CITEC SummerSchool 2013
Learning – From Physics to Knowledge
Selected Learning Methods

Robert Haschke

Neuroinformatics Group
CITEC, Bielefeld University

September, 11th 2013

Outline

Outline

Feature Space Expansion

High-dimensional feature spaces facilitate computations,
e.g. enable *linear separability* of class regions

- polynomial expansion: $(x_1..x_d) \to (x_1..x_d, .., x_i x_j .., x_i x_j x_k)$
- time history: use $(\mathbf{x}_t, \mathbf{x}_{t-1} \dots \mathbf{x}_{t-k}) \in \mathbb{R}^{d\cdot(k+1)}$
- filters – temporal convolution with kernels: $\bar{\mathbf{x}}(t) = \int K(t, t') \cdot \mathbf{x}(t')\, dt'$
- Kernel trick

Kernel Trick

- *linear* regressors or perceptrons often have the form

$$y(\mathbf{x}) = \mathbf{w}^t \cdot \mathbf{x} = \sum_{\alpha=1}^{N} \lambda_\alpha \cdot \mathbf{x}_\alpha \cdot \mathbf{x}$$

Kernel Trick

- *linear* regressors or perceptrons often have the form

$$y(\mathbf{x}) = \mathbf{w}^t \cdot \mathbf{x} = \sum_{\alpha=1}^{N} \lambda_\alpha \cdot \mathbf{x}_\alpha \cdot \mathbf{x} = \sum_{\alpha} \mathbf{w}_\alpha \cdot \phi(\mathbf{x}_\alpha) \cdot \phi(\mathbf{x})$$

- goal: introduce non-linear, high-dimensional features $\phi_k(\mathbf{x})$

Kernel Trick

- *linear* regressors or perceptrons often have the form

$$y(\mathbf{x}) = \mathbf{w}^t \cdot \mathbf{x} = \sum_{\alpha=1}^{N} \lambda_\alpha \cdot \mathbf{x}_\alpha \cdot \mathbf{x} = \sum_\alpha \mathbf{w}_\alpha \cdot \phi(\mathbf{x}_\alpha) \cdot \phi(\mathbf{x}) = \sum_\alpha \mathbf{w}_\alpha \cdot K(\mathbf{x}_\alpha, \mathbf{x})$$

- goal: introduce non-linear, high-dimensional features $\phi_k(\mathbf{x})$
- kernel directly computes scalar product of feature vectors

Kernel Trick

- *linear* regressors or perceptrons often have the form

$$y(\mathbf{x}) = \mathbf{w}^t \cdot \mathbf{x} = \sum_{\alpha=1}^{N} \lambda_\alpha \cdot \mathbf{x}_\alpha \cdot \mathbf{x} = \sum_\alpha \mathbf{w}_\alpha \cdot \phi(\mathbf{x}_\alpha) \cdot \phi(\mathbf{x}) = \sum_\alpha \mathbf{w}_\alpha \cdot K(\mathbf{x}_\alpha, \mathbf{x})$$

- goal: introduce non-linear, high-dimensional features $\phi_k(\mathbf{x})$
- kernel directly computes scalar product of feature vectors
- typical examples, e.g. in support vector machines (SVM):
  - polynomial kernel: $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^p$
  - Gaussian kernel: $K(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2})$

Example: Bayesian Linear Regression

$$f(x, w) = \mathbf{w}^t \cdot \phi(\mathbf{x})$$

$$y = f(\mathbf{x}, \mathbf{w}) + \eta$$

Example: Bayesian Linear Regression

$$f(x, w) = \mathbf{w}^t \cdot \phi(\mathbf{x})$$

$$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^t \phi(\mathbf{x}), \beta^{-1} \mathbf{1})$$

$$y = f(\mathbf{x}, \mathbf{w}) + \eta$$

Gaussian data distribution

Example: Bayesian Linear Regression

$$f(x, w) = \mathbf{w}^t \cdot \phi(\mathbf{x})$$

$$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^t \phi(\mathbf{x}), \beta^{-1}\mathbf{1})$$

$$\hat{\mathbf{w}}_{\text{ML}} = (\Phi^t \Phi)^{-1} \Phi^t \mathbf{y}$$

(maximum likelihood estimator)

$$y = f(\mathbf{x}, \mathbf{w}) + \eta$$

Gaussian data distribution

$$\Phi^t = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)] \in \mathbb{R}^{d \times N}$$

$$\mathbf{y}^t = [y_1, \ldots, y_N] \in \mathbb{R}^N$$

Example: Bayesian Linear Regression

$$f(x, w) = \mathbf{w}^t \cdot \phi(\mathbf{x})$$

$$y = f(\mathbf{x}, \mathbf{w}) + \eta$$

$$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^t \phi(\mathbf{x}), \beta^{-1}\mathbf{1})$$

Gaussian data distribution

$$\hat{\mathbf{w}}_{\mathrm{ML}} = (\Phi^t \Phi)^{-1} \Phi^t \mathbf{y}$$

(maximum likelihood estimator)

$$\Phi^t = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)] \in \mathbb{R}^{d \times N}$$

$$\mathbf{y}^t = [y_1, \ldots, y_N] \in \mathbb{R}^N$$

$$p(\mathbf{w}) = \mathcal{N}(0, \alpha^{-1}\mathbf{1})$$

Gaussian a-priori distribution

Example: Bayesian Linear Regression

$$f(x, w) = \mathbf{w}^t \cdot \phi(\mathbf{x})$$

$$y = f(\mathbf{x}, \mathbf{w}) + \eta$$

$$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^t \phi(\mathbf{x}), \beta^{-1}\mathbf{1})$$

Gaussian data distribution

$$\hat{\mathbf{w}}_{\mathrm{ML}} = (\Phi^t \Phi)^{-1} \Phi^t \mathbf{y}$$

(maximum likelihood estimator)

$$\Phi^t = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)] \in \mathbb{R}^{d \times N}$$

$$\mathbf{y}^t = [y_1, \ldots, y_N] \in \mathbb{R}^N$$

$$p(\mathbf{w}) = \mathcal{N}(0, \alpha^{-1}\mathbf{1})$$

Gaussian a-priori distribution

$$p(\mathbf{w} \mid y) = \mathcal{N}(m_N, S_N)$$

a-posteriori distribution

$$\hat{\mathbf{w}}_{\mathrm{MAP}} = m_N = \beta S_N \Phi^t \mathbf{y}$$

MAP estimator

$$S_N = (\alpha \mathbf{1} + \beta \Phi^t \Phi)^{-1} \in \mathbb{R}^{d \times d}$$

Example: Bayesian Linear Regression

$$f(x, w) = \mathbf{w}^t \cdot \phi(\mathbf{x}) \qquad y = f(\mathbf{x}, \mathbf{w}) + \eta$$

$$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^t \phi(\mathbf{x}), \beta^{-1}\mathbf{1})$$ Gaussian data distribution

$$\hat{\mathbf{w}}_{\text{ML}} = (\Phi^t \Phi)^{-1} \Phi^t \mathbf{y} \qquad \Phi^t = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)] \in \mathbb{R}^{d \times N}$$

(maximum likelihood estimator) $\qquad \mathbf{y}^t = [y_1, \ldots, y_N] \in \mathbb{R}^N$

$$p(\mathbf{w}) = \mathcal{N}(0, \alpha^{-1}\mathbf{1})$$ Gaussian a-priori distribution

$$p(\mathbf{w} \mid y) = \mathcal{N}(m_N, S_N)$$ a-posteriori distribution

$$\hat{\mathbf{w}}_{\text{MAP}} = m_N = \beta S_N \Phi^t \mathbf{y}$$ MAP estimator

$$S_N = (\alpha \mathbf{1} + \beta \Phi^t \Phi)^{-1} \qquad \in \mathbb{R}^{d \times d}$$

$$\hat{y}(\mathbf{x}) = \phi(\mathbf{x})^t \cdot \hat{\mathbf{w}}_{\text{MAP}} = \beta \cdot \phi(\mathbf{x})^t S_N \Phi^t \cdot \mathbf{y} = \sum_\alpha \beta \cdot \phi(\mathbf{x})^t S_N \phi(\mathbf{x}_\alpha) \cdot y_\alpha$$

Example: Bayesian Linear Regression

$$f(x, w) = \mathbf{w}^t \cdot \phi(\mathbf{x}) \qquad y = f(\mathbf{x}, \mathbf{w}) + \eta$$

$$p(y \mid \mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{w}^t \phi(\mathbf{x}), \beta^{-1}\mathbf{1})$$ Gaussian data distribution

$$\hat{\mathbf{w}}_{\text{ML}} = (\Phi^t \Phi)^{-1} \Phi^t \mathbf{y} \qquad \Phi^t = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N)] \in \mathbb{R}^{d \times N}$$

(maximum likelihood estimator) $\qquad \mathbf{y}^t = [y_1, \dots, y_N] \in \mathbb{R}^N$

$$p(\mathbf{w}) = \mathcal{N}(0, \alpha^{-1}\mathbf{1})$$ Gaussian a-priori distribution

$$p(\mathbf{w} \mid y) = \mathcal{N}(m_N, S_N)$$ a-posteriori distribution

$$\hat{\mathbf{w}}_{\text{MAP}} = m_N = \beta S_N \Phi^t \mathbf{y}$$ MAP estimator

$$S_N = (\alpha \mathbf{1} + \beta \Phi^t \Phi)^{-1} \qquad \in \mathbb{R}^{d \times d}$$

$$\hat{y}(\mathbf{x}) = \phi(\mathbf{x})^t \cdot \hat{\mathbf{w}}_{\text{MAP}} = \beta \cdot \phi(\mathbf{x})^t S_N \Phi^t \cdot \mathbf{y} = \sum_\alpha \beta \cdot \phi(\mathbf{x})^t S_N \phi(\mathbf{x}_\alpha) \cdot y_\alpha$$

$$= \sum_\alpha K(\mathbf{x}, \mathbf{x}_\alpha) \cdot y_\alpha \qquad K(\mathbf{x}, \mathbf{x}_\alpha) = \beta \phi(\mathbf{x})^t S_N \phi(\mathbf{x}_\alpha)$$

Outline

Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$\begin{aligned} p(y \mid \mathbf{x}) &= \mathcal{N}(\hat{\mathbf{w}}^t_{\text{MAP}} \cdot \phi(\mathbf{x}),\ \sigma^2_N(\mathbf{x})) \\ \sigma^2_N(\mathbf{x}) &= \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x}) \end{aligned}$$

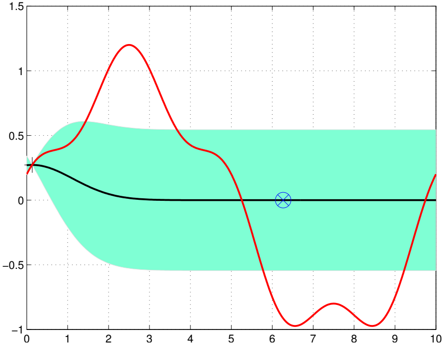
- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\text{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

