COMP 551 – Applied Machine Learning
Lecture 3: Linear regression (cont'd)

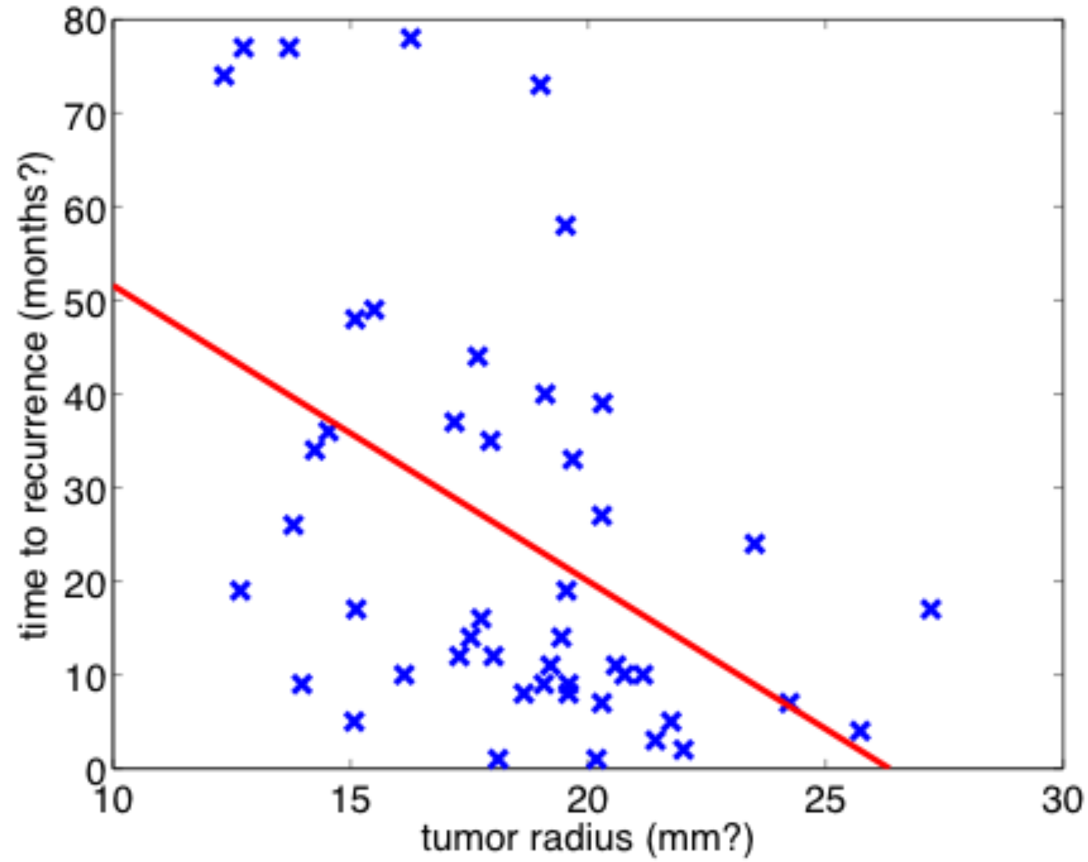
Instructor: Joelle Pineau (*jpineau@cs.mcgill.ca)*

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Predicting recurrence time from tumor size

This function looks complicated, and a linear hypothesis does not seem very good.

What should we do?

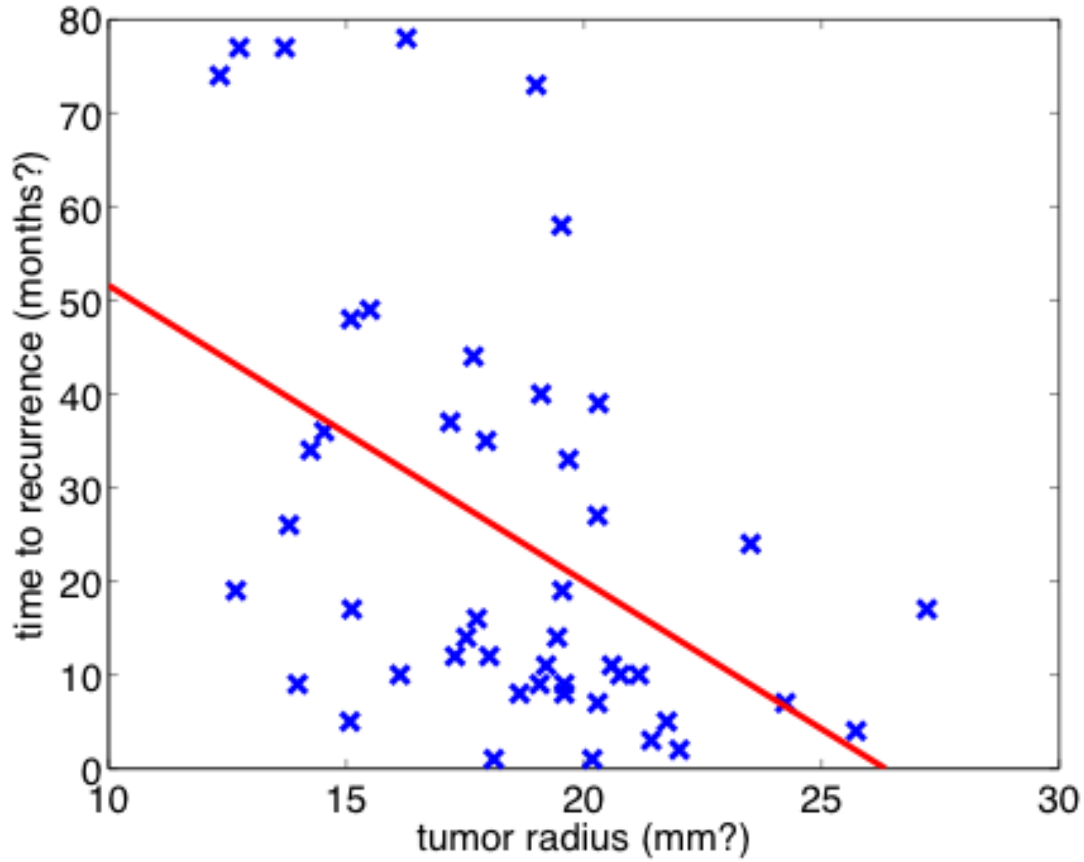


Predicting recurrence time from tumor size

This function looks complicated, and a linear hypothesis does not seem very good.

What should we do?

- *Pick a better function?*
- *Use more features?*
- *Get more data?*



Dealing with difficult cases of $(X^T X)^{-1}$

- **Case #1**: The weights are not uniquely defined.

  Solution: Re-code or drop some redundant columns of X.

- **Case #2**: The number of features/weights (m) exceeds the number of training examples (n).

  Solution: Reduce the number of features using various techniques (to be studied later.)

