

No Free Lunch, Bias-Variance & Ensembles

May 27, 2009

Acknowledgement: Material derived from slides for the book Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997 <http://www-2.cs.cmu.edu/~tom/mlbook.html> and slides by Andrew W. Moore available at <http://www.cs.cmu.edu/~awm/tutorials> and the book Data Mining, Ian H. Witten and Eibe Frank, Morgan Kaufman, 2000. <http://www.cs.waikato.ac.nz/ml/weka> and the book Pattern Classification, Richard O. Duda, Peter E. Hart, and David G. Stork. Copyright (c) 2001 by John Wiley & Sons, Inc. and the book "Elements of Statistical Learning", Trevor Hastie, Robert Tibshirani and Jerome Friedman. (c) 2001, Springer.

Aims

This lecture aims to develop your understanding of some recent advances in machine learning. Following it you should be able to:

- outline the No Free Lunch Theorem
- describe the framework of the bias-variance decomposition
- define the method of bagging
- define the method of boosting

Relevant Weka methods: Bagging, Random Forests, AdaBoostM1, Stacking, SMO

Some questions about Machine Learning

- Are there reasons to prefer one learning algorithm over another ?
- Can we expect any method to be superior overall ?
- Can we even find an algorithm that is overall superior to random guessing ?

No Free Lunch Theorem

- uniformly averaged over all target functions, the expected off-training-set error for all learning algorithms is the same
- even for a fixed training set, averaged over all target functions no learning algorithm yields an off-training-set error that is superior to any other

Therefore, all statements of the form “learning algorithm 1 is better than algorithm 2” are ultimately statements about the relevant target functions.

No Free Lunch example

Assuming that the training set \mathcal{D} can be learned correctly by all algorithms, averaged over all target functions no learning algorithm gives an off-training set error superior to any other:

$$\Sigma_F[\mathcal{E}_1(E|F, \mathcal{D}) - \mathcal{E}_2(E|F, \mathcal{D})] = 0$$

No Free Lunch example

	x	F	h_1	h_2
\mathcal{D}	000	1	1	1
	001	-1	-1	-1
	010	1	1	1
	011	-1	1	-1
	100	1	1	-1
	101	-1	1	-1
	110	1	1	-1
	111	1	1	-1

$$\mathcal{E}_1(E|F, \mathcal{D}) = 0.4$$

$$\mathcal{E}_2(E|F, \mathcal{D}) = 0.6$$

No Free Lunch example

BUT

if we have *no prior knowledge* about which F we are trying to learn, neither algorithm is superior to the other

both fit the training data correctly, but there are 2^5 target functions consistent with \mathcal{D}

and for each there is exactly one other function whose output is inverted with respect to each of the off-training set patterns

so the performance of algorithms 1 and 2 will be inverted thus ensuring average error difference of zero

A Conservation Theorem of Generalization Performance

For every possible learning algorithm for binary classification the sum of performance over all possible target functions is exactly zero.

- on some problems we get positive performance
- so there *must* be other problems for which we get an *equal and opposite amount* of negative performance

It is the *assumptions* about the learning domains that are relevant.

Experience with a *broad range of techniques* is the best insurance for solving arbitrary new classification problems.

Ugly Duckling Theorem

In the absence of assumptions there is no privileged or “best” feature representation.

In fact, even the notion of similarity between patterns depends on assumptions.

- Using a finite number of predicates to distinguish any two patterns, the number of predicates shared by any two such patterns is constant and independent of those patterns.

Bias-variance decomposition

- Theoretical tool for analyzing how much specific training set affects performance of classifier
- Assume we have an infinite number of classifiers built from different training sets of size n
 - The *bias* of a learning scheme is the expected error of the combined classifier on new data
 - The *variance* of a learning scheme is the expected error due to the particular training set used
 - Total expected error: bias + variance

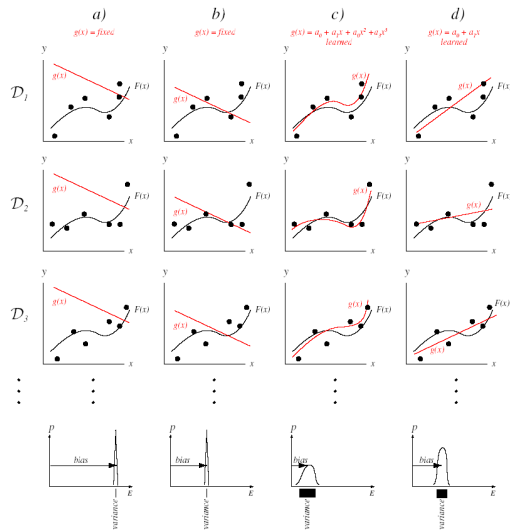
Bias-variance: a trade-off

Easier to see with regression in the following figure ¹ (to see the details you will have to zoom in in your viewer):

