Extending linear regression algorithm with basis functions

- Original form:

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

- Using basis functions instead:

$$h(x) = \sum_{i=0}^{P} \theta_i \, \phi_i(x)$$

Ordinary linear regression as a special case

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

$$h(x) = \sum_{i=0}^{P} \theta_i \phi_i(x)$$

- The original form can be obtained by having $P = n$ and the following basis functions:

$\phi_0(x) = 1$ $\quad \phi_1(x) = x_1$ $\quad \phi_i(x) = x_i$

Polynomial Curve Fitting

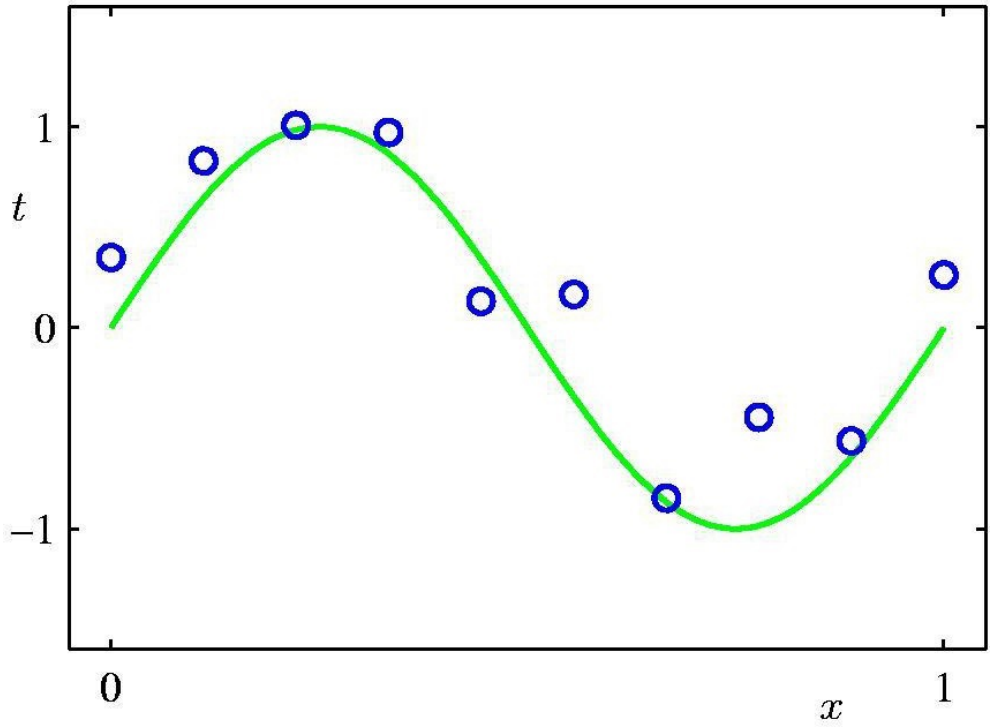
