

CS 886  
Applied Machine Learning  
Introduction Part 2 - Regression, Model Selection,  
Performance Evaluation

Dan Lizotte

University of Waterloo

13 Sept 2012

## Notation reminder

- Consider a function  $f(u_1, u_2, \dots, u_m) : \mathbb{R}^m \mapsto \mathbb{R}$  (for us, this will usually be an error function)
- The *gradient*  $\nabla f(u_1, u_2, \dots, u_m) : \mathbb{R}^n \mapsto \mathbb{R}^n$  is a function which outputs a vector containing the partial derivatives.

That is:

$$\nabla f = \left\langle \frac{\partial}{\partial u_1} f, \frac{\partial}{\partial u_2} f, \dots, \frac{\partial}{\partial u_m} f \right\rangle$$

- If  $f$  is differentiable and convex, we can find the global minimum of  $f$  by solving  $\nabla f = \mathbf{0}$ .
- The partial derivative is the derivative along the  $u_i$  axis, keeping all other variables fixed.

# The Least Squares Solution

- Recalling some multivariate calculus:

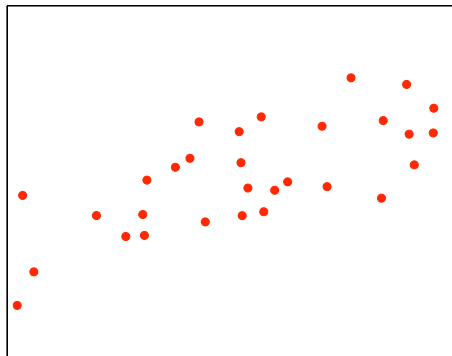
$$\begin{aligned}\nabla_{\mathbf{w}} J &= \nabla_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \nabla_{\mathbf{w}} (\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{y}^T \mathbf{X} \mathbf{w} - \mathbf{w}^T \mathbf{X}^T \mathbf{y} + \mathbf{y}^T \mathbf{y}) \\ &= \nabla_{\mathbf{w}} (\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{y}^T \mathbf{X} \mathbf{w} + \mathbf{y}^T \mathbf{y}) \\ &= 2\mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{X}^T \mathbf{y}\end{aligned}$$

- Setting gradient equal to zero:

$$\begin{aligned}2\mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{X}^T \mathbf{y} &= 0 \\ \Rightarrow \mathbf{X}^T \mathbf{X} \mathbf{w} &= \mathbf{X}^T \mathbf{y} \\ \Rightarrow \mathbf{w} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}\end{aligned}$$

- The inverse exists if the columns of  $\mathbf{X}$  are linearly independent.

# Example of linear regression



x	y
0.86	2.49
0.09	0.83
-0.85	-0.25
0.87	3.10
-0.44	0.87
-0.43	0.02
-1.10	-0.12
0.40	1.81
-0.96	-0.83
0.17	0.43

# Data matrices

$$X = \begin{bmatrix} 1 & 0.86 \\ 1 & 0.09 \\ 1 & -0.85 \\ 1 & 0.87 \\ 1 & -0.44 \\ 1 & -0.43 \\ 1 & -1.10 \\ 1 & 0.40 \\ 1 & -0.96 \\ 1 & 0.17 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \end{bmatrix} \times \begin{bmatrix} 1 & 0.86 \\ 1 & 0.09 \\ 1 & -0.85 \\ 1 & 0.87 \\ 1 & -0.44 \\ 1 & -0.43 \\ 1 & -1.10 \\ 1 & 0.40 \\ 1 & -0.96 \\ 1 & 0.17 \end{bmatrix}$$
$$= \begin{bmatrix} 10 & -1.39 \\ -1.39 & 4.95 \end{bmatrix}$$

$$X^T \mathbf{y} =$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \end{bmatrix} \times \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$
$$= \begin{bmatrix} 8.34 \\ 6.49 \end{bmatrix}$$

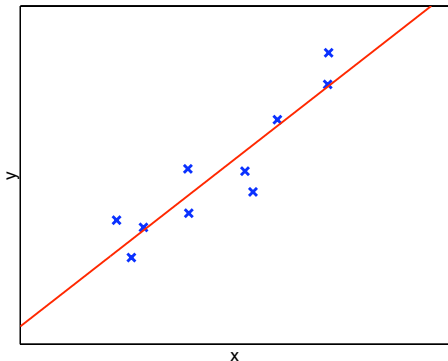
## Solving for $\mathbf{w}$

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y} = \begin{bmatrix} 10 & -1.39 \\ -1.39 & 4.95 \end{bmatrix}^{-1} \begin{bmatrix} 8.34 \\ 6.49 \end{bmatrix} = \begin{bmatrix} 1.05 \\ 1.60 \end{bmatrix}$$

So the best fit line is  $y = 1.05 + 1.60x$ .



