Regression and generalization

Machine Learning

Hamid R Rabiee – Zahra Dehghanian

Spring 2025

Sharif University
of Technology

Topics

- Beyond linear regression models
- Evaluation & model selection
- Regularization

Recall: Linear regression (squared loss)

- Linear regression functions

$$f : \mathbb{R} \rightarrow \mathbb{R} \quad f(x; \boldsymbol{w}) = w_0 + w_1 x$$

$$f : \mathbb{R}^{\mathrm{d}} \rightarrow \mathbb{R} \quad f(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 x_1 + \ldots w_d x_d$$

$\boldsymbol{w} = [w_0, w_1, \ldots, w_d]^T$ are the parameters we need to set.

- Minimizing the squared loss for linear regression

$$J(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

- We obtain $\widehat{\boldsymbol{w}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$

Beyond linear regression

- How to extend the linear regression to non-linear functions?
  - Transform the data using basis functions
  - Learn a linear regression on the new feature vectors (obtained by basis functions)

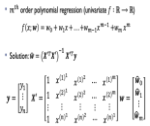
Beyond linear regression

- m^{th} order polynomial regression (univariate $f : \mathbb{R} \longrightarrow \mathbb{R}$)

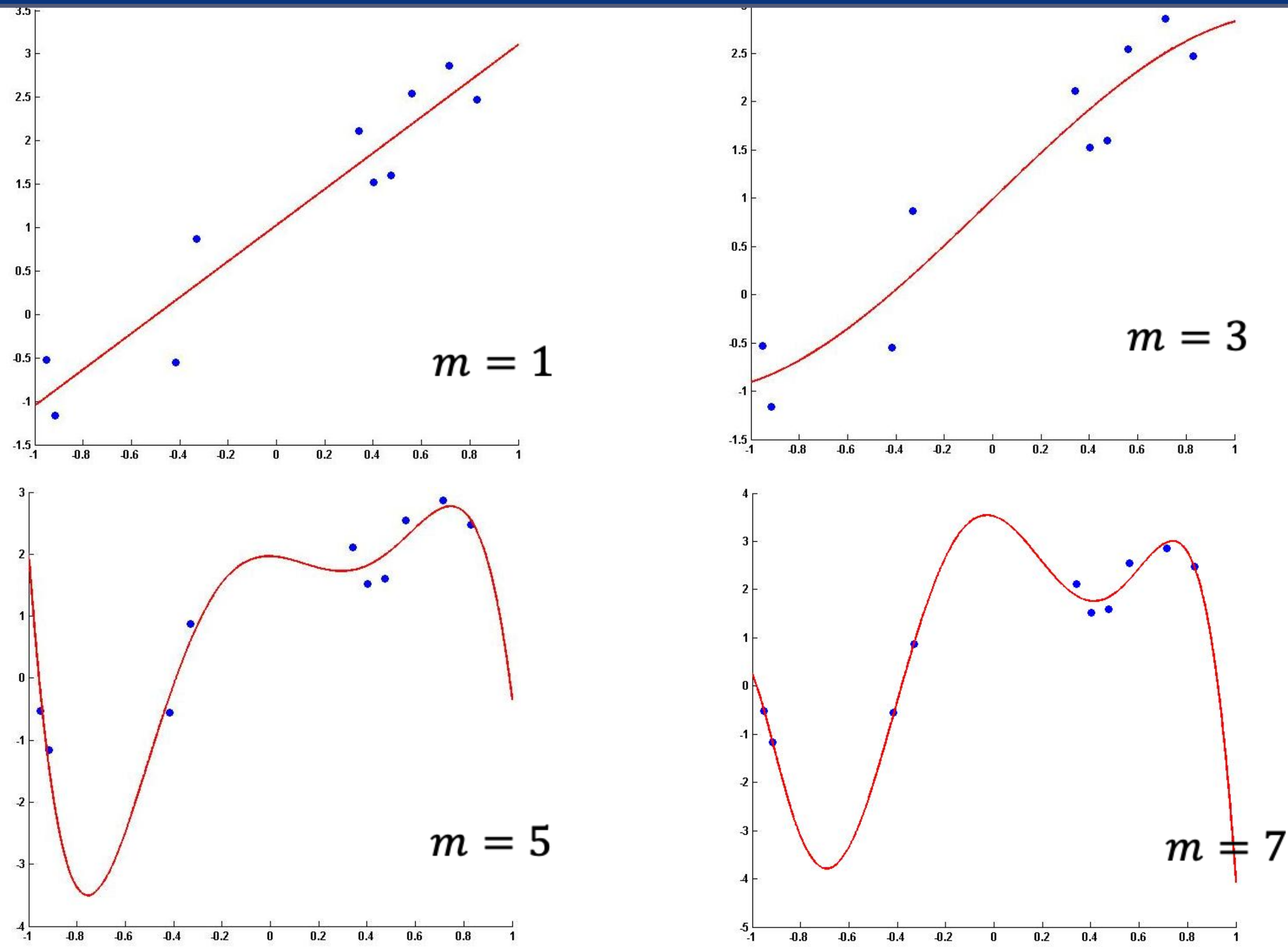
$$f(x; \boldsymbol{w}) = w_0 + w_1 x + \ldots + w_{m-1} x^{m-1} + w_m x^m$$

- Solution: $\widehat{\boldsymbol{w}} = \left(\boldsymbol{X}'^T \boldsymbol{X}'\right)^{-1} \boldsymbol{X}'^T \boldsymbol{y}$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \boldsymbol{X}' = \begin{bmatrix} 1 & x^{(1)^1} & x^{(1)^2} & \cdots & x^{(1)^m} \\ 1 & x^{(2)^1} & x^{(2)^2} & \cdots & x^{(2)^m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x^{(n)^1} & x^{(n)^2} & \cdots & x^{(n)^1} \end{bmatrix} \boldsymbol{w} = \begin{bmatrix} \widehat{\boldsymbol{w}}_0 \\ \widehat{\boldsymbol{w}}_1 \\ \vdots \\ \widehat{\boldsymbol{w}}_m \end{bmatrix}$$



Polynomial regression: example



Generalized linear

- Linear combination of fixed non-linear function of the input vector

$$f(\boldsymbol{x};\boldsymbol{w}) = w_0 + w_1\phi_1(\boldsymbol{x}) + \ldots w_m\phi_m(\boldsymbol{x})$$

$\{\phi_1(\boldsymbol{x}),\ldots,\phi_m(\boldsymbol{x})\}$: set of basis functions (or features)

$$\phi_i(\boldsymbol{x}): \mathbb{R}^d \rightarrow \mathbb{R}$$

Basis functions: examples

- Linear

If $m = d$, $\phi_i(\mathbf{x}) = x_i$, $i = 1, \ldots, d$, then

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_d x_d$$

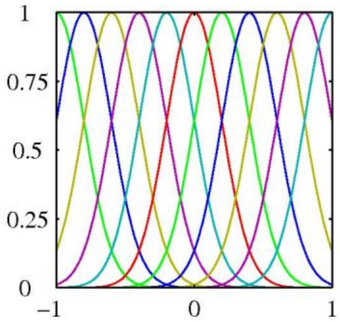
- Polynomial (univariate)

