Generalization

CE-477: Machine Learning - CS-828: Theory of Machine Learning
Sharif University of Technology
Fall 2024

Fatemeh Seyyedsalehi

References of the lecture are mentioned in the last slide

Where are we now?

- The learning problem? What is learning?
- Basic supervised learning models How to do it?
  - Linear regression
  - Linear and probabilistic classifiers
  - kernels
- Generalization and regularization Can we learn?
- Computational learning theory Can we learn?
- Supervised learning How to do it? Paradigms in machine learning
  - SVM
  - Neural nets
  - Decision trees
  - Instance based learning
  - Ensemble learning
- Unsupervised learning
  - Clustering – EM – GMM
  - Dimensionality reduction
- Reinforcement Learning
- Interpretability What did we learn?

Where are we now?

- We will attempt to understand the generalization of the learning models, i.e. their performance on unseen data.

- Up to now, we have found the model $h_w(\boldsymbol{x})$, which minimizes a training cost function $J(\boldsymbol{w})$ over the training set, $\mathrm{D} = \{(\boldsymbol{x}^i, \boldsymbol{y}^i), i = 1, \dots, n\}$.
  - For example, the SSE cost function:

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

Where are we now?

- However, it is not our main goal.
- In fact it is our approach towards the goal of learning a predictive model.
- The most important evaluation metric is the model performance on unseen test example, which is called the test error.
- We sample a test example from the data distribution, $p(x, y)$ and measure the expected model's error.
  - Example, expected of the SSE error,
  $$J(\boldsymbol{w}) = \mathrm{E}_{p(x,y)}[(y - h_w(x))^2]$$
  - Can be approximated by the average error on many sampled test examples

Where are we now?

- In classical statistical learning the training set is drawn from the same distribution as the test distribution $p(x, y)$.
  - The key difference between the training set and test examples are that the test examples are unseen during the training procedure.

- Therefore, the test error is not necessarily close to the training error.

Where are we now?

- Overfitting: The model predicts accurately on the training dataset but doesn't generalize well to other test examples, that is, if the training error is small but the test error is large.

- Underfitting: The training error is relatively large, in this case, typically the test error is also relatively large.

In this chapter …

- We investigate how the test error is influenced by the learning procedure especially the choice of model parameterizations

- We decompose the test error into bias and variance terms
  - How each of them are affected by the model parameterizations
  - Their tradeoffs
  - Overfitting and underfitting situations

- Model selection and regularization

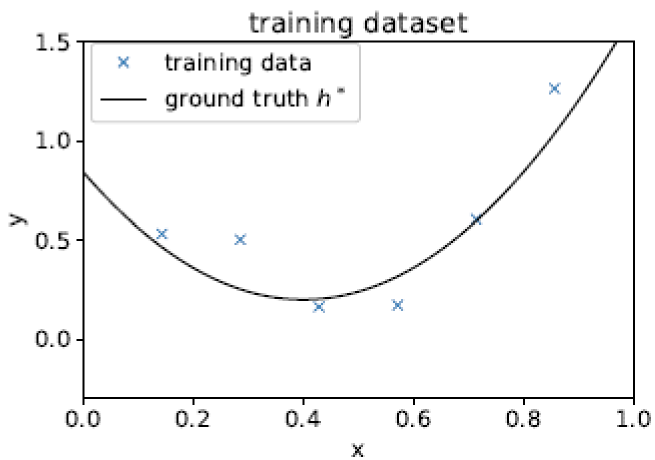
Bias-variance tradeoff
Example

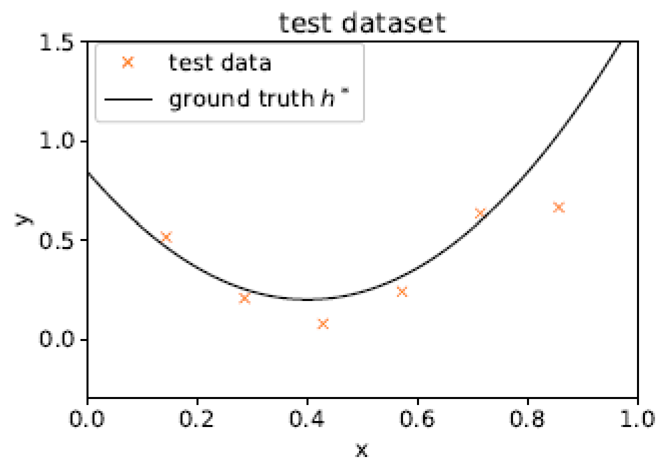
- As an example consider the following target function,

$$y = h_w^t(x) + \varepsilon$$

where $h_w^t(x)$ is a quadratic form and ε is the noise observation.

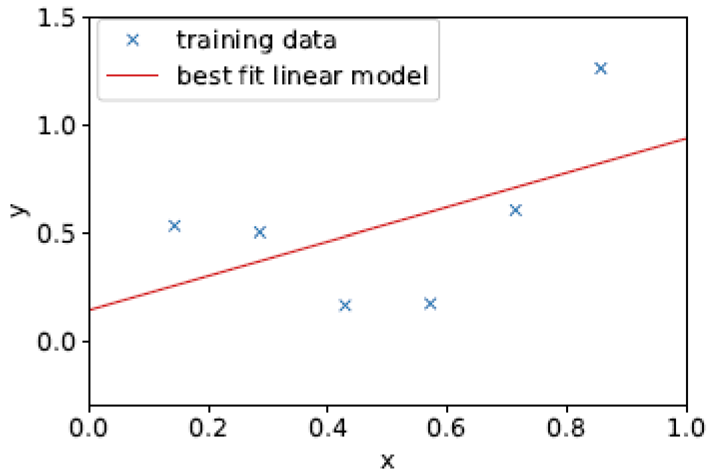
- It is impossible to predict the noise ε and our goal is to recover the function $h_w^t(x)$.

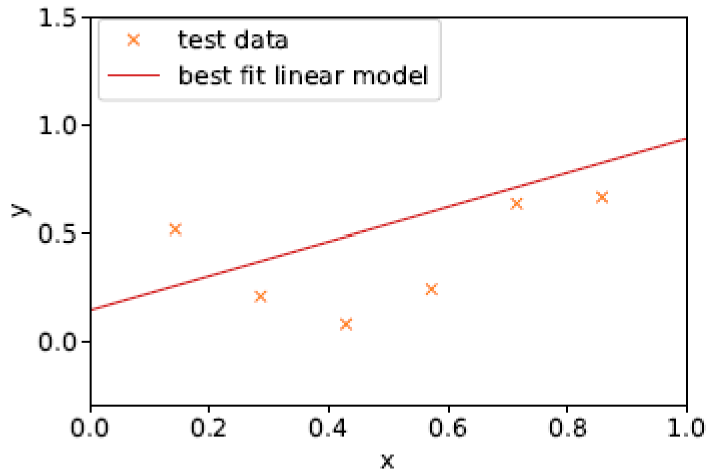




Bias-variance tradeoff
Example

- The hypothesis space of “linear models”

