STA314, Lecture 2

Sept 17, 2018

Stanislav Volgushev

Recap from previous lecture

K-nn estimator

$$\widehat{f}(x_0) = \frac{\sum_{i=1}^{n} Y_i I\{x_i \text{ among closest K to } x_0\}}{K}$$

Bias-Variance decomposition

$$\mathbb{E}[(\hat{Y}_0 - f(x_0))^2] = \mathrm{Var}\,(\hat{Y}_0) + \{\mathbb{E}[\hat{Y}_0] - f(x_0)\}^2.$$

Bias and variance of K-nn (derived in lectures for specific setting)

$$\mathrm{Var}\,[\hat{f}(x_0)] = \sigma^2/K,$$

$$\mathbb{E}[\hat{f}(x_0)] - f(x_0) \approx \frac{1}{24} f''(x_0)(K/n)^2.$$

Training error (in-sample prediction error) :

$$MSE_{train} = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}(x_i) - Y_i)^2$$

not useful for selecting k in k-nn (not useful for selecting tuning parameters in general).

This lecture: better alternatives to training error.

Test error

- what we are really interested in: error when predicting outcomes for *new observations* that were not used to learn $\hat{f}$.
- this is called *test error (out-of sample prediction error, generalization error)*.
- if we had an additional sample, called *test sample*, $(x_1^{te}, y_1^{te}), ..., (x_N^{te}, y_N^{te})$ which was not used to build $\widehat{f}$, we could try to approximate this error by

$$MSE_{test} = \frac{1}{N} \sum_{i=1}^{N} (\widehat{f}(x_i^{te}) - y_i^{te})^2.$$

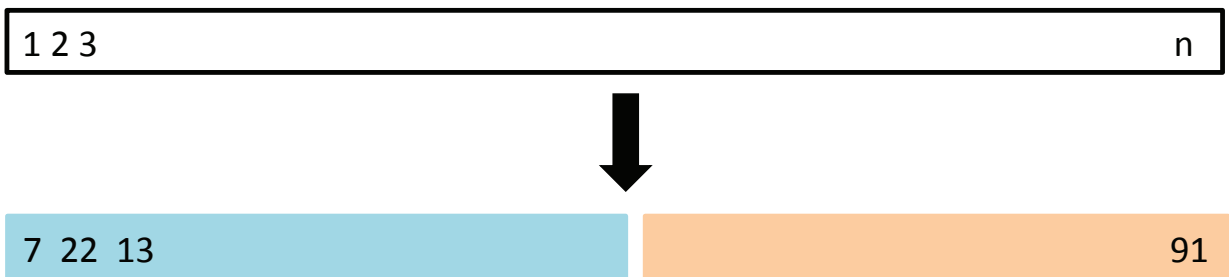
- if we knew the data-generating process, we could simulate additional data to obtain a test sample.
- in real-world scenarios we can not obtain more data, or only do so at additional cost. We have to work with the available sample.

The validation set approach

In real-world scenarios we can not obtain more data, or only do so at additional cost. So what do we do about it?

One possible idea: split the available data set into two parts, estimate $\widehat{f}$ from one part (*the training set*) and use the other part (*validation set*) to select tuning parameters.

The simplest version of this is to split into two (approximately) equal parts. This is sometimes called *validation set approach*.



Potential problems:

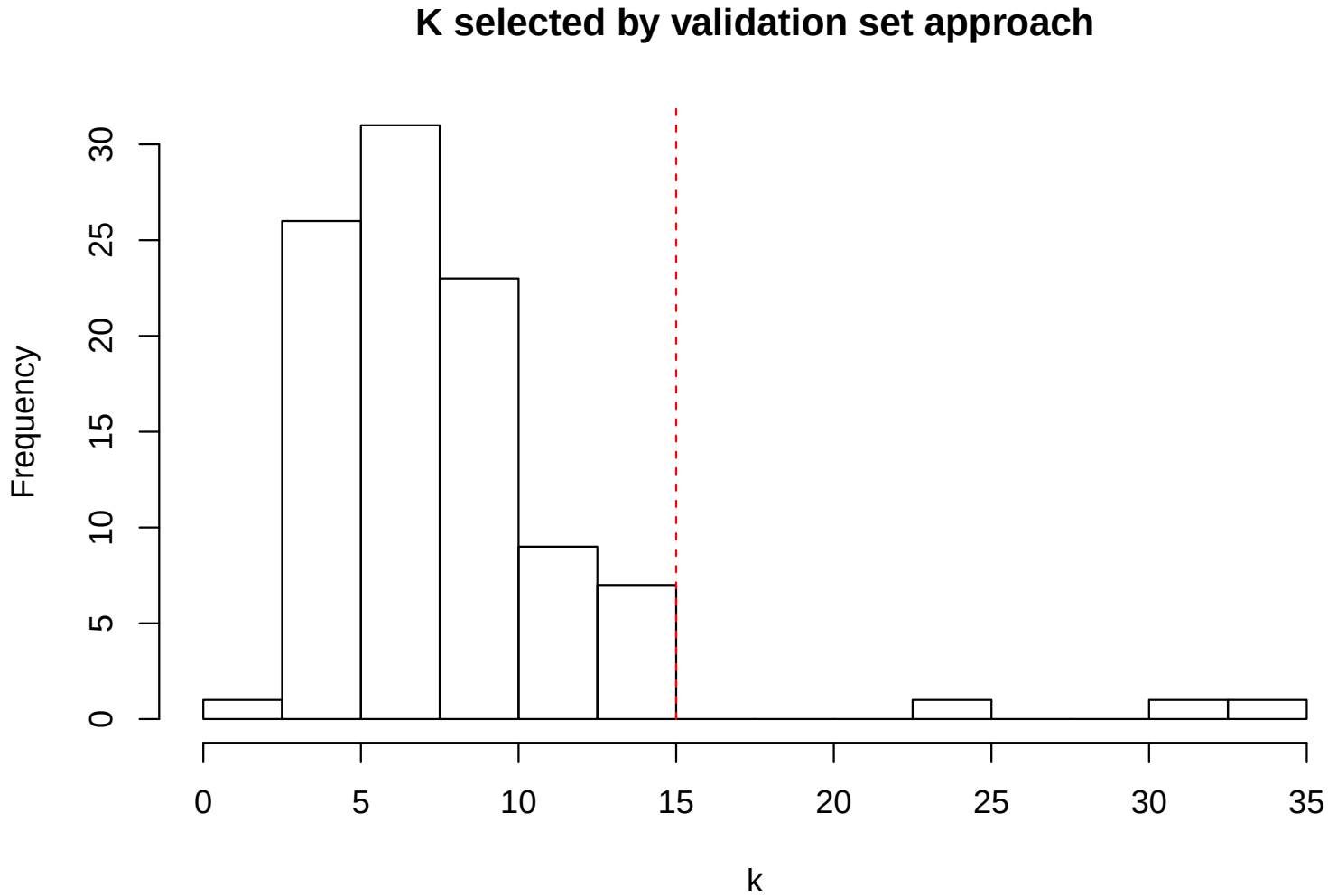
1. Since the split is random the answer we get might strongly depend on the split.
2. Only half of the data are used for learning the model. Usually more data for learning means better performance – splitting half-half might give biased answers.

Example of a function in R: implementing the validation set approach

```
# this is a function with three inputs: X,Y,range of values for k
valset_knn = function(X,Y,krange){
  n = length(Y)
  l.tr = floor(n/2)
  l.val = n - l.tr
  train_index = sample(1:n,l.tr) # randomly sample l.tr points
  # this will correspond to the training set

  # create array where results of validation error will be stored
  valset_error = array(0,length(krange))
  # loop over values of K, for each store error on validation set
  for(k in 1:length(krange)){
     K = krange[k]
     # only use data with index in train_index to fit the model
     fit = knnreg(as.matrix(x[train_index]),y[train_index],k=K)
     # now use fitted model to predict data which are not in train_index
     pr = predict(fit, newdata = data.frame(x = x[-train_index]))
     # compute and store the validation error
     valset_error[k] = mean((y[-train_index] - pr)^2)
  } # end loop over k
  return(valset_error) # this will be function output
} # end of function
```

The validation set approach: example



Histogram after 100 simulation repetitions. Red line: k that gives optimal MSE (from previous simulation). Observation: tends to select too small k. This is consistent with k-nn theory (can you derive why?).

Improving the validation set approach

Problem: the split is random so the answer we get might strongly depend on the split.

