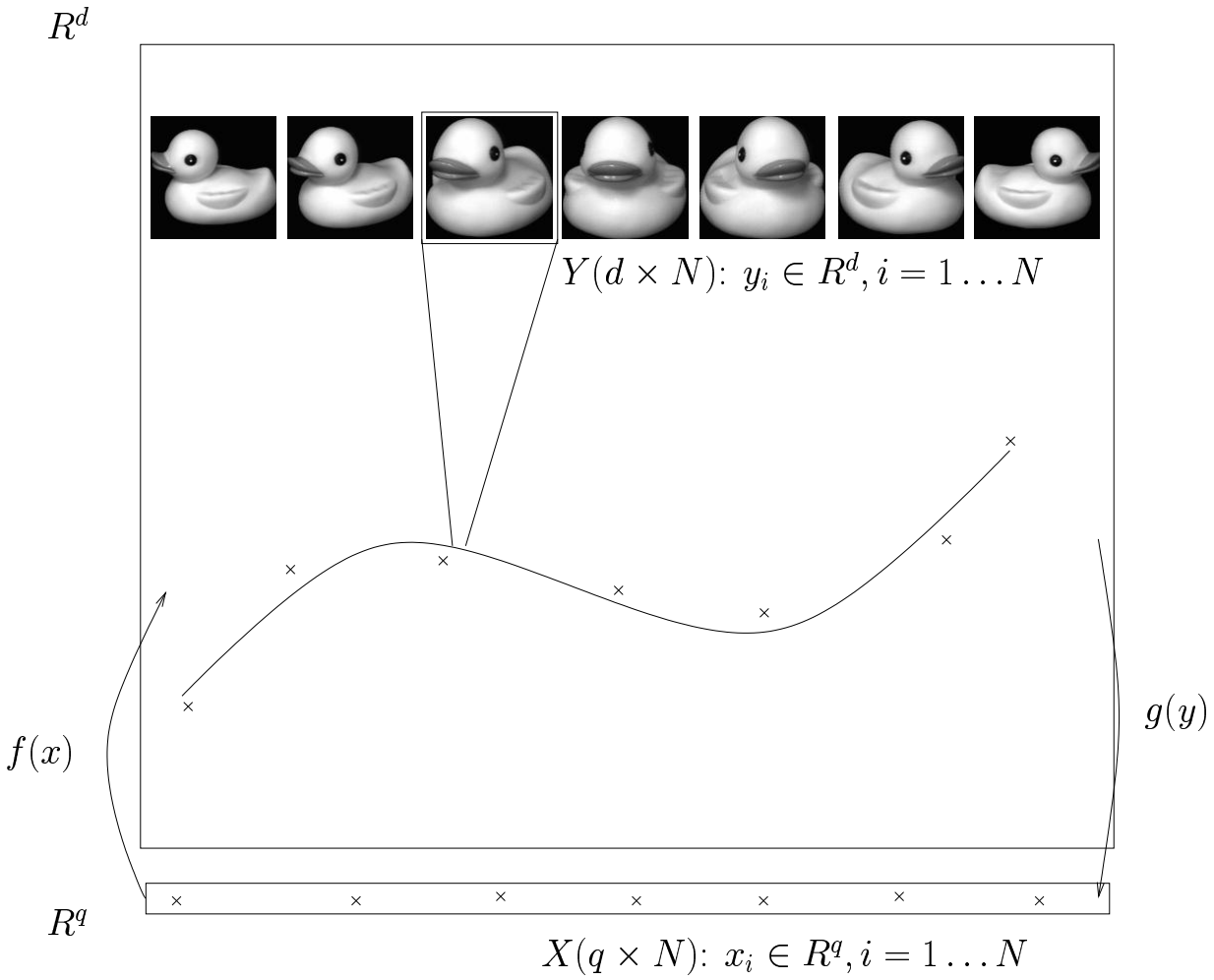
Unsupervised Kernel Regression

Roland Memisevic

February 9, 2004

Joint work with Peter Meinicke and Stefan Klanke


R^d
$Y(d \times N)$: $y_i \in R^d, i = 1 \ldots N$
$f(x)$
$g(y)$
R^q
$X(q \times N)$: $x_i \in R^q, i = 1 \ldots N$

