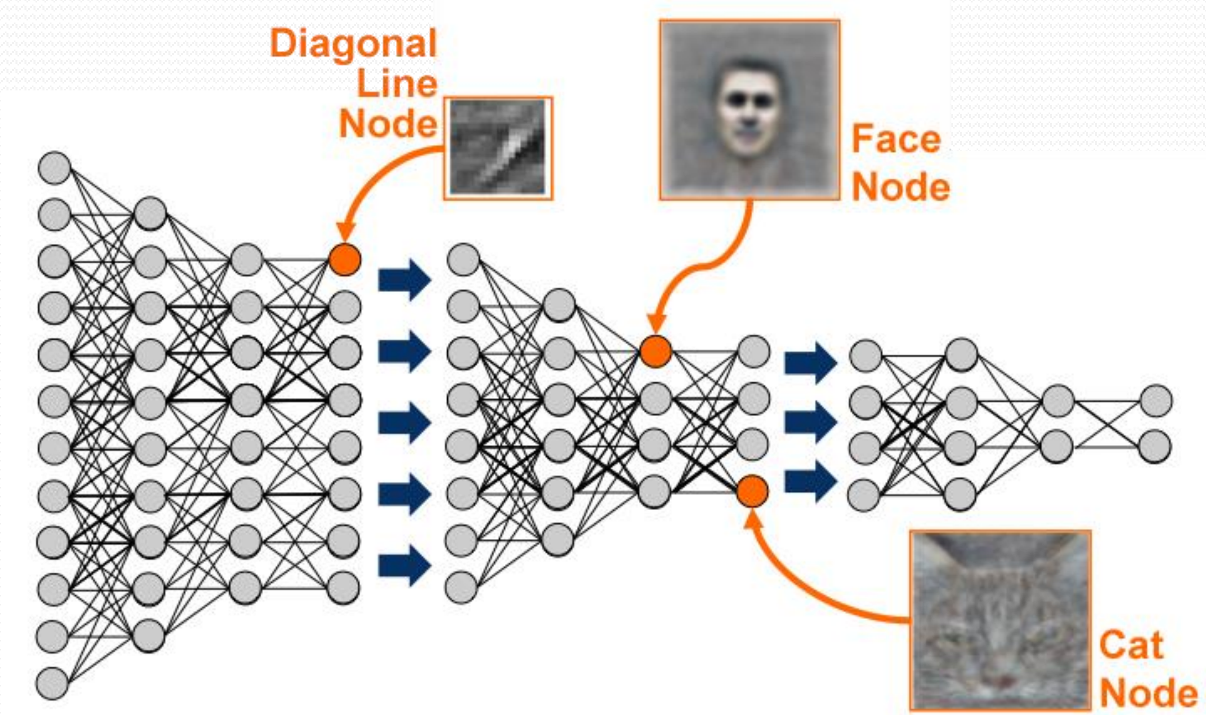
From Deep Learning to Minimum Probability Flow

Shaowei Lin (SUTD)
11 Nov 2016
Brain Lab

Cat Videos

- 2012 Experiment by Google, Stanford (Andrew Ng)
- 3 days, 1000 machines, 16,000 cores, 9-layered neural network, 1 billion connections, 10 million YouTube thumbnails



Le, Quoc V. "Building high-level features using large scale unsupervised learning." IEEE ICASSP, 2013.
MIT Tech Review, "10 Breakthrough Technologies 2013." http://www.technologyreview.com/featuredstory/513696/deep-learning/

Speech Translation

- EN Speech => EN Text => CH Text => CH Speech

Mastering Computer Games

Space Invaders

Google DeepMind

General Artificial Intelligence

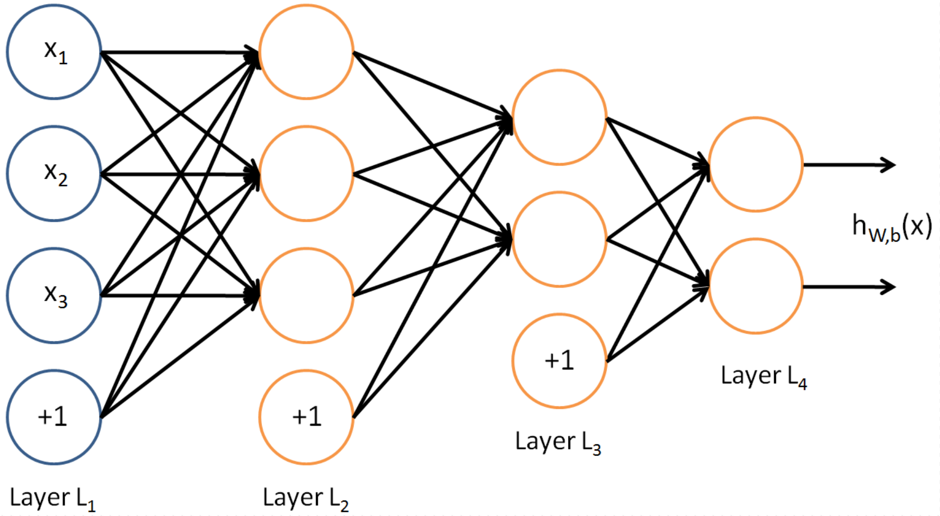
Outline

1. Deep Learning
2. Regularization
3. Probability Flow

Deep Learning

What is Deep Learning?

- Biologically-inspired multilayer neural networks



- Unsupervised learning (data without labels)

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

What is Deep Learning?

Example. Face recognition (Facebook)

Edges — Eyes, Noses, Mouths — Faces

- Deeper layers learn higher-order features

LISA lab, “Deep Learning Tutorials.” (http://www.deeplearning.net/tutorial/) accessed 2014.

Lee, Honglak, et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009.

