Linear Models for Regression

Alberto Paccanaro
EMAp – FGV

www.paccanarolab.org

Material and images in these slides are from (or adapted from):
C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006

The Regression problem

Predict the value of **continuous** *target* variables t given the value of a D-dimensional vector $\mathbf{x}$ of *input* variables.

Given N observations $\{\mathbf{x}_n\}$, where $n = 1, \ldots, N$, together with corresponding target values $\{t_n\}$, the goal is to predict the value of t <u>for a new value of $\mathbf{x}$</u>.

The Roadmap

We model the predictive distribution $p(t|\boldsymbol{x})$ expressing our uncertainty about the value of t for each value of $\boldsymbol{x}$.

- learn $p(t|\mathbf{x})$ by minimizing a loss function.
- common choice of loss function for real-valued variables: Sum of Squared Errors – **consequence of maximizing likelihood under the assumption of a Gaussian noise distribution.**

➔ the optimal solution is given by the conditional expectation of t (from decision theory)

Linear Models: linear functions of the adjustable parameters

IMPORTANT: can be nonlinear with respect to the input variables (e.g. the polynomial we saw earlier)

- simplest form: models are also linear functions of the input variables.

- ***basis functions***: nonlinear functions of the input variables, of which we take linear combinations

Regression by linear combination of basis functions

- Simplest linear model for regression (often just called *linear regression*):

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D = w_0 + \sum_{i=1}^{D} w_i x_i$$

- linear combinations of fixed nonlinear Basis Functions, $\phi_j(\mathbf{x})$:

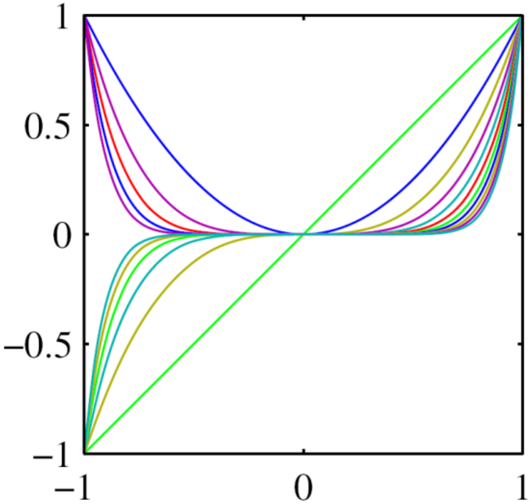
$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

Nonlinear basis functions allow ***y*(x,w)** to be nonlinear !

- w_0, called **bias** parameter, can be included in the sum, by defining one extra basis function $\phi_0(\mathbf{x}) = 1$

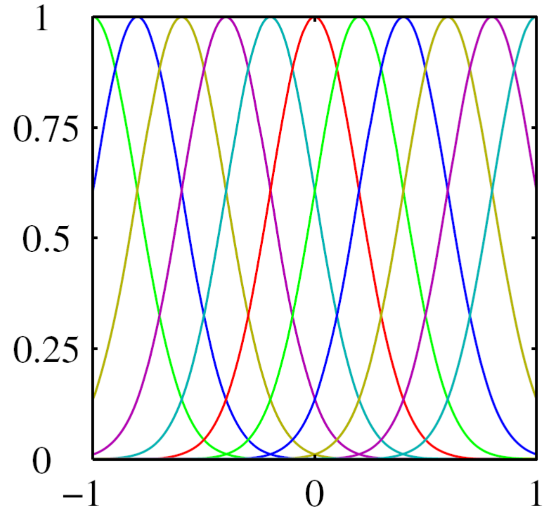
$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

Classic basis functions



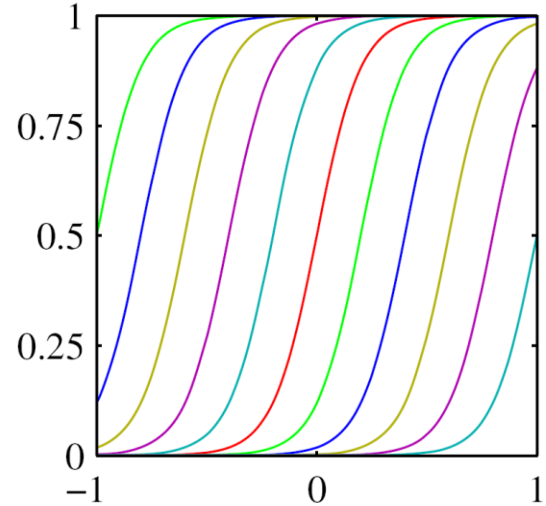
Polynomial

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



Gaussian

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



Sigmoidal

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

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

μ_i control the locations of the basis functions
s controls the scale of the basis function

