# MACHINE LEARNING - I

UNIT # 7



1

## TODAY'S AGENDA

- Recap of Previous Lecture and Code (grid search, ensemble)
- Stacking
- Huber Loss Regression
- Feature importance using random forest and boosting
- Meta-Heuristics Search for Feature Selection:
  - Simulated Annealing and
  - Genetic Algorithms



2

## RECAP: GRADIENT BOOSTING

- An important parameter of gradient boosting is the learning_rate, which controls how strongly each tree tries to correct the mistakes of the previous trees.
- A higher learning rate means each tree can make stronger corrections, allowing for more complex models.
- Adding more trees to the ensemble, which can be accomplished by increasing n_estimators, also increases the model complexity, as the model has more chances to correct mistakes on the training set.



## RECAP: GRADIENT BOOSTING (CONT'D)

- In contrast to random forests, where a higher n_estimators value is always better, increasing n_estimators in gradient boosting leads to a more complex model, which may lead to overfitting.
- A common practice is to fit n_estimators depending on the time and memory budget, and then search over different learning_rates.
- Another important parameter is max_depth (or alternatively max_leaf_nodes), to reduce the complexity of each tree. Usually max_depth is set very low for gradient boosted models, often not deeper than five splits.

## GRID SEARCH

- Grid search is a technique used in machine learning to optimize the performance of a model by searching over a set of hyperparameters.
- Hyperparameters are the configuration settings of a model that are not learned by the algorithm during training, such as the learning rate, regularization strength, or the number of hidden units in a neural network.
- Grid search works by defining a set of hyperparameters and their corresponding values, and then training and evaluating the model with all possible combinations of these hyperparameters.



## GRID SEARCH (CONT'D)

- For example, if we want to tune the learning rate and regularization strength of a logistic regression model, we might define a grid of hyperparameters like this:
  - Learning rate: 0.001, 0.01, 0.1
  - Regularization strength: 0.01, 0.1, 1
- This results in a grid with 9 combinations of hyperparameters.
- The model is then trained and evaluated for each combination of hyperparameters, using a performance metric such as accuracy or mean squared error.
- The combination of hyperparameters that produces the best performance on the validation set is selected as the optimal hyperparameters, and the model is trained on the entire training set using these hyperparameters.

## SKLEARN IMPLEMENTATION OF BAGGING AND VOTING

- A Bagging regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.
- A voting regressor is an ensemble meta-estimator that fits several base regressors, each on the whole dataset. Then it averages the individual predictions to form a final prediction.



## SUMMARY OF THE TREE ENSEMBLE MODELS

- In bagging, the trees are grown independently on random samples of the observations. Consequently, the trees tend to be quite similar to each other. Thus, bagging can get caught in local optima and can fail to thoroughly explore the model space.
- In random forests, the trees are once again grown independently on random samples of the observations. However, each split on each tree is performed using a random subset of the features, thereby decorrelating the trees, and leading to a more thorough exploration of model space relative to bagging.
- In boosting, the trees are grown successively, using a "slow" learning approach: each new tree is fit to the signal that is left over from the earlier trees, and shrunken down before it is used.

## VARIABLE IMPORTANCE MEASURE

- Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (for bagging classification trees).
- In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.
- Similarly, in the context of bagging classification trees, we can add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.



## PRACTICAL ADVICE

- As both gradient boosting and random forests perform well on similar kinds of data, a common approach is to first try random forests, which work quite robustly.
- If random forests work well but prediction time is at a premium, or it is important to squeeze out the last percentage of accuracy from the machine learning model, moving to gradient boosting often helps.
- In other words, Random Forest works quite robustly on any kind of data, but it might be slower than Gradient Boosting, especially on large-scale problems.

## STACKING

- Stacking takes the output values of machine learning models and then uses them as input values for another algorithm.
- You can, of course, feed the output of the higher-level algorithm to another predictor. It's possible to use any arbitrary topology but, for practical reasons, you should try a simple setup first as also dictated by Occam's razor.
- A fun fact is that stacking is commonly used in the winning models in the Kaggle competition.