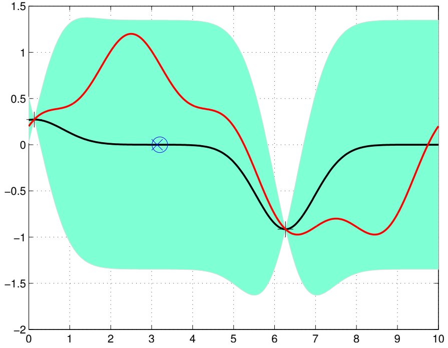


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$\begin{aligned} p(y \mid \mathbf{x}) &= \mathcal{N}(\hat{\mathbf{w}}_{\text{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x})) \\ \sigma_N^2(\mathbf{x}) &= \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x}) \end{aligned}$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

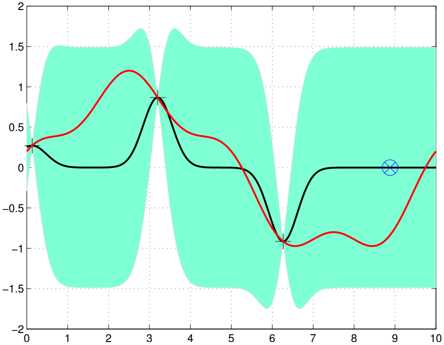


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\text{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

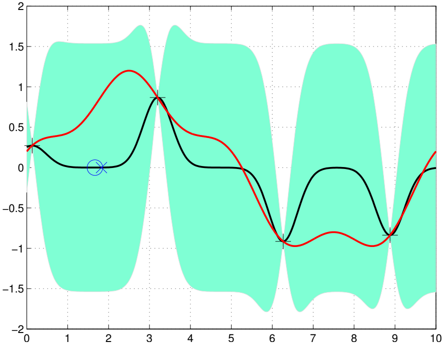


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\text{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

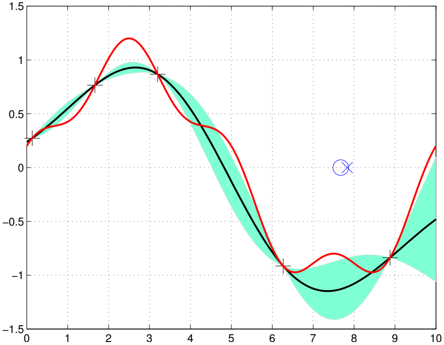


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\text{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

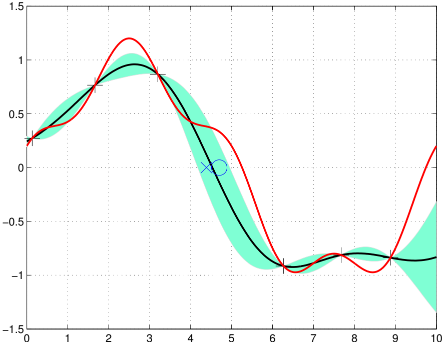


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\mathrm{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\mathrm{MAP}}$

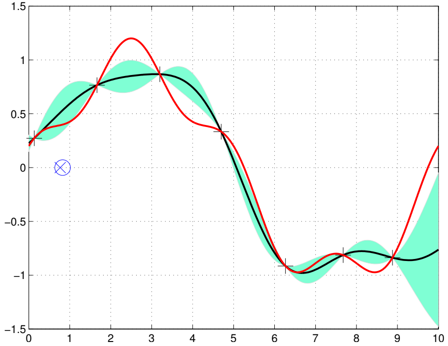


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}^t_{\text{MAP}} \cdot \phi(\mathbf{x}),\ \sigma^2_N(\mathbf{x}))$$
$$\sigma^2_N(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

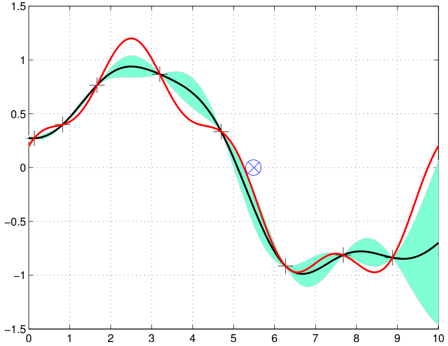


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\text{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

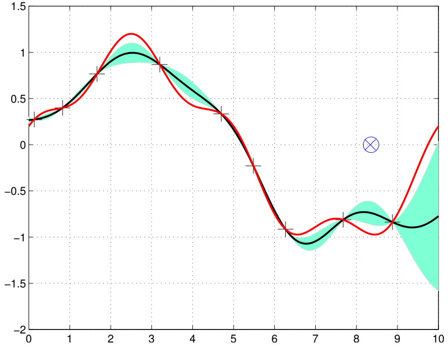


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$
\begin{aligned}
p(y \mid \mathbf{x}) &= \mathcal{N}(\hat{\mathbf{w}}_{\text{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x})) \\
\sigma_N^2(\mathbf{x}) &= \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})
\end{aligned}
$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

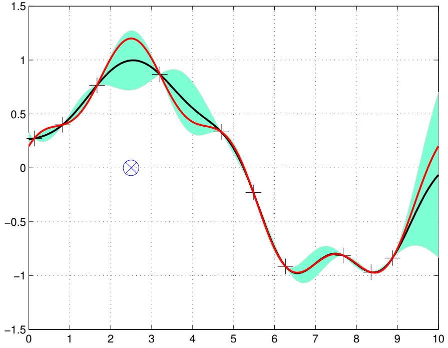


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\mathrm{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\mathrm{MAP}}$

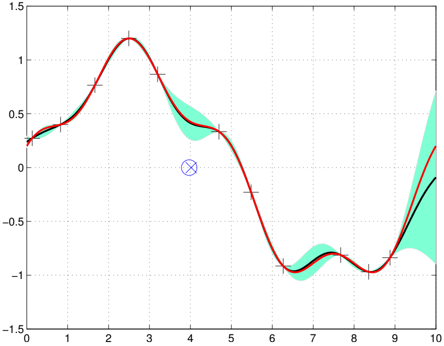


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$
\begin{aligned}
p(y \mid \mathbf{x}) &= \mathcal{N}(\hat{\mathbf{w}}_{\mathrm{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x})) \\
\sigma_N^2(\mathbf{x}) &= \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})
\end{aligned}
$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\mathrm{MAP}}$

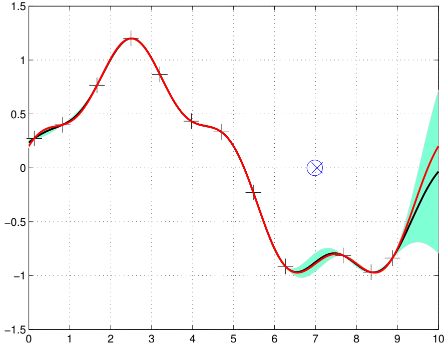


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\text{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\text{MAP}}$

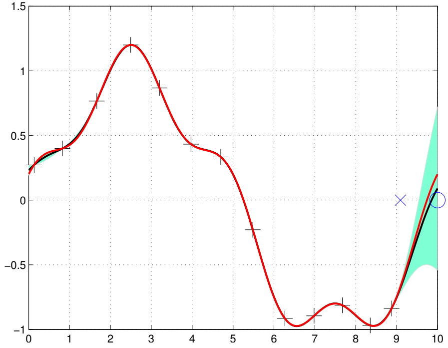


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$
\begin{aligned}
p(y \mid \mathbf{x}) &= \mathcal{N}(\hat{\mathbf{w}}^t_{\mathrm{MAP}} \cdot \phi(\mathbf{x}),\ \sigma^2_N(\mathbf{x})) \\
\sigma^2_N(\mathbf{x}) &= \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})
\end{aligned}
$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\mathrm{MAP}}$

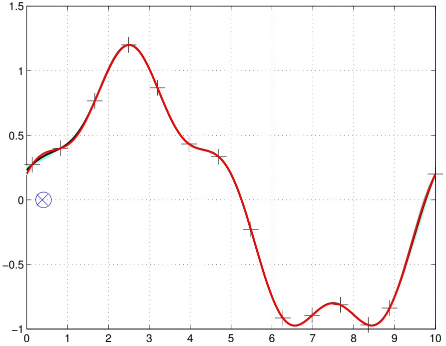


Gaussian Processes

- predictive distribution of Bayesian Linear Regression:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\hat{\mathbf{w}}_{\mathrm{MAP}}^t \cdot \phi(\mathbf{x}),\ \sigma_N^2(\mathbf{x}))$$
$$\sigma_N^2(\mathbf{x}) = \beta^{-1} + \phi(\mathbf{x})^t \cdot S_N \cdot \phi(\mathbf{x})$$

- variance: data noise + uncertainty of $\hat{\mathbf{w}}_{\mathrm{MAP}}$



Gaussian Processes

- distribution over parameters $\mathbf{w}$ induces distribution over estimator functions
- idea of Gaussian Processes: directly operate on function distributions, skipping the parameter representation

Gaussian Processes

- distribution over parameters $\mathbf{w}$ induces distribution over estimator functions
- idea of Gaussian Processes: directly operate on function distributions, skipping the parameter representation

Definition

- A Gaussian process is a collection of random variables, any finite collection of which has a joint Gaussian distribution.
- A Gaussian process is fully specified by a mean function $\bar{f}(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$.

Gaussian Processes

- distribution over parameters $\mathbf{w}$ induces distribution over estimator functions
- idea of Gaussian Processes: directly operate on function distributions, skipping the parameter representation

Definition

- A Gaussian process is a collection of random variables, any finite collection of which has a joint Gaussian distribution.
- A Gaussian process is fully specified by a mean function $\bar{f}(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$.

Example

- linear model $f(\mathbf{x}) = \mathbf{w}^t \phi(\mathbf{x})$ with Gaussian prior $p(\mathbf{w}) = \mathcal{N}(0, \alpha^{-1}\mathbf{1})$
- any tuple $[f(\mathbf{x}_1) \dots f(\mathbf{x}_N)]$ has Gaussian distribution (as linear combination of Gaussian distributed variables $\mathbf{w}$)

Gaussian Processes

Definition

- A Gaussian process is a collection of random variables, any finite collection of which has a joint Gaussian distribution.
- A Gaussian process is fully specified by a mean function $\bar{f}(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$.

Example

- linear model $f(\mathbf{x}) = \mathbf{w}^t \phi(\mathbf{x})$ with Gaussian prior $p(\mathbf{w}) = \mathcal{N}(0, \alpha^{-1}\mathbf{1})$
- any tuple $[f(\mathbf{x}_1) \dots f(\mathbf{x}_N)]$ has Gaussian distribution (as linear combination of Gaussian distributed variables $\mathbf{w}$)

Specification: mean + covariance function

- mean: $\mathbb{E}[f(\mathbf{x})] = \mathbb{E}[\mathbf{w}^t] \cdot \phi(\mathbf{x}) = 0$
- covariance: $\mathbb{E}[f(\mathbf{x}) f(\mathbf{x}')] = \phi(\mathbf{x})^t \, \mathbb{E}[\mathbf{w}\mathbf{w}^t] \, \phi(\mathbf{x}') = \alpha^{-1} \phi(\mathbf{x})^t \cdot \phi(\mathbf{x}') \equiv k(\mathbf{x}, \mathbf{x}')$

Gaussian Process Regression

- How we can exploit GPs for regression?
- N training points $\mathbf{x}_1 \dots \mathbf{x}_N$ induce a Gaussian distribution of associated function values $\mathbf{f}_N = [f(\mathbf{x}_1) \dots f(\mathbf{x}_N)]$.

