Imperial College London

iBug Tutorial: Multiple Kernel Learning for Regression and Classification

Sebastian Kaltwang

Kernel Methods: Overview

Learn:

Features | Function | Prediction
$$\boxed{\mathbf{x}} \xrightarrow{y(\mathbf{x};\mathbf{w})} y$$

Given:

Data: Features $\mathbf{x}_n$ Targets t_n

Loss $\boxed{L(y,t)}$

Regularizer $\boxed{R(\mathbf{w})}$

Objective:

Minimize $\sum_n \boxed{L(y(\mathbf{x}_n;\mathbf{w}),t_n)} + \boxed{R(\mathbf{w})}$

Kernel Methods Example: SVM and RVM

Features

x $\in \mathbb{R}^d$

Function

y(**x**;**w**) $= \sum_m w_m \boxed{\kappa(\mathbf{x}, \mathbf{x}_m)} + b$

Kernel function

Target

t

Loss

L(y,t)

Regularizer

R(**w**)

Objective

Classification SVM

$$t \in \{-1, 1\}$$

Hinge-Loss

$$\max(0, 1 - t_n y_n)$$

Kernel-induced Norm

$$||y||^2_{\mathcal{H}}$$

$$\min_{y,b,\xi} \quad \frac{1}{2}||y||^2_{\mathcal{H}} + C \sum_n \xi_n$$

$$s.t. \quad t_n y_n \geq 1 - \xi_n \quad \forall n$$

$$\xi_n \geq 0 \quad \forall n$$

Regression RVM

$$t \in \mathbb{R}$$

i.i.d. Gaussian

$$p(t_n|\mathbf{w}, \sigma^2) \sim \mathcal{N}(y_n, \sigma^2)$$

Hierarchical Gaussian

$$p(\mathbf{w}|\boldsymbol{\alpha}) \sim \mathcal{N}(0, \mathrm{diag}(\boldsymbol{\alpha})^{-1})$$

$$\max_{\boldsymbol{\alpha}, \sigma^2} \int \prod_n p(t_n|\mathbf{w}, \sigma^2) p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w}$$

Imperial College London

Kernel Types

Function $\xrightarrow{y(\mathbf{x};\mathbf{w})}$ $= \sum_m w_m \kappa(\mathbf{x}, \mathbf{x}_m) + b$

$\kappa(\mathbf{x}, \mathbf{x}')$ defines a dot product in the corresponding RKHS $\mathcal{H}$

Type	$\kappa(\mathbf{x}, \mathbf{x}')$
Linear	$\mathbf{x}^\top \mathbf{x}'$
Polynomial	$(a\mathbf{x}^\top \mathbf{x}')^b$
Gaussian	$\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{x}')\right)$ special cases: $\boldsymbol{\Sigma}^{-1} = \sigma^{-2}\mathbf{I}$, $\boldsymbol{\Sigma}^{-1} = diag(\boldsymbol{\sigma})^{-2}$
Intersection	$\sum_d \min(\mathbf{x}(d), \mathbf{x}'(d))$

Combining Kernels

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= c k_1(\mathbf{x}, \mathbf{x}') && (6.13)\\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') && (6.14)\\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.15)\\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.16)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') && (6.17)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') && (6.18)\\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) && (6.19)\\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' && (6.20)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}_a') + k_b(\mathbf{x}_b, \mathbf{x}_b') && (6.21)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}_a') k_b(\mathbf{x}_b, \mathbf{x}_b') && (6.22)
\end{aligned}
$$

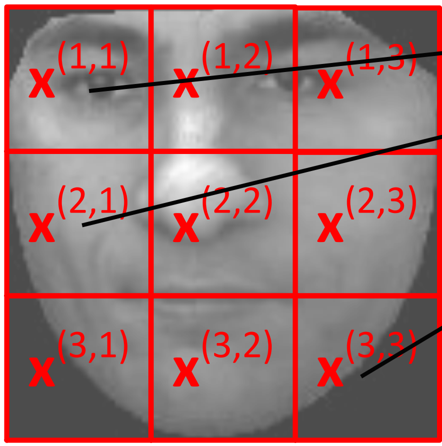
Kernel that combines different features $\mathbf{x}_a$ and $\mathbf{x}_b$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

(from C. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.)

