

# Machine Teaching for Crowdsourcing

Nancy Cao  
Andrew Chico  
Betsy Fu  
Daniel Wang

# Table of Contents

- Background and Teaching Process
  - Review of Submodularity
- Model of the Learner
- Teaching Algorithm
- Experiments
- Experimental Results
- Conclusion

# Background and Teaching Process

# Overview

- Last time: Machine Teaching
  - How do we design an optimal example set to show a learner?
    - Optimal: minimizing some objective function, often size
    - Classic example:



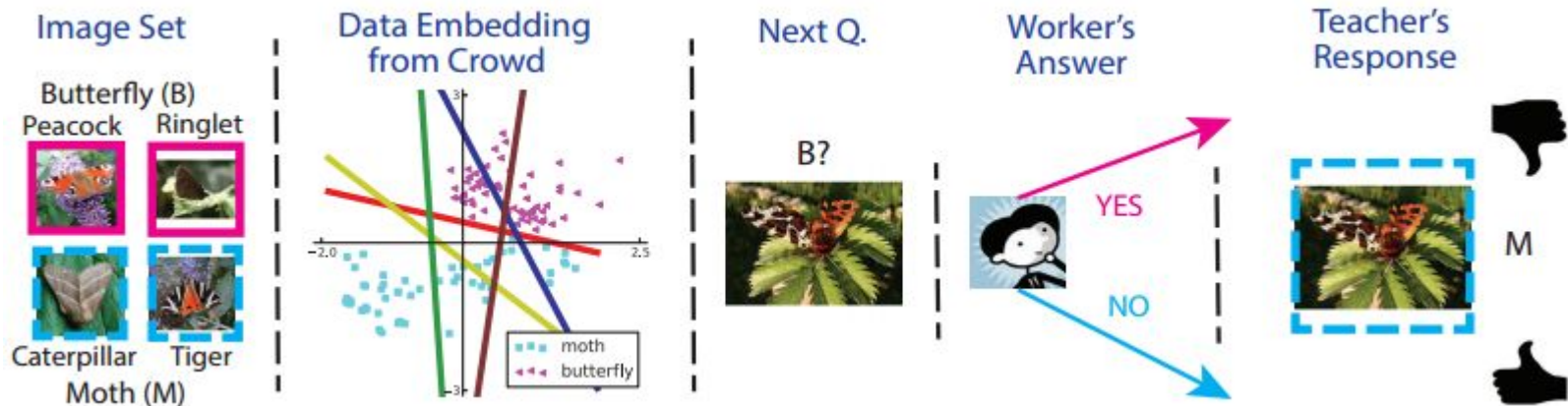
- How do we teach humans?
  - “Extreme”/curriculum strategy vs boundary strategy

# Overview

- Crowdsourcing
  - Useful for obtaining a large number of labels
  - But low-paid workers make mistakes, and may not work very hard/accurately
  - Result: Lots of noise in the labels.
- Using MTurk or other crowd platforms to gather labels, how do we obtain the highest-quality labels with a given budget?

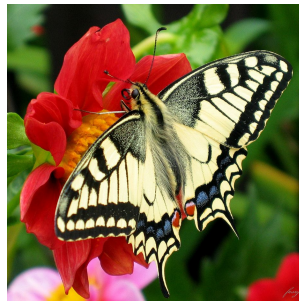
# Machine Teaching for Crowdsourcing

- Can we teach workers in crowdsourcing services in order to improve their accuracy?
- Combination of the previous topics
- Explores orthogonal direction to the previous crowdsourcing paper
  - Instead of testing worker reliability, teach workers to be more reliable



# Problem Setup

- Problem: We wish to teach the crowd to label a large set of images.
  - Restrictions: We only have ground truth labels for a small “teaching set” of examples.
1. From the teaching set, we elicit a set of candidate features and a collection of hypotheses (linear classifiers) that the crowd might be using.
  2. We use a teaching algorithm to select training examples and steer the learner towards the target hypothesis.



Butterfly or Moth?

# Noise-tolerant teaching

- **Noise-free:** Learners immediately abandon hypotheses inconsistent with observed examples.
- **Noise-tolerant:** There are less strict assumptions on how workers treat inconsistent hypotheses.





# STRICT

- Submodular Teaching for Crowdsourcing Classification
- Noise-tolerant
- Approximately optimal (chooses the smallest teaching set)
- With a few assumptions (linear separators, etc), obtain learner error less than  $\epsilon$  with  $81\log^2(\frac{2}{\epsilon})$  examples.

# Review: Submodularity


- Problem: How can we choose as few examples as possible while maximizing efficiency?
  - Minimize number of examples
  - Don't want examples to overlap
- We can phrase this problem as a **submodularity optimization problem**.
- Submodularity problems can be optimized using a **greedy algorithm** (Nemhauser et al., 1978).
  - In order to find the global optimum, we make the locally optimal choice at each stage.

# Submodularity Example


- We would like to estimate the **average temperature of a building** using sensors placed in various locations in the building.
  - Near-optimal sensor placements: Maximizing information while minimizing communication cost (Krause et al., 2006)
- However, we only have a **limited number** of sensors.
- Where do we place the sensors to **cover the most space** and **get the most accurate reading** of the building temperature?