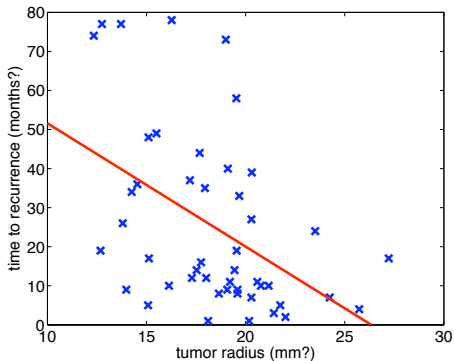
# Data and line $y = 1.05 + 1.60x$



# Linear regression summary

- The optimal solution (minimizing sum-squared-error) can be computed in polynomial time in the size of the data set.
- The solution is  $\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$ , where  $X$  is the data matrix augmented with a column of ones, and  $\mathbf{y}$  is the column vector of target outputs.
- A very rare case in which an analytical, exact solution is possible

# Predicting recurrence time based on tumor size



# Is linear regression enough?

- Linear regression should be the **first thing** you try for real-valued outputs!
- ...but it is sometimes not expressive enough.<sup>1</sup>
- Two possible solutions:
  - ① Explicitly transform the data, i.e. create additional features
    - Add cross-terms, higher-order terms
    - More generally, apply a transformation of the inputs from  $\mathcal{X}$  to some other space  $\mathcal{X}'$ , then do linear regression in the transformed space
  - ② Use a different hypothesis class
- Idea (1) and idea (2) are two views of the strategy.  
Today we focus on the first approach

---

<sup>1</sup>Problems can also occur if  $X^T X$  is not invertible.

# Polynomial fits

- Suppose we want to fit a higher-degree polynomial to the data. (E.g.,  $y = w_0 + w_1x_1 + w_2x_1^2$ .)
- Suppose for now that there is a single input variable  $x_{i,1}$  per training sample.
- How do we do it?

## Answer: Polynomial regression

- Given data:  $(x_{1,1}, y_1), (x_{1,2}, y_2), \dots, (x_{1,n}, y_n)$ .
- Suppose we want a degree- $d$  polynomial fit.
- Let  $\mathbf{y}$  be as before and let

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,1}^2 & \cdots & x_{1,1}^d \\ 1 & x_{1,2} & x_{1,2}^2 & \cdots & x_{1,2}^d \\ \vdots & & \vdots & \vdots & \vdots \\ 1 & x_{1,n} & x_{1,n}^2 & \cdots & x_{1,n}^d \end{bmatrix}$$

- We are **making up features** to add to our design matrix
- Solve the linear regression  $X\mathbf{w} \approx \mathbf{y}$ .

## Example of quadratic regression: Data matrices

$$X = \begin{bmatrix} 1 & 0.86 & 0.75 \\ 1 & 0.09 & 0.01 \\ 1 & -0.85 & 0.73 \\ 1 & 0.87 & 0.76 \\ 1 & -0.44 & 0.19 \\ 1 & -0.43 & 0.18 \\ 1 & -1.10 & 1.22 \\ 1 & 0.40 & 0.16 \\ 1 & -0.96 & 0.93 \\ 1 & 0.17 & 0.03 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

$$X^T X =$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 0.75 & 0.01 & 0.73 & 0.76 & 0.19 & 0.18 & 1.22 & 0.16 & 0.93 & 0.03 \end{bmatrix} \times \begin{bmatrix} 1 & 0.86 & 0.75 \\ 1 & 0.09 & 0.01 \\ 1 & -0.85 & 0.73 \\ 1 & 0.87 & 0.76 \\ 1 & -0.44 & 0.19 \\ 1 & -0.43 & 0.18 \\ 1 & -1.10 & 1.22 \\ 1 & 0.40 & 0.16 \\ 1 & -0.96 & 0.93 \\ 1 & 0.17 & 0.03 \end{bmatrix}$$
$$= \begin{bmatrix} 10 & -1.39 & 4.95 \\ -1.39 & 4.95 & 1.64 \\ 4.95 & 1.64 & 4.11 \end{bmatrix}$$



$$X^T \mathbf{y} =$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\ 0.75 & 0.01 & 0.73 & 0.76 & 0.19 & 0.18 & 1.22 & 0.16 & 0.93 & 0.03 \end{bmatrix} \times \begin{bmatrix} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{bmatrix}$$

