CS722/822: Machine Learning

Instructor: Jiangwen Sun
Computer Science Department

Algorithm of gradient descent

1. Set iteration $k = 0$, make an initial guess $\mathbf{w}_0$
2. repeat:
3. Compute the negative gradient of $E(\mathbf{w})$ at $\mathbf{w}_k$ and set it to be the search direction $\mathbf{d}_k$
4. Choose a step size α_k to sufficiently reduce
$$E(\mathbf{w}_k + \alpha_k \mathbf{d}_k)$$
5. Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{d}_k$
6. $k = k + 1$
7. Until a termination rule is met

Methods to choose step size α_k

1. Can use a small constant, e.g., $\alpha_k = 10^{-5}$
2. Can use a decay scheme, e.g., $\alpha_0 = 1, \alpha_{k+1} = \alpha_k/2$
3. Can perform an exact line search, i.e., finding the value of α_k that minimizes

$$E(\mathbf{w}_k + \alpha_k \mathbf{d}_k)$$

4. Can perform an inexact line search
   - Backtracking line search

     Given $\gamma \in (0,0.5)$, $\beta \in (0,1)$

     $\alpha_k = 1$

     While $E(\mathbf{w}_k + \alpha_k \mathbf{d}_k) > E(\mathbf{w}_k) - \gamma \alpha_k d_k^T \mathbf{d}_k$, $\alpha_k = \beta \alpha_k$
   - Barzilai-Borwein method

$$\alpha_k = \frac{(\mathbf{w}_k - \mathbf{w}_{k-1})^T \left(\nabla E(\mathbf{w}_k) - \nabla E(\mathbf{w}_{k-1})\right)}{\|\nabla E(\mathbf{w}_k) - \nabla E(\mathbf{w}_{k-1})\|^2}$$

Reference for BB method

J. Barzilai and J.M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

Possible termination rules

1. If the function value of $E(\mathbf{w})$ does not change any more with the iterations

$$|E(\mathbf{w}_{k+1}) - E(\mathbf{w}_k)| < \varepsilon$$

2. If the magnitude of the gradient is small enough

$$\|\nabla E(\mathbf{w}_{k+1})\| < \varepsilon$$

3. If the difference between $\mathbf{w}$'s from the two consecutive iterates is small enough

$$\|\mathbf{w}_{k+1} - \mathbf{w}_k\| < \varepsilon$$

4. Fixed number of iterations: $nMaxIter$

Summary

- How to solve least squares to obtain a model f
  - If the hypothesis space contains a set of functions that are linear in terms of weight parameters $\mathbf{w}$, there is a closed-form solution
  - This closed-form solution might be time-consuming depending on the size of the data matrix. In this case, gradient descent can be used
  - If the hypothesis space contains a set of functions that are nonlinear in terms of weight $\mathbf{w}$, there may not be an analytic solution, use the gradient descent
  - Other algorithms, such as Newton-Raphson method, might also be used depending on the property of $\boldsymbol{E}$

Reference

S. Boyd. Convex Optimization. *Cambridge University Press*, 2004.

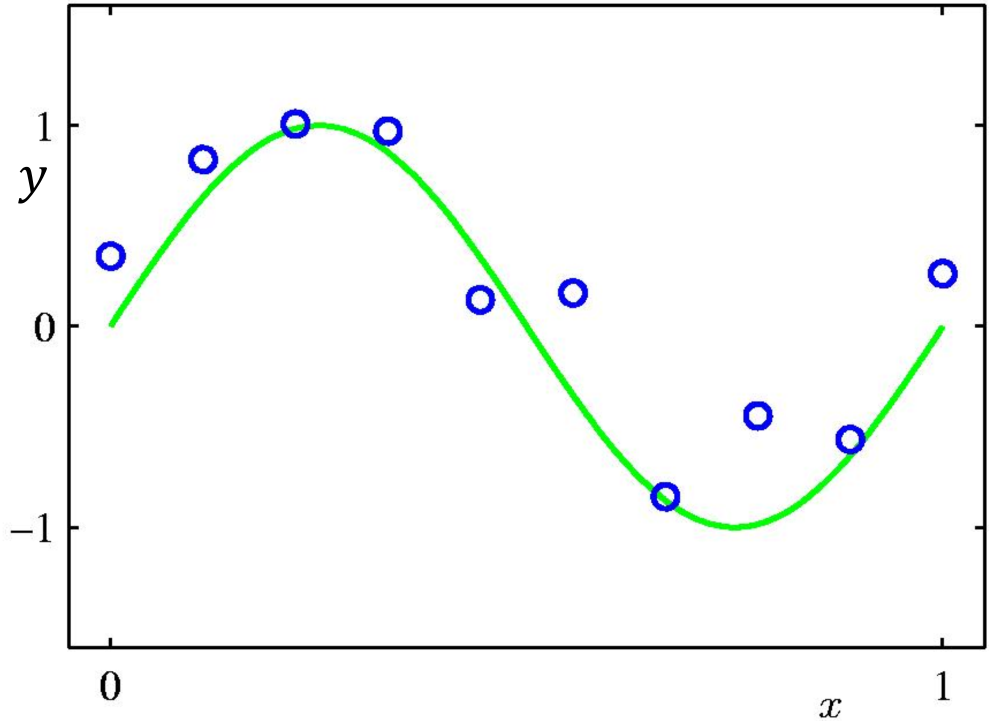
Overfitting versus Underfitting

- Hypothesis space complexity vs data (or sample) complexity
  - Hypothesis space complexity: # of functions in the space
  - Data complexity: # of examples

- Overfitting: happens when the hypothesis space is more complex than the data complexity

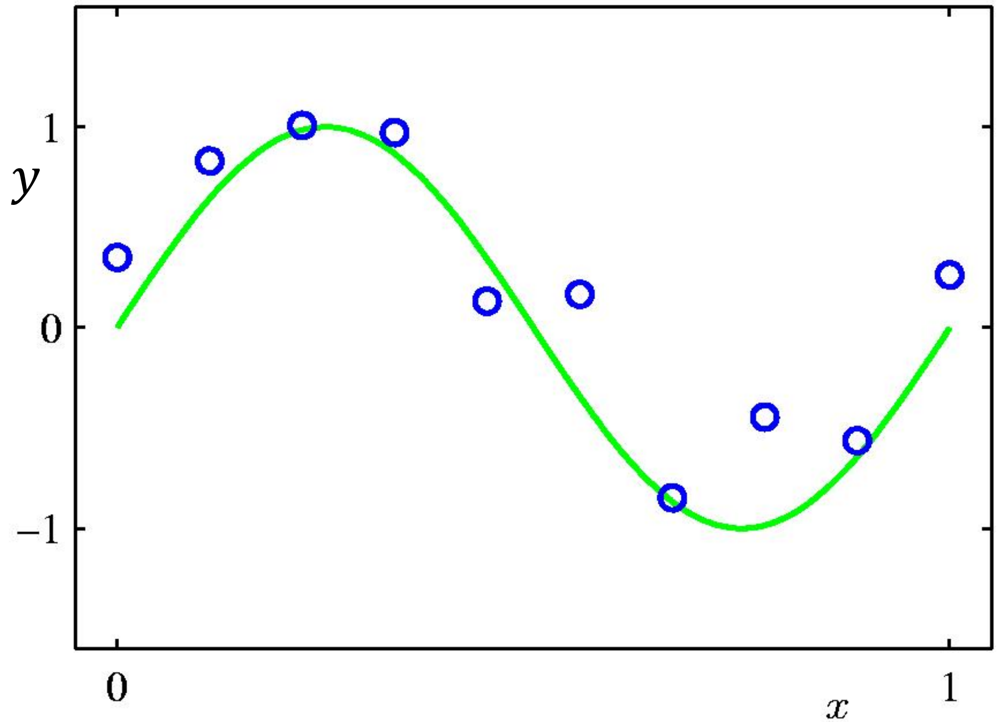
- Underfitting: happens when the hypothesis space is too simple to cover the data complexity

