Lecture 2. Supervised learning

Machine Learning

Sergey Muravyov

25.01.2024

Lecture plan

- **Supervised learning**
- Overfitting problem
- Cross-validation
- Supervised learning problem evaluation

The task of Supervised learning (reminder)

Supervised learning, Supervised learning on labeled data (examples, precedents), approximation, Supervised learning

- A machine learning task in which the training dataset contains correct answers that the algorithm must learn to predict for new data.
- Labeled object - an object which has a value for a target feature.

Possible supervised learning tasks depending on the type of Y :

- Classification problem: $Y = \{c_1, c_2, \ldots, c_k\}$.
- Probabilistic classification problem: $Y = \Pr^k$.
- Regression problem: $Y = \mathbb{R}^k$.
- Ranking problem: Y permutation or ordinal type.

Naive algorithms

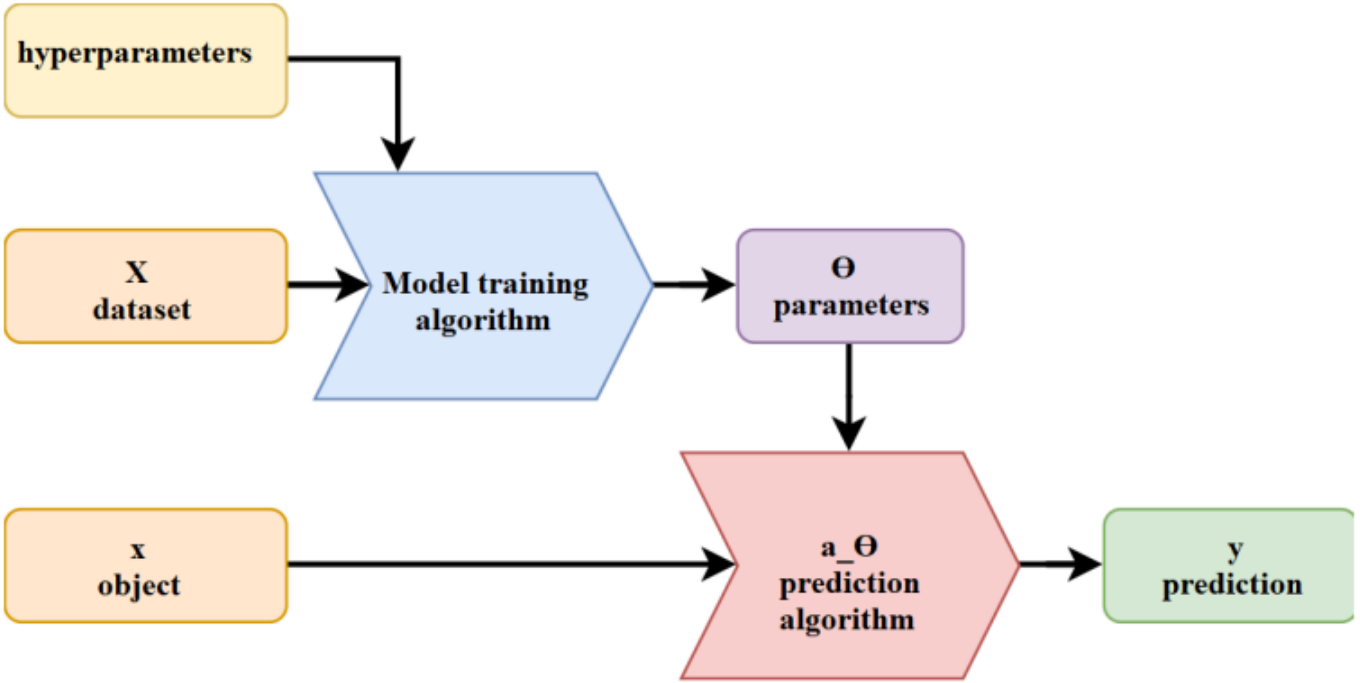
An algorithm that uses only Y

- **Classification** problem: $a(x) = Mode[Y]$
  (Most commonly encountered meaning).
- **Regression** recovery problem: $a(x) = E[Y]$
  (minimizes not all error functions).

Comparison with the naive algorithm

- If the algorithm performs worse or comparable to the naive algorithm, it means that it does not find the dependency of Y on X and therefore only uses Y .
- Consequently, the model is not strong enough to recover the dependency or there is no dependency at all.

