Sparsity

Will Penny

24th March 2011

Relevance Vector Regression

Relevance Vector Regression (RVR) comprises a linear regression model (Tipping, 2001)

$$y(m) = \sum_{n=1}^{d} K(x_m, x_n) w_n + e(m)$$

where $m = 1..d$, $n = 1..d$ index d data points, K is a kernel or basis function, and w are regression coefficients. The independent variable, x, is uni- or multi-variate and the dependent variable y is univariate.

This can be written as the usual General Linear Model

$$y = Xw + e$$

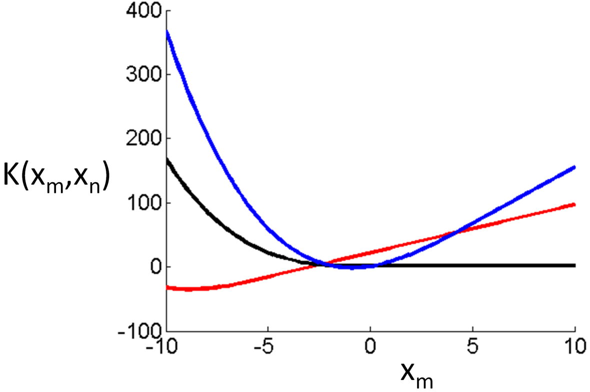
with $[dx1]$ data vector y, known $[dxp]$ design matrix X and p regression coefficients. We have $X(m, n) = K(x_m, x_n)$, $p = d$ (or $p = d + 1$ including offset term). The noise, e, is zero mean with isotropic precision λ_y.

Kernel

For example, a univariate linear spline kernel is given by

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

Three splines at $x_n = -5$ (red), $x_n = 0$ (black) and $x_n = 5$ (blue).



Prior

RVR is a Bayesian method with prior (Tipping, 2001)

$$p(w) = \prod_{i=1}^{p} \mathrm{N}(w_i; 0, \lambda_w(i)^{-1})$$

That is, each regression coefficient w_i has prior precision $\lambda_w(i)$.

This sort of prior, with a precision parameter for every regression coefficient is an example of an Automatic Relevance Determination (ARD) prior (Mackay, 1994).

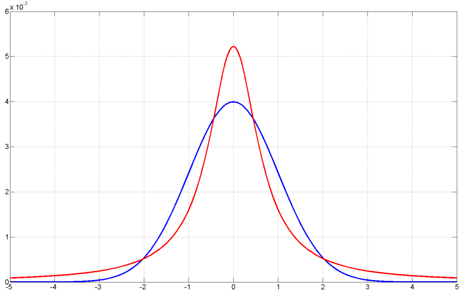
Inference in this model leads to irrelevant predictors being automatically removed from the model.

Prior

The implicit prior over each regression coefficient is

$$p(w_i) = \int p(w_i|\lambda_w(i))p(\lambda_w(i))dw_i$$

For $p(\lambda_w(i))$ given by a (constrained) Gamma density, $p(w_i)$ is a t-distribution, which is sparser than a Gaussian.



Inference

Inference in this model is very similar to the Empirical Bayes method for isotropic covariances (previous lecture). In the E-step we compute a posterior over regression coefficients

$$\begin{aligned} p(w|\alpha, Y) &= \mathrm{N}(w; m, S) \\ S^{-1} &= \lambda_y X^T X + \mathrm{diag}(\lambda_w) \\ m &= \lambda_y S X^T y \end{aligned}$$

In the M-step, we first compute

$$\gamma_i = 1 - \lambda_w(i) S_{ii}$$

where S_{ii} is the ith diagonal element of the posterior covariance matrix. γ_i is approximately unity if the ith parameter has been determined by the data and zero if determined by the prior.

