

# **Deep Restricted Kernel Machines**

**Johan Suykens**

KU Leuven, ESAT-STADIUS  
Kasteelpark Arenberg 10  
B-3001 Leuven (Heverlee), Belgium  
Email: [johan.suykens@esat.kuleuven.be](mailto:johan.suykens@esat.kuleuven.be)  
<http://www.esat.kuleuven.be/stadius/>

**UGent Data Science Seminar, Feb 22 2018**

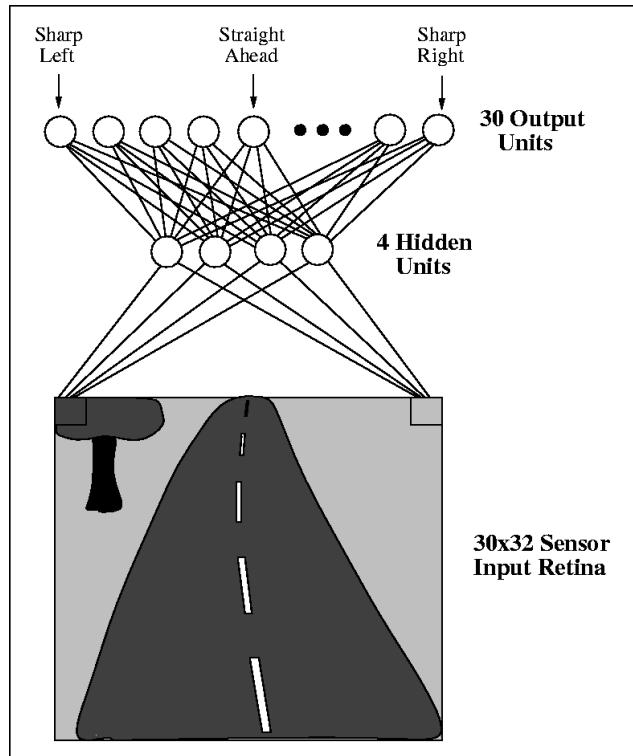
# Outline

- Introduction and motivation
- Restricted kernel machines
- Deep restricted kernel machines
- Examples

# *Introduction*

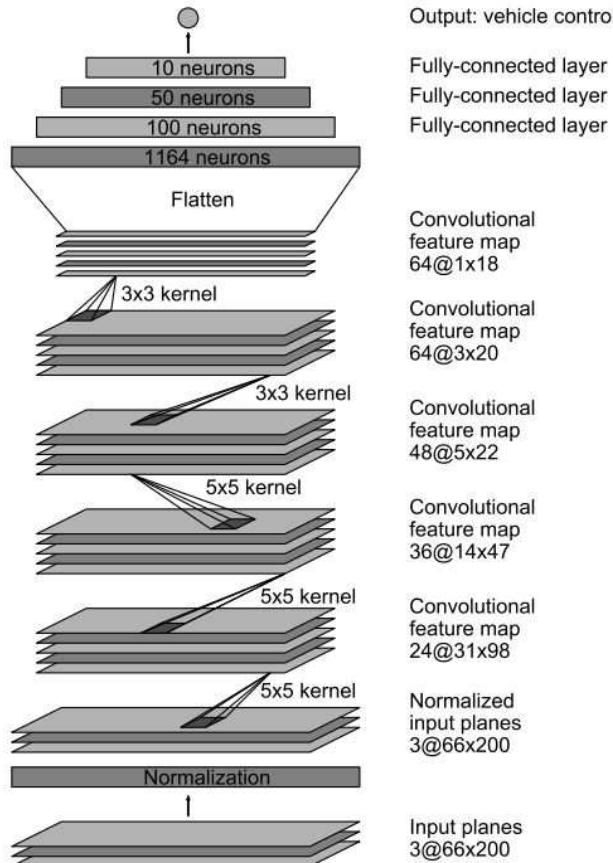
# Self-driving cars and neural networks

in the early days of neural networks:



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

# Self-driving cars and deep learning



Waymo / Google Self-Driving Car



Tesla Autopilot



Uber



nuTonomy

(27 million connections)

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

## Historical context

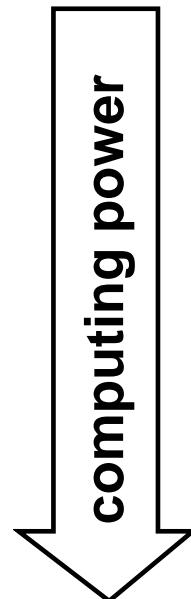
- 1942 McCulloch & Pitts: mathematical model for neuron
- 1958 Rosenblatt: perceptron learning
- 1960 Widrow & Hoff: adaline and lms learning rule
- 1969 Minsky & Papert: limitations of perceptron
  
- 1986 Rumelhart et al.: error backpropagating neural networks  
→ *booming of neural network universal approximators*

## Historical context

- 1942 McCulloch & Pitts: mathematical model for neuron
- 1958 Rosenblatt: perceptron learning
- 1960 Widrow & Hoff: adaline and lms learning rule
- 1969 Minsky & Papert: limitations of perceptron
  
- 1986 Rumelhart et al.: error backpropagating neural networks  
→ *booming of neural network universal approximators*
  
- 1992 Vapnik et al.: support vector machine classifiers  
→ *convex optimization, kernel machines*

## Historical context

- 1942 McCulloch & Pitts: mathematical model for neuron
- 1958 Rosenblatt: perceptron learning
- 1960 Widrow & Hoff: adaline and lms learning rule
- 1969 Minsky & Papert: limitations of perceptron
- 1986 Rumelhart et al.: error backpropagating neural networks  
→ *booming of neural network universal approximators*
- 1992 Vapnik et al.: support vector machine classifiers  
→ *convex optimization, kernel machines*
- 1998 LeCun et al.: convolutional neural networks
- 2006 Hinton et al.: deep belief networks
- 2010 Bengio et al.: stacked autoencoders  
→ *booming of deep neural networks*



# Convolutional neural networks

PROC. OF THE IEEE, NOVEMBER 1998

7

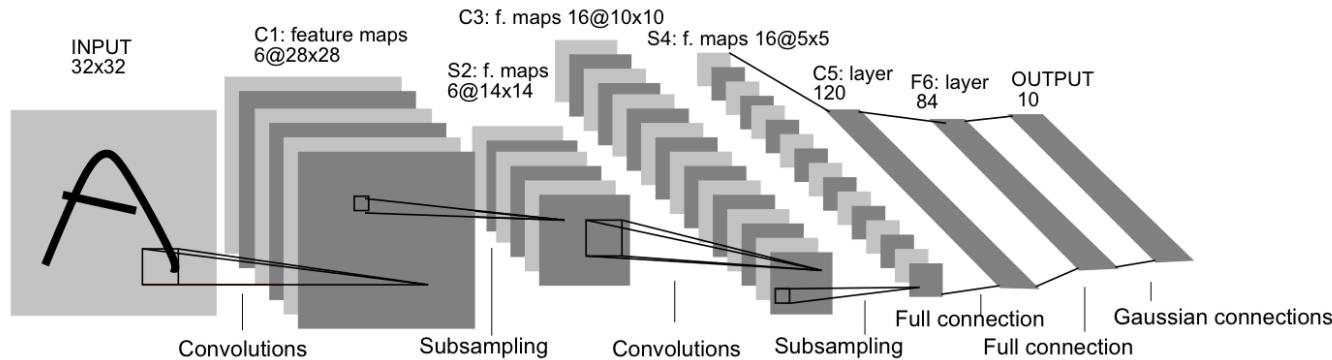


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

[LeCun et al., Proc. IEEE 1998]

Further advanced architectures:

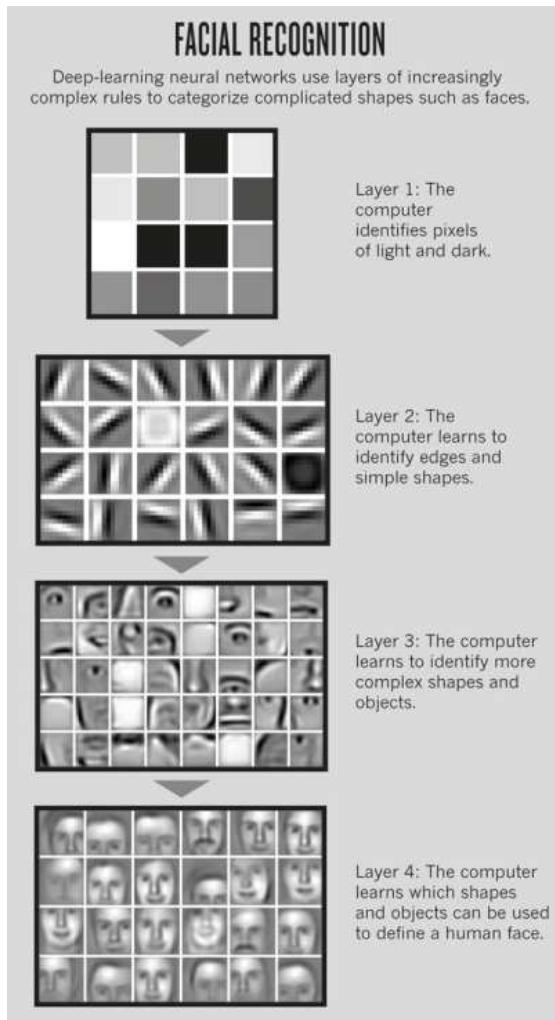
Alexnet (2012): 5 convolutional layers, 3 fully connected

VGGnet (2014): 19 layers

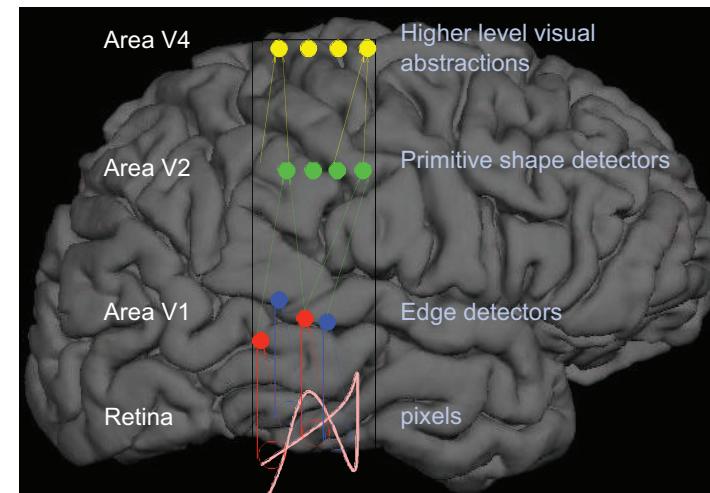
GoogLeNet (2014): 22 layers

ResNet (2015): 152 layers

# Hierarchical features

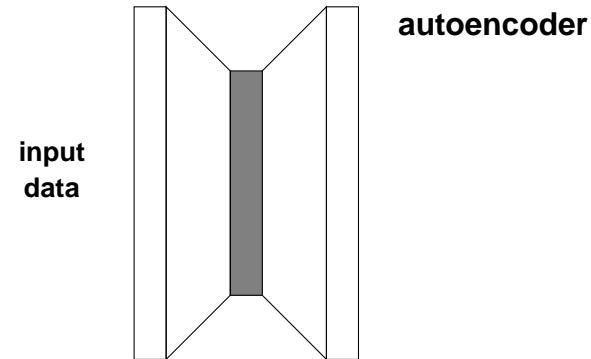


## Deep Architecture in the Brain

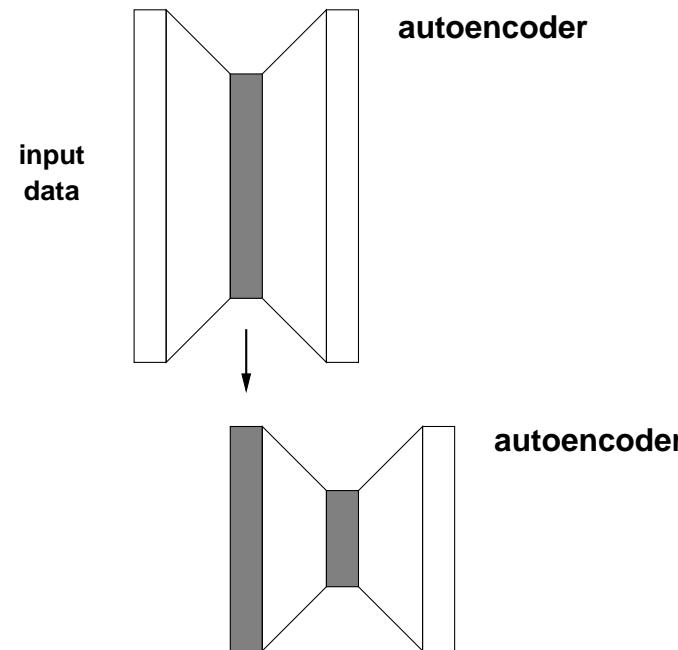


[Jones, The learning machines, Nature 2014; Bengio & LeCun, ICML2009]

