Reductions for Machine Learning

John Langford

TTI-Chicago

With:

Naoki Abe, Alina Beygelzimer, and Bianca Zadrozny - IBM TJ Watson

Varsha Dani, Tom Hayes - University of Chicago

What is a Reduction?

Reduction = Machine that turns problem A solver into problem B solver

What is a Reduction?

Reduction = Machine that turns problem A solver into problem B solver

A = Classification, Regression, Density Estimation, Reinforcement Learning, etc...

Why Reductions for Machine Learning?

1. Conservation of Effort

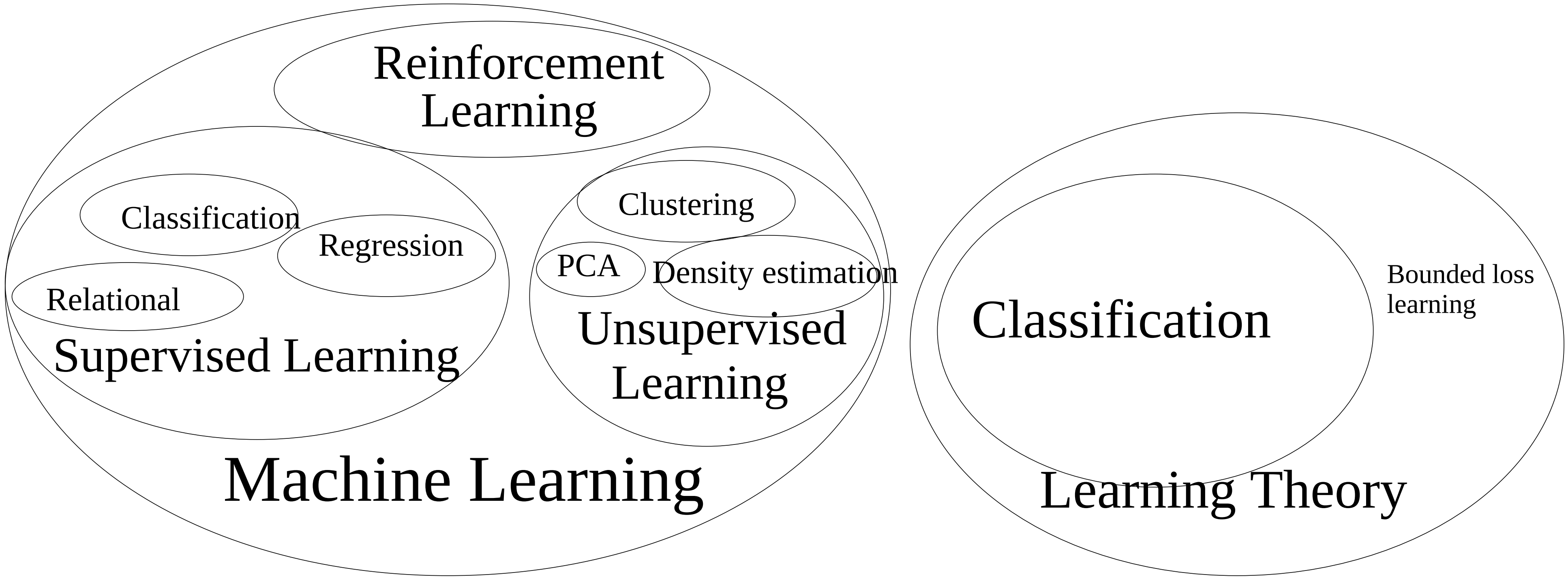
2. Crystallization of Learning Ability