Gaussian Process Regression

- How we can exploit GPs for regression?
- N training points $\mathbf{x}_1 \ldots \mathbf{x}_N$ induce a Gaussian distribution of associated function values $\mathbf{f}_N = [f(\mathbf{x}_1) \ldots f(\mathbf{x}_N)]$.
- Embedding an additional $N+1$-th "query point" $\mathbf{x}_{N+1}$ again yields a Gaussian distribution, now of $\mathbf{f}_{N+1} = [f(\mathbf{x}_1) \ldots f(\mathbf{x}_N), f(\mathbf{x}_{N+1})]$.

Gaussian Process Regression

- How we can exploit GPs for regression?
- N training points $\mathbf{x}_1 \dots \mathbf{x}_N$ induce a Gaussian distribution of associated function values $\mathbf{f}_N = [f(\mathbf{x}_1) \dots f(\mathbf{x}_N)]$.
- Embedding an additional $N+1$-th "query point" $\mathbf{x}_{N+1}$ again yields a Gaussian distribution, now of $\mathbf{f}_{N+1} = [f(\mathbf{x}_1) \dots f(\mathbf{x}_N), f(\mathbf{x}_{N+1})]$.
- Evaluate predictive distribution $p(f(\mathbf{x}_{N+1}) \mid f(\mathbf{x}_1) \dots f(\mathbf{x}_N))$.

Gaussian Process Regression

Computational Steps

$$
\begin{aligned}
p(\mathbf{f}_{N+1}) &= \mathcal{N}(0, K_{N+1}) \\
K_{N+1} &= \begin{pmatrix} K_N & \mathbf{k} \\ \mathbf{k}^t & c \end{pmatrix} \in \mathbb{R}^{(N+1)\times(N+1)} \\
K_N(\mathbf{x}_n, \mathbf{x}_m) &= k(\mathbf{x}_n, \mathbf{x}_m) \\
\mathbf{k} &= [k(\mathbf{x}_1, \mathbf{x}_{N+1}) \ldots k(\mathbf{x}_N, \mathbf{x}_{N+1})] \in \mathbb{R}^N \\
c &= k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1})
\end{aligned}
$$

Gaussian Process Regression

Computational Steps

$$
\begin{aligned}
p(\mathbf{f}_{N+1}) &= \mathcal{N}(0, K_{N+1}) \\
K_{N+1} &= \begin{pmatrix} K_N & \mathbf{k} \\ \mathbf{k}^t & c \end{pmatrix} \in \mathbb{R}^{(N+1)\times(N+1)} \\
K_N(\mathbf{x}_n, \mathbf{x}_m) &= k(\mathbf{x}_n, \mathbf{x}_m) \\
\mathbf{k} &= [k(\mathbf{x}_1, \mathbf{x}_{N+1}) \dots k(\mathbf{x}_N, \mathbf{x}_{N+1})] \in \mathbb{R}^N \\
c &= k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1})
\end{aligned}
$$

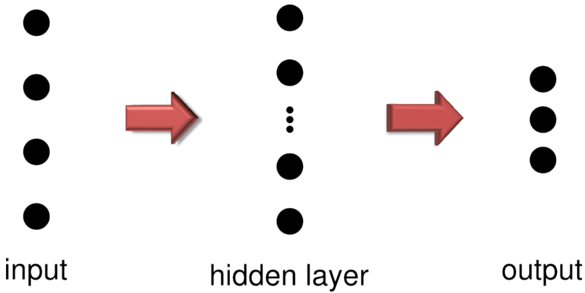
predictive distribution

- $p(f_{N+1} \mid \mathbf{f}_N)$ is again Gaussian
- with mean $\mu(\mathbf{x}_{N+1}) = \mathbf{k}^t \cdot K_N^{-1} \cdot \mathbf{f}_N$
- and variance $\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^t \cdot K_N^{-1} \cdot \mathbf{k}$

Outline

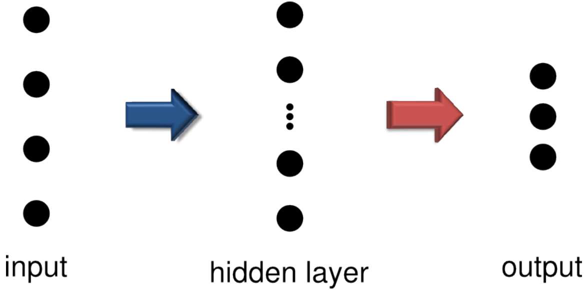
Multilayer Perceptron

- hidden layer serves as high-dimensional feature space
- back propagation creates "optimal" features
- but: input weights adapt only slowly



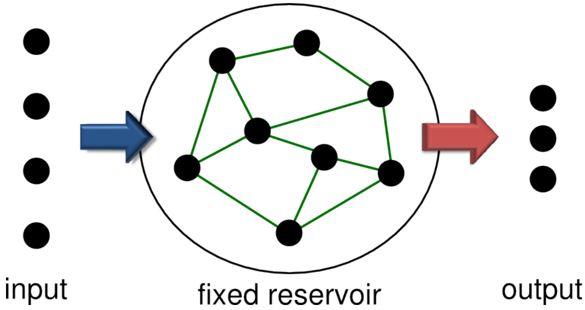
Extreme Learning Machine

- simplification:
  fixed, random input features
- linear readout facilitates
  learning: regression methods



Reservoir Computing

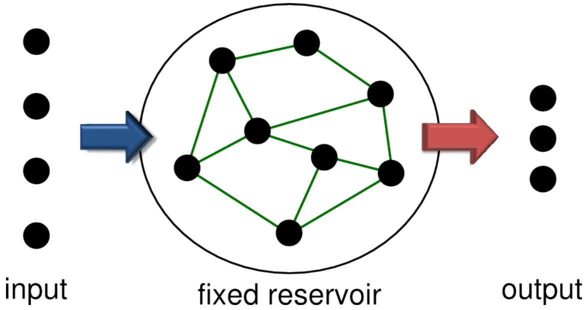
- recurrent reservoir
- allows for temporal dynamics
- even more rich feature space
- linear readout: regression



Reservoir Computing

- recurrent reservoir
- allows for temporal dynamics
- even more rich feature space
- linear readout: regression

- echo state network
- liquid state machine
- backpropagation-decorrelation

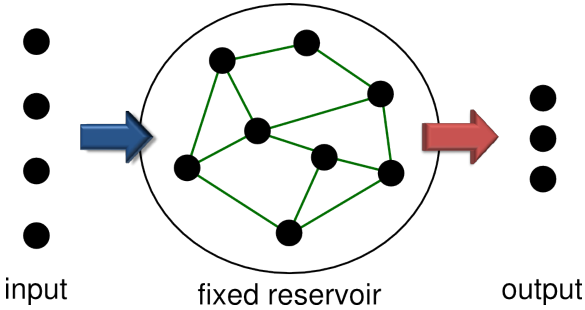


Reservoir Computing

- recurrent reservoir
- allows for temporal dynamics
- even more rich feature space
- linear readout: regression

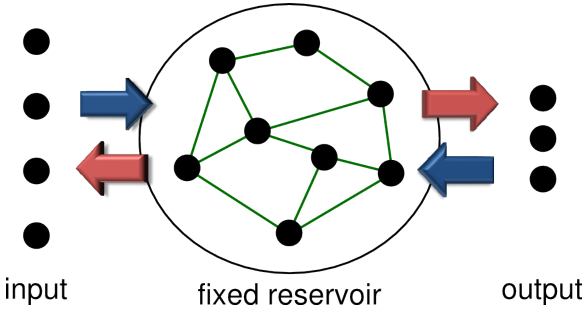
- echo state network
- liquid state machine
- backpropagation-decorrelation

- How to tune the reservoir?
  echo state property: activity decays without excitation



Associative Reservoir Network [Steil]

- learning in both directions
- input forcing
- bidirectional association
  - forward mapping
  - inverse mapping

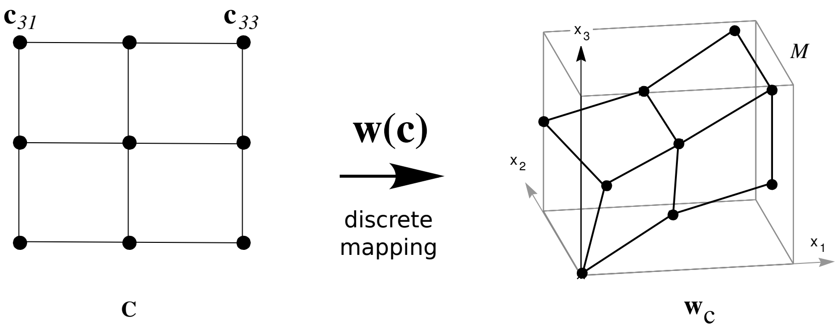


Outline

Outline

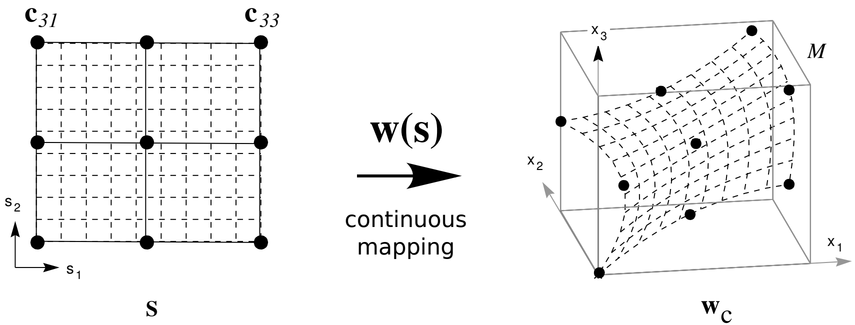
Parameterized SOM [Ritter 1993]

- fast learning of functions and inverses
- generalization of self-organizing maps (SOM) from discrete to continuous manifolds



Parameterized SOM [Ritter 1993]

- fast learning of functions and inverses
- generalization of self-organizing maps (SOM) from discrete to continuous manifolds



coordinate system S
spanned by nodes $\boldsymbol{c} \in A$

manifold M
in input space X

PSOM: Function Modeling

- linear superposition of basis functions $H(c, s)$:

$$\mathbf{w}(s) = \sum_{c \in \mathcal{A}} H(c, s)\mathbf{w}_c$$

- $H(c, s)$ should form an orthonormal basis system:

$$H(c, c') = \delta_{cc'}$$

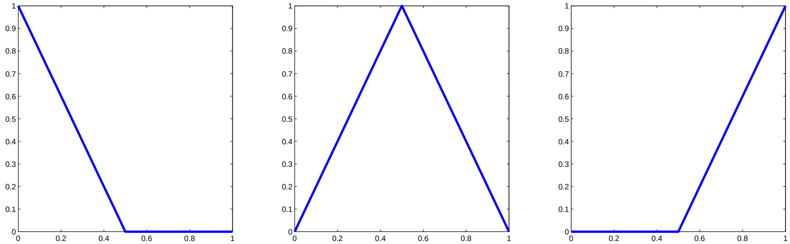
- representing constant functions:

$$\forall s \in S \quad \sum_{c \in \mathcal{A}} H(c, s) = 1$$

PSOM: Function Modeling

- simple polynomials along every grid dimension

  piecewise linear functions:

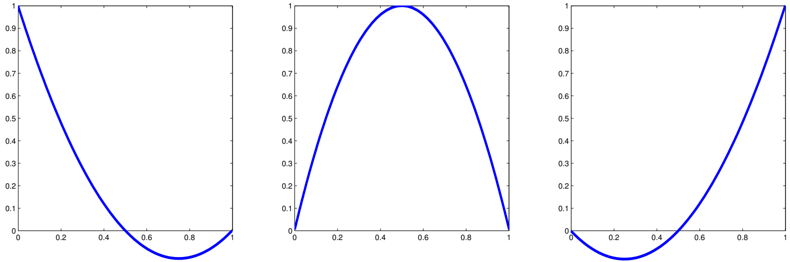


- multiplicative combination of multiple grid dimensions: $H(\mathbf{c},\mathbf{s}) = \prod_{\nu=1}^{m} h^{\nu}(s_{\nu}, \mathcal{A}_{\nu})$

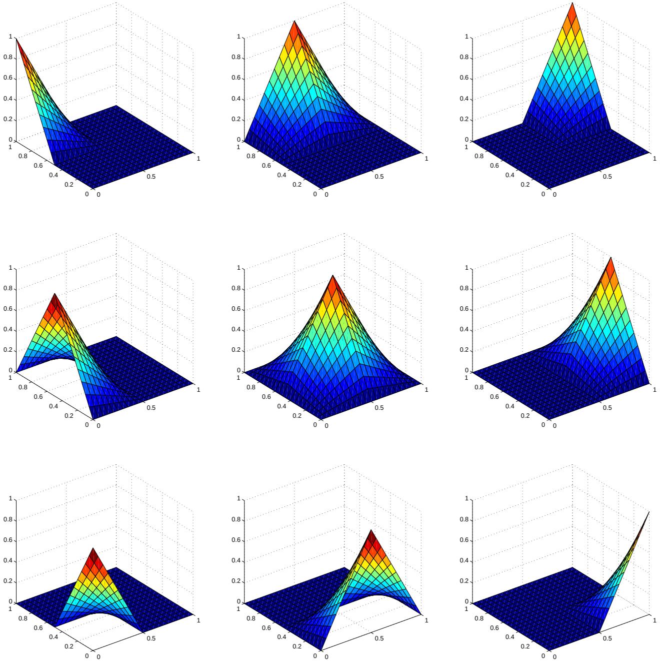
PSOM: Function Modeling

- simple polynomials along every grid dimension

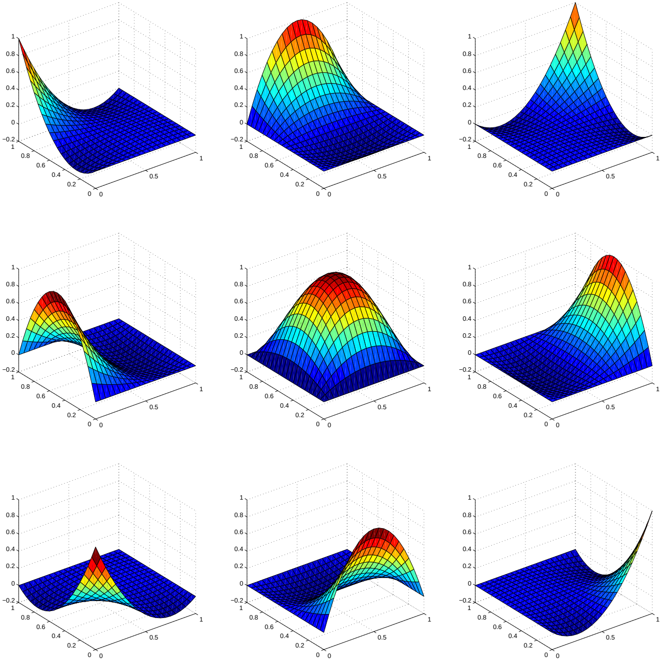
  Lagrange polynomials: $h(c,s) = \prod_{c' \in \mathcal{A},\, c' \neq c} \frac{s - c'}{c - c'}$



- multiplicative combination of multiple grid dimensions: $H(\mathbf{c}, \mathbf{s}) = \prod_{\nu=1}^{m} h^{\nu}(s_{\nu}, \mathcal{A}_{\nu})$



linear basis functions



quadratic basis functions

Inverse Mapping

- find coordinates $\mathbf{s}^*$ closest to observation $\mathbf{w}$

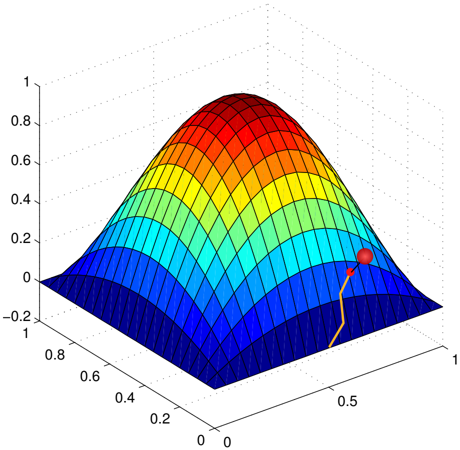
$$\mathbf{s}^* = \arg\min_{\mathbf{s}\in S} \|\mathbf{w} - \mathbf{w}(\mathbf{s})\|$$

- using gradient descent

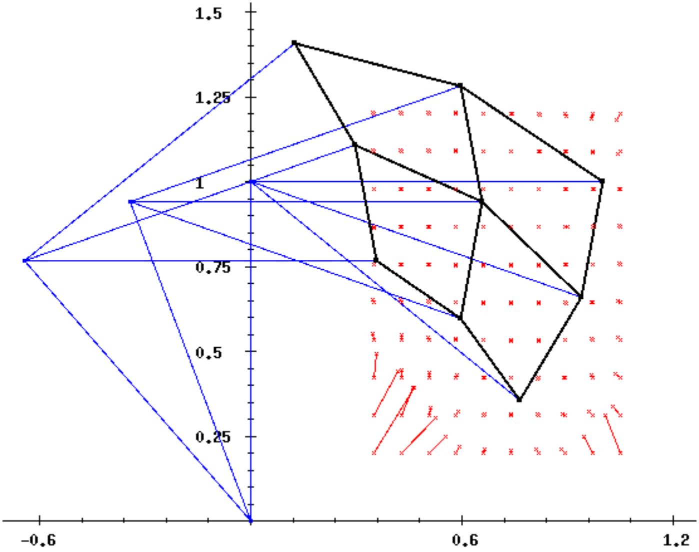
$$\mathbf{s}_{t+1} = \mathbf{s}_t - \eta \, \nabla_{\mathbf{s}} \mathbf{w}(\mathbf{s}_t) \cdot (\mathbf{w} - \mathbf{w}(\mathbf{s}_t))$$

- starting at closest discrete node

$$\mathbf{s}_0 = \arg\min_{\mathbf{c}\in \mathcal{A}} \|\mathbf{w} - \mathbf{w}(\mathbf{s})\|$$



Example: Kinematics Learning



Outline

Unsupervised Kernel Regression [Klanke 2006]

- learning of continuous non-linear manifolds

- generalizes from fixed PSOM grid
- employs unsupervised formulation of Nadaraya-Watson estimator

$$\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\} \in \mathbb{R}^{d \times N} \quad \text{observed data}$$

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \in \mathbb{R}^{q \times N} \quad \text{latent parameters}$$

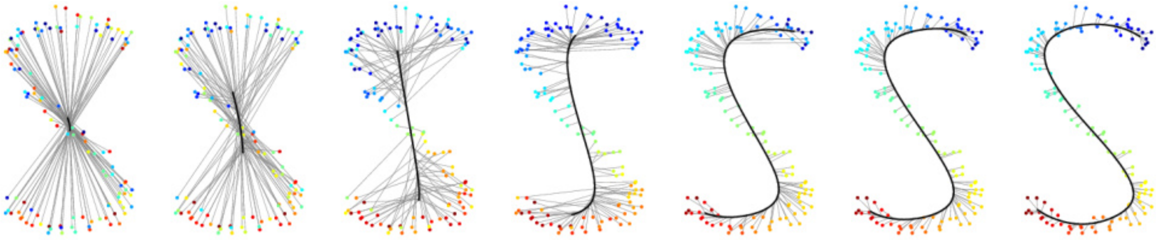
$$\mathbf{y} = \mathbf{f}(\mathbf{x}; \mathbf{X}) \quad \text{corresp. functional relationship}$$

$$\mathbf{f}(\mathbf{x}; \mathbf{X}) = \sum_i \mathbf{y_i} \frac{K(\mathbf{x} - \mathbf{x_i})}{\sum_j K(\mathbf{x} - \mathbf{x_j})} \qquad K(\mathbf{x} - \mathbf{x_i}) = \mathcal{N}(0, \Sigma) \quad - \quad \text{Gaussian kernel}$$

UKR Learning

Minimize reconstruction error

$$R(\mathbf{X}) = \frac{1}{N}\sum_{m}\|\mathbf{y}_m - f(\mathbf{x}_m;\mathbf{X})\|^2$$

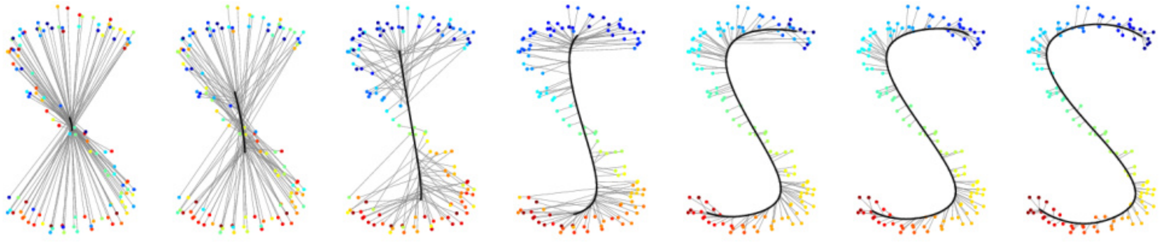


init $t = 3$ $t = 7$ $t = 10$ $t = 14$ $t = 20$ $t = 50$

UKR Learning

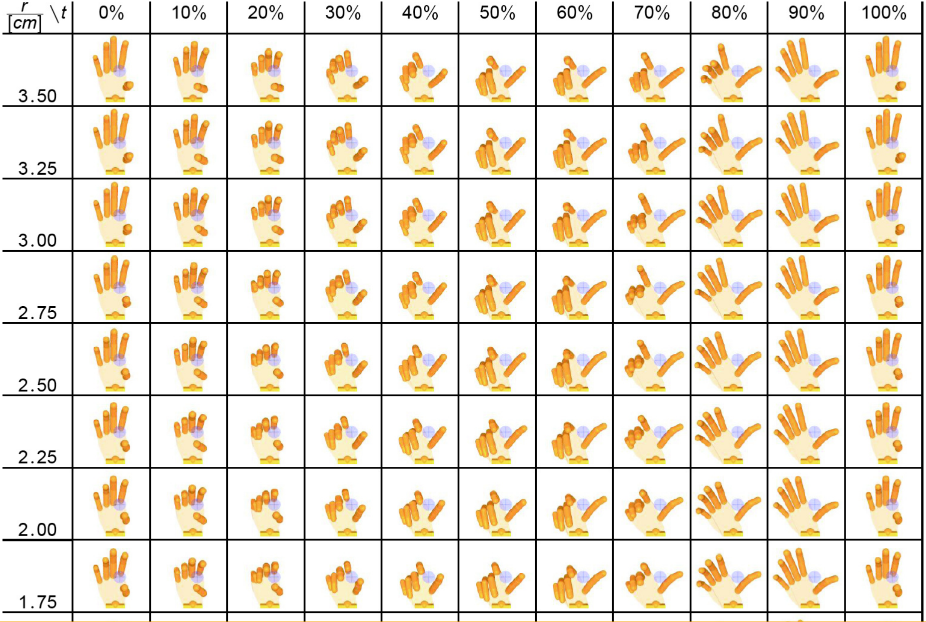
Minimize reconstruction error

$$R(\mathbf{X}) = \frac{1}{N} \sum_{m} \| \mathbf{y}_m - f_{-m}(\mathbf{x}_m; \mathbf{X}) \|^2 \qquad \text{(leave-one-out CV)}$$

