Ensemble Learning

Carlos Valle

Departamento de Informática
Universidad Técnica Federico Santa María

cvalle@inf.utfsm.cl

November 22, 2015

Overview

Ensemble Learning

- Ensemble learning is the discipline which studies the use of a committee of models $\{f_1, f_2, \ldots, f_N\}$ to construct a joint predictor which improves the performance over a single more complex model.
- Building a decision function F by means of an aggregation operator that combines the individual decisions instead of carefully designing the complete map between $\mathcal{X}$ and $\mathcal{Y}$ in a single step.
- Its key assumption is that a predictor with higher accuracy can be constructed by combining models, instead of using one model of higher complexity.

Ensemble Learning (2)

- Thus, the weighted average of a collection of models could improve the performance of predicting a dataset as a linear combination of them, it could generate a lower bias than any of the individual predictors (even being able to choose the best individual predictor)
- It seems intuitive that when combining predictors it is necessary that they have some degree of heterogeneity or diversity.
- Ensemble methods can differ in three different stages: manipulating data, manipulating the base learner, and the function that the ensemble uses to generate the "consensus" output.

Ambiguity decomposition

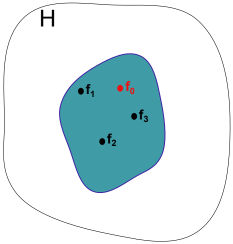
- Ensemble algorithms can also be motivated by the ambiguity decomposition

$$(y - F(\mathbf{x}))^2 = \sum_{n=1}^{N} w_n (y - f_n(\mathbf{x}))^2 - \sum_{n=1}^{N} w_n (f_n(\mathbf{x}) - F(\mathbf{x}))^2. \quad (1)$$

- It states that the quadratic loss of the ensemble is the weight average of the individual errors minus the weighted average of the individual deviations respect to the ensemble output.
- This suggests that the ensemble error can be reduced by increasing the second term which is called the *ambiguity* term.
- However, uncontrolled deviations can lead to increase the first term in (1), hence, increasing the ensemble error.

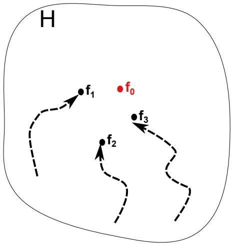
Why ensemble?

- Statistical: As we have seen the selected single hypothesis $f \in \mathcal{H}$ depends on the instances which fed the learning algorithm.
- We can create a set of classifiers with zero training error, and therefore, all classifiers will become indistinguishable.
- We can't be sure how close is each hypothesis (in black) to the optimum (in red), since we do not know the exact probability function underlying the data.
- We can minimize the risk of choosing a wrong individual classifier by constructing an ensemble and "averaging" the individual predictions.



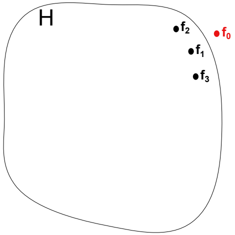
Why ensemble? (2)

- Computational: Some training methods such as hill-climbing or gradient descent may fall into local optima, and therefore, the initial parameters of each machine can affect the choice of the final hypothesis, obtaining a solution that is far from the optimal classifier (red).
- Once again, we expect that averaging the individual functions and ensemble can get closer to the unknown optimum.



Why ensemble? (3)

- Representational: The hypothesis space of the classifiers used to approximate the underlying function f_0 to the data may not contain the optimal classifier.
- Some base learners, such as neural networks, using finite training sample select the first hypothesis that fits the training data.
- Thus the effective space of hypotheses $\mathcal{H}'$ by the learner is quite smaller than the hypotheses space $\mathcal{H}$.



Wisdom of the crowds

- The expected performance of the combined predictor can be better than the average performance of individual predictors, even if each of them is weakly good.
- Each learner needs to perform better than random guessing, i.e., their error-rate $\varepsilon(f_n)$ must be lower than 0.5. The classifiers which meet this condition are known as weak learners.
- Suppose each binary classifier f_n is a weak learner, i.e, its error-rate $\varepsilon(f_n) < 0.5$, $n = 1, \dots, N$.
- Let's define the ensemble output $F(\mathbf{x})$ for a single example $\mathbf{x}$

$$F(\mathbf{x}) = \sum_{n=1}^{N} f_n(\mathbf{x})$$

Wisdom of the crowds (2)

- If the individual errors are independent and let $C_1(\mathbf{x}) = \sum_{n=1}^{N} I(f_n(\mathbf{x}) = 1)$ be the consensus vote for class 1.
- Then, $C_1(\mathbf{x}) \sim \text{Bin}(N, 1-e)$, where $e = \max_{n=1}^{N}\{\varepsilon(f_n)\}$.
- Thus, $P(C_1 > N/2)$ converges to 1 when N gets large.
- However, this result is not extendable to multi-class problems.
- In the business world this concept has become popular as "The Wisdom of Crowds" (Surowiecki, 2005) which states that the collective decisions (using average or voting) are often better than any single individual.

