Ensemble Methods

$Attrs$	$Cl.$
$AttrVec_1$	a
$AttrVec_2$	b
$AttrVec_3$	b
$AttrVec_4$	c
$\vdots$	$\vdots$
$AttrVec_n$	a

(a) original training set

$Classifier_i$		
a	b	c
$P_{i,a1} = 0.90$	$P_{i,b1} = 0.05$	$P_{i,c1} = 0.05$
$P_{i,a2} = 0.15$	$P_{i,b2} = 0.70$	$P_{i,c2} = 0.15$
$P_{i,a3} = 0.10$	$P_{i,b3} = 0.80$	$P_{i,c3} = 0.10$
$P_{i,a4} = 0.20$	$P_{i,b4} = 0.20$	$P_{i,c4} = 0.60$
$\vdots$	$\vdots$	$\vdots$
$P_{i,an} = 0.80$	$P_{i,bn} = 0.10$	$P_{i,cn} = 0.10$

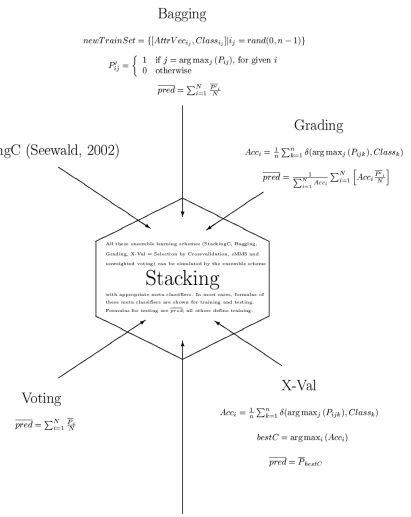
(b) sample class probability distribution

$Classifier_1$			$Classifier_2$				$Classifier_N$			$class$
a	b	c	a	b	c		a	b	c	$= a?$
$P_{1,a1}$	$P_{1,b1}$	$P_{1,c1}$	$P_{2,a1}$	$P_{2,b1}$	$P_{2,c1}$	$\dots$	$P_{N,a1}$	$P_{N,b1}$	$P_{N,c1}$	1
$P_{1,a2}$	$P_{1,b2}$	$P_{1,c2}$	$P_{2,a2}$	$P_{2,b2}$	$P_{2,c2}$	$\dots$	$P_{N,a2}$	$P_{N,b2}$	$P_{N,c2}$	0
$P_{1,a3}$	$P_{1,b3}$	$P_{1,c3}$	$P_{2,a3}$	$P_{2,b3}$	$P_{2,c3}$	$\dots$	$P_{N,a3}$	$P_{N,b3}$	$P_{N,c3}$	0
$P_{1,a4}$	$P_{1,b4}$	$P_{1,c4}$	$P_{2,a4}$	$P_{2,b4}$	$P_{2,c4}$	$\dots$	$P_{N,a4}$	$P_{N,b4}$	$P_{N,c4}$	0
	$\vdots$			$\vdots$				$\vdots$		$\vdots$
$P_{1,an}$	$P_{1,bn}$	$P_{1,cn}$	$P_{2,an}$	$P_{2,bn}$	$P_{2,cn}$	$\dots$	$P_{N,an}$	$P_{N,bn}$	$P_{N,cn}$	1

(c) meta training set for class a, Stacking with MLR

$Classifier_1$	$Classifier_2$		$Classifier_N$	$class$
a	a		a	$= a?$
$P_{1,a1}$	$P_{2,a1}$	$\dots$	$P_{N,a1}$	1
$P_{1,a2}$	$P_{2,a2}$	$\dots$	$P_{N,a2}$	0
$P_{1,a3}$	$P_{2,a3}$	$\dots$	$P_{N,a3}$	0
$P_{1,a4}$	$P_{2,a4}$	$\dots$	$P_{N,a4}$	0
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$P_{1,an}$	$P_{2,an}$	$\dots$	$P_{N,an}$	1