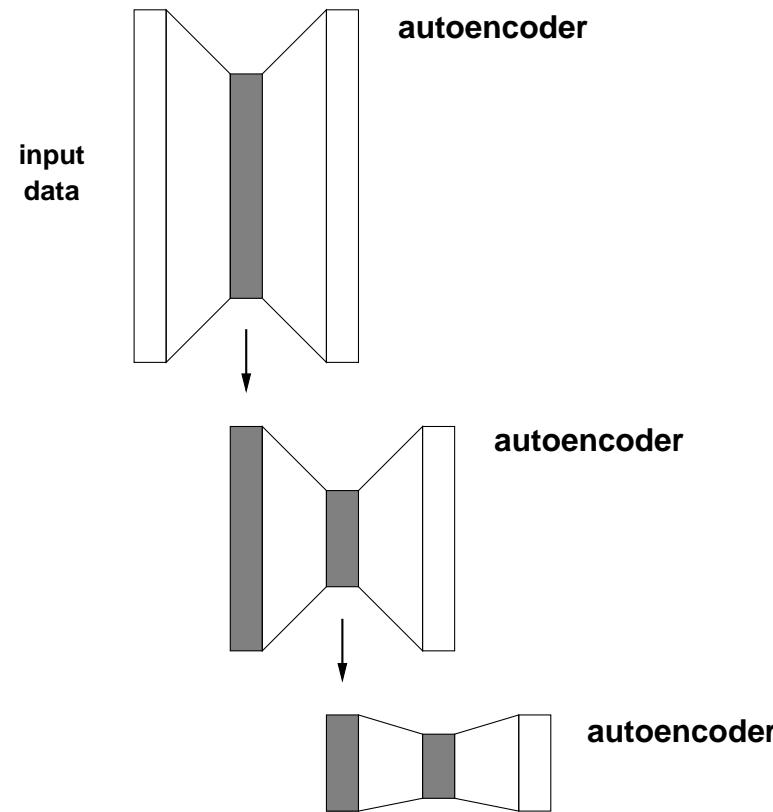
## Stacked autoencoders



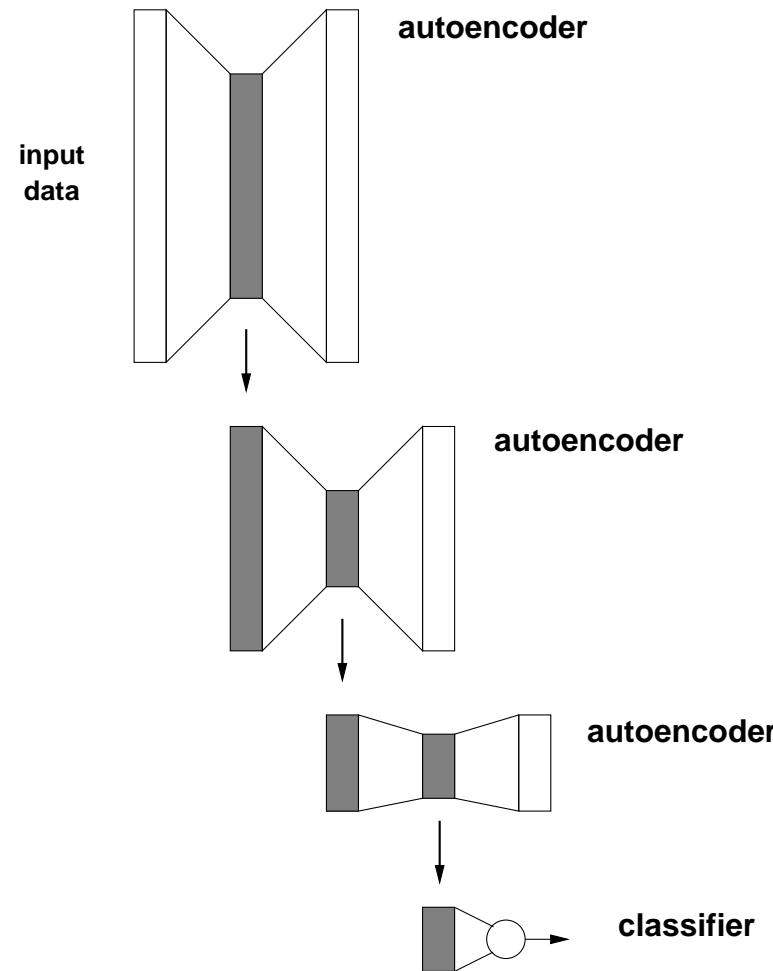
## Stacked autoencoders



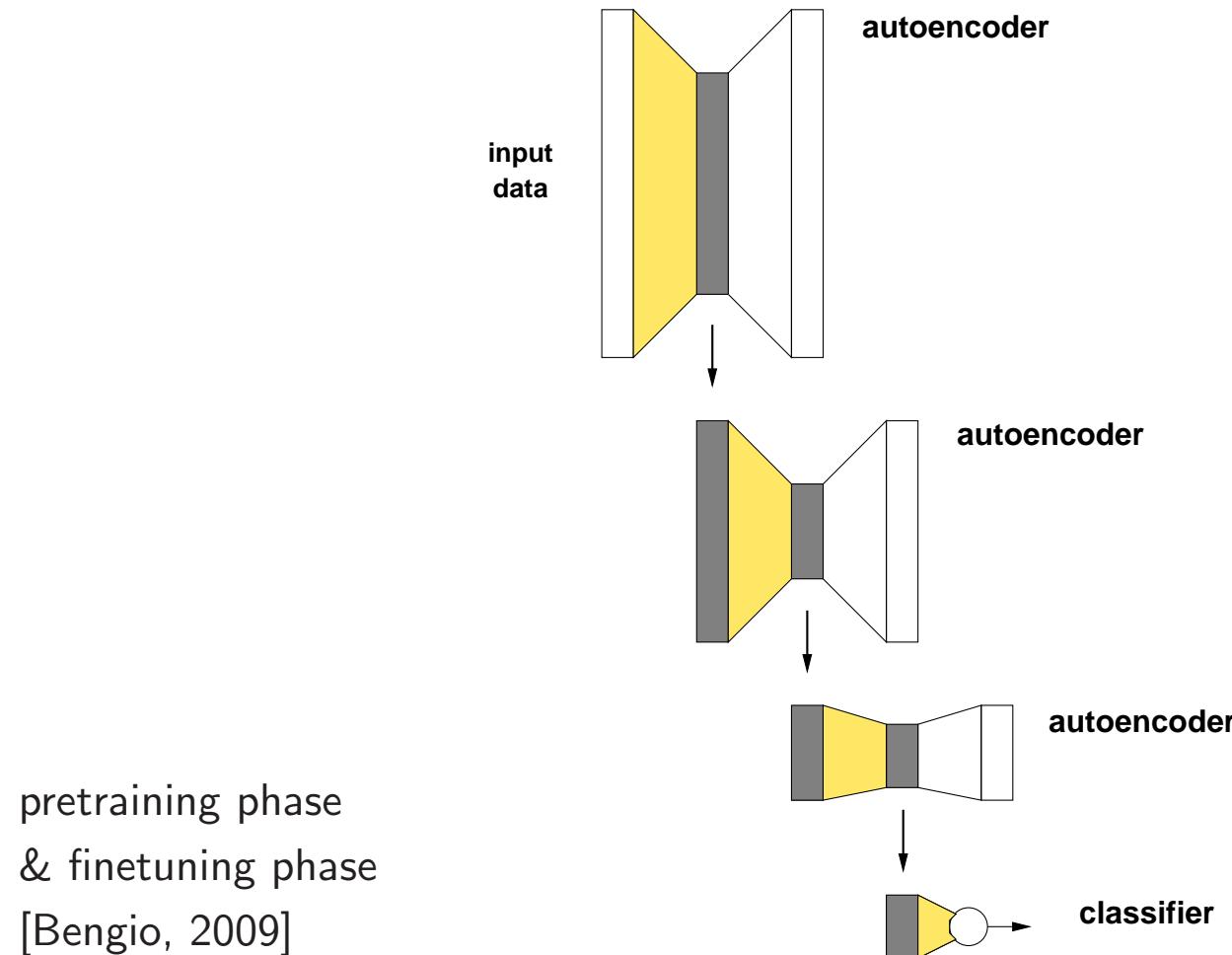
## Stacked autoencoders



# Stacked autoencoders



# Stacked autoencoders



## Kernel machines & deep learning

previous approaches:

- 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]
- other

## Different paradigms

Deep  
Learning

Neural  
Networks

SVM, LS-SVM &  
Kernel methods

## Different paradigms

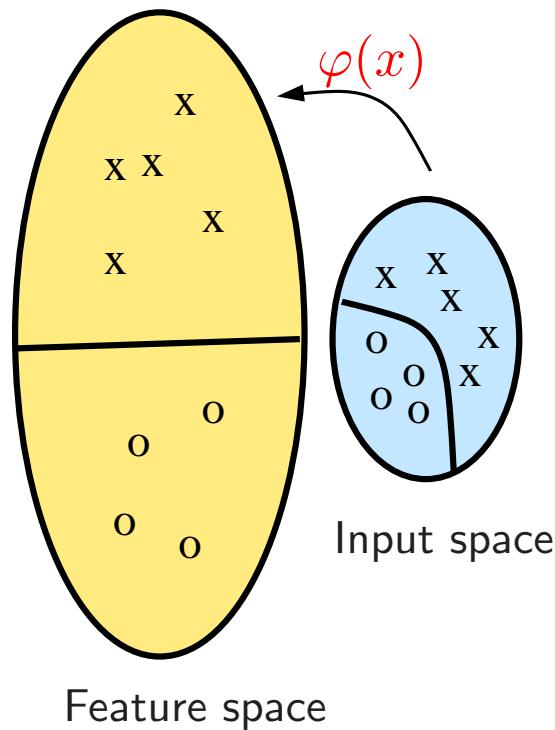
Deep  
Learning

Neural  
Networks

new synergies?

SVM, LS-SVM &  
Kernel methods

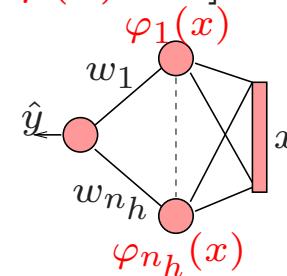
# Support vector machines and neural networks



## Primal space

Parametric

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

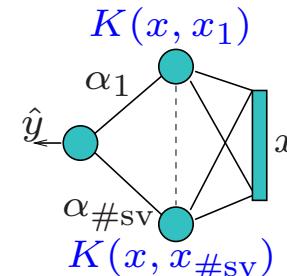


$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$

## Dual space

Nonparametric

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



[Suykens et al., 2002]

## 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

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

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

## Kernel machines & deep learning: New Challenges

- *new synergies and new foundations* between support vector machines & kernel methods and deep learning architectures?
- possible to extend primal and dual model representations (as occurring in SVM and LS-SVM models) *from shallow to deep architectures?*
- possible to handle *deep feedforward neural networks* and *deep kernel machines* within a common setting?

## Kernel machines & deep learning: New Challenges

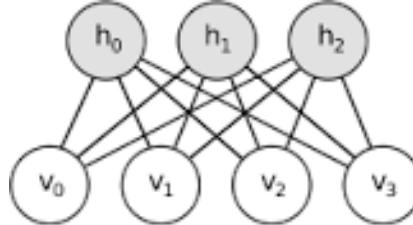
- *new synergies and new foundations* between support vector machines & kernel methods and deep learning architectures?
- possible to extend primal and dual model representations (as occurring in SVM and LS-SVM models) *from shallow to deep architectures?*
- possible to handle *deep feedforward neural networks* and *deep kernel machines* within a common setting?

