Advanced Machine Learning
Lecture 5
Gaussian Processes 2
31.10.2012

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/
leibe@vision.rwth-aachen.de
This Lecture: *Advanced Machine Learning*

- **Regression Approaches**
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes

- **Bayesian Estimation & Bayesian Non-Parametrics**
  - Mixture Models & EM
  - Dirichlet Processes
  - Latent Factor Models
  - Beta Processes

- **SVMs and Structured Output Learning**
  - SV Regression, SVDD
  - Large-margin Learning
Topics of This Lecture

• Kernels
  - Recap: Kernel trick
  - Constructing kernels

• Gaussian Processes
  - Recap: Definition, Prediction, GP Regression
  - Influence of hyperparameters

• Learning Gaussian Processes
  - Bayesian Model Selection
  - Model selection for Gaussian Processes

• Gaussian Processes for Classification
  - Linear models for classification
  - Gaussian Process classification

• Applications
Recap: Kernel Ridge Regression

• Dual definition
  - Instead of working with $w$, substitute $w = \Phi^T a$ into $J(w)$ and write the result using the kernel matrix $K = \Phi \Phi^T$:
    $$J(a) = \frac{1}{2} a^T K Ka - a^T K t + \frac{1}{2} t^T t + \frac{\lambda}{2} a^T K a$$
  - Solving for $a$, we obtain
    $$a = (K + \lambda I_N)^{-1} t$$

• Prediction for a new input $x$:
  - Writing $k(x)$ for the vector with elements $k_n(x) = k(x_n, x)$
    $$y(x) = w^T \phi(x) = a^T \Phi \phi(x) = k(x)^T (K + \lambda I_N)^{-1} t$$

⇒ The dual formulation allows the solution to be entirely expressed in terms of the kernel function $k(x, x')$. 

B. Leibe
Recap: Properties of Kernels

• Theorem
  - Let $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel function. Then there exists a Hilbert Space $\mathcal{H}$ and a mapping $\varphi : \mathcal{X} \to \mathcal{H}$ such that
    $$k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}$$
  - where $\langle . , . \rangle_{\mathcal{H}}$ is the inner product in $\mathcal{H}$.

• Translation
  - Take any set $\mathcal{X}$ and any function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
  - If $k$ is a positive definite kernel, then we can use $k$ to learn a classifier for the elements in $\mathcal{X}$!

• Note
  - $\mathcal{X}$ can be any set, e.g. $\mathcal{X} = "all videos on YouTube"$ or $\mathcal{X} = "all permutations of \{1, \ldots, k\}"$, or $\mathcal{X} = "the internet"$. 
Recap: The “Kernel Trick”

Any algorithm that uses data only in the form of inner products can be kernelized.

- How to kernelize an algorithm
  - Write the algorithm only in terms of inner products.
  - Replace all inner products by kernel function evaluations.

⇒ The resulting algorithm will do the same as the linear version, but in the (hidden) feature space \( \mathcal{H} \).
  - Caveat: working in \( \mathcal{H} \) is not a guarantee for better performance. A good choice of \( k \) and model selection are important!
How to Check if a Function is a Kernel

• Problem:
  - Checking if a given $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ fulfills the conditions for a kernel is difficult:
  - We need to prove or disprove
    $$\sum_{i,j=1}^{n} t_i k(x_i, x_j) t_j \geq 0$$
    for any set $x_1, \ldots, x_n \in \mathcal{X}$ and any $t \in \mathbb{R}^n$ for any $n \in \mathbb{N}$.

• Workaround:
  - It is easy to construct functions $k$ that are positive definite kernels.
Constructing Kernels

1. We can construct kernels from scratch:
   - For any $\varphi : \mathcal{X} \rightarrow \mathbb{R}^m$, $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathbb{R}^m}$ is a kernel.
     Example: $\varphi(x) = (\# \text{ of red pixels in image } x$, green, blue).
   - Any norm $\| \cdot \| : V \rightarrow \mathbb{R}^m$ that fulfills the parallelogram equation
     $$\| x + y \|^2 + \| x - y \|^2 = 2\| x \|^2 + 2\| y \|^2$$
     induces a kernel by polarization:
     $$k(x, y) := (\| x + y \|^2 + \| x \|^2 - \| y \|^2)$$
     Example: $\mathcal{X} = \text{ time series with bounded values}$, $\| x \|^2 = \sum_{t=1}^{\infty} \frac{1}{2t} x_t$
Constructing Kernels (2)

