Chapter 6. Ensemble Methods

Wei Pan

Division of Biostatistics, School of Public Health, University of Minnesota, Minneapolis, MN 55455
Email: weip@biostat.umn.edu

PubH 7475/8475
©Wei Pan
Have a base learner/algorithm; use multiple versions of it to form a final classifier (or regression model). Goal: improve over the base/weaker learner (and others). Often the base learner is a simple tree (e.g. stump).

Include Bagging (§8.7), boosting (Chapter 10), random forest (Chapter 15). Others: Bayesian model averaging (Chapter 8); Model averaging and stacking (§8.8); ARM (Yang, JASA), ...
Bagging

- Bootstrap Aggregation (Bagging) (§8.7).
- Training data: \( D = \{(X_i, Y_i) | i = 1, \ldots, n\} \).
- A bootstrap sample is a random sample of \( D \) with size \( n \) and **with** replacement.
- Bagging regression:
  1) Draw \( B \) bootstrap samples \( D_1^*, \ldots, D_B^* \);
  2. Fit a (base) model \( f_b^*(x) \) with \( D_b^* \) for each \( b = 1, \ldots, B \);
  3. The bagging estimate is \( \hat{f}_B(x) = \frac{\sum_{b=1}^{B} f_b^*(x)}{B} \).
- If \( f(x) \) is linear, then \( \hat{f}_B(x) \to \hat{f}(x) \) as \( B \to \infty \); but not in general.
- A surprise (Breiman 1996): \( \hat{f}_B(x) \) can be much better than \( \hat{f}(x) \), especially so if the base learner is not stable (e.g. tree).
Classification: same as regression but
1) $\hat{G}_B(x) = \text{majority of } (\hat{G}_1^*(x), ..., \hat{G}_B^*(x))$; or
2) if $\hat{f}(x) = (\hat{\pi}_1, ..., \hat{\pi}_K)'$, then
$\hat{f}_B(x) = \sum_{b=1}^{B} f_b^*(x)/B$, $\hat{G}_B(x) = \arg\max_k \hat{f}_B(x)$. 2) may be better than 1);

Example: Fig 8.9, Fig 8.10.

Why does bagging work? to reduce the variance of the base learner.
but not always, while always increases bias! (Buja) explains why sometimes not the best.
FIGURE 8.9. Bagging trees on simulated dataset.
FIGURE 8.10. Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.
Bayesian Model Averaging

- Suppose we have \( M \) models \( M_m \), \( m = 1, \ldots, M \).
- Suppose \( \xi \) is parameter of interest: given training data \( Z \),

\[
Pr(\xi|Z) = \sum_{m=1}^{M} Pr(\xi|M_m, Z) Pr(M_m|Z),
\]

\[
E(\xi|Z) = \sum_{m=1}^{M} E(\xi|M_m, Z) Pr(M_m|Z).
\]

- Need to specify models, ..., complex!

\[
Pr(M_m|Z) \propto Pr(M_m) Pr(Z|M_m)
= Pr(M_m) \int Pr(Z|M_m, \theta_m) Pr(\theta_m|M_m) d\theta_m.
\]
An approximation:

\[
BIC(\mathcal{M}_m) = \log \Pr(Z|\mathcal{M}_m, \hat{\theta}_m(Z)) - \log(n)p/2 \\
\approx \propto \log \Pr(\mathcal{M}_m|Z).
\]

hence, use weights \( \propto \exp[BIC(\mathcal{M}_m)] \).


\[
AIC(\mathcal{M}_m) = \log \Pr(Z|\mathcal{M}_m, \hat{\theta}_m(Z)) - p \\
\approx E_{Z^*} \log \Pr(Z^*|\mathcal{M}_m, \hat{\theta}_m(Z)).
\]

ARM (Yang 2001): use sample-splitting (or CV),

\[
\log \Pr(Z^{ts}|\mathcal{M}_m, \hat{\theta}_m(Z^{tr})).
\]
Stacking

- §8.8; Wolpert (1992, Neural Networks), Breiman (1996, ML).
- $\hat{f}(x) = \sum_{m=1}^{M} w_m \hat{f}_m(x), \ w = (w_1, \ldots, w_M)'$.
- Ideally, if $P$ is the distr for $(X, Y)$,

$$\hat{w} = \arg \min_w E_P[Y - \sum_{m=1}^{M} w_m \hat{f}_m(X)]^2.$$ 

- But $P$ is unknown, use its empirical distr:

$$\hat{w} = \arg \min_w \sum_{i=1}^{n} [Y_i - \sum_{m=1}^{M} w_m \hat{f}_m(X_i)]^2.$$ 

Good? why? think about best subset selection ...

