Ensimag – Grenoble INP – UGA — Year 2024-2025
Introduction to Statistical Learning and Applications
Pedro L. C. Rodrigues — pedro.rodrigues@inria.fr
Alexandre Wendling — alexandre.wendling@univ-grenoble-alpes.fr

TD 5: Gradient boosting

▶ Exercise 1: Decision trees

Suppose we have a training dataset with N samples $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i = (x_{i1}, \dots, x_{ip}) \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$.

We would like to learn a regression model on this dataset using a decision stump, i.e. a decision tree with just one level that splits the data into just two regions as per:

$$f(\mathbf{x}) = c_1 \mathbf{I}(\mathbf{x} \in \mathcal{R}_1) + c_2 \mathbf{I}(\mathbf{x} \in \mathcal{R}_2)$$

Note that regions $\mathcal{R}_1$ and $\mathcal{R}_2$ are defined based on a choice of splitting value $s \in \mathbb{R}$ and predictor $j \in \{1, \dots, p\}$,

$$\mathcal{R}_1 = \{\mathbf{x}_i \mid x_{ij} \leq s\} \quad \text{and} \quad \mathcal{R}_2 = \{\mathbf{x}_i \mid x_{ij} > s\} \ .$$

Our criterion for choosing c_1, c_2, s, j is based on the minimization of the variance in each of the regions, i.e.

$$\min_{j,s,c_1,c_2} \left(\sum_{\mathbf{x}_i \in \mathcal{R}_1} (y_i - c_1)^2 + \sum_{\mathbf{x}_i \in \mathcal{R}_2} (y_i - c_2)^2 \right)$$

(a) Show that for a fixed choice of j and s, we can minimize the above problem with

$$c_1 = \frac{1}{|\mathcal{R}_1|} \sum_{\mathbf{x}_i \in \mathcal{R}_1} y_i \quad \text{and} \quad c_2 = \frac{1}{|\mathcal{R}_2|} \sum_{\mathbf{x}_i \in \mathcal{R}_2} y_i$$

First note that the optimization problem can be split into two independent ones as per

$$\min_{c_1,c_2} \left(\sum_{\mathbf{x}_i \in \mathcal{R}_1} (y_i - c_1)^2 + \sum_{\mathbf{x}_i \in \mathcal{R}_2} (y_i - c_2)^2 \right) = \min_{c_1} \sum_{\mathbf{x}_i \in \mathcal{R}_1} (y_i - c_1)^2 + \min_{c_2} \sum_{\mathbf{x}_i \in \mathcal{R}_2} (y_i - c_2)^2$$

And if we now derive each objective function with respect to c_j, we see that

$$\frac{\partial}{\partial c_j} \left(\sum_{\mathbf{x}_i \in \mathcal{R}_1} (y_i - c_j)^2 \right) = 0 \iff c_j = \frac{1}{|\mathcal{R}_j|} \sum_{\mathbf{x}_i \in \mathcal{R}_j} y_i$$

For the following items, choose the only sentence which is true about decision trees:

(b) A given split node in a decision tree classifier makes:

- a binary decision considering a single feature at a time
- a binary decision considering a combination of all the input features
- multiple binary decisions considering a single feature
- a binary decision considering a non-linear combination of all input features

(c) A decision tree split is built:

- using a random threshold
- using the median value of a single feature as a threshold
- using a threshold that minimizes an error

(d) Decision tree regressors can predict:

- any values, including values larger or smaller than the y_i observed in the training dataset.
- only values in the range from $\min(y_i)$ to $\max(y_i)$.

▶ Exercise 2

Recall from class that in the first round of gradient boosting we look for a predictor f_1 such that

$$f_1 = \underset{h \in \mathbb{H}}{\text{argmin}} \ \langle \nabla \mathcal{L}(f_0, \mathcal{D}), h \rangle$$

where for a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ we have that

$$\mathcal{L}(f, \mathcal{D}) = \sum_{i=1}^{N} \ell\Big(f(\mathbf{x}_i), y_i\Big)$$

where ℓ is a function used to measure the error of an estimator, e.g. MSE in regression or classification error.

In what follows, we will assume that

$$\forall h \in \mathbb{H}, \sum_i h(\mathbf{x}_i)^2 = 1 \ ,$$

which is easy to ensure by making the weak learners normalized on the dataset.

(a) Show that

$$f_1 = \underset{h \in \mathbb{H}}{\text{argmin}} \sum_{i=1}^{N} \Big(h(\mathbf{x}_i) - r_i^{(1)}\Big)^2 \quad \text{where} \quad r_i^{(1)} = -\frac{\partial \ell(f_0(\mathbf{x}_i), y_i)}{\partial f_0(\mathbf{x}_i)}$$

Interpret this result.

