Radial Basis Function Networks

Sargur Srihari

Topics

- Basis Functions
- Radial Basis Functions
- Gaussian Basis Functions
- Nadaraya Watson Kernel Regression Model

Basis Functions

- Summary of Linear Regression Models
- 1. Polynomial Regression with one variable

$$y(x,w) = w_0 + w_1 x + w_2 x^2 + ... = \sum w_i x$$

2. Simple Linear Regression with *D* variables

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + ... + w_D x_D = \mathbf{w}^T \mathbf{x}$$

In one-dimensional case

$$y(x, \mathbf{w}) = w_0 + w_I x$$

which is a straight line

3. Linear Regression with Basis functions $\phi_i(x)$

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x) = w^T \phi(x)$$

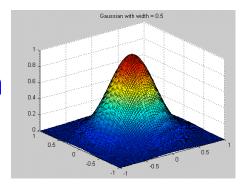
- There are now M parameters instead of D parameters
- What form should the basis functions take?

Radial Basis Functions

 A radial basis function depends only on the radial distance (typically Euclidean) from the origin

$$\phi(\mathbf{x}) = \phi(||\mathbf{x}||)$$

• If the basis function is centered at μ_j then $\phi_i(\mathbf{x}) = h(||\mathbf{x}-\mu_i||)$



 We look at radial basis functions centered at the data points x_n, n =1,...,N

History of Radial Basis Functions

- Introduced for exact function interpolation
 - Given set of input vectors $x_1,...,x_N$ and target values $t_1,...,t_N$
 - Goal is to find a smooth function f(x) that fits every target value exactly so that $f(x_n) = t_n$ for n=1,...,N
- Achieved by expressing f(x) as a linear combination of radial basis functions, one per data point

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n h(\|\mathbf{x} - \mathbf{x}_n\|)$$

- Values of coefficients w_n are found by least squares
 - Very large number of weights
- Since there are same number of coefficients as constraints, result is a function that fits target values exactly
 - When target values are noisy exact interpolation is undesirable

Another Motivation for Radial Basis Functions

- Interpolation when Input rather than target is noisy
- If noise on input variable x is described by variable ξ
 having distribution v(ξ) then sum of squared error
 function becomes

$$E = \frac{1}{2} \sum_{n=1}^{N} \int \{y(x_n + \xi) - t_n\}^2 v(\xi) d\xi$$

v: pronounced "nu" for noise

Using calculus of variations, optimize wrt y(x) to give

$$y(\mathbf{x}) = \sum_{n=1}^{N} t_n h(\mathbf{x} - \mathbf{x}_n)$$

where the basis functions are given by

$$h(x-x_n) = \frac{v(x-x_n)}{\sum_{n=1}^{N} v(x-x_n)}$$

- There is a basis function centered on every data point
- Known as Nadaraya-Watson Model

Normalized Basis Functions

If noise v(x) is isotropic so that it is only a function of ||x|| then the basis functions are radial

Functions are normalized

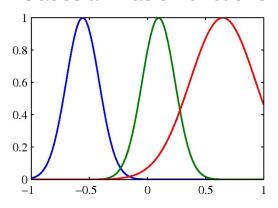
so that

$$h(\mathbf{x} - \mathbf{x}_n) = \frac{\mathbf{v}(\mathbf{x} - \mathbf{x}_n)}{\sum_{n=1}^{N} \mathbf{v}(\mathbf{x} - \mathbf{x}_n)}$$

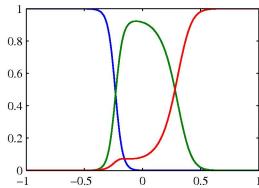
 $\sum h(x - x_n) = 1$ for any value of x \longrightarrow

Normalization is useful in regions of input space where all basis functions are small

Gaussian Basis Functions



Normalized Basis Functions



 $h(x-x_n)$ is called a kernel function since we use it with every sample to determine value at x