(d) meta training set for class a, StackingC with MLR



Univ.-Lektor Dr.techn. Alexander K. Seewald
Österreichisches Forschungsinstitut
für Artificial Intelligence

alex@seewald.at / alex.seewald.at

Ensemble Methods

Idea: Reduce bias/variance of learning systems by combining a set of *different* learning systems

Difference can be created in multiple ways:

- Using the same learning system with different training data sets (Bagging, Boosting, MetaCost)
- Using different learning systems (Voting), additionally also learning how to combine them (Stacking(C))
- Using different parameter settings for each learner

...

alex@seewald.at / alex.seewald.at

Ensemble Methods (2)

Some classifiers naturally output class probabilities (e.g. Naïve Bayes), or can be modified to do so (Decision Trees, SVMs, ..) For these, averaging over class probabilities instead of predictions generally gives better results even for small *M* (number of combined models).

In Theory: Ensemble methods trade off predictive accuracy with understandability, yielding complex and less comprehensible models which perform better.

In Practice: For datasets with small error rates, almost all examples are classified correctly by all learners, thereby limiting the gains from *any* combination approach.

alex@seewald.at / alex.seewald.at

Selection by Crossvalidation

Common procedure in ML & DM: Given a new dataset,

- Run a set of learning systems on the dataset, and determine error estimate via cross-validation.
- Choose algorithm with lowest CV error.

⇒**Ensemble Method w/o combination (=Selection by CV)**

An alternative approach

- Study given task and decide on requirements (e.g. understandability, performance, speed...)
- Run only those set of algorithms which can reasonably be expected to fulfil all the requirements.
- If understandability is not an issue, Ensemble Methods can be a better way to make full use of experimental runs.

alex@seewald.at / alex.seewald.at

Bagging

Inspired by *0.632 Bootstrap* (*B*ootstrap *agg*regation (*ing*))

- Create a set of *M* *0.632*-bootstrap samples from the original training set by sampling with replacement. Train one classifier on each bootstrap sample and average over their class predictions.

- Mainly a variance-reducing technique. Not useful for high bias, low variance learning methods (Linear methods, including SVMs with linear and nonlinear kernels). Commonly used with Decision Trees or Stumps (=Decision Trees with a single level, related to *OneR*)

alex@seewald.at / alex.seewald.at

Boosting

AdaBoost.M1:

- Apply the same learner sequentially to reweighted training sets, given more weight to examples which have been misclassified previously.
- Final classification is a weighted vote of the component learners.
- Improves dramatically on the performance of even weak base learners; reduces both bias *and* variance of base learners.

[Breiman,96]: AdaBoost + C4.5 = best off-the-shelf classifier in the world

1. Initialize the example weights w_i=1/N for i=1,2,..,N
2. For m=1 to M:
   (a) Fit a classifier $f_m(\boldsymbol{x})$ to training data weighted with w_i
   (b) Compute weighted error $Err_m=\Sigma w_i\, I(y_i \neq \text{sign}(f_m(\boldsymbol{x_i})))/\Sigma w_i$
   (c) Compute $a_m=\log((1- Err_m)/Err_m)$
   (d) Set new w_i to $w_i\,(\exp(a_m(I(y_i \neq \text{sign}(f_m(\boldsymbol{x_i})))))$, i=1,2,..,N
3. Output final $f(\boldsymbol{x}) = \Sigma a_m f_m(\boldsymbol{x})$

Equivalent to *Forward Stagewise Additive Modeling* from Statistics.

alex@seewald.at / alex.seewald.at

Voting

Unweighted Voting

- Choose a set of base learners to combine (e.g. SVM poly., NB, C4.5, RIPPER, IB1)
- Train each base learner on the *same* training data.
- For new, unseen example: Query each base learner, and average over their predictions / class probabilities.

- Very simple; almost no additional computational cost
- Mainly a bias-reducing technique: if the bias of the combined learning system is sufficiently different, the overall bias of the combined system should be smaller.

alex@seewald.at / alex.seewald.at

Grading

Grading: Learning to predict errors of multiple learning systems via another learning system, and only vote those not predicted to err.

