Tutorial 10

CS 337 Artificial Intelligence & Machine Learning, Autumn 2019

November, 2019

1 Adaboost

Recall the main idea behind Adaboost.Let $C_t(\mathbf{x}) = \sum_{j=1}^{t} \alpha_j T_j(\mathbf{x})$ be the boosted linear combination of classifiers until t^{th} iteration. Let the error to be minimized over α_t be the sum of its exponential loss on each data point,

$$\mathbf{E}_t = \sum_{i=1}^{m} \exp\left(-y^{(i)} C_t\left(\mathbf{x}^{(i)}\right)\right) = \sum_{i=1}^{m} \exp\left(-\left(y^{(i)} \sum_{j=1}^{t} \alpha_j T_j(\mathbf{x}^{(i)})\right)\right)$$

Prove the following claims for Adaboost

1. Claim1: The error that is the sum of exponential loss on each data point is an upper bound on the simple sum of training errors on each data point

   Solution:

   The simple sum of training errors on each data point is

   $$\sum_{i=1}^{m} \delta\left(y^{(i)} \neq sign\left(C_t\left(\mathbf{x}^{(i)}\right)\right)\right) \leq \sum_{i=1}^{m} \exp\left(\delta\left(y^{(i)} \neq sign\left(\sum_{i=1}^{m} \alpha_i T_i(\mathbf{x}^{(i)})\right)\right)\right)$$

   We next note that the exponential function[1] is a convex function. That is,

   $$\exp\left(\sum_{i=1}^{m} \beta_i r_i\right) \leq \sum_{i=1}^{m} \beta_i \exp(r_i)$$

   for each $\beta_i \in [0, 1]$ such that $\sum_{i=1}^{m} \beta_i = 1$. By this convexity, one can derive (see Figure 1 on the next page:

   $$\mathbf{E}_t = \sum_{i=1}^{m} \delta\left(y^{(i)} \neq sign\left(C_t\left(\mathbf{x}^{(i)}\right)\right)\right) \leq \sum_{i=1}^{m} \exp\left(-y^{(i)} C_t\left(\mathbf{x}^{(i)}\right)\right)$$

