Is Learning the Whole Easier than Learning the Sum of the Parts?

Rich Caruana
Cornell University

Yes, and No.

Thank You.

Questions?

Yes, and No?

Approach: Inductive Transfer

- A.K.A.
  - Bias Learning
  - Multitask learning
  - Learning (Internal) Representations
  - Learning-to-learn
  - Lifelong learning
  - Continual learning
  - Speedup learning
  - Hints
  - Hierarchical Bayes
  - …

Goal

- *Not* to learn a complex structure
  - *Not* worried about consistency among parts
  - *No* constraints among predictions
- Instead, trying to learn a simple thing (atom) well by learning a more complex structure
  - Learn you risk of dying from pneumonia
  - Learn to steer a car
  - Learn to recognize doorknobs
  - …
- Goal is better generalization from finite data
- *Not* faster, *not* more intelligible, *not* one model, …

Toy Multitask Learning Example

- 4 tasks defined on eight bits B_1-B_8:

$$\text{Task 1} = B_1 \vee \text{Parity}(B_2 - B_6)$$

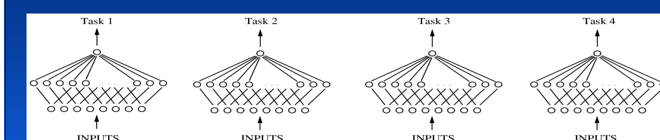
$$\text{Task 2} = \neg B_1 \vee \text{Parity}(B_2 - B_6)$$

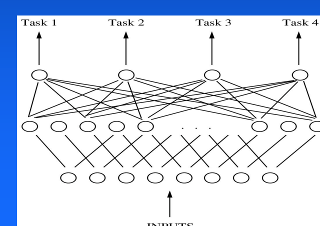
$$\text{Task 3} = B_1 \wedge \text{Parity}(B_2 - B_6)$$

$$\text{Task 4} = \neg B_1 \wedge \text{Parity}(B_2 - B_6)$$

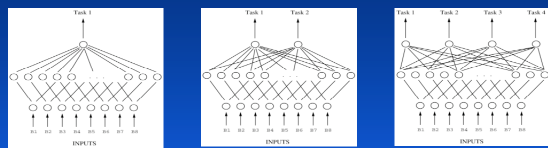
- all tasks ignore input bits B_7-B_8

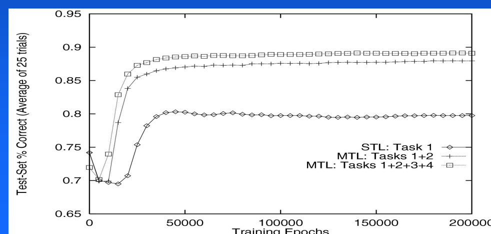
Toy Example: STL & MTL



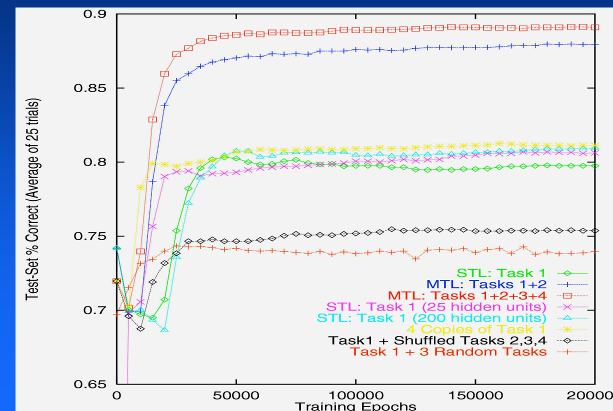


Toy Example: Results





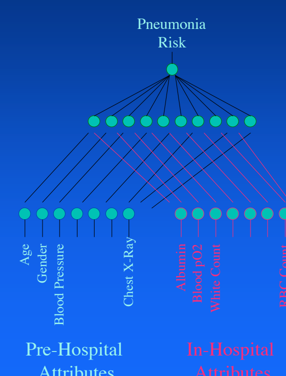
Toy Example: Why?