Energy-based Learning

- Discrete model with states $x \in \mathcal{X}$
- Parameters w and probabilities $p(x|w)$

Boltzmann Machine

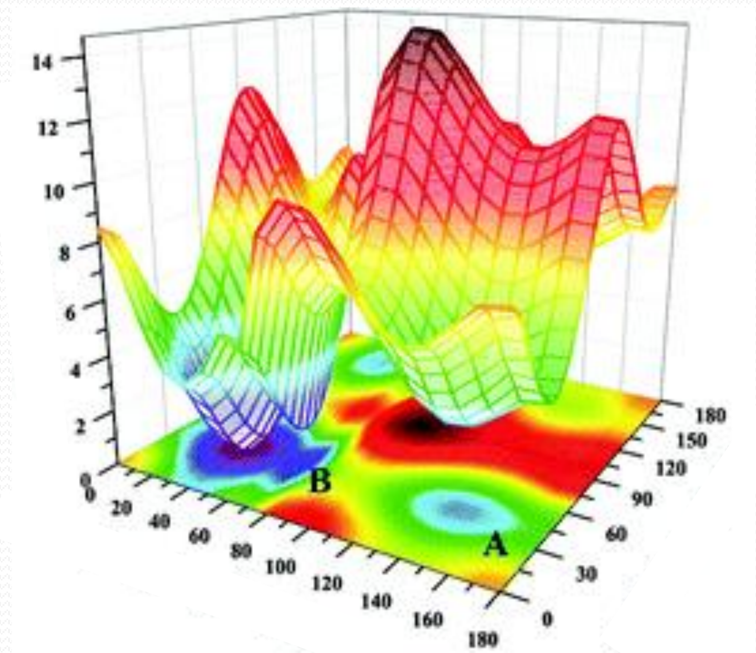
- Graph with n binary nodes (neurons)
- $\mathcal{X} = \{0, 1\}^n, |\mathcal{X}| = 2^n$
- Each neuron i has a bias b_i
- Each edge (i, j) has a weight W_{ij}

Energy-based Learning

- Probabilities defined in terms of *energy* $f(x|w)$

$$p(x|\omega) = \frac{e^{-f(x|\omega)}}{Z(\omega)} \quad \text{where } Z(\omega) = \sum_x e^{-f(x|\omega)}$$

- Energy wells represent likely states of the model



Boltzmann Machine

$$f(x|w) = -\sum\nolimits_{\text{edge }(i,j)} W_{ij} x_i x_j - \sum\nolimits_{\text{node } i} b_i x_i$$

Markov Chain Monte Carlo

- Partition function $Z(w)$ often difficult to compute, but it is easy to sample from model distributions using MCMC techniques such as Gibbs sampling

- A Markov chain is a sequence $X_0, X_1, X_2, \ldots \in \mathcal{X}$ of random variables such that
$$p(X_{t+1} | X_1, X_2, \ldots, X_t) = p(X_{t+1} | X_t).$$
- Consider the matrix T of *transition probabilities*
$$T_{yx} = p(X_{t+1} = y | X_t = x).$$

Markov Chain Monte Carlo

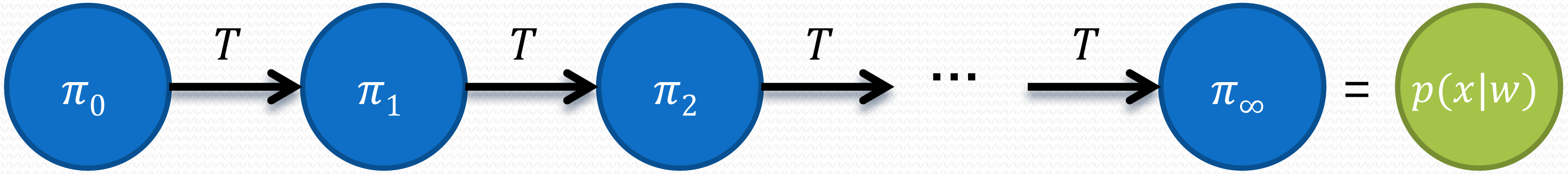
- Let the vector $\pi_t \in \mathbb{R}^{|\mathcal{X}|}$ be the distribution of X_t
- Then, we have $\pi_{t+1} = T\pi_t$ since

$$\begin{aligned}\pi_{t+1,y} &= p(X_{t+1} = y) \\ &= \sum_x p(X_{t+1} = y | X_t = x)\, p(X_t = x) \\ &= \sum_x T_{yx}\pi_{t,x}\end{aligned}$$

- By induction, $\pi_t = T^t\pi_0$.

Markov Chain Monte Carlo

- By choosing T carefully, we can get $\pi_t \to p(\cdot\,|w)$.
- This means that we can use the Markov chain to sample from $p(\cdot\,|w)$, provided
  - t is sufficiently large, and
  - the number of steps between samples is also large.