→ new framework:

"Deep Restricted Kernel Machines" [Suykens, Neural Computation, 2017]

## *Restricted Kernel Machines*

## 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:

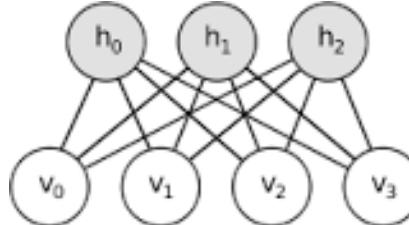
$$E(v, h; \theta) = -v^T Wh - c^T v - a^T h \quad \text{with } \theta = \{W, c, a\}$$

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:

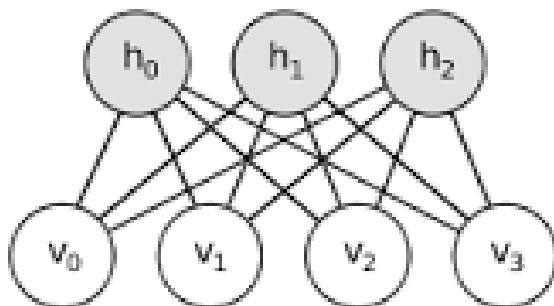
$$E(v, h; \theta) = -v^T Wh - c^T v - a^T h \quad \text{with } \theta = \{W, c, a\}$$

Joint distribution:

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

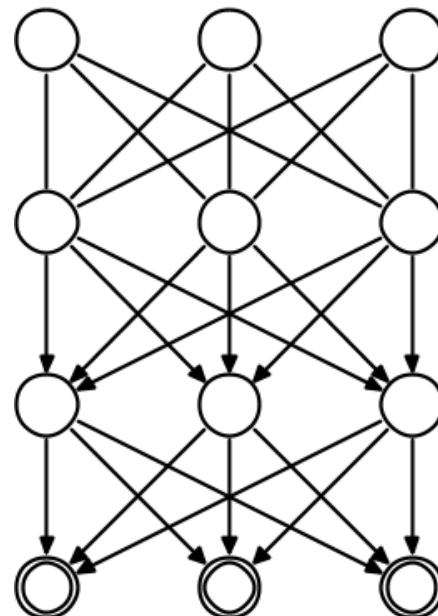
with partition function  $Z(\theta) = \sum_v \sum_h \exp(-E(v, h; \theta))$   
[Hinton, Osindero, Teh, Neural Computation 2006]

## RBM and deep learning

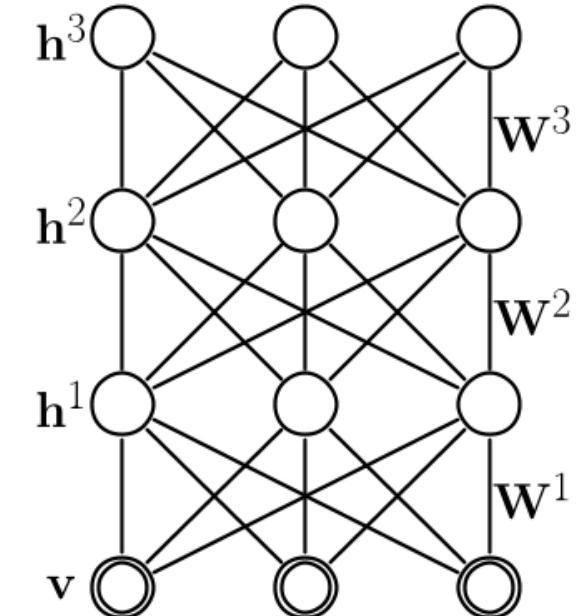


$$p(v, h)$$

Deep Belief Network



Deep Boltzmann Machine



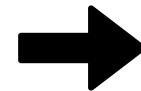
$$p(v, h^1, h^2, h^3, \dots)$$

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

in other words ...

*"deep sandwich"*

*"sandwich"*



## Restricted Kernel Machines (RKM)

Main characteristics:

- 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)

## Model: living in two worlds ...

Original model:

$$\hat{y} = W^T \varphi(x) + b, e = y - \hat{y}$$

objective  $J$   
= regularization term  $\text{Tr}(W^T W)$   
+  $(\frac{1}{\lambda})$  error term  $\sum_i e_i^T e_i$

## Model: living in two worlds ...

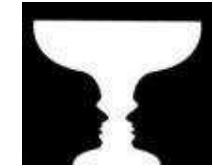
Original model:

$$\hat{y} = W^T \varphi(x) + b, e = y - \hat{y}$$

objective  $J$   
= regularization term  $\text{Tr}(W^T W)$   
+  $(\frac{1}{\lambda})$  error term  $\sum_i e_i^T e_i$

$$\downarrow \quad \frac{1}{2\lambda} e^T e \geq e^T h - \frac{\lambda}{2} h^T h$$

## Model: living in two worlds ...



Original model:

$$\hat{y} = W^T \varphi(x) + b, e = y - \hat{y}$$

objective  $J$   
 = regularization term  $\text{Tr}(W^T W)$   
 +  $(\frac{1}{\lambda})$  error term  $\sum_i e_i^T e_i$

$$\downarrow \quad \frac{1}{2\lambda} e^T e \geq e^T h - \frac{\lambda}{2} h^T h$$

New representation:

$$\hat{y} = \sum_j h_j K(x_j, x) + b$$

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

where  $\underline{J} = \sum_{i=1}^N (y_i^T - x_i^T W - b^T) h_i - \frac{\lambda}{2} \sum_{i=1}^N h_i^T h_i + \frac{\eta}{2} \text{Tr}(W^T W)$

## RKM regression problem

- Stationary points of  $\underline{J}(h_i, W, b)$  (nonlinear case, feature map  $\varphi(\cdot)$ )

$$\left\{ \begin{array}{l} \frac{\partial \underline{J}}{\partial h_i} = 0 \Rightarrow y_i = W^T \varphi(x_i) + b + \lambda h_i, \quad \forall i \\ \frac{\partial \underline{J}}{\partial W} = 0 \Rightarrow W = \frac{1}{\eta} \sum_i \varphi(x_i) h_i^T \\ \frac{\partial \underline{J}}{\partial b} = 0 \Rightarrow \sum_i h_i = 0. \end{array} \right.$$

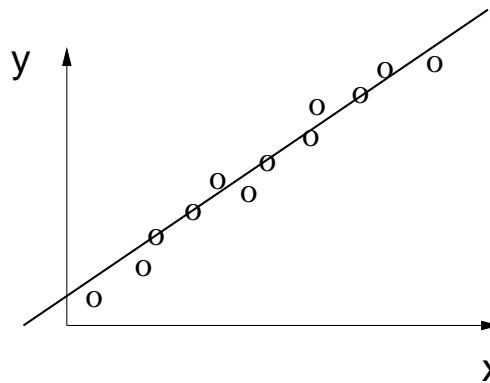
- Solution in  $h_i$  and  $b$  with positive definite kernel  $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$

$$\left[ \begin{array}{c|c} \frac{1}{\eta} K + \lambda I_N & 1_N \\ \hline 1_N^T & 0 \end{array} \right] \left[ \begin{array}{c} H^T \\ b^T \end{array} \right] = \left[ \begin{array}{c} Y^T \\ 0 \end{array} \right]$$

with  $K = [K(x_i, x_j)]$ ,  $H = [h_1 \dots h_N]$ ,  $Y = [y_1 \dots y_N]$ .