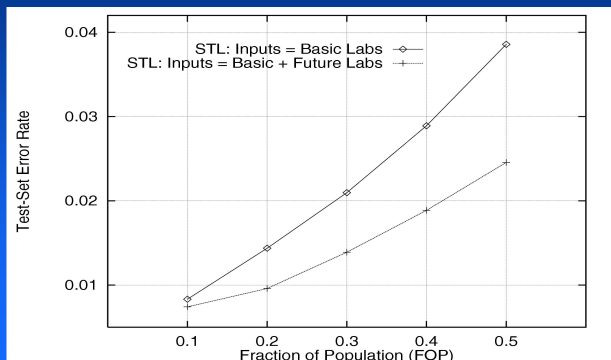
Outline

- Application of MTL to Pneumonia Risk
- MTL nets cluster tasks by function
- When is MTL likely to be useful?
- MTL in K-Nearest Neighbor
- MTL for Bayes Net Structure Learning
- Learn globally, predict locally?
- Different approach to structure learning
- Model Compression

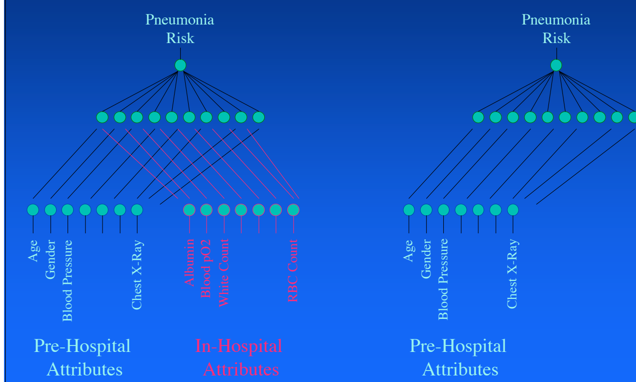
Predicting Pneumonia Risk



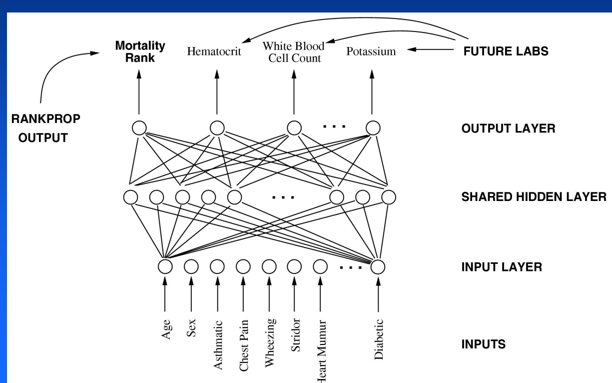
Pneumonia: Hospital Labs as Inputs



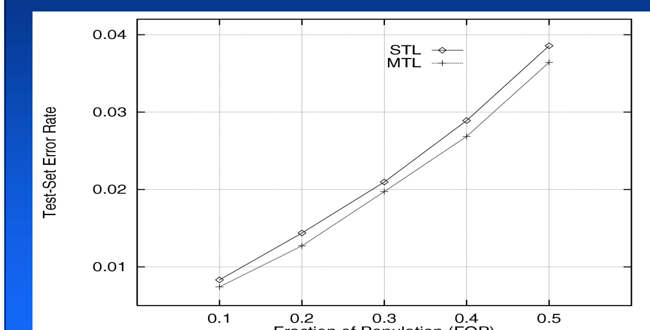
Predicting Pneumonia Risk



Pneumonia #1: Medis



Pneumonia #1: Results

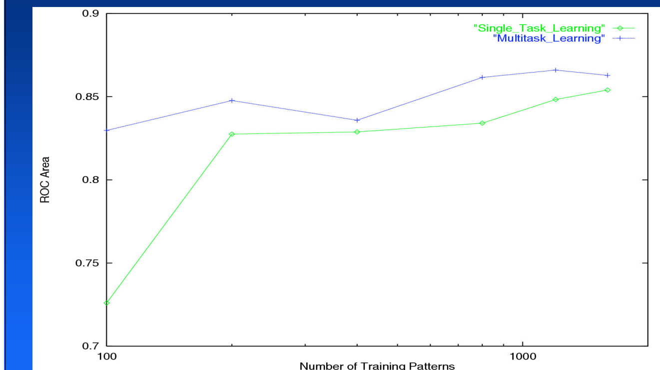


-10.8% -11.8% -6.2% -6.9% -5.7%

Pneumonia #2: PORT

- 10X fewer cases (2286 patients)
- 10X more input features (200 feats)
- missing features (5% overall, up to 50%)
- main task: dire outcome
- 30 extra tasks currently available
  - dire outcome disjuncts (death, ICU, cardio, ...)
  - length of stay in hospital
  - cost of hospitalization
  - etiology (gramnegative, grampositive, ...)
  - . . .

