12. Bagging and Boosting

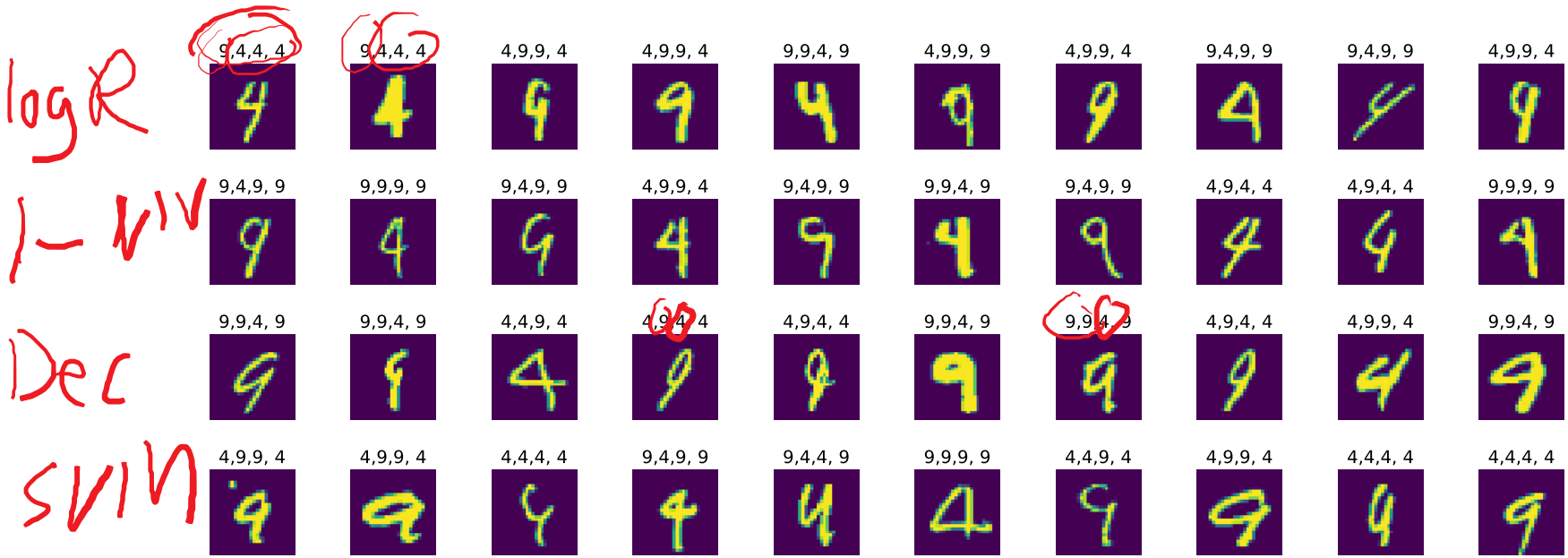
- Ensemble learning
- Bagging
- Boosting

Ensemble learning

Weak learners

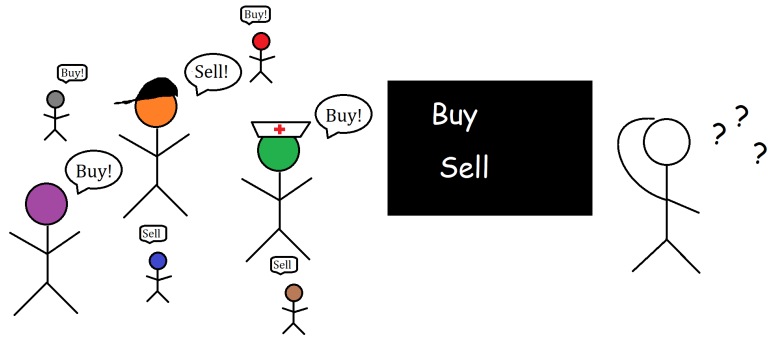
- Last time, we talked about using weak learners to control overfitting
- Weakening a learner:
  - polynomials of fixed degree (extreme = linear)
  - decision trees with fixed depth (extreme = stumps)

Mixture of experts



Logistic regression, 1-NN, Decision tree, SVM. (Demo time!)

Mixture of experts



- Type of experts? (Often, decision "stumps", trees of depth $= 2$)
- How to merge decision? Voting? Sum? (Bagging)
- Some experts better than others. How to identify / reweight? (Random forest regression)
- Can we actively pick the best expert given current weaknesses? (Boosting)

Ensemble method

... is a method which takes methods as inputs.

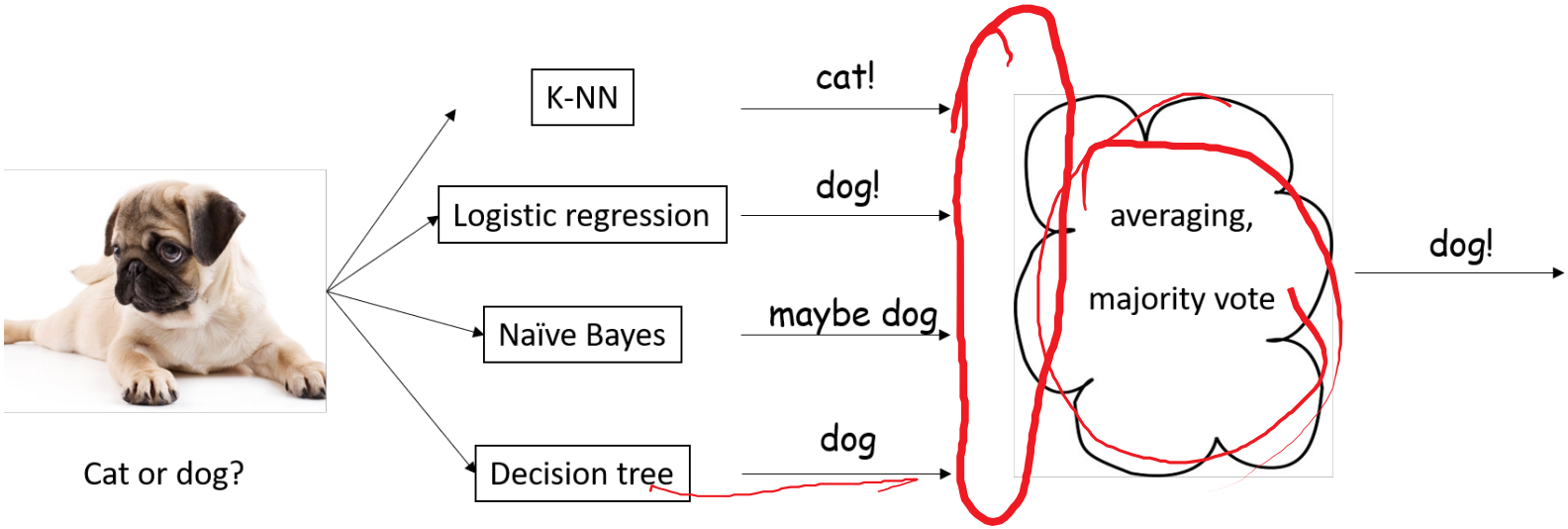
Recipe:

