T-61.5040

Learning Models and Methods, (5 cr) L

Spring 2007

Lecturer: **Petteri Pajunen**

Course Assistant: **Ville Viitaniemi**

LECTURE 12: 18.4.2007

GAUSSIAN PROCESSES

Gaussian Processes

- so far the emphasis has been on solving a learning problem with a given model
- the probabilistic model links the unknown quantities θ and data
- the model should be constructed using the information available about the problem
- without a model (implicit or explicit), there is no learning

- practical difficulties:
  - closed-form results often difficult or impossible
  - approximations computationally heavy (simulation)
  - sometimes difficult to specify the model if θ has no clear interpretation

- above difficulties may be unavoidable if accurate results are needed
- *local learning* was discussed early in the course
- it was demonstrated that certain "all-purpose" learning methods essentially learn locally
- this means that prediction will depend on nearby observed data
- the intuitive idea: if inputs x_i, x_j are close, then outputs y_i, y_j should have similar values

- this amounts to an implicit model which favours "smooth" solutions
- technically this works no better than any other method (NFL theorems)
- but what if the information available is "smooth solutions are more probable"?
- a model is needed which places a high prior probability on smooth regression functions

- consider a parametric regression model, such as $y = f(x|\theta) + n$
- the prior is over parameter θ, not the function $f(x|\theta)$
- given θ, is $f(x|\theta)$ regular or rapidly varying?
- for complicated models this may not be easy to determine

- solving a given learning problem, prior for θ comes from problem-specific information
- in general-purpose methods, parameters often have no direct meaning
- defining the prior over the functions $f(x|\theta)$ may be easier
- no fundamental difference: with "equivalent" priors, all predictions will be identical

- example: likelihoods $g(x|\gamma)$ and $f(x|\theta)$
- assumption: for any γ there is θ so that $g(x|\gamma) = f(x|\theta)$ for all x (and vice versa)
- also assume that $p(\gamma)$ and $p(\theta)$ are such that the same functions get the same prior probability
- then *all predictions are the same* whether you use $f(x|\theta)$ or $g(x|\gamma)$

- *Gaussian Processes* allow solving a regression problem in closed form without requiring a prior on abstract parameters
- can be developed as a parametric model, or using a prior for a stochastic process generating the data
- parametric model corresponds to earlier results, but the equivalent process approach is simpler
- GP classification does not yield closed-form results, but requires approximations

Bayesian Regression

- Gaussian regression: likelihood is

$$p(y|H, \theta) = N(y|H\theta, \sigma^2 I)$$

- H is a known matrix, and is a function of inputs x_i
- example:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

- this means $y_i = \theta_1 + \theta_2 x_i + n_i,\ n_i \sim N(0, \sigma^2)$

- one can also define $[H]_{ij} = h_j(x_i)$, where h_j is some known, nonlinear basis function
- note that this is similar to the Support Vector Machine
- the linear regression on the “features” $h(x_i)$ becomes nonlinear regression on the inputs x_i
- the solution is equally easy in both cases, except for computational complexity

- we are interested in predicting the output $\tilde{y}$ for a new input $\tilde{x}$:

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$
$$p(\tilde{y}|y) = \int p(\tilde{y}|\theta)p(\theta|y)d\theta$$

- the prediction does not depend on θ, since it is marginalized out
- however, we need the prior for θ before it can be marginalized

- lets choose a Normal prior for θ:

$$p(\theta) = N(\theta|0, \sigma_0^2 I)$$

- the outputs of the regression function form a vector

$$Y = H\theta = \sum_{j=1}^{m} \theta_j h_j$$

- Y is a random vector and its distribution is induced by $p(\theta)$
- Y is a zero-mean Normal vector since it is a linear combination of Normally distributed zero-mean variables

- the covariance matrix of Y is

$$\mathrm{E}(YY') = H\,\mathrm{E}(\theta\theta')H' = \sigma_0^2 HH'$$

- this gives the induced prior on Y:

$$p(Y) = N(Y|0, \sigma_0^2 HH') = N(Y|0, Q)$$

- $p(\theta)$ is the parametric prior, and $p(Y)$ is a prior *on the data itself*

- the random vector Y contains the noiseless, linear outputs of the model
- we ultimately need the observable, noisy outputs y
- the predictive distribution of y is