How to introduce diversity?

- Data Manipulation
  - Resampling
  - Manipulating the targets
  - Subsampling (Horizontal / Vertical partitioning)
- Manipulating the base learner
  - Starting point in hypothesis space
  - Manipulating the inducers parameters
  - Adding a regularization term
  - Combining different base learners
- Aggregation function
  - Voting
  - Linear combination (fixed or optimized)

Bagging

- Introduced by Breiman trains each learner using a bootstrap sample from the training set S and combines each output by uniform averaging.
- This intrinsically parallel technique can be viewed as an algorithm with only an implicit search for diversity,

Algorithm 1 Bagging Algorithm

```
1: **Input:** Training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
2: **for** $n \leftarrow 1, \ldots, N$ **do**
3:     $S^n \leftarrow$ Generate a bootstrap sample with replacement from $S$.
4:     $f_n \leftarrow \mathcal{A}(S^n)$ .
5: **end for**
6: For any new example $\mathbf{x}$, define the ensemble hypothesis as
```

$$F(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} f_n(\mathbf{x}).$$

Why Bagging works?

- Breiman: "Averaging reduces variance and leaves bias unchanged."
- Let $f^{b*}(\mathbf{x})$ the learner trained with a bootstrap sample.
- Then $F(\mathbf{x}) = E[f^{b*}(\mathbf{x})]$
- The quadratic error of a single predictor trained with a bootstrap sample is

$$\begin{aligned}
E\left[(Y - f^{b*}(\mathbf{x}))^2\right] &= E\left[(Y - F(\mathbf{x}) + F(\mathbf{x}) - f^{b*}(\mathbf{x}))^2\right] \\
&= E\left[(Y - F(\mathbf{x}))^2\right] + E\left[(F(\mathbf{x}) - f^{b*}(\mathbf{x}))^2\right] \\
&\geq E\left[(Y - F(\mathbf{x}))^2\right]
\end{aligned}$$

Why Bagging works? (2)

- Grandvalet (2004): "Bagging equalizes the influence of the examples."
- "Common" points appear in groups. Remove one, doesn't change drastically the model.
- When $M \to \infty$, the probability of removing a group of size k.

$$\left(1 - \frac{k}{M}\right)^M \to e^{-k}.$$

- This probability decays exponentially with k. For $k = 1$, $(1 - 1/M)^M \sim 0.37$,
- while for $k = 3$, $(1 - 3/M)^M \sim 0.05$.
- The probability of a point appearing in a bootstrap sample converges ≈ 0.632.
- Outliers are those points whose removal of the training set is highly unstabilized.

Adaboost

- The gist of this model is to create a method that combines a group of weak learners to produce a strong learning algorithm.
- At each round of the algorithm the weights of the incorrectly classified examples are increased, in such a way that in the following step the weak learner is forced to focus on the misclassified examples of the training set.
- Thus, the algorithm concentrates on the "hard" examples, instead of on the correctly classified examples.
- In normal scenarios, this approach usually outperforms Bagging, however, under noisy data the robust performance of Bagging is commonly superior rather than Adaboost which considerably decreases its generalization capacity.

Adaboost (2)

Algorithm 2 Adaboost.M1 Algorithm

```
1: Initialize the weights $w_m^1 \leftarrow \frac{1}{M}, m = 1, \ldots, M$
2: for $n \leftarrow 1, \ldots, N$ do
3:     Train the classifier $f_n \in \{-1, 1\}$ under the weight distribution $w_m^n$.
4:     Compute
           $$\alpha_n \leftarrow \frac{\sum_{m=1}^{M} w_m^n I(y_m \neq f_n(\mathbf{x}_m))}{\sum_{m=1}^{M} w_m^n}.$$
5:     Compute the machine's *confidence*
           $$\beta_n \leftarrow \frac{1}{2} \log \left( \frac{1 - \alpha_n}{\alpha_n} \right).$$
6:     Update the weights $w_m^{n+1} \leftarrow w_m^n \exp(2\beta_n I(y_m \neq f_n(\mathbf{x}_m)))$.
7: end for
8: For any new example $\mathbf{x}$, define the ensemble hypothesis as
           $$F(\mathbf{x}) = \text{sign} \left( \frac{1}{N} \sum_{n=1}^{N} \beta_n f_n(\mathbf{x}) \right).$$
```

