Machine Learning

Supervised Learning. Features. Loss Functions. Cross-validation

Aleksandr Petiushko

ML Research

October 9th, 2023

Content

Content

Content

Content

Content

Content

1. Supervised Learning Setting
2. Objects' features
3. Model outputs
4. Loss functions
5. Cross-validation
6. Hyperparameters tuning

Machine Learning Paradigms: reminder

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the unknown dependency

Machine Learning Paradigms: reminder

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the <u>unknown</u> dependency

Machine learning paradigms

- **Supervised** (now)
  — Sufficient amount of training material, i.e. pairs (x_i, y_i)

Machine Learning Paradigms: reminder

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the unknown dependency

Machine learning paradigms

- **Supervised** (now)
  - Sufficient amount of training material, i.e. pairs (x_i, y_i)
- Semi-supervised
  - Few labeled data (x_i, y_i) and many unlabeled examples x_j

Machine Learning Paradigms: reminder

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the <u>unknown</u> dependency

Machine learning paradigms

- **Supervised** (now)
  — Sufficient amount of training material, i.e. pairs (x_i, y_i)
- Semi-supervised
  — Few labeled data (x_i, y_i) and many unlabeled examples x_j
- *Unsupervised* (in future lectures?)
  — No labeled pairs, only x_i examples

Machine Learning Paradigms: reminder

Definitions

- X — set of objects
- Y — set of (correct) answers/labels
- $y : X \to Y$ — the unknown dependency

Machine learning paradigms

- **Supervised** (now)
  - Sufficient amount of training material, i.e. pairs (x_i, y_i)
- Semi-supervised
  - Few labeled data (x_i, y_i) and many unlabeled examples x_j
- *Unsupervised* (in future lectures?)
  - No labeled pairs, only x_i examples
- Reinforced
  - Action generation based on interaction with the environment

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.
- Need to clarify:

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.
- Need to clarify:
  - How objects are defined

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.
- Need to clarify:
  - How objects are defined
  - How answers are given

Problem setting for supervised learning

- Given:
  - $\{(x_1, y_1), ..., (x_n, y_n)\} \subset X \times Y$ — training set
- Find
  - A decision function $a : X \to Y$ that approximates the target dependency y.
- Need to clarify:
  - How objects are defined
  - How answers are given
  - What does it mean that one dependency approximates another

How objects are defined

Definition

Object = set of features

How objects are defined

Definition

Object = set of features

Feature types

- Categorical feature

How objects are defined

Definition

Object = set of features

Feature types

- Categorical feature
- Binary attribute
  - A special case of categorical, when category = "does this property exist or not"

How objects are defined

Definition

Object = set of features

Feature types

- Categorical feature
- Binary attribute
  - A special case of categorical, when category = “does this property exist or not”
- Ordinal attribute
  - Full (or partial) order within categories

How objects are defined

Definition

Object = set of features

Feature types

- Categorical feature
- Binary attribute
  - A special case of categorical, when category = “does this property exist or not”
- Ordinal attribute
  - Full (or partial) order within categories
- Quantitative attribute

How answers are given

Classification tasks

- Binary classification $Y = \{-1, 1\}$ or $Y = \{0, 1\}$

How answers are given

Classification tasks

- Binary classification $Y = \{-1, 1\}$ or $Y = \{0, 1\}$
- Multiclass classification $Y = \{0, 1, ..., M - 1\}$

How answers are given

Classification tasks

- Binary classification $Y = \{-1, 1\}$ or $Y = \{0, 1\}$
- Multiclass classification $Y = \{0, 1, ..., M - 1\}$
- Multivalued binary classification $Y = \{0, 1\}^M$

How answers are given

Classification tasks

- Binary classification $Y = \{-1, 1\}$ or $Y = \{0, 1\}$
- Multiclass classification $Y = \{0, 1, ..., M - 1\}$
- Multivalued binary classification $Y = \{0, 1\}^M$

Regression Tasks

$Y = \mathbb{R}$ or $Y = \mathbb{R}^n$

Loss Function

Definition

Loss function $L(a, x)$ — error value of algorithm a on object x

Loss Function

Definition

Loss function $L(a, x)$ — error value of algorithm a on object x

Loss functions for classification problems

$L(a, x) = [a(x) \neq y]$ — error indicator function (either 0 or 1)

Loss Function

Definition

Loss function $L(a, x)$ — error value of algorithm a on object x

Loss functions for classification problems

$L(a, x) = [a(x) \neq y]$ — error indicator function (either 0 or 1)

Loss functions for regression problems

$L(a, x) = (a(x) - y)^2$ — squared error

Comparison of machine learning models

How do you know that one model is better than another?

