Boosting

Boosting

proschool
An ims Initiative

- **What is Boosting ?**
  - Boosting is an ensemble method that uses number of weak classifiers to create a strong classifier.
  - **Weak Classifier:** is defined as one whose performance is at least slightly better than random classifying.
  - First we create first model(Weak Model) by using the training data. The second model is built to reduce the error in classification of the first model and by assigning weights for wrongly classified data and so on.
  - This process will be continued till maximum number of models added or else till we reach perfect predictions.
  - Boosting controls both bias & variance, and is more effective than the bagging which controls only variance.

AdaBoost

AdaBoost

proschool
An ims Initiative

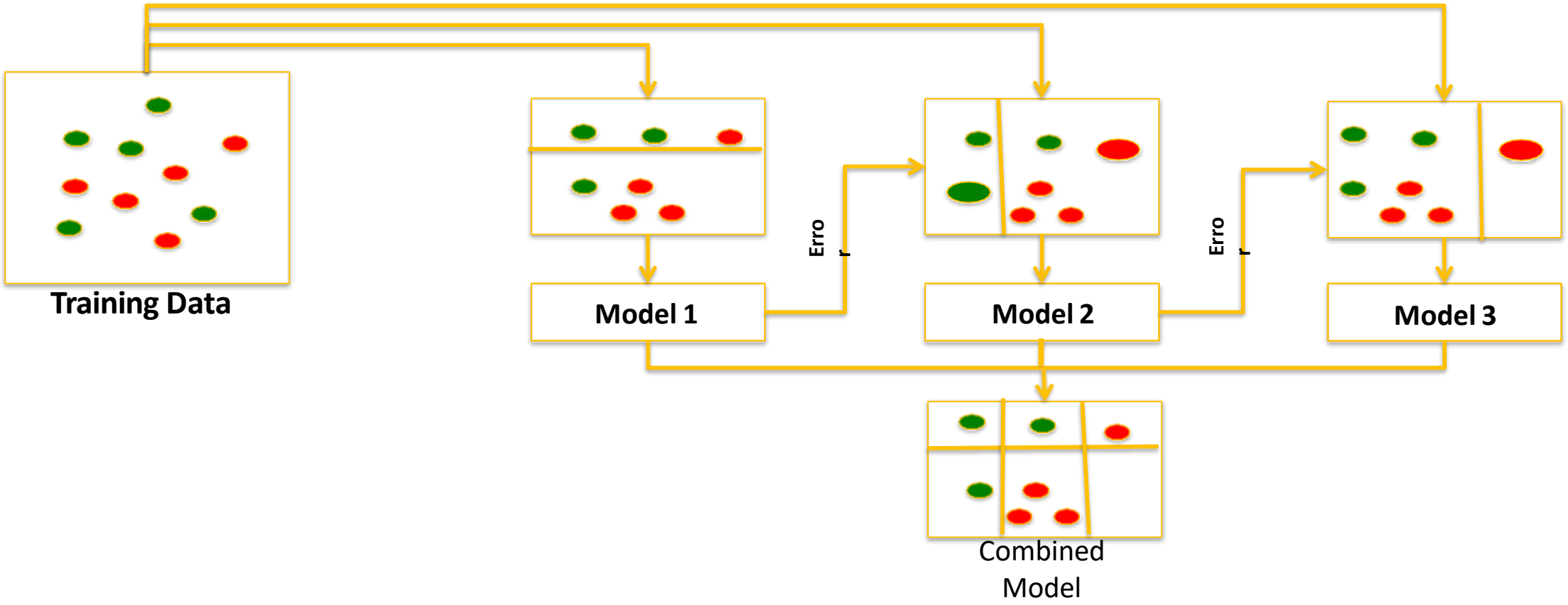
- **Adaptive boosting is a very popular boosting algorithm and it's known as AdaBoost.**
- AdaBoost builds n-number of weak classifiers by using the class labeled data in n-rounds. And Merges the n-week classifiers to get a complex classifiers
- Each Week learner in AdaBoosts are **decision stumps** of a decision tree i.e. decision trees with a single split
- Let us assume D is a data set of class labeled observations given as (X_1, y_1), (X_2, y_2), (X_3, y_3),…, (X_d, y_d), here y is class label of X.
- Training data for each model is taken by the dataset D by sampling with/without replacement.
- AdaBoost initially(For first week model) assigns equal weights for each observation. In this case initial weight is **1/d.**
- After each week classifier is built the miss classified observations are given more weight and the sample data is used to build next classifier.

