Deep learning, kernel machines and robustness

Johan Suykens

KU Leuven, ESAT-STADIUS and Leuven.AI Institute
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
Email: johan.suykens@esat.kuleuven.be
http://www.esat.kuleuven.be/stadius/

ICORS 2021, Vienna Austria, Sept 2021



Overview

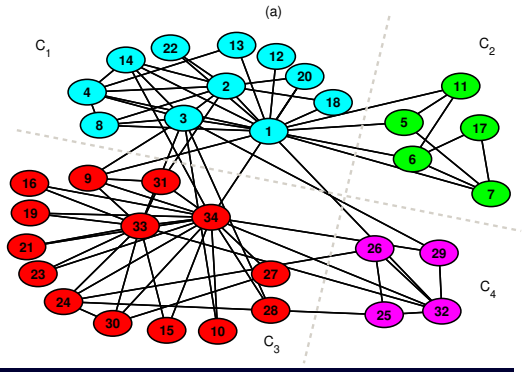
- Introduction
- Function estimation and model representations
- Least squares support vector machines (LS-SVM) as core models
- Robustness

- Restricted kernel machines (RKM), generative models
- Deep learning and kernel machines: new synergies
- Robust generative RKMs
- Conclusions

Introduction

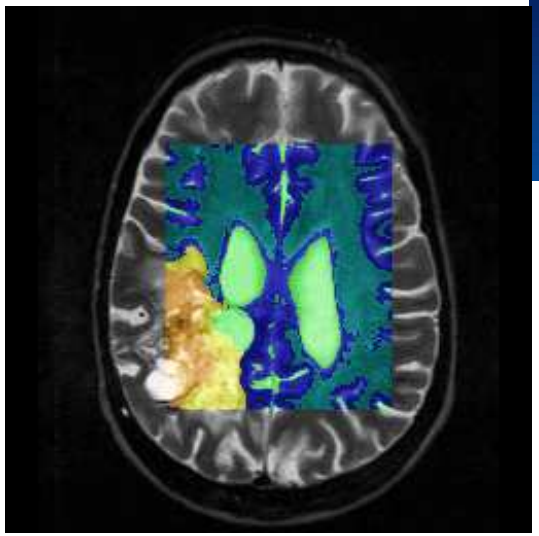
Data world

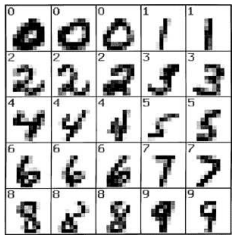



(a)
C1
C2
C3
C4
22
13
14
12
20
2
4
18
8
3
1
11
5
17
6
7
16
9
31
19
34
33
21
26
29
23
27
24
30
15
10
28
25
32



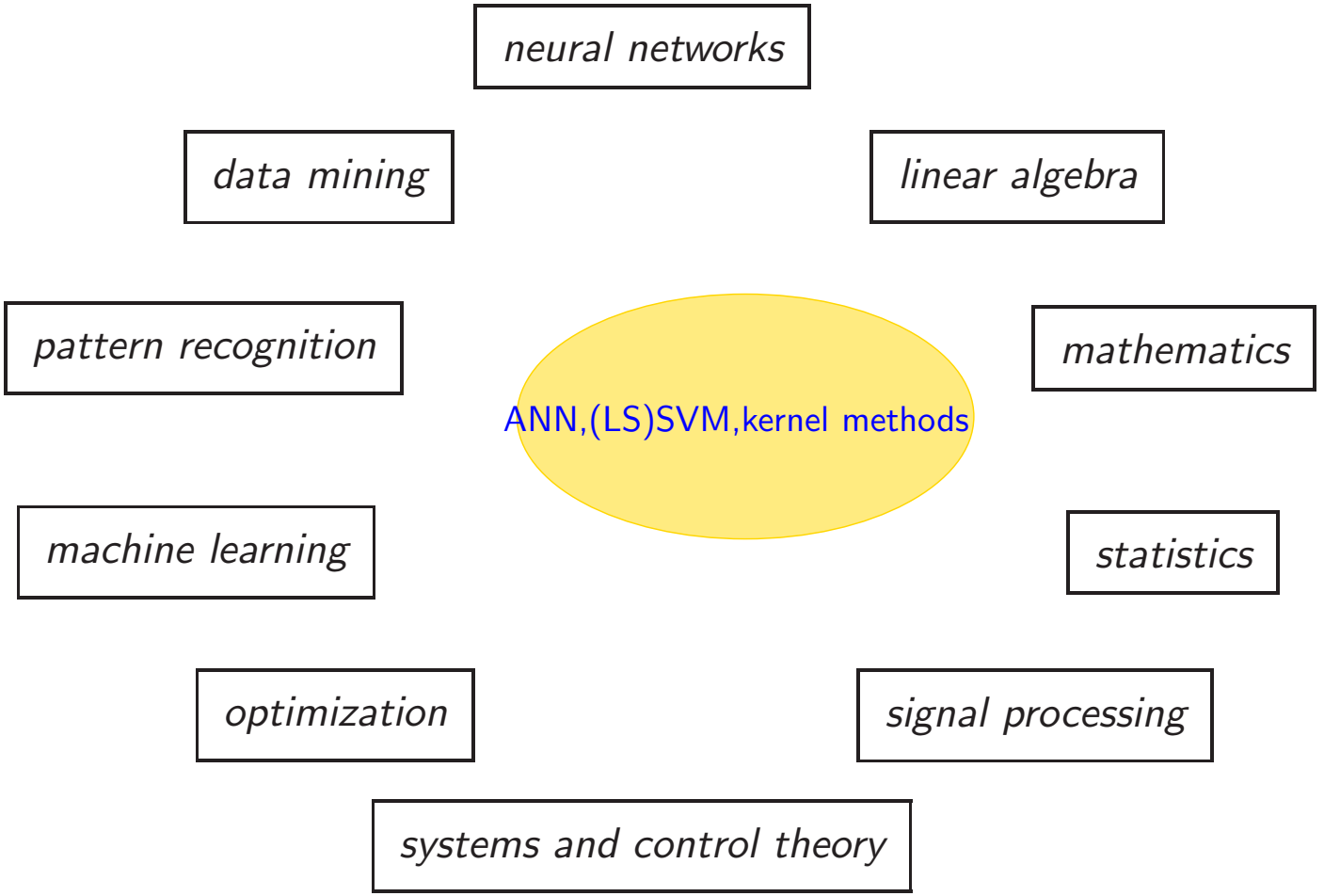






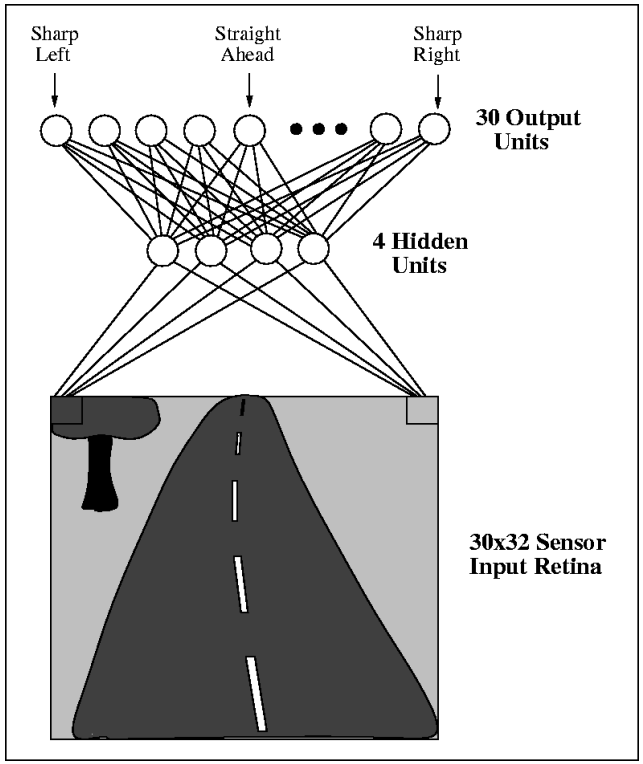


Interdisciplinary challenges



Neural networks: powerful mappings (1)

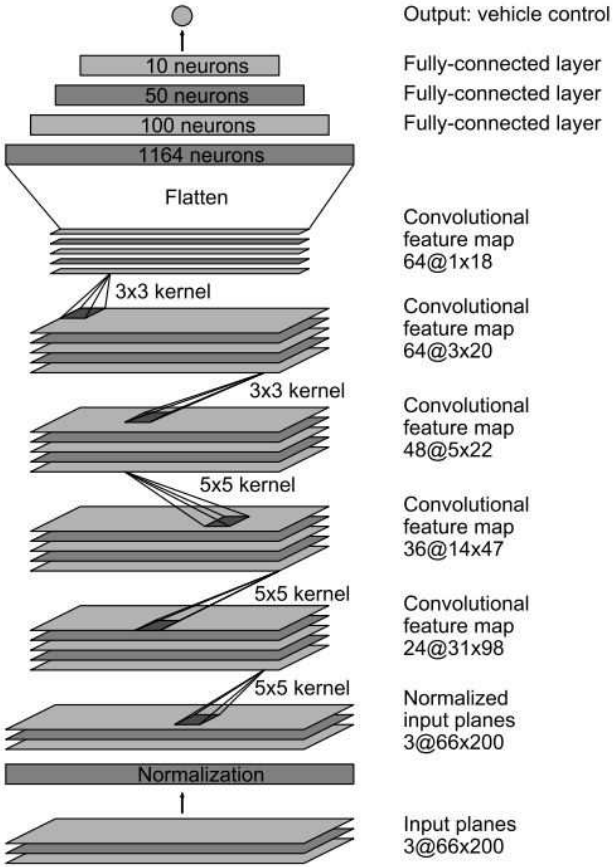
in the early days of neural networks:





ALVINN (Autonomous Land Vehicle In a Neural Network)
[Pomerleau, Neural Computation 1991]

Neural networks: powerful mappings (2)



(27 million connections)