To do this, we use a set which is independent of **training** set, which is called **test** set

Comparison of machine learning models

How do you know that one model is better than another?

To do this, we use a set which is independent of **training** set, which is called **test** set

Why even bother with this?

- There are many machine learning algorithms and it is important to understand which one is more applicable to a particular task

Comparison of machine learning models

How do you know that one model is better than another?

To do this, we use a set which is independent of **training** set, which is called **test** set

Why even bother with this?

- There are many machine learning algorithms and it is important to understand which one is more applicable to a particular task
- Even within the same model, there can be many (hyper)parameters to choose from

How to choose the best model

Naive approach

Train models with different parameters and choose the best one on the test

How to choose the best model

Naive approach

Train models with different parameters and choose the best one on the test

Disadvantages of the naive approach

- Since the test usually consists of a random subset of the original sample, the result on the test is also some approximation of a random variable

How to choose the best model

Naive approach

Train models with different parameters and choose the best one on the test

Disadvantages of the naive approach

- Since the test usually consists of a random subset of the original sample, the result on the test is also some approximation of a random variable
- If all models are tested on a test dataset and thus choose the best one, then implicit training will occur on the test, and surprises are possible on another independent test

How to choose the best model

Naive approach

Train models with different parameters and choose the best one on the test

Disadvantages of the naive approach

- Since the test usually consists of a random subset of the original sample, the result on the test is also some approximation of a random variable
- If all models are tested on a test dataset and thus choose the best one, then implicit training will occur on the test, and surprises are possible on another independent test

So... what to do?

In order not to implicitly learn from test data — you need to use **cross-validation**

Cross Validation

General idea

The main idea of cross-validation is to split the training set into two non-overlapping sets (possibly multiple times):

$$X^{learn} = X^{train} \sqcup X^{val}$$

On one of them, training takes place, and on the other, the model is validated.

Cross Validation

General idea

The main idea of cross-validation is to split the training set into two non-overlapping sets (possibly multiple times):

$$X^{learn} = X^{train} \sqcup X^{val}$$

On one of them, training takes place, and on the other, the model is validated.

Why validate?

Usually, any machine learning algorithm contains a whole set of so-called "**hyperparameters**" (i.e. parameters that are not learned, but set initially): dimension, various weighting factors, etc.
And in order to select these parameters "fairly", without using any test data at all, a validation procedure is carried out.

Cross Validation

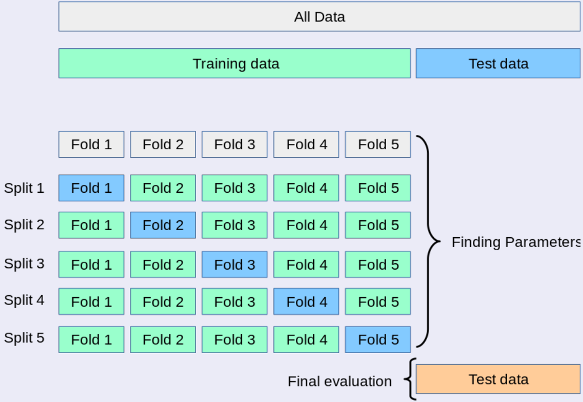
Special cases

1. The simplest cross-validation is **hold-out** control, in which the set is split once:

Cross Validation

Special cases

2. k block control (**k-fold** validation)[1]:



[1] Image source: `https://scikit-learn.org/`

Cross Validation

Special cases

3. Control by individual objects (**leave-one-out**, or LOO validation) — a special case of k-fold validation, if k is equal to the cardinality of the training set

Cross Validation

Special cases

3. Control by individual objects (**leave-one-out**, or LOO validation) — a special case of k-fold validation, if k is equal to the cardinality of the training set
4. **Multiple k-fold** validation — repeat k-fold validation several times with different splits.

Cross-validation: the main stages of correct training

1. We come up with a model and hyperparameter space

[2] Image source: https://scikit-learn.org/

Cross-validation: the main stages of correct training

1. We come up with a model and hyperparameter space
2. We select a test set from the initial data, and divide the remaining set into training and validation

[2] Image source: `https://scikit-learn.org/`

Cross-validation: the main stages of correct training

1. We come up with a model and hyperparameter space
2. We select a test set from the initial data, and divide the remaining set into training and validation
3. Train the model

[2] Image source: `https://scikit-learn.org/`

Cross-validation: the main stages of correct training

1. We come up with a model and hyperparameter space
2. We select a test set from the initial data, and divide the remaining set into training and validation
3. Train the model
4. We carry out the cross-validation procedure across the space of hyperparameters and find their optimal values