- each column represents a different model class $g(x)$ shown in red
- each row represents a different set of $n = 6$ training points, D_i , randomly sampled from target function $F(x)$ with noise, shown in black
- probability functions of mean squared error E are shown

¹from: “Elements of Statistical Learning” by Hastie, Tibshirani and Friedman (2001)

Bias-variance: a trade-off



COMP9417: May 27, 2009

No Free Lunch, Bias-Variance & Ensembles: Slide 11

Bias-variance: a trade-off

- a) is very poor: a linear model with fixed parameters independent of training data; high bias, zero variance
- b) is better: a linear model with fixed parameters independent of training data; slightly lower bias, zero variance
- c) is a cubic model with parameters trained by mean-square-error on training data; low bias, moderate variance
- d) is a linear model with parameters adjusted to fit each training set; intermediate bias and variance
- training with data $n \rightarrow \infty$ would give
 - c) with bias approaching small value due to noise
 - but not d)
 - variance for all models would approach zero

COMP9417: May 27, 2009

No Free Lunch, Bias-Variance & Ensembles: Slide 12

Ensembles: combining multiple models

- Basic idea of *ensembles* or “meta” learning schemes: build different “experts” and let them vote
- Advantage: often improves predictive performance
- Disadvantage: produces output that is very hard to interpret
- Notable schemes: bagging, boosting, stacking
 - can be applied to both classification and numeric prediction problems

COMP9417: May 27, 2009

No Free Lunch, Bias-Variance & Ensembles: Slide 13

Bootstrap error estimation

Estimating error rate of a learning method on a data set

- sampling from data set *with replacement*
- e.g. sample from n instances, with replacement, n times to generate another data set of n instances
- (almost certainly) new data set contains some duplicate instances
- and does not contain others – used as the test set
- chance of *not* being picked $(1 - \frac{1}{n})^n \approx e^{-1} = 0.368$
- 0.632 training set
- error estimate = $0.632 \times \text{err}_{\text{test}} + 0.368 \times \text{err}_{\text{train}}$
- repeat and average with different bootstrap samples

COMP9417: May 27, 2009

No Free Lunch, Bias-Variance & Ensembles: Slide 14

Bagging

“Bootstrap Aggregation”

- Employs simplest way of combining predictions: voting/averaging
- Each model receives equal weight
- Generalized version of bagging:
 - Sample several training sets of size n (instead of just having one training set of size n)
 - Build a classifier for each training set
 - Combine the classifiers' predictions
- This improves performance in almost all cases if learning scheme is unstable (i.e. decision trees)

Bagging

- Bagging reduces variance by voting/averaging, thus reducing the overall expected error
 - In the case of classification there are pathological situations where the overall error might increase
 - Usually, the more classifiers the better
- Problem: we only have one dataset!
- Solution: generate new datasets of size n by sampling with replacement from original dataset
- Can help a lot if data is noisy

Bagging algorithm

Learning (model generation)

Let n be the number of instances in the training data.

For each of t iterations:

- Sample n instances with replacement from training set.
- Apply the learning algorithm to the sample.
- Store the resulting model.

Classification

For each of the t models:

- Predict class of instance using model.
- Return class that has been predicted most often.

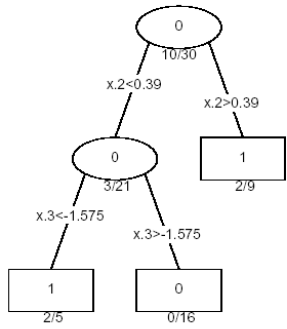
Bagging trees

An experiment with simulated data:

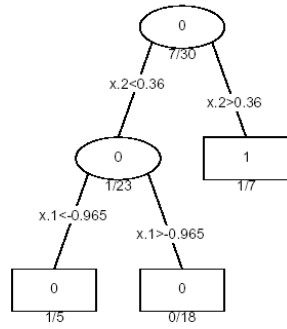
- sample of size $n = 30$, two classes, five features
- $Pr(Y = 1|x_1 \leq 0.5) = 0.2$ and $Pr(Y = 1|x_1 > 0.5) = 0.8$
- test sample of size 2000 from same population
- fit classification trees to training sample, 200 bootstrap samples
- trees are different (tree induction is *unstable*)
- therefore have high variance
- averaging reduces variance and leaves bias unchanged
- (graph: test error for original and bagged trees, with green – vote; purple – average probabilities)