If $\phi_i(x) = x^i$, $i = 1, \ldots, m$, then

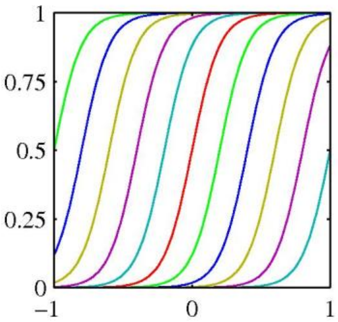
$$f(x; \mathbf{w}) = w_0 + w_1 x + \ldots + w_{m-1} x^{m-1} + w_m x^m$$

Basis functions: examples

- Gaussian: $\phi_j(\boldsymbol{x}) = exp\left\{-\frac{(\boldsymbol{x}-\boldsymbol{c}_j)^2}{2\sigma_j^2}\right\}$



- Sigmoid: $\phi_j(\boldsymbol{x}) = \sigma\left(\frac{\|\boldsymbol{x}-\boldsymbol{c}_j\|}{\sigma_j}\right)$ $\sigma(a) = \frac{1}{1+\exp(-a)}$



Radial Basis Functions: prototypes

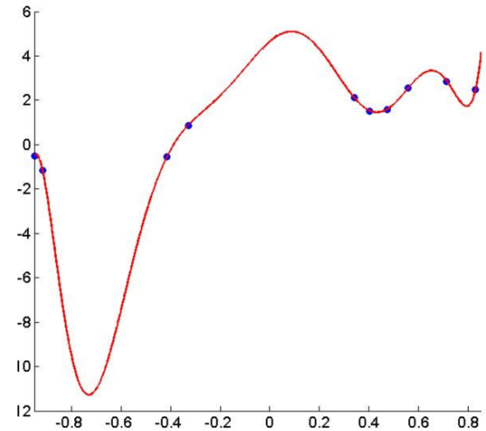
- Predictions based on similarity to "prototypes":

$$\phi_j(\boldsymbol{x}) = exp\left\{-\frac{1}{2\sigma_j^2}\left\|\boldsymbol{x} - \boldsymbol{c}_j\right\|^2\right\}$$

- Measuring the similarity to the prototypes $\boldsymbol{c}_1, \dots, \boldsymbol{c}_m$
  - σ^2 controls how quickly it vanishes as a function of the distance to the prototype.
  - Training examples themselves could serve as prototypes

Model complexity and overfitting

- With limited training data, models may achieve zero training error but a large test error.



Training (empirical) loss

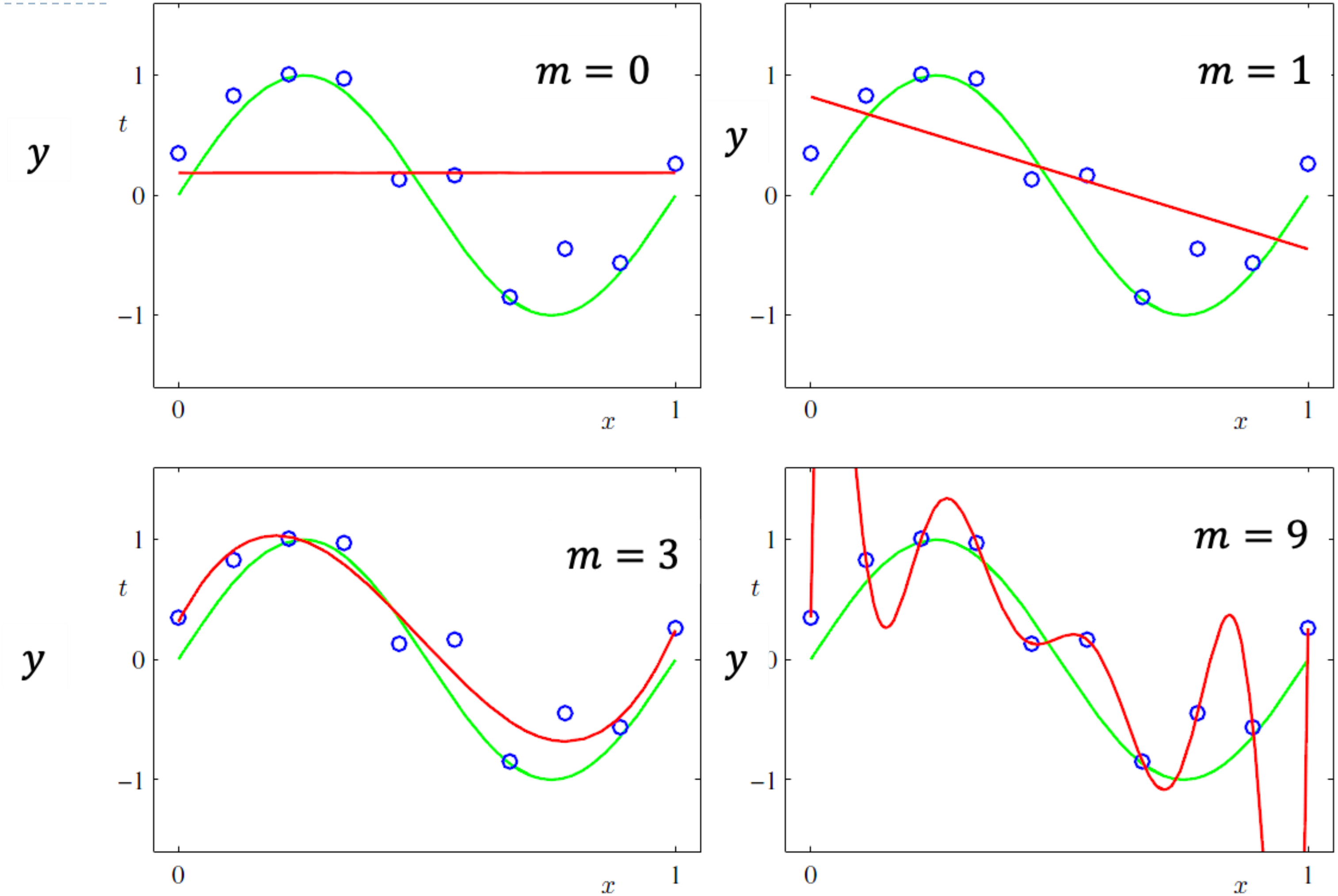
$$\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)\right)^2 \approx 0$$

Expected (true) loss

$$\mathrm{E}_{\mathbf{x},\mathbf{y}}\left\{\left(y - f(\boldsymbol{x};\boldsymbol{\theta})\right)^2\right\} \gg 0$$

- Over-fitting: when the training loss no longer bears any relation to the test (generalization) loss.
  - Fails to generalize to unseen examples.

