Item 237
git_comments:

1. * * Construct instance of this class. * * @param aggregatingTrainer Aggregator trainer. * @param aggregatingInputMerger Function used to merge submodels outputs into one.
2. * {@inheritDoc}
3. * * Keep original features using {@link IgniteFunction#identity()} as submodelInput2AggregatingInputConverter. * * @return This object.
4. **comment:** TODO: IGNITE-10441 -- Look for options to avoid boilerplate overrides.
   label: code-design
5. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
6. * * Construct instance of this class. * * @param aggregatingTrainer Aggregator trainer. * @param aggregatingInputMerger Function used to merge submodels outputs into one. * @param submodelInput2AggregatingInputConverter Function used to convert input of submodel to output of submodel * this function is used if user chooses to keep original features.
7. * * {@link DatasetTrainer} with same type of input and output of submodels. * * @param <I> Type of submodels input. * @param <O> Type of aggregator model output. * @param <AM> Type of aggregator model. * @param <L> Type of labels.
8. * * Constructs instance of this class.
9. * Aggregating trainer.
10. * * Specify binary operator used to merge submodels outputs to one. * * @param merger Binary operator used to merge submodels outputs to one. * @return This object.
11. Make sure there is at least one way for submodel input to propagate to aggregator.
12. * Operator that merges inputs for aggregating model.
13. * * Constructs instance of this class. * * @param aggregatorTrainer Trainer of model used for aggregation of results of submodels. * @param aggregatingInputMerger Binary operator used to merge outputs of submodels into one output passed to * aggregator model. * @param submodelInput2AggregatingInputConverter Function used to convert input of submodel to output of submodel * this function is used if user chooses to keep original features.
14. * * Get feature extractor which will be used for aggregator trainer from original feature extractor. * This method is static to make sure that we will not grab context of instance in serialization. * * @param featureExtractor Original feature extractor. * @param subMdls Submodels. * @param <K> Type of upstream keys. * @param <V> Type of upstream values. * @return Feature extractor which will be used for aggregator trainer from original feature extractor.
15. * {@inheritDoc}
16. * * Adds submodel trainer along with converters needed on training and inference stages. * * @param trainer Submodel trainer. * @return This object.
17. * * Set function used for conversion of {@link Vector} to submodel input. This function is used during * building of dataset for training aggregator model. This dataset is augmented with results of submodels * applied to {@link Vector}s in original dataset. * * @param vector2SubmodelInputConverter Function used for conversion of {@link Vector} to submodel input. * @return This object.
18. * Function used for conversion of submodel output to {@link Vector}.
19. * * {@link DatasetTrainer} encapsulating stacking technique for model training. * Model produced by this trainer consists of two layers. First layer is a model {@code IS -> IA}. * This layer is a "parallel" composition of several "submodels", each of them itself is a model * {@code IS -> IA} with their outputs {@code [IA]} merged into single {@code IA}. * Second layer is an aggregator model {@code IA -> O}. * Training corresponds to this layered structure in the following way: * <pre> * 1. train models of first layer; * 2. train aggregator model on dataset augmented with outputs of first layer models converted to vectors. * </pre> * During second step we can choose if we want to keep original features along with converted outputs of first layer * models or use only converted results of first layer models. This choice will also affect inference. * This class is a most general stacked trainer, there is a {@link

