An Empirical Comparison of Learning Methods++

Rich Caruana
Cornell University

joint work with Alex Niculescu-Mizil, Cristi Bucila, Art Munson

Preliminaries:

What is Supervised Learning?

What is Supervised Learning?

- training set of labeled cases:

```
0 1 1 0 0 0.25 0.16 0.68  --> 0
0 1 0 1 0 0.20 0.09 0.77  --> 0
1 0 1 1 1 0.42 0.31 0.54  --> 1
0 0 1 1 0 0.58 0.29 0.63  --> 1
1 1 1 0 0 0.18 0.13 0.82  --> 0
…
1 0 1 0 1 0.66 0.43 0.52  --> 1
```

- learn model that predicts outputs in train set from input features
- use model to make predictions on (future) cases not used for training

```
0 0 1 1 1 0.64 0.38 0.55  --> ?
```

Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

- Each algorithm has many variations and free parameters:
  - SVM: margin parameter, kernel, kernel parameters (e.g. gamma), …
  - ANN: # hidden units, # hidden layers, learning rate, momentum, …
  - DT: splitting criterion, pruning options, smoothing options, …
  - KNN: K, distance metric, distance weighted averaging, …
- Must optimize to each problem:
  - failure to optimize makes superior algorithm inferior
  - optimization depends on criterion
    - e.g., for kNN: $k_{accuracy} \neq k_{ROC} \neq k_{RMSE}$
  - optimization depends on size of train set

Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

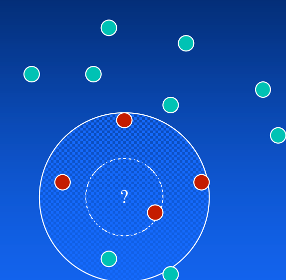
We have been too successful at developing new (improved?) learning methods.

Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

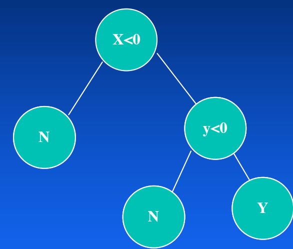
Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

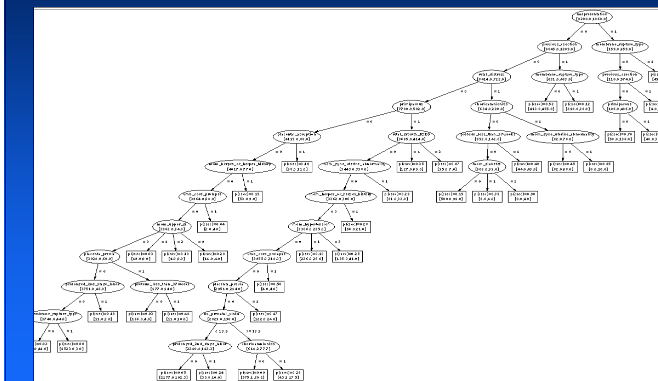


Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

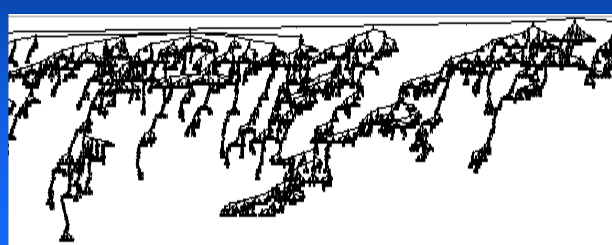


A Real Decision Tree



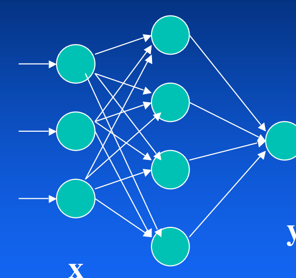
Not *ALL* Decision Trees Are Intelligible

This is part of one high-performing decision tree.

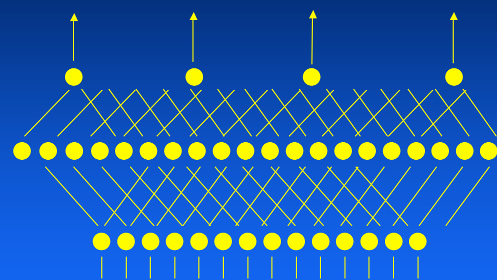


Sad State of Affairs: Supervised Learning

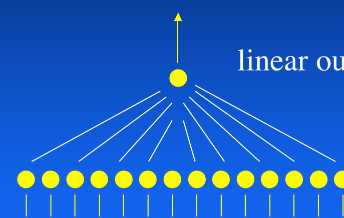
- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …



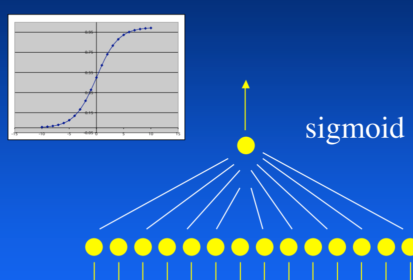
A Typical Neural Net



Linear Regression



Logistic Regression



Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

Sad State of Affairs: Supervised Learning

- Linear/polynomial Regression
- Logistic/Ridge regression
- K-nearest neighbor
- Linear perceptron
- Decision trees
- SVMs
- Neural nets
- Naïve Bayes
- Bayesian Neural nets
- Bayes Nets (Graphical Models)
- Bagging (bagged decision trees)
- Random Forests
- Boosting (boosted decision trees)
- Boosted Stumps
- ILP (Inductive Logic Programming)
- Rule learners (C2, …)
- Ripper
- Gaussian Processes
- …

- Each algorithm has many variations and free parameters:
  - SVM: margin parameter, kernel, kernel parameters (e.g. gamma), …
  - ANN: # hidden units, # hidden layers, learning rate, momentum, …
  - DT: splitting criterion, pruning options, smoothing options, …
  - KNN: K, distance metric, distance weighted averaging, …
- Must optimize to each problem:
  - failure to optimize makes superior algorithm inferior
  - optimization depends on criterion
    - e.g., for kNN: $k_{accuracy} \neq k_{ROC} \neq k_{RMSE}$
  - optimization depends on size of train set

Questions

- Is one algorithm "better" than the others?
- Are some learning methods best for certain loss functions?
  - SVMs for classification?
  - ANNs for regression or predicting probabilities?
- If no method(s) dominate, can we at least ignore some algs?
- Why are some methods good on loss X, but poor on loss Y?
- How do different losses relate to each other?
- Are some losses "better" than others?
- …
- What should you use ???

Data Sets

- 8 binary classification data sets (now 10 sets)
  - Adult
  - Cover Type
  - Letter.p1
  - Letter.p2

  UCI
  - Pneumonia
  - Hyper Spectral
  - SLAC Particle Physics
  - Mg

  "Real"
- 4 k train sets
- 1 k validation sets

  Training Data
- Large final test sets (usually 20k)

Binary Classification Performance Metrics

- Threshold Metrics:
  - Accuracy
  - F-Score
  - Lift
- Ordering/Ranking Metrics:
  - ROC Area
  - Average Precision
  - Precision/Recall Break-Even Point
- Probability Metrics:
  - Root-Mean-Squared-Error
  - Cross-Entropy
  - Probability Calibration

Normalized Scores

- Small Difficulty:
  - some metrics, 1.00 is best (e.g. ACC)
  - some metrics, 0.00 is best (e.g. RMS)
  - some metrics, baseline is 0.50 (e.g. AUC)
  - some metrics, best depends on data (e.g. Lift)
  - some problems/metrics, 0.60 is excellent performance
  - some problems/metrics, 0.99 is poor performance
- Solution: Normalized Scores:
  - baseline performance => 0.00
  - best observed performance => 1.00 (proxy for Bayes optimal)
  - puts all metrics/problems on equal footing

Massive Empirical Comparison

10 learning methods
X
100's of parameter settings per method
X
5-fold cross validation
=
10,000+ models trained per problem
X
11 Boolean classification test problems
=
110,000+ models
X
9 performance metrics
=
1,000,000+ model evaluations

Look at Predicting Probabilities First

- Why?
  - don't want to hit you with results for nine metrics all at once
  - if you can predict correct conditional probabilities, you're done–all reasonable performance metrics are optimized by predicting true probabilities
  - results for probabilities are interesting by themselves*

