Basic Learning Approaches and Complexity Control (2)

CS 6316 – Machine Learning
Fall 2017

3.3 Generalization & Complexity Control

Outline:

- Prediction Accuracy (generalization)
- Complexity Control: examples
- Resampling (K-fold Cross Validation)

Prediction Accuracy

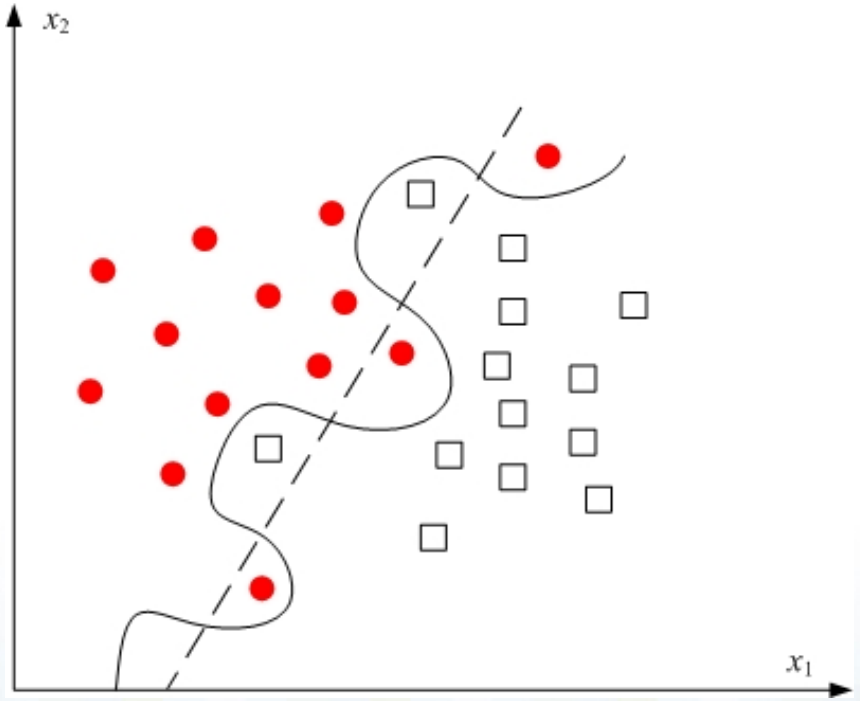
- Inductive Learning ~ function estimation
- All modeling approaches implement 'data fitting' ~ explaining the data
- BUT True goal ~ prediction
- Two possible goals of learning:
  - estimation of 'true function'
  - good generalization for future data

QUESTIONS TO CONSIDER:

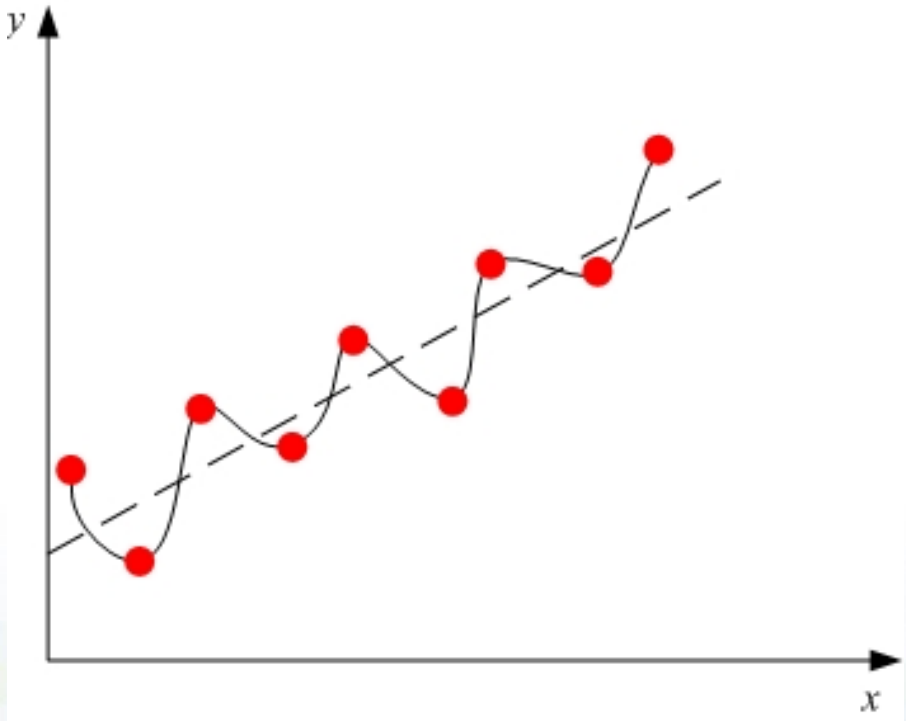
- Are these two goals equivalent?
- If not, which one is more practical?

