ISTA 421/521
Introduction to Machine Learning

Lecture 4:
Nonlinear response, CV, Regularization

Clay Morrison
clayton@sista.arizona.edu
Gould-Simpson 819
Phone 621-6609

4 September 2014

Next Topics

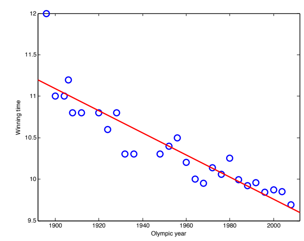
- Moving to higher dimensions
  - Linear Algebra: matrix operators
  - Some Geometry of Linear Algebra
  - Least Mean Squares in Matrix formulation
  - The Geometry of LMS solution
- Nonlinear Response
- Model Selection
  - Generalization and Overfitting
  - Method 1: Cross Validation
- Regularized Least Squares

The Normal Equations

For model: $t = f(x_1, ..., x_k; w_0, ..., w_k) = \sum_{i=0}^{k} x_i w_i$

$$w_0 = \bar{t} - w_1 \bar{x}$$

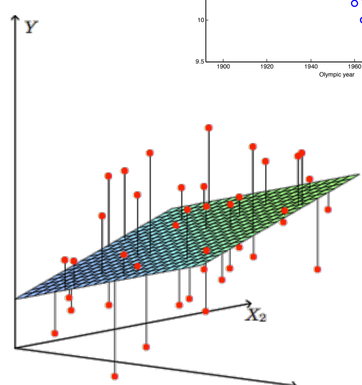
$$w_1 = \frac{\overline{xt} - \overline{x}\overline{t}}{\overline{x^2} - (\overline{x})^2}$$



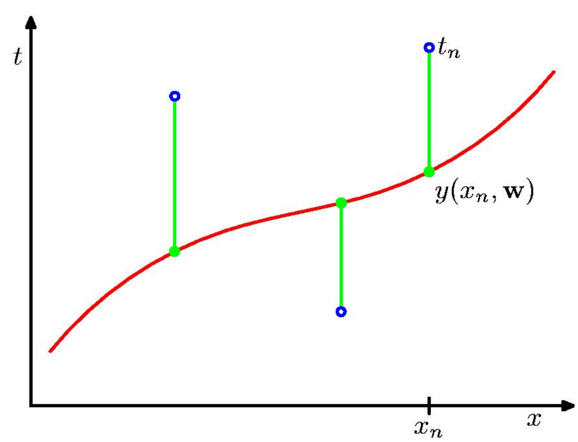
$$\hat{\mathbf{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{t}$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \quad \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$



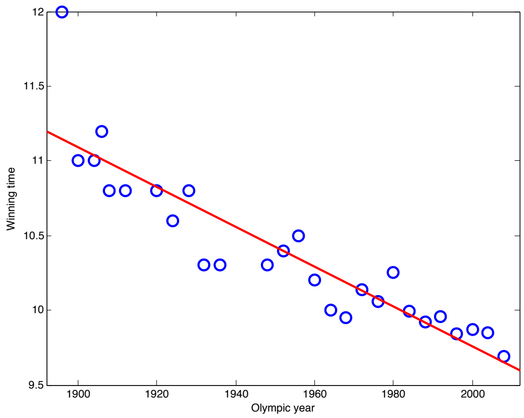
Sum-of-Squares Loss (Error) Function

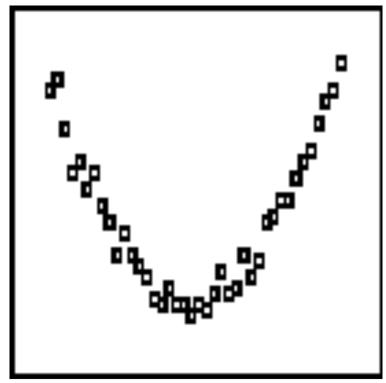


$$\mathcal{L} = \frac{1}{N}(\mathbf{t} - \mathbf{X}\mathbf{w})^\top(\mathbf{t} - \mathbf{X}\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(t_n - \mathbf{w}^\top \mathbf{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(t_n - (w_0 + w_1 x_n))^2$$

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$ Another formulation, from Bishop (2006)

Linear (in variables) has its limit!



