Linear regression

ML Instruction Team, Fall 2022

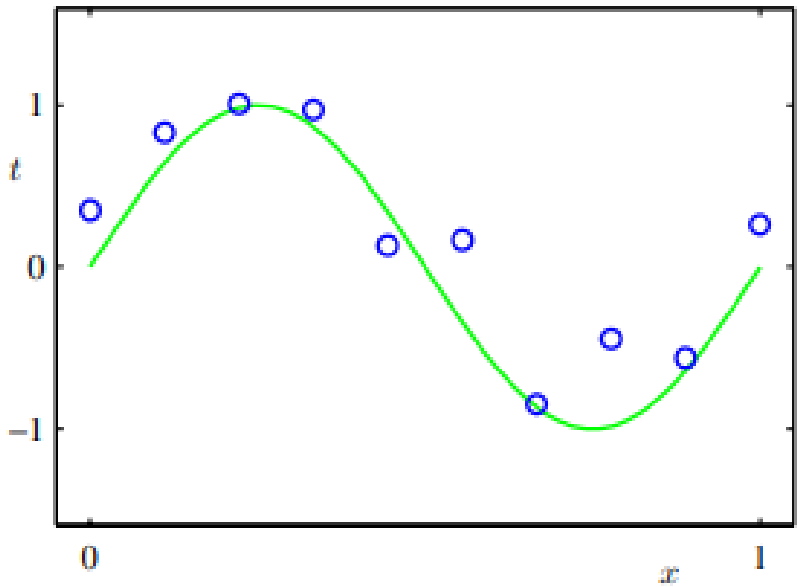
CE Department
Sharif University of Technology

Regression

- The goal of regression is to predict the value of one or more continuous target variables t given the value of D-dimensional vector x of input variables

Polynomial curve fitting

- For an example of curve fitting we will study Polynomial curve fitting. The goal in polynomial curve fitting is to find the ”best” polynomial of degree at most m for the representation of input-output curve. For simplicity we will assume for now that there is only one feature x and a single target value t.
- We will use a synthetic in order to illustrate this the curve is generated by $t = \mathtt{sin}(2\pi x) + \zeta$ where ζ is a centered random variable with standard deviation 0.3.

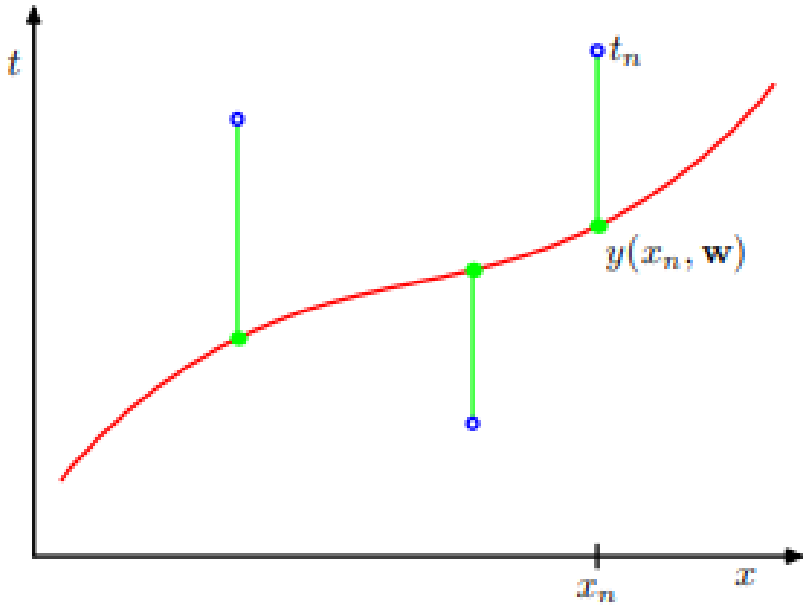


Least squares loss

- In order to formalize the notion of "best" polynomial we need to define a metric to measure the performance of the model. A least squares loss is used to do so let's assume $y(x)$ is the predictive function of the model the aim is to choose y in such a way that the following loss function is minimized:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n) - t_n\}^2$$

- The loss can be interpreted as sum of lengths of each training point from the corresponding point with same x on the curve.



The model

- We will try to solve the optimization problem in the space of polynomials of degree at most M namely:

$$y(x, w) = \sum_{j=0}^{M} w_j x^j$$

- Parameter vector w will be chosen in a way that minimizes the least squares error.

Solving the optimization problem

- To solve the optimization problem we will set the gradient of loss function w.r.t parameter vector w equal to zero.

$$\frac{\partial E(w)}{\partial w_j} = \sum_{n=1}^{N} \frac{\partial (y(x_n, w) - t_n)^2}{\partial w_j} = \sum_{n=1}^{N} 2(y(x_n, w) - t_n) x_n^j$$

setting the derivative equal to zero yields to n linear equations. In order to describe those equations we first introduce some notation.

