Deep Learning, Neural Networks and Kernel Machines

Johan Suykens

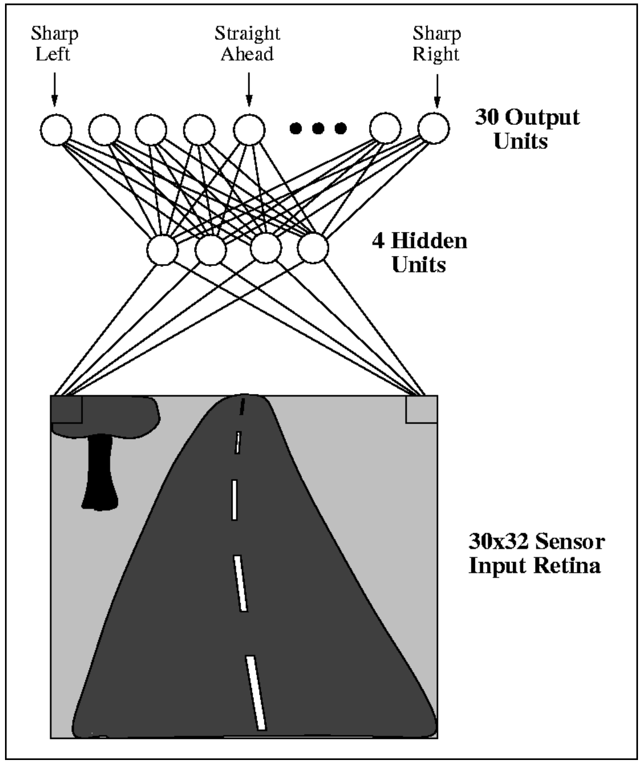
KU Leuven, ESAT-STADIUS
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
Email: johan.suykens@esat.kuleuven.be
http://www.esat.kuleuven.be/stadius/

Deeplearn 2019, Warsaw Poland, July 2019

Introduction

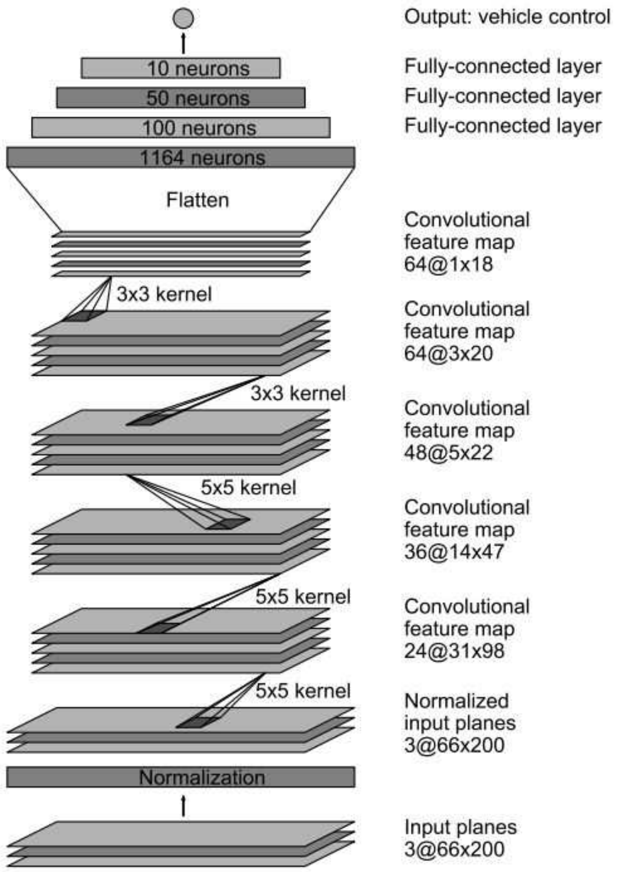
Self-driving cars and neural networks

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



(27 million connections)

Waymo / Google Self-Driving Car

Tesla Autopilot

Uber

nuTonomy

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

Convolutional neural networks

PROC. OF THE IEEE, NOVEMBER 1998 7

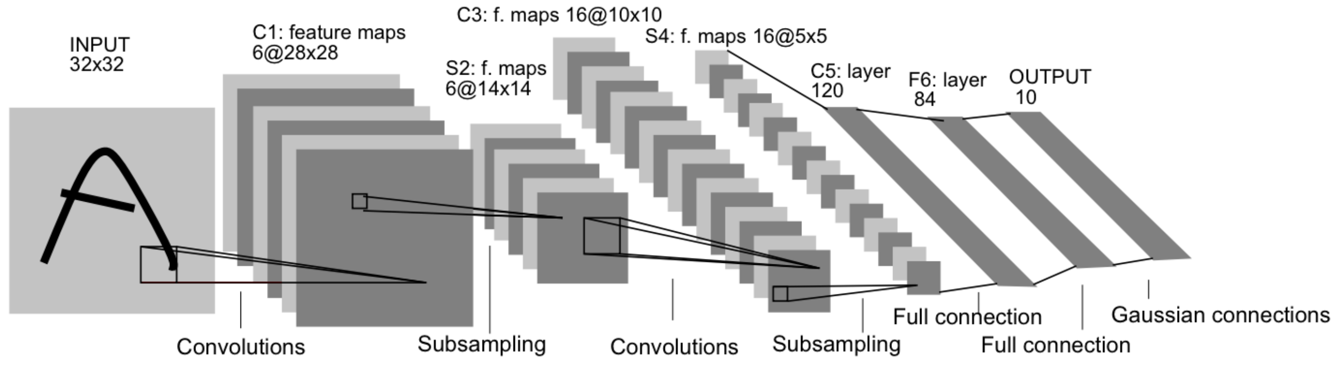


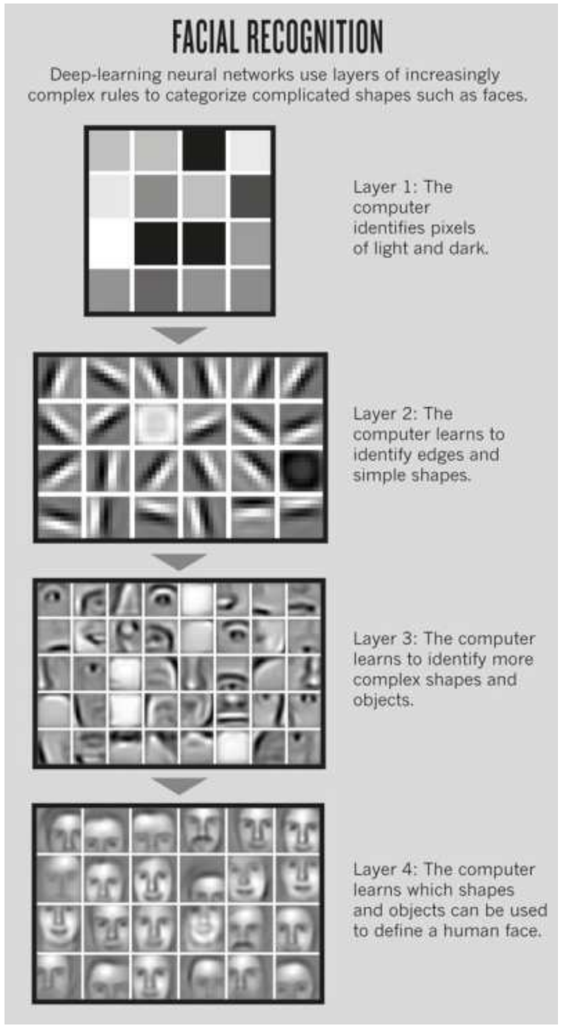
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

[LeCun et al., Proc. IEEE 1998]

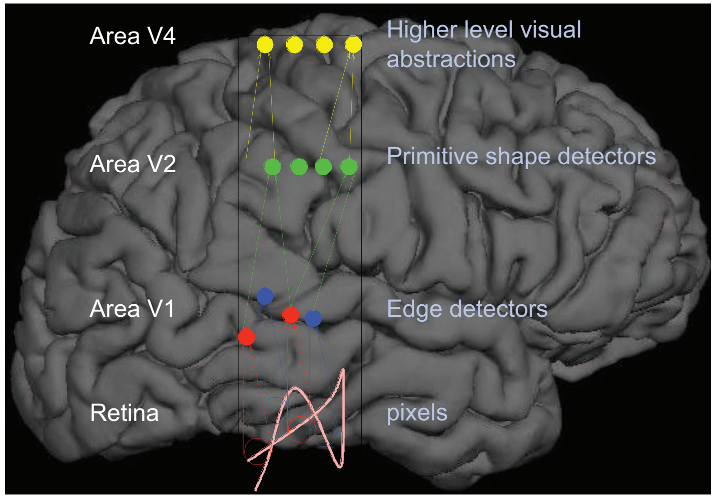
Further advanced architectures:

Alexnet (2012):	5 convolutional layers, 3 fully connected
VGGnet (2014):	19 layers
GoogLeNet (2014):	22 layers
ResNet (2015):	152 layers

Hierarchical features


FACIAL RECOGNITION
Deep-learning neural networks use layers of increasingly complex rules to categorize complicated shapes such as faces.
Layer 1: The computer identifies pixels of light and dark.
Layer 2: The computer learns to identify edges and simple shapes.
Layer 3: The computer learns to identify more complex shapes and objects.
Layer 4: The computer learns which shapes and objects can be used to define a human face.

Deep Architecture in the Brain



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

Historical context

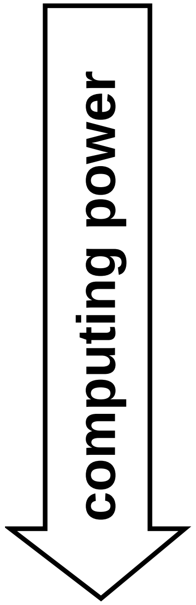
1942	McCulloch & Pitts: mathematical model for neuron
1958	Rosenblatt: perceptron learning
1960	Widrow & Hoff: adaline and lms learning rule
1969	Minsky & Papert: limitations of perceptron
1986	Rumelhart et al.: error backpropagating neural networks
	$\rightarrow$ *booming of neural network universal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