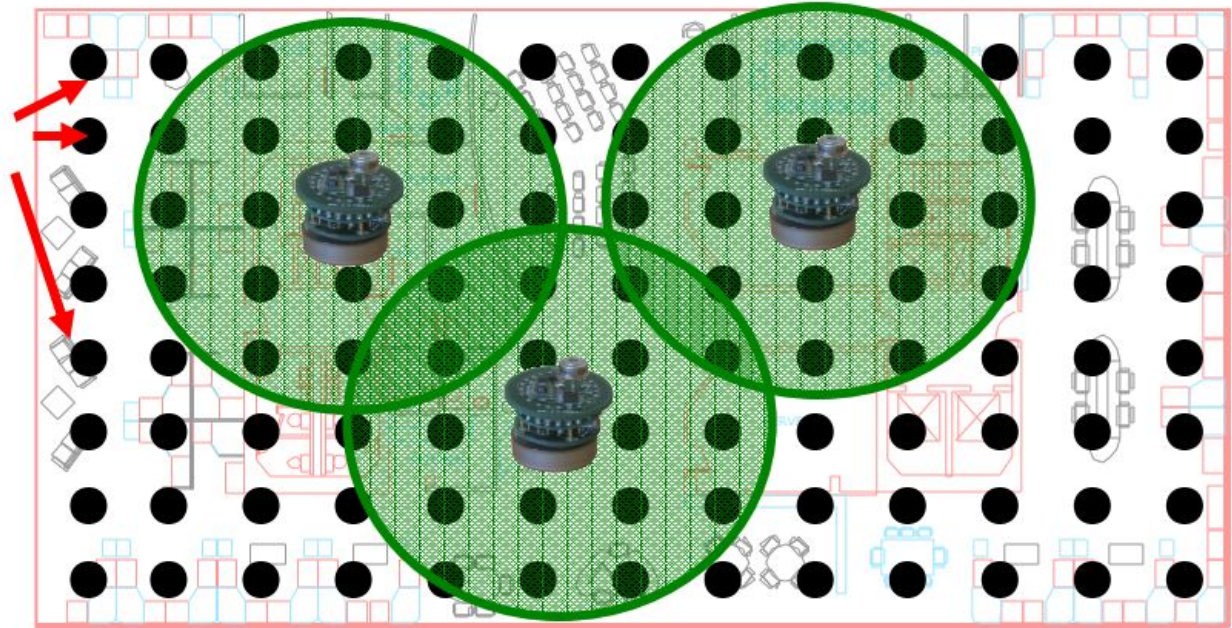


# Submodularity Example

 : sensors

● : possible locations  
for sensors


 : physical space  
we want to cover

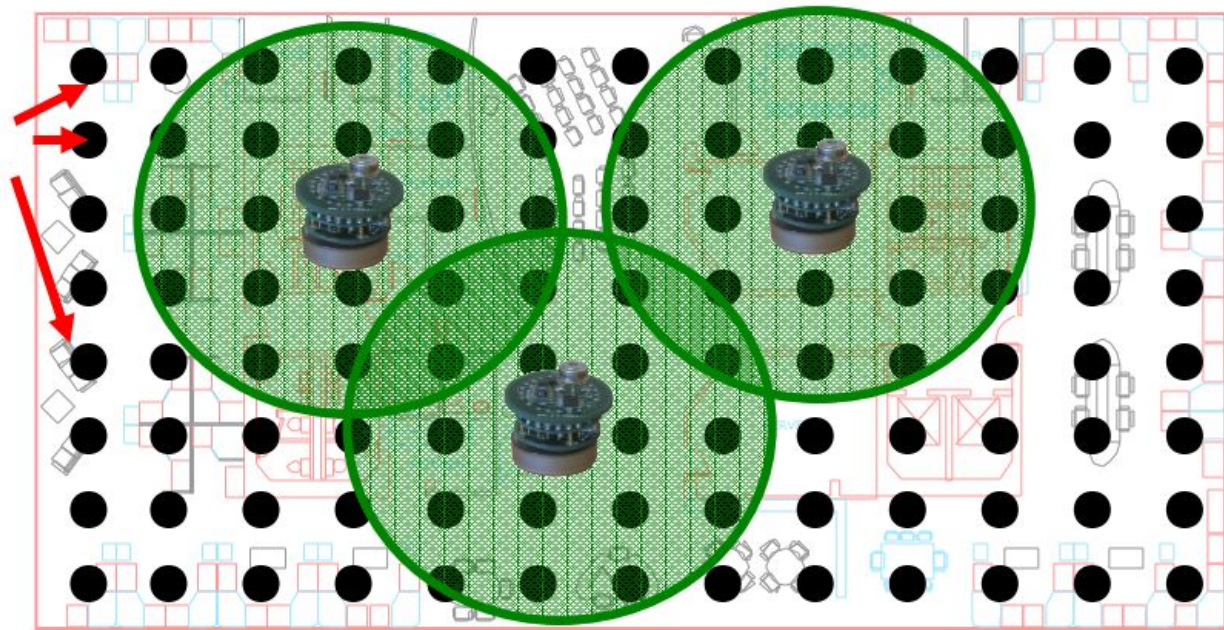


# Submodularity Example

 : **examples**

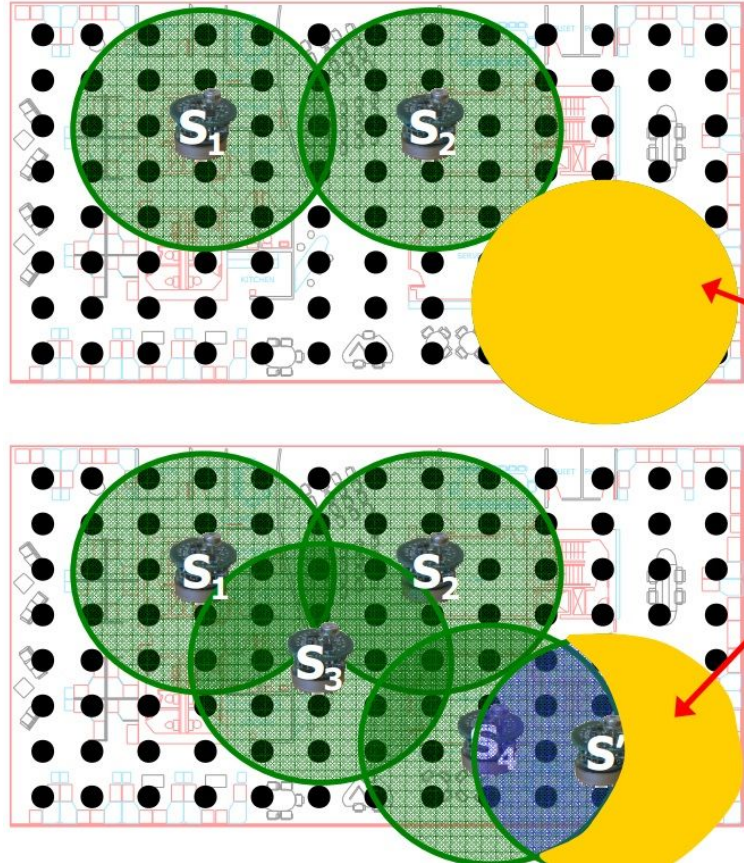
● : possible locations  
for **examples**

 : **feature** space  
we want to cover





# Submodularity Example



# Review: Submodularity

- In our average temperature example, we see that selecting two sensors that cover **some or all of the same area lowers our utility/coverage** of the building space we want to cover.
- Thus, it is in our best interest to choose **diverse sensor placement**, as opposed to redundant sensor placement.
- Analogously, to maximize utility/coverage, it is in our best interest to choose **diverse teaching examples**, as opposed to redundant teaching examples.



# Mathematical model

- $\mathcal{X}$ : set of examples (e.g. images) called the “*teaching set*”
  - Labeled examples given as  $(x,y)$ , with  $y \in \{1,-1\}$
- $\mathcal{H}$ : a finite class of hypotheses
  - $h(x) = \text{sgn}(x)$
  - $h : \mathcal{X} \mapsto \mathbb{R}$
  - $|h(x)|$ :  $h$ 's confidence in labeling  $x$
- Assume that  $\mathcal{X}$  and  $\mathcal{H}$  are known to both the teacher and the learner.



# Mathematical model (cont'd)

- Teacher
  - knows labels  $y(x)$  for all examples  $x \in \mathcal{X}$
  - knows correct hypothesis  $h^*$  for which  $\text{sgn}(h^*(x)) = y(x)$  for all  $x \in \mathcal{X}$
  - Note: realizable setting
- Learner
  - just knows  $\mathcal{H}$
  - wants to find  $h^*$

**Big idea:** if we can teach classification of  $\mathcal{X}$ , we should have similar performance on new examples drawn with the same distribution as  $\mathcal{X}$

# Model of the Learner

# Modeling the learner

- Modelled as a non-stationary Markov chain
- Carries out a **random walk** in  $\mathcal{H}$
- Starts at some hypothesis
- Stays there until an inconsistency arises
- Then, randomly jumps to an alternative hypothesis more consistent with the received training

# Formal Definition

1. Before the first example, the learner randomly chooses a hypothesis  $h_1$  from some prior distribution  $P_0$ .
2. Every round  $t$ , there are two possibilities.

The example  $(x_t, y_t)$  received **agrees** with the learner's prediction, i.e.  $\text{sgn}(h_t(x_t)) = y_t$ .

Keep the same hypothesis

$$h_{t+1} = h_t$$

The label  $y_t$  **disagrees** with the learner's prediction, i.e.  $\text{sgn}(h_t(x_t)) \neq y_t$ .

Draw a new hypothesis based on distribution  $P_t$  that reduces the probability of hypotheses that disagreed with the previous true labels.