3. Inherent Modularity/Code Reuse

4. A “language” of Construction

5. Structural understanding of learning

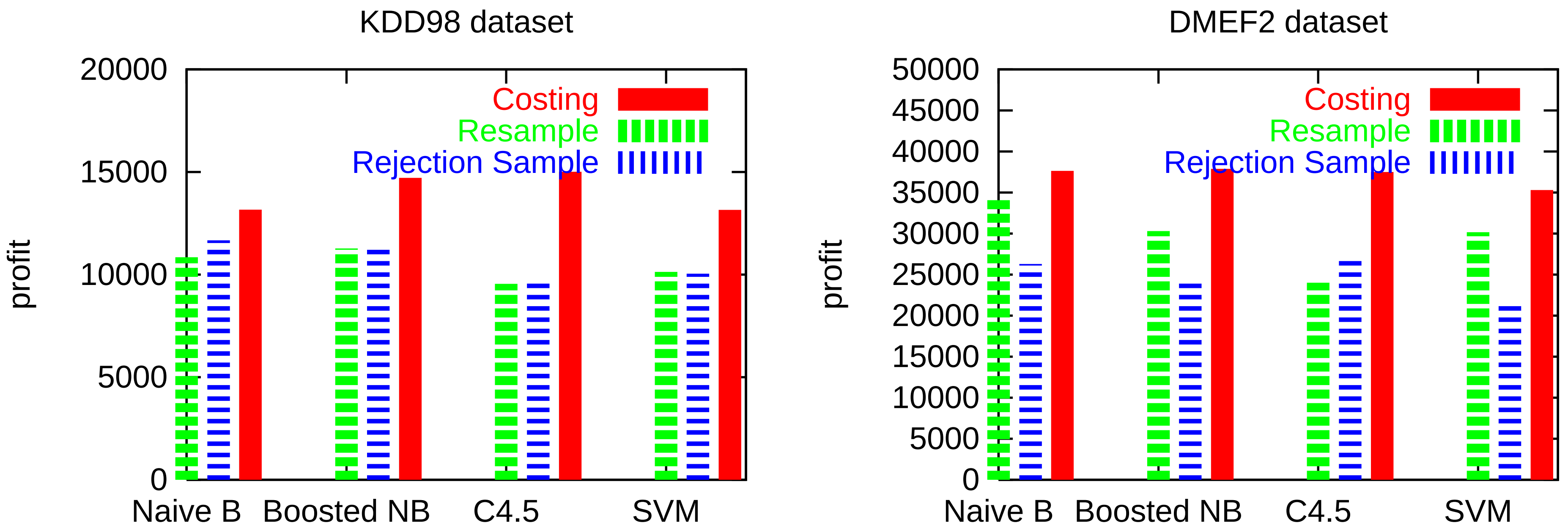
Th ICML and COLT view of Work Done

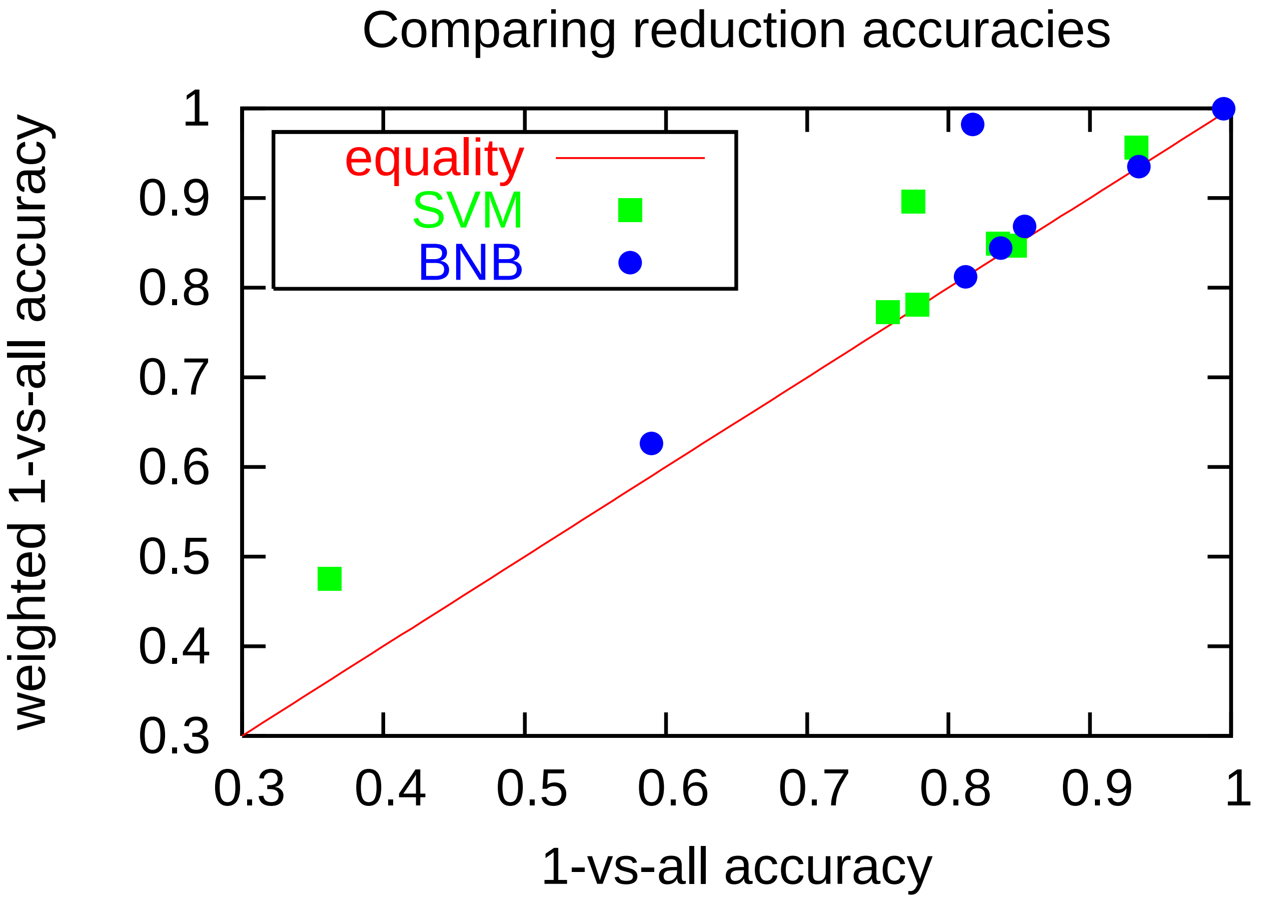


Why Not Reductions for Machine Learning?