M-Step

The hyperparameters are then updated as

$$\frac{1}{\lambda_w(i)} = \frac{m_i^2}{\gamma_i}$$
$$\frac{1}{\lambda_y} = \frac{e_y^T e_y}{d - \sum_i \gamma_i}$$

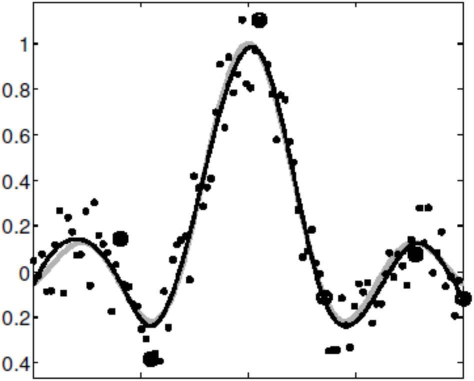
where the prediction error is

$$e_y = y - Xw$$

The learning algorithm then proceeds by repeated application of the E and M steps. Regression coefficients for which $\lambda_w(i)$ becomes very large are removed from the model, as are the corresponding columns of X. The remaining columns are referred to as relevance vectors.

Sinc Example

Tipping (2001) first generated $n = 1..100$ data points x_n and corresponding y_n values from the sinc function $y_n = sin(x_n)/x_n$ and added noise. He used the linear spline kernel. RVR found 6 relevance vectors.



Bottleneck in algorithm is computation of posterior covariance. See Tipping and Faul (2003) for more efficient version.

Visual Coding

For a 2D image V which is $[N_1 \times N_2]$ pixels

$$\begin{aligned} y &= vec(V) \\ &= V(:) \end{aligned}$$

Each image is modelled as a linear superposition of basis functions

$$y = Wx + e$$

with $Cov(e) = \lambda_y I$. The length of y is $d = N_1 N_2$. We have p basis functions.

The ith column of W contains the ith basis function, and $x(i)$ the corresponding coefficient. Different images, y, will be coded with a different set of coefficients, x. The basis functions W will be common to a set of images.

Visual Coding

We can also write

$$y = \sum_{i=1}^{p} w_i x_i + e$$

If there are d image elements then for $p > d$ we have an overcomplete basis. Usually $p < d$.

We wish to learn both w_i and x_i. If w_i were fixed (eg assume wavelets) then we can use ARD to select appropriate bases (Flandin et al 2007).

ML Learning

The likelihood is given by $p(y|W, x)$. We can learn both W and x using gradient ascent of the likelihood The ML estimate is given by

$$W_{ML} = \arg\max_{W} p(y|W, x)$$

Because the maxima of $\log x$ is the same as the maximum of x we can also write

$$W_{ML} = \arg\max_{W} L(W, x)$$

where

$$L = \log p(y|W, x)$$

is the log likelihood.

Learning basis functions

For the *i*th basis function

$$\tau_w \frac{dw_i}{dt} = \frac{dL}{dw_i}$$

This gives

$$\tau_w \frac{dw_i}{dt} = \lambda_y (y - Wx) x_i$$

which is simply the Delta rule (previous lecture).

Learning activations

For the activations

$$\tau_x \frac{dx}{dt} = \frac{dL}{dx}$$

This gives

$$\tau_x \frac{dx}{dt} = \lambda_y (W^T y - W^T W x)$$

This has the standard ML solution

$$x_{ML} = (W^T W)^{-1} W^T y$$

These dynamics can be implemented in two different ways in terms of neural circuits using either (i) Recurrent Lateral Inhibition or (ii) Predictive Coding.

Recurrent Lateral Inhibition

We have

$$\tau \frac{dx}{dt} = \lambda_y (W^T y - W^T W x)$$

The update for the ith activation can be written as

$$\tau \frac{dx(i)}{dt} = \lambda_y (x_{bu}(i) - x_{lat}(i))$$

where the bottom up and lateral terms are

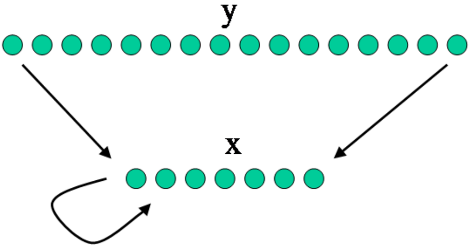
$$\begin{aligned} x_{bu} &= Uy \\ x_{lat} &= Vx \end{aligned}$$

and $U = W^T, V = W^T W$. V_{ij} is the strength of the recurrent lateral connection from unit j to unit i. Learning acts so as to match bottom up and lateral predictions.

Recurrent Lateral Inhibition

The update for the ith activation can be written as

$$\tau \frac{dx(i)}{dt} = \lambda_y (x_{bu}(i) - x_{lat}(i))$$



where the bottom up and lateral terms are

$$\begin{aligned} x_{bu} &= Uy \\ x_{lat} &= Vx \end{aligned}$$

where V_{ij} is the strength of the recurrent lateral connection from unit j to unit i.