- Pick a (class of) methods
- Pick a way of sampling data
- Each data sample + method = predictor
- Aggregate predictor = $f(\text{pred}_1, \text{pred}_2, ...)$

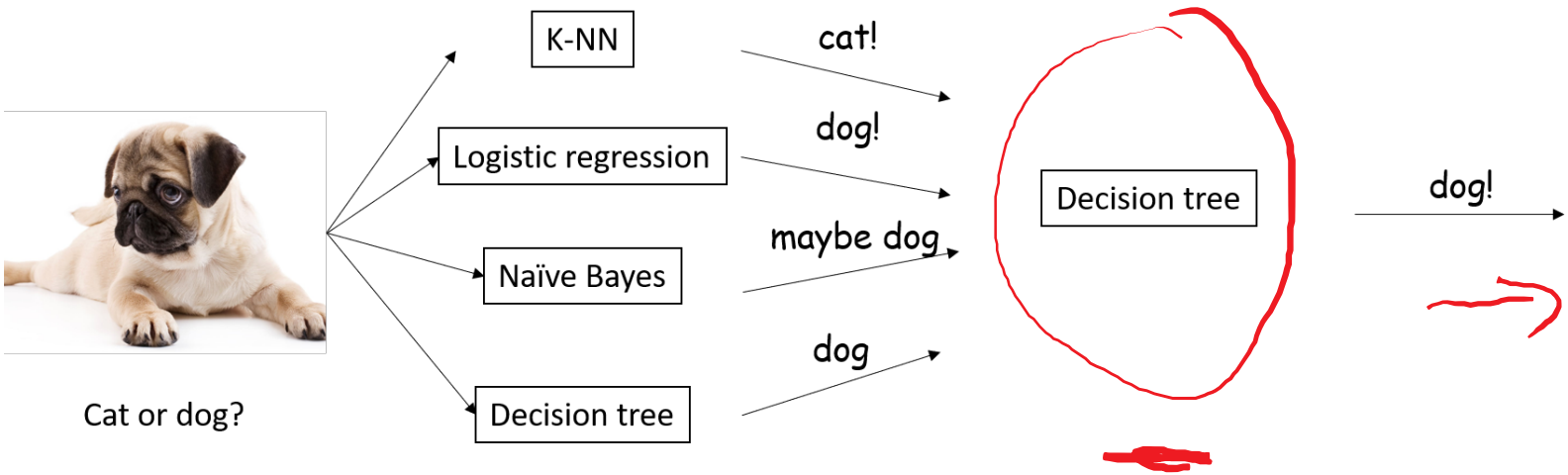
How to aggregate?

- Bootstrap / bagging: average
- Boosting: greedy merge

Simple aggregation



Stacking / cascading



When can mixture of experts do better?

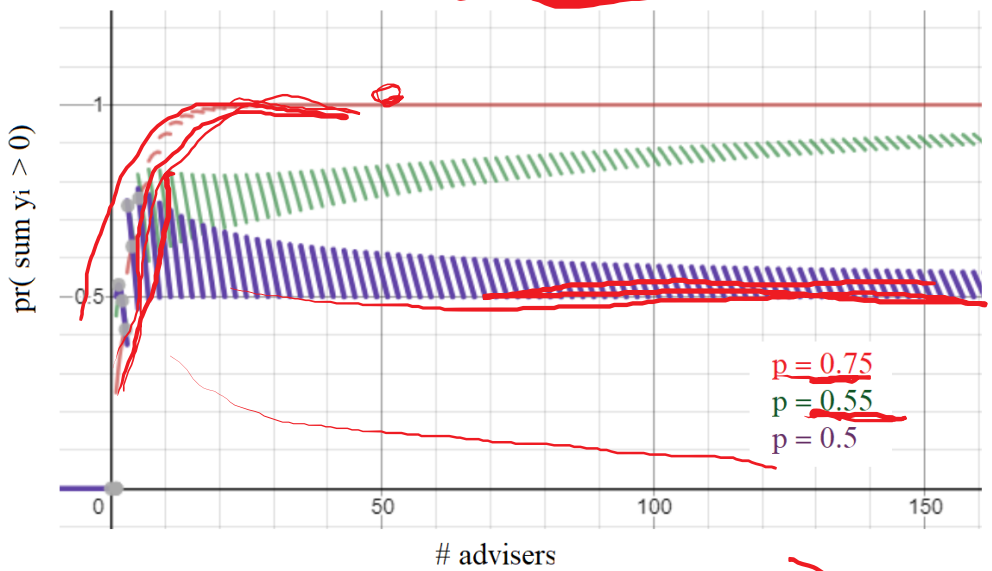
- If all the experts are the same, advice shouldn't change much

- If all the experts are diverse, advice should improve

When can mixture of experts do better?

- Advisers $y^{(1)}, y^{(2)}, ..., y^{(k)} \in \{-1, 1\}$, each adviser has $\Pr(y^{(i)} = 1) = p$
- Assume each adviser acts independently

$$\Pr\left(\sum_i y^{(i)} > 0\right) = \sum_{i=1}^{m/2} \binom{m}{k} \cdot p^k (1-p)^{m-k}$$



Key assumption: independence!

Key observation: A large enough ensemble of independent weak learners converges to correct answer

- Suppose $f_1, ..., f_p$ are a collection of independent weak learners
- Each f_i are "better than random"
  - e.g. for binary classification,

$$\Pr\left(f_i(x) = y\right) > 50\%, \quad \forall i$$

- Then, the ensembled learner "converges to perfect"
  - e.g. for binary classification,

$$\Pr\left(\frac{1}{p}\sum_{i=1}^{p} f_i(x) = y\right) \overset{p\to\infty}{\to} 1$$

ICML 2006

An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

A number of supervised learning methods have been introduced in the last decade. Unfortunately, the last comprehensive empirical evaluation of supervised learning was the Statlog Project in the early 90's. We present a large-scale empirical comparison between ten supervised learning methods: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. We also examine the effect that calibrating the models via Platt Scaling and Isotonic Regression has on their performance. An important aspect of our study is the use of a variety of performance criteria to evaluate the learning methods.

