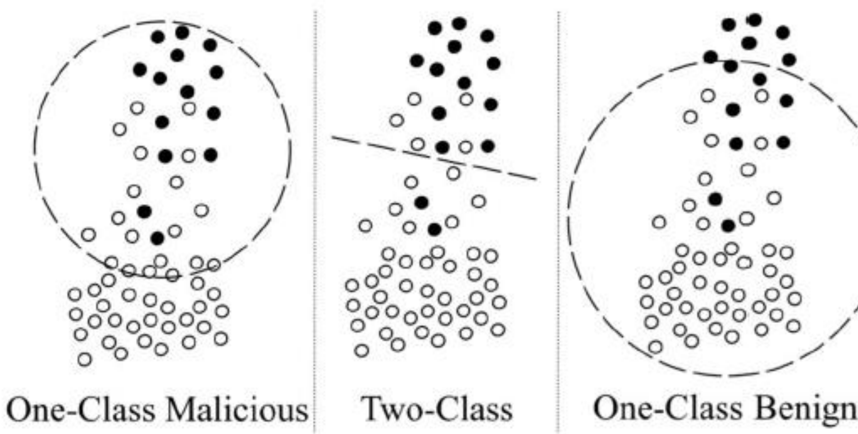
First Appearance in Literature

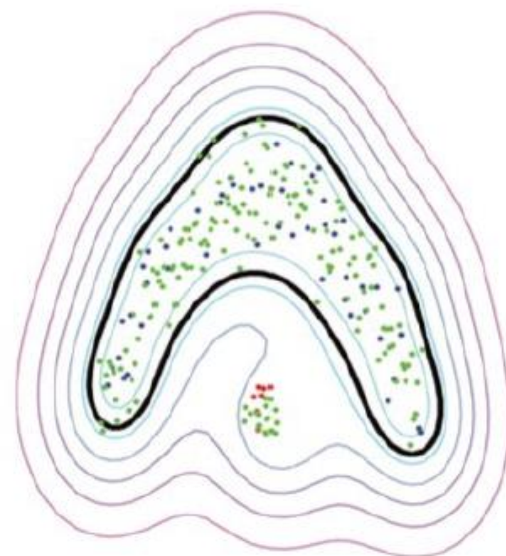
Moya & Hush (1996):'Network constraints and multi-objective optimization for one-class classification', Neural Networks

Close Relation To Two-class SVM

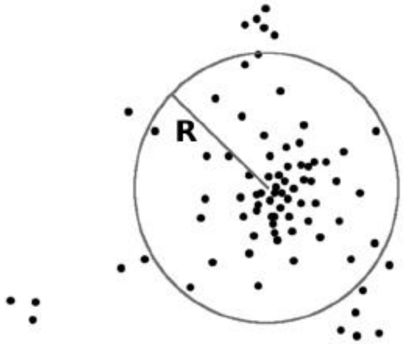
- The kernel trick applies !
- Primal & Dual formulation
- Approaching One Class Classification from supervised learning, e.g. via weighting unbalanced classes (for linear kernels)

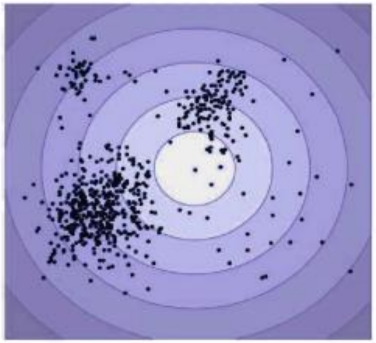
- Possibility of extending One Class Classification from unsupervised learning, e.g. via active (semi-supervised) learning

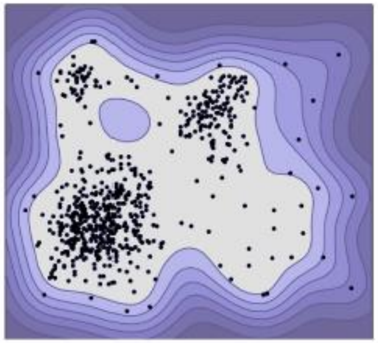




Support Vector Data description





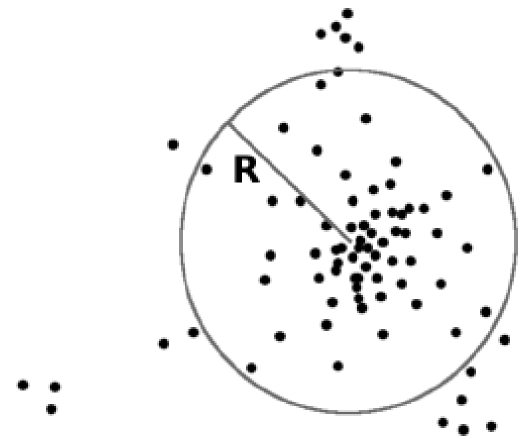


Support Vector Data Description (SVDD)

- Compute minimal enclosing sphere with center $\mathbf{c}$ and radius R
- Anomaly score as the distance to center $\mathbf{c}$, that is $f(\mathbf{x}) = \|\phi(\mathbf{x}) - \mathbf{c}\|$
- Accept data point $\mathbf{x}$ if $f(\mathbf{x}) \leq R$ and ...

  ... reject $\mathbf{x}$ if $f(\mathbf{x}) > R$

Support Vector Data description: optimization



Primal optimization problem $0 \le \nu \le 1$

$$\min_{R,\mathbf{c},\xi} \quad R^2 + \frac{1}{n\nu}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad \forall_{i=1}^{n} : \; \|\phi(\mathbf{x}_i) - \mathbf{c}\|^2 \le R^2 + \xi_i \quad \text{and} \quad \xi_i \ge 0$$

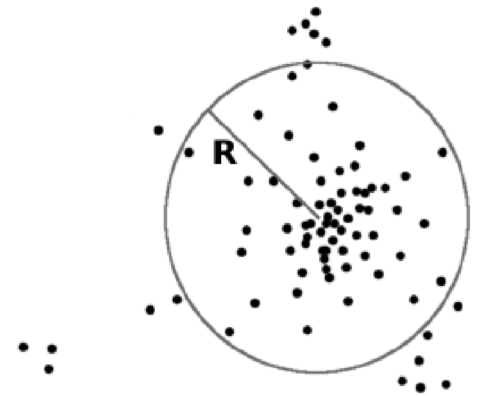
Dual optimization problem

$$\max_{\alpha} \quad \sum_{i=1}^{n}\alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i = 1 \quad \text{and} \quad 0 \le \alpha_i \le \frac{1}{n\nu} \quad \forall i$$

And center $\mathbf{c} = \sum_i^n \alpha_i \phi(\mathbf{x}_i)$

Support Vector Data description: properties

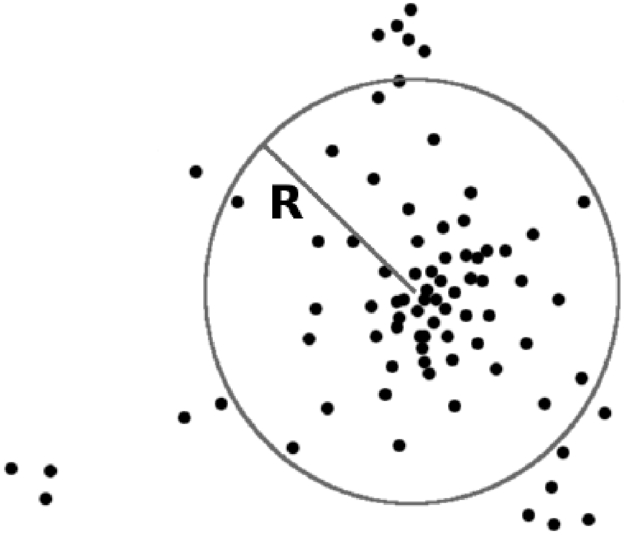


Remind: center $\mathbf{c} = \sum_i^n \alpha_i \phi(\mathbf{x}_i)$ and $0 \leq \nu \leq 1$

$$\sum_{i=1}^{n} \alpha_i = 1 \quad \text{and} \quad 0 \leq \alpha_i \leq \frac{1}{n\nu} \quad \forall i$$