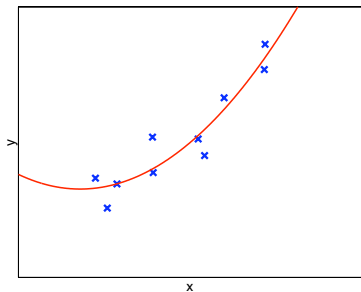
$$= \begin{bmatrix} 8.34 \\ 6.49 \\ 3.60 \end{bmatrix}$$

## Solving for $\mathbf{w}$

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y} = \begin{bmatrix} 10 & -1.39 & 4.95 \\ -1.39 & 4.95 & 1.64 \\ 4.95 & 1.64 & 4.11 \end{bmatrix}^{-1} \begin{bmatrix} 3.60 \\ 6.49 \\ 8.34 \end{bmatrix} = \begin{bmatrix} 0.73 \\ 1.74 \\ 0.68 \end{bmatrix}$$

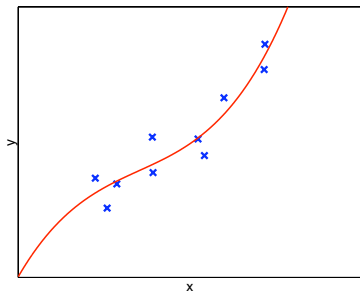
So the best order-2 polynomial is  $y = 0.73 + 1.74x + 0.68x^2$ .

Data and curve  $y = 0.68x^2 + 1.74x + 0.73$



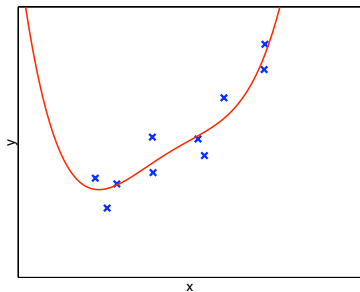
Is this a better fit to the data?

## Order-3 fit



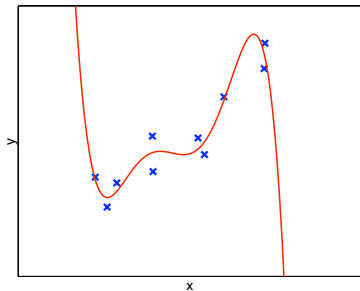
Is this a better fit to the data?

## Order-4 fit



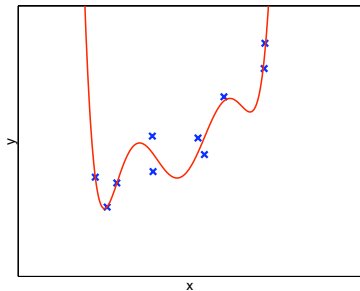
Is this a better fit to the data?

## Order-5 fit



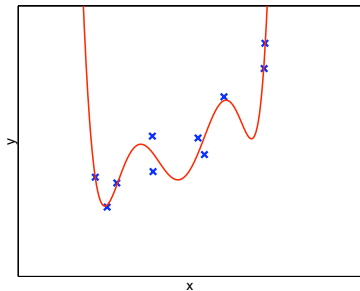
Is this a better fit to the data?

## Order-6 fit



Is this a better fit to the data?

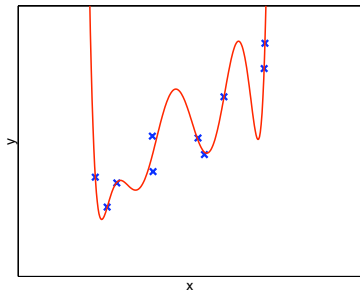
## Order-7 fit



Is this a better fit to the data?

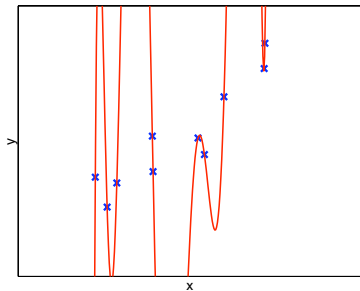


## Order-8 fit



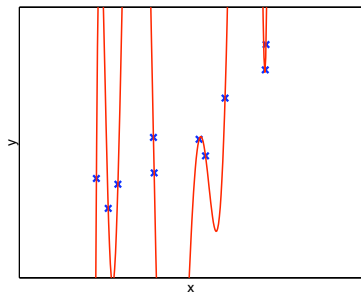
Is this a better fit to the data?