This paper presents results of a large-scale empirical comparison of ten supervised learning algorithms using eight performance criteria. We evaluate the performance of SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score, Lift, ROC Area, average precision, precision/recall break-even point, squared error, and cross-entropy. For each algorithm we examine common variations, and thoroughly explore the space of parameters. For example, we compare ten decision tree styles, neural nets of many sizes, SVMs with many kernels, etc.

Because some of the performance metrics we examine interpret model predictions as probabilities and models such as SVMs are not designed to predict probabilities, we compare the performance of each algorithm

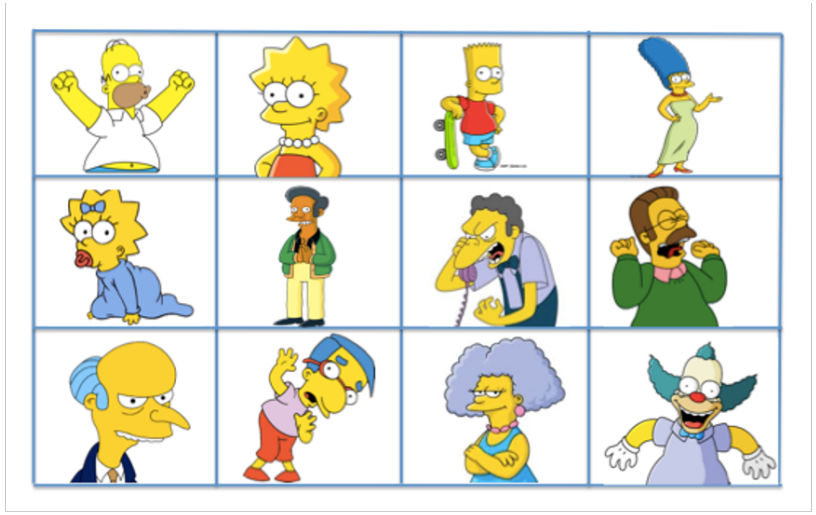
ICML 2006

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

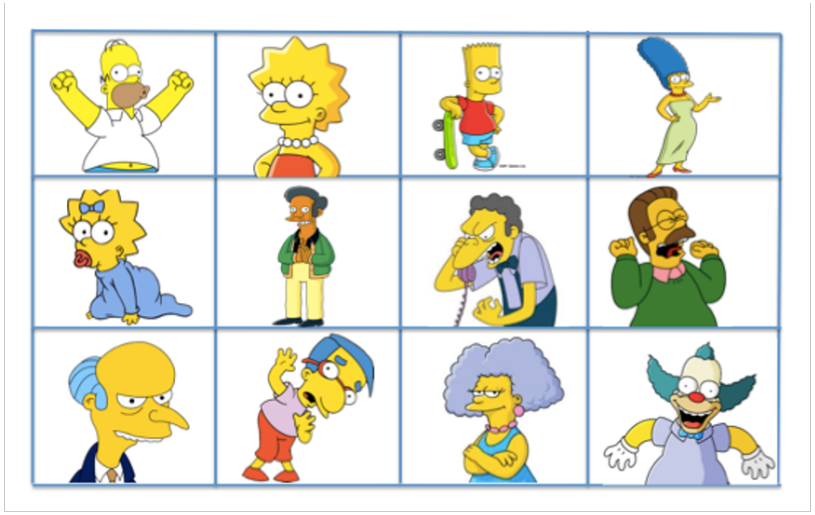
MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774
DT	–	.647	.639	.824	.843	.762	.777	.562	.607	.708	.763
DT	PLT	.651	.618	.824	.843	.762	.777	.575	.594	.706	.761
LR	–	.636	.545	.823	.852	.743	.734	.620	.645	.700	.710
LR	ISO	.627	.567	.818	.847	.735	.742	.608	.589	.692	.703
LR	PLT	.630	.500	.823	.852	.743	.734	.593	.604	.685	.695
NB	ISO	.579	.468	.779	.820	.727	.733	.572	.555	.654	.661
NB	PLT	.576	.448	.780	.824	.738	.735	.537	.559	.650	.654
NB	–	.496	.562	.781	.825	.738	.735	.347	-.633	.481	.489