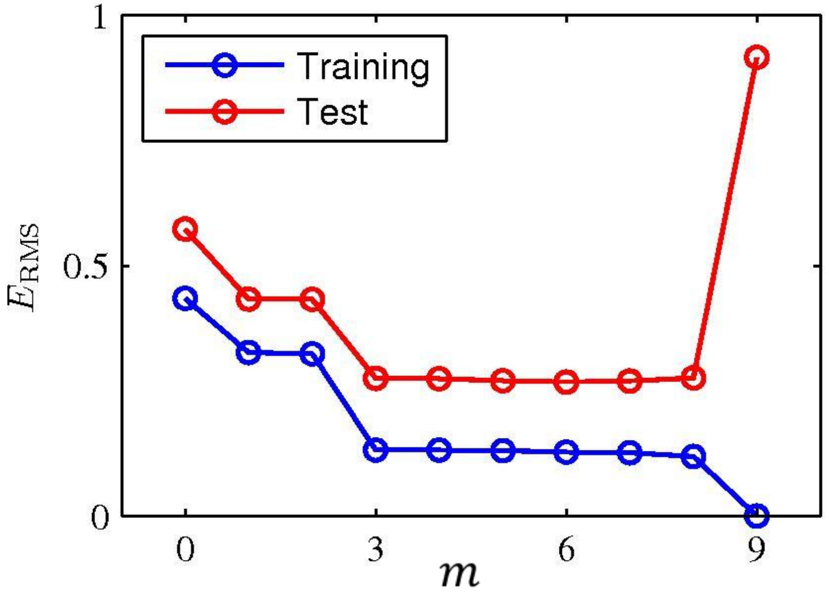
Polynomial regression



[Bishop]

Polynomial regression: training and test error

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(y^{(i)} - f\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)\right)^2}{n}}$$



[Bishop]

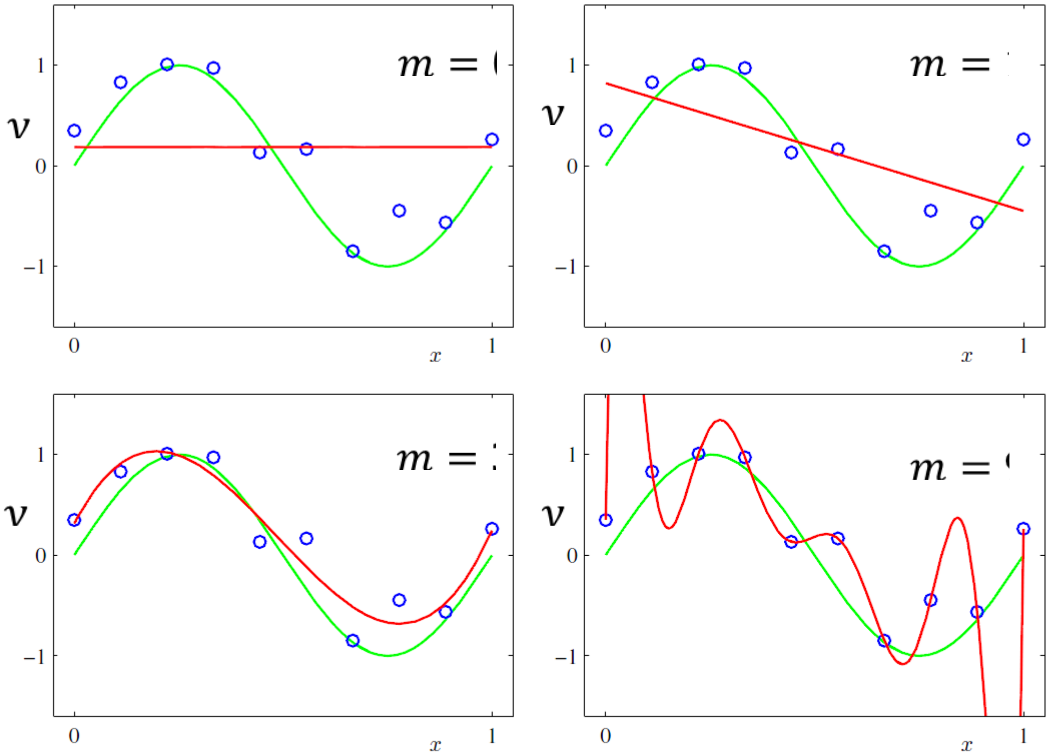
Over-fitting causes

- Model complexity
  - E.g., Model with a large number of parameters (degrees of freedom)

- Low number of training data
  - Small data size compared to the complexity of the model

Model complexity

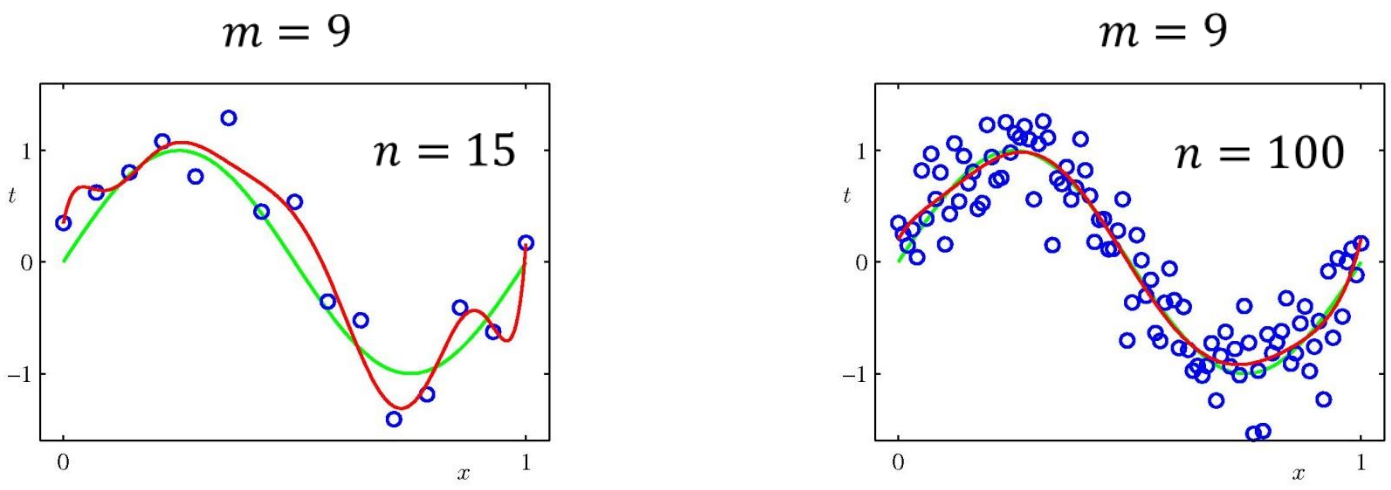
- **Example:**
  - Polynomials with larger m are becoming increasingly tuned to the random noise on the target values.



[Bishop]

Number of training data & overfitting

- Over-fitting problem becomes less severe as the size of training data increases.



[Bishop]

How to evaluate the learner's performance?