Explanation vs. Prediction

(a) Classification

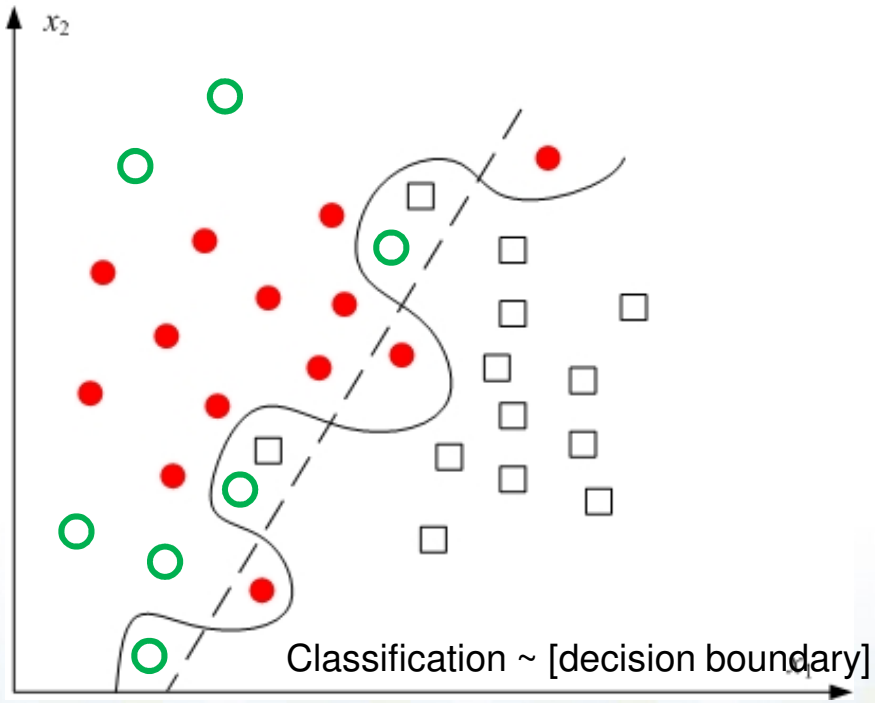


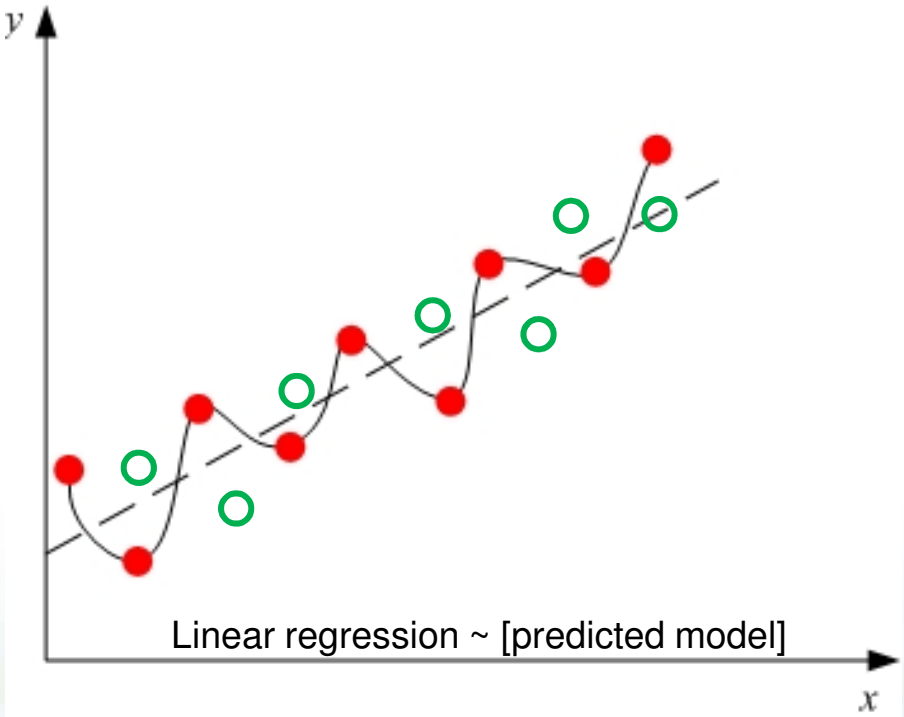
(b) Regression



Explanation vs. Prediction

○ = *New data points ~ class "circle" for (a), data point for (b)*





Inductive Learning Setting

- The *learning machine* observes samples $(\boldsymbol{x}, y)$, and returns an estimated response $\hat{y} = f(\boldsymbol{x}, w)$

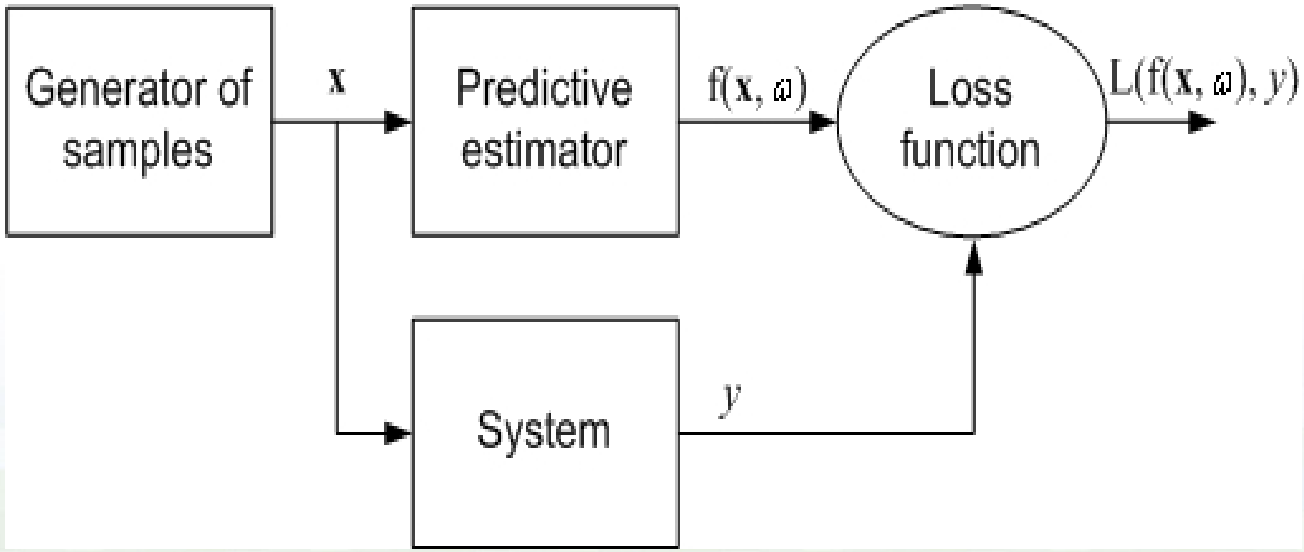
- → Two modes of inference:
  - *Explaining available data* – system identification
  - *Predicting output y for new inputs x* – imitation

  (approximate output)

Inductive Learning Setting

- The *learning machine* observes samples $(\boldsymbol{x}, y)$, and returns an estimated response $\hat{y} = f(\boldsymbol{x}, w)$
- → Two modes of inference: identification vs. imitation

- Risk $\int Loss\big(y, f(\boldsymbol{x}, w)\big) dP(\boldsymbol{x}, y) \rightarrow min$

