BTP THESIS
Learning with Ambiguous Labels

Pratheek D'Souza Rebello

2017CS10361

Advisors: Prof Mausam, Prof Parag Singla
PhD Supervisor: Yatin Nandwani

Contents

1 Introduction

We define Ambiguous Labelling as problem settings which depart from the classical fully supervised setting of one x and one y as the training input. These include:

1. **Partial Label Learning (PLL)**: One x, multiple candidate y, one and only one y is correct

2. **Multi Instance Learning (MIL)**: Multiple x in a bag, one label y, at least one x corresponds to this label y.

3. **1 of Many Learning (1oML)**: One x, multiple candidate y, all correct

4. **Noisy Label Learning (NLL)**: One x, One y, but y is a noisy label.

2 Unified Problem Definition

We build on 'Structured Prediction with Partial Labelling through the Infimum Loss' [2] and generalise their definition of PLL to Ambiguous Learning in general.

2.1 Notation

Input/Query space : $\mathcal{X}$.
Target/Output space: $\mathcal{Y}$.
Space of subsets of Queries: $\mathcal{Q} \subseteq 2^{\mathcal{X}}$
Space of subsets of targets: $\mathcal{S} \subseteq 2^{\mathcal{Y}}$

Probability distribution over $\mathcal{X} \times \mathcal{Y}$: ρ
Probability distribution over $\mathcal{X} \times \mathcal{Y} \times \mathcal{Q} \times \mathcal{S}$: π
Probability distribution over $\mathcal{Q} \times \mathcal{S}$: τ

A train data sample for fully supervised case: (x_i, y_i)
A train data sample for weak supervision case: (q_i, s_i)

2.2 Problem Statement:

Let the universe of inputs be $\mathcal{X}$, and of targets be $\mathcal{Y}$ Let $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be the loss function. Let the underlying joint distribution on $\mathcal{X} \times \mathcal{Y}$ be ρ. We want to find a function $f : \mathcal{X} \to \mathcal{Y}$ that minimises the risk, given this l and ρ

$$f = \arg\min_{f'} \mathcal{R}(f', \rho) \text{ where } \mathcal{R}(f', \rho) = \mathop{\mathbb{E}}_{(X,Y)\sim\rho}[l(f'(X), Y)]] \tag{1}$$

However we do not have direct access to distribution ρ. Instead each training example is given in the form $(q_i, s_i), q_i \in \mathcal{Q} \subseteq 2^{\mathcal{X}}, s_i \in \mathcal{S} \subseteq 2^{\mathcal{Y}}$. Training data is sampled from distribution τ over $\mathcal{Q} \times \mathcal{S}$.
ρ and τ are clearly related, and we formalise this notion with a concept of 'eligibility'. We say ρ is eligible for τ iff:
There exists a distribution, π over $\mathcal{X} \times \mathcal{Y} \times \mathcal{Q} \times \mathcal{S}$ such that:

1. τ is a marginal of π over $\mathcal{Q} \times \mathcal{S}$
2. ρ is a marginal of π over $\mathcal{X} \times \mathcal{Y}$
3. If $x \notin q$, then $\pi(x, Y, q, S) = 0$.
4. If $y \notin s$, then $\pi(X, y, Q, s) = 0$.

If ρ is eligible for τ, we denote this as: $\rho \vdash \tau$. Thus our goal is to retrieve f, given τ

3 Literature Survey of Techniques

1. Partial Label Learning: Summarised in Table 1 and 2
2. Multi Instance Learning: Summarised in Table 3
3. Noisy Label Learning: 'Learning from Noisy Labels with Deep Neural Networks: A Survey' provides a detailed summary of Noisy Label Learning Methods [11]. We use their table in Table 4.
4. 1 of Many Learning: We refer to the paper [8] for 1oML Learning. The paper described problems like sudoku which have many correct solution, and where our goal to be able to output any one of those correct solutions. Given an input unsolved problem and number of correct solutions, the paper proposes:
   (a) Naive Loss: Computing Loss on each solution and then taking an average
   (b) Min Loss: A greedy based approach where loss is computed on each solution, and then the minimum one is picked.
   (c) SelectR: Reinforcement Learning based method where an RL Agent picks a solution from the solution set which is 'best-suited' for learning, and passes it to the prediction network. It's reward is the improvement in prediction probability of the prediction net.

4 Datasets

Partial Label Learning

1. Discrete Output Space
   (a) *lost*: 1122 examples, 108 input features, 16 classes
   Video footage from TV Shows Lost and CSI were taken, and passed through

Year	Authors	Technique	Brief Summary
2003	Jin + Ghahramani	EM	EM over log likleihood: E step - Determine a distribution over labels, M step - Minimise loss function.
2008	Nguyen + Caruana	PL-SVM	Modify SVM loss to include for partial labels. Do an EM - alternate ground-truth label identification and margin maximization. $L(f(x), S) = max[0, 1 - maxf(x, y) - maxf(x, y^{'})]$
2017	Zhang + Yu		Max Margin Loss, but directly maximize the margin between the ground-truth label and all other labels. $L(f(x), S) = f(x, y) - maxf(x, y^{'})$
2015	Hullermeier		Generalised loss: $L(f(x), S) =$ minimum loss btw $f(x)$ and each $y \in S$
2006	Hullermeier + Beringer	PL-KNN	KNN: Assign label which is most frequently found in K nearest neighbours. Decision Tree: Instead of entropy, use a 'potential entropy' by picking most frequent label to label a set
2011	Cour	CLPL	Averaging based method: For each PLL training example (x, S), generate positive example as average of positive labels ($\in S$), and generate one negative example for each negative label ($\notin S$). Then apply any existing learning method.
2016	Zhang + Zhou + Liu	Feature Aware Disamb	Exploit smoothness in feature space
2012	Liu + Dietterich	LSB-CMM	LSB-CMM model, trained with Variational EM
2015	Zhang + Yu	IPAL - Instance based PLL	IPAL: A KNN method, give weights to neighbours as a weighted graph W. Start with an assumption that all labels are of equal weights. Use an iterative label propagation procedure to disambiguate the correct label based on nearest neighbours.
2018	Wu + Zhang	PALOC	Adapt Binary (one v. one) Decomposition for PLL

Table 1: Partial Label Learning Literature Survey

Year	Authors	Technique	Brief Summary
2018	Feng + An	Self Guided Re-training	Apply a model on the examples. Pick those above a confidence threshold and mark as ground truth. Repeat till no PL examples left. This is done via an infinity regularisation norm
2019	Xu + Jiaqi + Geng	PL-Label En-hance-ment	Define a GLD - Generalized Label Distribution for each training example to capture degree of relevance/irrelevance of labels

Table 2: Partial Label Learning Literature Survey (contd)

face detection software. These faces were then given a partial labelling of all characters that appear in the screenplay of the scene [3]

(b) *MSRCv2*: 1758 examples, 48 input features, 23 classes
Microsoft Research Cambridge's dataset is made up of images which have been segmented. Each segment has been labelled. This is transformed into a partial label dataset by giving a particular segment the labels of all segments in the image it belongs to [7].

(c) *BirdSong*: 4998 examples, 38 input features, 13 classes
There are 10 second recordings of birds available. These are broken into syllables. Each syllable has been uttered by a single bird, and that will be the gold label. The partial set is all birds that also sing during the 10 second period to which the syllable belongs [1].

