Predictive Learning from Data

LECTURE SET 3

Regularization and Complexity Control

Cherkassky, Vladimir, and Filip M. Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

Source: Dr. Vladimir Cherkassky (revised by Dr. Hsiang-Han Chen)

OUTLINE

(following Cherkassky and Mulier, 2007 Chapter 3)

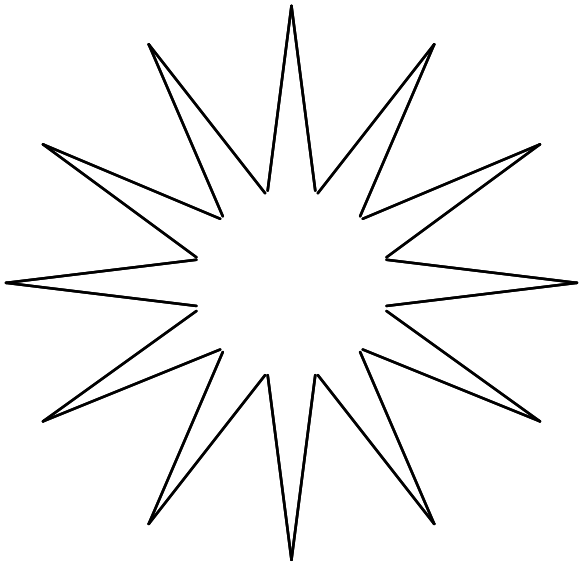
- **Curse of Dimensionality**
- Function approximation framework
- Penalization/ regularization inductive principle
- Bias-variance trade-off
- Complexity Control
- Predictive learning vs. function approximation
- Summary

Curse of Dimensionality

- In the learning problem, the goal is to estimate a function using a finite number of training samples.
- Meaningful estimation is possible only for sufficiently smooth functions, where the function smoothness is measured with respect to sampling density of the training data.
- For high-dimensional functions, it becomes difficult to collect enough samples to attain this high density.
- This problem is commonly referred to as the "curse of dimensionality."

Curse of Dimensionality

- High-dimensional distribution, e.g., hypercube, if it could be visualized, would look like a porcupine



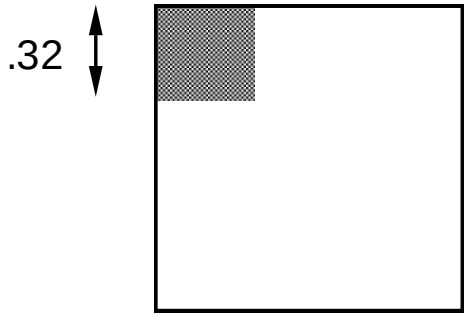
- Sample size needed for accurate fct estimation *grows exponentially* with dimensionality d

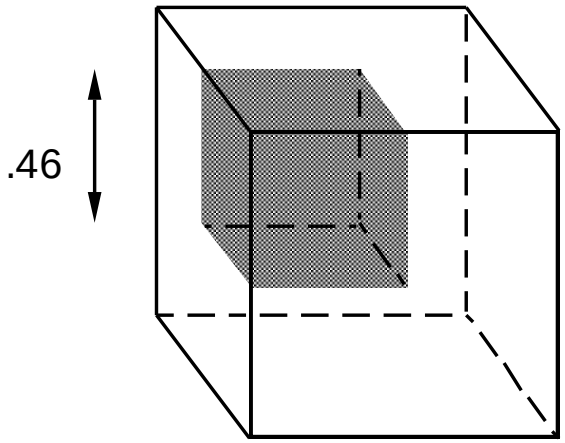
Four properties of high-dimensional distributions

1. Sample sizes yielding the same density increase exponentially with dimension.

   Let us assume that for R^1, a sample containing n data points is considered a dense sample. To achieve the same density of points in d dimensions, we need n^d data points.

2. A large radius is needed to enclose a fraction of the data points in a high-dimensional space.





Both gray regions enclose 10 percent of the samples

Four properties of high-dimensional distributions

3. Almost every point is closer to an edge than to another point.

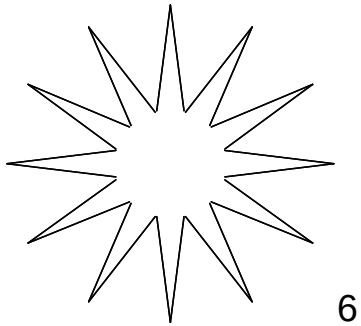
Consider a situation where *n* data points are uniformly distributed in a d-dimensional ball with unit radius. The median distance between the center of the distribution (the origin) and the closest data point is

$$D(d,n) = \left(1 - \frac{1}{2}^{1/n}\right)^{1/d}$$

	d = 10	d = 100
n = 10	0.763	0.973
n = 100	0.609	0.951

4. Almost every point is an outlier in its own projection.

Conceptual illustration: To someone standing on the end of a "quill" of the porcupine, facing the center of the distribution, all the other data samples will appear far away and clumped near the center.



Problems of curse of dimensionality

- Difficult to make local estimates for high-dimensional samples (from properties 1 and 2).
- Difficult to predict a response at a given point because any point will on average be closer to an edge than to the training data point (from properties 3 and 4).

Mathematical theorems contradict the curse of dim??

- The Curse: with finite samples learning in high-dimensions is very hard/ impossible ?

  BUT

- Kolmogorov's theorem states that any continuous function of *d* arguments can be represented as superposition of *univariate functions*

$$f(x_1, \ldots, x_d) = \sum_{j=1}^{2d+1} g_f \left(\sum_{i=1}^{k} a_i \gamma_j(x_i) \right)$$

→ no curse of dimensionality ???

Mathematical theorems contradict the curse of dim??

Explanation: difficulty of learning depends on the complexity of target functions, but Kolmogorov's theorem *does not* quantify complexity.

We can conclude that

- A function's dimensionality is not a good measure of its complexity.
- High-dimensional functions have the potential to be more complex than low-dimensional functions.
- There is a need to provide a characterization of a function's complexity that takes into account its smoothness and dimensionality.

OUTLINE

(following Cherkassky and Mulier, 2007 Chapter 3)

- Curse of Dimensionality
- **Function approximation framework**
- Penalization/ regularization inductive principle
- Bias-variance trade-off
- Complexity Control
- Predictive learning vs. function approximation
- Summary

Function Approximation

- How to represent/approximate any (continuous) *target function* via given class of *basis functions?*

Weierstrass theorem

- Any continuous function on a compact set can be *uniformly* approximated by a polynomial.
- In other words, for any such function $f(x)$ and any positive ε, there exists a polynomial of degree m, $p_m(x)$, such that

$$\| f(x) - p_m(x) \| < \varepsilon$$

for every x.

Universal approximation

- Any (continuous) function can be accurately approximated by another function from a given class (i.e., as in the Weierstrass theorem stated above).
- Most universal approximators represent a linear combination of basis functions (also known as the dictionary method)

$$\mathrm{f}_m(\mathbf{x}, \mathbf{w}) = \sum_{i=0}^{m} w_i \, \mathrm{g}_i(\mathbf{x})$$

where g_i are the basis functions, $\mathbf{w} = [w_0, \ldots, w_{m-1}]$ are parameters.

Universal approximation

- Algebraic polynomials

$$f_m(x, \mathbf{w}) = \sum_{i=0} w_i x^i$$

- Trigonometric polynomials

$$f_m(x, \mathbf{v}_m, \mathbf{w}_m) = \sum_{i=1} v_i \sin(ix) + \sum_{i=1} w_i \cos(ix) + w_0$$

Universal approximation

- Multilayer networks

$$f_m(\mathbf{x}, \mathbf{w}, V) = w_0 + \sum_{j=1}^{m} w_j \, g\left(v_{0j} + \sum_{i=1}^{d} x_i v_{ij} \right)$$

- Local basis function networks

$$f_m(x, \mathbf{v}, \mathbf{w}) = \sum_{i=0} w_i \, K_i \left(\frac{\| \, x - v_i \, \|}{\alpha} \right)$$

Rate-of-convergence

- Relate the accuracy of approximation to the *properties of target function* (its dimensionality *d* and smoothness *s*)
- Rate-of-convergence $\sim O(m^{-s/d})$

  where *s* = number of continuous derivatives

  (~smoothness)
- For a given approximation error, the number of parameters exponentially increases with *d* (for a fixed measure of ‘‘complexity’’ *s*).