## SKLEARN - STACKING

- Stack of estimators with a final regressor.
- Stacked generalization consists in stacking the output of individual estimator and use a regressor to compute the final prediction. Stacking allows to use the strength of each individual estimator by using their output as input of a final estimator.
- Note that estimators_ are fitted on the full X while final_estimator_ is trained using cross-validated predictions of the base estimators using cross_val_predict.

## ROBUST LOSS FUNCTIONS FOR REGRESSION

- On finite samples, squared-error loss places much more emphasis on observations with large absolute residuals |yi - f(xi)| during the fitting process.
- It is thus far less robust, and its performance severely degrades for long-tailed error distributions and especially for grossly mismeasured y-values ("outliers").
- Other more robust criteria, such as absolute loss, perform much better in these situations.
- One such criterion is the Huber loss criterion used for M-regression (Huber, 1964)

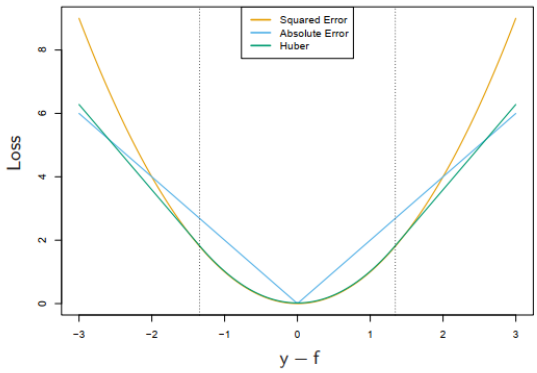
$$L(y, f(x)) = \begin{cases} [y - f(x)]^2 & \text{for } |y - f(x)| \le \delta, \\ 2\delta|y - f(x)| - \delta^2 & \text{otherwise.} \end{cases}$$





## HUBER LOSS

- A comparison of three loss functions for regression, plotted as a function of the margin y-f.
- The Huber loss function combines the good properties of squared-error loss near zero and absolute error loss when |y - f| is large.



## HUBER LOSS FROM SKLEARN

- Linear regression model that is robust to outliers.
- The Huber Regressor optimizes the squared loss for the samples where |(y - X'w) / sigma| < epsilon and the absolute loss for the samples where |(y - X'w) / sigma| > epsilon, where w and sigma are parameters to be optimized.
- The parameter sigma makes sure that if y is scaled up or down by a certain factor, one does not need to rescale epsilon to achieve the same robustness.
- Note that this does not take into account the fact that the different features of X may be of different scales.
- This makes sure that the loss function is not heavily influenced by the outliers while not completely ignoring their effect.



## SIMULATED ANNEALING

- Simulated annealing is a stochastic optimization algorithm that is used to find the global minimum (or maximum) of a given function.
- The algorithm is inspired by the annealing process in metallurgy, where a material is heated and slowly cooled to reduce defects and optimize its physical properties.
- Simulated annealing has been successfully applied to a wide range of optimization problems, including function optimization, combinatorial optimization, and machine learning tasks such as parameter tuning and feature selection.

## SIMULATED ANNEALING (CONT'D)

- In simulated annealing, we start with an initial solution (often chosen randomly) and then iteratively modify the solution in a randomized way to explore the search space.
- At each iteration, we compute the cost or objective function value of the new solution, and then accept or reject the new solution based on a probability distribution that is related to the difference in cost between the new and old solutions and a "temperature" parameter that decreases over time.
- Initially, the temperature is set high so that the algorithm can explore the search space more broadly and avoid getting stuck in local minima.
- As the temperature decreases, the algorithm becomes more selective and focuses on improving the best solutions found so far.