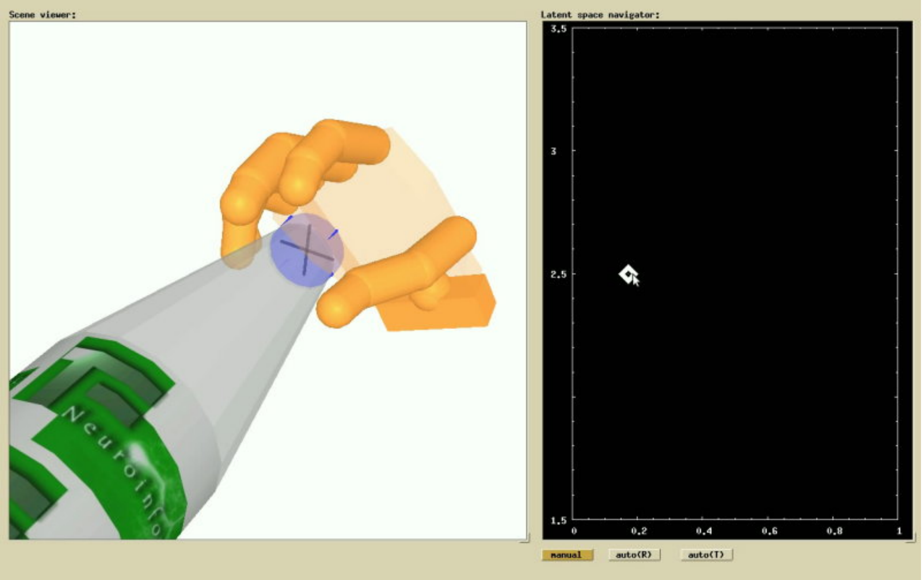


init | $t = 3$ | $t = 7$ | $t = 10$ | $t = 14$ | $t = 20$ | $t = 50$

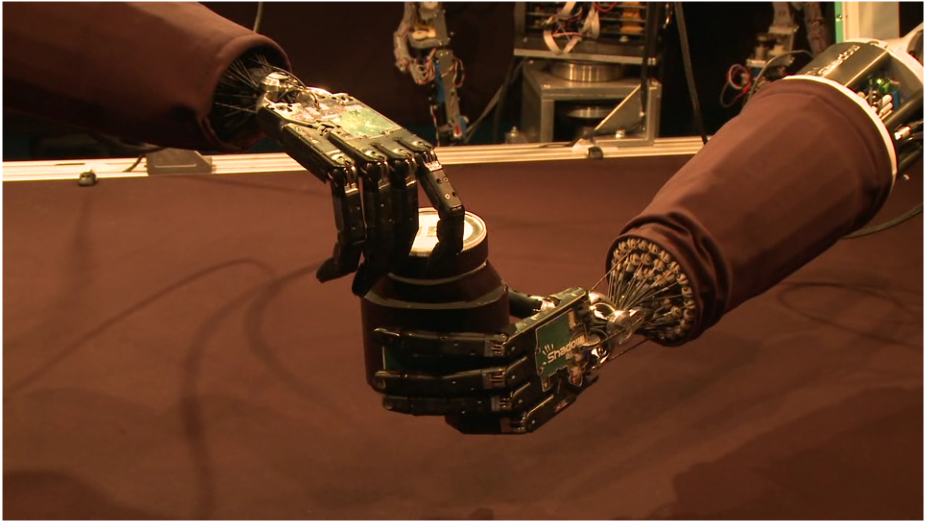
Learned Manipulation Manifold



Manipulation Manifold in action



Shadow Robot Hand Opening a Bottle Cap [Humanoids 2011]

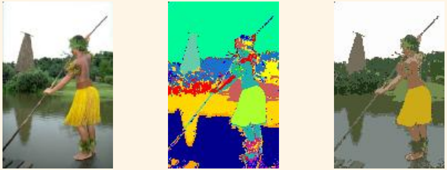


Outline

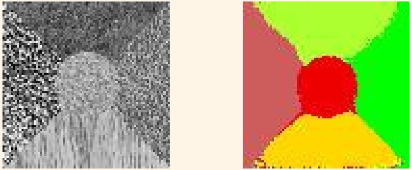
Outline

Perceptual Grouping

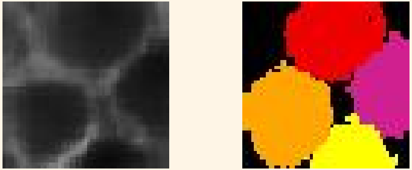
Colour Segmentation



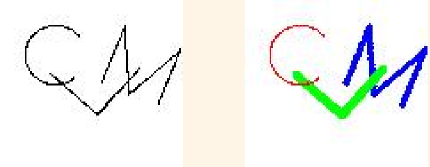
Texture Segmentation



Segmenting Cell Images



Contour Grouping

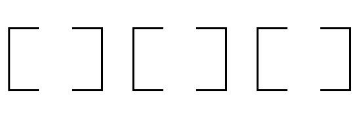


Gestalt Laws [Wertheimer 1923]

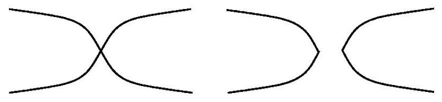
proximity

similarity

closure

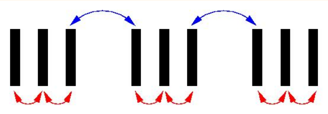


continuation

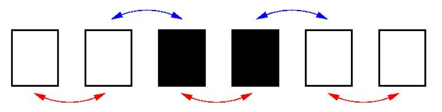


Gestalt Laws [Wertheimer 1923]

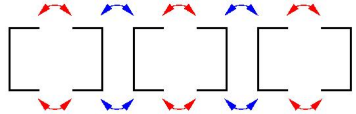
proximity



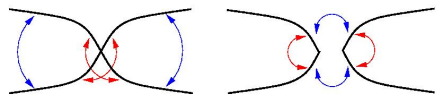
similarity



closure



continuation

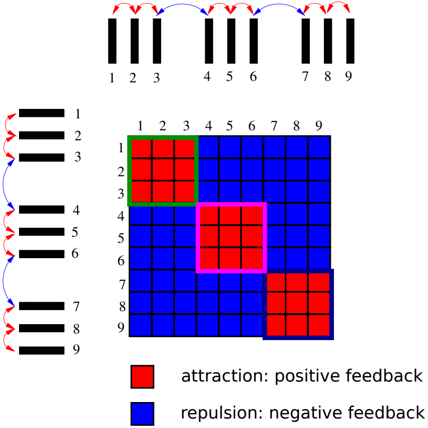


attraction

repulsion

Interaction Matrix

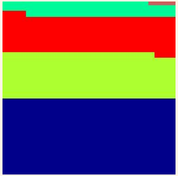
- compatibility of feature pairs induces interaction matrix



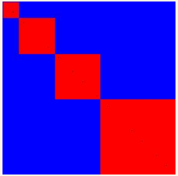
Interaction Matrix

- compatibility of feature pairs induces interaction matrix
- block structure defines groups

target segmentation



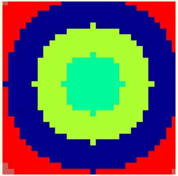
interaction matrix



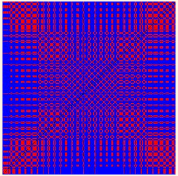
Interaction Matrix

- compatibility of feature pairs induces interaction matrix
- block structure defines groups
- real features unsorted

target segmentation



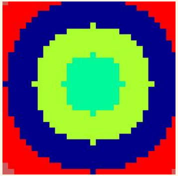
interaction matrix



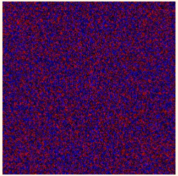
Interaction Matrix

- compatibility of feature pairs induces interaction matrix
- block structure defines groups
- real features unsorted
- and noisy

target segmentation



interaction matrix

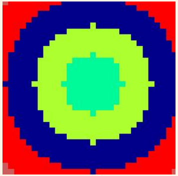


Interaction Matrix

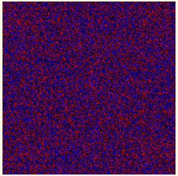
- compatibility of feature pairs induces interaction matrix
- block structure defines groups
- real features unsorted
- and noisy

⇒ robust grouping *dynamics*

target segmentation



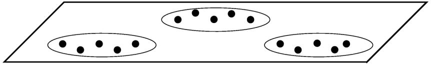
interaction matrix



Outline

Competitive Layer Model (CLM) [Ritter 1990]

1. input: set of features



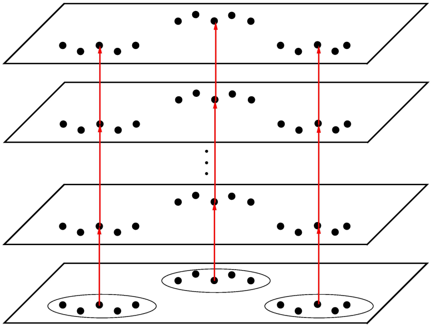
input: feature vector $\mathbf{m}_r$

examples:

- position: $\mathbf{m}_r = (x_r, y_r)^T$
- oriented line features: $\mathbf{m}_r = (x_r, y_r, \phi_r)^T$

Competitive Layer Model (CLM) [Ritter 1990]

2. layered architecture of neurons

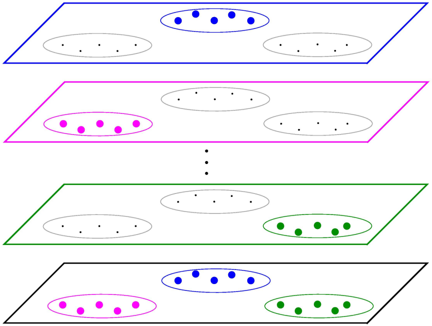


neurons $x_{r\alpha}$

- L layers $\alpha = 1, \ldots, L$
- columns r of neurons
- activation: linear threshold
  $\sigma(x) = \max(0, x)$

Competitive Layer Model (CLM) [Ritter 1990]

3. goal: grouping as activation within layers

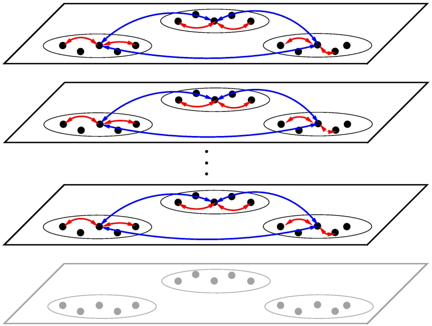


label $\hat{\alpha}(r)$

- • : $\hat{\alpha}(r) = 1$
- • : $\hat{\alpha}(r) = 2$
- • : $\hat{\alpha}(r) = L$