Adaboost (3)

- In the n-th iteration the aim is to minimize:

$$(\beta_n, f_n) = \text{argmin}_{\beta,f} \sum_{m=1}^{M} \exp(-y_m F(\mathbf{x}_m))$$

- Note that the previous learners have been already trained. Then, we can arrange this equation as

$$\begin{aligned}(\beta_n, f_n) &= \text{argmin}_{\beta,f} \sum_{m=1}^{M} \exp(-y_m F(\mathbf{x}_m)) \\ &= \text{argmin}_{\beta,f} \sum_{m=1}^{M} w_m^n \exp(-y_m F(\mathbf{x}_m))\end{aligned}$$

- Where $w_m^n = \exp(-y_m F_{n-1}(\mathbf{x}_m))$

Adaboost (4)

- Considering the correctly and incorrectly classified examples we have,

$$(\beta_n, f_n) = \text{argmin}_{\beta,f} e^{-\beta} \sum_{y_m = f_n(\mathbf{x}_m)} w_m^n + e^{\beta} \sum_{y_m \neq f_n(\mathbf{x}_m)} w_m^n$$

- Adding $e^{-\beta} \sum_{y_m \neq f_n(\mathbf{x}_m)} w_m^n - e^{-\beta} \sum_{y_m \neq f_n(\mathbf{x}_m)} w_m^n$
- This is equivalent to

$$(\beta_n, f_n) = \text{argmin}_{\beta,f} e^{-\beta} \sum_{m=1}^{M} w_m^n + \left(e^{\beta} - e^{-\beta}\right) \sum_{m=1}^{M} w_m^n I(y_m \neq f_n(\mathbf{x}))$$

- Minimizing w.r.t. f_n means to choose the learner which minimizes the step 4 in Algorithm 2.

Adaboost (5)

- Minimizing respect to β we have

$$\begin{aligned}
\sum_{m=1}^{M} w_m^n I(y_m \neq f_n(\mathbf{x}_m)) \left(e^{\beta} - e^{-\beta}\right) - e^{-\beta} \sum_{m=1}^{M} w_m^n &= 0 \\
\sum_{m=1}^{M} w_m^n I(y_m \neq f_n(\mathbf{x}_m)) e^{\beta} - \sum_{m=1}^{M} w_m^n I(y_m \neq f_n(\mathbf{x}_m)) e^{-\beta} - e^{-\beta} \sum_{m=1}^{M} w_m^n &= 0
\end{aligned}$$

- Now we divide by $e^{-\beta}$ and consider that $\sum_{m=1}^{M} w_m^n = 1$

$$e^{2\beta} = \frac{1 - \alpha_n}{\alpha_n}$$

- This implies to compute β as in step 5.

Adaboost (6)

- We stated that $w_m^n = \exp(-y_m F_{n-1}(\mathbf{x}_m))$ and the ensemble is updated

$$F_n = F_{n-1} + \beta_n f_n$$

- Thus, the weights in the next iteration are

$$w_{m+1}^n = w_m^n \exp(-\beta_n f_n(\mathbf{x}_m) y_m))$$

- Considering that $f_n(\mathbf{x}_m) y_m = 1 - 2I(y_m \neq f_n(\mathbf{x}_m))$

$$w_{m+1}^n = w_m^n \exp(-\beta_n/2) \exp(\beta_n I(y_m \neq f_n(\mathbf{x}_m)))$$

What is Adaboost doing?

- The weight for the new classifier f_n added to the ensemble is

$$\beta_n = \frac{1}{2}\log\left(\frac{1-\alpha_n}{\alpha_n}\right) = \frac{1}{2}\log\left(\frac{d_+}{d_-}\right),$$

- where $d_- = \sum_{y_m \neq f_n(\mathbf{x}_m)} w_m^n$ and $d_+ = \sum_{y_m = f_n(\mathbf{x}_m)} w_m^n = 1 - d_-$
- $r_n = \sum_{m=1} M w_m^n f_n(\mathbf{x}_m) y_m$be the edge of the n-th weak classifier.
- Thus, $r_n = d_+ - d_- = 1 - 2d_-$.
- Note that $d_+ = (1 + r_n)/2$, $d_- = (1 - r_n)/2$
- Then,

$$\beta_n = \frac{1}{2}\log\left(\frac{1+r_n}{1-r_n}\right) = \tanh^{-1}(r_n)$$

What is Adaboost doing? (2)