## Order-9 fit



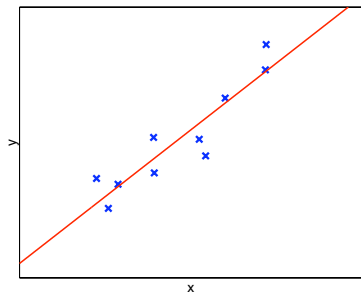
Is this a better fit to the data?

# Evaluating Performance



Which do you prefer and why?

Fits the data we have right now



Fits data we will see in the future

## Performance of a Fixed Hypothesis

- Assume that data  $(\mathbf{x}, y)$  are drawn from some fixed, unknown probability distribution  $P(\mathbf{x}, y)$
- Given a hypothesis  $h$ , (which could have come from anywhere), its *generalization error* is:

$$J_h^* = \mathbb{E}[L(h(\mathbf{x}), y)]$$

- We don't have access to  $P(\mathbf{x}, y)$ , but if we have access to a *test set* of data, we can compute the *test error*

$$\hat{J}_h^* = \frac{1}{n} \sum_{i=1}^n L(h(\mathbf{x}_i), y_i)$$

- $\hat{J}_h^*$  is an *unbiased* estimate of  $J_h^*$  **so long as the  $(\mathbf{x}_i, y_i)$  do not influence  $h$** . Can use  $\hat{J}_h^*$  to get a confidence interval for  $J_h^*$ .

## Test Error: The Gold Standard

$$\hat{J}_h^* = \frac{1}{n} \sum_{i=1}^n L(h(\mathbf{x}_i), y_i)$$

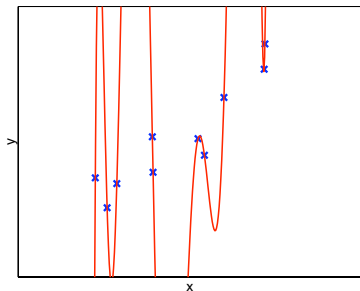
- $\hat{J}_h^*$  is an *unbiased* estimate of  $J_h^*$  **so long as the  $(\mathbf{x}_i, y_i)$  do not influence  $h$** . Can use  $\hat{J}_h^*$  to get a confidence interval for  $J_h^*$ .
- Gives a strong statistical guarantee about the true performance of our system, *if we didn't use the test data to choose  $h$* .
- Note we can write training error for hypothesis class  $\mathcal{H}$  as

$$\hat{J}_{\mathcal{H}} = \min_{h' \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n L(h'(\mathbf{x}_i), y_i)$$

- Obviously, for any data set,  $\hat{J}_{\mathcal{H}} \leq \hat{J}_h^*$ .

## Problem 1 with Training Error

Training error  $\hat{J}_{\mathcal{H}}$  systematically underestimates generalization error  $J_h^*$

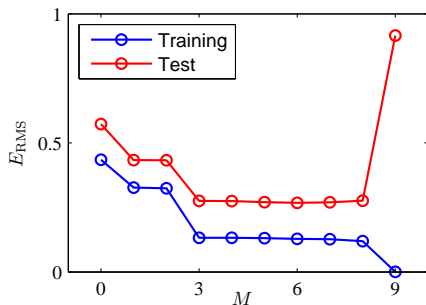


- Training error of the degree-9 polynomial is 0.
- Training error of the degree-9 polynomial *on any set of 10 points* is 0.
- The more complex the model and the smaller the training set, the worse this is.

## Problem 2 with Training Error

Smaller training error does not mean smaller generalization error.

- Suppose  $\mathcal{H}_1$  is the space of all linear functions,  $\mathcal{H}_2$  is the space of all quadratic functions. Note  $\mathcal{H}_1 \subset \mathcal{H}_2$ .
- Let  $h_1 = \arg \min_{H \in \mathcal{H}_1} \hat{J}_{H'}^*$  and  $h_2 = \arg \min_{H \in \mathcal{H}_2} \hat{J}_{H'}^*$
- We **must** have  $\hat{J}_{h_2}^* \leq \hat{J}_{h_1}^*$ , but we may have  $J_{h_2}^* > J_{h_1}^*$ .



Training error is no good for choosing the hypothesis class.

