Machine Learning on Functional Data

Barnabás Póczos

June 9, 2015

Overview

- **Motivation**
- **Function Classification/Regression**
- **Finding anomalous functions and distributions**
- **Function-to-Function Regression**
- **FuSSO = Functional Shrinkage and Selection Operator (Functional Lasso)**

ML on complex objects

Most ML algorithms operate on feature vectors...

input → output

- sets
- distributions
- functions
- graphs
- stochastic processes
- …

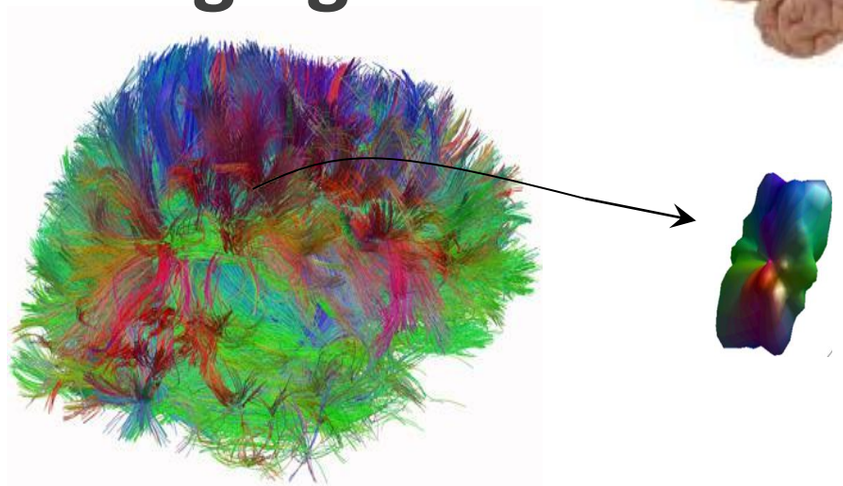
Questions:

- ❑ How to learn?
- ❑ What can we learn?
- ❑ How fast can we learn?
- ❑ How many sample points do we need?
- ❑ Supervised/unsupervised/active learning …

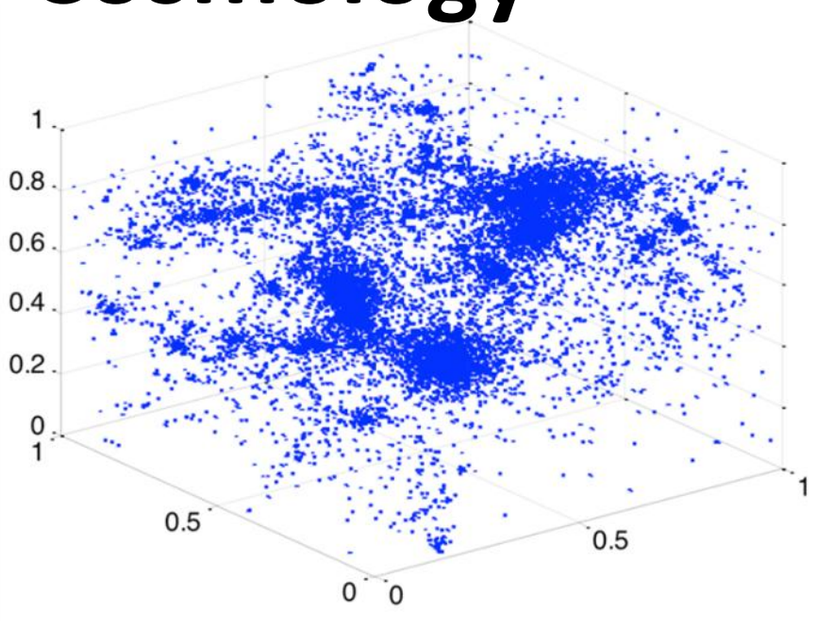
Functional Data is Everywhere

Finance

Neuroscience Diffusion Weighted Imaging



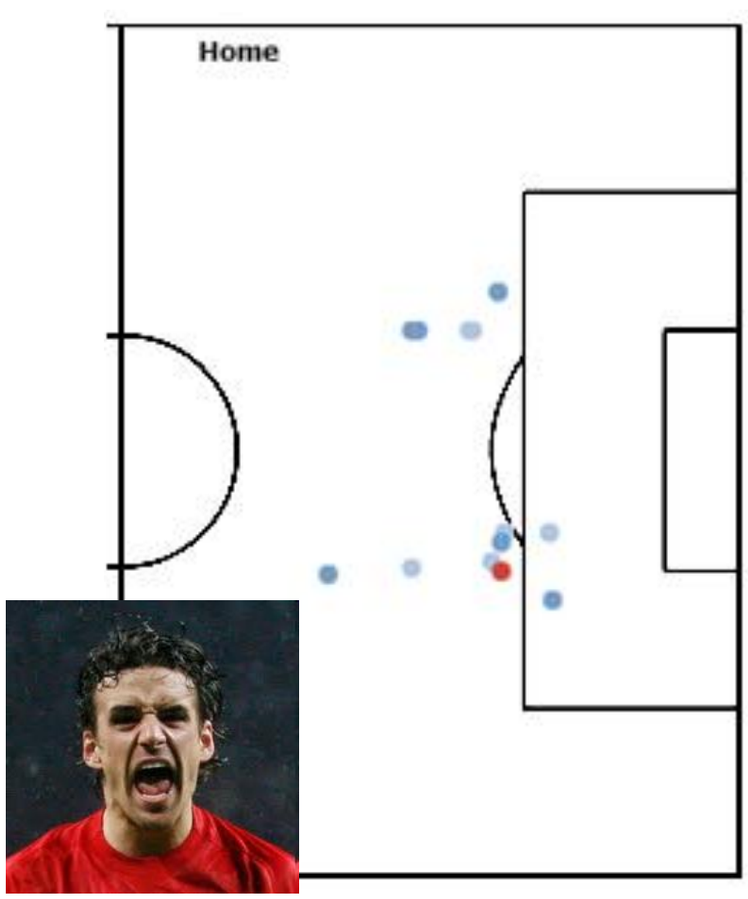
Cosmology



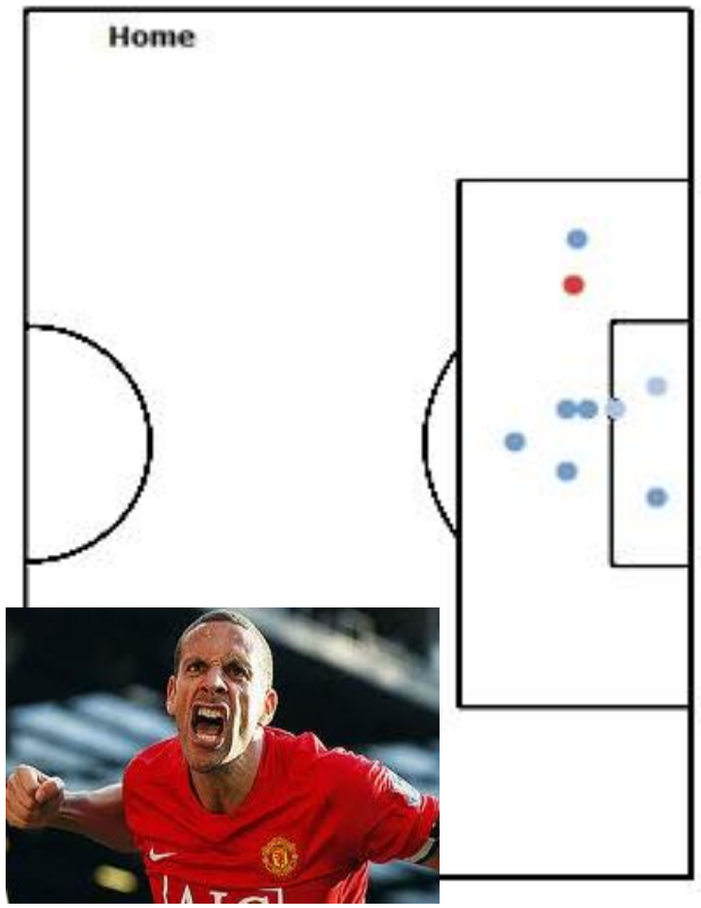
Images

Motivating Examples

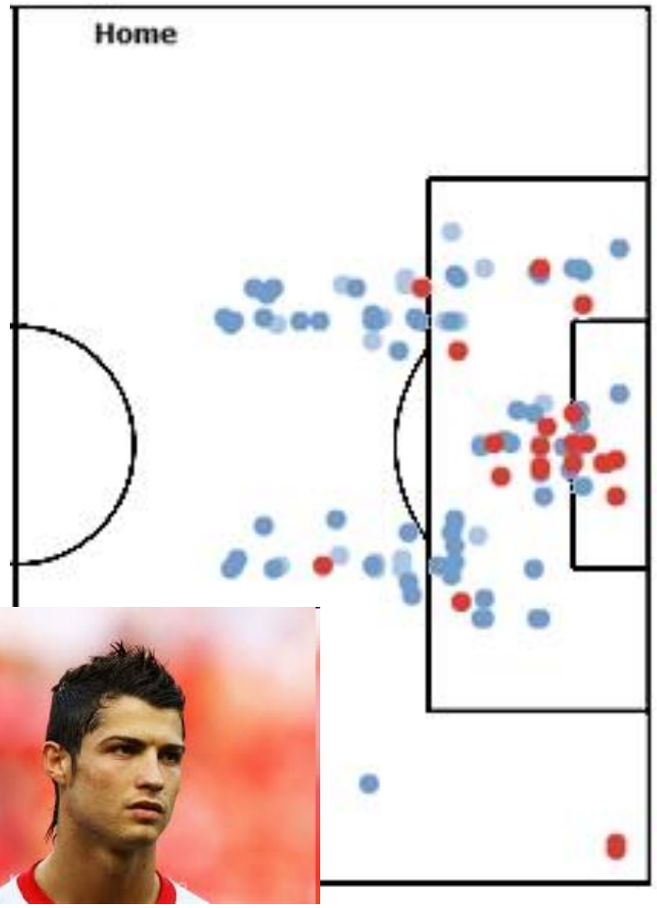
Manchester United 07/08



Owen Hargreaves



Rio Ferdinand



Cristiano Ronaldo

Shot Type

- Goals
- Shots on Goal
- **Shots**

www.juhokim.com/projects.php

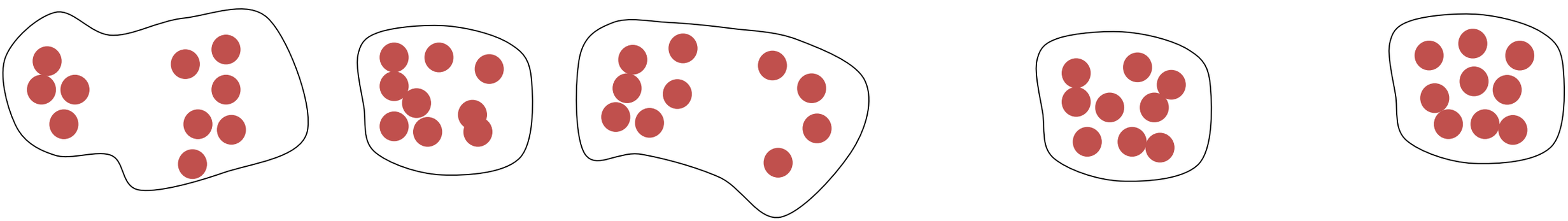
Distribution Regression / Classification

$Y_1=1$ $Y_2=0$ $Y_3=1$ $Y_m=0$?

Differences compared to standard methods on vectors

- ❑ The inputs are distributions, density functions (not vectors)
- ❑ We don't know these distributions, only sample sets are available (error in variables model)

$\mathcal{X}_1$ $\mathcal{X}_2$ $\mathcal{X}_3$ $\mathcal{X}_m$ $\mathcal{X}_{m+1}$



ML on Distributions

healthy or sick?

Standard machine learning

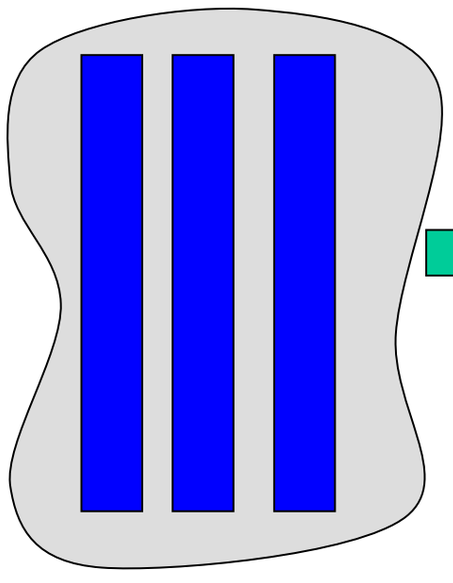
ML on sets/distributions

Medical tests:

blood pressure,
heart rate,
temperature,
blood sample
…

Feature vector

Set of feature vectors



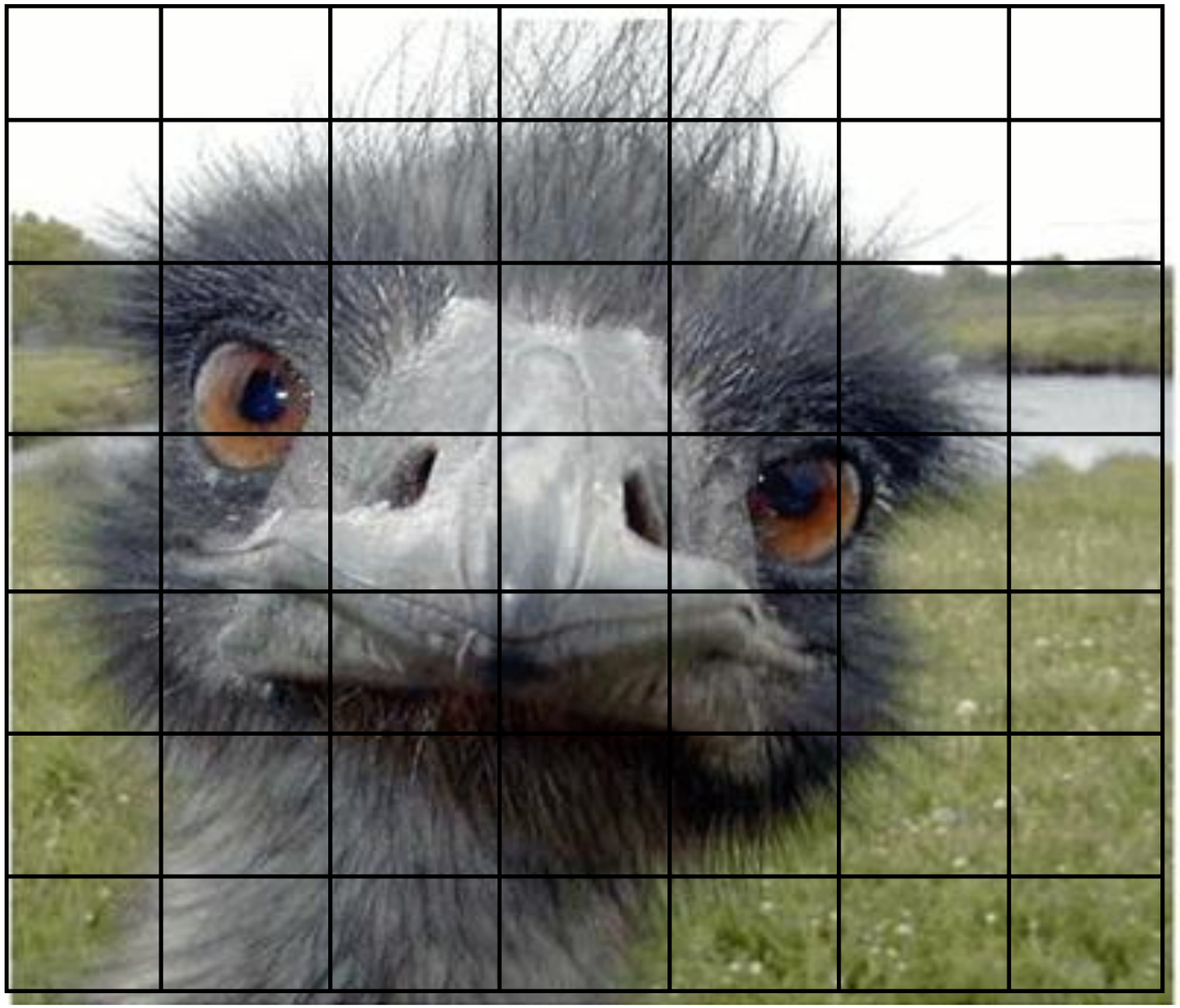
Classifier

Healthy

Sick

What happens if we repeat the medical tests?

ML on Distributions



Dealing with complex objects

- ❑ break into smaller parts, represent the input as a **set** of smaller parts
- ❑ treat the set elements as sample points from some **unknown distribution**
- ❑ do ***ML on these unknown distributions*** represented by sets

Distribution Classification

We have T sample sets, $(\mathbf{X}_1, \ldots, \mathbf{X}_T)$. [**Training data**]
$\{X_{t,1}, \ldots, X_{t,m_t}\} = \mathbf{X_t} \sim p_t$. $\mathbf{X_t}$ has class $Y_t \in \{-1, +1\}$.

What is the class label Y of $\mathbf{X} = \{X_1, \ldots, X_m\} \sim p$?

Solution: Use RKHS based SVM!

Calculate the Gram matrix $K_{ij} \doteq \langle \phi(p_i), \phi(p_j) \rangle_{\mathcal{K}} = K(p_i, p_j)$

Dual form of SVM:

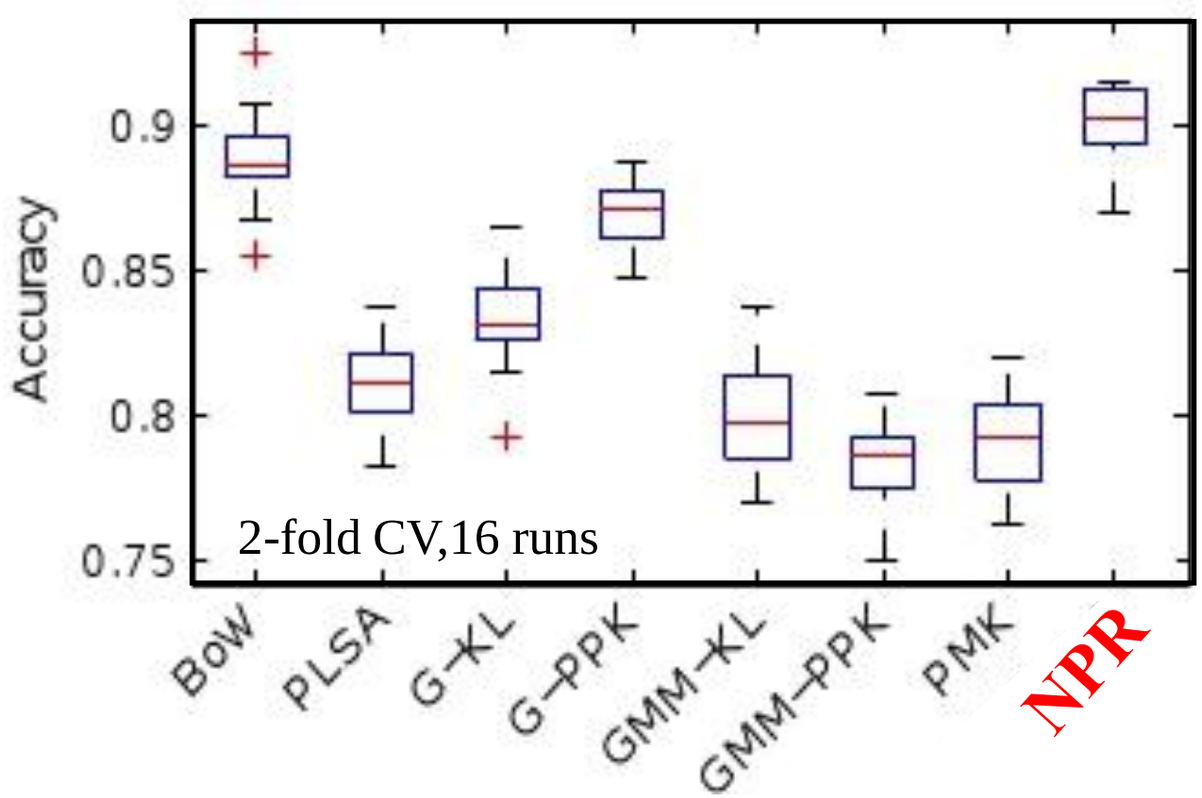
$$\hat{\alpha} = \arg\max_{\alpha \in \mathbb{R}^T} \sum_{i=1}^{T} \alpha_i - \frac{1}{2} \sum_{i,j}^{T} \alpha_i \alpha_j y_i y_j K_{ij}, \quad \text{subject to } \textstyle\sum_i \alpha_i y_i = 0, \; 0 \leq \alpha_i \leq C.$$

$$Y = \text{sign}(\sum_{i=1}^{T} \hat{\alpha}_i y_i K(p_i, p)) \in \{-1, +1\}$$

Problems: We do not know p_i, p, $K(p_i, p_j)$, or $K(p_i, p)$...

