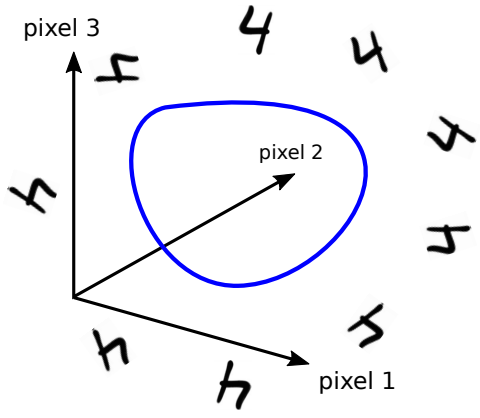
Non-linear Dimensionality Reduction

Rich Turner

Applications and reasons to study dimensionality reduction

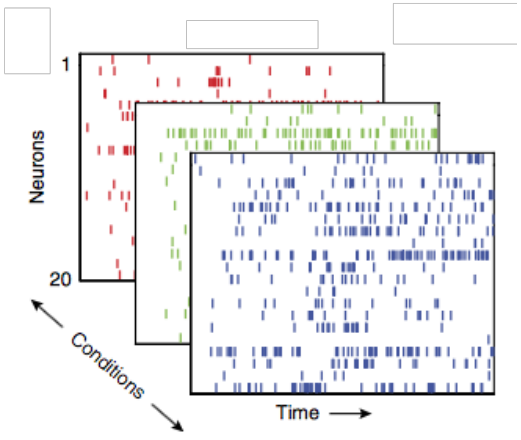
modelling data on/near manifolds

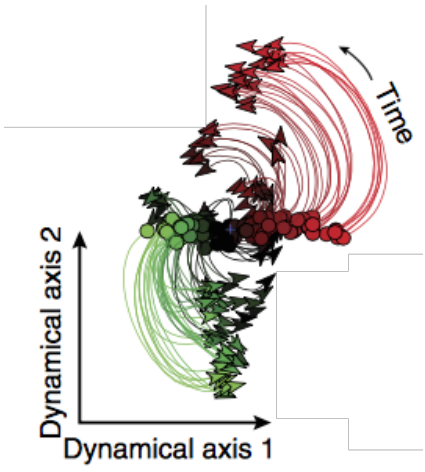
e.g. objects + transformations
= non-linear manifolds



visualisation

understanding structure
in high-dimensional data



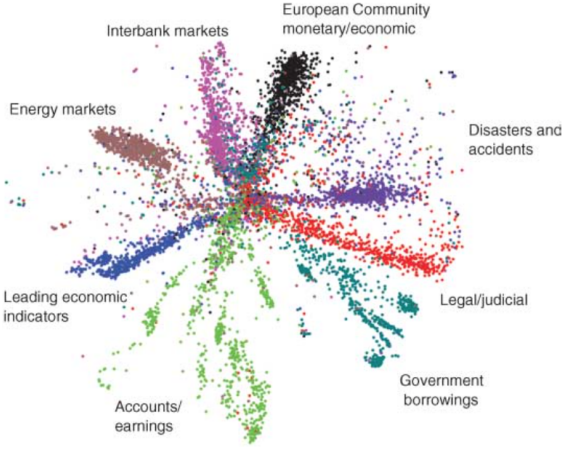


Cunningham and Yu,
Nature Neuro, 2014

simple building blocks

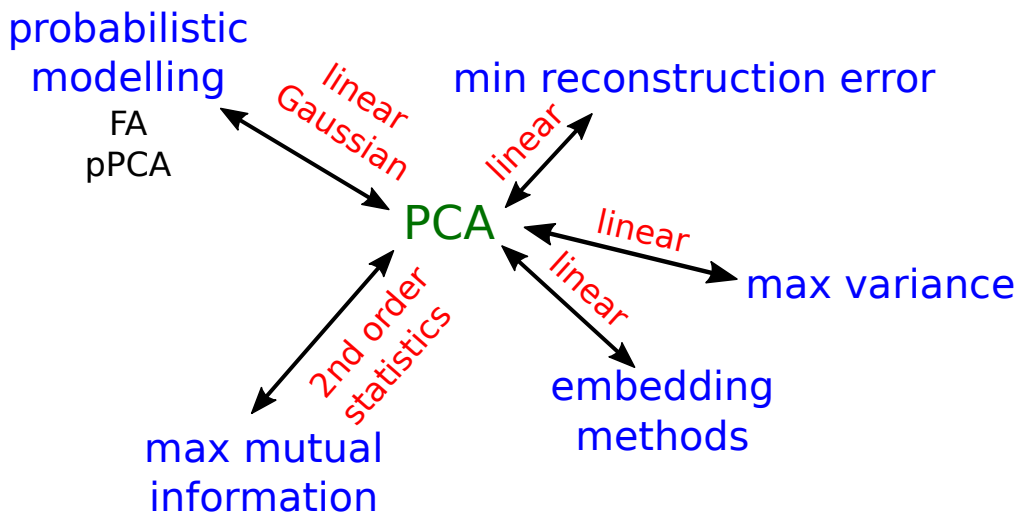
preprocessing/feature learning:
reducing computational complexity
improving statistical efficiency

compose into complex models:
FA -> LGSSMs
GPLVM -> GPSSMs

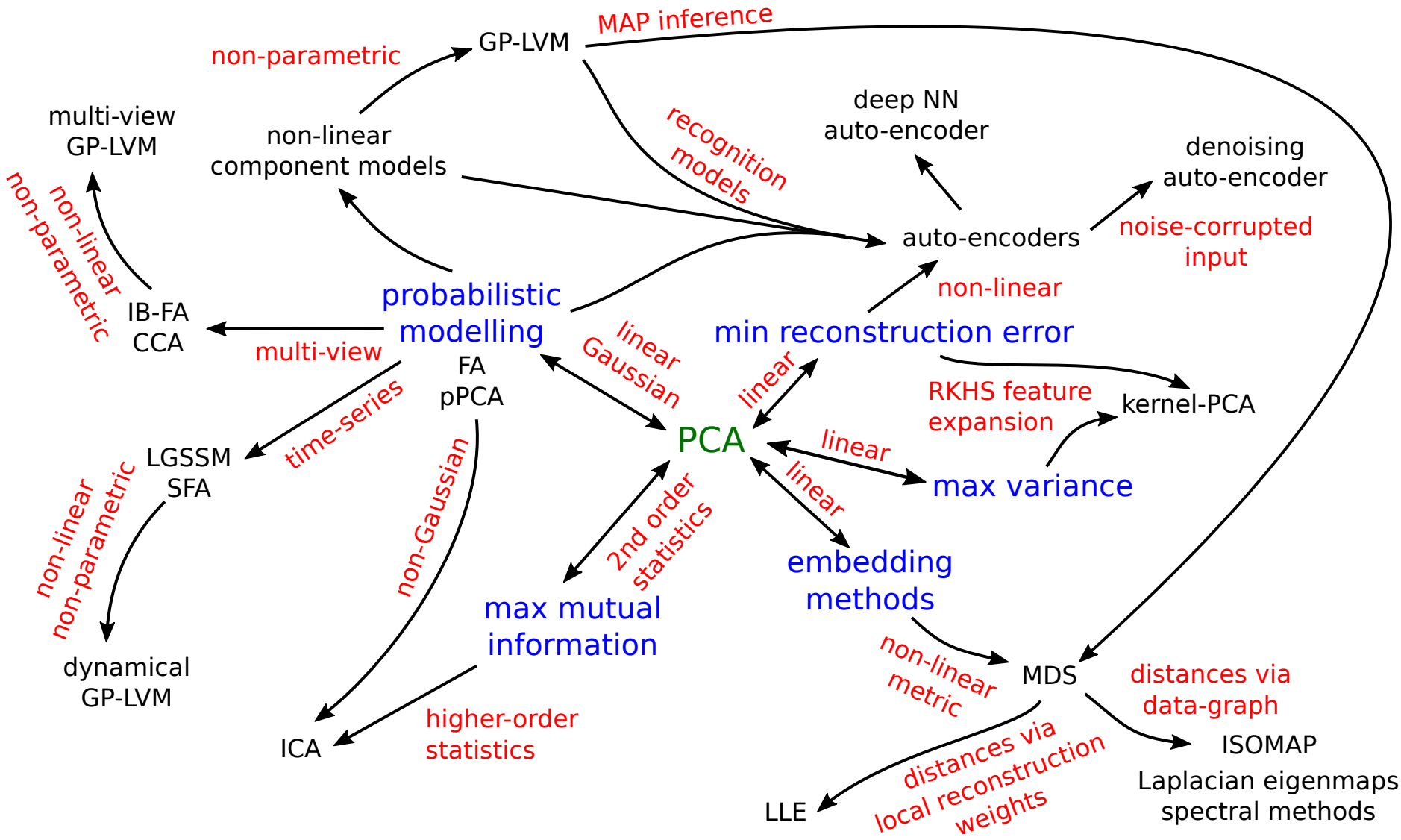


Hinton and Salakhutdinov
Science, 2006

Dimensionality reduction: conceptual space



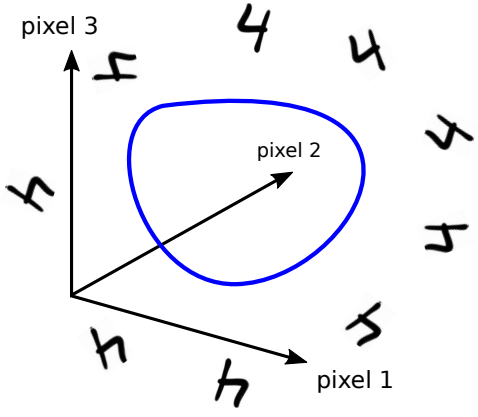
Dimensionality reduction: conceptual space

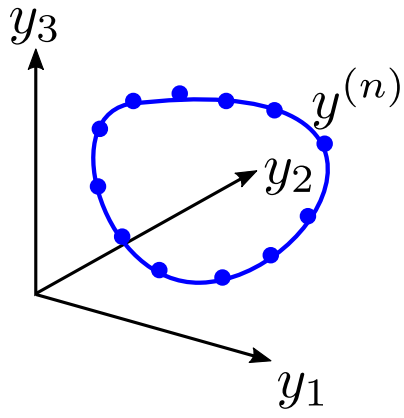


Dimensionality reduction via distance preserving embeddings

original dataset

high-dimensional

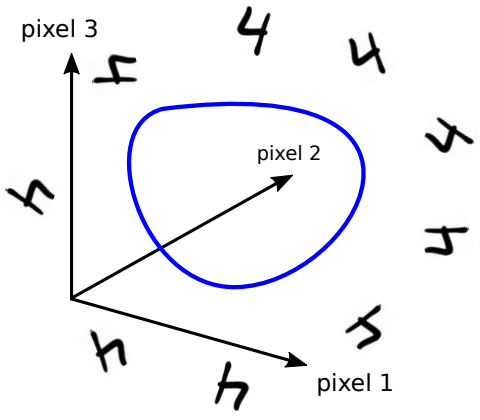




$$\dim(y) = D$$

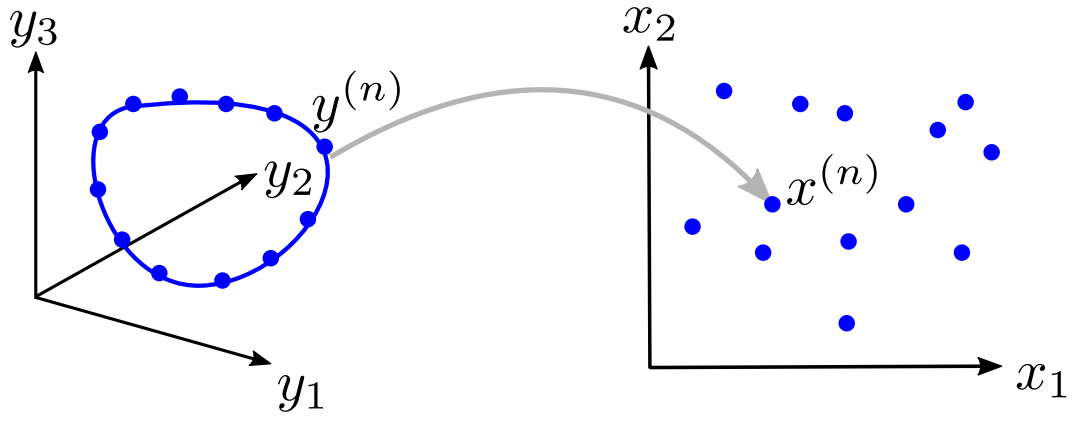
$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}||$$

Dimensionality reduction via distance preserving embeddings



original dataset
high-dimensional

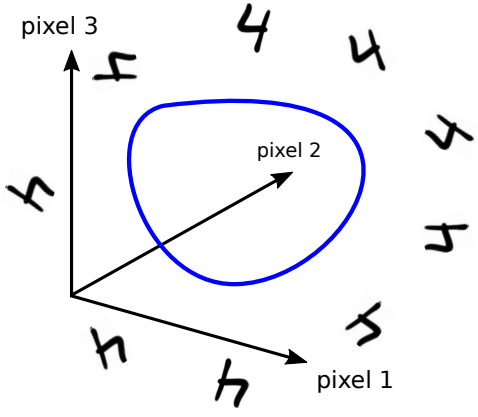
new dataset
low-dimensional



$$\dim(y) = D \quad > \quad \dim(x) = K$$

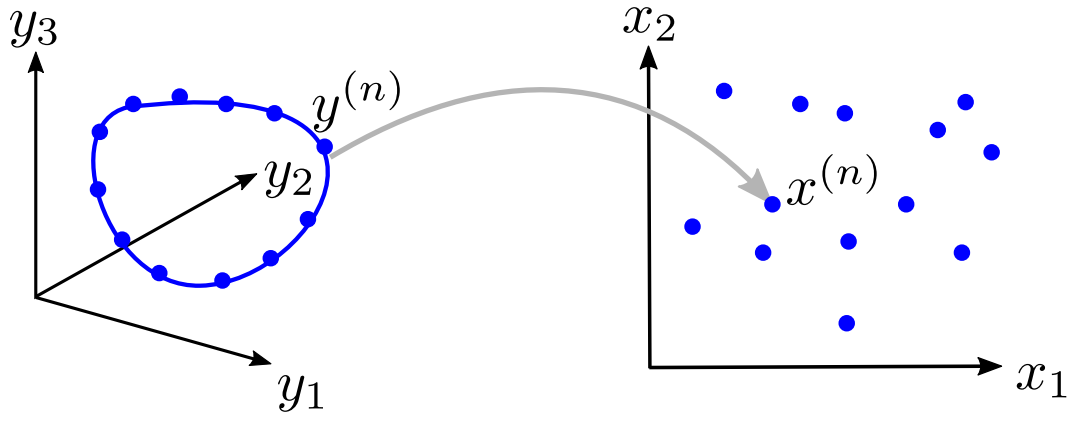
$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d_{nm}^{(x)} = ||x^{(n)} - x^{(m)}||$$

Dimensionality reduction via distance preserving embeddings



original dataset
high-dimensional

new dataset
low-dimensional



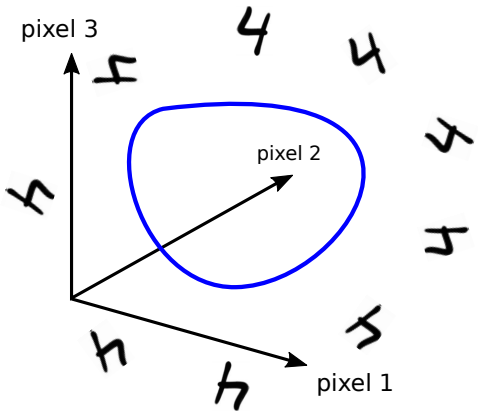
$$\dim(y) = D \quad > \quad \dim(x) = K$$

$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d_{nm}^{(x)} = ||x^{(n)} - x^{(m)}||$$

embed by optimising new datapoints to match distances:

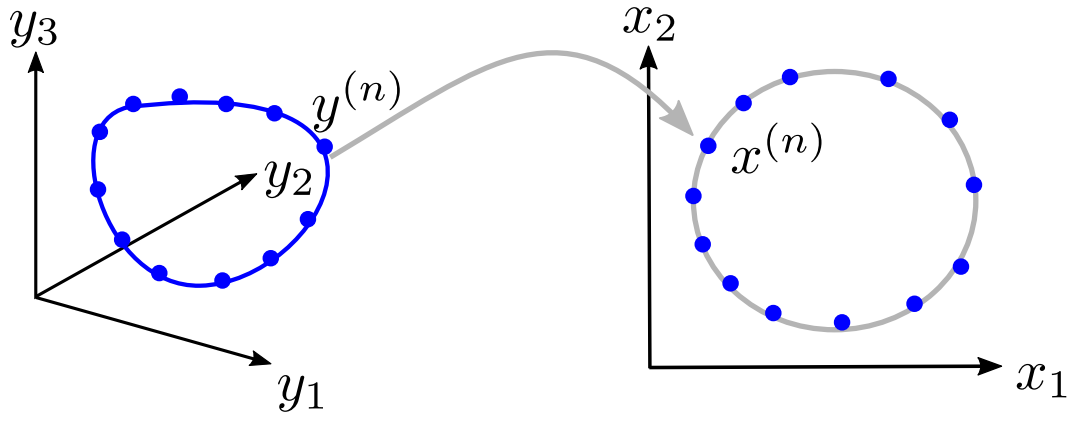
$$\underset{\{x^{(n)}\}_{n=1}^{N}}{\arg\min} \sum_{n<m} ||d_{nm}^{(y)} - d_{nm}^{(x)}||$$

Dimensionality reduction via distance preserving embeddings



original dataset
high-dimensional

new dataset
low-dimensional



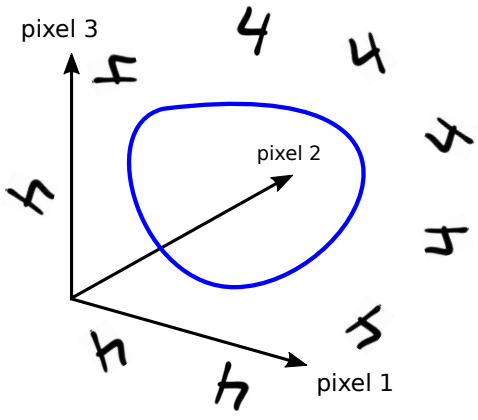
$$\dim(y) = D \quad > \quad \dim(x) = K$$

$$d^{(y)}_{nm} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d^{(x)}_{nm} = ||x^{(n)} - x^{(m)}||$$

embed by optimising new datapoints to match distances:

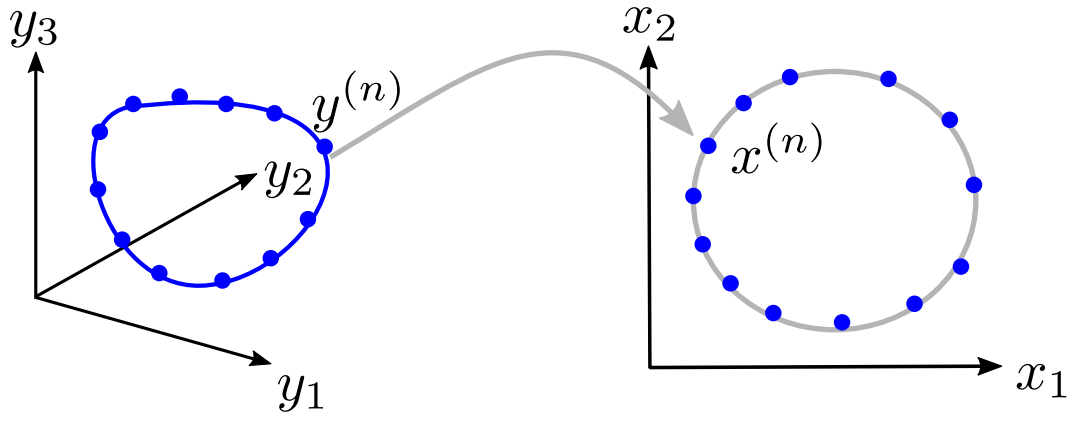
$$\underset{\{x^{(n)}\}_{n=1}^{N}}{\arg\min} \sum_{n<m} ||d^{(y)}_{nm} - d^{(x)}_{nm}||$$

Dimensionality reduction via distance preserving embeddings



original dataset
high-dimensional

new dataset
low-dimensional



$$\dim(y) = D \quad > \quad \dim(x) = K$$

$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d_{nm}^{(x)} = ||x^{(n)} - x^{(m)}||$$

embed by optimising new datapoints to match distances:

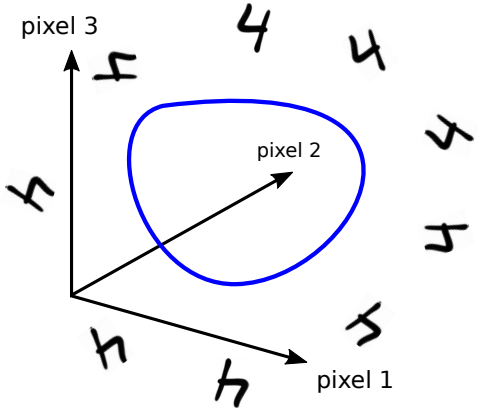
$$\arg\min_{\{x^{(n)}\}_{n=1}^{N}} \sum_{n<m} ||d_{nm}^{(y)} - d_{nm}^{(x)}||$$

PCA: Euclidean metric (squared error)

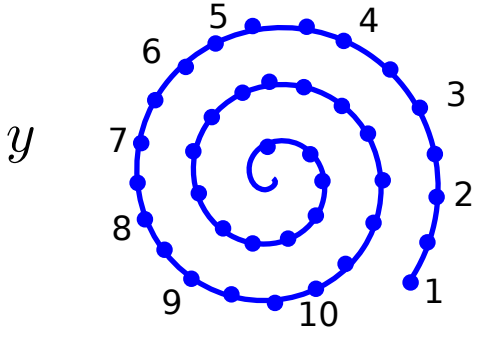
MDS: general distance metric e.g. ISOMAP

Tenenbaum et al
Science, 2000

Dimensionality reduction via distance preserving embeddings

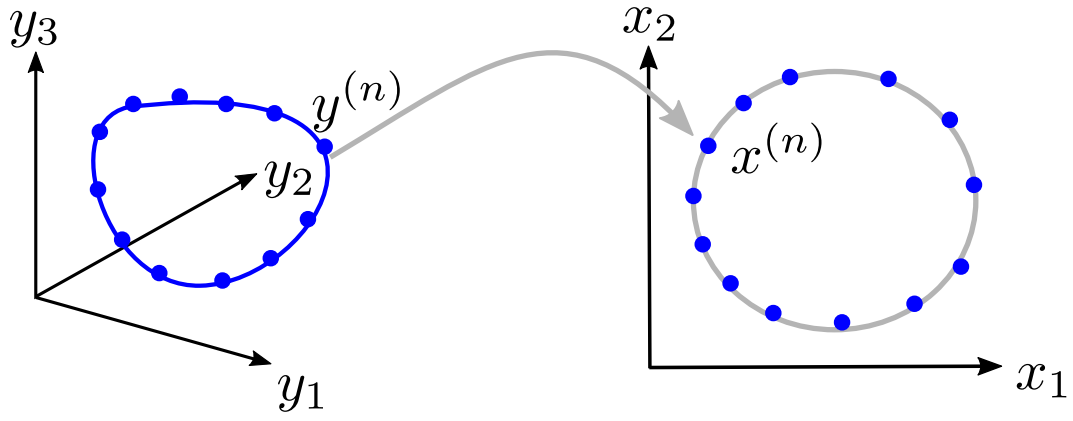


Question:
Will MDS with Euclidean distances make sensible 1D embeddings of:



original dataset
high-dimensional

new dataset
low-dimensional



$$\dim(y) = D \quad > \quad \dim(x) = K$$

$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d_{nm}^{(x)} = ||x^{(n)} - x^{(m)}||$$

embed by optimising new datapoints to match distances:

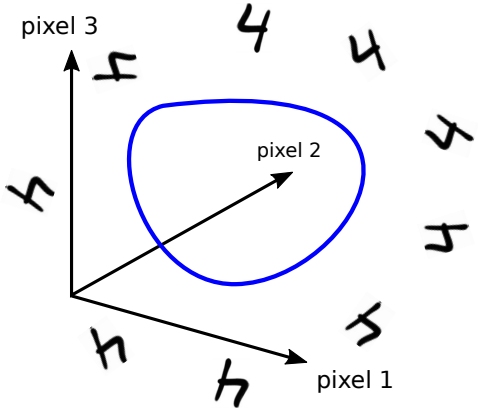
$$\arg\min_{\{x^{(n)}\}_{n=1}^{N}} \sum_{n<m} ||d_{nm}^{(y)} - d_{nm}^{(x)}||$$

PCA: Euclidean metric (squared error)

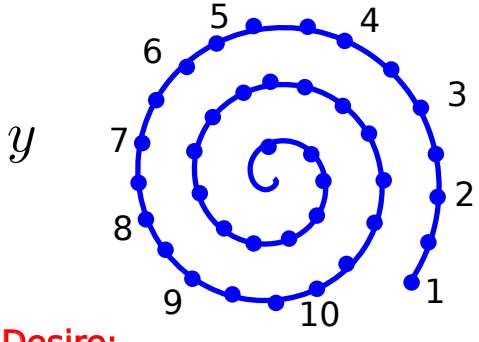
MDS: general distance metric e.g. ISOMAP

Tenenbaum et al
Science, 2000

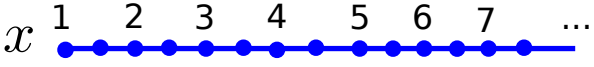
Dimensionality reduction via distance preserving embeddings



Question:
Will MDS with Euclidean distances make sensible 1D embeddings of:

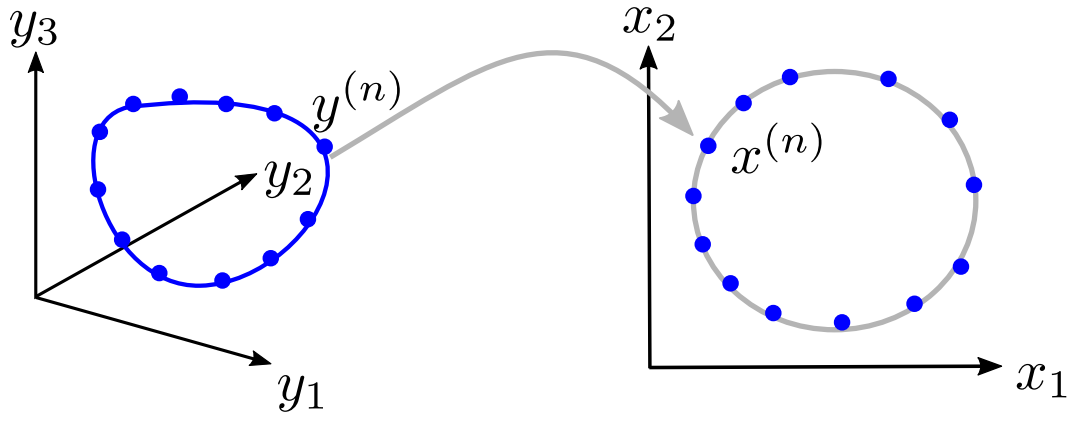


Desire:



original dataset
high-dimensional

new dataset
low-dimensional



$$\dim(y) = D \quad > \quad \dim(x) = K$$

$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d_{nm}^{(x)} = ||x^{(n)} - x^{(m)}||$$

embed by optimising new datapoints to match distances:

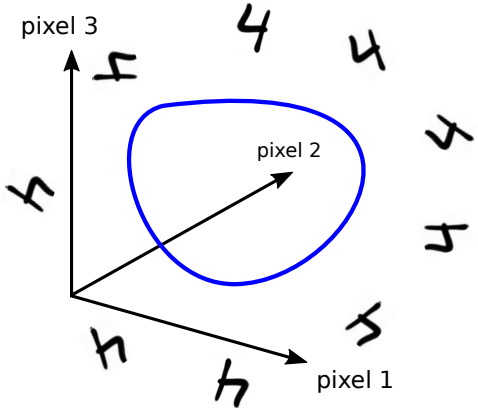
$$\arg\min_{\{x^{(n)}\}_{n=1}^{N}} \sum_{n<m} ||d_{nm}^{(y)} - d_{nm}^{(x)}||$$

PCA: Euclidean metric (squared error)

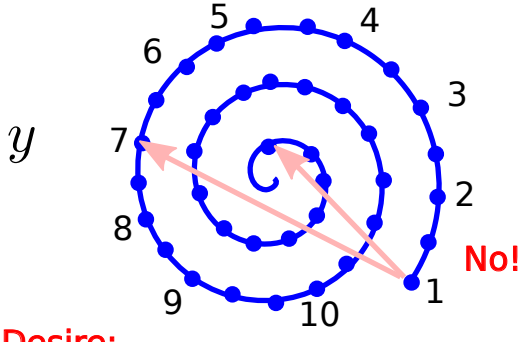
MDS: general distance metric e.g. ISOMAP

Tenenbaum et al
Science, 2000

Dimensionality reduction via distance preserving embeddings

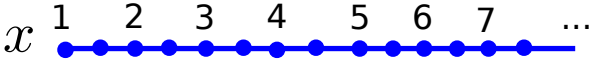


Question:
Will MDS with Euclidean distances make sensible 1D embeddings of:



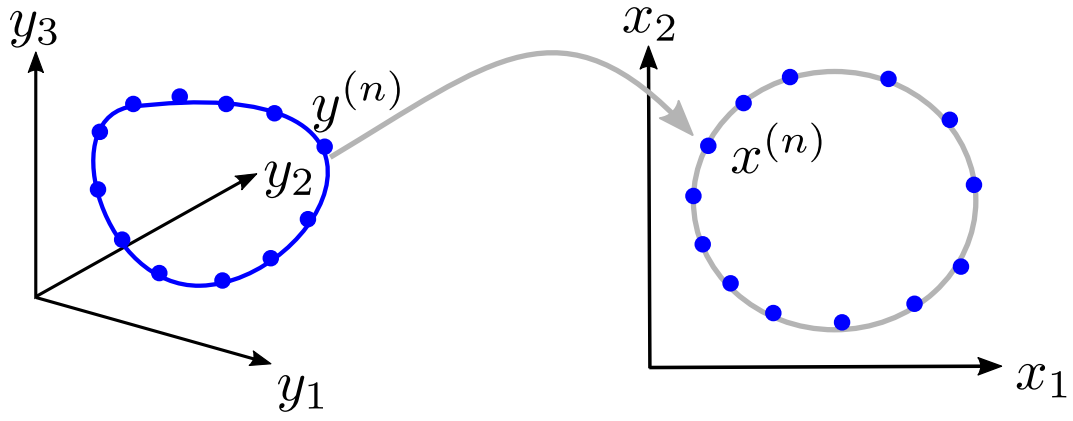
No!

Desire:



original dataset
high-dimensional

new dataset
low-dimensional



$$\dim(y) = D \quad > \quad \dim(x) = K$$

$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d_{nm}^{(x)} = ||x^{(n)} - x^{(m)}||$$

embed by optimising new datapoints to match distances:

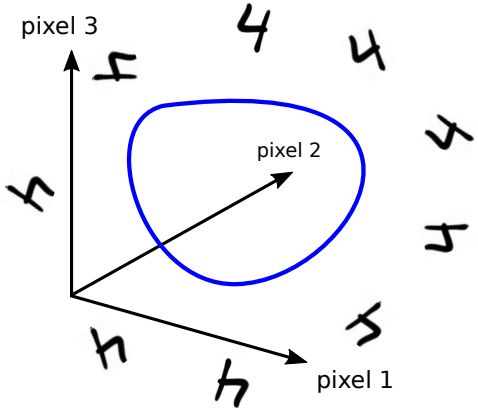
$$\arg\min_{\{x^{(n)}\}_{n=1}^{N}} \sum_{n<m} ||d_{nm}^{(y)} - d_{nm}^{(x)}||$$

PCA: Euclidean metric (squared error)

MDS: general distance metric e.g. ISOMAP

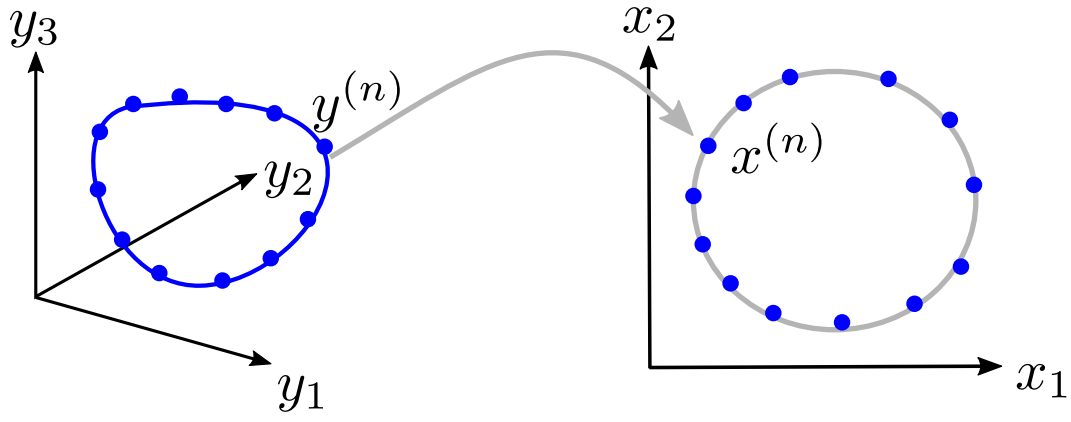
Tenenbaum et al
Science, 2000

Dimensionality reduction via distance preserving embeddings



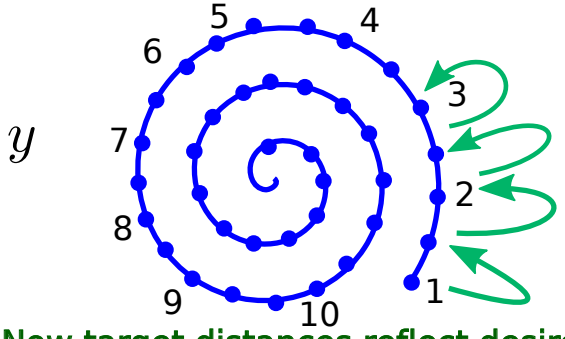
original dataset
high-dimensional

new dataset
low-dimensional

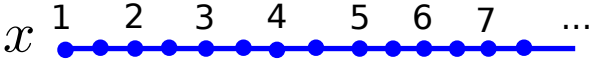


$$\dim(y) = D \quad > \quad \dim(x) = K$$

ISOMAP:
Geodesic distance via neighbourhood graph



Now target distances reflect desires



$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d_{nm}^{(x)} = ||x^{(n)} - x^{(m)}||$$

embed by optimising new datapoints to match distances:

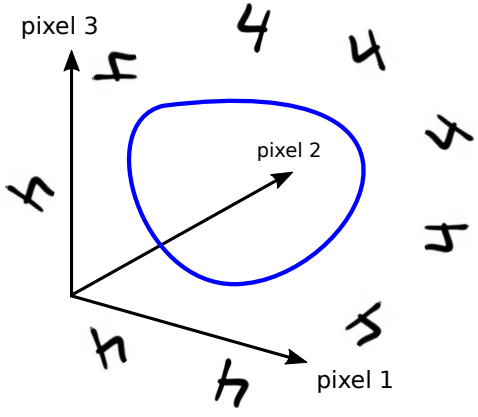
$$\arg\min_{\{x^{(n)}\}_{n=1}^{N}} \sum_{n<m} ||d_{nm}^{(y)} - d_{nm}^{(x)}||$$

PCA: Euclidean metric (squared error)

MDS: general distance metric e.g. ISOMAP

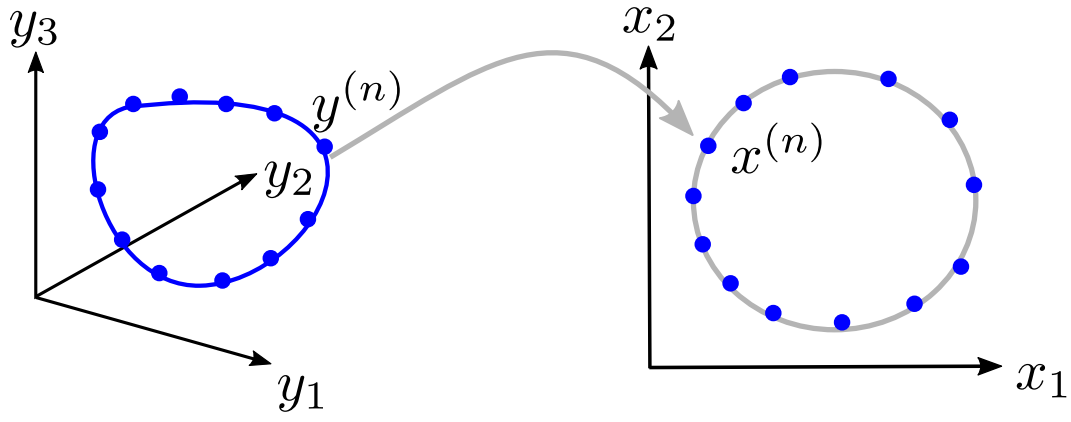
Tenenbaum et al
Science, 2000

Dimensionality reduction via distance preserving embeddings



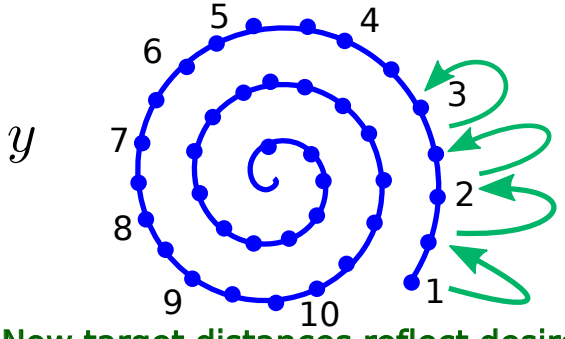
original dataset
high-dimensional

new dataset
low-dimensional



$$\dim(y) = D \quad > \quad \dim(x) = K$$

ISOMAP:
Geodesic distance via neighbourhood graph



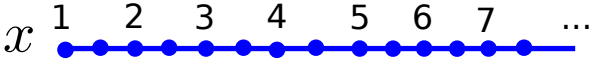
requires dense sampling of manifold

$$d_{nm}^{(y)} = ||y^{(n)} - y^{(m)}|| \quad \approx \quad d_{nm}^{(x)} = ||x^{(n)} - x^{(m)}||$$

embed by optimising new datapoints to match distances:

$$\arg\min_{\{x^{(n)}\}_{n=1}^{N}} \sum_{n<m} ||d_{nm}^{(y)} - d_{nm}^{(x)}||$$

Now target distances reflect desires



PCA: Euclidean metric (squared error)

MDS: general distance metric e.g. ISOMAP

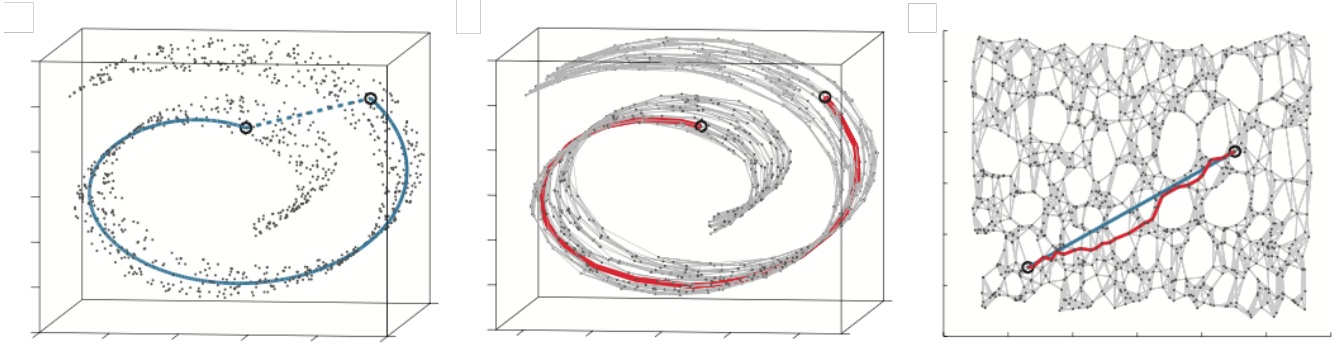
Tenenbaum et al
Science, 2000

Dimensionality reduction via distance preserving embeddings

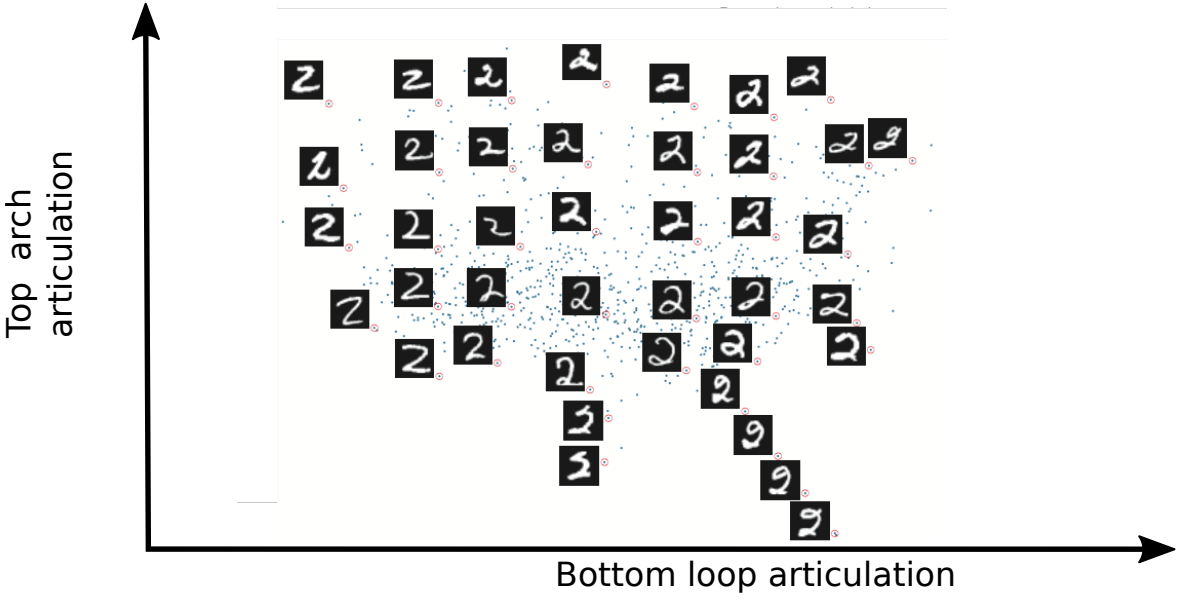
original dataset

geodesic distance via neighbourhood graph

new dataset



y y x



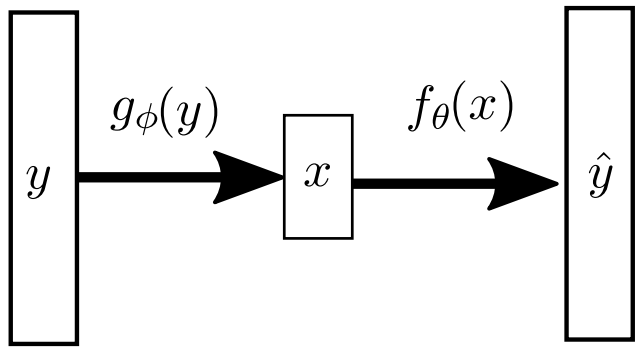
ISOMAP, Tenenbaum et al Science 2000
LLE, Roweis et al. Science 2000
tSNE, Hinton and van der Maaten JMLR 2008

Limitations (also strengths?)

- non-linear embedding-based methods require optimisation of new representation x ($N \times K$ parameters)
- works well for low-dimensional embeddings $K = 2$ or 3, but slow for higher dimensions
- does not provide quick way to map new data-points into new representation ($y^{(new)} \rightarrow x^{(new)}$ involves optimisation)
- does not provide a way for mapping out of reduced space ($x^{(new)} \rightarrow y^{(new)}$?)

Next: auto-encoders fix above using supervised learning algorithm (non-linear regression) for unsupervised dimensionality reduction

Dimensionality reduction using auto-encoders



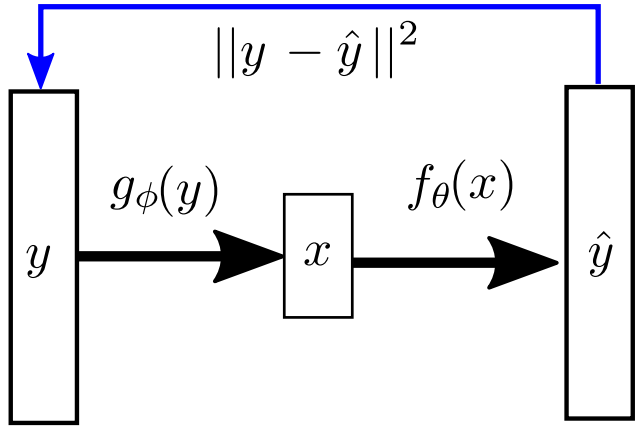
data

low-dim code

reconstruction

$$x(y) = g_\phi(y) \qquad \hat{y}(x) = f_\theta(x)$$

Dimensionality reduction using auto-encoders



Want to learn interesting embedding
(not identity mapping)

Achieve via constraint:
dimensionality

data

low-dim code

$$x(y) = g_\phi(y)$$

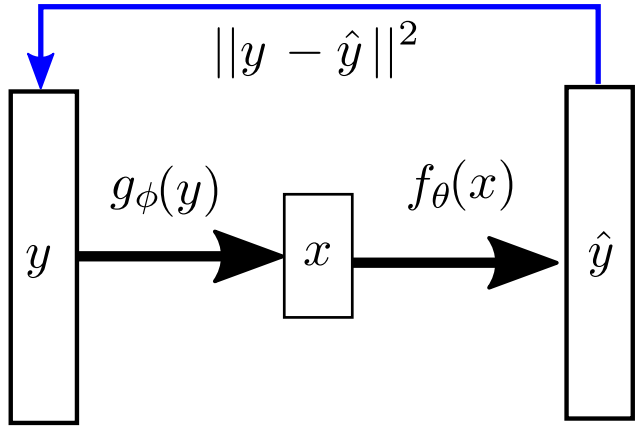
reconstruction

$$\hat{y}(x) = f_\theta(x)$$

cost function

$$\arg\min_{\theta,\phi} \sum_{n=1}^{N} ||y^{(n)} - \hat{y}^{(n)}||^2$$

Dimensionality reduction using auto-encoders



Want to learn interesting embedding
(not identity mapping)

Achieve via constraint:
dimensionality, sparsity, function complexity

data

low-dim code

reconstruction

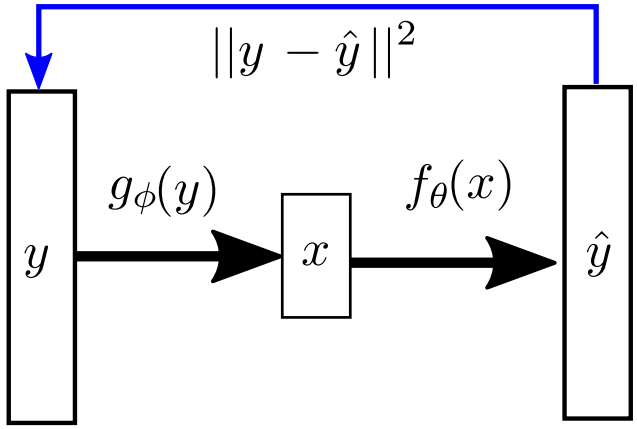
cost function

$$x(y) = g_\phi(y)$$

$$\hat{y}(x) = f_\theta(x)$$

$$\arg\min_{\theta,\phi} \sum_{n=1}^{N} ||y^{(n)} - \hat{y}^{(n)}||^2 + \text{constraints}$$

Dimensionality reduction using auto-encoders



Want to learn interesting embedding
(not identity mapping)

Achieve via constraint:
dimensionality, sparsity, function complexity

data

low-dim code

$$x(y) = g_\phi(y)$$

reconstruction

$$\hat{y}(x) = f_\theta(x)$$

cost function

$$\arg\min_{\theta,\phi} \sum_{n=1}^{N} ||y^{(n)} - \hat{y}^{(n)}||^2 + \text{constraints}$$

PCA:

linear

$$x = \Phi y$$

$$\dim(\Phi) = K \times D$$

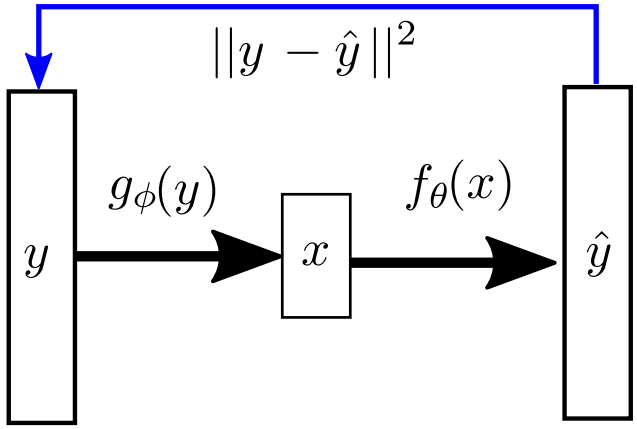
linear

$$\hat{y} = \Theta x$$

$$\dim(\Theta) = D \times K$$

$$\arg\min_{\Theta,\Phi} \sum_{n=1}^{N} ||y^{(n)} - \Theta\Phi y^{(n)}||^2$$

Dimensionality reduction using auto-encoders



Want to learn interesting embedding
(not identity mapping)

Achieve via constraint:
dimensionality, sparsity, function complexity

data

low-dim code

$$x(y) = g_\phi(y)$$

reconstruction

$$\hat{y}(x) = f_\theta(x)$$

cost function

$$\arg\min_{\theta,\phi} \sum_{n=1}^{N} ||y^{(n)} - \hat{y}^{(n)}||^2 + \text{constraints}$$

PCA:

linear

$$x = \Phi y$$

$$\dim(\Phi) = K \times D$$

linear

$$\hat{y} = \Theta x$$

$$\dim(\Theta) = D \times K$$

$$\arg\min_{\Theta,\Phi} \sum_{n=1}^{N} ||y^{(n)} - \Theta\Phi y^{(n)}||^2$$

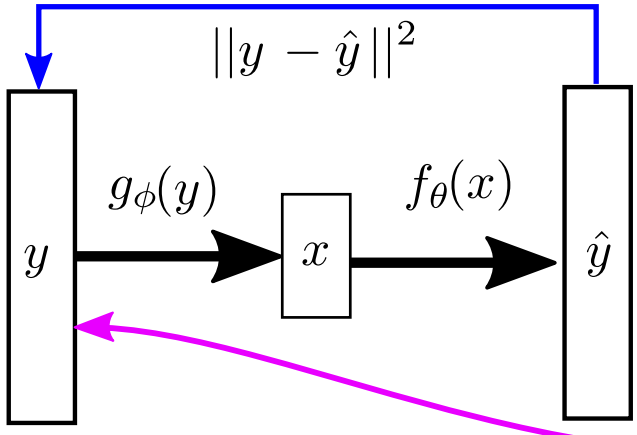
deep neural AA:

deep neural network

deep neural network

Hinton et al Science, 2006

Dimensionality reduction using auto-encoders



Want to learn interesting embedding
(not identity mapping)

Achieve via constraint:
dimensionality, sparsity, function complexity

Achieve via data corruption:
add noise, drop-out, transform

Vincent, Bengio et al
JMLR, 2010

data

low-dim code

$$x(y) = g_\phi(y)$$

reconstruction

$$\hat{y}(x) = f_\theta(x)$$

cost function

$$\arg\min_{\theta,\phi} \sum_{n=1}^{N} ||y^{(n)} - \hat{y}^{(n)}||^2 + \text{constraints}$$

PCA:

linear

$$x = \Phi y$$

$$\dim(\Phi) = K \times D$$

linear

$$\hat{y} = \Theta x$$

$$\dim(\Theta) = D \times K$$

$$\arg\min_{\Theta,\Phi} \sum_{n=1}^{N} ||y^{(n)} - \Theta\Phi y^{(n)}||^2$$

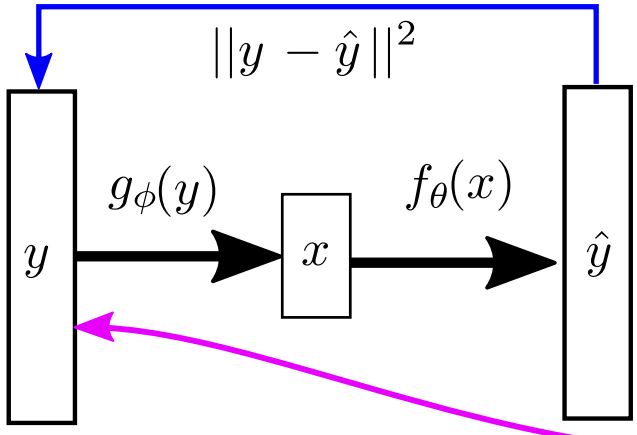
deep neural AA:

deep neural network

deep neural network

Hinton et al Science, 2006

Dimensionality reduction using auto-encoders



Want to learn interesting embedding
(not identity mapping)

