SML — LEC 3 — CROSS-VALIDATION & MODEL SELECTION — SEPT. 2, 2014

LECTURER: PROF. HOMRIGHAUSEN — SCRIBE: ZACH WELLER

1 Cross Validation (CV)

Recall, we are attempting to get at an estimator of risk, $R(f) = \text{Bias} + \text{Variance}$.

1.1 Leave-one-out CV

Intuitive idea: set aside one observation and predict it,

Example 1.1. *Set aside (X_1, Y_1), estimate $\hat{f}^{(1)}$ with the remaining $n-1$ observations, and predict Y_1. We can then compute $R_1(\hat{f}^{(1)}) = [Y_1 - \hat{f}^{(1)}(X_1)]^2$. Note: $\hat{f}^{(1)}$ symbolizes leaving out the first observation before fitting $\hat{f}$*

Since the left off data point is <u>independent</u> of the data points used to fit f,

$$\mathbb{E}_{(X_1,Y_1)|\mathcal{D}_{(1)}} R_1(\hat{f}^{(1)}) \overset{D}{=} R(\hat{f}(\mathcal{D}_{n-1})) \approx R(\hat{f}(\mathcal{D})) \tag{1}$$

Iterating over all observations and taking the average produces **<u>leave-one-out CV</u>**:

$$CV_n(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} R_i(\hat{f}^{(i)}) = \frac{1}{n}\sum_{i=1}^{n}[Y_i - \hat{f}^{(i)}(X_i)]^2 \tag{2}$$

1.2 More general CV schemes

Let $\mathcal{N} = \{1, \ldots, n\}$ be the index set for the data, $\mathcal{D}$. Define a distribution $\mathcal{V}$ over $\mathcal{N}$ with (random) variable v. Then, we can perform a general <u>cross-validation</u> estimator as

$$\text{CV}_{\mathcal{V}}(\hat{f}) = \mathbb{E}_{\mathcal{V}}\hat{\mathbb{P}}_v \ell_{\hat{f}(v)} \tag{3}$$

Examples:

- K-fold: fix $V = \{v_1, , v_k\}$ such that $\mathbf{v}_k \cap \mathbf{v}_j = \emptyset$ and $\cup_j v_j = \mathcal{N}$. Then **<u>K-fold CV estimator</u>** is given as:

$$CV_K(\hat{f}) = \frac{1}{K}\sum_{v\in V}\frac{1}{|v|}\sum_{i\in v}(Y_i - \hat{f}^{(v)}(X_i))^2 \tag{4}$$

- Bootstrap: let $\mathcal{V}$ be given by the bootstrap distribution over $\mathcal{N}$ (i.e. sample with replacement many times).
- Factorial: let $\mathcal{V}$ be given by all subsets (or a subset of all subsets) of $\mathcal{N}$ (putting mass of $1/(2^n - 2)$ on each subset)

1.3 Comparison of CV schemes

- CV_k gets more computationally demanding as $K \to n$.
- The bias of CV_k goes down, but the variance increases as $K \to n$.
- Factorial version isn't commonly used (except when doing a 'real' data example for paper)
- Many other flavors of CV (e.g. "consistent cross validation": see homework)

2 Brief Summary

2.1 Risk Estimation Methods: chosen based on the application

- CV
- AIC
- BIC

2.2 General Recipe

1. Select a model suited to your task
2. Chose a risk estimation method that has desirable properties
3. Perform necessary computations to minimize (2), constrained by the family of procedures in (1).
4. Show theoretically that your procedure has desirable properties.

3 Brief Optimization & Convexity Detour

3.1 Optimization

An optimization problem can generally be formulated as

$$\begin{aligned} \text{minimize } & F(x) && (5)\\ \text{subject to } & f_j(x) \leq 0 \text{ for } j = 1, \ldots, m && (6)\\ & h_k(x) = 0 \text{ for } k = 1, \ldots, q && (7) \end{aligned}$$

where

$x = (x_1, \ldots, x_n)^\top$ are the parameters

$F : \mathbb{R}^n \to \mathbb{R}$ is the objective function

$f_j, h_k : \mathbb{R}^n \to \mathbb{R}$ are constraint functions

3.2 Convexity

The main dichotomy of optimization problems is convex vs. nonconvex.

Intuition: Convexity means that the function values at a point x' are above the supporting hyperplane given by the tangent space at any point x. See Figure 1 for an example.