* Alex Niculescu-Mizil won best student paper award at ICML05 for this work on predicting probabilities

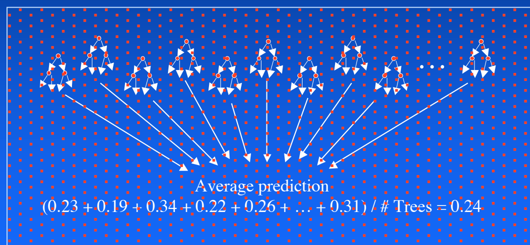
Results on Test Sets (Normalized Scores)

	Probability Metrics			
Model	**Squared Error**	**Cross-Entropy**	**Calibration**	**Mean**
ANN	**0.872**	**0.878**	**0.826**	**0.859**
BAG-DT	**0.875**	**0.901**	0.637	0.804
RND-FOR	**0.882**	**0.899**	0.567	0.783
KNN	0.783	0.769	0.684	0.745
LOG-REG	0.614	0.620	0.678	0.637
DT	0.583	0.638	0.512	0.578
BST-DT	0.596	0.598	0.045	0.413
SVM [0,1]	0.484	0.447	0.000	0.310
BST-STMP	0.355	0.339	0.123	0.272
NAÏVE-B	0.271	0.00	0.00	0.090

- Best probabilities overall:
  - Neural Nets
  - Bagged Decision Trees
  - Random Forests
- Not competitive:
  - Boosted decision trees and stumps (exponential loss)
  - SVMs (standard hinge-loss)
- SVMs scaled to [0,1] via simple min/max scaling

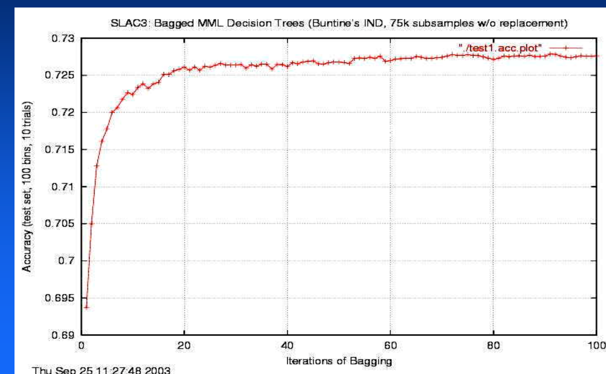
Bagged Decision Trees

- Draw 100 bootstrap samples of data
- Train trees on each sample -> 100 trees
- Un-weighted average prediction of trees



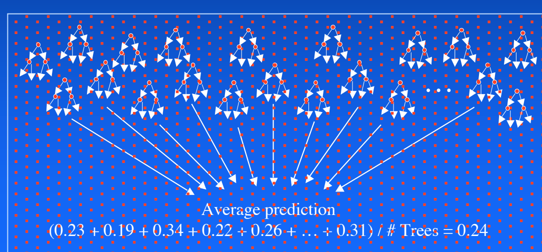
- Highly under-rated!

Bagging Results



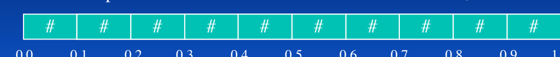
Random Forests (Bagged Trees++)

- Draw *1000+* bootstrap samples of data
- ***Draw sample of available attributes at each split***
- Train trees on each sample/attribute set -> *1000+* trees
- Un-weighted average prediction of trees

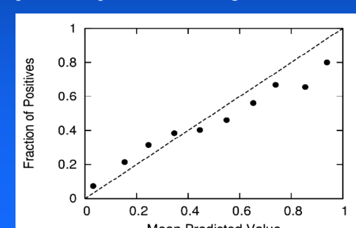


Calibration & Reliability Diagrams

- If on 100 days the forecast is 70% chance of rain, predictions well calibrated at p = 0.70 if it actually rains about 70/100 days
- Put cases with predicted values between 0 and 0.1 in first bin, …



- For each bin, plot mean predicted value against true fraction of positives:

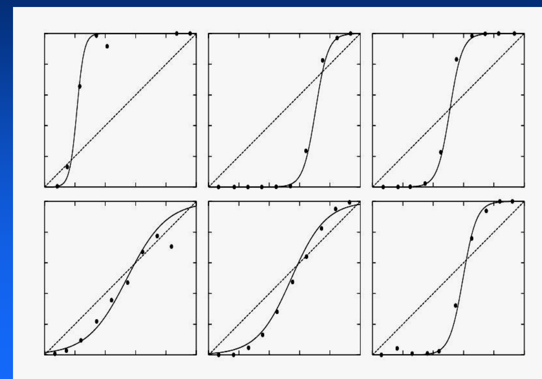


Back to SVMs: Results on Test Sets

Model	Probability Metrics: Squared Error	Cross-Entropy	Calibration	Mean
ANN	**0.872**	**0.878**	**0.826**	**0.859**
BAG-DT	**0.875**	**0.901**	0.637	0.804
RND-FOR	**0.882**	**0.899**	0.567	0.783
KNN	0.783	0.769	0.684	0.745
LOG-REG	0.614	0.620	0.678	0.637
DT	0.583	0.638	0.512	0.578
BST-DT	0.596	0.598	0.045	0.413
SVM [0,1]	0.484	0.447	0.000	0.310
BST-STMP	0.355	0.339	0.123	0.272
NAÏVE-B	0.271	0.00	0.00	0.090

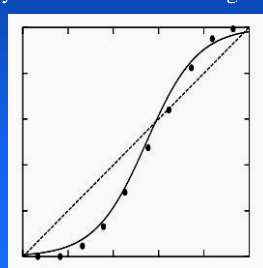
- Best probabilities overall:
  - Neural Nets
  - Bagged Probabilistic Trees
  - Random Forests
- Not competitive:
  - Boosted decision trees and stumps (with exponential loss)
  - SVMs (with hinge loss)
- **SVMs scaled to [0,1] via simple min/max scaling**

SVM Reliability Plots



Platt Scaling by Fitting a Sigmoid

- Linear scaling of SVM [-∞,+∞] predictions to [0,1] is bad
- Platt's Method [Platt 1999]:
  - scale predictions by fitting sigmoid on a *validation set* using 3-fold CV and Bayes-motivated smoothing to avoid overfitting



Results After Platt Scaling SVMs

Model	Probability Metrics: Squared Error	Cross-Entropy	Calibration	Mean
ANN	**0.872**	**0.878**	**0.826**	**0.859**
SVM-PLT	**0.882**	**0.880**	**0.769**	**0.844**
BAG-DT	**0.875**	**0.901**	0.637	0.804
RND-FOR	**0.882**	**0.899**	0.567	0.783
KNN	0.783	0.769	0.684	0.745
LOG-REG	0.614	0.620	0.678	0.637
DT	0.583	0.638	0.512	0.578
BST-DT	0.596	0.598	0.045	0.413
SVM [0,1]	0.484	0.447	0.000	0.310
BST-STMP	0.355	0.339	0.123	0.272
NAÏVE-B	0.271	0.00	0.00	0.090

- Platt's Method (Platt 1999) for obtaining posterior probabilities from SVMs by fitting a sigmoid
- SVM probabilities as good as Neural Net probabilities after scaling with Platt's Method
- SVMs slightly better than Neural Nets on 2 of 3 metrics!
- Would other learning methods benefit from calibration with Platt's Method?

Results After Platt Scaling SVMs

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics			
Model	Accuracy	F-Score	Lift	ROC Area	Average Precision	Break Even Point	Squared Error	Cross-Entropy	Calibration	Mean
ANN	0.817	**0.875**	**0.947**	0.963	0.926	0.929	**0.872**	**0.878**	**0.826**	**0.892**
SVM-PLT	**0.823**	0.851	0.928	0.961	0.931	0.929	**0.882**	**0.880**	0.769	**0.884**
BAG-DT	**0.836**	0.849	**0.953**	**0.972**	**0.950**	0.928	**0.875**	**0.901**	0.637	**0.878**
RND-FOR	**0.844**	0.845	**0.958**	**0.977**	**0.957**	**0.948**	**0.882**	**0.899**	0.567	**0.875**
KNN	0.759	0.839	0.914	0.937	0.893	0.898	0.783	0.769	0.684	0.831
BST-DT	**0.861**	**0.885**	**0.956**	**0.977**	**0.958**	**0.952**	0.596	0.598	0.045	0.758
DT	0.612	0.789	0.856	0.871	0.789	0.808	0.583	0.638	0.512	0.717
BST-STMP	0.701	0.782	0.898	0.926	0.871	0.854	0.355	0.339	0.123	0.650
LOG-REG	0.602	0.623	0.829	0.849	0.732	0.714	0.614	0.620	0.678	0.696
NAÏVE-B	0.414	0.637	0.746	0.767	0.698	0.689	0.271	0.00	0.00	0.469

- Boosted trees outperform everything else on 5 of 6 non-probability metrics.
- But boosting predicts poor probabilities.