17

## SIMULATED ANNEALING

**Algorithm 12.1:** Simulated annealing for feature selection.

```
Create an initial random subset of features and specify the number of iterations;
for each iteration of SA do
    Perturb the current feature subset;
    Fit model and estimate performance;
    if performance is better than the previous subset then
        Accept new subset;
    else
        Calculate acceptance probability;
        if random uniform variable > probability then
            Reject new subset;
        else
            Accept new subset;
        end
    end
end
```

$$Pr[accept] = \exp\left[-\frac{i}{c}\left(\frac{old - new}{old}\right)\right]$$

By default, c is set to 1



18

## SIMULATED ANNEALING (CONT'D)

- a modification called restarts provides and additional layer of protection from lingering in inauspicious locales.
- If a new optimal solution has not been found within R iterations, then the search resets to the last known optimal solution and proceeds again with R being the number of iterations since the restart.
- The restart threshold was set to 10 iterations.

| Iteration | $R^2$ | Probability | Random Uniform | Status |
|---|---|---|---|---|
| 1 | 0.776 | — | — | Improved |
| 2 | 0.781 | — | — | Improved |
| 3 | 0.770 | 0.958 | 0.767 | Accepted |
| 4 | 0.804 | — | — | Improved |
| 5 | 0.793 | 0.931 | 0.291 | Accepted |
| 6 | 0.779 | 0.826 | 0.879 | Discarded |
| 7 | 0.779 | 0.799 | 0.659 | Accepted |
| 8 | 0.776 | 0.756 | 0.475 | Accepted |
| 9 | 0.798 | 0.929 | 0.879 | Accepted |
| 10 | 0.774 | 0.685 | 0.846 | Discarded |
| 11 | 0.788 | 0.800 | 0.512 | Accepted |
| 12 | 0.783 | 0.732 | 0.191 | Accepted |
| 13 | 0.790 | 0.787 | 0.060 | Accepted |
| 14 | 0.778 | — | — | Restart |
| 15 | 0.790 | 0.982 | 0.049 | Accepted |



## SIMULATED ANNEALING (CONT'D)

- The randomness imparted by this process allows simulated annealing to escape local optima in the search for the global optimum.
- To understand the acceptance probability formula, consider an example where the objective is to optimize predictive accuracy.
- Suppose the previous solution had an accuracy of 0.85 and the current solution has an accuracy of 0.80. The proportionate decrease of the current solution relative to the previous solution is 5.9%.
- If this situation occurred in the first iteration, then the acceptance probability would be 0.94. At iteration 5, the probability of accepting the inferior subset would be 0.75.
- At iteration 50, this the acceptance probability drops to 0.05. Therefore, the probability of accepting a worse solution decreases as the algorithm iteration increases.

## EVOLUTIONARY ALGORITHMS

- Evolutionary algorithms are a family of optimization algorithms that are inspired by the principles of natural evolution and genetics.
- The goal of evolutionary algorithms is to search for the optimal solution in a given search space, typically by generating a population of candidate solutions and iteratively improving them over time through the application of genetic operators such as selection, crossover, and mutation.
- Like simulated annealing, evolutionary algorithms have been successfully applied to a wide range of optimization problems, including function optimization, combinatorial optimization, and machine learning tasks such as parameter tuning and feature selection.

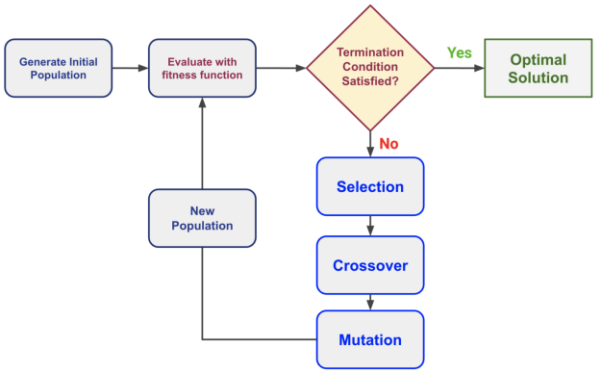


## EVOLUTIONARY ALGORITHMS (CONT'D)

- The key idea behind evolutionary algorithms is that they explore the search space by generating and modifying candidate solutions using the principles of natural selection and genetics.
- This allows them to search for solutions in a robust and flexible way, and to handle complex, non-linear optimization problems with many local optima.
- It must be noted that they can be computationally expensive and require careful tuning of parameters and operators to achieve good performance on a given problem.

