Contents – lecture 5

1. Summary of lecture 4
2. Bayesian non-parametric (BNP) models
3. Stochastic processes
4. Gaussian processes (GP)
   - Construct a GP from a Bayesian linear regression model
   - GP regression
   - Examples where we have made use of GPs in recent research
5. Support vector machines (SVM)

Chapter 6.4 – 7.2 (Chapter 12 in HTF, GP not covered in HTF)

Summary of lecture 4 (I/II)

A neural network is a nonlinear function from a set of input variables to a set of output variables controlled by adjustable parameters $w$.

A neural network is a sequential/recursive construction of generalized linear regressions,

$$ z^{(l+1)} = f^{(l)} \left( w^{(l+1)} z^{(l)} + w_0^{(l+1)} \right), \quad z^{(0)} = x. $$

The network is trained by formulating the problem as usual, which results in a (non-convex) optimization problem. This problem is solved using numerical methods.

Backpropagation amounts to computing the gradients via (recursive) use of the chain rule, combined with reuse of information that is needed for more than one gradient.

Summary of lecture 4 (II/II)

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.

Introduced the 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.

The use of kernels allows us to implicitly use basis functions of high, even infinite, dimensions ($M \to \infty$).
Parametric vs. non-parametric models

Parametric models are built using a fixed and finite set of parameters $w$. When more data is accumulated the model complexity remains the same, since the size of $w$ is fixed.

Non-parametric models are built in such a way that the number of parameters grows with the size of the data. Hence, the number of parameters is not fixed, nor is it finite. These models are often defined in terms of an infinite dimensional $w$. In fact, we usually think of $w$ as a function.

The model complexity is allowed to grow as more and more data is accumulated. This makes the non-parametric models very flexible.

Bayesian non-parametric (BNP) models

A Bayesian non-parametric model formally has infinitely many parameters. However, in practice there in only a finite subset of them that are nonempty.

Since the complexity of a BNP model grows as more and more data are acquired, it offers a more flexible model that adapts as more data arrives.

Two of the classical building blocks for BNP models are:
1. Gaussian Process (GP), today,
2. Dirichlet Process (DP), lecture 11,

Ex. parametric vs. non-parametric models

Linear regression: $t_n = y(x_n, w) + \varepsilon_n$.

Ex. (parametric model):

$$y(x_n, w) = w^T \phi(x_n).$$

Ex. (non-parametric model): Recall the use of the equivalent kernel from lecture 1, slide 30 (with $w = m_N$), allowing us to write:

$$y(x, m_N) = \sum_{n=1}^{N} k(x, x_n) t_n,$$

where $k(x, x') = \beta \phi(x)^T S_N \phi(x')$ is the equivalent kernel. Direct use of a kernel $k(x, x')$ gives a flexible model where the complexity adapts as more data arrives.

Constructing BNP models

A good strategy for deriving non-parametric models is to consider their finite dimensional counterparts and study what happens when we “make them infinitely large”.

<table>
<thead>
<tr>
<th>Parametric</th>
<th>Non-parametric</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>polynomial regression</td>
<td>Gaussian process (GP)</td>
<td>function approx.</td>
</tr>
<tr>
<td>logistic regression</td>
<td>GP classifiers</td>
<td>classification</td>
</tr>
<tr>
<td>mixture models</td>
<td>Dirichlet process (DP)</td>
<td>mixtures, cluster.</td>
</tr>
<tr>
<td>hidden Markov model</td>
<td>$\infty$ HMMs</td>
<td>time series</td>
</tr>
<tr>
<td>factor analysis/pPCA</td>
<td>$\infty$ latent factor model</td>
<td>feature discovery</td>
</tr>
</tbody>
</table>

Table curtsey of Zoubin Gharamani.

**Stochastic processes**

Distributions on infinite dimensional objects are studied in stochastic processes theory.

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

**Properties:**
- For a fixed \( x \in \mathcal{X} \), \( y(x) \) is a random variable.
- A realization from a stochastic process is a function.

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

---

**Gaussian process (GP)**

**Definition (Gaussian process):** A stochastic process \( \{y\} \) is a Gaussian process iff for any finite subset of inputs \( \{x_1, x_2, \ldots, x_N\} \), the corresponding random variables \( y(x_1), y(x_2), \ldots, y(x_N) \) have a joint Gaussian distribution.