- ν is an upper bound on the fraction of outliers
- ν is a lower bound on the fraction of support vectors
- Center-of-mass method: $\nu = 1$

Support Vector Data description: summary



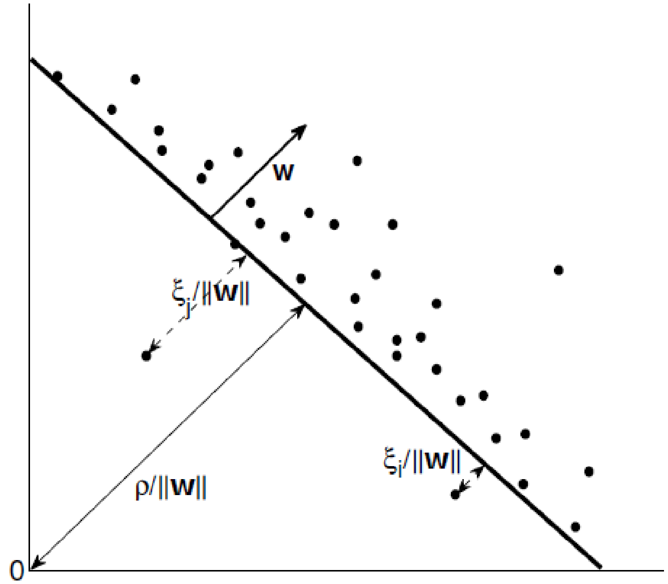
Advantages:

- Neat idea, easy to explain..
- No labels required
- Training set can be comprised of nominal and some anomalous data
- Convex problem: every optimal solution is a global optimal solution
- Center $\mathbf{c}$ and radius R are infered depending on the location of the data points

Drawbacks:

- Experts can hardly influence with the adaptation process
- Resulting classifications may not be interpretable

Alternative: One class SVM



One-class SVM

- Separate data from origin with hyperplane with maximum distance to origin
- Model function: $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle - \rho$

One class SVM:optimization

Primal optimization problem $0 \leq \nu \leq 1$

$$\min_{\mathbf{w},\rho,\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 - \rho + \frac{1}{n\nu}\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad \forall_{i=1}^{n} : \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle \geq \rho - \xi_i \quad \text{and} \quad \xi_i \geq 0$$

Dual optimization problem

$$\max_{\alpha} \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i = 1 \quad \text{and} \quad 0 \leq \alpha_i \leq \frac{1}{n\nu} \quad \forall i$$

And expansion $\mathbf{w} = \sum_i^n \alpha_i \phi(\mathbf{x}_i)$

One-class SVMs vs. SVDD

One-class SVM vs SVDD

They are equal under fairly general assumption!

Remember SVDD dual optimization objective (constraints are equal for one-class SVM and SVDD: $\sum_{i=1}^{n} \alpha_i = 1$ and $0 \leq \alpha_i \leq \frac{1}{n\nu}$)?

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$$

- Now, we assume that $\|\phi(\mathbf{x})\|^2 = k(\mathbf{x}, \mathbf{x}) = \sigma$ (e.g. Gaussian kernel!), then ...

$$\max_{\boldsymbol{\alpha}} \quad \sigma \underbrace{\sum_{i=1}^{n} \alpha_i}_{=1 \text{ (due to equality constraint)}} - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$$

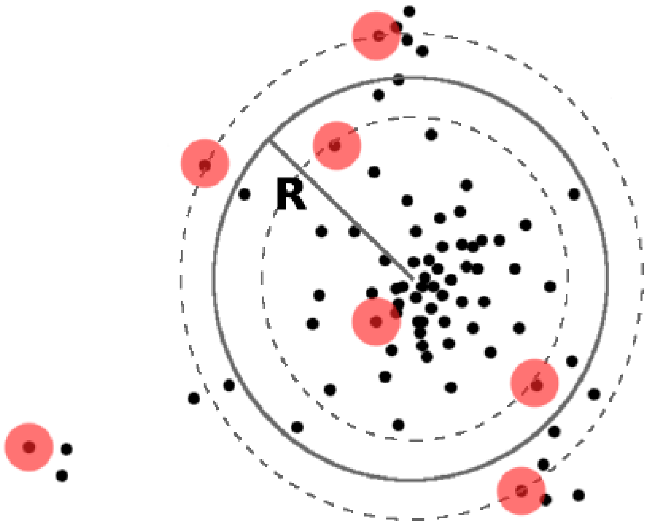
$$= \max_{\boldsymbol{\alpha}} \quad \sigma - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$$

$$= \max_{\boldsymbol{\alpha}} \quad -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \quad \underline{= \text{One-class SVM}}$$

Semi-supervised anomaly detection (SSAD): idea

Generalize the SVDD by including labels in a SVM fashion

- Exploit prior and oracle knowledge to increase anomaly detection accuracy
- Enable experts to verify uncertain guesses

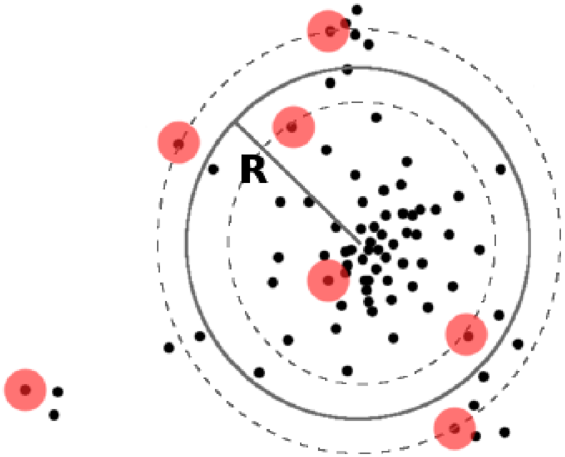


SSAD: problem formulation

Semi-supervised generalization of the SVDD

Allows for the inclusion of **unlabeled** and labeled data

Parameters: center $\mathbf{c}$, radius R and confidence γ



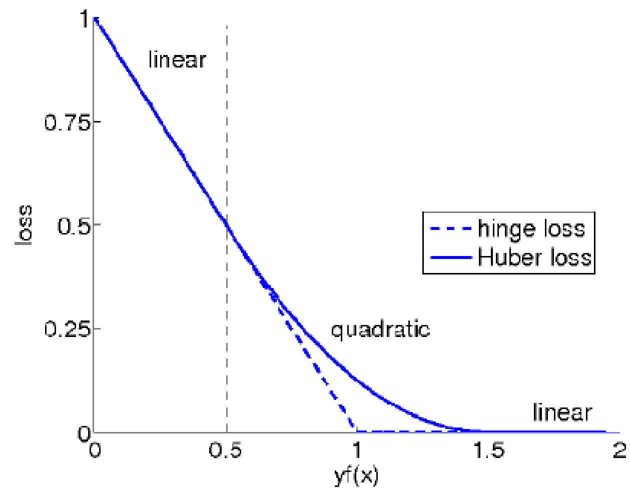
Extended SVDD problem formulation:

$$\min_{R,\gamma,\mathbf{c},\xi} \quad R^2 - \kappa\gamma + \eta_u \sum_{i=1}^{n} \xi_i + \eta_l \sum_{j=n+1}^{n+m} \xi_j$$

$$\begin{aligned}
\text{s.t.} \quad & \forall_{i=1}^{n} : \|\phi(\mathbf{x}_i) - \mathbf{c}\|^2 \leq R^2 + \xi_i \\
& \forall_{j=n+1}^{n+m} : y_j \left(\|\phi(\mathbf{x}_j) - \mathbf{c}\|^2 - R^2 \right) \leq -\gamma + \xi_j \\
& \forall_{i=1}^{n} : \xi_i \geq 0, \\
& \forall_{j=n+1}^{n+m} : \xi_j \geq 0.
\end{aligned}$$

...BUT non-convex

SSAD - reformulation



- Optimization problem is non-convex
- Remedy: translate the constrained, uncontinuous problem into an unconstrained, continuous problem
- Substitute slack variables:

$$\xi_i = \ell_{0,\epsilon}\left(R^2 - ||\phi(\mathbf{x}_i) - \mathbf{c}||^2\right)$$

$$\xi_j = \ell_{0,\epsilon}\left(y_j\left(R^2 - ||\phi(\mathbf{x}_j) - \mathbf{c}||^2\right) - \gamma\right)$$