Examples of AdaBoost

- **Let us consider an example where you are suggesting a good mobile phone to your friend, who wants to purchase a good phone.**
- For simplicity say you are checking the features like:
  - Build Quality
  - Battery Backup
  - Hardware Specification
- You have a data which explains the features of D-number of mobile phones in the current market.
- If you suggest a mobile **only based build quality(First Week Classifier)** there may be chances that you end up purchasing good looking phone with lower battery backup and lower specifications.
- Similarly If you suggest **only on bases of Battery backup(Second Week Classifier)** or **only on the bases of Hardware Specification (Third Week Classifier)** you may end up suggesting bad phone to your friend**.**

Examples of AdaBoost (Cont.)

- To over come this let us merge all the classifiers which gives a better classification between good and bad phones.
- In the below example: Red Bubbles: Bad phone, Green Bubbles- Good Phones



AdaBoost Ensemble

- Weak models are added sequentially, trained using the weighted training data.
- The process continues until a pre-set number of weak learners have been created (a user parameter) or no further improvement can be made on the training dataset.
- Once completed, you are left with a pool of weak learners each with a stage value.

N Gradient Boosting

Gradient Boosting

- Gradient boosting is one of the most powerful techniques for building predictive models.
- Is a powerful machine learning algorithm. It can do,
  - Regression
  - Classification
- **What is the literal meaning of Gradient ?**
  - the degree of inclination, or slope
- **What is the literal meaning of descent ?**
  - an action of moving downward, dropping, or falling
- **What is the literal meaning of boosting?**
  - help or encourage (something) to increase or improve.
- Boosting is an ensemble technique where new models are added to correct the errors made by existing models.
- Models are added sequentially until no further improvements can be made.

Gradient Boosting

- In gradient boosting we use gradient descent for boosting instead of weights.
- Using a special loss function AdaBoost is formulated as gradient descent.
- To find the values of coefficients of a function, which minimizes the cost of that function we use Gradient descent algorithm.
- An example of loss function is the squared error between the actual and predicted value.
  - $L = (Y_i - h(x_i))^2$
- We need to minimize the cost of function $f(x) = \sum_{i=1} L(Y_i, h(x_i))$, where (xi,yi) are coefficients, N is the number of points and $h(x_i)$ prediction of i^{th} week classifier.

GBM Elements

- **A loss function to be optimized** - regression may use a squared error and classification may use logarithmic loss

- **A weak learner to make predictions** - Decision trees are used as the weak learner in gradient boosting.