Results After Platt Scaling SVMs

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics		
Problem	Accuracy	F-Score	Lift	ROC Area	Average Precision	Break Even Point	Squared Error	Cross-Entropy	Calibration
Adult	BST-ST	DT	BST-ST	BST-ST	BST-ST	BST-ST	BAG-DT	BAG-DT	LOGREG
Covtype	BST-DT	BST-DT	BST-DT	BST-DT	BST-DT	BST-DT	RF	BAG-DT	KNN
HyperSpec	ANN	ANN	ANN	BST-DT	BST-DT	BST-DT	ANN	ANN	ANN
Letter 1	BST-DT	BST-DT	SVM	BST-DT	BST-DT	BST-DT	KNN	KNN	ANN
Letter 2	BST-DT	BST-DT	KNN	BST-DT	BST-DT	BST-DT	KNN	KNN	ANN
Medis	BST-ST	NB	ANN	ANN	ANN	ANN	ANN	ANN	ANN
MG2	BAG-DT	DT	BST-ST	BAG-DT	BAG-DT	BAG-DT	BAG-DT	BAG-DT	ANN
SLAC	RF	SVM	BAG-DT	RF	BAG-DT	RF	BAG-DT	BAG-DT	ANN

Summary of Model Performances

Model	Best Count	Mean NS
ANN	17	0.892
SVM-PLT	2	0.884
BAG-DT	13	0.878
RND-FOR	4	0.875
KNN	6	0.831
BST-DT	19	0.758
DT	2	0.717
BST-STMP	7	0.650
LOG-REG	1	0.696
NAÏVE-B	1	0.469

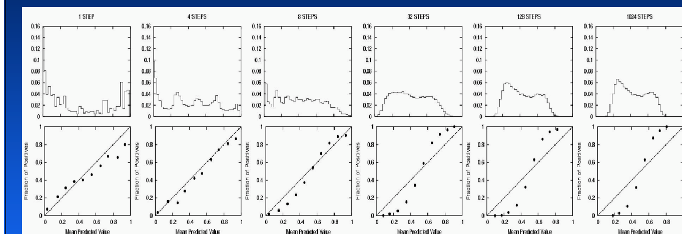
Smart Model ≠ Good Probs

- Model can be accurate, but be poorly calibrated
  - Only sensitive to side of threshold case falls on
  - Use threshold ≠ 0.5 if poorly calibrated
- Model can have good ROC (Google-like ordering), but predict poor probabilities
  - ROC insensitive to scaling/stretching
  - Only ordering has to be correct, not probabilities

Ada Boosting

- Initialization:
  - Weight all training samples equally
- Iteration (typically requires 100’s to 1000’s of iterations):
  - Train model on (weighted) train set
  - Compute error of model on train set
  - Increase weights on cases model gets wrong
- Return final model:
  - Carefully weighted prediction of each model

Why Boosting is Not Well Calibrated



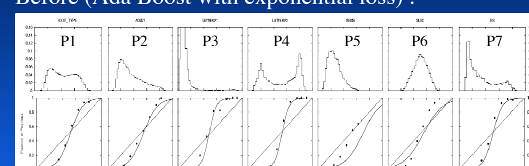
- Predicted values pushed away from 0 and 1
- Calibration becomes increasingly worse
- Shape of the reliability plot becomes sigmoidal
- Looks a lot like SVM predictions

Consistent With Interpretations of Boosting

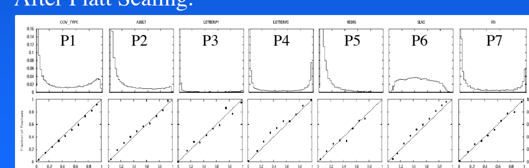
- Boosting is (almost) a maximum-margin method (Schapire et al. 1998, Rosset et al. 2004)
  - Trades lower margin on easy cases for higher margin on harder cases
- Boosting is an additive logistic regression model (Friedman, Hastie and Tibshirani 2000)
  - Tries to fit the logit of the true conditional probabilities
- Boosting is an *equalizer* (Breiman 1998) (Friedman, Hastie, Tibshirani 2000)
  - Weighted proportion of times example is misclassified by base learners tends to be the same for all training cases

Platt Scaling of Boosted Trees (7 problems)

Before (Ada Boost with exponential loss) :



After Platt Scaling:



Results After Platt Scaling All Models

Model	Probability Metrics: Squared Error	Cross-Entropy	Calibration	Mean
BST-DT	**0.929**	**0.932**	**0.808**	**0.890**
ANN	0.872	0.878	**0.826**	0.859
SVM-PLT	0.882	0.880	0.769	0.844
RND-FOR	0.892	0.898	0.702	0.831
BAG-DT	0.875	0.901	0.637	0.804
KNN	0.786	0.805	0.706	0.766
BST-STMP	0.740	0.783	0.678	0.734
LOG-REG	0.614	0.620	0.678	0.637
DT	0.586	0.625	0.688	0.633
NAÏVE-B	0.539	0.565	0.161	0.422

- Models that benefit from calibration:
  - SVMs
  - Boosted decision trees
  - Boosted stumps
  - Random forests
  - Naïve Bayes
  - Vanilla decision trees
- Do not benefit from calibration:
  - Neural nets
  - Bagged trees
  - Logistic regression
  - MBL/KNN
- Boosting full trees dominates

Revenge of the Decision Tree!

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics			
Model	Accuracy	F-Score	Lift	ROC Area	Average Precision	Break Even Point	Squared Error	Cross-Entropy	Calibration	Mean
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884
BAG-DT	0.836	0.849	0.953	0.972	**0.950**	0.928	0.875	0.901	0.637	0.878
KNN	0.759	0.820	0.914	0.937	0.893	0.898	0.786	0.805	0.706	0.835
BST-STMP	0.698	0.760	0.898	0.926	0.871	0.854	0.740	0.783	0.678	0.801
DT	0.611	0.771	0.856	0.871	0.789	0.808	0.586	0.625	0.688	0.734
LOG-REG	0.602	0.623	0.829	0.849	0.732	0.714	0.614	0.620	0.678	0.696
NAÏVE-B	0.536	0.615	0.786	0.833	0.733	0.730	0.539	0.565	0.161	0.611

- After Platt Scaling, boosted trees are best models overall across all metrics
- Neural nets are best models overall if no calibration is applied post-training

Methods for Achieving Calibration

- **Optimize directly to appropriate criterion:**
  - Boosting with log-loss (Collins, Schapire & Singer 2001)
  - SVM to maximize likelihood (e.g. Wahba 1999)
    - + performance comparable to Platt Scaling (Platt 1999)
    - + yields non-sparse solution
  - No need for post-training calibration with these approaches
- **Train models with "usual" criterion and post-calibrate:**
  - Logistic Correction
    - + **Analytic** method justified by the Friedman et al.'s analysis
  - Platt Scaling
    - + Method used by Platt to calibrate SVMs by fitting a sigmoid
    - + Is sigmoid right calibration function for most learning methods?
  - Isotonic Regression
    - + Very **general** calibration method used by Zadrozny & Elkan (2001)
    - + PAV (Pair Adjacent Violators) algorithm (Ayer et al. 1955)
    - + **Optimal for squared error**
    - + Efficient linear-time algorithm