(d) *Soccer Player*: 17472 examples, 279 input features, 171 classes
Soccer Player images were collected from newspaper articles, together with their captions. The faces of players were selected out and image transformations, followed by PCA were applied. Each face is given the labels of all players that appear in the caption [12].

(e) *Yahoo! News*: 22991 examples, 163 input features, 219 classes
Similar to Soccer Player, it is an image-caption generated dataset [6]

2. Structured Output Space

(a) Sudoko Dataset: Input is a sudoku board, partially filled. Label is the fully filled, correct solution to the board. The dataset contains boards with have unique solutions. We have created several

We have created several additional artificial datasets using these datasets as a base, but these will be described in detail in later sections.

Year	Authors	Brief Summary
1997	Dietterich	Axis Parallel Rectangles
1998	Maron	Diverse Density: For Standard MIL - Positive Bag means exactly one positive x, Negative Bag means all negative x. Pick a representatve instance t, which is 'dense' - close to more instances from positive than negative bags; and 'diverse' - close to at least one instance from each positive bag. Model 'closeness' as a distance metric
2002	Zhang + Goldman	EM-DD: E step - Pick the most likely instance from each bag for the label using hypothesis h; M step - Using these most likely instances, find a new hypothesis h'to maximise DD(h'). M step is from Maron.
2004	Xu + Frank	MI-Boosting
2013	Doran + Ray	SMILe: Shuffled MIL - Use resampling to create new bags. This creates additional constraints for learning
2007	Andrews	MI-SVM: Max margin is defined as maximum distance from the separating hyperplane of all instances in the positive bag. This is a Mixed Integer Program, so use EM to speed up: E Step: Identify most important instance in the bag, M Step: Train the SVM classifier
2015	Zeng + Liu + Chen	CNN: Assumes at least one sentences expresses the relation. Selects only one most likely sentence using max pooling
2016	Lin	Sentence Level Attention: Extract semantics with a CNN. But then give attention weights to each sentence, lowering the weight of noisy sentences
2017	Ji	Improve attention by exploiting external knowledge about the entity pair of the relation
2019	Qin + Xu + Wang	RL: Train a policy network to identify false positives and move them from the positive set into the negative set. State: Current sentences + Sentences removed in the past Action: Retain or remove a 'sentence'? Is it a false positive? Reward: Change in Relation Classifier accuracy (F1 score)
2018	Feng	RL: Instance Selector State: Current sentence, already selected sentences Action: Select the current instance or not? Reward: Prediction Probability

Table 3: Multi Instance Learning Literature Survery

Category	Summary	Techniques
Robust Loss Function	Modify the loss function to be robust to noise	Robust MAE, Generalized Cross Entropy, Symmetric Cross Entropy, Curriculum Loss
Robust Architecture	Modify the architecture, for example by adding a transition matrix at the top to simulate noise.	Webly Learning, Noise Model, Dropout Noise Model, S–model, C–model, NLNN, Probabilistic Noise Model, Masking, Contrastive-Additive Noise Network
Robust Regularization	By preventing overfitting and encouraging a highly generalised model robustness to noise improves	Adversarial Training, Label Smoothing, Mixup, Bilevel Learning, Annotator Confusion, Pre-training
Loss Adjustment	Assign weights to samples as well as noisy labels within samples	Backward Correction, Forward Correction, Gold Loss Correction, Importance Reweighting, Active Bias, Bootstrapping, Dynamic Bootstrapping, D2L, SELFIE
Sample Selection	Select out samples more likely to be correct and train on them. Typically done in collaboration with another network.	Decouple, MentorNet, Co-teaching [78], Co-teaching+, Iterative Detection, ITLM, INCV, SELFIE, SELF, Curriculum Loss
Semi-supervised Learning	Convert a noisy labelling problem to semi-supervised one	Label Aggregation, Two-Stage Framework, SELF, DivideMix

Table 4: Noisy Label Learning Literature Survey

5 Loss Functions

We experiment with different loss functions, and report results in 5 and 6

5.1 Naive Loss

A naive way to treat the problem is to assume that each label in the partial set is right, find the supervised loss under that assumption, and average out this loss per data point. This gives rise to a baseline 'naive' loss. [8]

$$L_{naive} = -\frac{\sum_{j \in S} \log(p^j)}{|S|} = -\frac{\log \prod_{j \in S} (p^j)}{|S|} \tag{2}$$

It can thus be interpreted as an "AND" of all the labels in the partial set (sum of log is log of products). Herein lies the weakness of a naive loss - it forces every label in S to be 'correct' - instead of just one.

5.2 CC Loss

CC Loss is log of sum of probabilities for labels in the partial set. It can be interpreted as an "OR" function, as opposed to "AND" in Naive Loss.

$$L_{cc} = -\log(\sum_{j \in S} p^j) \tag{3}$$

We borrow the term CC (Classifier Consistent) from 'Provably Consistent PLL'. [4]

5.3 Min Loss

Min Loss picks one label in the partial set, and attempts to minimise the loss wrt it.

$$L_{min} = \min_{j \in S} -\log(p^j) \tag{4}$$

It can be interpreted as a greedy approach which can often pick a wrong label initially, and then get caught in a suboptimal local optimum. [8]

5.4 IexplR Loss

[4] We derive IexplR Loss as a lower bound for CC Loss via Jensen's Inequality.

$$L_{IexplR} = -\sum_{j \in S} q^j \log(p^j); \quad q^j = \frac{p^j}{\sum_{j \in S} p^j} \tag{5}$$

It is important to note that q_i are only weights, and are thus detached (no back-propagation is done through them).

5.5 Regularized CC Loss: Exploiting Peaky Distribution

Partial Label Learning has a strict constraint that has not yet been captured by any of the mentioned losses - it allows only one label in the partial set to be the true label. This implies that the distribution over the labels must be peaky - only one label should have a high probability, the rest should be close to zero.
Naive, CC and Iexplr are unable to capture this, and will allow even a uniform distribution. We thus must introduce a constraint to these losses.

$$\text{minimise } L_{cc}, \text{ s.t.} \max_{j \in S} p^j = 1 \tag{6}$$

Knowing that log is an increasing function:

$$\equiv \text{minimise } L_{cc}, \text{ s.t.} \max_{j \in S} \log p^j = 0$$

This can be rewritten by using a penalty term λ

$$\equiv \text{minimise } L_{cc} + \lambda.(- \max_{j \in S} \log p^j)$$

$$\equiv \text{minimise } L_{cc} + \lambda.(\min_{j \in S} - \log p^j)$$

$$\equiv \text{minimise } L_{cc} + \lambda.L_{min} \tag{7}$$

We call this our regularized loss, and experiement with various values of λ.

5.6 SVM Loss

Let us first define

$$L_{Hinge}(y) = H(y) = max(0, 1 - y) \tag{8}$$

We now try to maximize the gap between the maximum correct and maximum incorrect label. In this case, we use logits, instead of probabilities. This can be understood intuitively from the definition of PLL. There is only one correct label, thus we must take max logit in the partial set as one end point. We also know the label is guaranteed to be in the partial set S. Thus the other end of the gap must be the max point of the complement set.

$$L_{SVM} = H(\max_{j \in S} s^j - \max_{j \notin S} s^j) \tag{9}$$

Note: We use scores (prediction net output before softmax), instead of probabilities.