- Generalization error: true (or expected) error that we would like to optimize

- Two ways to assess the generalization error are:
  - Practical: Use a separate data set to test the model
  - Theoretical: Law of Large numbers
    - Bias-variance decomposition of out-of-sample error
    - statistical bounds on the difference between training and expected errors

Avoiding over-fitting

- Determine a suitable value for model complexity (Model Selection)
  - **Simple hold-out method**
  - **Cross-validation**
- Regularization (Occam's Razor)
  - Explicit preference towards simple models
  - Penalize for the model complexity in the objective function
- Bayesian approach

Avoiding over-fitting

- Determine a suitable value for model complexity (Model Selection)
  - **Simple hold-out method**
  - **Cross-validation**

- Regularization (Occam's Razor)
  - Explicit preference towards simple models
  - Penalize for the model complexity in the objective function

- Bayesian approach

Evaluation and model selection

- **Evaluation:**
  - We need to measure how well the learned function can predict the target for unseen examples
- **Model selection:**
  - Most of the time we need to select among a set of models
    - Example: polynomials with different degree m
  - and thus we need to evaluate these models first

Model Selection

- **Learning algorithm** defines the data-driven search over the hypothesis space
  - search for good parameters
- **Hyper-parameters** are the tunable aspects of the model, that the learning algorithm does *not* select

This slide has been adopted from CMU ML course:
http://www.cs.cmu.edu/~mgormley/courses/10601-s18/

Model Selection

- **Model selection** is the process by which we choose the "best" model among a set of candidates
  - assume access to a function capable of measuring the quality of a model
  - typically done "outside" the main training algorithm

- Model selection / hyper-parameter optimization is just another form of learning

Simple hold-out: model selection

- Steps:
  - Divide training data into training and validation set v_set
  - Use only the training set to train a set of models
  - Evaluate each learned model on the validation set
    - $J_v(\boldsymbol{w}) = \frac{1}{|v\_set|} \sum_{i \in v\_set} \left(y^{(i)} - f\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right) \right)^2$
  - Choose the best model based on the validation set error

Simple hold-out: model selection

- Steps:
  - Divide training data into training and validation set v_set
  - Use only the training set to train a set of models
  - Evaluate each learned model on the validation set
    - $J_v(\boldsymbol{w}) = \frac{1}{|v\_set|}\sum_{i \in v\_set}\left(y^{(i)} - f\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right)\right)^2$
  - Choose the best model based on the validation set error

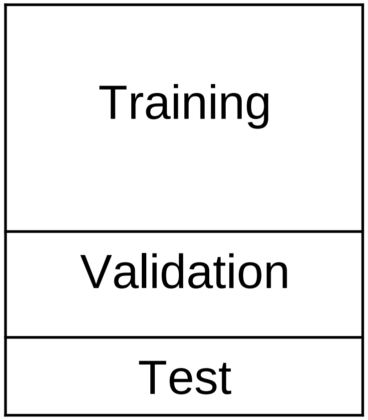
- Usually, too wasteful of valuable training data
  - Training data may be limited.
  - On the other hand, small validation set obtains a relatively noisy estimate of performance.

Simple hold out: training, validation, and test sets

- Simple hold-out chooses the model that minimizes error on validation set.

- $J_v(\hat{\boldsymbol{w}})$ is likely to be an optimistic estimate of generalization error.
  - extra parameter (e.g., degree of polynomial) is fit to this set.

- Estimate generalization error for the test set
  - performance of the selected model is finally evaluated on the test set



Avoiding over-fitting

- Determine a suitable value for model complexity (Model Selection)
  - **Simple hold-out method**
  - **Cross-validation**

- Regularization (Occam's Razor)
  - Explicit preference towards simple models
  - Penalize for the model complexity in the objective function

Regularization

- Adding a penalty term in the cost function to discourage the coefficients from reaching large values.

Regularization

- Adding a penalty term in the cost function to discourage the coefficients from reaching large values.

- Ridge regression (weight decay):

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right) \right)^2 + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

Regularization

- Adding a penalty term in the cost function to discourage the coefficients from reaching large values.

- Ridge regression (weight decay):

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\right)^2 + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

$$\widehat{\boldsymbol{w}} = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{y}$$

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \quad \boldsymbol{\Phi} = \begin{bmatrix} 1 & \phi_1(\boldsymbol{x}^{(1)}) & \cdots & \phi_m(\boldsymbol{x}^{(1)}) \\ 1 & \phi_1(\boldsymbol{x}^{(2)}) & \cdots & \phi_m(\boldsymbol{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(\boldsymbol{x}^{(n)}) & \cdots & \phi_m(\boldsymbol{x}^{(n)}) \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$$

Regularization

- Adding a penalty term in the cost function to discourage the coefficients from reaching large values.

- Ridge regression (weight decay):

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}^T\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\right)^2 + \lambda \boldsymbol{w}^T\boldsymbol{w}$$

$$\widehat{\boldsymbol{w}} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}^T\boldsymbol{y}$$

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \boldsymbol{\Phi} = \begin{bmatrix} 1 & \phi_1(\boldsymbol{x}^{(1)}) & \cdots & \phi_m(\boldsymbol{x}^{(1)}) \\ 1 & \phi_1(\boldsymbol{x}^{(2)}) & \cdots & \phi_m(\boldsymbol{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(\boldsymbol{x}^{(n)}) & \cdots & \phi_m(\boldsymbol{x}^{(n)}) \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$$

Polynomial order

- Polynomials with larger m are becoming increasingly tuned to the random noise on the target values.
  - magnitude of the coefficients typically gets larger by increasing m.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	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

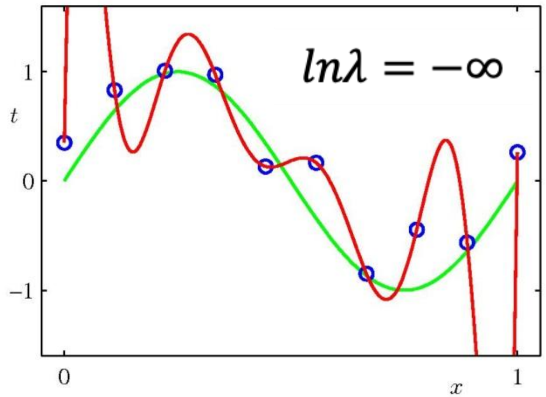
[Bishop]

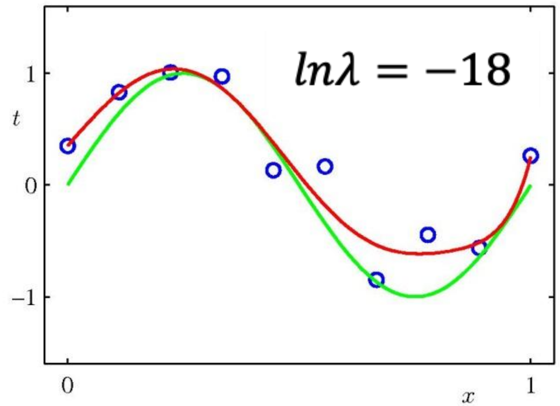
Regularization parameter

$m = 9$

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$\widehat{w}_0$	0.35	0.35	0.13
$\widehat{w}_1$	232.37	4.74	-0.05
$\widehat{w}_2$	-5321.83	-0.77	-0.06
$\widehat{w}_3$	48568.31	-31.97	-0.05
$\widehat{w}_4$	-231639.30	-3.89	-0.03
$\widehat{w}_5$	640042.26	55.28	-0.02
$\widehat{w}_6$	-1061800.52	41.32	-0.01
$\widehat{w}_7$	1042400.18	-45.95	-0.00
$\widehat{w}_8$	-557682.99	-91.53	0.00
$\widehat{w}_9$	125201.43	72.68	0.01

