Learning the Parameters of Markov Networks

srihari@buffalo.edu

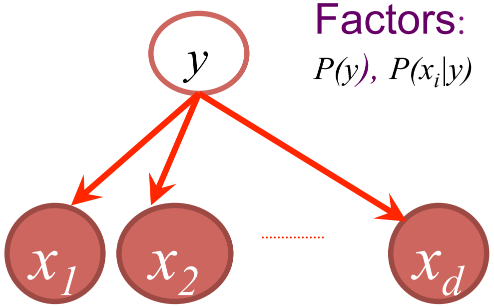
Topics

- BN parameter learning vs MN parameter learning
- Learning for Energy-based models
- Learning for RBMs
- Learning for Deep Belief Networks

Determining Parameters: BN vs. MN

Classification Problem: Features $\mathrm{x} = \{x_1, .. x_d\}$ and two-class label y

BN: Naïve Bayes (Generative): CPD parameters



Factors:
$P(y),\ P(x_i|y)$

Joint Probability:

$$P(y,\mathrm{x}) = P(y)\prod_{i=1}^{d} P(x_i \mid y)$$

From joint get required conditional $P(y|\mathrm{x})$

If each x_i is discrete with k values
  independently estimate $d(k\text{-}1)$ parameters
But independence is false
For sparse data generative is better
C-class problem: $d(k\text{-}1)(C\text{-}1)$ parameters

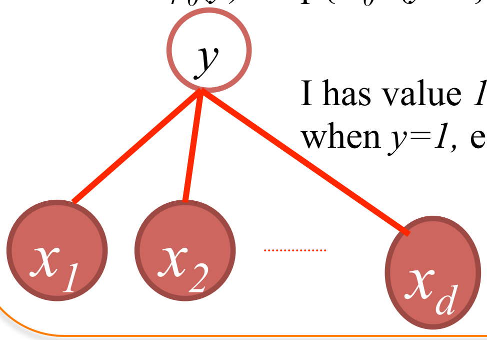
MN: Logistic Regression (Discrim): parameters $\mathrm{w_i}$

Factors (log-linear): $D_i=\{x_i, y\}$

$f_i(D_i)=x_i I(y)$

$\phi_i(x_i,y)=\exp\{w_i x_i\, \mathrm{I}\{y=1\}\},$

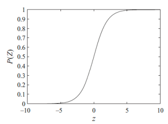
$\phi_0(y)=\exp\{w_0\, \mathrm{I}\{y=1\}\}$



I has value 1 when $y=1$, else 0

Conditional Unnormalized $\tilde{P}(y=1 \mid \mathrm{x}) = \exp\left\{w_0 + \sum_{i=1}^{d} w_i x_i\right\}$ $\tilde{P}(y=0 \mid \mathrm{x}) = \exp\{0\} = 1$

Normalized $P(y=1 \mid \mathrm{x}) = sigmoid\left\{w_0 + \sum_{i=1}^{d} w_i x_i\right\}$ where $sigmoid(z) = \dfrac{e^z}{1+e^z} = \dfrac{1}{1+e^{-z}}$



Logistic Regression

Z has term 1 because $\tilde{P}(y=0\ |\mathrm{x})=1$

Jointly optimize d parameters
High dimensional estimation
but correlations accounted for
Can use much richer features:
  Edges, image patches sharing same pixels

C-class

$$p(y_c \mid \mathrm{x}) = \frac{\exp(\mathrm{w}_c^T \mathrm{x})}{\sum_j^C \exp(\mathrm{w}_j^T \mathrm{x})}$$

C x d parameters

Energy-based Models (EBMs)

- Boltzmann distribution is an energy model
  - Probability distribution: associates a scalar energy with each configuration of its variables
- Energy-based probability distribution

$$p(x) = \frac{1}{Z}\exp(-E(x))$$

  - Where Z is the partition function $Z = \sum_x \exp(-E(x))$
- Learning corresponds to modifying energy function so its shape has desirable properties
  - E.g., plausible configurations have low energy