$p(y) = N(y|0, Q + \sigma^2 I) = N(y|0, C)$ (noise is zero-mean and inc

- the prior $p(y)$ is a function of known quantities H, σ_0^2, σ^2

- the covariance $C = \sigma_0^2 HH' + \sigma^2 I$ defines $p(y)$
- each element of the covariance matrix is

$$[C]_{ij} = \sigma_0^2 \sum_k h_k(x_i) h_k(x_j) + \sigma^2 \delta_{ij}$$

- the “feature vector” of input x_i is

$$h(x_i) = [h_1(x_i), h_2(x_i), \ldots, h_m(x_i)]'$$

- then $[C]_{ij}$ is essentially an inner product between $h(x_i)$ and $h(x_j)$ (compare with SVM again)

- summary:
  - linear regression becomes nonlinear with nonlinear feature space
  - parameters θ can be integrated out, leaving a Normal distribution over outputs y
  - covariance matrix contains inner products of feature vectors

- can we do regression without parameters?
- define a prior on *observable outputs* y as $p(y) = N(y|0, C)$
- define covariances as functions of inputs: $[C]_{ij} = D(x_i, x_j)$
- it can be computed if D is known, since all inputs x_i are known

- the *covariance function* $D(x, x')$ must be chosen (compare with the kernel function in SVM)
- all covariances $\mathrm{E}(y_i y_j) = D(x_i, x_j)$ can then be computed
- only restriction for $D(x, x')$ is that the matrix C computed using it must be positive semidefinite
- the *Gaussian Process prior* $N(y|0, C)$ is then completely defined without a parametric model

Gaussian Process Predictions

- lets solve the regression problem using

$$p(y) = N(y|0, C)$$

- predict $\tilde{y}$ at an input $\tilde{x}$, given y and x
- the predictive distribution is obtained by

$$p(\tilde{y}, y) = p(\tilde{y}|y)p(y)$$
$$\implies p(\tilde{y}|y) = p(y, \tilde{y}) / p(y)$$

- the joint distribution $p(y, \tilde{y})$ is Normal and not conditional to anything
- it is given by the prior $N(0, \tilde{C})$
- the matrix $\tilde{C}$ is computed at all pairs from $\{x_1, \ldots, x_n, \tilde{x}\}$
- the predictive distribution is easily seen to be Normal since it is obtained by considering $p(y, \tilde{y})$ with y known

- the covariance matrix $\tilde{C}$ of $(y, \tilde{y})$ can be partitioned as

$$\tilde{C} = \begin{bmatrix} C & k \\ k' & c \end{bmatrix}$$

$$C = \text{ covariance matrix of } y$$
$$k = [D(x_1, \tilde{x}), \ldots, D(x_n, \tilde{x})]', \text{ a vector}$$
$$c = D(\tilde{x}, \tilde{x}), \text{ a scalar}$$

- this gives a predictive distribution

$$p(\tilde{y}|y) \propto \exp[-\frac{1}{2}(y, \tilde{y})'\tilde{C}^{-1}(y, \tilde{y})]$$

- a partitioned matrix $\tilde{C}$ has a partitioned inverse:

$$\tilde{C}^{-1} = \begin{bmatrix} M & m \\ m' & \mu \end{bmatrix}$$

where

$$\mu = (c - k'C^{-1}k)^{-1}$$
$$m = -\mu C^{-1}k$$
$$M = C^{-1} + \frac{mm'}{\mu}$$

- putting these into the equation for the predictive distribution, we find

$$p(\tilde{y}|y) \propto \exp\left[-\frac{(\tilde{y}-k'C^{-1}y)^2}{2(c-k'C^{-1}k)}\right]$$

- can we compute this?
- yes, since c, C, and k depend only on points x and $\tilde{x}$ which we know
- note that the matrix C^{-1} depends only on training data and not $\tilde{x}$

- summary of regression using a Gaussian Process:
  1. choose a covariance function $D(x_i, x_j)$, use it to compute C and then invert C
  2. choose $\tilde{x}$ as the point at which you want to predict $\tilde{y}$: compute k and c using the covariance function
  3. compute the mean of predictive distribution: $k'C^{-1}y$
  4. compute the variance of the predictive distribution: $c - k'C^{-1}k$

- the results depend on the covariance function, which directly speficies properties of the actual data
- one can easily implement various kinds of smoothness assumptions through it
- for example, a high positive covariance when $\|x - x'\|$ is somewhat large favours very smooth solutions