Examples for Structuring Data using Kernels

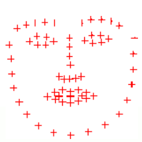
Structure according **spatial regions**



$$\kappa_1 = \kappa(\mathbf{x}^{(1,1)}, \mathbf{x}'^{(1,1)})$$
$$\kappa_2 = \kappa(\mathbf{x}^{(2,1)}, \mathbf{x}'^{(2,1)})$$
$$\vdots$$
$$\kappa_9 = \kappa(\mathbf{x}^{(3,3)}, \mathbf{x}'^{(3,3)})$$

Structure according **feature types**

$$\kappa_1 = \kappa(\mathbf{x}^{(\text{appearance})}, \mathbf{x}'^{(\text{appearance})})$$



$$\kappa_2 = \kappa(\mathbf{x}^{(\text{shape})}, \mathbf{x}'^{(\text{shape})})$$

$$\kappa_3 = \kappa(\mathbf{x}^{(\text{audio})}, \mathbf{x}'^{(\text{audio})})$$

Combining Kernels

Weighted sums and products of kernels are kernels:

For K kernels κ_k and weights $v_k \geq 0$

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_k v_k \kappa_k(\mathbf{x}, \mathbf{x}')$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \prod_k \left(\kappa_k(\mathbf{x}, \mathbf{x}')\right)^{v_k}$$

The MKL Problem:

Learn the combination weights v_k

(additional to the base learner, e.g. SVM or RVM)

Short Overview of selected MKL Methods

MKL History: Minimize Validation Error

[1] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, “Choosing multiple parameters for support vector machines,” Mach. Learn., vol. 46, no. 1–3, pp. 131–159, 2002

Repeat until local minimum is reached:

(1) Solve **w** given fixed **v** by **original SVM**

(2) Minimize the **estimated validation error** w.r.t. **v** with a gradient step

Function

$\longrightarrow y(\mathbf{x};\mathbf{w},\mathbf{v}) \longrightarrow$

$$= \sum_m w_m \kappa(\mathbf{x}, \mathbf{x}_m; \mathbf{v}) + b$$

MKL History: Boosting

[2] K. P. Bennett, M. Momma, and M. J. Embrechts, “MARK: A boosting algorithm for heterogeneous kernel models,” in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp. 24–31.

- **Boosting** of the kernel-columns by using **ridge regression** as base
- Each of the kernel-columns $\kappa_k(\mathbf{x}, \mathbf{x}_m)$ is taken as hypothesis
- Optimizing of $w_{m,k}$ by **coordinate descent**

$$\xrightarrow{\text{Function}} y(\mathbf{x};\mathbf{w}) \longrightarrow \; = \sum_k \sum_m w_{m,k} \kappa_k(\mathbf{x}, \mathbf{x}_m) + b$$

Imperial College
London

MKL History: Convex Formulation

- First formulation of linear MKL as **convex problem** with **convergence guarantees**
- Many following papers solve the same objective with different methods

[3] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, “Learning the kernel matrix with semidefinite programming,” J. Mach. Learn. Res., vol. 5, pp. 27–72, 2004.

[4] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, “Large Scale Multiple Kernel Learning,” J. Mach. Learn. Res., vol. 7, pp. 1531–1565, 2006.

[5] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, “SimpleMKL,” J. Mach. Learn. Res., vol. 9, pp. 2491–2521, 2008.

Function

$$\xrightarrow{\; y(\mathbf{x};\mathbf{w},\mathbf{v}) \;} = \sum_m \sum_k w_m v_k \kappa_k(\mathbf{x}, \mathbf{x}_m) + b$$

Imperial College London

MKL Example: SimpleMKL and DSRVM

Function

—y(**x**;**w**,**v**)→ $= \sum_m \sum_k w_m v_k \kappa_k(\mathbf{x}, \mathbf{x}_m) + b$