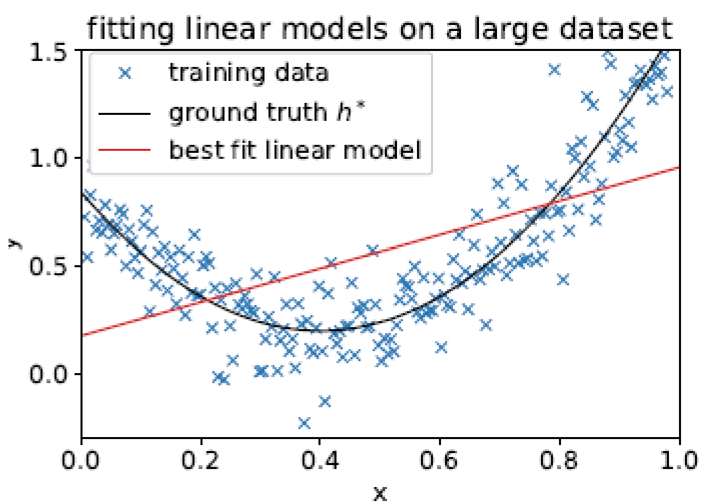


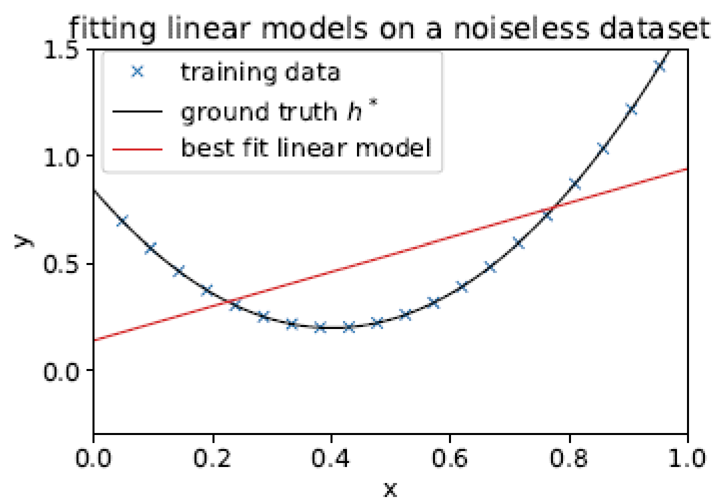


- The best linear model has large training and test errors
  - Underfitting

Bias-variance tradeoff Example

- The hypothesis space of “linear models”
- Even with a very large amount of, or even infinite training examples, the best fitted linear model is still inaccurate and fails to capture the structure of the data.
  - This issue still occurs even without the presence of the noise





Bias-variance tradeoff
Example

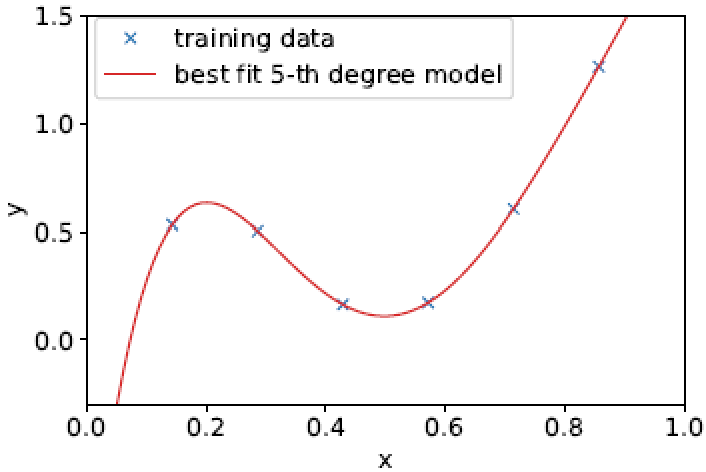
- The main problem is that linear models can not represent a quadratic function $h_w^t(x)$.

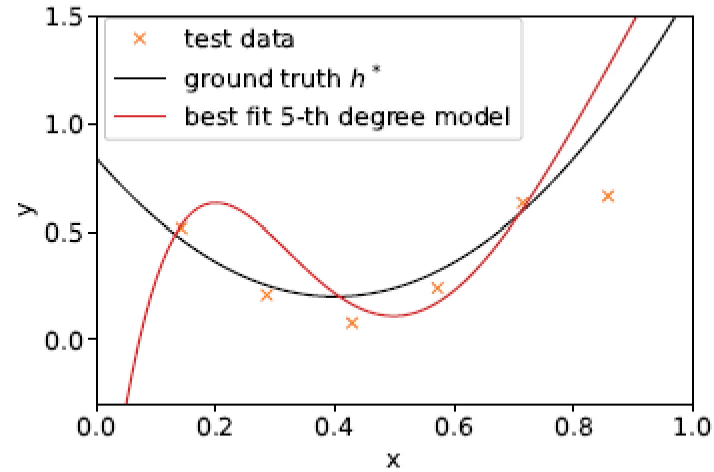
- Informally, we define the **bias** of a model to be the test error when we train the model with a very large (infinite) training dataset.

- In this example, the linear model suffers from large **bias**, and underfits, i.e. fails to capture the structure exhibited by the data.

Bias-variance tradeoff
Example

- The hypothesis space of "5th degree polynomial"

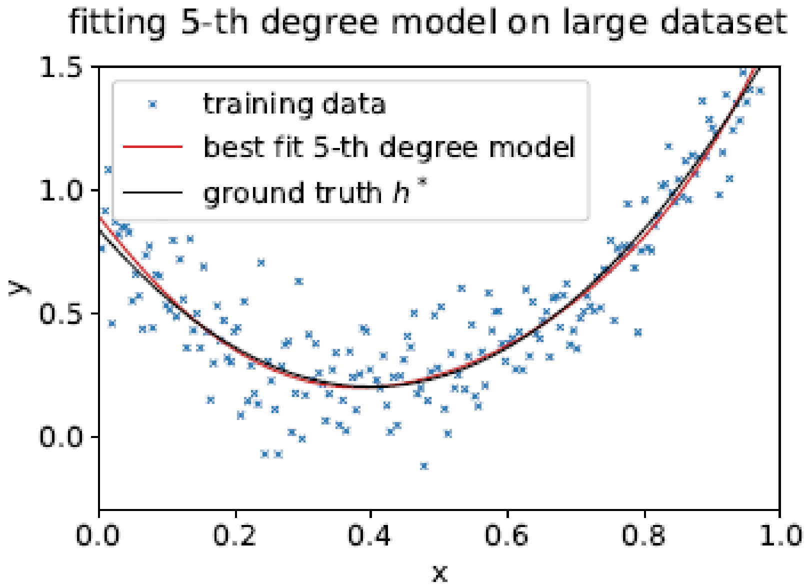




- The model learnt from the training set does not generalize well to test examples.
- Best 5^{th} degree polynomial has zero training error, but still has a large test error
  - Overtting

Bias-variance tradeoff
Example

- The hypothesis space of "5th degree polynomial"
- Fitting to an extremely large dataset, the resulting model would be close to a quadratic function and be accurate.
  - This is because the family of 5th degree polynomials contains all the quadratic functions and capable of capturing the structure of the data.