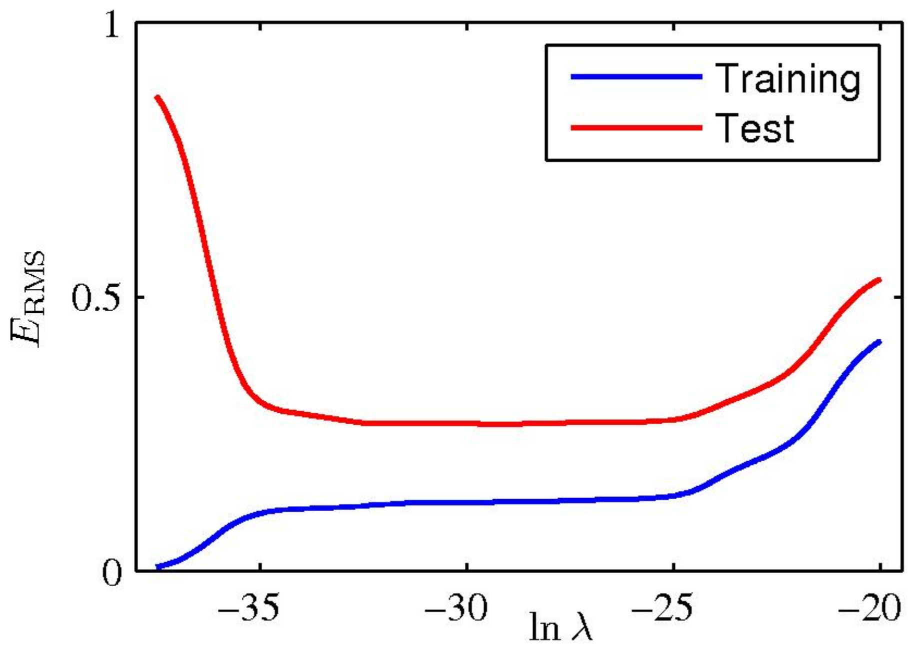
[Bishop]





Regularization parameter

- Generalization
  - λ now controls the effective complexity of the model and hence determines the degree of over-fitting



[Bishop]

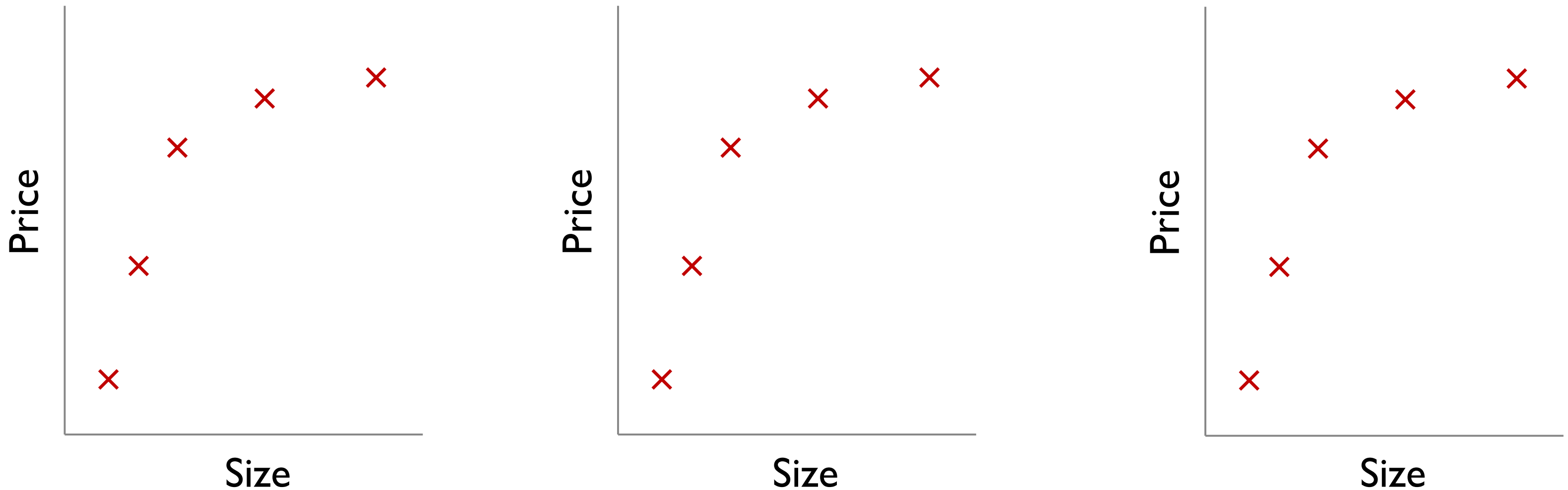
Choosing the regularization parameter

- A set of models with different values of λ.
- Find $\hat{\boldsymbol{w}}$ for each model based on training data
- Find $J_v(\hat{\boldsymbol{w}})$ (or $J_{cv}(\hat{\boldsymbol{w}})$) for each model
  - $J_v(\boldsymbol{w}) = \frac{1}{n\_v} \sum_{i \in v\_set} \left(y^{(i)} - f\left(x^{(i)}; \boldsymbol{w}\right)\right)^2$
- Select the model with the best $J_v(\hat{\boldsymbol{w}})$ (or $J_{cv}(\hat{\boldsymbol{w}})$)

The approximation-generalization trade-off

- Small true error shows good approximation of f out of sample
- More complex $\mathcal{H} \Rightarrow$ better chance of approximating f
- Less complex $\mathcal{H} \Rightarrow$ better chance of generalization out of f

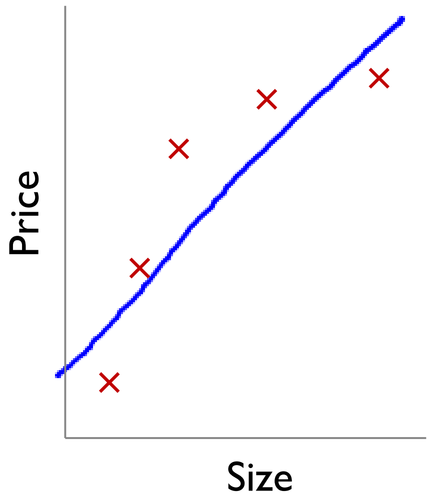
Complexity of Hypothesis Space: Example



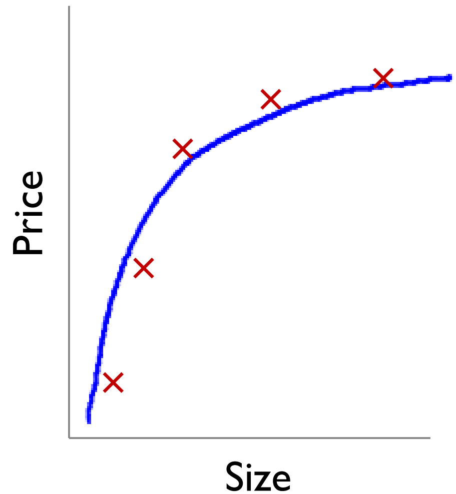
$w_0 + w_1 x$ $\qquad$ $w_0 + w_1 x + w_2 x^2$ $\qquad$ $w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

