
Intro to Key Themes in Predictive Modeling

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 ".
(Also see *EA*:4.6-4.8, *KM*: 1.2.2)

Outline/Objectives

- True Performance (Generalization)
- Overfitting /Underfitting
 - Bias – Variance Tradeoff

All of the above inform Effective Model Choice and Complexity

- Regularization (Ridge and Lasso)

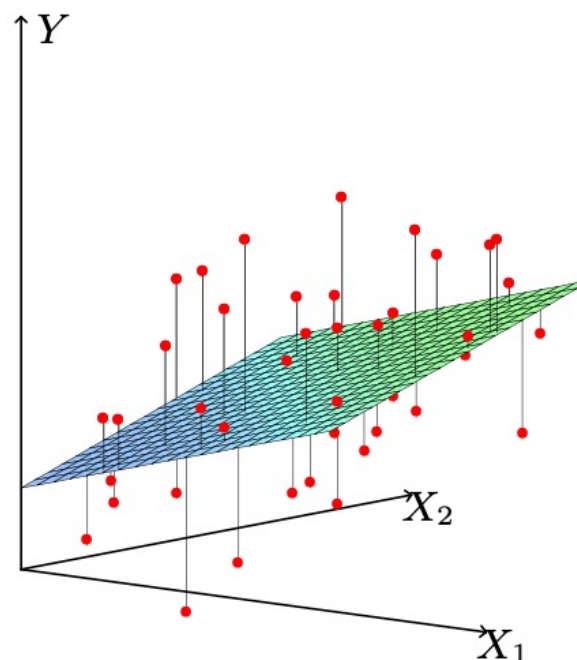
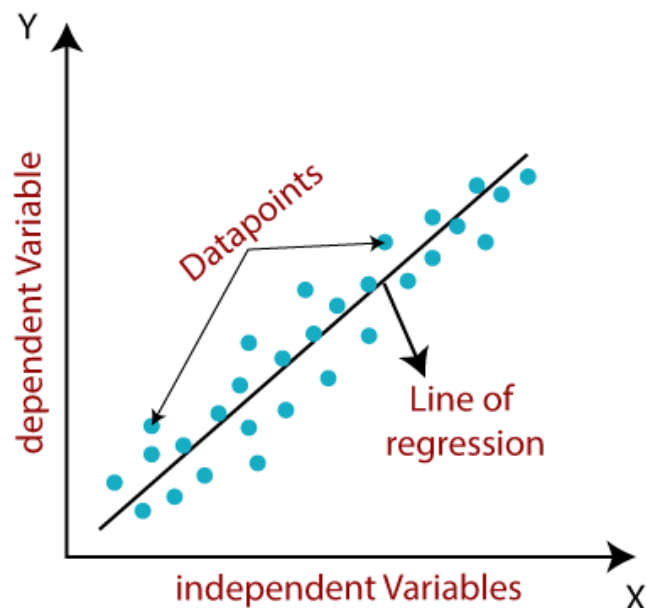
Concepts carry over to classification problems.

Remember, why we're studying linear regression

- When you're fundraising, it's AI
- When you're hiring, it's ML
- When you're implementing, it's **linear regression**
- When you're debugging, it's `printf()`

credit: internet

Multiple Linear Regression (MLR): Stats notation



$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

Coefficients are called "beta"s, the dependent variable is Y, and estimates are indicated by "hats".

Learning by Ordinary Least Squares (OLS): Hyperplane is fitted by minimizing the mean square error (MSE)

MLR: Assumptions

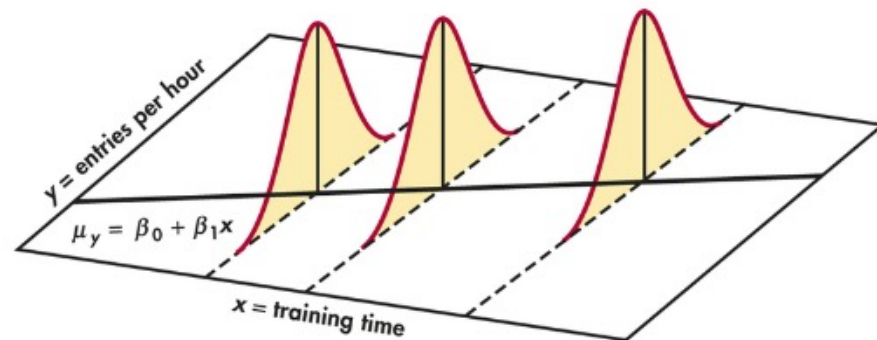
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \epsilon \quad (1)$$

OR

$$Y|X_1 \dots X_p \sim N(\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p, \sigma^2) \quad (2)$$

Modeling Assumptions:

- (i) The conditional mean of Y (dependent variable) is **linear** in the independent variables.
- (ii) The error term (deviations from the linear fit)
 - are normally distributed
 - independent from each other
 - identically distributed (i.e., they have constant variance)



Then minimizing Mean Squared Error (MSE) on the training data yields the Maximum Likelihood Estimate (MLE) solution of the assumed **generative model**.

(see Bishop pg. 140 for the math)

Interpretation; Collinearity

- How do you interpret the betas?

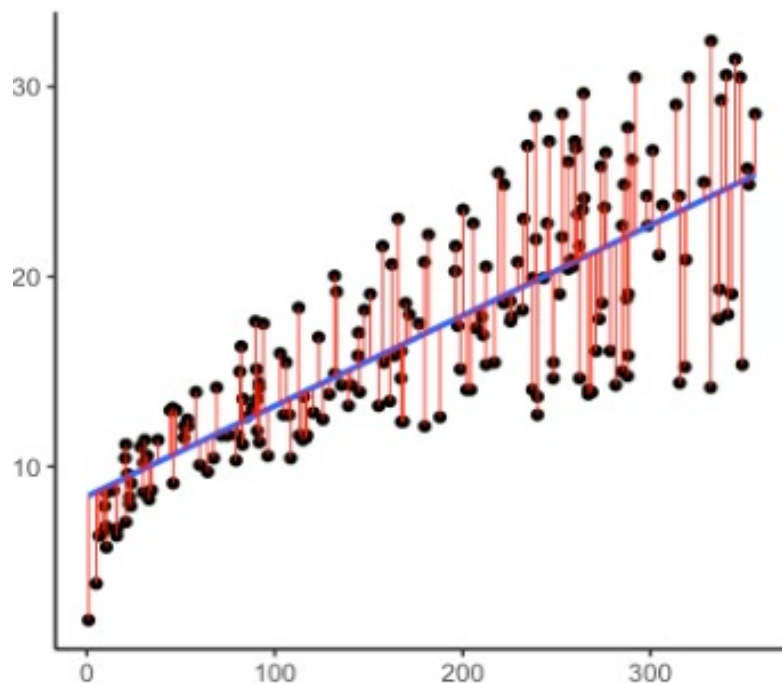
- $$\beta_j = \frac{\partial E[Y|X_1, \dots, X_p]}{\partial X_j}$$

- Problems with Collinearity
 - Dependencies between the X's **affect our interpretation** of a multiple regression.
 - Inflates the uncertainty in the estimates of the coefficients
 - But not the value of MSE

MLR Diagnostics

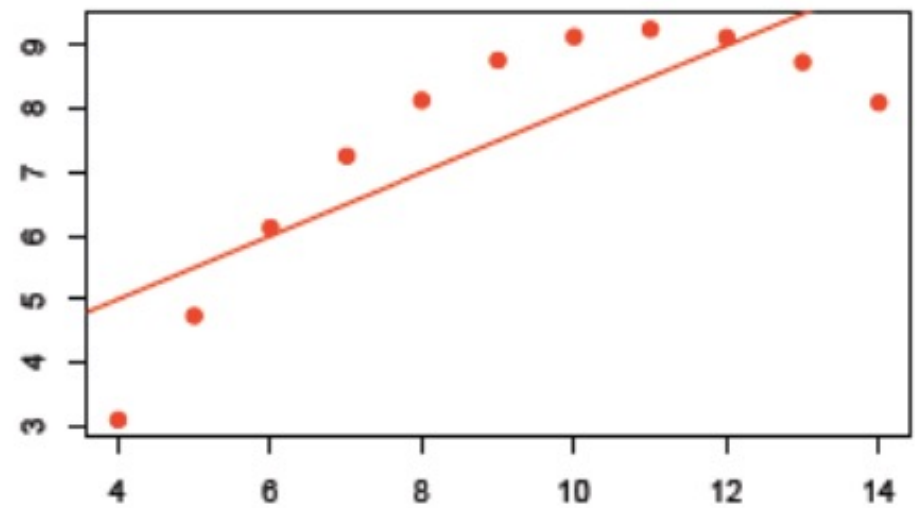
- Look at residuals!
 - (*Test for Gaussianity: look up [Q-Q plots](#))

Heteroscedasticity: Variance not constant



variables, adding extra variables, etc, may resolve some issues

Linear relation suspect



So then, What is X?

- How many X's do you have and what are they?
 - | Bank Example: dummy coding and interaction effects
- What Happens if there are Outliers in X or in Y?

A Generalized view of MLR

(following Bishop Ch 3.1, 3.2)

and switching to ML notation

parameters are w 's (for weights)

dependent variable is “ t ” for target,

and model produces output “ y ”.

MLRs are Parametric Models

Learning a Parametric Model:

1. Determine **functional form** of model (e.g. polynomials, neural nets,...
2. “learn” the parameters (weights) of the model using the training data.

