Distributed Machine Learning

ML Libraries. Ensemble Methods: Bagging, Boosting, Stacking.

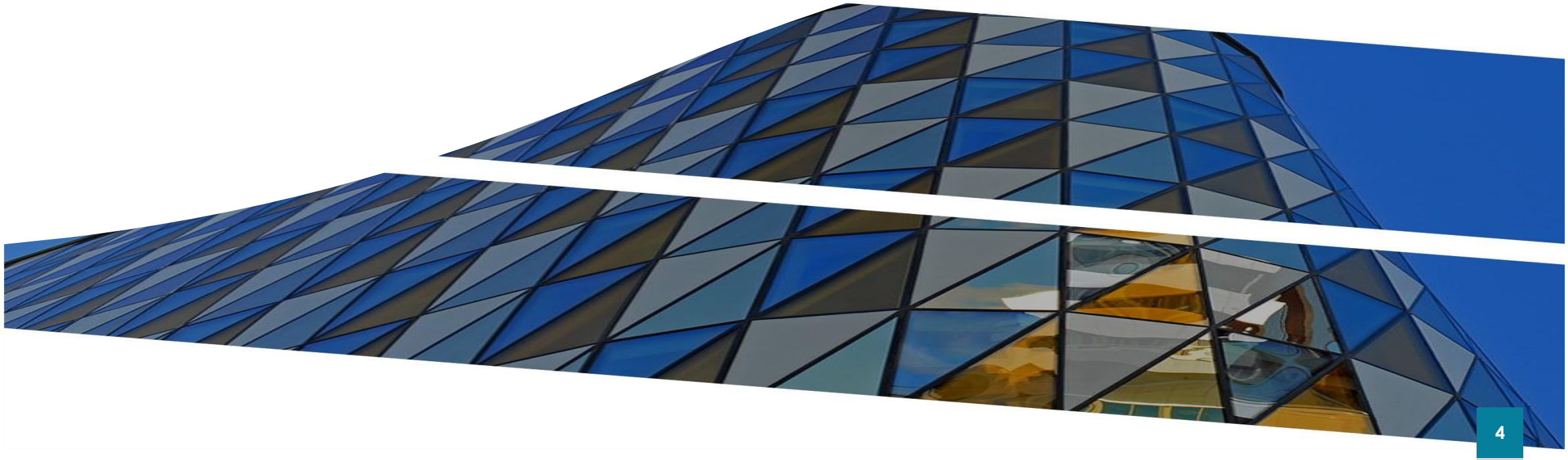
Machine Learning + Big Data in Real Time + Cloud Technologies

=> The Future of Intelligent Systems

Where to Find The Code and Materials?

https://github.com/iproduct/course-ml

Machine Learning Libraries



Top Machine Learning Libraries - I

- Apache Spark Mllib – fast, in-memory, big-data processing, which has also made it a go-to framework for ML, highly scalable, can be coupled with any Hadoop data source, supports Java, Scala, R and Python, new ML algorithms are constantly being added, and existing enhanced.

- H_20 + AutoML - open source, in-memory, distributed, fast, and scalable machine learning and predictive analytics platform allowing to learn from big data, written in Java, uses Distributed Key/Value store for accessing data, models, objects, etc. across all nodes and machines, uses distributed Map/Reduce, utilizes Java Fork/Join multi-threading, data read in parallel, distributed across the cluster, and stored in memory in a columnar format in a compressed way, data parser has built-in intelligence to guess the schema of the incoming dataset and supports data ingest from multiple sources in various formats, APIs: REST, Flow UI, H2O-R, H2O-Python, supports Deep Learning, Tree Ensembles, and Generalized Low Rank Models (GLRM).

Top Machine Learning Libraries - II

- Microsoft Azure ML Studio - Microsoft's ML framework runs in their Azure cloud, offering high-capacity data processing on a pay-as-you-go model. Its interactive, visual workspace can be used to create, test and iterate ML "experiments" using the built-in ML packages; they can then be published and shared as web services. Python or R custom coding is also supported. A free trial is available for new users.

