Polynomial Curve Fitting

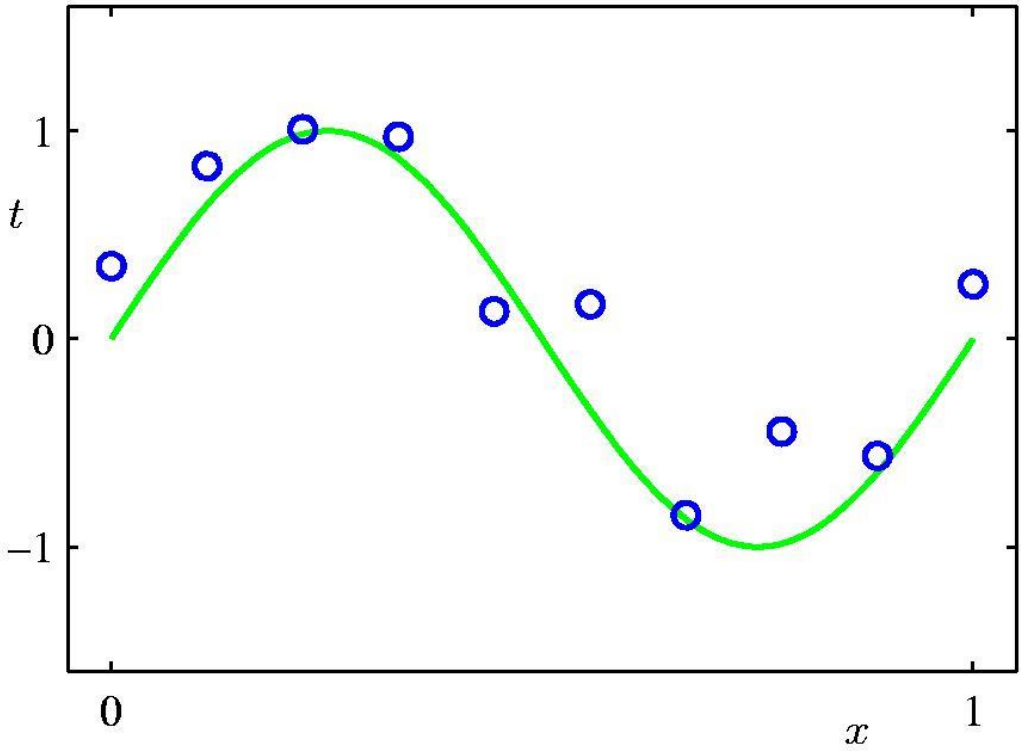
Polynomial Curve Fitting

A Simple Regression Problem

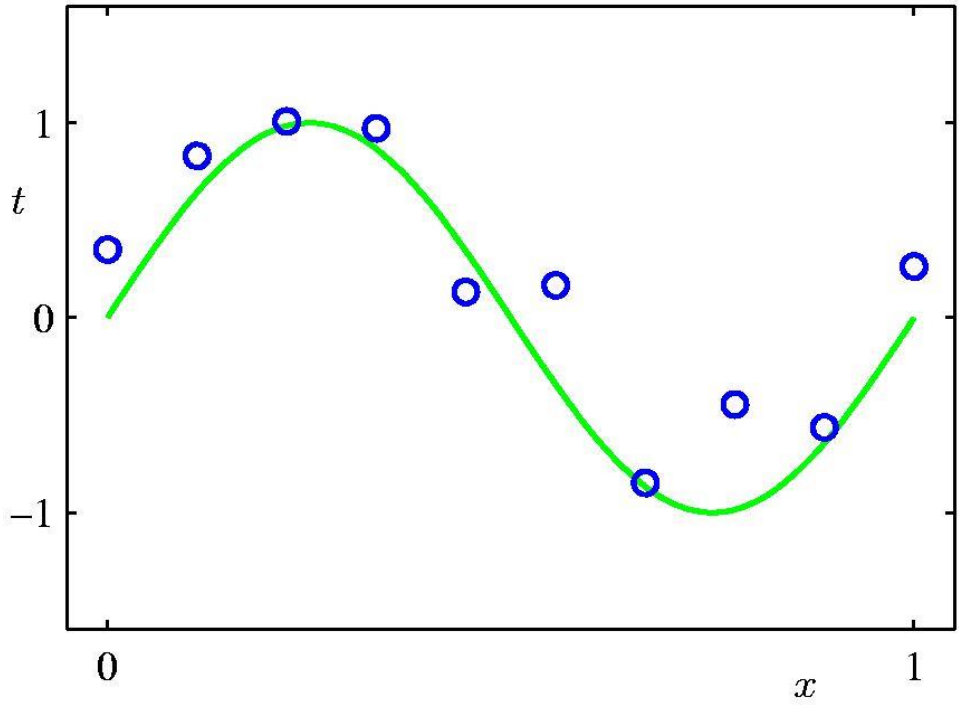
- We observe a real-valued input variable x and we wish to use this observation to predict the value of a real-valued target variable t.
- We use synthetically generated data from the function $\sin(2\pi x)$ with random noise included in the target values.
  - A small level of random noise having a Gaussian distribution
- We have a training set comprising N observations of x, written $x \equiv (x_1, \ldots, x_N)^T$, together with corresponding observations of the values of t, denoted $t \equiv (t_1, \ldots, t_N)^T$.
- Our goal is to predict the value of t for some new value of x,