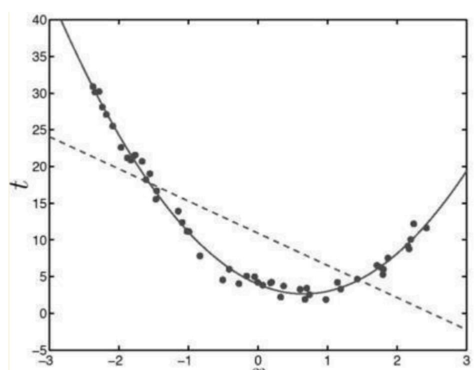


Nonlinear Response

- We can extend the power of linear LMS best fit to models that have a non-linear response.

$$f(x; \mathbf{w}) = \mathbf{w}^{\mathsf{T}}\mathbf{x} = w_0 + w_1 x + w_2 x^2$$

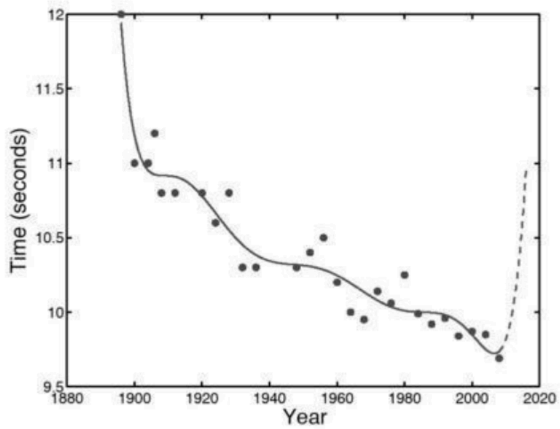
$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ x_n^2 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$



Fitting the parameters **w** still works the same! The only difference is that we square the *x* values at the input phase (for each of the elements of the third column vector)

Generalize to Models of kth-order Polynomials

$$f(x; \mathbf{w}) = \sum_{k=0}^{K} w_k x^k \qquad \mathbf{X} = \begin{bmatrix} x_1^0 & x_1^1 & x_1^2 & \cdots & x_1^K \\ x_2^0 & x_2^1 & x_2^2 & \cdots & x_2^K \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x_N^0 & x_N^1 & x_N^2 & \cdots & x_N^K \end{bmatrix}$$



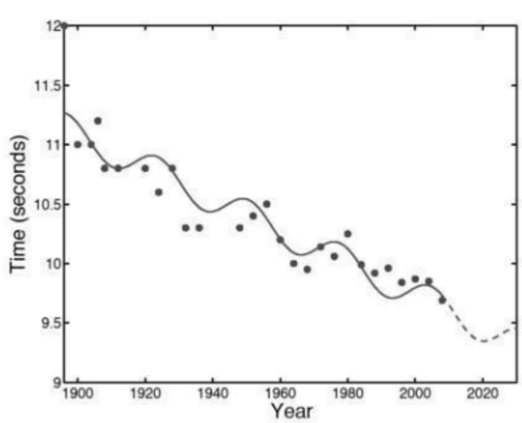
Note: this is **not** creating more independent sources of information about individuals, but it ***IS*** giving the model the capacity to consider non-linear *components* of what original inputs there are.