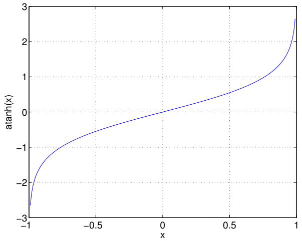
- We can see Adaboost as a coordinate descent algorithm for minimizing

$$\sum_{m=1}^{M} \exp\left(-y_m F(\mathbf{x}_m)\right)$$

- At each iteration we follow a line search along the direction of β_n

$$F_n = F_{n-1} + \beta_n f_n$$

- Note that the weak learner condition over the weight distribution implies $\beta_n > 0$



Minimizing the conditional expectation

- Friedman et al. show that Adaboost minimizes the conditional expectation $E\left[e^{-yF(\mathbf{x})}|\mathbf{x}\right]$ at

$$F(\mathbf{x}) = \frac{1}{2}\log\frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})}.$$

- Considering that $P(y=1|\mathbf{x}) = 1 - P(y=-1|\mathbf{x})$ we can compute the conditional probabilities for both labels as

$$\begin{aligned} P(y=1|\mathbf{x}) &= \frac{e^{F(\mathbf{x})}}{e^{-F(\mathbf{x})}+e^{F(\mathbf{x})}}, \\ P(y=-1|\mathbf{x}) &= \frac{e^{-F(\mathbf{x})}}{e^{-F(\mathbf{x})}+e^{F(\mathbf{x})}}. \end{aligned}$$

Minimizing the conditional expectation (2)

- Hence, the conditional expectation $E\left[e^{-yF(\mathbf{x})}|\mathbf{x}\right]$ can be computed as

$$E\left[e^{-yF(\mathbf{x})}|\mathbf{x}\right] = P(y=1|\mathbf{x})e^{-F(\mathbf{x})} + P(y=-1|\mathbf{x})e^{F(\mathbf{x})} \qquad (2)$$

- Minimizing (2) respect to the ensemble we have

$$\frac{\partial E\left[e^{-yF(\mathbf{x})}|\mathbf{x}\right]}{\partial F(\mathbf{x})} = -P(y=1|\mathbf{x})e^{-F(\mathbf{x})} + P(y=-1|\mathbf{x})e^{F(\mathbf{x})} = 0$$

- By multipliying both numerator and denominator by $e^{F(\mathbf{x})}$in (2) we obtain

$$P(y=1|\mathbf{x}) = \frac{e^{2F(\mathbf{x})}}{1+e^{2F(\mathbf{x})}},$$

which is the usual logistic model up to a factor 2.

Real Adaboost

- They proposed the Real Adaboost algorithm which considers base learners with real-valued contributions
- This method uses the additive logistic model to estimate the conditional expectations. In order to show this, we rewrite the conditional expectation for an ensemble of n predictors

$$\begin{aligned} E\left[e^{-yF_n(\mathbf{x})}|\mathbf{x}\right] &= E\left[e^{-y(F_{n-1}(\mathbf{x})+f_n(\mathbf{x}))}|\mathbf{x}\right] \\ &= e^{-f_n(\mathbf{x})}E\left[e^{-yF_n(\mathbf{x})}1_{[y=1]}|\mathbf{x}\right] + e^{f_n(\mathbf{x})}E\left[e^{-yF_n(\mathbf{x})}1_{[y=-1]}|\mathbf{x}\right] \end{aligned}$$

- The contribution of the last predictor is measured dividing by the conditional expectation of the previous ensemble $E\left[e^{-yF_{n-1}(\mathbf{x})}|\mathbf{x}\right]$, this is

$$\frac{E\left[e^{-yF_n(\mathbf{x})}|\mathbf{x}\right]}{E\left[e^{-yF_{n-1}(\mathbf{x})}|\mathbf{x}\right]} = e^{-f_n(\mathbf{x})}E_{w^n}\left[1_{[y=1]}|\mathbf{x}\right] + e^{f_n(\mathbf{x})}E_{w^n}\left[1_{[y=-1]}|\mathbf{x}\right]. \tag{3}$$

Real Adaboost (2)