1. We can construct kernels from scratch:
   - If \( d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) is conditionally positive definite, i.e.
     \[
     \sum_{i,j=1}^{n} t_i d(x_i, x_j) t_j \geq 0 \text{ for any } t \in \mathbb{R}^n \text{ with } \sum_i t_i = 0,
     \]
     for \( x_1, \ldots, x_n \in \mathcal{X} \) for any \( n \in \mathbb{N} \), then
     \( k(x, x') := \exp(-d(x, x')) \) is a positive kernel.

   Example: \( d(x, x') = \|x - x'\|^2 \).

     \[
     k(x, x') = \exp \left\{ -\|x - x'\|^2_{L_2} \right\}
     \]
Constructing Kernels (3)

2. We can construct kernels from other kernels:
   - If \( k \) is a kernel and \( \alpha > 0 \), then \( k \) and \( k + \alpha \) are kernels.
   - if \( k_1, k_2 \) are kernels, then \( k_1 + k_2 \) and \( k_1 \cdot k_2 \) are kernels.
   - if \( k \) is a kernel, then \( \exp(k) \) is a kernel.

- Examples for kernels for \( \mathcal{X} = \mathbb{R}^d \):
  - Any linear combination \( \sum_j \alpha_j k_j \) with \( \alpha_j \geq 0 \),
  - **Polynomial kernels** \( k(x, x') = (1 + \langle x, x' \rangle)^m \), \( m > 0 \)
  - **Gaussian a.k.a. RBF**
    \[
    k(x, x') = \exp \left\{ -\frac{\|x - x'\|^2}{2\sigma^2} \right\}
    \]
    with \( \sigma > 0 \).
Constructing Kernels (4)

2. We can construct kernels from other kernels:
   - If \( k \) is a kernel and \( \alpha > 0 \), then \( k \) and \( k + \alpha \) are kernels.
   - If \( k_1, k_2 \) are kernels, then \( k_1 + k_2 \) and \( k_1 \cdot k_2 \) are kernels.
   - If \( k \) is a kernel, then \( \exp(k) \) is a kernel.

• Examples for kernels for other \( \mathcal{X} \):
   - \( k(h, h') = \sum_{i=1}^{n} \min(h_i, h'_i) \) for \( n \)-bin histograms \( h, h' \).
   - \( k(p, p') = \exp(-\text{KL}(p, p')) \) with \( \text{KL} \) the symmetrized KL-divergence between positive probability distributions.
   - \( k(s, s') = \exp(-D(s, s')) \) for strings \( s, s' \) and \( D = \text{edit distance} \).

• Not an example: \( \tanh(a \langle x, x' \rangle + b) \) is not positive definite!
Topics of This Lecture

- **Kernels**
  - Recap: Kernel trick
  - Constructing kernels

- **Gaussian Processes**
  - Recap: Definition, Prediction, GP Regression
  - Influence of hyperparameters

- **Learning Gaussian Processes**
  - Bayesian Model Selection
  - Model selection for Gaussian Processes

- **Gaussian Processes for Classification**
  - Linear models for classification
  - Gaussian Process classification

- **Applications**
Recap: Gaussian Process

- **Gaussian distribution**
  - Probability distribution over scalars / vectors.

- **Gaussian process** (generalization of Gaussian distrib.)
  - Describes properties of functions.
  - Function: Think of a function as a long vector where each entry specifies the function value \( f(x_i) \) at a particular point \( x_i \).
  - Issue: How to deal with infinite number of points?
    - If you ask only for properties of the function at a finite number of points...
    - Then inference in Gaussian Process gives you the same answer if you ignore the infinitely many other points.