And we're still just learning **LINEAR COMBINATIONS** of those components

Linear Combination of Functions (not just polynomials)

$$\mathbf{X} = \begin{bmatrix} h_1(x_1) & h_2(x_1) & \cdots & h_K(x_1) \\ h_1(x_2) & h_2(x_2) & \cdots & h_K(x_2) \\ \vdots & \vdots & \cdots & \vdots \\ h_1(x_N) & h_2(x_N) & \cdots & h_K(x_N) \end{bmatrix}$$

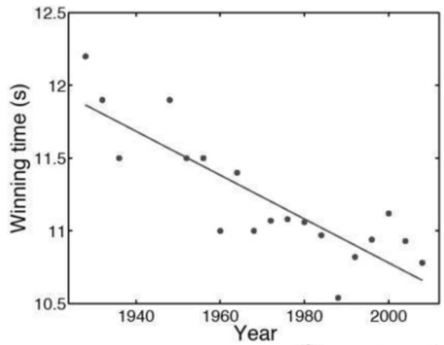
$$h_1(x) = 1$$
$$h_2(x) = x$$
$$h_3(x) = \sin\left(\frac{x-a}{b}\right)$$
$$f(x; \mathbf{w}) = w_0 + w_1 x + w_2 \sin\left(\frac{x-a}{b}\right).$$

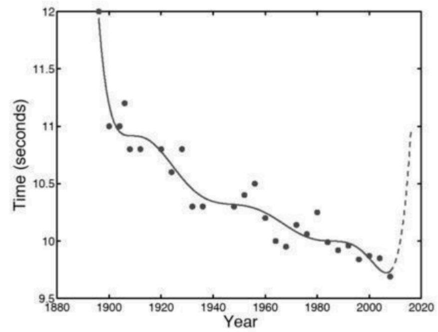


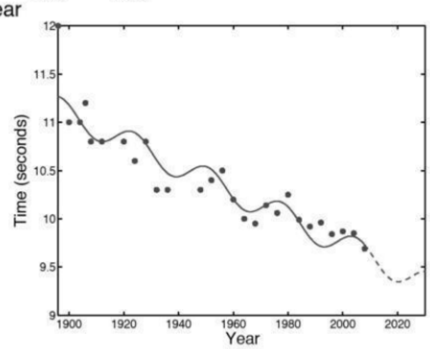
Careful !!
a and *b* must be **constants**

All variables must be ***linearly*** combined

Which Model is better: 1^{st} order, 8^{th} order ?

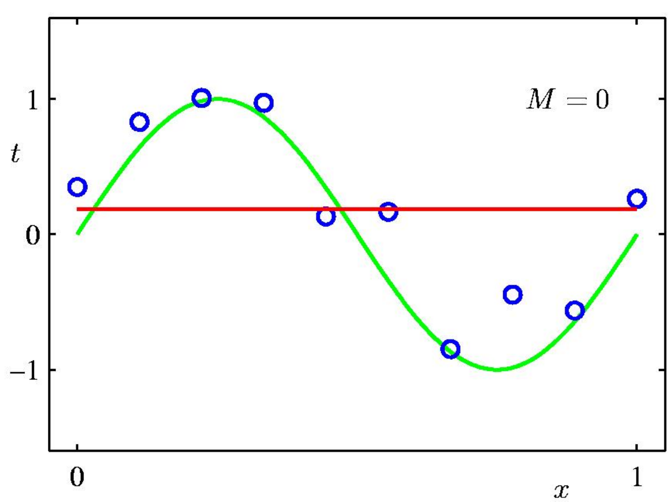




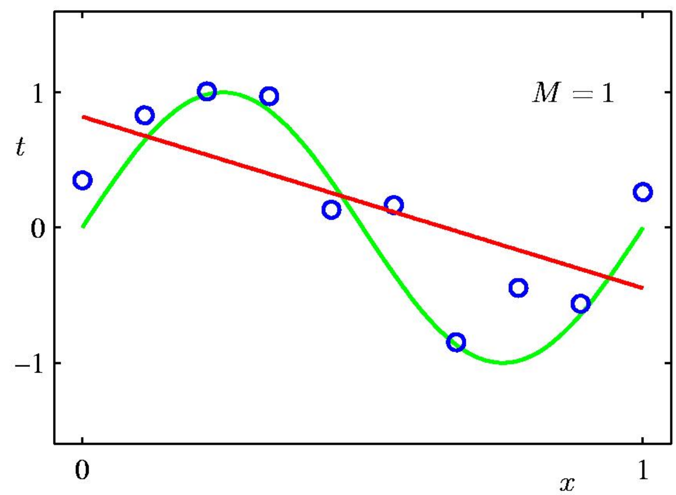


… periodic?

