Machine Intelligence (UE18CS303)
Unit 3

Aronya Baksy

October 2020

1 Ensemble Learning

- Ensemble Learning is a pseudo-algorithm wherein *multiple learning* algorithms are combined in a way that offers better performance than the constituent algorithms alone.
- Usually the models used in Ensemble learning are known as **weak learners**. Weak learners are those that deliver only slightly better performance than just randomly choosing predicting labels (which gives an accuracy of 0.5).
- An ensemble learning consists of constructing multiple, diverse predictive models from *adapted versions* of the training data (most often reweighted or resampled)
- Ensemble learning takes in multiple models having **high bias** (ie. very simple models), which when run together, produce a low-bias and low-variance classification output.
- The combined hypothesis space of the combination of learners may not be completely overlapping with the hypothesis space of the individual constituent learners.
- Following are some key ideas on ensemble learning:
  - Ensemble learning can reduce overfitting (overfitting is an outcome of high variance).
  - The confidence in the outcome of the ensemble model is

    $$C = 1 - (1 - A)^n$$

    Assuming that all learners have same accuracy A (this is not a nice assumption in real world).
  - Let n_1 learners predict class c_1 and n_2 learners predict class c_2. Let us assume with no loss of generality that $n_1 > n_2$. Then the probability that the class is actually class c_1 is:

    $$C(n, n_2)A^n(1 - A)^{n-n_2}$$

    where $n = max(n_1, n_2)$
  - The final prediction from N learners can be made using:

    $$y = \sum_{i=1}^{n} w_i d_i$$

    Where w_i is the weight given to the learner i and d_i is the prediction made by learner i.
  - The weights can be assigned in proportion to the accuracy of that learner, or in inverse proportion to their variance.
  - The most important constraint in ensemble learning is that all the learners must be *independent* of one another in terms of their parameters and all effects on one another. It should be possible to train all the models completely in parallel.
  - Learners can be made independent through learning techniques like **bagging** and **boosting**.

1.1 Bagging

- Let there be k learners in the ensemble. From the original dataset of size N (say), we create k datasets each of size N by doing random sampling with replacement from the original.
- The datasets will have approximately 63.2% of the unique examples of the original data, with the rest being duplicates.
- The learners are independently run on their corresponding datasets. The final output is either a vote or an average of all the individual model outcomes.
- The error calculation for a learner L_i using the dataset D_i is done by running the learner L_i on the examples that are a part of D (the original dataset) but not D_i. The total error is the average error on all learners.
- The advantages of bagging are:
  - Easy to implement and interpret
  - Models can grow in parallel.
  - Less variability than only DTs
  - Heterogeneous data can be passed, no pre processing is needed.

1.2 Boosting

- In a boosting setup, learners run serially and they learn from the misclassifications of the previous learners.
- Each learner has its own weight (weight by variance or accuracy as above).
- Each instance in the training dataset also has its own weight. When a learner L_1 makes a mistake on some instances, the next learner in the sequence L_2 is told to make other mistakes but it classifies the examples that L_1 went wrong in, correctly (this is done by assigning those instances a larger weight).
- All the learners use identical datasets (the original data) but with different instance weights.
- The final result is the weighted summation of all the individual learner results.

1.3 AdaBoost (Adaptive Boosting)

- Each instance in the training data (of size N examples) has an initial weight $\frac{1}{N}$
- Let the function $I(a, b)$ be defined as

$$I(a,b) = \begin{cases} 1 \text{ if } a \neq b \\ 0 \text{ if } a = b \end{cases}$$

- The error for the m^{th} classifier is given as

$$\varepsilon_m = \sum_{n=1}^{N} w_n^{(m)} I(y_n, t_n) \tag{1}$$

- The weight of the m^{th} classifier is given as:

$$\alpha_m = \frac{1}{2} ln \left(\frac{1-\varepsilon_m}{\varepsilon_m} \right) \tag{2}$$

- Let N_m be defined as

$$N_m = e^{\alpha_m} \sum w_i \text{ for wrongly classified examples } + e^{-\alpha_m} \sum w_i \text{ for correctly classified examples}$$

- Now the instance weights are modified for use by the $m+1^{th}$ classifier, as follows

$$w_n^{(m+1)} = \begin{cases} \frac{w_n^{(m)}}{N_m} e^{-\alpha_m} \text{ if } x_n^{(m)} \text{ is correctly classified} \\ \frac{w_n^{(m)}}{N_m} e^{+\alpha_m} \text{ if } x_n^{(m)} \text{ is wrongly classified} \end{cases} \tag{3}$$

- But, from equation (1) it is clear that

$$\sum w_i \text{ for wrongly classified examples } = \varepsilon$$
$$\sum w_i \text{ for correctly classified examples } = 1 - \varepsilon$$

This gives

$$N_m = 2\sqrt{\varepsilon(1-\varepsilon)} \tag{4}$$

- Substituting this in equation (3) gives

$$w_n^{(m+1)} = \begin{cases} \frac{w_n^{(m)}}{2(1-\varepsilon)} \text{ if } x_n^{(m)} \text{ is correctly classified} \\ \frac{w_n^{(m)}}{2\varepsilon} \text{ if } x_n^{(m)} \text{ is wrongly classified} \end{cases} \tag{5}$$

- Hence given a dataset D, number of learners as T and the learning algorithm A, the AdaBoost Algorithm is summarized as:

Algorithm 1 AdaBoost

```
procedure AdaBoost(D, T, A)
    w_i1 ← 1/|D| ∀ x_i ∈ D
    for t =1 to T do
        Run A on dataset D with weights w_i1 for ith example to get a model M_t
        ε_m ← Σ_{n=1}^{N} w_n^(m) I(y_n, t_n)
        if ε_t ≤ 0.5 then
            break
        end if
        α_t ← (1/2) ln((1−ε_m)/ε_m)
        w_{t+1}i ← w_(t)i / 2ε for wrongly classified
        w_{t+1}i ← w_(t)i / 2(1−ε) for correctly classified
    end for
     return M(x) ← Σ_{i=1}^{T} α_t M_t(x)
end procedure
```