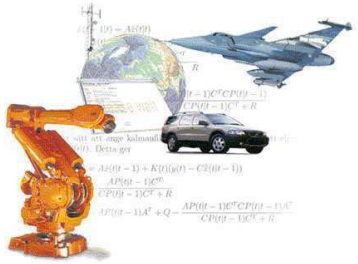
Machine Learning, Lecture 4
Neural Networks (NN), Introduction to Kernel Methods and Gaussian Processes



Thomas Schön

Division of Automatic Control
Linköping University
Linköping, Sweden.

Email: schon@isy.liu.se,
Phone: 013 - 281373,
www.control.isy.liu.se/~schon/

Outline Lecture 4

1. Summary of Lecture 3
2. Generalize the linear model to a nonlinear function expansion
3. Training of neural networks
4. Successful examples of NN in real life examples
   - Face detection
   - System identification
   - Handwritten digit classification
5. Introducing kernel methods
6. Different ways of constructing kernels
7. Gaussian Processes

Summary of Lecture 3 (I/II)

The Expectation Maximization (EM) algorithm computes maximum likelihood estimates of unknown parameters in probabilistic models involving latent variables.

Expectation (E) step: Compute

$$\mathcal{Q}(\theta, \theta_i) = \mathbf{E}_{\theta_i}\left\{\ln p_\theta(Z, X) \mid X\right\} = \int \ln p_\theta(Z, X) p_{\theta_i}(Z \mid X) dZ.$$

Maximization (M) step: Compute

$$\theta_{i+1} = \arg\max_\theta \mathcal{Q}(\theta, \theta_i).$$

Summary of Lecture 3 (II/II)

We constructed a Gaussian mixture density using latent variables z (multinomial)

$$p(z) = \prod_{k=1}^{K} \pi_k^{z_k}, \qquad p(x \mid z) = \prod_{k=1}^{K} \mathcal{N}(x \mid \mu_k, \Sigma_k)^{z_k}$$

This allowed us to derive an EM algorithm for estimating a Gaussian mixture.

Clustering is the problem of grouping N points $\{x_i\}_{i=1}^N$ into K clusters, where members of each cluster are “similar”.

The K-means algorithm tries to minimize $\sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|^2$. We can show that the K-means algorithm is a (deterministic) special case of the EM algorithm.

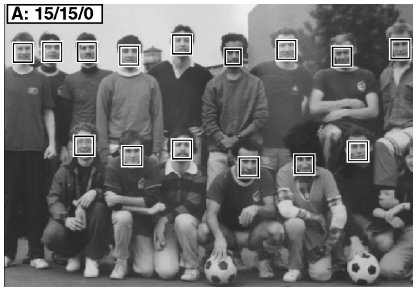
Three Examples of Neural Networks in Use

1. Face detection
2. System identification
3. Handwritten digit classification

These examples will provide a glimpse into a few real life applications of models based in nonlinear function expansions (i.e., neural networks) both for regression and classification problems. We provide references for a more complete treatment.

NN Example 1 - Face Detection (I/IV)

Goal: Detect upright frontal faces in images using a neural network.



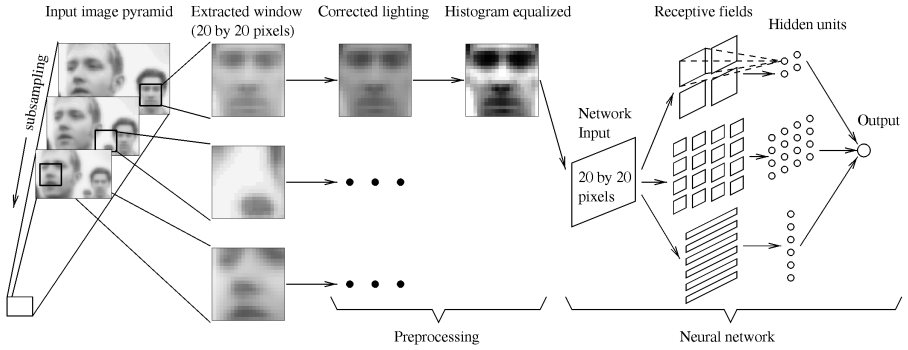
This example is borrowed from

Rowley, H.A. and Baluja, S. and Kanade, T. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998.

