Report on Sparse Bayesian Learning and the Relevance Vector Machine

Deependra Chansoliya, Rohan Mahnot and Ramandeep Maan

Roll No : 180227, 180621, 180589

Abstract

This report describes how to use Bayesian methods to get sparse solutions to regression and classification problems using linear models. While this paradigm is fully universal, we use the relevance vector machine(RVM) to demonstrate the approach. RVMs are models with same practical structure as that of SVMs. It is shown that by using a Bayesian Learning algorithm, reliable prediction models can be generated which are use much less basis functions as opposed to the primitive SVMs and still providing a host of other benefits including the ability to use arbitrary basis functions.

Index Terms

Support Vector Machine, Relevance Vector Machine, Bayesian Framework

I. Introduction

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input vectors $\{x_n\}_{n=1}^{N}$ and target vectors $\{t_n\}_{n=1}^{N}$.The target could be real values (*regression*) or could be corresponding class labels (in *classification*). We use an algorithm to learn some mapping function $y(\mathbf{x})$from the input to the target. The process of learning is to determine the parameters of this function. A typical classifier $y(\mathbf{x})$ takes the form:

$$y(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{M} w_i\phi_i(\mathbf{x}) = \mathbf{w^T}\phi(\mathbf{x}) \tag{1}$$

That is, the weighted sum of non-linear basis functions $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))^T$ The goal of a good model is to generalize well from the training data to any data from the problem domain. This allows us to make accurate predictions in the future on data the model has never seen. However, due to noise or class overlaps in the data, our model can over fit. Over fitting is more likely with non-parametric and non-linear models that have more flexibility when learning a target function.

The paper discusses a Bayesian probabilistic framework for models of the form (1). The characteristic of this approach is that not only it generalizes well, it infers parameters $(w_0, w_1, \ldots, w_N)$ that are extremely sparse i.e. only a w_i parameters are non-zero. While there are a great many models for classification, the paper discusses a special model, 'Relevance Vector Machine'(RVM). Which itself is the Bayesian treatment of Support Vector Machines(SVM). The objective of the support vector machine algorithm is to find a hyper-plane in an N-dimensional space that distinctly classifies the data points. It makes predictions based on the following functions:

$$y(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x}; \mathbf{x_i}) + w_0 \tag{2}$$

where $K(\mathbf{x}; \mathbf{x_i})$ is a kernel function, effectively defining one basis function for each example. The quintessential feature of SVM is that, for classification, its target function attempts to minimise a measure of error on the training set while simultaneously maximising the *margin* between the two classes. This generalises the model well by reducing the extent of over fitting and results in a sparse model dependent only on a subset of kernel functions. That is only a few parameters are non-zero, rest are zero and hence the reduces over fitting. Though popular, SVM suffer though some drawbacks:

- The number of support vectors increase linearly with the data-set size. SVM algorithm is not suitable for large data-sets.
- As the support vector classifier works by putting data points, above and below the classifying hyper-plane, there is no probabilistic explanation for the same.
- It is necessary to evaluate margin trade off parameter *C* via cross- validation process which is computationally complex.
- The kernel functions must satisfy Mercer's conditions.

To overcome the aforementioned drawbacks, we use Relevance Vector Machine(RVM). RVM is Bayesian Treatment of SVM. RVM adopts a fully probabilistic model for classification. It defines priors over the model weights governed by a set of hyper parameters. Sparsity is achieved because a lot of distributions that we use peak around or have maxima at zero.

II. Sparse Bayesian Learning for Regression

A. Model Specification

Given a data set of input-target pairs $\{\mathbf{x_n}, t_n\}_{n=1}^{N}$, we consider the scalar valued target functions and vector valued inputs. We use generative data modelling to model the data by adding additive Gaussian noise:

$$t_n = y(\mathbf{x_n}; \mathbf{w}) + \epsilon_n \tag{3}$$

where

$$\epsilon_n \sim \mathcal{N}(0,\, \sigma^2)$$

As the noise added is mean-zero Gaussian with variance σ^2, we can write $p(t_n|\mathbf{x}) = \mathcal{N}(t_n|y(\mathbf{x_n}),\, \sigma^2)$ where the function $y(\mathbf{x})$ is as defined in (2). We are assuming that t_n are independent, so the likelihood of the complete data-set is