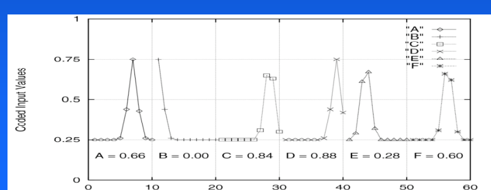
Pneumonia #2: Results



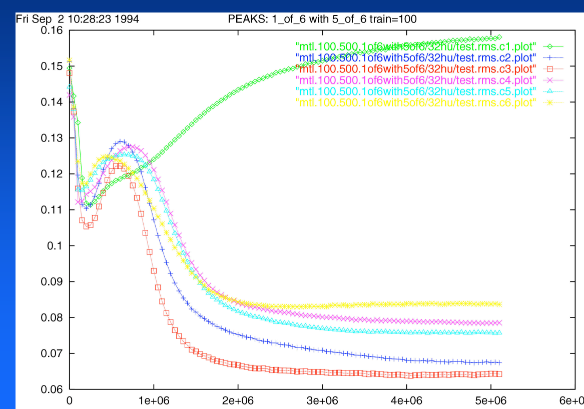
MTL reduces error >10%

120 Synthetic Tasks

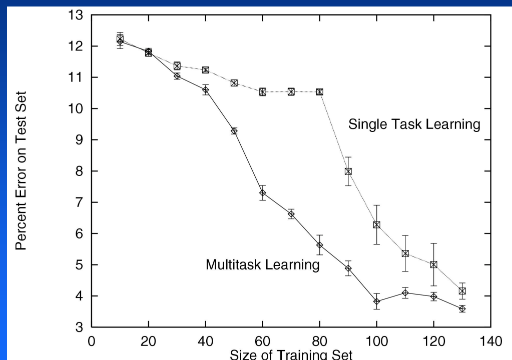
- backprop net not told how tasks are related, but ...
- 120 **Peaks Functions**: A,B,C,D,E,F $\in$ (0.0,1.0)
  - P 001 = If (A > 0.5) Then B, Else C
  - P 002 = If (A > 0.5) Then B, Else D
  - P 014 = If (A > 0.5) Then E, Else C
  - P 024 = If (B > 0.5) Then A, Else F
  - P 120 = If (F > 0.5) Then E, Else D



Peaks Functions: Results



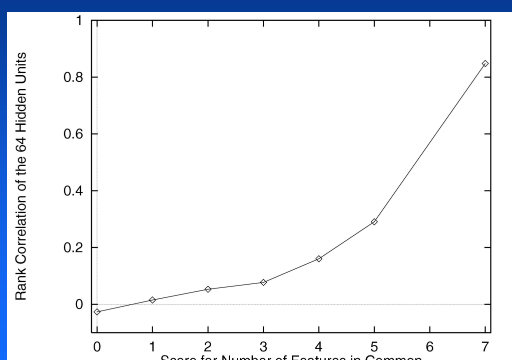
Peaks Functions: Results



courtesy Joseph O'Sullivan

MTL nets cluster tasks by *function*

Peaks Functions: Clustering



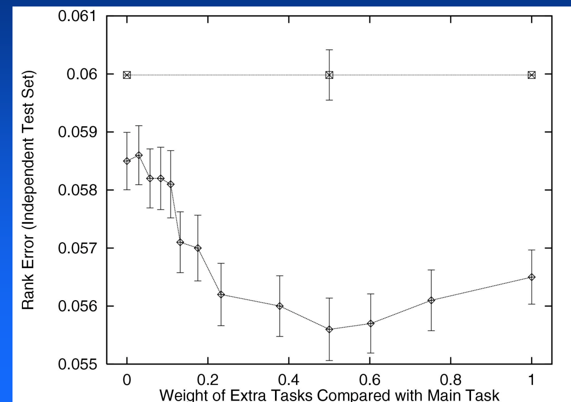
Related ≠ Correlated

- Some peaks functions have zero correlation yet are strongly related and help each other:
  - P 001 = If (A > 0.5) Then B, Else C
  - …
  - P 005 = If (A > 0.5) Then C, Else B
  - …
  - P 014 = If (A > 0.5) Then D, Else E
- Related ~ Mutual Information