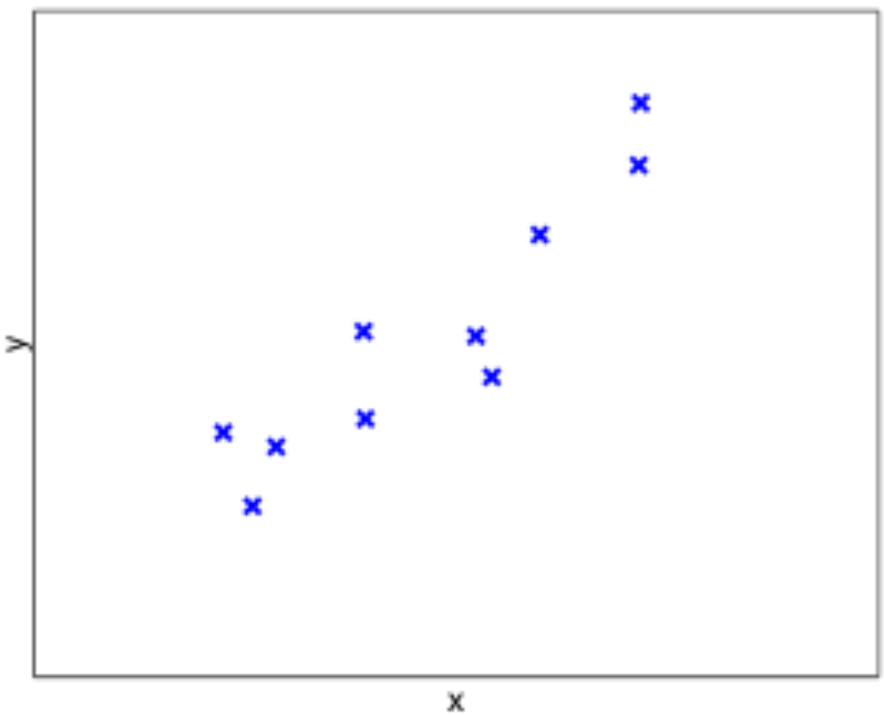
Input variables for linear regression

- Original quantitative variables $X_1, \ldots, X_m$
- Transformations of variables, e.g. $X_{m+1} = log(X_i)$
- Basis expansions, e.g. $X_{m+1} = X_i^2$, $X_{m+2} = X_i^3$, …
- Interaction terms, e.g. $X_{m+1} = X_i X_j$
- Numeric coding of qualitative variables, e.g. $X_{m+1} = 1$ if X_i is true and 0 otherwise.

In all cases, we can add $X_{m+1}, \ldots, X_{m+k}$ to the list of original variables and perform the linear regression.

Example of linear regression with polynomial terms

$$f_w(x) \quad = \quad w_0 + w_1\, x + w_2\, x^2$$

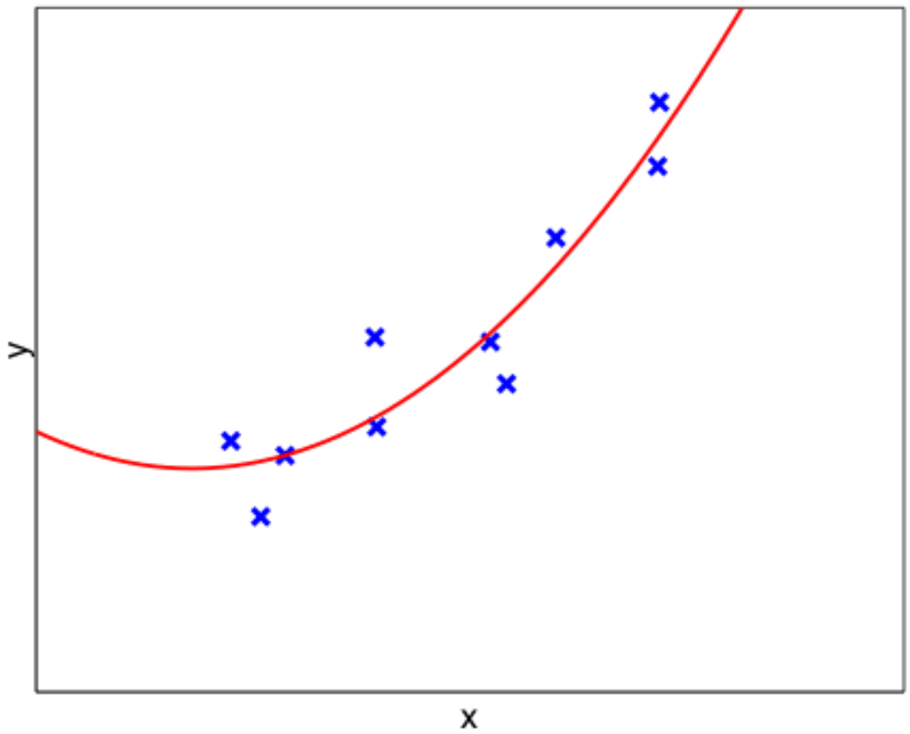


$$X = \begin{array}{c} \begin{array}{ccc} x^2 & x & \end{array} \\ \left[\begin{array}{ccc} 0.75 & 0.86 & 1 \\ 0.01 & 0.09 & 1 \\ 0.73 & -0.85 & 1 \\ 0.76 & 0.87 & 1 \\ 0.19 & -0.44 & 1 \\ 0.18 & -0.43 & 1 \\ 1.22 & -1.10 & 1 \\ 0.16 & 0.40 & 1 \\ 0.93 & -0.96 & 1 \\ 0.03 & 0.17 & 1 \end{array}\right] \end{array} \qquad Y = \left[\begin{array}{c} 2.49 \\ 0.83 \\ -0.25 \\ 3.10 \\ 0.87 \\ 0.02 \\ -0.12 \\ 1.81 \\ -0.83 \\ 0.43 \end{array}\right]$$