Less complex $\mathcal{H}$ $\qquad$ More complex $\mathcal{H}$

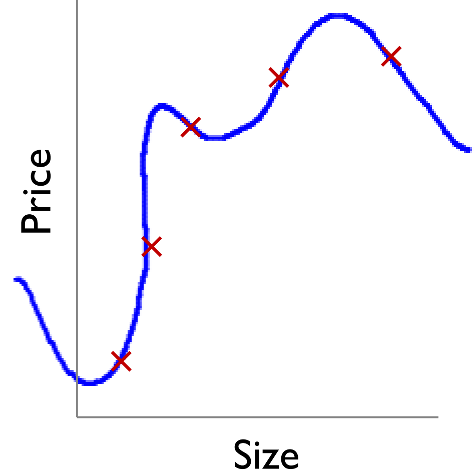
Complexity of Hypothesis Space: Example



$w_0 + w_1 x$



$w_0 + w_1 x + w_2 x^2$



$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

Underfitting

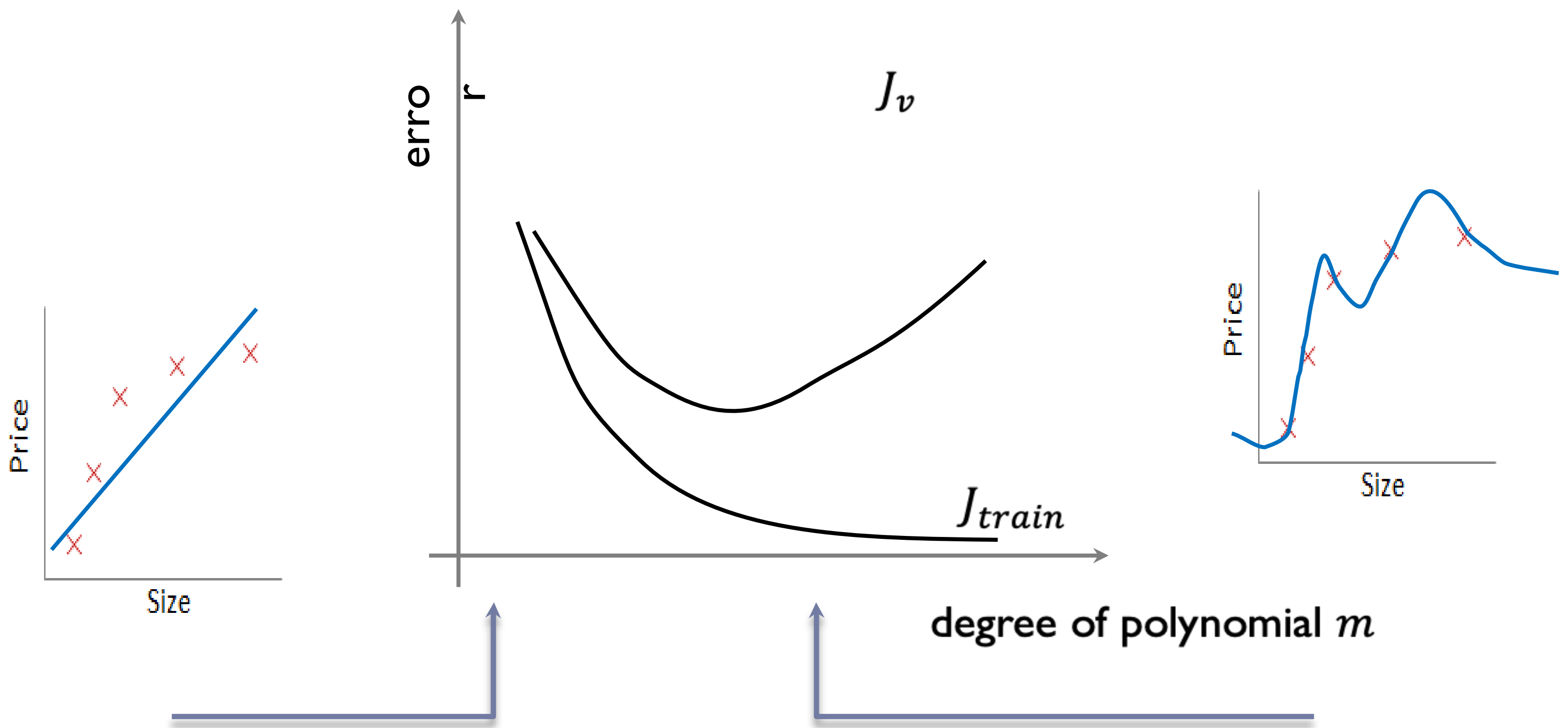
Overfitting

This example has been adapted from: Prof. Andrew Ng's slides

Complexity of Hypothesis Space: Example

$$J_v(\mathbf{w}) = \frac{1}{n\_v} \sum_{i \in val\_set} \left(y^{(i)} - f\left(\mathbf{x}^{(i)}; \mathbf{w}\right)\right)^2$$

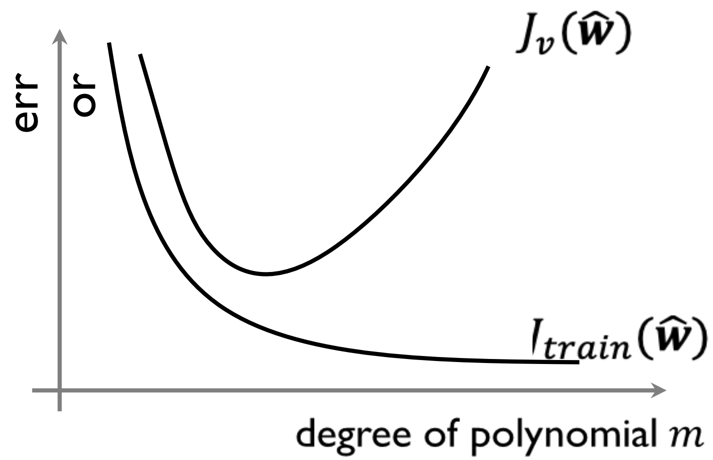
$$J_{train}(\mathbf{w}) = \frac{1}{n\_train} \sum_{i \in train\_set} \left(y^{(i)} - f\left(\mathbf{x}^{(i)}; \mathbf{w}\right)\right)^2$$