The Normal equations notation

$$A = \begin{pmatrix} n & \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} x_n^2 & \cdots & \sum_{n=1}^{N} x_n^m \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n^3 & \cdots & \sum_{n=1}^{N} x_n^{m+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum_{n=1}^{N} x_m & \sum_{n=1}^{N} x_n^{m+1} & \sum_{n=1}^{N} x_n^{m+2} & \cdots & \sum_{n=1}^{N} x_n^{2m} \end{pmatrix}$$

$$w = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{pmatrix}, b = \begin{pmatrix} \sum_{n=1}^{N} t_n \\ \sum_{n=1}^{N} t_n x_n \\ \vdots \\ \sum_{n=1}^{N} t_n x_n^M \end{pmatrix}$$

The Normal equations

- The normal equations are then $Aw = b$.
- The matrix A is equal to $X^T X$ and matrix b is equal to $X^T t$ where matrix X, t are defined by :

$$X = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{pmatrix}, t = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}$$

Choosing value of M

- Finding the polynomial with minimum least-squares error still one question remains how to choose M?

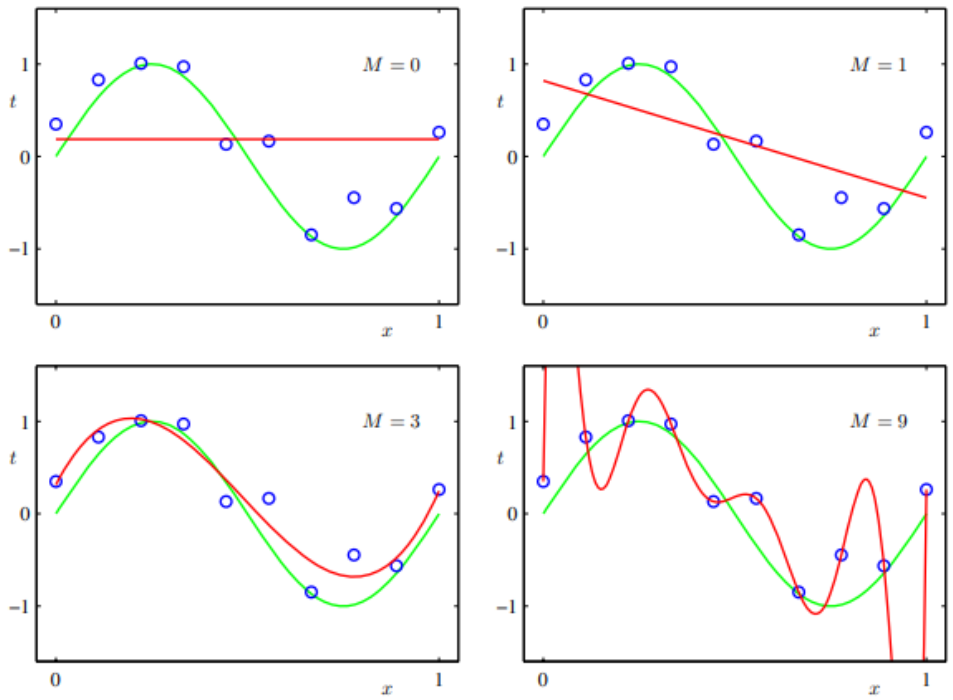


Figure: resulting polynomial for different values of M

Choosing value of M

- Higher values of M leads to more complex models which is not able to generalize well.(Overfitting)
- Small values of M leads to simple models which are unable to capture the structure of the data.(Underfitting)
- The ”optimal” value of M lies somewhere in between.

Performance on test set

- In order to see how well our model generalizes we will use a separate test set consisting of 100 points generated in the same way as before we will use RMS error to be able to compare performance on datasets of different size.

$$E_{RMS} = \sqrt{\frac{2E(w)}{n}}$$

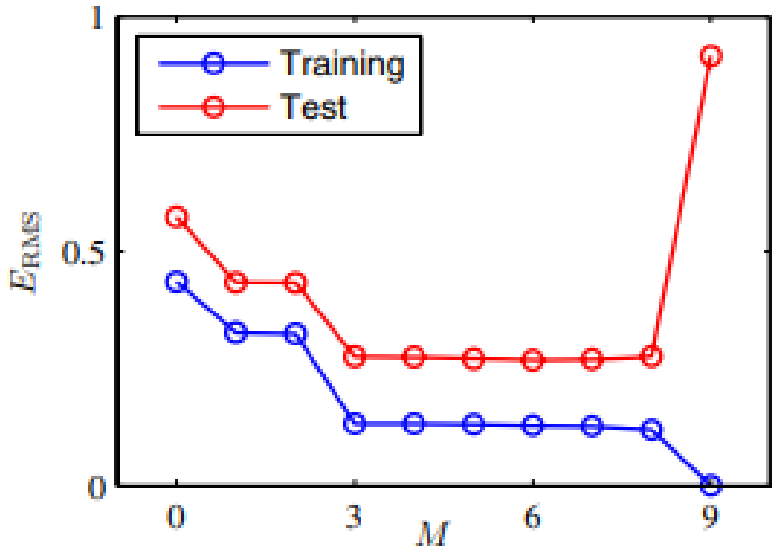


Figure: performance in terms of polynomial degree

Regularization intuition

- Looking at the coefficients generated by best polynomial of degree at most M we realize that when the degree of polynomial is large coefficients will be large in absolute value and usually alternating between positive and negative in order to cancel out effect of previous term. Regularization is an idea for preventing this behaviour by adding vector norm to the loss function.