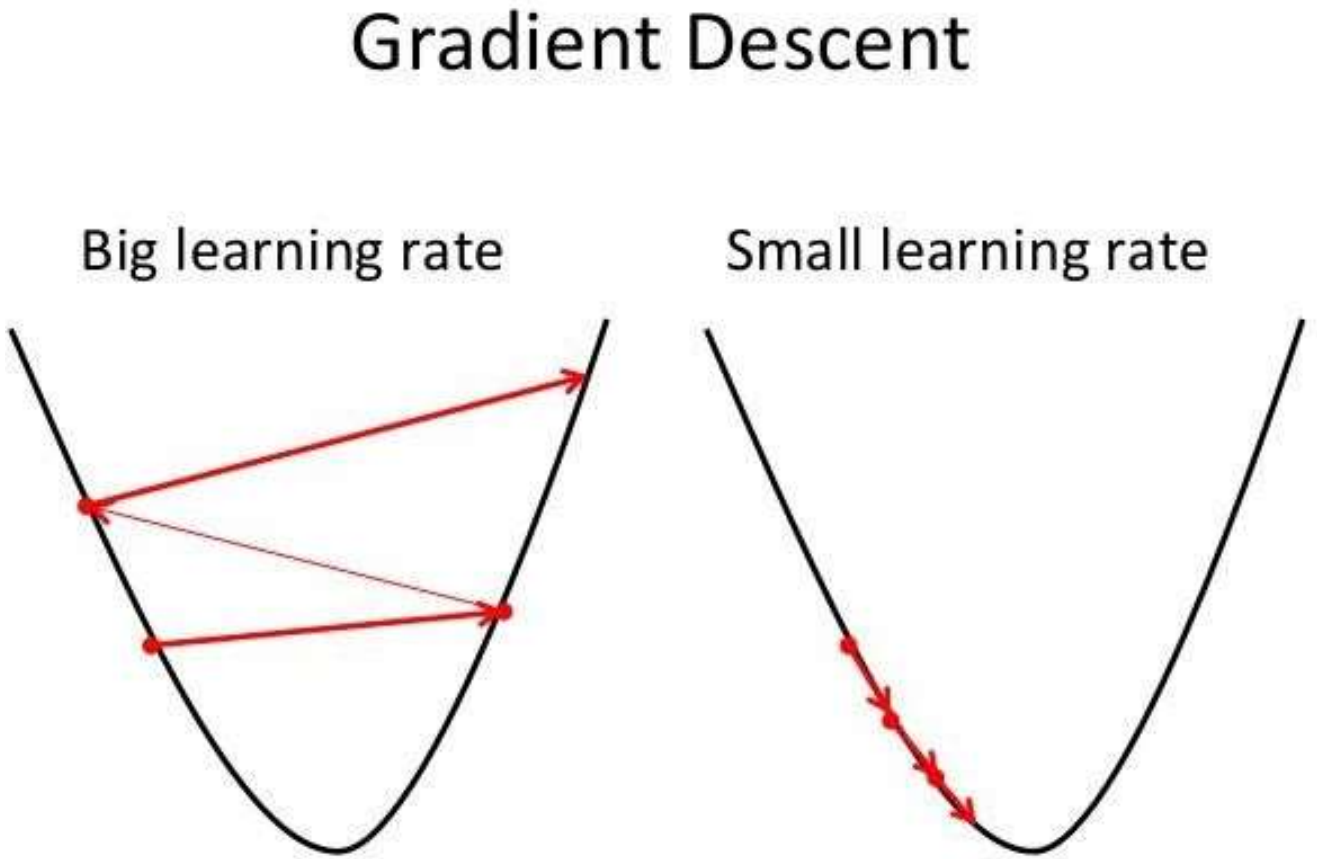
- **An additive model to add weak learners to minimize the loss function** - Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees.

Steps in Gradient Descent

- **Evaluating the Gradient Descent**:
  - Step 1: It starts with assuming a small value for coefficients, here let us assume 0 values for coefficients of the function.
    - Coefficients = 0
  - Step 2: Calculate the cost of the function using the coefficients.
    - Cost = Function (Coefficients)
  - Step 3: Find the slope of the function at a given cost. It can be done by taking the derivative of the cost function. This helps us to find the direction of the change in coefficients.
    - Delta(δ) = d(function(x))/dx
  - Step 4: Evaluate the new coefficients using Learning rate(α = 0.1) and delta values(δ)
    - Coefficients = Coefficients – (α*δ)
  - The process is repeated till we found the coefficients value which gives the cost of the function equal to zero or near to zero.