- Amazon Machine Learning - like Azure, the Amazon Machine Learning framework is cloud-based, supporting Amazon S3, Redshift and RDS. It combines ML algorithms with interactive tools and data visualizations which allow users to easily create, evaluate and deploy ML models. These models can be binary, categoric or numeric – making them useful in areas such as information filtering and predictive analytics.

Top Machine Learning Libraries - III

- Microsoft Distributed Machine Learning Toolkit (DMTK) - uses local data caching to make it more scalable and efficient on computing clusters that have limited resources on each node. Its distributed ML algorithms allow for fast training of gradient boosting, topic and word-embedding learning models:
  - LightLDA: Scalable, fast and lightweight system for large-scale topic modeling.
  - LightGBM: LightGBM is a fast, distributed, high performance gradient boosting (GBDT, GBRT, GBM or MART) framework based on decision tree algorithms, used for ranking, classification and many other machine learning tasks.
  - Distributed word embedding: a parallelization of the Word2Vec algorithm
  - Distributed Multi-sense Word Embedding (DMWE) - a parallelization of the Skip-Gram Mixture algorithm used for polysemous words.
- Google TensorFlow - open-source ML toolkit that has been applied in areas ranging from machine translation to biomedical science. Data flows are processed by a series of algorithms described by a graph. This allows for a flexible, multi-node architecture that supports multiple CPUs and GPUs. The recently released Tensorflow Lite adds support for neural processing on mobile phones.

Top Machine Learning Libraries - IV

- Caffe - developed by Berkeley AI Research, Caffe claims to offer one of the fastest implementations of convolutional neural networks. It was originally developed for machine vision projects but has since expanded to other applications, with its extensible C++ code designed to foster active development. Both CPU and GPU processing are supported.

- Veles - developed and released as open source by Samsung, Veles is a distributed ML platform designed for rapid deep-learning application development. Users can train various artificial neural net types – including fully connected, convolutional and recurrent. It can be integrated with any Java application, and its "snapshot" feature improves disaster recovery.

Top Machine Learning Libraries - V

- Massive Online Analysis (MOA) - developed at the University of Waikato in New Zealand, this open-source Java framework specializes in real-time mining of streaming data and large-scale ML. It offers a wide range of ML algorithms, including classification, regression, clustering, outlier detection and concept drift detection. Evaluation tools are also provided.

- Torch - Torch ML library makes ML easy and efficient, thanks to its use of the Lua scripting language and a GPU-first implementation. It comes with a large collection of community-driven ML packages that cover computer vision, signal processing, audio processing, networking and more. Its neural network and optimization libraries are designed to be both accessible and flexible.