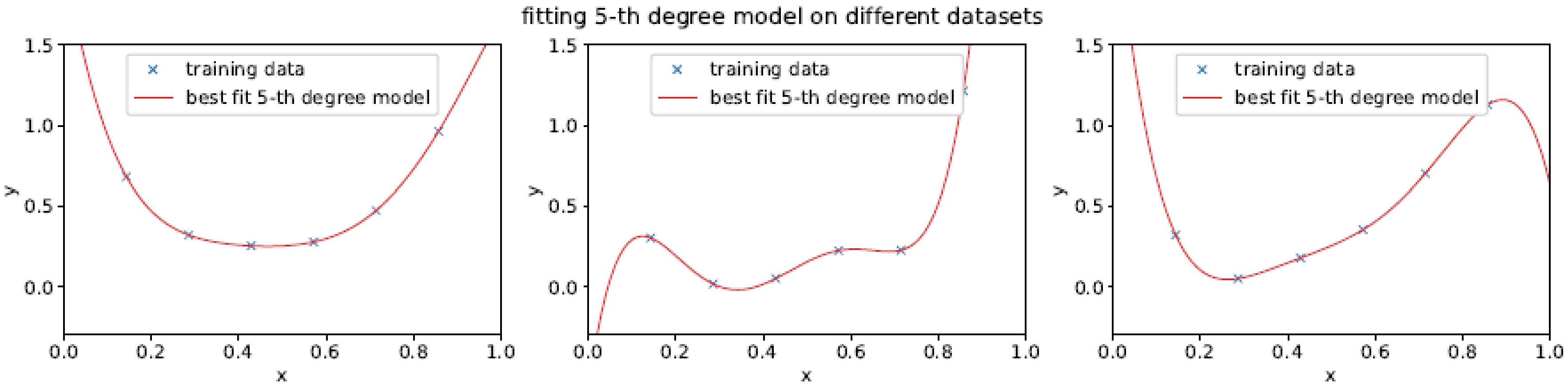
Bias-variance tradeoff Example

- The failure of fitting 5^{th} degree polynomials has different reason from failure of linear models.
  - The best 5^{th} degree polynomial on a huge dataset nearly recovers the ground-truth

- Their failure can be captured by an other component of the error called **variance.**

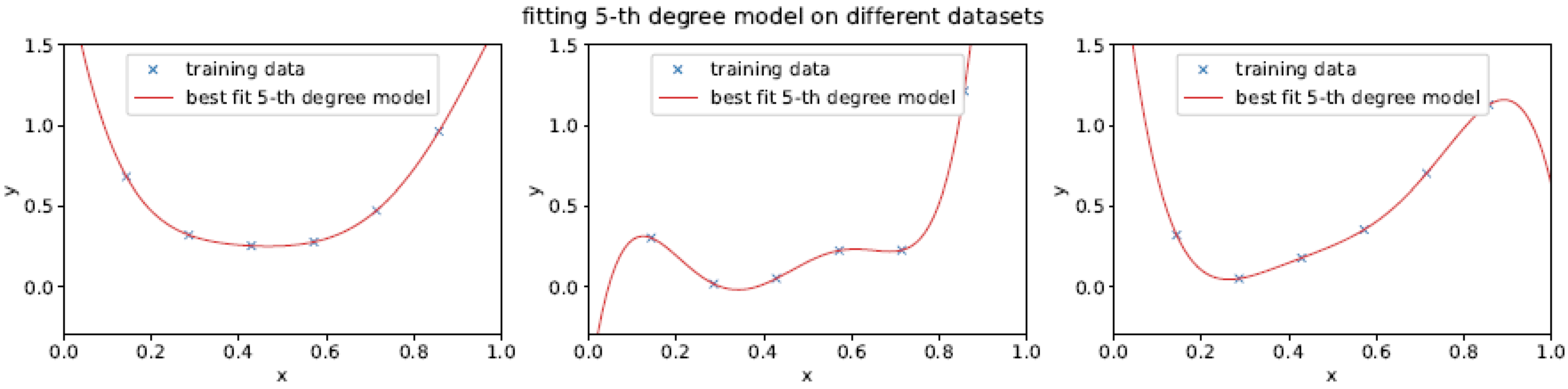
Bias-variance tradeoff Example

- The best 5^{th} degree models on three different datasets generated from the same distribution behave quite differently, suggesting the existence of a large **variance.**



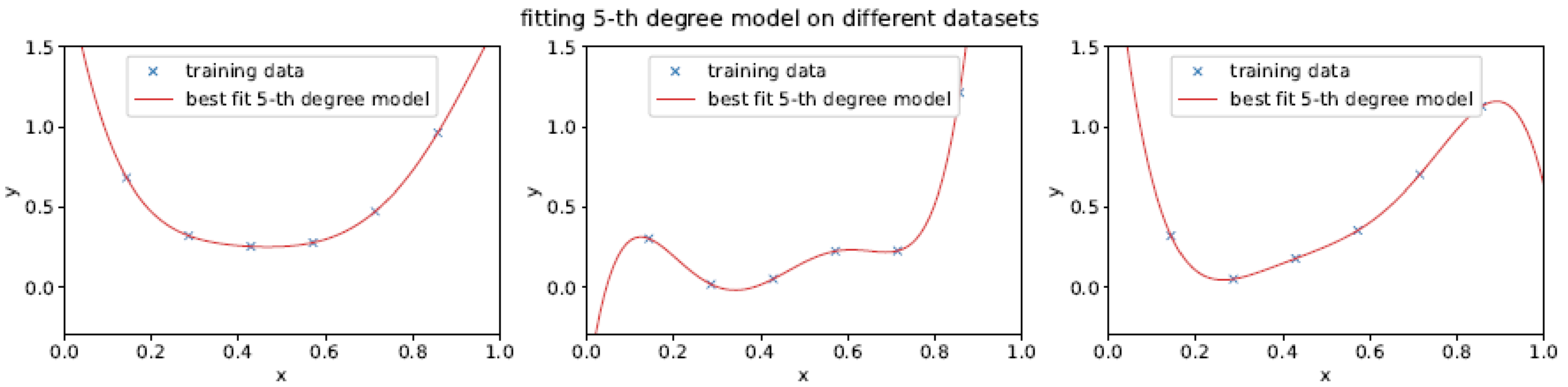
Bias-variance tradeoff
Example

- In this case, we fit patterns in the data that are present in our small, finite training set and do not reflect the wider pattern of the relationship between the input and outputs.
  - Spurious patterns resulted from noise



Bias-variance tradeoff
Example

- The **variance**: the amount of variations across models learnt on multiple different training datasets (drawn from the same underlying distribution).



Bias-variance tradeoff

- Often, there is a tradeoff between bias and variance.
  - Model is too simple and has very few parameters: it may have large bias (but small variance), and it typically may suffer from underfittng.
  - Model is too complex and has very many parameters: it may suffer from large variance (but have smaller bias), and thus overfitting