Unsupervised Learning as Generalized Regression

$$y = f^*(x;\theta) + u, \quad E(u) = \mathbf{0}$$

- Now call x 'latent'
- Need to solve for both x and θ.
- $\rightarrow$ iterate:
  - Projection - Regression
  - EM (For true random variable x)
- Most often we choose: $f(x) = Wb(x)$

Idea: Use Non-parametric Regression Function

Consider:

$$f(x) = E(y|x) = \frac{\int yp(x,y)dy}{\int p(x,y)dy}$$

Approximate $p(x,y)$ with Kernel Density Estimate $\hat{p}(x,y)$:

$$\hat{p}(x,y) = \frac{1}{N}\sum_{j=1}^{N} \mathcal{K}_q(x,x_j)\mathcal{K}_d(y,y_j),$$

for some (isotropic, ...) kernel-functions $\mathcal{K}_q$ and $\mathcal{K}_d$.

Then:

Nadaraya-Watson Estimator

$$f(x) = \sum_{j=1}^{N} \frac{K(x, x_j)}{\sum_{k=1}^{N} K(x, x_k)} \, y_j =: Yb(x)$$

with (un-normalized) kernel functions $K(\cdot, \cdot)$, e.g. RBF:

$$K(x_i, x_j) = \exp(\frac{1}{2h_l^2} \|x_i - x_j\|^2)$$

Note: model complexity <—> Kernel bandwidth h_l

Now call x latent...

Unsupervised Kernel Regression (1)

...and minimize the **data-space reconstruction error**:

$$E^{\text{obs}}(X) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - f(x_i)\|^2$$

$$= \frac{1}{N}\sum_{i=1}^{N} \|y_i - \sum_{j=1}^{N} \frac{K(x_i, x_j)}{\sum_{k=1}^{N} K(x_i, x_k)}\, y_j\|^2$$

Set $h_l = 1.0$

model complexity <—> scale of X

Unsupervised Kernel Regression (1)

Learning with gradient descent.

Complexity: $O(N^2 dq)$ for both E and $\frac{\partial E}{\partial X}$

Unsupervised Kernel Regression (1)

Regularization (a): 'Weight Decay':

instead of E^{obs} minimize

$$E^P(X) = E^{\text{obs}}(X) + \lambda P(X),$$

for some penalty $P(X)$, e.g.

$$P(X) = \|X\|_F^2, \text{ or } P(X) = -\sum_{i=1}^{N} \hat{p}(x_i)$$

$$\hat{X} = \arg\min_X E^P(X)$$

Unsupervised Kernel Regression (1)

Regularization (b): Constrain the solution:

$$\hat{X} = \arg\min_{X} E^{\mathrm{obs}}(X),$$

$$\text{subject to} \quad c(X) <= 0$$

... or subject to bound constraints.

Unsupervised Kernel Regression (1)

Regularization (c): Crossvalidation

Use the modified objective function:

$$E^{\mathrm{cv}} = \frac{1}{N} \sum_{i=1}^{N} \| y_i - f_{-i}(x_i) \|^2$$

$$:= \sum_{i=1}^{N} \| y_i - \sum_{j \neq i} \frac{K(x_i, x_j)}{\sum_{k \neq i} K(x_i, x_k)} y_j \|^2$$

'Built in' Leave-One-Out Crossvalidation !

Unsupervised Kernel Regression (1)

To avoid local minima use 'Deterministic Annealing':

Begin with strong regularization.

Then annealing.

$\longrightarrow$ Regularization variants (a) und (b).