# Fix the problems with Training Error?

- ① Training error  $\hat{J}_{\mathcal{H}}$  underestimates generalization error  $J_h^*$ 
  - If you really want a good estimate of  $J_h^*$ , you need a **test set**
  - (But new stat methods can produce a CI using training error)
  - Could report test error, then deploy whatever you train on the whole data. (Probably won't be worse.)
- ② Smaller training error does not mean smaller generalization error.
  - Known as **overfitting**
  - Hypothesis class choice problem is called *model selection*
  - A **validation set** can be used for this. Train on the training set using each proposed hypothesis class, evaluate each on the validation set, choose the one with lowest *validation error*



# Training, Model Selection, and Error Estimation

- A general procedure for estimating the true error of a specific learned hypothesis using model selection
- The data is randomly partitioned into three subsets:
  - A *training set* used only to find the parameters  $\mathbf{w}$
  - A *validation set* used to find the right hypothesis class (e.g., the degree of the polynomial)
  - A *test set* used to report the prediction error of the algorithm
- The test set *must be disjoint from training and validation!*
- Can generate standard confidence intervals for the test error of the learned hypothesis

# Problems with the Single-Partition Approach

- Pros:
  - Measures what we want. Performance of the actual learned hypothesis.
- Cons:
  - Why don't we use all the data we have? Is it rational to "throw away" data that could have been used for training/model selection?
  - Can produce a high-variance estimate, especially for classification. (actually the bigger concern) "What if I get a weird test/train/validation set 'by accident?' "
  - For a test set of size 100, with 60 correct classifications, 95% C.I. for actual accuracy is (0.497, 0.698).

## k-fold cross-validation

- Divide the instances into  $k$  disjoint partitions or folds
- Loop through the partitions  $i = 1 \dots k$ :
  - Partition  $i$  is for testing (i.e., estimating the performance of the algorithm after learning is done)
  - One of the other partitions is for validation (e.g., choosing the hypothesis class or the parameters of the learning algorithm)
  - The rest are used for training (e.g., choosing the specific hypothesis within the class)
- *Report average error on the testing partitions*
- You should also compute and report standard error based on the testing errors on the different folds
- Magic number:  $k = 10$
- To deploy at the end of the day, train on all the data using your chosen hypothesis class. If you want to estimate its error, go get more data.

## Cross Validation [HTF 7.12]

- Error on each test fold is an unbiased estimate of generalization error of a hypothesis *trained on the rest of the data*
- It is an **average of error estimates for  $k$  different hypotheses**
- They're similar: Each was trained on a slightly different dataset from the same distribution
- CV estimate approximately unbiased for the *expected generalization error*
  - ① Draw many datasets of size  $n$
  - ② Train on each one (maybe split for validation; doesn't matter)
  - ③ Average the true generalization error of each of the hypotheses
- **This is not the generalization error of the hypothesis learned from the data we actually have.**
- More like an evaluation of the learning method.

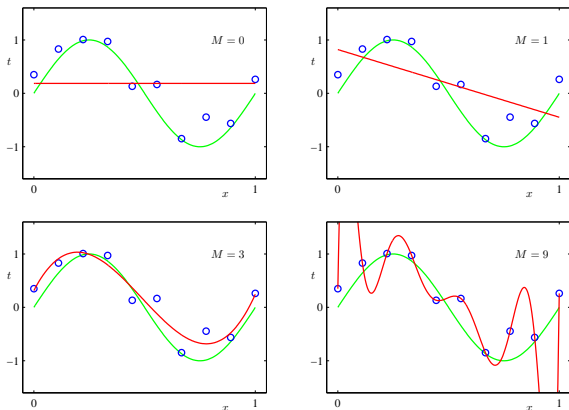
# Cross Validation

- Not exactly what we want, but close
- Standard errors are usually shown (i.e. standard deviation of test errors) but cannot<sup>2</sup> be used to produce valid confidence intervals
- Well-accepted

---

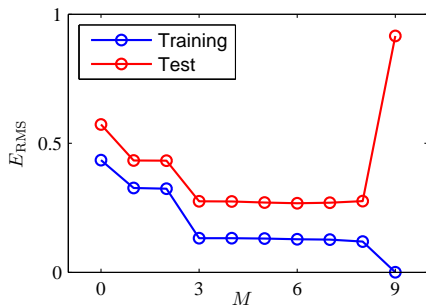
<sup>2</sup>easily: “Some progress has been made on constructing confidence intervals around cross-validation estimates, but this is considered a difficult problem.” - Wikipedia. The WP CV article is pretty good.

# Summary: Overfitting



- The higher the degree of the polynomial  $M$ , the more degrees of freedom, and the more capacity to “overfit” the training data
- Typical overfitting means that error on the training data is very low, but error on new instances is high

## Summary: Overfitting



- The training error decreases with the degree of the polynomial  $M$ , i.e. *the complexity (size) of the hypothesis class*
- Generalization error decreases at first, then starts increasing
- Set aside a *validation set* helps us find a good hypothesis class
- We then can report unbiased error estimate, using a *test set*, **untouched during both parameter training and validation**
- Cross-validation is a lower-variance but possibly biased version of this approach. It is standard.