Object Classification
ETH-80 [Leibe and Schiele, 2003]

8 categories, 400 images, each image is represented by 576 18 dim points



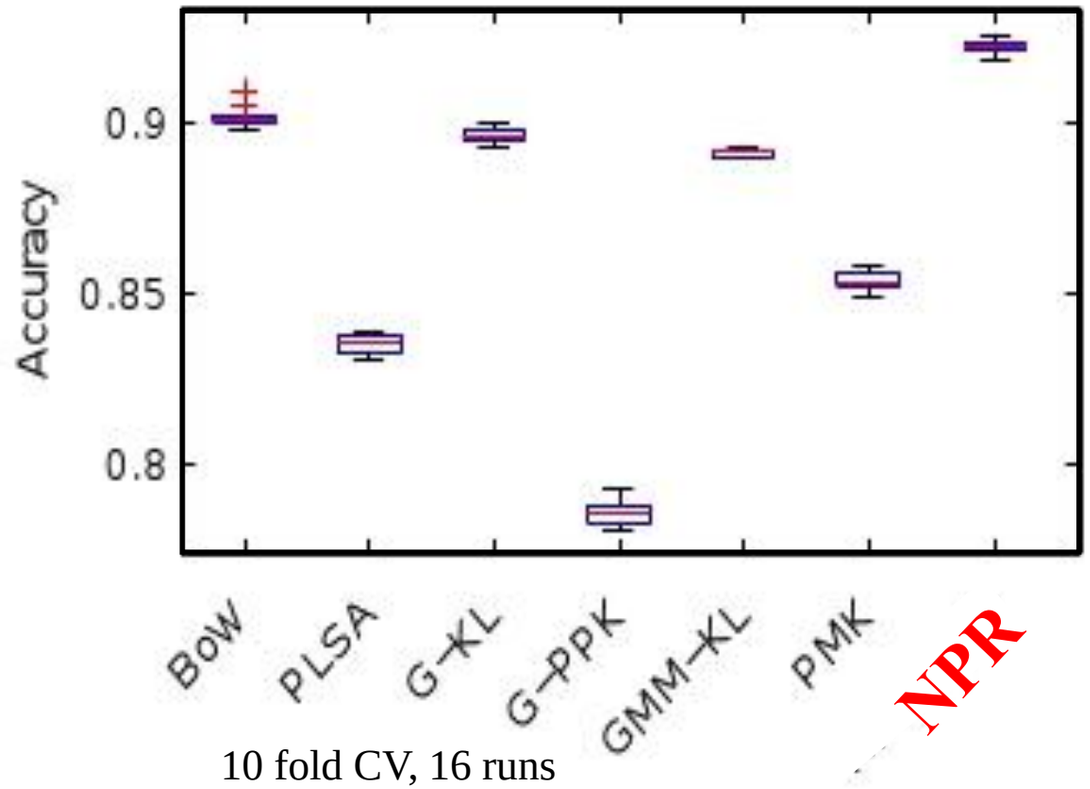
- BoW: **88.9%**
- NPR: **90.1%**

Póczos, Xiong, Sutherland, & Schneider, *CVPR 2012*

Outdoor Scenes Classification
[Oliva and Torralba, 2001]

8 categories, 2688 images,
each represented by 1815 53 dim points.

- Best published: **91.57%**
  (Qin and Yung, ICMV 2010)
- NPR: **92.3%**

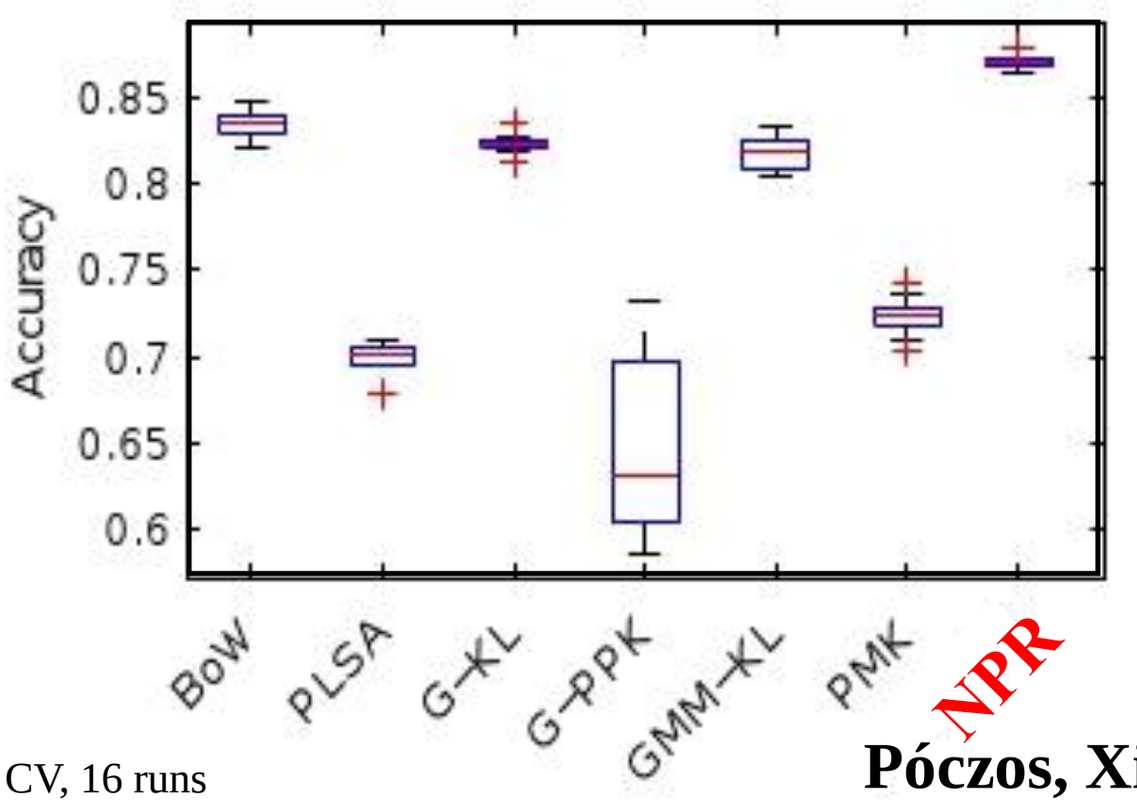


10 fold CV, 16 runs

Sport Events Classification
[Li and Fei Fei, 2007]

badminton bocce croquet polo climbing rowing sailing snowboard

8 categories, 1040 images, each represented by 295 to 1542 57 dim points.

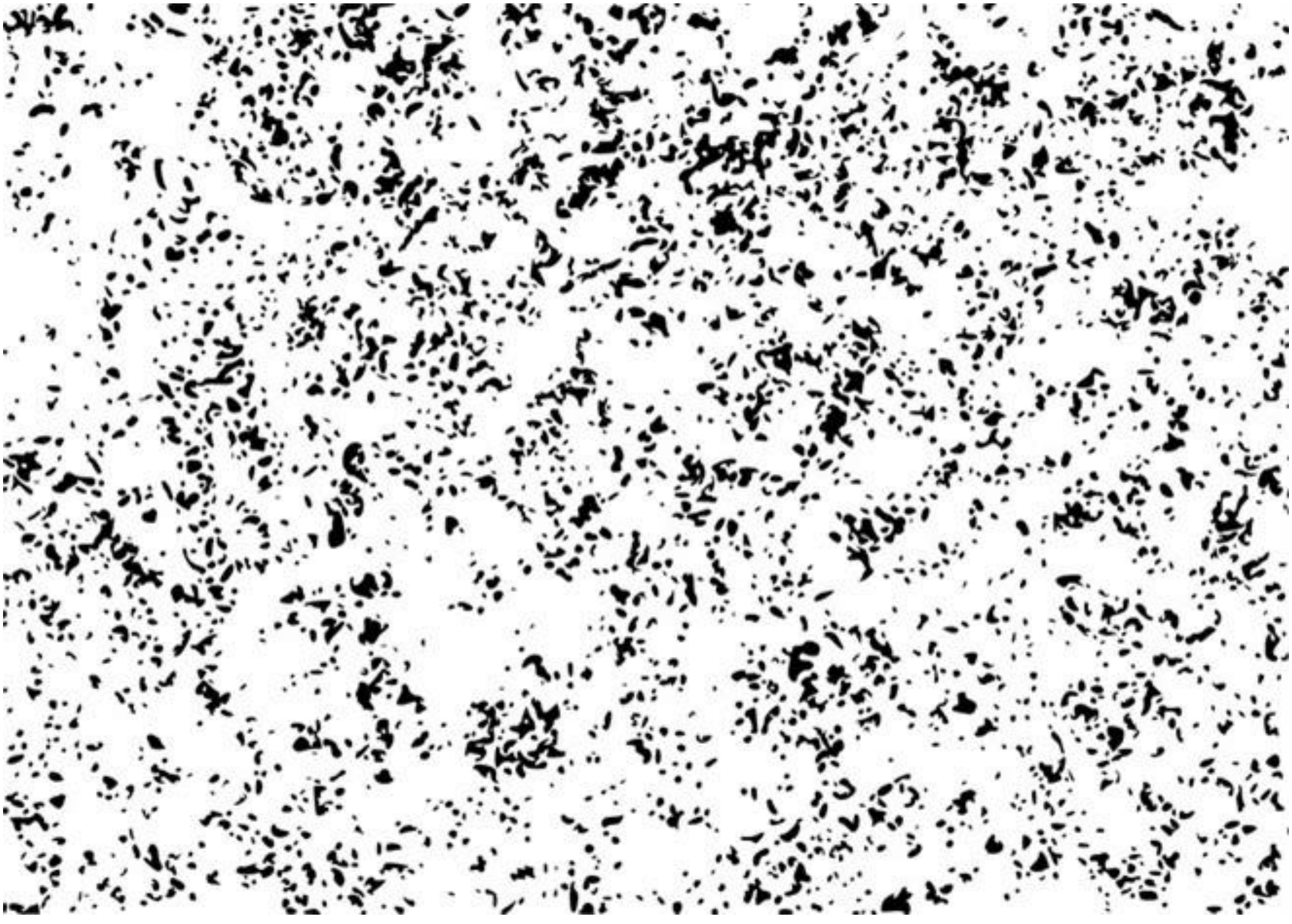


- Best published: **86.7**%
  (Zhang et al, CVPR 2011)
- NPR: **87.1**%

2 fold CV, 16 runs

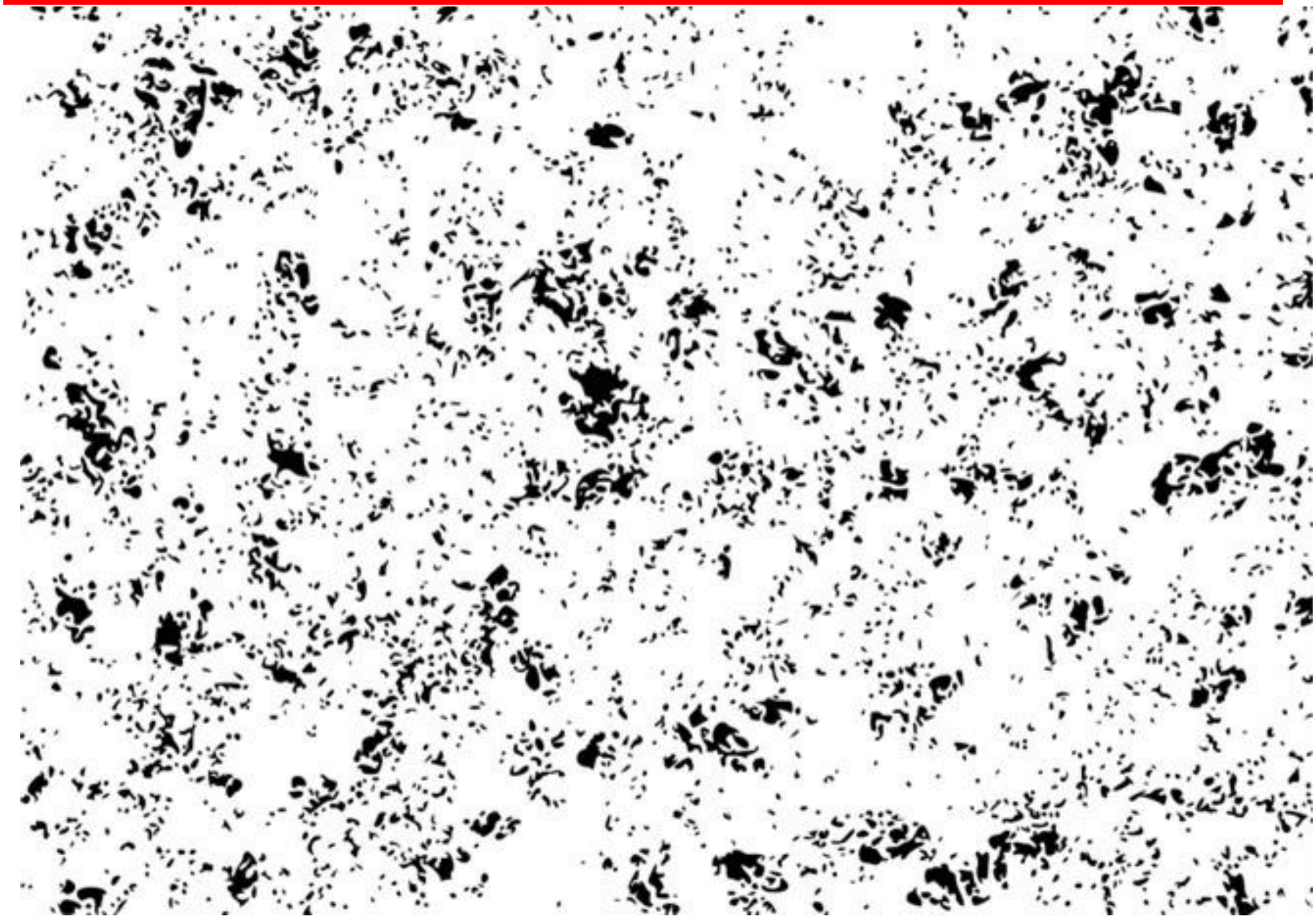
Póczos, Xiong, Sutherland, & Schneider, *CVPR 2012*

Emerging images



Niloy J. Mitra, 2009

Emerging images



Niloy J. Mitra, 2009

Image Representation with Distributions

Dealing with complex objects

- ❑ break into smaller parts,
- ❑ represent the object as a sample set of these parts

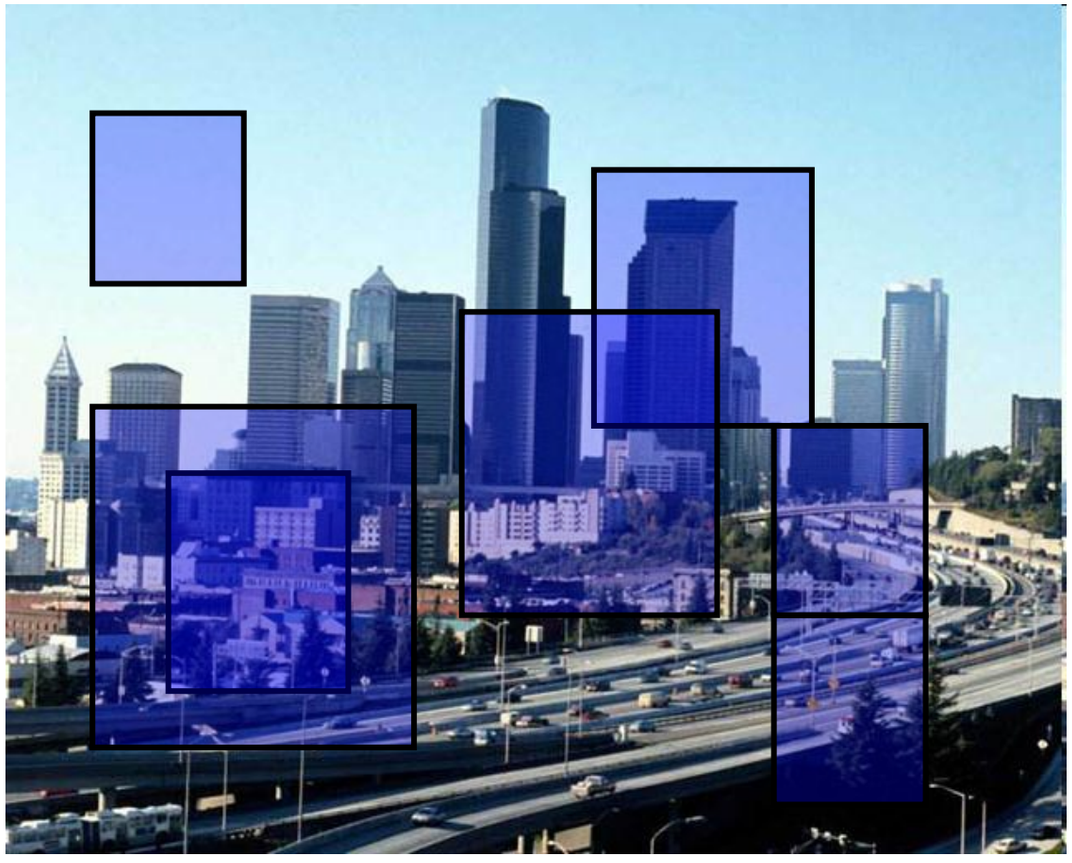


Image patches
- •Overlapping
- •Non-overlapping

Patch locations
- •Grid points
- •Interesting points
- •Random

Patch sizes
- •Same
- •Different,
- •Hierarchy

d-dimensional sample set representation of the image

- ❑ Each image ***patch*** is represented by PCA compressed SIFT vectors.

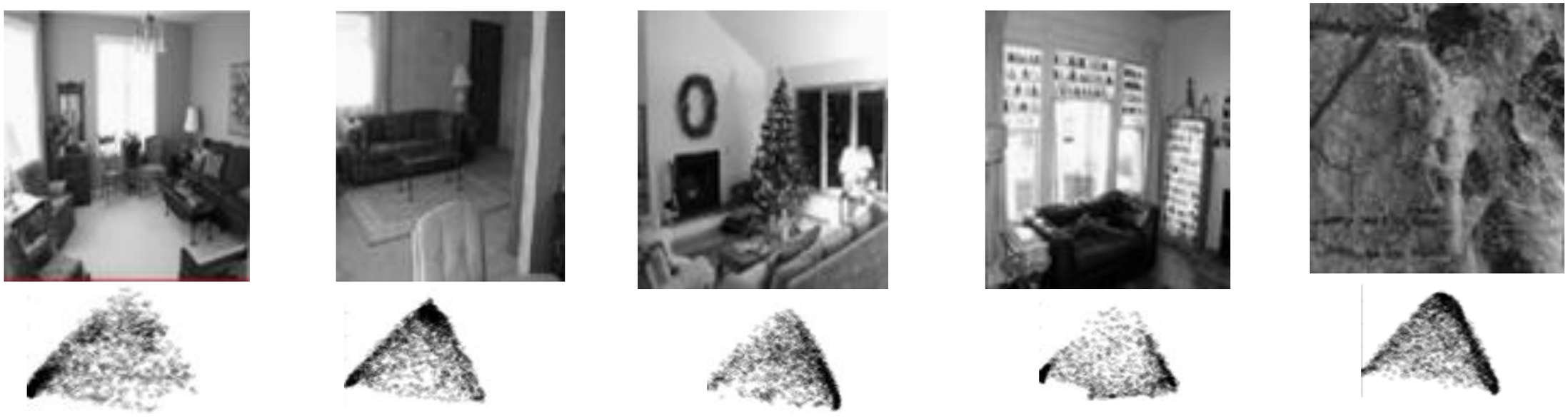
  SIFT = *Scale-invariant feature transform.* ***PCA:*** *128dim⇒ d dim*
- ❑ Each ***image*** is represented as a ***set*** of these *d* dim feature vectors.
- ❑ Each ***set*** is considered as a sample set from some ***unknown distribution***.