- **Generalized linear regression:** 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 regression: (with scalar x)

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

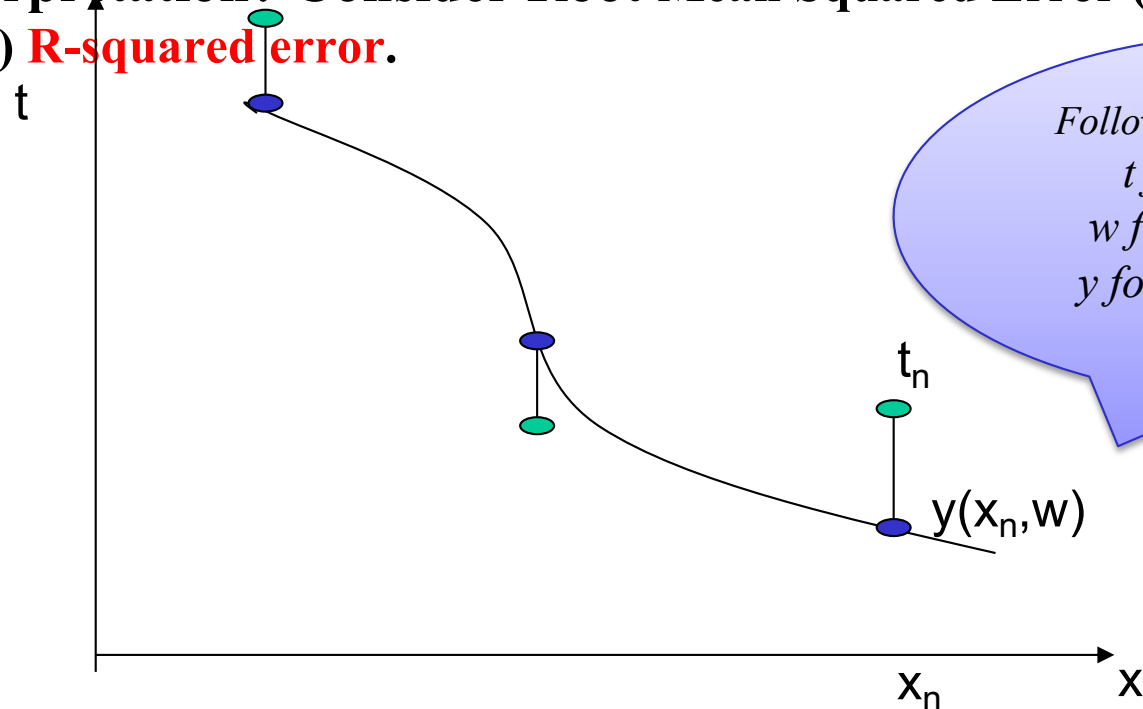
i.e., the basis functions are given by $\phi_i(x) = x^i$

Ordinary Least Squares (OLS)

- Minimize a **loss function** $E(\mathbf{w})$ given by sum-of-squares **error (SSE)** (t 's are the target values)

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

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



(Batch Mode) Least Squares Solution*

- Exact **closed-form** minimizer (ML solution)

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

where $\vec{t} = (t_1, \dots, t_N)^T$

– “Pseudo-inverse solution”

and Φ is the *design matrix* given by

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

Takeaways:

- Direction solution involves inversion of an $(M+1) \times (M+1)$ matrix*
- Computation Linear in data set size, cubic in M*
- Batch mode training*
- Explicitly shows collinearity problem*

Collinearity problem: parameter estimates have high uncertainty if two or more independent variables are highly collinear

Why OLS?

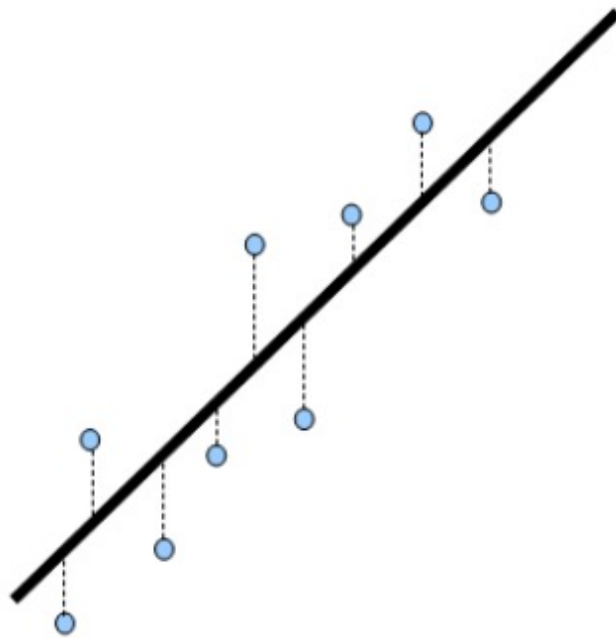
- Minimizing Mean Squared Error (MSE) on the training data yields the Maximum Likelihood Estimate (MLE) solution of the following (assumed) model:
 - Expected value of T given the basis function vector ϕ is linear in ϕ .
 - i.e. Conditional mean is linear in the predictors.
 - All distributions around the expected values are assumed to be i.i.d. zero mean Gaussian with constant variance.
 - In stats notation: $Y|X_1 \dots X_p \sim N(\beta_0 + \beta_1 X_1 \dots + \beta_p X_p, \sigma^2)$
 - In new notation: (fill in)

(For Proof see **Bishop pg. 140**)

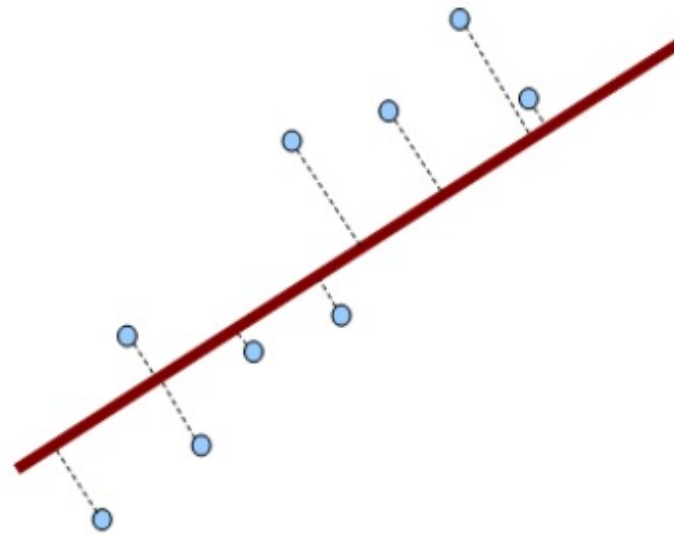
Visualize: <http://setosa.io/ev/ordinary-least-squares-regression/>

How can you “verify” that the assumptions seem reasonable?

Total Least Squares (Aside)



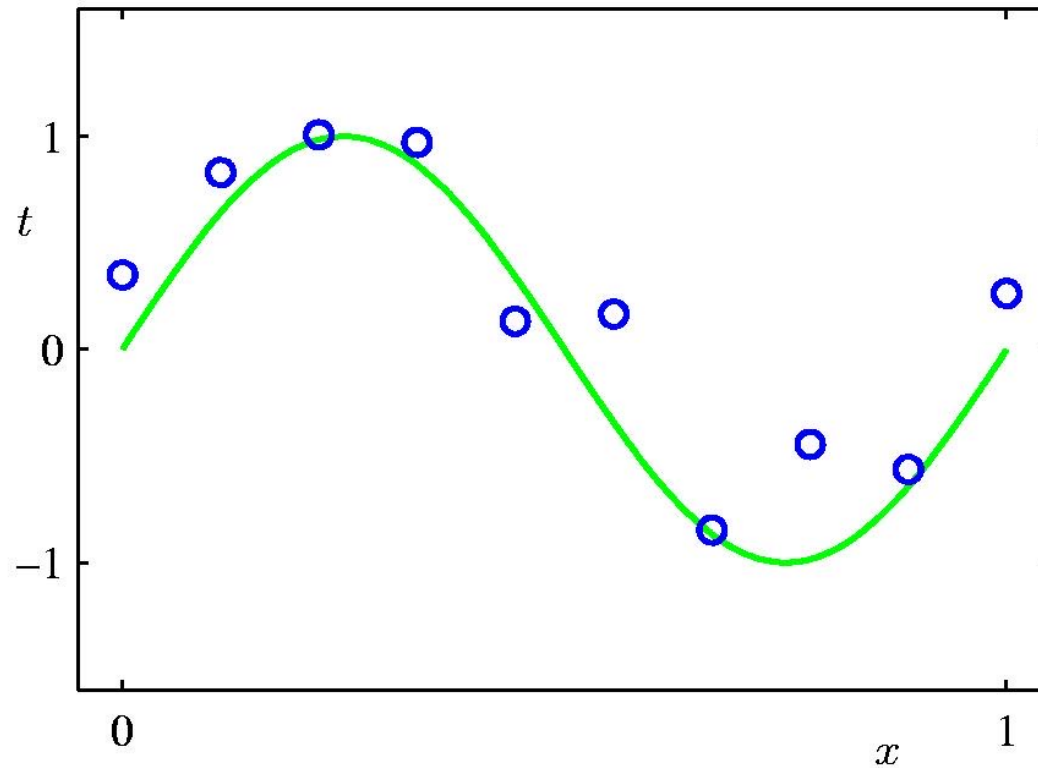
Linear least squares



Total least squares

Which one is better?
Which one should you choose?

