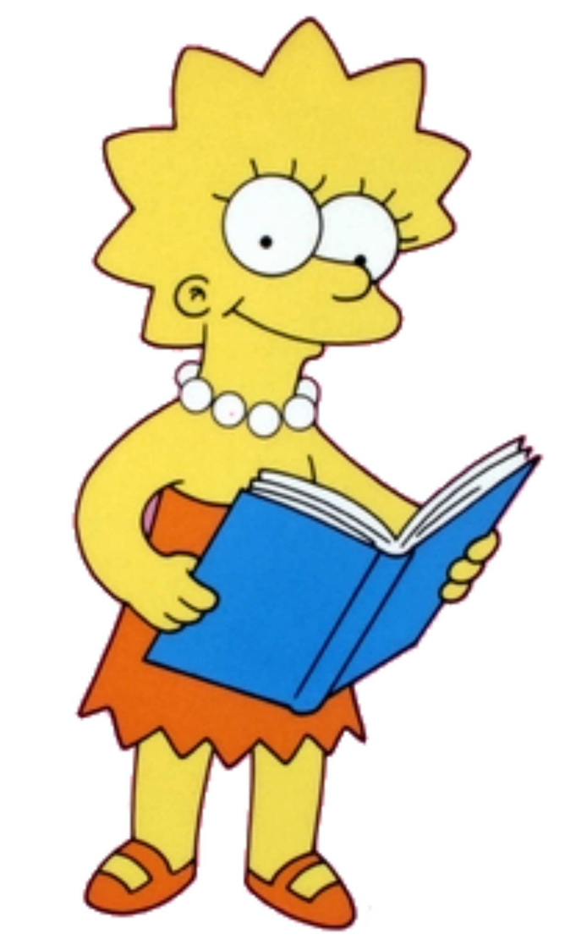
Deep Woods

Université de Montréal

Yoshua Bengio   Hugo Larochelle   Joseph Turian



Introduction

(Hinton, Osindero and Teh, 2006; Hinton and Salakhutdinov, 2006; Bengio et al., 2007; Ranzato et al., 2007; Lee, Ekanadham and Ng, 2008) successfully train deep architectures
Underlying commonalities in these works:

- Unsupervised learning to initialize each layer in the network
- Each layer learns a representation of its input that serves as input for the next layer, and training progresses greedily

Principle of training a deep architecture by greedy layer-wise unsupervised training has been shown to be successful for deep **connectionist** architectures

Hypothesis

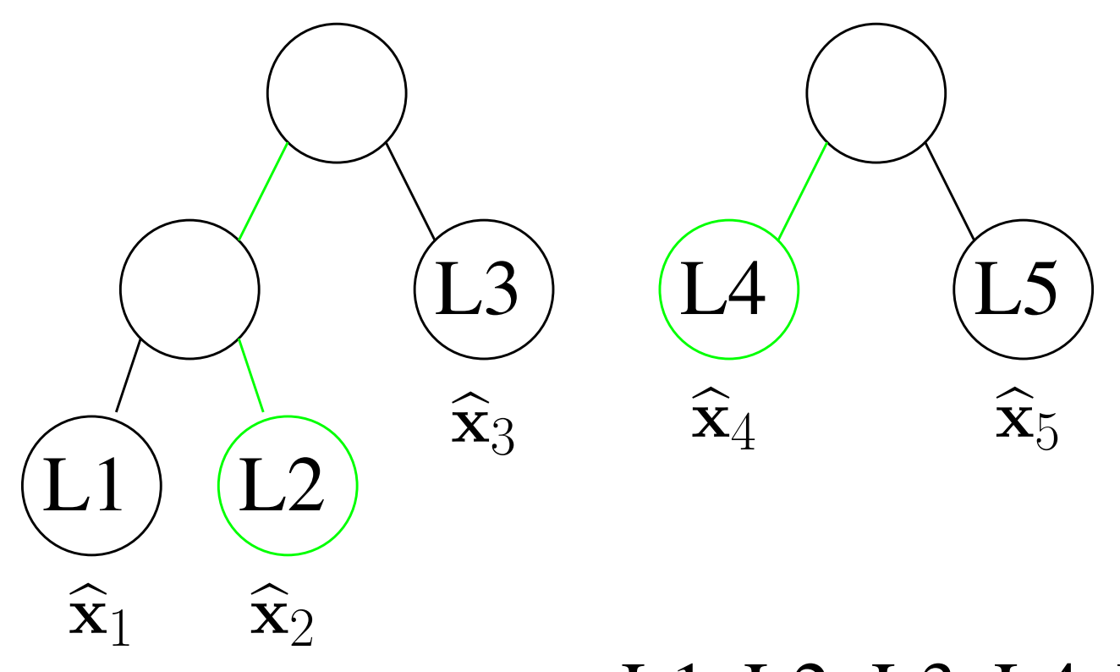
This deep training principle applies to deep architectures that comprise **other kinds** of primitive units
Attempt to exploit this principle $\Rightarrow$ new deep architectures based on ensembles of deterministic or stochastic decision trees?
Distributed representation (Hinton, 1986) output at each level:

