**Machine Learning — Statistical Methods for Machine Learning**

# Hyperparameter tuning and risk estimates

Instructor: *Nicolò Cesa-Bianchi* version of March 21, 2023

Learning algorithms often have hyperparameters. These are special parameters (like $k$ in $k$-NN or the learning rate in neural networks) whose value must be determined before the training phase can start. Crucially, setting the hyperparameters in the wrong way can lead to underfitting or overfitting.

A learning algorithm with one or more hyperparameters is not really an algorithm, but rather a family of algorithms, one for each possible assignment of values to the hyperparameters. Let $\{A_\theta : \theta \in \Theta\}$ be such a family of learning algorithms, where $\Theta$ is the set of all possible hyperparameter values. Fix a learning problem $(\mathcal{D}, \ell)$ and let $A_\theta(S)$ be the predictor output when $A_\theta$ is run on the training set $S$. Let $\ell_{\mathcal{D}}\big(A_\theta(S)\big)$ be the risk of the predictor $A_\theta(S)$, and let $\mathbb{E}\big[\ell_{\mathcal{D}}(A_\theta)\big]$ be the expected risk of $A_\theta(S)$ where the expectation is with respect to the random draw of the training set $S$ of a given fixed size. Intuitively, $\mathbb{E}\big[\ell_{\mathcal{D}}(A_\theta)\big]$ measures the performance of $A_\theta$ on a typical training set of that size.

**Evaluating a learning algorithm using external cross-validation.** Assume for now the hyperparameter $\theta$ is fixed and focus on the problem of estimating $\mathbb{E}\big[\ell_{\mathcal{D}}(A)\big]$. To do so we can use a technique called $K$-fold (external) cross-validation.

Let $S$ be our entire dataset. We partition $S$ in $K$ subsets (also known as *folds*) $S_1, \dots, S_K$ of size $m/K$ each (assume for simplicity that $K$ divides $m$). The extreme case $K = m$ provides an estimate known as *leave-one-out*. Now let $S_{-i} \equiv S \setminus S_i$. We call $S_i$ the **testing part** of the $i$-th fold while $S_{-i}$ is the **training part**.

For example, if we partition $S = \big\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_{20}, y_{20})\big\}$ in $K = 4$ subsets

$$\begin{array}{ll} S_1 = \big\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_5, y_5)\big\} & S_2 = \big\{(\boldsymbol{x}_6, y_6), \dots, (\boldsymbol{x}_{10}, y_{10})\big\} \\ S_3 = \big\{(\boldsymbol{x}_{11}, y_{11}), \dots, (\boldsymbol{x}_{15}, y_{15})\big\} & S_4 = \big\{(\boldsymbol{x}_{16}, y_{16}), \dots, (\boldsymbol{x}_{20}, y_{20})\big\} \end{array}$$

then $S_{-2} = S_1 \cup S_3 \cup S_4$.

The *$K$-fold CV estimate* of $\mathbb{E}\big[\ell_{\mathcal{D}}(A)\big]$ on $S$, denoted by $\ell_S^{\text{cv}}(A)$, is then computed as follows: we run $A$ on each training part $S_{-i}$ of the folds $i = 1, \dots, K$ and obtain the predictors $h_1 = A\big(S_{-i}\big), \dots, h_K = A\big(S_{-K}\big)$. We then compute the (rescaled) errors on the testing part of each fold,

$$\ell_{S_i}(h_i) = \frac{K}{m} \sum_{(\boldsymbol{x},y) \in S_i} \ell\big(y, h_i(\boldsymbol{x})\big)$$

Finally, we compute the CV estimate by averaging these errors

$$\ell_S^{\text{cv}}(A) = \frac{1}{K} \sum_{i=1}^{K} \ell_{S_i}\big(h_i\big)$$

**Tuning hyperparameters on a given training set.** In practice, we face the problem of choosing the hyperparameters so to obtain a predictor with small risk. This is typically done by minimizing a risk estimate computed using the training data. As $\Theta$ may be very large, possibly infinite, the minimization is generally not over $\Theta$, but over a suitably chosen subset $\Theta_0 \subset \Theta$ (for example, if $\Theta = [0,1]$, then $\Theta_0$ could by a finite grid of equally spaced values in $[0,1]$). If $S$ is our training set, then we want to find $\theta^* \in \Theta_0$ such that

$$\ell_{\mathcal{D}}\big(A_{\theta^*}(S)\big) = \min_{\theta \in \Theta_0} \ell_{\mathcal{D}}\big(A_\theta(S)\big) \tag{1}$$

The estimate is computed by splitting the training data in two subsets $S_{\text{train}}$ and $S_{\text{dev}}$. The development set $S_{\text{dev}}$ (also called validation set) is used as a surrogate test set. The algorithm is run on $S_{\text{train}}$ once for each value of the hyperparameter in $\Theta_0$. The resulting predictors are tested on the dev set. In order to obtain the final predictor, the learning algorithm is run once more on the original training set $S$ using the value of the hyperparameter corresponding to the predictor with smallest error on the validation set.

**Tuning parameters via nested cross-validation.** What if we want to estimate the expected value of (1) with respect to the random draw of the training set of fixed size?

$$\mathbb{E}\Big[\min_{\theta \in \Theta_0} \ell_{\mathcal{D}}\big(A_\theta\big)\Big] \tag{2}$$

In other words, we want to estimate the performance of $A_\theta$ on a typical training set of a given size when $\theta$ is tuned on the training set.

Given a dataset $S$, a cheap way of estimating (2) is to use the best CV-estimate over $\{A_\theta : \theta \in \Theta_0\}$,

$$\min_{\theta \in \Theta_0} \ell_S^{\text{cv}}(A_\theta)$$

Although this estimate tends to underestimate (2), in practice the difference is typically small.

A better, though more computationally intensive estimate of (2) is computed through nested CV.

```
Data: Dataset $S$
Split $S$ into folds $S_1, \dots, S_K$
for $i = 1, \dots, K$ do
    Compute training part of $i$-th fold: $S_{-i} \equiv S \setminus S_i$
    Run CV on $S_{-i}$ for each $\theta \in \Theta_0$ and find $\theta_i = \operatorname*{argmin}_{\theta \in \Theta_0} \ell_{S_{-i}}^{\text{cv}}(A_\theta)$
    Re-train $A_{\theta_i}$ on $S_{-i}$: $h_i = A_{\theta_i}\big(S_{-i}\big)$
    Compute error of $i$-th fold: $\varepsilon_i = \ell_{S_i}(h_i)$
end
Output: $(\varepsilon_1 + \cdots + \varepsilon_K)/K$
```

**Algorithm 1:** $K$-fold nested cross-validation

Note that in each run of internal cross-validation we optimize $\theta$ locally, on the training part $S^{(i)}$ of the external cross-validation fold. Hence, the nested cross-validation estimate is computed by averaging the performance of predictors obtained with potentially different values of their hyperparameters.