Ensemble methods

COMS 4771 Fall 2019

Overview

- Bagging and Random Forests
- Boosting
- Margins and over-fitting

Motivation

- Recall model averaging: given T real-valued predictors $\hat{f}^{(1)}, \dots, \hat{f}^{(T)}$, form ensemble predictor $\hat{f}_{\mathrm{avg}}$

$$\hat{f}_{\mathrm{avg}}(x) := \frac{1}{T}\sum_{t=1}^{T} \hat{f}^{(t)}(x).$$

- (Squared loss) risk is

$$\mathcal{R}(\hat{f}_{\mathrm{avg}}) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{R}(\hat{f}^{(t)}) - \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[(\hat{f}_{\mathrm{avg}}(X) - \hat{f}^{(t)}(X))^2\right].$$

- For classification, analogue is *majority-vote classifier* $\hat{f}_{\mathrm{maj}}$:

$$\hat{f}_{\mathrm{maj}}(x) := \begin{cases} +1 & \text{if } \sum_{t=1}^{T} \hat{f}^{(t)}(x) > 0 \\ -1 & \text{otherwise} \end{cases}$$

($\hat{f}_{\mathrm{avg}}$ is the scoring function used for $\hat{f}_{\mathrm{maj}}$)

How to get classifiers to combine?

- Starting anew; how should we train classifiers to combine in majority-vote?
- Recall: model averaging works well when
  - all $\hat{f}^{(t)}$ have similar risks, and
  - all $\hat{f}^{(t)}$ predict very differently from each other
- To first point, use same learning algorithm for all $\hat{f}^{(t)}$
- To second point, learning algorithm should have "high variance"

Using the same learning algorithm multiple times I

- Running same learning algorithm T times on the same data set yields T identical classifiers – not helpful!
- Instead, want to run same learning algorithm on T separate data sets.

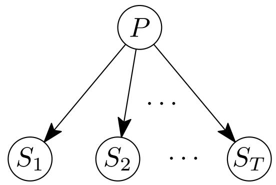


Figure 1: What we want is T data sets drawn from P

Using the same learning algorithm multiple times II

- Invoke plug-in principle
  - In IID model, regard empirical disitribution on training examples P_n as estimate of the example distribution P.
  - Draw T independent data sets from P_n; and run learning algorithm on each data set.
  - This is called *bootstrap resampling*.

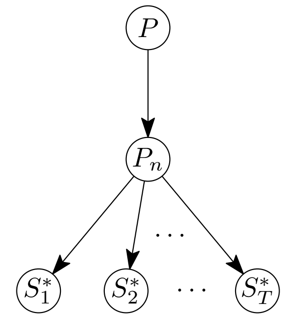


Figure 2: What we can get is T data sets from P_n

Bagging

- *Bagging*: bootstrap aggregating (Breiman, 1994)
- Given training data $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \{-1, +1\}$
- For $t = 1, \ldots, T$:
  - Randomly draw n examples with replacement from training data: $S_t^* := ((x_i^{(t)}, y_i^{(t)}))_{i=1}^n$ (*bootstrap sample*)
  - Run learning algorithm on S_t^* to get classifier $\hat{f}^{(t)}$
- Return majority-vote classifier over $\hat{f}^{(1)}, \ldots, \hat{f}^{(T)}$

Aside: Sampling with replacement

- Pick n individuals from a population of size n with replacement.
- What is the chance that a given individual is not picked?

- Implications for bagging:
  - Each bootstrap sample contains about 63% of the training examples
  - Remaining 37% can be used to estimate error rate of classifier trained on bootstrap sample

Random forests

- *Random Forests* (Breiman, 2001): Bagging with randomized variant of decision tree learning algorithm
  - Each time we need to choose a split, pick random subset of $\sqrt{d}$ features and only choose split from among those features.
- Main idea: trees may use very different features, so less likely to make mistakes in the same way.

Classifiers with independent errors

- Say we have T binary classifiers $\hat{f}^{(1)}, \dots, \hat{f}^{(T)}$
- Assume on a given x, each provides an incorrect prediction with probability 0.4:

$$\Pr(\hat{f}^{(t)}(X) \neq Y \mid X = x) = 0.4.$$

  Moreover, assume error events are independent.
- Use majority-vote classifier $\hat{f}_{\text{maj}}$.
- What is chance that more than half of the classifiers give the incorrect prediction?

Coping with non-independent errors

- Classifier errors are unlikely to be independent; do something else to benefit from majority-vote
- Change how we obtain the individual classifiers:
  - Adaptively choose classifiers
  - Re-weight training data
- Start with uniform distribution over training examples
- Loop:
  - Use learning algorithm to get new classifier for ensemble
  - Re-weight training examples to emphasize examples on which new classifier is incorrect

Adaptive Boosting

- *AdaBoost* (Freund and Schapire, 1997)
- Training data $(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \{-1, +1\}$
- Initialize $D_1(i) = 1/n$ for all $i = 1, \dots, n$
- For $t = 1, \dots, T$:
  - Run learning algorithm on D_t-weighted training examples, get classifier $f^{(t)}$
  - Update weights:

$$z_t := \sum_{i=1}^{n} D_t(i) \cdot y_i f^{(t)}(x_i) \in [-1, +1]$$

$$\alpha_t := \frac{1}{2} \ln \frac{1 + z_t}{1 - z_t} \in \mathbb{R}$$

$$D_{t+1}(i) := \frac{D_t(i) \exp(-\alpha_t \cdot y_i f^{(t)}(x_i))}{Z_t} \quad \text{for } i = 1, \dots, n.$$

    Here Z_t is normalizer that makes D_{t+1} a probability distribution.
- Final classifier: $\hat{f}(x) = \operatorname{sign}(\sum_{t=1}^{T} \alpha_t \cdot f^{(t)}(x))$

Plot of α_t as function of z_t

$$\alpha_t = \frac{1}{2} \ln \frac{1 + z_t}{1 - z_t} \in \mathbb{R}$$

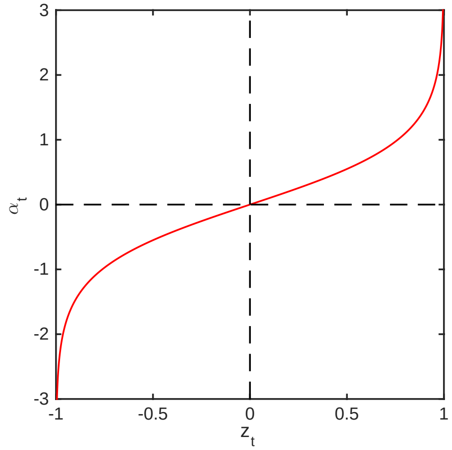


Figure 3: α_t as function of z_t

Example: AdaBoost with decision stumps

- (From Figures 1.1 and 2.2 of Schapire & Freund text.)
- Use "decision stump" learning algorithm with AdaBoost
  - Each $f^{(t)}$ has the form

$$f^{(t)}(\boldsymbol{x}) = \begin{cases} +1 & \text{if } x_i > \theta \\ -1 & \text{if } x_i \leq \theta \end{cases} \quad \text{or} \quad f^{(t)}(\boldsymbol{x}) = \begin{cases} -1 & \text{if } x_i > \theta \\ +1 & \text{if } x_i \leq \theta \end{cases}$$

  - Straightforward to handle importance weights $D_t(i)$ in decision tree learning algorithm

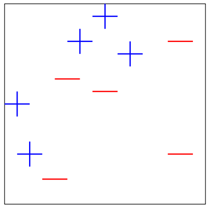
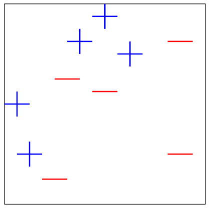
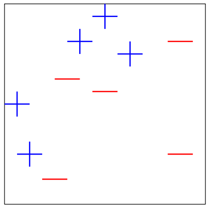


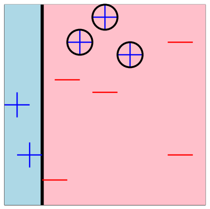
Figure 4: Training data for example execution

Example execution of AdaBoost I



Example execution of AdaBoost II

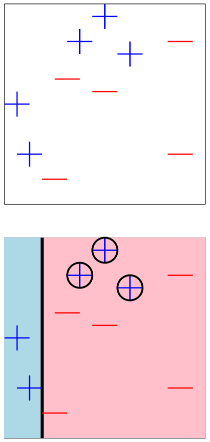




$f^{(1)}$

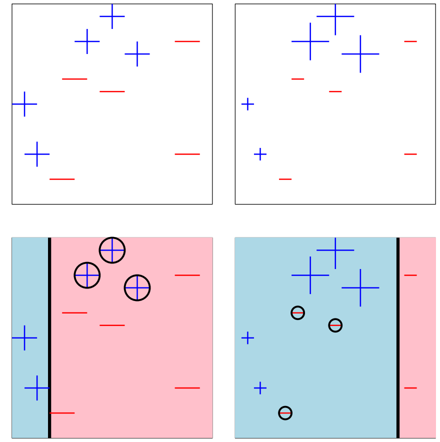
$z_1 = 0.40$, $\alpha_1 = 0.42$

Example execution of AdaBoost III



$f^{(1)}$
$z_1 = 0.40,\ \alpha_1 = 0.42$

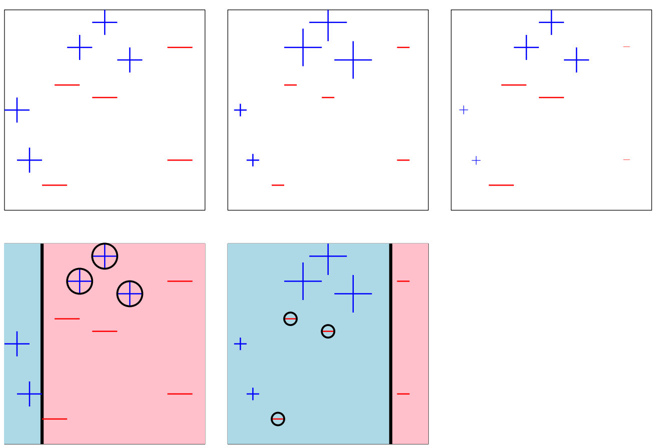
Example execution of AdaBoost IV



$f^{(1)}$ $z_1 = 0.40,\ \alpha_1 = 0.42$

$f^{(2)}$ $z_2 = 0.58,\ \alpha_2 = 0.65$

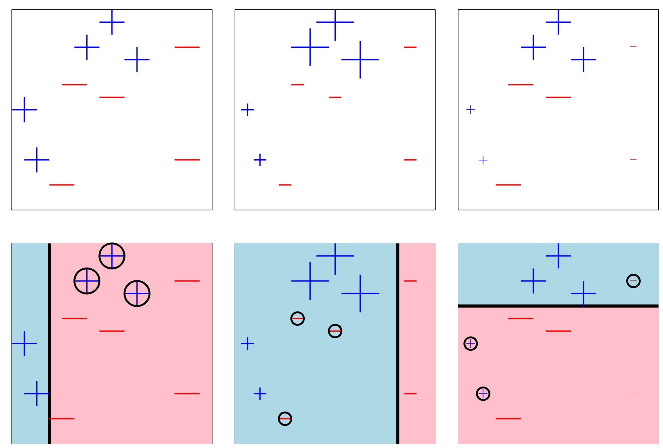
Example execution of AdaBoost V



$f^{(1)}$ $z_1 = 0.40,\ \alpha_1 = 0.42$

$f^{(2)}$ $z_2 = 0.58,\ \alpha_2 = 0.65$

Example execution of AdaBoost VI

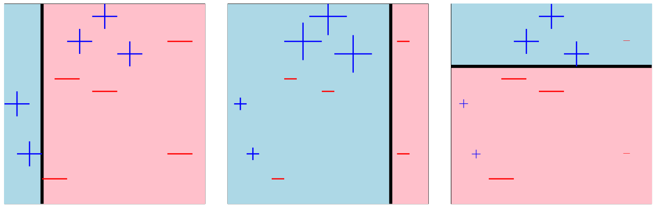


$f^{(1)}$ $z_1 = 0.40,\ \alpha_1 = 0.42$

$f^{(2)}$ $z_2 = 0.58,\ \alpha_2 = 0.65$

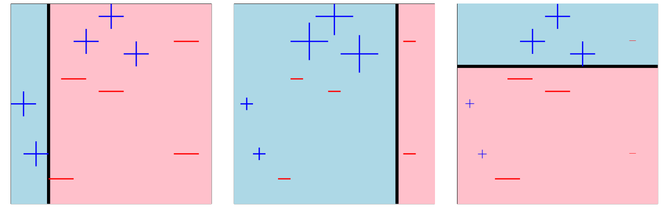
$f^{(3)}$ $z_3 = 0.72,\ \alpha_3 = 0.92$

Example execution of AdaBoost VII



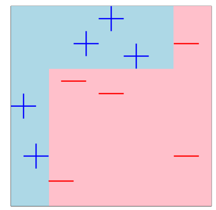
$f^{(1)}$ $z_1 = 0.40,\ \alpha_1 = 0.42$ | $f^{(2)}$ $z_2 = 0.58,\ \alpha_2 = 0.65$ | $f^{(3)}$ $z_3 = 0.72,\ \alpha_3 = 0.92$

Example execution of AdaBoost VIII



$f^{(1)}$ $z_1 = 0.40,\ \alpha_1 = 0.42$ | $f^{(2)}$ $z_2 = 0.58,\ \alpha_2 = 0.65$ | $f^{(3)}$ $z_3 = 0.72,\ \alpha_3 = 0.92$

Final classifier: $\hat{f}(x) = \text{sign}\left(0.42 f^{(1)}(x) + 0.65 f^{(2)}(x) + 0.92 f_3(x)\right)$



Training error rate of final classifier

- Let $\gamma_t := z_t/2$: advantage over random guessing achieved by $f^{(t)}$
- **Theorem**: Training error rate of final classifier is

$$\text{err}(\hat{f}, ((x_i, y_i))_{i=1}^n) \le \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right) = \exp\left(-2\bar{\gamma}^2 T\right)$$

where

$$\bar{\gamma}^2 := \frac{1}{T}\sum_{t=1}^{T}\gamma_t^2.$$

- AdaBoost is "adaptive":
  - Some γ_t can be small (even negative)—only care about average $\bar{\gamma}^2$
- What about true error rate in IID model?
  - A very complex model as T becomes large!

Surprising behavior of boosting

- AdaBoost + C4.5 decision tree learning on "letters" data set

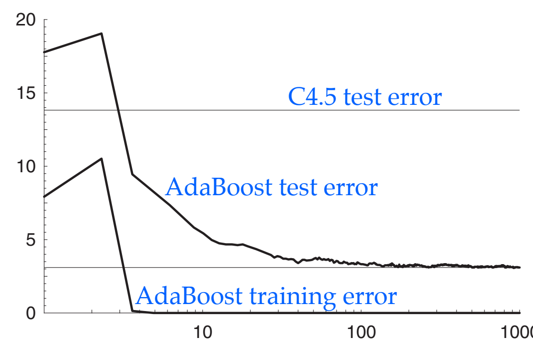


Figure 5: Figure 1.7 from Schapire & Freund text

- Training error rate is zero after five iterations.
- Test error rate continues to decrease, even up to 1000 iterations.

Margins theory

- Look at scoring function of final classifier (appropriately normalized)

$$\hat{h}(x) := \frac{\sum_{t=1}^{T} \alpha_t \cdot f^{(t)}(x)}{\sum_{t=1}^{T} |\alpha_t|} \in [-1, +1].$$

- Say $y \cdot \hat{h}(x)$ is margin achieved by $\hat{h}$ on example (x, y)
- **Theorem** (Schapire, Freund, Bartlett, and Lee, 1998):
  - Larger margins on training examples $\Rightarrow$ better resistance to over-fitting in IID model
  - AdaBoost tends to increase margins on training examples
- (Similar to but not same as SVM margins)
- On "letters" data set

	$T = 5$	$T = 100$	$T = 1000$
training error rate	0.0%	0.0%	0.0%
test error rate	8.4%	3.3%	3.1%
% margins ≤ 0.5	7.7%	0.0%	0.0%
min. margin	0.14	0.52	0.55