# *Automatic Relevance Determination*

Elia Liitiäinen (eliitiai@cc.hut.fi)

Time Series Prediction Group
Adaptive Informatics Research Centre
Helsinki University of Technology, Finland

October 24, 2006

# *Introduction*

- Automatic Relevance Determination is a classical method based on Bayesian interference.
- In this presentation we show how it can be applied to Least Squares Support Vector Machines.
- As a result we get a method for estimating hyperparameters and choosing inputs.

# Outline

# *Least Squares Support Vector Machines*

- We assume that the dataset $(y_i, x_i)_{i=1}^{N}$ is available.
- The inputs $(x_i)$ are in $\Re^n$ for a finite $n$.
- Assume $\phi : \Re^n \rightarrow \mathcal{H}$ is a mapping to some high (infinite) dimensional space.
- We model the outputs $y$ by $y = \omega^T \phi(x) + b$.
- As is common, we won't do totally rigorous mathematical analysis.

# *The Cost Function*

- LS-SVM differs from SVM in the cost function:

$$\mathcal{I} = \gamma E_1 + \xi E_2 = \frac{\gamma}{2}\|\omega\|^2 + \frac{\xi}{2}\sum_{i=1}^{N} e_i^2, \tag{1}$$

  where $e_i = (y_i - \omega^T \phi(x_i) - b)$.
- Note that we use two hyperparameters to get a Bayesian interpretation.

# *Optimization of the Cost*

- The mapping $\phi$ is very hard to handle as such.
- The solution: we require $\phi(x)^T\phi(y) = K(x, y)$ for some kernel $K$.
- The kernel often contains an additional parameter.
- By a simple manipulation with the Lagrangian, it can be seen that the approximator becomes

$$y(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i) + b \tag{2}$$

  with the condition $\sum_{i=1}^{N} \alpha_i = 0$ and a $L^2$ regularized cost function.
- Thus we have ended up with a well-known Gaussian process model.
- Solving for $\alpha_i$ is elementary (this is in fact a form of RBF).

# *Bayesian Interference*

- We skip the philosophical questions behind Bayesian methods.
- Automatic Relevance Determination is based on Bayesian interference on three levels.
- ARD is a classical method and can be applied to many other models (MLP,RBF...).
- In what follows, $\mathcal{H}$ denotes the model and $D$ is the data.
- We assume no prior knowledge of the problem which means that flat priors are used whenever necessary.

# *First Level of Interference*

- Assume that the sample $(x_i, y_i)$ is iid. Recall the cost

$$\mathcal{I} = \gamma E_1 + \xi E_2. \tag{3}$$

- In the first level the hyperparameters $\gamma$ and $\xi$ are assumed to be fixed.
- We assume the prior $p(w) \sim \exp(-\gamma \|w\|^2)$.
- For the observations we assume $p(y_i|x_i, \omega, b, \xi, \mathcal{H}) \sim \exp(-\frac{\xi}{2} e_i^2)$.
- This is a model with a Gaussian prior and a Gaussian noise model.

# *Interference on the First Level*

- With the assumptions of the previous slide, we get

$$p(\omega, b|D, \gamma, \xi, \mathcal{H}) \sim \exp(-\mathcal{I}(D, \gamma, \xi, \omega, b)) \tag{4}$$

- It follows that given the hyperparameters, finding the maximum likelihood for $p(\omega, b|D, \gamma, \xi, \mathcal{H})$ is equivalent to minimizing the cost $\mathcal{I}$.

# Second Level of Interference

- In the second level we examine $p(\xi, \gamma | D, \mathcal{H})$.
- We write

$$p(\xi, \gamma | D, \mathcal{H}) \sim \int p(D|w, b, \mathcal{H}) p(w, b|\xi, \gamma, \mathcal{H}) p(\xi, \gamma|\mathcal{H}) dwdb$$

- We assume a non-informative prior for the hyperparameters.
- This can be solved in closed form. Thus no approximation is needed on the second level.

# *The Cost Function on the Second Level (1)*

- Using the previously derived formula, we get

$$p(\xi, \gamma | D, \mathcal{H}) \sim \frac{\gamma^{n_f/2} \xi^{N/2}}{|H|^{-1/2}} \exp(-\mathcal{I}(\omega_{MAP}, b_{MAP})). \quad (5)$$

- Here $H$ is the Hessian of the cost function and $n_f$ is the dimension of the space in which $\phi$ maps the inputs.
- Typically $n_f >> 1$ and the Hessian $H$ is not available as such. However, it turns out that this is not a problem.

# *The Cost Function on the Second Level (2)*

- By recalling the condition $\phi^T(x_i)\phi(x_j) = K(x_i, x_j)$, it is possible to derive a maximum likelihood cost function for the hyperparameters.
- The compute a value of the cost, a first level optimization must be done together with solving the eigenvalues of the so-called centered Gram matrix.
- The optimization problem is one dimensional.
- The derivation in the paper can certainly be done without the SVM context.

# *The Third Level of Interference*

- Recall that by $\mathcal{H}$ we denoted the model structure (including kernel parameters, selected inputs).
- In the third level we write (assuming non-informative priors)

$$\begin{aligned} p(D|\mathcal{H}) &= \int p(D|\gamma,\xi,\mathcal{H})P(\xi,\gamma|\mathcal{H})d\xi d\gamma \\ \sim \quad & p(D|\gamma_{MAP},\xi_{MAP},\mathcal{H})D_\gamma D_\xi . \end{aligned} \tag{6}$$

- The terms $D_\gamma$ and $D_\xi$ are the second derivatives of the second level cost function at the optimum.
- All the approximations made are well-known.

# *Input Selection*

- Now that we can evaluate the evidence $p(D|\mathcal{H})$ of models, input selection is easy.
- A combination of inputs is evaluated by doing the three level of interference to calculate kernel parameters and hyperparameters.
- Scaling of input variables is implemented in the same way.
- All this is already done in the LS-SVM toolbox.

# *Practical Point of View*

- The method seems too heavy for many applications (this holds to LS-SVM in general).
- In the course we will use second level interference.
- An alternative to ARD is to use LOO or use them together.
- LOO has a computational cost of the same order.

# *Experiment*

- We examine a linear model with ten Gaussian inputs and Gaussian noise.
- Backward selection with ARD is made.

# *Results*

- The first experiment was done with noise 15% of the output.
- The second experiment was done with noise 30% of the output.
- The first experiment was solved optimally. The results of the second are in the figure.

# *Results (2)*

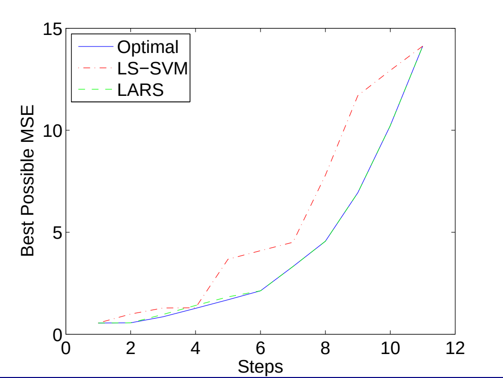


*Figure:* The best possible MSE for backward selection with optimal inputs, ARD chosen inputs and LARS chosen inputs.

# *Conclusion*

- ARD is a classical method for choosing model parameters.
- In this presentation we showed how to use it for LS-SVM.
- The resulting algorithm is heavy to calculate but fully automatic.
- In the project work we will combine ARD with grid-search for hyperparameter estimation.