Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

8. Linear models for regression

L. Iocchi, F. Patrizi

now the output is a real value

Overview

- Linear models for regression
- Maximum likelihood and Least squares
- Sequential learning
- Regularization

References
C. Bishop. Pattern Recognition and Machine Learning. Sect. 3.1

Linear Models for Regression

Learning a function $f : X \to Y$, with

- $X \subseteq \Re^d$
- $Y = \Re$

tn is a real value

from data set $D = \{(\mathbf{x}_n, t_n)_{n=1}^N\}$

input-output pairs

Linear Models for Regression

The simplest linear model for regression is one that involves a linear combination of the input variables

Define a model $y(\mathbf{x}; \mathbf{w})$ with parameters $\mathbf{w}$ to approximate the target function f.

x_n is a d dimensional vector

Linear model for linear functions

input

params of the model

$$y(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d = \mathbf{w}^T \mathbf{x}$$

linear model of a linear function

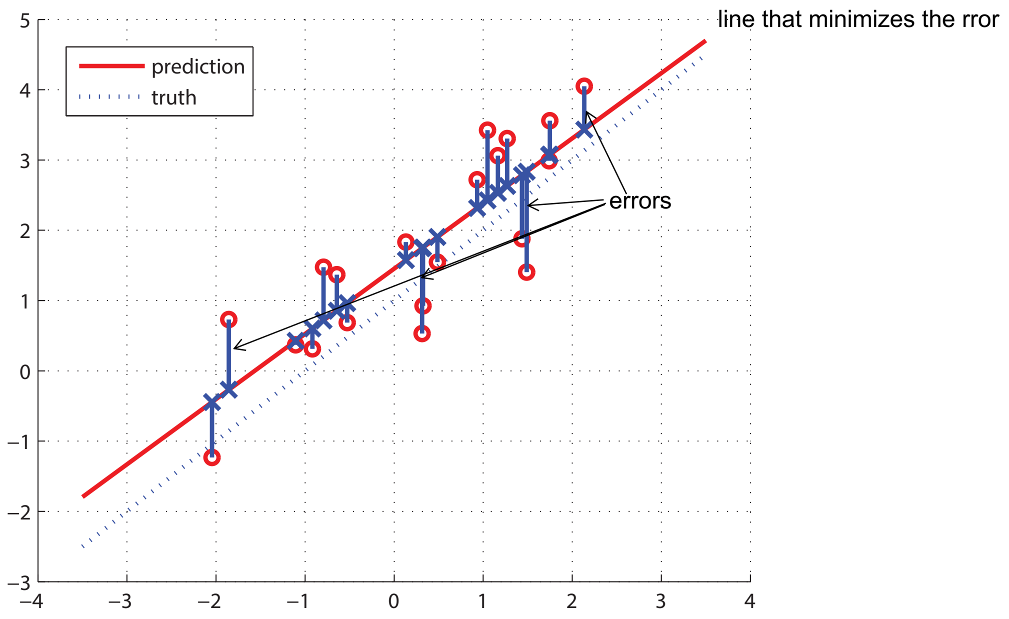
$$\text{with } \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \text{ and } \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

This is often simply known as linear regression. The key property of this model is that it is a linear function of the parameters w0,...,wD. It is also, however, a linear function of the input variables xi, and this imposes significant limitations on the model. We therefore extend the class of models by considering linear combinations of fixed nonlinear functions of the input variables (SLIDE 7)

Example: 2D line fitting

determine the orientation of the line

$$y = w_0 + w_1 x_1$$



Linear Models for Regression

Linear Basis Function Models

in general there is also the bias w0+ in front of the formula, but It is often convenient to define an additional dummy "basis function" ϕ0(x) = 1 so that w0 disappears

Using nonlinear functions of input variables:

the model is linear in w because w is linear

for ex.

$$\phi = \left.\begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}\right\} M+1$$

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

this formula is called LINEAR MODEL

this is non-linear in x

$$\text{with } \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_M \end{bmatrix}, \ \phi(\mathbf{x}) = \begin{bmatrix} \phi_0(\mathbf{x}) \\ \vdots \\ \phi_M(\mathbf{x}) \end{bmatrix}, \text{ and } \phi_0(\mathbf{x}) = 1.$$

- Still linear in the parameters **w**!