Detecting Anomalous Images

B. Póczos, L. Xiong & J. Schneider, UAI, 2011.

50 highway images

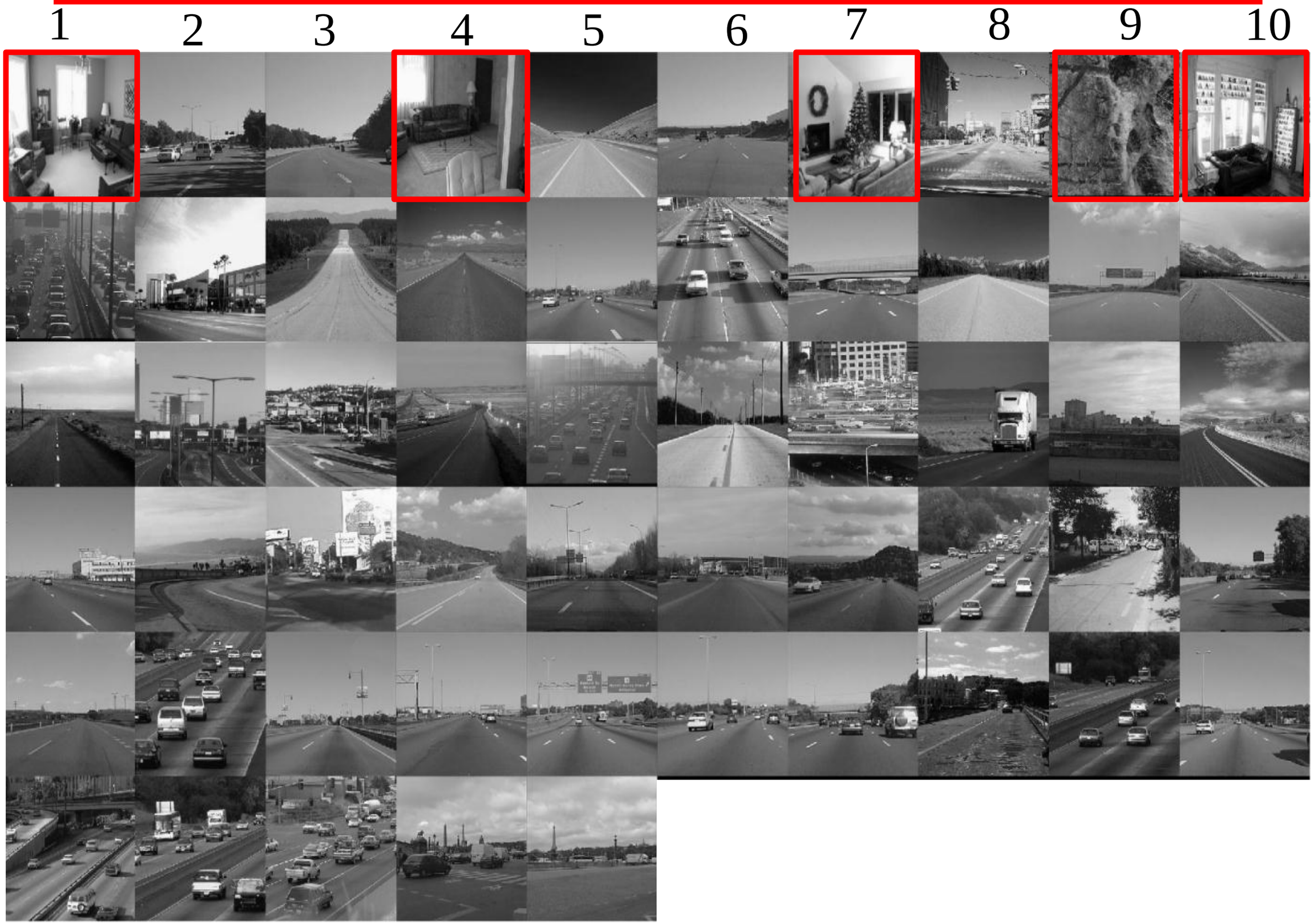
5 anomalies



2-dimensional sample set representation of images (128 dim SIFT ⇒ 2 dim)

Anomaly score: divergences between the distributions of these sample sets

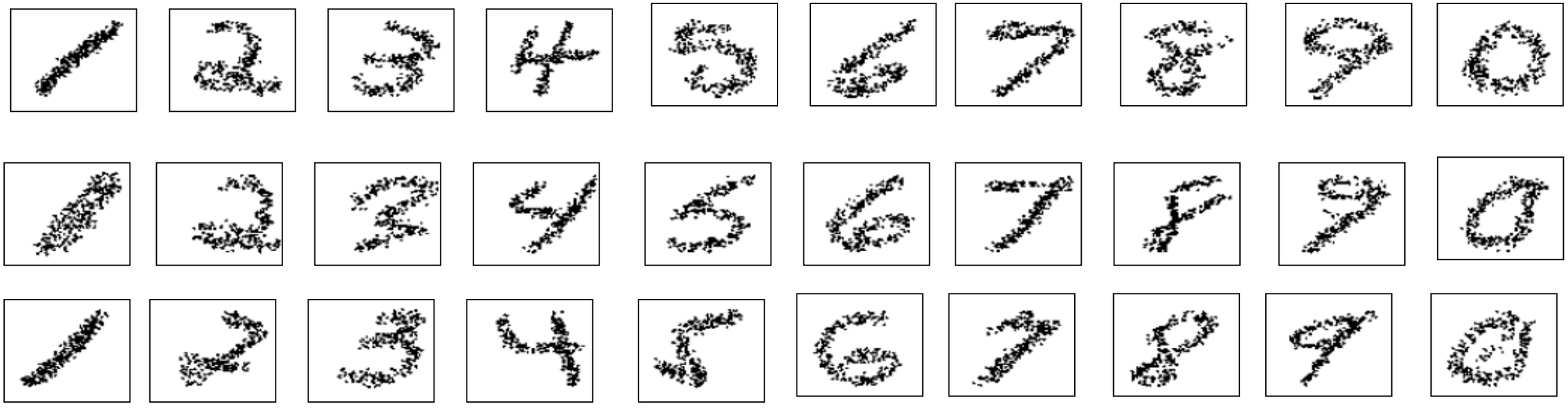
Detecting Anomalous Images



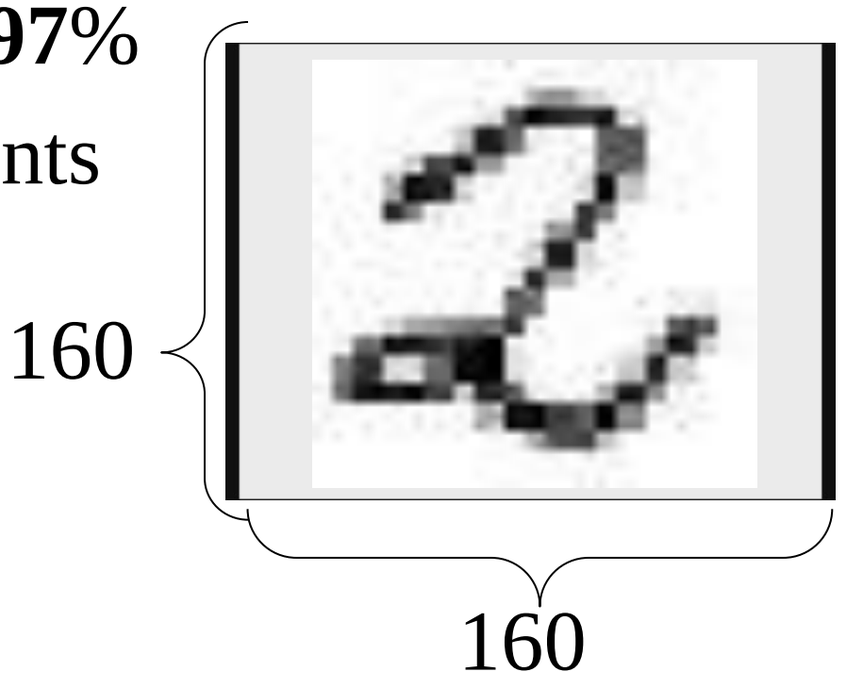
1 2 3 4 5 6 7 8 9 10

51 52 53 54 55

Noisy USPS Dataset Classification with SDM



- Original (noiseless) USPS dataset is easy ~**97**%
- Each instance (image) is a set of 500 2d points
- 1000 training and 1000 test instances



Results:

SVM on raw images **82.1 ± .5%** accuracy

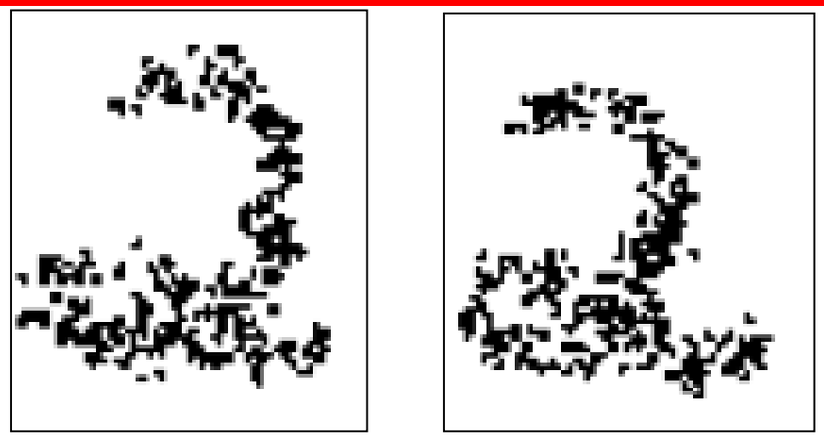
SDM on the 2D distributions, Rényi divergence: **96.0 ± .3%** accuracy

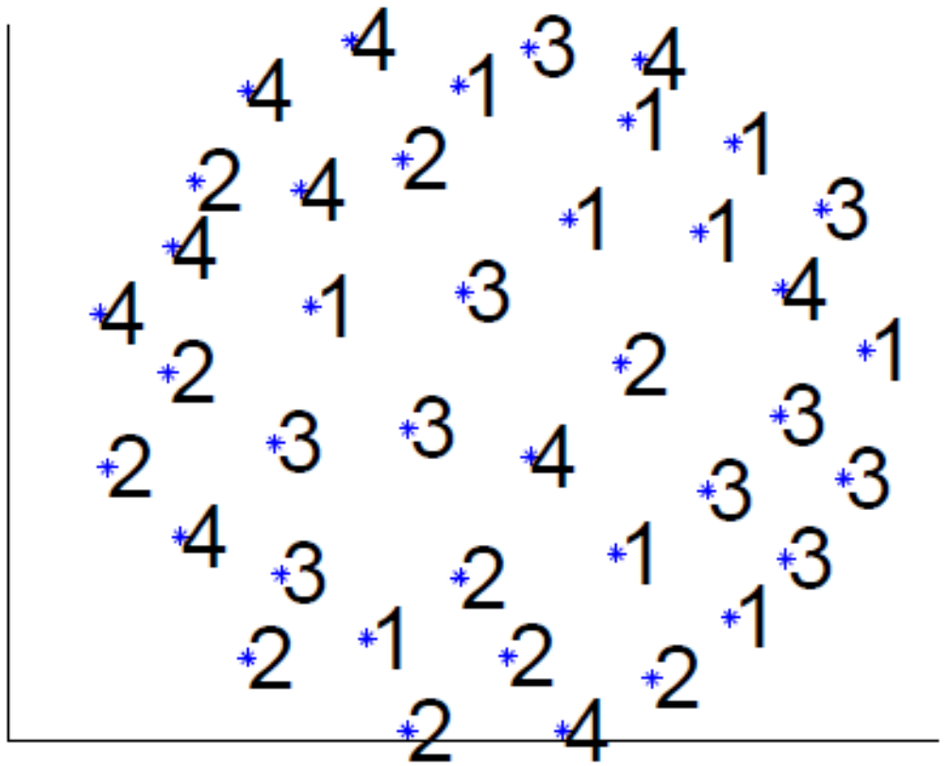
Multidimensional Scaling of USPS Data

10 instances from figures 1,2,3,4.

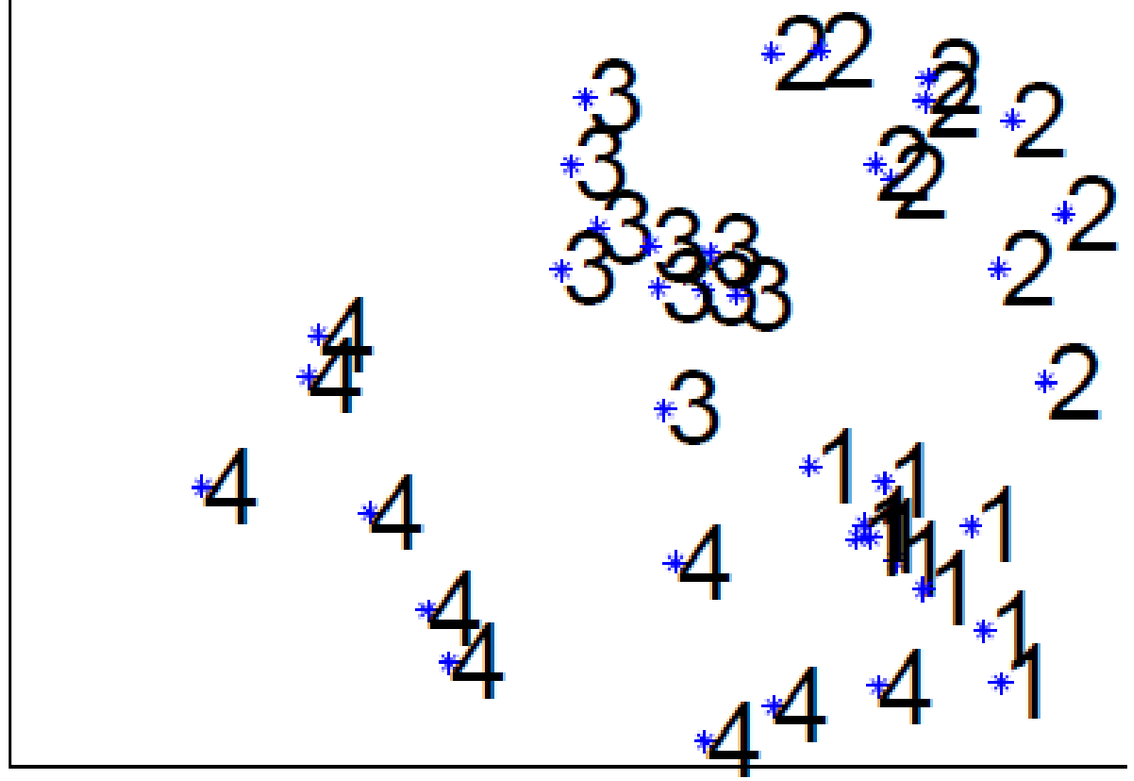
Calculate pairwise Euclidean distances.

Nonlinear embedding with MDS into 2d.





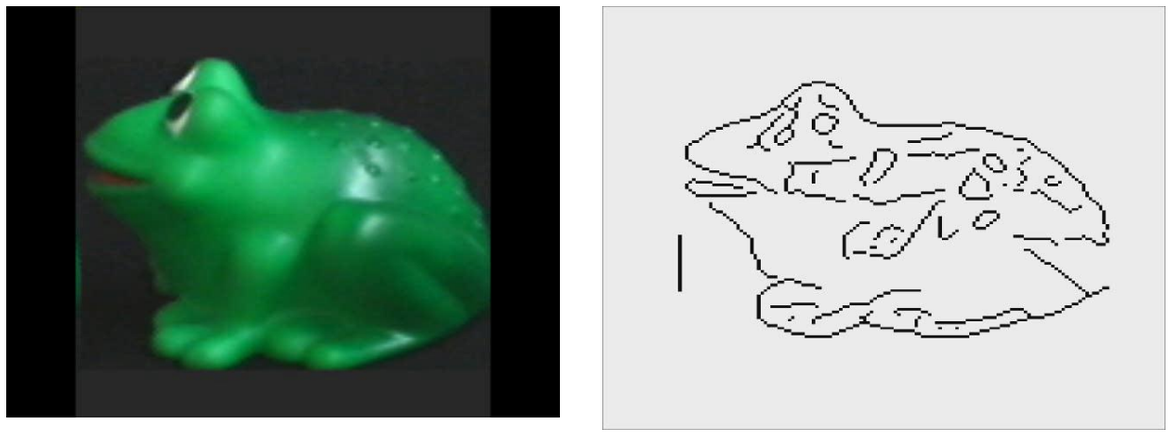
Raw **images**
using Euclidean distance



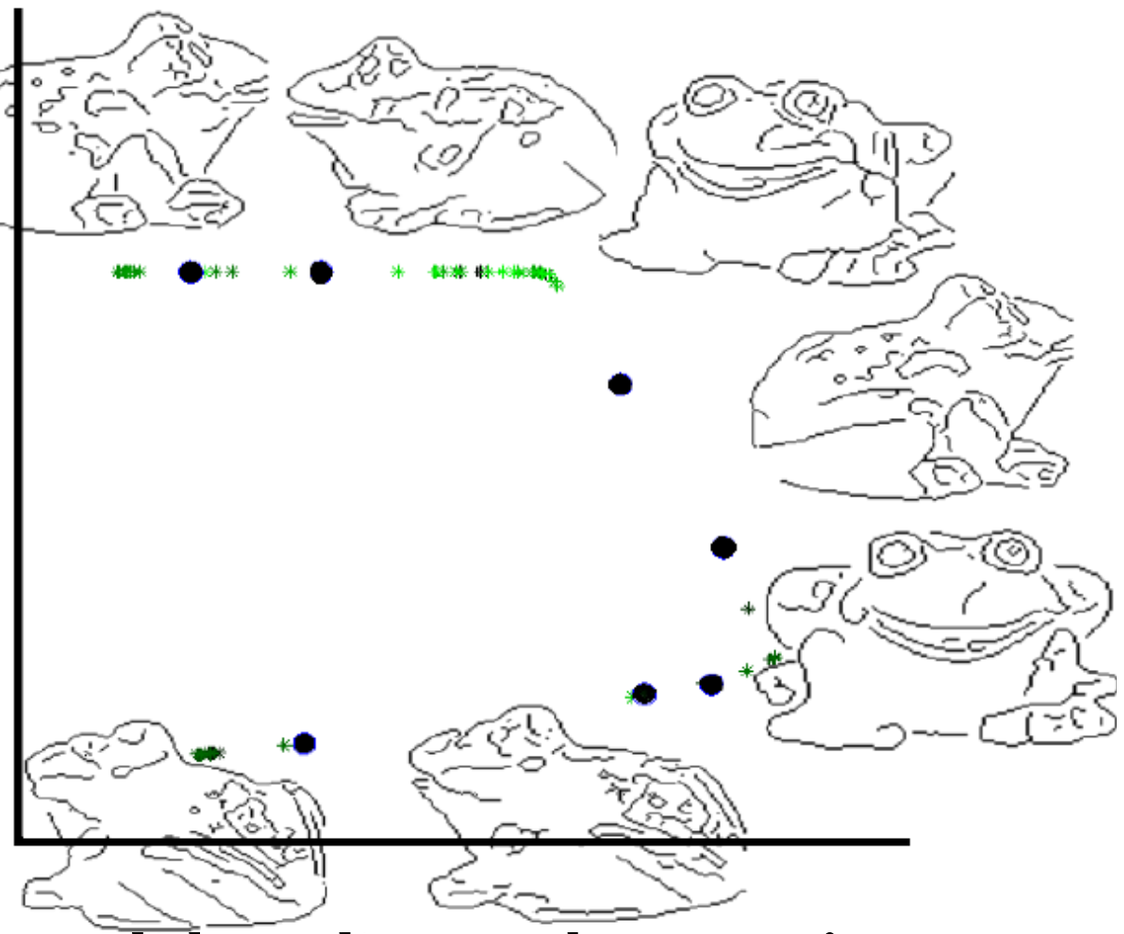
Estimated Euclidean distance
between the **distributions**

Local Linear Embedding of Distributions

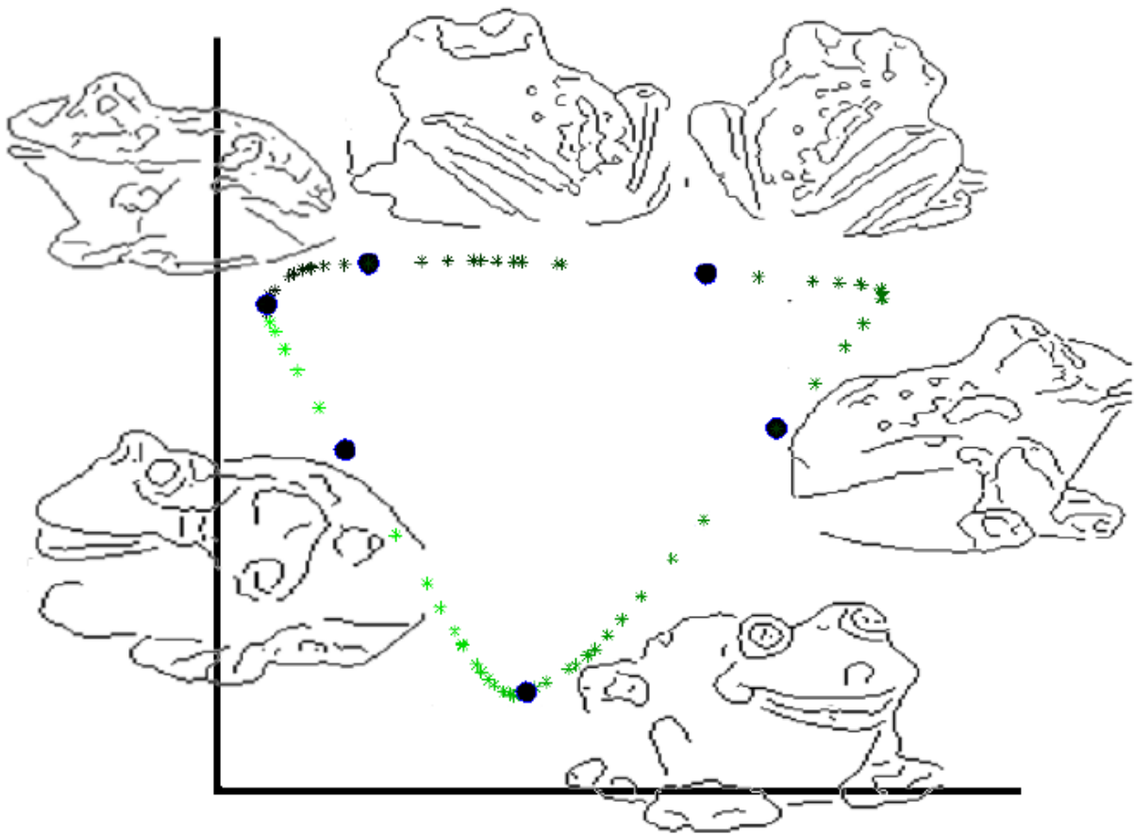
72 rotated COIL froggies



Edge detected COIL froggy



Euclidean distance between **images**

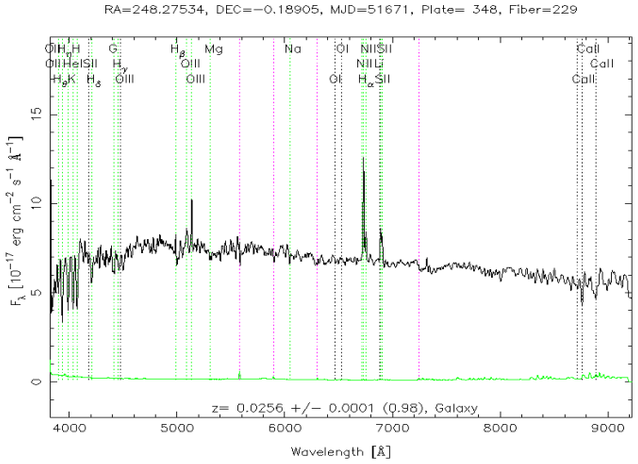


Euclidean distance between **distributions**

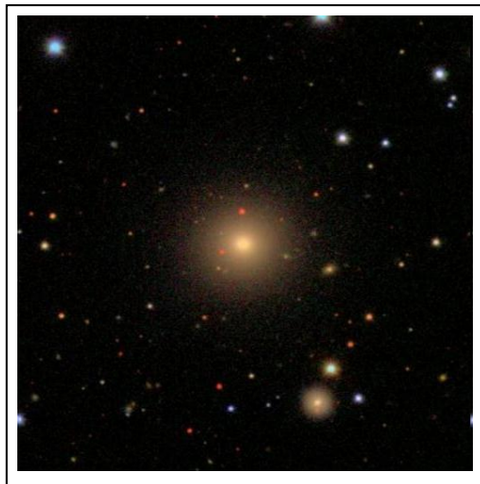
Finding Unusual Galaxy Clusters

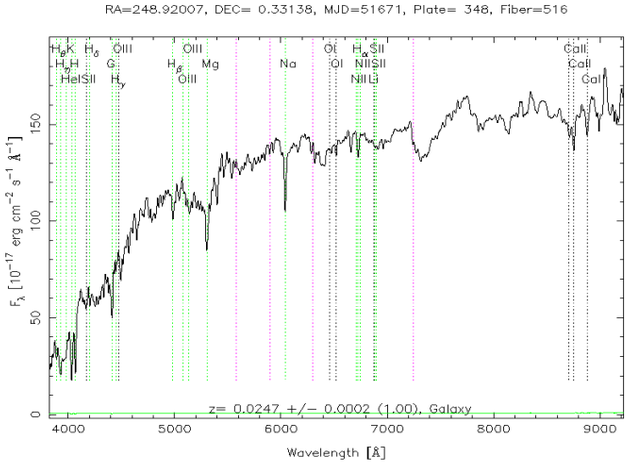
Sloan Digital Sky Survey (SDSS)

- ❑ continuum spectrum
- ❑505 galaxy clusters
  (10-50 galaxies in each)
- ❑7530 galaxies



Blue galaxy





Red galaxy

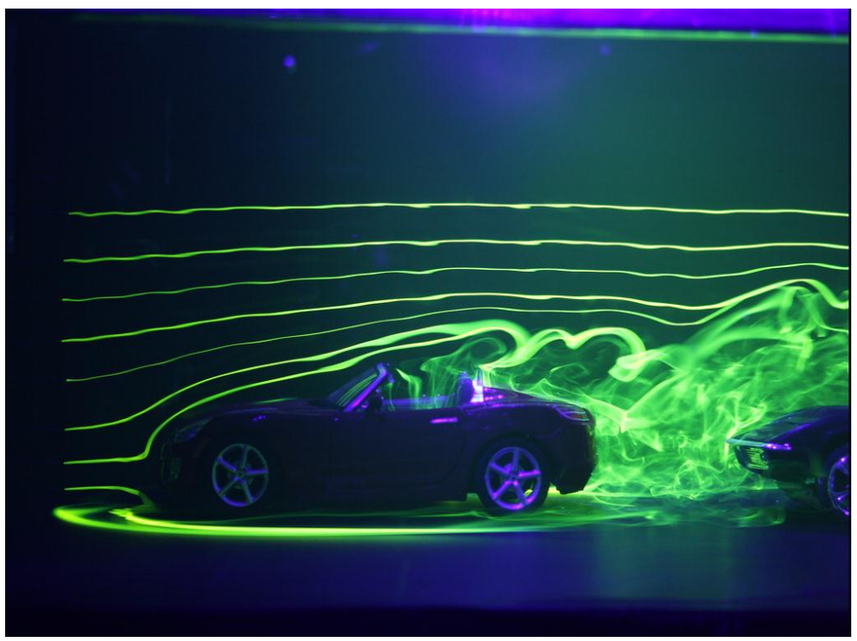
What are the most anomalous galaxy clusters?