Competitive Layer Model (CLM) [Ritter 1990]

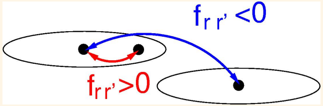
4. lateral compatibility interaction $f_{rr'}$ induces grouping



lateral interaction $f_{rr'}$

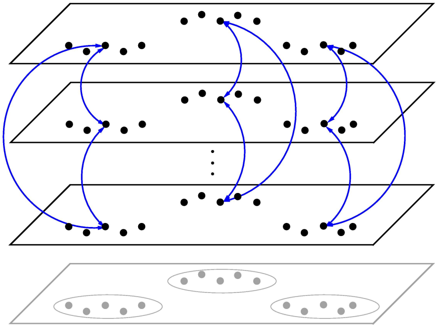
$f_{rr'}$ – compatibility of feature pair

$\mathbf{m}_r - \mathbf{m}_{r'}$

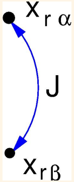


Competitive Layer Model (CLM) [Ritter 1990]

5. vertical competition: pushes groups to layers

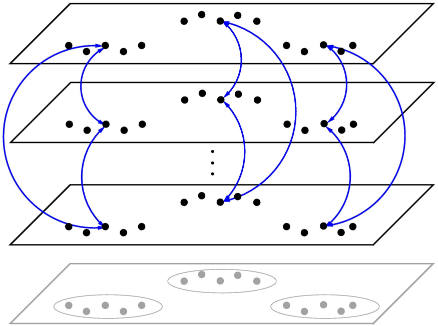


vertical inhibition J



Competitive Layer Model (CLM) [Ritter 1990]

6. column-wise stimulation with feature-dependent bias

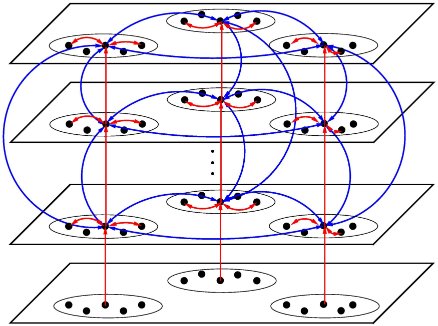


bias h_r

- overall activity per column
- significance of feature $\mathbf{m}_r$

Competitive Layer Model (CLM) [Ritter 1990]

6. simulation of recurrent dynamics



overall dynamics

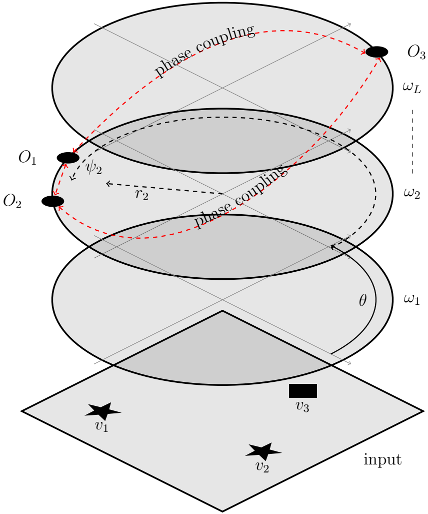
$$\dot{x}_{r\alpha} = -x_{r\alpha} + \sigma\Big(J(h_r - \sum_{\beta} x_{r\beta}) + \sum_{r'} f_{rr'} x_{r'\alpha}\Big)$$

Outline

Kuramoto Oscillator Network [Meier 2013]

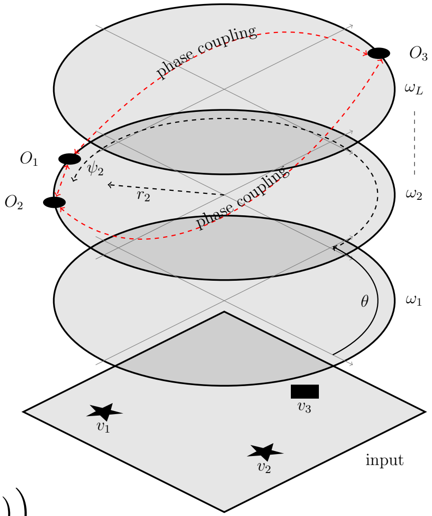
- more efficient grouping dynamics
- based on coupled oscillators
  - phase θ and frequency ω
  - phase coupling by $f_{rr'}$

$$\dot{\theta}_r = \omega_r + \frac{K}{N} \sum_{r'=1}^{N} f_{rr'} \cdot \sin(\theta_{r'} - \theta_r)$$



Kuramoto Oscillator Network [Meier 2013]

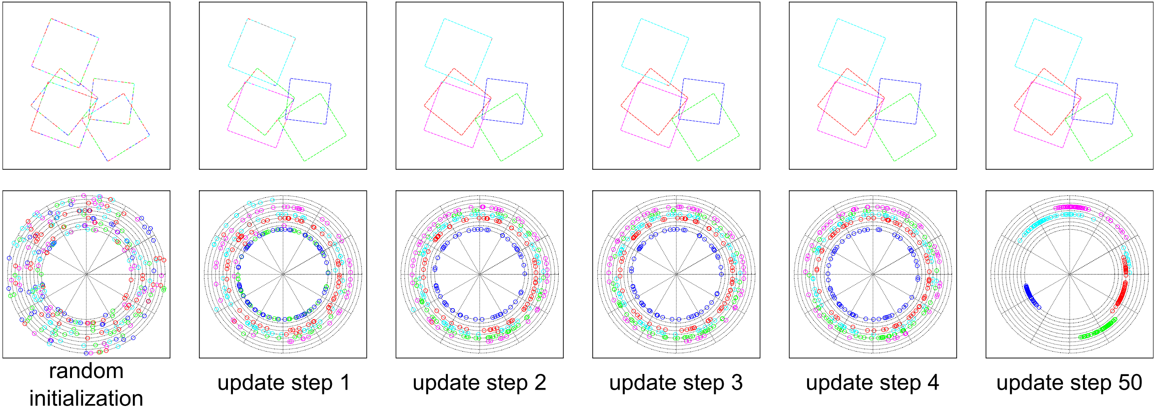
- more efficient grouping dynamics
- based on coupled oscillators
  - phase θ and frequency ω
  - phase coupling by $f_{rr'}$
- phase similarity determines frequency grouping
- similar features share same frequency



$$\dot{\theta}_r = \omega_r + \frac{K}{N}\sum_{r'=1}^{N} f_{rr'} \cdot \sin(\theta_{r'} - \theta_r)$$

$$\omega_r = \omega_0 \cdot \operatorname*{argmax}_{\alpha}\left(\sum_{r' \in \mathcal{N}(\alpha)} f_{rr'} \cdot \frac{1}{2}\left(\cos(\theta_{r'} - \theta_r) + 1\right)\right)$$

Example: Contour Grouping



Outline

Evaluation: CLM vs. Oscillator Network

- 10 groups á 100 features
- 100 layers resp. 100 frequencies (only 10 needed)
- different amount of noise in interaction matrices

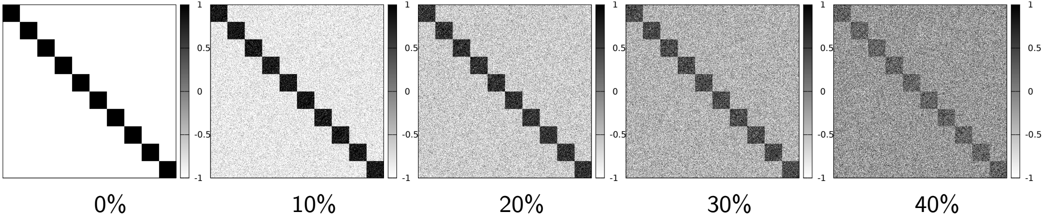
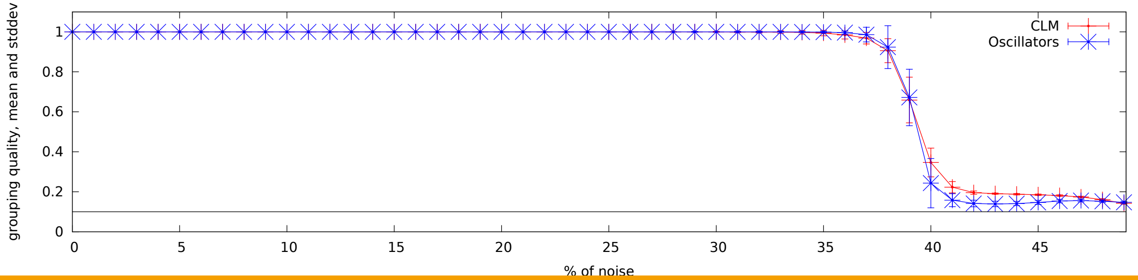


Figure : Interaction matrices with different amounts of inverted interaction values. Black pixel represents attraction, white is repelling.

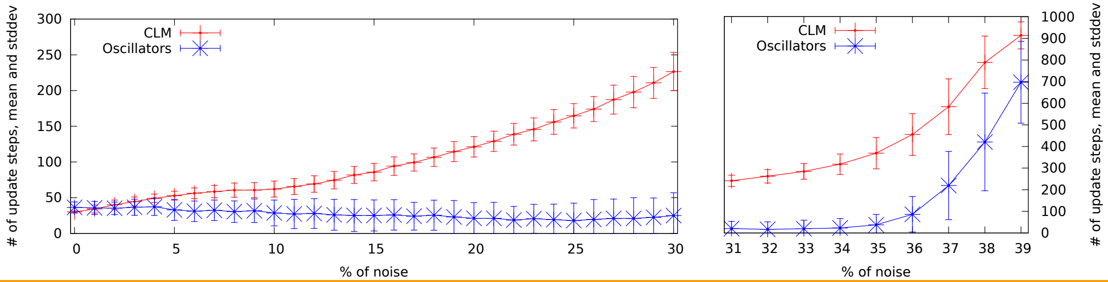
Evaluation Results

- similar grouping quality



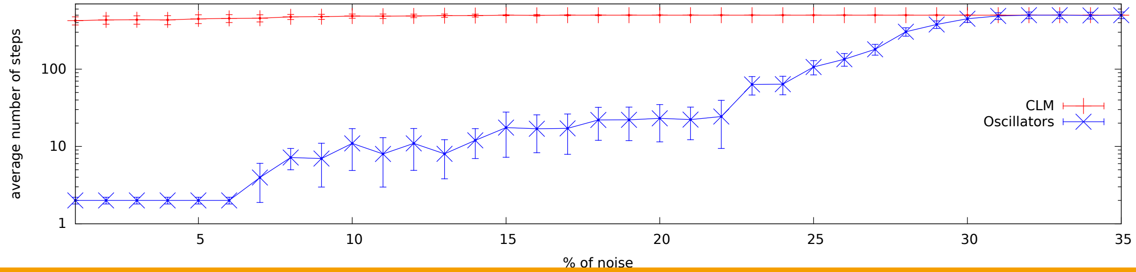
Evaluation Results

- similar grouping quality
- reduced computational complexity
- increased convergence speed



Evaluation Results

- similar grouping quality
- reduced computational complexity
- increased convergence speed
- faster recovery on changing interaction matrix
  (splitting from 10 to 20 groups after initial convergence)



Outline

How to integrate learning?