Distributed Learning: Ensemble Learning

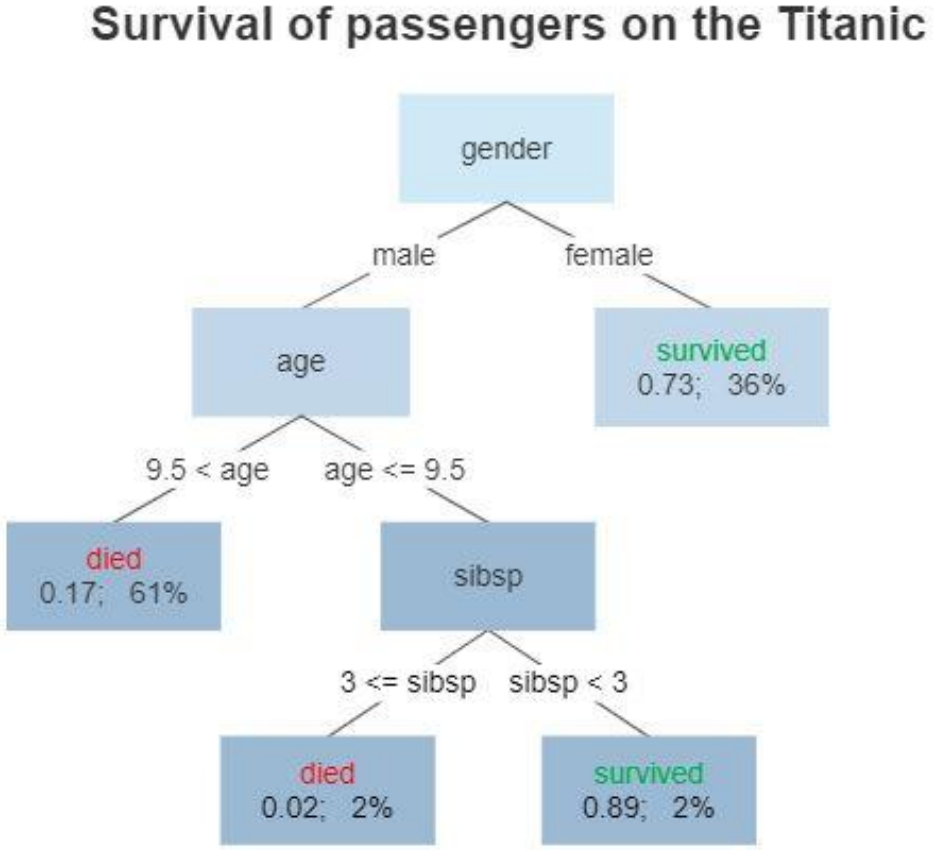
Ensemble Learning

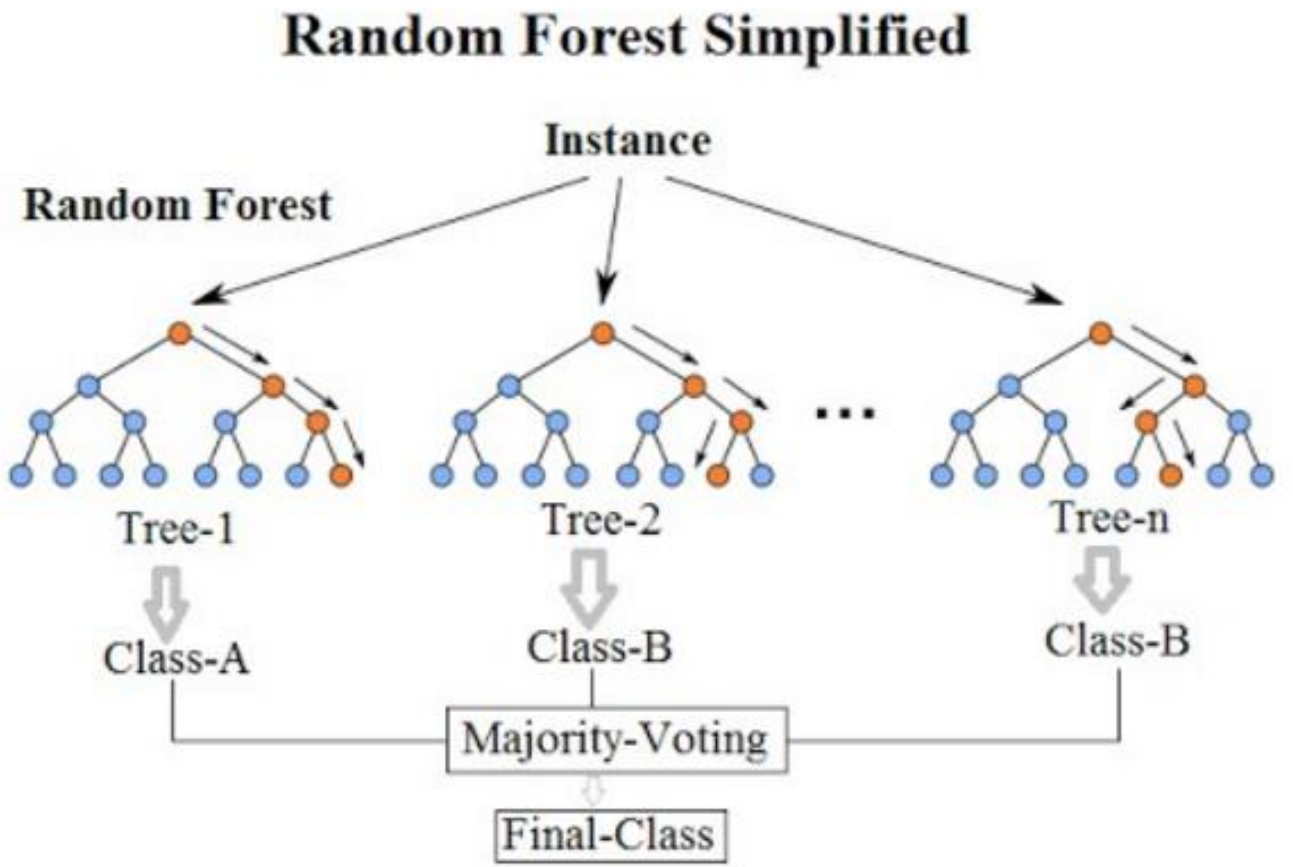
- Ensemble methods – using multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. The term ensemble usually means using the same type of base learners.
- Even if the hypothesis space contains hypotheses that are very well-suited for a particular problem, it may be very difficult to find a good one. Ensembles combine multiple hypotheses to form a better hypothesis. Resultant hypothesis, is not necessarily contained within the hypothesis space of the building models – can be more complex.
- Ensemble over-fits the training data easier than a single model -> Bagging.
- Ensembles yield better results when there is a diversity among the models. Using variety of strong learning algorithms, has been shown to be more effective than using techniques that attempt to dumb-down the models in order to promote diversity.
- number of independent classifiers == number of classes => highest accuracy

