Development of Unsupervised Model Using RBM for the Purposes of Cultural Classification

Project Advisor - Dr. Mohsen Beheshti
Committee Member - Dr. Roman Tankelevich
Committee Member - Dr. Alireza Izaddoost

Brian Reza Smith
Computer Science Masters Project
California State University Dominguez Hills
2021

Presentation Overview

1. Introduction/Motivation
2. High-Level Project Milestones
3. Problem Domain: What is culture?
4. What are Restricted Boltzmann Machines?
5. Related Work
6. The Experiments
7. The Results
8. Future Steps
9. References

Introduction / Motivation

The purpose of this project is to use machine learning to better understand people. Understanding a person's cultural affiliations can have broad organizational benefits (team cohesion, hiring, role placement) and personal benefits (self-improvement, building relationships). Artificial neural networks have provided new powerful tools towards the analysis of problems involving large amounts of data. People and the concept of culture are particularly complex and dynamic constructs that seem to be an appropriate problem domain for unsupervised machine learning. An excellent source of data about people is direct input from those people about themselves. The benefit is that it is undistilled and raw, but the downside is that people have a tendency to make misattributions about themselves.

This sets the design constraints of this model for the classification of culture. On one hand, all aspects of it should be data-driven and defined by the members of the cultures that are being modeled. On the other hand, people can not be trusted to define themselves too directly.This is where machine learning comes in. Unsupervised learning can pick up on patterns people may not be aware of. Labels can later be applied to various groupings in ways that circumvent those misattributions.

High-level Project Milestones

1. **Understand problem domain**
2. **Core learning mechanism**
    a. Select appropriate type of data (binary for convenience)
    b. Select appropriate type of model
    c. Implement model
    d. Identify optimal model parameters
    e. Test model
3. **User-facing application (future)**
    a. Interface
    b. Survey
    c. Results
    d. Label voting system
    e. Adding new survey questions

Problem Domain: What is culture?

The definition of a particular culture can be characterized by 3 points:

1. **It must be meaningful to those that participate in the culture (Boyd 2009)**

2. **It must work at all levels of human organization from simple units to complex multi-level ones(Geertz, 1978)**

3. **It must be ever-changing (Fisher 2007).**

Problem Domain: What is culture? Relation to Model

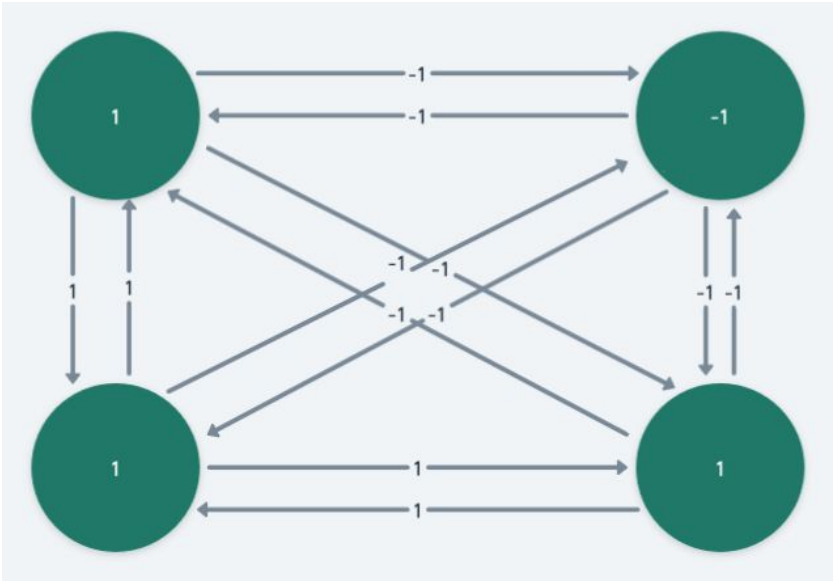
Encompass all levels of human organization: many different groupings of people must be able to be identified as cultural groups. "American" is a cultural group as well as "Los Angelino", "Mathematician", or "Foodie". Individuals may belong to multiple cultural groups. Model must be able to make multiple classification attributions for single sample.

Ever-changing: the model chosen must also be able to change. As time progresses new data should be able to update the model, and observations that once suggested that a person was a "Foodie" might now be used to suggest the person is a "Chef". On top of this, new known features should be able to be added (like new survey questions) without breaking the original model or invalidating old observations.

What is an RBM? Hopfield Neural Networks and Associative Memory

- Ising Spin Model -> Hopfield Net -> Boltzmann Machine->Restricted Boltzmann Machine
- Hopfield Net: Graphical model to store memories, allow retrieval of memories given partial memories
- After training model with memories

  [
  [1, -1, 1, 1],
  [1, 1, -1, -1],
  [-1, -1, -1, -1]
  ]
- Then setting node states to similar to first memory:
  - [1, 1, -1, 1]
- Model will auto-correct by flipping node states until it reaching closest stable state: first memory



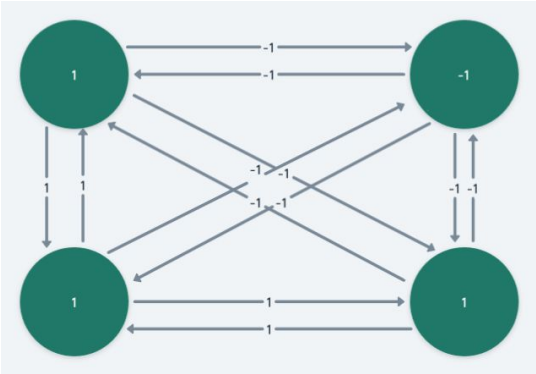
What is an RBM: Hopfield Neural Networks and Associative Memory

$$field(i) = \Sigma_{j \neq i} W_{j,i} x_j + b_i$$

$$state(i) = \begin{cases} state(i) & if sign(state(i)) = sign(field(i)) \\ state(i) * (-1) & if sign(state(i)) \neq sign(field(i)) \end{cases}$$

$$Energy(systemState) = -\Sigma_{i,i \neq j} state(i) * state(j) - \Sigma_i b_i * state(i)$$

