



STEVENS
INSTITUTE *of* TECHNOLOGY
THE INNOVATION UNIVERSITY®

CPE/EE 695: Applied Machine Learning

Lecture 2 – 2 (optional): Basis Function Models

Dr. Shucheng Yu, Associate Professor
Department of Electrical and Computer Engineering
Stevens Institute of Technology

Slides of this section is mainly borrowed from Dr. Bishop's PRML lectures.

Linear Basis Function Models (1)

Generally

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

where $\phi_j(\mathbf{x})$ are known as *basis functions*.

Typically, $\phi_0(\mathbf{x}) = 1$, so that w_0 acts as a bias.

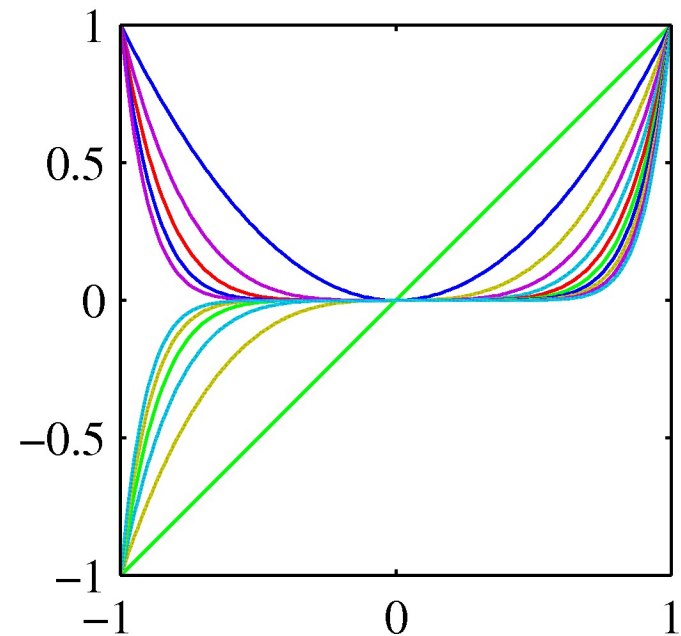
In the simplest case, we use linear basis functions : $\phi_d(\mathbf{x}) = x_d$.

Linear Basis Function Models (2)

Polynomial basis functions:

$$\phi_j(x) = x^j.$$

These are global; a small change in x affect all basis functions.

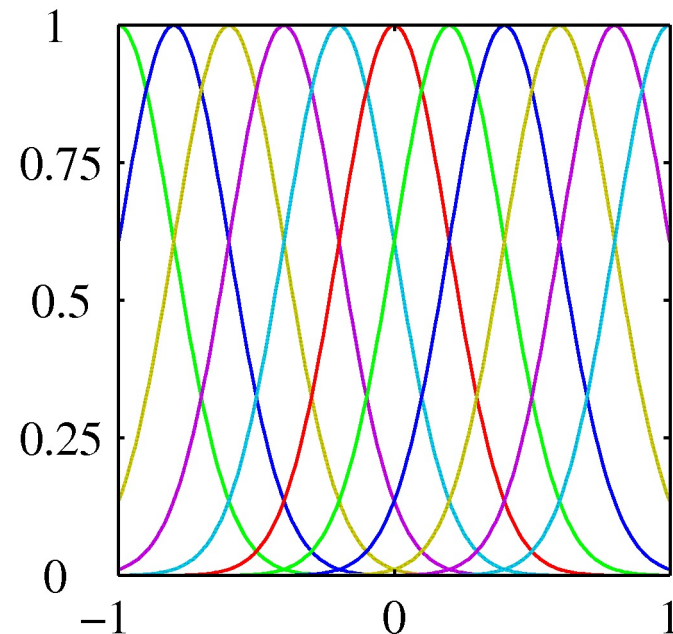


Linear Basis Function Models (3)

Gaussian basis functions:

$$\phi_j(x) = \exp \left\{ -\frac{(x - \mu_j)^2}{2s^2} \right\}$$

These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).