Architecture	3layer				1layer			
Dataset	lost	MSRCv2	BirdSong	Soccer Player	lost	MSRCv2	BirdSong	Soccer Player
Fully Supervised	81.96	67.14	75.59	64.63	82.86	61.89	74.37	56.94
Naive Loss	54.91	38.34	61.22	53.29	53.93	34.97	62.48	48.54
Min Loss	68.75	49.26	71.56	54.11	48.39	41.66	67.52	51.31
CC Loss	74.64	50.23	71.5	55.64	67.86	45.09	70.06	53.84
IexplR Loss	74.64	50.57	71.22	55.47	67.86	44.91	70.06	53.8
Regularized ($\lambda = 0.1$)	72.95	50.91	70.48	55.2	64.73	44.46	70.28	53.44
Regularized ($\lambda = 0.2$)	71.52	48.69	71.72	55.27	63.04	43.31	70.14	53.23
Regularized ($\lambda = 0.5$)	73.21	48.29	70.68	54.77	59.73	42.91	69.4	52.9
Regularized ($\lambda = 1.0$)	68.84	50.29	69.74	54.76	57.95	43.54	69.14	52.99
SVM Loss	65.893	52.286	71.182	52.988	-	-	-	
Cour Loss	71.339	46.114	68.778	52.387	-	-	-	

Table 5: Loss Function Experiemnts: Real Test Accuracies

5.7 Cour Loss

We define this loss from Cour's paper [3]. The paper transforms a single example into $|\mathscr{Y} - S| + 1$ samples. We create one positive sample as the average of all the points in the partial set S, and we create one negative sample per point in the complement set $\mathscr{Y} - S$. We compute Hinge Loss on each new sample, and take a sum.

$$L_{Cour} = H(\frac{1}{|S|}.\sum_{j \in S} s^j) + \sum_{j \notin S} H(-s^j) \tag{10}$$

Note: We use scores (prediction net output before softmax), instead of probabilities.

Architecture	3layer				1layer			
Dataset	lost	MSRCv2	BirdSong	Soccer Player	lost	MSRCv2	BirdSong	Soccer Player
Fully Supervised	99.97	87.99	89.59	99.6	97.28	75.22	76.23	81.18
Naive Loss	99.84	86.74	86.66	99.88	86.28	69.21	75.41	78.48
Min Loss	99.54	89.53	96.23	99.95	97.28	79.23	81.55	98.19
CC Loss	99.96	90.16	95.11	99.92	99.17	81.37	81.69	99.38
IexplR Loss	99.96	88.98	95.29	99.95	99.17	81.35	81.64	99.34
Regularized ($\lambda = 0.1$)	99.84	90.48	93.98	99.92	97.43	80.31	81.44	99.32
Regularized ($\lambda = 0.2$)	99.58	87.13	95.28	99.94	97.38	79.8	81.33	99.59
Regularized ($\lambda = 0.5$)	99.82	87.74	94.32	99.94	96.94	79.09	81.16	99.35
Regularized ($\lambda = 1.0$)	98.49	89.1	95.49	99.91	96.5	79.29	80.67	99.93
SVM Loss	99.107	89.343	92.993	94.699	-	-	-	-
Cour Loss	99.978	88.079	96.881	99.359	-	-	-	-

Table 6: Loss Function Experiemnts: Surrogate Train Accuracies

6 CC Loss - IexplR Loss Relation

Claim: IExplR is a proxy (lower bound) for CC Loss.

$$L_{cc}(\theta) = \log(p_1(\theta) + p_2(\theta) + \ldots + p_k(\theta)) \tag{11}$$

Let us introduce a probability distribution, $q_1, q_2 \ldots, q_k$. Divide and multiply each p_i with q_i.

$$\begin{aligned}
L_{cc}(\theta) &= \log\left(p_1(\theta) + p_2(\theta) + \ldots + p_k(\theta)\right) \\
&= \log(q_1 * p_1(\theta)/q_1 + q_2 * p_2(\theta)/q_2 + \ldots + q_k * p_k(\theta)/q_k) \\
&= \log\left(\sum_i q_i * p_i(\theta)/q_i\right) \\
&= \log\left(\mathbb{E}_q[p(\theta)/q]\right) \\
&\geq \mathbb{E}_q[\log p(\theta)/q] \\
&= \sum_i q_i \log\left(p_i(\theta)/q_i\right)
\end{aligned}$$

The lower bound holds for any choice of probability distribution q. We select q such that the bound is tight for $\theta = \theta_t$, *i.e.*, q is nothing but normalized $p(\theta_t)$, *i.e.*, $q_i = p_i(\theta_t)/\sum_i(p_i(\theta_t))$. We get:

$$\begin{aligned}
L_{cc} &\geq \frac{1}{\sum_i p_i(\theta_t)}\left(\sum_i p_i(\theta_t) log(p_i(\theta))\right) + Constant \\
&= \frac{1}{\sum_i p_i(\theta_t)} L_{IExplR}(\theta) + Constant
\end{aligned} \tag{12}$$

Where $L_{IExplR}(\theta)$ is nothing but the IExplR loss, defined as:

$$L_{IexplR}(\theta) = p_1(\theta_t)\log p_1(\theta) + p_2(\theta_t)\log p_2(\theta) + \ldots + p_k(\theta_t)\log p_k(\theta) \tag{13}$$

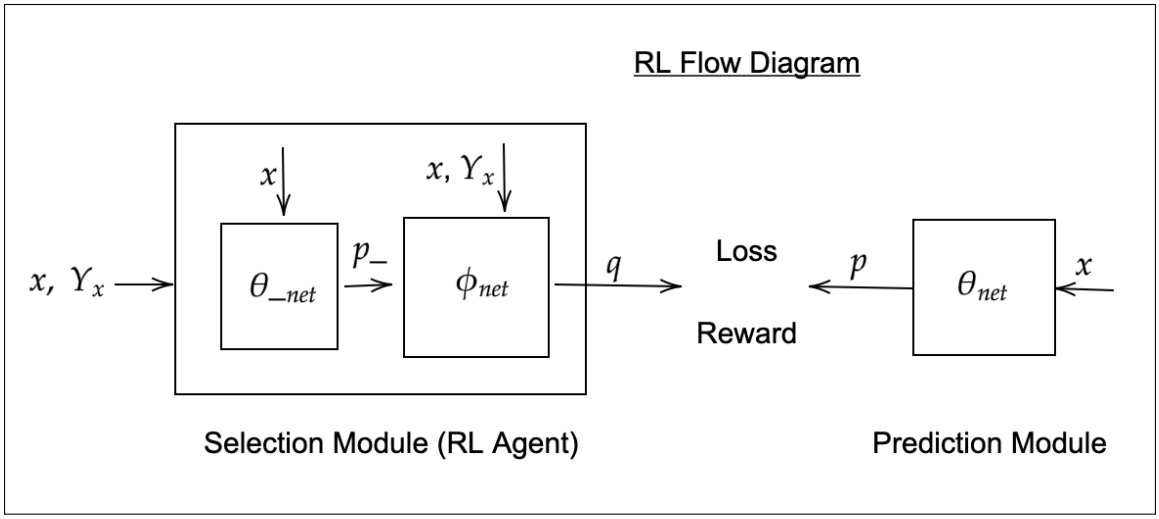
7 Reinforcement Learning

We use SelectR from previous work done by this group in the domain on 1 of Many Learning [8].
For a given sudoku problem, it is possible that strictly one, or more than one solutions exist. It is found that existing learning techniques perform well on single solution boards, but poorly on multi solution boards (assuming naive loss is used).
To overcome this 2 approaches were propsed:

1. Greedy Min Loss: Similar to our min loss, the network attempts to minimise loss on the best (lowest loss) board in the solution set.

