Training Ising Models on Images using Sparsitron

Author:
Guneet Singh Dhillon

Supervisor:
Dr. Adam Klivans

Co-Supervisor:
Dr. Philipp Krähenbühl

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science in Computer Science: Turing Scholars Honors in the Department of Computer Science

May 14, 2018
Learning the distribution of data is an important machine learning problem. Once the distribution is learnt, it can be used for learning, reasoning, interpreting and predicting. In this paper, we attempt to learn the distribution behind dataset of images like MNIST and CIFAR-10, using Ising models that are trained using the Sparsitron algorithm.

We motivate the assumption that the underlying distribution of the images is an Ising model as it is equivalent to saying that each pixel in the image is a generalized linear model, or that it is modeled by a single layered neural network with a sigmoid activation. This is more powerful than convolutional filters that are linear models.

We analyze the Ising model assumption and also the Sparsitron algorithm in learning such Ising models. We test models by calculating the mean likelihoods obtained on the validation datasets. We see that the Ising models perform well with regards to this measure. We also study the convergence time Sparsitron and notice that it converges much faster than what its theoretical bound requires.
Acknowledgements

I would like to convey my sincere gratitude to Dr. Adam Klivans and Dr. Philipp Krähenbühl who guided and motivated me throughout my research. Their invaluable comments and feedback helped me stay on track and stay focused.

I would also like to thank the Department of Computer Science at the University of Texas at Austin. I have learnt a great deal by taking courses and interacting with the well renowned faculty members throughout my undergraduate studies. This experience has been invaluable to me, and I will continue to push myself to strive for academic excellence.

Last but not the least, I would like to thank my parents, my sister and my friends for supporting and encouraging me throughout all these years, even when I was not able to spend much time with them. They have helped me overcome difficulties and have been with me during stressful times. Without them, it would not be possible to complete my thesis.
3.7.2 Nearest Neighbors Model 16  
3.7.3 Generative Adversarial Network 17  
3.8 Testing 17  
3.8.1 Likelihood 17  
3.8.2 Probability Heat Map 18  
3.8.3 Convergence Time 18  

4 Experiments 19  
4.1 MNIST 20  
4.2 CIFAR-10 21  
4.3 Probability Heat Maps 21  
4.4 Sparsitron Convergence Time 22  

5 Conclusion 33  
5.1 Ising Model Assumption 33  
5.2 Scaling of Interactions 33  
5.3 Low Mean Likelihoods 34  
5.4 Convergence of Sparsitron 34  
5.5 Negation of Images 34  
5.6 Future Work 34  

Bibliography 37  

A Sigmoid Function 39  
A.1 Non-Decreasing 39  
A.2 1-Lipschitz 39  

B Scaling Interactions 41
Chapter 1

Introduction

Machine learning is a field of computer science that deals with techniques to create models or algorithms that can learn from and make predictions about data. A fundamental aspect of most machine learning models is the underlying probability distribution from which the data is produced. The success of such machine learning models eventually boils down to how well they are able to capture the underlying probability distribution.

Using graphical models is one way of assuming a basic layout of the underlying probability distribution. It provides an easy way of representing relationships between different random variables in the model, dependencies that are often seen in real world data. Markov random fields are undirected graphical models, where the relationship between any pair of random variables is interchangeable (non-directional or bi-directional). Ising model (Cipra, 1987) is a special type of Markov random field. It restricts the values of its random variables to be either $-1$ or $+1$, and the interactions between the random variables is defined by the weight of edges in its corresponding graph.

Assuming a graphical model layout is easy. The difficult part is learning the correct parameters of the model. Sparsitron is a multiplicative weight update algorithm, proposed by Klivans and Meka (2017), to learn graphical models. It is a simple algorithm, that breaks down learning of the graphical model into simpler subproblems. In the case of an Ising model, each of these subproblems is equivalent to learning a generalized linear model. This generalized linear model is learnt by iteratively updating the weights of the model, increasing the weights of the features that correctly predict the desired value, and decreasing the weight of the features otherwise. This algorithm intuitively is easy to understand, and uses nearly optimal number of samples to learn Ising models.

In this thesis, we analyze the use of Sparsitron to learn Ising models for the MNIST and CIFAR-10 datasets. The motivation behind representing the probability distribution of the images as an Ising model is that each pixel is treated as a generalized linear model of the other pixel values. So we end up learning a single layered neural network for each pixel, with a sigmoid activation layer. We make an assumption that a given pixel is only related to its neighboring pixels, which is a standard assumption to make
in computer vision tasks. In order to compare different Ising models learnt using Sparsitron, and other models, we will look at the mean likelihood of the models over the datasets, which is the mean probability of an image from the dataset occurring given the learnt distribution. We will also investigate the structure of the images learnt by these models by looking at the probability heat maps generated by the distributions. Additionally, we will analyze the time taken for the Ising models to be learnt using Sparsitron.

This analysis is helpful because it gives us a way of knowing the probability distribution of a set of images. Once we have learnt that, we can use the distribution for synthetic image generation. This could be used for data augmentation purposes, to feed in more data into learning models. It can also be used to differentiate between different distributions. It is hard to do so given just the images from different datasets, but knowing the underlying distributions gives a new form of comparison. Additionally, this can be used to perform the function of the discriminator in a generative adversarial network architecture (Goodfellow et al., 2014). Currently, the generator and discriminator are trained simultaneously under such an architecture, where the generator is supposed to learn to produce synthetic images, and the discriminator is supposed to learn to differentiate between the original and synthetic images. But now we can learn the distribution of the original set of images, and train the generator to produce a set of synthetic images that has a distribution that is close to the original distribution.
Chapter 2

Background

In this chapter, we provide the background on which the thesis is built.

2.1 Notation

We will be using the following notations and conventions in this thesis

- For $x \in \mathbb{R}$, where $x > 0$, we will define the set $[x]$ as $\{1, 2, \ldots, x\}$.
- For $n \in \mathbb{R}$, $x_1, x_2, \ldots, x_n \in \mathbb{R}$ and $x = \{x_i\}_{i=1}^n$, we will define the set $x_\sim$ as $x \setminus \{x_i\}$, for $i \in [n]$.

2.2 Probability Distributions

Probability distributions are mathematical functions that, in simple terms, provide the probabilities of occurrence of different possible outcomes of a given experiment. In this thesis, we will come across the well known Gaussian or normal distribution.

2.2.1 Gaussian or Normal Distribution

A random variable $x \sim N(\mu, \sigma^2)$ is said to have a Gaussian or normal distribution with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$, where $\sigma^2 \geq 0$, if the probability density function of $x$ is given by

$$ f(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right). $$

This is a continuous random variable, where $x$ can take on values in $\mathbb{R}$. 
2.3 Prediction

Given $m$ data samples $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$, with associated values $y_1, y_2, \ldots, y_m \in \mathbb{R}$, the task of prediction is to be able to make a good guess about the associated value of a new data sample, $x \in \mathbb{R}^n$. For example, given data samples that correspond to features of a car, like number of seats, trunk capacity, etc., and their associated costs, the prediction task might be to be able to guess the cost of a new car that you come across.

This involves learning a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, such that $\text{dist}(f(X), y)$ is minimized, for a predetermined distance function $\text{dist}$.

2.3.1 Linear Models

The model is said to be linear if $f$ is a linear function, i.e., $f(x) = \beta_0 + x \cdot \beta_1$, for $\beta_0 \in \mathbb{R}$ and $\beta_1 \in \mathbb{R}^n$. The prediction problem then changes to learning a good set of parameters in the following way

$$\hat{\beta}_0, \hat{\beta}_1 = \arg \min_{\beta_0 \in \mathbb{R}, \beta_1 \in \mathbb{R}^n} \sum_{i=1}^{m} \text{dist}(\beta_0 + x_i \cdot \beta_1, y_i).$$

2.3.2 Generalized Linear Models

The linear function is one of the simplest functions. So it might not always be a good idea to assume that $f$ is a linear function. For example, if one is trying to predict the probability of a person having cancer or not, instead of assuming that $f$ is a linear function, it would make more sense to change $f$ to only output values in the range $[0, 1]$.