- **Definition**
  - A **Gaussian process (GP)** is a collection of random variables any finite number of which has a joint Gaussian distribution.
Recap: Gaussian Process

- A Gaussian process is completely defined by
  - **Mean function** \( m(x) \) and
    \[
    m(x) = \mathbb{E}[f(x)]
    \]
  - **Covariance function** \( k(x, x') \)
    \[
    k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]
    \]
  - We write the Gaussian process (GP)
    \[
    f(x) \sim \mathcal{GP}(m(x), k(x, x'))
    \]
Recap: GPs Define Prior over Functions

- Distribution over functions:
  - Specification of covariance function implies distribution over functions.
  - I.e. we can draw samples from the distribution of functions evaluated at a (finite) number of points.
  - Procedure
    - We choose a number of input points $X_*$
    - We write the corresponding covariance matrix (e.g. using SE) element-wise: $K(X_*, X_*)$
    - Then we generate a random Gaussian vector with this covariance matrix: $f_* \sim \mathcal{N}(0, K(X_*, X_*))$

Example of 3 functions sampled

Image source: Rasmussen & Williams, 2006
Recap: Prediction with Noise-free Observations

- Assume our observations are noise-free:
  \[ \{(x_n, f_n) \mid n = 1, \ldots, N\} \]

  - Joint distribution of the training outputs \( f \) and test outputs \( f_* \) according to the prior:
    \[
    \begin{bmatrix}
    f \\
    f_*
    \end{bmatrix}
    \sim \mathcal{N}
    \left(
    0, 
    \begin{bmatrix}
    K(X, X) & K(X, X_*) \\
    K(X_*, X) & K(X_*, X_*)
    \end{bmatrix}
    \right)
    \]

  - Calculation of posterior corresponds to conditioning the joint Gaussian prior distribution on the observations:
    \[
    f_* | X_*, X, f \sim \mathcal{N}(\bar{f}_*, \text{cov}[f_*]) \quad \bar{f}_* = \mathbb{E}[f_* | X, X_*, t]
    \]

  - with:
    \[
    \bar{f}_* = K(X_*, X)K(X, X)^{-1}f \\
    \text{cov}[f_*] = K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)
    \]
Recap: Prediction with Noisy Observations

- Joint distribution of the observed values and the test locations under the prior:

\[
\begin{bmatrix} t \\ f_* \end{bmatrix} \sim \mathcal{N} \left( 0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right)
\]

- Calculation of posterior corresponds to conditioning the joint Gaussian prior distribution on the observations:

\[
f_* | X_*, X, t \sim \mathcal{N}(\bar{f}_*, \text{cov}[f_*]) \quad \bar{f}_* = \mathbb{E}[f_* | X, X_*, t]
\]

- with:

\[
\bar{f}_* = K(X_*, X) \left( K(X, X) + \sigma_n^2 I \right)^{-1} t
\]

\[
\text{cov}[f_*] = K(X_*, X_*) - K(X_*, X) \left( K(X, X) + \sigma_n^2 I \right)^{-1} K(X, X_*)
\]

⇒ This is the key result that defines Gaussian process regression!

- Predictive distribution is Gaussian whose mean and variance depend on test points \( X_* \) and on the kernel \( k(x, x') \), evaluated on \( X \).
GP Regression Algorithm

• Very simple algorithm

\[
\begin{align*}
\text{input}: & \ X \ (\text{inputs}), \ y \ (\text{targets}), \ k \ (\text{covariance function}), \ \sigma_n^2 \ (\text{noise level}), \ x_* \ (\text{test input}) \\
2: \ L & := \text{cholesky}(K + \sigma_n^2 I) \\
\alpha & := L^\top (L \ y) \\
4: \ \bar{f}_* & := k_*^\top \alpha \\
v & := L \ k_* \\
6: \ \mathbb{V}[f_*] & := k(x_*, x_*) - v^\top v \\
\log p(y|X) & := -\frac{1}{2} y^\top \alpha - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi \\
8: \ \text{return}: & \ \bar{f}_* \ (\text{mean}), \ \mathbb{V}[f_*] \ (\text{variance}), \ \log p(y|X) \ (\log \text{marginal likelihood})
\end{align*}
\]

- Based on the following equations (Matrix inv. \(\leftrightarrow\) Cholesky fact.)

\[
\begin{align*}
\bar{f}_* & = k_*^T (K + \sigma_n^2 I)^{-1} t \\
\text{cov}[f_*] & = k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_* \\
\log p(t|X) & = -\frac{1}{2} t^T (K + \sigma_n^2 I)^{-1} t - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{N}{2} \log 2\pi
\end{align*}
\]
Recap: Computational Complexity

