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

simple building blocks
- preprocessing/feature learning: reducing computational complexity improving statistical efficiency
- compose into complex models:
  - FA -> LGSSMs
  - GPLVM -> GPSSMs

Cunningham and Yu, Nature Neuro, 2014

Hinton and Salakhutdinov, Science, 2006
Dimensionality reduction: conceptual space

- Probabilistic modelling
- FA
- pPCA
- PCA
- Min reconstruction error
- Max variance
- Linear Gaussian
- Linear
- Max mutual information
- Embedding methods
- 2nd order statistics
Dimensionality reduction: conceptual space

- **PCA**
- **probabilistic modelling**
- **FA**
- **pPCA**
- **max mutual information**
- **embedding methods**
- **MDS**
- **ISOMAP**
- **min reconstruction error**
- **auto-encoders**
- **deep NN auto-encoder**
- **denoising auto-encoder**
- **noise-corrupted input**
- **map inference**
- **GP-LVM**
- **non-linear component models**
- **linear Gaussian**
- **2nd order statistics**
- **linear**
- **max variance**
- **linear**
- **RKHS feature expansion**
- **kernel-PCA**
- **multi-view GP-LVM**
- **IB-FA**
- **CCA**
- **non-linear non-parametric**
- **IB-FA CCA**
- **multi-view**
- **linear**
- **non-parametric**
- **non-linear**
- **non-linear**
- **non-parametric**
- **non-parametric**
- **kernel-PCA**
- **RKHS feature expansion**
- **non-Gaussian**
- **non-linear**
- **distances via local reconstruction weights**
- **LLE**
- **higher-order statistics**
- **distances via data-graph**
- **ISOMAP**
- **Laplacian eigenmaps**
- **spectral methods**
- **ICA**
- **dynamical GP-LVM**
- **LGSSM**
- **SFA**
- **time-series**
- **non-Gaussian**
- **MAP inference**
- **multi-view**
- **GP-LVM**
- **non-parametric**
- **non-linear**
- **non-parametric**
- **dynamical GP-LVM**
- **recognition models**
- **distances via local reconstruction weights**
- **LLE**
- **kernel-PCA**
- **RKHS feature expansion**
- **non-Gaussian**
- **non-linear**
- **distances via data-graph**
- **ISOMAP**
- **Laplacian eigenmaps**
- **spectral methods**

Further details on non-parametric, non-linear, non-Gaussian, and multi-view models are provided in the diagram.
Dimensionality reduction via distance preserving embeddings

original dataset
high-dimensional

\[ d_{n,m}^{(y)} = ||y^{(n)} - y^{(m)}|| \]
Dimensionality reduction via distance preserving embeddings

Original dataset
High-dimensional

New dataset
Low-dimensional

\[ \text{dim}(y) = D \quad \Rightarrow \quad \text{dim}(x) = K \]

\[ d_{nm}^{(y)} = \| y^{(n)} - y^{(m)} \| \quad \approx \quad d_{nm}^{(x)} = \| x^{(n)} - x^{(m)} \| \]
Dimensionality reduction via distance preserving embeddings

\[ \dim(y) = D > \dim(x) = K \]

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

embed by optimising new datapoints to match distances:

\[
\arg \min_{\{x(n)\}_{n=1}^{N}} \sum_{n < m} \| d_{nm}^{(y)} - d_{nm}^{(x)} \|
\]
Dimensionality reduction via distance preserving embeddings

original dataset
high-dimensional

new dataset
low-dimensional

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

embed by optimising new datapoints to match distances:

\[
\arg \min_{\{x^{(n)}\}_{n=1}^{N}} \sum_{n<m} \|d^{(y)}_{nm} - d^{(x)}_{nm}\|
\]
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
Question:
Will MDS with Euclidean distances make sensible 1D embeddings of:

\[ y_1, y_2, y_3 \]

\[ x_1, x_2 \]

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

embed by optimising new datapoints to match distances:

\[ \arg \min \sum_{n<m} \| d^{(y)}_{nm} - d^{(x)}_{nm} \| \]

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:
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

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

**Desire:**

embed by optimising new datapoints to match distances:

\[
\text{arg min} \sum_{n<m} ||d_{nm}^{(y)} - d_{nm}^{(x)}||
\]

PCA: Euclidean metric (squared error)
MDS: general distance metric e.g. ISOMAP

Tenenbaum et al.
*Science*, 2000
Dimensionality reduction via distance preserving embeddings

**ISOMAP:**
Geodesic distance via neighbourhood graph

Now target distances reflect desires

**PCA:** Euclidean metric (squared error)

**MDS:** general distance metric e.g. ISOMAP

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

Embed by optimising new datapoints to match distances:

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

Tenenbaum et al Science, 2000
Dimensionality reduction via distance preserving embeddings

**original dataset**

- High-dimensional

**new dataset**

- Low-dimensional

**ISOMAP:**

Geodesic distance via
neighbourhood graph

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

embed by optimising new datapoints to match distances:

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

**PCA:** Euclidean metric (squared error)

**MDS:** general distance metric e.g. ISOMAP

Tenenbaum et al

Science, 2000
Dimensionality reduction via distance preserving embeddings

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

**Original dataset**

**Geodesic distance via neighbourhood graph**

**New dataset**

Bottom loop articulation
Top arch articulation

---

ISOMAP, Tenenbaum et al Science 2000
LLE, Roweis et al. Science 2000
tSNE, Hinton and van der Maaten JMLR 2008
Dimensionality reduction via distance preserving embeddings

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

\[ y \xrightarrow{g_\phi(y)} x \xrightarrow{f_\theta(x)} \hat{y} \]

- **Data**
- **Low-dim code**
- **Reconstruction**

\[ x(y) = g_\phi(y) \quad \hat{y}(x) = f_\theta(x) \]
Dimensionality reduction using auto-encoders

Want to learn interesting embedding (not identity mapping)

Achieve via constraint: dimensionality

\[
\arg\min_{\theta, \phi} \sum_{n=1}^{N} ||y^{(n)} - \hat{y}^{(n)}||^2
\]
Dimensionality reduction using auto-encoders

Want to learn interesting embedding (not identity mapping)

Achieve via constraint: dimensionality, sparsity, function complexity

\[ ||y - \hat{y}||^2 \]

\[ x(y) = g_{\phi}(y) \]
\[ \hat{y}(x) = f_{\theta}(x) \]

\[ \arg \min_{\theta, \phi} \sum_{n=1}^{N} ||y^{(n)} - \hat{y}^{(n)}||^2 + \text{constraints} \]
Dimensionality reduction using auto-encoders

Want to learn interesting embedding (not identity mapping)
Achieve via constraint: dimensionality, sparsity, function complexity

\[
\begin{align*}
\text{data} & \quad \text{low-dim code} & \quad \text{reconstruction} & \quad \text{cost function} \\
\mathbf{y} & \quad \mathbf{x} & \quad \hat{\mathbf{y}} & \quad \arg \min_{\theta, \phi} \sum_{n=1}^{N} \| \mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)} \|^2 + \text{constraints} \\
g_\phi(y) & \quad f_\theta(x) & \quad \hat{\mathbf{y}} & \quad \arg \min_{\Theta, \Phi} \sum_{n=1}^{N} \| \mathbf{y}^{(n)} - \Theta \Phi \mathbf{y}^{(n)} \|^2
\end{align*}
\]

PCA:
- Linear
  \[
  \begin{align*}
  \mathbf{x} &= \Phi \mathbf{y} \\
  \dim(\Phi) &= K \times D \\
  \mathbf{\hat{y}} &= \Theta \mathbf{x} \\
  \dim(\Theta) &= D \times K
  \end{align*}
  \]
Dimensionality reduction using auto-encoders

\[ \| y - \hat{y} \|^2 \]

\( y \rightarrow g_{\phi}(y) \rightarrow x \rightarrow f_{\theta}(x) \rightarrow \hat{y} \)

Want to learn interesting embedding (not identity mapping)

Achieve via constraint: dimensionality, sparsity, function complexity

**Cost function**

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

**PCA:**

- linear
  - \( x = \Phi y \)
  - \( \text{dim}(\Phi) = K \times D \)
  - \( \hat{y} = \Theta x \)
  - \( \text{dim}(\Theta) = D \times K \)

**Deep neural AA:**

- deep neural network
  - \( x = g_{\phi}(y) \)
  - \( \text{dim}(g_{\phi}(y)) = \Phi y \)
  - \( \text{dim}(\Phi) = K \times D \)
  - \( \hat{y} = f_{\theta}(x) \)
  - \( \text{dim}(f_{\theta}(x)) = \Theta x \)
  - \( \text{dim}(\Theta) = D \times K \)

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

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

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

PCA: linear
\[
x = \Phi y
\]
\[
dim(\Phi) = K \times D
\]

deep neural AA: deep neural network

Hinton et al Science, 2006
Dimensionality reduction using auto-encoders

$$\|y - \hat{y}\|^2$$

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

Want to learn interesting embedding (not identity mapping)

Achieve via constraint: dimensionality, sparsity, function complexity

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

**PCA:**

- Linear
- $$x = \Phi y$$
- $$\dim(\Phi) = K \times D$$

**deep neural AA:**

- Linear
- $$\hat{y} = \Theta x$$
- $$\dim(\Theta) = D \times K$$

**auto-encoders:**

1. learn functions for both mappings
2. num. params. does not scale with N

Vincent, Bengio et al JMLR, 2010
Dimensionality reduction using probabilistic models: PCA

\[ \text{dim}(x) = K \]
\[ p(x) = \mathcal{G}(x; 0, I) \]

\[ \text{dim}(y) = D \]
\[ y = \theta x + \sigma \epsilon \]
\[ p(\epsilon) = \mathcal{G}(\epsilon; 0, I) \]

fuzzy pancake
Dimensionality reduction using probabilistic models: PCA

\[ \text{dim}(x) = K \]
\[ p(x) = \mathcal{G}(x; 0, I) \]