Waymo / Google Self-Driving Car



Tesla Autopilot



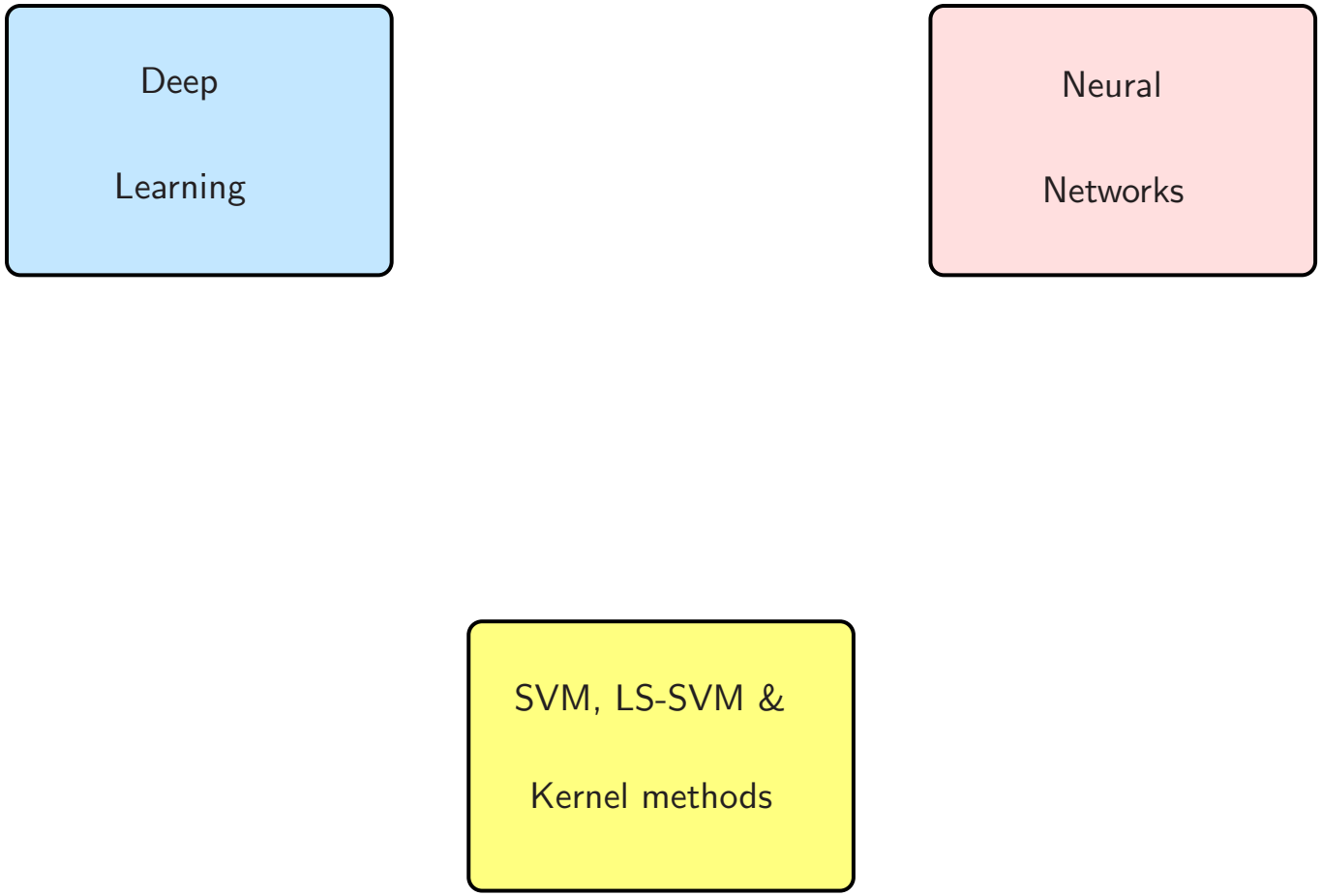
Uber



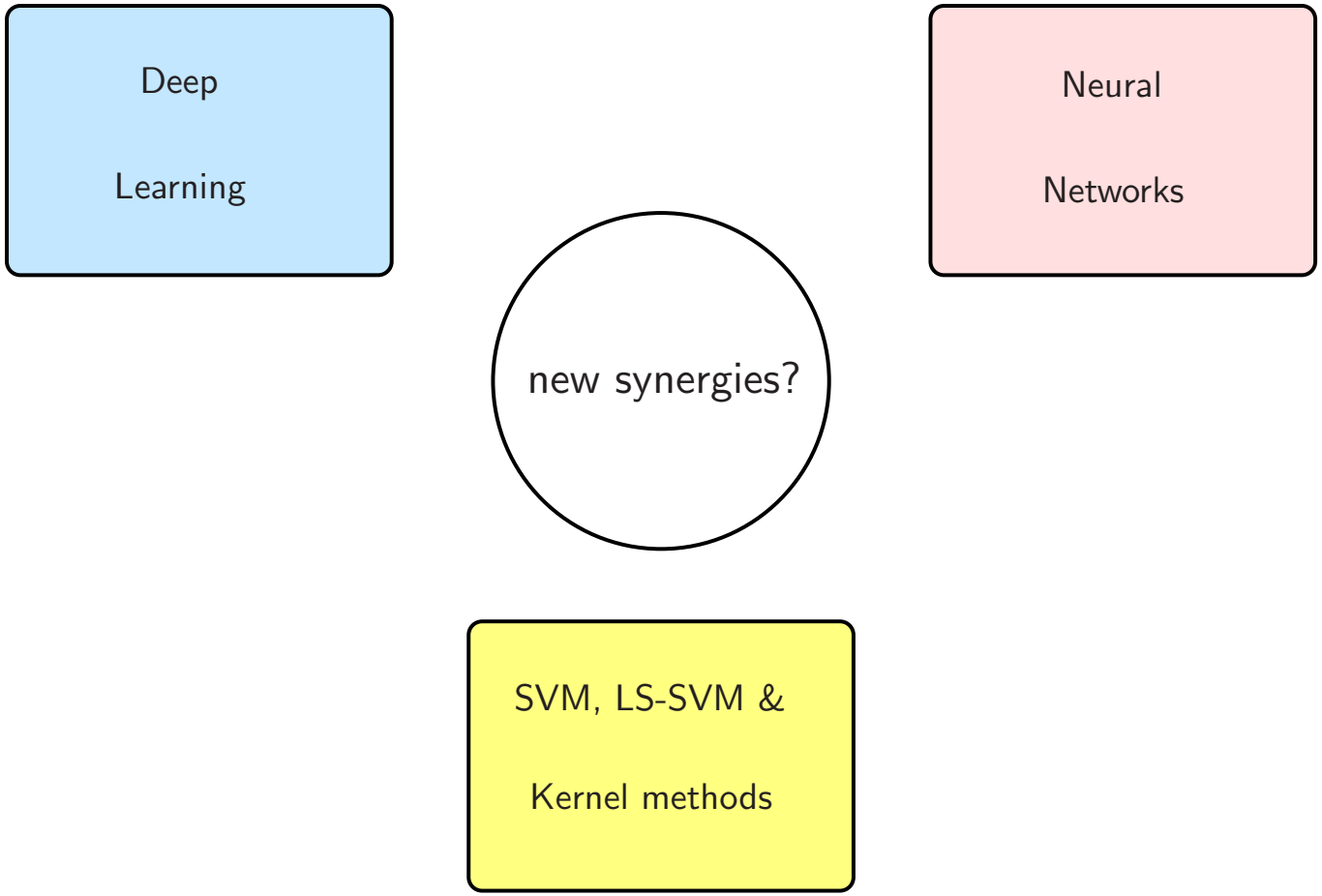
nuTonomy

from: [selfdrivingcars.mit.edu (Lex Fridman et al.), 2017] - deep learning

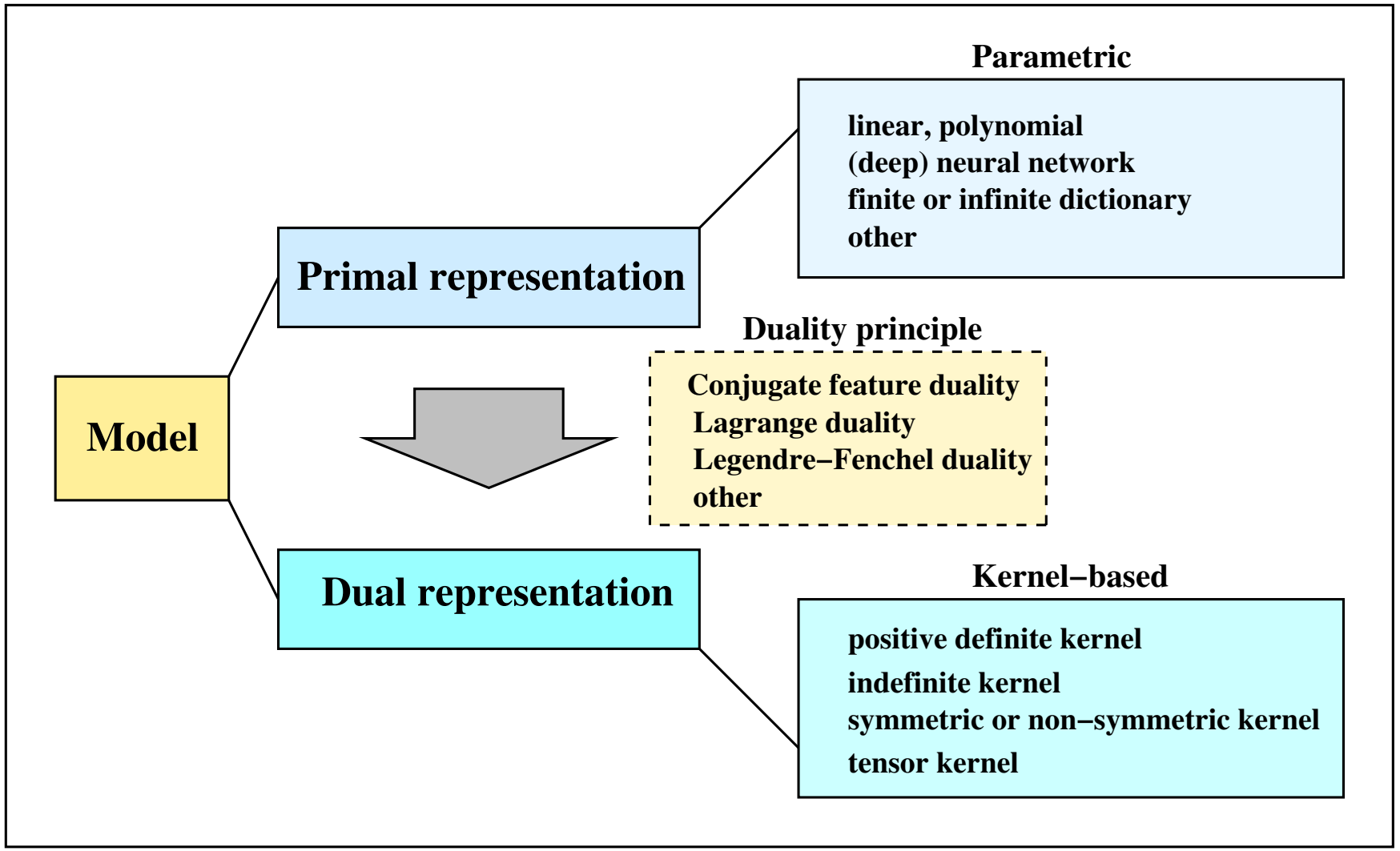
Different paradigms



Different paradigms



Towards a unifying picture



[Suykens 2017]

Function estimation and model representations

Linear function estimation (1)

- Given $\{(x_i, y_i)\}_{i=1}^N$ with $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, consider $\hat{y} = f(x)$ where f is parametrized as

$$\hat{y} = w^T x + b$$

  with $\hat{y}$ the estimated output of the linear model.

- Consider estimating w, b by

$$\min_{w,b} \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{i=1}^{N} (y_i - w^T x_i - b)^2$$

  $\rightarrow$ one can directly solve in w, b

Linear function estimation (2)

- ... or write as a constrained optimization problem:

$$\begin{array}{ll} \min\limits_{w,b,e} & \frac{1}{2}w^T w + \gamma\frac{1}{2}\sum_i e_i^2 \\ \text{subject to} & e_i = y_i - w^T x_i - b, \quad i = 1, ..., N \end{array}$$

Lagrangian: $\mathcal{L}(w, b, e_i, \alpha_i) = \frac{1}{2}w^T w + \gamma\frac{1}{2}\sum_i e_i^2 - \sum_i \alpha_i(e_i - y_i + w^T x_i + b)$

- From optimality conditions:

$$\hat{y} = \sum_i \alpha_i \, x_i^T x + b$$

where α, b follows from solving a linear system

$$\left[\begin{array}{c|c} 0 & 1_N^T \\ \hline 1_N & \Omega + I/\gamma \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right]$$

with $\Omega_{ij} = x_i^T x_j$ for $i, j = 1, ..., N$ and $y = [y_1; ...; y_N]$.

Linear model: solving in primal or dual?

inputs $x \in \mathbb{R}^d$, output $y \in \mathbb{R}$
training set $\{(x_i, y_i)\}_{i=1}^N$

$$(P): \quad \hat{y} = w^T x + b, \qquad w \in \mathbb{R}^d$$

Model $\nearrow$

Linear model: solving in primal or dual?

inputs $x \in \mathbb{R}^d$, output $y \in \mathbb{R}$
training set $\{(x_i, y_i)\}_{i=1}^N$

$$\text{Model} \begin{cases} \nearrow & (P): \quad \hat{y} = w^T x + b, \qquad w \in \mathbb{R}^d \\ \searrow & (D): \quad \hat{y} = \sum_i \alpha_i \, x_i^T x + b, \qquad \alpha \in \mathbb{R}^N \end{cases}$$

Linear model: solving in primal or dual?

few inputs, many data points: $d \ll N$

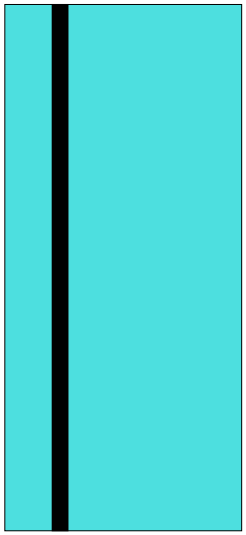


primal: $w \in \mathbb{R}^d$

dual: $\alpha \in \mathbb{R}^N$ (large kernel matrix: $N \times N$)

Linear model: solving in primal or dual?

many inputs, few data points: $d \gg N$



primal: $w \in \mathbb{R}^d$

dual: $\alpha \in \mathbb{R}^N$ (small kernel matrix: $N \times N$)

Feature map and kernel

From linear to nonlinear model:

$$\text{Model} \begin{cases} \nearrow & (P): \quad \hat{y} = w^T \varphi(x) + b \\ \searrow & (D): \quad \hat{y} = \sum_i \alpha_i \, K(x_i, x) + b \end{cases}$$

Mercer theorem (one can **either** choose φ **or** positive definite K):

$$K(x, z) = \varphi(x)^T \varphi(z)$$

Feature map φ, Kernel function $K(x, z)$ (e.g. linear, polynomial, RBF, ...)

- SVMs: feature map and positive definite kernel [Cortes & Vapnik, 1995]
- Neural networks: hidden layer as feature map [Suykens & Vandewalle, IEEE-TNN 1999]
- Least squares support vector machines [Suykens et al., 2002]: L_2 loss and regularization

Least Squares Support Vector Machines: "core models"

- Regression

$$\min_{w,b,e} w^T w + \gamma \sum_i e_i^2 \quad \text{s.t.} \quad y_i = w^T \varphi(x_i) + b + e_i, \quad \forall i$$

- Classification

$$\min_{w,b,e} w^T w + \gamma \sum_i e_i^2 \quad \text{s.t.} \quad y_i(w^T \varphi(x_i) + b) = 1 - e_i, \quad \forall i$$

- Kernel pca $(V = I)$, Kernel spectral clustering $(V = D^{-1})$

$$\min_{w,b,e} -w^T w + \gamma \sum_i v_i e_i^2 \quad \text{s.t.} \quad e_i = w^T \varphi(x_i) + b, \quad \forall i$$

- Kernel canonical correlation analysis/partial least squares

$$\min_{w,v,b,d,e,r} w^T w + v^T v + \nu \sum_i (e_i - r_i)^2 \text{ s.t. } \begin{cases} e_i &= w^T \varphi^{(1)}(x_i) + b \\ r_i &= v^T \varphi^{(2)}(y_i) + d \end{cases}$$

[Suykens & Vandewalle, NPL 1999; Suykens et al., 2002; Alzate & Suykens, 2010]

Sparsity: through regularization or loss function

- through regularization: model $\hat{y} = w^T x + b$

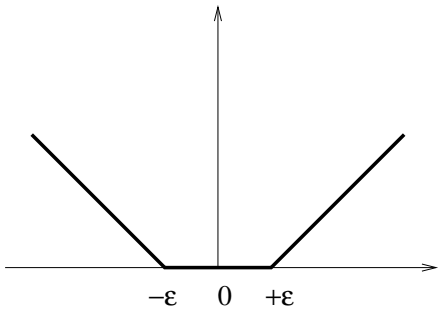
$$\min \sum_j |w_j| + \gamma \sum_i e_i^2$$

$\Rightarrow$ sparse w (e.g. Lasso)

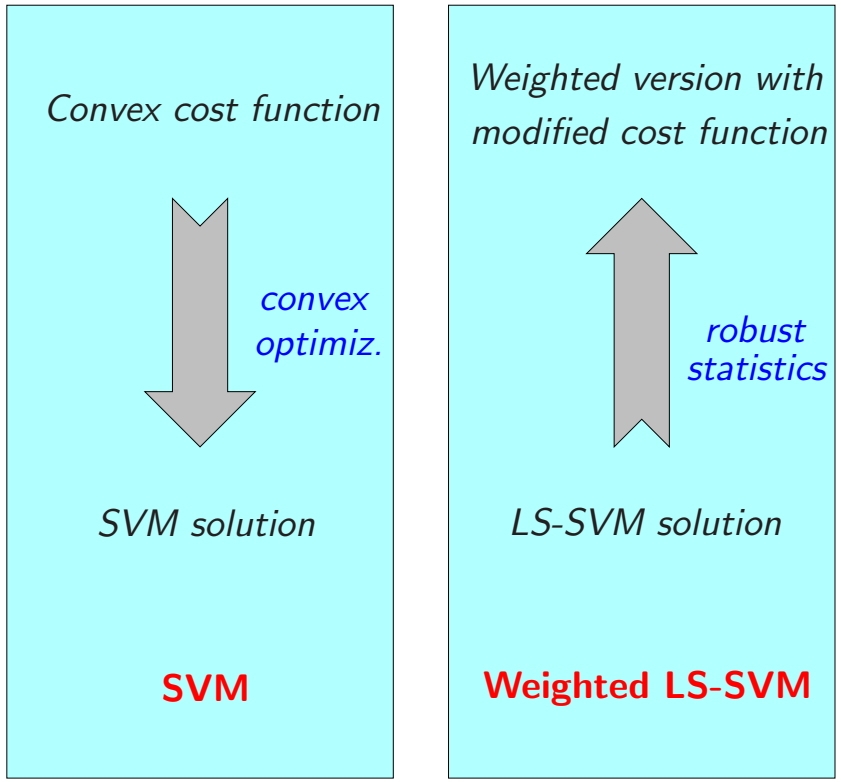
- through loss function: model $\hat{y} = \sum_i \alpha_i K(x, x_i) + b$

$$\min \; w^T w + \gamma \sum_i L(e_i)$$

$\Rightarrow$ sparse α (e.g. SVM)



