Don't ask what it means, but rather how it is used.

- L. Wittgenstein

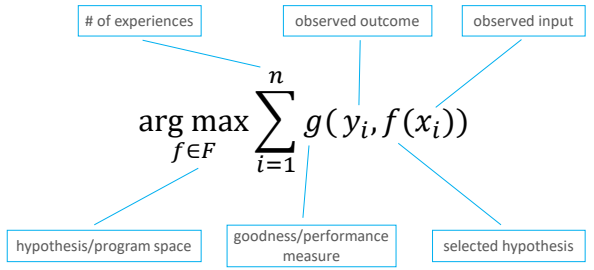
CSE 552
Machine Learning

Spring 2025

Ensemble Learning

1

Supervised Machine Learning



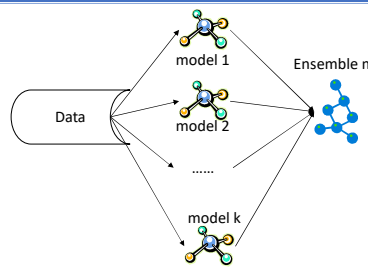
2

Slide Credits

- Gao et al.: Jing Gao, Wei Fan, Jiawei Han, http://ews.uiuc.edu/~jinggao3/sdm10ensemble.htm
- Holloway: Todd Holloway, 2007.

3

Ensemble

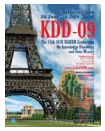


Combine multiple models into one!

Applications: classification, clustering, collaborative filtering, anomaly detection……

4

Stories of Success



- Million-dollar prize
  - Improve the baseline movie recommendation approach of Netflix by 10% in accuracy
  - The top submissions all combine several teams and algorithms as an ensemble
    - Data mining competitions
      - Classification problems
      - Winning teams employ an ensemble of classifiers

Netflix Prize

- Supervised learning task
  - Training data is a set of users and ratings (1,2,3,4,5 stars) those users have given to movies.
  - Construct a classifier that given a user and an unrated movie, correctly classifies that movie as either 1, 2, 3, 4, or 5 stars
  - $1 million prize for a 10% improvement over Netflix's current movie recommender (MSE = 0.9514)
- Competition
  - Began October 2006
  - At first, single-model methods are developed, and performances are improved
  - However, improvements slowed down
  - Later, individuals and teams merged their results, and significant improvements are observed

Netflix Prize

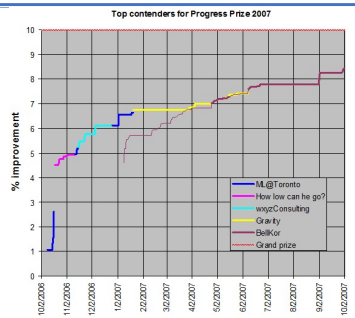
Just three weeks after it began, at least 40 teams had bested the Netflix classifier.

Top teams showed about 5% improvement.

Team Name	Best Score	% Improvement
No Grand Prize candidates yet	--	--
Grand Prize - RMSE <= 0.8563		
How low can he go?	0.9046	4.92
ML@UToronto A	0.9046	4.92
ssorkin	0.9089	4.47
wxyzconsulting.com	0.9103	4.32
The Thought Gang	0.9113	4.21
NIPS Reject	0.9118	4.16
simonfunk	0.9145	3.88
Bozo_The_Clown	0.9177	3.54
Elliptic Chaos	0.9179	3.52
datcracker	0.9183	3.48
Foreseer	0.9214	3.15
bsdfish	0.9229	3.00
Three Blind Mice	0.9234	2.94
Bocsimacko	0.9238	2.90
Remco	0.9252	2.75
karmatics	0.9301	2.24
Chapelator	0.9314	2.10
Fimod	0.9325	1.99
mthrox	0.9328	1.96

Netflix Prize

However, improvement slowed…



from http://www.research.att.com/~volinsky/netflix/

Netflix Prize

Today, the top team has posted a 8.5% improvement.

Ensemble methods are the best performers…

	No Progress Prize candidates yet		
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72

9

Netflix Prize

Rookies

"Thanks to Paul Harrison's collaboration, a simple mix of our solutions improved our result from 6.31 to 6.75"

	No Progress Prize candidates yet		
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72

10

Netflix Prize

Arek Paterek

"My approach is to **combine the results of many methods** (also two-way interactions between them) using linear regression on the test set. The best method in my ensemble is regularized SVD with biases, post processed with kernel ridge regression"

http://rainbow.mimuw.edu.pl/~ap/ap_kdd.pdf

	No Progress Prize candidates yet		
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72

11

Netflix Prize

U of Toronto

"When the predictions of **multiple** RBM models and **multiple** SVD models are linearly combined, we achieve an error rate that is well over 6% better than the score of Netflix's own system."

http://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf

	No Progress Prize candidates yet		
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72

12

Netflix Prize

Gravity

Table 5: Best results of single approaches and their combinations

Method/Combination	RMSE
MF	0.9190
NB	0.9313
CL	0.9606
NB + CL	0.9275
MF + CL	0.9137
MF + NB	0.9089
MF + NB + CL	0.9089

home.mit.bme.hu/~gtakacs/download/**gravity**.pdf

--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72

13

Netflix Prize

When Gravity and Dinosaurs Unite

"Our common team blends the result of team Gravity and team Dinosaur Planet."

Might have guessed from the name…

--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72

14

Netflix Prize

BellKor / KorBell

And, yes, the top team which is from AT&T…

"Our final solution (RMSE=0.8712) consists of blending 107 individual results. "

--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72

15

Leaderboard

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
	Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor's Pragmatic Chaos			
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8611	9.59	2009-06-12 08:20:22
9	Feeds2	0.8612	9.58	2009-05-24 23:06:52
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
	Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos			
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
	Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell			
	Cinematch score - RMSE = 0.9525			

"Our final solution (RMSE=0.8712) consists of blending 107 individual results. "

"Predictive accuracy is substantially improved when blending multiple predictors. Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a single technique. "

16

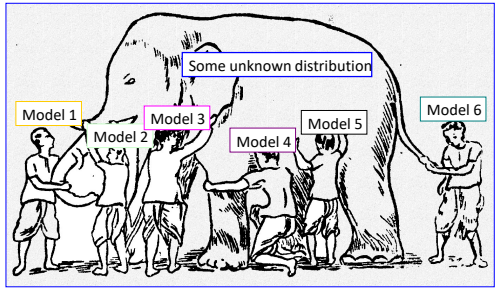
Motivations

- Motivations of ensemble methods
  - Ensemble model improves accuracy and robustness over single model methods
  - Applications:
    - distributed computing
    - privacy-preserving applications
    - large-scale data with reusable models
    - multiple sources of data
  - Efficiency: a complex problem can be decomposed into multiple sub-problems that are easier to understand and solve (divide-and-conquer approach)

Why Ensemble Works?

- Intuition
  - combining diverse, independent opinions in human decision-making as a protective mechanism (e.g. stock portfolio)
- Uncorrelated error reduction
  - Suppose we have 5 completely independent classifiers for majority voting
  - If accuracy is 70% for each
    - 10 (.7^3)(.3^2)+5(.7^4)(.3)+(.7^5)
    - **83.7% majority vote accuracy**
  - 101 such classifiers
    - **99.9% majority vote accuracy**

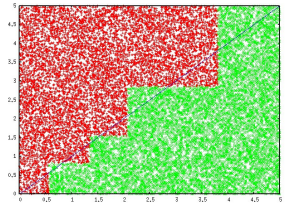
Why Ensemble Works?