$$p(\mathbf{t}|\mathbf{w}, \sigma^2) = (2\pi\sigma^2)^{-N/2} exp\{\frac{-||\mathbf{t} - \mathbf{\Phi w}||}{2\sigma^2}\} \tag{4}$$

where $\mathbf{t} = (t_1, t_2, \ldots, t_N)^T, \mathbf{w} = (w_0, w_1, \ldots, w_N)^T$ and Φ is the design matrix of order $N \times (N+1)$ As the number of parameters approach to be same as the number of training examples, the maximum likelihood estimate of $\mathbf{w}$ and σ^2 would lead to severe over-fitting. To avoid this, we can introduce a penalty term in the cost function. Here, we adopt a Bayesian framework, and 'constrain' the parameters by defining an explicit *prior* distribution over them.
For this we use mean-zero Gaussian prior distribution over $\mathbf{w}$

$$p(\mathbf{w}|\alpha) = \prod_{i=0}^{N} \mathcal{N}(w_i|0,\, \alpha_i^{-1}) \tag{1}$$

where α is a vector of N+1 *hyperparameters*.

B. Inference

By Bayes' rule, the posterior distribution over all the unknown parameters is given as

$$p(w, \alpha, \sigma^2|t) = \frac{p(t|\alpha, w, \sigma^2)p(w, \alpha, \sigma^2)}{p(t)} \tag{7}$$

This posterior is used to evaluate the predictive distribution

$$\int p(t_*|\alpha, \sigma^2)p(\alpha, \sigma^2|t)\, d\alpha\, d\sigma^2 \tag{8}$$

where t_* is the target of the new test point x_*.
The posterior may not always be computed directly using (7) because it is not always possible to

write the term $p(t) = \int p(t|\alpha, w, \sigma^2)p(w, \alpha, \sigma^2)\, dw\, d\alpha\, d\sigma^2$ in closed form. However, the posterior can be computed by decomposing the posterior as

$$p(w, \alpha, \sigma^2|t) = p(w|t, \alpha, \sigma^2)p(\alpha, \sigma^2|t) \tag{9}$$

The posterior distribution over the weights is given by :

$$p(w|t, \alpha, \sigma^2) = \frac{p(t|w, \sigma^2)p(w|\alpha)}{p(t|\alpha, \sigma^2)} \tag{10}$$

$$= (2\pi)^{-(N+1)/2}|\Sigma|^{-1/2}exp(-\frac{1}{2}(w-\mu)^T\Sigma^{-1}(w-\mu)) \tag{11}$$

where the posterior covariance and mean are

$$\Sigma = (\sigma^{-2}\Phi^T\Phi + A)^{-1} \tag{12}$$

$$\mu = \sigma^{-2}\Sigma\Phi^T t \tag{13}$$

$$A = diag(\alpha_0, \alpha_1, ..., \alpha_N)$$

The hyperparameter posterior $p(\alpha, \sigma^2|t)$ can be approximated using its MAP estimate *i.e.* $\alpha_{MP}, \sigma^2_{MP}$. This is done because the functions generated from the posterior mode values are similar to those obtained from sampling from the actual posterior distribution. However, this doesn't mean that the delta function is an exact estimate of the posterior.We only desire that the function obtained using the modes to be a good approximate of the function obtained using the samples of the probability distribution *i.e.*

$$\int p(t_*|\alpha, \sigma^2)\delta(\alpha_{MP}, \sigma^2_{MP})\, d\alpha\, d\sigma^2 \approx \int p(t_*|\alpha, \sigma^2)p(\alpha, \sigma^2|t)\, d\alpha\, d\sigma^2 \tag{14}$$