A generalization of the linear model is the generalized linear model (GLM), where $f(x) = g(\beta_0 + x \cdot \beta_1)$ for $\beta_0 \in \mathbb{R}$, $\beta_1 \in \mathbb{R}^n$ and for a non-linear function $g$. Now the prediction problem changes to learning parameters in the following way

$$\hat{\beta}_0, \hat{\beta}_1 = \arg \min_{\beta_0 \in \mathbb{R}, \beta_1 \in \mathbb{R}^n} \sum_{i=1}^{m} \text{dist}(g(\beta_0 + x_i \cdot \beta_1), y_i).$$

Sigmoid Function

The non-linear function that we would be using in this thesis is the sigmoid function, $\sigma : \mathbb{R} \rightarrow (0, 1)$. $\forall x \in \mathbb{R}$, the function is defined as

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$
2.4 Graphical Models

Most of the real world domains involve interdependent variables. It is difficult to isolate variables that are truly independent of others. On the other hand, a holistic analysis of all the variable interactions is very expensive. So we need to make some simplifying assumptions of independence to decompose the complex problem into a simpler solvable one.

One such useful assumption is that of conditional independence, where two sets of variables are independent of each other given that some condition holds true. For example, words in a document can be assumed to be independently generated if the topic of the document has been fixed, i.e., words are conditionally independent of each other given that the topic of the document is fixed. In an image, the value of a given pixel can be assumed to have no relation with the pixels far away from it, if the neighborhood of the pixel is known, i.e., pixels are conditionally independent of pixels farther away from them given that the neighborhood pixel values are known.

These assumptions make the problem simpler to solve. However, the examples given above might not always hold true. One might be concerned about the semantic structure of a document. So making assumptions as before does not help solve the problem. Thus, depending on the nature of the problem, we want to vary the level of interdependency and use a framework such that learning, reasoning, interpreting and predicting can be done efficiently.

Graphical models seamlessly integrate graph theory and probability theory and provide with the framework as mentioned above. A graphical model is defined on a graph $G = (V, E)$, with a set of nodes $V$ and a set of edges $E$. Each node in $G$ corresponds to a random variable that is part of the model, and the edges, along with their weights, define the relationship between the random variables corresponding to the end-points of the edges.

There are two types of graphical models - directed and undirected. Directed graphical models, also called Bayesian networks (Pearl, 2011) and belief networks, provide a graphical representation of causalities and influences. Undirected graphical models, also known as Markov random fields and Markov networks, encode correlations between variables with edges between its corresponding nodes.

We will assume that the graph $G$ models $n$ random variables $x_1, x_2, \ldots, x_n$. We will set $V = [n]$, where $i \in V$ corresponds to random variable $x_i$. We will also denote the joint set of random variables by $x = \{x_i\}_{i=1}^{n}$. 
2.4.1 Markov Random Fields

Markov random fields (MRFs) specify a joint distribution $Pr[x]$ over an undirected graph $G$. Under this model, for $i, j \in [n]$ and $i \neq j$, random variables $x_i$ and $x_j$ are conditionally independent of each other if we know values of a subset of random variables that block all paths from node $i$ to node $j$ on $G$.

A Markov blanket is the set of all neighbors of a node. It is a useful set of blocking nodes as once the Markov blanket for node $i$, $i \in [n]$, is known, $x_i$ will be independent of any other random variable in $x_{-i}$. This condition is called the Markov property, and it results in the following factorization

$$Pr[x] = \frac{1}{z} \prod_{c \in C} \phi_c(x_c),$$

where $z = \sum_{x} \prod_{c \in C} \phi_c(x_c)$, where $C$ is the set of all maximal cliques, i.e., completely connected subgraphs, in $G$, and $\phi_c$ defines the probability distribution on the set of nodes in the maximal clique $c$.

2.4.2 Ising Models

Ising models (Cipra, 1987) are a special case of MRFs where each random variable, $x_i \in \{-1, +1\}$, for $i \in [n]$. Given the weight matrix of $G$, $A \in \mathbb{R}^{n \times n}$, and the mean-field vector, $\theta \in \mathbb{R}^n$, the probability distribution is given by

$$Pr[x] = \frac{1}{z} \exp \left( \sum_{i,j \in [n], i \neq j} A_{ij} x_i x_j + \sum_{i \in [n]} \theta_i x_i \right),$$

where $z = \sum_{x} \exp \left( \sum_{i,j \in [n], i \neq j} A_{ij} x_i x_j + \sum_{i \in [n]} \theta_i x_i \right)$.

Here, $A_{ij}$ denotes the weight of the edge connecting nodes $i$ and $j$ in $G$, and $\theta_i$ denotes the self-loop for node $i$ in $G$.

In addition to the joint distribution, the conditional distribution of each random variable, $x_i$, for $i \in [n]$ is given by

$$Pr[x_i = -1 | x_{-i}] = \frac{1}{1 + \exp(2 \sum_{j \in [n] \setminus \{i\}} A_{ij} x_j + \theta_i)}.$$  \hspace{1cm} (2.1)

Ising Models for Images

The motivation behind using Ising models for images comes from Eq. 2.1. If we model each pixel in an image as a random variable, representing the underlying distribution of images as an Ising model is equivalent to learning a single layered neural network for
each pixel value, with a sigmoid activation function, and the other pixel values being the input into the network.

In computer vision, traditionally, convolutional filters are used, which are linear functions of the neighboring pixels. However, adding a layer of non-linearity to the model, like what the Ising model does, makes the model more powerful.

## 2.5 Sparsitron

Assuming that the given data comes from an underlying graphical model is easy. The difficult part is to learn the graphical model, which can then be used for reasoning, interpretation and prediction. Learning a graphical model involves being able to learn the probability distribution of the variables, either as a joint distribution, or a conditional distribution. There are different ways of learning graphical models. The most common class of methods used are Markov chain Monte Carlo methods like Metropolis-Hastings due to Metropolis et al. (1953) and Hastings (1970) and Gibbs sampling due to Geman and Geman (1984). Existing work on learning Ising models mostly include learning the structure of the graphical model, and not the weights of the graph (Bresler, 2015; Loh and Wainwright, 2012). And the methods that exist to learn the weights of the graph require many independent samples from the graph of the model (Ravikumar, Wainwright, and Lafferty, 2010).

In this thesis, we will be using the Sparsitron algorithm, due to Klivans and Meka (2017), to learn the Ising models. This algorithm makes use of the Hedge algorithm due to Freund and Schapire (1997), and updates weights of attributes or features iteratively. Intuitively, a weight is multiplied by a factor greater than 1 if its corresponding feature correctly predicts the desired value, or less than 1, if that feature does not correctly predict the desired value. Sparsitron is easy to implement, and uses nearly optimal number of samples to learn Ising models.

### 2.5.1 Learning Generalized Linear Models

This section describes how the Sparsitron algorithm is used to learn sparse generalized linear models.

**Theorem 2.5.1 (Theorem 3.1 in Klivans and Meka (2017))**

Let $\mathcal{D}$ be a distribution on $[-1, 1]^n \times \{0, 1\}$ where for $(X, Y) \sim \mathcal{D}$, $\mathbb{E}[Y|X = x] = u(w \cdot x)$ for a non-decreasing 1-Lipschitz function $u : \mathbb{R} \rightarrow [0, 1]$. Suppose that $\|w\|_1 \leq \lambda$ for a known $\lambda \geq 0$. Then, there exists an algorithm that for all $\epsilon, \delta \in [0, 1]$ given $T = O(\lambda^2 (\ln(n / \delta \epsilon)) / \epsilon^2)$
Algorithm 1 Sparsitron

1: Initialize $w^0 = \frac{1}{n}$.
2: for $t=1, \ldots, T$ do
3:  Let $p^t = \frac{w^{t-1}}{||w^{t-1}||_1}$.
4:  Define $l^t \in \mathbb{R}^n$ by $l^t = \frac{1}{2}(1 + (\lambda p^t \cdot x^t - y^t)x^t)$.
5:  Update the weight vector $w^t$: for each $i \in [n]$, set $w^t_i = w^{t-1}_i \cdot \beta l^t_i$.
6: end for
7: Compute the empirical risk $\hat{\epsilon}(\lambda p^t) = \frac{1}{M} \sum_{j=1}^{M} (u(\lambda p^t \cdot a^j) - b^j)^2$.
8: return $\nu = \lambda p^j$ for $j = \arg \min_{t \in [T]} \hat{\epsilon}(\lambda p^t)$.

independent examples from $D$, produces a vector $\nu \in \mathbb{R}^n$ such that with probability at least $1 - \delta$,

$$
\mathbb{E}_{(X,Y) \sim D}[(u(\nu \cdot X) - u(w \cdot X))^2] \leq \epsilon.
$$

The run-time of the algorithm is $O(nT)$. Moreover, the algorithm can be run in an online manner.

