Function approximation (Regression)

Some Common Themes

Illustrated through (Generalized) MLR

Readings/Notation: I'll closely follow Bishop Ch 3.1, 3.2, which uses machine learning notation: parameters are w's (for weights), dependent variable is "t" for target, and model produces output "y".

Function Approximation / Regression/Prediction

- A predictive modeling technique
  - Given:
    - A set of input (AI) /independent (math)/ explanatory or predictor (stats) variables X
    - corresponding (set of) output/dependent/response variables Y
  - Build: a model relating X to Y
    - single value for Y given X (more common)
      - e.g. E[Y|X], the "regression of y on X.
      - Assumes Y = function of X + (zero-mean, symmetric) noise
      - Add Confidence Interval (e.g. based on the Normally distributed noise term in MLR)
    - (Arbitrary) Distribution of Y given X

Parametric Models

Determine functional form of model (e.g. polynomials)

- "learn" the parameters (weights) of the model using the training data.

- Example: linear regression

- Generalize: linear combination of basis functions (basis function expansion)

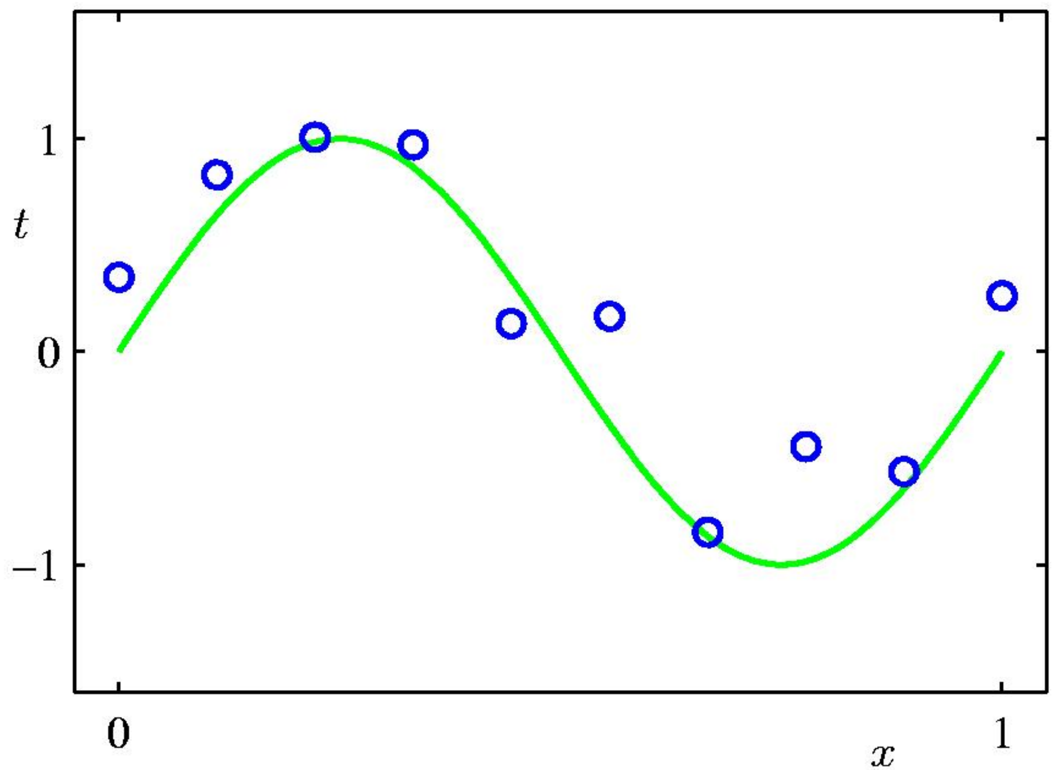
$$y(x, \mathbf{w}) = \sum_{i=0}^{M} w_i \phi_i(x) = \mathbf{w}^\top \boldsymbol{\phi}(x)$$

  – Special Case: linear regression.
  – Special Case: polynomial: (with scalar x)

$$y(x, \mathbf{w}) \;=\; w_0 + w_1 x + \ldots + w_M x^M$$

so that the basis functions are given by $\phi_i(x) = x^i$

Polynomial Curve Fitting



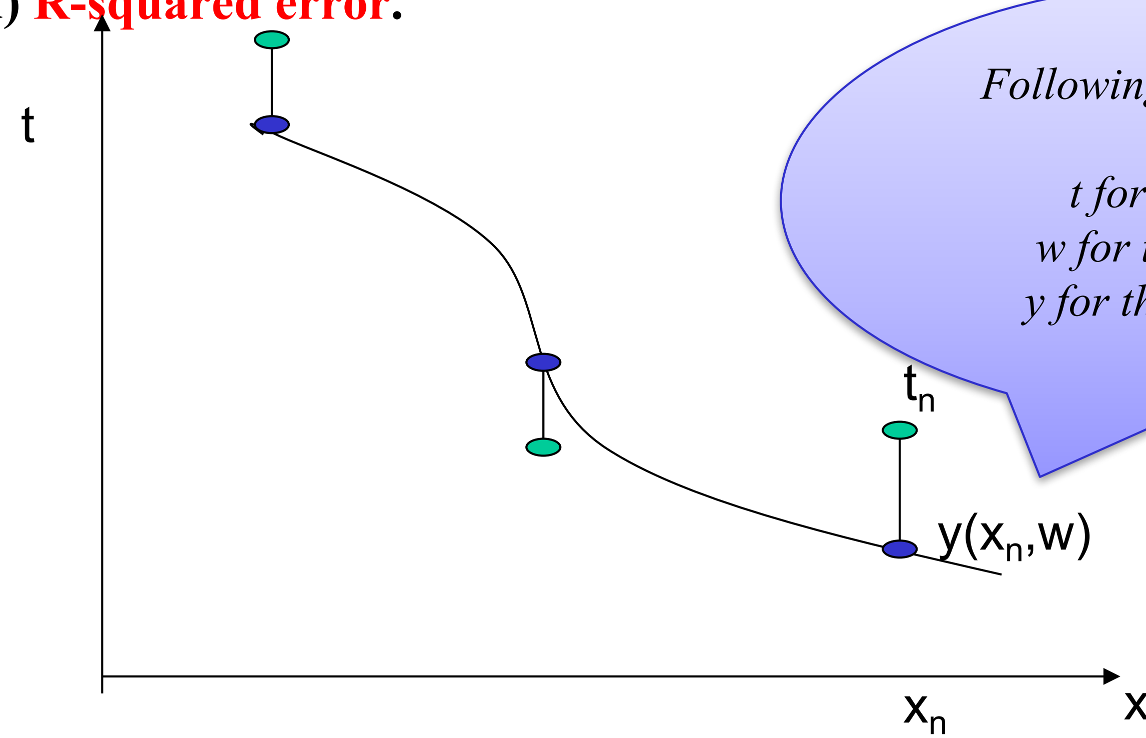
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Least Squares

- Minimize sum-of-squares **error (SSE) (t's are the target values)**

$$E(\mathbf{w}) = \sum_{n=1}^{N} \{\mathbf{w}^{\top}\phi(x_n) - t_n\}^2$$

Best interpretation? Consider Root Mean Squared Error (RMSE) or (adjusted) R-squared error.