Polynomial Curve Fitting



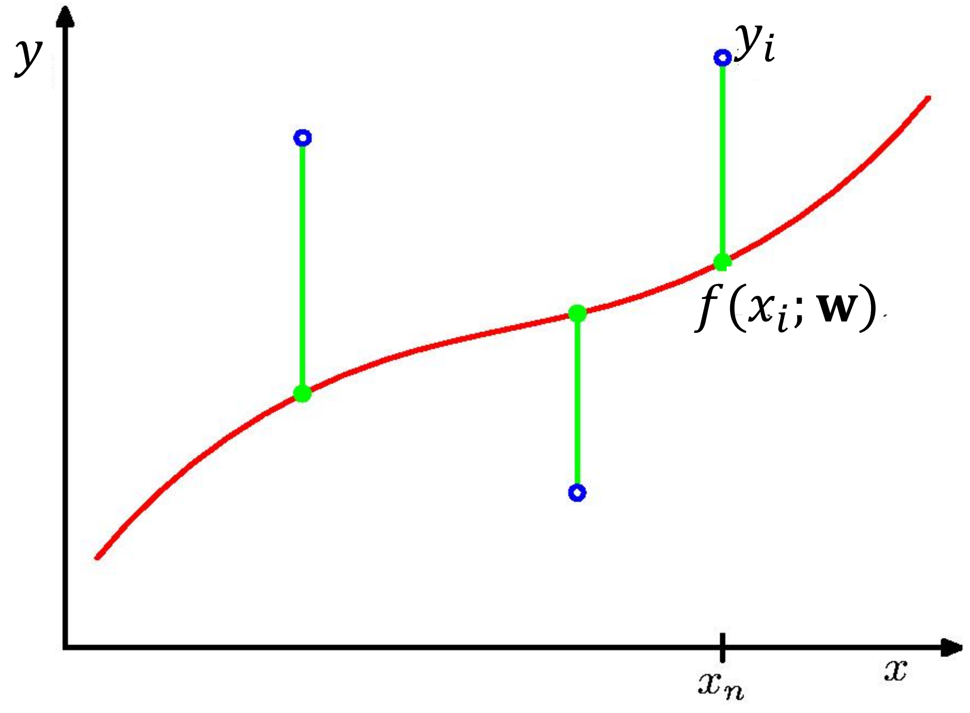
- $f(x) = \sin(2\pi\, x)$
- x is evenly distributed from [0,1]
- $y = f(x)$ + random error
  - e.g., $y = \sin(2\pi\, x) + \varepsilon, \varepsilon \sim N(0, \sigma^2)$

Polynomial Curve Fitting



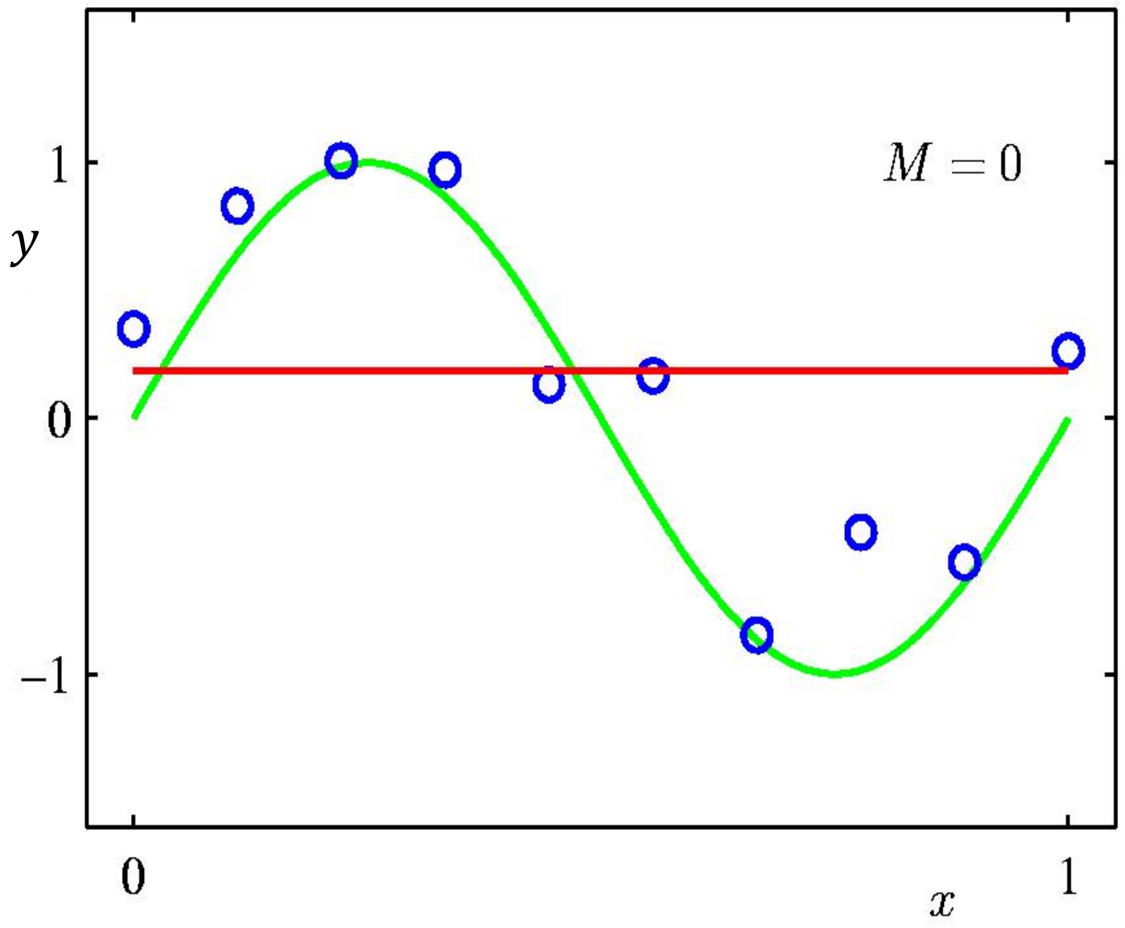
$$f(x; \mathbf{w}) = \sum_{j=0}^{M} w_j x^j$$