For this algorithm to work and the theorem to hold, the following needs to be done

- $\forall i \in [n], w_i \geq 0$ and $||w||_1 = \lambda$. If not, we can map each data entry $(x, y)$ to $((x, -x, 0), y)$ and work in this new $(2n + 1)$ dimensional space. This would incorporate coefficients for both the features and their negations. The final value of the coefficient can be determined by subtracting the coefficient corresponding to the negation of the feature from the coefficient corresponding to the feature itself.

- $\beta = \frac{1}{1+\sqrt{\ln n}/T}$.

- $M = O\left(\frac{\ln(T/\delta)}{\epsilon^2}\right)$.

The algorithm updates the weight vector iteratively. It starts off weighting each feature equally. But as the algorithm progresses, the coefficients of the features that are correlated with the value to be predicted increase, and the others decrease. The algorithm is described in Alg. 1.

2.5.2 Learning Ising Models

This section describes how the Sparsitron algorithm is used to learn Ising models.

Let $A \in \mathbb{R}^{n \times n}$ be a weight matrix and $\theta \in \mathbb{R}^n$ be a mean-field vector. The associated $n$ variable Ising model is a distribution $D(A, \theta)$ on $\{-1, +1\}^n$, with dependency graph $G$. We define $\lambda(A, \theta) = \max_{i \in [n]} (\sum_j |A_{i,j}| + |\theta_i|)$ to be the width of the model.
Theorem 2.5.2 (Theorem 5.2 in Klivans and Meka (2017))

Let $\mathcal{D}(A, \theta)$ be an $n$-variable Ising model with width $\lambda(A, \theta) \leq \lambda$. There exists an algorithm that given $\lambda$, for $\epsilon, \rho \in (0, 1)$, and $N = O(\lambda^2 \exp(\Theta(\lambda))/\epsilon^4 \cdot (\log(n/\rho \epsilon))$ independent samples $Z^1, Z^2, \ldots, Z^N \leftarrow \mathcal{D}(A, \theta)$ produces $\hat{A}$ such that with probability at least $1 - \rho$,

$$\|A - \hat{A}\|_\infty \leq \epsilon.$$

The run-time of the algorithm is $O(n^2 N)$. Moreover, the algorithm can be run in an online manner.

In order to use Sparsitron, it is important to observe that for $Z \leftarrow \mathcal{D}(A, \theta)$, for $i \in [n]$ and any $x \in \{-1, +1\}^{[n]\setminus\{i\}},$

$$\Pr[Z_i = -1|Z_{-i} = x] = \frac{1}{1 + \exp(2 \sum_{j \in [n]\setminus\{i\}} A_{ij} x_j + \theta_i)} = \sigma(w(i) \cdot x - \theta_i),$$

where $w(i) \in \mathbb{R}^{n-1}$, with $w(i)_j = -2A_{ij} \forall j \in [n]\setminus\{i\}$. This enables us to use the Sparsitron algorithm for learning generalized linear models (since $\sigma$ is a non-decreasing 1-Lipschitz function, proof in Appendix A).

In order to learn $w(i)$, for $i \in [n]$, using Sparsitron, let the data entry be, $x = (Z_1, \ldots, Z_{i-1}, 1, Z_{i+1}, \ldots, Z_n)$ and its corresponding label be $y = \frac{1 - Z_i}{2}$. Then,

$$\mathbb{E}[y|x] = \sigma(w(i) \cdot x),$$

where $w(i) \in \mathbb{R}^n$, with $w(i)_i = -2A_{ij} \forall j \in [n]\setminus\{i\}$, and $w(i)_i = -\theta_i$.

Since each random variable of the Ising model corresponds to a GLM, and each GLM is independent of each other, each one of them can be learnt in parallel to one another, using Sparsitron.

### 2.6 Mean-Field Variational Inference

Once the graphical model is learnt, it is natural to ask what the independent distribution of each random variable in the graphical model is. This is a difficult question to answer as the random variables are dependent on each other. But the independent distribution can be approximated using variational inference.

In the case of the Ising model, let us assume that we are given the weight matrix $A$ and mean-field vector $\theta$. Let $\mu \in [0, 1]^n$ be a random assignment. If we iteratively
update the vector using the following formula

$$\mu_i = \sigma \left( -2 \sum_{j \in [n] \setminus \{i\}} A_{i,j} \mu_j - \theta_i \right),$$

\(\forall i \in [n]\), then the fixed point that the values converge to are a good approximation for the independent probabilities of the pixels taking on the value \(-1\). This is the variational inference algorithm for Ising models (Jordan et al., 1999).

This is a useful algorithm as it approximates the independent probabilities of the pixels in the image, and can be used for other calculations. For example, the joint probability, \(Pr[x]\), can now be approximated by \(\prod_{i=1}^{n} Pr[x_i]\).
Chapter 3

Methodology

The motivation behind the thesis is to learn the underlying distribution for a set of images. We will assume that the underlying distribution of the images is an Ising model, i.e., each pixel in the image is a generalized linear model of the other pixel values, with a sigmoid non-linearity. We will try to learn the weights of this Ising model using Sparsitron. Given images of size $n = s \times s$, where $s \in \mathbb{R}$, we build an Ising model with $n$ nodes, where each pixel of the image corresponds to a node in the Ising model. In this chapter, we will assume that the set of training images is $X, X^2, \ldots, X^m$.

In order to use the Sparsitron algorithm, we also need to assume the width of the Ising model. Since the choice of the width is not obvious, we try different width values in our experiments.

3.1 Image Preprocessing

In order for the model to be an Ising model, the values that each pixel can take is $-1$ or $+1$. This requires some preprocessing of the images.

If the image is RGB, it first needs to be changed to a grayscale image. This can be done by taking a linear combination of the different color channels of the image, i.e., given that the red, green and blue channels of the image are given by the matrices $R$, $G$ and $B$, where $R, G, B \in \{0, 1, \ldots, 255\}^{s \times s}$, the grayscale image could be calculated by $\alpha R + \beta G + \gamma B$, where $\alpha, \beta, \gamma \in [0, 1]$, and $\alpha + \beta + \gamma = 1$. However, this conversion, in general, is not unique. Different weightings of the color channels represent different effects of shooting black-and-white film, with different colored photographic filters.

For most digital standard definition formats, the luma or brightness of an image is calculated by

$$0.30R + 0.59G + 0.11B,$$
and is treated as the grayscale image. To change the range of values for each pixel from \( \{0, 1, \ldots, 255\} \) to \([0, 1]\), we can simply divide each pixel value by 255, and use float values.

Now we have a grayscale image represented by a matrix \( X'' \), where each pixel or entry is in the range \([0, 1]\). Now we need to change these values to \(-1\) or \(+1\), so some discretization is required. There is no unique way of doing this either. One easy way of doing this is to threshold the values at 0.5, i.e.,

\[
X'_{i,j} = \begin{cases} 
0 & \text{if } X''_{i,j} < 0.5 \\
1 & \text{if } X''_{i,j} \geq 0.5 
\end{cases}
\]

\( \forall i, j \in [s] \), and \( X' \) is the discretized image matrix. In order to change this to an Ising model sample, the new image matrix is given by

\[
X = 2X' - 1.
\]

Fig. 3.1 shows the effect of this image preprocessing on a grayscale image and a colored image.
3.2 Random Noise

Theoretically, the Sparsitron algorithm requires a certain number of data samples in order to converge to the true Ising model distribution. However, the number of data samples in datasets is finite and might be fewer than what is required by the algorithm. In order to still be able to provide different samples, we can add random noise to the images, which is a form of data augmentation, used commonly in training deep neural networks.