Towards a unifying picture

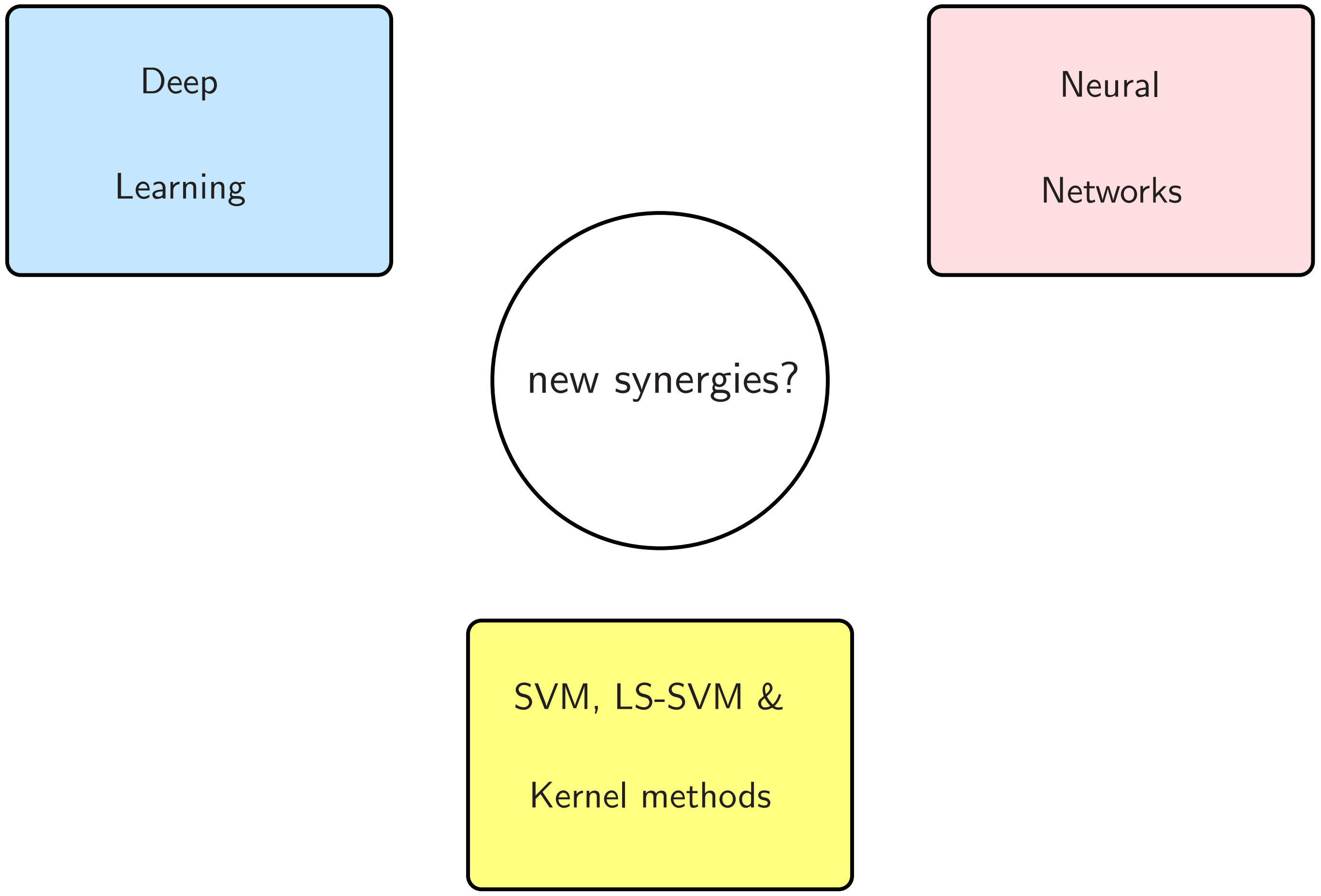
Different paradigms

Deep Learning

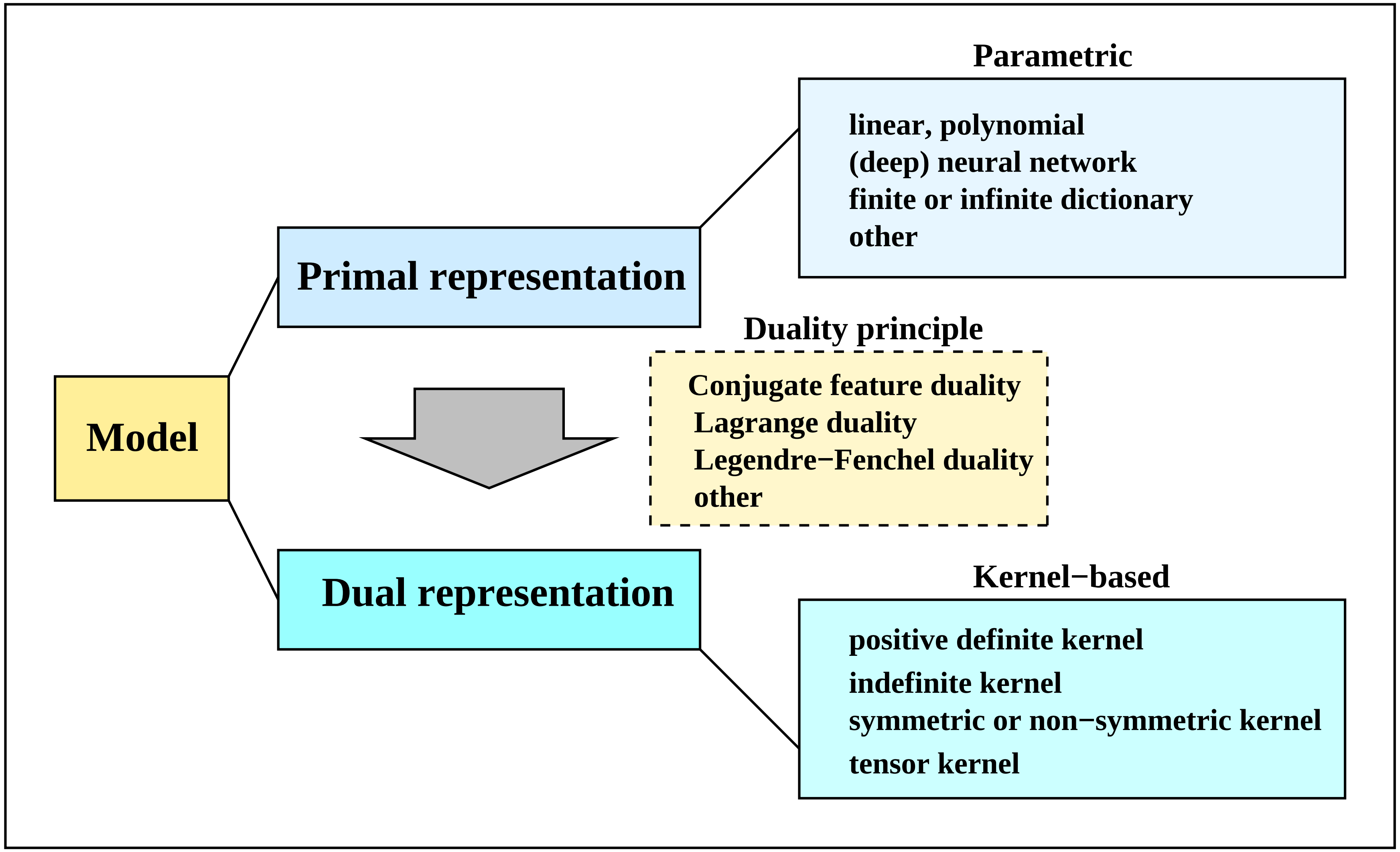
Neural Networks

SVM, LS-SVM & Kernel methods

Different paradigms



Towards a unifying picture



[Suykens 2017]

Outline

- **Part I:** Neural networks, support vector machines and kernel methods
- **Part II:** Restricted Boltzmann machines, kernel machines and deep learning
- **Part III:** Deep restricted kernel machines and future perspectives

Part I: Neural networks, (LS)-SVMs and kernel methods

- Function estimation: primal and dual model representations
- Neural networks, support vector machines and kernel-based methods
- Least squares support vector machines in supervised and unsupervised learning
- Sparsity, robustness, loss functions, re-weighting
- Fixed-size kernel methods for large scale problems, use for deep learning
- Multi-view learning: deep neural networks and kernel methods
- Applications: black-box weather forecasting, pollution modelling, prediction of energy consumption, community detection in networks

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:

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

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 $\{(x_i, y_i)\}_{i=1}^N$

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

$\nearrow$

Model

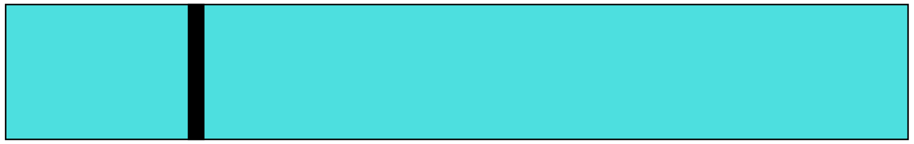
Linear model: solving in primal or dual?

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

$$\begin{tikzcd}
 & (P) : \quad \hat{y} = w^T x + b, \qquad w \in \mathbb{R}^d \\
\text{Model} \arrow[ur] \arrow[dr] & \\
 & (D) : \quad \hat{y} = \sum_i \alpha_i x_i^T x + b, \qquad \alpha \in \mathbb{R}^N
\end{tikzcd}$$

$\boxed{\text{Linear model: solving in primal or dual?}}$

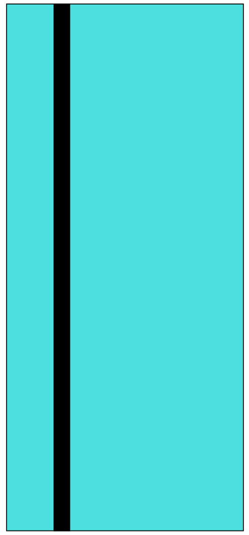
few inputs, many data points: $d \ll N$



$\boxed{\textbf{primal}}$: $w \in \mathbb{R}^d$
dual: $\alpha \in \mathbb{R}^N$ (large kernel matrix: $N \times N$)

$\boxed{\text{Linear model: solving in primal or dual?}}$

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



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

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

Feature map and kernel

From linear to nonlinear model:

$$\begin{tikzcd}[column sep=small] & (P): \quad \hat{y} = w^T \varphi(x) + b \\ \text{Model} \arrow[ur] \arrow[dr] & \\ & (D): \quad \hat{y} = \sum_i \alpha_i K(x_i, x) + b \end{tikzcd}$$

Mercer theorem:

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

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

- SVMs: feature map and positive definite kernel [Cortes & Vapnik, 1995]
- Explicit or implicit choice of the feature map
- Neural networks: consider hidden layer as feature map [Suykens & Vandewalle, 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

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

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

Sparsity: through regularization or loss function

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

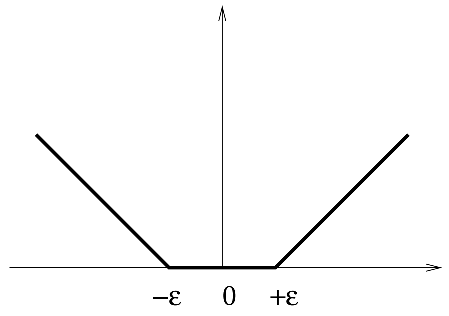
$$\min \sum_j |w_j| + \gamma \sum_i e_i^2$$

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

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

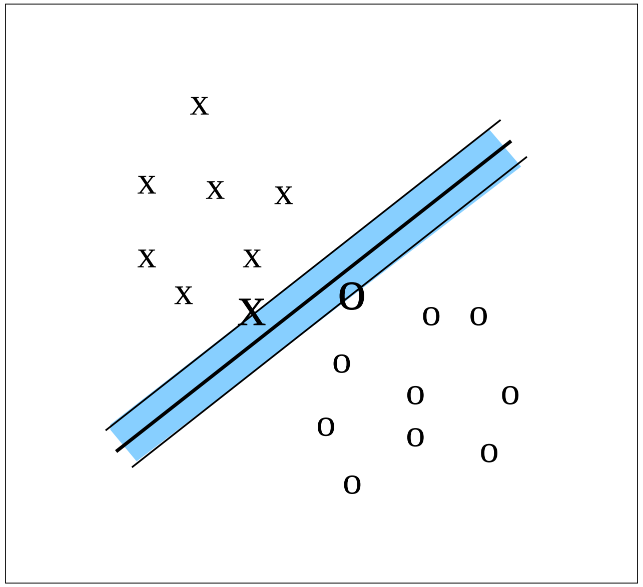
$$\min \; w^T w + \gamma \sum_i L(e_i)$$

$\Rightarrow$ sparse α (e.g. SVM)



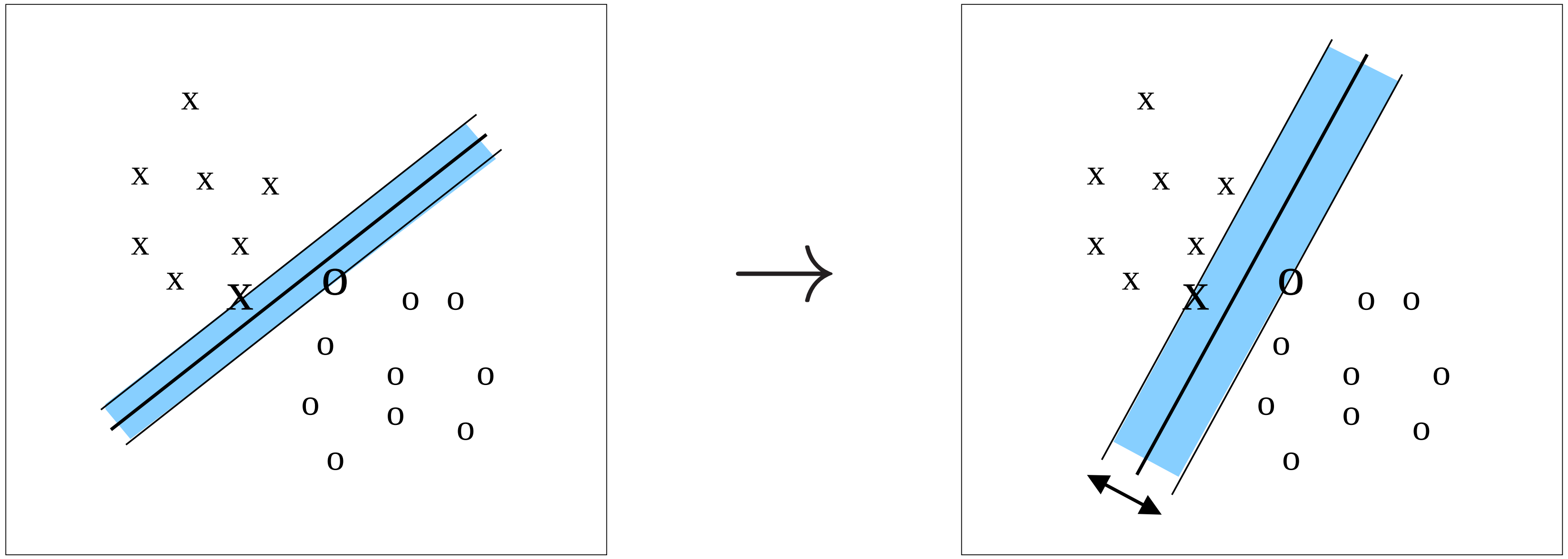
SVM classifier: maximal margin

- Training set $\{(x_i, y_i)\}_{i=1}^{N}$: inputs $x_i \in \mathbb{R}^d$; class labels $y_i \in \{-1, +1\}$
- Classifier: $\hat{y} = \text{sign}[w^T \varphi(x) + b]$
  with $\varphi(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{n_h}$ the mapping to a high dimensional feature space
- **Maximize the margin** for good generalization ability (margin $= \frac{2}{\|w\|_2}$)



SVM classifier: maximal margin

- Training set $\{(x_i, y_i)\}_{i=1}^N$: inputs $x_i \in \mathbb{R}^d$; class labels $y_i \in \{-1, +1\}$
- Classifier: $\hat{y} = \text{sign}[w^T \varphi(x) + b]$
  with $\varphi(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^{n_h}$ the mapping to a high dimensional feature space
- **Maximize the margin** for good generalization ability (margin $= \frac{2}{\|w\|_2}$)



SVM classifier: primal and dual problem

- **Primal problem:** [Vapnik, 1995]

$$\min_{w,b,\xi} \frac{1}{2} w^T w + c \sum_{i=1}^{N} \xi_i \ \text{ s.t. } \left\{ \begin{array}{c} y_i [w^T \varphi(x_i) + b] \geq 1 - \xi_i \\ \xi_i \geq 0, \quad i = 1, ..., N \end{array} \right.$$

  Trade-off between margin maximization and tolerating misclassifications

- Conditions for optimality from Lagrangian.
  Express the solution in the Lagrange multipliers.

SVM classifier: primal and dual problem

- **Primal problem:** [Vapnik, 1995]

$$\min_{w,b,\xi} \frac{1}{2} w^T w + c \sum_{i=1}^{N} \xi_i \ \text{ s.t. } \left\{ \begin{array}{c} y_i [w^T \varphi(x_i) + b] \geq 1 - \xi_i \\ \xi_i \geq 0, \quad i = 1, ..., N \end{array} \right.$$

  Trade-off between margin maximization and tolerating misclassifications

- Conditions for optimality from Lagrangian.
  Express the solution in the Lagrange multipliers.