---

**Linear regression model on matrix form**

Write the linear regression model (without noise) as

\[
y_n = y(x_n) = w^T \phi(x_n), \quad n = 1, \ldots, N,
\]

where

\[
w = \begin{pmatrix} w_0 & w_1 & \ldots & w_{M-1} \end{pmatrix}^T
\]

and

\[
\phi(x_n) = \begin{pmatrix} \phi_0(x_n) & \phi_1(x_n) & \ldots & \phi_{M-1}(x_n) \end{pmatrix}^T
\]

on matrix form

\[
Y = \Phi w,
\]

where

\[
Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, \quad \Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \ldots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \ldots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \ldots & \phi_{M-1}(x_N) \end{pmatrix}
\]

---

**Starting from the Gaussian process**

Rather than constructing the GP from a parametric model and a certain choice of basis function, we can start by defining the mean function \( m(x) \) and the kernel function \( k(x, x') \) directly. Opens up for new, interesting and useful models!

We write that

\[
y(x) \sim \mathcal{GP}(m(x), k(x, x')),
\]

where

\[
m(x) = \mathbb{E}[y(x)],
\]

\[
k(x, x') = \mathbb{E}\left[ (y(x) - m(x))(y(x') - m(x')) \right].
\]
Starting from the GP – more concrete

Assume that \( y(x) \) is governed by a GP, i.e.
\[
y(x) \sim \mathcal{GP}(m(x), k_\theta(x,x')),
\]
By definition it holds that for any finite \( x_1, x_2, \ldots, x_N \), the distribution \( p(Y) = p(y(x_1), y(x_2), \ldots, y(x_N)) \) is given by,
\[
N \left( \begin{pmatrix} m(x_1) \\ m(x_2) \\ \vdots \\ m(x_N) \end{pmatrix}, \begin{pmatrix} k_\theta(x_1, x_1) & k_\theta(x_1, x_2) & \cdots & k_\theta(x_1, x_N) \\ k_\theta(x_2, x_1) & k_\theta(x_2, x_2) & \cdots & k_\theta(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ k_\theta(x_N, x_1) & k_\theta(x_N, x_2) & \cdots & k_\theta(x_N, x_N) \end{pmatrix} \right).
\]

Multivariate Gaussian (IV/VI)

Theorem (Affine transformations)

Assume that \( x_a \), as well as \( x_b | x_a \) are Gaussian distributed
\[
p(x_a) = \mathcal{N}(x_a | \mu_a, \Sigma_a), \quad p(x_b | x_a) = \mathcal{N}(x_b | M x_a + b, \Sigma_{b|a}),
\]
where \( M \) is a matrix and \( b \) is a constant vector. The marginal density of \( x_b \) is then given by
\[
p(x_b) = \mathcal{N}(x_b | \mu_b, \Sigma_b),
\]
\[
\mu_b = M \mu_a + b,
\]
\[
\Sigma_b = \Sigma_{b|a} + M \Sigma_a M^T.
\]
Use this theorem with \( x_a = Y, x_b = T, M = I \) and \( b = 0 \).

Samples from a GP

Let \( y(x) \sim \mathcal{GP}(0, k_\theta(x,x')) \), with \( k_\theta(x,x') = e^{-(x-x')^2/\theta} \). Let \( x = 1 : 1 : 20 \). Samples from this GP are shown below.

Multivariate Gaussian (II/VI)

Theorem (Conditioning)

Let \( x \) be Gaussian distributed and partitioned \( x = (x_a^T \quad x_b^T)^T \), then the conditional density \( p(x_a | x_b) \) is given by
\[
p(x_a | x_b) = \mathcal{N}(x_a | \mu_{a|b}, \Sigma_{a|b}),
\]
\[
\mu_{a|b} = \mu_a + \Sigma_{ab} \Sigma_{bb}^{-1} (x_b - \mu_b),
\]
\[
\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab} \Sigma_{bb}^{-1} \Sigma_{ba}.
\]
Use this theorem with \( x_a = t_{N+1} \) and \( x_b = T \).
Given:

\[ X = \begin{bmatrix} 9.8 & 15.4 & 7.9 & 5.4 & 0.7 \end{bmatrix}^T \]
\[ Y = \begin{bmatrix} 0.1 & 2.1 & 1.3 & -1.7 & -0.01 \end{bmatrix}^T \]

Assume that \( y(x) \) is governed by a GP, i.e.

\[ y(x) \sim \mathcal{GP}(m(x), k_{\theta}(x, x')) \]

where

\[ m(x) = 0, \]
\[ k_{\theta}(x, x') = e^{-\frac{(x-x')^2}{\theta}}. \]

This was the noiseless case (i.e. \( t_n = y(x_n) \)). Let’s add noise to the measurements (i.e. \( t_n = y(x_n) + \varepsilon_n \)).