Following Bishop (06), am using
t for target values
w for the parameters
y for the model output

Least Squares Solution

- Exact closed-form minimizer (ML solution)

$$\mathbf{w}^* = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \vec{t}$$

where $\vec{t} = (t_1, \ldots, t_N)^\top$

  – "Pseudo-inverse solution"
    and $\boldsymbol{\Phi}$ is the *design matrix* given by

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \cdots & \phi_M(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \cdots & \phi_M(\mathbf{x}_2) \\ \vdots & \vdots & \vdots \\ \phi_0(\mathbf{x}_N) & \cdots & \phi_M(\mathbf{x}_N) \end{pmatrix}$$

Takeaway: direction solution involves inversion of an (M+1)X(M+1) matrix

Explicitly shows collinearity problem

Multiple outputs?

$$\mathbf{w}_k^* = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \vec{t}_k$$

(psuedo-inverse portioned part shared by all outputs; rest de-coupled.)

Geometry of Least Squares

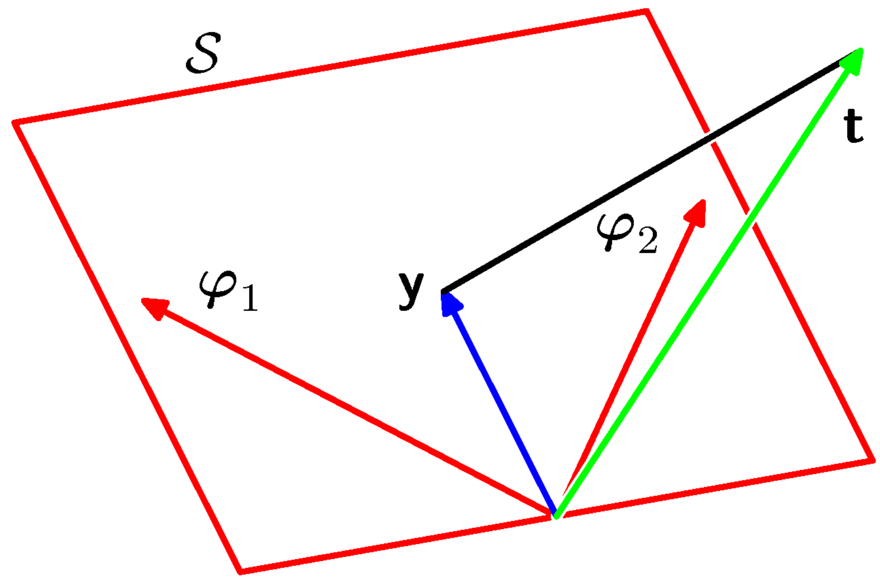
•Consider

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w}_{\mathrm{ML}} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M]\,\mathbf{w}_{\mathrm{ML}}.$$

$$\mathbf{y} \in \mathcal{S} \subseteq \mathcal{T} \qquad \mathbf{t} \in \mathcal{T}$$

N-dimensional

M-dimensional



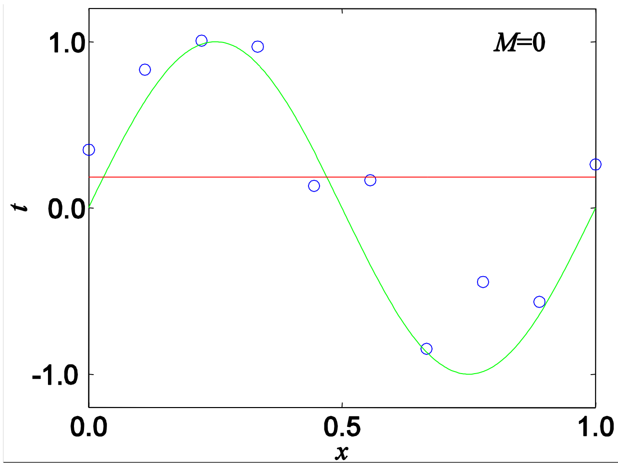
•S is spanned by $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M$.

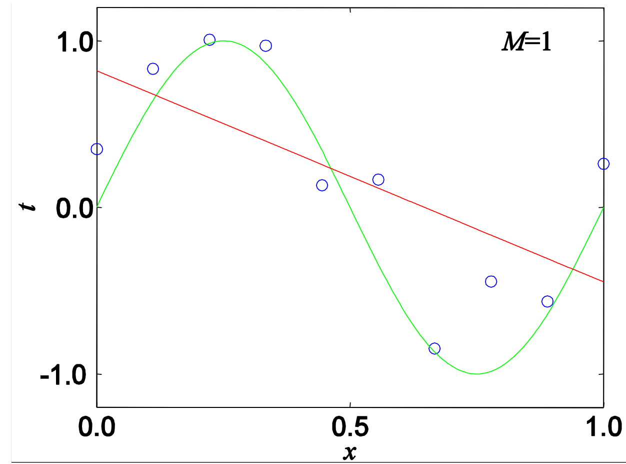
•w_{ML} minimizes the distance between t and its orthogonal projection on S, i.e. y.

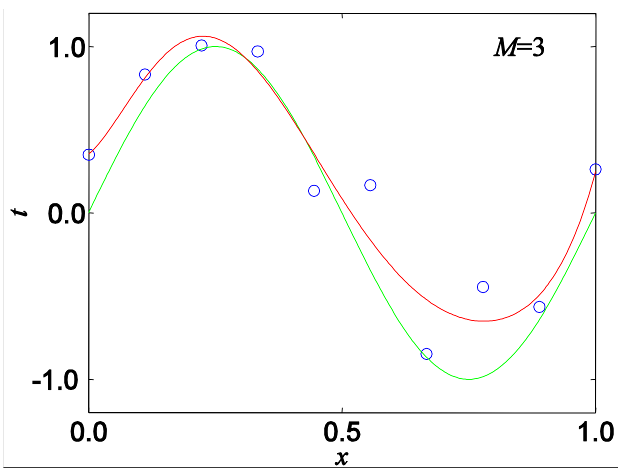
Takeaway: You are restricted by your choice of the features