Algorithm 3 Real Adaboost Algorithm

```
1: Initialize the weights $w_m^1 \leftarrow \frac{1}{M}$, $m = 1, \dots, M$
2: for $n \leftarrow 1, \dots, N$ do
3:     Train the classifier $f_n$ using the learner $\mathcal{A}(S^M)$ under the weight distribution $w^n$.
4:     Estimate
           $$p_n(\mathbf{x}_m) = \frac{e^{f_n(\mathbf{x}_m)}}{e^{f_n(\mathbf{x}_m)} + e^{-f_n(\mathbf{x}_m)}}.$$
5:     Compute the real-valued contributions
           $$g_n(\mathbf{x}_m) \leftarrow \frac{1}{2} \log \left( \frac{p_n(\mathbf{x}_m)}{1 - p_n(\mathbf{x}_m)} \right).$$
6:     Update the weights
           $$w_m^{n+1} \leftarrow [w_m^n \exp(-g_n(\mathbf{x}_m) y_m)]/Z_n,$$
   where $Z_n$ is the normalization factor.
7: end for
8: For any new example $\mathbf{x}$, define the ensemble hypothesis as
           $$F(\mathbf{x}) = \text{sign}\left( \sum_{n=1}^{N} g_n(\mathbf{x}) \right).$$
```

Real Adaboost (3)

- Minimizing it respect to $f_n(\mathbf{x})$ we obtain

$$f_n(\mathbf{x}) = \frac{1}{2}\log\frac{E_{w^n}\left[1_{[y=1]}|\mathbf{x}\right]}{E_{w^n}\left[1_{[y=-1]}|\mathbf{x}\right]} = \frac{1}{2}\log\frac{P_{w^n}\left(y=1]|\mathbf{x}\right)}{P_{w^n}\left(y=-1|\mathbf{x}\right)},$$

  where $w^n = \exp(-yF_n(\mathbf{x}))$.
  Thus, the update rule for the weights is $w_m^n = w_m^{n-1}\exp(-yf_n(\mathbf{x}))$.
- Log ratios can be numerically unstable, in order to deal with this problem, the Gentle Adaboost (Gentleboost) algorithm was proposed to optimize $E\left[e^{-yF_n(\mathbf{x})}\right]$ by Newton stepping. Instead of minimizing the overall test error greedily as the RealAdabost, Gentle Adaboost minimizes the error by using a bounded step size.

Gentle Adaboost

Algorithm 4 Gentle Adaboost Algorithm

```
1: Initialize the weights $w_m^1 \leftarrow \frac{1}{M}$, $m = 1, \ldots, M$.
2: for $n \leftarrow 1, \ldots, N$ do
3:     $f_n \leftarrow$ Train the learner $\mathcal{A}(S^M)$ under the weight distribution $w_m^n$.
4:     Update the weights
           $$w_m^{n+1} \leftarrow w_m^n \exp(-y_m f_n(\mathbf{x}_m))/Z_n,$$
       where $Z_n$ is the normalization factor.
5: end for
6: For any new example $\mathbf{x}$, define the ensemble hypothesis as
           $$F(\mathbf{x}) = \text{sign}\left(\sum_{t=n}^{N} f_n(\mathbf{x})\right)$$
```

Gentle Adaboost (2)

- The first derivative of this expected value at $f(\mathbf{x})$ is

$$\left.\frac{\partial E\left[e^{-yF_n(\mathbf{x})}|\mathbf{x}\right]}{\partial f_n(\mathbf{x})}\right|_{f_n(\mathbf{x})=0} = -E\left[e^{-yF_n(\mathbf{x})}y|\mathbf{x}\right].$$

- Since $y^2 = 1$ the second derivative is

$$\left.\frac{\partial^2 E\left[e^{-yF_n(\mathbf{x})}|\mathbf{x}\right]}{\partial f_n(\mathbf{x})^2}\right|_{f_n(\mathbf{x})=0} = E\left[e^{-yF_n(\mathbf{x})}y^2|\mathbf{x}\right] = E\left[e^{-yF_n(\mathbf{x})}|\mathbf{x}\right].$$

- Therefore, the ensemble updating rule is

$$\begin{aligned} F_n(\mathbf{x}) &= F_{n-1}(\mathbf{x}) + \frac{E\left[e^{-yF_n(\mathbf{x})}y|\mathbf{x}\right]}{E\left[e^{-yF_n(\mathbf{x})}|\mathbf{x}\right]} \\ &= F_{n-1}(\mathbf{x}) + E_w[y|\mathbf{x}], \end{aligned}$$

- where $w^n = \exp(-yF_n(\mathbf{x}))$.

Gentle Adaboost (3)

- In this algorithm the update is computed by using $E_w[y|\mathbf{x}] = P_w(y=1|\mathbf{x}) - P_w(y=-1|\mathbf{x}) \in [-1,1]$ rather than the log ratio used in Real AdaBoost which can lead to very large update amounts when the ratio tends to zero.

Any questions?