Boosting with Log-Loss

Model	Probability Metrics: Squared Error	Cross-Entropy	Calibration	Mean
BST-DT PLATT	**0.929**	**0.932**	**0.808**	**0.890**
ANN	0.872	0.878	**0.826**	0.859
SVM-PLT	0.882	0.880	0.769	0.844
RND-FOR	0.892	0.898	0.702	0.831
BAG-DT	0.875	0.901	0.637	0.804
KNN	0.786	0.805	0.706	0.766
BST-STMP PLATT	0.740	0.783	0.678	0.734
LOG-REG	0.614	0.620	0.678	0.637
DT	0.586	0.625	0.688	0.633
BST-DT LOG-LOSS	0.581	0.590	0.231	0.467
BST-STMP LOG-LOSS	0.627	0.659	0.094	0.460
NAÏVE-B	0.539	0.565	0.161	0.422

- Log-loss does improve calibration of boosting, but
- Most effective with weak models such as 1-level stumps
- Less effective with more complex models such as full decision trees (or even 2-level stumps)
- Post-calibration with Platt's Method far more effective, particularly with complex models such as full trees
- Best probabilities come from full trees boosted with exponential loss then calibrated with Platt's Method

Boosting with Log-Loss

	Probability Metrics			
Model	Squared Error	Cross-Entropy	Calibration	Mean
BST-DT PLATT	**0.929**	**0.932**	**0.808**	**0.890**
ANN	0.872	0.878	**0.826**	0.859
SVM-PLT	0.882	0.880	0.769	0.844
RND-FOR	0.892	0.898	0.702	0.831
BAG-DT	0.875	0.901	0.637	0.804
KNN	0.786	0.805	0.706	0.766
BST-STMP PLATT	0.740	0.783	0.678	0.734
LOG-REG	0.614	0.620	0.678	0.637
DT	0.586	0.625	0.688	0.633
BST-DT LOG-LOSS	0.581	0.590	0.231	0.467
BST-STMP LOG-LOSS	0.627	0.659	0.094	0.460
NAÏVE-B	0.539	0.565	0.161	0.422

- Log-loss does improve calibration of boosting, but
- Most effective with weak models such as 1-level stumps
- Less effective with more complex models such as full decision trees (or even 2-level stumps)
- Post-calibration with Platt's Method far more effective, particularly with complex models such as full trees
- Best probabilities come from full trees boosted with exponential loss then calibrated with Platt's Method

Isotonic Regression

- Basic assumption - there exists an isotonic (monotonically increasing) function *m* s.t.:

$$y_i = m(f_i) + \epsilon_i$$

- We want to find an isotonic function *m* s.t.:

$$\hat{m} = argmin_z \sum (y_i - z(f_i))^2$$

- Bianca Zadrozny and Charles Elkan (2001) first to use isotonic regression for calibration in ML community

PAV Algorithm

Algorithm 1. PAV algorithm for estimating posterior probabilities from uncalibrated model predictions.

1 Input: training set (f_i, y_i) sorted according to f_i

2 Initialize $m_{i,i} = y_i$, $w_{i,i} = 1$

3 While $\exists\, i\ s.t.\ \hat{m}_{k,i-1} \geq \hat{m}_{i,l}$

Set $w_{k,l} = w_{k,i-1} + w_{i,l}$

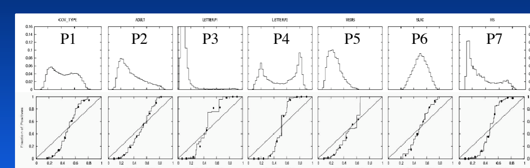
Set $\hat{m}_{k,l} = (w_{k,i-1}\hat{m}_{k,i-1} + w_{i,l}\hat{m}_{i,l})/w_{k,l}$

Replace $\hat{m}_{k,i-1}$ and $\hat{m}_{i,l}$ with $\hat{m}_{k,l}$

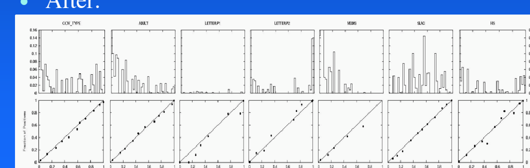
4 Output the stepwise const. function generated by $\hat{m}$

Isotonic Regression

- Before:

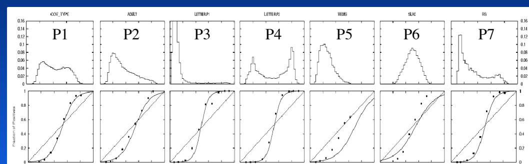


- After:

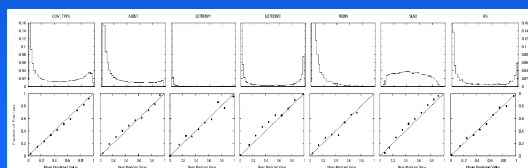


Platt Scaling

- Before:

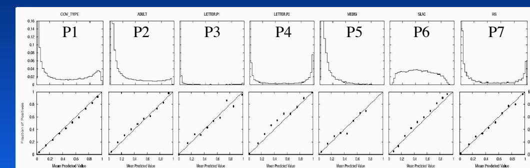


- After:

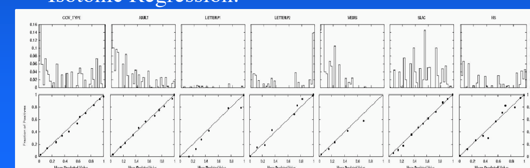


Platt Scaling vs. Isotonic Regression

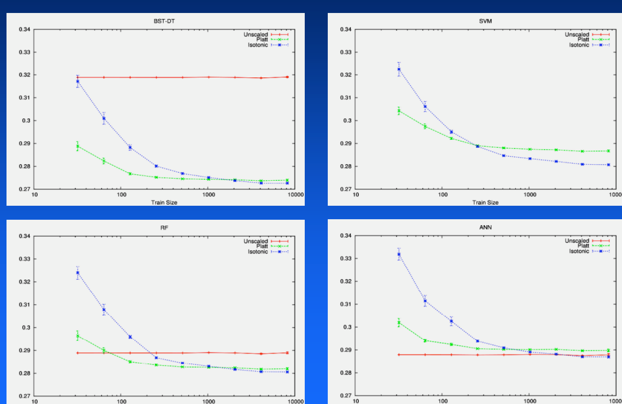
- Platt Scaling:



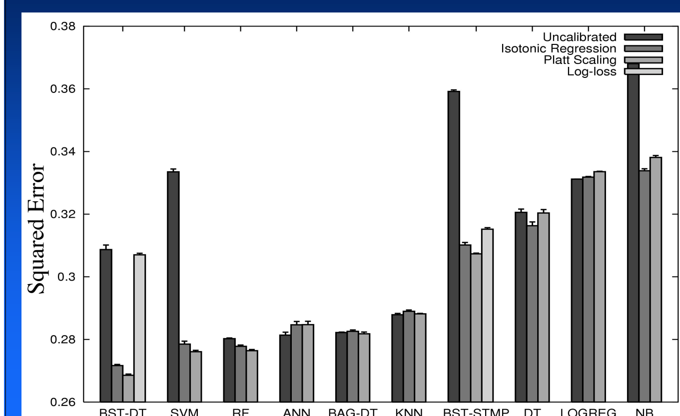
- Isotonic Regression:



Platt Scaling vs. Isotonic Regression



Summary: Before/After Calibration



Summary

- Boosting full trees outperforms boosting weaker models
- Calibration via Platt Scaling or Isotonic Regression more effective than boosting log-loss when boosting trees
- Platt Scaling better with small data (< 1000 points)
- Isotonic Regression better with large data (> 1000 points)
- Not all learning methods benefit from calibration
- Before calibration, well-tuned neural nets predict the best probabilities
- After calibration, boosted probabilistic decision trees predict best probabilities

Where Does That Leave Us?

- Calibration via Platt Scaling or Isotonic Regression improves probs from max-margin methods such as Boosted Trees and SVMs
- Boosted Trees + Calibration best overall
- Are we done?
- No!