StackedVectorDatasetTrainer}: a shortcut version of * it with some types and functions specified. * * @param <IS> Type of submodels input. * @param <IA> Type of aggregator input. * @param <O> Type of aggregator output. * @param <L> Type of labels.
20. * Function transforming input for submodels to input for aggregating model.
21. * Function used for conversion of {@link Vector} to submodel input.
22. * * Create instance of this class. * * @param aggregatorTrainer Trainer of model used for aggregation of results of submodels. * @param aggregatingInputMerger Binary operator used to merge outputs of submodels into one output passed to * aggregator model. * @param submodelInput2AggregatingInputConverter Function used to convert input of submodel to output of submodel * this function is used if user chooses to keep original features. * @param submodelsTrainers List of submodel trainers.
23. * * <pre> * 1. Obtain models produced by running specified tasks; * 2. run other specified task on dataset augmented with results of models from step 2. * </pre> * * @param taskSupplier Function used to generate tasks for first step. * @param aggregatorProcessor Function used * @param featureExtractor Feature extractor. * @param <K> Type of keys in upstream. * @param <V> Type of values in upstream. * @return {@link StackedModel}.
24. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
25. Add new columns consisting in submodels output in features.
26. * * Keep original features during training and propagate submodels input to aggregator during inference * using given function. * Note that if this object is on, training will be done on vector obtaining from * concatenating features passed to submodels trainers and outputs of submodels converted to vectors, this can, * for example influence aggregator model input vector dimension (if {@code IS = Vector}), or, more generally, * some {@code IS} parameters which are not reflected just by its type. So converter should be * written accordingly. * * @param submodelInput2AggregatingInputConverter Function used to propagate submodels input to aggregator. * @return This object.
27. **comment:** This method is never called, we override "update" instead.
    label: requirement
28. * * Drop original features during training and inference. * * @return This object.
29. * * Specify aggregator trainer. * * @param aggregatorTrainer Aggregator trainer. * @return This object.
30. * Trainers of submodels with converters from and to {@link Vector}.
31. Unsafely coerce DatasetTrainer<M1, L> to DatasetTrainer<Model<IS, IA>, L>, but we fully control usages of this unsafely coerced object, on the other hand this makes work with submodelTrainers easier.
32. * * Apply submodel to {@link Vector}. * * @param mdl Submodel. * @param submodelOutput2VectorConverter Function for conversion of submodel output to {@link Vector}. * @param vector2SubmodelInputConverter Function used for conversion of {@link Vector} to submodel input. * @param v Vector. * @param <IS> Type of submodel input. * @param <IA> Type of submodel output. * @return Result of application of {@code submodelOutput2VectorConverter . mdl . vector2SubmodelInputConverter} * where dot denotes functions composition.
33. * * Set function used for conversion of submodel output to {@link Vector}. This function is used during * building of dataset for training aggregator model. This dataset is augmented with results of submodels * converted to {@link Vector}. * * @param submodelOutput2VectorConverter Function used for conversion of submodel output to {@link Vector}. * @return This object.
34. * * Constructs instance of this class.
35. * Binary operator merging submodels outputs.
36. * {@inheritDoc}
37. * Submodels layer.
38. * * Get submodels constituting first layer of this model. * * @return Submodels constituting first layer of this model.
39. * Models constituting submodels layer.
40. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the

"License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
41. * * Model consisting of two layers: * <pre> * 1. Submodels layer {@code (IS -> IA)}. * 2. Aggregator layer {@code (IA -> O)}. * </pre> * Submodels layer is a "parallel" composition of several models {@code IS -> IA} each of them getting same input * {@code IS} and produce own output, these outputs outputs {@code [IA]} * are combined into a single output with a given binary "merger" operator {@code IA -> IA -> IA}. Result of merge * is then passed to the aggregator layer. * Aggregator layer consists of a model {@code IA -> O}. * * @param <IS> Type of submodels input. * @param <IA> Type of submodels output (same as aggregator model input). * @param <O> Type of aggregator model output. * @param <AM> Type of aggregator model.
42. * * Get aggregator model. * * @return Aggregator model.
43. * Aggregator model.
44. * * Add submodel into first layer. * * @param subMdl Submodel to add.
45. * * Constructs instance of this class. * * @param aggregatorMdl Aggregator model. * @param aggregatingInputMerger Binary operator used to merge submodels outputs. * @param subMdlInput2AggregatingInput Function converting submodels input to aggregator input. (This function * is needed when in {@link StackedDatasetTrainer} option to keep original features is chosen).
46. * {@inheritDoc}
47. * * Shortcut for adding trainer {@code Matrix -> Matrix} where this trainer is treated as {@code Vector -> Vector}, where * input {@link Vector} is turned into {@code 1 x cols} {@link Matrix} and output is a first row of output {@link Matrix}. * * @param trainer Submodel trainer. * @param <M1> Type of submodel trainer model. * @return This object.
48. * * Shortcut for adding trainer {@code Vector -> Double} where this trainer is treated as {@code Vector -> Vector}, where * output {@link Vector} is constructed by wrapping double value. * * @param trainer Submodel trainer. * @param <M1> Type of submodel trainer model. * @return This object.
49. **comment:** TODO: IGNITE-10441 -- Look for options to avoid boilerplate overrides.
    label: code-design
50. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
51. * * {@link StackedDatasetTrainer} with {@link Vector} as submodels input and output. * * @param <O> Type of aggregator model output. * @param <L> Type of labels. * @param <AM> Type of aggregator model.
52. * * Constructs instance of this class. * * @param aggregatingTrainer Aggregator trainer.
53. * * Constructs instance of this class.
54. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
55. * * <!-- Package description. --> * Contains classes used for training with stacking technique.
56. * * Construct instance of this class. * * @param before Function applied before wrapped model. * @param mdl Inner model. * @param after Function applied after wrapped model.
57. * {@inheritDoc}
58. * * Model which is composition of form {@code before `andThen` inner Mdl `andThen` after}. * * @param <I> Type of input of this model. * @param <O> Type of output of this model. * @param <IW> Type of input of inner model. * @param <OW> Type of output of inner model. * @param <M> Type of inner model.