We have that

$$\begin{aligned}
f_1 &= \underset{h \in \mathbb{H}}{\text{argmin}} \ \langle \nabla \mathcal{L}(f_0, \mathcal{D}), h \rangle \\
&= \underset{h \in \mathbb{H}}{\text{argmin}} \sum_{i=1}^{N} t_i^{(1)} h(\mathbf{x}_i) \quad \text{where} \quad t_i^{(1)} = \frac{\partial \ell(f_0(\mathbf{x}_i), y_i)}{\partial f_0(\mathbf{x}_i)} \\
&= \underset{h \in \mathbb{H}}{\text{argmin}} \ -2 \sum_{i=1}^{N} r_i^{(1)} h(\mathbf{x}_i) \quad \text{where} \quad r_i^{(1)} = -\frac{\partial \ell(f_0(\mathbf{x}_i), y_i)}{\partial f_0(\mathbf{x}_i)} \\
&= \underset{h \in \mathbb{H}}{\text{argmin}} \underbrace{\sum_{i=1}^{N} (r^{(1)})^2}_{\text{cst wrt } h} - 2 \sum_{i=1}^{N} r_i^{(1)} h(\mathbf{x}_i) + \underbrace{\sum_{i=1}^{N} h(\mathbf{x}_i)^2}_{\text{cst by hyp}} \\
&= \underset{h \in \mathbb{H}}{\text{argmin}} \sum_{i=1}^{N} \Big(h(\mathbf{x}_i) - r_i^{(1)}\Big)^2
\end{aligned}$$

So we note that the weak learner is trained via MSE regression targeting the values of the gradients of the loss function. It is worth really understanding that this is the case for both regression and classification! Meaning that even if we were doing classification (with an appropriate choice of ℓ) the gradient boosting would still be doing a regression step over the gradients to learn the weak learners.

(b) Consider that we are in a regression setting for which

$$\ell(f(\mathbf{x}), y) = \frac{1}{2}(f(\mathbf{x}) - y)^2 \ .$$

What would be the expression for the optimization problem from item (a) in this case?

We get $r_i^{(1)} = y_i - f_0(\mathbf{x}_i)$ which is simply the residual of the weak learner f_0.

(c) The most common version of gradient boosting is to assume that the weak learners are regression trees, such as the decision stumps from Exercise 1. Describe how the parameters c_1, c_2, s, j are estimated in this case for a general choice of ℓ.

We should just plug in the expression of the decision stump into the regression loss and see what we get

$$\begin{aligned} f_1 &= \underset{h\in\mathbb{H}}{\operatorname{argmin}} \sum_{i=1}^N \left(h(\mathbf{x}_i) - r_i^{(1)}\right)^2 \\ &= \underset{c_1,c_2,s,j}{\operatorname{argmin}} \sum_{i=1}^N \left([c_1\mathbf{I}(\mathbf{x}_i \in \mathcal{R}_1) + c_2\mathbf{I}(\mathbf{x}_i \in \mathcal{R}_2)] - r_i^{(1)}\right)^2 \\ &= \underset{c_1,c_2,s,j}{\operatorname{argmin}} \sum_{\mathbf{x}_i\in\mathcal{R}_1} \left(c_1 - r_i^{(1)}\right)^2 + \sum_{\mathbf{x}_i\in\mathcal{R}_2} \left(c_2 - r_i^{(1)}\right)^2 \end{aligned}$$

Note that we have fallen back to the same kind of estimation problem from Exercise 1.

▶ Exercise 3

In this exercise, we will see how the AdaBoost algorithm can be obtained using the gradient boosting framework. We will consider $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, +1\}$ and the goal is to fit a set of weak learners $f_0, f_1, \ldots, f_T$ such that the classification of a data point $\mathbf{x}$ can be done as per

$$c(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=0}^T \alpha_t f_t(\mathbf{x})\right)$$

(a) Since we're in a classification setting, we need to choose suitable loss function ℓ. At first glimpse, we could be tempted to choose the 0-1 loss:

$$\ell_{0-1}(f(\mathbf{x}), y) = \mathbf{1}(f(\mathbf{x}) \neq y) \ .$$

Explain why this would not be a good choice for gradient boosting?

In gradient boosting we need to calculate the gradients of the loss function. The 0-1 loss is both non-differentiable at the origin and a constant everywhere else. This means that the gradient boosting would have quite a lot of difficulties to work.