Complexity of Hypothesis Space

- Less complex $\mathcal{H}$:
  - $J_{train}(\widehat{\boldsymbol{w}}) \approx J_v(\widehat{\boldsymbol{w}})$ and $J_{train}(\widehat{\boldsymbol{w}})$ is very high

- More complex $\mathcal{H}$:
  - $J_{train}(\widehat{\boldsymbol{w}}) \ll J_v(\widehat{\boldsymbol{w}})$ and $J_{train}(\widehat{\boldsymbol{w}})$ is low

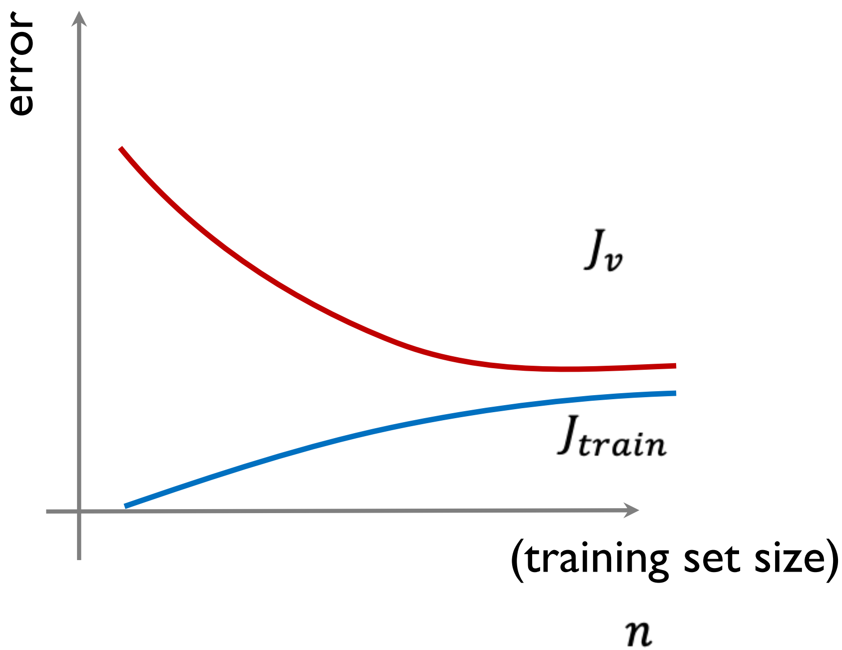


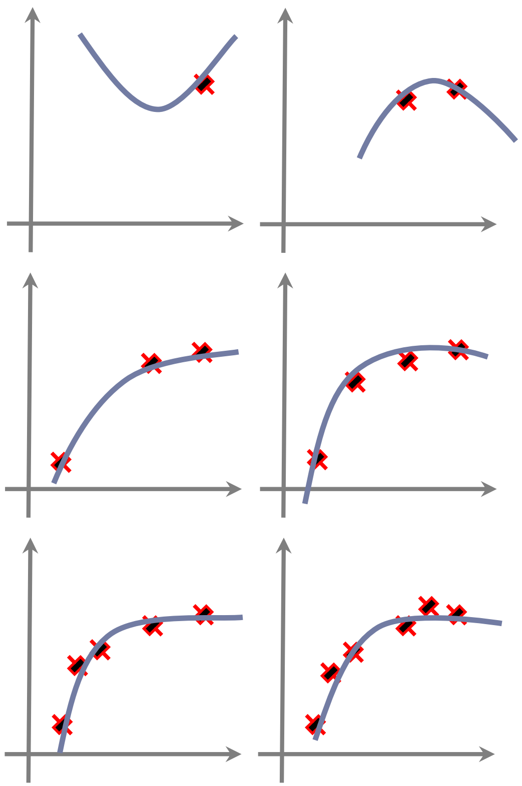
Size of training set

$$J_v(\boldsymbol{w}) = \frac{1}{n\_v}\sum_{i \in val\_set}\left(y^{(i)} - f\left(x^{(i)};\boldsymbol{w}\right)\right)^2$$

$$J_{train}(\boldsymbol{w}) = \frac{1}{n\_train}\sum_{i \in train\_set}\left(y^{(i)} - f\left(x^{(i)};\boldsymbol{w}\right)\right)^2$$

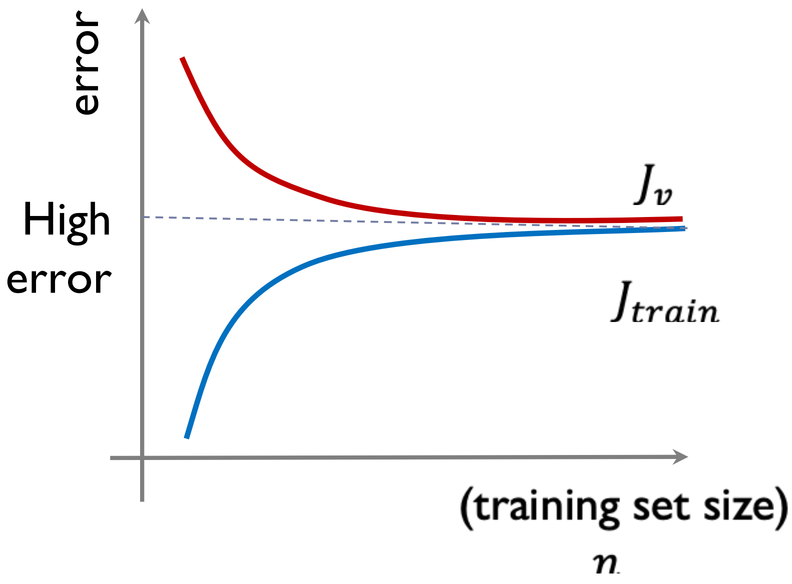
$$f(x;\boldsymbol{w}) = w_0 + w_1 x + w_2 x^2$$