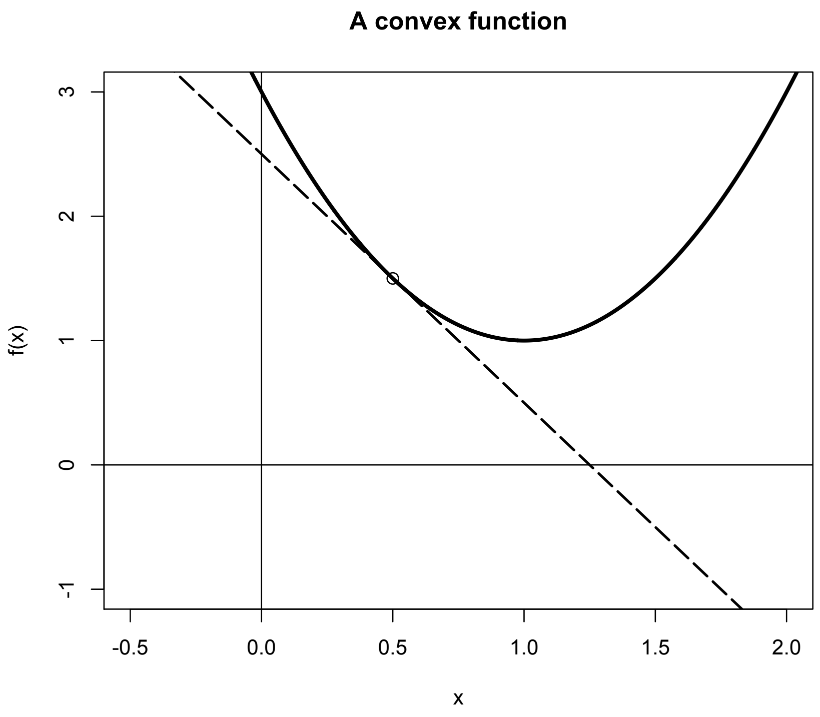


Figure 1: Convex function, $f(x) = 2(x-1)^2 + 1$, with tangent line at $x = 0.50$.

Methods for convex optimization programs are (roughly) always global and fast.

For general nonconvex problems, we have to give up one of these:

- Local optimization methods that are fast, but need not find global solution
- Global optimization methods that find global solutions, but are not always fast (indeed, are often slow)

4 Model Selection

If there are p predictors then there are $2^p - 1$ possible models ,without considering interactions or transformations. In general, this is a nonconvex problem. We must sift through the models in a computationally feasible way.

4.1 All Subsets: specific case of Branch and Bound

This can efficiently be computed via the leaps package in R, using either the leaps or regsubsets functions.

The statistical implementation is based on the paper Furnival and Wilson (1974)

It is by far the most widely used tool for solving large scale NP-hard combinatorial optimization problems.

Note, however, that though it can speed up the optimization immensely, it cannot reduce the complexity of the problem (still exponential)

4.2 Branch and Bound

Let $M = \{M_1, \ldots, M_K\}$ be the set of all possible solutions and a partition comprised of branches, respectively.

Statistically, we think of M as the set of all possible models.

Let F be the objective function and $m_* = \max_{m \in M} F(m)$.

For each M_k, define
$$m_k = \max_{m \in M_k} F(m)$$
and let $\underline{m}_k, \overline{m}_k$ be a bracket such that
$$\underline{m}_k \leq m_k \leq \overline{m}_k$$
(Note that m_k is in general not explicitly constructed)

Then
$$\max_k \underline{m}_k = \underline{m} \leq m_* \leq \overline{m} = \max_k \overline{m}_k$$

The main realization is that the branch M_k does not need to be explored if either of the following occur

i. Bound
$$\overline{m}_k \leq \underline{m}$$

ii. Optimality
$$\max_{m \in M_k} F(m) \text{ has been found}$$

The two main questions remain:

1. How to choose the partition(s)?
2. How to form the upper/lower bounds?

These are very case specific. Let's return to model selection

4.2.1 Branch and Bound for Model Selection

Let's suppose we set[1]

$$F(m) = n\log(\hat{R}_{\text{train}}(\hat{\beta}_m)) + 2|m|$$

For the M_k, let

$m_{k,inf}$ be the largest model contained[2] in every model in M_k

$m_{k,sup}$ be a smallest model that contains every model in M_k

Example 4.1. *Let $x_1, \ldots, x_5$ be covariates*

$$M = \cup_{k=1}^{3} M_k,$$

where

$$\begin{aligned} M_1 &= \{\{x_1, x_3\}, \{x_2\}\}, \\ M_2 &= \{\{x_2, x_3, x_4\}, \{x_3, x_4\}\}, \\ M_3 &= \{\{x_3, x_5\}, \{x_3\}\}, \end{aligned}$$

