Zurich University of Applied Sciences

Machine Learning
V05: Ensemble Methods

Meta learning
Ensembles in practice
AdaBoost

Based on material from
Todd Holloway, Indiana University
Igor Labutov, Cornell University
Zhuowen Tu, University of California Los Angeles

Zurich University of Applied Sciences and Arts
InIT Institute of Applied Information Technology (stdm)

Educational objectives

- **Know** **when** ensembles **should work** in practice
- Present **argument**s **how** & **why** ensembles **work** in practice
- **Know** and **apply** the **AdaBoost** algorithm to problems of classification and feature selection

1. META LEARNING

Not to be confused with "learning to learn", which also sometimes go by "meta learning": http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/ & appendix

Zurich University of Applied Sciences

Ensembles are meta learning algorithms

Learning to combine learners

same or different $\mathcal{H}$

Ensembles in a nutshell

- Goal: **Combining** multiple **complementary classifiers** to increase performance
- Idea: Build different "experts", and **let them vote**

Pros & cons

✓ Very **effective** in practice
✓ Good **theoretical guarantees**
✓ **Easy** to **implement**, not too much parameter tuning
✗ The result is not so transparent (**black box**)
✗ **Not** a **compact** representation

Formal problem description

- Given T binary classification hypotheses $(h_1,\ldots,h_T)$, find a combined classifier with better performance of the form

$$\hat{h}(x) = sgn\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

For regression, use average instead

individual weight

Why do they work? I
Intuitive thoughts

Intuition
- Utility of **combining diverse, independent opinions** in human decision-making
  E.g., stock portfolio diversity
- **Identifying** single **best model** (i.e., **proper level of** model **complexity) is hard**
  Example of Ockham's 2nd razor (*"simplicity is always good"*) being "blunt" → see [Domingos, 1998] and V03

Example of possible error reduction
- Suppose there are **25 binary** base **classifiers**, each classifier has error rate $\varepsilon = 0.3$
- **Assume independence** among classifiers (i.e., classifiers are complementary)
- **Probability** that the final **ensemble** classifier makes a **wrong** prediction:

Complete independence is often unrealistic!

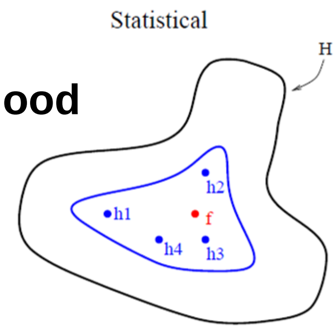
$$p(ensemble\ commits\ error) = \underbrace{\sum_{r=13}^{25} \underbrace{\binom{25}{r} \cdot \varepsilon^r \cdot (1-\varepsilon)^{25-r}}_{\text{prob. that r out of 25 classifiers are wrong (binomial distribution)}}}_{\text{prob. that > 50\% ensemble members are wrong (assuming independence)}} \approx 0.06$$

→ That is: combining **25 *completely* independent classifiers with 70% accuracy** simply by majority vote **yields a 94% accurate classifier**! (→ see appendix for derivation)

Why do they work? II
Three fundamental reasons why they *may* work better

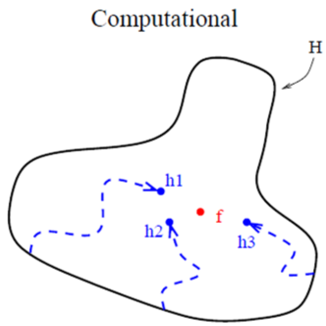
We *cannot* **know** the best ➔ so we average

Statistical
- Given **finite** amount of **data**, many hypothesis typically **appear equally good**
- **Averaging** may be a better approximation to the true f



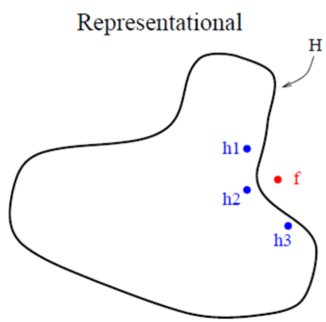
We *may not* **find** the best ➔ so we average

Computational
- **Search** for h **is heuristic** due to interesting $\mathcal{H}$'s being huge/infinite
- Strategy to **avoid local minima**:
  repeat with random restarts, construct an ensemble



We *cannot* **find** the best ➔ so we average

Representational
- The desired **target function may not be realizable** using individual classifiers from $\mathcal{H}$
- It may be **approximated by** ensemble **averaging**



Why do they work? III

In terms of bias and variance (→ see also V06)

Attention: the bias-variance trade-off for **classification** has a very **different** (unintuitive) **form** → see appendix

Assume a regression task

- $E_{MSE} = bias^2 + variance + noise$

distance from true f

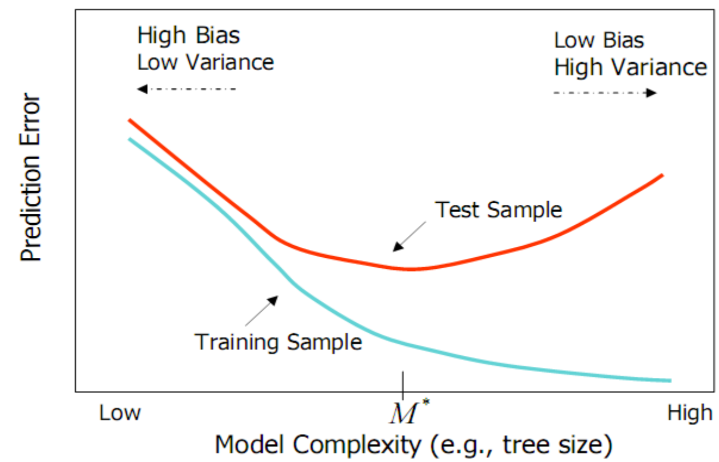
variance in predictions

independent of h (i.e., the Bayes error)

- **Bias** problem:
  E.g., $\mathcal{H}$ used by particular learning method **doesn't include** sufficient h's (near true f)
- **Variance** problem:
  E.g., $\mathcal{H}$ is **too "expressive" for** the training **data** ➔ selected h may **not generalize** well

Example: decision trees

- Small trees have high bias (i.e., too restricted $\mathcal{H}$)
- Large trees have high variance
  (i.e., very unstable decisions in the leaves)



Bias & variance in Ensembles

- **Bias remains equal** w.r.t. the base learners
- **Variance is reduced** with each added member

Example: Bagging
Constructing for Diversity

Bootstrap Aggregating [Breiman, 1996]

- Almost always **improves** results **if base learner** is **unstable** (i.e., high variance)
- Why? $bias\left(\hat{h}(x)\right) = \frac{1}{T}\sum_{t=1}^{T} bias\left(h_t(x)\right)$, $variance\left(\hat{h}(x)\right) \approx \frac{1}{T} variance\left(h_t(x)\right)$