Bagging – e.g. Random Forest

- Bagging (Bootstrap AGGregatING) – involves having each model in the ensemble vote with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set.
- Example: Random Forest algorithm combines random decision trees with bagging to achieve very high classification accuracy. Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples + feature bagging:
  1. For b = 1, ..., B:
  2. Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
  3. Train a classification or regression tree fb on Xb, Yb, by selecting a limited number of features (ayttributes) to be used during the training (feature bagging).
  4. After training, predictions for unseen samples x' can be made by averaging the predictions / taking the majority vote from all the individual regression trees on x‘.

Decision Trees. Random Forests



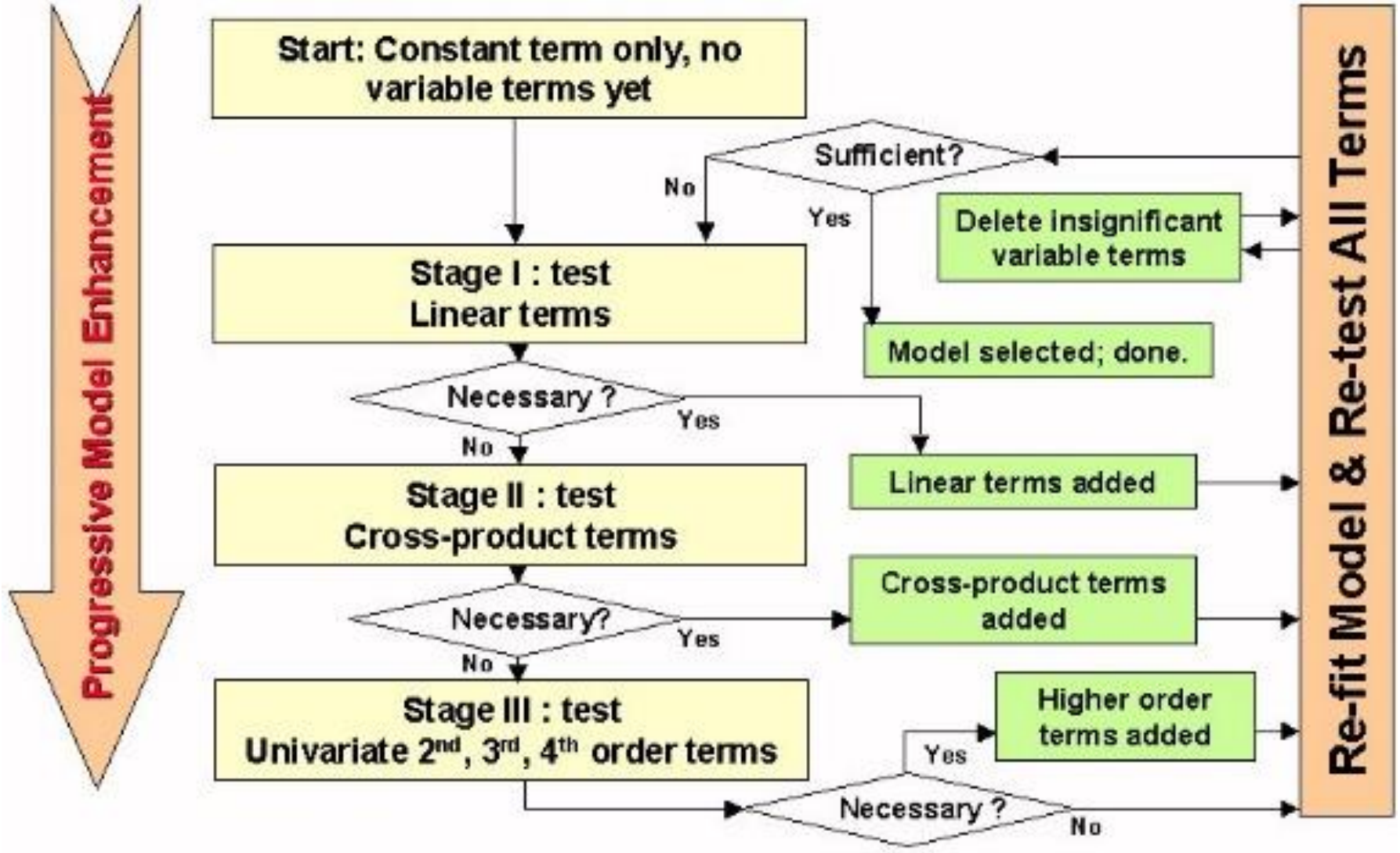


Bayesian Model Averaging. Bayesian Model Combination

- Bayesian Model Averaging (BMA)– makes predictions using an average over several models with weights given by the posterior probability of each model given the data. BMA is known to generally give better answers than a single model, obtained, e.g., via stepwise regression, especially where very different models have nearly identical performance in the training set but may otherwise perform quite differently.

- Bayesian model combination (BMC) - an algorithmic correction to Bayesian model averaging (BMA). Instead of sampling each model in the ensemble individually, it samples from the space of possible ensembles (with model weightings drawn randomly from a Dirichlet distribution having uniform parameters). This modification overcomes the tendency of BMA to converge toward giving all of the weight to a single model. Although BMC is somewhat more computationally expensive than BMA, it tends to yield dramatically better results. The results from BMC have been shown to be better on average (with statistical significance) than BMA, and bagging.

Stepwise Regression



Bucket of Models

- Bucket of Models – ensemble technique in which a model selection algorithm is used to choose the best model for each problem. When tested with only one problem, a bucket of models can produce no better results than the best model in the set, but when evaluated across many problems, it will typically produce much better results, on average, than any model in the set.