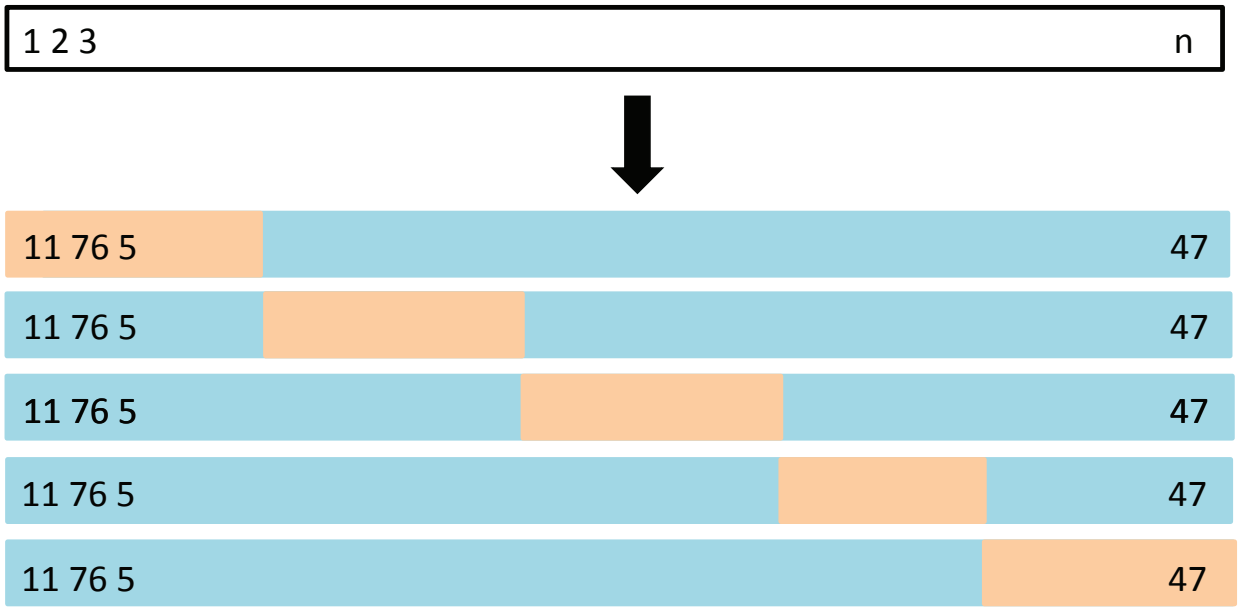
- Perform averaging over several random splits.

Problem: Only half of the data are used for learning the model. Usually more data for learning means better performance, so splitting half-half might give biased answers. Specifically for k-nn regression: k will usually be too small.

- Use a larger portion of data for learning and a smaller portion for validation.

The above ideas are combined in **K-fold cross-validation**.

K-fold cross-validation



1. Randomly split $\{1, ..., n\}$ into non-overlapping subsets (also called *folds*) $S_1, ..., S_K$ of roughly equal size.
2. For ℓ taking values $1, ..., K$: use data from all subsets except S_ℓ to estimate f, call estimators $\widehat{f}^{(\ell)}$.
3. The cross-validated error is

$$\frac{1}{n} \sum_{\ell=1}^{K} \sum_{j \in S_\ell} (\widehat{f}^{(\ell)}(x_j) - y_j)^2.$$

Comments on cross-validation

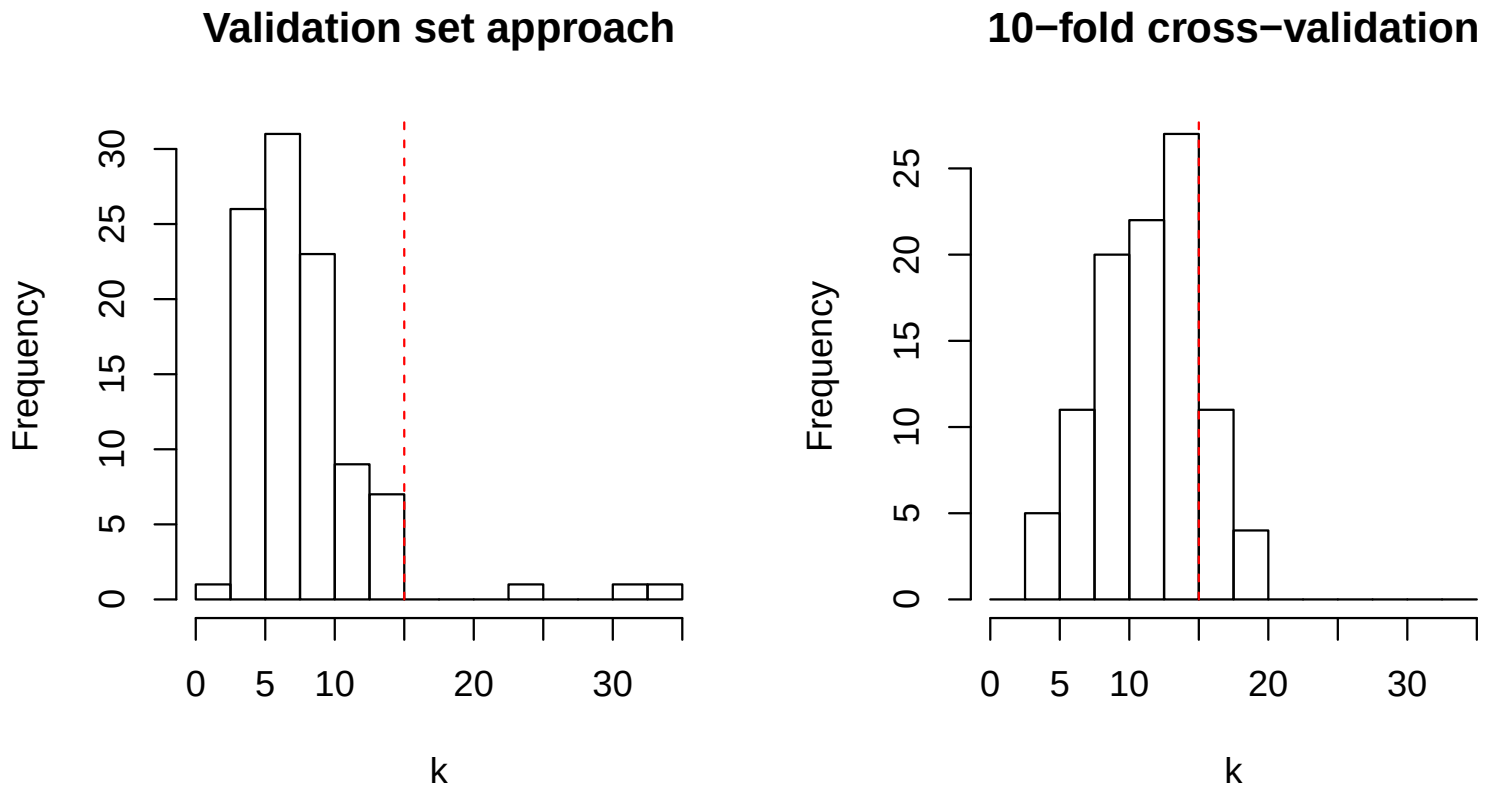
So how do we choose the number of folds? There are several aspects to consider...

