Machine Learning

Ensemble Learning

Dom Huh

Ensemble Learning

Use of multiple learning algorithms at the same time to obtain better predictions (ie. by voting system)

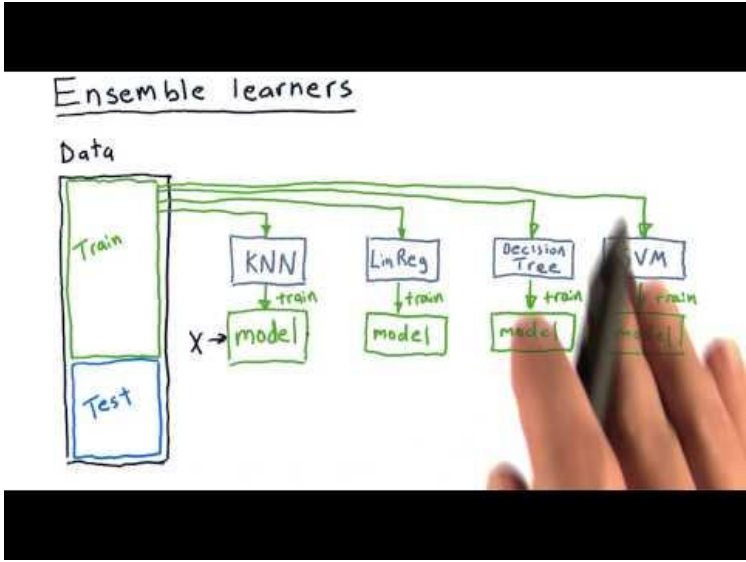
Many benefits of this (ie. avoids high likelihood of overfitting) (Who wants to be a millionaire?)

Commitees - Unweighted votes/average

Weighted averages- upweight better predictors

Feedback/Reinforcement- Incorporates output of pre-predictors, post-processing predictor

Mixture of Experts - restricting region of predictors



Bootstrap Aggregating (Bagging)

Ensemble learning of algorithms trains with randomly selected subsets of the data then combine prediction through model averaging

Bootstrap (statistics):

Confidence Intervals: Takes information/variance from samples, what is the confidence? Find the exact interval of confidences. How?

Duplicate samples by a factor, and randomly sample and take information.

Complexity control: random sampling

HIGH VARIANCE : Therefore low bias. Avoids overfitting but may be too dependent on training set. (Variance in model in different training), meaning higher complexity

Random Forest

Can be used for both regression and **classification**

Uses a ‘forest’ of decision trees

- Each tree gives a classification and gets a vote.
- Takes the average of all the votes

Handles large data sets with high dimensionality

Very little insight of the model’s operation

How it works

Takes the dataset and splits it randomly by sets of feature into a specified number of trees, which are grown to the largest extent possible with no pruning. Then, for classification, the votes from each tree are aggregated for a final prediction.

What parameters would be useful? What are their restrictions?

Isolation Forest (based on ExtraTreeRegressor)

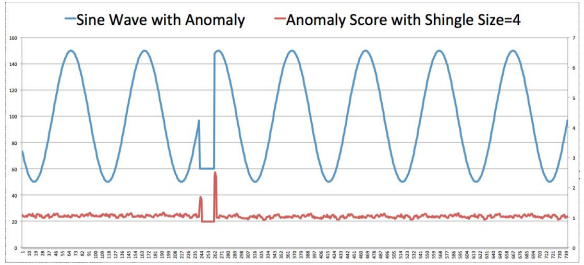
Anomaly detection: Outliers or deviants

Methods: Density (DBSCAN, LOF), Distance (KNN, k-means, hyperplane distance), Parameteric (GMM, 1-class SVM, extreme value theory), statistical tests

Applications: Fraud Detection

Issues: Imbalances, overlaps, drift, uncertainty (Unsupervised)

How it works: Regressor, uses isolation to **separate anomalies,** normalized estimated number of edges in tree, threshold anomaly score



Bagging Implementation

Boosting (Sequential)