Characterization of a function's complexity

1. **Define the measure of complexity for a class of target functions.**

   This class of functions should be very general, so that it is likely to include most target functions in real-life applications.

2. **Specify a class of approximating functions of a learning machine.**

   For example, choose a particular dictionary in representation. This dictionary should have ‘‘the universal approximation’’ property.

   Flexibility of approximating functions is specified by the number of basis functions *m*.

Characterization of a function's complexity

3. **Estimate the (best possible) rate of convergence**, defined as the accuracy of approximating an arbitrary function in the class.

   In other words, estimate how quickly the approximation error of a method goes to zero when the number of its parameters grows large.

   It is of particular interest to see how the rate of convergence depends on the dimensionality of the class of functions.

Some related issues

- **How to measure function complexity?**

  Number of continuous derivatives s (smoothness).

  Frequency content (i.e., max frequency)

- **Fundamental restriction**:

  good function approximation (in high dimension) is possible only for very smooth functions

- **Implications for machine learning methods**:

  - good generalization ***not possible*** (for high-dim. data),

  - need to ***constrain/control smoothness*** for multivariate problems → **penalization inductive principle**.

OUTLINE

- Curse of Dimensionality
- Function approximation framework
- **Penalization/ regularization inductive principle**
- Bias-variance trade-off
- Complexity Control
- Predictive learning vs. function approximation
- Summary

Penalization Inductive Principle

- Specify a wide set of models $f(\mathbf{x}, \omega)$
- Find a model ω^* minimizing Penalized Risk

$$R_{pen}(\omega) = R_{emp}(\omega) + \lambda \phi[\mathrm{f}(\mathbf{x}, \omega)]$$

$\phi[\mathrm{f}(\mathbf{x}, \omega)]$ ~ non-negative penalty functional;
its larger values penalize complex functions.

$\lambda > 0$ ~ regularization parameter controls the strength of penalty relative to empirical risk (data term)

Model Selection problem: select λ so that the solution $f_\lambda(\mathbf{x}, \omega)$ found by minimizing $R_{pen}(\omega)$ provides minimum *expected risk* aka *prediction risk* aka *test error*.

→ *Need to estimate expected risk for each solution*

Penalization of linear models

- Linear function:

$$\hat{y} = w[0] \times x[0] + w[1] \times x[1] + ... + w[n] \times x[n] + b$$

- Cost function: $\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2$

- LASSO (L1 regularization):

$$L_{lasso}(\hat{\beta}) = \sum_{i-1}^{n} (y_i - x_i' \hat{\beta})^2 + \lambda \sum_{j-1}^{m} |\hat{\beta}_j|.$$

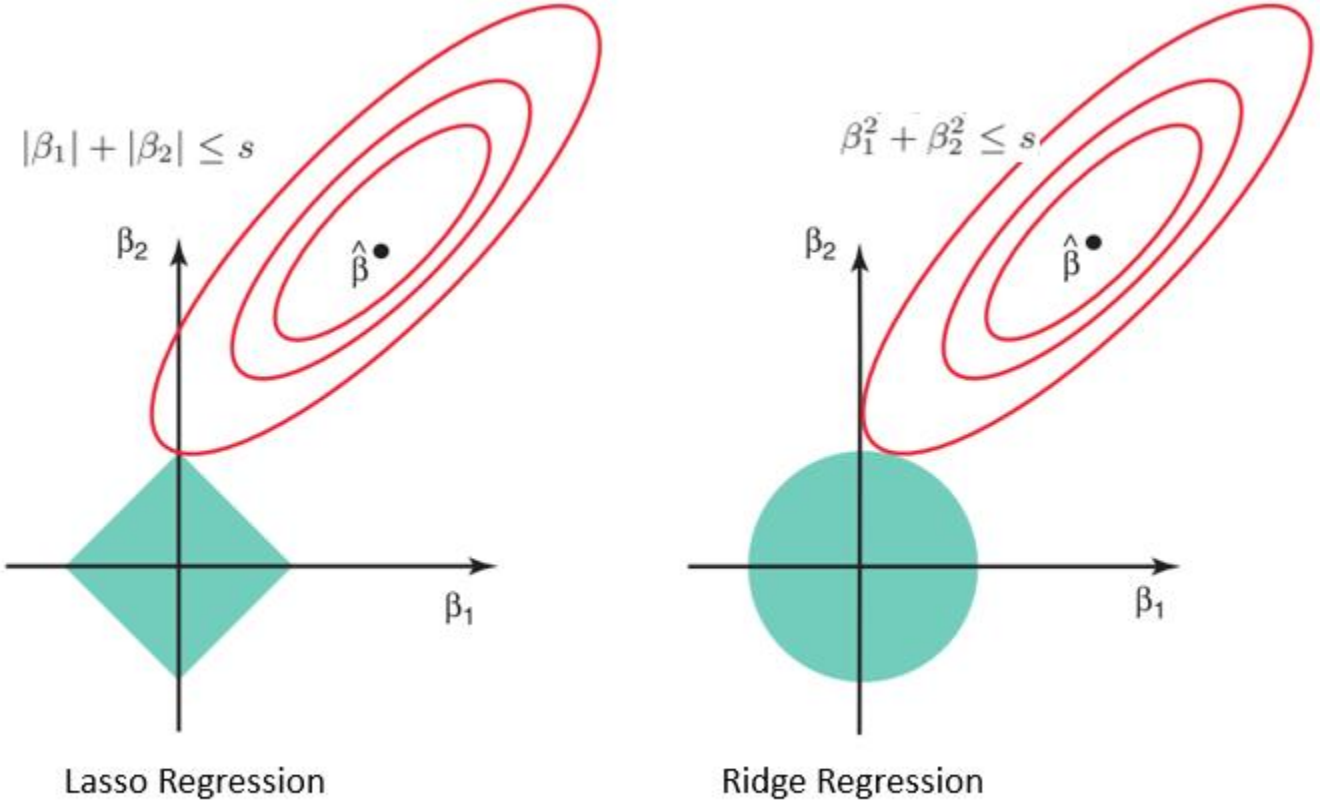
- Ridge (L2 regularization):

$$L_{hridge}(\hat{\beta}) = \sum_{i-1}^{n} (y_i - x_i' \hat{\beta})^2 + \lambda \sum_{j-1}^{m} w_j \hat{\beta}_j^2.$$

https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression

Penalization of linear models

- Which one can perform feature selection?



https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression

Modeling Issues for Penalization

- **Choice of admissible models** $f(\mathbf{x}, \omega)$

  (1) continuous functions

  (2) wide class of parametric functions

- **Type of penalty** $\phi[\mathrm{f}(\mathbf{x}, \omega)]$ **~ prior knowledge**

  (1) nonparametric (smoothness constraints)

  (2) parametric, e.g. ridge penalty, subset selection

- **Method for minimizing** $R_{pen}(\omega)$
- **Choice of** λ **~ complexity control**

→ Final model depends on all factors above *BUT*
different tradeoffs for small vs. large samples