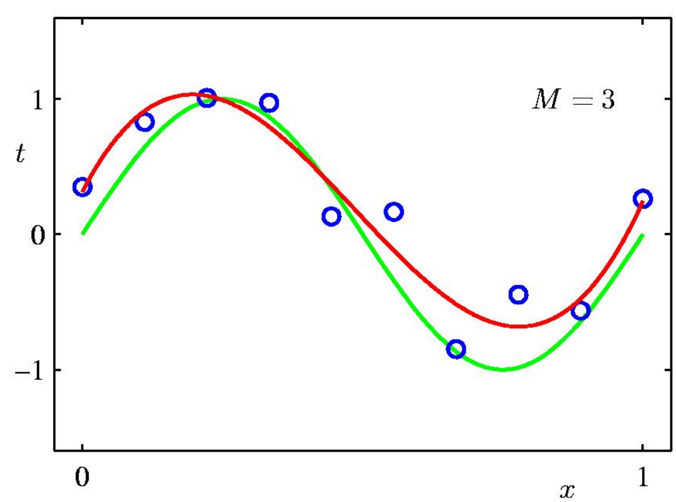
0^{th} Order Polynomial



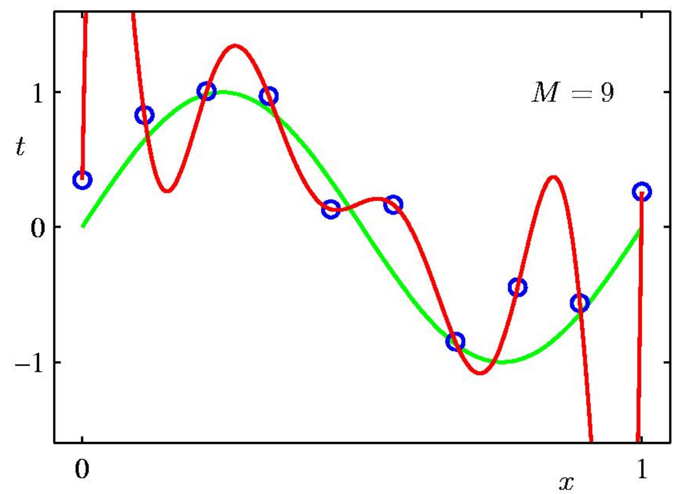
1^{st} Order Polynomial



3^{rd} Order Polynomial



9^{th} Order Polynomial

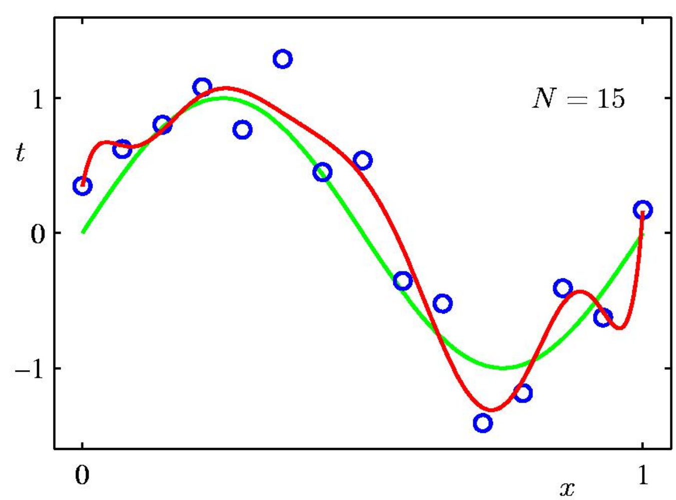


Overfitting

Data Set Size:

$N = 15$

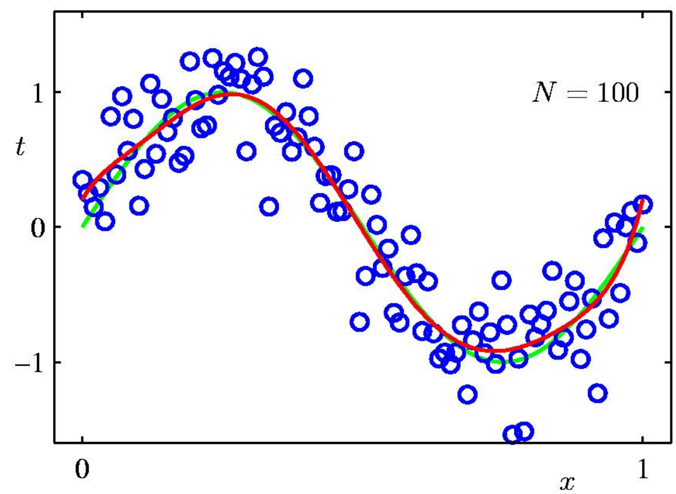
9^{th} Order Polynomial



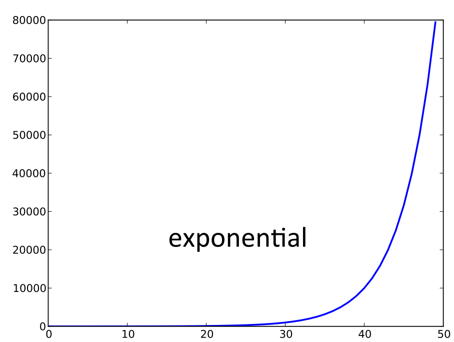
Data Set Size:

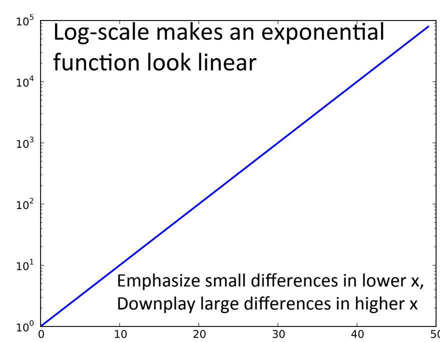
$$N = 100$$

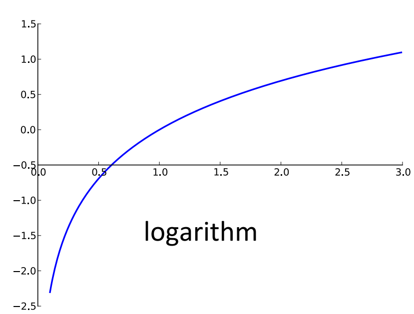
9th Order Polynomial



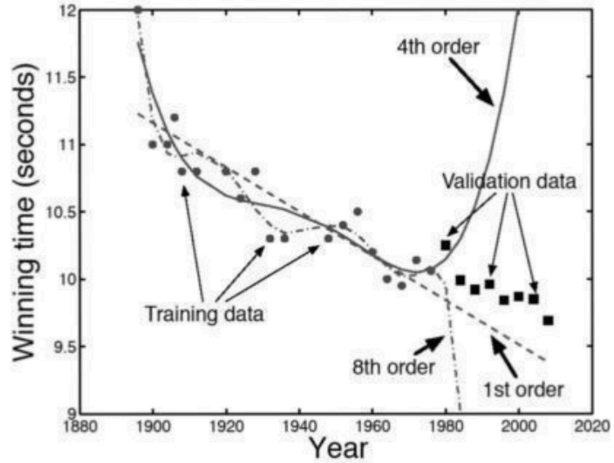
Sidenote: Log scale

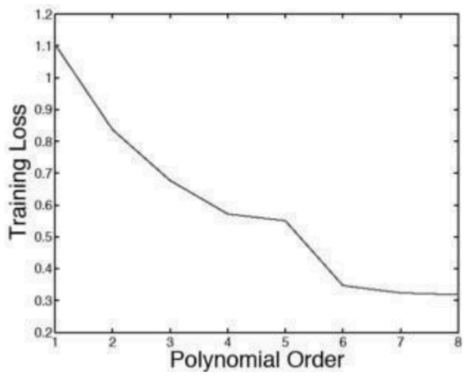


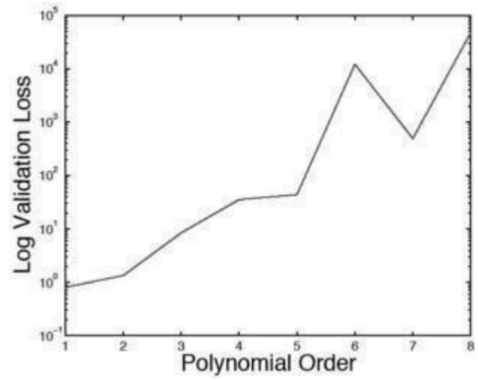




Training versus Testing





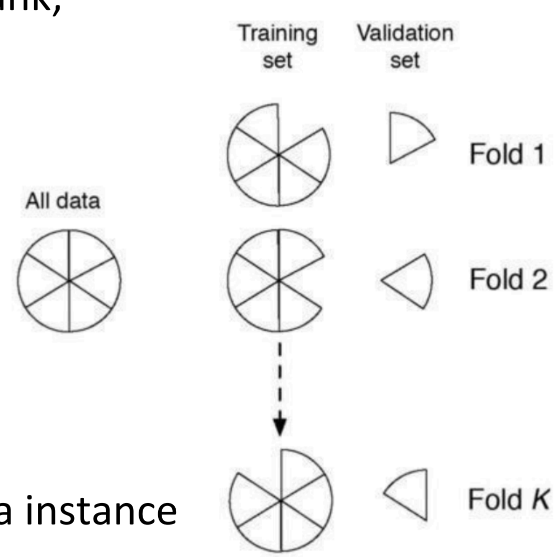


Cross-Validation

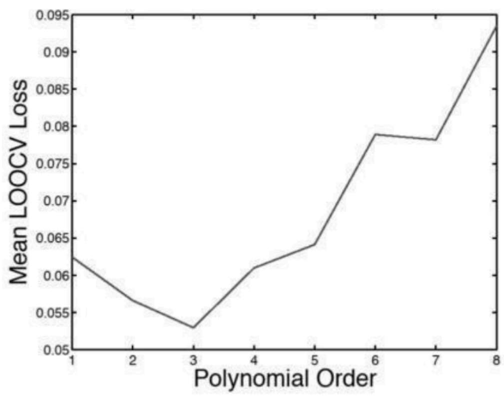
Randomly split your data into a set of *k* chunks
"hold out" a chunk of the data set;
train on everything but that chunk;
test with the chunk
Repeat this for all chunks

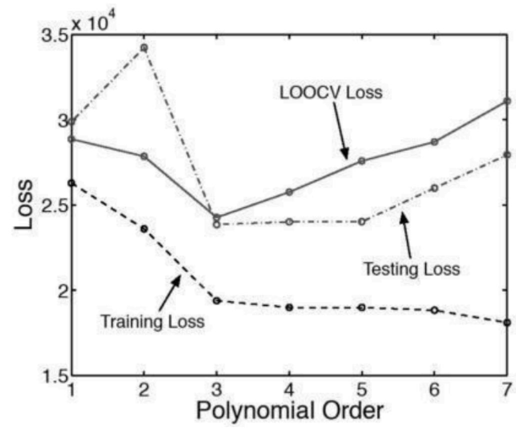
What this does:
Estimates the error
Of a number of possible
Models trained on data subsets