Achieve via constraint:
dimensionality, sparsity, function complexity

Achieve via data corruption:
add noise, drop-out, transform

Vincent, Bengio et al
JMLR, 2010

data

low-dim code

$$x(y) = g_\phi(y)$$

reconstruction

$$\hat{y}(x) = f_\theta(x)$$

cost function

$$\arg\min_{\theta,\phi} \sum_{n=1}^{N} ||y^{(n)} - \hat{y}^{(n)}||^2 + \text{constraints}$$

PCA:

linear

$$x = \Phi y$$

$$\dim(\Phi) = K \times D$$

linear

$$\hat{y} = \Theta x$$

$$\dim(\Theta) = D \times K$$

$$\arg\min_{\Theta,\Phi} \sum_{n=1}^{N} ||y^{(n)} - \Theta\Phi y^{(n)}||^2$$

deep neural AA:

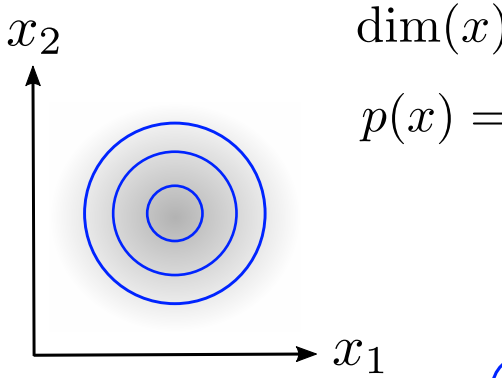
deep neural network

deep neural network

Hinton et al Science, 2006

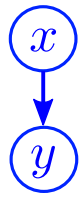
auto-encoders:
1. learn functions for both mappings
2. num. params. does not scale with N

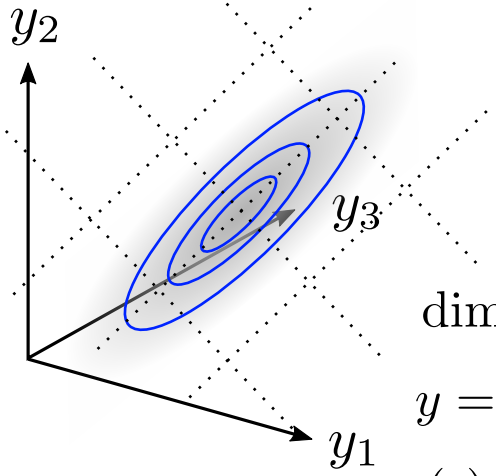
Dimensionality reduction using probabilistic models: PCA



$$\dim(x) = K$$

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$





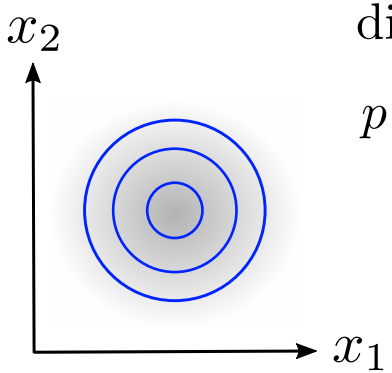
$$\dim(y) = D$$

$$y = \theta x + \sigma\epsilon$$

$$p(\epsilon) = \mathcal{G}(\epsilon; 0, \mathrm{I})$$

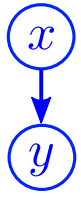
fuzzy pancake

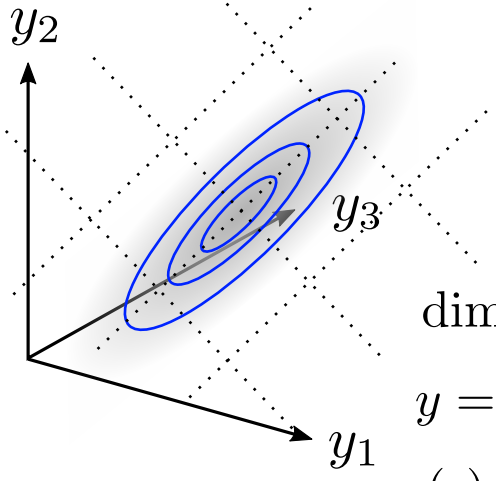
Dimensionality reduction using probabilistic models: PCA



$$\dim(x) = K$$

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$





fuzzy pancake

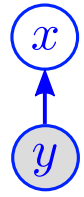
$$\dim(y) = D$$

$$y = \theta x + \sigma\epsilon$$

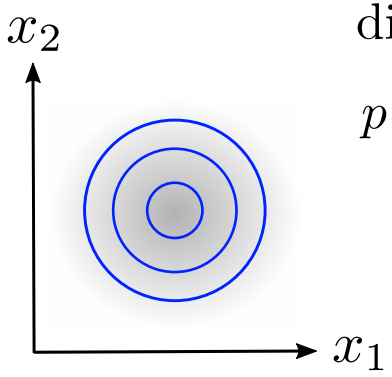
$$p(\epsilon) = \mathcal{G}(\epsilon; 0, \mathrm{I})$$

inference

$$p(x^{(n)}|y^{(n)}\phi, \sigma) = \mathcal{G}(x^{(n)}; \phi y^{(n)}, \Sigma_{x|y})$$

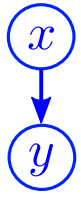


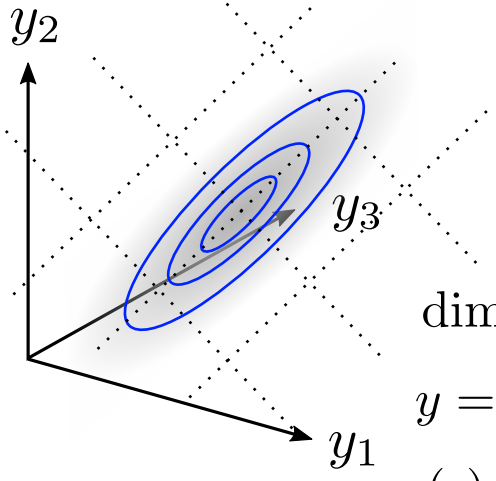
Dimensionality reduction using probabilistic models: PCA



$$\dim(x) = K$$

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$





$$\dim(y) = D$$

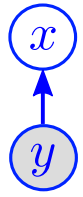
$$y = \theta x + \sigma\epsilon$$

$$p(\epsilon) = \mathcal{G}(\epsilon; 0, \mathrm{I})$$

fuzzy pancake

inference

$$p(x^{(n)}|y^{(n)}\phi, \sigma) = \mathcal{G}(x^{(n)}; \phi y^{(n)}, \Sigma_{x|y})$$

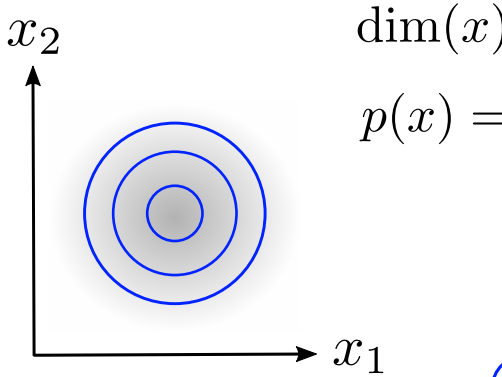


maximum-likelihood learning

$$\theta^{ML}, \sigma^{ML} = \arg\max_{\phi,\sigma} \sum_n \log p(y^{(n)}|\theta, \sigma)$$

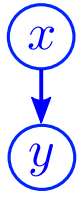
$$p(y^{(n)}|\theta, \sigma) = \mathcal{G}(y^{(n)}; 0, \theta\,\theta^\top + \mathrm{I}\sigma^2)$$

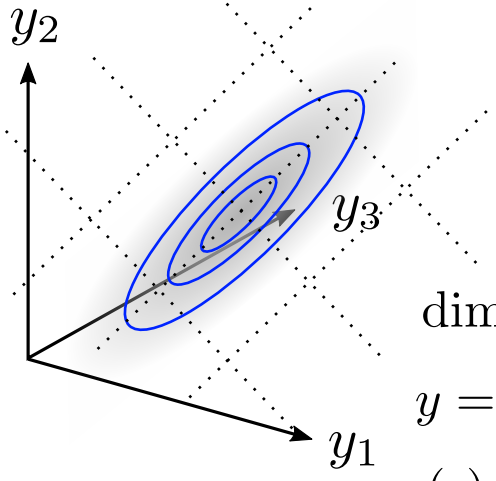
Dimensionality reduction using probabilistic models: PCA



$$\dim(x) = K$$

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$





$$\dim(y) = D$$

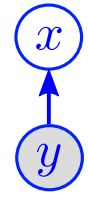
$$y = \theta x + \sigma\epsilon$$

$$p(\epsilon) = \mathcal{G}(\epsilon; 0, \mathrm{I})$$

fuzzy pancake

inference

$$p(x^{(n)}|y^{(n)}\phi, \sigma) = \mathcal{G}(x^{(n)}; \phi y^{(n)}, \Sigma_{x|y})$$



maximum-likelihood learning

$$\theta^{ML}, \sigma^{ML} = \arg\max_{\phi,\sigma} \sum_n \log p(y^{(n)}|\theta, \sigma)$$

$$p(y^{(n)}|\theta, \sigma) = \mathcal{G}(y^{(n)}; 0, \theta\,\theta^\top + \mathrm{I}\sigma^2)$$

differentiate, set to zero, find:

$$\sigma^{ML} = \frac{1}{D-K} \sum_{k=K+1}^{D} \lambda_k$$

$E_K = [e_1, \ldots, e_K]$
$\Lambda_K = \mathrm{diag}(\lambda_1, \ldots, \lambda_K)$

$$\theta^{ML} = E_K(\Lambda_K - \sigma^2\mathrm{I})^{1/2} R$$

arbitrary rotation matrix

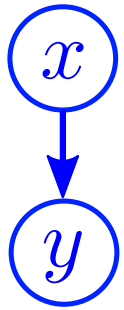
data-covariance

$$\hat{\Sigma}_y e_k = \lambda_k e_k \qquad \hat{\Sigma}_y = \tfrac{1}{N} \textstyle\sum_n y^{(n)} (y^{(n)})^\top$$

eigenvalues eigenvectors

Tipping and Bishop 1999, Roweis 1997

Dimensionality reduction using probabilistic models: a family of models



MODEL CLASS

full linear — factor analysis (FA)

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

diagonal

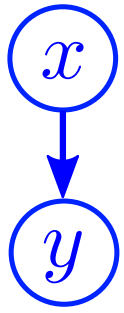
$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

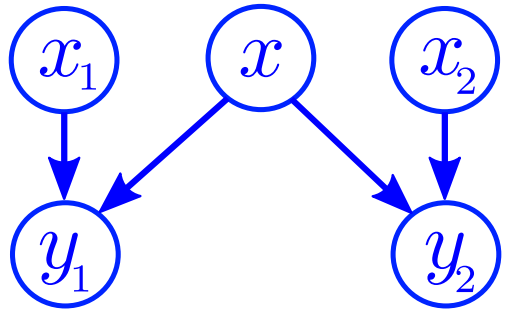
special linear — PCA

$$D = \sigma^2 \mathrm{I}$$

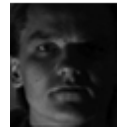
train: eigenvalue problem

Dimensionality reduction using probabilistic models: a family of models





audio



video

MODEL CLASS

full linear — factor analysis (FA)

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

diagonal

$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

inter-battery FA

$$p(x), p(x_i) = \mathcal{G}(x; 0, \mathrm{I})$$

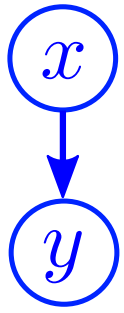
$$p(y_i|x) = \mathcal{G}(y_i; \theta_i^{\text{sh}} x + \theta_i^{\text{pri}} x_i, D)$$

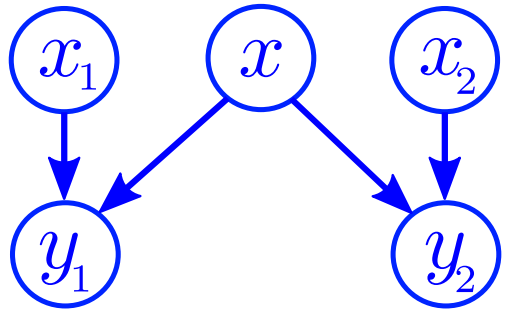
special linear — PCA

$$D = \sigma^2 \mathrm{I}$$

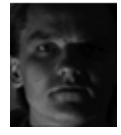
train: eigenvalue problem

Dimensionality reduction using probabilistic models: a family of models





audio



video

MODEL CLASS

full linear

factor analysis (FA)

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

diagonal

$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

inter-battery FA

$$p(x), p(x_i) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_i|x) = \mathcal{G}(y_i; \theta_i^{\text{sh}} x + \theta_i^{\text{pri}} x_i, D)$$

special linear

PCA

$$D = \sigma^2 \mathrm{I}$$

train: eigenvalue problem

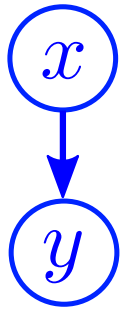
canonical correlation analysis

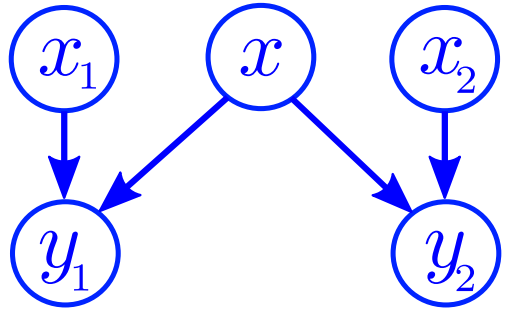
find linear projections of y_i that max correlation

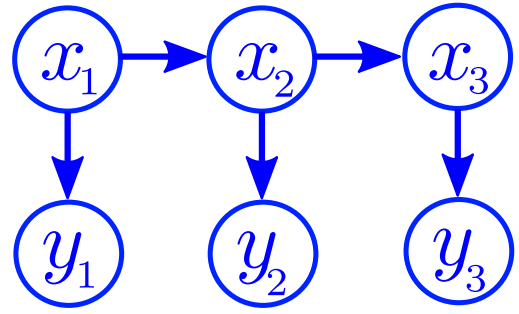
$$D = \sigma^2 \mathrm{I}$$

train: generalised eigenvalue problem

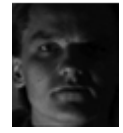
Dimensionality reduction using probabilistic models: a family of models







audio



video

MODEL CLASS

full linear

factor analysis (FA)

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

diagonal

$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

inter-battery FA

$$p(x), p(x_i) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_i|x) = \mathcal{G}(y_i; \theta_i^{\text{sh}} x + \theta_i^{\text{pri}} x_i, D)$$

special linear

PCA

$$D = \sigma^2 \mathrm{I}$$

train: eigenvalue problem

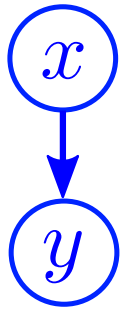
canonical correlation analysis

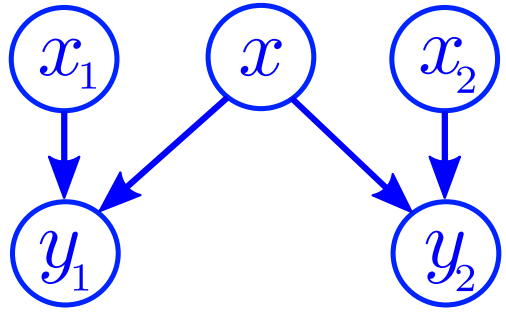
find linear projections of y_i that max correlation

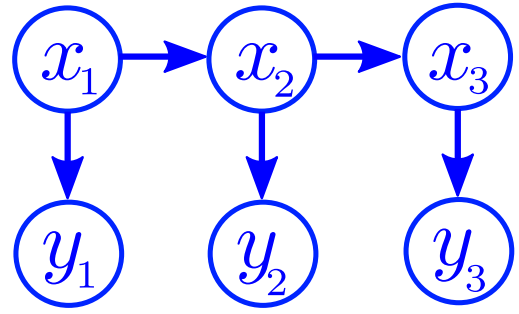
$$D = \sigma^2 \mathrm{I}$$

train: generalised eigenvalue problem

Dimensionality reduction using probabilistic models: a family of models

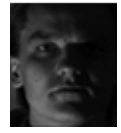






MODEL CLASS

audio



video

full linear

factor analysis (FA)

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$
$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

diagonal

inter-battery FA

$$p(x), p(x_i) = \mathcal{G}(x; 0, \mathrm{I})$$
$$p(y_i|x) = \mathcal{G}(y_i; \theta_i^{\mathrm{sh}} x + \theta_i^{\mathrm{pri}} x_i, D)$$

LGSSM

$$p(x_t|x_{t-1}) = \mathcal{G}(x_t; \Psi x_{t-1}, \Sigma)$$
$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

special linear

PCA

$$D = \sigma^2 \mathrm{I}$$

train: eigenvalue problem

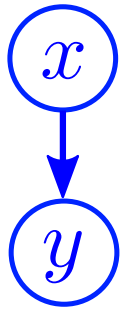
canonical correlation analysis

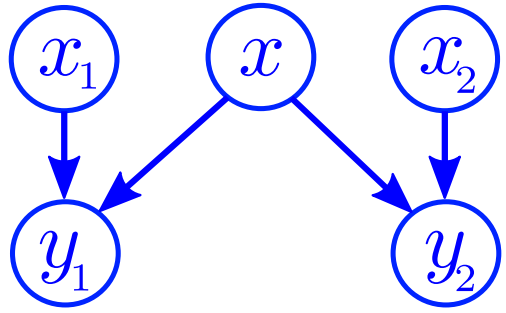
find linear projections of y_i that max correlation

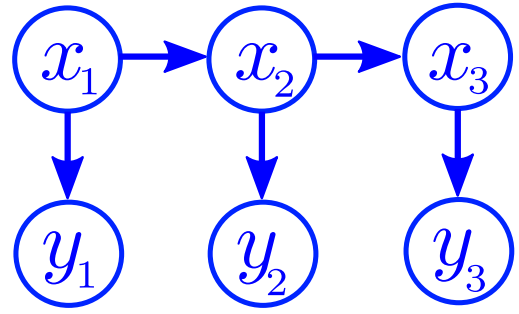
$$D = \sigma^2 \mathrm{I}$$

train: generalised eigenvalue problem

Dimensionality reduction using probabilistic models: a family of models

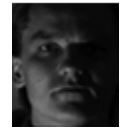






MODEL CLASS

audio



video

full linear

factor analysis (FA)

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$
$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

diagonal

inter-battery FA

$$p(x), p(x_i) = \mathcal{G}(x; 0, \mathrm{I})$$
$$p(y_i|x) = \mathcal{G}(y_i; \theta_i^{\mathrm{sh}} x + \theta_i^{\mathrm{pri}} x_i, D)$$

LGSSM

$$p(x_t|x_{t-1}) = \mathcal{G}(x_t; \Psi x_{t-1}, \Sigma)$$
$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

special linear

PCA

$$D = \sigma^2 \mathrm{I}$$

train: eigenvalue problem

canonical correlation analysis

find linear projections of y_i that max correlation

$$D = \sigma^2 \mathrm{I}$$

train: generalised eigenvalue problem

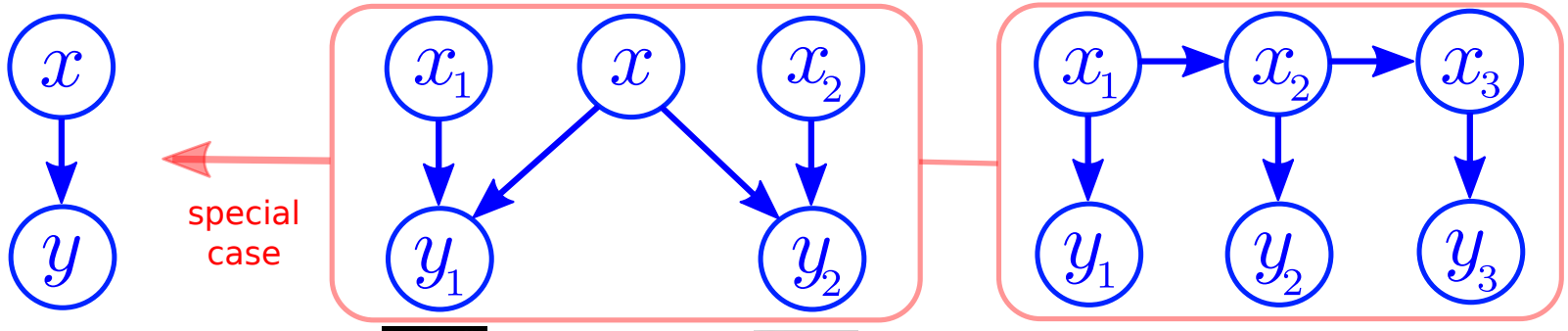
slow feature analysis

find slowest projections of unit variance

$$D = \sigma^2 \mathrm{I} \quad \Sigma = 1 - \Psi^2$$
$$\Psi = \mathrm{diag}(\psi_1, \ldots, \psi_K)$$

train: generalised eigenvalue problem

Dimensionality reduction using probabilistic models: a family of models



audio

video

MODEL CLASS

full linear

factor analysis (FA)

$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

diagonal

$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

inter-battery FA

$$p(x), p(x_i) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_i|x) = \mathcal{G}(y_i; \theta_i^{\mathrm{sh}} x + \theta_i^{\mathrm{pri}} x_i, D)$$

LGSSM

$$p(x_t|x_{t-1}) = \mathcal{G}(x_t; \Psi x_{t-1}, \Sigma)$$

$$p(y|x) = \mathcal{G}(y; \theta\, x, D)$$

special linear

PCA

$$D = \sigma^2 \mathrm{I}$$

train: eigenvalue problem

canonical correlation analysis

find linear projections of y_i that max correlation

$$D = \sigma^2 \mathrm{I}$$

train: generalised eigenvalue problem

slow feature analysis

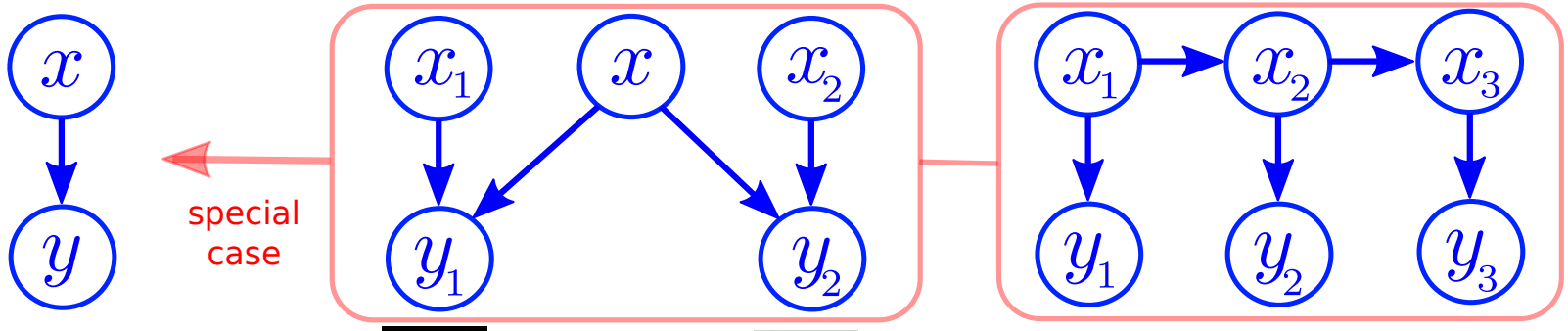
find slowest projections of unit variance

$$D = \sigma^2 \mathrm{I} \quad \Sigma = 1 - \Psi^2$$

$$\Psi = \mathrm{diag}(\psi_1, \ldots, \psi_K)$$

train: generalised eigenvalue problem

Dimensionality reduction using probabilistic models: a family of models



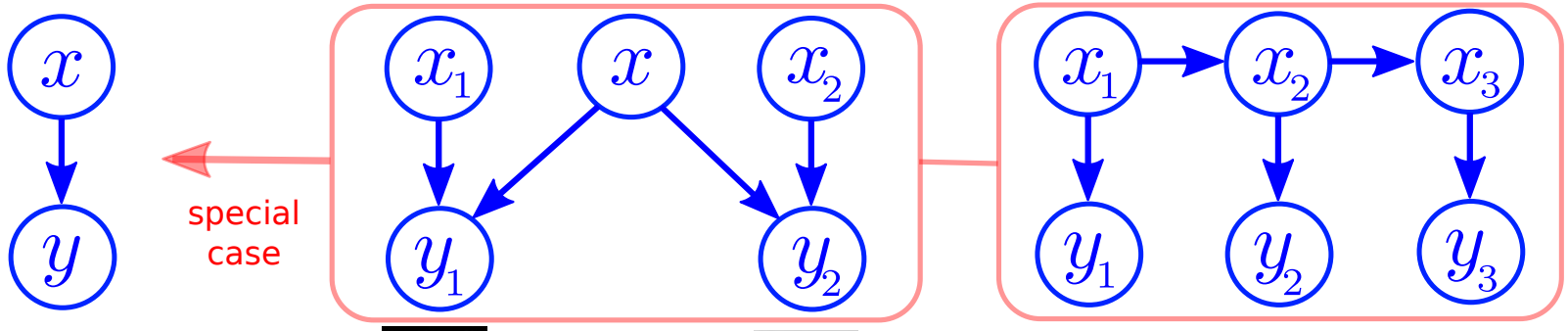
audio

video

MODEL CLASS

full linear	factor analysis (FA)	inter-battery FA	LGSSM
	$p(x) = \mathcal{G}(x; 0, \mathrm{I})$ $p(y\|x) = \mathcal{G}(y; \theta\, x, D)$ diagonal	$p(x), p(x_i) = \mathcal{G}(x; 0, \mathrm{I})$ $p(y_i\|x) = \mathcal{G}(y_i; \theta_i^{\mathrm{sh}} x + \theta_i^{\mathrm{pri}} x_i, D)$	$p(x_t\|x_{t-1}) = \mathcal{G}(x_t; \Psi x_{t-1}, \Sigma)$ $p(y\|x) = \mathcal{G}(y; \theta\, x, D)$
special linear	PCA	canonical correlation analysis	slow feature analysis
	$D = \sigma^2 \mathrm{I}$	find linear projections of y_i that max correlation $D = \sigma^2 \mathrm{I}$	find slowest projections of unit variance $D = \sigma^2 \mathrm{I} \quad \Sigma = 1 - \Psi^2$ $\Psi = \mathrm{diag}(\psi_1, \ldots, \psi_K)$
	train: eigenvalue problem	train: generalised eigenvalue problem	train: generalised eigenvalue problem
GP map y=f(x)	GP-LVM	multi-view GP-LVM	GP-dynamical system

