Homework 6

Handed out: Monday, March 18, 2019
Due: Friday, April 08, 2019 11:55 pm

Notes:

- We *highly* encourage typed (Latex or Word) homework. Compile as single report containing solutions, derivations, figures, etc.
- Submit all files including report pdf, report source files (e.g. .tex or .docx files), data, figures produced by computer codes and programs files (e.g. .py or .m files) in a **.zip** folder. Programs should include a Readme file with instructions on how to run your computer programs.
- Zipped folder should be turned in on Sakai with the following naming scheme: **HW6_LastName_FirstName.zip**
- Collaboration is encouraged however all submitted reports, programs, figures, etc. should be an individual student's writeup. Direct copying could be considered cheating.
- Homework problems that simply provide computer outputs with no technical discussion, Algorithms, etc. will receive no credit.
- Software resources set can be downloaded from this link.

1 Support vector machine

A. The problem with kernelized ridge regression is that the solution vector w depends on all the training inputs. Vapnik (Vapnik et al. 1997) proposed a variant of the Huber loss function called the epsilon insensitive loss function, defined by

$$L_\epsilon \triangleq \begin{cases} 0 \text{ if } |y-\hat{y}| < \epsilon \\ |y-\hat{y}| - \epsilon \text{ otherwise} \end{cases} \tag{1}$$

The corresponding objective function is usually written in the following form

$$J = C\sum_{i=1}^{N} L_\epsilon\left(y_i, \hat{y}_i\right) + \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2, \tag{2}$$

where

$$\hat{y}_i = w_0 + \boldsymbol{w}^T\boldsymbol{x}, \tag{3}$$

and $C = \frac{1}{\lambda}$ is a regularization constant. The objective is convex and unconstrained, but not differentiable, because of the absolute value function in the loss term. One popular approach is to formulate the problem as a constrained optimization problem. In particular, we introduce slack variables to represent the degree to which each point lies outside the tube:

$$y_i \le f\left(\boldsymbol{x}_i\right) + \epsilon + \xi_i^+ \tag{4}$$

$$y_i \geq f\left(\boldsymbol{x}_i\right) - \epsilon - \xi_i^- \tag{5}$$

Given this, we can rewrite the objective as follows

$$J = C \sum_{i=1}^{N} \xi_i^+ + \xi_i^- + \frac{1}{2} \left|\left|\boldsymbol{w}\right|\right|^2, \tag{6}$$

with this setup, the optimal solution has the form

$$\hat{\boldsymbol{w}} = \sum_i \alpha_i \mathbf{x}_i \tag{7}$$

where

$$\alpha_n = \left(a_n - \hat{a}_n\right) \tag{8}$$

where a_n and $\hat{a}_n$ are the Lagrange multipliers corresponding to Eqs. 4 and 5.

(a) For the regression support vector machine considered above, show that all training data points for which $\xi_n > 0$ will have $a_n = C$ and similarly all points for which $\hat{\xi}_n > 0$ will have $\hat{a}_n = C$.

(b) Compute the dual lagrangian for the support vector regression (Refer Eq. 7.61 of Bishop)

B. For the data-set given at this link, train a support vector machine with polynomial kernel p. Perform for various polynomial orders and plot order vs. error. To ensure hard margin, use $C = 10^6$.

2 Relevance vector machine

A. The relevance vector machine for regression is a linear model but with a modified prior that results in sparse solutions. The model defines a conditional distribution for a real-valued target variable t given an input vector $\boldsymbol{x}$, which takes the form

$$p\left(t|\boldsymbol{x}, \boldsymbol{w}, \beta\right) = \mathcal{N}\left(t|y\left(\boldsymbol{x}\right), \beta^-1\right) \tag{9}$$

The symbols in above equation takes the usual form.

Suppose we are given a set of N observations of the input vector $\boldsymbol{x}$, which we denote collectively by a data matrix $\boldsymbol{X}$ whose n-th row is $\boldsymbol{x}_n^T$ with $n = 1, \ldots, N$. The corresponding target values are given by $\boldsymbol{t} = \left(t_1, \ldots, t_N\right)^T$. Thus, the likelihood function is given by

$$p\left(t|\boldsymbol{X}, \boldsymbol{w}, \beta\right) = \prod_{i=1}^{N} p\left(t_n|\boldsymbol{x}_n, \boldsymbol{w}, \beta\right) \tag{10}$$

Next we introduce a prior distribution over the parameter vector $\boldsymbol{w}$ as

$$p\left(\boldsymbol{w}|\boldsymbol{\alpha}\right) = \prod_{i=1}^{M} \mathcal{N}\left(\boldsymbol{w}|\mathbf{0}, \alpha_i^-1\right), \tag{11}$$

where α_i represents the precision of the corresponding parameter w_i. When we maximize the evidence with respect to these hyperparameters, a significant proportion of them go to infinity, and the corresponding weight parameters have posterior distributions that are concentrated at zero.