Leave-one-out-CV (LOOCV)
… same thing, but chunk = 1 data instance



LOOCV for Model Selection

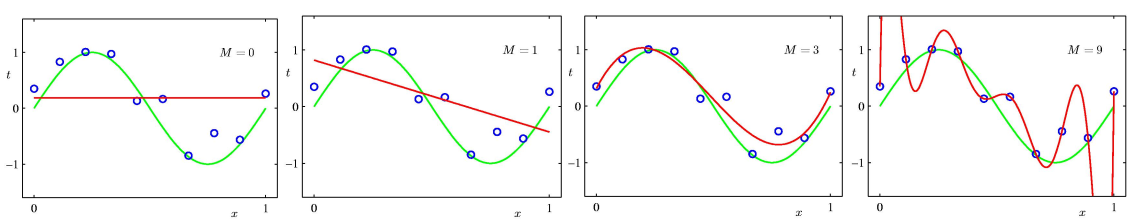




On Men's 100 meter data
Trying different orders of polynomials
for the models

Study with artificial data (3rd order poly)
Sample size: 50
Test error based on 1000 indep samples

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

Regularization

- Penalize large coefficient values: add magnitude of all of the weights (e.g., their sum) as part of the loss.

$$\sum_i w_i^2 = \mathbf{w}^\top \mathbf{w} \qquad \mathcal{L}' = \mathcal{L} + \lambda \mathbf{w}^\top \mathbf{w}$$

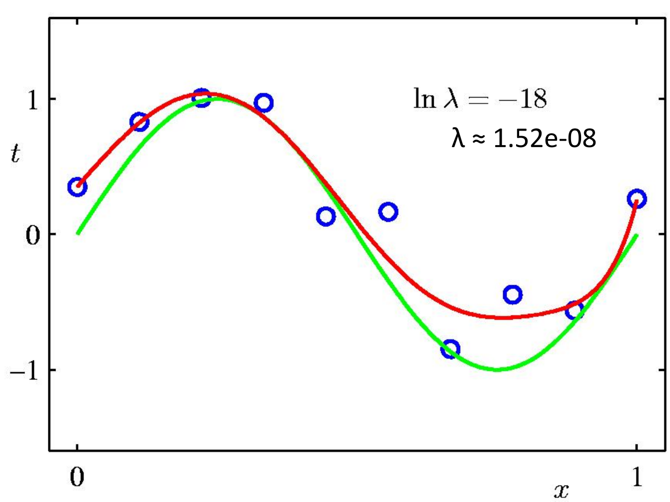
$$\begin{aligned}\mathcal{L}' &= \mathcal{L} + \lambda \mathbf{w}^\top \mathbf{w} \\ &= \frac{1}{N}\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - \frac{2}{N}\mathbf{w}^\top \mathbf{X}^\top \mathbf{t} + \lambda \mathbf{w}^\top \mathbf{w}\end{aligned}$$

$$\frac{\partial \mathcal{L}'}{\partial \mathbf{w}} = \frac{2}{N}\mathbf{X}^\top \mathbf{X}\mathbf{w} - \frac{2}{N}\mathbf{X}^\top \mathbf{t} + 2\lambda \mathbf{w}$$

$$\begin{aligned}\frac{2}{N}\mathbf{X}^\top \mathbf{X}\mathbf{w} - \frac{2}{N}\mathbf{X}^\top \mathbf{t} + 2\lambda \mathbf{w} &= 0 \\ \left(\mathbf{X}^\top \mathbf{X} + N\lambda \mathbf{I}\right)\mathbf{w} &= \mathbf{X}^\top \mathbf{t} \\ \hat{\mathbf{w}} &= \left(\mathbf{X}^\top \mathbf{X} + N\lambda \mathbf{I}\right)^{-1}\mathbf{X}^\top \mathbf{t}\end{aligned}$$

