EEC S 545: Machine Learning

Lecture 2. Linear Regression

Honglak Lee

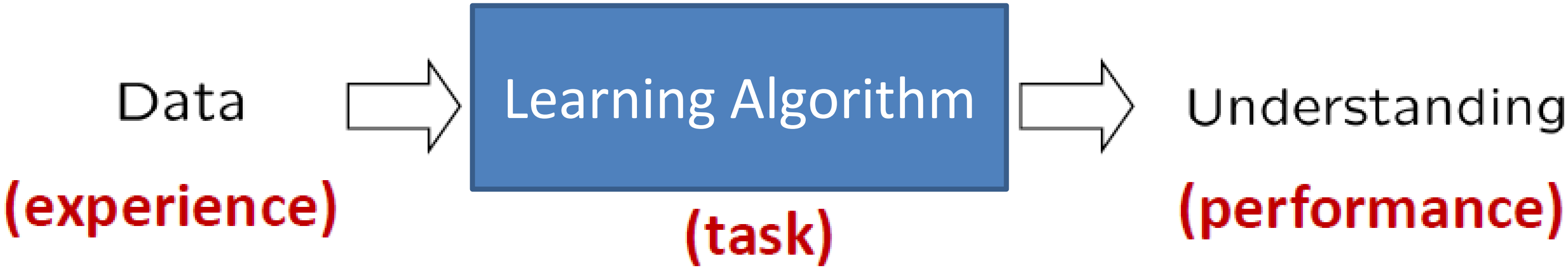
1/10/2011

Outline

- Recap: ML, Supervised Learning
- Linear Regression

Informal definition of ML

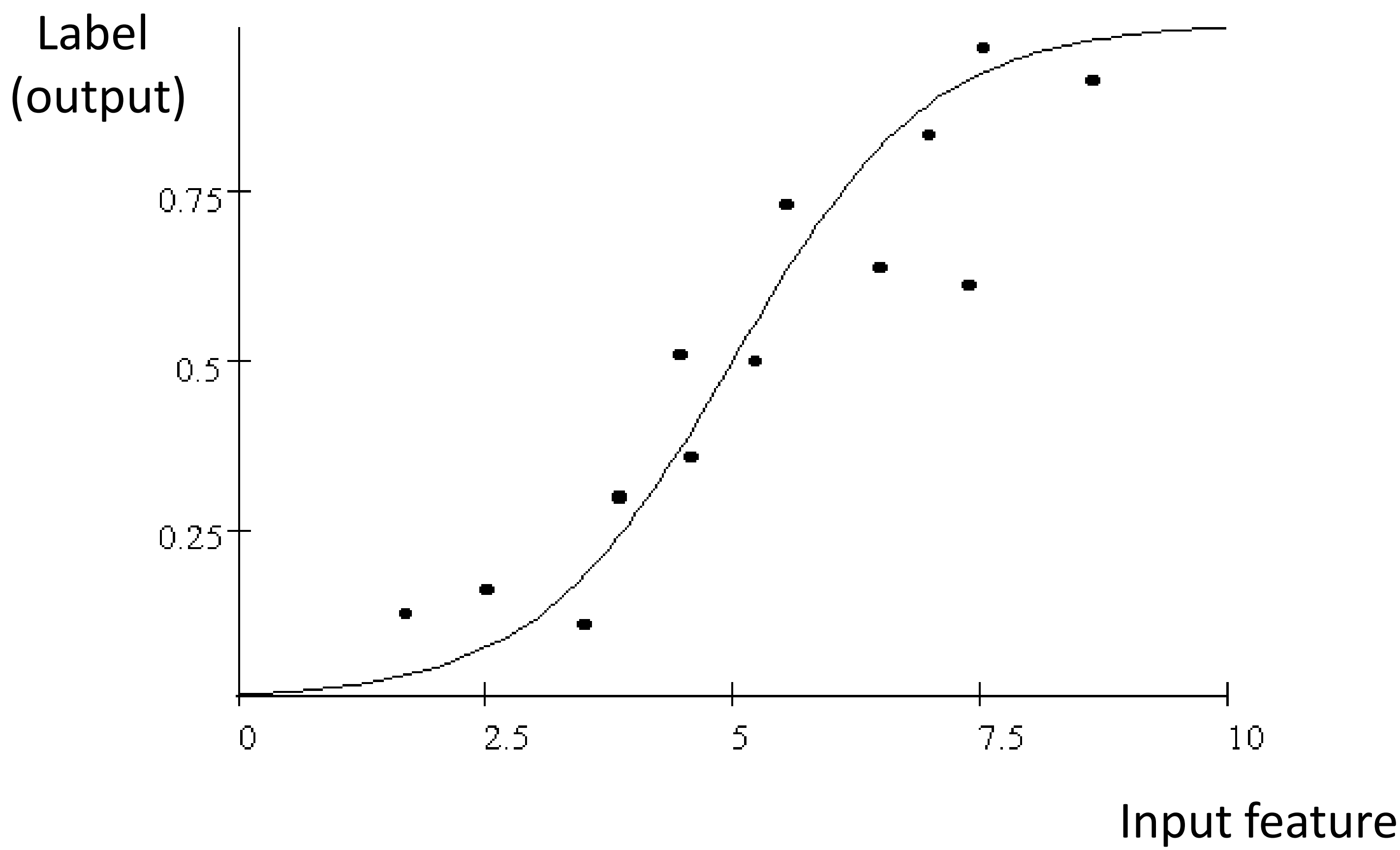
- Algorithms that improve their performance at some task with experience.



Supervised Learning

- Goal:
  - Given data X in feature space and the labels Y
  - Learn to predict Y from X
- Labels could be discrete or continuous
  - Discrete labels: classification
  - Continuous labels: regression

Supervised Learning - Regression

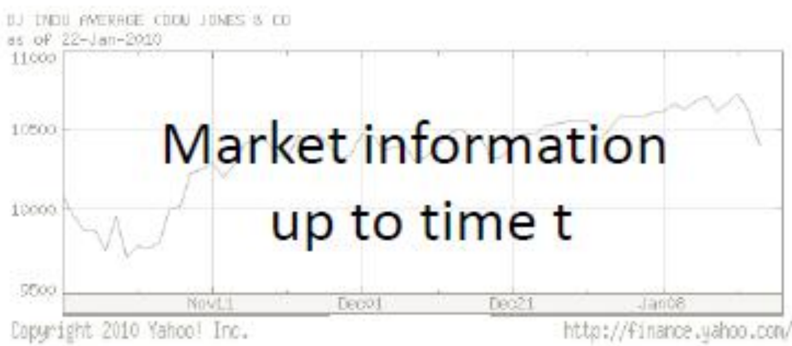


"Learning regression function f(X)"

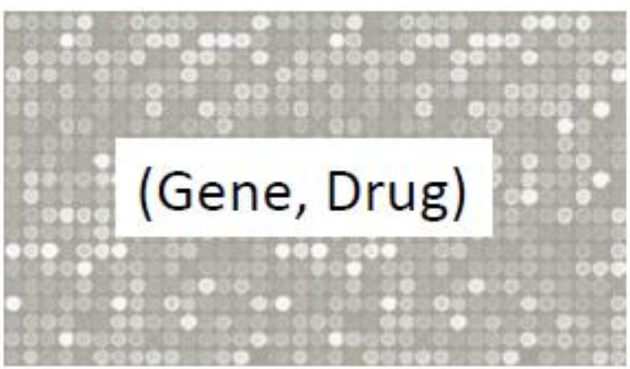
Supervised Learning - Regression

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$



Share Price
"$ 24.50"



Expression level
"0.01"

Continuous Labels

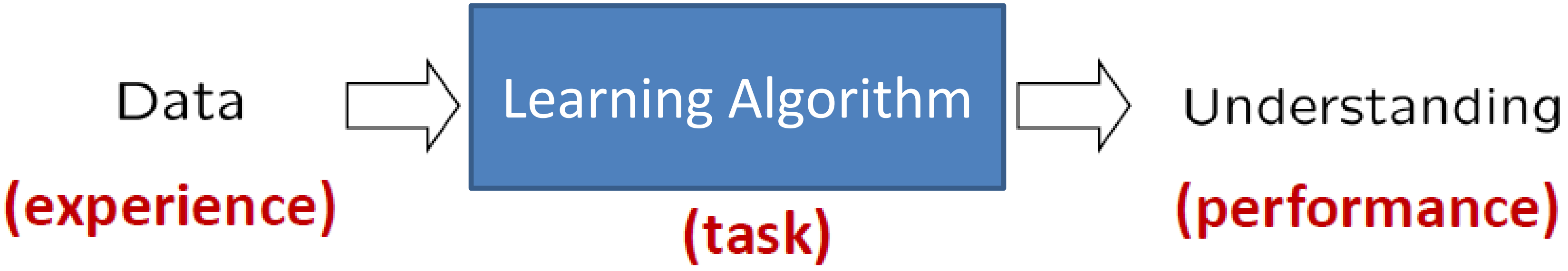
Slide credit: Aarti Singh

Example: Housing price prediction

- **Given statistics about houses in a local area, predict median value of homes.**
- **Features:**
  - 1. Average number of rooms per dwelling
  - 2. Average area (in square foot)
  - 3. Per capita crime rate by town
  - 4. Proportion of residential land zoned for lots
  - 6. Proportion of non-retail business acres per town
  - 7. Nitric oxides concentration (parts per 10 million)
  - …..
- **Label: median value of the houses**

Overview of linear regression

- In this lecture, we will assume
  - Data (or features): vector or scalar representation
  - Learning algorithm: linear regression to predict y from X (parameterized by w)
  - Performance: measured by sum of squared errors between prediction and labels (objective function)