Polynomial Curve Fitting

- A training data set of N = 10 points, (blue circles),
- The green curve shows the actual function sin(2πx) used to generate the data.
- Our goal is to predict the value of t for some new value of x, without knowledge of the green curve.



Polynomial Curve Fitting

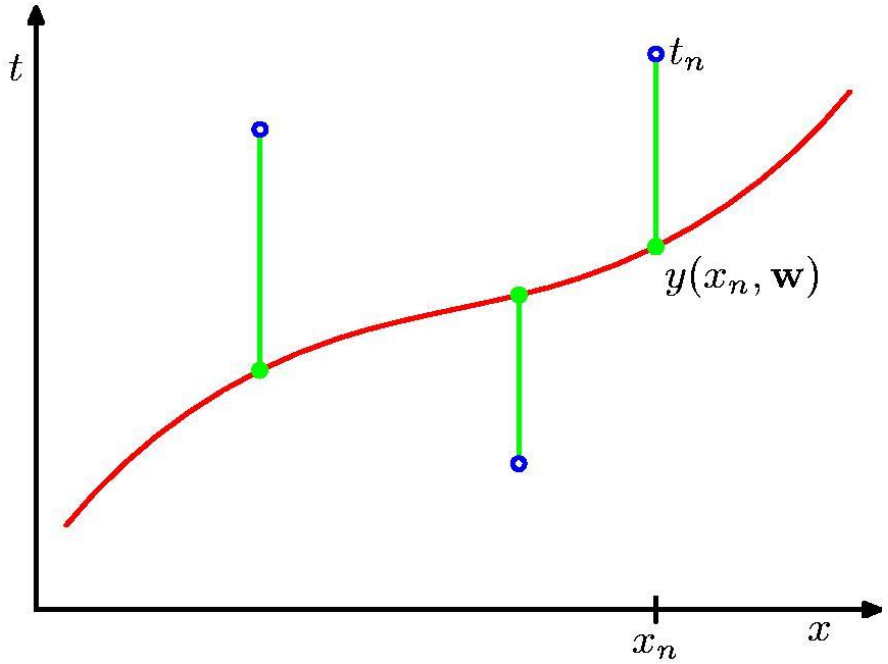


- We try to fit the data using a polynomial function of the form

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

Polynomial Curve Fitting

- The values of the coefficients will be determined by fitting the polynomial to the training data.
- This can be done by minimizing an error function that measures the misfit between the function y(x,w), for any given value of w, and the training set data points.
- Error Function: the sum of the squares of the errors between the predictions $y(x_n,w)$ for each data point x_n and the corresponding target values t_n.