  ➔ usually, the more ensemble members, the better

Algorithm

1. **for** $t := 1..T$
2. $\quad X_t :=$ **sample i.i.d. from** X **with replacement**
3. $\quad h_t :=$ **train any algorithm on** X_t
4. **Return** $\hat{h} := sgn\left(\sum_{t=1}^{T} 1 \cdot h_t(x)\right)$

   #(majority vote; for regression use average instead)

➔ The process is remarkably **simple** (also to implement)

➔ See appendix for Breiman's extension into Random Forests®

Further Reading

- [Breiman, 1996]: *«Bagging Predictors»*, Machine Learning, 24, 123-140, 1996

Zurich University
of Applied Sciences

2. ENSEMBLES IN PRACTICE

The Netflix Prize of 2006–2009
Ca. 3 years of challenging the global data science community

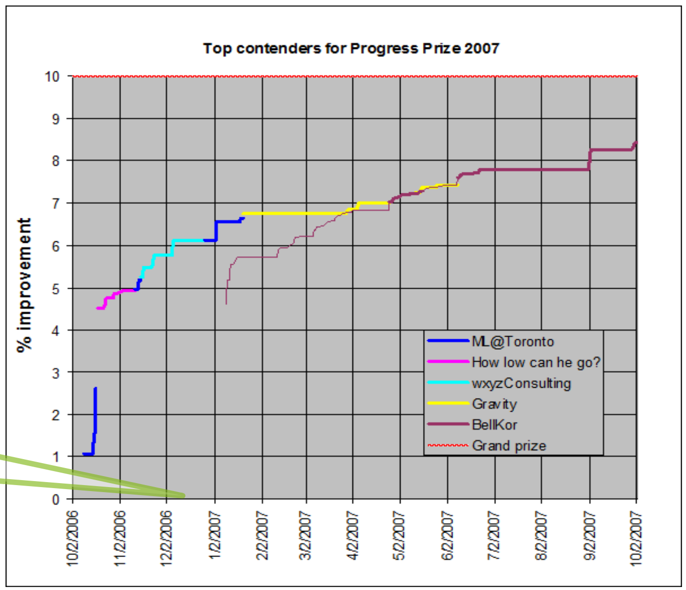
Supervised learning task

- Goal: Construct a classifier that, given a user and an unrated movie, correctly classifies that movie as either 1, 2, 3, 4, or 5 stars (i.e., **predict rating by user**)
- Input: Training data is set of users and ratings (1,2,3,4,5 stars) for movies
- Incentive: $1'000'000 for a 10% improvement over Netflix's current movie recommender (E_{MSE}=0.9514)

➔ See http://www.netflixprize.com

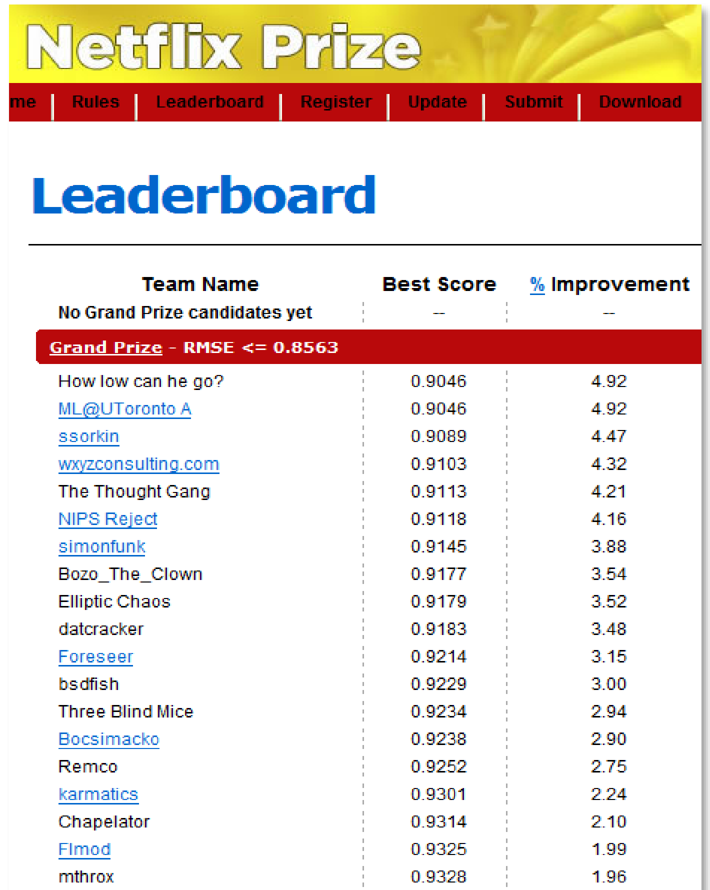
Evolving results I
Low hanging fruits and slowed down progress

- **After 3 weeks**, at least **40 teams** had **improved** the Netflix classifier
- Top teams showed about 6% improvement
- However, **improvement slowed**:



Mind the 1 year gap!

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



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
Flmod	0.9325	1.99
mthrox	0.9328	1.96

Zurich University
of Applied Sciences

Evolving results II
A leader board full of ensembles

Intermediate results

- Top team has posted a 8.5% improvement
- **Ensemble** methods are the **best** performers…

- …as we will see on the **next slides**

--	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
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Details: Rookies

Quote

- *“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
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Details: Arek Paterek

Quote

- *“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
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Details: University of Toronto

Quote

- *“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
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Details: Gravity

Quote

- **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
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Details: When Gravity and Dinosaurs Unite

Quote

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

--	**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
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Details: BellKor / KorBell

Quote

- *"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
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Evolving results III
Final results

The winner was an **ensemble of ensembles** (including BellKor)

- Gradient boosted decision trees [http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf]

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 **leaders.**

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - 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_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
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				

E.g. on Kaggle, **pattern recognition** benchmarks like ImageNet, etc.