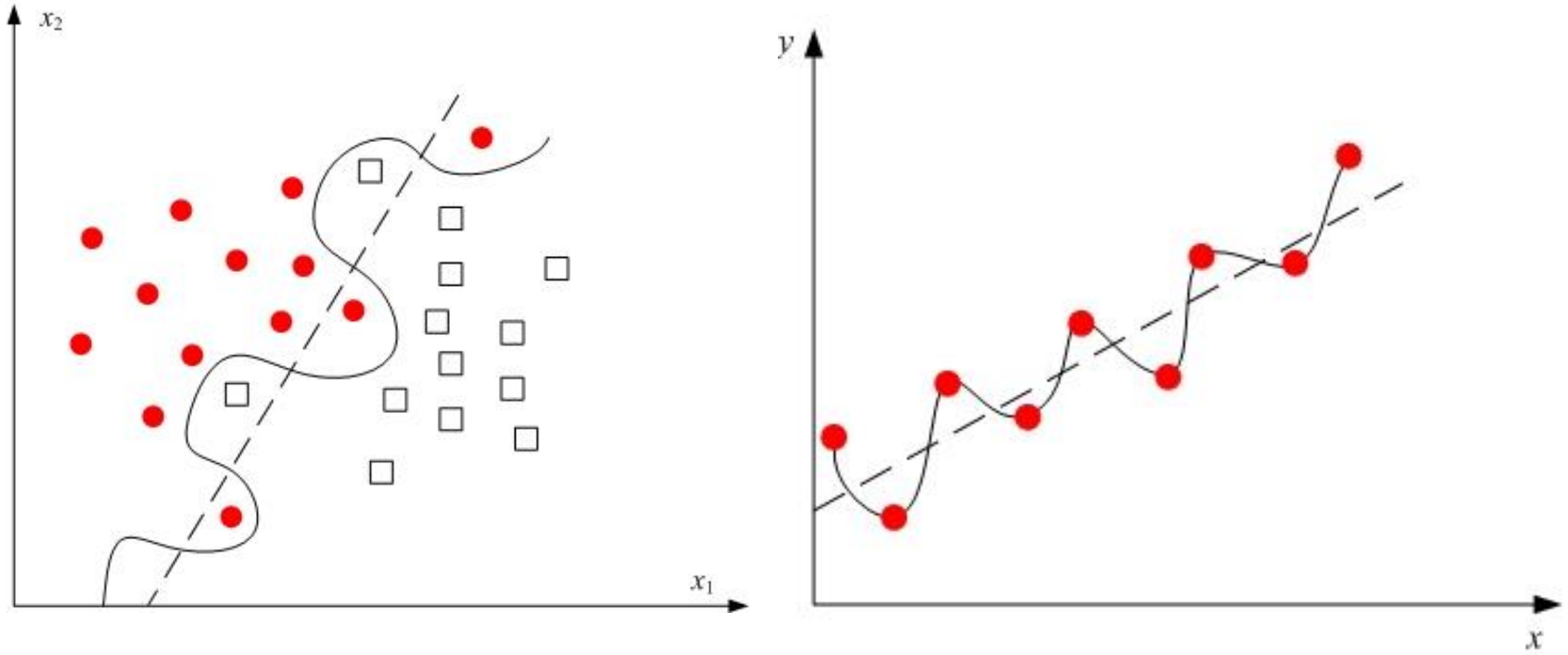
OUTLINE

- Curse of Dimensionality
- Function approximation framework
- Penalization/ regularization inductive principle
- **Bias-variance trade-off**
- Complexity Control
- Predictive learning vs. function approximation
- Summary

Bias-Variance Trade-Off

Statistical ‘explanation’ for model complexity control

Recall (a) Classification (b) Regression



Imagine many training data sets of the same size

Bias-Variance Trade-Off

- For regression problems with squared loss:

  Consider MSE btwn an estimate and the target fct.
  Note: MSE is averaged over many possible training samples of the same size *n*

$$mse(f(\mathbf{x},\omega)) = \int E_n\left[(g(\mathbf{x}) - f(\mathbf{x},\omega))^2\right] p(\mathbf{x}) d\mathbf{x}$$

  this MSE can be expressed as

$$mse(f(\mathbf{x},\omega)) = bias^2(f(\mathbf{x},\omega)) + var(f(\mathbf{x},\omega))$$

  where $bias^2(f(\mathbf{x},\omega)) = \int (g(\mathbf{x}) - E[f(\mathbf{x},\omega)])^2 p(\mathbf{x}) d\mathbf{x}$

$$var(f(\mathbf{x},\omega)) = \int E\left[(f(\mathbf{x},\omega) - E[f(\mathbf{x},\omega)])^2\right] p(\mathbf{x}) d\mathbf{x}$$

Bias vs Varaince

- The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting). when the model is too simple, there will be a larger bias(difference) than the actual data

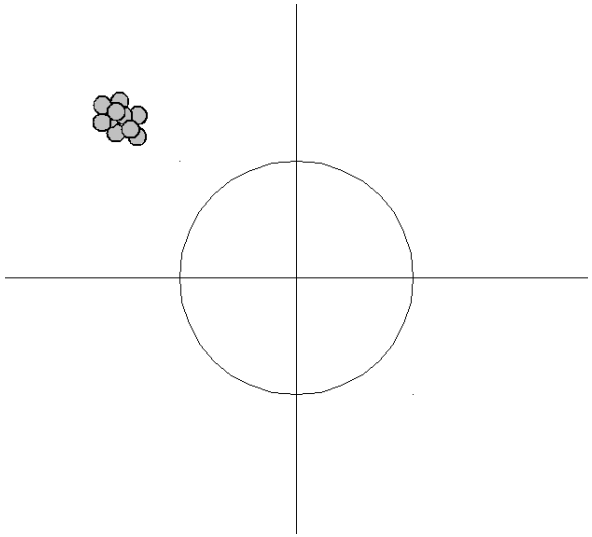
$$bias^2\left(f(\mathbf{x},\omega)\right)=\int\left(g(\mathbf{x})-E\left[f(\mathbf{x},\omega)\right]\right)^2 p(\mathbf{x})d\mathbf{x}$$

- The variance is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (overfitting). When the data model is too complex and is prove to overfit, so larger variance will be a problem to the prediction

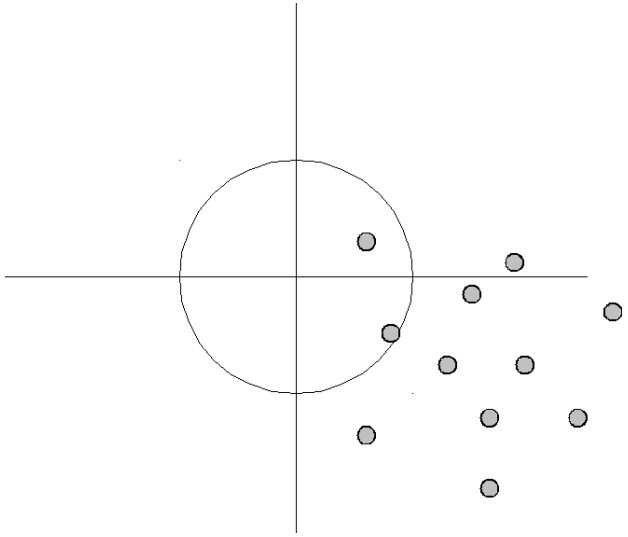
$$var\left(f(\mathbf{x},\omega)\right)=\int E\left[\left(f(\mathbf{x},\omega)-E\left[f(\mathbf{x},\omega)\right]\right)^2\right]p(\mathbf{x})d\mathbf{x}$$