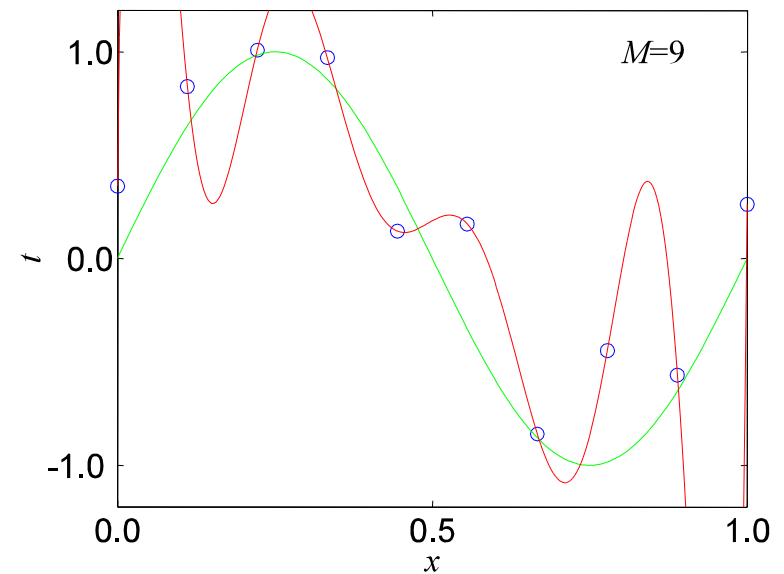
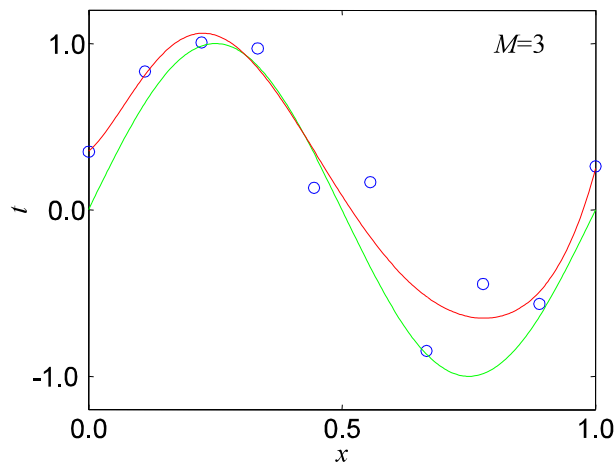
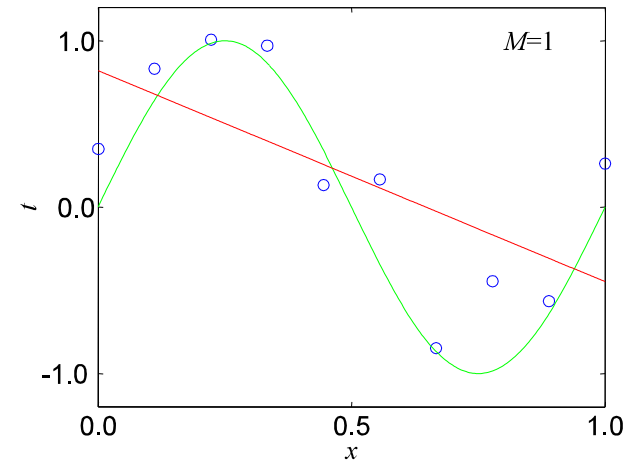
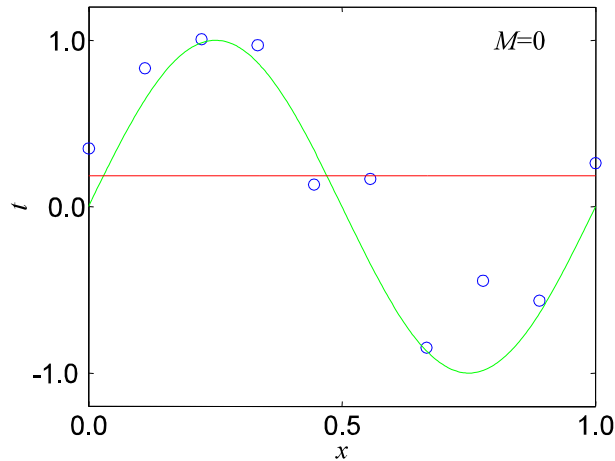
Polynomial Curve Fitting



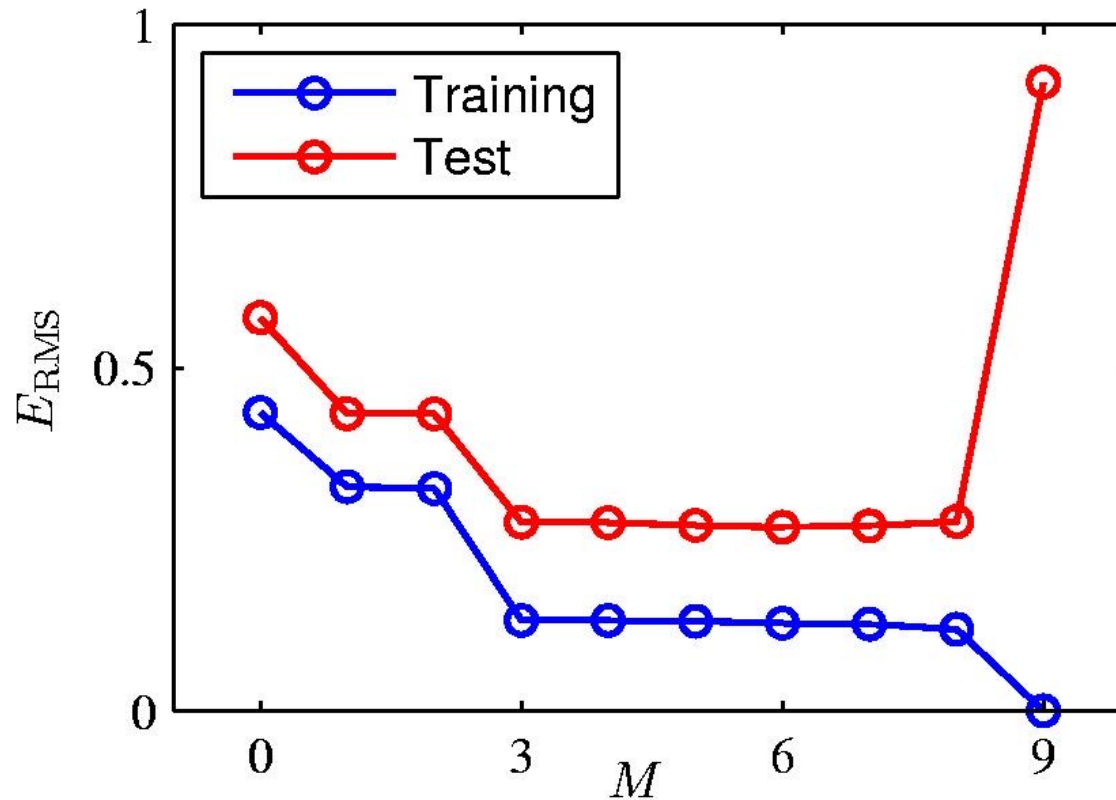
$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