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

\[ \text{dim}(y) = D \]
\[ y = \theta x + \sigma \epsilon \]
\[ p(\epsilon) = \mathcal{G}(\epsilon; 0, I) \]
Dimensionality reduction using probabilistic models: PCA

\[ \text{dim}(x) = K \]
\[ p(x) = \mathcal{G}(x; 0, I) \]

**inference**
\[ p(x^{(n)}|y^{(n)}; \phi, \sigma) = \mathcal{G}(x^{(n)}; \phi y^{(n)}, \Sigma_{x|y}) \]

**maximum-likelihood learning**
\[ \theta^{ML}, \sigma^{ML} = \arg \max_{\phi, \sigma} \sum_n \log p(y^{(n)}|\theta, \sigma) \]
\[ p(y^{(n)}|\theta, \sigma) = \mathcal{G}(y^{(n)}; 0, \theta \theta^\top + \sigma^2) \]

\[ \text{dim}(y) = D \]
\[ y = \theta x + \sigma \epsilon \]
\[ p(\epsilon) = \mathcal{G}(\epsilon; 0, I) \]

fuzzy pancake
Dimensionality reduction using probabilistic models: PCA

$$\dim(x) = K$$

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

Inference

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

Maximum-likelihood learning

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

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

differentiate, set to zero, find:

$$\sigma^{ML} = \frac{1}{D - K} \sum_{k=K+1}^D \lambda_k \quad E_K = [e_1, \ldots, e_K] \quad \Lambda_K = \text{diag}(\lambda_1, \ldots, \lambda_K)$$

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

Data-covariance

$$\hat{\Sigma}_y e_k = \lambda_k e_k$$

$$\hat{\Sigma}_y = \frac{1}{N} \sum_n y^{(n)} (y^{(n)})^\top$$

Eigenvectors, eigenvalues

Tipping and Bishop 1999, Roweis 1997
Dimensionality reduction using probabilistic models: a family of models

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

\[ D = \sigma^2 I \]

train: eigenvalue problem
Dimensionality reduction using probabilistic models: a family of models

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

train: eigenvalue problem
Dimensionality reduction using probabilistic models: a family of models

**MODEL**
- full linear factor analysis (FA)
- inter-battery FA

**CLASS**
- audio
- video

**Factor Analysis (FA)**
- Prior: $p(x) = \mathcal{G}(x; 0, I)$
- Conditional: $p(y|x) = \mathcal{G}(y; \theta x, D)$

**Inter-battery FA**
- Prior: $p(x), p(x_i) = \mathcal{G}(x; 0, I)$
- Conditional: $p(y_i|x) = \mathcal{G}(y_i; \theta_i^{sh} x + \theta_i^{pri} x_i, D)$

**Special Linear**
- PCA
- Canonical Correlation Analysis

**Parameters**
- $D = \sigma^2 I$
- Train: eigenvalue problem
- Train: generalised eigenvalue problem
Dimensionality reduction using probabilistic models: a family of models

- **Full linear factor analysis (FA)**
  - $p(x) = \mathcal{G}(x; 0, I)$
  - $p(y|x) = \mathcal{G}(y; \theta x, D)$

- **Special linear factor analysis (FA)**
  - $p(x_i; x) = \mathcal{G}(x_i; 0, I)$
  - $p(y_i|x) = \mathcal{G}(y_i; \theta_i^{sh} x + \theta_i^{pri} x_i, D)$

- **PCA**
  - $D = \sigma^2 I$
  - **Train**: eigenvalue problem

- **Inter-battery FA**
  - $p(x), p(x_i) = \mathcal{G}(x; 0, I)$
  - **Train**: generalised eigenvalue problem

- **Canonical correlation analysis**
  - $D = \sigma^2 I$
  - **Train**: find linear projections of $y_i$ that max correlation
Dimensionality reduction using probabilistic models: a family of models

**MODEL**
- full linear
- factor analysis (FA)
- inter-battery FA
- PCA
- canonical correlation analysis
- LGSSM

**CLASS**
- audio
- video

**formulas**
- full linear factor analysis (FA): $p(x) = \mathcal{G}(x; 0, I)$
- diagonal factor analysis (FA): $p(x_i) = \mathcal{G}(x_i; 0, I)$
- inter-battery FA: $p(y|x) = \mathcal{G}(y; \theta x, D)$
- canonical correlation analysis: $p(y|x) = \mathcal{G}(y; \theta x, D)$
- LGSSM: $p(x_t|x_{t-1}) = \mathcal{G}(x_t; \Psi x_{t-1}, \Sigma)$