- Compute cross-validation estimate of the prediction for each example and base learner. Learn a set of associated error learner which are trained to predict the CV errors of the base learners based on the original input attributes.
- Vote all learning systems, weighted by one minus the predicted probability of an error. Probability of error is estimated by the associated error learner.

  Shown to be qualitatively equivalent to *accuracy-weighted Voting* (where accuracy is estimated via CV) for the best-performing error learning system, so most of the above is not really necessary....

alex@seewald.at / alex.seewald.at

StackingC

StackingC: From *unweighted Voting* to *weighted Voting*

- Compute cross-validation estimate of the prediction for each example and base learner. Retrain all learners on full training data afterwards.
- The predicted class probabilites from the internal CV are then used with a regression learner (e.g. Linear Regression) to find the best weights to give to each classifier-class combination. For each class *StackingC* thus computes a weighted vote of all classifiers' predicted probabilities for that class; and chooses the class with highest probability.
- As *Grading*, *StackingC* is at least ten times slower (CV!) than unweighted Voting.

alex@seewald.at / alex.seewald.at

Stacking

Similar to *StackingC*, but can use any learner to learn the final prediction from the predictions or class probabilities of the base learners.

- Slower than *StackingC*, since the training sets are larger (for base learners which return class probabilities) and most learners are more complex than Linear Regression.
- Performance slightly worse than *StackingC*.
- Very general approach. Can simulate each shown Ensemble Method (except Boosting)

Attrs	*Cl.*
$AttrVec_1$	a
$AttrVec_2$	b
$AttrVec_3$	b
$AttrVec_4$	c
⋮	⋮
$AttrVec_n$	a

(a) original training set

a	b	c
0.90	0.05	0.05
0.15	*0.70*	0.15
0.10	*0.80*	0.10
0.20	0.20	*0.60*
⋮	⋮	⋮
0.80	0.10	0.10

(b) sample class probability distribution

$Classifier_1$			$Classifier_2$				$Classifier_N$			
a	b	c	a	b	c		a	b	c	*class*
$P_{1,a1}$	$P_{1,b1}$	$P_{1,c1}$	$P_{2,a1}$	$P_{2,b1}$	$P_{2,c1}$	...	$P_{N,a1}$	$P_{N,b1}$	$P_{N,c1}$	a
$P_{1,a2}$	$P_{1,b2}$	$P_{1,c2}$	$P_{2,a2}$	$P_{2,b2}$	$P_{2,c2}$	...	$P_{N,a2}$	$P_{N,b2}$	$P_{N,c2}$	b
$P_{1,a3}$	$P_{1,b3}$	$P_{1,c3}$	$P_{2,a3}$	$P_{2,b3}$	$P_{2,c3}$	...	$P_{N,a3}$	$P_{N,b3}$	$P_{N,c3}$	b
$P_{1,a4}$	$P_{1,b4}$	$P_{1,c4}$	$P_{2,a4}$	$P_{2,b4}$	$P_{2,c4}$	...	$P_{N,a4}$	$P_{N,b4}$	$P_{N,c4}$	c
	⋮			⋮				⋮		⋮
$P_{1,an}$	$P_{1,bn}$	$P_{1,cn}$	$P_{2,an}$	$P_{2,bn}$	$P_{2,cn}$	...	$P_{N,an}$	$P_{N,bn}$	$P_{N,cn}$	a

(c) training set for Stacking's meta classifier

alex@seewald.at / alex.seewald.at

Stacking vs. StackingC

Stacking directly predicts the true class from the predictions / class probabilities of the base learners; can use *any* learner to predict true class. Can simulate all shown ensemble schemes (except Boosting) – at additional computational cost.

StackingC predicts each class separately. Base learners need to output class probabilities. Learns only from the class probabilities relevant to each class instead of all of them! This focusses the learning process, is faster and gives slightly better results.