Best of the Best of the Best

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics			
Model	Accuracy	F-Score	Lift	ROC Area	Average Precision	Break Even Point	Squared Error	Cross-Entropy	Calibration	Mean
BEST	**0.928**	**0.918**	**0.975**	**0.987**	**0.958**	**0.958**	**0.919**	**0.944**	**0.989**	**0.9533**
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884
BAG-DT	0.836	0.849	0.953	0.972	**0.950**	0.928	0.875	0.901	0.637	0.878
KNN	0.759	0.820	0.914	0.937	0.893	0.898	0.786	0.805	0.706	0.835
BST-STMP	0.698	0.760	0.898	0.926	0.871	0.854	0.740	0.783	0.678	0.801
DT	0.611	0.771	0.856	0.871	0.789	0.808	0.586	0.625	0.688	0.734
LOG-REG	0.602	0.623	0.829	0.849	0.732	0.714	0.614	0.620	0.678	0.696
NAÏVE-B	0.536	0.615	0.786	0.833	0.733	0.730	0.539	0.565	0.161	0.611

Best of the Best of the Best

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics			
Model	Accuracy	F-Score	Lift	ROC Area	Average Precision	Break Even Point	Squared Error	Cross-Entropy	Calibration	Mean
BEST	**0.928**	**0.918**	**0.975**	**0.987**	**0.958**	**0.958**	**0.919**	**0.944**	**0.989**	**0.9533**
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884

- Selecting best of all learning methods yields significantly better performance
- No one model type can do it all (yet)
- If you really need best performance, need to try multiple learning algorithms
- If we look at individual problems, even more variation in what works best
  - Boosted stumps outperform boosted trees on 3 of 8 problems!
  - Logistic regression is best model on one problem!

If we need to train all models and pick best, can we do better than picking best?

"A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse."

-- Tom Dietterich (2000)

Current Ensemble Methods

- Bagging
- Boosting
- Random Forests
- Error Correcting Output Codes (ECOC) …

- Average of multiple models
- Bayesian Model Averaging
- Stacking …

- Ensemble methods differ in:
  - how models are generated
  - how models are combined

Normalized Scores of Ensembles

	Threshold Metrics			Rank/Ordering Metrics			Probability Metrics			
Model	**Accuracy**	**F-Score**	**Lift**	**ROC Area**	**Average Precision**	**Break Even Point**	**Squared Error**	**Cross-Entropy**	**Calibration**	**Mean**
BAYESAVG	0.9258	0.8906	0.9785	0.9851	0.9773	0.9557	0.9504	0.9585	0.9871	0.9566
BEST	0.9283	0.9188	0.9754	0.9876	0.9588	0.9581	0.9194	0.9443	0.9891	0.9533
AVG_ALL	0.8363	0.8007	0.9815	0.9878	0.9721	0.9606	0.8271	0.8086	0.9856	0.9067
STACK_LR	0.2753	0.7772	0.8352	0.7992	0.7860	0.8469	0.3317	-0.9897	0.8221	0.4982
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884
BAG-DT	0.836	0.849	0.953	0.972	**0.950**	0.928	0.875	0.901	0.637	0.878
KNN	0.759	0.820	0.914	0.937	0.893	0.898	0.786	0.805	0.706	0.835
BST-STMP	0.698	0.760	0.898	0.926	0.871	0.854	0.740	0.783	0.678	0.801

New Ensemble Method: ES

- Train *many* different models:
  - different algorithms
  - different parameter settings
  - all trained on same train set
  - all trained to "natural" optimization criterion
- Add *all* models to library:
  - no model selection
  - no validation set
  - some models bad, some models good, a few models excellent
  - ***yields diverse set of models, some of which are good on almost any metric***
- Forward stepwise *model selection* from library:
  - start with empty ensemble
  - try adding each model one-at-a-time to ensemble
  - commit model that maximizes performance on metric on a test set
  - repeat until performance stops getting better

Basic Ensemble Selection Algorithm

Model Library	Ensemble
Model 1	
Model 2	
Model 3	
Model 4	
Model 5	
Model 6	
Model 7	
Model 8	
Model 9	

Basic Ensemble Selection Algorithm

Model Library	AUC Score on the 1k validation set	Ensemble
Model 1	0.8453	
Model 2	0.8726	
Model 3	0.9164	
Model 4	0.8142	
Model 5	0.8453	
Model 6	0.8745	
Model 7	0.9024	
Model 8	0.7034	
Model 9	0.8342	

Basic Ensemble Selection Algorithm

Model Library	AUC Score on the 1k validation set	Ensemble
Model 1	0.8453	
Model 2	0.8726	
Model 3	0.9164	
Model 4	0.8142	
Model 5	0.8453	
Model 6	0.8745	
Model 7	0.9024	
Model 8	0.7034	
Model 9	0.8342	

Basic Ensemble Selection Algorithm

Model Library	Ensemble
Model 1	Model 3 0.9164
Model 2	
Model 4	
Model 5	
Model 6	
Model 7	
Model 8	
Model 9	

Basic Ensemble Selection Algorithm

Model Library		AUC Score on the 1k validation set		Ensemble	
Model 1	+ Ensemble =	0.8327	0.8453	Model 3	0.9164
Model 2		0.8702	0.8726		
Model 4		0.9284	0.8142		
Model 5		0.9047	0.8453		
Model 6		0.8832	0.8745		
Model 7		0.9126	0.9024		
Model 8		0.8245	0.7034		
Model 9		0.9384	0.8342		

Basic Ensemble Selection Algorithm

Model Library		AUC Score on the 1k validation set	Ensemble	
Model 1	+ Ensemble =	0.8327	Model 3	0.9164
Model 2		0.8702		
Model 4		0.9284		
Model 5		0.9047		
Model 6		0.8832		
Model 7		0.9126		
Model 8		0.8245		
Model 9		0.9384		

Basic Ensemble Selection Algorithm

Model Library	Ensemble	
Model 1	Model 3	0.9164
Model 2	Model 9	0.9384
Model 4		
Model 5		
Model 6		
Model 7		
Model 8		

Basic Ensemble Selection Algorithm

Model Library		AUC Score on the 1k validation set		Ensemble	
Model 1	+ Ensemble =	0.8502	0.8327	Model 3	0.9164
Model 2		0.9243	0.8702	Model 9	0.9384
Model 4		0.8992	0.9284		
Model 5		0.8090	0.9047		
Model 6		0.9424	0.8832		
Model 7		0.9045	0.9126		
Model 8		0.9243	0.8245		

Basic Ensemble Selection Algorithm

Model Library

AUC Score on the 1k validation set

Model Library		AUC Score on the 1k validation set
Model 1		0.8502
Model 2		0.9243
Model 4		0.8992
Model 5	+ Ensemble =	0.8090
Model 6		0.9424
Model 7		0.9045
Model 8		0.9243

Ensemble

Model 3 0.9164
Model 9 0.9384

Basic Ensemble Selection Algorithm

Model Library

Model 1
Model 2

Model 4
Model 5

Model 7
Model 8

Ensemble

Model 3 0.9164
Model 9 0.9384
Model 6 0.9424

Big Problem: Overfitting

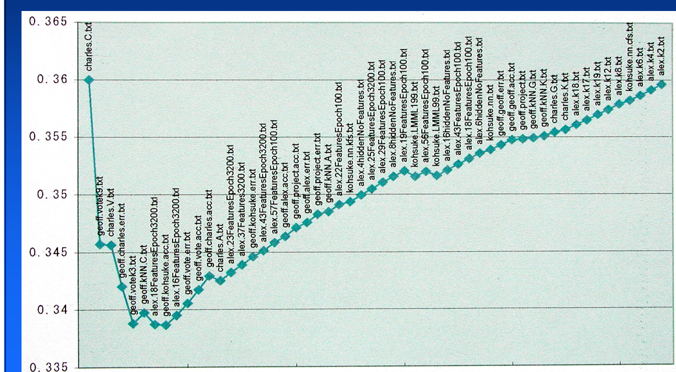
- More models ==> better chance of finding combination with good performance on any given problem and metric,
- but …
- also better chance of overfitting to the hillclimb set

- Tricks to Reduce Overfitting:
  - Eliminate Inferior Models: prevents mistakes
  - Ensemble Initialization: give “inertia” to initial ensemble
  - Stepwise Selection with Replacement: stopping point less critical
  - Calibrate Models in Ensemble: all models speak same language
  - Bagged Ensemble Selection: reduces variance