Unsupervised Kernel Regression (1)

After training:

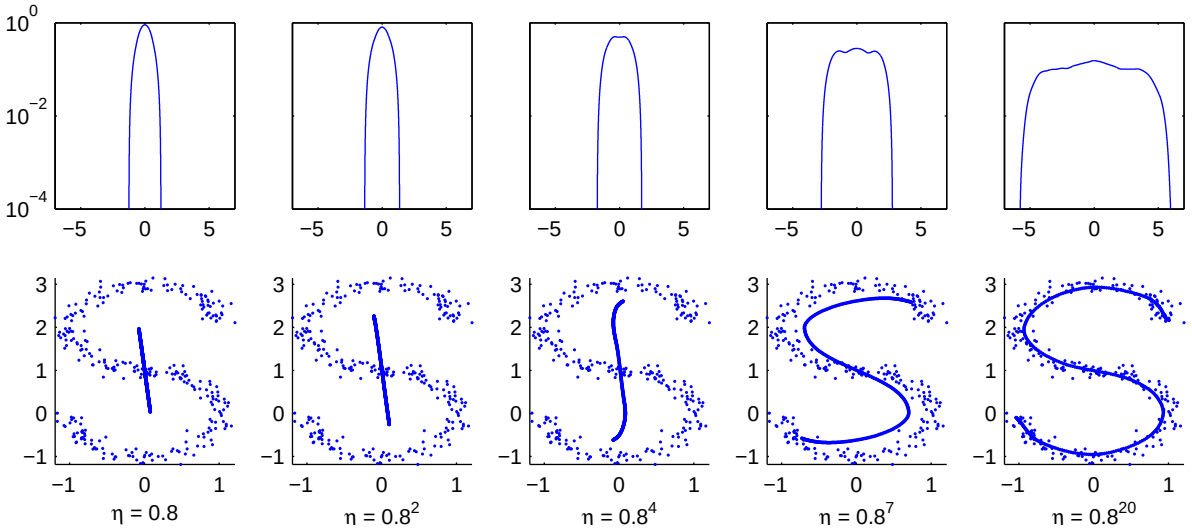
- Learned X.
- Learned 'Forward mapping' implicitly.
- Define 'Backward mapping' by orthogonal projection:

$$g(y) := \arg\min_x \|y - f(x)\|_2^2$$

(Initialize with 'nearest reconstruction'

$$x_0 := \arg\min_{x_i} \|y - f(x_i)\|_2^2, \quad i = 1, \ldots, N)$$

Unsupervised Kernel Regression (1)



Discussion UKR(1)

- Solves NLDR-problem 'completely'
- Principled
- 'Built in' regularizer
- Arbitrary latent space dimensionalities
- Latent space *density* (Consider non-uniformly distributed data!)
- Introduction of prior knowledge

Unsupervised Kernel Regression (2)

Idea:

Turn Nadaraya-Watson Estimator upside down...

Instead of $E(y|x)$ estimate $E(x|y)$

$$g(y) := \sum_{j=1}^{N} \frac{K(y, y_j)}{\sum_{k=1}^{N} K(y, y_k)} x_j$$

Justification? E.g. 'Noisy Interpolation' (Webb, 1994).

Unsupervised Kernel Regression (2)

$\longrightarrow$ Objective function now:

$$E^{\text{lat}}(X) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - g(y_i)\|^2,$$

$$= \frac{1}{N} \sum_{i=1}^{N} \|x_i - \sum_{j=1}^{N} \frac{K(y_i, y_j)}{\sum_{k=1}^{N} K(y_i, y_k)} x_j\|^2,$$

$$=: \frac{1}{N} \|X - XB(Y)\|_F^2$$

Unsupervised Kernel Regression (2)

Closed form solution, because

$$E^{\text{lat}}(X) = \frac{1}{N}\text{tr}(X(\mathbf{I}_N - B(Y))(\mathbf{I}_N - B(Y))^T X^T)$$

$$= \frac{1}{N}\text{tr}(XQX^T)$$

with

$$Q := (\mathbf{I}_N - B(Y))(\mathbf{I}_N - B(Y))^T$$

Have to minimize quadratic form Q!