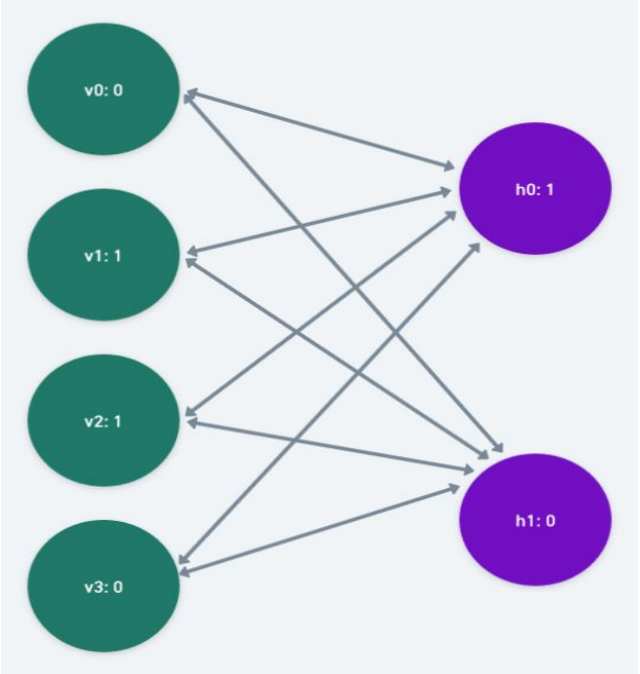
- After adjusting weights through training, energy function quantifies how far current system state is from stable state
- Goal in training is to decrease energy for more probable states (memories we wish to store)
- Stored memories are local minima of Energy function
- RBM uses Energy function in many places



What is an RBM: Differences from HNN

How RBM differs from HNN

- Restrictions on edges between nodes (unlike Boltzmann Machines also)
- Stochastic as opposed to deterministic
- Encoded data is probabilities instead of node states
- Bipartite graph with 2 sets of layers: hidden and visible
- Each node in visible connected to each node in hidden with bidirectional edge
- Nodes within a layer are not connected to each other



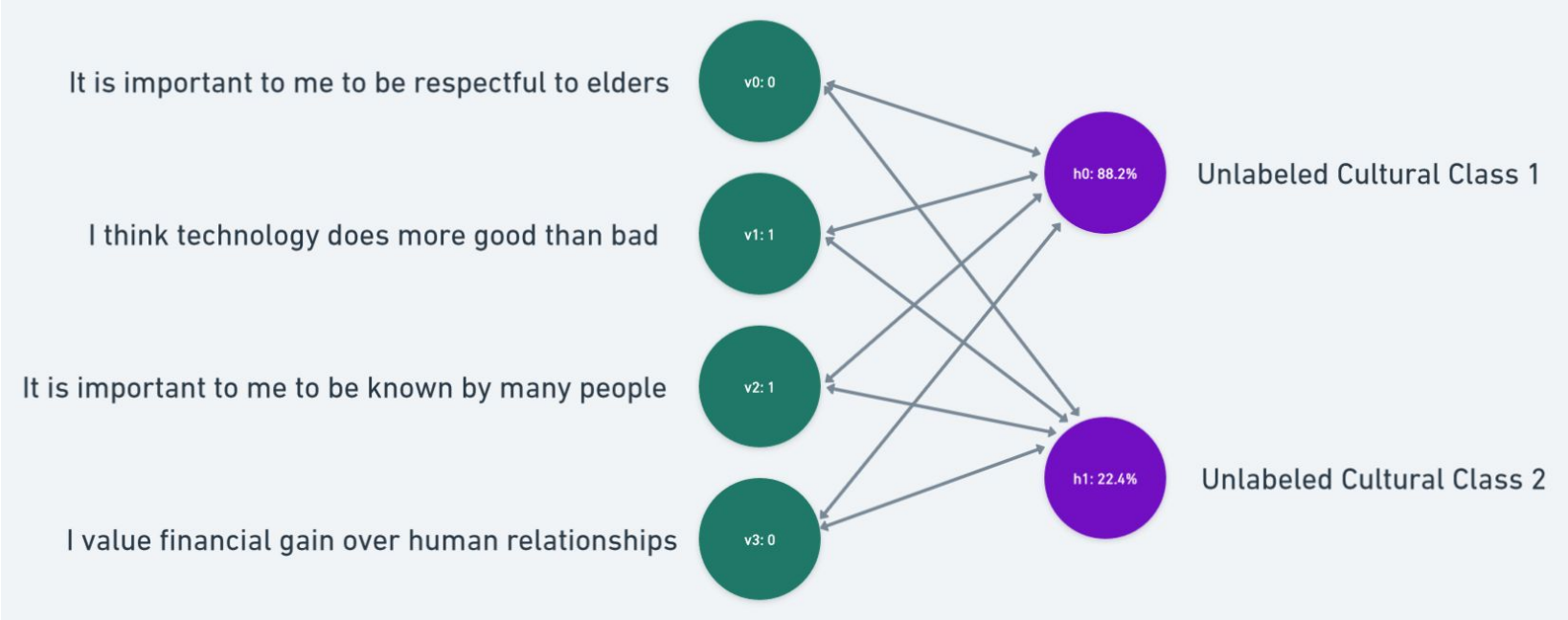
What is an RBM: Why RBM is Right Tool for the Job

- Statistical tool: unlike clustering, can tell be the probability new sample (respondent) is associated with 1 or more latent features (cultural classes)
- Relatively computationally efficient: after training, calculating probability of hidden node activation in linear time
- Stochastic: better able to cope with unpredictable human data. More robust to overfitting.
- Still being developed: advancements made in training algorithms as recently as 2008(Tieleman, 2008)
- Generative: can reconstruct incomplete samples. This can be used when new survey questions are added and older samples are missing values.
- Neural Networks are cool!

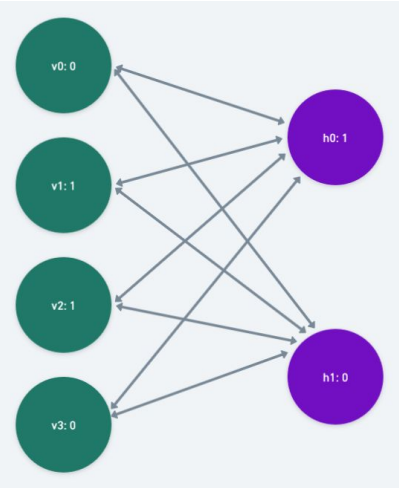
What is an RBM: How Can We Relate to Culture?