- For deterministic trees, transform input $\mathbf{x}$ into a binary code $\mathbf{h}(\mathbf{x})$ representing the leaves in the ensemble into which the input falls, with one bit per node
- For probabilistic trees, transform input $\mathbf{x}$ into the probability vector $\widehat{\mathbf{h}}(\mathbf{x})$ representing the likelihood of the input falling into each node in the ensemble, with one entry per node

Reconstruction Trees

Idea

Boost an ensemble of decision trees to reconstruct the input, minimizing some choice of reconstruction loss, e.g. log-loss for binary inputs
Also, can be very sparse in features used for splitting, i.e. splitting features need not be equivalent to features used in reconstruction



code: $\mathbf{h}(\mathbf{x})$ =

L1	L2	L3	L4	L5
–1	+1	–1	+1	–1

reconstruction: $\widehat{\mathbf{x}} = \widehat{\mathbf{x}}_2 + \widehat{\mathbf{x}}_4$

Training technique

Gradient-based greedy ensemble induction (Turian, 2007), generalized to multilabel task, one label per reconstruction dimension
Choose decision tree splits with steepest reconstruction loss gradient

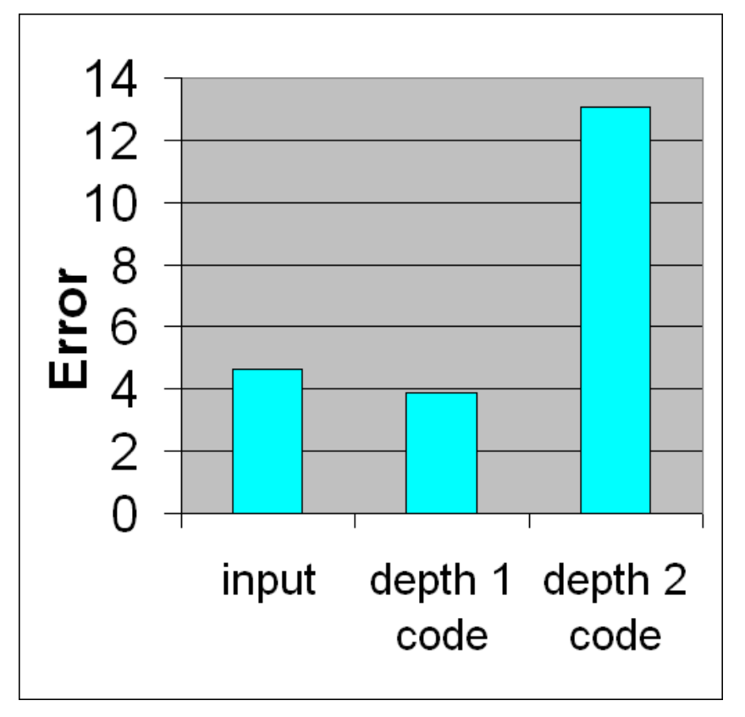
Disadvantage

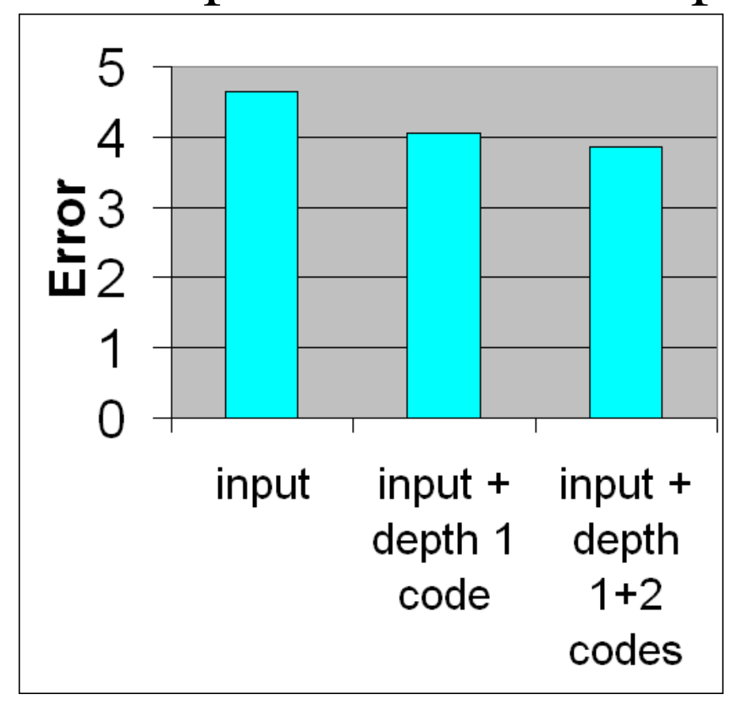
Slow to train
Speed of ordinary tree boosting $\times$ # reconstruction dimensions

Results

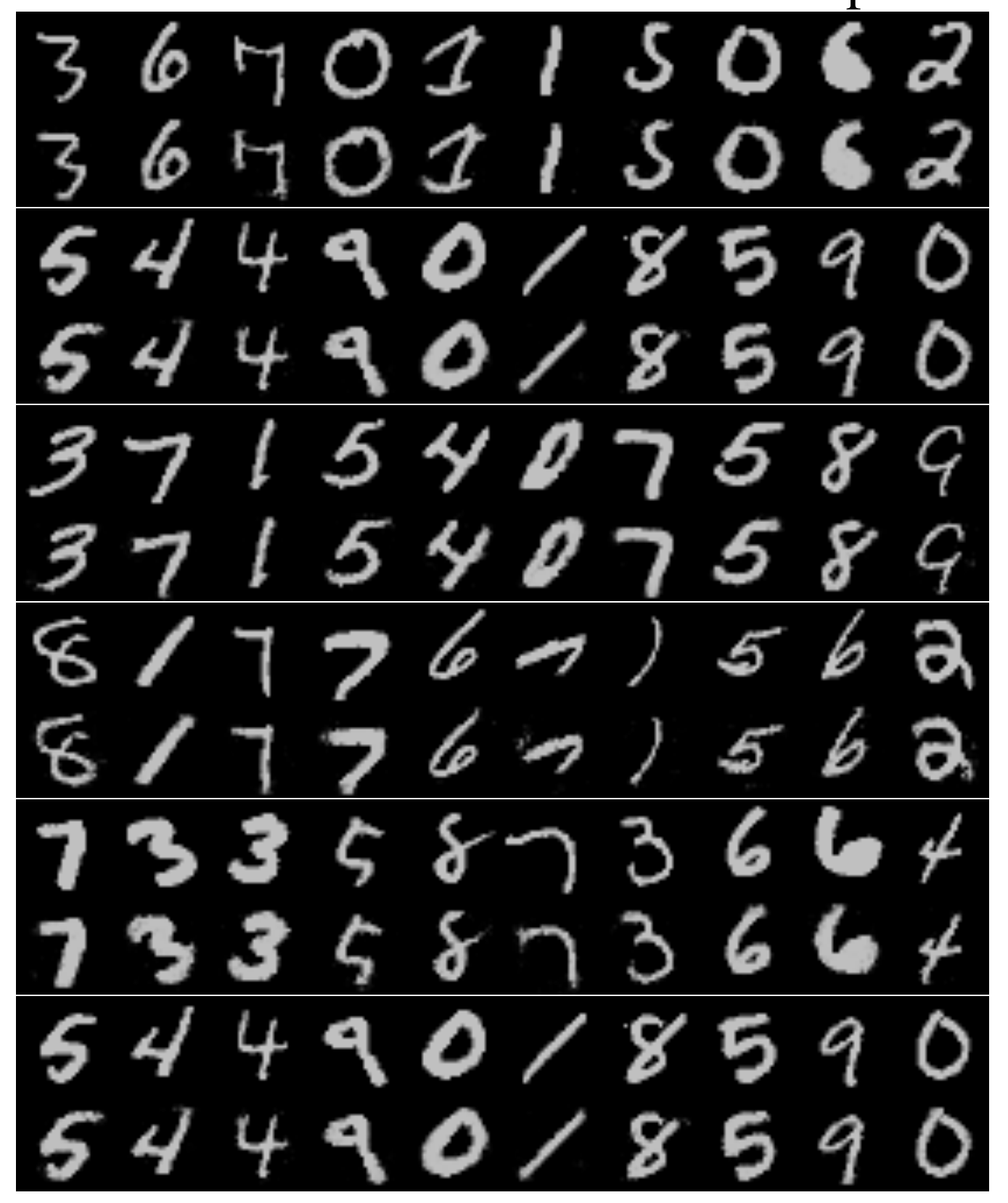
Data: MNIST discretized to four levels of grayscale, downsampled to 10K training examples

