How Classifiers can be use to Solve Any Reasonable(*) Loss

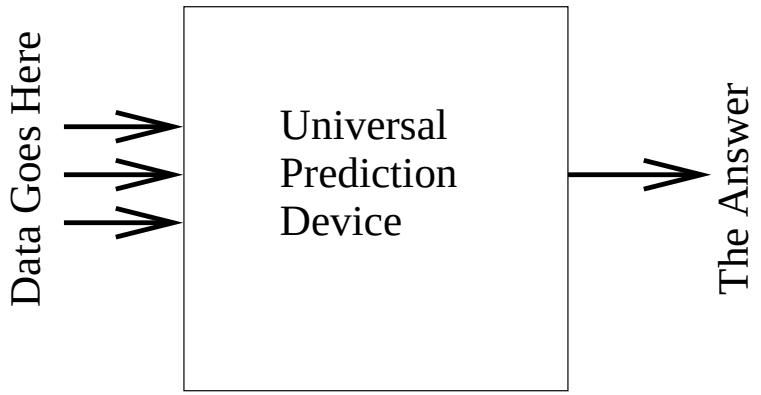
John Langford, TTI Chicago

(Joint with Alina Beygelzimer, Varsha Dani, Tom Hayes,
Bianca Zadrozny, and others)

Workshop on Modeling in Classification and Statistical Learning

October 4, 2005

A Machine Learning Dream



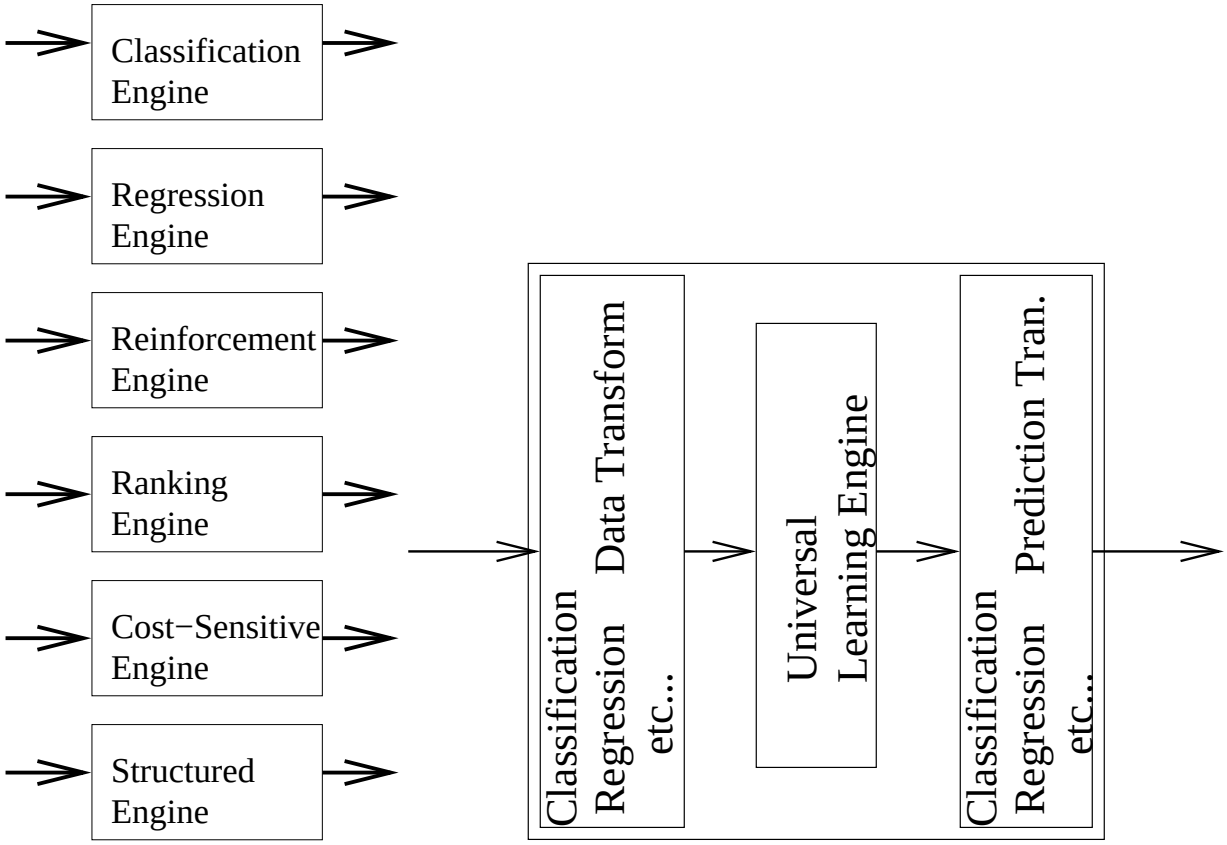
Is the dream possible?