Once RBM has been trained, we can make probabilistic assertions about association between respondents and cultural classes.
In this case, a respondent has answered true / false questions whose values correspond to visible layer states.

It is hypothesized that hidden layer node activations correspond with cultural classes.



What is RBM: Boltzmann Energy and Joint Probability



c = column vector of visible node biases with j elements
b = column vector of hidden node biases with k elements
W = j x k matrix of weights between hidden and visible nodes
X = set training set of column vectors j x 1

$$E(x,h) = -\sum_j \sum_k W_{j,k} x_j h_k - \sum_j x_j cj - \sum_k h_k b_k$$

$$p(x,h) = \frac{e^{-E(x,h)}}{Z}$$

Z is normalizing factor: sum of all $e^{-Energy}$ values for all permutations of x and h. It is intractable to compute and so joint probability for particular configuration is also intractable.

However, conditional probability of node activation given other node activations are not intractable...

What is RBM: Conditional Probability

$$p(h_i = 1|x) = logistic(b_i + \sum_j W_{j,i} x_j)$$

$$p(x_m = 1|h) = logistic(c_m + \sum_k W_{m,k} h_k)$$

$$p(h|x) = \prod_k p(h_k|x)$$

$$p(x|h) = \prod_m p(x_m|h)$$

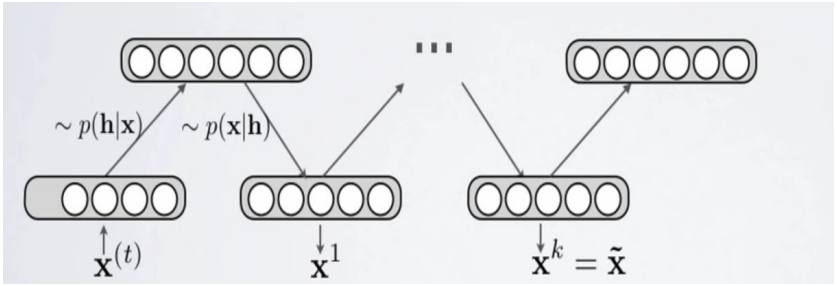
Same idea for sampling in either direction.
We will generate layer states for both training and for reconstructing incomplete samples.

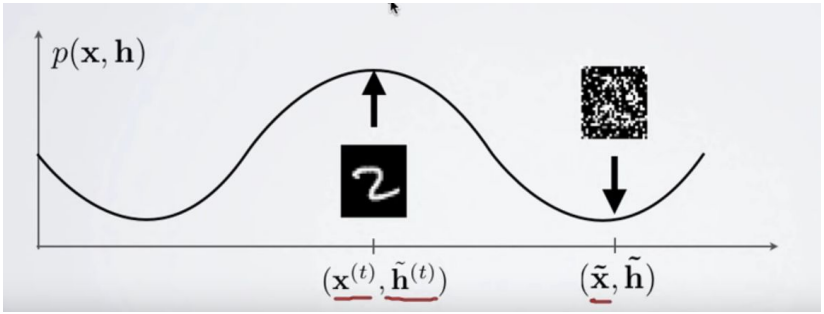
What is RBM: Generating Samples Through Conditional Probability

```
def probXGivenHVector(self, vIndex, hv):
        """
                @param int vIndex : index of visible node for which we wish to calculate activation probability
                @param colVector hv : binary hidden value activations from which we wish to calculate activation of particular visible node

                @return float probability : activation probability
        """
        #linear combination on hv
        sumOfVectorWeightProducts = 0
        for i in range(len(hv)):
                sumOfVectorWeightProducts += hv[i][0]*self.weights[vIndex][i]
        return logistic(self.visibleBiases[vIndex][0] + sumOfVectorWeightProducts)

def expectedXGivenHVector(self, vIndex, hv):
        """
                Notice comparison to random samples from uniform distribution.
                Here we can see stochastic nature of RBM: even with strong collective push from neighboring weights/node states, target may not fire.

                @param int vIndex : index of visible node for which we wish to calculate state
                @param colVector hv : binary hidden value activations from which we wish to calculate particular visible node state

                @return int activation : 0 or 1 value for visible node state
        """
        probX = self.probXGivenHVector(vIndex,hv)
        random.seed()
        randUniform = random.random()
        if randUniform <= probX:
                return 1
        else:
                return 0
        #return round(self.probXGivenHVector(vIndex, hv))


def expectedXVectorGivenHVector(self, hv):
        """
                @param colVector hv : binary hidden value vector from which to generate an entire visible layer activation vector

                @return colVector generatedVisibleLayer : binary visible layer vector from given hidden layer vector
        """
        tempXVector = np.zeros((self.visibleUnitCount,1))
        for i in range(self.visibleUnitCount):
                tempXVector[i] = self.expectedXGivenHVector(i, hv)
        return np.array(tempXVector)
```

$$p(x_m = 1|h) = logistic(c_m + \sum_k W_{m,k} h_k)$$

What is RBM: Training RBM Through Contrastive Divergence





(Larochelle, 2013)

1. Generate samples using Gibbs Sampling
2. Adjust weights/biases to suggest dataset values are *more likely*
3. Adjust weights/biases to suggest model generated values are *less likely*
4. RBM is trained when samples and their reconstructions overlap. In this project I use metrics based on hamming distance.

What is RBM: RBM Learning Rules

Training an RBM involves update weights and bias values. This can be done in multiple passes through the training set.