Then,

$m_{2,inf} = \{x_3, x_4\}$

$m_{2,sup} = \{x_2, x_3, x_4\}$

From the above definitions, we have: $\forall m \in M_k$

$$F(m) \geq n\log(\hat{R}_{\text{train}}(\hat{\beta}_{m_{k,\text{sup}}})) + 2|m_{k,\text{inf}}| = L_k$$

$$F(m) \leq n\log(\hat{R}_{\text{train}}(\hat{\beta}_{m_{k,\text{inf}}})) + 2|m_{k,\text{sup}}| = U_k$$

4.2.2 Branch and Bound for Model Selection: Algorithm

1. Define a global variable $b = F(m)$ for any $m \in M$
   As an aside, every time $F(m)$ is computed, update b if $F(m) < b$
2. Partition $M = \{M_1, \ldots, M_K\}$
3. For each k, if $L_k > b$, eliminate the branch M_k
4. Else, recurse and return to 2., substituting M_k for M

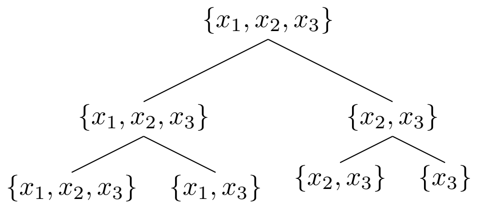
Zach's notes:

- Hand (1981) provides a very readable paper describing the algorithm with a nice general example and examples of how the algorithm is applied to specific statistical problems (e.g. selecting variables in a regression analysis).

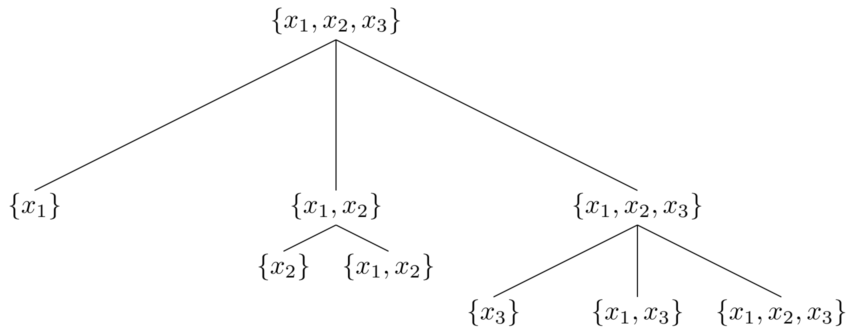
[1] Note: we are trying to minimize F, not maximize

[2] This does not have to be in M_k

- Question: what is the most efficient way to "grow" the tree? (i.e. how to partition M?) In their 1974 paper titled "Regression by Leaps and Bounds" (hence the names of the R function), Furnival and Wilson propose 4 different algorithms for partitioning and traversing the tree.
- For just partitioning the tree, Furnival and Wilson illustrate two strategies:
    1. Binary trees: each parent is split into two notes. For example:



    2. Add-one-at-a-time: (or nesting idea)



- Note: the partitioning of the above trees is incomplete.
- In general, I have not come across an "optimal" method for partitioning trees.
- I've found some literature on the most efficient way to select exactly g features from a set of p features ($g < p$). (e.g. see Narendra and Fukunaga, 1977)

4.3 Stepwise

4.3.1 Forward Stepwise Selection

In the likely event that $|M|$ is too large to be searched over exhaustively, a common greedy approximation is the following

1. Find $b = F(\emptyset)$, where $\emptyset$ is the empty set
2. Search over all p singleton sets and record $m_{1,\min} = F(m)$. If $F(m_{1,\min}) < b$ set $b \leftarrow F(m_{1,\min})$, else return $\emptyset$
3. Now search over all $p - 1$ models that contain $m_{1,\min}$ and form $m_{2,\min} = F(m_{1,\min} \cup \{x_j\})$. If $F(m_{2,\min}) < b$ set $b \leftarrow F(m_{2,\min})$, else return $m_{1,\min}$
4. $\cdots$

4.3.2 General Stepwise Selection

This algorithm can can adapted to..

- start with the full model and stepwise remove covariates
  useful if the full model isn't too large and a superset of the important covariates is desired
- consider both adding and removing covariates at each step

4.4 Regularization: see next set of notes