Sum-of-Squares Error Function

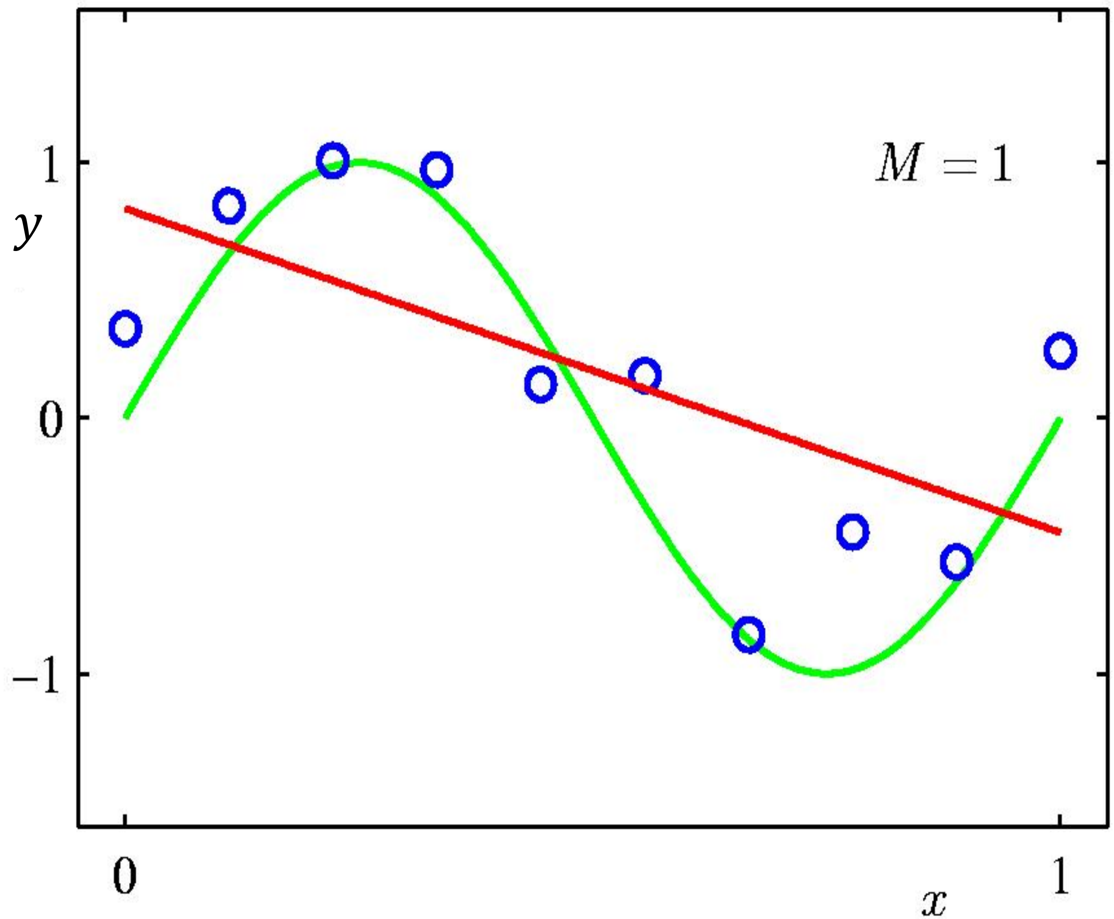


$$E(\mathbf{w}) = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2$$

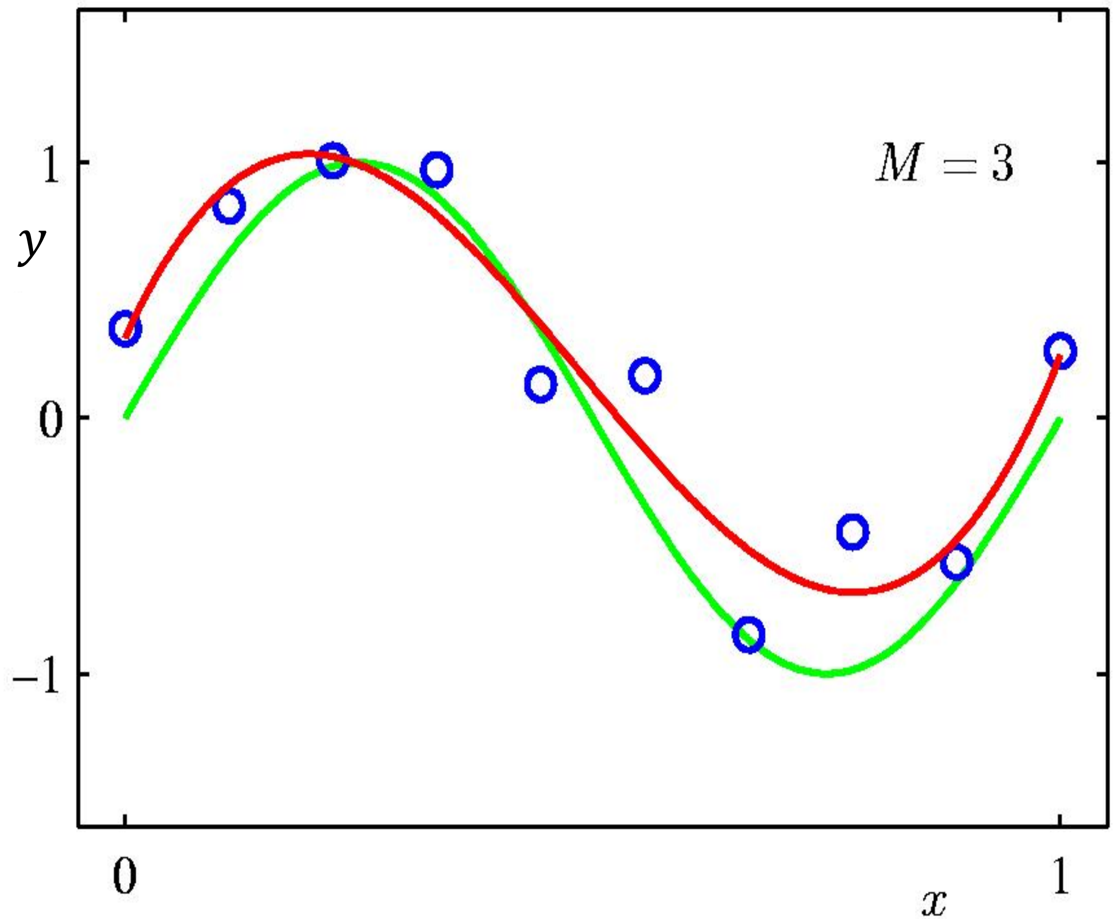
0^{th} Order Polynomial



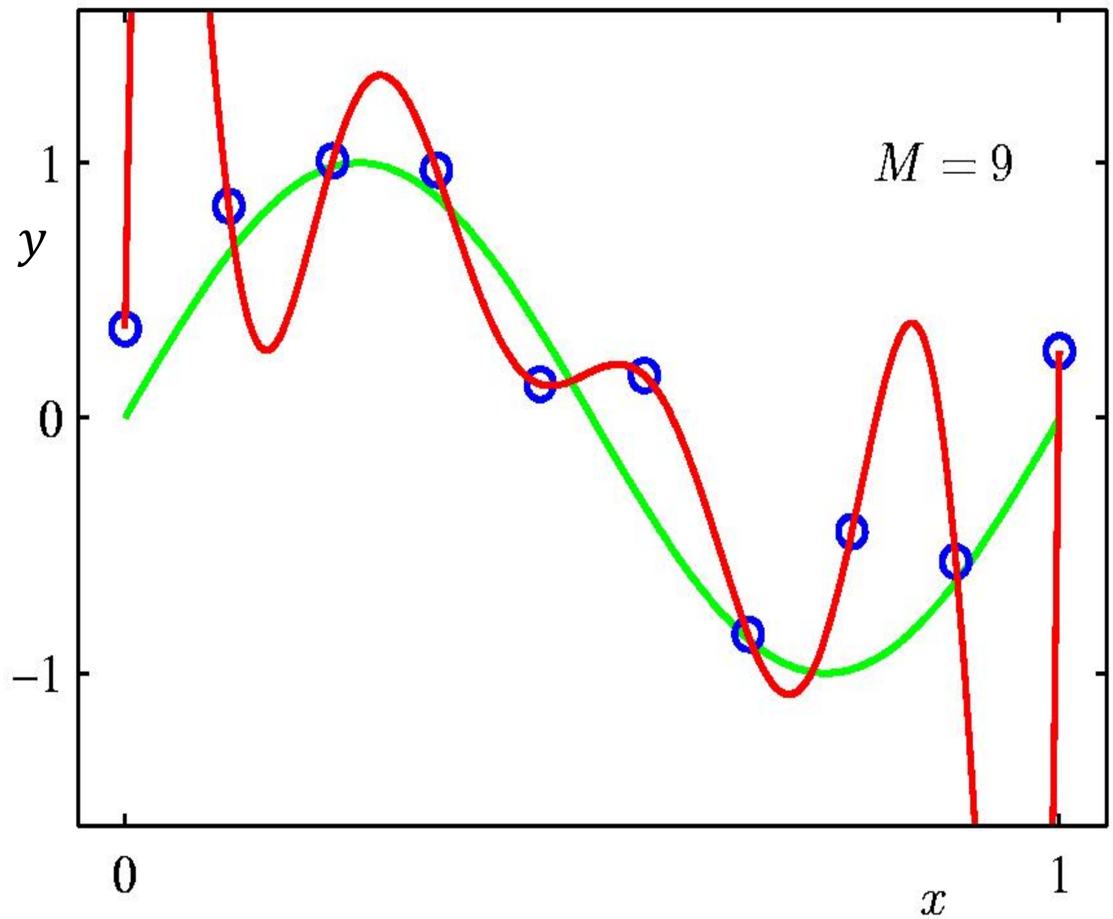