- Stacking: $\hat{f}_{m}^{-i}$: $f_m$ fitted without $(X_i, Y_i)$; LOOCV.

$$\hat{w}^{st} = \arg \min_w \sum_{i=1}^{n} [Y_i - \sum_{m=1}^{M} w_m \hat{f}_{m}^{-i}(X_i)]^2.$$ 

- How? OLS; but QP if impose $\hat{w}^{st} \geq 0$ and $\sum_{m=1}^{M} w_m^{st} = 1$. 
Adaptive Regression by Mixing

- \( \hat{f}(x) = \sum_{m=1}^{M} w_m \hat{f}_m(x), \ w = (w_1, ..., w_M)' \).
- Key: how to estimate \( w \)?
- ARM:
  1. Partition the data into two parts \( D = D_1 \cup D_2 \);
  2. Use \( D_1 \) to fit the candidate models \( \hat{f}_m(x; \hat{\theta}_m(D_1)) \);
  3. Use \( D_2 \) to estimate weights: \( w_m \propto \prod_{i \in D_2} \hat{f}_m(X_i; \hat{\theta}_m(D_1)) \).

- Note: AIC is asymptotically unbiased for the predictive log-likelihood, so ARM \( \approx \ldots \)?
Other topics

- Model selection vs model mixing (averaging).
  Theory: Yang (2003, Statistica Sinica); Shen & Huang (2006; JASA);
  My summary: if easy, use the former; o/w use the latter.

- Generalize model averaging to input-dependent weighting:
  \[ w_m = w_m(x). \]


- Model selection: AIC or BIC or CV? LOOCV or k-fold CV?
Random Forest

- RF (Chapter 15); by Breiman (2001).

- Main idea: similar to bagging,
  1) use bootstrap samples to generate many trees;
  2) In generating each tree,
     i) at each node, rather than using the best splitting variable among all the predictors, use the best one out of a random subset of predictors (the size $m$ is a tuning parameter to be determined by the user; not too sensitive); $m \sim \sqrt{p}$.
     ii) each tree is grown to the max size; no pruning;
Why do so?
1) Better base trees improve the performance;
2) The correlations among the base trees decrease the performance.
Reducing $m$ decreases the correlations (& performance of a tree).

Output: Give an OOB estimate of the prediction error. Some obs’s will not be in some bootstrap samples and can be treated as test data (for the base trees trained on these bootstrap samples)!

Output: Give a measure of the importance of each predictor.
1) use the original data to get an OOB estimate $e_0$;
2) permute the values of $x_j$ across obs’s, then use the permuted data to get an OOB estimate $e_j$;
3) Importance of $x_j$ is defined as $e_j - e_0$.

RF can handle large datasets, and can do clustering!

Example code: ex6.1.R
FIGURE 15.1. Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each “step” in the figure corresponds to a change in a single misclassification (in a test set of 1536).
FIGURE 15.3. Random forests compared to gradient boosting on the California housing data. The curves represent mean absolute error on the test data as a function of the number of trees in the models. Two random forests are shown, with $m = 2$ and $m = 6$. The two gradient boosted models use a shrinkage parameter $\nu = 0.05$ in (10.41), and have interaction depths of 4 and 6. The boosted models outperform random forests.
FIGURE 15.4. OOB error computed on the spam training data, compared to the test error computed on the test set.
Chapter 10.

AdaBoost: proposed by Freund and Schapire (1997).

Main idea: see Fig 10.1
1. Fit multiple models using weighted samples;
2. Misclassified obs’s are weighted more and more;
3. Combine the multiple models by weighted majority voting.

Training data: \( \{(Y_i, X_i) | i = 1, ..., n\} \) and \( Y_i = \pm 1 \).
**Final Classifier**

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

**FIGURE 10.1.** Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.
Alg 10.1 AdaBoost

1. Initialize $w_i = 1/n$ for $i = 1, \ldots, n$.
2. For $m = 1$ to $M$:
   2.1 Fit a classifier $G_m(x)$ to the training data with weights $w_i$'s;
   2.2 $\text{err}_m = \frac{\sum_{i=1}^{n} w_i I(Y_i \neq G_m(X_i))}{\sum_{i=1}^{n} w_i}$.
   2.3 $\alpha_m = \log[(1 - \text{err}_m) / \text{err}_m]$.
   2.4 Set $w_i \leftarrow w_i \exp[\alpha_m I(Y_i \neq G_m(X_i))]$, $i = 1, \ldots, n$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$. 
Example: use stumps (trees with only two terminal nodes) as the base learner; $X_i$ iid $N_{10}(0, I)$, $Y_i = 1$ if $\|X_i\|^2_2 > \chi^2_{10}(0.5) = 9.34$ and $Y_i = -1$ o/w.

