8. Linear models for regression

Now we consider the problem of learning a function where the output is contiguous:

$f: X \to Y$

$X \subseteq \mathbb{R}^d$

$Y \subseteq \mathbb{R}$ ← real numbers

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

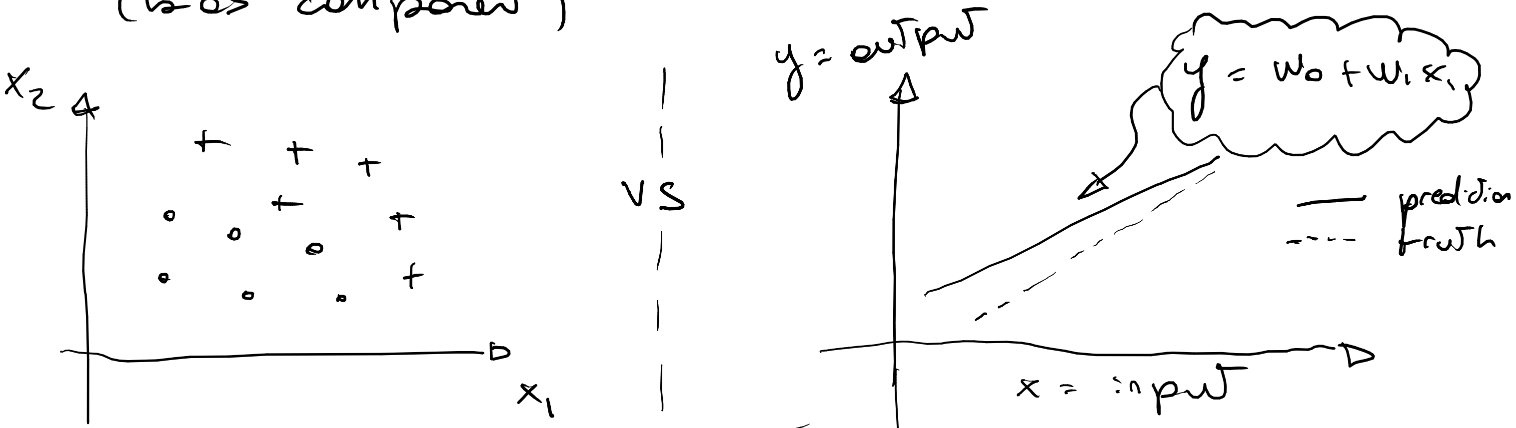
Our dataset is still a set of pairs, t_n now can be a real value

The approximation of this function should return a real value. The easy way to define a real model, we define a linear combination of weights with the component vector input:

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

$$\vec{x}^T = (1 \quad x_1 \dots x_d) \qquad \vec{w}^T = (w_0 \quad w_1 \dots w_d)$$

dummy component to include w_0 (bias component)



There is a difference

2-Dimensional Classification Problem

$f: \mathbb{R}^2 \to \{\circ, +\}$

1-Dimensional Regression problem representation

$f: \mathbb{R} \to \mathbb{R}$

With linear combinations we can learn only linear function. We can use the linear basis function model to generate LINEAR MODELS of NON-LINEAR - FUNCTIONS.

$$y(\vec{x}:\vec{w}) = \sum_{j=0}^{M} w_j \phi_j(\vec{x}) = \vec{w}^T \Phi(\vec{x})$$

with $\vec{w}^T = (w_0 \dots w_M)^T$ $\quad \Phi(\vec{x}) = \begin{pmatrix} \phi_0(\vec{x}) \\ \vdots \\ \phi_M(\vec{x}) \end{pmatrix}$ $\quad \phi_0(\vec{x}) \to 1$

$\boxed{\text{FROM } \vec{w}\text{-point of view is a LINEAR MODEL}}$

We consider a transformation of the input values.

Is a Linear combination of the weights and the transf. of the input by using the function Φ (non-linear)

Since we are interested in computing $\vec{w}$ we can apply the same reasoning of linear models

The fact that the function is now linear in $\vec{x}$ does not affect the solution. As an example we can use as BIAS function a polynomial:

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

Linear Models for Regression

Linear Basis Function Models

Using nonlinear functions of input variables:

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

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

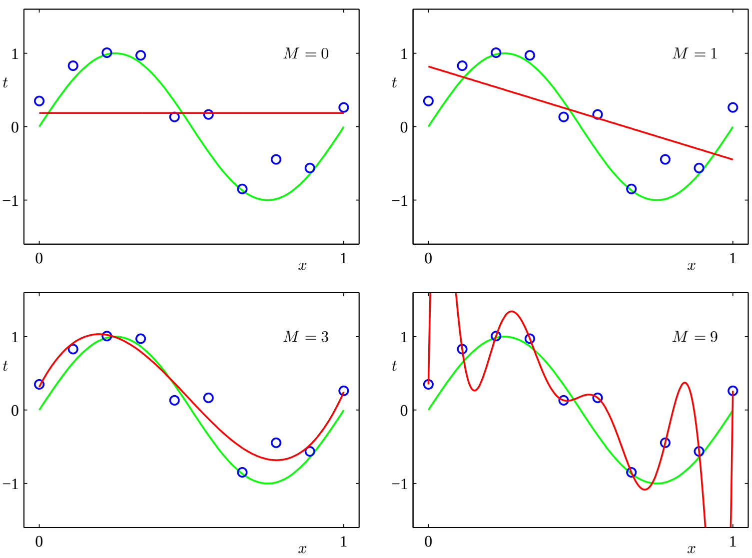
- Still linear in the parameters **w**!

Example: Polynomial curve fitting

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

only w_0

A line with $y = w_0 + w_1 x$



better, quite good approx.

Very special case because the points are 10, you have a function that makes 0 error

Warning: overfitting!!!

If we just consider the error (of course we don't know the real function), the function with $M=9$ is the best, but we don't like it because of OVERFITTING. We prefer the function with $M=3$ (Case of UNDERFITTING, we can see the shape of the data).

We can use different basis functions. How can compute parameters of our model? We will maximize the error.

The target value t is given by $y(\vec{x}; \vec{w})$ affected by additive noise ε:

$$\boxed{t = y(\vec{x}; \vec{w}) + \varepsilon}$$

We define an error model assuming samples in dataset are not perfect

We assume that the error ε is gaussian, with $\mu = 0$.

The error is a white noise process:

$p(\varepsilon | \beta) = \mathcal{N}(\varepsilon | 0, \beta^{-1})$ with precision β that is the inverse of the variance.

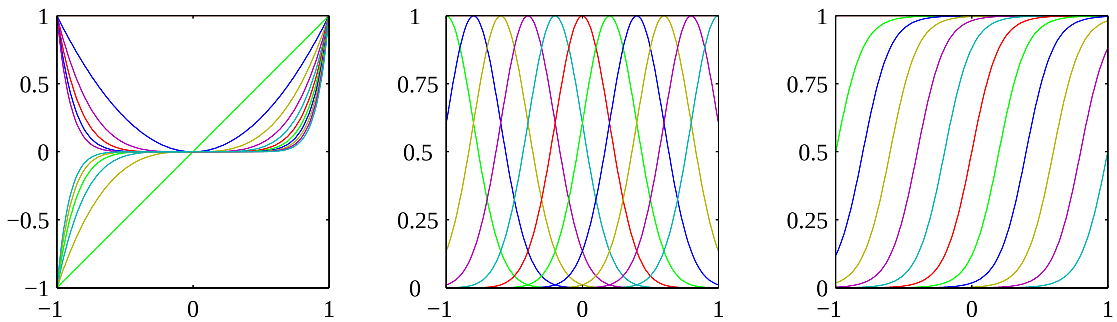
We have: $\boxed{p(t | \vec{x}, \vec{w}, \beta) = \underline{\mathcal{N}(t | y(\vec{x}; \vec{w}), \beta^{-1})}}$

likelihood expressed with the Gaussian

exploited the fact that the Gaussian is additive

Linear Regression Basis Functions

Examples of basis functions



Polynomial Radial Sigmoid / Tanh

Linear Regression - Algorithms

Maximum likelihood and least squares

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

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

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

We have:

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

We want to compute $\vec{w}$ given the dataset.

Assuming that observations are independent and identically distributed, we can compute the probability in this way:

$$p(\{t_1, \dots, t_N\} \mid x_1, \dots, x_N, \vec{w}, \beta) =$$

$$= \prod_{n=1}^{N} \ln \mathcal{N}(t_n \mid \vec{w}^T \Phi(x_n), \beta^{-1})$$

Changing $\vec{w}$ you change the curve.

