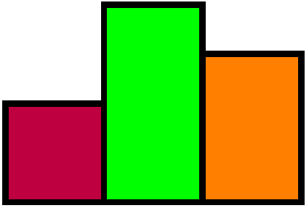
Introduction to Machine Learning

Evaluation: Introduction and Remarks



Learning goals

- Understand the goal of performance estimation
- Understand the difference between outer and inner loss
- Know the definition of generalization error

PERFORMANCE ESTIMATION

- After training our model, we are naturally interested in its **performance**.
- Recall that supervised learning is about finding the optimal model for our data at hand, given a set of hyperparameters:

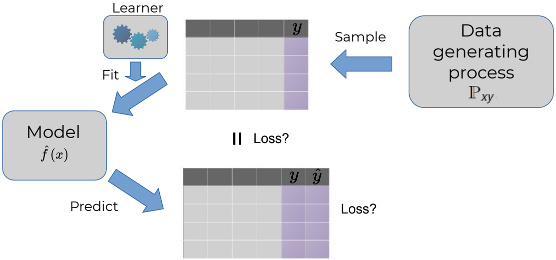
$$\mathcal{I} : \mathbb{D} \times \mathbf{\Lambda} \to \mathcal{H}, \quad (\mathcal{D}, \boldsymbol{\lambda}) \mapsto \hat{f}_{\mathcal{D}, \boldsymbol{\lambda}}.$$

- We obtain $\hat{f}_{\mathcal{D}, \boldsymbol{\lambda}}$ by means of empirical risk minimization, based on what we will now call **inner loss**.
- However, the inner loss does not necessarily tell us something about the performance of our learner – after all, we chose our model precisely so it would be loss-minimal on the data we trained it on.

$\rightarrow$ We cannot hope for $\hat{f}_{\mathcal{D}, \boldsymbol{\lambda}}$ to perform equally well on unseen data.
$\rightarrow$ Evaluation based on the inner loss would be **optimistically biased**.

PERFORMANCE ESTIMATION

- We wish to compute the true expected loss of our learner, referred to as **generalization error** or **outer loss**.



- In order to estimate performance on previously unseen observations, we need independent **test data**.
- As such a test set is not always available, we split the data at hand into non-overlapping sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$, with respective sizes $n_{\text{train}} + n_{\text{test}} = n$.

SET-BASED PERFORMANCE METRICS

- For the outer loss, we introduce a **set-based metric** ρ.
- This allows us to use outer losses that are defined on the entire test set rather than point-wise (e.g., the AUC), and potentially differ from the inner loss.
- For arbitrary data sets of size m and a prediction matrix

$$\boldsymbol{F} = \left[\hat{f}(\mathbf{x}^{(1)})^\top \quad \dots \quad \hat{f}(\mathbf{x}^{(m)})^\top\right]^\top \in \mathbb{R}^{m \times g}$$

returned by our model $\hat{f}$, ρ is defined as:

$$\rho : \bigcup_{m \in \mathbb{N}} \left(\mathcal{Y}^m \times \mathbb{R}^{m \times g}\right) \to \mathbb{R}, \quad (\mathbf{y}, \boldsymbol{F}) \mapsto \rho(\mathbf{y}, \boldsymbol{F}).$$

- We can easily translate this to our usual notion of point-wise loss L:

$$\rho_L(\mathbf{y}, \boldsymbol{F}) = \frac{1}{m} \sum_{i=1}^{m} L(y^{(i)}, \boldsymbol{F}^{(i)}) \quad \left(= \frac{1}{m} \sum_{i=1}^{m} L(y^{(i)}, \hat{f}(\mathbf{x}^{(i)}) \right).$$

GENERALIZATION ERROR

- This allows us to define the **generalization error** as:

$$\mathrm{GE}(\mathcal{I}, \boldsymbol{\lambda}, n_{\mathrm{train}}, \rho) := \lim_{n_{\mathrm{test}} \to \infty} \mathbb{E}\left[\rho\left(\mathbf{y}, \boldsymbol{F}_{\mathcal{D}_{\text{test}}, \mathcal{I}(\mathcal{D}_{\mathrm{train}}, \boldsymbol{\lambda})}\right)\right]$$