# Constructing $P_t$

More precisely, if  $\text{sgn}(h_t(x_t)) \neq y_t$ , then  $h_{t+1}$  is drawn from:

$$P_t(h) = \frac{1}{Z_t} P_0(h) \prod_{\substack{s=1 \\ y_s \neq \text{sgn}(h(x_s))}}^t P(y_s | h, x_s) \quad (1)$$

This has normalization factor

$$Z_t = \sum_{h \in \mathcal{H}} P_0(h) \prod_{\substack{s=1 \\ y_s \neq \text{sgn}(h(x_s))}}^t P(y_s | h, x_s)$$

# Updating the learner hypotheses

If we model label likelihood with a sigmoid, we get, for some  $\alpha > 0$ ,

$$P(y_s|h, x_s) = \frac{1}{1 + \exp(-\alpha h(x_s)y_s)}$$

$\alpha$ : scaling parameter that controls the effect of inconsistent examples

As  $\alpha \rightarrow \infty$ , our learner never jumps to a hypothesis with an inconsistency, corresponding to the **noise-free case**.

The intuition here is that **hypotheses that are strongly inconsistent to observed data have a low likelihood of being chosen**.

# Toy example

- Butterfly vs moths
- 3 hypotheses: bright color == butterfly, large body == moth, purple == moth
- Prior probabilities:  $\frac{2}{5}$ ,  $\frac{2}{5}$ ,  $\frac{1}{5}$
- Pick a random hypothesis to start with: purple.
- Get example:



# Toy example

- We would predict butterfly with high confidence, but this would be an error.
- Update probabilities:  $P_1 = (.48, .48, .02)$
- Choose new hypothesis: bright color.
- Get new example:





# Toy example

- No error, but lower confidence prediction.
- $P_2 = (.485, .485, .01)$
- Even so, we keep the same hypothesis.

# Notes on the learner

- Not a very good approximation for a human learner
- Infinite memory
- Only changes on error
- Always reconsiders on error
- Even so, experiments show interesting results

# Teaching Algorithm

# Teaching Motivation

- Now that we've defined the learner's behavior, **how should the teacher choose examples to help the learner narrow down her belief to accurate hypotheses?**
- The ultimate goal of the teacher is to steer the learner towards a distribution with which she makes few mistakes.

# Teaching Algorithm

- $A = \{x_1, \dots, x_t\}$ : a set of examples shown to the learner
- Then, the posterior distribution is referred to by  $P_t(\cdot)$  and  $P(\cdot | A)$  interchangeably.
- With the new notation, we have the **learner's posterior after showing  $A$** :

$$P(h|A) = \frac{1}{Z(A)} P_0(h) \prod_{\substack{x \in A \\ y(x) \neq \text{sgn}(h(x))}} P(y(x)|h, x)$$

# Expected Error-Rate of the Learner

$$\mathbb{E}[\text{err}_L|A] = \sum_{h \in \mathcal{H}} P(h|A) \text{err}(h, h^*), \text{ where}$$

$$\begin{aligned} \text{err}(h, h^*) &= \frac{|\{x \in \mathcal{X} : \text{sgn}(h(x)) \neq \text{sgn}(h^*(x))\}|}{|\mathcal{X}|} \\ &= \frac{\text{\# of examples in } \mathcal{X} \text{ on which } h \text{ and } h^* \text{ disagree about the label}}{\text{Total \# of examples in } \mathcal{X}} \end{aligned}$$

- If our hypothesis  $h^*$  is **very inconsistent** with other hypotheses in  $P$  that are likely to be correct, it's **more likely that  $h^*$  will be wrong**, and vice versa.
- Weighted average of these errors

# Finding a set of examples

- Given an allowed tolerance  $\epsilon$  for the learner's error, we would like to find the **smallest set of examples  $A^*$  achieving this error**, i.e.

$$A_{\epsilon}^* = \arg \min_{A \subseteq \mathcal{X}} |A| \text{ s.t. } \mathbb{E}[\text{err}_L \mid A] \leq \epsilon. \quad (2)$$

---

**Proposition 1.** Problem (2) is NP-hard.

# Approximating Problem (2)

We will approximate this NP-hard problem by framing it as the problem of **maximizing Problem (2) with a budget constraint** as follows.

Let us first consider the following objective function.

$$\begin{aligned} R(A) &= \mathbb{E}[\text{err}_L] - \mathbb{E}[\text{err}_L \mid A] \\ &= \sum_{h \in \mathcal{H}} (P_0(h) - P(h|A)) \text{err}(h, h^*), \end{aligned}$$

This quantifies the *expected reduction in error* upon teaching  $A$ .

Solving Problem (2) is equivalent to finding the smallest set  $A$  achieving error reduction  $\mathbb{E}[\text{err}_L] - \epsilon$ .



## Approximating Problem (2) (cont'd)

- We would like to find, for each  $k$ , a set  $A$  of size  $k$  that maximizes  $R(A)$ .
- Define the *surrogate function*

$$F(A) = \sum_{h \in \mathcal{H}} (Q(h) - Q(h|A)) \text{err}(h, h^*)$$

- with *unnormalized posterior* of the learner

$$Q(h|A) = P_0(h) \prod_{\substack{x \in A \\ y(x) \neq \text{sgn}(h(x))}} P(y(x)|h, x)$$

## Approximating Problem (2) (cont'd)

$$F(A) = \sum_{h \in \mathcal{H}} (Q(h) - Q(h|A)) \text{err}(h, h^*)$$

- This surrogate objective function satisfies *submodularity*.
  - Natural diminishing returns function
- As previously mentioned, submodular functions can be effectively optimized using a **greedy algorithm**.
  - At each iteration, we will add the example that maximally increases  $F$ .
  - As before, as  $\alpha \rightarrow \infty$ ,  $F(A)$  corresponds to the **noise-free case**.
- Running the algorithm until  $F(A) \geq \mathbb{E}[\text{err}_L] - P_0(h^*)\epsilon$  is sufficient to produce a feasible solution to Problem (2).

---

## Policy 1 Teaching Policy STRICT

---

- 1: **Input:** examples  $\mathcal{X}$ , hyp.  $\mathcal{H}$ , prior  $P_0$ , error  $\epsilon$ .
  - 2: **Output:** teaching set  $A$
  - 3:  $A \leftarrow \emptyset$
  - 4: **while**  $F(A) < \mathbb{E}[\text{err}_L] - P_0(h^*)\epsilon$  **do**
  - 5:      $x \leftarrow \arg \max_{x \in \mathcal{X}} (F(A \cup \{x\}))$
  - 6:      $A \leftarrow A \cup \{x\}$
  - 7: **end while**
-

# Approximation Guarantees

**Theorem 1.** *Fix  $\epsilon > 0$ . The STRICT Policy 1 terminates after at most  $\text{OPT}(P_0(h^*)\epsilon/2) \log \frac{1}{P_0(h^*)\epsilon}$  steps with a set  $A$  such that  $\mathbb{E}[\text{err}_L \mid A] \leq \epsilon$ .*

# Sketch of Proof of Theorem 1

- We can write

$$F(A) = \sum_{h \in \mathcal{H}} P_0(h) G_h(A) \text{err}(h, h^*)$$

$$G_h(A) = 1 - \prod_{\substack{x \in A \\ y(x) \neq \text{sgn}(h(x))}} P(y(x)|h, x)$$

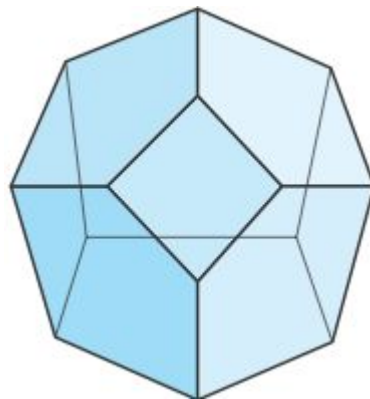
Since  $G_h(A)$  is submodular for every  $h$ ,  $F(A)$  is also submodular.

# Sketch of Proof of Theorem 1 (cont'd)

- Since  $F(A)$  is submodular, as previously mentioned, the greedy algorithm gives a set reasonably close to  $F$ 's optimum.
- We analyze the connection between **maximizing  $F(A)$**  and **minimizing the expected error of the learner**.
- Maximizing  $F(A)$  is both **necessary and sufficient** to achieve  $\varepsilon$  precision.
- See the entire proof in the supplementary material.