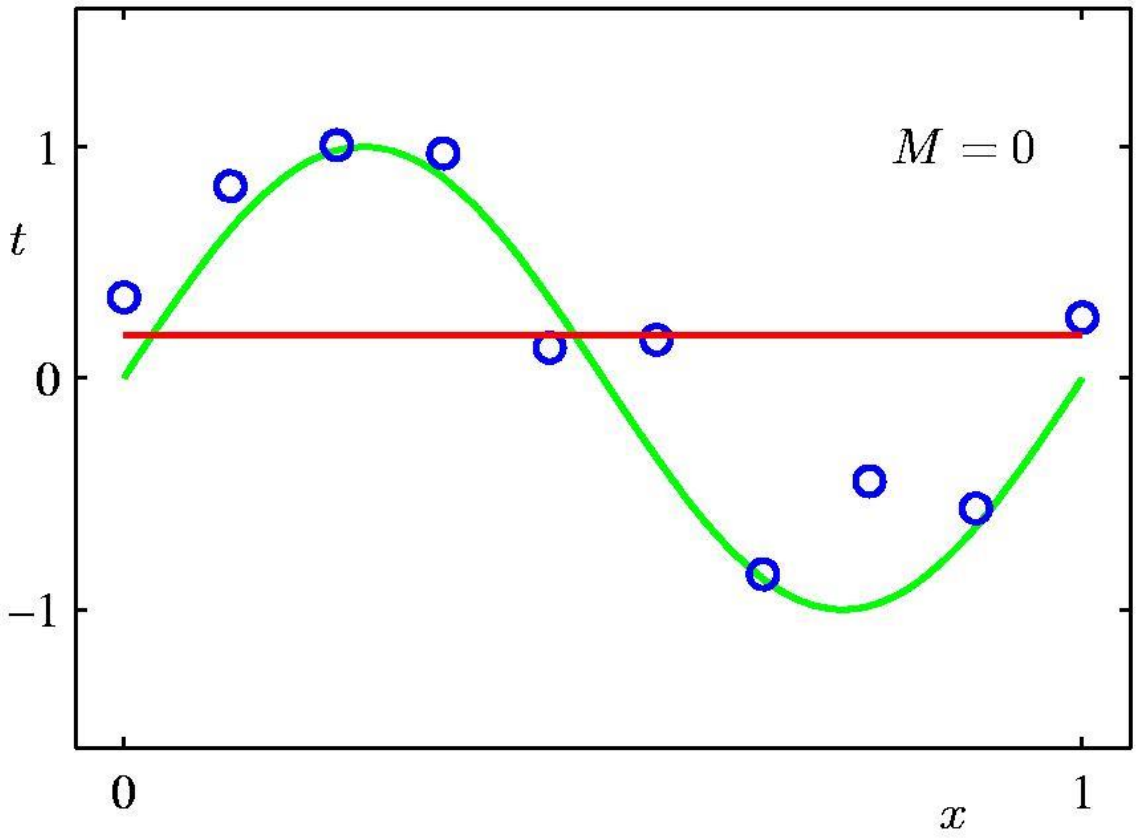


$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

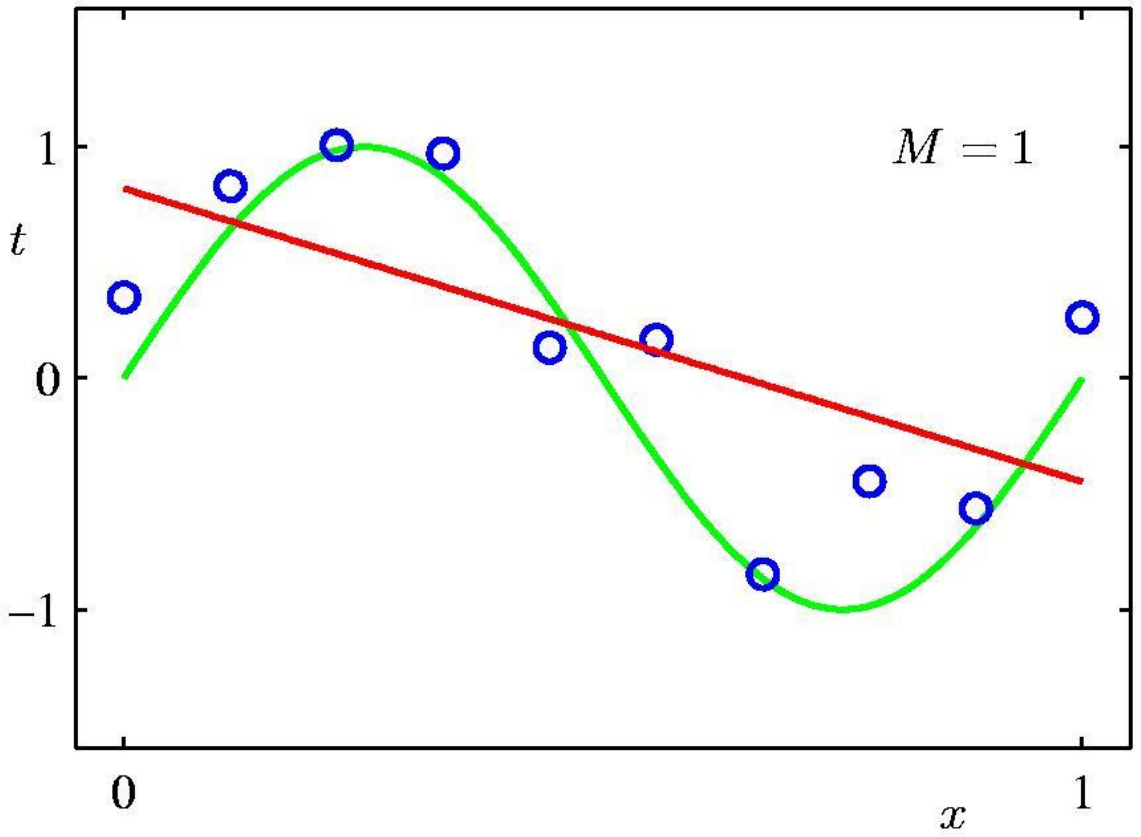
Polynomial Curve Fitting

- We can solve the curve fitting problem by choosing the value of w for which E(w) is as small as possible.
- Since the error function is a quadratic function of the coefficients w, its derivatives with respect to the coefficients will be linear in the elements of w, and so the minimization of the error function has a unique solution, denoted by w*,
- The resulting polynomial is given by the function y(x,w*).
- Choosing the order M of the polynomial ➔ model selection.