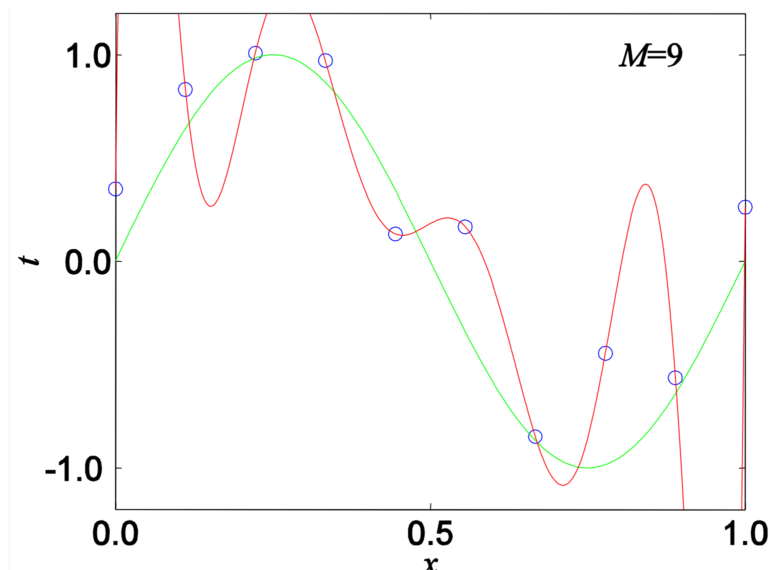
Model Complexity and Overfitting

- "Noisy sine" example from Chris Bishop

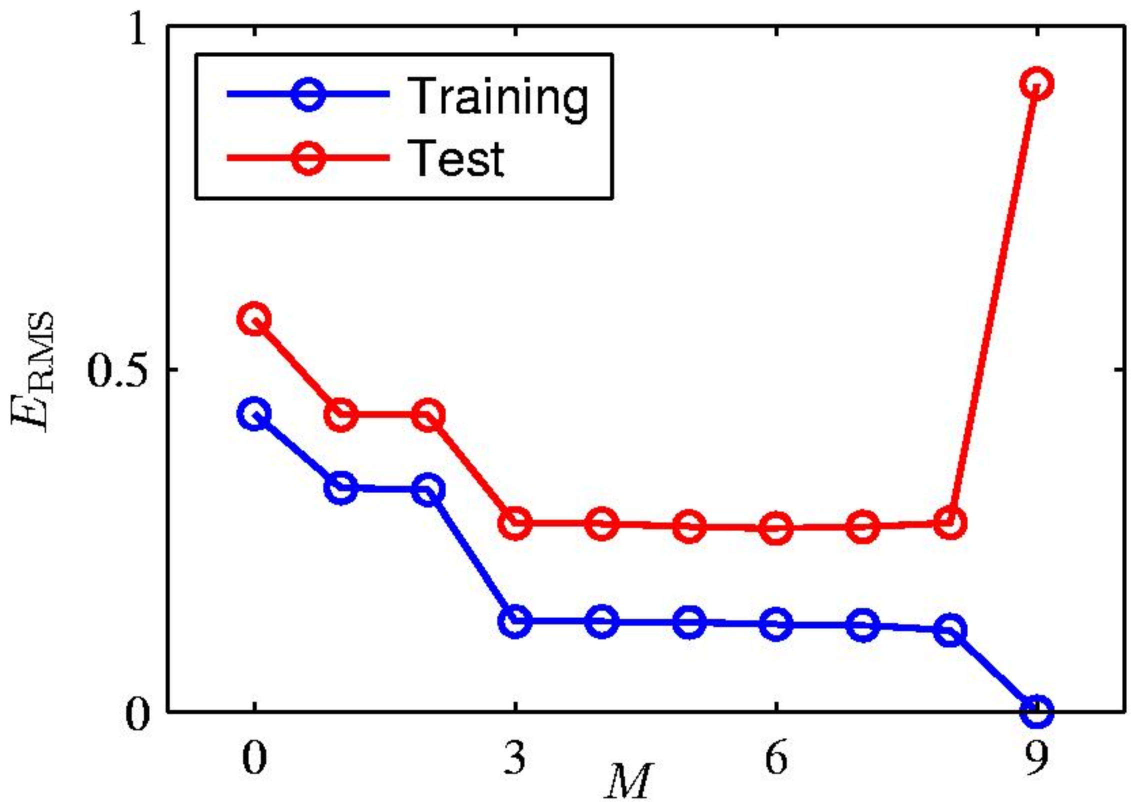








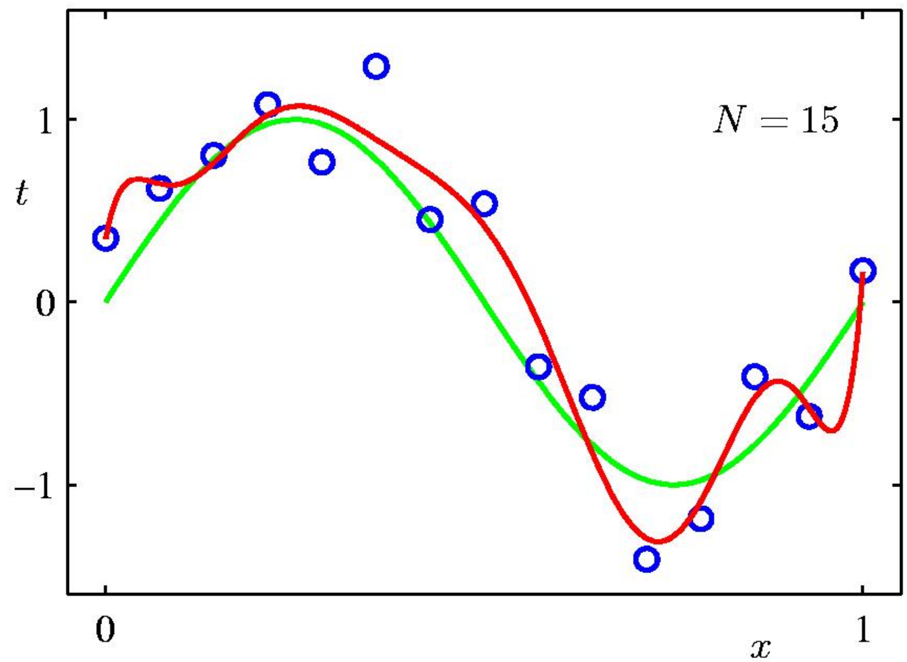
Over-fitting



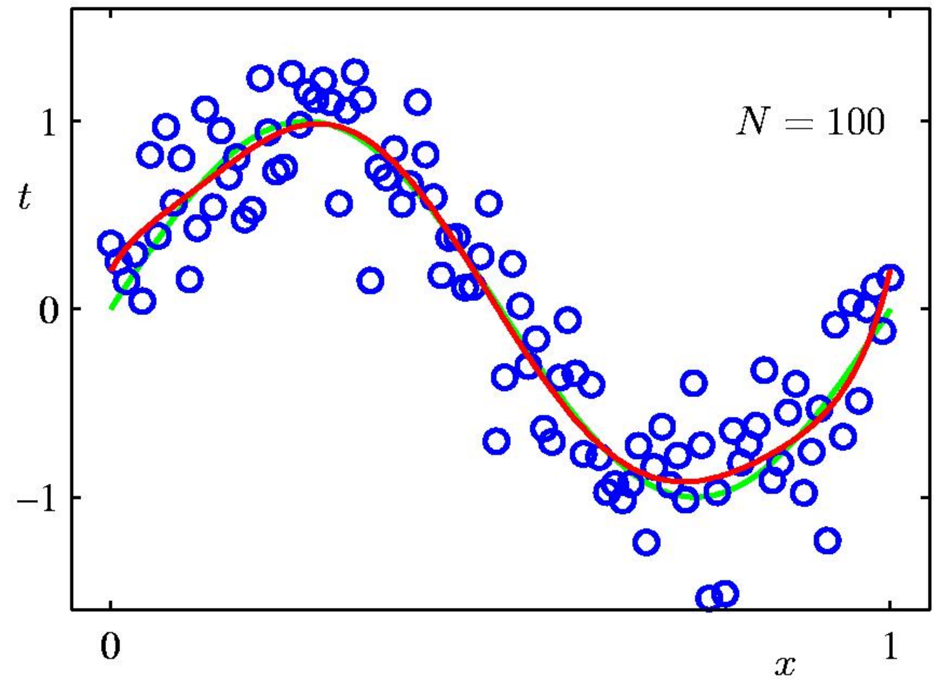
Root-Mean-Square (RMS) Error vs Polynomial order

Data Set Size:

9th Order Polynomial



$N = 15$



$N = 100$

Regularization (to avoid overfitting)

- “regularization term” imposes penalty on less desirable solutions
  - Cost = MSE + λ Penalty (f)
  - Regularization Penalty is a functional (maps each function f onto a number)
- Popular Penalties
  - **ridge regression** (sum squared of weights)
  - **Lasso** (sum of |w|; for large λ yields sparse models)