`http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=655647&tag=1`

NN Example 1 - Face Detection (II/IV)

System overview:



NN Example 1 - Face Detection (III/IV)

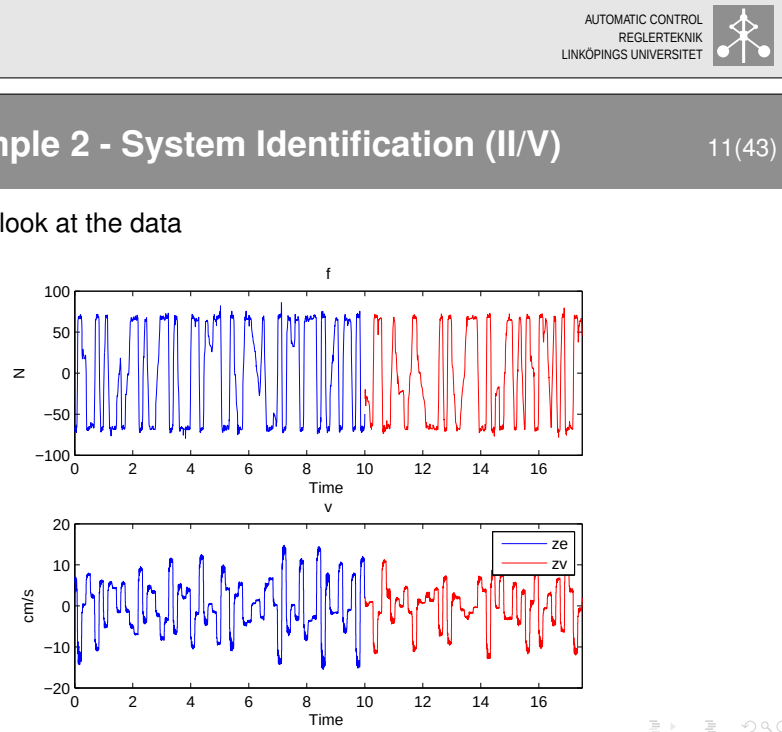
Example of transformations applied to the input data (random rotation, translation and scalings by small amounts) in order to introduce invariance in the detector (Section 5.5.3).

NN Example 1 - Face Detection (IV/IV)

Lessons learned from this example:

- Transformation of inputs to always have similar type of input data.
- Include transformed versions in the training data (to enhance invariance).
- Adapt the model to the knowledge you have about the problem (nearby pixels are more strongly correlated than more distant pixels).

See Section 5.5 for a more general discussion.



NN Example 2 - System Identification (I/V)

Neural networks are one of the standard models used in nonlinear system identification.

Problem background: The task here is to identify a dynamical model of a Magnetorheological (MR) fluid damper. The MR fluid (typically some kind of oil) will greatly increases its so called apparent viscosity when the fluid is subjected to a magnetic field.

MR fluid dampers are semi-active control devices which are used to reduce vibrations.

Input signal: velocity $v(t)$ [cm/s] of the damper
Output signal: Damping force $f(t)$ [N].

NN Example 2 - System Identification (II/V)

Have a look at the data

NN Example 2 - System Identification (III/V)

As usual, we try simple things first, that is a linear model. The best linear model turns out to be an output error (OE) model which gives 51% fit on validation data.

```
LinMod2 = oe(ze, [4 2 1]);  % OE model  y = B/F u + e
```

Try a neural network sigmoidal neural network using 10 hidden units

```
Options = {'MaxIter',50, 'SearchMethod', 'LM'};
Narx1 = nlarx(ze, [2 4 1], 'sigmoidnet',Options{:})
```

This model already gives a 72% fit on validation data.

```
compare(zv, Narx1);
```

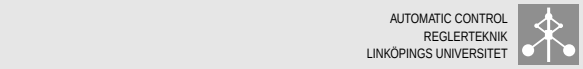
NN Example 2 - System Identification (IV/V)

Using 12 hidden units and only making use of some of the regressors,

```
Sig   = sigmoidnet('NumberOfUnits',12); % create SIGMOIDNET object
Narx5 = nlarx(ze, [2 3 1], Sig, 'NonlinearRegressors', [1 3 4],...
        Options{:});
```

the performance can be increased to a 85% fit on validation data.

Of course, this model need further validation, but the improvement from 51% fit for the best linear model is substantial.



NN Example 2 - System Identification (V/V)

This example if borrowed from

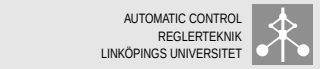
Wang, J., Sano, A., Chen, T. and Huang. B. Identification of Hammerstein systems without explicit parameterization of nonlinearity. *International Journal of Control*, 82(5):937–952, May 2009.

and it is used as one example in illustrating Lennart's toolbox,

`http://www.mathworks.com/products/sysid/demos.html?file=/products/demos/shipping/ident/idnlbbdemo_damper.html`

More about the use of neural networks in System Identification can be found in,

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P-Y., Hjalmarsson, H. and Juditsky, A. Nonlinear black-box modeling in system identification: a unified overview, *Automatica*, 31(12):1691–1724, December 1995.

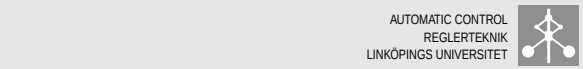


NN Example 3 - Digit Classification (I/IV)

You have tried solving this problem using linear methods before. Let us see what can be done if we generalize to nonlinear function expansions (neural networks) instead.