Model Complexity and Overfitting

- “Noisy sine” example from Chris Bishop

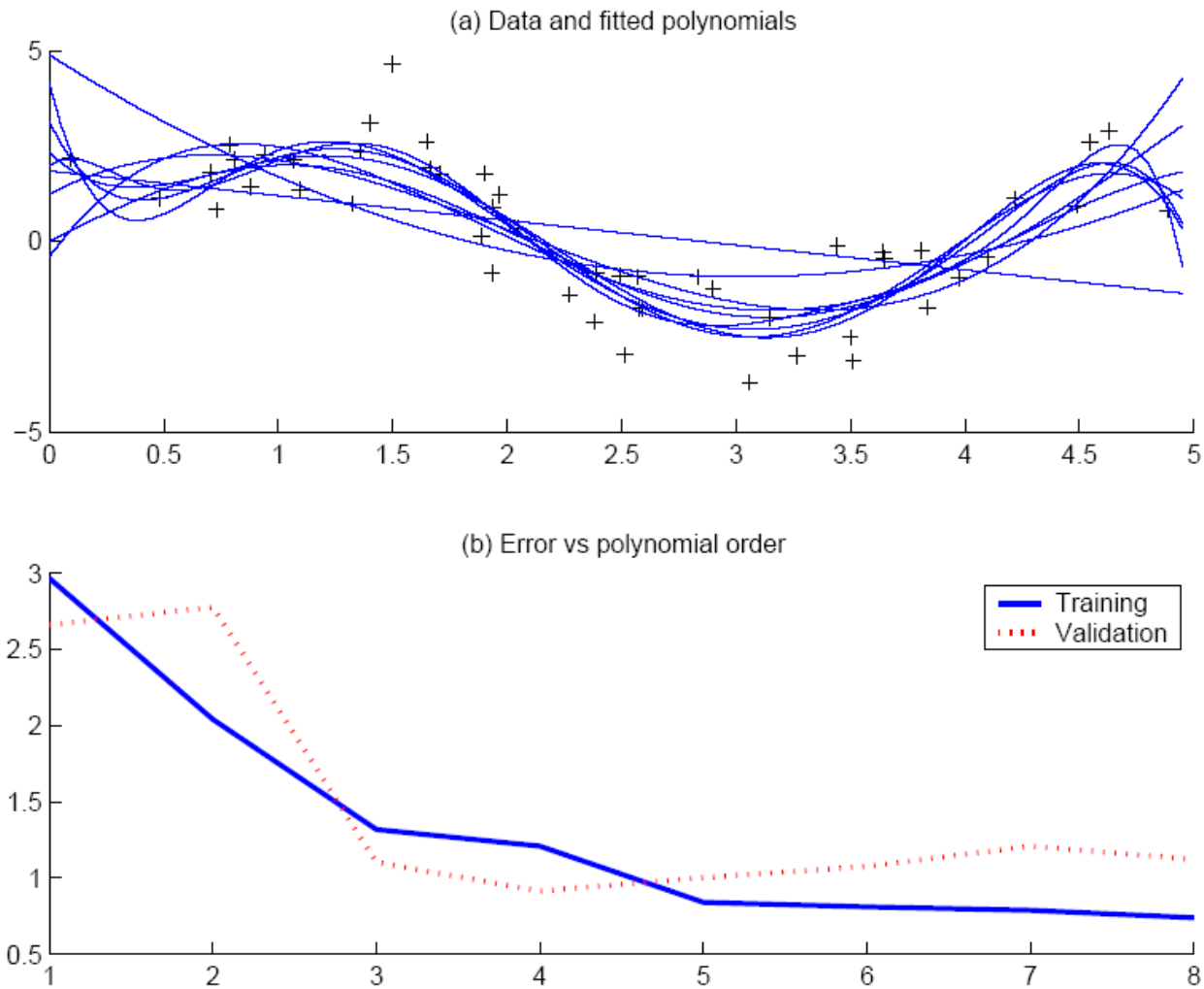


Over-fitting



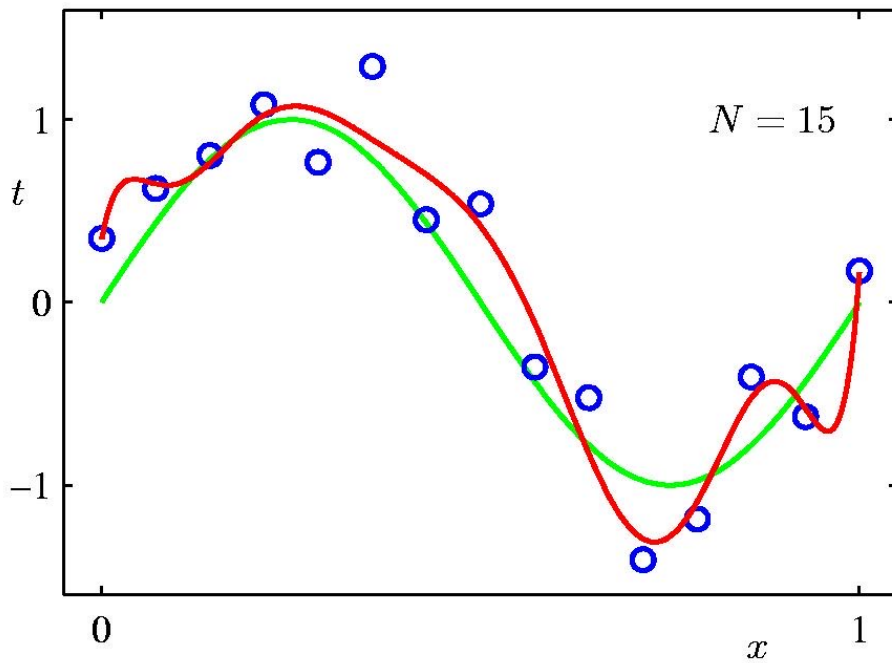
Root-Mean-Square (RMS) Error vs Polynomial order

Another Example

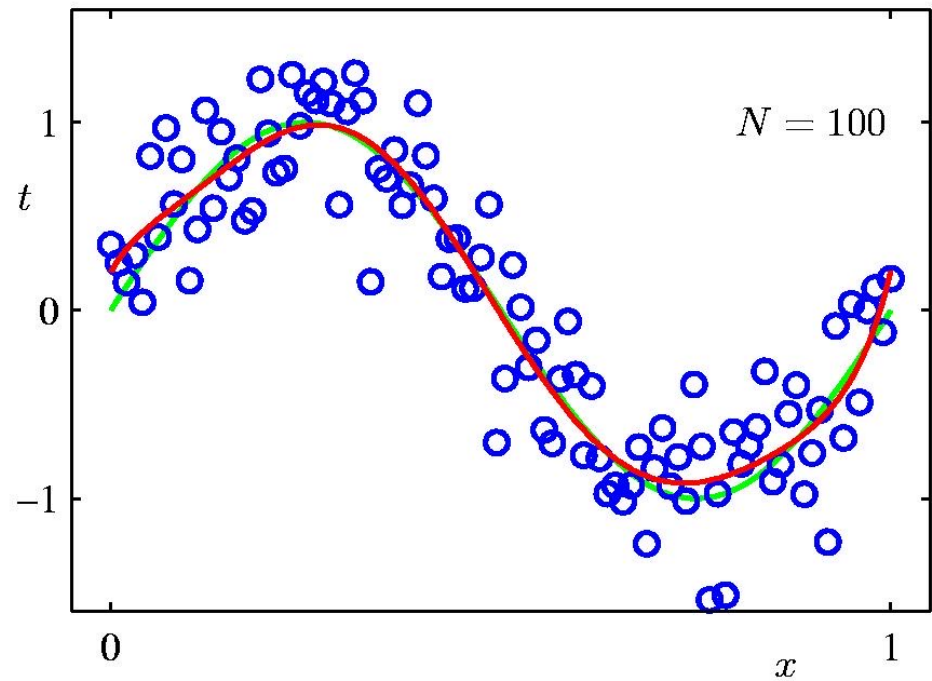


Effect of Data Set Size

9th Order Polynomial



$N = 15$



$N = 100$

Regularization (to avoid overfitting)

- “regularization term” imposes penalty on less desirable solutions
 - $\text{Cost} = \text{MSE} + \lambda \text{ Penalty}(f)$
 - Scikit uses “alpha” to denote lambda.
 - 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:** 1. “intercept”, i.e. w_0 , not included in penalties
2. customary to standardize all independent variables first (why?)

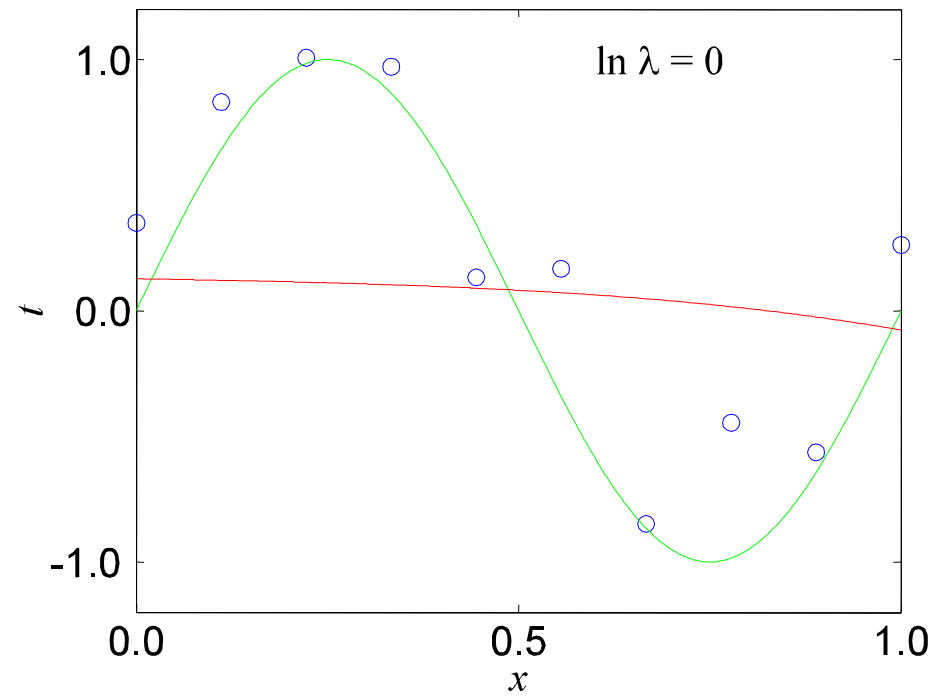
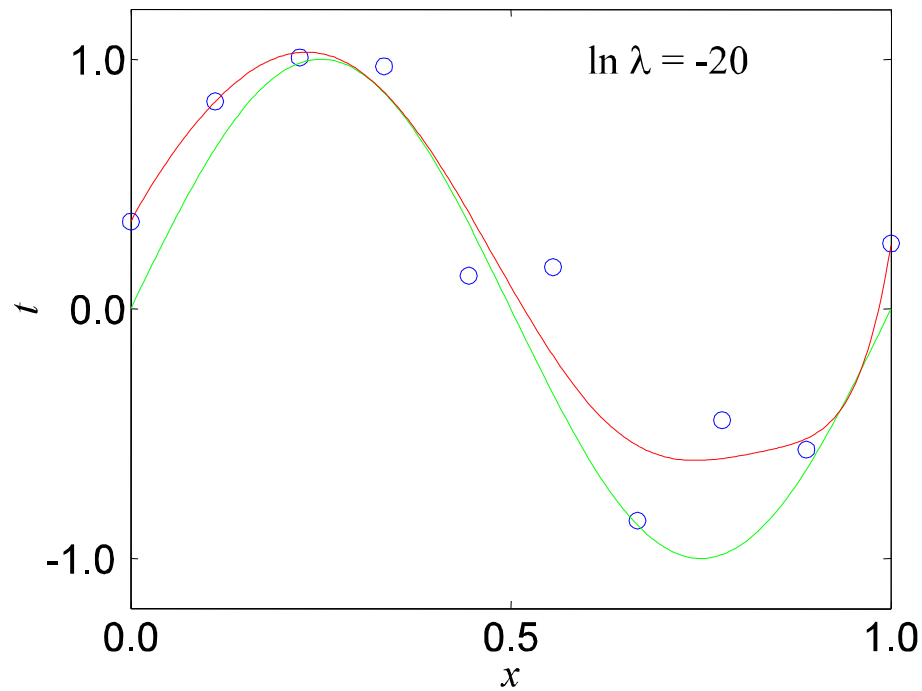
Ridge Regression Example