2. Reinforcement Learning: Here an RL agent is used, whose job is to select out the 'best' board from the solution set for the purpose of training. Reward is the prediction probability of the prediction network.

We use a similar idea: Now our RL agent must choose the best label from the partial set, instead of the multiple solution set.



7.1 SelectR: Linear Action Space

We build on the idea from 1oML for sudoku. The job of the RL agent is to pick the best label from the partial set for the prediction net to use as the gold label.
RL Action: Pick best label out of the partial set for training. We call this technique RL Linear as the Action Space is linear in the size of the partial set.
RL Reward: We use the probability output of the prediction network.
Since the Action Space is linear, we can compute both Loss and Reward in Expectation. Let the output for j^{th} label of the prediction network be p^j and RL agent be q^j

$$\text{Loss (for Prediction Network)} = \sum_{j \in S} -q^j.log(p^j) \tag{14}$$

$$\text{Reward (for Selection Network)} = \sum_{j \in S} q^j.p^j \tag{15}$$

7.2 SelectR: Exponential Action Space

We take our idea from Relation Extraction [5] [9]. This is an example of Multi Instance Learning: We have a set of x, labelled by a single y.

Our goal is to identify the relation expressed in a sentence. We are provided with a bag of sentences mentioning an entity. The bag is labelled by a relation, but only some sentences in the bag express the relation. We must sift out false positives from the bag in order to learn well.
Feng [5] uses an RL agent for this purpose, and determines at each step, whether to include or exclude sentences. The reward is the prediction probability.

In Partial Label Learning, we have one x, but multiple candidate y. We use the idea of removing false positives by selecting a subset of the given partial set.
RL Action: Pick a subset of the partial set and give it to the prediction network to train. We call this technique RL Exponential as the action space is exponential in the size of the partial set.
RL Reward: We use the probability output of the prediction network.
Given an exponential action space, we must use REINFORCE, for loss and reward. Let the output for j^{th} label of the prediction network be p^j and RL agent be q^j. Let a be the action vector: where a^j is sampled to be 1 with probability q^j, 0 otherwise

$$\text{Loss (for Prediction Network)} = \text{CC LOSS} = -log(\sum_{j \in S} p^j.a^j) \tag{16}$$

$$\text{Reward} = a * p + (1 - a) * (1 - p)$$

$$\text{Log Action Probability} = a * \log(q) + (1 - a) * \log(1 - q)$$

$$\text{Reinforce Reward} = \text{Log Action Probability} * \text{Reward} \tag{17}$$

7.3 Weak Prediction Network

We can see in Table 6, that a 3 layer model attains a surrogate train accuracy close to 100%. This would imply that the prediction model has already learnt as much as it could, given the data. The RL agent will therefore have no further benefit to provide during training.
We therefore experiment to see if RL helps in the case when the prediction net under-fits to the training data - and reduce the number of layers in the prediction net to 1.

7.4 Pretraining Prediction Network

We experiment with different pre-training strategies. We can pretrain both the Prediction Network and the RL Agent's Selection Network.
Pretraining Prediction Network simply involves training using CC LOSS.
Pretraining the Selection Network involves RL training of the Selection Network with a less powerful 1-layer Prediction Network. The motivation behind this is similar to the one expressed in Section 7.4 - if a Selection Network is to gain insight, it can only do so if the Prediction Network is unable to overfit to the training data.

7.5 Pretraining Selection Network: Using a Weak Prediction Network

We experiment with different pre-training strategies. We can pretrain both the Prediction Network and the RL Agent's Selection Network.
Pretraining Prediction Network simply involves training using CC LOSS.
Pretraining the Selection Network involves RL training of the Selection Network with a less powerful 1-layer Prediction Network. The motivation behind this is similar to the one expressed in Section 7.4 - if a Selection Network is to gain insight, it can only do so if the Prediction Network is unable to overfit to the training data.

7.6 Exploiting Predictive Power of Selection Network

We note that the Selection Network also has some amount of predictive power, it takes in x and the partial set S and outputs a probability distribution over label space. We thus try to use it to make predictions. Let us call outputs of Prediction Network p and Selection Network q. We try using:

1. Only q
2. Average of p and q
3. Most confident between p and q (i.e. compare max probabilities btw Prediction and Selection and use the higher one for prediction.

None works as well as using the Prediction network alone. So we continue with the Predction Network.

7.7 RL Experiments on Synthetic Data: Varying the Size of Partial Set

We experiment to see if Reinforcement Learning shows benefits after a given size of partial set. Thus we create synthetic data. We take the original x and partial set S, and add n additional labels to the set, picked at random. We then compare cc loss with Linear RL with pretraining of P and Q.
The results are reported in Tables 8 and 8. Again we find, cc loss outperforms linear RL almost universally.

7.8 RL for PLL in Structured Output Space

We create artificial PLL datasets over our Sudoku dataset. We create a partial set containing the correct solution, 4 artifically created incorrect solutions:

1. Shuffle: We shuffle the positions of all the known numbers on the board

Architecture	3layer				1layer			
Dataset	lost	MSRCv2	BirdSong	Soccer Player	lost	MSRCv2	BirdSong	Soccer Player
Min Loss	68.75	49.26	71.56	54.11	48.39	41.66	67.52	51.31
CC Loss	74.64	50.23	71.5	55.64	67.86	45.09	70.06	53.84
Linear RL P100%	68.93	45.89	69.74	52.76	67.23	42.57	69.34	51.27
Linear RL P70%	64.29	39.6	67.25	50.89	59.46	38.91	66.25	49.52
Linear RL P80%	64.29	38.97	66.21	51.25	59.46	38.91	66.21	49.52
Linear RL P90%	64.2	41.83	67.03	51.19	61.61	40.4	65.57	49.47
Linear RL P100% + Q	72.59	45.77	70.32	53.78	-	-	-	-
Linear RL P70% + Q	67.5	44.46	65.75	51.93	-	-	-	-
Linear RL P80% + Q	67.5	44.51	67.37	52.07	-	-	-	-
Linear RL P90% + Q	70	44.63	67.39	51.89	-	-	-	-
Linear RL Q	54.02	44	66.93	51.31	-	-	-	-
Exponential RL P100%	78.39	54.74	68.75	57.00	-	-	-	-

Table 7: Reinforcement Learning compared to CC and Min Loss.
P indicates pretraining of Predicition Network, Q indicates pretraining of Selection Network, Percent is the percent of maximum to which Prediction Network is trained

lost	Original	2	4	6	8	10	12	14
CC Loss	74.64	69.02	63.39	55.98	45.62	35	22.86	15.36
Linear RL P100% + Q	71.34	65.8	60.45	52.77	45.71	32.23	24.11	12.5
MSRCv2	Original	2	4	8	12	16	20	22
CC Loss	50.23	47.26	46.17	41.94	36.91	30.29	17.03	7.03
Linear RL P100% + Q	47.77	45.2	43.03	36.91	30.97	25.89	14.34	10.51
BirdSong	Original	1	2	4	6	8	10	12
CC Loss	71.56	69.52	68.92	67.84	66.61	61.8	57.19	50.16
Linear RL P100% + Q	71.52	69.92	67.6	65.37	62.77	58.08	52.77	39.14
Soccer Player	Original	2	4	8	16	32	64	128
CC Loss	55.64	53.59	51.73	51.05	49.17	49.11	48.62	36.12
Linear RL P100% + Q	54.18	52.47	51.39	50.19	49.16	48.98	48.56	48.95