No: That's what crypto is all about.

Yes: Humans do it well on the problems we care about.

It is all a question of extent.

Two Versions of the Dream



"New problem/new algorithm" vs. "it's all one problem"

Basic Questions

1. Can this work, mathematically? [yes]

2. Can this work, empirically? [yes]

3. How do you reason about the {data, prediction transform} pair? [new theorem style]

The Reductionist Style for Solving Learning Problems

1. Define the real problem.

2. Create a {data, prediction transform} pair.

3. Analyze (2).

   (a) If satisfied use with a standard learning algorithm.

   (b) If not, use (3) and go to (2) again.

Defining the Learning Problem

Learning Problem $= D, X, K, Y, \ell$

$K =$ "advice"

$X =$ Features

$D =$ measure on $X \times K$ you care about

$Y =$prediction space

$\ell : K \times Y \to [0, \infty) =$ loss

[see KDDcup 2004]

Step (1) Class Probability Estimation

- Problem: A measure D on $X \times \{0,1\}$ where X is a feature 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:

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

Step (2) The Probing Method: Observations

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

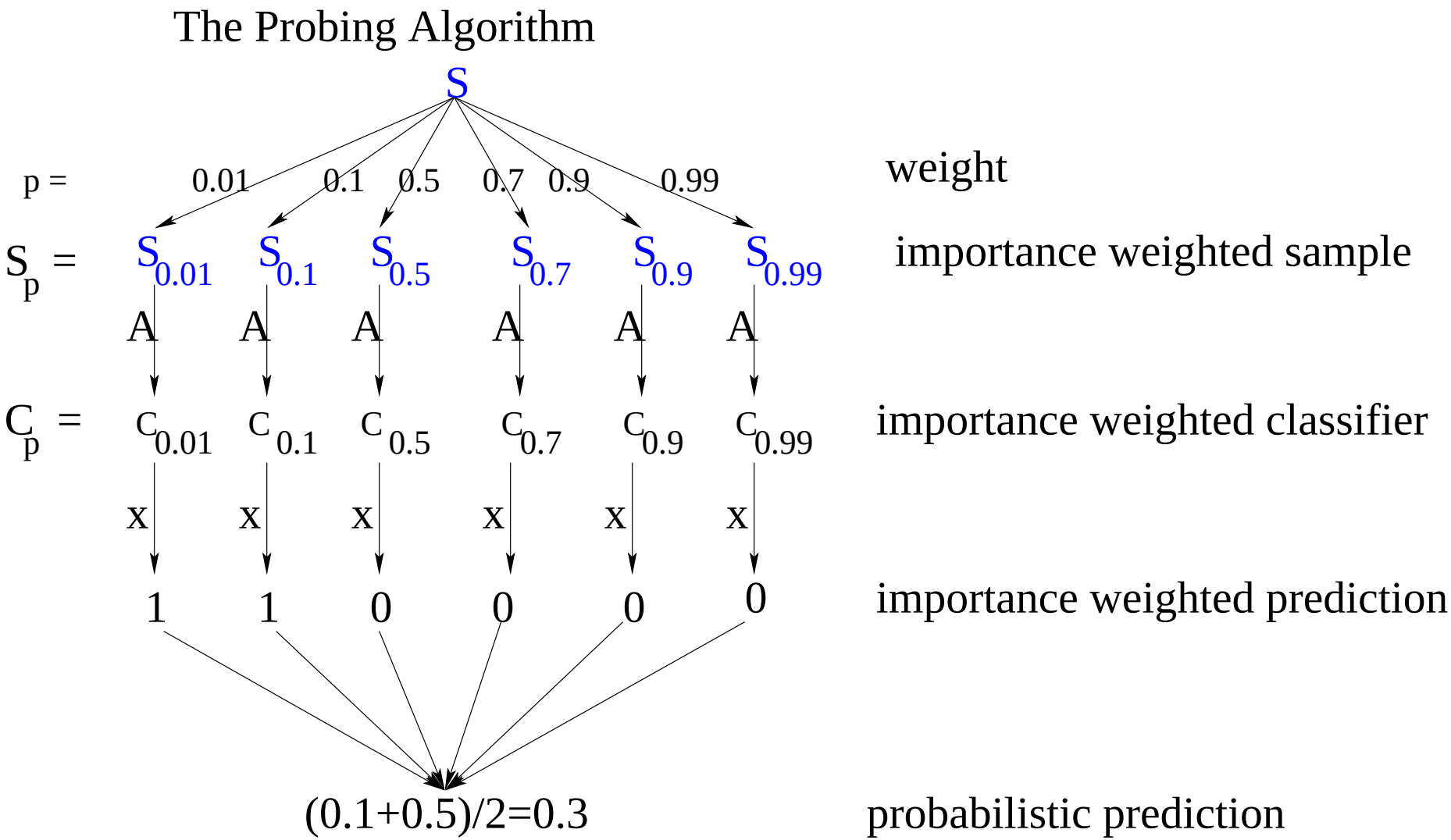
1. Pick $p \in (0, 1)$.