SSAD: optimization

- Unconstrained, continuous objective:

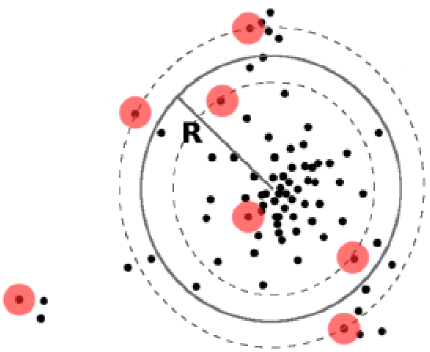
$$\min_{R,\gamma,\mathbf{c}} P = \min_{R,\gamma,\mathbf{c}} \quad \underbrace{R^2}_{\substack{\textit{small}\\ \textit{Radius}}} - \underbrace{\kappa\gamma}_{\substack{\textit{large}\\ \textit{Margin}}} + \underbrace{\eta_u \sum_{i=1}^{n} \ell_{0,\epsilon}\left(R^2 - ||\phi(\mathbf{x}_i) - \mathbf{c}||^2\right)}_{\textit{Error on unclassified Data}}$$

$$+ \underbrace{\eta_l \sum_{j=n+1}^{n+m} \ell_{0,\epsilon}\left(y_j\left(R^2 - ||\phi(\mathbf{x}_j) - \mathbf{c}||^2\right) - \gamma\right)}_{\textit{Error on classified Data}}$$

- Generalization of SVDD to a semi-supervised method
- Non-convex optimization problem, dual optimization is prohibitive
- Efficiently solvable using gradient descent methods
- Use Representer Theorem to obtain a non-linear version:

$$\mathbf{c} = \sum_i \alpha_i \phi(\mathbf{x}_i) + \sum_j \alpha_j y_j \phi(\mathbf{x}_j)$$

SSAD - summary



Semi-supervised Anomaly Detection

Exploit prior and oracle knowledge to increase anomaly detection accuracy

Enable experts to verify uncertain guesses

Important Special Case

Similar to the relation between SVDD and One-class SVM, there is a one-class SVM-style counterpart for SSAD (if $\|\phi(\mathbf{x}_i)\| = const$) with a convex formulation ($\mathbf{1}_i = 0$ for unlabeled examples and $\mathbf{1}_i = 1$ for positive or negative labeled examples):

$$\min_{\mathbf{w},\rho,\gamma\geq 0,\xi\geq 0} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 - \rho - \kappa\gamma + \sum_{i=1}^{n+m}(\mathbf{1}_i\eta_l + (1-\mathbf{1}_i)\eta_u)\xi_i$$

$$\text{s.t.} \quad \forall_{i=1}^{n+m}: \; y_i\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle \geq y_i\rho + \mathbf{1}_i\gamma - \xi_i$$

Overview of Anomaly detection methods

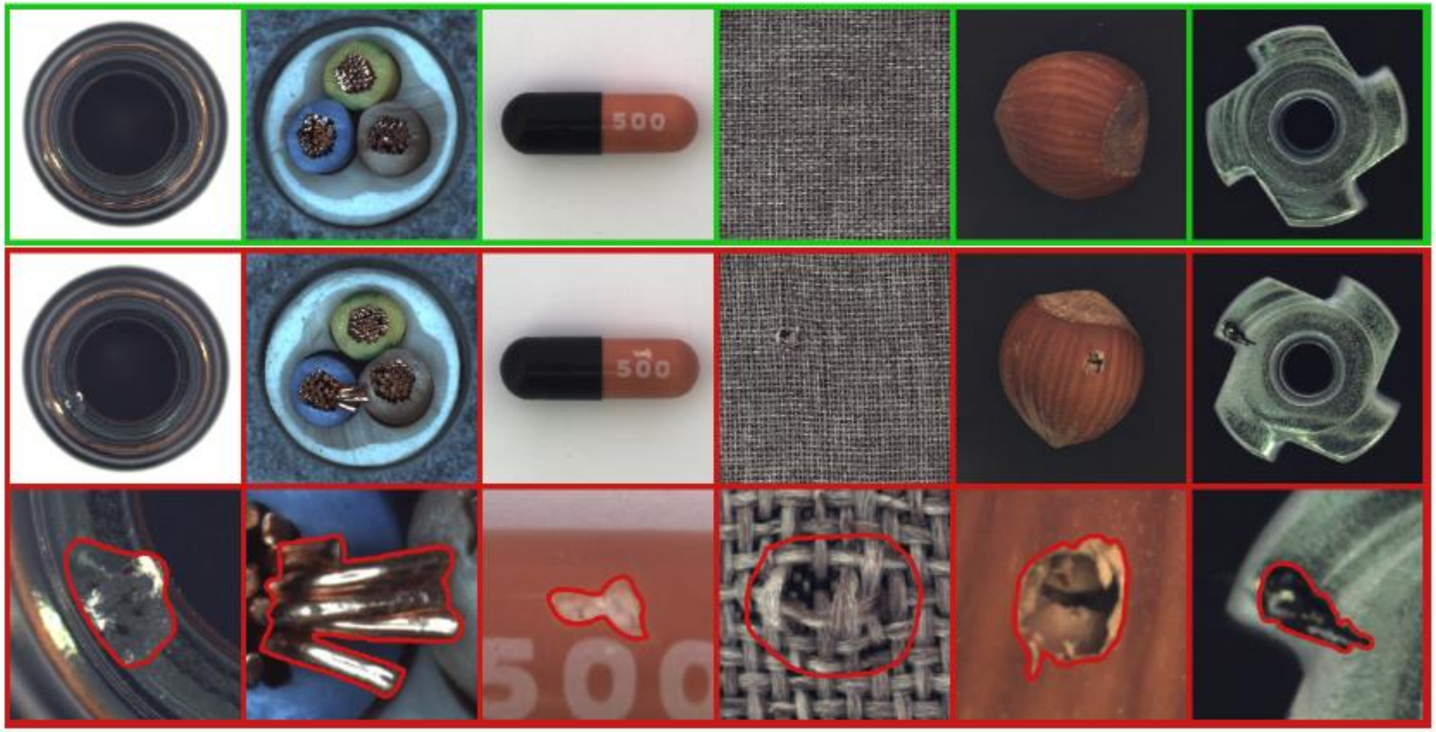
Many methods for anomaly detection have been proposed. They can be roughly organized in the following table:

	Kernel	Deep
Density-based	*KDE*	DBMs, Hierarchical Latent
Reconstruction-based	*KPCA*	Autoencoder
Boundary-based	*OC-SVM*	GAN, deep OC-SVM

- None of the methods above is strictly superior. Each method has its strengths and weaknesses.
- Many other methods exist that are not in this table (isolation forests, local outlier factor). Cf. Ruff et al. [4] for a review of anomaly detection.

Anomaly detection in practice

The MVTec dataset: Finding anomalies for industrial inspection.



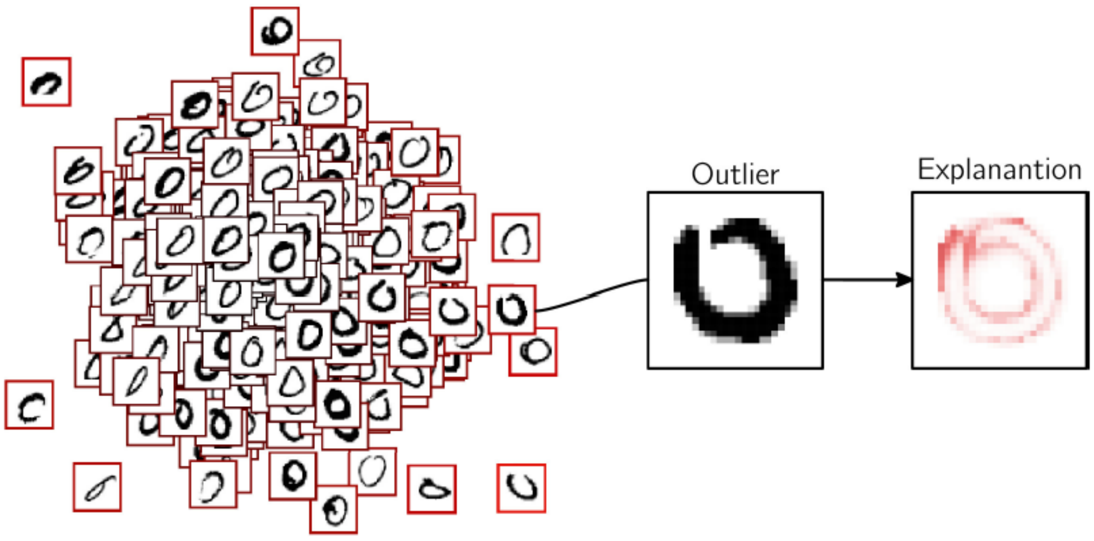
Anomaly detection in practice

Performance of different anomaly detection models on different MVTec classes (AUC metric)