- **Dual problem**: QP problem (convex problem)

$$\max_{\alpha} \ -\frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \, K(x_i, x_j) \, \alpha_i \alpha_j + \sum_{j=1}^{N} \alpha_j \ \text{ s.t. } \left\{ \begin{array}{l} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq c, \ \forall i \end{array} \right.$$

SVM classifier: model representations

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

  Kernel trick (Mercer Theorem):

$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j) = \sum_{l=1}^{n_h} \varphi_l(x_i)\varphi_l(x_j)$$

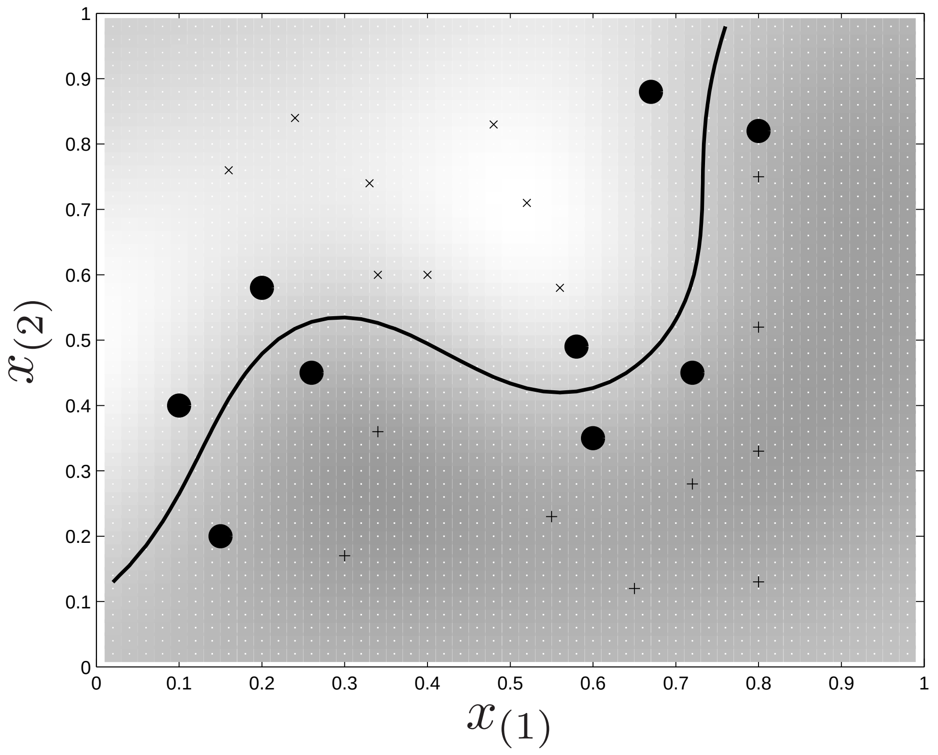
- Dual representation:

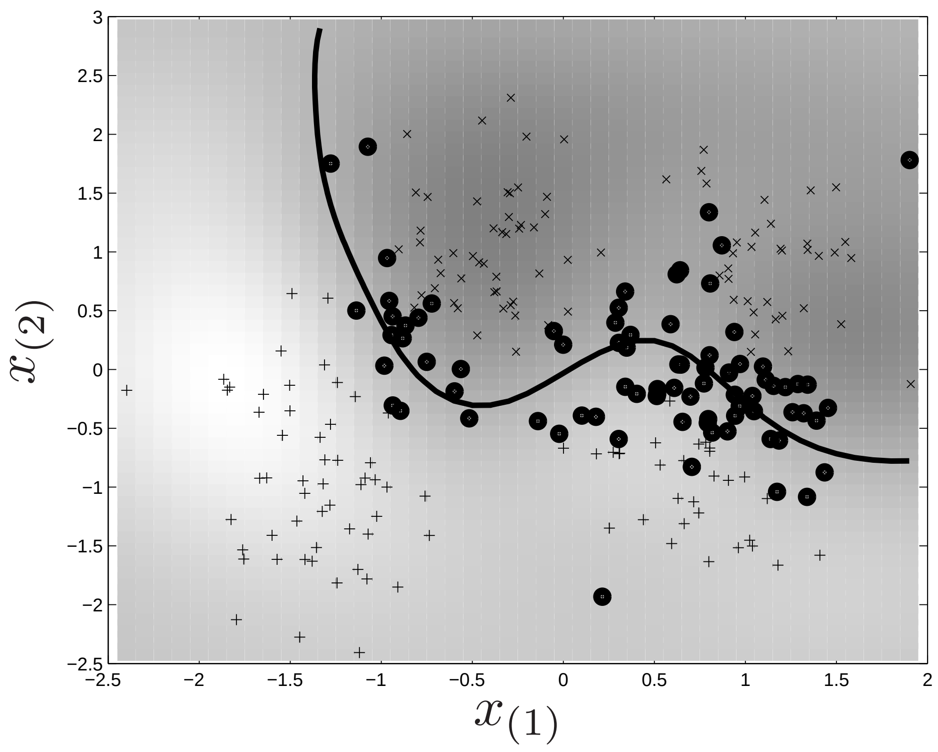
$$\hat{y} = \text{sign}[\sum_i \alpha_i \, y_i \, K(x, x_i) + b]$$

  Some possible kernels:

  $K(x, x_i) = x_i^T x$ (linear)
  $K(x, x_i) = (x_i^T x + \tau)^d$ with $\tau \geq 0$ (polynomial)
  $K(x, x_i) = \exp(-\|x - x_i\|_2^2/\sigma^2)$ (RBF Gaussian)
  $K(x, x_i) = \tanh(\kappa \, x_i^T x + \theta)$ (MLP)

Support vectors

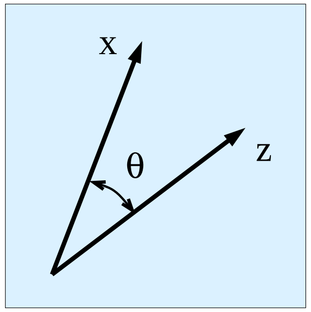




- Decision boundary can be expressed in terms of a limited number of support vectors (subset of given training data); sparseness property

- Classifier follows from the solution to a convex QP problem.

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:

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

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]

Kernels

Wide range of positive definite kernel functions possible:

- linear	$K(x,z) = x^T z$
- polynomial	$K(x,z) = (\eta + x^T z)^d$
- radial basis function	$K(x,z) = \exp(-\|x - z\|_2^2/\sigma^2)$

- splines
- wavelets
- string kernel
- kernels from graphical models
- kernels for dynamical systems
- Fisher kernels
- graph kernels
- data fusion kernels
- other

[Schölkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004; Jebara et al., 2004; other]

Krein spaces: indefinite kernels

- LS-SVM for indefinite kernel case:

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

  and **indefinite kernel** $K(x_i, x_j) = K_+(x_i, x_j) - K_-(x_i, x_j)$
  with positive definite kernels K_+, K_-

$$K_+(x_i, x_j) = \varphi_+(x_i)^T \varphi_+(x_j) \text{ and } K_-(x_i, x_j) = \varphi_-(x_i)^T \varphi_-(x_j)$$

- also: KPCA with indefinite kernel [X. Huang et al. 2017], KSC and semi-supervised learning [Mehrkanoon et al., 2018]

[X. Huang, Maier, Hornegger, Suykens, ACHA 2017]
[Mehrkanoon, X. Huang, Suykens, Pattern Recognition, 2018]
Related work of RKKS: [Ong et al 2004; Haasdonk 2005; Luss 2008; Loosli et al. 2015]

Banach spaces: tensor kernels

- Regression problem:

$$\begin{array}{ll} \min\limits_{(w,b,e)\in\ell^r(\mathbb{K})\times\mathbb{R}\times\mathbb{R}^N} & \rho(\|w\|_r) + \frac{\gamma}{N}\sum_{i=1}^N L(e_i) \\ \text{subject to} & y_i = \langle w, \varphi(x_i)\rangle + b + e_i \,, \forall i = 1, ..., N \end{array}$$

  with $r = \frac{m}{m-1}$ for even $m \geq 2$, ρ convex and even.

  For m large this approaches ℓ^1 regularization.

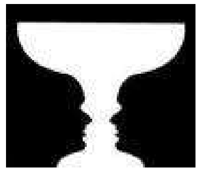
- **Tensor-kernel representation**

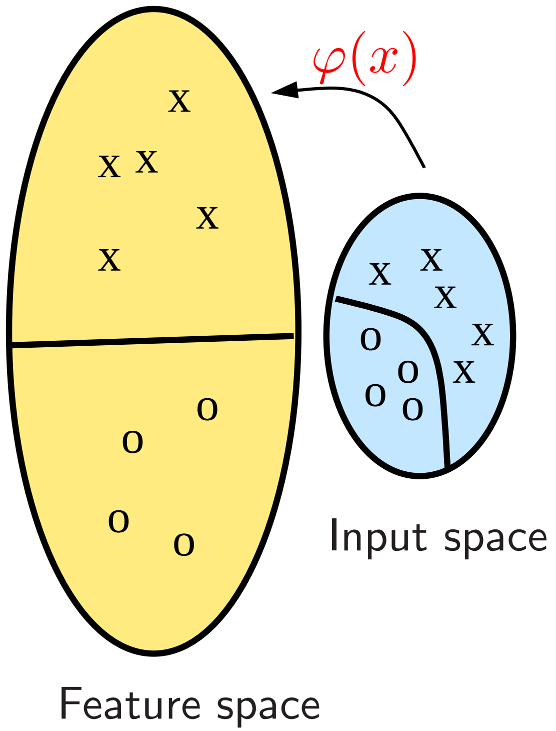
$$\hat{y} = \langle w, \varphi(x)\rangle_{r,r^*} + b = \frac{1}{N^{m-1}} \sum_{i_1,\ldots,i_{m-1}=1}^{N} u_{i_1} \ldots u_{i_{m-1}} K(x_{i_1}, \ldots, x_{i_{m-1}}, x) + b$$

[Salzo & Suykens, arXiv 1603.05876; Salzo, Suykens, Rosasco, AISTATS 2018]

related: RKBS [Zhang 2013; Fasshauer et al. 2015]

SVMs and neural networks

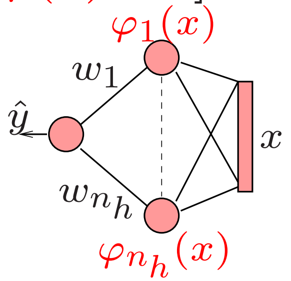




Primal space

Parametric

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

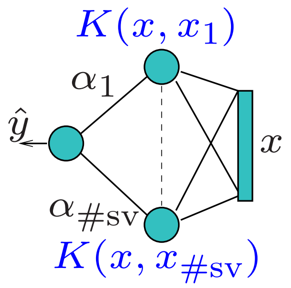


$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$ **(Mercer)**

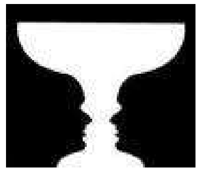
Dual space

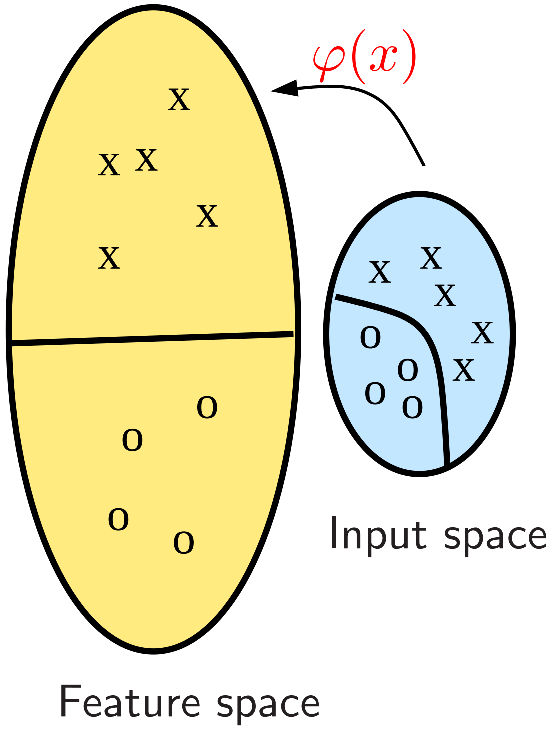
Nonparametric

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



SVMs and neural networks

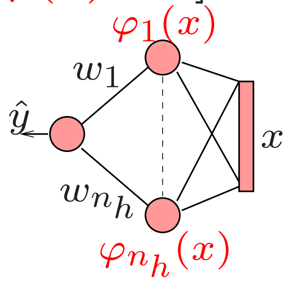




Primal space

Parametric

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

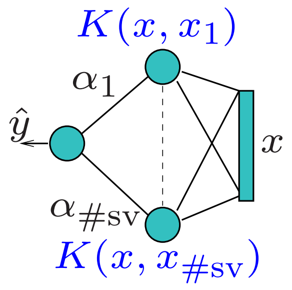


$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$ **("Kernel trick")**

Dual space

Nonparametric

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



Parametric

Non-parametric

Generalization, deep learning and kernel methods

Recently one has observed in deep learning that **over-parametrized neural networks, that would "overfit", may still perform well on test data**. This phenomenon is currently not yet fully understood. A number of researchers have stated that **understanding kernel methods in this context is important** for understanding the generalization performance.