Bootstrap and bagging

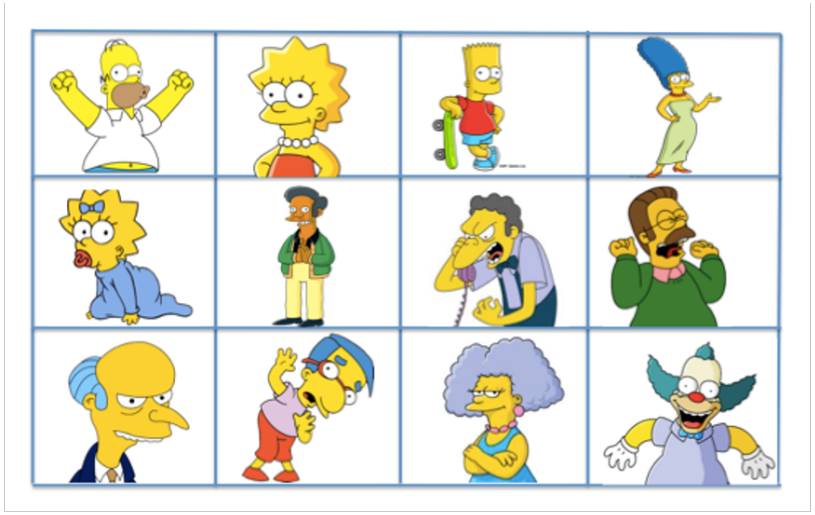
Bootstrap = sampling with replacement



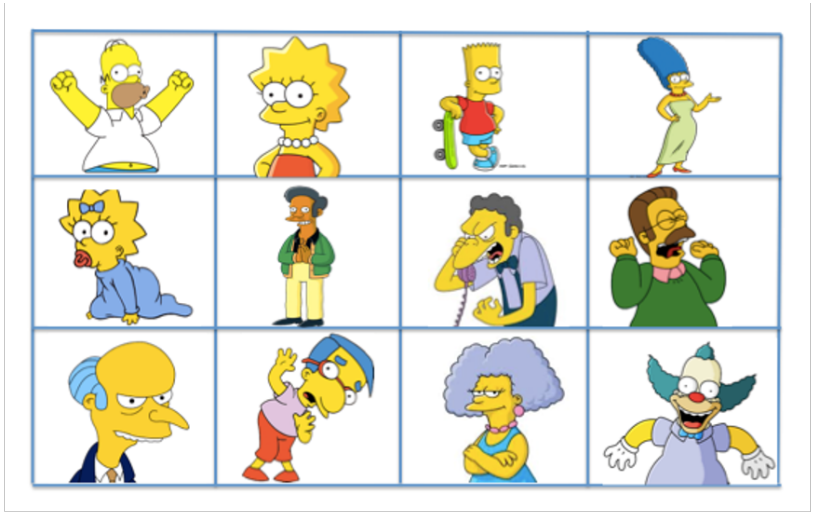
Bootstrap = sampling with replacement



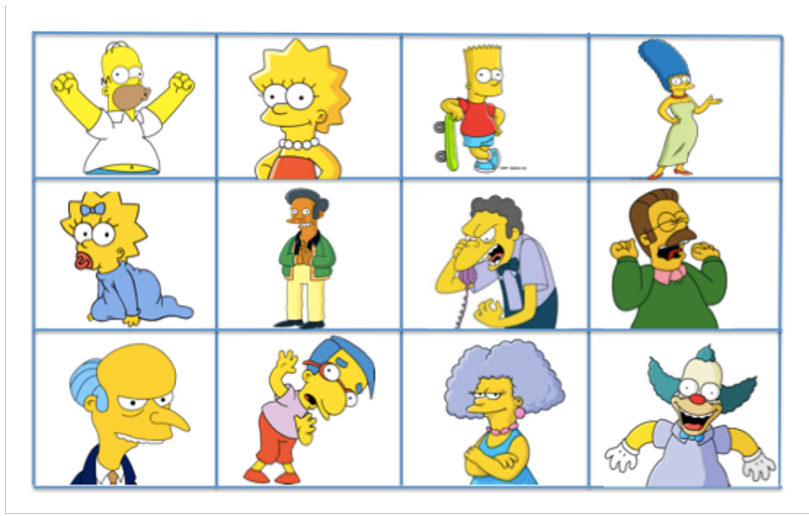
Bootstrap = sampling with replacement



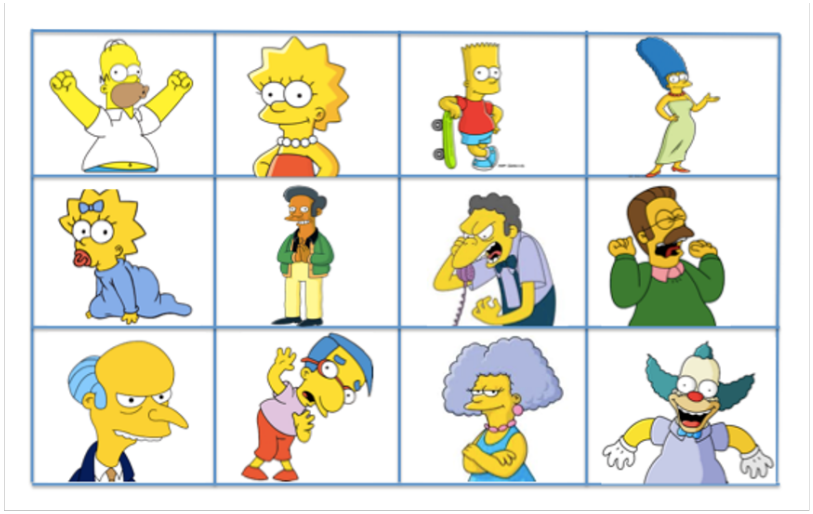
Bootstrap = sampling with replacement

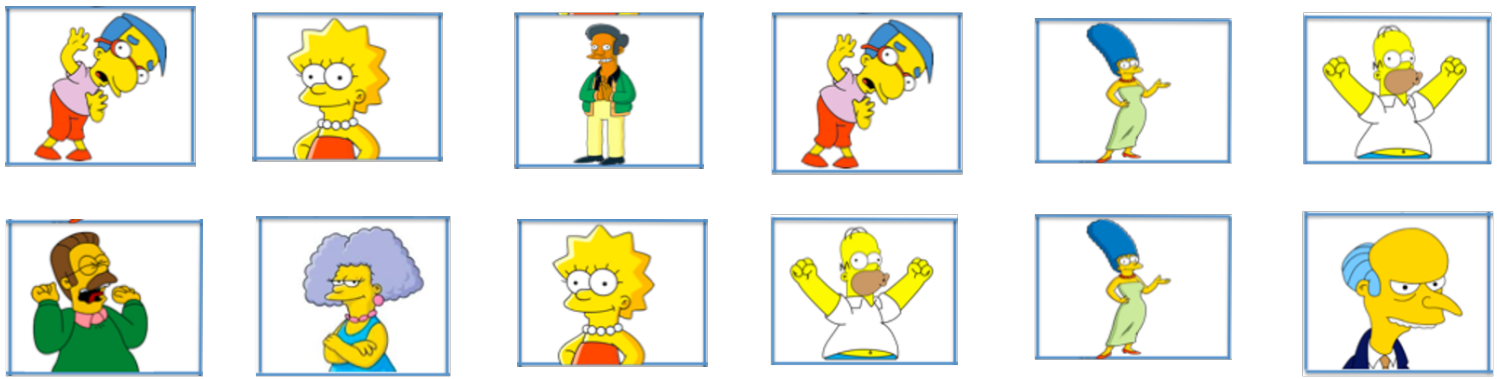


Bootstrap = sampling with replacement



Bootstrap = sampling with replacement





Never selected:

Bootstrap analysis

Chance of being picked?

- Take m training samples, bootstrap T times, subsamples size $s < m$
- Probability that a sample is never selected:

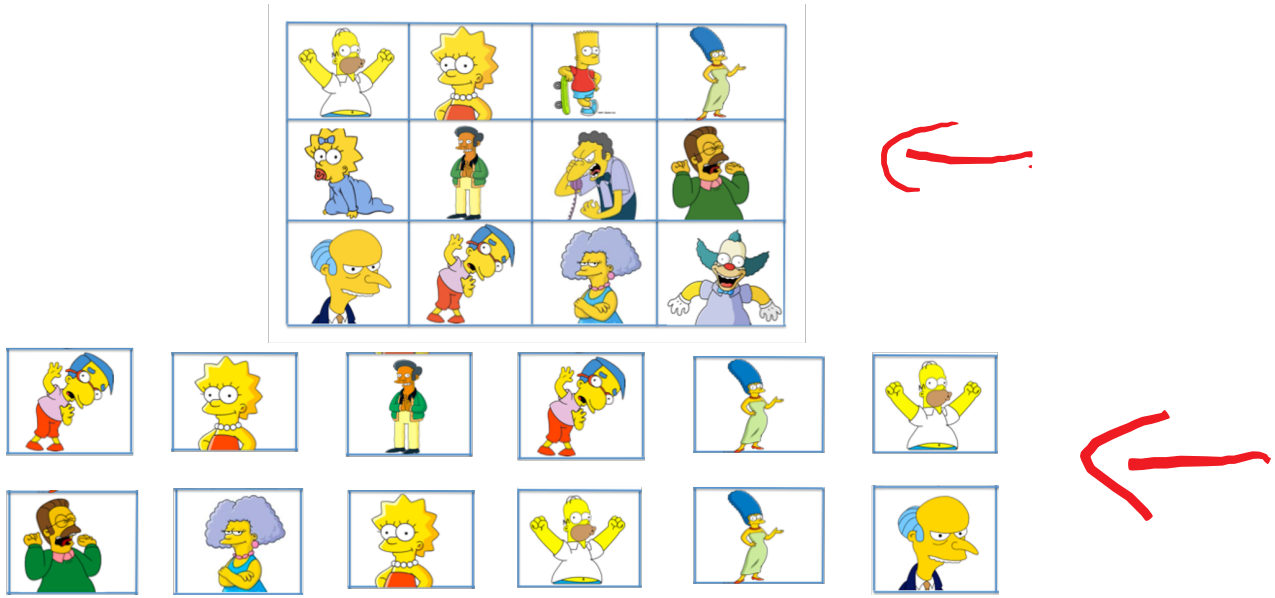
$$\Pr(\text{sample } x_i \text{ never picked}) = \left(1 - \frac{s}{m}\right)^T$$

- Constrain s, T such that $sT = m \iff 1/T = s/m$.

$$\lim_{T \to +\infty} \Pr(\text{sample } x_i \text{ never picked}) = \lim_{T \to +\infty} (1 - 1/T)^T = 1/e \approx 0.368$$

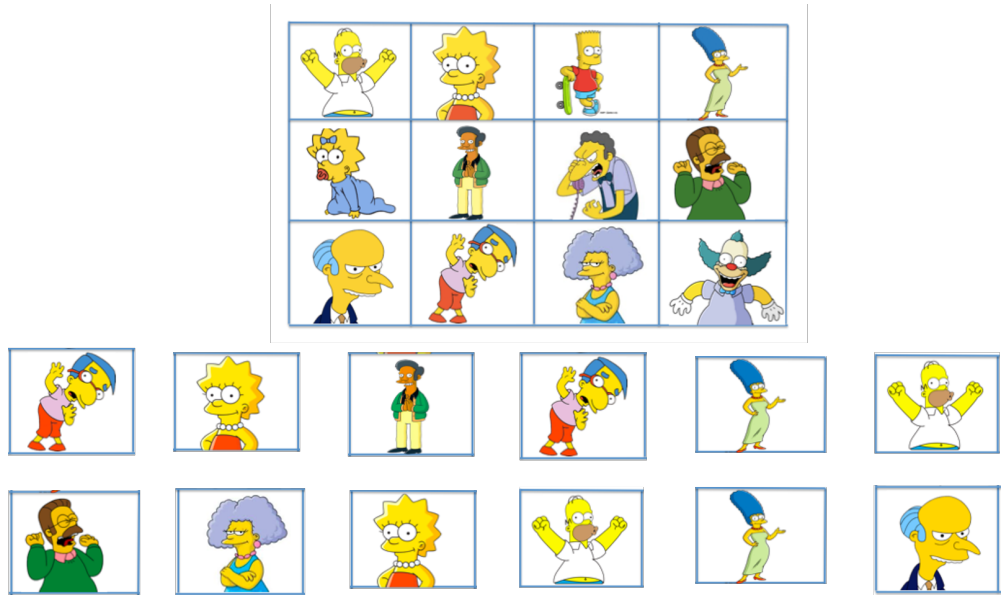
- In this scenario, asymptotically, $\approx$ 63.2% of data points will be selected

Statistical consistency



$$\Pr(\text{a female is selected}) =$$
$$\Pr(\text{a blue haired person is selected}) =$$
$$\Pr(\text{Homer is selected}) =$$

