UNIVERSITY OF CALIFORNIA, BERKELEY

Lecture 24: CS - 189

Oscar Ortega

July 16, 2021

1 ADABOOST

AdaBoost("adaptive boosting") is an ensemble method for classification (or regression) that

- trains multiple learners on **weighted** sample points
- uses different weight for each learner
- increases weight of mis-classified sample points
- gives bigger votes to more accurate learners

Input: $n \times d$ design matrix X, vector of labels $y \in \mathbf{R}^n$ with $y_i \in \{-1, +1\}$ Ideas:

- train T classifiers $G_1, ..., G_\tau$
- Weight for sample pt $X_i \in G_\tau$ grows according to how many of the classifiers misclassified it.
- Train G_t to try harder to correctly classify
- Meta learner is a linear combination of learners. For test point z,

$$M(z) = \sum_{\tau \in T} B_\tau G_\tau(z)$$

- Each G_t is either + or - 1, but M is continuous. The label predicted of a test point z corresponds to the sign of $M(z)$ In iteration t, what classifier G_t and coefficient B_t should

we choose?
Pick a loss fn L(prediction, label). Find G_t, β_t that minimize.

$$\textbf{Risk} = \frac{1}{n}\sum_{i=1}^{n} L(M(x_i), y_i), M(X_i) = \sum_{t=1}^{\tau} B_t G_t(X_i)$$

AdaBoost metalearner uses **exponential loss fn**

$$L(p,l) = e^{-pl} = \begin{cases} e^{-p} & l = 1 \\ e^{p} & l = -1 \end{cases}$$

Important: label l is binary, G_t is binary, but $p = M(x_i)$ is continuous.

$$n\text{Risk} = \sum_i L(M(x_i), y_i) = \sum_i e^{-y_i} M(x_i) \tag{1.1}$$

$$= e^{-b_t} \sum_i w_i^{(t)} + (e^{-b_t} - e^{-b_t}) \sum_{y_i \neq G_t} (X_i)\, w_i^{(t)} \tag{1.2}$$

$$\tag{1.3}$$

The learner that minimizes the risk is the learner that minimizes the sum of $w_i^{(T)}$ where

$$w_i^{(t)} = \prod_{t=1}^{T-1} e^{-b_t y_t G_t(X_i)}$$

Recursive definition of weights:

$$w_i^{T+1} = w_i^T e^{-b_t y_i G_t(X_i)}$$

again, recall that the functions G are binary and only output a plus or a minus one. When we classify correctly, the weight corresponding to that point will become smaller and the opposite will also hold.
To choose b_t, set its corresponding derivative with respect to the Risk function equal to zero.

$$\text{err}_t = \frac{\sum_{y_i \neq G_t(X_i)} w_i^{(t)}}{\sum_i w_i^{(t)}}$$

$$b_t = \frac{1}{2} ln(\frac{1 - \text{err}_t}{\text{err}_t})$$

If the error is equal to zero note how weight b_t is driven to an infinite value. and if the error was 1/2, this would weight down the corresponding weight to 0

1.1 AdaBoost Alg:

1. Initialize weights $w_i = \frac{1}{n}, \forall i$
2. for $t \geq 1$ to

- Train G_t with weights w_i
- Compute weighted error rate err
- coefficient b_t is computed
- reweight points: $w_i = \{w_i e^{b_t}$ misclassfied, else$w_i^{-b_t}\}$

return meta learner based on a linear combination of the individual learners.

Why boost decision trees? Why short trees?

- fast
- No hyper-parameter search
- easy to make a tree beat 55 percent training accuracy consistently
- easy bias-variance control. Boosting can overfit
- Adaboost trees are usually short to reduce over-fitting
- AdaBoost + short trees is a form of subset selection.
- Linear classifiers do not boost well.

More about AdaBoost:

- Posterior Probability can be approximated: $P(Y = 1|x) \approx \frac{1}{1+e^{-2M(x)}}$
- if every learner has $\geq \mu : \mu > 50$ percent, then the meta-learner training accuracy will be 100 percent.

2 Nearest Neighbor Classification

Idea: Given query point q, find the k sample pts nearest q.
Can use any distance metric
Take note that this can be used for either regression or classification:
Regression: we would want to return the average value of the labels
Classification: we would return either a histogram of class probabilities or simply return the class with the most votes from the k closest points.

2.1 Theorom: Cover and Hart, 1967

As $n \to \infty$, the 1-NN error rate $< B(2-B)$, where B = BayesRisk If only two classes. $\leq 2B(1-B)$

2.2 Theorom: Fix and Hodges, 1951

As $n \to \infty, k \to \infty, \frac{k}{n} \to 0$, k-NN error rate converges to B.