

Sparse vs. Ensemble Approaches to Supervised Learning

Greg Grudic
CSCI 5622

1

Goal of Supervised Learning?

- Minimize the probability of model prediction errors on **future** data
- Two Competing Methodologies
 - Build **one** really good model
 - Traditional approach
 - Build **many** models and average the results
 - Ensemble learning (more recent)

2

The Single Model Philosophy

- Motivation: Occam's Razor
 - “one should not increase, beyond what is necessary, the number of entities required to explain anything”
- Infinitely many models can explain any given dataset
 - Might as well pick the smallest one...

3

Which Model is Smaller?

$$\hat{y} = f_1(x) = \sin(x)$$

or

$$\hat{y} = f_2(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

- In this case $f_1(x) \equiv f_2(x)$
- It's not always easy to define small!

4

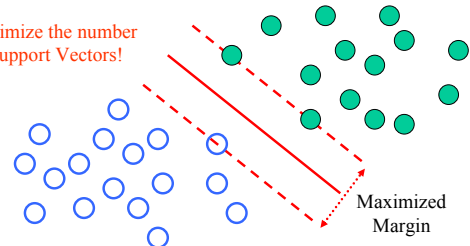
Exact Occam's Razor Models

- Exact approaches find optimal solutions
- Examples:
 - Support Vector Machines
 - Find a model structure that uses the smallest percentage of training data (to explain the rest of it).
 - Bayesian approaches
 - Minimum description length

5

How Do Support Vector Machines Define Small?

Minimize the number of Support Vectors!



6

Approximate Occam's Razor Models

- Approximate solutions use a greedy search approach which is not optimal
- Examples
 - Kernel Projection Pursuit algorithms
 - Find a minimal set of kernel projections
 - Relevance Vector Machines
 - Approximate Bayesian approach
 - Sparse Minimax Probability Machine Classification
 - Find a minimum set of kernels and features

7

Other Single Models: **Not Necessarily Motivated by Occam's Razor**

- Minimax Probability Machine (MPM)
- Trees
 - Greedy approach to sparseness
- Neural Networks
- Nearest Neighbor
- Basis Function Models
 - e.g. Kernel Ridge Regression

8

Ensemble Philosophy

- Build many models and combine them
- Only through averaging do we get at the truth!
- It's too hard (*impossible?*) to build a single model that works best
- Two types of approaches:
 - Models that don't use randomness
 - Models that incorporate randomness

9

Ensemble Approaches

- Bagging
 - **B**ootstrap **a**ggregating
- Boosting
- Random Forests
 - Bagging reborn

10

Bagging

- Main Assumption:
 - Combining many unstable predictors to produce a ensemble (stable) predictor.
 - Unstable Predictor: small changes in training data produce large changes in the model.
 - e.g. Neural Nets, trees
 - Stable: SVM (sometimes), nearest Neighbor.
- Hypothesis Space
 - Variable size (nonparametric):
 - Can model any function if you use an appropriate predictor (e.g. trees)

11

The Bagging Algorithm

Given data: $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$

For $m = 1 : M$

- Obtain bootstrap sample D_m from the training data D
- Build a model $G_m(\mathbf{x})$ from bootstrap data D_m

12

The Bagging Model

- Regression

$$\hat{y} = \frac{1}{M} \sum_{m=1}^M G_m(\mathbf{x})$$

- Classification:
 - Vote over classifier outputs $G_1(\mathbf{x}), \dots, G_M(\mathbf{x})$

13

Bagging Details

- Bootstrap sample of N instances is obtained by drawing N examples at random, with replacement.
- On average each bootstrap sample has 63% of instances
 - Encourages predictors to have uncorrelated errors
 - This is why it works

14

Bagging Details 2

- Usually set $M \approx 30$
 - Or use validation data to pick M
- The models $G_m(\mathbf{x})$ need to be unstable
 - Usually full length (or slightly pruned) decision trees.