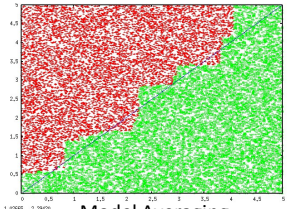


Ensemble gives the global picture!

Why Ensemble Works?

- Overcome limitations of single hypothesis
  - The target function may not be implementable with individual classifiers, but may be approximated by model averaging


Decision Tree


Model Averaging

Intuitions

- Utility of combining diverse, independent opinions in human decision-making
  - Protective Mechanism (e.g. stock portfolio diversity)
- **Violation of Ockham's Razor**
  - Identifying the best model requires identifying the proper "model complexity"

See Domingos, P. Occam's two razors: the sharp and the blunt. KDD. 1998.

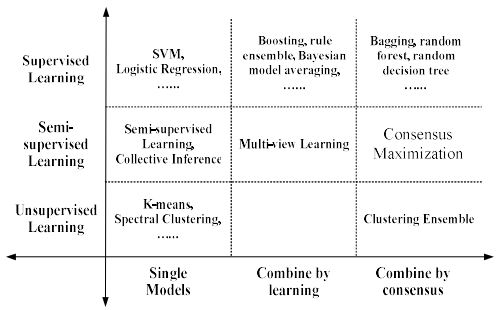
Intuitions

Majority vote

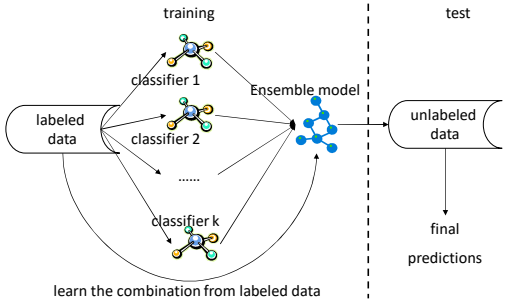
Suppose we have 5 completely independent classifiers…

- If accuracy is 70% for each
  - $10(.7^3)(.3^2)+5(.7^4)(.3)+(.7^5)$
  - **83.7% majority vote accuracy**
- 101 such classifiers
  - **99.9% majority vote accuracy**

Summary

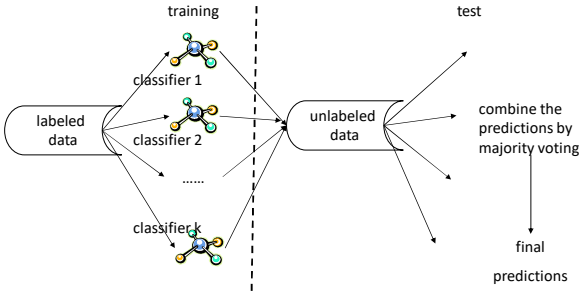


Ensemble of Classifiers—Learn to Combine



Algorithms: boosting, stacked generalization, rule ensemble, Bayesian model averaging…

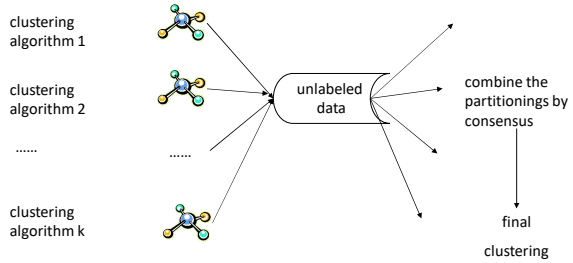
Ensemble of Classifiers—Consensus



Algorithms: bagging, random forest, random decision tree, model averaging of probabilities…

25

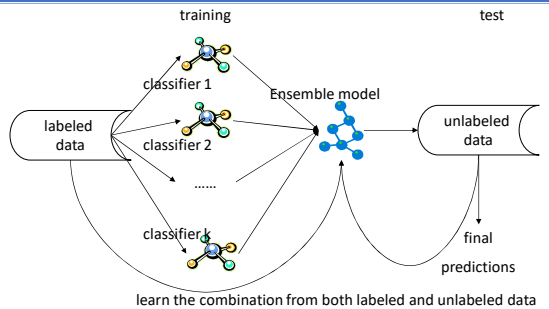
Clustering Ensemble—Consensus



Algorithms: direct approach, object-based, cluster-based, object-cluster-based approaches, generative models

26

Semi-Supervised Ensemble-Learn to Combine

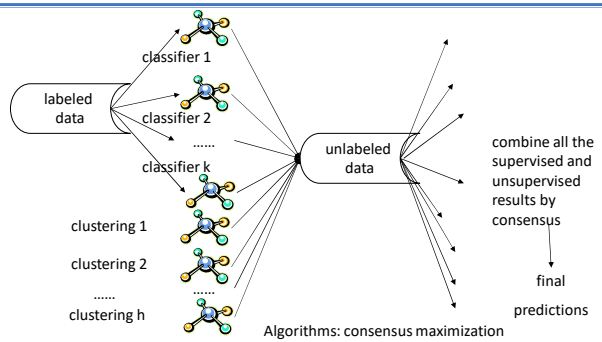


learn the combination from both labeled and unlabeled data

Algorithms: multi-view learning

27

Semi-supervised Ensemble—Consensus



Algorithms: consensus maximization

28

Pros and Cons

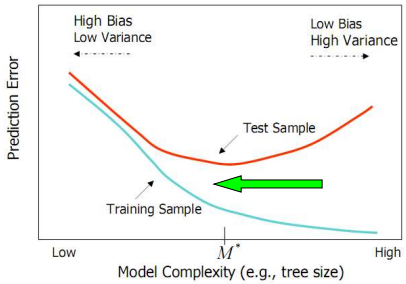
	Combine by learning	Combine by consensus
Pros	Get useful feedbacks from labeled data Can potentially improve accuracy	Do not need labeled data Can improve the generalization performance
Cons	Need to keep the labeled data to train the ensemble May overfit the labeled data Cannot work when no labels are available	No feedbacks from the labeled data Require the assumption that consensus is better

Supervised Ensemble Methods

- Problem
  - Given a data set $D=\{x_1,x_2,\ldots,x_n\}$ and their corresponding labels $L=\{l_1,l_2,\ldots,l_n\}$
- An ensemble approach computes:
  - A set of classifiers $\{f_1,f_2,\ldots,f_k\}$, each of which maps data to a class label: $f_j(x)=l$
  - A combination of classifiers f^* which minimizes generalization error: $f^*(x)= w_1f_1(x)+ w_2f_2(x)+\ldots+ w_kf_k(x)$

Bias and Variance

- Ensemble methods
  - Combine learners to reduce variance



Generating Base Classifiers

- Sampling training examples
  - Train k classifiers on k subsets drawn from the training set
- Using different learning models
  - Use all the training examples, but apply different learning algorithms
- Sampling features
  - Train k classifiers on k subsets of features drawn from the feature space
- Learning "randomly"
  - Introduce randomness into learning procedures