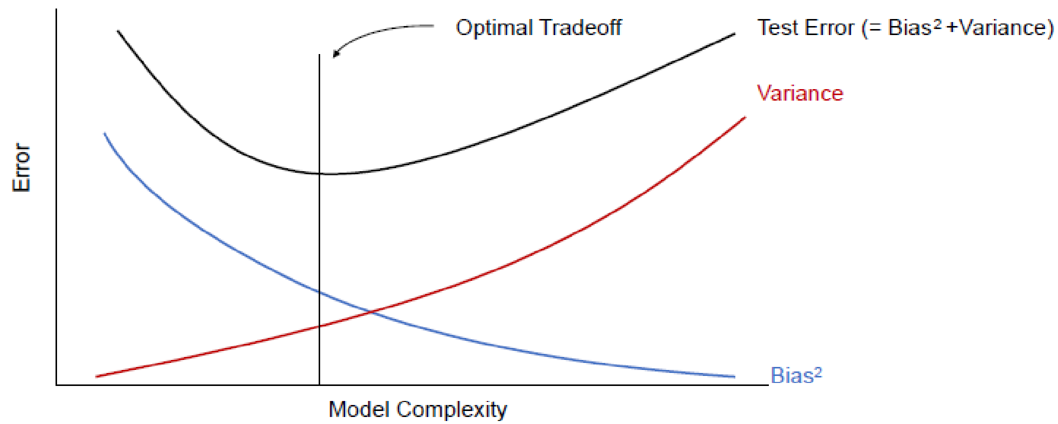
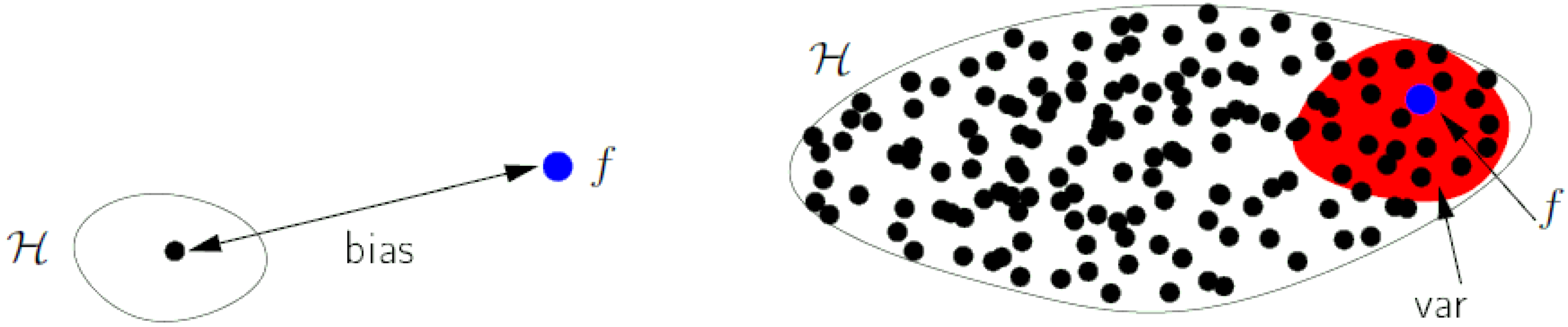


Figure 8.8: An illustration of the typical bias-variance tradeoff.

Bias-variance decomposition



Regularization

- Overfitting: a result of using too complex models
- We need to choose a proper model complexity to achieve the optimal bias-variance tradeoff

- However, the correct, informative complexity measure of the models can be **a function of the parameters** which may not necessarily depend on **the number of parameters.**

- **Regularization**: an important technique in machine learning, control the model complexity and prevent overfitting.

Regularization

- Adding a term, which is called a regularizer to the training cost function:

$$J_\lambda(w) = J(w) + \lambda R(w)$$

  - $J_\lambda(w)$: Regularized loss
  - λ: Regularization parameter
  - $R(w)$: The regularizer, in classical method it is only the function of parameters

Regularization

- Adding a term, which is called a regularizer to the training cost function:

$$J_\lambda(w) = J(w) + \lambda R(w)$$

- The regularizer function is typically chosen to be a measure of the complexity of the model
- A model which trained by the regularized loss $J_\lambda(w)$, both fit the training data (a small loss $J(w)$) and have a small complexity (a small $R(w)$).
  - Balance with parameter λ

Regularization

- The most common regularization: ℓ_2 norm

$$R(w) = \frac{1}{2}\ ||w||_2^2$$

- It encourages the optimizer to find a model with small ℓ_2 norm

- Remember form the probabilistic view of learning:
  - Equivalent to considering a Normal prior on parameters and using the map estimation for the regression perobleme

Regularization

- The ℓ_2 norm as regularizer:
  - In deep learning we refer to it as weight decay

$$w \leftarrow w - \eta \nabla J(w) = w - \eta \lambda w - \eta \nabla J(w)$$
$$= (1 - \eta\lambda) w - \eta \nabla J(w)$$

In gradient descent technique using a regularized loss leads to decaying w by a scalar factor $(1 - \eta\lambda)$ and then applying the standard gradient.

Regularization

- Can impose structure on the model parameters when we have a prior belief about the parameters

- For example, when we know most of the parameters are zero, we can use the ℓ_1 norm to impose the sparsity.
  - Remember from the probabilistic view of learning chapter, when we use a Laplace prior for parameters.

- Imposing additional structure of the parameters narrows our search space and make the complexity of our model smaller
  - For example, the family of sparse model is smaller than the family of all models.

Regularization

- However, we should be careful that imposing additional structure may increase the risk of high bias.

- It may leads to a very small hypothesis space with a large bias
  - For example considering a large value for the regularization parameter in the ℓ_1 norm regularization
  - Leads to search only in the family of very sparse models and none of them may not be able to perform the prediction task
    - Similar the situation we use the linear models for data generated from a quadratic function.

$$J_\lambda(w) = J(w) + \lambda|w|$$

Regularization

- The ℓ_2 norm regularization is much more commonly used with kernel methods because ℓ_1 norm regularization is typically not compatible with the kernel trick.
  - The optimal solution cannot be written as functions of inner products of features.

- The ℓ_2 norm is also the most common regularizer in deep learning (another example dropout technique).

Implicit regularization effect

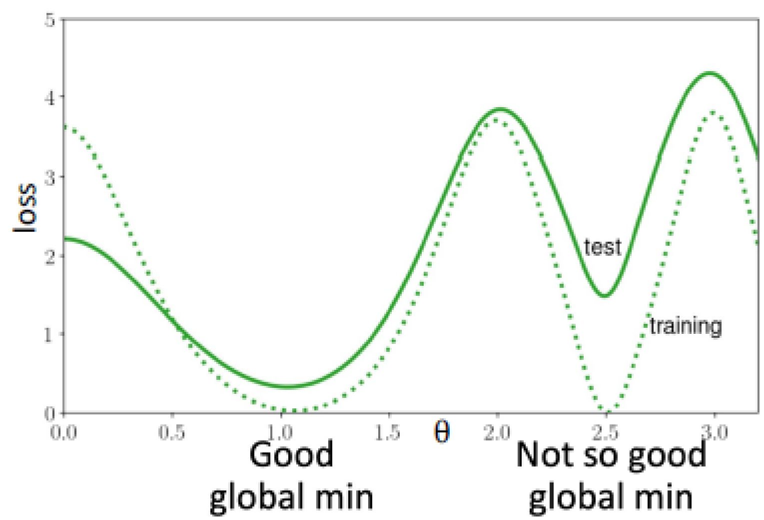
- Optimizers
  - Are learning algorithms which search the hypothesis space to find the best model

- Optimizers attempt to find a model with the least training cost function.

- The way that an optimizer discovers the space may help to find the solutions with less test error.

Implicit regularization effect

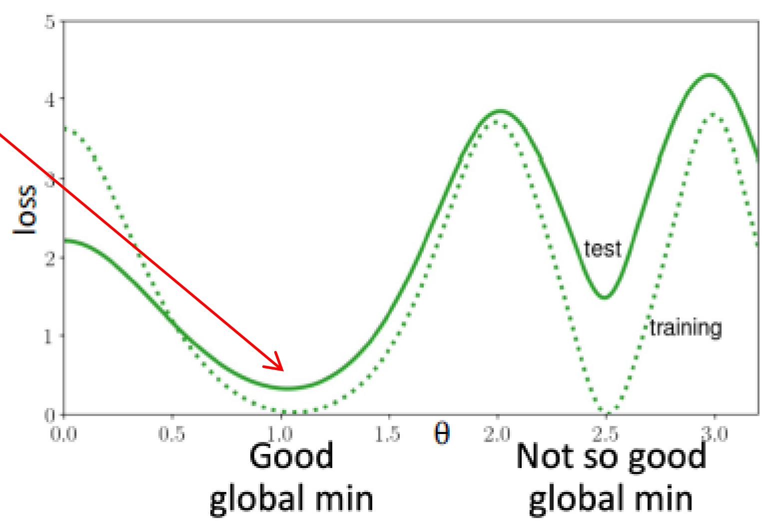
- We may have more than one global optimum with the same value of cost function.
- However the test error (generalization error or true error) may be different for them.
- Different optimizers algorithms with different parameters may prefer different optimum points.