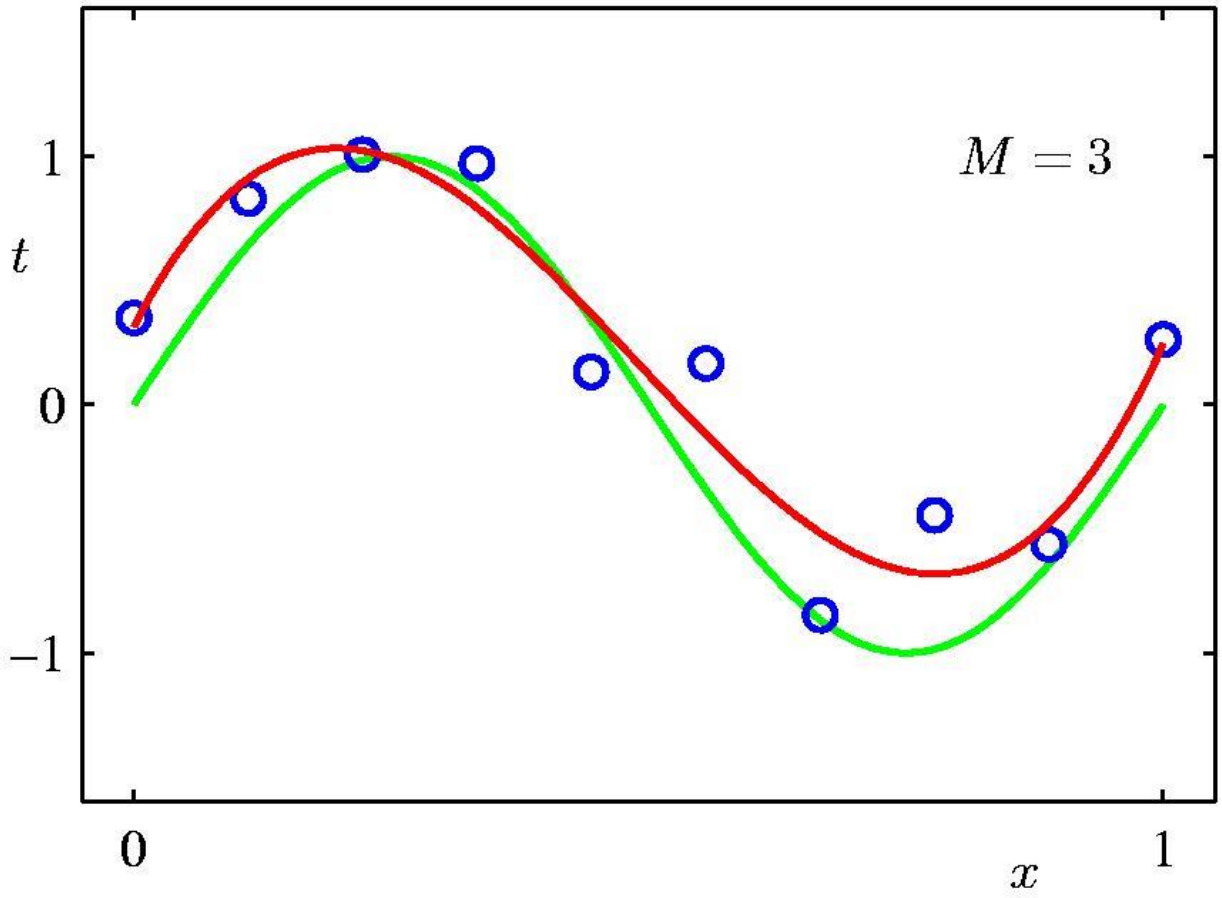
0^{th} Order Polynomial



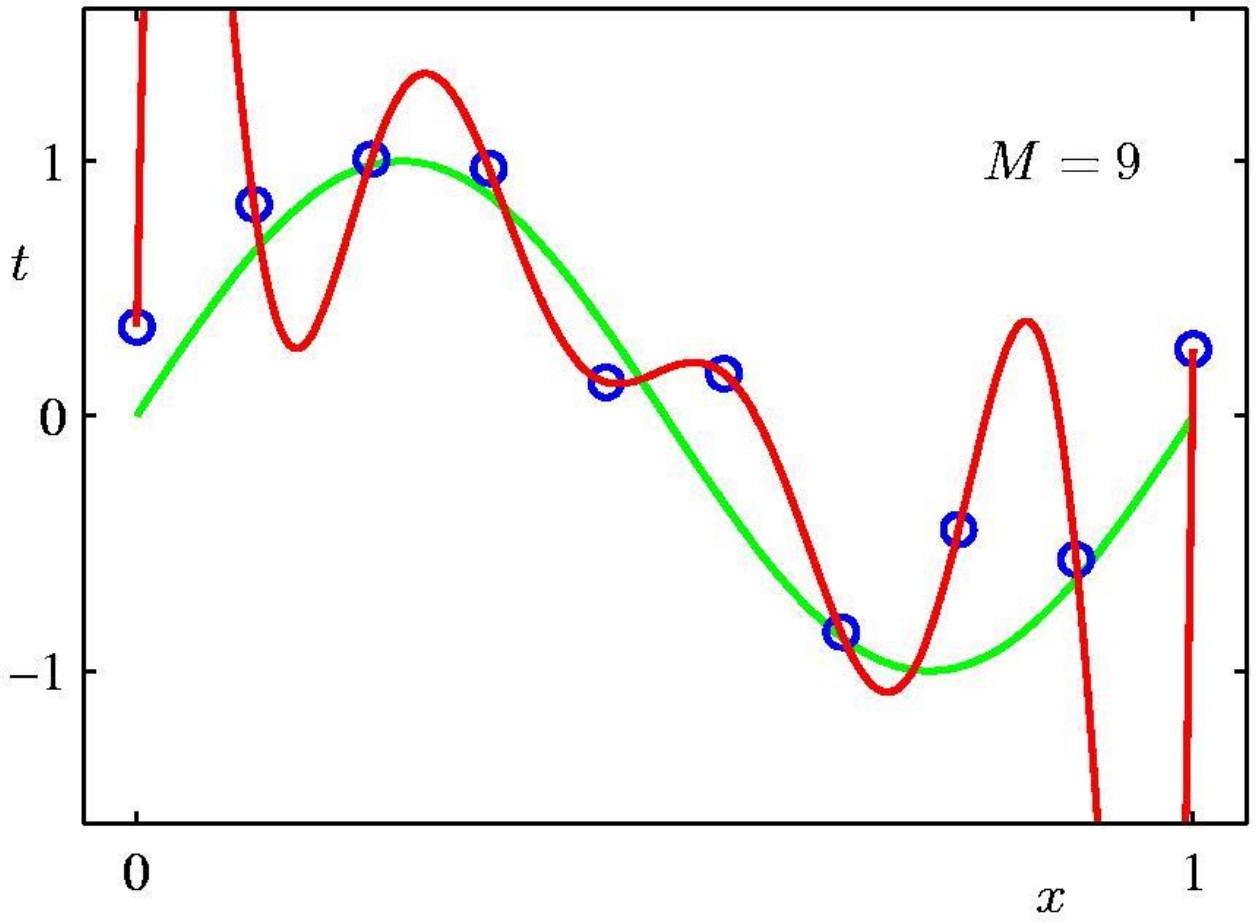
1st Order Polynomial



3rd Order Polynomial

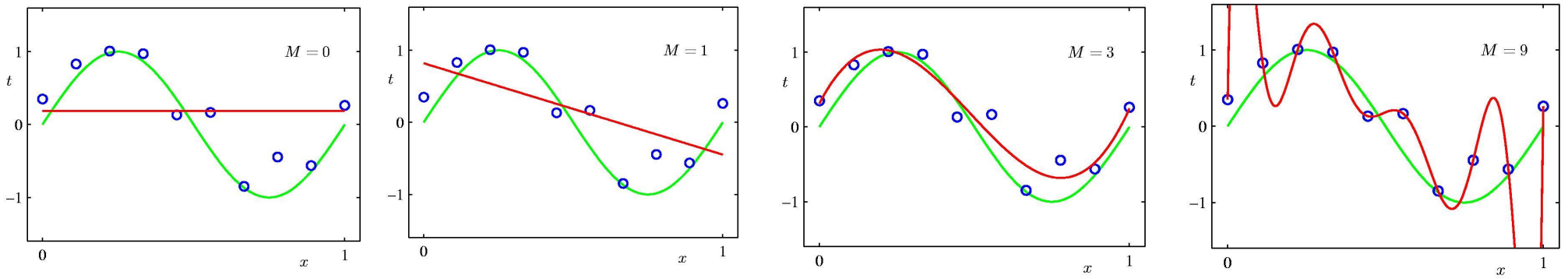


9th Order Polynomial



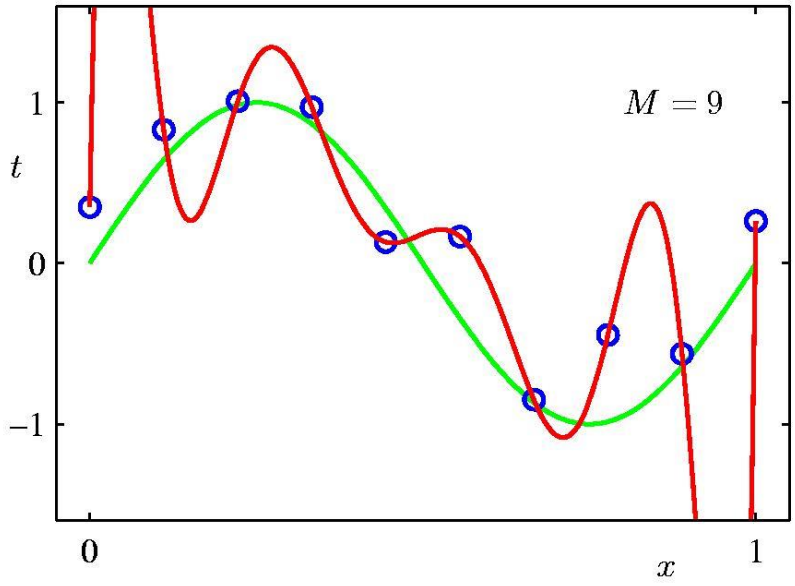
Polynomial Curve Fitting

- The 0^{th} order (M=0) and first order (M=1) polynomials give rather poor fits to the data and consequently rather poor representations of the function sin(2πx).