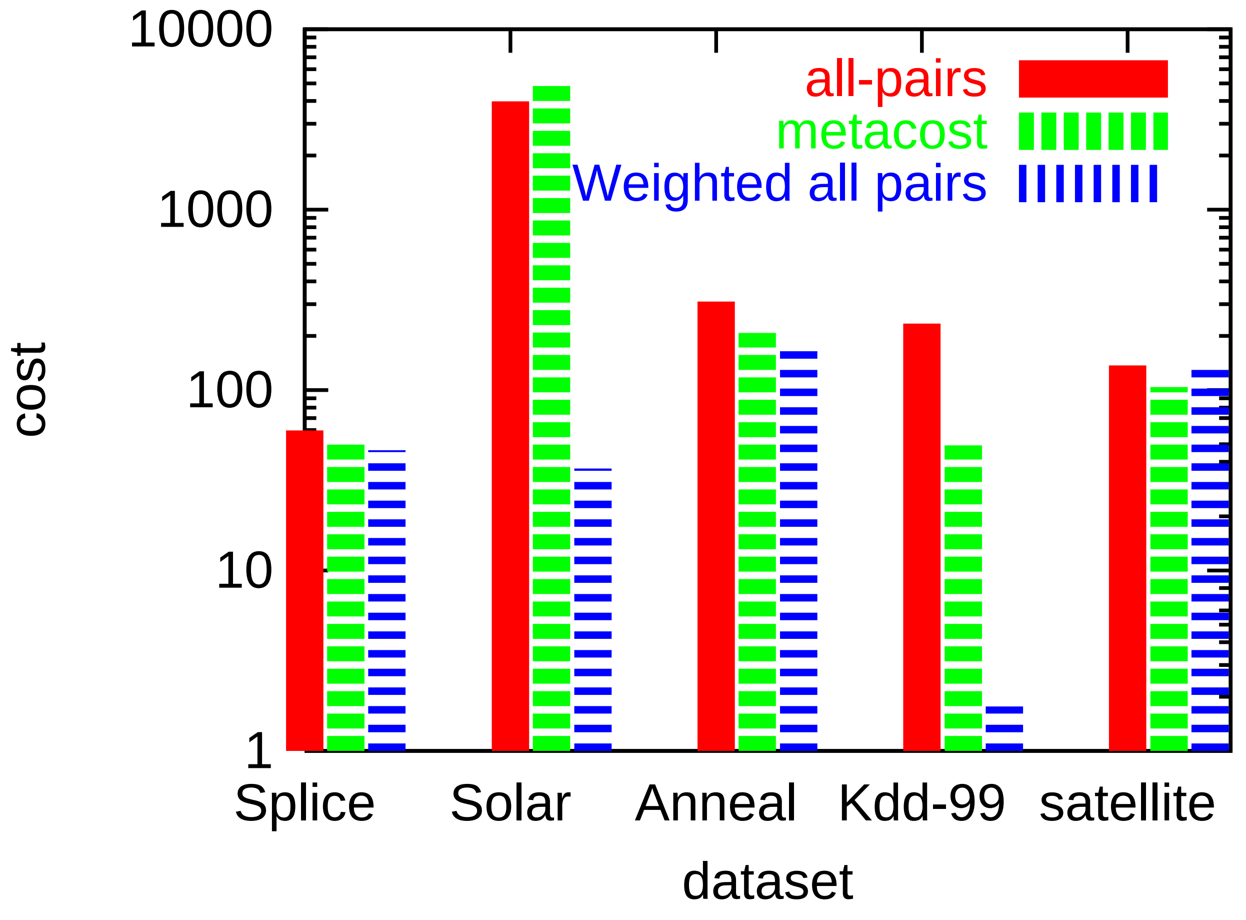
1. Maybe specialized problem solvers inherently work better?

Importance Weighted Classification to Binary Classification




Comparing reduction accuracies
equality
SVM
BNB
weighted 1-vs-all accuracy
1-vs-all accuracy
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Cost Sensitive Classification to Binary Classification



Outline

1. Define Reduction
   (a) Define learning Task
   (b) Learning problems exist within a learning Task.
   (c) Learning algorithms solve learning problems within a Task.
   (d) Reductions transform learning algorithm Task.
2. Show Typical Theorem
3. Discuss Structure

Learning Tasks

Learning Tasks $= (K, Y, l)$

K = information about *what* should be predicted.

Y = the space of predictions

$l : K \times Y \to [0, \infty)$ = "loss" function

L	K	Y	$l(k,y)$
Classification	$\{0,1\}$	$\{0,1\}$	$I(k \neq y)$
Importance Classification	$\{0,1\} \times [0,\infty)$	$\{0,1\}$	$k_2 I(k_1 \neq y)$
ℓ-class Classification	$\{1,...,\ell\}$	$\{1,...,\ell\}$	$I(k \neq y)$
Ranking	$\{1,...,\ell\}$	$\{1,...,\ell\}$	$\lvert k-y \rvert$
n-of-ℓ Classification	$2^{\{1,...,\ell\}}$	$\{1,...,\ell\}$	$I(y \not\in k)$
Cost Sensitive Classification	$[0,\infty)^\ell$	$\{0,...,\ell\}$	k_y
Regression	$(-\infty,\infty)$	$(-\infty,\infty)$	$(k-y)^2$

