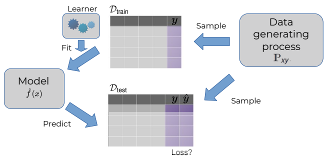
Introduction to Machine Learning

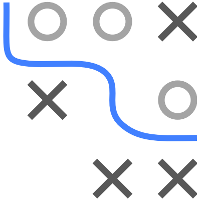
Evaluation: In a Nutshell


Learner
$\mathcal{D}_{\text{train}}$
y
Sample
Data generating process $\mathbb{P}_{xy}$
Fit
Model $\hat{f}(x)$
$\mathcal{D}_{\text{test}}$
y $\hat{y}$
Sample
Predict
Loss?

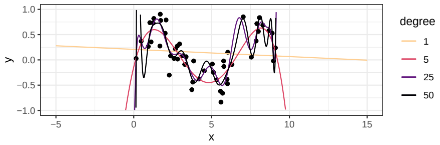
Learning goals

- Understand what the Generalization Error is
- Get an overview on how we evaluate performance of learners
- Know some evaluation metrics for classification and regression
- Understand why we do resampling

EVALUATING A MODEL

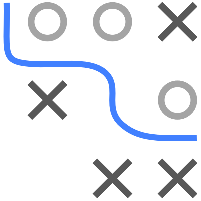


- We have seen how to train models that are optimal in some sense, relative to other possible models. However, how can we assess how good they actually are, in absolute numbers?
- Idea: Use risk $\sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}}}\left[L\left(y,\hat{f}(\mathbf{x})\right)\right]$ after training.
- Problem: This value can be very optimistic.
- Example: Overfitting of a polynomial regression.

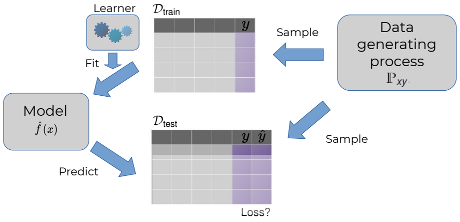


- Degree 50 will result in lowest training loss, however, degree 5 seems to be the "best" model.
- "Best" means that using new data, this model will probably produce the most meaningful predictions.

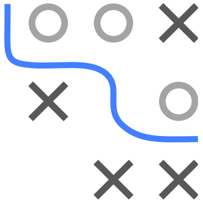
GENERALIZATION ERROR



- In other words, the "best" model will generalize well and have a low **G**eneralization **E**rror.¸
- Formally, for a fixed model, the GE can be expressed via:
  $\mathrm{GE}\left(\hat{f}, L\right) := \mathbb{E}\left[L\left(y, \hat{f}(\mathbf{x})\right)\right]$,
- i.e., "what is the expected loss for a new observation?"
- Ideally, the GE should be estimated with new, unseen data.
- Usually, we have no access to new **unseen** data, though.
- Thus, we divide our data set manually into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ and use the latter to estimate the GE via some metric $\rho()$.


Learner
$\mathcal{D}_{\text{train}}$
y
Sample
Data generating process $\mathbb{P}_{xy}$
Fit
Model $\hat{f}(x)$
$\mathcal{D}_{\text{test}}$
y $\hat{y}$
Sample
Predict
Loss?

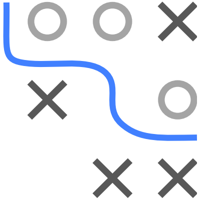
METRICS



But what is a good metric $\rho()$?

- While we can always use the (inner) loss function that we trained the model on as outer loss to construct a metric $\rho()$, this may not always be ideal.
- For both, classification and regression there is a large variety of evaluation metrics, of which we will just cover a fraction.

METRICS FOR CLASSIFICATION

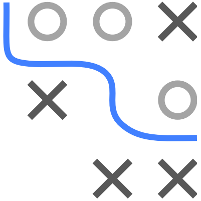


Commonly used evaluation metrics include:

- Accuracy:
  - $\rho_{ACC} = \frac{1}{m} \sum_{i=1}^{m} [y^{(i)} = \hat{y}^{(i)}] \in [0, 1]$.
  - "Proportion of correctly classified observations."
- Misclassification error (MCE):
  - $\rho_{MCE} = \frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \neq \hat{y}^{(i)}] \in [0, 1]$.
  - "Proportion of incorrectly classified observations."
- Brier Score:
  - $\rho_{BS} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{\pi}^{(i)} - y^{(i)}\right)^2$
  - "Squared error btw. predicted probability and actual label."
- Log-loss:
  - $\rho_{LL} = \frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} \log\left(\hat{\pi}^{(i)}\right) - \left(1 - y^{(i)}\right) \log\left(1 - \hat{\pi}^{(i)}\right)\right)$.
  - "Distance of predicted and actual label distribution."

The probabilistic metrics, Brier Score and Log-Loss, penalize false confidence, i.e. predicting the wrong label with high probability, heavily.