- The most common approach used for model-selection is cross-validation selection (sometimes called a "bake-off contest"):

For each model m in the bucket:

    Do c times: (where 'c' is some constant)

        Randomly divide the training dataset into two datasets: A, and B.

        Train m with A

        Test m with B

Select the model that obtains the highest average score (pick the one that works best)

Stacking

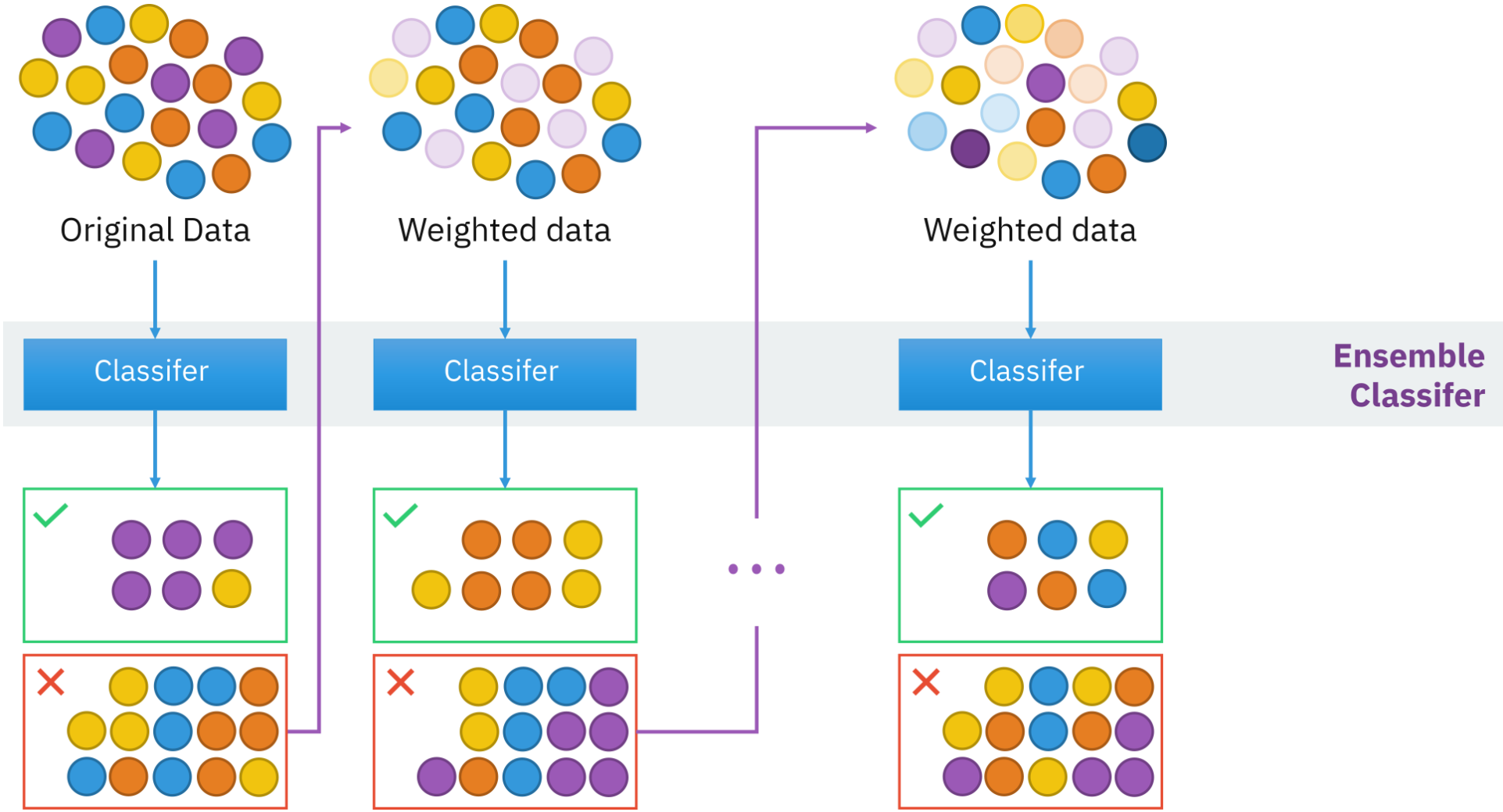
- Stacking (stacked generalization) – involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. If an arbitrary combiner algorithm is used, then stacking can theoretically represent any of the ensemble techniques described in this article, although, in practice, a logistic regression model is often used as the combiner.