Learning EBM parameters

- To determine parameters of

$$\boxed{p(x) = \frac{1}{Z}\exp(-E(x))}$$

- Perform stochastic gradient-descent on negative log-likelihood
- Log-likelihood $\mathcal{L}(\theta, \mathcal{D}) = \frac{1}{N} \sum_{x^{(i)} \in \mathcal{D}} \log\ p(x^{(i)})$
- Loss function $\ell(\theta, \mathcal{D}) = -\mathcal{L}(\theta, \mathcal{D})$
  - Gradient is $-\frac{\partial \log p(x^{(i)})}{\partial \theta}$ where θ are parameters

$$\theta^{(\tau+1)} = \theta^{(\tau)} - \eta \nabla \ell$$

EBMs with hidden units

- Want to include non-observed variables to increase expressive power of model

$$P(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x,h)}}{Z}.$$

- Introducing free-energy $\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)}$

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \text{ with } Z = \sum_x e^{-\mathcal{F}(x)}.$$

- Data negative log-likelihood gradient

$$-\frac{\partial \log p(x)}{\partial \theta} = \frac{\partial \mathcal{F}(x)}{\partial \theta} - \sum_{\tilde{x}} p(\tilde{x}) \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}.$$

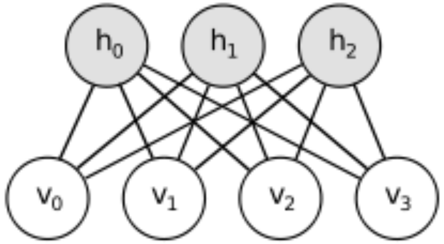
First term increases probability of training data. Second term decreases probability of samples generated by model

- Sampling version (with samples from P)

$$-\frac{\partial \log p(x)}{\partial \theta} \approx \frac{\partial \mathcal{F}(x)}{\partial \theta} - \frac{1}{|\mathcal{N}|} \sum_{\tilde{x} \in \mathcal{N}} \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}.$$