Dimensionality reduction using probabilistic models: a family of models



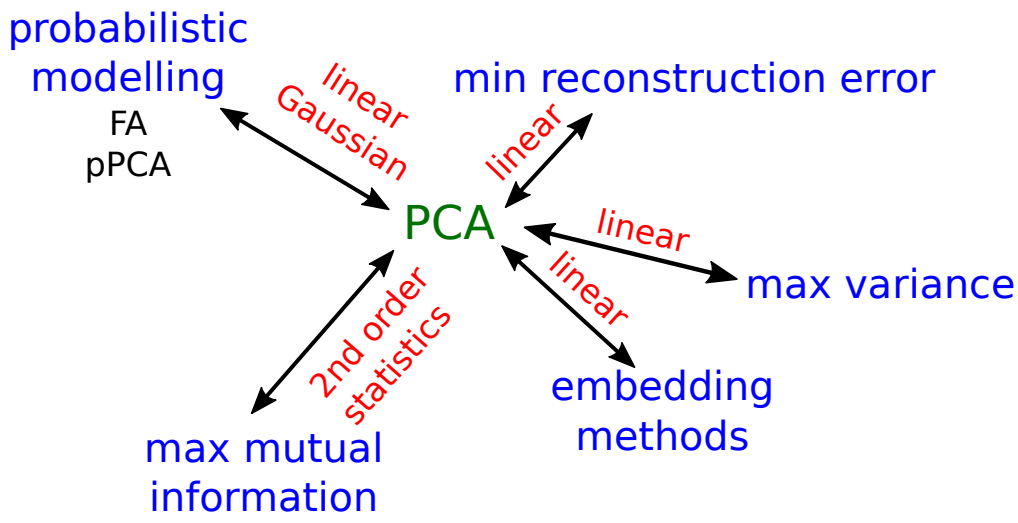

audio


video

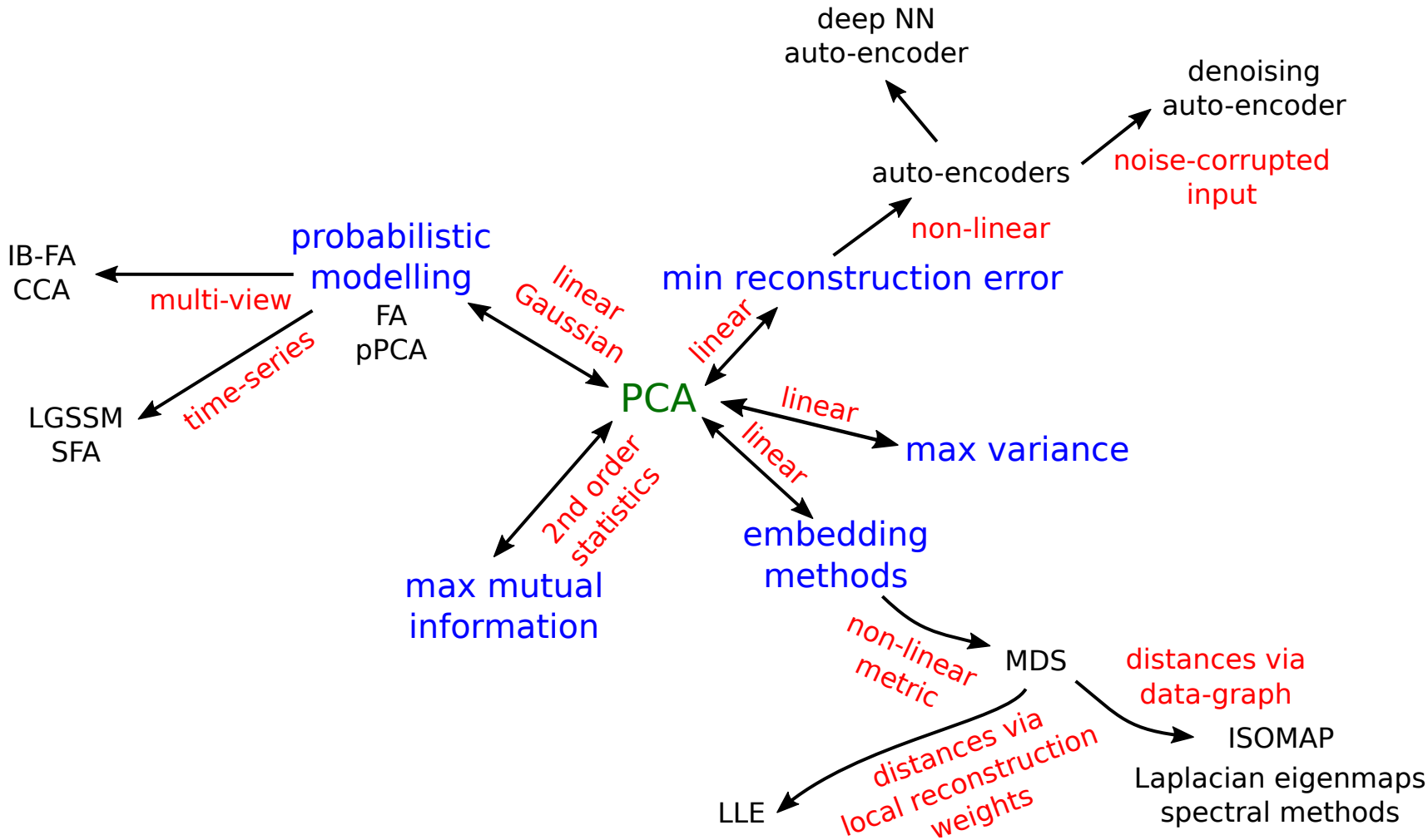
MODEL CLASS

full linear	factor analysis (FA)	inter-battery FA	LGSSM
	$p(x) = \mathcal{G}(x; 0, \mathrm{I})$ $p(y\|x) = \mathcal{G}(y; \theta\, x, D)$ diagonal	$p(x), p(x_i) = \mathcal{G}(x; 0, \mathrm{I})$ $p(y_i\|x) = \mathcal{G}(y_i; \theta_i^{\mathrm{sh}} x + \theta_i^{\mathrm{pri}} x_i, D)$	$p(x_t\|x_{t-1}) = \mathcal{G}(x_t; \Psi x_{t-1}, \Sigma)$ $p(y\|x) = \mathcal{G}(y; \theta\, x, D)$
special linear	PCA	canonical correlation analysis	slow feature analysis
	$D = \sigma^2 \mathrm{I}$	find linear projections of y_i that max correlation $D = \sigma^2 \mathrm{I}$	find slowest projections of unit variance $D = \sigma^2 \mathrm{I} \quad \Sigma = 1 - \Psi^2$ $\Psi = \mathrm{diag}(\psi_1, \ldots, \psi_K)$
	train: eigenvalue problem	train: generalised eigenvalue problem	train: generalised eigenvalue problem
GP map y=f(x)	GP-LVM	multi-view GP-LVM	GP-dynamical system
other	many e.g. ICA	information bottleneck style and content	many e.g. GP-SSM

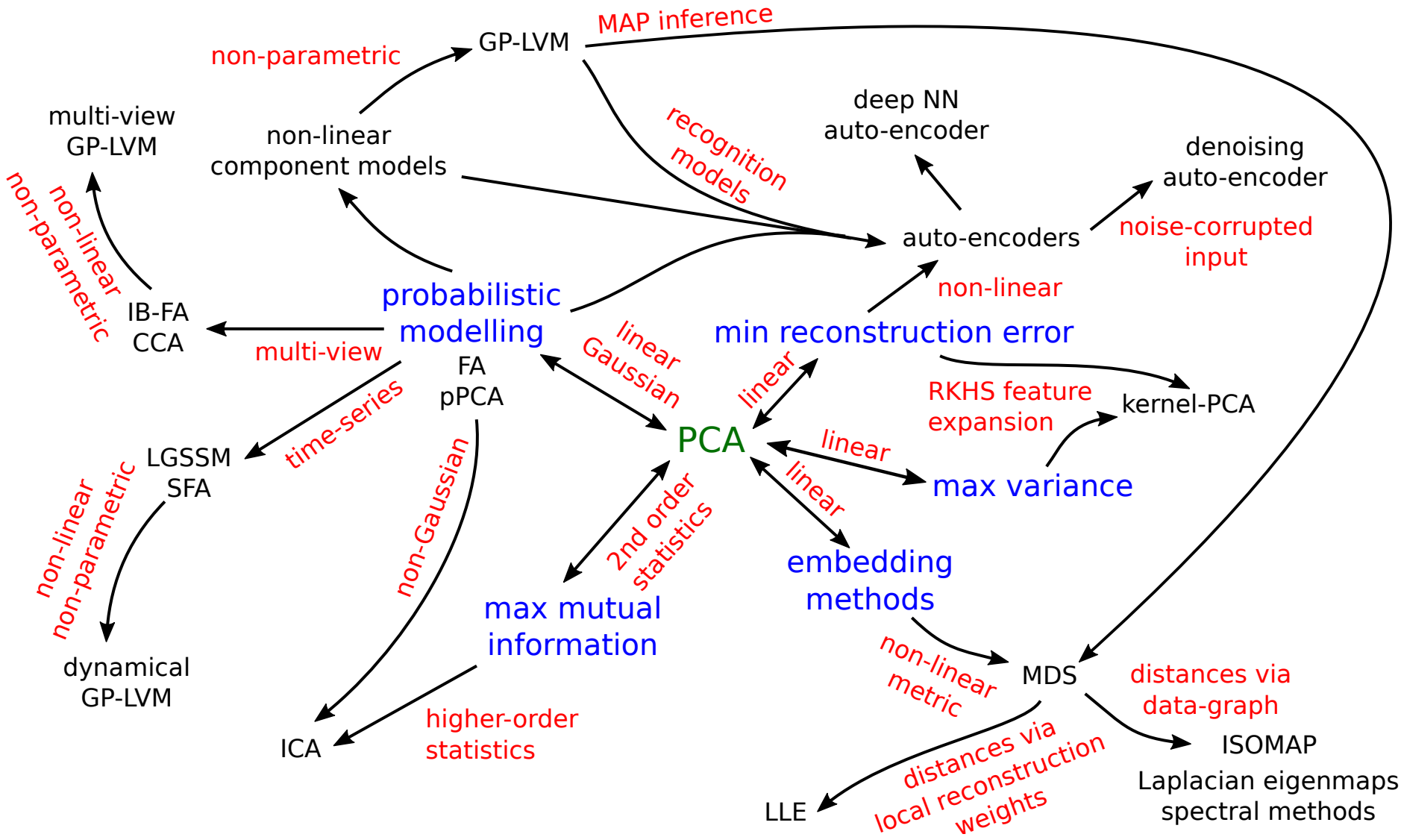
Roadmap so far...



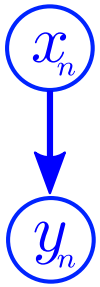
Roadmap so far...



Roadmap so far...



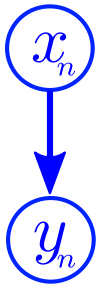
Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

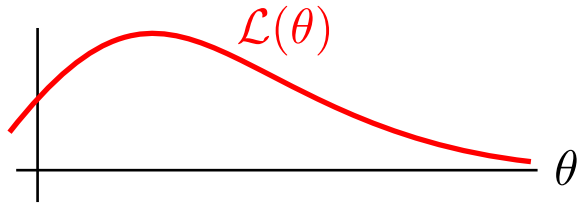
$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

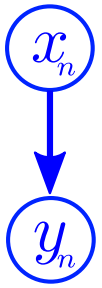
$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



Goal: learn parameters via approximate maximum-likelihood

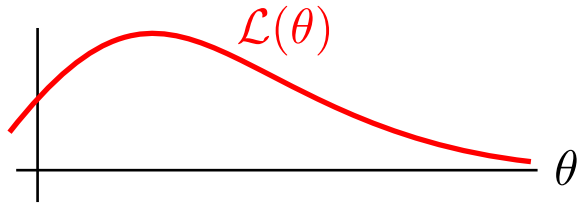
$$\mathcal{L}(\theta) = \sum_n \log p(y_n|\theta)$$

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

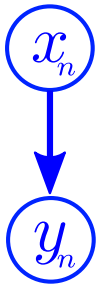
$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



Goal: learn parameters via approximate maximum-likelihood

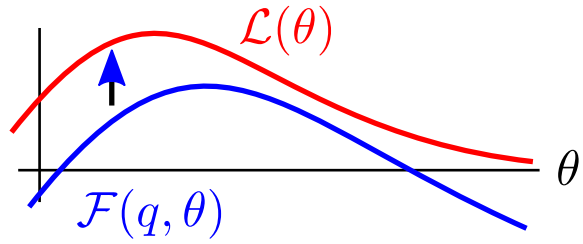
$$\mathcal{L}(\theta) = \sum_n \log p(y_n|\theta) = \sum_n \log \int p(y_n, x_n|\theta) \mathrm{d}x_n = \sum_n \log \int \frac{q(x_n)}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n$$

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



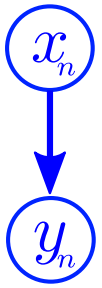
Goal: learn parameters via approximate maximum-likelihood

$$\mathcal{L}(\theta) = \sum_n \log p(y_n|\theta) = \sum_n \log \int p(y_n, x_n|\theta) \mathrm{d}x_n = \sum_n \log \int \frac{q(x_n)}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n$$

Jensen's

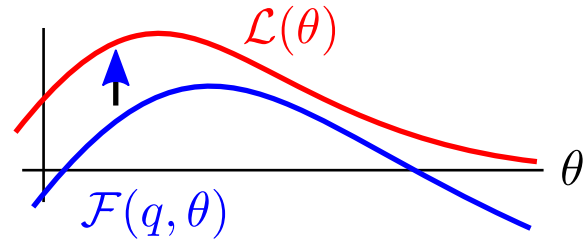
$$\mathcal{L}(\theta) \geq \sum_n \int q(x_n) \log \frac{1}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n = \mathcal{F}(q,\theta)$$ ← free-energy

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



Goal: learn parameters via approximate maximum-likelihood

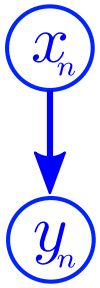
$$\mathcal{L}(\theta) = \sum_n \log p(y_n|\theta) = \sum_n \log \int p(y_n, x_n|\theta) \mathrm{d}x_n = \sum_n \log \int \frac{q(x_n)}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n$$

Jensen's

$$\mathcal{L}(\theta) \geq \sum_n \int q(x_n) \log \frac{1}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n = \mathcal{F}(q,\theta)$$ ← free-energy

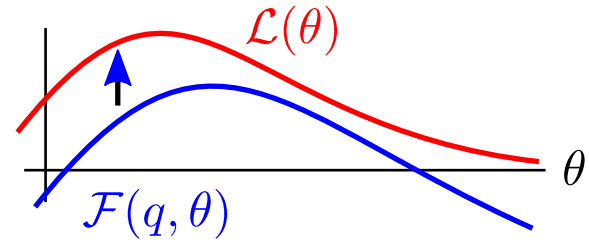
$$\mathcal{F}(q,\theta) = \sum_n \int q(x_n) \log p(y_n|x_n,\theta) \mathrm{d}x_n \quad + \quad \sum_n \int q(x_n) \log \frac{p(x_n|\theta)}{q(x_n)} \mathrm{d}x_n$$

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



Goal: learn parameters via approximate maximum-likelihood

$$\mathcal{L}(\theta) = \sum_n \log p(y_n|\theta) = \sum_n \log \int p(y_n, x_n|\theta) \mathrm{d}x_n = \sum_n \log \int \frac{q(x_n)}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n$$

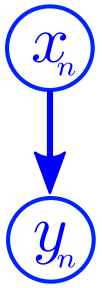
Jensen's

$$\mathcal{L}(\theta) \geq \sum_n \int q(x_n) \log \frac{1}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n = \mathcal{F}(q,\theta)$$ ← free-energy

$$\mathcal{F}(q,\theta) = \sum_n \int q(x_n) \log p(y_n|x_n,\theta) \mathrm{d}x_n + \sum_n \int q(x_n) \log \frac{p(x_n|\theta)}{q(x_n)} \mathrm{d}x_n$$

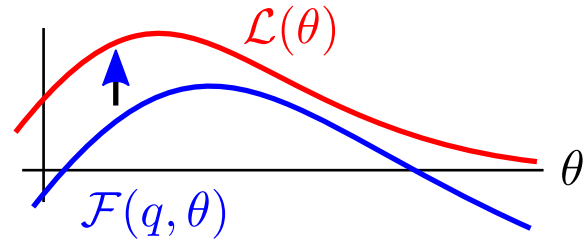
$$\mathcal{F}(q,\theta) = -\sum_n \frac{1}{2\sigma^2} \langle ||y_n - f_\theta(x_n)||^2 \rangle_{q(x_n)} - \frac{ND}{2} \log \sigma^2 - \sum_n \mathrm{KL}(q(x_n)||p(x_n))$$

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



Goal: learn parameters via approximate maximum-likelihood

$$\mathcal{L}(\theta) = \sum_n \log p(y_n|\theta) = \sum_n \log \int p(y_n, x_n|\theta) \mathrm{d}x_n = \sum_n \log \int \frac{q(x_n)}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n$$

Jensen's

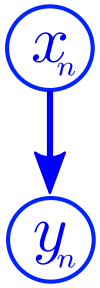
$$\mathcal{L}(\theta) \geq \sum_n \int q(x_n) \log \frac{1}{q(x_n)} p(y_n, x_n|\theta) \mathrm{d}x_n = \mathcal{F}(q,\theta)$$ ← free-energy

$$\mathcal{F}(q,\theta) = \sum_n \int q(x_n) \log p(y_n|x_n,\theta) \mathrm{d}x_n + \sum_n \int q(x_n) \log \frac{p(x_n|\theta)}{q(x_n)} \mathrm{d}x_n$$

reconstruction cost — soft-constraint

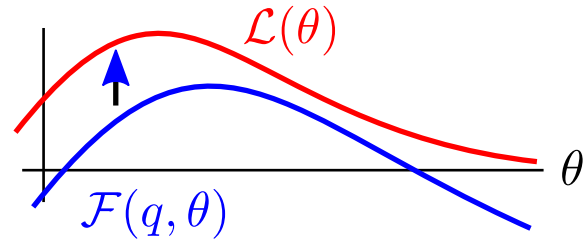
$$\mathcal{F}(q,\theta) = -\sum_n \frac{1}{2\sigma^2} \langle ||y_n - f_\theta(x_n)||^2 \rangle_{q(x_n)} - \frac{ND}{2} \log \sigma^2 - \sum_n \mathrm{KL}(q(x_n)||p(x_n))$$

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



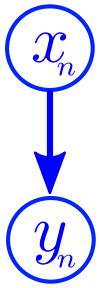
Goal: learn parameters via approximate maximum-likelihood

flavours of variational inference:

reconstruction cost soft-constraint

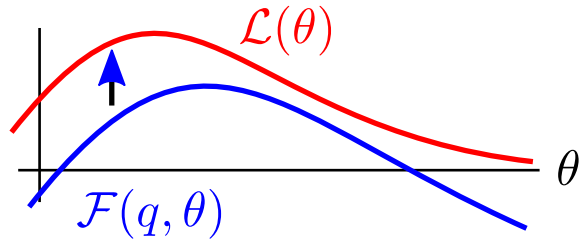
$$\mathcal{F}(q,\theta) = -\sum_n \frac{1}{2\sigma^2} \langle ||y_n - f_\theta(x_n)||^2 \rangle_{q(x_n)} - \frac{ND}{2} \log \sigma^2 - \sum_n \mathrm{KL}(q(x_n)||p(x_n))$$

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



Goal: learn parameters via approximate maximum-likelihood

flavours of variational inference:

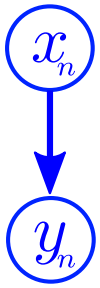
fixed family $\quad q(x_n) = \mathcal{G}(x_n; \mu_{q_n}, \Sigma_{q_n})$

$$\arg\max_{\theta, \mu_{q_{1:N}}, \Sigma_{q_{1:N}}} \mathcal{F}(\mu_{q_{1:N}}, \Sigma_{q_{1:N}}, \theta)$$

reconstruction cost soft-constraint

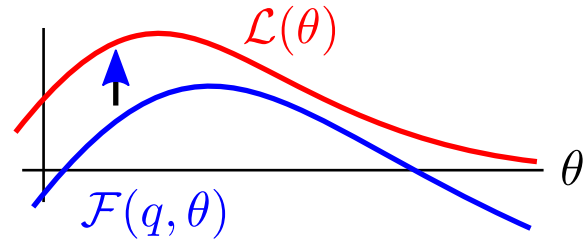
$$\mathcal{F}(q,\theta) = -\sum_n \frac{1}{2\sigma^2} \langle ||y_n - f_\theta(x_n)||^2 \rangle_{q(x_n)} - \frac{ND}{2} \log \sigma^2 - \sum_n \mathrm{KL}(q(x_n)||p(x_n))$$

Probabilistic inference as an auto-encoder



$p(x_n|\theta)$

$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$



Goal: learn parameters via approximate maximum-likelihood

flavours of variational inference:

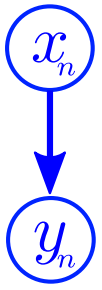
fixed family $\quad q(x_n) = \mathcal{G}(x_n; \mu_{q_n}, \Sigma_{q_n})$ $\qquad \arg\max_{\theta, \mu_{q_{1:N}}, \Sigma_{q_{1:N}}} \mathcal{F}(\mu_{q_{1:N}}, \Sigma_{q_{1:N}}, \theta)$

structured $\quad q(x_n) = \prod_{k=1}^K q_k(x_{nk})$ $\qquad \arg\max_{\{q_k(x_{nk})\}_{k=1,n=1}^{K,N}, \theta} \mathcal{F}(\{q_k(x_{nk})\}_{k=1,n=1}^{K,N}, \theta)$

reconstruction cost $\qquad$ soft-constraint

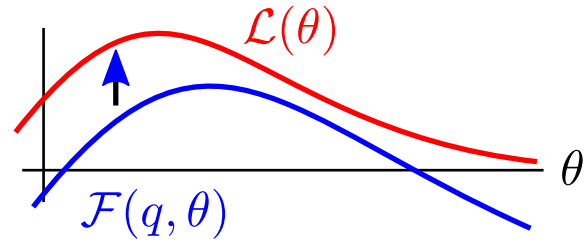
$$\mathcal{F}(q,\theta) = -\sum_n \frac{1}{2\sigma^2} \langle ||y_n - f_\theta(x_n)||^2 \rangle_{q(x_n)} - \frac{ND}{2} \log \sigma^2 - \sum_n \mathrm{KL}(q(x_n)||p(x_n))$$

Probabilistic inference as an auto-encoder



$p(x_n|\theta)$

$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$



Goal: learn parameters via approximate maximum-likelihood

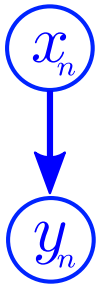
flavours of variational inference:

fixed family	$q(x_n) = \mathcal{G}(x_n; \mu_{q_n}, \Sigma_{q_n})$	$\arg\max_{\theta, \mu_{q_{1:N}}, \Sigma_{q_{1:N}}} \mathcal{F}(\mu_{q_{1:N}}, \Sigma_{q_{1:N}}, \theta)$
structured	$q(x_n) = \prod_{k=1}^K q_k(x_{nk})$	$\arg\max_{\{q_k(x_{nk})\}_{k=1,n=1}^{K,N}, \theta} \mathcal{F}(\{q_k(x_{nk})\}_{k=1,n=1}^{K,N}, \theta)$
recognition model	$q_\phi(x_n) = \mathcal{G}(x_n; \mu_\phi(y_n), \Sigma_\phi(y_n))$	$\arg\max_{\phi,\theta} \mathcal{F}(\phi, \theta)$ variational auto-encoder

reconstruction cost soft-constraint

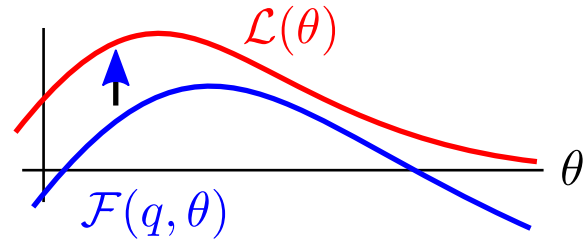
$$\mathcal{F}(q,\theta) = -\sum_n \frac{1}{2\sigma^2} \langle ||y_n - f_\theta(x_n)||^2 \rangle_{q(x_n)} - \frac{ND}{2} \log \sigma^2 - \sum_n \mathrm{KL}(q(x_n)||p(x_n))$$

Probabilistic inference as an auto-encoder



$p(x_n|\theta)$

$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$



Goal: learn parameters via approximate maximum-likelihood

flavours of variational inference:

Roweis and Ghahramani, 1999 (AEs and exact inference)
Kingma and Welling, 2013 (AEs and approximate inference)

fixed family $\quad q(x_n) = \mathcal{G}(x_n; \mu_{q_n}, \Sigma_{q_n})$ $\quad \arg\max_{\theta, \mu_{q_{1:N}}, \Sigma_{q_{1:N}}} \mathcal{F}(\mu_{q_{1:N}}, \Sigma_{q_{1:N}}, \theta)$

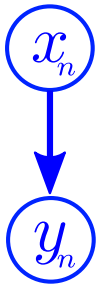
structured $\quad q(x_n) = \prod_{k=1}^K q_k(x_{nk})$ $\quad \arg\max_{\{q_k(x_{nk})\}_{k=1,n=1}^{K,N}, \theta} \mathcal{F}(\{q_k(x_{nk})\}_{k=1,n=1}^{K,N}, \theta)$

recognition model $\quad q_\phi(x_n) = \mathcal{G}(x_n; \mu_\phi(y_n), \Sigma_\phi(y_n))$ $\quad \arg\max_{\phi,\theta} \mathcal{F}(\phi, \theta)$ variational auto-encoder

reconstruction cost soft-constraint

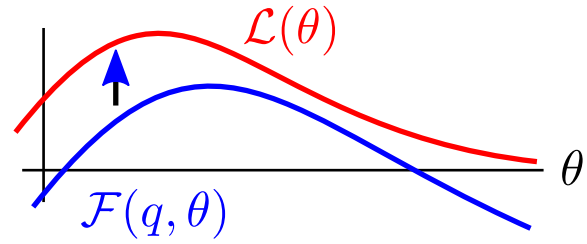
$$\mathcal{F}(q,\theta) = -\sum_n \frac{1}{2\sigma^2} \langle ||y_n - f_\theta(x_n)||^2 \rangle_{q(x_n)} - \frac{ND}{2} \log \sigma^2 - \sum_n \mathrm{KL}(q(x_n)||p(x_n))$$

