

How to train a mixture of 100.000.000.000.000.000 classifiers

Roland Memisevic

May 17, 2010

Log-linear models

- ▶ Task: Map $\mathbf{x} \in \mathcal{R}^n$ to class-label y .
- ▶ Linear approach:
 1. Define the *linear* score function

$$s_y(\mathbf{x}) = \mathbf{w}_y^t \mathbf{x}$$

2. Turn scores into probabilities

$$p(y|\mathbf{x}) = \frac{\exp(\mathbf{w}_y^t \mathbf{x})}{\sum_{y'} \exp(\mathbf{w}_{y'}^t \mathbf{x})}$$

- ▶ Training by maximizing $\sum_{\alpha} \log p(y^{\alpha}|\mathbf{x}^{\alpha})$.
- ▶ Benefits: Training convex; outputs probabilistic.
- ▶ AKA logistic regression, “maximum entropy”, ...

A log-bilinear model

- ▶ Introduce a set of *binary latent variables* $\mathbf{h} = (h_1, \dots, h_K)$
- ▶ A bilinear approach:

1. Define the *bilinear* score function

$$s_y(\mathbf{x}, \mathbf{h}) = \mathbf{h}^t W_y \mathbf{x}$$

2. Turn scores into probabilities

$$p(y, \mathbf{h}|\mathbf{x}) = \frac{\exp(\mathbf{h}^t W_y \mathbf{x})}{\sum_{y', \mathbf{h}'} \exp(\mathbf{h}'^t W_{y'} \mathbf{x})}$$

3. Marginalize over \mathbf{h} : $p(y|\mathbf{x}) = \sum_{\mathbf{h}} p(y, \mathbf{h}|\mathbf{x})$

- ▶ Training non-linear (see below); outputs still probabilistic.

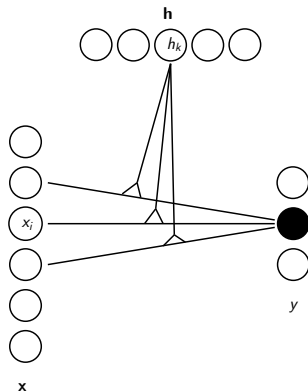
A log-bilinear model

- ▶ Marginalization tractable!

$$\begin{aligned} p(y|\mathbf{x}) &= \sum_{\mathbf{h}} p(y, \mathbf{h}|\mathbf{x}) \\ &\propto \sum_{\mathbf{h}} \exp(\mathbf{x}^t W_y \mathbf{h}) \\ &= \sum_{\mathbf{h}} \exp\left(\sum_{ik} W_{yik} x_i h_k\right) \\ &= \prod_k \left(1 + \exp\left(\sum_i W_{yik} x_i\right)\right) \end{aligned}$$

- ▶ Product of uni-softmax.
- ▶ Old trick. Used, for example, in [Hinton, 2007], [Larochelle et.al, 2008], [Nair, Hinton, 2008], ...
- ▶ Here, **three-way** cliques combine x_i , h_k and y .
- ▶ Model is a special case of [Memisevic, Hinton, 2006]:

Transforming pixels into labels



- ▶ Each single **hidden unit** can blend in a matrix W_k .
- ▶ Each **mixture component** is therefore a blend of K matrices.
- ▶ So each **mixture component** is **one** logistic regressor!

Training

Objective

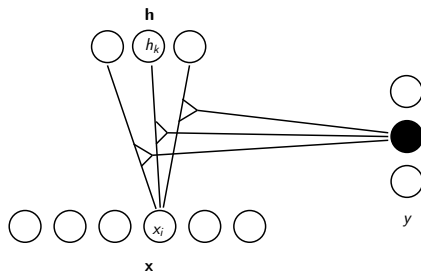
$$\log p(y|\mathbf{x}) = \sum_k \log(1 + \exp(\sum_i W_{yik} x_i)) - \log \sum_{y'} \prod_k (1 + \exp(\sum_i W_{y'ik} x_i))$$

Derivatives

$$\frac{\partial \log p(y|\mathbf{x})}{\partial W_{yik}} = (\delta_{\bar{y}y} - p(\bar{y}|\mathbf{x})) \sigma(\sum_i x_i W_{yik} h_k) x_i$$

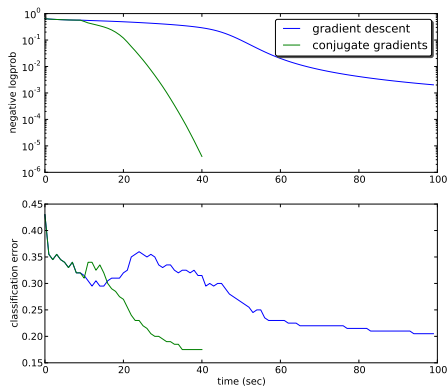
- ▶ Similar for biases.
- ▶ Works OK, but
- ▶ Local optima an issue.

