Tutorial on Machine Learning Reductions

John Langford

Chicago 2005 Machine Learning Summer School

May 17

Learning Problems are prediction problems

The *type* of learning problems differ.

1. Given view outside, will it rain? (binary)
2. Which commute is faster? (cost sensitive)
3. Which stock will go up? (cost sensitive, multiclass)
4. Can I make the light before it turns red? (binary, importance weighted)

Reductions turn type A learning algorithm into type B learning algorithm

1. How do I solve importance weighted binary classification with a classifier?

2. ... Cost Sensitive Classification with a classifier?

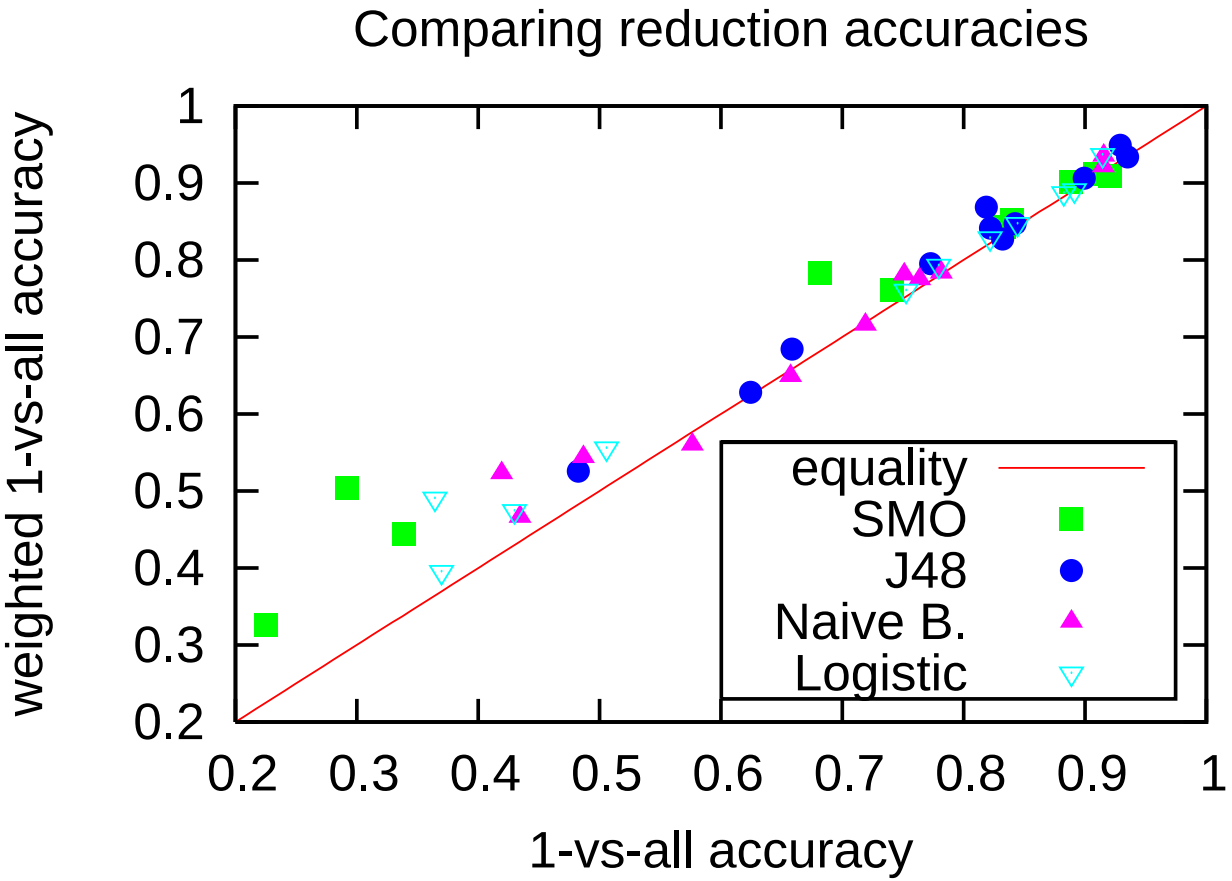
3. ... Reinforcement Learning with a classifier?

4. ... Regression with a classifier?

Why Learning Reductions?

1. It works.
2. It's modular.
3. It's reductionist.
4. It's easy.

It works (= you want to use it)



It's modular (= easy code)

Reduction needs to know nothing about oracle learning algorithm except its type.

Modularity $\Rightarrow$

1. Can reuse old learning algorithms

2. Can reuse old code

It's reductionist (= good research direction)

Reductionist = cut problems into small problems, solve small problems, and compose to solve big problem.

Some other reductionist things:

1. The transistor for computations

2. Rendering triangles for rendering scenes

3. Much of science

It's easy (= you can use it too)

The reductions method to solving learning problems:

1. Identify the type of learning problem B.
2. Find premade reduction R and oracle learning algorithm A.
3. Build a B predictor using R^A + data.

Classification Definition

- Problem: A measure D on $X \times \{0,1\}$ where X is an arbitrary space.
- Classifier: $c : X \to \{0,1\}$ = predictor
- Given $S = (X \times \{0,1\})^*$ find classifier c with small error rate

$$e(D,c) = \Pr_{(x,y)\sim D}(c(x) \neq y)$$

Note: D unknown. Impossible in general, but maybe possible in particular. We hope S drawn IID from D, but it isn't always so.

Outline

1. Importance Weighted Classification to Classification
2. Class Probability to Classification
3. Multiclass To Classification
4. Cost Sensitive to Classification

Importance Weighted Classification

- Problem: A measure D_i on $X \times \{0,1\} \times [0,\infty)$ where X is an arbitrary space.

- Classifier: $c : X \to \{0,1\}$ = predictor

- Given $S = (X \times \{0,1\}, [0,\infty))^*$ find classifier c with small importance weighted loss

$$e_i(D_i, c) = E_{(x,y,i)\sim D_i}[iI(c(x) \neq y)]$$

The core theorem: folklore

Theorem: (Distribution shift) For all c, for all D_i, let $D(x,y,i) = \frac{iD_i(x,y,i)}{E_{(x,y,i)\sim D_i}[i]}$. Then:

$$e_i(D_i,c) = e(D,c)E_{(x,y,i)\sim D_i}[i]$$

... so minimizing D error rate = minimizing D_i importance weighted error rate. Proof:

$e_i(D_i,c) = \sum_{(x,y,i)}[iD_i(x,y,i)I(c(x) \neq y)]$

$= E_{(x,y,i)\sim D_i}[i] \sum_{(x,y,i)}[D(x,y,i)I(c(x) \neq y)]$

$= E_{(x,y,i)\sim D_i}[i] \Pr_{(x,y)\sim D}[I(c(x) \neq y)]$

$= E_{(x,y,i)\sim D_i}[i]e(D,c)$

Distribution Transform: Rejection Sampling.

1. Pick a constant c larger than any importance ($\forall i \ c > i$)

2. For each sample (x, y, i), flip a coin with bias $\frac{i}{c}$. If the result is "heads" keep it, and otherwise discard it.

Other methods do not work that well, in practice. Rejection sampling $\Rightarrow$ samples in S are IID if samples in S_i are IID.