Probabilistic inference as an auto-encoder



$$p(x_n|\theta)$$

$$p(y_n|x_n,\theta) = \mathcal{G}(y_n; f_\theta(x_n), \sigma^2 \mathrm{I})$$



Goal: learn parameters via approximate maximum-likelihood

flavours of variational inference:

Roweis and Ghahramani, 1999 (AEs and exact inference)
Kingma and Welling, 2013 (AEs and approximate inference)

fixed family $\quad q(x_n) = \mathcal{G}(x_n; \mu_{q_n}, \Sigma_{q_n})$ $\qquad \arg\max_{\theta, \mu_{q_{1:N}}, \Sigma_{q_{1:N}}} \mathcal{F}(\mu_{q_{1:N}}, \Sigma_{q_{1:N}}, \theta)$

structured $\quad q(x_n) = \prod_{k=1}^K q_k(x_{nk})$ $\qquad \arg\max_{\{q_k(x_{nk})\}_{k=1,n=1}^{K,N}, \theta} \mathcal{F}(\{q_k(x_{nk})\}_{k=1,n=1}^{K,N}, \theta)$

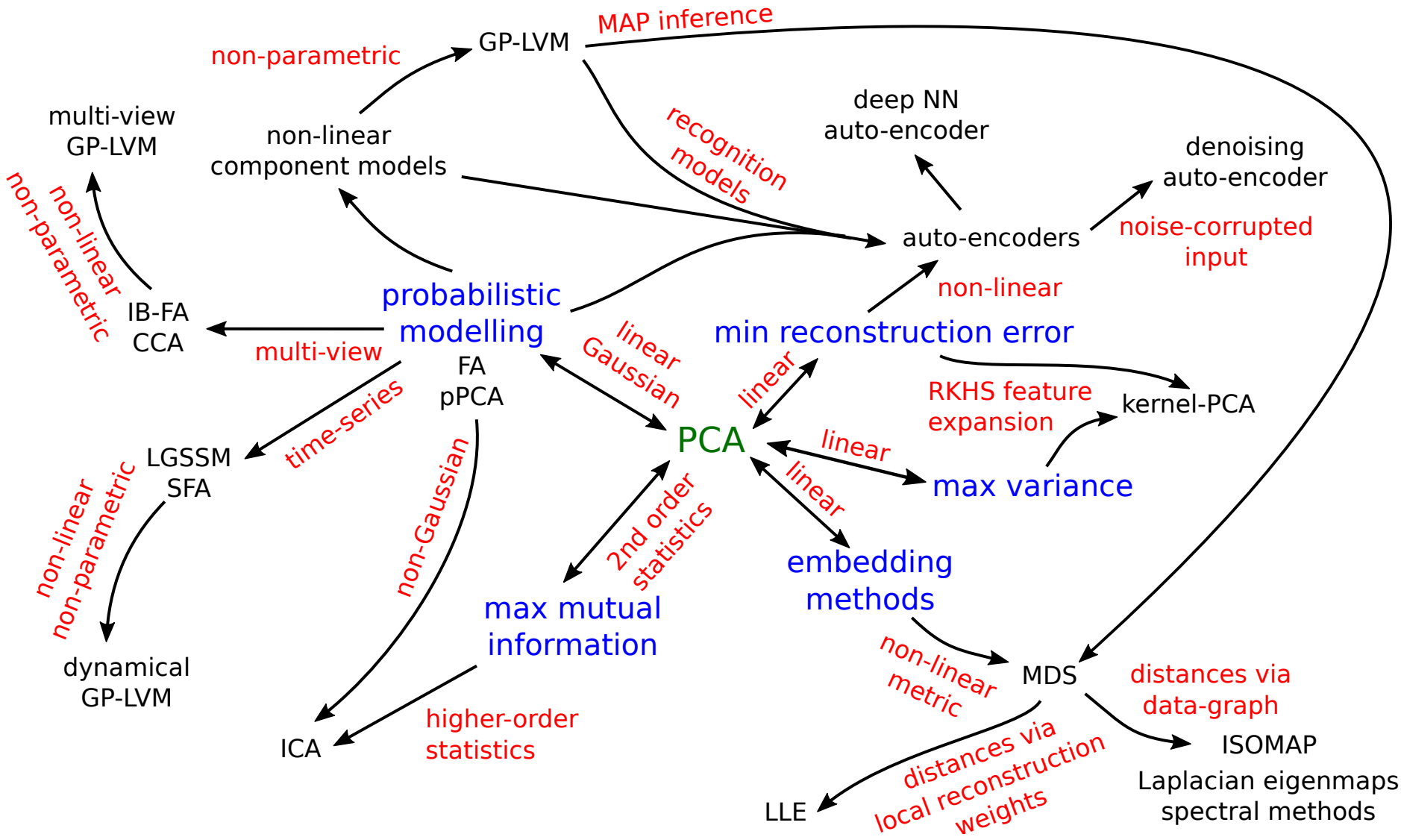
recognition model $\quad q_\phi(x_n) = \mathcal{G}(x_n; \mu_\phi(y_n), \Sigma_\phi(y_n))$ $\qquad \arg\max_{\phi,\theta} \mathcal{F}(\phi, \theta)$ variational auto-encoder

GP LVM $\quad q_\phi(x) = \delta(x - g_\phi(y))$

reconstruction cost soft-constraint

$$\mathcal{F}(q,\theta) = -\sum_n \frac{1}{2\sigma^2} \langle ||y_n - f_\theta(x_n)||^2 \rangle_{q(x_n)} - \frac{ND}{2} \log \sigma^2 - \sum_n \mathrm{KL}(q(x_n)||p(x_n))$$

Dimensionality reduction: conceptual space



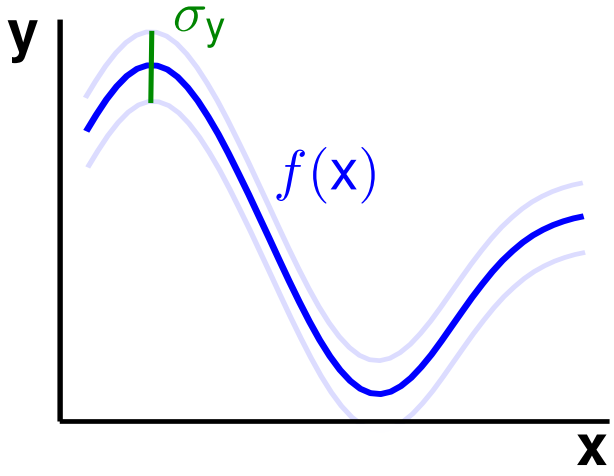
Additional Slides

Gaussian Process Regression Model: Recap

Generative model (like non-linear regression)

$$\mathsf{y}(\mathsf{x}) = f(\mathsf{x}) + \epsilon\sigma_{\mathsf{y}}$$

$$p(\epsilon) = \mathcal{N}(0, 1)$$



place GP prior over the non-linear function

$$p(f(\mathsf{x})|\theta) = \mathcal{GP}(0, \mathsf{K}(\mathsf{x}, \mathsf{x}'))$$

"multivariate Gaussian of infinite dimension"
(any finite subset of variables are multivariate Gaussian)

$$\mathsf{K}(\mathsf{x}, \mathsf{x}') = \sigma^2 \exp\left(-\frac{1}{2l^2}(\mathsf{x} - \mathsf{x}')^2\right)$$ (smoothly wiggling functions expected)

since the sum of two Gaussians is a Gaussian, the model induces a GP over $\mathsf{y}(\mathsf{x})$

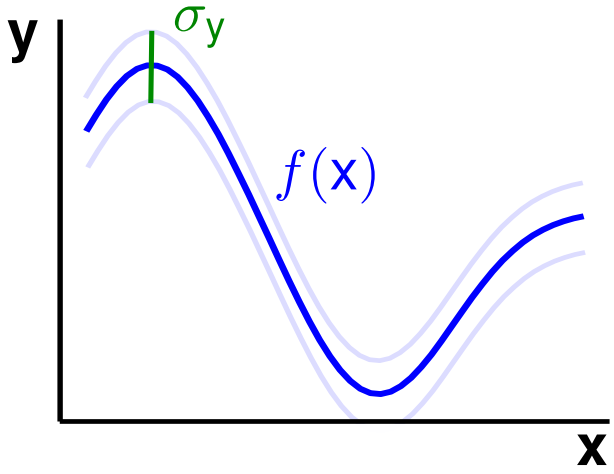
$$p(\mathsf{y}(\mathsf{x})|\theta) = \mathcal{GP}(0, \mathsf{K}(\mathsf{x}, \mathsf{x}') + \mathrm{I}\sigma_{\mathsf{y}}^2)$$

Gaussian Process Regression Model: Recap

Generative model (like non-linear regression)

$$\mathsf{y}(\mathsf{x}) = f(\mathsf{x}) + \epsilon\sigma_{\mathsf{y}}$$

$$p(\epsilon) = \mathcal{N}(0, 1)$$



place GP prior over the non-linear function

$$p(f(\mathsf{x})|\theta) = \mathcal{GP}(0, \mathsf{K}(\mathsf{x}, \mathsf{x}'))$$

"multivariate Gaussian of infinite dimension"
(any finite subset of variables are multivariate Gaussian)

$$\mathsf{K}(\mathsf{x}, \mathsf{x}') = \sigma^2 \exp\left(-\frac{1}{2l^2}(\mathsf{x} - \mathsf{x}')^2\right)$$ (smoothly wiggling functions expected)

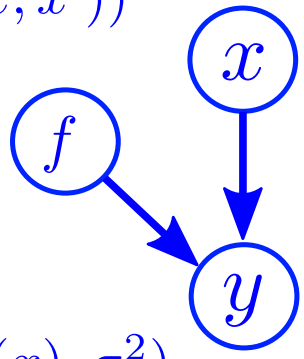
since the sum of two Gaussians is a Gaussian, the model induces a GP over $\mathsf{y}(\mathsf{x})$

$$p(\mathsf{y}(\mathsf{x})|\theta) = \mathcal{GP}(0, \mathsf{K}(\mathsf{x}, \mathsf{x}') + \mathrm{I}\sigma_{\mathsf{y}}^2)$$

multi-output regression $\mathsf{y}_{\mathsf{i}}(\mathsf{x}) = f_{\mathsf{i}}(\mathsf{x}) + \epsilon_{\mathsf{i}}\sigma_{\mathsf{y}}$ $p(f_{\mathsf{i}}(\mathsf{x})|\theta) = \mathcal{GP}(0, \mathsf{K}_{\mathsf{i}}(\mathsf{x}, \mathsf{x}'))$

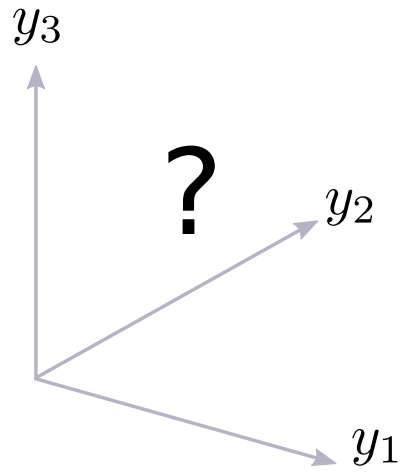
Gaussian Process Latent Variable Model

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

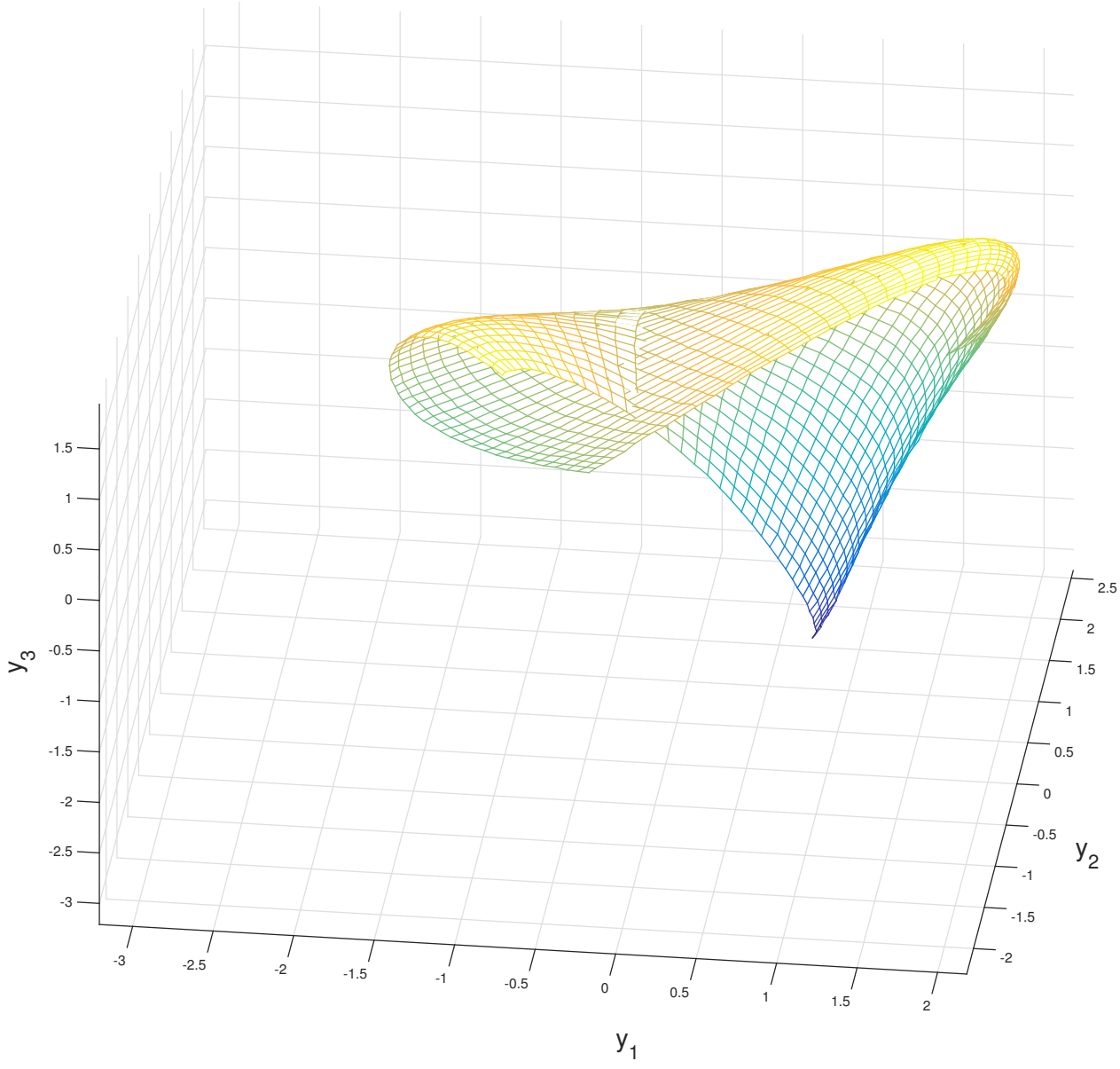
$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$



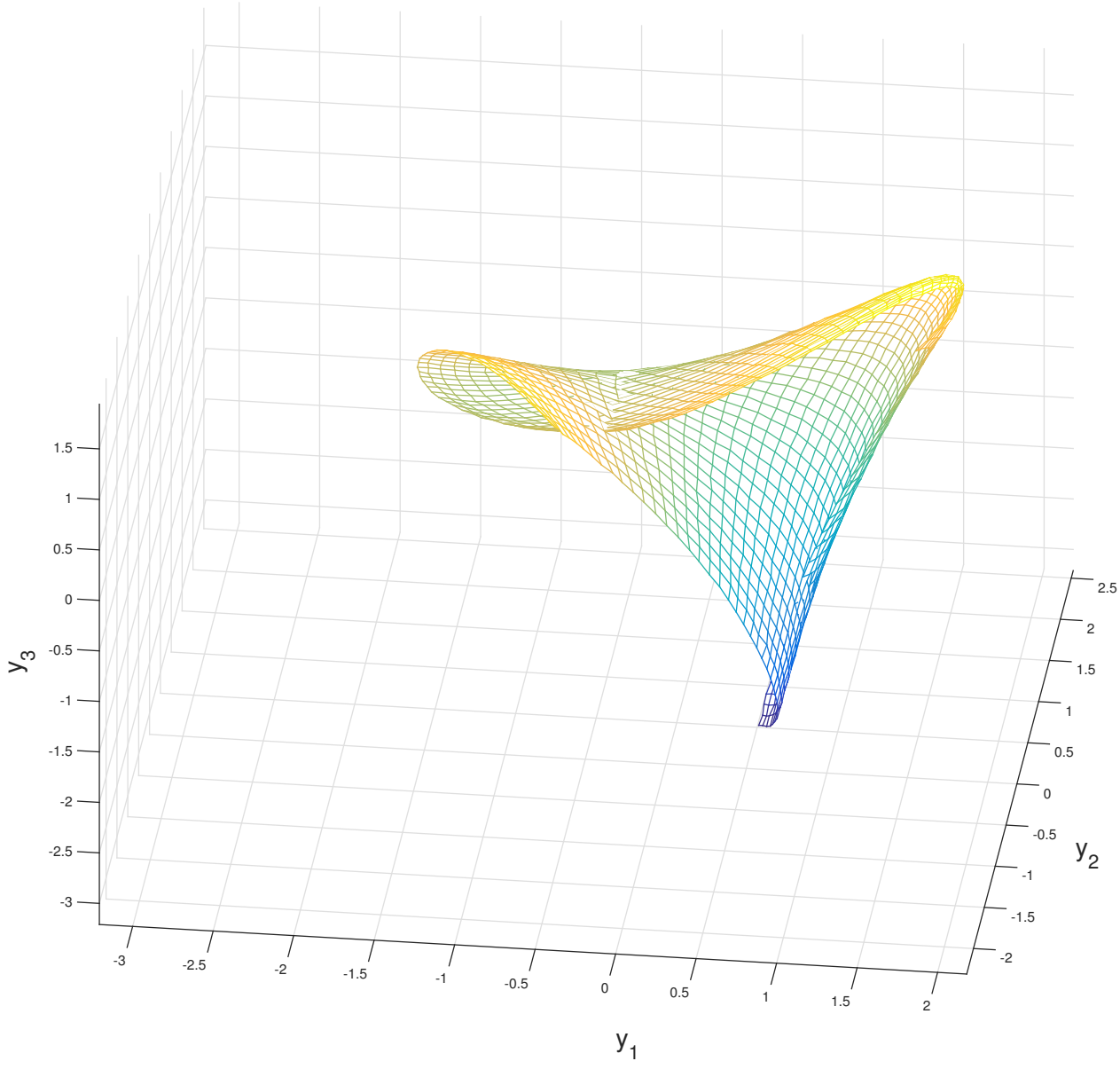
toy example:
2 dimensional latents x
3 dimensional observed y
sample f
evaluate observed variables y
corresponding to a grid of x

what does this look like?

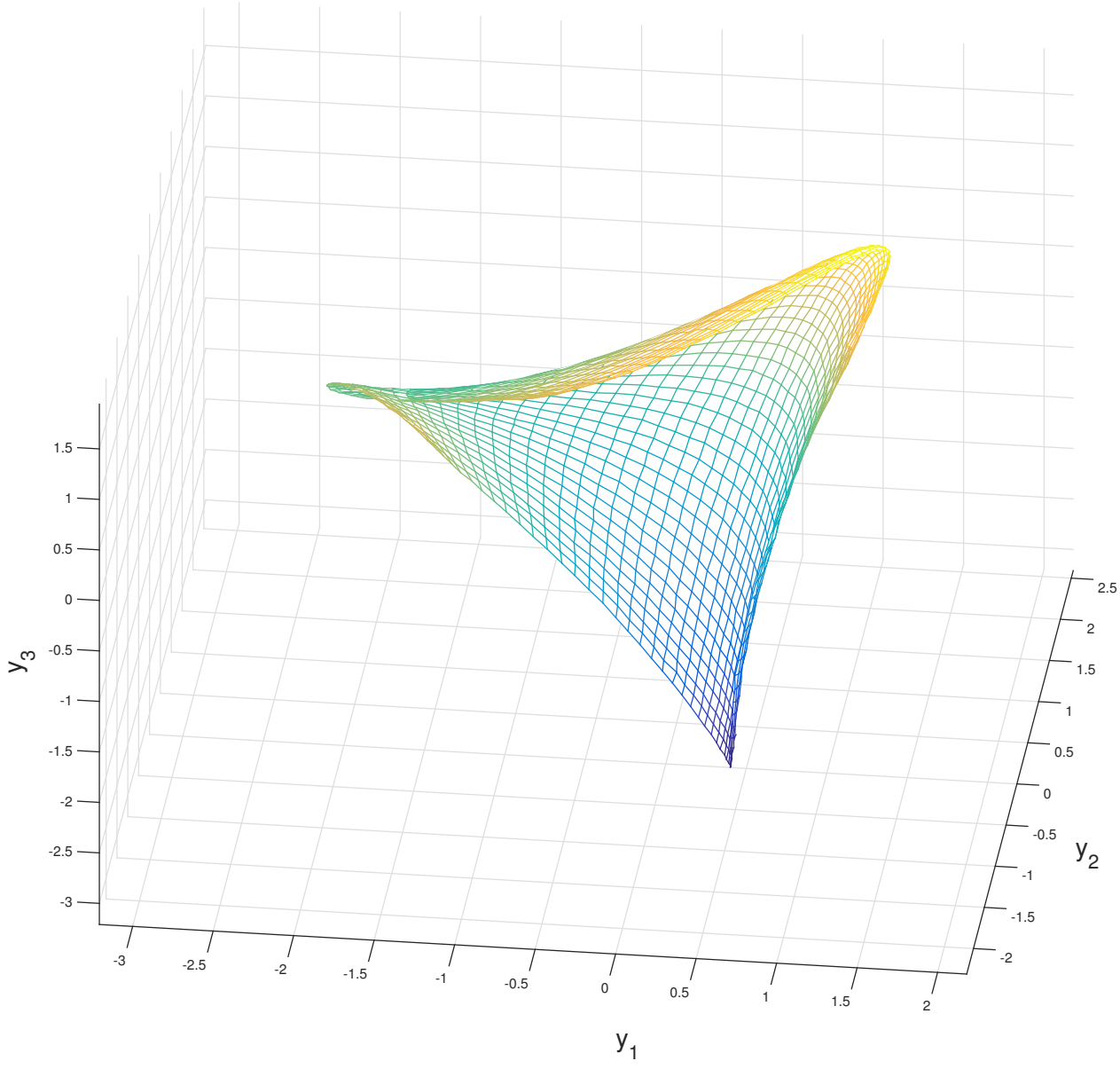
Gaussian Process Latent Variable Model: manifold samples



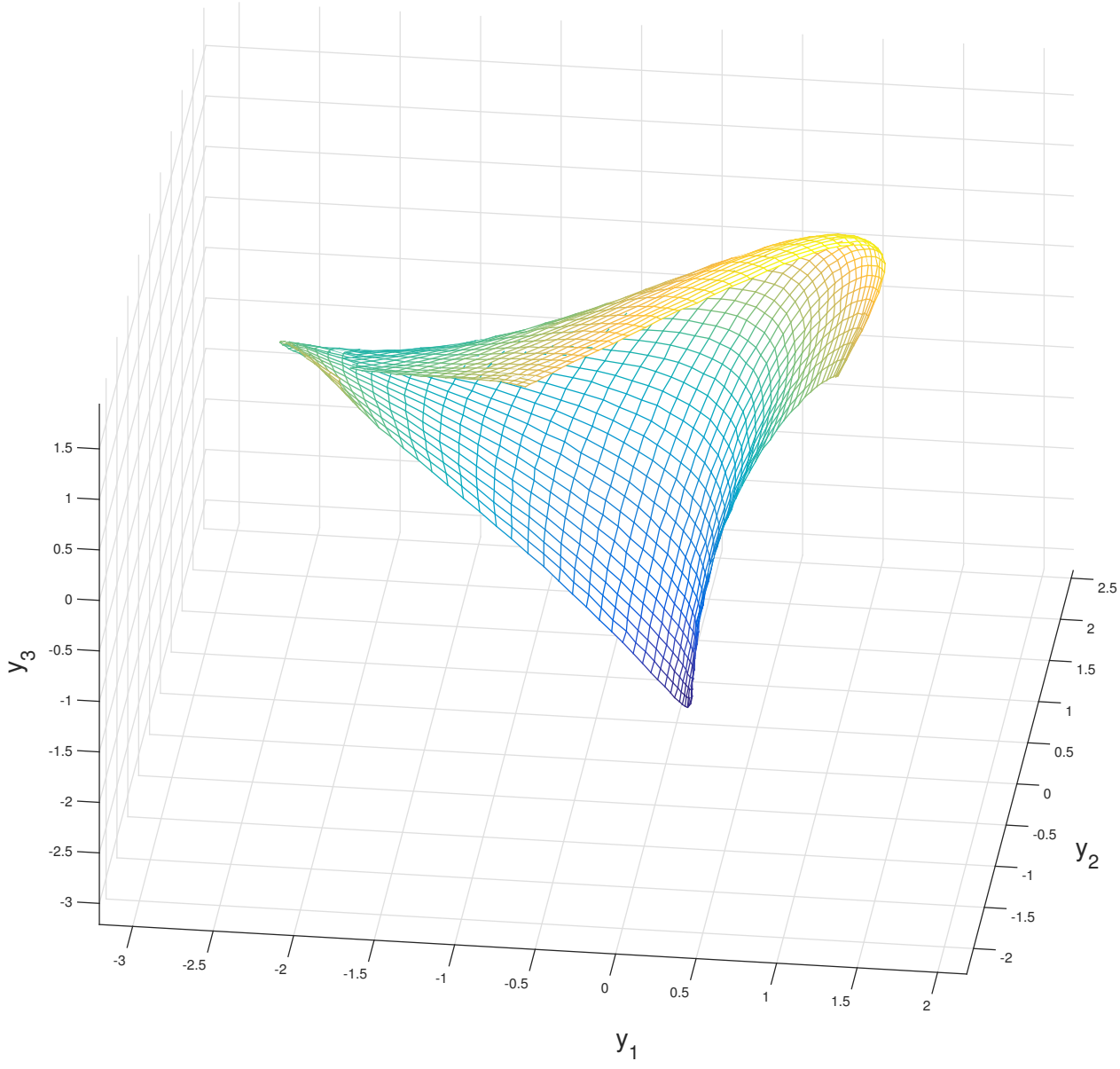
Gaussian Process Latent Variable Model: manifold samples