$\phi_0(x) = 1$

$\phi_1(x) = x_1$

$\phi_2(x) = x_1^2$

$\vdots$

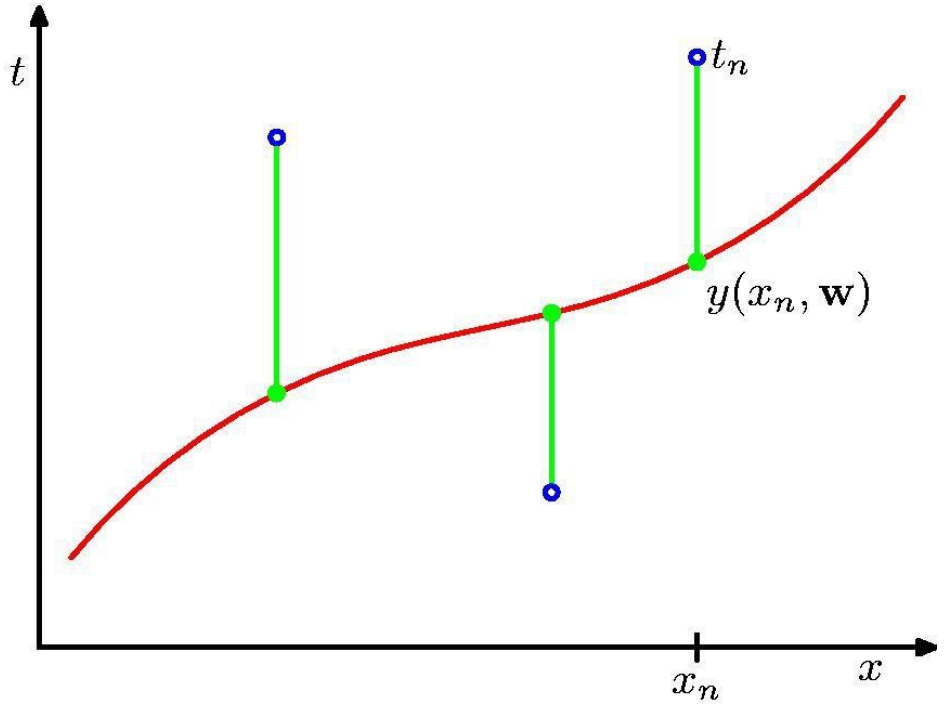
$\phi_j(x) = x_1^j$



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

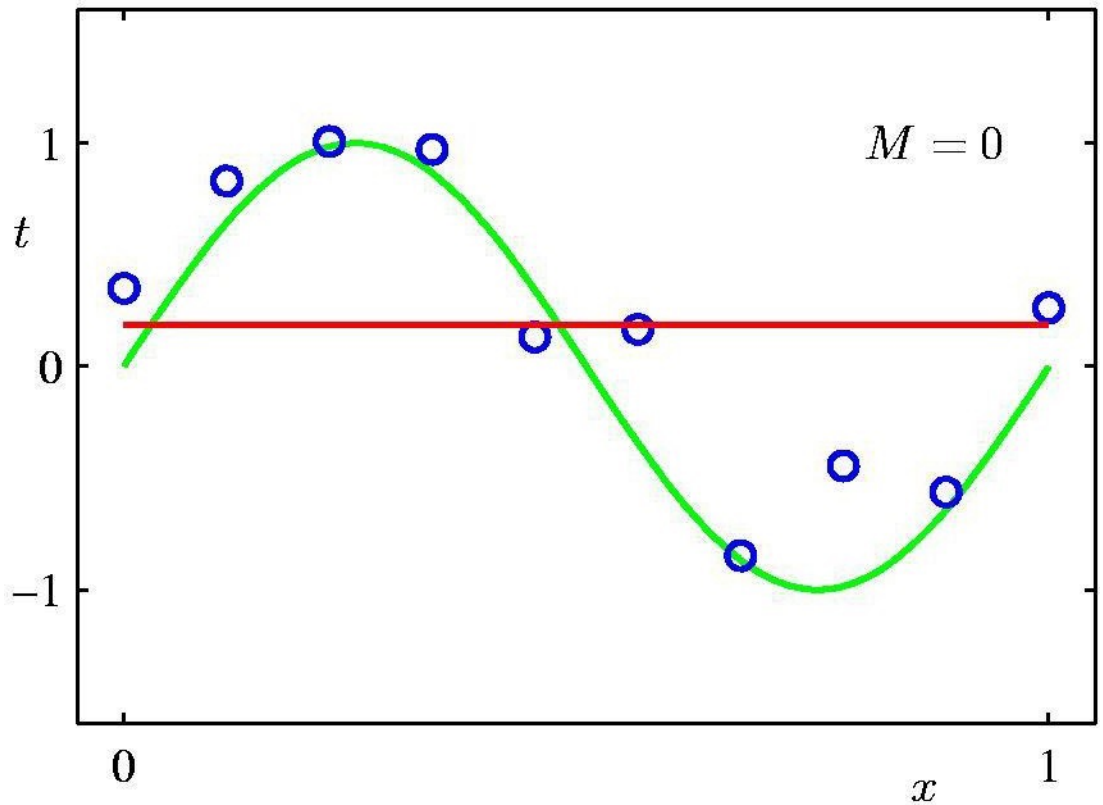
$\theta = w$

Sum-of-Squares Error Function

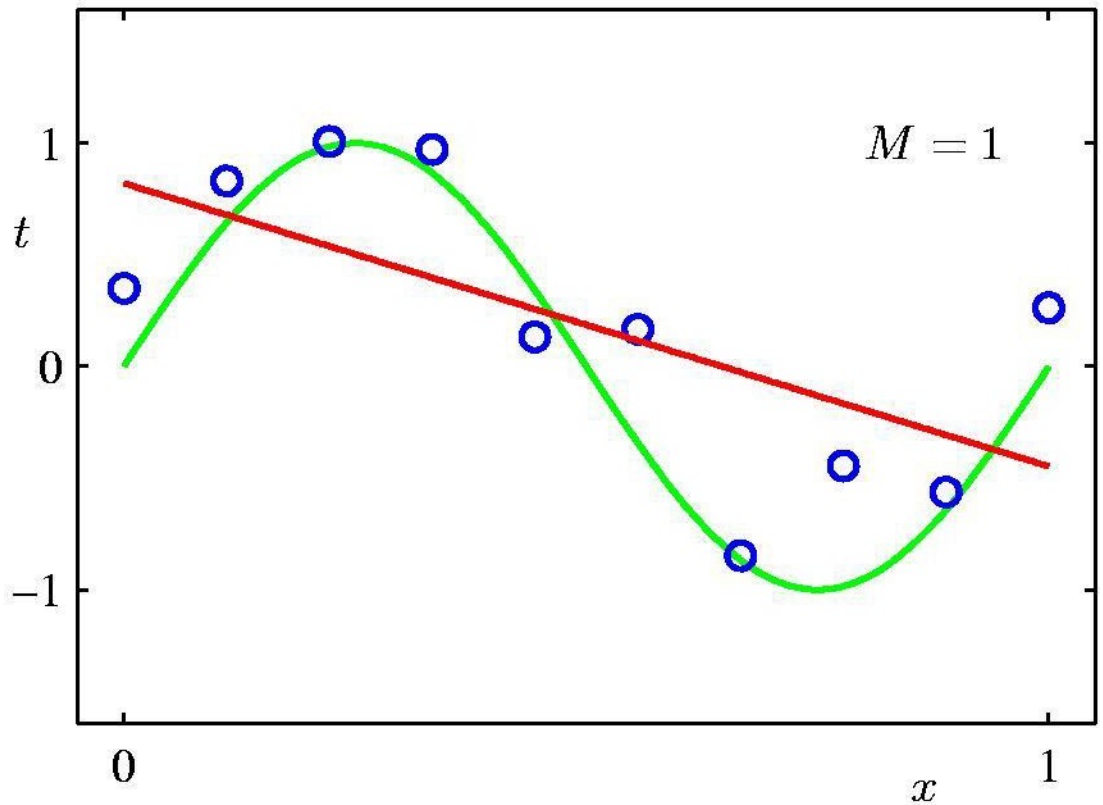


$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

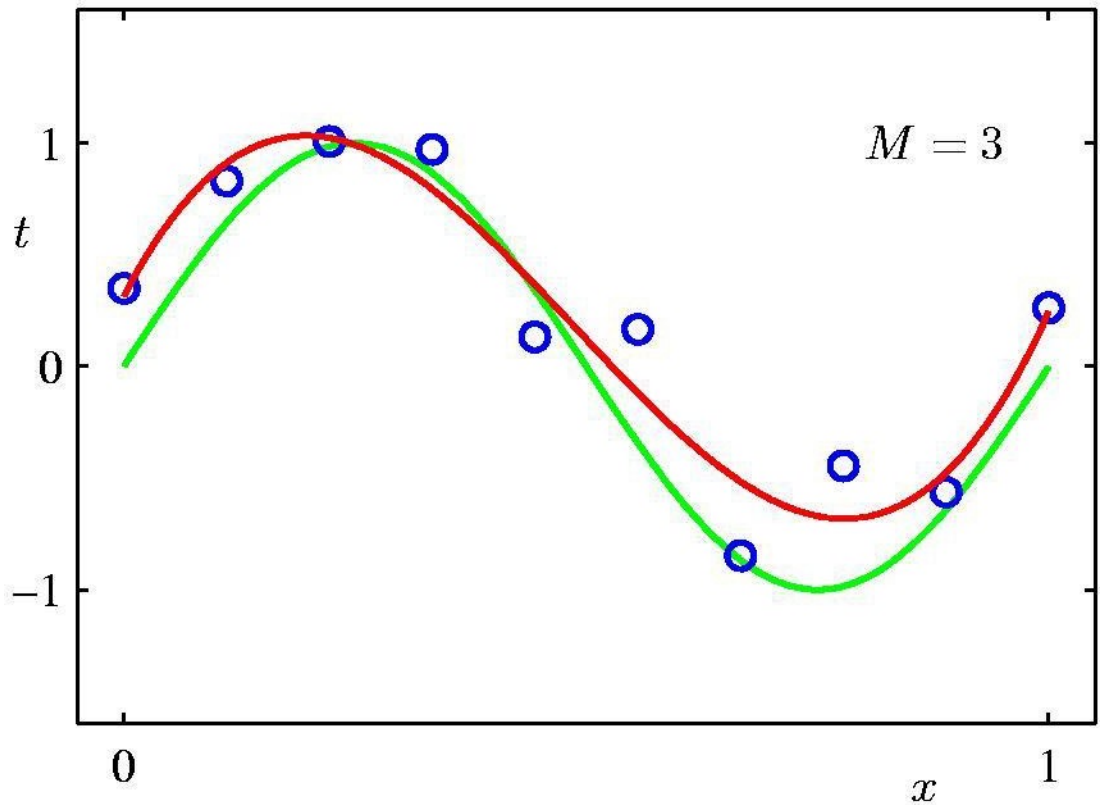
0^{th} Order Polynomial



1st Order Polynomial



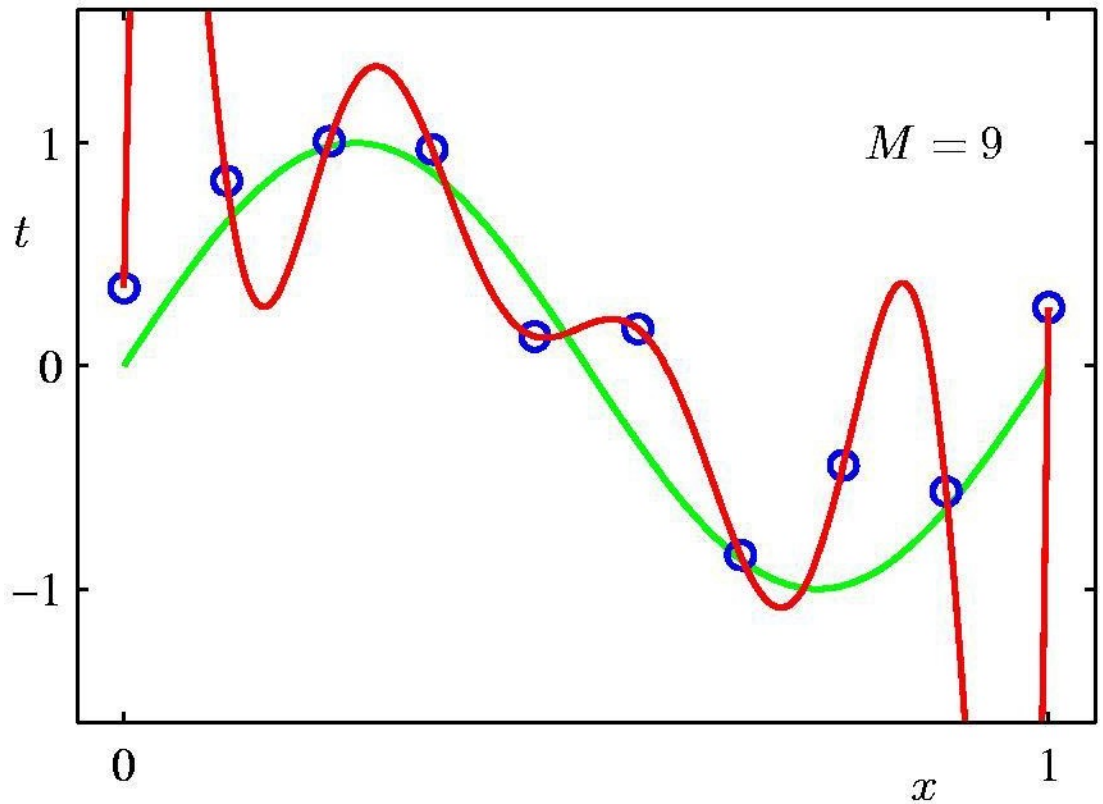
3rd Order Polynomial x^2 x^3



Polynomial regression in Scikit Learn