Related references:

- Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals, Understanding deep learning requires rethinking generalization, 2016, arXiv:1611.03530
- Amit Daniely, SGD Learns the Conjugate Kernel Class of the Network, 2017, arXiv:1702.08503
- Arthur Jacot, Franck Gabriel, Clement Hongler, Neural Tangent Kernel: Convergence and Generalization in Neural Networks, 2018, arXiv:1806.07572
- Tengyuan Liang, Alexander Rakhlin, Just Interpolate: Kernel "Ridgeless" Regression Can Generalize, 2018, arXiv:1808.00387
- Mikhail Belkin, Siyuan Ma, Soumik Mandal, To understand deep learning we need to understand kernel learning, 2018, arXiv:1802.01396

Generalization and deep learning - Double U curve

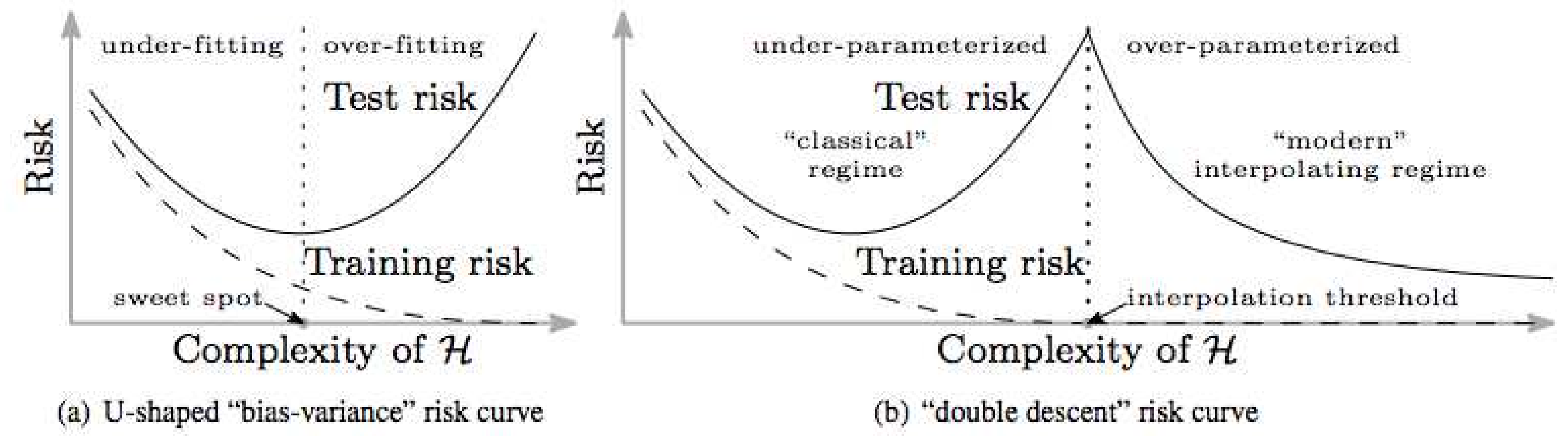
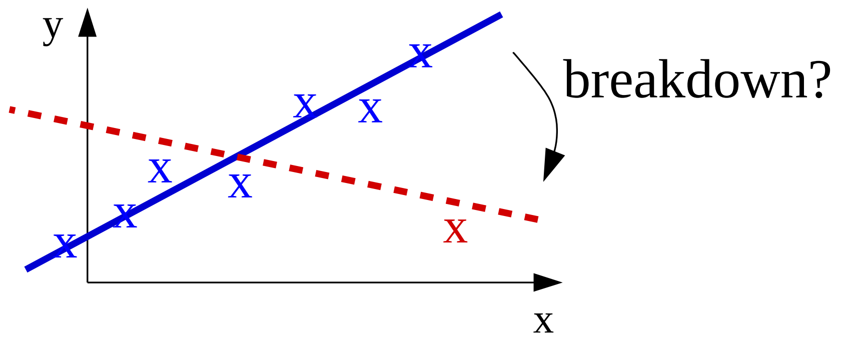


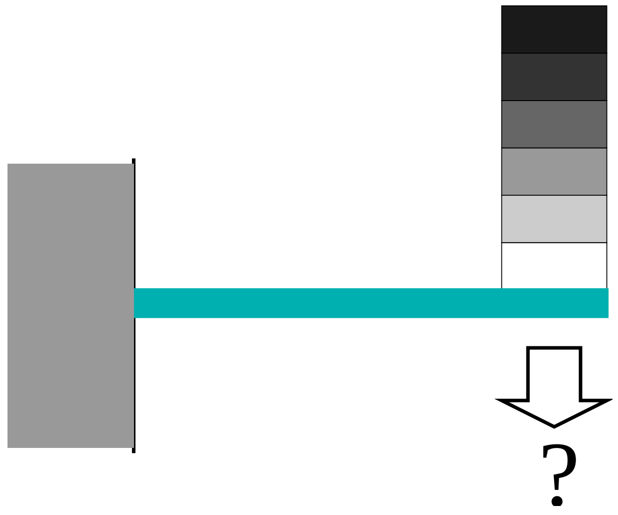
Figure 1: Curves for training risk (dashed line) and test risk (solid line). (a) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (b) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high complexity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Figure: Mikhail Belkin, Daniel Hsu, Siyuan Ma, Soumik Mandal, Reconciling modern machine learning and the bias-variance trade-off, 2018, arXiv:1812.11118

Robustness

Outliers and robustness

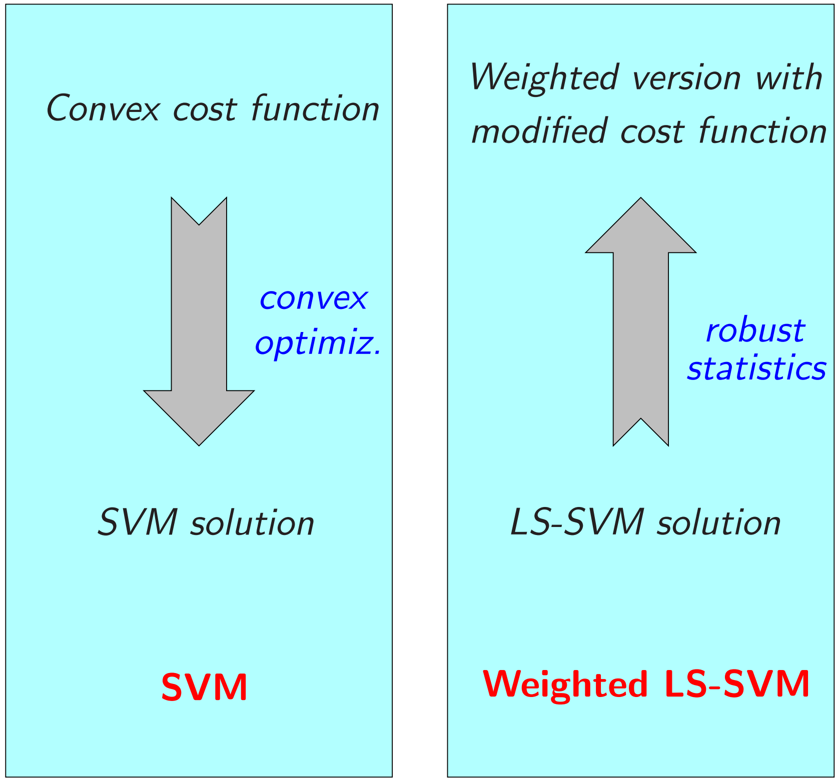




Robust statistics: Bounded derivative of loss function, bounded kernel

[Huber, 1981; Hampel et al., 1986; Rousseeuw & Leroy, 1987]

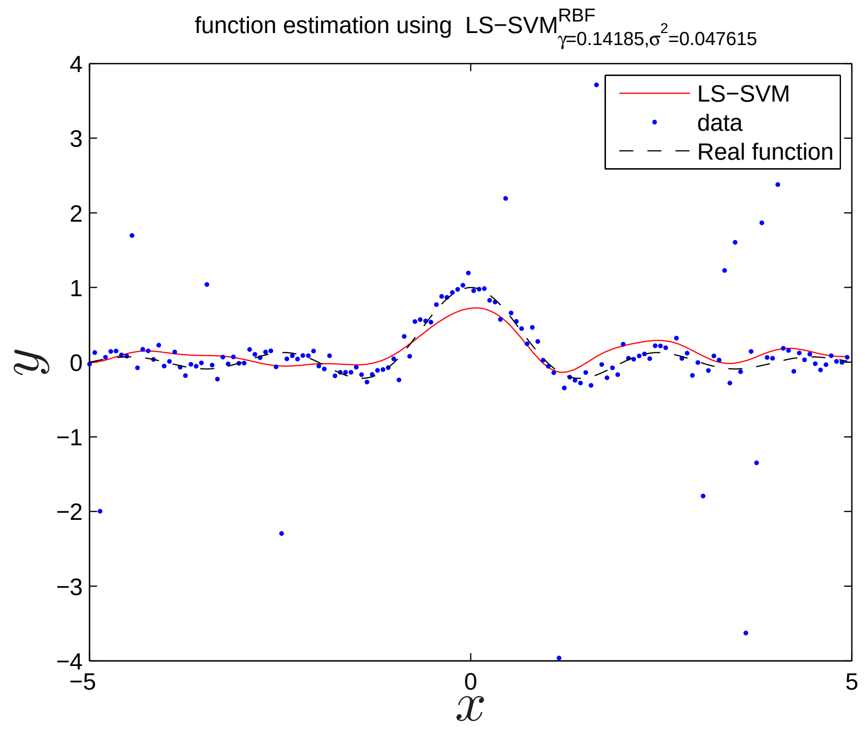
Weighted versions and robustness

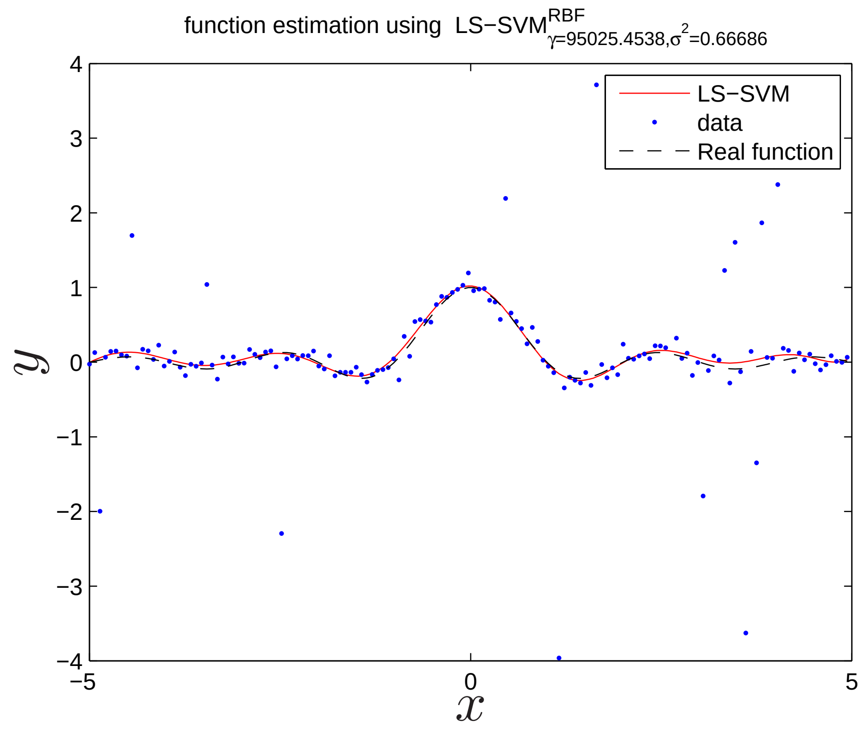


- Weighted LS-SVM: $\min_{w,b,e} \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{i=1}^{N} v_i e_i^2 \text{ s.t. } y_i = w^T \varphi(x_i) + b + e_i, \forall i$

  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/

Fixed-size kernel methods for large scale data

Fixed-size kernel method

- Find finite dimensional approximation to feature map $\tilde{\varphi}(\cdot) : \mathbb{R}^p \rightarrow \mathbb{R}^M$ based on the eigenvalue decomposition of the kernel matrix (on a **subset** of size $M \ll N$).
- Based on [Williams & Seeger, 2001]:
relates KPCA to a Nyström approximation of the integral equation

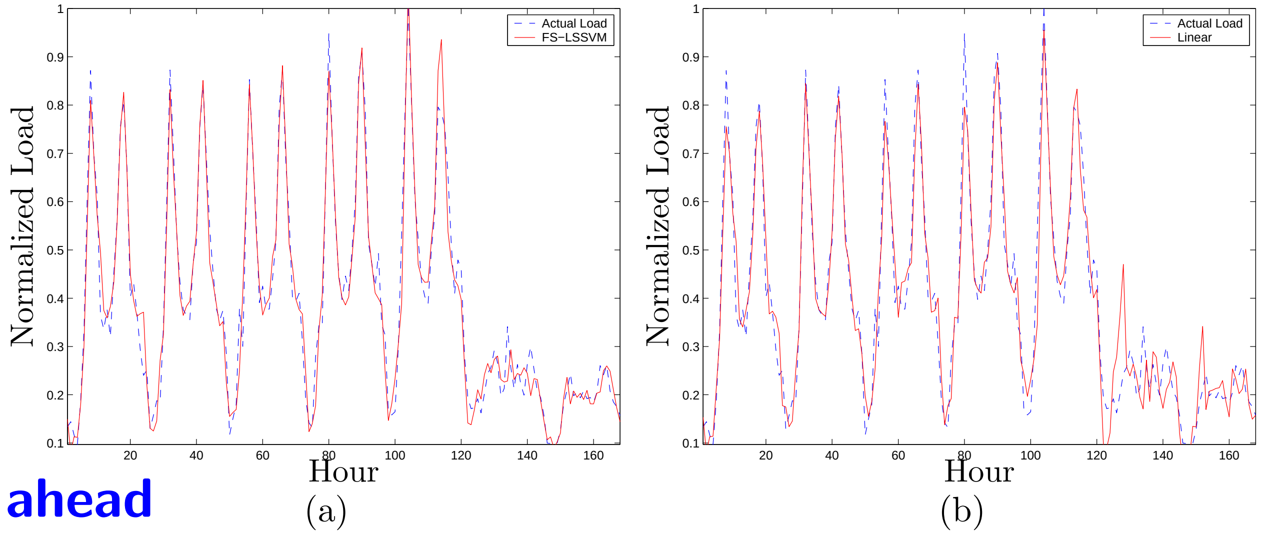
$$\int K(z,x)\phi_i(x)dP_X = \lambda_i \phi_i(z)$$

- Fixed-size method [Suykens et al., 2002; De Brabanter et al., 2009]:
- selects subset such that it represents the data distribution P_X
- optimizes quadratic Renyi entropy citerion (instead of random subset)
- LS-SVM: estimate in **primal** (**sparse** representation):