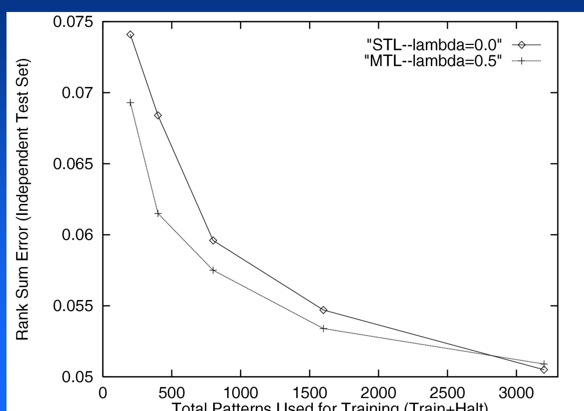
MTL in K-Nearest Neighbor

- Most learning methods can MTL:
  - shared representation
  - combine performance of extra tasks
  - control the effect of extra tasks
- MTL in K-Nearest Neighbor:
  - shared representation: distance metric
  - MTLPerf = $(1-\lambda)*\text{MainPerf} + \Sigma\ (\lambda*\text{ExtraPerf})$

MTL/KNN for Pneumonia #1



MTL/KNN for Pneumonia #1



Related ≠ Correlated

- KNN tasks do not need to be correlated for the distance function learned for one to be effective for another
- Note there are no constraints (no structure) on related tasks

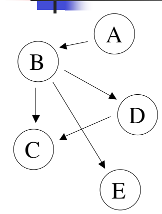
Learning the Structure of Related Tasks

Alexandru Niculescu-Mizil

Rich Caruana

Cornell University

Bayesian Networks



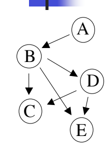
A \ B	0	1
0	θ_0	$1-\theta_0$
1	θ_1	$1-\theta_1$

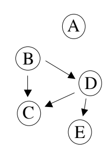
- A Bayesian Network is a compact encoding of the joint distribution of a set of variables.
- A Bayes Net consists of:
  - A DAG that encodes the dependency structure of the domain.
  - A set of conditional probability functions.
- One can learn from data:
  - The dependency structure.
  - The parameters of the conditional probability functions.

Motivation

- Learning Bayes Net structure from data can provide useful information.
  - E.g. from gene expression data, one can discover regulatory relations between genes for one yeast type. [Friedman et al. `00]
- Related tasks should have similar dependency structures, so more accurate Bayes Net structures can be learned by taking advantage of these similarities. (Inductive Transfer).
  - E.g. from gene expression data for different species of yeast more accurate regulatory relations can be learned for each of them by taking advantage of the fact that different species of yeast should have similar regulatory structures.

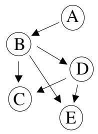
Multi-task Structure Learning





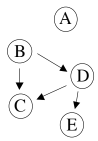
- $D_1, D_2, \ldots, D_k$ complete iid data from k related tasks.
- Simultaneously learn k Bayes Net structures, one for each task.
- Take advantage of the similarity between tasks by biasing the learning algorithm towards learning similar structures.
- Configuration = a set of structures, one for each task.

Probability of a Configuration



- The posterior probability of a configuration given the data:

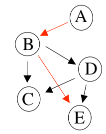
$$P(G_1, ..., G_k|D_1, ..., D_k) \propto P(G_1, ..., G_k)P(D_1, ..., D_k|G_1, ..., G_k)$$



- Under some assumptions we get:

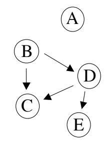
$$P(G_1, ..., G_k|D_1, ..., D_k) \propto P(G_1, ..., G_k) \prod_{p=1}^{k} P(D_p|G_p)$$

The Prior



- Penalize differences between structures. Prior for two tasks:

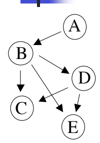
$$P(G_1, G_2) = Z_\delta P(G_1)P(G_2) \prod_{\substack{(X_i, X_j) \in \\ G_1 \Delta G_2}} (1 - \delta_{ij})$$