BAGGING

33

Main types of Ensemble methods

Combine multiple models together

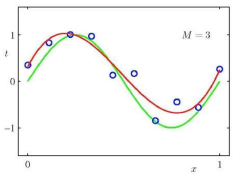
- Committees – Bagging
  - *Regression*: Take an average of the predictions made by each model
  - *Classification*: Make classification by voting over a collection of classifiers
- Boosting – Adaboost
  - Train multiple models in sequence
- Decision trees
  - Different models are responsible for making predictions in different regions of input space

34

Committees [1]

Construct a committee ----
average the predictions of a set of individual models

For example, regression problem:



- T(x) = sin 2PI x + noise
- N = 10
- Polynomial, order 3

35

Polynom Fitting

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

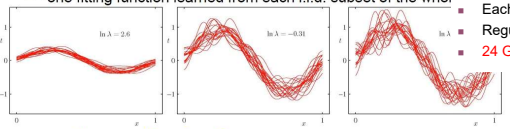
where λ is the regularization parameter

36

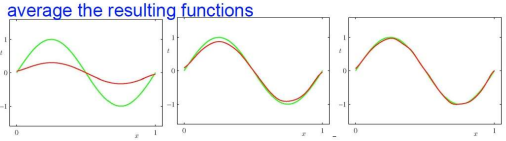
Committees [2]

train multiple polynomials

-- one fitting function learned from each i.i.d. subset of the whol



average the resulting functions



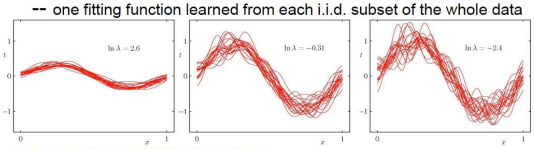
- T(x) = sin 2Pl x + noise
- L = 100 data sets → 100 fits (only 20 shown)
- Each dataset: N = 25 data points
- Regularisation factor λ
- 24 Gaussian basis functions (+bias)

37

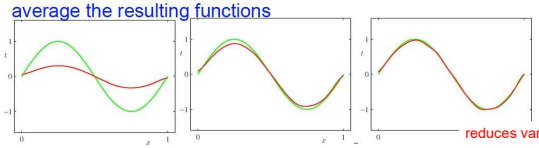
Committees [3]

train multiple polynomials

-- one fitting function learned from each i.i.d. subset of the whole data



average the resulting functions



reduces variance and helps to avoid overfitting

38

Error Difference [1]

- true regression function h(x),
- output of each model learned from one of the M data sets,

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x})$$

"bootstrap"

- average error made by the models acting individually

$$E_{\mathrm{AV}} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

- Committee prediction

$$y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$

- the expected error from the committee

$$E_{\mathrm{COM}} = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\mathbf{x})\right\}^2\right]$$

39

Error Difference [2]

- If we assume that the errors have zero mean and are uncorrelated, so that

$$\begin{aligned}\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\right] &= 0\\ \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})\right] &= 0, \qquad m \neq l\end{aligned}$$

- then we obtain

$$E_{\mathrm{COM}} = \frac{1}{M}E_{\mathrm{AV}}$$

The average error of a model can be reduced by a factor of *M* simply by averaging *M* versions of the model.

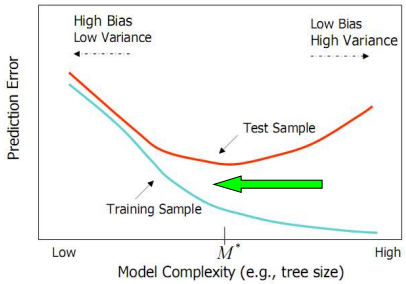
But, the assumption may not be true. The errors are typically highly correlated, but still

$$E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}$$

40

Bias and Variance

- Ensemble methods
  - Combine learners to reduce variance



41

Main types of Ensemble methods

Combine multiple models together

- Committees – Bagging
  - *Regression*: Take an average of the predictions made by each model
  - *Classification*: Make classification by voting over a collection of classifiers
- Boosting – Adaboost
  - Train multiple models in sequence
- Decision trees
  - Different models are responsible for making predictions in different regions of input space

42

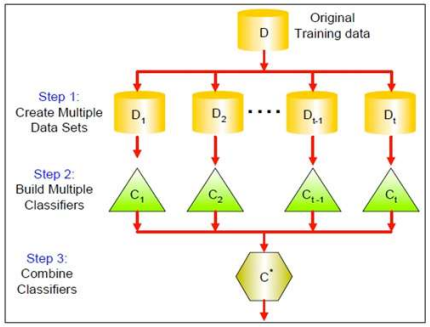
Bagging

- Bootstrap
  - Sampling (random) with replacement
  - Contains around 63.2% original records in each sample, the rest dublicates
- Bootstrap Aggregation
  - Train a classifier on each bootstrap sample
  - Use majority voting to determine the class label of ensemble classifier
- Note that we only have 1 single dataset

*[Breiman96]

43

Bagging



44

The Bagging Algorithm

BAGGING

Training phase

1. Initialize the parameters
   - $\mathcal{D} = \emptyset$, the ensemble.
   - L, the number of classifiers to train.
2. For $k = 1, \ldots, L$
   - Take a bootstrap sample S_k from $\mathbf{Z}$.
   - Build a classifier D_k using S_k as the training set.
   - Add the classifier to the current ensemble, $\mathcal{D} = \mathcal{D} \cup D_k$.
3. Return $\mathcal{D}$.

Classification phase

4. Run $D_1, \ldots, D_L$ on the input $\mathbf{x}$.
5. The class with the maximum number of votes is chosen as the label for $\mathbf{x}$.

45

Bagging

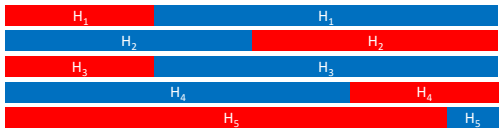
- Works for unstable base classifiers
  - Neural Networks
  - Decision Trees
  - KNN
- Sampling with replacement…
  - Can emphasize replaced samples