Statistical consistency

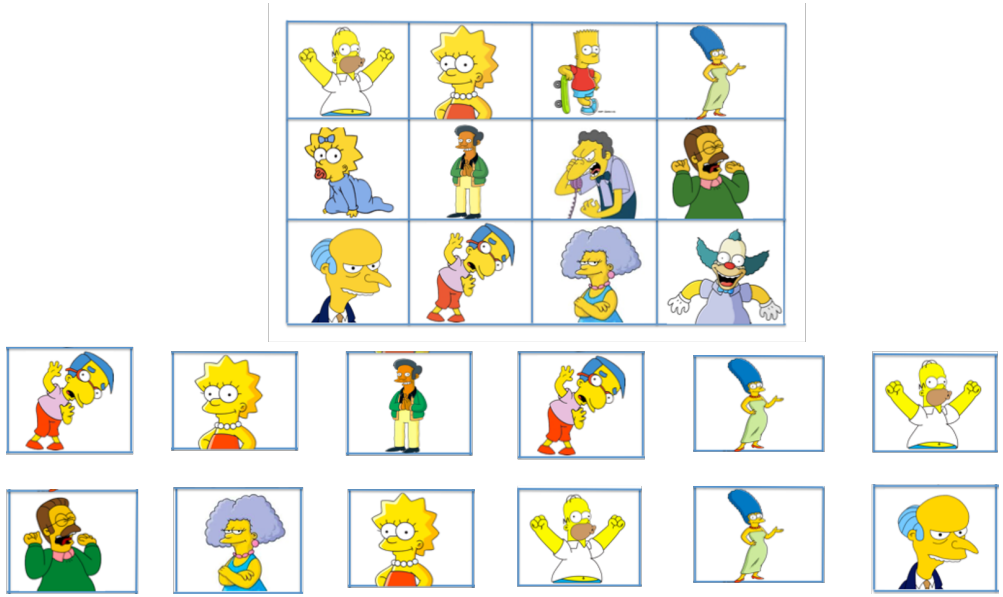


$$\Pr(\text{a female is selected}) = 1/3$$

$$\Pr(\text{a blue haired person is selected}) =$$

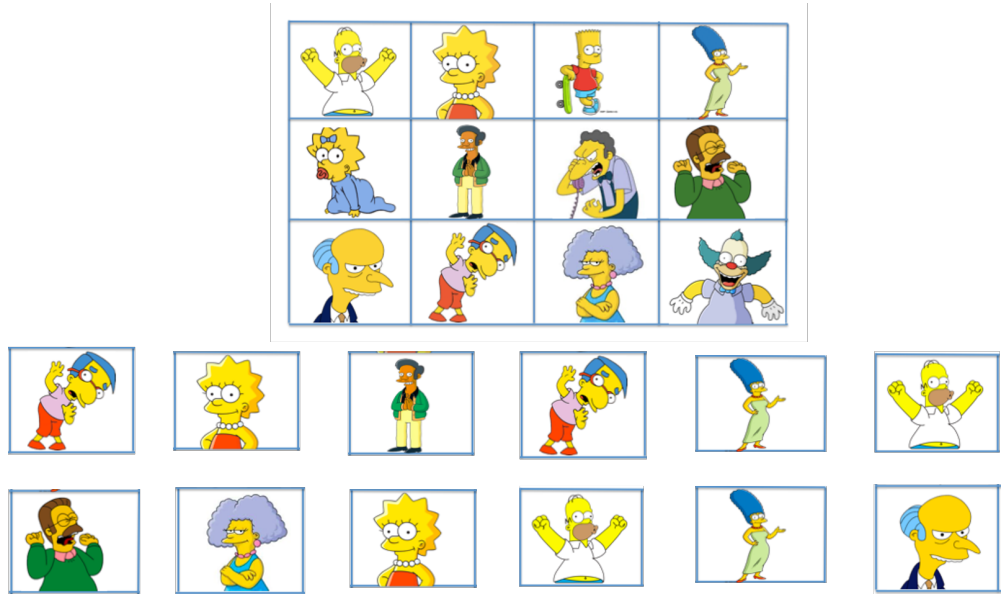
$$\Pr(\text{Homer is selected}) =$$

Statistical consistency



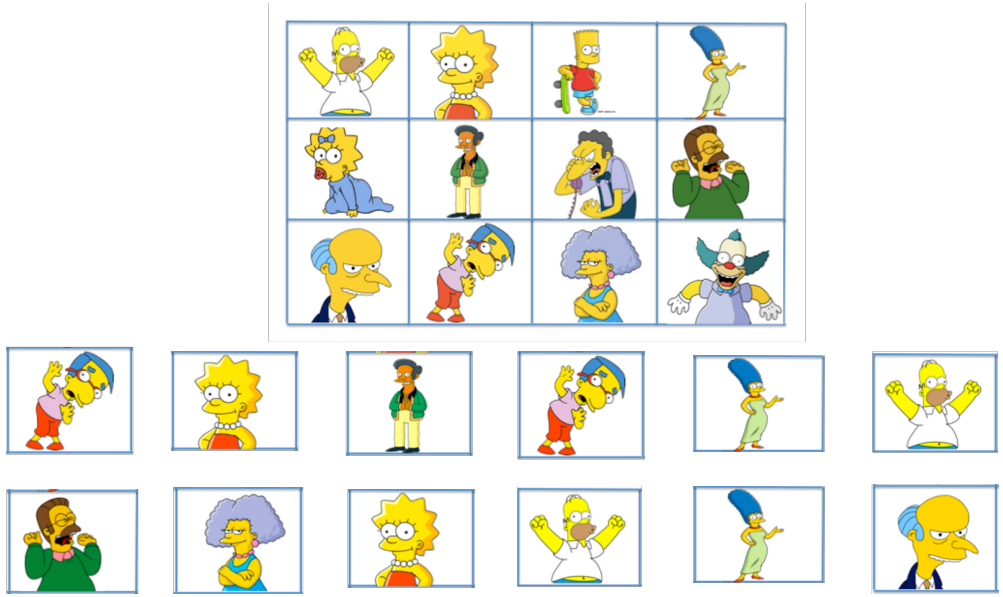
$$\begin{aligned} \Pr(\text{a female is selected}) &= 1/3 \\ \Pr(\text{a blue haired person is selected}) &= 1/4 \\ \Pr(\text{Homer is selected}) &= \end{aligned}$$

Statistical consistency



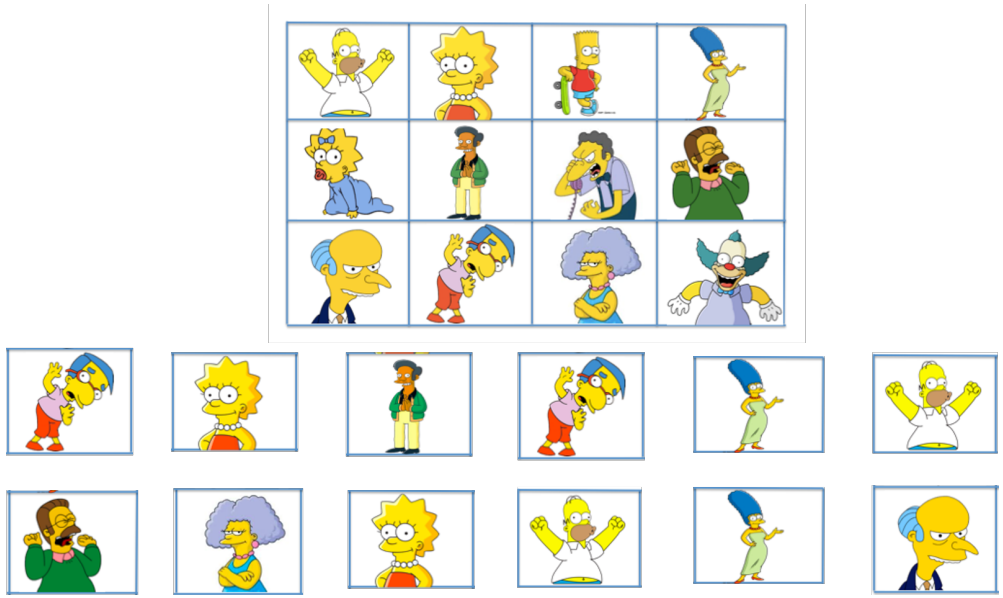
$$\begin{aligned}
\Pr(\text{a female is selected}) &= 1/3 \\
\Pr(\text{a blue haired person is selected}) &= 1/4 \\
\Pr(\text{Homer is selected}) &= 1/12
\end{aligned}$$

Statistical consistency



$$\Pr(\text{a female is selected}) = 1/3 \text{ (bootstrap } 5/12)$$
$$\Pr(\text{a blue haired person is selected}) = 1/4 \text{ (bootstrap } 1/3)$$
$$\Pr(\text{Homer is selected}) = 1/12 \text{ (bootstrap } 1/6)$$