Bagging trees

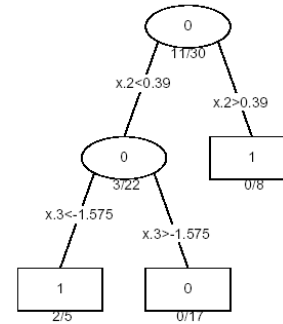
Original Tree



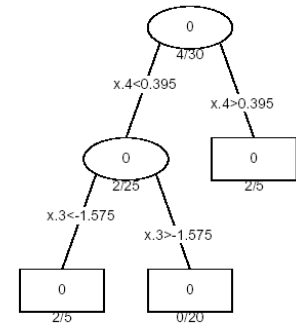
Bootstrap Tree 1



Bootstrap Tree 2

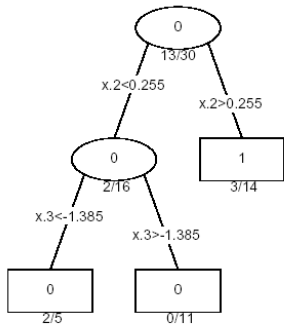


Bootstrap Tree 3

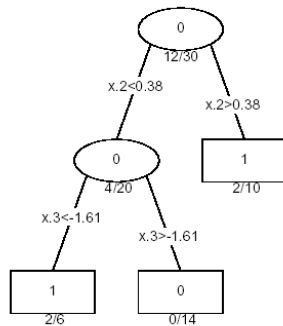


Bagging trees

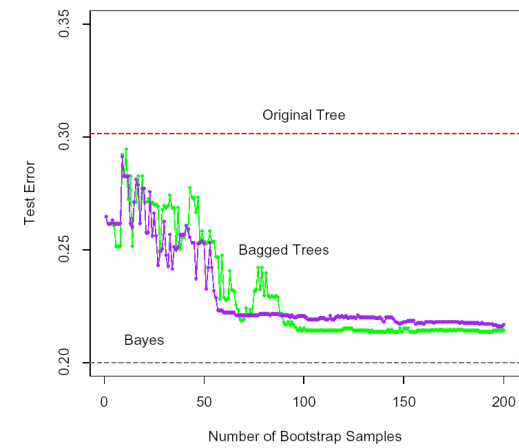
Bootstrap Tree 4



Bootstrap Tree 5



Bagging trees



Bagging trees

The news is not all good:

- when we bag a model, any simple structure is lost
- this is because a bagged tree is no longer a tree ...
- ... but a forest
- this drastically reduces any claim to comprehensibility
- *stable* models like nearest neighbour not very affected by bagging
- *unstable* models like trees most affected by bagging
- usually, their design for interpretability (bias) leads to instability
- more recently, *random forests* (see Breiman's web-site)

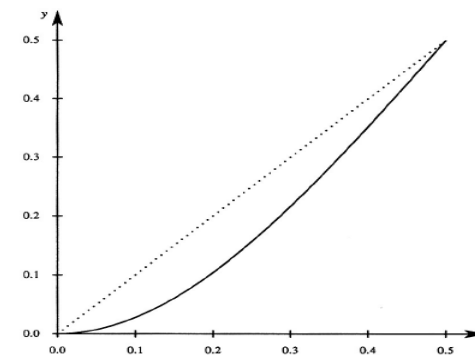
Boosting

- Also uses voting/averaging but each model is *weighted* according to their performance
- Iterative procedure: new models are influenced by performance of previously built ones
 - New model is encouraged to become “expert” for instances classified incorrectly by earlier models
 - Intuitive justification: models should be experts that complement each other
- There are several variants of this algorithm ...

The strength of weak learnability

- Schapire (1990) - first boosting algorithm
- showed that *weak* learners can be boosted into *strong* learners
- original setting:
 - weak learner learns initial hypothesis h_1 from N examples
 - next learns hypothesis h_2 from new set of N examples, half of which are misclassified by h_1
 - then learns hypothesis h_3 from N examples for which h_1 and h_2 disagree
 - “boosted” hypothesis h gives voted prediction on instance x :
 - * if $h_1(x) = h_2(x)$ then return agreed prediction, else
 - * return $h_3(x)$
- if h_1 gets error $\alpha < 0.5$ then error of h bounded by $3\alpha^2 - 2\alpha^3$, i.e. better than α

Boosting a weak learner reduces error



A graph of the function $g(x) = 3x^2 - 2x^3$.