Training algorithm schema

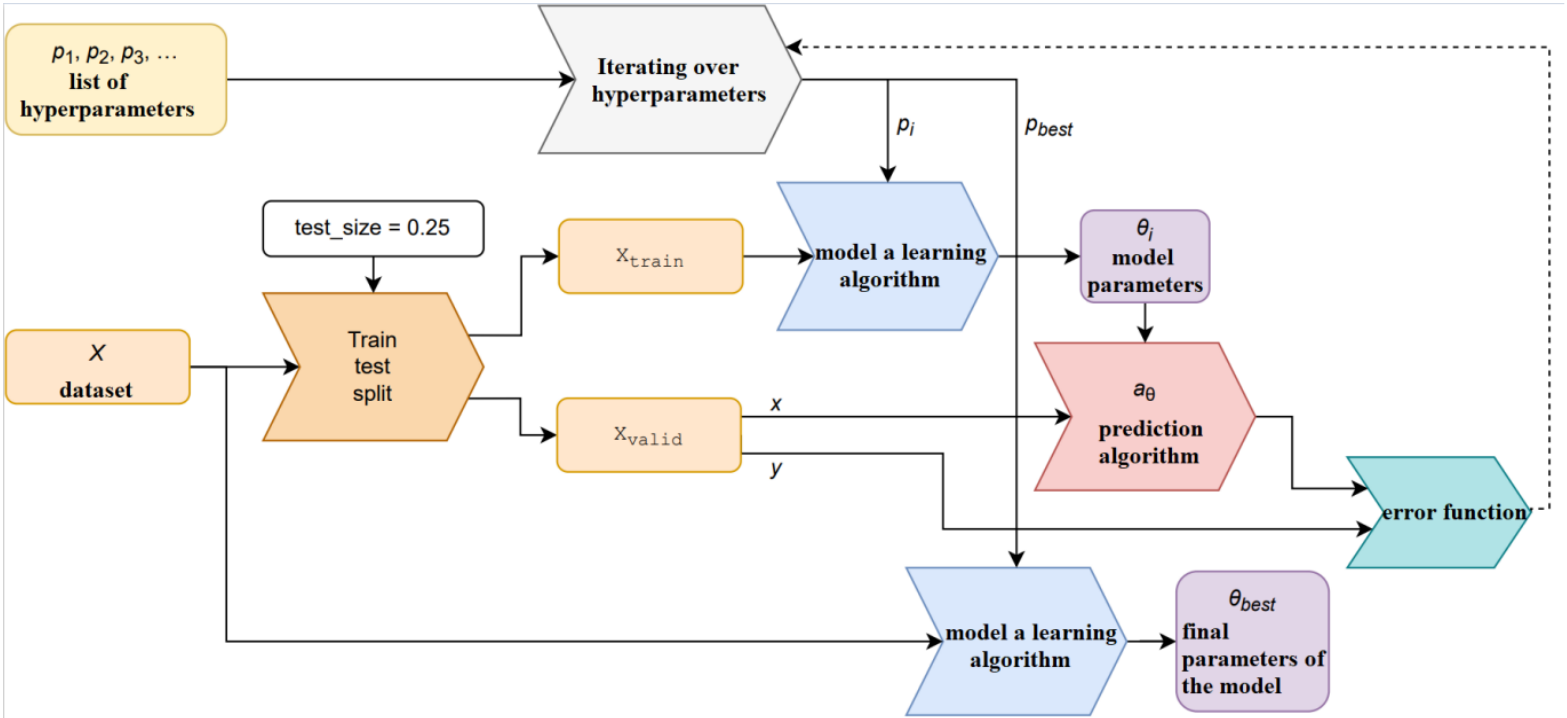


Hyperparameters

Different definitions:

- Parameters of the learning algorithm.
- Parameters that do not change during learning.
- Parameters that can be set prior to observing a dataset.
- Structural parameters.

Hyperparameter tuning as part of learning (1/2)

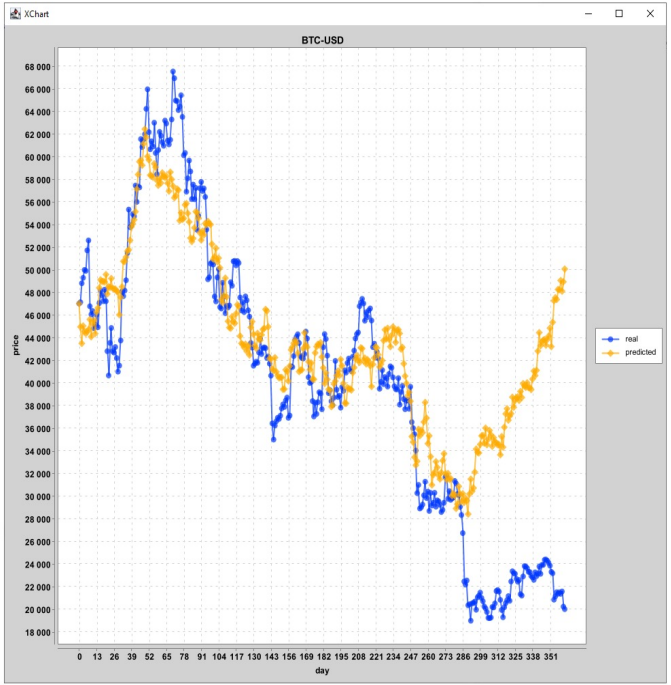


Hyperparameter tuning as part of learning (2/2)

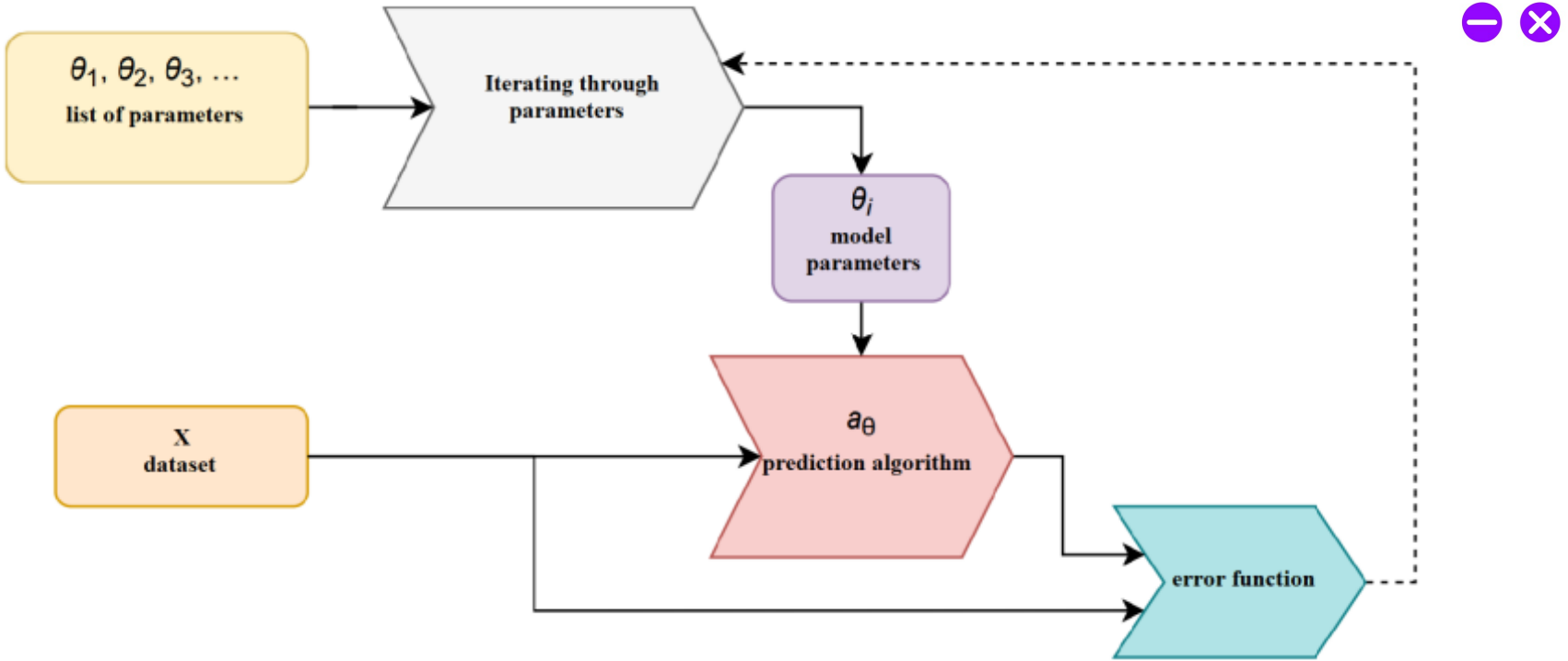
- **Validation set** is a test set that is used to validate hyperparameters during their tuning process.
- Sometimes, the tuning algorithm doesn't perform a final learning.
- Typically, the learning algorithm implicitly tunes the hyperparameters if it can do so more efficiently than explicitly. For example, if you change model hyperparameters, you may not need to retrain the model.
- The initial state of a pseudorandom number generator (seed) cannot be tuned!