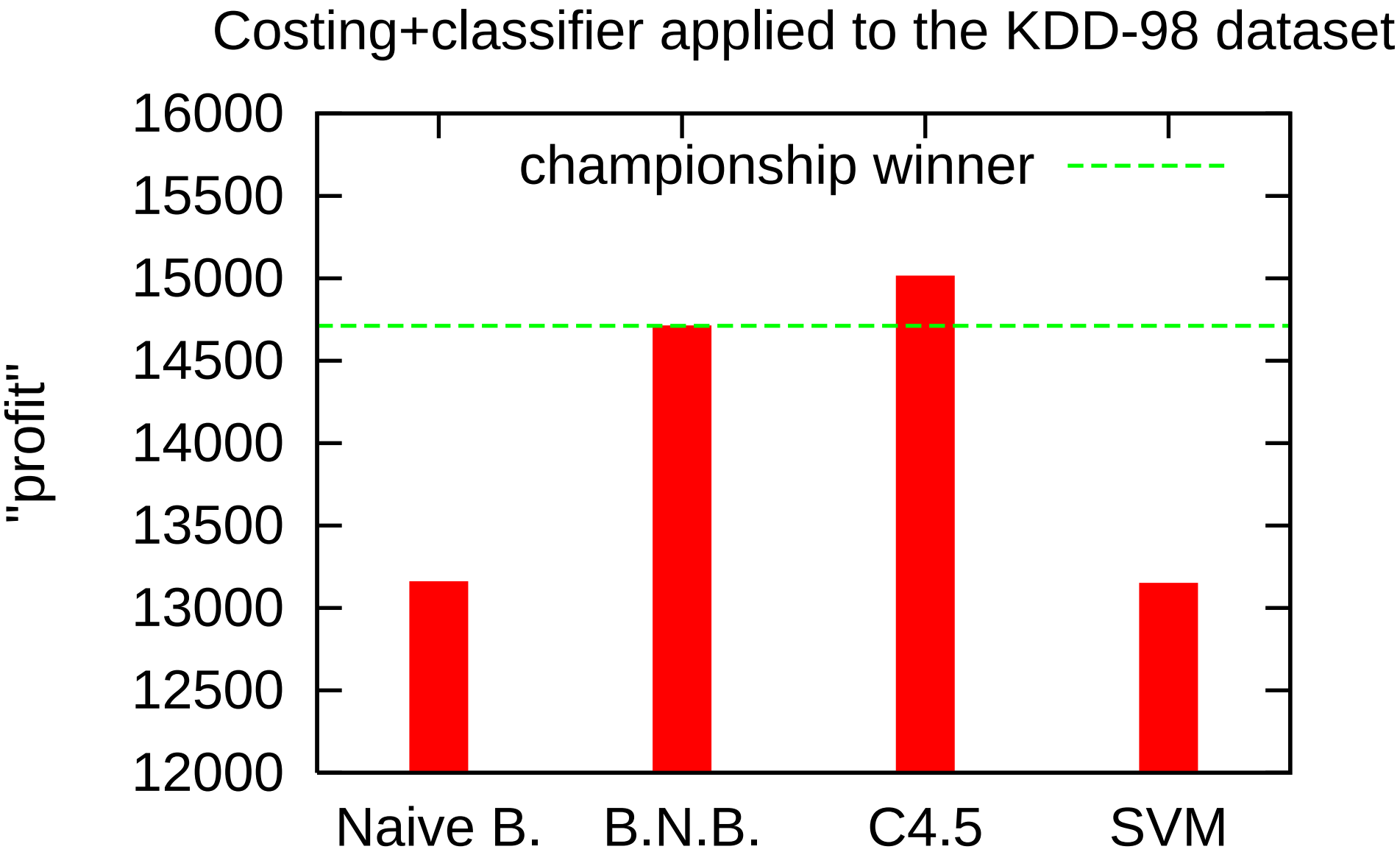
Costing(S_i,A)

1. Repeat 10 times

   (a) Rejection sample to form S from S_i.

   (b) Learn $c = A(S)$

To predict: use majority vote over learned c.


Costing+classifier applied to the KDD-98 dataset
"profit"
16000
15500
15000
14500
14000
13500
13000
12500
12000
championship winner
Naive B.
B.N.B.
C4.5
SVM

Outline

1. Importance Weighted Classification to Classification
2. Class Probability to Classification
3. Multiclass Probability To Classification
4. Cost Sensitive to Classification

Class Probability Estimation

- Problem: A measure D on $X \times \{0,1\}$ where X is an arbitrary space.

- Probabilistic Classifier: $c_p : X \to [0,1]$ = predictor

- Given $S = (X \times \{0,1\})^*$ find probabilistic classifier c_p with small squared error:

$$e_p(D, c_p) = E_{(x,y)\sim D}[(c_p(x) - y)^2]$$

Reasons for The Probability Estimation Problem

1. Doctor wants "advice" from a machine, but not a decision.

2. Distributed system requires efficient communication of beliefs.

3. Compatibility between prediction and probabilistic prediction worlds.

The Probing Method: Observations

Observation: if c is perfect, $c(x) = 1 \Rightarrow D(y = 1|x) > 0.5$

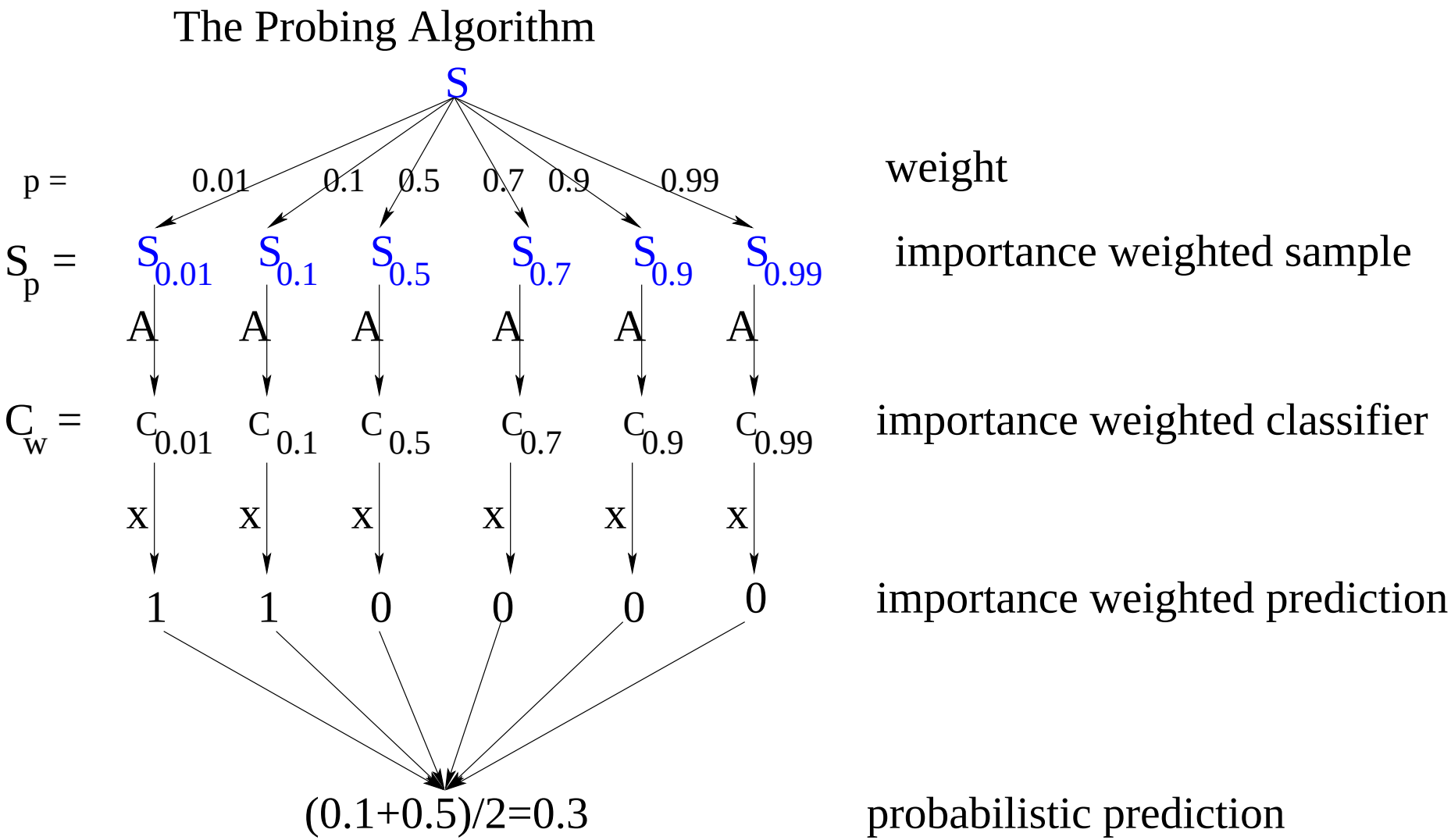
1. Pick $p \in (0, 1)$.