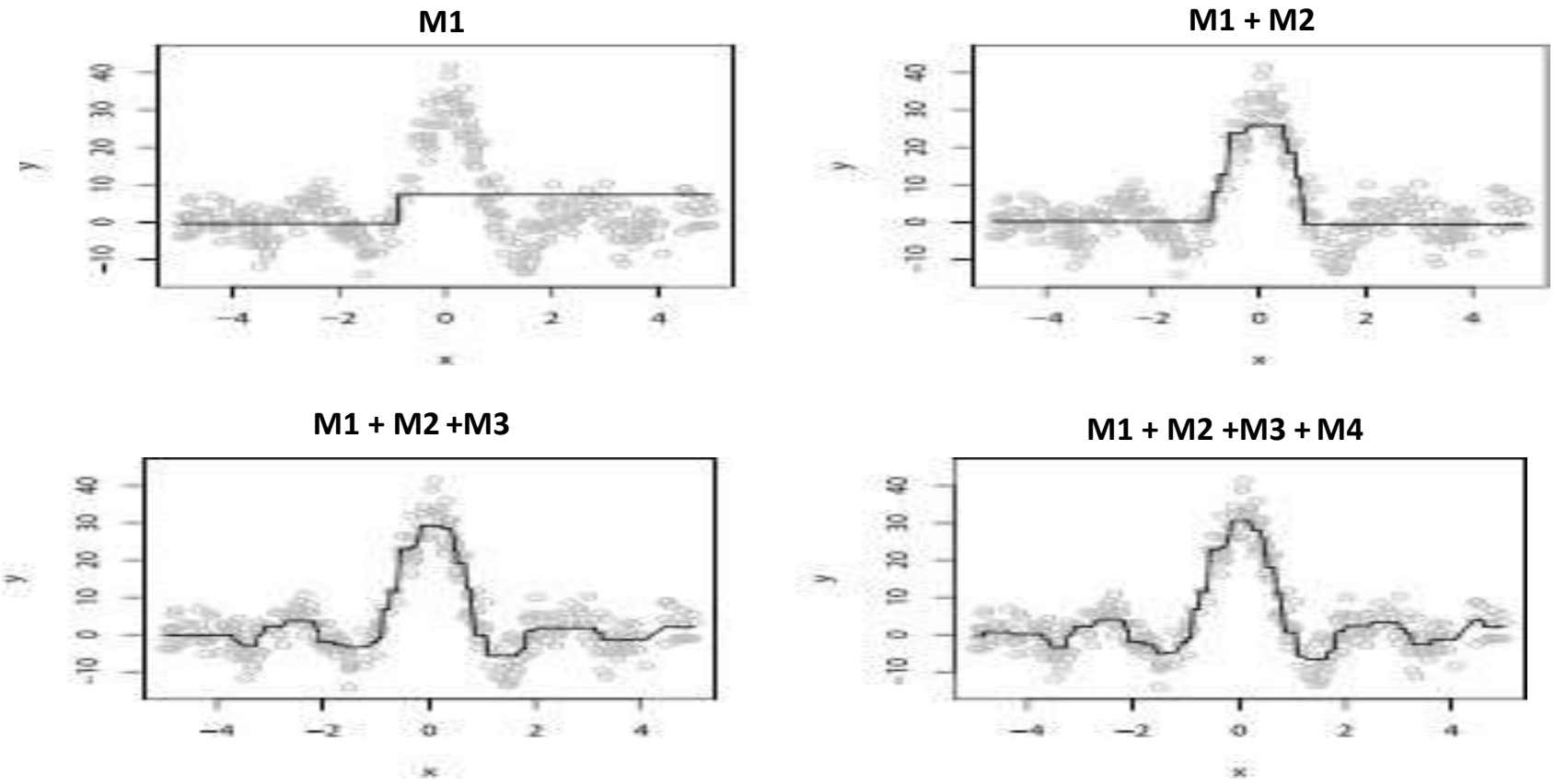
Learning Rate

- The Learning rate determines how fast or slow we will move towards the optimal weights.
- If the λ is very large we will skip the optimal solution.
- If it is too small we will need too many iterations to converge to the best values. So using a good λ is crucial.



GBM in Regression

- The following figure depicts how the Gradient boosting improvises the regression line of Decision tree stump.
- In each step the coefficients are optimized using the previous error terms.



XGBoost

XGBoost

- **XGBoost** is the short form of **eXtreme Gradient Boosting.**
- It is an machine learning package. The XGBoost library implements the gradient boosting decision tree algorithm. It is created by Tianqi Chen.
- XGBoost is a fantastic open source implementation of Gradient Boosting Machines, one of the most accurate general purpose supervised learning algorithms.
- We can use Xgboost package with languages like Python, R, C++, Matlab and Java etc.
- **What makes XGBoost Special ?**:
  - The computational speed of the XGBoost is very high because it has the system feature of **Parallelism, Distributed Computing, Out-of-core computing and Cache optimization.**

XGBoost

- Native implementations are slow, because the algorithm requires one tree to be created at a time to attempt to correct the errors of all previous trees in the model.

- This sequential procedure results in models with really great predictive capability, but can be very slow to train when hundreds or thousands of trees need to be created from large datasets.

- In XGBoost, individual trees are created using multiple cores and data is organized to minimize the lookup times, all good computer science tips and tricks.