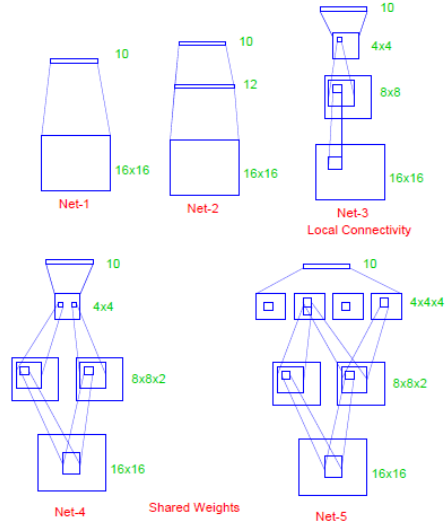
Let us now investigate 4 nonlinear models and one linear model for solving the same task,

- **Net-1:** No hidden layer (equivalent to logistic regression).
- **Net-2:** One hidden layer, 12 hidden units fully connected.
- **Net-3:** Two hidden layers locally connected.
- **Net-4:** Two hidden layers, locally connected with weight sharing.
- **Net-5:** Two hidden layers, locally connected with two levels of weight sharing.



NN Example 3 - Digit Classification (II/IV)

Local connectivity (Net3-Net5) means that each hidden unit is connected only to a small number of units in the layer before. It makes use of the key property that nearby pixels are more strongly correlated than distant pixels.



NN Example 3 - Digit Classification (III/IV)

Network Architecture	Links	Weights	Correct
Net-1: 0 hidden layer	2570	2570	80.0%
Net-2: 1 hidden layer network	3214	3214	87.0%
Net-3: 2 hidden layer, locally connected	1226	1226	88.5%
Net-4: 2 hidden layer, constrained network	2266	1132	94.0%
Net-5: 2 hidden layer, constrained network	5194	1060	98.4%

Net-4 and Net-5 are referred to as convolutional networks.

This example clearly illustrates that knowledge about the problem at hand can be very useful in order to improve the performance!

NN Example 3 - Digit Classification (IV/IV)

This example if borrowed from Section 11.7 in HTF, who in turn borrowed it from,

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E. Hubbard, W. and Jackel, L. D. Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Computation*, 1(4):541-551, 1989.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. Gradient-Based Learning Applied to Document Recognition, *Proceedings of the IEEE*, 86(11):2278–2324, November 1998,

There is plenty of software support for neural networks, for example MATLAB's neural network toolbox (for general problems) and the system identification toolbox (for sys. id.).

Kernel Representation of ℓ_2-reg. LS (I/II)

Inserting the solution $\widehat{w} = \Phi^T \widehat{a} = \Phi^T (K + \lambda I)^{-1} T$ into $y(x, w)$ provides the following prediction for a new input x

$$
\begin{aligned}
y(x, \widehat{w}) = \widehat{w}^T \phi(x) = \widehat{a}^T \Phi \phi(x) &= \left(\left((K + \lambda I)^{-1} T \right)^T \Phi \phi(x) \right)^T \\
&= \phi(x)^T \Phi^T (K + \lambda I)^{-1} T \\
&= \phi(x)^T \begin{pmatrix} \phi(x_1) & \phi(x_2) & \cdots & \phi(x_N) \end{pmatrix} (K + \lambda I)^{-1} T \\
&= \begin{pmatrix} k(x, x_1) & k(x, x_2) & \cdots & k(x, x_N) \end{pmatrix} (K + \lambda I)^{-1} T,
\end{aligned}
$$

where we have made use of the definition of a kernel function

$$
k(x, z) \triangleq \phi(x)^T \phi(z)
$$

Hence, the solution to the ℓ_2-regularized least squares problem is expressed in terms of the kernel function $k(x, z)$.

Kernel Representation of ℓ_2-reg. LS (II/II)

Note (again) that the prediction at x is given by a linear combination of the target values from the training set (expensive).

Furthermore, we are required to invert an $N \times N$ matrix (compared to an $M \times M$ matrix in the original formulation), where typically $N \gg M$.

Relevant question, So what is the point?

The fact that it is expressed only using the kernel function $k(x, z)$ implies that we can work entirely using kernels and avoid introducing basis functions $\phi(x)$. This in turn allows us to implicitly use basis functions of high, even infinite, dimensions ($M \to \infty$).

We will provide examples clearly illustrating the gain here.

Kernel Representation of ℓ_2-reg. LS (II/II)

Note (again) that the prediction at x is given by a linear combination of the target values from the training set (expensive).

Furthermore, we are required to invert an $N \times N$ matrix (compared to an $M \times M$ matrix in the original formulation), where typically $N \gg M$.