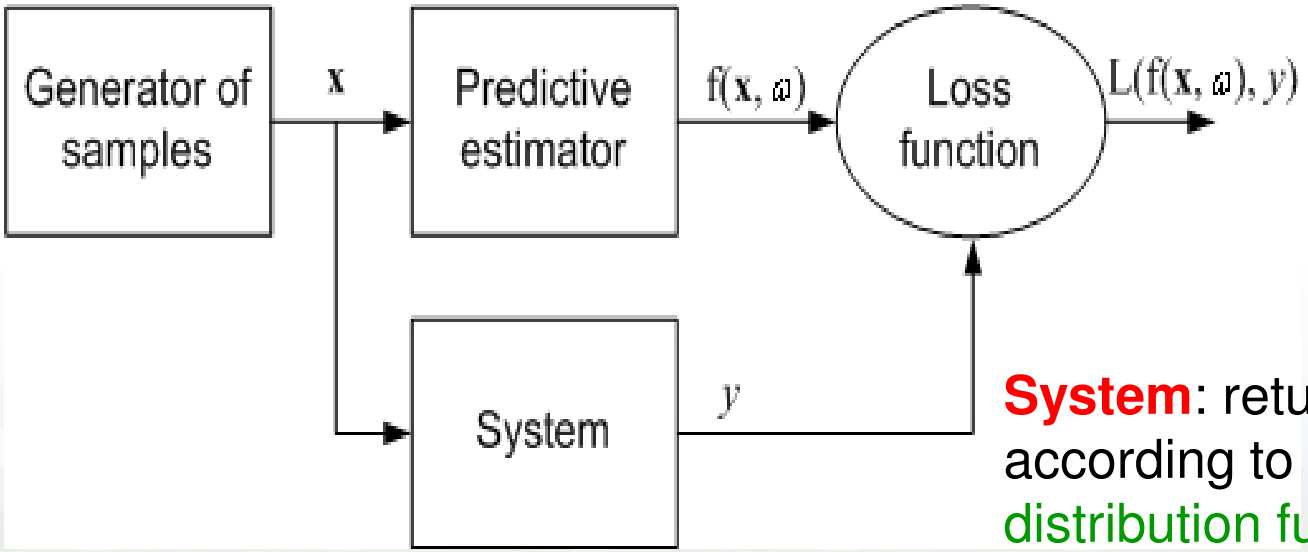


Inductive Learning Setting

Risk: $\int Loss\big(y, f(\boldsymbol{x}, w)\big) dP(\boldsymbol{x}, y) \rightarrow min$

Goal: given function f, loss function L, and probability distribution P(x,y) … minimize risk

Generator: independent identically distributed (i.i.d.) input samples (x) from some fixed (but unknown) statistical distribution, described by probability density function (PDF) **P(x)**



System: returns y (for each x) according to a conditional distribution function **P(y|x)**

3 PROBLEM OF RISK MINIMIZATION

In order to choose the best available approximation to the supervisor's response, we measure the loss or discrepancy $L(y, f(x, w))$ between the response y of the supervisor to a given input x and the response $f(x, w)$ provided by the learning machine. Consider the expected value of the loss, given by the risk functional

$$R(w) = \int L(y, f(x, w)) dP(x, y). \tag{2}$$

The goal is to minimize the risk functional $R(w)$ over the class of functions $f(x, w)$, $w \in W$. But the joint probability distribution $P(x, y) = P(y|x)P(x)$ is unknown and the only available information is contained in the training set (1).

4 EMPIRICAL RISK MINIMIZATION

In order to solve this problem, the following induction principle is proposed: the risk functional $R(w)$ is replaced by the empirical risk functional

$$E(w) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i, f(x_i, w)) \tag{3}$$

Discussion

- Math formulation useful for quantifying
  - explanation ~ fitting error (training data)
  - generalization ~ prediction error
- Natural assumptions
  - future similar to past: stationary $P(\boldsymbol{x},y)$, i.i.d. data
  - discrepancy measure or loss function, i.e. MSE

QUESTION:

- What if these assumptions do not hold?

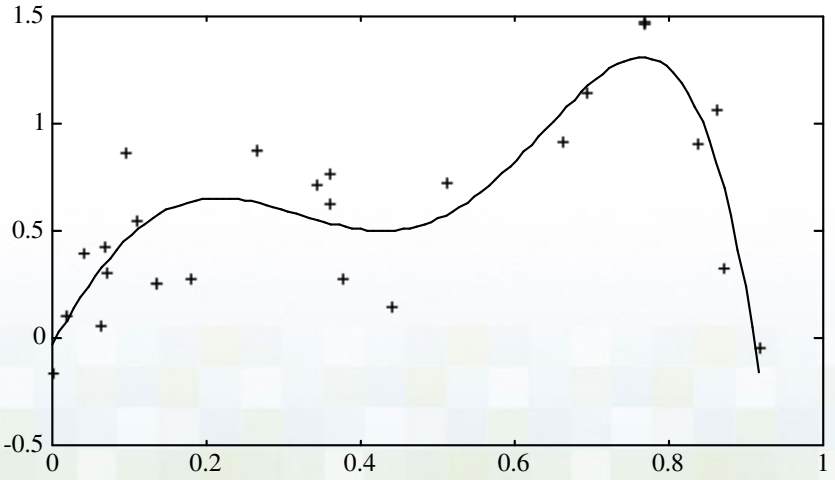
Example: Regression

- Given: training data $(\boldsymbol{x}_i, y_i), i = 1,2, \ldots, n$
- Find: a function $f(\boldsymbol{x}, w^*)$ that minimizes squared error for a large number (N) of future samples:

$$\sum_{k=1}^{N} [(y_k, - f(\boldsymbol{x}_k, w)]2 \rightarrow min$$

$$\int (y - f(\boldsymbol{x}, w))^2 \, dP(x, y) \rightarrow min$$

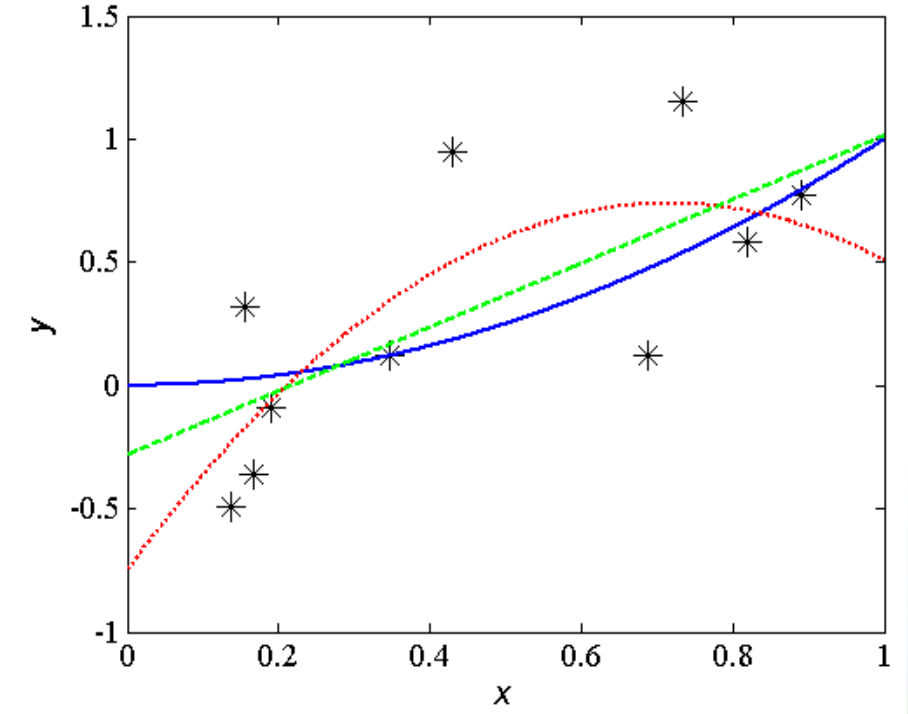
- **BUT** future data is unknown ~ P(x, y) unknown



Complexity Control: parametric modeling

Consider regression estimation

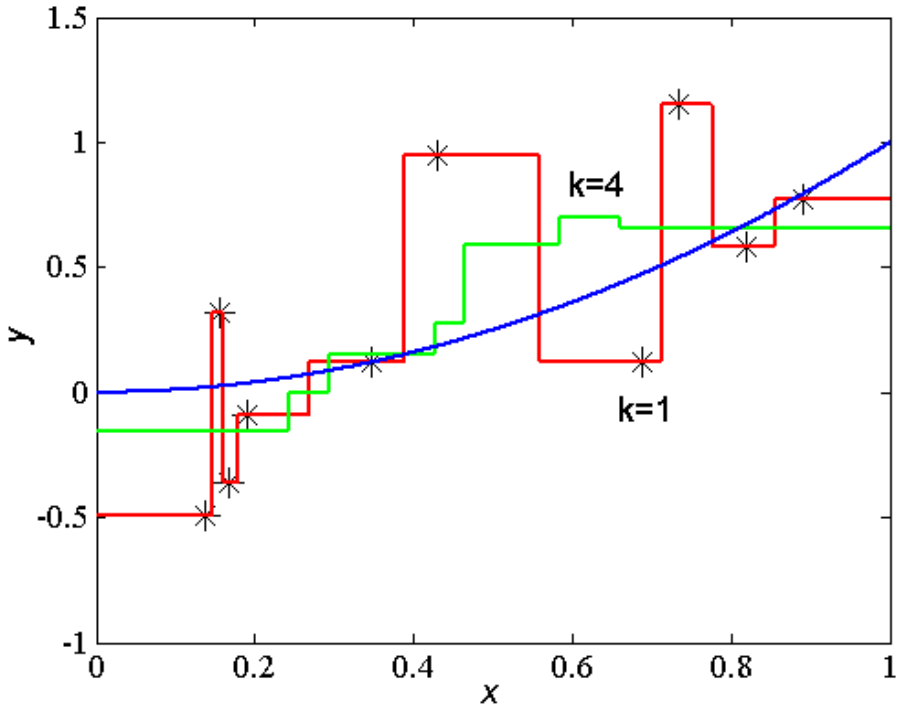
- Ten training samples $y = x^2 + N(0, \sigma^2), where\ \sigma^2 = 0.25$
- Fitting linear and 2nd order polynomial:



Complexity Control: local estimation

Consider regression estimation

- Ten training samples $y = x^2 + N(0, \sigma^2), where\ \sigma^2 = 0.25$
- Using K-NN regression with k=1 and k=4



Complexity Control (cont'd)

- Complexity (of admissible models) affects generalization (*for future data*)
- Specific complexity indices for
  - Parametric models: ~ # of parameters
  - Local modeling: *size of local region*
  - Data reduction: # of clusters
- Complexity control = choosing good complexity (~ good generalization) for a given (training) data

How to Control 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 = $\frac{DoF}{n}$, n ~ sample size, DoF ~ degrees of freedom

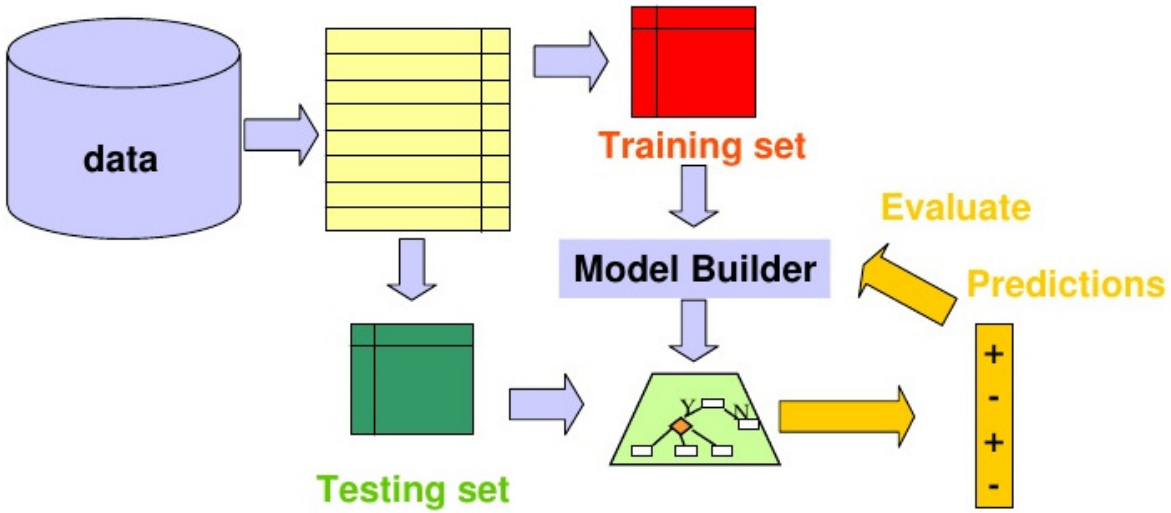
Predictive Learning Experiment

- There are two steps in predictive learning:

1. [TRAINING (+ TESTING)] Learning (estimating) unknown dependencies or rules from data samples (using "training data") and assessing the quality of these rules to obtain their "prediction accuracy" (using "testing data")

2. [PREDICTING] Using dependencies/rules learned in (1) to predict output(s) for future input values

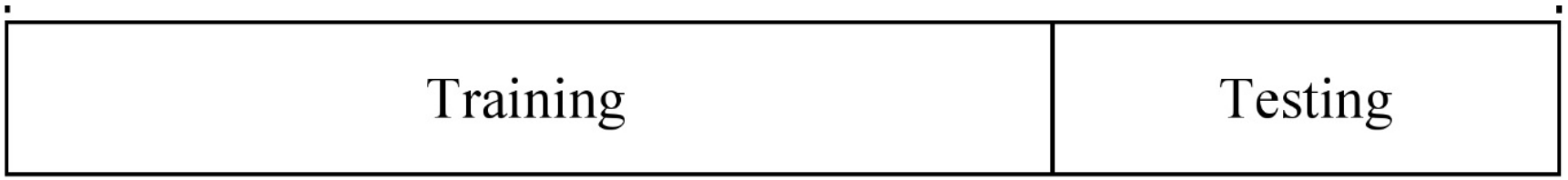
Training and Test Splits



Resampling

- Split available data into 2 sets: Training + Testing

  (1) Use **training** set for model estimation (via data fitting)
  (2) Use **testing** data to estimate the prediction error of the model
- Change model complexity index and repeat (1) and (2)
- Select the final model providing lowest (estimated) prediction error



Predictive Learning Experiment: Notes

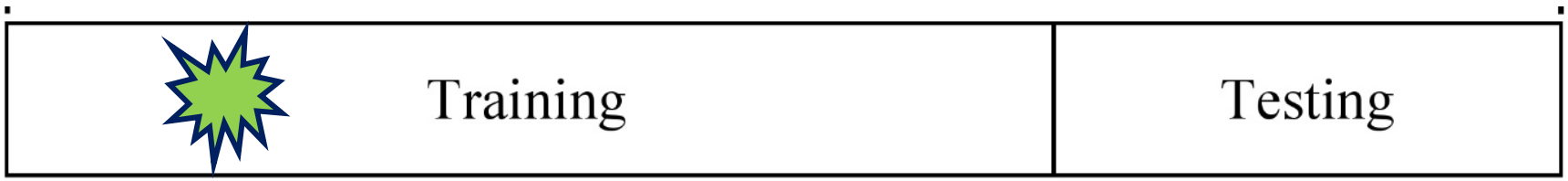
- Training data is typically 80% of the original data set and Testing data is the remaining 20%
- The Training data is always separate from the Testing data!
  - The intersection of these two sets should be *null*
  - Would be experimentally **inaccurate** to test the accuracy of the rules produced by an algorithm on the same data that was used to train/learn the rules in the first place (there would be a **large bias** which would artificially **raise** the prediction accuracy)

Problems?

- Might not have enough data (sacrificing, for example, ¼ of the data for testing might be too costly, resulting in too few training data samples. Too few data samples for training means the algorithm would have a difficulty in learning – would not have enough information to learn the patterns and regularities
- Happen across an unfortunate split! Results are sensitive to *data splitting*
  - Nature of the data plays a part
  - Anomaly in the data

Anomaly

- Nature of the data plays a part
- Sensitive to chosen data split



Solution?

- The **nature of the data** in the training set might make a different in the accuracy of the rules produced
  - To **mitigate** the variance in the training data **k-fold cross-validation** is employed
    - Typical values of 'k' are 4, 10, and 20. Other values of k can be used (especially larger than 20) if the original data set is very large
    - Picking a higher value of k doesn't always give you better results, however, picking a value that is too small (especially relative to the size of the data) may not be very helpful