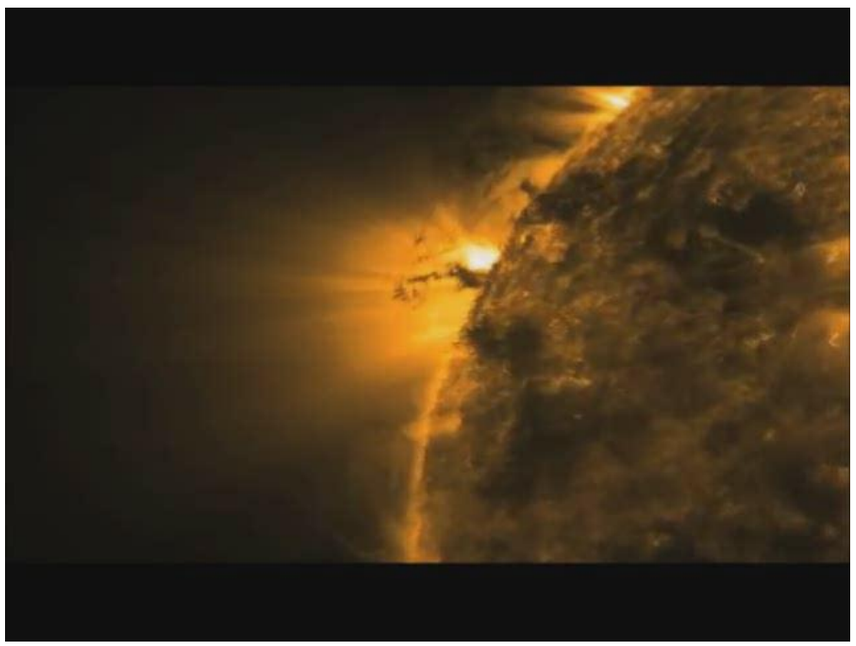
The most anomalous galaxy cluster contains mostly

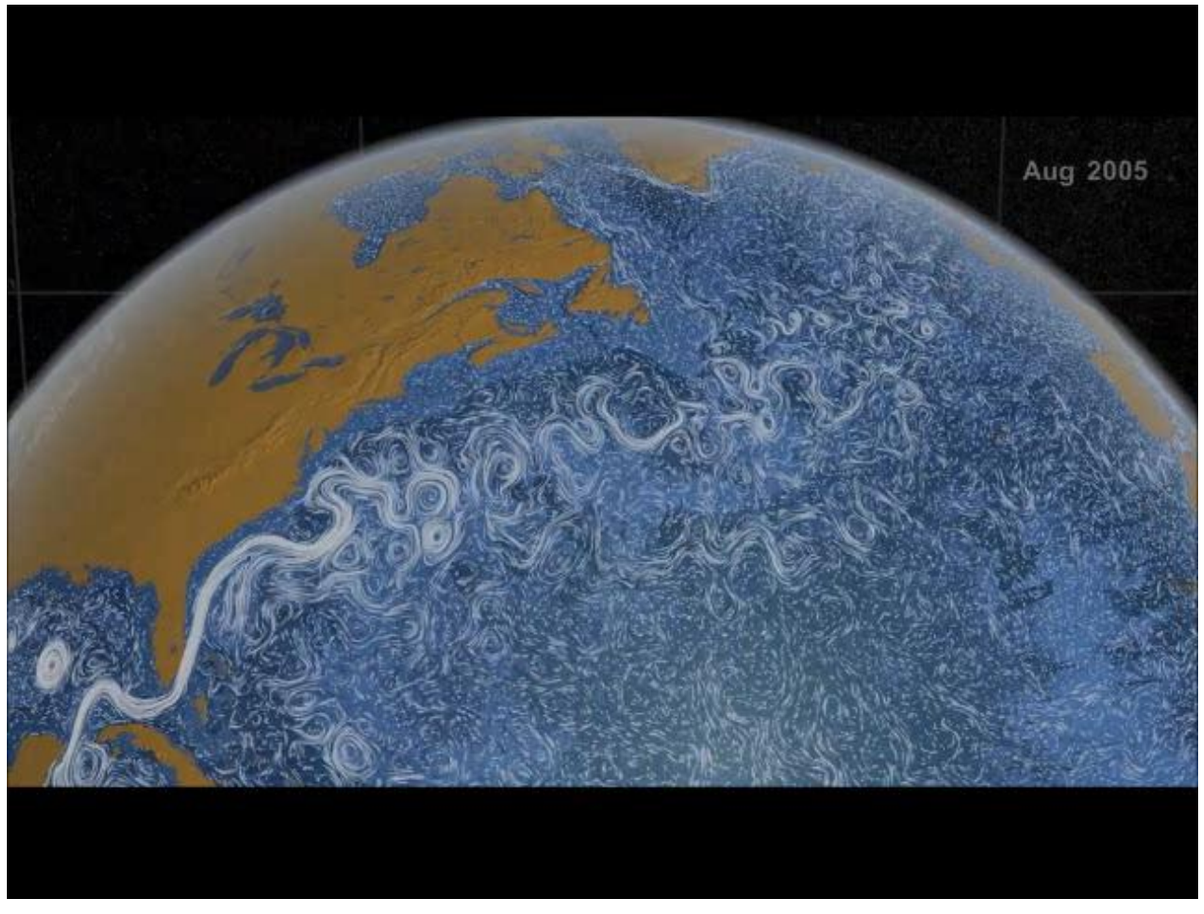
- ❑ star forming blue galaxies
- ❑ irregular galaxies

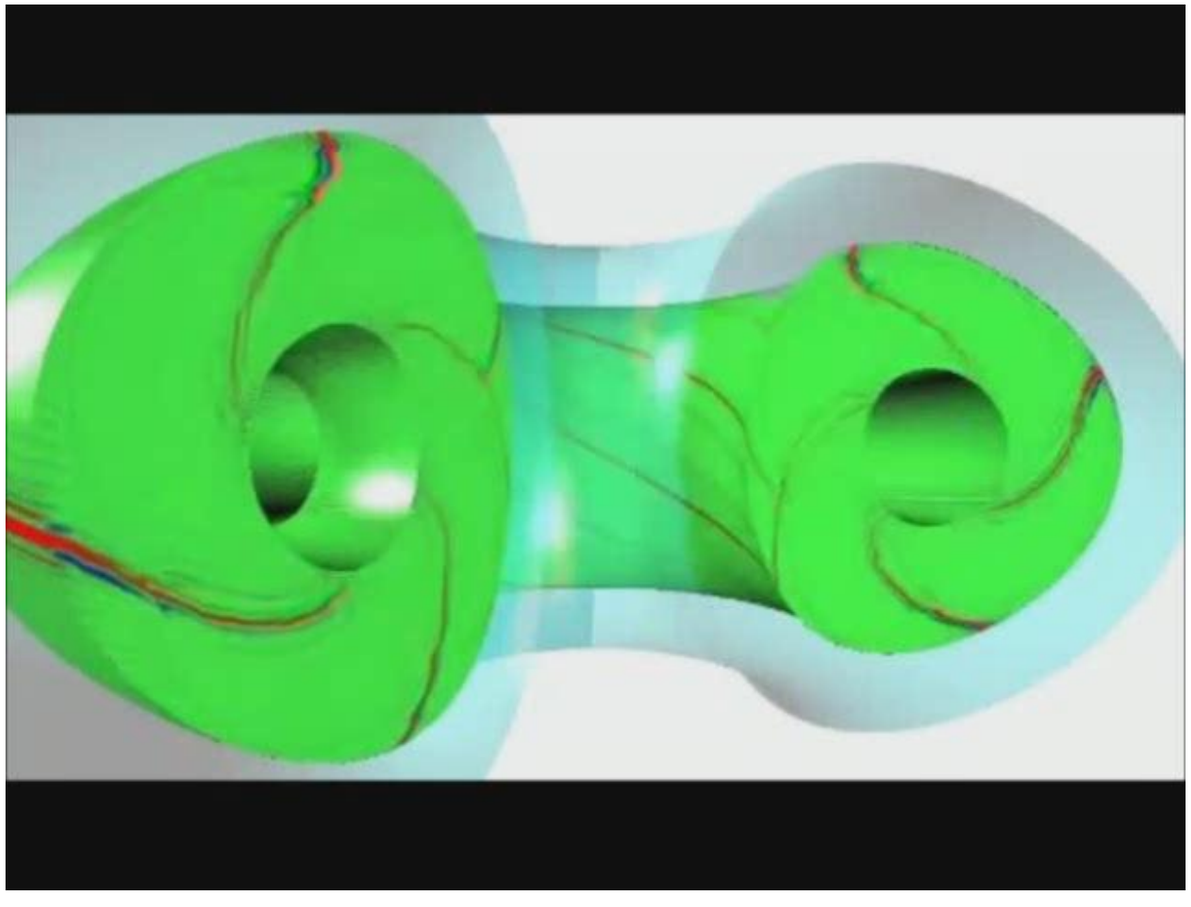
B. Póczos, L. Xiong & J. Schneider, UAI, 2011. Credits: ESA, NASA

Understanding Turbulences







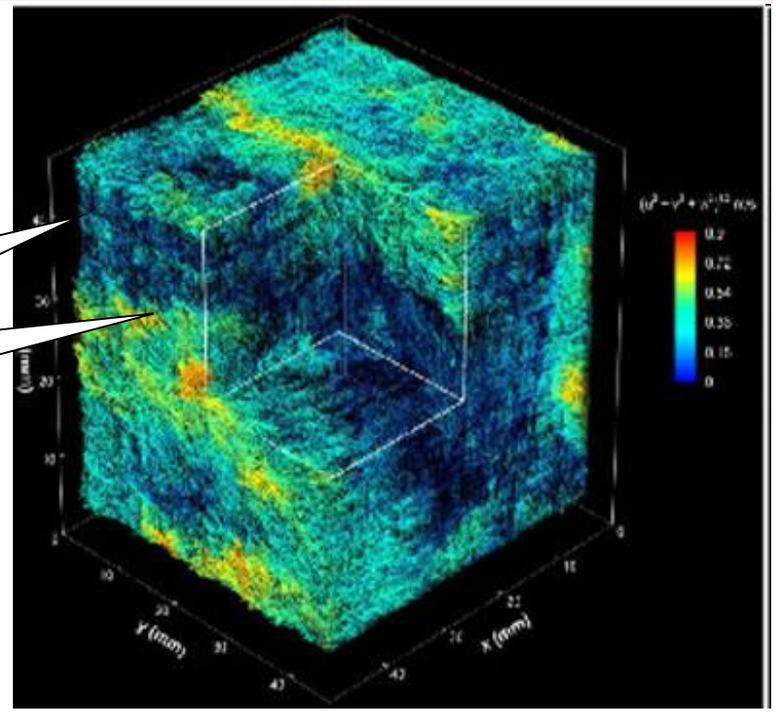


Credits: ESA, NASA, PPPL, Wikipedia

Turbulence Data Classification

Simulated fluid flow through time

(JHU Turbulence Research Group, Alex Szalay)



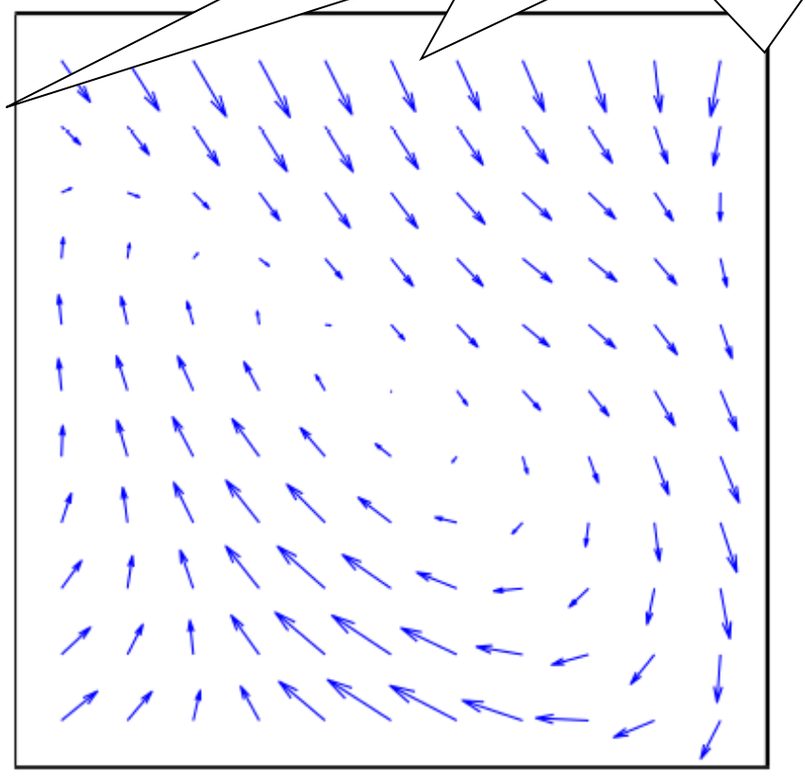
Goal: *find interesting events, patterns, phenomena*

Why is this interesting?

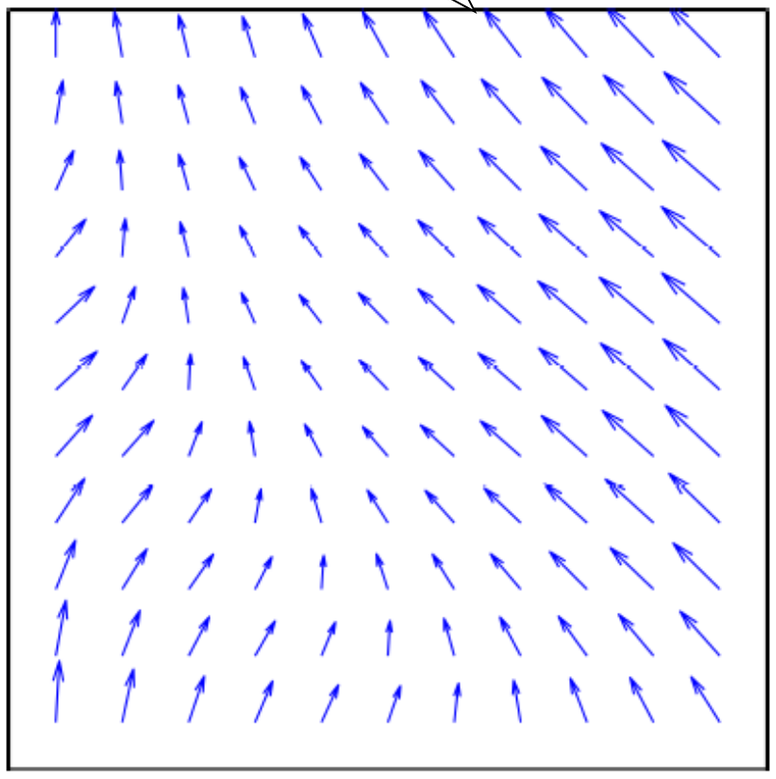
Something interesting happened?

- 11 positive, 2[illegible]
- **Results:** Leave one out cross-validation : 97%

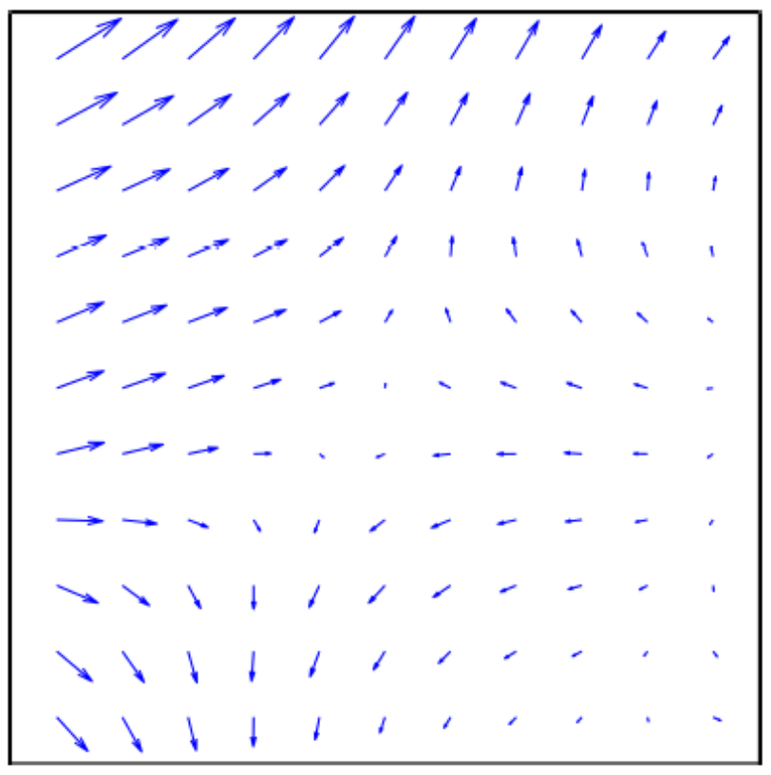
Volumetric data



Positive (vortex)