46

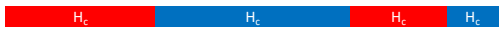
Bagging

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

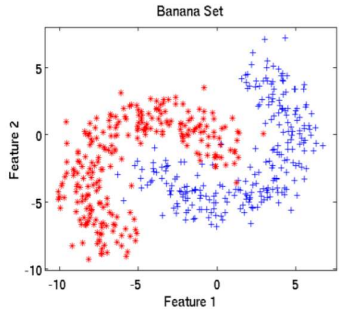


Combine predictions by majority voting



47

Bagging – Example 2

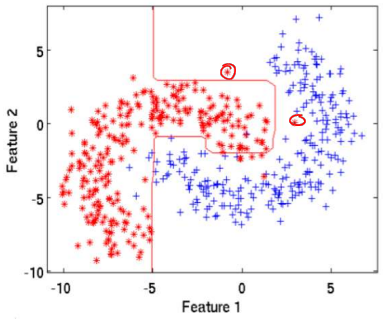


Training data…

Base classifiers:
Decision trees

48

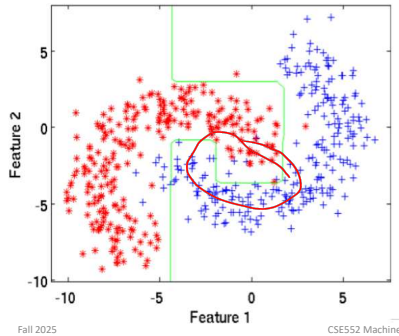
Bagging – Example 2



Decision boundaries produced by first tree…

49

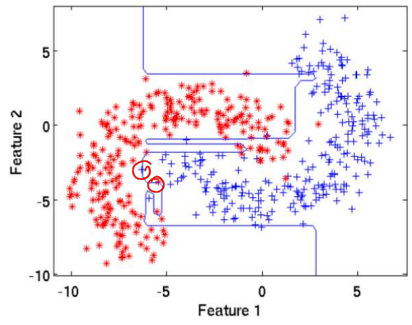
Bagging – Example 2



Decision boundaries produced by second tree…

50

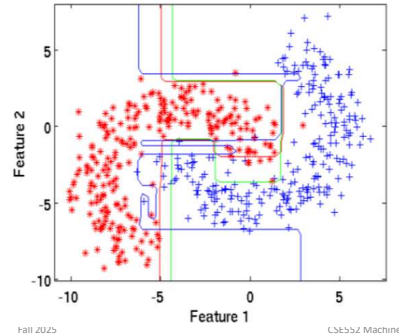
Bagging – Example 2



Decision boundaries produced by third tree…

51

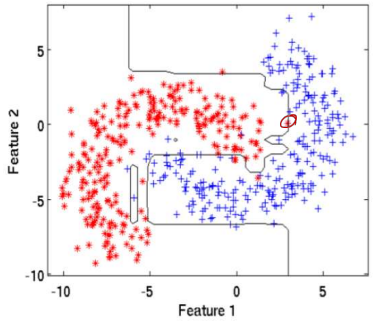
Bagging – Example 2



Decision boundaries produced by all trees…

52

Bagging – Example 2



Final decision boundaries produced by bagging all trees…

53

When Bagging Works

- Expected error is the expected discrepancy between the estimated $\bar{f}(x) = E[f(x)]$ and true function $\hat{f}(x)$

$$E\left[\left(\hat{f}(x) - E[f(x)]\right)^2\right]$$

- Bias is squared discrepancy between averaged estimated and true function

$$\left(E[\hat{f}(x)] - E[f(x)]\right)^2$$

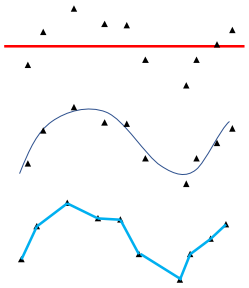
- Variance is expected divergence of the estimated function vs. its average value

$$E[\left(\hat{f}(x) - E[f(x)]\right)^2]$$

54

When Bagging Works

- Under-fitting:
  - High bias (models are not accurate)
  - Small variance (smaller influence of examples in the training set)
- Over-fitting:
  - Small bias (models flexible enough to fit well to training data)
  - Large variance (models depend very much on the training set)



55

Averaging Decreases Variance

- Example
  - Assume we measure a random variable x with a $N(\mu, \sigma^2)$ distribution
  - If only one measurement x_1 is done,
    - The expected mean of the measurement is μ
    - Variance is $\mathrm{Var}(x_1) = \sigma^2$
- If random variable x is measured K times $(x_1, x_2, \dots, x_k)$ and the value is estimated as: $(x_1, x_2, \dots, x_K)/K$,
  - Mean of the estimate is still μ
  - But, variance is smaller:

$$\frac{[\mathrm{Var}(x_1) + \mathrm{Var}(x_2) + \cdots + \mathrm{Var}(x_K)]}{K^2} = \frac{K\sigma^2}{K^2} = \frac{\sigma^2}{K}$$

- Observe: Bagging is a kind of averaging!

56

When Bagging Works

- Main property of Bagging (no proof will be given)
  - Bagging decreases variance of the base model without changing the bias!!!
  - Why? Averaging!
- Bagging typically helps
  - When applied with an over-fitted base model
    - High dependency on actual training data
- It does not help much
  - High bias. When the base model is robust to the changes in the training data (due to sampling)

Boosting

Main types of Ensemble methods

Combine multiple models together

- Committees – Bagging
  - *Regression*: Take an average of the predictions made by each model
  - *Classification*: Make classification by voting over a collection of classifiers
- Boosting – Adaboost
  - Train multiple models in sequence
- Decision trees
  - Different models are responsible for making predictions in different regions of input space

Boosting

- Principles
  - Boost a set of weak learners to a strong learner
  - Make records currently misclassified more important
- Example
  - Record 4 is hard to classify
  - Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

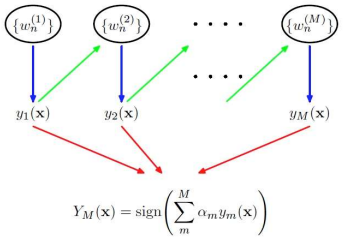
*[FrSc97]

Boosting

- AdaBoost
  - Initially, set uniform weights on all the records
  - At each round
    - Create a bootstrap sample based on the weights
    - Train a classifier on the sample and apply it on the original training set
    - Records that are wrongly classified will have their weights increased
    - Records that are classified correctly will have their weights decreased
    - If the error rate is higher than 50%, start over
  - Final prediction is weighted average of all the classifiers with weight representing the training accuracy

61

Boosting



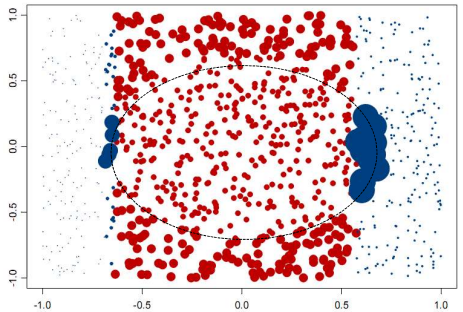
62

Boosting

- Determine the weight
  - For classifier *i*, its error is $\varepsilon_i = \frac{\sum_{j=1}^{N} w_j \delta(C_i(x_j) \neq y_j)}{\sum_{j=1}^{N} w_j}$
  - The classifier's importance is represented as: $\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$
  - The weight of each record is updated as: $w_j^{(i+1)} = \frac{w_j^{(i)} \exp\left(-\alpha_i y_j C_i(x_j)\right)}{Z^{(i)}}$
  - Final combination: $C^*(x) = \arg\max_y \sum_{i=1}^{K} \alpha_i \delta(C_i(x) = y)$

63

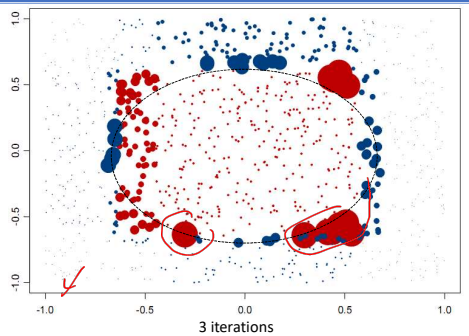
Example



Classifications (colors) and weights (size) after *1 iteration* of AdaBoost

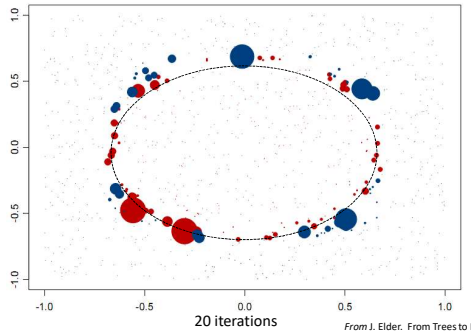
64

Example



65

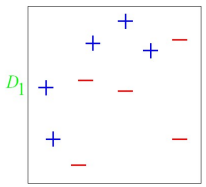
Example



From J. Elder. From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods. 2007.