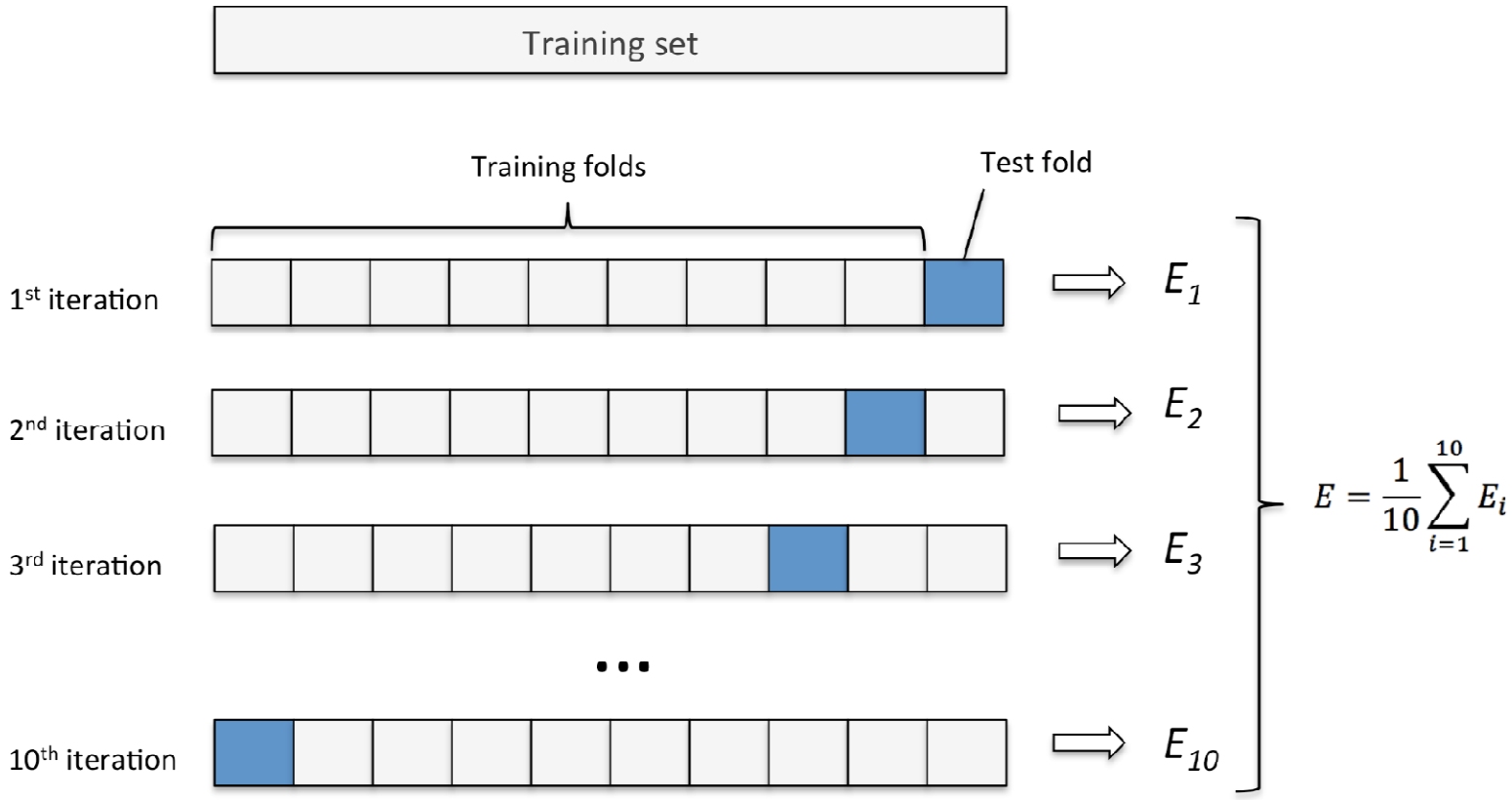
K-fold Cross-Validation

1. Divide the training data $\mathbf{Z}$ 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 $\mathbf{Z}_i$:

   - use remaining data to estimate the model $\hat{y} = f_i(\mathbf{x})$

   - estimate prediction error on $\mathbf{Z}_i$: $r_i = \frac{k}{n}\sum_{\mathbf{z}_i}\left(f_i(\mathbf{x}) - y\right)^2$
3. Estimate avg prediction risk as $R_{cv} = \frac{1}{k}\sum_{i=1}^{k} r_i$

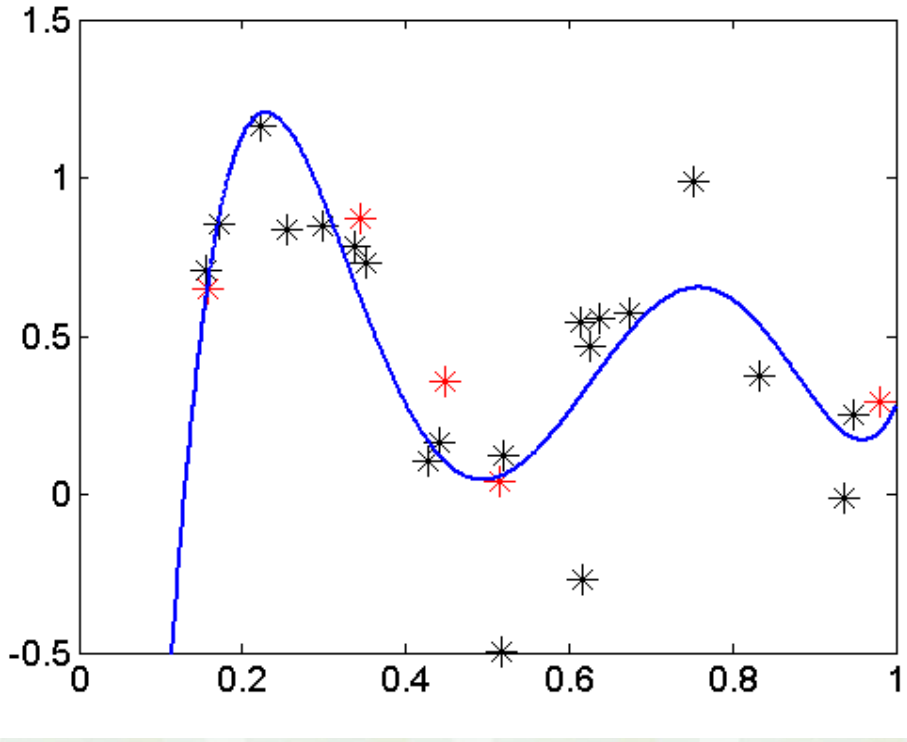
K-fold Cross-Validation



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 testing (*) data points, for one partitioning

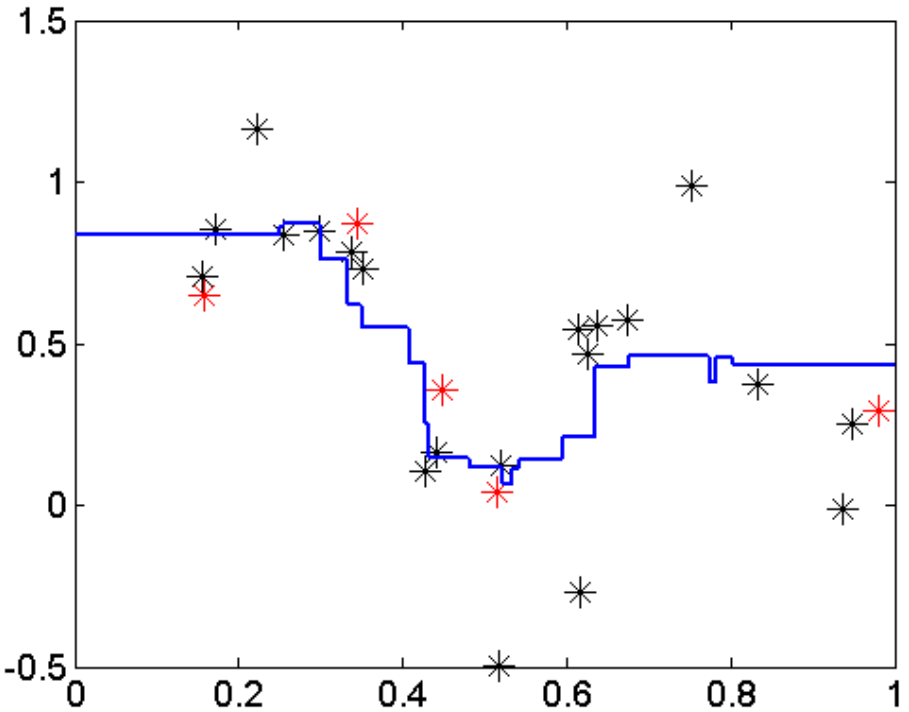
m	Estimated R via Cross validation
1	0.1340
2	0.1356
3	0.1452
4	0.1286
5	**0.0699**
6	0.1130
7	0.1892
8	0.3528
9	0.3596
10	0.4006



