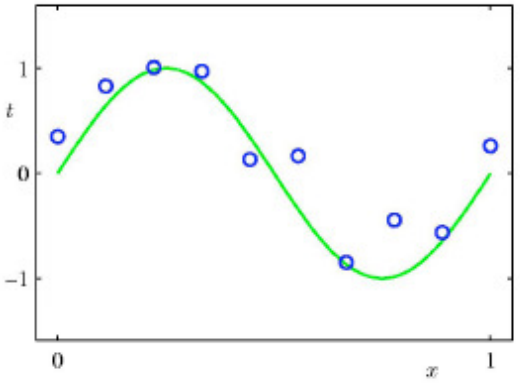
Basics of regression

Problem statement

- N observations of x

  $\mathbf{x} = (x_1,..,x_N)^T$

  $\mathbf{t} = (t_1,..,t_N)^T$
- Goal is to exploit training set to predict value of $\hat{t}$ from x
- Inherently a difficult problem
- Probability theory allows us to make a prediction



Data Generation:

N = 10

Spaced uniformly in range [0,1]

Generated from $sin(2\pi x)$ by adding small Gaussian noise

Noise typical due to unobserved variables

Polynomial fitting (for illustration)

- Polynomial function

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

- Where M is the order of the polynomial

- Coefficients $w_0, \ldots w_M$ are denoted by vector $\mathbf{w}$
- Nonlinear function of x, linear function of coefficients $\mathbf{w}$
- Called Linear Models

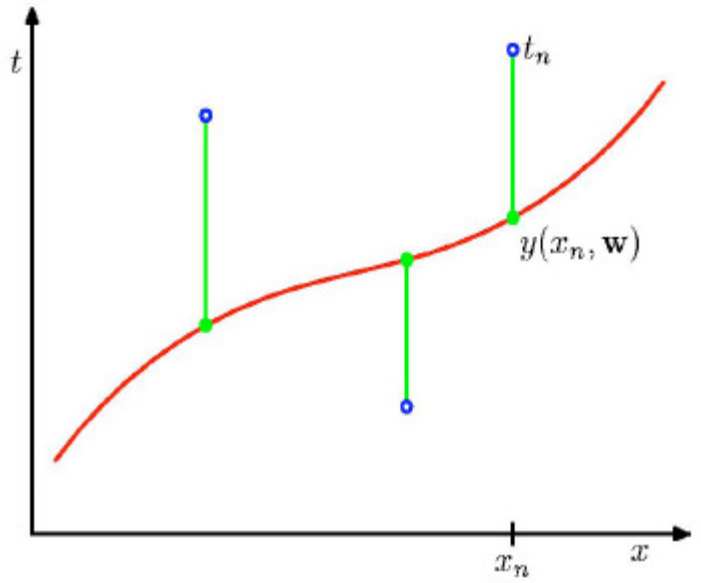
Polynomial fitting (for illustration)

- Sum of squares of the errors between the predictions $y(x_n, \mathbf{w})$ for each data point x_n and target value t_n

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

- Solve by choosing value of w for which $E(\mathrm{w})$ is as small as possible

Red line is best polynomial fit



- Note that the regressor minimizes:

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

This equation relates to empirical risk minimization --→ Minimizing the error on the training samples

Learning the weights

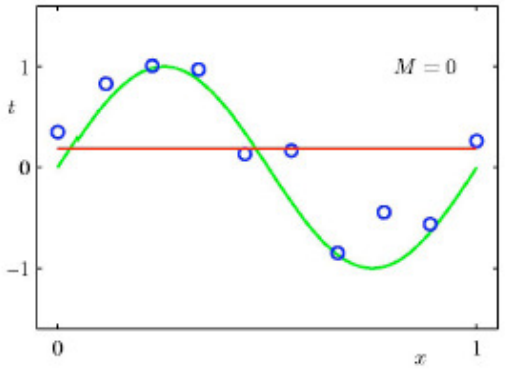
- Error function is a quadratic in coefficients $\mathbf{w}$
- Derivative with respect to coefficients will be linear in elements of $\mathbf{w}$
- Thus error function has a unique minimum denoted $\mathbf{w}^*$
- Resulting polynomial is $y(x, w^*)$

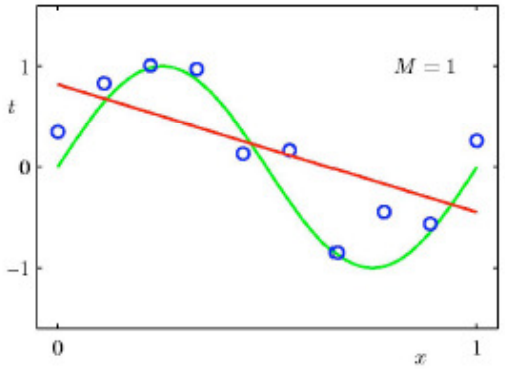
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Use of Simple Model