- The third order (M=3) polynomial seems to give the best fit to the function sin(2πx) of the examples.
- When we go to a much higher order polynomial (M=9), we obtain an excellent fit to the training data.
  - In fact, the polynomial passes exactly through each data point and E(w*) = 0.



Polynomial Curve Fitting

- We obtain an excellent fit to the training data with 9th order.
- However, the fitted curve oscillates wildly and gives a very poor representation of the function sin(2πx).

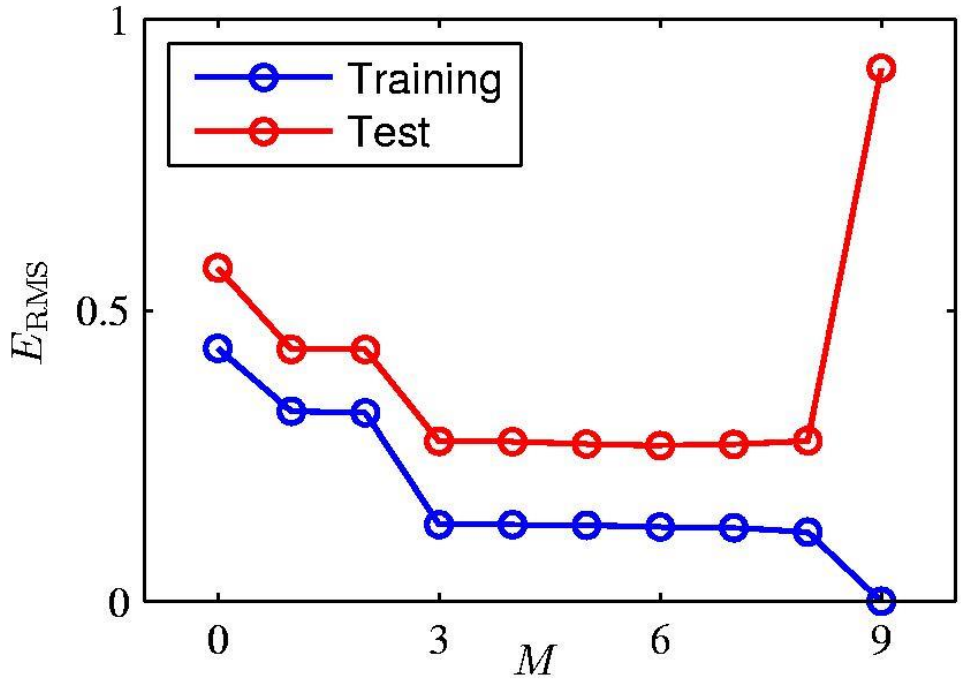


- This behaviour is known as **over-fitting**

Polynomial Curve Fitting

Over-fitting

- We can then evaluate the residual value of E(w*) for the training data, and we can also evaluate E(w*) for the test data set.
- Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$
  - in which the division by N allows us to compare different sizes of data sets, and the square root ensures that E_{RMS} is measured on the same scale as the target variable t.