1st Order Polynomial



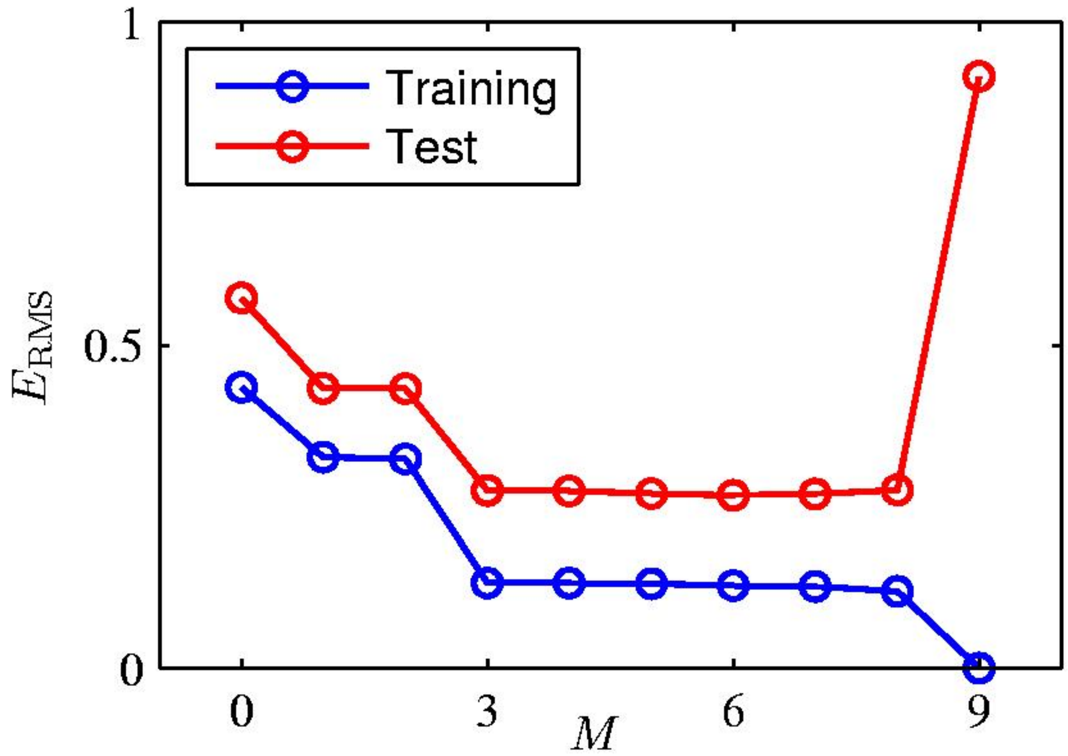
3^{rd} Order Polynomial



9^{th} Order Polynomial



Over-fitting



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

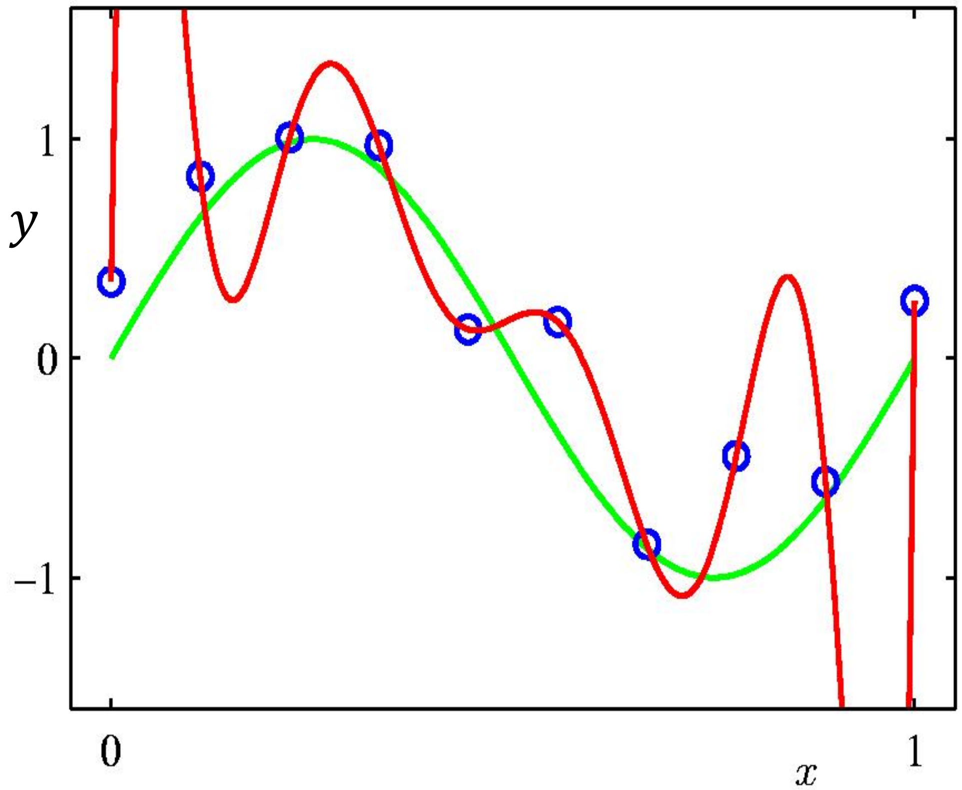
Polynomial Coefficients