→ **Hint: Ensembles still win** competitions, but **Deep Learning** has **better** performance for **unstructured** data (→ see later and https://www.import.io/post/how-to-win-a-kaggle-competition/)

→ The **winner model was never used** in Netflix' practice due to its complexity

XGBoost: A scalable tree boosting system
[Chen & Guestrin, 2016] → using gradient boosting, see appendix

A **skillfully engineered**, **highly optimized** implementation
- Used by $17/29$ **winning** teams on **Kaggle** 2015
- Open source (Python, R, Spark, …): https://github.com/dmlc/xgboost
- Scalable: $10\times$ faster than usual implementations, scales to $\sim 10^9$ training points
  - Massive use of parallelization/distribution (e.g. on Hadoop/Spark, but also on desktop)

Both types of novelties **purely increase** the **computational performance**, not learning in general

Algorithmic novelties
- Distributed **approximate best split** finding („weighted quantile sketch" using quantile statistics)
- **Exploit sparsity** (induced by missing values/one-hot encoding → via default directions for branching)

Parallelization Cache-aware access (for gradient statistics)
- Efficient **out-of-core computation** (i.e., computation on data not fitting into main memory)

General tricks for tree boosting
- Use aggressive sub-sampling (e.g., selecting only 50% of the data)
- Using column sub-sampling prevents over-fitting even more so than row sub-sampling

3. ADABOOST

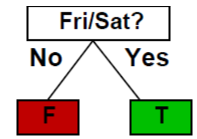
Boosting

Zurich University of Applied Sciences

General idea

- **Boost** the performance of weak learners (error slightly >chance) **iteratively**
- **Make** currently **misclassified examples** more **important**, then combine hypotheses
  ➔ Each **stage** (additively) corrects shortcomings of previous stage by **reweighting**, then **majority vote**
- Origins in computer science: [Kearns & Valiant, 1988] (as opposed to Bagging: statistics)

Adaptive Boosting algorithm [Freund & Schapire, 1997]



- Weak learner: decision stump (=decision tree of height 1; but generalizable to others)
  ➔ Important: weak learners have skill but remain weak (to not lose the ensemble effect)

initialize weights: $w_i \coloneqq \frac{1}{N}$ #each sample gets same weight

for $t := 1..T$

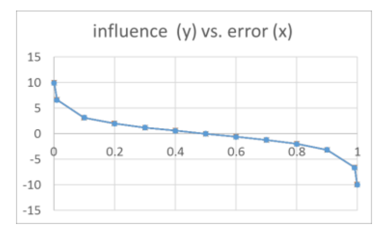
  $h_t \coloneqq$ train decision stump on the x_i, weighted by the w_i

  $\varepsilon_t \coloneqq \frac{\sum_{i=1}^{N} w_i \cdot I(y_i \neq h_t(x_i))}{\sum_{i=1}^{N} w_i}$ #compute **error**; $I()$ is the identity function

  $\alpha_t \coloneqq \log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ #compute **influence** of weak learner

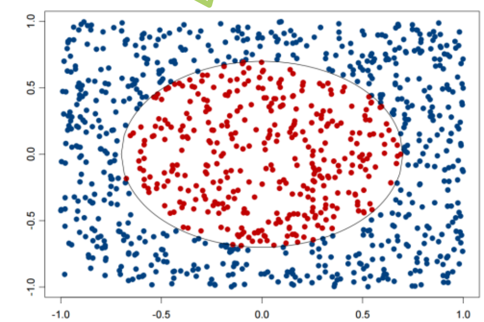
  $w_i \coloneqq w_i \cdot e^{\alpha_t \cdot I(y_i \neq h_t(x_i))}$ #increase weight by exp(influence) in case of error

return $\hat{h} \coloneqq sgn(\sum_{t=1}^{T} \alpha_t \cdot h_t(x))$ #majority vote

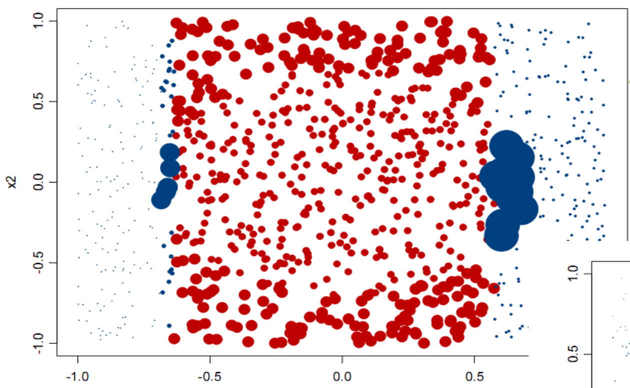


Example run

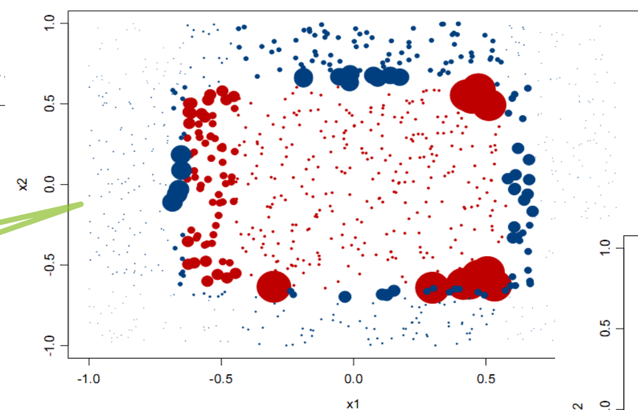
Goal



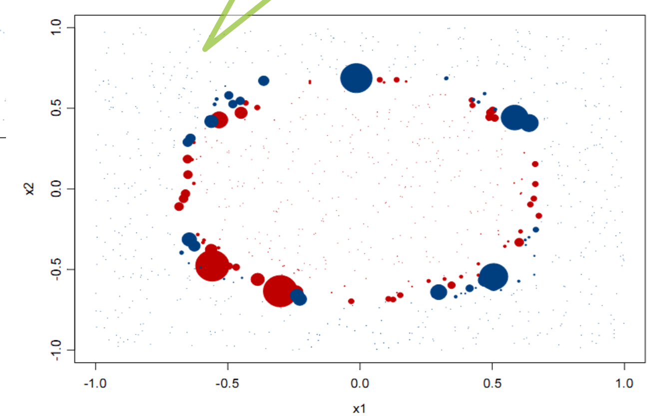
AdaBoost's classifications (colors) and weights (size) after **1 iteration ➔** still looks like a single decision tree with rectangular decision boundary



3 iterations



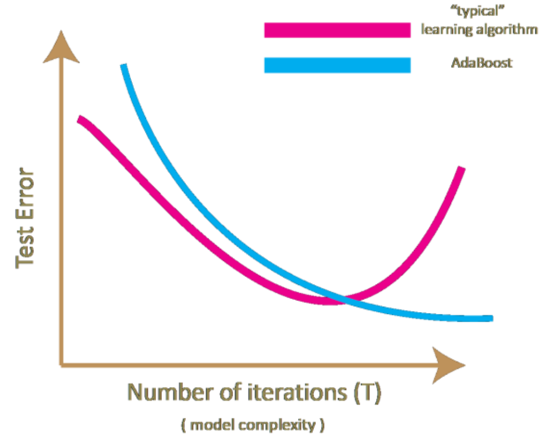
20 iterations



Source

- [Elder, 2007]: *«From Trees to Forests and Rule Sets – A Unified Overview of Ensemble Methods»*