		Gaussian	MVE	PCA	KDE	SVDD	kPCA	AGAN	DOCC	AE
Textures	carpet	48.8	63.5	45.6	34.8	48.7	41.9	83.1	**90.6**	36.8
	grid	60.6	67.8	81.8	71.7	80.4	76.7	**91.7**	52.4	74.6
	leather	39.6	49.5	60.3	41.5	57.3	61.1	58.6	**78.3**	64.0
	tile	68.5	79.7	56.4	68.9	73.3	63.2	74.1	**96.5**	51.8
	wood	54.0	80.1	90.4	**94.7**	94.1	90.6	74.5	91.6	88.5
Objects	bottle	78.9	67.0	97.4	83.3	89.3	96.3	90.6	**99.6**	95.0
	cable	56.5	71.9	77.6	66.9	73.1	75.6	69.7	**90.9**	57.3
	capsule	71.6	65.1	75.7	56.2	61.3	71.5	60.7	**91.0**	52.5
	hazelnut	67.6	80.4	89.1	69.9	74.3	83.8	**96.4**	95.0	90.5
	metal nut	54.7	45.1	56.4	33.3	54.3	59.0	79.3	**85.2**	45.5
	pill	65.5	71.5	**82.5**	69.1	76.2	80.7	64.6	80.4	76.0
	screw	53.5	35.5	67.9	36.9	8.6	46.7	**99.6**	86.9	77.9
	toothbrush	93.9	76.1	**98.3**	93.3	96.1	**98.3**	70.8	96.4	49.4
	transistor	70.2	64.8	81.8	72.4	74.8	80.0	78.8	**90.8**	51.2
	zipper	50.1	65.2	82.8	61.4	68.6	81.0	69.7	**92.4**	35.0

Beyond prediction: explaining anomalies

Sometimes, it is important to not only detect that a point is anomalous, but also to *understand* why a data point has been classified to be anomalous (to verify that the detection is justified).



Question: How to go backward in the model to identify pixels that are responsible for outlierness?

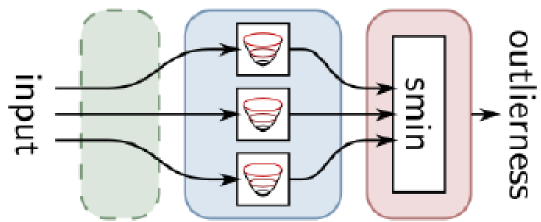
Explaining KDE and OC-SVMs

Insight: Models of the type:

$$f(\boldsymbol{x}) = \sum_i \alpha_i \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}_i\|^2)$$

e.g. one-class SVM and kernel density estimation (KDE) can be rewritten as:

$$\begin{aligned} o(\boldsymbol{x}) &= -\frac{1}{\gamma} \log f(\boldsymbol{x}) \\ &= -\frac{1}{\gamma} \log \sum_i \exp(-\gamma(\|\boldsymbol{x} - \boldsymbol{x}_i\|^2 - \log \alpha_i)) \\ &= \min_i{}^{\gamma} \{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2 - \log \alpha_i\} \end{aligned}$$



i.e. a soft minimum over squared distances [5].

Explaining KDE and OC-SVMs

The outlier score

$$o(\boldsymbol{x}) = \min_i{}^{\gamma}\{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2 - \log \alpha_i\}$$

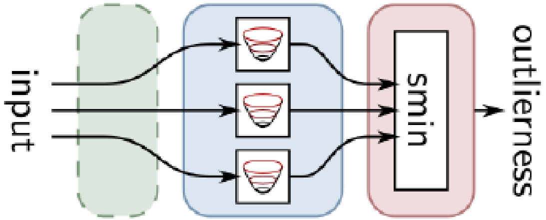
can therefore be redistributed in two steps, (1) min-take-most in the pooling layer, and directional redistribution in the distance layer. I.e. using

$$\operatorname{argmin}_i{}^{\gamma}\{\cdot\}$$

for the pooling part, and

$$(\boldsymbol{x} - \boldsymbol{x}_i)^2 / \|\boldsymbol{x} - \boldsymbol{x}_i\|^2$$

for the squared distance (cf. [5]).

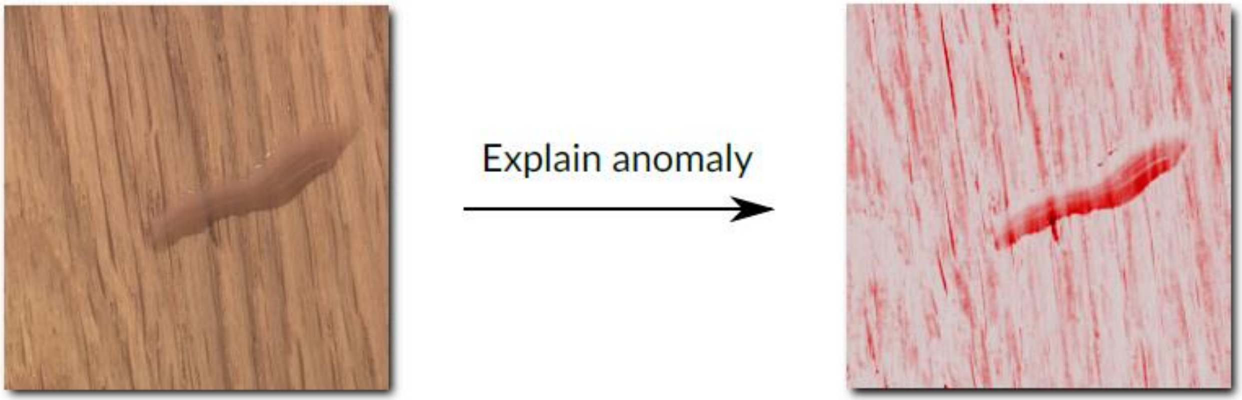


Explaining anomaly detection

		Gaussian	MVE	PCA	KDE	SVDD	kPCA	AGAN	DOCC	AE
Textures	carpet	48.8	63.5	45.6	34.8	48.7	41.9	83.1	**90.6**	36.8
	grid	60.6	67.8	81.8	71.7	80.4	76.7	**91.7**	52.4	74.6
	leather	39.6	49.5	60.3	41.5	57.3	61.1	58.6	**78.3**	64.0
	tile	68.5	79.7	56.4	68.9	73.3	63.2	74.1	**96.5**	51.8
	wood	54.0	80.1	90.4	**94.7**	94.1	90.6	74.5	91.6	88.5
Objects	bottle	78.9	67.0	97.4	83.3	89.3	96.3	90.6	**99.6**	95.0
	cable	56.5	71.9	77.6	66.9	73.1	75.6	69.7	**90.9**	57.3
	capsule	71.6	65.1	75.7	56.2	61.3	71.5	60.7	**91.0**	52.5
	hazelnut	67.6	80.4	89.1	69.9	74.3	83.8	**96.4**	95.0	90.5
	metal nut	54.7	45.1	56.4	33.3	54.3	59.0	79.3	**85.2**	45.5
	pill	65.5	71.5	**82.5**	69.1	76.2	80.7	64.6	80.4	76.0
	screw	53.5	35.5	67.9	36.9	8.6	46.7	**99.6**	86.9	77.9
	toothbrush	93.9	76.1	**98.3**	93.3	96.1	**98.3**	70.8	96.4	49.4
	transistor	70.2	64.8	81.8	72.4	74.8	80.0	78.8	**90.8**	51.2
	zipper	50.1	65.2	82.8	61.4	68.6	81.0	69.7	**92.4**	35.0

What prediction strategy the KDE model uses to successfully predict the class 'wood'?