Outline

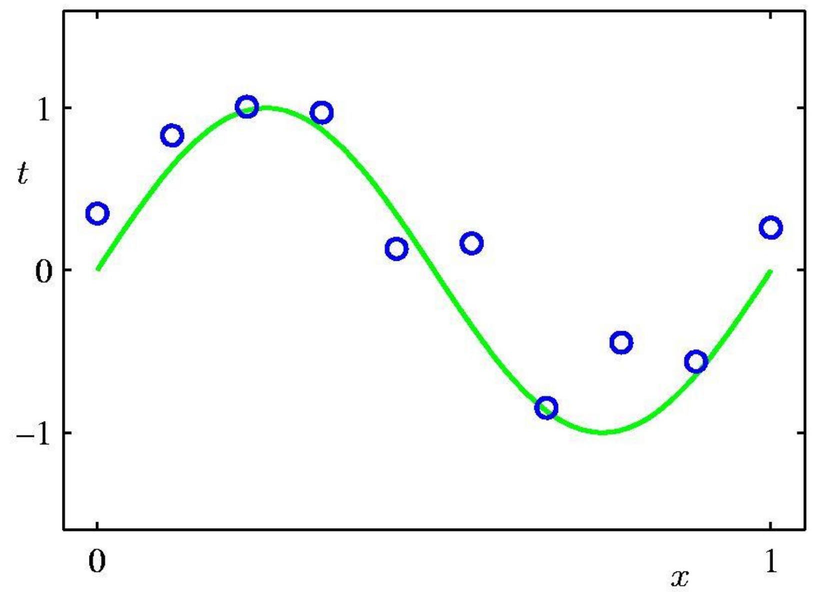
- Recap: ML, Supervised Learning
- Linear Regression

Notation

- In this lecture, we will use
  - x: data (scalar or vector)
  - $\phi(x)$: features for x
  - t (or y): continuous-valued labels (target values)
- We will interchangeably use
  - $x^{(n)} \stackrel{\text{def}}{=} x_n$ to denote n-th training example.
  - $t^{(n)} \stackrel{\text{def}}{=} t_n$ to denote n-th target value.

Regression

- Given a set of observations
  - $\mathbf{x} = \{ x_1 \ldots x_N \}$
- And corresponding target values:
  - $\mathbf{t} = \{ t_1 \ldots t_N \}$



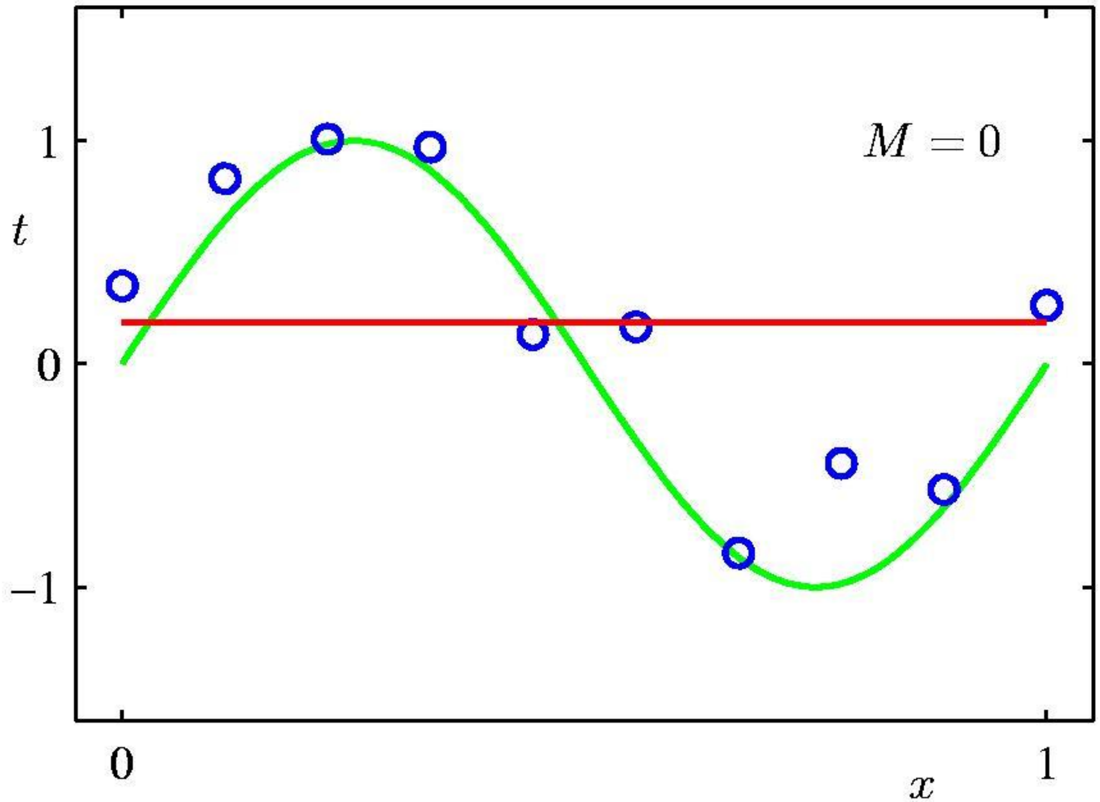
- We want to learn a function $y(x,w)=t$ to predict future values.

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

Slide credit: Ben Kuipers

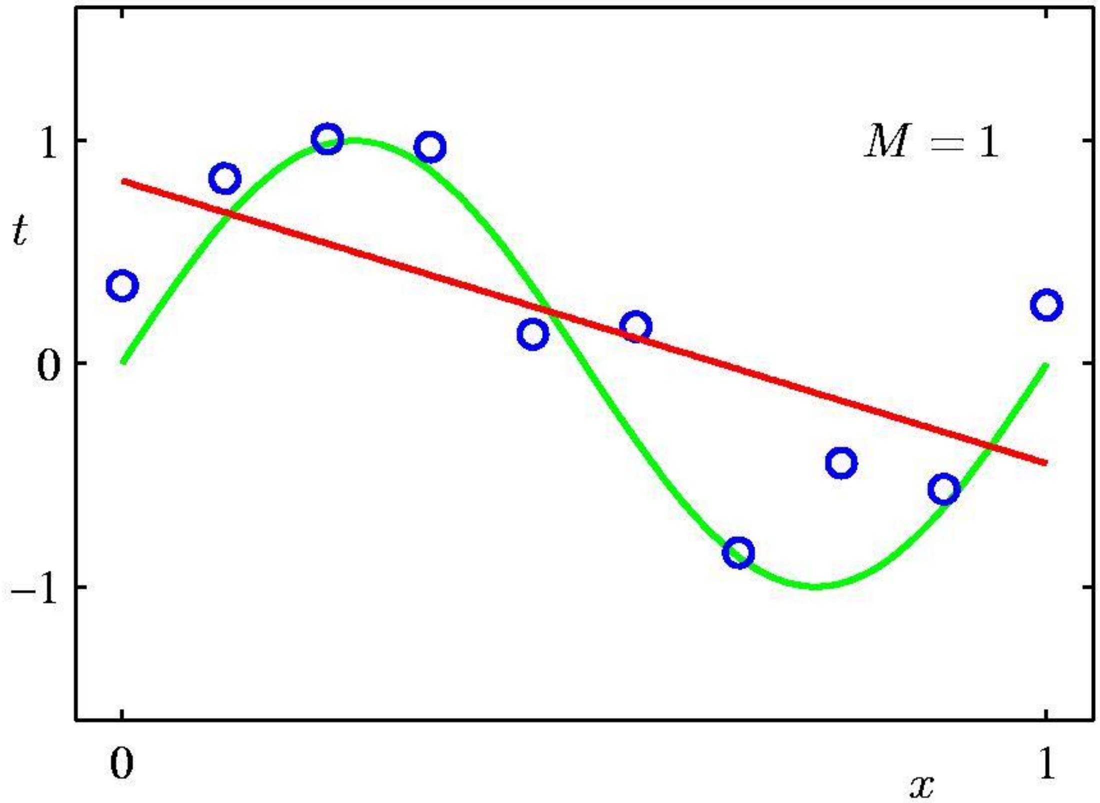
0^{th} Order Polynomial

$y = w_0$



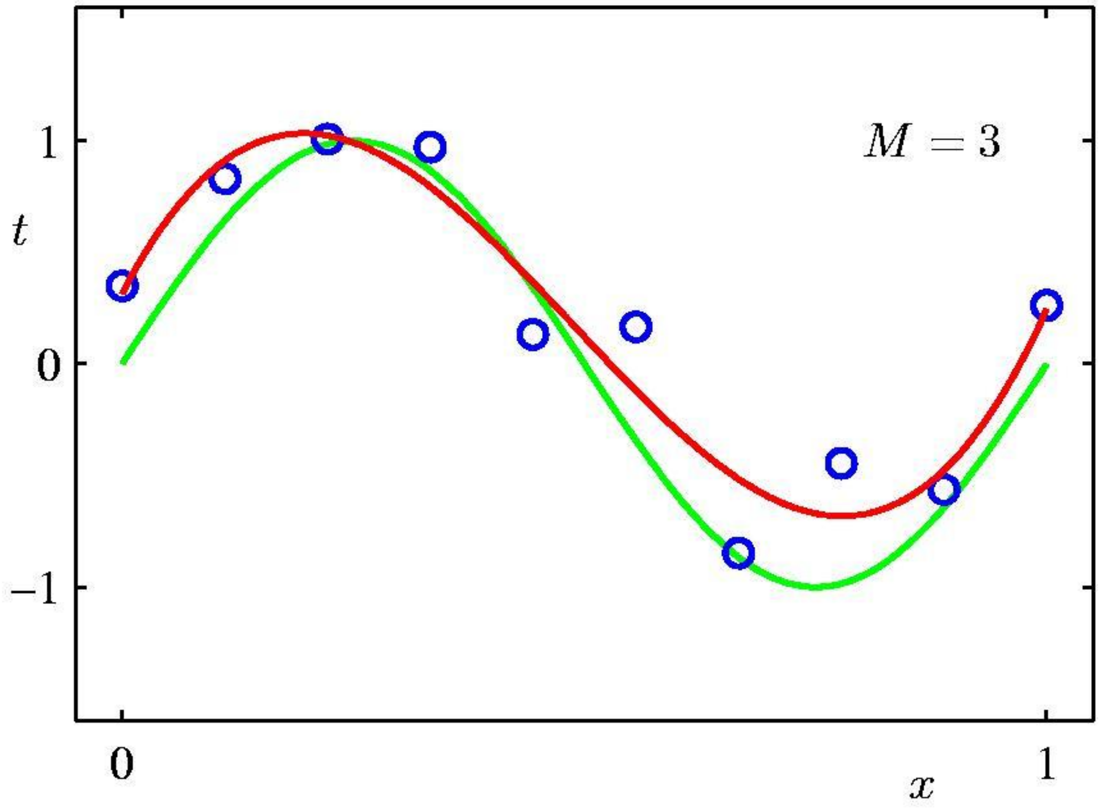
1^{st} Order Polynomial

$y = w_0 + w_1 x$



3rd Order Polynomial

$y(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$



Linear Regression

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

- The function *y*(x,w) is linear in parameters w.
  - Goal: find the best value for the weights, **w**.
- For simplicity, add a *bias function* $\phi_0(\mathbf{x}) = 1$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

$$\mathbf{w} = (w_0, \ldots, w_{M-1})^T \qquad \phi = (\phi_0, \ldots, \phi_{M-1})^T$$