2. Map $(x, y) \rightarrow (x, y, |y - p|)$ (= importance weighted example)

if c perfect then, $c(x) = 1 \Rightarrow D(y = 1|x) > p$

Proof:

Prediction	Expected Importance given x
0	$(1 - D(y = 1|x)p$
1	$D(y = 1|x)(1 - p)$

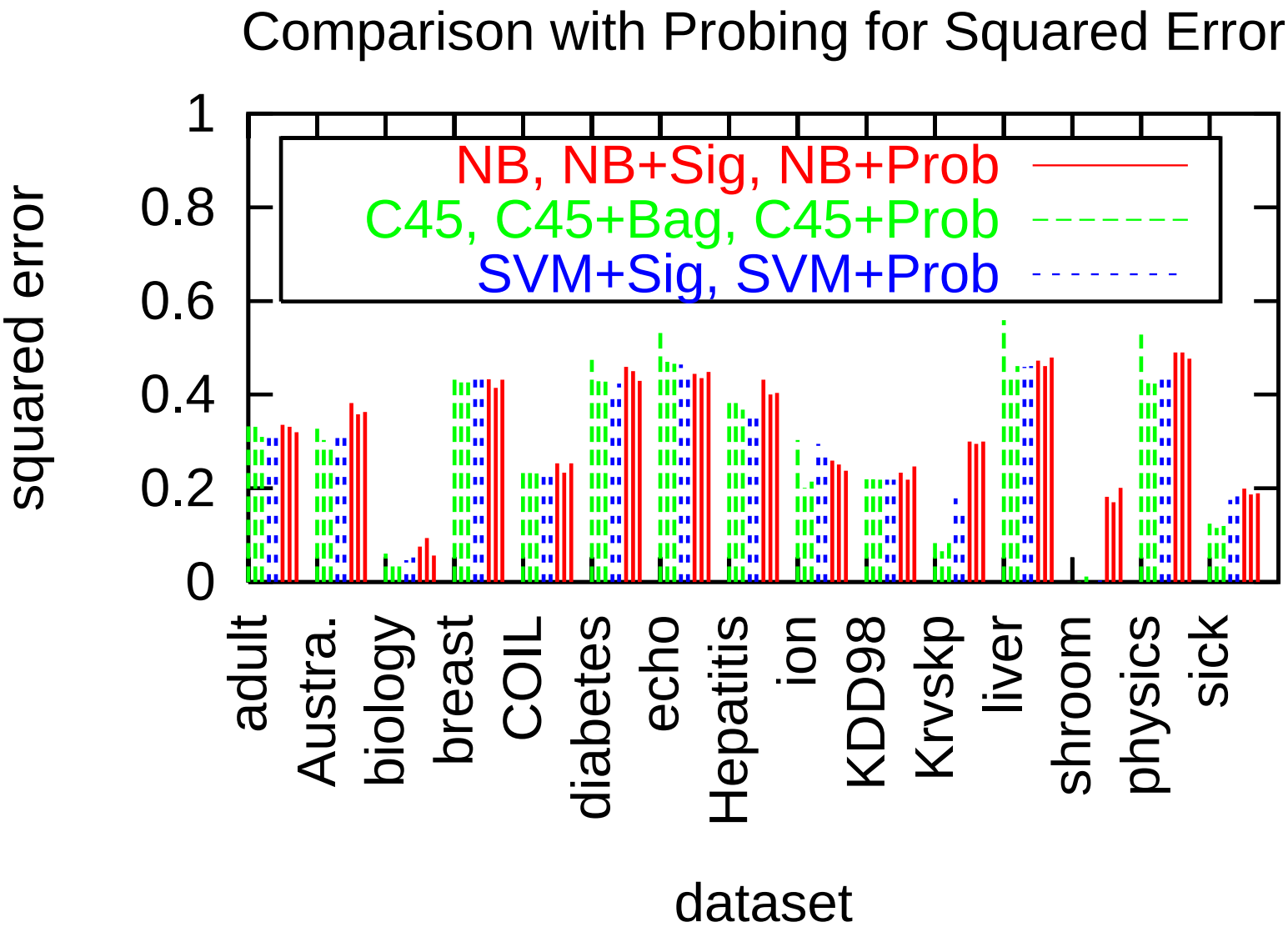

The Probing Algorithm
S
p =
0.01
0.1
0.5
0.7
0.9
0.99
weight
S_p =
$S_{0.01}$ $S_{0.1}$ $S_{0.5}$ $S_{0.7}$ $S_{0.9}$ $S_{0.99}$
importance weighted sample
A A A A A A
C_w =
$C_{0.01}$ $C_{0.1}$ $C_{0.5}$ $C_{0.7}$ $C_{0.9}$ $C_{0.99}$
importance weighted classifier
x x x x x x
1 1 0 0 0 0
importance weighted prediction
(0.1+0.5)/2=0.3
probabilistic prediction

The Probing Method: Details

1. How do you make classifier take weights? Costing Reduction

2. How do you deal with nonmonotonic predictions? Sort

3. How do you discretize on p? Uniform grid or on demand


Comparison with Probing for Squared Error
squared error
1
0.8
0.6
0.4
0.2
0
NB, NB+Sig, NB+Prob
C45, C45+Bag, C45+Prob
SVM+Sig, SVM+Prob
adult
Austra.
biology
breast
COIL
diabetes
echo
Hepatitis
ion
KDD98
Krvskp
liver
shroom
physics
sick
dataset

The one classifier trick

We learn many classifiers in parallel. They *could* be one classifier.

1. Let $S = \cup_p \{(< x, p >, y, i) : (x, y, i) \in S_p\}$
2. Let $c = \text{Costing}(S, A)$
3. Let $c_p(x) = c(x, p)$

Do you really want to do this in practice? Unknown.

Handy for theory: we can think about drawing from induced distribution on c by drawing from $(x, y) \sim D$ and $p \sim U(0, 1)$ to generate a sample from induced distribution $\text{Probing}(D)$.