- A small number of folds corresponds to more variation and smaller training sample sizes. Might result in highly variable and distorted results.
- Choosing more folds means more computation. Most extreme case: leave out one observation at a time, also known as *leave one out cross validation, LOOCV*
- LOOCV does not involve randomness, so for a single data set we get a single answer regarding the best model.
- Outside of special models with computational short-cuts, LOOCV involves a lot of computation – see calculations on blackboard for a example.
- Theoretical comparisons show that LOOCV is not the best approach. We will not do theory, but we will run some computer experiments to compare the performance of different methods.

The 'rule of thumb' recommended in textbook and elsewhere: usually 5 or 10-fold cross-validation gives a good compromise.

Let's try some R code!

Comparison of validation set approach and 10-fold CV



Comparison based on same data sets. 10-fold cross-validation does better.

- MSE of k-nn with with different ways to select k: 4.41 (validation set approach), 3.27 (10-fold CV), 2.42 (optimal k based on simulations).
- Advantage in selecting k carries over to MSE, but not as good as optimal k.

The one standard error rule I

- Recall: for each value of k, K-fold cross-validation returns estimated errors on each of the K folds. Call those errors $e_1(k), ..., e_K(k)$.
- Those errors $e_1(k), ..., e_K(k)$ are random variables (random splits, randomness in data...).
- To reduce noise in model selection, and to select simpler models, one can use the 'one standard error rule'.

---

1. Find k_0 which minimizes $K^{-1} \sum_{j=1}^{K} e_j(k)$ (classical cross validation would output this as solution).
2. Compute $\widehat{sd}(k_0)$, the sample standard deviation of $e_1(k_0), ..., e_K(k_0)$.
3. Select the largest k^* among those k which satisfy

$$K^{-1} \sum_{j=1}^{K} e_j(k) \leq K^{-1} \sum_{j=1}^{K} e_j(k_0) + K^{-1/2} \widehat{sd}(k_0)$$

See picture on blackboard for more explanations.

The one standard error rule II

1. Find k_0 which minimizes $K^{-1}\sum_{j=1}^{K} e_j(k)$ (classical cross validation would output this as solution).
2. Compute $\widehat{sd}(k_0)$, the sample standard deviation of $e_1(k_0), ..., e_K(k_0)$.
3. Select the largest k^* among those k which satisfy

$$K^{-1}\sum_{j=1}^{K} e_j(k) \leq K^{-1}\sum_{j=1}^{K} e_j(k_0) + K^{-1/2}\widehat{sd}(k_0)$$

- This idea is applicable to any problem when selecting tuning and will appear again.
- Will tend to select 'simpler' models than classical cross-validation (never selects more complicated models), i.e. larger k for k-nn. Good for building models that are more interpretable.
- Recommended in textbook and other places, but bot universally accepted. No clear theoretical justification.