- an exponential covariance can be derived from a neural network with an arbitrary number of neurons (exercise)
- the covariance works even when the number of neurons approaches infinity
- lets examine the effects of covariance functions using

$$D(x, x') = a\exp(-\frac{1}{2}\frac{(x-x')^2}{b^2}) + c\delta(x, x')$$

- demo: `gpreg.R`

- the simple solution presented above relies on a known $D(x, x')$
- this may be unrealistic: some properties of the covariance may be unknown
- the noise variance is the most obvious example
- with fixed $D(x, x')$, smoothing of data is constant wrt number of training points
- Bayesian solution: add parameters θ to the covariance function

- predictions are averaged as

$$p(\tilde{y}|y) = \int p(\tilde{y}|y, \theta) p(\theta|y) d\theta$$

- the first term in the integrand is the Normal predictive distribution for a given θ
- but its dependence on θ is quite complicated: θ appears in the covariance typically in a nonlinear manner

- we can approximate by computing the MAP estimate $\hat{\theta}$, and then using $p(\tilde{y}|y, \hat{\theta})$
- to do this, maximize

$$\log p(\theta|y) = \log p(y|\theta) + \log p(\theta) + D$$

- the log-likelihood part and its derivative are

$$\log p(y|\theta) = -\frac{1}{2} \log |C| - \frac{1}{2} y^T C^{-1} y + D_2$$

$$(\log p(y|\theta))' = -\frac{1}{2} \operatorname{tr}(C^{-1} C') + \frac{1}{2} y^T C^{-1} C' C^{-1} y$$

- the log-prior must also be derivated: then we obtain a gradient for $\log p(\theta|y)$ wrt θ, and we can use gradient ascent to estimate the most probable θ
- example: use the same covariance as before, with parameters a, b and c, and use gradient ascent to optimize covariance parameters
- demo: `gphier.R`

Gaussian Processes for Classification

- technically classification is regression, but a realistic model for classification is usually different
- example: class labels (outputs) in $\{-1, 1\}$
- outputs are not assumed to have noise (labels are either -1 or 1, not e.g. 0.2)
- even if outputs are modeled as noisy class labels, the training outputs are noise-free

- goal is to classify a new input $\tilde{x}$ given D (containing inputs $x_1, \ldots, x_n$ and outputs $y_1, \ldots, y_n$)
- GP regression is not directly applicable because it requires a Normal distribution for the outputs
- use *latent variables* $u_1, \ldots, u_n$ which are obtained by GP regression on inputs x_i
- the class label distribution $p(y = 1|u)$ must be defined

- examples:
  - 'hard' classifier: $p(y = 1|u) = 1$ when $u > 0$
  - 'soft' classifier: $p(y = 1|u) = \Phi(u/\alpha)$, where $\Phi(x) = \int_{-\infty}^{x} N(y|0, 1)dy$ (cdf of a Normal distribution)
- the regression $x \mapsto u$ does most of the work
- the nonlinear $p(y = 1|u)$ simply scales the outputs to probabilities

- denote by $\tilde{u}$ the latent variable corresponding to $\tilde{x}$
- also denote by $u = (u_1, \ldots, u_n)$ the latent variables corresponding to the training inputs
- the predictive distribution (write $U = (u, \tilde{u})$) is

$$\begin{aligned} p(\tilde{y} = 1|\tilde{x}, D) &= \int p(\tilde{y} = 1, U|\tilde{x}, D)dU \\ &= \int p(\tilde{y} = 1|\tilde{u})p(U|\tilde{x}, D)dU \\ &= \int p(\tilde{y} = 1|\tilde{u})p(\tilde{u}|u, \tilde{x}, D)p(u|\tilde{x}, D)dU \end{aligned}$$

- lets examine the factors in the integral:
  - $p(\tilde{y} = 1|\tilde{u})$ is a known function of $\tilde{u}$ (chosen earlier)
  - $p(\tilde{u}|u, \tilde{x}, D)$ is GP regression (predict output $\tilde{u}$ using u)
  - $p(u|\tilde{x}, D)$ is difficult, since D has only the outputs $y_i \in \{-1, 1\}$
- for realistic classifier models, no closed-form solution exists

- there are several methods to solving GP classification approximately:
  - simulation (Radford Neal)
  - mean-field algorithms (Opper and Winther)
  - variational approximation (Gibbs and MacKay)
- these methods are not discussed here due to lengthy details: see references on the course webpage

- to demonstrate the feasibility of solving classification problems using GP's, the naive mean-field algorithm of Opper and Winther is applied to certain datasets
- the mean-field approximation assumes that $\tilde{u}$ is Normally distributed, given the training data
- the algorithm is an iterative method for finding coefficients α_i that appear as linear weights in the solution (similar to SVM)
- demo: `opper2.R`