    - **Elastic net:** combines both ridge and Lasso
  - number of non-zero weights
  - smoothness of function

(note: “intercept”, i.e. w_0, not included in penalties)

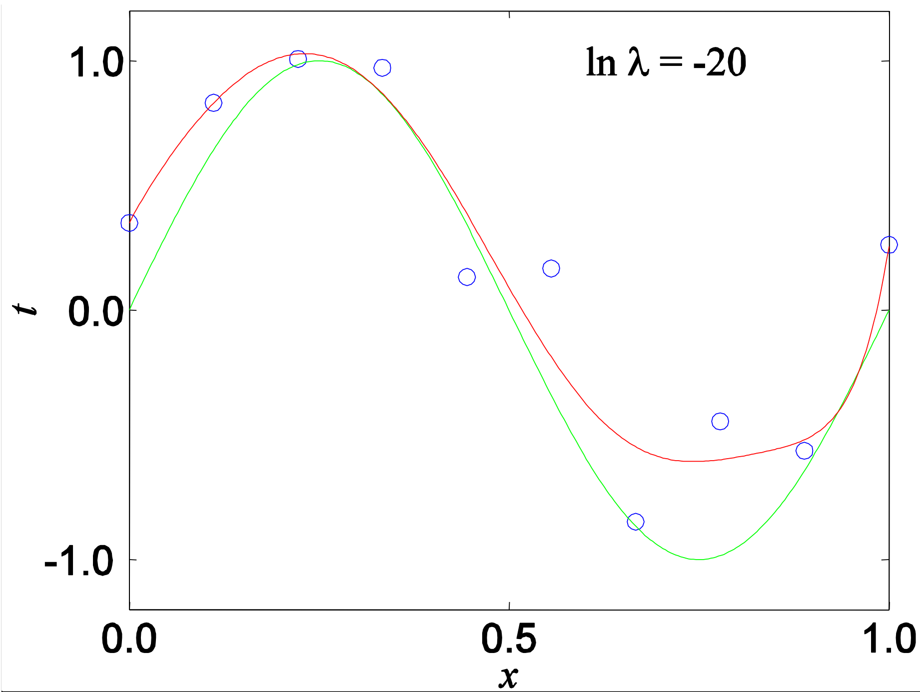
Ridge Regression Example

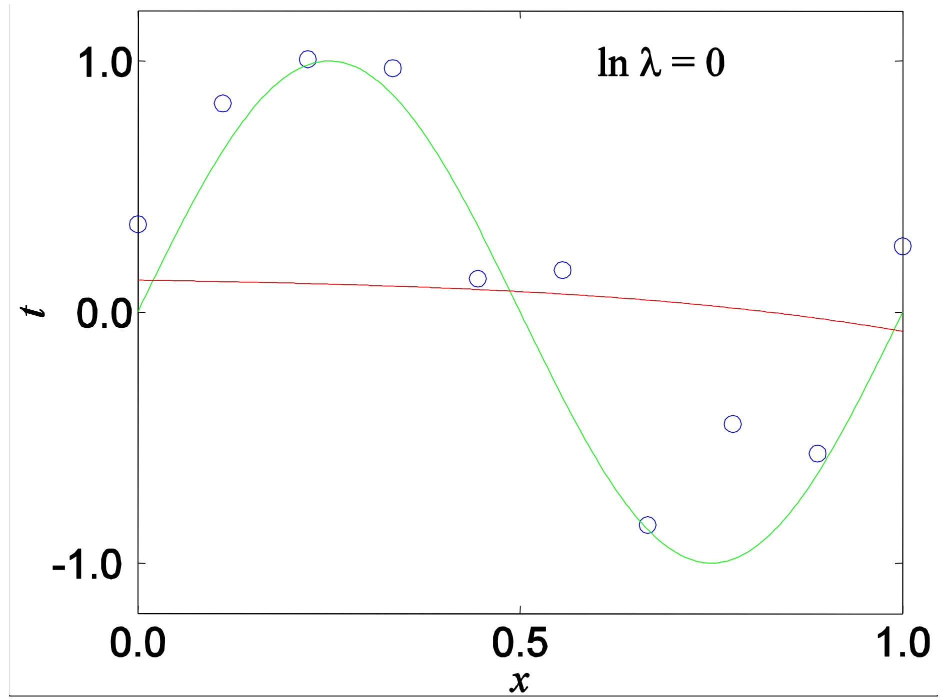
- Discourage large values by adding penalty term to error

$$E(\mathbf{w}) = \sum_{n=1}^{N} \{\mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

- Also called *shrinkage* (stats) or *weight decay (neural nets)*
- The regularization coefficient λ now controls the effective model complexity
- *Closed form solution: $\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$
  - Leads to numerical stability as well!