Relevant question, So what is the point?

The fact that it is expressed only using the kernel function $k(x,z)$ implies that we can work entirely using kernels and avoid introducing basis functions $\phi(x)$. This in turn allows us to implicitly use basis functions of high, even infinite, dimensions ($M \to \infty$).

We will provide examples clearly illustrating the gain here.

Constructing Kernels

1. Chose a feature mapping $\phi(x)$ and then use this to find the corresponding kernel,

$$k(x,z) = \phi(x)^T\phi(z) = \sum_{i=1}^{M} \phi_i(x)\phi_i(z)$$

2. Chose a kernel function directly. In this case it is important to verify that it is in fact a kernel. (we will see two examples of this)
   A function $k(x,z)$ is a kernel iff the Gram matrix K is positive semi-definite for all possible inputs.
3. Form new kernels from simpler kernels.
4. Start from probabilistic generative models.

Techniques for Constructing new Kernels

Given valid kernels $k_1(x,z)$ and $k_2(x,z)$, the following are also valid kernels

$$\begin{aligned}
k(x,z) &= ck_1(x,z), & k(x,z) &= f(x)k_1(x,z)f(z),\\
k(x,z) &= q(k_1(x,z)), & k(x,z) &= \exp(k_1(x,z)),\\
k(x,z) &= k_1(x,z) + k_2(x,z), & k(x,z) &= k_1(x,z)k_2(x,z),\\
k(x,z) &= k_3(\phi(x),\phi(z)), & k(x,z) &= x^TAx,
\end{aligned}$$

where $c > 0$ is a constant, f is a function, q is a polynomial with nonnegative coefficients, $\phi(x)$ is a function from x to $\mathbb{R}^M$, k_3 is a valid kernel in $\mathbb{R}^M$ and $A \succeq 0$.

Ex. Polynomial Kernel

Let us investigate if the polynomial kernel

$$k(x,z) = (x^Tz + c)^n, c > 0$$

is a kernel for the special case $n = 2$ and a 2D input space $x = (x_1, x_2)^T$,

$$\begin{aligned}
k(x,z) &= (x^Tz)^2 = (x_1z_1 + x_2z_2 + c)^2\\
&= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2 + 2cx_1z_1 + 2cx_2z_2 + c^2\\
&= \phi(x)^T\phi(z),
\end{aligned}$$

where

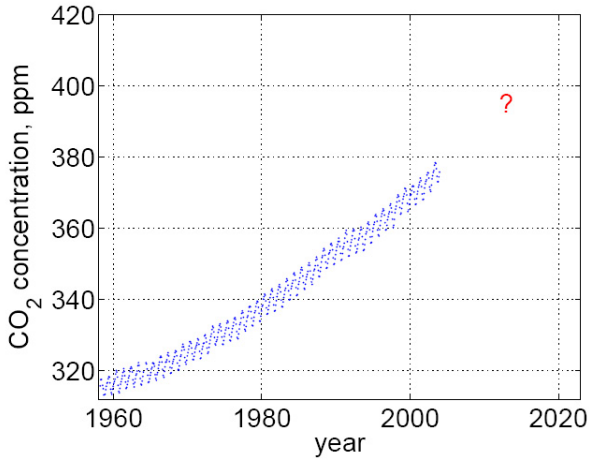
$$\phi(x) = \begin{pmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 & \sqrt{2c}x_1 & \sqrt{2c}x_2 & c \end{pmatrix}^T$$

Hence, it contains all possible terms (constant, linear and quadratic) up to order 2.

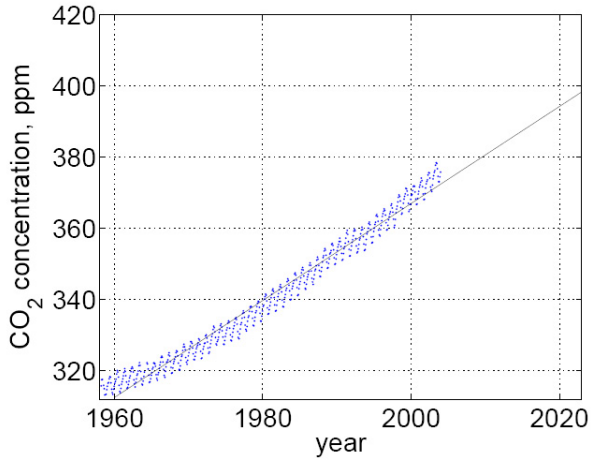
Outline Lecture 4

1. Summary of Lecture 3
2. Generalize the linear model to a nonlinear function expansion
3. Training of neural networks
4. Successful examples of NN in real life examples
   - Face detection
   - System identification
   - Handwritten digit classification
5. Introducing kernel methods
6. Different ways of constructing kernels
7. Gaussian Processes