Gibbs Sampling

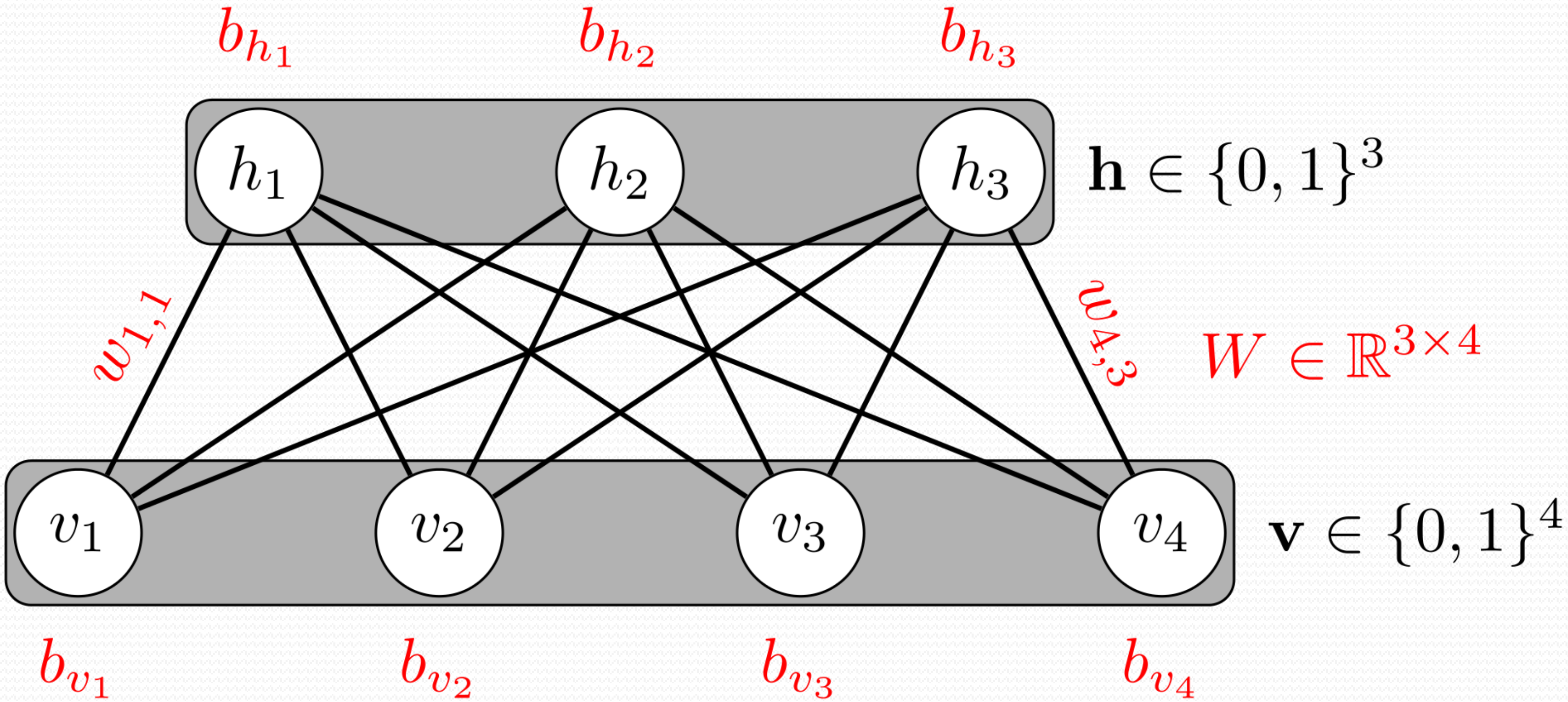
- Suppose each state x is a vector $(x_1, \dots, x_n)$.
- Let x_{-i} denote the vector with the entry x_i deleted.
- Consider the Markov chain where T_{yx} is nonzero only if the vectors y and x differ by one entry, say x_i.

$$T_{yx} = \frac{1}{n} p(y_i | x_{-i}) = \frac{1}{n} \frac{p(y_i, x_{-i})}{\sum_z p(z, x_{-i})}$$

$$= \frac{1}{n} \frac{\exp\left(-f(y_i, x_{-i})\right)}{\sum_z \exp\left(-f(z, x_{-i})\right)}$$

- If the x_i are binary, then $T_{yx} = \frac{1}{n} \operatorname{sig}\left(b_i + \sum_{j \neq i} W_{ij} x_j\right)$.

Restricted Boltzmann Machines



M. A. Cueto, J. Morton, B. Sturmfels. "Geometry of the restricted Boltzmann machine". Algebraic Methods in Statistics and Probability (AMS) 516 (2010)

M. Aoyagi. "Stochastic complexity and generalization error of a restricted Boltzmann machine in Bayesian estimation." JLMR 11 (2010): 1243-1272.

Contrastive Divergence

- MLE minimizes

$$\boxed{\ell(\omega) = \log Z(\omega) + \frac{1}{N}\sum_i f(x_i|\omega)}$$

- Gradient

$$\begin{aligned}\frac{\partial \ell}{\partial \omega} &= \mathbb{E}_{X_0}\left(\frac{\partial f(x|\omega)}{\partial \omega}\right) + \frac{\partial \log Z(\omega)}{\partial \omega} \\ &= \mathbb{E}_{X_0}\left(\frac{\partial f(x|\omega)}{\partial \omega}\right) - \mathbb{E}_{X_\infty}\left(\frac{\partial f(x|\omega)}{\partial \omega}\right)\end{aligned}$$

- CD relaxes it to

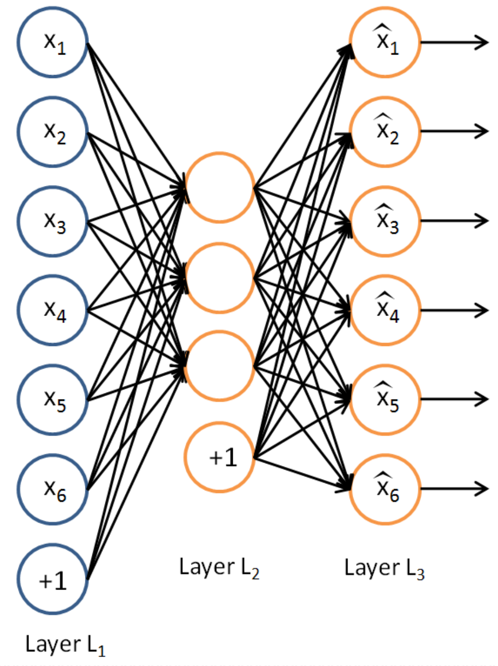
$$\boxed{\frac{\partial \ell}{\partial \omega} = \mathbb{E}_{X_0}\left(\frac{\partial f(x|\omega)}{\partial \omega}\right) - \mathbb{E}_{X_1}\left(\frac{\partial f(x|\omega)}{\partial \omega}\right)}$$

- This relaxation works well in experiments. It seems to regularize the model to prevent overfitting.

Hinton, Geoffrey E. "Training products of experts by minimizing contrastive divergence." Neural computation 14.8 (2002): 1771-1800.