Suggestion: Reproduce these figures.

GP regression to find a static one dim. nonlinear function \( h(z) \).
Finding the hyperparameters

There are typically unknown parameters $\theta$ in the kernel (and the mean) function that needs to be learned from data. Two approaches:

1. **Empirical Bayes** (lecture 2). Maximizing the marginal likelihood

$$\max_{\theta} \log p(T) = \max_{\theta} \log \int p(T | Y) p(Y) dY$$

$$= \max_{\theta} \log \int N(T | Y, \beta^{-1} I_N) N(Y | 0, K_{\theta}) dY$$

2. Assume $\theta$ to be a random variable, assign a prior to it and then integrate it out (i.e. marginalize over $\theta$).


Some GP properties

- Probabilistic
- Discriminative
- Nonparametric (a member of the model class referred to as Bayesian nonparametric (BNP) models (lecture 11)).
- Can also be used for classification (e.g. Gaussian face).
- 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, $O(N^3)$ cost. There are techniques to reduce this (e.g. sparse GPs).
- Strong relations to neural networks.

GP application – ambient magnetic field map

The Earth’s magnetic field sets a background for the ambient magnetic field. Deviations make the field vary from point to point.

**Aim:** Build a map (i.e., a model) of the magnetic environment based on measurements from magnetometers.

**Solution:** Customized Gaussian process that obey Maxwell’s equations.

Show movie!


GP application – nonlinear state space model

Consider the Gaussian Process SSM (GP-SSM):

$$x_{t+1} = f(x_t) + w_t, \quad \text{s.t. } f(x) \sim GP(0, \kappa_{\theta,f}(x,x')),$$

$$y_t = g(x_t) + e_t, \quad \text{s.t. } g(x) \sim GP(0, \kappa_{\theta,g}(x,x')).$$

The model functions $f$ and $g$ are assumed to be realizations from Gaussian process priors and $w_t \sim \mathcal{N}(0, Q), e_t \sim \mathcal{N}(0, R)$.

We can now find the posterior distribution

$$p(f, g, Q, R, \theta | y_{1:T}),$$

by making use of new MCMC algorithms.


GP application – nonlinear state space model

This gives us a flexible nonparametric model where the GP prior on f takes on the role of a regularizer. This provides a data-driven way of tuning the model complexity.

Toy example:
\[ x_{t+1} = -10 \frac{x_t}{1+3x_t^2} + w_t, \]
\[ y_t = x_t + e_t. \]

Support Vector Machines (SVM)

Very popular classifier.
- Non-probabilistic
- Discriminative
- Can also be used for regression (then called support vector regression, SVR).
- Convex optimization
- Sparse
- SVM are often used to illustrate the interplay between optimization and machine learning.

SVM for classification – the problem

Assume: \( \{(t_n, x_n)\}_{n=1}^N \), \( x_n \in \mathbb{R}^{n_x} \) and \( t_n \in \{-1, 1\} \), is a given training data set (linearly separable).

Task: Given \( x^* \), what is the corresponding label?

SVM is a discriminative classifier, i.e. it provides a decision boundary. The decision boundary is given by \( \{x \mid w^T \phi(x) + b = 0\} \).

The margin is defined as the perpendicular distance between the decision boundary and the closest point.

Goal: Find the decision boundary that maximizes the margin!