Example of setting seed for a useless algorithm

```
for (int rep = 0; rep < 1000000; rep++) {
    Random random = new Random(rep);

    double sigma = random.nextGaussian() * 500;

    double[] c = new double[n];
    c[0] = p[0];

    for (int i = 1; i < n; i++) {
        c[i] = c[i - 1] + random.nextGaussian() * sigma;
    }

    double sum = 0;
    int trainPart = 3 * n / 4;
    for (int i = 0; i < trainPart; i++) {
        double e = c[i] - p[i];
        sum += e * e;
    }
    if (sum < diff) {
        diff = sum;
        r = c;
    }
}
```



Hyperparameter tuning as part of training



Hyperparameter tuning problem

$$p_{\text{best}} = HyperParameterTuning(A, \mathcal{D}, \mathcal{L}) = \arg\min_p \mathcal{L}(A_p, \mathcal{D})$$

where $\mathcal{D}$ – dataset, A – training algorithm,
p – hyperparameters of the training algorithm,
$\mathcal{L}$ – loss function (validation).

What is being solved:

- Grid search / Random search
- Evolutionary algorithms
- Bayesian optimization

Lecture plan

- Supervised learning
- **Overfitting and regularization**
- Cross-validation
- Supervised learning problem evaluation

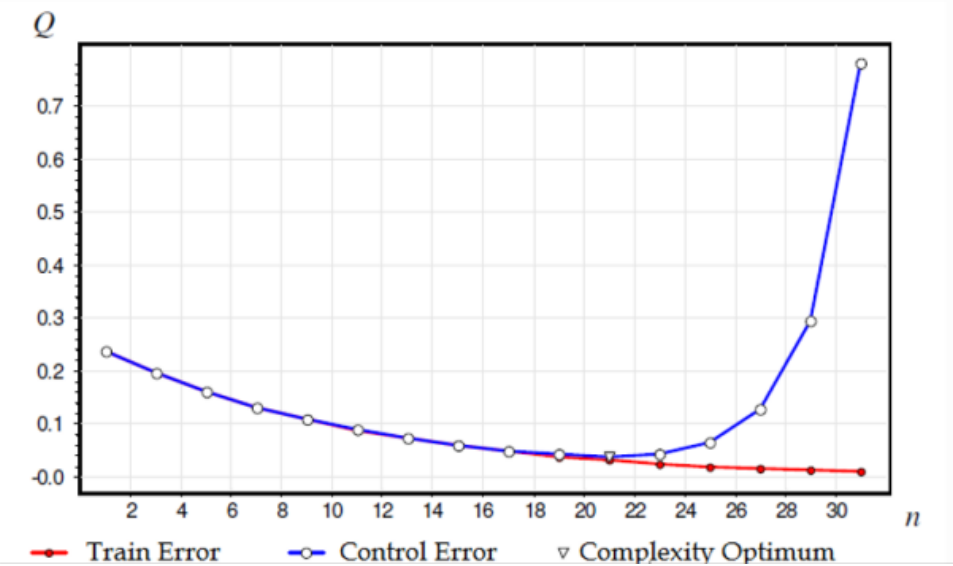
Overfitting problem

Overfitting problem starting from a certain model complexity level, the better an algorithm performs on train set D the worse it performs on real world objects.