- Discourage large values by adding penalty term to error

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

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

Regularized $M = 9$ Polynomial



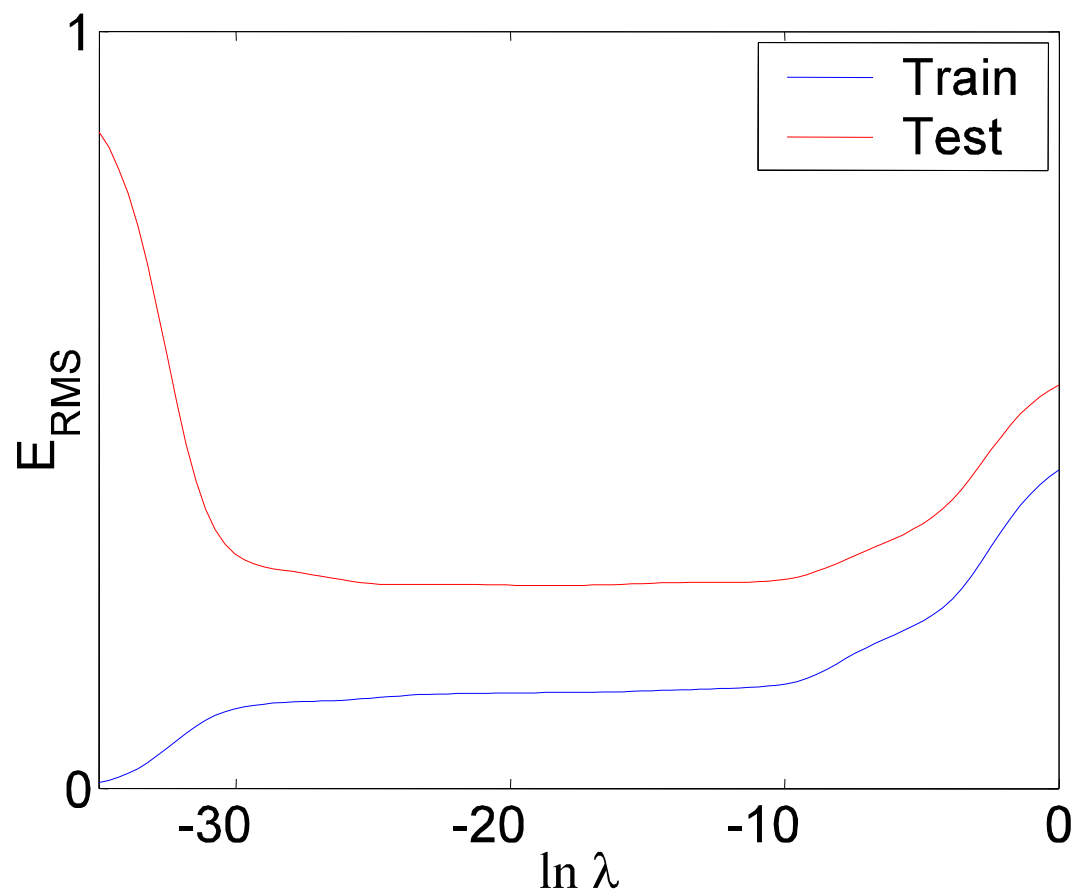
Regularized Parameters

- First col is the unregularized solution

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

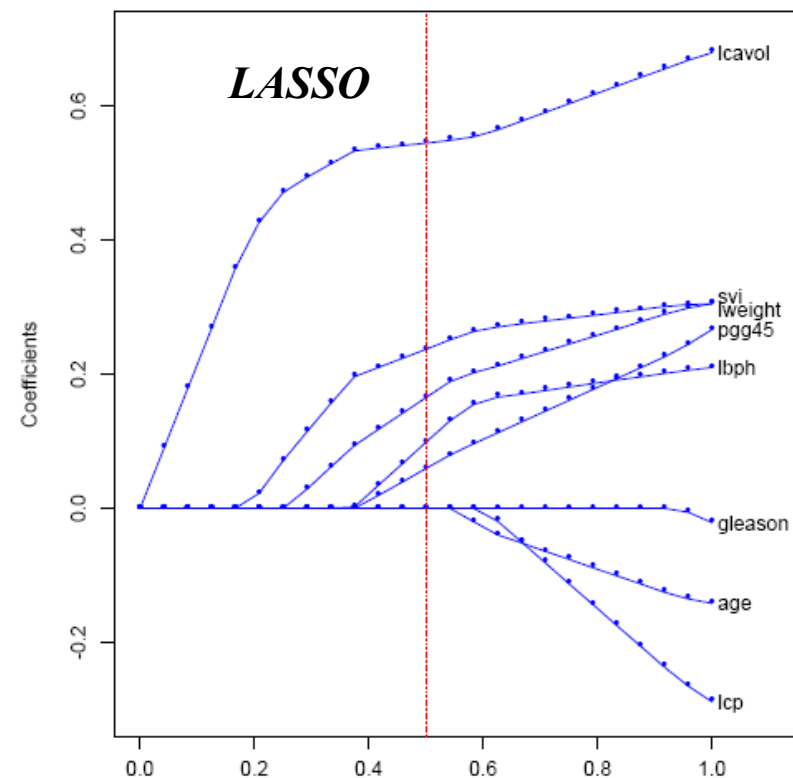
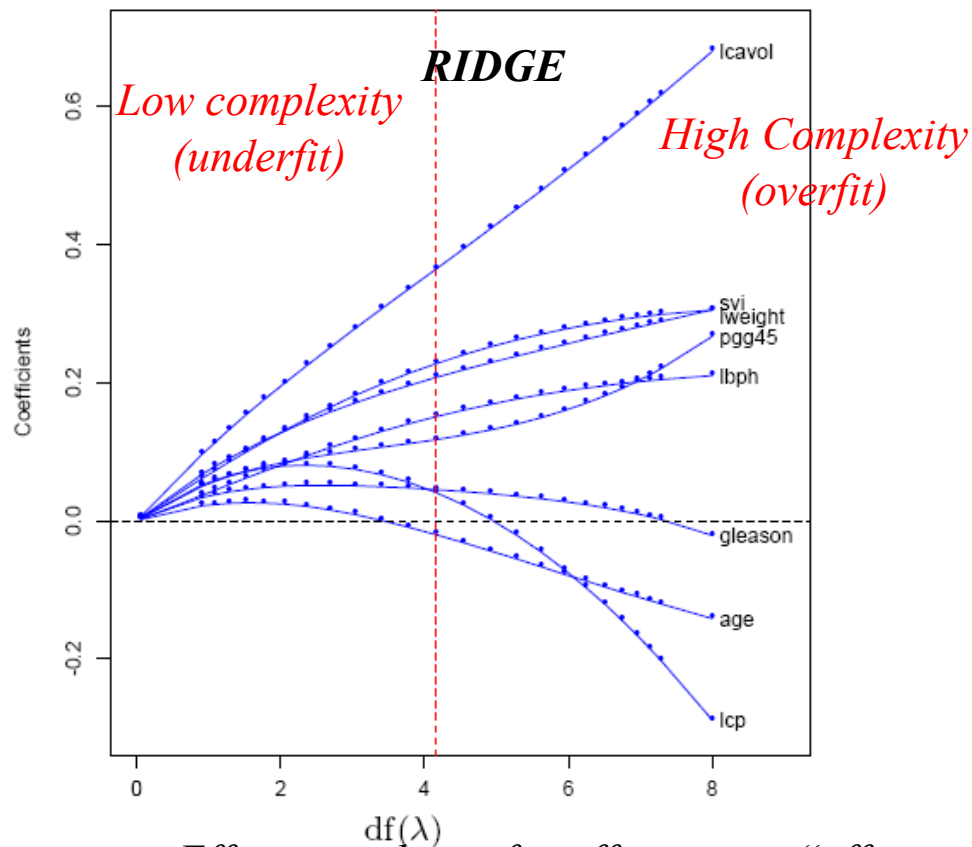
Generalization

- 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 (df)” is increased for
(a) Ridge regression (left) and (b) Lasso (Right).

High λ translates to low df, so λ is being progressively decreased from left to right along the x-axis.

Evaluation

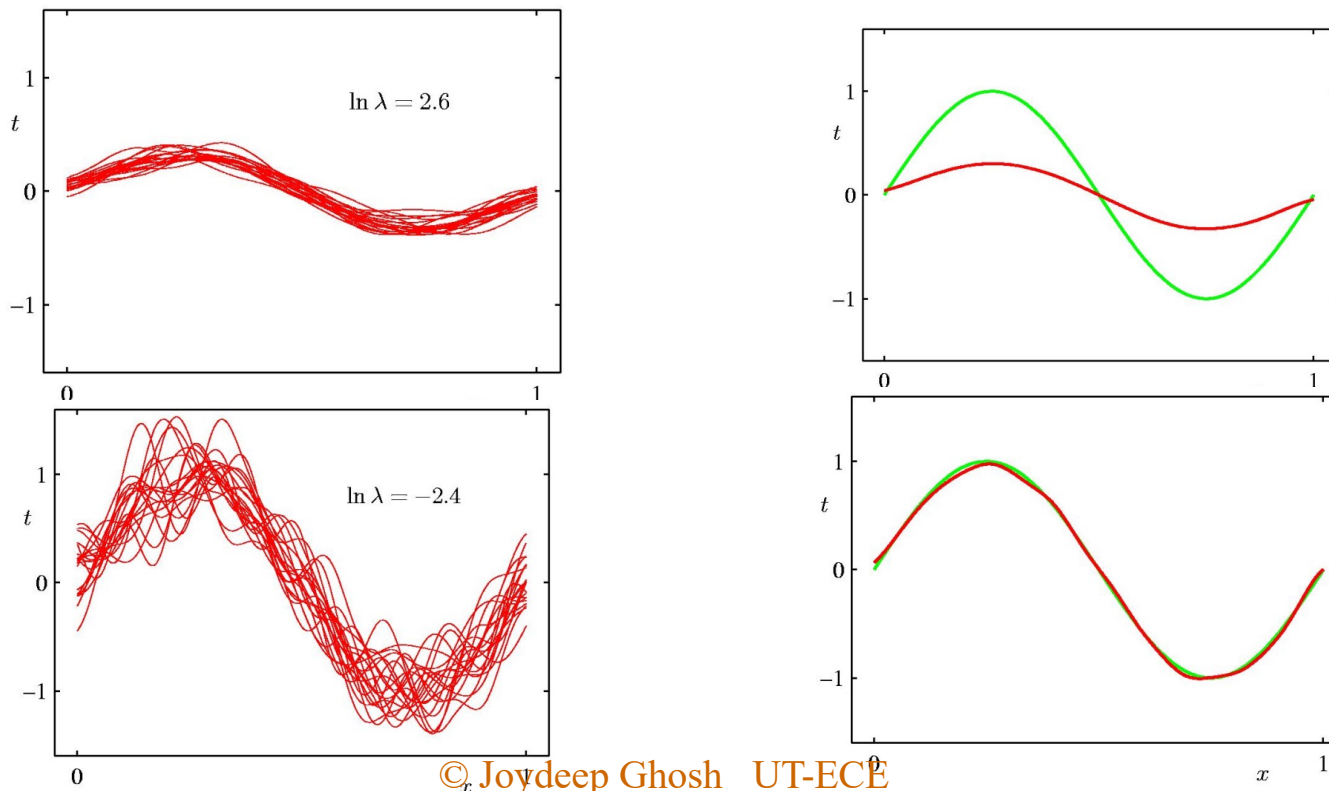
- Quality criterion for regression
 - Mean squared error (MSE) or equivalent, e.g. SSE, RMSE
 - true vs. empirical
 - normalized (R^2 value = % of variance explained)
 - Adjusted R^2