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

Data Set Size

9^{th} Order Polynomial

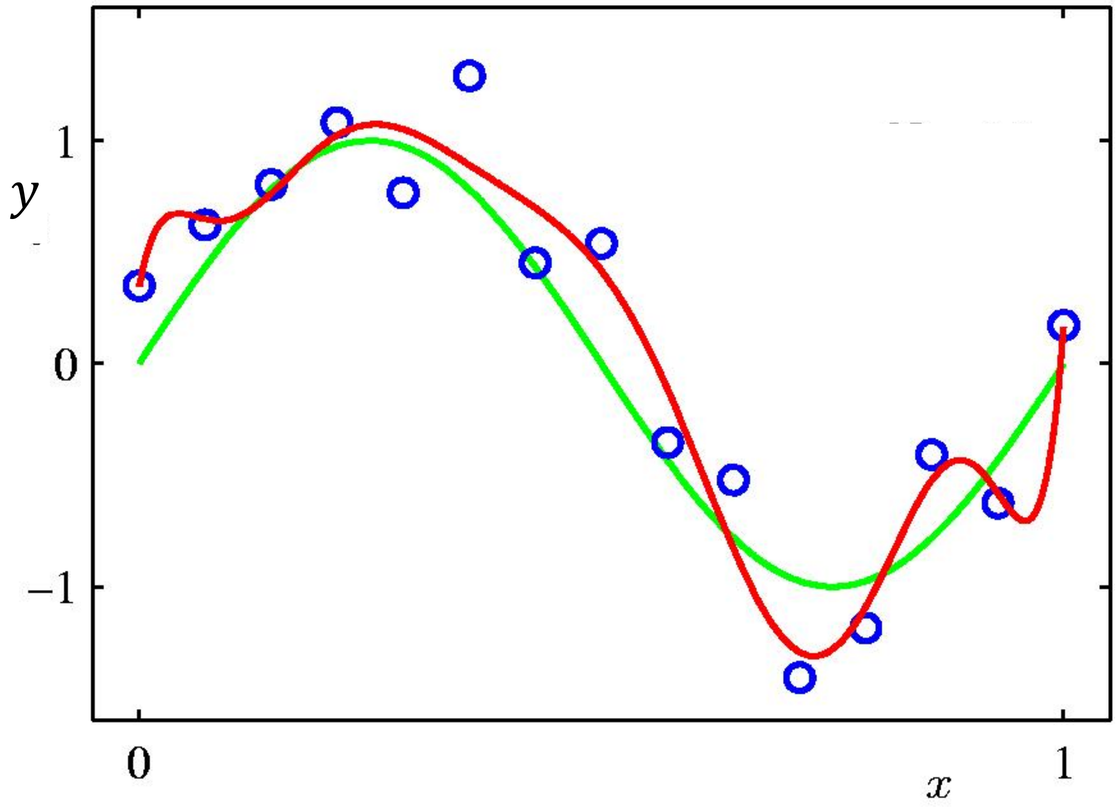
$$N = 10$$



Data Set Size

9th Order Polynomial

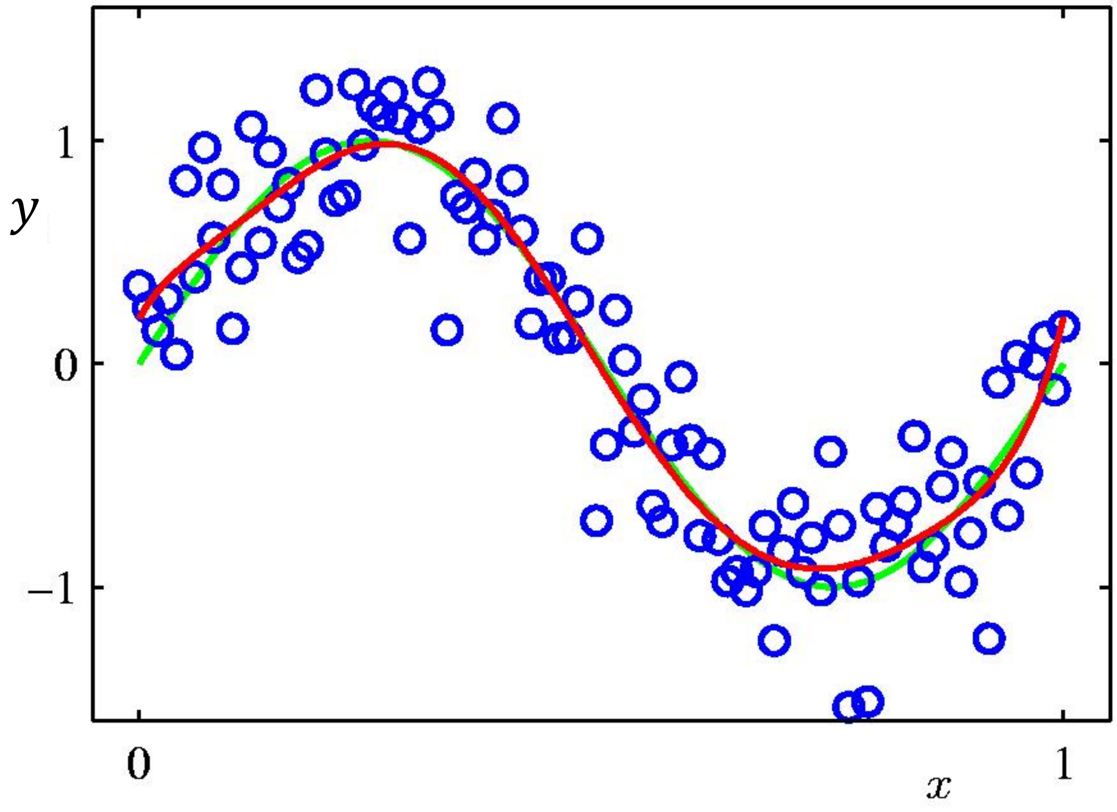
$$N = 15$$



Data Set Size:

9th Order Polynomial

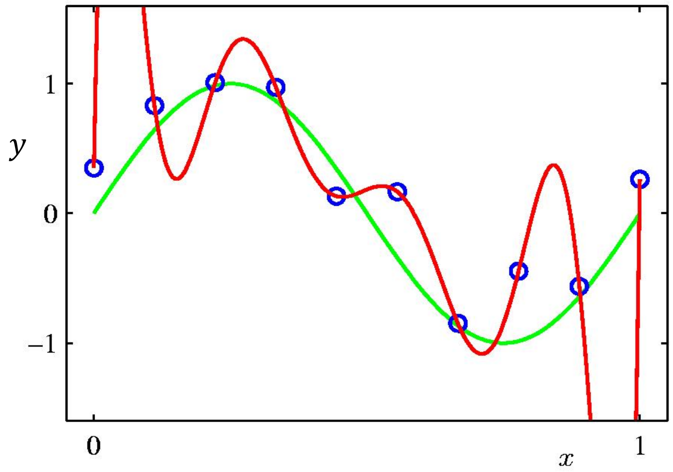
$$N = 100$$



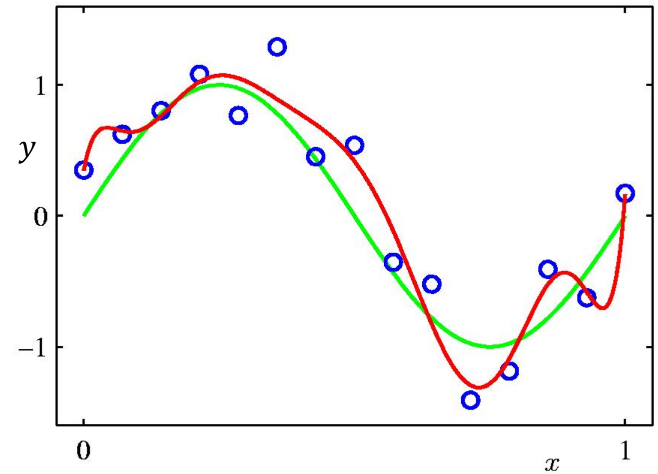
Data Set Size:

9th Order Polynomial

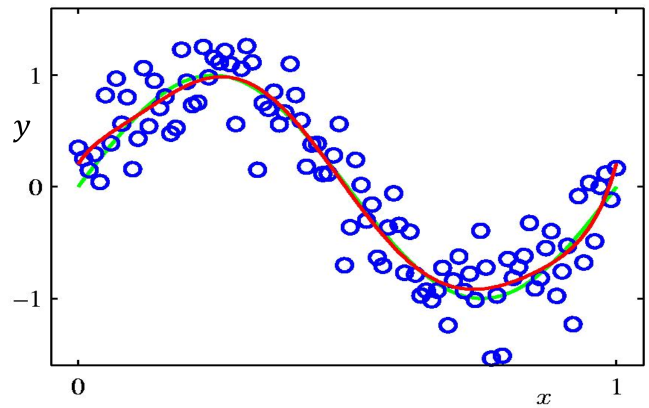
$N = 10$



$N = 15$



$N = 100$



Regularization

- Penalize large coefficient values

- Ridge regression – using two norm regularizer

$$E(\mathbf{w}) = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2 + \lambda \|\mathbf{w}\|^2$$

- LASSO – using one norm regularizer

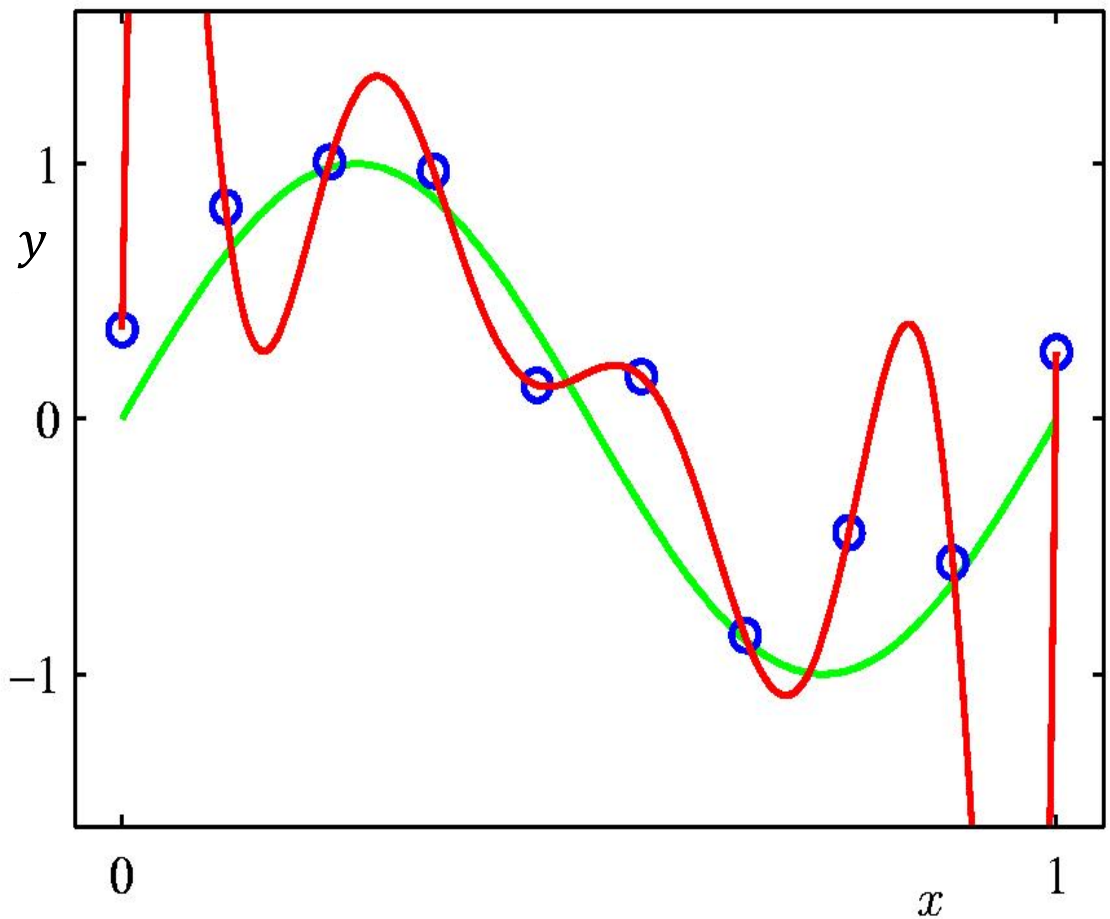
$$E(\mathbf{w}) = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2 + \lambda \|\mathbf{w}\|_1$$