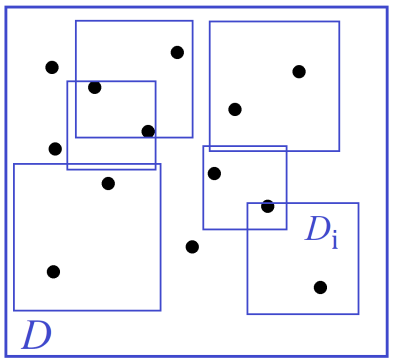
Statistical consistency



$$\begin{aligned}
\Pr(\text{a female is selected}) &= 1/3 \quad (\text{bootstrap } 5/12) \\
\Pr(\text{a blue haired person is selected}) &= 1/4 \quad (\text{bootstrap } 1/3) \\
\Pr(\text{Homer is selected}) &= 1/12 \quad (\text{bootstrap } 1/6)
\end{aligned}$$

Probability distributions in bootstrap shouldn't change

Bootstrap AGgregation (Bagging)



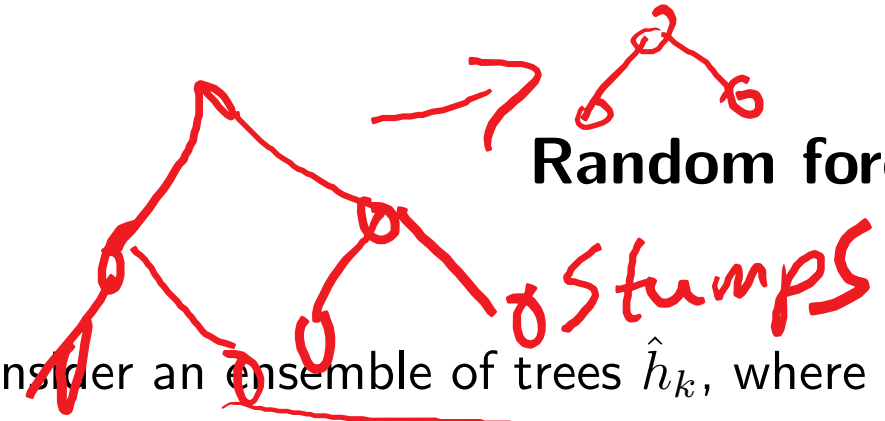
Given training set $\mathcal{D}$, sample (with replacement)

$$\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_B \subset \mathcal{D}$$

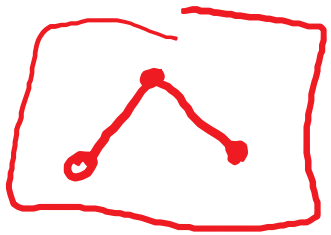
Model prediction:

$$\text{pred}_{\mathcal{D}}(x) = \begin{cases} \frac{1}{B}\sum_{i=1}^{B} \text{pred}_{\mathcal{D}_i}(x) & \text{(averaging)} \\ \underset{y}{\text{argmax}} \, |\{i : \text{pred}_{\mathcal{D}_i}(x) = y\}| & \text{(majority vote)} \\ \text{Some other method} \end{cases}$$

Random forest



- Consider an ensemble of trees $\hat{h}_k$, where each $\hat{h}_k$ is trained on
  - a random subset of data
  - a random subset of features *



- Classification: Final model via majority vote or averaging
- Regression:

$$\hat{H}_{\text{final}} = \sum_{k=1}^{\#\text{ models}} \alpha_k \hat{h}_k, \quad \alpha = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{\#\text{ data}} \| \sum_{k=1}^{\#\text{ models}} \alpha_k \hat{h}_k(x_i) - y_i \|_2^2$$

for this reason, was initially called "random subspace method" (Ho, '98)

Bagging analysis

- Take m training samples, bootstrap T times, subsamples size $s < m$
- $s \approx m$ (very large subsample) → stronger learner
  - Each predictor is strong, but variance amongst predictors is small
  - Why bother aggregating?
- $s \approx 0$ (very small subsample) → weaker learner
  - Each predictor is weak
  - Aggregation helps reduce variance, but bias still present

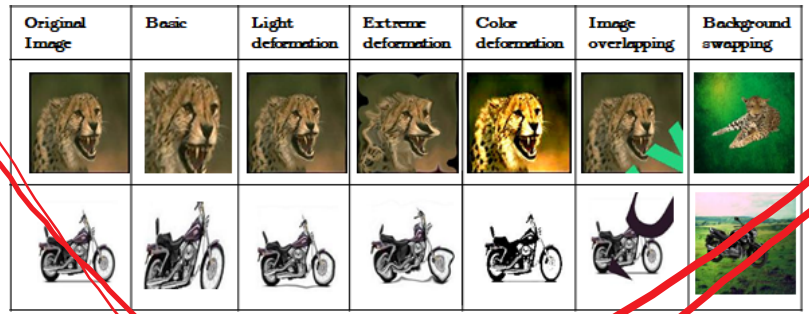
Variance reduction via bagging

- $h(x)$:= true signal
- $\hat{h}_{\mathcal{D}}(x)$:= model trained on one data draw
- $\hat{h}_{\mathcal{B}}(x) := \frac{1}{T}\sum_{i=1}^{T} \hat{h}_{\mathcal{D}_i}(x)$ = model trained via bagging
- Assume all draws are equivalent ($|\mathcal{D}_i| = |\mathcal{D}|$, $\forall i$)

Then:

- $\hat{h}_{\mathcal{B}}(x)$ is a sample average of i.i.d. draws of functions $\hat{h}_{\mathcal{D}}(x)$
- Therefore $\mathbf{var}(\hat{f}_{\mathcal{B}}(x)) = \frac{1}{T}\mathbf{var}(\hat{f}_{\mathcal{D}}(x)) \rightarrow$ increased stability
- If $\mathbf{var}(\hat{f}_{\mathcal{D}}(x))$ smaller than bias, then bagging will not help much

Related ideas: Data augmentation



Data Augmentation Benchmark for Deep Learning (Ràfols)

- Model aggregation: Fixed input, multiple models

$$y = \text{Agg}(\hat{h}_1(x), \hat{h}_2(x), ..., \hat{h}_T(x))$$

- Test-time data augmentation: Fixed model, multiple inputs

$$y = \text{Agg}(\hat{h}(x_1), \hat{h}(x_2), ..., \hat{h}(x_T))$$

where each x_i is a distorted copy of x

Experimental results

- UCI datasets, classification trees
- Bagging: 50 bootstrap replicates
- Repeat experiment 100 times on random train/test splits
- $\bar{e}_S$ is single tree, $\bar{e}_B$ is bagging