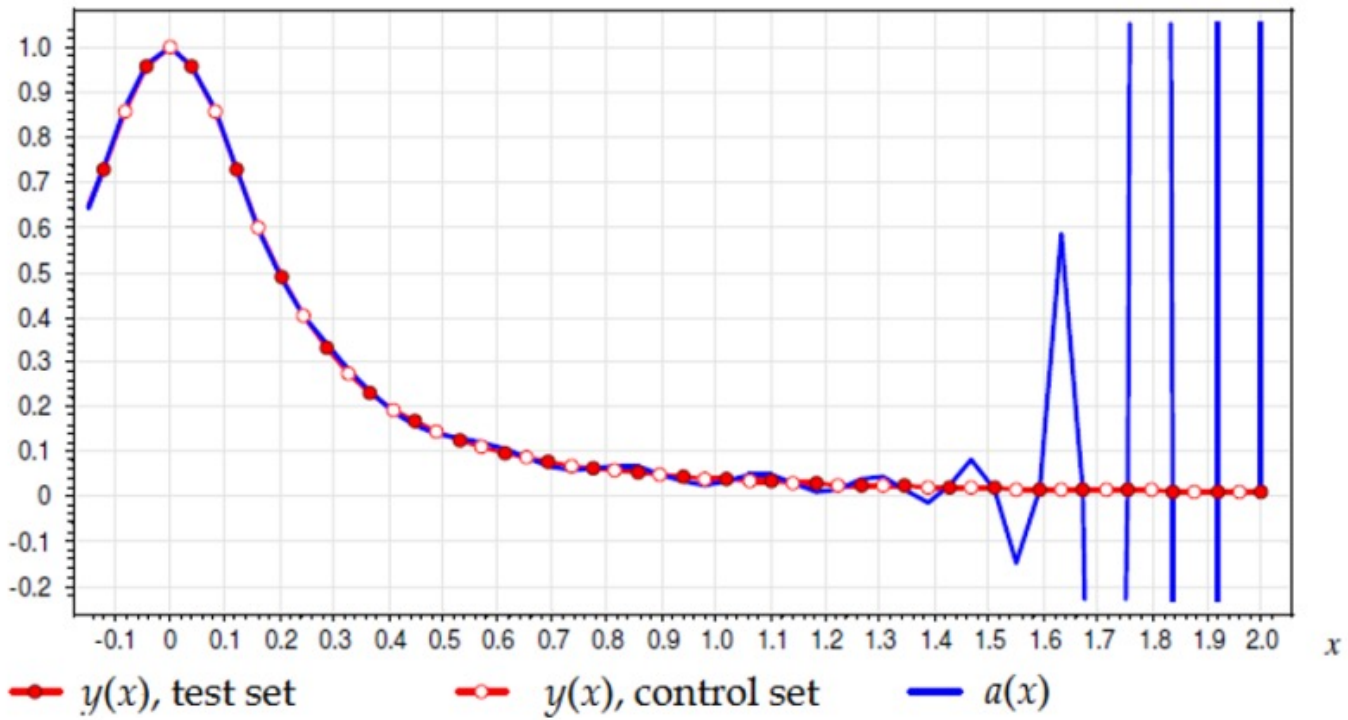
Example of overfitting

Dependency defined on $x \in [-2, 2]$. $y(x) = \frac{1}{1+25x^2}$

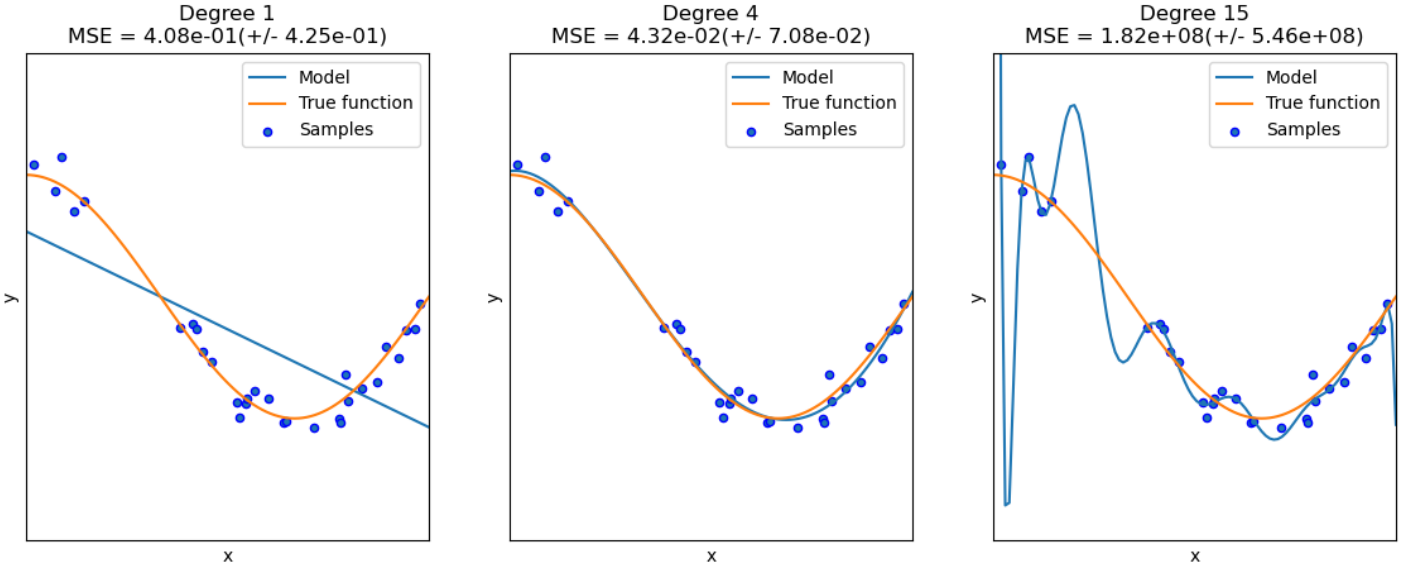
Let search a function among polynomials with degree.



Overfitted algorithm



Underfitting and Overfitting



Regularization

Regularization is a method of adding additional constraints to a condition in order **to prevent overfitting** or **improve the generalization** of models. It is typically in the form of a penalty for model complexity.

- Soft (not strict) regularization:

$$\mathcal{L}_{\text{reg}}(\theta, \mathcal{D}) = \mathcal{L}(\theta, \mathcal{D}) + \lambda \cdot complexity(\theta)$$

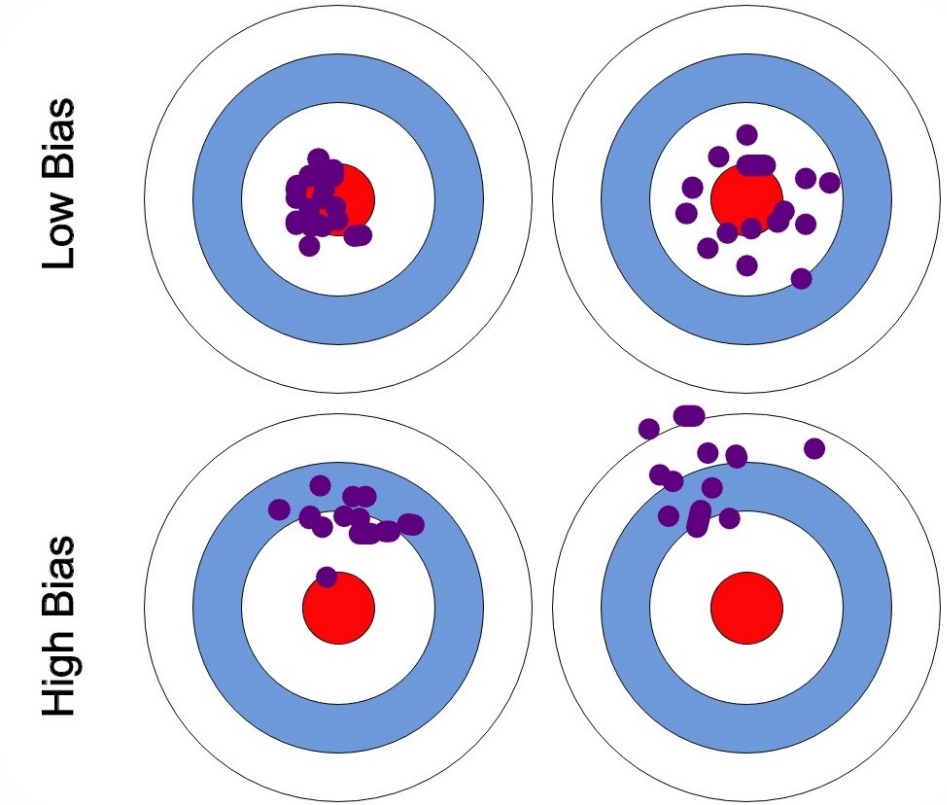
where λ regularization factor.