Sparse Autoencoders

- Real-valued neurons (Andrew Ng)



Given data vectors $y_1, y_2, \ldots, y_N$, let

$$\begin{aligned} x_i^{(0)} &= y_i \\ x_i^{(1)} &= \mathrm{sig}(\, A^{(0)} x_i^{(0)} + b^{(0)}\,) \\ x_i^{(2)} &= \mathrm{sig}(\, A^{(1)} x_i^{(1)} + b^{(1)}\,) \end{aligned}$$

Minimize over all weights $A^{(0)}, b^{(0)}, A^{(1)}, b^{(1)}$

$$\sum_i \left\| x_i^{(2)} - x_i^{(0)} \right\|^2 + \beta \left\| x_i^{(1)} \right\|_1 + \lambda \left\| A^{(0)} \right\|_2^2 + \lambda \left\| A^{(1)} \right\|_2^2$$

Sparsity Penalty

Weight Decay

- Mean-field approximation of RBM

Andrew Ng. “Deep Learning, Self-Taught Learning and Unsupervised Feature Learning,” (http://www.youtube.com/watch?v=n1ViNeWhC24) 2012.
A. Ng, J. Ngiam, C.Y. Foo, Y. Mai, and C. Suen. “Unsupervised feature learning and deep learning tutorial.” (http://ufldl.stanford.edu/wiki/index.php) 2011.

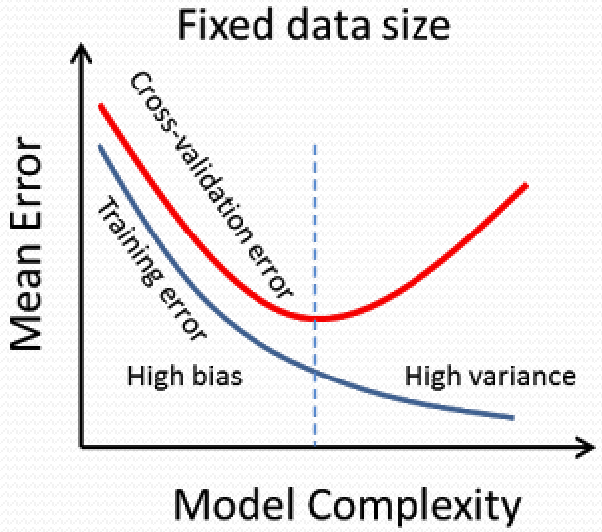
Deep Issues

- Slow hyperparameter search (uses cross-validation)

$$\sum_i \left\| x_i^{(2)} - x_i^{(0)} \right\|^2 + \beta \left\| x_i^{(1)} \right\|_1 + \lambda \left\| A^{(0)} \right\|_2^2 + \lambda \left\| A^{(1)} \right\|_2^2$$

Sparsity Penalty

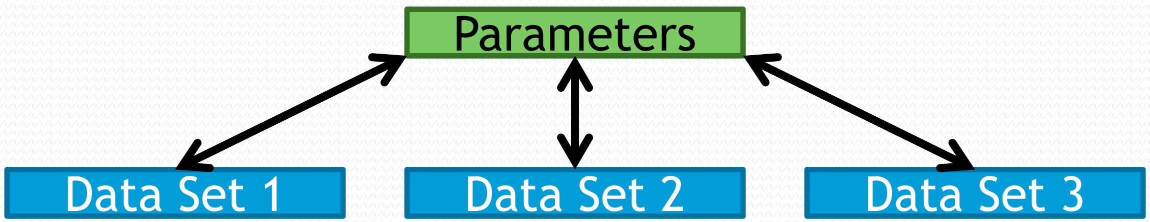
Weight Decay



Deep Issues

- Costly to train massive neural networks

e.g. cat video experiment (2012): 9 layers + 10m images + 16k cores => 3 days of training



“What we’re missing is the ability to parallelize the training of the network, mostly because the **communication is the killer**.”

Yann LeCun, NYU

“A lot of people are trying to solve this problem at Baidu, Facebook, Google, Microsoft and elsewhere […] it is **both a hardware and software issue**.”

“Eventually a lot of the deep learning task will be done on the device, which will keep pushing the need for **on-board neural network accelerators** that operate at very low power and can be trained and used quickly.”

http://www.theplatform.net/2015/05/11/deep-learning-pioneer-pushing-gpu-neural-network-limits/

Regularization

Why does deep learning work?

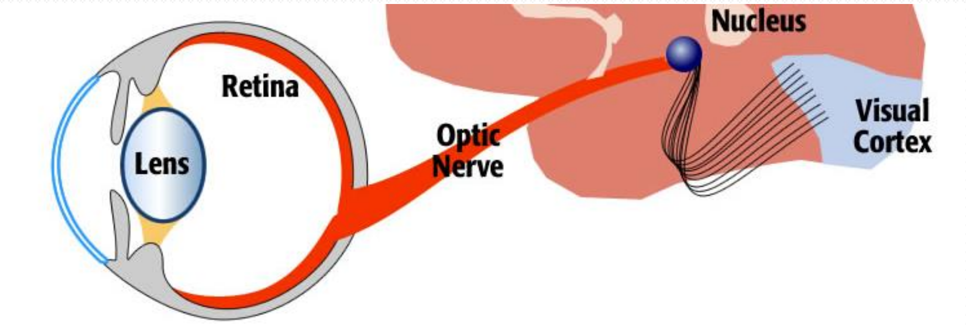
It's not really about the RBMs…

- Greedy layer-wise initialization of neuron weights can be any of the following:
  - Contrastive divergence
  - Sparse autoencoder
  - Sparse coding
  - K-means clustering
  - Random data vectors

Coates, Adam, and Andrew Y. Ng. "Learning feature representations with k-means." Neural Networks: Tricks of the Trade. Springer 2012. 561-580.
Coates, Adam, and Andrew Y. Ng. "The importance of encoding versus training with sparse coding and vector quantization." ICML 2011.

Why does deep learning work?

… but it has something to do with regularization.