Weighted versions and robustness

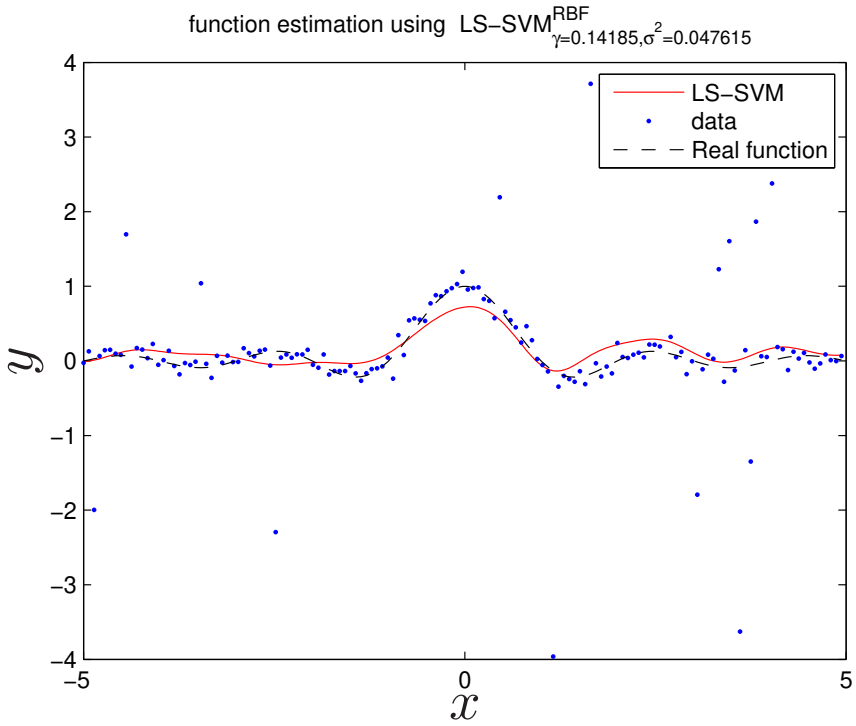


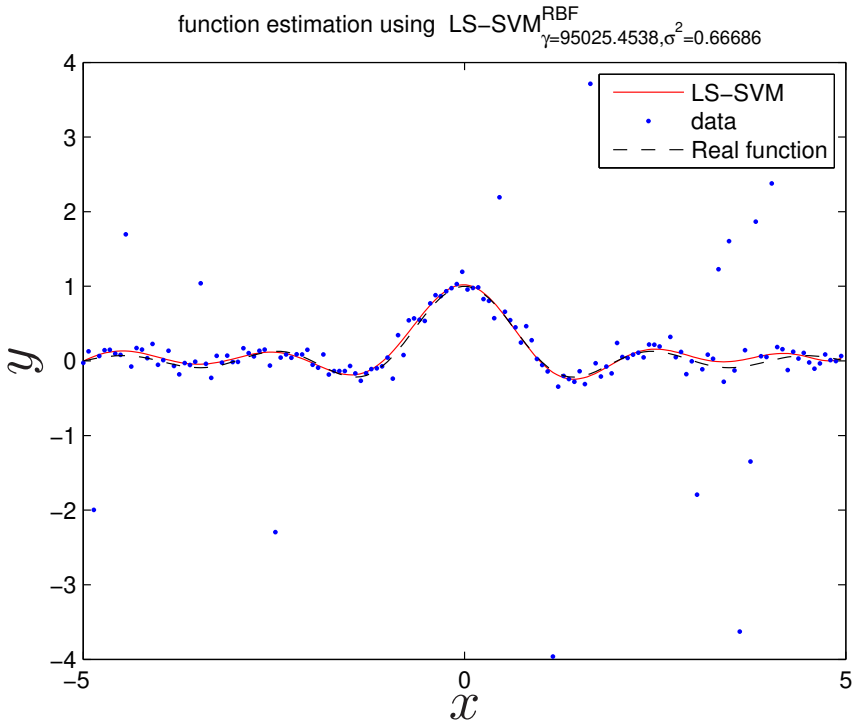
- Weighted LS-SVM: $\min_{w,b,e} \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{i=1}^{N} v_i e_i^2 \ \text{s.t.} \ y_i = w^T \varphi(x_i) + b + e_i, \ \forall i$

  with v_i determined from $\{e_i\}_{i=1}^N$ of unweighted LS-SVM [Suykens et al., 2002].

  Robustness and stability [Debruyne et al., JMLR 2008, 2010].
- SVM solution by applying iteratively weighted LS [Perez-Cruz et al., 2005]

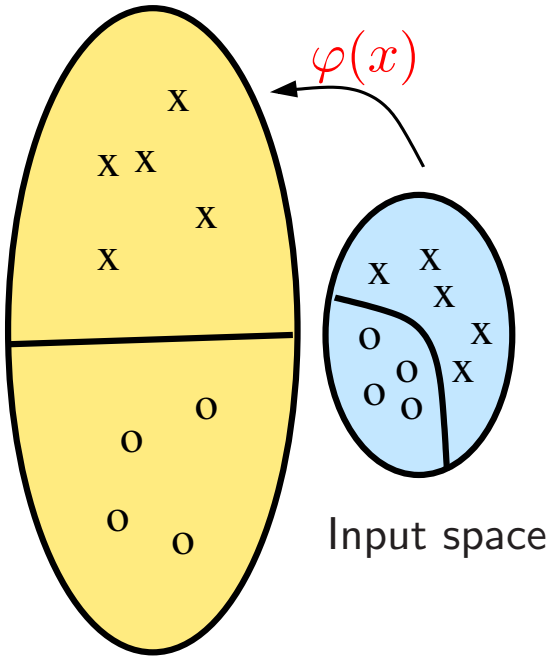
Example: robust regression using weighted LS-SVM





using LS-SVMlab v1.8 http://www.esat.kuleuven.be/sista/lssvmlab/

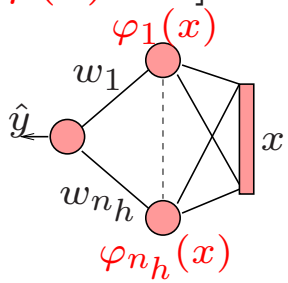
SVMs and neural networks



Primal space

Parametric

$$\hat{y} = \text{sign}[w^T \varphi(x) + b]$$



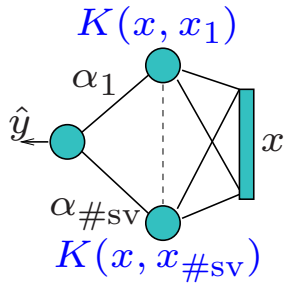
Parametric

$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$ (“Kernel trick”)

Dual space

Nonparametric

$$\hat{y} = \text{sign}[\sum_{i=1}^{\#\text{sv}} \alpha_i y_i K(x, x_i) + b]$$



Non–parametric

[Suykens et al., 2002]

Function estimation in RKHS

- Find function f such that [Wahba, 1990; Evgeniou et al., 2000]

$$\min_{f \in \mathcal{H}_K} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \|f\|_K^2$$

  with $L(\cdot, \cdot)$ the loss function. $\|f\|_K$ is norm in RKHS $\mathcal{H}_K$ defined by K.

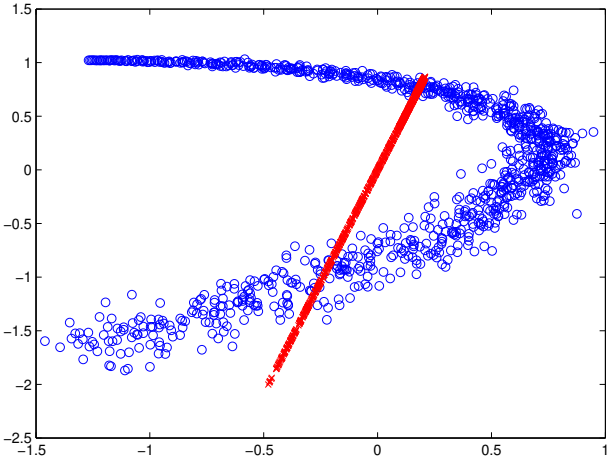
- Representer theorem: for convex loss function, solution of the form

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)$$

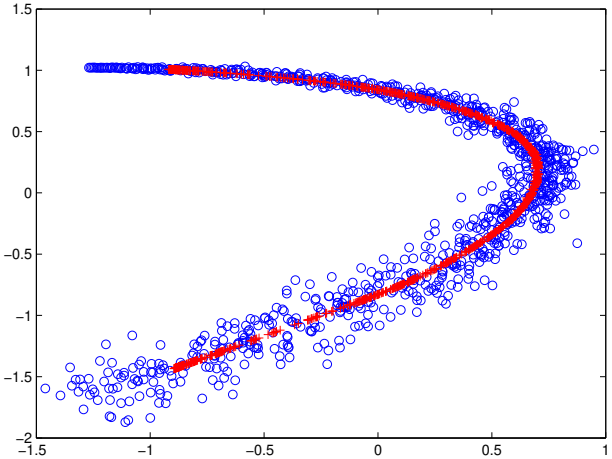
  Reproducing property $f(x) = \langle f, K_x \rangle_K$ with $K_x(\cdot) = K(x, \cdot)$

- Sparse representation by hinge and ϵ-insensitive loss [Vapnik, 1998]

Kernel principal component analysis (KPCA)



linear PCA



kernel PCA (RBF kernel)

Kernel PCA [Schölkopf et al., 1998]:
take eigenvalue decomposition of the kernel matrix

$$\begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_N) \\ \vdots & & \vdots \\ K(x_N, x_1) & \dots & K(x_N, x_N) \end{bmatrix}$$

(applications in dimensionality reduction and denoising)

Kernel PCA

- Primal problem: [Suykens et al., 2002]

$$\min_{w,b,e} \ \frac{1}{2}w^T w - \frac{1}{2}\gamma \sum_{i=1}^{N} e_i^2 \ \text{ s.t. } \ e_i = w^T \varphi(x_i) + b, \ i = 1, ..., N.$$

- Dual problem corresponds to kernel PCA [Scholkopf et al., 1998]

$$\Omega_c \alpha = \lambda \alpha \ \text{ with } \ \lambda = 1/\gamma$$

with $\Omega_{c,ij} = (\varphi(x_i) - \hat{\mu}_\varphi)^T (\varphi(x_j) - \hat{\mu}_\varphi)$ the *centered kernel matrix*.

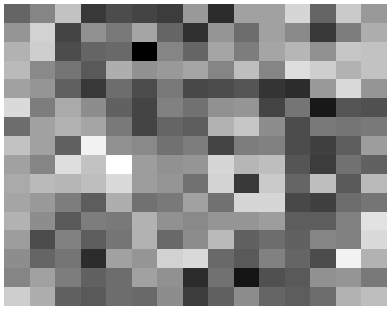
- Robust and sparse versions [Alzate & Suykens, 2008]: by taking other loss functions

Robustness: Kernel Component Analysis

original image



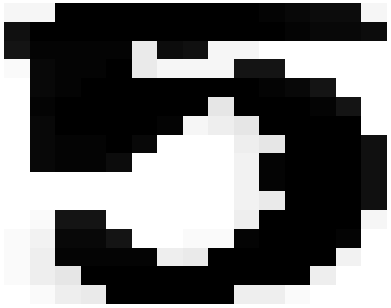
corrupted image



KPCA reconstruction



KCA reconstruction

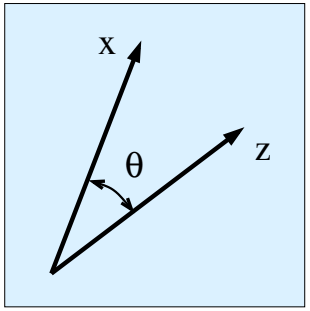


Weighted LS-SVM [Alzate & Suykens, IEEE-TNN 2008]: robustness and sparsity

Advantages of kernel-based setting

- **model-based** approach
- **out-of-sample extensions**, applying model to new data
- consider **training, validation and test data**
  (training problem corresponds to eigenvalue decomposition problem)
- model selection procedures
- **sparse representations and large scale methods**

Wider use of the "kernel trick"



- **Angle between vectors:** (e.g. correlation analysis)
  Input space:

$$\cos\theta_{xz} = \frac{x^T z}{\|x\|_2 \|z\|_2}$$

  Feature space:

$$\cos\theta_{\varphi(x),\varphi(z)} = \frac{\varphi(x)^T \varphi(z)}{\|\varphi(x)\|_2 \|\varphi(z)\|_2} = \frac{K(x,z)}{\sqrt{K(x,x)}\sqrt{K(z,z)}}$$

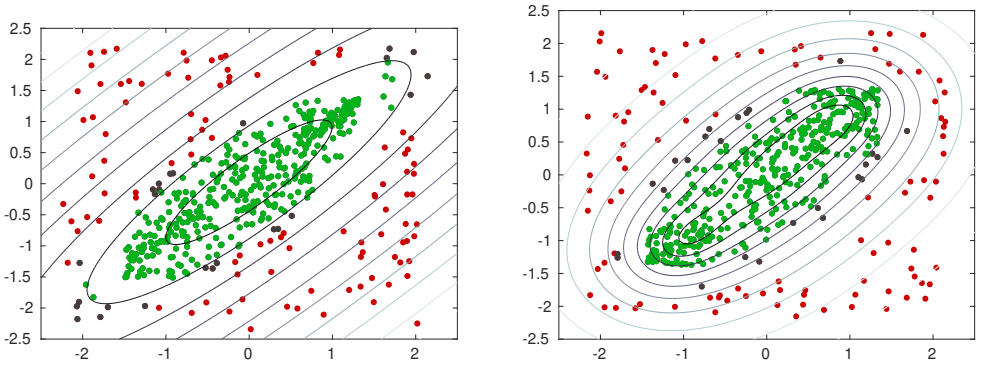
- **Distance between vectors:** (e.g. for "kernelized" clustering methods)
  Input space:

$$\|x - z\|_2^2 = (x-z)^T(x-z) = x^T x + z^T z - 2x^T z$$

  Feature space:

$$\|\varphi(x) - \varphi(z)\|_2^2 = K(x,x) + K(z,z) - 2K(x,z)$$

Kernel MRCD



- Minimum regularized covariance determinant method (**MRCD**): robust estimator for multivariate location and scatter, detecting outliers by fitting a robust covariance matrix

- MRCD assumption: non-outlying observations are elliptically distributed

- **Kernel MRCD**: not restricted to elliptical data

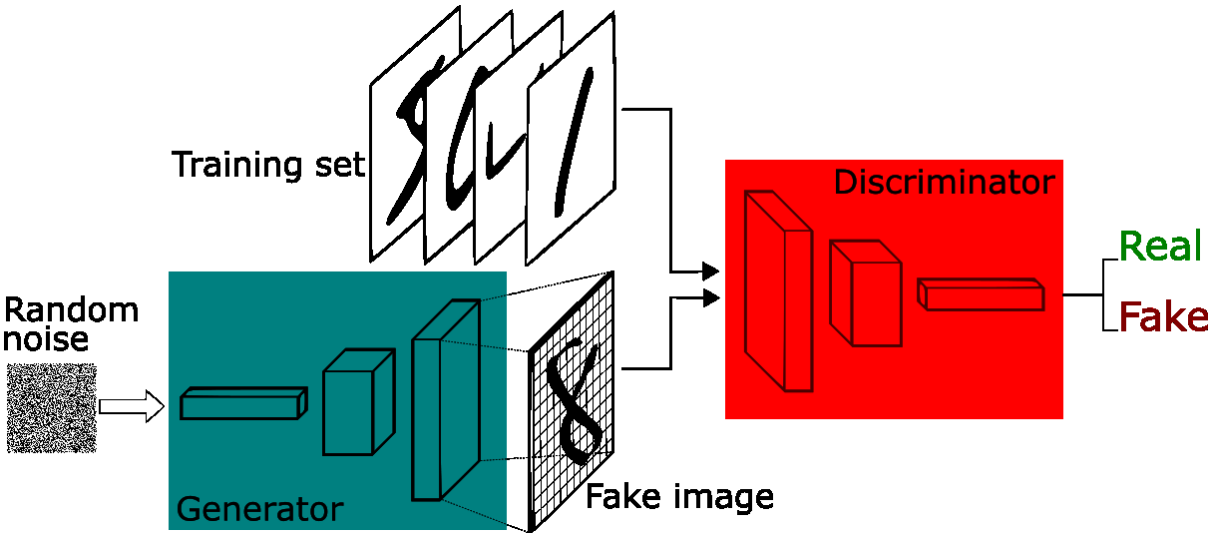
[Schreurs J., Vranckx I., Hubert M., Suykens J.A.K., Rousseeuw P.J.. Outlier detection in non-elliptical data by kernel MRCD. Stat Comput 31, 66, 2021]

Generative models

Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN) [Goodfellow et al., 2014]
Training of two competing models in a zero-sum game:

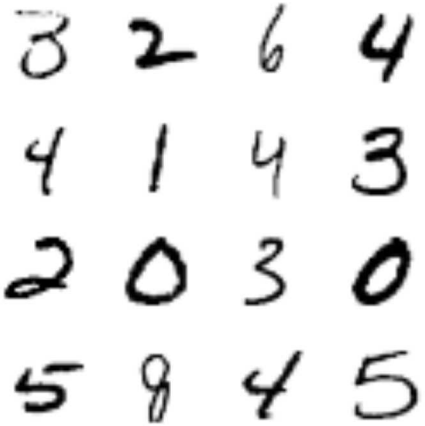
(Generator) generate fake output examples from random noise
(Discriminator) discriminate between fake examples and real examples.