MNIST error of NN trained on different representations of input





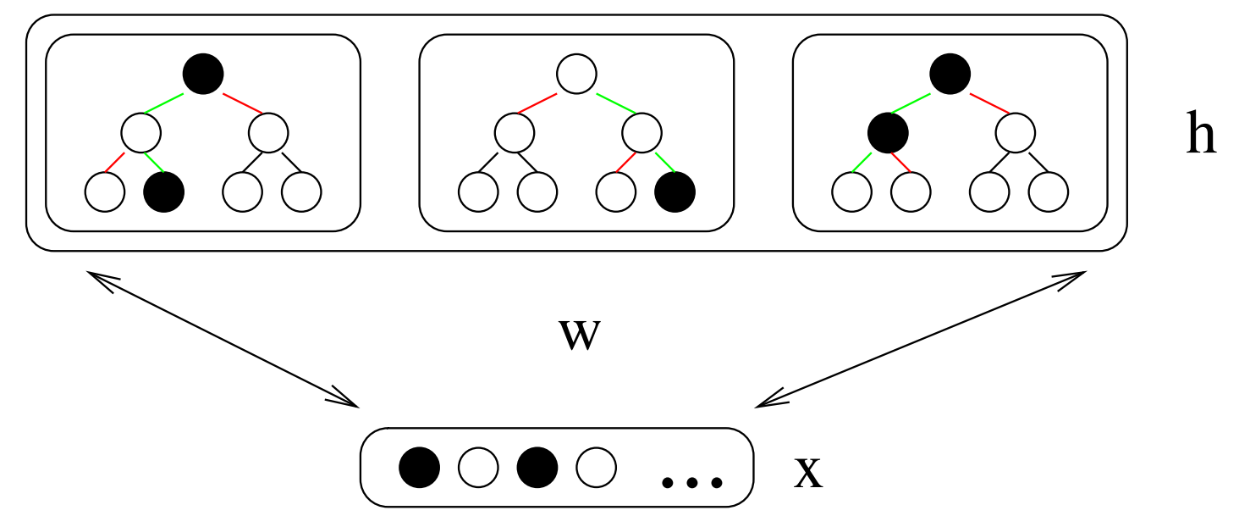
Reconstruction of validation examples



Future work

(Bengio et al., 2007) indicate that greedy unsupervised layer-wise training is most useful when followed by supervised fine-tuning.