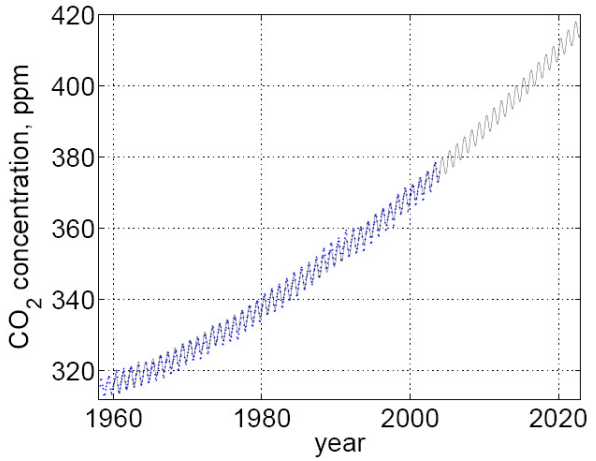
Introductory Example (I/IV)



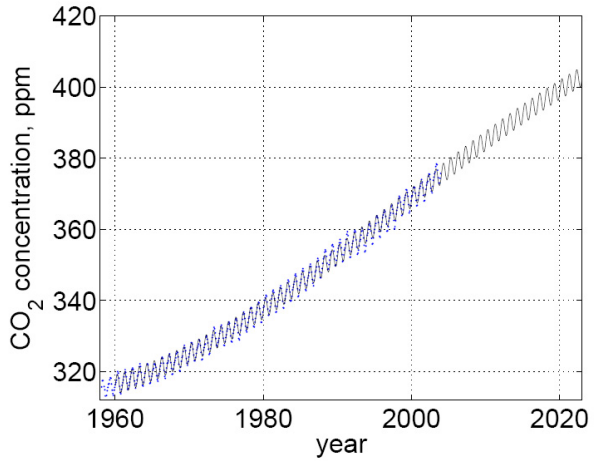
Introductory Example (II/IV)



Introductory Example (III/IV)



Introductory Example (IV/IV)



Stochastic Processes

Definition: A stochastic process can be defined as a family of random variables $\{y(x),\, x \in \mathcal{X}\}$.

Property: For a fixed $x \in \mathcal{X}$, $y(x)$ is a random variable.

Examples: Wiener process, Chinese restaurant process, Dirichlet processes, Poisson process, Gaussian process, Markov process.

Åström K. J. (2006). Introduction to Stochastic Control Theory. Dover Publications, Inc., NY, USA.

A Gaussian Process (GP)

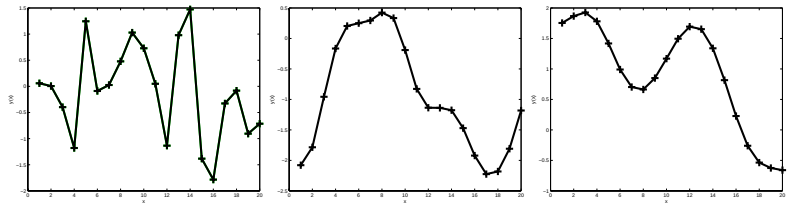
Definition (Rasmussen & Williams, 2006):

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

What does this mean?

Samples from a GP

Let $y(x)$ be a Gaussian process with mean function $m = 0$ and a covariance function $\mathrm{Cov}(y(x), y(x')) = k(x, x') = e^{-(x-x')^2/l}$. Let $x = 1 : 20$. Samples from the GP is shown below.

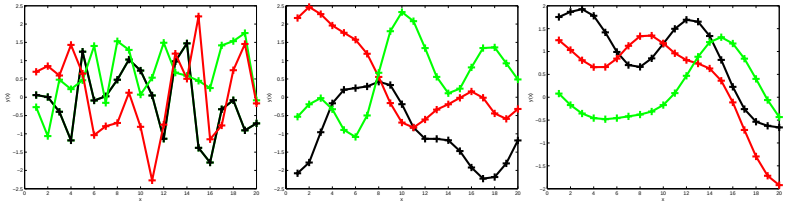


$l = 1$ $l = 10$ $l = 20$

Note that each sample is a function. A GP can hence be seen giving a distribution over functions.

Samples from a GP

Let $y(x)$ be a Gaussian process with mean function $m = 0$ and a covariance function $\mathrm{Cov}(y(x), y(x')) = k(x, x') = e^{-(x-x')^2/l}$. Let $x = 1 : 20$. Samples from the GP is shown below.



$l = 1$ $\quad$ $l = 10$ $\quad$ $l = 20$

Note that each sample is a function. A GP can hence be seen giving a distribution over functions.