One way of doing so is to add Gaussian noise to each pixel of the image, before discretizing the pixel values. This Gaussian noise should have a mean of 0 and a low variance, so that the expected new pixel value is the same as the original pixel value and so that it does not differ much from the original pixel value. We set the variance to be 0.1 in our experiments. This type of noise does not effect the image structure itself, and the underlying distribution of the images remains roughly the same. Fig. 3.2 shows the difference between the original image and the same image with random noise added to it. Only the figure outline is slightly distorted, but otherwise the image structure itself remains the same.

Theoretically, Sparsitron needs to be trained on an independent set of samples. However, practically, this cannot be achieved if the underlying probability distribution is unknown and is not given to us. For this thesis, we will feed Sparsitron the training set samples, with random noise added to them.

3.3 Negation of Images

Given an image from the dataset, $X$, that has gone through the preprocessing steps mentioned before, the negation of the image would be $-X$. The negation of the image would change all pixels that were previously $-1$ to $+1$, and vice versa. But it is important to see that the image structure itself would be preserved. If the original image contained
a shape outline, it would still contain the outline in the negation of the image. Fig. 3.3 shows the difference between an image and its negation, which is just a pixel value swap.

This is a valid form of data augmentation, and any model that we train should be able to see the similarity between the two images, and learn their structure.

### 3.4 Locality or Neighborhood

Given large images, it might be inefficient and time consuming to assume that all pixels in the image have some correlation, i.e., all pixels in the image have non-zero weight edges between their corresponding nodes in the Ising model dependency graph. To simplify, we assume that a given pixel, \( X_{i,j}, \forall i, j \in [s] \), is only connected to its neighboring pixels.

This assumption is a standard assumption that people in the computer vision field make. It is rational to think that a pixel’s value is only dependent on its neighboring pixel. A pixel at the top-right corner of the image would probably not have any correlation with the pixel at the bottom-left corner. Pixels heavily depend on their neighboring pixels only, which is also the reason why convolutional filters work well for computer vision tasks.

Even though we make this assumption, it is not clear how many neighbors a pixel should be assumed to be correlated with. Because of this, we vary this number while running experiments. Let us define a distance metric as follows

\[
\text{dist}(X_{i,j}, X_{k,l}) = \max\{|i-k|, |j-l|\},
\]

for \( i, j, k, l \in [s] \). Then if we can assume two pixels, \( X_{i,j} \) and \( X_{k,l} \), to be correlated if \( \text{dist}(X_{i,j}, X_{k,l}) \leq \alpha \), for some \( \alpha \in \mathbb{R} \) and \( \alpha \geq 0 \). So the set of neighbors of node \( X_{i,j} \) is the set \( \{X_{k,l} | \text{dist}(X_{i,j}, X_{k,l}) \leq \alpha \} \).
3.5. Using Sparsitron

This is equivalent to taking a \((2\alpha + 1) \times (2\alpha + 1)\) sized window centered at the pixel \(X_{i,j}\), and the set of pixels the window covers is the set of neighbors of \(X_{i,j}\). This is again similar to convolutional filters.

3.5 Using Sparsitron

Using a window size of \(2\alpha + 1\), where \(\alpha \geq 0\), to determine the neighbors of a given pixel, each node in the Ising model would be connected to \((2\alpha + 1)^2\) nodes (including itself). When we use Sparsitron to recover the weights of a given pixel, we need only use the values of its neighbors as training data, and set 0 as the weight for all other edges.

For each pixel \(X_{i,j}\), where \(i, j \in [s]\), the Sparsitron algorithm outputs a weight vector \(w^X_{i,j}\) of size \(2(2\alpha + 1)^2 + 1\). One of the weights corresponds to the feature 0. The other \(2(2\alpha + 1)^2\) weights correspond to the values of the neighboring pixels of \(X_{i,j}\), and their respective negations. We determine the true weight of the edge between the nodes corresponding to \(X_{i,j}\) and its neighbor \(X_{k,l}\) as follows

\[
\text{edge weight between nodes corresponding to } X_{i,j} \text{ and } X_{k,l} = w^X_{X_{i,j}} - w^\bar{X}_{X_{i,j}}^{X_{k,l}},
\]

where \(w^X_{X_{i,j}}\) and \(w^\bar{X}_{X_{i,j}}^{X_{k,l}}\) correspond to the weight learnt for the neighboring pixel \(X_{k,l}\) and its negation respectively (this is also explained in Section 2.5.1).

Additionally, the edge weight between any pair of nodes, and hence the relation between any pair of pixels \(X_{i,j}\) and \(X_{k,l}\), in the Ising model is learnt twice, once when the edge weights from \(X_{i,j}\) are learnt, and once when the edge weights from \(X_{k,l}\) are learnt. Theoretically, these two values should be exactly the same. We take the average of the two values to incorporate both the values found.

3.6 Scaling Interactions

Given a weight matrix \(A\) and a mean-field vector \(\theta\) that describes an Ising model, an interesting question to ask is what happens when these interactions are scaled by a certain factor. For \(\gamma \in \mathbb{R}\), where \(\gamma > 0\), we can think of the new Ising model to be defined by the weight matrix \(\gamma A\) and the mean-field vector \(\gamma \theta\). It would be interesting to find a relation between the new probability distribution and the original one.

Switching from the original probability distribution to the new one preserves the ranking of the samples with respect to the probability of them occurring. However, it is hard to determine how the probabilities themselves change. This result is shown in Appendix B.
This is an interesting result and could lead to faster sampling from a given Ising model distribution. This is because if the weight matrix is scaled up, the correlations are intensified (both positive and negative), which can help techniques like Gibbs sampling and variational inference to possibly converge faster.

### 3.7 Other Models

While we compare several different Ising models with different model widths, different neighborhood window sizes, and different interaction scalings, we also want to compare our approach to other methods and models. We would compare the Ising model against linear models, in order to compare the linear model with the generalized linear model.

#### 3.7.1 Averaging Model

Under this model, we assume that the probability of a pixel being $-1$ (or $+1$) is given by the fraction of times that the pixel has been equal to $-1$ (or $+1$) in the given training dataset, i.e.,

$$
Pr[X_{i,j} = -1] = \frac{|\{m'|m' \in [m], X_{i,j}^{m'} = -1\}|}{m},
$$

$$
Pr[X_{i,j} = +1] = \frac{|\{m'|m' \in [m], X_{i,j}^{m'} = +1\}|}{m},
$$

$\forall i,j \in [s]$. This is a very naive model, and does not incorporate any pixel-pixel interactions at all.

#### 3.7.2 Nearest Neighbors Model

Under this model, we assume that the probability of a pixel being $-1$ (or $+1$) is given by the fraction of its neighbors that are $-1$ (or $+1$), averaged over all the images in the training dataset, i.e.,

$$
Pr[X_{i,j} = -1] = \frac{\sum_{m'=1}^{m} |\{X_{k,l}^{m'} | X_{k,l}^{m'} \in N(X_{i,j}^{m'}), X_{i,j}^{m'} = -1\}|}{\sum_{m'=1}^{m} |N(X_{i,j}^{m'})|},
$$

$$
Pr[X_{i,j} = +1] = \frac{\sum_{m'=1}^{m} |\{X_{k,l}^{m'} | X_{k,l}^{m'} \in N(X_{i,j}^{m'}), X_{i,j}^{m'} = +1\}|}{\sum_{m'=1}^{m} |N(X_{i,j}^{m'})|},
$$

$\forall i,j \in [n]$ and $N(X_{i,j}^{m'}) = \{X_{k,l}^{m'} | k,l \in [s], 0 < \text{dist}(X_{i,j}^{m'}, X_{k,l}^{m'}) \leq \alpha\}$, for some $\alpha \in \mathbb{R}$, $\alpha > 0$, is the set of neighboring pixels of $X_{i,j}^{m'}$ in image $m'$. This model does take into account the
interactions between a pixel and its neighbors, but all neighbors are treated in the same way, with the same weight. This is analogous to a convolutional filter used in computer vision, and the \(k\)-nearest neighbors approach for prediction tasks.

Similar to what we do for the Ising models, we will take different window sizes to determine which pixels are neighbors of a given pixel.

### 3.7.3 Generative Adversarial Network

Generative adversarial network (GAN), due to Goodfellow et al. (2014), is a deep neural network architecture that comprises of two neural networks. The generator is the neural network that learns how to generate synthetic images, while the discriminator learns how to differentiate synthetic images from real images. The two models are trained simultaneously.