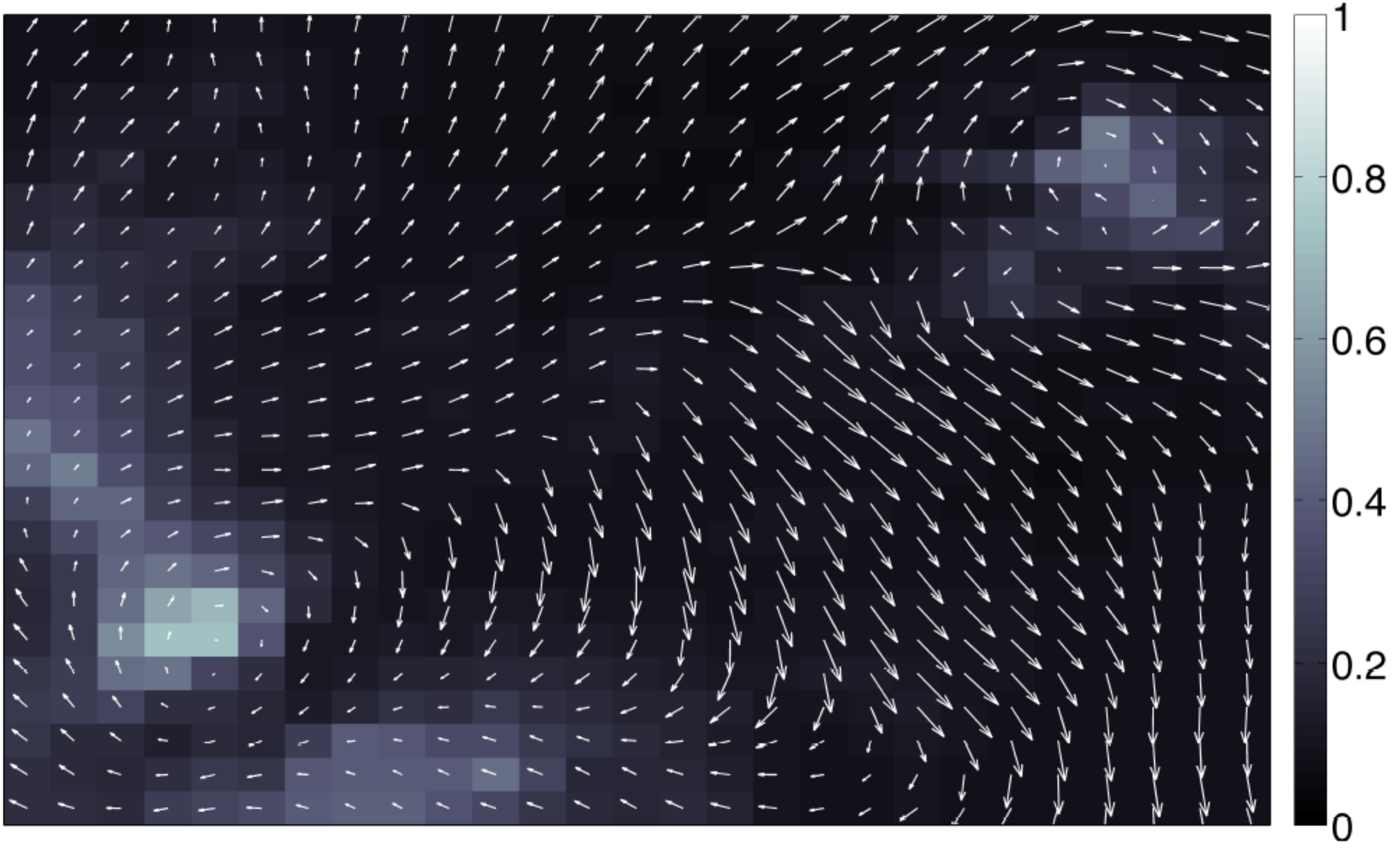


Negative



Negative

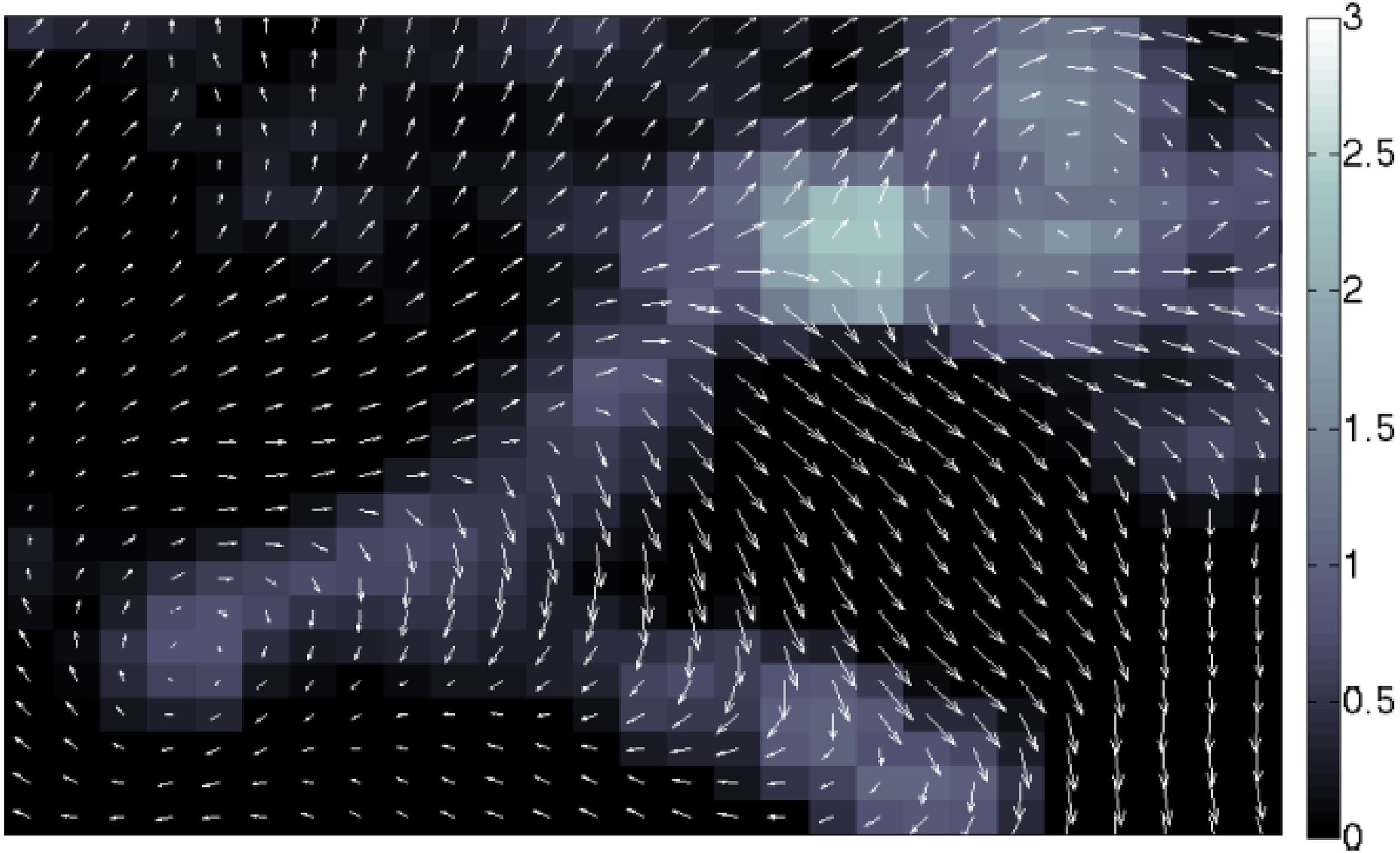
Finding Vortices



Classification probabilities

Find Interesting Phenomena in Turbulence Data

Anomaly detection with 1-class SDM

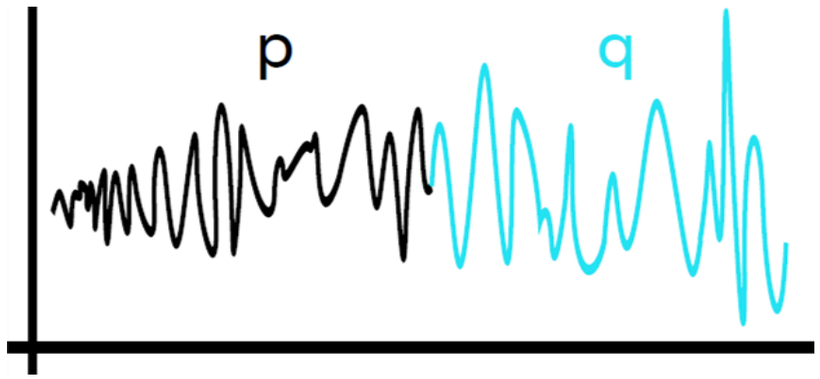


Anomaly scores

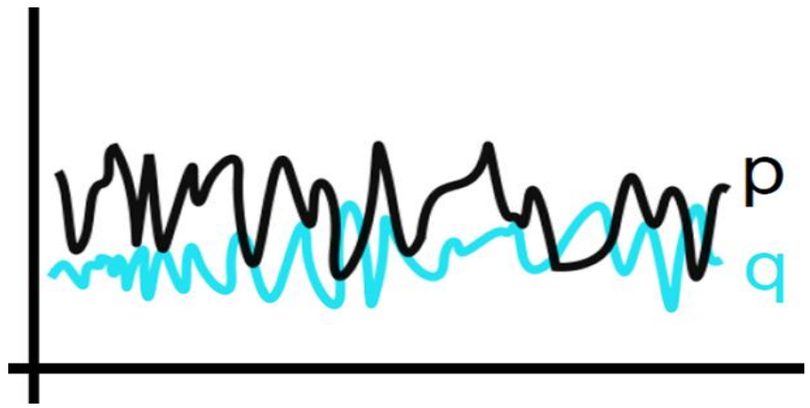
Function-to-Function regression

Function to Function (F2F) Regression

Given an ***input function p,*** can we predict an ***output function q***?

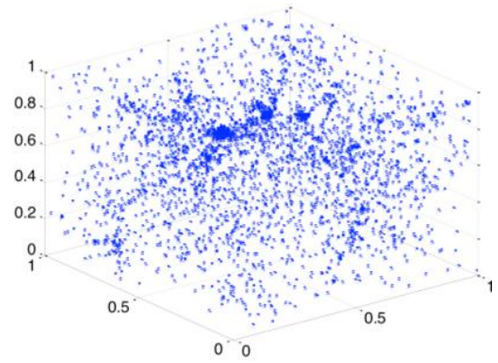


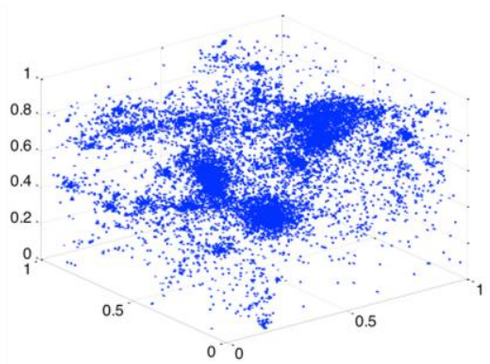
Forward Prediction



Co-occurring Prediction

Image Segmentation



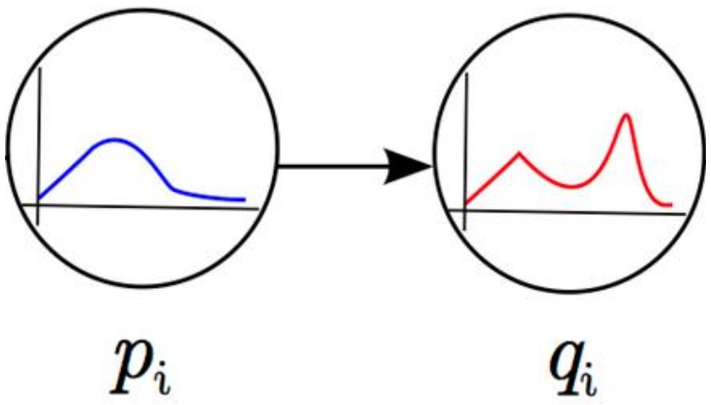


Future Distribution Prediction

Data-Model

We observe functional observation *pairs* (P_i, Q_i), for $i=1,..,N$ corresponding to input/output functions (p_i, q_i), where $f(p_i) = q_i$.

We predict output function q_0 given unseen observation P_0.



estimated density

observation

true density

$i=1,\ldots,N$

$$P_i = \left\{ p_i(u_{ij}) + \epsilon_{ij} \right\}_{j=1}^{n_i},$$
$$Q_i = \left\{ q_i(v_{ij}) + \xi_{ij} \right\}_{j=1}^{m_i}$$

Kernel Methods Don't Scale

Previous Approaches:

- ❑ RKHS approach (Kadri et al. 2010)
- ❑ Kernel smoothing approach (Oliva et al. 2013)

$$\left(K\left(\tilde{p}_1, \tilde{p}_0\right), \ldots, K\left(\tilde{p}_N, \tilde{p}_0\right)\right)^T$$

N evaluations

$$\begin{pmatrix} K\left(\tilde{p}_1, \tilde{p}_1\right) & \cdots & K\left(\tilde{p}_1, \tilde{p}_N\right) \\ \vdots & \ddots & \vdots \\ K\left(\tilde{p}_N, \tilde{p}_1\right) & \cdots & K\left(\tilde{p}_N, \tilde{p}_N\right) \end{pmatrix}$$

N x N

Kernel Methods Don't Scale

Kernel smoothing approach

$$\hat{f}(\tilde{p}_0) = \sum_{i=1}^{N} W(\tilde{p}_i, \tilde{p}_0)\tilde{q}_i, \text{ where}$$

$$W(\tilde{p}_i, \tilde{p}_0) = \begin{cases} \frac{K(D(\tilde{p}_i,\tilde{p}_0))}{\sum_{j=1}^{N} K(D(\tilde{p}_j,\tilde{p}_0))} & \text{if } \sum_{j=1}^{N} K(D(\tilde{p}_j, \tilde{p}_0)) > 0 \\ 0 & \text{otherwise .} \end{cases}$$

$D(p, q)$, is a distance or divergence between densities p and q.

Our Approach: Triple Basis Estimator (3BE)

High-level idea:

- ❑ Embed input functions **p** with a *non-linear* map **z(p)**
- ❑ then use a *linear* map to estimate **q**.

p ⟶ **z(p)** ⟶ **q**

Orthonormal Basis

Let $\{\varphi_i\}_{i\in\mathbb{Z}}$ be an orthonormal basis for $L_2([0,1])$.

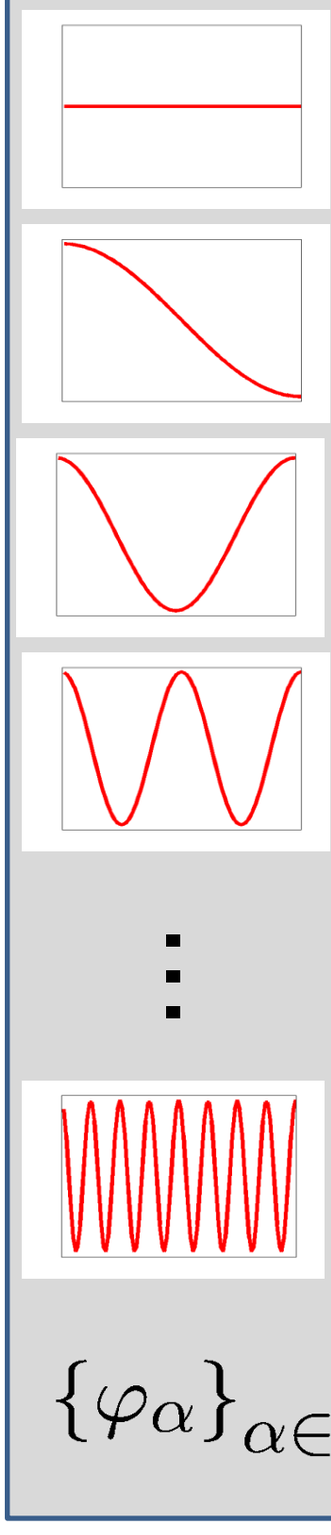
$\{\varphi_\alpha\}_{\alpha\in\mathbb{Z}^d}$ where $\varphi_\alpha(x) = \prod_{i=1}^{d} \varphi_{\alpha_i}(x_i),\ x \in [0,1]^d$.

Let $h \in L_2([0,1]^d)$, then

$$h(x) = \sum_{\alpha\in\mathbb{Z}^d} a_\alpha(h)\varphi_\alpha(x)$$

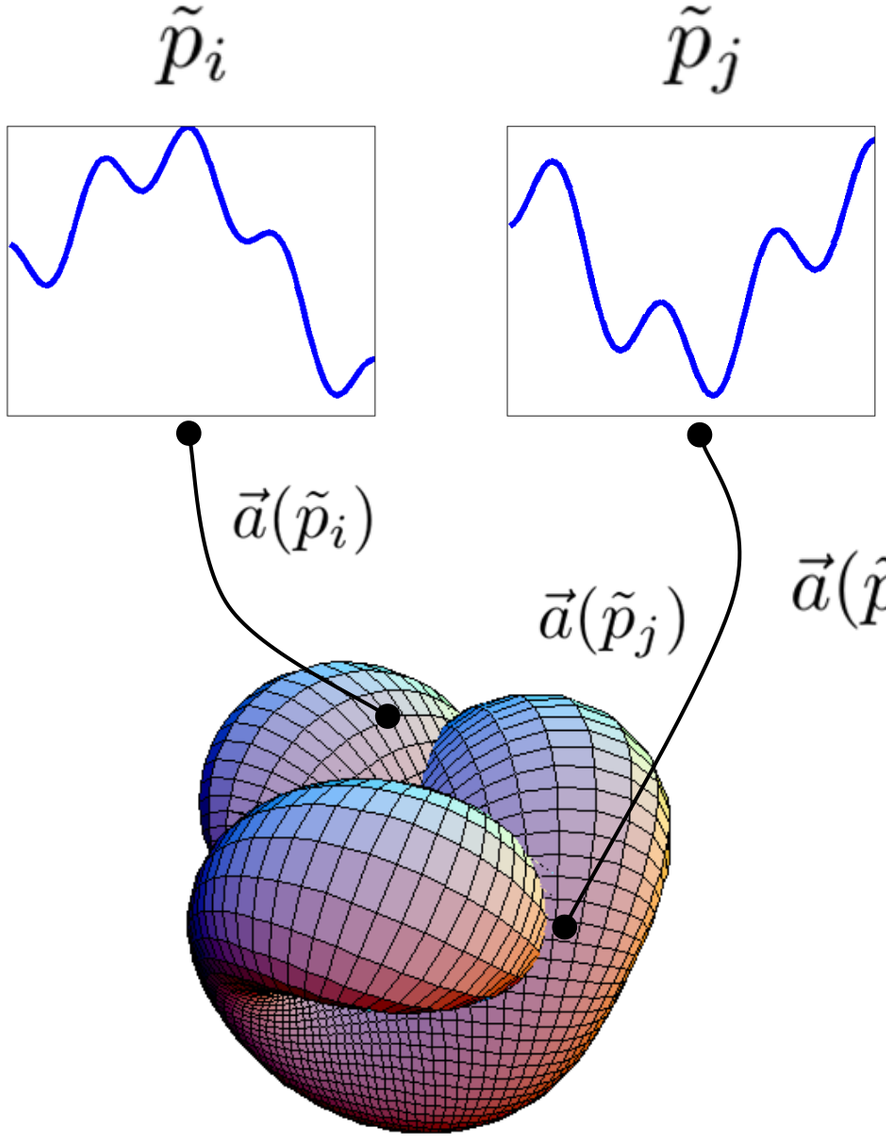
where $a_\alpha(h) = \langle \varphi_\alpha, h\rangle$

$$= \int_{[0,1]^d} \varphi_\alpha(z)h(z)\mathrm{d}z \ \in \mathbb{R}.$$

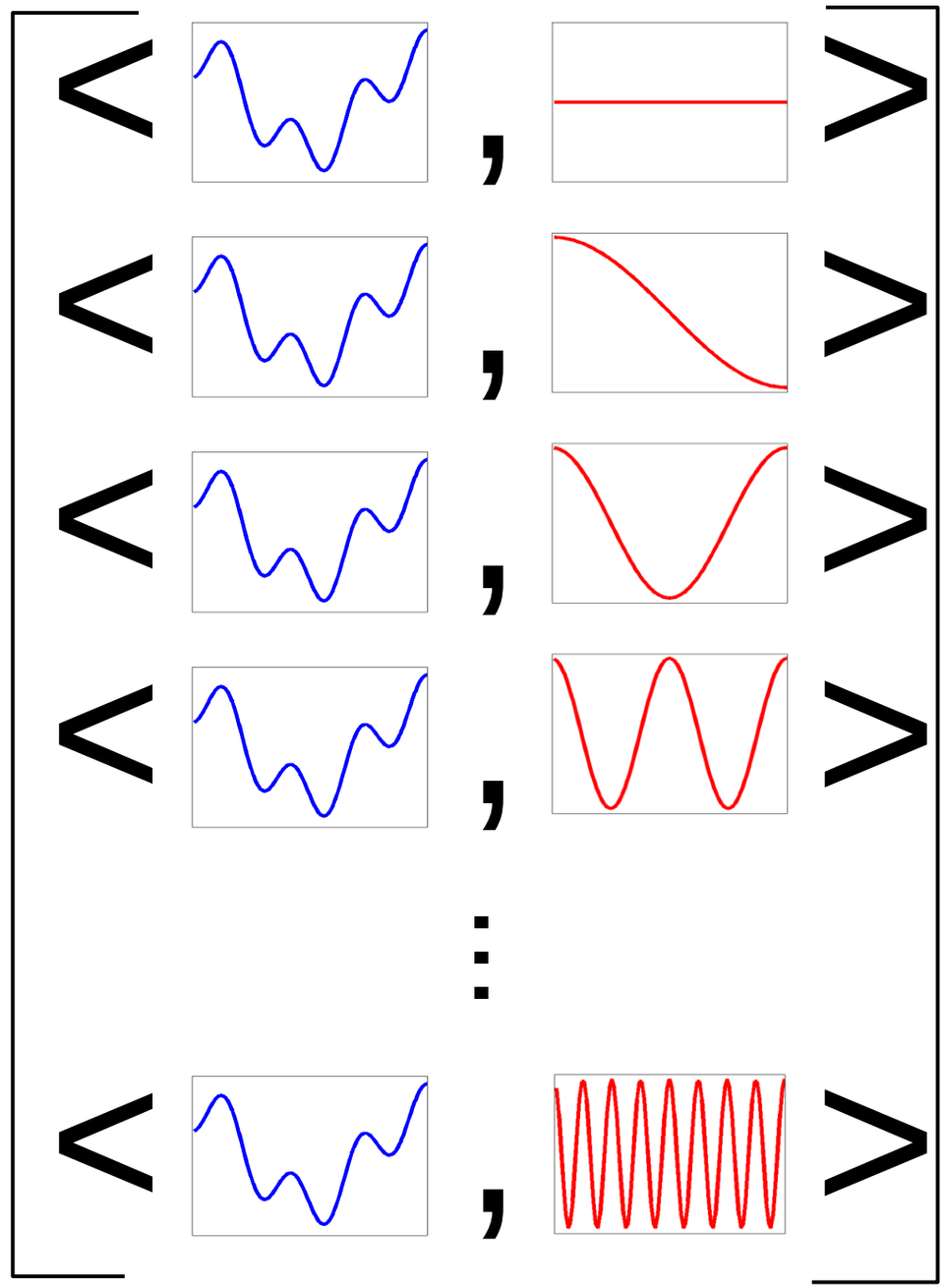


Orthogonal Projection Embedding

Calculating a(p) projection coefficient in the training set (Basis 1):



$$\vec{a}(\tilde{p}_j) = \begin{bmatrix} a_{\alpha_1}(\tilde{p}_j) \\ \vdots \\ a_{\alpha_s}(\tilde{p}_j) \end{bmatrix} =$$