# Teaching Complexity for Linear Separators

- Under some additional assumptions, we can show that the **optimal number of examples is not too large**.
- First, we will define  **$\lambda$ -richness**.
- Note that  $\mathcal{H}$  partitions  $\mathcal{X}$  into **polytopes** (intersections of half-spaces), where with a polytope, all examples are labeled the same by every hypothesis.
- Then,  $\mathcal{X}$  is  **$\lambda$ -rich** if any  $\mathcal{P}$  contains at least  $\lambda$  examples.



# Teaching Complexity for Linear Separators

- Under some additional assumptions, we can show that the **optimal number of examples is not too large**.

**Theorem 2.** *Fix  $\epsilon > 0$ . Suppose that the hypotheses are hyperplanes in  $\mathbb{R}^d$  and that  $(\mathcal{X}, \mathcal{H})$  is  $(8 \log^2 \frac{2}{\epsilon})$ -rich. Then the STRICT policy achieves learner error less than  $\epsilon$  after at most  $m = 8 \log^2 \frac{2}{\epsilon}$  teaching examples.*



# Sketch of Proof of Theorem 2

- Let us consider the randomized teaching policy **Relaxed-Greedy Teaching Policy** (next slide).
- With positive probability, this policy reduces the learner error exponentially.
- Thus, there must exist a sequence of examples that reduce the learn error exponentially.
- See the entire proof in the supplementary material.

---

**Policy 2** Relaxed-Greedy Teaching Policy (RGTP)

---

- 1: **Input:** examples  $\mathcal{X}$ , hypothesis  $\mathcal{H}$ , prior  $P_0$ , error  $\epsilon$ .
  - 2:  $t = 0, P_0^{(t)}(h) = P_0$
  - 3: **while**  $1 - P_t^{(t)}(h^*) > \epsilon$  **do**
  - 4:     **if** there exists two neighboring polytopes  $\mathcal{P}$  and  $\mathcal{P}'$  s.t.  $\sum_h P_t^{(t)}(h)h(\mathcal{P}) > 0$  and  $\sum_h P_h^{(t)}(t)h(\mathcal{P}') < 0$  **then**
  - 5:         select  $x_t$  uniformly at random from  $\mathcal{P}$  or  $\mathcal{P}'$ .
  - 6:     **else**
  - 7:         select  $x_t$  from polytop  $\mathcal{P} = \arg \min_{\mathcal{P} \in \Pi} |\sum_h P_t^{(t)}(h)h(\mathcal{P})|$
  - 8:     **end if**
  - 9:      $\forall h \in \mathcal{H}$  update  $P_{t+1}^{(t)}(h)$  according to (3) and  $t \rightarrow t + 1$ .
  - 10: **end while**
-

# RGTP vs. STRICT

- At each step, RGTP picks an example uniformly at random that is relatively good, but not necessarily the absolute greediest choice.
  - if condition

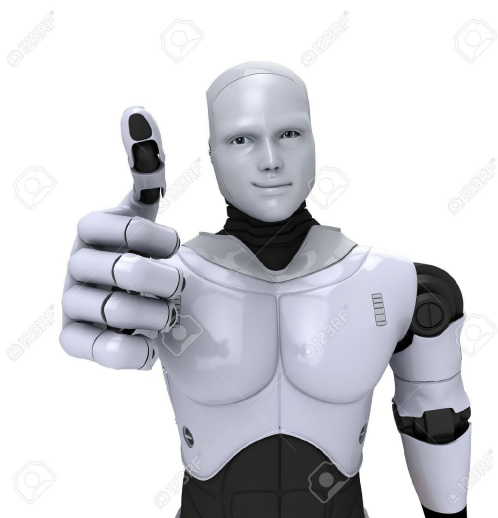
**if** there exists two neighboring polytopes  $\mathcal{P}$  and  $\mathcal{P}'$  s.t.  $\sum_h P_t^{(t)}(h)h(\mathcal{P}) > 0$  and  $\sum_h P_h^{(t)}(t)h(\mathcal{P}') < 0$  **then** select  $x_t$  uniformly at random from  $\mathcal{P}$  or  $\mathcal{P}'$ .

- As one would intuitively guess, RGTP is more relaxed than STRICT.

# Experiments

# Evaluating efficacy of STRICT

A set of three experiments were created to test the overall effectiveness of the STRICT algorithm. These experiments were both simulated and carried out using the Mechanical Turk platform.



# Experiment 1 (VW Classification)

- The first experiment deals with classification of insectile silhouettes as belonging to a weevil or to a vespula.

Vespula (V)



Weevil (W)



Vespula and Weevil (VW)

# Experiment 1 (VW Classification)

- These insect images are synthetically generated. In general, weevils are depicted as creatures with a small head-to-body ratio and relatively low color contrast between the head and the abdomen.
- Vespulae, on the other hand, are shown having relatively large heads and heads that don't quite match their body color.

Vespula (V)

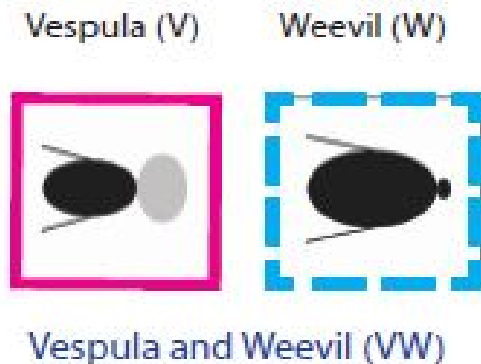
Weevil (W)



Vespula and Weevil (VW)

# Synthetic VW images

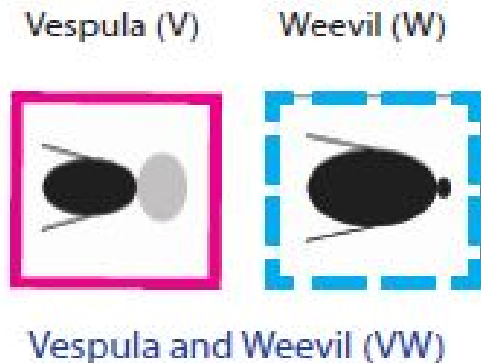
- VW images are generated along two dimensions. As alluded to previously, only head-to-abdomen ratio and color contrast between the head and abdomen are different between each image.
- Each image is represented as a feature vector.





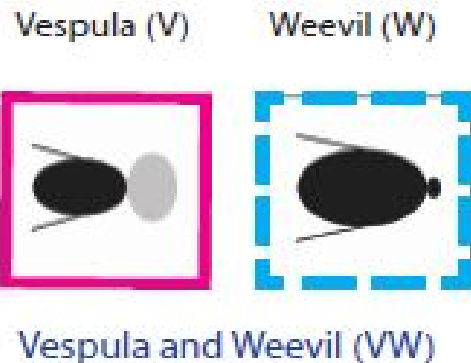
# Synthetic VW images

- These feature vectors are expressed as  $x_i = [x_{i,1} = f_1, x_{i,2} = f_2]$
- Both parameters are independent from each other and drawn from a bivariate Gaussian distribution (with zero correlation/covariances). Thus they are each effectively drawn from independent univariate Normal distributions.