GANs have been used for synthetic image generation and they work well. However, since both the generator and discriminator are trained simultaneously, it takes many epochs and a lot of computational power to train a GAN for the generator to output good synthetic images. On the other hand, it is faster to use Sparsitron to learn Ising models for low model widths, as it requires nearly optimal number of samples to train on.

Additionally, it is not clear how to compare an Ising model to a GAN, as a forward pass through the generator would result in a synthetic image, but the distribution itself is hard to infer from this.

Because of these reasons, we will not be able to compare our approach with a GAN, but we do show that our method is fast and easy to implement, while it generates good results too.

### 3.8 Testing

In order to compare the different Ising models and other models as mentioned above, we need some quantitative measure. We would be looking at the mean likelihood of the validation dataset. Ideally, the model learnt should have a high mean likelihood for the validation dataset. We will also study the probability heat maps created by the Ising models, and the convergence of Sparsitron.

#### 3.8.1 Likelihood

Likelihood is the probability of a sample occurring given a probability distribution. In the case of an Ising model, given a sample \(x \in \{-1, +1\}^n\), the weight matrix \(A\), and the
mean-field vector \( \theta \), the likelihood is given by

\[
\text{Likelihood}_{A, \theta}(x) = Pr[x|A, \theta] = \frac{1}{z} \exp \left( \sum_{i,j \in [n], i \neq j} A_{i,j} x_i x_j + \sum_{i \in [n]} \theta_i x_i \right),
\]

where \( z = \sum_x \exp \left( \sum_{i,j \in [n], i \neq j} A_{i,j} x_i x_j + \sum_{i \in [n]} \theta_i x_i \right). \)

Variational inference on an Ising model results in an approximation of the independent probability distribution of each pixel, i.e., it will approximate the value of \( Pr[X_{i,j} = -1|A, \theta] \) (or \( Pr[X_{i,j} = +1|A, \theta] \)), \( \forall i,j \in [s] \). So if the validation dataset is given by the set of images \( \{X^1, X^2, \ldots, X^t\} \), the mean likelihood is given by

\[
\text{Mean Likelihood}_{A, \theta} = \frac{\sum_{t=1}^t Pr[X^t_{i,j}|A, \theta]}{t} = \frac{\sum_{i,j \in [s]} \prod_{t=1}^t Pr[X^t_{i,j}|A, \theta]}{t}.
\]

The higher the mean likelihood of the validation dataset, the better is the model.

### 3.8.2 Probability Heat Map

Probability heat maps are a good way of investigating what image structure is being learnt by the model. These are probability matrices that describe the independent probability of each pixel being \(-1\) (or \(+1\)). This is equivalent to the output of variational inference. So instead of drawing random samples from the distribution, which could be done using Gibbs sampling, the probability heat map can be investigated, which would be the same as the expected image from the distribution.

### 3.8.3 Convergence Time

While theoretically Sparsitron requires a certain number of samples to converge to the underlying Ising model, in practice the number of samples may be fewer or more than this theoretical bound.

In this case, convergence has to be determined for each row of the weight matrix of the Ising model. So when the GLM of each pixel is being learnt iteratively, we can calculate the mean likelihood of the neighborhood values given the weights at that iteration. This can be averaged over all pixels to give a mean likelihood at a particular iteration of Sparsitron. If this value does not change much after a couple of iterations, that means that the algorithm has converged. The mean likelihood in this case can be calculated on the training and validation datasets.
Chapter 4

Experiments

The datasets that we would be looking at in our experiments are the MNIST dataset (LeCun et al., 1998) and the CIFAR-10 dataset (Krizhevsky, 2009). The MNIST dataset contains grayscale images of size $28 \times 28$. It contains 60,000 training and 10,000 test images. The CIFAR-10 dataset contains colored (or RGB) images of size $32 \times 32$. It contains 50,000 training and 10,000 test images. In both cases, there are 10 possible labels for each image.

For these experiments, the Sparsitron algorithm for learning Ising models is used (Section 2.5.2). For all experiments, $\epsilon = 0.2 = \rho$, but the value of $\lambda$ was experimented with. The number of training samples used for the experiments was $T = \frac{\lambda^2 \log(n/\rho \epsilon)}{\epsilon^4}$ (the factor of $\exp(O(\lambda))$ was removed after consultation with the author of Klivans and Meka (2017)). The value of $\alpha$ was also varied in order to determine the number of neighbors of each pixel in the Ising model (Section 3.4). The interactions were also scaled by factors $\gamma = 0.5, 1, 2, 4$ (Section 3.6). The models were trained on two kinds of datasets, with or without the negations of the images. Similarly, testing was also done on two datasets, with or without the negations of the images. While models should be able to handle the negation of the images as well, different training and testing datasets give us a point of comparison for the models.

The Sparsitron algorithm for learning a GLM is an iterative multiplicative weight update algorithm, which cannot be parallelized. However, Sparsitron extended to learn an Ising model is equivalent to learning multiple GLMs for each given pixel in the image. This can be done in parallel as learning each one of these GLMs is independent of each other. We make use of this fact in running our experiments to get gains in the run-time of the algorithm.

Once the Ising model was learnt, the individual pixel probabilities were determined using variational inference. The initial probabilities were all set to 0.5, so an equal chance of being $-1$ or $+1$ (random initial values were tried too, and all runs resulted in the same fixed point). The variational inference algorithm was run till a fixed point was obtained, or for a maximum of $1000n$, where $n$ is the number of pixels in the image.
Figure 4.1: Similarity between images of the same digit label in the MNIST dataset. (A) and (B) represent digit 0, (C) and (D) represent digit 8.

4.1 MNIST

The MNIST (Modified National Institute of Standards and Technology) dataset contains images of handwritten digits. Since there is only one way of writing a digit, all images for a given digit are very similar to each other. They might be slightly scaled (shrunk or enlarged), rotated, etc., but they all look very similar to each other. This similarity among the images of the same digit is shown in Fig. 4.1. Because of this, we would expect the averaging models and the nearest neighbors models to perform well.

Tables 4.1 and 4.2 summarize the results of the mean likelihood calculations on the different set of models used, with and without the negation of images, respectively. The table only reports the mean likelihoods of the top 3 models for each given label.

The mean likelihoods calculated are extremely low. We would expect them to be between $\frac{1}{2^{28\times28}} \approx e^{-238}$ (since there are $2^{28\times28}$ possible permutations of assigning values to each pixel in the image), and $\frac{1}{70000} \approx e^{-5}$ (since there are 70,000 images in the dataset). However, these mean likelihoods are very low. But out of the models tested, the Ising models did well consistently on all labels.

An interesting observation is that the top 3 models for the validation dataset with the negation of images are exactly the same as the models without the negation of images, and their mean likelihoods drop by a factor of half on the validation dataset that contains the negations.
Another interesting result is that models where the interactions were scaled by factors of 2 and 4 do better than the original models.

### 4.2 CIFAR-10

The CIFAR-10 (Canadian Institute For Advanced Research) dataset contains images of 10 different class types - airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Images in each class, in general, can be quite different from each other, so this dataset is harder to learn as compared to the MNIST dataset.

Tables 4.3 and 4.4 summarize the results of the mean likelihood calculations on the different set of models used, with and without the negation of images, respectively. Here too, the table only reports the mean likelihoods of the top 3 models for each given label.

In this case, the mean likelihoods calculated are low in certain cases. We would expect them to be between \(\frac{1}{2^{32 \times 32}} \approx e^{-310}\) (since there are \(2^{32 \times 32}\) possible permutations of assigning values to each pixel in the image), and \(\frac{1}{60000} \approx e^{-5}\) (since there are 60,000 images in the dataset). Some mean likelihoods calculated for certain classes are greater than \(e^{-5}\). Images of automobiles seem to be harder to learn than images of ships. Fig. 4.2 illustrates the difference in the two label images. The ships dataset does look very similar to each other. Blue sea, similar orientation of the ships, etc. Whereas, the automobiles dataset is more varying with different background, different car angles, different colored cars, etc.

Here again, we see that models where the interactions were scaled by factors of 2 and 4 do better than the original models.