Solving the problem

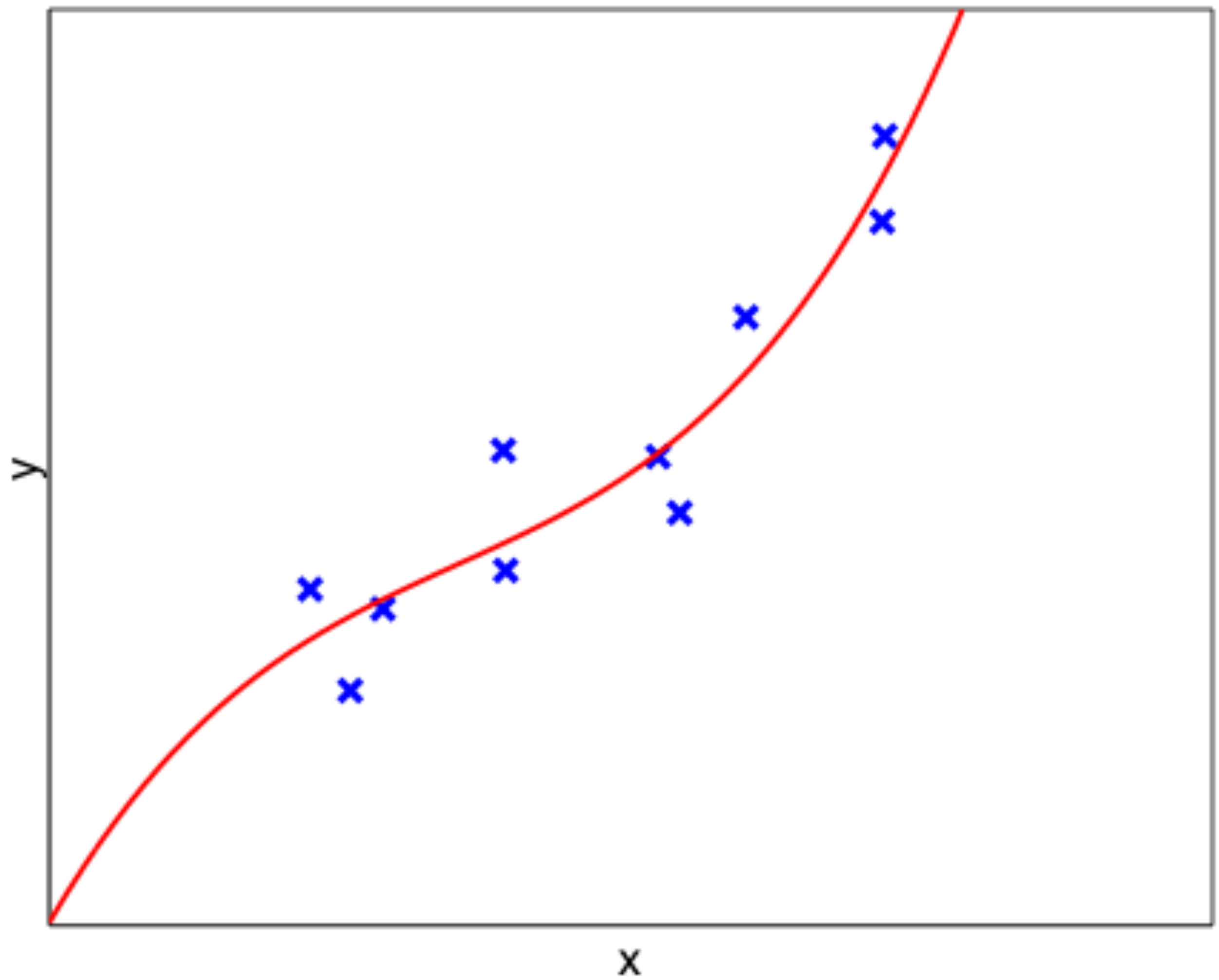
$$\mathbf{w} = (X^T X)^{-1} X^T Y = \begin{bmatrix} 4.11 & -1.64 & 4.95 \\ -1.64 & 4.95 & -1.39 \\ 4.95 & -1.39 & 10 \end{bmatrix}^{-1} \begin{bmatrix} 3.60 \\ 6.49 \\ 8.34 \end{bmatrix} = \begin{bmatrix} 0.68 \\ 1.74 \\ 0.73 \end{bmatrix}$$

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

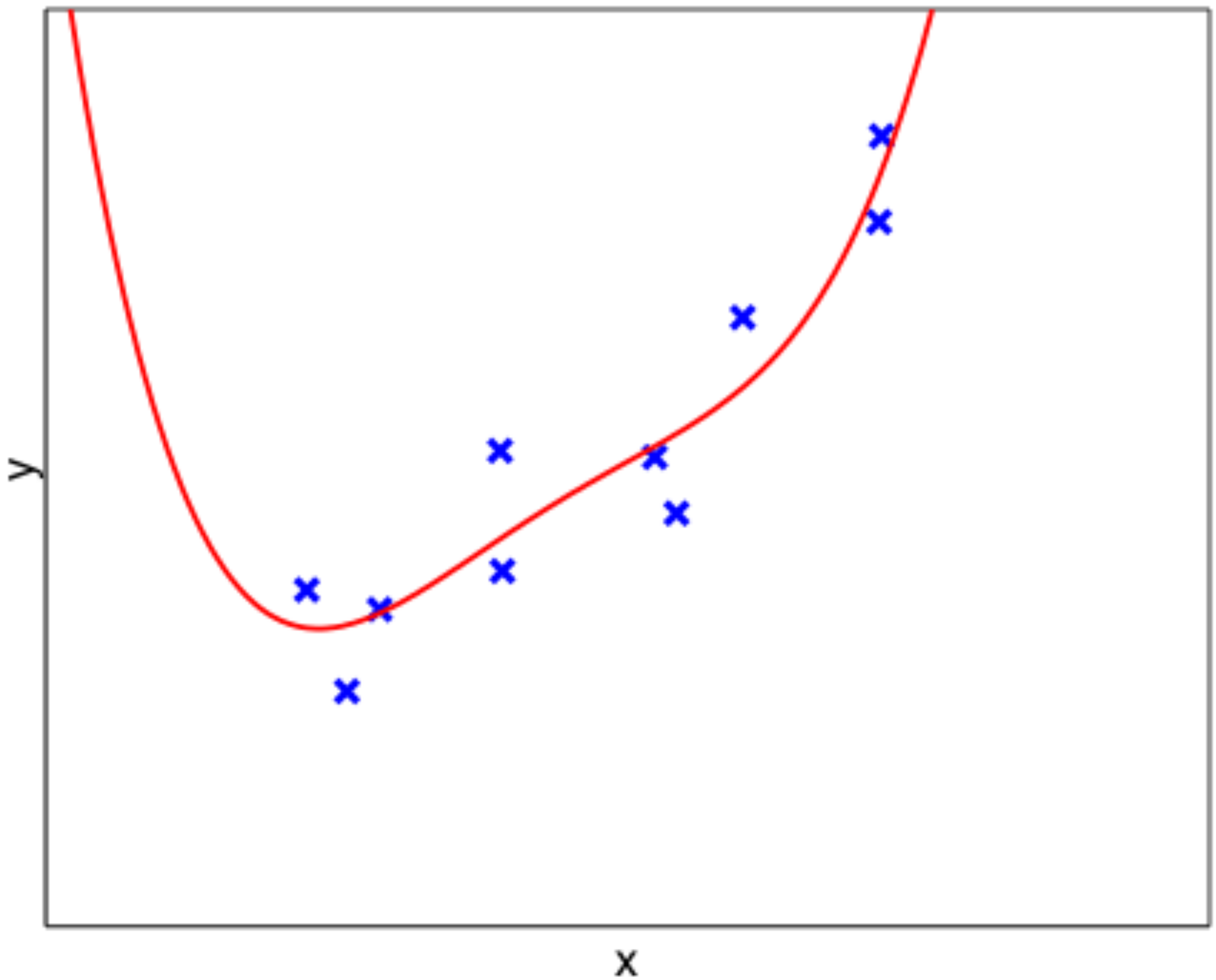


Compared to *y* = 1.6*x* + 1.05 for the order-1 polynomial.