59. * * Create new instance of this class with changed inner model. * * @param mdl Inner model. * @param <M1> Type of inner model. * @return New instance of this class with changed inner model.
60. * Function applied before inner model.
61. * Function applied after inner model.
62. * * Get inner model. * * @return Inner model.
63. * * Result of this model application is a result of composition {@code before `andThen` inner mdl `andThen` after}.
64. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
65. * * Create new {@code AdaptableDatasetModel} which is a composition of the form {@code thisMdl . before}. * * @param before Function applied before this model. * @param <I1> Type of function applied before this model. * @return New {@code AdaptableDatasetModel} which is a composition of the form {@code thisMdl . before}.
66. * Inner model.
67. * Wrapped trainer.
68. * * Let this trainer produce model {@code mdl}. This method produces a trainer which produces {@code mdl1}, where * {@code mdl1 = f `andThen` mdl}. * * @param before Function inserted before produced model. * @param <I1> Type of produced model input. * @return New {@link DatasetTrainer} which produces composition of specified function and model produced by * original trainer.
69. * Function used to convert input type of wrapped trainer.
70. * {@inheritDoc}
71. * * Let this trainer produce model {@code mdl}. This method produces a trainer which produces {@code mdl1}, where * {@code mdl1 = mdl `andThen` after}. * * @param after Function inserted before produced model. * @param <O1> Type of produced model output. * @return New {@link DatasetTrainer} which produces composition of specified function and model produced by * original trainer.
72. * Function used to convert output type of wrapped trainer.
73. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
74. * * Construct instance of this class with specified wrapped trainer and converter functions. * * @param before Function used to convert input type of wrapped trainer. * @param wrapped Wrapped trainer. * @param after Function used to convert output type of wrapped trainer.
75. * * Type used to adapt input and output types of wrapped {@link DatasetTrainer}. * Produces model which is composition of form {@code before `andThen` wMdl `andThen` after} where wMdl is model produced by * wrapped trainer. * * @param <I> Input type of model produced by this trainer. * @param <O> Output type of model produced by this trainer. * @param <IW> Input type of model produced by wrapped trainer. * @param <OW> Output type of model produced by wrapped trainer. * @param <M> Type of model produced by wrapped model. * @param <L> Type of labels.
76. * * Construct instance of this class from a given {@link DatasetTrainer}. * * @param wrapped Wrapped trainer. * @param <I> Input type of wrapped trainer. * @param <O> Output type of wrapped trainer. * @param <M> Type of model produced by wrapped trainer. * @param <L> Type of labels. * @return Instance of this class.
77. * * Tests simple stack training.
78. Convert model trainer to produce Vector -> Vector model
79. * * Tests stacked trainers.
80. * Rule to check exceptions.

81. * * Tests that if there is no any way for input of first layer to propagate to second layer, * exception will be thrown.
82. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
83. * * Get a composition model of the form {@code x -> after(mdl(x))}. * * @param after Function to apply after this model. * @param <V1> Type of input of function applied before this model. * @return Composition model of the form {@code x -> after(mdl(x))}.
84. * * Identity function. * * @param <T> Type of input and output. * @return Identity function.
85. **comment:** TODO: IGNITE-10653 Maybe we should add toString description to identity and constant.
    label: documentation
86. * * Wrap specified value into vector. * * @param val Value to wrap. * @return Specified value wrapped into vector.
87. * * Turn number to 1-sized array. * * @param val Value to wrap in array. * @return Number wrapped in 1-sized array.
88. * * Concatenates given vectors. * * @param v1 First vector. * @param vs Other vectors. * @return Concatenation result.
89. * * Concatenates two given vectors. * * @param v1 First vector. * @param v2 Second vector. * @return Concatenation result.
90. * * Concatenates given vectors. * * @param vs Other vectors. * @return Concatenation result.
91.
92. * * Creates {@code DatasetTrainer} with same training logic, but able to accept labels of given new type * of labels. * * @param new2Old Converter of new labels to old labels. * @param <L1> New labels type. * @return {@code DatasetTrainer} with same training logic, but able to accept labels of given new type * of labels.
93. * {@inheritDoc}
94. * * Returns trainer which independently of dataset outputs given model. * * @param ml Model. * @param <I> Type of model input. * @param <O> Type of model output. * @param <M> Type of model. * @param <L> Type of dataset labels. * @return Trainer which independently of dataset outputs given model.
95. * {@inheritDoc}
96. * xor truth table.
97. * * Create cache mock. * * @param vals Values for cache mock. * @return Cache mock.

git_commits:

1. **summary:** IGNITE-10480: [ML] Stacking for training and inference
   message: IGNITE-10480: [ML] Stacking for training and inference This closes #5635

github_issues:

github_issues_comments:

github_pulls:

1. **title:** IGNITE-10480: [ML] Stacking for training and inference
   body:

github_pulls_comments:

github_pulls_reviews:

1. "No usages found in All Places"
2. **body:** In my opinion such interface is not convenient. Combiner requires working with Monoids like List[Double], but use may expect working with Doubles.

label: code-design
3. javadoc
4. it looks good but we don't use toString for functions)
5. -> addModelTrainerWithDoubleOutput
6. s/withAdded/add/g
7. Agree, done.
8. Agree, done.
9. **body:** Seems like this class in unnecessary, removed it completely.
   label: code-design
10. Yeah, got StackedVectorDatasetTrainer#addTrainerWithDoubleOutput for this which essentially lifts `Double` to `Vector` monoid (with concatenation as `mappend`).
11. you provide composition interface through andThen-function "f1 andThen f2" works as "f2(f1(x))" but "f1 compose f2" as "f1(f2())" In my opinion you should provide uniform semantic (preferably "andThen") in all comments to avoid problems of understanding and using of such interface
12. javadoc
13. javadoc
14. I think "the most general stacked trainer" will be better
15. it is just wrapper without additional logic what is the reason of using it?
16. **body:** maybe we should rewrite updateModel function from abstract form to form like this to avoid such code: class DatasetTrainer { .... protected ... updateModel(...) { throw NotImplementedException() } .... } ?
    label: code-design
17. May you merge update and fit logic?
18. **body:** please avoid functional composition notation in Java world) in this wild world there are a lot of non-functional programmers may to beat you))
    label: code-design
19. Agree, done.
20. Fixed.
21. Fixed.
22. Agree, done.
23. Ok, since we have `andThen`, I'll use it. ~~P.S. I will camouflage in a State monad to look imperativish~~
24. **body:** Yes, intention was to make debugging less painful with at least some of the lambdas listed with meaningful names, but maybe we should make a ticket for that.
    label: code-design
25. Okay, let's add it on demand.
26. **body:** Reason is to convert everything into `DatasetTrainer<Model<IS, IA>, L>` (in contrast to `DatasetTrainer<? extends Model<IS, IA>, L>`) to make work with the list of `submodelTrainers` less painful. This is unsafe conversion, but since we have control of all `sumbodelTrainers` list usages inside our class, it's IMHO a reasanoble tradeoff.
    label: code-design
27. **body:** Hmm... We will avoid boilerplate, but seems like it will make code more error prone. Developer can forget to override this method for trainer which potentially supports updating and get an error while trying to update this model in the future whereas keeping it abstract forces developer to think if this trainer supports update and insert `NotImplementedException` more cautiously.
    label: code-design
28. Agree, done.
29. to (avoid numbers)
30. the same thing
31. ?
32. Do we really need this?
33. to
34. to
35. new20 old? newToOld or something better
36. withTrainer
37. StackedModel getting looks pretty, many thanks!
38. ConvertedLabels seems like reinvention of labelExtractor sometimes
39. **body:** Maybe we need a preprocessor like LabelTransformer with map function to change them
    label: code-design
40. This is shortcut "specially" for MLP :)

41. **body:** The reasons behind "2" are 1. I saw in code some methods with such naming, and decided to be consistent 2. "2" Is a good visual separator (at least for me) between "from" and "to" parts which (again, at least for me) rises readability,
    label: code-design
42. See https://github.com/apache/ignite/pull/5635#discussion_r241412240
43. This is a shortcut specially for people who prefer to pass all arguments via `with`s. And the one with aggregator trainer as a single param is for IDE automatic type inference when introducing new variable.
44. Thanks! As for `withTrainer`, it was a tough choice: `withTrainer` looks like we substitute new trainer in place of the old trainer, not adding it. Previously it was `withAddedTrainer`, which looked a bit clumsy, so I went with `addTrainer`.
45. This is for the case when we use one `DatasetBuilder` for many trainers like we do in `StackedDatasetTrainer`. In this case we want the ability to adapt each of the trainers to be able to work with this dataset, this method is just for that.
46. If you don't mind, I wouldn't change it now because otherwise for consistency I should also change other such methods. I think it should be done atomically after some discussion.
47. See https://github.com/apache/ignite/pull/5635#discussion_r241417139

jira_issues:

1. **summary:** [ML] Stacking for training and inference
   description: Stacking is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features.

jira_issues_comments:

1. GitHub user artemmalykh opened a pull request: https://github.com/apache/ignite/pull/5635 IGNITE-10480: [ML] Stacking for training and inference You can merge this pull request into a Git repository by running: $ git pull https://github.com/gridgain/apache-ignite ignite-10480 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/ignite/pull/5635.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #5635 ---- ----
2. reviewed by [~aplatonov], [~zaleslaw] and [~chief] merged
3. Github user asfgit closed the pull request at: https://github.com/apache/ignite/pull/5635