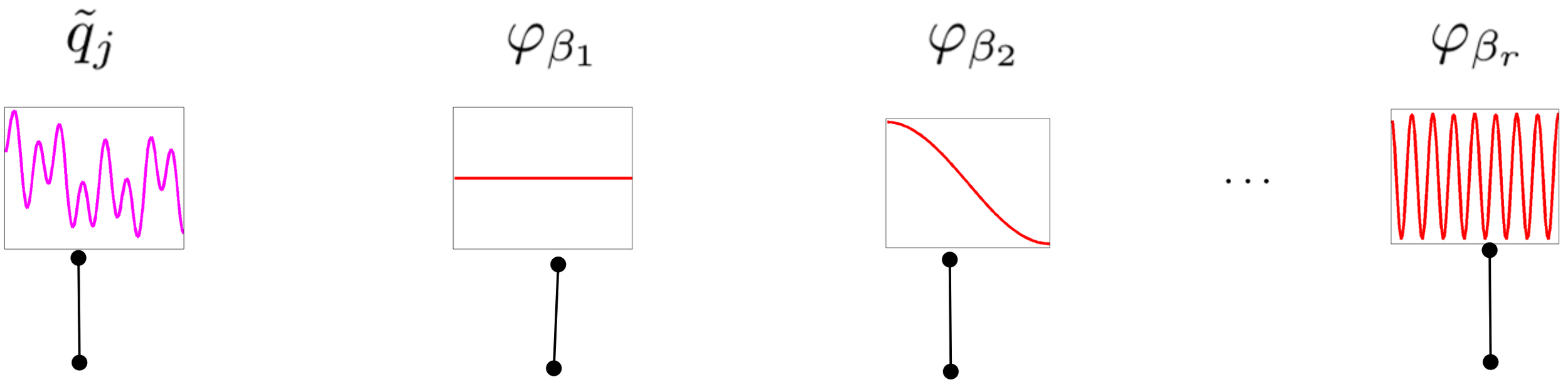


$$\|\vec{a}(\tilde{p}_i) - \vec{a}(\tilde{p}_j)\|_2 = \|\tilde{p}_i - \tilde{p}_j\|_2 = \int (\tilde{p}_i - \tilde{p}_j)^2$$

p ⟶ a(p) ⟶ z(a(p)) ⟶ a(q) ⟶ q

Function from Orthogonal Projections

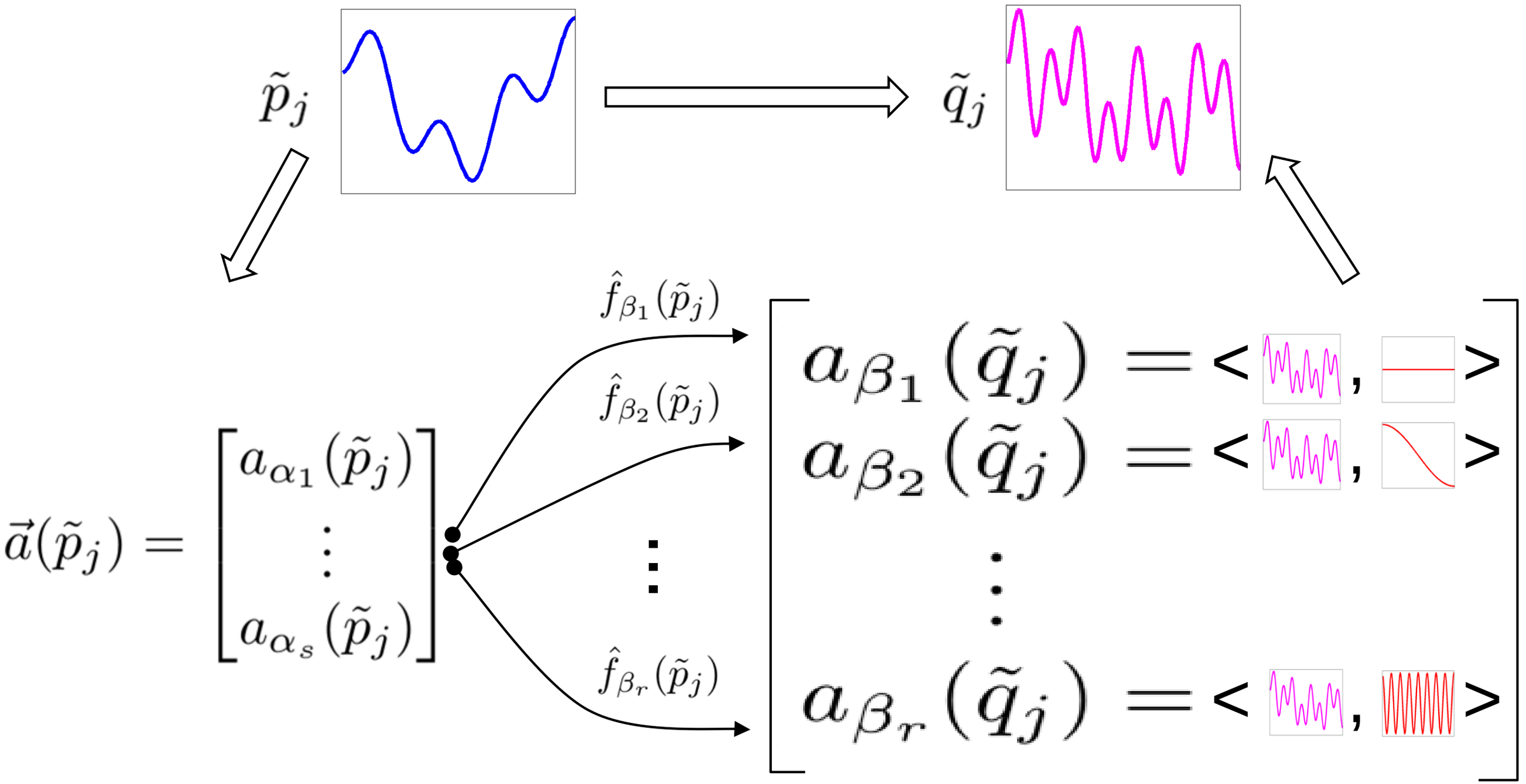
Calculating a(q) projection coefficients in the training set (Basis 2):



$$\tilde{q}_j(x) = \sum_{k=1}^{r} \langle \tilde{q}_j, \varphi_{\beta_k} \rangle \varphi_{\beta_k}(x)$$

p ⟶ a(p) ⟶ z(a(p)) ⟶ **a(q)** ⟶ **q**

Function-to-Function mapping



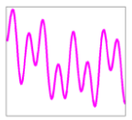
r regression problems (Multi task learning)

p ⟶ a(p) ⟶ z(a(p)) ⟶ a(q) ⟶ q

Function-to-Function mapping

The kth regression problem:

Map $\tilde{p}_j$, (or its projection coefficients $\vec{a}(\tilde{p}_j)$) to the kth projection coefficent of $\tilde{q}_j$, $a_{\beta_k}(\tilde{q}_j)$ (for all $j = 1, \ldots, N$).

$$< \text{  } , \text{  } > \quad = a_{\beta_k}(\tilde{q}_j) \approx \hat{f}_{\beta_k}(\tilde{p}_j) \qquad \vec{a}(\tilde{p}_j) = \begin{bmatrix} a_{\alpha_1}(\tilde{p}_j) \\ \vdots \\ a_{\alpha_s}(\tilde{p}_j) \end{bmatrix}$$

Let us use a kernel smoother for this regression:

$$\hat{f}_{\beta_k}(\tilde{p}_j) = \sum_{i=1}^{N} \theta_i K_\sigma(\|\tilde{p}_i - \tilde{p}_j\|_2)$$

$$= \sum_{i=1}^{N} \theta_i K_\sigma(\|\vec{a}(\tilde{p}_i) - \vec{a}(\tilde{p}_j)\|_2)$$

$$K_\sigma(\|\tilde{p}_i - \tilde{p}_j\|_2) = \exp\left(-\frac{\|\tilde{p}_i - \tilde{p}_j\|_2^2}{2\sigma^2}\right)$$

Random Kitchen Sinks

Theorem [Rahimi & Recht, NIPS 2007]

If one has a shift-invariant kernel K
(in particular we consider the RBF kernel $K(x) = \exp(-x^2/2)$),
then for fixed $\omega_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^{-2} I_d)$, $b_i \overset{iid}{\sim} U([0, 2\pi])$,
we have that for each $x, y \in \mathbb{R}^d$:

$$K(\|x - y\|_2 / \sigma) \approx z(x)^T z(y),$$
$$\text{where } z(x) \equiv \sqrt{\frac{2}{D}} \left[\cos(\omega_1^T x + b_1) \cdots \cos(\omega_D^T x + b_D)\right]^T,$$

and D is the number of random basis functions
(see Rahimi & Recht (NIPS 2007) for approximation quality).

Fast Projection Estimation (Basis 3)

$$< \quad , \quad > = a_{\beta_k}(\tilde{q}_j) \approx \hat{f}_{\beta_k}(\tilde{p}_j)$$

$$K_\sigma(\|\tilde{p}_i - \tilde{p}_j\|_2) = \exp\left(-\frac{\|\tilde{p}_i - \tilde{p}_j\|_2^2}{2\sigma^2}\right)$$

$$\begin{aligned}
\hat{f}_{\beta_k}(\tilde{p}_0) &= \sum_{i=1}^{N} \theta_i K_\sigma(\|\tilde{p}_i - \tilde{p}_0\|_2) \\
&= \sum_{i=1}^{N} \theta_i K_\sigma(\|\vec{a}(\tilde{p}_i) - \vec{a}(\tilde{p}_0)\|_2) \\
&\approx \sum_{i=1}^{N} \theta_i z(\vec{a}(\tilde{p}_i))^T z(\vec{a}(\tilde{p}_0)) \\
&= \left(\sum_{i=1}^{N} \theta_i z(\vec{a}(\tilde{p}_i))\right)^T z(\vec{a}(\tilde{p}_0)) \\
&= \psi_{\beta_k}^T z(\vec{a}(\tilde{p}_0))
\end{aligned}$$

RKS (Rahimi & Recht 2007):

$$K_\sigma(\|\vec{a}(\tilde{p}_i) - \vec{a}(\tilde{p}_j)\|_2) \approx z(\vec{a}(\tilde{p}_i))^T z(\vec{a}(\tilde{p}_j))$$

$$z(\vec{a}(\tilde{p}_j)) = \begin{bmatrix} z_1(\vec{a}(\tilde{p}_j)) \\ \vdots \\ z_D(\vec{a}(\tilde{p}_j)) \end{bmatrix}$$

D dim linear regression!

We have r of these. This is the kth.

where $\psi_{\beta_k} = \sum_{i=1}^{N} \theta_i z(\vec{a}(\tilde{p}_i)) \in \mathbb{R}^D$.

p ⟶ a(p) ⟶ **z(a(p))** ⟶ **a(q)** ⟶ q

Sample Projection Function Estimate

Let us write the r D-dim linear regression in a matrix form on the N training data

$$\vec{a}(\tilde{q}_0) \approx \hat{\Psi}^T z(\vec{a}(\tilde{p}_0)) \quad \text{where}$$

$$\hat{\Psi} \equiv \underset{\Psi \in \mathbb{R}^{D\times r}}{\arg\min} \left\| \begin{pmatrix} — \ \vec{a}(\tilde{q}_1)^T \ — \\ \vdots \\ — \ \vec{a}(\tilde{q}_N)^T \ — \end{pmatrix} - \begin{pmatrix} — \ z(\vec{a}(\tilde{p}_1))^T \ — \\ \vdots \\ — \ z(\vec{a}(\tilde{p}_N))^T \ — \end{pmatrix} \Psi \right\|_F^2$$

$$A \in \mathbb{R}^{N\times D} \qquad Z \in \mathbb{R}^{N\times D}$$

$$\hat{\Psi} \equiv \underset{\Psi \in \mathbb{R}^{D\times r}}{\arg\min} \|A - \mathbf{Z}\Psi\|_F^2 = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T A$$

p ⟶ a(p) ⟶ z(a(p)) ⟶ a(q) ⟶ q

Computational Complexity for Prediction

N : # of instances (input/output pairs)

n : # of sample points per function observation

s : # of projection coefficients for p

r : # of projection coefficients for q

D : # of random basis functions

The total computation time for prediction is $O(rD + Ds + sn)$.

If $D = O(n \log(n))$, $s = O(n)$, and $r = O(n)$, then, the total runtime for evaluating $\widehat{f}(\tilde{p}_0)$ is $O(n^2 \log(n))$.

Since we are considering data-sets where the number of instances N far outnumbers the number of points per sample set n, $O(n^2 \log(n))$ is a substantial improvement over $\Omega(Nn)$ for the LSE.

Risk Upper Bound

$$\mathbb{E}\left[\|q_0 - \hat{q}_0\|_2^2\right]$$
$$\leq O\left(\left(n^{-1/(2+\gamma_{\mathcal{I}}^{-1})} + \frac{n\log(n)\log(N)}{N}\right)^{2/(2+\gamma_{\mathcal{O}}^{-1})}\right)$$

N : # of instances (input/output pairs)

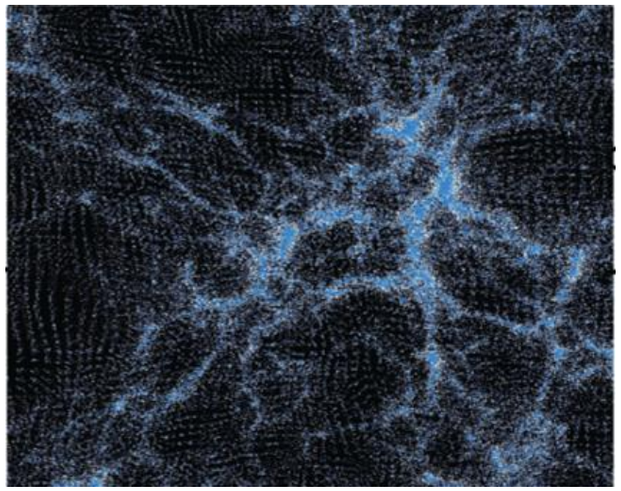
n : # of sample points per function observation

$\gamma_{\mathcal{I}}^{-1}$: input function smoothness parameter

$\gamma_{\mathcal{O}}^{-1}$: output function smoothness parameter

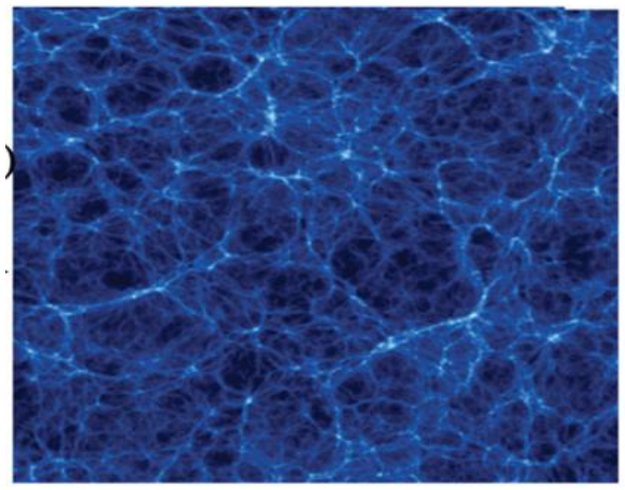
Rectifying 2LPT Simulations
(2nd order Lagrangian Perturbation Theory)

2LPT

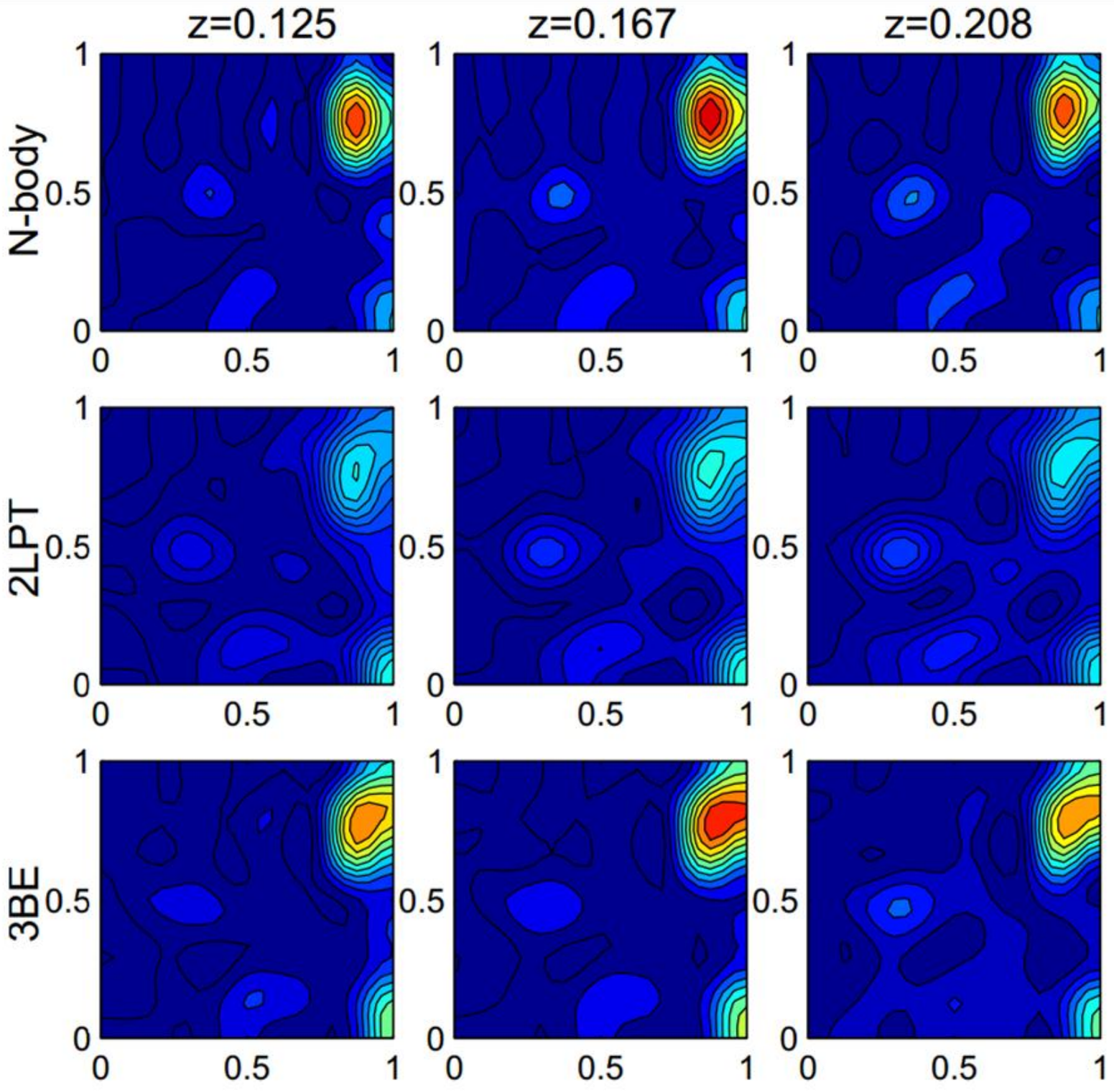


⇩

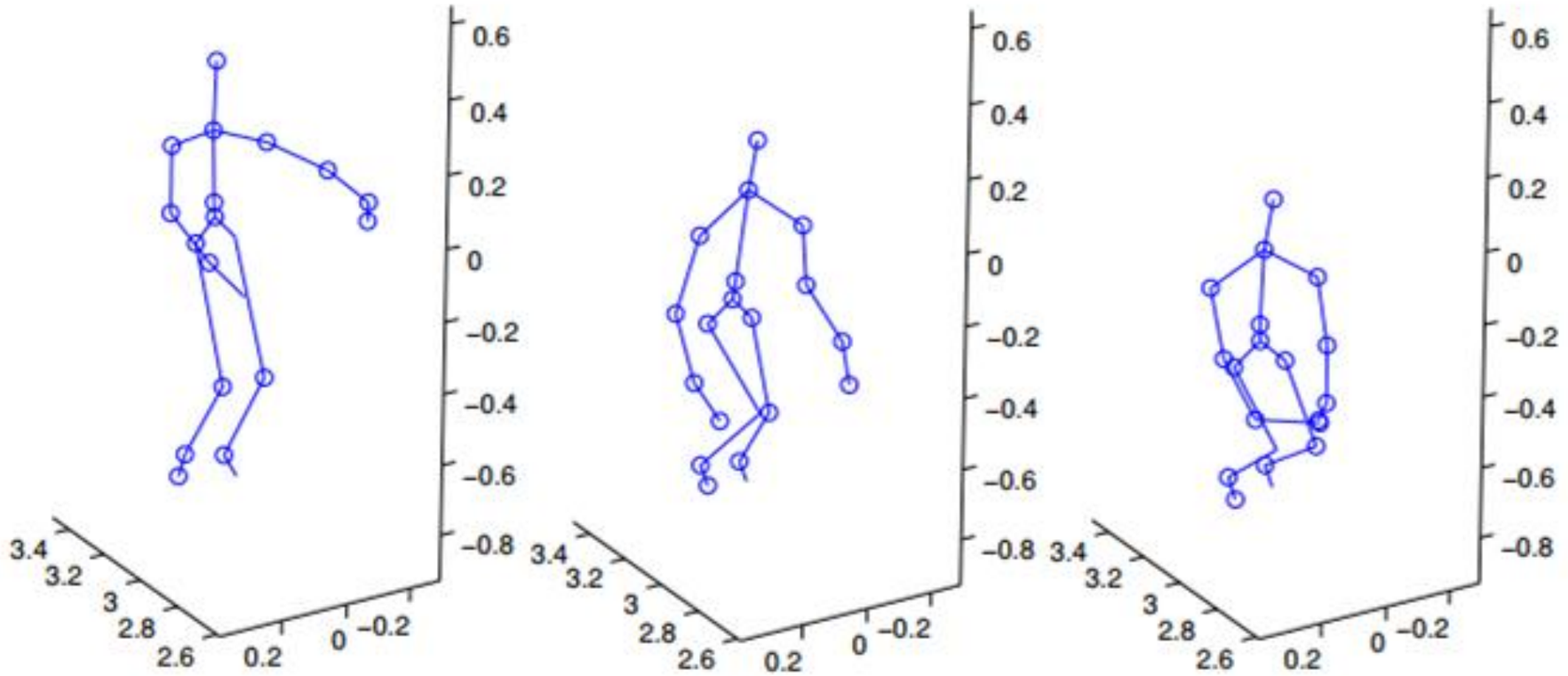
N-body



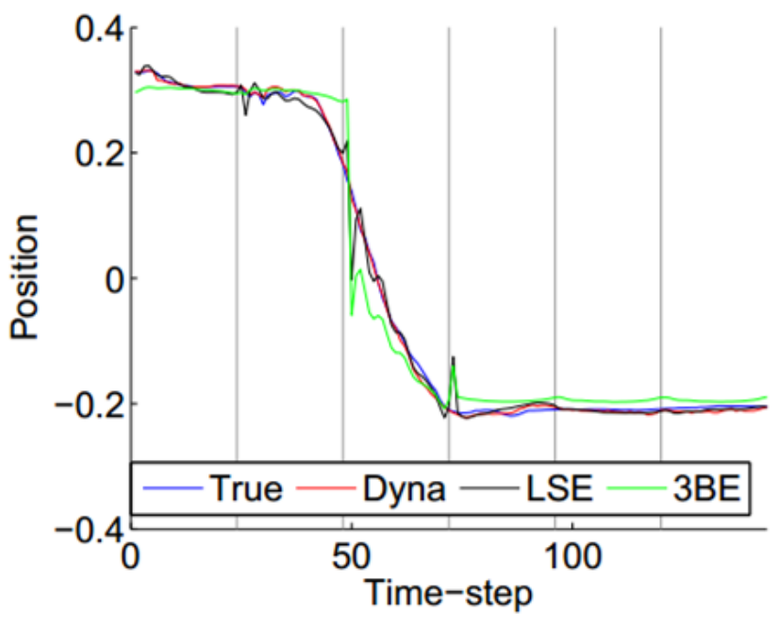
Method	MSE	MPT
3BE	**4.958**	**0.009**
LSE	6.816	4.977
2LPT	6.424	NA