Model assessment

- k-nn: a way to learn regression function $\hat{f}$ from data. Depends on tuning parameter k.
- Way to select tuning parameter: different variants of cross-validation. Applicable to k nearest neighbours and to selecting tuning parameters in any other methods.
- So far we have evaluated performance of methods based on simulation or our knowledge of the true regression function.

Next questions:

- How do we compare the performance of methods on real data where it is impossible to simulate new data and we don't know the true regression function?
- How will our selected method perform on 'new' data that were *never* used in the whole training process (including model or tuning parameter selection)?

The second question is related to *model assessment*. It can be important, for instance when we report the performance to collaborators.

Protocol for evaluating methods on a real data set

Step 1: randomly divide the data set into a training and test set. Set the test set aside.

Step 2: use the training set to fit candidate models. This includes selecting tuning parameters such as K in K-nn regression via cross-validation. *Do not use any data from the test set for this step.*

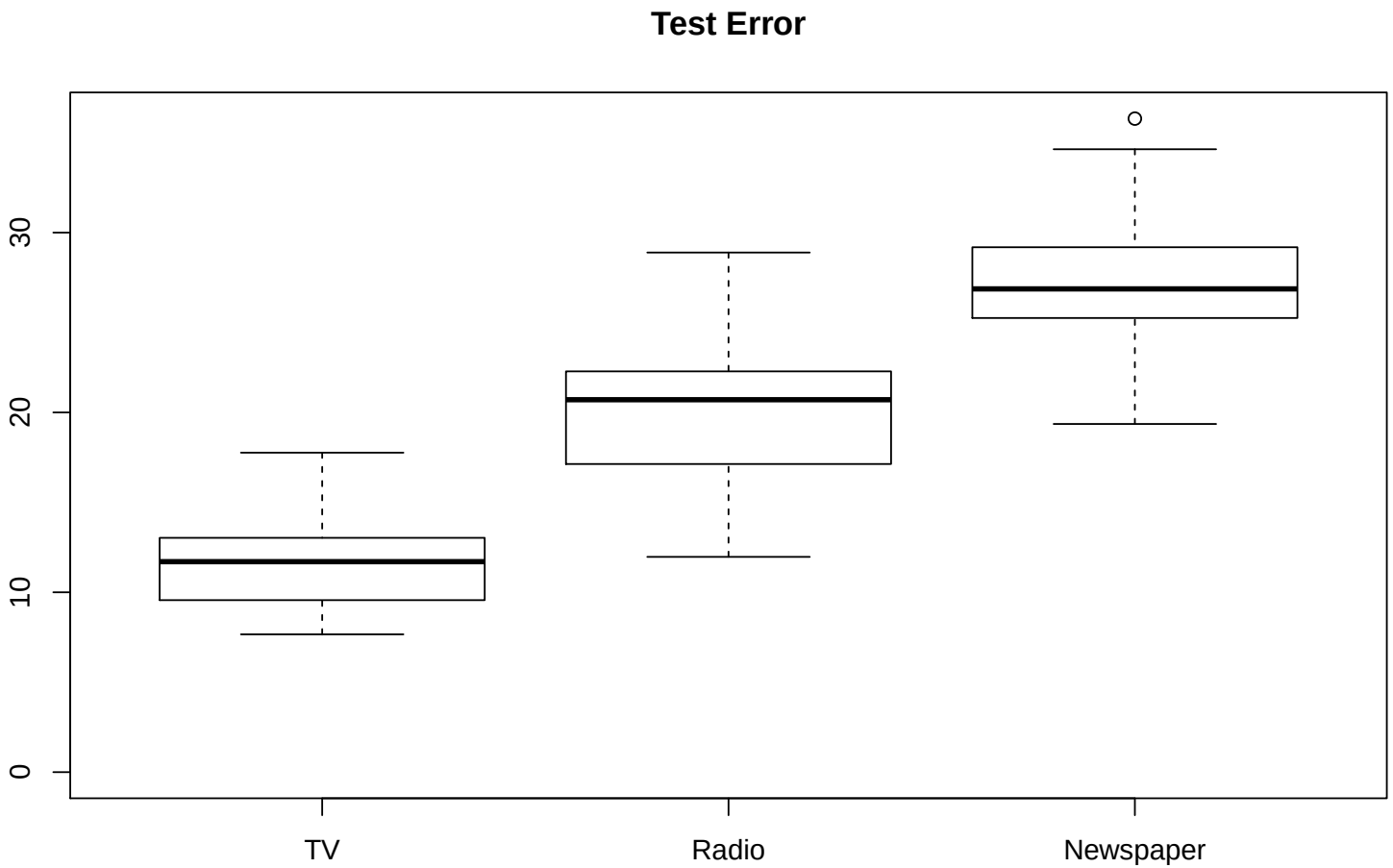
Step 3: Use the final models obtained in Step 2 to predict data in the test set and compute the test error. Use this error to compare models without further tuning.