The task of relevance vector learning has been reduced to optimization of hyperparameter posterior mode *i.e.* maximization of $p(\alpha, \sigma^2|t) \propto p(t|\alpha, \sigma^2)p(\sigma^2)$ with respect to α and β. This can be computed as:

$$\begin{aligned} p(t|\alpha, \sigma^2) &= \int p(t|w, \sigma^2)p(w|\alpha)\, dw \\ &= (2\pi)^{-N/2}|\sigma^2\mathbf{I} + \mathbf{\Phi A^{-1}\Phi^T}|^{-1/2}exp\{-\frac{1}{2}\mathbf{t^T}(\sigma^2\mathbf{I} + \mathbf{\Phi A^{-1}\Phi^T})^{-1}\mathbf{t}\} \end{aligned} \tag{15}$$

C. Optimising the Hyperparameters

Closed form solutions of (15) can't be obtained. The values of α and σ^2 are estimated iteratively. For α, differentiate (15) and set to zero. After rearrangement, the equation reduces to:

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2}$$

where μ_i is the i^{th} posterior mean weight from (13) and γ_i is defined by:

$$\gamma_i = 1 - \alpha_i \Sigma_{ii} \tag{2}$$

For variance of noise σ_2, differentiating (15) partially and setting it to zero gives

$$(\sigma^2)^{new} = \frac{\|\mathbf{t} - \mathbf{\Phi}\mu\|^2}{N - \Sigma_i \gamma_i}$$

where N is the number of training examples.
The algorithm continues by repeating these steps and updating posterior $\mathbf{\Sigma}$ and μ from (12) and (13) until they converge. A suitable convergence criteria needs to be chosen for the same. In practice, many of the α_i achieve high values(and are indistinguishable from infinity for the machines). From (11) this implies that $p(w_i|\mathbf{t}, \alpha, \sigma^2)$ starts attaining maximum at zero. This provides some certainty that w_i are zero and hence sparsity is achieved.

D. Making Predictions

III. Sparse Bayesian Classification

For classification, an additional approximation step is needed. In two class classification, the posterior probability is predicted as to which class $\mathbf{x}$ should belong. Linear model $y(\mathbf{x})$ is passed through sigmoid function $\sigma(y) = \frac{1}{(1+e^{-y})}$. Bernoulli distribution is chosen for $P(t|\mathbf{x})$. Likelihood is given as

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma\{y(\mathbf{x_n};\mathbf{w})\}^{t_n}[1 - \sigma\{y(\mathbf{x_n};\mathbf{w})\}]^{1-t_n}, \tag{23}$$

targets t_n are either 0 or 1.There is no noise variance here. Here the likelihood and prior are non conjugate to each other so we cannot carry out integration for finding $p(\mathbf{w}|\mathbf{t}, \alpha)$ or $P(\mathbf{t}|\alpha)$. We have to use approximation methods available. One such method is Laplace's method.
For α fixed most probable $\mathbf{w}_{MP}$ weights are found which provide the mode's location of posterior distribution.
$P(\mathbf{w}|\mathbf{t}, \alpha) \propto P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\alpha)$ so we can find max over $\mathbf{w}$ of

$$log\{P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\alpha)\} = \sum_{n=1}^{N}[t_n log y_n + (1-t_n)log(1-y_n)] - \frac{1}{2}\mathbf{w^T A w}, \tag{24}$$

where $y_n = \sigma\{y(\mathbf{x_n};\mathbf{w})\}$. In Laplace's method posterior is approximated by a Gaussian with mean as posterior mode w_{MP} and inverse of co-variance is negative of Hessian at its mode

$$\Sigma^{-1} = -\frac{\partial^2}{\partial \mathbf{w} \partial \mathbf{w^T}}(log\{P(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\alpha)\}), \tag{25}$$

which gives

$$\mathbf{\Sigma} = (\mathbf{\Phi^T B \Phi} + \mathbf{A})^{-1}, \tag{26}$$

$$\mu = \mathbf{w_{MP}} = \Sigma\mathbf{\Phi^T B t}, \tag{27}$$

where $\mathbf{B} = diag(\beta_1, \beta_2, \ldots, \beta_w)$ is a diagonal matrix with $\beta_n = \sigma\{y(\mathbf{x_n})\}[1 - \sigma\{y(\mathbf{x_n})\}]$
Using $\mathbf{\Sigma}$ and μ of Gaussian approximation the hyper parameter alpha are updated using (16) similar to case of Regression.