Table 8: 3 Layer Prediction Network: Increasing Partial Set Size

lost	Original	2	4	6	8	10	12	14
CC Loss	67.86	62.5	53.04	44.38	36.34	27.5	18.84	9.2
Linear RL P100% + Q	66.96	61.34	51.25	44.55	35.45	25.54	20.09	10.62
MSRCv2	Original	2	4	8	12	16	20	22
CC Loss	45.09	43.2	43.03	36.51	32.46	27.77	15.66	12.29
Linear RL P100% + Q	43.26	41.66	40.97	35.37	29.83	23.83	16.57	7.89
BirdSong	Original	1	2	4	6	8	10	12
CC Loss	70.06	71	69.18	65.89	64.71	60.58	53.07	46.45
Linear RL P100% + Q	70.76	70.7	66.71	65.63	60.46	50.68	52.44	38.14
Soccer Player	Original	2	4	8	16	32	64	128
CC Loss	53.84	52.18	50.95	50.02	48.76	47.64	46.7	36.62
Linear RL P100% + Q	52.78	51.41	50.15	49.42	48.45	48.2	48.22	47.93

Table 9: 1 Layer Prediction Network: Increasing Partial Set Size

Dataset	Shuffle	Switch	Flip 0.2	Flip 0.4	Flip 0.6	Flip 0.8
Fully Supervised	90.14	90.14	90.14	90.14	90.14	90.14
Min Loss	86.82	90.26	89.12	88.7	88.86	87.76
CC Loss	86.6	89.24	89.23	89.3	87.58	89.04
Linear RL	86.78	89.84	89.6	88.88	89.18	87.66

Table 10: RL compared to CC and Min Loss on Structured Datasets

2. Switch: We pick 8-10 pairs of known points and switch their position
3. Flip ϵ: For each of the known points, we flip it to a random new value with probability ϵ

Results are reported in Table 10.

8 Generative Modelling

We experiment with Generative Models of varying degrees of complexity as follows: 'Provably Consistent Partial-Label Learning' [4] gives a theoretical derivation for CC (Classifier Consistent) Loss.
Given instances of the form $\{x_i, S_i\}$, where S_i is the set of labels out of which one is the true label y_i. The paper assumes that the probability of observing a particular partial set is randomly uniform. We must remember however, that a partial set can only be observed if the true label is contained in it. From these facts:

$$P(S_i/x_i, y_i^j) = \begin{cases} \frac{1}{2^{n-1}} & y_i^j \in S_i \\ 0 & y_i^j \notin S_i \end{cases} \tag{18}$$

where n is the size of the label space.
Now we maximise log likelihood of seeing this data : $\sum_i \log P(S_i/x_i, \theta)$
Conditioning over all possible values of true labels:

$$= \sum_i \sum_j \log P(y_i^j/x_i, \theta).P(S_i/x_i, y_i^j, \theta) \tag{19}$$

Putting (6) in (7): $= \frac{1}{2^{n-1}} \sum_i \sum_{j \in S_i} \log P(y_i^j/x_i, \theta) \equiv L_{CC}$
Ignoring the constant term we have arrived at cc loss.
We however can break the uniform set assumption. Our new loss function will be (We ignore the instance variable i for simplicity):

$$L_{Generative} = \log \sum_{j \in S} P(y^j/x).P(S/x, y^j) \tag{20}$$

The left term is our standard output of the prediction network $P(\theta)$, which takes in an x, and gives out a distribution over the label space.
The right term will now be computed by a generative model $G(\phi)$.
We can take an IexplR style bound on this:

$$L_{GenerativeIexplr} = \sum_{j \in S} q^j j \log P(y^j/x).P(S/x, y^j) \tag{21}$$

where q^j are the normalised arguments of log.

8.1 Y-Matrix Generative Model: Naive Bayes Dependence on Y only

We start our experiments by making two simplifying assumptions:

1. Naive Bayes Assumption: Partial Set S can be generated from the label set by a Naive Bayes Model. Thus:

$$P(S/x, y^{j=gold}) = \prod_{y^k \in S} P(y^k/x, y^{j=gold}). \prod_{y^k \notin S} (1 - P(y^k/x, y^{j=gold})) \tag{22}$$

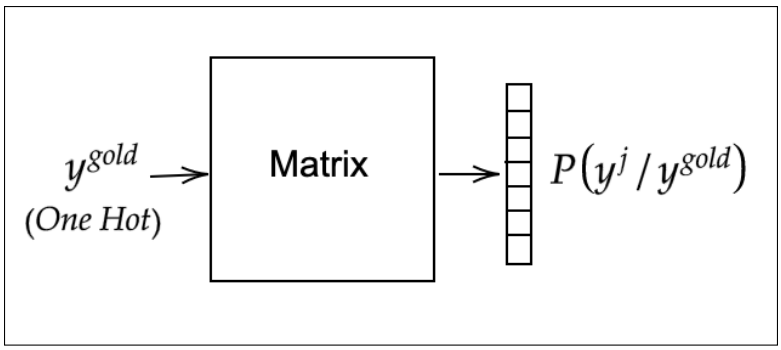
2. Independence of x Assumption:

$$P(S/x, y^{j=gold}) = P(S/y^{j=gold}) \tag{23}$$

Combining 1. and 2., we deduce that our generative model is simply a matrix M of size n x n, where $M_{jk} = P(y^k/y^j)$.

$$P(S/x, y^{j=gold}) = \prod_{y^k \in S} P(y^k/y^{j=gold}). \prod_{y^k \notin S} (1 - P(y^k/y^{j=gold})) \tag{24}$$

Note: A key observation to make here is that $P(y^j/y^{j=gold}) = 1$, since a partial set **must** contain y^j, given that y^j is gold. This matrix thus has 1's on the diagonal.



8.1.1 Artifical Datasets

We create artificial datasets, on top of our original datasets. For each dataset, we construct an artificial matrix M, where $M_{jk} = P(y^k/y^{j=gold})$, and where the diagonal is set to 1. We then take each x and gold y from the original dataset, and sample using this matrix to create a partial set. The datasets are named after the techniques used to create M:

1. one: For each i, set $M_{ii} = 1$. Pick another label 'j' at random, and set $M_{ij} =$ Uniform[0.3, 0.7]
2. two: For each i, set $M_{ii} = 1$. Pick two other labels 'j' at random, and set $M_{ij} =$ Uniform[0.3, 0.7]
3. flip: Set $M_{ii} = 1$. For each remaining j, flip a coin with head probability = 0.2:
   (a) If Heads: Set $M_{ij} =$ Uniform[0.5, 0.8]
   (b) If Tails: Set $M_{ij} =$ Uniform[0.08, 0.1]

8.1.2 Training Techniques

We report the results of our joint training. We run experiments on a clean dataset (fully supervised learning), as well as baseline cc and iexplr losses on partial datasets. We then run our generative model training. We have tried different variants:

1. Standard: Simple joint training of the P and G networks.
2. Freeze Diagonal: Joint training of P and G network, but we artificially force diagonal entries, i.e. $G(y^j/y^j) = 1$
3. Data Statistics: Instead of learning the G Network during training, we estimate it using data statistics. We need to estimate $P(y^k/y^j = \text{gold})$. However we do not yet have access to gold data. Thus we must make an estimate of gold labels by using a baseline Prediction Network. Once this is done, we can compute $P(y^k/y^j = \text{gold}) = \frac{\text{Times } y^k \text{ co-occurs with } y^j, \text{ when } y^j \text{ is gold}}{\text{Times } y^j \text{ is gold}}$

As a baseline, we also run Artificial Matrix, by artificially using the M we used to create the dataset as our Generative Model.