- Hard (strict) regularization:

$$\begin{cases} \mathcal{L}(\theta,\mathcal{D}) \xrightarrow{\theta} \min \\ \text{complexity}(\theta) < C \end{cases}$$

where C complexity constraint.

Bias and Variance



- Bias is an estimation error that arises from incorrect assumptions in the learning algorithm (underfitting).

- Variance is an error caused by sensitivity to small fluctuations in the training set (overfitting).

Decomposition of bias-variance trade-off in quadratic error

- Reminder: $D[Z] = E[(Z - E[Z])^2] = E[Z^2] - E[Z]^2 \Rightarrow E[Z^2] = D[Z] + E[Z]^2$.

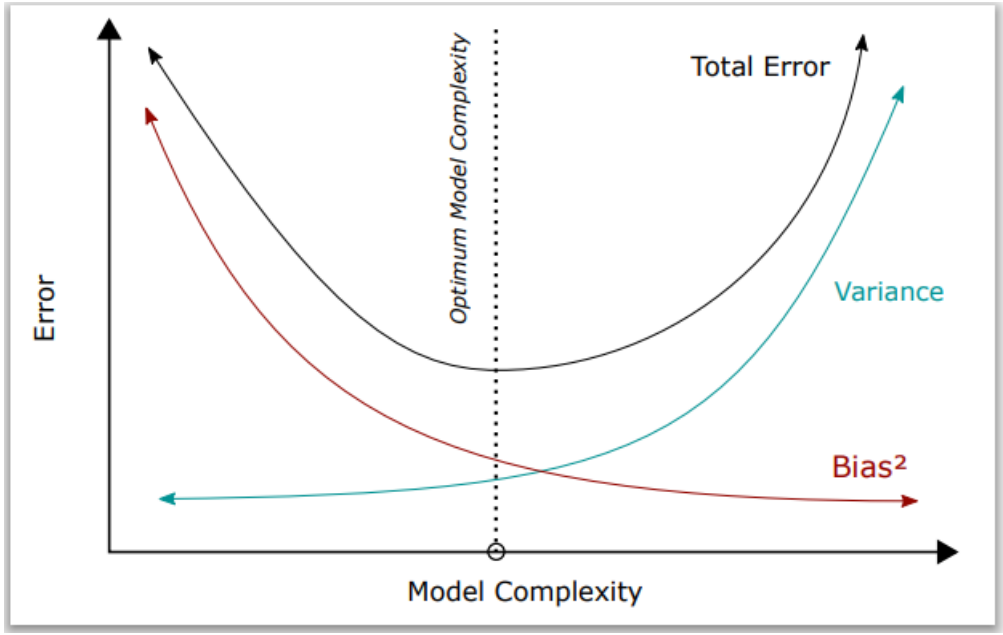
- Let f be the actual relationship and $\hat{y}$ be the prediction of the model.
- $y = f(x) + \varepsilon$ Measurement with noise $\varepsilon, E[\varepsilon] = 0$ и $D[\varepsilon] = \sigma^2$.
- $E[y] = E[f + \varepsilon] = E[f] + E[\varepsilon] = E[f] = f$.
- $D[y] = E[(y - E[y])^2] = E[(y - f)^2] = E[(f + \varepsilon - f)^2] = E[\varepsilon^2] = D[\varepsilon] + E[\varepsilon]^2 = \sigma^2$.

- Then the expectation of the quadratic error $E[(y - \hat{y})^2]$

$$
\begin{aligned}
&= E[y^2 + \hat{y}^2 - 2y\hat{y}] \\
&= E[y^2] + E[\hat{y}^2] - E[2y\hat{y}] \\
&= D[y] + E[y]^2 + D[\hat{y}] + E[\hat{y}]^2 - 2f\mathrm{E}[\hat{y}] \\
&= D[y] + D[\hat{y}] + (f^2 - 2f\,E[\hat{y}] + E[\hat{y}]^2) \\
&= D[y] + D[\hat{y}] + (f - E[\hat{y}])^2 \\
&= \sigma^2 + D[\hat{y}] + Bias[\hat{y}]^2
\end{aligned}
$$

- Therefore, the error is composed of an irreducible measurement error, the model's, **variance** and the square of the model's **bias**.

Bias-variance tradeoff



Double Descent

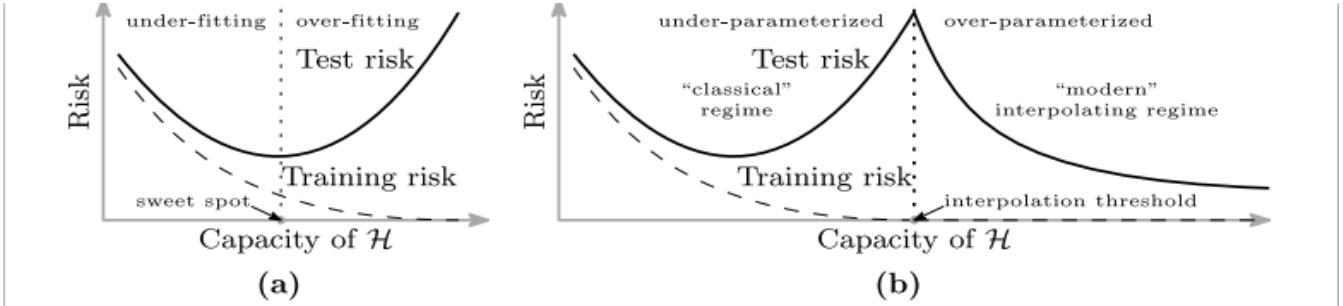


Figure 1: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Source: Belkin, M., Hsu, D.J., Ma, S., & Mandal, S. (2018). Reconciling modern machine learning and the bias-variance trade-off.