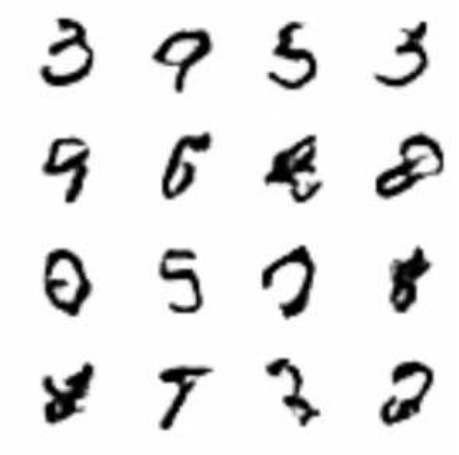
source: https://deeplearning4j.org/generative-adversarial-network

GAN: example on MNIST

MNIST training data:

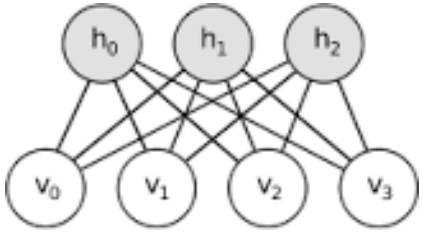


GAN generated examples:



source: https://www.kdnuggets.com/2016/07/mnist-generative-adversarial-model-keras.html

Restricted Boltzmann Machines (RBM)



- Markov random field, bipartite graph, stochastic binary units
  Layer of visible units v and layer of hidden units h
  No hidden-to-hidden connections
- Energy:

$$E(v, h; \theta) = -v^T W h - c^T v - a^T h \quad \text{with} \quad \theta = \{W, c, a\}$$

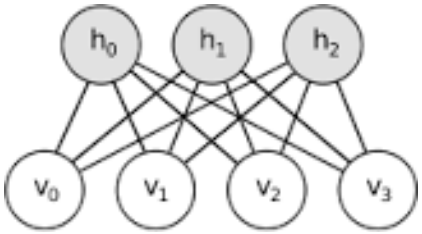
Joint distribution:

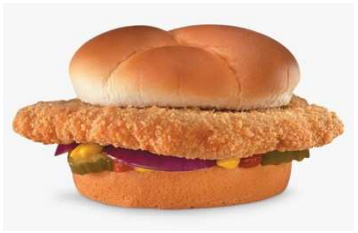
$$P(v, h; \theta) = \frac{1}{Z(\theta)} \exp(-E(v, h; \theta))$$

with partition function $Z(\theta) = \sum_v \sum_h \exp(-E(v, h; \theta))$

[Hinton, Osindero, Teh, Neural Computation 2006]

Restricted Boltzmann Machines (RBM)





- Markov random field, bipartite graph, stochastic binary units
  Layer of visible units v and layer of hidden units h
  No hidden-to-hidden connections
- Energy:

$$E(v, h; \theta) = -v^T W h - c^T v - a^T h \quad \text{with} \quad \theta = \{W, c, a\}$$

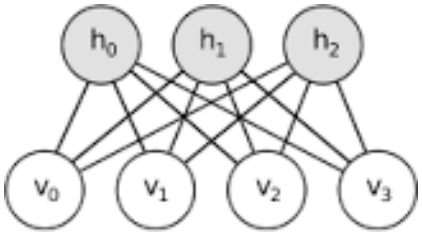
Joint distribution:

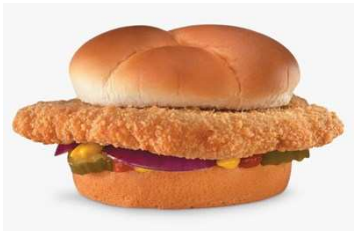
$$P(v, h; \theta) = \frac{1}{Z(\theta)} \exp(-E(v, h; \theta))$$

with partition function $Z(\theta) = \sum_v \sum_h \exp(-E(v, h; \theta))$

[Hinton, Osindero, Teh, Neural Computation 2006]

Restricted Boltzmann Machines (RBM)





- Markov random field, bipartite graph, stochastic binary units
  Layer of visible units v and layer of hidden units h
  No hidden-to-hidden connections
- Energy:

$$E(v,h;\theta) = -(v^T W)h - c^T v - a^T h \quad \text{with} \quad \theta = \{W, c, a\}$$

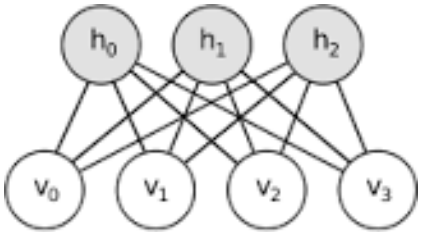
Joint distribution:

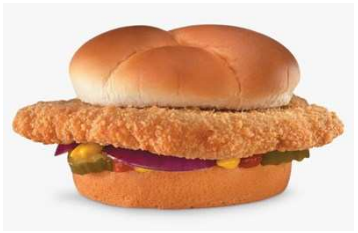
$$P(v,h;\theta) = \frac{1}{Z(\theta)} \exp(-E(v,h;\theta))$$

with partition function $Z(\theta) = \sum_v \sum_h \exp(-E(v,h;\theta))$

[Hinton, Osindero, Teh, Neural Computation 2006]

Restricted Boltzmann Machines (RBM)





- Markov random field, bipartite graph, stochastic binary units
  Layer of visible units v and layer of hidden units h
  No hidden-to-hidden connections
- Energy:

$$E(v,h;\theta) = -v^T(Wh) - c^T v - a^T h \quad \text{with} \quad \theta = \{W, c, a\}$$

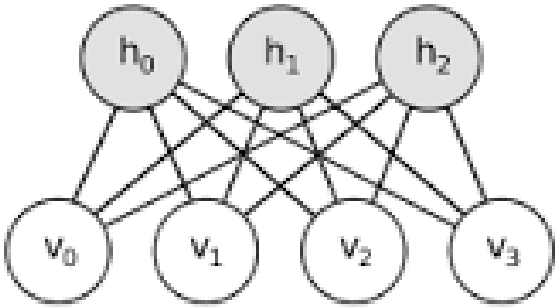
Joint distribution:

$$P(v,h;\theta) = \frac{1}{Z(\theta)} \exp(-E(v,h;\theta))$$

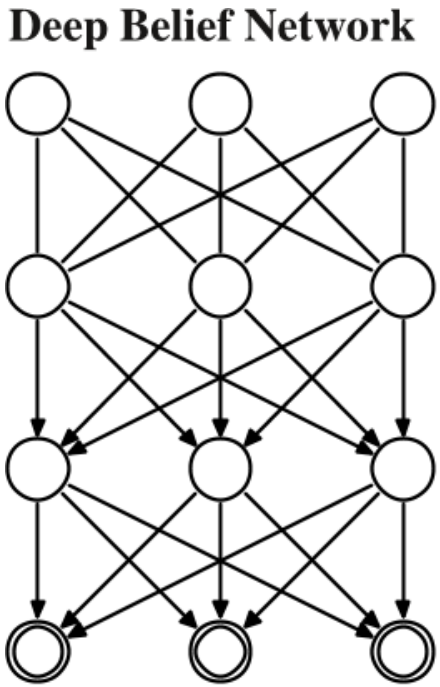
with partition function $Z(\theta) = \sum_v \sum_h \exp(-E(v,h;\theta))$

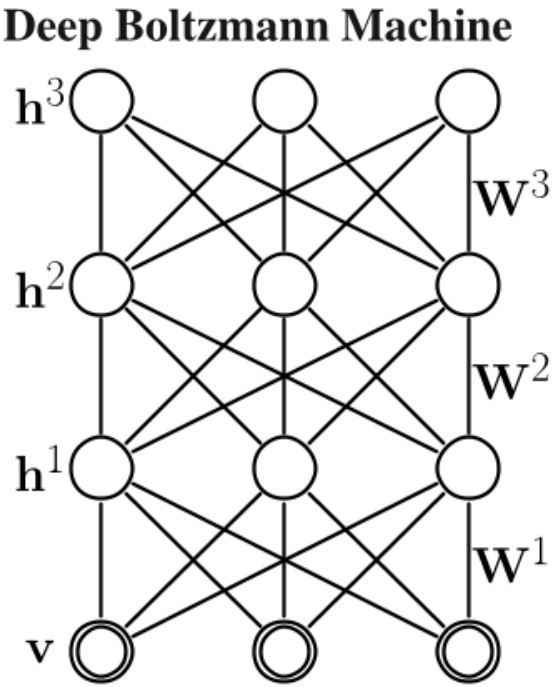
[Hinton, Osindero, Teh, Neural Computation 2006]

RBM and deep learning



$p(v, h)$



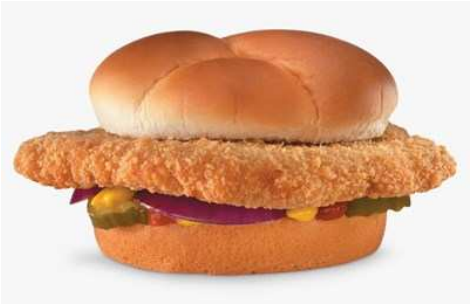


$p(v, h^1, h^2, h^3, ...)$

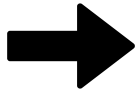
[Hinton et al., 2006; Salakhutdinov, 2015]

in other words ...

"sandwich"



$$E = -v^T W h$$



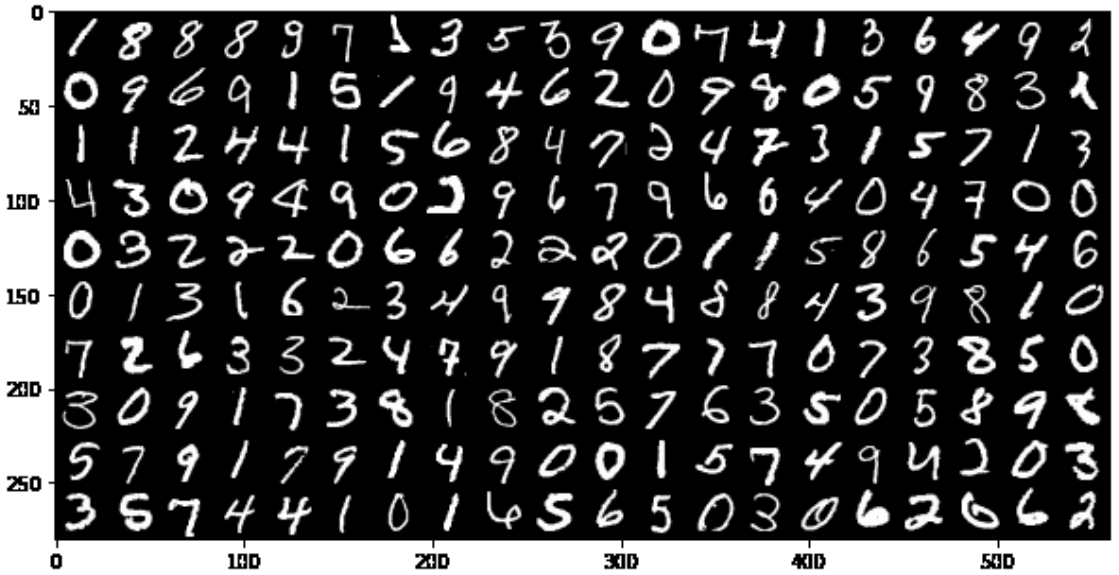
"deep sandwich"



$$E = -v^T W^1 h^1 - h^{1T} W^2 h^2 - h^{2T} W^3 h^3$$

RBM: example on MNIST

MNIST training data:



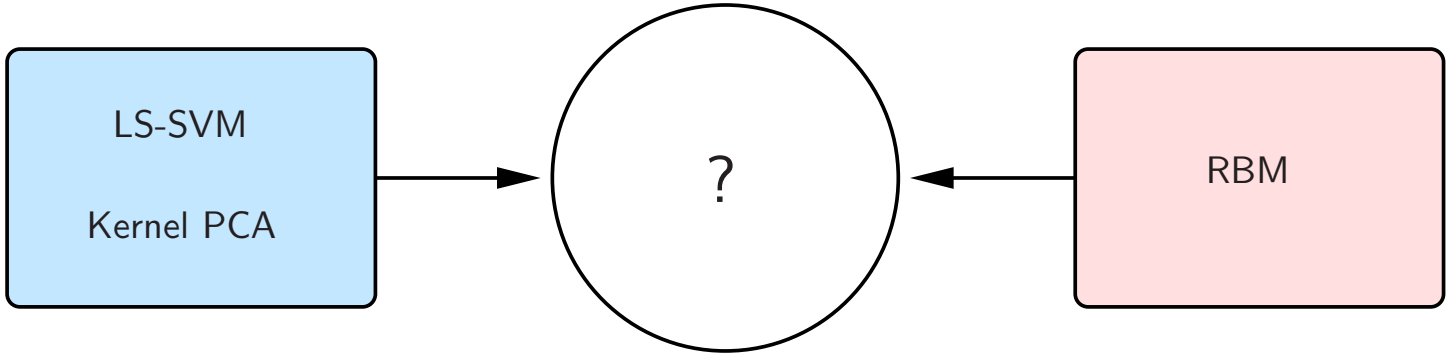
Generating new images:



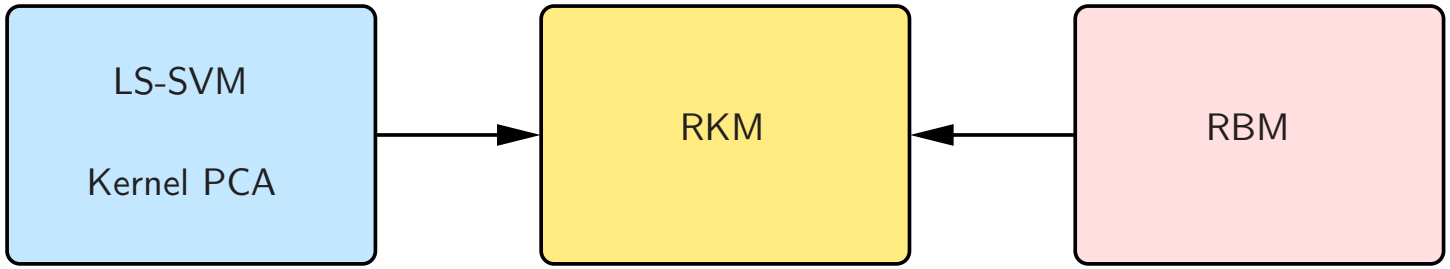
source: https://www.kaggle.com/nicw102168/restricted-boltzmann-machine-rbm-on-mnist