For each sample in training set:

$$W = W + \alpha(\tilde{h}x_t\top - h^k x^k\top)$$
$$c = c + \alpha(x_t - x^k)$$
$$b = b + \alpha(\tilde{h} - h^k)$$

What is RBM: Training RBM Through Contrastive Divergence

```
def train(self,learningRate, gibbsIterations, timesThroughTrainingSet):
        """
                one stop shop for batch training of RBM. set all training parameters here.

                @param float learningRate : weight applied to incremental adjustments made to learned values (weights, biases)
                @param int gibbsIterations : number of times to chain generating visible and hidden vectors
                @param int timesThroughTrainingSet : number of cycles through the training set to update learned values for each training sample

                @return None
        """
        for trainingSetIterations in range(timesThroughTrainingSet):

                #metric for evaluating reconstruction error
                sumAcrossSamplesForPercentGeneratedCorrectly = 0

                #for all samples in training set
                for i in range(len(self.trainingSet)):

                        #save visible sample from training set, and hidden vector generated from training sample
                        hiddenFromTrainingVisible = self.expectedHVectorGivenXVector(self.trainingSet[i])

                        #initialize fabricated visible and hidden vectors to be generated using gibbs sampling
                        gibbsVisible = np.copy(self.trainingSet[i])
                        hiddenFromGibbsVisible = np.zeros((self.visibleUnitCount))

                        #do gibbs sampling to set values for fabricated visible and hidden samples
                        for gibbsIter in range(gibbsIterations):
                                hiddenFromGibbsVisible = self.expectedHVectorGivenXVector(gibbsVisible)
                                gibbsVisible = self.expectedXVectorGivenHVector(hiddenFromGibbsVisible)


                        #weights += learningRate (hiddenFromTrainingVisible*trainingVisible) - (hiddenFromGibbsVisible * gibbsVisible)
                        posPhase = np.matmul(self.trainingSet[i], np.reshape(hiddenFromTrainingVisible, (1, self.hiddenUnitCount)))
                        negPhase = np.matmul(gibbsVisible, np.reshape(hiddenFromGibbsVisible, (1, self.hiddenUnitCount)))
                        self.weights = np.add(self.weights, (learningRate * np.subtract(posPhase, negPhase)))

                        #hidden += learningRate (hiddenFromTrainingVisible - hiddenFromGibbsVisible)
                        self.hiddenBiases = np.add(self.hiddenBiases, (learningRate * (np.subtract(hiddenFromTrainingVisible, self.expectedHVectorGivenXVector(gibbsVisible)))))

                        #visible += learningRate(trainingVisible - gibbsVisible)
                        self.visibleBiases = np.add(self.visibleBiases, (learningRate * (np.subtract(self.trainingSet[i], gibbsVisible))))

                        #for each sample, add % of visible units reconstructed correctly to sum. If model has been trained, this should approach 100%
                        sumAcrossSamplesForPercentGeneratedCorrectly += self.percentOfOverlapBetweenVectors(self.trainingSet[i], gibbsVisible)

                #avg trainingVisible == generatedVisible for trainingVisible->hidden->generatedVisible
                #across all training set items
                self.logFileObject.write(f"RBM.train:: {learningRate} {gibbsIterations} {trainingSetIterations} {sumAcrossSamplesForPercentGeneratedCorrectly / len(self.trainingSet)}\n")
```

Related Work

- Many statistical models used in sociological studies on culture, but none using artificial neural network

- RBMs can be used as classifiers by using labeled data. *"If labeled training data is given, and the RBM is trained on the joint distribution of inputs and labels, one can sample the missing label for a represented image from the distribution or assign a new image to the class with the highest probability under the model"* (Fischer, 2014)

- In an article entitled "Growing Restricted Boltzmann Machine on the Fly for Unsupervised Representation" Savitha et al. offered a hidden unit count discovery solution that made the decision as training examples streamed in. Based on a 'uniqueness score' of each new training set element, decide to:
    - add a new hidden unit
    - update the weights and biases
    - or if the sample was not very unique, to pass on

Finally, the Experiments: How to Discover RBM Parameters?

- After generating 3 datasets to simulate populations with multiple cultural groups my main focus became ways to discover the appropriate parameters, namely number of gibbs iterations, learning rate, and number of hidden units.

- Initial weights and biases were set following Hinton's "A practical Guide to Training Restricted Boltzmann Machines".

- "what is the smallest number of hidden units I can get away with?"

- learning rate and gibbs sample iterations were adjusted incrementally after observing reconstruction error across all samples

- H was set using 3 strategies:
  1. Prior knowledge that my features in samples in my population have some number of elements that covary

  2. An intuition that maybe there is a saturation point where H >= number of visible features

  3. A strategy using k-modes clustering algorithm

K-modes Hypothesis

```
k-modes(k, vectorList):
        Randomly select k samples from vectorList to serve as initial "mode" for each cluster
        cluster each element from vectorList such that each element is in cluster with closest mode (hamming distance)
        While any samples have been moved between clusters
                Calculate new mode for each cluster
                Shuffle samples such that each element is in cluster with closest mode (hamming distance)
        return clustersOfSamples
```

It is hypothesized that the smallest number of hidden units still able to reconstruct samples accurately can be found by:

1. Incrementally clustering training samples using Kmodes with k = k+1 each iteration
2. For each set of clusters, calculate hamming distance between each node and it's respective cluster mode
3. Find average across all samples

The expectation is that, when k is small, the average hamming distances would be high because fewer cluster modes would be near each samples. Avg hamming distance will decrease as k increases up until a point where the modes are in positions that reach samples in a balanced way.

The number of clusters where this measure first repeated itself would be a good starting point for the number of latent cultural identities or hidden units, since each sample had found a grouping it was most similar to.

Iterations before Avg Overlap Metric

The evaluation metric for the RBM experiment was based on the accuracy of reconstructions of input samples. If the RBM has correctly encoded the probability distribution of the population the samples come from, then if we generate a reconstruction of a training sample it should match that training sample.

