Learning from Data: Adaptive Basis Functions

Amos Storkey, School of Informatics

November 21, 2005

http://www.anc.ed.ac.uk/~amos/lfd/

Neural Networks

- Hidden to output layer - a linear parameter model
- But adapt the “features” of the model.
- Neural Network features pick particular directions in input space.
- But could use other features - eg localisation features

Radial Basis Functions

- Radial Basis Functions are also linear parameter models.
- Have localised features.
- But so far we have only considered fixed basis functions
- Instead could adapt the basis functions as we do with neural networks.
- The rest is just the same.

End of Lecture

- Well pretty much.
- But for completeness we can reiterate the process!
- Compare with Neural Networks
- Some pictures.
- Error functions.

Neural Networks

- Output of a node is nonlinear function of linear combination of parents.
- In other words a function of a projection to a particular direction

$$y_i = g_i\left(\sum_j w_{ij}x_j + \mu_i\right)$$

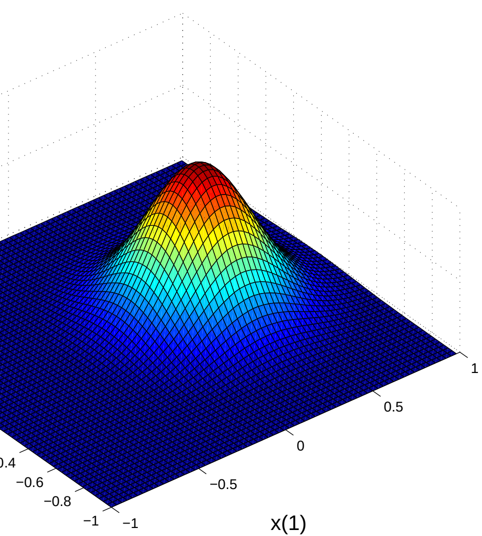
Radial Basis Function

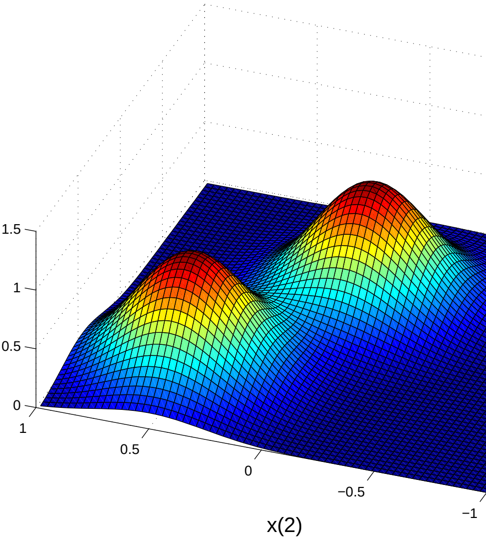
- Output of a node is a nonlinear function of the *distance* of the input from a particular point.
- Nonlinear function is usually decaying: hence it is a local model.

$$y(\mathbf{x}, \theta) = \sum_i w_i \phi_i(\mathbf{x}, \mathbf{b}^i)$$

  ϕ_i has parameters $\mathbf{b}^i$ and for radial basis functions is generally a function of $|\mathbf{x} - \mathbf{r}_i|$ for some centre $\mathbf{r}_i \subset \mathbf{b}^i$.
- Of course all the sums work for anything of this form, including radial basis functions, neural networks etc.
- Call general form adaptive basis functions. Only requirement is differentiability w.r.t parameters.

Radial Basis Functions





Error Functions

- Regression - sum squared error:

$$E_{train} = \sum_{\mu} (y^{\mu} - f(\mathbf{x}^{\mu}, \boldsymbol{\theta}))^2$$

- Classification:

$$E_{train} = -\sum_{\mu} (y^{\mu} \log f^{\mu} + (1 - y^{\mu}) \log(1 - f^{\mu}))$$

Regularisation and Initialisation

- Regularisation: width of basis functions determines smoothness. Could ensure basis width is not too small to prevent overfitting, or use a validation set to set the basis width.
- Initialisation: matters. Best try multiple restarts as with neural networks. Given basis initialisations, can get good initialisations for the weights in the regression case by treating it as a linear parameter model and solving.

Optimisation

- Can calculate all the derivatives just as before.
- Gather all the parameters together into a vector. Optimise using e.g. conjugate gradients.
- Example code is on the lecture notes.
- Another approach for regression involves iteration of solving the linear parameter model and updating the basis functions (see notes)

Comparison

- Radial basis functions give local models. Hence away from the data we get a prediction of zero. Does our data really tell us nothing about what happens in non-local regions? Even so should we really predict zero?
- Both RBF and MLP subject to local minima.
- Understanding the result is slightly easier for radial basis functions.

Committees and Error Bars

- Getting a prediction is one thing, but what about prediction uncertainty?
- We could get some gauge of uncertainty by looking at the variation in predictions across different learnt models.
- Use committees.

Committee Approach

- Pick a number of different models (could even be different starting points of the same model).
- Predict using the average prediction of the models.
- Get a measure of the confidence in the prediction by looking at the variance of each model prediction around the average prediction.

Better/Other Ways

- Bayesian methods. Calculate the posterior distribution of the parameters. Use that to obtain error bars in data space.
- Take the limit of an infinite number of Bayesian neural networks: gives Gaussian process models, where the non-linear prediction and error bar problem can be solved analytically.
- Look at dependence of the prediction on the training data, by resampling the training data: bootstrap and bagging.