- Other choices of regularizer

Regularization: $\ln \lambda = -\infty$

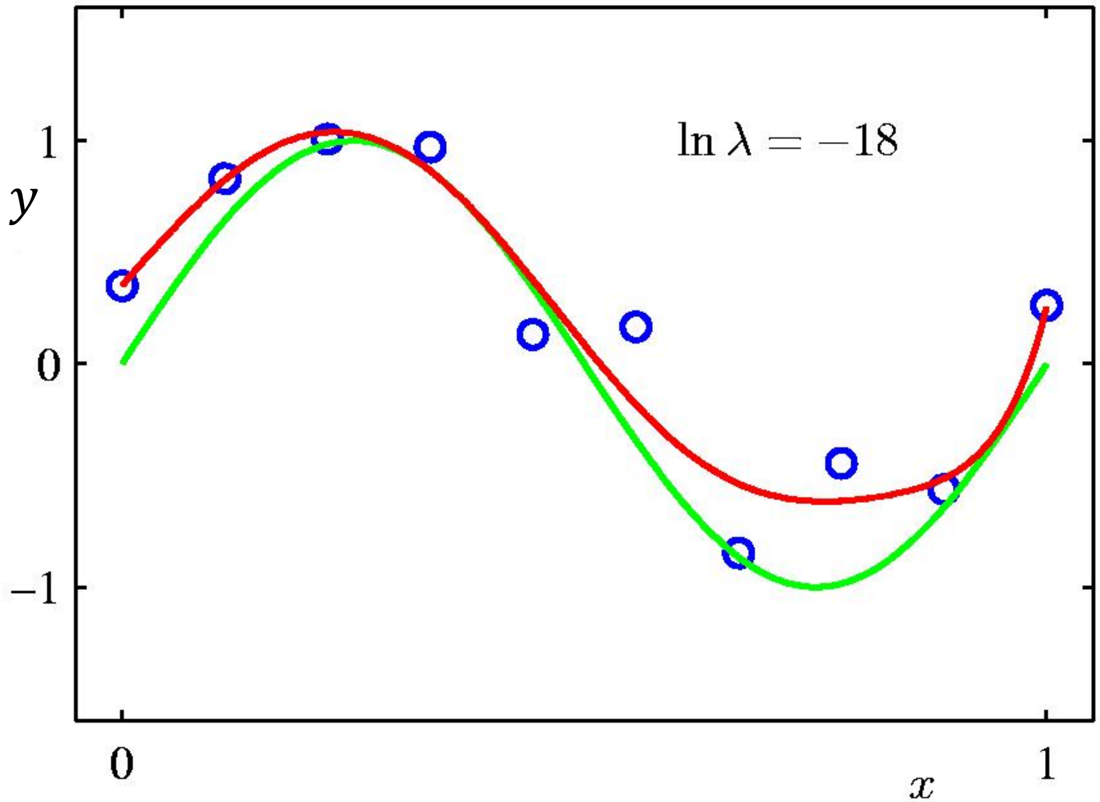
9th Order Polynomial

$N = 10$



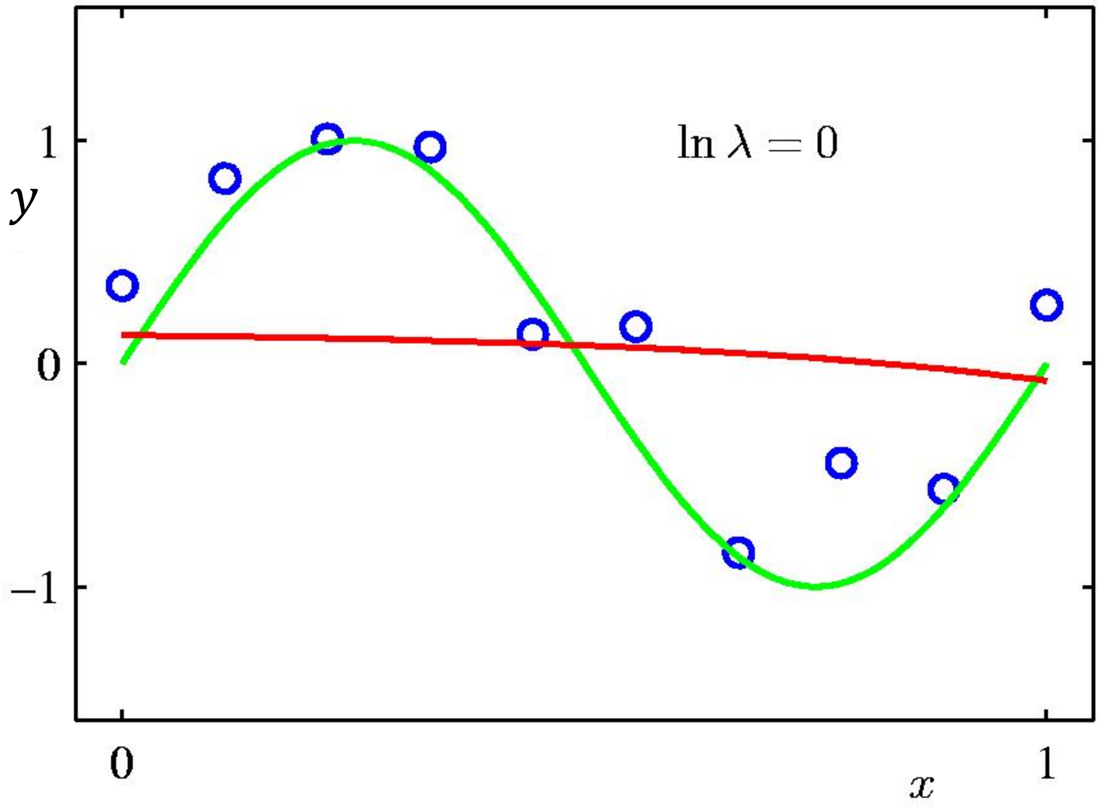
Regularization: $\ln \lambda = -18$

9th Order Polynomial

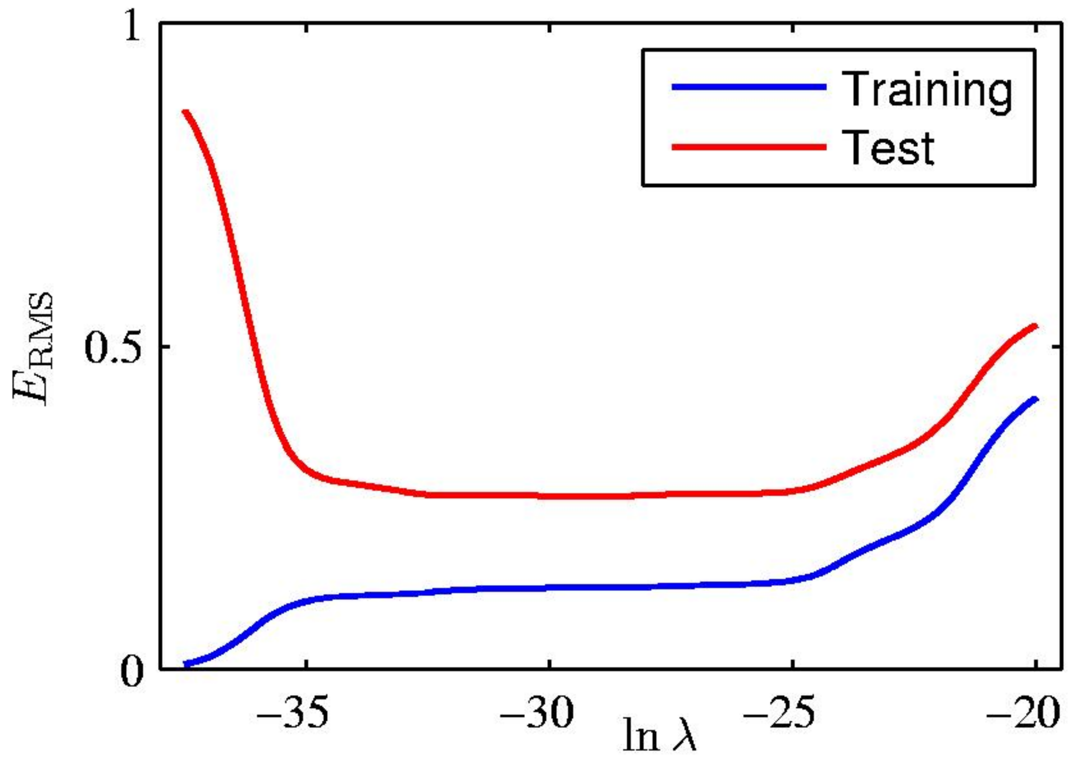


Regularization: $\ln \lambda = 0$

9th Order Polynomial



Regularization: E_{RMS} VS $\ln \lambda$



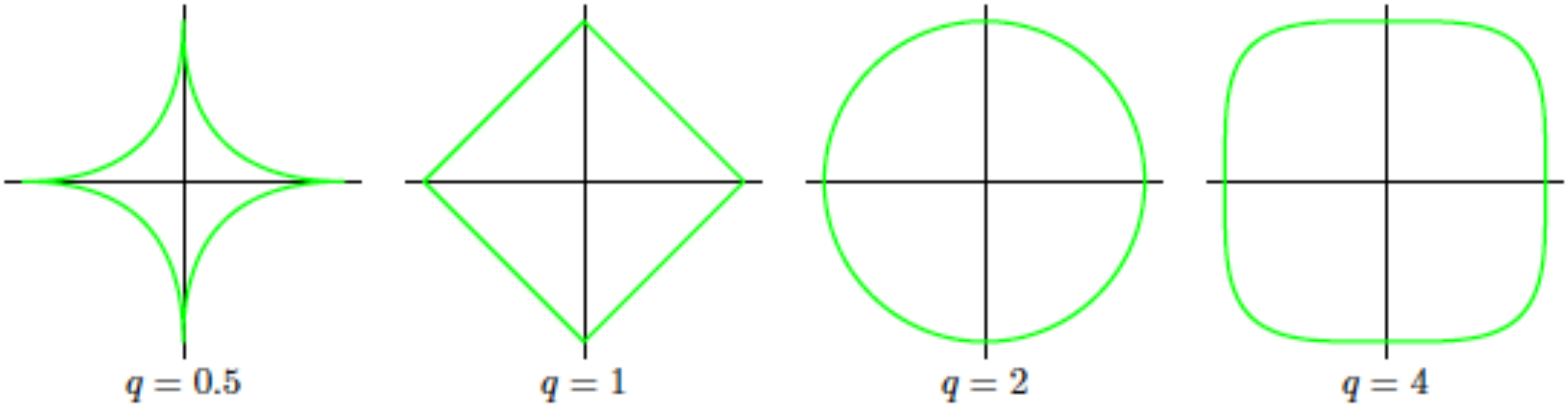
Polynomial Coefficients

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

Choices of Regularizer

- There are different choices of regularization