If the model is trained and we have a generated sample using the conditional probabilities of the RBM.

```
generatedX = expectedXgivenH(expectedHGivenX(trainingX))
```

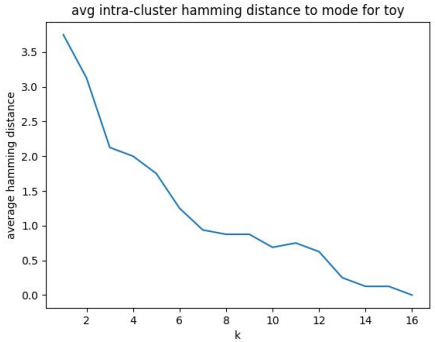
Then:

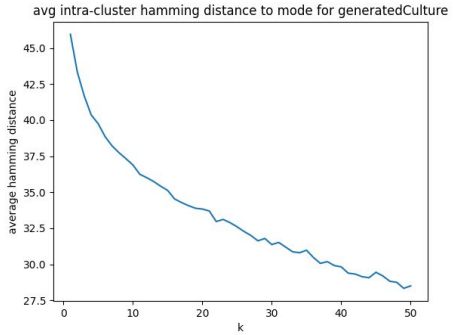
`hammingDistance(generatedX, trainingX) / len(trainingX)` should approach 1

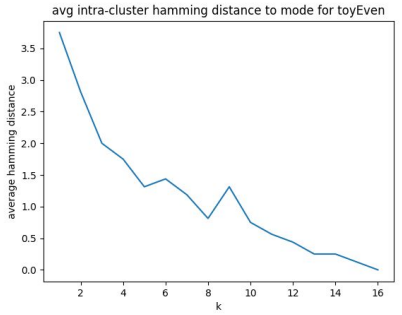
This distance is averaged across each sample for each experiment.

Results

1. The K-Modes method was not useful



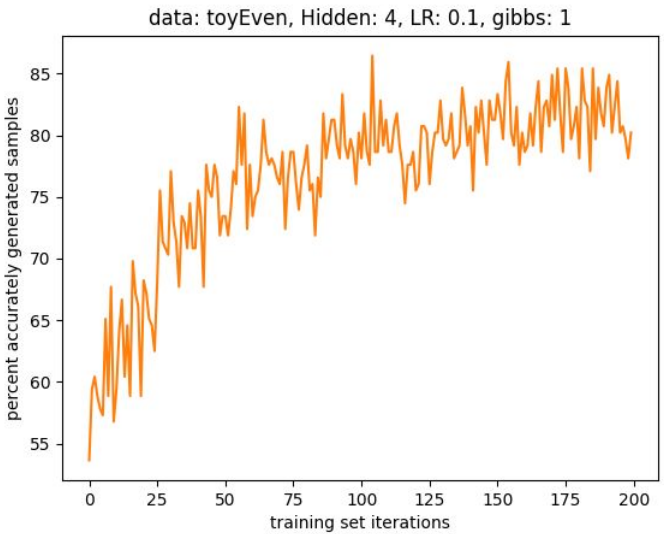


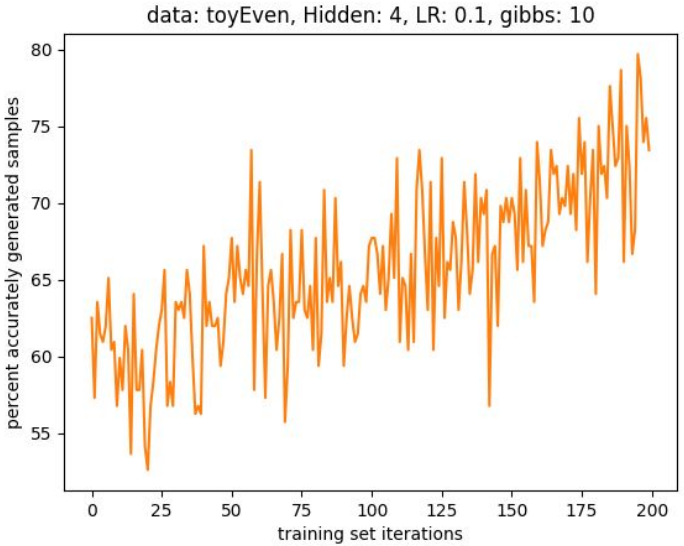


- There was no 'stable point'. Avg hamming distance to mode always decreased as k increased for all datasets
- It was not worth the effort: k-modes took a long time to run and I could have just ran experiments directly on the RBM with different hidden unit values

Results

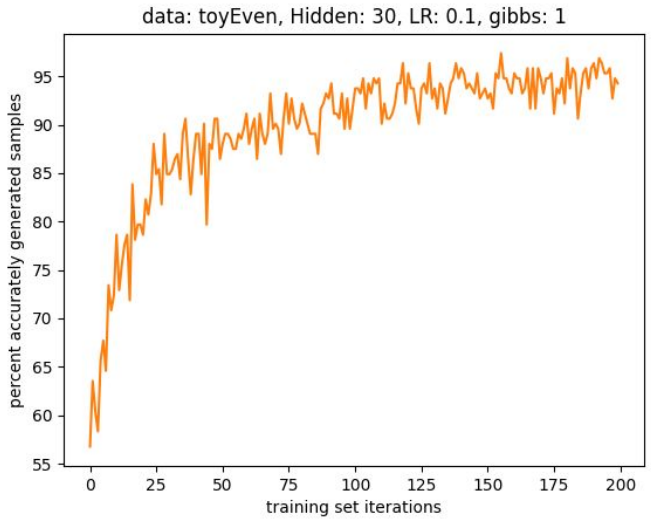
2. In all cases, more Gibbs Iterations led to lower avg overlap between training samples and generated samples.

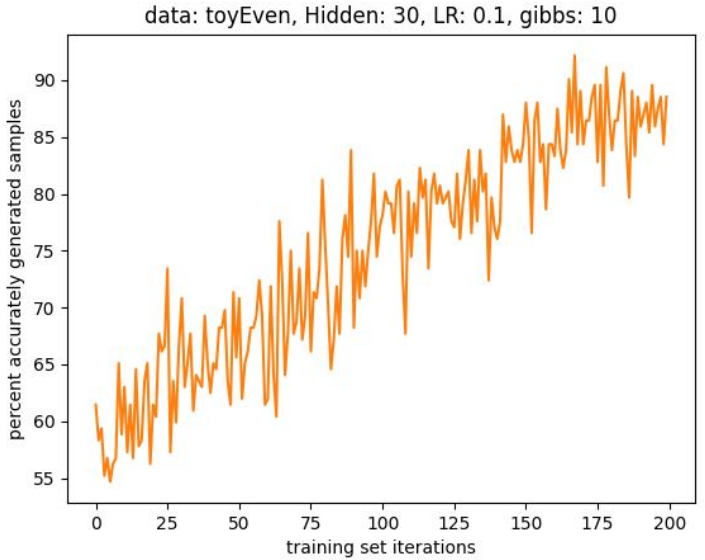




Results

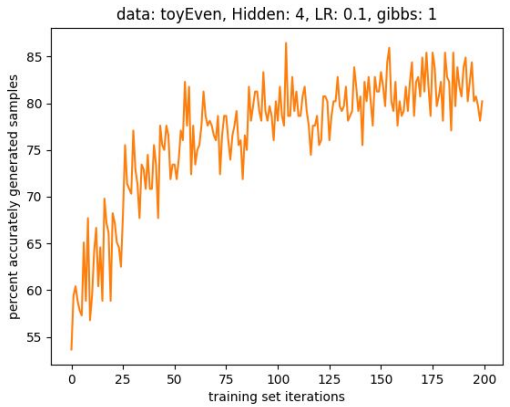
2. In all cases, more Gibbs Iterations led to lower avg overlap between training samples and generated samples.