Table 1 Missclassification Rates (Percent)

Data Set	$\bar{e}_S$	$\bar{e}_B$	**Decrease**
waveform	29.0	19.4	33%
heart	10.0	5.3	47%
breast cancer	6.0	4.2	30%
ionosphere	11.2	8.6	23%
diabetes	23.4	18.8	20%
glass	32.0	24.9	22%
soybean	14.5	10.6	27%

Bagged Missclassification Rates (%)

No. Bootstrap Replicates	Missclassification Rate
10	21.8
25	19.5
50	19.4
100	19.4

It is reasonable to use between 25-50 bootstrap samples

Breiman, '94

Boosting

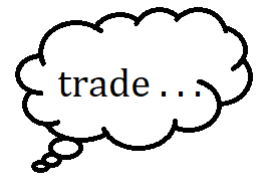
Sequential weak learners

$$\hat{H}(x) = \sum_{t=1}^{T} \alpha_t \hat{h}^{(t)}(x)$$

- Bagging: $\hat{h}^{(t)}$ picked independently, parallelizable
- Boosting: $\hat{h}^{(t)}$ picked sequentially

Example: Settlers of Catan

- Goal: collect resources, build roads
- Strategy: trade to overcome shortages
- Not everything is always equally valuable

Round					
1	2	0	0	0	Could use anything
2	3	0	1	0	Sheep or brick?
3	3	1	2	0	Brick?
4	5	1	2	0	Brick?!
5	7	1	2	0	No really, brick?!?!

Boosting: big idea

$$\hat{H}^{(T)}(x) = \sum_{t=1}^{T} \alpha_t \hat{h}^{(t)}(x)$$

- Pick a function class $\mathcal{F} = \{\hat{h}^{(t)}\}$
  - Decision trees
  - Neural networks
  - SVMs
- Pick best weak learner $\hat{h}^{(t)}$ to address current error

$$\alpha_t, \hat{h}^{(t)} = \underset{\alpha, h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{m} \mathcal{L}\left(\hat{H}^{(t-1)}(x_i) + \alpha_t \hat{h}^{(t)}(x_i), y_i\right)$$

- Merge weak learner $\alpha_t \hat{h}^{(t)}$ to aggregate learner $\hat{H}^{(t)}$

Boosting in linear regression

Goal:

$$\underset{h \in \mathcal{H}}{\text{minimize}} \sum_{i=1}^{m} \underbrace{(h(x_i) - y_i)^2}_{\mathcal{L}(y_i, h(x_i))}$$

Procedure:

$$\hat{h}^{(t)} = \underset{h \in \mathcal{H}}{\text{argmin}} \sum_{i=1}^{m} \left(\underbrace{y_i - H^{(t-1)}(x_i)}_{\text{residual } r_i^{(t)}} - \hat{h}(x_i) \right)^2$$

$$\hat{H}^{(t)} = H^{(t-1)} + \hat{h}^{(t)}$$

Observation: residual is negative gradient of $\mathcal{L}$ w.r.t. model, at time $t-1$

$$r_i^{(t)} = -\nabla_{\hat{y}} \mathcal{L}(\hat{y}, y_i)|_{\hat{y} = H^{(t-1)}(x_i)}$$

Gradient boosting

$$\underset{h \in \mathcal{H}}{\text{minimize}} \sum_{i=1}^{m} \mathcal{L}(h(x_i), y_i)$$

Gradient boosting

$$r_i^{(t)} := -\nabla_{\hat{y}} \mathcal{L}(\hat{y}, y_i)|_{\hat{y}=H^{(t-1)}(x_i)}, \quad i = 1, ..., m$$ Compute residual

$$\hat{h}^{(t)} = \underset{h \in \mathcal{H}}{\text{argmin}} \sum_{i=1}^{m} \left(\hat{h}(x_i) - r_i^{(t)}\right)$$ Train next weak learner

$$\hat{H}^{(t)} := \hat{H}^{(t-1)}(x_i) + \hat{h}^{(t)}$$ Merge weak learner

- Friedman 1999
- Weak learners are usually decision stumps

XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

ABSTRACT

Tree boosting is a highly effective and widely used machine learning method. In this paper, we describe a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. We propose a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

Keywords

Large-scale Machine Learning

1. INTRODUCTION

Machine learning and data-driven approaches are becoming very important in many areas. Smart spam classifiers protect our email by learning from massive amounts of spam data and user feedback; advertising systems learn to match the right ads with the right context; fraud detection systems protect banks from malicious attackers; anomaly event detection systems help experimental physicists to find events that lead to new physics. There are two important factors that drive these successful applications: usage of effective (statistical) models that capture the complex data

problems. Besides being used as a stand-alone predictor, it is also incorporated into real-world production pipelines for ad click through rate prediction [15]. Finally, it is the de-facto choice of ensemble method and is used in challenges such as the Netflix prize [3].

In this paper, we describe XGBoost, a scalable machine learning system for tree boosting. The system is available as an open source package[2]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions [3] published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success of the system was also witnessed in KDDCup 2015, where XGBoost was used by every winning team in the top-10. Moreover, the winning teams reported that ensemble methods outperform a well-configured XGBoost by only a small amount [1].

These results demonstrate that our system gives state-of-the-art results on a wide range of problems. Examples of the problems in these winning solutions include: store sales prediction; high energy physics event classification; web text classification; customer behavior prediction; motion detection; ad click through rate prediction; malware classification; product categorization; hazard risk prediction; massive on-

XGBoost

- Main improvement: regularization

$$f(\alpha) = \sum_{i=1}^{m} (H^{(t)}(x_i) - y_i)^2 + \lambda_0 \|\alpha\|_0 + \lambda_2 \|\alpha\|_2^2, \quad H^{(T)}(x) = \sum_{t=1}^{T} \alpha_t \hat{h}^{(t)}(x)$$

- Decision stumps $\rightarrow$ full trees, pruned via regularization
- Gradient boosting strategy using 2nd-order approximations of complicated loss functions
- Computation: Boosting = sequential, not parallelizable

Summary

- Ensemble learning: Mixture of experts for a better "advisory board"
- Bagging:
  - Bootstrap: sampling with replacement
  - Aggreagation: majority vote, averaging, you pick
  - Notable example: random forest
- Boosting
  - Unlike bagging, sequential in nature
  - Greedy quality, picking best classifier/regressor to cover current error
  - Regression: use gradient as residual
  - Notable example: XGBoost

Some material borrowed from Machine Learning course at UBC
https://www.cs.ubc.ca/ schmidtm/Courses/340-F19/L7.pdf