Restricted Boltzmann Forests (RBFs)



Idea

Extend RBM framework to hidden layers with tree-structured groups of units, in order to obtain more complex representation $\widehat{\mathbf{h}}(\mathbf{x})$ of inputs
RBM = RBF with depth 0 trees

Model

$$E_\lambda(\mathbf{x},\mathbf{h}) = -\mathbf{b}^T\mathbf{x} - \mathbf{h}^T\mathbf{W}\mathbf{x} - \mathbf{c}^T\mathbf{h} + \lambda \cdot \Omega(\mathbf{h})$$

$\Omega(h)$ penalizes violations of tree constraints: one of two children subtrees under **node** gets shut off based on **node** value
Train with $\lambda \to \infty$, thus completely disallowing non-tree configurations

- $\Pr(\mathbf{x}|\mathbf{h})$ is same as with traditional RBM
- $\Pr(\mathbf{h}|\mathbf{x}) = \prod_{tree} \Pr(\mathbf{h}_{nodes \in tree}|\mathbf{x})$. Each $\Pr(\mathbf{h}_{nodes \in tree}|\mathbf{x})$ can be computed in $O(|nodes \in tree|)$, so inference remains efficient by exploiting tree structure and sharing computations made at nodes in different levels

Training

Contrastive Divergence (Hinton, 2000) can be used just as in a regular RBM
Difference is in:

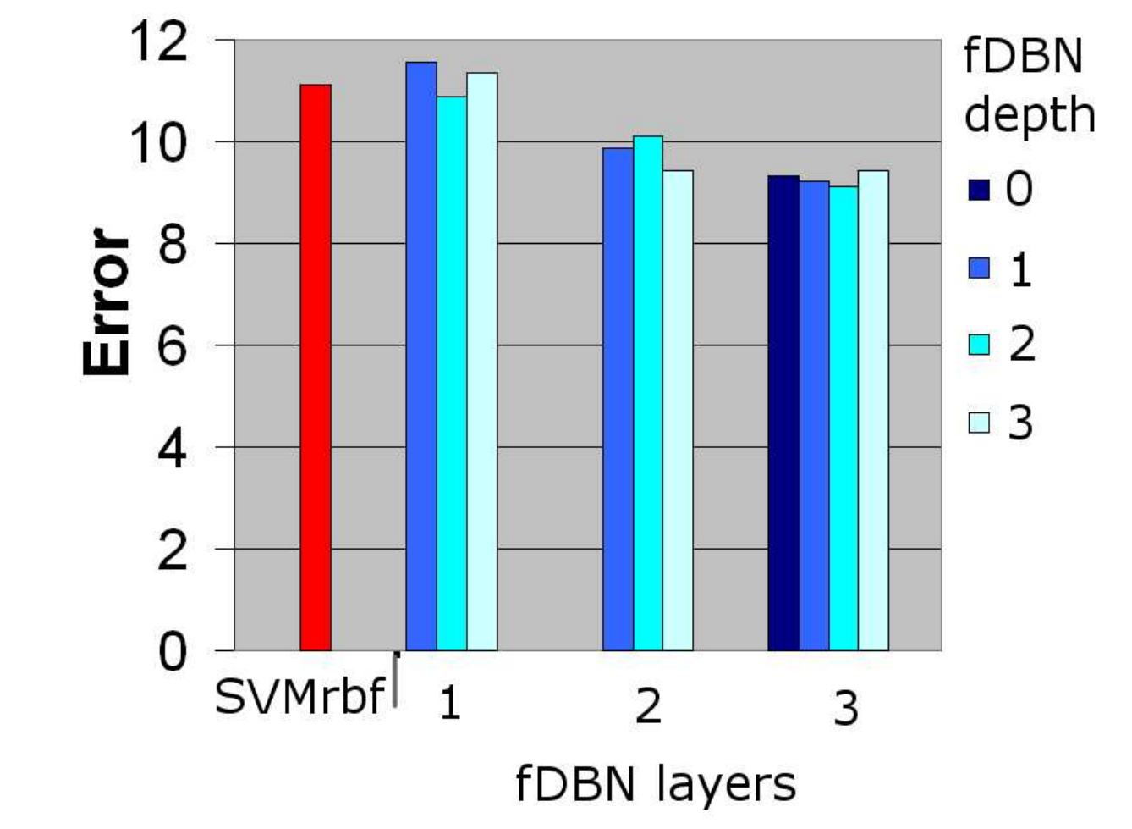
- sampling procedure of hidden layer given a value for input layer

  and
- in computation of $\Pr(h_k = 1|\mathbf{x})$ for positive and negative phase updates