$n_{tr} = 1000 + 1000$, $n_{ts} = 10,000$.

Fig 10.2.

Puzzles:

1) AdaBoost worked really well! why?

3) no over-fitting? even after training error=0, test error still goes down.
FIGURE 10.2. Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.
Forward Stagewise Additive Modeling

- \( f(x) = \sum_{m=1}^{M} \beta_m b_m(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m) \).
  To estimate each \((\beta_m, \gamma_m)\) stagewise (sequentially).

- Algorithm 10.2: FSAM
  1) Initialize \( f_0(x) = 0 \);
  2) For \( m = 1 \) to \( M \):
     2.a) \((\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^{n} L(Y_i, f_{m-1}(x) + \beta b(x; \gamma))\).
     2.b) Set \( f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m) \).

- Exponential loss: \( Y \in \{-1, 1\} \),
  \[ L(Y, f(x)) = \exp(-Yf(x)) \].

- Stat contribution:
  Adaboost = FSAM using the exponential loss function!
  Why important?
FIGURE 10.3. Simulated data, boosting with stumps: misclassification error rate on the training set, and average exponential loss: $(1/N) \sum_{i=1}^{N} \exp(-y_if(x_i))$. After about 250 iterations, the misclassification error is zero, while the exponential loss continues to decrease.
- Why exponential loss?

\[ f^*(x) = \arg \min_{f(x)} E_{Y|x} \exp(-Yf(x)) = \frac{1}{2} \log \frac{Pr(Y = 1|x)}{Pr(Y = -1|x)}. \]

Explain why use \( \text{sign}(\hat{f}(x)) \) to do prediction.

- AdaBoost estimates \( f^*(x) \) stagewisely.

- Other loss functions: Fig 10.4
  Misclassification: \( I(\text{sign}(f) = y) \);
  Squared error: \( (y - f)^2 \);
  Binomial deviance: \( \log[1 + \exp(-2yf)] \);
  Hinge loss (SVM): \( (1 - yf)I(yf < 1) = (1 - yf)_+ \);

- Loss functions for regression: Fig 10.5
FIGURE 10.4. Loss functions for two-class classification. The response is $y = \pm 1$; the prediction is $f$, with class prediction $\text{sign}(f)$. The losses are misclassification: $I(\text{sign}(f) \neq y)$; exponential: $\exp(-yf)$; binomial deviance: $\log(1 + \exp(-2yf))$; squared error: $(y - f)^2$; and support vector: $(1 - yf)^+$ (see Section 12.3). Each function has been scaled so that it passes through the point $(0, 1)$. 
**FIGURE 10.5.** A comparison of three loss functions for regression, plotted as a function of the margin $y - f$. The Huber loss function combines the good properties of squared-error loss near zero and absolute error loss when $|y - f|$ is large.
Boosting trees

- Each \( f_m(x; \gamma) = T(x; \theta) \) is a tree.
- Gradient boosting: Alg 10.3.
  Also called MART; in R package gbm; weka:
  \[ \text{http://www.cs.waikato.ac.nz/ml/weka/index.html} \]
- Can perform better than AdaBoost; Fig 10.9
- **And**, more flexible: can be extended to \( K > 2 \) classes, regression...
  Q: Is it possible to apply a binary classifier to a \( K \)-class problem with \( K > 2 \)?
- Regularization/shrinkage: Fig 10.11
  \[ f_m(x) = f_{m-1}(x) + \gamma T(x; \theta) \text{ with } 0 < \gamma \leq 1. \]
- Relative importance of predictors: Fig 10.6
  how often used in the trees as a splitting var and how much it improves fitting/prediction.
- Example code: ex6.2.R
FIGURE 10.9. Boosting with different sized trees, applied to the example (10.2) used in Figure 10.2. Since the generative model is additive, stumps perform the best. The boosting algorithm used the binomial deviance loss in Algorithm 10.3; shown for comparison is the AdaBoost Algorithm 10.1.
FIGURE 10.11. Test error curves for simulated example (10.2) of Figure 10.9, using gradient boosting (MART). The models were trained using binomial deviance, either stumps or six terminal-node trees, and with or without shrinkage.
Boosting vs Forward Stagewise Reg

- Forward stagewise univar linear reg \( \approx \) Lasso; Alg 3.4, p.86
- “Boosting as a regularized ... classifier.” (Rosset et al 2004).
FIGURE 3.19. Coefficient profiles for the prostate data. The left panel shows incremental forward stage-wise regression with step size $\epsilon = 0.01$. The right panel shows the infinitesimal version $FS_0$ obtained letting $\epsilon \to 0$. This profile was fit by the modification 3.2b to the LAR Algorithm 3.2. In this example the $FS_0$ profiles are monotone, and hence identical to those of lasso and LAR.