Polynomial Curve Fitting

Polynomial Coefficients

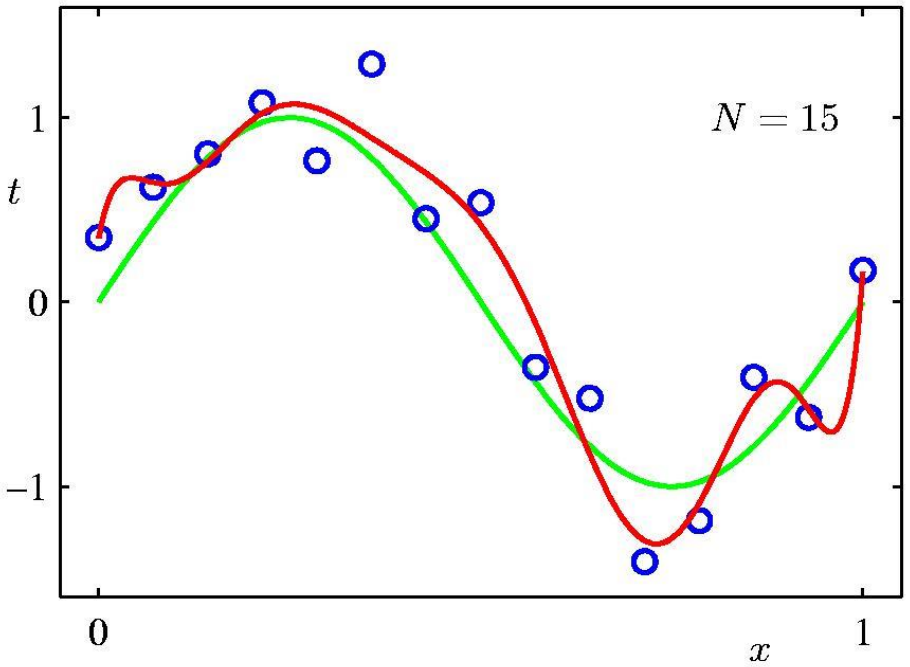
- Magnitude of coefficients increases dramatically as order of polynomial increases.
- Large positive and negative values so that the corresponding polynomial function matches each of the data points exactly, but between data points the function exhibits the large oscillations ➔ over-fitting

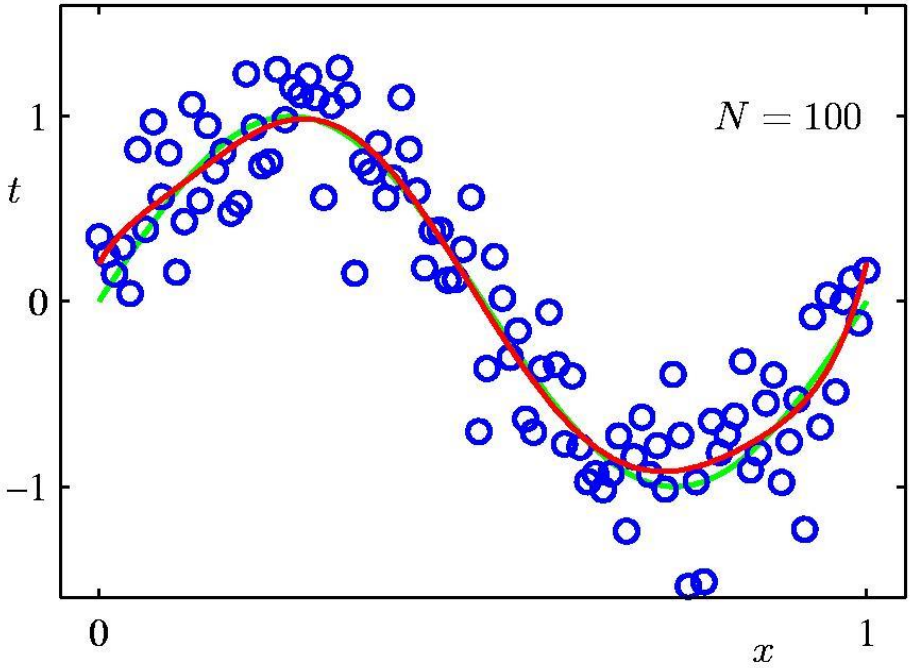
	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Polynomial Curve Fitting

- Increasing the size of the data set reduces the over-fitting problem.

- 9th Order Polynomial.





Polynomial Curve Fitting

regularization

- We may wish to use relatively complex and flexible models with data sets of limited size.
- The over-fitting phenomenon can be controlled with **regularization**, which involves adding a penalty term to the error function.

- **Regularization:** Penalize large coefficient values

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\text{where } \|\mathbf{w}\|^2 \equiv \mathbf{w}^{\mathrm{T}}\mathbf{w} = w_0^2 + w_1^2 + \ldots + w_M^2$$

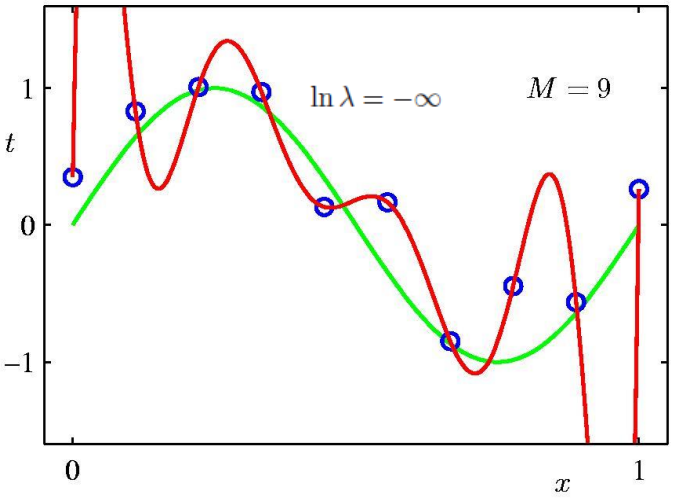
- the coefficient λ governs the relative importance of the regularization term compared with the sum-of-squares error term.