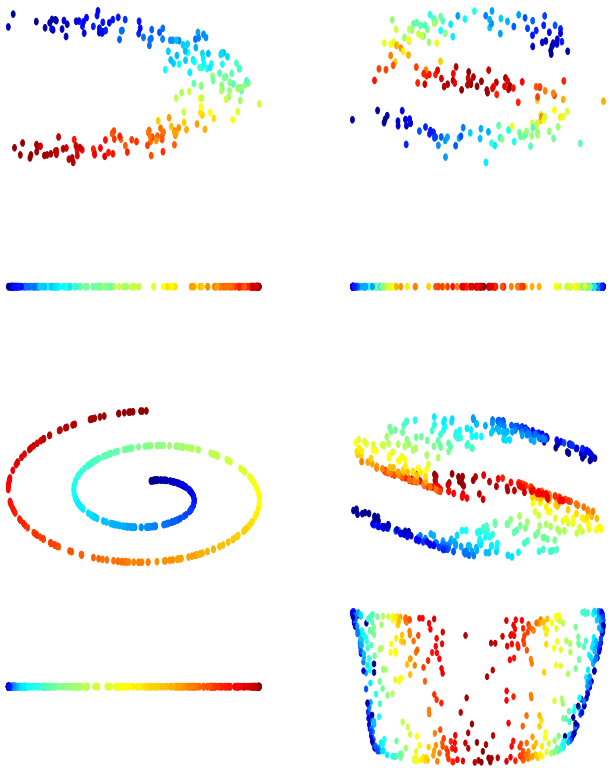
$\rightarrow$ Constraining: $X\mathbf{1}_N = \mathbf{0}$ and $XX^T = \mathbf{I}_q$, **Solution by EVD.**

Unsupervised Kernel Regression (2)

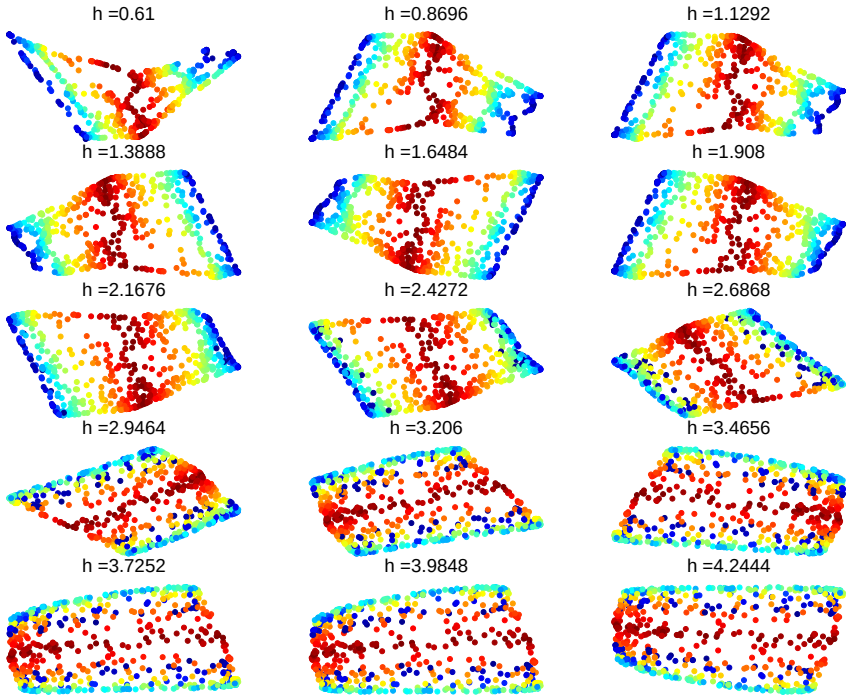
After training:

- Learned X.
- Learned 'Backward mapping' implicitly.
- ...but not f

Unsupervised Kernel Regression (2)



Unsupervised Kernel Regression (2)



correct Kernel Bandwidth!?

Discussion UKR(2)

- Learns only X and g
- 'Yet another Spectral Method' - however, from a regression perspective!
- Generalizes to new, unseen (observable space) data
- Fast
- Can provide useful initializations...

Unsupervised Kernel Regression (2+1)

A way to combine the advantages of both methods:

Use UKR(2) as an initialization for UKR(1).

Problem: UKR(2) solution will be destroyed after a few gradient steps.