15

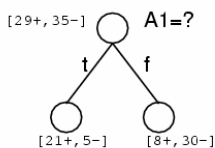
Boosting

- Main Assumption:
 - Combining many weak predictors (e.g. tree stumps or 1-R predictors) to produce an ensemble predictor
 - The weak predictors or classifiers need to be stable
- Hypothesis Space
 - Variable size (nonparametric):
 - Can model any function if you use an appropriate predictor (e.g. trees)

16

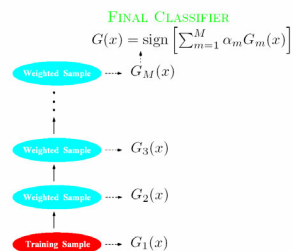
Commonly Used Weak Predictor (or classifier)

A Decision Tree Stump (1-R)



17

Boosting



Each classifier $G_m(\mathbf{x})$ is trained from a weighted Sample of the training Data

18

Boosting (Continued)

- Each predictor is created by using a biased sample of the training data
 - Instances (training examples) with high error are weighted higher than those with lower error
- Difficult instances get more attention
 - This is the motivation behind boosting

19

Background Notation

- The $I(s)$ function is defined as:

$$I(s) = \begin{cases} 1 & \text{if } s \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

- The $\log(x)$ function is the natural logarithm

20

The AdaBoost Algorithm

(Freund and Schapire, 1996)

Given data: $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$

- Initialize weights $w_i = 1/N, i = 1, \dots, N$
- For $m = 1 : M$
 - Fit classifier $G_m(\mathbf{x}) \in \{-1, 1\}$ to data using weights w_i
 - Compute
$$err_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(\mathbf{x}_i))}{\sum_{i=1}^N w_i}$$
 - Compute $\alpha_m = \log((1 - err_m) / err_m)$
 - Set $w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq G_m(\mathbf{x}_i))], \quad i = 1, \dots, N$

21

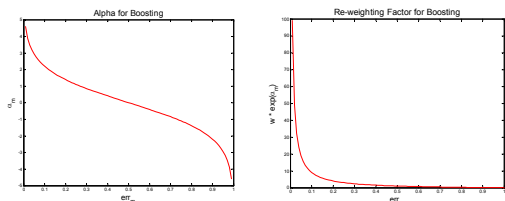
The AdaBoost Model

$$\hat{y} = \text{sgn} \left[\sum_{m=1}^M \alpha_m G_m(\mathbf{x}) \right]$$

AdaBoost is NOT used for Regression!

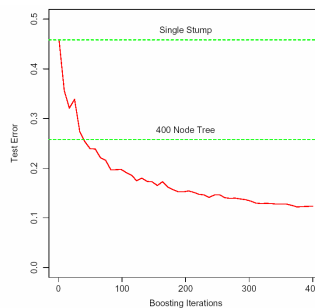
22

The Updates in Boosting



23

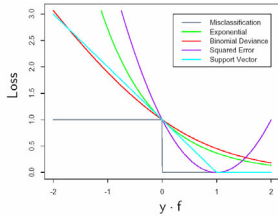
Boosting Characteristics



Simulated data: 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 400 node tree.

24

Loss Functions for $y \in \{-1, +1\}, f \in \mathcal{R}$



- Misclassification
 $I(\text{sgn}(f) \neq y)$
- Exponential (Boosting)
 $\exp(-yf)$
- Binomial Deviance (Cross Entropy)
 $\log(1 + \exp(-2yf))$
- Squared Error
 $(y - f)^2$
- Support Vectors
 $(1 - yf) \cdot I(yf > 1)$

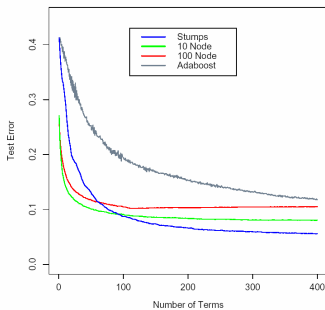
25

Other Variations of Boosting

- Gradient Boosting
 - Can use any cost function
- Stochastic (Gradient) Boosting
 - Bootstrap Sample: Uniform random sampling (with replacement)
 - Often outperforms the non-random version

26

Gradient Boosting



27

Boosting Summary

- Good points
 - Fast learning
 - Capable of learning any function (given appropriate weak learner)
 - Feature weighting
 - Very little parameter tuning
- Bad points
 - Can overfit data
 - Only for binary classification
- Learning parameters (picked via cross validation)
 - Size of tree
 - When to stop
- Software
 - <http://www-stat.stanford.edu/~jhf/R-MART.html>

28

Random Forests

(Leo Breiman, 2001)

<http://www.stat.berkeley.edu/users/breiman/RandomForests/>

- Injecting the right kind of randomness makes accurate models
 - As good as SVMs and sometimes better
 - As good as boosting
- **Very little playing with learning parameters is needed to get very good models**
- Traditional tree algorithms spend a lot of time choosing how to split at a node
 - Random forest trees put very little effort into this