### 4.3 Probability Heat Maps

Fig. 4.3 illustrates the probability heat maps of different models trying to learn the structure of the digit 0, over different \(\alpha\) values, without the negation of images included during training. All models are able to learn a good structure of the image, including the averaging models and the nearest neighbors models. Fig. 4.4 illustrates the same thing, but with the negation of images included during training. In this case, only the Ising models are able to incorporate some of the structure. The other linear models fail to learn anything in this case.

Fig. 4.5 illustrates the probability heat maps of different Ising models trying to learn the structure of the digit 0, over different \(\gamma\) values, without the negation of images included during training. The only scalings of the interaction does affect the probability distribution, but the structure of the images remains almost the same.
Fig. 4.6 illustrates the probability heat maps of some Ising models learnt for the different digits on the MNIST dataset.

### 4.4 Sparsitron Convergence Time

Fig. 4.7 illustrates the converge of Sparsitron over the theoretically prescribed number of iterations. This is done over both the MNIST and CIFAR-10 datasets, and for different image labels. Sparsitron does converge much before what the theoretical bound on the number of samples for its convergence is.
### 4.4. Sparsitron Convergence Time

#### Table 4.1: Top 3 models for each label, tested on the MNIST validation dataset with the negation of images.

<table>
<thead>
<tr>
<th>Label</th>
<th>Model</th>
<th>Mean Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sparsitron ($\lambda = 1, \alpha = 2, \gamma = 4$)</td>
<td>$6.81e - 61$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$1.61e - 63$</td>
</tr>
<tr>
<td></td>
<td>Nearest Neighbor Model ($\alpha = 1$)</td>
<td>$6.40e - 71$</td>
</tr>
<tr>
<td>1</td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$)</td>
<td>$2.92e - 28$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$5.63e - 29$</td>
</tr>
<tr>
<td></td>
<td>Nearest Neighbor Model ($\alpha = 1$)</td>
<td>$8.06e - 33$</td>
</tr>
<tr>
<td>2</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$5.67e - 64$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$2.92e - 28$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$2.03e - 66$</td>
</tr>
<tr>
<td>3</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$4.72e - 54$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$3.22e - 54$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$6.91e - 58$</td>
</tr>
<tr>
<td>4</td>
<td>Averaging Model</td>
<td>$5.40e - 53$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$1.51e - 53$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$1.22e - 53$</td>
</tr>
<tr>
<td>5</td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$1.16e - 51$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$2.86e - 52$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$)</td>
<td>$8.24e - 58$</td>
</tr>
<tr>
<td>6</td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$1.25e - 46$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$8.62e - 47$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$)</td>
<td>$3.94e - 51$</td>
</tr>
<tr>
<td>7</td>
<td>Averaging Model</td>
<td>$4.52e - 46$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$9.54e - 50$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$8.90e - 50$</td>
</tr>
<tr>
<td>8</td>
<td>Sparsitron ($\lambda = 1, \alpha = 4, \gamma = 4$)</td>
<td>$8.28e - 51$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 2$)</td>
<td>$3.58e - 54$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$)</td>
<td>$2.08e - 58$</td>
</tr>
<tr>
<td>9</td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$4.30e - 50$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$3.32e - 50$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$1.26e - 51$</td>
</tr>
</tbody>
</table>
Table 4.2: Top 3 models for each label, tested on the MNIST validation dataset without the negation of images.

<table>
<thead>
<tr>
<th>Label</th>
<th>Model</th>
<th>Mean Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sparsitron ($\lambda = 1, \alpha = 2, \gamma = 4$)</td>
<td>$1.36e - 60$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$3.21e - 63$</td>
</tr>
<tr>
<td></td>
<td>Nearest Neighbor Model ($\alpha = 1$)</td>
<td>$1.28e - 70$</td>
</tr>
<tr>
<td>1</td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$)</td>
<td>$5.83e - 28$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$1.13e - 28$</td>
</tr>
<tr>
<td></td>
<td>Nearest Neighbor Model ($\alpha = 1$)</td>
<td>$1.61e - 32$</td>
</tr>
<tr>
<td>2</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$1.13e - 63$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$2.25e - 64$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$4.05e - 66$</td>
</tr>
<tr>
<td>3</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$9.44e - 54$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$6.44e - 54$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$1.38e - 57$</td>
</tr>
<tr>
<td>4</td>
<td>Averaging Model</td>
<td>$1.08e - 52$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$3.02e - 53$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$2.44e - 53$</td>
</tr>
<tr>
<td>5</td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$2.32e - 51$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$5.72e - 52$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$)</td>
<td>$1.65e - 57$</td>
</tr>
<tr>
<td>6</td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$2.49e - 46$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$1.72e - 46$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$)</td>
<td>$7.88e - 51$</td>
</tr>
<tr>
<td>7</td>
<td>Averaging Model</td>
<td>$9.03e - 46$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$1.91e - 49$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$1.78e - 49$</td>
</tr>
<tr>
<td>8</td>
<td>Sparsitron ($\lambda = 1, \alpha = 4, \gamma = 4$)</td>
<td>$1.66e - 50$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 2$)</td>
<td>$7.16e - 54$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$)</td>
<td>$4.15e - 58$</td>
</tr>
<tr>
<td>9</td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$)</td>
<td>$8.59e - 50$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 2$)</td>
<td>$6.64e - 50$</td>
</tr>
<tr>
<td></td>
<td>Averaging Model</td>
<td>$2.53e - 51$</td>
</tr>
</tbody>
</table>
4.4. Sparsitron Convergence Time

TABLE 4.3: Top 3 models for each label, tested on the CIFAR-10 validation dataset with the negation of images.

<table>
<thead>
<tr>
<th>Label</th>
<th>Model</th>
<th>Mean Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$) with negation</td>
<td>$1.77e^{-3}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$)</td>
<td>$1.77e^{-3}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$)</td>
<td>$1.77e^{-3}$</td>
</tr>
<tr>
<td>automobile</td>
<td>Sparsitron ($\lambda = 2, \alpha = 1, \gamma = 4$)</td>
<td>$4.12e^{-11}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 2, \gamma = 4$)</td>
<td>$4.03e^{-11}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$)</td>
<td>$4.00e^{-11}$</td>
</tr>
<tr>
<td>bird</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$) with negation</td>
<td>$1.42e^{-3}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 1, \gamma = 4$)</td>
<td>$1.42e^{-3}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 2, \gamma = 4$)</td>
<td>$1.42e^{-3}$</td>
</tr>
<tr>
<td>cat</td>
<td>Sparsitron ($\lambda = 2, \alpha = 1, \gamma = 4$)</td>
<td>$4.00e^{-11}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 1, \gamma = 4$) with negation</td>
<td>$3.99e^{-11}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 2, \gamma = 4$)</td>
<td>$3.99e^{-11}$</td>
</tr>
<tr>
<td>deer</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$) with negation</td>
<td>$1.77e^{-3}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$)</td>
<td>$1.77e^{-3}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$)</td>
<td>$1.77e^{-3}$</td>
</tr>
<tr>
<td>dog</td>
<td>Sparsitron ($\lambda = 2, \alpha = 1, \gamma = 4$)</td>
<td>$4.50e^{-18}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 1, \gamma = 4$) with negation</td>
<td>$4.50e^{-18}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 2, \gamma = 4$)</td>
<td>$4.49e^{-18}$</td>
</tr>
<tr>
<td>frog</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$) with negation</td>
<td>$3.55e^{-4}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$)</td>
<td>$3.55e^{-4}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$)</td>
<td>$3.55e^{-4}$</td>
</tr>
<tr>
<td>horse</td>
<td>Sparsitron ($\lambda = 1, \alpha = 8, \gamma = 4$) with negation</td>
<td>$8.74e^{-21}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$) with negation</td>
<td>$8.74e^{-21}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 4, \gamma = 4$) with negation</td>
<td>$8.74e^{-21}$</td>
</tr>
<tr>
<td>ship</td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$) with negation</td>
<td>$3.55e^{-4}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 4, \gamma = 4$) with negation</td>
<td>$3.55e^{-4}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 8, \gamma = 4$)</td>
<td>$3.55e^{-4}$</td>
</tr>
<tr>
<td>truck</td>
<td>Sparsitron ($\lambda = 1, \alpha = 1, \gamma = 4$)</td>
<td>$2.12e^{-29}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 2, \alpha = 1, \gamma = 4$)</td>
<td>$1.96e^{-29}$</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ($\lambda = 1, \alpha = 2, \gamma = 4$)</td>
<td>$1.90e^{-29}$</td>
</tr>
</tbody>
</table>
**Table 4.4**: Top 3 models for each label, tested on the CIFAR-10 validation dataset without the negation of images.