AdaBoost in practice



Pros & cons
- ✓ Very **little code**
- ✓ **Reduces** bias **&** variance
- ✓ Still learns when others overfit ➔ **margin optimization**
- ✘ Sensitive to noise and outliers

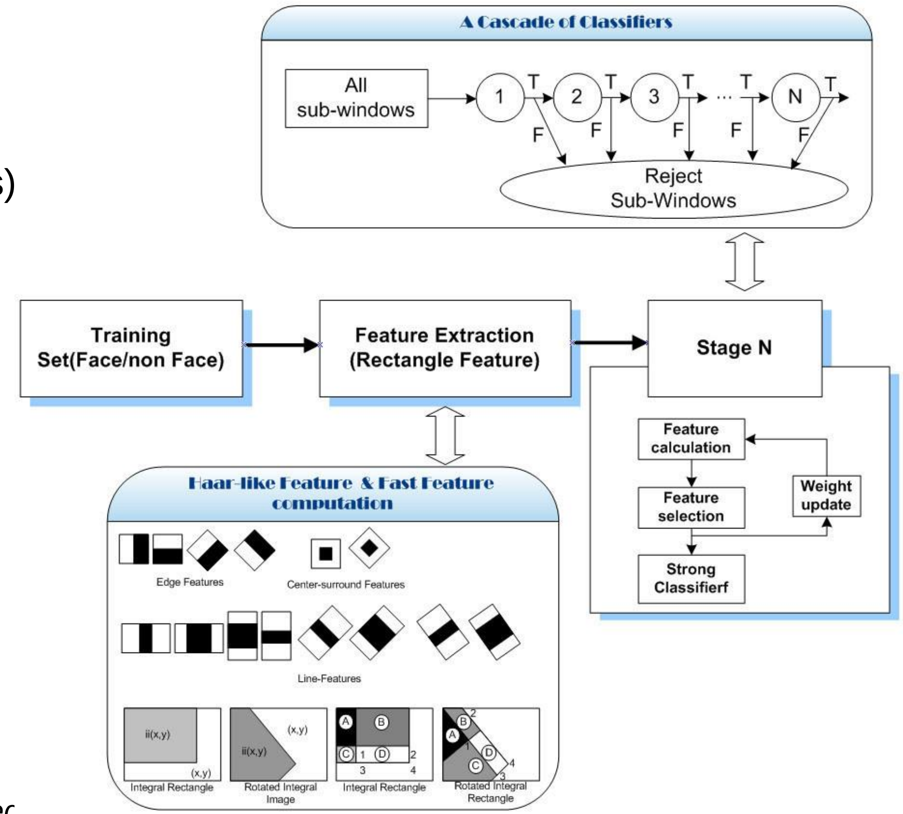
Implementation choices
- A **good start** for implementation is the variant "AdaBoost.M1" from [Frank & Witten, 2005], combined with ideas from "Real AdaBoost.MH" of [Schapire & Singer, 1999]
- For **cost-sensitive** binary classification, use "AdaC2" from [Sun et al., 2007]

Further reading
- [Freund & Schapire, 1997]: *«A decision-theoretic generalization of on-line learning and an application to boosting»*
- [Sun et al., 2007]: *«Cost-Sensitive Boosting for Classification of Imbalanced Data»*
- [Frank & Witten, 2005]: *«Data Mining - Practical Machine Learning Tools and Techniques»*, 2^{nd} Ed.
- [Schapire & Singer, 1999]: *«Improved Boosting Algorithms Using Confidence-rated Predictions»*

Example application: Real-time face detection
AdaBoost as a feature selector

Viola & Jones face detector

- The **first** method for object *detection* in images with **human-like performance**
  (today outperformed by deep learning approaches)
- AdaBoost applied to **>160’000 features**
- First k selected features of decision stumps are deemed meaningful
- Trained on very **unbalanced data**
  (faces ↔ non-faces)



Ju et al., “Outline of face detection using AdaBoost algorithm”, Journal of NeuroEngineering and Rehabilitation, 6:33, 2009

Further reading → see appendix

- [Viola & Jones, 2001]: *«Rapid object detection using a boosted cascade of simple features»*
- [Viola & Jones, 2003]: *«Robust Real-Time Face Detection»*

Review

- Ensembles can be seen as **meta learners** (operating on learners, not data): **learning** to make the **best of many base learners**

- Building ensembles can be as **easy** as **Bagging**: **train** any $\boldsymbol{T}$ **classifiers** on different **bootstrap** samples, **then** take a (classification:) **majority vote** or (regression:) average

- Ensembles work because they use **averaging in a clever way**: **reduce variance**, reach $\widehat{\boldsymbol{h}} \notin \boldsymbol{\mathcal{H}}$, **overcome small data** sets

- Ensembles have been **very successful** in the past; it is good advice to **always build an ensemble of complementary models** as the final classifier

- **AdaBoost** is very **immune to overfitting** and can be used for feature selection (→ see appendix)

P04.3: Building ensembles

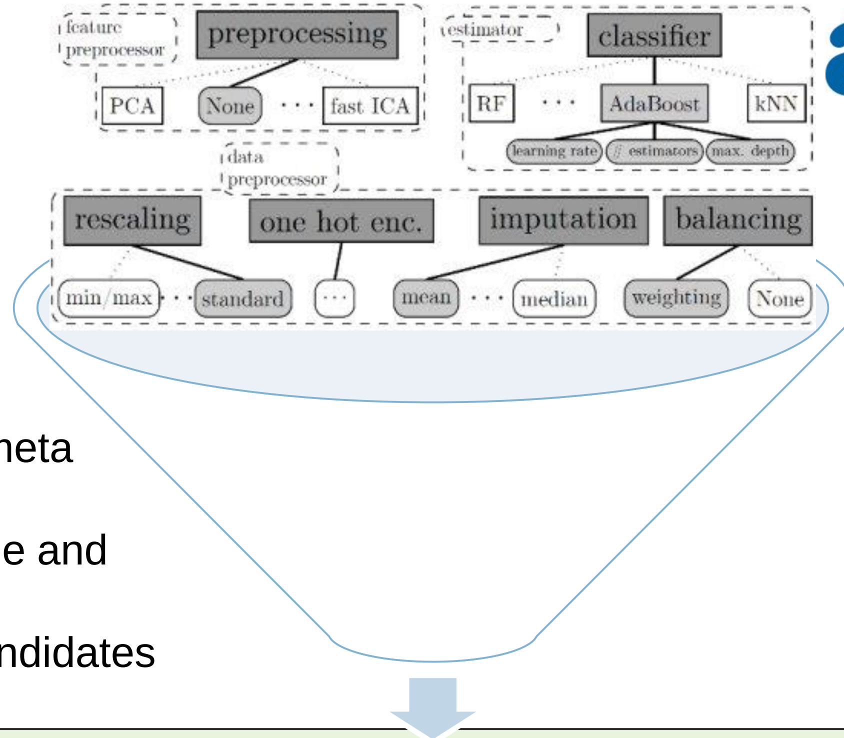
Work through exercise P04.3

- Goal is to build a final classifier for SPAM classification
- Which one of different algorithms performs best?
- Is a combination beneficial on this task?