See picture on blackboard.

- Setting aside around $20\% - 30\%$ of the data for the test set is typical.
- **The test set should never be used to tune parameters.** It appears **only** at the final stage when models are compared.
- Since the split into test and training set is random, the above procedure will produce results that depend on the split. To get a better idea of the overall performance of our model we will use repeated splitting in lectures. However, this is not standard and often not feasible.

Example: evaluating the performance of different models for Sales

Setting: run k-nn with k selected by cross-validation using TV, Radio or Newspaper to predict Sales. Split data into 20% test and 80% training once with given seed (see R code). Gives estimated errors: 14.3 (TV), 24.8 (Radio), 36.3 (News), 39.0 (mean).



Plot shows errors resulting from 25 different (random) splits.

K-nn regression continued

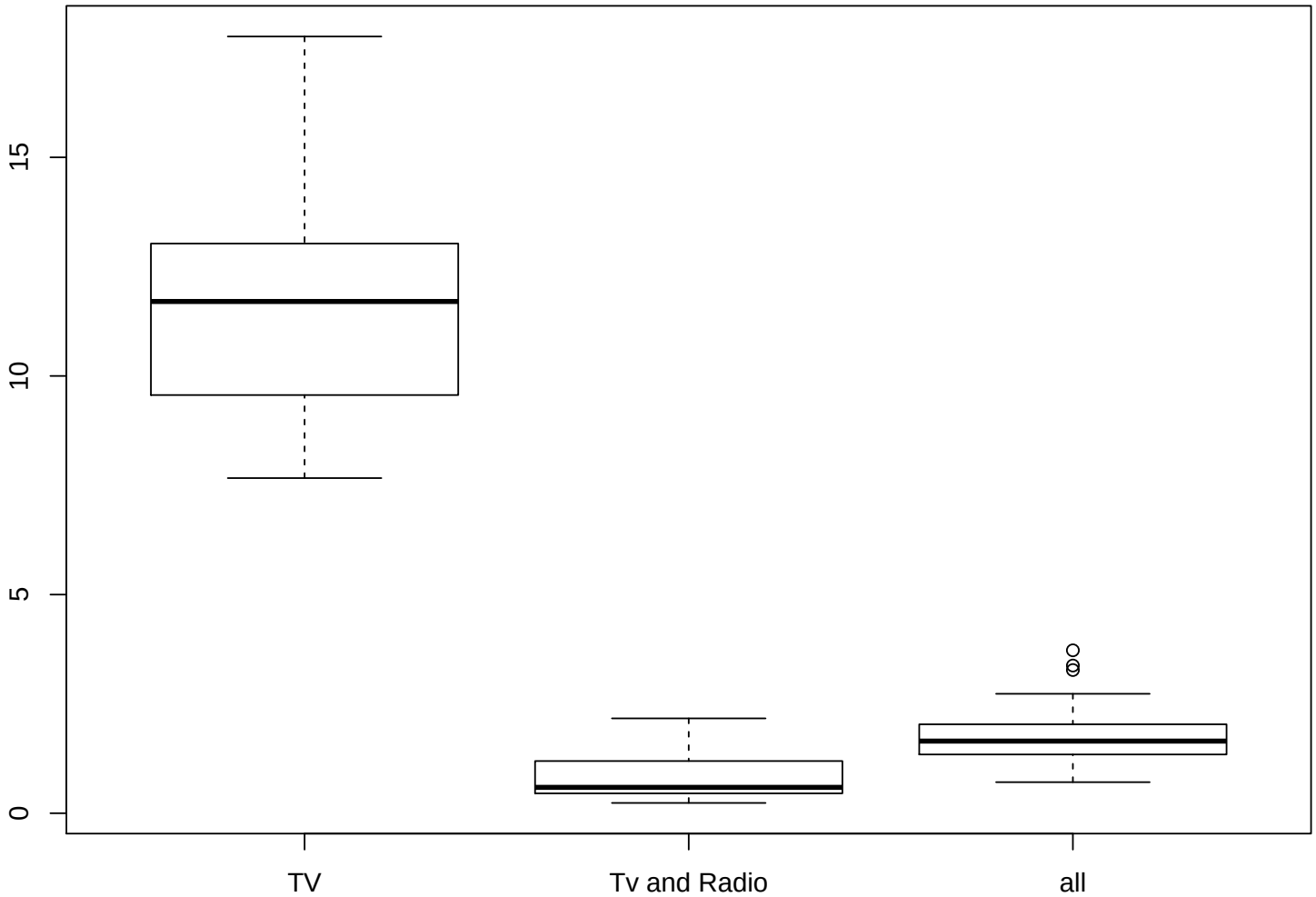
An idea to try to improve results on the previous slide: use several predictors simultaneously!