[1] https://en.wikipedia.org/wiki/Exponential_function

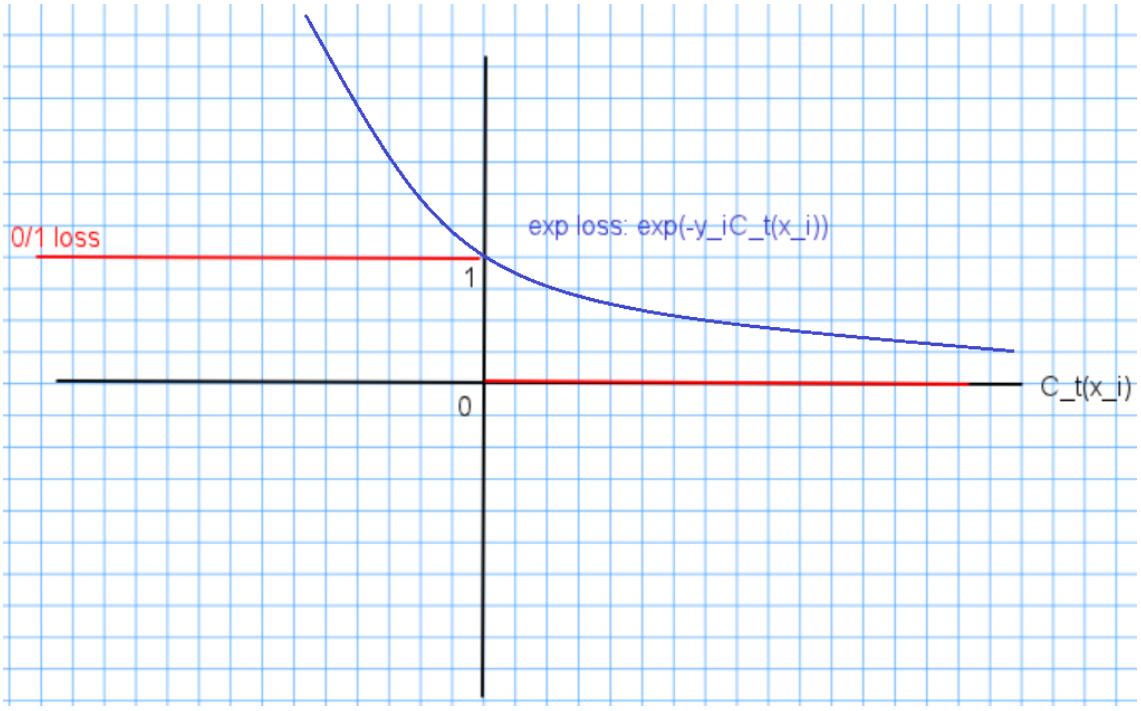

Figure 1: Figure Illustrating how the **exponential loss** is an upper bound for the **0/1 loss**

2. Claim2: $\alpha_t = (1/2)\ln((1 - err_t)/err_t)$ actually minimizes this upper bound.

3. (Advanced) Claim3: If each classifier is slightly better than random, that is if $err_t < 1/K$, Adaboost achieves zero training error exponentially fast

2 Boost Cluster

Consider the points in Figure 2. There are 4 clusters of points, (a) Profs who are Linguists, (b) Profs who are Computer Scientists, (c) Students who are Linguists and (d) Students who are Computer Scientists.

Suppose we wanted to cluster the points using the K means algorithm, with a value of $K = 2$. With this, it is not clear which are the two clusters that we would like the K means algorithm to generate. But suppose we were given constraints between points of the form **Must-link** and **Cannot-link**. For example, we see that the two constraints in Figure 2 assert that

1. one of the points corresponding to (Prof, Linguist) belongs to the same cluster as the point corresponding to (Student, Linguist)

2. the point corresponding to (Prof, Linguist) does not belong to the same cluster as the point corresponding to (Prof, Computer Scientist)

Thus, based on the above constraints, one would expect the **ideal** two clusters to correspond to 'Linguist' and 'Computer Scientist' respectively.

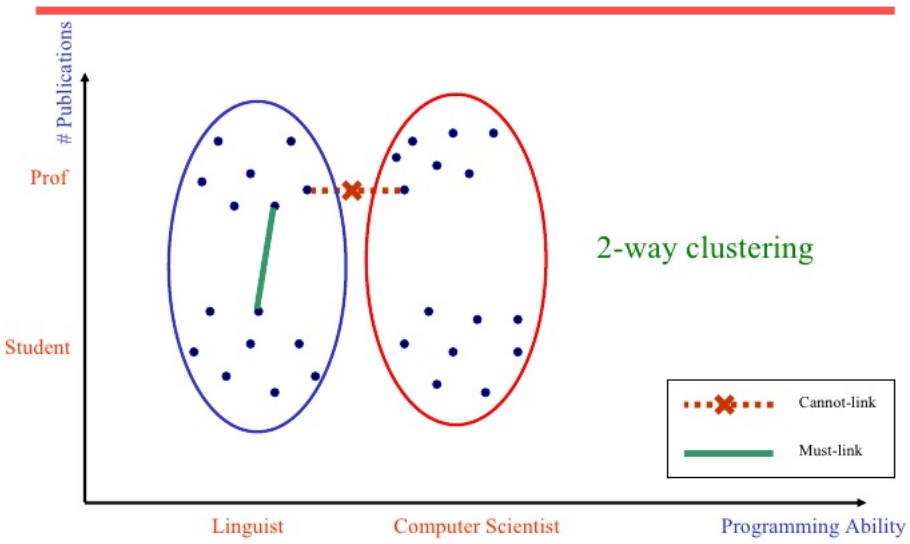

Figure 2: Clustering with Must Link and Cannot Link Constraints

Let $X = (x_1, \ldots, x_n)$ denote the collection of examples to be clustered, where n is the total number of examples and each example $x_i \in \Re^d$ is a vector of d dimensions. We use a matrix $S^+ \in \Re^{n \times n}$ to represent all the must-link pairs, where $S^+_{i,j} = 1$ when examples x_i and x_j form a must-link pair, and $S^+_{i,j} = 0$ otherwise. Similarly, we use a matrix $S^- \in \Re^{n \times n}$ to represent all the cannot-link pairs, where $S^-_{i,j} = 1$ when examples x_i and x_j form a cannot-link pair, and zero otherwise.

