How to train a mixture of 100.000.000.000.000.000 classifiers

Roland Memisevic

May 17, 2010
Log-linear models

- Task: Map $x \in \mathcal{R}^n$ to class-label $y$.
- Linear approach:
  1. Define the linear score function
     
     $$s_y(x) = w_y^T x$$
  2. Turn scores into probabilities
     
     $$p(y|x) = \frac{\exp(w_y^T x)}{\sum_{y'} \exp(w_{y'}^T x)}$$
- Training by maximizing $\sum_\alpha \log p(y^\alpha|x^\alpha)$.
- Benefits: Training convex; outputs probabilistic.
- AKA logistic regression, “maximum entropy”, . . .
A log-bilinear model

- Introduce a set of *binary latent variables* $\mathbf{h} = (h_1, \ldots, h_K)$
- A bilinear approach:
  1. Define the *bilinear* score function
     
     $$ s_y(x, \mathbf{h}) = \mathbf{h}^t W_y x $$

  2. Turn scores into probabilities
     
     $$ p(y, \mathbf{h}|x) = \frac{\exp(\mathbf{h}^t W_y x)}{\sum_{y', \mathbf{h}'} \exp(\mathbf{h}'^t W_{y'} x)} $$

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

- Training non-linear (see below); outputs still probabilistic.
A log-bilinear model

- Marginalization tractable!

\[
p(y|\mathbf{x}) = \sum_h p(y, h|\mathbf{x}) \propto \sum_h \exp(\mathbf{x}^\top W_y h) = \sum_h \exp(\sum_{ik} W_{yik} x_i h_k) = \prod_k (1 + \exp(\sum_i W_{yik} x_i))
\]

- 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

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

Derivatives

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

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

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

Performance:

<table>
<thead>
<tr>
<th>SVMRBF</th>
<th>SVMPOL</th>
<th>NNet</th>
<th>DBN3</th>
<th>SAA3</th>
<th>DBN1</th>
<th>gated softmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.15</td>
<td>2.15</td>
<td>7.16</td>
<td>2.60</td>
<td>2.41</td>
<td>4.71</td>
<td>0.564</td>
</tr>
</tbody>
</table>

- 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

\[ W_{yik} = \sum_f W_{if} W_{yf} W_{kf} \]

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

\[ W_{\text{yik}} = \sum_{f} W_{\text{if}}^{x} W_{\text{yf}}^{y} W_{\text{kf}}^{h} \]
Factorization

\[
\log p(y, h|x) \propto \sum_f \left( \sum_i x_i W^x_{if} \right) \left( \sum_k h_k W^h_{kf} \right) W^y_{yf}
\]
MNIST

<table>
<thead>
<tr>
<th>Implicit RBM mixtures</th>
<th>SVM</th>
<th>Logistic Regression</th>
<th>Gated softmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.36</td>
<td>1.40</td>
<td>7.28</td>
<td>1.74</td>
</tr>
</tbody>
</table>

Roland Memisevic
Gated softmax classification
Rectangles, factored model

<table>
<thead>
<tr>
<th>SVMRBF</th>
<th>SVMPOL</th>
<th>NNet</th>
<th>DBN3</th>
<th>SAA3</th>
<th>DBN1</th>
<th>gated softmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.15</td>
<td>2.15</td>
<td>7.16</td>
<td>2.60</td>
<td>2.41</td>
<td>4.71</td>
<td>0.826</td>
</tr>
</tbody>
</table>
Rectangles with images

<table>
<thead>
<tr>
<th>SVMRBF</th>
<th>SVMPOL</th>
<th>NNet</th>
<th>DBN3</th>
<th>SAA3</th>
<th>DBN1</th>
<th>gated softmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.04</td>
<td>24.05</td>
<td>33.20</td>
<td>22.50</td>
<td>24.05</td>
<td>23.69</td>
<td>24.81</td>
</tr>
<tr>
<td>SVMRBF</td>
<td>SVMPOL</td>
<td>NNet</td>
<td>DBN3</td>
<td>SAA3</td>
<td>DBN1</td>
<td>gated softmax</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>---------------</td>
</tr>
<tr>
<td>11.11</td>
<td>15.42</td>
<td>18.11</td>
<td>10.30</td>
<td>10.30</td>
<td>14.69</td>
<td>12.75</td>
</tr>
</tbody>
</table>
Rotated digits

Roland Memisevic

Gated softmax classification
## Convex vs Nonconvex

<table>
<thead>
<tr>
<th></th>
<th>SVMRBF</th>
<th>SVMPOL</th>
<th>NNet</th>
<th>DBN3</th>
<th>SAA3</th>
<th>DBN1</th>
<th>gated softmax</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>19.13</td>
<td>19.82</td>
<td>32.25</td>
<td>18.63</td>
<td>18.41</td>
<td>19.92</td>
<td>17.08</td>
</tr>
</tbody>
</table>

(unfactored model: 21.026)
### Gated softmax classification

<table>
<thead>
<tr>
<th>SVMRBF</th>
<th>SVMPOL</th>
<th>NNet</th>
<th>DBN3</th>
<th>SAA3</th>
<th>DBN1</th>
<th>gated softmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.58</td>
<td>16.62</td>
<td>20.04</td>
<td>06.73</td>
<td>11.28</td>
<td>09.80</td>
<td>10.48</td>
</tr>
</tbody>
</table>
### Digits, background image

<table>
<thead>
<tr>
<th>Method</th>
<th>SVMRBF</th>
<th>SVMPOL</th>
<th>NNet</th>
<th>DBN3</th>
<th>SAA3</th>
<th>DBN1</th>
<th>Gated softmax</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>22.61</td>
<td>24.01</td>
<td>27.41</td>
<td>16.31</td>
<td>23.00</td>
<td>16.15</td>
<td>23.65</td>
</tr>
</tbody>
</table>
Digits, rotated + background

<table>
<thead>
<tr>
<th>SVM RBF</th>
<th>SVM POL</th>
<th>NNNet</th>
<th>DBN3</th>
<th>SAA3</th>
<th>DBN1</th>
<th>Gated softmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>55.18</td>
<td>56.41</td>
<td>62.16</td>
<td>47.39</td>
<td>51.93</td>
<td>52.21</td>
<td>55.82</td>
</tr>
</tbody>
</table>
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 \(\infty\)-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 (!)