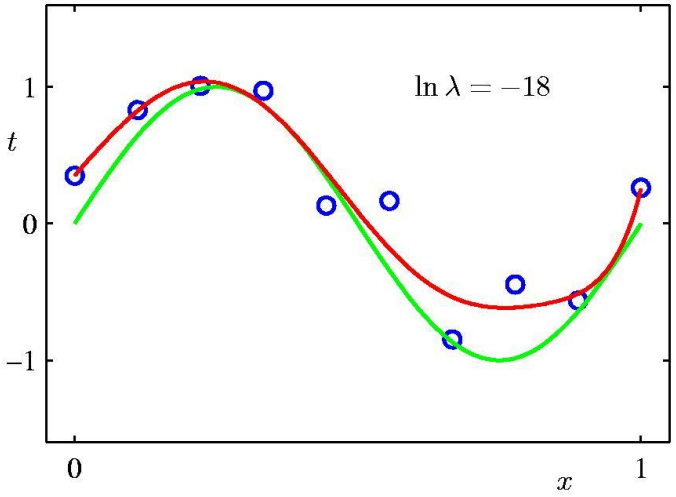
Polynomial Curve Fitting

regularization

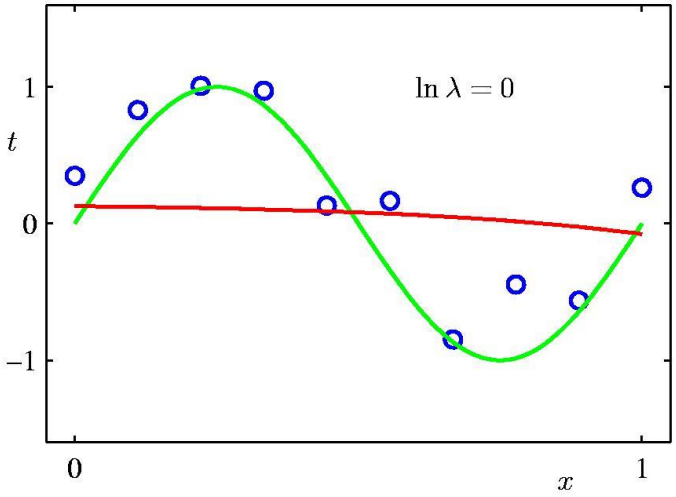
- Plots of M = 9 polynomials fitted to the data set using the regularized error function

no regularization (λ=0)





too much regularization

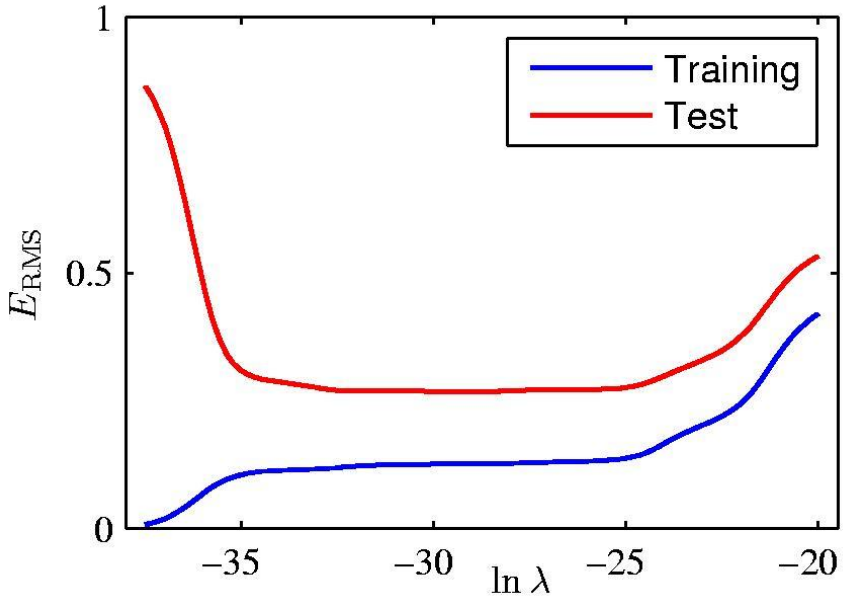


Polynomial Curve Fitting

regularization

- Polynomial 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



Graph of the root-mean-square error versus ln λ for the M=9 polynomial.