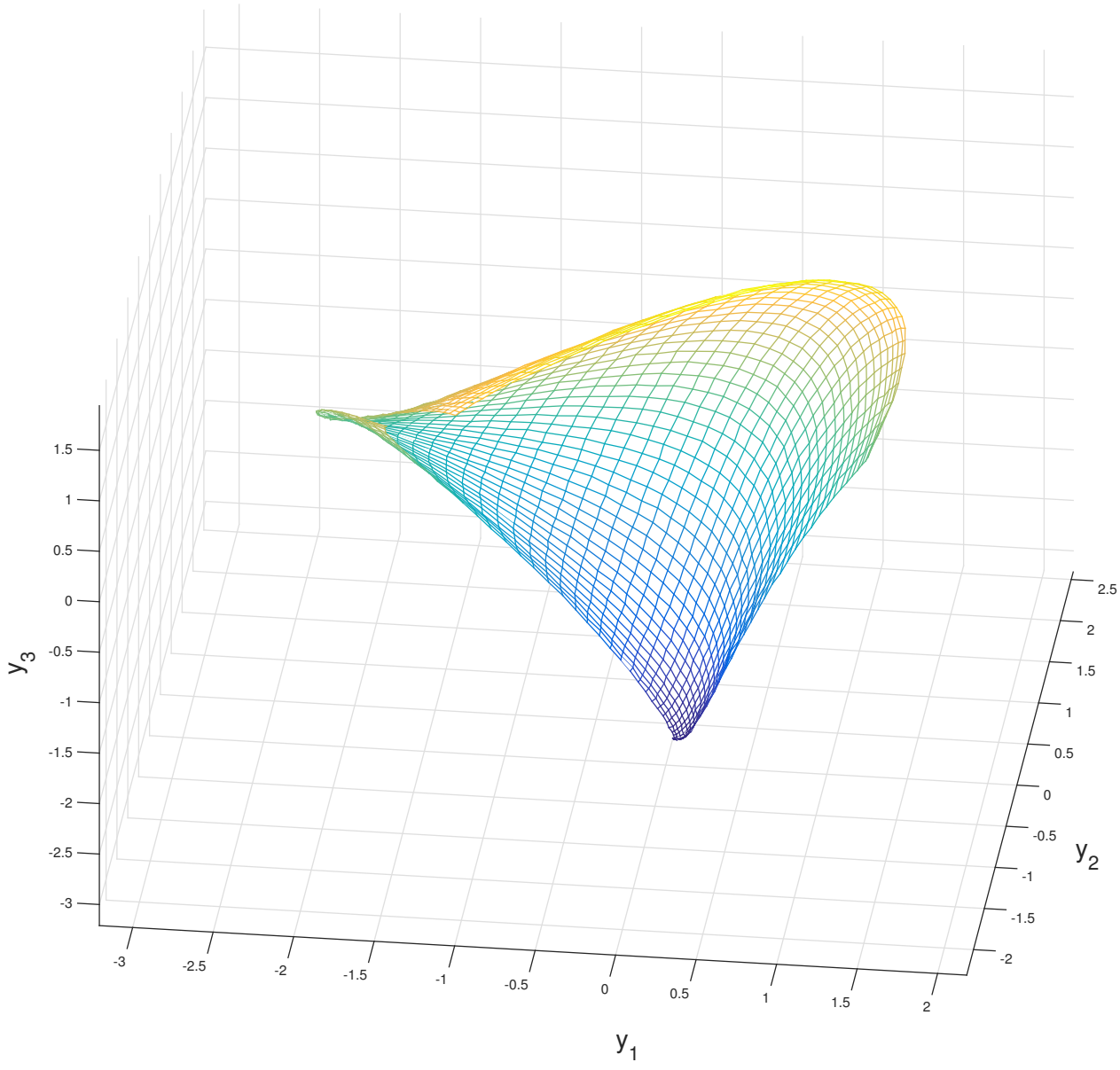
Gaussian Process Latent Variable Model: manifold samples



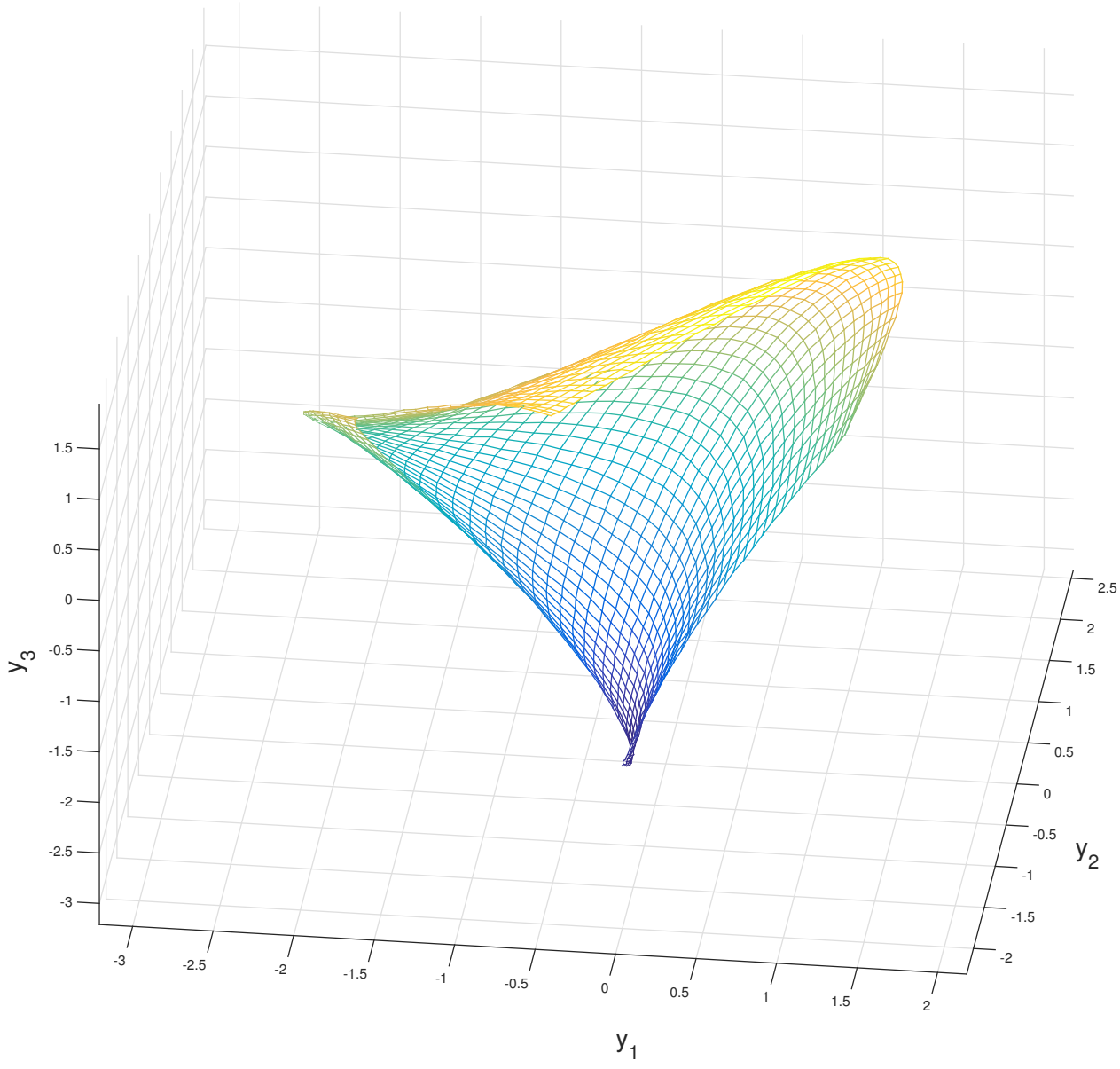
Gaussian Process Latent Variable Model: manifold samples



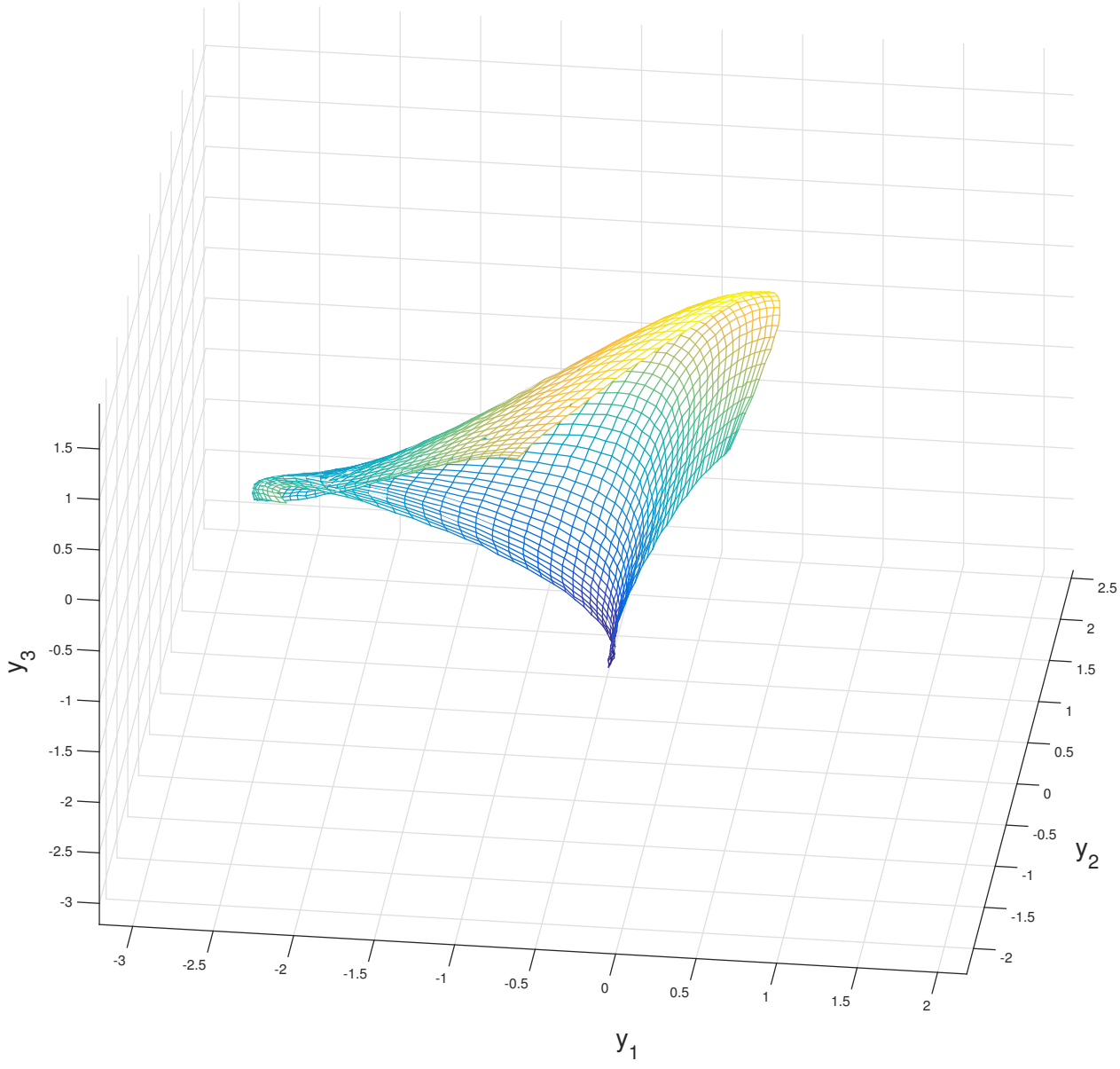
Gaussian Process Latent Variable Model: manifold samples



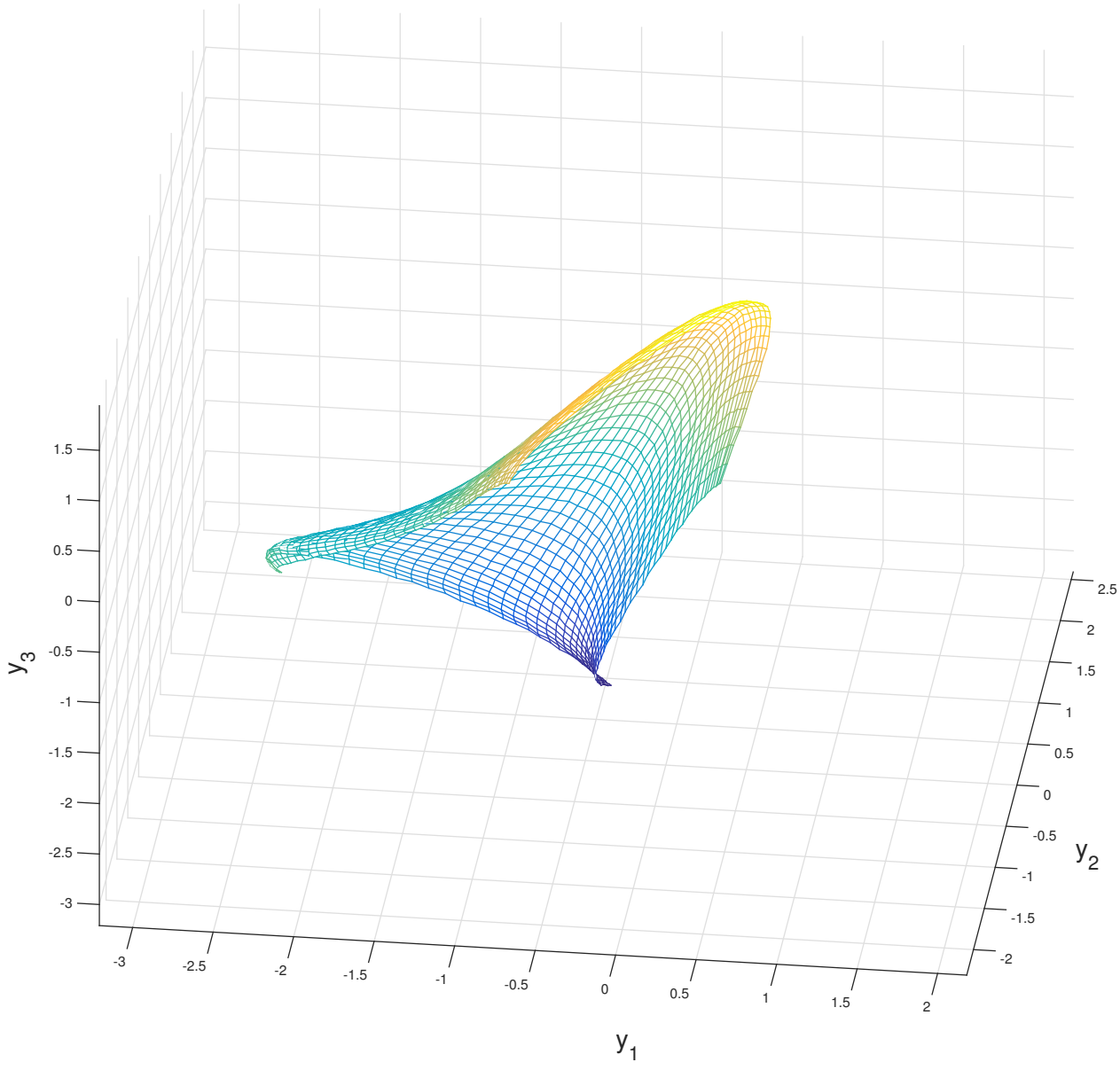
Gaussian Process Latent Variable Model: manifold samples



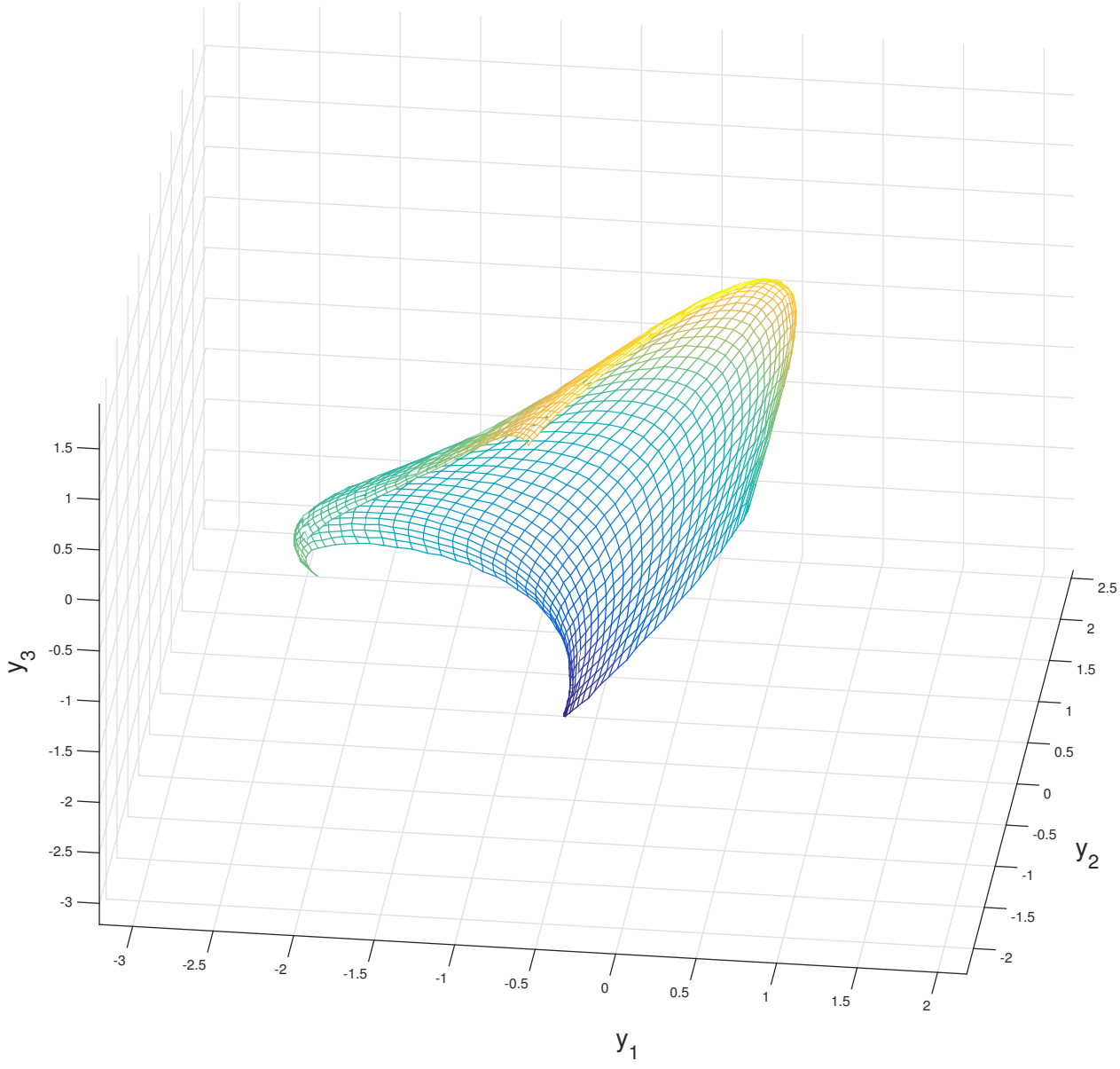
Gaussian Process Latent Variable Model: manifold samples



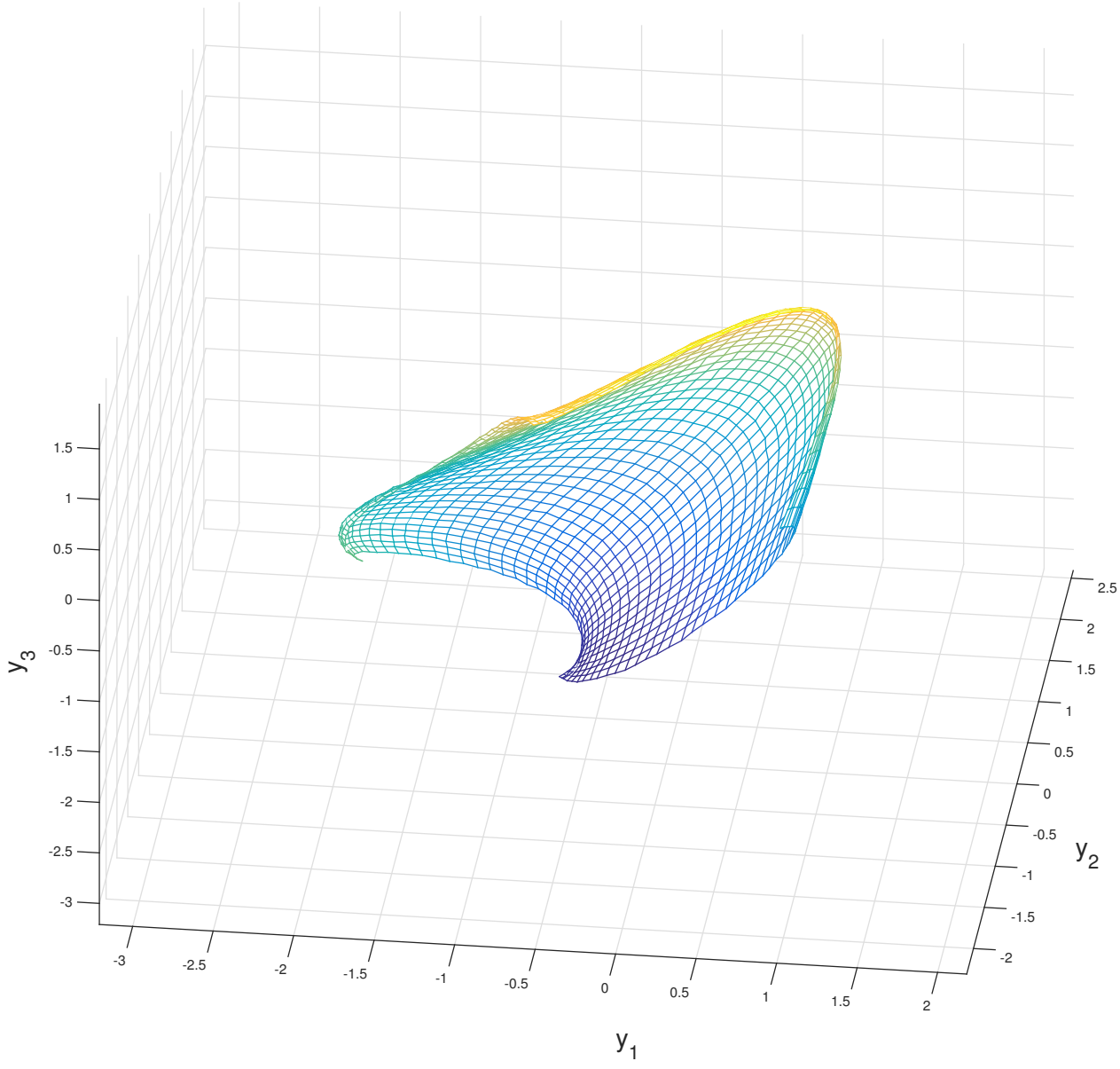
Gaussian Process Latent Variable Model: manifold samples



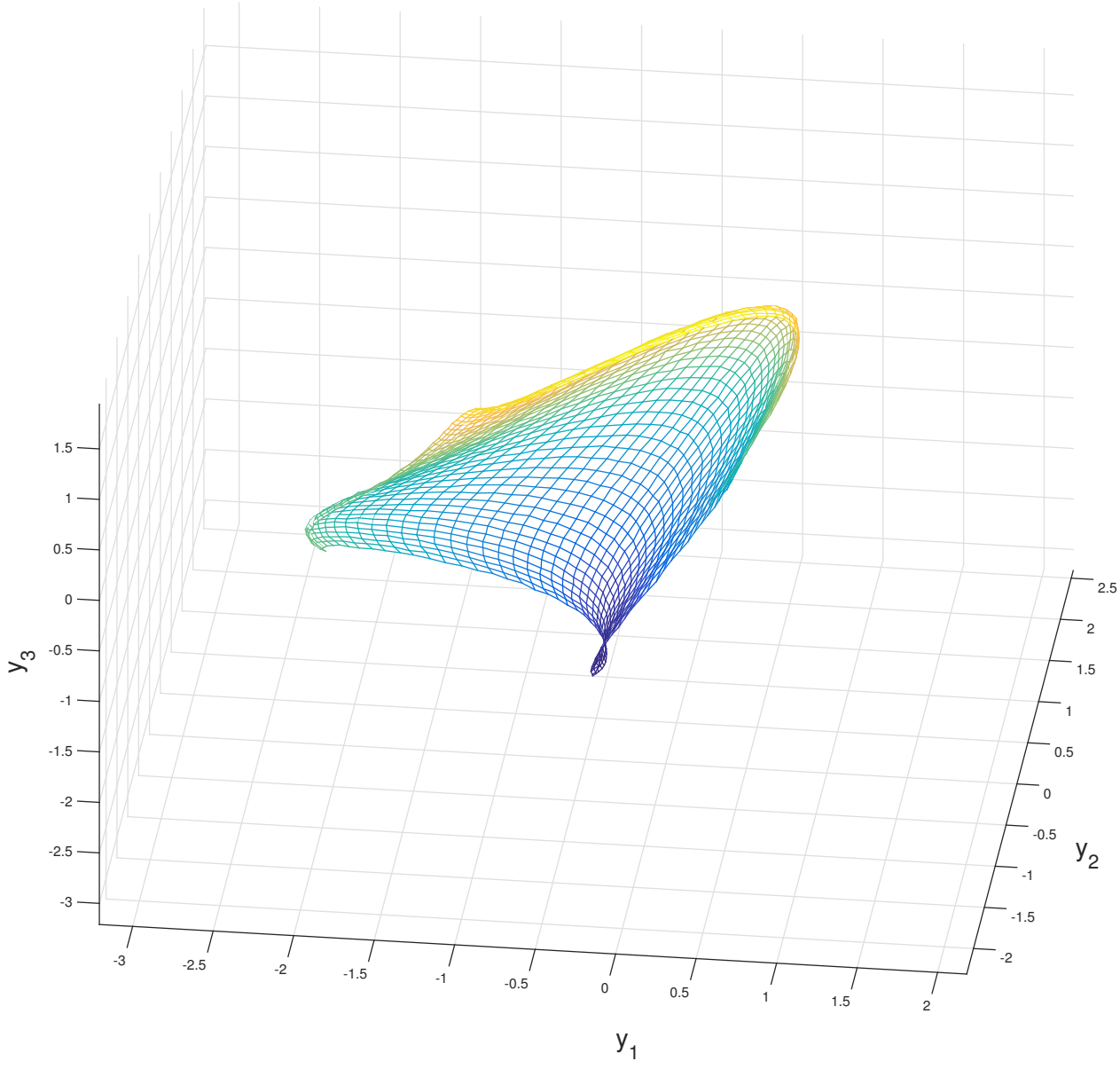
Gaussian Process Latent Variable Model: manifold samples