## Simple example: line fitting

Given data:  $\{(x_i, y_i)\}_{i=1}^N, x_i, y_i \in \mathbb{R}$



Linear model:

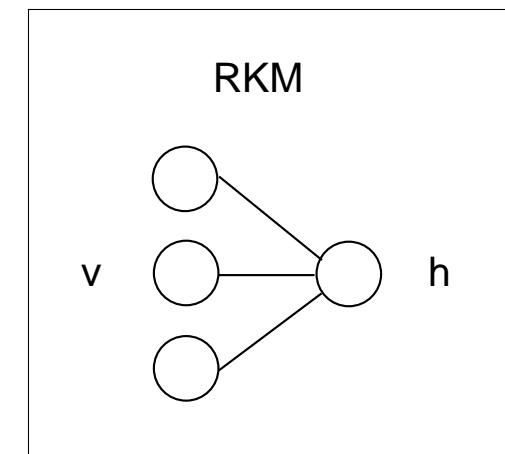
$$\hat{y} = wx + b, e = y - \hat{y}$$

RKM representation:

$$\hat{y} = \sum_i h_i x_i x + b$$

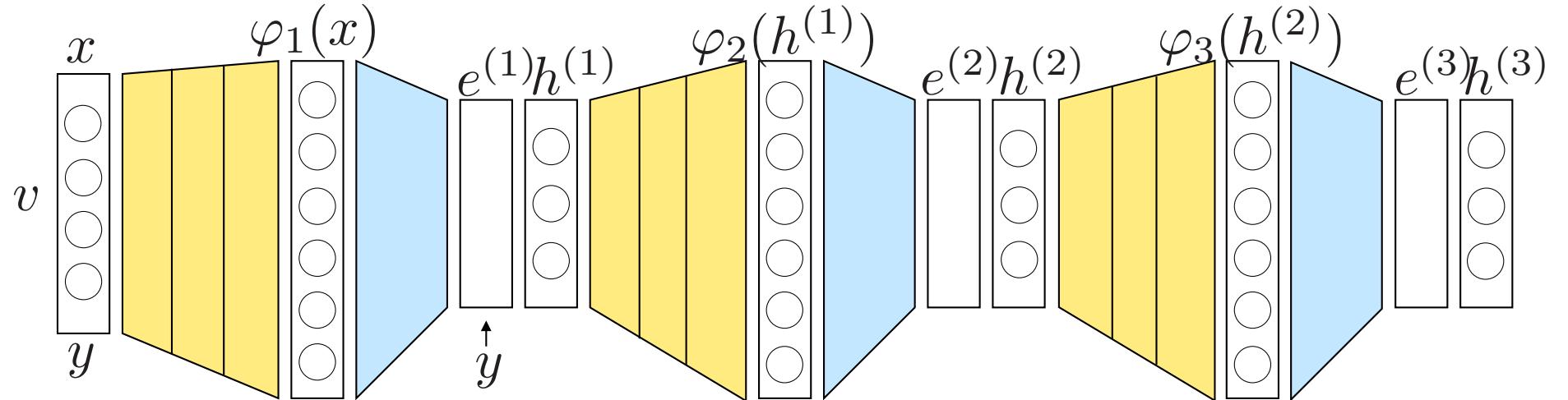
3 visible units:  $v = [x; 1; -y]$

1 hidden unit:  $h \in \mathbb{R}$



## *Deep Restricted Kernel Machines*

## Deep RKM: Example



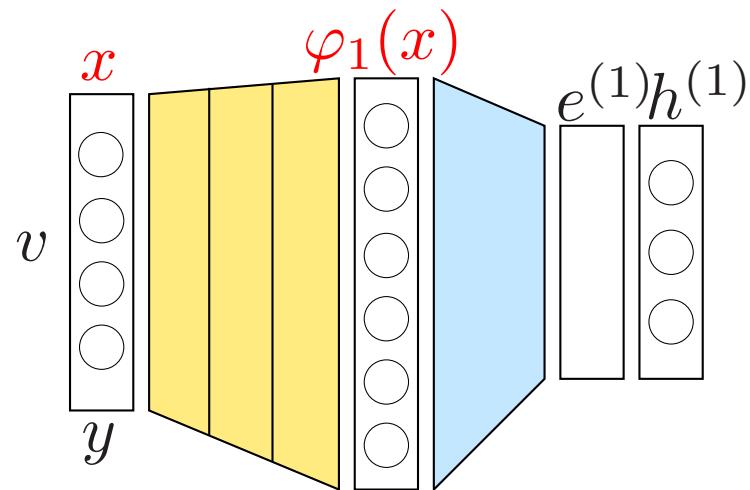
Deep RKM: LSSVM + KPCA + KPCA

Coupling of RKMs by taking sum of the objectives

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

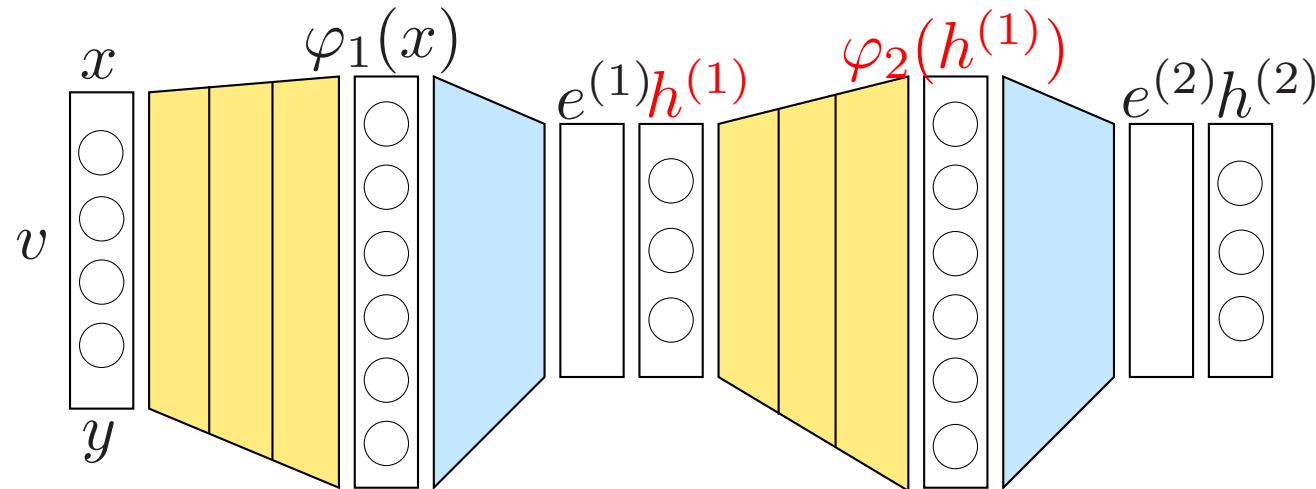
Multiple *levels* and multiple *layers* per level.

in more detail ...



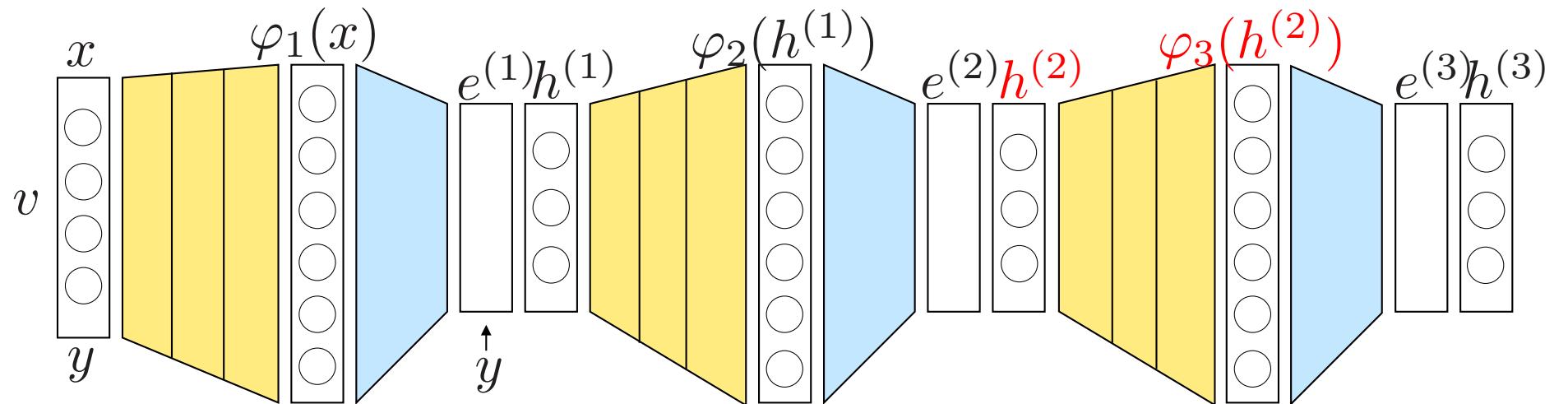
$$J_{\text{deep}} = \sum_{i=1}^N (y_i^T - \varphi_1(x_i)^T W_1 - b^T) h_i^{(1)} - \frac{\lambda_1}{2} \sum_{i=1}^N h_i^{(1)T} h_i^{(1)} + \frac{\eta_1}{2} \text{Tr}(W_1^T W_1)$$

in more detail ...