Advantages of XGBoost

- Parallelization of tree construction using all of your CPU cores during training.
- Distributed Computing for training very large models using a cluster of machines.
- Out-of-Core Computing for very large datasets that don't fit into memory.
- Cache Optimization of data structures and algorithm to make best use of hardware.
- The XGBoost supports the model features like **Gradient Boosting, Stochastic Gradient Boosting And Regularized Gradient Boosting** (L1 and L2 Regularizations).
- XGBoost handles missing values effectively.

NHyper-Parameter Tuning

HyperParameter Tuning

- **Parameter Tuning:** These are algorithmic method for tuning hyper parameters of machine learning algorithms.
- When we build a model using any machine learning algorithm the parameters has some default values assigned to them, they some times give good results but by tuning them we can get more better results.
- Hyper parameter tuning has big impact on prediction accuracy of machine learning algorithms.
- Optimizing the hyper parameters will differ for each data set there for they should be tuned for each data set.
- The model is trained and results are observed for every single change in parameter values.

Grid Search

- **Grid search:** As name says it takes up a grid of parameter values and run the model on each different values to get better parameters.
- **Working:**
  - A list of values given as input for each parameter associated with the machine learning model.
  - The Cartesian product of these list is taken and each element of the Cartesian product machine evaluates the cost function.
  - The parameter values which yield lower Cartesian values are selected.
  - The grid search requires a minimum and maximum values to set up parameter range, this can be done by running a small grid which a rough estimate of the range.
- Grid search is expensive, as grid search run the model for all the parameter values, it consumes lots of computation time.

Randomized Seacrh

- **Random search:** Instead of selecting a grid of values and running the model we choose the values randomly using probability distribution.
- Based on this random parameters we evaluate the cost function a model.
- **Computational Time:** Random search is slight variation on grid search instead of searching over the entire grid, random search evaluates the model for randomly chosen sample of values hence it reduce the computational time drastically.

Parameters to Tune (1/2)

- **n.trees:** Using this, we can specify number trees to be used in the model.
- **interaction.depth:** It used to specify maximum number of split nodes to be used. By this we can control the overfitting of the tree. By default 6 nodes will give good accuracy.
- **n.minobsinnode:** Minimum number of observation/instances that must be present in a node after splitting. Most of the times we use n.minobsinnode = 10 .
- **Bag.fraction:** All the tree based models divide the data into sub samples for training i.e if bag.fraction = 0.5 then the half of the data is used in each iteration. We can increase the sample size for small data by increasing the **bag.fraction** value more than 0.5.
- **train.fraction:** The first train.fraction * nrows(data) observations are used to fit the gbm and the remainder are used for computing out-of-sample estimates (Prediction values) of the loss function (like out of bag error in random forest). By default, it is 1.

Parameters to Tune (2/2)

proschool
An ims Initiative

- **Shrinkage (Learning Rate)**: a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction.
  - It determines the impact of each tree with the outcome. It starts with the outcome of first tree and it is updated using subsequent trees outcomes.
  - The change from one tree estimate to another tree is controlled by shrinkage factor/learning rate.
  - By default we use 0.1 as a learning rate.
  - If learning rate is near 1.0 then there may be higher chances of overfitting.
  - Usually lower values are preferred as they generalize the results well.
  - Using lower values will be computationally expensive as the lower rates uses more number of trees.

Packages used in Case Studies

- **datetime** - In Python, date, time and datetime classes provides a number of function to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps. Whenever you manipulate dates or time, you need to import datetime function.

- **Xgboost –** Xgboost package is used to perform XG boosting in Python.

- **Sklearn -** scikit-learn is a collection of Python modules relevant to machine/statistical learning and data mining.

- **Joblib -** Joblib is a set of tools to provide lightweight pipelining in Python. In particular, joblib offers: transparent disk-caching of the output values and lazy re-evaluation (memoize pattern).

- **Pandas_profiling** - Generates profile reports from a pandas DataFrame. The pandas df.describe() function is great but a little basic for serious exploratory data analysis. pandas_profiling extends the pandas DataFrame with df.profile_report() for quick data analysis.

- **Seaborn** - Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- **Warnings** - Warning are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn’t warrant raising an exception and terminating the program.

Thank You.