66

AdaBoost - Example

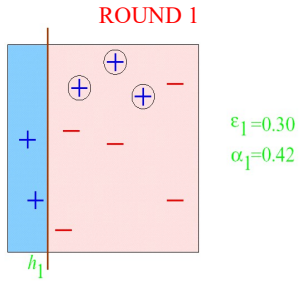


Round 0: Equal Weights to all training samples

Taken from "A Tutorial on Boosting" by Yoav Freund and Rob Schapire

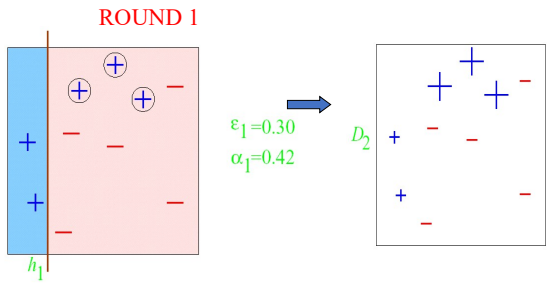
67

AdaBoost - Example



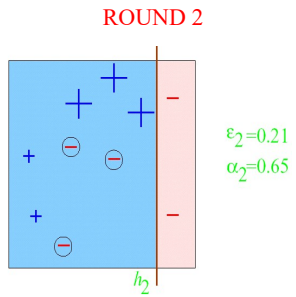
68

AdaBoost - Example



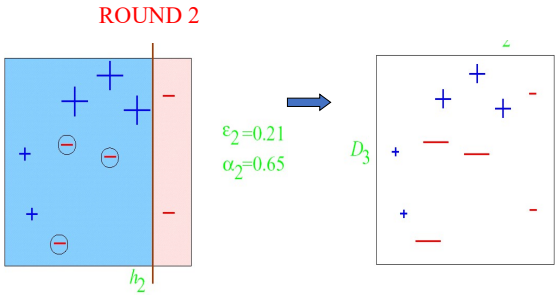
69

AdaBoost - Example



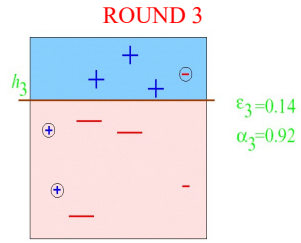
70

AdaBoost - Example



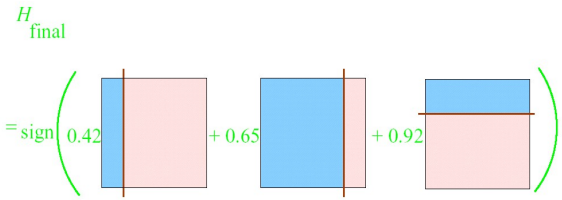
71

AdaBoost - Example



72

AdaBoost - Example



73

Boosting – Theoretical Foundations

- PAC: Probably Approximately Correct framework
  - $\epsilon - \delta$ solution
- PAC learning:
  - Learning with the pre-specified accuracy ϵ and confidence δ
  - The probability that the misclassification error is larger than ϵ is smaller than δ

$$P(ME(c) > \epsilon) \leq \delta$$

- Accuracy(ϵ): Percent of correctly classified samples in test
- Confidence(δ): The probability that in one experiment some accuracy will be achieved

74

PAC Learnability

Strong (PAC) learnability

- There exists a learning algorithm that **efficiently** learns the classification with a pre-specified **accuracy** and **confidence**

Strong (PAC) learner:

- A learning algorithm P that given an arbitrary
  - classification error $\epsilon(< \frac{1}{2})$, and
  - confidence $\delta(< \frac{1}{2})$
- Outputs a classifier
  - With a classification accuracy $> (1 - \epsilon)$
  - A confidence probability $> (1 - \delta)$
  - And runs in time polynomial in $\frac{1}{\delta}, \frac{1}{\epsilon}$
    - Implies: number of samples N is polynomial in $\frac{1}{\delta}, \frac{1}{\epsilon}$

75

Weak Learner

Weak learner:

- A learning algorithm (learner) W
  - Providing classification accuracy $> (1 - \epsilon_0)$
  - With probability $> (1 - \delta_0)$
- For some **fixed** and **uncontrollable**
  - Classification error $\epsilon_0 \left(< \frac{1}{2}\right)$
  - Confidence $\delta_0 \left(< \frac{1}{2}\right)$
- And this on an arbitrary distribution of data entries

76

Weak Learnability = Strong (PAC) Learnability

- Assume there exists a weak learner
  - it is better that a random guess (50%) with confidence higher than 50% on any data distribution
- Question:
  - Is problem also PAC-learnable?
  - Can we generate an algorithm P that achieves an arbitrary $\epsilon - \delta$ accuracy?
- Why is this important?
- Usual classification methods (decision trees, neural nets), have specified, but uncontrollable performances
- Can we improve performance to achieve pre-specified accuracy (confidence)?

Weak=Strong Learnability!!!

- Proof due to R. Schapire
  - An arbitrary $(\epsilon - \delta)$ improvement is possible
- Idea: combine multiple weak learners together
  - Weak learner W with confidence δ_0 and maximal error ϵ_0
  - It is possible:
    - To improve (boost) the confidence
    - To improve (boost) the accuracy

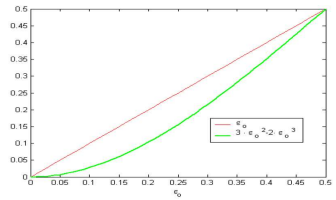
  by training different weak learners on slightly different datasets

Boosting Accuracy

- Training
  - Sample randomly from the distribution of examples
  - Train hypothesis H_1 on the sample
  - Evaluate accuracy of H_1 on the distribution
  - Sample randomly such that for the half of samples H_1 provides correct, and for another half, incorrect results; Train hypothesis H_2.
  - Train H_3 on samples from the distribution where H_1 and H_3 classify differently
- Test
  - For each example, decide according to the majority vote of H_1, H_2 and H_3

Theorem

- If each hypothesis has an error ϵ_0, the final classifier has error $< g(\epsilon_0) = 3{\epsilon_0}^2 - 2{\epsilon_0}^3$
- Accuracy improved !!!!
- Apply recursively to get to the target accuracy !!!



Theoretical Boosting Algorithm

- Similar to boosting the accuracy we can boost the confidence at some restricted accuracy cost
- **The key result**: we can improve both the accuracy and confidence
- Problems with the theoretical algorithm
  - A good (better than 50 %) classifier on all data problems
  - We cannot properly sample from data-distribution
  - Method requires large training set
- Solution to the sampling problem:
  - Boosting by sampling
    - AdaBoost algorithm and variants

81

Boosting - AdaBoost

- In practice overfitting rarely occurs (Bishop)

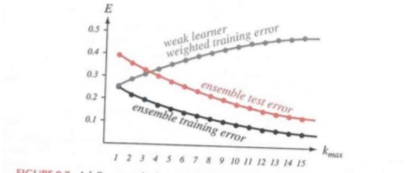


FIGURE 9.7. AdaBoost applied to a weak learning system can reduce the training error E exponentially as the number of component classifiers, k_{max}, is increased. Because AdaBoost "focuses on" *difficult* training patterns, the training error of each successive component classifier (measured on its own weighted training set) is generally larger than that of any previous component classifier (shown in gray). Nevertheless, so long as the component classifiers perform better than chance (e.g., have error less than 0.5 on a two-category problem), the weighted ensemble decision of Eq. 36 ensures that the training error will decrease, as given by Eq. 37. It is often found that the test error decreases in boosted systems as well, as shown in red.