Slide credit: Ben Kuipers

Basis Functions

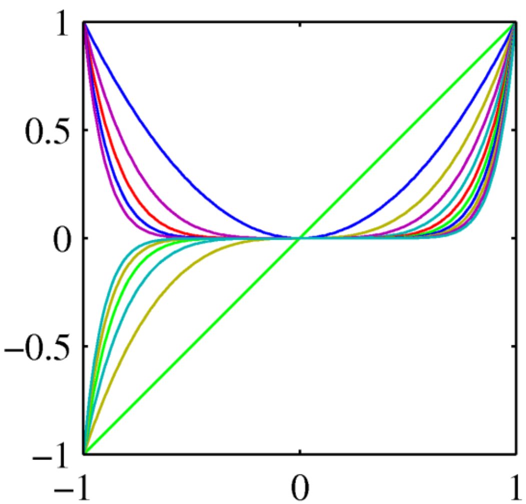
- The basis functions $\phi_j(\mathbf{x})$ need not be linear

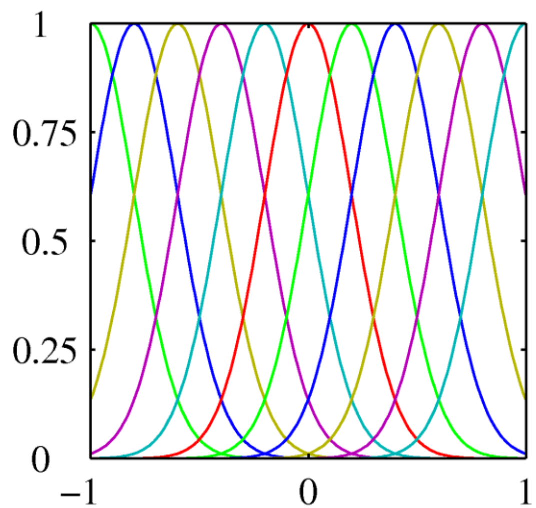
$$\phi_j(x) = x^j$$

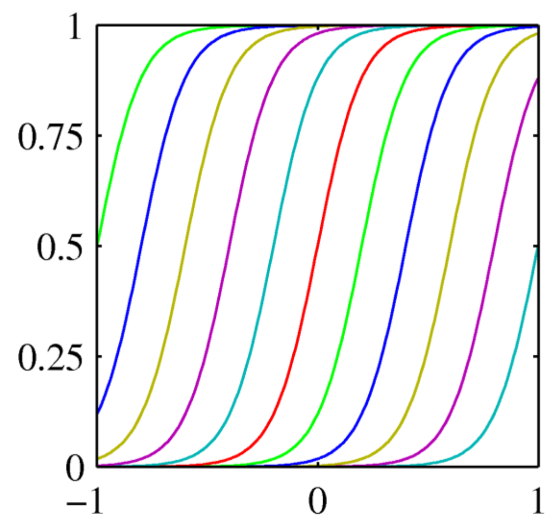
$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

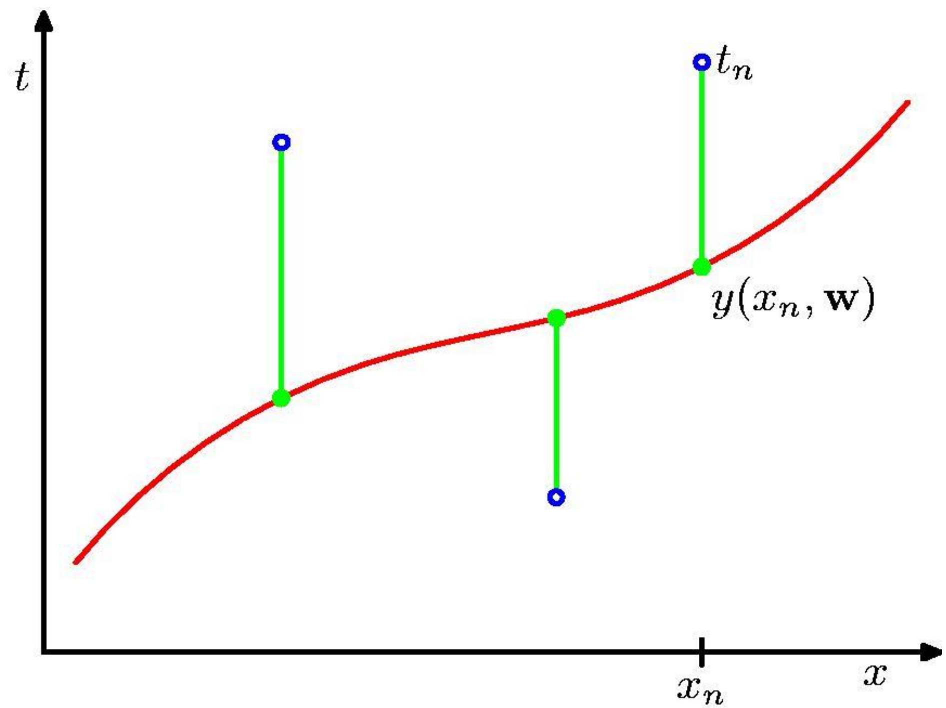






Slide credit: Ben Kuipers

Sum-of-Squares Error Function

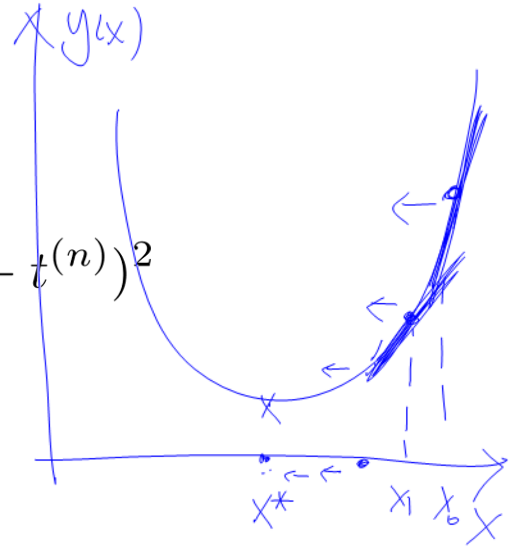


$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

Least squares problem

- Objective function

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\Big(\sum_{j=0}^{M-1} w_j\phi_j(x^{(n)}) - t^{(n)}\Big)^2$$



- Gradient

$$\begin{aligned}
\frac{\partial E(\mathbf{w})}{\partial w_j} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_{n=1}^{N}\Big(\sum_{j=0}^{M-1} w_j\phi_j(x^{(n)}) - t^{(n)}\Big)^2 \\
&= \sum_{n=1}^{N}\Big(\sum_{j\ =0}^{M-1} w_j\ \phi_j\ (x^{(n)}) - t^{(n)}\Big)\frac{\partial}{\partial w_j}\Big(\sum_{j\ =0}^{M-1} w_j\ \phi_j\ (x^{(n)}) - t^{(n)}\Big) \\
&= \sum_{n=1}^{N}\Big(\sum_{j\ =0}^{M-1} w_j\ \phi_j\ (x^{(n)}) - t^{(n)}\Big)\phi_j(x^{(n)})
\end{aligned}$$

$$\begin{bmatrix} \frac{\partial E(w)}{\partial w_1} \\ \frac{\partial E(w)}{\partial w_2} \\ \vdots \\ \frac{\partial E(w)}{\partial w_M} \end{bmatrix}$$

Least squares problem

- Gradient (compact, vectorized form)

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_{n=1}^{N} (\sum_{j'=0}^{M-1} w_{j'} \phi_{j'}(x^{(n)}) - t^{(n)}) \phi(x^{(n)})$$

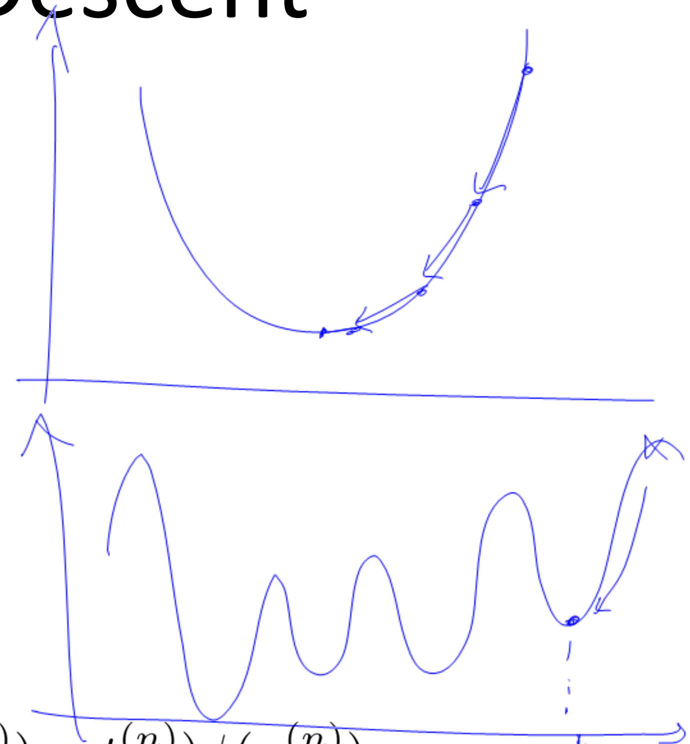
def $\|$

$$\begin{bmatrix} \frac{\partial E(w)}{\partial w_1} \\ \frac{\partial E(w)}{\partial w_2} \\ \vdots \\ \frac{\partial E(w)}{\partial w_M} \end{bmatrix} = \sum_{n=1}^{N} (\mathbf{w}^T \phi(x^{(n)}) - t^{(n)}) \phi(x^{(n)})$$