29

Algorithmic Goal of Random Forests

- Create many trees (50-1,000)
- Inject randomness into trees such that
 - Each tree has maximal strength
 - i.e. a fairly good model on its own
 - Each tree has minimum correlation with the other trees
 - i.e. the errors tend to cancel out

30

RFs Use Out-of-Bag Samples

- Out-of-bag samples for tree T_i in a forest are those training examples that are NOT used to construct tree T_i
- Out-of-bag samples can be used for model selection. They give unbiased estimates of:
 - Error on future data
 - **Don't need to use cross validation!!!!**
 - Internal strength
 - Internal correlation

31

R.I. (Random Input) Forests

- For K trees:
 - Build each tree by:
 - Selecting, at random, at each node a small set of features (F) to split on. Common values of F are

$$F = 1$$

$$F = \log(M) + 1$$
 - For each node split on the best of this subset
 - Grow tree to full length

Regression: average over trees

Classification: vote

32

R.C. (Random Combination) Forests

- For K trees:
 - Build each tree by:
 - Create F random linear sums of L variables

$$A_f = \sum_{i=1}^L b_{fi} x_{ind(i)}, \dots, A_F = \sum_{i=1}^L b_{Fi} x_{ind(i)}$$

$$b_{fi} = \text{uniform random } [-1, 1]$$
 - At each node split on the best of these linear boundaries
 - Grow tree to full length

Regression: average over trees

Classification: vote

33

Classification Data:

Data set	Train size	Test size	Inputs	Classes
Glass	214	-	9	6
Breast cancer	699	-	9	2
Diabetes	768	-	8	2
Sonar	208	-	60	2
Vowel	990	-	10	11
Ionosphere	351	-	34	2
Vehicle	846	-	18	4
Soybean	685	-	35	19
German credit	1000	-	24	2
Image	2310	-	19	7
Ecoli	336	-	7	8
Votes	435	-	16	2
Liver	345	-	6	2
Letters	15000	5000	16	26
Sat-images	4435	2000	36	6
Zip-code	7291	2007	256	10
Waveform	300	3000	21	3
Twonorm	300	3000	20	2
Threonorm	300	3000	20	2
Ringnorm	300	3000	20	2

34

Results: RI

Data set	Adaboost	Selection	Forest-RI single input	One tree
Glass	22.0	20.6	21.2	36.9
Breast cancer	3.2	2.9	2.7	6.3
Diabetes	26.6	24.2	24.3	33.1
Sonar	15.6	15.9	18.0	31.7
Vowel	4.1	3.4	3.3	30.4
Ionosphere	6.4	7.1	7.5	12.7
Vehicle	23.2	25.8	26.4	33.1
German credit	23.5	24.4	26.2	33.3
Image	1.6	2.1	2.7	6.4
Ecoli	14.8	12.8	13.0	24.5
Votes	4.8	4.1	4.6	7.4
Liver	30.7	25.1	24.7	40.6
Letters	3.4	3.5	4.7	19.8
Sat-images	8.8	8.6	10.5	17.2
Zip-code	6.2	6.3	7.8	20.6
Waveform	17.8	17.2	17.3	34.0
Twonorm	4.9	3.9	3.9	24.7
Threonorm	18.8	17.5	17.5	38.4
Ringnorm	6.9	4.9	4.9	25.7

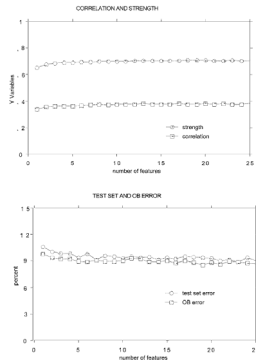
35

Results: RC

Data set	Adaboost	Forest-RC		
		Selection	Two features	One tree
Glass	22.0	24.4	23.5	42.4
Breast cancer	3.2	3.1	2.9	5.8
Diabetes	26.6	23.0	23.1	32.1
Sonar	15.6	13.6	13.8	31.7
Vowel	4.1	3.3	3.3	30.4
Ionosphere	6.4	5.5	5.7	14.2
Vehicle	23.2	23.1	22.8	39.1
German credit	23.5	22.8	23.8	32.6
Image	1.6	1.6	1.8	6.0
Ecoli	14.8	12.9	12.4	25.3
Votes	4.8	4.1	4.0	8.6
Liver	30.7	27.3	27.2	40.3
Letters	3.4	3.4	4.1	23.8
Sat-images	8.8	9.1	10.2	17.3
Zip-code	6.2	6.2	7.2	22.7
Waveform	17.8	16.0	16.1	33.2
Twonorm	4.9	3.8	3.9	20.9
Threonorm	18.8	16.8	16.9	34.8
Ringnorm	6.9	4.8	4.6	24.6