	SimpleMKL	DSRVM
Loss: L(y,t)	Hinge-Loss $\max(0, 1 - t_n y_n)$	i.i.d. Gaussian $p(t_n \mid \mathbf{w}, \sigma^2) \sim \mathcal{N}(y_n, \sigma^2)$
Regularizer: R(**w**)	Kernel-induced Norm $\lvert\lvert y \rvert\rvert_{\mathcal{H}}^2$	Hierarchical Gaussian $p(\mathbf{w} \mid \boldsymbol{\alpha}) \sim \mathcal{N}(0, \mathrm{diag}(\boldsymbol{\alpha})^{-1})$
R(**v**)	Convex Sum $\sum_k v_k = 1, \quad v_k \geq 0 \quad \forall k$	$p(\mathbf{v} \mid \boldsymbol{\beta}) \sim \mathcal{N}(0, \mathrm{diag}(\boldsymbol{\beta})^{-1})$
Objective	$\min_{y,b,\xi} \frac{1}{2} \lvert\lvert y \rvert\rvert_{\mathcal{H}}^2 + C \sum_n \xi_n$ $s.t. \quad t_n y_n \geq 1 - \xi_n \quad \forall n$ $\xi_n \geq 0 \quad \forall n$ $\sum_k v_k = 1, \quad v_k \geq 0 \quad \forall k$	$\max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma^2} \int \int \prod_n p(t_n \mid \mathbf{w}, \mathbf{v}, \sigma^2) p(\mathbf{w} \mid \boldsymbol{\alpha}) p(\mathbf{v} \mid \boldsymbol{\beta}) d\mathbf{w} d\mathbf{v}$

Practical differences between SVM/RVM and MKL

	SVM / RVM	MKL
Application domain	Same application domain	
#Kernels	single	K
Training input	$N \times 1$ target vector $\mathbf{t}$ $N \times N$ kernel gram matrix $\mathbf{G}$	$N \times 1$ target vector $\mathbf{t}$ $N \times N \times K$ kernel gram tensor $\mathbf{G}$
Testing input	$N_{\text{test}} \times N_{\text{SV}}$ gram matrix $(N_{\text{SV}} \leq N)$	$N_{\text{test}} \times N_{\text{SV}} \times K_{\text{active}}$ gram tensor $(N_{\text{SV}} \leq N, \quad K_{\text{active}} \leq K)$
Kernel parameters	Commonly set by cross-validation	Cross-validation usually not possible → resort to heuristic or optimizing kernels separately

N: # of training samples
K: # of kernels

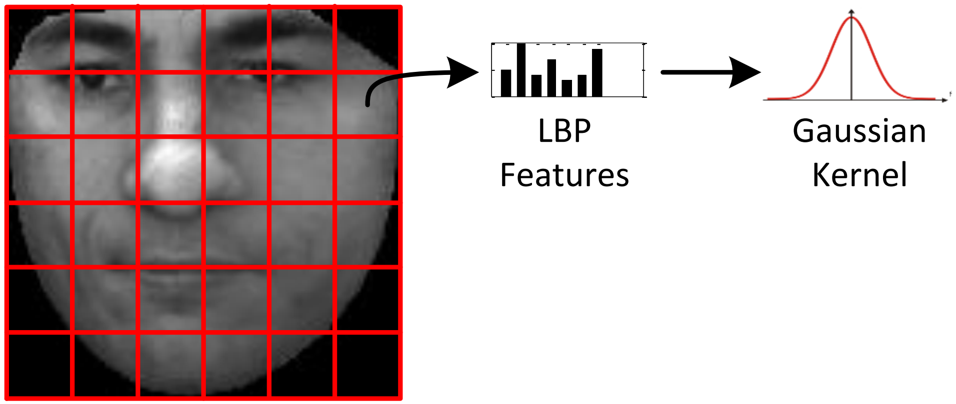
MKL Example: AU Recognition

Input:

- Divide face into 6x6 patches
- LBP features and Gaussian kernel applied to each patch