<table>
<thead>
<tr>
<th>Label</th>
<th>Model</th>
<th>Mean Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane</td>
<td>Sparsitron ((\lambda = 2, \alpha = 1, \gamma = 4)) with negation</td>
<td>(7.99 \times 10^{-11})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 1, \gamma = 4))</td>
<td>(7.99 \times 10^{-11})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 1, \gamma = 4)) with negation</td>
<td>(7.98 \times 10^{-11})</td>
</tr>
<tr>
<td>automobile</td>
<td>Sparsitron ((\lambda = 0.5, \alpha = 1, \gamma = 4))</td>
<td>(2.37 \times 10^{-142})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 0.5, \alpha = 2, \gamma = 4))</td>
<td>(5.23 \times 10^{-143})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 0.5, \alpha = 4, \gamma = 4))</td>
<td>(1.14 \times 10^{-143})</td>
</tr>
<tr>
<td>bird</td>
<td>Sparsitron ((\lambda = 1, \alpha = 1, \gamma = 4))</td>
<td>(1.76 \times 10^{-53})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 1, \alpha = 2, \gamma = 4))</td>
<td>(1.73 \times 10^{-53})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 1, \alpha = 4, \gamma = 4))</td>
<td>(1.21 \times 10^{-32})</td>
</tr>
<tr>
<td>cat</td>
<td>Sparsitron ((\lambda = 1, \alpha = 8, \gamma = 4)) with negation</td>
<td>(3.50 \times 10^{-20})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 8, \gamma = 2)) with negation</td>
<td>(3.50 \times 10^{-20})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 8, \gamma = 2))</td>
<td>(3.50 \times 10^{-20})</td>
</tr>
<tr>
<td>deer</td>
<td>Sparsitron ((\lambda = 1, \alpha = 1, \gamma = 4))</td>
<td>(1.21 \times 10^{-32})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 1, \alpha = 2, \gamma = 4))</td>
<td>(1.21 \times 10^{-32})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 1, \alpha = 2, \gamma = 4))</td>
<td>(1.21 \times 10^{-32})</td>
</tr>
<tr>
<td>dog</td>
<td>Sparsitron ((\lambda = 2, \alpha = 1, \gamma = 4))</td>
<td>(9.00 \times 10^{-18})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 1, \gamma = 4)) with negation</td>
<td>(8.99 \times 10^{-18})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 2, \gamma = 4))</td>
<td>(8.98 \times 10^{-18})</td>
</tr>
<tr>
<td>frog</td>
<td>Sparsitron ((\lambda = 2, \alpha = 1, \gamma = 4))</td>
<td>(2.39 \times 10^{-07})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 2, \gamma = 4))</td>
<td>(2.38 \times 10^{-07})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 1, \gamma = 4)) with negation</td>
<td>(2.38 \times 10^{-07})</td>
</tr>
<tr>
<td>horse</td>
<td>Sparsitron ((\lambda = 1, \alpha = 1, \gamma = 4))</td>
<td>(1.73 \times 10^{-86})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 1, \alpha = 1, \gamma = 2))</td>
<td>(1.72 \times 10^{-86})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 1, \alpha = 1, \gamma = 4)) with negation</td>
<td>(1.70 \times 10^{-86})</td>
</tr>
<tr>
<td>ship</td>
<td>Sparsitron ((\lambda = 2, \alpha = 8, \gamma = 4)) with negation</td>
<td>(7.10 \times 10^{-04})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 4, \gamma = 4)) with negation</td>
<td>(7.10 \times 10^{-04})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 2, \alpha = 8, \gamma = 4))</td>
<td>(7.10 \times 10^{-04})</td>
</tr>
<tr>
<td>truck</td>
<td>Sparsitron ((\lambda = 0.5, \alpha = 1, \gamma = 4))</td>
<td>(3.74 \times 10^{-115})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 0.5, \alpha = 2, \gamma = 4))</td>
<td>(3.34 \times 10^{-115})</td>
</tr>
<tr>
<td></td>
<td>Sparsitron ((\lambda = 0.5, \alpha = 4, \gamma = 4))</td>
<td>(1.58 \times 10^{-115})</td>
</tr>
</tbody>
</table>
4.4. Sparsitron Convergence Time

Figure 4.2: Difference in the different label classes in the CIFAR-10 dataset. (A)-(D) are images of ships. (E)-(H) are images of automobiles.
FIGURE 4.3: Probability heat maps for the different models learnt on the digit 0, without the negation of images included during training. From left to right, the columns denote $\alpha = 1, 2, 4, 8$. (A)-(L) are Ising models, where the rows, from top to bottom denote $\lambda = 0.5, 1, 2$. (M)-(P) are nearest neighbors models. (Q) is the averaging model.
4.4. Sparsitron Convergence Time

Figure 4.4: Probability heat maps for the different models learnt on the digit 0, with the negation of images included during training. From left to right, the columns denote $\alpha = 1, 2, 4, 8$. (A)-(L) are Ising models, where the rows, from top to bottom denote $\lambda = 0.5, 1, 2$. (M)-(P) are nearest neighbors models. (Q) is the averaging model.
Figure 4.5: Probability heat maps for the different Ising models learnt on the digit 0, without the negation of images included during training. From left to right, the columns denote $\gamma = 0.5, 1, 2, 4$. From top to bottom denote $\lambda = 0.5, 1, 2$. 
4.4. Sparsitron Convergence Time

Figure 4.6: Probability heat maps for the different Ising models learnt on the MNIST dataset.
Figure 4.7: Convergence plots for Sparsitrón ($\lambda = 1$, $\alpha = 1$, $\gamma = 1$) run on different labels and different datasets. (A) and (B) correspond to the mean likelihood calculated on the training dataset for MNIST and CIFAR-10 respectively. (C) and (D) correspond to the mean likelihood calculated on the validation dataset for MNIST and CIFAR-10 respectively.
Chapter 5

Conclusion

5.1 Ising Model Assumption

From the results, it is clear that our assumption of modeling the underlying probability distributions as an Ising model is a valid one. In almost all cases, the Ising model achieves higher mean likelihoods on the validation set, compared to the average model and the nearest neighbors model. This improvement in performance comes from the fact that in the Ising model models each pixel in the image is a generalized linear model. It is actually a single layered neural network, with sigmoid activation and input being the other pixel values. It takes Sparsitron an optimal number of samples to learn this distribution, which can also be parallelized to achieve a good run-time.

The Ising model can be made more complex by extending it to the general alphabet, where each pixel can take on more than just 2 discrete values. Sparsitron can be extended to learn such models, which can then be used to learn the distribution of images, which would be especially useful for colored (or RGB) images.

5.2 Scaling of Interactions

Initially, our motivation for scaling up interactions of the Ising model was to help techniques like Gibbs sampling and variational inference converge faster. The results from the experiments have indicated that scaling the interactions, at least in this case, also resulted in a better model than the original one. One reason why this happened could be because even though we assume that the underlying probability distribution is modeled by an Ising model, the width of the model is unknown. So while we do experiment with different widths, scaling the interactions is also a way of doing so.

The result in Appendix B justifies why we might scale the values, though it is not clear how the probabilities of a sample in the scaled and unscaled models are related. This would be something worth investigating further, and would help in learning Ising models with unknown model widths.
5.3 Low Mean Likelihoods

The mean likelihoods calculated on the various models were low, especially on the MNIST dataset. It is possible that more similar images could be generated without effecting the structure of the image, which was not part of the validation dataset.

For the CIFAR-10 dataset, we saw that the mean likelihood values varied a lot among the different label types. This could be because some labels are just harder to learn because of the variety of colors used in the images, the different angles from which the images are taken, etc.

5.4 Convergence of Sparsitron

Even though Sparsitron has a theoretical bound as to how many samples it needs to converge to the true distribution, empirically we see that the algorithm in fact converges much before that theoretical bound. This means that larger Ising models could potentially be learnt quickly using Sparsitron. This also opens up the question of updating the weights in a simultaneous fashion, rather than iterative fashion, which would further decrease the time taken by the algorithm. This requires further investigation.