Learning Problem

Learning problem =

- (K, Y, l) Learning Task

-X = “feature space” = the thing you predict from

-D a distribution over $X \times K$

Outline

1. Define Reduction
   (a) Define learning Task
   (b) Learning problems exist within a learning Task.
   (c) Learning algorithms solve learning problems within a Task.
   (d) Reductions transform learning algorithm Task.
2. Show Typical Theorem
3. Discuss Structure

Learning Algorithm = Task solver

Examples:

1. Support Vector Machines (Task = Classification)
2. EM (Task = Density estimation)
3. Neural Networks (Task = Classification or Regression)

Reduction = Task Solver given Task' Solver

Reduction satisfies one property

1. (Error limitation) If Task' Solver does well on derived problems then reduction does well on problem.

Outline

1. Define Reduction
2. Show Typical Theorem
   (a) Multiclass to Binary Classification
   (b) One-vs-All
   (c) Weighted one vs-all
3. Discuss Structure

Task Comparison

	Multiclass C.	Binary C.	Importance C.
K	$\{1, ..., \ell\}$	$\{0, 1\}$	$\{0, 1\} \times [0, \infty)$
Y	$\{1, ..., \ell\}$	$\{0, 1\}$	$\{0, 1\}$
$l(k, y)$	$I(k \neq y)$	$I(k \neq y)$	$k_2 I(k_1 \neq y)$

One vs. All: Multiclass $\Rightarrow$ Binary

Given $S = (x, y_M)^m$, $A : (X \times \{0,1\})^* \to (X \to \{0,1\})$

1. Create $\forall i \in \{1, ..., \ell\}$ $S_i = \{(x, I(y_M = i)) : (x, y_M) \in S\}$
2. Learn $h_i = A(S_i)$
3. Output: $h_M(x) = \text{Uniform}\{i : h_i(x) = 1\}$ or Uniform over all labels

One vs all: basic theory

Theorem:

If h_i has error rate ϵ, then h_M has error rate at most $(\ell - 1)\epsilon$.

Proof:

(1) If example is (x, i) and $h_i(x) = 0$, error probability is $\frac{\ell-1}{\ell}$

(2) No other combination of errors is more efficient.

ℓ classifiers with binary error rate ϵ

$\Rightarrow$ multiclass error rate $\frac{\ell-1}{\ell}\ell\epsilon$

One vs all: Basic observation

"false negative" error mode worse than "false positive" error mode.

$\Rightarrow$ "false negative" more expensive $\Rightarrow$ better performance.

Weighted One vs all: Multiclass $\Rightarrow$ Importance Classification

Given $S = (x, y_M)^m$, $A : (X \times \{0,1\} \times [0,\infty))^* \rightarrow (X \rightarrow \{0,1\})$

1. Let $c_0 = \frac{\ell}{2}$ and $c_1 = \ell - 1$
2. Create $S_i = \{(x, I(y_M = i), c_{I(y_M=i)}) : (x, y_M) \in S\}$
3. Learn $h_i = A(S_i)$
4. Output: $h_M(x) = \text{Uniform}\{i : h_i(x) = 1\}$ or Uniform over all labels

Weighted One vs all: basic theory

Theorem: (Importance to Classification)

If h_{Binary} has error rate ϵ, then h_{Import} has loss rate at most $\epsilon E_{(x,y,i)\sim D} i$.

Theorem: (Multiclass to Importance)

If h_i has loss rate $\epsilon E_{(x,y,i)\sim D} i$, then h_{Multi} has error rate at most $\frac{\ell+1}{2}\epsilon$.