Compressed Sensing

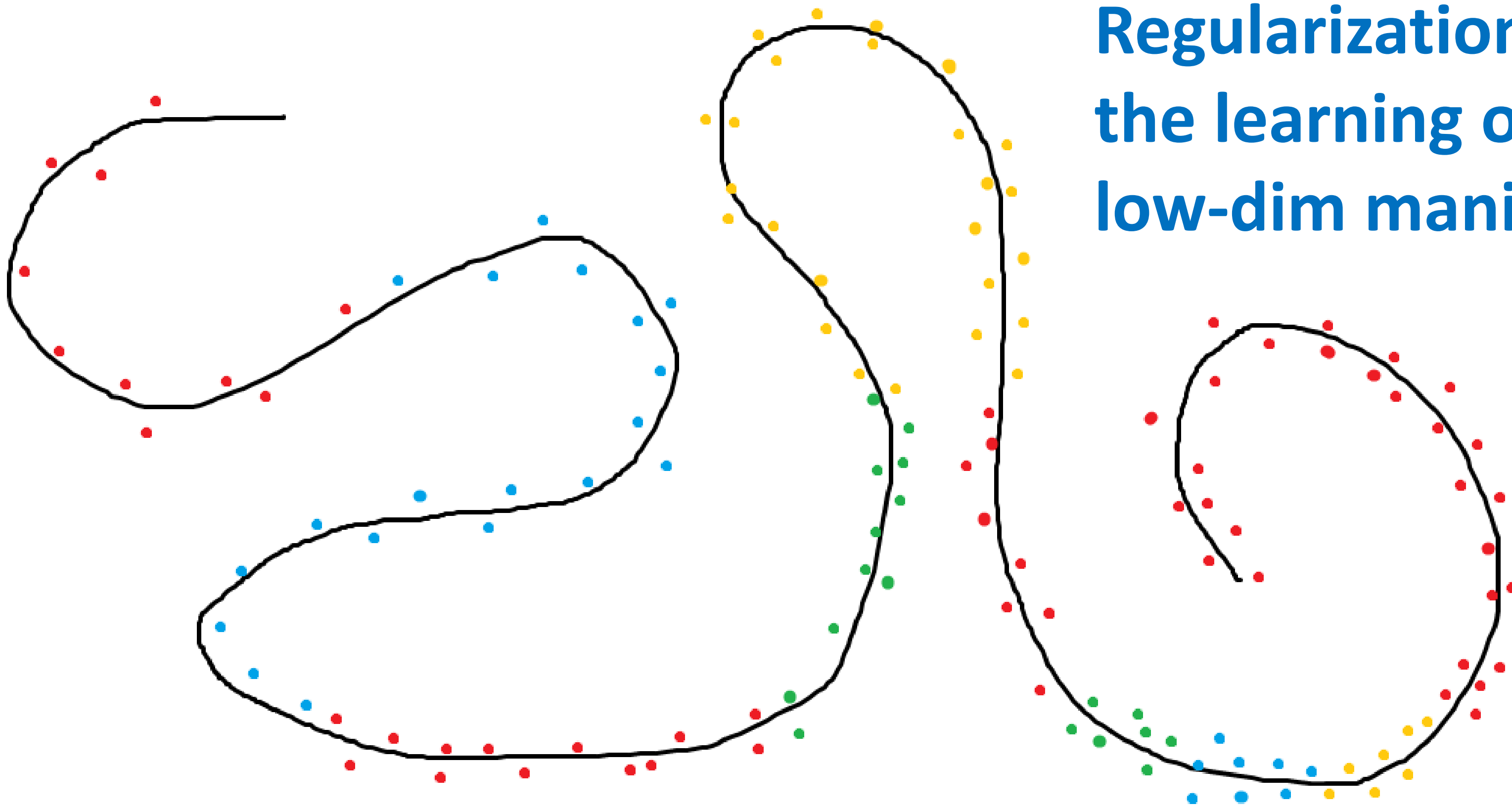
- If input is sparse in some known basis, then it can be reconstructed from almost any random projections.

Donoho, D.L. (2006). "Compressed sensing". IEEE Transactions on Information Theory 52 (4): 1289.
Candès, Emmanuel J.; Romberg, Justin K.; Tao, Terence (2006). "Stable signal recovery from incomplete and inaccurate measurements". CPAM 59 (8): 1207.
Hillar, Christopher, and Friedrich T. Sommer. "When can dictionary learning uniquely recover sparse data from subsamples?." arXiv:1106.3616 (2011).

Why Regularization?