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

in more detail ...



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

## Stationary points

Stationary points of  $J_{\text{deep}}(h_i^{(1)}, W_1, b, h_i^{(2)}, W_2, h_i^{(3)}, W_3)$  are given by

$$\left\{ \begin{array}{l} \frac{\partial J_{\text{deep}}}{\partial h_i^{(1)}} = 0 \Rightarrow y_i - W_1^T \varphi_1(x_i) - b = \lambda_1 h_i^{(1)} + \frac{\partial}{\partial h_i^{(1)}} [\varphi_2(h_i^{(1)})^T W_2 h_i^{(2)}], \forall i \\ \frac{\partial J_{\text{deep}}}{\partial W_1} = 0 \Rightarrow W_1 = \frac{1}{\eta_1} \sum_i \varphi_1(x_i) h_i^{(1)T} \\ \frac{\partial J_{\text{deep}}}{\partial b} = 0 \Rightarrow \sum_i h_i^{(1)} = 0 \\ \frac{\partial J_{\text{deep}}}{\partial h_i^{(2)}} = 0 \Rightarrow W_2^T \varphi_2(h_i^{(1)}) = \lambda_2 h_i^{(2)} - \frac{\partial}{\partial h_i^{(2)}} [\varphi_3(h_i^{(2)})^T W_3 h_i^{(3)}], \forall i \\ \frac{\partial J_{\text{deep}}}{\partial W_2} = 0 \Rightarrow W_2 = \frac{1}{\eta_2} \sum_i \varphi_2(h_i^{(1)}) h_i^{(2)T} \\ \frac{\partial J_{\text{deep}}}{\partial h_i^{(3)}} = 0 \Rightarrow W_3^T \varphi_3(h_i^{(2)}) = \lambda_3 h_i^{(3)}, \forall i \\ \frac{\partial J_{\text{deep}}}{\partial W_3} = 0 \Rightarrow W_3 = \frac{1}{\eta_3} \sum_i \varphi_3(h_i^{(2)}) h_i^{(3)T} \end{array} \right.$$

## Kernel trick

Elimination of  $W_1, W_2, W_3$  and application kernel trick to each level 1,2,3  
 $\rightarrow$  set of nonlinear equations:

$$\left\{ \begin{array}{l} y_i = \frac{1}{\eta_1} \sum_j h_j^{(1)} K_1(x_j, x_i) + b + \lambda_1 h_i^{(1)} + \frac{1}{\eta_2} \sum_j \frac{\partial K_2(h_i^{(1)}, h_j^{(1)})}{\partial h_i^{(1)}} h_j^{(2)T} h_i^{(2)}, \quad \forall i \\ \sum_i h_i^{(1)} = 0 \\ \frac{1}{\eta_2} \sum_j h_j^{(2)} K_2(h_j^{(1)}, h_i^{(1)}) = \lambda_2 h_i^{(2)} - \frac{1}{\eta_3} \sum_j \frac{\partial K_3(h_i^{(2)}, h_j^{(2)})}{\partial h_i^{(2)}} h_j^{(3)T} h_i^{(3)}, \quad \forall i \\ \frac{1}{\eta_3} \sum_j h_j^{(3)} K_3(h_j^{(2)}, h_i^{(2)}) = \lambda_3 h_i^{(3)}, \quad \forall i \end{array} \right.$$

$$\begin{array}{c} (P)_{\text{DeepRKM}} : \quad \hat{y} = W_1^T \varphi_1(x) + b \\ \nearrow \\ \mathcal{M} \\ \searrow \\ (D)_{\text{DeepRKM}} : \quad \hat{y} = \frac{1}{\eta_1} \sum_j h_j^{(1)} K_1(x_j, x) + b \end{array}$$

## Data fusion interpretation

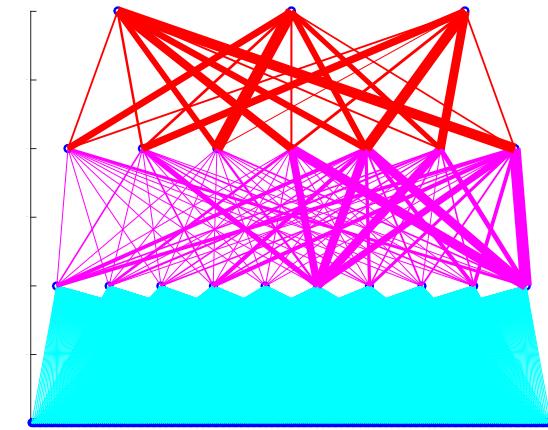
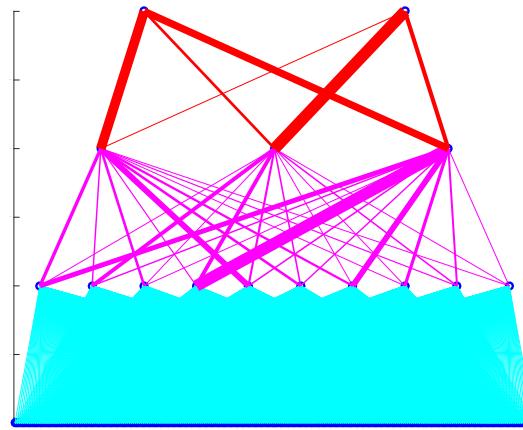
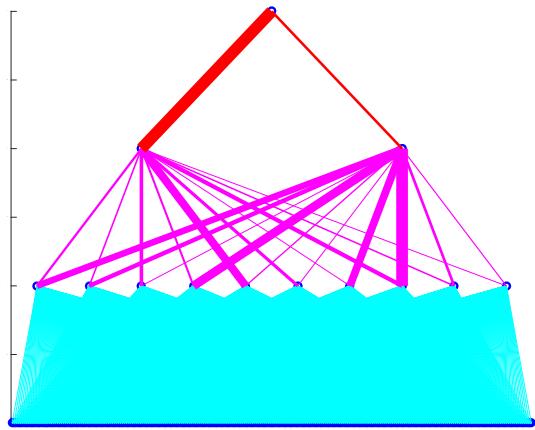
- Special case of linear kernel PCA levels

$$\left\{ \begin{array}{l} \text{Level 1 : } \left[ \begin{array}{c|c} \frac{\frac{1}{\eta_1}[K_1(x_j, x_i)] + \frac{1}{\eta_2}[K_{\text{lin}}(h_j^{(2)}, h_i^{(2)})] + \lambda_1 I_N}{1_N^T} & \begin{matrix} 1_N \\ 0 \end{matrix} \\ \hline & \begin{matrix} H_1^T \\ b^T \end{matrix} \end{array} \right] = \left[ \begin{array}{c} Y^T \\ 0 \end{array} \right] \\ \text{Level 2 : } \left( \frac{1}{\eta_2}[K_{2,\text{lin}}(h_j^{(1)}, h_i^{(1)})] + \frac{1}{\eta_3}[K_{\text{lin}}(h_j^{(3)}, h_i^{(3)})] \right) H_2^T = H_2^T \Lambda_2 \\ \text{Level 3 : } \left( \frac{1}{\eta_3}[K_{3,\text{lin}}(h_j^{(2)}, h_i^{(2)})] \right) H_3^T = H_3^T \Lambda_3 \end{array} \right.$$

with  $H_l = [h_1^{(l)} \dots h_N^{(l)}]$ ,  $l = 1, 2, 3$ .