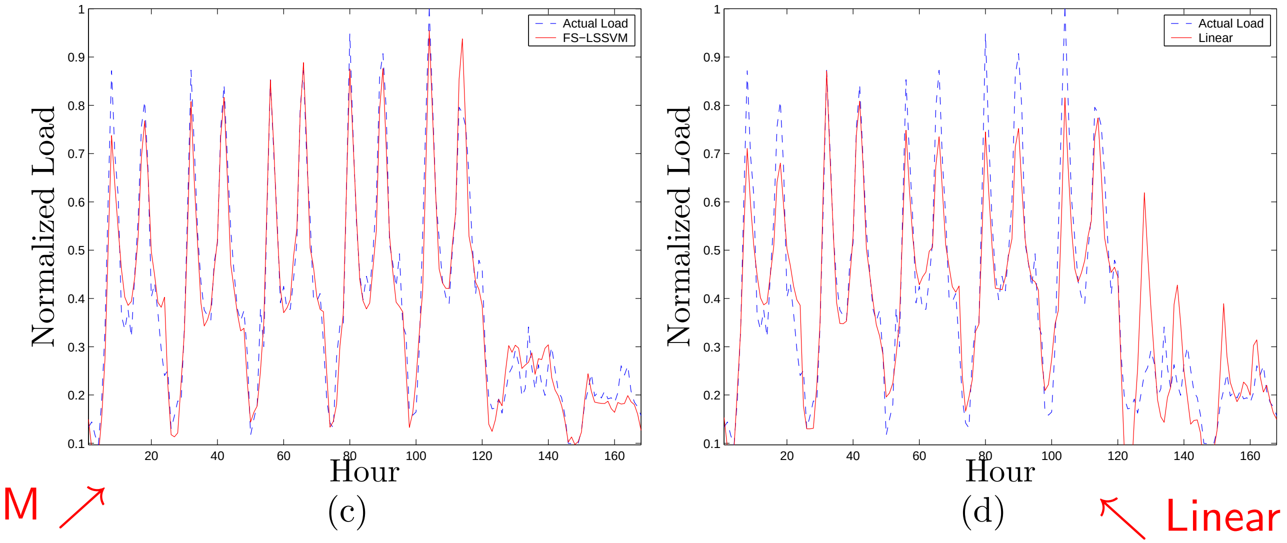
$$\min_{\tilde{w},b} \frac{1}{2}\tilde{w}^T\tilde{w} + \gamma \frac{1}{2}\sum_{i=1}^{N}(y_i - \tilde{w}^T\tilde{\varphi}(x_i) - b)^2$$

Electricity load forecasting

1-hour ahead



24-hours ahead



Fixed-size LS-SVM $\nearrow$ $\nwarrow$ Linear ARX model

[Espinoza, Suykens, Belmans, De Moor, IEEE CSM 2007]

Fixed-size method: performance in classification

	pid	spa	mgt	adu	ftc
N	768	4601	19020	45222	581012
$N_{\rm cv}$	512	3068	13000	33000	531012
$N_{\rm test}$	256	1533	6020	12222	50000
d	8	57	11	14	54
FS-LSSVM (# SV)	150	200	1000	500	500
C-SVM (# SV)	290	800	7000	11085	185000
ν-SVM (# SV)	331	1525	7252	12205	165205
RBF FS-LSSVM	76.7(3.43)	92.5(0.67)	86.6(0.51)	85.21(0.21)	81.8(0.52)
Lin FS-LSSVM	77.6(0.78)	90.9(0.75)	77.8(0.23)	83.9(0.17)	75.61(0.35)
RBF C-SVM	75.1(3.31)	92.6(0.76)	85.6(1.46)	84.81(0.20)	81.5(no cv)
Lin C-SVM	76.1(1.76)	91.9(0.82)	77.3(0.53)	83.5(0.28)	75.24(no cv)
RBF ν-SVM	75.8(3.34)	88.7(0.73)	84.2(1.42)	83.9(0.23)	81.6(no cv)
Maj. Rule	64.8(1.46)	60.6(0.58)	65.8(0.28)	83.4(0.1)	51.23(0.20)

- Fixed-size (FS) LSSVM: good performance and sparsity wrt C-SVM and ν-SVM
- Challenging to achieve high performance by very sparse models

[De Brabanter et al., CSDA 2010]

Two stages of sparsity

primal	
dual	subset selection Nyström approximation

Two stages of sparsity

	stage 1
primal	FS model estimation
	$\uparrow$
dual	subset selection Nyström approximation

Two stages of sparsity

	stage 1		stage 2
primal	FS model estimation	$\longrightarrow$	reweighted ℓ_1
	$\uparrow$		
dual	subset selection Nyström approximation		

Synergy between parametric & kernel-based models

[Mall & Suykens, IEEE-TNNLS 2015], reweighted ℓ_1 [Candes et al., 2008]

Two stages of sparsity

	stage 1		stage 2
primal	FS model estimation	$\longrightarrow$	reweighted ℓ_1
	$\uparrow$		
dual	subset selection		
	Nyström approximation		

Synergy between parametric & kernel-based models

[Mall & Suykens, IEEE-TNNLS 2015], reweighted ℓ_1 [Candes et al., 2008]

Other possible approaches with improved sparsity: SCAD [Fan & Li, 2001]; coefficient-based ℓ_q $(0 < q \leq 1)$ [Shi et al., 2013]; two-level ℓ_1 [Huang et al., 2014]

Fixed-size method combined with stochastic learning (1)

- Pegasos (Primal estimated subgradient solver for SVM) [Shalev-Shwartz et al., ICML 2007]:

  Objective function for SVM with hinge loss

  $$\frac{\lambda}{2} w^T w + \frac{1}{m} \sum_{(x,y) \in \mathcal{A}_t} L(w,(x,y))$$

  with hinge loss $L(w,(x,y)) = \max\{0, 1 - y \langle w, x \rangle\}$, $\mathcal{A}_t$ random subsample at iteration t and decision function $\hat{y} = \mathrm{sign}[\langle w, x \rangle]$

- Combination with fixed-size method [Jumutc et al., IJCNN 2013]

Fixed-size method combined with stochastic learning (2)

Algorithm 1: Pegasos with hinge loss

```
   Data: $\mathcal{S}, \lambda, T, k, \epsilon$
1  Select $w_1$ randomly s.t. $\|w^{(1)}\| \leq 1/\sqrt{\lambda}$
2  for $t = 1 \to T$ do
3      Set $\eta_t = \frac{1}{\lambda t}$
4      Select $\mathcal{A}_t \subseteq \mathcal{S}$, where $|\mathcal{A}_t| = k$
5      $\rho = \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{A}_t} (y - \langle w_t, x\rangle), \forall i$
6      $\mathcal{A}_t^+ = \{(x,y) \in \mathcal{A}_t : y(\langle w_t, x\rangle + \rho) < 1\}, \forall i$
7      $w_{t+\frac{1}{2}} = w_t - \eta_t(\lambda w_t - \frac{1}{k} \sum_{(x,y)\in\mathcal{A}_t^+} yx)$
8      $w_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|w_{t+\frac{1}{2}}\|}\right\} w_{t+\frac{1}{2}}$
9      if $\|w_{t+1} - w_t\| \leq \epsilon$ then
10         return $(w_{t+1}, \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{S}} (y - \langle w_t, x\rangle))$
11     end
12 end
13 return $(w_{T+1}, \frac{1}{|\mathcal{S}|} \sum_{(x,y)\in\mathcal{S}} (y - \langle w_t, x\rangle))$
```

Fixed-size method combined with stochastic learning (3)

Algorithm 2: Fixed-Size Pegasos with hinge loss

```
   input : training data $\mathcal{S}$, labeling $Y$, parameters $\lambda, T, k, \epsilon, m$
   output: mapping $\hat{\Phi}(x), \forall x \in \mathcal{S}$, SVM model given by $w$ and $\rho$
1  begin
2      $\mathcal{S}_r \leftarrow$ FindActiveSet($\mathcal{S}, m$);
3      $\hat{\Phi}(x) \leftarrow$ ComputeNystromApprox($\mathcal{S}_r$);
4      $X \leftarrow [\hat{\Phi}(x_1)^T, \ldots, \hat{\Phi}(x_n)^T]$;
5      $[w, \rho] \leftarrow$ PegasosHL($X, Y, \lambda, T, k, \epsilon$);
6  end
```

[Jumutc, Huang, Suykens, IJCNN 2013]

Random Fourier Features

- Proposed by [Rahimi & Recht, 2007].

- It requires a positive definite shift-invariant kernel $K(x, y) = K(x - y)$. One obtains a randomized feature map $z(x) : \mathbb{R}^d \to \mathbb{R}^{2D}$ so that

$$z(x)^T z(y) \simeq K(x - y).$$

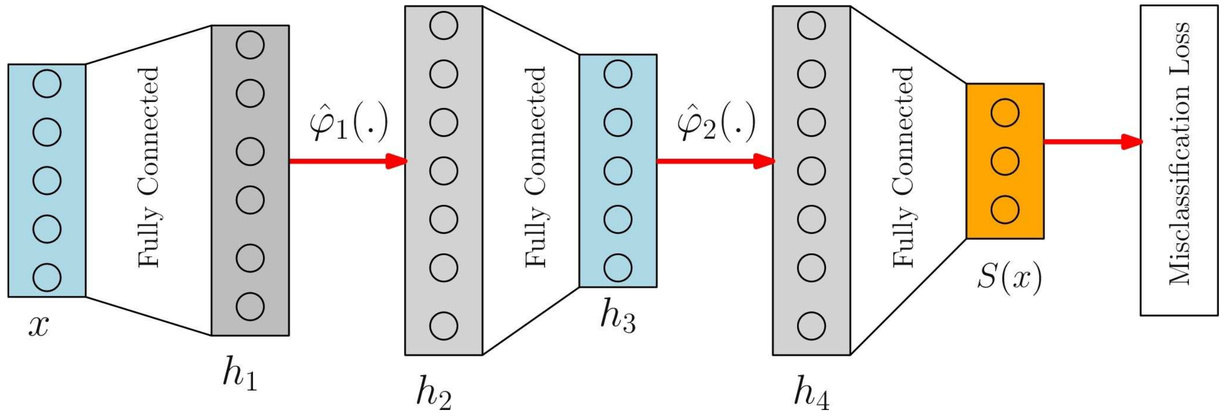
- Compute the Fourier transform p of the kernel K:

$$p(\omega) = \frac{1}{2\pi} \int \exp(-j\omega^T \Delta) K(\Delta) d\Delta$$

Draw D iid samples $\omega_1, ..., \omega_D \in \mathbb{R}^d$ from p.
Obtain $z(x) = \sqrt{\frac{1}{D}}[\cos(\omega_1^T x) ... \cos(\omega_D^T x) \sin(\omega_1^T x) ... \sin(\omega_D^T x)]^T$.

Deep neural-kernel networks using random Fourier features

Use of Random Fourier Features [Rahimi & Recht, NIPS 2007] to obtain an approximation to the feature map in a deep architecture



[Mehrkanoon & Suykens, Neurocomputing 2018]

Kernel PCA and kernel spectral clustering

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]

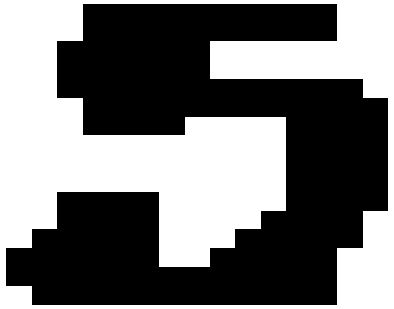
$$\Omega_c \alpha = \lambda \alpha \ \text{ with } \ \lambda = 1/\gamma$$

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

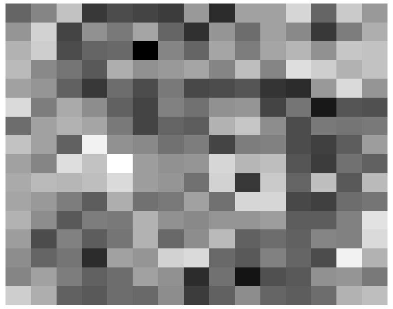
- Robust and sparse versions [Alzate & Suykens, 2008]: by taking other loss functions

Robustness: Kernel Component Analysis

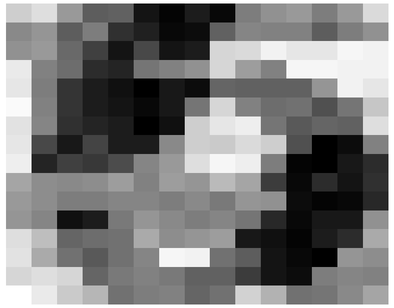
original image



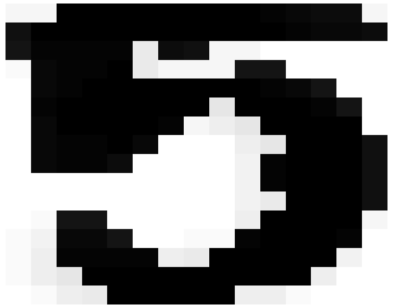
corrupted image



KPCA reconstruction



KCA reconstruction



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

Kernel Spectral Clustering (KSC)

- **Primal problem:** training on given data $\{x_i\}_{i=1}^N$

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

  with weighting matrix V and $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^h$ the feature map.