And we want the value of $\vec{w}$ for which the probability distribution is maximum. We solve the problem again by defining an error function:

$$\ln(p(\{t_1, \dots t_N\} \mid x_1, \dots, x_N, \vec{w}, \beta)) =$$

$$= \sum \ln \mathcal{N}(t_n \mid \vec{w}^T \phi(\vec{x}_n), \beta^{-1}) =$$

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

$\rightarrow$ error function: it has negative sign

maximum likelihood $\Rightarrow$ minimization of $E_D(\vec{w})$

In this case it corresponds to least square error minimization, but t_n here are real values

Linear Regression - Algorithms

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

We seek the maximum of the likelihood function:

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

or equivalently:

$$\begin{aligned} \ln P(\{t_1, \ldots, t_N\} | \mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \\ &= -\beta \underbrace{\frac{1}{2} \sum_{n=1}^{N} [t_n - \mathbf{w}^T \phi(\mathbf{x}_n)]^2}_{E_D(\mathbf{w})} - \frac{N}{2} \ln(2\pi\beta^{-1}). \end{aligned}$$

Linear Regression - Algorithms

Maximum likelihood (zero-mean Gaussian noise assumption)

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

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:

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

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

Optimality condition:

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

Hence:

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

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Linear Regression - Algorithms

Sequential Learning

Stochastic gradient descent algorithm:

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

η: learning rate parameter

Therefore:

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

Algorithm converges for suitable small values of η.

Linear Regression - Regularization

Regularization is a technique to control over-fitting.

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

with $\lambda > 0$ being the regularization factor

A common choice:

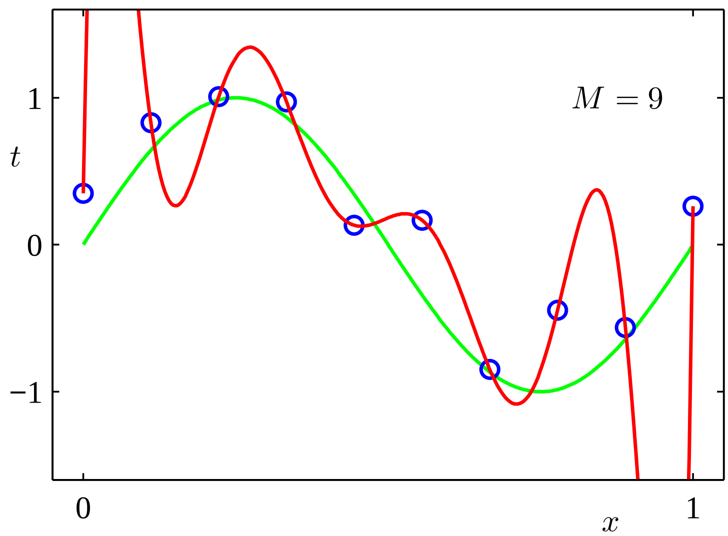
$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}.$$

Other choices:

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

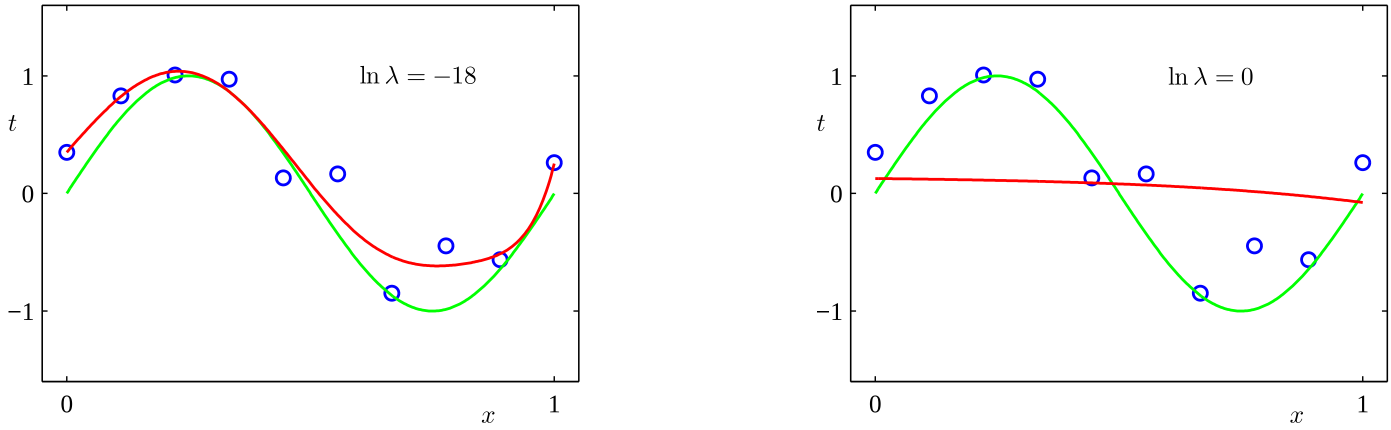
Linear Regression - Regularization

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



Linear Regression - Regularization

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



Linear Regression - Multiple outputs

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

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

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

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

Similarly as before we obtain:

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