Restricted kernel machines

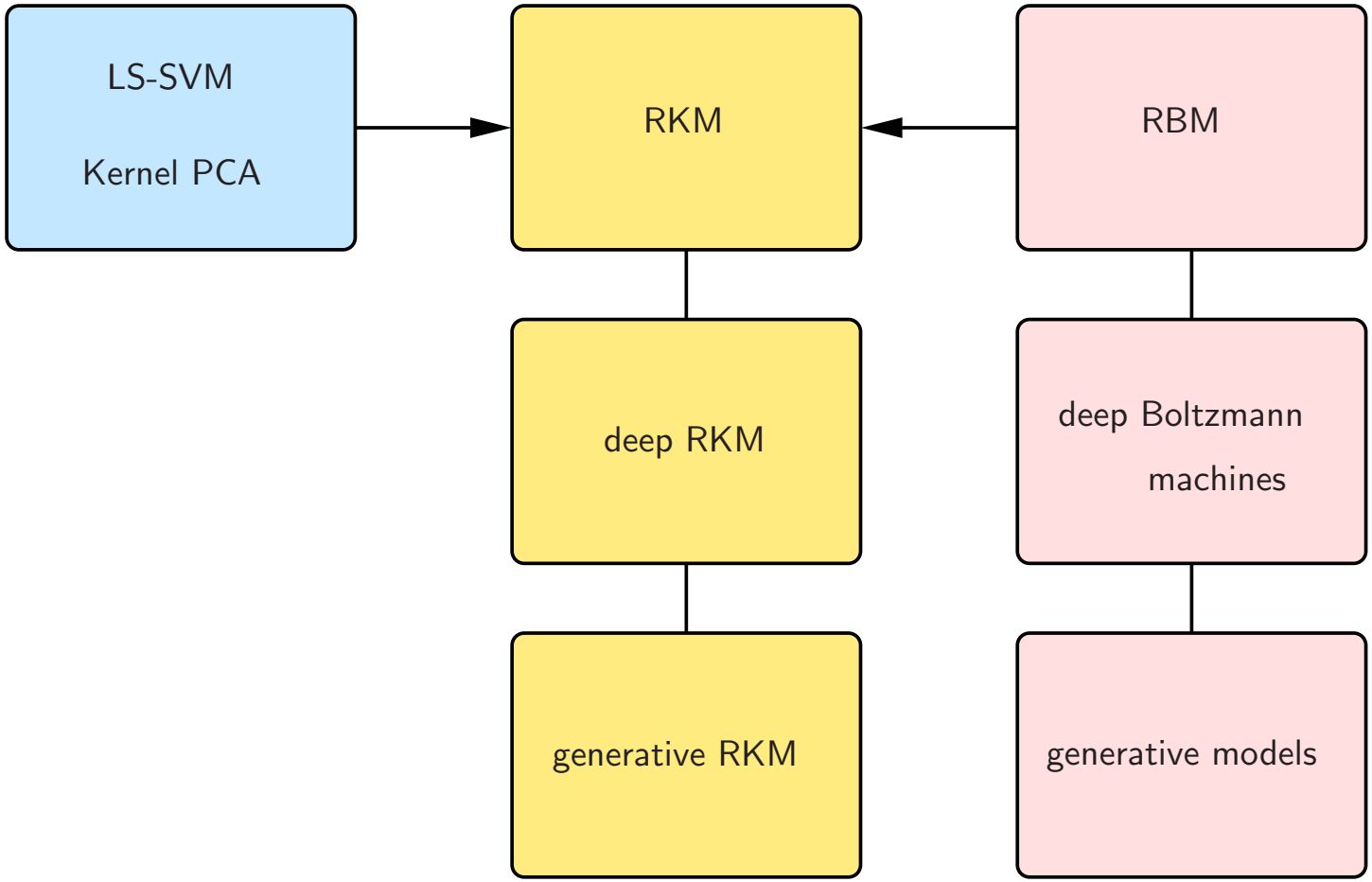
New connections



New connections



New connections



Restricted Kernel Machines (RKM)

- Kernel machine interpretations in terms of **visible and hidden units** (similar to Restricted Boltzmann Machines (**RBM**))

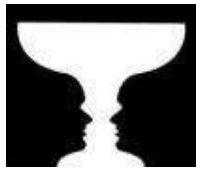
- Restricted Kernel Machine (**RKM**) representations for
  - LS-SVM regression/classification
  - Kernel PCA
  - Matrix SVD
  - Parzen-type models
  - other

- Based on principle of **conjugate feature duality** (with hidden features corresponding to dual variables)

- Deep Restricted Kernel Machines (Deep RKM)

[Suykens, Neural Computation, 2017]

From KPCA to RKM representation (1)



Model:

$e = W^T \varphi(x)$

objective J
$=$ regularization term $\mathrm{Tr}(W^T W)$
- $(\frac{1}{\lambda})$ variance term $\sum_i e_i^T e_i$

$\downarrow$ use property $e^T h \leq \frac{1}{2\lambda} e^T e + \frac{\lambda}{2} h^T h$

RKM representation:

$e = \sum_j h_j K(x_j, x)$

obtain $J \leq \overline{J}(h_i, W)$
solution from stationary points of $\overline{J}$:
$\frac{\partial \overline{J}}{\partial h_i} = 0$, $\frac{\partial \overline{J}}{\partial W} = 0$

From KPCA to RKM representation (2)

- Objective

$$
\begin{aligned}
J &= \frac{\eta}{2}\mathrm{Tr}(W^TW) - \frac{1}{2\lambda}\sum_{i=1}^{N} e_i^T e_i \text{ s.t. } e_i = W^T\varphi(x_i), \forall i \\
&\leq -\sum_{i=1}^{N} e_i^T h_i + \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\mathrm{Tr}(W^TW) \text{ s.t. } e_i = W^T\varphi(x_i), \forall i \\
&= -\sum_{i=1}^{N} \varphi(x_i)^T W h_i + \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\mathrm{Tr}(W^TW) \triangleq \overline{J}
\end{aligned}
$$

- Stationary points of $\overline{J}(h_i, W)$:

$$
\begin{cases}
\dfrac{\partial \overline{J}}{\partial h_i} = 0 \quad \Rightarrow \quad W^T\varphi(x_i) = \lambda h_i, \ \forall i \\
\dfrac{\partial \overline{J}}{\partial W} = 0 \quad \Rightarrow \quad W = \dfrac{1}{\eta}\sum_i \varphi(x_i) h_i^T
\end{cases}
$$

From KPCA to RKM representation (3)

- Elimination of W gives the eigenvalue decomposition:

$$\frac{1}{\eta} K H^T = H^T \Lambda$$

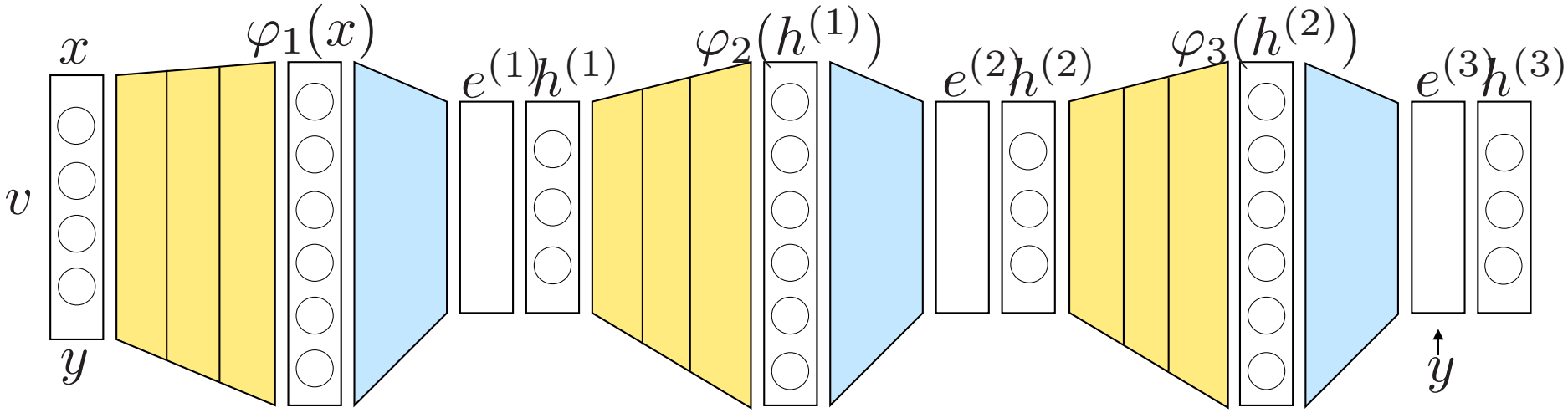
where $H = [h_1 ... h_N] \in \mathbb{R}^{s \times N}$ and $\Lambda = \mathrm{diag}\{\lambda_1, ..., \lambda_s\}$ with $s \leq N$

- Primal and dual model representations

$$\mathcal{M} \begin{cases} \nearrow & (P)_{\mathrm{RKM}}: \quad \hat{e} = W^T \varphi(x) \\ \searrow & (D)_{\mathrm{RKM}}: \quad \hat{e} = \frac{1}{\eta} \sum_j h_j K(x_j, x). \end{cases}$$

Deep Restricted Kernel Machines

Deep RKM: example



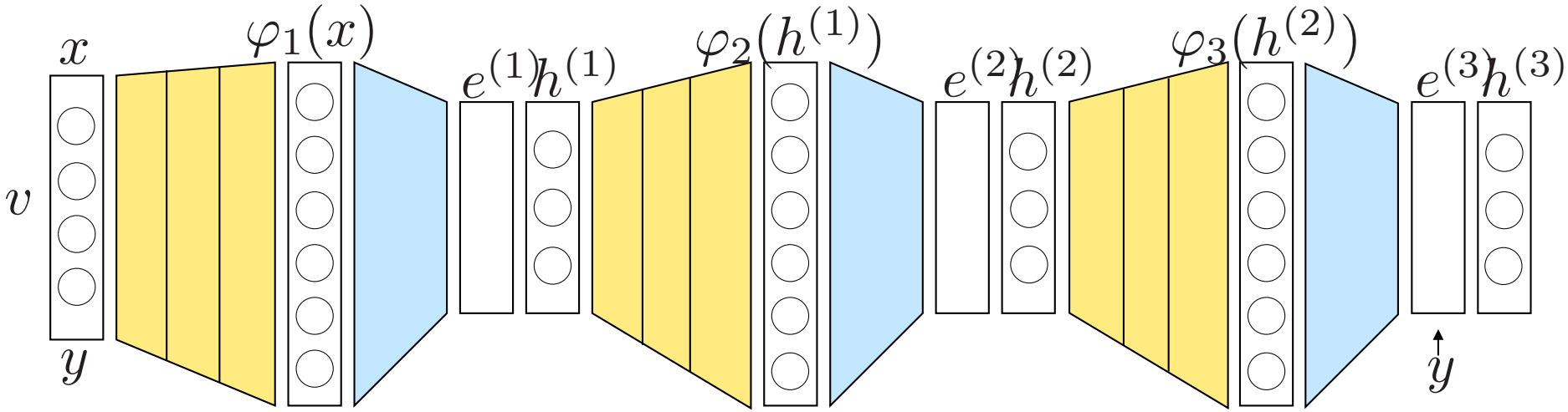
Deep RKM: KPCA + KPCA + LSSVM [Suykens, 2017]

Coupling of RKMs by taking sum of the objectives

$$J_{\text{deep}} = \overline{J}_1 + \overline{J}_2 + \underline{J}_3$$

Multiple *levels* and multiple *layers* per level.

in more detail ...



$$
\begin{aligned}
J_{\text{deep}} \quad = \quad & -\sum_{i=1}^{N} \varphi_1(x_i)^T W_1 h_i^{(1)} + \frac{\lambda_1}{2} \sum_{i=1}^{N} {h_i^{(1)}}^T h_i^{(1)} + \frac{\eta_1}{2} \text{Tr}(W_1^T W_1) \\
& -\sum_{i=1}^{N} \varphi_2(h_i^{(1)})^T W_2 h_i^{(2)} + \frac{\lambda_2}{2} \sum_{i=1}^{N} {h_i^{(2)}}^T h_i^{(2)} + \frac{\eta_2}{2} \text{Tr}(W_2^T W_2) \\
& +\sum_{i=1}^{N} (y_i^T - \varphi_3(h_i^{(2)})^T W_3 - b^T) h_i^{(3)} - \frac{\lambda_3}{2} \sum_{i=1}^{N} {h_i^{(3)}}^T h_i^{(3)} + \frac{\eta_3}{2} \text{Tr}(W_3^T W_3)
\end{aligned}
$$

Primal and dual model representations

$$
\mathcal{M} \begin{cases} \nearrow & (P)_{\text{DeepRKM}}: \quad \begin{aligned} \hat{e}^{(1)} &= W_1^T \varphi_1(x) \\ \hat{e}^{(2)} &= W_2^T \varphi_2(\Lambda_1^{-1} \hat{e}^{(1)}) \\ \hat{y} &= W_3^T \varphi_3(\Lambda_2^{-1} \hat{e}^{(2)}) + b \end{aligned} \\ \searrow & (D)_{\text{DeepRKM}}: \quad \begin{aligned} \hat{e}^{(1)} &= \frac{1}{\eta_1} \sum_j h_j^{(1)} K_1(x_j, x) \\ \hat{e}^{(2)} &= \frac{1}{\eta_2} \sum_j h_j^{(2)} K_2(h_j^{(1)}, \Lambda_1^{-1} \hat{e}^{(1)}) \\ \hat{y} &= \frac{1}{\eta_3} \sum_j h_j^{(3)} K_3(h_j^{(2)}, \Lambda_2^{-1} \hat{e}^{(2)}) + b \end{aligned} \end{cases}
$$

The framework can be used for training deep feedforward neural networks and deep kernel machines [Suykens, 2017].