82

Boosting

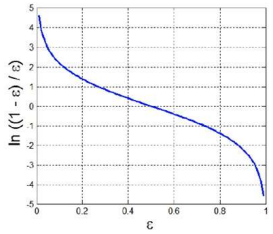
- Explanation
  - Among the classifiers of the form:

$$f(x) = \sum_{i=1}^{K} \alpha_i C_i(x)$$

  - The classifier's importance is represented as:

$$\alpha_i = \frac{1}{2} \ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

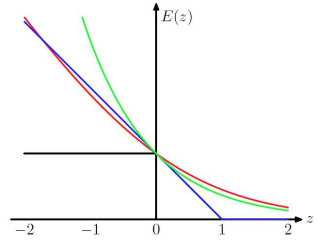
  - We seek to minimize the exponential loss function: $\sum_{j=1}^{N} e^{-y_j f(x_j)}$



83

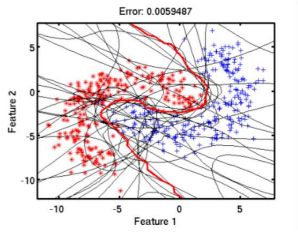
AdaBoost – Loss Function

- Sequential minimization of the exponential error function
  - Penalizes large negative values heavily
  - Not robust to outliers!
  - No probabilistic interpretation
  - Only for 2 classes

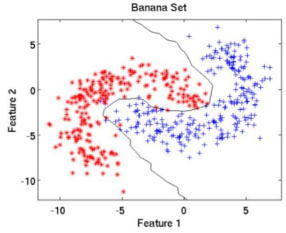


84

Boosting Example Again



AdaBoost using 20 neural nets
[bpxnc] default settings



Final output of AdaBoost with 20 neural nets

85

Boosting vs Bagging Example

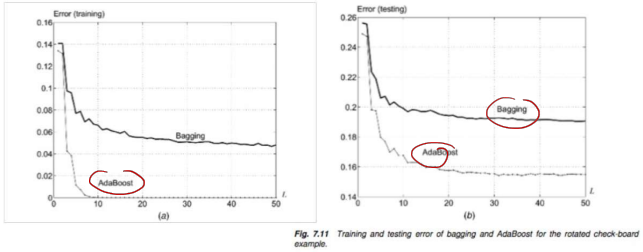


Fig. 7.11 Training and testing error of bagging and AdaBoost for the rotated check-board example.

Krogh 95

86

Boosting vs Bagging

- Committee/Bagging
  - Base classifiers are trained in parallel on samples of data set
- Boosting
  - Base classifiers are trained in sequence by using weighted form of the data set
  - Weights for each data point depend on the performance of the previous classifiers
  - Misclassified points are given greater weight when used to train the next classifier in the sequence
  - Boosting tends to achieve greater accuracy, but also risks overfitting the model to misclassified data

87

Boosting and Computer Vision

88

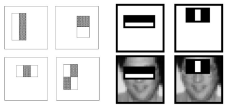
Example: Face Detection

- Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:
  - Regular 2D structure
  - Center of face almost shaped like a "patch"/window

- Let see how the Viola-Jones face detector works using AdaBoost

Features

"Rectangular" filters



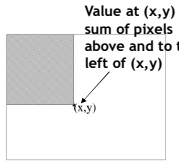
Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

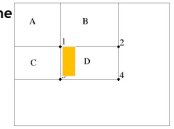
Avoid scaling images → scale features directly for same cost

Viola & Jones, CVPR 2001

Value at (x,y) is sum of pixels above and to the left of (x,y)

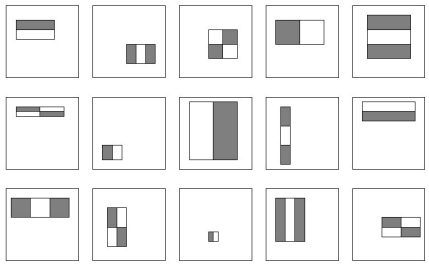


Integral image



$$D = 1+4-(2+3)$$
$$= A+(A+B+C+D)-(A+C+A+B)$$
$$= D$$

Large Library of Filters



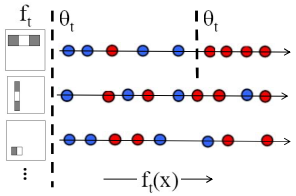
Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

AdaBoost for Feature+Classifier Selection

Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.



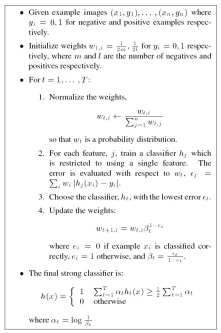
Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

For next round, reweight the examples according to errors, choose another filter/threshold combo.

AdaBoost Algorithm



- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_i}$$
  where $\epsilon_i = 0$ if example x_i is classified correctly, $\epsilon_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples

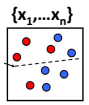
For T rounds

Evaluate *weighted* error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995



93

AdaBoost for Efficient Feature Selection

- Image Features = Weak Classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filters values
  - Select best threshold for each filter (min error)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Reweight examples

Viola & Jones, CVPR 2001

94

Problem

- Even if the filters are fast to compute, each new image has a lot of possible windows to search
- How to make the detection more efficient?

95

Cascading Classifiers for Detection

For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,

- Filter for promising regions with an initial inexpensive classifier
- Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain

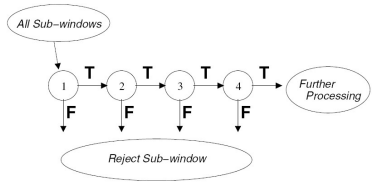
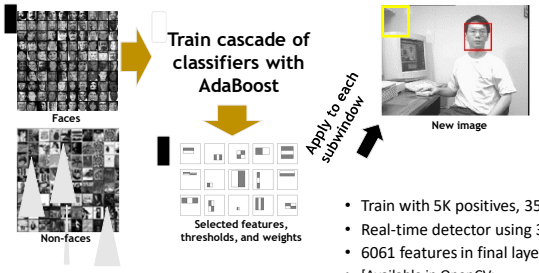


Figure from Viola & Jones CVPR 2001

Fleuret & Geman, IJCV 2001
Rowley et al., PAMI 1998
Viola & Jones, CVPR 2001

96

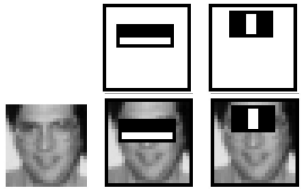
Viola-Jones Face Detector: Summary



- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Available in OpenCV: http://www.intel.com/technology/computing/opencv/]