SVM for classification (I/III)

The decision boundary that maximizes the margin is given as the solution to the quadratic program (QP)
\[
\min_{w,b} \frac{1}{2} \|w\|^2 \\
\text{s.t. } t_n(w^T \phi(x_n) + b) - 1 \geq 0, \quad n = 1, \ldots, N.
\]

To enable the dimension of the feature space (dim. of \( \phi(x_n) \)) to go to infinity, we have to work with the dual problem.
SVM for classification (II/III)

First, the Lagrangian is
\[ L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n \left( t_n (w^T \phi(x_n) + b) - 1 \right) \]
and minimizing w.r.t. \( w \) and \( b \) we obtain the dual objective \( g(a) \). Taking the derivative w.r.t. \( w \) and \( b \) and set them to zero,
\[
\frac{dL(w, b, a)}{db} = \sum_{n=1}^{N} a_n t_n = 0, \quad \frac{dL(w, b, a)}{dw} = w - \sum_{n=1}^{N} a_n t_n \phi(x_n) = 0.
\]
This gives
\[
g(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} a_n a_m t_n t_m \phi(x_m)^T \phi(x_n).
\]

Support vectors – sparse version of training data

It can be shown that the KKT conditions for this optimization problem satisfies
\[
a_n \geq 0,
\]
\[
t_n y(x_n) - 1 \geq 0,
\]
\[
a_n (t_n y(x_n) - 1) = 0.
\]
The result is that for each training data the following is true
1. Either \( a_n = 0 \) or
2. \( t_n y(x_n) = 1 \).
Training data with \( a_n = 0 \) do not appear in the solution. The remaining training data (i.e., where \( t_n y_n = 1 \)) are referred to as support vectors (training data that lie on the maximum margin decision boundary).

SVM for classification (III/III)

Let \( k(x_i, x_j) = \phi(x_i)^T \phi(x_j) \). The dual objective then becomes
\[
g(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} a_n a_m t_n t_m k(x_m, x_n)
\]
which we can maximize w.r.t. \( a \) and subject to
\[
a_n \geq 0,
\]
\[
\sum_{n=1}^{N} a_n t_n = 0.
\]
The maximizing \( a \) (let us call it \( \hat{a} \)) gives (using \( w^T \phi(x^*) = (\sum_{n=1}^{N} a_n t_n \phi(x_n))^T \phi(x^*) \))
\[
y(x^*) = \sum_{n=1}^{N} \hat{a}_n t_n k(x^*, x_n) + b.
\]
Many \( \hat{a} \)'s will be zero (sparseness) ⇒ computational remedy.

SVM for classification – non-separable classes

If points are on the right side of the decision boundary, then \( t_n (w^T \phi(x_n) + b) \geq 1 \). To allow for some violations, we introduce slack variables \( \zeta_n \), \( n = 1, \ldots, N \). The modified optimization becomes
\[
\min_{w, b, \zeta} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \zeta_n
\]
s.t. \( t_n (w^T \phi(x_n) + b) + \zeta_n - 1 \geq 0, \quad n = 1, \ldots, N, \quad \zeta_n \geq 0, \quad n = 1, \ldots, N. \)
Example – CVX to compute SVM (I/II)

Linearly separable data:

```matlab
cvx_begin
variables w(nx,1) b
minimize (0.5*w'*w)
subject to
y.*(w'*x+b*ones(1,N))-ones(1,N) >= 0
cvx_end
```

Non-separable data:

```matlab
cvx_begin
variables w(nx,1) b zeta(1,N)
minimize (0.5*w'*w + C*ones(1,N)*zeta')
subject to
y.*(w'*x+b*ones(1,N))-ones(1,N)+zeta >= 0
zeta >= 0
cvx_end
```

Example – CVX to compute SVM (II/II)

SVM – Solving the dual:

```matlab
for t=1:N;for s=t:N
K(t,s)=k(x(:,t),x(:,s));K(s,t)=K(t,s);
end;end

cvx_begin
variables a(N,1)
minimize( 1/2*(a.*y')'*K*(a.*y') - ones(1,N)*a)
subject to
ones(1,N)*(a.*y') == 0
a >= 0
cvx_end
ind=find(a>0.01);
wphi = @(xstar) ones(1,N)*(a.*y'.*k(xstar,x))
b=0;
for i=1:length(ind)
b=b+1/y(ind(i))-wphi(x(:,ind(i)));
end
b=b/length(ind);
ystar = @(xstar) wphi(xstar)+b
```

A few concepts to summarize lecture 5

**Kernel:** Kernel is another name for covariance function, i.e., a function $k$ that depends on input data in the following way

$$k(x_m, x_n) = \phi(x_m)^T \phi(x_n).$$

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

**Gaussian process regression:** The GP defines a distribution over functions $p(y)$ that can then be used for Bayesian regression. There are closed-form expressions for GP regression.

**Support vector machines:** A discriminative classifier that gives the maximum margin decision boundary.

**Support vector:** A training data that lies on the maximum margin decision boundary.