- recurrent dynamics robustly creates grouping
- dynamics determined by interaction matrix $f_{rr'}$
- How to learn compatibilities?

How to integrate learning?

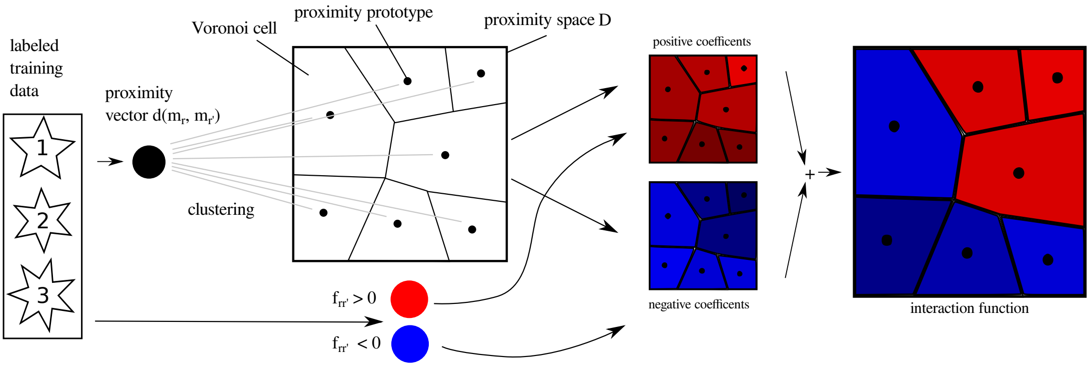
- recurrent dynamics robustly creates grouping
- dynamics determined by interaction matrix $f_{rr'}$
- How to learn compatibilities?

- CLM dynamics extremly robust wrt. noise in interaction

$\rightarrow$ only learn *coarse* interaction matrix

Learning Architecture [Weng 2006]

- Compute more general *distance function* on feature pairs
- Learn distance prototypes from labeled samples (VQ)
- Exploit labels to count pos./neg. weighted prototypes
- Overview:



Outline

Imitation Learning

- learn from observations
  - How to observe actions?
  - Which elements are important? What to learn?
  - How to represent observed actions?
- improving + adapting motion
  - autonomous exploration
  - Reinforcement Learning

Imitation Learning

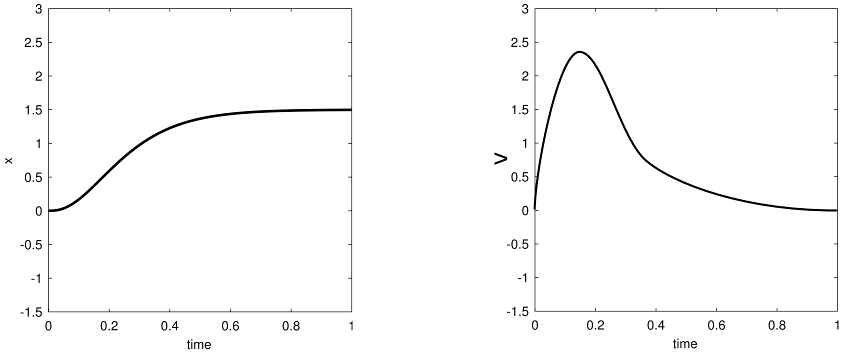
- learn from observations
  - How to observe actions?
  - Which elements are important? What to learn?
  - How to represent observed actions?
    Dynamic Movement Primitives
- improving + adapting motion
  - autonomous exploration
  - Reinforcement Learning
    Policy Improvement with Path Integrals (PI^2)

Outline

Dynamic Movement Primitives [Ijspeert, Nakanishi, Schaal, ICRA'02]

- spring-damper system generates basic motion towards goal

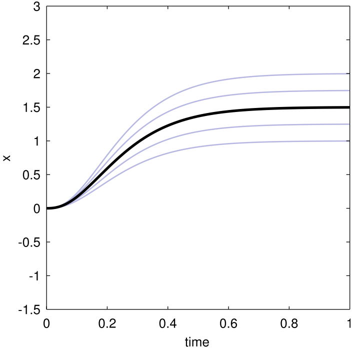
$$\tau \ddot{x}_t = k \cdot (g - x_t) - c \cdot \dot{x}_t$$



Dynamic Movement Primitives [Ijspeert, Nakanishi, Schaal, ICRA'02]

- spring-damper system generates basic motion towards goal

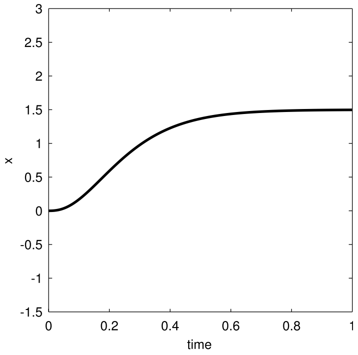
$$\tau \ddot{x}_t = k \cdot (g - x_t) - c \cdot \dot{x}_t$$



Dynamic Movement Primitives [Ijspeert, Nakanishi, Schaal, ICRA'02]

- add external force to represent complex trajectory shapes

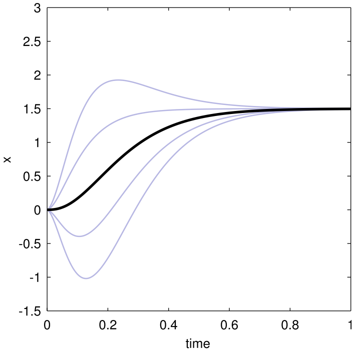
$$\tau \ddot{x}_t = k \cdot (g - x_t) - c \cdot \dot{x}_t + \mathbf{g}_t^T \boldsymbol{\theta}$$



Dynamic Movement Primitives [Ijspeert, Nakanishi, Schaal, ICRA'02]

- add external force to represent complex trajectory shapes

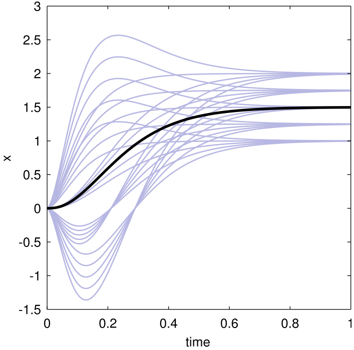
$$\tau \ddot{x}_t = k \cdot (g - x_t) - c \cdot \dot{x}_t + \mathbf{g}_t^T \boldsymbol{\theta}$$



Dynamic Movement Primitives [Ijspeert, Nakanishi, Schaal, ICRA'02]

- add external force to represent complex trajectory shapes

$$\tau \ddot{x}_t = k \cdot (g - x_t) - c \cdot \dot{x}_t + \mathbf{g}_t^T \boldsymbol{\theta}$$



External Force

$$f(s) = \mathbf{g}_t^T \boldsymbol{\theta} = \frac{\sum_i \theta_i \psi_i(s)}{\sum_i \psi_i(s)} \cdot (g - x_0) \cdot s$$

- weighted sum of basis functions ψ_i

External Force

$$f(s) = \mathbf{g}_t^T \boldsymbol{\theta} = \frac{\sum_i \theta_i \psi_i(s)}{\sum_i \psi_i(s)} \cdot (g - x_0) \cdot s$$

- weighted sum of basis functions ψ_i
- soft-max normalization

External Force

$$f(s) = \mathbf{g}_t^T \boldsymbol{\theta} = \frac{\sum_i \theta_i \psi_i(s)}{\sum_i \psi_i(s)} \cdot (g - x_0) \cdot s$$

- weighted sum of basis functions ψ_i
- soft-max normalization
- amplitude scaled by initial distance to goal

External Force

$$f(s) = \mathbf{g}_t^T \boldsymbol{\theta} = \frac{\sum_i \theta_i \psi_i(s)}{\sum_i \psi_i(s)} \cdot (g - x_0) \cdot s$$

- weighted sum of basis functions ψ_i
- soft-max normalization
- amplitude scaled by initial distance to goal
- influence weighted by *canonical time* $s \rightarrow 0$

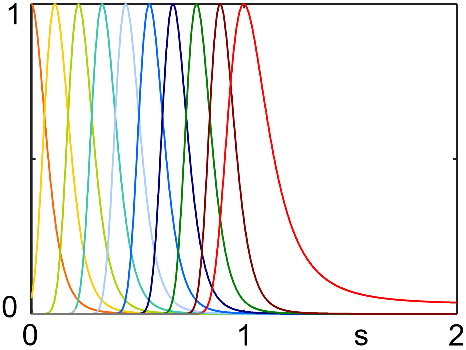
External Force

$$f(s) = \mathbf{g}_t^T \boldsymbol{\theta} = \frac{\sum_i \theta_i \psi_i(s)}{\sum_i \psi_i(s)} \cdot (g - x_0) \cdot s$$

- weighted sum of basis functions ψ_i
- soft-max normalization
- amplitude scaled by initial distance to goal
- influence weighted by *canonical time* $s \to 0$
- Gaussian basis functions ψ_i

$$\psi_i(s) = \exp(-h_i \, (s - c_i)^2)$$

- c_i logarithmically distributed in $[0 \dots 1]$



[Schaal et al, Brain Research, 2007]

Canonical System

$$f(s) = \mathbf{g}_t^T \boldsymbol{\theta} = \frac{\sum_i \theta_i \psi_i(s)}{\sum_i \psi_i(s)} \cdot (g - x_0) \cdot s$$

- decouple external force from spring-damper evolution
- new phase / time variable s

$$\tau \dot{s} = -\alpha \cdot s$$

- s initially set to 1 ...
- ... exponentially converges to 0

Canonical System

$$f(s) = \mathbf{g}_t^T \boldsymbol{\theta} = \frac{\sum_i \theta_i \psi_i(s)}{\sum_i \psi_i(s)} \cdot (g - x_0) \cdot s$$

- decouple external force from spring-damper evolution
- new phase / time variable s

$$\tau \dot{s} = -\alpha \cdot s \cdot \frac{1}{1 + \alpha_c \cdot (x_{actual} - x_{expected})^2}$$

- s initially set to 1 ...
- . . . exponentially converges to 0
- pause influence of force on perturbations

Properties

$$\tau \ddot{x}_t = k \cdot (g - x_t) - c \cdot \dot{x}_t + \mathbf{g}_t^T \boldsymbol{\theta}$$

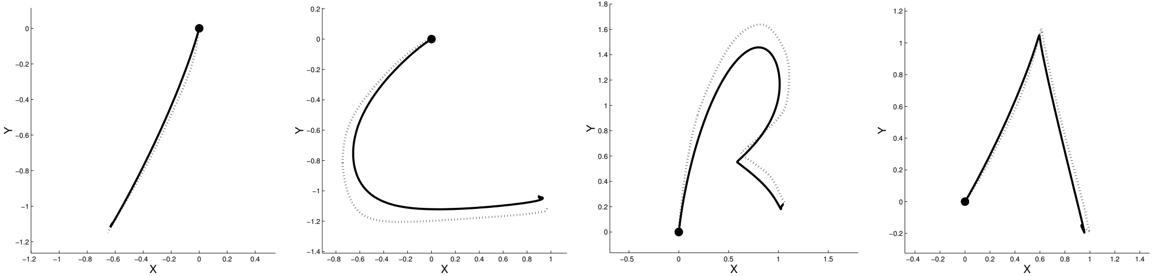
$$\tau \dot{s} = -\alpha \cdot s$$

$$f(s) = \mathbf{g}_t^T \boldsymbol{\theta} = \frac{\sum_i \theta_i \psi_i(s)}{\sum_i \psi_i(s)} \cdot (g - x_0) \cdot s$$