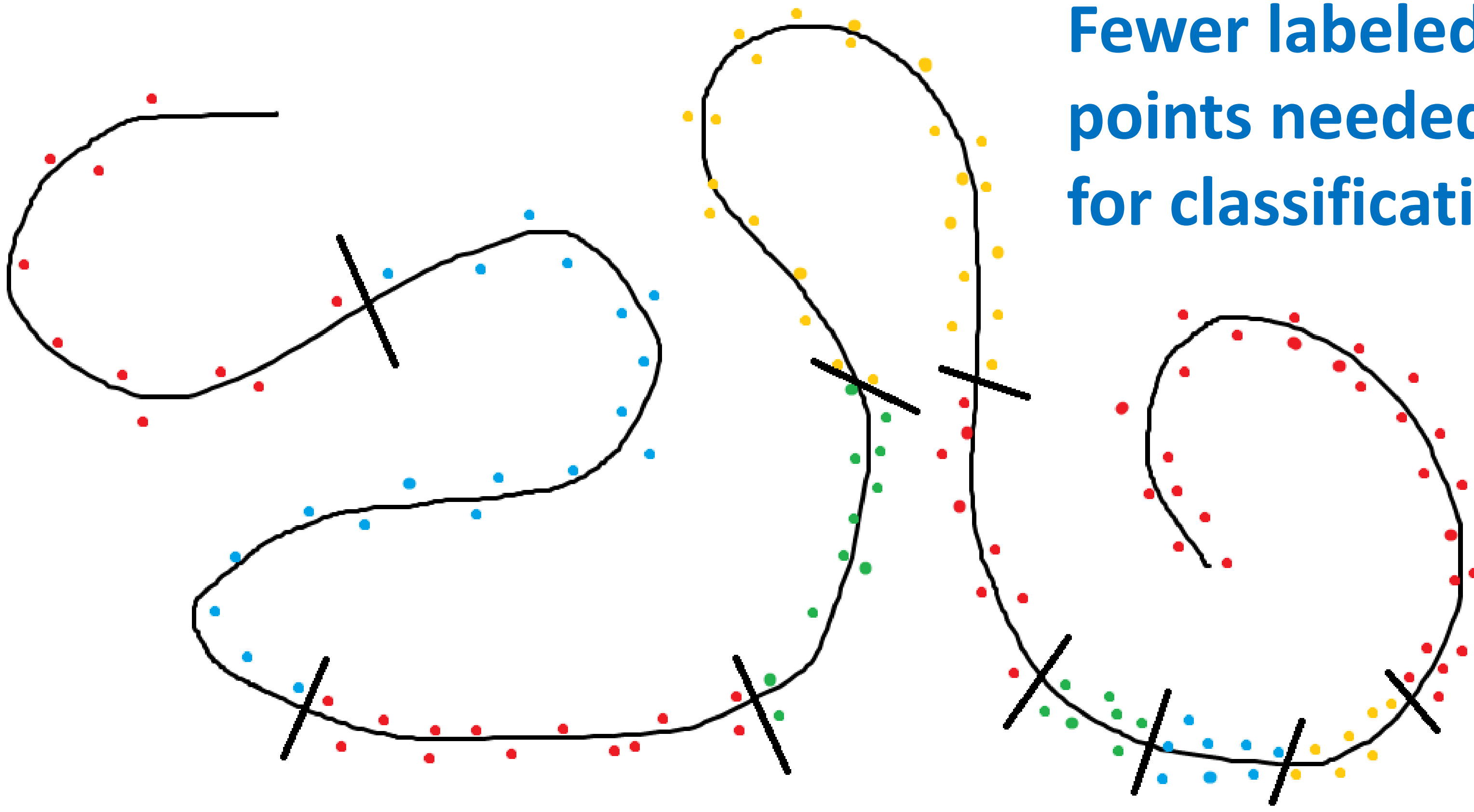
Data is often near low-dim manifold in high-dim space

Why Regularization?



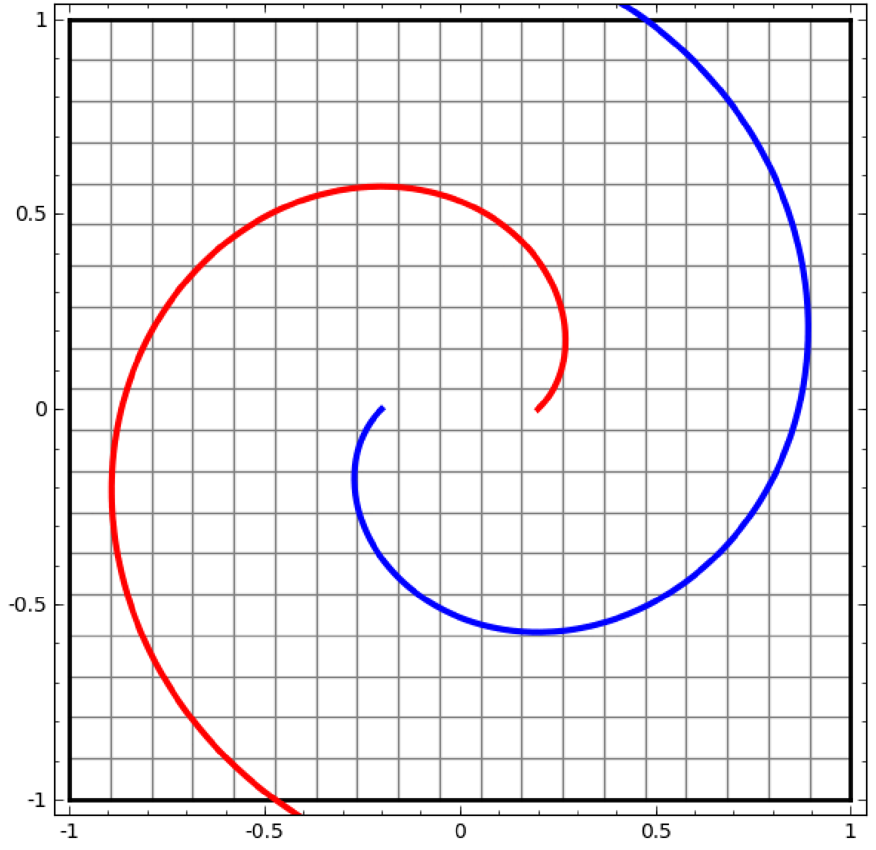
Regularization aids the learning of the low-dim manifold

Why Regularization?