Outline

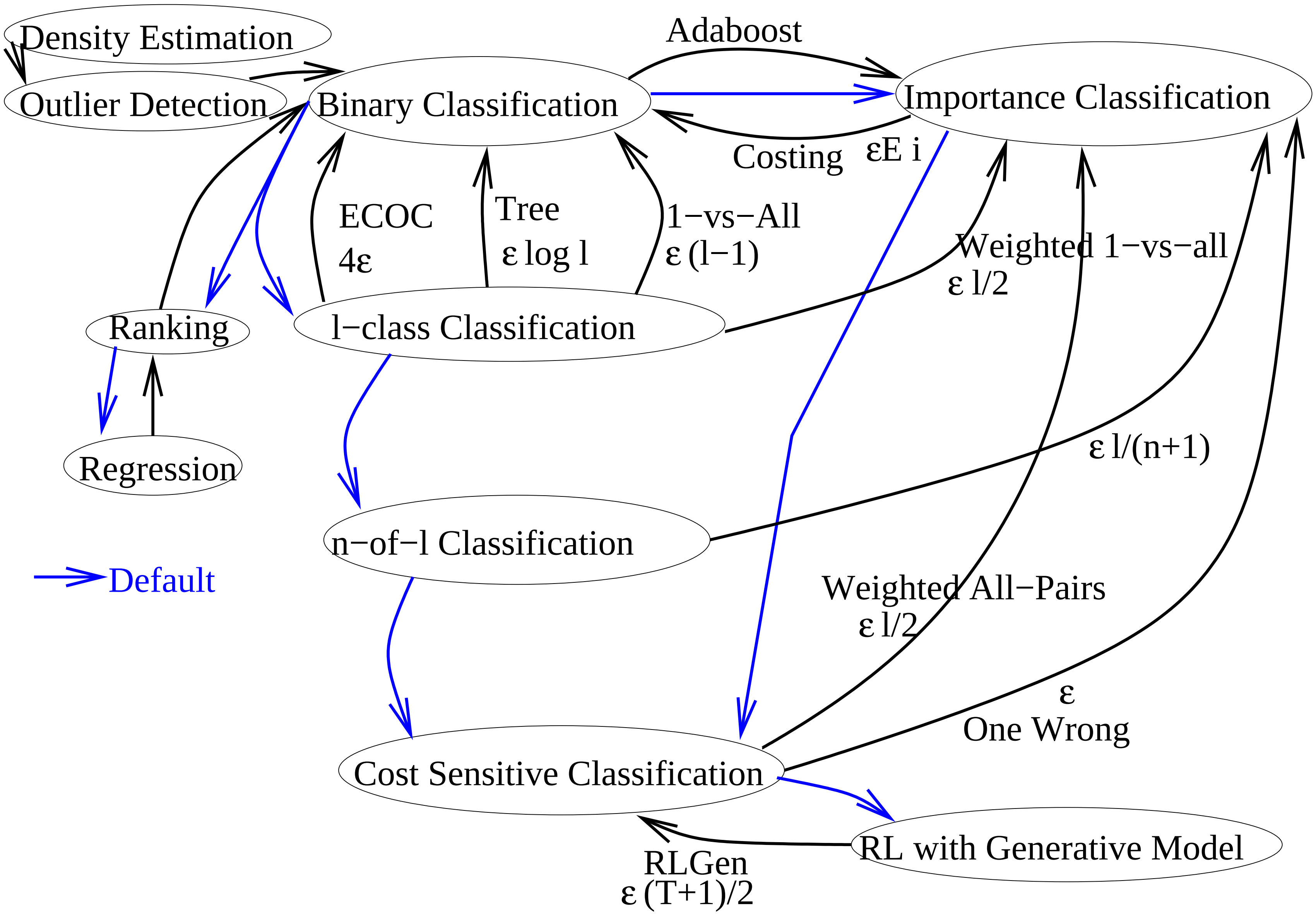
1. Define Reduction

2. Show Typical Theorem

3. Discuss Structure

   (a) (some of) What's known

   (b) What we may infer



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

Easy reductions:

1. Multiclass to Binary (many done previously)

2. Importance Classification to Binary

3. Ranking/Probability to Binary Classification

Hard Reductions:

1. Reinforcement Learning to Cost Sensitive Classification

2. Cost Sensitive Classification to Classification

"I am not a predictor." - George W. Bush

I predict reductions will help us predict.

Several papers at:

http://hunch.net/~jl/projects/reductions/reductions.html