**Linear (special) models**
- PCA: $D = \sigma^2 I$
- find linear projections of $y_i$ that max correlation

**Diagonal models**
- train: eigenvalue problem
- train: generalised eigenvalue problem
Dimensionality reduction using probabilistic models: a family of models

\begin{align*}
  & \text{full linear factor analysis (FA)} \\
  & p(x) = \mathcal{G}(x; 0, I) \\
  & p(y|x) = \mathcal{G}(y; \theta x, D) \\
  \text{diagonal} \\
  & \text{inter-battery FA} \\
  & p(x), p(x_i) = \mathcal{G}(x; 0, I) \\
  & p(y_i|x) = \mathcal{G}(y_i; \theta_i^{sh} x + \theta_i^{pri} x_i, D) \\
  \text{special linear} \\
  & \text{PCA} \\
  & D = \sigma^2 I \\
  & \text{train: eigenvalue problem} \\
  \text{special linear} \\
  & \text{canonical correlation analysis} \\
  & D = \sigma^2 I \\
  & \text{train: generalised eigenvalue problem} \\
  \text{special linear} \\
  & \text{slow feature analysis} \\
  & D = \sigma^2 I \\
  & \Sigma = 1 - \Psi^2 \\
  & \Psi = \text{diag}(\psi_1, \ldots, \psi_K) \\
  & \text{train: generalised eigenvalue problem} \\
  \end{align*}
Dimensionality reduction using probabilistic models: a family of models

MODEL

CLASS

full linear

factor analysis (FA)

\[ p(x) = \mathcal{G}(x; 0, I) \]

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

factor analysis (FA)

\[ p(x), p(x_i) = \mathcal{G}(x; 0, I) \]

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

inter-battery FA

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

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

LGSSM

slow feature analysis

\[ D = \sigma^2 I \]

\[ D = \sigma^2 I \]

\[ \Sigma = 1 - \Psi^2 \]

\[ \Psi = \text{diag}(\psi_1, \ldots, \psi_K) \]

special linear

PCA

\[ D = \sigma^2 I \]

train: eigenvalue problem

train: generalised eigenvalue problem

train: generalised eigenvalue problem

canonical correlation analysis

find linear projections of \( y_i \) that max correlation

find slowest projections of unit variance

special case

full linear
diagonal

special
linear

Factor analysis (FA)

Inter-battery FA

Latent Gaussian state space model (LGSSM)

Slow feature analysis

Audio

Video
Dimensionality reduction using probabilistic models: a family of models

- **Full linear**
  - Factor analysis (FA)
    - $p(x) = \mathcal{G}(x; 0, I)$
    - $p(y|x) = \mathcal{G}(y; \theta x, D)$
  - Inter-battery FA
    - $p(x), p(x_i) = \mathcal{G}(x; 0, I)$
    - $p(y_i|x) = \mathcal{G}(y_i; \theta_i^{sh}x + \theta_i^{pri}x_i, D)$

- **Special linear**
  - PCA
    - $D = \sigma^2 I$
    - train: eigenvalue problem
  - Canonical correlation analysis
    - $D = \sigma^2 I$
    - train: generalised eigenvalue problem
  - GP-LVM
    - multi-view GP-LVM

- **GP map**
  - $y = f(x)$

- **Special case**
  - Audio
  - Video

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

- **Slow feature analysis**
  - Find linear projections of $y_i$ that max correlation
  - Find slowest projections of unit variance
  - $D = \sigma^2 I$
  - $\Sigma = 1 - \Psi^2$
  - $\Psi = \text{diag}(\psi_1, \ldots, \psi_K)$
  - train: generalised eigenvalue problem
  - GP-dynamical system
Dimensionality reduction using probabilistic models: a family of models

**MODEL CLASS**

**full linear**
- factor analysis (FA)
  
  \[ p(x) = \mathcal{G}(x; 0, I) \]
  \[ p(y|x) = \mathcal{G}(y; \theta x, D) \]

**inter-battery FA**
- canonical correlation analysis
  
  \[ D = \sigma^2 I \]
  \[ \mathbf{D} = \sigma^2 I \]

**special linear**
- PCA
  
  \[ D = \sigma^2 \mathbf{I} \]

- GP-LVM
  
  \[ y = \mathbf{f}(x) \]

- GP-LVM multi-view
  
  \[ \Psi = \text{diag}(\psi_1, \ldots, \psi_K) \]

- GP-dynamical system
  
  \[ \Psi = \text{diag}(\psi_1, \ldots, \psi_K) \]

**other**
- many e.g. ICA
  
  \[ \Psi = \text{diag}(\psi_1, \ldots, \psi_K) \]

- information bottleneck style and content

- GP-SSM
  
  \[ \Psi = \text{diag}(\psi_1, \ldots, \psi_K) \]
Roadmap so far...

- **PCA**
- Probabilistic Modelling
- **FA**
- **pPCA**
- Max mutual information
- Embedding methods
- Min reconstruction error
- Max variance
- Linear Gaussian
- 2nd order statistics
- Linear
Roadmap so far...

- PCA
- probabilistic modelling
- FA
- pPCA
- max mutual information
- embedding methods
- MDS
- ISOMAP
- min reconstruction error
- deep NN auto-encoder
- denoising auto-encoder
- auto-encoders
- non-linear noise-corrupted input
- auto-encoders
- non-linear
- input
- linear
- linear
- linear
- distances via local reconstruction weights
- LLE
- distances via data-graph
- ISOMAP
- Laplacian eigenmaps spectral methods
- IB-FA
- CCA
- multi-view
- time-series
- LGSSM
- SFA
- 2nd order statistics
- max variance
- embedding methods
- distances via data-graph
- MDS
- non-linear metric
- distances via data-graph
Roadmap so far...