https://en.wikipedia.org/

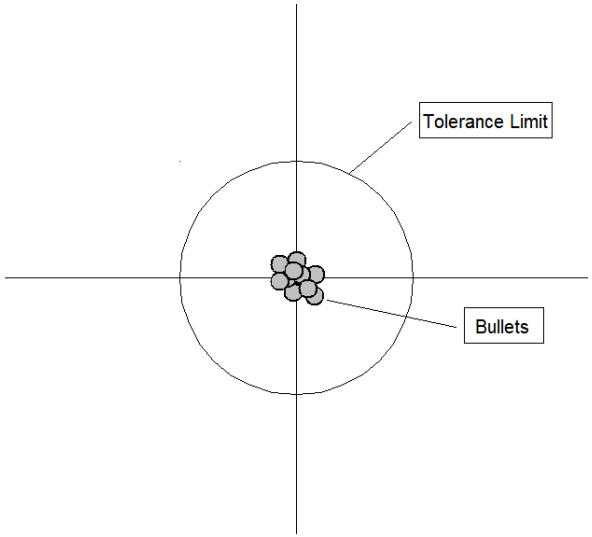
High bias, low variance



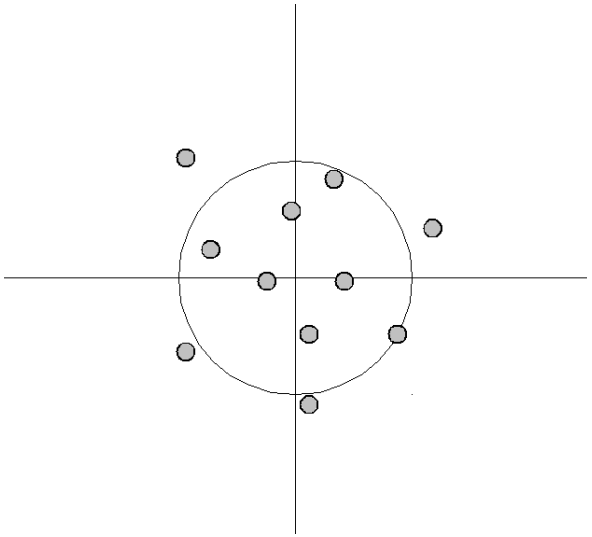
High bias, high variance



Low bias, low variance



Low bias, high variance



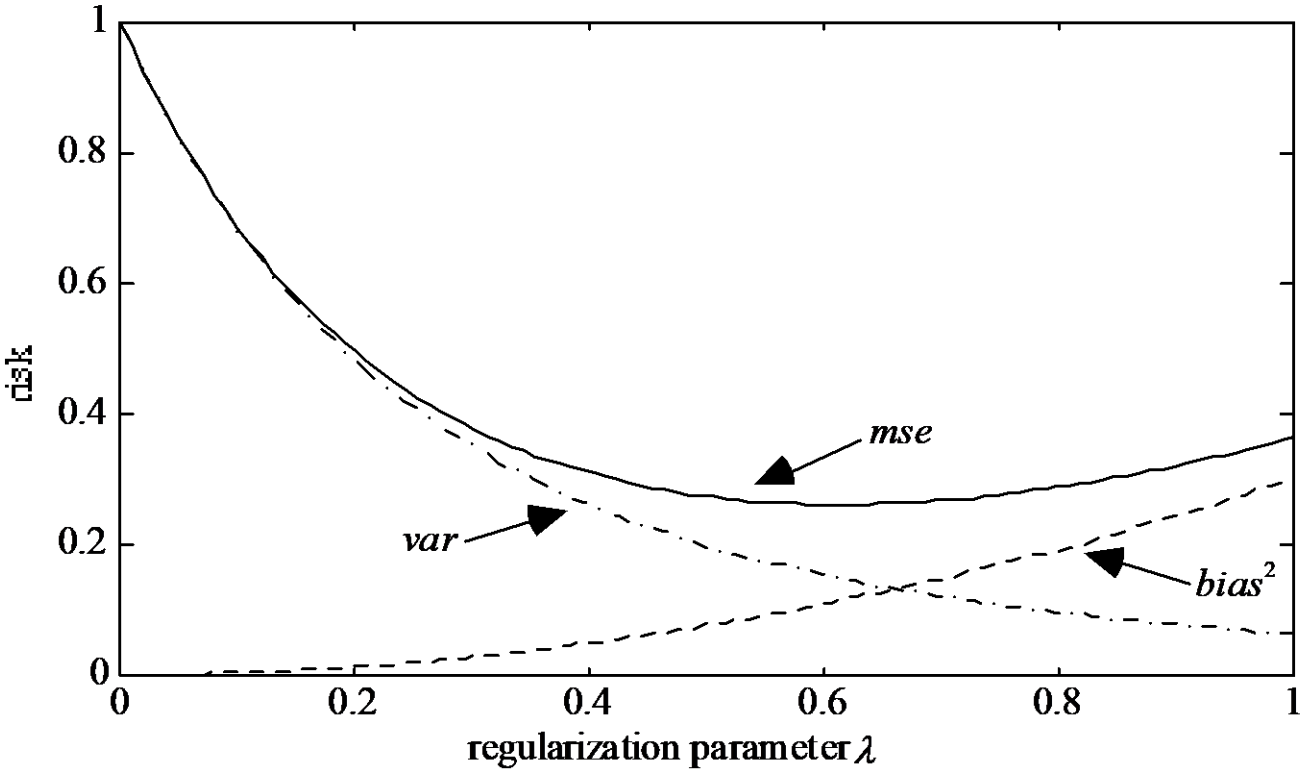
https://en.wikipedia.org/

Bias-Variance trade-off and Penalization

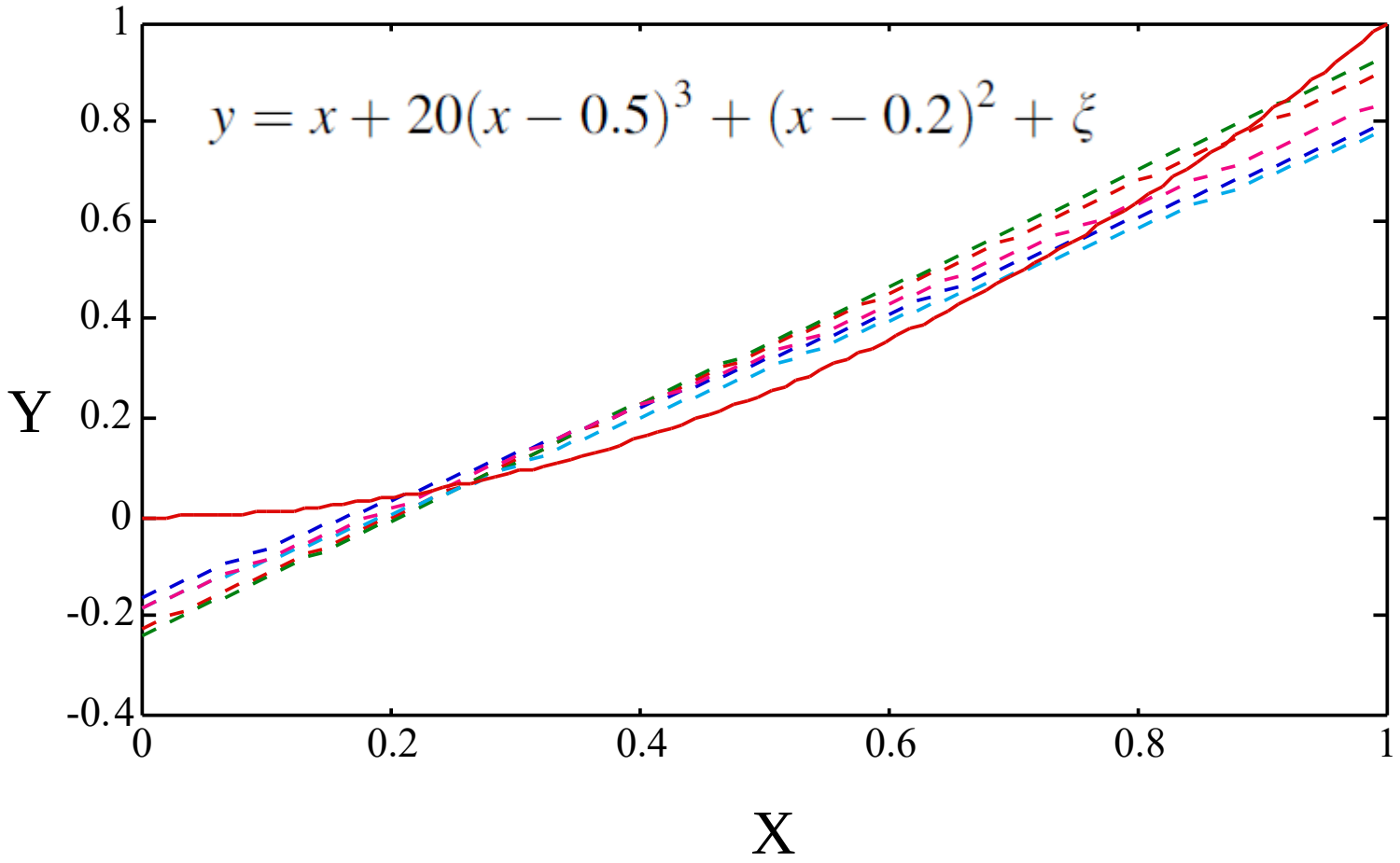
- Parameter λ controls bias-variance trade-off:

  - larger λ values → smaller variance/ larger bias

  - smaller λ values → larger variance (the model becomes more dependent on the training data)