1. Propose a heuristic and intuitive modification to the K-means clustering algorithm in order to accommodate such constraints in clustering.

   Hint: You can use the boosting paradigm.

2. We next attempt to solve the problem move formally by proposing an optimization problem (in exponential form, very much like we did for Adaboost). In order to identify which constraint pairs are not well satisfied, we introduce the kernel similarity matrix $K \in \Re^{n \times n}$, where $K_{i,j} \geq 0$ indicates the confidence of assigning examples x_i and x_j to the same cluster. We propose the following objective function to be minimized to find/determine the optimal kernel matrix K through some modified clustering algorithm:

$$L = \sum_{i,j=1}^{n} \sum_{a,b=1}^{n} S^+_{i,j} \times \_\_\_\__1 \times \exp\left(\_\_\_\__2 \times \_\_\_\__3\right)$$

   Fill in the three blanks $\_\_\_\__1$, $\_\_\_\__2$ and $\_\_\_\__3$ as functions of K, S^+ and S^-. Explain the objective thus formed. Note that $\exp(t)$ stands for e^t and $\times$ stands for the multiplication operation.

SOLUTION. Solution to problem 1: The goal of the modified boosting algorithm is to identify the subspace that keeps the data points in the unsatisfied must-link pairs close to

each other, and keeps the data points from the unsatisfied cannot-link pairs well separated. One option is to weight each point in K-means based on its importance and boost the memberships of must-link points to the same clusters when land up belonging to different clusters at the end of any specific iteration. We will also entertain other sound heuristic variants than just the one specified.

SOLUTION. Solution to problem 2: (a) $\_\_\_\__1 = S^-_{a,b}$ (b) $\_\_\_\__2 = K_{a,b}$ and (c) $\_\_\_\__2 = 1 - K_{i,j}$ so that

$$L = \sum_{i,j=1}^{n} \sum_{a,b=1}^{n} S^+_{i,j} S^-_{a,b} \exp(K_{a,b} - K_{i,j})$$

or (c) $= (K_{a,b} - K_{i,j})/K_{a,b}$ etc... The objective measures the inconsistency between the kernel matrix K and the given pairwise constraints. When all the constraints are satisfied, we expect to observe a large value for kernel similarity $K_{i,j}$ if x_i and x_j form a must-link pair, and a small value for $K_{i,j}$ if x_i and x_j form a cannot-link pair. In the above, each term within the summation compares $K_{a,b}$, i.e., the similarity between two points from a cannot-link pair, to $K_{i,j}$, i.e., the similarity between two data points from a must-link pair. By minimizing the objective function we will ensure that all the data points in the must-link pairs are more similar to each other than the data points in the cannot-link pairs.

3 Decision Trees and Feature Selection

- **QUESTION:** We discussed the information gain criterion for feature splitting in Decision trees. Suggest two other criteria. Motivate each.

3.1 ANSWER

Solution[2]**:** The splitting attribute is selected greedily as mentioned in the last class and is based on maximum reduction in impurity. The expression is given by:

$$_{V(\phi_i),\phi_i} \left(Imp(S) - \sum_{v_{ij} \in V(\phi_i)} \frac{|S_{v_{ij}}|}{|S|} Imp(S_{v_{ij}}) \right)$$

where $S_{ij} \subseteq \mathcal{D}$ is a subset of dataset such that each instance x has attribute value $\phi_i(x) = v_{ij}$.