Linear Basis Function Models (4)

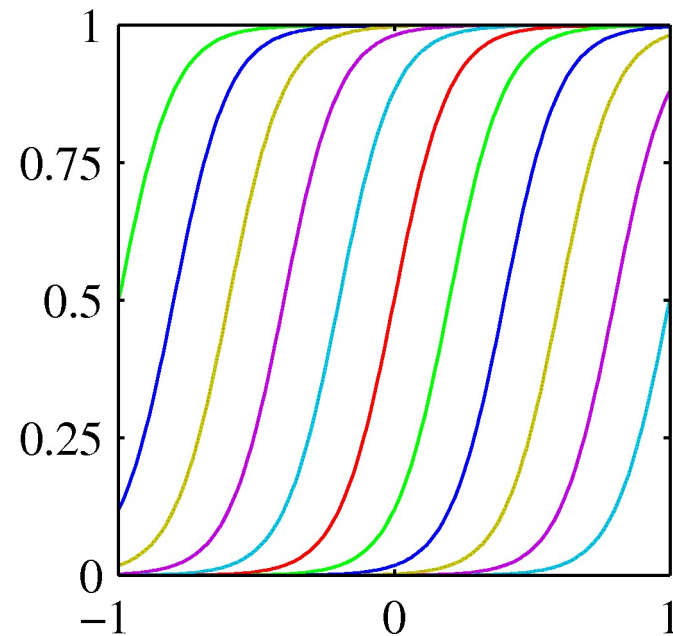
Sigmoidal basis functions:

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

Also these are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (slope).



Maximum Likelihood and Least Squares (1)

Assume observations from a deterministic function with added Gaussian noise:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \quad \text{where} \quad p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

which is the same as saying,

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

Given observed inputs, $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and targets, $\mathbf{t} = [t_1, \dots, t_N]^T$, we obtain the likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}).$$

Maximum Likelihood and Least Squares (2)

Taking the logarithm, we get

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

where

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

is the sum-of-squares error.

Maximum Likelihood and Least Squares (3)

Computing the gradient and setting it to zero yields

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\} \phi(\mathbf{x}_n)^T = \mathbf{0}.$$

Solving for \mathbf{w} , we get

$$\mathbf{w}_{\text{ML}} = \left(\Phi^T \Phi \right)^{-1} \Phi^T \mathbf{t}$$

The Moore-Penrose
pseudo-inverse, Φ^\dagger .

where

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

Geometry of Least Squares

Consider

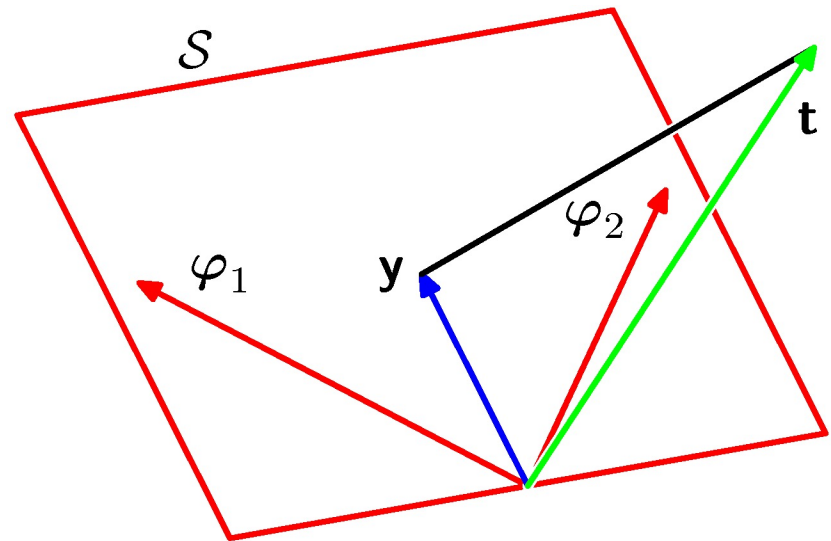
$$\mathbf{y} = \Phi \mathbf{w}_{\text{ML}} = [\varphi_1, \dots, \varphi_M] \mathbf{w}_{\text{ML}}.$$

$$\mathbf{y} \in \mathcal{S} \subseteq \mathcal{T} \quad \mathbf{t} \in \mathcal{T}$$

$\begin{array}{c} \uparrow \\ \text{N-dimensional} \\ \uparrow \\ \text{M-dimensional} \end{array}$

\mathcal{S} is spanned by $\varphi_1, \dots, \varphi_M$.

\mathbf{w}_{ML} minimizes the distance between \mathbf{t} and its orthogonal projection on \mathcal{S} , i.e. \mathbf{y} .



Sequential Learning

Data items considered one at a time (a.k.a. online learning); use stochastic (sequential) gradient descent:

$$\begin{aligned}\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - \eta \nabla E_n \\ &= \mathbf{w}^{(\tau)} + \eta (t_n - \mathbf{w}^{(\tau)\top} \phi(\mathbf{x}_n)) \phi(\mathbf{x}_n).\end{aligned}$$

This is known as the *least-mean-squares (LMS) algorithm*. Issue: how to choose η ?