Gaussian Process Regression (GPR)

So how do we use this for regression? Assume that we are given the training data $\{(y_t, x_t)\}_{t=1}^N$ and that we seek an estimate for $y(x^*)$. If we then assume that $y(x)$ can be modeled by a GP, we have

$$\begin{bmatrix} \mathbf{y} \\ y(x^*) \end{bmatrix} \sim N\left(\begin{bmatrix} m(\mathbf{x}) \\ m(x^*) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}, \mathbf{x}) & k(\mathbf{x}, x^*) \\ k(x^*, \mathbf{x}) & k(x^*, x^*) \end{bmatrix} \right)$$

and using standard Gaussian identities we obtain the predictive (or conditional) density

$$y(x^*)|\mathbf{y} \sim N\Big(k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}\big(\mathbf{y} - m(\mathbf{x})\big) + m(x^*), \\ k(x^*, x^*) - k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}k(\mathbf{x}, x^*)\Big)$$

Let us try this.

GP Predictive Distribution

Given:

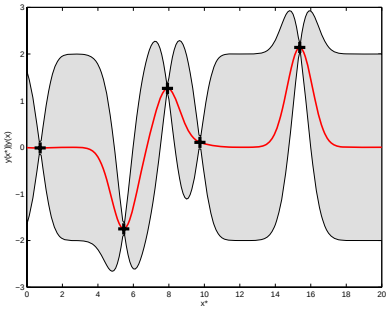
$\mathbf{x} = \begin{bmatrix} 9.8 & 15.4 & 7.9 & 5.4 & 0.7 \end{bmatrix}^T$

$\mathbf{y} = \begin{bmatrix} 0.1 & 2.1 & 1.3 & -1.7 & -0.01 \end{bmatrix}^T$

Assume: Data generated by GP with $m = 0$, $k(x, x') = e^{-(x-x')^2/l}$.

Predictive GP:

$$y(x^*)|\mathbf{y} \sim N\Big(k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}\mathbf{y}, \\ k(x^*, x^*) - k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}k(\mathbf{x}, x^*)\Big)$$



$l = 1$

GP Predictive Distribution

Given:

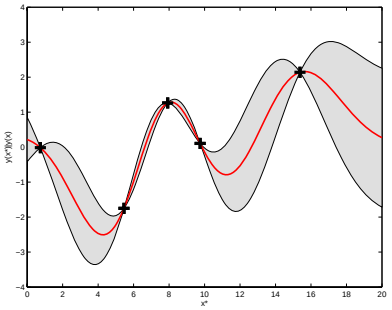
$\mathbf{x} = \begin{bmatrix} 9.8 & 15.4 & 7.9 & 5.4 & 0.7 \end{bmatrix}^T$

$\mathbf{y} = \begin{bmatrix} 0.1 & 2.1 & 1.3 & -1.7 & -0.01 \end{bmatrix}^T$

Assume: Data generated by GP with $m = 0$, $k(x, x') = e^{-(x-x')^2/l}$.

Predictive GP:

$$y(x^*)|\mathbf{y} \sim N\Big(k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}\mathbf{y}, \\ k(x^*, x^*) - k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}k(\mathbf{x}, x^*)\Big)$$



$l = 10$

GP Predictive Distribution

Given:

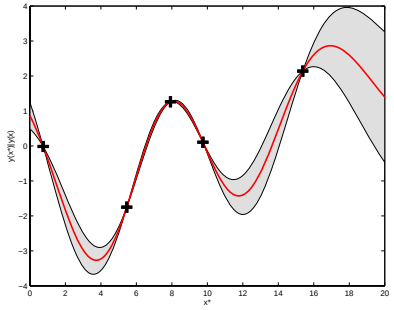
$\mathbf{x} = \begin{bmatrix} 9.8 & 15.4 & 7.9 & 5.4 & 0.7 \end{bmatrix}^T$

$\mathbf{y} = \begin{bmatrix} 0.1 & 2.1 & 1.3 & -1.7 & -0.01 \end{bmatrix}^T$

Assume: Data generated by GP with $m = 0$, $k(x, x') = e^{-(x-x')^2/l}$.

Predictive GP:

$$y(x^*)|\mathbf{y} \sim N\Big(k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}\mathbf{y}, \\ k(x^*, x^*) - k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}k(\mathbf{x}, x^*)\Big)$$



$l = 20$

GP Predictive Distribution

33(43)

Given:

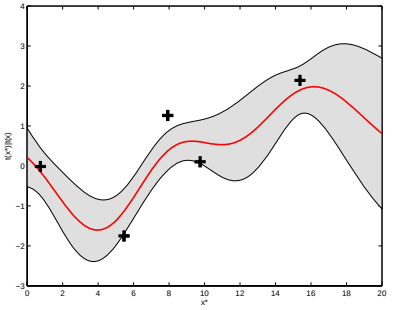
$\mathbf{x} = \begin{bmatrix} 9.8 & 15.4 & 7.9 & 5.4 & 0.7 \end{bmatrix}^T$

$\mathbf{t} = \begin{bmatrix} 0.1 & 2.1 & 1.3 & -1.7 & -0.01 \end{bmatrix}^T$

Assume: Data generated by GP with $m = 0$, $k(x, x') = e^{-(x-x')^2/l} + \sigma^2\delta_{x,x'}$, measurement noise σ^2.