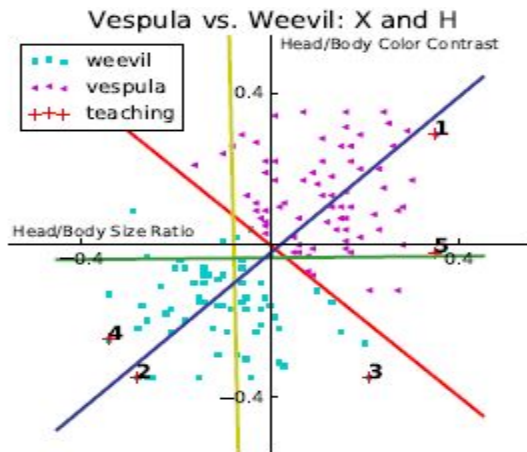
# Synthetic VW images

- A total of 80 images are generated for usage by the teaching algorithm, while another 20 are generated to assess learning and gather data on the rate of learning.
- The hypothesis class  $\mathcal{H}$  is then generated. They are linear functions with parameters also generated by bivariate Normal distributions.



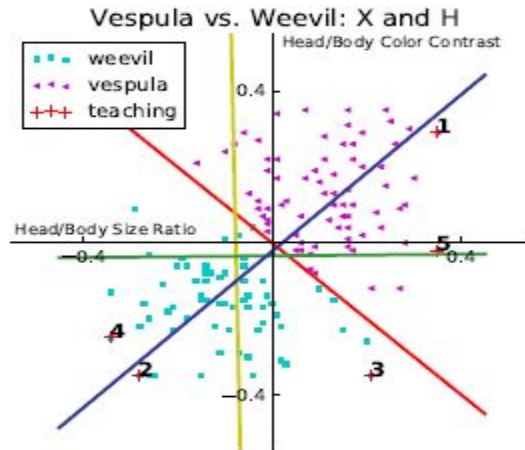
# VW hypotheses

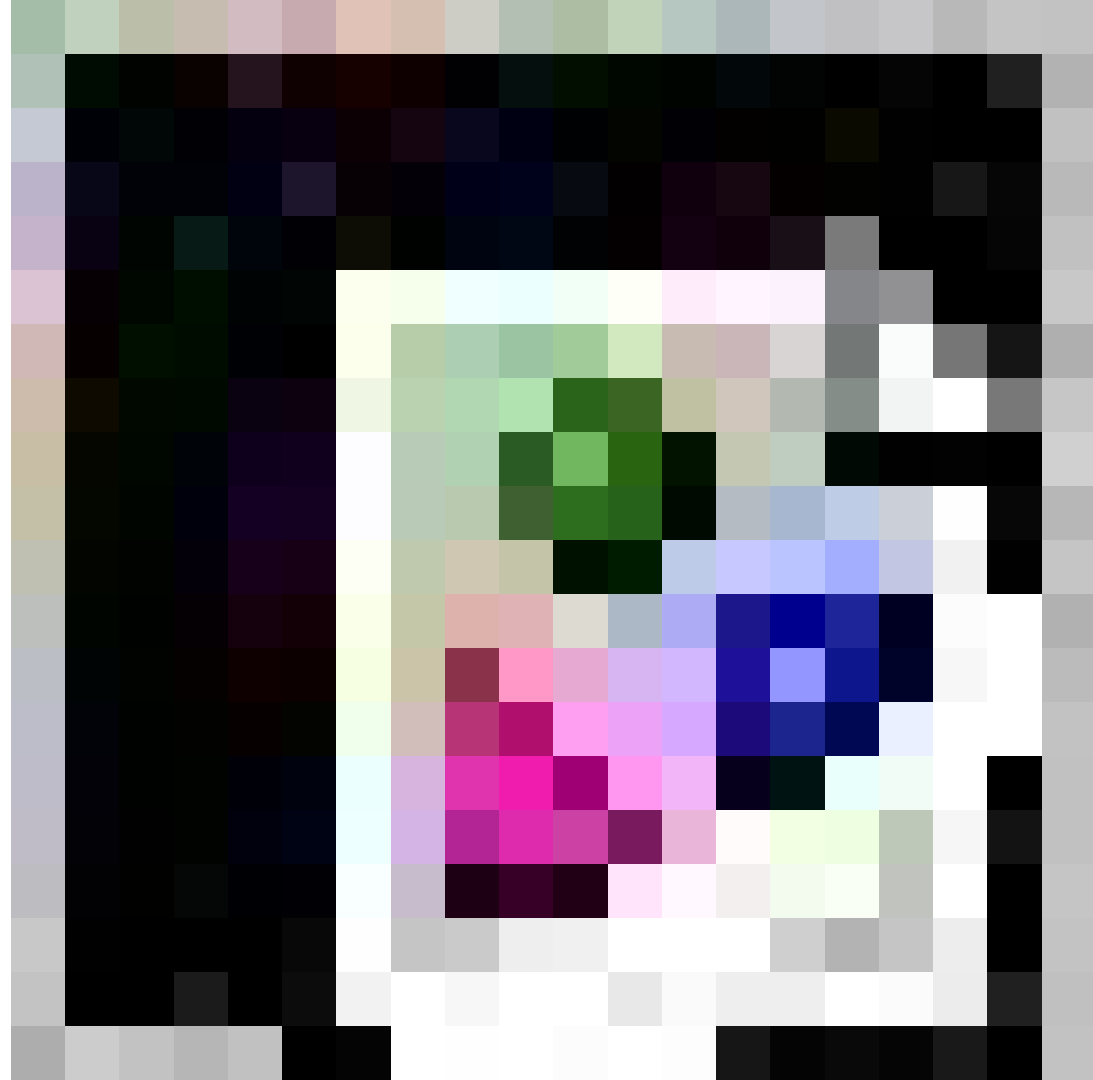
- There are eight clusterings of hypotheses (with respect to the slope of the dividing line). When considering the angle each line makes with the x-axis, there is a cluster for each multiple of 45 degrees. Within each cluster, the exact y-intercept and slope is determined again by a bivariate Normal distribution.



# VW hypotheses

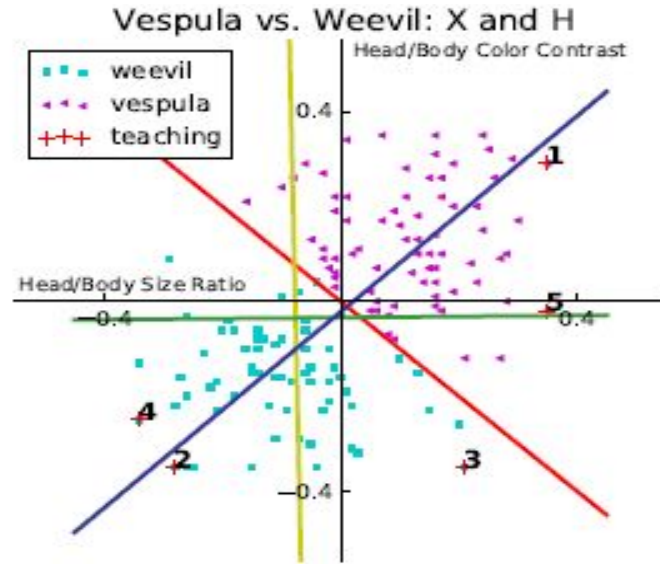
- Clusters are chosen for a reason.
- Each cluster represents a way of interpreting the data (ignoring a parameter, incorrectly assessing implication of trend in parameter, correctly assessing the impact of the parameter).