- Basic idea of k-nn: pick the k closest points for prediction. Using any distance in $\mathbb{R}^d$ this idea can be applied to multivariate predictors. Typical: use Euklidean distance.

- Some care must be taken with scale of individual predictors. Example: a maximal amount of 300 units was spent on TV advertisement, but only a maximal amount of 50 units on Radio advertisement.

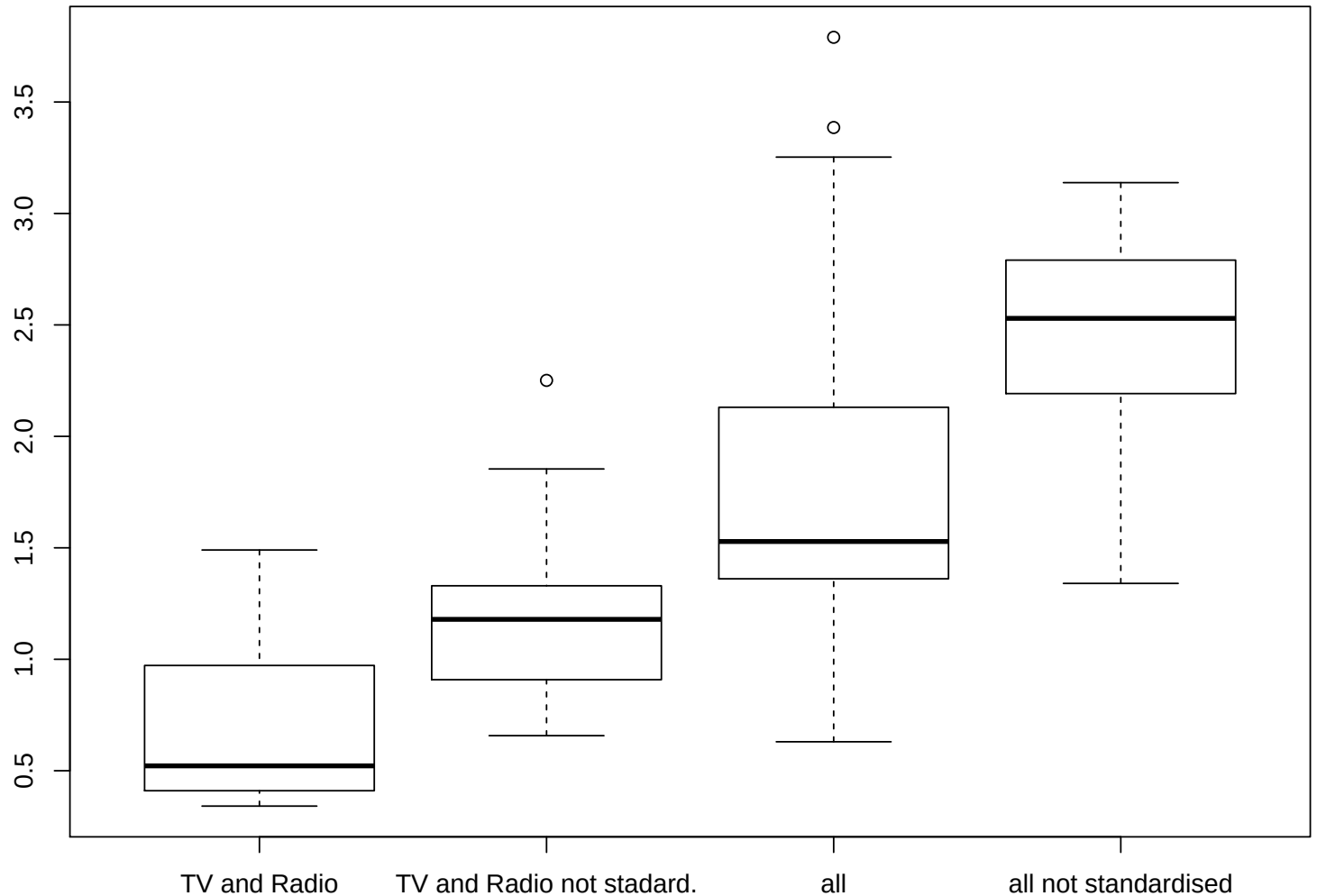
- Unless there are good reasons to not do that, all predictors should first be standardised to have the same scale.