Probing Theory

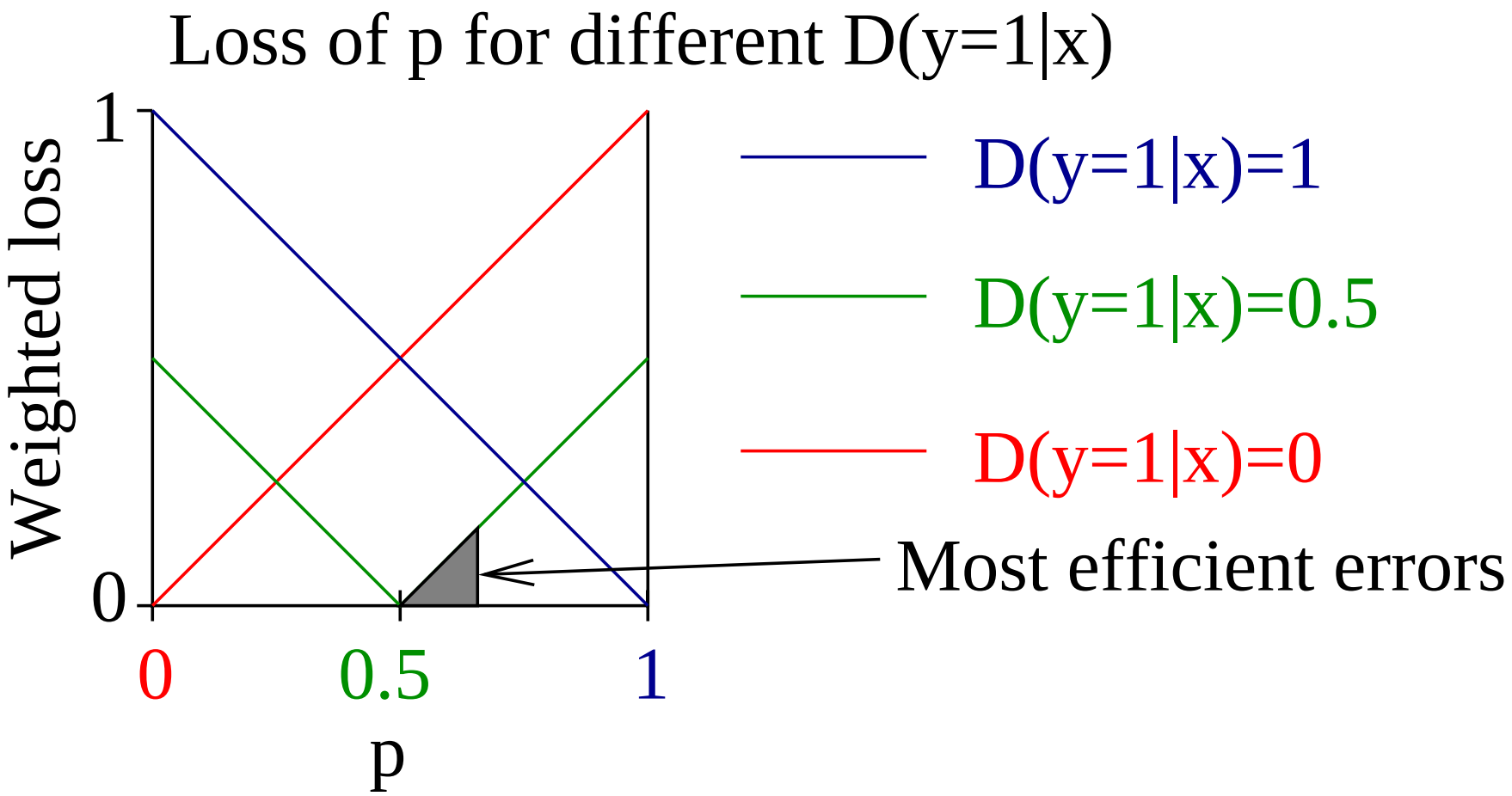
Theorem: (Probing Translation) For all $c : X \times [0,1] \to \{0,1\}$, for all D on $X \times \{0,1\}$

$$E_{x,y\sim D}(D(y|x) - \text{Probing}_c(x))^2$$

$$\leq e(\text{Probing}(D), c) - \min_{c'} e(\text{Probing}(D), c')$$

... spooky. You don't know $D(y|x)$, yet minimizing $e(\text{Probing}(D), c)$ always implies good estimates of $D(y|x)$.

The proof, pictorially



The proof, mathematically

Expected importance $= \frac{1}{2}$ so:

$e(\text{Probing}(D), c) - \min_{c'} e(\text{Probing}(D), c')$

$= 2E_{p,(x,y)\sim D}|y-p|I(c(x,p) \neq y) - \min\{(1-p)D(1|x), p(1-D(1|x)\}$

("2" comes from the distribution shift theorem)

$= 2E_{x,p}E_{y\sim D|x}|y-p|I(c(x,p) \neq y) - \min\{(1-p)D(1|x), p(1-D(1|x)\}$

For any x, p, either c predicts perfectly (difference $= 0$) or not.

If not, difference $= 2|(1-p)D(1|x) - p(1-D(1|x)| = 2|p-D(1|x)|$

$\Rightarrow$ cost of misclassification $= 2|p - D(1|x)|$.

Proof II: Properties of most efficient error inducing method

We have a budget of ϵ errors. How can probability estimation errors be maximized?

1. Only classifications $c(x, p)$ on one side of $D(1|x)$ err. (otherwise sort = cancellation of errors = wasted binary errors)

2. Errors for p closer to $D(1|x)$ are preferred over errors further from $D(1|x)$ (sort makes these equivalent, and the importance weighted loss is smaller for nearer errors)

$\Rightarrow$ deviation Δ requires at least importance weighted loss

$$\int_{D(1|x)}^{D(1|x)+\Delta} 2|D(1|x) - z|dz = \Delta^2$$

Some Caveats

1. No bound holds for cross entropy. (Can't be done without extra assumptions/constraints.)

2. No direct bound on error rate for relative ranking.

Outline

1. Importance Weighted Classification to Classification
2. Class Probability to Classification
3. Multiclass Probability To Classification
4. Cost Sensitive to Classification

Multiclass Classification

- Problem: A measure D on $X \times \{1, ..., k\}$ where X is an arbitrary space.

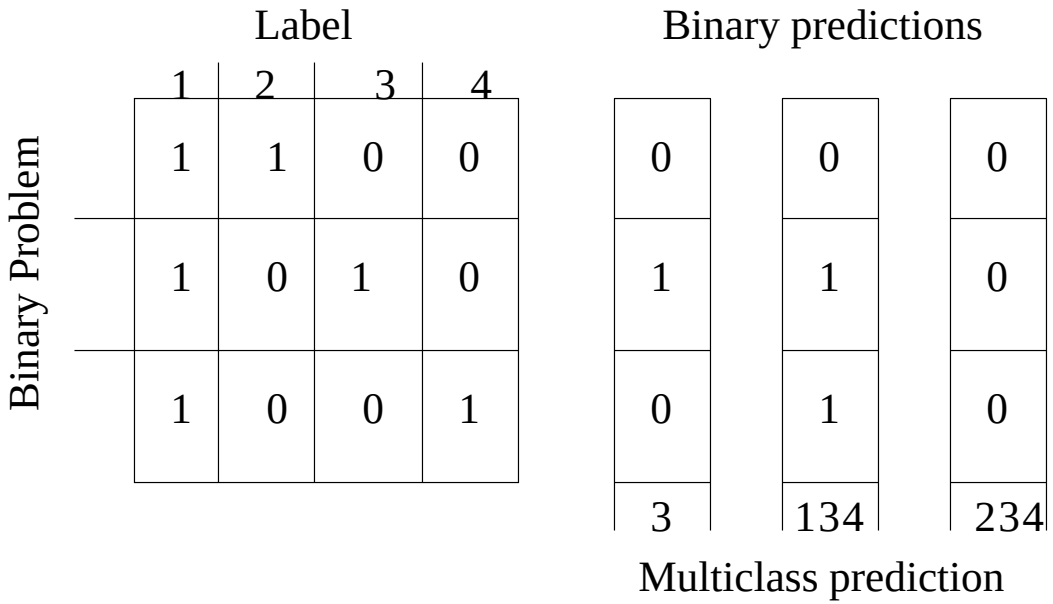
- Multiclass Classifier: $c_m : X \to \{1, ..., k\}$ = predictor

- Given $S = (X \times \{1, ..., k\})^*$ find multiclass classifier c_m with small error rate:

$$e(D, c_m) = \Pr_{(x,y)\sim D}(c_m(x) \neq y)$$

Error Correcting Output Codes

Use Binary learning algorithms $A : (X, \{0, 1\})^* \to (X \to \{0, 1\})$ to solve Multiclass classification.