- Data fusion:
  - Level 1: between  $K_1$  and  $K_{\text{lin}}$
  - Level 2: between  $K_{2,\text{lin}}$  and  $K_{\text{lin}}$

## Deep RKM - Example USPS data



USPS (10 classes): Deep RKM: LSSVM ( $K_{\text{rbf}}$ ) + KPCA ( $K_{\text{lin}}$ ) + KPCA ( $K_{\text{lin}}$ )

Training algorithm: forward & backward phases, kernel fusion between levels

$N = 2000$ : test error 3.26% (basic) - 3.18% (deep) ( $N_{\text{test}} = 5000$ )

$N = 4000$ : test error 2.14% (basic) - 2.12% (deep) ( $N_{\text{test}} = 5000$ )

## Deep RKM - Example MNIST data

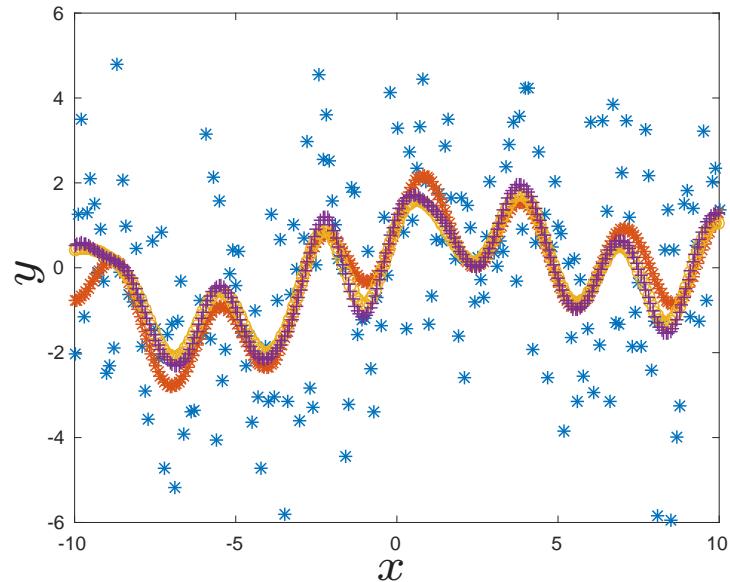
$d = 784$  (images of size 28x28 for each of the 10 classes),  $N = 50000$  partitioned into non-overlapping subsets of size 50 (i.e. 5 data points/class)

Deep RKMs on subsets: LSSVM ( $K_{\text{rbf}}$ ) + KPCA ( $K_{\text{lin}}$ ) + KPCA ( $K_{\text{lin}}$ )  
forward-backward phases  
extra 50000 noise corrupted training data  
linearly combined submodels (tanh applied to their outputs)

Test error: 1.28%

Deep belief networks	(1.2%)
Deep Boltzmann machines	(0.95, 1.01%)
SVM with Gaussian kernel	(1.4%)

## Deep RKM - Example nonlinear regression

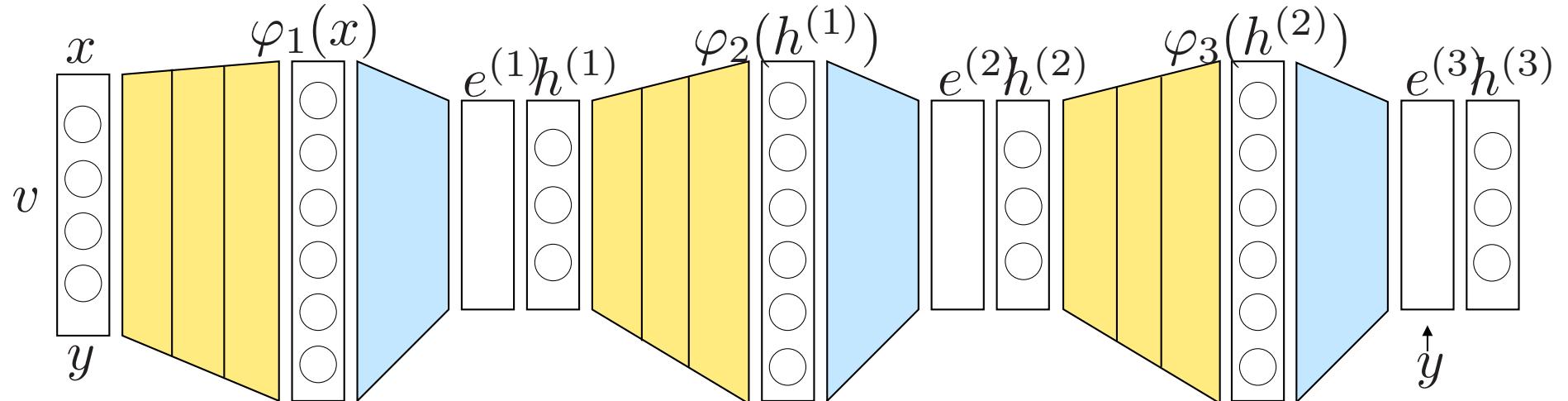


$$f(x) = \sin(0.3x) + \cos(0.5x) + \sin(2x)$$

with zero mean Gaussian noise with standard deviation 0.1, 0.5, 1, 2

noise	LS-SVM	deep (1+1)	deep (7+2)
0.1	$0.0019 \pm 4.3 \cdot 10^{-4}$	$0.0018 \pm 4.2 \cdot 10^{-4}$	$0.0019 \pm 4.4 \cdot 10^{-4}$
0.5	$0.0403 \pm 0.0098$	$0.0374 \pm 0.0403$	$0.0397 \pm 0.0089$
1	$0.1037 \pm 0.0289$	$0.0934 \pm 0.0269$	$0.0994 \pm 0.0301$
2	$0.3368 \pm 0.0992$	$0.2902 \pm 0.0875$	$0.3080 \pm 0.0954$

## Deep RKM: Other example



Deep RKM: KPCA + KPCA + LSSVM

Coupling of RKMs by taking sum of the objectives

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

## Primal and dual model representations

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

## Deep reduced set kernel-based models (1)

Subset of training set  $\{\tilde{x}_j\}_{j=1}^M \subset \{x_i\}_{i=1}^N$  with  $M \ll N$ :

$$\begin{aligned} W_1 &= \frac{1}{\eta_1} \sum_{i=1}^N \varphi_1(x_i) h_i^{(1)T} \simeq \tilde{W}_1 = \frac{1}{\eta_1} \sum_{j=1}^M \varphi_1(\tilde{x}_j) \tilde{h}_j^{(1)T} \\ \tilde{W}_2 &= \frac{1}{\eta_2} \sum_{j=1}^M \varphi_2(\tilde{h}_j^{(1)}) \tilde{h}_j^{(2)T} \\ \tilde{W}_3 &= \frac{1}{\eta_3} \sum_{j=1}^M \varphi_2(\tilde{h}_j^{(2)}) \tilde{h}_j^{(3)T} \end{aligned}$$

Predictive model:

$$\begin{aligned} \hat{e}^{(1)} &= \frac{1}{\eta_1} \sum_{j=1}^M \tilde{h}_j^{(1)} K_1(\tilde{x}_j, x) \\ \hat{e}^{(2)} &= \frac{1}{\eta_2} \sum_{j=1}^M \tilde{h}_j^{(2)} K_2(\tilde{h}_j^{(1)}, \Lambda_1^{-1} \hat{e}^{(1)}) \\ \hat{y} &= \frac{1}{\eta_3} \sum_{j=1}^M \tilde{h}_j^{(3)} K_3(\tilde{h}_j^{(2)}, \Lambda_2^{-1} \hat{e}^{(2)}) + b \end{aligned}$$

## Deep reduced set kernel-based models (2)

Training in primal:

$$\begin{aligned}
 \min_{\tilde{h}_j^{(1)}, \tilde{h}_j^{(2)}, \tilde{h}_j^{(3)}, b, \Lambda_1, \Lambda_2} J_{\text{deep}, P_{\text{stab}}} = & -\frac{1}{2} \sum_{j=1}^M e_j^{(1)T} \Lambda_1^{-1} e_j^{(1)} + \frac{\eta_1}{2} \text{Tr}(\tilde{W}_1^T \tilde{W}_1) \\
 & -\frac{1}{2} \sum_{j=1}^M e_j^{(2)T} \Lambda_2^{-1} e_j^{(2)} + \frac{\eta_2}{2} \text{Tr}(\tilde{W}_2^T \tilde{W}_2) \\
 & + \frac{1}{2\lambda_3} \sum_{j=1}^M e_j^{(3)T} e_j^{(3)} + \frac{\eta_3}{2} \text{Tr}(\tilde{W}_3^T \tilde{W}_3) \\
 & + \frac{1}{2} \mathbf{c}_{\text{stab}} \left( -\frac{1}{2} \sum_{j=1}^M e_j^{(1)T} \Lambda_1^{-1} e_j^{(1)} + \frac{\eta_1}{2} \text{Tr}(\tilde{W}_1^T \tilde{W}_1) \right)^2 \\
 & + \frac{1}{2} \mathbf{c}_{\text{stab}} \left( -\frac{1}{2} \sum_{j=1}^M e_j^{(2)T} \Lambda_2^{-1} e_j^{(2)} + \frac{\eta_2}{2} \text{Tr}(\tilde{W}_2^T \tilde{W}_2) \right)^2
 \end{aligned}$$

with stabilization terms.