8.1.3 Key Observations

1. Fix Diagonal is almost uniformly the best method on artificial datasets, outperforming cc baseline upto 6 points.
2. Generative Modelling methods do not perform well on real world datasets

Dataset	MSRCv2 one	MSRCv2 two	MSRCv2 flip	MSRCv2	BirdSong one	BirdSong two	BirdSong flip	BirdSong
Fully Supervised	66.63	66.63	66.63	65.94	76.07	76.07	76.07	76.11
CC Loss	66.11	61.77	54.86	49.60	74.05	72.91	72.85	71.92
Iexplr Loss	65.23	61.83	55.26	50.46	73.87	73.09	72.36	72.16
Artificial Matrix	66.23	64.74	60.57		75.63	75.03	74.43	
Standard .	66.97	62.86	59.54	45.37	75.13	74.57	73.47	70.02
Fix Diagonal	67.77	65.14	60.17	44.74	76.37	75.43	73.77	65.03
Data Statistics	65.54	64.63	56.34	49.20	75.21	74.13	72.71	71.18

Dataset	Soccer Player one	Soccer Player two	Soccer Player flip	Soccer Player
Fully Supervised	65.81	65.81	65.81	65.82
CC Loss	63.99	62.56	51.35	55.53
Iexplr Loss	63.78	62.20	51.26	55.93
Artificial Matrix	63.49	62.50	55.82	
Standard .	63.17	60.91	54.96	55.31
Fix Diagonal	63.05	62.00	55.21	50.53
Data Statistics	63.42	61.67	53.31	56.48

Table 11: Generative Models for Y Dependence on real and synthetic data

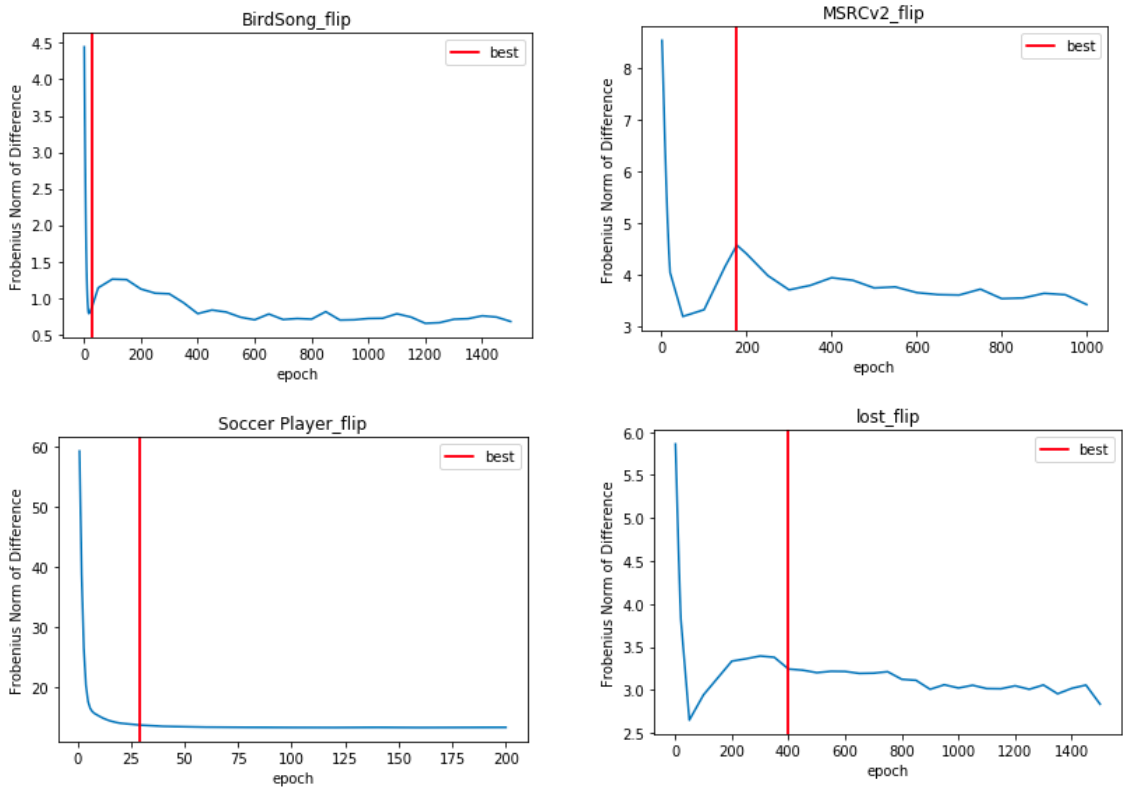


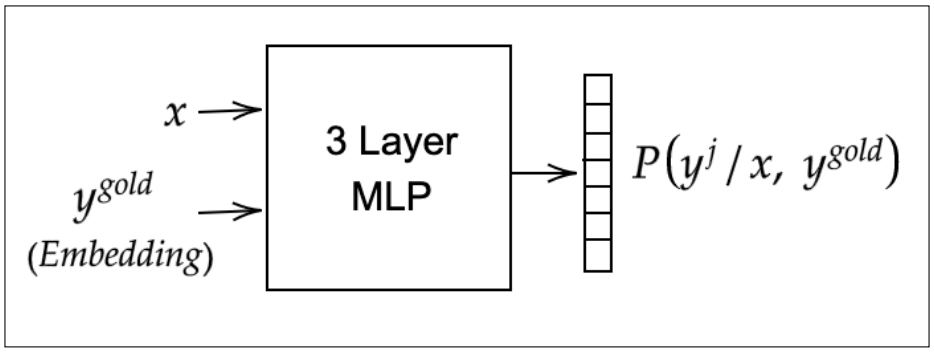
Figure 1: Frobenius Norm Difference btw target and learnt matrix during training

3. Fix Diagonal often even outperforms Artificial Matrix, meaning that our generative model learns more information that even if we had full information about the data generation process.

8.2 XY Generative Model: Naive Bayes Dependence on X and Y

We remove assumption 2. from the previous section, and constrain ourselves to only a Naive Bayes assumption, now with dependence on x.
We use Equation (12) as is:

$$P(S/x, y^{j=gold}) = \prod_{y^k \in S} P(y^k/x, y^{j=gold}). \prod_{y^k \notin S} (1 - P(y^k/x, y^{j=gold}))$$



8.2.1 Artificial Datasets

We again create artificial datasets, on top of our original datasets, incorporating dependence on x:

1. Top: We train a network on the partial data, but stop before saturation. We choose to stop when average train confidence exceeds 95%. We then use this trained network as a noisy annotator. For each x, we choose the top 3 predictions made by the annotator, and add it to the partial set. We then add the gold label, if not present.

2. Sample: We train a network as in Top. But now we interpret the output of the network $P(y^k/x, \theta)$ as probability of including the label in the partial set and use it to sample i.e. for each x, we include label k in the partial set with probability $P(y^k/x, \theta)$.

8.2.2 Key Observations

We analyse the learnt generative model for MSRCv2, fold 0: We find the learnt distribution is **highly skewed**.
The average value of $P_{max} = \max P(S/x, y^{gold}) = 0.948$. 67.35% of examples have $P_{max} > 0.99$

1. Sensitivity of Product:

$$P(S/x, y^{j=gold}) = \prod_{y^k \in S} P(y^k/x, y^{j=gold}). \prod_{y^k \notin S} (1 - P(y^k/x, y^{j=gold}))$$

   This product of terms is highly sensitive to small terms. If even a single probability is wrong, it will impact the entire outcome.