Idea: minimize UKR(1)-error with respect to scale of X:

$$E(S) = \sum_{i=1}^{N} \| y_i - \sum_{j \neq i} \frac{K(Sx_i, Sx_j)}{\sum_{k \neq i} K(Sx_i, Sx_k)} y_j \|^2$$

with S diagonal scaling matrix.

Now:

- Using SX, we obtain a 'complete model'.
- Can continue with UKR(1).
- Or just do model assessment.

Experiments

UKR(2) vs. LLE

Use UKR(2) and LLE as initialization. Then determine reconstruction error.

Performance on toy data:

	halfcircle		scurve		spiral	
σ^2	0.0	0.2	0.0	0.5	0.0	0.05
LLE + rescaling	0.00060	0.0735	0.3279	0.4677	0.2364	0.2453
UKR(2) + rescaling	0.00035	0.0472	0.1107	0.1911	0.1971	0.2057
LLE + UKR(1)	0.00059	0.0300	0.0790	0.0725	0.0582	0.0400
UKR(2) + UKR(1)	0.00035	0.0218	0.0481	0.0685	0.0319	0.0369

Experiments

UKR vs. PPS and GTM: (Chang and Ghosh, 2001)

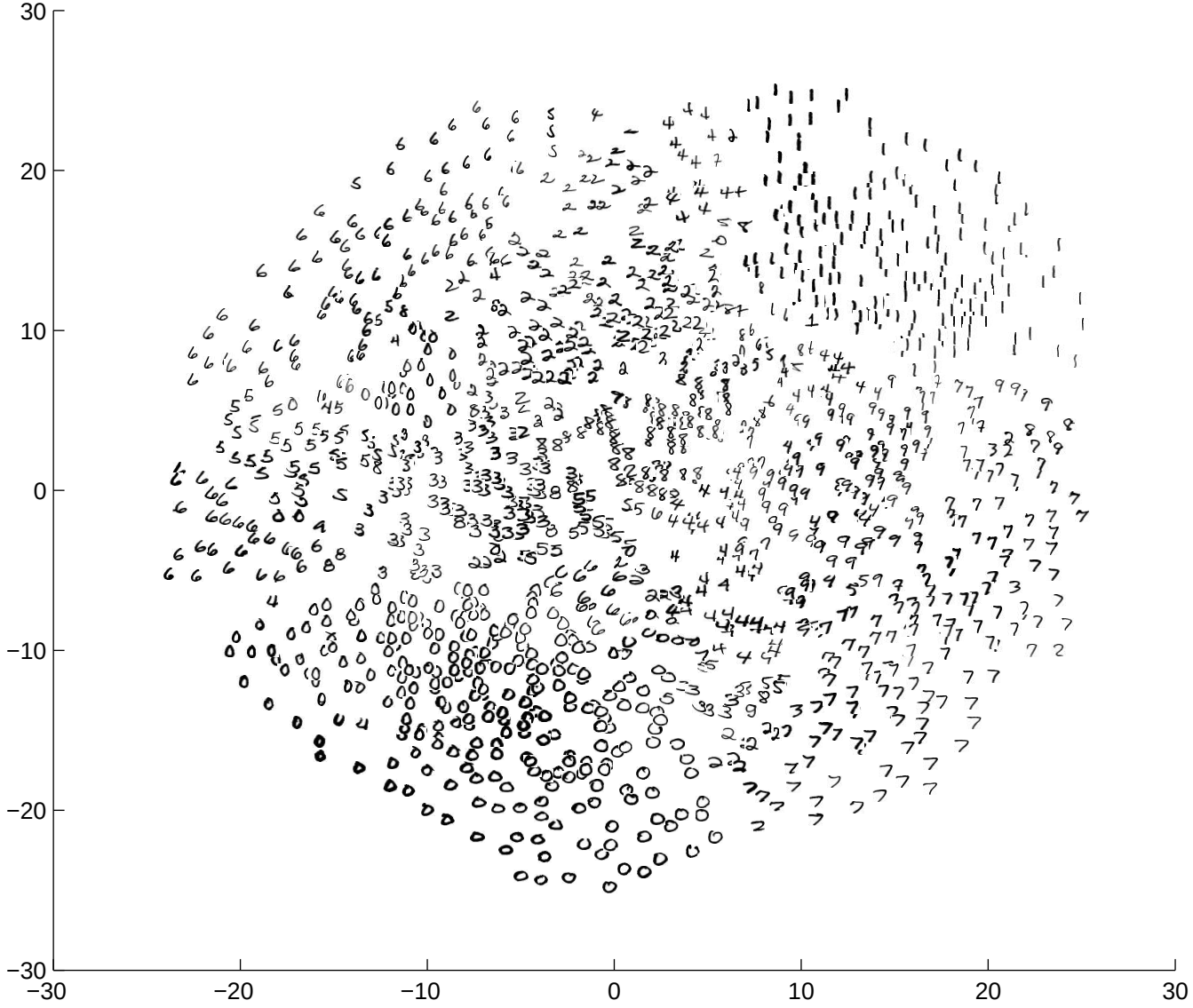
Approximation of UCI-data:

	iris		glass		diabetes	
q	1	2	1	2	1	2
GTM	2.7020	0.9601	8.0681	1.9634	6.7100	1.8822
PPS	2.5786	0.8757	7.9465	1.8616	6.5509	1.8202
UKR(2)	2.1164	1.2667	7.6977	6.1930	6.4130	4.9041
UKR(2) + UKR(1)	1.2017	0.5961	5.8665	4.3678	5.8018	3.9466
UKR(1) Homotopy	0.9414	0.5651	4.7515	3.6402	6.2488	3.9619

Experiments

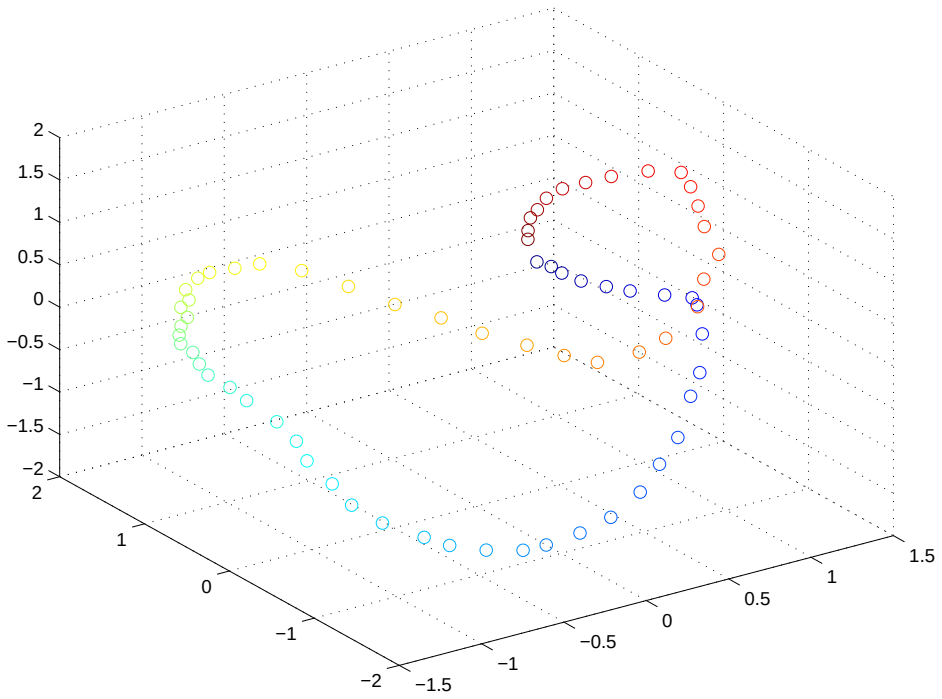
Visualization: MNIST-Digits

Unsupervised Kernel Regression



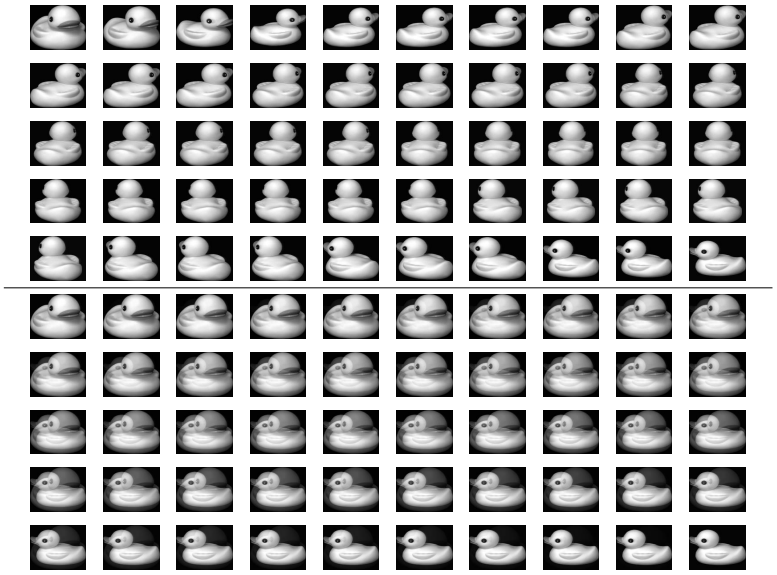
Experiments

The duck from the beginning in 3d:



Experiments

Interpolation in Latent Space
vs.
Interpolation in Observable Space



Experiments

Experiments

Introducing prior information:

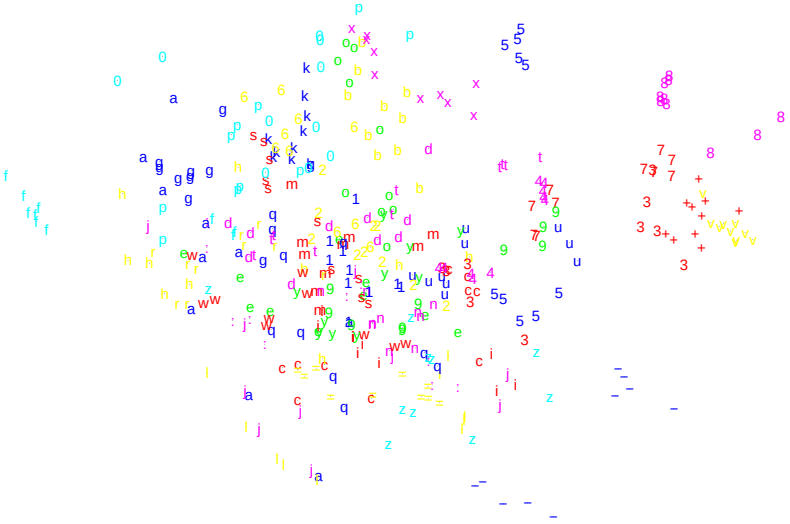
Let user determine what shall end up in which dimension.

Approach: For different properties, sort data into pre-defined equivalence classes.

In the (latent) dimensions that shall reflect some specific property, penalize distance of objects that belong to the same class (with respect to this property).