Explaining anomaly detection

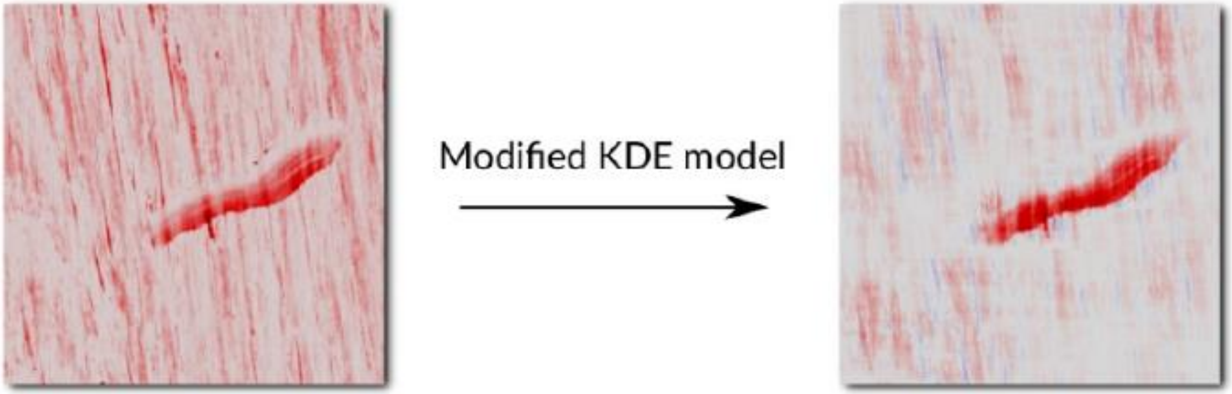


- The model detects the anomalous liquid stain, but also reacts to wood's vertical stripes (these are perfectly normal in a wood image!).
- Reliance on vertical stripes could harm generalization on new wood images.

Explaining anomaly detection

Idea: Replace in the original KDE model the Euclidean metric by a Malahanobis metric with covariance Σ hardcoded to reduce the high horizontal frequencies.

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \frac{1}{N} \exp(-\gamma\,(\boldsymbol{x}-\boldsymbol{x}_i)^\top \Sigma (\boldsymbol{x}-\boldsymbol{x}_i))$$



- The anomaly decision is now supported by the correct features.

More applications

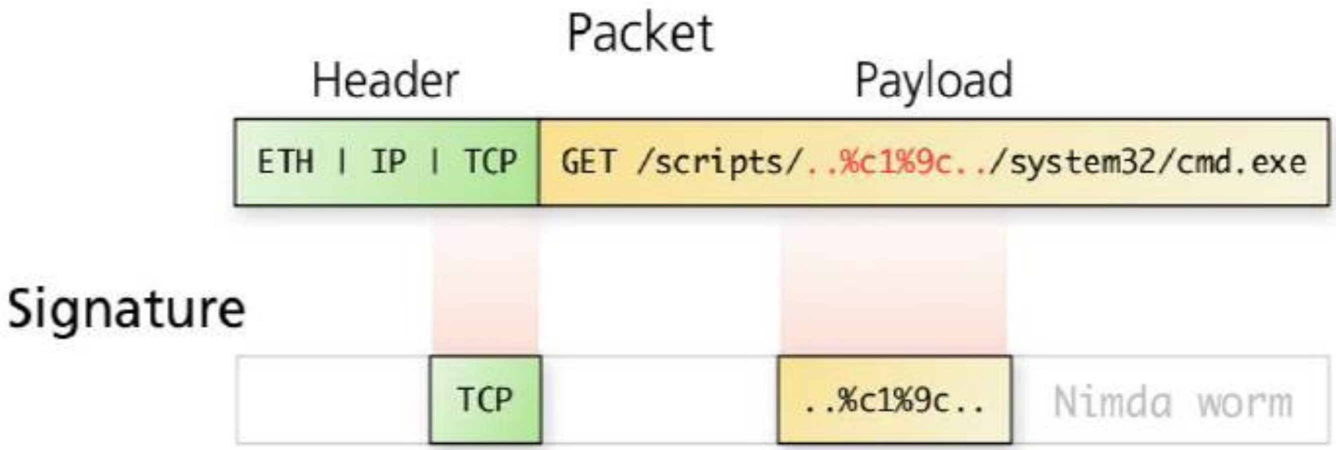
Application: Detecting attacts in network traffic

- An **Attack** is an attempt to compromise the confidentiality, integrity or availability of a system
- An **Intrusion Detection System (IDS)** is a system monitoring a stream of events for attacks

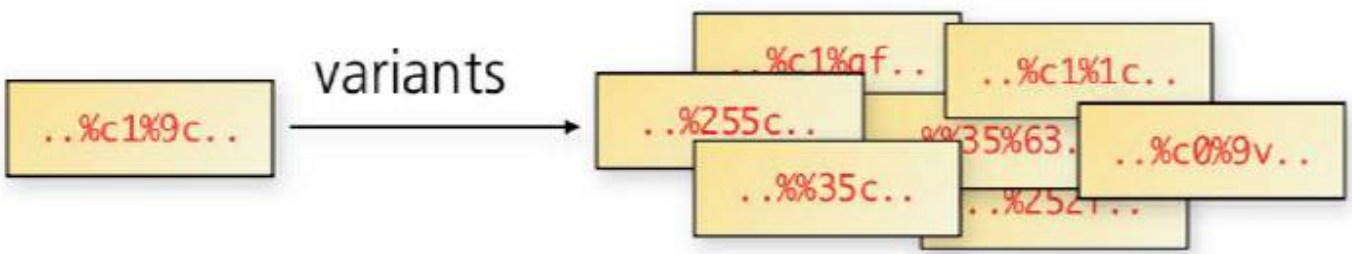
- Imagine: You write a love-letter to your friend

  Only the recipient should read the letter → Confidentiality
  Your message should not be changed → Integrity
  The target mailbox should not be blocked → Availability

Network intrusion detection – signature based



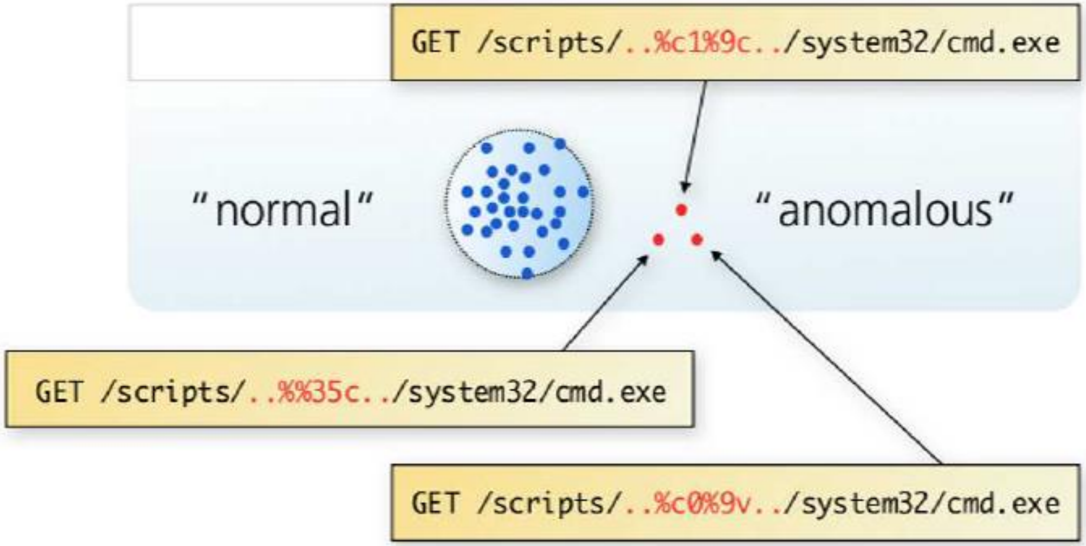
- Build signatures by searching for significant patterns in malicious data
- Use those signatures for identification of attacks



- Can only detect already known attacks
- Ineffective against attack variants and polymorphism