A different view: The RBM Bayes Classifier



- ▶ Each class defines its own RBM.
- ▶ Discriminative version of a (would-be) RBM Bayes classifier.
- ▶ Suggests to use class-specific RBM as initialization.
- ▶ Also suggests how to make an RBM generative classifier work: Plug it in!

Training



- ▶ Good strategy (empirically):
- ▶ Train class-specific RBMs.
- ▶ Then switch to conjugate gradients.

Rectangles

- ▶ Distinguish horizontal from vertical rectangles. [Larochelle et.al, 2007]

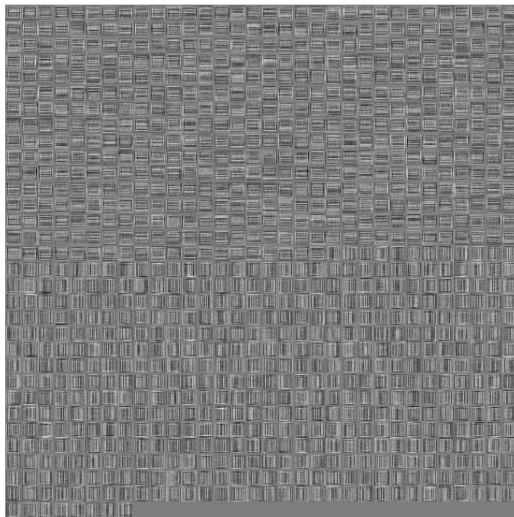


- ▶ Performance:

SVMRBF	SVMPOL	NNet	DBN3	SAA3	DBN1	gated softmax
2.15	2.15	7.16	2.60	2.41	4.71	0.564

- ▶ Learned W^y :

Rectangles filters



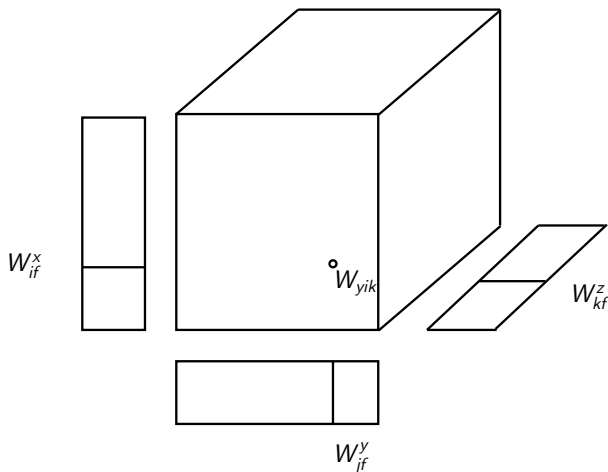
Class-independent filters

- ▶ Model learns one set of basis functions for each class.
- ▶ To learn a **class-independent** basis we can factorize

$$W_{yik} = \sum_f W_{if}^x W_{yf}^y W_{kf}^h$$

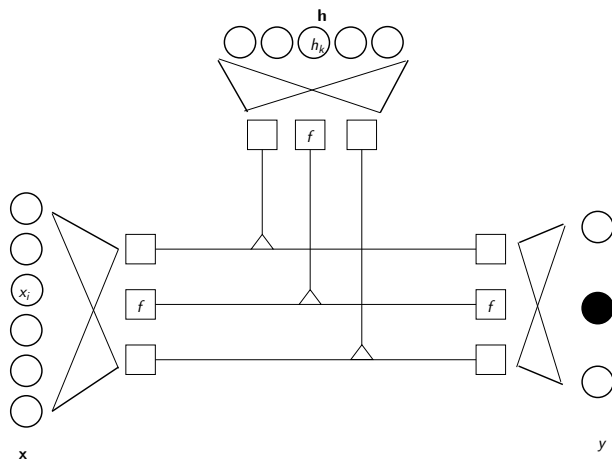
- ▶ This can also help reduce the number of parameters.