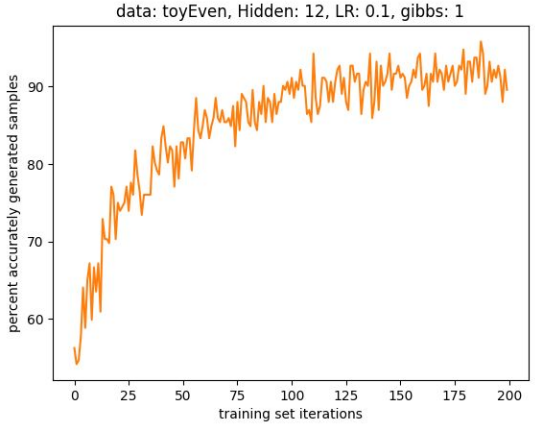


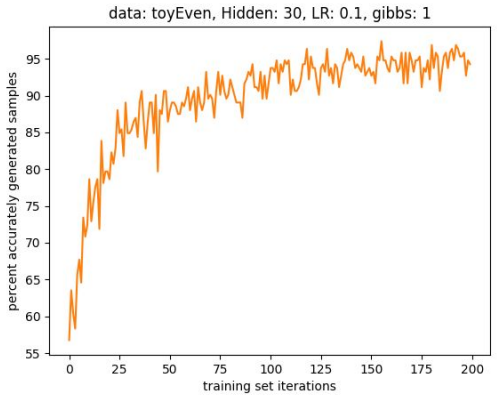


Results

3. The biggest beneficial impact to accuracy was increasing the hidden unit count. Accuracy benefitted from hidden unit count higher than visible unit count suggesting rbm is encoding more than simple associations with visible features.

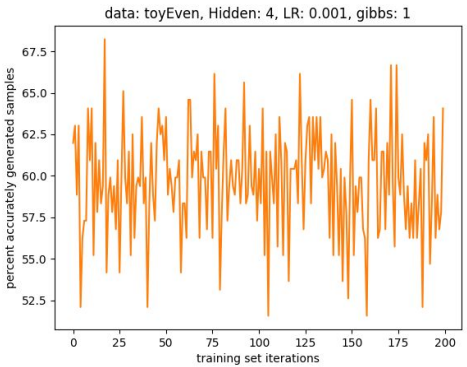


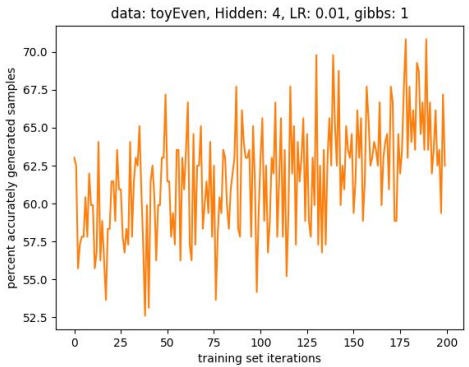


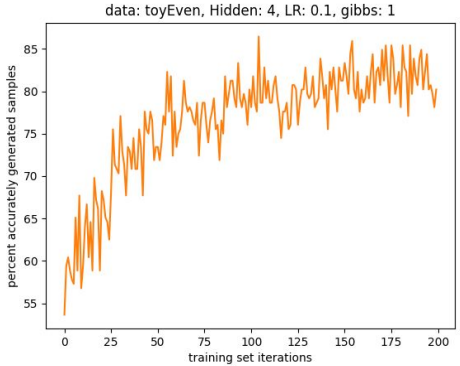


Results

4. Learning rate required the most manual fine-tuning. For similar data, try lr = .1

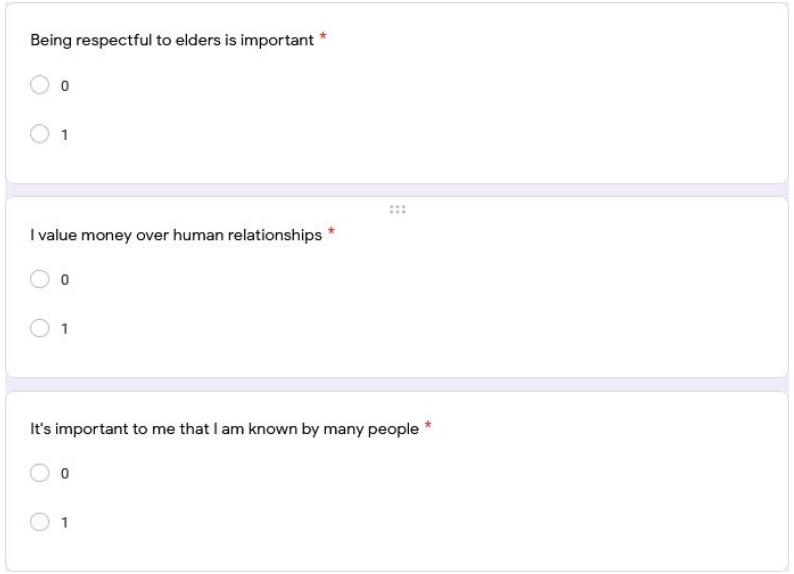






Future Directions

Collect actual human data:

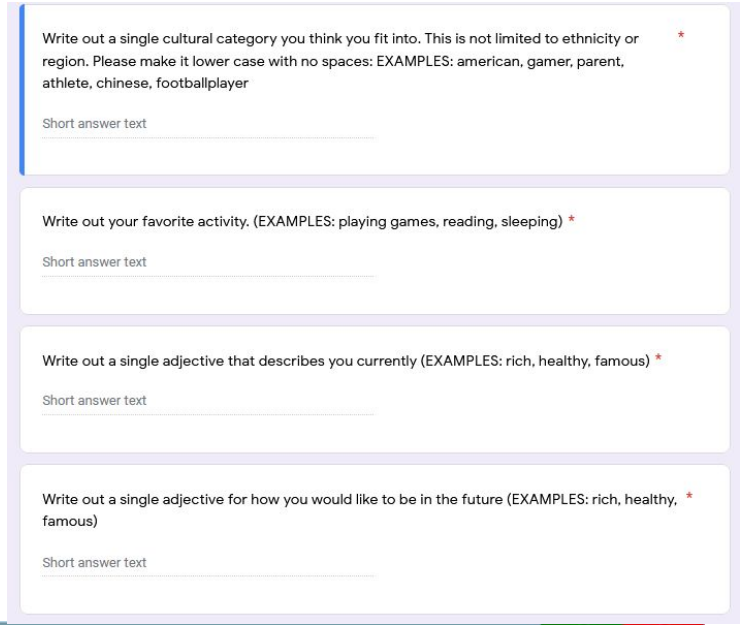


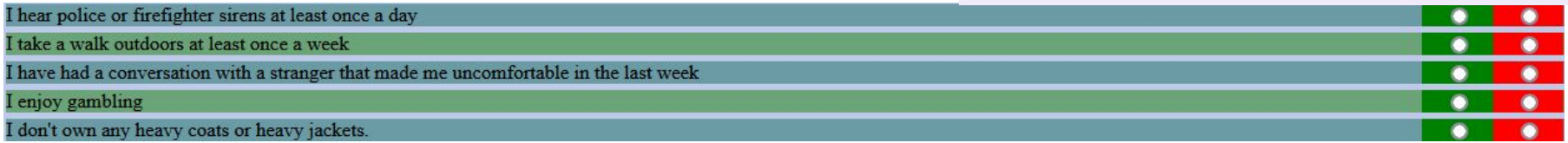
main.py + (~\dynamicSurvey) - VIM2

```
68        #clear any previous data in log
69        fo.truncate(0)
70
71        #pick dataset
72        if config['DATASET'] == 'toy':
73                #TOY MOVIE DATASET: 25% romance, 75% other
74                SHEET_ID = '1QtGC9wmIANYb7DFBBVMrbRqh4QZsx0LCzn8WWH1D474'
75                SHEET_RANGE = 'B2:M17'
76        if config['DATASET'] == 'toyEven':
77                #TOY MOVIE DATASET: 25% romance, 25% scifi, 25% action, 25% random
78                SHEET_ID = '1d2SJXqawVG5g7SuI6-XU6cST392m7h3j6Y-Ngl5gwpo'
79                SHEET_RANGE = 'B2:M17'
80        elif config['DATASET'] == 'culture':
81                SHEET_ID='17y07Dg89WMv-bOZkh_GxhuoZJPjKGw9Xg3R5am_nMOs'
82                SHEET_RANGE = 'F2:BA60'
83        elif config['DATASET'] == 'generatedCulture':
84                #300 samples, 100 features, 30 samples per pattern
85                #see generateData.py for details
86                SHEET_ID='1oGcEJ_xQFHz61HgFv2gdxM65C6NG3AQEz6o0HD30Rz0'
87                SHEET_RANGE = 'Sheet1!A1:CV300'
88
89
90
91        #get 2d list from google sheets
92        service = get_service('sheets', 'v4', './serviceCred.json')
93        sheet = service.spreadsheets()
94        result = sheet.values().get(spreadsheetId = SHEET_ID, range=SHEET_RANGE).execute()
95        values = result.get('values', [])
96
97
98        #convert list to numpy array of column vectors
99        numpyTrainingSet = []
100       if not values:
101               print('no data found')
102       else:
103               numpyTrainingSet = np.asarray(values, dtype=np.int8)
104
105       numpyTrainingSet = np.reshape(numpyTrainingSet, (len(numpyTrainingSet),len(numpyTrainingSet[0]),-1))
106
```

Future Directions

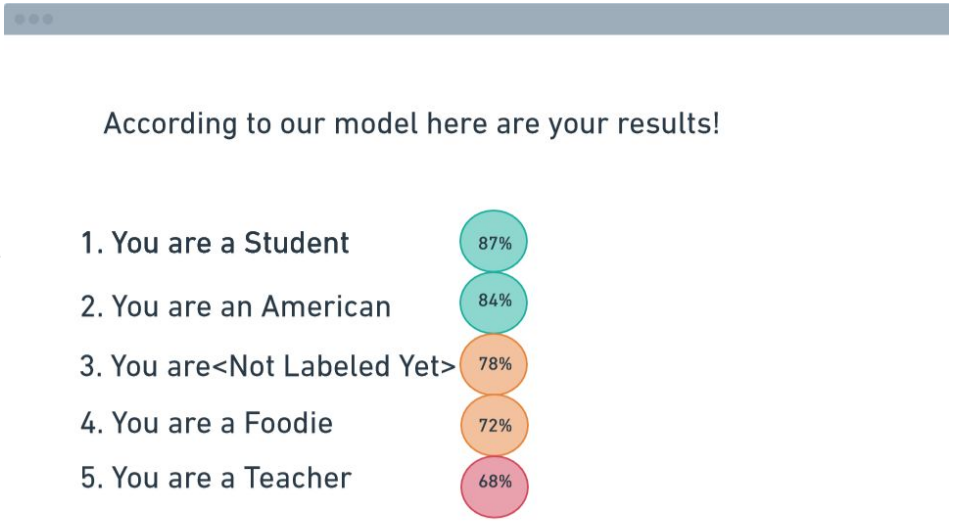
Implement Dynamic User-Facing flow for taking surveys, including binary questions and qualitative questions used in label voting system.