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> from sklearn.linear_model import LinearRegression
>>> from sklearn.pipeline import Pipeline
>>> import numpy as np
>>> model = Pipeline([('poly', PolynomialFeatures(degree=3)),
...                   ('linear', LinearRegression(fit_intercept=False))])
>>> # fit to an order-3 polynomial data
>>> x = np.arange(5)
>>> y = 3 - 2 * x + x ** 2 - x ** 3
>>> model = model.fit(x[:, np.newaxis], y)
>>> model.named_steps['linear'].coef_
array([ 3., -2.,  1., -1.])
```

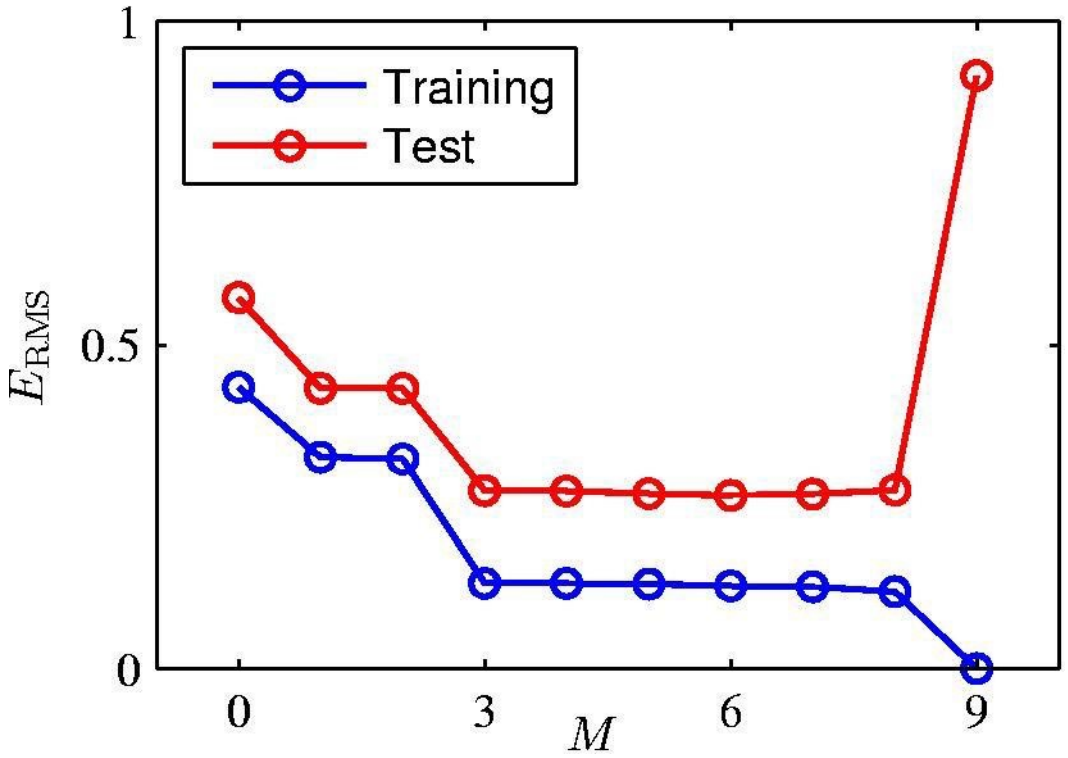
9th Order Polynomial