Regularized $M = 9$ Polynomial





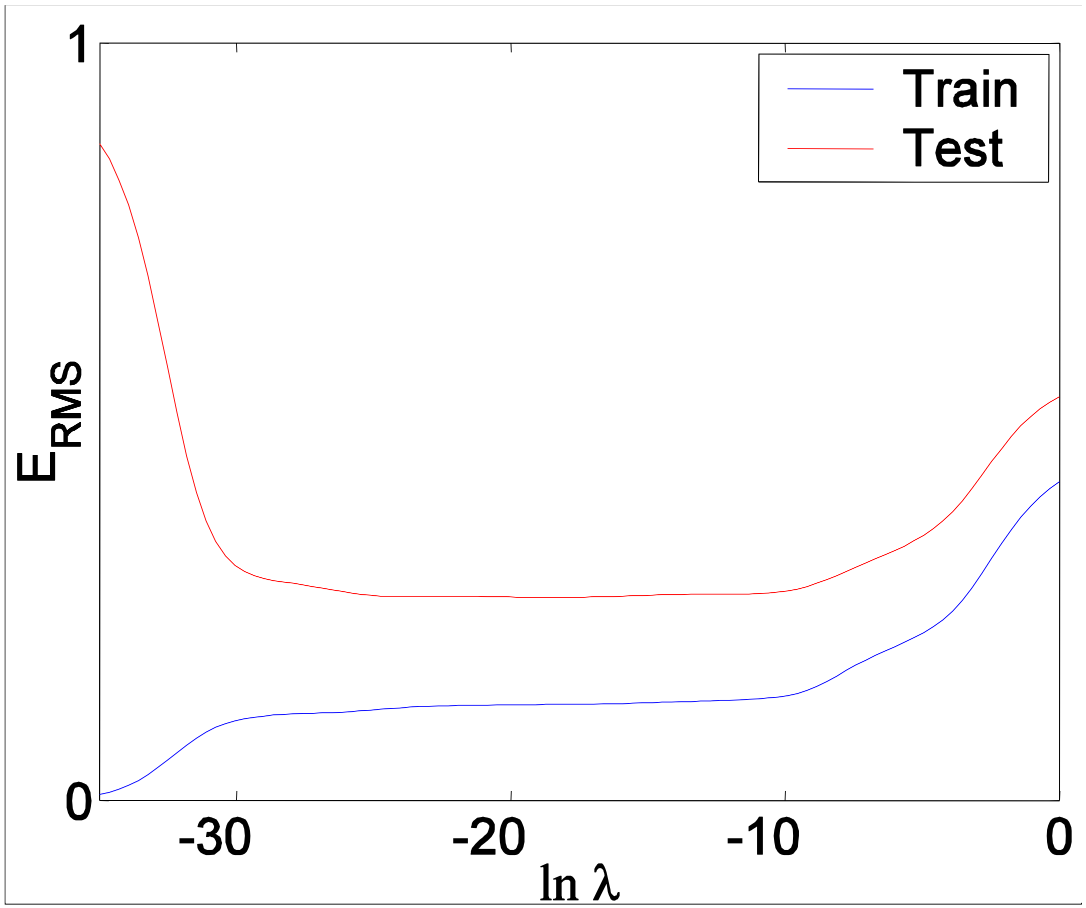
Regularized Parameters

- First col is the unregularized solution

	$\ln \lambda = -\infty$	$\ln \lambda = -20$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.1273
$w_1^\star$	232.37	5.56	-0.0459
$w_2^\star$	-5321.83	-12.27	-0.0578
$w_3^\star$	48568.31	19.01	-0.0460
$w_4^\star$	-231639.30	-82.58	-0.0321
$w_5^\star$	640042.26	46.49	-0.0201
$w_6^\star$	-1061800.52	141.84	-0.0104
$w_7^\star$	1042400.18	-29.57	-0.0028
$w_8^\star$	-557682.99	-231.55	0.0032
$w_9^\star$	125201.43	142.98	0.0080