Regularized Least Squares (1)

Consider the error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

With the sum-of-squares error function and a quadratic regularizer, we get

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

which is minimized by

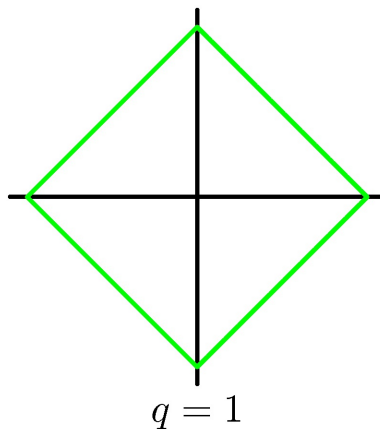
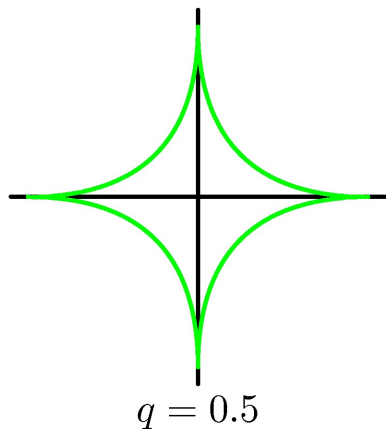
$$\mathbf{w} = \left(\lambda \mathbf{I} + \Phi^T \Phi \right)^{-1} \Phi^T \mathbf{t}.$$

λ is called the regularization coefficient.

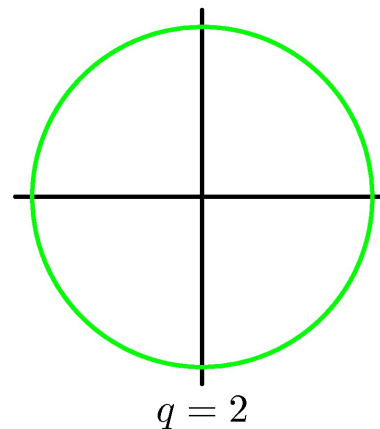
Regularized Least Squares (2)

With a more general regularizer, we have

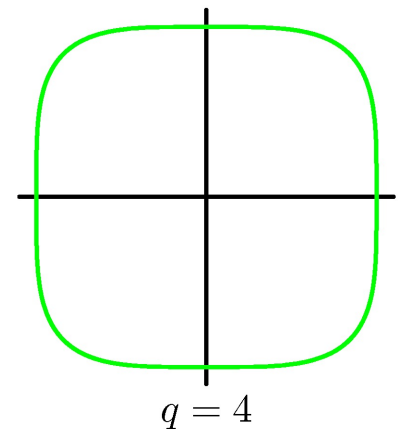
$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$



Lasso



Quadratic



Multiple Outputs (1)

Analogously to the single output case we have:

$$\begin{aligned} p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta) &= \mathcal{N}(\mathbf{t}|\mathbf{y}(\mathbf{W}, \mathbf{x}), \beta^{-1}\mathbf{I}) \\ &= \mathcal{N}(\mathbf{t}|\mathbf{W}^T\phi(\mathbf{x}), \beta^{-1}\mathbf{I}). \end{aligned}$$

Given observed inputs, $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and targets, $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_N]^T$, we obtain the log likelihood function

$$\begin{aligned} \ln p(\mathbf{T}|\mathbf{X}, \mathbf{W}, \beta) &= \sum_{n=1}^N \ln \mathcal{N}(\mathbf{t}_n|\mathbf{W}^T\phi(\mathbf{x}_n), \beta^{-1}\mathbf{I}) \\ &= \frac{NK}{2} \ln \left(\frac{\beta}{2\pi} \right) - \frac{\beta}{2} \sum_{n=1}^N \|\mathbf{t}_n - \mathbf{W}^T\phi(\mathbf{x}_n)\|^2. \end{aligned}$$

Multiple Outputs (2)

Maximizing with respect to \mathbf{W} , we obtain

$$\mathbf{W}_{\text{ML}} = \left(\Phi^T \Phi \right)^{-1} \Phi^T \mathbf{T}.$$

If we consider a single target variable, t_k , we see that

$$\mathbf{w}_k = \left(\Phi^T \Phi \right)^{-1} \Phi^T \mathbf{t}_k = \Phi^\dagger \mathbf{t}_k$$

where $\mathbf{t}_k = [t_{1k}, \dots, t_{Nk}]^T$, which is identical with the single output case.