## GENETIC/EVOLUTIONARY ALGORITHM

- For the purpose of feature selection, the population consists of all possible combinations of features for a particular data set.
- A chromosome in the population is a specific combination of features (i.e., genes), and the combination is represented by a string which has a length equal to the total number of features.
- The fitness criterion is the predictive ability of the selected combination of features.








## EVOLUTIONARY ALGORITHM (CONT'D)

- Once the initial generation has been created, the fitness (or predictive ability) of each feature subset is estimated.
- A subset of these feature sets will be selected as parents to reproduce and form the next generation.

| ID | | | | | | | | | | Fitness | Probability (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | C | | | F | | | | 0.54 | 6.4 |
| 2 | A | | | D | E | F | | H | | 0.55 | 6.5 |
| 3 | | | | D | | | | | | 0.51 | 6.0 |
| 4 | | | | | E | | | | | 0.53 | 6.2 |
| 5 | | | | D | | | G | H | I | 0.75 | 8.8 |
| 6 | | B | | | E | | G | | I | 0.64 | 7.5 |
| 7 | | B | C | | | F | | | I | 0.65 | 7.7 |
| 8 | A | | C | | E | | G | H | I | 0.95 | 11.2 |
| 9 | A | | C | D | | F | G | H | I | 0.81 | 9.6 |
| 10 | | | C | D | E | | | | I | 0.79 | 9.3 |
| 11 | A | B | | D | E | | G | H | | 0.85 | 10.0 |
| 12 | A | B | C | D | E | F | G | | I | 0.91 | 10.7 |

## EVOLUTIONARY ALGORITHM (CONT'D)

- The most common approach to select parents is to use a weighted random sample with a probability of selection as a function of predictive performance.
- Suppose subsets 6 and 12 were selected as parents for the next generation.

| ID | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 6 | | B | | | E | | G | I |
| 12 | A | B | C | D | E | F | G | I |

| ID | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 13 | | B | | | E | F | G | I |
| 14 | A | B | C | D | E | | G | I |

| ID | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 13 | | B | | | E | F | G | |
| 14 | A | B | C | D | E | | G | I |



## PSEUDO CODE

1. Initialization: A population of candidate solutions is randomly generated.
2. Evaluation: The fitness of each candidate solution is evaluated using a fitness function, which measures how well the solution solves the problem at hand.
3. Selection: A subset of the population is selected for the next generation based on their fitness values. The idea is to favor solutions that are more fit, so they are more likely to pass their genes to the next generation.
4. Reproduction: The selected solutions are used to generate new candidate solutions through the application of genetic operators such as crossover and mutation.
5. Evaluation: The fitness of the new candidate solutions is evaluated.
6. Termination: The algorithm terminates when a stopping criterion is met, such as when a satisfactory solution is found, or when a maximum number of generations or evaluations is reached.

## FUNCTIONS FOR EXPERIMENTATIONS

$$f(x,y) = [1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)]\times$$
$$[30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)].$$

$$y = x_1 + \sin(x_2) + \log(|x_3|) + x_4^2 + x_5x_6 + I(x_7x_8x_9 < 0) + I(x_{10} > 0) +$$
$$x_{11}I(x_{11} > 0) + \sqrt{|x_{12}|} + \cos(x_{13}) + 2x_{14} + |x_{15}| + I(x_{16} < -1) +$$
$$x_{17}I(x_{17} < -1) - 2x_{18} - x_{19}x_{20} + \epsilon.$$