# Task 3. Sparse Supervised Learning: Relevance Vector Machine

Garoe Dorta-Perez
CM50246: Machine Learning and AI

January 16, 2015

## 1 Introduction

Support vector machines (SVM) are a popular non-probabilistic choice for classification problems. A technique that uses the same intuition, is the Relevance Vector Machine (RVM), however, they use a full Bayesian approach with a prior that encourages sparseness.

## 2 The problem

First a dual model is derived, we begin by modelling the data with a normal distribution, as shown in Equation 1, where $w$ is a one dimensional array with the world state, $x_i$ is a data point, $\phi$ are the parameters of a linear function of the data and $\sigma$ is the standard deviation. We now apply a transformation on the parameters as shown in Equation 2, so $\phi$ is now a weighted sum over the observed data points, where $\psi$ is a vector with the weights. If the data dimensionality is larger than the amount of data points, the weighted sum can be computed in less time, as it has less terms than the original linear dependency.

$$Pr(w_i|\mathbf{x}_i) = Norm_{x_i}[\phi^T\mathbf{x}_i, \sigma^2], \tag{1}$$

$$\phi = \mathbf{X}\psi. \tag{2}$$

when we are using the dual parameters $\psi$, sparseness can be encouraged in the model by including the prior defined in Equation 3, where $I$ is the number of data points, *Stud* is a Student's t-distribution with $\nu$ degrees of freedom and $\Gamma$ is the Gamma function. Using a product of t-distributions produces the desired sparseness since the areas with higher probability density are in the origin and along the axis.

$$\begin{aligned} Pr(\psi) &= \prod_{i=1}^{I} Stud_{\psi_i}[0, 1, \nu], \\ &= \prod_{i=1}^{I} \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{\psi_i^2}{\nu}\right)^{-(\nu+1)/2}. \end{aligned} \tag{3}$$

A t-distribution is not conjugate to a Normal distribution, so there is no simple closed form solution for the posterior. To overcome this issue, we will approximate the t-distributions by maximizing with respect to their hidden variables $h$, introduced in Equation 4, where $Gam_{h_i}$ is the Gamma distribution for the hidden variable $h_i$. This leads to the marginal likelihood shown in Equation 5, where $\mathbf{I}$ is the identity matrix, $\mathbf{X}$ is a matrix with the data points, $\mathbf{H}$ is a matrix with all the hidden variables, $D$ is the data dimensionality and $\mathbf{w}$ is a one dimensional array with the world state.

$$Pr(\psi) = \prod_{i=1}^{I} \int Norm_{\mathbf{w}}[0, 1/h_i] Gam_{h_i}[\nu/2, \nu/2] dh_i, \tag{4}$$

$$Pr(\mathbf{w}|\mathbf{X}, \sigma^2) \approx \max_{\mathrm{H}} \left[ Norm_{\mathbf{w}} \left[ \mathbf{0}, \mathbf{X}^T \mathbf{X} \mathbf{H}^{-1} \mathbf{X}^T \mathbf{X} + \sigma^2 \mathbf{I} \right] \prod_{d=1}^{D} Gam_{h_d} \left[ \nu/2, \nu/2 \right] \right]. \tag{5}$$

The optimization is performed in three steps:

1. Optimize the marginal likelihood with respect to the hidden variables, using Equation 6.

$$h_i^{new} = \frac{1 - h_i \sum_{ii} + \nu}{\mu_i^2 + \nu}. \tag{6}$$

2. Update $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, using Equation 7.

$$\begin{aligned} \boldsymbol{\mu} &= \frac{1}{\sigma^2} \mathbf{A}^{-1} \mathbf{X}^T \mathbf{X} \mathbf{w}, \\ \boldsymbol{\Sigma} &= \mathbf{A}^{-1}, \\ \mathbf{A} &= \frac{1}{\sigma^2} \mathbf{X}^T \mathbf{X} \mathbf{X}^T \mathbf{X} + \mathbf{H}. \end{aligned} \tag{7}$$

3. Optimize the marginal likelihood with respect to the variance parameter, using Equation 8.

$$(\sigma^2)^{new} = \frac{1}{\mathbf{I} - \sum_i (1 - h_i \sum_{ii})} \left( \mathbf{w} - \mathbf{X}^T \mathbf{X} \boldsymbol{\mu} \right)^T \left( \mathbf{w} - \mathbf{X}^T \mathbf{X} \boldsymbol{\mu} \right). \tag{8}$$

After the training process, we only take the data points whose hidden variable $h_i$ is smaller than a threshold, larger one means a small $\psi_i$, and hence no significance in the weighted sum. In the implementation, a Gaussian kernel $k[x_i, x_j]$ is used, as shown in Equation 9, where $\lambda$ controls the scale of the output. Instead of calculating the inner products $\mathbf{X}^T \mathbf{X}$ a kernel matrix is computed. This is more efficient than calculating the inner product, specially for highly dimensional data.

$$k[x_i, x_j] = exp\left[-\frac{1}{2}\left(\frac{(x_i - x_j)^T(x_i - x_j)}{\lambda^2}\right)\right]. \tag{9}$$

Since our world states $\mathbf{w_i}$ are multivariate, a separate regressor is used for each dimension of $\mathbf{w_i}$. However, the hidden variables matrix $\mathbf{H}$ is shared among all of them, in order to conserve a sparse structure.

# 3 Results

In Figures 1 and 2 we show results for a font data set. The RVM is trained with pictures of $n$ in different fonts and their corresponding $m$ in the same font. For testing, $n$ with new fonts are used.

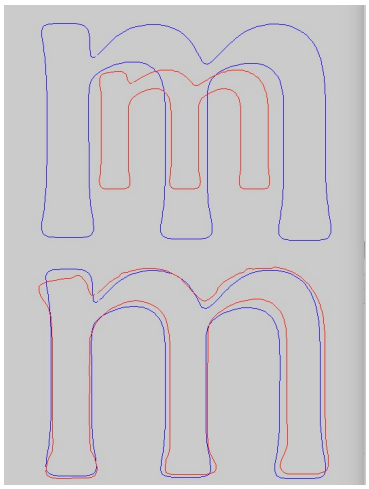

Figure 1: Blue is ground truth and red is RVM result. Top image corresponds to a kernel $\lambda = 2$, bottom image has $\lambda = 7$.

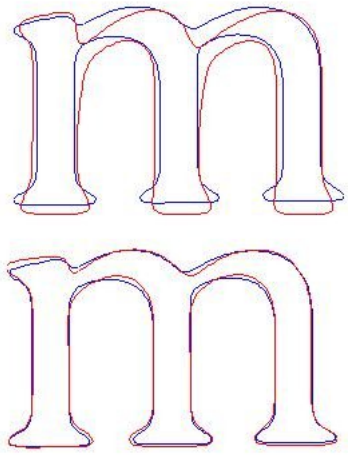

Figure 2: Blue is ground truth and red is RVM result. Top image corresponds $\nu = 0.001$, bottom image has $\nu = 1$

In Figure 1 we see the effects of changing the value of $\lambda$. Since it controls the scale, we can see how the $m$ varies in size. In Figure 2, results for different values of $\nu$ are shown. Smaller $\nu$ encourages more sparsity, and so less relevance vectors been used. For the top image 5 relevance vectors are used, while for the bottom one 10 vectors are used. Fewer vectors lead to a more general and sparse model used, and the error in the prediction increases as we take less relevance vectors. Our dataset had only 10 training samples, so there is not much space for sparse learning, as RVM sparseness advantages are more evident with bigger datasets.