Lecture plan

- Supervised learning
- Overfitting and regularization
- **Cross-validation**
- Supervised learning problem evaluation

Full cross-validation

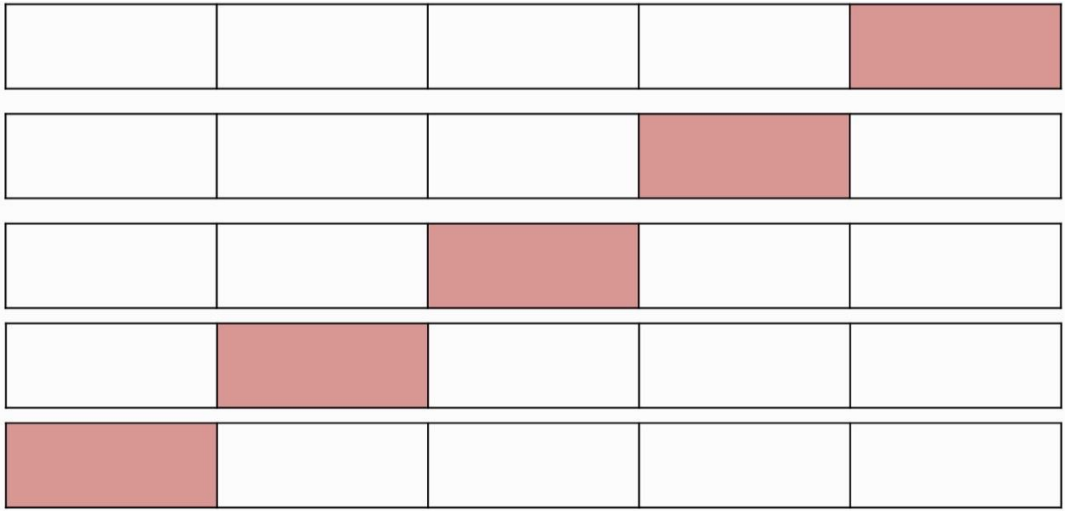
1. Fix the values r and e:
   $$|\mathcal{D}_{train}| = r,\ |\mathcal{D}_{test}| = e.$$
2. Split $\mathcal{D}$ to $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ in all possible ways of the appropriate size.
3. Validate A against $\mathcal{D}$ using the formula:

$$\mathcal{L}(A, \mathcal{D}) = \frac{1}{C_{e+r}^{e}} \sum_{\mathcal{D}=\mathcal{D}_{train} \cup \mathcal{D}_{test}} \mathcal{L}\left(A\left(\mathcal{D}_{train}\right), \mathcal{D}_{test}\right)$$

K-fold cross-validation (1/2)

k-fold Cross-Validation, CV

Split the training dataset into k parts, where k is the number of folds



K-fold cross-validation (2/2)

ITMO

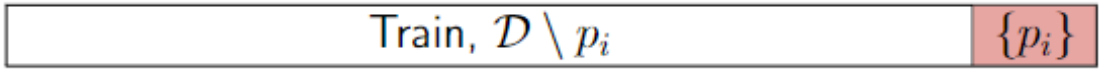
- Each of the k blocks is tested exactly once.
- k is usually 10 (sometimes 5, in rare cases 3).

1. Split $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_k, |\mathcal{D}_i| \approx |\mathcal{D}_j|$.
2. Validate A against $\mathcal{D}$ using the formula:

$$\mathcal{L}(A, \mathcal{D}) = \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A(\mathcal{D} \setminus \mathcal{D}_i), \mathcal{D}_i)$$

Leave-one-out cross-validation

Leave-one-out cross-validation (LOO)



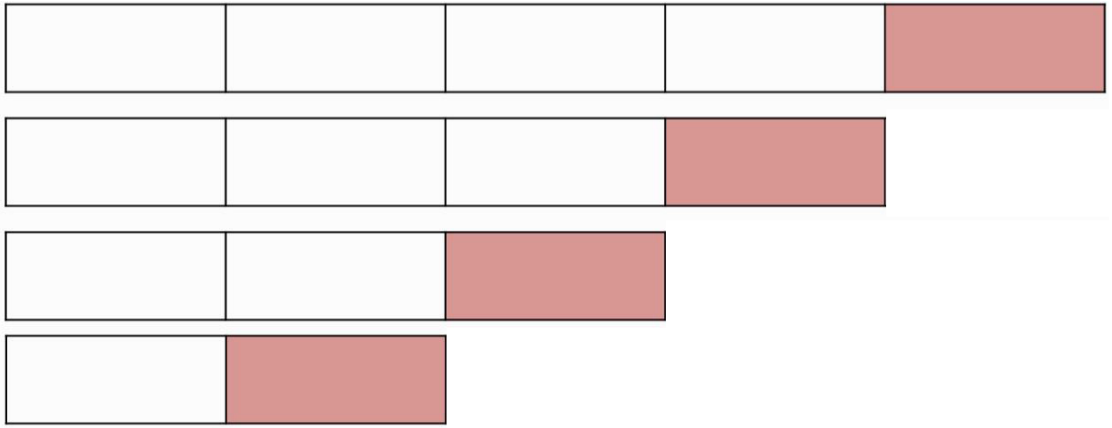
1. Split the training set into $|\mathcal{D}| - 1$ and 1 object $|\mathcal{D}|$ times.
2. Validate A against $\mathcal{D}$ using the formula:

$$\mathcal{L}(A, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathcal{L}(A(\mathcal{D} \setminus x)(x), y(x))$$

Stratified k-fold cross-validation

Stratified k-fold cross-validation
If objects of certain classes or with certain feature values are infrequently observed in the dataset, it is better to split them into blocks so that the statistics in each block are proportional to the statistics in the dataset.

Time-aware cross-validation



- Earlier blocks will result in a poorer outcome.
- The arithmetic mean is replaced with a weighted exponential moving average of blocks.

Note

- It is only possible to average with a sum ∑ for additive error/quality functions.
- To perform averaging t-fold cross-validation can be used with k blocks ($t \times k$-fold cross-validation) repeated t times cross-validation with k blocks.
- If cross-validation is used for hyperparameter tuning, then the model needs to be retrained on the entire dataset.

What else can lead to overfitting

- Biased (non-representative) dataset.
- Poorly chosen quality metric for measuring algorithms (e.g. accuracy for rare classes).
- Systematic biases in validation methods.
- Lack of understanding of hidden hyperparameters.