Learning (model generation)

Assign equal weight to each training instance.

For each of t iterations:

 Apply learning algorithm to weighted dataset and store resulting model.

 Compute error e of model on weighted dataset and store error.

 If e equal to zero, or e greater or equal to 0.5:

 Terminate model generation.

 For each instance in dataset:

 If instance classified correctly by model:

 Multiply weight of instance by $e / (1 - e)$.

 EndFor

 Normalize weight of all instances.

EndFor

Classification

Assign weight of zero to all classes.

For each of the t (or less) models:

 Add $-\log(e / (1 - e))$ to weight of class predicted by model.

EndFor

Return class with highest weight.

More on boosting

- Can be applied without weights using resampling with probability determined by weights
 - Disadvantage: not all instances are used
 - Advantage: resampling can be repeated if error exceeds 0.5
- Stems from computational learning theory
- Theoretical result: training error decreases exponentially
- Also: works if base classifiers not too complex and their error doesn't become too large too quickly

A bit more on boosting

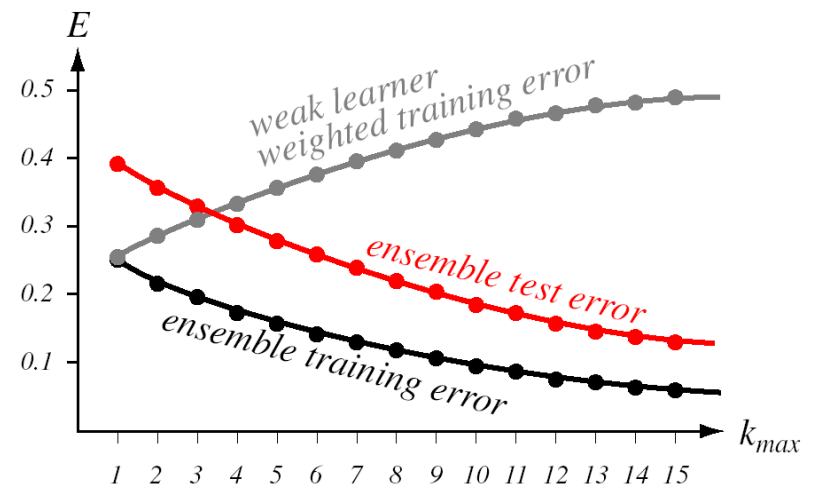
- Puzzling fact: generalization error can decrease long after training error has reached zero
 - Seems to contradict Occam's Razor !
 - However, problem disappears if *margin* (confidence) is considered instead of error
 - * Margin: difference between estimated probability for true class and most likely other class (between -1, 1)
- Boosting works with *weak* learners: only condition is that error α doesn't exceed 0.5 (slightly better than random guessing)
- LogitBoost: more sophisticated boosting scheme in Weka (based on additive logistic regression)

Boosting reduces error

Adaboost applied to a weak learning system can reduce the training error exponentially as the number of component classifiers is increased.

- focuses on “difficult” patterns
- training error of successive classifier on its own weighted training set is generally larger than predecessor
- training error of ensemble will decrease
- typically, test error of ensemble will decrease also

Boosting reduces error

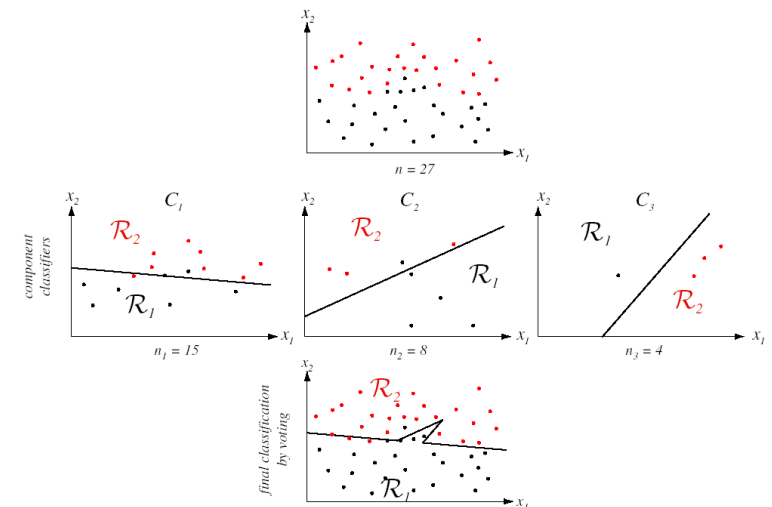


Boosting enlarges the model class

A two-dimensional two-category classification task

- three component linear classifiers
- final classification is by voting component classifiers
- gives a non-linear decision boundary
- each component is a weak learner (slightly better than 0.5)
- ensemble classifier has error lower than any single component
- ensemble classifier has error lower than single classifier on complete training set