## Deep feedforward neural networks in the primal

Model

$$\begin{aligned}\hat{e}^{(1)} &= W_1^T \sigma(U_1 x + \beta_1) \\ \hat{e}^{(2)} &= W_2^T \sigma(U_2 \Lambda_1^{-1} \hat{e}^{(1)} + \beta_2) \\ \hat{y} &= W_3^T \sigma(U_3 \Lambda_2^{-1} \hat{e}^{(2)} + \beta_3) + b\end{aligned}$$

Training objective:

$$\begin{aligned}\min_{W_{1,2,3}, U_{1,2,3}, \beta_{1,2,3}, b, \Lambda_1, \Lambda_2} J_{\text{deep, P}_{\text{stab}}} = & -\frac{1}{2} \sum_{j=1}^M e_j^{(1)T} \Lambda_1^{-1} e_j^{(1)} + \frac{\eta_1}{2} \text{Tr}(W_1^T W_1) \\ & -\frac{1}{2} \sum_{j=1}^M e_j^{(2)T} \Lambda_2^{-1} e_j^{(2)} + \frac{\eta_2}{2} \text{Tr}(W_2^T W_2) \\ & + \frac{1}{2\lambda_3} \sum_{j=1}^M e_j^{(3)T} e_j^{(3)} + \frac{\eta_3}{2} \text{Tr}(W_3^T W_3) \\ & + \frac{1}{2} \mathbf{c}_{\text{stab}} \left( -\frac{1}{2} \sum_{j=1}^M e_j^{(1)T} \Lambda_1^{-1} e_j^{(1)} + \frac{\eta_1}{2} \text{Tr}(W_1^T W_1) \right)^2 \\ & + \frac{1}{2} \mathbf{c}_{\text{stab}} \left( -\frac{1}{2} \sum_{j=1}^M e_j^{(2)T} \Lambda_2^{-1} e_j^{(2)} + \frac{\eta_2}{2} \text{Tr}(W_2^T W_2) \right)^2\end{aligned}$$

## Toeplitz matrices (convolutional)

Taking matrices  $U \in \mathbb{R}^{n_1 \times n_2}$  as Toeplitz matrices:

$$\begin{bmatrix} a & b & c & d & & \\ e & a & b & c & & \\ f & e & a & b & & \\ g & f & e & a & & \\ & & & & \ddots & \end{bmatrix}$$

→ reduces to  $n_1 + n_2 - 1$  instead of  $n_1 n_2$  unknowns

## Experimental results

	pid	bld	ion	adu
$\mathcal{M}_{1,a}$	19.53 [20.02(1.53)]	26.09 [30.96(3.34)]	<b>0</b> [0.68(1.60)]	16.99 [17.46(0.65)]
$\mathcal{M}_{1,b}$	<b>18.75</b> [19.39(0.89)]	<b>25.22</b> [31.48(4.11)]	<b>0</b> [5.38(12.0)]	17.08 [17.48(0.56)]
$\mathcal{M}_{1,c}$	21.88 [24.73(5.91)]	28.69 [32.39(3.48)]	<b>0</b> [8.21(6.07)]	17.83 [21.21(4.78)]
$\mathcal{M}_{2,a}$	21.09 [20.20(1.51)]	27.83 [28.86(2.83)]	1.71 [5.68(2.22)]	15.07 [15.15(0.15)]
$\mathcal{M}_{2,b}$	<b>18.75</b> [20.33(2.75)]	28.69 [28.38(2.80)]	10.23 [6.92(3.69)]	14.91 [15.08 (0.15)]
$\mathcal{M}_{2,b,T}$	19.03 [19.16(1.10)]	26.08 [27.74(9.40)]	6.83 [6.50(8.31)]	15.71 [15.97(0.07)]
$\mathcal{M}_{2,c}$	24.61 [22.34(1.95)]	32.17 [27.61(3.69)]	3.42 [9.66(6.74)]	15.21 [15.19(0.08)]
bestbmark	22.7(2.2)	29.6(3.7)	4.0(2.1)	14.4(0.3)

$\mathcal{M}_1$ : Deep reduced set kernel-based models (with RBF kernel)

$\mathcal{M}_{1,a}$ : additional term  $-c_0(\text{Tr}(\Lambda_1) + \text{Tr}(\Lambda_2))$  and  $\text{Tr}(\tilde{H}^{(l)}\tilde{H}^{(l)T})$  regularization

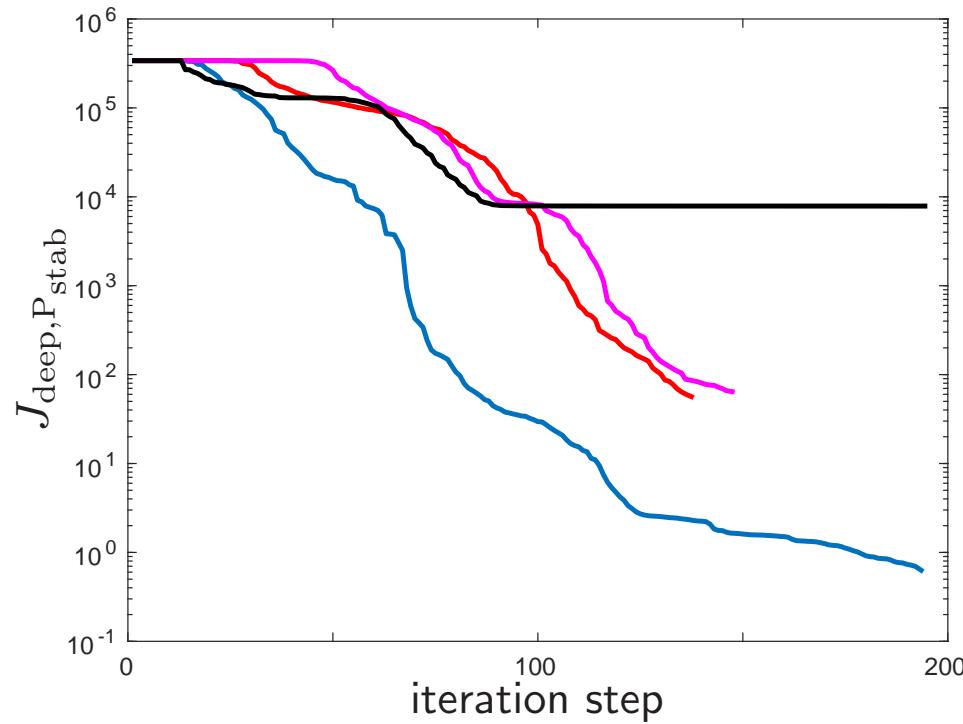
$\mathcal{M}_{1,b}$ : without term  $-c_0(\text{Tr}(\Lambda_1) + \text{Tr}(\Lambda_2))$

$\mathcal{M}_{1,c}$ : with objective function  $\frac{1}{2\lambda_3} \sum_{j=1}^M e_j^{(3)T} e_j^{(3)} + \frac{\eta_3}{2} \text{Tr}(W_3^T W_3)$

$\mathcal{M}_2$ : Deep feedforward neural networks

$\mathcal{M}_{2,b,T}$ : with Toeplitz matrices for  $U$  matrices

## Training process



Evolution of the objective function (logarithmic scale) during training on the ion data set. Shown are training curves for the model  $\mathcal{M}_{2,a}$  for different choices of  $c_{\text{stab}}$  (equal to 1, 10, 100 in blue, red, magenta color, respectively) in comparison with  $\mathcal{M}_{2,c}$  (level 3 objective only, in black color), for the same initialization, with quasi-Newton method.

## Future challenges

- efficient algorithms and implementations for large data
- extension to other loss functions and regularization schemes
- multimodal data, tensor models
- different coupling schemes
- models for deep clustering and semi-supervised learning
- combination with convolutional layers
- choice kernel functions, invariances
- generative models

## Conclusions

- RKM: new representation of kernel machines by conjugate feature duality
- RKM & RBM: visible and hidden units
- Deep RKM: new framework for deep kernel machines and deep feedforward neural networks

More information:

Suykens J.A.K., “Deep Restricted Kernel Machines using Conjugate Feature Duality”, *Neural Computation*, 29(8), 2123-2163, Aug 2017

## Acknowledgements



**Thank you**