97

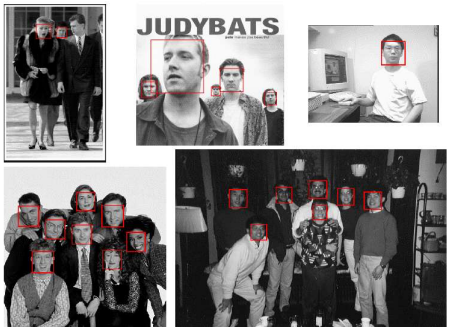
Viola-Jones Face Detector: Results



First two features selected

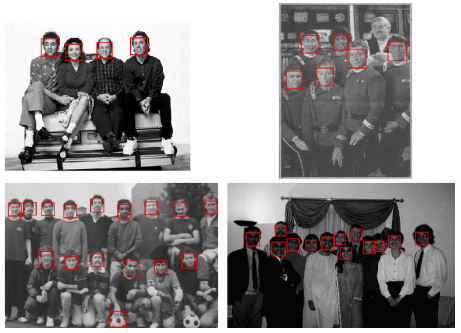
98

Viola-Jones Face Detector: Results



99

Viola-Jones Face Detector: Results



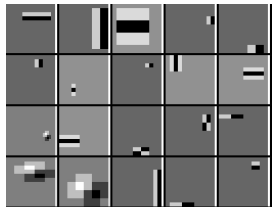
100

Viola-Jones Face Detector: Results


101

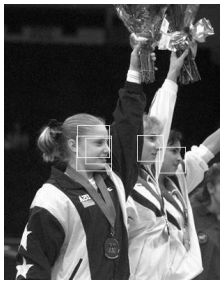
Detecting Profile Faces?

Detecting profile faces requires training separate detector with profile examples.

102

Viola-Jones Face Detector: Results

103

Limitations

- High computational complexity
  - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - If training binary detectors independently, means cost increases linearly with number of classes
- With so many windows, false positive rate better be low

104

RANDOM FORESTS

105

Main types of Ensemble methods

Combine multiple models together

- Committees – Bagging
  - *Regression*: Take an average of the predictions made by each model
  - *Classification*: Make classification by voting over a collection of classifiers
- Boosting – Adaboost
  - Train multiple models in sequence
- Decision trees
  - Different models are responsible for making predictions in different regions of input space

106

Random Forests

Algorithm

- Choose T — number of trees to grow
- Choose $m \ll M$ (M is the number of total features) — number of features used to calculate the best split at each node (typically 20%)
- For each tree
  - Choose a training set by choosing N times (N is the number of training examples) with replacement from the training set
  - For each node, randomly choose m features and calculate the best split
  - Fully grown and not pruned
- Use majority voting among all the trees (or average for regression)

[Breiman'01]

107

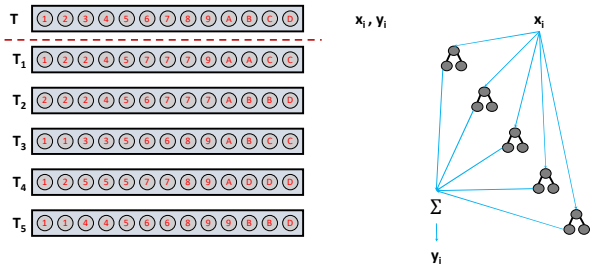
Random Forests

The common element in all of these procedures is that for the kth tree, a random vector Θ_k is generated, independent of the past random vectors $\Theta_1, \ldots, \Theta_{k-1}$ but with the same distribution; and a tree is grown using the training set and Θ_k, resulting in a classifier $h(\mathbf{x}, \Theta_k)$ where $\mathbf{x}$ is an input vector. For instance, in bagging the random vector Θ is generated as the counts in N boxes resulting from N darts thrown at random at the boxes, where N is number of examples in the training set. In random split selection Θ consists of a number of independent random integers between 1 and K. The nature and dimensionality of Θ depends on its use in tree construction.

Definition 1.1. A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\mathbf{x}, \Theta_k),\ k = 1, \ldots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$.

108

Random Forest Classifier



109

Random Forests

- Discussions
  - Bagging + random features
  - Improve accuracy
    - Incorporate more diversity and reduce variances
  - Improve efficiency
    - Searching among subsets of features is much faster than searching among the complete set

110

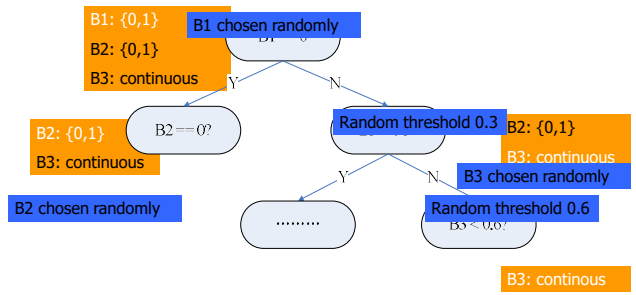
Random Decision Tree

- Single-model learning algorithms
  - Fix structure of the model, minimize some form of errors, or maximize data likelihood (eg., Logistic regression, Naive Bayes, etc.)
  - Use some "free-form" functions to match the data given some "preference criteria" such as information gain, gini index and MDL. (eg., Decision Tree, Rule-based Classifiers, etc.)
- Such methods will make mistakes if
  - Data is insufficient
  - Structure of the model or the preference criteria is inappropriate for the problem
- Learning as Encoding
  - Make no assumption about the true model, neither parametric form nor free form
  - Do not prefer one base model over the other, just average them

[FWM+03]

111

Random Decision Tree



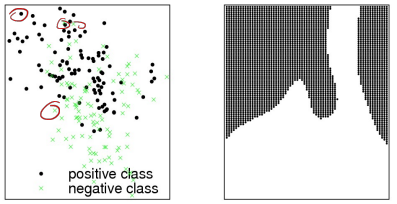
112

Random Decision Tree

- Potential Advantages
  - Training can be very efficient – particularly true for very large datasets
    - No cross-validation based estimation of parameters for some parametric methods
  - Natural multi-class probability
  - Imposes very little about the structures of the model

Optimal Decision Boundary

Figure 3.5: Gaussian mixture training samples and optimal boundary.



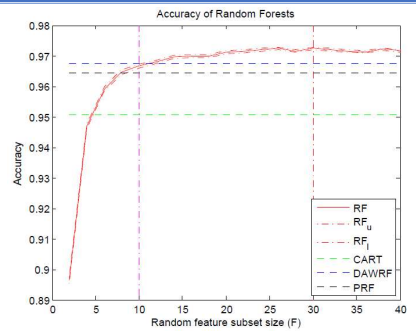
training samples optimal boundary

Optimal Decision Boundary

Another Example

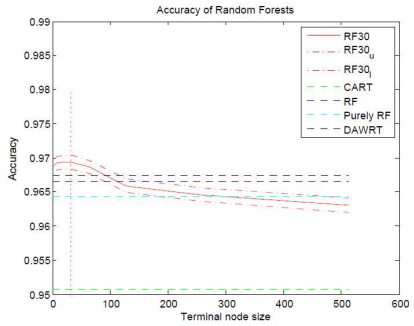
- Number of Trees: 500
- Iteration: 400
- Accuracy estimation: 10 fold cross validation
- Maximal terminal node size for PRF: 20