(Other approaches: e.g. kernels for deep learning [Cho & Saul, 2009], mathematics of the neural response [Smale et al., 2010], deep gaussian processes [Damianou & Lawrence, 2013], convolutional kernel networks [Mairal et al., 2014], multi-layer support vector machines [Wiering & Schomaker, 2014])

Generative RKM

RKM objective for training and generating

- RBM energy function

$$E(v, h; \theta) = -v^{\mathrm{T}} W h - c^{\mathrm{T}} v - a^{\mathrm{T}} h$$

with model parameters $\theta = \{W, c, a\}$

- RKM "super-objective" function (for training and for generating)

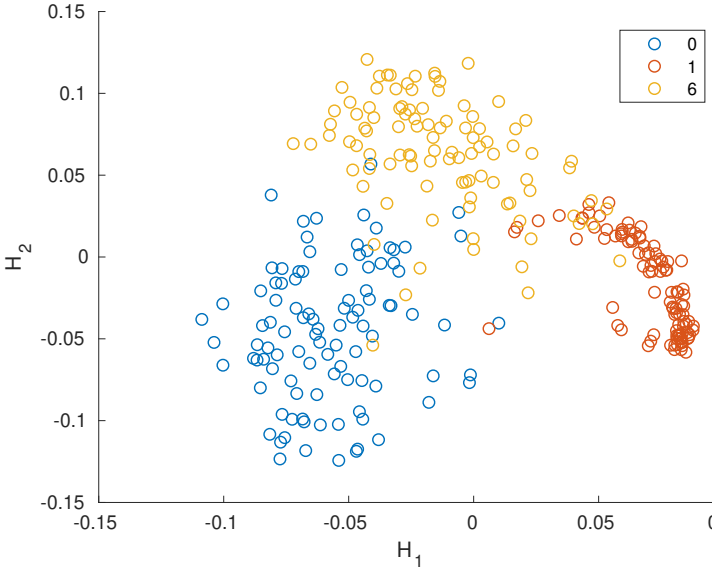
$$\bar{J}(v, h, W) = -v^{\mathrm{T}} W h + \frac{\lambda}{2} h^{\mathrm{T}} h + \frac{1}{2} v^{\mathrm{T}} v + \frac{\eta}{2} \mathrm{Tr}(W^{\mathrm{T}} W)$$

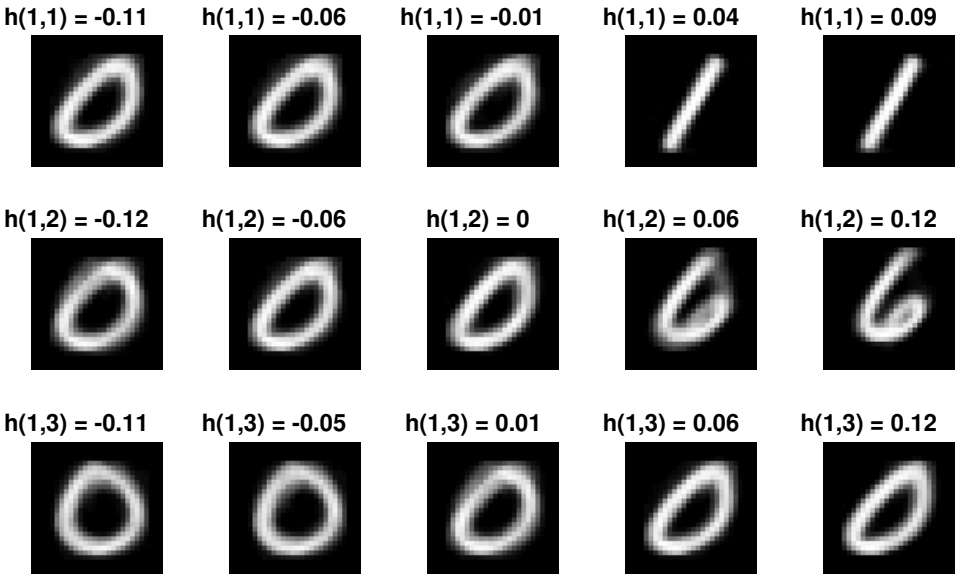
Training:	clamp v	$\rightarrow$	$\bar{J}_{\mathrm{train}}(h, W)$
Generating:	clamp h, W	$\rightarrow$	$\bar{J}_{\mathrm{gen}}(v)$

[Schreurs & Suykens, ESANN 2018]

Explainable AI: latent space exploration

hidden units: exploring the **whole continuum**:

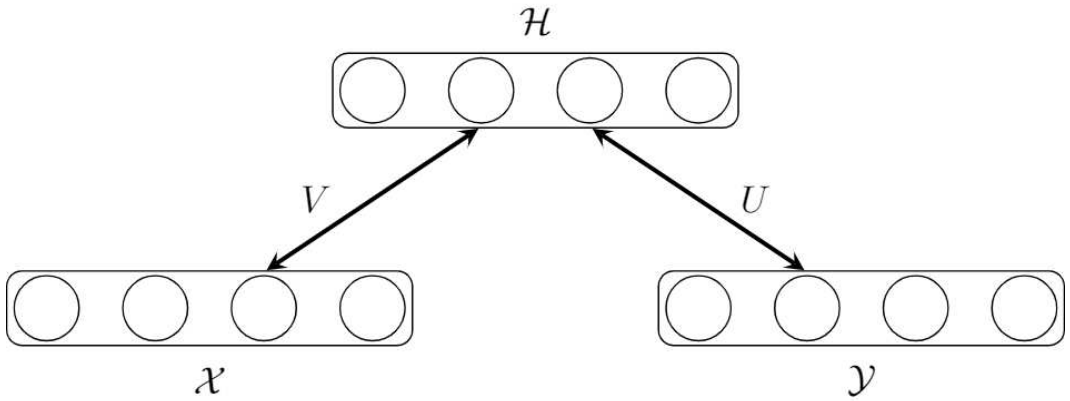




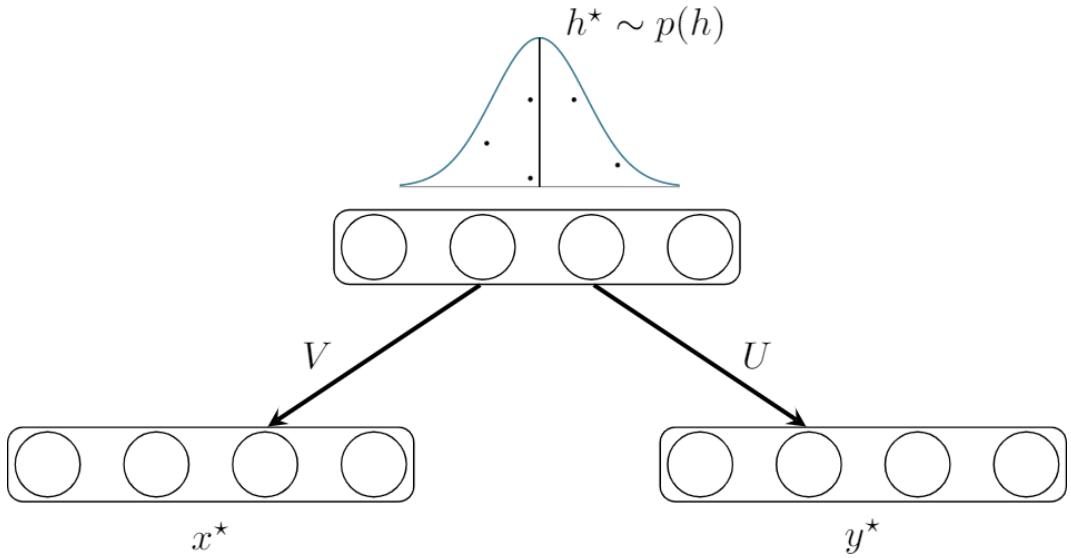
[figures by Joachim Schreurs]

Generative RKM (Gen-RKM) (1)

Train:

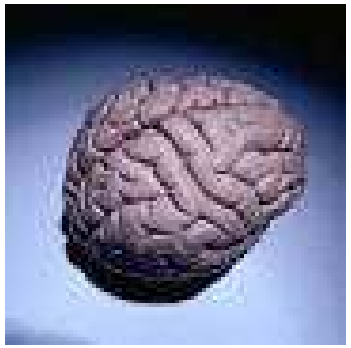


Generate:

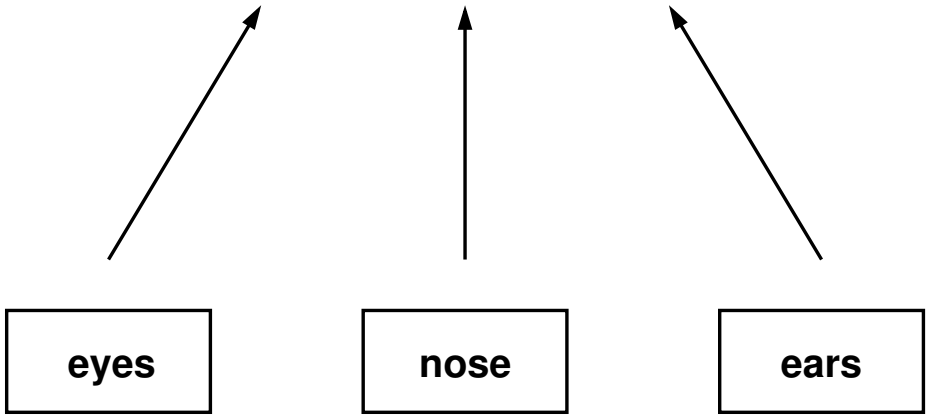


[Pandey, Schreurs & Suykens, 2019, arXiv:1906.08144]

Analogy with the human brain



shared feature representation



multimodal data

Gen-RKM (2)

The objective

$$J_{\text{train}}(\boldsymbol{h}_i, \boldsymbol{U}, \boldsymbol{V}) = \sum_{i=1}^{N} \left(-\phi_1(\boldsymbol{x}_i)^T \boldsymbol{U} \boldsymbol{h}_i - \phi_2(\boldsymbol{y}_i)^T \boldsymbol{V} \boldsymbol{h}_i + \frac{\lambda}{2} \boldsymbol{h}_i^T \boldsymbol{h}_i\right) + \frac{\eta_1}{2}\text{Tr}(\boldsymbol{U}^T \boldsymbol{U}) + \frac{\eta_2}{2}\text{Tr}(\boldsymbol{V}^T \boldsymbol{V})$$

results for **training** into the eigenvalue problem

$$(\frac{1}{\eta_1}\boldsymbol{K}_1 + \frac{1}{\eta_2}\boldsymbol{K}_2)\boldsymbol{H}^T = \boldsymbol{H}^T \boldsymbol{\Lambda}$$

with $\boldsymbol{H} = [\boldsymbol{h}_1 ... \boldsymbol{h}_N]$ and kernel matrices $\boldsymbol{K}_1, \boldsymbol{K}_2$ related to ϕ_1, ϕ_2.

[Pandey, Schreurs & Suykens, 2019, arXiv:1906.08144]

Gen-RKM (3)

Generating data is based on a newly generated $\boldsymbol{h}^\star$ and the objective

$$J_{\text{gen}}(\phi_1(\boldsymbol{x}^\star), \varphi_2(\boldsymbol{y}^\star)) = -\phi_1(\boldsymbol{x}^\star)^T \boldsymbol{V} \boldsymbol{h}^\star - \phi_2(\boldsymbol{y}^\star)^T \boldsymbol{U} \boldsymbol{h}^\star + \frac{1}{2}\phi_1(\boldsymbol{x}^\star)^T \phi_1(\boldsymbol{x}^\star) + \frac{1}{2}\phi_2(\boldsymbol{y}^\star)^T \phi_2(\boldsymbol{y}^\star)$$

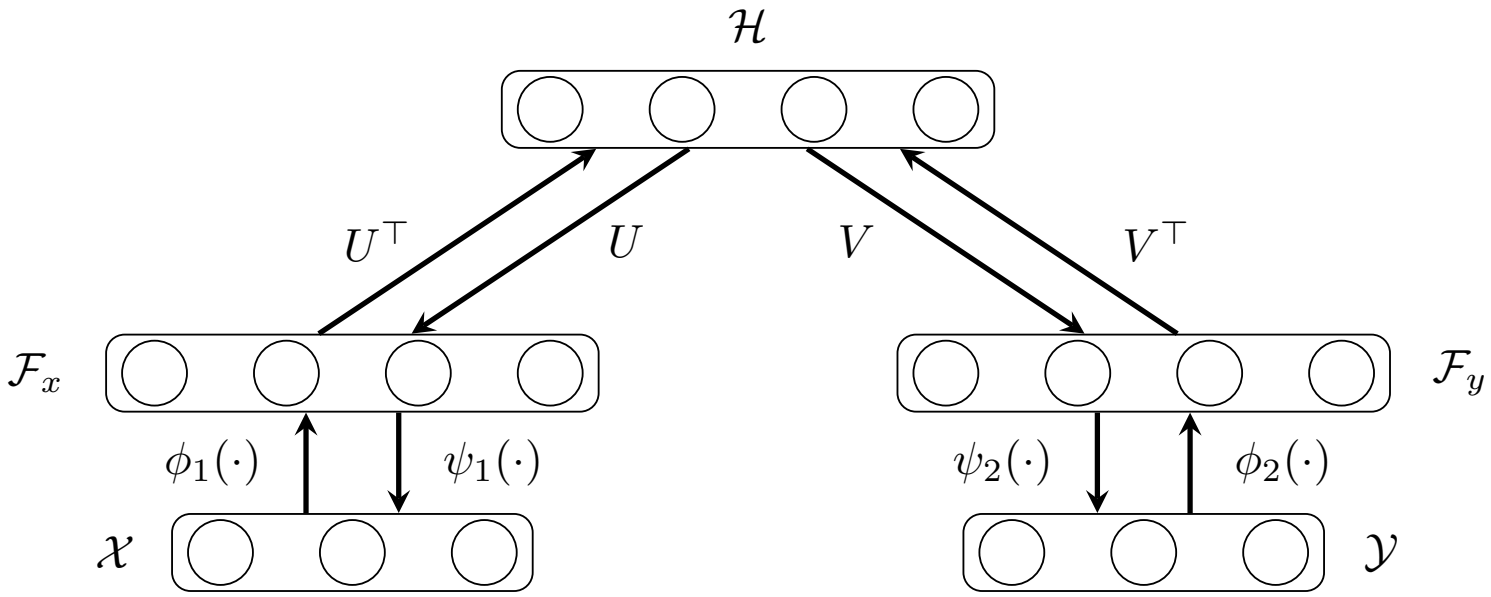
giving

$$\phi_1(\boldsymbol{x}^\star) = \frac{1}{\eta_1}\sum_{i=1}^{N} \phi_1(\boldsymbol{x}_i)\boldsymbol{h}_i^T \boldsymbol{h}^\star, \;\; \phi_2(\boldsymbol{y}^\star) = \frac{1}{\eta_2}\sum_{i=1}^{N} \phi_2(\boldsymbol{y}_i)\boldsymbol{h}_i^T \boldsymbol{h}^\star.$$

For generating $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}$ one can either work with the kernel smoother or work with an explicit feature map using a (deep) neural network or CNN.

[Pandey, Schreurs & Suykens, 2019, arXiv:1906.08144]

Gen-RKM (4)



Gen-RKM schematic representation modeling a common subspace $\mathcal{H}$ between two data sources $\mathcal{X}$ and $\mathcal{Y}$. The ϕ_1, ϕ_2 are the feature maps ($\mathcal{F}_x$ and $\mathcal{F}_y$ represent the feature-spaces) corresponding to the two data sources. While ψ_1, ψ_2 represent the pre-image maps. The interconnection matrices U, V model dependencies between latent variables and the mapped data sources.

[Pandey, Schreurs & Suykens, 2019, arXiv:1906.08144]

Gen-RKM: implicit feature map (choose kernel)

Obtain

$$\boldsymbol{k}_{\boldsymbol{x}^\star} = \frac{1}{\eta_1}\boldsymbol{K}_1\boldsymbol{H}^\top\boldsymbol{h}^\star, \quad \boldsymbol{k}_{\boldsymbol{y}^\star} = \frac{1}{\eta_2}\boldsymbol{K}_2\boldsymbol{H}^\top\boldsymbol{h}^\star,$$

with $\boldsymbol{k}_{\boldsymbol{x}^\star} = [k(\boldsymbol{x}_1, \boldsymbol{x}^\star), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x}^\star)]^\top$.