- **Dual:**

$$V M_V \Omega \alpha = \lambda \alpha$$

  with $\lambda = 1/\gamma$, $M_V = I_N - \frac{1}{1_N^T V 1_N} 1_N 1_N^T V$ weighted centering matrix, $\Omega = [\Omega_{ij}]$ kernel matrix with $\Omega_{ij} = \varphi(x_i)^T \varphi(x_j) = K(x_i, x_j)$

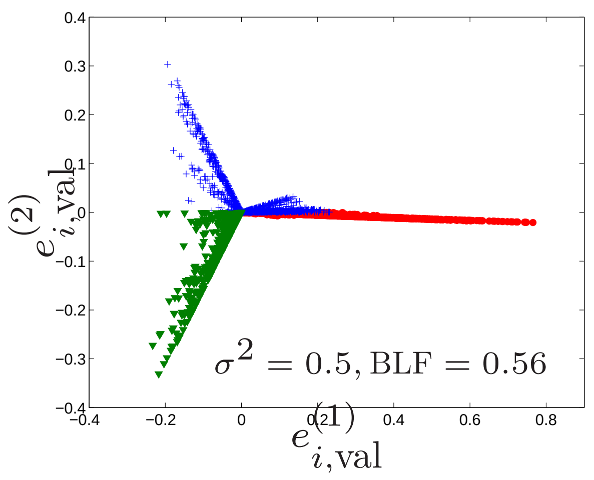
- Taking $V = D^{-1}$ with degree matrix $D = \text{diag}\{d_i\}$, $d_i = \sum_{j=1}^N \Omega_{ij}$ relates to random walks algorithm [Chung, 1997; Shi & Malik, 2000; Ng 2002]

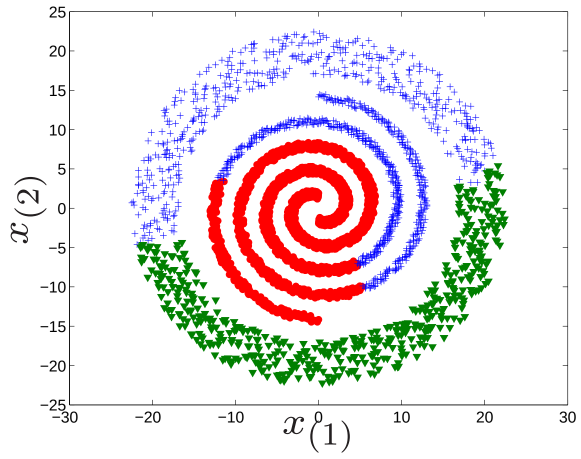
[Alzate & Suykens, IEEE-PAMI, 2010]

Advantages of kernel-based setting

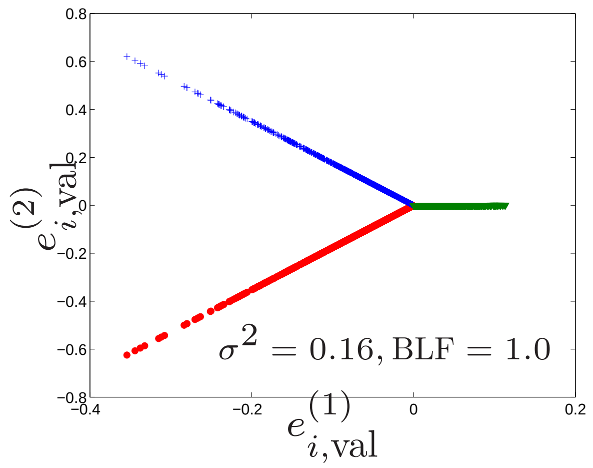
- **model-based** approach
- **out-of-sample extensions**, applying model to new data
- consider **training, validation and test data**
  (training problem corresponds to eigenvalue decomposition problem)
- model selection procedures
- **sparse representations and large scale methods**

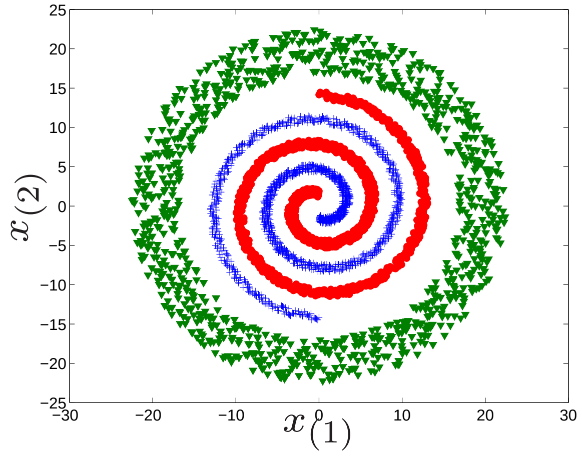
Model selection: toy example





BAD



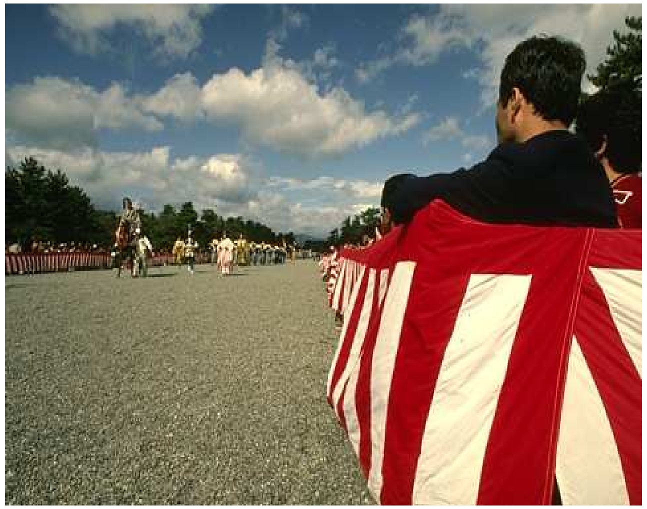


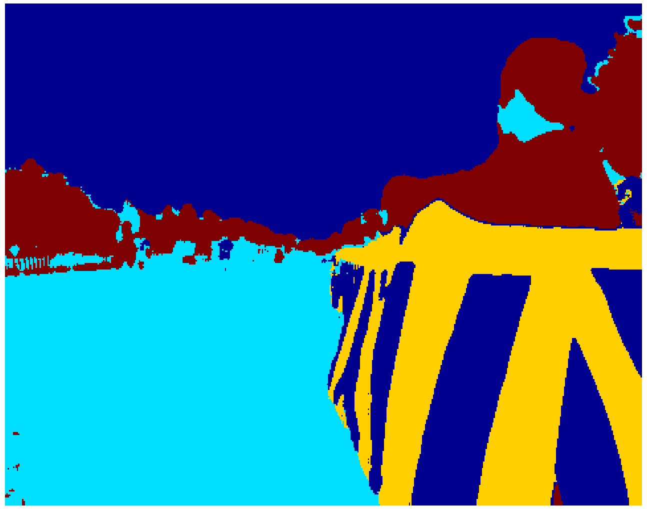
GOOD

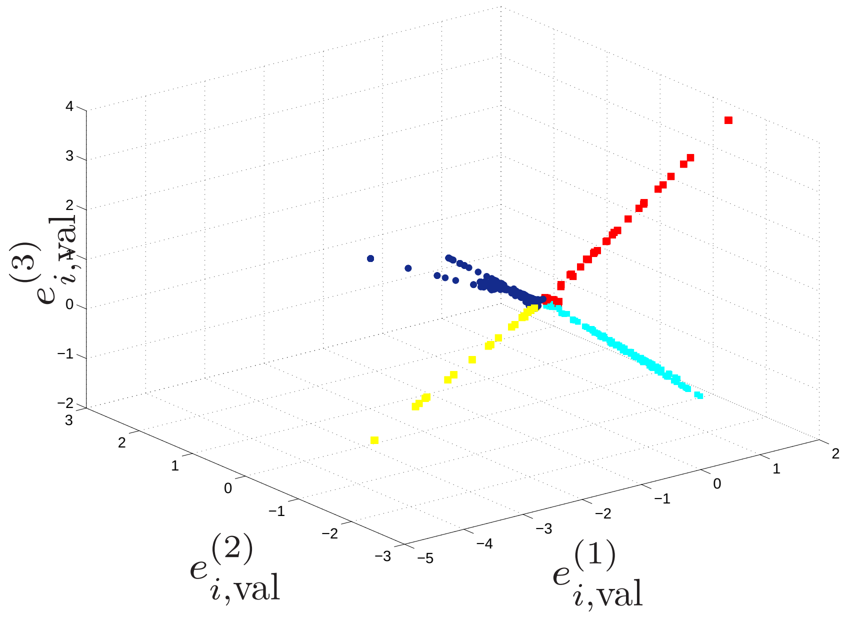
validation set

train + validation + test data

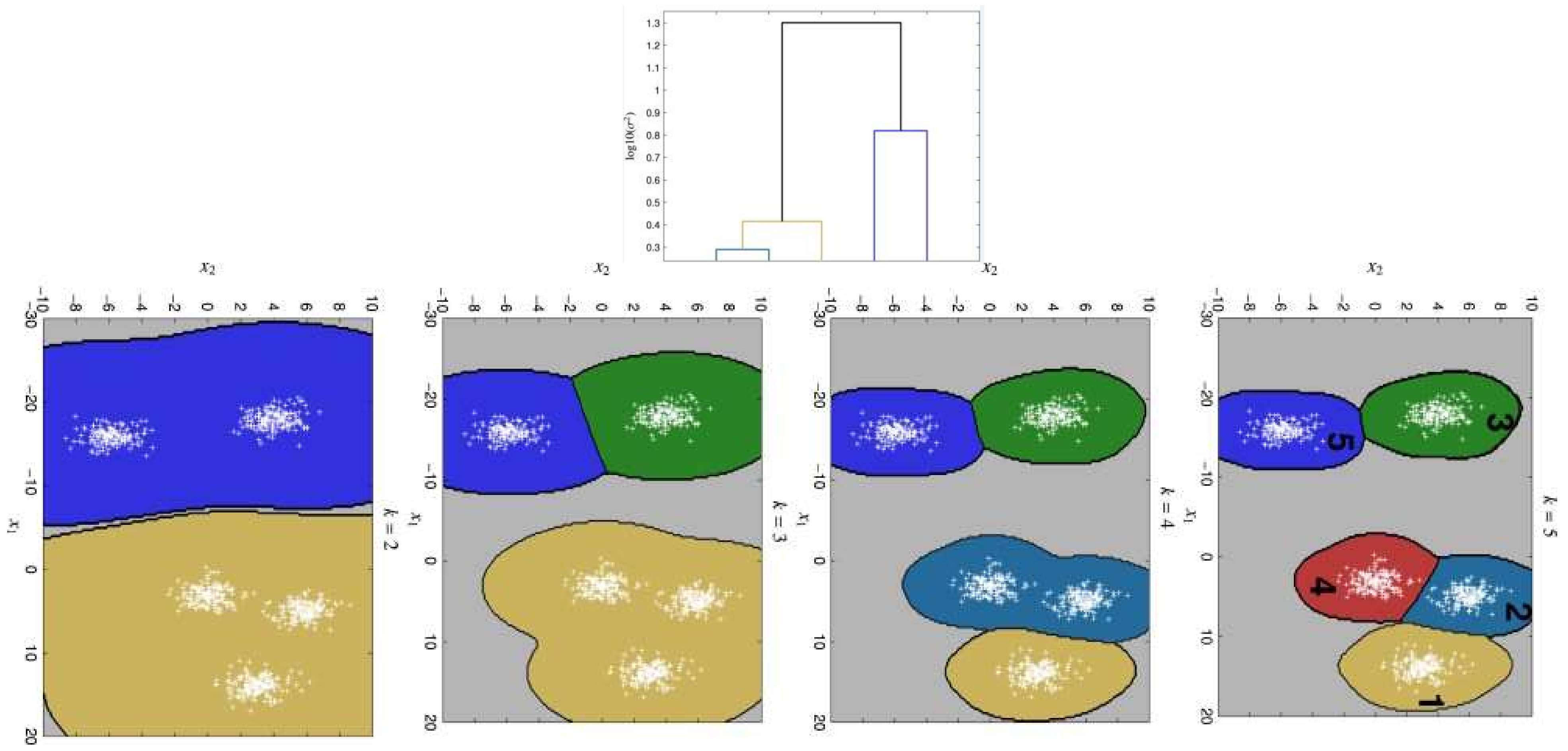
Example: image segmentation







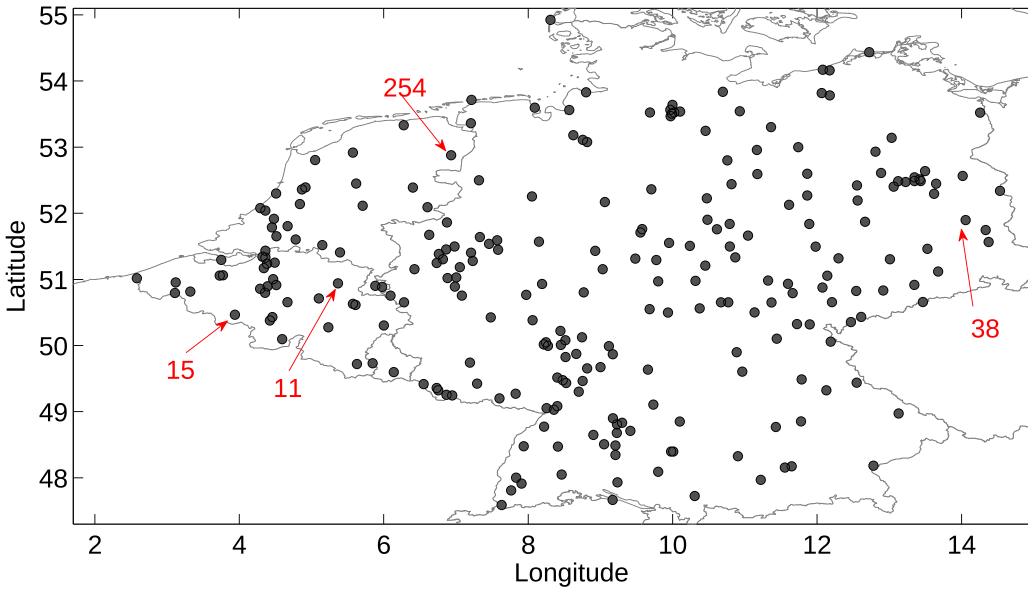
Hierarchical KSC

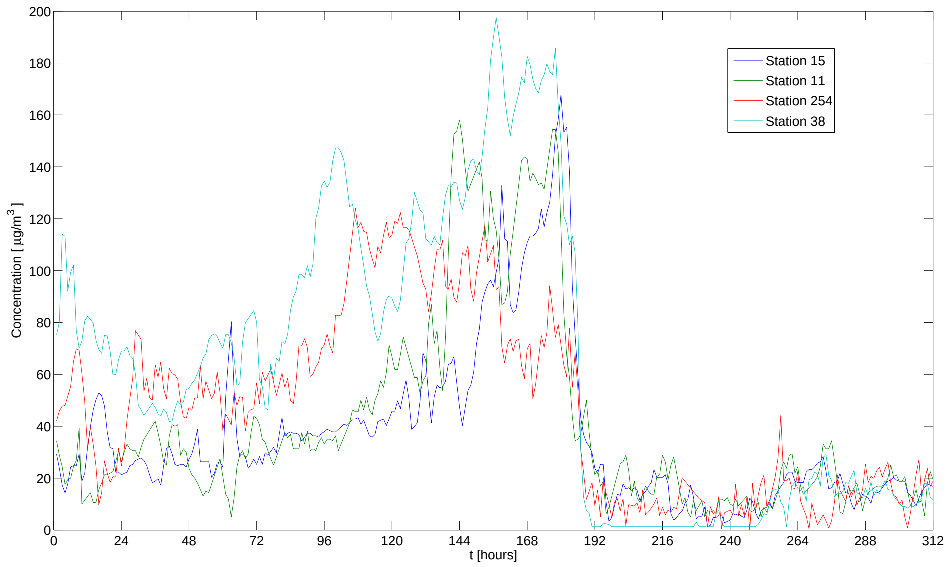


[Alzate & Suykens, 2012]

Dynamic clustering of PM10 concentrations (1)