Another Example



117

Another Example



118

Another Example

	CART	PRF	DAWRF	RF	RF-best
mean	0.9508	0.9643	0.9674	0.9666	0.9724
sd	0.0112	0.0102	0.0096	0.0114	0.0086
F	NA	1	NA	10	30

119

Good

- Accuracy is as good as Adaboost and sometimes better
  - Randomness induces vastly more between-tree differences… (mutations in genetic algorithms)
  - Bootstrapping (even weights) alone does not …
- Relatively robust to outliers and noise
- Fast Computation
- Gives a wealth of important insights (e.g. Estimate of error, variable importance, proximity)
- Simple

120

Class Vote Proportions and Margin

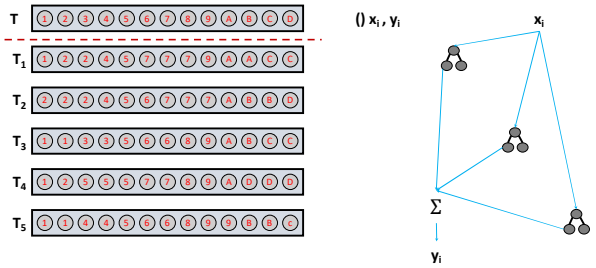
- At the end of an RF run, for every record, the proportion of votes for each class represents the probability of class membership
- The **Margin –** the proportion of votes for the true class minus the maximum proportion of votes for the other classes
- The larger the margin, the higher the confidence of classification

Out-of-bag Error

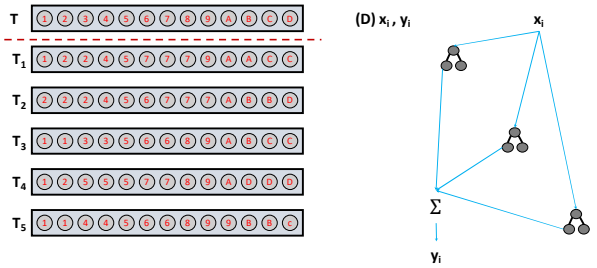
Assume a method for constructing a classifier from any training set. Given a specific training set T, form boostrap training sets T_k, construct classifiers $h(\mathbf{x}, T_k)$ and let these vote to form the bagged predictor. For each $y, \mathbf{x}$ in the training set, aggregate the votes only over those classifiers for which T_k does not containing $y, \mathbf{x}$. Call this the out-of-bag classifier.

Breiman's studies provides empirical evidence showing that **the out-of-bag estimate is as accurate as using a test set of the same size as the training set**.
→ Using the out-of-bag error estimate removes the need for a set aside test set.

Out-of-bag Error



Out-of-bag Error



Variable Importance

- The concept of margin allows new "unbiased" definition of variable importance
- To estimate the importance of the m^{th} variable:
  - Take the OOB (out of bag) cases for the k^{th} tree, assume that we already know the margin for those cases M0
  - Randomly permute all values of the variable m
  - Apply the kthtree to the OOB cases with the permuted values
  - Compute the new margin M
  - Compute the difference M0-M
- The variable importance is defined as the average lowering of the margin across all OOB cases and all trees in the RF
- This procedure is fundamentally different from the intrinsic variable importance scores computed by CART –the latter are always based on the LEARN data and are subject to the overfittingissues

125

Proximity Measure

- RF introduces a novel way to define proximity between two observations:
  - Initialize proximities to zeroes
  - For any given tree, apply the tree to all cases
  - If case i and case j both end up in the same node, increase proximity prox(ij) between i and j by one
  - Accumulate over all trees in RF and normalize by twice the number of trees in RF
- The resulting matrix of size NxN provides intrinsic measure of proximity
  - The measure is invariant to monotone transformations
  - The measure is clearly defined for any type of independent variables, including categorical

126

Using Proximities

- Having the full intrinsic proximity matrix opens new horizons
  - Informative data views using metric scaling
  - Missing value imputation
  - Outlier detection
- Unfortunately, things get out of control when dataset size exceeds 5,000 observations (25,000,000+ cells are needed)
- RF switches to "compressed" form of the proximity matrix to handle large datasets –for any case, only M closest cases are recorded. M is usually less than 100.

127

Scaling Coordinates

- The values 1-prox(ij) can be treated as Euclidean distances in a high dimensional space
- The theory of metric scaling solves the problem of finding the most representative projections of the underlying data "cloud" onto low dimensional space using the data proximities
- The theory is similar in spirit to the principal components analysis and discriminant analysis
- The solution is given in the form of ordered "scaling coordinates"
- Looking at the scatter plots of the top scaling coordinates provides informative views of the data

128

Outlier Detection

- Outliers are defined as cases having small proximities to all other cases belonging to the same target class
- The following algorithm is used:
  - For a case n, compute the sum of the squares of prox(nk) for all k in the same class as n
  - Take the inverse –it will be large if the case is "far away" from the rest
  - Standardize using the median and standard deviation
  - Look at the cases with the largest values –those are potential outliers
- Generally, a value above 10 is reason to suspect the case of being an outlier

Missing Value Imputation

- RF offers two ways of missing value imputation
- The Cheap Way – conventional median imputation for continuous variables and mode imputation for categorical variables
- A Better Way:
  - Suppose case n has x coordinate missing
  - 1. Do the Cheap Way imputation for starters
  - 2. Grow a full size RF
  - 3. We can now re-estimate the missing value by a weighted average over all cases k with non-missing x using weights prox(nk)
  - 4. Repeat steps 2 and 3 several times to ensure convergence

Bad?

- Why maximal tree (no pruning)?
- Optimal random feature subset size(F)?
- Bootstrap sample?

Why Maximal Tree?

- Lin and Jeon (2006), JASA
  - Breiman's classifier can be viewed as adaptively weighted k-potential nearest neighbors methods in regression.
  - Terminal node size should be made to increase with the sample size.
- Biau et al (2008), JMLR
  - Using stopping rule is not necessary in some cases.
- Empirical evidence…

Why are Single Random Trees Predictive?

- A single tree in an RF forest can be predictive because it is a form of nearest neighbor classifier
- We know that nearest neighbor prediction is completely model free
- A good nearest neighbor system should achieve an error rate about twice that of the theoretically best model
  - If the best possible model is correct for 90% of all cases it is wrong for 10%
  - Nearest neighbor should then be correct for 80% and wrong for 20%
- Follow the path down a big tree is …

133

Optimal Random Feature Subset Size(F)?

- Many empirical studies

134

Bootstrap Sample?

- Bootstrap sample is not essential for prediction
- Using bootstrap sample provides useful information
- But we can get same information by cross validation

135

Supervised and Unsupervised RFs

- A Random Forest can be supervised or unsupervised
- Supervised:
  - In a supervised Random Forest, groupings for the training data are input to the algorithm
  - Estimated classification error is computed using out-of-bag data

136

The Road to Clustering

- RF to solve unsupervised learning problems, in particular, clustering problems and missing value imputation in the general sense
- RF generates a synthetic target variable in order to proceed with a regular run:
  - Give class label 1 to the original data
  - Create a copy of the data such that each variable is sampled independently from the values available in the original dataset
  - Give class label 2 to the copy of the data
  - RF on two class → dissimilarity measure for clustering…
  - Note that the second copy has marginal distributions identical to the first copy, whereas the possible dependency among predictors is completely destroyed
  - A necessary drawback is that the resulting dataset is twice as large as the original

137

Thanks for listening!

138