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_p =
$C_{0.01}$ $C_{0.1}$ $C_{0.5}$ $C_{0.7}$ $C_{0.9}$ $C_{0.99}$
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

Step (2) 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]

Step (3) 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 $c_p(x) = c(x, p)$

Good for 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 $\text{Probing}(D)$.

Step (3) Probing Theory

Cute observation: $\text{Probing}_c(x) = \text{Pr}_{p \sim U(0,1)}(c(x,p) = 1)$

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 $D(y|x)$.

Step (3) The proof

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 $= 2|p - D(1|x)|$.

Step (3), The Proof II

How can ϵ binary errors maximize probability estimation errors?

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

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

Step 4: Application

[See KDDcup 2004 results]

What Are Learning Reductions?

1. Mechanism for creating many learning algorithms for new learning problems.

2. A simple, easy-to-use theory gun.

3. A method for understanding learning problem relationships.

4. An atomization method: decomposes learning into subproblems and reassembles them to solve a "harder" problem.

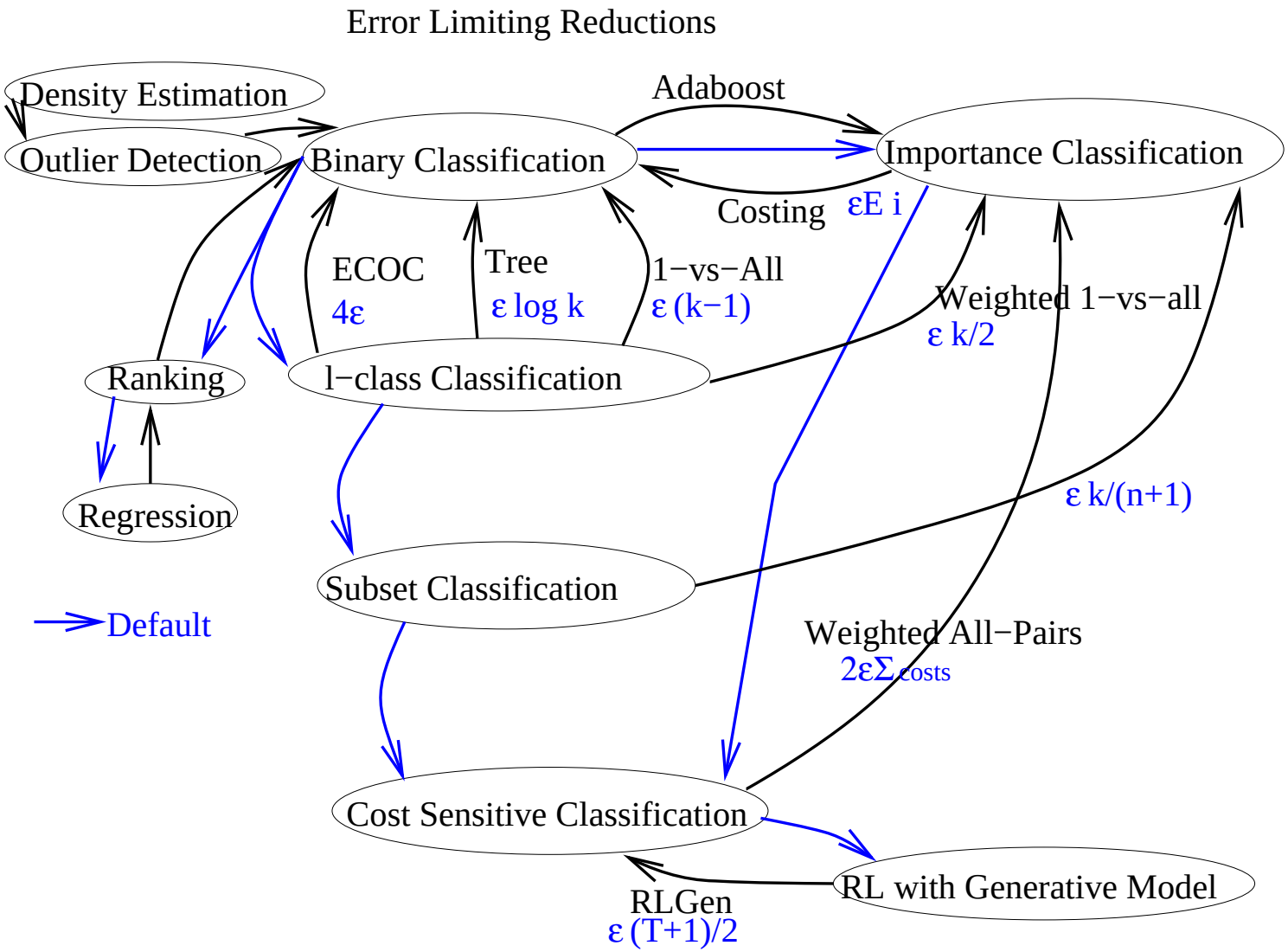
5. A formal specification of some intuitions often used in applied machine learning. Formalization facilitates.

Disadvantages of Learning Reductions

1. Not a complete story. Reductions are a "rule of thumb" kind of analysis where easy reduction does not imply it works, always. Example: reductions that encrypt the feature space.

2. The computational requirements may sometimes be nontrivial (multiplication by x10, perhaps). The "one classifier trick" may be helpful.