Co-occurring Prediction with MoCap



Method	MSE
3BE	7.78E-4
LSE	1.3E-3
Dyna	2.40E-4



Forward Prediction with Music Data

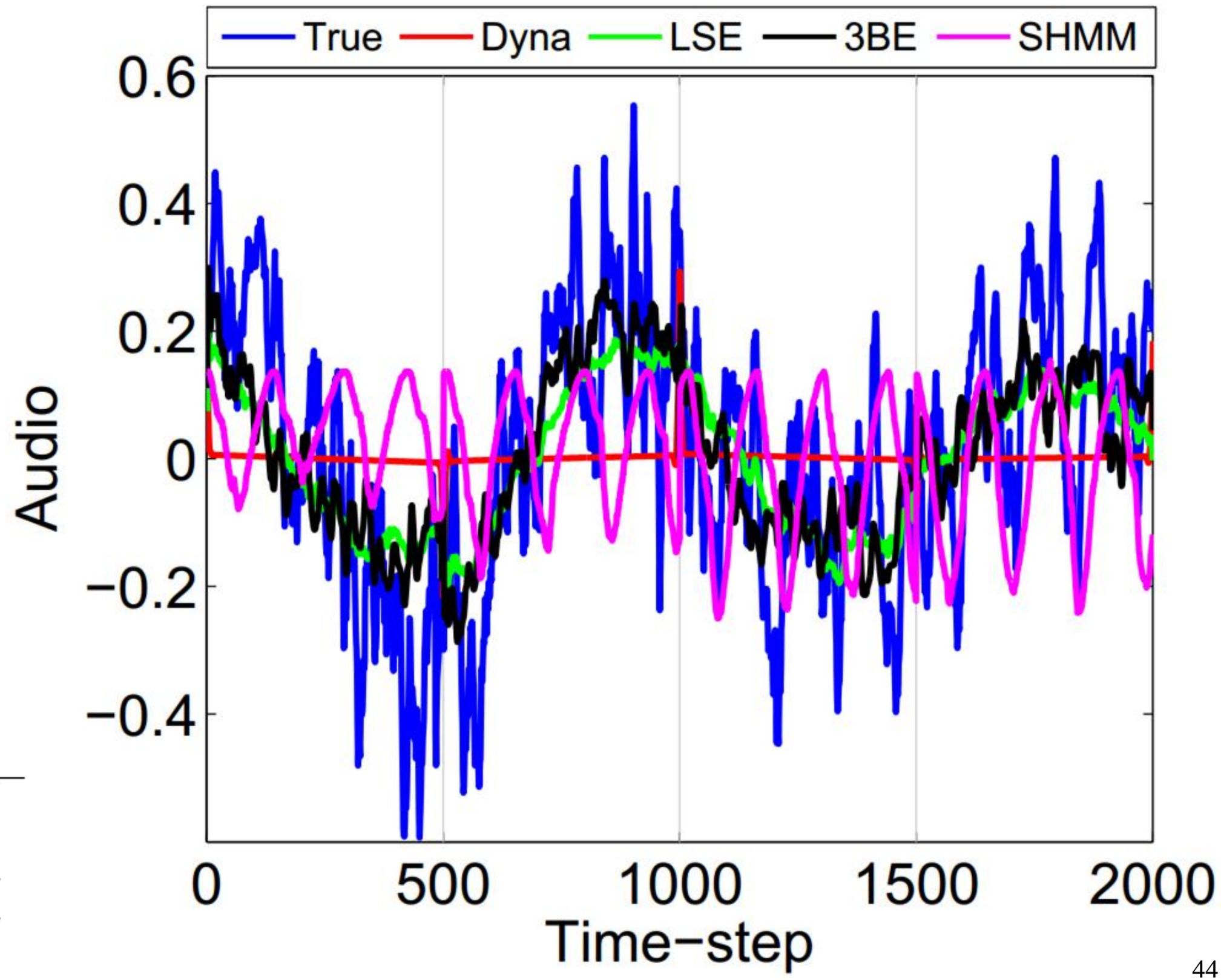
True sound

SHMM

Dyna

LSE

3BE



Method	MSE
3BE	0.0327
LSE	0.0351
Dyna	0.0492
SHMM	0.1082

F2F Learning - Summary

Scalable Function-to-Function regression with Triple Basis Estimator (3BE)

- Orthonormal basis to represent input functions
- Orthonormal basis to represent output functions
- Random basis mapping input functions to output functions

FuSSO = Functional Shrinkage and Selection Operator (Functional Lasso)

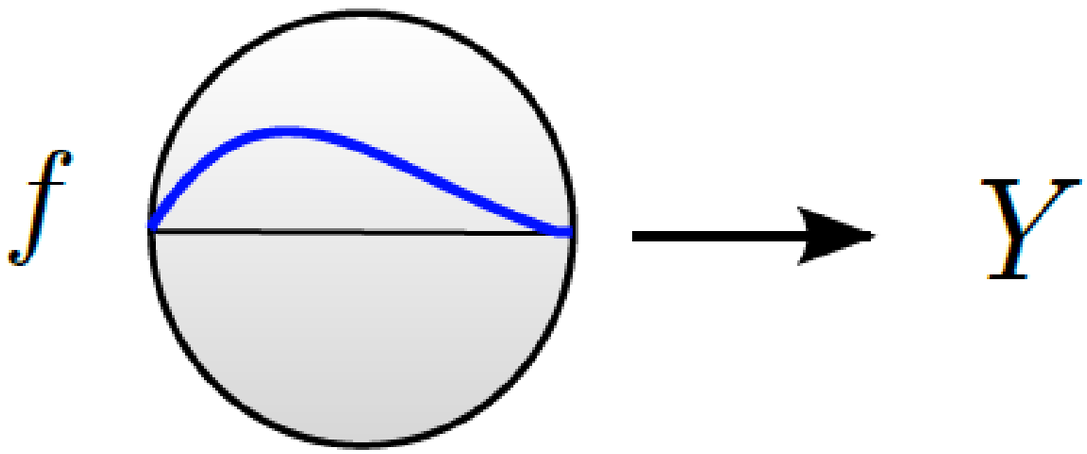
FuSSO

We present the FuSSO, a functional analogue to the LASSO:

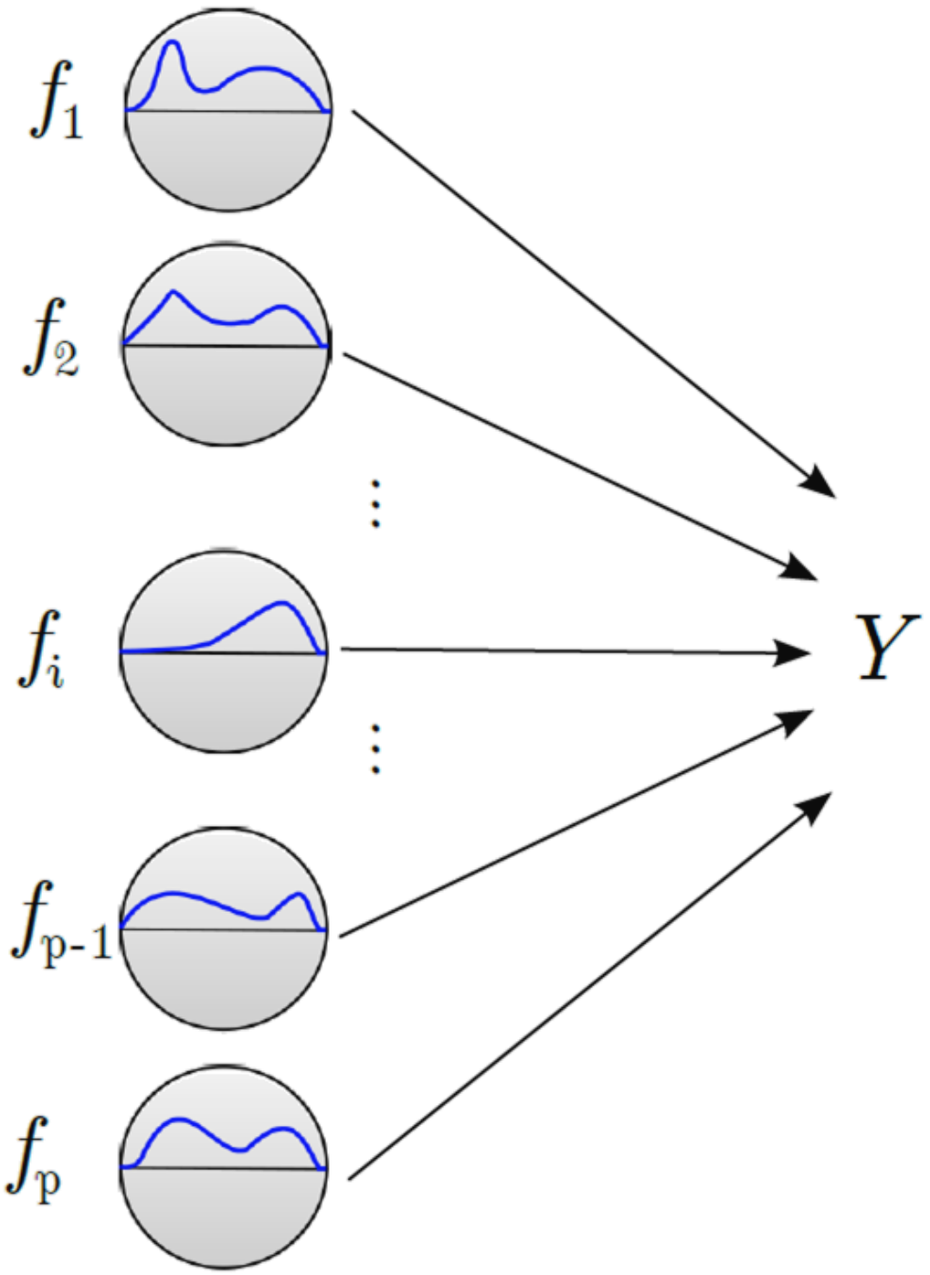
- Finds a sparse set of functional input covariates to regress a real-valued response
- The FuSSO is semi-parametric: No parametric assumptions on input functional covariates
- Assumes linear form to response from functional covariates
- We provide a statistical backing for use of the FuSSO via proof of asymptotic sparsistency under various conditions
- Furthermore, we observe good results on both synthetic and real-world data

Real Valued Regression with Multiple Functional Covariates

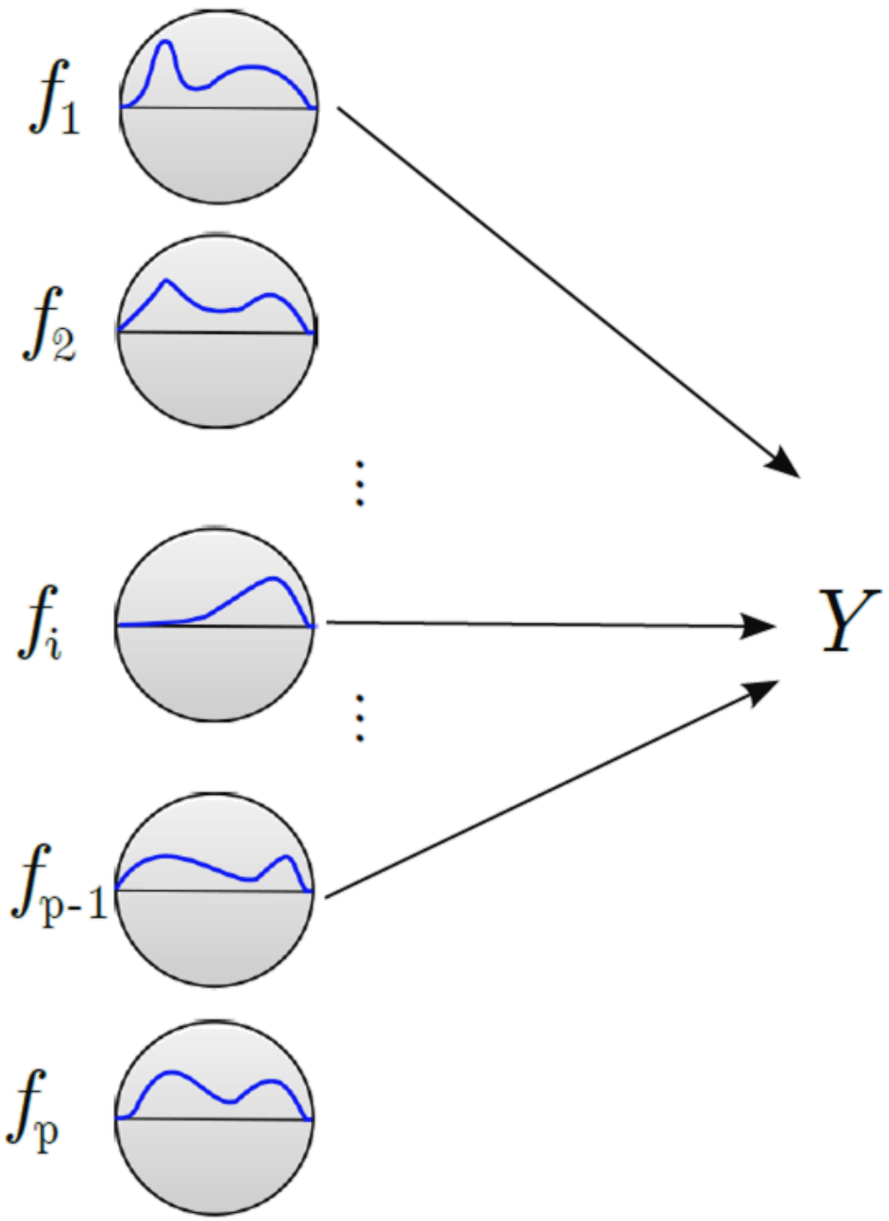
Function to real regression is a previously studied problem (e.g. in functional analysis); here the mapping takes in one function:



Similarly, one may consider a mapping that takes in multiple functions:

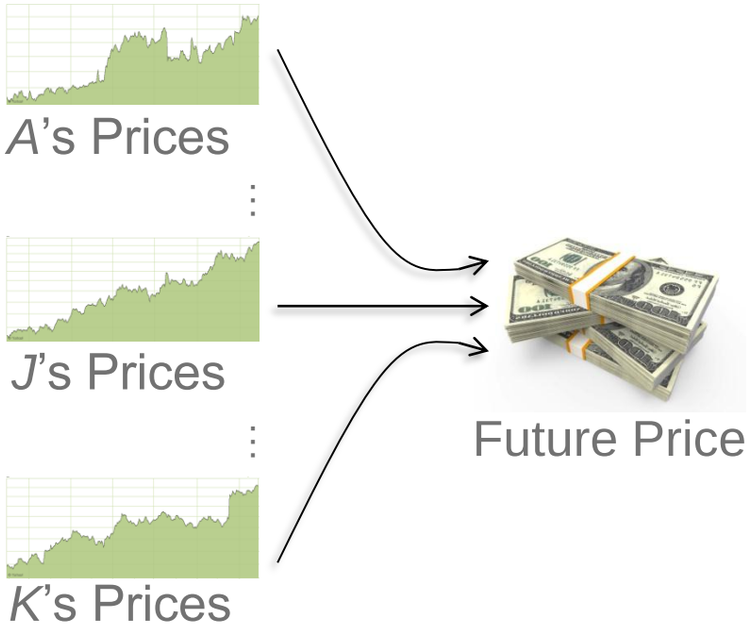


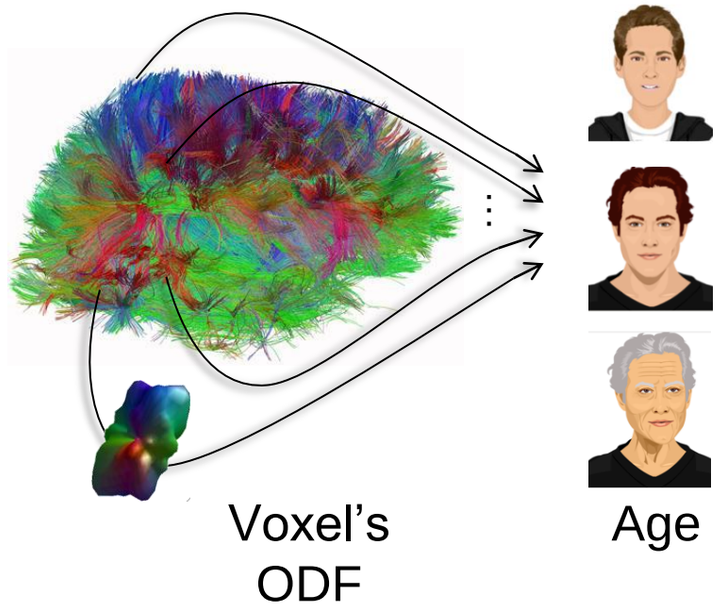
The number of functional input covariates may be very large; thus, a sparse model that depends only on a few of the functional covariates may be preferred:



Example Applications

- Finance:
  - *Inputs:* Time-series of several commodities prices in the past
  - *Output:* Price of a particular commodity in the nearby future
- Neuroscience:
  - *Inputs:* Functions at each voxel (e.g. orientation distribution functions)
  - *Output:* The age of the subject


A's Prices
J's Prices
K's Prices
Future Price


Voxel's
ODF
Age

Linear Functional Regression

There are functional analogues to familiar real-vector linear regression models; for $Y_i \in \mathbb{R}$, $\epsilon_i \sim \mathcal{N}(0, \sigma)$, and $\Psi \subseteq \mathbb{R}^k$, a compact set:

One Real Vector vs. Functional Covariate:

Real Vector Covariate	**Functional Covariate**
$Y_i = \langle X_i, w \rangle + \epsilon_i$	$Y_i = \langle f^{(i)}, g \rangle + \epsilon_i$
where	
$X_i, w \in \mathbb{R}^d$ and $\langle X_i, w \rangle = \sum_{j=1}^{d} X_{ij} w_j$	$f^{(i)}, g \in L_2(\Psi)$ and $\langle f^{(i)}, g \rangle = \int_\Psi f^{(i)}(t) g(t) \mathrm{d}t$

Similarly, instead of having one input covariate per data instance, one may have p feature vectors, $\{X_{i1}, \ldots, X_{ip}|,\ X_{ij} \in \mathbb{R}^d\}$, or functions, $\{f_1^{(i)}, \ldots, f_p^{(i)} \mid f_j^{(i)} \in L_2(\Psi)\}$, associated to each data instance i:

Multiple Real Vectors vs. Functional Covariates:

Real Vector Covariates		**Functional Covariates**
$Y_i = \sum_{j=1}^{p} \langle X_{ij}, w_j \rangle + \epsilon_i$	where	$Y_i = \sum_{j=1}^{p} \langle f_j^{(i)}, g_j \rangle + \epsilon_i$
$w_1, \ldots, w_p \in \mathbb{R}^d$		$g_1, \ldots, g_p \in L_2(\Psi)$

FuSSO: Functional Shrinkage and Selection Operator

Regression with Basis Functions:

Let $\Psi = [0,1]$, $\Phi = \{\varphi_m\}_{m=1}^{\infty}$ be an orthonormal basis for $L_2(\Psi)$:

$$f_j^{(i)}(x) = \sum_{m=1}^{\infty} \alpha_{jm}^{(i)} \varphi_m(x), \text{ where } \alpha_{jm}^{(i)} = \int_0^1 f_j^{(i)}(t)\varphi_m(t)\mathrm{d}t$$

Similarly $g_j(x) = \sum_{m=1}^{\infty} \beta_{jm}^{*} \varphi_m(x)$, then by orthonormality:

$$\langle f_j^{(i)}, g_k \rangle = \sum_{m=1}^{\infty} \alpha_{jm}^{(i)} \beta_{km}^{*}$$

Let $\{\tilde{\alpha}_{jm}\}_{m=1}^{M_n}$ approximate $\{\alpha_{jm}\}_{m=1}^{M_n}$, and

$$f_j^{(i)}(x) \approx \tilde{f}_j^{(i)}(x) = \sum_{m=1}^{M_n} \tilde{\alpha}_{jm}^{(i)} \varphi_m(x)$$

Optimization Problem:

We may then estimate $g_j = \sum_{m=1}^{\infty} \beta^*_{jm} \varphi_m$ as $\hat{g}_j = \sum_{m=1}^{M_n} \hat{\beta}_{jm} \varphi_m$ where

$$\hat{\beta} \equiv \text{argmin}_{\beta} \frac{1}{2N} \sum_{i=1}^{N} \left(Y_i - \sum_{j=1}^{p} \sum_{m=1}^{M_n} \tilde{\alpha}^{(i)}_{jm} \beta_{jm} \right)^2 + \lambda_N \sum_{j=1}^{p} \sqrt{\sum_{m=1}^{M_n} \beta^2_{jm}}$$

Theory: Sparsistency

$\mathbb{P}\left(\text{supp}(\hat{\beta}) = \text{supp}(\beta^*)\right) \to 1$ when given a dataset $\mathcal{D} = \{(\{\vec{y}_j^{(i)}\}_{j=1}^p, Y_i)\}_{i=1}^N$, where

$$\vec{y}_j^{(i)} = \vec{f}_j^{(i)} + \xi_j^{(i)},$$
$$\vec{f}_j^{(i)} = \left(f_j^{(i)}(1/n),\ f_j^{(i)}(2/n),\ \ldots,\ f_j^{(i)}(1)\right)^T,$$
$$\xi_j^{(i)} \overset{iid}{\sim} \mathcal{N}(0, \sigma_\xi^2 I_n),$$

and $f_j^{(i)}, g_j : [0,1] \mapsto \mathbb{R}$. The proof is generalizable to higher dimensions, and when $f_j^{(i)}$ are pdfs and one observes samples.