(a) For RVM discussed above, compute mean and covariance of the posterior distribution over weights.

(b) Derive the result for the marginal likelihood function in the regression RVM, by performing the Gaussian integral over w in (7.84) using the technique of completing the square in the exponential.

(c) Derive the re-estimation equations as discussed in the notes.

B. For the data-set provided at this link, train a regression model by using RVM. Use

$$y\left(\boldsymbol{x}\right)=\sum_{i=1}^{N} w_n k\left(\boldsymbol{x},\boldsymbol{x}_n\right)+b \tag{12}$$

where b is the bias parameter. Consider the kernel k to be Gaussian with kernel width 5.5. Plot the predictions along with proper prediction bounds.

3 Gaussian process

Six training points are given, with the input $X = [1.71, 2.33, 4.33, 4.84, 4.86, 5.54]$ and the output $Y = [-0.2138, 1.0389, 0.7630, 0.2271, 0.2733, 1.0565]$. Implement the zero-mean Gaussian process regression model with the following covariance kernel:

$$k\left(x,x'\right)=\theta_1 \exp\left(-\frac{\left(x-x'\right)^2}{2\theta_2}\right)+\theta_3\delta\left(x,x'\right) \tag{13}$$

where δ is the kronecker delta.

A. Learn the values of θ_{1-3} by maximizing the log-likelihood (MLE) based on the training points. The log-likelihood is known to be non-convex, so try several starting points for optimization to see if that affects the results.

B. Run your Gaussian process with the learnt parameters to predict for the testing points on the interval $x^* \in [1,6]$. Plot the prediction mean as well as the 95% confidence interval (two-sigma) of the learnt Gaussian process, and compare with their exact values computed by the formula:$y =\sim (2x)$. Comment on the prediction performance.

4 Gaussian process latent variable model

A. One approach to unsupervised learning with GPs is the Gaussian process latent variable model (GP-LVM) proposed by Lawrence (2004, 2005). GP-LVM can be considered as a multiple-output GP regression model where only the output data are given.

The inputs are unobserved and are treated as latent variables, however instead of integrating out the latent variables, they are optimized. This trick makes the model tractable and some theoretical grounding for the approach is given by the fact that the model can be seen as a nonlinear extension of the linear probabilistic PCA (PPCA). Considering, $\boldsymbol{Y} \in \mathbb{R}^{N\times D}$ be the observed data and $\boldsymbol{X} \in \mathbb{R}^{N\times Q}$ to be the associated latent variables where $D > Q$,

(a) provide an expression for the likelihood of the data. Assume, the GPs to be independent across the features.

(b) Considering a prior of the form

$$p\left(\boldsymbol{X}\right) = \prod_{n=1}^{N} \mathcal{N}\left(\boldsymbol{x}_n|\mathbf{0}, \mathbf{I}_Q\right), \tag{14}$$

The marginal likelihood of the data can be written as

$$p\left(Y\right) = \int p\left(\boldsymbol{X}\right) p\left(\boldsymbol{Y}|\boldsymbol{X}\right). \tag{15}$$

The marginal distribution shown in Eq. 15 is not tractable. Why?

(c) Provide a variational approximation for the marginal distribution.

B. For the oil data given at this link, use GPLVM to identify the latent dimensions. Consider latent dimensions to be 3. Show the three latent dimensions.

5 Boosting

An alternative approach to the kernel methods is to learn useful features directly from the input data. Wit this, we can create what we call an adaptive basis function model (ABM), which is a model of the form

$$f\left(\boldsymbol{x}\right) = w_0 + \sum_{m=1}^{M} w_m \phi_m\left(\boldsymbol{x}\right), \tag{16}$$

where $\phi_m\left(\boldsymbol{x}\right)$ is the mth basis function, which is learned from data.

Boosting (Schapire and Freund 2012) is a greedy algorithm for fitting adaptive basis-function models of the form shown above where ϕ_m are generated by an algorithm called a weak learner or a base learner. One of the popular boosting algorithms is the *adaboost*.

For the data-set given at this link, determine the decision boundary by using adaboost. Run the the algorithm for 250 steps. Show the decision boundary at steps 10, 100, 200 and 250. For additional details, refer to `demo_boosting` code in the PMTK package associated with Murphy [1].

References

[1] K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT Press, 2012.