Cross-validation is powerless in these cases.

Lecture plan

- Supervised learning
- Overfitting and regularization
- Cross-validation
- **Supervised learning problem evaluation**

Evaluation of the prediction result

- The model a approximates the dependence of y.
- It is required to evaluate the prediction result of $\hat{y}$ using the function $\mathcal{L}(y, \hat{y})$.
- For classification problem: y and $\hat{y}$ are arrays of categories.
- For regression problem: y and $\hat{y}$ are arrays of numbers.

Classification problem evaluation

Reminder: Everything that can be done with categories is to test for equality.

- **Accuracy**: $Accuracy(y, \hat{y}) = \frac{1}{n} \cdot \sum_{i=1}^{n} [y_i = \hat{y}_i]$, where y vector of the real answers, $\hat{y}$ vector of the predictions, n the number of objects in the test set.
  1. Is 70% accuracy good?
  2. What if 70% of objects in the test dataset belong to one class?
- **Kappa** Coefficient: $\kappa(po, pe) = \frac{po - pe}{1 - pe}$,
  where po obtained result (accuracy), and pe random / naive.
- **Error**: ErrorRate$(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} [y_i \neq \hat{y}_i] = 1 - Accuracy$.

Exercise
Let p_c% of objects belong to class c in the test set. Analytically find the accuracy for: naive solution (mode), random answer with prior probabilities p, equiprobable random answer

Confusion Matrix

Confusion matrix
CM a square matrix of size $k \times k$, where $CM_{t,c}$ the number of objects of class t, classified as class c, and k the number of classes.

$$CM(y, \hat{y})_{t,c} = \sum_{i=1}^{n} [(y_i = t) \wedge (\hat{y}_i = c)]$$

- The most general method is to reflect information about the testing results.
- The result of the testing for cross-validation: **one** matrix!

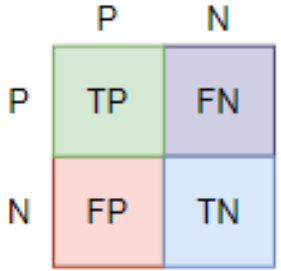
Exercise
Express Accuracy and errorRate through the CM.

Errors of the first and second kind

- TP (True Positive) Number of true positive answers
- TN (True Negative) Number of true negative answers
- FP (False Positive) Number of Type I errors
- FN (False Negative) Number of Type II errors

- **Precision** (not Accuracy): Precision = $\frac{TP}{TP + FP}$
- **Recall**: Recall = $\frac{TP}{TP + FN}$
- **JaccardIndex**: JaccardIndex = $\frac{TP}{TP + FP + FN}$

Class order bias!



F-score

- F-score the **harmonic** mean between precision and recall.

$$F1 = \frac{2}{\text{Precision}^{-1} + \text{Recall}^{-1}} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

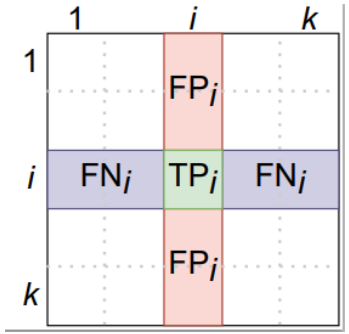
- If we take the **geometric** mean instead of the harmonic mean, we get the Fowlkes-Mallows index. It is a less strict measure.
- **Weighted** harmonic mean:

$$F\beta = (1 + \beta^2) \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

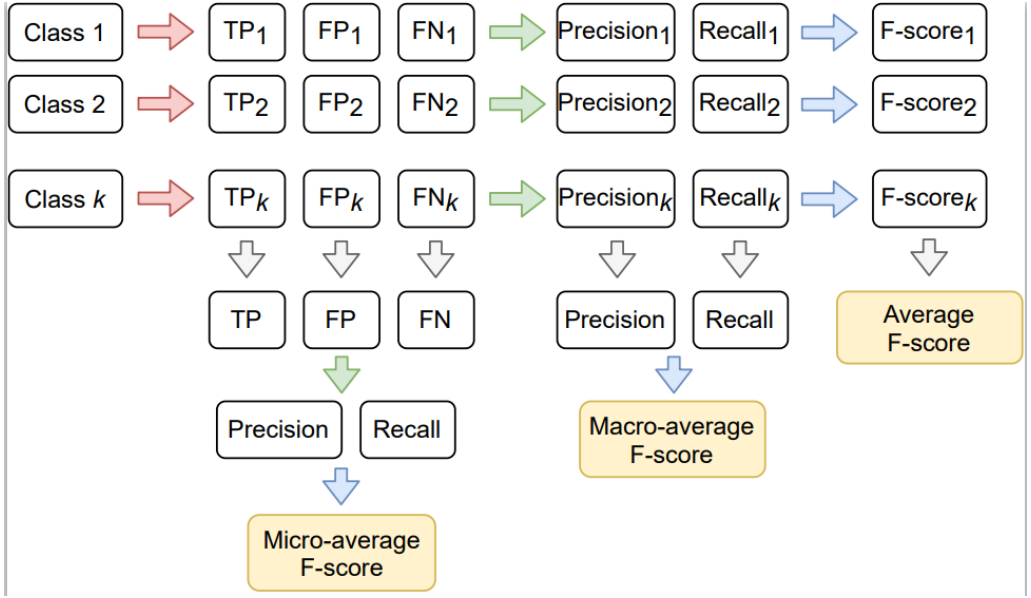
Exercise

- Find the weights of recall and precision for the weighted harmonic mean, which results in the F_β -score.
- What will happen to the mean F-score if multiple confusion matrices are used during LOO cross-validation?

F-score for multiple classes

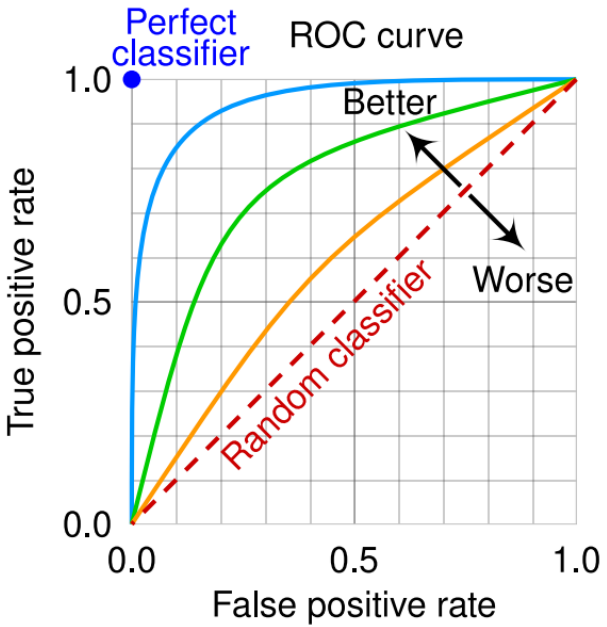


One vs. all: each class becomes ``positive'' exactly once, while the rest become negative.