Binary training data $= \{(x, I(y_m \in \text{set}) : (x, y_m) \in \text{multiclass data}\}$

Multiclass prediction $=$ closest label vector (ties $\Rightarrow$ randomize).

ECOC Analysis

Use one classifier trick $c(x, s) = c_s(x)$. Let $\mathsf{ECOC}(D_m)$= induced distribution on binary predictions.

Theorem: (ECOC transform) For all k, there exists code such that for all c, for all D_m on (x, y_m),

$$e(D_m, \mathsf{ECOC}_c) \leq 4e(\mathsf{ECOC}(D_m), c)$$

Proof: Exists code such that $< \frac{1}{4}$ binary errs $\Rightarrow$ no error

Intuition: two random bit vectors disagree in $\frac{1}{2}$ of locations so $\frac{1}{4}$ positions must be flipped to change which vector is nearest.

Exact example = rows of Hadamard matrix.

A little problem

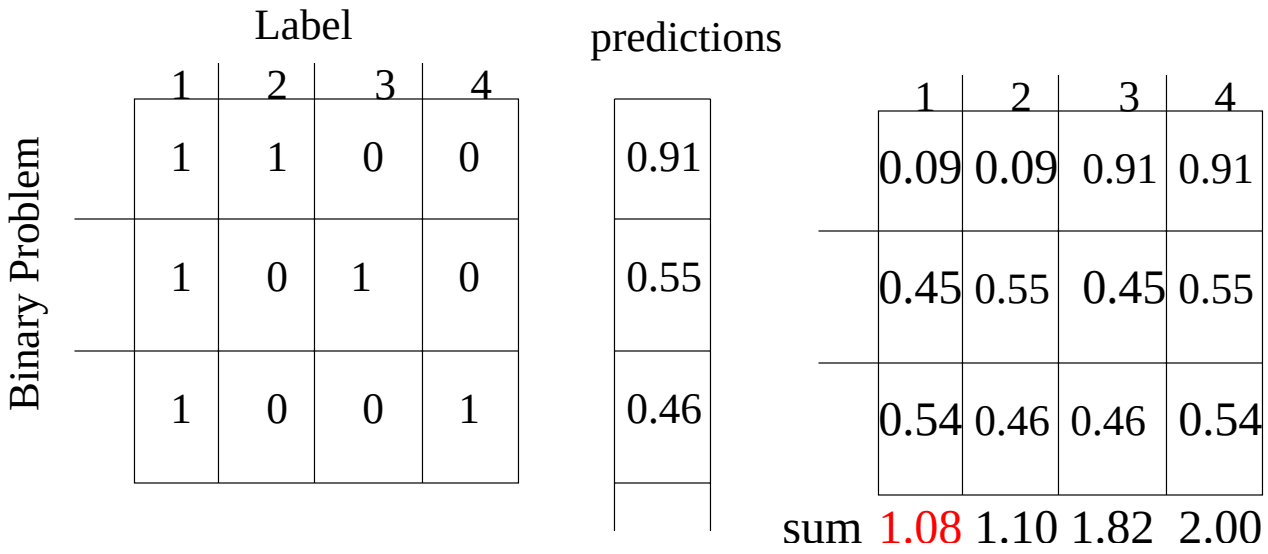
Theorem: (ECOC suboptimality) There exists a D_m such that for all c, for all codes,

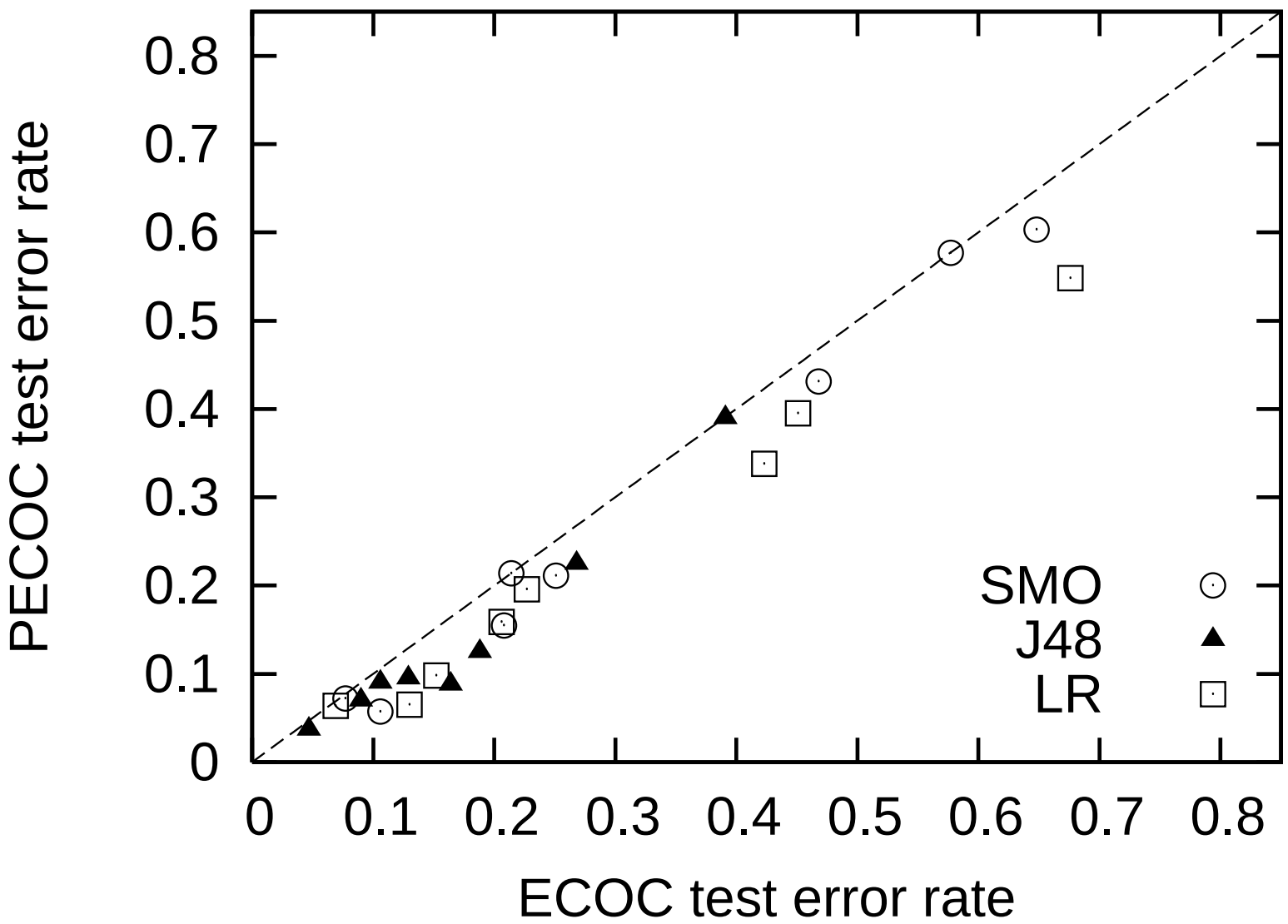
$$e(D_m, \mathsf{ECOC}_c) - \min_{c_m} e(D_m, c_m) > 0$$

Probabilistic Error Correcting Output Code: PECOC

As ECOC, except use Probing to get probabilistic predictions.