APPENDIX

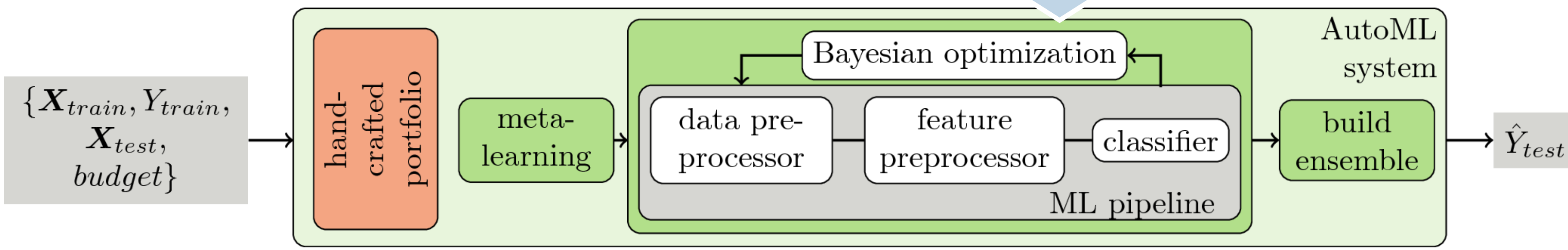
- More on ensembles and error analysis
- Random Forest® and gradient boosting
- The Viola-Jones face detector

Learning to learn



The Auto-sklearn pipeline approach

- 2 times winner of AutoML challenge (2015/16 & 2017/18)
- Utilizes good initialization by starting from a well performing model on a similar dataset seen as seen during meta learning
- Uses Bayesian optimization of pipeline and hyperparameters to tweak this model
- Finally builds an ensemble of best candidates



Source: https://www.automl.org/wp-content/uploads/2018/07/autosklearn.png, https://www.kdnuggets.com/wp-content/uploads/auto-sklearn-overview.jpg
See also: Tuggener et al., "Automated Machine Learning in Practice: State of the Art and Recent Results", Proc. 6th Swiss Conference on Data Science (SDS), 2019

Derivation: Ensemble error of t independent binary classifiers

t being an uneven integer

- Suppose there are t independent base classifiers, each classifier has error rate Ɛ
- They form an ensemble via majority voting:
  $\left\lceil \frac{t}{2} \right\rceil$ base classifiers have to be correct for the ensemble to be correct
- Let E_r be the event that r out of t base classifiers vote incorrectly:
  Its probability follows a binomial distribution $p(E_r) = \binom{t}{r} \cdot \varepsilon^r \cdot (1-\varepsilon)^{t-r}$

The binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ gives the number of subsets of size k of a superset of size n

- Let E be the event that the whole ensemble is wrong (i.e., at least $\left\lceil \frac{t}{2} \right\rceil$ incorrect votes):
  Its probability is given by $p(E) = \sum_{r=\left\lceil \frac{t}{2} \right\rceil}^{t} p(E_r)$

Reasoning

- E occurs if $\left\lceil \frac{t}{2} \right\rceil$ base classifiers are wrong, or if $\left\lceil \frac{t}{2} \right\rceil + 1$ base classifiers are wrong, or if … t base classifiers are wrong
- Assuming independence among these events, their probabilities are added

Discussion: Bias-variance trade-off for 0/1 loss
Going from regression to classification

Definitions: Bias and variance of a learner w.r.t a single instance x [Domingos, 2000]
- $bias :=$ deviation of best possible prediction from main prediction
- $variance :=$ average deviation (over all training sets) from actual to main prediction

Regression
- The **bias-variance trade-off** has **originally** been defined **for regression** problems
- Typical loss function is the **mean squared error** (MSE)
  - $L_{MSE} = bias^2 + variance + noise$ (→ see V03)

Classification
- Usually binary classification is studied in depth first→ result may then be extended to multi-class
- Binary classification uses **classification error as** its **typical loss** function (a.k.a. 0/1 loss)
  - The main prediction is the most frequent prediction; we subsequently ignore the additive noise term
  - $L_{0/1} = bias + variance$ in case of $bias = 0$ (i.e., classifier is correct $> 50\%$ of the time)
  - $L_{0/1} = bias - variance$ in case of $bias = 1$ (i.e., classifier's accuracy is $\leq 50\%$)

Discussion: Bias-variance trade-off for 0/1 loss
Counter-intuitive implications

Consequences for classification

- Bias and variance have a **complicated, multiplicative interaction** [Friedman, 1997]
  (→ not directly visible in the form shown on the last slide due to the 2 cases)
- Good classifiers become better with less variance;
  bad classifiers become better with more variance!
- This explains why **highly unstable classifiers** (e.g., decision trees; kNN in high dimensions; naïve Bayes) **work well in practice**
- Casting classification as a regression problem by **estimating class probabilities** instead often **doesn't pay off**:
  - Good regression results don't imply good classification performance
  - Reason: Different behavior of errors

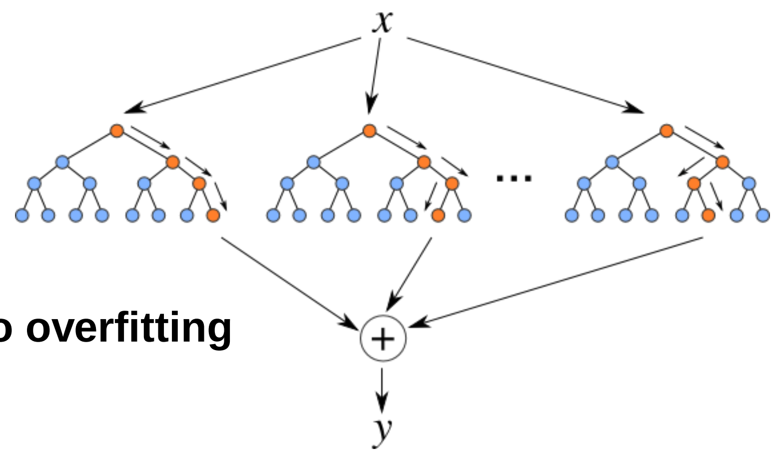
Further reading

- [Domingos, 2000]: *«A Unified Bias-Variance Decomposition for Zero-One and Squared Loss»*
- [Friedman,1997]: *«On Bias, Variance, 0/1-Loss, and the Curse of Dimensionality»*