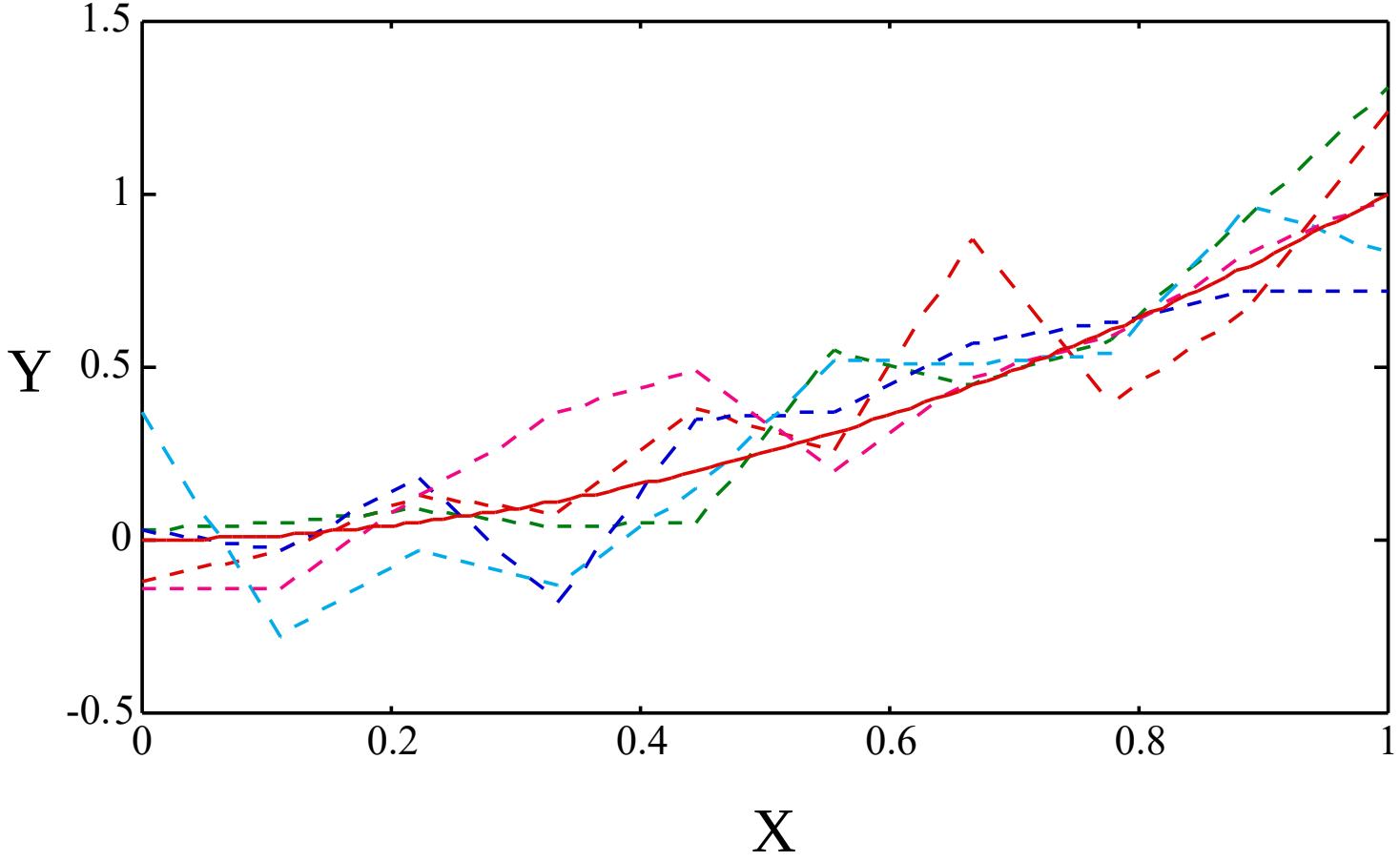
Example of high bias (underfitting)



50 training samples (5 data sets)

Gaussian kernel width ~ 80%

Example of high variance (overfitting)



Same training data (5 data sets)

Piecewise-linear regression fitting (10 components)

Summary: bias-variance trade-off

- The concept of complexity control
- Model estimation depends on two terms:

  - data (empirical risk)

  - penalization term (~ a priori knowledge)
- A particular set of models (approximating functions) cannot be ‘best’ for all data sets
- Bias-variance formalism does not provide practical mechanism for model selection

OUTLINE

- Curse of Dimensionality
- Function approximation framework
- Penalization/ regularization inductive principle
- Bias-variance trade-off
- **Complexity Control**
- Predictive learning vs. function approximation
- Summary

How to Control Model Complexity ?

- Two approaches: analytic and resampling
- Analytic criteria estimate prediction error as a function of fitting error and model complexity

  For regression problems:

$$R(\omega) \cong r\left(\frac{DoF}{n}\right) R_{emp}$$

Representative analytic criteria for regression

- Schwartz Criterion: $r(p,n) = 1 + p(1-p)^{-1} \ln n$

- Akaike's FPE: $r(p) = (1+p)(1-p)^{-1}$

where p = DoF/n, n~sample size, DoF~degrees-of-freedom

Analytical Model Selection

- Consider estimating the data using the set of polynomial approximating functions of arbitrary degree

$$f_m(x, \mathbf{w}_m) = \sum_{i=0}^{m-1} w_i x^i$$

- For practical purposes, we will limit the polynomial degree m ≤ 10.

when m goes up, the model complexity goes up

empirical risk

penalty term, higher complexity gives bigger penalty

optimal m with lowest R

TABLE 3.1 Model Selection Using fpe for Estimating Prediction Risk

m	R_{emp}	Final Prediction Error $r(m/n)$	Estimated R via fpe
1	0.1892	1.0833	0.2049
2	0.1400	1.1739	0.1644
3	0.1230	1.2727	0.1565
4	0.1063	1.3810	0.1468
5	0.0531	1.5000	0.0797
6	0.0486	1.6316	0.0792
7	0.0485	1.7778	0.0863
8	0.0418	1.9412	0.0812
9	0.0417	2.1250	0.0886
10	0.0406	2.3333	0.0947

Resampling

- Split available data into 2 subsets: Training + Validation

  (1) Use training set for model estimation (via data fitting)

  (2) Use validation data to estimate the prediction error of the model
- Change model complexity index and repeat (1) and (2)
- Select the final model providing the lowest (estimated) prediction error

BUT results are sensitive to data splitting

K-fold cross-validation

1. Divide the training data **Z** into *k* (randomly selected) disjoint subsets $\{\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_k\}$ of size *n/k*

2. For each 'left-out' validation set $\mathbf{Z}_i$: $\hat{y} = f_i(\mathbf{x})$

   - use remaining data to estimate the model
   - estimate prediction error on $\mathbf{Z}_i$: $r_i = \frac{k}{n} \sum_{j \in \mathbf{Z}_i} \left(f(\mathbf{x}_j) - y_j \right)^2$

3. Estimate ave. prediction risk as $R_{cv} = \frac{1}{k} \sum_{i=1}^{k} r_i$

K-fold cross-validation

- Consider 25 samples for cross-validation

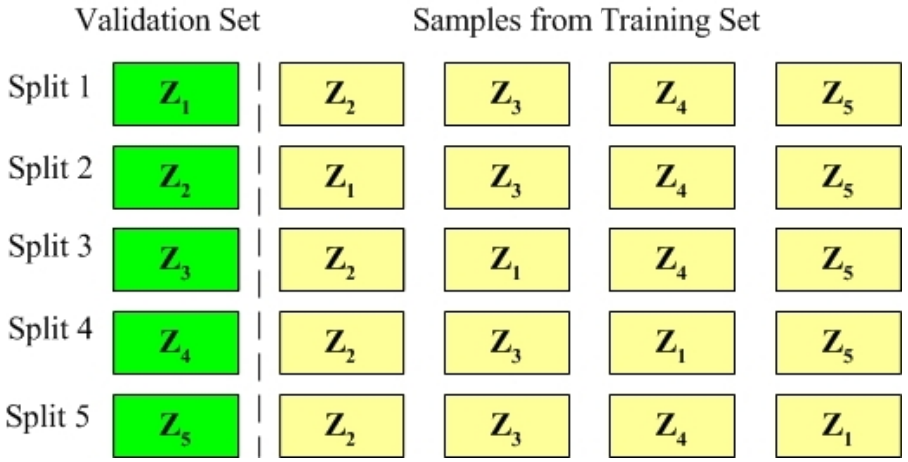


TABLE 3.2 Validation Sets for Fivefold Cross-Validation

Validation set	Samples from training set
Z_1	$[(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)]$
Z_2	$[(x_6, y_6), (x_7, y_7), (x_8, y_8), (x_9, y_9), (x_{10}, y_{10})]$
Z_3	$[(x_{11}, y_{11}), (x_{12}, y_{12}), (x_{13}, y_{13}), (x_{14}, y_{14}), (x_{15}, y_{15})]$
Z_4	$[(x_{16}, y_{16}), (x_{17}, y_{17}), (x_{18}, y_{18})(x_{19}, y_{19})(x_{20}, y_{20})]$
Z_5	$[(x_{21}, y_{21}), (x_{22}, y_{22}), (x_{23}, y_{23}), (x_{24}, y_{24}), (x_{25}, y_{25})]$