- convergence to goal g
- motions are self-similar for different goal or start points
- coupling of multiple DOF through canonical phase s
- adapt τ for temporal scaling
- robust to perturbations due to attractor dynamics
- decoupling basic goal-directed motion from task-specific trajectory “shape”
- weights θ_i can be learned with linear regression

Learning from Demonstration

- record motion $x(t), \dot{x}(t), \ddot{x}(t)$
- choose τ to match duration
- evolve canonical system $\rightarrow s(t)$
- $f_{target}(s) = \tau\ddot{x}(t) - k(g - x(t) - c\dot{x}(t))$
- minimize $E = \sum_s (f_{target}(s) - f(s))^2$ with regression



[Ijspeert, Nakanishi, Schaal, ICRA'02]

Outline

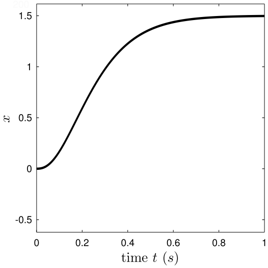
Robustifying Motions

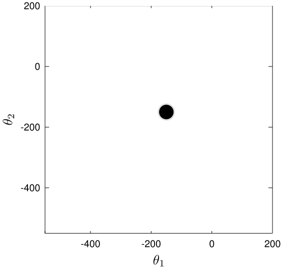
- motion from imitation learning is fragile
- robustify by self-exploration
- competing RL methods:
  - PI^2 – Policy Improvement with Path Integrals [Stefan Schaal]
  - PoWeR – Policy Learning by Weighting Exploration with the Returns [Jan Peters]

Policy Improvement with Path Integrals – PI^2 [Evangelos 2011]

- Optimize shape parameters $\boldsymbol{\theta}$ w.r.t. cost function J
- Use direct reinforcement learning
  - Exploration directly in policy parameter space $\boldsymbol{\theta}$
- Use Policy Improvement with Path Integrals – PI^2
  - Derived from principles of optimal control
  - Update rule based on cost-weighted averaging (next slide)

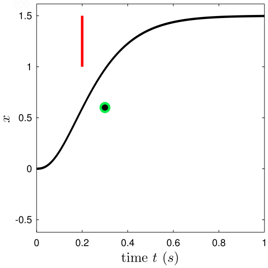
Input: DMP with initial parameters $\boldsymbol{\theta}$ [Figures from Stulp]

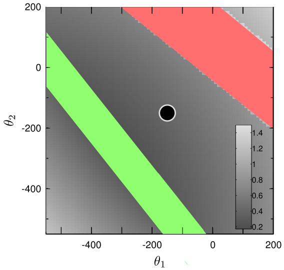




$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T \ \boldsymbol{\theta}$$

Input: DMP with initial parameters $\boldsymbol{\theta}$, cost function J [Figures from Stulp]





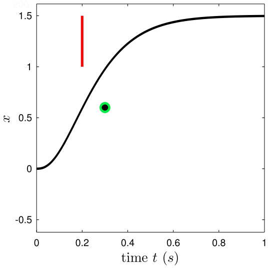
$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T \ \boldsymbol{\theta}$$

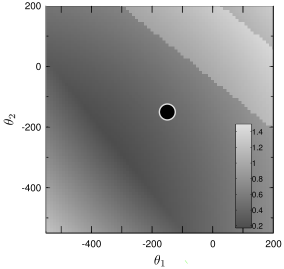
$J(\boldsymbol{\tau}_i)$

Input: DMP with initial parameters $\boldsymbol{\theta}$, cost function J [Figures from Stulp]

While (cost not converged)

Explore





$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T \ \boldsymbol{\theta}$$

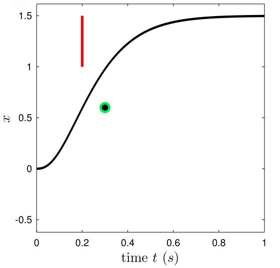
$J(\boldsymbol{\tau}_i)$

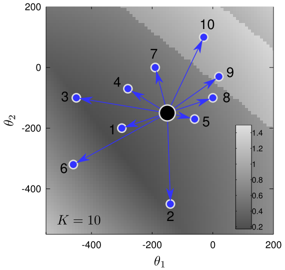
Input: DMP with initial parameters $\boldsymbol{\theta}$, cost function J [Figures from Stulp]

While (cost not converged)

Explore

sample exploration vectors





$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{i,k})$$

$$J(\boldsymbol{\tau}_i)$$

$$\boldsymbol{\epsilon}_{i,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

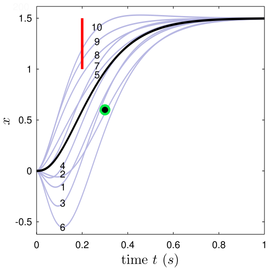
Input: DMP with initial parameters $\boldsymbol{\theta}$, cost function J [Figures from Stulp]

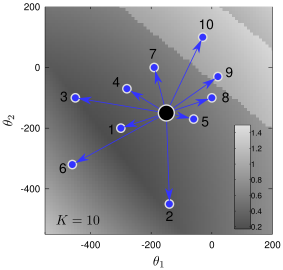
While (cost not converged)

Explore

sample exploration vectors

execute DMP





$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{i,k})$$

$$\boldsymbol{\epsilon}_{i,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

$J(\boldsymbol{\tau}_i)$

Input: DMP with initial parameters $\boldsymbol{\theta}$, cost function J [Figures from Stulp]

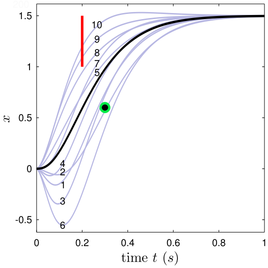
While (cost not converged)

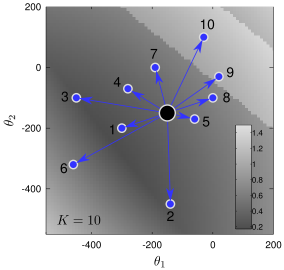
Explore

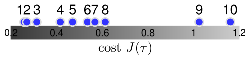
sample exploration vectors

execute DMP

determine cost







$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{i,k})$$

$$\boldsymbol{\epsilon}_{i,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

$J(\boldsymbol{\tau}_i)$

Input: DMP with initial parameters $\boldsymbol{\theta}$, cost function J [Figures from Stulp]

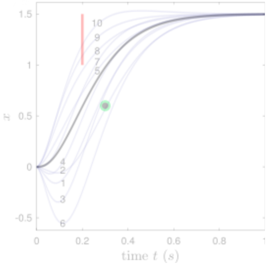
While (cost not converged)

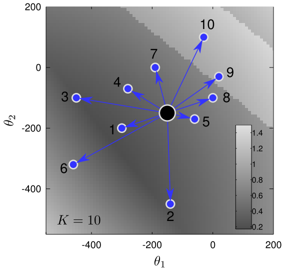
Explore

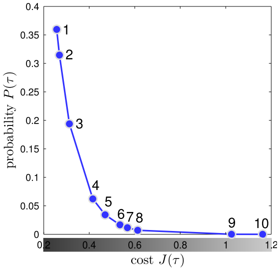
sample exploration vectors
execute DMP
determine cost

Update

weighted averaging
with Boltzmann dist.







$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{i,k})$$

$$J(\boldsymbol{\tau}_i)$$

$$\boldsymbol{\epsilon}_{i,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

$$P(\boldsymbol{\tau}_{i,k}) = \frac{\exp(-\lambda^{-1} J(\boldsymbol{\tau}_{i,k}))}{\sum_k \exp(-\lambda^{-1} J(\boldsymbol{\tau}_{i,k}))}$$

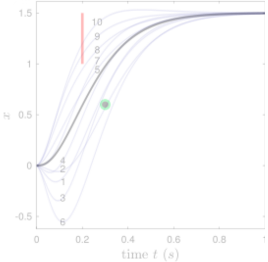
$$\Delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K} P(\boldsymbol{\tau}_{i,k})\boldsymbol{\epsilon}_{i,k}$$

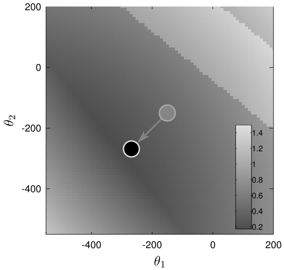
Input: DMP with initial parameters $\boldsymbol{\theta}$, cost function J [Figures from Stulp]

While (cost not converged)

Explore

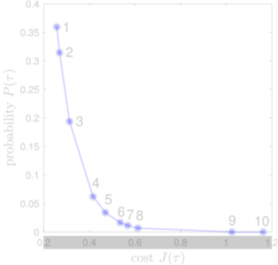
sample exploration vectors
execute DMP
determine cost





Update

weighted averaging
with Boltzmann dist.
parameter update



$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \epsilon_{i,k})$$

$$J(\boldsymbol{\tau}_i)$$

$$\epsilon_{i,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$$

$$P(\boldsymbol{\tau}_{i,k}) = \frac{\exp(-\lambda^{-1} J(\boldsymbol{\tau}_{i,k}))}{\sum_k \exp(-\lambda^{-1} J(\boldsymbol{\tau}_{i,k}))}$$

$$\Delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K} P(\boldsymbol{\tau}_{i,k})\epsilon_{i,k}$$

Input: DMP with initial parameters $\boldsymbol{\theta}$, cost function J [Figures from Stulp]

While (cost not converged)

Explore

sample exploration vectors

execute DMP

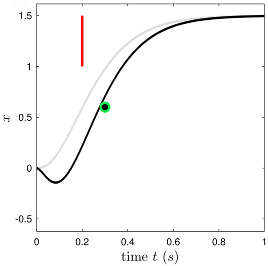
determine cost

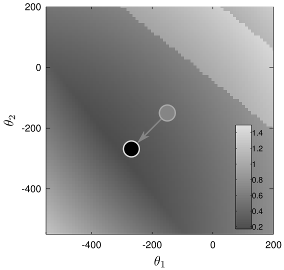
Update

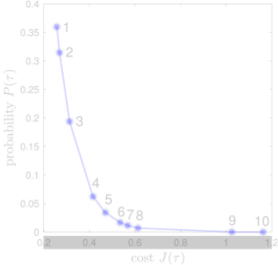
weighted averaging

with Boltzmann dist.

parameter update







$$\tau \ddot{x}_t = k(g - x_t) - c\dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{i,k})$$

$$J(\boldsymbol{\tau}_i)$$

$$\boldsymbol{\epsilon}_{i,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$$

$$P(\boldsymbol{\tau}_{i,k}) = \frac{\exp(-\lambda^{-1} J(\boldsymbol{\tau}_{i,k}))}{\sum_k \exp(-\lambda^{-1} J(\boldsymbol{\tau}_{i,k}))}$$

$$\Delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K} P(\boldsymbol{\tau}_{i,k}) \boldsymbol{\epsilon}_{i,k}$$