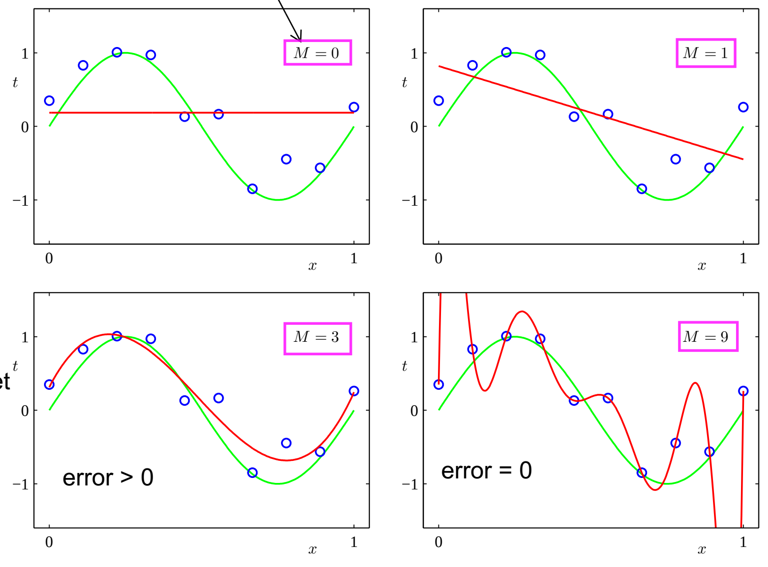
Example: Polynomial curve fitting

One limitation of polynomial basis functions is that they are global functions of the input variable, so that changes in one region of input space affect all other regions. This can be resolved by dividing the input space up into regions and fit a different polynomial in each region, leading to spline functions

$$y = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

it means that the model is just a constant so the line is orizontal

3 degrees polinomial, line that minimize distances of the dataset



line that minimizes the distances

error = 0 becuse this matches perfectly the dataset. this solution is perfect in the dataset but we are interested in classify samples that are not in the dataset

Warning: overfitting!!! → in fact the best solution will be M=3

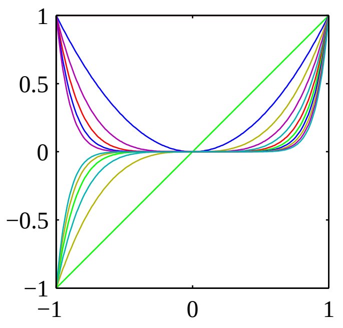
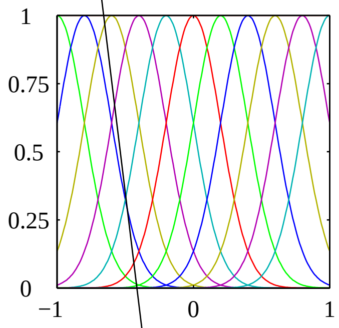
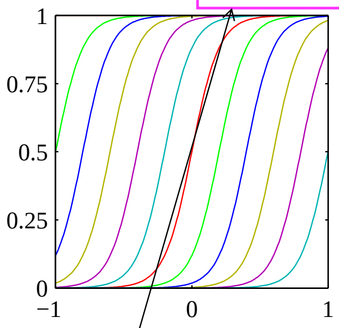
Linear Regression Basis Functions

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

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

Examples of basis functions

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

Polynomial

Radial

Sigmoid / Tanh

$$\tanh(a) = 2\sigma(a) - 1,$$

Linear Regression - Algorithms

we have just fitted polynomial functions to data sets by minimizing a sumof-squares error function. We also showed that this error function could be motivated as the maximum likelihood solution under an assumed Gaussian noise model. Let us return to this discussion and consider the least squares approach, and its relation to maximum likelihood

Maximum likelihood and least squares

Target value t is given by $y(\mathbf{x}; \mathbf{w})$ affected by additive noise ϵ

t is a deterministic function with additive gaussian noise

$$\underset{\text{target value}}{t} = \overset{\text{thruth function}}{y(\mathbf{x}; \mathbf{w})} + \overset{\text{some noise}}{\epsilon}$$

Assume Gaussian noise $P(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$, with precision (inverse variance) β.

We have:

$$P(t|\mathbf{x}, \mathbf{w}, \beta) = \underset{\text{gaussian}}{\mathcal{N}}(t|\overset{\text{mean}}{y(\mathbf{x}; \mathbf{w})}, \overset{\text{variance}}{\beta^{-1}})$$

Linear Regression - Algorithms

Assume observations independent and identically distributed (i.i.d.)