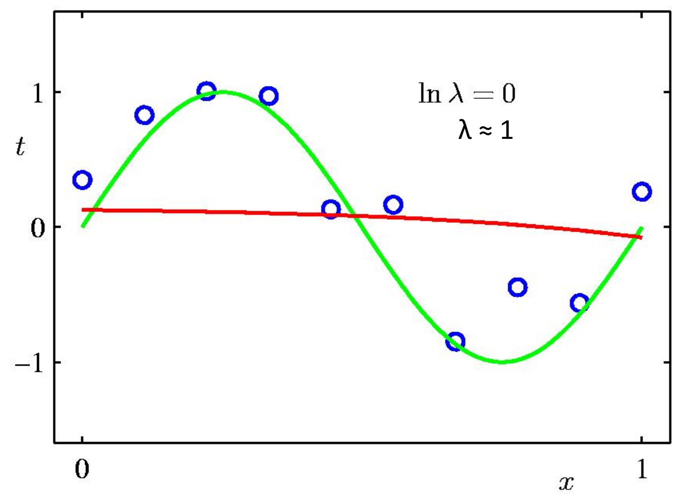
Regularization:

$$\ln \lambda = -18$$



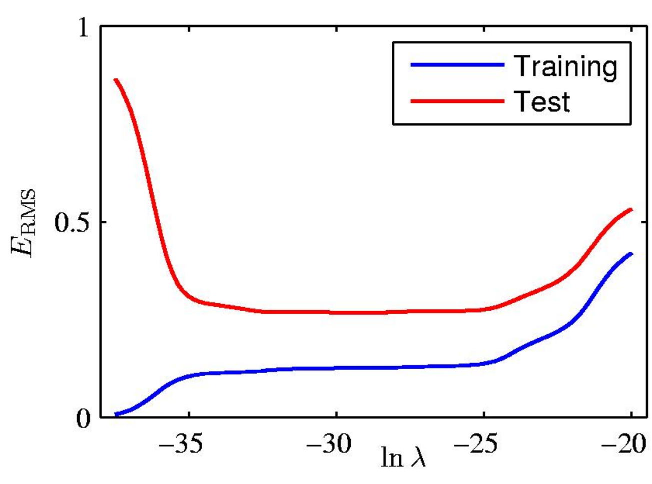
Regularization:

$\ln \lambda = 0$



Regularization:

E_{RMS} vs. $\ln \lambda$

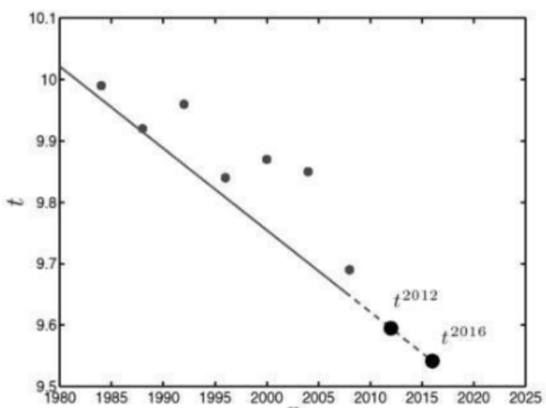


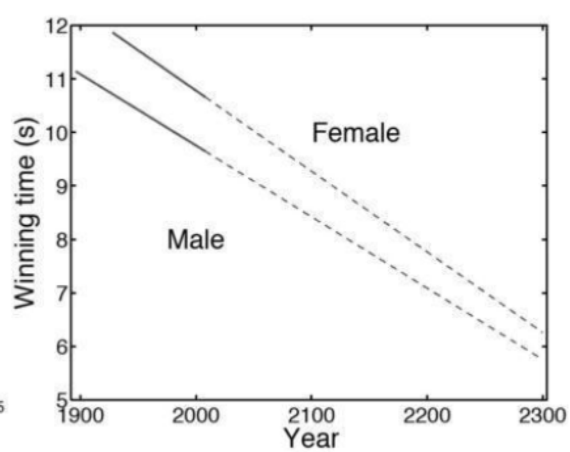
Polynomial Coefficients

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

Predicting with a learned model

Prediction: $$t_{new} = \hat{\mathbf{w}}^\top \mathbf{x}_{new} = \sum_{i=0}^{k} x_{new,i} w_i$$





2592: look out boys!

3000: -3.5 seconds ??!