$$\sum_{i=1}^{N}(y_i - f(\mathbf{x}_{\boldsymbol{i}};\mathbf{w}))^2 + \lambda \boxed{\sum_{j=1} |w_j|^q}$$

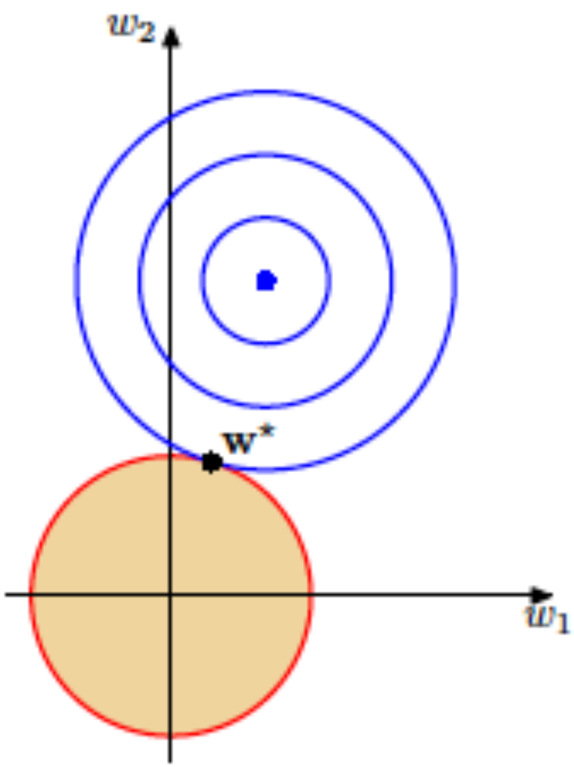


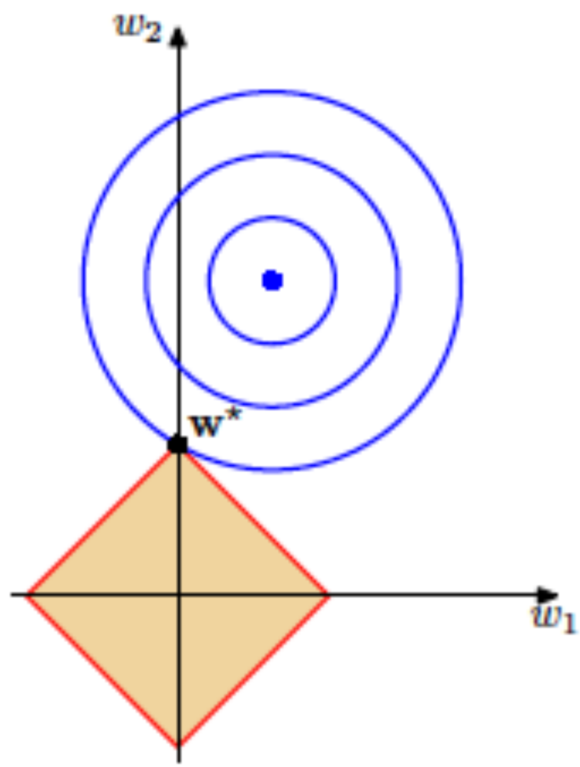
Choices of Regularizer

- Comparing $\ell 1$-norm versus $\ell 2$-norm regularizer

$$E(\mathbf{w}) = \sum_{i=1}^{N} (y_i - f(\mathbf{x_i}; \mathbf{w}))^2 + \boxed{\lambda \|w\|^2}$$

$$E(\mathbf{w}) = \sum_{i=1}^{N} (y_i - f(\mathbf{x_i}; \mathbf{w}))^2 + \boxed{\lambda \|w\|_1}$$





Why $\ell 1$-norm regularizer leads to more sparse $\mathbf{w}$?