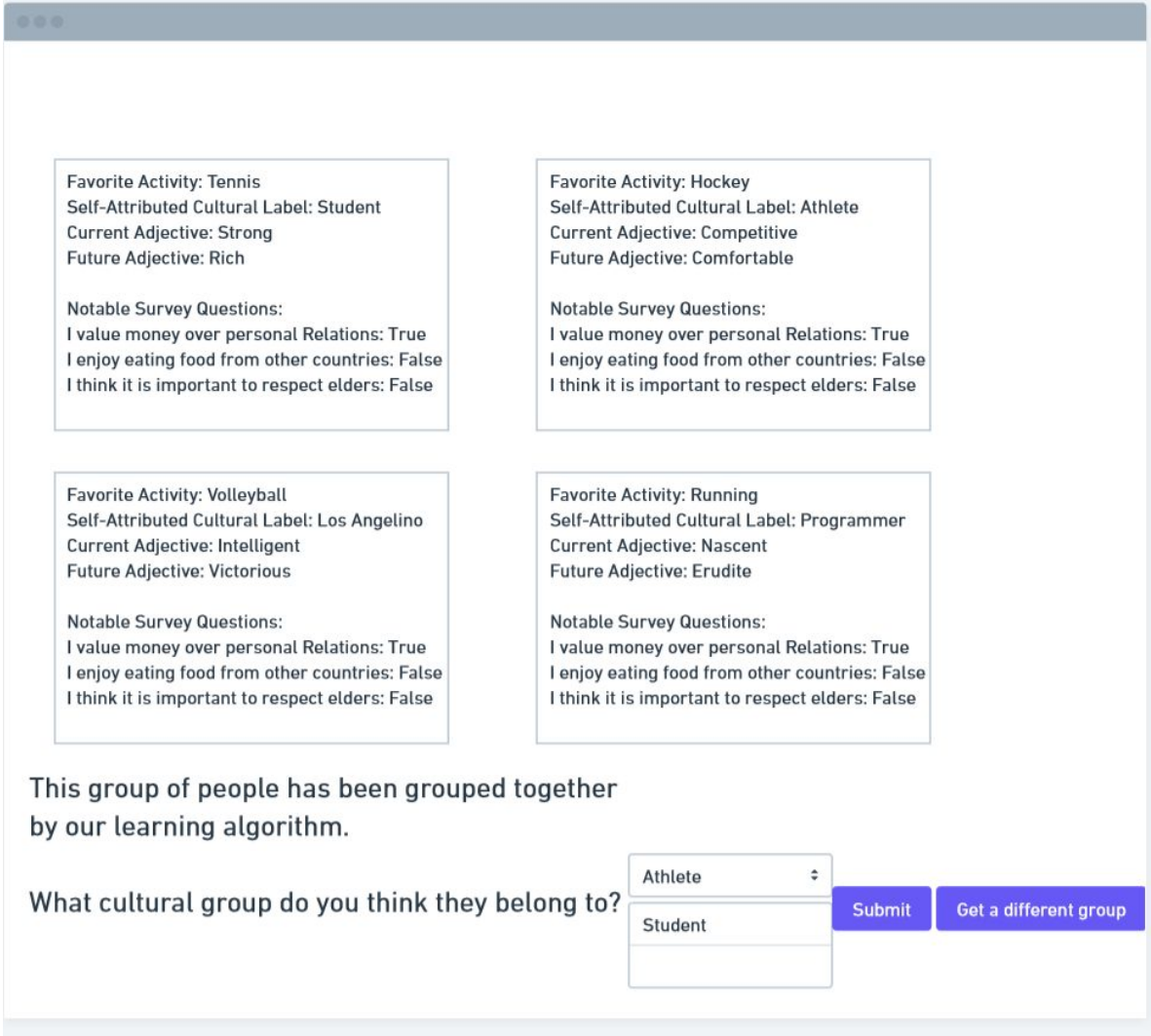
Future Directions

Implement results view.
For respondent, calculate probability for each hidden node activation and keep track of 5 highest



Future Directions

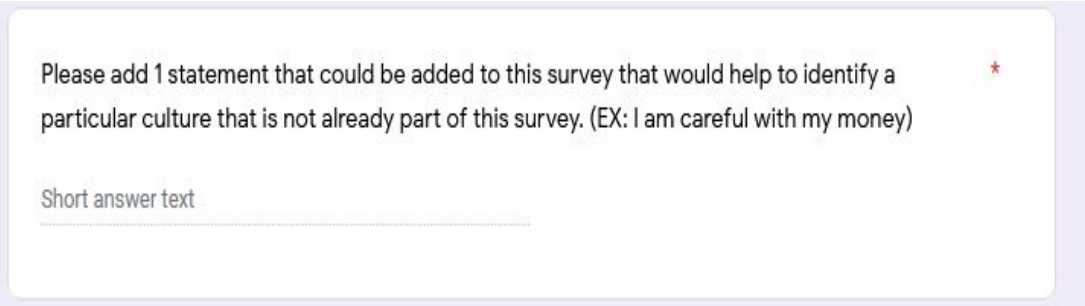
Implement label voting system



Future Directions

Allow users to submit new cultural features.

This would necessitate generating visible layer values for older incomplete samples.



Thanks

Thank you to my esteemed committee members for taking times from their busy schedules to assist me and review my work.

Special thanks to Dr. Tankelevich for spending many hours in video chat and exchanging many many emails with me.

References

Boyd, D. (2009, December 10). People looked at me like I was an alien. The Guardian, 5.

Fischer, Asja & Igel, Christian. (2014). Training restricted Boltzmann machines: An introduction. Pattern Recognition. 47. 25-39. 10.1016/j.patcog.2013.05.025.

Fisher, M. M. J. (2007). Culture and cultural analysis as experimental systems. Current Anthropology, 22 (February), 1-62.

Geertz, H. (1978). The interpretation of cultures: Selected essays. New York: Basic Books.MLA 8th Edition (Modern Language Assoc.)

Hinton, G.E. (2012). A Practical Guide to Training Restricted Boltzmann Machines. *Neural Networks: Tricks of the Trade*.

Hopfield J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

Larochelle, H. [Hugo Larochelle] (2013, November 15). *Neural networks [5.4] : Restricted Boltzmann machine - contrastive divergence* [Video]. Youtube. https://www.youtube.com/watch?v=MD8qXWucJBY

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. *Proceedings of the 25th International Conference on Machine Learning*, 1064–1071. https://doi.org/10.1145/1390156.1390290