Classification results

fDBN = DBN stacking RBFs instead of RBMs
DBN = fDBN with depth 0 trees

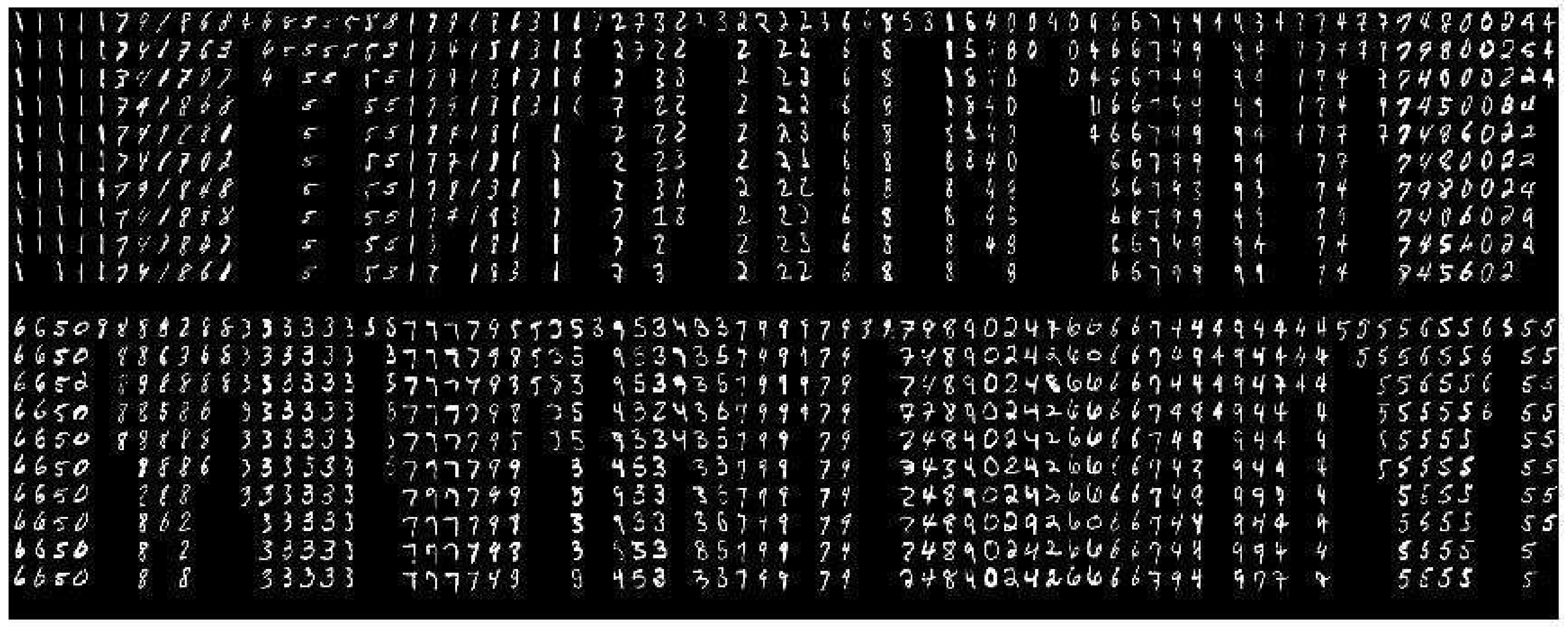
Results on the rotated MNIST digit dataset



Results

Clusterings learned by an RBF with a single tree of depth 6 (64 leaves, 128 clusters)

Clusterings learned by an RBF with two trees of depth 5 (64 leaves total, 4096 clusters total)



Future work

- Extentions for trees with arbitrary graph structure and branching factors can be derived
- Learn the topology of the trees, possibly different in each tree, using gradually decreasing ℓ_1 regularization + greedy constructive training