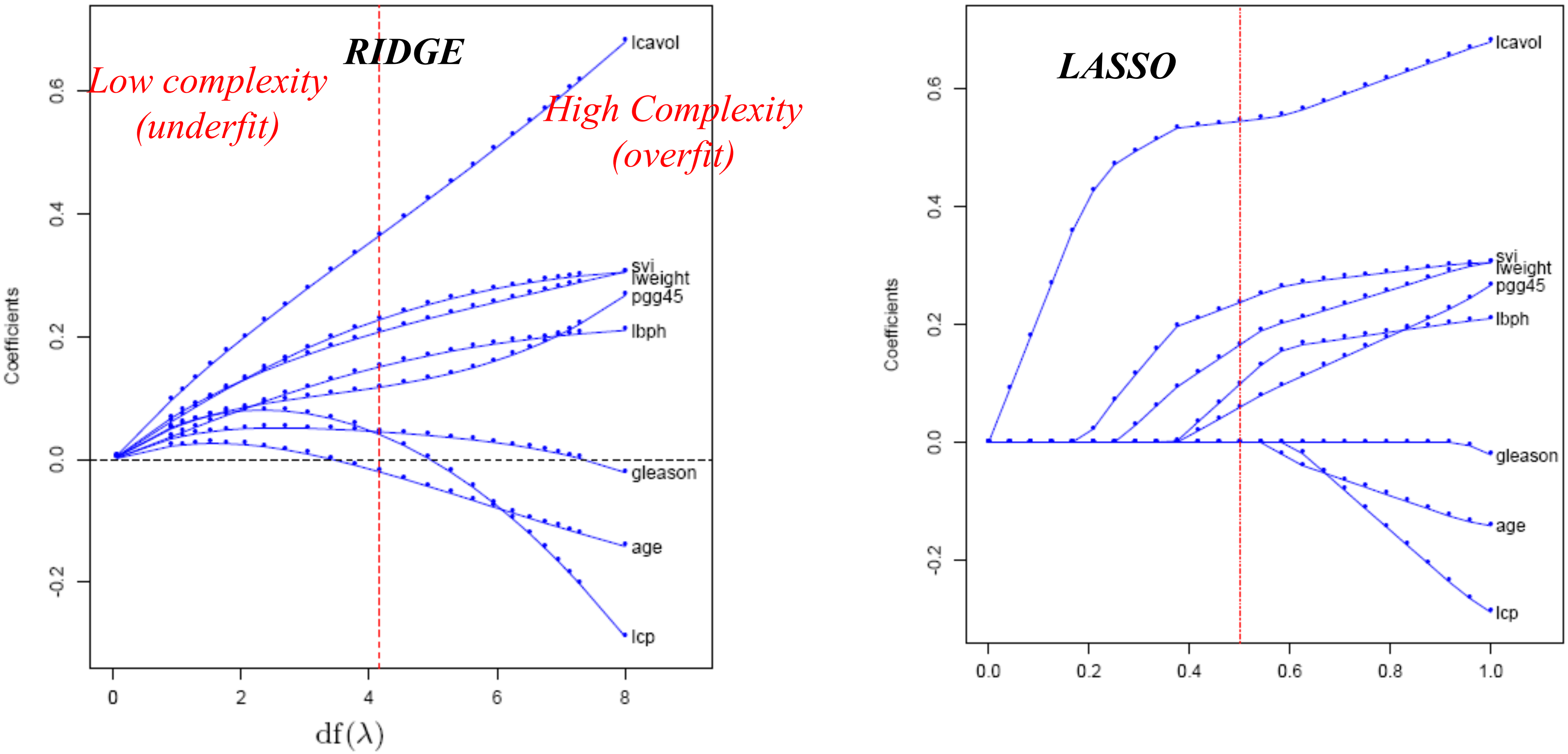
Generalization

- Noisy sine problem



Ridge vs. Lasso

- HTF figs 3.7, 3.9: Prostate Cancer example. Red line chosen by Cross-validation



Effect on values of coefficients as "effective degrees of freedom (DOF)" is increased for
(a) Ridge regression (left) and (b) Lasso (Right).
High λ translates to low DOF, so λ is being progressively decreased from left to right along the x-axis.

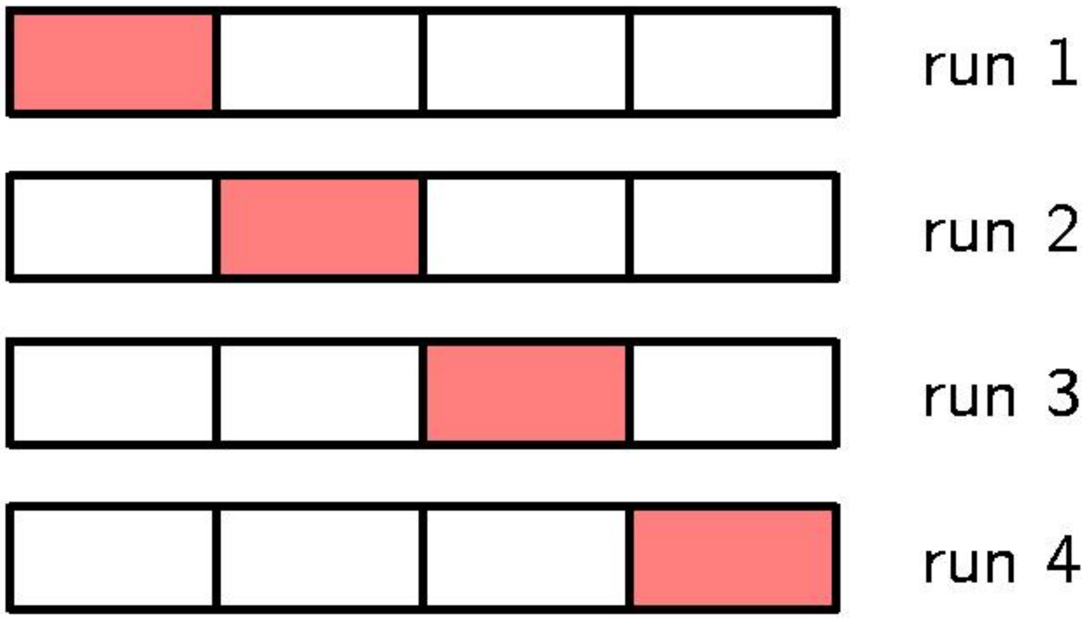
Evaluation

- Quality criterion for regression
  - Mean squared error (MSE) or equivalent, e.g. SSE, RMSE
    - true vs. empirical
    - normalized (R^2 value = % of variance explained)
    - Adjusted R^2

Estimating True Performance (Data Driven)

- enough data? Use "holdout" to estimate
- Moderately large? Use k-fold cross-validation
  - extreme case (small dataset) : Leave One Out (LOO)
-

K = 4 example



Bias-Variance Dilemma

Usually *measured* output is not a deterministic function of *given* inputs

Assume: t = h (x) + zero-mean noise

- your model gives y (x). The *expected squared loss*,

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

- best predictor: $\mathbb{E}$[t | x] = h(x) ;

  – MSE_{opt} = variance of the noise inherent in the random variable t.
  (2nd term on RHS)

- What does the first term comprise of ?

The Bias-Variance Decomposition

- Suppose we were given multiple data sets, each of size N. Any particular data set, D, will give a particular function y(x;D).
- For any x, The expected loss (over datasets of size N) is

$$
\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right] \\
&= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}.
\end{aligned}
$$

(try to express both terms in words)

The Bias-Variance Decomposition II

Considering all possible values of x, we can write

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

where

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

$$\text{noise} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

Bias: how good the average model is;
Variance: how sensitive the model is to variations in data.

NOTE: the bias and variance concepts here apply to a predictive model, rather than to an estimator of a specific value.

Bias-Variance Tradeoff

- Change model type? Affect bias
- More training data: decrease variance
  - "consistent estimators" converge to ideal solution as |D|→ infinity
  - For small data sets, lower complexity models may be preferred.
- **Ideal solution: suitable model type & complexity**