Advertising: K-nn with several predictors



k selected by 10-fold cross-validation. Including Radio and TV in the same model leads to substantial improvement. Additionally including Newspaper makes things worse.

Advertising: K-nn with several predictors, scaling vs no scaling

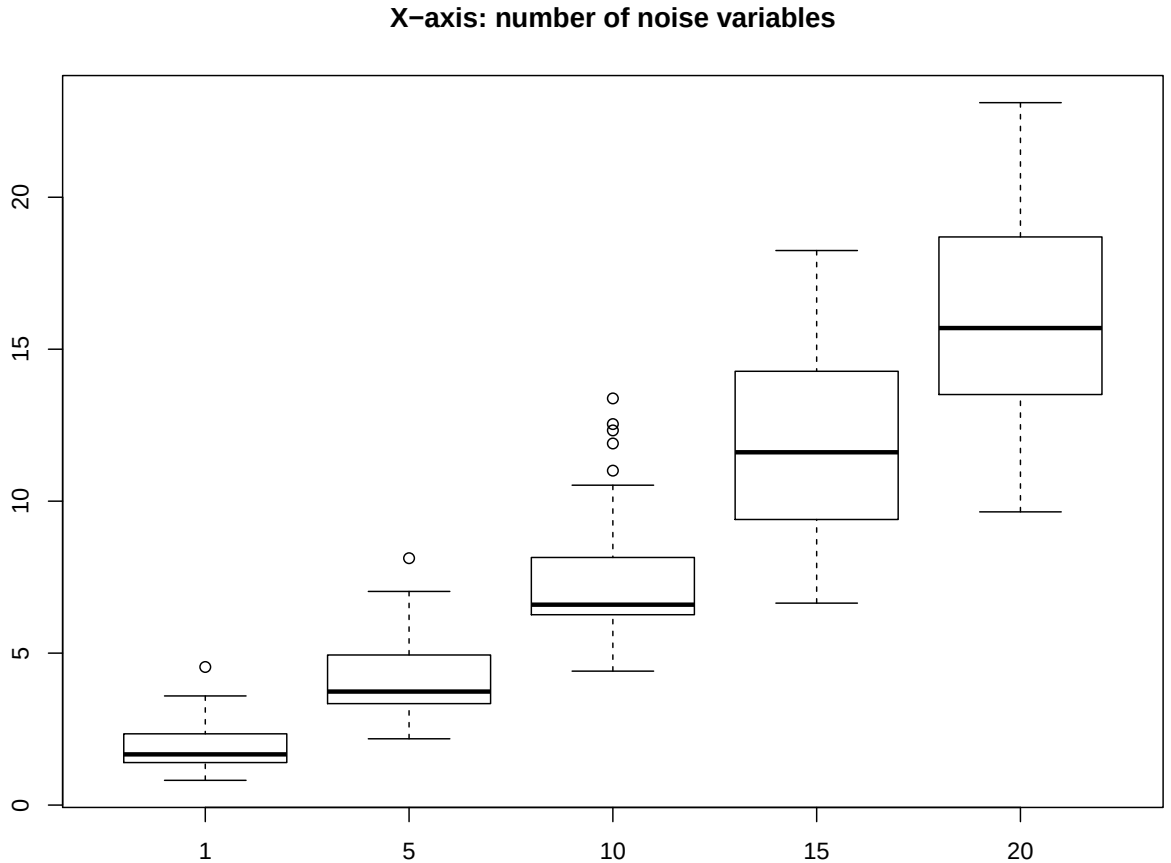


Same as before, comparison of using scaled and not scaled predictors. Scaling predictors leads to better results in this example.

Problem with K-nn regression: selecting relevant predictors

In the Sales data, we saw that selecting 'the right' predictors is important. There, it was possible to look at all combinations of predictors because we had just 3. But what if we have many?

Example below: performance of k-nn with cross-validation when different number of 'noise' predictors (random, independent of Sales) are added to data (see R examples).



Concluding remarks on K-nn regression

Pros

- Does not make any structural assumptions on 'true' function f. This is called *non-parametric method*. Can approximate fairly different kinds of regression function.

Cons

- Not very interpretable, 'black box' method.
- Does not produce 'smooth' functions $\widehat{f}$, in fact $\widehat{f}$ are step functions.
- Does not work well when there are 'many' (irrelevant) predictors: curse of dimensionality.
- Does not allow to easily select relevant predictors: in the sales example we simply tried all possible combinations. Not possible when there are 'many' predictors.
- Difficulties with extension to *qualitative* predictors: e.g. gender, race etc.

Next: a very simple and interpretable class of regression methods, called linear models. You might have heard of those before...

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.