5.5 Negation of Images

While using negation of images is a reasonable way of data augmentation, in the case of the MNIST dataset, this augmentation did not improve the Ising models. But on the CIFAR-10 dataset, the models trained on the augmented dataset performed better. While the structure of the image is preserved, why such data augmentation worked for one dataset and not the other requires further investigation.

5.6 Future Work

In this thesis, in order to discretize the pixel values, we used thresholding. However, other methods could be utilized in order to accomplish that task. One method that would preserve the structure of the image would be image segmentation, which is a computer vision task of partitioning the image into meaningful pieces. This could help detect the foreground and background of an image. Once that is achieved, one value could be assigned to the pixels belonging to the foreground, and another value could be assigned to the pixels belonging to the background.

Once the underlying probability distribution is learnt, synthetic images could be generated from it by sampling from the distribution. This could be used as a data
augmentation technique to append existing datasets. This would help learning models learn from more data.

Additionally, Sparsitron could potentially be used in place of the discriminator in a generative adversarial network (GAN) architecture (Goodfellow et al., 2014). Currently, the generator and discriminator are trained simultaneously which is hard to converge. But instead we could train the generator to produce images that belong to a similar Ising model distribution as the original dataset. This would be a new way of training GANs, and requires further investigation.
Bibliography


Pearl, Judea (2011). “Bayesian networks”. In:
Appendix A

Sigmoid Function

The sigmoid function, \( \sigma : \mathbb{R} \to (0, 1) \), is defined \( \forall x \in \mathbb{R} \) as follows

\[
\sigma(x) = \frac{1}{1 + \exp(-x)}.
\]

We will prove some properties about the sigmoid function.

A.1 Non-Decreasing

A function \( f : \mathbb{R} \to \mathbb{R} \) is said to be non-decreasing if \( \forall x, y \in \mathbb{R} \), where \( x < y \), \( f(x) \leq f(y) \).

For \( x, y \in \mathbb{R} \), where \( x < y \),

\[
x < y \Rightarrow -x > -y \Rightarrow \exp(-x) > \exp(-y) \Rightarrow \frac{1}{1 + \exp(-x)} < \frac{1}{1 + \exp(-y)} \Rightarrow \sigma(x) < \sigma(y).
\]

Therefore the sigmoid function is a non-decreasing function.

A.2 1-Lipschitz

A function \( f : \mathbb{R} \to \mathbb{R} \) is said to be \( k \)-Lipschitz, for \( k \in \mathbb{R} \), if \( \forall x, y \) in the domain of \( f \),

\[
|f(x) - f(y)| \leq k \cdot |x - y|.
\]

We will also need the definition of the mean value theorem. For \( a, b \in \mathbb{R} \), and a given function \( f : \mathbb{R} \to \mathbb{R} \) that is continuous on \([a, b]\) and differentiable on \((a, b)\), there is at
least one point \( c \in (a, b) \) such that
\[
f'(c) = \frac{f(b) - f(a)}{b - a},
\]
where \( f' \) is the differential of \( f \).

The sigmoid function is continuous and differentiable on \( \mathbb{R} \). Let \( \sigma' \) be the derivative of the sigmoid function. \( \forall x \in \mathbb{R} \) it is defined as
\[
\sigma'(x) = \frac{\exp(-x)}{(1 + \exp(-x))^2} = \frac{1 + \exp(-x)}{(1 + \exp(-x))^2} - \frac{1}{(1 + \exp(-x))^2} = \sigma(x)(1 - \sigma(x)).
\]

\( \forall x, y \in \mathbb{R}, \) where \( x < y \), using the mean value theorem and the result that the sigmoid function is non-decreasing, for some \( z \in (x, y) \),
\[
\sigma'(z) = \frac{\sigma(y) - \sigma(x)}{y - x} = \frac{\left| \sigma(y) - \sigma(x) \right|}{|y - x|}. \tag{A.1}
\]

Since the range of \( \sigma \) is \((0, 1)\),
\[
\max_{u \in \mathbb{R}} \sigma'(u) = \max_{u \in \mathbb{R}} \sigma(u)(1 - \sigma(u)) \leq 1. \tag{A.2}
\]

From Eqs. A.1 and A.2,
\[
\frac{\left| \sigma(y) - \sigma(x) \right|}{|y - x|} = \sigma'(z) \leq 1
\]
\[
\implies \left| \sigma(y) - \sigma(x) \right| \leq 1 \cdot |y - x|.
\]

Therefore \( \sigma \) is a 1-Lipschitz function.
Appendix B

Scaling Interactions

Let $A \in \mathbb{R}^{n \times n}$ be a weight matrix and $\theta \in \mathbb{R}^{n}$ be a mean-field vector for a given Ising model. For $\gamma \in \mathbb{R}$, where $\gamma > 0$, $\gamma A$ and $\gamma \theta$ are also valid pair of weight matrix and mean-field vector to define a new Ising model.

For $x, y \in \{-1, +1\}^{n}$, where $Pr_{A,\theta}[x] > Pr_{A,\theta}[y],$

\[
Pr_{A,\theta}[x] > Pr_{A,\theta}[y] \\
\implies \frac{1}{Z_{A,\theta}} \exp \left( \sum_{i,j \in [n], i \neq j} A_{i,j}x_i x_j + \sum_{i \in [n]} \theta_i x_i \right) > \frac{1}{Z_{A,\theta}} \exp \left( \sum_{i,j \in [n], i \neq j} A_{i,j}y_i y_j + \sum_{i \in [n]} \theta_i y_i \right) \\
\implies \exp \left( \sum_{i,j \in [n], i \neq j} A_{i,j}x_i x_j + \sum_{i \in [n]} \theta_i x_i \right) > \exp \left( \sum_{i,j \in [n], i \neq j} A_{i,j}y_i y_j + \sum_{i \in [n]} \theta_i y_i \right) \\
\implies \exp \left[ \gamma \left( \sum_{i,j \in [n], i \neq j} A_{i,j}x_i x_j + \sum_{i \in [n]} \theta_i x_i \right) \right] > \exp \left[ \gamma \left( \sum_{i,j \in [n], i \neq j} A_{i,j}y_i y_j + \sum_{i \in [n]} \theta_i y_i \right) \right] \\
\implies \exp \left( \sum_{i,j \in [n], i \neq j} \gamma A_{i,j}x_i x_j + \sum_{i \in [n]} \gamma \theta_i x_i \right) > \exp \left( \sum_{i,j \in [n], i \neq j} \gamma A_{i,j}y_i y_j + \sum_{i \in [n]} \gamma \theta_i y_i \right) \\
\implies \frac{1}{Z_{\gamma A,\gamma \theta}} \exp \left( \sum_{i,j \in [n], i \neq j} \gamma A_{i,j}x_i x_j + \sum_{i \in [n]} \gamma \theta_i x_i \right) > \frac{1}{Z_{\gamma A,\gamma \theta}} \exp \left( \sum_{i,j \in [n], i \neq j} \gamma A_{i,j}y_i y_j + \sum_{i \in [n]} \gamma \theta_i y_i \right) \\
\implies Pr_{\gamma A,\gamma \theta}[x] > Pr_{\gamma A,\gamma \theta}[y],
\]

where $Z_{A,\theta} = \sum_{x} \exp \left( \sum_{i,j \in [n], i \neq j} A_{i,j}x_i x_j + \sum_{i \in [n]} \theta_i x_i \right)$ is the normalizing constant for the original distribution, and $Z_{\gamma A,\gamma \theta} = \sum_{x} \exp \left( \sum_{i,j \in [n], i \neq j} \gamma A_{i,j}x_i x_j + \sum_{i \in [n]} \gamma \theta_i x_i \right)$ is the normalizing constant for the new distribution.

This means that scaling the weight matrix and the mean-field vector with the same positive factor preserves the order of the samples with respect to their respective probabilities. So if $x$ is more likely to occur than sample $y$ under the original probability
distribution, it will still be more likely to occur under the new probability distribution as well.

The relation between $Pr_{A,θ}[x]$ and $Pr_{γA,γθ}[x]$ is hard to interpret as the normalizing constants are different in the two cases ($z_{A,θ}$ and $z_{γA,γθ}$ respectively).