- Stacking typically yields performance better than any single one of the trained models. It has been successfully used on both supervised learning tasks (regression, classification and distance learning) and unsupervised learning (density estimation). It has also been used to estimate bagging's error rate. It has been reported to out-perform Bayesian model-averaging. The two top-performers in the Netflix competition utilized blending, which may be considered to be a form of stacking.

Boosting

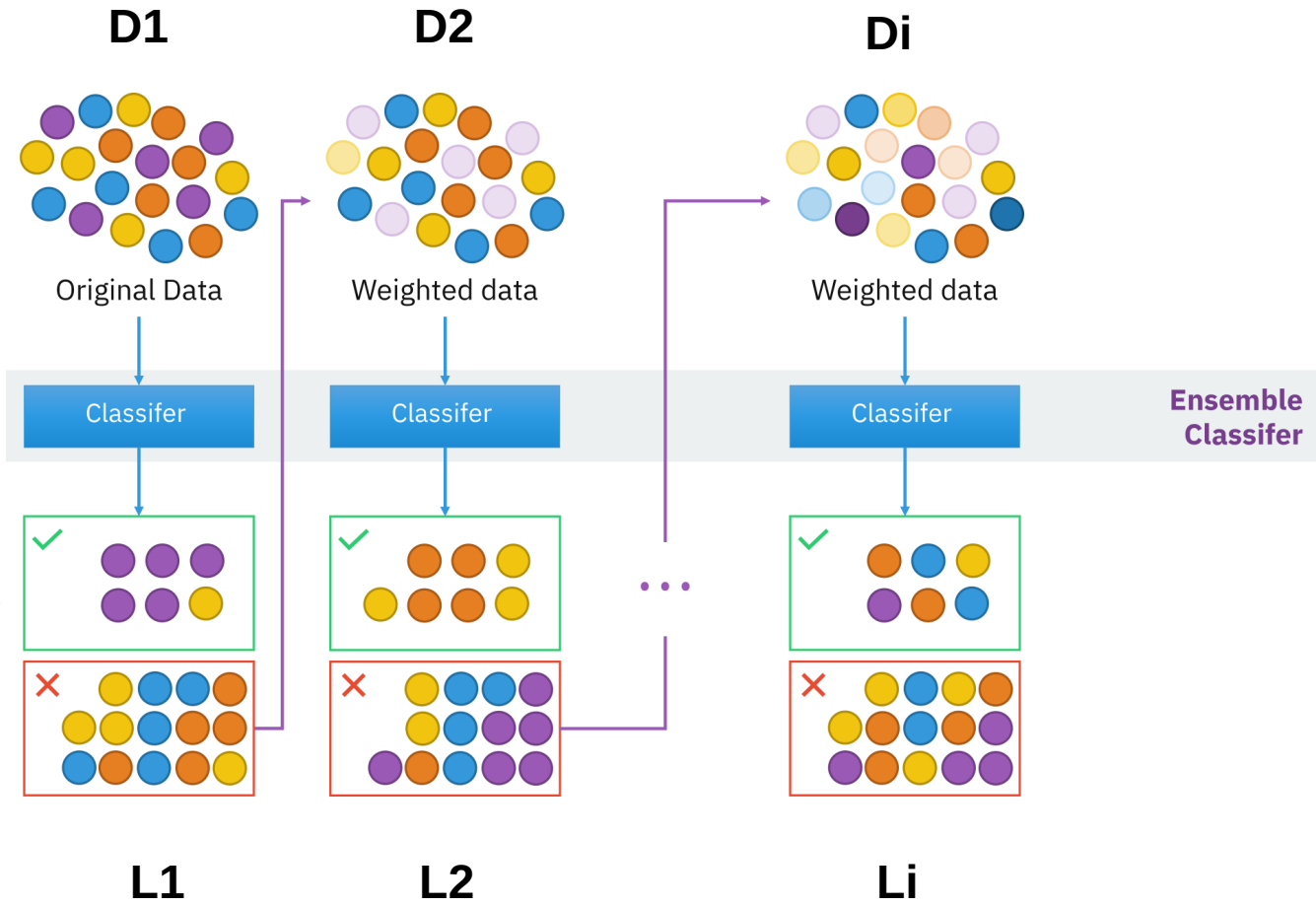
- Boosting - ensemble meta-algorithm for primarily reducing bias, and also variance, based on the question posed by Kearns and Valiant (1988, 1989"Can a set of weak learners create a single strong learner?"
- In supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones
- Weak learner – a classifier that is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.
- Informally, [the hypothesis boosting] problem asks whether an efficient learning algorithm […] that outputs a hypothesis whose performance is only slightly better than random guessing [i.e. a weak learner] implies the existence of an efficient algorithm that outputs a hypothesis of arbitrary accuracy [i.e. a strong learner]."