Predictive GP:

$$t(x^*)|\mathbf{t} \sim N\Big(k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}\mathbf{t}, \\ k(x^*, x^*) - k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}k(\mathbf{x}, x^*)\Big)$$



$l = 20$

Machine Learning
T. Schön

AUTOMATIC CONTROL
REGLERTEKNIK
LINKÖPINGS UNIVERSITET

GP Predictive Distribution

33(43)

Given:

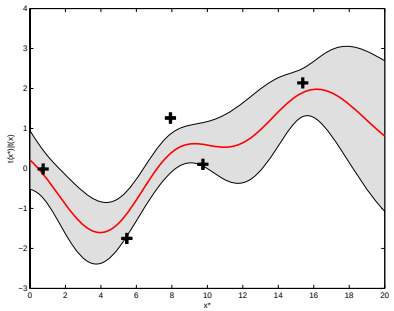
$\mathbf{x} = \begin{bmatrix} 9.8 & 15.4 & 7.9 & 5.4 & 0.7 \end{bmatrix}^T$

$\mathbf{t} = \begin{bmatrix} 0.1 & 2.1 & 1.3 & -1.7 & -0.01 \end{bmatrix}^T$

Assume: Data generated by GP with $m = 0$, $k(x, x') = e^{-(x-x')^2/l} + \sigma^2\delta_{x,x'}$, measurement noise σ^2.

Predictive GP:

$$t(x^*)|\mathbf{t} \sim N\Big(k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}\mathbf{t}, \\ k(x^*, x^*) - k(x^*, \mathbf{x})k(\mathbf{x}, \mathbf{x})^{-1}k(\mathbf{x}, x^*)\Big)$$



$l = 20$

How do we pick l?

Machine Learning
T. Schön

AUTOMATIC CONTROL
REGLERTEKNIK
LINKÖPINGS UNIVERSITET

Computing the Hyperparameters

The parameters of the kernels (e.g. l) are often referred to as hyperparameters and found through

- Cross validation
- Maximizing the log marginal likelihood (aka *empirical Bayes*)

$$\begin{aligned} \max_{\text{hyperpar}} logp(\mathbf{t}) &= \max_{\text{hyperpar}} log \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})d\mathbf{y} \\ &= \max_{\text{hyperpar}} \log \int N(\mathbf{t}; \mathbf{y}, \sigma^2)N(\mathbf{y}; 0, k(\mathbf{x}, \mathbf{x}))d\mathbf{y} \\ &= \max_{\text{hyperpar}} -\frac{1}{2}\mathbf{t}^T\left(\sigma^2 I + k(\mathbf{x}, \mathbf{x})\right)^{-1}\mathbf{t} - \frac{1}{2}\log|\sigma^2 I + k(\mathbf{x}, \mathbf{x})| - \frac{N}{2}\log 2\pi \end{aligned}$$

Machine Learning
T. Schön

AUTOMATIC CONTROL
REGLERTEKNIK
LINKÖPINGS UNIVERSITET

Design Choices

m is very often set to zero. The kernel k, the covariance function, should ideally be $k(x_t, x_s) = cov(y_t, y_s)$. Some common choices are:

- The Gaussian (also called squared exponential) kernel

$$k(x_t, x_s) = \gamma e^{-\|x_t - x_s\|^2/l}$$

  γ, l has to be chosen.
- The exponential kernel

$$k(x_t, x_s) = e^{-\|x_t - x_s\|/l}$$

  l has to be chosen.
- Linear kernel
- ...

Some Properties

- Probabilistic
- Discriminative
- Nonparametric
- Classification can also be done.
- Known under many names, e.g. Kriging (Daniel Krige, 1951).
- Can only handle Gaussian measurement noise.
- Multidimensional output
- Have to invert an $N \times N$ matrix.
- Strong relations to neural networks.

Example – CO_2

What covariance functions to chose?

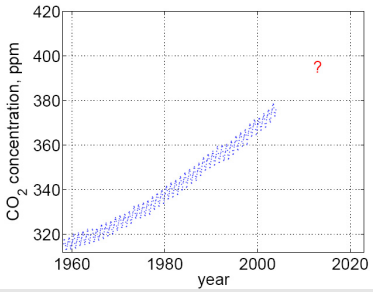
- There is a long-term smooth trend

$$k_1(x, x') = \theta_1^2 e^{-(x-x')^2/\theta_2^2}$$

- There is a periodic component

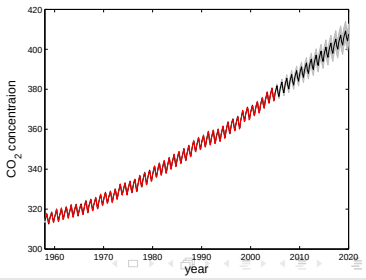
$$k_2(x, x') = \theta_3^2 e^{-\left(\sin\left(\pi(x-x')\right)\right)^2/\theta_4^2} e^{-(x-x')^2/\theta_5^2}$$

- ...