METRICS FOR CLASSIFICATION / 2



For hard-label classification, the confusion matrix is a useful tool:

		True Class y	
		$+$	$-$
Pred.	$+$	True Positive (TP)	False Positive (FP)
$\hat{y}$	$-$	False Negative (FN)	True Negative (TN)

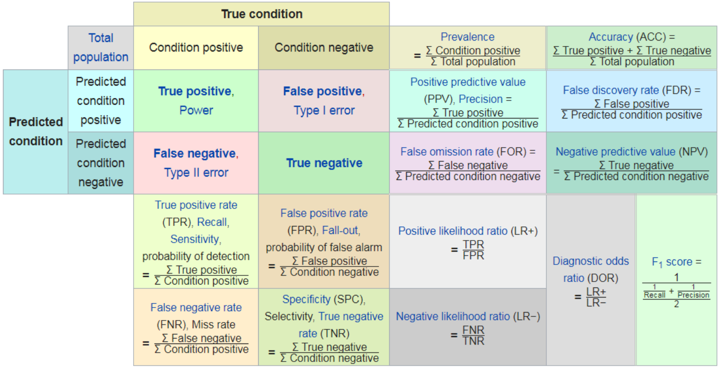
From this matrix a variety of evaluation metrics, including precision and recall, can be computed.

$$\mathit{Precision} = \frac{TP}{TP + FP}$$

$$\mathit{Recall} = \frac{TP}{TP + FN}$$

METRICS FOR CLASSIFICATION / 3

- Other frequently used metrics like the False Negative Rate can also be derived from the confusion matrix.
- Many of these metrics can go with different names.

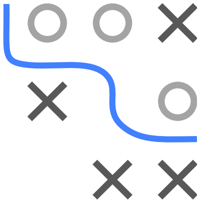


		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= \frac{1}{\frac{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}{2}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

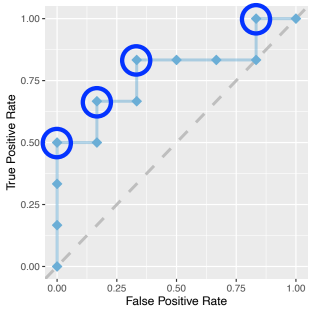
▸ Clickable version/picture source

▸ Interactive diagram

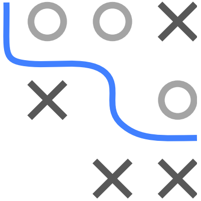
ROC-CURVE



- The ROC-Curve allows to evaluate binary classifiers beyond single metrics. It compares classifiers using their TPR and FPR, for different thresholds.
- We aim to identify good thresholds that dominate others.
- The area under this curve (AUC) can also be used as metric.



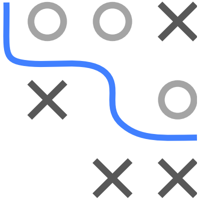
METRICS FOR REGRESSION



Commonly used evaluation metrics include:

- Sum of Squared Errors (SSE): $\rho_{SSE} = \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$
- Mean Squared Error (MSE): $\rho_{MSE} = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$
- Root Mean Squared Error (RMSE): $\rho_{RMSE} = \sqrt{MSE}$
- R-Squared: $\rho_{R^2} = 1 - \frac{\sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{m} (y^{(i)} - \bar{y})^2}$
- Mean Absolute Error (MAE): $\rho_{MAE} = \frac{1}{m} \sum_{i=1}^{m} |y^{(i)} - \hat{y}^{(i)}|$

IMPROVING ESTIMATION OF GE



We can estimate the GE with a test data set via:

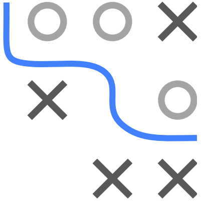
$$\widehat{\mathrm{GE}}(\hat{f}, L) := \frac{1}{m} \sum_{(\mathbf{x}, y) \in \mathcal{D}_{\text{test}}} \left[L\left(y, \hat{f}(\mathbf{x})\right)\right],$$

i.e. we compute the selected metric $L\left(y, \hat{f}(\mathbf{x})\right)$ for each observation in the test set and compute the mean.

This will give an appropriate estimate for the GE. However, with only a few test observations (small m), this estimate will be unstable or, in other words, have high variance. We have two options to decrease it:

- Increase m.
- Compute $\widehat{\mathrm{GE}}(\hat{f}, L)$ for multiple test sets and aggregate them.

RESAMPLING



As we do not have access to infinite data and increasing *m* will mean a reduction of the number of training observations, aggregation over *B* sets is the preferred option:

$$\mathcal{J} = ((J_{\text{train},1}, J_{\text{test},1}), \dots, (J_{\text{train},B}, J_{\text{test},B})) .$$

We compute $\widehat{\mathrm{GE}}(\hat{f}, L)$ for each set and aggregate the estimates. These *B* distinct sets are generated through **resampling**.

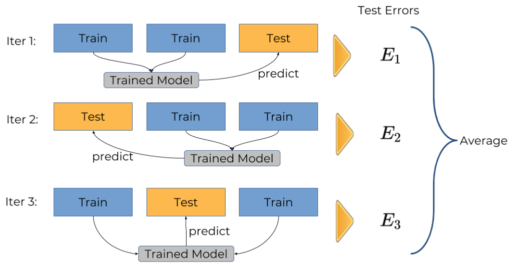
There exist a number of well-established resampling strategies:

- (Repeated) Hold-out / Subsampling
- Cross validation
- Bootstrap

RESAMPLING

All methods aim to generate the train-test splits $\mathcal{J}$ by splitting the full data set repeatedly. The model is trained on the respective train set and evaluated on the test set.

Example: 3-fold cross validation



In order to robustify performance estimates, we can repeat resampling.