Using the kernel-smoother:

$$\hat{\boldsymbol{x}} = \psi_1\left(\phi_1(\boldsymbol{x}^\star)\right) = \frac{\sum_{j=1}^{n_r}\tilde{k}_1(\boldsymbol{x}_j, \boldsymbol{x}^\star)\boldsymbol{x}_j}{\sum_{j=1}^{n_r}\tilde{k}_1(\boldsymbol{x}_j, \boldsymbol{x}^\star)}, \quad \hat{\boldsymbol{y}} = \psi_2\left(\phi_2(\boldsymbol{y}^\star)\right) = \frac{\sum_{j=1}^{n_r}\tilde{k}_2(\boldsymbol{y}_j, \boldsymbol{y}^\star)\boldsymbol{y}_j}{\sum_{j=1}^{n_r}\tilde{k}_2(\boldsymbol{y}_j, \boldsymbol{y}^\star)},$$

with $\tilde{k}_1(\boldsymbol{x}_i, \boldsymbol{x}^\star)$ and $\tilde{k}_2(\boldsymbol{y}_i, \boldsymbol{y}^\star)$ the scaled similarities between 0 and 1.
n_r is the number of closest points based on the similarity defined by kernels $\tilde{k}_1$ and $\tilde{k}_2$.

Gen-RKM: choose explicit (CNN) feature map

Parametrized feature maps: $\phi_{\boldsymbol{\theta}}(\cdot)$, $\psi_{\boldsymbol{\zeta}}(\cdot)$ (e.g. CNN and transposed CNN).

Overall objective function, using a stabilization mechanism [Suykens, 2017]:

$$\begin{aligned}\min_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\boldsymbol{\zeta}_1,\boldsymbol{\zeta}_2} \mathcal{J}_c \;=\; & \mathcal{J}_{\text{train}} + \tfrac{c_{\text{stab}}}{2}\mathcal{J}^2_{\text{train}} \\ & + \tfrac{c_{\text{acc}}}{2N}\left(\sum_{i=1}^{N}\left[\mathcal{L}_1(\boldsymbol{x}_i^\star, \psi_{1_{\boldsymbol{\zeta}_1}}(\phi_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i^\star))) + \mathcal{L}_2(\boldsymbol{y}_i^\star, \psi_{2_{\boldsymbol{\zeta}_2}}(\phi_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i^\star)))\right]\right)\end{aligned}$$

with reconstruction errors

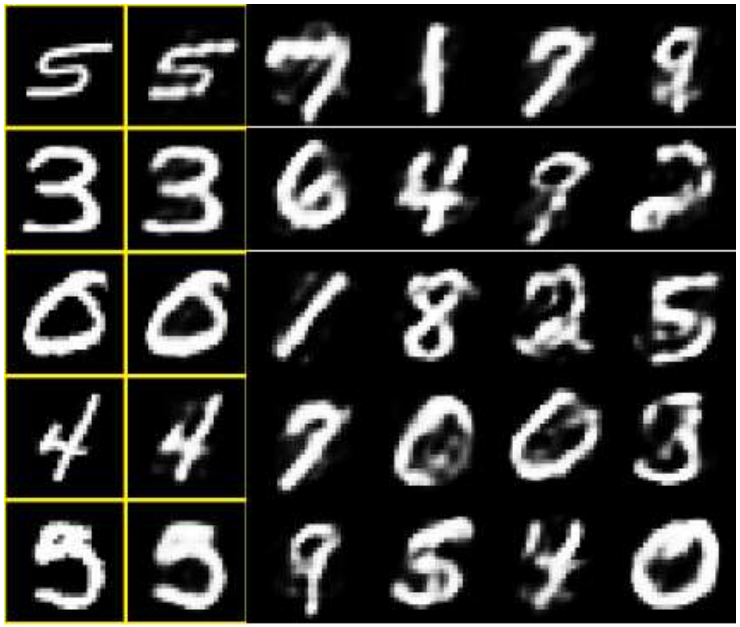
$$\begin{aligned}\mathcal{L}_1(\boldsymbol{x}_i^\star, \psi_{1_{\boldsymbol{\zeta}_1}}(\phi_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i^\star))) &= \tfrac{1}{N}\|\boldsymbol{x}_i^\star - \psi_{1_{\boldsymbol{\zeta}_1}}(\phi_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i^\star))\|_2^2, \\ \mathcal{L}_2(\boldsymbol{y}_i^\star, \psi_{2_{\boldsymbol{\zeta}_2}}(\phi_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i^\star))) &= \tfrac{1}{N}\|\boldsymbol{y}_i^\star - \psi_{2_{\boldsymbol{\zeta}_2}}(\phi_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i^\star))\|_2^2\end{aligned}$$

and with $\Phi_{\boldsymbol{x}} = [\phi_1(\boldsymbol{x}_1), \ldots, \phi_1(\boldsymbol{x}_N)], \Phi_{\boldsymbol{y}} = [\phi_2(\boldsymbol{y}_1), \ldots, \phi_2(\boldsymbol{y}_N)]$, U, V from

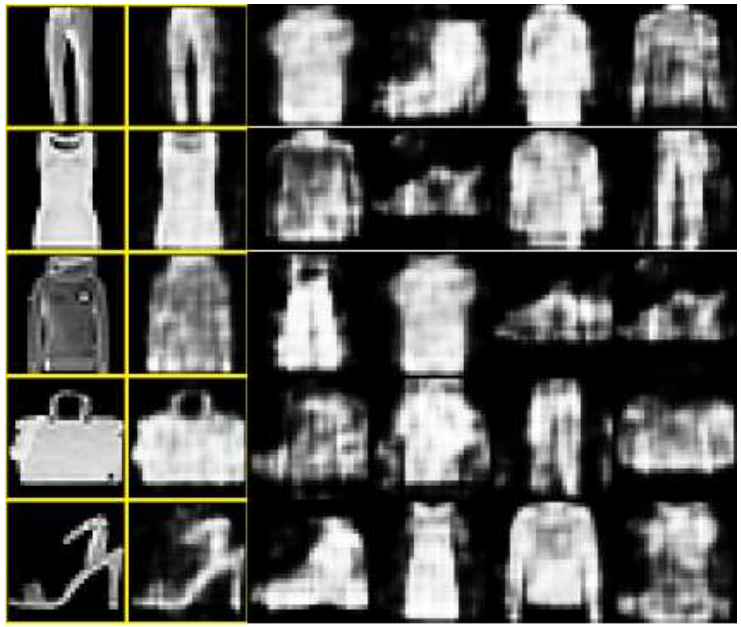
$$\begin{bmatrix} \frac{1}{\eta_1}\Phi_{\boldsymbol{x}}\Phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_1}\Phi_{\boldsymbol{x}}\Phi_{\boldsymbol{y}}^\top \\ \frac{1}{\eta_2}\Phi_{\boldsymbol{y}}\Phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_2}\Phi_{\boldsymbol{y}}\Phi_{\boldsymbol{y}}^\top \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix} \Lambda.$$

Hence, joint feature learning and subspace learning.

Gen-RKM: examples (1)



MNIST

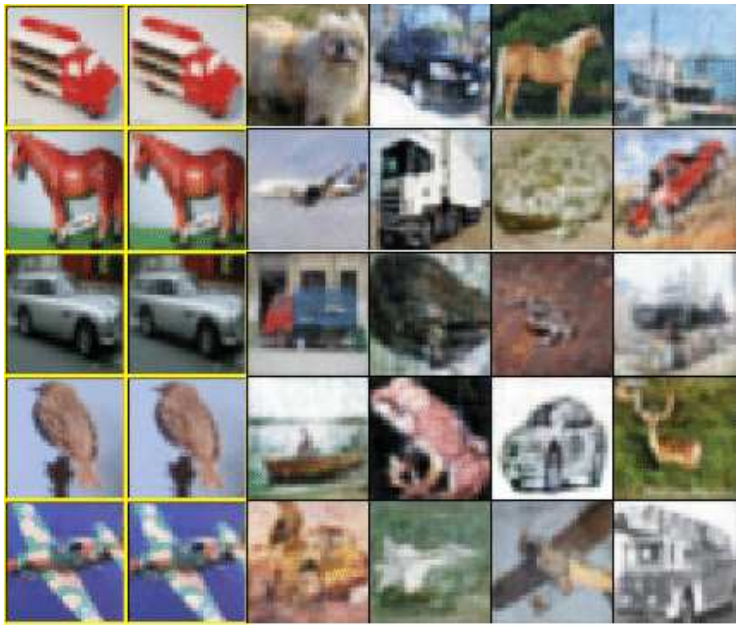


Fashion-MNIST

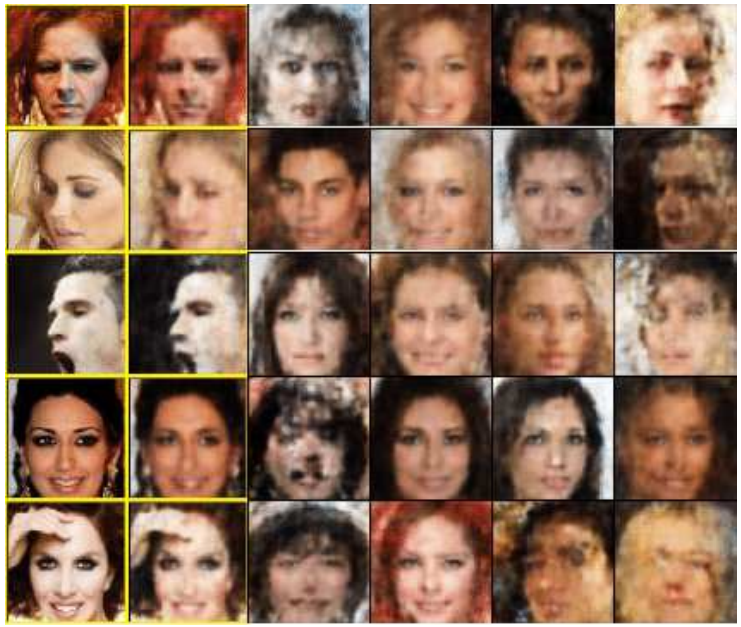
Generated samples from the model using CNN as explicit feature map in the kernel function. The yellow boxes show training examples and the adjacent boxes show the reconstructed samples. The other images (columns 3-6) are generated by random sampling from the fitted distribution over the learned latent variables.

[Pandey, Schreurs & Suykens, Neural Networks 2021]

Gen-RKM: examples (2)



CIFAR-10



CelebA

Generated samples from the model using CNN as explicit feature map in the kernel function. The yellow boxes show training examples and the adjacent boxes show the reconstructed samples. The other images (columns 3-6) are generated by random sampling from the fitted distribution over the learned latent variables.

[Pandey, Schreurs & Suykens, Neural Networks 2021]

Gen-RKM: multi-view generation (1)

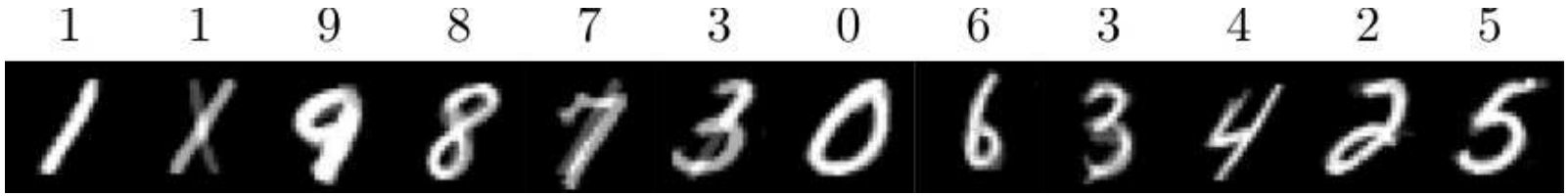


CelebA

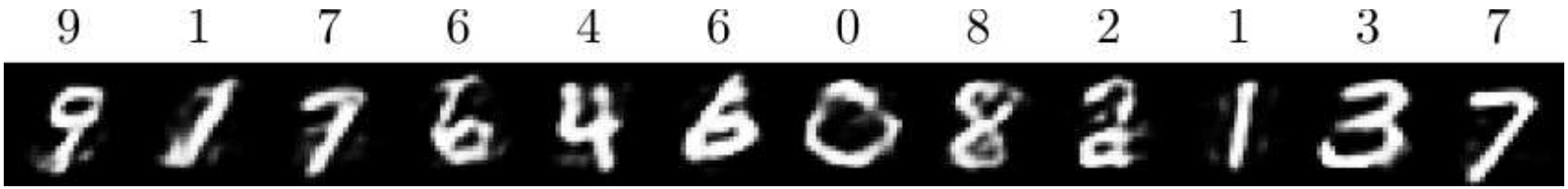
Multi-view generation on CelebA dataset showing images and attributes

[Pandey, Schreurs & Suykens, Neural Networks 2021]

Gen-RKM: multi-view generation (2)



MNIST: Implicit feature maps with Gaussian kernel + generation by kernel-smoother



MNIST: Explicit feature maps using Convolutional Neural Networks

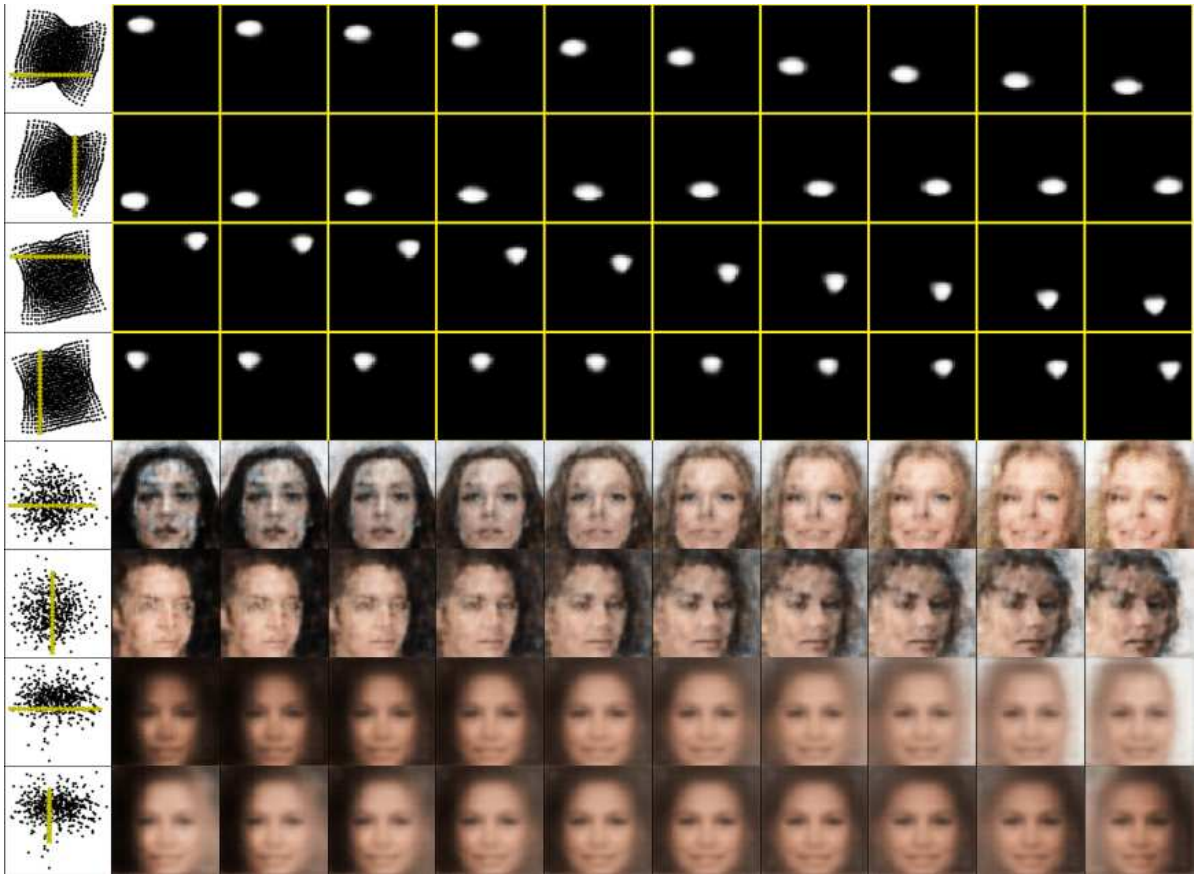


CIFAR-10: Explicit feature maps using CNNs + Transposed CNNs

Multi-view Generation (images and labels) using implicit and explicit feature maps