1. An example choice of $Imp(S)$ discussed in the class notes is the entropy[3] $Imp(S) = H(S) = -\sum_{i=1}^{K} Pr(C_i) \bullet \log(Pr(C_i))$.

[2]Section 5.1 of `https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725.pdf`

[3]See slide 5 of `http://23.253.82.180/course/307/858/2217`

2. Alternative impurity measures are shown in Table 1 and they all measure the extent of spread of the probabilities over the classes. In other words, as shown in Figure 3 each impurity measure indicates the extent to which the data is "confused" about the classes.

Name	Imp (D)
Entropy	$-\sum_{i=1}^{K} Pr(C_i) \bullet \log(Pr(C_i))$
Gini Index	$\sum_{i=1}^{K} Pr(C_i)(1 - Pr(C_i))$
Class (Min Prob) Error	$_i(1 - Pr(C_i))$

Table 1: Decision Tree: Impurity measurues

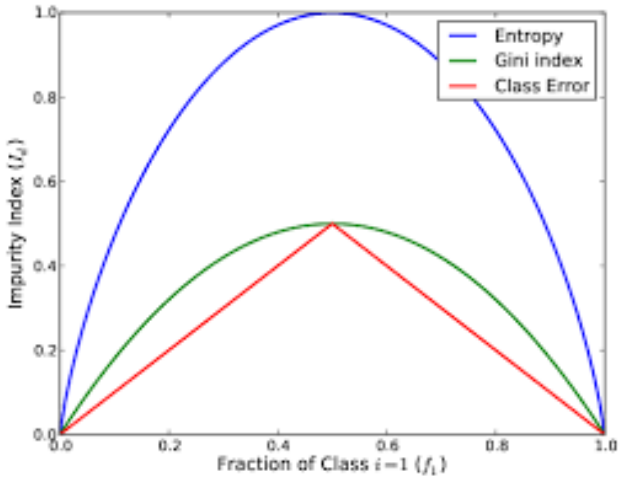

Figure 3: Plot of Entropy, Gini Index and Misclassification Accuracy. Source: `https://inspirehep.net/record/1225852/files/TPZ_Figures_impurity.png`

The second term in the above expression is the expected new impurity. V is a function which returns the split values given an attribute ϕ_i. So $V(\phi_i)$ can be varied for any ϕ_i. It could have many values or a range of values. A example of $V(\phi_i)$ is $V(\phi_i) = \{1, 3, 5\}$ which translates to the following split points $\phi_i < 1,\ 1 \leq \phi_i < 3,\ 3 \leq \phi_i < 5,\ \phi_i \geq 5$

3.1.1 An empirical observation

Since smaller range of attribute values in each split in V_i tends to lead to more skewed class distribution in that split, larger $|V(\phi_i)|$ generally yields larger reduction in impurity. This makes the algorithm to choose more skewed and complex trees and leads to the problem of `overfitting`, if not fixed. Recall that in overfitting, the system

learns a model which is specific to the training collection and does not generalize well on unseen data.

3.1.2 Need to address this empirical observation

This could be achieved by the maximization of the following expression:

$$Imp_{skew}(S) = Imp(S) \; - \; \left(\frac{\sum_{v_{ij} \in V(\phi_i)} \frac{|S_v|}{|S|} Imp(S_{v_{ij}})}{-\sum_{v \in V(\phi_i)} \frac{|S_v|}{|S|} \log(S_v)} \right)$$

The second term in the above expression is called $\triangle Imp(S)$. The intuition for using this term is that, more the skew, lower will be the denominator and is therefore better at countering lowered impurity. In other words, this new measure prefers a less skewed tree as shown in Figure 4. We will refer to the above modified measure $Imp_{skew}(S)$ as the Skew Adjusted Information Gain.

Figure 4: Skewness and empirical observation

3.1.3 Summing it all up

Below we present the overall decision tree learning algorithm with stopping criteria as described above. Ideally, this algorithm goes on until all the data points at the leaf are of the same single class and the two stopping criterion added to the algorithm make it terminate even if such a condition does not occur.