2. Size of Partial Set: Differs from example to example. This could impact product. Average Size of Set ($P_{max} > 0.99$) = 2.81 v/s ($P_{max} < 0.99$) = 3.88

We try to overcome these by an LSTM model which omits the naive bayes assumption, and also implicitly accounts for all sizes of partial sets.

8.3 LSTM Model: Universal Approximator

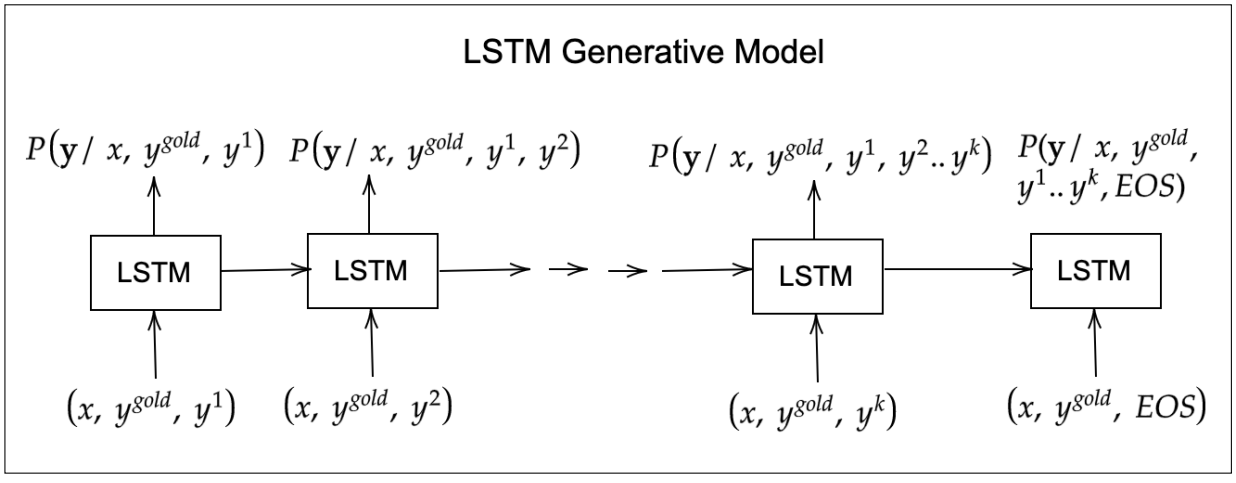
In this section we remove our Naive Bayes Assumption as well, and attempt to directly model $P(S/x, y^j = gold)$.
First let us define a bijective function $M : \mathscr{Y} \rightarrow \mathbb{N}$. Given a partial set $S = \{y^{i1}, y^{i2}..y^{ik}\} \subset \mathscr{Y}$, we map it to the sequence $y^{i1}, y^{i2}..y^{ik}$, s.t. $M(y^{i1}) < M(y^{i2})$ and so on.
We now use an LSTM model to predict this sequence.
At the first step, we feed in $\{x, y^{gold}, y^{gold}\}$. We compute cross entropy loss of the output wrt y^{i1}.
At then next step, we feed in $\{x, y^{gold}, y^{i1}\}$. We compute cross entropy loss of the output wrt wrt y^{i2}.
This goes on till we feed in $\{x, y^{gold}, y^{ik}\}$. We now compute cross entropy loss wrt an End of Sequence Token.
The output at stage j is $P(y^{ij}/x, y^{gold}, y^{i1}, y^{i2}...y^{i(j-1)})$.
Taking a product of this over all j's:

$$= P(y^{i1}/x, y^{gold}).P(y^{i2}/x, y^{gold}, y^{i1})....P(y^{ik}/x, y^{gold}, y^{i1}, y^{i2}...y^{i(k-1)}).$$
$$P(y^{EOS}/x, y^{gold}, y^{i1}, y^{i2}...y^{i(k)})$$

$$= P(\{y^{i1}, y^{i2}..y^{ik}, y^{EOS}\}/x, y^{gold})$$

Since we have a one-one mapping between sequence and set

$$= P(S/x, y^{gold})$$



It is important to note that EOS token will signify that **no other label** is in the partial set. In effect, it eliminates high sensitivity to negative probabilities.

Fold	Avg	0	1	2	3	4	5	6	7	8	9
CC Loss	48.51	49.71	46.86	48.00	43.43	52.00	50.29	38.86	49.71	52.57	53.71
LSTM No Pretrain P	41.71	45.14	37.71	42.86	46.86	36.57	34.86	34.86	41.71	50.29	46.29
LSTM P50%	42.63	42.86	40.00	50.29	41.71	42.86	38.29	35.43	41.14	46.29	47.43
LSTM P80%	42.57	42.86	40.00	50.29	41.14	42.86	38.29	35.43	41.14	46.29	47.43
LSTM P90%	43.09	38.29	42.86	50.86	40.57	44.57	41.14	34.86	45.71	45.14	46.86
LSTM P100%	47.37	48.57	38.86	55.43	48.00	54.86	45.71	29.71	46.29	55.43	50.86

Table 12: Results for different Pretrained P Bases, on MSRCv2

8.3.1 Pretraining P Model

Once again, we experiment with different pretrained bases of prediction network, to determine which is most suitable for joint training with the LSTM.
It is clear that pretraining to the best model of the CC Loss gives the best results.

8.3.2 Pretrain Generative Model (LSTM)

We have two techniques:

1. Freeze P: Freeze the prediction network and allow the generative model to train for 50 epochs in order to warm up weights before commencing joint training.
2. Pretrain G: We take most confident predictions ($> 95\%$ confidence) of the P network and assume them to be the gold label. We then train the LSTM inputting x and this gold label, against the partial set as the target.

We find (Table 12) that these strategies improve over our baselines. Pretrain G is the best strategy, we combine it with Freeze P.

8.3.3 Final LSTM Results

Finally adopting a 100% P Pretraining Base, and a Pretrain G with freeze stategy, we train for all datasets and report the results. We get improvements in 3/5 real world datasets, and also in synthetic data. Results are reported in Table 14 and Table 15.

8.3.4 Testing the LSTM Generative Model

We also check if our LSTM is learning a sensible model by computing the Intersection over Union score in prediciton the partial set S, given x and y

Fold	Avg	0	1	2	3	4	5	6	7	8	9
CC Loss	48.29	49.71	46.86	48.00	43.43	52.00	50.29	41.14	49.71	54.86	46.86
LSTM Without Pretrain G	46.40	49.71	35.43	48.57	49.14	53.71	47.43	38.86	44.57	45.71	50.86
LSTM Freeze P	48.34	49.71	42.86	50.29	49.14	49.71	50.86	40.57	49.71	53.71	46.86
LSTM Pretrain G	49.83	49.71	46.86	52.57	49.14	55.43	50.29	41.14	49.71	56.57	46.86

Table 13: Results for different G Pretraining, on MSRCv2

Dataset	lost	MSRCv2	BirdSong	Yahoo! News	Soccer Player
CC Loss	72.86	48.29	70.64	67.38	55.19
LSTM	73.21	49.83	69.76	67.77	55.12

Table 14: Test Acc: CC Loss v/s LSTM on real world datasets

Dataset	lost flip (Synthetic)	MSRCv2 flip (Synthetic)
CC Loss	69.02	54.80
LSTM	70.54	57.71