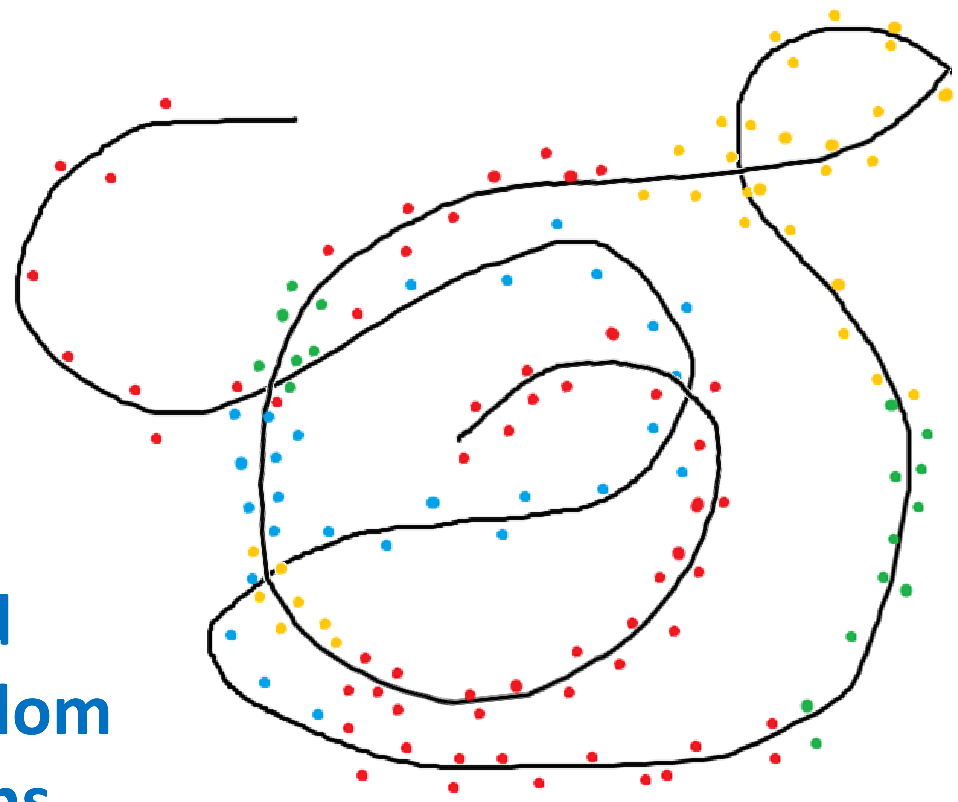
Fewer labeled points needed for classification

Why Regularization?



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Why Regularization?



Structure preserved after random projections

Probability Flow

Through the Lens of Computation

Model = Family of Distributions

- Statistical learning often focuses only on deriving optimal parameters or posterior distributions
- Computational considerations are put off till later

Model = Family of Distributions + Computations

- What if models are defined and optimized with the computations to be performed in mind?
- Study models whose goal is to approximate data distributions and generate samples using MCMC.

Through the Lens of Computation

Model = Family of Distributions

1. Define the model
2. Define the objective
3. Optimize

Model = Family of Distributions + Computations

1. Define the model
2. Define the computation
3. Define an objective that depends on computation
4. Optimize

Minimum Probability Flow

$$\pi_0 \xrightarrow{T} \pi_1 \xrightarrow{T} \pi_2 \xrightarrow{T} \cdots \xrightarrow{T} \pi_\infty = p(x|w)$$

- New look at learning
  - Tweaking w so that the π_i are "close" to the data
  - Use Kullback-Leibler (KL) distance for closeness
- Different kinds of learning
  - min $\mathrm{KL}(\pi_0 \,||\, \pi_\infty)$ ~ Maximum likelihood
  - min $\mathrm{KL}(\pi_0 \,||\, \pi_\infty) - \mathrm{KL}(\pi_1 \,||\, \pi_\infty)$ ~ Contrastive divergence
  - min $\mathrm{KL}(\pi_0 \,||\, \pi_\varepsilon)$ ~ Minimum probability flow

Minimum Probability Flow

$$\pi_0 \xrightarrow{T} \pi_1 \xrightarrow{T} \pi_2 \xrightarrow{T} \cdots \xrightarrow{T} \pi_\infty = p(x|w)$$

- Think of π_t as a flow in continuous time with transition rate matrix Γ, i.e. $\mathrm{T} = \exp(\Gamma)$ and $\pi_t = \exp(\Gamma t)\,\pi_0$.
- In MPF, we assume that Γ has the form

$$\Gamma_{xy}(\omega) = g_{xy} \exp\left[\frac{f(y|\omega) - f(x|\omega)}{2}\right]$$

where x, y are states, and g_{xy} the connectivity matrix

Minimum Probability Flow

$$\pi_0 \xrightarrow{T} \pi_1 \xrightarrow{T} \pi_2 \xrightarrow{T} \cdots \xrightarrow{T} \pi_\infty = p(x|w)$$

- In MPF, we minimize the objective function [Sohl-Dickstein, Battaglino, DeWeese 2011]

$$\mathrm{KL}(X_0 \| X_\varepsilon) = \varepsilon \frac{\partial \mathrm{KL}(X_0 \| X_t)}{\partial t}\bigg|_{t=0} = \frac{\varepsilon}{N} \sum_{x \in X_0} \sum_{y \notin X_0} g_{xy} \exp\left[\frac{f(y|\omega) - f(x|\omega)}{2}\right]$$

- Minimize total flow out of data points

Sohl-Dickstein, Jascha, Peter B. Battaglino, and Michael R. DeWeese. "New method for parameter estimation in probabilistic models: minimum probability flow." Physical review letters 107.22 (2011): 220601.

$$
\begin{aligned}
\frac{\partial}{\partial t}\mathrm{KL}(p^{(0)}\|p^{(t)}(\theta))|_{t=0} &= \sum_{x\in\mathcal{D}} p^{(0)}(x)\frac{\partial}{\partial t}\log p^{(t)}(x|\theta)|_{t=0} \\
&= -\sum_{x\in\mathcal{D}} p^{(0)}(x)\frac{\dot{p}^{(t)}(x|\theta)}{p^{(t)}(x|\theta)}|_{t=0} \\
&= -\sum_{x\in\mathcal{D}} \dot{p}^{(0)}(x|\theta) \\
&= -\sum_{x\in\mathcal{D}} \Bigg(\sum_{\substack{y\in\mathcal{X}\\ y\neq x}} \Gamma_{xy}p^{(0)}(y) - \sum_{\substack{y\in\mathcal{X}\\ y\neq x}} \Gamma_{yx}p^{(0)}(x)\Bigg) \\
&= -\sum_{x\in\mathcal{X}}\sum_{\substack{y\in\mathcal{X}\\ y\neq x}} \mathbb{1}_{x\in\mathcal{D}}\Gamma_{xy}p^{(0)}(y) + \sum_{y\in\mathcal{X}}\sum_{\substack{x\in\mathcal{X}\\ x\neq y}} \mathbb{1}_{x\in\mathcal{D}}\mathbb{1}_{y\in\mathcal{D}}\Gamma_{xy}p^{(0)}(y) \\
&= \sum_{y\in\mathcal{D}} p^{(0)}(y)\sum_{\substack{x\notin\mathcal{D}\\ x\neq y}} \Gamma_{xy}
\end{aligned}
$$