Over-fitting

if we put sometest data aside, the error in test data will increase

need to adjust complexity of the model, start with smal valuesof polynomial and start to slowly increaseit to see when test goes up



Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$

Polynomial Coefficients

todo, make notes from echo for above^^, which is 29th march but 1hour - 1:15 section

m=0 only need one weight

m=1 only needs 2 w

m = 3 needs 4 w

m = 9 w = 9, where w9 is a very big numbers

as m increase the magnitude of needed weights w increases

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	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

when you have a higher degree m, then need lots of correcting

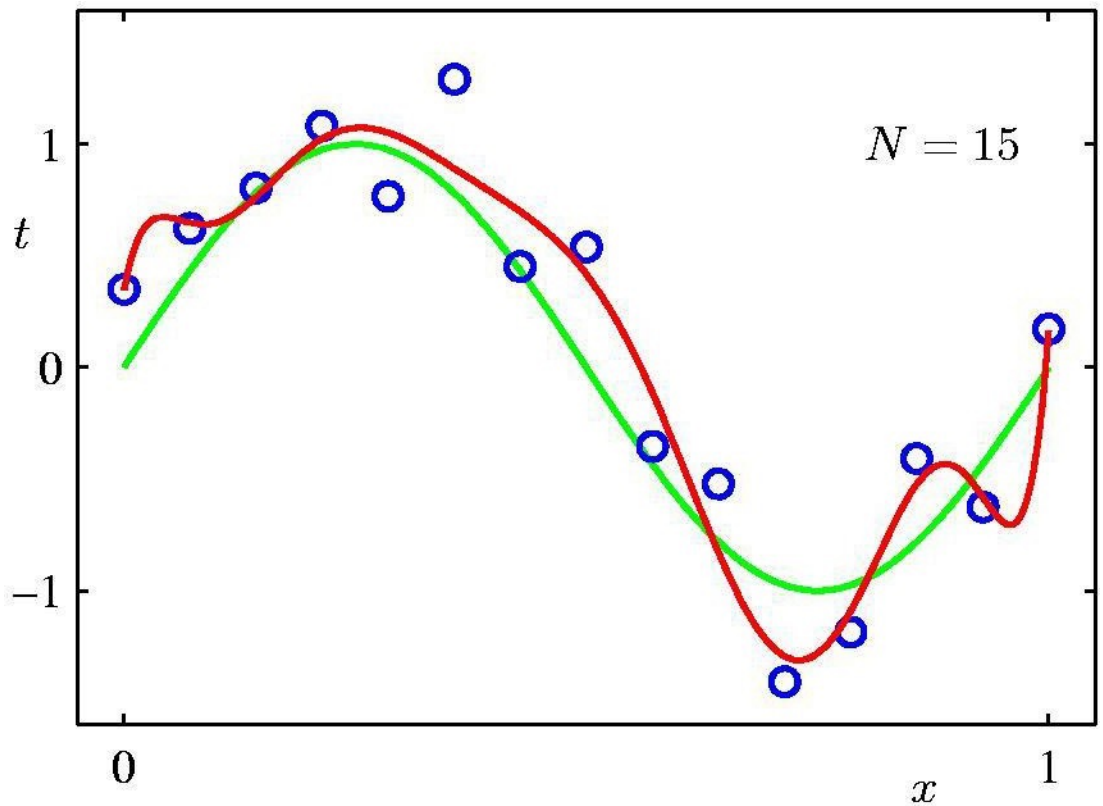
to see which of thses features of x^9 has been c to predict data, have 2^9 different choices.

9th degree polynomial with 15 points, more points the more closer you will get, bt will get large values for the coefficient

greed

Data Set Size: $N = 15$

9th Order Polynomial

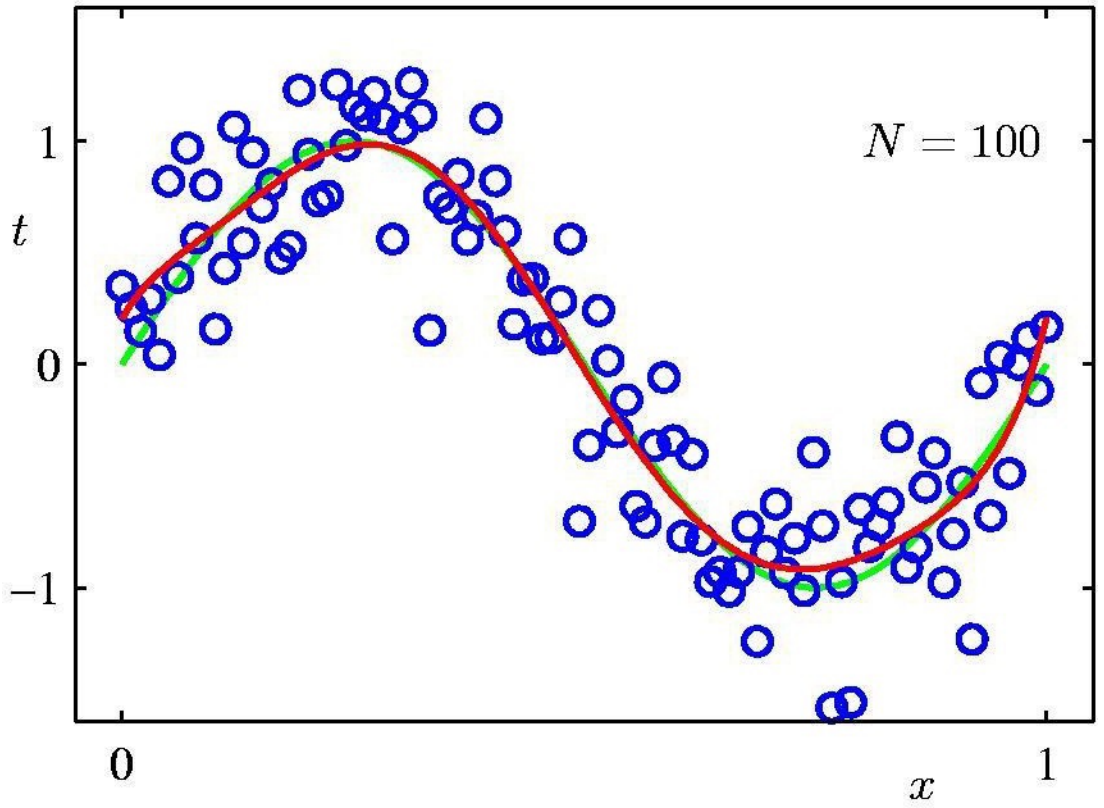


Data Set Size: $N = 100$

9th Order Polynomial