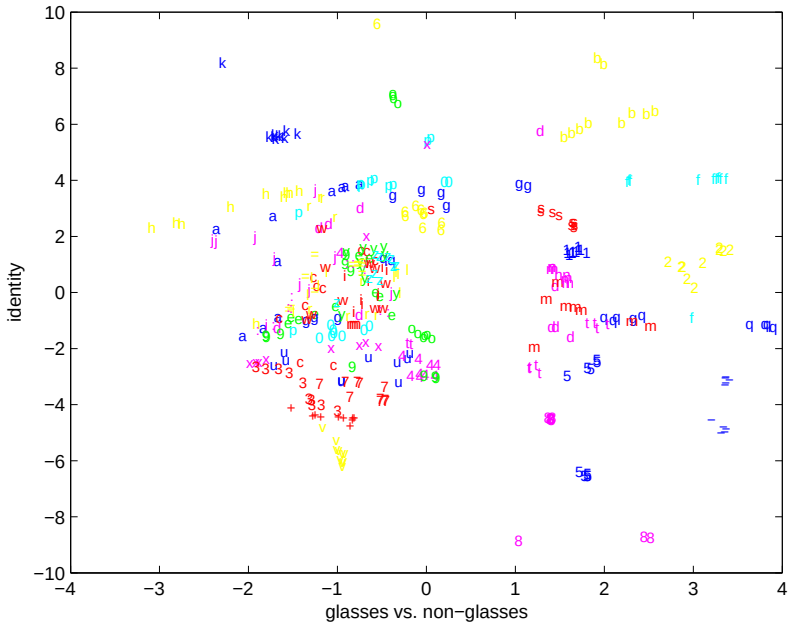
Experiments

The (2-dimensional) PCA representation of the 'olivettifaces'-data looks like:



Experiments

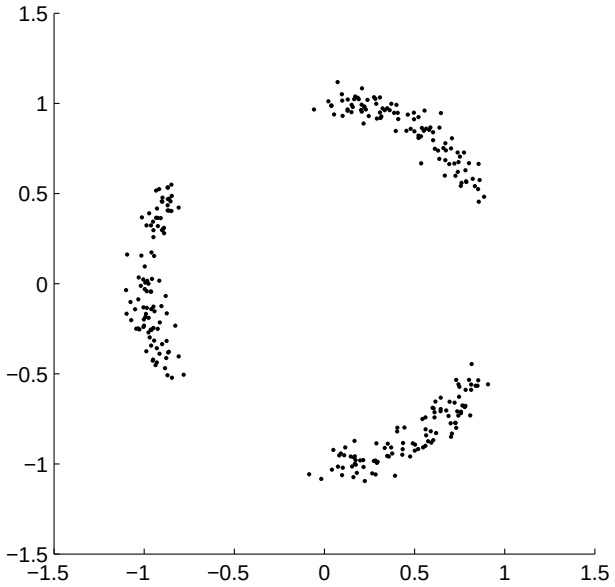
Considering the two properties 'identity' and 'wears glasses', the latent representation might look like:



Experiments

Now we can give people glasses:

'Bridge the gap'



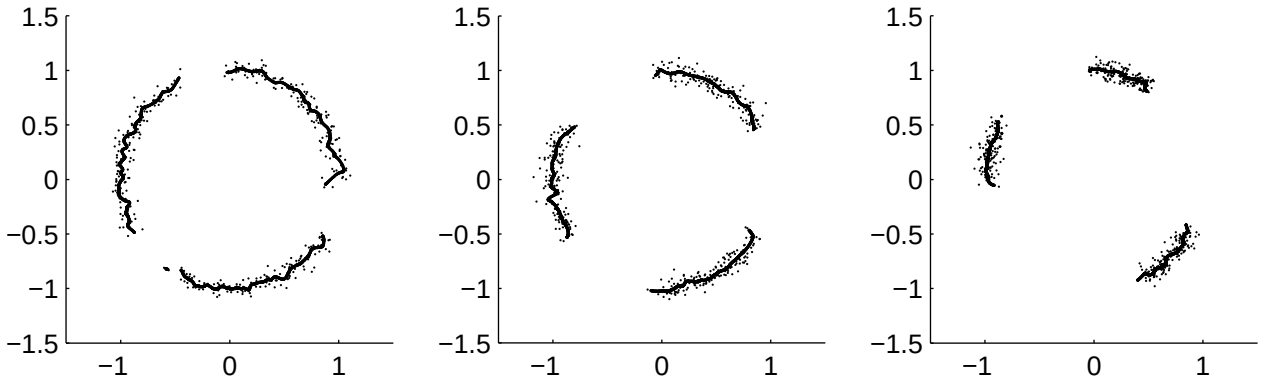
Clustering or Dimensionality Reduction?!

'Bridge the gap'

- Recall: UKR learns a latent space *density*, not just latent representatives and not just a principal manifold!

- We can use the latent density constraints at test time:

'Bridge the gap'



Things Not Covered/Future Work:

- (Mercer-)Kernel Feature Space Variant
- Approximate (whole) data set using a *reduced set of prototypes*
- Condensing
- 'Two data spaces', CCA
- Matlab Toolbox