K-fold cross-validation

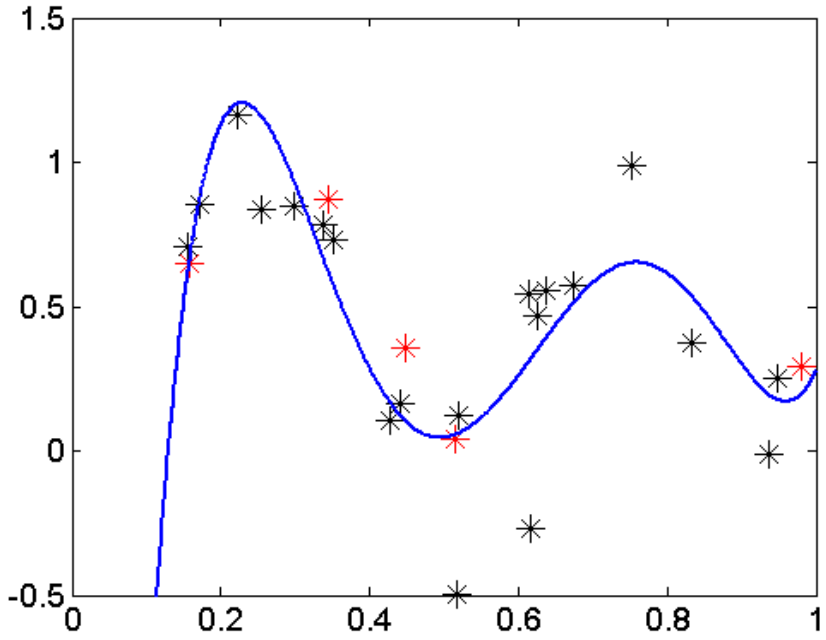
TABLE 3.3 Calculation of the Risk Estimate via Fivefold Cross-Validation

Polynomial estimate of degree m	Data to construct polynomial estimate	Validation set to estimate risk	Estimate of expected risk for each validation set
$f_1(x)$	$[\mathbf{Z}_2, \mathbf{Z}_3, \mathbf{Z}_4, \mathbf{Z}_5]$	$\mathbf{Z}_1$	$r_1 = \frac{1}{5}\sum_{i=1}^{5} (f_1(x_i) - y_i)^2$
$f_2(x)$	$[\mathbf{Z}_1, \mathbf{Z}_3, \mathbf{Z}_4, \mathbf{Z}_5]$	$\mathbf{Z}_2$	$r_2 = \frac{1}{5}\sum_{i=6}^{10} (f_2(x_i) - y_i)^2$
$f_3(x)$	$[\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_4, \mathbf{Z}_5]$	$\mathbf{Z}_3$	$r_3 = \frac{1}{5}\sum_{i=11}^{15} (f_3(x_i) - y_i)^2$
$f_4(x)$	$[\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3, \mathbf{Z}_5]$	$\mathbf{Z}_4$	$r_4 = \frac{1}{5}\sum_{i=16}^{20} (f_4(x_i) - y_i)^2$
$f_5(x)$	$[\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3, \mathbf{Z}_4]$	$\mathbf{Z}_5$	$r_5 = \frac{1}{5}\sum_{i=21}^{25} (f_5(x_i) - y_i)^2$
Risk estimate			$R_{\text{cv}}(m) = \frac{1}{5}\sum_{i=1}^{5} r_i$

Example of model selection(1)

- 25 samples are generated as $y = \sin^2(2\pi x) + \xi$ with x uniformly sampled in [0,1], and noise ~ $N(0,1)$
- Regression estimated using polynomials of degree m=1,2,…,10
- Polynomial degree m = 5 is chosen via 5-fold cross-validation. The curve shows the polynomial model, along with training (*) and validation (*) data points, for one partitioning.

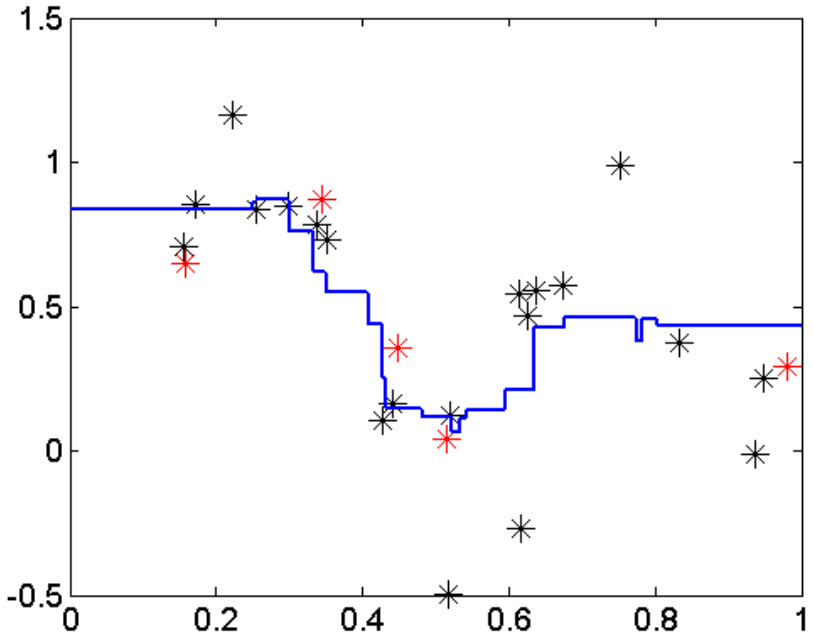
m	Estimated R via Cross validation
1	0.1340
2	0.1356
3	0.1452
4	0.1286
5	**0.0699**
6	0.1130
7	0.1892
8	0.3528
9	0.3596
10	0.4006



Example of model selection(2)

- Same data set, but estimated using *k-nn* regression.
- Optimal value k = 7 chosen according to 5-fold cross-validation model selection. The curve shows the *k-nn* model, along with training (*) and validation (*) data points, for one partitioning.

k	Estimated *R* via Cross validation
1	0.1109
2	0.0926
3	0.0950
4	0.1035
5	0.1049
6	0.0874
7	**0.0831**
8	0.0954
9	0.1120
10	0.1227



More on Resampling

- Leave-one-out (LOO) cross-validation
  - extreme case of *k*-fold when *k=n* (# samples)
  - efficient use of data, but requires *n* estimates
- Final (selected) model depends on:
  - random data
  - random partitioning of the data into K subsets (folds)

  → the same resampling procedure may yield different model selection results
- Some applications may use non-random splitting of the data into (training + validation)
- Model selection via resampling is based on estimated prediction risk (error).
- Does this estimated error measure reflect true prediction accuracy of the final model?

Resampling for estimating true risk

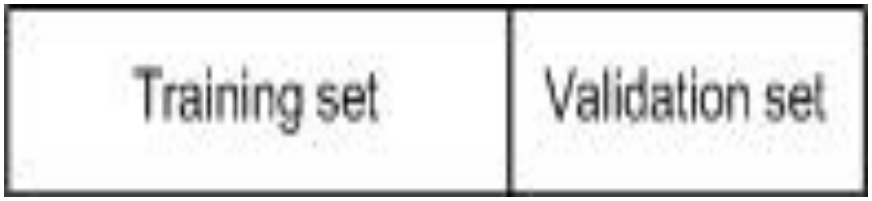
- Prediction risk (test error) of a method can be also estimated via resampling
- Partition the data into: Training/ validation/ test
- Test data should be never used for model estimation
- Double resampling method:

  - for complexity control

  - for estimating prediction performance of a method
- Estimation of prediction risk (test error) is critical for comparison of different learning methods