[2] Image source: `https://scikit-learn.org/`

Cross-validation: the main stages of correct training

1. We come up with a model and hyperparameter space
2. We select a test set from the initial data, and divide the remaining set into training and validation
3. Train the model
4. We carry out the cross-validation procedure across the space of hyperparameters and find their optimal values
5. We train on the full training set with selected hyperparameters

[2] Image source: https://scikit-learn.org/

Cross-validation: the main stages of correct training

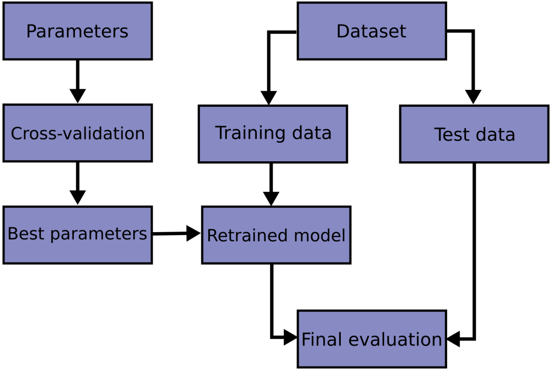
1. We come up with a model and hyperparameter space
2. We select a test set from the initial data, and divide the remaining set into training and validation
3. Train the model
4. We carry out the cross-validation procedure across the space of hyperparameters and find their optimal values
5. We train on the full training set with selected hyperparameters
6. Let's check it on the test!

[2] Image source: `https://scikit-learn.org/`

Cross-validation: the main stages of correct training

1. We come up with a model and hyperparameter space
2. We select a test set from the initial data, and divide the remaining set into training and validation
3. Train the model
4. We carry out the cross-validation procedure across the space of hyperparameters and find their optimal values
5. We train on the full training set with selected hyperparameters
6. Let's check it on the test!

General scheme[2]:



[2] Image source: `https://scikit-learn.org/`

Tuning Hyperparameters

- So far, we can split the examples set into training and testing subsets

Tuning Hyperparameters

- So far, we can split the examples set into training and testing subsets
- We also know how to subdivide the training subset for the validation procedure and even measure the quality during this procedure

Tuning Hyperparameters

- So far, we can split the examples set into training and testing subsets
- We also know how to subdivide the training subset for the validation procedure and even measure the quality during this procedure
- But the main goal is to select the optimal set of hyperparameters!

Tuning Hyperparameters

- So far, we can split the examples set into training and testing subsets
- We also know how to subdivide the training subset for the validation procedure and even measure the quality during this procedure
- But the main goal is to select the optimal set of hyperparameters!
- Usually two approaches are used:
  - Grid Search: to traverse a predefined range of hyperparameters

Tuning Hyperparameters

- So far, we can split the examples set into training and testing subsets
- We also know how to subdivide the training subset for the validation procedure and even measure the quality during this procedure
- But the main goal is to select the optimal set of hyperparameters!
- Usually two approaches are used:
  - Grid Search: to traverse a predefined range of hyperparameters
  - Randomized Search: to generate hyperparameters randomly (according to their given distributions)

Tuning Hyperparameters

- So far, we can split the examples set into training and testing subsets
- We also know how to subdivide the training subset for the validation procedure and even measure the quality during this procedure
- But the main goal is to select the optimal set of hyperparameters!
- Usually two approaches are used:
  - Grid Search: to traverse a predefined range of hyperparameters
  - Randomized Search: to generate hyperparameters randomly (according to their given distributions)
  - Usually there is not much difference — and if you do not need to check **specific** values of hyperparameters in advance, then it is better to limit yourself to a random search

Takeaway notes

1. While doing Supervised Learning, need to define the input and output of the model, and what is the optimization criteria for training

Takeaway notes

1. While doing Supervised Learning, need to define the input and output of the model, and what is the optimization criteria for training
2. It is necessary to divide the available data into training, validation and test sets — from the very beginning

Takeaway notes

1. While doing Supervised Learning, need to define the input and output of the model, and what is the optimization criteria for training
2. It is necessary to divide the available data into training, validation and test sets — from the very beginning
3. Cross-validation can be of the very different types, but the main goal is the same: to test the generalization ability of the ML model (generalization means performance on an independent data set)

Takeaway notes

1. While doing Supervised Learning, need to define the input and output of the model, and what is the optimization criteria for training
2. It is necessary to divide the available data into training, validation and test sets — from the very beginning
3. Cross-validation can be of the very different types, but the main goal is the same: to test the generalization ability of the ML model (generalization means performance on an independent data set)
4. Hyperparameters tuning is needed for almost every ML model

Thank you!