36

Strength, Correlation and Out-of-Bag Error



37

Adding Noise: random 5% of training data with outputs flipped

Data set	Adaboost	Forest-RI	Forest-RC
Glass	1.6	.4	-.4
Breast cancer	43.2	1.8	11.1
Diabetes	6.8	1.7	2.8
Sonar	15.1	-6.6	4.2
Ionosphere	27.7	3.8	5.7
Soybean	26.9	3.2	8.5
Ecoli	7.5	7.9	7.8
Votes	48.9	6.3	4.6
Liver	10.3	-.2	4.8

Percent increase in error

RFs robust To noise!

38

Random Forests Regression: The Data

Data set	Nr. inputs	#Training	#Test
Boston Housing	12	506	10%
Ozone	8	330	10%
Servo	4	167	10%
Abalone	8	4177	25%
Robot Arm	12	15,000	5000
Friedman#1	10	200	2000
Friedman#2	4	200	2000
Friedman#3	4	200	2000

39

Results: RC 25 Features 2 Random Inputs

Mean Square Test Error

Data set	Bagging	Adapt. bag	Forest
Boston Housing	11.4	9.7	10.2
Ozone	17.8	17.8	16.3
Servo $\times 10 - 2$	24.5	25.1	24.6
Abalone	4.9	4.9	4.6
Robot Arm $\times 10 - 2$	4.7	2.8	4.2
Friedman #1	6.3	4.1	5.7
Friedman #2 $\times 10 + 3$	21.5	21.5	19.6
Friedman #3 $\times 10 - 3$	24.8	24.8	21.6

40

Results: Test Error and OOB Error

Data Set	Test error	OB error	PE ² (tree)	Cor.
Boston Housing	10.2	11.6	26.3	.45
Ozone	16.3	17.6	32.5	.55
Servo $\times 10 - 2$	24.6	27.9	56.4	.56
Abalone	4.6	4.6	8.3	.56
Robot Arm $\times 10 - 2$	4.2	3.7	9.1	.41
Friedman #1	5.7	6.3	15.3	.41
Friedman #2 $\times 10 + 3$	19.6	20.4	40.7	.51
Friedman #3 $\times 10 - 3$	21.6	22.9	48.3	.49

41

Random Forests Regression: Adding Noise to Output vs. Bagging

Data set	With bagging	With Noise
Boston Housing	10.2	9.1
Ozone	17.8	16.3
Servo $\times 10 - 2$	24.6	23.2
Abalone	4.6	4.7
Robot Arm $\times 10 - 2$	4.2	3.9
Friedman #1	5.7	5.1
Friedman #2 $\times 10 + 3$	19.6	20.4
Friedman #3 $\times 10 - 3$	21.6	19.8

42

Random Forests Summary

- Adding the right kind of noise is good!
- Inject randomness such that
 - Each model has maximal strength
 - i.e. a fairly good model on its own
 - Each model has minimum correlation with the other models
 - i.e. the errors tend to cancel out

43

Polynomial Cascade Regression

- My Ph.D. work (1996)
- Another example of effective use of randomization

44

Problem Domain Characteristics

- thousands of relevant input variables
 - each contributing a small but significant amount to the final model
- no subset of these variables can adequately describe the desired function
- the relevant variables are confounded by thousands of irrelevant variables

45

Why is this a difficult domain?

- Very large input space!
- Problem is intrinsically nonparametric
 - Don't know which inputs are significant
 - Don't know an optimal model structure
- Problems are in general nonlinear

46

Polynomial Cascade Algorithm: Conceptual Motivation (IJCAI 97)

- **Problem # 1:** Simultaneous construction of model using all input is not generally feasible.
 - **Soln:** use low dimensional projections (building blocks).
 - *simplest approach:* 2 dimensional: $g_l(u, v)$
- **Problem # 2:** Finding the *best* low dimensional projection infeasible.
 - **Soln:** Don't find the *best* - use selection criteria which are independent of dimension.
 - *simplest approach:* random building block selection.