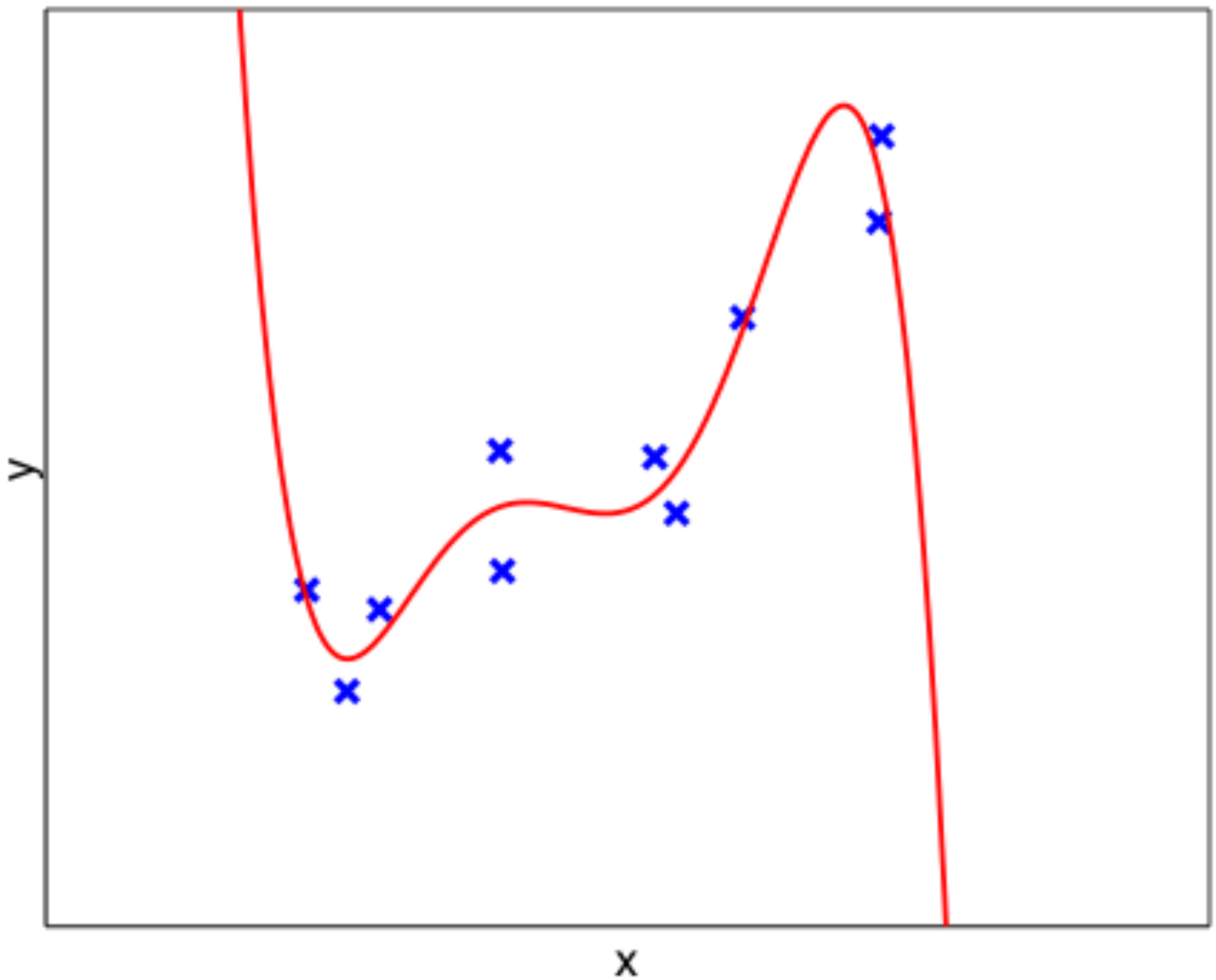
Order-3 fit: Is this better?