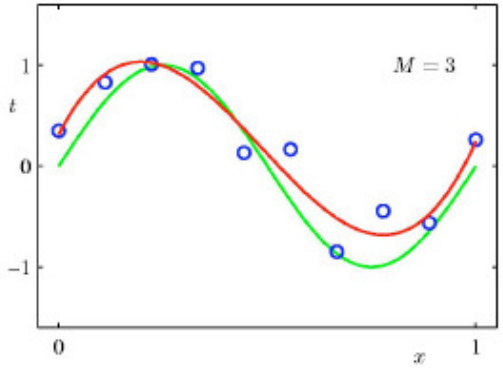
Choosing the order of M

- Model Comparison or Model Selection
- Red lines are best fits with
  - $M = 0,1,3,9$ and $N=10$

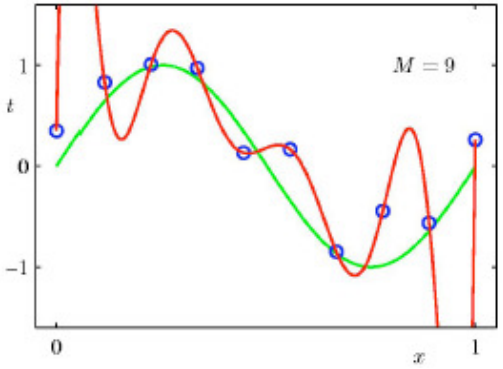




← Poor representations of $\sin(2\pi x)$



← Best Fit to $\sin(2\pi x)$



Over Fit Poor representation of $\sin(2\pi x)$

Performance of regression

- Consider separate *test* set of *100* points
- For each value of M evaluate

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

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

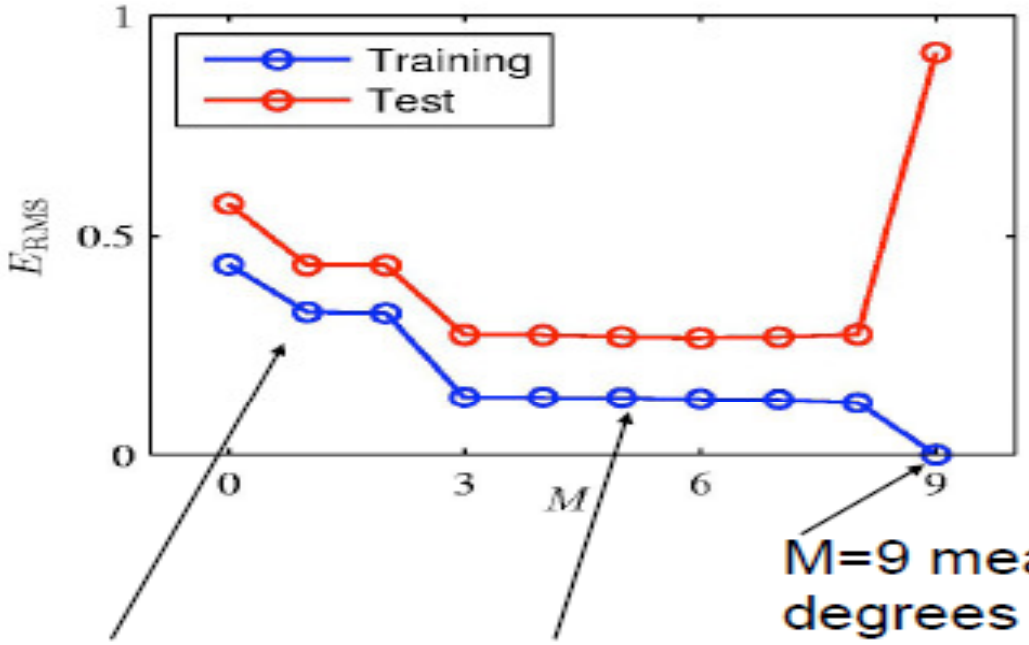
for training data and test data

- Use RMS error

$$E_{RMS} = \sqrt{2E(w^*)/N}$$

  - Division by N allows different sizes of N to be compared on equal footing
  - Square root ensures E_{RMS} is measured in same units as t

Performance of regression



Poor due to Inflexible polynomials

Small Error

M=9 means ten degrees of freedom. Tuned exactly to 10 training points (wild oscillations in polynomial)

Complexity of the model

- When we are fitting a higher order model, with limited training samples, we are making the models ---'complex'

- There arises a problem of over fitting….and the curse of dimensionality in a way would plague in making the performance deteriorate on the test set.

- Complex models have large oscillations in the learned weights.