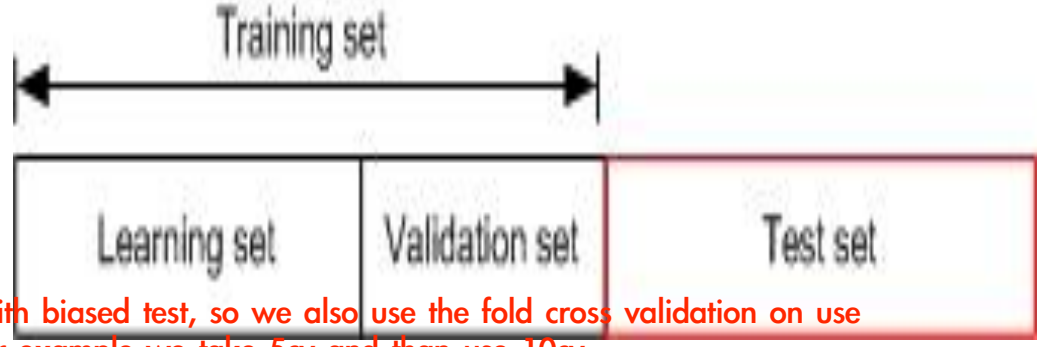
On resampling terminology

- Often confusing and inconsistent terminology
  - Resampling for model selection:



If the degree of the model is already given, we wont need the validation set
i.e validation set is only for model selection

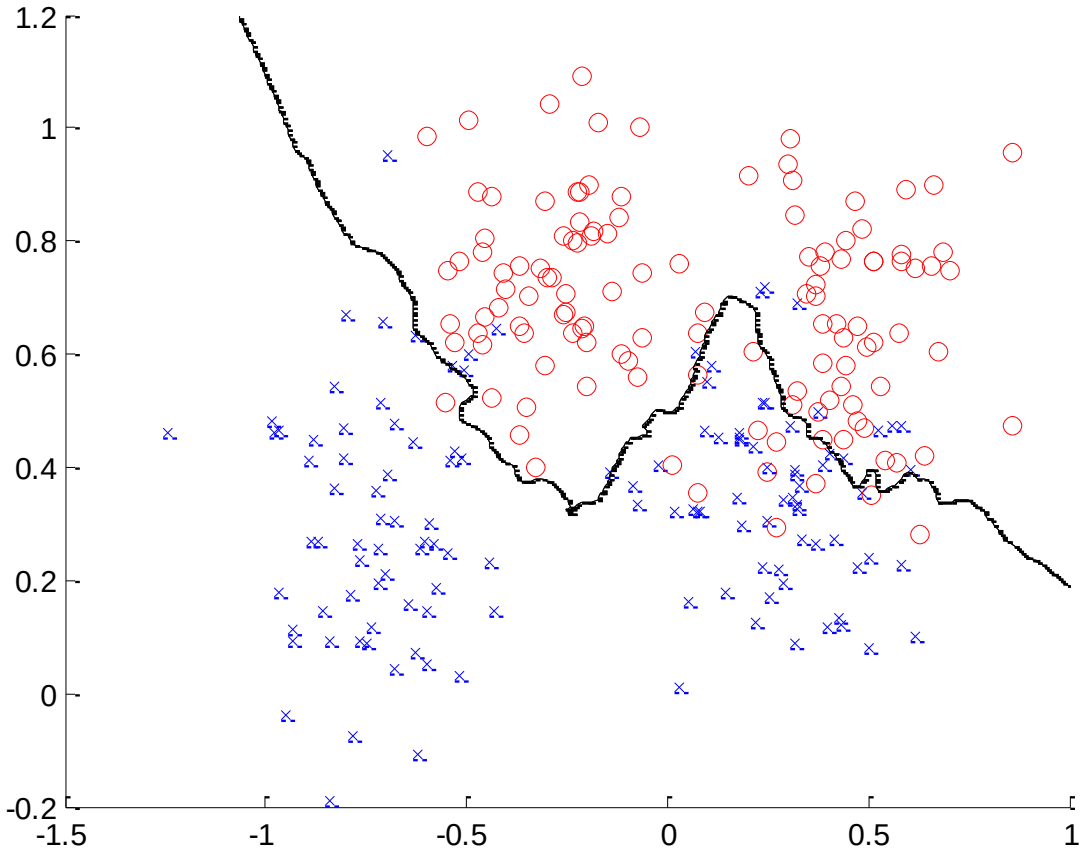
- Double resampling for estimating test error *and* model selection



might select test set with biased test, so we also use the fold cross validation on use
for example we take 5cv and than use 10cv
i.e take 20% as test, 80% to do validation, with each validation of 8%(80/10) and do this 5 time(picking 20% each time as test set to make sure we didnt just select a bad test set

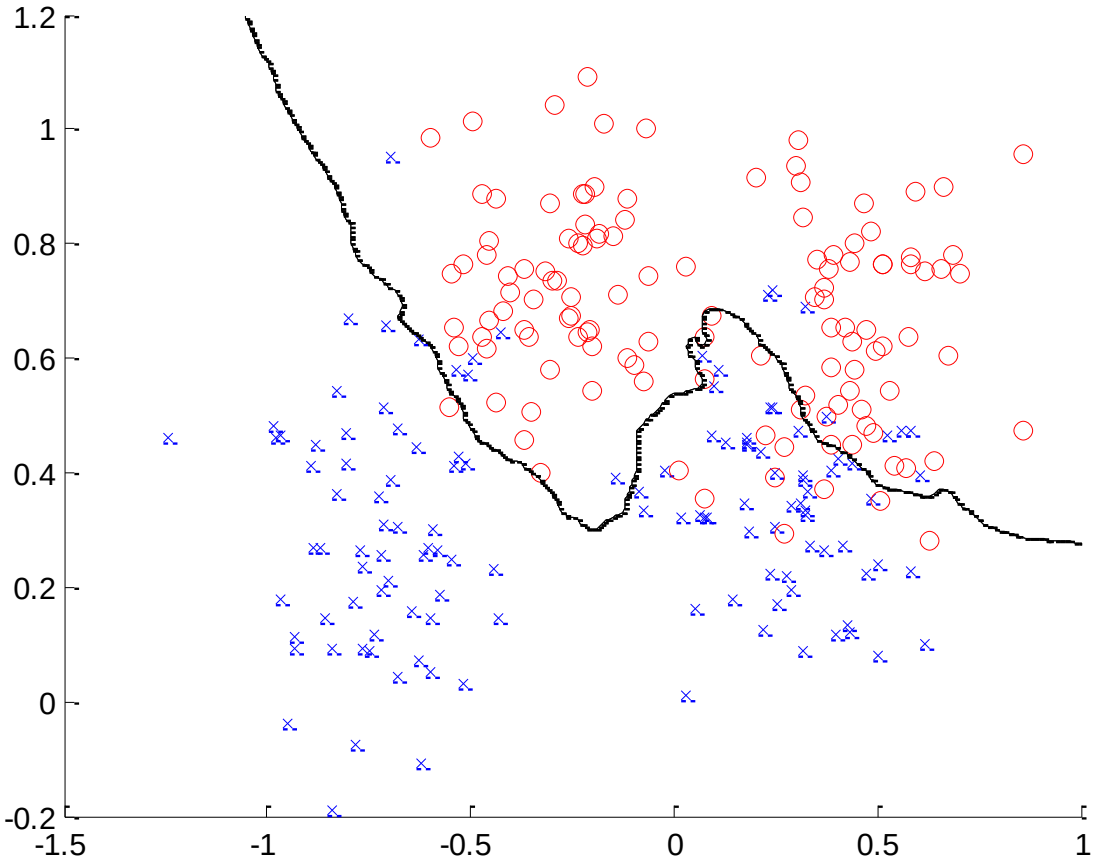
Example of model selection for k-NN classifier via 6-fold x-validation: Ripley's data. Optimal decision boundary for k=14

k	Error (%)
1	14.48%
3	13.61%
7	14.42%
14	11.19%
30	11.59%
45	11.61%
47	11.61%
50	11.19%
53	11.61%
57	12.44%
60	12.44%
80	14.84%



Example of model selection for k-NN classifier via 6-fold x-validation: Ripley's data. Optimal decision boundary for k=50

which one is better? k=14 or 50



Estimating test error of a method

- For the same example (Ripley's data) what is the true test error of k-NN method ?
- Use **double resampling**, i.e. 5-fold cross validation to estimate test error, and 6-fold cross-validation to estimate optimal k for each training fold:

prorblem with small data set k may not be insistent

Fold #	k	Validation	Test error
1	20	11.76%	14%
2	9	0%	8%
3	1	17.65%	10%
4	12	5.88%	18%
5	7	17.65%	14%
mean		**10.59%**	**12.8%**

Use test error to do comparison between different models regarless of the model complexity

- **Note:** opt k-values are different; errors vary for each fold, due to high variability of random partitioning of the data