AUC ROC

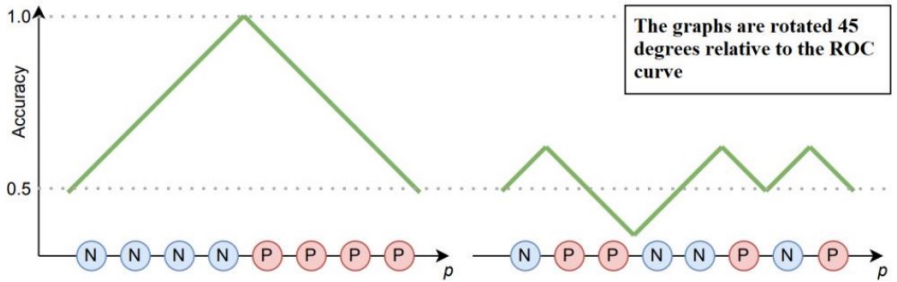
- **Sensitivity** True Positive Rate (TPR):
$$TPR = \frac{TP}{P} = \frac{TP}{TP+FN} = \text{Recall}.$$
- **Specificity** True Negative Rate (TNR):
$$TNR = \frac{TN}{N} = \frac{TN}{TN+FP}.$$
- **Receiver Operating Characteristic** dependency of TPR on TNR.
- **AUC Area Under Curve** binary classification quality function.
- Unbiased to the order of classes.



Building a ROC curve

Soft classification is required for solving a binary classification problem.

1. For each object i in the test set, the algorithm reports a number p_i the probability that this object belongs to the positive class.
2. Set the “pencil” to the lower left corner.
3. Go through the pairs (p_i, y_i) in ascending order of p_i:
   - If the class y_i is positive, move the “pencil” to the right.
   - If the class y_i is negative, move the “pencil” up.



Error functions for regression problem (1/8)

Suppose n objects were tested: y_i is the real target value of the i-th object, and $\hat{y}_i$ is the corresponding prediction.

Sum of Squares (SS)

$$SS(y, \hat{y}) = \|y - \hat{y}\|_2^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Popular error function for training.
- Derivative $\frac{\partial SS}{\partial \hat{y}} = 2(\hat{y}_i - y_i)$.

\+ It can be quickly computed as the dot product of the vector of differences with itself.

− It is incorrect to compare results for arrays of different sizes.

Error functions for regression problem (2/8)

Mean Square Error (MSE)

$$MSE(y, \hat{y}) = \frac{SS(y, \hat{y})}{n} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- If $\hat{y}_i = E[Y]$, then $MSE(y, \hat{y}) = D[Y]$ the variance.
- It is often confused with SS when used for training.

– The dimensionality does not match the dimensionality of the target feature.

Error functions for regression problem (3/8)

Root-Mean-Square Error (RMSE)

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} = \sqrt{MSE(y,\hat{y})}$$

- If $\hat{y}_i = E[Y]$, then $RMSE(y, \hat{y}) = \sigma[Y]$ the standard deviation.

+ The minimum coincides with MSE.

+ The dimensionality matches the dimensionality of the target feature.

− A baseline is needed to interpret the result.

Error functions for regression problem (4/8)

Normalized Root-Mean-Square Error (NRMSE)

$$NRMSE(y, \hat{y}) = \frac{RMSE(y, \hat{y})}{\sigma[Y]} = \sqrt{\frac{SS(y, \hat{y})}{SS(y, \bar{y})}}$$

- If $\hat{y}_i = E[Y]$, then $NRMSE(y, \hat{y}) = 1$.

+ If the dataset is normalized, then $NRMSE = MSE$.
+ Easier to interpret the result without a naive solution (baseline).
+ Dimensionless.
+ The minimum coincides with MSE

− The type of normalization needs to be specified, sometimes.

$$NRMSE(y, \hat{y}) = \frac{RMSE(y, \hat{y})}{\max[Y] - \min[Y]}.$$

Error functions for regression problem (5/8)

Coefficient of determination

$$R^2 = 1 - (NRMSE(y, \hat{y}))^2 = 1 - \frac{SS(y,\hat{y})}{SS(y, \overline{y})}$$

This is a function of quality, not errors.

Mean Absolute Error (MAE)

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- Optimal naive solution: $\hat{y}_i$ = Median$[Y]$.

\+ The dimensionality matches the dimensionality of the target feature.

Error functions for regression problem (6/8)

Mean Absolute Percentage Error (MAPE)

$$MAPE(y, \hat{y}) = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

+ Dimensionless.
− Needs further clarification at $y_i = 0$.

Error functions for regression problem (7/8)

Symmetric Mean Absolute Percentage Error (SMAPE)

$$SMAPE(y, \hat{y}) = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|}$$

+ Dimensionless.
+ Symmetric with respect to its arguments.
● y_i = 100 and $\hat{y}_i$ = 110 will give a smaller $SMAPE$ than y_i = 100 and $\hat{y}_i$= 90.
− Requires clarification. Sometimes $|y_i| + |\hat{y}_i|$ is replaced with $y_i + \hat{y}_i$.
− Requires clarification. Sometimes 100% is replaced with 200%.
− Needs further clarification at y_i = 0.

Error functions for regression problem (8/8)

Note

- Any distance function is an error function, any similarity function is a quality function.
- Error functions are mainly used for regression problems, while quality functions are used for classification problems.
- The result of cross-validation is a single vector of predictions.

Exercise

How to generalize metrics for the case of regression with multiple features?

THANK YOU
FOR YOUR ATTENTION!

ITMO more than a UNIVERSITY

THANK YOU
FOR YOUR ATTENTION!

ITMO more than a UNIVERSITY