Network intrusion detection – machine learning based

- Find unknown attacks
- Embed byte stream in a vector space



- Assumption: Malicious byte streams deviates from normal byte streams
- Learn a concise description of normal data
- Intrusion detection ≈ anomaly detection

Network intrusion detection – Feature spaces

- N-gram vector space: any substring s of length n is represented as a dimension
- Binary: 1 if substring s occures in message x and 0 otherwise
- Frequency: count occurances of substrings s in x

- Example:

```
GET /scripts/..%c1%9c../system32/cmd.exe
```

$$\longmapsto \begin{array}{cc} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 0 \end{pmatrix} & \begin{array}{l} \mathtt{GE} \\ \mathtt{ET} \\ \mathtt{T}\square \\ \square\mathtt{/} \\ \vdots \\ \mathtt{bb} \end{array} \end{array}$$

Empirical result

Data

Recorded within 10 Days at Fraunhofer FIRST Institute

145,069 normal HTTP Connections, mean length 489 bytes

27 real Attack classes with 2 – 6 Instances each (Metasploit)

Setup

3-gram representation of bytestream

Training (966+34), Holdout (795+27), and Test (795+27)

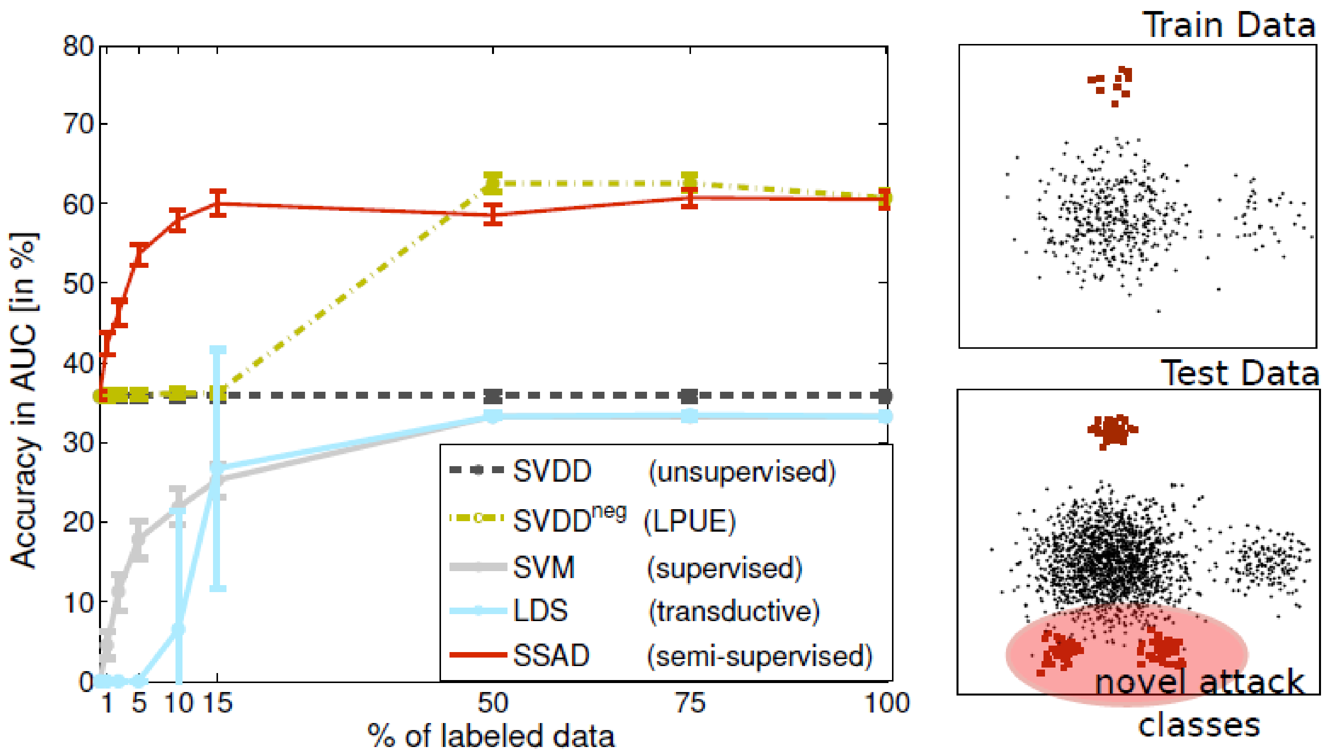
Attacks of same class occur either in train or test set

10 repetitions, AUC in the false positive interval $[0, 0.01]$

Obfuscation: attacks fake normal network traffic
attack = randomly chosen normal HTTP-Header + malicious payload

Effect: 3-gram representation is close to that of normal traffic

Useful for anomaly detection



- Lots of data of one type available but with no or only a few labels:

  e.g. network traffic with only a few attacks

- missing information, e.g. future attacks are different from todays (varying training and test distributions)

Anomalies in mobile communication

[Schwenk et al]

Anomaly detection on structured communication data with P3

- Heterogenous, structured Drive-Test Data
  - Fleet of cars all over europe, equipped with multiple smartphones & roof-mounted antennas
  - Smartphones run pre-defined test-sequences for scenarios like:
    - VoLTE (Voice over LTE), ViLTE (Video over LTE)
    - Data up-/downloads, website visits, etc.
  - Scale of Data: Several campaigns per year, with each campaign > 500.000 test sequences
  - Types of Data for the voice and data services
    - Logfiles of a proprietary analyzer
    - Raw Network Traces (pcap-files)
    - Radio Chipset Traces (like connection quality, bitrate changes, etc.)
  - Data is structured by the utilized network protocols, and also temporally and spatially