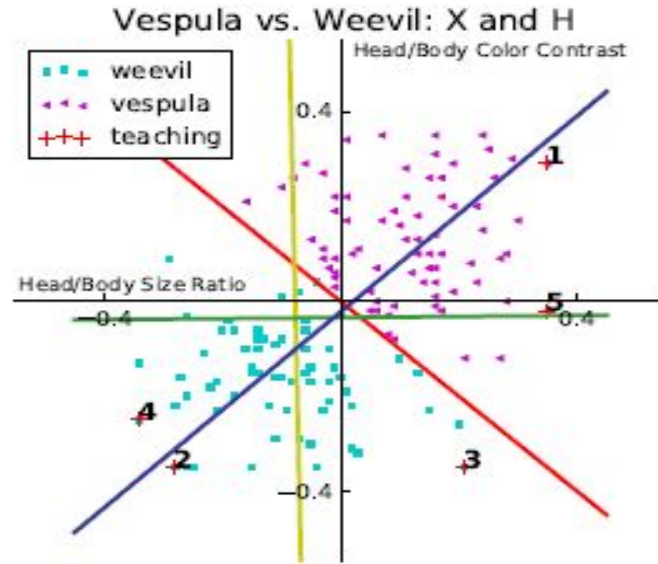
# VW hypotheses

- Yellow: Ignores head/body color contrast altogether.



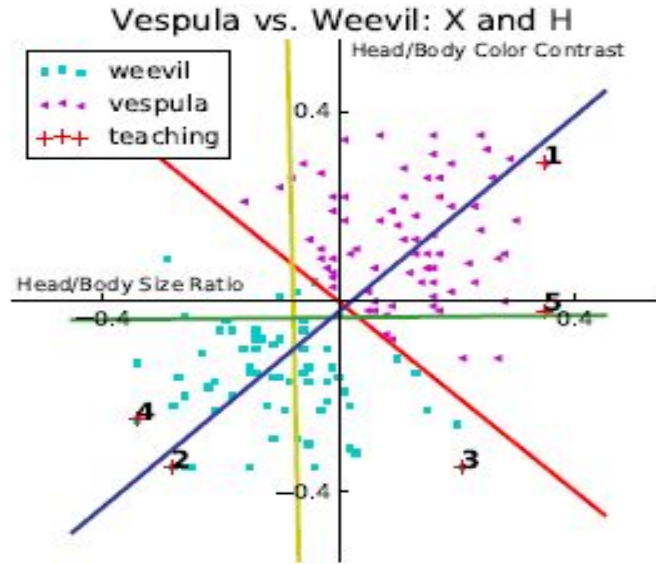
# VW hypotheses

- Blue: Correctly identifies color trend, incorrectly identifies body ratio trend.  
(could also correctly identify neither)



# VW hypotheses

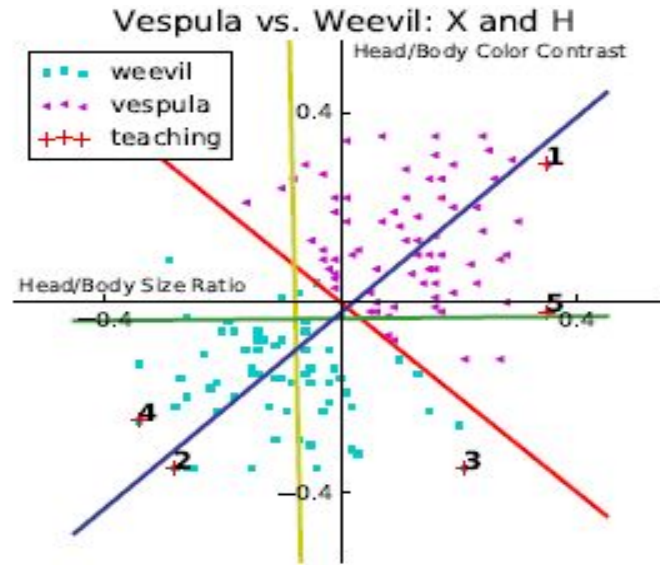
- Green: Ignores body ratio altogether..





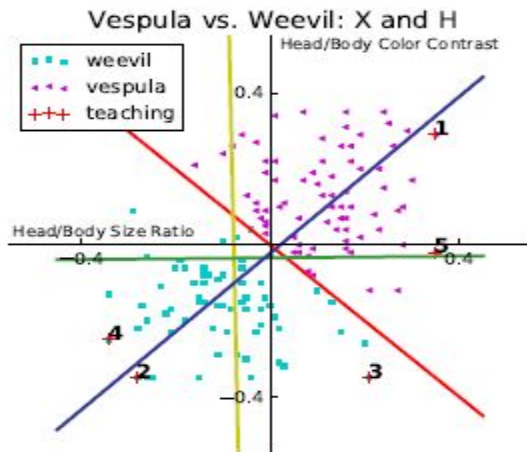
# VW hypotheses

- Red: The target hypothesis clearly identifies all trends correctly. Note that all elements of the training set are correctly identified.



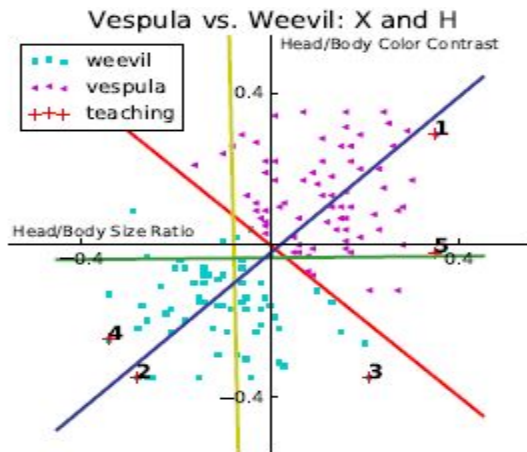
# VW hypotheses

- After all hypotheses were generated, a target hypothesis  $h^*$  is selected with minimal error. From here, elements of the teaching set that violate  $h^*$  are removed to induce realizability.



# VW hypotheses

- Realizability is a major assumption here.
- STRICT is meant to be used in realizable scenarios; the agnostic case violates many of our assumptions.



## Experiment 2 (BM)

- The BM experiment uses images of various moths and butterflies, and asks participants to determine if they are looking at a moth or a butterfly
- Peacock butterflies and Caterpillar moths are easy to identify, but Ringlet butterflies and Tiger moths are more difficult to classify.



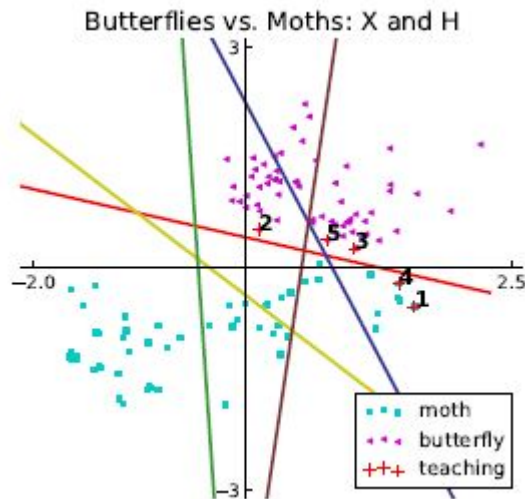
# Experiment 2 (BM)

- 160 total images used for the teaching set, 40 additional images used for testing.
- Euclidean embedding not available due to nature of data.
- Can use Bayesian model to create embedding.



# Embedding BM data

- Requested binary labels (of whether an image is of a butterfly) of all 160 teaching images from 60 Mechanical Turk workers.
- Used the methods of Welinder et al. to create a two dimensional embedding.
- Also generated a linear hypothesis for each of the 60 workers.



# BM hypotheses

- To create the hypothesis class  $\mathcal{H}$ , a random sample of 15 of the 60 generated hypotheses are taken.
- A linear classifier is fitted which best separates the groups, and is taken to be the target hypothesis. Elements of the data set which conflict with this target hypothesis are removed to induce realizability.
- This embedding (without the removed elements) and the hypotheses are now used to teach the crowd. Afterwards they will be assessed with the 40 image testing set.