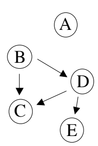


$$P(G_1, ..., G_k) = Z \left(\prod_{1 \le i < j \le k} P(G_i, G_j) \right)^{\frac{1}{k-1}}$$

- δ is a parameter that needs to be specified

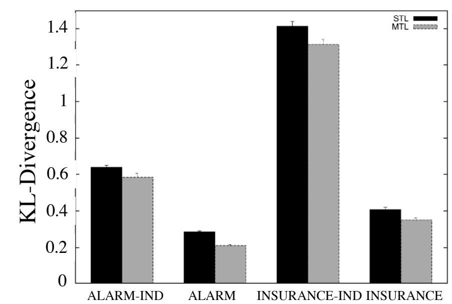
Multi-Task Structure Learning





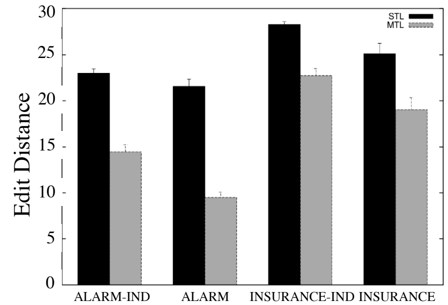
- Find a configuration with a high posterior probability via greedy hill climbing:
  1. Start from an initial configuration
  2. Find neighboring configuration with the highest probability
  3. If the configuration found at step 2 has higher probability than the current one then move to it and iterate.
     Else return the current configuration.

KL-Divergence Performance

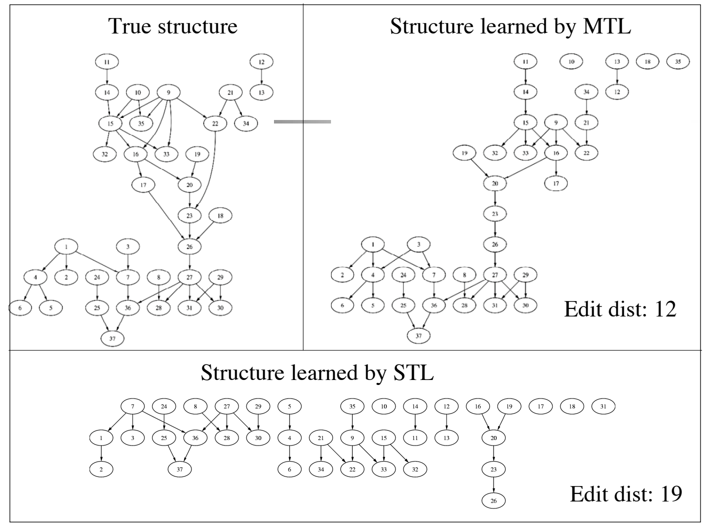


- MTL reduces the KL-divergence by 7% - 26%

Edit Distance Performance



- MTL reduces the number of incorrect arcs by 20% - 55%.



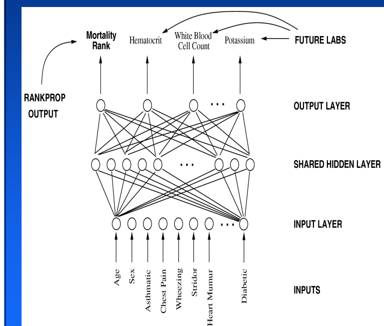
When to use MTL?

- using future to predict present
- time series
- disjunctive/conjunctive tasks
- multiple error metric
- quantized or stochastic tasks
- focus of attention
- sequential transfer
- different data distributions
- hierarchical tasks
- some input features work better as outputs

Multiple Tasks Occur Naturally

- Mitchell's Calendar Apprentice (CAP)
  - time-of-day (9:00am, 9:30am, ...)
  - day-of-week (M, T, W, ...)
  - duration (30min, 60min, ...)
  - location (Tom's office, Dean's office, 5409, ...)
- Often correlation, but no constraints (structure), among tasks