- Critical to take steps to reduce overfitting

1st Trick: Ensemble Initialization

Instead of starting with empty ensemble, initialize with best N models



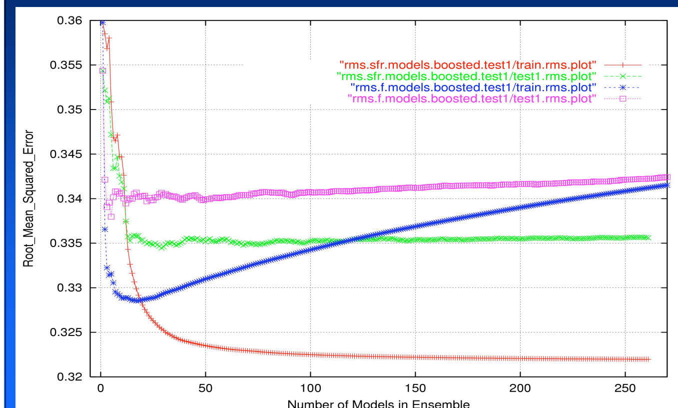
2nd Trick: Selection with Replacement

- After initializing ensemble with best N models

- Forward Selection with Replacement:
  - add each model one-at-a-time to ensemble
  - models added by averaging predictions
  - calculate performance metric
  - commit model that improves performance most
  - repeat until ensemble too large (we typically use ~ 250 steps)
  - return ensemble with best performance on validation set

  - models added 3 times have 3X weight of models added once
  - simple form of model weighting is less prone to overfitting

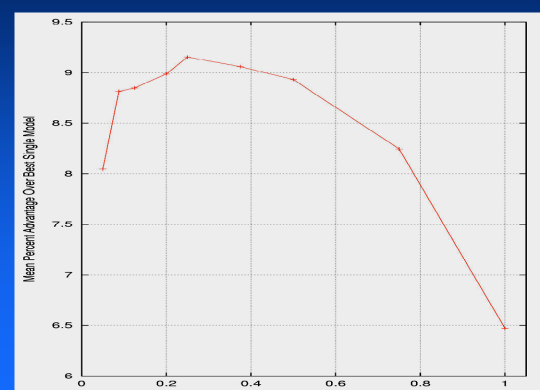
Forward Selection With Replacement



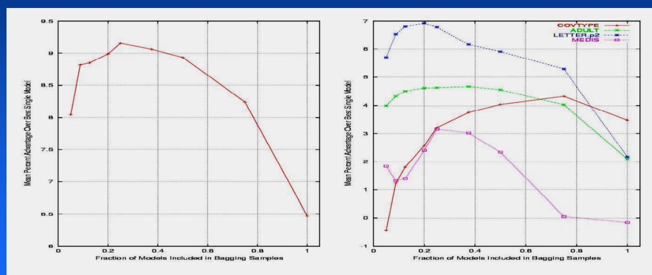
3rd Trick: Bagged Selection

- Draw a sample of models from library (we use p = 0.5)
- Do ensemble selection from this sample of models
- Repeat N times (we use N=20)
- Final model is average of the N ensembles
  - each ensemble is simple weighted average of base-level models
  - average of N such ensembles also is a simple weighted average of the base-level models

Benefit of Bagged Selection



Benefit of Bagged Selection



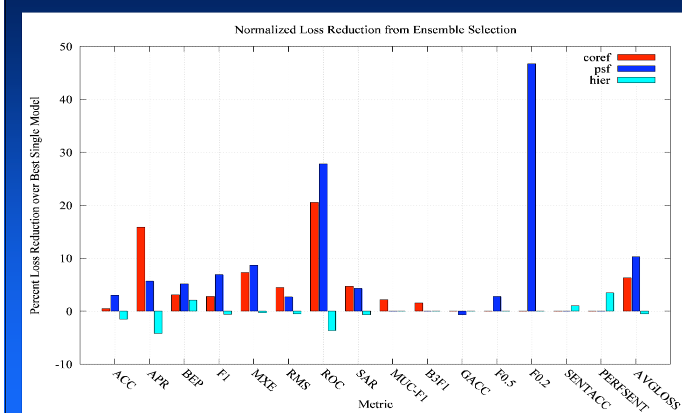
Normalized Scores for ES

Model	Threshold Metrics: Accuracy	Threshold Metrics: F-Score	Threshold Metrics: Lift	Rank/Ordering Metrics: ROC Area	Rank/Ordering Metrics: Average Precision	Rank/Ordering Metrics: Break Even Point	Probability Metrics: Squared Error	Probability Metrics: Cross-Entropy	Probability Metrics: Calibration	Mean
ES	**0.9560**	**0.9442**	**0.9916**	**0.9965**	**0.9846**	**0.9786**	**0.9795**	**0.9808**	0.9877	**0.9777**
BAYESAVG	0.9258	0.8906	0.9785	0.9851	0.9773	0.9557	0.9504	0.9585	0.9871	0.9566
BEST	0.9283	0.9188	0.9754	0.9876	0.9588	0.9581	0.9194	0.9443	0.9891	0.9533
AVG_ALL	0.8363	0.8007	0.9815	0.9878	0.9721	0.9606	0.8271	0.8086	0.9856	0.9067
STACK_LR	0.2753	0.7772	0.8352	0.7992	0.7860	0.8469	0.3317	-0.9897	0.8221	0.4982
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884
BAG-DT	0.836	0.849	0.953	0.972	**0.950**	0.928	0.875	0.901	0.637	0.878
KNN	0.759	0.820	0.914	0.937	0.893	0.898	0.786	0.805	0.706	0.835
BST-STMP	0.698	0.760	0.898	0.926	0.871	0.854	0.740	0.783	0.678	0.801

Normalized Scores for ES

Model	Threshold Metrics: Accuracy	Threshold Metrics: F-Score	Threshold Metrics: Lift	Rank/Ordering Metrics: ROC Area	Rank/Ordering Metrics: Average Precision	Rank/Ordering Metrics: Break Even Point	Probability Metrics: Squared Error	Probability Metrics: Cross-Entropy	Probability Metrics: Calibration	Mean
ES	**0.9560**	**0.9442**	**0.9916**	**0.9965**	**0.9846**	**0.9786**	**0.9795**	**0.9808**	0.9877	**0.9850**
BAYESAVG	0.9258	0.8906	0.9785	0.9851	0.9773	0.9557	0.9504	0.9585	0.9871	0.9566
BEST	0.9283	0.9188	0.9754	0.9876	0.9588	0.9581	0.9194	0.9443	0.9891	0.9533
AVG_ALL	0.8363	0.8007	0.9815	0.9878	0.9721	0.9606	0.8271	0.8086	0.9856	0.9067
STACK_LR	0.2753	0.7772	0.8352	0.7992	0.7860	0.8469	0.3317	-0.9897	0.8221	0.4982
BST-DT	**0.860**	**0.854**	**0.956**	**0.977**	**0.958**	**0.952**	**0.929**	**0.932**	**0.808**	**0.914**
RND-FOR	**0.866**	0.871	**0.958**	**0.977**	**0.957**	**0.948**	0.892	0.898	0.702	0.897
ANN	**0.817**	**0.875**	**0.947**	0.963	0.926	0.929	0.872	0.878	**0.826**	0.892
SVM	0.823	0.851	0.928	0.961	0.931	0.929	0.882	0.880	0.769	0.884
BAG-DT	0.836	0.849	0.953	0.972	**0.950**	0.928	0.875	0.901	0.637	0.878
KNN	0.759	0.820	0.914	0.937	0.893	0.898	0.786	0.805	0.706	0.835
BST-STMP	0.698	0.760	0.898	0.926	0.871	0.854	0.740	0.783	0.678	0.801

Ensemble Selection vs Best: 3 NLP Problems



[Art Munson, Claire Cardie, Rich Caruana. EMNLOP/HLDT 2005]

Ensemble Selection Works, But Is It Worth It?

Computational Cost

- Have to train multiple models anyway
  - models can be trained in parallel
    - + different packages, different machines, at different times, by different people
  - just generate and collect (no optimization necessary, no test sets)
  - saves human effort -- no need to examine/optimize models
  - ~ 48 hours on 10 workstations to train 2000 models with 5k train sets
  - model library can be built before optimization metric is known
  - anytime selection -- no need to wait for all models
- Ensemble Selection is cheap:
  - each iteration, consider adding 2000+ models to ensemble
  - adding model is simple unweighted averaging of predictions
  - caching makes this very efficient
  - compute performance metric when each model is added
  - for 250 iterations, evaluate 250*2000 = 500,000 ensembles
  - ~ 1 minute on workstation if metric is not expensive

What Models are Used in Ensembles?