- PCA
- probabilistic modelling
- FA
- pPCA
- max mutual information embedding methods
- MDS
- ISOMAP
- min reconstruction error
- auto-encoders
- denoising auto-encoder
- noise-corrupted input
- deep NN auto-encoder
- non-linear models
- auto-encoders
- non-linear
- multi-view GP-LVM
- non-parametric
- multi-view
- IB-FA CCA
- non-linear component models
- non-linear
- non-parametric
- IB-FA CCA
- multi-view
- GP-LVM
- MAP inference
- recognition models
- auto-encoders
- linear
- linear
- linear
- non-linear
- multi-view GP-LVM
- non-parametric
- non-linear
- non-parametric dynamical GP-LVM
- non-Gaussian
- SFA
- linear
- Gaussian
- time-series
- FA pPCA
- non-Gaussian
- max mutual information
- higher-order statistics
- ICA
- 2nd order statistics
- non-Gaussian
- max variance
- embedding methods
- MDS
- distances via data-graph
- ISOMAP
- Laplacian eigenmaps spectral methods
- non-linear metric
- distances via local reconstruction weights
- LLE
- kernel-PCA
- RKHS feature expansion
- linear
- linear
- linear
- min reconstruction error
- PCA
- probabilistic modelling
- linear
- 2nd order statistics
- higher-order statistics
- ICA
- non-parametric
- non-linear
- non-linear
- non-parametric
- non-linear
- non-parametric
- non-linear
Gaussian Process Regression Model: Recap

Generative model (like non-linear regression)

\( y(x) = f(x) + \epsilon \sigma_y \)

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

place GP prior over the non-linear function

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

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

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

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

\( p(y(x)|\theta) = \mathcal{GP}(0, K(x, x') + I\sigma_y^2) \)
Gaussian Process Regression Model: Recap

Generative model (like non-linear regression)

\[ y(x) = f(x) + \epsilon \sigma_y \]

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

place GP prior over the non-linear function

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

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

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

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

\[ p(y(x)|\theta) = \mathcal{GP}(0, K(x, x') + I\sigma^2_y) \]

multi-output regression \( y_i(x) = f_i(x) + \epsilon_i \sigma_y \)

\[ p(f_i(x)|\theta) = \mathcal{GP}(0, K_i(x, x')) \]
Gaussian Process Latent Variable Model

\[ p(f_d) = \mathcal{GP}(f; 0, C(x, x')) \]
\[ p(x) = \mathcal{G}(x; 0, I) \]
\[ p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2) \]

**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

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

\[ p(x) = \mathcal{G}(x; 0, I) \]

selects position on manifold

distribution over manifolds

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

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

\[ p(f_d) = \mathcal{GP}(f; 0, C(x, x')) \]
\[ p(x) = \mathcal{G}(x; 0, 1) \]
\[ p(y_d | x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2) \]

\[ x_{\text{MAP}} = \arg \max_x p(x|y) = \arg \max_x \log p(x|y) \]
Gaussian Process Latent Variable Model: MAP Inference

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

\[ p(x) = \mathcal{G}(x; 0, I) \]

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

\[ x_{\text{MAP}} = \arg \max_x p(x|y) = \arg \max_x \log p(x|y) \]

\[ p(x|y) = \frac{1}{p(y)} p(y|x) p(x) \]
Gaussian Process Latent Variable Model: MAP Inference

\[ p(f_d) = \mathcal{GP}(f; 0, C(x, x')) \]
\[ p(x) = \mathcal{G}(x; 0, I) \]
\[ p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2) \]

\[ x_{\text{MAP}} = \arg \max_x p(x|y) = \arg \max_x \log p(x|y) \]
\[ p(x|y) = \frac{1}{p(y)} p(y|x) p(x) \]
\[ p(y|x) = \int p(y|x, f)p(f)df = \mathcal{G}(y_{1:N}; 0, \Sigma(x_{1:N})) \]

Lawrence, NIPS 2004

multi-output regression

\[ y_{1:N} = [y_{1:N,1}; \ldots; y_{1:N,D}] \]
Gaussian Process Latent Variable Model: MAP Inference

\[ p(f_d) = \mathcal{GP}(f; 0, C(x, x')) \]
\[ p(x) = \mathcal{G}(x; 0, I) \]

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

\[ x_{\text{MAP}} = \arg \max_x p(x|y) = \arg \max_x \log p(x|y) \]
\[ p(x|y) = \frac{1}{p(y)} p(y|x)p(x) \]
\[ p(y|x) = \int p(y|x, f)p(f)df = \mathcal{G}(y_{1:N}; 0, \Sigma(x_{1:N})) \]

\[ x_{\text{MAP}} = \arg \max_x \log p(x) - \frac{1}{2} \log \det \Sigma(x_{1:N}) - \frac{1}{2} \text{trace}(\Sigma(x_{1:N})^{-1} y_{1:N} y_{1:N}^\top) \]
Gaussian Process Latent Variable Model: MAP Inference