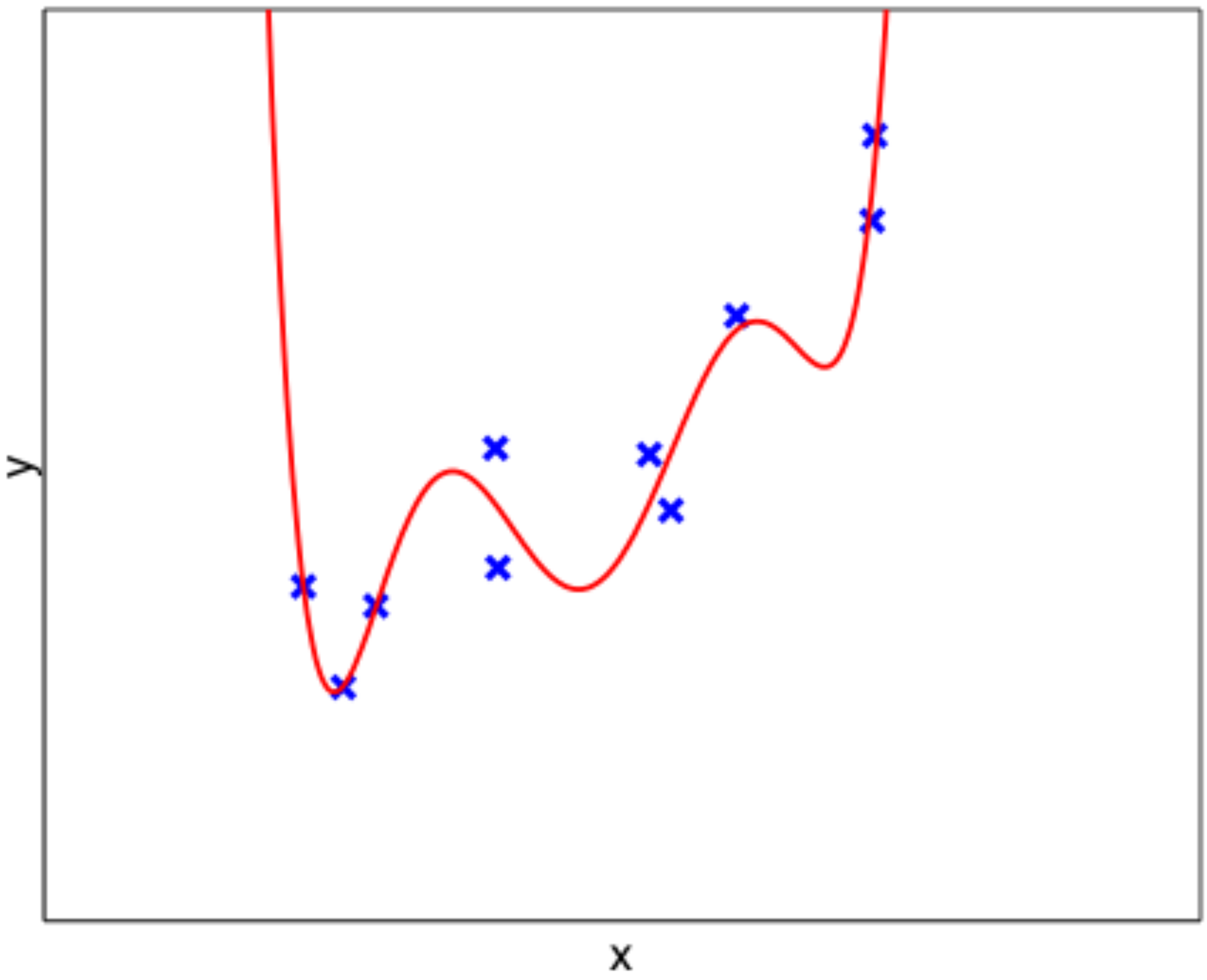
Order-4 fit



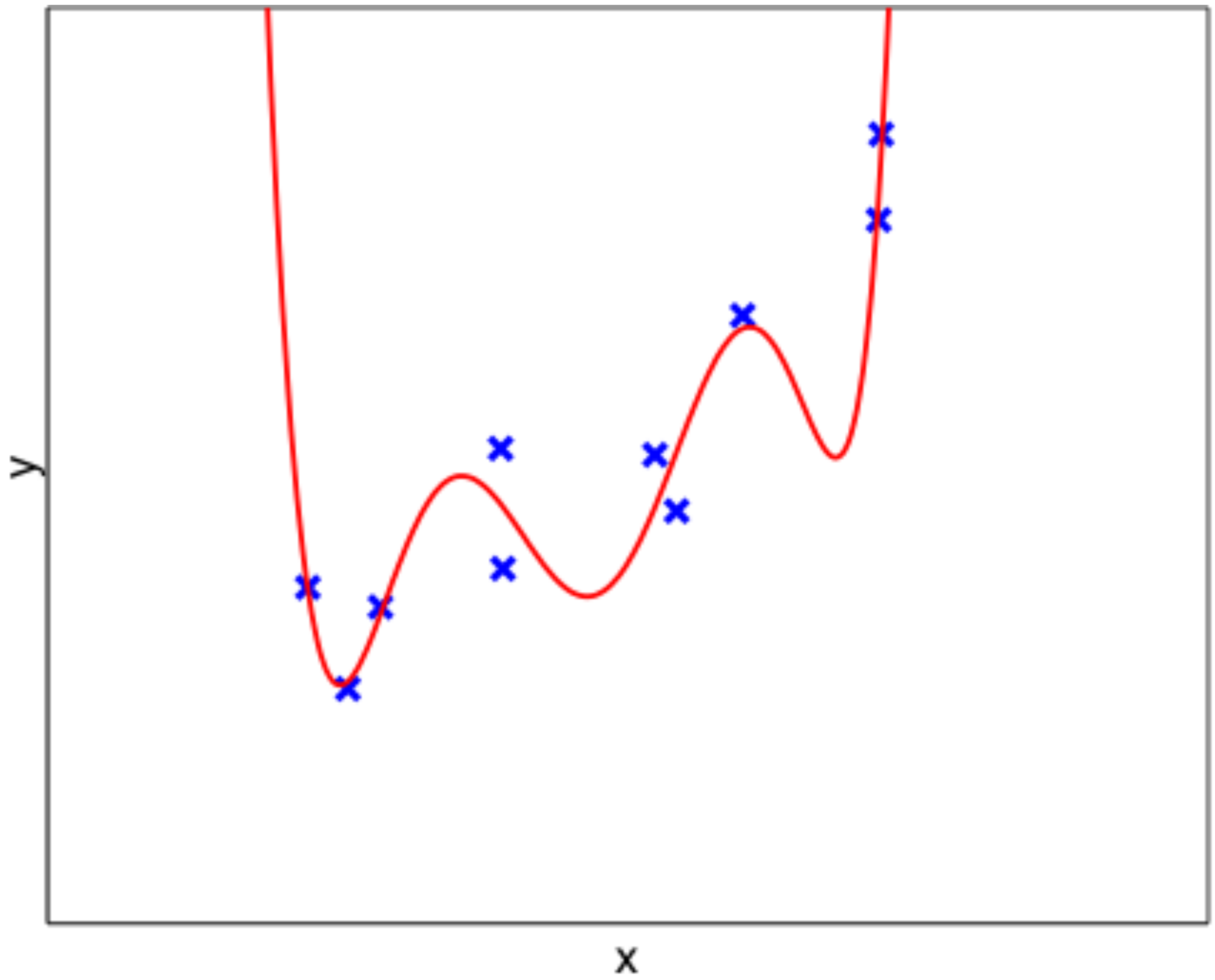
Order-5 fit



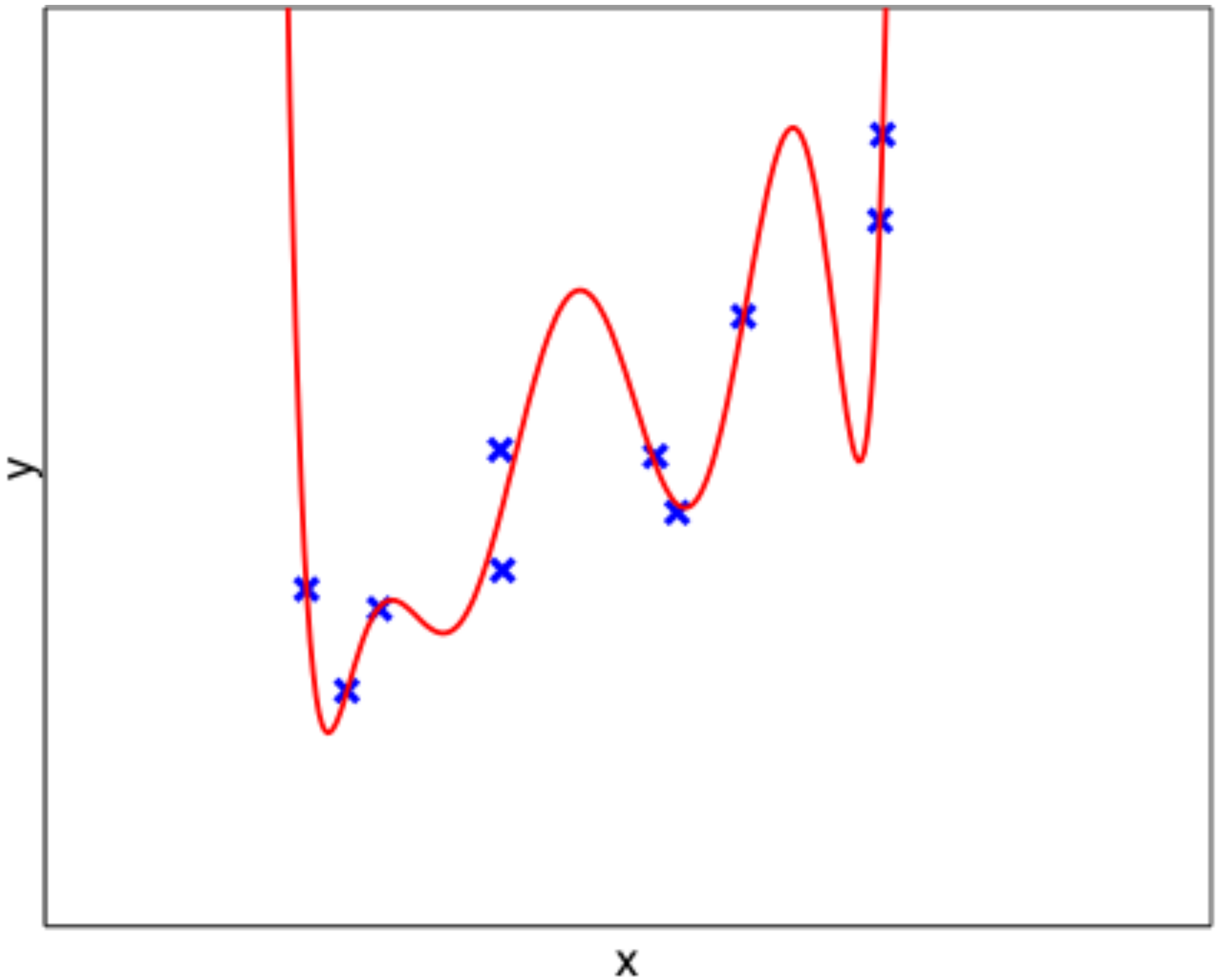
Order-6 fit



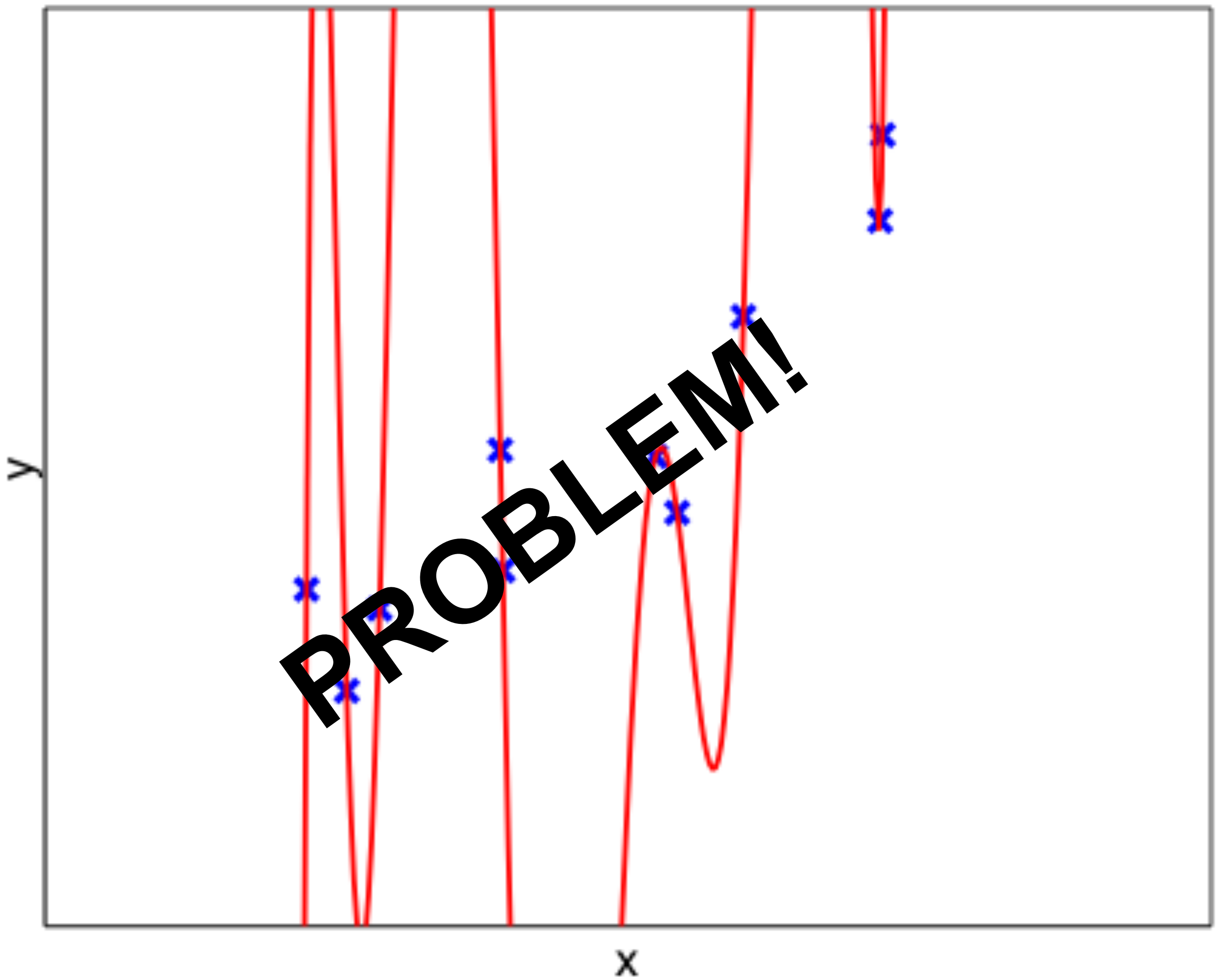
Order-7 fit



Order-8 fit



Order-9 fit



This is overfitting!

- We can find a hypothesis that explains perfectly the training data, but **does not generalize** well to new data.

- **In this example**: we have a lot of parameters (weights), so the
  hypothesis matches the data points exactly,
  but is wild everywhere else.

- A **very important** problem in machine learning.