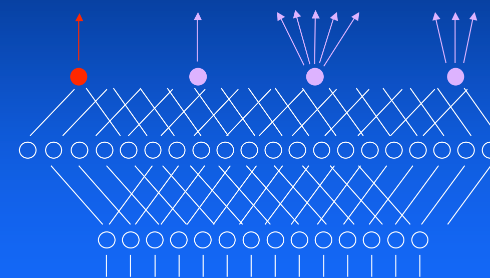
Using Future to Predict Present



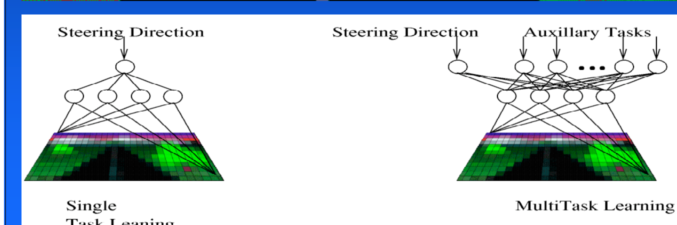
- **medical domains**
- autonomous vehicles and robots
- **time series**
  - stock market
  - economic forecasting
  - weather prediction
  - spatial series
- many more

Decomposable Tasks

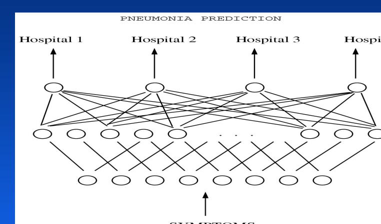
DireOutcome = ICU v Complication v Death



Focus of Attention: ALVINN



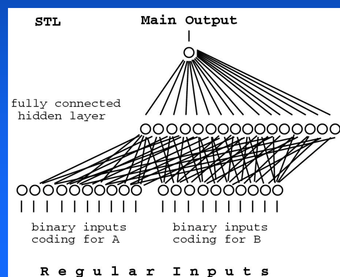
Different Data Distributions



- Hospital 1: 50 cases, rural (Ithaca or Williamstown)
- Hospital 2: 500 cases, urban (Des Moines)
- Hospital 3: 1000 cases, elderly suburbs (Florida)
- Hospital 4: 5000 cases, young urban (LA,SF)

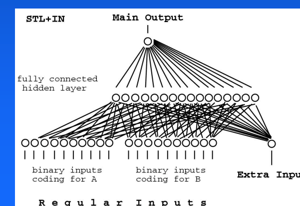
Some Inputs are Better as Outputs

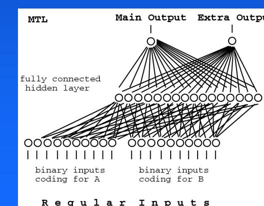
- MainTask = Sigmoid(A)+Sigmoid(B)
- A, B ∈ (–5.0, +5.0)
- Inputs A and B coded via 10-bit binary code



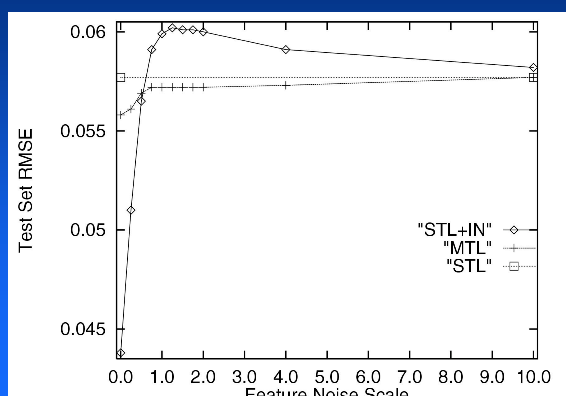
Some Inputs are Better as Outputs

- MainTask = Sigmoid(A)+Sigmoid(B)
- Extra Features:
  - EF1 = Sigmoid(A) + λ * Noise
  - EF2 = Sigmoid(B) + λ * Noise
  - where λ ∈ (0.0, 10.0), Noise ∈ (-1.0, 1.0)

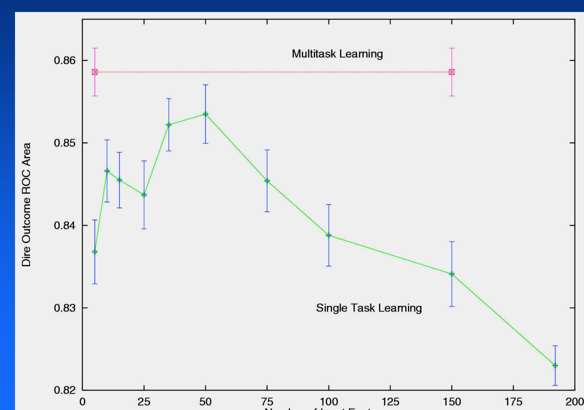




Inputs Better as Outputs: Results

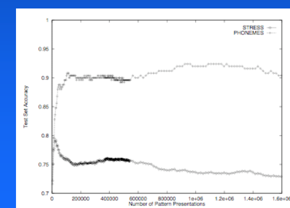


Inputs Better as Outputs: Results



Inductive Transfer Does Not Mean Should Learn One Model

- A helps B ≠ B helps A
  - sometimes benefit is mutual
  - sometimes A helps B but B hurts A
  - sometimes model best for A is suboptimal for B