Implicit regularization effect

- The better optimizer should find the global optimum that also minimizes the test error
  - Components of the optimizer may affect the ability of finding the more generalized optimum point
    - Learning rate, initialization point, batch size, …

This global optimum for the training cost function has a better test error



Model selection via cross validation

- **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
  - Usually a finite set of hyperparameters that we should choose before running the learning algorithm
    - Example: The degree of the polynomial in the regression problem, The number of layers in a neural network, …

Model selection via cross validation

- Suppose we are trying select among several different models for a learning problem.
  - Models with different hyperparamters

- Considering we have some finite number of candidate hypothesis, and we wan to select between them

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
    - $J_v(\boldsymbol{w}) = \frac{1}{|v\_set|}\sum_{i \in v\_set}\left(y^{(i)} - h\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right)\right)^2$
  - Choose the best model based on the validation set error

Simple hold-out: model selection

- 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: model selection

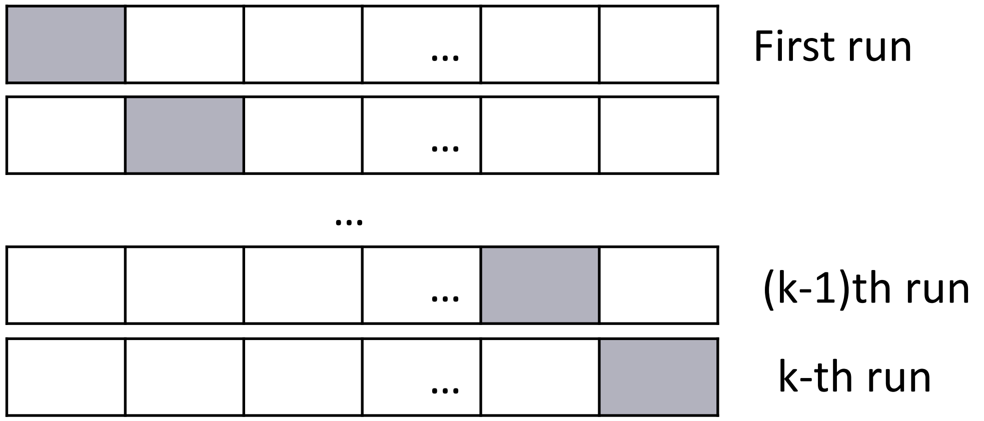
- Simple hold-out chooses the model that minimizes error on validation set.

- $J_v(\hat{\boldsymbol{w}})$ 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

Cross-Validation (CV): Evaluation

- k-fold cross-validation steps:
  - Shuffle the dataset and randomly partition training data into k groups of approximately equal size
  - for $i = 1$ to k
    - Choose the i-th group as the held-out validation group
    - Train the model on all but the i-th group of data
    - Evaluate the model on the held-out group
  - Performance scores of the model from k runs are **averaged**.
    - The average error rate can be considered as an estimation of the true performance of the model.



Cross-Validation (CV): Evaluation

- For each model, we first find the average error by CV.

- The model with **the best average performance** is selected.

Cross-Validation (CV): Evaluation

- When data is particularly scarce, cross-validation with $k = N$
  - Leave-one-out treats each training sample in turn as a test example and all other samples as the training set.

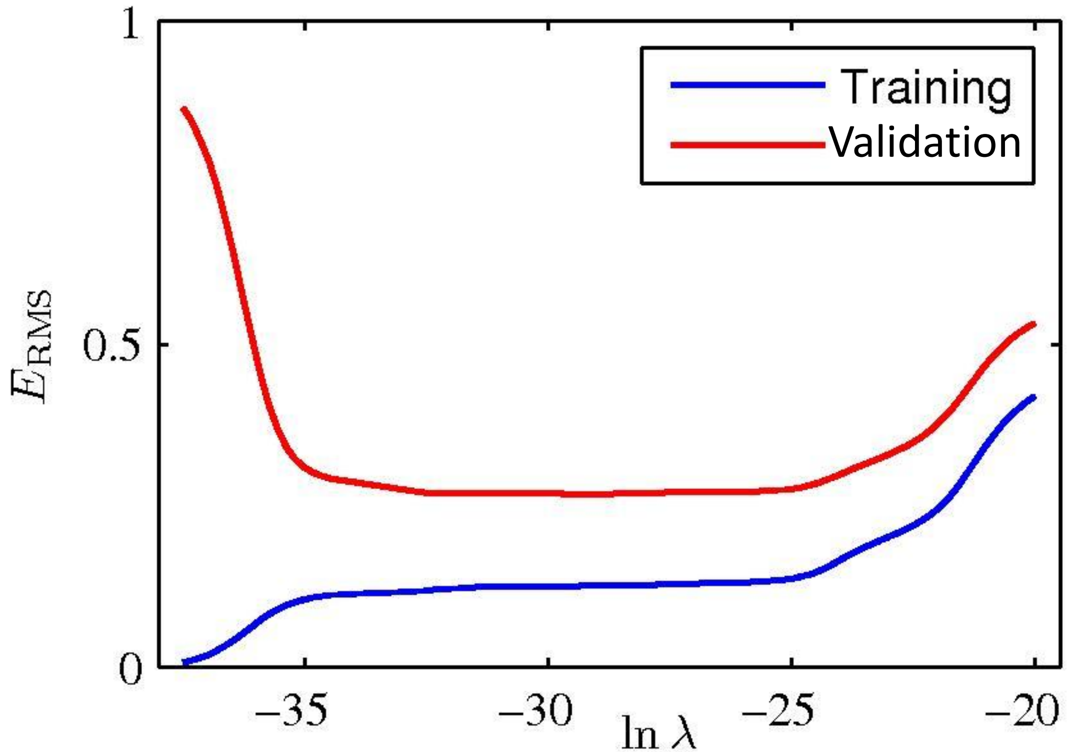
- Use for small datasets
  - When training data is valuable
  - LOOCV can be time expensive as N training steps are required.