Batch Gradient Descent

- Given data (x, y), initial w
  - Repeat until convergence

$$\mathbf{w} := \mathbf{w} - \eta \nabla_{\mathbf{w}} E(\mathbf{w})$$



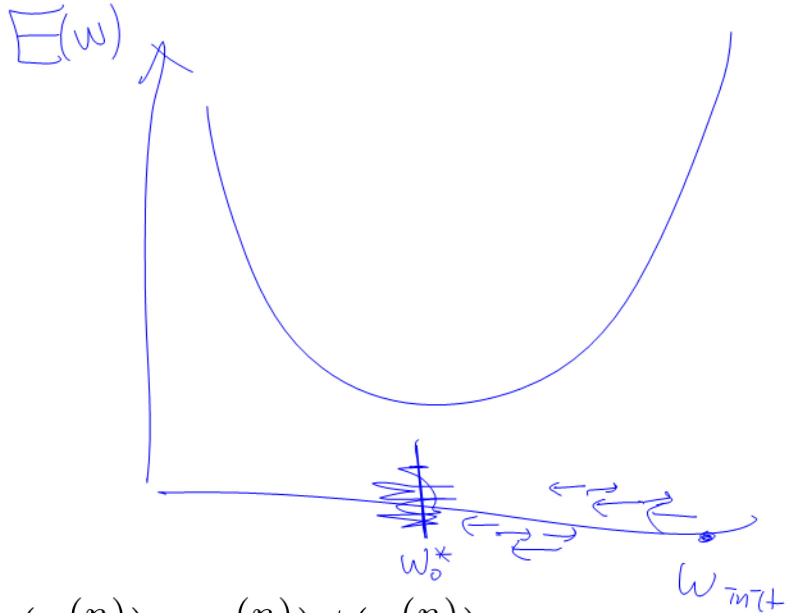
where

$$\begin{aligned}\nabla_{\mathbf{w}} E(\mathbf{w}) &= \sum_{n=1}^{N} \left(\sum_{j'=0}^{M-1} w_{j'} \phi_{j'}(x^{(n)}) - t^{(n)} \right) \phi(x^{(n)}) \\ &= \sum_{n=1}^{N} \left(\mathbf{w}^T \phi(x^{(n)}) - t^{(n)} \right) \phi(x^{(n)})\end{aligned}$$

Stochastic Gradient Descent

- Main idea: instead of computing batch gradient (over entire training data), just compute gradient for individual example and update
- Repeat until convergence
  - for n=1,…,N

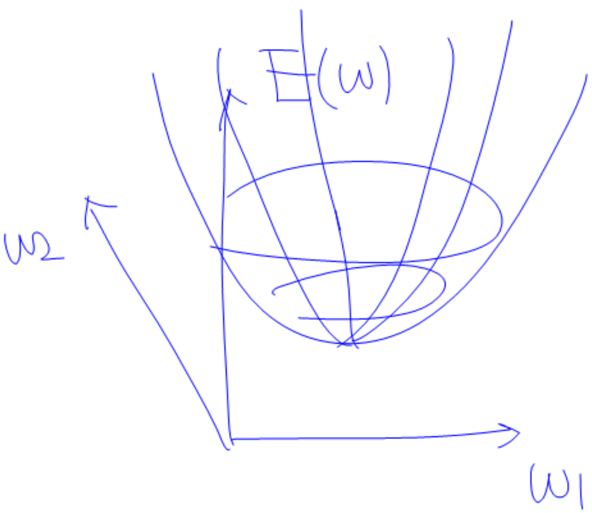
$$\mathbf{w} := \mathbf{w} - \eta \nabla_{\mathbf{w}} E(\mathbf{w}|x^{(n)})$$

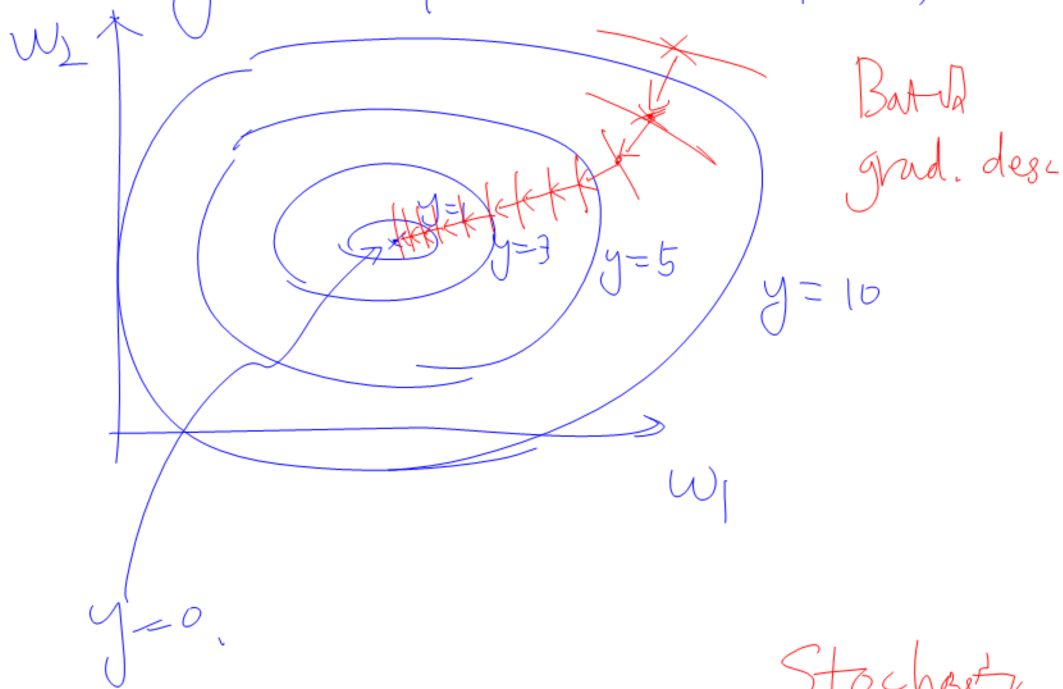


where

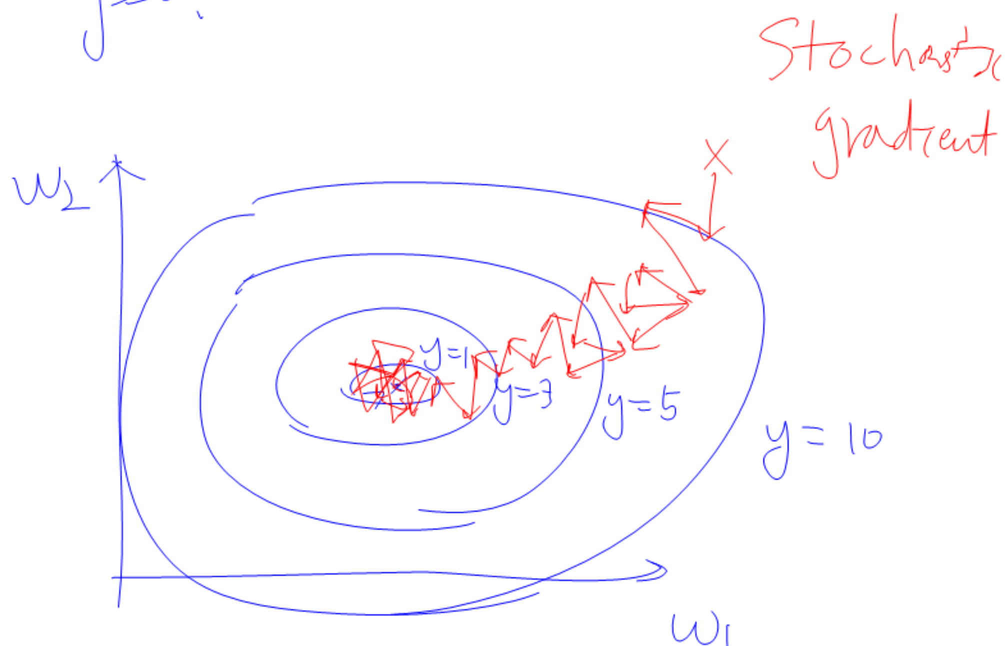
$$\begin{aligned}\nabla_{\mathbf{w}} E(\mathbf{w}|x^{(n)}) &= \Big(\sum_{j'=0}^{M-1} w_{j'}\phi_{j'}(x^{(n)}) - t^{(n)}\Big)\phi(x^{(n)}) \\ &= (\mathbf{w}^T\phi(x^{(n)}) - t^{(n)})\phi(x^{(n)})\end{aligned}$$

$$\phi_1(x),\quad \phi_2(x) \longmapsto y = w_1\phi_1(x) + w_2\phi_2(x)$$





$$\eta(t) = \frac{\eta_0}{1 + \frac{t}{t_0}}$$



Closed form solution

- Main idea:
  - Compute gradient and set gradient to 0. (condition for optimal solution)
  - Solve the equation in a closed form
- Objective function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\sum_{j=0}^{M-1} w_j \phi_j(x^{(n)}) - t^{(n)}\right)^2$$

- We will derive the gradient from matrix calculus

Closed form solution

- Objective function:

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\Big(\sum_{j=0}^{M-1} w_j\phi_j(x^{(n)}) - t^{(n)}\Big)^2 \\
&= \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\phi(x^{(n)}) - t^{(n)})^2 \\
&= \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\phi(x^{(n)}))^2 - \sum_{n=1}^{N} t^{(n)}\mathbf{w}^T\phi(x^{(n)}) + \frac{1}{2}\sum_{n=1}^{N} t^{(n)2} \\
&= \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}^T\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t}
\end{aligned}
$$

- Recap: matrix calculus (check previous review session)

$$w^T \phi(x^{(n)}) = \sum_{j=0}^{M} w_j \underbrace{\phi_j(x^{(n)})}_{\parallel}$$

$$= \sum_{j=0}^{M} \Phi_{n,j} w_j$$

$$= (\underbrace{\underset{N\times M}{\Phi} \; \underset{M\times 1}{w}}_{N\times 1})_n$$

$$\Phi w = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

N-dim

$$\sum_{n=1}^{N} \left(w^T \phi(x^{(n)})\right)^2 = \sum_{n=1}^{N} \underbrace{(\Phi w)_n^2}$$

$$= (\Phi w)^T (\Phi w) = w^T \Phi^T \Phi w$$

The Data

- The design matrix is an NxM matrix, applying
  - the M basis functions (across)
  - to N data points (down)

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \phi_0(x_n) & \phi_1(x_n) & \cdots & \phi_{M-1}(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

nth example

$\Phi_{n,j} = \phi_j(x_n)$

$$\Phi \mathbf{w} \approx \mathbf{t}$$

Slide credit: Ben Kuipers

Recap on Matrix Calculus

The Gradient

- Suppose that $f : R^{m \times n} \rightarrow R$ is a function that takes as input a matrix A of size m × n and returns a real value (scalar). Then the gradient of f (with respect to $A \in R^{m \times n}$) is the matrix of partial derivatives, defined as:

$$\nabla_A f(A) \in \mathbb{R}^{m \times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

$$(\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}.$$

The Gradient

Note that the size of $\nabla_A f(A)$ is always the same as the size of A. So if, in particular, A is just a vector $x \in \mathbb{R}^n$,

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}.$$

- $\nabla_x(f(x) + g(x)) = \nabla_x f(x) + \nabla_x g(x)$.
- For $t \in \mathbb{R}$, $\nabla_x(t\, f(x)) = t\nabla_x f(x)$.

Gradients and Hessians of Quadratic and Linear Functions (Recap)

- $\nabla_x b^T x = b$ $\quad \frac{\partial}{\partial x_i}\left(\sum_{i=1}^{N} b_i x_i\right) = b_i$
- $\nabla_x x^T A x = 2Ax$ (if A symmetric)
- $\nabla_x^2 x^T A x = 2A$ (if A symmetric)

$$\frac{\partial}{\partial x_i}\left[\sum_{i,j} x_i A_{ij} x_j\right] = 2\sum_{j} A_{ij} x_j$$

Gradient via matrix calculus

- Compute gradient and set to zero

$$
\begin{aligned}
\nabla_{\mathbf{w}} E(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2}\mathbf{w}^T \overbrace{\Phi^T\Phi}^{A}\mathbf{w} - \mathbf{w}^T \underbrace{\Phi^T\mathbf{t}}_{= b} + \frac{1}{2}\mathbf{t}^T\mathbf{t} \right] \\
&= \underbrace{\Phi^T\Phi}_{= A}\mathbf{w} - \underbrace{\Phi^T\mathbf{t}}_{= b}
\end{aligned}
$$

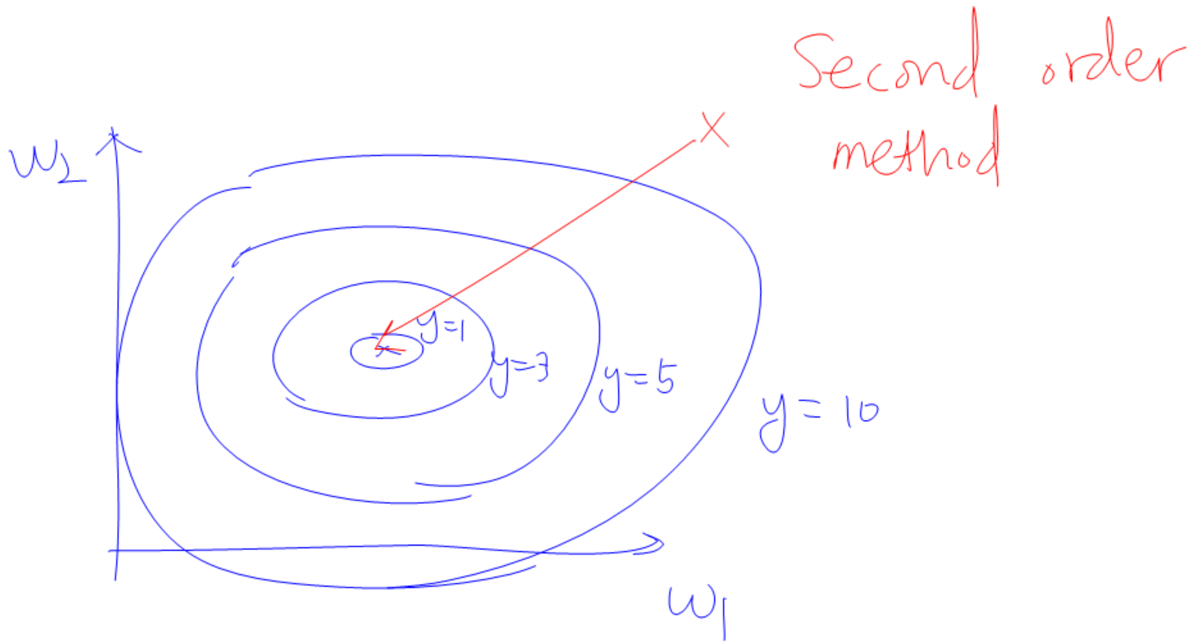
- Solve the resulting equation (normal equation)

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

$$
\mathbf{w}_{ML} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t}
$$

This is the *Moore-Penrose pseudo-inverse*: $\boldsymbol{\Phi}^\dagger = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T$

applied to: $\boldsymbol{\Phi}\mathbf{w} \approx \mathbf{t}$



$y=0.$

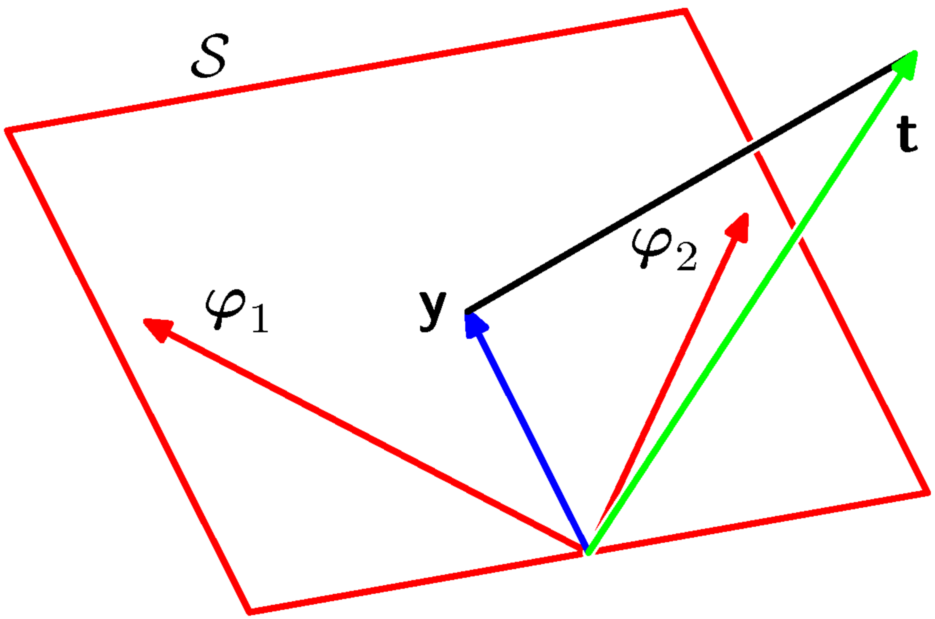
Geometric Interpretation

- Assuming many more observations (N) than the M basis functions $\phi_j(\mathbf{x})$
- View the observed target values t={t_1 … t_N} as a vector in an N-dimensional space.
- The M basis functions $\phi_j(\mathbf{x})$ span an M-dimensional subspace.
- y(x,w_{ML}) is the point in the subspace with minimal squared error from t.
- It's the projection of t onto that subspace.