[Pandey, Schreurs & Suykens, Neural Networks 2021]

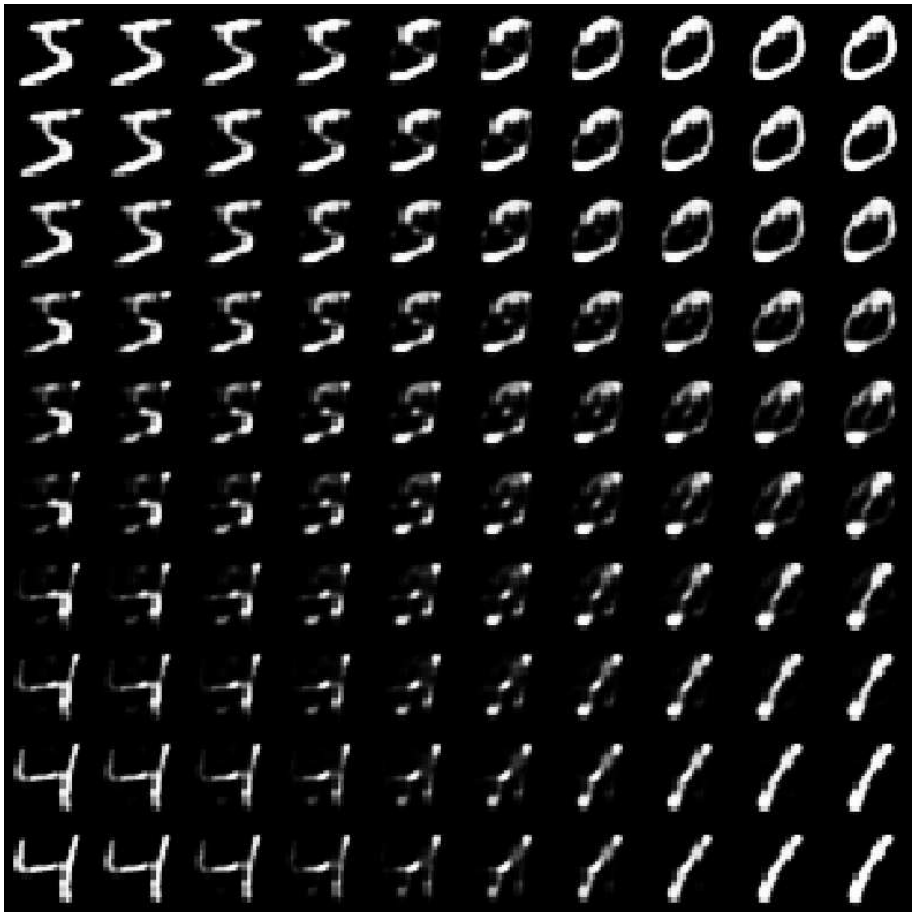
Gen-RKM: latent space exploration (1)



Exploring the learned **uncorrelated-features** by traversing along the eigenvectors

Explainability: changing one single neuron's hidden feature changes the hair color while preserving face structure! [Pandey, Schreurs & Suykens, Neural Networks 2021]

Gen-RKM: latent space exploration (2)



MNIST reconstructed images by bilinear-interpolation in latent space
[Pandey, Schreurs & Suykens, Neural Networks 2021]

Gen-RKM: latent space exploration (3)



CelebA reconstructed images by bilinear-interpolation in latent space

[Pandey, Schreurs & Suykens, Neural Networks 2021]

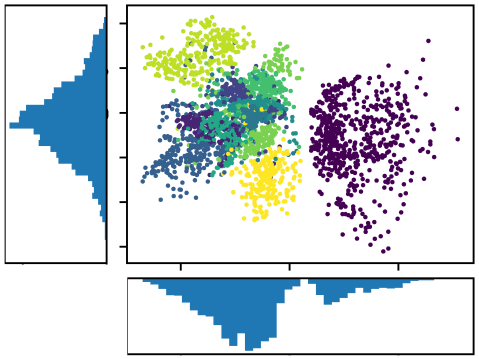
Gen-RKM - traversals along principal components



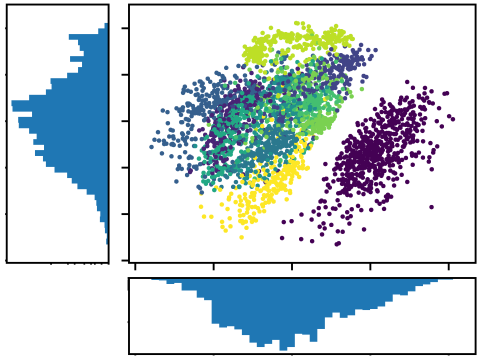
Disentangled Representation Learning and Generation with Manifold Optimization

[Pandey, Fanuel, Schreurs & Suykens, 2019, arXiv:2006.07046]

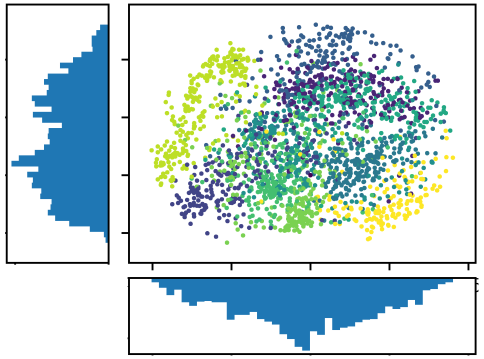
Robust Gen-RKM



VAE



Gen-RKM



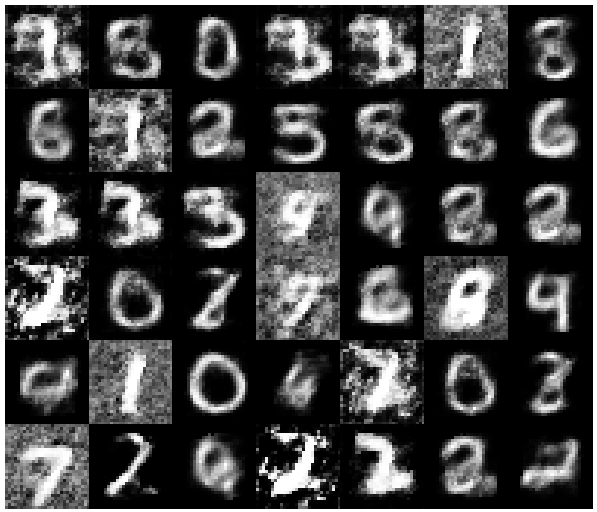
Robust Gen-RKM

- VAE and Gen-RKM: presence of outliers distorts the distribution of the latent variables
- Robust Gen-RKM: down-weighting of the outliers makes close to Gaussian distribution

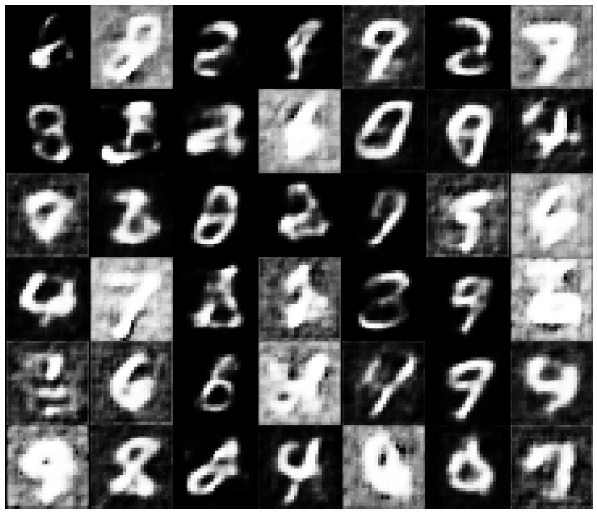
Weighted conjugate feature duality: $\frac{1}{2\lambda}e^T De + \frac{\lambda}{2}h^T D^{-1}h \geq e^T h$

[Pandey, Schreurs & Suykens, 2019, arXiv:2002.01180, LOD 2020]

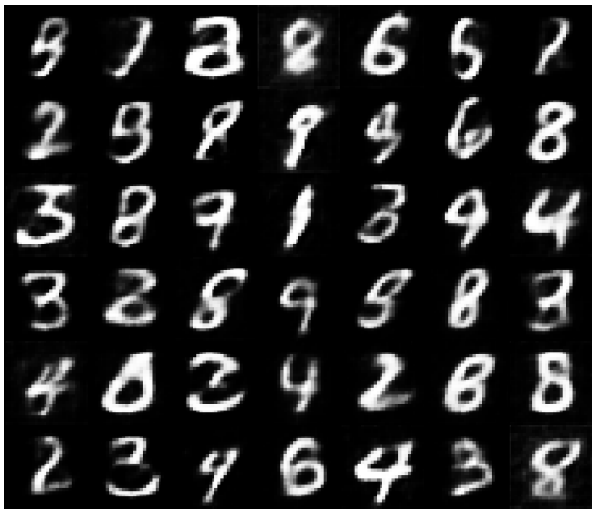
Robust Gen-RKM - Robust generation



VAE



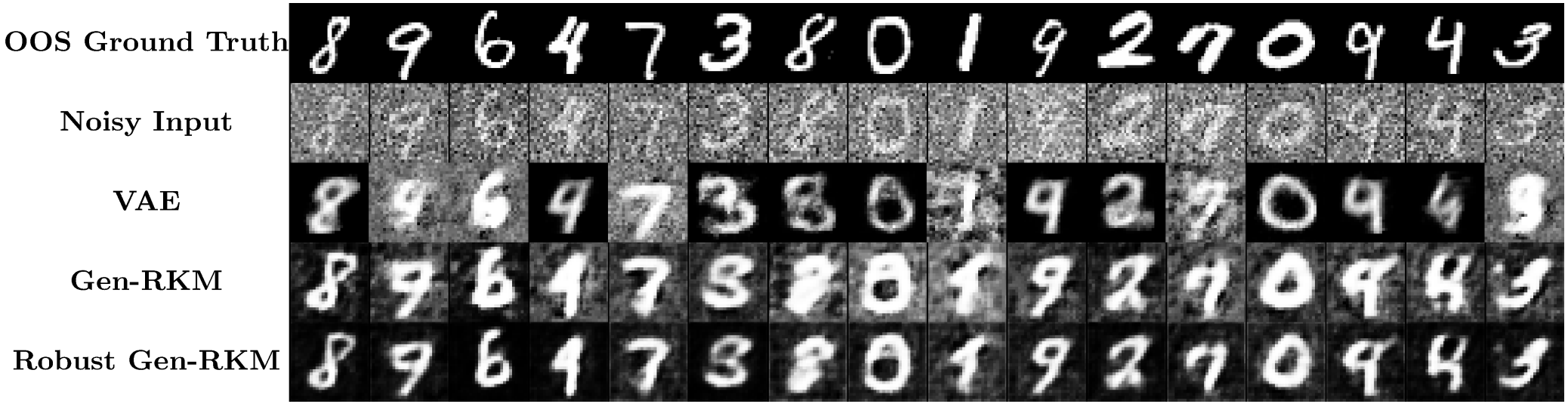
Gen-RKM



Robust Gen-RKM

[Pandey, Schreurs & Suykens, 2019, arXiv:2002.01180, LOD 2020]

Robust Gen-RKM - Robust denoising



[Pandey, Schreurs & Suykens, 2019, arXiv:2002.01180, LOD 2020]

RKM references and software

- Suykens J.A.K., "Deep Restricted Kernel Machines using Conjugate Feature Duality", *Neural Computation*, vol. 29, no. 8, pp. 2123-2163, Aug. 2017. https://www.mitpressjournals.org/doi/pdf/10.1162/neco_a_00984
- Houthuys L., Suykens J.A.K., "Tensor Learning in Multi-View Kernel PCA",*International Conference on Artificial Neural Networks 2018 (ICANN 2018)*, Rhodes, Greece, Oct. 2018, pp. 205-215.
- Schreurs J., Suykens J.A.K., "Generative Kernel PCA", *European Symposium on Artificial Neural Networks (ESANN 2018)*, Bruges, Belgium, Apr. 2018, pp. 129-134.
- Pandey A., Schreurs J., Suykens J.A.K., "Generative Restricted Kernel Machines: A framework for multi-view generation and disentangled feature learning", *Neural Networks*, Vol. 135, 177-191, 2021
- Pandey A., Fanuel M., Schreurs J., Suykens J.A.K., Disentangled Representation Learning and Generation with Manifold Optimization, arXiv:2006.07046
- Pandey A., Schreurs J., Suykens J.A.K., Robust Generative Restricted Kernel Machines using Weighted Conjugate Feature Duality, arXiv:2002.01180, *LOD 2020*.

Software: see https://www.esat.kuleuven.be/stadius/E/software.php

Conclusions

- function estimation: **parametric versus kernel-based**, **primal and dual** model representations
- **neural network interpretations** in primal and dual
- (re)weighted LS-SVMs for **robustness**
- RKM: **new connections** between RBM, kernel PCA and LS-SVM
- **deep kernel machines**, **generative models**
- **explainability:** latent space exploration, understanding the role of each individual neuron
- **robustness** through weighted conjugate feature duality

Acknowledgements (1)

- Current and former co-workers at ESAT-STADIUS:

  C. Alzate, Y. Chen, J. De Brabanter, K. De Brabanter, B. De Cooman, L. De Lathauwer, H. De Meulemeester, B. De Moor, H. De Plaen, Ph. Dreesen, M. Espinoza, T. Falck, M. Fanuel, Y. Feng, B. Gauthier, X. Huang, L. Houthuys, V. Jumutc, Z. Karevan, R. Langone, F. Liu, R. Mall, S. Mehrkanoon, G. Nisol, M. Orchel, A. Pandey, P. Patrinos, K. Pelckmans, S. RoyChowdhury, S. Salzo, J. Schreurs, M. Signoretto, Q. Tao, F. Tonin, J. Vandewalle, T. Van Gestel, S. Van Huffel, C. Varon, Y. Yang, and others

- Many other people for joint work, discussions, invitations, organizations

- Support from ERC AdG E-DUALITY, ERC AdG A-DATADRIVE-B, KU Leuven, OPTEC, IUAP DYSCO, FWO projects, IWT, Flanders AI, Leuven.AI

Acknowledgements (2)



KATHOLIEKE UNIVERSITEIT
LEUVEN

FWO
VLAANDEREN

Acknowledgements (3)



ERC Advanced Grant E-DUALITY

Exploring duality for future data-driven modelling

Thank you