Factorization



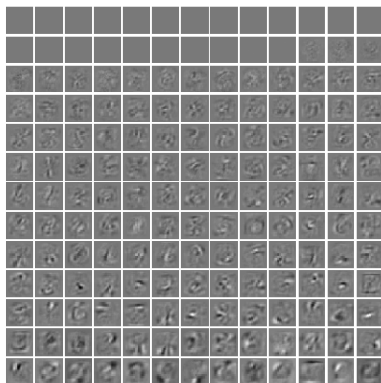
$$W_{yik} = \sum_f W_{if}^x W_{yf}^y W_{kf}^z$$

Factorization



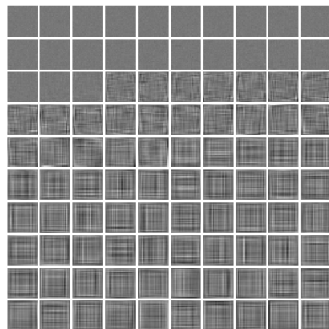
$$\log p(y, \mathbf{h}|\mathbf{x}) \propto \sum_f \left(\sum_i x_i W_{if}^x \right) \left(\sum_k h_k W_{kf}^h \right) W_{yf}^y$$

MNIST



Implicit RBM mixtures	SVM	Logistic Regression	Gated softmax
3.36	1.40	7.28	1.74

Rectangles, factored model



SVMRBF	SVMPOL	NNet	DBN3	SAA3	DBN1	gated softmax
2.15	2.15	7.16	2.60	2.41	4.71	0.826

Rectangles with images

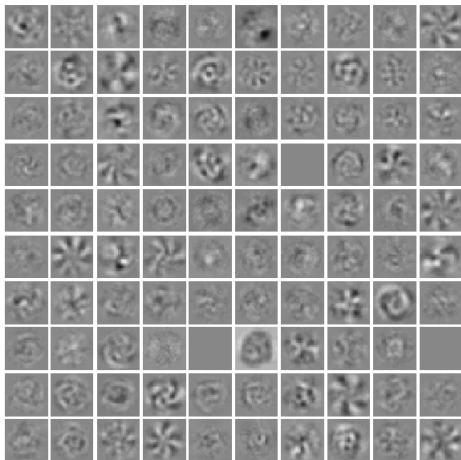


SVMRBF	SVMPOL	NNet	DBN3	SAA3	DBN1	gated softmax
24.04	24.05	33.20	22.50	24.05	23.69	24.81

Rotated digits 

SVMRBF	SVMPOL	NNet	DBN3	SAA3	DBN1	gated softmax
11.11	15.42	18.11	10.30	10.30	14.69	12.75

Rotated digits



Convex vs Nonconvex



SVMRBF	SVMPOL	NNet	DBN3	SAA3	DBN1	gated softmax
19.13	19.82	32.25	18.63	18.41	19.92	17.08

(unfactored model: 21.026)

Digits, random background



SVMRBF	SVMPOL	NNet	DBN3	SAA3	DBN1	gated softmax
14.58	16.62	20.04	06.73	11.28	09.80	10.48

Digits, background image



SVMRBF	SVMPOL	NNet	DBN3	SAA3	DBN1	gated softmax
22.61	24.01	27.41	16.31	23.00	16.15	23.65

Digits, rotated + background



SVMRBF	SVMPOL	NNet	DBN3	SAA3	DBN1	gated softmax
55.18	56.41	62.16	47.39	51.93	52.21	55.82

Conclusions

- ▶ Averaging many things has often proven to be a good thing.
- ▶ Results indicate that:

100.000.000.000.000.000 *logistic regressors*
tend to outperform one
 ∞ -*dimensional SVM*

- ▶ Training the ensemble typically faster and much simpler than training a single kernel classifier.
- ▶ Deep learning has been suggested as a way to discover subtle or complicated concepts.
- ▶ Results suggest that *rich interactions* can be useful, too.
- ▶ Model is inherently three-way (but shallow).
- ▶ Still left to try: *Sparsity* (!)