Using cross validation
Example: 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 $J_v(\hat{\boldsymbol{w}})$ for each model
  - $J_v(\boldsymbol{w}) = \frac{1}{n\_v} \sum_{i \in v\_set} \left(y^{(i)} - h\left(x^{(i)}; \boldsymbol{w}\right)\right)^2$
- Select the model with the best $J_v(\hat{\boldsymbol{w}})$

Using cross validation
Example: Choosing the regularization parameter

- Using cross validation technique to select the best regularization parameter λ

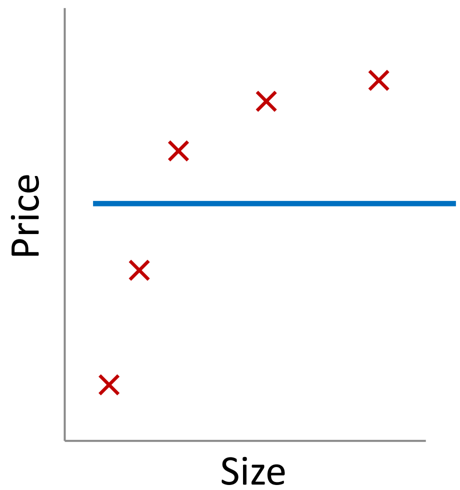


[Bishop]

Using cross validation
Example: Choosing the regularization parameter

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

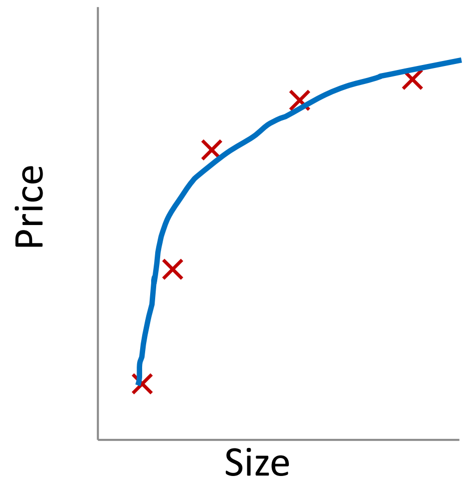
$$J(\boldsymbol{w}) = \frac{1}{n}\left(\sum_{i=1}^{n}\left(y^{(i)} - h\left(x^{(i)};\boldsymbol{w}\right)\right)^2 + \lambda \boldsymbol{w}^T\boldsymbol{w}\right)$$



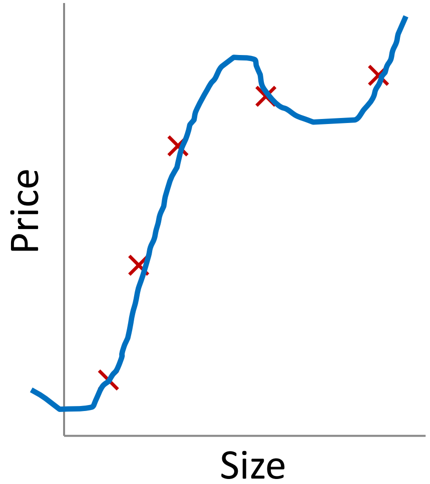
Large λ

(Prefer to more simple models)

$w_1 = w_2 \approx 0$



Intermediate λ



Small λ

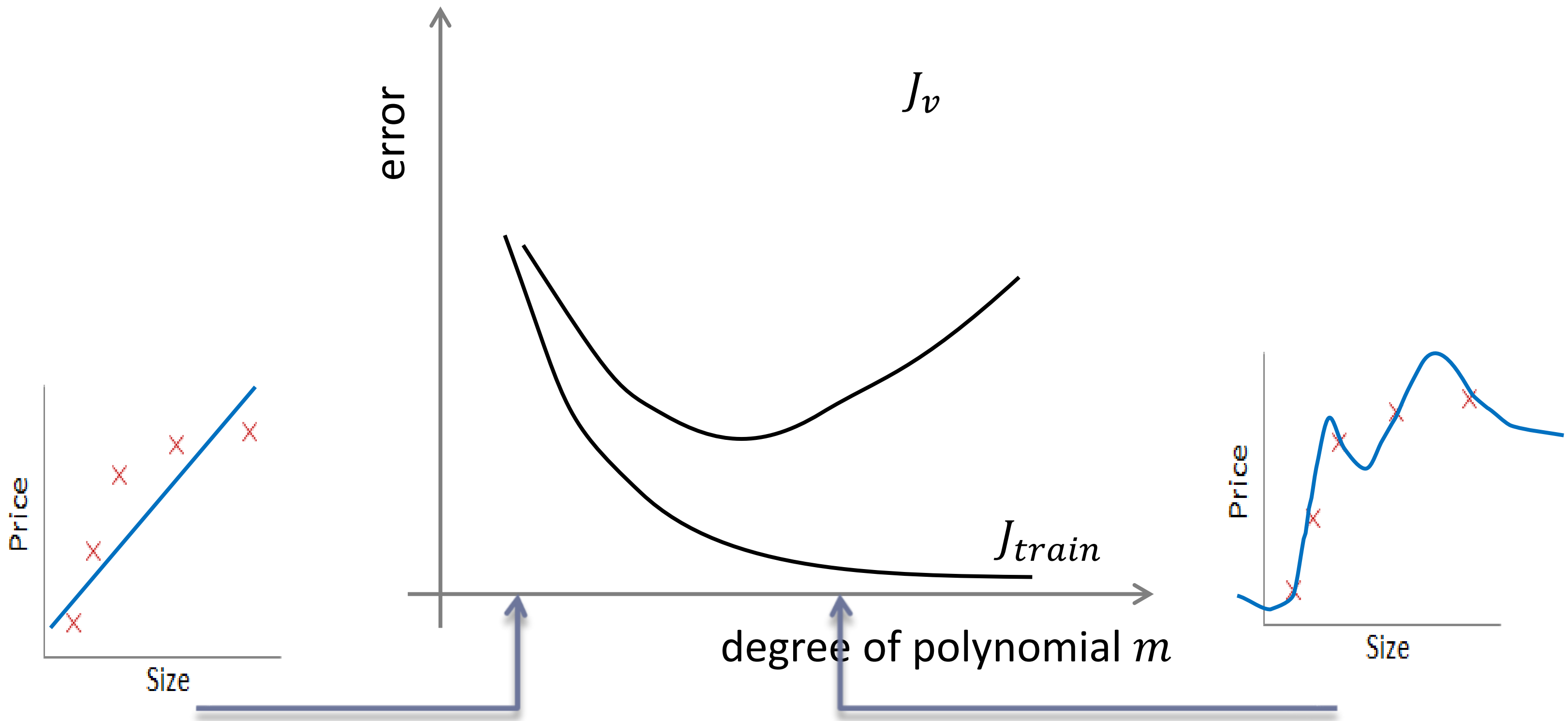
(Prefer to more complex models)

$\lambda = 0$

Using cross validation
Example: Choosing the model complexity

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

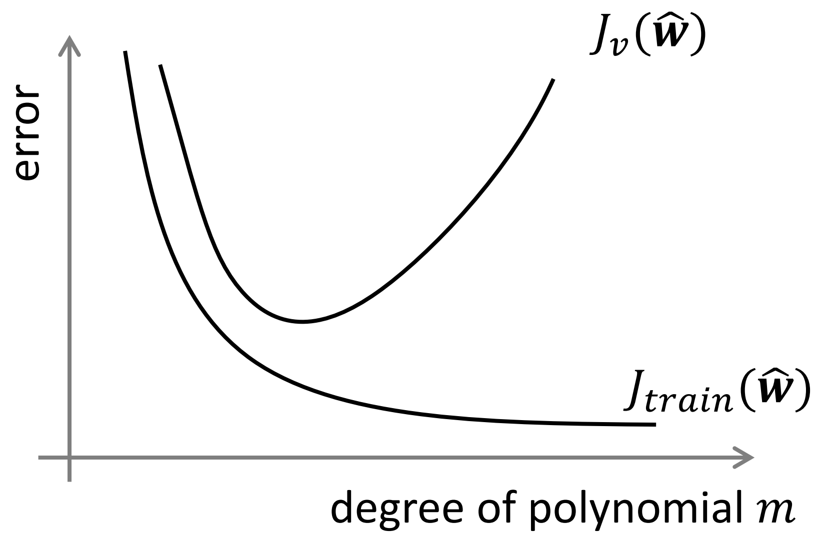
$$J_{train}(\boldsymbol{w}) = \frac{1}{n\_train} \sum_{i \in train\_set} \left(y^{(i)} - h\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right) \right)^2$$