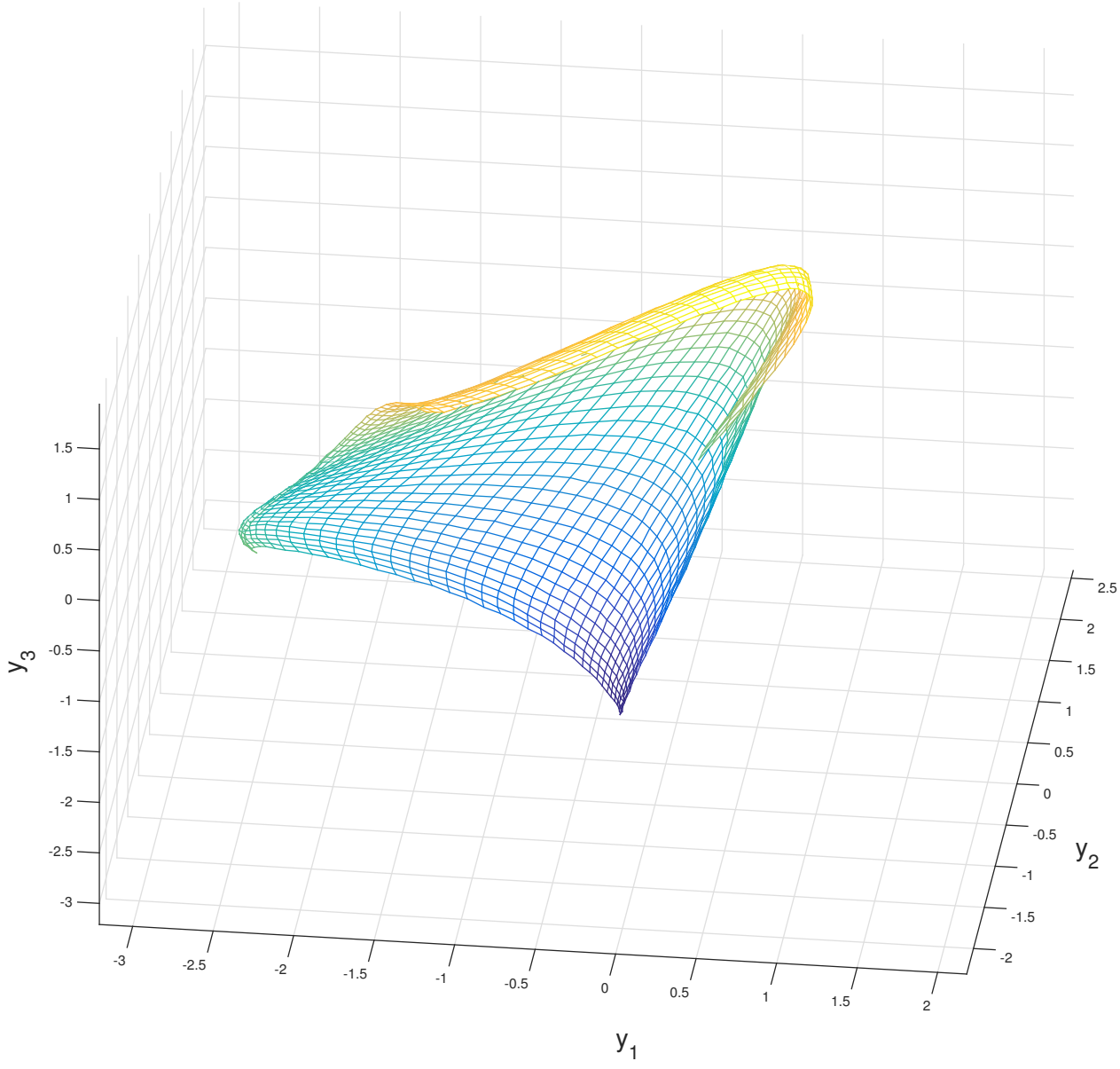
Gaussian Process Latent Variable Model: manifold samples



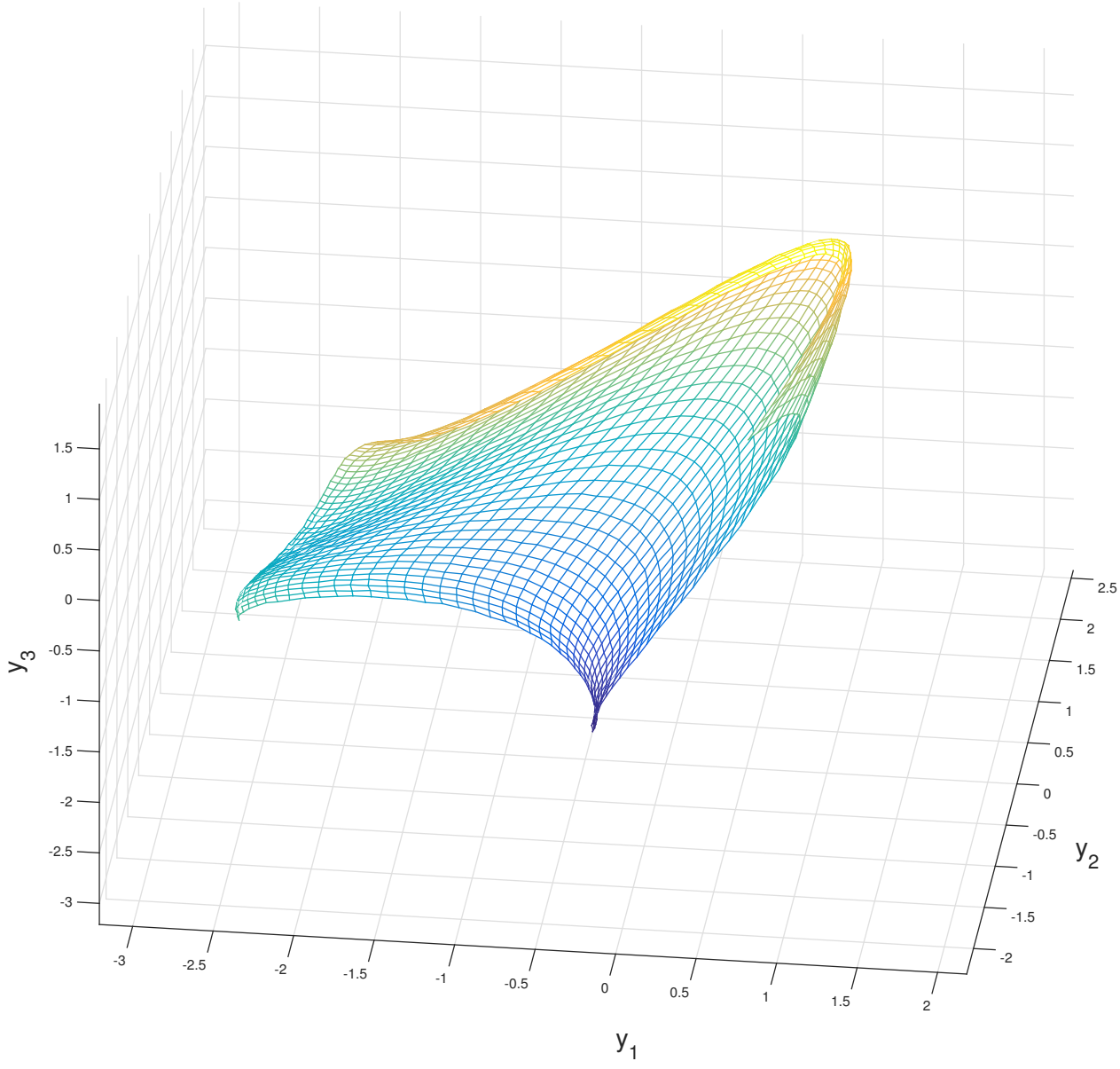
Gaussian Process Latent Variable Model: manifold samples



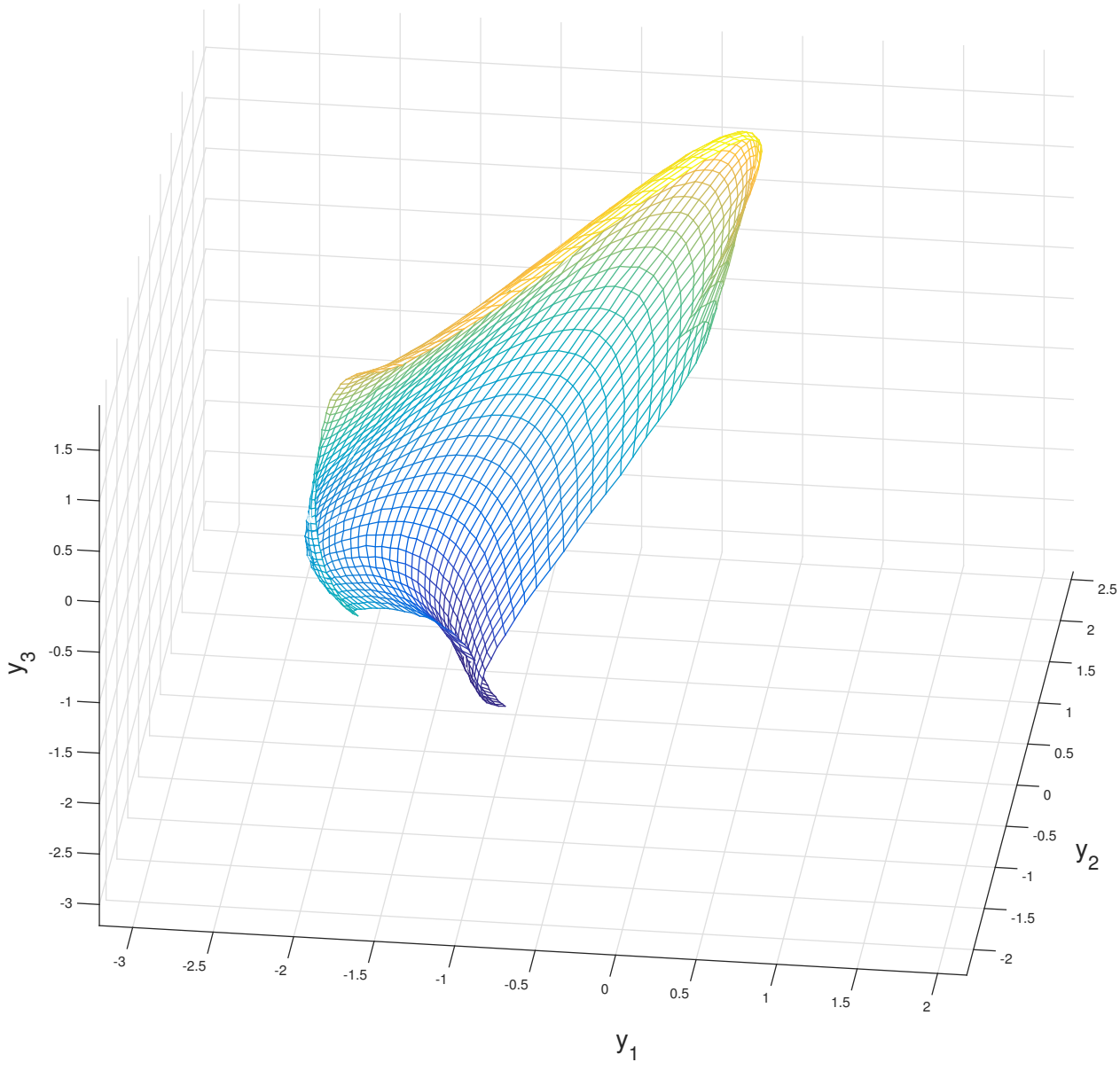
Gaussian Process Latent Variable Model: manifold samples



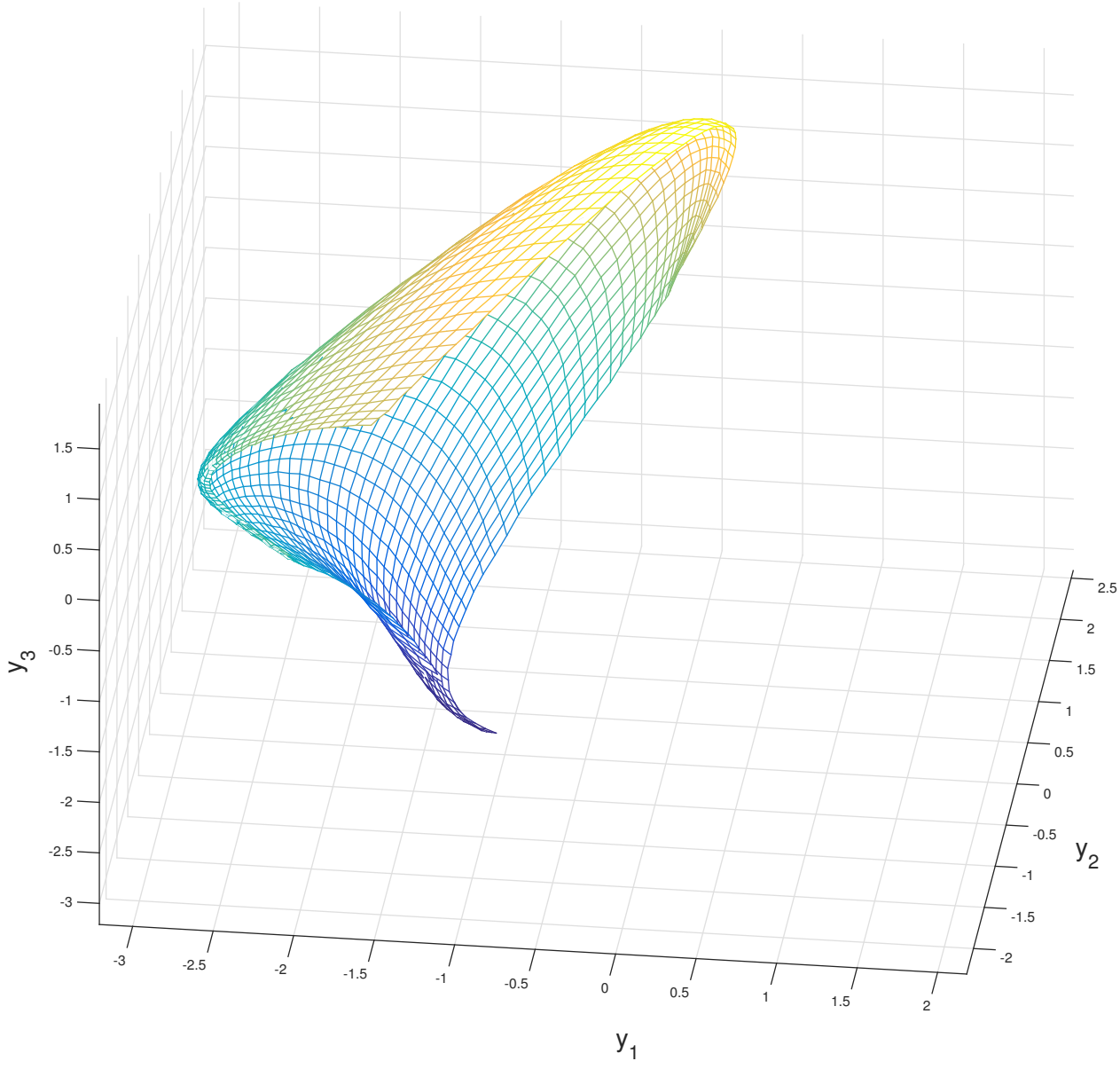
Gaussian Process Latent Variable Model: manifold samples



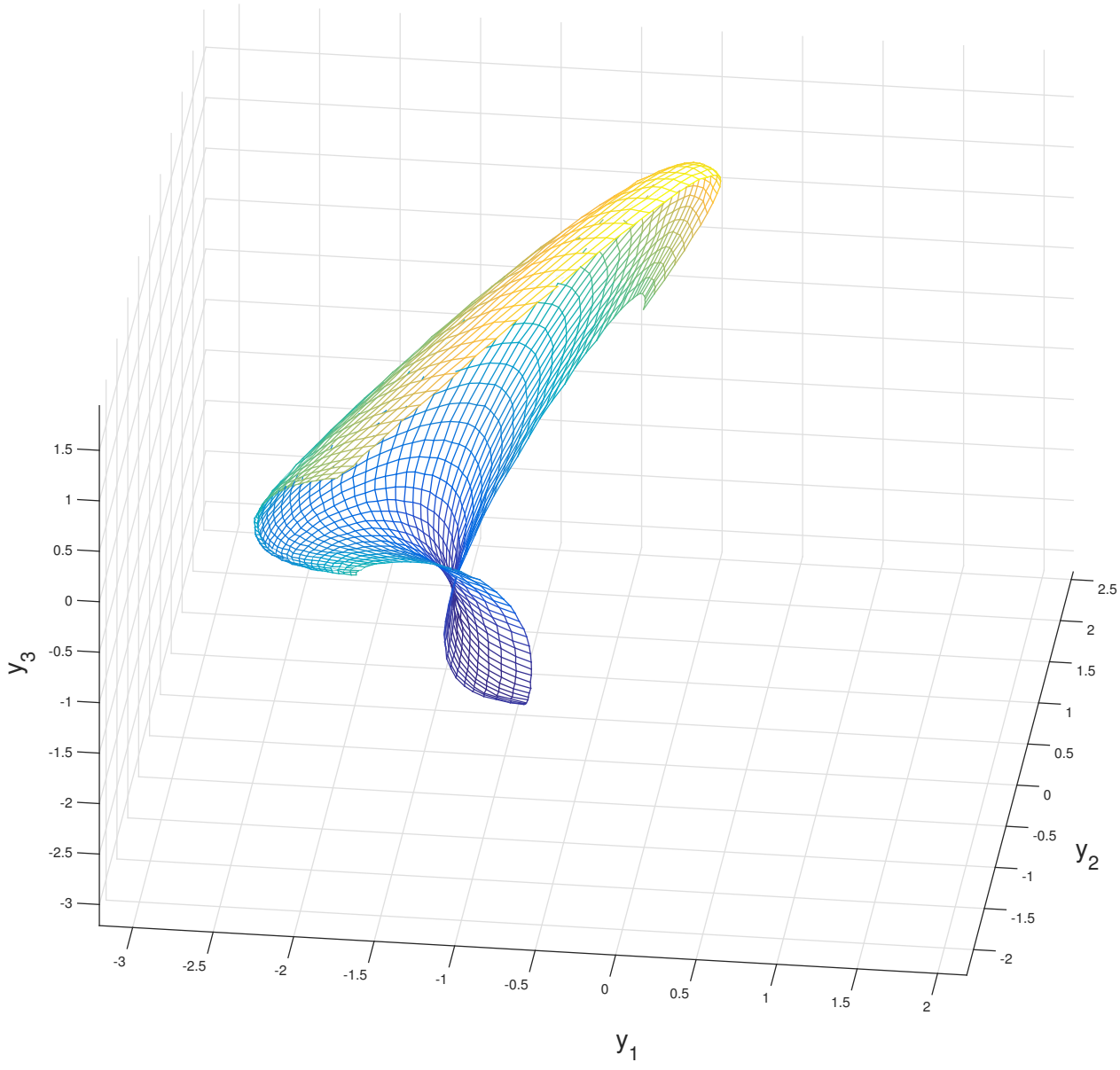
Gaussian Process Latent Variable Model: manifold samples



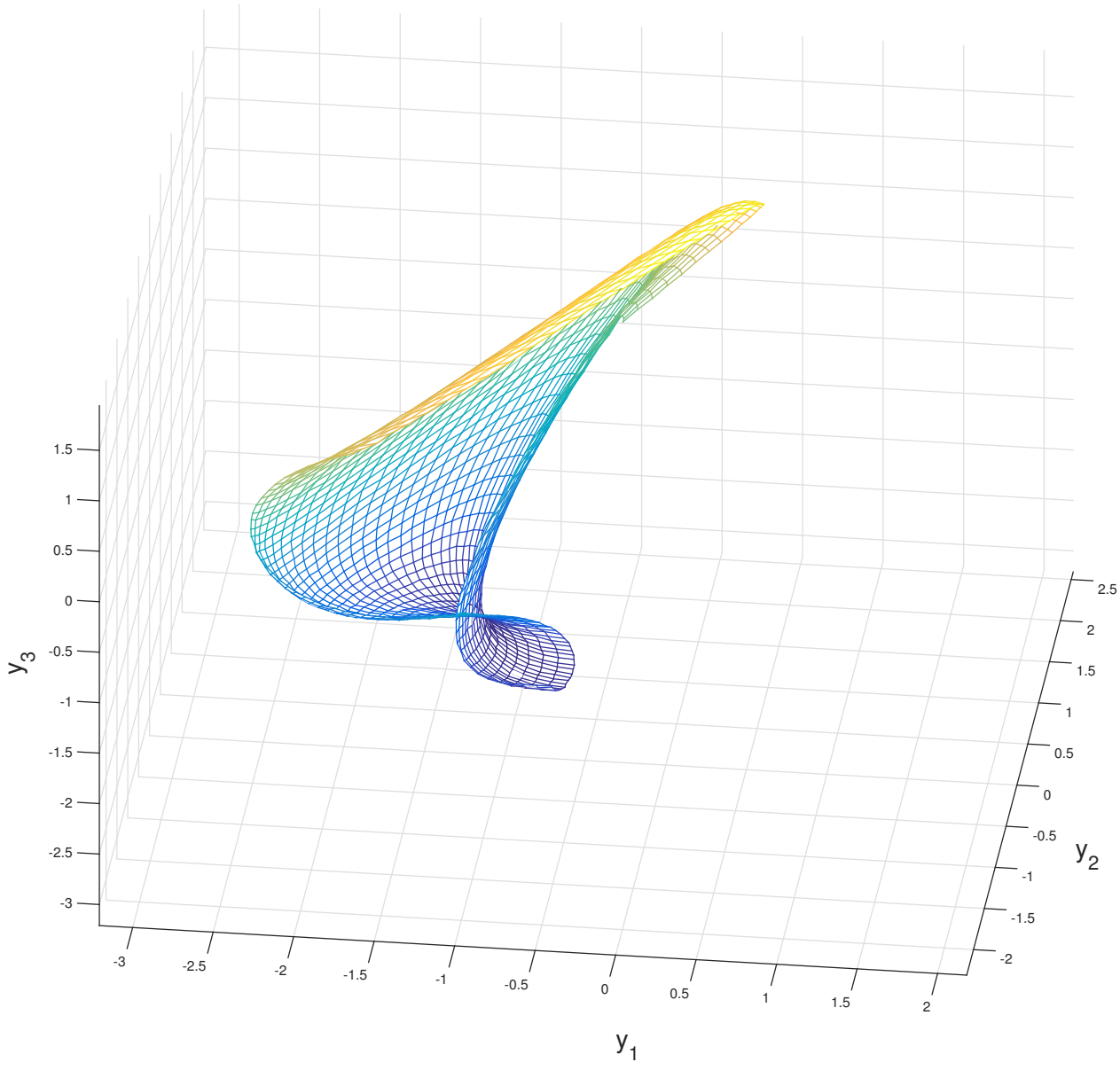
Gaussian Process Latent Variable Model: manifold samples



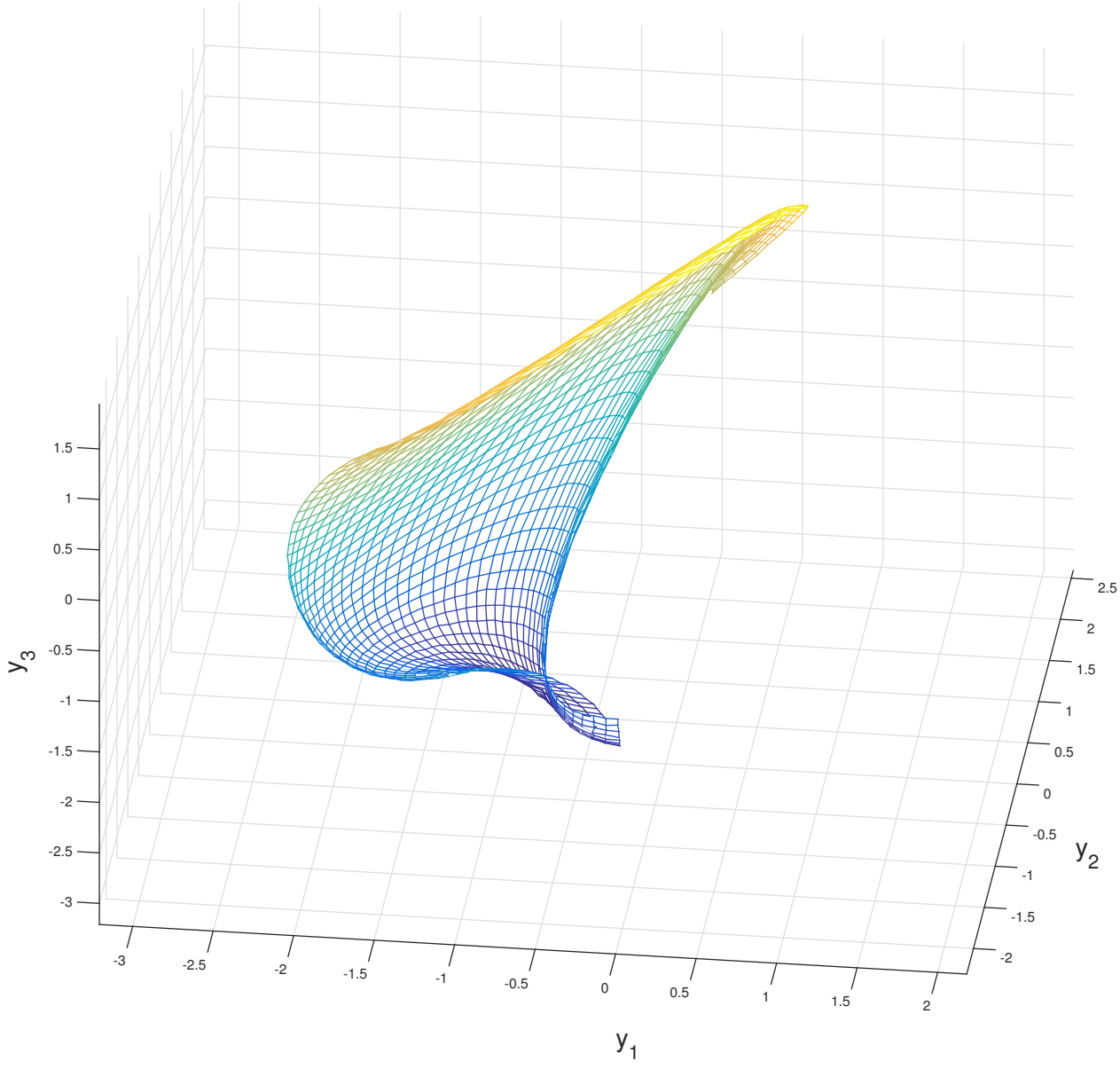
Gaussian Process Latent Variable Model: manifold samples