Trend of weights with different orders

Values of Coefficients w^* for different polynomials of order M

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	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

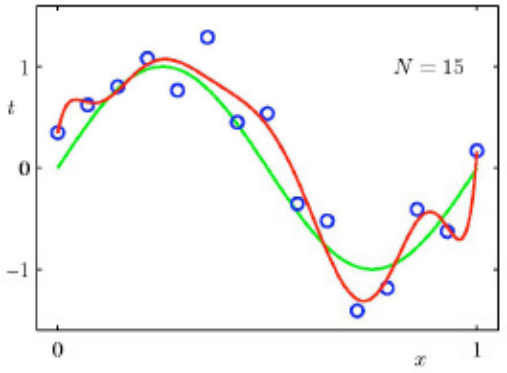
As M increases magnitude of coefficients increases
At $M=9$ finely tuned to random noise in target values

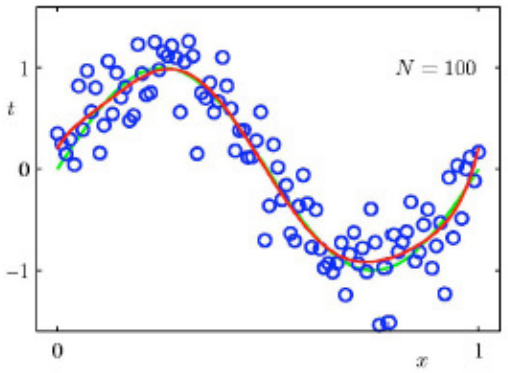
Techniques for controlling the overfitting

Increasing Size of Data Set

N=15, 100

- For a given model complexity overfitting problem is less severe as size of data set increases
- Larger the data set, the more complex we can afford to fit the data
- Data should be no less than 5 to 10 times adaptive parameters in model





Techniques to control overfitting

Regularization of Least Squares

- Using relatively complex models with data sets of limited size
- Add a penalty term to error function to discourage coefficients from reaching large values

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

where

$$\|\,\mathrm{w}^2\,\| \equiv w^T w = w_0^{\,2} + w_1^{\,2} + .. + w_M^{\,2}$$

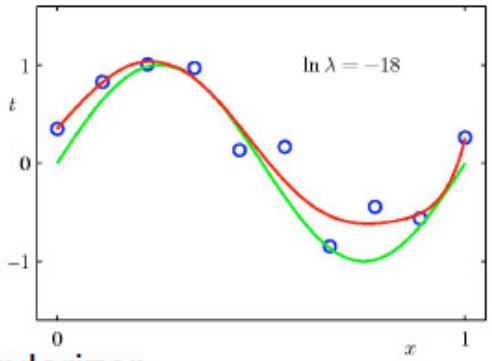
- λ determines relative importance of regularization term to error term
- Can be minimized exactly in closed form
- Known as *shrinkage* in statistics
  Weight decay in neural networks

Techniques for controlling the overfitting

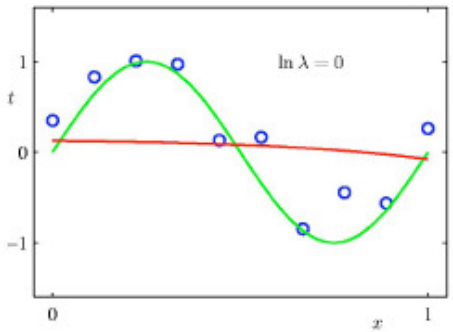
Effect of Regularizer

$M=9$ polynomials using regularized error function

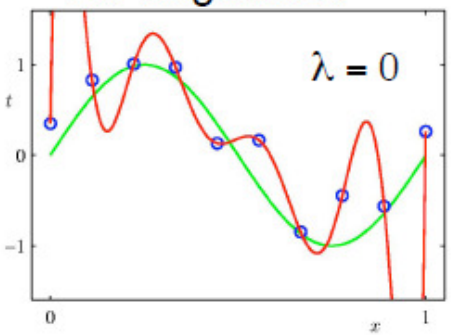
Optimal



Large Regularizer



No Regularizer



No Regularizer
$\lambda = 0$

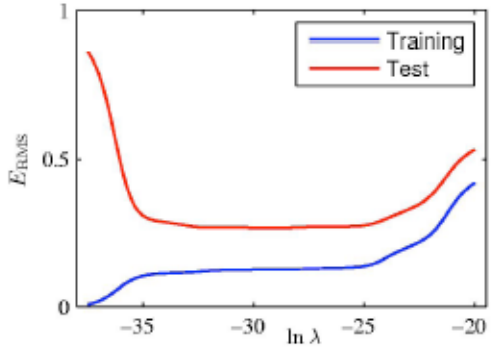
Large Regularizer
$\lambda = 1$

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

Techniques to control overfitting

Impact of Regularization on Error

- λ controls the complexity of the model and hence degree of over-fitting
  - Analogous to choice of M
- Suggested Approach:
- Training set
  - to determine coefficients w
  - For different values of (M or λ)
- Validation set (holdout)
  - to optimize model complexity (M or λ)



