Lecture 5
Ensembles – Part 1

(February 6, 2015)

Mu Zhu
University of Waterloo

An Annus Mirabilis

L. Breiman (1996), "Bagging predictors", *Machine Learning* **24**, pp. 123–140.

Y. Freund & R. Schapire (1996), "Experiments with a new boosting algorithm", in *Proceedings of the 13th International Conference on Machine Learning*, pp. 148–156.

- idea (same): instead of one classifier, beneficial to generate many different classifiers (from the same training data) and aggregate over their classification results
- details (different): how to generate; how to aggregate

Breiman (1996)

- suppose $y_i = \pm 1$
  - for direct comparison with Freund & Schapire (1996)
- generation:
  - use weighted training data $\{(\boldsymbol{x}_i, y_i, w_i^{(b)})\}$ to learn $f_b(\boldsymbol{x})$
- aggregation:

$$F(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} f_b(\boldsymbol{x})$$

- weights: independently determined by bootstrap

called Bagging ... for bootstraped aggregating

Freund & Schapire (1996)

- suppose $y_i = \pm 1$
- generation:
  - use weighted training data $\{(\boldsymbol{x}_i, y_i, w_i^{(b)})\}$ to learn $f_b(\boldsymbol{x})$
  - compute

$$\varepsilon_b = \frac{\sum_{i=1}^n w_i^{(b)} I(y_i \neq f_b(\boldsymbol{x}_i))}{\sum_{i=1}^n w_i^{(b)}} \quad \text{and} \quad R_b = \frac{1-\varepsilon_b}{\varepsilon_b}$$

- aggregation:

$$F(\boldsymbol{x}) = \sum_{b=1}^{B} a_b f_b(\boldsymbol{x}) \quad \text{where} \quad a_b \propto \log(R_b)$$

- weights: sequentially determined adaptively (next slide)

Freund & Schapire (1996)

- initially,

$$w_i^{(1)} = \frac{1}{n}, \quad \text{for all} \quad i = 1, 2, ..., n$$

- at iteration b, update

$$w_i^{(b+1)} = \begin{cases} w_i^{(b)} \times R_b, & y_i \neq f_b(\boldsymbol{x}_i); \\ w_i^{(b)}, & y_i = f_b(\boldsymbol{x}_i) \end{cases}$$

adaptively

called AdaBoost ... for adapative boosting

quite a mystery

Dawn of the New Millennium

J. H. Friedman, T. J. Hastie & R. J. Tibshirani (2000), "Additive logistic regression: A statistical view of boosting (with discussion)", *Ann. Stat.* **28**, pp. 337–407.

L. Breiman (2001), "Random forest", *Machine Learning* **45**, pp. 5–32.

Friedman, Hastie & Tibshirani (2000)

AdaBoost amounts to constructing the function

$$F(\boldsymbol{x}) = \sum_{b=1}^{B} a_b f_b(\boldsymbol{x})$$

sequentially by choosing a_b and f_b at each step to minimize the so-called exponential loss, i.e.,

$$(a_b, f_b) \quad = \quad \underset{\substack{a\in\mathbb{R}\\ f\in\mathcal{F}}}{\arg\min} \quad \frac{1}{n}\sum_{i=1}^{n} e^{-y_i[F_{b-1}(\boldsymbol{x}_i)+af(\boldsymbol{x}_i)]},$$

where

$$F_{b-1}(\boldsymbol{x}) = \sum_{b'=0}^{b-1} a_{b'} f_{b'}(\boldsymbol{x}),$$

and $\mathcal{F}$ is a given family of binary classifiers, e.g., classification trees.

Exponential Loss

- if y is coded as $+1$ versus -1 (rather than 1 versus 0), standard logistic regression model

$$\log \frac{\mathbb{P}[y = (+1)|\boldsymbol{x}]}{\mathbb{P}[y = (-1)|\boldsymbol{x}]} = F(\boldsymbol{x})$$

  means the conditional probability of y given $\boldsymbol{x}$ can be expressed as

$$\mathbb{P}(y|\boldsymbol{x}) = \frac{1}{1 + e^{-yF(\boldsymbol{x})}}, \quad \text{for} \quad y \in \{-1, +1\}$$

- big $\mathbb{P}(y|\boldsymbol{x}) \Leftrightarrow$ small $e^{-yF(\boldsymbol{x})}$

- exponential loss, $L[y, F(\boldsymbol{x})] = e^{-yF(\boldsymbol{x})}$, "makes sense"

Weights

- at iteration b, objective function can be written as

$$L(a, f) = \sum_{i=1}^{n} \underbrace{\frac{1}{n}\left[e^{-y_i F_{b-1}(\boldsymbol{x}_i)}\right]}_{w_i^{(b)}} \left[e^{-a y_i f(\boldsymbol{x}_i)}\right],$$

  where

$$w_i^{(b)} \equiv e^{-y_i F_{b-1}(\boldsymbol{x}_i)}$$

  does not depend on either a or f

- setting $w_i^{(1)} = 1/n$ for all $i \Leftrightarrow$ initializing with $F_0(\boldsymbol{x}) = 0$

Alternating Minimization

- given f, the function $L(a, f)$ is minimized at

$$a = \frac{1}{2}\log\underbrace{\frac{1-\varepsilon_b}{\varepsilon_b}}_{R_b}, \quad \varepsilon_b = \frac{\sum_{i=1}^n w_i^{(b)} I(y_i \neq f(\boldsymbol{x}_i))}{\sum_{i=1}^n w_i^{(b)}},$$

  which explains why AdaBoost weights each f_b by a quantity proportional to $\log(R_b)$

- given a, minimizing $L(a, f)$ over $f \in \mathcal{F}$ requires that we fit a classifier f to the training data with each observation $(\boldsymbol{x}_i, y_i)$ being weighted by $w_i^{(b)}$

Updating Weights

$$F_b(\boldsymbol{x}) = F_{b-1}(\boldsymbol{x}) + af(\boldsymbol{x}) \quad \Rightarrow \quad w_i^{(b+1)} = w_i^{(b)} \times e^{-ay_i f(\boldsymbol{x}_i)}$$

$$w_i^{(b+1)} = \begin{cases} w_i^{(b)} \times e^{a}, & \text{if } y_i \neq f(\boldsymbol{x}_i); \\ w_i^{(b)} \times e^{-a}, & \text{if } y_i = f(\boldsymbol{x}_i). \end{cases}$$

inflate wrong ones by e^a; deflate right ones by e^{-a}

$$\Updownarrow$$

inflate wrong ones by $e^{2a} = \frac{1-\varepsilon_b}{\varepsilon_b} = R_b$; leave right ones alone

Details: Minimize $L(\alpha, f)$ Over a

$$f \in \mathcal{F}, \quad f(\boldsymbol{x}_i) = \pm 1, \quad y_i = \pm 1$$

$$\Downarrow$$

$$\underline{\text{either}} \quad y_i f(\boldsymbol{x}_i) = +1 \text{ (right)} \quad \underline{\text{or}} \quad y_i f(\boldsymbol{x}_i) = -1 \text{ (wrong)}$$

$$L(a, f) = \sum_{y_i f(\boldsymbol{x}_i) = +1} w_i e^{-a} + \sum_{y_i f(\boldsymbol{x}_i) = -1} w_i e^{a}$$

$$\frac{\partial}{\partial a} L(a, f) = 0 \quad \Rightarrow \quad a = \frac{1}{2} \log \frac{1 - \varepsilon}{\varepsilon}, \quad \varepsilon = \frac{\sum_{i=1}^{n} w_i I(y_i \neq f(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i}$$

Details: Minimize $L(a, f)$ Over f

$$
\begin{aligned}
L(a,f) &= \sum_{y_i = f(\boldsymbol{x}_i)} w_i e^{-a} + \sum_{y_i \neq f(\boldsymbol{x}_i)} w_i e^{a} \\
&= \sum_{i=1}^{n} w_i e^{-a} - \sum_{y_i \neq f(\boldsymbol{x}_i)} w_i e^{-a} + \sum_{y_i \neq f(\boldsymbol{x}_i)} w_i e^{a} \\
&= \sum_{i=1}^{n} w_i e^{-a} + \left(e^{a} - e^{-a}\right) \sum_{y_i \neq f(\boldsymbol{x}_i)} w_i
\end{aligned}
$$

$$
\text{given } a > 0 \; [e^a > e^{-a}], \quad f = \arg\min_{g \in \mathcal{F}} \sum_{y_i \neq g(\boldsymbol{x}_i)} w_i
$$

$$
\Downarrow
$$

fit classifier to weighted training data, which guarantees $a > 0$

General Framework

1. Pick a functional class, $\mathcal{F}$. (WHERE)
2. Pick a loss function, $L[y, F(\boldsymbol{x})]$. (WHAT)
3. Pick an optimization method. (HOW)
4. Start with $F(\boldsymbol{x}) = 0$, and sequentially build up the function,

$$F(\boldsymbol{x}) = f_1(\boldsymbol{x}) + ... + f_B(\boldsymbol{x}),$$

by repeating the following steps:

(a) apply the chosen optimization method from (3) to find

$$f \quad = \quad \arg\min_{g \in \mathcal{F}} \quad L\left[y, F(\boldsymbol{x}) + g(\boldsymbol{x})\right];$$

(b) update $F \longleftarrow F + f$.

Breiman (2001)

- a random forest (RF) refers to

$$\{f_b(\boldsymbol{x}) \equiv f(\boldsymbol{x};\Theta_b) : \Theta_b \overset{iid}{\sim} \mathcal{P}_\Theta, b = 1, 2, ...\}$$

- each $f(\boldsymbol{x};\Theta_b)$ is a classifier completely parameterized by Θ_b
- meaning of "$\Theta_b \overset{iid}{\sim} \mathcal{P}_\Theta$":
  - each $f(\cdot;\Theta_b)$ is generated independently with an identical set of stochastic mechanisms, denoted collectively by $\mathcal{P}_\Theta$

"The" RF Algorithm by Breiman

Algorithm

(i) (*Bootstrap*) First, draw a bootstrap sample, $\mathbb{D}_*^{(b)}$, of the training data $\mathbb{D}$. Then, use $\mathbb{D}_*^{(b)}$— not $\mathbb{D}$ — to train a classification tree, f_b.

(ii) (*Random Subset*) When building f_b, randomly select a subset of predictors before making *each* split — say, $\mathcal{S}$, $|\mathcal{S}| < d$ — and make the best split over the set $\mathcal{S}$ rather than over all possible predictors.

Stochastic Mechanism $\mathcal{P}_\Theta$

(i) iid sampling from the empirical distribution

(ii) iid sampling from the set $\{1, 2, ..., d\}$

BIG Question

- clearly, bagging is random forest by mechanism (i) alone
- in five years (1996 to 2001), Breiman added mechanism (ii)

$$\Downarrow$$

????? WHY ?????

Breiman's Theorem

$$\varepsilon_{RF} \quad \leq \quad \bar{\rho}\left(\frac{1-s^2}{s^2}\right)$$

- ε_{RF} = prediction error of random forest
- $\bar{\rho}$ = average pairwise correlation between individual classifiers in the forest
- s = average strength of individual classifiers in the forest

will explain for the case of binary classification

Breiman's Theorem

- building block: for an individual classifier, $f(\cdot;\Theta)$, define its margin at $(\boldsymbol{x}, y)$ as [for binary case]

$$m(\Theta;\boldsymbol{x},y) = I(f(\boldsymbol{x};\Theta)=y) - I(f(\boldsymbol{x};\Theta)\neq y),$$

  so that

$$\begin{aligned} m(\Theta;\boldsymbol{x},y) &> 0, \qquad \text{if } f(\cdot;\Theta) \text{ right about } (\boldsymbol{x},y); \\ m(\Theta;\boldsymbol{x},y) &< 0, \qquad \text{if } f(\cdot;\Theta) \text{ wrong about } (\boldsymbol{x},y) \end{aligned}$$

- ε_{RF}, $\bar{\rho}$ and s all defined in terms of probability- and/or moment-transformations of $m(\Theta;\boldsymbol{x},y)$, w.r.t. $\mathcal{P}_{(\boldsymbol{x},y)}$ or $\mathcal{P}_{\Theta}$

- relationship can then be established by well-known inequalities that link $\mathbb{P}(Z>c)$ with $\mathbb{E}(Z)$, $\mathbb{V}\text{ar}(Z)$, etc

Breiman's Theorem: ε_{RF}

- first, generalize the notion of margin to the entire forest
- margin of forest at $(\boldsymbol{x}, y)$ is simply

$$M(\boldsymbol{x}, y) = \mathbb{E}_{\Theta}(m(\Theta; \boldsymbol{x}, y))$$

  or the average margin of all individual classifiers in the forest

- the prediction error of the forest is simply the probability of the forest making incorrect classifications, i.e.,

$$\varepsilon_{RF} = \mathbb{P}_{(\boldsymbol{x},y)}(M(\boldsymbol{x}, y) < 0)$$

Breiman's Theorem: s

- recall positive margin $m(\Theta; \boldsymbol{x}, y) > 0$ is good for classifier $f(\cdot; \Theta)$ and negative margin $m(\Theta; \boldsymbol{x}, y) < 0$ is bad

- natural to measure average strength of a typical classifier in the forest by the overall average of individual margins, i.e.,

$$s = \mathbb{E}_{\Theta}\left[\mathbb{E}_{(\boldsymbol{x},y)}(m(\Theta; \boldsymbol{x}, y))\right]$$

Breiman's Theorem: $\bar{\rho}$

- first, a correlation coefficient measuring if $f(\cdot;\Theta)$ and $f(\cdot;\Theta')$ are more or less likely to be correct together:

$$\rho(\Theta,\Theta') = \frac{\mathrm{Cov}_{(\boldsymbol{x},y)}[m(\Theta;\boldsymbol{x},y), m(\Theta';\boldsymbol{x},y)]}{\mathrm{Sd}_{(\boldsymbol{x},y)}[m(\Theta;\boldsymbol{x},y)] \times \mathrm{Sd}_{(\boldsymbol{x},y)}[m(\Theta';\boldsymbol{x},y)]}$$

- let

$$w(\Theta,\Theta') \equiv \mathrm{Sd}_{(\boldsymbol{x},y)}[m(\Theta;\boldsymbol{x},y)] \times \mathrm{Sd}_{(\boldsymbol{x},y)}[m(\Theta';\boldsymbol{x},y)],$$

the average pairwise correlation is defined as

$$\bar{\rho} = \frac{\mathbb{E}_{\Theta,\Theta'}\left[\rho(\Theta,\Theta') \times w(\Theta,\Theta')\right]}{\mathbb{E}_{\Theta,\Theta'}\left[w(\Theta,\Theta')\right]},$$

for $\Theta,\Theta' \overset{iid}{\sim} \mathcal{P}_\Theta$ ... [in fact, a weighted average]

Significance

Example Say $s = 0.4$, $\bar{\rho} = 0.5$, then theory says

$$\varepsilon_{RF} \leq \bar{\rho}\left(\frac{1-s^2}{s^2}\right) = (0.5) \times \left[\frac{1-(0.4)^2}{(0.4)^2}\right] = 2.625.$$

But of course $\varepsilon_{RF} \leq 1$. A useless bound?

Implication A good random forest should consist of individual classifiers that are

- as strong as possible (large s), and
- as uncorrelated as possible (small $\bar{\rho}$).

Adding mechanism (ii) further decorrelates the classifiers and diversifies the forest.

Strength-Diversity Trade-off

Unfortunately, there is a trade-off between the two, s and $\bar{\rho}$.

Discussion

(a) What's a tuning parameter in "the" random forest algorithm?

(b) What happens to s and to $\bar{\rho}$, respectively, as this parameter increases?

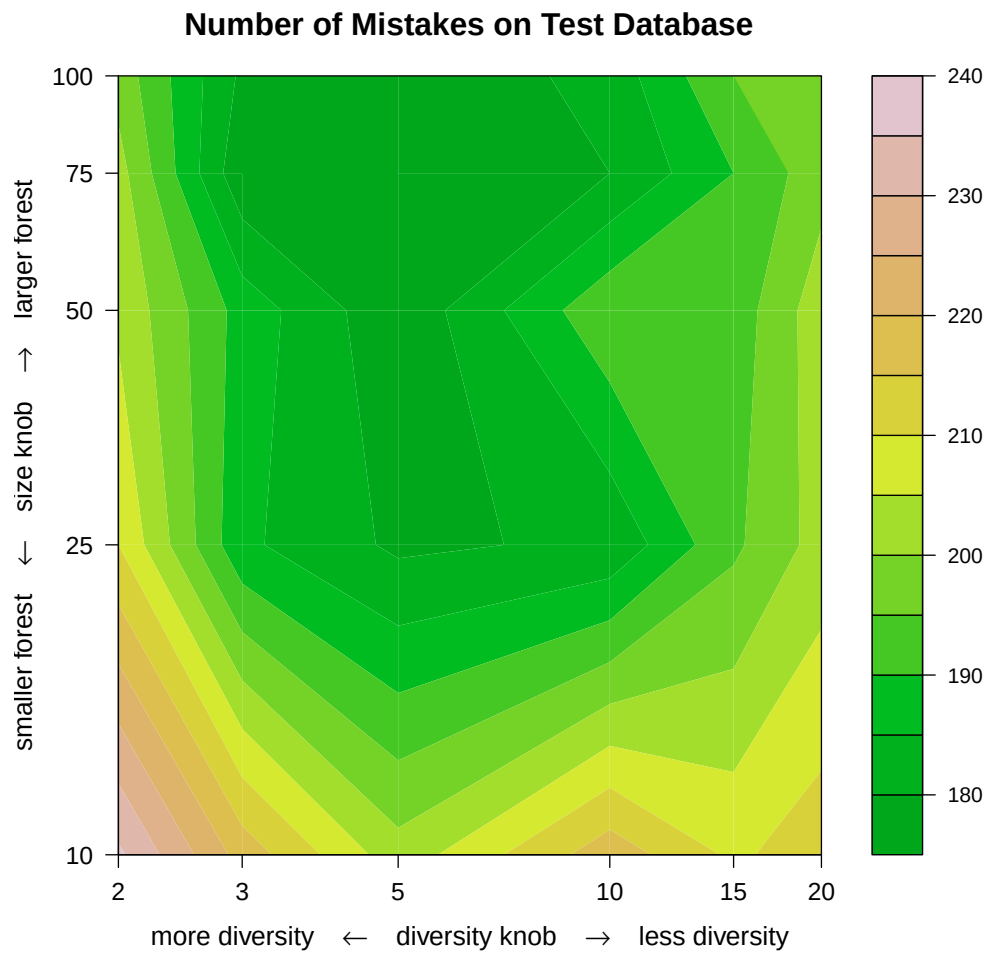

Illustration using the "spam" data set from `http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/spam.data`.

Decoupling in AdaBoost

$$\varepsilon_b = \frac{\sum_{i=1}^n w_i I(y_i \neq f_b(\boldsymbol{x}_i))}{\sum_{i=1}^n w_i} \quad \Rightarrow \quad \varepsilon_b \sum_{all} w_i = \sum_{wrong} w_i$$

$$(1-\varepsilon_b) \sum_{all} w_i = \sum_{right} w_i$$

$$w_i^{new} = \begin{cases} w_i \times \dfrac{1-\varepsilon_b}{\varepsilon_b}, & i \in \text{wrong}; \\ w_i, & i \in \text{right} \end{cases}$$

$$\sum_{wrong} w_i^{new} = \frac{1-\varepsilon_b}{\varepsilon_b} \sum_{wrong} w_i = \sum_{right} w_i = \sum_{right} w_i^{new}$$

Decoupling in AdaBoost

- error of f_b under w_i^{new} = exactly 50%, worst possible
- f_{b+1} trained on a (weighted) data set f_b cannot classify
- AdaBoost actively de-correlates f_{b+1} from f_b

Summary

- Part 1
  - bagging (1996)
  - AdaBoost (1996)
  - statistical view of boosting (2000)
  - random forest (2001)
- Part 2 (next, after short break)
  - gradient boosting machine (2001)
  - Breiman's theorem (2001)
  - some new directions (2005+)