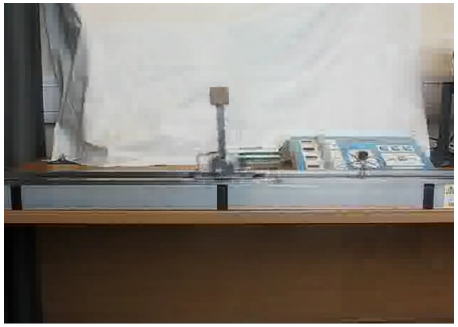


Example – CO_2


```
k1 = 'covSEiso';
k2 = {'covProd',{'covSEiso','covPeriodic'}};
k3 = 'covRQiso';
k4 = {'covSum',{'covSEiso', 'covNoise'}};
covfunc = {'covSum',{k1,k2,k3,k4}};
init=[1 1 1 1 0.3 −0.1 0.12 −0.4 −0.06 −2 −1.69 −1.6]';
loghypers = minimize(init,'gpr',−100,covfunc,x,y);
xstar=(1958:0.2:2020)';
[mu S2] = gpr(loghypers, covfunc, x, y, xstar);
```



Example – Inverted Pendulum

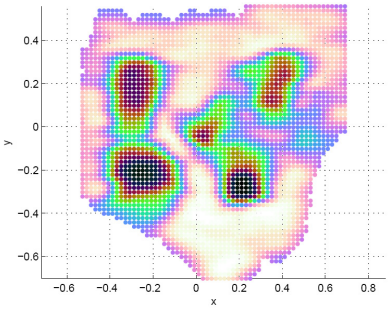


Movie: `http://www.youtube.com/watch?v=XiigTGKZfks`

Research conducted by Marc Deisenroth and Carl Edward Rasmussen at Cambridge.

Deisenroth, M. *Efficient Reinforcement Learning using Gaussian Processes*, PhD thesis, Karlsruhe Institute of Technology. 2011

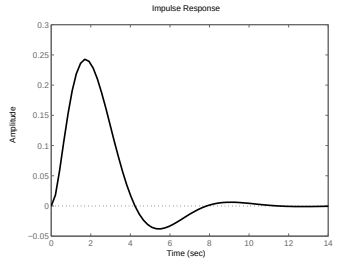
Example – LIDAR Terrain Volume Mapping Using GP



Almqvist, Erik (2010) *Airborne Mapping Using LIDAR*. MSc Thesis, Department of Electrical Engineering, Linköping University, Sweden.

Example – Impulse Response Estimation Using GP

Idea: model the impulse response from a linear system using a GP. If the system is assumed stable, we know the impulse response should approach zero as time goes to infinity. Kernels can be tailored to catch that.



Research conducted by Pillonetto et al. at University of Padova.

A Few Concepts to Summarize Lecture 4

Neural networks: A nonlinear function (as a function expansion) from a set of input variables $\{x_i\}$ to a set of output variables $\{y_k\}$ controlled by a vector w of parameters.

Backpropagation: Computing the gradients by making use of the chain rule, combined with clever reuse of information that is needed for more than one gradient.

Convolutional neural networks: The hidden units takes their inputs from a small part of the available inputs and all units have the *same* weights (called weight sharing).

Kernel function: A kernel function $k(x, z)$ is defined as an inner product $k(x, z) = \phi(x)^T \phi(z)$, where $\phi(x)$ is a fixed mapping.

Kernel trick: (a.k.a. kernel substitution) In an algorithm where the input data x enters only in the form of scalar products we can replace this scalar product with another choice of kernel.

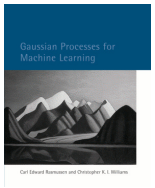
Gaussian processes (GP): A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Gaussian process regression: Assume the underlying process can be well modeled by a GP. Use Gaussian identities to compute the conditional distribution.

Further Readings and Code

Good readings:

- Rasmussen & Williams, 2006 (electronic version: `http://www.gaussianprocess.org/gpml/`).
- GP site `http://www.gaussianprocess.org/`
- Pillonetto et al. (impulse responses estimation using GP, `http://www.dei.unipd.it/~giapi/`)



GPR MATLAB toolbox:

`http://www.gaussianprocess.org/gpml/`

Video lecture on GPR:

`http://videolectures.net/mlss09uk_rasmussen_gp/`