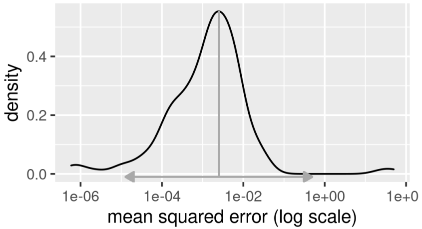
- The expectation is taken w.r.t. the unknown distribution $\mathbb{P}_{xy}$ from which both $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ are sampled independently.
- We must therefore estimate the generalization error, based on our train-test split:

$$\widehat{\mathrm{GE}}_{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}}(\mathcal{I}, \boldsymbol{\lambda}, n_{\mathrm{train}}, \rho) = \rho\left(\mathbf{y}_{\mathcal{D}_{\text{test}}}, \boldsymbol{F}_{\mathcal{D}_{\text{test}}, \mathcal{I}(\mathcal{D}_{\mathrm{train}}, \boldsymbol{\lambda})}\right).$$

- In practice, we will not rely on a single split, but use **resampling** to repeatedly carve out test observations from our data.

GENERALIZATION ERROR

- In order to get a better feeling for the quantities we are trying to estimate, let's look at the (pointwise) losses we can expect when applying a model on unseen test data.
- We fit a linear model to the `boston housing` regression task with $\mathcal{D}_{\text{train}}$ of fixed size $n_{\mathrm{train}} = 354$ (70% of total observations).
- From the remaining unseen data, we draw one observation at a time, feed it to the trained model, and compute the pointwise *L*2 loss.
- Then, we can plot the density of the stacked loss values:



- The result is a unimodal distribution with long tails.
- Mean and one standard deviation to either side are highlighted in grey.

INNER VS OUTER LOSS

- Supervised learning thus implies the following dichotomy:
  - **Learning**: minimize inner loss
  - **Evaluation**: estimate outer loss
- Beyond evaluating a single learner, the outer loss lends itself to comparing different types of learners, or learners with varying hyperparameter configurations $\boldsymbol{\lambda}$.
- Ideally, we have **inner loss = outer loss**. In this case, it holds by the law of the large numbers for the empirical risk $\mathcal{R}_{\text{emp}}$ that

$$\mathbb{E}_{\mathcal{D}\sim(\mathbb{P}_{xy})^{n_{\mathrm{train}}}}\left[\frac{1}{n}\sum_{i=1}^{n} L\left(y^{(i)}, \hat{f}_{\mathcal{D},\boldsymbol{\lambda}}(\mathbf{x}^{(i)})\right)\right] \xrightarrow{n\to\infty} \mathrm{GE}(\mathcal{I}, \boldsymbol{\lambda}, n_{\mathrm{train}}, \rho_L).$$

- This is not always possible – some special (set-based) metrics for evaluation, such as the AUC, are not applicable as inner loss.
- On the other hand, we sometimes wish to use losses that are hard to optimize or do not even specify one directly.

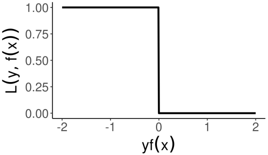
INNER VS OUTER LOSS

Example: Logistic regression

- An intuitive choice would be the share of incorrect predictions.
- This leads to the **mislassification error rate (MCE)**, computing the mean over pointwise **0-1 loss**.

- 0-1 loss simply assigns a loss of 1 for incorrect predictions and 0 otherwise:

$$L\left(y, h(\mathbf{x})\right) = \mathbb{I}_{\{y \neq h(\mathbf{x})\}}$$



- Problem: 0-1 loss is not differentiable (not continuous even).
  $\rightarrow$ This is why we use **binomial** loss as inner loss instead.
- For evaluation, differentiability is not required, so evaluation on **MCE** is feasible.