# Experiment 3 (WP)

- The third experiment involves participants determining whether or not a given image of a woodpecker is that of a Red-cockaded woodpecker.
- Other possibilities are the Red-bellied and Downy woodpeckers..
- 80 images in total are used in the training set, with 40 of them depicting the Red cockaded woodpecker, and 20 each of the Red-bellied and Downy woodpeckers.
- Testing set will be 20 images, with 10 Red-cockaded woodpeckers and 5 each of the other woodpeckers.

Red-cockaded



Red-bellied



Downy





# Embedding the WP data

- An approximate Euclidean embedding can be obtained. 312 binary attributes are considered for each picture, and workers on Mechanical Turk classify them.
- For each attribute (say, `has_forehead_color:black`), the workers indicate that it is 1 (present), 0 (ambiguous or not applicable), or -1 (not present). In this way, an embedding in  $\mathbb{R}^{312}$  is created.
- Working with 312 dimensions can be quite clunky, and may not even yield results that are more precise than simpler approximations.



# Embedding the WP data

- Instead of working with all 312 binary traits, the traits with the highest cross-species variance are selected.
- This reduces the embedding to thirteen dimensions. Still quite hard to visualize, but much more reasonable!



| Features                                  | % feature presence in species |             |       | h* |
|---|-------------------------------|-------------|-------|----|
|   | Red-cockaded                  | Red-bellied | Downy |    |
| has_breast_pattern: <i>solid</i>          | 22                            | 88          | 90    | -1 |
| has_upper_tail_color: <i>white</i>        | 55                            | 91          | 53    | 0  |
| has_bill_length: <i>same_as_head</i>      | 78                            | 93          | 25    | 0  |
| has_bill_length: <i>shorter_than_head</i> | 23                            | 7           | 76    | 0  |
| has_forehead_color: <i>black</i>          | 91                            | 0           | 96    | 0  |
| has_forehead_color: <i>red</i>            | 0                             | 78          | 0     | 0  |
| has_nape_color: <i>black</i>              | 85                            | 0           | 93    | 0  |
| has_nape_color: <i>red</i>                | 0                             | 87          | 15    | 0  |
| has_back_pattern: <i>spotted</i>          | 75                            | 27          | 29    | 0  |
| has_back_pattern: <i>striped</i>          | 18                            | 69          | 7     | 0  |
| has_belly_pattern: <i>solid</i>           | 25                            | 81          | 94    | -1 |
| has_crown_color: <i>black</i>             | 98                            | 0           | 96    | 0  |
| has_crown_color: <i>red</i>               | 2                             | 86          | 36    | -1 |

(c) Features for WP

# WP hypotheses

- Initially, the hypothesis class,  $\mathcal{H}$ , consists of every line in 13 dimensional space in the form  $h(x) = w^T x$  for  $w \in \{+1, 0, -1\}^d$ .
- Essentially a weight of  $\{+1, 0, -1\}$  is placed on each feature, and the resulting line is passed through the origin.
- This is very similar to the VW hypothesis generation.
- Unfortunately, there are 1594323 hypotheses to consider.

# WP hypotheses

- Workers probably will not pay attention to a wide variety of details when examining each woodpecker - most will look at colors and beak shape/size, but potentially little else.
- Thus, consider only hypotheses with non-zero weight for at most three features.
- The target hypothesis is that with minimal error on this data set.
- Now, images which disagree with this hypothesis are pruned to induce realizability.
- Finally, we consider in the final hypothesis class  $\mathcal{H}$  only those remaining with at most two features with non-zero weight.

# WP hypotheses

- Now, the experiment is carried out in a similar fashion to those before it. Mechanical Turk workers will be taught using the pruned training images and will be assessed on the pruned testing set.

| Features                                  | % feature presence in species |             |       | h* |
|---|-------------------------------|-------------|-------|----|
|   | Red-cockaded                  | Red-bellied | Downy |    |
| has_breast_pattern: <i>solid</i>          | 22                            | 88          | 90    | -1 |
| has_upper_tail_color: <i>white</i>        | 55                            | 91          | 53    | 0  |
| has_bill_length: <i>same_as_head</i>      | 78                            | 93          | 25    | 0  |
| has_bill_length: <i>shorter_than_head</i> | 23                            | 7           | 76    | 0  |
| has_forehead_color: <i>black</i>          | 91                            | 0           | 96    | 0  |
| has_forehead_color: <i>red</i>            | 0                             | 78          | 0     | 0  |
| has_nape_color: <i>black</i>              | 85                            | 0           | 93    | 0  |
| has_nape_color: <i>red</i>                | 0                             | 87          | 15    | 0  |
| has_back_pattern: <i>spotted</i>          | 75                            | 27          | 29    | 0  |
| has_back_pattern: <i>striped</i>          | 18                            | 69          | 7     | 0  |
| has_belly_pattern: <i>solid</i>           | 25                            | 81          | 94    | -1 |
| has_crown_color: <i>black</i>             | 98                            | 0           | 96    | 0  |
| has_crown_color: <i>red</i>               | 2                             | 86          | 36    | -1 |

(c) Features for WP

# Experimental Results

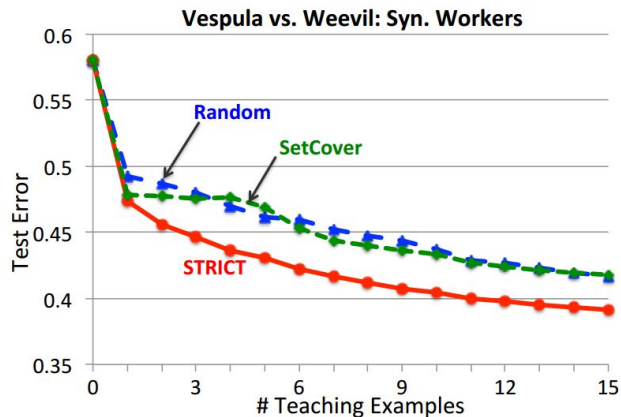
# Experimental Metrics and Baselines

- Experiments include
  - Simulations
  - Annotation tasks on MTurk
    - MTurk - a crowdsourcing Internet marketplace on Amazon to coordinate use of human intelligence to perform tasks that computers can't do yet
- Test error of STRICT compared against 2 baseline teachers
  - Random
  - SetCover (the classical noise-free teaching model)



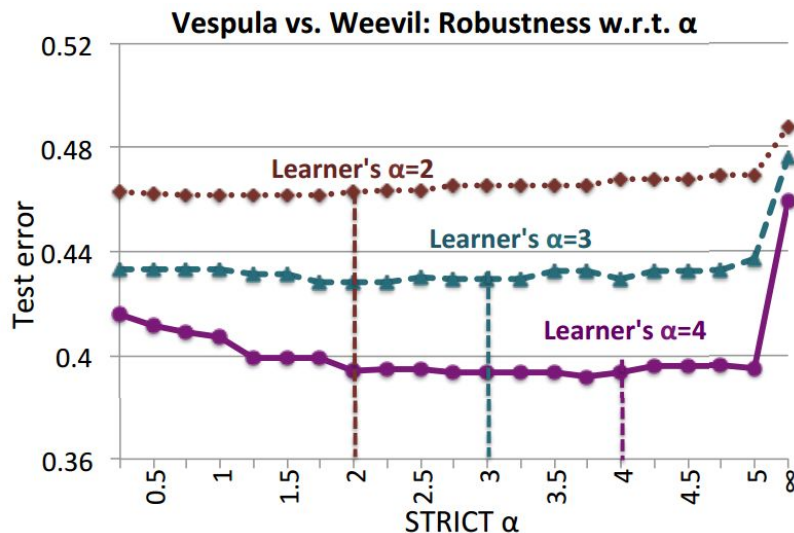
# Results from Simulated Learners

- Simulated 100 learners with  $\alpha = \{2, 3, 4\}$  randomly and different initial hypothesis of learners  $\in \mathcal{H}$
- Results show STRICT generates the lowest test error of the 3 teachers in simulated experiments
- Graph below is for  $\alpha = 2$  for STRICT for VW dataset