Boosting - Basic Algorithm II



Boosting – Basic Algorithm

1. Equal weight (uniform probability distribution) is given to the sample training data (say D_1) initially. Let **i = 1**
2. This data (D_i) is then given to a base learner (say L_i).
3. Mis-classified instances by ensemble $L_1, L_2, \ldots L_i$ are assigned a weight higher than the correctly classified instances (keeping the total probability distribution = 1) => D_{i+1} – boosted data
4. Let **i = i + 1**. Go to step 2 until error becomes acceptable (or other finish criteria is met).



AdaBoost

- AdaBoost - the output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to be strong learner.

- Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset. AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

Discrete AdaBoost

With:

- Samples $x_1 \dots x_n$
- Desired outputs $y_1 \dots y_n, y \in \{-1, 1\}$
- Initial weights $w_{1,1} \dots w_{n,1}$ set to $\frac{1}{n}$
- Error function $E(f(x), y, i) = e^{-y_i f(x_i)}$
- Weak learners $h: x \to \{-1, 1\}$

For t in $1 \dots T$:

- Choose $h_t(x)$:
  - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} w_{i,t}$
  - Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$
- Add to ensemble:
  - $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
- Update weights:
  - $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$ for i in $1 \dots n$
  - Renormalize $w_{i,t+1}$ such that $\sum_i w_{i,t+1} = 1$
  - (Note: It can be shown that $\frac{\sum_{h_{t+1}(x_i)=y_i} w_{i,t+1}}{\sum_{h_{t+1}(x_i)\neq y_i} w_{i,t+1}} = \frac{\sum_{h_t(x_i)=y_i} w_{i,t}}{\sum_{h_t(x_i)\neq y_i} w_{i,t}}$ at every step, which can simplify the calculation of the new weights.)

Source: https://en.wikipedia.org/wiki/AdaBoost

Gradient Boosting

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (or weak learner, e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Hastie, T.; Tibshirani, R.; Friedman, J. H. (2009). "10. Boosting and Additive Trees". The Elements of Statistical Learning (2nd ed.). New York: Springer. pp. 337–384. ISBN 978-0-387-84857-0. Archived from the original on 2009-11-10.

Thank's for Your Attention!

Trayan Iliev

IPT – Intellectual Products & Technologies

http://iproduct.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD