Ensembles

Mark Craven and David Page
Computer Sciences 760
Spring 2019

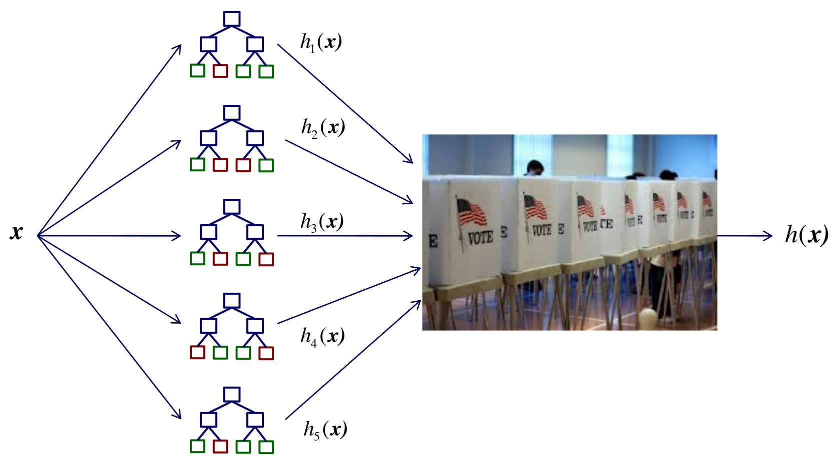
Goals for the lecture

you should understand the following concepts

- ensemble
- bootstrap sample
- bagging
- boosting
- random forests

2

What is an ensemble?



a set of learned models whose individual decisions are combined in some way to make predictions for new instances

When can an ensemble be more accurate?

- when the errors made by the individual predictors are (somewhat) uncorrelated, and the predictors' error rates are better than guessing (< 0.5 for 2-class problem)
- consider an idealized case…

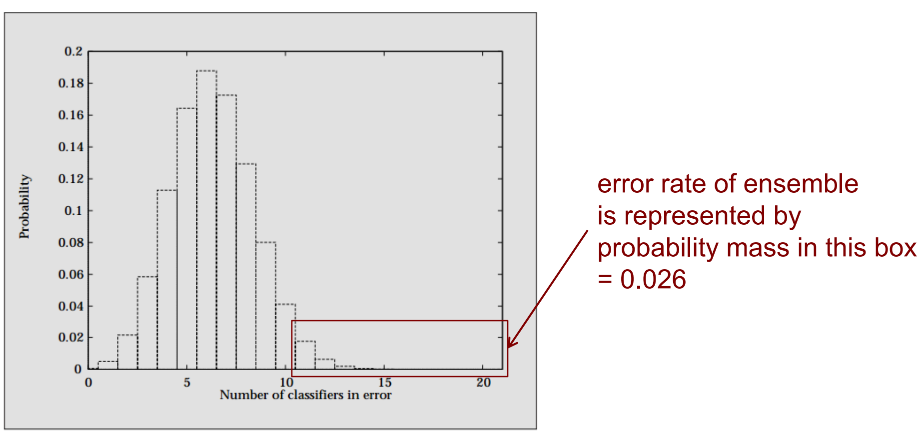


Figure 1. The Probability That Exactly ℓ (of 21) Hypotheses Will Make an Error, Assuming Each Hypothesis Has an Error Rate of 0.3 and Makes Its Errors Independently of the Other Hypotheses.

Figure from Dietterich, *AI Magazine*, 1997

How can we get diverse classifiers?

- In practice, we can't get classifiers whose errors are completely uncorrelated, but we can encourage diversity in their errors by
  - choosing a variety of learning algorithms
  - choosing a variety of settings (e.g. # hidden units in neural nets) for the learning algorithm
  - ✓ choosing different subsamples of the training set (*bagging*)
  - ✓ using different probability distributions over the training instances (*boosting, skewing*)
  - ✓ choosing different features and subsamples (*random forests*)

5

Bagging (Bootstrap Aggregation)

[Breiman, *Machine Learning* 1996]

learning:
given: learner L, training set $D = \{ \langle \boldsymbol{x}^{(1)}, y^{(1)} \rangle \ \ldots \ \langle \boldsymbol{x}^{(m)}, y^{(m)} \rangle \}$
for $i \leftarrow 1$ to T do
$D_i \leftarrow m$ instances randomly drawn with replacement from D
$h_i \leftarrow$ model learned using L on D_i

classification:
given: test instance $\boldsymbol{x}$
predict $y \leftarrow$ plurality_vote($h_1(\boldsymbol{x}) \ldots h_T(\boldsymbol{x})$)

regression:
given: test instance $\boldsymbol{x}$
predict $y \leftarrow$ mean($h_1(\boldsymbol{x}) \ldots h_T(\boldsymbol{x})$)

Bagging

- each sampled training set is a *bootstrap replicate*
  - contains m instances (the same as the original training set)
  - on average it includes 63.2% of the original training set
  - some instances appear multiple times
- can be used with any base learner
- works best with *unstable* learning methods: those for which small changes in D result in relatively large changes in learned models, i.e., those that tend to *overfit* training data

7

Empirical evaluation of bagging with C4.5

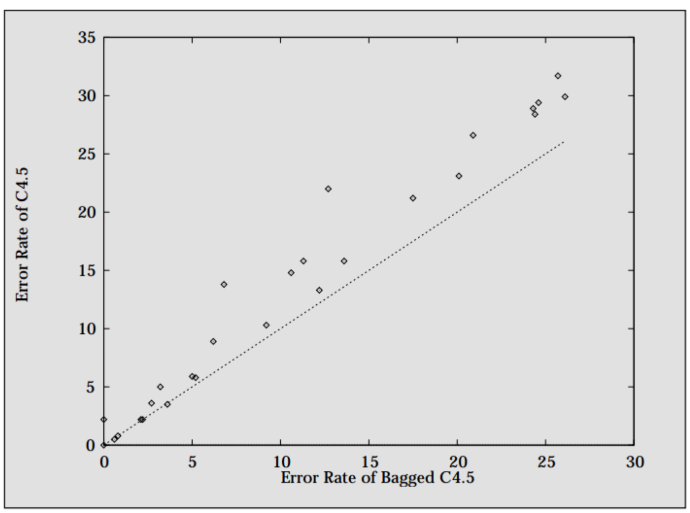


Figure from Dietterich, *AI Magazine*, 1997

Bagging reduced error of C4.5 on most data sets; wasn't harmful on any

8

Boosting

- Boosting came out of the PAC learning community
- A *weak PAC learning* algorithm is one that cannot PAC learn for arbitrary ε and δ, but it can for some: its hypotheses are at least slightly better than random guessing
- Suppose we have a *weak PAC learning* algorithm L for a concept class C. Can we use L as a subroutine to create a (strong) PAC learner for C?
  - Yes, by boosting! [Schapire, *Machine Learning* 1990]
  - The original boosting algorithm was of theoretical interest, but assumed an unbounded source of training instances
- A later boosting algorithm, AdaBoost, has had notable practical success

9

AdaBoost

[Freund & Schapire, Journal of Computer and System Sciences, 1997]

given: learner L, # stages T, training set $D = \{ \langle \boldsymbol{x}^{(1)}, y^{(1)} \rangle \ \dots \ \langle \boldsymbol{x}^{(m)}, y^{(m)} \rangle \}$

for all i : $w_1(i) \leftarrow 1/m$ // initialize instance weights
for $t \leftarrow 1$ to T do
    for all i : $p_t(i) \leftarrow w_t(i) \,/\, (\sum_j w_t(j))$ // normalize weights
    $h_t \leftarrow$ model learned using L on D and p_t
    $\varepsilon_t \leftarrow \sum_i p_t(i)(1 - \delta(h_t(\boldsymbol{x}^{(i)}), y^{(i)}))$ // calculate weighted error
    if $\varepsilon_t > 0.5$ then
        $T \leftarrow t - 1$
        break
    $\beta_t \leftarrow \varepsilon_t \,/\, (1 - \varepsilon_t)$
    for all i where $h_t(\boldsymbol{x}^{(i)}) = y^{(i)}$ // down-weight correct examples
        $w_{t+1}(i) \leftarrow w_t(i)\, \beta_t$

return:

$$h(\boldsymbol{x}) = \arg\max_y \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) \delta\left(h_t(\boldsymbol{x}),\, y\right)$$

Implementing weighted instances with AdaBoost

- AdaBoost calls the base learner L with probability distribution p_t specified by weights on the instances

- there are two ways to handle this
  1. Adapt L to learn from weighted instances; straightforward for decision trees and naïve Bayes, among others
  2. Sample a large ($>> m$) unweighted set of instances according to p_t ; run L in the ordinary manner

AdaBoost variants

- AdaBoost.M1: 1-of-n multiclass tasks

- AdaBoost.M2: arbitrary multiclass tasks

- AdaBoost.R: regression

- confidence-rated predictions (learners output their confidence in predicted class for each instance)

- etc.

Empirical evaluation of boosting with C4.5

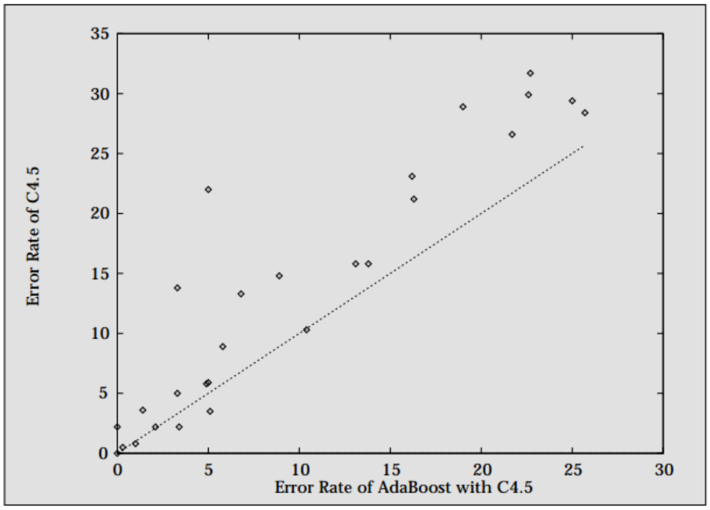


Figure from Dietterich, *AI Magazine*, 1997

13

Bagging and boosting with C4.5

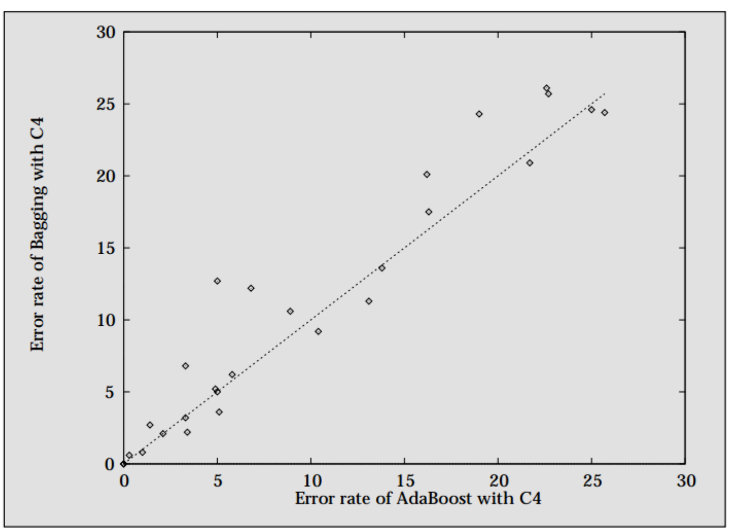


Figure from Dietterich, *AI Magazine*, 1997

14

Empirical study of bagging vs. boosting

[Opitz & Maclin, *JAIR* 1999]

- 23 data sets
- C4.5 and neural nets as base learners
- bagging almost always better than single decision tree or neural net
- boosting can be much better than bagging
- however, boosting can sometimes reduce accuracy (too much emphasis on outliers?)

15

Random forests

[Breiman, Machine Learning 2001]

given: candidate feature splits F,
training set $D = \{ \langle \boldsymbol{x}^{(1)}, y^{(1)} \rangle \ \ldots \ \langle \boldsymbol{x}^{(m)}, y^{(m)} \rangle \}$
for $i \leftarrow 1$ to T do
$D_i \leftarrow m$ instances randomly drawn with replacement from D
$h_i \leftarrow$ randomized decision tree learned with F, D_i

randomized decision tree learning:
to select a split at a node
$R \leftarrow$ randomly select (without replacement) f feature splits from F
(where $f << |F|$)
choose the best feature split in R
do not prune trees

classification/regression:
as in bagging

Large-scale comparison of learning methods

[Fernández-Delgado *JMLR* 2014]

- compared 179 classifiers on 121 data sets
- random forest was the best family of classifiers (3 classifiers in the top 5)

Rank	Acc.	κ	Classifier
32.9	82.0	63.5	parRF_t (RF)
33.1	**82.3**	**63.6**	rf_t (RF)
36.8	81.8	62.2	svm_C (SVM)
38.0	81.2	60.1	svmPoly_t (SVM)
39.4	81.9	62.5	rforest_R (RF)
39.6	82.0	62.0	elm_kernel_m (NNET)
40.3	81.4	61.1	svmRadialCost_t (SVM)
42.5	81.0	60.0	svmRadial_t (SVM)
42.9	80.6	61.0	C5.0_t (BST)
44.1	79.4	60.5	avNNet_t (NNET)
45.5	79.5	61.0	nnet_t (NNET)
47.0	78.7	59.4	pcaNNet_t (NNET)
47.1	80.8	53.0	BG_LibSVM_w (BAG)
47.3	80.3	62.0	mlp_t (NNET)
47.6	80.6	60.0	RotationForest_w (RF)
50.1	80.9	61.6	RRF_t (RF)
51.6	80.7	61.4	RRFglobal_t (RF)
52.5	80.6	58.0	MAB_LibSVM_w (BST)
52.6	79.9	56.9	LibSVM_w (SVM)
57.6	79.1	59.3	adaboost_R (BST)

One application of random forests: human pose recognition in the Xbox Kinect

[Shotton et al., *CVPR* 2011]

Classification task

- Given: a depth image
- Do: classify each pixel into one of 31 body parts

Comments on ensembles

- They very often provide a boost in accuracy over base learner
- It’s a good idea to evaluate an ensemble approach for almost any practical learning problem
- They increase runtime over base learner, but compute cycles are usually much cheaper than training instances
- Some ensemble approaches (e.g. bagging, random forests) are easily parallelized
- Prediction contests (e.g. Kaggle, Netflix Prize) often won by ensemble solutions
- Ensemble models are usually low on the comprehensibility scale
- Instead of voting or weighted voting, can also learn the combining function; this is called *stacking*