Table 15: Tess Acc: CC Loss v/s LSTM on synthetic datasets

	Gold Train IOU	Non Gold Train IOU	Gold Test IOU	Non Gold Test IOU
lost	0.82	0.54	0.59	0.47
MSRCv2	0.48	0.24	0.40	0.22
BirdSong	0.70	0.36	0.66	0.35
Yahoo! News	0.87	0.47	0.62	0.42
Soccer Player	0.69	0.44	0.50	0.31
lost_flip	0.66	0.15	0.46	0.13
MSRCv2_flip	0.58	0.11	0.38	0.10

Table 16: LSTM Intersection Over Union Values

We run the LSTM in test mode. At each stage we predict the next element of the sequence. We must ensure the 'next' element has a higher seqnum than the previous, and we enforce this by taking max only over the higher seqnum elements.
We do this for gold, as well as non gold labels, and report IOUs in Table 16

9 Details of Training

9.1 Models

A brief description of the neural models that we use. Detailed implementation can be studied in *networks.py*:

1. Prediction Network: It takes in input x, and gives a logits output over label space. We use a 3 layer MLP: input-dim $\times$ 512 $\times$ 256 $\times$ class-dim. We use Batch Normalization between layers, and initilaise using xavier uniform. A base description for our prediction model was obtained from [10]

2. Selection Network (for RL): It takes in the partial set S and optionally the input x as well. It outputs logits over the label space.
   We use a 3 layer MLP: input-dim $\times$ 512 $\times$ 512 $\times$ 256 $\times$ class-dim. We use Batch Normalization between layers, and initialise using xavier uniform.

3. Generative Models

   (a) Dependence on y: Takes in the assumed gold label, outputs logits over label space. This is implemented as a 1 layer MLP: class-dim $\times$ class-dim.
   We also ensure that all probabilities on the diagonal are set to 1. This is accomplished by setting logits on the diagonal to a high value ≈ 15. On taking sigmoid, this gives a 1.

(b) Dependence on y, Naive Bayes: Take in x and the assumed gold label y, outputs logits over label space. Instead of taking in y as a one hot vector, we use a trainable embedding of size 96.
This is implemented as a 4 layer MLP: (input-dim+96) $\times$ 512 $\times$ 256 $\times$ 128 $\times$ class-dim. We use Batch Normalization between layers, and initilaise using xavier uniform.
Once again, we force diagonal entries to 1, similar to (a).

(c) Dependence on y, LSTM: Let us consider set S to be a sequence of labels $\{y^{j1}, y^{j2}, y^{j3}...\}$. Then at each time step t we input x, the assumed gold label, and y^{jt}. Once the sequence is complete, we input x, assumed gold and the End of Sequence (EOS) Token. At each step, logits over label space is outputted.
The LSTM itself has a single hidden layer of dim = 100. The y's are inputted via an embedding of size 128.

9.2 Cross Validation

We train using 10 fold cross-validation on our datasets. The numbers we report are averaged over 10 folds.
As preprocessing, we standardise our input features, then shuffle the dataset and save into a new file as a finalised dataset..
On doing the i^{th} round of cross validation, we divide the dataset into 10 equal parts, use i^{th} part as test, $(i+1)^{th}$ part as validation and the rest as train, thus achieving a 8:1:1 train:val:test split.
We maintain the same folds and splits for our artificial datasets at well.

9.3 Training Technqiue

We use mini-batch gradient descent, with a batchsize of 64.
We train *lost* and *BirdSong* for 1500 epochs, *MSRCv2* for 1000 epochs and *Soccer Player* and *Yahoo! News* for 150 epochs.

9.4 Early Stopping

9.4.1 Surrogate v/s Real Criteria

In partial label learning, we only have access to the partial set, and not to the true labels at the time of validation. Thus we must make hyperparameter decisions based on accuracy on the partial set. For this purpose we make a distinction. Given our training instance (x_i, S_i), true gold label y_i and our predicted gold label $\tilde{y}_i$:

1. Real (Train/Val/Test) Accuracy: Prediction $\tilde{y}_i$ is 'correct' iff $\tilde{y}_i = y_i$

2. Surrogate (Train/Val/Test) Accuracy: Prediction $\tilde{y}_i$ is 'correct' iff $\tilde{y}_i \in S_i$

9.4.2 Accuracy v/s Loss Criteria

We can also pick best model based on Surrogate Accuracy, or Surrogate Loss Function. We performed experiments and find that accuracy is a better predictor for real test accuracy.

We thus perform early stopping by picking the model which provides maximum **surrogate** accuracy on the **validation** dataset.

9.5 Hyper Parameter Tuning

We find that our algorithms are sensitive to hyperparameters as follows:

1. Optimizer: We experiment between Adam and Stochastic Gradient Descent (SGD) Optimizers
2. Learning Rate: We experiment with learning rates {1, 0.1, 0.01, 0.001, 0.0001}
3. Weight Decay: We experiment with values of weight decay (L2 regularization norm) in our opimizer $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$

We thus choose best hyperparameters based on best surrogate validation accuracy. This is done **per fold**.

9.6 LR Scheduling

We schedule learning rate decay at epoch i as follows:

1. For epoch ≤ 200: $lr_i = (0.98)^i * lr_0$
2. Else: $lr_i = 0.01 * lr_0$

10 Conclusion

The work primarily focused on experimenting with different techniques in Partial Label Learning, taking inspiration from other domains of ambiguous learning.

1. Loss Functions: We have done a thorough survey of various loss functions - Naive, CC, Min, IExplr, SVM, Cour and Regularized Losses. We have run experiments using this losses to use as baselines, and also derived some theoretical relationships between them.
2. Reinforcement Learning: We find that RL does not improve performance in PLL over CC Loss baselines, both in linear as well as exponential action space settings.

3. Generative Modelling: We find that these techniques show promise. We obtained positive results on synthetic datasets in the Y-Matrix model, as well as on both real world and synthetic datasets in the LSTM Model. Possible future work in this direction would involve further understanding the generation process for real world datasets, and improving the generative model accordingly.

References

[1] Raich Briggs, Fern. Rank-loss support instance machines for miml instance annotation. 2012.

[2] Vivien Cabannes, Alessandro Rudi, and Francis Bach. Structured prediction with partial labelling through the infimum loss. *CoRR*, abs/2003.00920, 2020.

[3] Timothee Cour and Benjamin Sapp. Learning from Partial Labels. *JMLR*, 12:1501–1536, 2011.

[4] Han et al. Feng, Lv. Provably consistent partial-label learning. 2020.

[5] Zhao et al Feng, Huang. Reinforcement learning for relation classification from noisy data. 2018.

[6] Schmid Guillaumin, Verbeek. Multiple instance metric learning from au- tomatically labeled bags of faces. 2010.

[7] Dietterich Liu. A conditional multinomial mixture model for superset label learning. 2012.

[8] Yatin Nandwani, Deepanshu Jindal, Mausam, and Parag Singla. Neural learning of one-of-many solutions for combinatorial problems in structured output spaces, 2020.

[9] Wang Qin, Xu. Robust distant supervision relation extraction via deep reinforcement learning. 2018.

[10] Huh Seo. On the power of deep but naive partial label learning. 2020.

[11] Park Lee Song, Kim. Learning from noisy labels with deep neural networks: A survey. 2020.

[12] Jia et al Zeng, Xiao. Learning by as- sociating ambiguously labeled images. 2012.