To predict: Use l_1 closest label/codeword.




PECOC test error rate
ECOC test error rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
SMO
J48
LR

PECOC Analysis

Use one classifier trick $c(x, s, p) = c_{sp}(x)$. Let PECOC(D_m)= induced distribution on binary predictions.

Theorem: (PECOC transform) For all k, there exists code such that for all c, for all D_m on (x, y_m),

$$e(D_m, \text{PECOC}_c) - \min_{c'} e(D_m, c')$$

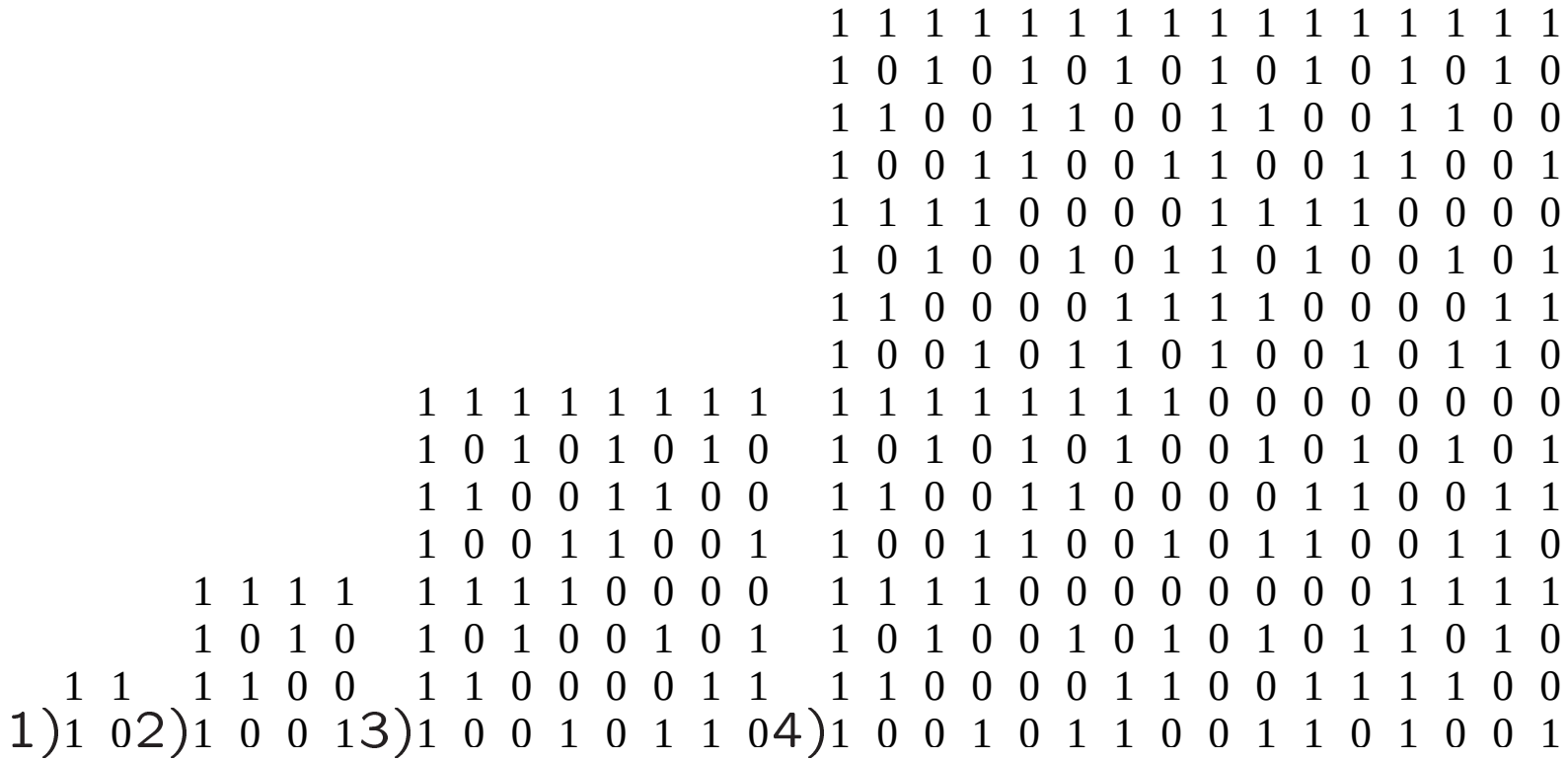
$$\leq 4\sqrt{e(\text{PECOC}(D_m), c) - \min_{c'} e(\text{PECOC}(D_m), c')}$$

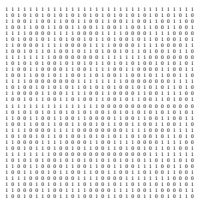
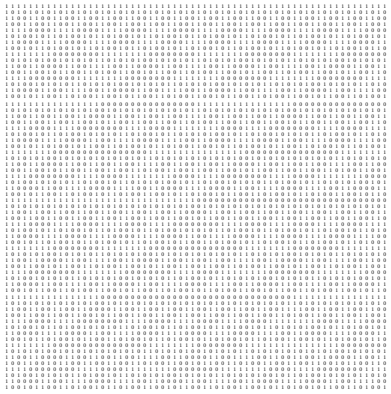
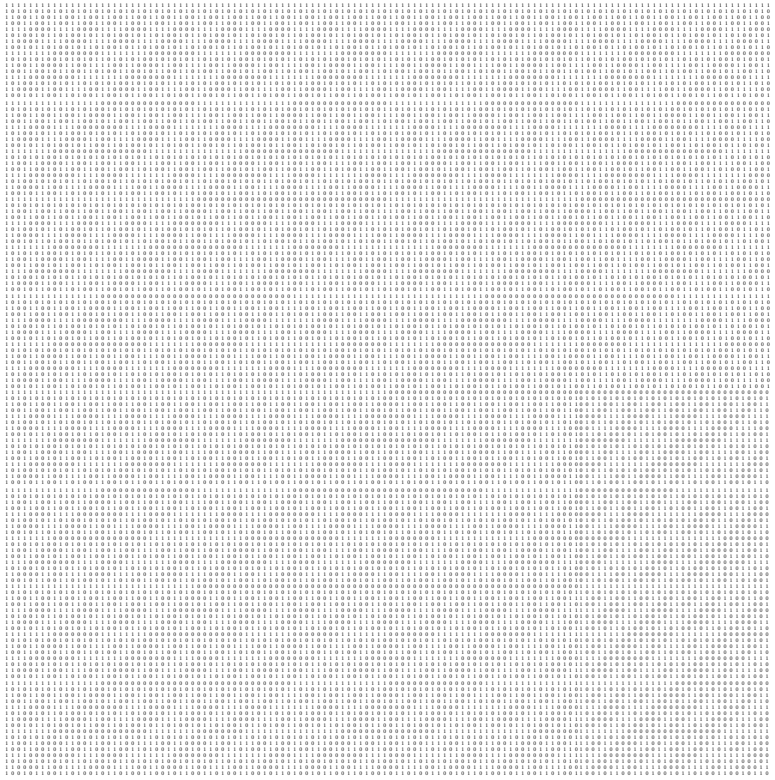
- Binary consistency $\Rightarrow$ Multiclass consistency

- Binary error rate $= 0.25$ OK when minimum error rate $= 0.25$

Proof

Pick code = subset of recursively defined Hadamard matrix.



5)  6)  7)  8)

...

Hadamard was a rug designer!