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 testing (*) 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)

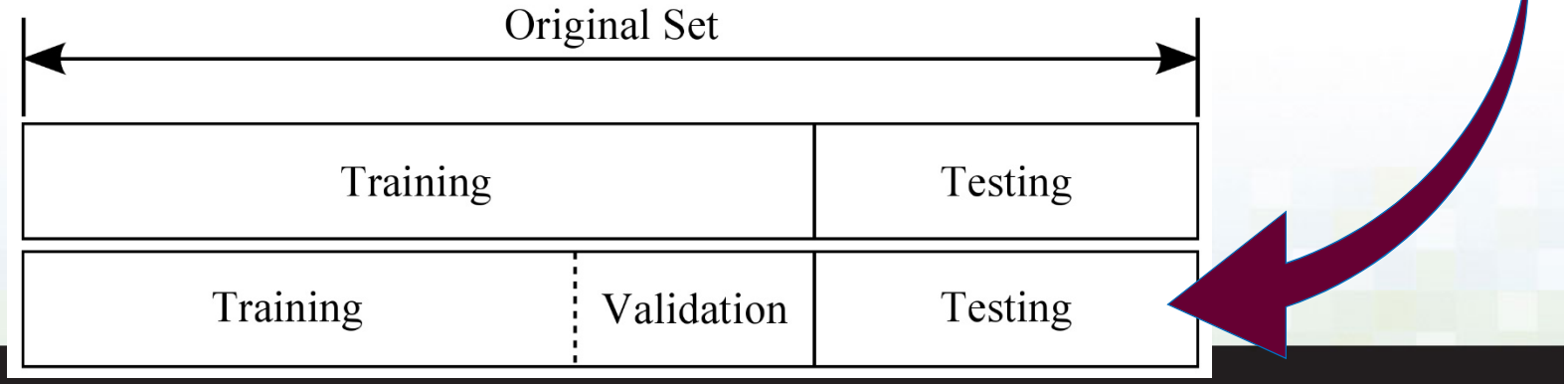
  → the same resampling procedure may yield different model selection results
- Some applications may use non-random splitting of the data into (training + testing)
- Model selection via resampling is based on estimated prediction risk (error).
- Does this estimated error measure reflect true prediction 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 **never** be used for model estimation
- DOUBLE RESAMPLING method:

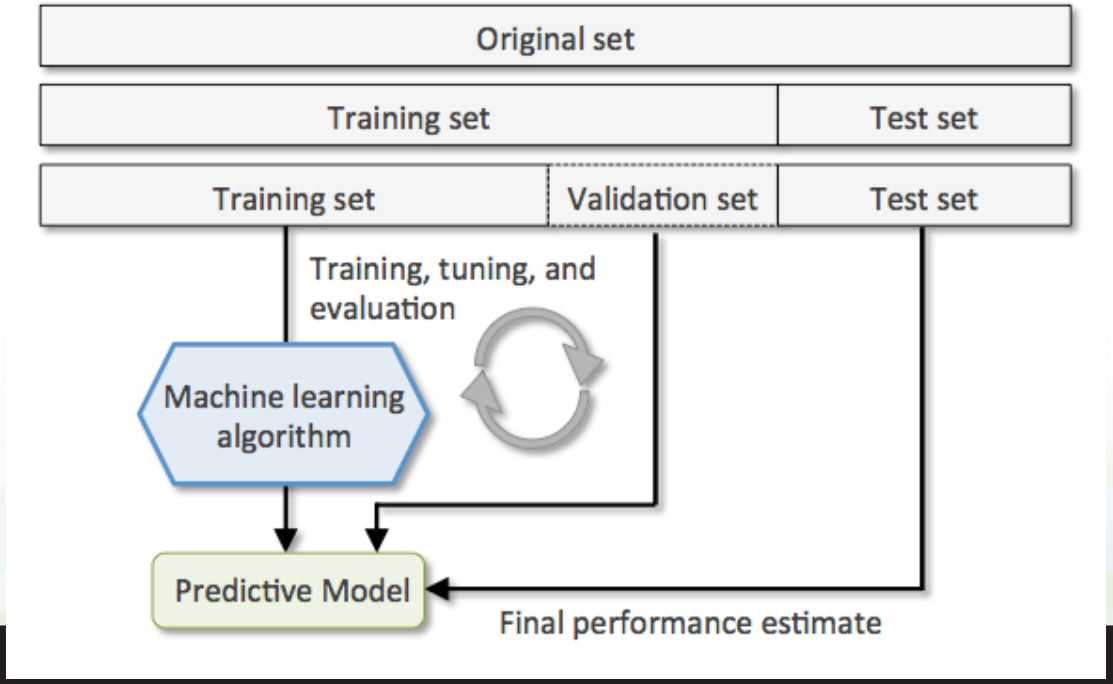
  - for complexity control

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



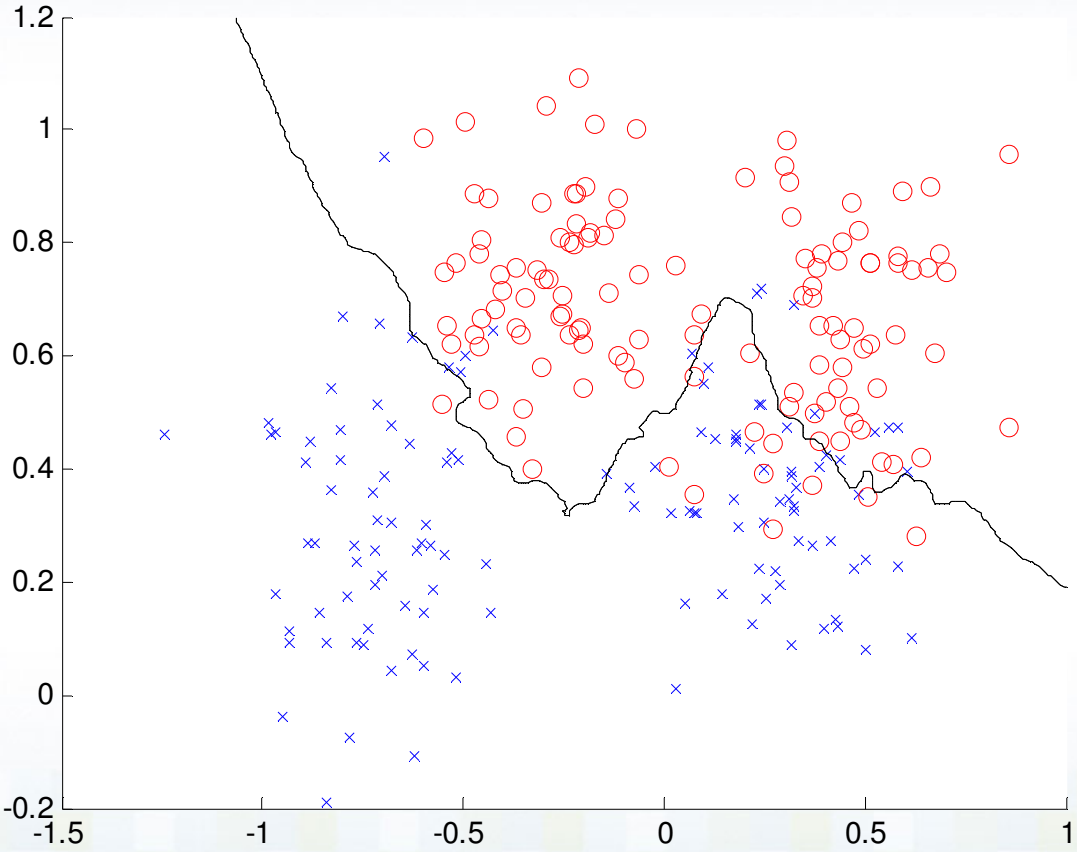
Training and Test Splits

- The data is divided into three parts
  - training set for training model
  - validation set for model selection and parameter adjustment
  - test set for final generalization capability assessment model



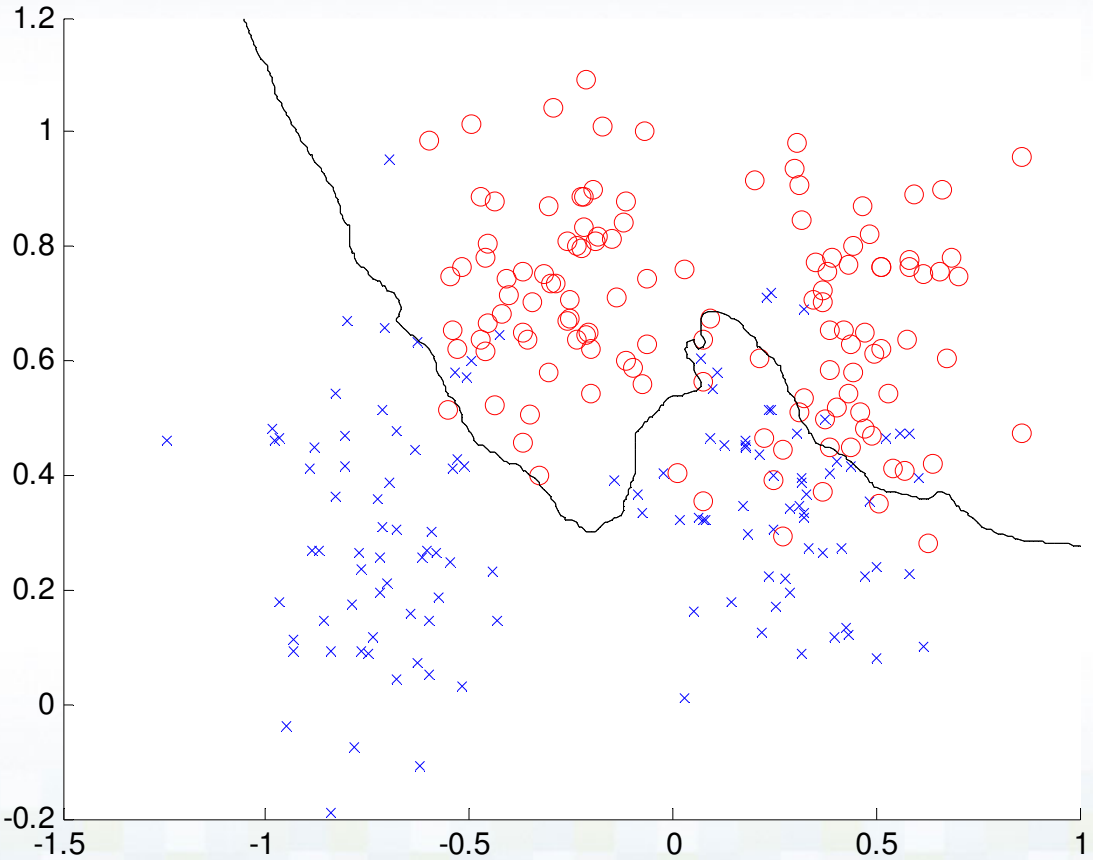
Example of model selection for k-NN classifier via 6-fold x-validation: Ripley's data. Optimal decision boundary for k=14

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:

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%**

*[For each one of these folds, **6-fold cross-validation** was performed to choose optimal value of 'k' – in this case, optimal k = 20]*

Note 1: *optimal* 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

- 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:

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%**

Note 2: there may be a large **variability** in validation and test errors observed for individual folds of the data.

This variability motivates the use of K-fold cross-validation.

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%) is usually higher than *minimized test error* (11%) for model selection

What Else Can You Do With The Data?

- ***Balanced Splits***

- Other than k-fold cross-validation you can ensure all splits of the data when performing cross-validation are ***balanced*** (based on class distribution)

- Often called "stratified" k-fold cross-validation
  - Maintain the proportion of the classes in each fold – so as to obtain a more accurate evaluation model

Balanced Folds? (for Cross Validation)

- Ensure ***balanced*** folds (based on class distribution)
  - Balanced in terms of even class representation in each split/fold. If this doesn't happen then the algorithm may only learn how to classify some of the classes (e.g. if for some reason in the train split, only two out of the three classes are represented, then this algorithm will perform poorly because it wouldn't know how to correctly classify data items from the third class.)
  - In short, all folds should have examples from *all* classes
    - All folds should have the same ratio of each of the classes too, if possible. May need to artificially boost an underrepresented class so that the number of data items matches the number of data items in the other classes