Slide credit: Ben Kuipers

Geometric Interpretation

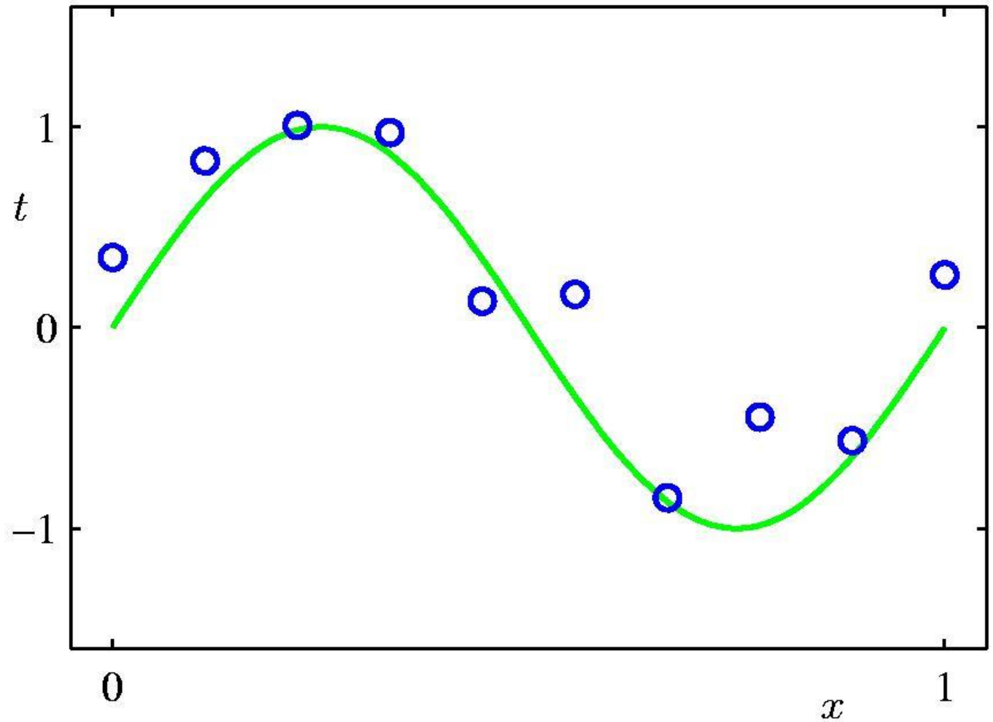
- $y(x, w_{ML})$ is the projection of t onto the subspace spanned by the M basis functions $\phi_j(\mathbf{x})$