ADULT	Acc	Fsc	Lft	Roc	Apr	Bep	Rms	Mxe	Sar	Avg
ANN	.071	.132	.101	.365	.430	.243	.167	.094	.573	.272
KNN	.020	.015	.586	.037	.029	.049	.000	.000	.049	.098
SVM	.001	.000	.110	.284	.274	.092	.057	.000	.022	.105
DT	.020	.035	.007	.088	.049	.019	.746	.867	.234	.258
BAG_DT	.002	.001	.000	.004	.005	.002	.006	.010	.014	.005
BST_DT	.110	.152	.025	.057	.032	.047	.024	.028	.075	.069
BST_STMP	.776	.666	.171	.166	.181	.548	.000	.000	.032	.317
COV-TYPE	Acc	Fsc	Lft	Roc	Apr	Bep	Rms	Mxe	Sar	Avg
ANN	.011	.010	.001	.038	.023	.009	.087	.097	.052	.041
KNN	.179	.166	.576	.252	.295	.202	.436	.427	.364	.362
SVM	.021	.016	.087	.104	.106	.051	.010	.013	.038	.056
DT	.061	.054	.012	.238	.242	.029	.408	.368	.200	.202
BAG_DT	.005	.006	.002	.010	.015	.006	.016	.022	.044	.016
BST_DT	.553	.613	.130	.278	.240	.644	.042	.073	.292	.358
BST_STMP	.170	.134	.194	.080	.080	.059	.000	.000	.009	.091

What Models are Used in Ensembles?

ADULT	Acc	Fsc	Lft	Roc	Apr	Bep	Rms	Mxe	Sar	Avg
ANN	.071	.132	.101	.365	.430	.243	.167	.094	.573	.272
KNN	.020	.015	.586	.037	.029	.049	.000	.000	.049	.098
SVM	.001	.000	.110	.284	.274	.092	.057	.000	.022	.105
DT	.020	.035	.007	.088	.049	.019	.746	.867	.234	.258
BAG_DT	.002	.001	.000	.004	.005	.002	.006	.010	.014	.005
BST_DT	.110	.152	.025	.057	.032	.047	.024	.028	.075	.069
BST_STMP	.776	.666	.171	.166	.181	.548	.000	.000	.032	.317
COV-TYPE	Acc	Fsc	Lft	Roc	Apr	Bep	Rms	Mxe	Sar	Avg
ANN	.011	.010	.001	.038	.023	.009	.087	.097	.052	.041
KNN	.179	.166	.576	.252	.295	.202	.436	.427	.364	.362
SVM	.021	.016	.087	.104	.106	.051	.010	.013	.038	.056
DT	.061	.054	.012	.238	.242	.029	.408	.368	.200	.202
BAG_DT	.005	.006	.002	.010	.015	.006	.016	.022	.044	.016
BST_DT	.553	.613	.130	.278	.240	.644	.042	.073	.292	.358
BST_STMP	.170	.134	.194	.080	.080	.059	.000	.000	.009	.091

What Models are Used by ES?

- Most ensembles use 10-100 of the 2000 models
- Different models are selected for different problems
- Different models are selected for different metrics
- Most ensembles use a diversity of model types
- Most ensembles use different parameter settings
- Selected Models often make sense:
  - Neural nets for RMS, Cross-Entropy
  - Max-margin methods for Accuracy
  - Large k in knn for AUC

ES Pros & Cons

- Disadvantages:
  - Have to train many models
    - + If you want the best, you were going to do it anyway
    - + Packages such as WEKA and MLC++ make it easier
  - Loss of intelligibility
  - No cool theory!
- Advantages:
  - Can optimize to almost any performance metric
  - Better performance than anything else we compared to

Ensemble Selection

- Good news:
  - A carefully selected ensemble that combines many models outperforms boosting, bagging, random forests, SVMs, and neural nets, … (because it builds on top of them)
- Bad news:
  - The ensembles are too big, too slow, too cumbersome to use for most applications

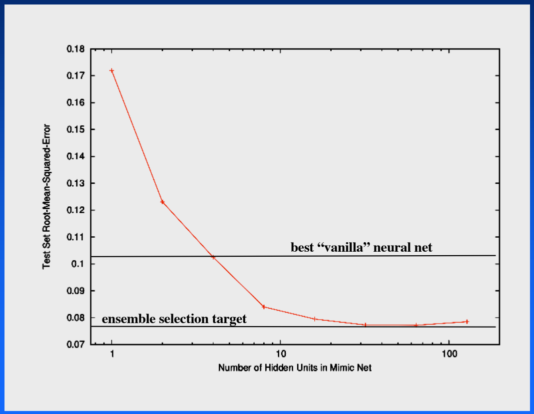
Best Ensembles are Big and Ugly!

- Best ensemble for one problem/metric has 422 models:
  - 72 boosted trees (28,642 individual decision trees!)
  - 1 random forest (1024 decision trees)
  - 5 bagged trees (100 decision trees in each model)
  - 44 neural nets (2,200 hidden units,total, >100,000 weights)
  - 115 knn models (both large and expensive!)
  - 38 SVMs (100's of support vectors in each model)
  - 26 boosted stump models (36,184 stumps total -- could compress)
  - 122 individual decision trees
  - …
- Best ensemble:
  - takes ~1GB to store model
  - takes ~2 seconds to execute per test case!

Solution: Model Compression

- Train simple model to mimic the complex model
- Pass large amounts of unlabeled data (synthetic data points or real unlabeled data) through ensemble and collect predictions
  - 100,000 to 10,000,000 synthetic training points
  - Extensional representation of the ensemble model
- Train ***copycat*** model on this large synthetic train set to mimic the high-performance ensemble
  - Train neural net to mimic ensemble
  - Potential to not only perform as well as target ensemble, but possibly outperform it

Work In Progress (Cristi Bucila)



Results

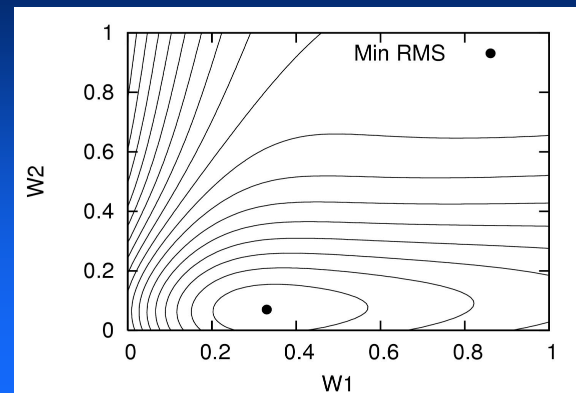
- Neural nets trained to mimic high performing bagged tree models
  - perform *better* than the target models on eight test problems and three test metrics
  - perform much better than any ANN we could train on the original data
- Massive experiment using ensemble selection predictions and nine performance metrics currently underway
  - getting ensemble predictions is much more expensive
  - willing to trade off cost at train- time for speed and compactness at run-time

Why Mimic with Neural Nets?

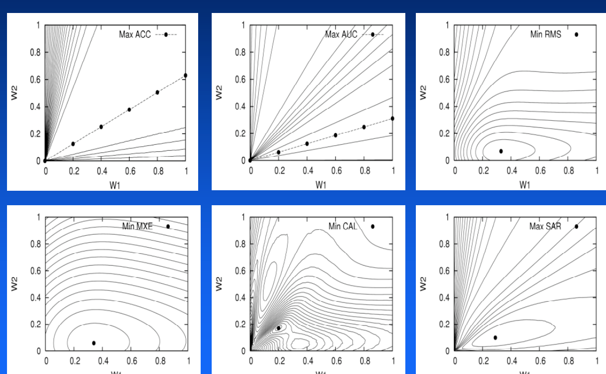
- Decision trees do not work well
  - synthetic data must be very large because of recursive partitioning
  - mimic decision trees are enormous (depth > 1000 and > 10^6 nodes) making them expensive to store and compute
  - single tree does not seem to model ensemble accurately enough
- SVMs
  - number of support vectors increases quickly with complexity
- Artificial Neural nets
  - can model complex functions with modest # of hidden units
  - can compress millions of training cases into thousands of weights
  - expense to train, but execution cost is low (just matrix multiplies)
  - models with few thousand weights have small footprint

How Important is it to Optimize to the Correct Performance Metric?