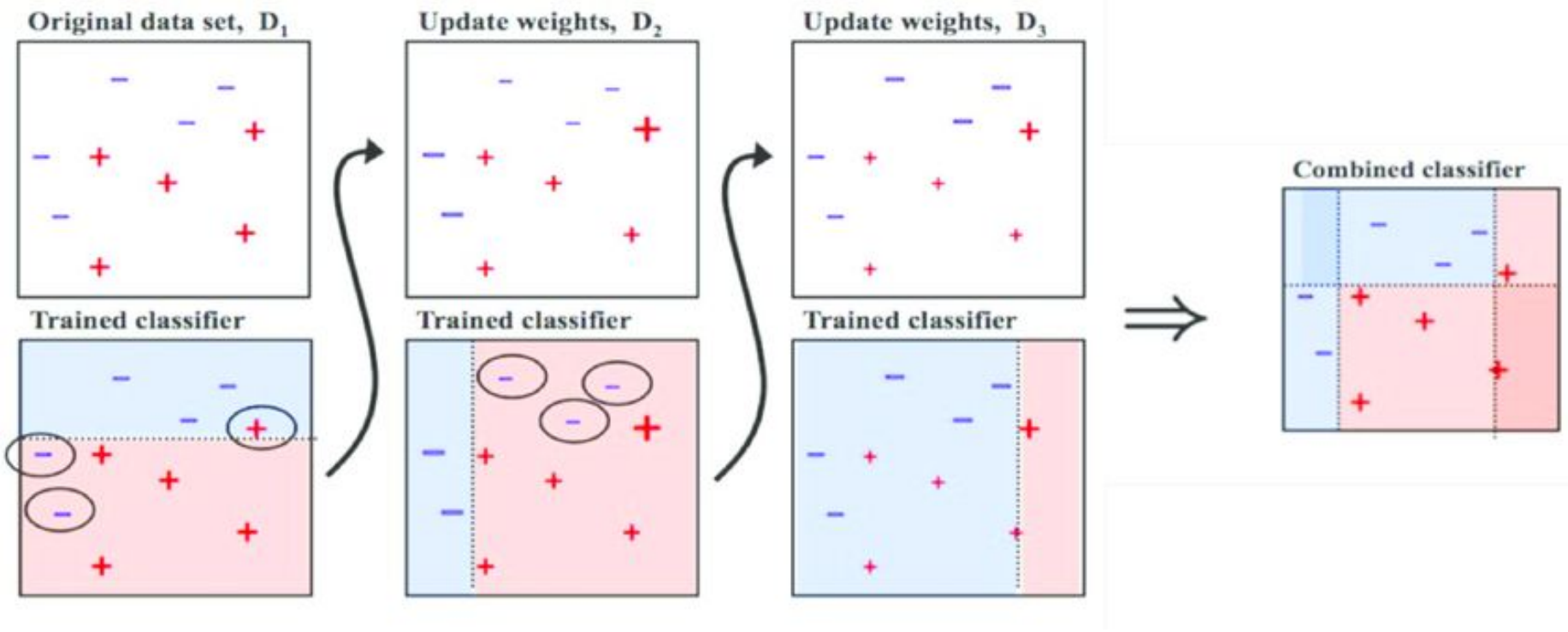
Runs through one algorithm, find incorrect predictions, pass through the same algorithms with adjusted weights in the dataset to accommodate for the errors to be correct, and repeat to minimize weighted error. Use all these models to create a combined classifier

(W/ BAG) Select ONE random subset of data and train onto a model. Test the model with testing data and collect the data that were predicted INCORRECTLY. Make a new random subset of data and add the data that were predicted incorrectly into this new subset. Train a new model with this subset and REPEAT

Look more into Boosting:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.5565&rep=rep1&type=pdf

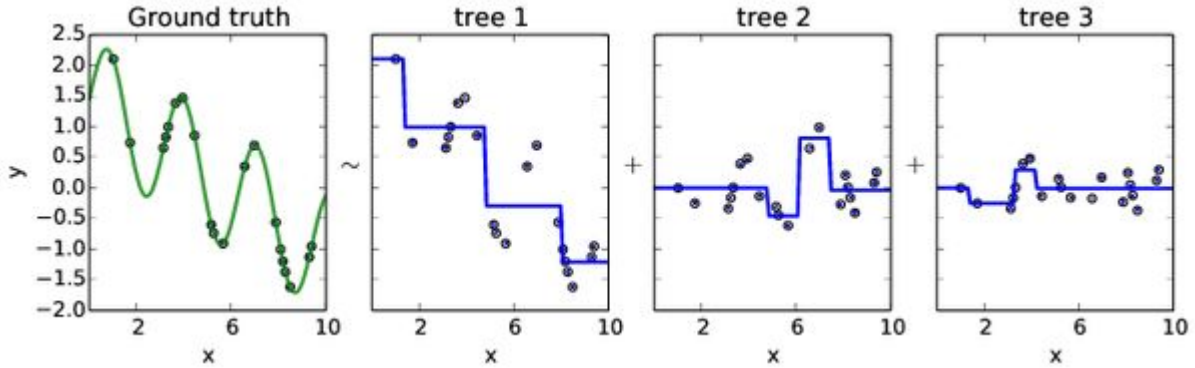
https://www.youtube.com/watch?v=UHBmv7qCey4