Overfitting

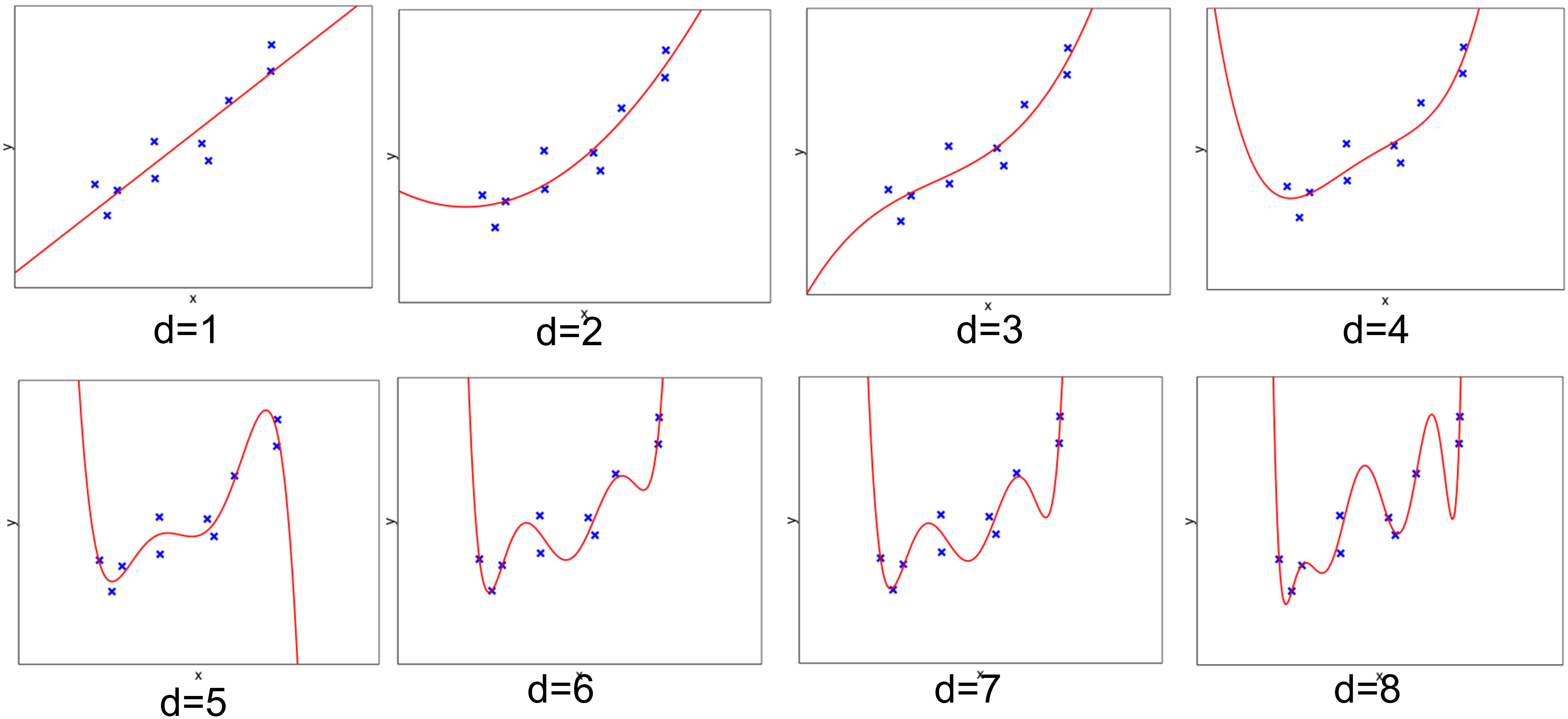
- Every hypothesis has a **true** error measured on all possible data items we could ever encounter (e.g. $f_{\mathbf{w}}(\mathbf{x}_i) - y_i$).

- Since we don't have all possible data, in order to decide what is a good hypothesis, we measure error over the training set.

- Formally: Suppose we compare hypotheses f_1 and f_2.
  - Assume f_1 has lower error on the training set.
  - If f_2 has lower true error, then our algorithm is overfitting.

Overfitting

- Which hypothesis has the lowest **true** error?



Cross-Validation

- Partition your data into a Training Set and a Validation set.
  - The proportions in each set can vary.

- Use the Training Set to find the best hypothesis in the class.

- Use the Validation Set to evaluate the true prediction error.
  - Compare across different hypothesis classes (different order polynominals.)

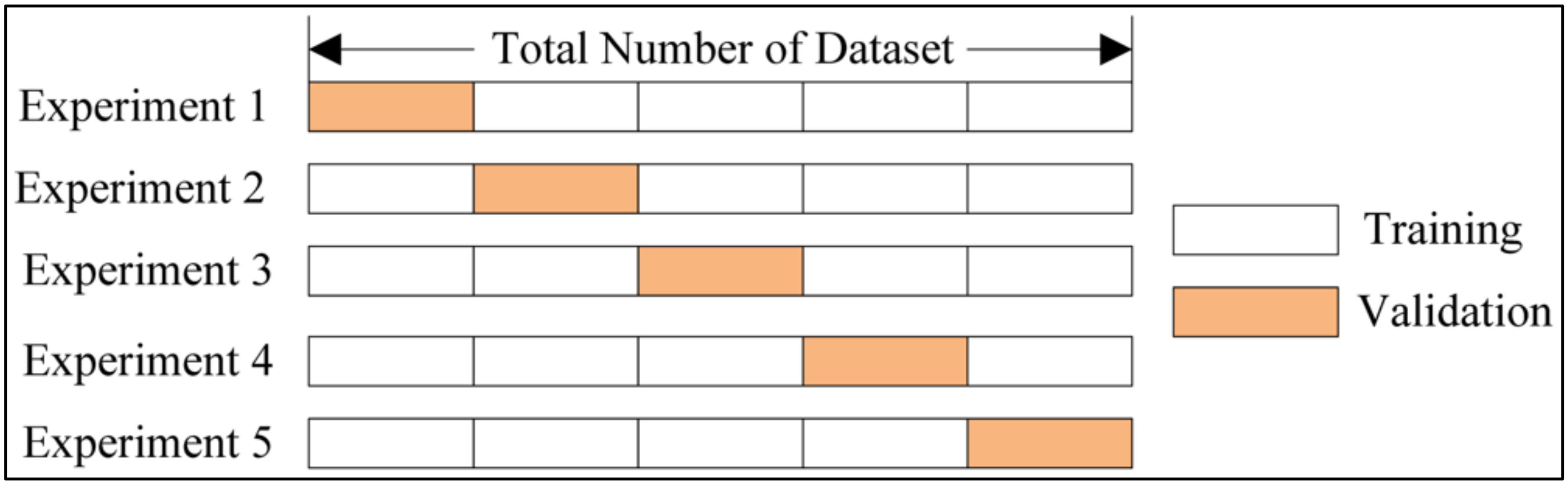
Train:

Validate:

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

k-fold Cross-Validation

- Consider *k* partitions of the data (usually of equal size).
- Train with *k-1* subset, validate on k^{th} subset. Repeat *k* times.
- Average the prediction error over the *k* rounds/folds.



Source: http://stackoverflow.com/questions/31947183/how-to-implement-walk-forward-testing-in-sklearn

- **Computation time is increased** by factor of *k*.

Leave-one-out cross-validation

- Let $k = n$, the size of the training set
- For each order-d hypothesis class,
  - Repeat n times:
    - Set aside *one instance* $<x_i, y_i>$ from the training set.
    - Use all other data points to find $\mathbf{w}$ (optimization).
    - Measure prediction error on the held-out $<x_i, y_i>$.
  - Average the prediction error over all n subsets.
- Choose the d with lowest **estimated true prediction error**.