greedy approach, forward selection and backward selection.

dropping those that can maximise your performance - backwrd selection



Regularization

previously defined error as the sum of square error function, E(w), that measure all training data using y instead of h, an dhow differnt it is from your target value t

Penalize large coefficient values

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

hyper-parameter regularization Parameter

difference between prediction x and the actual value t

hyper parameter in decision tree, max dept, end point, or other hyper parameters you can introduce.

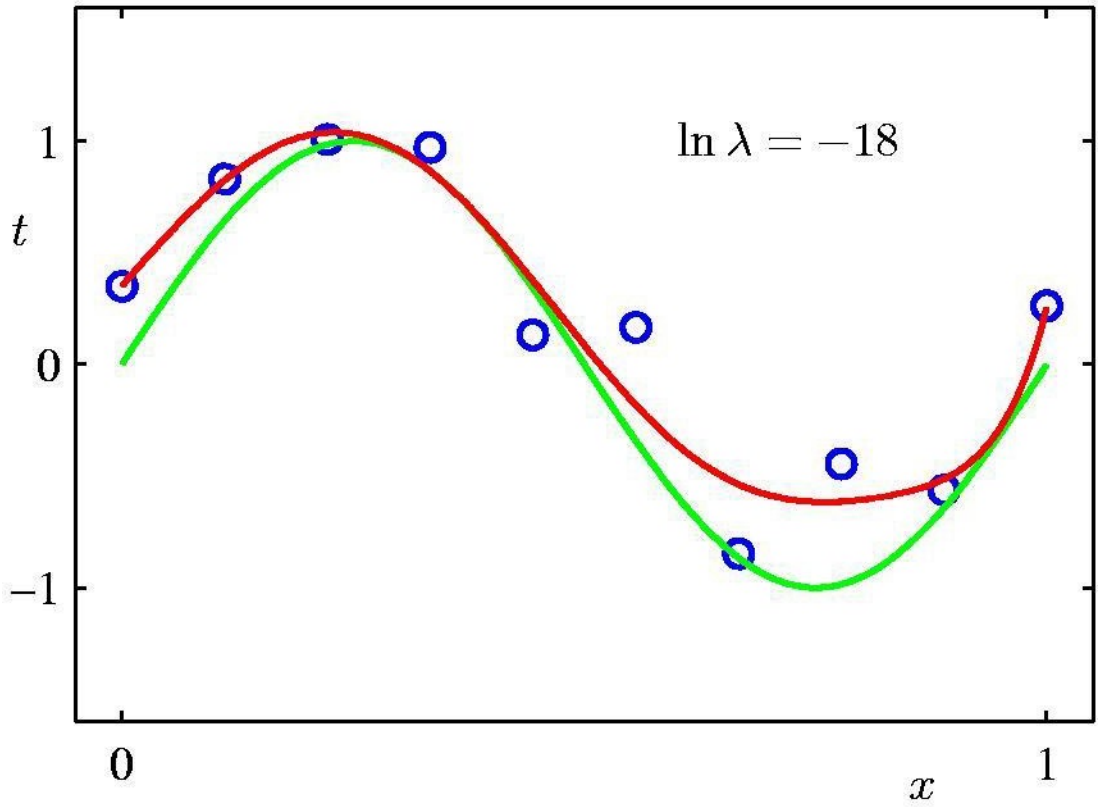
this objective function is what you have and difference between x and t, with an additional magnitude ||w||^2, this is size of the vector, so each component squared. a lamba hyper parameter

adding a new term to the cost, total magnitude for all the weight (eg. coming from all the w of the polynomial coefficient table). this can hurt the fitting coz the more w there is the bigger the magnitude so this is a trade off.

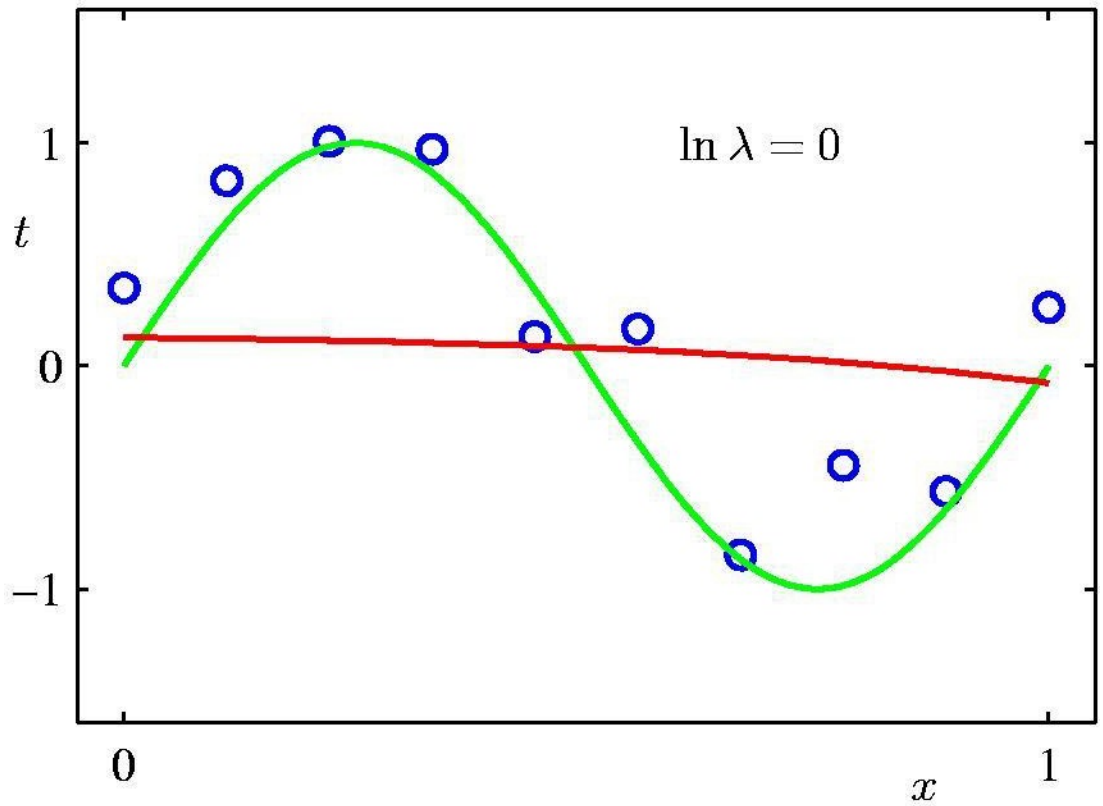