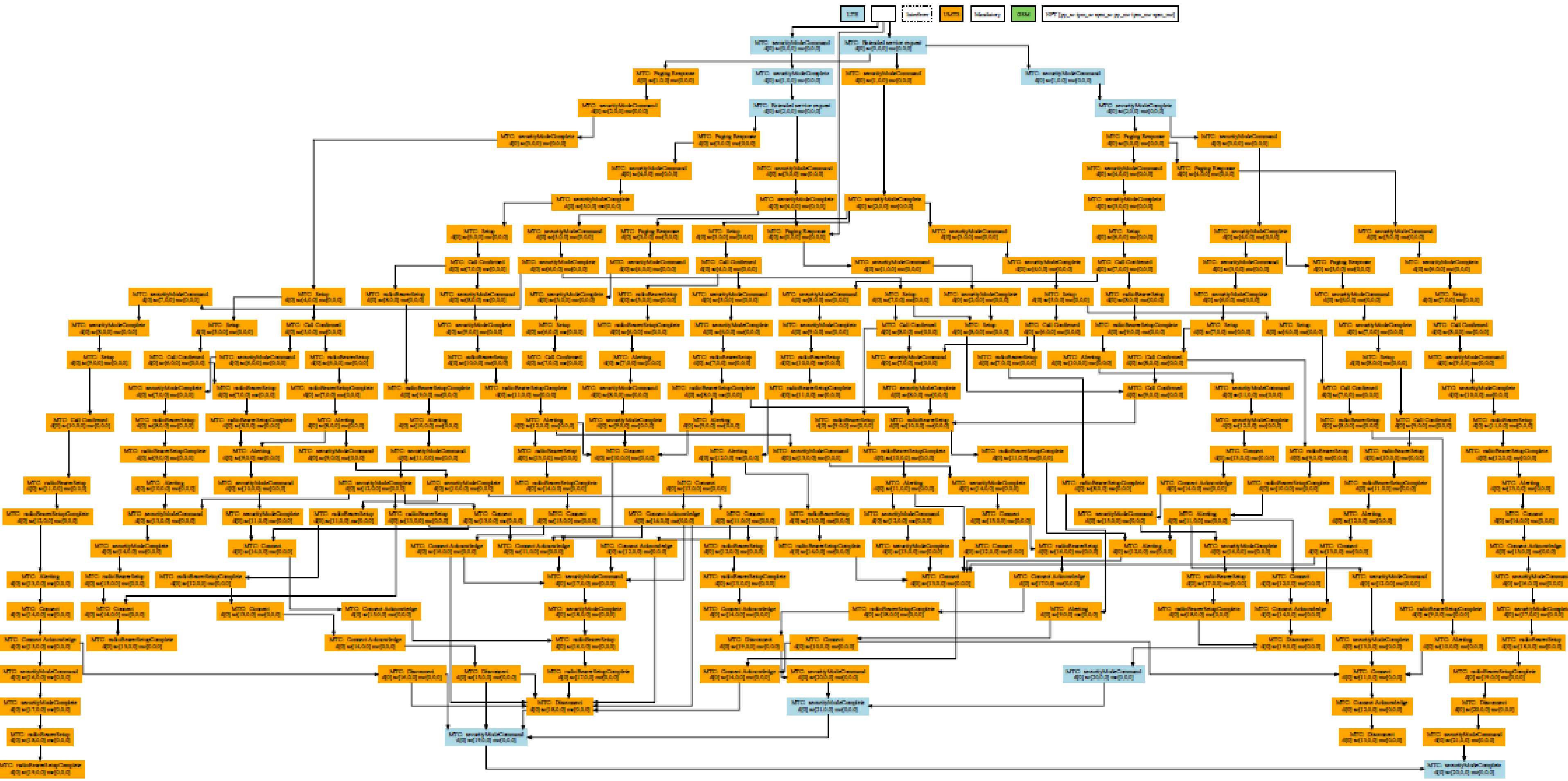
Set-up

7. Application:	HTTP, DNS
6. Presentation:	SSL, TLS
5. Session:	SIP
4. Transport:	TCP, ESP
3. Network:	IP, ICMP
2. Data link:	-
1. Physical:	-

OSI Hierarchy of Network Layers, inc. those used in Drive-Test Data

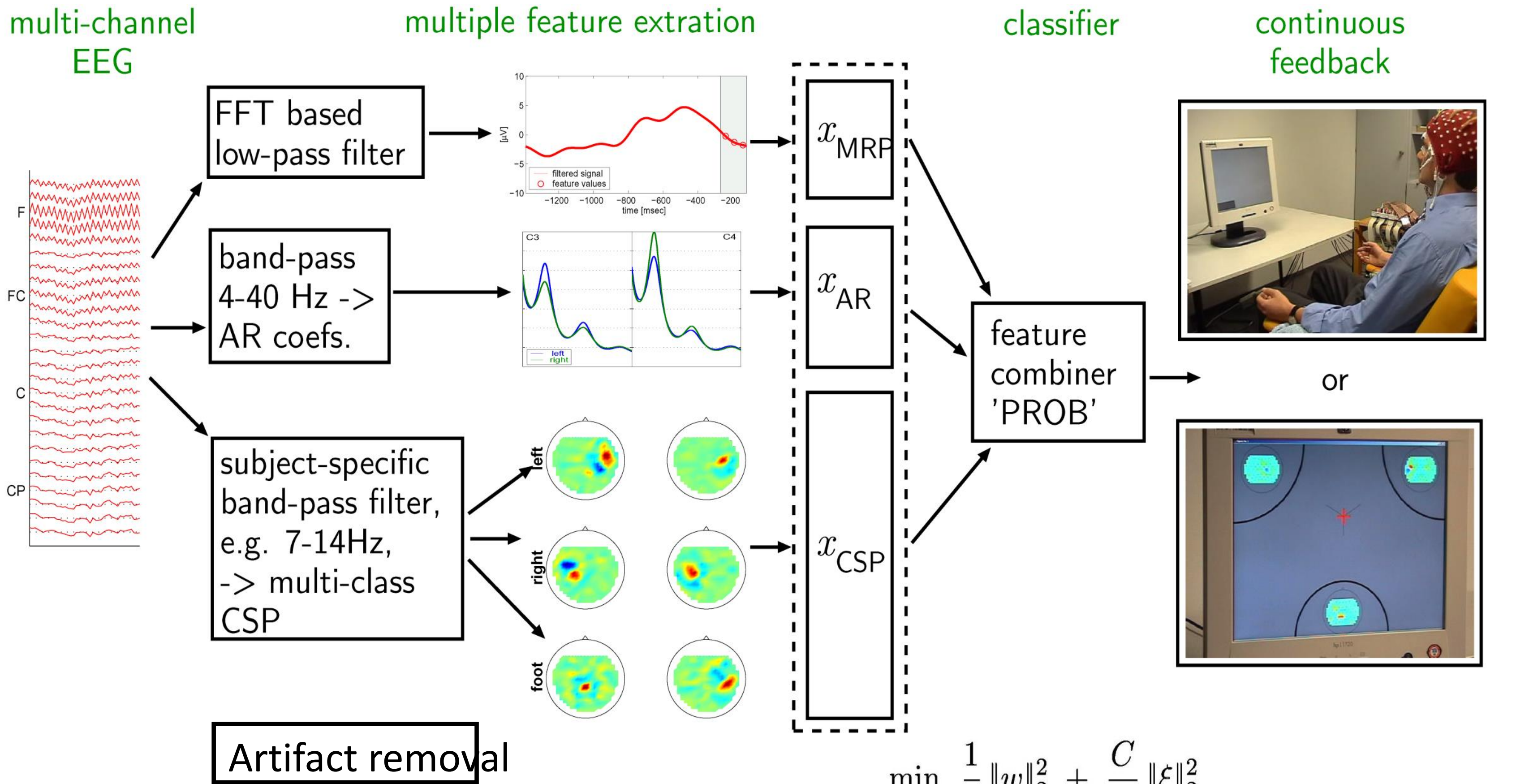
Proprietary Analysis
MsgTime: 2014-07-28 11:01:59.432 [STATEMACHINE] CommandPortState
MsgTime: 2014-07-28 11:01:59.436 [STATEMACHINE] CommandPortState-Result (Return: COM_RAS_BUSY)
MsgTime: 2014-07-28 11:01:59.438 [STATEMACHINE] GetConnectionState
MsgTime: 2014-07-28 11:01:59.438 [DASHBOARD ENGINE] TDashboardWrapperSmartphone.DoGetMobileInfo: Start
MsgTime: 2014-07-28 11:01:59.438 [DASHBOARD ENGINE] Sending command: SP GET MOBILE STATE
MsgTime: 2014-07-28 11:01:59.615 [SP_CTRL]VMCCSmarty-Version: 1.8.1.879, IPDumper-Version: 1.2.12.872, Android-Version: 4.3
MsgTime: 2014-07-28 11:01:59.615 [SP_CTRL]Operator: IMSI: 262010050810227; Telekom.de; TDG; 26201
MsgTime: 2014-07-28 11:01:59.615 [SP_CTRL]NETWORK_TYPE_LTE; Cid: 27535105; Lac: 13890; Psc: 0
Raw Network Traces
Source | Destination | Protocol | Length | Info
91.250.77.23 10.34.208.9 ICMP 844 Echo (ping) reply id=0x02ef, seq=1/256, ttl=49 (request
10.34.208.9 91.250.77.23 ICMP 844 Echo (ping) request id=0x02f0, seq=1/256, ttl=128 (reply i
91.250.77.23 10.34.208.9 ICMP 844 Echo (ping) reply id=0x02f0, seq=1/256, ttl=49 (request
10.34.208.9 91.250.77.23 ICMP 844 Echo (ping) request id=0x02f1, seq=1/256, ttl=128 (reply i
91.250.77.23 10.34.208.9 ICMP 844 Echo (ping) reply id=0x02f1, seq=1/256, ttl=49 (request
10.34.208.9 91.250.77.23 ICMP 844 Echo (ping) request id=0x02f2, seq=1/256, ttl=128 (reply i
91.250.77.23 10.34.208.9 ICMP 844 Echo (ping) reply id=0x02f2, seq=1/256, ttl=49 (request
10.34.208.9 139.7.30.126 DNS 79 Standard query 0xe445 A gdata.youtube.com
139.7.30.126 10.34.208.9 DNS 283 Standard query response 0xe445 CNAME www4.l.google.com A 1
10.34.208.9 173.194.39.4 TCP 76 52524→80 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 T
173.194.39.4 10.34.208.9 TCP 76 80→52524 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SA
10.34.208.9 173.194.39.4 TCP 68 52524→80 [ACK] Seq=1 Ack=1 Win=14656 Len=0 TSval=382656 TSe
10.34.208.9 173.194.39.4 HTTP 499 GET /feeds/api/videos?q=ZZ2cftjyHys&time=all_time&format=2%
173.194.39.4 10.34.208.9 TCP 68 80→52524 [ACK] Seq=1 Ack=432 Win=15872 Len=0 TSval=88297479
173.194.39.4 10.34.208.9 HTTP 1516 HTTP/1.1 200 OK [Packet size limited during capture]
173.194.39.4 10.34.208.9 TCP 1516 80→52524 [ACK] Seq=1449 Ack=432 Win=15872 Len=1448 TSval=88
173.194.39.4 10.34.208.9 TCP 1516 80→52524 [ACK] Seq=2897 Ack=432 Win=15872 Len=1448 TSval=88
173.194.39.4 10.34.208.9 TCP 1420 80→52524 [PSH, ACK] Seq=4345 Ack=432 Win=15872 Len=1352 TSv
TShark processing
27 33.811857 91.250.77.23 -> 10.122.13.169 TCP 68 80→38644 [ACK] Seq=1 Ack=427 Win=15616 Len=0 TSval=2459653175 TSecr=4294965931
28 33.816469 91.250.77.23 -> 10.122.13.169 TCP 1436 [TCP segment of a reassembled PDU]
29 33.817856 10.122.13.169 -> 91.250.77.23 TCP 68 38644→80 [ACK] Seq=427 Ack=1369 Win=17504 Len=0 TSval=4294966089 TSecr=2459653175
30 33.816505 91.250.77.23 -> 10.122.13.169 TCP 1436 [TCP segment of a reassembled PDU]
31 33.818158 10.122.13.169 -> 91.250.77.23 TCP 68 38644→80 [ACK] Seq=427 Ack=2737 Win=20416 Len=0 TSval=4294966089 TSecr=2459653175
32 33.819430 91.250.77.23 -> 10.122.13.169 TCP 1436 [TCP segment of a reassembled PDU]
33 33.819718 10.122.13.169 -> 91.250.77.23 TCP 68 38644→80 [ACK] Seq=427 Ack=4105 Win=23296 Len=0 TSval=4294966089 TSecr=2459653175
34 33.819463 91.250.77.23 -> 10.122.13.169 HTTP 859 HTTP/1.1 200 OK (text/html)
35 33.819973 10.122.13.169 -> 91.250.77.23 TCP 68 38644→80 [ACK] Seq=427 Ack=4896 Win=26208 Len=0 TSval=4294966089 TSecr=2459653175
36 33.908841 10.122.13.169 -> 91.250.77.23 HTTP 510 GET /kepler03/css/kepler.css HTTP/1.1
37 33.919008 10.122.13.169 -> 91.250.77.23 HTTP 506 GET /kepler03/css/f1.css HTTP/1.1