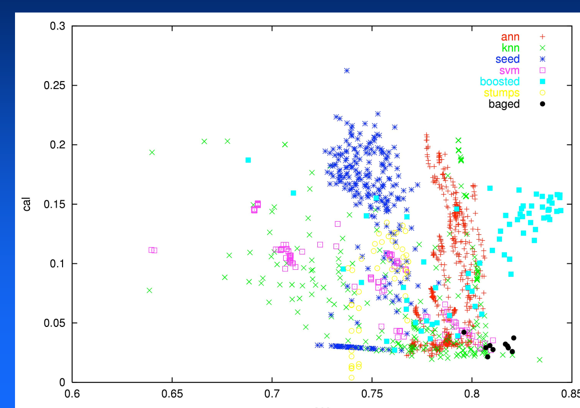
RMS Loss for Simple 2-Param Model



Loss on Six Metrics for 2-Param Model



COVTYPE: Calibration vs. Accuracy



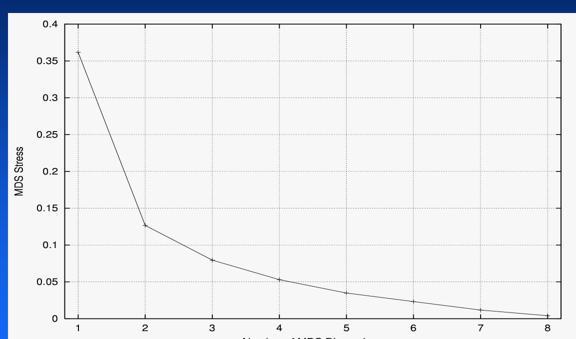
Multi Dimensional Scaling

	M1	M2	M3	M4	M5	M6	M7	. . .	M22,000
ACC	-	-	-	-	-	-	-	-	-
FSC	-	-	-	-	-	-	-	-	-
LFT	-	-	-	-	-	-	-	-	-
AUC	-	-	-	-	-	-	-	-	-
APR	-	-	-	-	-	-	-	-	-
BEP	-	-	-	-	-	-	-	-	-
RMS	-	-	-	-	-	-	-	-	-
MXE	-	-	-	-	-	-	-	-	-
CAL	-	-	-	-	-	-	-	-	-
SAR	-	-	-	-	-	-	-	-	-

Scaling, Ranking, and Normalizing

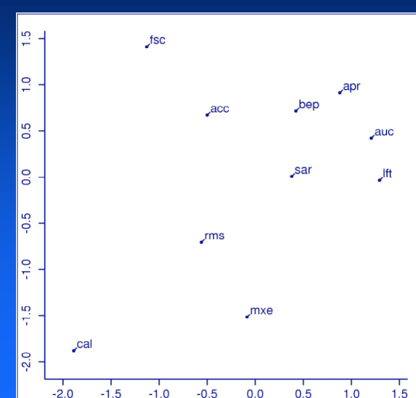
- Problem:
  - some metrics, 1.00 is best (e.g. ACC)
  - some metrics, 0.00 is best (e.g. RMS)
  - some metrics, baseline is 0.50 (e.g. AUC)
  - some problems/metrics, 0.60 is excellent performance
  - some problems/metrics, 0.99 is poor performance
- Solution 1: Normalized Scores:
  - baseline performance => 0.00
  - best observed performance => 1.00 (proxy for Bayes optimal)
  - puts all metrics on equal footing
- Solution 2: Scale by Standard Deviation
- Solution 3: Rank Correlation

Multi Dimensional Scaling



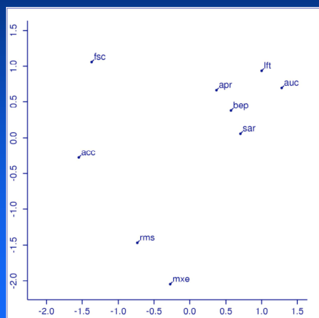
- The 10 metrics span a 2-5 dimension subspace

2-D Multi-Dimensional Scaling

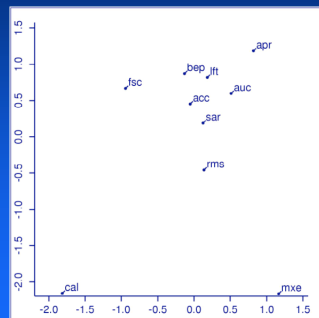


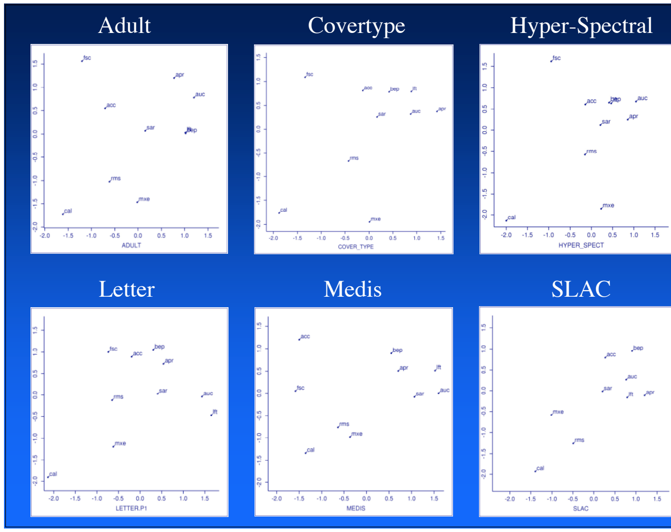
2-D Multi-Dimensional Scaling

Normalized Scores Scaling



Rank-Correlation Distance





Correlation Analysis

- 2000 performances for each metric on each problem
- Correlation between all pairs of metrics
  - 10 metrics
  - 45 pairwise correlations
- Average of correlations over 7 test problems

- Standard correlation
- Rank correlation

- Present rank correlation here

Rank Correlations

Metric	ACC	FSC	LFT	AUC	APR	BEP	RMS	MXE	CAL	SAR	Mean
ACC	1.00	0.87	0.85	0.88	0.89	0.93	0.87	0.75	0.56	0.92	0.852
FSC	0.87	1.00	0.77	0.81	0.82	0.87	0.79	0.69	0.50	0.84	0.796
LFT	0.85	0.77	1.00	0.96	0.91	0.89	0.82	0.73	0.47	0.92	0.832
AUC	0.88	0.81	0.96	1.00	0.95	0.92	0.85	0.77	0.51	0.96	0.861
APR	0.89	0.82	0.91	0.95	1.00	0.92	0.86	0.75	0.50	0.93	0.853
BEP	0.93	0.87	0.89	0.92	0.92	1.00	0.87	0.75	0.52	0.93	0.860
RMS	0.87	0.79	0.82	0.85	0.86	0.87	1.00	0.92	0.79	0.95	0.872
MXE	0.75	0.69	0.73	0.77	0.75	0.75	0.92	1.00	0.81	0.86	0.803
CAL	0.56	0.50	0.47	0.51	0.50	0.52	0.79	0.81	1.00	0.65	0.631
SAR	0.92	0.84	0.92	0.96	0.93	0.93	0.95	0.86	0.65	1.00	0.896

- Correlation analysis consistent with MDS analysis
- Ordering metrics have high correlations to each other
- ACC, AUC, RMS have best correlations of metrics in each metric class
- RMS has good correlation to other metrics
- SAR has best correlation to other metrics

Summary

- **Predicting Probabilities:**
  - Neural Nets, Bagged Trees, Random Forests best models ***overall*** right out of box
  - Calibration with Platt Scaling or Isotonic Regression yields better probabilities for Boosting, SVMs, Random Forests, Decision Trees, and Naïve Bayes
  - Where sigmoid is appropriate, Platt Scaling is more effective with little data
  - Isotonic Regression more powerful, use when data is plentiful
- **Empirical Comparison:**
  - Calibrated Boosted Trees and Random Forests yield best performance overall
  - Even after calibration, no one learning method does it all
  - Best method depends on problem, metric, and train set size
  - Picking best model yields much better performance than any one method
- **Ensemble Selection:**
  - ***Carefully*** selected ensemble of models yields further improvements
  - Can optimize ensemble to ***any*** performance metric
- **Performance Metrics:**
  - 9 metrics span 2-4 Dim subspace
  - Ordering Metrics Tightly Cluster: AUC ~ APR ~ BEP
  - RMS ~ MXE, but RMS more centrally located. RMS is king!

Thank You!