Random Forest®
A brief description

Build a majority-voting **ensemble of decision trees**; for each tree,

- Choose a **stratified training set** of n out of N instances by sampling **with replacement**
- At every level,
  - choose a **random feature set** (**with replacement**) of m out the p attributes
  - choose **the best** split among those attributes
- **No pruning** of the branches takes place



Advantages

- Fast training, parallelizable application
- High independence of base classifiers ➔ nearly **no overfitting**
- Few hyper parameters
- Applicable to large quantities of N, p and #classes

➔ **Very good out-of-the-box method**

Further reading

- [Breiman 2001]: *«Random Forests»*. Machine Learning 45 (1), 5-32

From AdaBoost to gradient boosting

Recall: In **AdaBoost**, ”**shortcomings**" are identified by high-**weight** data points

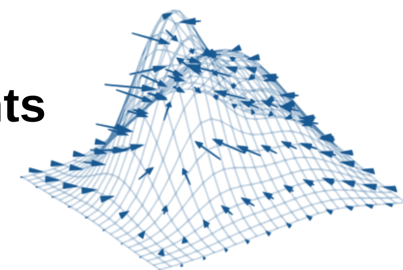
A brief history of modern boosting (selective, shortened)

1. Invention: **AdaBoost**, the **first** successful boosting **algorithm**
   [Freund et al., 1996], [Freund & Schapire, 1997]
2. Translation: **Formulation as gradient descent** with special loss function (→ cp. V02)
   [Breiman et al., 1998], [Breiman, 1999]
3. Generalization: Gradient boosting in order **to handle** a **variety of loss functions**
   [Friedman et al., 2000], [Friedman, 2001]

→ For a great example of cross-disciplinary fertilization, see
Breiman, *“Arcing classifiers (with discussion and a rejoinder by the author)”*, 1998

In **gradient boosting**, “**shortcomings**" are identified by **gradients**

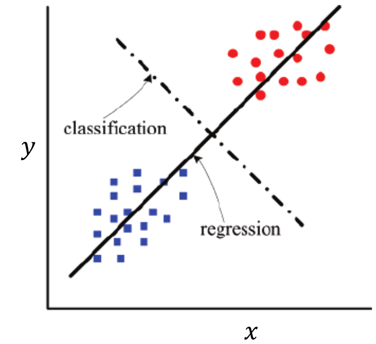
- Gradients of what? Why? → see next slides



Intuition for gradient boosting

Setup

- For ease of discussion we change the setting from (binary) classification to regression (i.e., real-valued labels)
- Results are again applicable to classification (but not intuitively as straight-forward)



Let's play a game

- You are given data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ and **the task to fit model** $\hat{h}(x)$
  ➔ minimize squared loss $\ell\big(y, h(x)\big) = \frac{1}{2}(y - h(x))^2$
- Suppose a friend helps by giving you an **initial model** $F(x)$ (a regression tree)
  ➔ You check his model and find the model is good but not perfect (e.g. $F(x_1) = 0.8$ while $y_1 = 0.9$)
- Rule: $F(x)$ must **not be changed** in any way, but another model might be added
  ➔ i.e. $\hat{h}(x) = F(x) + h(x)$
- How to train $h(x)$?

We want this to be true

Equivalently, we can fit the new regression tree h to:

$$F_1(x_1) + h(x_1) = y_1 \quad ➔ \quad h(x_1) = y_1 - F(x_1)$$

$$\vdots$$

$$F(x_N) + h(x_N) = y_N \quad ➔ \quad h(x_N) = y_N - F(x_N)$$

Intuition for gradient boosting (contd.)

Simple ensemble solution

- The $y_i - F(x_i)$ are called residuals
  ➔ These are the parts that the initial model F cannot do well
  ➔ The role of h is to compensate the shortcomings of F
- If the new model $F + h$ is still not satisfactory, we can add another regression tree...

How is this related to gradient descent?

w.r.t J's parameters θ

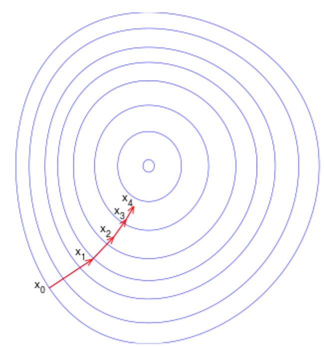
- Gradient Descent: **Minimize** function a J **by moving** into **opposite direction** of the **gradient**

$$\theta_i^{new} = \theta_i^{old} - \alpha \frac{\partial J}{\partial \theta_i^{old}}$$

i.e., $J = L$

- Want to **minimize loss** function: $\mathrm{L} = \sum_{i=1}^{N} \ell\big(y_i, F(x_i)\big) = \sum_{i=1}^{N} \frac{1}{2}\big(y_i - F(x_i)\big)^2$
  ➔ $F(x_i)$ are the parameters of L, so we can take derivatives:

$$\frac{\partial L}{\partial F(x_i)} = \frac{\partial \sum_{i=1}^{N} \ell\big(y_i, F(x_i)\big)}{\partial F(x_i)} = \frac{\partial \ell\big(y_i, F(x_i)\big)}{\partial F(x_i)} = F(x_i) - y_i$$



- That is: We can **interpret residuals as negative gradients**
  ➔ $y_i - F(x_i) = -\frac{\partial L}{\partial F(x_i)}$

Gradient boosting of regression trees

Algorithm

- Gradient boosting for regression

```
Start with an initial model, e.g. F = (Σ_{i=1}^N y_i)/N  always predict mean value)
repeat until convergence
    -g(x_i) = -∂ℓ(y_i,F(x_i))/∂F(x_i)
    fit regression tree h to -g(x_i)
    F := F + αh  #α is a tunable learning rate, e.g. = 1
```

True for ℓ = squared loss

- Residual ⇔ negative gradient
- Fit h_i to residual ⇔ fit h_i to negative gradient
- Update h_i based on residual ⇔ update h_i based on negative gradient

➔ So we **are actually** updating our model **using gradient descent**!