Receptive versus projective fields

The top-down or generative weights are W as

$$\hat{y} = Wx$$

W are the projective fields.

The bottom-up or recognition weights are U as

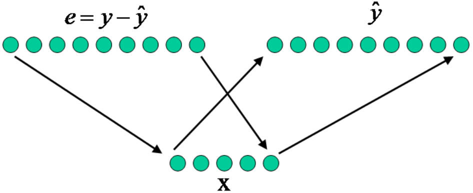
$$x_{bu} = Uy$$

U are the receptive fields.

We have $U = W^T$.

Predictive Coding Architecture

If first layer units are split into two pools (i) one for predictions from second layer and (ii) for prediction errors which are propagated back to the second layer



then activations are then driven by purely bottom up signals

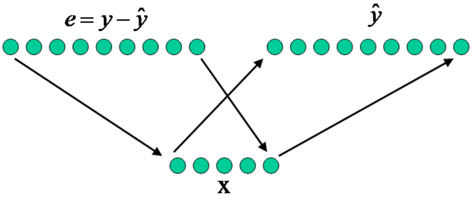
$$
\begin{aligned}
\tau \frac{dx}{dt} &= \lambda_y W^T (y - Wx) \\
&= \lambda_y W^T e
\end{aligned}
$$

For the ith activation unit we have simply

$$
\tau \frac{dx(i)}{dt} = \lambda_y \sum_j W_{ji} e_j
$$

There is no need for lateral connectivity.

Predictive Coding



Moreover, if the bottom up signals are prediction errors then Delta rule learning of basis functions (synapses)

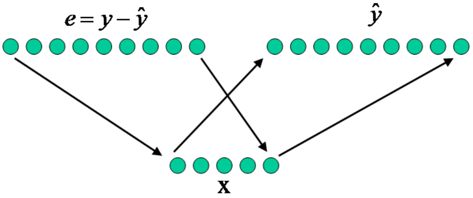
$$\tau \frac{dw_i}{dt} = \lambda_y (y - Wx) x_i$$

is seen to correspond to simple Hebbian Learning

$$\tau \frac{dW_{ji}}{dt} = \lambda_y e_j x_i$$

where e_j is the jth prediction error and x_i is the output of the ith unit.

Hebbian Learning



Hebbian learning modifies connections between two units by an amount proportional to the product of the activations of those units - 'cells that fire together wire together'.

$$\tau \frac{dW_{ji}}{dt} = \lambda_y e_j x_i$$

where e_j is the jth prediction error (jth input to ith unit) and x_i is the output of the ith unit.

Sparse Coding

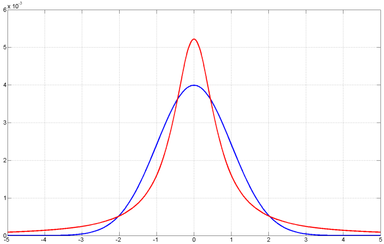
Olshausen and Field (1996) propose a sparse coding model of natural images. The likelihood is the same as before

$$p(y|W,x) = \mathrm{N}(Wx, \lambda_y I)$$

But importantly, they also define a prior over coefficients

$$p(x) = \prod_i p(x_i)$$

where $p(x_i)$ is a *sparse* prior. This can be any distribution which is more peaked around zero than a Gaussian.



This means we expect most coefficients to be small, with a few being particularly large.

MAP Learning

Again, we need to learn both W and x. The posterior density is given by Bayes rule

$$p(W, x|y) = \frac{p(y|W, x)p(x)}{p(y)}$$

The Maximum A Posterior (MAP) estimate is given by

$$W_{MAP} = \arg\max_{W} p(W, x|y)$$

Because the maxima of $\log x$ is the same as the maximum of x we can also write

$$W_{MAP} = \arg\max_{W} L(W, x)$$

where

$$L = \log[p(y|W, x)p(x)]$$

is the joint log likelihood.