Boosting enlarges the model class



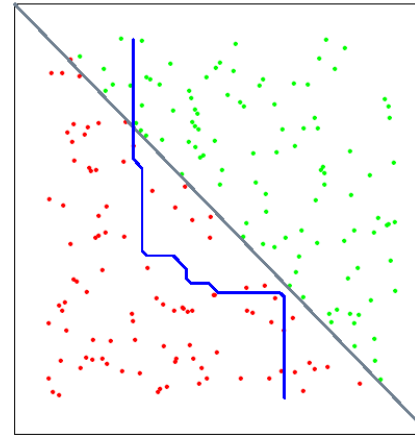
Boosting enlarges the model class

An experiment with simulated data:

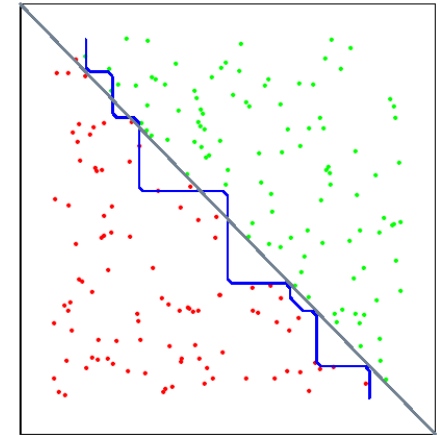
- 100 instances, two features, two classes
- target classification is $x_1 + x_2 = 1$
- learn classifier: single split in x_1 or x_2 to give largest decrease in training set misclassification error
- voting or averaging probabilities does not help over many single splits
- however, repeated iterations of boosting gets closer approximation to the diagonal

Boosting enlarges the model class

Bagged Decision Rule



Boosted Decision Rule



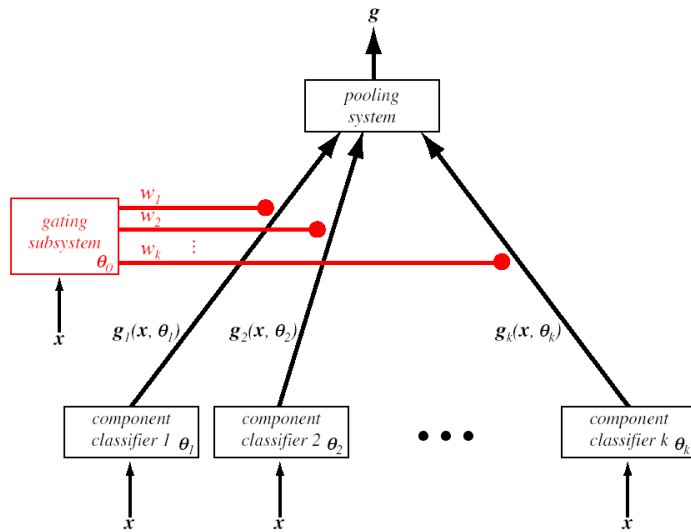
Stacking

- Hard to analyze theoretically: “black magic”
- Uses *meta learner* instead of voting to combine predictions of base learners
 - Predictions of base learners (level-0 models) are used as input for meta learner (level-1 model)
- Base learners usually different learning schemes
- Predictions on training data can't be used to generate data for level-1 model!
 - Cross-validation-like scheme is employed

Stacking

- If base learners can output probabilities it's better to use those as input to meta learner
- Which algorithm to use to generate meta learner?
 - In principle, any learning scheme can be applied
 - David Wolpert: “relatively global, smooth” model
 - * Base learners do most of the work
 - * Reduces risk of overfitting
- Stacking can also be applied to numeric prediction (and density estimation)

Stacking



Summary Points

1. No Free Lunch and Ugly Duckling Theorems → no “magic bullet”
2. Bias-variance decomposition breaks down the error, illustrates the “match” of a learning method to a problem
3. Bagging is a simple way to run ensemble methods
4. Boosting often works better but can be susceptible to very noisy data
5. Stacking not widely investigated but useful to combine different learners
6. Kernel methods around for a long time in statistics
7. SVMs a “modular” approach to machine learning with a choice of different kernels – many applications
8. Current “most favoured” off-the-shelf classifiers – boosting, SVMs