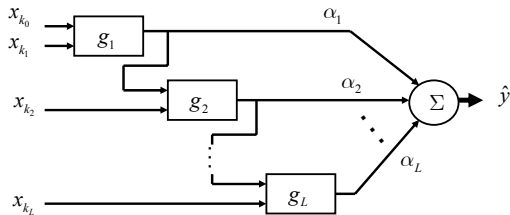
47

PC Algorithm: Conceptual Motivation (continued)

- **Problem # 3:** Low dimensional projections tend to be flat (i.e. $g_l(u, v) = C$).
 - **Soln:** Subdivide the input space.
 - *simplest approach:* random subdivision (bootstrap samples).

48

Polynomial Cascade Structure



49

Main PC Algorithm Characteristics

1. Building blocks (3rd order polynomials)

$$g_i(u, v) = a_{00} + a_{01}v + a_{10}u + a_{11}uv + a_{20}u^2 + a_{02}v^2 + a_{21}u^2v + a_{12}uv^2 + a_{30}u^3 + a_{03}v^3$$
2. $g_i(u, v)$ added one at a time, in order
3. Random (*repeated*) order of inputs:

$$(x_1, \dots, x_d)$$
4. $g_i(u, v)$ constructed using a bootstrap sample

50

PC Algorithm:

STEP 1: Initialize algorithm:

- Learning data divided into *training* set and *validation* set
- Random order of inputs

STEP 2: Construct new section: (Multiple levels)

- Use bootstrap sample to fit $g_i(u, v)$
- Set α_i to the normalized inverse MSE of $g_i(u, v)$ on *training* set
- Stop when error on *validation* set *stops* decreasing

51

PC Algorithm: (continued)

STEP 3: Prune section: Prune back to the block which has smallest error on the validation set

STEP 4: Update learning outputs. Replace outputs with residual errors:

$$y \leftarrow y - \sum_{l=i}^s \alpha_l g_l$$

STEP 4: Check stopping condition: GOTO STEP 2 if further error reduction is possible, otherwise STOP

52

Why does PC work?

- Over fitting avoided via appropriate injection of randomness: i.e like Random Forests (Breiman, 1999)
 - Bootstrap sampling
 - Random order of inputs
- Irrelevant inputs *not* excluded from cascade
 - Treated as noise and averaged out
 - No explicit variable selection is used

53

Why does PC work? (continued)

- Produces stable high dimensional models
 - Projections onto 2 dimensional structures
- Low dimensional projections are unlikely to be flat $g_i(u, v) \equiv C$
 - Bootstrap sampling avoids
 - PC algorithm effectively deals with parity problems of greater than 2 dimensions
 - e.g. 10 bit parity problem where $g_i(u, v) \equiv C$, for all levels, without random sampling

54

PC Effective on Low Dimensional Problems (surprise?)

- Does as well or better than most algorithms on low dimensional regression problems (IJCAI97)
- Produces competitive models without the need for parameter tuning or kernel selection
- **HOWEVER:**
 - Models are not sparse!

55

PC Algorithm Evaluation on Standard Regression Data

Table 1: Low Dimensional Data Sets

Published Source	Data	Pub. Error	Prop. Alg. Error (100/10 runs)		
			best	ave.	s.d.
Breiman [Breiman, 1996]	Housing	11.6	1.95	9.4	8.4
	Friedman 1	6.1	1.77	2.88	0.43
	Friedman 2	22,100	16,733	19,603	1413
	Friedman 3	0.0242	0.014	0.0189	0.0024
Rasmussen [Rasmussen, 1996]	Auto Price	(0.29)	0.21	0.29	0.03
	Cpu	(0.17)	0.086	0.14	0.025
	House	(0.15)	0.128	0.145	0.007
	Mpg	(0.10)	0.088	0.100	0.004
	Servo	(0.15)	0.198	0.25	0.02
Jordan and Jacobs [Jordan and Jacobs, 1994]	For. Dyn.	(0.09)	0.0378	0.041	0.009

56

Theoretical Results

1. PC's are universal approximators
2. Conditions for convergence to zero error:
 - Uncorrelated errors from level to level
 - Similar to bagging and random forests (Breiman)
3. Rate of convergence (to some local error minimum), as a function of the number of learning examples, is independent of the dimension of the input space
4. Computational complexity is linear in the number of training examples

57