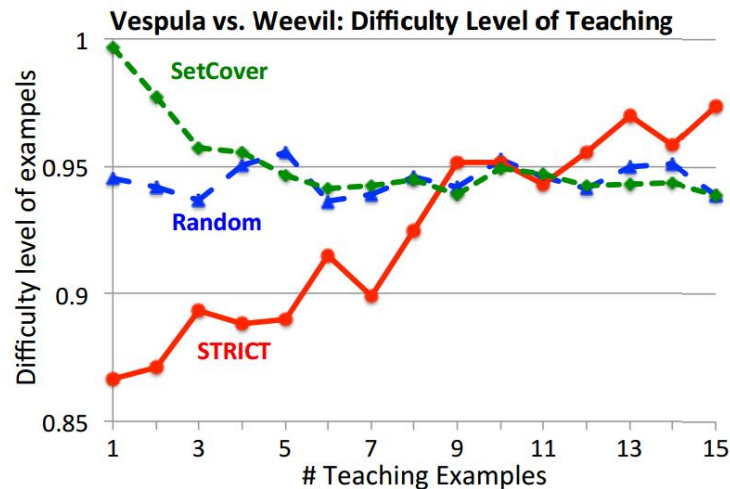
# Robustness of STRICT for Mismatched $\alpha$

- STRICT is robust despite not knowing value of  $\alpha$  (true for real world tasks)
- Experiment tested STRICT with  $\alpha = \{2, 3, 4\}$
- Graph reveals performance is just as good as knowing  $\alpha$ , up to  $\alpha = 5$



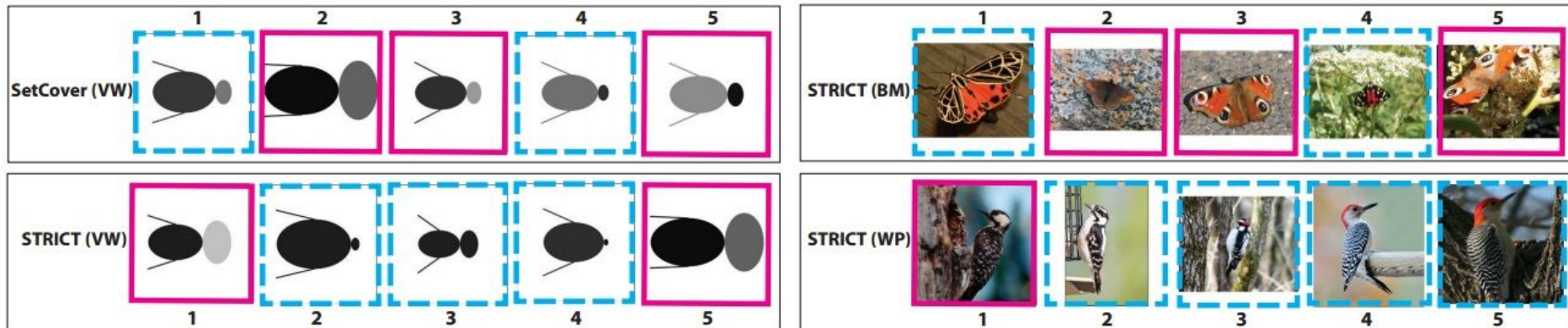
# Difficulty Level of Teaching

- Measured in terms of expected uncertainty (entropy)
- SetCover
  - Assumes learner is perfect
  - Starts with difficult lessons, then randomly selects lessons
- STRICT follows a curriculum based learning
  - Lessons get increasingly hard
  - A useful teaching mechanism proved by Basu & Christensen (2013)



# Results on MTurk Workers

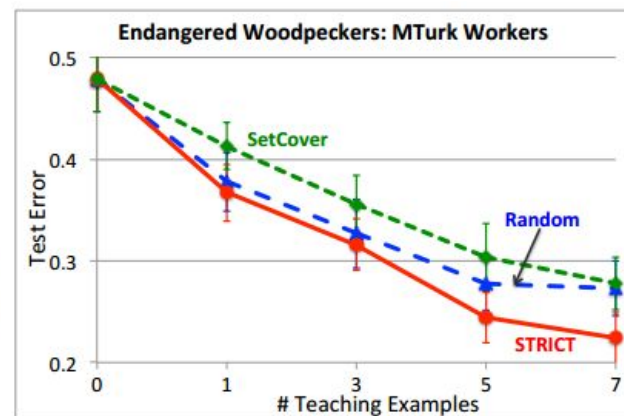
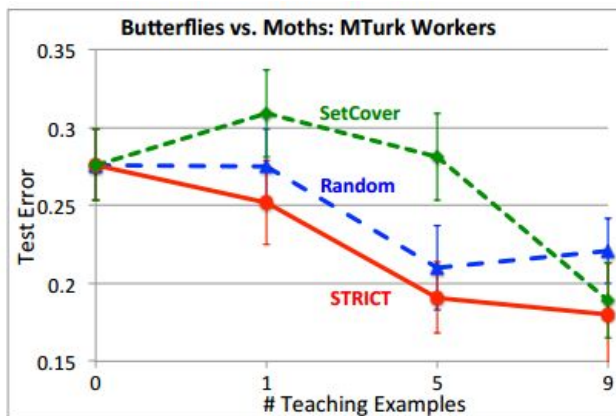
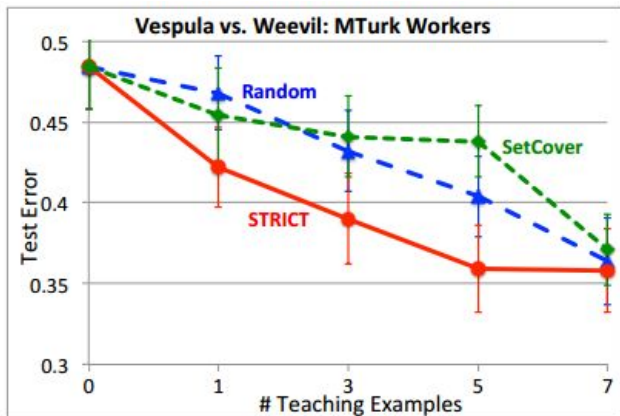
- Teaching sequence generated for STRICT, Random and SetCover
- $\alpha = 2$  for STRICT
- Feature spaces  $\mathcal{X}$  and hypothesis spaces  $\mathcal{H}$  used
- 780 workers recruited for VW dataset; 300 for BM; 520 for WP
- 10 examples for VW and BM; 16 examples for WP



(a) Examples in teaching sequence

# Test Error Results for MTurk Workers

- Accurate improvement as workers classify unseen images for all 3 datasets
- Results from both simulated and MTurk workers show that teaching does help



# Does STRICT Teaching Help?

- Improvement in STRICT is monotonic w.r.t the length of the teaching phase
- Welch's t-test shows STRICT significantly improves classification accuracy
  - $P < 0.001$  for VW and WP datasets
  - $P = 0.01$  for BM task
- VW results:
  - STRICT is significantly better than SetCover ( $P = .05$ )
  - STRICT is significantly better than Random ( $P = .05$ )
- WP results:
  - STRICT is significantly better than SetCover ( $P = .002$ )

# Conclusion

# Summary

- Proposed noise-tolerant stochastic model of learning process in crowdsourcing classification tasks
- Developed STRICT that uses this model
  - Increased robustness
  - Convergence to a desired error rate
  - Effectiveness of teaching approach
- Future applications with STRICT include:
  - Data-driven online education
  - Tutoring systems
- Learner model not necessarily good approximation for real human behavior, but nonetheless the experiments show interesting results



Questions?