Slide credit: Ben Kuipers

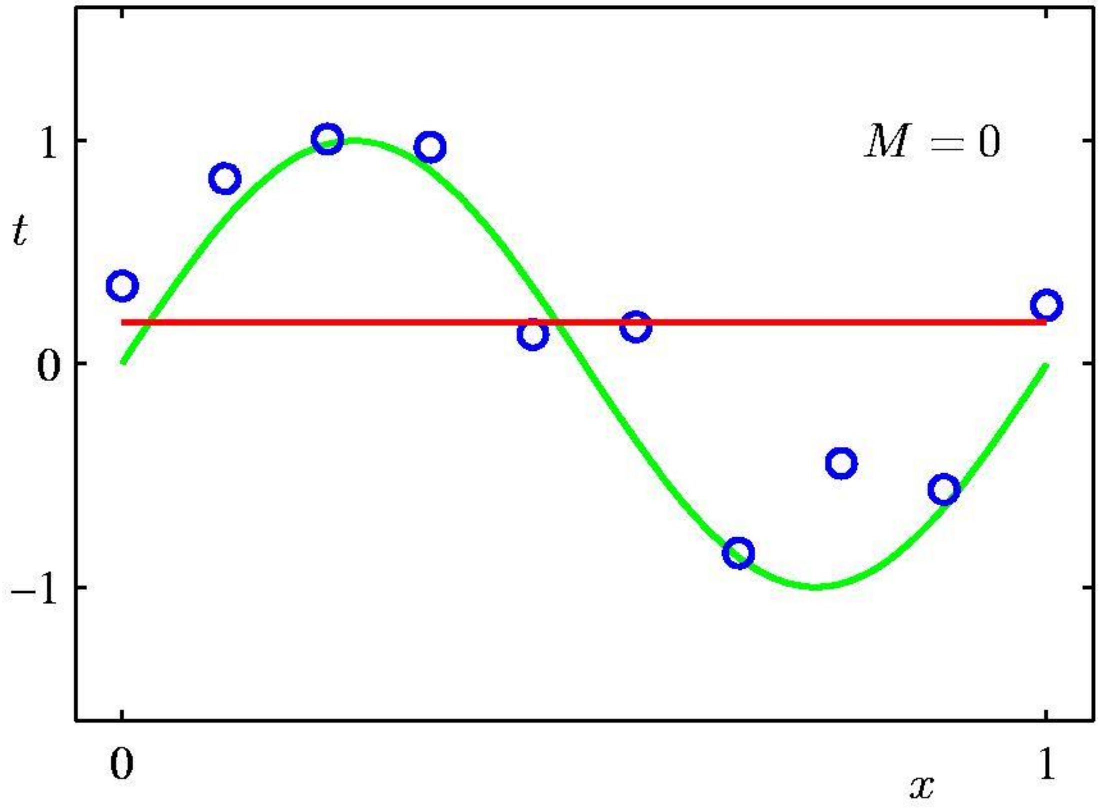
Back to curve-fitting examples

Polynomial Curve Fitting

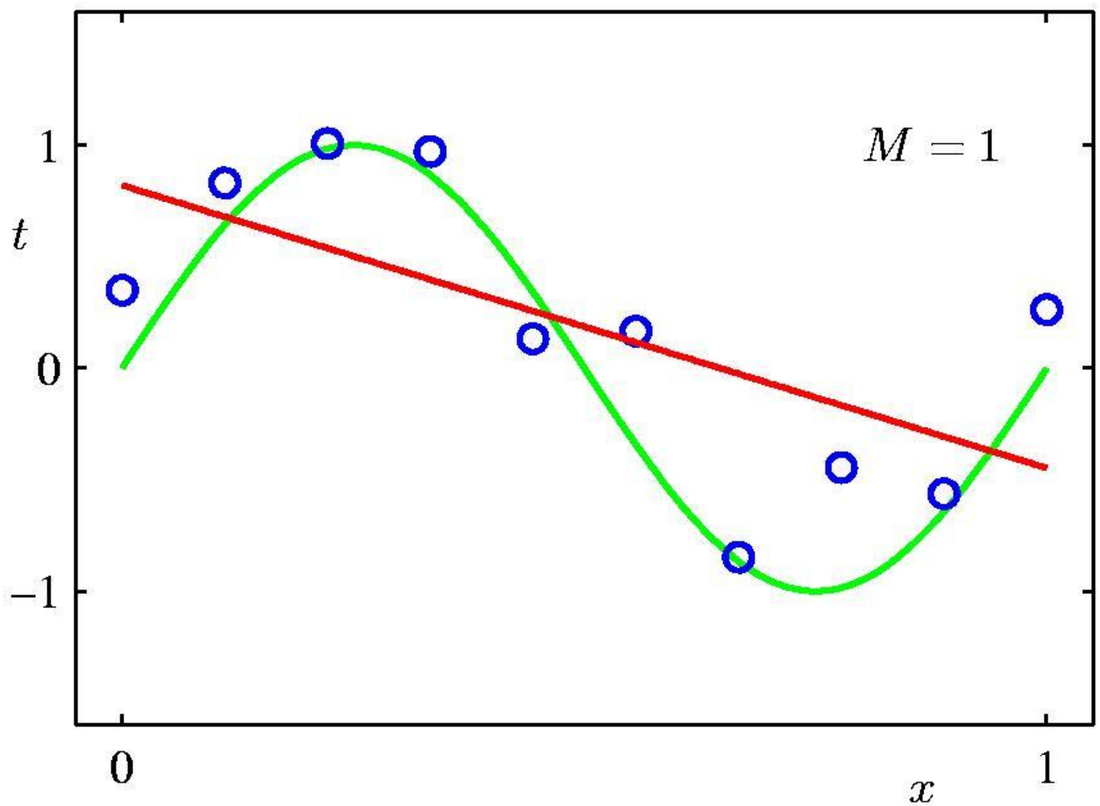


$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

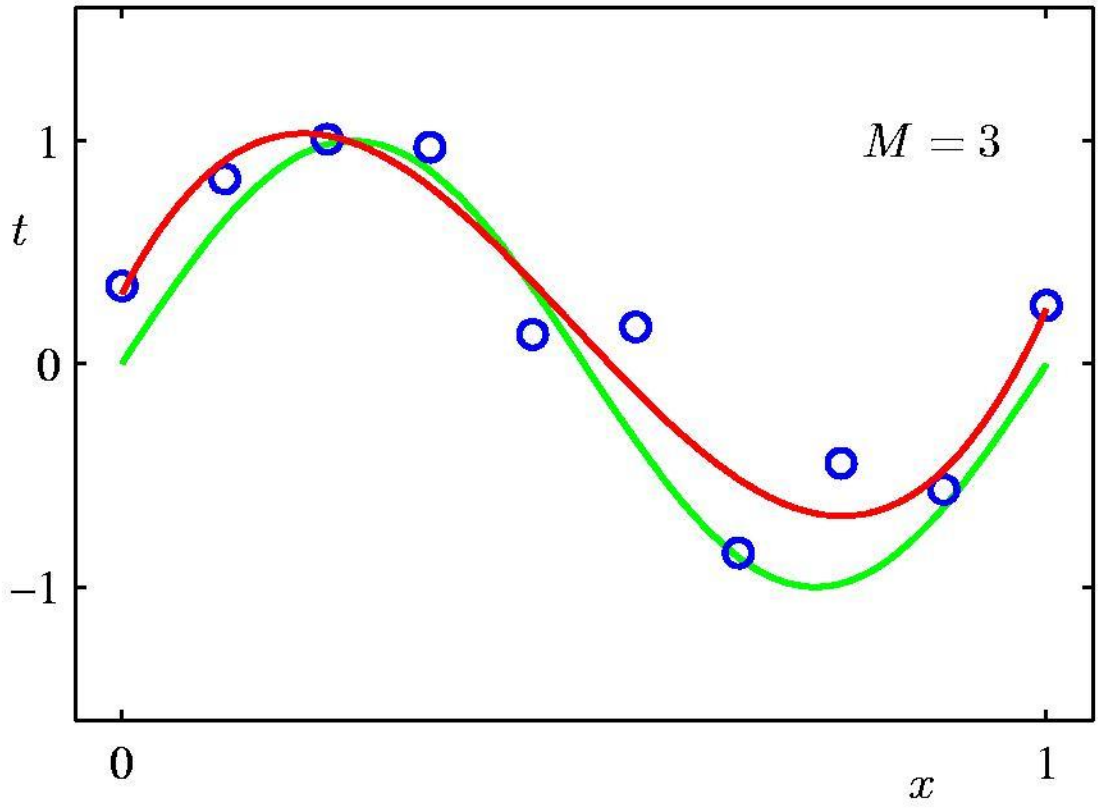
0^{th} Order Polynomial



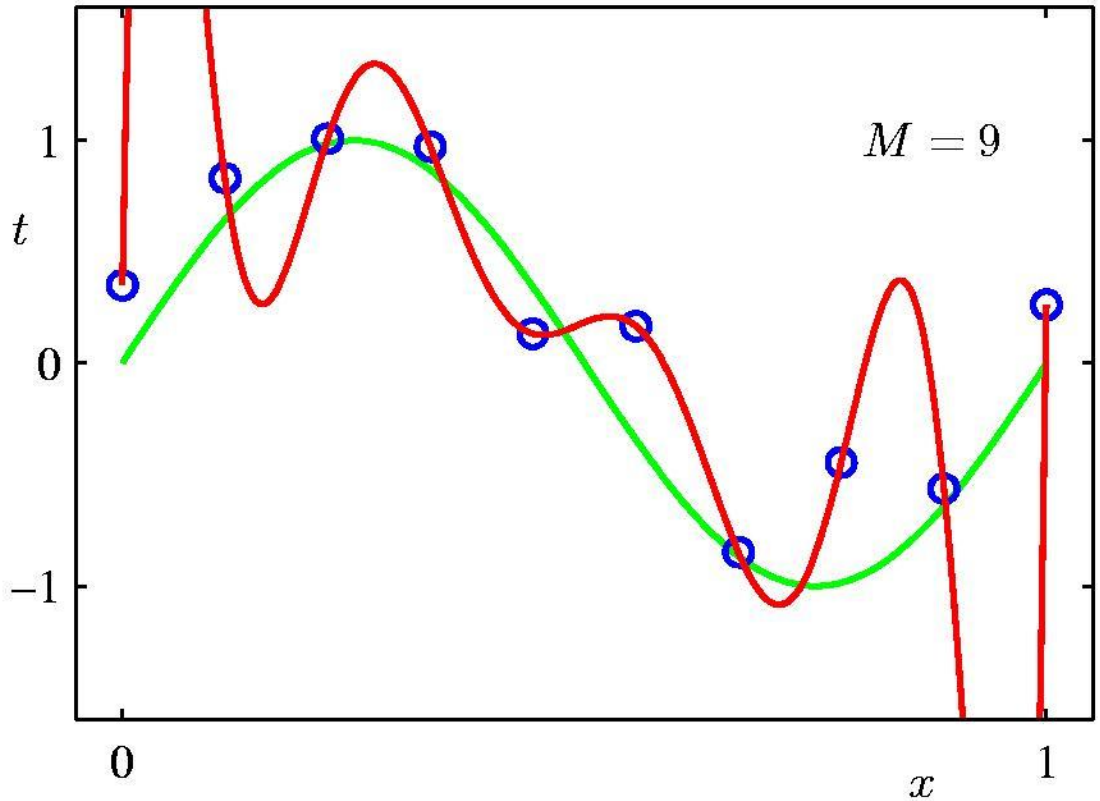
1st Order Polynomial



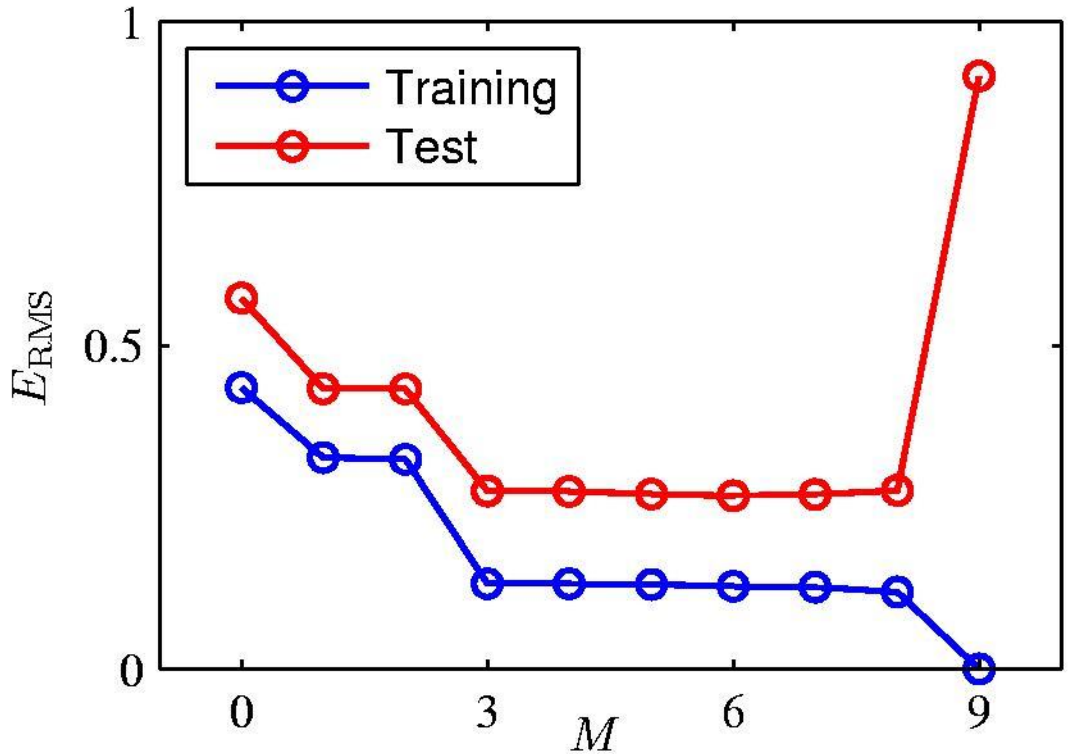
3rd Order Polynomial



9th Order Polynomial



Over-fitting



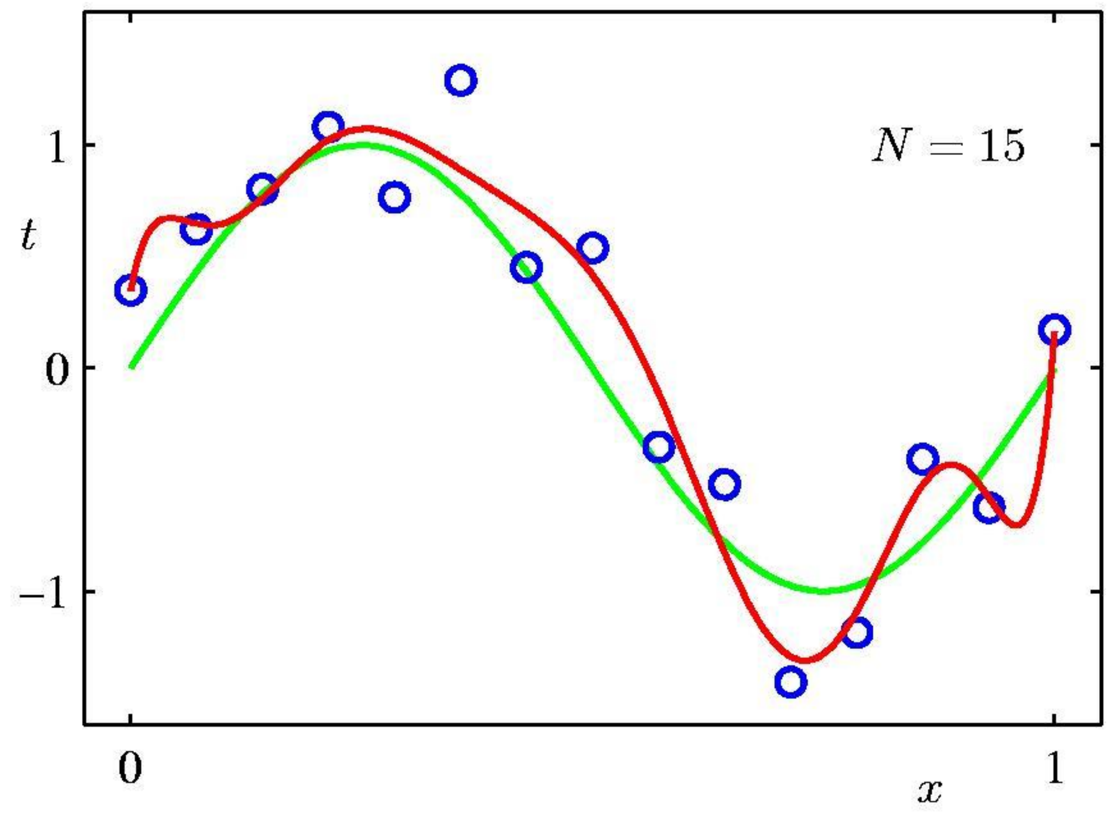
Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

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

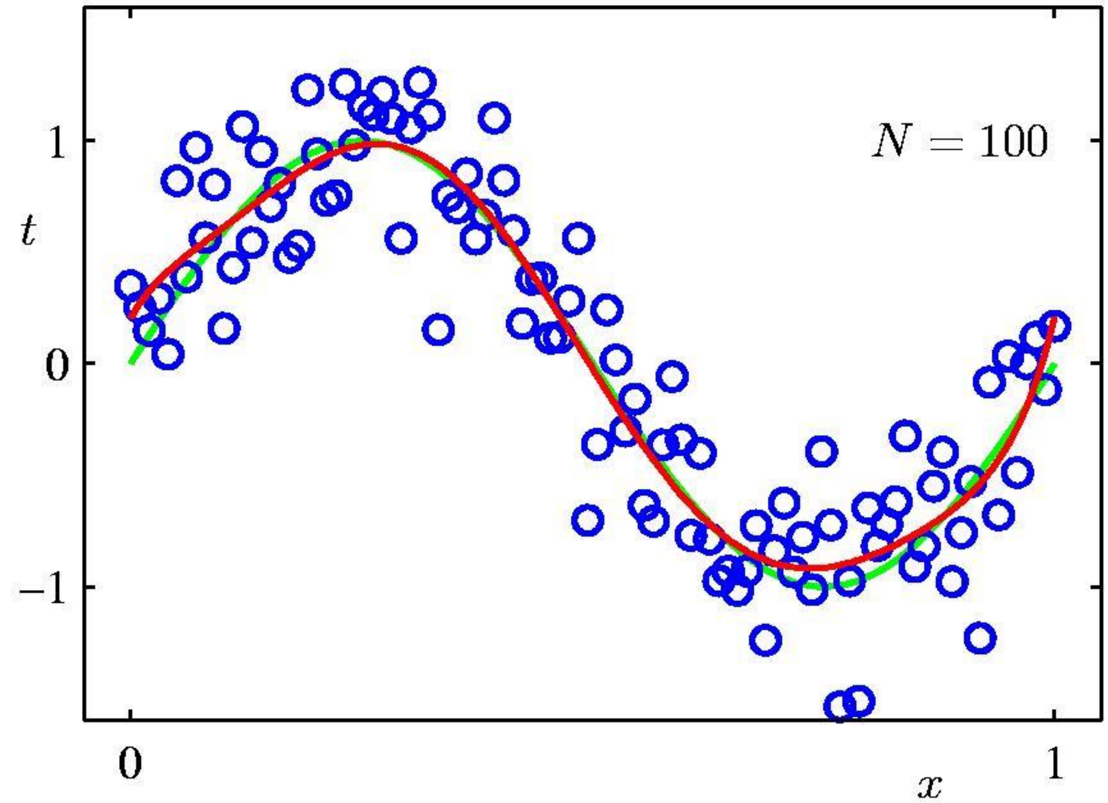
Data Set Size: $N = 15$

9th Order Polynomial



Data Set Size: $N = 100$

9th Order Polynomial



Regularization

- Penalize large coefficient values

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

Regularized Least Squares

- Add a regularization term to the error function

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

- Usual choices (sums of squares):