Learning with RBMs

- Energy function



$E(v,h) = -b'v - c'h - h'Wv$

where Wis weight matrix connecting hidden and visible units

$v = [v_0, v_1, ..], h = [h_0, h_1, ..]$, with offset vectors b, c

- Defining free energy as

$$\mathcal{F}(v) = -b'v - \sum_i \log \sum_{h_i} e^{h_i (c_i + W_i v)}.$$

- Due to structure of RBM

$$p(h|v) = \prod_i p(h_i|v)$$
$$p(v|h) = \prod_j p(v_j|h).$$

RBM with binary units

- Using $v_j,\ h_i \in \{0,1\}$

$$P(h_i = 1|v) = sigm(c_i + W_i v)$$
$$P(v_j = 1|h) = sigm(b_j + W'_j h)$$

- Free energy simplifies to

$$\mathcal{F}(v) = -b'v - \sum_i \log(1 + e^{(c_i + W_i v)}).$$

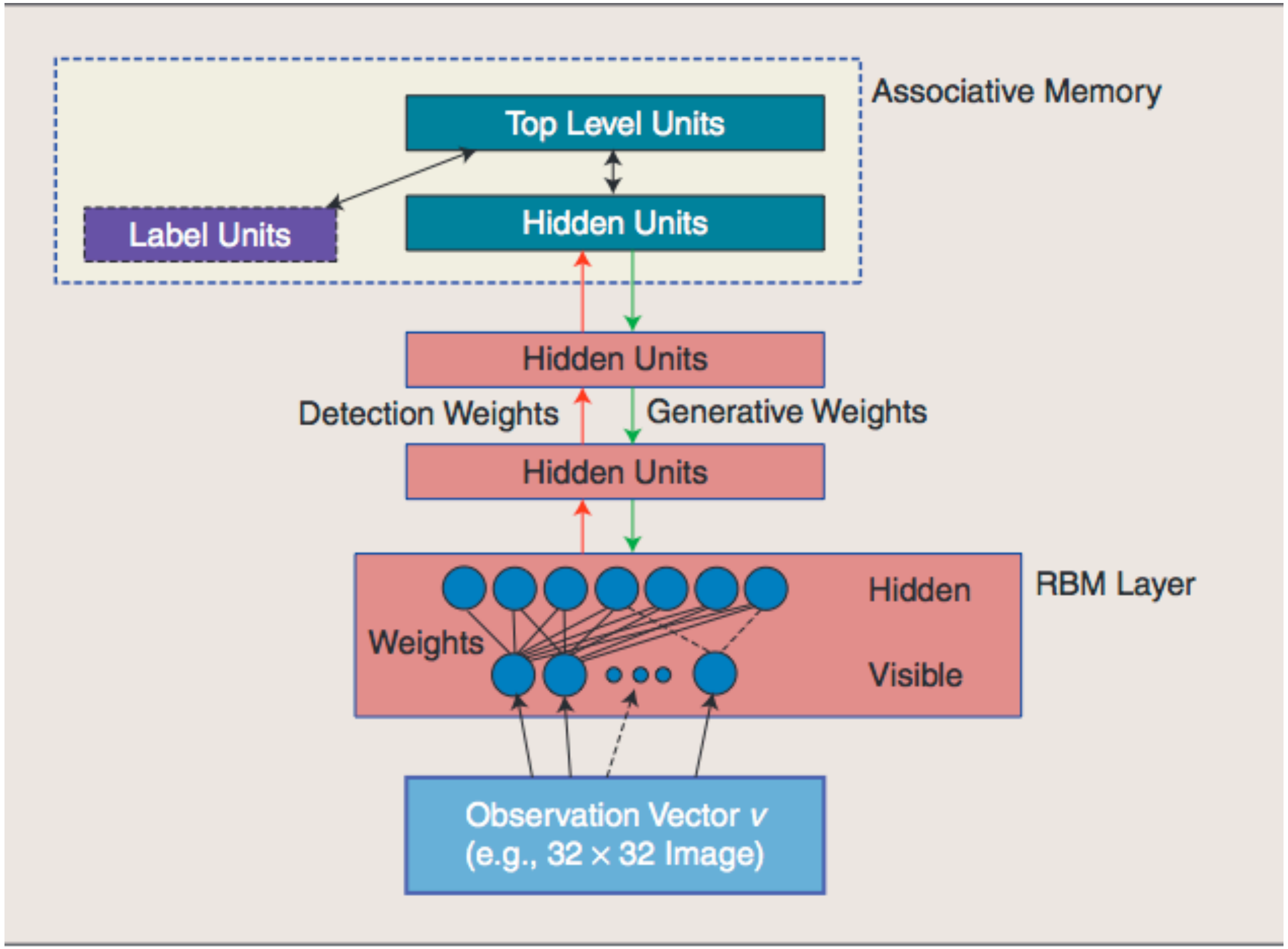
- Update equations

$$-\frac{\partial \log p(v)}{\partial W_{ij}} = E_v[p(h_i|v) \cdot v_j] - v_j^{(i)} \cdot sigm(W_i \cdot v^{(i)} + c_i)$$
$$-\frac{\partial \log p(v)}{\partial c_i} = E_v[p(h_i|v)] - sigm(W_i \cdot v^{(i)})$$
$$-\frac{\partial \log p(v)}{\partial b_j} = E_v[p(v_j|h)] - v_j^{(i)}$$

Training RBMs

- Contrastive Divergence
- A method to overcome exponential complexity in dealing with the partition function

Deep Belief Network Framework



Training DBNs

- Let X be a matrix of input feature vectors

1. Train an RBM on X to obtain weight matrix W
   - Between lower two layers (input and hidden)
2. Transform X by RBM to produce new data X'
   - by sampling or by computing mean activation of hidden units
3. Repeat procedure with $X \leftarrow X'$ for next layer pair
   - Until top two layers of network are reached (output and hidden)