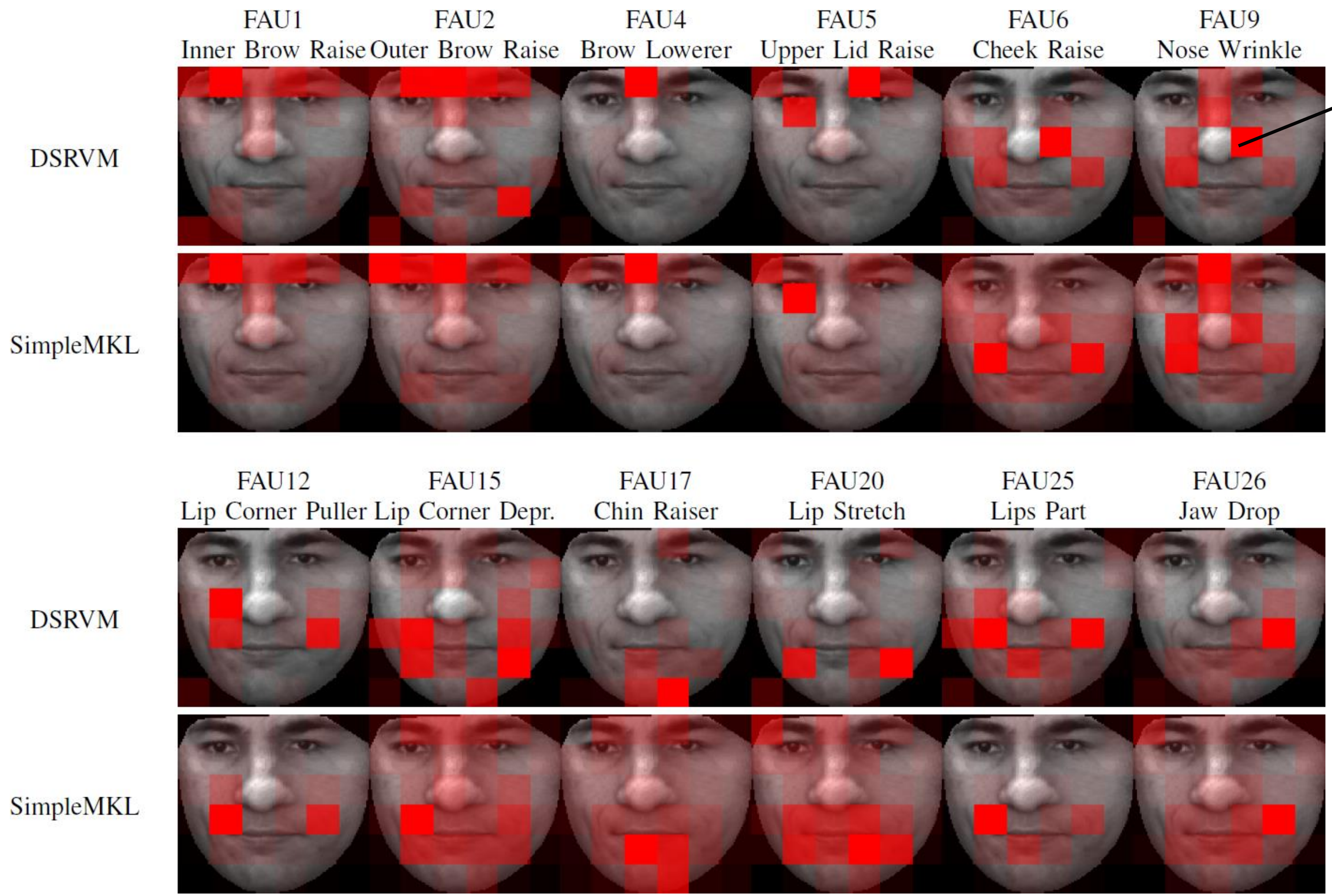
Target:

AU intensities from the DISFA database



Imperial College London

MKL Example: AU Recognition



Each patch represents a kernel and the redness its weight

MKL comes at a cost: training time

		#SV	#Active Kernels	Train Time	Test Time	CORR
RVM	RVM all	111.5	32	1.2	6.6	0.38
	RVM best	71.4	1	1.3	0.7	0.31
MKL	SimpleMKL	1913.9	33	149.9	38.8	0.39
	DSRVM	43.5	17	21.3	1.8	0.43
	mRVM	47.0	36	78.0	6.0	0.32

- MKL can be seen as a weighted ensemble of kernel methods that is jointly trained
- The training time increases with the number of kernels