Factorization into basis functions is not used

Computational Efficiency

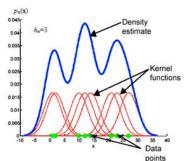
- Since there is one basis function per data point corresponding computational model will be costly to evaluate for new data points
- Methods for choosing a smaller set
 - Use a random subset of data points
 - Orthogonal least squares
 - At each step the next data point to be chosen as basis function center is the one that gives the greatest reduction in leastsquared error
 - Clustering algorithms which give basis function centers that no longer coincide with training data points

Nadaraya-Watson Regression Model

- More general formulation than before
 - Proposed in 1964. Nadaraya: a Russian statistician
- Begin with Parzen window estimation and derive the kernel **function**
 - Given a training set $\{x_n,t_n\}$ the joint distribution of two variables is

$$p(\mathbf{x},t) = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x} - \mathbf{x}_n, t - t_n)$$
Parzen window estimate

With one variable $p(x)$



- where f(x,t) is the component density function
- there is one such component centered at each data point
- Starting with $y(x)=E[t|x]=\int tp(t|x)dt$ we derive the kernel form
 - Where we need to note that component densities have zero mean

$$\int_{-\infty}^{\infty} f(\mathbf{x}, t) t \, \mathrm{d}t = 0$$

Derivation of Nadaraya-Watson Regression

Desired Regression function
$$y(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int_{-\infty}^{\infty} tp(t|\mathbf{x}) \, \mathrm{d}t$$
$$= \frac{\int tp(\mathbf{x},t) \, \mathrm{d}t}{\int p(\mathbf{x},t) \, \mathrm{d}t}$$

Using Parzen window estimate and change of variables

$$y(\mathbf{x}) = \frac{\sum_{n} g(\mathbf{x} - \mathbf{x}_{n})t_{n}}{\sum_{m} g(\mathbf{x} - \mathbf{x}_{m})}$$
$$= \sum_{n} k(\mathbf{x}, \mathbf{x}_{n})t_{n}$$

Where k is a normalized kernel

$$k(\mathbf{x}, \mathbf{x}_n) = \frac{g(\mathbf{x} - \mathbf{x}_n)}{\sum g(\mathbf{x} - \mathbf{x}_m)} \qquad g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x}, t) \, dt.$$

Example of Nadaraya-Watson Regression

Regression function is

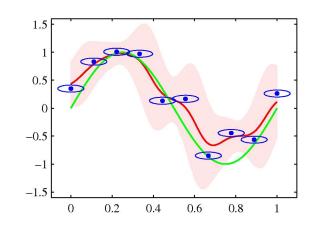
$$y(\mathbf{x}) = \sum_{n} k(\mathbf{x}, \mathbf{x}_n) t_n$$

Where the kernel function is

$$k(\mathbf{x}, \mathbf{x}_n) = \frac{g(\mathbf{x} - \mathbf{x}_n)}{\sum_{m} g(\mathbf{x} - \mathbf{x}_m)}$$

and we have defined

$$g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x}, t) dt$$



Green: Original sine function

Blue: Data points

Red: Resulting regression function

Pink region: Two std deviation region for distribution p(t|x)

Blue ellipses: one std deviation contour

Different scales on x and y axes make circles elongated

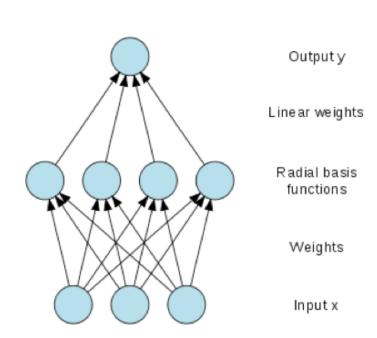
and f(x,t) is a zero-mean component density function centered at each data point

Radial Basis Function Network

A neural network that uses RBFs as activation functions

$$\varphi(\mathbf{x}) = \sum_{i=1}^{N} a_i \rho(||\mathbf{x} - \mathbf{c}_i||)$$

- In Nadaraya-Watson
 - Weights a_i are target values
 - ρ is component density (Gaussian)
 - Centers \mathbf{c}_i are samples



Speeding-up RBFs

- More flexible forms of Gaussian components can be used
 - Model p(x,t) using a Gaussian mixture model
 - Trained using EM
- Number of components in the mixture model can be smaller than the number of training samples
 - Faster evaluation for test data points
 - But increased training time