- Complexity of GP model
  - Training effort: $\mathcal{O}(N^3)$ through matrix inversion
  - Test effort: $\mathcal{O}(N^2)$ through vector-matrix multiplication

- Complexity of basis function model
  - Training effort: $\mathcal{O}(M^3)$
  - Test effort: $\mathcal{O}(M^2)$

- Discussion
  - If the number of basis functions $M$ is smaller than the number of data points $N$, then the basis function model is more efficient.
  - However, advantage of GP viewpoint is that we can consider covariance functions that can only be expressed by an infinite number of basis functions.
  - Still, exact GP methods become infeasible for large training sets.
Influence of Hyperparameters

- Most covariance functions have some free parameters.
  - Example:
    \[ k_y(x_p, x_q) = \sigma_f^2 \exp \left\{-\frac{(x_p - x_q)^2}{2 \cdot l^2}\right\} + \sigma_n^2 \delta_{pq} \]
  - Parameters: \((l, \sigma_f, \sigma_n)\)
    - Signal variance: \(\sigma_f^2\)
    - Range of neighbor influence (called “length scale”): \(l\)
    - Observation noise: \(\sigma_n^2\)

Slide credit: Bernt Schiele
Influence of Hyperparameters

\[ k_y(x_p, x_q) = \sigma_f^2 \exp \left\{ -\frac{(x_p - x_q)^2}{2 \cdot l^2} \right\} + \sigma_n^2 \delta_{pq} \]

- Examples for different settings of the length scale

\[ (l, \sigma_f, \sigma_n) = (0.3, 1.08, 0.00005) \]
\[ = (1, 1, 0.1) \]
\[ = (3.0, 1.16, 0.89) \]

(\(\sigma\) parameters set by optimizing the marginal likelihood)

Image source: Rasmussen & Williams, 2006
Topics of This Lecture

• Kernels
  ➢ Recap: Kernel trick
  ➢ Constructing kernels

• Gaussian Processes
  ➢ Recap: Definition, Prediction, GP Regression
  ➢ Influence of hyperparameters

• Learning Gaussian Processes
  ➢ Bayesian Model Selection
  ➢ Model selection for Gaussian Processes

• Gaussian Processes for Classification
  ➢ Linear models for classification
  ➢ Gaussian Process classification

• Applications
Learning Kernel Parameters

• Can we determine the length scale and noise levels from training data?
Bayesian Model Selection

• Goal
  - Determine/learn different parameters of Gaussian Processes

• Hierarchy of parameters
  - Lowest level
    - \( w \) - e.g. parameters of a linear model.
  - Mid-level (hyperparameters)
    - \( \theta \) - e.g. controlling prior distribution of \( w \).
  - Top level
    - Typically discrete set of model structures \( \mathcal{H}_i \).

• Approach
  - Inference takes place one level at a time.
Model Selection at Lowest Level

- Posterior of the parameters $w$ is given by Bayes’ rule

$$p(w|t, X, \theta, \mathcal{H}_i) = \frac{p(t|X, w, \theta, \mathcal{H}_i)p(w|\theta, X, \mathcal{H}_i)}{p(t|X, \theta, \mathcal{H}_i)} = \frac{p(t|X, w, \mathcal{H}_i)p(w|\theta, \mathcal{H}_i)}{p(t|X, \theta, \mathcal{H}_i)}$$

- with
  - $p(t|X, w, \mathcal{H}_i)$ likelihood and
  - $p(w|\theta, \mathcal{H}_i)$ prior parameters $w$,
  - Denominator (normalizing constant) is independent of the parameters and is called **marginal likelihood**.

$$p(t|X, \theta, \mathcal{H}_i) = \int p(t|X, w, \mathcal{H}_i)p(w|\theta, \mathcal{H}_i)dw$$

Slide credit: Bernt Schiele
Model Selection at Mid Level

• Posterior of parameters $\theta$ is again given by Bayes’ rule

$$p(\theta|t, X, H_i) = \frac{p(t|X, \theta, H_i)p(\theta|X, H_i)}{p(t|X, H_i)} = \frac{p(t|X, \theta, H_i)p(\theta|H_i)}{p(t|X, H_i)}$$