Minimum Probability Flow

$$\pi_0 \xrightarrow{T} \pi_1 \xrightarrow{T} \pi_2 \xrightarrow{T} \cdots \xrightarrow{T} \pi_\infty = p(x|w)$$

- No sampling is required (compare with CD) and edge updates depend only on state of the endpoints!

$$h_i = e^{z_i \delta_i}$$
$$\Delta b_i = \delta_i h_i$$
$$\Delta W_{ij} = x_j \delta_i h_i + x_i \delta_j h_j$$

where

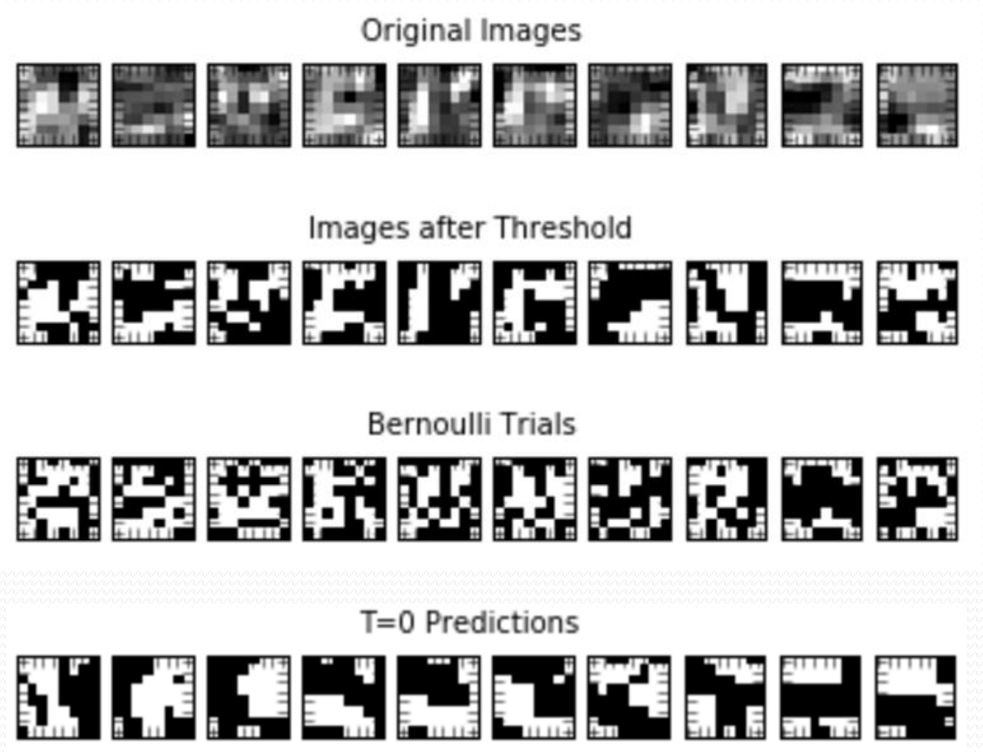
$$\delta = \frac{1}{2} - x$$
$$z = xW + b$$

- Joint work with Chris Hillar (UC Berkeley)

Sohl-Dickstein, Jascha, Peter B. Battaglino, and Michael R. DeWeese. "New method for parameter estimation in probabilistic models: minimum probability flow." Physical review letters 107.22 (2011): 220601.

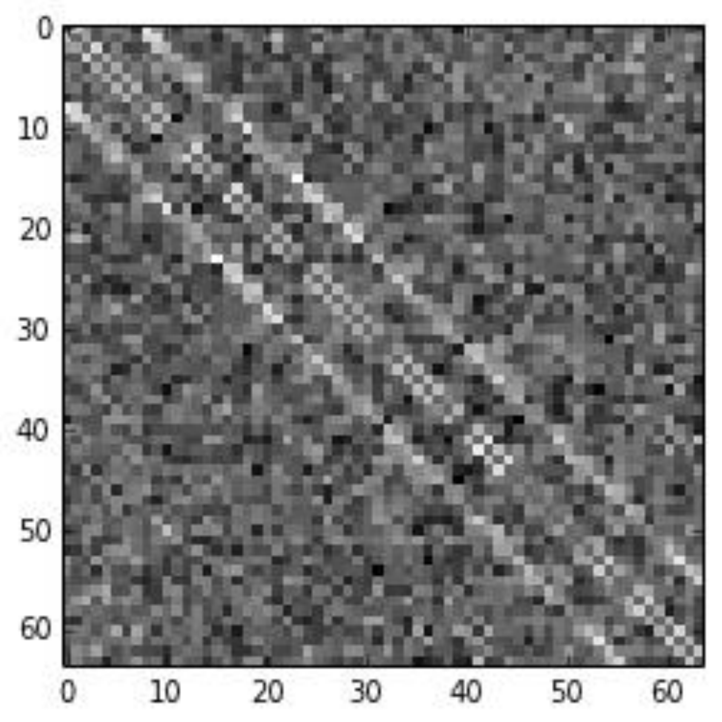
Natural Images

- Apply MPF to complete Boltzmann Machine (no hidden layer) to 8x8 patches from natural images



Natural Images

- Weight matrix learnt strong correlations between nodes corresponding to neighboring pixels



Massive Neural Networks

- Minimum probability flow
  - Greedily minimizing computation time needed for MCMC to converge near empirical distribution
  - Will allow us to train much bigger neural networks, through model-parallelism, on multiple GPUs
- Deep probability flow
  - Currently conducting experiments with a version of MPF that contains hidden variables
  - Learning algorithm derived using Variational Bayes

Thank you

http://people.sutd.edu.sg/~shaowei_lin/