ϕ = empty return a tree with only one branch C_j, where C_j is the majority class in S Stopping criterion all instances in S have label = c_i return a tree with only one branch c_i Stopping criterion $\phi_j =_{V(\phi_i),\phi_i} \left(\triangle Imp(S)\right) \quad \forall v \in V(\phi_i) \; T_v = dtree(S_v, \phi - \phi_i, V)$ return a tree rooted at ϕ_i and having all the T_v branches

- **QUESTION:** We also discussed how the information gain criterion can be used for feature selection in general. Suggest two other criteria. You can build on your answer to the question above.

3.2 ANSWER

The answer is similar to the one above. All the three impurity functions, *viz.*, entropy, gini-index and misclassification error could be used for feature selection. Moreover, the Skew Adjusted Information Gain measure $Imp_{skew}(S)$ could also be used for feature selection.

- **OPTIONAL QUESTION:** Suggest how you could used hypothesis testing for pruning or stopping decision tree construction.

3.3 ANSWER

Simpler trees are preferred over their complex counterparts for the following reasons:

1. They are faster to execute
2. They perform better on unseen data. In otherwords, they "generalize well". For instance, a simpler tree learnt in the class had lower accuracy on the train set but higher accuracy on the unseen test set.

3.3.1 Alternatives in Pruning

There are various strategies/heuristics to decrease the complexity of the tree learnt. Some of the options are as follows:

1. Early termination. Stop if $\triangle Imp(S) < \theta$, where θ is some threshold.
2. Majority class $\geq \alpha\%$, for some value of α
3. Pruning: The idea is to build complex trees and prune them. This is a good option, since the construction procedure can be greedy and does not have to look ahead. Some Hypothesis testing procedures could be used to achieve this. (For instance, the binomial and χ^2-tests).
4. Use an objective function like

$$max_{\phi_i}\Big(\triangle Imp(S,i) - Complexity(tree)\Big)$$

The complexity is characterized by the description length principle (MDL)

Please note that the section that follows is a completely optional reading. However, it could enhance your understanding significantly.

3.3.2 (OPTIONAL) Hypothesis Testing for Decision Tree Pruning

The question to answer while constructing a simpler decision tree is Do we have to split a node (using some attribute) or not ? Consider the situation in Figure 5. The numbers n_1, n_2 etc. indicate the number of instances of the particular class.

If the class ratios of instances remain similar to that before the split, then we might not gain much by splitting that node. To quantify this, we employ Hypothesis testing. The idea is to compare 2 probability distributions.

We will illustrate with a 2-class classification problem. If p is the probability of taking the left branch, the probability of taking the right branch is $1 - p$. Then we obtain the following:

$$\begin{aligned} n_{11} &= pn_1 \\ n_{21} &= pn_2 \\ n_{12} &= (1-p)n_1 \\ n_{22} &= (1-p)n_2 \end{aligned}$$

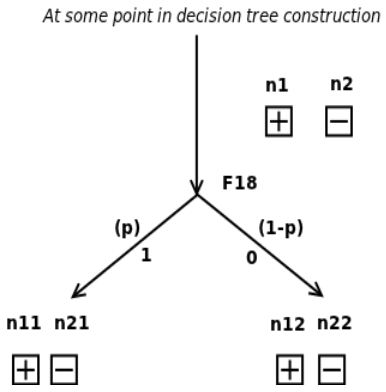

Figure 5: Splitting criterion

Consider the original ratio (also called reference distribution) of positive to total no. of tuples. If the same ratio is obtained after the split, we will **not** be interested in such splits. i.e.,

$$\frac{n_1}{n_1 + n_2} = \frac{n_{11}}{n_{11} + n_{21}}$$

or in general

$$\frac{n_j}{n_1 + n_2} = \frac{n_{j1}}{n_{1i} + n_{2i}} \quad \forall j \text{ no. of classes } \& \ i = 1, 2$$

Why? Because these splits only add to the complexity of the tree and do not convey any meaningful information not already present. Suppose we are interested not only in equal distributions but also in approximately equal distributions i.e.,

$$\frac{n_1}{n_1 + n_2} \approx \frac{n_{11}}{n_{11} + n_{21}}$$

The idea is to compare two probability distributions. It is here that the concept of hypothesis testing is employed.