Estimating true error for *d*=1

Data

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
-1.1	-0.12
0.40	1.81
-0.96	-0.83
0.17	0.43

Cross-validation results

Iter	D_{train}	D_{valid}	$\text{Error}_{\text{train}}$	$\text{Error}_{\text{valid}}$
1	$D - \{(0.86, 2.49)\}$	$(0.86, 2.49)$	0.4928	0.0044
2	$D - \{(0.09, 0.83)\}$	$(0.09, 0.83)$	0.1995	0.1869
3	$D - \{(-0.85, -0.25)\}$	$(-0.85, -0.25)$	0.3461	0.0053
4	$D - \{(0.87, 3.10)\}$	$(0.87, 3.10)$	0.3887	0.8681
5	$D - \{(-0.44, 0.87)\}$	$(-0.44, 0.87)$	0.2128	0.3439
6	$D - \{(-0.43, 0.02)\}$	$(-0.43, 0.02)$	0.1996	0.1567
7	$D - \{(-1.10, -0.12)\}$	$(-1.10, -0.12)$	0.5707	0.7205
8	$D - \{(0.40, 1.81)\}$	$(0.40, 1.81)$	0.2661	0.0203
9	$D - \{(-0.96, -0.83)\}$	$(-0.96, -0.83)$	0.3604	0.2033
10	$D - \{(0.17, 0.43)\}$	$(0.17, 0.43)$	0.2138	1.0490
		mean:	0.2188	0.3558

Cross-validation results

d	$\text{Error}_{\text{train}}$	$\text{Error}_{\text{valid}}$
1	0.2188	0.3558
2	0.1504	0.3095
3	0.1384	0.4764
4	0.1259	1.1770
5	0.0742	1.2828
6	0.0598	1.3896
7	0.0458	38.819
8	0.0000	6097.5

- Optimal choice: *d*=2. Overfitting for *d* > 2.

Evaluation

- We use cross-validation for ***model selection***.

- Available labeled data is split into two parts:
  - **Training set** is used to select a hypothesis *f* from a class of hypotheses *F* (e.g. regression of a given degree).
  - **Validation set** is used to compare the best *f* from each hypothesis class across different classes (e.g. different degree regression).
    - Must be untouched during the process of looking for *f* within a class *F*.

Evaluation

- We use cross-validation for ***model selection***.

- Available labeled data is split into two parts:
  - **Training set** is used to select a hypothesis *f* from a class of hypotheses *F* (e.g. regression of a given degree).
  - **Validation set** is used to compare the best *f* from each hypothesis class across different classes (e.g. different degree regression).
    - Must be untouched during the process of looking for *f* within a class *F*.

- **Test set**: Ideally, a separate set of (labeled) data is withheld to get a true estimate of the generalization error.

  (Often the "validation set" is called "test set", without distinction.)

Validation vs Train error

[From Hastie et al. textbook]

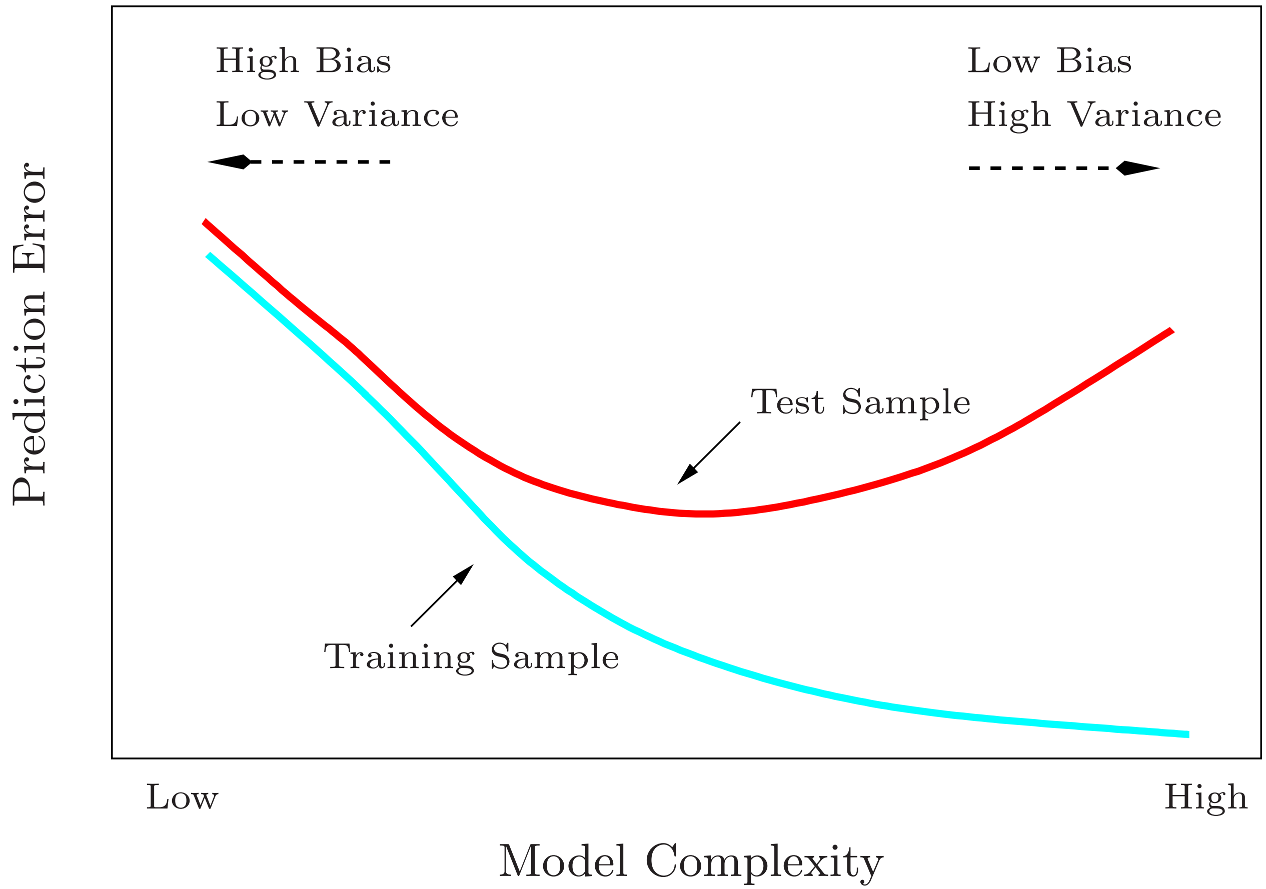


FIGURE 2.11. *Test and training error as a function of model complexity.*

What you should know

- Definition and characteristics of a supervised learning problem.
- Polynomial regression, feature subset selection, ridge, lasso.
- Overfitting (when it happens, how to avoid it).
- Cross-validation (how and why we use it).