How to Evaluate a Regression/Prediction Model?

- Typical Kaggle Challenge:
 - Cost function is given (e.g. MSE)
 - Evaluation is done on a specific test dataset
- What is the issue with this setup?

How to evaluate a Regression Model?

- Model = math form + learning method.
 - Will be impacted by given training (and validation) dataset!!
 - Want to evaluate the model irrespective of the specific train/validate dataset used.
 - Need to consider the collection of solutions obtained, not one specific solution.
- Bishop fig 3.5. Compares highly regularized solution (top row) vs. less regularized solution (bottom row). Individual solutions (left), averaged solution (right)



Bias-Variance Dilemma

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

Assume: $t = h(\mathbf{x}) + \text{zero-mean noise}$

- your model gives $y(\mathbf{x})$. The *expected squared loss*,

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

- **best predictor**: $\mathbb{E}[t | \mathbf{x}] = h(\mathbf{x})$;
 - $\text{MSE}_{\text{opt}} = \text{variance of the noise inherent in the random variable } t.$
(2nd term on RHS)
- **What does the first term comprise of ?** (Model_bias)² + Model_variance
 - Bias: how good the average model is;
 - Variance: how sensitive the model is to variations in data.
 - Tradeoff between the two terms as function of model complexity

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

Math Details: The Bias-Variance Decomposition*

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

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

(try to express both terms in words)

The Bias-Variance Decomposition II*

Considering all possible values of \mathbf{x} , we can write

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

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

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

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

Bias: how good the averaged 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.

Extra: Understanding the Bias-Variance Tradeoff

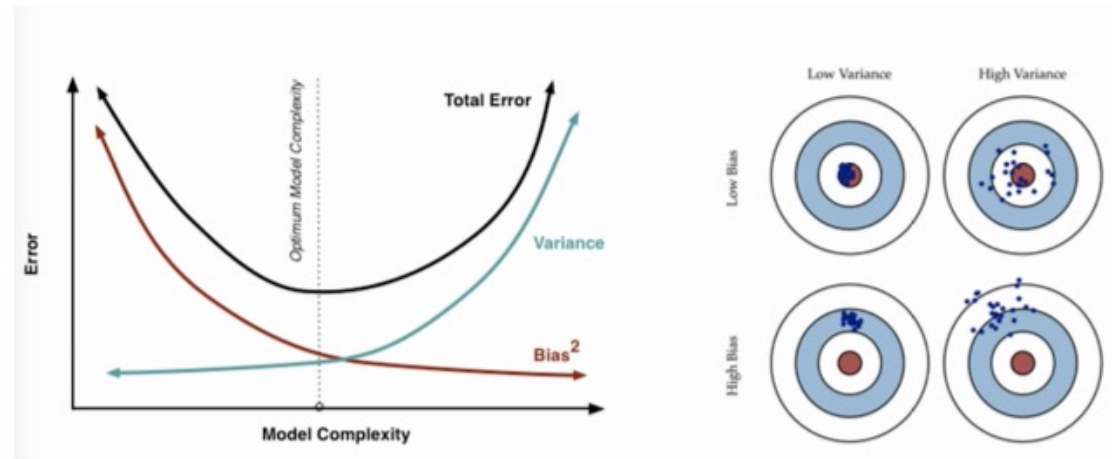


Figure 4.26: Cartoon illustration of the bias variance tradeoff. From <http://scott.fortmann-roe.com/docs/BiasVariance.html>. Used with kind permission of Scott Fortmann-Roe.

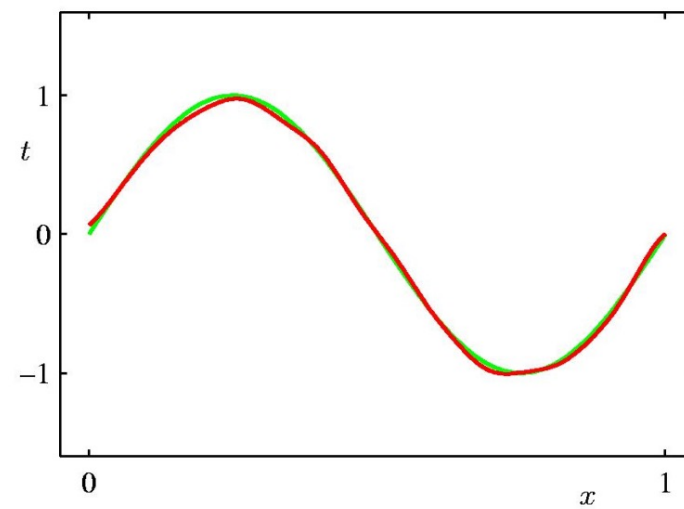
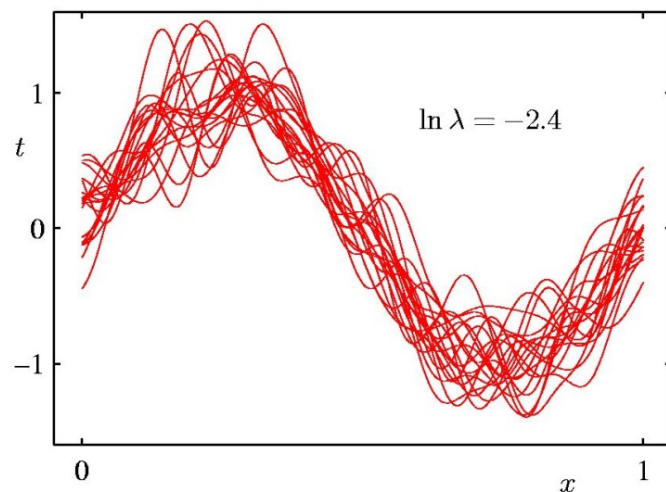
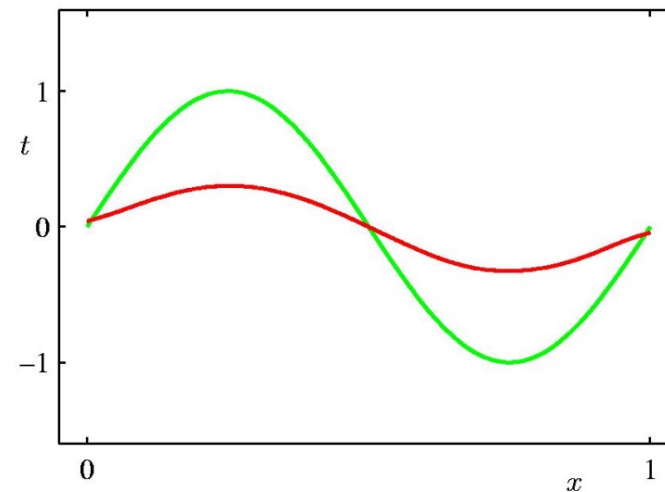
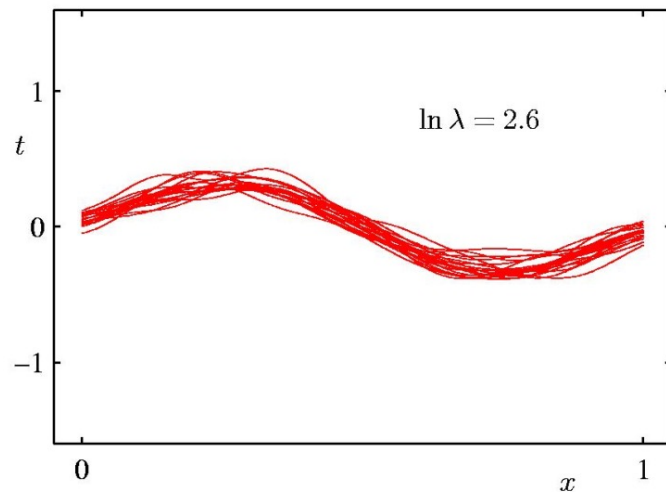
See [a simple video by Andrew Ng](#)

Including how to use the bias-variance tradeoff to diagnose issues with your model, e.g. how much will more data help, etc.