IV. Relevance Vector Examples

A. Relevance Vector Regression : 'sinc' function

The function sinc(x) = $\frac{sinx}{x}$ has been a popular choice to illustrate regression with SVMs. In classification, we have a decision boundary, however, here ϵ-insensitive region is used, i.e. a region in $\pm\epsilon$ around the function where errors are not penalized. The support vectors lie on the edge of,or outside this region.
For example, 36 support vectors are used for approximating sinc(x) function using 100 uniformly spaced samples for ϵ = 0.01 done using a uni-variate 'linear spline' kernel:

$$K(x_m, x_n) = 1 + x_m x_n + x_m x_n min(x_m, x_n) - \frac{x_m + x_n}{2} min(x_m, x_n)^2 + \frac{min(x_m, x_n)^3}{3} \tag{5}$$

In RVM, data is modelled using the basis functions using the same kernel in (5) For comparison, SVM, sinc function is modelled with relevance vector machines with a constant noise variance, 0.01^2. After that α is re-estimated. The constant noise variance is similar to setting the ϵ sensitivity.
Using this fixed σ^2, only 9 relevance vectors are required.In the case of RVM, largest error is 0.007 and for SVM,it is 0.0100. Hence, other than gain in accuracy, we also have a more sparsity. As a more practical problem, consider the case where uniform noise in [-0.2, 0.2] is added to the targets. The trained RVM uses 6 relevance vectors, compared to 29 for SVM. Here, the standard deviation is 0.0245 for RVM while for SVM its 0.0291.

B. Relevance Vector Classification : Ripley's Synthetic Data

A relevance vector classifier is compared to its support vector counterpart, using a 'Gaussian' kernel which is defined as

$$K(x_m, x_n) = exp(-r^{-2}\|\mathbf{x_m} - \mathbf{x_n}\|^2), \tag{30}$$

Using SVM and RVM we observe that the relevance vectors are some distance from the decision boundary (in x-space). There is no proof that the use of either boundary-located or prototypically-located functions is 'correct' in any sense both are correct in their own way.

C. Extnesions

In this section we discuss the ability to utilise arbitrary basis functions, and the facility to directly 'optimise' parameters within the kernel specification, such as those which moderate the input scales. That is to choose the type of kernel function and also to find the appropriate values for any parameters involved.

Consider the problem of estimating the bi-variate function:

$$y(x_1, x_2) = sinc(x_1) + 0.1x_2 \tag{3}$$

using 100 samples with noise $\mathcal{N}(0,\, 0.1^2)$ There are problems with direct application of a relevance or support vector model to this data are:

- The objective function is linear in x_2 but it will be modelled poorly using non-linear functions.
- The non-linearity in the target function is independent of x_2 due to which, it will only add noise and this will be reflected in the final approximator.

These 2 aspects make it difficult for the function to learn and the respective relevance or support vector approximation.

To improve upon the results mentioned in the paper [1], we introduce Bayesian modifications to our relevance vector model.

The second modification is to directly optimise the likelihood with respect to the kernel's scaling parameters. so parameters η_1 and η_2 are introduced such that, the kernel equation becomes:

$$\mathbf{K}(\mathbf{x_m}, \mathbf{x_n}) = \mathbf{exp}\{-\boldsymbol{\eta}_\mathbf{1}(\mathbf{x_{m1}} - \mathbf{x_{n1}})^\mathbf{2} - \boldsymbol{\eta}_\mathbf{2}(\mathbf{x_{m2}} - \mathbf{x_{n2}})^\mathbf{2} \tag{4}$$

To estimate these parameters, at each iteration of the hyperparameter updates (16) a cycle of maximisation of the marginal likelihood (15) with respect to η_1 and η_2 was performed (using a gradient-based method).

Post these 2 modifications, the final RVM approximation function is more accurate.

Although convincing, there are a few limitations of this approach:

- Due to 2 hyperparameters, α and η, the the algorithm doesn't specify which one to optimise
- The optimisation is computationally very complex.

REFERENCES

[1] Michael E. Tipping. *Sparse Bayesian Learning and the Relevance Vector Machine*. https://www.jmlr.org/papers/volume1/tipping01a/tipping01a.pdf