Making the assumption that these data points are drawn independently from the gaussian distribution, we obtain the following expression for the likelihood function, which is a function of the adjustable parameters w and β

We seek the maximum of the likelihood function:

$$P(\{t_1,\dots,t_N\}|\mathbf{x}_1,\dots,\mathbf{x}_N,\mathbf{w},\beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n),\beta^{-1}).$$

or equivalently: minimum of the negative loglikelihood

$$\ln P(\{t_1,\dots,t_N\}|\mathbf{x}_1,\dots,\mathbf{x}_N,\mathbf{w},\beta) = \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n),\beta^{-1})$$

$$= -\beta \underbrace{\frac{1}{2}\boxed{\sum_{n=1}^{N}[t_n - \mathbf{w}^T\phi(\mathbf{x}_n)]^2}}_{E_D(\mathbf{w})} - \frac{N}{2}\ln(2\pi\beta^{-1}).$$

we minimize this term because is the only that contains w

sum-of-squares error function

Having written down the likelihood function, we can use maximum likelihood to determine w and β

Linear Regression - Algorithms

Maximum likelihood (zero-mean Gaussian noise assumption)

$$\operatorname{argmax} P(\{t_1,\dots,t_N\}|\mathbf{x}_1,\dots,\mathbf{x}_N,\mathbf{w},\beta)$$

corresponds to least square error minimization

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

Linear Regression - Algorithms

Note:

$$E_D(\mathbf{w}) = \frac{1}{2}(\mathbf{t} - \mathbf{\Phi w})^T(\mathbf{t} - \mathbf{\Phi w}),$$

$$\boxed{\text{with } \mathbf{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}} \text{ and } \boxed{\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_M(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_M(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_M(\mathbf{x}_N) \end{bmatrix}}.$$

Optimality condition:

$$\nabla E_D = 0 \iff \mathbf{\Phi}^T\mathbf{\Phi w} = \mathbf{\Phi}^T\mathbf{t}.$$

Hence:

$$\mathbf{w}_{ML} = \underbrace{(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T}_{\mathbf{\Phi}^\dagger:\ \text{pseudo-inverse}}\ \mathbf{t}.$$

Moore-Penrose pseudo-inverse of the matrix Φ

Linear Regression - Algorithms

Batch techniques, such as the maximum likelihood solution, which involve processing the entire training set in one go, can be computationally costly for large data sets. If the data set is sufficiently large, it may be worthwhile to use sequential algorithms, also known as on-line algorithms in which the data points are considered one at a time, and the model parameters updated after each such presentation

Sequential learning is also appropriate for realtime applications in which the data observations are arriving in a continuous stream, and predictions must be made before all of the data points are seen.

Sequential Learning

Stochastic gradient descent algorithm:

$$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{w}} - \eta \nabla E_n$$

η: learning rate parameter

Therefore:

The value of w is initialized to some starting vector w(0). For the case of the sum-of-squares error function E_D this gives

$$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{w}} + \eta\left[t_n - \hat{\mathbf{w}}^T\phi(\mathbf{x}_n)\right]\phi(\mathbf{x}_n)$$

This is known as LEAST-MEAN-SQUARES or the LMS algorithm

Algorithm converges for suitable small values of η.

The value of η needs to be chosen with care to ensure that the algorithm converges

Linear Regression - Regularization

Regularization is a technique to control over-fitting.

to control overfitting and reduce it we need to add another term called regularization coeff

we introduced the idea of adding a regularization term to an error function in order to control over-fitting, so that the total error function to be minimized takes the form→

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

←it penalizes the weights that determine the overfitting

with $\lambda > 0$ being the regularization factor

penalizes coefficients that are too high

It controls the relative importance of the data-dependent error E_D(w) and the regularization term E_W(w)

A common choice:

of regularizer is given by the sum-of-squares of the weight vector elements

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

Other choices:

consider the sum-of-squares error function

$$\boxed{E_W(\mathbf{w}) = \sum_{j=0}^{M} |w_j|^q.}$$

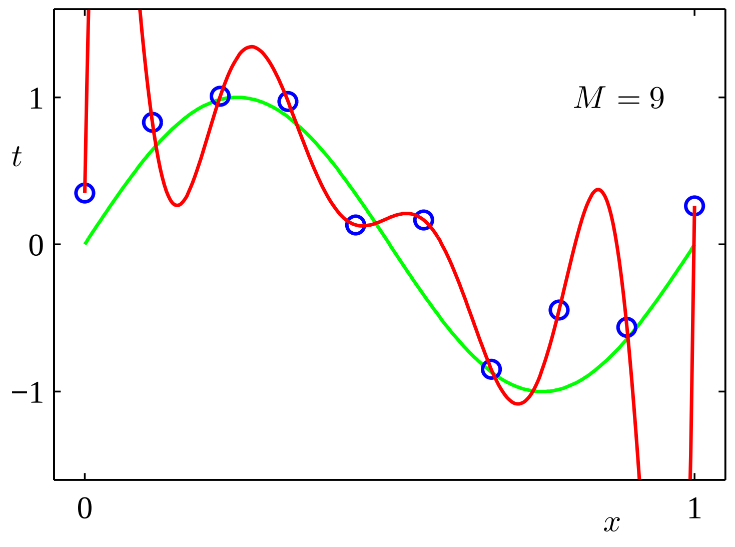
put all together

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

→this choice of regularizer is known in the machine learning literature as weight decay because in sequential learning algorithms, it encourages weight values to decay towards zero, unless supported by the data

Linear Regression - Regularization

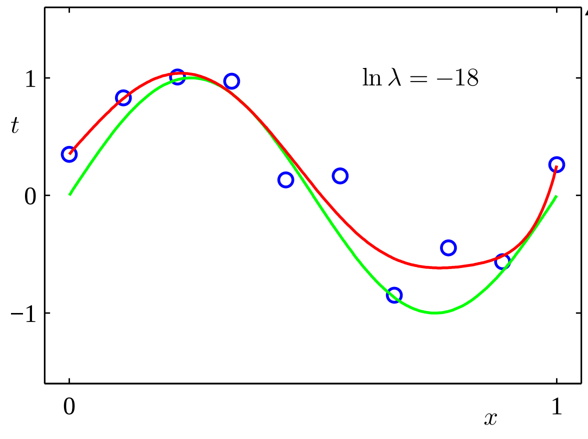
$$\text{argmin } E_D(\mathbf{w})$$

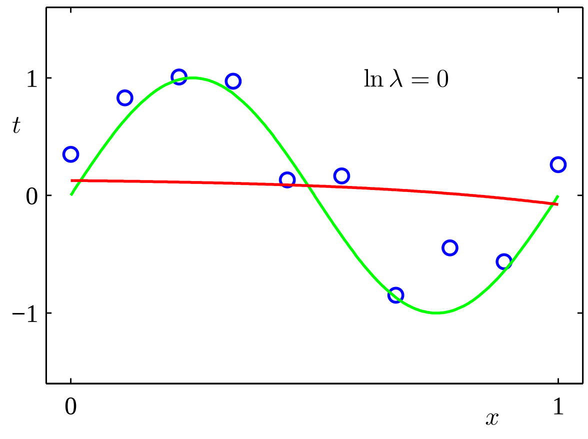


Linear Regression - Regularization

with use of regularization term

$$\text{argmin } E_D(\mathbf{w}) + \boxed{\lambda} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$





Linear Regression - Multiple outputs

So far, we have considered the case of a single target variable t. In some applications, we may wish to predict K>1 target variables, which we denote collectively by the target vector t. This could be done by introducing a different set of basis functions for each component of t, leading to multiple, independent regression problems. However, a more interesting, and more common, approach is to use the same set of basis functions to model all of the components of the target vector so that

$\mathbf{y}$: vector with K components

MxK matrix of parameters

$$\boxed{\mathbf{y}(\mathbf{x};\mathbf{W}) = \mathbf{W}^T\phi(\mathbf{x})}$$

M dimensional column vector

Target variable $\mathbf{T}$, with $\mathbf{t}_n$ vector of K output values for input $\mathbf{x}_n$

Suppose we take the conditional distribution of the target vector to be an isotropic Gaussian, and we make the log

$$\ln \boxed{P(\mathbf{T}|\mathbf{X},\mathbf{W},\beta)} = \sum_{n=1}^{N} \ln \boxed{\mathcal{N}(\mathbf{t}_n|\mathbf{W}^T\phi(\mathbf{x}_n),\beta^{-1}\mathbf{I})}$$

Similarly as before we obtain: As before, we can maximize this function with respect to W, giving

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

If we examine this result for each target variable tk, we have $\longrightarrow$ $\boxed{\mathbf{w}_k = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}_k = \mathbf{\Phi}^{\dagger}\mathbf{t}_k}$