The ratio $\frac{n_1}{n_1+n_2}$ is called the reference distribution. In general, it is:

$$p(C_i) = \mu_i = \frac{n_i}{\sum_{i=1}^{K} n_i} \quad \text{for a given class } i \text{ (pre-splitting distribution)}$$

3.3.3 Hypothesis testing: problem

Note: The text presented here in red refers specifically to the decision tree problem. The text in black is for the general Hypothesis testing problem and can be read up in Section 7 of `https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725_aut11.pdf`

Let $X_1, \ldots X_n$ be i.i.d random samples that correspond to class labels of instances that have gone into the left branch.

The null hypothesis is H_0 and the alternative hypothesis is H_1:

$$H_0: \quad X_1, \ldots, X_n \in C$$
$$H_1: \quad X_1, \ldots, X_n \notin C$$

The distribution of samples in the left branch is same as before splitting i.e. $\mu_1, \ldots \mu_k$.

Given a random sample, we need to test our hypothesis i.e., given an $\alpha \in [0, 1]$, we want to determine a C such that

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C) \leq \alpha \qquad \text{Type I error}$$

Given an $\alpha \in [0, 1]$, probability that we decide that the pre-distribution and the left-branch distribution are different, when in fact they are similar, is less than or eqaul to α.

[Currently we are not very much interested in the Type II error, i.e. $Pr_{H_1}(\{X_1, \ldots X_n\} \in C)$].

Here, C is the set of all possible "interesting" random samples. Also,

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C') \leq Pr_{H_0}(\{X_1, \ldots X_n\} \notin C) \qquad \forall C' \supseteq C$$

We are interested in the "smallest" / "tightest" C. This is called the critical region C_α. Consequently,

$$Pr_{H_0}(\{X_1, \ldots X_n\} \notin C_\alpha) = \alpha$$

3.3.4 Goodness-of-fit test for DTrees

Consider the test statistic $S_i = \sum_{j=1}^{n} \delta(X_j, C_i)$.

We are interested in the hypothesis H_0 where new-distribution = old-distribution$(\mu_1, \ldots \mu_j)$.

$$\begin{aligned}
\mathcal{E}_{H_0}[Y_i] &= \sum_{j=1}^{n} \mathcal{E}_{H_0}[\delta(X_j, C_i)] \\
&= \sum_{j=1}^{n} \mu_i * 1 + (1 - \mu_i) * 0 \\
&= n\mu_i
\end{aligned}$$

$$\begin{aligned}
C &= \left\{ (X_1, \ldots X_n) \middle| \sum_{i=1}^{K} \frac{(Y_i - \mathcal{E}_{H_0}(Y_i))^2}{\mathcal{E}_{H_0}(Y_i)} \leq c \right\} \qquad \text{where } c \text{ is some constant} \\
&= \left\{ (X_1, \ldots X_n) \middle| \sum_{i=1}^{K} \frac{(Y_i - n\mu_i)^2}{n\mu_i} \leq c \right\}
\end{aligned}$$

As we might have seen in a basic course on statistics[4], the above expression $\sim \chi^2_{K-1}$. We then use the chi-square tables to find c given the value of α.

[4] Section 7 of `https://www.cse.iitb.ac.in/~cs725/notes/classNotes/extra_lecturenotes_cs725.pdf`

3.3.5 Final Heuristic used for DTree construction

- Compute $\sum_{i=1}^{K} \frac{(Y_i - n\mu_i)^2}{n\mu_i} \sim t_s \quad \forall\ splits$, where t_s is the test statistic.
- Stop building the tree, if for a given α, $\quad t_s \leq c_\alpha \quad \forall\ splits$
- Compute c_α such that $Pr_{\chi^2_{K-1}}(x \geq c_\alpha) = \alpha$