Advantage of gradient descent formulation

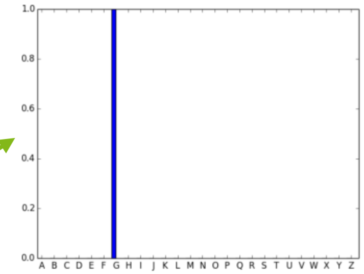
- Allows **consider**ing **other loss** functions (e.g. more outlier-robust, domain-specific, …)
  ➔ Derive the corresponding algorithms in the same way

Extension to (multiclass) classification

Model

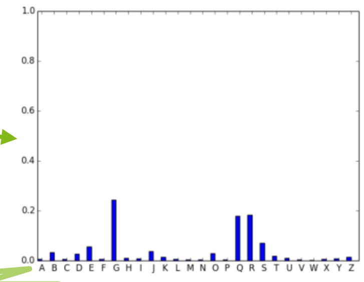
- Each class c has its own model $F_c(x)$ (binary classification tree, emitting $0/1$)
- Use outputs to compute class probabilities: $P_c(x) = \frac{e^{F_c(x)}}{\sum_i e^{F_i(x)}}$ (softmax)

  ➔ Final classification = class with highest probability



Loss function per data point

- Turn the label y_i into a (true) probability distribution $Y_c(x_i)$
- Calculate predicted probability distribution $P_c(x_i)$
  → Based on current models $F_c(x_i)$
- Calculate difference between true and predicted probability distribution
  → Use e.g. KL-divergence as loss



Example: Letter (A-Z) classification

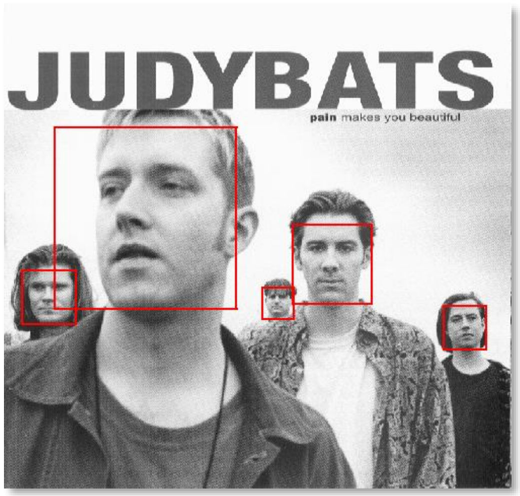
Overall objective

- Do gradient descent to make true and predicted distribution as close as possible $\forall x_i$
- We achieve this goal by adjusting our models F_c

AdaBoost for face detection
A detailed example of a boosted decision stumps application

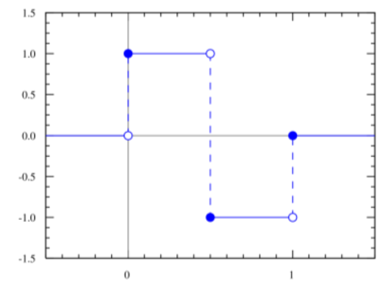
Challenges
- **Slide** a **window** across image and evaluate a face model **at every location & scale**
  ➔ Sliding window detector must evaluate tens of thousands of location/scale combinations
- Faces are **rare**: 0–10 per image
  - For **computational efficiency**, we should try spending as little time as possible on non-face windows
    ➔ A megapixel image has $\sim 10^6$ pixels and a comparable number of candidate face locations
  - To avoid having a false positive in every image, the **false positive rate has** to be **less** than $\mathbf{10^{-6}}$

The Viola-Jones face detector [Viola & Jones, 2001]
- A seminal approach to **real-time object detection**
  ➔Training is slow, but detection is very fast
- Key ideas
  - Integral images for fast feature evaluation
  - **Boosting for** feature selection amongst $\sim 10^5$ candidates
  - Attentional cascade for fast & accurate rejection of non-face windows



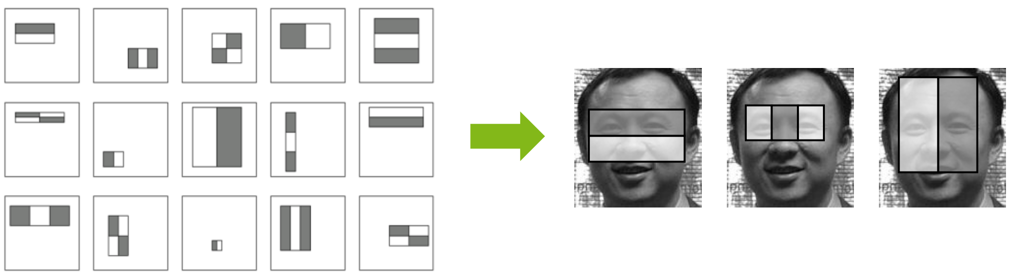
Rectangular facial features

…and their efficient calculation via the integral image



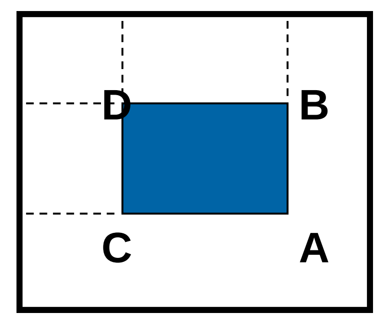
Pixel-based features for face detection

- Reminiscent of Haar wavelets
- Simple **sum of pixel intensities** within rectangular regions resemble typical shading patterns of faces



Integral images (ii)

- Let **each pixel** be the **sum** of all pixels **left** and **above**



Computing sums of pixels within a rectangle using ii

- $sum = ii_A - ii_B - ii_C + ii_D$
- Needs **only 3 additions** for any size of rectangle (**constant time**)

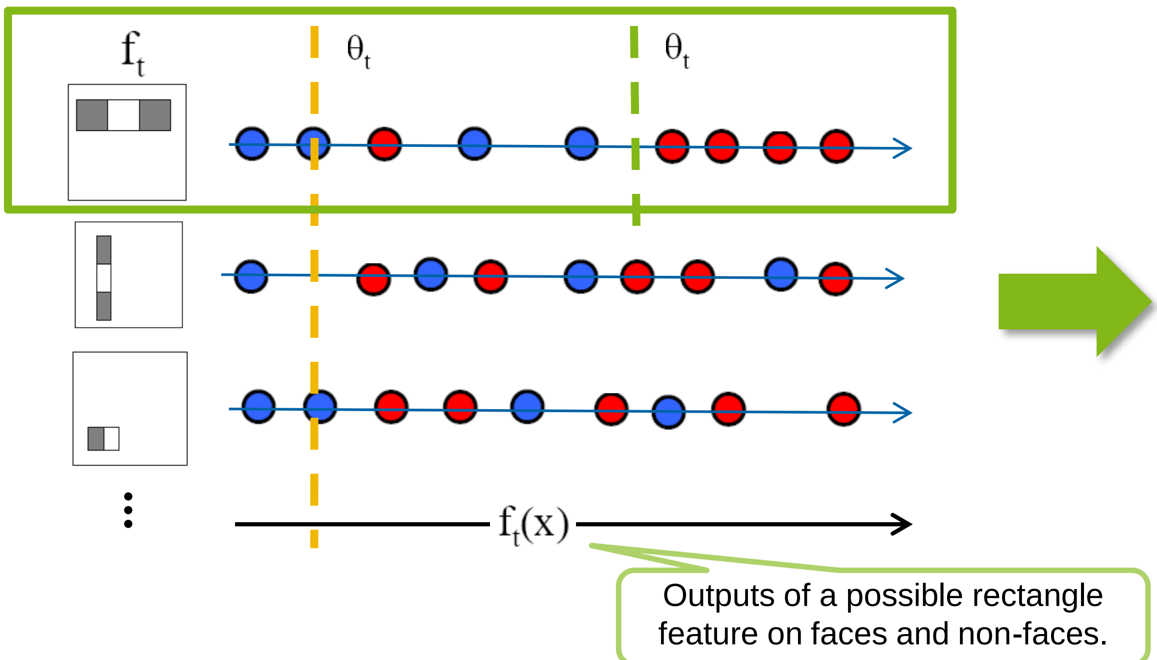
Feature selection via AdaBoost
Slide adapted from Grauman & Leibe's AAAI'08 tutorial