• where
  - The marginal likelihood of the previous level $p(t|X, \theta, H_i)$ plays the role of the likelihood of this level.
  - $p(\theta|H_i)$ is the hyperprior (prior of the hyperparameters)
  - Denominator (normalizing constant) is given by:

$$p(t|X, H_i) = \int p(t|X, \theta, H_i)p(\theta|H_i)d\theta$$

Slide credit: Bernt Schiele
Model Selection at Top Level

- At the top level, we calculate the posterior of the model

\[ p(\mathcal{H}_i | t, X) = \frac{p(t | X, \mathcal{H}_i) p(\mathcal{H}_i)}{p(t | X)} \]

- where
  - Again, the denominator of the previous level \( p(t | X, \mathcal{H}_i) \) plays the role of the likelihood.
  - \( p(\mathcal{H}_i) \) is the prior of the model structure.
  - Denominator (normalizing constant) is given by:

\[ p(t | X) = \sum_i p(t | X, \mathcal{H}_i) p(\mathcal{H}_i) \]
Bayesian Model Selection

- **Discussion**
  - Marginal likelihood is main difference to non-Bayesian methods
  - It automatically incorporates a trade-off between the model fit and the model complexity:
    - A simple model can only account for a limited range of possible sets of target values - if a simple model fits well, it obtains a high posterior.
    - A complex model can account for a large range of possible sets of target values - therefore, it can never attain a very high posterior.
Bayesian Model Selection

• Computational issues
  - Requires the evaluation of several integrals, which may or may not be analytically tractable, depending on details of the models.
  - In general, one may have to resort to analytic approximations or MCMC methods. (→ Lecture 7)

• Model selection for GP regression
  - GP regression models with Gaussian noise are an (important) exception:
    - Integrals over the parameters are analytically tractable and
    - At the same time, the models are flexible.
Example
Example
Example
Example
Example
Example
Example
Example

![Graph 1](image1)

![Graph 2](image2)

Slide credit: Bernt Schiele
Example
Topics of This Lecture

• Kernels
  • Recap: Kernel trick
  • Constructing kernels

• Gaussian Processes
  • Recap: Definition, Prediction, GP Regression
  • Influence of hyperparameters

• Learning Gaussian Processes
  • Bayesian Model Selection
  • Model selection for Gaussian Processes

• Gaussian Processes for Classification
  • Linear models for classification
  • Gaussian Process classification

• Applications
Classification

• Classic view of classification
  - Prediction: we want to assign an input pattern $x$ to one of $C$ classes.

• Probabilistic classification
  - Predictions take the form of class probabilities.
  - More general than class assignments
    - Class assignment is obtained by solving a decision problem that involves the predictive probabilities as well as costs in making the correct/wrong decision.

• Relation to regression
  - Both regression and classification can be seen as function approximation.
  - Solution of classification problems using Gaussian processes is (unfortunately) more demanding...
Classification Problem

• Setting
  - Given input patterns \( x \)
  - And corresponding class labels: \( y = C_1, \ldots, C_C \)
  - We are interested in \( p(y \mid x) \)

• Goal
  - Calculate posterior probabilities for each class using
    \[
    p(y \mid x) = \frac{p(x \mid y)p(y)}{\sum_{c=1}^{C} p(x \mid C_c)p(C_c)}
    \]
  - **Generative** approach
    - Learn model for \( p(x \mid y) \) (learning a model for \( p(y) \) is often simple)
  - **Discriminative** approach
    - Learn directly model for \( p(y \mid x) \).
Classification Problem

- Problem when applying Gaussian processes
  - Goal is to model the posterior probabilities of the target variable, which must lie in the interval $(0,1)$.
  - The GP makes predictions that lie in $(-\infty, \infty)$.

- Solution
  - Adapt Gaussian processes by transforming the output using an appropriate nonlinear activation function.

\[
\sigma(f(x))
\]
Classification Problem

- Discriminative approaches for the binary case (2 classes)
  - Linear logistic regression model
    - Combines the linear model with a logistic response function
      \[ p(C_1|x) = \lambda(x^T w) \quad \lambda(z) = \frac{1}{1 + \exp(-z)} \]
  - Linear probit regression model
    - Combines the linear model with the probit response function (cumulative density function of standard normal distribution)
      \[ p(C_1|x) = \Phi(x^T w) \quad \Phi(z) = \int_{-\infty}^{z} \mathcal{N}(x|0, 1) dx \]