Results

Synthetic Dataset:

- Tested on datasets of $\mathcal{D} = \{(\{\vec{y}_j^{(i)}\}_{j=1}^{p}, Y_i)\}_{i=1}^{N}$
  - $\vec{y}_j^{(i)}$ grid of n noisy grid function evaluations
- The number of true functional covariates in the support, s, was fixed to be 5
- Functions $f_j^{(i)}$ and g_j were generated randomly

For each experiment we ran 100 trials and recorded the following:

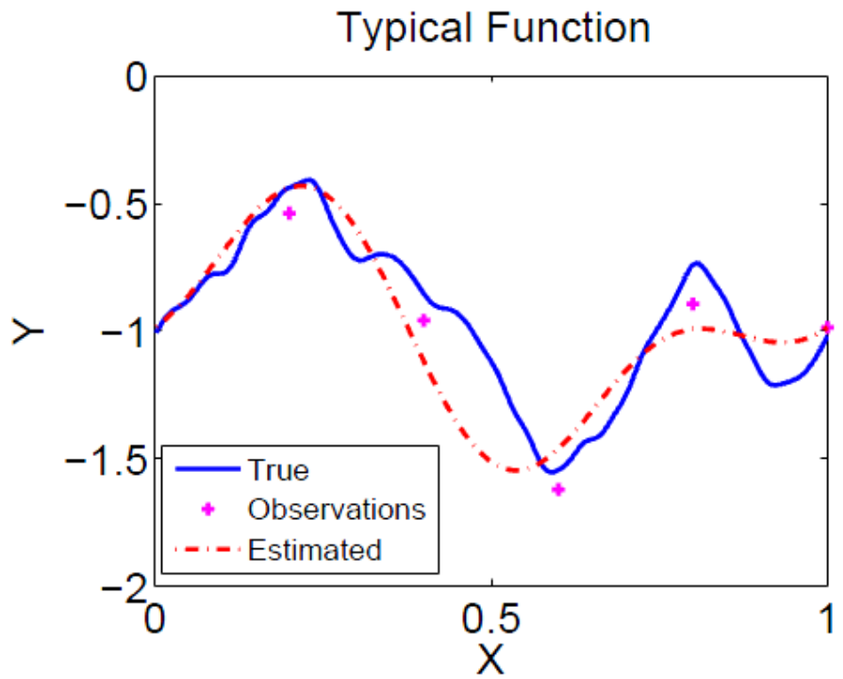
Experiments' Variables:

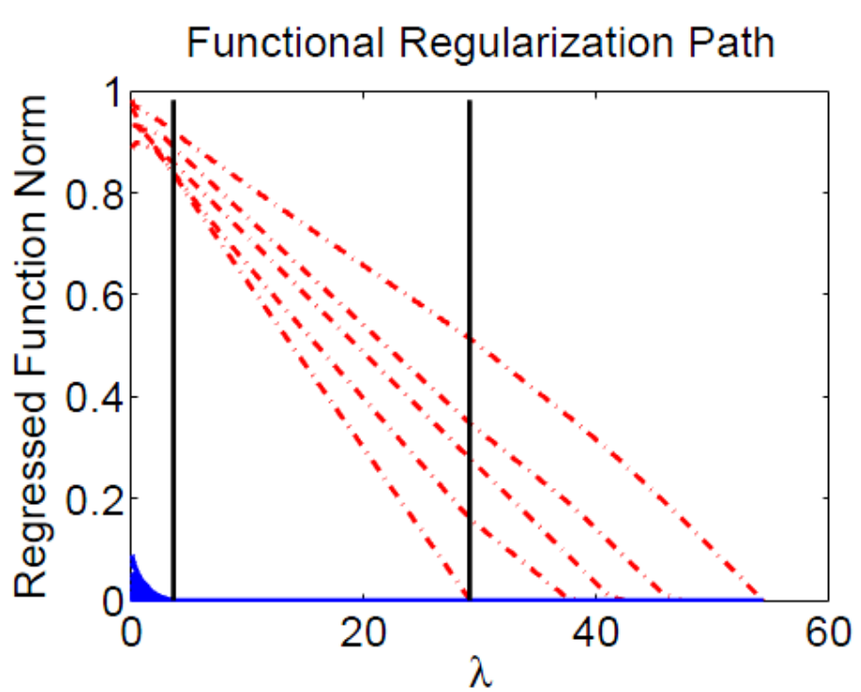
Name	Definition
p	# of input covariates
N	# of data instances
n	# of grid evaluations
r	% of trails w/ support found
Δ	avg. range of λ w/ support

Experiments' Results:

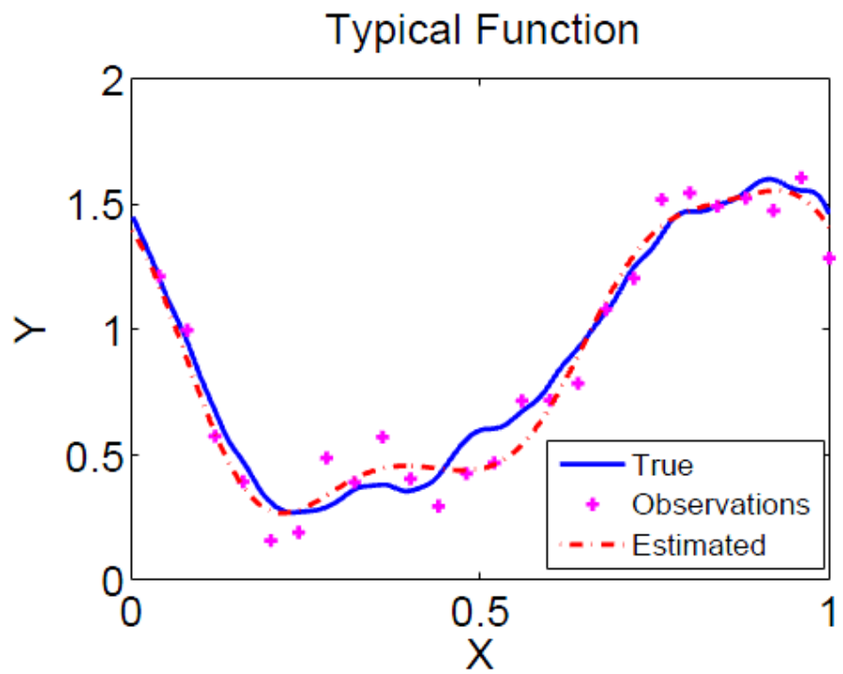
(p, N, n)	r	Δ
(100,50,5)	.68	.2125
(1000,500,25)	1	.4771
(20000,500,25)	1	.4729

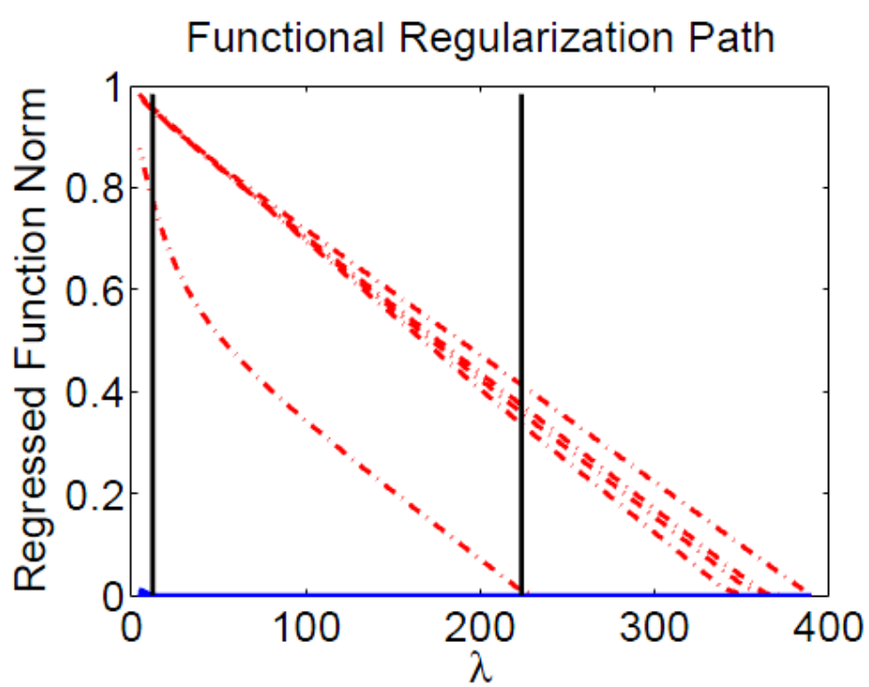
Figures for (*p*,*N*,*n*) = (100,50,5):





Figures for (*p*,*N*,*n*) = (1000,500,25):





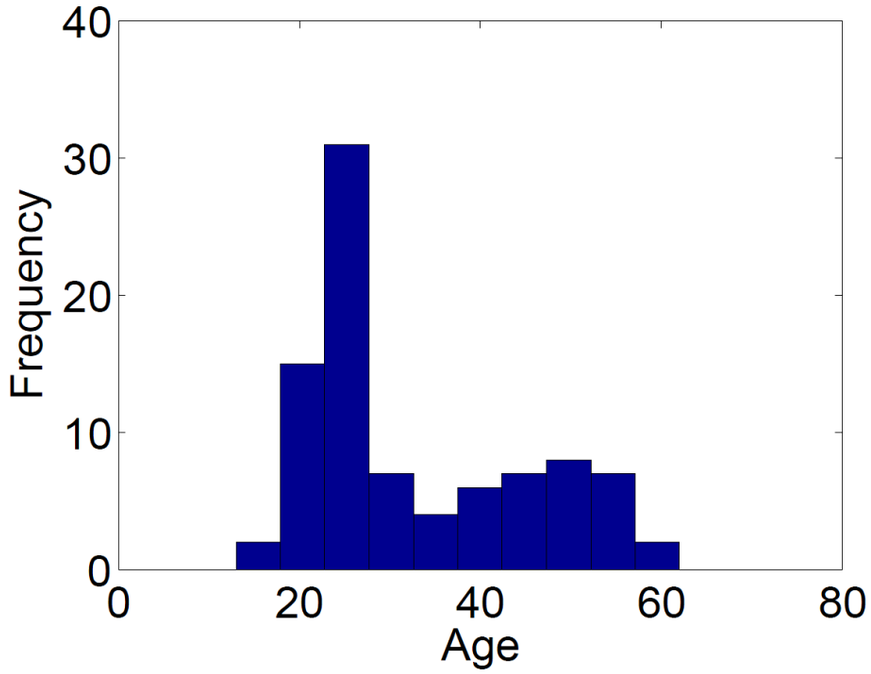
Neurological Dataset:

- Dataset with over 25K functions per subject for 89 total subjects
  - Orientation distribution functions (ODF) at white matter voxels
- We regress a subject's age, given ODFs
- We compared to using the LASSO with peak ODF (quantitative anisotropy) values

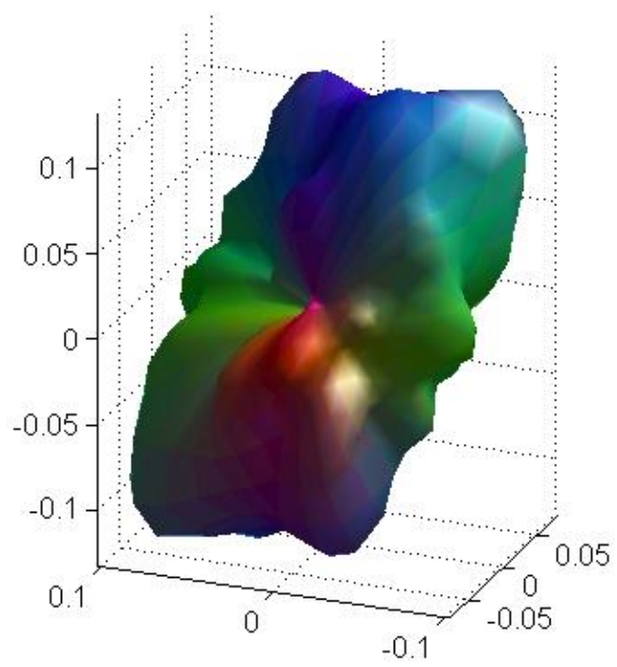
Results

Method:	FuSSO (ODFs)	LASSO (QAs)	Mean Predict
MSE:	70.85	77.13	156.43

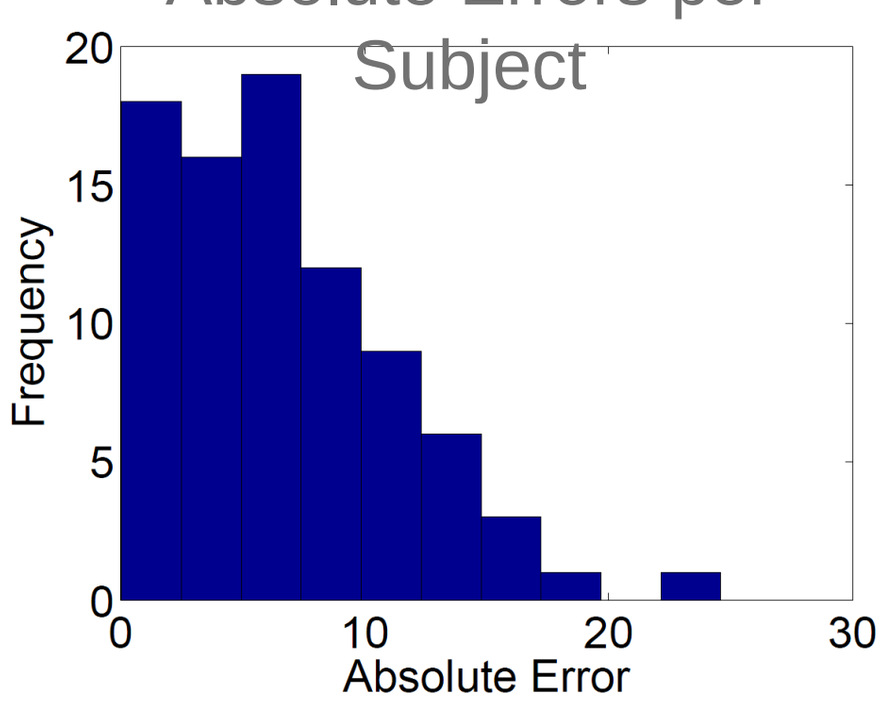
Ages



Voxel ODF



Absolute Errors per Subject



Bootstrap Selected Voxels

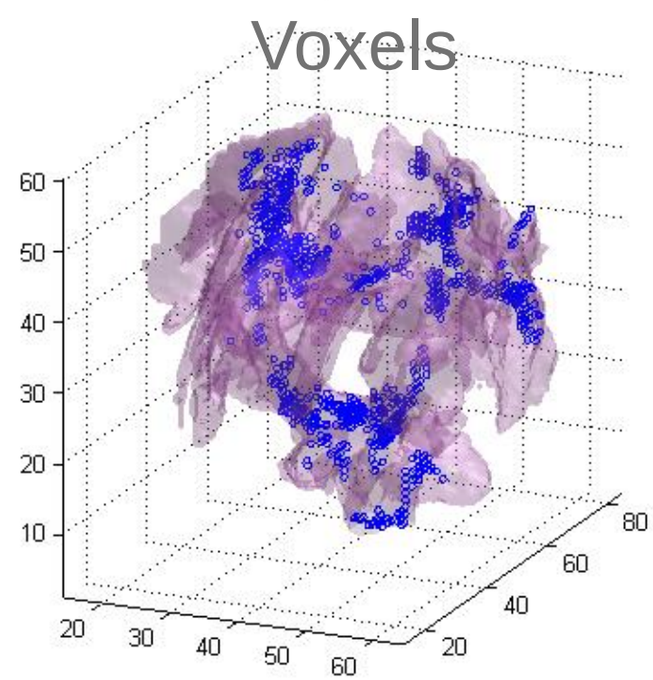
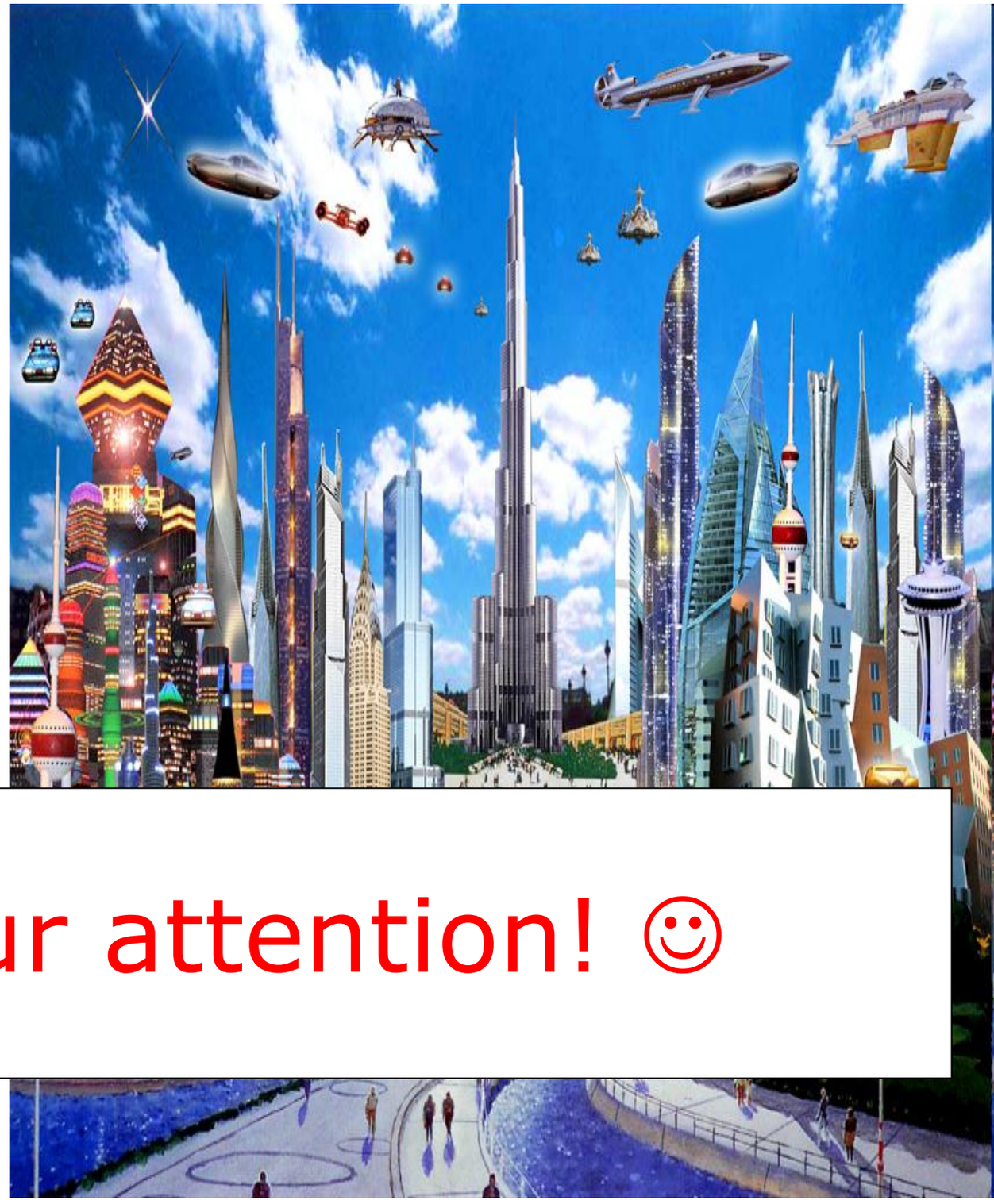


Image Sources: http://www.aging2.com/wp-content/uploads/2013/05/Screen-Shot-2013-05-28-at-9.48.49-PM.png; http://media.salon.com/2013/02/money1.jpg; http://3278as3udzze1hdk0f2th5nf18c1.wpengine.netdna-cdn.com/wp-content/uploads/2010/10/connectome-brain-diffusion-spectrum-imaging.jpg

If you are interested, contact me! ☺

bapoczos@cs.cmu.edu, GHC-8231

Functional data has so many applications!



Some results on regression/classification/anomaly detection/ Lasso

Lots of missing theoretical results: Lower bounds, active learning

Thanks for your attention! ☺