3. There is no guarantee that the created subproblems are "easy".

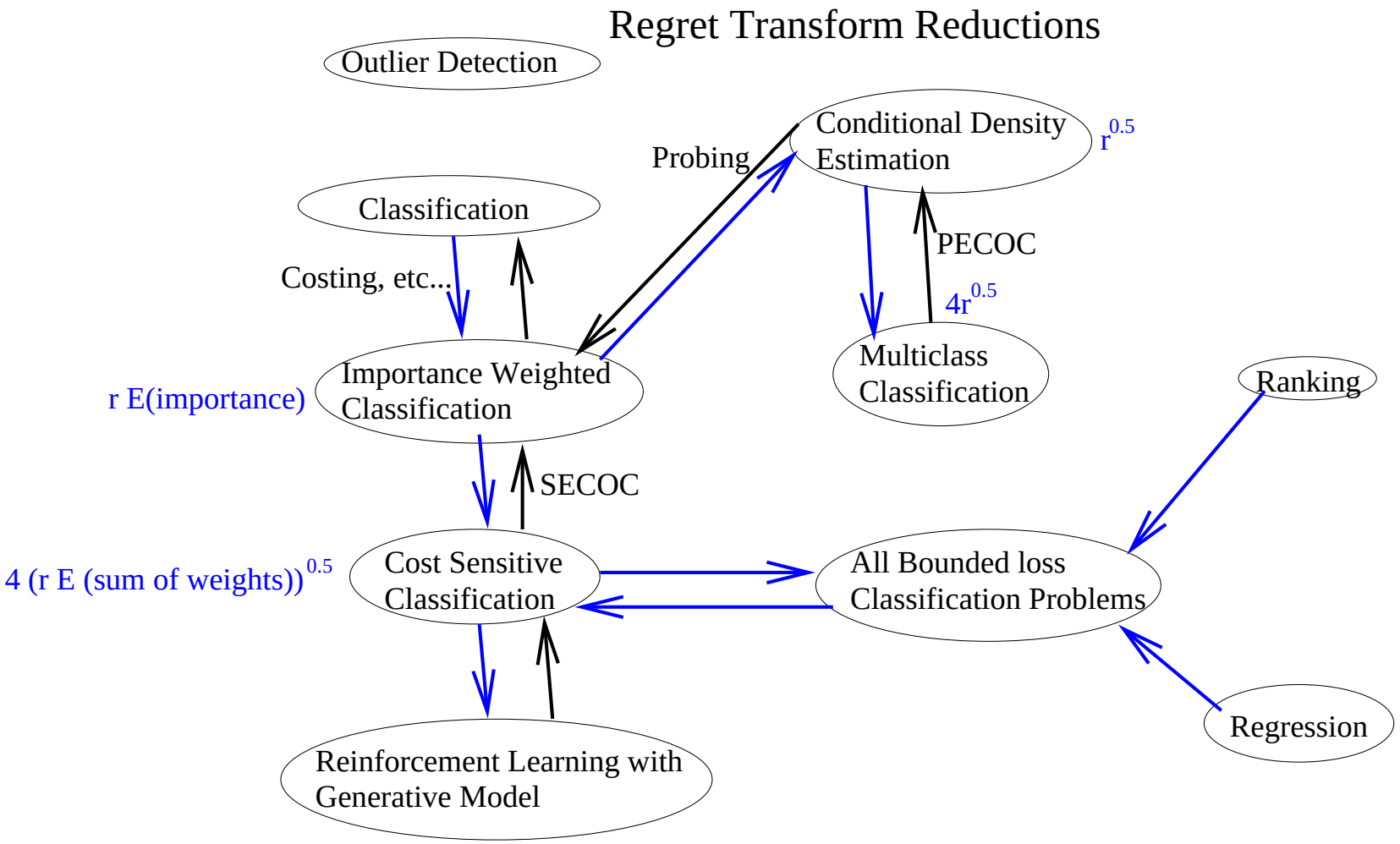

Error Limiting Reductions
Density Estimation
Outlier Detection
Binary Classification
Adaboost
Importance Classification
Costing
εE i
ECOC
4ε
Tree
ε log k
1−vs−All
ε (k−1)
Weighted 1−vs−all
ε k/2
Ranking
l−class Classification
Regression
ε k/(n+1)
Subset Classification
Default
Weighted All−Pairs
2εΣ costs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

Uses of Error Limiting Reductions

- Tom Dietterich can reason that ECOC is plausibly useful before doing experiments because it has superior mathematical robustness.

- Yann LeCun can reason that "end-to-end learning" is superior to piecewise learning, because it has superior robustness. More generally, suggests "learning digestion": more direct estimation of the prediction.

- Anyone can build dozens of learning algorithms to address new losses and new information sources with the effort of building one learning algorithm.


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

Uses of Regret Transform Reductions

- Tom Dietterich can prove using ECOC with probabilistic decoding is better than hard decoding.

- Hal Daume might build a better machine translation system using reductions-motivated modifications.

- Dan Roth can justify (and perhaps improve) his language learning system.

Big Questions

1. Is there a *constructive* reduction of reinforcement learning to classification without side information?

2. What lower bounds on reductions exist?

3. Are there better forms of reductions than {error minimizing, regret transform}?