- Note:
  - The Gaussian process classifiers developed in the following are discriminative.
Linear Models for Classification

• Setting
  - Binary classification: \( y = +1 \) and \( y = -1 \)
  - Likelihood is:
    \[
    \begin{align*}
    p(y = +1 | x, w) &= \sigma(x^T w) \\
    p(y = -1 | x, w) &= 1 - \sigma(x^T w)
    \end{align*}
    \]
    - For linear logistic regression: \( \sigma(z) = \lambda(z) \)
    - For linear probit regression: \( \sigma(z) = \Phi(z) \)
  
  - Given a data point \((x_i, y_i)\) and noting that \( \sigma(-z) = 1 - \sigma(z) \)
    the likelihood can be written in compact form:
    \[
    p(y_i | x_i, w) = \sigma(y_i x^T w)
    \]
Linear Models for Classification

• Learning
  - Given a data set: \( D = \{(x_i, y_i)|i = 1, \ldots, n\} \)
  - Assuming data is i.i.d.

• Maximum likelihood (ML):
  - Maximize data log-likelihood given by
    \[
    \log p(y|X, w) = \sum_{i=1}^{n} \log \sigma(y_i x_i^T w)
    \]

• Maximum a posteriori (MAP):
  - Assume Gaussian prior over weights:
  - Maximize \( p(w|X, y) \propto p(y|X, w)p(w) \)
  - I.e. \( \log p(w|X, y) \propto \sum_{i=1}^{n} \log \sigma(y_i x_i^T w) - \frac{1}{2} w^T \Sigma^{-1} w \)

Slide credit: Bernt Schiele
Linear Models for Classification

- Example of MAP-solution
  - 2D input space
  - 2D weight space (no offset)

\[ p(w) = \mathcal{N}(0, I) \]

Note: posterior is uni-modal but non-Gaussian.

Image source: Rasmussen & Williams, 2006
Linear Models for Classification

- Predictions
  - Predictions based on the training set $\mathcal{D}$ for a test point $x_*$ is given by
    \[ p(y_* | x_*, X, y) = \int p(y_* = +1 | w, x_*)p(w | X, y)dw \]
  - This leads to contours of the predictive distribution:
    \[ p(w | X, y) \]

Slide credit: Bernt Schiele

Image source: Rasmussen & Williams, 2006
Gaussian Process Classification

- Basic idea (for the binary case)
  - Place a GP prior over the latent function $f(x)$
  - “Squash” this function through the logistic function to obtain a prior on
    $$
    \pi(x) \triangleq p(y = +1 | x) = \sigma(f(x))
    $$
  - Note
    - $\pi$ is a deterministic function of $f$. Since $f$ is stochastic, so is $\pi$.
  - Example

\[\begin{align*}
  f(x) & \text{ drawn from GP prior} \\
  \pi(x) & = \lambda(f(x))
\end{align*}\]
Latent Function $f$

- Function $f$ plays the role of a nuisance function
  - We do not observe values of $f$ itself.
  - And we are not particularly interested in the values of $f$.
  - Rather, we are interested in values of $\pi(x)$ and
  - Specifically for test cases: $\pi(x_*)$

- Purpose of $f$:
  - To allow convenient formulation of the model.
  - Our computational goal is to eliminate $f$ (by means of integration over $f$).
Inference and Prediction

- Natural division into 2 steps
  1. Computing the distribution of the latent variable corresponding to test case:

\[
p(f_\star|X, y, x_\star) = \int p(f_\star|X, x_\star, f)p(f|X, y)df
\]

- where the posterior over the latent variables is given by

\[
p(f|X, y) = \frac{p(y|f)p(f|X)}{p(y|X)}
\]

2. Using the distribution over the latent \( f_\star \) to produce the probabilistic prediction

\[
\overline{\pi}_\star \triangleq p(y_\star = +1|X, y, x_\star) = \int \sigma(f_\star)p(f_\star|X, y, x_\star)df_\star
\]
Inference and Prediction

- For regression
  - Computation of predictions was easy, as the relevant integrals were Gaussian and could be computed analytically.

- For classification
  - Non-Gaussian likelihood makes the integral analytically intractable.
  - Approximations of the integrals
    - e.g. based on Monte Carlo sampling
    - e.g. Laplace approximation method
      (see Rasmussen & Williams, Chapter 3.4)
    - ...