Maximum likelihood and least squares

(it's similar to what we did earlier…)

Assume t is given by:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

ϵ gaussian random variable

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

Given data set of N inputs $\mathbf{X}$

Likelihood function (a function of **w** and β)

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

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

where: $E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$ is the sum-of-squares error function

Gradient of the log likelihood function

$$\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$$

Setting the gradient to 0:

$$0 = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} \right)$$

Solving for **w** we obtain

Moore-Penrose pseudo-inverse of Φ

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

the ***normal equations*** for the least squares problem

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

Design Matrix

$$\Phi_{nj} = \varphi_j(\mathbf{x}_n)$$

Important connection to help us understand…

The *Moore-Penrose pseudo-inverse* of the matrix $\mathbf{\Phi}$, $\mathbf{\Phi}^{\dagger}$ can be regarded as a generalization of the notion of matrix inverse to nonsquare matrices

$$\boxed{\mathbf{\Phi}^{\dagger} \equiv \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}}$$

If $\mathbf{\Phi}$ is square and invertible then using the property $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ we see that $\mathbf{\Phi}^{\dagger} \equiv \mathbf{\Phi}^{-1}$.

Sequential/online learning

Batch techniques can be computationally costly for large datasets.

Sequential algorithms: data points are considered one at a time. Model parameters updated after each presentation.

Stochastic gradient descent/ sequential gradient descent

$$\boxed{\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n}$$

E_n is the error for point n

For sum-of-squares error function:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)\mathrm{T}}\boldsymbol{\phi}_n)\boldsymbol{\phi}_n$$

Least-Mean-Squares or the LMS algorithm

$$\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$$

Regularized Least Squares

<u>Idea</u>: adding a regularization term to an error function to control over-fitting

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

1. Weight decay *(in machine learning)*

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

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

(ridge regression in statistics)

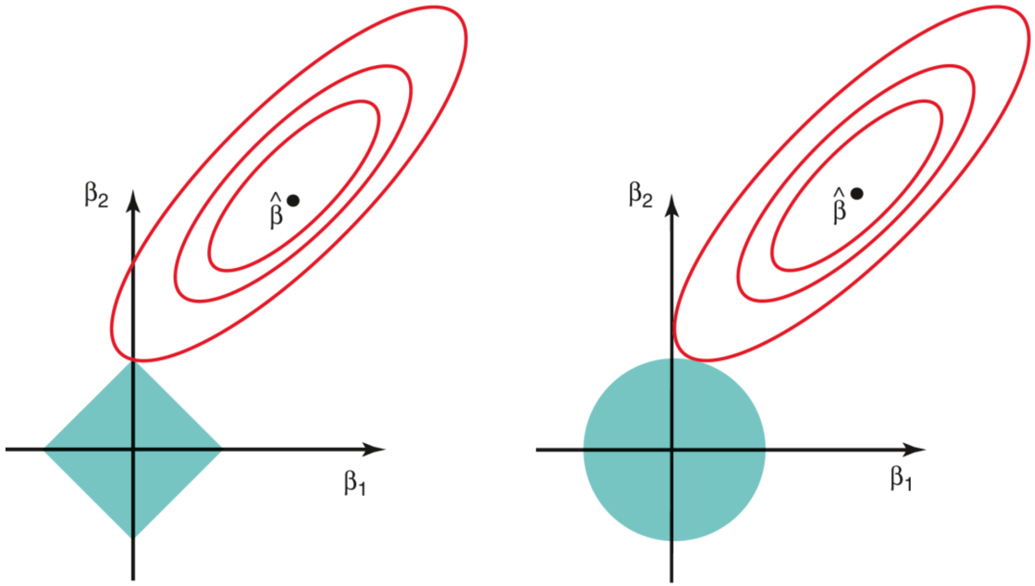
$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

normal equations with weight decay

2. Lasso (Least Absolute Shrinkage and Selection Operator)

$$E_W(\mathbf{w}) = \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|$$

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



some of the coefficients w_j become 0, leading to a sparse model

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

A.PA

3. Elastic Net

$$E_W(\mathbf{w}) = \frac{\lambda_1}{2}\sum_{j=1}^{M}|w_j| + \frac{\lambda_2}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

combines Lasso and Ridge regularization

Linear regression for multiple outputs

Predict $K > 1$ target variables – target vector $\mathbf{t}$

The generalization is straightforward

$$\mathbf{y}(\mathbf{x}, \mathbf{w}) = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$$

$$p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta) = \mathcal{N}(\mathbf{t}|\mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \beta^{-1}\mathbf{I})$$

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

$$\boxed{\mathbf{W}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{T}} \qquad \mathbf{w}_k = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}_k = \boldsymbol{\Phi}^{\dagger}\mathbf{t}_k$$