Effect of Regularization on Bias-Variance:

More regularization: increases bias, decreases variance

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



Example from KM, pg. 161

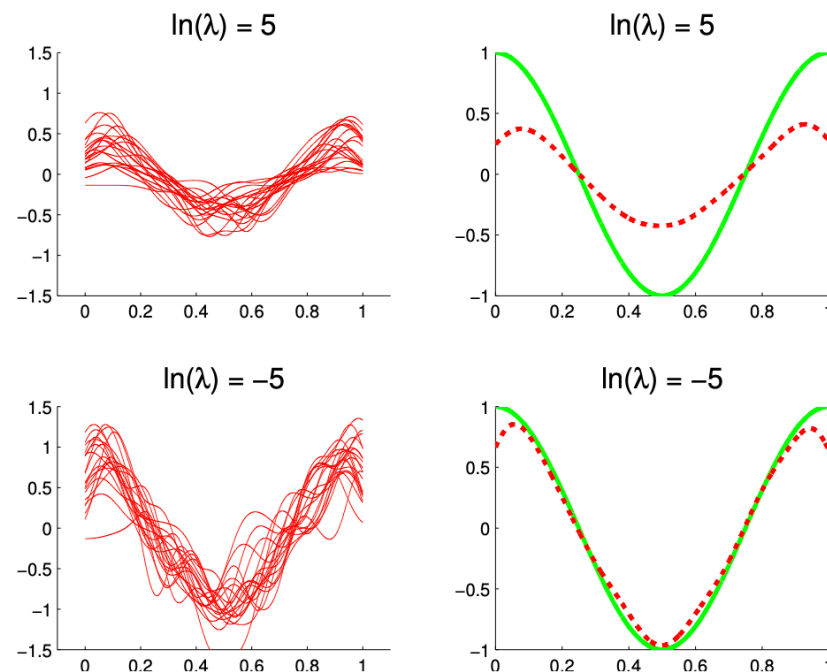


Figure 4.25: Illustration of bias-variance tradeoff for ridge regression. We generate 100 data sets from the true function, shown in solid green. Left: we plot the regularized fit for 20 different data sets. We use linear regression with a Gaussian RBF expansion, with 25 centers evenly spread over the $[0, 1]$ interval. Right: we plot the average of the fits, averaged over all 100 datasets. Top row: strongly regularized: we see that the individual fits are similar to each other (low variance), but the average is far from the truth (high bias). Bottom row: lightly regularized: we see that the individual fits are quite different from each other (high variance), but the average is close to the truth (low bias). Adapted from [Bis06] Figure 3.5. Generated by code.probml.ai/book1/4.25.

Bias-Variance Tradeoff

- **Your task:** qualitatively plot bias² and variance in fig 2.11
- Change model type? Affect bias
- More training data: decrease variance
 - “consistent estimators” converge to ideal solution as $|D| \rightarrow \text{infinity}$
 - For small data sets, lower complexity models may be preferred.
- **Ideal solution: suitable model type & complexity**

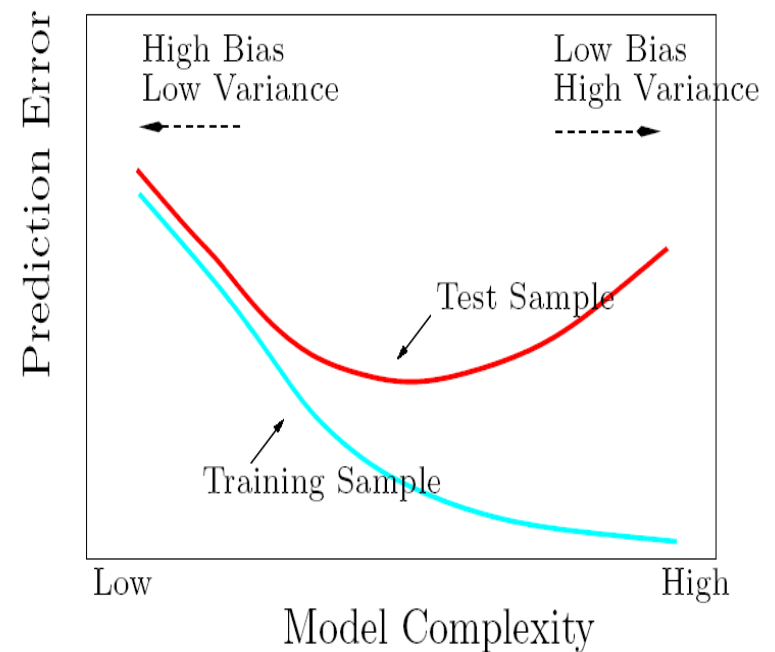


Figure 2.11: Test and training error as a function of model complexity.

*Bias –variance tradeoff is encountered in many situations
Example: determining # of bins for a histogram.*

Learning Curves

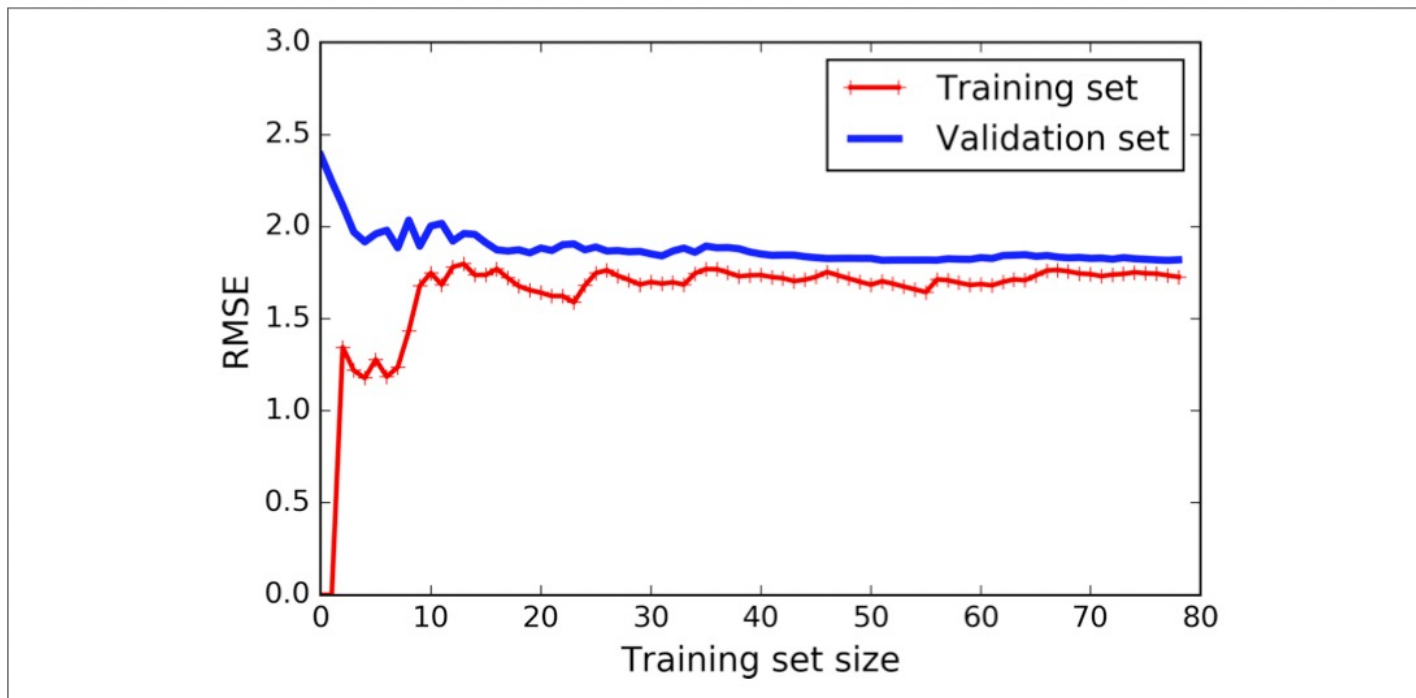


Figure 4-15. Learning curves (from AG, pg 132)

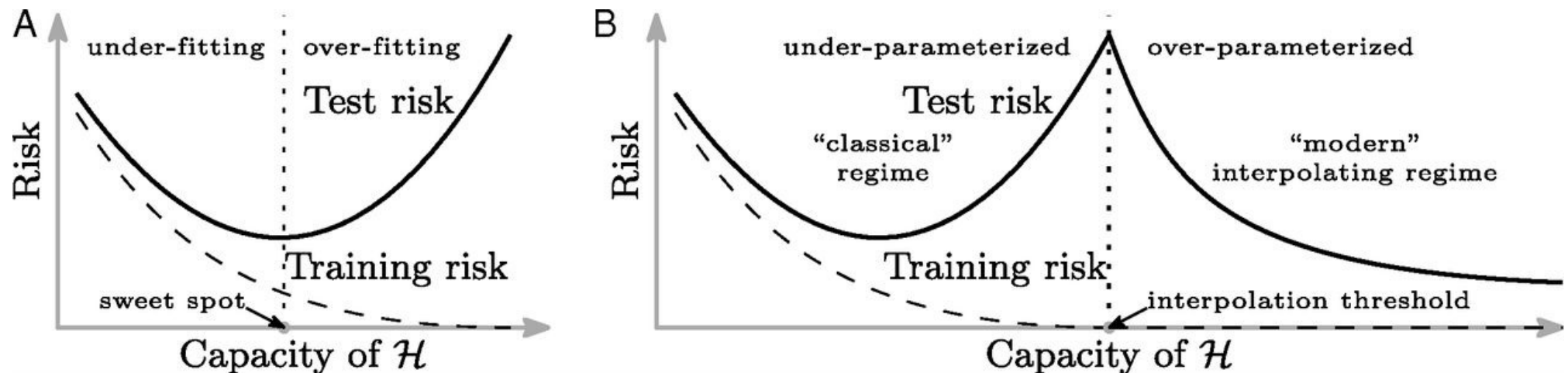
- How will these curves look like for different degree polynomials for the "sine curve" example?
- Understand using bias-variance

Application: How do you improve your model?