Slide credit: Bernt Schiele
Topics of This Lecture

• Kernels
  - Recap: Kernel trick
  - Constructing kernels

• Gaussian Processes
  - Recap: Definition, Prediction, GP Regression
  - Influence of hyperparameters

• Learning Gaussian Processes
  - Bayesian Model Selection
  - Model selection for Gaussian Processes

• Gaussian Processes for Classification
  - Linear models for classification
  - Gaussian Process classification

• Applications
Application: Non-Linear Dimensionality Reduction

- 2D manifold in 3D space
- 3D articulated body space
- 2D space
- 2D latent space

Slide credit: Andreas Geiger
Gaussian Process Latent Variable Model

- At each time step \( t \), we express our observations \( y \) as a combination of basis functions \( \psi \) of latent variables \( x \).

\[
y_t = \sum b_j \psi_j(x_t) + \delta_t
\]

- This is modeled as a Gaussian process...
Example: Style-based Inverse Kinematics

Learned GPLVMs using a walk, a jump shot and a baseball pitch

Slide credit: Andreas Geiger
Application: Modeling Body Dynamics

- **Task:** estimate full body pose in $m$ video frames.
  - High-dimensional $Y$
  - Model body dynamics using hierarchical Gaussian process latent variable model (hGPLVM) [Lawrence & Moore, ICML 2007].

$$
T = [t_i \in \mathbb{R}]
$$

$$
p(Z|T, \theta) = \prod_{i=1}^{q} \mathcal{N}(Z_{:,i}|0, K_T)
$$

$$
Z = [z_i \in \mathbb{R}^q]
$$

$$
p(Y|Z, \theta) = \prod_{i=1}^{D} \mathcal{N}(Y_{:,i}|0, K_z)
$$

$$
Y = [y_i \in \mathbb{R}^D]
$$
Application: Mapping b/w Pose and Appearance

- Appearance prediction
  - Regression problem
  - High-dimensional data on both sides
  \( \Rightarrow \) Low-dim. representation needed for learning!

- Training with Motion-capture data possible
  - Synthesized silhouettes for training
  - Background subtraction for test

\[ \text{Jaeggli, Koller-Meier, Van Gool, ACCV’07} \]
Learning a Generative Mapping

\[ X : \text{Body Pose (high dim.)} \rightarrow \text{PCA projection} \rightarrow y : \text{Appearance Descriptor: (low dim.)} \]

\[ x : \text{Body Pose (low dim.)} \rightarrow \text{learn LLE dim. red.} \rightarrow \text{reconstruct pose} \rightarrow \text{likelihood} \rightarrow \text{dynamic prior} \rightarrow \text{generative mapping} \]

\[ Y : \text{Image (high dim.)} \]

[Jaeggli, Koller-Meier, Van Gool, ACCV’07]
Experimental Results

- Difficulties
  - Changing viewpoints
  - Low resolution (50 px)
  - Compression artifacts
  - Disturbing objects
Articulated Motion in Latent Space (different work)

- Gaussian Process regression from latent space to
  - Pose \( = p(\text{Pose} \mid z) \) to recover original pose from latent space
  - Silhouette \( = p(\text{Silhouette} \mid z) \) to do inference on silhouettes

Walking cycles have one main (periodic) DOF
Additional DOF encodes „walking style“

B. Leibe

[Gammeter, Ess, Leibe, Schindler, Van Gool, ECCV’08]
Results

454 frames (~35 sec)
23 Pedestrians
20 detected by multi-body tracker

B. Leibe

[Gammeter, Ess, Leibe, Schindler, Van Gool, ECCV’08]
References and Further Reading

• Kernels and Gaussian Processes are (shortly) described in Chapters 6.1 and 6.4 of Bishop’s book.

  Christopher M. Bishop
  Pattern Recognition and Machine Learning
  Springer, 2006

  Carl E. Rasmussen, Christopher K.I. Williams
  Gaussian Processes for Machine Learning
  MIT Press, 2006

• A better introduction can be found in Chapters 3 and 5 of the book by Rasmussen & Williams (also available online: [http://www.gaussianprocess.org/gpml/](http://www.gaussianprocess.org/gpml/))