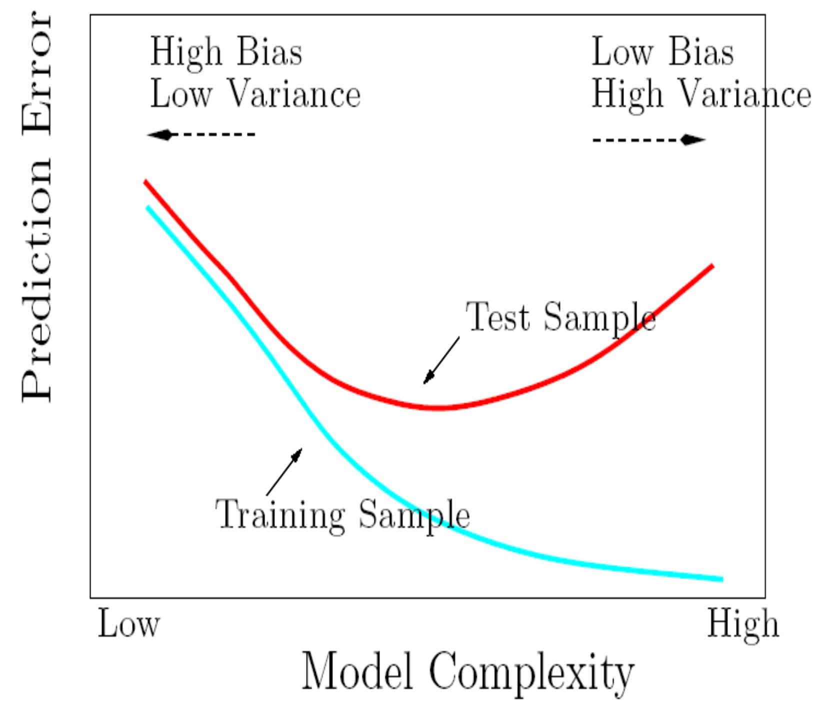
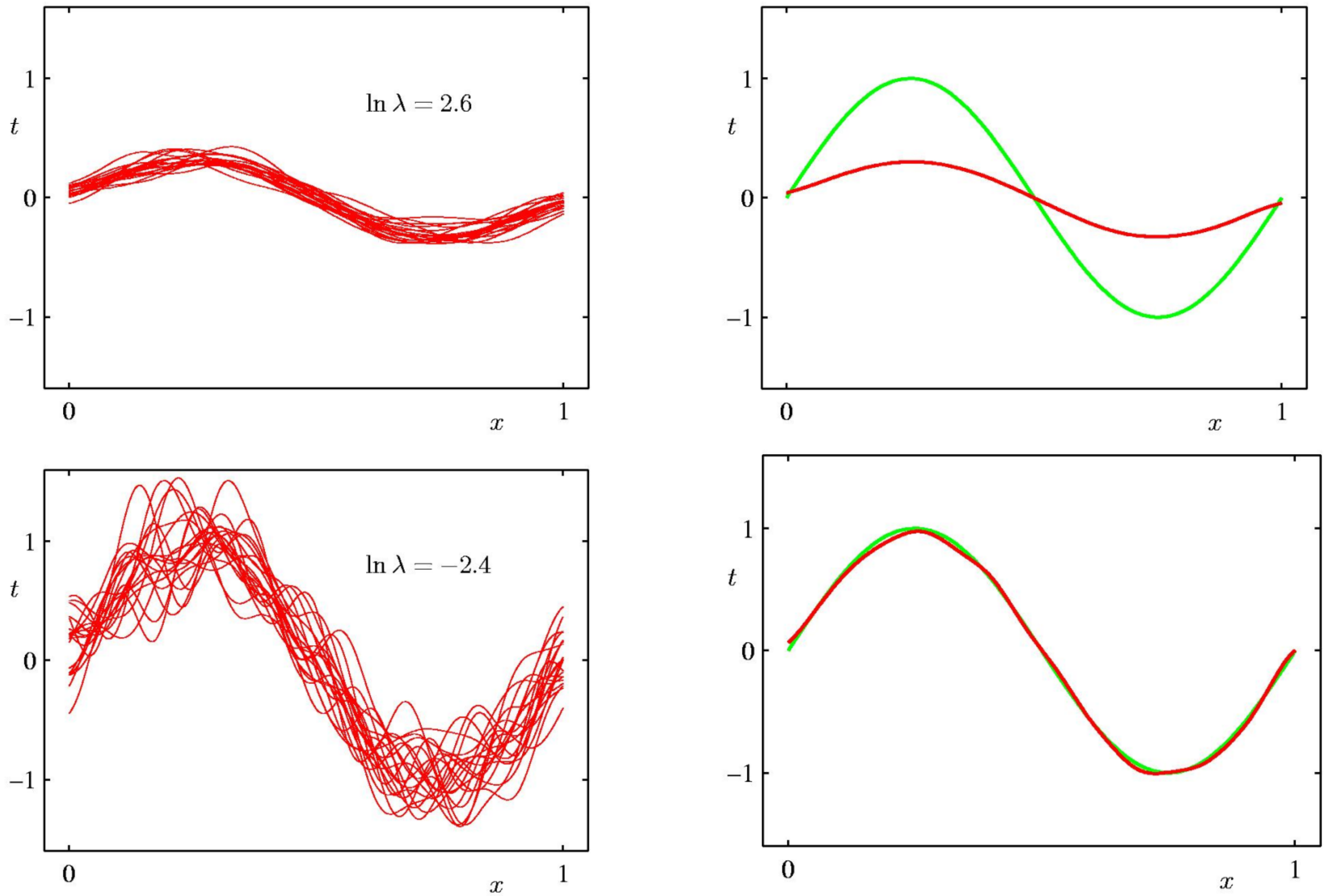


Figure 2.11: *Test and training error as a function of model complexity.*

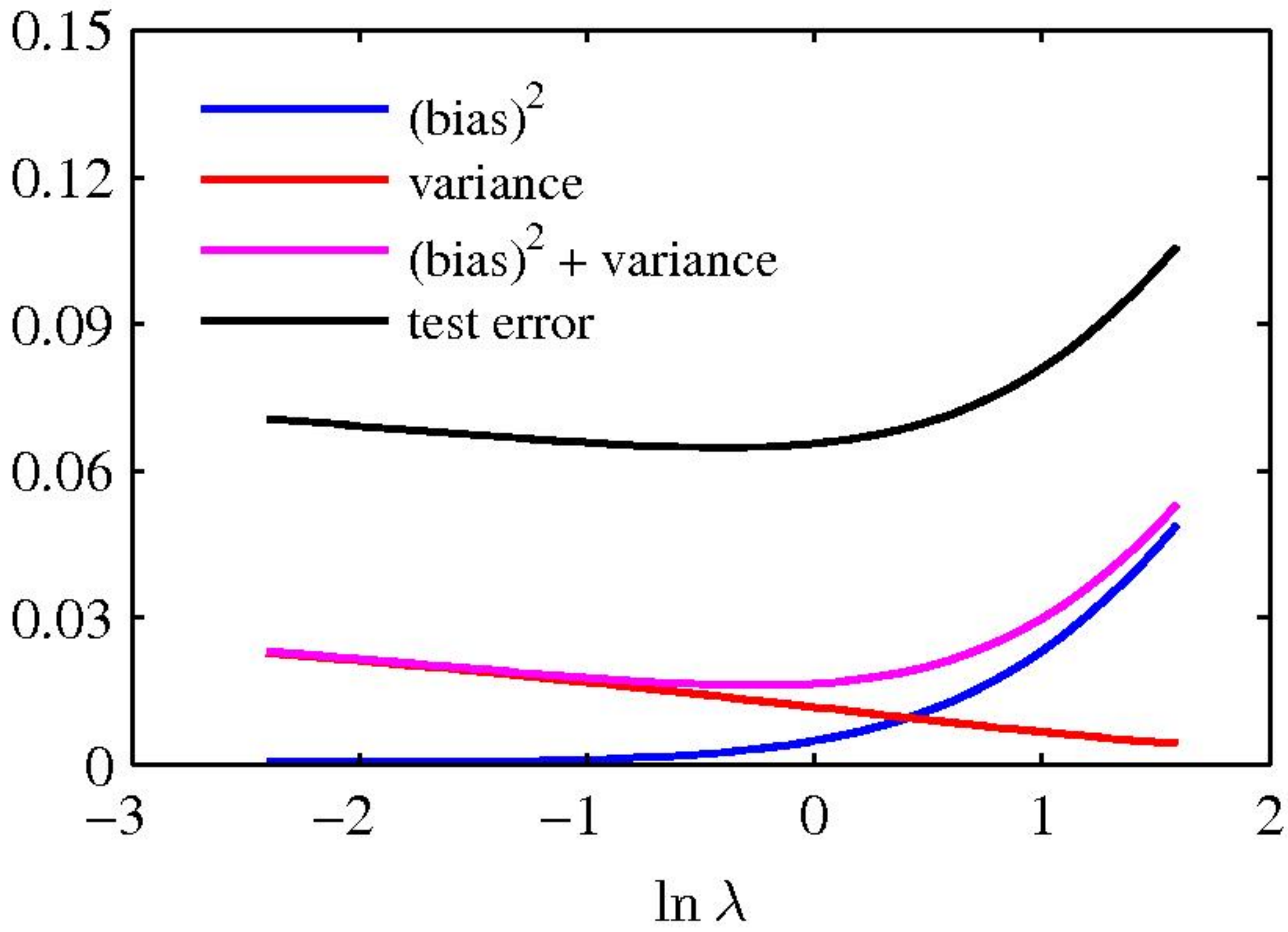
Effect of Regularization on Bias-Variance