- Get more training data
- Change complexity (e.g. via regularization)
- Change optimization method
- Change Model type
- Still not acceptable?
 - Change feature space

Breaking News (2019)*

- Variance can actually go down in the highly over-parameterized regime (going beyond interpolation)!!
 - Figure below from M. Belkin, D. Hsu, S. Ma, and S. Mandal, “Reconciling modern machine learning practice and the bias-variance trade-off,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15849–15854, 2019.



Extras

Function Approximation / Regression/Prediction

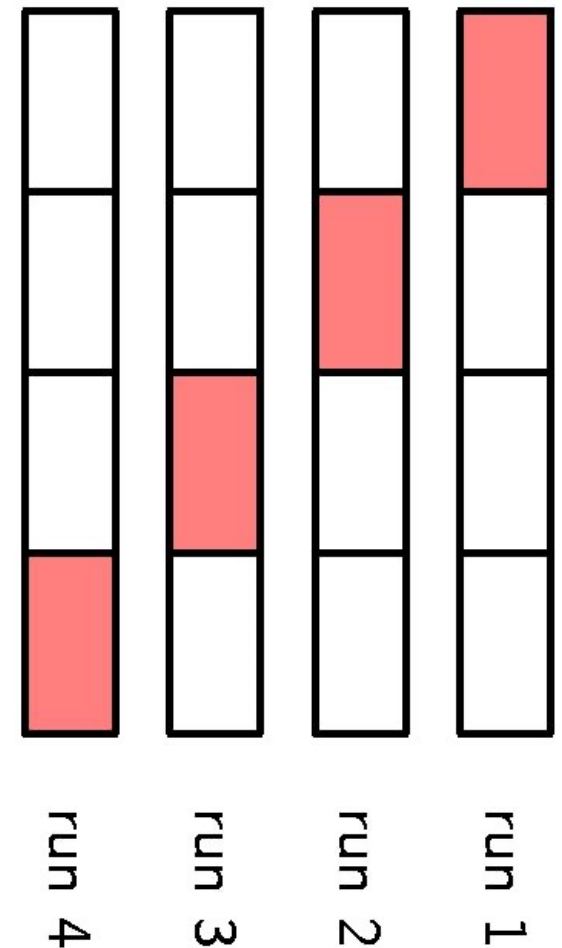
- A predictive modeling technique
 - Given:
 - A set of input (AI) /independent (math)/ explanatory or predictor (stats) variables X
 - corresponding (set of) output/dependent/response variables T
 - Think of training/test datasets as i.i.d. samples from an underlying joint distribution $p(X,T)$
 - Build: a model relating X to T
 - **single value for** T given X (most common)
 - e.g. $E[T | X]$, the “regression of t on X .”
 - Assumes $T = \text{function of } X + (\text{zero-mean, symmetric}) \text{ noise}$
 - **Add Confidence Interval** (e.g. based on the Normally distributed noise term in MLR)
 - **(Arbitrary) Distribution of T given X**

Estimating True Performance (Data Driven)

- enough data? Use “holdout” to estimate
- Moderately large? Use k-fold cross-validation

$k = 4$ example

Shaded subset: validation set for that run

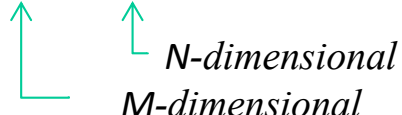


Geometry of Least Squares*

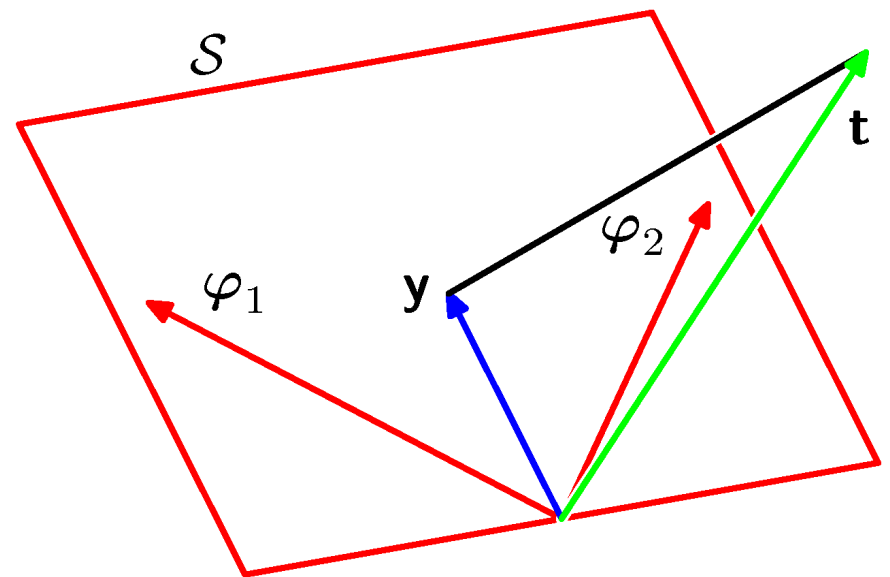
- Consider

$$\mathbf{y} = \Phi \mathbf{w}_{\text{ML}} = [\varphi_1, \dots, \varphi_M] \mathbf{w}_{\text{ML}}.$$

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


N-dimensional
M-dimensional

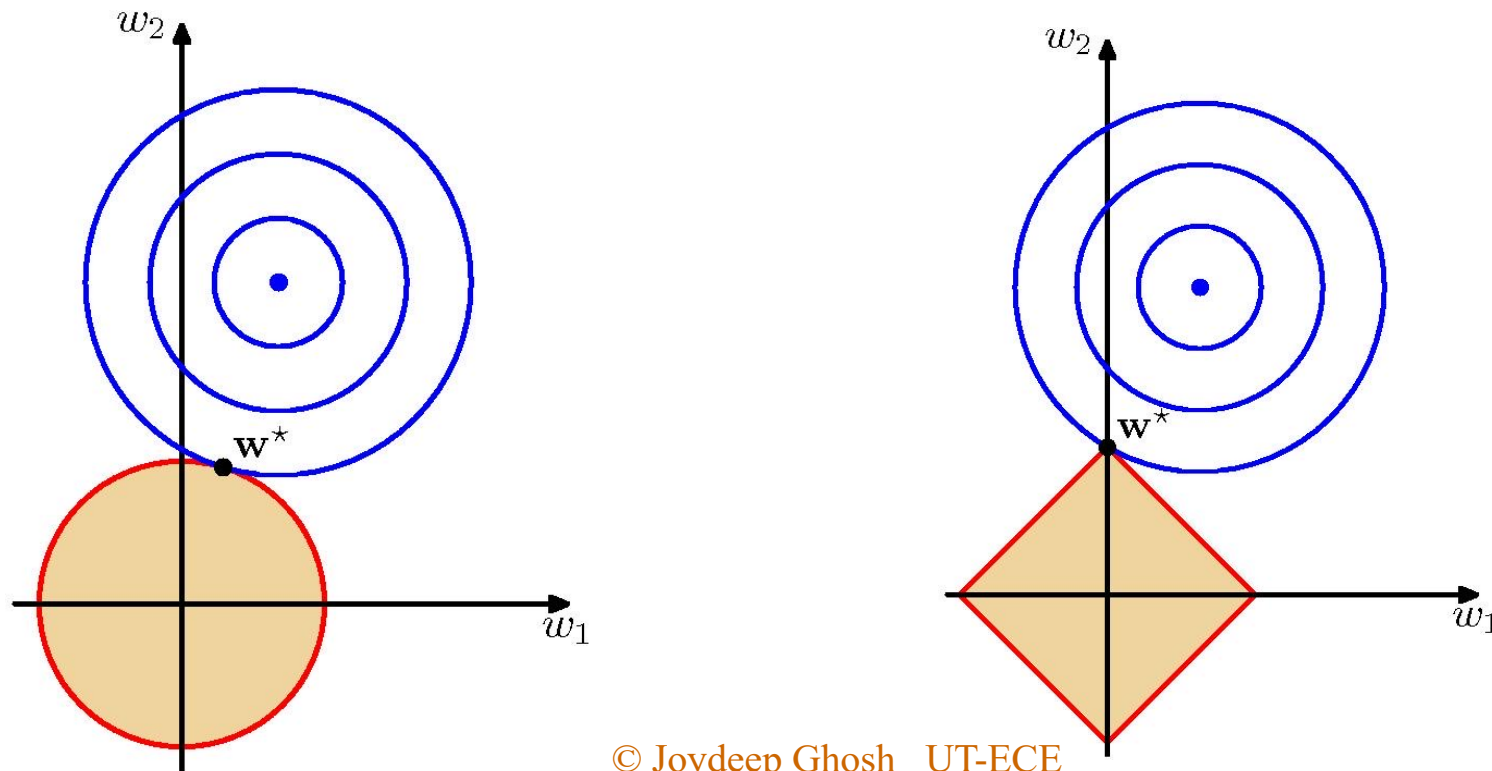
- \mathcal{S} is spanned by $\varphi_1, \dots, \varphi_M$.
- \mathbf{w}_{ML} minimizes the distance between \mathbf{t} and its orthogonal projection on \mathcal{S} , i.e. \mathbf{y} .



Takeaway: You are restricted by your choice of the features

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 = $f(\text{model type, \# of parameters, \# of training points})$
 - e.g. linear regression with N samples, P parameters
Akaike's Final Prediction error = $MSE_{\text{empirical}} (N+P) / (N - P)$
 - 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*

Bias-Variance vs. Regularization Amount

- What happens to the curves as amount of training data increases ? (note: effective model complexity is *decreasing* towards the right)*