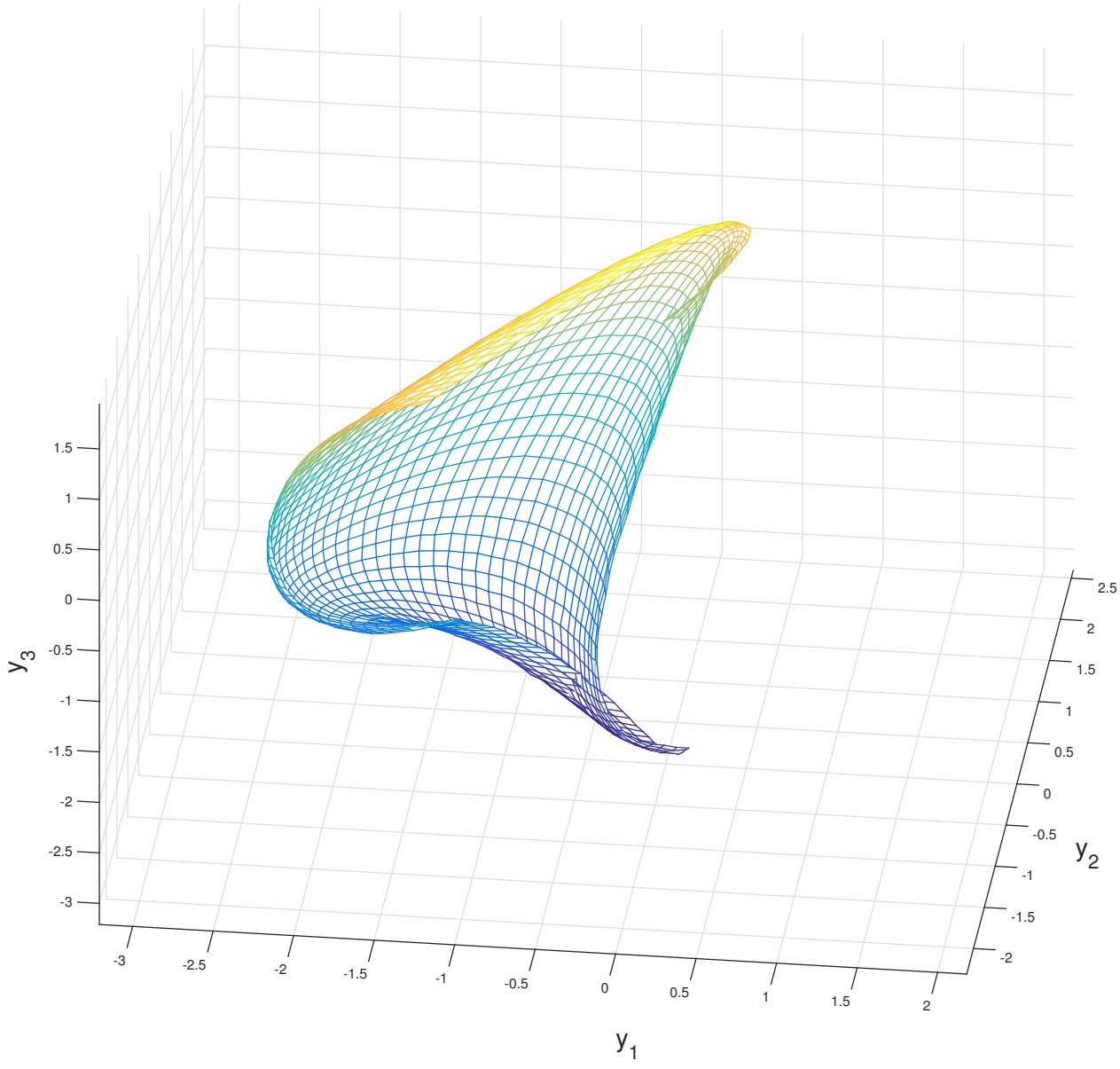
Gaussian Process Latent Variable Model: manifold samples



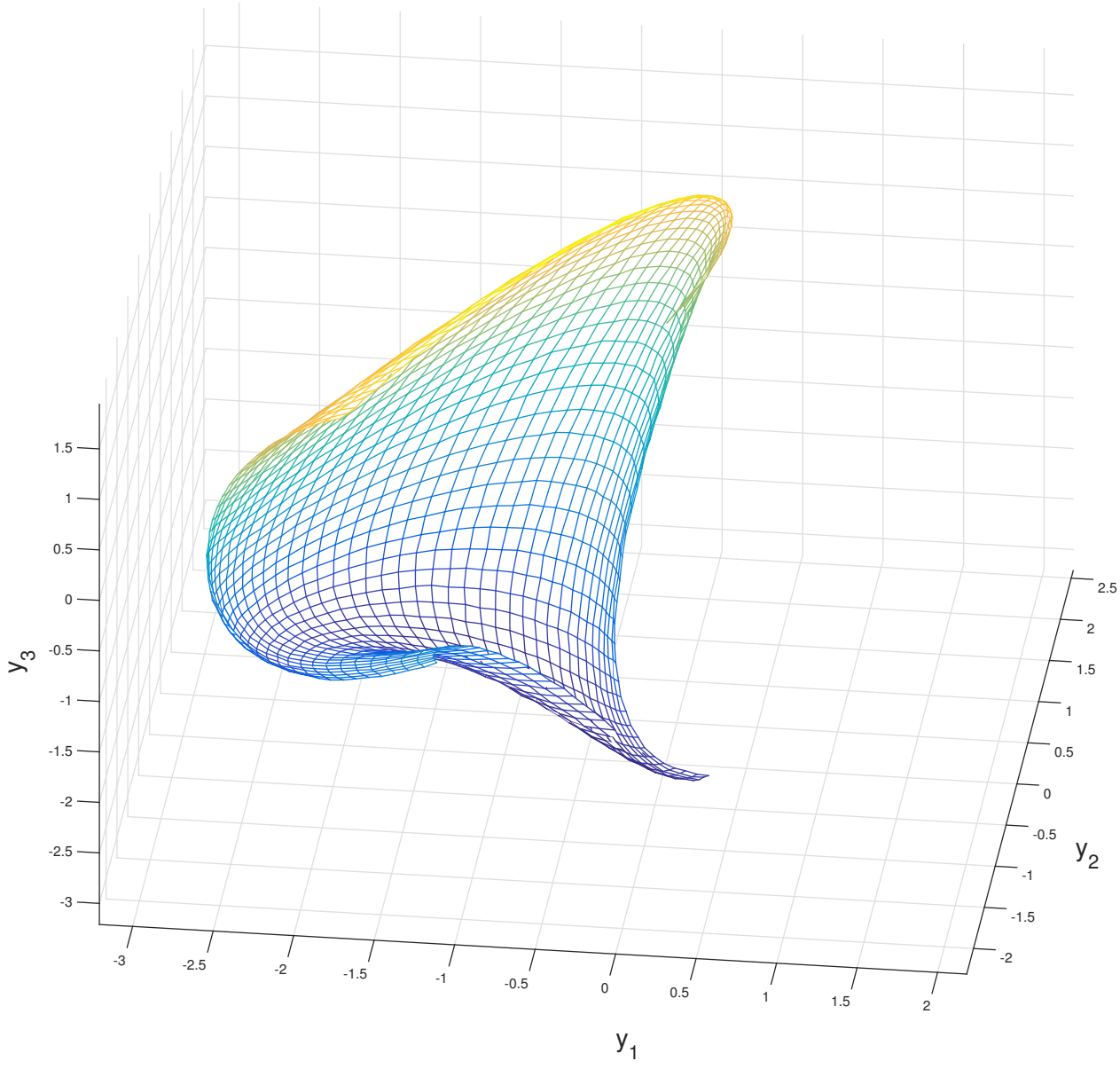
Gaussian Process Latent Variable Model: manifold samples



Gaussian Process Latent Variable Model: manifold samples

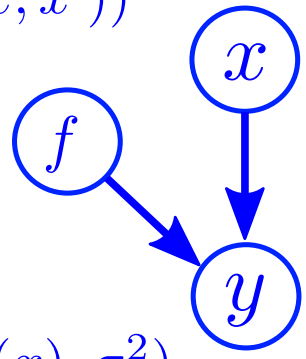


Gaussian Process Latent Variable Model: manifold samples



Gaussian Process Latent Variable Model

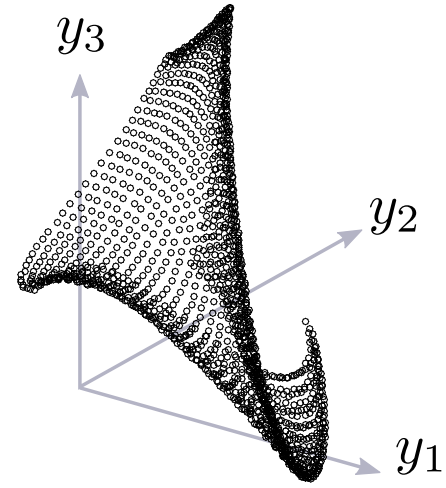
$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

selects position on manifold

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$



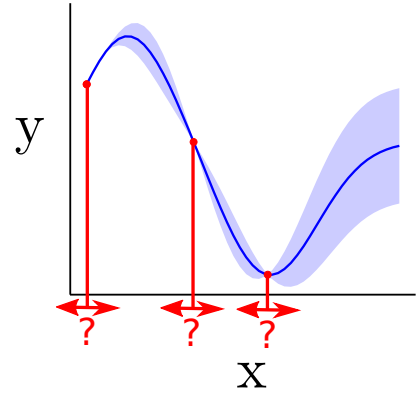
distribution over manifolds

recover PCA using linear covariance $C(x, x') = \sum_k x_k x'_k$

general case produces complex marginals

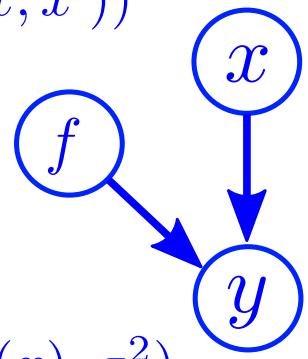
GP-regression with distribution over inputs (noisy/latent inputs)

Inference requires approximation...



Gaussian Process Latent Variable Model: MAP Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

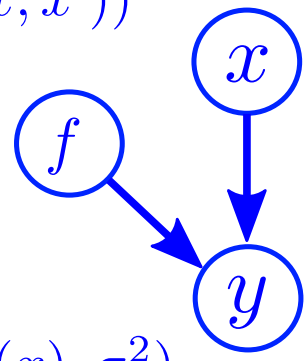
$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence, NIPS 2004

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y) = \arg\max_x \log p(x|y)$$

Gaussian Process Latent Variable Model: MAP Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

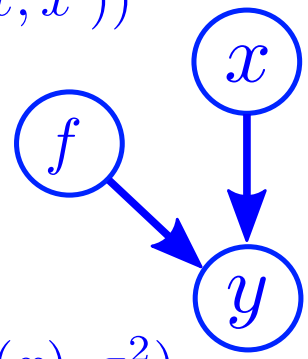
Lawrence, NIPS 2004

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y) = \arg\max_x \log p(x|y)$$

$$p(x|y) = \frac{1}{p(y)} p(y|x)p(x)$$

Gaussian Process Latent Variable Model: MAP Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence, NIPS 2004

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y) = \arg\max_x \log p(x|y)$$

$$p(x|y) = \frac{1}{p(y)} p(y|x)p(x)$$

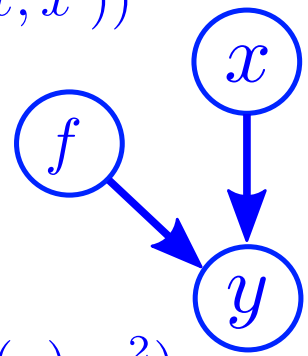
$$y_{1:N} = [y_{1:N,1}; \ldots; y_{1:N,D}]$$

$$p(y|x) = \int p(y|x, f)p(f)\mathrm{d}f = \mathcal{G}(y_{1:N}; 0, \Sigma(x_{1:N}))$$

multi-output regression

Gaussian Process Latent Variable Model: MAP Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence, NIPS 2004

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y) = \arg\max_x \log p(x|y)$$

$$p(x|y) = \frac{1}{p(y)} p(y|x)p(x)$$

$$y_{1:N} = [y_{1:N,1}; \ldots; y_{1:N,D}]$$

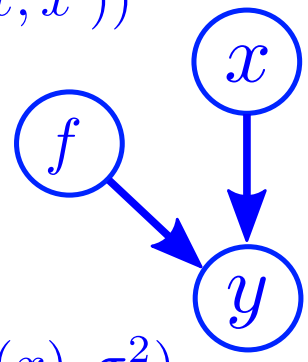
$$p(y|x) = \int p(y|x, f)p(f)\mathrm{d}f = \mathcal{G}(y_{1:N}; 0, \Sigma(x_{1:N}))$$

multi-output regression

$$x_{\mathrm{MAP}} = \arg\max_x \log p(x) - \frac{1}{2}\log\det\Sigma(x_{1:N}) - \frac{1}{2}\mathrm{trace}(\Sigma(x_{1:N})^{-1} y_{1:N} y_{1:N}^\top)$$

Gaussian Process Latent Variable Model: MAP Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence, NIPS 2004

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y) = \arg\max_x \log p(x|y)$$

$$p(x|y) = \frac{1}{p(y)} p(y|x)p(x)$$

$$y_{1:N} = [y_{1:N,1}; \ldots; y_{1:N,D}]$$

$$p(y|x) = \int p(y|x, f)p(f)\mathrm{d}f = \mathcal{G}(y_{1:N}; 0, \Sigma(x_{1:N}))$$

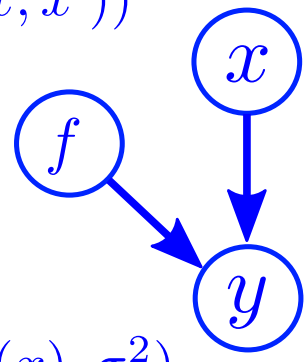
multi-output regression

$$x_{\mathrm{MAP}} = \arg\max_x \log p(x) - \frac{1}{2}\log\det\Sigma(x_{1:N}) - \frac{1}{2}\mathrm{trace}(\Sigma(x_{1:N})^{-1} y_{1:N} y_{1:N}^\top)$$

objective depends on: $||x_n||$ $||x_n - x_m||$ $||y_n - y_m||$ $||y_n||$

Gaussian Process Latent Variable Model: MAP Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence, NIPS 2004

MAP inference in GPLVM
-- optimise positions of data in latent space
-- objective depends on distances between points in new/old space
-- no explicit mappings

ISOMAP-like

But
-- implicit soft decoding function:

$$p(f|x_{\mathrm{MAP}}, y)$$

-- cost depends on mags.

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y) = \arg\max_x \log p(x|y)$$

$$p(x|y) = \frac{1}{p(y)} p(y|x)p(x)$$

$$y_{1:N} = [y_{1:N,1}; \ldots; y_{1:N,D}]$$

$$p(y|x) = \int p(y|x, f)p(f)\mathrm{d}f = \mathcal{G}(y_{1:N}; 0, \Sigma(x_{1:N}))$$

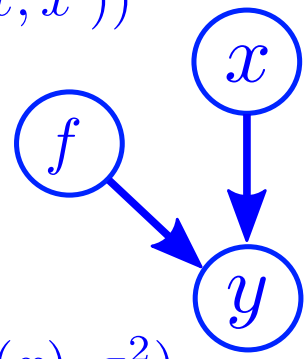
multi-output regression

$$x_{\mathrm{MAP}} = \arg\max_x \log p(x) - \frac{1}{2}\log\det\Sigma(x_{1:N}) - \frac{1}{2}\mathrm{trace}(\Sigma(x_{1:N})^{-1} y_{1:N} y_{1:N}^\top)$$

objective depends on: $||x_n||$ $||x_n - x_m||$ $||y_n - y_m||$ $||y_n||$

Gaussian Process Latent Variable Model: Back-constrained Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

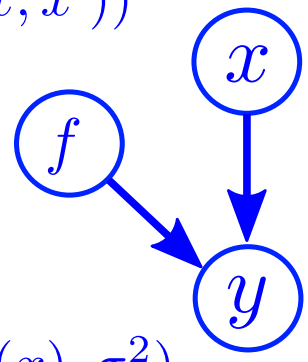
$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence et al, ICML 2006

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y, \theta) = \arg\max_x \log p(x, y|\theta)$$

Gaussian Process Latent Variable Model: Back-constrained Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence et al, ICML 2006

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y, \theta) = \arg\max_x \log p(x, y|\theta)$$

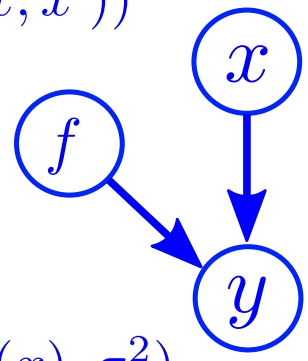
after optimisation:

$$x_{\mathrm{MAP}} = x_{\mathrm{MAP}}(y_{1:N})$$ approximate: $$x_{\mathrm{MAP}} \approx g_\phi(y_{1:N})$$

recognition model
or back-constraint

Gaussian Process Latent Variable Model: Back-constrained Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence et al, ICML 2006

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y, \theta) = \arg\max_x \log p(x, y|\theta)$$

after optimisation:

$$x_{\mathrm{MAP}} = x_{\mathrm{MAP}}(y_{1:N}) \quad \text{approximate:} \quad x_{\mathrm{MAP}} \approx g_\phi(y_{1:N})$$

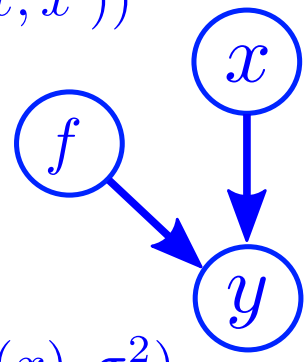
recognition model
or back-constraint

learn using same objective as before:

$$\phi = \arg\max_\phi p(x = g_\phi(y_{1:N})|y, \theta) = \arg\max_\phi \log p(x = g_\phi(y_{1:N}), y|\theta)$$

Gaussian Process Latent Variable Model: Back-constrained Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence et al, ICML 2006

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y, \theta) = \arg\max_x \log p(x, y|\theta)$$

after optimisation:

$$x_{\mathrm{MAP}} = x_{\mathrm{MAP}}(y_{1:N}) \quad \text{approximate:} \quad x_{\mathrm{MAP}} \approx g_\phi(y_{1:N})$$

recognition model
or back-constraint

learn using same objective as before:

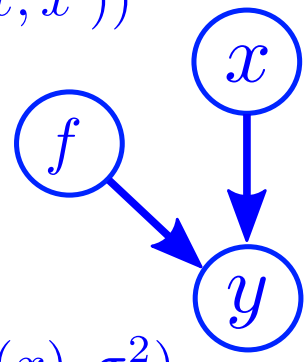
$$\phi = \arg\max_\phi p(x = g_\phi(y_{1:N})|y, \theta) = \arg\max_\phi \log p(x = g_\phi(y_{1:N}), y|\theta)$$

learn hyper-parameters using zero-temperature EM

$$\theta = \arg\max_\theta \log p(y|\theta) \approx \arg\max_\theta \log p(x = g_\phi(y_{1:N}), y|\theta)$$

Gaussian Process Latent Variable Model: Back-constrained Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence et al, ICML 2006

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y, \theta) = \arg\max_x \log p(x, y|\theta)$$

after optimisation:

$$x_{\mathrm{MAP}} = x_{\mathrm{MAP}}(y_{1:N}) \quad \text{approximate:} \quad x_{\mathrm{MAP}} \approx g_\phi(y_{1:N})$$

recognition model or back-constraint

learn using same objective as before:

$$\phi = \arg\max_\phi p(x = g_\phi(y_{1:N})|y, \theta) = \arg\max_\phi \log p(x = g_\phi(y_{1:N}), y|\theta)$$

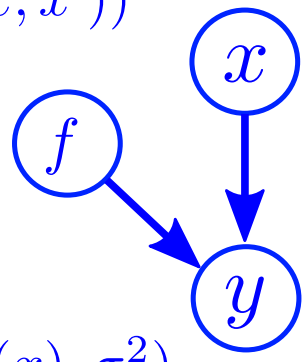
learn hyper-parameters using zero-temperature EM

$$\theta = \arg\max_\theta \log p(y|\theta) \approx \arg\max_\theta \log p(x = g_\phi(y_{1:N}), y|\theta)$$

optimise same function

Gaussian Process Latent Variable Model: Back-constrained Inference

$$p(f_d) = \mathcal{GP}(f; 0, C(x, x'))$$



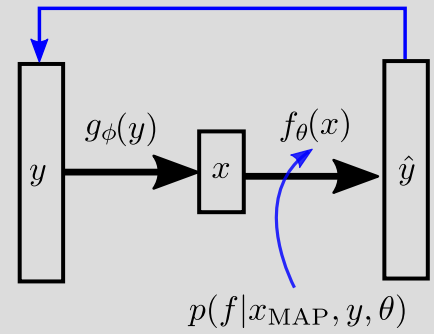
$$p(x) = \mathcal{G}(x; 0, \mathrm{I})$$

$$p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2)$$

Lawrence et al, ICML 2006

Back-constrained GPLVM
Accelerate inference/ learning via recognition model

$$\arg\max_{\theta,\phi} \log p(x = g_\phi(y_{1:N}), y|\theta)$$



auto-encoder-like

$$x_{\mathrm{MAP}} = \arg\max_x p(x|y, \theta) = \arg\max_x \log p(x, y|\theta)$$

after optimisation:

$$x_{\mathrm{MAP}} = x_{\mathrm{MAP}}(y_{1:N}) \quad \text{approximate:} \quad x_{\mathrm{MAP}} \approx g_\phi(y_{1:N})$$

recognition model or back-constraint

learn using same objective as before:

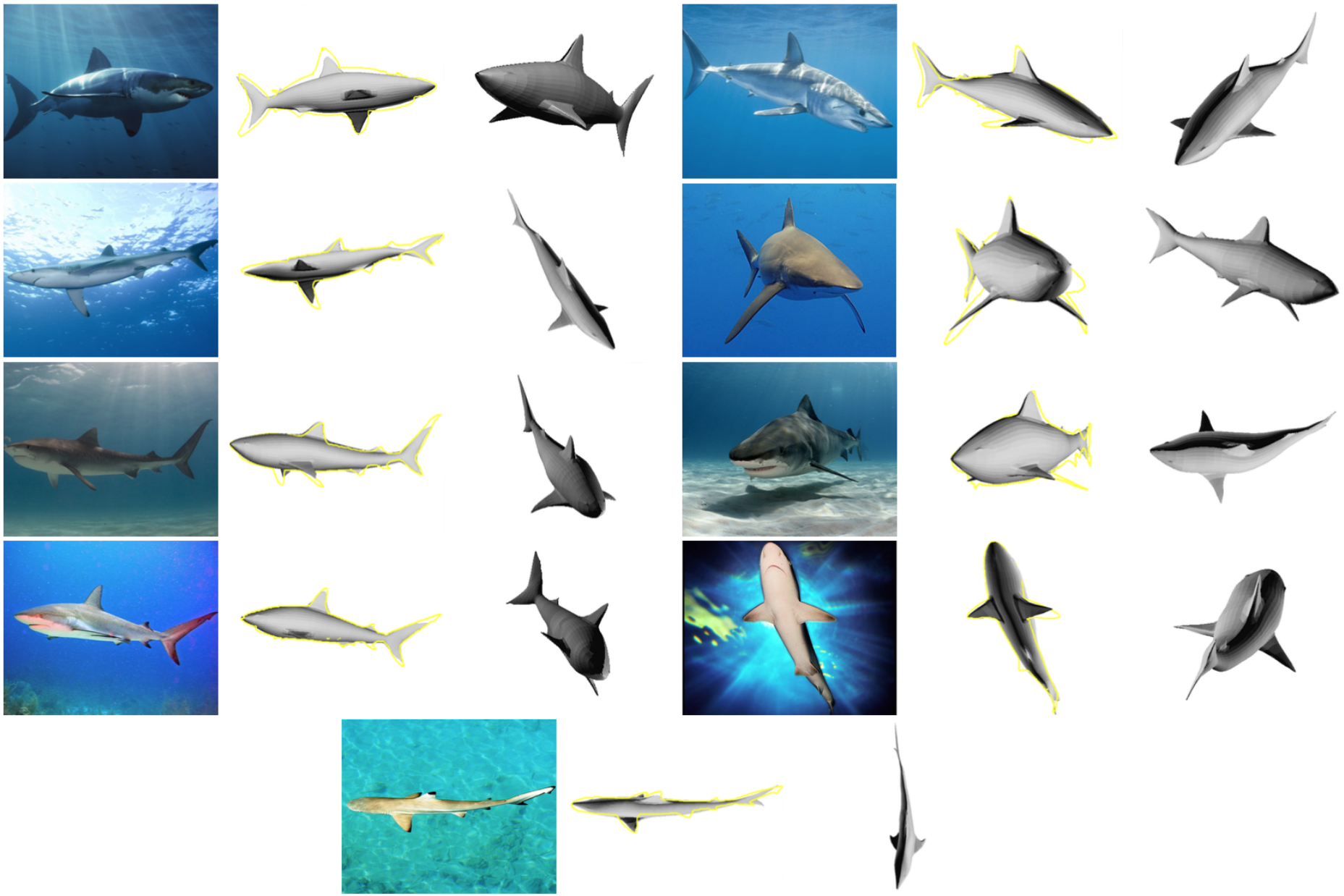
$$\phi = \arg\max_\phi p(x = g_\phi(y_{1:N})|y, \theta) = \arg\max_\phi \log p(x = g_\phi(y_{1:N}), y|\theta)$$

learn hyper-parameters using zero-temperature EM

$$\theta = \arg\max_\theta \log p(y|\theta) \approx \arg\max_\theta \log p(x = g_\phi(y_{1:N}), y|\theta)$$

optimise same function

Gaussian Process Latent Variable Model: Application



Chen, Kim and Cipolla, IEEE 13th Int. Conf. on Computer Vision, 2011 and https://www.youtube.com/watch?v=X5Z7ZJ39zAA