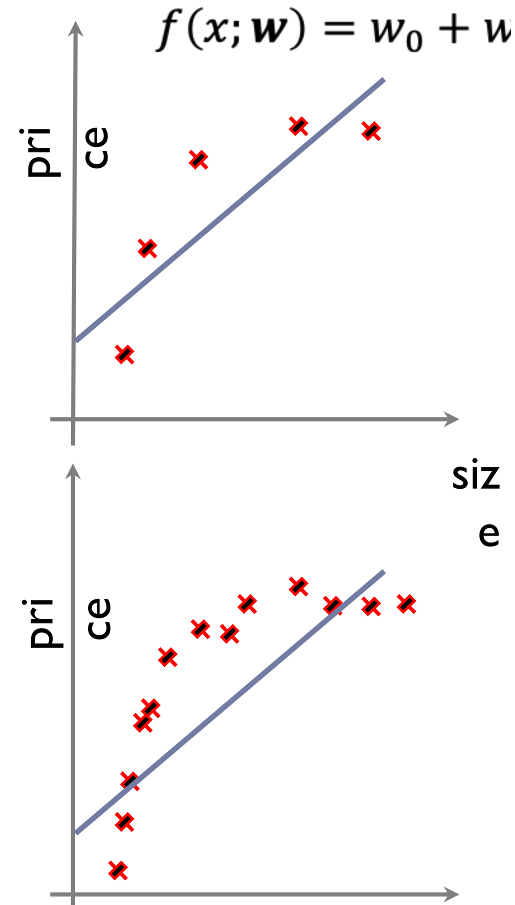




This slide has been adapted from: Prof. Andrew Ng's slides

Less complex $\mathcal{H}$

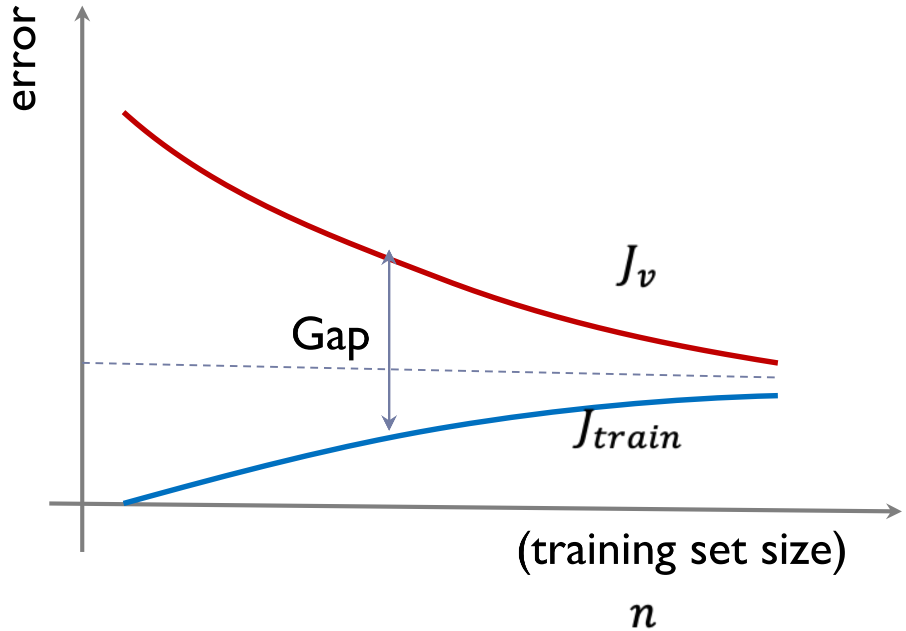


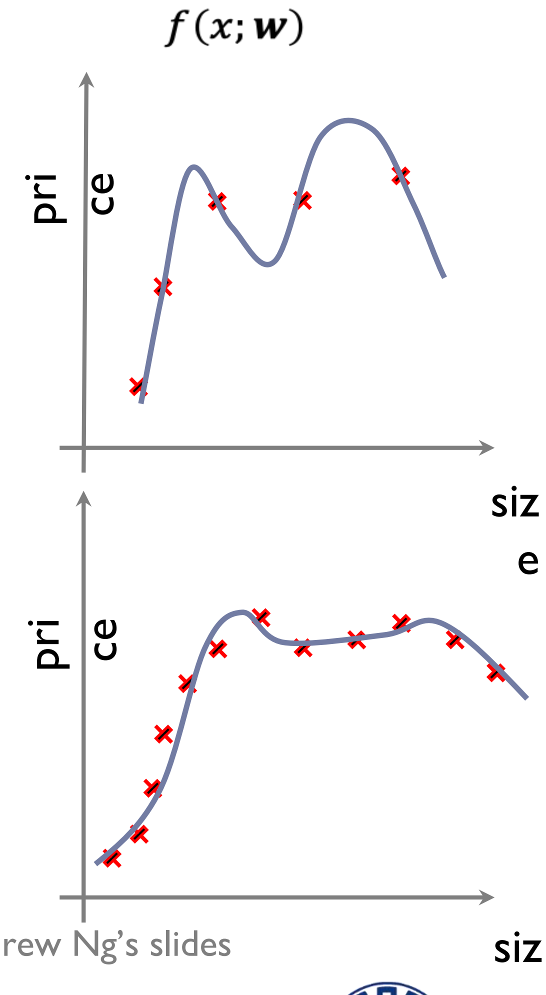


If model is very simple, getting more training data will not (by itself) help much.

This slide has been adapted from: Prof. Andrew Ng's slides

More complex $\mathcal{H}$





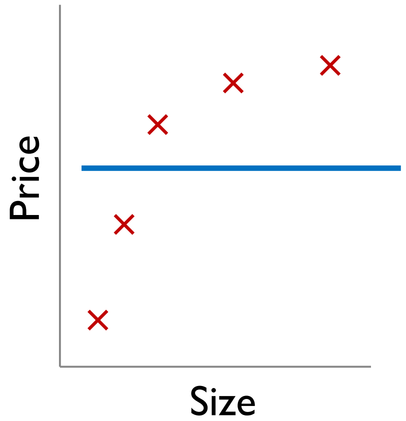
For more complex models, getting more training data is usually helps.

This slide has been adapted from: Prof. Andrew Ng's slides

Regularization: Example

$$f(x;\boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$

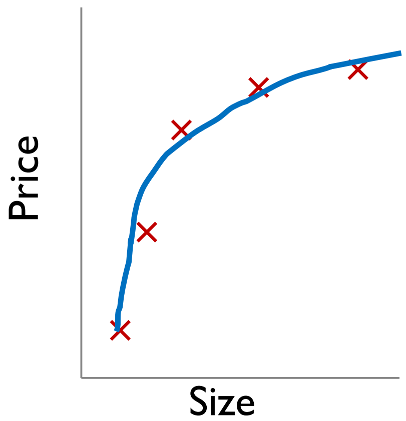
$$J(\boldsymbol{w}) = \frac{1}{n}\left(\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)};\boldsymbol{w}\right)\right)^2 + \lambda \boldsymbol{w}^T \boldsymbol{w}\right)$$



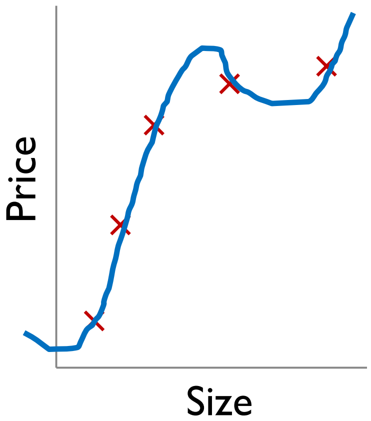
Large λ

(Prefer to more simple models)

$w_1 = w_2 \approx 0$



Intermediate λ



Small λ

(Prefer to more complex models)

$\lambda = 0$

This example has been adapted from: Prof. Andrew Ng's slides

Resources

- C. Bishop, “Pattern Recognition and Machine Learning”, Chapter 1.1,1.3, 3.1
- Course CE-717, Dr. M.Soleymani
- CMU ML course: http:// www.cs.cmu.edu /~mgormley /courses /10601-s18/