ML: Machine Learning **Spring 2014**

Lecture 28- Decision trees

Lecturer: Lorenzo Rosasco

In this section we consider learning algorithms based on local estimates on a partition of the input space. First, we assume the partition to be given. Then, we discuss how the partition can be constructed in a data driven way.

28.1 Partition based estimate

We first consider the case when a partition of the input space is given. If $X = \mathbb{R}^d$ is the input space a partition is a family of disjoint sets (cells) whose union is the whole input space, that is

$$\mathcal{A} = \{A_j \subseteq X, j = 1, \dots, J \mid X = \cup_{j=1}^J A_j, \quad A_j \cap A_i = \emptyset, i \neq j\}.$$

A basic example is a dyadic partition, where each cell A_j is a cube of sidelength 2^{-k} for some integer k.

Given a partition $\mathcal{A}$, the simplest partition based estimator learns a constant function approximation in each cell A_j of the partition. If we denote by $\widehat{c}_1, \dots, \widehat{c}_J$ the constant function values on $A_1, \dots, A_J$, a natural way to estimate them is to consider the problem

$$\min_{c_1,\dots,c_J \in \mathbb{R}} \sum_{j=1}^J \sum_{x_i \in A_j} (y_i - c_j)^2.$$

Each value $\widehat{c}_j$, $j = 1, \dots, J$ can be computed separately, considering for $j = 1, \dots, J$

$$\min_{c_j \in \mathbb{R}} \sum_{x_i \in A_j} (y_i - c_j)^2.$$

Then, a direct computation shows that for $j = 1, \dots, J$

$$\widehat{c}_j = \frac{1}{n_j} \sum_{x_i \in A_j} y_i,$$

where n_j is the number of input points in the cell A_j.

We add a few comments. First, it is easy to see that we can equivalently derive the above estimator introducing the class of piecewise constant functions

$$\mathcal{H} = \{f : X \to \mathbb{R} \mid f(x) = \sum_{j=1}^J c_j \mathbb{1}_{A_j}(x), \quad \forall x \in X, c_1, \dots, c_J \in \mathbb{R}\}$$

and considering the problem

$$\min_{f \in \mathcal{H}} \sum_{1=1}^n (y_i - f(x_i))^2.$$

As a byproduct, we see that the above approach is just a particular instance of ERM. Second, we can easily extend the above idea beyond piecewise constant functions. For example, we could consider piecewise linear functions of the form,

$$\mathcal{H} = \{f : X \to \mathbb{R} \mid f(x) = \sum_{j=1}^{J} (w_j^\top x + b_j) \mathbb{1}_{A_j}(x), \quad \forall x \in X, w_j, \in \mathbb{R}^d, b_j \in \mathbb{R}, j = 1, \dots, J\}.$$

Also in this case, it is easy to see that the computation of the estimator reduces again to a separate problem in each cell of the partition. In this case, the solution requires solving a least squares problem in each cell. Given the above discussion, it should then be clear that the idea can be extended considering

$$\mathcal{H} = \{f : X \to \mathbb{R} \mid f(x) = \sum_{j=1}^{J} g(x) \mathbb{1}_{A_j}(x), \quad \forall x \in X, g \in \mathcal{G}\}.$$

where $\mathcal{G}$ can be a space of functions defined by features or kernels, neural networks etc. The solution requires again solving a separate ERM problem in each cell.

Next we discuss how the partition can be computer in a data adaptive way.

28.2 Partition trees based estimators

Next we discuss how to select or construct a partition in a data driven way. The first approach is based on considering a partition tree, which is a family of partitions of increasing cardinality. We let $\mathcal{A}_0, \mathcal{A}_1, \dots \mathcal{A}_Q$ be such a family, so that each $\mathcal{A}_q$ is a partition. The intuition is that each partition is a refinement of another partition. Indeed we assume that $\mathcal{A}_0 = \{X\}$ and moreover for each $A \in \mathcal{A}_q$ there exists $C_1, \dots C_M \subset \mathcal{A}_{q+1}$ such that $A = \cup_{j=1}^{M} C_j$. Note that for the cardinality of each partition $\mathcal{A}_q$ is $J_q \leq M^q$. The case where $M = 2$ corresponds to dyadic partition and defines a dyadic partition tree.

Given a partition tree the idea is to select or construct a partition on which to perform the local partition estimate. A first approach is to view the *depth* parameter q as a tuning parameter, consider the estimator corresponding to each partition $\mathcal{A}_q$, and the select the best partition, and corresponding estimator, choosing q by cross validation. Such an approach is called uniform partitioning.

Still assuming a partition tree is given, a second approach is based on recursively building a partition by refining a cell if the error in such a cell is larger than a given threshold τ. For $A \in \mathcal{A}_q$ let

$$\widehat{L}_A = \min_{f \in \mathcal{H}} \frac{1}{n_A} \sum_{x_i \in A} (y_i - f_A(x_i))^2,$$

where n_A is the number of input points in A. The idea is that A is kept in the partition if $\widehat{L}_A \leq \tau$, otherwise it is split considering $C_1, \dots C_M \subset \mathcal{A}_{q+1}$ introduced above. The resulting partition contains cells of different size. The threshold τ becomes the regularization parameter. This latter approach is called adaptive partitioning.

Finally, we discuss greedy approaches to build a (dyadic) partition tree recursively based on data. Begin considering a parameterized family of partitions $\mathcal{A}(s,j)$, which is made of two sets

$$A_1(j,s) = \{x \in X \mid x^j \le s\}, \quad A_2(j,s) = \{x \in X \mid x^j > s\}$$

where x^j denoted the j-h component of x. Then, select one among such partitions solving the problem

$$\min_{j,s} \left[\widehat{L}_{A_1(s,j)} + \widehat{L}_{A_2(s,j)} \right].$$

Note that the minimization over s reduces to a search over finitely many thresholds since we have exactly n values for each coordinate x^j. Then if a pair j_*, s_* solves the above problem, we can then set $\mathcal{A}_1 = \mathcal{A}(s_*, j_*)$. Then, the algorithm proceeds recursively applying the above scheme to each cell $A_{1,1}$, $A_{1,2}$ in $\mathcal{A}_1$ by considering

$$B_{2,1}(j,s) = \{x \in A_{1,1} \mid x^j \le s\}, \quad B_{2,2}(j,s) = \{x \in A_{1,1} \mid x^j > s\},$$

and

$$B_{2,3}(j,s) = \{x \in A_{1,2} \mid x^j \le s\}, \quad B_{2,4}(j,s) = \{x \in A_{1,2} \mid x^j > s\}.$$

Following the same approach as above to select the best threshold and coordinate direction we can then define a partion $\mathcal{A}_2$. Other partitions in the tree can be built in an analogous way. We end noting that the least squares error to select the best partition can be replace by other error measures. Common error measures in classification include the cross entropy and the so called Gini index.