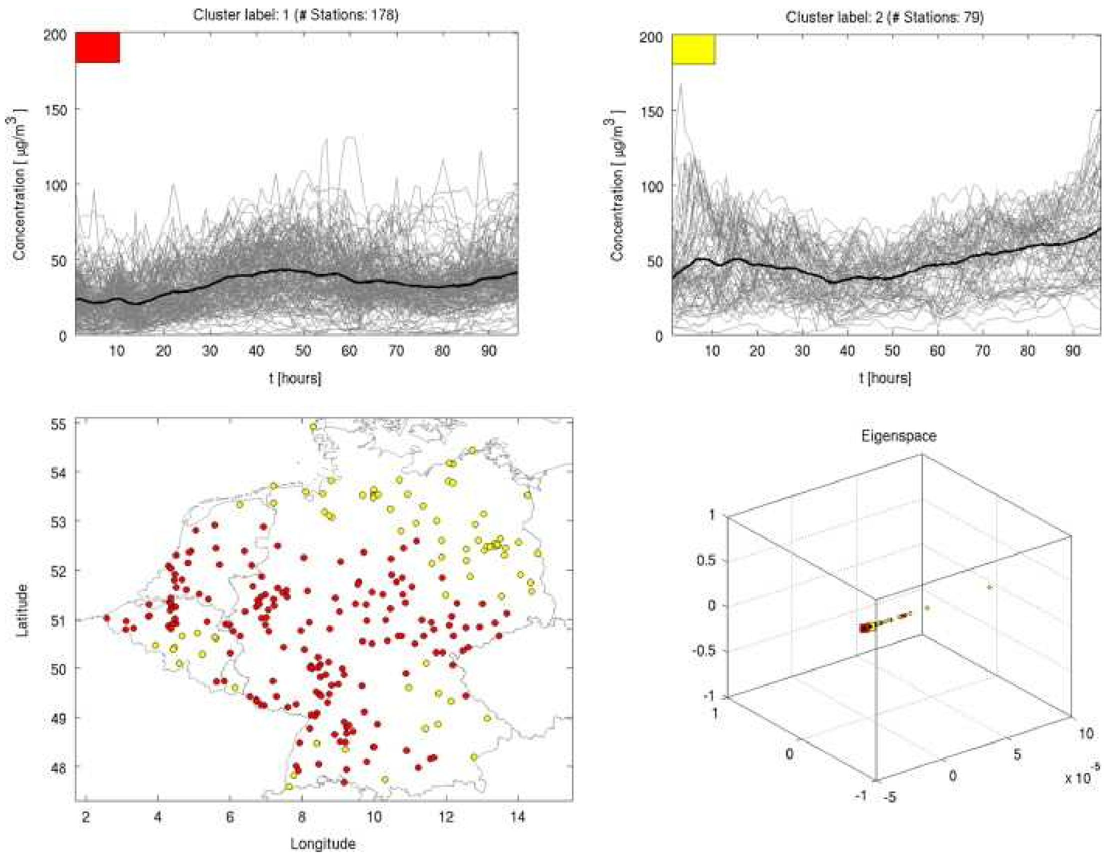
PM10 time-series: PM10 data (Particulate Matter) registered during a heavy pollution episode (Jan 20 2010 - Feb 1 2010) in Europe.





[Kernel spectral clustering [Langone, Agudelo, De Moor, Suykens, Neurocomputing, 2014]

Example: dynamic clustering of PM10 concentrations (2)



video - [Langone, Agudelo, De Moor, Suykens, Neurocomputing, 2014]

Big data networks

KSC for big data networks (1)

- **YouTube Network:** YouTube social network where users form friendship with each other and the users can create groups where others can join.
- **RoadCA Network:** a road network of California. Intersections and endpoints are represented by nodes and the roads connecting these intersections are represented as edges.
- **Livejournal Network**: free online social network where users are bloggers and they declare friendship among themselves.

Dataset	Nodes	Edges
YouTube	1,134,890	2,987,624
roadCA	1,965,206	5,533,214
Livejournal	3,997,962	34,681,189

[Mall, Langone, Suykens, Entropy, special issue Big data, 2013]

KSC for big data networks (2)

BAF-KSC:

- Select representative training subgraph using FURS
  (FURS = Fast and Unique Representative Subset selection [Mall et al., 2013]: selects nodes with high degree centrality belonging to different dense regions, using a deactivation and activation procedure)

- Perform model selection using BAF
  (BAF = Balanced Angular Fit: makes use of a cosine similarity measure related to the e-projection values on validation nodes)

- Train the KSC model by solving a small eigenvalue problem of size $\min(0.15N, 5000)^2$

- Apply out-of-sample extension to find cluster memberships of the remaining nodes

[Mall, Langone, Suykens, Entropy, special issue Big data, 2013]

KSC for big data networks (3)

BAF-KSC	[Mall, Langone, Suykens, 2013]	**Louvain**	[Blondel et al., 2008]
Infomap	[Lancichinetti, Fortunato, 2009]	**CNM**	[Clauset, Newman, Moore, 2004]

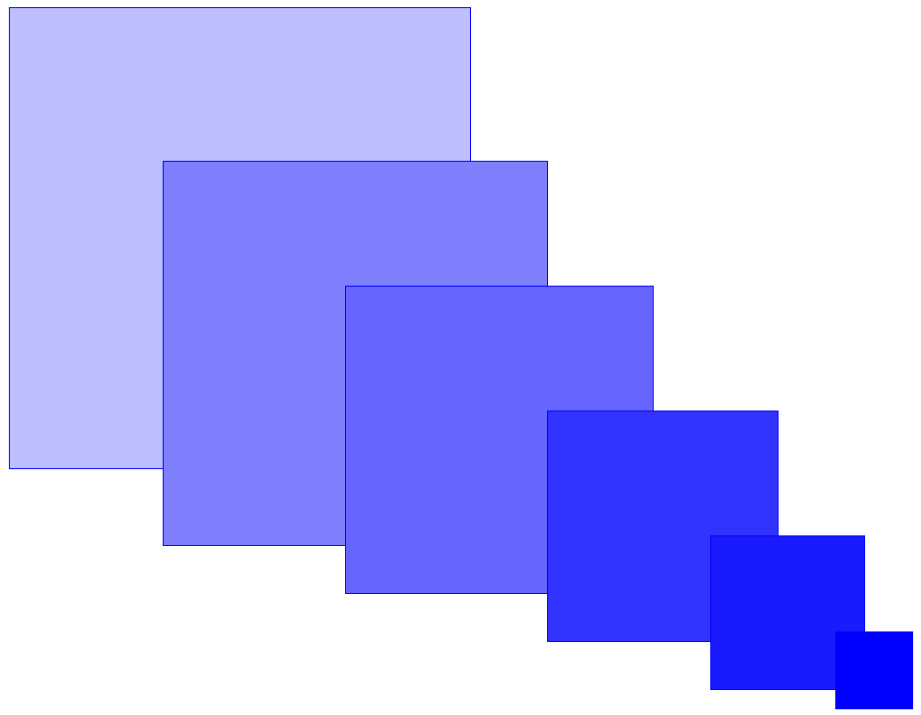
Dataset	BAF-KSC			Louvain			Infomap			CNM		
	Cl	Q	Con	Cl	Q	Con	Cl	Q	Con	Cl	Q	Con
Openflight	5	0.533	**0.002**	109	**0.61**	0.02	18	0.58	0.005	84	0.60	0.016
PGPnet	8	0.58	**0.002**	105	**0.88**	0.045	84	0.87	0.03	193	0.85	0.041
Metabolic	10	0.22	0.028	10	**0.43**	0.03	41	0.41	0.05	11	0.42	0.021
HepTh	6	0.45	**0.0004**	172	**0.65**	0.004	171	0.3	0.004	6	0.423	0.0004
HepPh	5	0.56	0.0004	82	**0.72**	0.007	69	0.62	0.06	6	0.48	0.0007
Enron	10	0.4	0.002	1272	**0.62**	0.05	1099	0.37	0.27	6	0.25	0.0045
Epinion	10	**0.22**	0.0003	33	0.006	0.0003	17	0.18	0.0002	10	0.14	**0.0**
Condmat	6	0.28	**0.0002**	1030	**0.79**	0.03	1086	0.79	0.025	8	0.38	0.0003

Flight network (Openflights), network based on trust (PGPnet), biological network (Metabolic), citation networks (HepTh, HepPh), communication network (Enron), review based network (Epinion), collaboration network (Condmat) [snap.stanford.edu]

Cl = Clusters, Q = modularity, Con = Conductance

BAF-KSC usually finds a smaller number of clusters and achieves lower conductance

Multilevel Hierarchical KSC for complex networks (1)



Generating a series of affinity matrices over different levels:
communities at level h become nodes for next level $h+1$

Multilevel Hierarchical KSC for complex networks (2)

- Start at ground level 0 and compute cosine similarities on validation nodes $S_{val}^{(0)}(i,j) = 1 - e_i^T e_j / (\|e_i\| \|e_j\|)$
- Select projections e_j satisfying $S_{val}^{(0)}(i,j) < t^{(0)}$ with $t^{(0)}$ a threshold value. Keep these nodes as the first cluster at level 0 and remove the nodes from matrix $S_{val}^{(0)}$ to obtain a reduced matrix.
- **Iterative procedure over different levels h:**
  Communities at level h become nodes for the next level $h+1$, by creating a set of matrices at different levels of hierarchy:

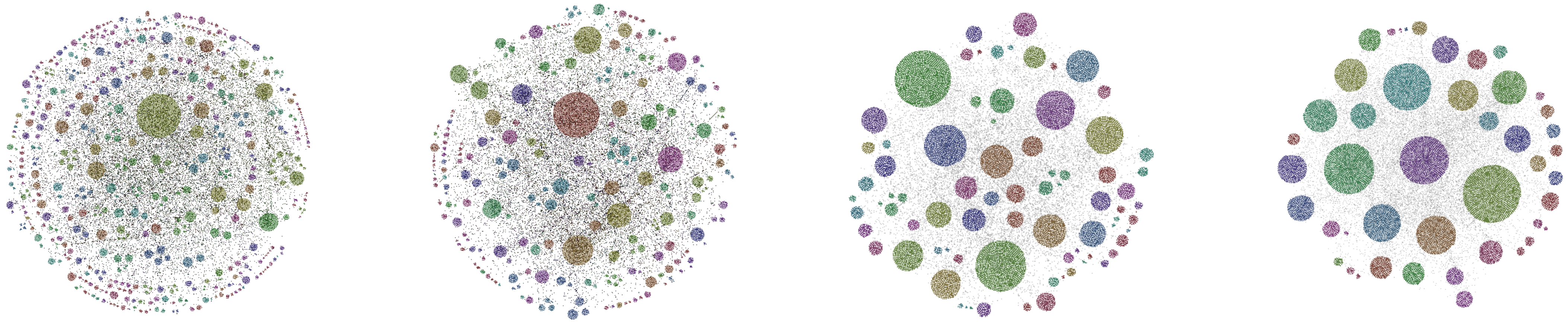
$$S_{val}^{(h)}(i,j) = \frac{1}{|C_i^{(h-1)}| \, |C_j^{(h-1)}|} \sum_{m \in C_i^{(h-1)}} \sum_{l \in C_j^{(h-1)}} S_{val}^{(h-1)}(m,l)$$

  and working with suitable threshold values $t^{(h)}$ for each level.

[Mall, Langone, Suykens, PLOS ONE, 2014]

Multilevel Hierarchical KSC for complex networks (3)

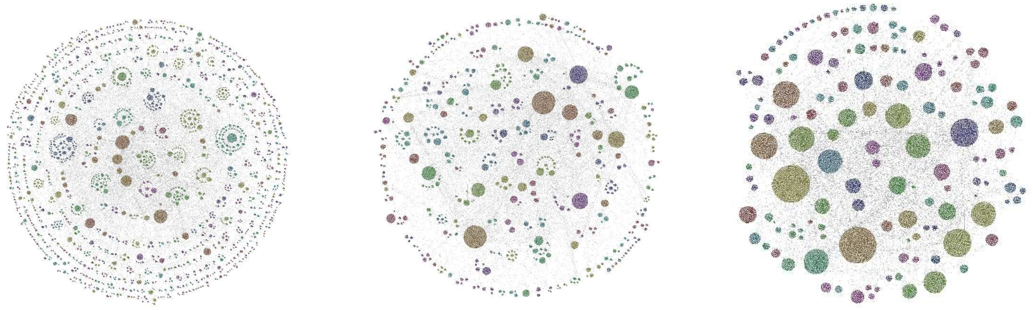
MH-KSC on PGP network:



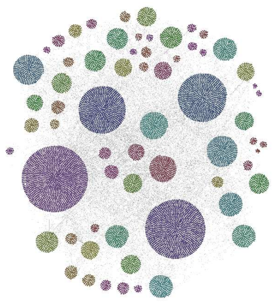
Multilevel Hierarchical KSC finds high quality clusters at coarse as well as fine and intermediate levels of hierarchy.