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(x_n)\}^2 \qquad E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

- Total error function becomes:

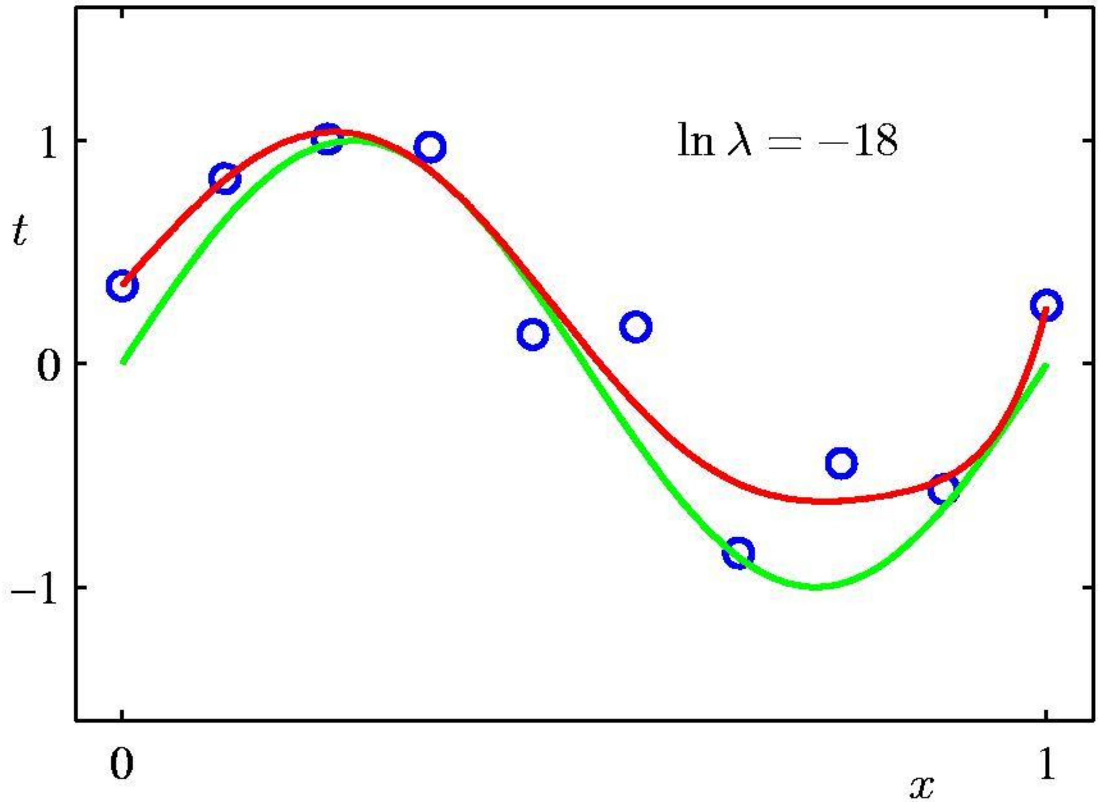
$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(x_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- Can still solve explicitly:

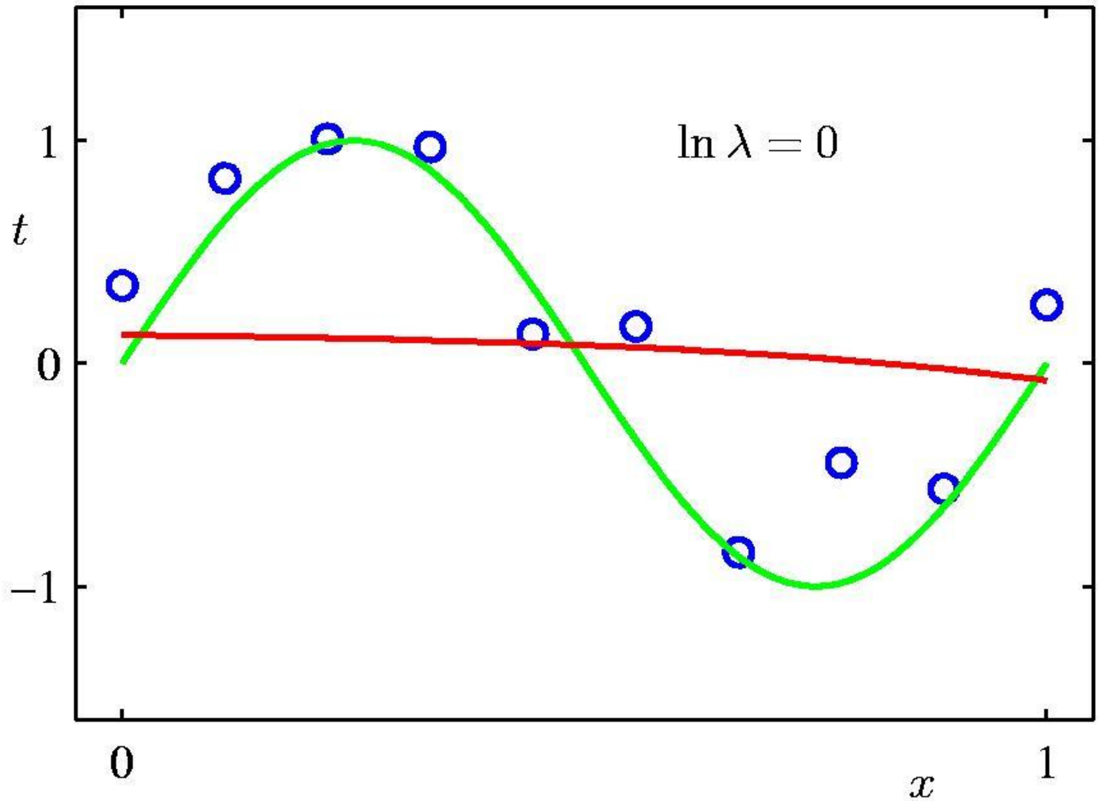
$$\mathbf{w}_{ML} = (\lambda\mathbf{I} + \mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t}$$

Slide credit: Ben Kuipers

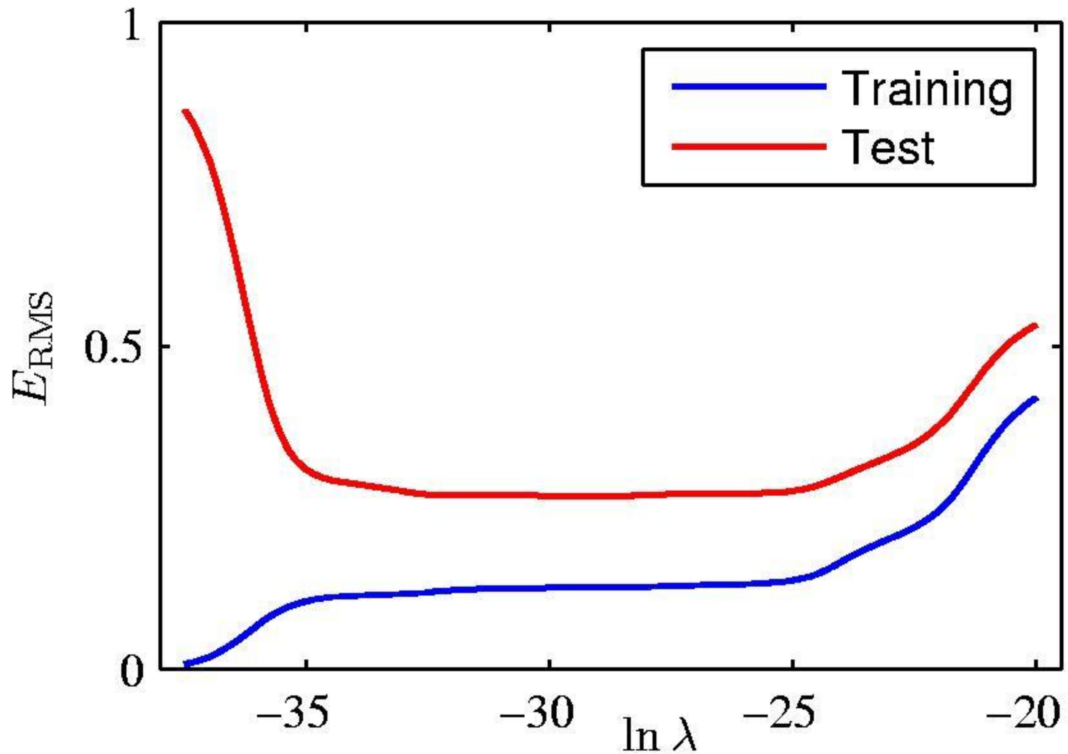
Regularization: $\ln \lambda = -18$



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