the lambda is for control of the new term. lambda = 0 can remove the magnitude effect, is lambda is larger, then can put more emphasis on the magnitude.

low lambda of - 18, e^-18

Regularization: $\ln \lambda = -18$



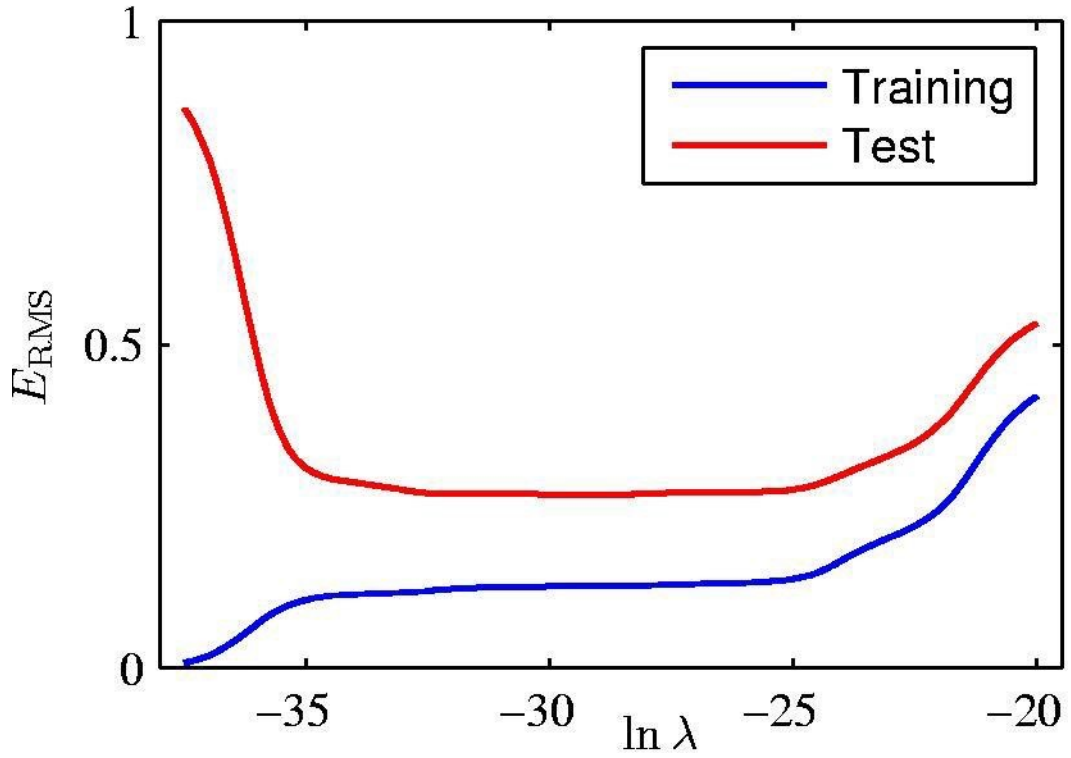
Regularization: $\ln \lambda = 0$



Regularization:

$||w||^2$

E_{RMS} **VS.** $\ln \lambda$



Polynomial Coefficients

the bias function to feature map/engineering treat polynomal and hand craft features

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	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

increase value of lambda, the coefficient decrease