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Figure: performance in terms of polynomial degree

Regularization

- General idea of regularization is to add an extra term $\lambda w^T w$ to the loss function to ensure that the parameters minimizing the loss will be small in absolute value.

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n) - t_n\}^2 + \frac{\lambda}{2}w^T w$$

- Using lagrange coefficients this is equivalent to minimizing the non-regularized loss function where the parameter space is a cube centered at origin instead of the whole $\mathbb{R}^{m+1}$.
- w_0 is usually omitted in the regularization term because it causes the solution to depend on the choice of origin for the target variable t.

Closed form solution for polynomial regression with regularization

- Setting the derivative equal to zero yields to the equatoins:

$$(X^T X + \lambda I)w^* = X^T t \Rightarrow w^* = (X^T X + \lambda I)^{-1} X^T t$$

  Where X, t are defined as before.
- $X^T X + \lambda I$ is always invertible(it's sum of a positive definite and positive semi-definite matrix) unlike the case without regularization.

Effect of regularization on the coefficients

- Fitting the same polynomial of degree 9 with different regularization parameters results in the coefficients:

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Figure: Polynomial coefficients with different regularization coefficient

- Again the optimal value of λ lies somewhere in between.

Performance with different regularization coefficients

- Performance in the test and train set with different regularization parameter indicate the true value of the parameter λ is neither to big nor to small and should be tuned using a validation set.

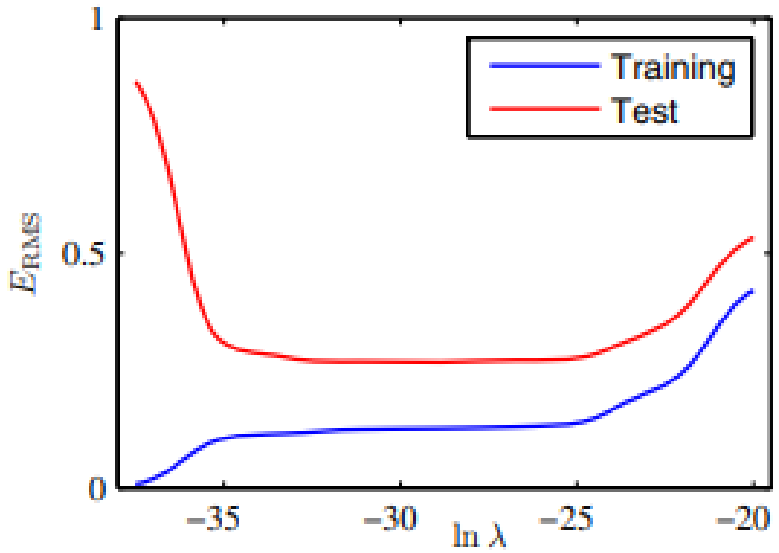


Figure: Effect of regularization parameter on train and test error

Linear regression in general

- In general you may have several inputs instead of one input and you can use other functions than powers of input for basis function. In general we will consider our model as:

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

  where $\phi(x)$ is a predefined function.
- Although the problem is not linear in terms of the input variable it is still linear in terms of adjustable parameter w.

The normal equation

- The design matrix for the linear regression problem is defined as:

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \end{pmatrix}$$

- Setting the derivative of loss function w.r.t w equal to zero we obtain:

$$\Phi^T \Phi w^* = \Phi^T t$$

which is known as normal equations because of it's geometrical interpretation.

Optimizing using gradient descent

- Solving the normal equations is computationally expensive($O(m^2 n)$). Using gradient descent we use $O(mn)$ time for each update.
- The update rule will be:

$$w^{(k)} = w^{(k-1)} - \eta \Phi^T (\Phi w - t)$$

where η is the learning rate.

- One can also use mini-batch gradient descent or stochastic gradient descent to optimize the loss function.

Overfitting and adding regularization term

- As in the polynomial regression we can overfit the data by choosing a complex model.
- Again regularization can be used to prevent overfitting the data in which case the normal equations will become:

$$(\Phi^T\Phi + \lambda I)w^* = \Phi^T t \Rightarrow w^* = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T t$$

References

- Christopher M. Bishop, Pattern recognition and machine learning

Thank You!

Any Question?