Using cross validation
Example: Choosing the model complexity

- 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

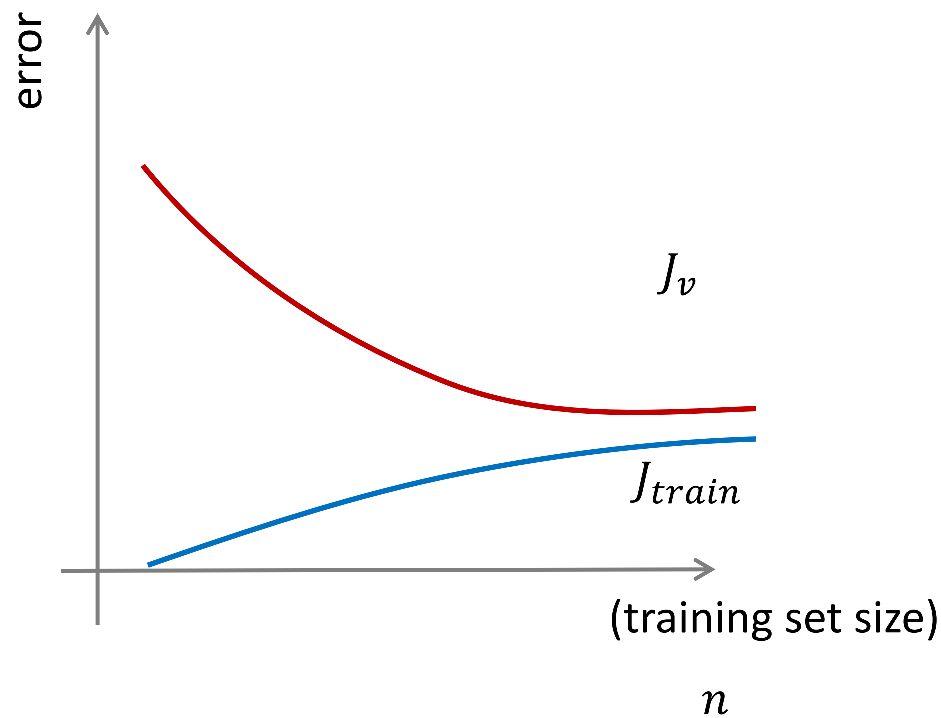


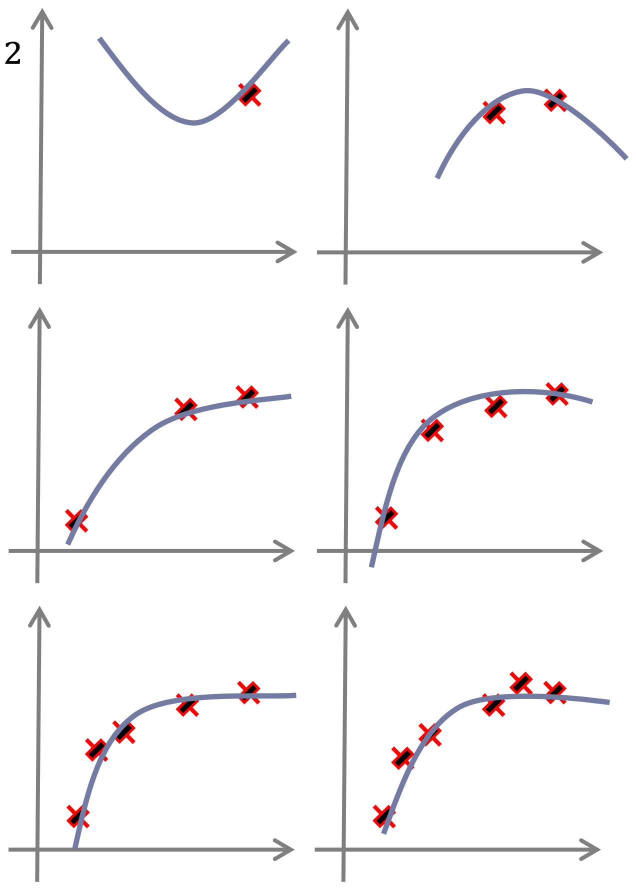
Effect of training set size on test/train error

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

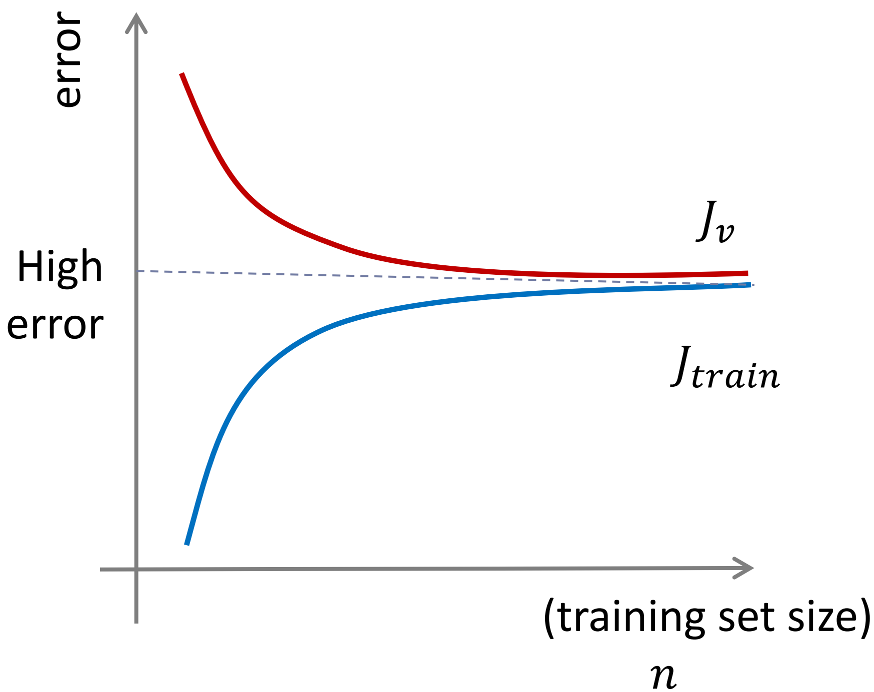
$$J_{train}(\boldsymbol{w}) = \frac{1}{n\_train} \sum\nolimits_{i \in train\_set} \left(y^{(i)} - h\left(x^{(i)}; \boldsymbol{w}\right) \right)^2$$