Feature Spaces over communication flow graphs



Perspectives: BCI, robustness and complex anomalies

BBCI Set-up: *Let the machines learn*



$$\min_{w,b,\xi} \ \frac{1}{2}\|w\|_2^2 + \frac{C}{K}\|\xi\|_2^2$$

$$\text{subject to} \quad y_k(w^\top x_k + b) = 1 - \xi_k \quad \text{for } k = 1, \ldots, K$$

[cf. Müller et al. 2001, 2007, 2008, Dornhege et al. 2003, 2007, Blankertz et al. 2004, 2005, 2006, 2007, 2008]

Brain Computer Interfacing: ‚Brain Pong‘

Berlin Brain Computer Ínterface

- ML reduces patient training from 300h -> 5min

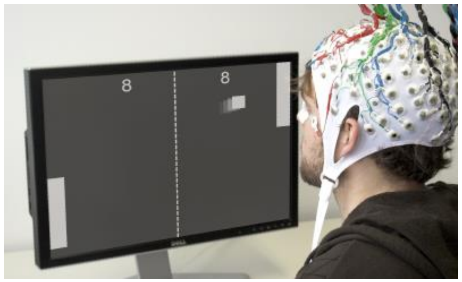
Applications

- help/hope for patients (ALS, stroke…)
- neuroscience
- neurotechnology (video coding, gaming, monitoring driving)

Leitmotiv: *›let the machines learn‹*

BCI goes out of lab

Today: In-lab Studies



Tomorrow: Out-of-lab applications

Robustness is key

Eye movement

Distractions

Swallowing

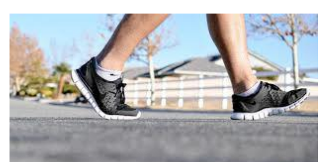
Grinding

Multi-Tasking

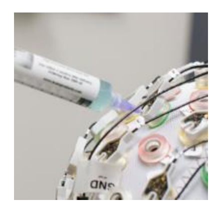
Tiredness

Blinks

Noise



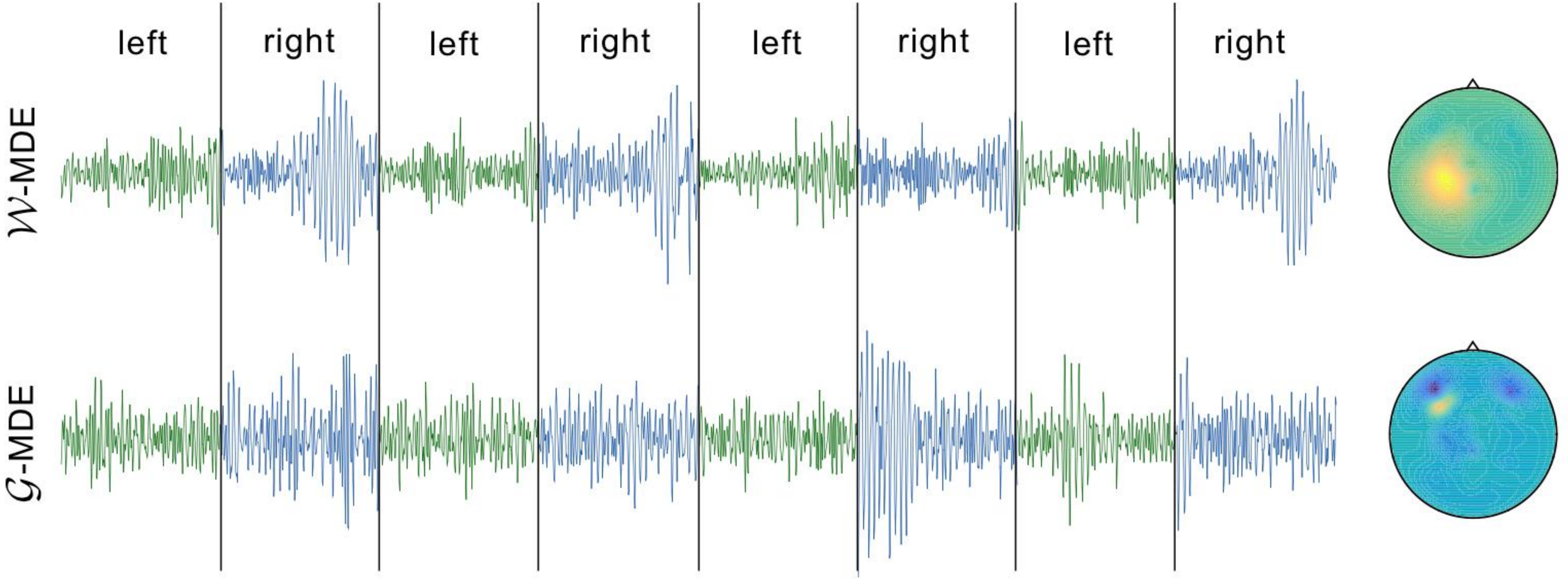
Movement



Impedance

Experiments

W-MDE better selects representative trials than G-MDE



Summary

Setting

- One-class learning is a harder task than, i.e. two class classification
- Try to learn properties of the given examples that potentially discriminates them from other

Methods

- One-class SVM learns a hyperplane that separates the data from the origin with maximum margin
- SVDD learns a center and a radius of a hypershpere that encloses the bulk of the data
- SVDD and One-class SVM are interchangable for a wide choice of kernels (including the Gaussian kernel)
- SSAD is a semi-supervised extension of SVDD: handles positiv and negative labeled examples as well as unlabeled examples

Results

- All approaches work well with high dimensions
- Incorporating prior knowledge into the learning problem (SSAD) significantly increases detection performance (not surprisingly)
- Active learning strategy