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

\[ p(x) = \mathcal{G}(x; 0, I) \]

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

\[ x_{\text{MAP}} = \arg \max_x p(x|y) = \arg \max_x \log p(x|y) \]

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

\[ p(y|x) = \int p(y|x, f)p(f)df = \mathcal{G}(y_{1:N}; 0, \Sigma(x_{1:N})) \]

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

objective depends on:

\[ ||x_n|| \quad ||x_n - x_m|| \quad ||y_n - y_m|| \quad ||y_n|| \]
Gaussian Process Latent Variable Model: MAP Inference

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

\[ p(x) = \mathcal{G}(x; 0, I) \]

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

\[ x_{\text{MAP}} = \arg \max_x p(x | y) = \arg \max_x \log p(x | y) \]

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

\[ p(y | x) = \int p(y | x, f) p(f) df = \mathcal{G}(y_{1:N}; 0, \Sigma(x_{1:N})) \]

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

objective depends on:

\[ ||x_n|| \quad ||x_n - x_m|| \quad ||y_n - y_m|| \quad ||y_n|| \]
Gaussian Process Latent Variable Model: Back-constrained Inference

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

\[
p(x) = \mathcal{G}(x; 0, I)
\]

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

Lawrence et al, ICML 2006

\[
x_{\text{MAP}} = \arg \max_x p(x|y, \theta) = \arg \max_x \log p(x, y|\theta)
\]
Gaussian Process Latent Variable Model: Back-constrained Inference

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

\[ p(x) = \mathcal{G}(x; 0, I) \]

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

Lawrence et al, ICML 2006

\[ x_{\text{MAP}} = \arg \max_x p(x|y, \theta) = \arg \max_x \log p(x, y|\theta) \]

after optimisation:

\[ x_{\text{MAP}} = x_{\text{MAP}}(y_1:N) \text{ approximate: } x_{\text{MAP}} \approx g_\phi(y_1:N) \]

recognition model or back-constraint
Gaussian Process Latent Variable Model: Back-constrained Inference

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

\[ p(x) = \mathcal{G}(x; 0, I) \]

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

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

after optimisation:

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

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

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

\[ p(x) = \mathcal{G}(x; 0, I) \]

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

Lawrence et al, ICML 2006

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

after optimisation:

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

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

\[ p(f_d) = \mathcal{G}\mathcal{P}(f; 0, C(x, x')) \]

\[ p(x) = \mathcal{G}(x; 0, I) \]

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

Lawrence et al, ICML 2006

\[ x_{\text{MAP}} = \arg \max_x p(x | y, \theta) = \arg \max_x \log p(x, y | \theta) \]

after optimisation:

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

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

\[ p(f_d) = \mathcal{GP}(f; 0, C(x, x')) \]
\[ p(x) = \mathcal{G}(x; 0, I) \]
\[ p(y_d|x, f_d) = \mathcal{G}(y; f_d(x), \sigma^2) \]

\[ x_{\text{MAP}} = \arg \max_x p(x|y, \theta) = \arg \max_x \log p(x, y|\theta) \]

after optimisation:
\[ x_{\text{MAP}} = x_{\text{MAP}}(y_{1:N}) \quad \text{approximate:} \quad x_{\text{MAP}} \approx g_\phi(y_{1:N}) \]

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: Application

Chen, Kim and Cipolla, IEEE 13th Int. Conf. on Computer Vision, 2011 and https://www.youtube.com/watch?v=X5Z7ZJ39zAA
Probabilistic inference as an auto-encoder

\[ p(x|\theta) \]

\[ p(y|x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 I) \]

Goal: learn parameters via approximate maximum-likelihood

\[ \mathcal{L}(\theta) = \log p(y|\theta) \]
Probabilistic inference as an auto-encoder

\[ p(x | \theta) \]

\[ p(y | x, \theta) = \mathcal{G}(y; f_{\theta}(x), \sigma^2 I) \]

Goal: learn parameters via approximate maximum-likelihood

\[ \mathcal{L}(\theta) = \log p(y | \theta) = \log \int p(y, x | \theta) dx = \log \int \frac{q(x)}{q(x)} p(x, y | \theta) dx \]
Probabilistic inference as an auto-encoder

$p(x|\theta)\]

$p(y|x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 I)$

Goal: learn parameters via approximate maximum-likelihood

\[
\mathcal{L}(\theta) = \log p(y|\theta) = \log \int p(y, x|\theta)dx = \log \int \frac{q(x)}{q(x)} p(x, y|\theta)dx
\]

\[
\mathcal{L}(\theta) \geq \int q(x) \log \frac{1}{q(x)} p(y, x|\theta)dx = \mathcal{F}(q, \theta)
\]

Jensen's inequality

free-energy
Probabilistic inference as an auto-encoder

\[
\begin{align*}
  x & \quad p(x|\theta) \\
  y & \quad p(y|x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 I)
\end{align*}
\]