If b = number of binary problems, distance between codewords $= \frac{b}{2}$

For all labels l,

$$\sum_{b \in \text{Binary Problems}} \text{Probing}_b(\text{set containing } l|x)$$

$$= \sum_{b \in \text{Binary Problems}} \left[\sum_{l' \in \text{set containing l}} D_m(l'|x) \right]$$

(Assuming perfect classifiers)

$$= b(D_m(l|x) + \frac{1}{2} \sum_{l' \neq l} D_m(l'|x)) = b\frac{D_m(l|x) + 1}{2}$$

Proof: Analyzing errors

From Probing analysis,

$$E_{x,y\sim D}|D(1|x) - \text{Probing}_c(x)|$$

$$\leq \sqrt{e(\text{Probing}(D), c) - \min_{c'} e(\text{Probing}(D), c')}$$

Let b = number of binary problems.

Most efficient method to disturb l_1 sum by ϵb is for each call to probing to err by $|D(1|x) - \text{Probing}(x)| = \epsilon$. (since $\sqrt{x}$ is convex)

Changing l_1 sum by $b\epsilon \Rightarrow$ choosing class with error rate at most 4ϵ worse than minimum.

Bonus round.

You can also do probabilistic multiclass prediction.

$$\sum_{b \in \text{Binary Problems}} \widehat{p}_b = b \frac{D_m(l|x) + 1}{2}$$

$$\Rightarrow D_m(l|x) = \frac{2 \sum_{b \in \text{Binary Problems}} \widehat{p}_b}{b} - 1$$

... and this turns out to be stable.

Theorem: (PECOC' Translation) For all $c : X \times [0,1] \to \{0,1\}$, for all D_m on $X \times \{1, ..., k\}$

$$E_{x,y \sim D}(D_m(y|x) - \text{PECOC'}_c(x))^2$$

$$\leq 4(e(\text{PECOC}(D_m), c) - \min_{c'} e(\text{PECOC}(D_m), c'))$$

Outline

1. Importance Weighted Classification to Classification
2. Class Probability to Classification
3. Multiclass Probability To Classification
4. Cost Sensitive to Classification

Cost Sensitive Classification

- Problem: A measure D_{cs} on $X \times [0, \infty)^k$ where X is an arbitrary space.

- Multiclass Classifier: $c_m : X \to \{1, ..., k\}$ = predictor

- Given $S = (X \times [0, \infty)^k)^*$ find classifier with small expected loss

$$e_{cs}(D_{cs}, c_m) = E_{(x,\vec{\ell}) \sim D_{cs}}[\ell_{c_m(x)}]$$

Sensitive Error Correcting Output Code

SECOC = cost sensitive $\Rightarrow$ importance weighted classification reduction, in two parts. Uses a code matrix M:

Subset \ Label	1	2	3	4
	1	1	0	0
	1	0	1	0
	1	0	0	1

SECOC, the training algorithm

SECOC-Train (cost sensitive examples S, importance weighted learning algorithm A)

1. For each subset s defined by the rows of M:
   (a) For $(x, \vec{\ell}) \in S$, let $|\vec{\ell}| = \sum_y \ell_y$ and $\ell_s = \sum_{y \in s} \ell_y$.
   (b) For random t in $[0, 1]$:
   Let $c_{st} = A(\{(x,\ I(\ell_s \geq t|\vec{\ell}|), |\ell_s - |\vec{\ell}|t|) : (x, \vec{\ell}) \in S\})$.
2. return $\{\ell_{st}\}$

SECOC, the prediction algorithm

SECOC-Predict (classifiers $\{c_{st}\}$, example $x \in X$)

return $\min_y 2E_sE_t\left[I(y \in s)c_{st}(x) + I(y \notin s)(1 - c_{st}(x))\right] - 1$

SECOC Analysis

Use the one classifier trick + Costing. Let

$$S'_{st} = \{((x, s, t), y, i) : (x, y, i) \in S_{st}\}$$

Then train to learn $c = A(\cup_{st} S'_{st})$ and define $c_{st}(x) = c(x, s, t)$

Theorem: (SECOC transform) For all k, there exists code such that for all c, for all D_{cs} on $(x, \vec{\ell})$,

$$e_{cs}(D_{cs}, \mathsf{SECOC}_c) - \min_{c'} e_{cs}(D_{cs}, c')$$

$$\leq 4\sqrt{(e(\mathsf{SECOC}(D_{cs}), c) - \min_{c'} e(\mathsf{SECOC}(D_{cs}), c'))E_{(x,\vec{\ell})\sim D_{cs}}|\vec{\ell}|}$$

Proof (sketch only, much like PECOC)

1. $E_t\left[I(y \in s)c_{st}(x) + I(y \notin s)(1 - c_{st}(x))\right] = \frac{E_{\vec{\ell} \sim D|x}[\ell_s]}{E_{\vec{\ell} \sim D|x}\left[|\vec{\ell}|\right]}$ when classifiers optimal.

2. $E_s \frac{E_{\vec{\ell} \sim D|x}[\ell_s]}{E_{\vec{\ell} \sim D|x}\left[|\vec{\ell}|\right]} = \frac{\frac{E_{\vec{\ell} \sim D|x}[\ell_y]}{E_{\vec{\ell} \sim D|x}\left[|\vec{\ell}|\right]} + 1}{2}$ when classifiers optimal.

3. Optimal method for adversary to cause loss without incurring importance weighted regret = small error t's for each s.

4. Cost of erring linear in $t \Rightarrow$ average regret growth quadratic.

More Bonus Rounds

Corollary: Soft Prediction

$$\Rightarrow E_{x \sim D_{cs}} \left(\text{Predict}(c, x, y) E_{\vec{\ell}' \sim D|x} \left[|\vec{\ell}'| \right] - E_{\vec{\ell}' \sim D|x} \left[\ell'_y \right] \right)^2$$

$$\leq 4(e(\text{SECOC}(D_{cs}), c) - \min_{c'} e(\text{SECOC}(D_{cs}), c')) E_{(x, \vec{\ell}) \sim D_{cs}} |\vec{\ell}|$$