Estimating test error of a method

- Another realization of **double resampling**, i.e. 5-fold cross validation to estimate test error, and 6-fold cross-validation to estimate optimal k for each training fold:

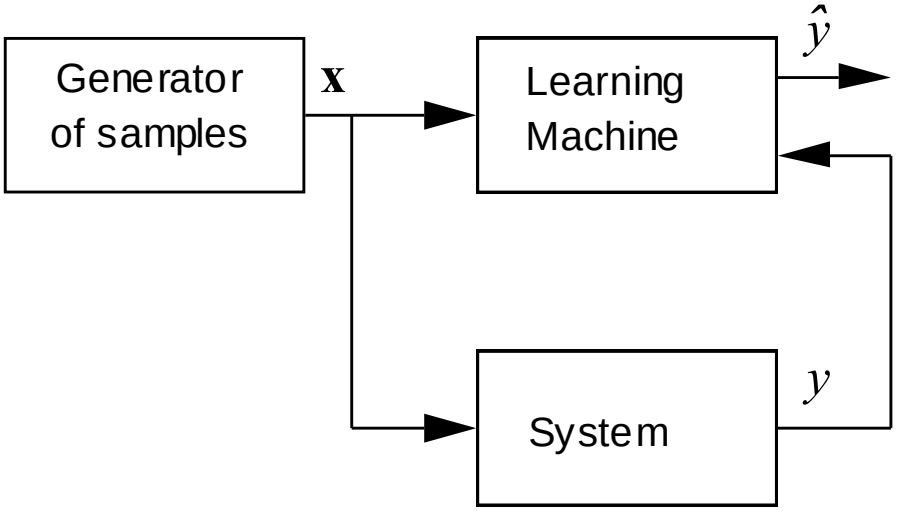
Fold #	k	Validation	Test error
1	7	14.71%	14%
2	31	8.82%	14%
3	25	11.76%	10%
4	1	14.71%	18%
5	62	11.76%	4%
mean		**12.35%**	**12%**

- **Note:** predicted *average test error* (12.8%) is usually higher than *minimized validation error* (10.6%) for model selection, as shown in the first realization

OUTLINE

- Curse of Dimensionality
- Function approximation framework
- Penalization/ regularization inductive principle
- Bias-variance trade-off
- Complexity Control
- **Predictive learning vs. function approximation**
- Summary

Inductive Learning Setting



- Consider regression problem: $y = \boldsymbol{t}(\mathbf{x}) + \xi$
- Goal of function approximation (system identification):

$$\int \left(f(\mathbf{x}, w) - t(\mathbf{x}) \right)^2 d\mathbf{x} \rightarrow min$$

- Goal of Predictive Learning:

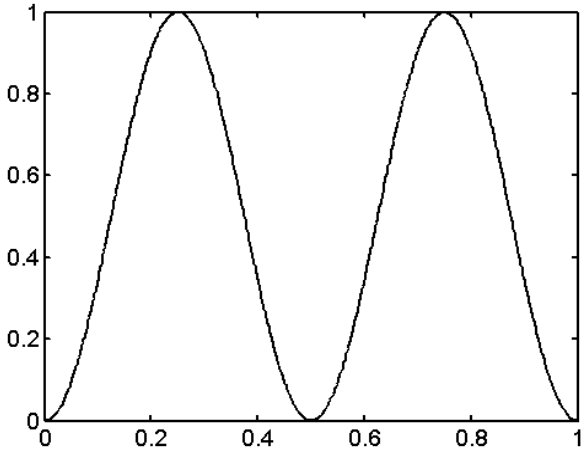
$$\int \left(f(\mathbf{x}, w) - t(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x} \rightarrow min$$

Note: p(x) denotes unknown pdf for the input (x) values.

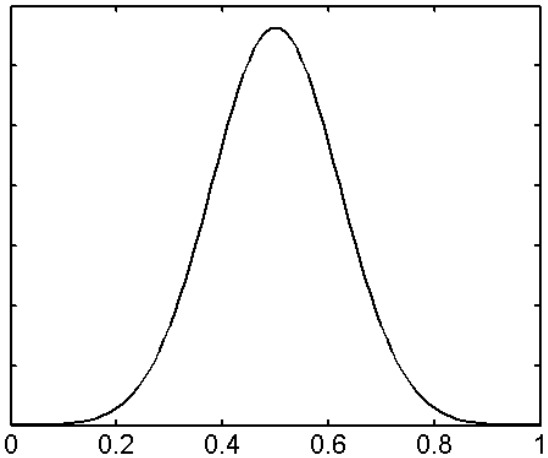
Example from [Cherkassky & Mulier, 2007, Ch. 3]

Regression estimation: penalized polynomials of degree 15); 30 training samples / 30 validation samples (for tuning lambda)

Target function



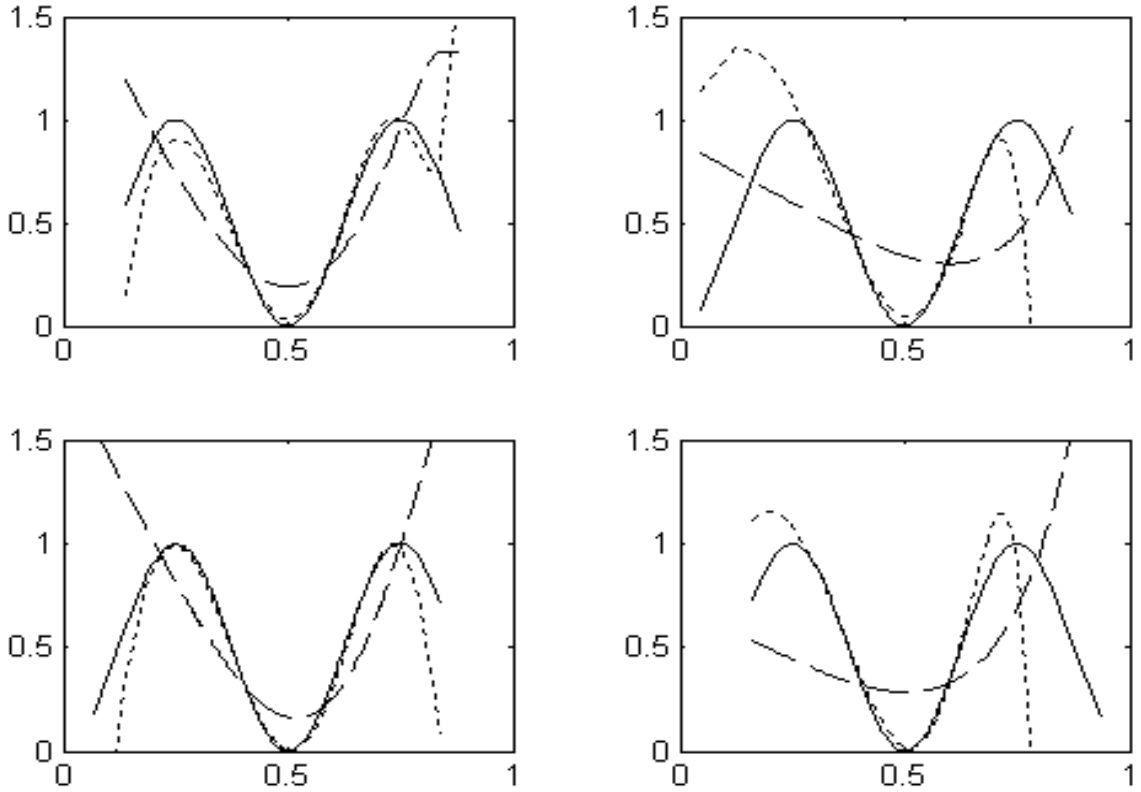
Distribution of x (training data)



Typical Results/ comparisons for

Predictive setting: validation data ~ normal distribution

Function approximation: validation ~ uniform distribution in [0,1]



Dotted line ~ predictive setting / Dashed ~ fct approximation

Conclusion

- The goal of predictive learning is different from function approximation
- Function approximation ~ accurate estimation everywhere in the input domain.
- Predictive learning ~ reduce the overall prediction risk (focus on the data which is more likely to be observed).

Summary

- Regularization/ Penalization ~ provides good math framework for predictive learning

- Important distinction:
  predictive learning vs. system identification

- Bias-variance trade-off

- Complexity control and resampling
  - analytic criteria (typically for regression only);
  - resampling methods (for all types of problems)