How You Learn can be Independent of How You Make Predictions

- To get best performance, must learn sets of related tasks in parallel
- Resulting models can be complex
  - forced sharing often hurts more than it helps
  - learned models can be large

Transfer vs. Structured Outputs

- Multitask Learning:
  - $T_1 = f_1(g(x))$
  - $T_2 = f_2(g(x))$

Transfer vs. Structured Outputs

- Multitask Learning:
  - $T_1 = f_1(g(x), a(x, x`))$
  - $T_2 = f_2(g(x), b(x, x``))$

Transfer vs. Structured Outputs

- Multitask Learning:
  - $T_1 = f_1(g(x))$
  - $T_2 = f_2(g(x))$
- Structure Learning:
  - $y_1 = f_1(x)$ (before constraint)
  - $y_2 = f_2(x)$ (before constraint)
  - $y_{1,}\ y_2 = g(f_1(x), f_2(x))$ (structural constraints in g)

Global Inference in Learning for Natural Language Processing

Vasin Punyakanok

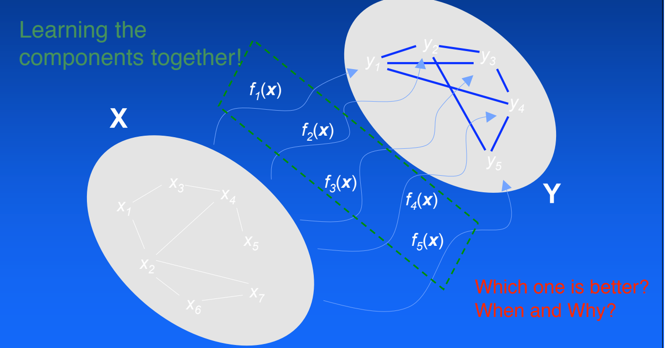
Department of Computer Science

University of Illinois at Urbana-Champaign

Joint work with Dan Roth, Wen-tau Yih, and Dav Zimak

Learning and Inference

IBT: Inference-based Training



Comparisons of Learning Approaches

Coupling (IBT)

- Optimize the true *global* objective function (this should be better in the limit)

Decoupling (L+I)

- More efficient
- Reusability of classifiers
- Modularity in training
  - No *global* examples required
  - Can use appropriate model for each piece of problem

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 + Ensemble = AUC Score on the 1k validation set

Model Library	AUC Score on the 1k validation set		Ensemble
Model 1	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 + Ensemble = AUC Score on the 1k validation set

Model Library	AUC Score on the 1k validation set	Ensemble
Model 1	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

Model 1
Model 2

Model 4
Model 5
Model 6
Model 7
Model 8

Ensemble

Model 3 0.9164
Model 9 0.9384

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	Ensemble
Model 1	+ Ensemble =	0.8502	Model 3 0.9164
Model 2		0.9243	Model 9 0.9384
Model 4		0.8992	
Model 5		0.8090	
Model 6		0.9424	
Model 7		0.9045	
Model 8		0.9243	

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

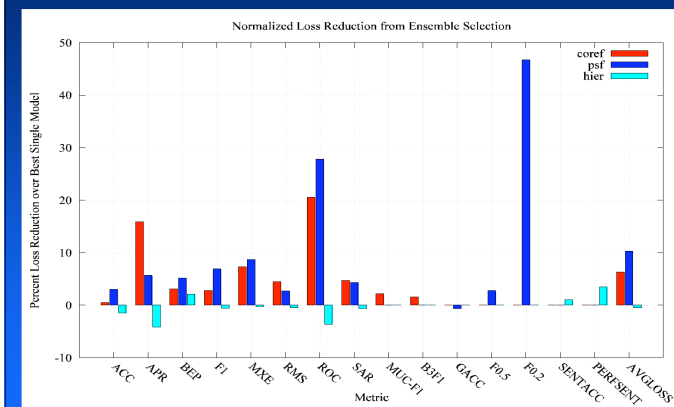
Best of the Best of the Best

	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

Normalized Scores for ES

	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
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