$Attrs$	$Cl.$
$AttrVec_1$	a
$AttrVec_2$	b
$AttrVec_3$	b
$AttrVec_4$	c
$\vdots$	$\vdots$
$AttrVec_n$	a

(a) original training set

$Classifier_i$		
a	b	c
$P_{i,a1} = 0.90$	$P_{i,b1} = 0.05$	$P_{i,c1} = 0.05$
$P_{i,a2} = 0.15$	$P_{i,b2} = 0.70$	$P_{i,c2} = 0.15$
$P_{i,a3} = 0.10$	$P_{i,b3} = 0.80$	$P_{i,c3} = 0.10$
$P_{i,a4} = 0.20$	$P_{i,b4} = 0.20$	$P_{i,c4} = 0.60$
$\vdots$	$\vdots$	$\vdots$
$P_{i,an} = 0.80$	$P_{i,bn} = 0.10$	$P_{i,cn} = 0.10$

(b) sample class probability distribution

$Classifier_1$			$Classifier_2$				$Classifier_N$			$class$
a	b	c	a	b	c		a	b	c	$= a?$
$P_{1,a1}$	$P_{1,b1}$	$P_{1,c1}$	$P_{2,a1}$	$P_{2,b1}$	$P_{2,c1}$	$\dots$	$P_{N,a1}$	$P_{N,b1}$	$P_{N,c1}$	1
$P_{1,a2}$	$P_{1,b2}$	$P_{1,c2}$	$P_{2,a2}$	$P_{2,b2}$	$P_{2,c2}$	$\dots$	$P_{N,a2}$	$P_{N,b2}$	$P_{N,c2}$	0
$P_{1,a3}$	$P_{1,b3}$	$P_{1,c3}$	$P_{2,a3}$	$P_{2,b3}$	$P_{2,c3}$	$\dots$	$P_{N,a3}$	$P_{N,b3}$	$P_{N,c3}$	0
$P_{1,a4}$	$P_{1,b4}$	$P_{1,c4}$	$P_{2,a4}$	$P_{2,b4}$	$P_{2,c4}$	$\dots$	$P_{N,a4}$	$P_{N,b4}$	$P_{N,c4}$	0
	$\vdots$			$\vdots$				$\vdots$		$\vdots$
$P_{1,an}$	$P_{1,bn}$	$P_{1,cn}$	$P_{2,an}$	$P_{2,bn}$	$P_{2,cn}$	$\dots$	$P_{N,an}$	$P_{N,bn}$	$P_{N,cn}$	1

(c) meta training set for class a, **Stacking** with MLR

$Classifier_1$	$Classifier_2$		$Classifier_N$	$class$
a	a		a	$= a?$
$P_{1,a1}$	$P_{2,a1}$	$\dots$	$P_{N,a1}$	1
$P_{1,a2}$	$P_{2,a2}$	$\dots$	$P_{N,a2}$	0
$P_{1,a3}$	$P_{2,a3}$	$\dots$	$P_{N,a3}$	0
$P_{1,a4}$	$P_{2,a4}$	$\dots$	$P_{N,a4}$	0
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$P_{1,an}$	$P_{2,an}$	$\dots$	$P_{N,an}$	1

(d) meta training set for class a, **StackingC** with MLR

Figure 5.1: Illustration of **Stacking** and **StackingC** on a dataset with three classes (a, b and c), n examples and N base classifiers. $P_{i,jk}$ refers to the probability given by base classifier i for class j on example number k

alex@seewald.at / alex.seewald.at

Other Ensemble Methods

- Meta-Decision Trees: DT predicts best model to use
- Cascaded Generalization: Stacking done sequentially
- Bagging+Voting, Boosting+Voting, Bagging+Boosting...
- Bagging/Boosting over attributes
- Random Decision Trees (=Bagging over attributes & examples, and learning an enormous number of DTs)
- Mixture Models from Statistical Theory

...