Size of feature space

- Ca. **160'000 distinct** rectangular features **per** detection **window** (via scaling/translation)
  ➔ Which ones are **good?** What is a good **subset?**

Finding a good succession of features

- **Start**: Select the **single** rectangle **feature** & **threshold** that best separates faces/non-faces



Resulting weak classifier:

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

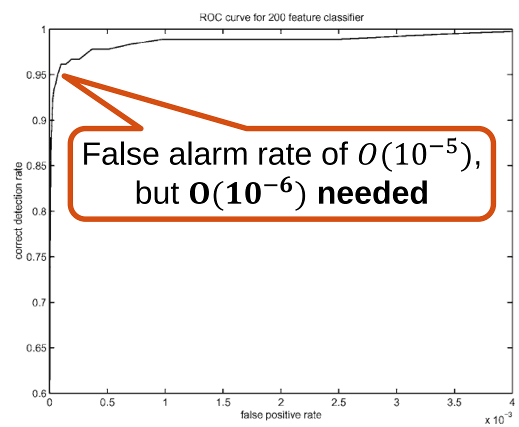
➔ **Continue** using **AdaBoost**

Training the boosting classifier
Incorporating feature selection

Training set contains face and non-face examples
- **5000 faces** (frontal, many variations among illumination/pose, rescaled to 24×24)
- **300 million non-faces** (extracted from $9'500$ non-face images)
- Faces are normalized (scale, translation)
- Initially, all have equal weights

For **each round** of boosting:
- Evaluate **each** rectangle **filter** on **each example**, select best threshold
- **Select best filter/threshold** combination
- Reweight examples

➔ Computational complexity: $O(rounds \times examples \times features)$

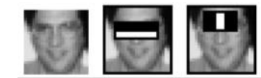


Result
- A 200-feature classifier can yield **95% detection rate** and a false positive rate of 1 in 14084

➔ **Not** yet **good** enough for practice!

Removing false alarms while retaining high detection rate

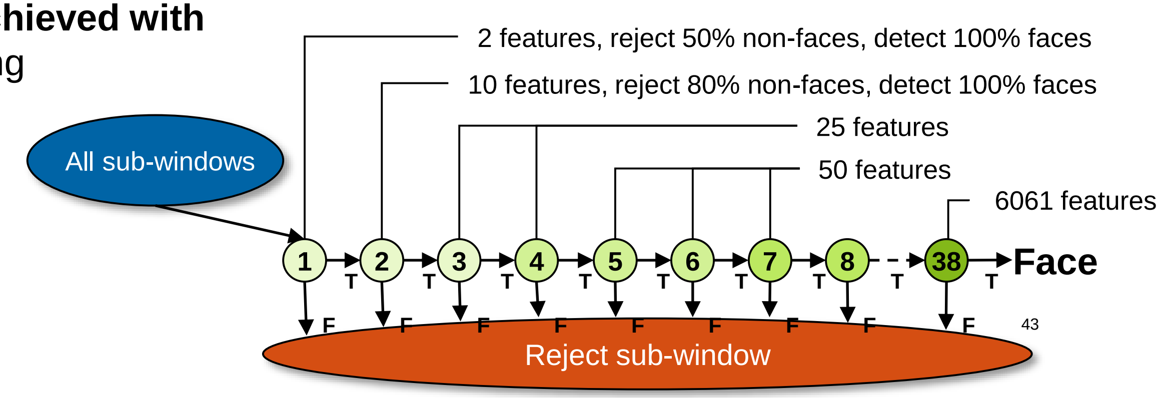
Attentional Cascade



- **Start** with a **simple** classifier (2 features)
  ➔ **Reject**ing **many** of the **negative** sub-windows **while detecting** almost **all** positive sub-windows
- **Positive** response from the first classifier **triggers** the evaluation the **next** classifier, etc.
  ➔ Subsequent classifiers get more complex, hence longer runtime but lower false alarm rate
- A negative outcome at any point leads to the immediate rejection of the sub-window
- Training:
  - Keep **add**ing **features** to **current stage until** its target rates (**TP, FP**) have been **met**
  - If overall **FP** is **not low** enough, then **add** another **stage**
  - Use false positives from current stage as the negative training examples for the next stage

Detection rate (TP) vs. false alarm rate (FP) for chained classifiers

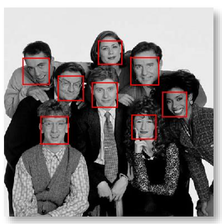
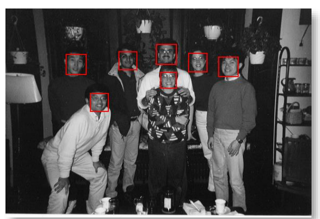
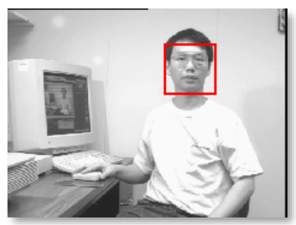
- Found by multiplying the respective rates of the individual stages
  ➔ **TP of $\mathbf{0.9}$ and FP of $\sim\mathbf{10^{-6}}$** can be **achieved with** a **$\mathbf{10}$-stage** cascade: each stage having
  - TP of 0.99 $(0.99^{10} \approx 0.9)$
  - FP of ~ 0.3 $(0.3^{10} \approx 6 \times 10^{-6})$



Final result of Viola-Jones face detection

After some more engineering…

- **Variance normalization** of pixel intensities to cope with different lighting
- Merging multiple detections
- Multi-scale detection by **scaling** the **detector** (factor of 1.25 yields good resolution)

Lasting effect

- Got **applied to more** visual detection problems
  ➔ facial feature localization, profile faces, male/female image classification, audio fingerprinting, …
- **Solved** the problem of face detection **in real time** (e.g. for digicams)
  ➔ available in OpenCV (http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html)
- One of the first mind-blowing computer vision applications before deep learning trend