Ensemble Selection vs Best: 3 NLP Problems



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

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

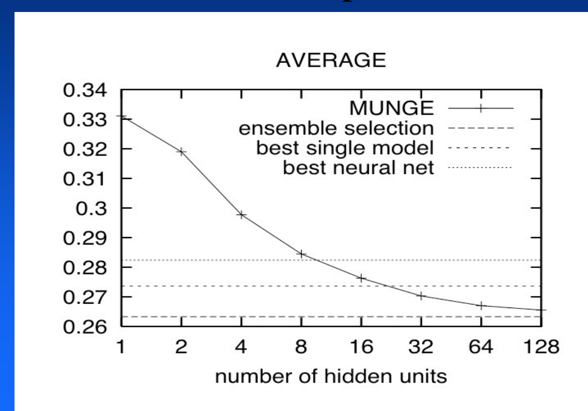
Best Ensembles are Big and Ugly!

- 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!

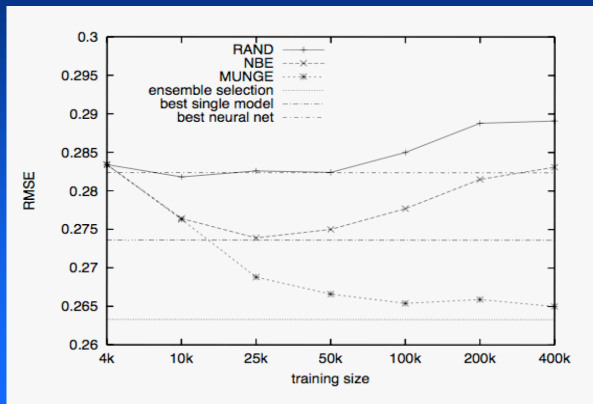
Solution: Model Compression

- 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

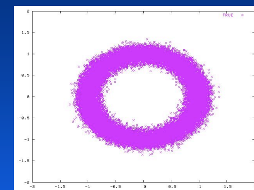
Model Compression



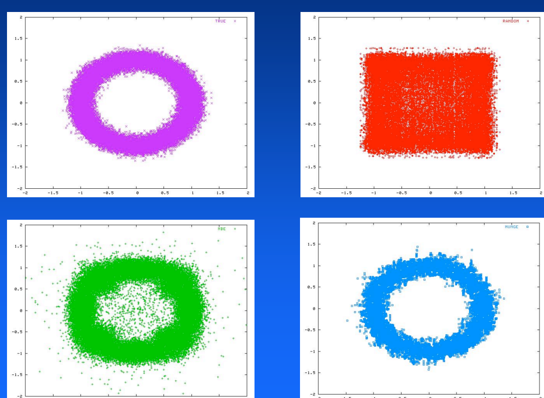
Model Compression



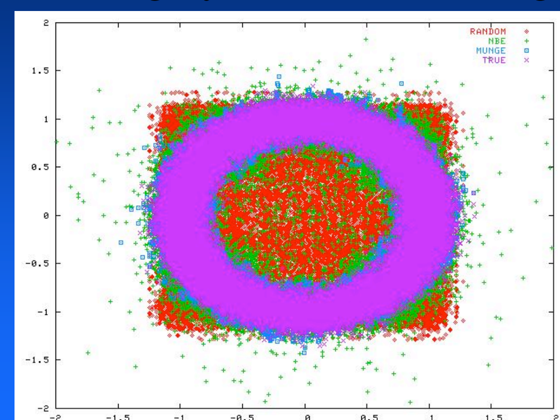
Generating Synthetic Data: *Munging*



Generating Synthetic Data: *Munging*



Generating Synthetic Data: Munging



Summary of Compression Results

- Neural nets trained to mimic high performing ensemble selection models
  - on average, captures more than 97% performance of target model
  - perform much better than any ANN we could train on original data
  - 1000 times faster than ensemble
  - 1000 times smaller than ensemble

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

Thank You.

Questions?