Computationally very costly approaches versus small improvements in error rate

= *The Law of Diminishing Returns*

alex@seewald.at / alex.seewald.at

Ensemble Methods on USPS Digits

Test Set Error: Error of model on *independent* test data

(i.e. not used for training)

ZeroR (Baseline)	82.11%
OneR	68.56%
Naïve Bayes	28.70%
RIPPER (Rule Learning)	16.64%
C4.5 (Decision Tree L.)	15.00%
Linear Regression	13.05%
Logistic Regression	10.91%
SVM w/ linear kernel	7.08%
IB1 (Instance-Based L.)	5.63%
SVM w/ poly. kernel	4.29%

Bagging C4.5:	10.76%
Vote:	6.73%
Boosting C4.5:	6.58%
Stacking-MJ48:	4.48%

= **Selection-By-CV**

alex@seewald.at / alex.seewald.at

Non-equal class distributions

In the US Postal Digit Dataset, all digits are equally represented (~10%). This is not always the case...

- Predicting relevant documents for SwissProt/PRINTS annotation: <1-5% of documents are relevant.
- Recognition of Species from MEDLINE: >7000 species, Top 20 most frequent account for 45% of examples.
- Not all errors have equal cost. E.g. when predicting susceptibility to cancer, we would like to err on the side of caution - i.e. the error of predicting a susceptibility if there is none is less grave than vice versa.

General solution: Cost-sensitive learning / classification

alex@seewald.at / alex.seewald.at

Cost-Sensitive Learning

Take the contingency table / confusion matrix and assign potentially different costs for each entry (without loss of generality, we can assume the cost of predicting the correct class is zero): CxC Matrix **Cost** $\{c_{ij}\}$. Until now, all errors had cost 1: **Zero-One-Loss**, equivalent to a cost matrix where $c_{ij}=1$ for $i \neq j$ and $c_{ii}=0$. Instead of maximizing accuracy (=sum of diagonal elements), we now aim to minimize total cost $TC=\Sigma c_{ij}e_{ij}$.

Predicted class

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	
355	0	2	0	0	0	0	1	0	1	*0*
0	**255**	0	0	6	0	2	1	0	0	*1*
6	1	**183**	2	1	0	0	2	3	0	*2*
3	0	2	**154**	0	5	0	0	0	2	*3*
0	3	1	0	**182**	1	2	2	1	8	*4*
2	1	2	4	0	**145**	2	0	3	1	*5*
0	0	1	0	2	3	**164**	0	0	0	*6*
0	1	1	1	4	0	0	**139**	0	1	*7*
5	0	1	6	1	1	0	1	**148**	3	*8*
0	0	1	0	2	0	0	4	1	**169**	*9*

True class

alex@seewald.at / alex.seewald.at

Over/Undersampling

Idea: Equalize unequal distribution of classes by removing examples of majority class (=undersampling) or by duplicating examples of minority class (=oversampling). Both approaches are also called *stratification* (~ strat.CV)

- Very simple, easy to do for all learners.
- For learners which can process weighted examples, reweighting instead of resampling is also an option.
- However, only applicable to two-class learning tasks where $c_{ij}=c'_j$ (i.e. where the cost of misclassifying example is independent of predicted class)
- Undersampling loses valuable data, Oversampling makes no difference for some learners (e.g. IB1)

alex@seewald.at / alex.seewald.at

ProbThreshold

Idea: Instead of choosing class with highest probability, weight each probability with associated cost summed over all classes ($=c'_j$), and choose class which minimizes cost.

- Only applicable to learners which output usable class probability distributions (e.g. Naïve Bayes, Logistic Reg.). A weak restriction: most current learners are of this kind.
- Applicable to multi-class tasks.
- Only applicable to learning tasks where $c_{ij}=c'_j$ (i.e. where the cost of misclassifying example is independent of predicted class)

alex@seewald.at / alex.seewald.at

MetaCost

Idea: Relabel (=change true class) of training examples according to class which is predicted to have minimum cost. Estimate probabilities of class by *bootstrapping*: Repeatedly sample training set and run the classifiers, then average over class predictions / probabilities for each example. "True" class of example is estimated after each training run. Equivalent to minimizing conditional risk:

$$R(i \mid x) = \sum_{\forall j} P(j \mid x) c_{ij}$$

- Applicable to *all* learners regardless of whether they output class predictions or probabilities for all classes and for arbitrary cost matrices.
- Similar to *Bagging* with *ProbThreshold*, but Output is a single model trained on relabeled training data.

alex@seewald.at / alex.seewald.at