Goal: learn parameters via approximate maximum-likelihood

\[
\mathcal{L}(\theta) = \log p(y|\theta) = \log \int p(y, x|\theta)dx = \log \int \frac{q(x)}{q(x)} p(x, y|\theta)dx
\]

Jensen's

\[
\mathcal{L}(\theta) \geq \int q(x) \log \frac{1}{q(x)} p(y, x|\theta)dx = \mathcal{F}(q, \theta) \quad \text{free-energy}
\]

\[
\mathcal{F}(q, \theta) = \int q(x) \log p(y|x, \theta)dx + \int q(x) \log \frac{p(x|\theta)}{q(x)} dx
\]
Probabilistic inference as an auto-encoder

\[ x \quad p(x|\theta) \]

\[ y \quad p(y|x, \theta) = G(y; f_\theta(x), \sigma^2I) \]

Goal: learn parameters via approximate maximum-likelihood

\[ \mathcal{L}(\theta) = \log p(y|\theta) = \log \int p(y, x|\theta)dx = \log \int \frac{q(x)}{q(x)}p(x, y|\theta)dx \]

Jensen's inequality:

\[ \mathcal{L}(\theta) \geq \int q(x) \log \frac{1}{q(x)}p(y, x|\theta)dx = \mathcal{F}(q, \theta) \]

\[ \mathcal{F}(q, \theta) = \int q(x) \log p(y|x, \theta)dx + \int q(x) \log \frac{p(x|\theta)}{q(x)}dx \]

\[ \mathcal{F}(q, \theta) = -\frac{1}{2\sigma^2} \langle||y - f_\theta(x)||^2\rangle_{q(x)} - \frac{D}{2} \log \sigma^2 - \text{KL}(q(x)||p(x)) \]
Probabilistic inference as an auto-encoder

\[ p(x|\theta) \]

\[ p(y|x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 I) \]

Goal: learn parameters via approximate maximum-likelihood free-energy reconstruction cost soft-constraint

\[ \mathcal{L}(\theta) = \log p(y|\theta) = \log \int p(y, x|\theta)dx = \log \int \frac{q(x)}{q(x)} p(x, y|\theta)dx \]

Jensen's inequality:

\[ \mathcal{L}(\theta) \geq \int q(x) \log \frac{1}{q(x)} p(y, x|\theta)dx = \mathcal{F}(q, \theta) \]

\[ \mathcal{F}(q, \theta) = \int q(x) \log p(y|x, \theta)dx + \int q(x) \log \frac{p(x|\theta)}{q(x)} dx \]

reconstruction cost soft-constraint

\[ \mathcal{F}(q, \theta) = -\frac{1}{2\sigma^2} \langle ||y - f_\theta(x)||^2 \rangle_{q(x)} - \frac{D}{2} \log \sigma^2 - \text{KL}(q(x) \| p(x)) \]
Probabilistic inference as an auto-encoder

\[ p(x|\theta) \]

\[ p(y|x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 I) \]

Goal: learn parameters via approximate maximum-likelihood flavours of variational inference:

\[ \mathcal{F}(q, \theta) = -\frac{1}{2\sigma^2} \langle \|y - f_\theta(x)\|^2 \rangle_{q(x)} - \frac{D}{2} \log \sigma^2 - \text{KL}(q(x)\|p(x)) \]

reconstruction cost  soft-constraint
Probabilistic inference as an auto-encoder

\[ p(x|\theta) \]

\[ p(y|x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 I) \]

Goal: learn parameters via approximate maximum-likelihood flavours of variational inference:

fixed family \[ q(x) = \mathcal{G}(x; \mu_q, \Sigma_q) \]

\[
\arg \max_{\theta, \mu_q, \Sigma_q} \mathcal{F}(\mu_q, \Sigma_q, \theta)
\]

reconstruction cost

\[
\mathcal{F}(q, \theta) = -\frac{1}{2\sigma^2} \langle \| y - f_\theta(x) \|^2 \rangle_q - \frac{D}{2} \log \sigma^2 - \text{KL}(q(x)||p(x))
\]
Probabilistic inference as an auto-encoder

\[ p(x|\theta) \]

\[ p(y|x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 I) \]

Goal: learn parameters via approximate maximum-likelihood flavours of variational inference:

fixed family \[ q(x) = \mathcal{G}(x; \mu_q, \Sigma_q) \]

structured \[ q(x) = \prod_{k=1}^{K} q_k(x_k) \]

\[
\begin{aligned}
\mathcal{F}(q, \theta) &= -\frac{1}{2\sigma^2} \langle \| y - f_\theta(x) \|^2 \rangle_{q(x)} - \frac{D}{2} \log \sigma^2 - \text{KL}(q(x) || p(x)) \\
\end{aligned}
\]
Probabilistic inference as an auto-encoder

\[ p(x|\theta) \]

\[ p(y|x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 I) \]

Goal: learn parameters via approximate maximum-likelihood flavours of variational inference:

- **fixed family**
  \[ q(x) = \mathcal{G}(x; \mu_q, \Sigma_q) \]
  \[ \arg \max_{\theta, \mu_q, \Sigma_q} \mathcal{F}(\mu_q, \Sigma_q, \theta) \]

- **structured**
  \[ q(x) = \prod_{k=1}^{K} q_k(x_k) \]
  \[ \arg \max_{\{q_k(x)\}_{k=1}^{K}, \theta} \mathcal{F}(\{q_k(x)\}_{k=1}^{K}, \theta) \]

- **recognition model**
  \[ q_\phi(x) = \mathcal{G}(x; \mu_\phi(y), \Sigma_\phi(y)) \]
  \[ \arg \max_{\phi, \theta} \mathcal{F}(\phi, \theta) \text{ variational auto-encoder} \]

**reconstruction cost**
\[ \mathcal{F}(q, \theta) = -\frac{1}{2\sigma^2} \langle \|y - f_\theta(x)\|^2 \rangle_{q(x)} - \frac{D}{2} \log \sigma^2 - \text{KL}(q(x)||p(x)) \]

**soft-constraint**
\[ \mathcal{L}(\theta) \]
Probabilistic inference as an auto-encoder

Goal: learn parameters via approximate maximum-likelihood flavours of variational inference:

- **fixed family**
  \[ q(x) = \mathcal{G}(x; \mu_q, \Sigma_q) \]
  \[ \arg \max_{\theta, \mu_q, \Sigma_q} \mathcal{F}(\mu_q, \Sigma_q, \theta) \]

- **structured**
  \[ q(x) = \prod_{k=1}^{K} q_k(x_k) \]
  \[ \arg \max_{\{q_k(x)\}_{k=1}^{K}, \theta} \mathcal{F}(\{q_k(x)\}_{k=1}^{K}, \theta) \]

- **recognition model**
  \[ q_\phi(x) = \mathcal{G}(x; \mu_\phi(y), \Sigma_\phi(y)) \]
  \[ \arg \max_{\phi, \theta} \mathcal{F}(\phi, \theta) \]

**variational auto-encoder**

**GP-LVM:**
\[ q_\phi(x) = \delta(x - g_\phi(y)) \]

**reconstruction cost**
\[ \mathcal{F}(q, \theta) = -\frac{1}{2\sigma^2} \langle \|y - f_\theta(x)\|^2 \rangle_{q(x)} - \frac{D}{2} \log \sigma^2 - \text{KL}(q(x)\|p(x)) \]
Probabilistic inference as an auto-encoder

\[ p(x | \theta) \]

\[ p(y | x, \theta) = \mathcal{G}(y; f_\theta(x), \sigma^2 \mathbf{I}) \]

Goal: learn parameters via approximate maximum-likelihood

flavours of variational inference:

- fixed family
  \[ q(x) = \mathcal{G}(x; \mu_q, \Sigma_q) \]
  \[ \arg \max_{\theta, \mu_q, \Sigma_q} \mathcal{F}(\mu_q, \Sigma_q, \theta) \]

- structured
  \[ q(x) = \prod_{k=1}^{K} q_k(x_k) \]
  \[ \arg \max_{\{q_k(x)\}_{k=1}^{K}, \theta} \mathcal{F}(\{q_k(x)\}_{k=1}^{K}, \theta) \]

- recognition model
  \[ q_\phi(x) = \mathcal{G}(x; \mu_\phi(y), \Sigma_\phi(y)) \]
  \[ \arg \max_{\phi, \theta} \mathcal{F}(\phi, \theta) \]

reconstruction cost

\[ \mathcal{F}(q, \theta) = -\frac{1}{2\sigma^2} \mathbb{E}[\|y - f_\theta(x)\|^2]_{q(x)} - \frac{D}{2} \log \sigma^2 - \text{KL}(q(x) \| p(x)) \]

GP-LVM:

\[ q_\phi(x) = \delta(x - g_\phi(y)) \]
Dimensionality reduction: conceptual space

- PCA
- probabilistic modelling
- FA
- pPCA
- max mutual information
- embedding methods
- MDS
- ISOMAP
- min reconstruction error
- auto-encoders
- deep NN auto-encoder
- denoising auto-encoder
- noise-corrupted input
- multi-view
- GP-LVM
- non-parametric
- recognition models
- linear NN recognition models
- non-linear NN recognition models
- multi-view GP-LVM
- non-linear component models
- IB-FA
- CCA
- non-parametric
- linear Gaussian component models
- PCA
- FA
- pPCA
- non-linear component models
- LGSSM
- SFA
- non-parametric
- dynamical GP-LVM
- ICA
- higher-order statistics
- 2nd order statistics
- time-series
- non-Gaussian
- non-Gaussian component models
- MAP inference
- recognition models
- MAP inference
- deep NN recognition models
- non-linear recognition models
- k-PCA
- RKHS feature expansion
- kernel-PCA
- max variance
- embedding methods
- non-linear embedding methods
- MDS
- distances via data-graph
- ISOMAP
- Laplacian eigenmaps
- spectral methods
- LLE
- distances via local reconstruction weights
- non-linear metric
- local reconstruction weights