Learning

The updates for the basis functions are exactly the same as before. For the activations we have

$$\tau \frac{dx}{dt} = \frac{dL}{dx}$$

This gives

$$\tau \frac{dx}{dt} = \lambda_y W^T e - \sum_i g(x_i)$$

where

$$g(x_i) = \frac{d \log p(x_i)}{dx_i}$$

is the derivative of the log of the prior. Olshausen and Field have used a Cauchy density

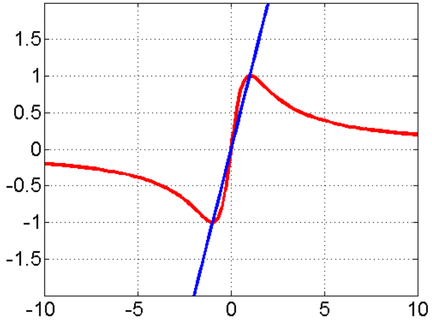
$$p(x) = \frac{1}{\pi(1 + x^2)}$$

Learning

This gives

$$\tau \frac{dx_i}{dt} = \lambda_y w_i^T e - g(x_i)$$

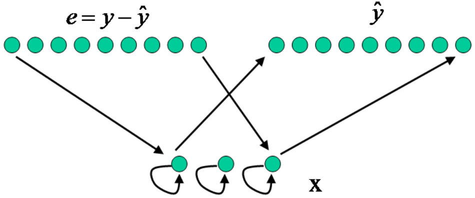
The figures shows $g(x_i) = x_i$ for Gaussian priors (blue) and $g(x_i) = 2x_i/(1 + x_i^2)$ for Cauchy priors (red)



Self-Inhibition

In terms of the neural implementation we must add *self-inhibition* to the activation units, which is linear for Gaussian priors and nonlinear for Cauchy priors

$$\tau \frac{dx_i}{dt} = \lambda_y w_i^T e - g(x_i)$$



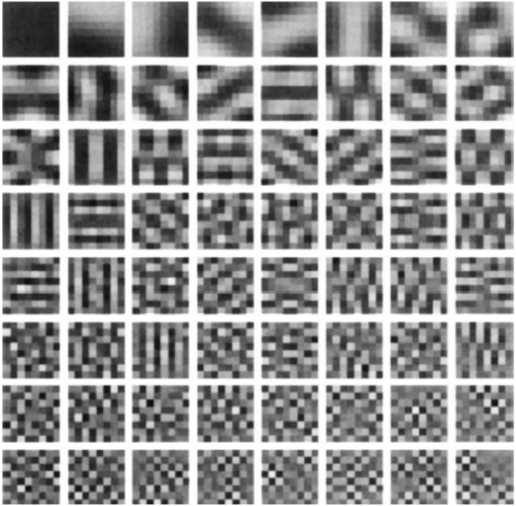
For Gaussian priors the amount of inhibition is proportional to the activation, whereas for Cauchy priors large activations are not inhibited.

Original Images

Ten images of natural scenes were low-pass filtered.

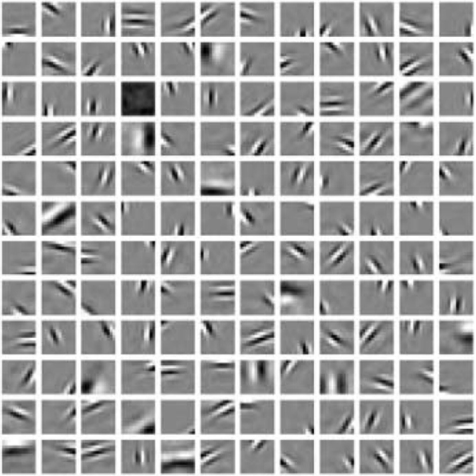
Principal Component Analysis

Receptive fields from PCA.



Receptive Fields from Sparse Coding

This produced receptive fields that are spatially localised, oriented and range over different spatial scales, much like the simple cells in V1.



References

C. Bishop (2006) Pattern Recognition and Machine Learning, Springer.

G. Flandin and W.D. Penny. NeuroImage, 34(3):1108-1125, 2007

D. Mackay (1995) Probable networks and plausible predictions. Network, IOPP.

D. Mackay (2003) Information Theory, Inference and Learning Algorithms. Cambridge.

B. Olshausen and D. Field (1996) Nature 381, 607-609.

M. Tipping (2001) Journal of Machine Learning Research, 211-214.

M. Tipping and A. Faul (2003) Proc 9th Workshop AI Stats, FL.