M=9 polynomial

- Approach suggests partitioning data into training set to determine coefficients $\mathbf{w}$
- Separate validation set (or hold-out set) to optimize model complexity M or λ
- More sophisticated approaches are not as wasteful of training data
- More principled approach is based on probability theory
- Classification is a special case of regression where target value is discrete values

Math behind regression : learning the weight vector

We are given *N* training samples of '*d*' dimensions with scalar target values

$$(x_1, t_1), (x_2, t_2) \ldots\ldots (x_N, t_N)$$

Define (M+1) non-linear basis functions

$$\varphi(x) = [\varphi_0(x) \quad \varphi_1(x) \quad \varphi_2(x) \quad \ldots\ldots\ldots \varphi_M(x)]$$

For a polynomial fit,

$$\varphi(x) = [1 \quad x \quad \ldots\ldots\ldots x^M]$$

Notations

Denote $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \\ \\ w_M \end{bmatrix}_{(M+1)X1}$ Vector of weights

$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \\ \\ y_N \end{bmatrix}_{NX1}$

Vector of predicted values

$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \\ \\ t_N \end{bmatrix}_{NX1}$

Vector of targets

Predicted value

$$y_1 = \boldsymbol{\varphi}^T(x_1)\mathbf{w}$$

$$y_2 = \boldsymbol{\varphi}^T(x_2)\mathbf{w}$$

Linear combinations of these basis functions gives the predicted value for each of the samples.

$$y_N = \boldsymbol{\varphi}^T(x_N)\mathbf{w}$$

The idea is to learn the (M+1) weights.

Error function

$$\begin{bmatrix} ---\varphi^T(x_1)---- \\ ---\varphi^T(x_2)---- \\ \\ \\ ---\varphi^T(x_N)--- \end{bmatrix}_{NXM} \begin{bmatrix} w_1 \\ w_2 \\ \\ \\ w_M \end{bmatrix}_{MX1} = \begin{bmatrix} y_1 \\ y_2 \\ \\ \\ y_N \end{bmatrix}_{NX1}$$

Let $$\Phi = \begin{bmatrix} ---\varphi^T(x_1)---- \\ ---\varphi^T(x_2)---- \\ \\ \\ ---\varphi^T(x_N)--- \end{bmatrix}_{NX(M+1)}$$

We can write $$\mathbf{y} = \Phi\mathbf{w}$$

Error function $$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2$$

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

Error function : quadratic

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2$$

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

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

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

Using $\mathbf{w}^T\Phi^T\mathbf{t} = \mathbf{t}^T\Phi\mathbf{w}$

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

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2$$

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

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

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

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \Phi^T \mathbf{t} - \Phi^T \Phi \mathbf{w}$$

At the optimal weight vector , we have

$$\frac{\partial E(\mathbf{w}^*)}{\partial \mathbf{w}} = 0$$

$$\Rightarrow \Phi^T \mathbf{t} - \Phi^T \Phi \mathbf{w}^* = 0$$

$$\mathbf{w}^* = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

- With regularization

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(t_i - \mathbf{w}^T\phi(\mathbf{x}_i)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- It can shown :

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

Extension to high dimension input feature

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots\ldots\ldots + w_M x_M$$

Let $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \\ \\ w_M \end{bmatrix}_{(M+1)X1}$ and $\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \\ \\ x_M \end{bmatrix}_{(M+1)X1}$

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$$

$\mathbf{x}$: M dimensional training featur

Predicted value

$$y_1 = \phi^T(\mathbf{x}_1)\mathbf{w}$$

$$y_2 = \phi^T(\mathbf{x}_2)\mathbf{w}$$

$$y_N = \phi^T(\mathbf{x}_N)\mathbf{w}$$

Linear combinations of basis functions $\phi(\mathbf{x})$ gives the predicted value for each of the samples.

The idea is to learn the M weights.

- We can derive a similar expression as

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

where

$$\Phi = \begin{bmatrix} ---\phi^T(\mathbf{x}_1)---- \\ ---\phi^T(\mathbf{x}_2)---- \\ \\ \\ ---\phi^T(\mathbf{x}_N)--- \end{bmatrix}_{NX(M+1)}$$

- Later we will look at regression problem and derive the weights (a) from the probabilistic framework and (b) from optimization techniques (gradient descent).

- We ll now switch gears to explore more on the generative model of classifiers --- parametric and non-parametric approaches.

Reading

- Sections 1.1, 1.3, 1.4, 3.1

Christopher Bishop – “Pattern Recognition and Machine Learning”