Regression

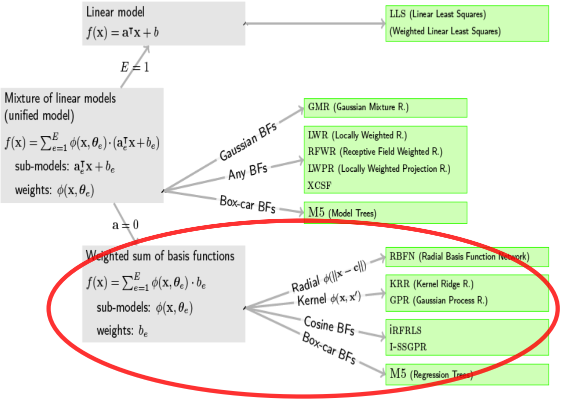
4. Batch non-linear projection methods

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

Radial Basis Function Networks

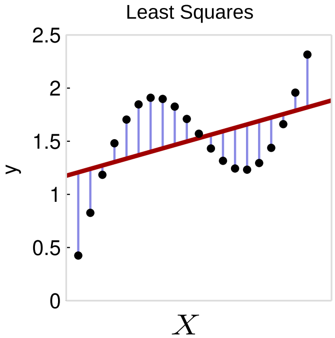
Reminder: Outline of methods



- Projecting the input space into a feature space using non-linear basis functions (shown with RBFNs)

Stulp, F. and Sigaud, O. (2015). Many regression algorithms, one unified model: A review. *Neural Networks*, 69:60–79.

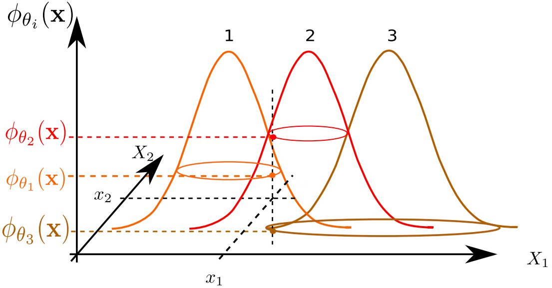
Least Square Projection Methods: framework



- With linear regression, we look for $\hat{f}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$
- This is not general enough for non-linear functions
- More general form: $\hat{f}(\mathbf{x}) = \sum_{e=0}^{E} w_e \cdot \phi_{\boldsymbol{\theta}_e}(\mathbf{x})$ with $\phi_{\boldsymbol{\theta}_0}(\mathbf{x}) = 1$
- This can be seen as projecting the input to a different space...
- ... where the latent function is linear

Bishop, C. M. (2007) *Pattern recognition and machine learning*. Springer Berlin/Heidelberg, Germany

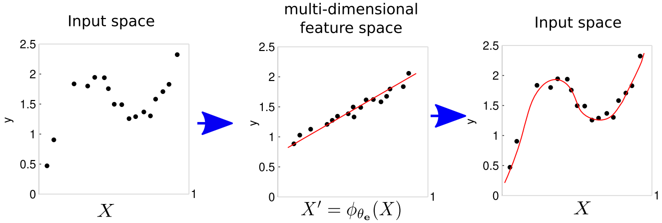
Understanding projection



- Given the functions $\phi_{\boldsymbol{\theta}_e}(.)$, each point $\mathbf{x}$ in the dataset $\mathbf{X}$ is projected into a point $\mathbf{x}' = \phi_{\boldsymbol{\theta}_e}(\mathbf{x})$
- The number E of functions determines the dimension of $\mathbf{x}'$
- The point $\mathbf{x} = (x_1, x_2)$ is projected to $\mathbf{x}' = (\phi_{\theta_1}(\mathbf{x}), \phi_{\theta_2}(\mathbf{x}), \phi_{\theta_3}(\mathbf{x}))$

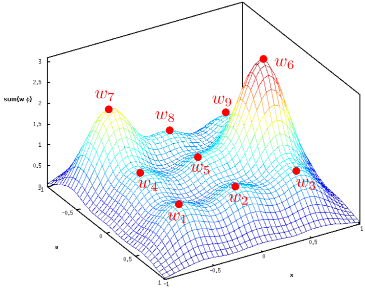
Cybenko, G. (1989) Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314

Basis Function Networks: finding the weights



- Now we have a dataset $\mathbf{X}', \mathbf{y}$
- We want to find the function $y = \hat{f}(\mathbf{x}') = \sum_{e=0}^{E} w_e \cdot \mathbf{x}'$
- This is a linear regression problem
- Thus we perform regression in the projected space
- The larger E, the smaller the error (perfect fit in the limit of $E \rightarrow \infty$)

Standard features: Gaussian basis functions



- The more features, the better the approximation
- … but the more expensive the computation
- If the features are given and constant, this is a linear architecture

Kernel Ridge Regression (KRR) = Kernel Regularised Least Squares (KRGLS)

- Define features with a kernel function $k(\mathbf{x}, \mathbf{x}_i)$ per point $\mathbf{x}_i$
- Define the Gram matrix as a kernel matrix:

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & k(\mathbf{x}_N, \mathbf{x}_2) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}. \quad (1)$$