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



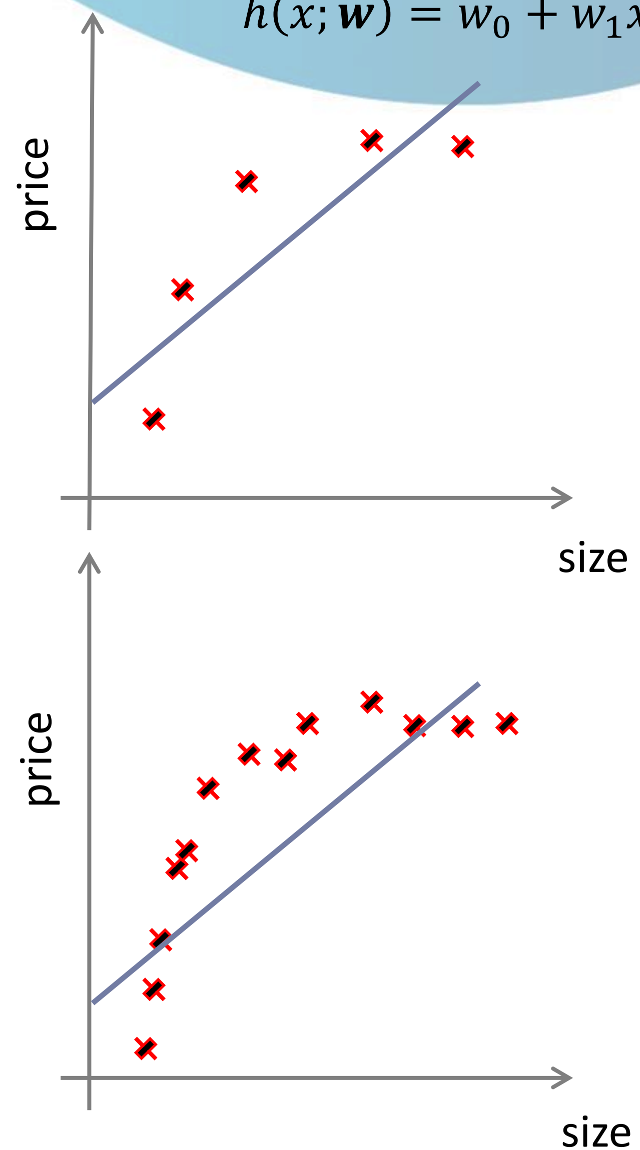


Effect of training set size on test/train error
Less complex $\mathcal{H}$

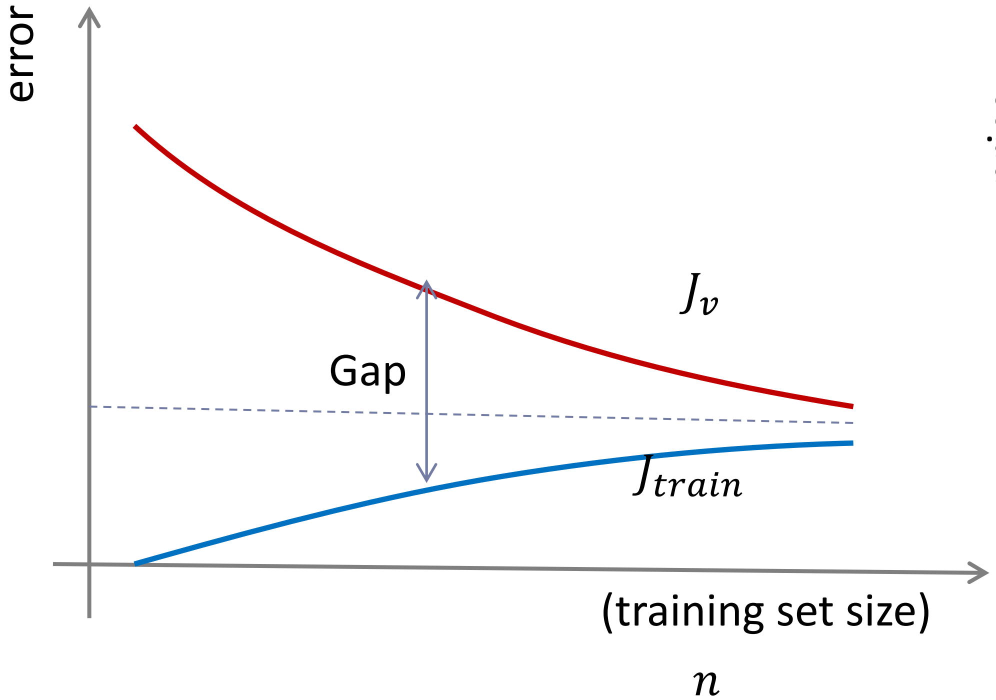


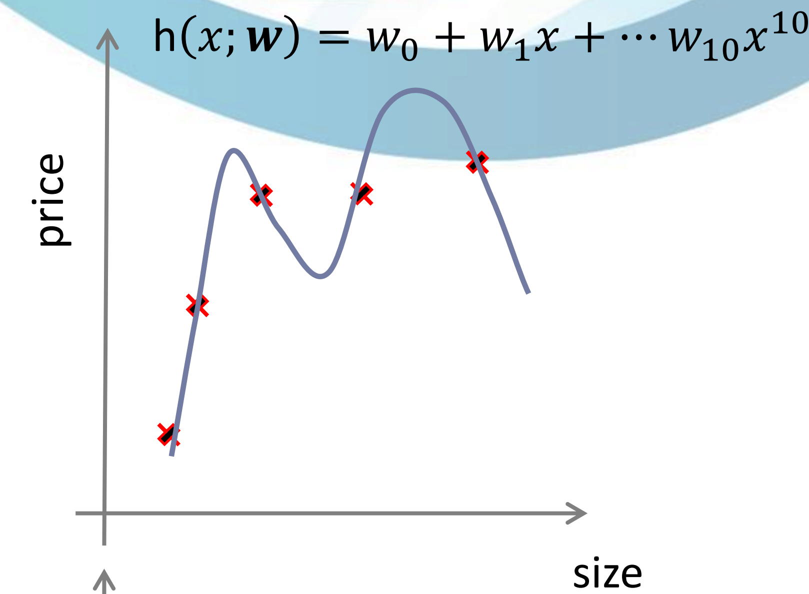
If model is very simple, getting more training data will not (by itself) help much.

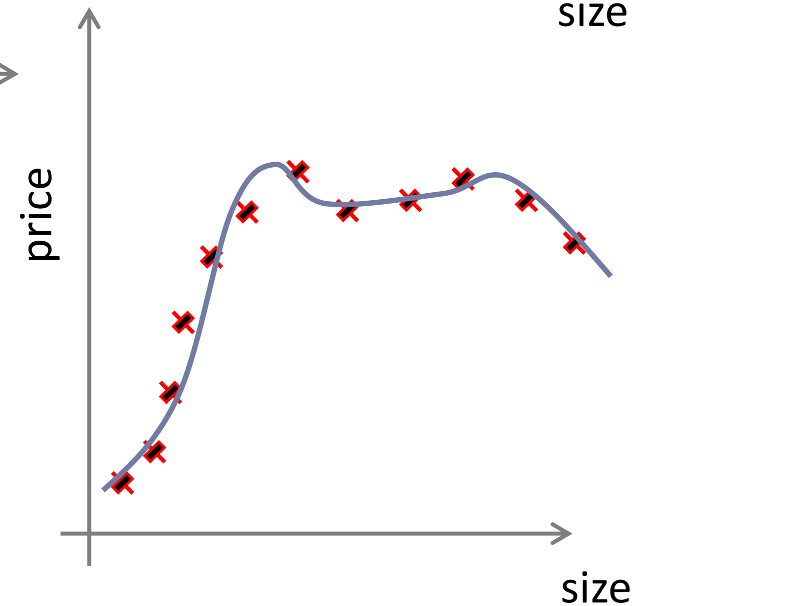
$h(x; \boldsymbol{w}) = w_0 + w_1 x$



Effect of training set size on test/train error
More complex $\mathcal{H}$







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

References

- [1]: Andrew Ng, Machine learning, Stanford (Slides and main_notes.pdf)