- Bishop 06, fig 3.5. Model is sum of 24 gaussians, with ridge regression



Bias-Variance vs. Regularization Amount

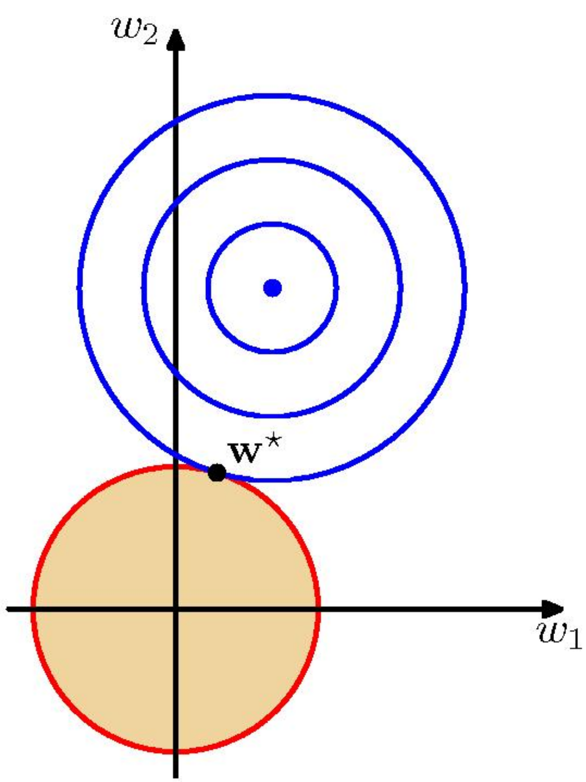
- *What happens to the curves as amount of training data increases?*

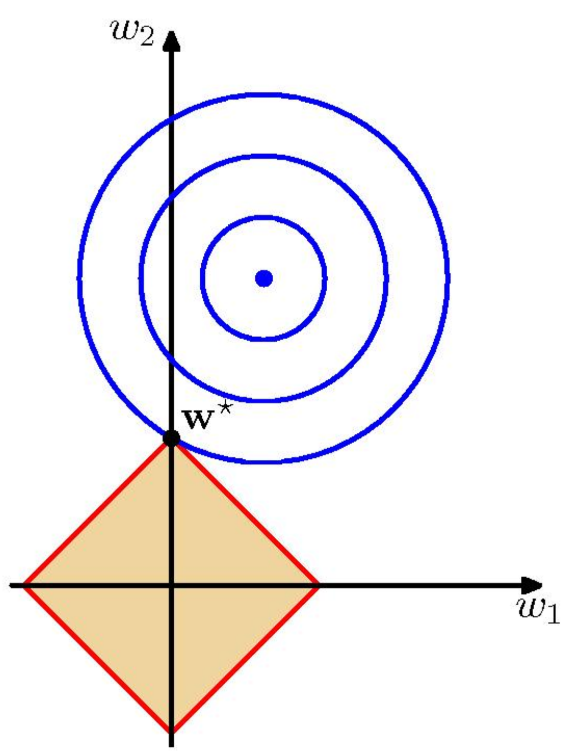


Extras

Comparing Shrinkage Methods B06: fig 3.4

- ridge regression (Regularization Penalty = sum squared of weights)
  vs
- **Lasso ((Regularization Penalty = sum of |w|)**
  red: constant penalty contour; blue: unregularized error contours





Estimating True Performance (Formula Driven)*

- true mean squared error (MSE = SSE/N) = empirical error + complexity term
  - complexity term = f (model type, # of parameters, # of training points)
    - e.g. linear regression with N samples, P parameters
      Akaike's Final Prediction error = $MSE_{empirical} (N+P) / (N - P)$
    - for nonlinear models, find "effective number of parameters" and plug into linear formulae

Takeaway: Formula Driven Estimates of True Performance specialized for linear models. Not so relevant in data mining context

Least Angle Regression (LAR; HTF 3.4.4)*

- Takeaway: Efficient procedure for fitting an entire lasso sequence with the cost of a single least squares fit.
- R code: lar

Algorithm 3.2 *Least Angle Regression.*

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.
2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.
3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.
4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor $\mathbf{x}_l$ has as much correlation with the current residual.
5. Continue in this way until all p predictors have been entered. After $\min(N-1, p)$ steps, we arrive at the full least-squares solution.

Group Lasso for Sparse Learning*

- SLEP package
  http://www.public.asu.edu/~jye02/Software/SLEP/overview.htm
- ℓ_1-Regularized (Constrained) Sparse Learning
- ℓ_1/ℓ_q-Regularized Sparse Learning (q>1)
- Fused Lasso
- Sparse Inverse Covariance Estimation
- Sparse Group Lasso
- Tree Structured Group Lasso
- Overlapping Group Lasso
- *Takeaway: A variety of methods exist to shrink parameters in different ways*