- If we had an infinity of data points, the linear approximation in feature space would become perfect
- Intuition: the error is a function of the distance to data points
- Computing the weights is done with RR using

$$\boldsymbol{\theta}^* = (\lambda \mathbf{I} + \mathbf{K})^{-1} \mathbf{y}, \quad (2)$$

- Note that $\mathbf{K}$ is symmetric
- The kernel matrix $\mathbf{K}$ grows with the number of points (kernel expansion)
- The matrix inversion may become too expensive
- Solution: finite set of features (RBFNs), incremental methods

Radial Basis Function Networks: definition and solution

- Radial Basis Functions versus Kernels (Gaussians $\phi(\mathbf{x}, \boldsymbol{\theta}_e) = e^{-\frac{1}{2}(\mathbf{x}-\mu_e)^T \boldsymbol{\Sigma}_e^{-1}(\mathbf{x}-\mu_e)}$ are both)
- We define a set of E basis functions (often Gaussian)

$$\hat{f}(\mathbf{x}) = \sum_{e=1}^{E} w_e \cdot \phi(\mathbf{x}, \boldsymbol{\theta}_e) \tag{3}$$

$$= \boldsymbol{\theta}^{\intercal} \cdot \phi(\mathbf{x}). \tag{4}$$

- We also define the *Gram matrix*

$$\mathbf{G} = \begin{bmatrix} \phi(\mathbf{x}_1, \boldsymbol{\theta}_1) & \phi(\mathbf{x}_1, \boldsymbol{\theta}_2) & \cdots & \phi(\mathbf{x}_1, \boldsymbol{\theta}_E) \\ \phi(\mathbf{x}_2, \boldsymbol{\theta}_1) & \phi(\mathbf{x}_2, \boldsymbol{\theta}_2) & \cdots & \phi(\mathbf{x}_2, \boldsymbol{\theta}_E) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\mathbf{x}_N, \boldsymbol{\theta}_1) & \phi(\mathbf{x}_N, \boldsymbol{\theta}_2) & \cdots & \phi(\mathbf{x}_N, \boldsymbol{\theta}_E) \end{bmatrix} \tag{5}$$

- and we get the least squares solution

$$\boldsymbol{\theta}^* = (\mathbf{G}^{\intercal}\mathbf{G})^{-1}\mathbf{G}^{\intercal}\mathbf{y}. \tag{6}$$

Least Square Projection Methods: summary of computations

- Linear case

$$\boldsymbol{\theta}^* = (\bar{\mathbf{X}}^\intercal \bar{\mathbf{X}})^{-1} \bar{\mathbf{X}}^\intercal \mathbf{y} \qquad (LS) \qquad (7)$$

$$\boldsymbol{\theta}^* = (\lambda \mathbf{I} + \bar{\mathbf{X}}^\intercal \bar{\mathbf{X}})^{-1} \bar{\mathbf{X}}^\intercal \mathbf{y}. \qquad (RLS) \qquad (8)$$

- Gram matrix case

$$\boldsymbol{\theta}^* = (\mathbf{G}^\intercal \mathbf{G})^{-1} \mathbf{G}^\intercal \mathbf{y} \qquad (RBFN) \qquad (9)$$

- Kernel matrix case

$$\boldsymbol{\theta}^* = \mathbf{K}^{-1} \mathbf{y}, \qquad (GPR) \qquad (10)$$

$$\boldsymbol{\theta}^* = (\lambda \mathbf{I} + \mathbf{K})^{-1} \mathbf{y}. \qquad (KRR) \qquad (11)$$

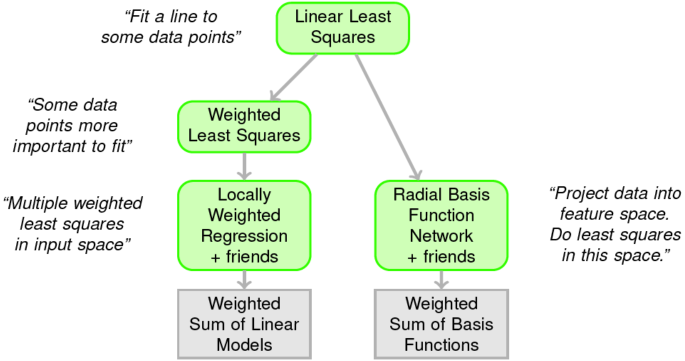
LWR versus RBFNs

$$\hat{f}(\mathbf{x}) = \sum_{e=1}^{E} \phi(\mathbf{x}, \boldsymbol{\theta}_e) \cdot (b_e + \mathbf{a}_e^{\intercal}\mathbf{x}) \tag{12}$$

$$\hat{f}(\mathbf{x}) = \sum_{e=1}^{E} \phi(\mathbf{x}, \boldsymbol{\theta}_e) \cdot \, w_e, \tag{13}$$

- Eq. (13) is a special case of (12) with $\mathbf{a}_e = \mathbf{0}$ and $b_e = w_e$.
- RBFNs: performs one LS computation in a projected space
- LWR: performs many LS computation in local domains

Wrap-up



- Image taken from Freek Stulp's IROS 2018 Tutorial

Bishop, C. M. (2007).
Pattern recognition and machine learning.
Springer Berlin/Heidelberg, Germany.

Stulp, F. and Sigaud, O. (2015).
Many regression algorithms, one unified model: A review.
Neural Networks, 69:60–79.