Corollary: PECOC = Embed multiclass in cost sensitive + SECOC

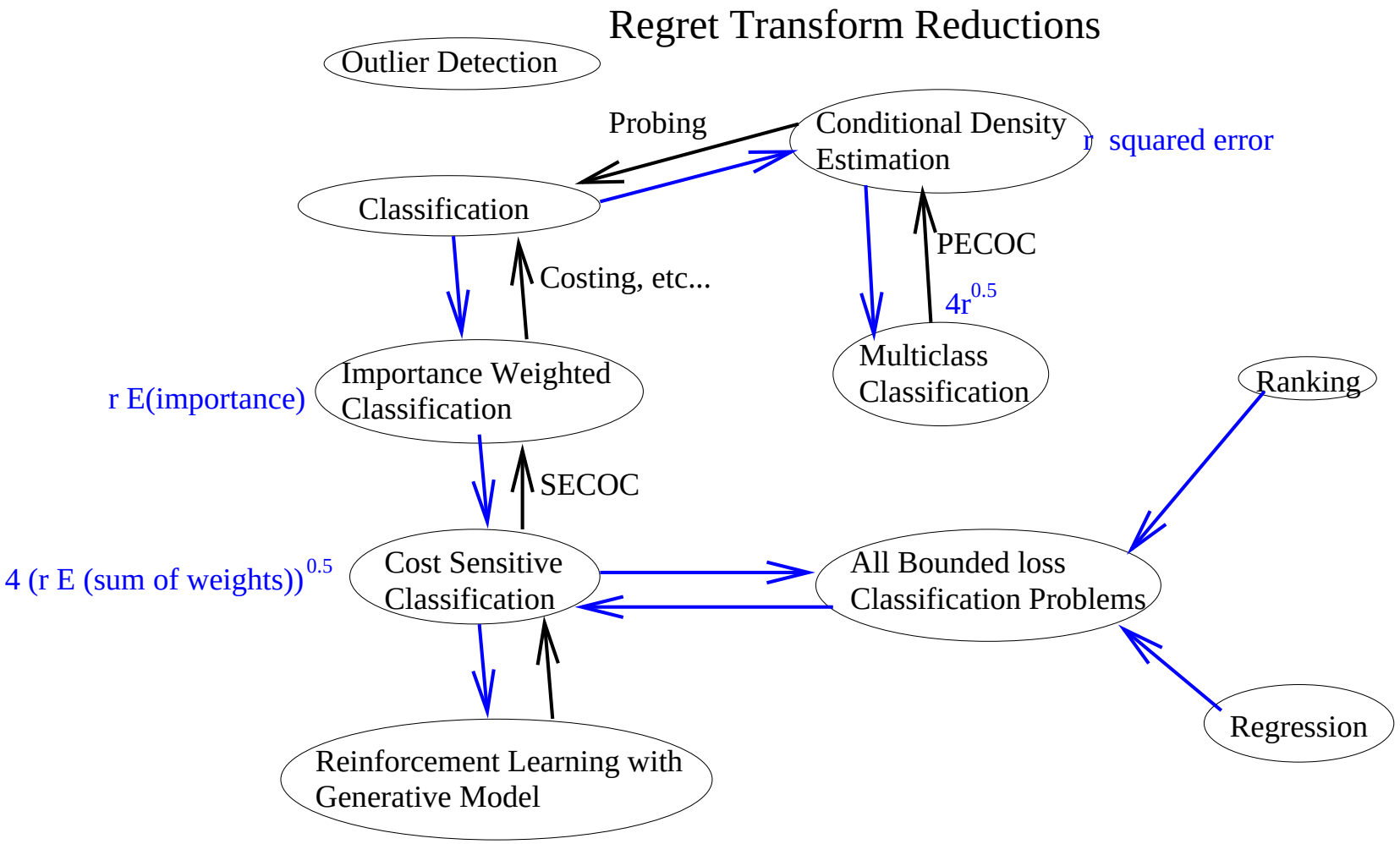
Embedding: $(x, y_m) \rightarrow (x, \forall i \;\; I(y_m \neq i))$

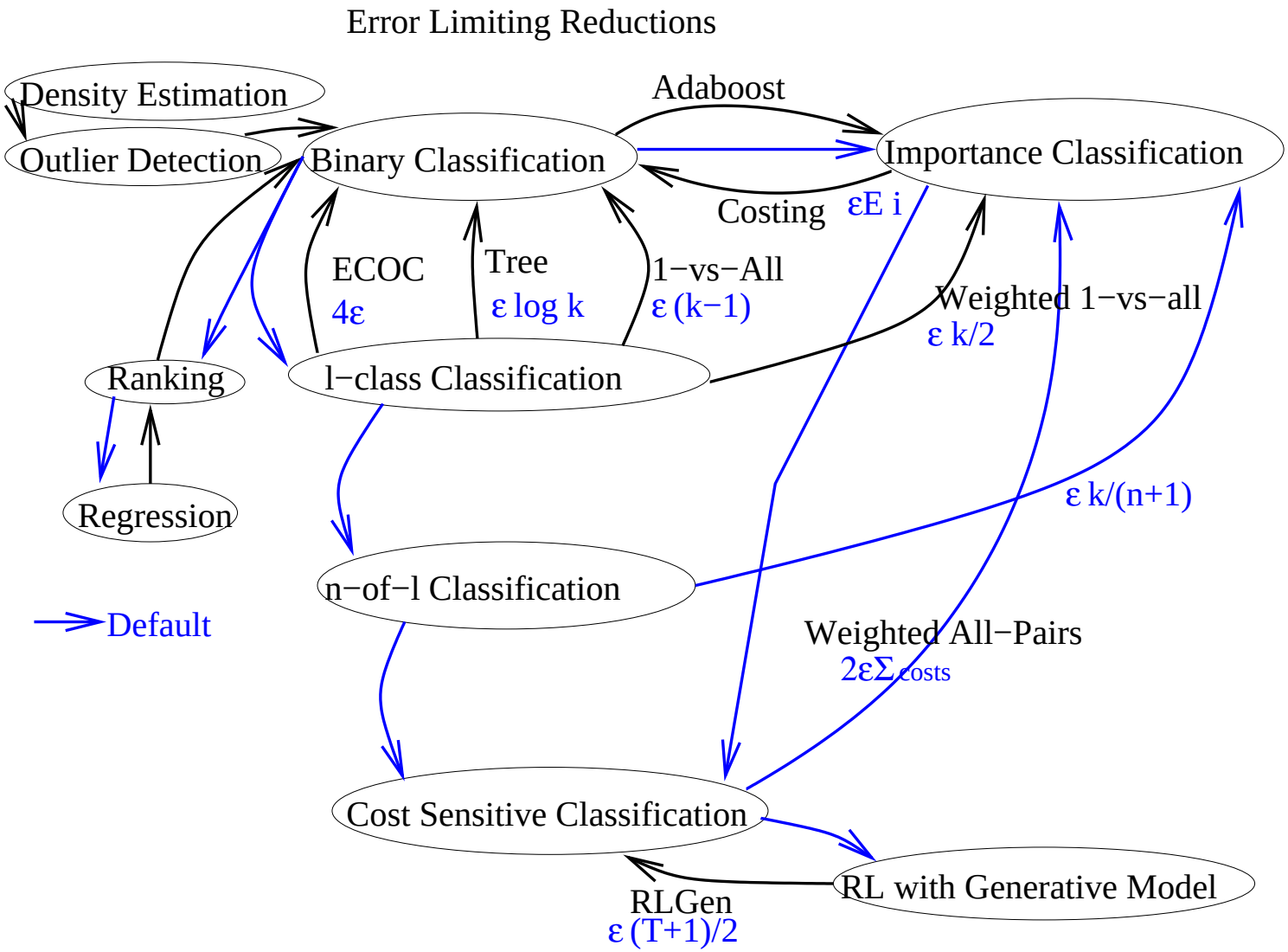
Some Things to Think About

Experiments: Not done yet. How well does this work in practice?

Which code do you use in practice?

Shannon $\Rightarrow$ a random code of size $O(\log k)$ is near optimal for above theory.


Regret Transform Reductions
Outlier Detection
Probing
Conditional Density Estimation
r squared error
Classification
PECOC
Costing, etc...
4r^0.5
Multiclass Classification
r E(importance)
Importance Weighted Classification
Ranking
SECOC
4 (r E (sum of weights))^0.5
Cost Sensitive Classification
All Bounded loss Classification Problems
Regression
Reinforcement Learning with Generative Model


Error Limiting Reductions
Density Estimation
Outlier Detection
Binary Classification
Adaboost
Importance Classification
Costing
εE i
ECOC
4ε
Tree
ε log k
1−vs−All
ε (k−1)
Weighted 1−vs−all
ε k/2
Ranking
l−class Classification
Regression
ε k/(n+1)
n−of−l Classification
Default
Weighted All−Pairs
2εΣ costs
Cost Sensitive Classification
RL with Generative Model
RLGen
ε (T+1)/2

Boosting Is Really Cool

A method for reusing a learning algorithm over and over to address its own shortcomings.

Be sure to catch Robert Schapire's tutorial.

Final Thoughts

Formal study of learning reductions is relatively new.

- $\Rightarrow$ The limits of the possible are not entirely known.
- $\Rightarrow$ You-a-fellow-researcher could easily have important insights.

Coherent codebase, coming soon.

Coherent tutorial in paper form, coming soon.