[Mall, Langone, Suykens, PLOS ONE, 2014]

Multilevel Hierarchical KSC for complex networks (4)



Louvain



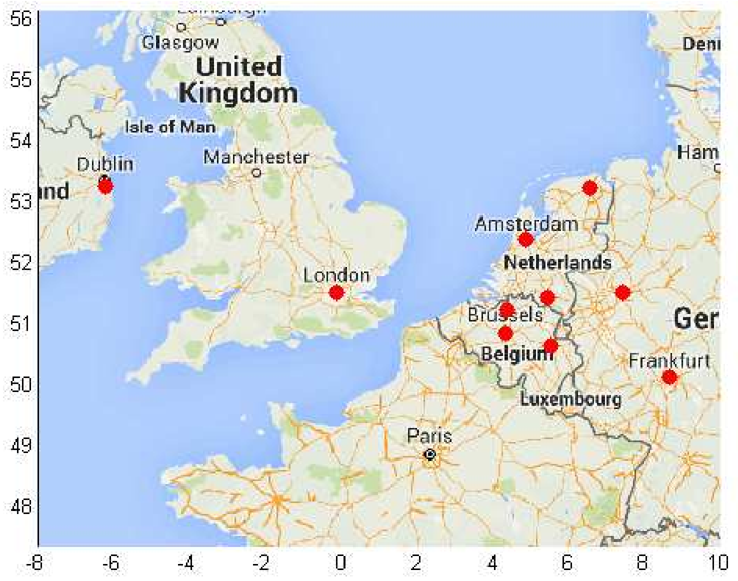
Infomap

OSLOM

Louvain, Infomap, and OSLOM seem biased toward a particular scale in comparison with MH-KSC, based upon ARI, VI, Q metrics

Application study: black-box weather forecasting within primal-dual framework

Black-box weather forecasting



- Black-box weather forecasting: prediction temperature in Brussels

- Multi-view learning:
  - Multi-view LS-SVM regression [Houthuys, Karevan, Suykens, IJCNN 2017]
  - Multi-view Deep Neural Networks [Karevan, Houthuys, Suykens, ICANN 2018]

Multi-view learning: kernel-based (1)

- Primal problem:

$$\begin{array}{ll}\min\limits_{w^{[v]},e^{[v]}} & \frac{1}{2}\sum_{v=1}^{V} w^{[v]^T} w^{[v]} + \frac{1}{2}\sum_{v=1}^{V} \gamma^{[v]} e^{[v]^T} e^{[v]} + \rho \sum_{v,u=1;v\neq u}^{V} e^{[v]^T} e^{[u]} \\ \text{subject to} & y = \Phi^{[v]} w^{[v]} + b^{[v]} 1_N + e^{[v]}, \;\; v = 1, ..., V\end{array}$$

- Dual:

$$\left[\begin{array}{c|c} 0_{V\times V} & 1_M^T \\ \hline \Gamma_M 1_M + \rho\, \mathcal{I}_M 1_M & \Gamma_M \Omega_M + \mathbb{I}_{NV} + \rho\, \mathcal{I}_M \Omega_M \end{array}\right] \left[\begin{array}{c} b_M \\ \hline \alpha_M \end{array}\right] = \left[\begin{array}{c} 0_V \\ \hline \Gamma_M y_M + (V-1)\rho\, y_M \end{array}\right]$$

- Prediction:

$$\hat{y}(\mathbf{x}) = \sum_{v=1}^{V} \beta_v \sum_{k=1}^{N} \alpha_k^{[v]} K^{[v]}(\mathbf{x}^{[v]}, \mathbf{x}_k^{[v]}) + b^{[v]}$$

[Houthuys et al., 2017]

Multi-view learning: kernel-based (2)

- Data set:
  - Real measurements for weather elements such as temperature, humidity, etc.
  - From 2007 until mid 2014
  - Two test sets:
    - mid-November 2013 until mid-December 2013
    - from mid-April 2014 to mid-May 2014

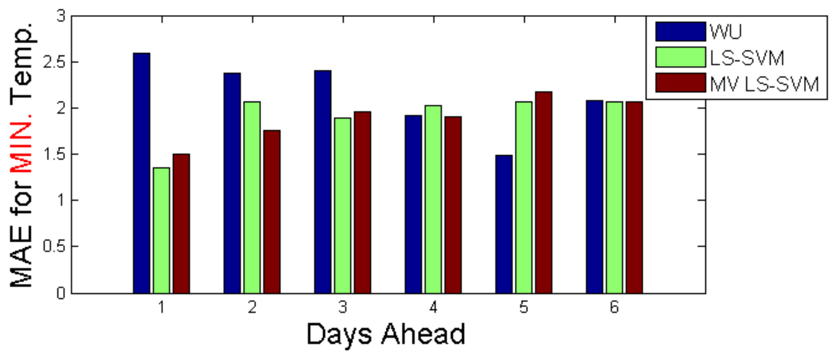
- **Goal:** Forecasting minimum and maximum temperature for one to six days ahead in Brussels Belgium

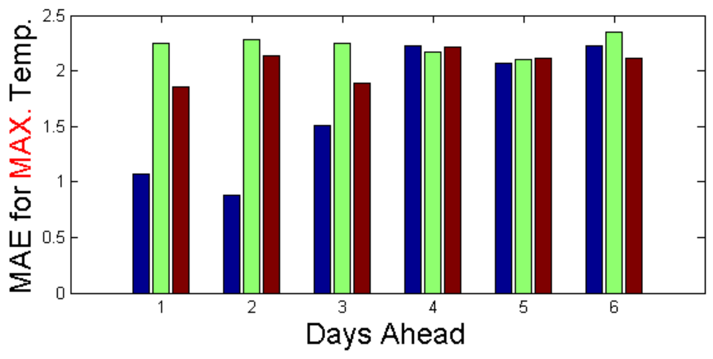
- **Views:** Brussels together with 9 neighboring cities

- Tuning parameters:
  - kernel parameters for each view
  - regularization parameters $\gamma^{[v]}$
  - coupling parameter ρ

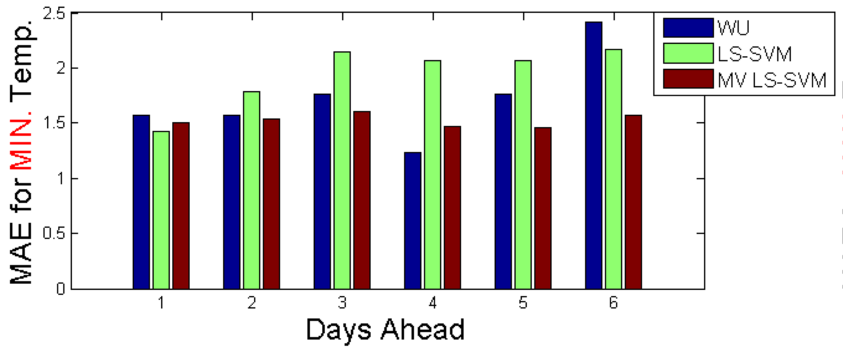
Multi-view learning: kernel-based (3)

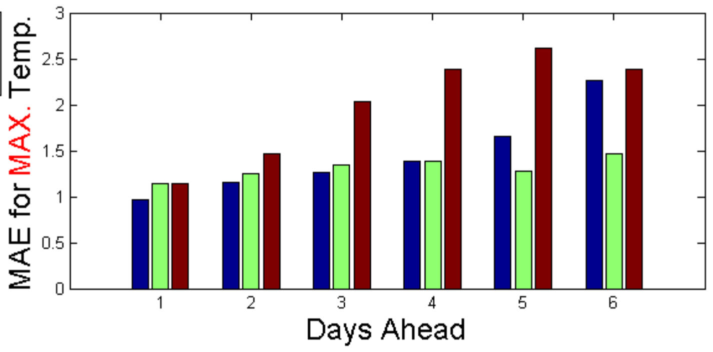
Apr/May





Nov/Dec





Multi-view learning: deep neural network (1)

- Primal formulation of multi-view LS-SVM:

$$\begin{array}{ll} \min\limits_{\mathbf{w}^{[v]},\mathbf{e}^{[v]},b^{[v]}} & \frac{1}{2}\sum_{v=1}^{V} {\mathbf{w}^{[v]}}^T \mathbf{w}^{[v]} + \frac{1}{2}\sum_{v=1}^{V} \gamma^{[v]} {\mathbf{e}^{[v]}}^T \mathbf{e}^{[v]} + \rho \sum_{v,u=1;v\neq u}^{V} {\mathbf{e}^{[v]}}^T \mathbf{e}^{[u]} \\ \text{subject to} & \mathbf{y} = \mathbf{\Phi}^{[v]}\mathbf{w}^{[v]} + b^{[v]}\mathbf{1}_N + \mathbf{e}^{[v]} \text{ for } v = 1, ..., V \end{array}$$

- Weighted Multi-view approach:

$$\min_{\mathbf{w}^{[v]},\mathbf{e}^{[v]}} \frac{1}{2}\sum_{v=1}^{V} s^{[v]} \left({\mathbf{w}^{[v]}}^T \mathbf{w}^{[v]} + \gamma^{[v]} {\mathbf{e}^{[v]}}^T \mathbf{e}^{[v]} \right) + \sum_{v,u=1;v\neq u}^{V} \rho^{[v,u]} \sqrt{s^{[v]}}\sqrt{s^{[u]}}\, {\mathbf{e}^{[v]}}^T \mathbf{e}^{[u]}$$

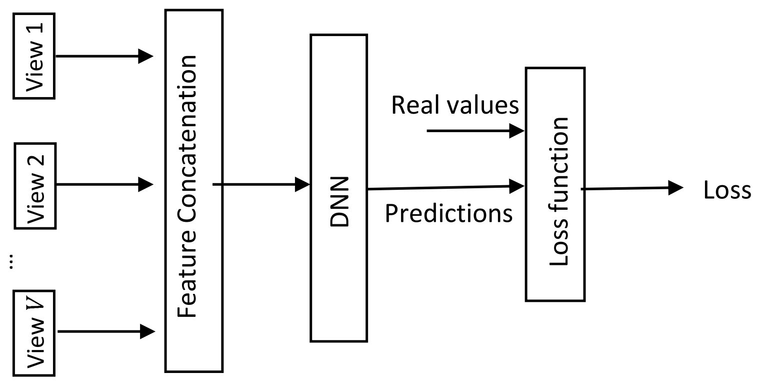
  – $s^{[v]}$: weight of the view v (can be manually determined by an expert, or calculated during a pre-processing step)
  – $\rho^{[v,u]}$: coupling parameter for pairwise combination of views
  – $0 \leq \rho^{[v,u]} \leq \min\{\gamma^{[v]}, \gamma^{[u]}\}$

[Karevan et al., 2018]

Multi-view learning: deep neural network (2)

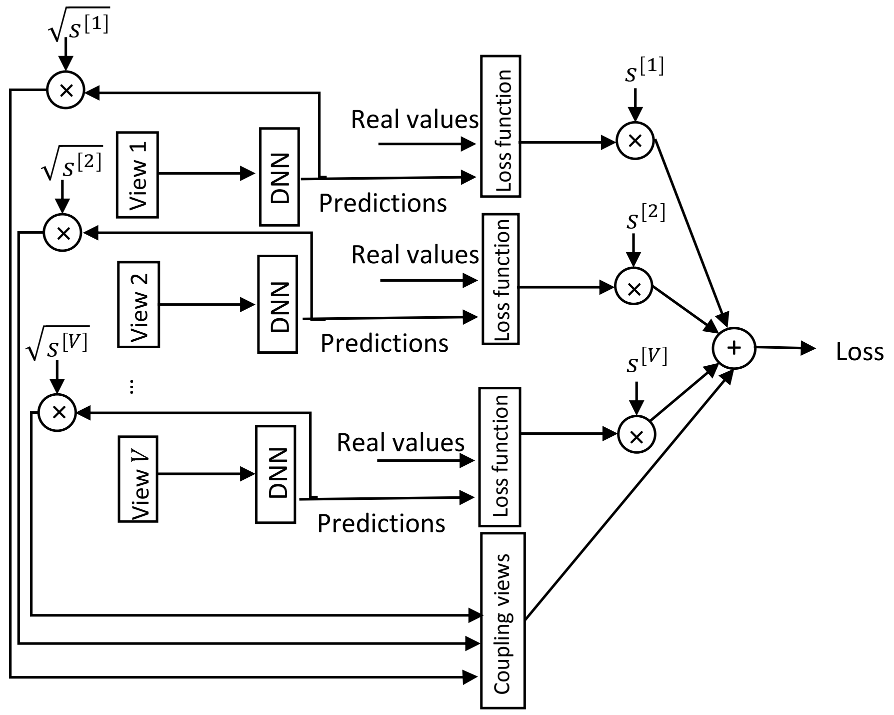
Early fusion

(Unweighted MV-DNN version)



Late fusion

(Weighted MV-DNN version)



Multi-view learning: deep neural network (3)

- Weather forecasting is a time series prediction problem $\rightarrow$ Consider each delay as a view
- Consider 5 views (i.e. the delay is considered to be 5)
- Tuning parameters: regularization parameter $\gamma^{[v]}$ and number of neurons for each view, and $\rho^{[v,u]}$ coupling parameter for each part of views
- The weight of each view is defined based on its error the validation set: $s^{[v]} = \exp(-mse_{val}^{[v]})$
- Forecasting minimum and maximum temperature for one to six days ahead in Brussels, Belgium

Multi-view learning: deep neural network (4)

Median MAE of the predictions in Unweighted MV-DNN and Weighted MV-DNN with two hidden layers on Apr/May test set

Step ahead	Temp.	Unweighted MV-DNN	Weighted MV-DNN *(average)*	Weighted MV-DNN *(weighted average)*
1	Min	1.89±0.002	2.01±0.0001	**1.67±0.01**
	Max	2.41±0.01	2.52±0.01	**2.11±0.009**
2	Min	2.22±0.004	2.17±0.03	**1.96±0.02**
	Max	2.53±0.02	2.49±0.03	**2.47±0.02**
3	Min	2.29±0.01	2.28±0.002	**2.06±0.01**
	Max	2.66±0.01	2.60±0.006	**2.42±0.0008**
4	Min	2.28±0.006	2.26±0.02	**2.17±0.01**
	Max	2.65±0.005	2.74±0.03	**2.56±0.01**
5	Min	**2.46±0.0006**	**2.46±0.0002**	2.49±0.004
	Max	2.78±0.01	2.83±0.03	**2.73±0.02**
6	Min	2.64±0.002	2.60±0.01	**2.59±0.002**
	Max	2.85 ±0.01	2.87±0.03	**2.75±0.03**

Conclusions

- **Function estimation:** parametric versus kernel-based
- Towards **unifying framework** for data-driven modelling
- **Synergies** between neural networks and kernel based-modelling
- **Extensions** to indefinite kernels, tensor kernels and deep learning
- Sparse kernel models using **fixed-size method**
- Applications in **supervised and unsupervised learning**
- Large scale **complex networks and big data**

Acknowledgements (1)

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

  C. Alzate, Y. Chen, J. De Brabanter, K. De Brabanter, 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, R. Mall, S. Mehrkanoon, G. Nisol, M. Orchel, A. Pandey, K. Pelckmans, S. RoyChowdhury, S. Salzo, J. Schreurs, M. Signoretto, Q. Tao, 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, iMinds, BIL, COST

Acknowledgements (2)

erc

KATHOLIEKE UNIVERSITEIT
LEUVEN

FWO
VLAANDEREN

Acknowledgements (3)

NEW: ERC Advanced Grant E-DUALITY

Exploring duality for future data-driven modelling

Thank you