Original data set, D_1
Update weights, D_2
Update weights, D_3
Trained classifier
Trained classifier
Trained classifier
Combined classifier

Gradient Boosting

Learns a regression predictor, computer error residual, learn to predict the residual

Make a set a predictions, and calculates error and adjust regression to reduce the error.



Stochastic, Regularized, XGBoost

Stochastic: At each iteration of the algorithm, a base learner should be fit on a subsample of the training set drawn at random without replacement. (Similar to bagging, incorporates OOB errors)

Regularized: Implement some sort of regularization algorithm

Extreme Gradient Boosting: implements decision tree boosting with an additional custom regularization term in the objective function.

XGBoost

Comparisons + Intuition behind internal equations:
https://towardsdatascience.com/boosting-algorithm-xgboost-4d9ec0207d

Adaboost

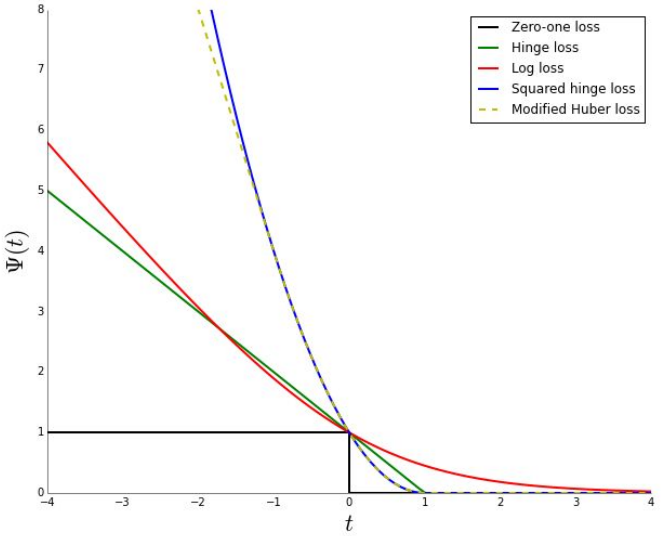
Adaptive Boosting

Train with initial weights on any model and performs boosting

Minimizes exponential surrogate loss functions

$$C_{ada} = \sum_i \exp[-y^{(i)} f(x^i)]$$

Applications: CV Detections



Stacking

Using different types of algorithms, base estimators, to fit and predict in an optimal way to make the meta estimator

For better accuracy, more flexibility included

Not for really large datas (NN), makes training process harder to understand, or for known algorithm for solution

For interpretability, use **L2 penalized logistic regression** for classification, and use **non-negative least squares** for regression.

Use continuous outputs from base estimator in classification problems

KEEP IN MIND

Don’t get carried away with stacking!

Note that diversity is key!

Note that content of data (restrictions) may prevent some approaches!

Note that (true for all ML models) performance does plateau!

Wolpert Stacking

Split into two disjoint sets

Train several base learners on one set

Predict with the other set

Use this predictions as inputs with the correct labels into another layer of higher level learner(s)

Github: Stacking Regressor/Classifiers

https://rasbt.github.io/mlxtend/user_guide/regressor/StackingRegressor/

https://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/

To install: pip install mlxtend

StackNet (Kaggle Use)

Meta-modelling methodology:

General intuition: Gets outputs of models, post processes by layers of higher level models biased on model type based on past knowledge.

Models prediction as inputs to another model (Uses Wolpert's stacking generalization)

Neural Networks with nodes of machine learning algorithms

K-folding for division of data (reshuffling data) [k is hyperparameter of how many division]

NOT COMPUTATIONALLY EFFICIENT AT ALL

https://github.com/kaz-Anova/StackNet

Stacked classifier implementation on kddcups99

Extra: Pipelining and Model Averaging

Next time,

Look into Theano, Keras, PyTorch, Tensorflow

Then, finally neural networks (MLPs, RBMs)