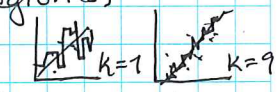
Computational Statistics. Lecture 7. KNN and Cross-Validation

15 March 2018

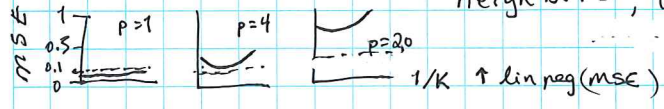
- K-nearest neighbor regression (KNN).

$$\hat{f}(x_0) = \Sigma_{x \in N_0} y_i$$ whose N_0 is the set of K training observations whose x values are closest to x_0.
↳ want training MSE = 0.

- By LSQ, $\hat{y} = Py$, whose weights depend on the distance of points from center of data
- $K \uparrow \Rightarrow$ more smoothing, ie. piecewise constant over (a) smaller region(s)
  KNN works best if $1/K$ is small, ie. K is large. eg. Fig 3.16 (ISLR)



- "Curse of dimensionality": in high dimensional (p) model, pts don't have any close neighbors, unless there is a lot of data; the MSE ↑ and p ↑.



(Fig 3.20) "Test MSEs for linear regression and KNN as # of variables p ↑. The true function is non-linear in the first variable and does not depend on additional variables. The performance of linear regression deteriorates slowly in the presence of these additional noise variables, whereas KNN's performance degrades much more quickly as p increases."

- Generalization: Loess smoother (kernel-weigh central pts more heavily than distant ones)
  - α controls smoothing; small $\alpha \Rightarrow$ less smoothing
  - If α is too high, then there is a systematic bias in error (the various loess curves are smooth and have a pattern).
  - Curse of dimensionality also affects loess.

(See R code lines 18 ... for example, use norm(0 fn)

- Cross Validation We want $\mathbb{E}_{\text{training data}, x_{new}, y_{new}} (Y_{new} - \hat{f}(x_{new}))$; $\hat{f}$ depends on training data

We are interested in the expected test MSE:

$$\mathbb{E}(\text{test MSE}) = \mathbb{E}\left[(Y_{new} - \hat{f}(x_{new}))^2\right] = \mathbb{E}_{\text{training data}}\left[\mathbb{E}_{x_{new}, y_{new}}\left[(Y_{new} - \hat{f}(x_{new}))^2 \mid \text{training data}\right]\right]$$

- Why do we care about these quantities? → We can use this...
  ① to estimate the test error: model assessment. and
  ② to select tuning parameters: model selection.

  * We cannot use training MSE.

If there is a large test set $(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_m, \tilde{y}_m)$, we compute

$$\frac{1}{m}\sum_{i=1}^{m} (\tilde{y}_i - \hat{f}(\tilde{x}_i))^2$$. $\hat{f}$ trained on training set.

- How do we estimate the expected test MSE?
  - use training MSE? NO
  - use large unseen test set? Okay but often not available.
  * . hold out some observations as test set.

(Estimating expected test MSE...)

(A) Validation set approach: Split the data randomly into two halves. Use one to train $\hat{f}$, and the other for testing, from which we evaluate performance of $\hat{f}$.

Pro: - "fair" estimate of test MSE.

Con: - too pessimistic (we trained on only half of the data) (ie biased upward (MSE))
(estimates MSE is higher than what it should be)
- varies a lot, depending on the split.

↳ Solutions...

(B) ("n-fold") Leave One Out Cross Validation (LOOCV) (Fig 5.3)

- For $i = 1, \ldots, n$, train $\hat{f}_i$ on (x_j, y_j), $j \in \{1, \ldots, n\} \setminus \{i\}$
- Evaluate $\hat{f}$ at $x_i \equiv \hat{f}^{(-i)}(x_i)$ so

$$MSE_i = \left(y_i - \hat{f}^{(-i)}(x_i)\right)^2$$

(LOOCV ≡ n-fold Xvalid'n)

- LOOCV estimate of expected test MSE is $\frac{1}{n}\sum_{i=1}^{n} MSE_i$

Pros: less biased as $\hat{f}$ is trained on a larger (size $n-1$) data set
Cons: computationally intensive, especially if n is large, though there are shortcuts for some types, eg. LSQ.

(C) K-fold CV

$K = 5$ to 10 fold usually used (namely $k = 10$) (and is comp. easier)
Lie split data set into k non-overlapping groups, & group remove that from training set, and use the group for test data to get

- Find k (flexibility) with min MSE, which is a combination of the bias + the variance @ that k.