CS5011: Introduction to Machine Learning **Fall 2014**

Lecture 16: Ensemble Methods

Lecturer: Prof. B. Ravindran *Scribes: J.P. Sagar (CS12B039)*

In this lecture notes we look at the following topics:

- Bagging
- Boosting
- Stacking

1 Bagging

Bagging is bootstrap aggregation. It is essentially that if we have a data set say 'D' and we sample it with replacement to get 'M' data sets say $D_1, D_2, ..., D_M$ each of size 'N' same as that of 'D'. These $D_1, D_2, ..., D_M$ are also called as bags. Now train a classifier on each of these data sets and let them be $G_1, G_2, ..., G_M$. Final classifier is the aggregation of all these classifiers 'G' thereby reducing the variance. The bagging estimate is defined by

$$G(x) = \frac{1}{M}\sum_{i=1}^{M} G_i(x)$$

K-NN is unstable. Decision trees are highly unstable if 'N' the number of points in the dataset is very small. Stability is how an algorithm is perturbed by small changes to its inputs.
Support Vector Machines are stable unless the support vectors are changed. As the random perturbation has less probability for changing the SV's they are taken to be stable. In bagging we do random perturbations. Generally SVM's are very powerful classifiers. Bagged decision trees are simple and are better than SVM's at some point.

Bag Tree(Bag Stump):-

Stump is prematurely stopped tree. Tree grows big and then gets pruned. We take a random attribute and split and in this way we form a stump. For different bags we get different stumps which we aggregate. This is called as a random forest.

2 Boosting

In boosting we take a series of weak classifiers and improve their performance. A weak classifier is one whose error rate is only slightly better than random guessing. It is a series not a collection. We generate data sets in a sequence or series or a specific fashion where we keep reducing the error. Let the output be from +1,-1.

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$$

Boosting falls in the class of Forward Stagewise Additive models. We try to minimize the loss.

$$\min_{(\beta_m,\gamma_m)_{m=1\ldots M}} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta_m b(x_i,\gamma_m)) \; and \; \forall_m \beta_m > 0$$

where $b(x, \gamma_m)$ is a classifier.

Here we take all the classifiers to be same just for convenience of analysis. Practically we can use different types of classifiers. We will solve the above minimization problem one by one (β, γ) at a time. At every point 'm' we will be fitting the residual left by the previous 'm-1' classifiers.
So now our objective function can be rewritten as

$$\min_{\beta,\gamma} \; L(y_i, f_{m-1}(x) + \beta b(x_i, \gamma))$$

from this we get β_m, γ_m. Consider the case where the loss function is a squared loss.

$$\begin{aligned} L &= \sum_{i=1}^{N} [(y_i - f_{m-1}(x) - \beta b(x_i,\gamma))^2] \\ &= \sum_{i=1}^{N} [(r_{im} - \beta b(x_i,\gamma))^2] \end{aligned}$$

where r_{im} is the residual at m^{th} stage.
We use some different loss function for classification and it is called as adaboost.

Adaboost:-

Adaboost known as adaptive boosting is used for classification. Here loss function is a exponential loss function.

$$L(y, f(x)) = exp(-yf(x))$$

This is very close to logistic loss function which is $log(1 + e^{-yf(x)})$. Minimization of exponential loss is same as that of minimization of logistic loss.
Weakness is important here because we are trying to fit the residual. So if weak classifier is used then we will have some residual to fit and then we can improve in stages or else we can't.

$$\begin{aligned} G_m(x) &= b(x, \gamma_m) \\ (\beta_m, G_m) &= argmin_{\beta,G} \sum_{i=1}^{N} exp[-y_i(f_{m-1}(x_i) + \beta G(x_i))] \end{aligned}$$

$y_i f_{m-1}(x_i)$ does not have any dependence on β and G_m. So

$$(\beta_m, G_m) = \operatorname*{argmin}_{\beta,G} \sum_{i=1}^{N} exp[-y_i f_{m-1}(x_i)] exp[-y_i \beta G(x_i)]$$

$$= \operatorname*{argmin}_{\beta,G} \sum_{i=1}^{N} w_i^{(m)} exp[-y_i \beta G(x_i)]$$

where $exp[-y_i f_{m-1}(x_i)]$ is the weight of the datapoint x_i in the m^{th} stage.

Suppose G(x_i) can get only one data point correct then which should it be?
It should be the one with maximum weight from the previous stage. We have to see that the points we classify correctly should have higher weight and the points that we classify incorrectly should have lower weight. Minimizing the weights for incorrectly classified points and maximizing the weights for the correctly classified points are same. So we try to minimize weights for incorrectly classified points.

$$G_m = \operatorname*{argmin}_{G} \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i))$$

when $y_i = G(x_i)$ then loss is $e^{-y_i \beta G(x_i)} = e^{-\beta}$ and
when $y_i \neq G(x_i)$ then loss is $e^{-y_i \beta G(x_i)} = e^{\beta}$
So the loss is

$$\sum_{i=1}^{N} w_i^{(m)} [e^{-\beta} I(y_i = G(x_i)) \; + \; e^{\beta} I(y_i \neq G(x_i))]$$

and we want $I(y_i \neq G(x_i)) = 1$ for which weights are small. Now to find β we minimize the weighted error

$$E \; = \; e^{-\beta} \sum w_i^{(m)} \; + \; e^{\beta} \sum w_i^{(m)}$$

$$\frac{\partial E}{\partial \beta} \; = \; -w_c e^{-\beta} \; + \; w_e e^{\beta}$$

Now let sum of weights for which $y_i = G(x_i)$ be w_c and the sum of the weights for which $y_i \neq G(x_i)$ be w_e.
Equating the derivative to 0 we get

$$\beta_m \; = \; \frac{1}{2} ln(\frac{w_c}{w_e})$$

$$= \frac{1}{2} ln(\frac{w - w_e}{w_e})$$

$$= \frac{1}{2} ln(\frac{1 - err_m}{err_m})$$

where $err_m \; = \; \frac{w_e}{w}$
Putting all this together

$$f_m(x) \; = \; f_{m-1}(x) \; + \; \beta_m G_m(x)$$

$$w_i^{(m+1)} \; = \; w_i e^{-\beta_m y_i G_m(x_i)}$$

Data point for which lot of classifiers have difficulty to classify have higher weight and vice versa. So we are trying to get the point right for which most of the previous classifiers done wrong. We are going to resample the data set but the probability of point getting into the dataset

$$\text{p}(x_i \text{ sampled in stage m}) = \frac{w_i^{(m)}}{w^{(m)}}$$

Adaboost typically doesn't suffer from over fitting. Both errors testing and training decrease with adaboost. But after some time the decrease will be very less.

If classifier that we take is not weak then the weights will be very low resulting in very smaller probabilities for all points as strong classifier can classify most of the points correctly. So if we sample then we get same dataset again. So we do not get any advantage.

3 Stacking

Stacking involves training a learning algorithm to combine the predictions of several other learning algorithms. Suppose if we use different types of classifiers on same data then the errors will be correlated very less. Then combine them by voting. These type of methods are called as committee machines.

$$\hat{f}(x_i) = \sum_{m=1}^{M} w_m \hat{f_m}(x_i)$$

and now we find the weights by

$$\hat{w} = \underset{w}{\text{argmin}} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))$$

This is like converting x_i to another 'm' dimensional input space. Here all features will be '+1' or '-1'. This is called as stacking. The weights can even be dependent on the inputs x_i also.

References

[1] Notes taken during class

[2] T. Hastie, R. Tibshirani and J. Freidman, The Elements of Statistical Learning,

[3] Wikipedia