(b) In AdaBoost we actually consider the exponential loss given as

$$\ell_{\exp}(f(\mathbf{x}), y) = e^{-yf(\mathbf{x})}$$

Compare ℓ_{0-1} and $\ell_{\exp}$ and explain why using $\ell_{\exp}$ should be fine.

The idea here is to first notice that we can write $\ell_{0-1}(f(\mathbf{x}), y) = \mathbf{1}(f(\mathbf{x}) \neq y) = \mathbf{1}(f(\mathbf{x})y < -1)$. The easiest way to compare the two loss functions is to define a new variable called margin $s = yf(x)$ and show how they compare in a 2D plot where X-axis is for s and Y-axis is for the values of $\ell(s)$. One should note that $\ell_{\exp}$ is differentiable everywhere and that it is an upper-bound for ℓ_{0-1}. As such, if we manage to find a function f that minimizes the latter, then we should be in a good place for minimizing the former as well.

(c) Using the notation from class, calculate the expression for the $t_i^{(1)}$ using $\ell_{\exp}$ as loss.

We have that $t_i^{(1)} = -y_i \exp\Big(-y_i f_0(\mathbf{x}_i)\Big)$

(d) Give an interpretation in terms of a weighted classification error to the optimization problem

$$f_1 = \underset{h\in\mathbb{H}}{\operatorname{argmin}} \sum_{i=1}^N t_i^{(1)} h(\mathbf{x}_i)$$

Plugging in the expression for $t_i^{(1)}$ we get:

$$\begin{aligned} f_1 &= \underset{h\in\mathbb{H}}{\operatorname{argmin}} \ -\sum_{i=1}^N y_i \exp\Big(-y_i f_0(\mathbf{x}_i)\Big) h(\mathbf{x}_i) \\ &= \underset{h\in\mathbb{H}}{\operatorname{argmin}} \ -\sum_{i=1}^N y_i h(\mathbf{x}_i) \frac{\exp\Big(-y_i f_0(\mathbf{x}_i)\Big)}{Z} \end{aligned}$$

where $Z = \sum_{j=1}^{N} \exp\Big(-y_j f_0(\mathbf{x}_j)\Big)$ is a normalization constant.

We can pursue things and write

$$f_1 \quad = \quad \operatorname*{argmin}_{h \in \mathbb{H}} \; -\sum_{i=1}^{N} y_i h(\mathbf{x}_i) w_i \quad \text{with} \quad w_i = \frac{\exp\Big(-y_i f_0(\mathbf{x}_i)\Big)}{Z}$$

Note that $y_i h(\mathbf{x}_i) \in \{-1, +1\}$ with $y_i h(\mathbf{x}_i) = 1 \iff h(\mathbf{x}_i) = y_i$ (i.e. correct classification).

Therefore, we can rewrite things to get

$$f_1 \quad = \quad \operatorname*{argmin}_{h \in \mathbb{H}} \sum_{y_i \neq h(\mathbf{x}_i)} w_i - \sum_{y_i = h(\mathbf{x}_i)} w_i$$

but since w_i sums to one over the N datapoints, we can rewrite

$$\sum_{y_i = h(\mathbf{x}_i)} w_i = 1 - \sum_{y_i \neq h(\mathbf{x}_i)} w_i$$

and obtain

$$\begin{aligned} f_1 &= \operatorname*{argmin}_{h \in \mathbb{H}} 2 \sum_{y_i \neq h(\mathbf{x}_i)} w_i - 1 \\ &= \operatorname*{argmin}_{h \in \mathbb{H}} \sum_{y_i \neq h(\mathbf{x}_i)} w_i \\ &= \operatorname*{argmin}_{h \in \mathbb{H}} \sum_{i=1}^{N} \mathbf{1}\big(y_i \neq h(\mathbf{x}_i)\big) w_i \end{aligned}$$

This means that f_1 is trained so to minimize weighted errors on a dataset for which the data points are weighted with w_i. For a given pair $(\mathbf{x}_i, y_i)$ the weights acts so that, using the combined classifier from the previous round:

- If we have a misclassification, i.e $f_0(\mathbf{x}_i) \neq y_i$, then $w(\mathbf{x}_i, y_i) \times Z \geq 1$
- If we have a correct classification, i.e. $f_0(\mathbf{x}_i) = y_i$, then $w(\mathbf{x}_i, y_i) \times Z \leq 1$

As such, we can say that the weights w are giving more weight to points that were misclassified in the previous rounds and less weight to those that were correcty classified. In this way, the new weak learner f_1 is trained so to better classify the points that were incorrectly labeled previously.